

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERÍAS FÍSICAS Y FORMALES

ESCUELA PROFESIONAL DE INGENIERÍA MECÁNICA, MECÁNICA
ELÉCTRICA Y MECATRÓNICA



**“DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE BRAZO ROBÓTICO
(4GL) TELEOPERADO PARA MANIPULACIÓN DE SUSTANCIAS TÓXICAS
ASISTIDO CON VISIÓN ARTIFICIAL Y REDES NEURONALES PARA
LABORATORIOS FARMACÉUTICOS”**

TESIS PRESENTADO POR:

Bsc. Breslly Bravo Romero

Bsc. Jaime Enrique Villegas Medina

Para optar por el Título Profesional de:

Ingeniero Mecatrónico

Asesores:

Ing. Sergio Mestas Ramos

Ing. Juan Carlos Cuadros Machuca

AREQUIPA – PERÚ

AÑO - 2017

ÍNDICE

CAPÍTULO I: FUNDAMENTACIÓN.....	1
1.1. Identificación del problema:	1
1.1.1. Descripción del Problema:.....	1
1.2. Antecedentes:	2
1.3. Objetivos:	3
1.3.1. Objetivo General:.....	3
1.3.2. Objetivos Específicos:.....	3
1.4. Hipótesis:	4
1.5. Justificación:.....	4
1.6. Alcances:	4
1.7. Metodología de la investigación:	5
CAPÍTULO II: MARCO TEÓRICO	6
2.1. Laboratorios Farmacéuticos:.....	6
2.1.1. Instrumentos en Laboratorios Farmacéuticos:.....	6
2.1.2. Manipulación de Sustancias Peligrosas:.....	6
2.2. Robótica:	9
2.2.1. Definición de Robot:	10
2.2.2. Clasificación de los Robots:	10
2.2.3. Morfología del Robot:	16
2.2.4. Herramientas Matemáticas para la localización espacial:	23
2.2.5. Cinemática del Robot:	28
2.2.6. Dinámica del Robot:.....	37
2.3. Herramientas Software a utilizar:.....	40
2.3.1. Solidwork:.....	40
2.3.2. Matlab:	41
2.3.3. Repetier Host:.....	42
2.4. Teleoperación:.....	43
2.4.1. Elementos y Arquitectura de un Sistema de Teleoperación:	44
2.4.2. Teleoperación frente a la robótica:	49
2.5. Visión Artificial:.....	51
2.5.1. Configuración de un Sistema de Visión Artificial:	52
2.5.2. Disciplinas relacionadas con la Visión Artificial:	53
2.5.3. Aplicaciones:.....	53

2.6.	Redes Neuronales:	54
2.6.1.	Modelo de una Neurona Artificial:	55
2.6.2.	Red Neuronal Artificial:	56
2.6.3.	Arquitectura de las RNA:	57
2.6.4.	Multiplicando la Potencia, Redes Multicapa:	58
2.6.5.	Aprendizaje de las RNA:	58
2.6.6.	Red Back-Propagation:	59
2.6.7.	Cinemática Inversa de un robot Manipulador:	59
2.7.	Torque Computado:	60
CAPÍTULO III: DISEÑO		64
3.1.	Diseño Mecánico:	66
3.1.1.	Estructura Mecánica:	66
3.1.2.	Diseño Preliminar Manual:	67
3.1.3.	Análisis Cinemático:	71
3.1.4.	Análisis Dinámico:	92
3.1.5.	Materiales:	106
3.1.6.	Elección de Actuadores y Diseño Virtual en Solidwork:	107
3.1.7.	Espacio de Trabajo:	134
3.2.	Diseño Electrónico:	135
3.2.1.	Elección de Componentes Electrónicos:	136
3.2.2.	Diseño Preliminar del Circuito:	146
3.3.	Diseño y Construcción de Maqueta:	148
CAPÍTULO IV: IMPLEMENTACIÓN		151
4.1.	Implementación Mecánica:	151
4.1.1.	Herramientas usadas para la implementación mecánica del brazo robótico:	151
4.1.2.	Modelo Virtual del Brazo Robótico Armado:	153
4.1.3.	Descripción de los Eslabones:	154
4.1.4.	Impresión 3D:	160
4.1.5.	Pruebas individuales:	165
4.1.6.	Orden de Montaje:	176
4.1.7.	Montaje y Ajustes:	188
4.1.8.	Prueba de Funcionamiento del Manipulador:	192
4.1.9.	Correcciones y/o Modificaciones al Ensamble Final:	195
4.1.10.	Nuevo Espacio de Trabajo:	204
4.1.11.	Pruebas Finales:	205
4.2.	Implementación Electrónica:	207

4.2.1.	Herramientas usadas para la implementación mecánica del brazo robótico: ..	207
4.2.2.	Instalación del Sistema Operativo del Procesador:	208
4.2.3.	Pruebas Individuales:	208
4.2.4.	Correcciones y/o Modificaciones:	211
4.2.5.	Armado Preliminar:.....	213
4.2.6.	Prueba de Funcionamiento del Circuito:.....	213
4.2.7.	Diseño e impresión de Placa:.....	215
CAPÍTULO V: CONTROL E INTERFAZ GRAFICA DE USUARIO		217
5.1.	SimMechanics:	217
5.2.	Modelamiento Kinect:	218
5.3.	Control Cinemático: Redes Neuronales:.....	223
5.3.1.	Ingreso de Datos:.....	224
5.3.2.	Creación de la Red:	225
5.3.3.	Entrenamiento de Red:	226
5.3.4.	Resultados del Entrenamiento:	227
5.4.	Comunicación entre Matlab y Raspberry Pi:.....	232
5.4.1.	Computadora:.....	233
5.4.2.	Raspberry Pi:.....	234
5.5.	Variables de Estado:	235
5.6.	Control Dinámico: Torque Computado:.....	237
5.7.	Programación en Python	241
5.8.	Prueba de Funcionamiento Virtual:.....	241
5.9.	Interfaz Gráfica de Usuario:	245
5.9.1.	Sintonización de Ganancias para el Control:.....	246
5.10.	Control Manual:.....	248
5.11.	Prueba y Simulaciones:	249
5.12.	Trayectorias:.....	250
5.13.	Dreamy:.....	256
CAPÍTULO VI: PRUEBAS Y RESULTADOS		257
CONCLUSIONES:.....		261
RECOMENDACIONES:.....		262
BIBLIOGRAFIA:		263
ANEXOS:		266

Índice de Ilustraciones

Figura 2.1: Clasificación de Robots según su Configuración.	14
Figura 2.2: Clasificación de Robots según el tipo de Control.	16
Figura 2.3: Componentes estructurales de un robot industrial.	16
Figura 2.4: Grados de Libertad de un brazo robótico.	17
Figura 2.5: a) Cadena Cinemática Cerrada. b) Cadena Cinemática Abierta.	17
Figura 2.6: Punto terminal de un manipulador.	18
Figura 2.7: Espacio de trabajo del robot KUKA KR 6-2.	18
Figura 2.8: Configuraciones para el desplazamiento de los dedos de una pinza. De derecha a izquierda Angulares y Paralelos.....	23
Figura 2.9: Representación de un vector en coordenadas cartesianas en 2 y 3 dimensiones....	24
Figura 2.10: Diagrama de Relación entre cinemática directa e inversa.....	29
Figura 2.11: Parámetros D-H para un eslabón giratorio.	30
Figura 2.12: Matriz jacobiana directa e inversa.....	34
Figura 2.13: Matriz jacobiana geométrica directa e inversa.	35
Figura 2.14: Elementos Básicos de un sistema de teleoperación.....	45
Figura 2.15: Niveles de modo de control Remoto.	47
Figura 2.16: Arquitectura genérica de un sistema de Teleoperación.	49
Figura 2.17: Evolución de la Robótica y la Teleoperación en cuanto a flexibilidad y autonomía.	51
Figura 2.18: Modelo de una red neuronal	55
Figura 2.19: Función de activación	56
Figura 2.18: Modelo de una red neuronal	56
Figura 2.21: Arquitectura de RNA.....	57
Figura 3.1: Estructura básica del brazo robótico	67
Figura 3.2: Plano de la base.....	68
Figura 3.3: Plano del Eslabón 1.....	68
Figura 3.4: Plano del Eslabón 2.....	69
Figura 3.5: Plano del Eslabón 3.....	69
Figura 3.6: Base 1 solidworks	70
Figura 3.7: Base 2 solidworks	70
Figura 3.8: Eslabón 1 solidworks.....	70
Figura 3.9: Eslabón Solidwork.....	71
Figura 3.10: Eslabón 3 Solidwork.....	71
Figura 3.11: Representación esquemática del brazo robótico 4GDL.....	74
Figura 3.12: Primeros 3GDL del brazo robótico.....	76
Figura 3.13: Diagrama para la obtención de la variable articular q_2	77
Figura 3.14: Diagrama para la obtención de la variable articular q_3	78
Figura 3.15: Configuraciones aleatorias aplicando el Robotics Toolbox para comprobar el análisis cinemático.....	92
Figura 3.16: Representación del 1er eslabón para análisis dinámico.....	94
Figura 3.17: Representación del 2do eslabón para análisis dinámico.....	95
Figura 3.18: Representación del 3er eslabón para análisis dinámico.....	97
Figura 3.19: Representación del 4to eslabón para análisis dinámico.....	99
Figura 3.20: Torque(kg-cm) vs Velocidad(RPM) de Servomotores Dynamixel.....	108
Figura 3.21: Bosquejo preliminar de la Garra.....	109
Figura 3.22: Fuerzas aplicadas a la garra.....	110

Figura 3.23: Cálculos para malla	110
Figura 3.24: Gráfico de Tensiones.....	111
Figura 3.25: Gráfica de desplazamientos	111
Figura 3.26: Gráfica de Factor de Seguridad	112
Figura 3.27: Definición de suceso	112
Figura 3.28: Definición de material.....	113
Figura 3.29: Gráfica de Vida Útil	113
Figura 3.30: Segundo Bosquejo	114
Figura 3.31: Gráficas de Ensayo Estático	114
Figura 3.32: Gráfica de Ensayo de Fatiga	115
Figura 3.33: Tercer Bosquejo	115
Figura 3.34: Gráficas de Ensayo Estático	116
Figura 3.35: Gráfica de Ensayo de Fatiga	116
Figura 3.36: Bosquejo Final	117
Figura 3.37: Gráficas de Ensayo Estático	117
Figura 3.38: Gráfica de Ensayo de Fatiga	118
Figura 3.39: Izquierda: bosquejo inicial de portagarra; Derecha: bosquejo final de portagarra	119
Figura 3.40: Ensamble de la garra.....	119
Figura 3.41: Momento de Inercia y masa obtenidos del Software Solidwork	121
Figura 3.42: Longitud al centro de masa y Longitud total del portagarra obtenidas del software Solidwork.....	122
Figura 3.43: Aplicando programa "TRAYECTORIACIRC.m"	122
Figura 3.44: Torque Máximo Garra.....	123
Figura 3.45: Izquierda: bosquejo inicial del eslabón 2; Derecha: bosquejo final del eslabón 2	123
Figura 3.46: Momento de Inercia y masa obtenidos del Software Solidwork	124
Figura 3.47: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software Solidwork	124
Figura 3.48: Torque Máximo Eslabón 2.....	124
Figura 3.49: Izquierda: bosquejo inicial del eslabón 1; Derecha: bosquejo final del eslabón 1	125
Figura 3.50: Momento de Inercia y masa obtenidos del Software Solidwork	125
Figura 3.51: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software Solidwork	126
Figura 3.52: Torque Máximo Eslabón 1.....	126
Figura 3.53: Fuerzas aplicadas al eslabón 2.....	127
Figura 3.54: Tamaño mínimo de malla.....	127
Figura 3.55: Gráfica de Tensiones.....	128
Figura 3.56: Gráfica de Desplazamientos.....	128
Figura 3.57: Gráfica de Factor de Seguridad	128
Figura 3.58: Definición de suceso	129
Figura 3.59: Definición de material.....	129
Figura 3.60: Gráfica de Vida Total.....	130
Figura 3.61: Izquierda: bosquejo inicial de la base superior; Derecha: bosquejo final de la base superior	130
Figura 3.62: Momento de Inercia y masa obtenidos del Software Solidwork	131
Figura 3.63: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software Solidwork	131
Figura 3.64: Torque Máximo Base	132

Figura 3.65: Izquierda: bosquejo inicial de la base inferior; Derecha: bosquejo final de la base inferior.....	132
Figura 3.66: Representación del manipulador robótico de acuerdo a su configuración	133
Figura 3.67: Evolución de cada uno de los torques en función al tiempo	133
Figura 3.68: Tendencias de los momentos de torsión en el tiempo.....	134
Figura 3.69: Vista lateral del espacio de trabajo	135
Figura 3.70: Vista superior del espacio de trabajo	135
Figura 3.71: Servomotores Dynamixel.....	136
Figura 3.72: Servomotor Tower Pro SG5010	136
Figura 3.73: Fuente de Alimentación Regulada.....	137
Figura 3.74: Regulador de Tensión LM7805.....	138
Figura 3.75: Circuito de Conexión LM7805	138
Figura 3.76: Procesador Digital de Señales	139
Figura 3.77: Raspberry Pi.....	139
Figura 3.78: Raspberry Pi2 Model B.....	140
Figura 3.79: Circuito de conexión 74ls241 y MAX485.....	140
Figura 3.80: Pantalla LCD LG Display LP101WH4-SLAF	141
Figura 3.81: Conversor VSLC Display con sus partes indicadas	141
Figura 3.82: Elementos para conexión con la pantalla LCD	141
Figura 3.83: Kinect XBOX 360	142
Figura 3.84: Cámara HD Pro Webcam c920 - Logitech	143
Figura 3.85: Sensor de fuerza MF01	144
Figura 3.86: Gráfica de Resistencia ($K\Omega$) vs Fuerza(g).....	144
Figura 3.87: Medidas del sensor de fuerza MF01.....	144
Figura 3.88: ADC MCP3202	145
Figura 3.89: Regulador de tensión Lm7805.....	145
Figura 3.90: Supresor de picos de 6 salidas.....	146
Figura 3.91: Estabilizador de 8 salidas	146
Figura 3.92: Diseño preliminar del circuito final.....	147
Figura 3.93: Diseño de la maqueta	148
Figura 3.94: a) Parte superior hecha de melamina y b) Estructura de ensamblaje de la maqueta	149
Figura 3.95: Patas de aluminio	149
Figura 3.96: Topes triangulares para sujeción.....	149
Figura 3.97: Maqueta final	150
Figura 4.1: Brazo Robótico en subensambles.....	153
Figura 4.2: Subensamble 1: Base 1	154
Figura 4.3: Motor MX-28.....	155
Figura 4.4: Rodamiento SKF 51114	155
Figura 4.5: Subensamble 2: Base 2	156
Figura 4.6: Subensamble 3: Eslabón 1	157
Figura 4.7: Subensamble 4: Eslabón 2	158
Figura 4.8: Subensamble 5: Garra.....	159
Figura 4.9: Motor AX-12.....	159
Figura 4.10: Parámetros de configuración de la impresora	160
Figura 4.11: Parámetros de configuración del extrusor.....	161
Figura 4.12: Visualización del elemento a imprimir.....	161
Figura 4.13: Pestaña Objetos.....	162

Figura 4.14: Pestaña Slicer	162
Figura 4.15: Pestaña Control Manual.....	163
Figura 4.16: Impresión 3D de Portagarra	164
Figura 4.17: Impresión 3D de Engranaje	164
Figura 4.18: Conexión del motor TowerPRO SG5010 para puesta en funcionamiento.....	165
Figura 4.19: Esquemático eléctrico de conexión del motor TowerPRO SG5010	166
Figura 4.20: Prueba de funcionamiento de motor TowerPRO SG5010	166
Figura 4.21: Conexión de actuadores Dynamixel AX-12 para puesta en funcionamiento.....	167
Figura 4.22: Esquemático eléctrico de conexión de motores Dynamixel AX-12	167
Figura 4.23: Prueba de funcionamiento de actuadores Dynamixel AX-12	168
Figura 4.24: Conexión de actuadores Dynamixel RX-24F para puesta en funcionamiento	168
Figura 4.25: Esquemático eléctrico de conexión de motores Dynamixel RX-24F	169
Figura 4.26: Prueba de funcionamiento de actuadores Dynamixel RX-24F.....	169
Figura 4.27: Conexión del motor Dynamixel MX-28T para puesta en funcionamiento.....	170
Figura 4.28: Esquemático eléctrico de conexión del motor Dynamixel MX-28T.....	170
Figura 4.29: Prueba de funcionamiento de motor Dynamixel MX-28T	171
Figura 4.30: a) Disposición de piezas, b) Ajuste de Servomotor en Base 1	177
Figura 4.31: Montaje de Rodamiento en Base 1	177
Figura 4.32: Instalación de Círculo de Unión MX28.....	178
Figura 4.33: Montaje de Motores RX-24f.....	178
Figura 4.34: Montaje de Ovnitapa	179
Figura 4.35: Ensamble de Soporte Motor	179
Figura 4.36: Instalación de Motor AX12.....	180
Figura 4.37: Montaje de Círculos de Unión RX24F	180
Figura 4.38: Montaje de Tapas laterales	181
Figura 4.39: Instalación del Servomotor TowerPro SG-5010	181
Figura 4.40: Montaje de Soporte Motor	182
Figura 4.41: Montaje de Tapas laterales	182
Figura 4.42: Ensamble Base1 y Base 2	183
Figura 4.43: Montaje de Eslabón 1 a la Base del robot.....	183
Figura 4.44: Montaje de Eslabón 2 al Brazo Robótico	184
Figura 4.45: Ensamble de Círculo de Unión TowerPro.....	184
Figura 4.46: Montaje de Portagarra.....	185
Figura 4.47: Instalación de Servomotor AX-12 de la Garra	185
Figura 4.48: Ensamble de Engranaje Maestro	186
Figura 4.49: Montajes de Engranajes en el Portagarra.....	186
Figura 4.50: Sujeción de Caja Motor	187
Figura 4.51: Montaje de Palitos.....	187
Figura 4.52: Montaje de Dedos de la Garra	187
Figura 4.53: Brazo robótico ensamblado	188
Figura 4.54: Subensamble Base 1	188
Figura 4.55: Subensamble Base 2	189
Figura 4.56: Subensamble Eslabón 2	189
Figura 4.57: Subensamble Garra.....	189
Figura 4.58: a) Montaje de servomotores RX-24f en Eslabón 1 b) Montaje del Eslabón 1 en la base del robot.....	190
Figura 4.59: Ajuste de Piezas	190
Figura 4.60: Brazo Robótico 4GDL	191

Figura 4.61: Conexión de los servomotores en simultáneo para puesta en funcionamiento...	192
Figura 4.62: Esquemático eléctrico de conexión de los servomotores en simultáneo.....	193
Figura 4.63: Prueba de funcionamiento del Manipulador Robótico	194
Figura 4.64: Prueba de funcionamiento del Efecto Final.....	194
Figura 4.65: Soporte para estabilidad del Efecto Final.	195
Figura 4.66: Soporte para estabilidad ubicado entre el eslabón 2 y la garra.....	195
Figura 4.67: Instalación de bujes en los puntos de unión del Efecto Final.	196
Figura 4.68: a) Diseño anterior de Garra y b) Diseño final de garra	197
Figura 4.69: a) Diseño anterior de Soporte y b) Diseño final de Soporte	198
Figura 4.70: a) Diseño anterior de engranaje y b) Diseño final de Engranaje	198
Figura 4.71: Tensiones de Prueba aplicadas a la garra con bujes.....	199
Figura 4.72: Resultado de Tensión.....	199
Figura 4.73: Resultado de Deformaciones	200
Figura 4.74: Resultado de Factor de Seguridad	200
Figura 4.75: Suceso para ensayo de Fatiga	201
Figura 4.76: Curvas S – N de cada material	201
Figura 4.77: Gráfica de vida total.....	201
Figura 4.78: Montaje de Efecto Final.....	203
Figura 4.79: Ensamble final del brazo robótico	203
Figura 4.80: Vista lateral del espacio de Trabajo	204
Figura 4.81: Vista superior del espacio de Trabajo	204
Figura 4.82: Secuencia de desplazamiento del brazo robótico.	206
Figura 4.83: Prueba de funcionamiento de la Raspberry y la pantalla LCD	209
Figura 4.84: Instalación de la Librería “Kinect for Windows Sensor Support Package”	209
Figura 4.85: Imagen RGB y Skeleton captada por el sensor Kinect	210
Figura 4.86: Prueba de Funcionamiento de Logitech C920.....	211
Figura 4.87: HUB	211
Figura 4.88: Prueba de funcionamiento Raspberry Pi y pantalla LCD	212
Figura 4.89: Instalación OPEN CV	212
Figura 4.90: Armado preliminar del circuito	213
Figura 4.91: Conexión de elementos electrónicos para prueba de funcionamiento.....	213
Figura 4.92: Programación de los GPIO de Control de ambas librerías	214
Figura 4.93: Prueba de Funcionamiento del Robot	215
Figura 4.94: Diseño del circuito en Eagle	215
Figura 4.95: Diseño del circuito impreso y pistas en placa electrónica de cobre.....	216
Figura 4.96: Conexión y Disposición de Componentes Electrónicos.....	216
Figura 5.1: SimMechanics aplicado al manipulador robótico en simulink.....	217
Figura 5.2: Diagrama completo del Robot simulado	217
Figura 5.3: Subsistema ROBOT	218
Figura 5.4: Bloque NID IMAQ y ventana de configuración.....	218
Figura 5.5: Bloque NID IMAGE, VIDEO VIEWER y ventana de configuración	219
Figura 5.6: Bloque NID Skeleton y ventana de configuración	219
Figura 5.7: Bloque Hand Viewer	220
Figura 5.8: Funciones para la adquisición de Coordenadas articulares	220
Figura 5.9: Funciones Transformar	221
Figura 5.10: Bloques de posición Inicial	222
Figura 5.11: Subsistema Kinect.....	222
Figura 5.12: Prueba de Funcionamiento	223

Figura 5.13: Encapsulado	223
Figura 5.14: Neural Network ToolBox.....	224
Figura 5.15: Asignación de datos.....	224
Figura 5.16: Asignación de datos.....	225
Figura 5.17: Creación de la Red Neuronal.....	226
Figura 5.18: Rango de valores	226
Figura 5.19: Parámetros de entrenamiento.....	227
Figura 5.20: Proceso de entrenamiento.....	227
Figura 5.21: Performance: Error vs Épocas	228
Figura 5.22: Estado de entrenamiento	228
Figura 5.23: Regresión.....	229
Figura 5.24: Uso del comando “Gensim”	231
Figura 5.25: Programa de prueba de las Redes Neuronales	231
Figura 5.26: Subsistema RN q1, q2, q3, q4.....	232
Figura 5.27: Librería Raspberry PI.....	232
Figura 5.28: Comunicación entre la PC y Raspberry	233
Figura 5.29: Bloques de envío y recepción de datos	233
Figura 5.30: Parámetros de configuración del bloque UDP Send.....	233
Figura 5.31: Parámetros de configuración del Bloque UDP Receive	234
Figura 5.32: Programación en Python de envío de datos hacia la PC.....	235
Figura 5.33: Programación de datos en Python para la recepción de datos.....	235
Figura 5.34: Programación en Python de la Lógica Combinacional.....	237
Figura 5.35: Definición de entradas al bloque de Control	237
Figura 5.36: Desarrollo interno del Bloque de Control	238
Figura 5.37: Bloque PID y estructura interna	238
Figura 5.38: Bloque Dinámica Inversa y estructura interna	239
Figura 5.39: Zona de Realimentación No - Lineal	240
Figura 5.40: Bloques de Definición de Límites.....	240
Figura 5.41: Subsistema Control.....	241
Figura 5.42: Control Virtual	242
Figura 5.43: Parámetros de prueba	242
Figura 5.44: Gráfica de Posición articular deseada vs obtenida.....	243
Figura 5.45: Simulación Simmechanics	244
Figura 5.46 : Gráfica de Posición articular deseada vs obtenida	244
Figura 5.47: Sistema de Control Real	245
Figura 5.48: Ventana de Menú Principal.....	245
Figura 5.49: Ventana de Sintonización de ganancias.....	246
Figura 5.50: Ventana Sintonización_de_ganancias.slx	246
Figura 5.51: Ventana de SimMechanics	247
Figura 5.52: Scope de posición deseada y obtenida	247
Figura 5.53: Ventana de Control Manual	248
Figura 5.54: Ventana Control_Manual.slx.....	248
Figura 5.55: Ventana pruebas_simulaciones aplicando cinemática directa	249
Figura 5.56: Ventana pruebas_simulaciones aplicando cinemática inversa mediante Cálculos Matemáticos.....	250
Figura 5.57: Ventana pruebas_simulaciones aplicando cinemática inversa.....	250
Figura 5.58: Ventana Trayectorias.....	251
Figura 5.59: Sección Trayectoria punto a punto.....	251

Figura 5.60: Programa en Simulink “trayectoria_punto_punto.slx”	252
Figura 5.61: Prueba de trayectoria punto a punto	252
Figura 5.62: Prueba de trayectoria punto a punto observada en SimMechanics	253
Figura 5.63: Gráfica de la posición articular (izquierda) y la posición cartesiana (derecha)	253
Figura 5.64: Sección Trayectoria lineal	253
Figura 5.65: Subsistema Trayectoria	254
Figura 5.66: Subsistema “Trayectoria” internamente	254
Figura 5.67: Programa en simulink “trayectoria_lineal.slx”	255
Figura 5.68: Prueba de trayectoria lineal	255
Figura 5.69: Gráfica de la posición articular (izquierda) y la posición cartesiana (derecha)	256
Figura 5.70: Botón “Dreamy”	256
Figura 5.71: Ventana del bloque Scada llamado por el botón Dreamy	256
Figura 6.1: Conexión a la red eléctrica	257
Figura 6.2: Comprobación de conectividad del Sensor Kinect	257
Figura 6.3: Encendido de todos los componentes electrónicos	258
Figura 6.4: Programa “Dreamy” corriendo en Python	258
Figura 6.5: Posición Inicial o Home	259
Figura 6.6: Proceso de traslado de sustancia tóxica	260

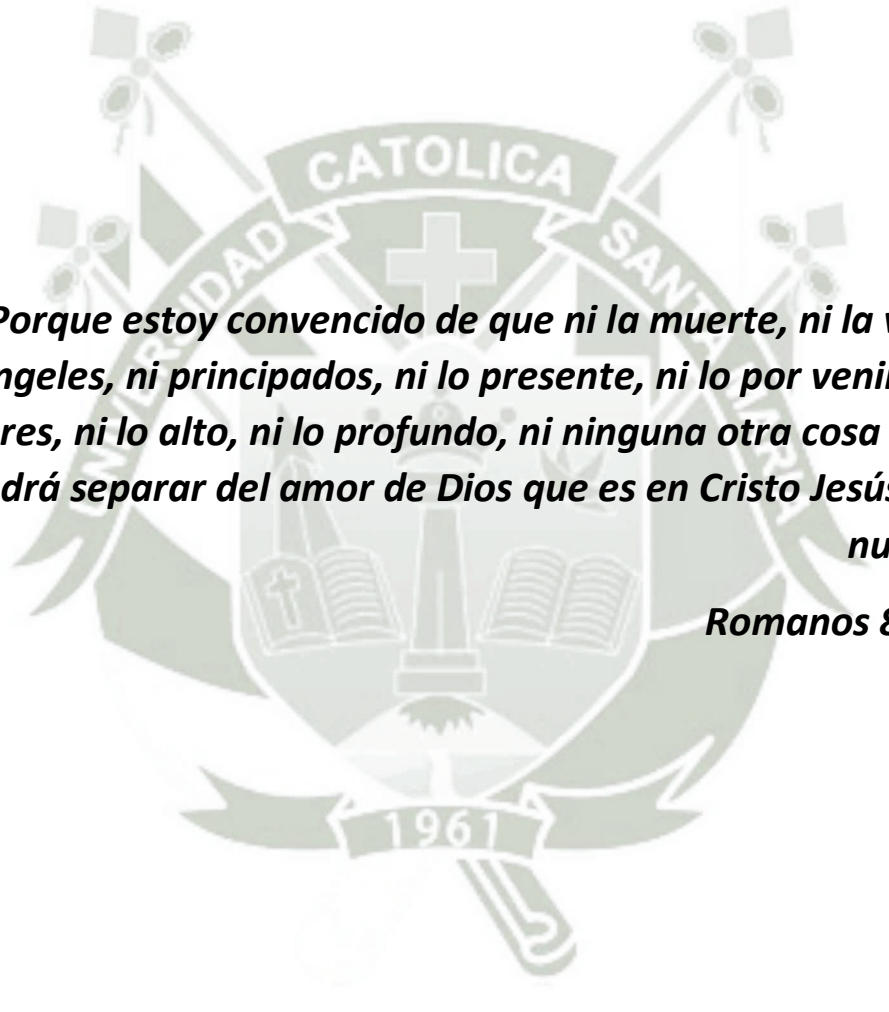


Índice de Tablas

Tabla 2.1: Historia de la Robótica	9
Tabla 2.2: Clasificación de los Robots según la Generación	11
Tabla 2.3: Clasificación de las aplicaciones de los Robots industriales manipuladores, según IFR.	12
Tabla 2.4: Clasificación de los robots de servicio por Áreas de aplicación, según IFR.	12
Tabla 2.5: Sistemas de transmisión para robots.....	20
Tabla 2.6: Características de distintos tipos de actuadores para robots.	20
Tabla 2.7: Sistemas de Sujeción para Robots.	23
Tabla 2.8: Aplicaciones de la Visión Artificial	54
Tabla 2.9: Entidades del modelo dinámico mediante Lagrange-Euler	61
Tabla 2.10: Entidades de la Ley de Control	61
Tabla 2.11: Representaciones del error	62
Tabla 3.1: Cuadro de Requerimientos	65
Tabla 3.2: Medidas a considerar.....	67
Tabla 3.3: Normas a seguir para la aplicación del algoritmo D-H.....	73
Tabla 3.4: Parámetros D-H del brazo robótico.	74
Tabla 3.5: Propiedades Mecánicas determinantes del ABS	106
Tabla 3.6: Datos obtenidos de Ensayo Estático	114
Tabla 3.7: Datos obtenidos de Ensayo Fatiga	115
Tabla 3.8: Datos obtenidos de Ensayo Estático	116
Tabla 3.9: Datos obtenidos de Ensayo Fatiga	116
Tabla 3.10: Datos obtenidos de Ensayo Estático	118
Tabla 3.11: Datos obtenidos de Ensayo Fatiga	118
Tabla 3.12: Datos de cada Eslabón	134
Tabla 3.13: Ángulos de rotación de cada articulación	135
Tabla 4.1: Orden y descripción de subensambles.	154
Tabla 4.2: Subensamble 1	154
Tabla 4.3: MX - 28	155
Tabla 4.4: Rodamiento SKF 51114	155
Tabla 4.5: RX – 24f	156
Tabla 4.6: Subensamble 2	156
Tabla 4.7: AX - 12	157
Tabla 4.8: Subensamble 3	157
Tabla 4.9: Subensamble 4	158
Tabla 4.10: Tower Pro SG 5010	158
Tabla 4.11: Portagarra.....	159
Tabla 4.12: Comparación Subensamble 1	172
Tabla 4.13: Comparación Subensamble 2	173
Tabla 4.14: Comparación Subensamble 3	173
Tabla 4.15: Comparación Subensamble 4	174
Tabla 4.16: Comparación Portagarra	175
Tabla 4.17: Comparación de propiedades físicas entre el efector final precedente y actual ...	202
Tabla 4.18: Ángulos de rotación de cada articulación	205
Tabla 5.1: Resumen de Entrenamiento de Redes.....	229
Tabla 5.2: Lógica de Estados.....	236

Tabla 5.3: Lógica de Estados Final	236
Tabla 5.4: Lógica Combinacional	236
Tabla 5.5: Valores de Ganancias de Sintonización Virtual	242
Tabla 5.6: Ganancias por Articulación.....	243





“Porque estoy convencido de que ni la muerte, ni la vida, ni ángeles, ni principados, ni lo presente, ni lo por venir, ni los poderes, ni lo alto, ni lo profundo, ni ninguna otra cosa creada nos podrá separar del amor de Dios que es en Cristo Jesús Señor nuestro.”

Romanos 8:38-39

AGRADECIMIENTOS

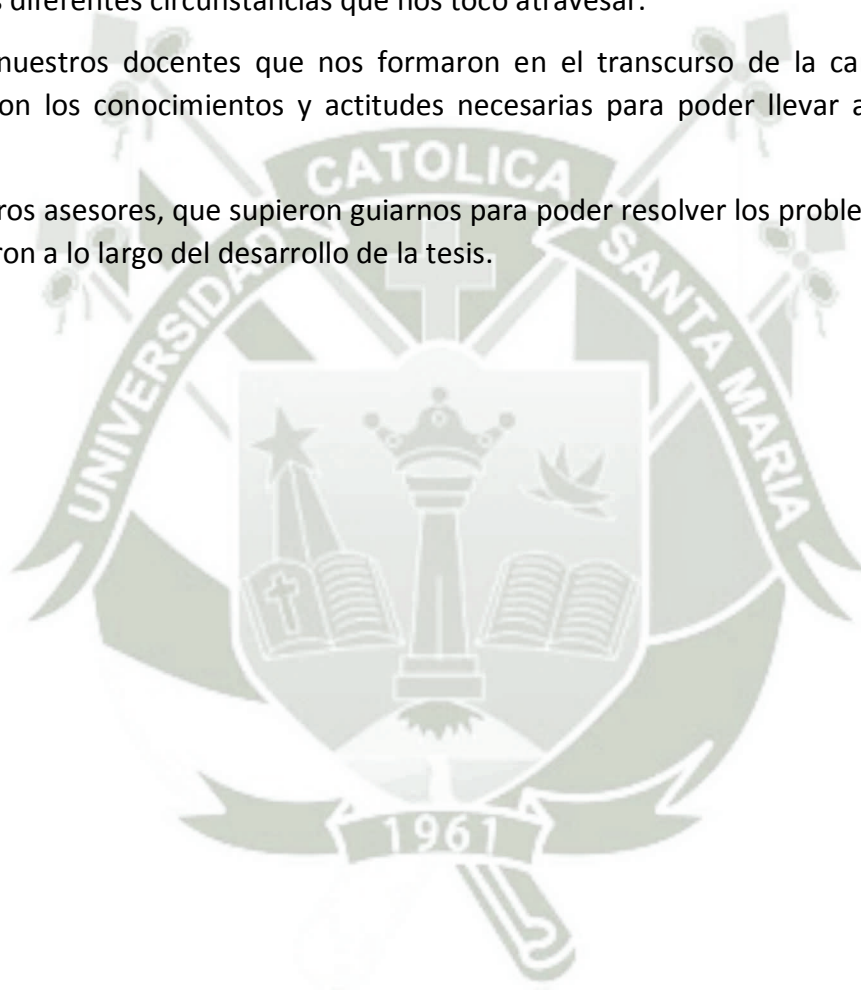
Le agradecemos a Dios por habernos permitido vivir hasta este día, habernos creado tal y como somos, perseverantes y resilientes, para poder soportar todas y cada una de las frustraciones asimiladas a lo largo del proyecto y acompañarnos en cada decisión.

Le damos gracias a nuestros padres por apoyarnos a lo largo de este camino y tener paciencia para soportar nuestras reacciones en los momentos de debilidad. Por mostrarnos con su ejemplo lo que significa ser profesional y guiarnos siempre especialmente durante este periodo de tiempo.

A nuestra familia que siempre nos dieron sus palabras de aliento y confiaron en nosotros pese a las diferentes circunstancias que nos tocó atravesar.

A todos nuestros docentes que nos formaron en el transcurso de la carrera y nos impartieron los conocimientos y actitudes necesarias para poder llevar a cabo este proyecto.

Y a nuestros asesores, que supieron guiarnos para poder resolver los problemas que se presentaron a lo largo del desarrollo de la tesis.



RESUMEN

El presente trabajo de investigación tiene por finalidad, dar una alternativa de solución a la manipulación de sustancias tóxicas para laboratorios específicamente farmacéuticos en salvaguarda de la salud del personal involucrado, razón por la cual esta tesis está evocada a la creación desde el diseño inicial de un prototipo de un brazo robótico de cuatro grados de libertad teleoperado y asistido con visión artificial y redes neuronales.

El proyecto consta de seis capítulos, los mismos que serán descritos a continuación:

El **Capítulo I Fundamentación** muestra la base científica sobre la que se construye la investigación, la misma que inicia con la identificación y descripción de la problemática a abordar; para luego dar un vistazo a los antecedentes con mayor relación a la presente tesis. Posteriormente se planteó los objetivos que se desean lograr al término del proyecto, se formuló una hipótesis y se dio a conocer la justificación y los alcances respectivos, para finalizar con la metodología de investigación que se siguió durante la realización del trabajo.

El **Capítulo II Marco Teórico** recopila de forma sintetizada todo el soporte teórico necesario para comprender cada uno de los temas implicados en la investigación, que fueron aplicados para su desarrollo. Este apartado está dividido principalmente en seis secciones, siendo la primera referida a los laboratorios farmacéuticos con la finalidad de dar una visión global de la aplicación, para luego evocarnos a la parte de ingeniería propiamente del proyecto, definiendo criterios teóricos acerca de Robótica, Teleoperación, Visión Artificial, Redes Neuronales y Control por Torque Computado.

El **Capítulo III Diseño** engloba en su totalidad el proceso de diseño del manipulador robótico desde el modelamiento matemático, pasando por la concepción de forma del robot, hasta la elección de los actuadores tomando en cuenta el torque que se necesita, en condiciones críticas para la parte mecánica. Y en cuanto al diseño electrónico, se consideró la elección de los componentes necesarios para posteriormente llevar a cabo el diseño preliminar del circuito. Finalmente se realizó el diseño de la maqueta, la misma que tiene por finalidad representar el ambiente para efectuar la aplicación del brazo robótico.

El **Capítulo IV Implementación** se divide en dos grandes secciones, la primera referida a la Implementación Mecánica, en la que, a partir del diseño planteado, se fabricó y se realizó el montaje de cada una de las piezas que componen al brazo robótico, para luego llevar a cabo las pruebas respectivas y verificar así un buen funcionamiento, y la segunda, la Implementación Electrónica, en donde se desarrolló la integración de cada uno de los elementos ya elegidos y la impresión final del circuito, para finalmente someterlos en conjunto a diferentes ensayos a fin de comprobar su correcta operatividad.

El **Capítulo V Control e Interfaz Gráfica de Usuario** contiene el desarrollo de toda la estrategia de control aplicada al sistema robótico y la interfaz HMI la misma que facilita la teleoperación del manipulador. En cuanto al Control se puede identificar claramente

tres partes primordiales, iniciando con el modelamiento Kinect, el mismo que alberga la visión artificial del proyecto y se encarga de transformar el movimiento gestual del operador en coordenadas que sean entendibles por el brazo robótico; posteriormente, tenemos el control cinemático basado en redes neuronales, para la obtención de las coordenadas articulares del robot, las mismas que fueron utilizadas como entrada para el control Dinámico por Torque computado. Finalmente, en este apartado se puede encontrar también la programación usada en la tarjeta de Adquisición y Envío de datos, Raspberry, y la integración en su totalidad de cada una de las partes ya descritas para dar lugar al Sistema de Control del Brazo Robótico y realizar así las respectivas pruebas de funcionamiento virtuales.

Y el **Capítulo VI Pruebas y Resultados** muestra la realización del ensayo llevado a cabo a fin de verificar de manera real el buen funcionamiento del proyecto, a nivel de diseño, implementación y control, con ayuda de la Interfaz Gráfica creada y demostrar que se alcanzó el objetivo propuesto.

PALABRAS CLAVE: Brazo robótico, Teleoperación, Visión Artificial, Redes Neuronales, Laboratorios Farmacéuticos.



ABSTRACT

The purpose of this research is to provide an alternative solution to the manipulation of toxic substances for specific laboratories specially in pharmaceutical laboratories to safeguard the health of staff involved, so that, this thesis is evoked to the creation from initial design of a robotic arm is a prototype of 4° of freedom, teleoperated and assisted with artificial vision and neural networks.

The project presents six chapters, which will be described below:

Chapter I Fundamentation shows the scientific basis on which the research is constructed, it begins with the identification and description of the problem to will study and then check similitude to the present thesis.

Afterward, the objectives to be achieved at the end of the project were presented, a hypothesis was formulated and the justification and the respective scope were announced, to finish with the research methodology during the project.

Chapter II Theoretical Framework collects in a synthesized way all needed theoretical support to understand each of the issues involved in the research that was applied for its development. This section is divided mainly into six sections, the first one referring to pharmaceutical laboratories with the aim of giving a global view of the application, then evoking the engineering part of the project, defining theoretical criteria about Robotics, Teleoperation, Artificial Vision, Neural Networks and Computed Torque Control.

Chapter III Design encompasses the whole process of designing the robotic manipulator from mathematical modeling, from the robot structure shape, to the selection of actuators considering the needed torque under mechanical critical conditions. For electronic design, we consider the selection of necessary components to carry out the preliminary circuit design. Finally, we perform the model design, who has the purpose of represent the environment for the robotic arm application.

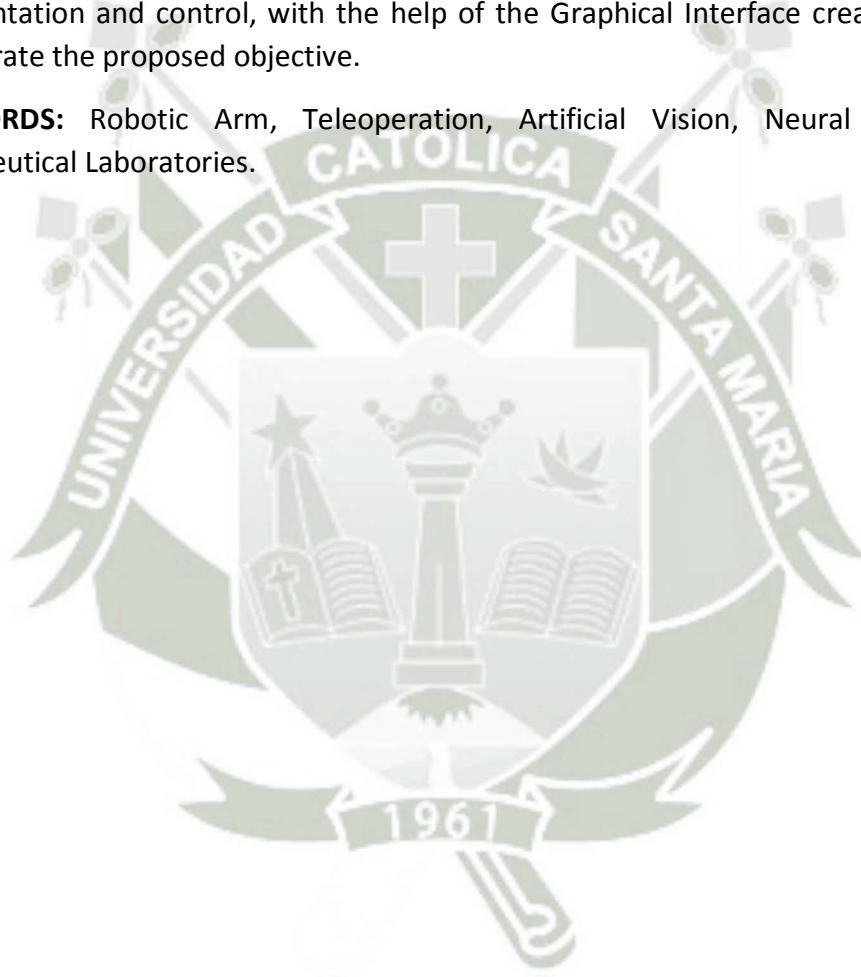
Chapter IV Implementation is divided into two main sections, the first one referred to the Mechanical Implementation, who from the proposed design, was manufactured and we perform the assembly of each piece that compose the robotic arm, to carry out the respective tests and verify the good functioning, and the second one, the Electronic Implementation, where the integration of each of the already chosen elements and the final printing of the circuit was developed, to finally submit them together to different tests In order to verify its correct operation.

Chapter V Control and Graphical User Interface contains the development of the entire control strategy applied to the robotic system and the HMI interface which facilitates the manipulator teleoperation. As for the Control, three primordial parts can be clearly

identified, starting with Kinect modeling, which is the artificial vision of the project and is responsible for transforming the gesture movement of the operator into understandable coordinates by the robotic arm; Later we have the kinematic control based on neural networks, to obtain the articular coordinates of the robot, the same ones that were used as input for the Computational Torque Dynamic control. Finally, in this section you can also find the used programming in the Acquisition and Data sending, Raspberry card, and the entirety integration of each of the parts already described to give rise to the Robotic Arm Control System and perform as well as the respective virtual performance tests.

And **Chapter VI Tests and Results** shows the execution of the test carried out in order to verify in a real way the good operation of the project, referring to the design, implementation and control, with the help of the Graphical Interface created and to demonstrate the proposed objective.

KEY WORDS: Robotic Arm, Teleoperation, Artificial Vision, Neural Networks, Pharmaceutical Laboratories.



INTRODUCCIÓN

“La salud de nuestro cuerpo la gastamos al por mayor; mas una vez perdida, la compramos al por menor”, cautivadora reflexión concebida por el dramaturgo español Albert Llanas¹, el mismo que nos lleva a meditar sobre la importancia que se le da a la vida; donde la mayoría de veces, no se toma en cuenta el riesgo presente en las actividades que se realiza de manera diaria, exponiéndose así al peligro de afectar la salud.

En la actualidad y de manera sorprendente, el hombre pierde salubridad y no posee una noción clara sobre seguridad en el trabajo, lo que resulta realmente alarmante; por ello es necesario saber que las condiciones y los ambientes sanos en la faena, son un derecho básico humano que contribuyen a la productividad y a la creatividad, y que corresponden a las demandas diarias de productos y servicios.

Dentro de las diferentes y numerosas actividades laborales, nos enfocamos en el campo de la investigación y producción de bienes farmacéuticos, tarea que es llevada a cabo en laboratorios en donde el personal está expuesto a sustancias tóxicas que pueden tener consecuencias dañinas a mediano y largo plazo en materia de salud. Se trata de un problema, que, al ser expuesto, conlleva a proponer el uso de la ingeniería; específicamente de la robótica teleoperada, como alternativa de solución.

Así, con la finalidad de garantizar una atmósfera segura de trabajo en un laboratorio farmacéutico, podemos plantear entre otras medidas, el diseño e implementación de un brazo robótico de 4 GDL teleoperado asistido con visión artificial y redes neuronales como métodos sofisticados de control, trabajo llevado a cabo en la presente tesis, siendo esta la clave para el desarrollo tangible del concepto de seguridad en el manejo de sustancias tóxicas en este tipo de laboratorio.

¹ **Albert Llanas:** (1840- 1915) Dramaturgo y humorista español en lengua catalana. Es autor de comedias de costumbres, como El viaje a Montserrat (1881), Don Gonzalo o el orgullo de la chaqueta (1891) y El sí de las muchachas (1911).

CAPÍTULO I: FUNDAMENTACIÓN

1.1. Identificación del problema:

La industria farmacéutica y específicamente los laboratorios de investigación de productos farmacológicos toman sus propias precauciones en materia de riesgo sobre la salud, pudiendo exponer al personal involucrado ya que en estos laboratorios se manipula materia prima tóxica.

1.1.1. Descripción del Problema:

El área de investigación y producción de bienes farmacológicos a nivel industrial es un proceso largo, científico y técnicamente complejo, que ocupa un importantísimo sector de la industria química y la propiamente farmacéutica contribuyendo así en el campo de la salud, y cuya principal finalidad consiste en elaborar medicamentos de alta calidad para el tratamiento de enfermedades, haciendo de ella una profesión de gran responsabilidad social. Razón por la cual, es esencial tratar con vital importancia los diferentes aspectos que aparecen durante el desarrollo de esta actividad, entre los que se tiene a la prevención de riesgos laborales, control de calidad y evaluación del impacto medioambiental.

Según un estudio realizado por la licenciada en Ciencias Biológicas M. DOLORES SÁNCHEZ GARCÍA, Máster en Seguridad y Salud Laboral, Especialidad en Higiene Industrial de la Universidad Autónoma de Madrid², el análisis de los estudios epidemiológicos y experimentales ponen de manifiesto que una exposición continuada a los contaminantes presentes en el ambiente laboral, puede producir serios problemas sobre la salud de los trabajadores de la industria farmacéutica, incluyendo a personal de laboratorios, de planta de producción y de servicio auxiliar de limpieza, es posible nombrar diferentes tipos de alteraciones sobre la salud de los trabajadores ocasionados por la exposición a sustancias tóxicas, entre otros pueden citarse:

- a) Riesgo en la incidencia de asma o alergia ocupacional provocada por inhalación de sustancias químico-farmacéuticas y de laboratorio.
- b) Riesgo en la incidencia de cáncer.
- c) Riesgo de alteraciones en la reproducción entre los trabajadores de la industria farmacéutica.
- d) Resistencia a los antibióticos.

² Estudio de la situación de riesgos específicos en la industria farmacéutica: Este artículo es el resumen del trabajo presentado a la Fundación MAPFRE como resultado final de la investigación desarrollada durante 1997 a raíz de una beca concedida en su convocatoria 1996/1997.

Y de todas ellas quizá sea el riesgo de cáncer y de asma, las enfermedades de mayor impacto que se pueden desarrollar a largo plazo en los empleados, y que representan una interrogante preocupante.

Entonces, al considerar que la exposición del trabajador puede durar años con efectos potenciales sobre su salud; podríamos hacer uso del desarrollo de la tecnología para la manipulación de forma remota de agentes químicos tóxicos, durante los procesos de investigación y producción de medicamentos, con el fin de salvaguardar la salud de los trabajadores de la industria farmacéutica, evitando el contacto directo, para la no contaminación tanto de las personas como de los propios productos.

- **Campo, Área y Línea de Investigación:**

Campo: Ingeniería

Área: Ingeniería Mecatrónica

Línea: Robótica y Teleoperación.

1.2. Antecedentes:

En 1947 comenzaron las primeras investigaciones, lideradas por Raymond Goertz del Argonne National Laboratory en Estados Unidos, encaminadas al desarrollo de algún manipulador de fácil manejo a distancia a través de la operación de otro manipulador equivalente. El primer fruto se obtuvo en 1948 con el desarrollo del primer manipulador mecánico, denominado M1, el mecanismo de este sistema permitía que la pinza en el extremo del manipulador esclavo reprodujera de forma fiel los movimientos hechos por la mano del operador al extremo del manipulador maestro, pues ambos manipuladores eran prácticamente iguales, y los movimientos entre ambos se reproducían eje a eje, de tal manera que el extremo de las dos máquinas describiese la misma trayectoria.

A principios de los años cincuenta se comenzaron los desarrollos encaminados a motorizar ambos manipuladores maestro y esclavo; es así que, en el año 1954, Goertz presentó el primer manipulador maestro-esclavo con accionamiento eléctrico y servocontrol llamado E1.

En los años sesenta se extendieron las investigaciones hasta el campo de las aplicaciones submarinas, incluyendo cámaras y dispositivos para aumentar la telepresencia del operador y a finales del mismo y principios de los setenta la teleoperación alcanzó su máximo desarrollo en aplicaciones espaciales; de tal manera, que aparecieron nuevos retos y problemas, siendo de especial relevancia la existencia de retrasos temporales en la comunicación entre la zona local y la zona remota.

Al mismo tiempo con la evolución de las técnicas de teleoperación, ha habido una evolución tecnológica motivada por los desarrollos de control, la informática y la robótica; de esta manera, encontramos un desarrollo en los sistemas de comunicación,

pasando de los mecánicos a los eléctricos, fibra óptica, radio e Internet, medios que suprimen prácticamente las limitaciones de distancia. La incorporación de los progresos de la robótica y la tecnología multimedia han permitido incrementar las capacidades del sistema remoto, especialmente en lo que se refiere a su autonomía, y del puesto local de control, mejorando fundamentalmente las prestaciones de la interfaz hombre máquina.

Los primeros robots de laboratorio comerciales se presentaron en la Conferencia de Pittsburgh celebrada en 1982. La implantación de tecnologías robotizadas ha ido creciendo paulatinamente en los laboratorios de todo el mundo, actualmente existen publicaciones periódicas internacionales exclusivamente enfocadas hacia la automatización en campo como "Laboratory Robotics and Automation, Journal of Automatic Chemistry y Advances in Laboratory Automation Robotics "(1995).

Dentro de las más destacadas invenciones, tenemos el robot de doble brazo CSDA10F, desarrollado en el año 2013 por la empresa YASKAWA de manera industrial, para satisfacer los rigurosos requisitos de salubridad de los entornos en laboratorios. Según pruebas realizadas en el campo de la biomédica y la química analítica, se ha demostrado que este robot es capaz de trabajar prácticamente con gran parte de los equipos disponibles e instrumentos de análisis convencionales en laboratorios estándar.

1.3. Objetivos:

1.3.1. Objetivo General:

Diseñar e implementar un prototipo de un brazo robótico de 4 grados de libertad teleoperado para la manipulación de sustancias tóxicas asistido con visión artificial y redes neuronales para laboratorios farmacéuticos.

1.3.2. Objetivos Específicos:

- Realizar el diseño mecánico del sistema de prueba, así como el diseño mecánico, cinemático y dinámico del prototipo, generando planos y especificaciones técnicas.
- Implementar el prototipo, adjuntar lista de componentes y procedimiento de su construcción.
- Definir los algoritmos inteligentes para el procesamiento de imágenes y control por redes neuronales.
- Controlar el prototipo a través de visión artificial y redes neuronales para lograr la teleoperación del sistema.

1.4. Hipótesis:

La manipulación de sustancias tóxicas en la industria farmacéutica es peligrosa para la salud del personal involucrado. Para lo cual se empleará tecnología adecuada para estas operaciones, como los robots manipuladores controlados con tecnología de última generación: visión artificial con redes neuronales. Así de esta forma se garantizará seguridad en todos los agentes que intervienen en la manipulación de sustancias tóxicas de la industria farmacéutica.

1.5. Justificación:

En la actualidad en los laboratorios farmacéuticos de nuestra región, el tratamiento de sustancias tóxicas se realiza de manera semiautomática. Sin embargo, esto aún es riesgoso para la salud del personal, por lo que se propone el diseño y la implementación de un manipulador robótico que permita realizar este procedimiento mediante la teleoperación y visión artificial englobada en un control basado en redes neuronales.

El manipulador será de 4 GDL ya que con este número se cumple con el volumen de trabajo requerido para el proceso a realizar. Se usará visión artificial por la necesidad de que sea inalámbrico y eludir así el contacto entre el usuario y el producto final por dos razones: para evitar algún problema de salud del usuario y para evitar la contaminación del producto final por algún agente de parte del empleado.

La investigación será segmentada de manera que cada parte anteceda a la siguiente, se comenzará con una recopilación adecuada de información lo que permitirá el diseño inicial del proyecto para luego añadir uno a uno los temas restantes hasta obtener el resultado final.

El trabajo tendrá una finalidad práctica con resultados concretos en la industria farmacéutica ya que permitirá el cuidado de la vida humana sin dejar de lado la necesidad de investigación.

1.6. Alcances:

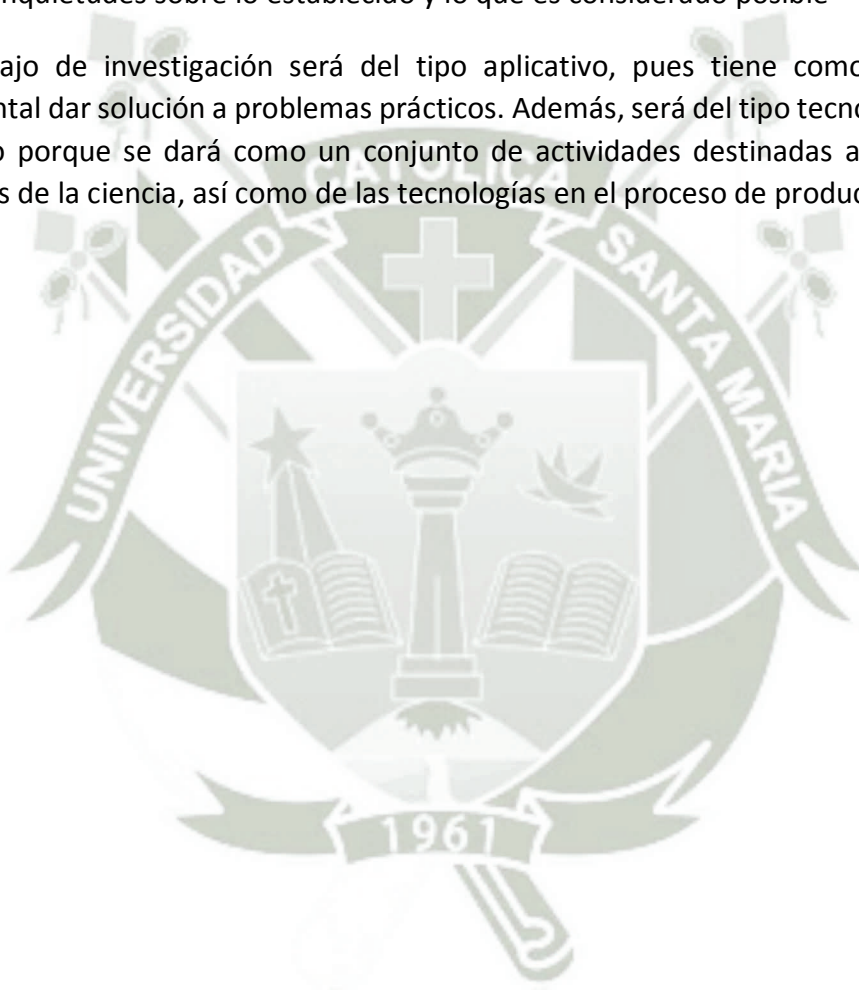
La presente tesis tiene como alcance el diseño e implementación de un manipulador robótico de cuatro grados de libertad, teniendo solo movimientos rotativos y el actuador final será una pinza apropiada para el manejo de instrumentos de laboratorio. En cuanto a la implementación de esta nueva tecnología, se utilizarán materiales que cumplan con las especificaciones de higiene necesarias durante la investigación de nuevos productos farmacológicos, además se usarán equipos tecnológicos de alta precisión para lograr un adecuado control y teleoperación del sistema.

1.7. Metodología de la investigación:

El método científico tiene ventajas indiscutibles: por su naturaleza analítica, es capaz de pasar de lo simple a lo complejo, construyendo teorías que experimentan un continuo progreso. La investigación nos permite el contacto con la realidad en la que estamos inmersos, despierta la curiosidad y lo más importante, puede resolver problemas, tanto en el presente como en el futuro. Cumpliendo con ello una gran función social.

La función social que tiene la investigación radica en que el ser humano interactúa en grupo, por eso, busca entenderse como parte del grupo. La participación de los jóvenes durante la investigación es muy importante, ya que en dicha etapa de la vida surgen mayores inquietudes sobre lo establecido y lo que es considerado posible

Este trabajo de investigación será del tipo aplicativo, pues tiene como propósito fundamental dar solución a problemas prácticos. Además, será del tipo tecnológico y de desarrollo porque se dará como un conjunto de actividades destinadas a utilizar los resultados de la ciencia, así como de las tecnologías en el proceso de producción.



CAPÍTULO II: MARCO TEÓRICO

2.1. Laboratorios Farmacéuticos:

Son aquellas personas físicas o jurídicas que, previamente autorizadas por la autoridad competente, fabriquen de forma industrial medicamentos o participen en alguna de sus fases, tales como el envasado, acondicionamiento, presentación para su venta o comercialización.

2.1.1. Instrumentos en Laboratorios Farmacéuticos:

Los materiales que usualmente se utilizan en el laboratorio son de vidrio, plástico o porcelana. Las ventajas primordiales que determinan la elección de uno u otro de estos materiales son:

- **Vidrio:** su gran estabilidad y óptima resistencia térmica.
- **Plástico:** Es económico, apropiado para contener soluciones alcalinas y sirve para fabricar materiales desechables.
- **Porcelana:** De gran utilidad cuando se requiere una gran resistencia térmica.

2.1.2. Manipulación de Sustancias Peligrosas:

Una sustancia química peligrosa es aquella que por sus características físicas puede causar daño directa o indirectamente a seres humanos o materiales. Tenemos entre otras, los siguientes tipos de sustancias:

- **Muy tóxicos:** Sustancias que por su penetración en el organismo suponen riesgos extremadamente graves, agudos o crónicos e incluso la muerte. Pueden introducirse en el organismo por inhalación, ingestión o a través de la piel. Por ejemplo: la nicotina, el cianuro de sodio, el fósforo blanco, etc.
- **Tóxicos:** Sustancias que por inhalación, ingestión o penetración cutánea pueden implicar graves riesgos, agudos o crónicos e incluso la muerte. Por ejemplo, el mercurio, el arsénico, el selenio, etc.
- **Nocivos:** Sustancias que por inhalación, ingestión o penetración cutánea pueden tener riesgos de gravedad limitada. Por ejemplo, el yodo, sulfuro de bario, permanganato de potasio, etc. Estas sustancias no deben manipularse con las manos, evitando el contacto con el cuerpo y no inhalar los vapores emanados.
- **Corrosivos:** Sustancias que en contacto con los tejidos vivos pueden ejercer sobre ellos una acción destructora. Por ejemplo, el bromo, el hidróxido de sodio. Debe evitarse todo contacto del cuerpo y vestimenta con estas sustancias y no inhalar sus vapores.
- **Comprometido con el medio ambiente:** Sustancias que causan o pueden causar peligro inmediato o futuro para el entorno físico, químico o biológico del medio ambiente.

Tomando como referencia para el proyecto los elementos tóxicos y muy tóxicos ahondaremos un poco en 4 de ellos:

- **Arsénico:**

Los compuestos orgánicos trivalentes del arsénico (sulfarsenol, amebarsone, cacodilatos, etc.) y sus formas inorgánicas (arsenito de sodio y de potasio) se usan para la elaboración de productos farmacéuticos como por ejemplo los usados en el tratamiento de enfermedades de la piel, la psoriasis. En odontología se usa para destruir la pulpa dentaria.

El Arsénico es uno de los elementos más tóxicos que se encuentran en la naturaleza, debido a sus efectos dañinos, pues este afecta a los seres humanos que están expuestos directa o indirectamente a través de la comida, agua y aire. Es decir, La exposición puede también ocurrir a través del contacto de la piel con suelo o agua que contenga este elemento.

La exposición al Arsénico inorgánico causa efectos sobre la salud humana, como la irritación del estómago e intestinos, disminución en la producción de glóbulos rojos y blancos, cambios en la piel, e irritación de los pulmones. También la toma de significantes cantidades de Arsénico inorgánico puede potenciar las posibilidades de desarrollar cáncer, especialmente el desarrollo de cáncer de piel, pulmón, hígado, linfa.

La exposición de grado mayor de Arsénico inorgánico causa infertilidad y abortos en mujeres, también causar perturbación en la piel, baja resistencia a infecciones, trastornos en el corazón y daño del cerebro tanto en hombres como en mujeres. Y lo más grave, el Arsénico inorgánico puede alterar el ADN. El Arsénico orgánico no puede causar cáncer, ni tampoco daño al ADN. Pero exposiciones a dosis elevadas puede causar ciertos efectos sobre la salud humana, como lesiones de nervios y dolores estomacales.

- **Selenio:**

Los seres humanos están expuestos al selenio en varias formas. Entre ellas se da a través de la ingesta de comida o el agua, o cuando se pone en contacto con tierra o aire que contiene altas concentraciones de selenio. Lo podemos encontrar en productos como el Holoram Detox que es un catalizador natural que tiene entre sus componentes al selenito sódico pentahidratado.

También la exposición al selenio a través del aire ocurre en el centro laboral, que puede provocar mareos, fatiga e irritaciones de las membranas mucosas.

La sobreexposición a vapores de selenio puede producir acumulación de líquido en los pulmones, mal aliento, bronquitis, neumonía, asma bronquítica, náuseas, escalofríos, fiebre, dolor de cabeza, dolor de garganta, falta de aliento, conjuntivitis, vómitos, dolores abdominales, diarrea y agrandamiento del hígado. El selenio es irritante y sensibilizador de los ojos y del sistema respiratorio superior.

La sobreexposición puede resultar en manchas rojas en las uñas, dientes y pelo. El dióxido de selenio reacciona con la humedad para formar ácido selénico, que es corrosivo para la piel y ojos. El envenenamiento por selenio puede volverse tan agudo en algunos casos que puede incluso causar la muerte.

- **Cianuro de sodio:**

El cianuro de sodio es un polvo blanco (como la sal) con un ligero olor de almendra. Es usado en solución para extraer minerales metálicos, en galvanoplastia y baños de limpieza de metales, en el endurecimiento de metales y en insecticidas.

También se usa comercialmente en la industria farmacéutica para componer sustancias como el cloruro cianúrico, y varios nitrilos. El cianuro se utiliza en productos farmacéuticos como el laetril, una sustancia para combatir el cáncer, y el nitroprusiato, una droga para reducir la presión arterial. Los compuestos de cianuro también se utilizan en vendas quirúrgicas que promueven la cicatrización y reducen las cicatrices.

Puede causar efectos perjudiciales al absorberse a través de la piel y en contacto con este elemento puede irritar la piel y los ojos. Respirar cianuro de sodio irrita la nariz, la garganta y los pulmones, causando tos, respiración con silbido o falta de aire.

La sobre exposición causa dolor de cabeza, mareo, latidos del corazón rápidos e incluso pérdida de conocimiento y muerte. Podría causar también el agrandamiento de la glándula tiroides, daño al sistema nervioso y alteración en el recuento de glóbulos sanguíneos

La exposición puede causar daño al sistema nervioso y cambios en el recuento de glóbulos sanguíneos. La exposición baja repetida puede causar secreciones, hemorragia y lesiones en la nariz.

- **Mercurio:**

Se encuentra en productos como tiomersal que es un agente antiséptico y antifúngico, un conservante compuesto de etilmercurio y tiosalicilato. El Mercurio metálico es usado en una variedad de productos de las casas, como barómetros, termómetros, bombillas fluorescentes. El Mercurio en estos elementos está encapsulado y por lo general no causa problema de salud al ser humano. De cualquier manera, cuando un termómetro se rompe una exposición significativamente alta al Mercurio ocurre a través de la respiración, esto ocurrirá por un periodo de tiempo corto mientras este se evapora. Esto puede causar efectos dañinos, como daño a los nervios, al cerebro y riñones, irritación de los pulmones, irritación de los ojos, reacciones en la piel, vómitos y diarreas.

El Mercurio tiene una variedad de efectos dañinos sobre los humanos, que pueden ser resumidos así:

- Daño al sistema nervioso
- Daño a las funciones del cerebro
- Daño al ADN y cromosomas
- Reacciones alérgicas, irritación de la piel, cansancio, y dolor de cabeza
- Efectos negativos en la reproducción, daño en el espermatozoides, defectos de nacimientos y abortos

El daño a las funciones del cerebro puede causar la degradación de la habilidad para aprender, cambios en la personalidad, temblores, cambios en la visión, sordera, incoordinación de músculos y pérdida de la memoria. Daño en el cromosoma y es conocido que causa mongolismo, etc.

2.2. Robótica:

A lo largo de toda la historia, el ser humano se ha sentido impresionado por máquinas y dispositivos capaces de reproducir las funciones y los movimientos de los seres vivos. Podemos resumir en la siguiente tabla el desarrollo histórico de la robótica en el tiempo.

Tabla 2.1: Historia de la Robótica

Año	Acontecimiento
Previo al siglo VIII	Los griegos denominaron a los robots: <i>autómatas</i> , <i>máquinas que imitan la figura y movimientos de un ser animado</i> .
Siglo VIII a XV	La cultura árabe difundió los conocimientos griegos, dándoles una aplicación práctica e introduciéndolos en la vida cotidiana de la realeza.
Siglo XV y XVI	Leonardo Da Vinci construyó un León mecánico para el rey Luis XII de Francia, el mismo que habría su pecho con su garra y mostraba el escudo de armas del rey.
Siglos XVII y XVIII	Fueron creados ingenios mecánicos, cuyo fin era el de entretener a la gente de la corte y servir de atracción en las ferias. Como ejemplos relevantes Se tiene el pato de Vaucanson y los muñecos de la familia Droz y de Mailladert.
A finales del siglo XVIII y principios del siglo XIX	Se desarrollaron invenciones mecánicas, utilizadas en la industria textil, entre las que destaca el telar de Jacquard (1801). A partir de ese momento se dió paso a la automatización industrial.
Año 1921	Karel Capek estrena en el teatro nacional de Praga su obra "Rossum's Universal Robot".
Octubre de 1945	El escritor ruso Isaac Asimov publicó en la revista Galaxy Science Fiction en la que enunció por primera vez sus tres leyes de la robótica ⁴ .
Año 1948	Goertz desarrolló el primer telemanipulador que consistía en un dispositivo mecánico maestro-esclavo.
Año 1954	Goertz creó el primer telemanipulador bilateral con transmisión eléctrica.
Años 1958	Ralph Mosher desarrolló un dispositivo denominado Handy-Man.
Años 1960	Uso de los manipuladores, base del robot industrial. Primera máquina Unimate, en la fábrica de General Motors de Trenton, Nueva Jersey
1968	J.F. Engelberger, construcción de robots Unimate
Año 1973	Firma sueca ASEA, construyó el robot IRb6, seguido más tarde del IRb60.
Año 1982	El profesor Makino, desarrolla el concepto de robot SCARA

30 años en adelante	Evolución de robots industriales, espaciales, para aplicaciones submarinas y subterráneas. Robots militares, robots móviles para aplicaciones médicas, aplicaciones agrícolas. Telemanipuladores.
---------------------	---

Fuente: Propia

2.2.1. Definición de Robot:

Los primeros intentos de establecer una definición formal de robot surgen en el año 1979 por parte de la RIA (*Robot Institute of America, actualmente Robotic Industries Association*), según la cual:

“Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas”.

Esta temprana definición, matizada y acotada, ha sido la referencia para las sucesivas definiciones que se han ido dando al robot hasta llegar a la actual, establecida por la Asociación Internacional de Estándares (ISO), que define en su norma ISO 8373, al Robot manipulador industrial como:

“Manipulador de 3 o más ejes, con control automático, reprogramable, multiplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia)”.

Una definición también dada por la Asociación Francesa de Normalización (AFNOR), que define primero el manipulador, y basándose en dicha definición, el robot:

“Manipulador: mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivos lógicos”.

“Robot: manipulador automático servocontrolado, reprogramable, polivalente, capaz de posicionar y orientar piezas, útiles o dispositivos especiales, siguiendo trayectorias variables reprogramables, para la ejecución de tareas variadas. Normalmente tiene la forma de uno o varios brazos terminados en una muñeca. Su unidad de control incluye un dispositivo de memoria y ocasionalmente de percepción del entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material”.

2.2.2. Clasificación de los Robots:

Un robot se puede clasificar teniendo en cuenta diferentes criterios o características, siendo algunas de ellas dependientes de su propia naturaleza, otras de la aplicación o tarea a la que se dedica.

2.2.2.1. Clasificación según la generación:

En general, al hacer referencia a la generación de un robot se está tomando en cuenta al tiempo y momento tecnológico en que éste aparece; así se puede considerar que se pasa de una a la siguiente generación cuando se tiene un hito que supone un avance importante en las capacidades de dichas máquinas.

Esta clasificación es interesante, pues nos permite tener una idea del progreso tecnológico del robot. En la Tabla 2.2 se agrupa una clasificación en generaciones. En la historia temporal podemos mencionar que la primera generación va desde el comienzo de la robótica hasta los años ochenta. La segunda generación evoluciona en los años ochenta y es la que en la actualidad se puede encontrar en las industrias. Y la tercera generación está aún desarrollándose en estos días, siendo así, objeto de un futuro próximo.

Tabla 2.2: Clasificación de los Robots según la Generación

1. ^a Generación	Repite la tarea programada secuencialmente. No toma en cuenta las posibles alteraciones de su entorno
2. ^a Generación	Adquiere información limitada de su entorno y actúa en consecuencia. Puede localizar, clasificar (visión) y detectar esfuerzos y adaptar sus movimientos en consecuencia.
3. ^a Generación	Su programación se realiza mediante el empleo de un lenguaje natural. Posee capacidad para la planificación automática de tareas.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.2.2.2. Clasificación según el área de aplicación:

Desde el punto de vista del uso que se le puede dar al robot podemos clasificarlos, o en base al sector económico en el que trabaja o al tipo de aplicación que desarrolla, independientemente del sector económico en el que trabaje.

La International Federation of Robotics (IFR), clasifica las aplicaciones de los robots industriales manipuladores, tal como se muestra en la Tabla 2.3.

Tabla 2.3: Clasificación de las aplicaciones de los Robots industriales manipuladores, según IFR.

000	Sin especificar.
110	Manipulación en fundición.
130	Manipulación en moldeo de plásticos.
140	Manipulación en tratamientos térmicos.
150	Manipulación en la forja y estampación.
160	Soldadura.
170	Aplicación de materiales.
180	Mecanización.
190	Otros procesos.
200	Montaje.
210	Paletización y empaquetado.
220	Medición, inspección, control de calidad.
230	Manipulación de materiales.
240	Formación, enseñanza e investigación.
900	Otros.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

De la misma manera, la IFR propone una clasificación según el tipo de aplicación. En ésta, se dividen las aplicaciones en Personales o Domésticas, Profesionales y en general aplicaciones de I+D, tal y como se muestra en la Tabla 2.4.

Tabla 2.4: Clasificación de los robots de servicio por Áreas de aplicación, según IFR.

Sección I	ROBOTS PERSONALES Y DOMÉSTICOS
1-5	Robots para tareas domésticas
6-10	Robots de entretenimiento
11-14	Asistenciales, ayuda a discapacitados
15	Transporte Personal
16	Seguridad y vigilancia de la vivienda
17	Otros usos personales y domésticos
Sección II	ROBOTS DE SERVICIOS PROFESIONALES
18-23	Robots de exteriores
24-28	Limpieza profesional
29-31	Sistemas de inspección
32-36	Construcción y demolición
37-40	Sistemas logísticos
41-44	Medicina
45-50	Defensa, rescate y seguridad
51	Submarinos
52	Plataformas móviles de uso general
53-55	Robots de laboratorio
56-59	Relaciones públicas
60-61	Propósito especial
62	Humanoides
63	Robots a medida
64	Otros no especificados
Sección III	I+D EN ROBÓTICA
64	Percepción
65-67	Actuación
68	Micro y nano robots
69	Arquitecturas e integración
70	Navegación y control
71	Interfases con usuario y otras

72	Otras actividades de I+D no especificadas
73	Investigación básica

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.2.2.3. Clasificación según el tipo de actuador:

Dependiendo del tipo de energía a utilizar por los ejes principales del robot, éste también puede ser clasificado como:

- Robot Neumático.
- Robot Hidráulico.
- Robot Eléctrico.

Actualmente, la mayoría de los robots tienen accionamiento eléctrico, no descartando que se puedan tener casos particulares de robots con accionamiento hidráulico o neumático. Los accionamientos hidráulicos en particular son recomendables cuando se desea una elevada relación capacidad de carga-peso del robot o cuando se tenga la necesidad de disponer aislamiento eléctrico entre el robot y el resto del sistema.

2.2.2.4. Clasificación según el número de ejes:

Esta característica se aplica a los robots o telerobot con cadena cinemática; es decir, a los robots manipuladores, pero no, a los robots móviles.

Entiéndase por eje a cada uno de los movimientos independientes con que está provisto el robot. En concordancia a la definición ISO, "...el robot manipulador industrial debe tener al menos 3 ejes y extendiendo esta condición a los robots de servicio manipuladores, se podrán encontrar robots de cualquier número de ejes superior o igual a 3".

En la realidad, la mayoría de los robots están dotados de 6 ejes, seguidos por los de 4. Los robots con más de 6 ejes son muy poco frecuentes, y se justifica este número para potenciar la capacidad de maniobra del robot, siendo en muchas ocasiones telerobots.

2.2.2.5. Clasificación según su configuración:

Esta clasificación, al igual que el caso anterior, es sólo aplicable a robots o telerobots con cadena cinemática.

La configuración de un robot queda determinada por el tipo de movimientos permitidos entre 2 eslabones consecutivos de la cadena. De acuerdo a esto se tiene los siguientes:

- **Cartesiano:** cuyo posicionamiento en el espacio se lleva a cabo mediante articulaciones lineales.
- **Cilíndrico:** cuenta con una articulación rotacional sobre una base y articulaciones lineales para el movimiento en altura y en radio.
- **Polar o Esférico:** consta de dos articulaciones rotacionales y una lineal.
- **Angular:** todas sus articulaciones son rotacionales.
- **SCARA:** posee varios tipos de articulaciones, combinaciones de las anteriores.
- **Paralelo:** posee brazos con articulaciones prismáticas o rotacionales concurrentes.

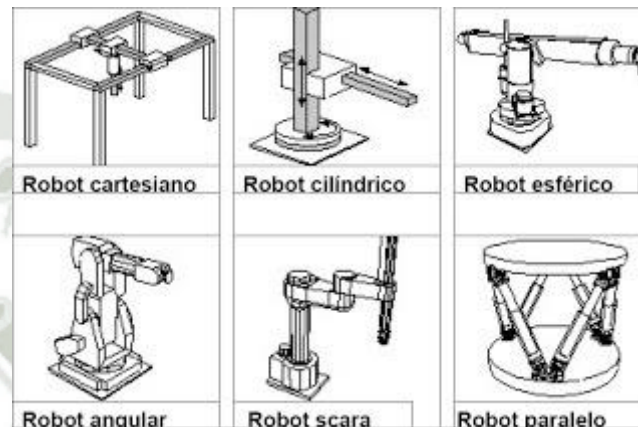


Figura 2.1: Clasificación de Robots según su Configuración.

Fuente: "Actuadores (Robots)", Borjanen, 2010.

2.2.2.6. Clasificación según el tipo de control:

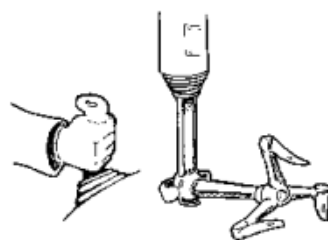
Atendiendo al tipo de control, la norma ISO 8373 y, en consonancia la IFR, distingue entre los siguientes:

- **Robot secuencial (ISO):** Robot en el que su sistema de control permite que el conjunto de movimientos se efectúe eje a eje en un orden determinado, de forma tal que la finalización de un movimiento inicie al siguiente.
En este tipo de robots es solo posible controlar una secuencia de puntos de parada, teniendo un movimiento punto a punto.
- **Robot controlado por trayectoria (ISO):** Robot que ejecuta una tarea controlada en el cual los movimientos de tres o más ejes controlados se van desarrollando según ordenes que especifican en el tiempo, el camino requerido para llegar a la siguiente posición, que se obtiene normalmente por interpolación.
Los robots controlados por trayectoria permiten que se ejecuten movimientos en los que puede ser especificado toda la trayectoria de manera continua.
- **Robot adaptativo (ISO):** Robot que tiene sistemas de control que incorporan sensores, control adaptativo, o funciones de control de aprendizaje.

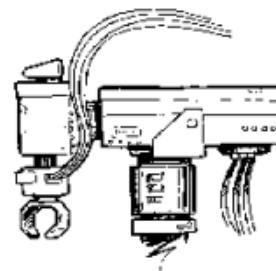
De esta manera el robot puede modificar su labor en concordancia a la información captada del entorno de trabajo, por ejemplo, con un sistema de visión por computador o por sensores de fuerza o contacto.

- **Robot Teleoperado (ISO):** Un robot que puede ser controlado en forma remota por un operador humano, extendiendo la capacidad sensorial y motora de éste a localizaciones remotas.

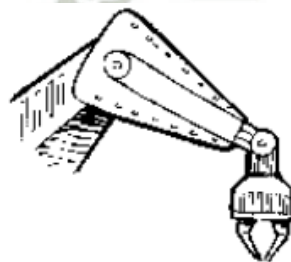
La Figura 2.2 muestra gran parte de los diferentes tipos de robots que se pueden encontrar en la actualidad.



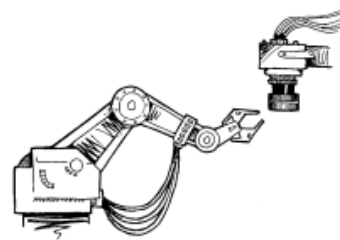
a. Telemanipulador.



b. Manipulador neumático secuencial.



c. Robot manipulador industrial.



d. Robot manipulador industrial sensorizado.



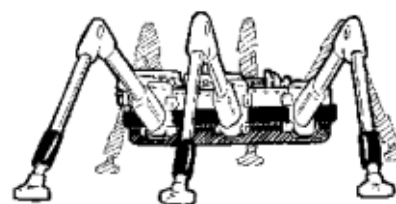
e. Robot de intervención en exteriores.



f. Robot doméstico.



g. Robot aéreo.



h. Robot con patas.



i. Robot submarino.



j. Robot humanoide.

Figura 2.2: Clasificación de Robots según el tipo de Control.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.2.3. Morfología del Robot:

La definición de "morfología" según el diccionario de la lengua española es el "estudio de la forma o estructura de alguna cosa".

En esta definición aplicada a la de un robot como "cosa", podremos decir que este está formado por los elementos constitutivos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales.

2.2.3.1. Estructura Mecánica:

Dentro de esta estructura, un manipulador robótico está formado por varios elementos de armaduras rígidas, llamados cadenas o eslabones, que están enlazados entre sí por articulaciones que permiten el movimiento relativo entre dos eslabones consecutivos.

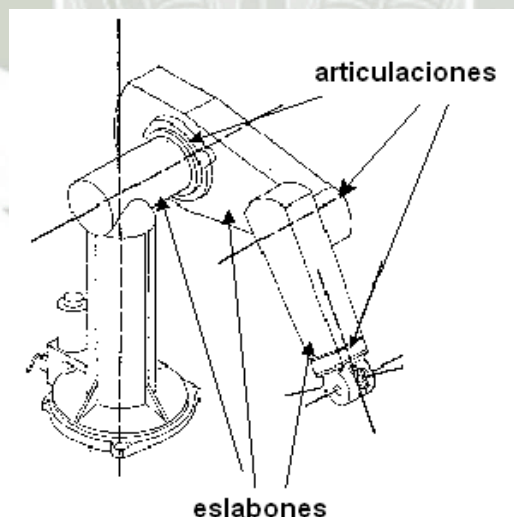


Figura 2.3: Componentes estructurales de un robot industrial.

Fuente: "Estructura de un Robot Industrial", Victor. R. Gonzales 2002.

Uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior, se denomina grado de libertad (GDL). El número de grados de

libertad del robot viene dado por la suma de los grados de libertad de las articulaciones que lo componen. Ya que las articulaciones empleadas son únicamente las de rotación y prismática con un solo GDL cada una, el número de GDL del robot suele coincidir con el número de articulaciones de las que se compone.

Para posicionar y orientar un cuerpo de cualquier manera en el espacio son necesarios seis parámetros, tres para definir la posición y tres para la orientación, si se pretende que un robot posicione y oriente su extremo de cualquier modo en el espacio, se precisarán al menos seis GDL. Sin embargo, en la práctica, a pesar de ser necesarios los seis GDL, muchos robots industriales cuentan con sólo cuatro o cinco GDL, por ser estos suficientes para llevar a cabo las tareas que se les encomiendan.

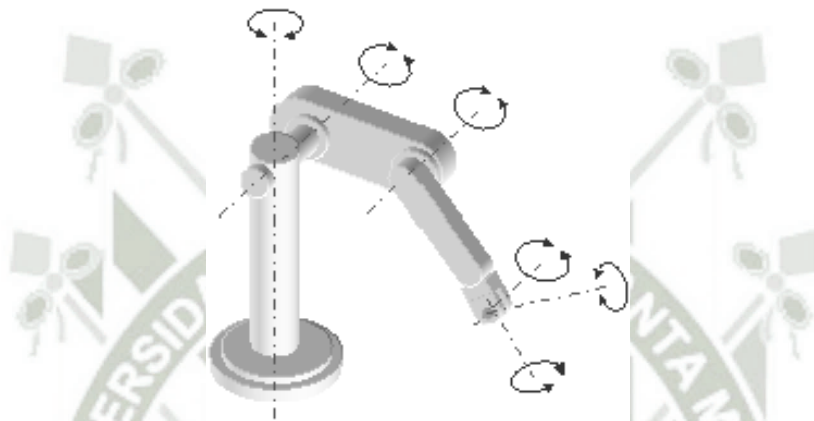


Figura 2.4: Grados de Libertad de un brazo robótico.
Fuente: "Estructura de un Robot Industrial", Víctor. R. Gonzales 2002.

El empleo de diferentes combinaciones de articulaciones en un robot da lugar a diferentes configuraciones, con características a tener en cuenta tanto en el diseño y construcción del robot como en su aplicación.

El conjunto de eslabones y articulaciones se denomina cadena cinemática. Se dice que una cadena cinemática es abierta si cada eslabón se conecta mediante articulaciones exclusivamente al anterior y al siguiente, exceptuando el primero, que se suele fijar a un soporte, y el último, cuyo extremo final queda libre.

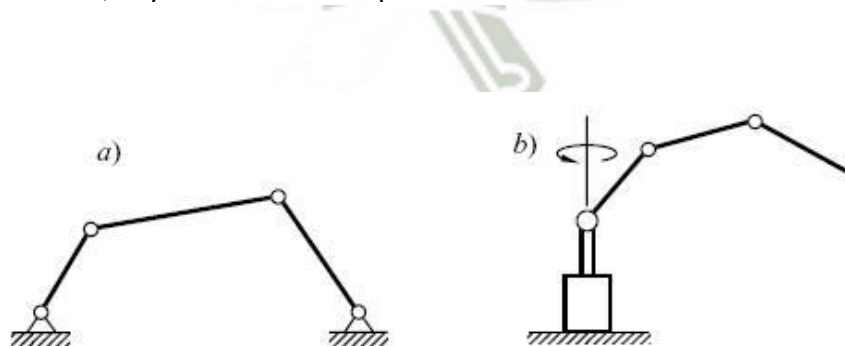


Figura 2.5: a) Cadena Cinemática Cerrada. b) Cadena Cinemática Abierta.
Fuente: "Máquina y Mecanismo", Sitenordeste 2015.

Al extremo del robot se le puede conectar un elemento terminal o actuador final: una herramienta especial que permite al robot de uso general realizar una aplicación

particular, y que debe diseñarse específicamente para dicha aplicación: una herramienta de sujeción, de soldadura, de pintura, etc. El punto más significativo del elemento terminal se denomina punto terminal (Fig. 2.6). En el caso de una pinza, el punto terminal vendría a ser el centro de sujeción de la misma.

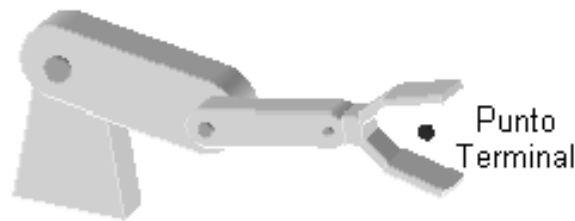


Figura 2.6: Punto terminal de un manipulador.

Fuente: "Estructura de un Robot Industrial", Victor. R. Gonzales 2002.

2.2.3.2. Espacio de Trabajo:

También denominado volumen de trabajo está formado por las posiciones dentro del espacio considerado, que son potencialmente alcanzables por el extremo del robot. Depende de la configuración geométrica del manipulador. Un punto del espacio se dice totalmente accesible si el efector final puede situarse en él, en todas las orientaciones que permita la constitución del manipulador y se dice parcialmente accesible si es accesible, pero no en todas las orientaciones posibles. La Figura 2.7 muestra el espacio de trabajo del robot KUKA KR 6-2.

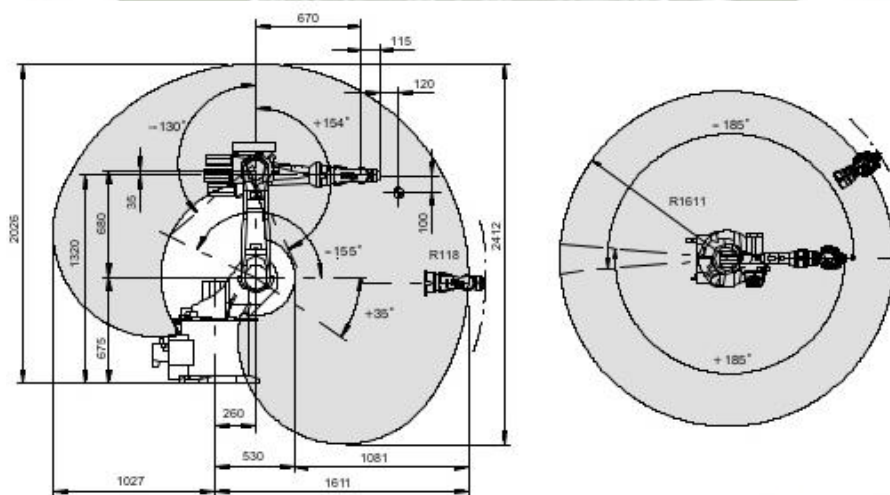


Figura 2.7: Espacio de trabajo del robot KUKA KR 6-2.

Fuente: "Estudio del robot KUKA KR 6-2I", Nelson A. Sierra 2013.

2.2.3.3. Materiales:

Dentro de los aspectos aun en desarrollo y en aras de encontrar la similitud con un brazo humano, se tiene el problema de encontrar nuevos materiales para el uso en robótica. Esto se traduce, en solucionar un compromiso entre “ligereza” y “resistencia”, que hoy ya se está encontrando con materiales rígidos que cumplen estas dos características.

Por lo general las características de los materiales que conforman elementos importantes como mecanismos, engranajes, etc., al estar sometidos a desgastes, será muy conveniente que estén fabricados con materiales bastante resistentes con la finalidad de tener una mayor precisión en el máximo tiempo posible de funcionamiento. Por lo que se destaca que un eslabón debe ser lo más ligero posible, ya que el peso de este da en muchos casos una carga para los otros elementos del robot.

En cuanto a los materiales a usar se tiene una amplia diversidad, desde madera, plásticos, gomas, cauchos hasta aleaciones metálicas complejas. Pero se puede considerar que a nivel comercial los más empleados para las estructuras son el aluminio y el acero, siendo el primero predominante por sus destacables características mecánicas respecto a la resistencia a curvatura, a la vez que es más ligero y posee mejor resistencia a la corrosión.

2.2.3.4. Transmisiones:

Estos son los elementos que transmiten el movimiento desde los actuadores hasta las articulaciones.

Se debe dar que un eficiente sistema de transmisión satisfaga una serie de características básicas: tener un tamaño y peso reducido, evitar que presente juegos u holguras considerables y tener transmisiones con gran rendimiento.

No existe un sistema de transmisión específico para robots, pero sí existen algunos que son usados con mayor frecuencia que otros y que se muestran ya clasificados en la Tabla 2.5. Esta clasificación se ha elaborado teniendo en cuenta el tipo de movimiento posible en la entrada y la salida, que puede ser lineal o circular. Es fundamental que el sistema de transmisión a ser usado no afecte al movimiento que transmite, sea por el rozamiento inherente a su funcionamiento o por las holguras producidas por desgaste. Incluso se tiene que tener en cuenta que el sistema de transmisión sea suficientemente apto para soportar un funcionamiento continuo a un par elevado, y entre grandes distancias.

Tabla 2.5: Sistemas de transmisión para robots.

Entrada - Salida	Denominación	Ventajas	Inconvenientes
Circular-Circular	Engranaje Correa dentada Cadena Paralelogramo Cable	Pares altos Distancia grande Distancia grande - -	Holguras - Ruido Giro limitado Deformabilidad
Circular-Lineal	Tornillo sin fin Cremallera	Poca holgura Holgura media	Rozamiento Rozamiento
Lineal-Circular	Paral. Articulado Cremallera	- Holgura media	Control difícil Rozamiento

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.2.3.5. Actuadores:

Los actuadores tienen por misión generar el movimiento de los elementos del robot según las órdenes dadas por una unidad de control.

Estos elementos utilizados en robótica son semejantes en funcionamiento a los usados en otras máquinas industriales, aunque constructivamente resultan más complejos a la hora de buscar que las relaciones potencia – tamaño y potencia – peso sean lo más grandes posibles con el fin de minimizar el momento de inercia de la estructura final.

Tabla 2.6: Características de distintos tipos de actuadores para robots.

	Neumáticos	Hidráulicos	Eléctricos
Energía	Aire a presión (5-10 bar)	Aceite mineral (50-100 bar)	Corriente eléctrica
Opciones	Cilindros Motor de paletas Motor de pistón	Cilindros Motor de paletas Motor de pistones axiales	Corriente continua Corriente alterna Motor paso a paso Servomotor
Ventajas	Baratos Rápidos Sencillos Robustos	Rápidos Alta relación potencia-peso Autolubricantes Alta capacidad de carga Estabilidad frente a cargas estáticas	Precisos Fiables Fácil control Sencilla instalación Silenciosos
Desventajas	Dificultad de control continuo Instalación especial (compresor, filtros) Ruidoso	Difícil mantenimiento Instalación especial (filtros, eliminación aire) Frecuentes fugas Caros	Potencia limitada

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

Los actuadores se clasifican en función del tipo de energía que emplean para producir movimiento: eléctrica, neumática o hidráulica. El empleo de un tipo de actuador u otro viene fundamentalmente dado por las características propias de ellos y por las

condiciones de trabajo requeridas, aunque como norma general, los más usados son los de tipo eléctrico por poseer características de control, sencillez y precisión. De esta manera la Tabla 2.6 muestra una comparación entre los distintos tipos de actuadores.

Según el tipo de corriente y el modo de funcionamiento empleado, los motores eléctricos, se pueden clasificar en: motores de corriente continua, motores de corriente alterna, motores paso a paso y servomotores, siendo el último uno de los más empleados en robótica, razón por la cual que en este apartado se llevará a cabo un desarrollo más detallado del mismo.

Los servomotores son dispositivos en los que puede controlarse la velocidad de funcionamiento y posición dentro de un rango de operación para llevar a cabo la acción requerida.

La palabra “servo” deriva de siervo, que fundamentalmente quiere decir que puede cumplir cualquier función que le sea programada desde un control maestro, teniendo siempre el control de la posición en la que se encuentra.

Los servomotores están compuestos esencialmente por tres elementos: motor eléctrico, encoder o sensor de posición que mide el desplazamiento articular y un amplificador electrónico, que es el encargado de acoplar y acondicionar todas las señales provenientes de un sistema mínimo digital.

Cualquier sistema servo básicamente establece comunicación por medio de pulsos eléctricos a través de un circuito de control y determina así su ángulo de posición. El servo espera recibir un pulso cada 20 milisegundos, en donde la longitud del pulso determinará los giros de motor; así un pulso de 1.5 ms., hará que el motor vaya a una posición de 90 grados, si el pulso es menor de 1.5 ms., entonces el motor se acercará a 0 grados y si el pulso es mayor de 1.5ms, el eje se moverá acercándose a los 180 grados. La lógica de control está en un programa que tiene la habilidad de completar la acción de una aplicación específica, monitoreando la posición del motor para comunicar así al servo la necesidad de moverse a la posición comandada.

Los servomotores también sustituyen a los accionamientos neumáticos e hidráulicos y son la alternativa de mejor desempeño frente a accionamientos con convertidores de frecuencia, dado que éstos no proporcionan control de posición y no son tan efectivos a bajas velocidades.

Entre las ventajas de este tipo de actuadores se tiene:

- Es muy robusto para su tamaño, pues a pesar de ser pequeño, posee altos torques.
- Potencia proporcional para cargas mecánicas.
- Tiene bajo consumo de energía, pues la magnitud de voltaje aplicado al servomotor es proporcional a la distancia de su desplazamiento. En este caso, si se desea que el eje requiera regresar una longitud amplia, el motor regresará a máxima velocidad, si sólo requerirá regresar un pequeño trayecto, el motor correrá a velocidad lenta. Esta manera de uso se le llama control proporcional, que emplea la energía necesaria sin desperdiciarla.

- Mayor precisión.
- Requieren menor mantenimiento porque es electrónico; pues a falta de fricción entre los elementos, el desgaste mecánico es nulo.

2.2.3.6. Subsistema Sensorial:

El robot como dispositivo mecánico puede posicionarse y orientarse en el espacio para ejecutar determinadas tareas. Ciertamente, para asegurar que esas tareas se realicen convenientemente es necesario efectuar un control de los movimientos a través de las posiciones y orientaciones que toma el robot en cada instante de tiempo. Para tal fin es necesario disponer de una variedad de sensores que hagan posible tener la información necesaria entre otras, la de posición, orientación, velocidad, etc. del robot. Por lo general, este tipo de sensores están ubicados en la misma estructura mecánica del robot, y por eso se les conoce como sensores internos.

Y cuando además al robot se les requiere dotar de cierta flexibilidad y autonomía de actuación, los sensores externos son los más precisos para reconocer e interpretar el ambiente de trabajo que le rodea y así poder ejecutar de forma correcta las labores propias encargadas. El término externo viene a que la información percibida es externa al robot, pudiendo estar situado físicamente el sensor en el mismo robot, o fuera de él. Estos sensores sólo están presentes cuando se requiera darle cierta inteligencia al robot.

2.2.3.7. Elementos Terminales:

Los elementos terminales, también llamados efectores finales son los encargados de interaccionar directamente con el entorno del robot. Si bien un mismo robot industrial es, dentro de unos límites lógicos, versátil y readaptable a una gran variedad de aplicaciones, no ocurre así con los elementos terminales, que son en muchos casos específicamente diseñados para cada tipo de trabajo.

Los elementos terminales se pueden clasificar en:

- Elementos de aprehensión que permiten sujetar y manipular objetos.
- Herramientas que permiten realizar operaciones sobre objetos.
- Otros dispositivos.

Entre los efectores finales correspondientes a esta tesis tendremos los elementos que permiten manipular objetos, como los elementos de aprehensión, que son componentes de sujeción denominados pinzas, que se utilizan para coger y dejar piezas en posiciones y orientaciones distintas. Es claro que el tipo de pinza debe estar en correspondencia con el tamaño, peso y forma del objeto a manipular. Existen numerosos tipos de elementos de sujeción que son diseñados particularmente para

determinadas aplicaciones y se pueden clasificar según el sistema de sujeción empleado, como se muestra en la Tabla 2.7

Tabla 2.7: Sistemas de Sujeción para Robots.

Tipos de Sujeción	Accionamiento	Uso
Pinza de Presión - desp. angular • desp. Lineal	Neumático o eléctrico	Transporte y manipulación de piezas sobre las que no importe presionar.
Pinza de enganche	Neumático o Eléctrico	Piezas de grandes dimensiones o sobre las que no se puede ejercer presión.
Ventosas de Vacío	Neumático	Cuerpos con superficie lisa poco porosa (cristal, plástico).
Electroimán	Eléctrico	Piezas ferromagnéticas

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

La pinza es una garra que dispone de dos o más dedos para sujetar objetos. Estos dedos no son articulados, por lo tanto, resultan ser simples desde el punto de vista constructivo. Existen dos configuraciones para el desplazamiento de los dedos, paralelos y angulares (Fig. 2.8)

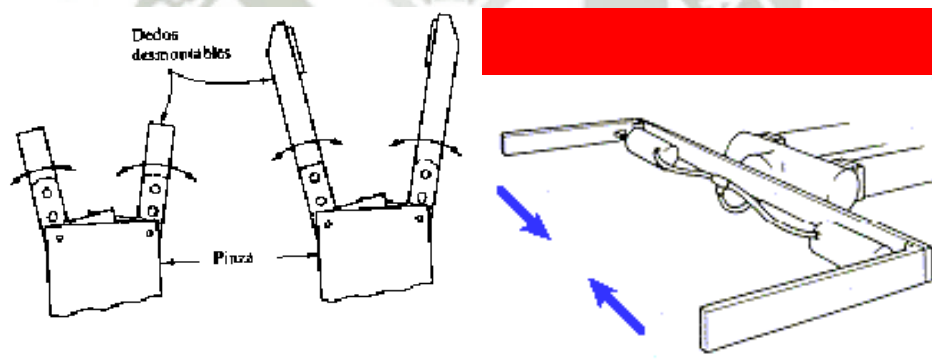


Figura 2.8: Configuraciones para el desplazamiento de los dedos de una pinza. De derecha a izquierda Angulares y Paralelos

Fuente: "Elementos terminales o actuadores finales", Victor. R. Gonzales 2002.

2.2.4. Herramientas Matemáticas para la localización espacial:

La manipulación de objetos realizada por un robot implica el movimiento espacial de su extremo, por lo tanto, es necesario conocer la posición y orientación de dicha pieza con respecto a la base del manipulador. Se tiene entonces la necesidad de contar con una variedad de herramientas matemáticas que permitan especificar estas características en el espacio.

Estas herramientas han de ser lo suficientemente robustas como para permitir establecer de forma sencilla relaciones espaciales entre distintos objetos y el manipulador.

2.2.4.1. Representación de la Posición (Coordenada Cartesianas):

Para ubicar un cuerpo rígido en el espacio es necesario contar con herramientas que permitan la localización espacial de sus puntos. En un plano se tiene dos grados de libertad, y consecuentemente la posición de un punto vendrá definida por dos componentes independientes. En el espacio tridimensional para esta localización, será necesario emplear tres componentes.

Cualquier sistema de referencia se define por medio de ejes perpendiculares entre sí y con un origen definido. La forma más intuitiva y utilizada de especificar la posición de un punto son las coordenadas cartesianas, que en el caso de trabajar en el plano (2 dimensiones), el sistema de referencia OXY correspondiente queda definido por dos vectores coordenados OX y OY perpendiculares entre sí con un punto de intersección común O.

En el caso de que se trabaje en tres dimensiones, un vector en este espacio se define con respecto al sistema de referencia OXYZ mediante las coordenadas correspondientes a cada uno de los ejes coordenados. Tal como se muestra en la Figura 2.9-b, el vector p estará definido por las componentes cartesianas (x,y,z) .

Hay otros métodos, igualmente válidos, y ampliamente extendidos, como son las coordenadas polares para dos dimensiones, y las cilíndricas y esféricas para espacios de tres dimensiones.

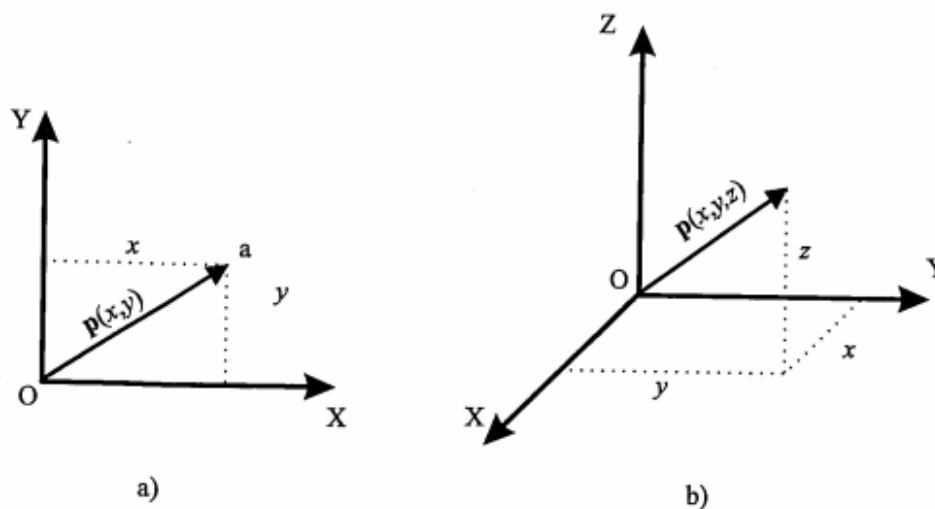


Figura 2.9: Representación de un vector en coordenadas cartesianas en 2 y 3 dimensiones.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.2.4.2. Representación de la Orientación:

Cualquier punto en el espacio queda totalmente definido a través de los datos de su posición. Para el caso de un sólido, es necesario además definir cuál es su orientación con respecto a un sistema de referencia. En el caso de un robot, no es suficiente con especificar cuál debe ser la posición de su extremo, sino que en general, es también necesario indicar su orientación.

Una orientación en el espacio tridimensional viene definida por tres grados de libertad o tres componentes que son desde el punto de vista matemático linealmente independientes. Se puede describir de forma sencilla la orientación de un objeto respecto a un sistema de referencia, asignando solidariamente al él, un nuevo sistema para luego estudiar la relación espacial existente entre los dos sistemas. De forma normal, esta relación vendrá dada por la posición y orientación del sistema asociado al objeto respecto al de referencia.

Existen diferentes métodos para representar la orientación de un elemento, entre ellos se pueden mencionar:

- **Matrices de Rotación**

Una forma de indicar la orientación de un sistema con respecto a otro es mediante la matriz de rotación.

Tomando en cuenta el libro Fundamentos de Robótica de Antonio Barrientos, explicamos este concepto.

En un espacio tridimensional, supóngase los sistemas OXYZ y OUVW, coincidentes en el origen, siendo el OXYZ el sistema de referencia fijo, y el OUVW el solidario al objeto cuya orientación se desea definir. Los vectores unitarios del sistema OXYZ serán $\mathbf{i}_x$, $\mathbf{j}_y$, $\mathbf{k}_z$, mientras que los del OUVW serán $\mathbf{i}_u$, $\mathbf{j}_v$, $\mathbf{k}_w$. Un vector $\mathbf{p}$ del espacio podrá ser referido a cualquiera de los sistemas de la siguiente manera:

$$\mathbf{p}_{uvw} = [p_u, p_v, p_w]^T = p_u \cdot \mathbf{i}_u + p_v \cdot \mathbf{j}_v + p_w \cdot \mathbf{k}_w \quad [2.1]$$

$$\mathbf{p}_{xyz} = [p_x, p_y, p_z]^T = p_x \cdot \mathbf{i}_x + p_y \cdot \mathbf{j}_y + p_z \cdot \mathbf{k}_z \quad [2.2]$$

Realizando una sencilla serie de transformaciones se puede llegar a la siguiente equivalencia:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = R \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} \quad [2.3]$$

Donde:

$$R = \begin{bmatrix} i_x i_u & i_x j_v & i_x k_w \\ j_y i_u & j_y j_v & j_y k_w \\ k_z i_u & k_z j_v & k_z k_w \end{bmatrix} \quad [2.4]$$

Es la matriz de rotación que define la orientación del sistema OUVW con respecto al sistema OXYZ. Recibe el nombre de matriz de cosenos directores y se trata de una matriz ortonormal, tal que la inversa de la matriz R es igual a su transpuesta.

La principal utilidad de esta matriz de rotación corresponde a la representación de la orientación de sistemas girados únicamente sobre uno de los ejes principales del sistema de referencia.

En caso de que el efector final del robot tenga una orientación compuesta por varias rotaciones continuas respecto a cada uno de los ejes indistintamente, se debe realizar el producto de matrices considerando el orden en que se realizan dichas rotaciones; puesto que esta operación no es conmutativa.

- **Ángulos de Euler:**

Para representar la orientación en el espacio tridimensional usando la matriz de rotación, es necesario definir nueve elementos. Aunque la utilización de las matrices de rotación presente múltiples ventajas, existen otros métodos para definir la orientación que hacen únicamente uso de tres componentes para su descripción. Este es el caso de los llamados ángulos de Euler.

Todo sistema OUVW que es solidario al cuerpo cuya orientación se desea describir, puede definirse con respecto al sistema OXYZ mediante tres ángulos: ϕ , Θ , ψ , llamados ángulos de Euler. Para obtener el sistema OUVW se gira sucesivamente el sistema OXYZ sobre unos ejes determinados de un triedro ortonormal los valores de ϕ , Θ , ψ . Es necesario, por consiguiente, conocer además de los valores de estos ángulos, cuáles son los ejes sobre los que se realizan los giros. Existen múltiples posibilidades, siendo solo tres las más usuales:

- Ángulos de Euler ZXZ
- Ángulos de Euler ZYZ
- Roll, pitch and yaw (alabeo, cabeceo y guiñada)

2.2.4.3. Matrices de Transformación Homogénea (MTH):

En los párrafos anteriores se han mencionado distintos métodos de representar la posición o la orientación de un sólido en el espacio, pero ninguno de estos métodos por sí solo permite una representación conjunta de ambos (localización). Para solucionar dicho problema se introducen las denominadas coordenadas homogéneas.

Tomando en cuenta el libro Fundamentos de Robótica de Antonio Barrientos, explicamos el concepto de Matrices de Transformación Homogénea.

La representación mediante coordenadas homogéneas de la localización de sólidos en un espacio n-dimensional se realiza a través de coordenadas de un espacio (n+1)-dimensional. Entonces una matriz de transformación homogénea que definida como una matriz de dimensión 4x4 representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

$$T = \begin{bmatrix} R_{3 \times 3} & p_{3 \times 1} \\ f_{1 \times 3} & w_{1 \times 1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} \quad [2.5]$$

La matriz T de transformación se suele escribir de la siguiente forma:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.6]$$

Donde n,o,a es una terna ortonormal que representa la orientación y P es un vector que representa la posición. Es decir, el vector columna n representa las coordenadas del eje O'U del sistema O'UVW con respecto del sistema OXYZ, el vector columna o representa las coordenadas del eje O'V del sistema O'UVW con respecto del sistema OXYZ, y que el vector columna a representa las coordenadas del eje O'W del sistema O'UVW con respecto del sistema OXYZ.

2.2.4.4. Relación entre ángulos de Euler y MTH:

Los métodos nombrados para la representación espacial son equivalentes, lo que quiere decir, que expresan lo mismo de forma distinta, por lo tanto, existe un modo de pasar de un tipo de representación a otro.

Como se ha mencionado, los ángulos de Euler sólo son capaces de realizar una representación de la orientación. Por esta razón, al momento de obtener la matriz homogénea equivalente a un conjunto de ángulos de Euler dados, únicamente quedará definida la submatriz de rotación $R_{3 \times 3}$. La obtención de la matriz homogénea correspondiente a cada conjunto de ángulos de Euler es inmediata, bastará con componer las matrices que representan las rotaciones que definen los propios ángulos:

- **Sistema ZXZ**

Este sistema responde, a la composición de la siguiente secuencia de rotaciones:

$$T_{ZXZ} = T(z, \varphi)T(u, \theta)T(w, \psi) \quad [2.7]$$

Que desarrollado en forma matricial:

$$T_{ZXZ} = \begin{bmatrix} C\varphi C\psi - S\varphi C\theta S\psi & -C\varphi S\psi - S\varphi C\theta C\psi & S\varphi S\theta & 0 \\ S\varphi C\psi + C\varphi C\theta S\psi & -S\varphi S\psi + C\varphi C\theta C\psi & -C\varphi S\theta & 0 \\ S\theta S\psi & S\theta C\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.8]$$

- **Sistema ZYZ**

El paso del sistema fijo al girado se hace realizando la siguiente secuencia de rotaciones:

$$T_{ZYZ} = T(z, \varphi)T(v, \theta)T(w, \psi) \quad [2.9]$$

Que desarrollado en forma matricial:

$$T_{ZYZ} = \begin{bmatrix} C\varphi C\theta C\psi - S\varphi S\psi & -C\varphi C\theta S\psi - S\varphi C\psi & C\varphi S\theta & 0 \\ S\varphi C\theta C\psi + C\varphi S\psi & -S\varphi C\theta S\psi + C\varphi C\psi & S\varphi S\theta & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.10]$$

- **Roll-Pitch-Yaw**

De igual forma que en los casos anteriores, estos ángulos de Euler se pueden representar mediante la concatenación de las siguientes rotaciones:

$$T_{XYZ} = T(z, \varphi)T(y, \theta)T(x, \psi) \quad [2.11]$$

Y de forma matricial:

$$T_{XYZ} = \begin{bmatrix} C\varphi C\theta & C\varphi S\theta S\psi - S\varphi C\psi & C\varphi S\theta C\psi - S\varphi S\psi\theta & 0 \\ S\varphi C\theta & S\varphi S\theta S\psi + C\varphi C\psi & S\varphi S\theta C\psi - C\varphi S\psi\theta & 0 \\ -S\theta & C\theta S\psi & C\theta C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.12]$$

2.2.5. Cinemática del Robot:

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia. Así, la cinemática aborda la descripción analítica del movimiento espacial del robot como una función temporal, y en particular por la relación que hay entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Se dan dos problemas primordiales a solucionar en la cinemática del robot (Fig. 2.10), el primero se conoce como el problema cinemático directo, que consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot. El segundo denominado problema cinemático inverso que da solución a la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas.

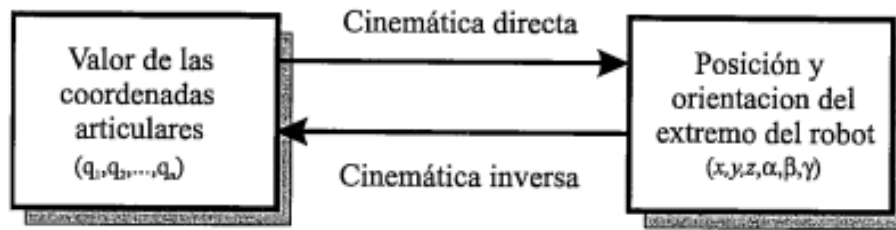


Figura 2.10: Diagrama de Relación entre cinemática directa e inversa.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

Así también, la cinemática del robot trata de encontrar las relaciones existentes entre velocidades del movimiento de las articulaciones con las del extremo. Esta relación viene dada por el modelo diferencial expresado mediante la matriz Jacobiana.

2.2.5.1. Cinemática Directa:

Podemos definir a un robot como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, en el que se puede establecer un sistema de referencia fijo situado en la base del mismo y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia utilizando herramientas matemáticas como el álgebra vectorial y matricial. Por lo que, el problema cinemático directo se reduce a hallar una matriz de transformación homogénea, que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo ubicado en la base del mismo. Esta matriz estará en función de las coordenadas articulares.

Así, si se han escogido coordenadas cartesianas y ángulos de Euler para representar la posición y orientación del extremo de un robot de seis grados de libertad, la solución al problema cinemático directo vendrá dada por las relaciones:

$$x = f_x(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.13]$$

$$y = f_y(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.14]$$

$$z = f_z(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.15]$$

$$\varphi = f_\varphi(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.16]$$

$$\theta = f_\theta(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.17]$$

$$\psi = f_\psi(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.18]$$

La obtención de estas relaciones no es en general complicada, siendo incluso en robots de pocos grados de libertad, fácil de encontrar mediante simples consideraciones geométricas.

Para robots de más grados de libertad puede proponerse un método seguro basado en la utilización de las matrices de transformación homogénea, que consiste en asociar un

sistema de referencia solidario a cada eslabón del robot y, utilizando las transformaciones homogéneas, representar las rotaciones y traslaciones relativas entre los distintos eslabones que lo componen.

Para detallar la relación que existe entre dos elementos contiguos, se puede hacer uso de cualquier sistema de referencia asignado a cada elemento; sin embargo, la forma más común que se suele utilizar en robótica es la representación de Denavit-Hartenberg.

Denavit y Hartenberg propusieron en 1955 un método matricial que permite establecer de manera sistemática un sistema de coordenadas (S_i) ligado a cada eslabón i de una cadena articulada, pudiéndose determinar luego las ecuaciones cinemáticas de la cadena completa.

Según la representación de D-H, escogiendo adecuadamente los sistemas de coordenadas asociados a cada eslabón, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón:

- Rotación alrededor del eje z_{i-1} un ángulo θ_i .
- Traslación a lo largo de z_{i-1} una distancia d_i ; vector $d_i(0,0,d_i)$.
- Traslación a lo largo de x_i una distancia a_i ; vector $a_i(0,0,a_i)$.
- Rotación alrededor del eje x_i , un ángulo α_i .

Después de realizado el producto de las transformaciones en el orden especificado, se obtiene como resultado la matriz ${}^{i-1}A_i$ definida en la ecuación 2.19.

$${}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.19]$$

Donde θ_i , a_i , d_i , α_i , son los parámetros D-H del eslabón i . De este modo, basta con identificar dichos parámetros para obtener matrices ${}^{i-1}A_i$ y relacionar así todos y cada uno de los eslabones del robot.

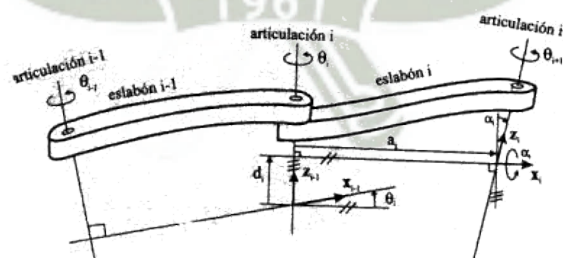


Figura 2.11: Parámetros D-H para un eslabón giratorio.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

Los cuatro parámetros de D-H (θ_i , d_i , a_i , α_i) dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente. En concreto ellos representan (Fig. 2.11):

θ_i : Es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_{i-1} , utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

d_i : Es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas (i-1)-esimo hasta la intersección del eje z_{i-1} con el eje x_i . Se trata de un parámetro variable en articulaciones prismáticas.

a_i : Es a la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-esimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes z_{i-1} y z_i .

α_i : Es el ángulo de separación del eje z_{i-1} y el eje z_i , medido en un plano perpendicular al eje x_i , utilizando la regla de la mano derecha.

Una vez obtenidos los parámetros DH, el cálculo de las relaciones entre los eslabones consecutivos del robot es inmediato, ya que vienen dadas por las matrices ${}^{i-1}A_i$, que se calcula según la expresión general [2.19]. Las relaciones entre eslabones no consecutivos vienen dadas por las matrices T que se obtienen como producto de un conjunto de matrices ${}^{i-1}A_i$.

Obtenida la matriz T, ésta expresará la orientación y posición del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

2.2.5.2. Cinemática Inversa:

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = [q_1, q_2, \dots, q_n]^T$ para que su extremo se posicione y oriente según una determinada localización espacial.

A diferencia de la cinemática directa, en el problema cinemático inverso, el procedimiento de obtención de las ecuaciones es fuertemente dependiente de la configuración del robot.

Se han desarrollado procedimientos genéricos que pueden ser programados, de modo que un computador pueda, a partir del conocimiento de la cinemática del robot (con sus parámetros de DH) obtener la n-upla de valores articulares que posicionan y orientan su extremo. Estos procedimientos son métodos numéricos iterativos, cuya velocidad de convergencia e incluso su convergencia en si no está siempre garantizada por lo que será un inconveniente de cálculo.

A la hora de resolver el problema cinemático inverso es mucho más adecuado encontrar una solución cerrada. Esto es, encontrar una relación matemática explícita de la forma:

$$q_k = f_k(x, y, z, \alpha, \beta, \gamma) \quad k = 1 \dots n \text{ (GDL)} \quad [2.20]$$

Este tipo de solución presenta, entre otras, las siguientes ventajas:

- En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real. Una solución de tipo iterativo no garantiza tener la solución en el momento adecuado.
- Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única; existiendo diferentes n-uplas $q = [q_1, \dots, q_n]^T$ que posicionan y orientan el extremo del robot de mismo modo. En estos casos una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada de entre las posibles.

Pese a las dificultades mencionadas, la mayoría de los robots tienen cinemáticas relativamente simples que facilitaran en cierta medida la resolución de su problema cinemático inverso.

Existen diferentes métodos para dar solución a la cinemática inversa de un robot, dentro de los cuales es posible identificar:

- **Métodos Geométricos:**

Este procedimiento es apropiado para robots que tienen pocos grados de libertad o para el caso de que se consideren solo los primeros grados de libertad, dedicados a posicionar el extremo. El método se basa en hallar relaciones trigonométricas y geométricas en donde intervienen las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos. Usualmente, se recurre a la forma de resolución de triángulos formados por los elementos y articulaciones del robot.

- **Matrices de Transformación Homogénea:**

Es posible tratar de obtener el modelo cinemático inverso de un robot a partir del conocimiento del modelo directo del mismo mediante transformaciones homogéneas.

Si suponiendo conocidas las relaciones que expresan el valor de la posición y orientación del extremo del robot en función de sus coordenadas articulares, obtener por manipulación de aquéllas las relaciones inversas.

Sin embargo, en la práctica esta tarea no es común siendo en muchas ocasiones tan compleja que obliga a desecharla.

- **Desacoplo Cinemático:**

Los métodos geométricos o los procedimientos basados en matrices de transformación homogénea permiten obtener los valores de las tres primeras variables articulares del robot, aquellas que posicionan su extremo en unas coordenadas (p_x, p_y, p_z) determinadas, aunque pueden ser igualmente utilizadas para la obtención de las 6 a costa de una mayor complejidad.

Ahora bien, como se sabe, en general no basta con posicionar el extremo del robot en un punto del espacio, sino que siempre es preciso también conseguir que la herramienta que aquel maneja se oriente de una manera definida. Para esto, los robots cuentan con otros tres grados de libertad adicionales, situados al final de la cadena cinemática y cuyos ejes, generalmente, se cortan en un punto, denominado muñeca del robot. Si bien la variación de estos tres últimos grados de libertad origina un cambio en la posición final del extremo real del robot, su verdadero propósito es orientar la herramienta libremente en el espacio.

El método de desacoplo cinemático obtiene ventaja de este hecho, separando ambos problemas: Posición y orientación; y así realizar su resolución de manera independiente y por cualquier procedimiento. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los 3 últimos ejes (muñeca del robot) calculándose los valores de las tres primeras variables articulares (q_1, q_2, q_3) que consiguen posicionar este punto. A continuación, a partir de los datos de orientación y de los ya calculados (q_1, q_2, q_3) obtiene los valores del resto de las variables articulares.

2.2.5.3. Modelo Diferencial:

El modelamiento cinemático de un robot busca las relaciones entre las variables articulares y la posición y orientación del extremo del robot. En esta relación no se tienen en cuenta las fuerzas o pares que actúan sobre el robot (actuadores, cargas, fricciones, etc.) y que pueden originar el movimiento del mismo. Sin embargo, sí debe permitir conocer, además de la relación entre las coordenadas articulares y del extremo, la relación entre sus respectivas derivadas. Así, el sistema de control del robot debe establecer que velocidades debe imprimir a cada articulación, a través de sus respectivos actuadores, para conseguir que el extremo desarrolle una trayectoria temporal concreta.

El modelo diferencial queda concretado en la denominada matriz Jacobiana, la misma que relaciona el vector de velocidades articulares $(\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)$ con otro vector de velocidades expresado en un espacio distinto. Existen diferentes posibilidades a la hora de seleccionar este espacio. Una primera elección es la de considerar la relación con las velocidades de la localización del extremo del robot, siendo esta la posición y orientación $(\dot{x}, \dot{y}, \dot{z}, \dot{\varphi}, \dot{\theta}, \dot{\psi})$. Esta relación viene dada por la denominada Jacobiana Analítica.

Una segunda elección es relacionar las velocidades articulares, con los vectores de velocidad lineal y angular ($v_x, v_y, v_z, w_x, w_y, w_z$) con que se mueve el extremo del robot, expresados en un sistema de referencia determinado. La relación entre ambas velocidades (articulares y lineal-angular del extremo) se obtiene a través de la denominada matriz Jacobiana geométrica.

La matriz Jacobiana directa posibilita conocer una expresión de las velocidades del extremo del robot a partir de los valores de las velocidades de cada articulación. Y, la matriz Jacobiana inversa permitirá conocer las velocidades articulares necesarias para obtener unas velocidades determinadas en el extremo del robot.

- **Jacobiana Analítica:**

La Jacobiana analítica relaciona el vector de velocidades articulares ($\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dots, \dot{q}_n$) con las velocidades de ubicación y orientación del efector final del robot, expresadas en base de coordenadas cartesianas y ángulos de Euler ($\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}$).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = J_a \cdot \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad [2.21]$$

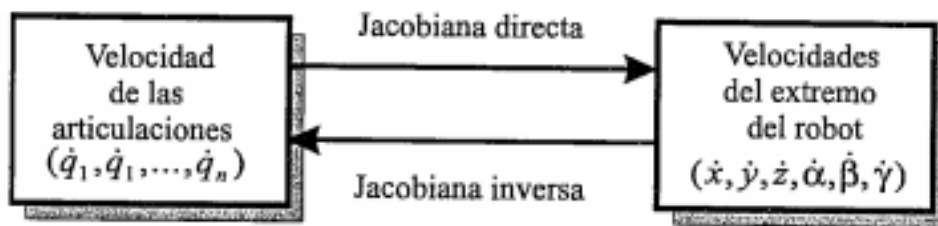


Figura 2.12: Matriz jacobiana directa e inversa.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

El método más directo para determinar la matriz Jacobiana Analítica, consiste en derivar las ecuaciones correspondientes al modelo cinemático directo del manipulador.

- **Jacobiana Geométrica**

Entre otras relaciones, interesa la que se establece entre velocidades articulares ($\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n$) y la velocidad lineal y angular del extremo del robot ($v_x, v_y, v_z, w_x, w_y, w_z$), expresadas normalmente con respecto a la base del robot. En este caso la matriz de velocidades se le llama Jacobiana geométrica.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_g \cdot \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad [2.22]$$

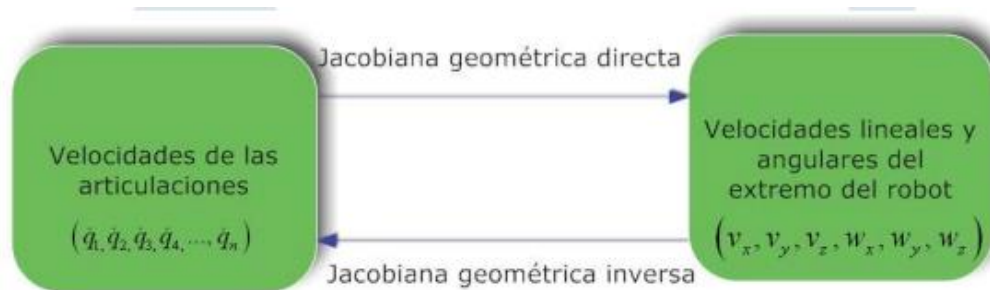


Figura 2.13: Matriz jacobiana geométrica directa e inversa.

Fuente: "Curso - Robótica", UNAD, Freddy F. Valderrama

La deducción de esta matriz es menos directa que la Jacobiana Analítica. Existen diferentes procedimientos que permiten la obtención numérica de la Jacobiana partiendo de la información que hay en las matrices que definen el modelo cinemático.

- **Jacobiana Inversa**

Del mismo modo que se ha obtenido la relación directa que permite hallar las velocidades del extremo partiendo de las velocidades articulares, puede también lograrse la relación inversa, que permite calcular las velocidades articulares partiendo de las del extremo. En la obtención de la relación inversa pueden emplearse diferentes procedimientos, alternativas.

Como primera alternativa, tenemos que suponiendo conocida la relación directa, dada por la matriz Jacobiana [2.21] y [2.22], se obtiene la relación inversa invirtiendo simbólicamente la matriz.

$$\begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_a^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} \quad ; \quad \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_g^{-1} \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \quad [2.23]$$

Esta alternativa, es en la práctica difícil de realizarla; pues al suponer que la matriz J sea cuadrada, la inversión simbólica de una matriz 6x6, cuyos elementos son funciones trigonométricas, es muy complejo.

Como la segunda alternativa se plantea la evaluación numérica de la matriz J para una configuración (q_i) específica del robot, luego invirtiendo numéricamente esta matriz encontrar la relación inversa válida para esta configuración. En este caso hay que considerar, en primer lugar, que el valor numérico de la Jacobiana va

cambiando a medida que el robot se mueve y, por lo tanto, la Jacobiana inversa ha de ser recalculada dinámicamente, es decir constantemente. Además, pueden existir n-uplas $(q_1, \dots, q_n)$ para las cuales la matriz Jacobiana J no sea invertible por ser su determinante nulo. Estas configuraciones del robot en las que el Jacobiano se anula se denominan configuraciones singulares. Una dificultad que puede surgir con este y otros procedimientos de cómputo de la matriz Jacobiana inversa, se deriva de la circunstancia de que la matriz J no sea cuadrada, teniendo, así como alternativa de solución el uso de la matriz Jacobiana Pseudoinversa. Cuando el número de grados de libertad del robot no coincide con la dimensión del espacio de la tarea, la matriz Jacobiana tendrá más filas que columnas; por lo tanto, será necesario utilizar la Matriz Jacobiana Pseudoinversa por la izquierda (ecuación 2.24). Sin embargo, en los casos en el que el robot sea redundante; es decir que la matriz J , tenga mayor número de columnas que de filas, será preciso utilizar la matriz Jacobiana Pseudoinversa por la derecha, definida en la ecuación 2.25.

$$J_i^+ = (J^T \cdot J)^{-1} \cdot J^T \quad [2.24]$$

$$J_i^+ = J^T \cdot (J \cdot J^T)^{-1} \quad [2.25]$$

La tercera alternativa para obtener la matriz Jacobiana inversa es repetir el procedimiento seguido por la obtención de la Jacobiana directa, pero ahora partiendo del modelo cinemático inverso. Como en el caso de la primera alternativa, este método puede ser algebraicamente complicado.

- **Configuraciones Singulares**

Se denominan configuraciones singulares de un robot a aquéllas en las que el determinante de su matriz Jacobiana se anula. Por esta circunstancia, en las configuraciones singulares no existe Jacobiana inversa. Al anularse el Jacobiano, un incremento infinitesimal de las coordenadas cartesianas supondría un incremento infinito de las coordenadas articulares, lo que en la práctica se traduce en que, en las inmediaciones de las configuraciones singulares, el pretender que el extremo del robot se mueva a velocidad constante, obligaría a movimientos de las articulaciones a velocidades no admitidas por los actuadores. Por ello, alrededor de las configuraciones singulares se pierde alguno de los grados de libertad del robot, siendo imposible que su extremo se mueva en una determinada dirección cartesiana.

Las configuraciones singulares del robot pueden ser clasificadas como:

- **Singularidades en los límites del espacio de trabajo del robot:**

Se dan cuando el extremo del robot está en algún punto del límite de trabajo interior o exterior. En esta situación resulta evidente que el robot no podrá desplazarse en las direcciones que lo alejan de este espacio de trabajo.

- **Singularidades en el interior del espacio de trabajo del robot:**

Sucede dentro de la zona de trabajo y se producen, generalmente, por el alineamiento de dos o más ejes de las articulaciones del robot.

2.2.6. Dinámica del Robot:

La dinámica se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. Por tanto, el modelo dinámico de un robot tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo. Esta relación se obtiene mediante el denominado modelo dinámico, que establece la relación matemática entre:

- La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.
- Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot)
- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La obtención de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el número de grados de libertad aumenta, el planteamiento y obtención del modelo dinámico se complica enormemente. El modelo dinámico debe ser resuelto de manera iterativa mediante la utilización de un procedimiento numérico.

El problema de la obtención del modelo dinámico de un robot es, por tanto, uno de los aspectos más complejos de la robótica, lo que ha llevado a ser obviado en numerosas ocasiones. Sin embargo, el modelo dinámico es imprescindible para conseguir los siguientes fines:

- Simulación del movimiento del robot.
- Diseño y evaluación de la estructura mecánica del robot.
- Dimensionamiento de los actuadores.
- Diseño y evaluación del control dinámico del robot.

Este último fin es evidentemente de gran importancia, pues de la calidad del control dinámico del robot depende la precisión y velocidad de sus movimientos. La gran complejidad ya comentada existente en la obtención del modelo dinámico del robot ha motivado que se realicen ciertas simplificaciones, de manera que pueda así ser utilizado no sólo en el diseño del controlador, sino también en línea con el control, cuando así lo requiera la técnica de control empleada.

2.2.6.1. Formulación Lagrange – Euler:

La formulación Lagrangiana permite describir la dinámica de un robot a partir de un balance de energía. Desde este punto de vista el robot se considera como una caja negra. Las ecuaciones tienen únicamente en cuenta la energía almacenada, que se expresa en términos de energía cinética y potencial.

Más concretamente, el Lagrangiano es una función escalar que se define como la diferencia entre las energías cinética y potencial de un sistema mecánico, en función de las llamadas coordenadas generalizadas que en el caso de un robot son las variables articulares:

$$\mathcal{L}(q, \dot{q}) = E_C(q, \dot{q}) - E_P(q) \quad [2.26]$$

Donde $\mathcal{L}$ es el Lagrangiano, E_C la energía cinética, y E_P la energía potencial.

A partir de la ecuación [2.26] se puede definir las ecuaciones del movimiento para un robot:

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}} \right) - \frac{\partial \mathcal{L}}{\partial q} = \tau \quad [2.27]$$

Donde τ es el vector correspondiente a los pares de los actuadores, es decir, un vector (nx1) formado por los pares de los actuadores de las articulaciones rotacionales y por las fuerzas de los actuadores de las articulaciones prismáticas, siendo n el número de articulaciones del robot.

Teniendo en cuenta la ecuación [2.26], se obtiene para la expresión [2.27]:

$$\frac{d}{dt} \left(\frac{\partial E_C - \partial E_P}{\partial \dot{q}} \right) - \frac{\partial E_C}{\partial q} + \frac{\partial E_P}{\partial q} = \tau \quad [2.28]$$

- **Energía Cinética:**

La energía cinética total del robot es la suma de las energías cinéticas de los n eslabones que integran la cadena cinemática:

$$E_C = \sum_{i=1}^n E_{C_i} \quad [2.29]$$

Donde la energía cinética de un eslabón i es la suma de dos términos, el primero debido a la velocidad lineal, y el segundo a la velocidad angular del eslabón:

$$E_{C_i} = \frac{1}{2} m_i v_{C_i}^C v_{C_i}^C + \frac{1}{2} w_i^C I_i^C w_i^C \quad [2.30]$$

Donde:

m_i = masa del eslabón i.

$v_{C_i}^C$ = velocidad lineal del centro de masas del eslabón i con relación al sistema de referencia situado en el centro de masas.

w_i^C = velocidad angular del eslabón i con relación al sistema de referencia situado en el centro de masas.

I_i^C = tensor de inercia del eslabón i respecto al sistema de referencia situado en el centro de masas.

Mientras que la velocidad angular es la misma para todos los puntos de un eslabón, no se puede decir lo mismo para la velocidad lineal, que resulta distinta en cada punto del eslabón, por ello se considera la velocidad del centro de masas del eslabón, v_{C_i} .

Queda por determinar el sistema de referencia en la ecuación [2.30]. La forma más cómoda para calcular la energía cinética es hacerlo respecto a un sistema de referencia posicionado en el centro de masa del eslabón. Como el factor debido a la velocidad angular es independiente del sistema de referencia con respecto al que se calcule, se toma el que resulte más cómodo para el cálculo de la energía cinética, el posicionado en el centro de masas del eslabón. El tensor de inercia del eslabón también puede ser expresado respecto al mismo sistema de referencia.

Puesto que las velocidades lineales v_{C_i} y angulares w_i de la ecuación [2.30] son función de las variables articulares q y las velocidades articulares $\dot{q}$, la ecuación [2.30] que representa la energía cinética queda como $E_C(q, \dot{q})$, y teniendo en cuenta la matriz Jacobiana, resulta:

$$E_C = \frac{1}{2} \dot{q}^T \sum_{i=1}^n \left[m_i \cdot J_{v_{C_i}}^T(q) \cdot J_{v_{C_i}}(q) + J_{w_i}^T(q) \cdot I_i^C \cdot J_{w_i}(q) \right] \dot{q} \quad [2.31]$$

Que se puede expresar de forma matricial como:

$$E_C = \frac{1}{2} \dot{q}^T D(q) \dot{q} \quad [2.32]$$

- **Energía Potencial**

Por otro lado, la energía potencial total almacenada en el robot E_P se corresponde con la suma de las energías potenciales almacenadas por cada uno de los n eslabones que integran la cadena cinemática:

$$E_P = \sum_{i=1}^n E_{P_i} \quad [2.33]$$

Donde la energía potencial de un eslabón i es:

$$E_{P_i} = -m_i g^T p_{C_i}^M \quad [2.34]$$

Donde:

m_i = masa del eslabón i .

g = vector gravedad.

$p_{C_i}^M$ = es el vector que localiza el centro de masas del eslabón i respecto al sistema de referencia del mundo M .

Puesto que el vector $p_{C_i}^M$ es función de las variables articulares q , la ecuación [2.33] que representa la energía potencial queda como:

$$E_P = E_P(q) \quad [2.35]$$

- **Ecuaciones del Movimiento**

Aplicando las expresiones [2.32] y [2.34] a la ecuación [2.26], se tiene:

$$\mathcal{L}(q, \dot{q}) = \frac{1}{2} \dot{q}^T D(q) \dot{q} - E_P(q) \quad [2.36]$$

Y para obtener las ecuaciones del movimiento según la expresión [2.28], es necesario efectuar previamente las derivadas correspondientes, y finalmente obtener el modelo matricial:

$$D(q) \cdot \ddot{q} + C(q, \dot{q}) \cdot \dot{q} + g(q) = \tau \quad [2.37]$$

En esta última ecuación, el término $D(q)$ es el inercial del sistema, $C(q, \dot{q})$ es el término de Coriolis y $g(q)$ es el efecto de la gravedad.

2.3. Herramientas Software a utilizar:

2.3.1. Solidwork:

SolidWorks es un software CAD (diseño asistido por computadora) para modelado mecánico en 3D, desarrollado en la actualidad por SolidWorks Corp., una filial de Dassault Systèmes, S.A. (Suresnes, Francia), para el sistema operativo Microsoft Windows. Su primera versión fue lanzada al mercado en 1995 con el propósito de hacer la tecnología CAD más accesible. El programa permite modelar piezas y conjuntos y extraer de ellos tanto planos técnicos como otro tipo de información necesaria para la producción. Es un programa que funciona con base en las nuevas técnicas de modelado con sistemas CAD. El proceso consiste en traspasar la idea mental del diseñador al sistema CAD, "construyendo virtualmente" la pieza o conjunto. Posteriormente todas las extracciones (planos y ficheros de intercambio) se realizan de manera bastante automatizada.

La mayoría de la gente piensa en el software CAD mecánico (MCAD) en 3D. Y, aunque puede que sea por lo que más se conoce, conviene recordar que SolidWorks es mucho más. Se tienen 2 pilares que avalan este producto y que tomamos en cuenta para su elección:

- **Productividad:**

Cuenta con un diseño 3D centrado en la innovación, inteligencia integrada que acelera el proceso de diseño, colaboración mejorada que aporta nuevas ideas para su negocio y una disminución del coste total de propiedad con una mayor eficiencia de procesos.

- **Capacidad:**

Permite la creación de dibujos en 2D más rápidamente con actualizaciones automáticas para obtener una precisión del 100 %, aporta rapidez al diseño y garantiza la precisión con herramientas industriales especializadas, se pueden realizar pruebas en condiciones del mundo real y asegurarse que el producto es correcto antes de fabricarlo además ayuda a diseñar teniendo en cuenta el entorno para innovar y crecer

2.3.2. Matlab:

MATLAB es un entorno de cálculo técnico de altas prestaciones para cálculo numérico y visualización. Es un entorno fácil de usar, donde los problemas y las soluciones son expresados como se escriben matemáticamente, sin la programación tradicional. El nombre MATLAB proviene de "MATrix LABoratory" (Laboratorio de Matrices). MATLAB fue escrito originalmente para proporcionar un acceso sencillo al software matricial desarrollado por los proyectos LINPACK y EISPACK, que juntos representan lo más avanzado en programas de cálculo matricial.

MATLAB es un sistema interactivo cuyo elemento básico de datos es una matriz que no requiere dimensionamiento. Esto permite resolver muchos problemas numéricos en una fracción del tiempo que llevaría hacerlo en lenguajes como C, BASIC o FORTRAN. MATLAB ha evolucionado en los últimos años a partir de la colaboración de muchos usuarios. En entornos universitarios se ha convertido en la herramienta de enseñanza estándar para cursos de introducción en álgebra lineal aplicada, así como cursos avanzados en otras áreas. En la industria, MATLAB se utiliza para investigación y para resolver problemas prácticos de ingeniería y matemáticas, con un gran énfasis en aplicaciones de control y procesamiento de señales. MATLAB también proporciona una serie de soluciones específicas denominadas TOOLBOXES.

Entre los problemas particulares que se resuelven en este entorno tenemos los siguientes:

- Procesamiento de señales

- Diseño de sistemas de control
- Simulación de sistemas dinámicos
- Identificación de sistemas
- Redes neuronales y otros.

La decisión de elegir como programa de control al MATLAB dentro de nuestra tesis se basó principal en su capacidad de crecimiento, la cual permite al usuario convertirse en un autor contribuyente, creando propias aplicaciones mediante este entorno; y permitiéndonos llegar hasta donde la imaginación nos lleve, lo cual no nos ponga limite.

2.3.3. Repetier Host:

Repetier Host es un software que permite la realización de impresiones 3D mediante la configuración de parámetros específicos, teniendo las siguientes características:

- **Soporte multiextrusor:**

Repetier Host puede manejar hasta 16 extrusoras con diferentes tipos de filamentos y colores simultáneamente y visualizar el resultado con los colores de filamentos individuales, para que pueda ver el resultado antes de imprimir.

- **Soporte multicorte:**

Repetier Host incluye directamente 3 máquinas de cortar diferentes: Slic3r, CuraEngine y Skeinforge Si eso no es suficiente, se puede utilizar cualquier otra máquina de cortar que quiera con Repetier -Host. Después de cortar se puede ver todos los movimientos de la extrusora y estructuras de apoyo. Usted controla todo el modelo, un área en particular o capas individuales y reconoce los problemas potenciales antes de la impresión, esto ahorra mucho tiempo y dinero.

- **Impresión fácil de varias partes:**

Puede cargar o duplicar el mayor número de modelos que se ajustan en su plato de construcción e imprimirlos de forma simultánea, lo que permite el uso eficiente de la placa de construcción ahorrando así mucho tiempo. Se puede organizar de forma manual o simplemente haciendo clic en el botón de Posición Automática y Repetier hace el resto.

- **Obtener el control total sobre sus impresiones:**

Repetier - Host es fácil de usar, incluso para los principiantes, pero le da un control completo a su proceso de impresión, máquina de cortar y la impresora de ser necesario. Esta es la razón por la cual se puede imprimir cualquier filamento en perfecta calidad como PLA, ABS, PETG. Las compañías de biotecnología incluso pueden imprimir órganos con Repetier -Host.

2.4. Teleoperación:

Desde hace varios años atrás, el hombre ha venido utilizando distintas herramientas para poder aumentar el alcance de su capacidad de manipulación. A mediados del siglo pasado se vio la necesidad de tener que manipular materiales con propiedades radioactivas, y fue entonces cuando se comenzaron a desarrollar dispositivos complejos para la manipulación a distancia. Estos desarrollos desembocaron, finalmente, en lo que se conoce como sistemas de teleoperación maestro-esclavo, en los que un manipulador denominado esclavo reproduce fielmente los movimientos marcados por un dispositivo o manipulador maestro, controlado a su vez manualmente por un operador humano.

De forma general, la teleoperación comprende todas aquellas tecnologías que permiten a un ser humano operar a distancia, con aplicación específica a la realización de tareas impredecibles y/o no repetitivas en ambientes hostiles y/o inaccesibles.

En los últimos años, se ha experimentado un renovado interés en las tecnologías de teleoperación y al mismo tiempo una apertura a nuevos campos de aplicación, como el espacial o la cirugía, diferentes del nuclear y submarino, donde la teleoperación tuvo su origen y evolución principal durante muchos años. Esto se ha debido a que poco a poco, con la ayuda de nuevos desarrollos en computadores digitales y la integración de tecnologías propias de la robótica y la inteligencia artificial, las prestaciones de los sistemas de teleoperación han aumentado considerablemente. Se han desarrollado, además, nuevos elementos que mejoran la eficiencia de la manipulación, como pueden ser novedosos diseños de dispositivos de entrada y nuevas técnicas de control.

La teleoperación, por tanto, puede considerarse hoy en día como una tecnología madura que forma parte de la robótica, aunque sus orígenes fueran bien distintos. En ambas tecnologías el objetivo fundamental es el realizar el control de un brazo articulado de varios grados de libertad, con la diferencia que en la teleoperación es el operador, y no un programa, el que comanda en todo momento al manipulador, con la posibilidad de que existan diversos modos de acoplamiento entre ambos.

Es así que se crea el término telerobótica y se define como un conjunto de tecnologías que comprenden la monitorización y reprogramación a distancia de un robot por un ser humano. Se tendrá entonces la teleoperación de un robot, que se denominará *telerobot* o *robot teleoperado*.

2.4.1. Elementos y Arquitectura de un Sistema de Teleoperación:

Si en la robótica industrial el principal elemento es el propio robot con su armario de control, un sistema de teleoperación cuenta además del sistema teleoperado, ya sea manipulador o robot móvil, con otros elementos igualmente necesarios. Además, no existe una única configuración posible de esos elementos dentro del sistema de teleoperación, y su interrelación puede ser de muy distinta naturaleza. Es decir, que, dentro de una arquitectura general, pueden existir diversas implementaciones específicas.

2.4.1.1. Elementos de un Sistema de Teleoperación:

Un robot industrial prácticamente está compuesto por dos únicos elementos: un brazo manipulador y un armario de control. Para que el robot pueda ser fácilmente programable, se le suele añadir una unidad simple de programación. La forma de trabajar está dividida en dos partes bien diferenciadas. En la primera, el operador programa el robot, indicándole las acciones y movimientos que tiene que realizar en cada momento. En la segunda fase, que es la fase de trabajo del robot propiamente dicha, éste ejecuta el programa una y otra vez de forma automática, estando únicamente bajo las órdenes y la supervisión del armario de control.

Se ha explicado esta forma de trabajar con un robot industrial como contraste de la forma de trabajar con un sistema de teleoperación, y para poder justificar los distintos elementos que lo componen. La primera diferencia está en que en un sistema de teleoperación es necesario contar con un operador humano, que ha de estar siempre presente durante la realización de la tarea; pero, es más, a no ser que se esté teleoperando un robot, es el propio operador el que cierra en todo momento el bucle de control más externo.

A continuación, se va a describir la forma más habitual de trabajar con un sistema básico de telemanipulación.

Primero, el operador maneja un manipulador maestro para indicar los movimientos y demás acciones del manipulador esclavo que realiza el trabajo en la zona remota. Como esta zona suele ser peligrosa o estar a una considerable distancia, es necesario contar con algún tipo de interfaz que proporcione algo de telepresencia al operador. Lo más básico es contar con un interfaz visual que le permita ver los objetos del entorno y cómo se mueve el sistema teleoperado.

Tanto para la transmisión de las señales desde el maestro hacia el esclavo y viceversa, es necesario que existan unos buenos canales de comunicación, adaptados a las necesidades de ancho de banda y a los requerimientos propios del entorno en el que se trabaja. Generalmente, en los sistemas un poco más avanzados se cuenta con un

computador encargado de procesar todo este flujo de señales y de adaptarlas o utilizarlas para otros fines cuando sea necesario.

En resumen, de forma general, un sistema de teleoperación consta de los siguientes elementos (Fig. 2.14):

- **Operador o teleoperador:** Ser humano que realiza a distancia el control de la operación. Su acción puede ir desde un control continuo hasta una intervención intermitente, con la que únicamente se ocupa de monitorizar y de indicar objetivos y planes cada cierto tiempo.
- **Dispositivo teleoperado:** Podrá ser un manipulador, un robot, un vehículo o dispositivo similar. Es la máquina que trabaja en la zona remota y que está siendo comandada o controlada por el operador.
- **Dispositivos de control:** Conjunto de dispositivos que permiten la interacción del operador con el sistema de teleoperación, permitiendo generar comandos.
- **Dispositivos de realimentación:** Dispositivos que realimenten al operador algún tipo de información de la zona remota.
- **Control y canales comunicación:** Conjunto de dispositivos que modulan, transmiten y adaptan el conjunto de señales que se transmiten entre la zona remota y la local. Generalmente, se contará con uno o varias unidades de procesamiento.
- **Sensores:** Conjunto de dispositivos que recogen la información de la zona remota para ser utilizada por el interfaz y el control.

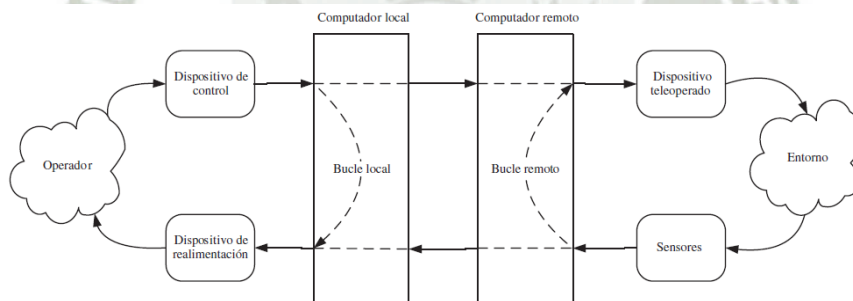


Figura 2.14: Elementos Básicos de un sistema de teleoperación.
Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.4.1.2. Arquitectura de control de un Sistema de Teleoperación:

En este apartado se presenta la arquitectura genérica de un sistema de teleoperación como generalización de los elementos básicos anteriormente citados, identificando y describiendo sus elementos principales.

- **Control manual y control supervisado:**

El grado y modo de intervención en la realización de la tarea con un sistema de teleoperación podrá variar según circunstancias y necesidades. En la Figura 3.15 se muestran diversos grados de control, desde un control totalmente manual a un

control totalmente automático. En este último, caso ya no se estaría trabajando con un sistema de teleoperación, sino con un sistema robotizado.

Entre estos dos extremos se tiene en primer lugar un control manual en el que las señales de control y realimentación son procesadas por un ordenador, principalmente para adecuarlas a las necesidades del dispositivo teleoperado o del operador. En este esquema el operador sigue teniendo el control total del sistema y de la generación de comandos.

Cuando ciertos bucles de control se cierran a través de un ordenador sin que lleguen al operador, se hablará de control supervisado. Es lo que se representa en el esquema tercero y cuarto de la Figura 2.15. Lógicamente, existirán distintos grados de supervisión posibles, en los que operador se verá más o menos liberado de realizar ciertas tareas.

En el tercer esquema, el operador todavía mantiene un cierto grado de control y genera comandos de forma continua, mientras el ordenador se encarga de forma autónoma de adaptar o mejorar los comandos manuales o de cerrar bucles de control complementarios. En este tipo de control supervisado existen dos opciones dependiendo de cómo se comparta temporalmente la realización de la tarea entre el operador y el sistema de control:

- **Control compartido:**

Ambos comparten la ejecución de la tarea simultáneamente, relevando o ayudando el computador al operador en la operación.

- **Control negociado:**

Computador y operador trabajan de forma alternativa. El computador se encarga de ejecutar tareas sencillas de forma automática y el operador de aquellas que impliquen un grado mayor de complejidad o de decisión.

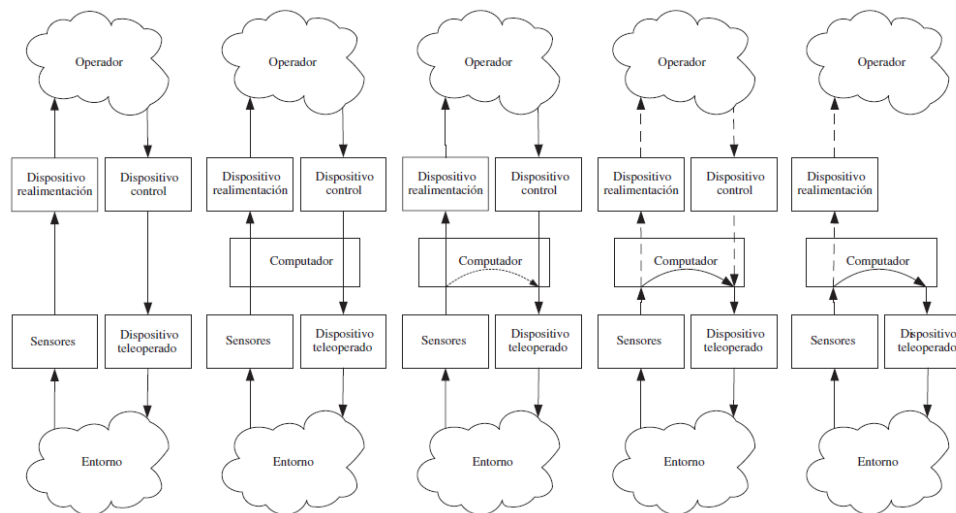


Figura 2.15: Niveles de modo de control Remoto.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

Finalmente, en el cuarto esquema la mayor parte del control la realiza el computador. El operador se ocupa principalmente de la generación intermitente de comandos de alto nivel, que son interpretados y transformados por el ordenador a comandos de bajo nivel para los actuadores. El propio ordenador es el que se encarga de cerrar todos los bucles de control en la zona remota, enviando información al operador para que éste sepa cómo transcurre el desarrollo de la misma. Este tipo de control es especialmente útil para situaciones en las que existe un retardo en las comunicaciones entre la zona local y la zona remota. Una particularización de este tipo de control supervisado es lo que se denomina teleprogramación. En ella, los comandos del operador se generan a través de la interpretación que hace el computador de cómo realizar la tarea tras ser ejecutada por el operador en un entorno virtual que modela de forma realista el entorno de la zona remota.

- **Arquitectura genérica de un sistema de teleoperación**

La Figura 2.16 muestra una arquitectura genérica de un sistema de teleoperación, con los bloques de procesamiento y los canales de comunicación. Consiste en una generalización con mayor grado de detalle del sistema básico presentado en el apartado anterior.

A continuación, se describe la funcionalidad de cada uno de los bloques principales de esta arquitectura.

- **Dispositivo de control:**

Es el dispositivo que controla el operador, generalmente un joystick o un brazo maestro articulado, con el que se generan los comandos hacia el dispositivo teleoperado. A este dispositivo también le pueden llegar órdenes provenientes de bloque que representa la realimentación háptica de fuerzas.

- **Preprocesamiento de comandos:**
Este bloque representa el procesamiento de las señales generadas por los dispositivos de la zona local y que contienen los comandos a enviar a la zona remota.
- **Lazo de control autónomo:**
Dependiendo del grado de autonomía, generalmente existe un lazo de control autónomo, que basado en los comandos provenientes de la zona local y de los propios sensores en el dispositivo teleoperado, modifica o adapta los comandos o incluso genera órdenes nuevas.
- **Sistema teleoperado:**
Se trata en sí del dispositivo teleoperado que está bajo el control del operador a través del sistema de teleoperación. Tendrá unos sensores, que, dependiendo del tipo de realimentación existente en el sistema, enviará distintos tipos de medidas a la zona local, entre ellas generalmente imágenes.
- **Preprocesamiento de realimentación:**
Este bloque representa el procesamiento de las señales de realimentación que se envían desde la zona remota a la zona local. Las señales se preparan y combinan en conjunto con los comandos recibidos y se envían a la zona local.
- **Combinación y distribución de información:**
En este bloque se concentra toda la información proveniente de la zona remota, que comparada con los comandos que se están generando, proporcionan información útil tanto para la posible realimentación de fuerzas como para una realimentación visual, ya sea a través de simulador predictivo u otras formas de representación.
- **Realimentación visual básica:**
Dispositivos básicos de realimentación de la información visual.
- **Realimentación visual aumentada:**
Representa pantallas en las que se representa en forma de gráficos información proveniente de la zona remota.
- **Simulador:**
Muchos de los sistemas de teleoperación cuentan con un simulador en la zona local. Puede tratarse de un simulador predictivo, muy útil para prever el resultado de los comandos en sistemas con retardos en la comunicación entre zona local y zona remota. Pero también puede estar dotado de algo más de inteligencia y ser capaz de, a partir de los comandos básicos generados por el

operador mediante el dispositivo de entrada, crear los comandos finales a transmitir a la zona remota.

○ **Realimentación háptica:**

Los sistemas de teleoperación pueden también incluir un sistema de realimentación de fuerzas directamente sobre el brazo o la mano del operador, a través del dispositivo de control. La generación de esta fuerza puede estar basada directamente en la fuerza de contacto medida en la zona remota, o incluso basada en un modelo existente en el simulador.

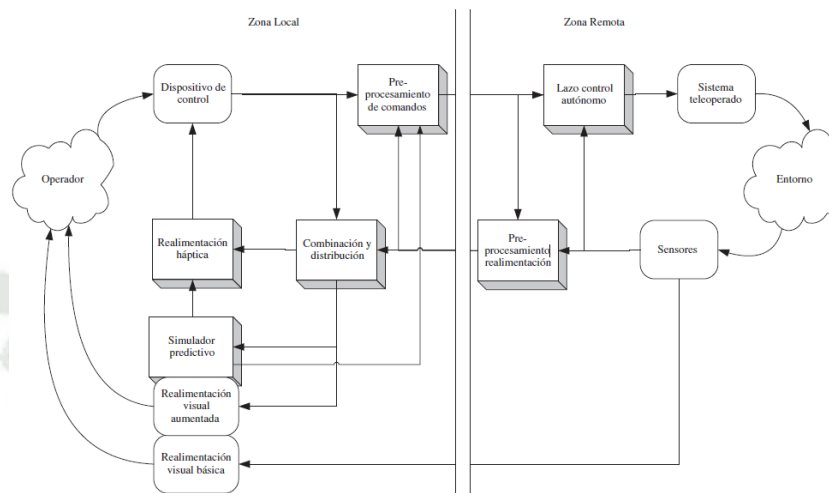


Figura 2.16: Arquitectura genérica de un sistema de Teleoperación.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.4.2. Teleoperación frente a la robótica:

Los robots industriales actuales se diseñan, y se aplican, en la realización de tareas predecibles y repetitivas. Cuando la operación o tarea no puede estructurarse de una forma definida porque el entorno es variable, no se conoce o la tarea no es fija, entonces un robot industrial no podrá ser utilizado. Habrá que recurrir a la inteligencia y capacidad de adaptación de un ser humano. El problema aparece cuando el entorno en el que se tiene que realizar la tarea es especialmente hostil, como puede ser la manipulación de materiales radioactivos o tóxicos, el espacio exterior o las profundidades submarinas. Es en estas situaciones donde el concepto de teleoperación o telemanipulación toma sentido.

La teleoperación comprende todo aquel conjunto de tecnologías que permiten a un ser humano operar a distancia, siendo especialmente útil en la realización de tareas impredecibles a priori y/o no repetitivas en ambientes hostiles o inaccesibles. En estas situaciones se tiene que seguir contando con la inteligencia de un operador, que es el que toma las decisiones.

Aunque tradicionalmente la robótica y la teleoperación han llevado caminos separados, sin embargo tratan con el mismo tipo de dispositivos: los manipuladores. El que una, la robótica, trabaje de forma automática en entornos estructurados y generalmente fijos y con operaciones repetitivas; y la otra, la telemanipulación, centre su aplicación en entorno complejos, impredecibles y normalmente hostiles, define por sí mismo cómo cada una de ellas ha venido abordando el mismo problema: el de la manipulación.

Una diferencia importante se encuentra en la estructura mecánica empleada. Los robots industriales suelen tener cadenas cinemáticas con articulaciones prismáticas o rotativas de 4 a 6 grados de libertad, combinadas según su funcionalidad, y con efectores finales de diseño muy específico y ligado a la aplicación, colocados en el extremo de la cadena. En cambio, los manipuladores teleoperados adoptan configuraciones basadas en articulaciones rotativas en serie (salvo excepciones muy especiales en las que se puede usar cadenas paralelas) con 6 grados de libertad y asemejándose en lo posible al movimiento de un brazo humano.

Además, cuentan la mayoría de las veces, en el extremo con una pinza de propósito general, con la que se pueden coger, como si fuera con una mano, las herramientas necesarias para realizar la tarea.

Otra diferencia, también asociada a su estructura mecánica, es la diferente rigidez y reversibilidad de los actuadores y las articulaciones. Los robots industriales cuentan con unas articulaciones muy rígidas, con muy poco juego angular y ningún tipo de reversibilidad. Esto se debe a los requerimientos tan grandes de velocidad y posicionamiento a los que se les somete,

y que su interacción con el entorno está perfectamente definida, generándose en la misma pocas fuerzas de reacción. Por el contrario, los manipuladores teleoperados poseen unas articulaciones reversibles, poco rígidas y con algún juego angular. Sus requerimientos de posicionamiento y velocidad no son excesivos, pero sí su constante interacción con el entorno sobre el que se generan gran fuerza de reacción, que habrá que minimizar con la flexibilidad.

Es necesario, por último, hacer hincapié, que éstas y otras características contrapuestas son fruto de las diferencias en las aplicaciones y en la forma de trabajar para la que han sido pensados.

En los robots se busca trabajar con total autonomía, mientras que en la teleoperación se busca la adaptación ante el entorno, es decir, flexibilidad. Estas dos capacidades, por las que la teleoperación y la robótica han ido evolucionando desde sus comienzos (Fig. 2.17) y son contrapuestas.

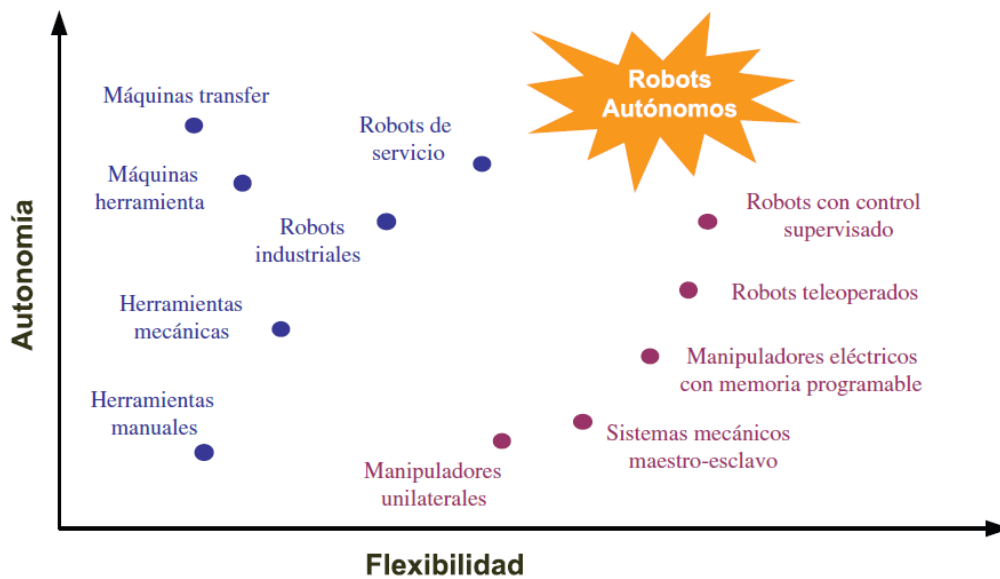


Figura 2.17: Evolución de la Robótica y la Teleoperación en cuanto a flexibilidad y autonomía.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

2.5. Visión Artificial:

La visión es el sentido más importante que tiene el ser humano, definida por Marr como un "proceso que produce, a partir de las imágenes del mundo exterior, una descripción que es útil para el observador y que no tiene información relevante". Así, mientras, para el oído se tiene alrededor de treinta mil terminaciones nerviosas, en la vista hay más de dos millones. Nuestros ojos reciben radiación exterior que deben transformar señales que sean procesadas por el cerebro, por lo tanto, el ojo es un elemento transductor mientras que el cerebro es el que procesa la información.

El hombre, a lo largo de los tiempos, ha imitado, en la construcción de sus artefactos, a la Naturaleza. En este caso también se cumple, dando nacimiento a la visión artificial, en la que las cámaras de vídeo con sus ópticas representan el globo ocular, mientras el computador realiza las tareas de procesamiento, emulando el comportamiento del cerebro.

Se puede definir la "Visión Artificial" como un campo de la "Inteligencia Artificial" que, mediante la utilización de las técnicas adecuadas, permite la obtención, procesamiento y análisis de cualquier tipo de información especial obtenida a través de imágenes digitales. Lo que hace es construir nuevos y más sofisticados algoritmos que sean capaces de obtener información de bajo nivel visual. Y aunque todavía se esté lejos de la percepción visual de los seres vivos, la Visión Artificial es muy eficaz en tareas visuales repetitivas y alienantes para el hombre.

Las principales ventajas de la visión humana respecto a la artificial y viceversa, son:

- **Sistema humano:**
 - Mejor reconocimiento de objetos.
 - Mejor adaptación a situaciones imprevistas.
 - Utilización de conocimiento previo.
 - Mejor en tareas de alto nivel de proceso.
- **Sistema artificial:**
 - Mejor midiendo magnitudes físicas.
 - Mejor para la realización de tareas rutinarias.
 - Mejor en tareas de bajo nivel de proceso.

2.5.1. Configuración de un Sistema de Visión Artificial:

Los dos pilares del sistema físico de visión artificial son: el sistema de formación de las imágenes y el sistema de procesamiento de éstas. El primer apartado estaría constituido por el subsistema de iluminación, de captación de la imagen y de adquisición de la señal en el computador. Una vez introducida la señal en el computador, ésta es procesada mediante los algoritmos para transformarla en información de alto nivel. La cual puede ser utilizada para su representación visual, para actuar en el planificador de un robot o ser fuente de datos para un autómata programable. En definitiva, múltiples periféricos pueden ser receptores de esta información y vincularse con el sistema de procesamiento de las imágenes.

- **Subsistema de iluminación:**
conjunto de artefactos que producen radiación electromagnética para que incidan sobre los objetos a visualizar. Se puede citar algunos elementos como lámparas, pantallas fotográficas, filtros de luz, láseres, etc.
- **Subsistema de captación:**
Son los transductores que convierten la radiación reflejada luminosa en señales eléctricas. Fundamentalmente se habla de las cámaras CCD, no sólo en el espectro visible, sino que van desde la radiación gamma hasta la radiofrecuencia o microondas, dando paso a sensores de ultrasonidos, sonar, radar, telescopía, etc.
- **Subsistema de adquisición:**
La señal eléctrica procedente de las cámaras forma la señal de vídeo. Hay una tendencia creciente a que su naturaleza sea de tipo digital, pero todavía existen muchas señales de vídeo de carácter analógico. Para ser tratadas hay que muestrearlas y cuantificarlas. Ambas tareas son realizadas por las tarjetas de adquisición. Se depositan en el bus de expansión del computador.

- **Subsistema de procesamiento:**

Suele ser un computador o un cluster³ de computadores, dependiendo de las necesidades de los algoritmos de Visión Artificial. Parten de una representación digital de las imágenes y procesan esta información hasta alcanzar otro tipo de información de más alto nivel. La transformación dependerá de la algoritmia.

- **Subsistemas de periféricos:**

Conjunto de elementos receptores de la información de alto nivel. Puede ser un monitor de altas prestaciones gráficas, un automatismo, una impresora sacando las características, etc.

2.5.2. Disciplinas relacionadas con la Visión Artificial:

Un proyecto de Visión Artificial suele ser de tipo multidisciplinar. La ejecución de las distintas etapas, mencionadas en el apartado anterior, requiere del manejo de las siguientes técnicas:

- **Fotografía y Óptica:**

Crear el ambiente de iluminación adecuada en la adquisición de las imágenes, muchas veces requiere del uso de técnicas profesionales de fotografía y vídeo.

- **Procesamiento Digital de las Imágenes:**

Hace referencia a los algoritmos de computación que convierte la imagen digital adquirida en otra de mayor relevancia.

- **Reconocimiento de Patrones:**

Disciplina, dentro de la Inteligencia Artificial, dedicada a la clasificación de las señales y a la búsqueda de patrones existentes dentro de éstas. Se encuentran incluidas las técnicas de clasificadores estadísticos, Redes Neuronales, Sistemas Expertos, Lógica Borrosa, etc.

- **Computación Gráfica:**

Presenta el problema inverso de la Visión Artificial. Si en Visión se desea extraer las características físicas de las imágenes, la Computación.

La naturaleza del proyecto hace que se incida en una disciplina más que en otra.

2.5.3. Aplicaciones:

El número de aplicaciones relacionadas con la Visión Artificial aumenta cada día. En la Tabla 2.8 se citan algunos de los campos donde es empleada esta disciplina:

³ **Cluster:** Conjunto de elementos interrelacionados que funcionan en un mismo sector.

Tabla 2.8: Aplicaciones de la Visión Artificial

Área de producción	Aplicación		
Control de calidad	Inspección de productos (papel, aluminio, acero,...)	Astronomía	Exploración del Espacio
	Identificación de piezas	Reconocimiento de caracteres	Control de cheques, inspección de textos, ...
	Etiquetados (fechas de caducidad,...)	Control de tráfico	Matrículas de coches
	Inspección de circuitos impresos		Tráfico viario
	Control de calidad de los alimentos (naranjas,...)	Meteorología	Predicción del tiempo
Robótica	Control de soldaduras	Agricultura	Interpretación de fotografías aéreas
	Guiado de robots (vehículos no tripulador)		Control de plantaciones
Biomédicas	Análisis de imágenes de microscopía (virus, células, proteínas)	Militares	Seguimiento de objetivos
	Resonancias magnéticas, tomografías, genoma humano		Vigilancia por satélites

Fuente: "Introducción a la Visión Artificial", UPM, Carlos Platero

2.6. Redes Neuronales:

Las Redes Neuronales son un campo muy importante dentro de la Inteligencia Artificial. Inspirándose en el comportamiento conocido del cerebro humano (principalmente el referido a las neuronas y sus conexiones), trata de crear modelos artificiales que solucionen problemas difíciles de resolver mediante técnicas algorítmicas convencionales.

A pesar de su nombre, las redes neuronales no tienen un concepto demasiado complicado detrás de ellas. El nombre, como se puede imaginar, viene de la idea de imitar el funcionamiento de las redes neuronales de los organismos vivos: un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para "aprender" algo que se queda fijo en el tejido.

Ahora bien, por bonito que suene esto, el enfoque biológico no ha sido especialmente útil: las redes neuronales han ido moviéndose para tener un foco en matemáticas y estadística. Se basan en una idea sencilla, dados unos parámetros hay una forma de combinarlos para predecir un cierto resultado. Por ejemplo, sabiendo los píxeles de una imagen habrá una forma de saber qué número hay escrito, o conociendo la carga de servidores de un Centro de Procesamiento de Datos (CPD)⁴, su temperatura y demás existirá una manera de saber cuánto van a consumir, como hacía Google. El problema, claro está, es que no sabemos cómo combinarlos. Las redes neuronales permiten buscar la combinación de parámetros que mejor se ajusta a un determinado problema.

⁴ Se denomina **centro de procesamiento de datos** (CPD) a aquella ubicación donde se concentran los recursos necesarios para el procesamiento de la información de una organización.

Las redes neuronales son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. En el lenguaje propio, encontrar la combinación que mejor se ajusta es "entrenar" la red neuronal. Una red ya entrenada se puede usar luego para hacer predicciones o clasificaciones, es decir, para "aplicar" la combinación.

2.6.1. Modelo de una Neurona Artificial:

El modelo de Rumelhart y McClelland (1986) define un elemento de proceso (EP), o neurona artificial, como un dispositivo que, a partir de un conjunto de entradas, x_i ($i=1\dots n$) o vector $\mathbf{x}$, genera una única salida y .

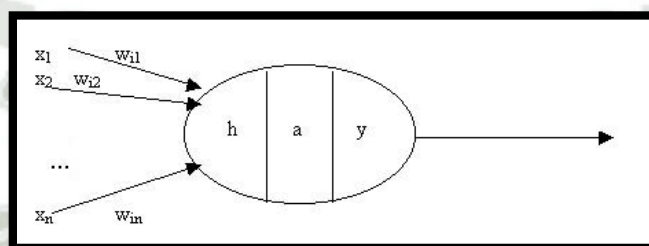


Figura 2.18: Modelo de una red neuronal

Fuente: "Universidad de Salamanca, facultad de ciencias", Luis Alonso Romero, "Redes Neuronales"

Esta neurona artificial consta de los siguientes elementos:

- Conjunto de entradas o vector de entradas $\mathbf{x}$, de n componentes
- Conjunto de *pesos sinápticos* w_{ij} . Representan la interacción entre la neurona presináptica j y la postsináptica i .
- *Regla de propagación* $d(w_{ij}, x_j(t))$: proporciona el potencial postsináptico, $h_i(t)$.
- *Función de activación* $a_i(t)=f(a_i(t-1), h_i(t))$: proporciona el estado de activación de la neurona en función del estado anterior y del valor postsináptico.
- *Función de salida* $F_i(t)$: proporciona la salida $y_i(t)$, en función del estado de activación.

Las señales de entrada y salida pueden ser señales binarias (0,1 – neuronas de McCulloch y Pitts), bipolares (-1,1), números enteros o continuos, variables borrosas, etc.

La regla de propagación suele ser una suma ponderada del producto escalar del vector de entrada y el vector de pesos:

$$h_i(t) = \sum w_{ij} x_j$$

También se usa a menudo la distancia euclídea entre ambos vectores:

$$h_i(t) = \sum (x_j - w_j)^2$$

Existen otro tipo de reglas menos conocidas como la distancia de Voronoi, de Mahalanobis, etc.

La función de activación no suele tener en cuenta el estado anterior de la neurona, sino sólo el potencial $h_i(t)$. Suele ser una función determinista y, casi siempre, continua y monótona creciente. Las más comunes son la función signo (+1 si $h_i(t) > 0$, -1 en caso contrario), la función semilineal y las funciones sigmoideas:

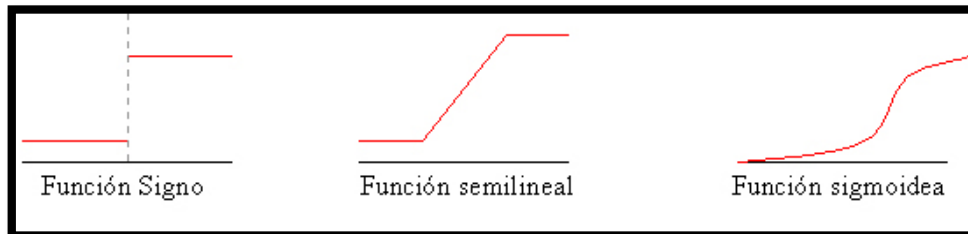


Figura 2.19: Función de activación

Fuente: "Universidad de Salamanca, facultad de ciencias", Luis Alonso Romero, "Redes Neuronales"

La función de salida suele ser la identidad. En algunos casos es un valor umbral (la neurona no se activa hasta que su estado supera un determinado valor).

Con todo esto, el modelo de neurona queda bastante simplificado (Fig. 2.20):

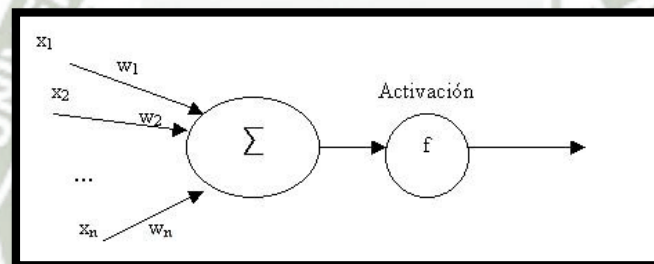


Figura 2.20: Modelo de neurona simplificado

Fuente: "Universidad de Salamanca, facultad de ciencias", Luis Alonso Romero, "Redes Neuronales"

2.6.2. Red Neuronal Artificial:

Una red neuronal artificial (RNA) se puede definir (Hecht – Nielssen 93) como un grafo dirigido con las siguientes restricciones:

1. Los nodos se llaman *elementos de proceso* (EP).
2. Los enlaces se llaman *conexiones* y funcionan como caminos unidireccionales instantáneos
3. Cada EP puede tener cualquier número de conexiones.
4. Todas las conexiones que salgan de un EP deben tener la misma señal.
5. Los EP pueden tener *memoria local*.
6. Cada EP posee una *función de transferencia* que, en función de las entradas y la memoria local produce una señal de salida y / o altera la memoria local.

7. Las entradas a la RNA llegan del mundo exterior, mientras que sus salidas son conexiones que abandonan la RNA.

2.6.3. Arquitectura de las RNA:

La arquitectura de una RNA es la estructura o patrón de conexiones de la red. Es conveniente recordar que las conexiones sinápticas son direccionales, es decir, la información sólo se transmite en un sentido.

En general, las neuronas suelen agruparse en unidades estructurales llamadas *capas*. Dentro de una capa, las neuronas suelen ser del mismo tipo. Se pueden distinguir tres tipos de capas:

- De *entrada*: reciben datos o señales procedentes del entorno.
- De *salida*: proporcionan la respuesta de la red a los estímulos de la entrada.
- *Ocultas*: no reciben ni suministran información al entorno (procesamiento interno de la red).

Generalmente las conexiones se realizan entre neuronas de distintas capas, pero puede haber conexiones intracapa o *laterales* y conexiones de *realimentación* que siguen un sentido contrario al de entrada-salida.

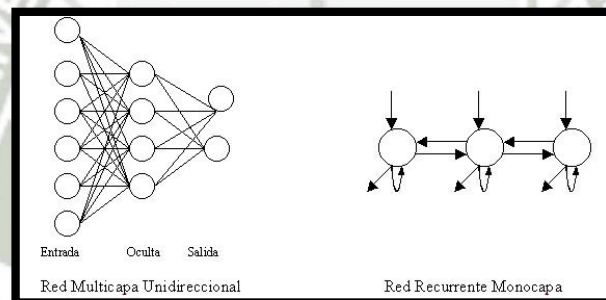


Figura 2.21: Arquitectura de RNA

Fuente: "Universidad de Salamanca, facultad de ciencias", Luis Alonso Romero, "Redes Neuronales"

2.6.4. Multiplicando la Potencia, Redes Multicapa:

Las redes multicapa están formadas por varias capas de neuronas (2,3...). Estas redes se pueden a su vez clasificar atendiendo a la manera en que se conexionan sus capas.

Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Ese tipo de conexiones se denominan conexiones feedforward o hacia delante.

Por el contrario, existen algunas redes en que las capas aparte del orden normal algunas capas están también unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información. Las conexiones de este tipo se llaman conexiones hacia atrás, feedback o retroalimentadas.

2.6.5. Aprendizaje de las RNA:

Es el proceso por el que una RNA actualiza los pesos (y, en algunos casos, la arquitectura) con el propósito de que la red pueda llevar a cabo de forma efectiva una tarea determinada.

Hay tres conceptos fundamentales en el aprendizaje:

- **Paradigma de aprendizaje:** información de la que dispone la red.
- **Regla de aprendizaje:** principios que gobiernan el aprendizaje.
- **Algoritmo de aprendizaje:** procedimiento numérico de ajuste de los pesos.

Existen dos paradigmas fundamentales de aprendizaje:

- **Supervisado:** la red trata de minimizar un error entre la salida que calcula y la salida deseada (conocida), de modo que la salida calculada termine siendo la deseada.
- **No supervisado o autoorganizado:** la red conoce un conjunto de patrones sin conocer la respuesta deseada. Debe extraer rasgos o agrupar patrones similares.

En cuanto a los algoritmos de aprendizaje, tenemos cuatro tipos:

- **Minimización del error:** reducción del gradiente, retropropagación, etc. La modificación de pesos está orientada a que el error cometido sea mínimo.
- **Boltzmann:** para redes estocásticas, donde se contemplan parámetros aleatorios.
- **Hebb:** cuando el disparo de una célula activa otra, el peso de la conexión entre ambas tiende a reforzarse (Ley de Hebb).
- **Competitivo:** sólo aprenden las neuronas que se acercan más a la salida deseada.

2.6.6. Red Back-Propagation:

La red Back-Propagation está basada en la generalización de la regla delta.

Al igual que el Perceptrón, ADALINE y MADALINE, la red Back-Propagation se caracteriza por tener una arquitectura en niveles y conexiones estrictamente hacia adelante entre las neuronas, además de utilizar aprendizaje supervisado.

Consiste en un aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo propagación-adaptación de dos fases.

- **Primera Fase:** se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, después se compara el resultado obtenido en las neuronas de salida con la salida que se desea obtener y se calcula un valor del error para cada neurona de salida.
- **Segunda Fase:** estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original.

La importancia de este algoritmo consiste en su capacidad de auto adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes.

Ya entrenada la red se podrá aplicar esa misma relación (terminado el entrenamiento), a nuevos vectores de entradas con ruido o incompletos, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje.

2.6.7. Cinemática Inversa de un robot Manipulador:

Los robots suelen programarse en términos de las coordenadas cartesianas de su entorno, mientras que su control se realiza mediante variables articulares. De aquí la importancia que, para el control de robots, tiene el disponer de funciones precisas que hagan corresponder a una posición y orientación cualesquiera en el espacio, los valores de las variables articulares que hacen que el elemento terminal del robot alcance la posición y orientación mencionadas.

La utilización de redes neuronales para aprender la cinemática inversa de robots manipuladores presenta especial interés cuando no se dispone de un modelo preciso de algunas de las articulaciones (éste es el caso, por ejemplo, de los robots flexibles) o cuando, debido a las condiciones de operación del robot (en el espacio, bajo el mar, etc.), no es posible recalibrarlo.

2.6.7.1. Redes Multicapa con retropropagación del Error:

La manera más sencilla de abordar el aprendizaje de la cinemática inversa es aplicar el esquema de modelización inversa directa a una red multicapa que incorpore la regla de retropropagación del error. Muchos investigadores han seguido este enfoque y lo han aplicado a una variedad de modelos de robot. La manera de asegurar que la función sea inyectiva, requisito indispensable para poder aplicar este esquema, es elegir para el conjunto de aprendizaje sólo configuraciones del robot en que las articulaciones permanezcan siempre en los mismos semiespacios.

Jordan y Rumelhart⁵ en el año 1992 aplicaron el esquema de modelización hacia adelante al aprendizaje de la cinemática inversa de un robot manipulador con 3 articulaciones que se mueve en un plano. Aunque en este caso la cinemática inversa no es una función, sino una correspondencia de 2 a 3 variables, el controlador neuronal consigue converger a una función cinemática inversa de 2 a 2 variables. Los autores mencionan que es fácil incorporar al procedimiento de aprendizaje la restricción de minimización de la norma o restricciones de suavidad temporal para guiar la selección de la función inversa obtenida.

La conclusión alcanza después de una extensa experimentación con redes multicapa siguiendo ambos enfoques es que una aproximación burda de la cinemática inversa puede obtenerse rápidamente, pero que una representación precisa de la verdadera función es extremadamente difícil e incluso a veces imposible. La razón de esta dificultad parece ser el carácter global de la aproximación obtenida con este tipo de redes, ya que ésta se ve afectada por todos y cada uno de los pesos.

Una manera obvia de evitar este efecto global es por supuesto la utilización de representaciones locales, de modo que cada parte de la red responsable de sólo una pequeña porción del espacio de entradas. Esta es la idea subyacente al trabajo descrito en el siguiente subapartado.

2.7. Torque Computado:

Esta técnica de control se basa en las ecuaciones de movimiento de Lagrange-Euler o Newton-Euler, que describen el comportamiento dinámico de un manipulador robotizado. La técnica del par calculado emplea realimentación no-lineal para desacoplar las interacciones entre los eslabones del manipulador, y utiliza un regulador proporcional más derivativo o PID para controlar los actuadores, por lo tanto, realiza un control de tipo anticipativo.

⁵ Jordan, M.I. y Rumelhart, D.E., "Forward Models: Supervised learning with a distal teacher", Cognitive Science, 16, (1992), 307-354.

Si se utiliza la formulación de Lagrange-Euler para modelar dinámicamente un manipulador de n grados de libertad, se tendrá:

$$\tau_{c_k} = \sum_{j=1}^n M_{kj} * \ddot{q}_j + \sum_{j=1}^n c_{kj} * \dot{q}_j + G_k, \quad [2.38]$$

Que vectorialmente se puede expresar como:

$$\tau_c = M(q(t)) * \ddot{q}(t) + N(q(t), \dot{q}(t)) + G(q(t)), \quad [2.39]$$

Dónde: $N(q(t), \dot{q}(t)) = C(q(t), \dot{q}(t)) * \dot{q}(t)$ y, además:

Tabla 2.9: Entidades del modelo dinámico mediante Lagrange-Euler

Variable	Definición
τ_{c_k}	Elemento de fuerza generalizada perteneciente al vector de fuerza generalizada τ_c , con $\tau_c \in R^{nx1}$
M_{kj}	Elemento constituyente de la matriz inercial del sistema $M(q(t))$, $M(q(t)) \in R^{nxn}$
$\ddot{q}_j$	Aceleración de la j -ésima junta, que constituye el vector $\ddot{q}(t)$ con $\ddot{q}(t) \in R^{nx1}$
c_{kj}	Elemento perteneciente a la matriz fuerza/torque de Coriolis y centrífuga de la k -ésima junta, $C(q(t), \dot{q}(t))$, con $C(q(t), \dot{q}(t)) \in R^{nxn}$
$\dot{q}_j$	Velocidad de la j -ésima junta, que constituye el vector $\dot{q}(t)$ con $\dot{q}(t) \in R^{nx1}$
G_k	Fuerza de carga gravitatoria sobre la k -ésima junta, que constituye el vector de fuerza de la carga gravitatoria $G(t)$, con $G(t) \in R^{nx1}$

Fuente: "Técnica del par calculado, Robótica Industrial", Universidad de Santiago de Chile, 2016

El torque que los actuadores deben producir para mover la carga τ_c es calculado empleando dinámica inversa sobre la ecuación [2.39] por lo tanto, los siguientes elementos se emplean en el cálculo de dicho torque: $\tilde{M}(q)$, $\tilde{N}(q, \dot{q})$ y $\tilde{G}(q)$ donde $\sim$ indica un valor estimado. De este modo, la ley de control tiene la siguiente estructura:

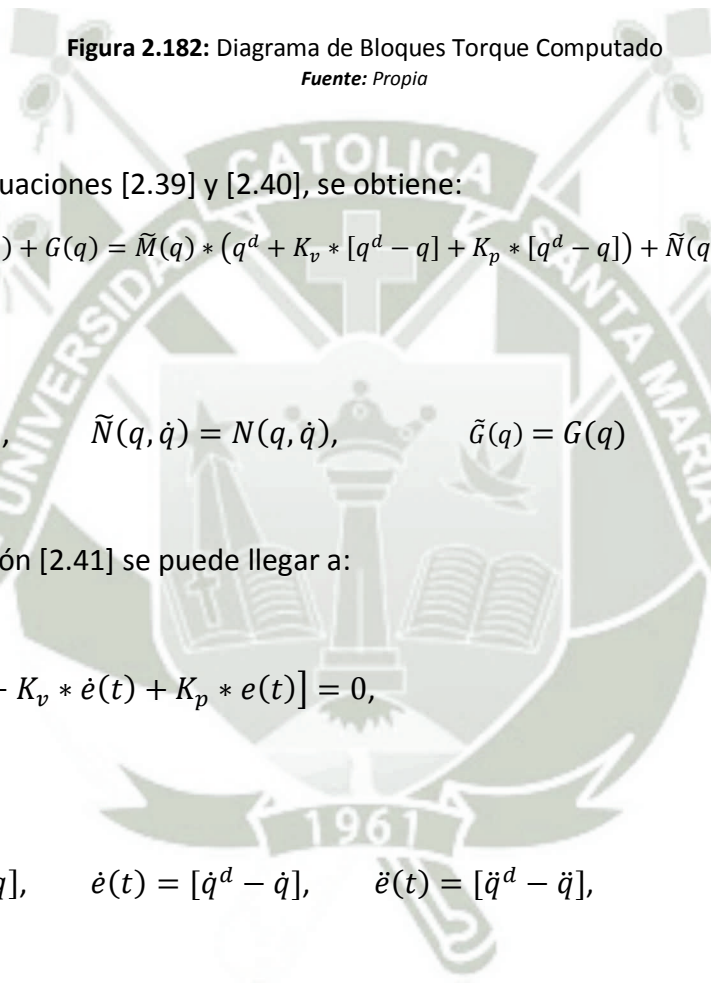
$$\tau_c = \tilde{M}(q) * (\ddot{q}^d + K_v * [\dot{q}^d - \dot{q}] + K_p * [q^d - q]) + \tilde{N}(q, \dot{q}) + \tilde{G}(q), \quad [2.40]$$

Dónde:

Tabla 2.10: Entidades de la Ley de Control

Variable	Definición
$\ddot{q}^d$	Vector generalizado de aceleraciones deseadas $\in R^{nx1}$
$\dot{q}^d$	Vector generalizado de velocidades deseadas $\in R^{nx1}$
q^d	Vector generalizado de posiciones deseadas $\in R^{nx1}$
K_v	Matriz de ganancia de realimentación de velocidad $\in R^{nxn}$
K_p	Matriz de ganancia de realimentación de posición $\in R^{nxn}$

Fuente: "Técnica del par calculado, Robótica Industrial", Universidad de Santiago de Chile, 2016



Fuente: Propia

Si:

$$\tilde{M}(q) = M(q), \quad \tilde{N}(q, \dot{q}) = N(q, \dot{q}), \quad \tilde{G}(q) = G(q) \quad [2.42]$$

$$M(q) * [\ddot{e}(t) + K_v * \dot{e}(t) + K_p * e(t)] = 0, \quad [2.43]$$
$$e(t) = [q^d - q], \quad \dot{e}(t) = [\dot{q}^d - \dot{q}], \quad \ddot{e}(t) = [\ddot{q}^d - \ddot{q}], \quad [2.44]$$
Tabla 2.11: Representaciones del error

Fuente: "Técnica del par calculado, Robótica Industrial", Universidad de Santiago de Chile, 2016

Además, como $M(q)$ es siempre no singular, K_p y K_v se pueden escoger de forma tal que las raíces de la ecuación característica (ecuación [2.43]) tengan partes reales negativas, y el vector de error de posición tienda asintóticamente a cero.

Algunos investigadores simplifican la ley de control empleada en la técnica del par calculado (ecuación [2.40]), debido a que el cálculo de los pares/fuerzas de articulación es muy ineficaz, por lo cual, se desprecian los términos de acoplamiento relacionados con la velocidad, es decir $\tilde{N}(q, \dot{q})$, y los elementos que están fuera de la diagonal de $\tilde{M}(q)$ permitiendo que la estructura de control se reduzca a:

$$\tau_c = \text{Diagonal}\langle M(q) * (\ddot{q}^d + K_v * [\dot{q}^d - \dot{q}] + K_p * [q^d - q]) \rangle. \quad [2.45]$$

Dónde:

$$\text{Diagonal}\langle *_{k} \rangle = \begin{bmatrix} *_{1} & 0 & \dots & 0 \\ 0 & *_{2} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & *_{n} \end{bmatrix} \quad [2.46]$$

Con $k=1,2, \dots, n$.

De acuerdo a lo establecido anteriormente, la técnica del par calculado es un control de compensación anticipativo, cuya principal desventaja es que la convergencia de vector de error de posición depende de los coeficientes dinámicos $M(q)$, $N(q, \dot{q})$ y $G(q)$, de la ecuación de movimiento, lo cual exige un modelo muy preciso de la dinámica del manipulador, y además sufre de sensibilidad a los errores en la estimación de $\tilde{M}(q)$, $\tilde{N}(q, \dot{q})$ y $\tilde{G}(q)$.

CAPÍTULO III: DISEÑO

El Consejo de Acreditación para la Ingeniería y la Tecnología (ABET), define al diseño en Ingeniería como “el proceso de concepción de un sistema para satisfacer unas necesidades. Éste es un proceso de toma de decisiones, a menudo iterativo, en la cual las ciencias básicas, las matemáticas y los conocimientos en ingeniería son aplicados para transformar óptimamente los recursos y satisfacer los objetivos”. De esta manera, llevar a cabo un diseño implica realizar un procedimiento, el mismo que inicia con la especificación de un problema que nace de una necesidad, para luego esquematizar su solución y diseñar de forma preliminar la máquina, estructura o sistema, estableciendo las características globales y específicas de cada componente, para finalmente realizar un análisis basado en estudios de ingeniería, de todos y cada uno de los elementos, y preparar así los dibujos necesarios con sus respectivas especificaciones tanto estéticas como funcionales.

Se debe tener en cuenta que los objetivos de diseño hasta el momento pueden tener alguna modificación en esta parte de la tesis; sin embargo, en etapas futuras estos cambios generarían mayor complejidad y costo en el proyecto final. Definir los requerimientos es de vital importancia porque guiarán y contralarán todo el proceso de diseño marcando un camino a seguir desde las etapas iniciales donde se es más propenso a perder el enfoque del objetivo final. Siendo esta parte de gran envergadura, es que se debe revisar y comprender a profundidad los criterios y especificaciones necesarias para una solución satisfactoria, puesto que si estos requisitos de diseño son muy estrictos pueden llevar a aumento del costo del proyecto y si son muy laxos puede que no permitan alcanzar el objetivo propuesto.

En la Tabla 3.1 se definen los requerimientos de diseño necesarios para la construcción del brazo robótico.

Tabla 3.1: Cuadro de Requerimientos

Características	Deseos o Exigencias	Condiciones
Aplicación	E	El robot deberá ser diseñado para que pueda manipular sustancias tóxicas usadas en un laboratorio farmacéutico.
Geometría	D	El espacio de trabajo del brazo robótico tiene una altura de 0.6 m y un radio de 0.4 m.
	E	El elemento de sujeción del brazo robótico tiene que tener una superficie de contacto adecuada para posibilitar el correcto agarre de los instrumentos farmacéuticos a manipular.
	D	La maqueta del proyecto tendrá las siguientes dimensiones: altura (0.8 m), largo (1 m) y ancho (0.5 m).
	E	La colocación de los complementos necesarios para la operación del robot no deben obstaculizar el desplazamiento del mismo ni de los usuarios.
Cinemática	E	El brazo robótico contará con 4 GDL
	E	Todas sus articulaciones deben ser del tipo rotacional.
	E	Los algoritmos de control se encargarán de la posición y orientación del efector final.
	E	La velocidad angular de cada articulación del brazo robótico variará según la operación pero deberá ser tal que evite el derrame de las sustancias a desplazar.
Fuerza	E	La presión de la garra debe ser tal que permita la sujeción adecuada del instrumento a manipular y que no provoque la fractura del mismo.
	D	El peso del brazo robótico a considerar no debe ser mayor a 4kg aproximadamente.
Energía	E	Los actuadores del robot deben estar alimentados por un fuente de corriente continua
Señales	E	Las señales de control emitidas deberán ser de tipo digital.
	E	Comprobación de señales de autodiagnóstico en la unidad de control previamente al primer funcionamiento (niveles de tensión de las fuentes de alimentación, leds indicadores, mensajes de error, etc.).
Seguridad	D	El brazo robótico debe contar con señalización de visualización para la seguridad tanto del operador como de los usuarios externos referente a su funcionamiento
	E	La carga a soportar por el brazo robótico no debe superar 0.5 kg. de peso para proteger a los actuadores de sobreesfuerzos.
	E	Los actuadores deben contar un grado de protección, IP 65, protección contra polvo y agua para evitar fallos en un futuro.
	E	Uso de un supresor de picos y estabilizador para evitar elevaciones repentinas de tensiones peligrosas sobre los componentes a utilizar.
Ergonomía	E	El operador llevará a cabo las tareas teniendo una postura de trabajo adecuada y no forzada. deberá no ser forzada
	E	El ambiente de trabajo debe tener una iluminación adecuada
	E	El tablero de operación estará ubicado dentro de un campo visual adecuado para el operador
Fabricación	E	El material a utilizar debe garantizar la higiene del proceso de trabajo
	E	El material a utilizar debe ser de alta resistencia, ligero y que permita una fácil fabricación
	E	El proceso de fabricación de las piezas debe realizarse a través de una máquina que garantice un acabado de alta precisión y la menor cantidad de desperdicio de material.
Control	E	El tipo de control aplicado debe ser en lazo cerrado.
	E	El control que gobierna el funcionamiento del brazo robótico debe ser teleoperado y asistido por visión artificial y redes neuronales.
	E	Uso del lenguaje propio del Software Matlab para el sistema de control y de python para el envío y recepción de señales
Montaje	E	El manipulador robótico debe ser armado mediante subensambles para facilitar su montaje final.
	D	La apariencia del robot deberá ser similar a la de un brazo humano al final de su montaje
	E	Los puntos de amarre deben ser de fácil acceso para su montaje y desmontaje
Transporte	E	El robot deberá ser de fácil transporte tanto en partes como en su ensamble final
Uso	E	El brazo robótico operará a condiciones ambientales
	E	El nivel de ruido no debe superar los 55 Db según la OMS para no alterar la salud de los operarios
Mantenimiento	E	El diseño del manipulador robótico deberá asegurar un fácil mantenimiento mecánico y eléctrico de acuerdo al régimen de uso
	E	Las piezas (elementos de sujeción) y dispositivos (servomotores) deberán ser de uso comercial para su disponibilidad en caso de cambio o reparación.
Costos	D	Los costos de adquisición de dispositivos y de fabricación deberán ser lo más bajo posible sin sacrificar la calidad del prototipo final.

Fuente: Propia

En el campo de la Robótica, la acción de diseñar implica no sólo esquematizar y hacer tangible la parte mecánica del manipulador, sino también su parte electrónica; por esta razón, es que el desarrollo de este capítulo contiene dos grandes sectores primordialmente, evocado el primero a la parte estructural del brazo, y el segundo a la definición de componentes electrónicos, los mismos que permiten y hacen efectivo el control del robot durante su funcionamiento. Así mismo, se incluye una parte final referida al diseño y construcción de la maqueta, que tiene como finalidad la representación del ambiente donde se lleva a cabo la aplicación.

3.1. Diseño Mecánico:

En este apartado se desarrollará de forma práctica, todos los conocimientos teóricos adquiridos, describiendo así cada uno de los pasos llevados a cabo para el diseño mecánico del manipulador robótico. Conlleva inicialmente un diseño preliminar, en donde se definen características generales del robot, para que después con el uso de diferentes herramientas matemáticas, se pueda solucionar tanto la cinemática como la dinámica del mismo, lo que finalmente nos permitirá la elección de los actuadores y el diseño virtual de cada uno de los eslabones a través del Solidwork. Se debe considerar, además, que el proceso desarrollado no contempla cálculos matemáticos de sujeciones.

3.1.1. Estructura Mecánica:

Se considera al brazo robótico en desarrollo, como una cadena cinemática abierta de 5 eslabones enlazados entre sí mediante 4 articulaciones de un solo grado de libertad cada una, cuya constitución física guarda similitud con la anatomía humana, pudiendo identificar en él dedos, mano, muñeca, brazo, codo, antebrazo, hombro, tronco y cintura; de esta manera se satisface el requerimiento cinemático de un manipulador de 4 GDL, siendo éstos suficientes para posicionar y orientar el elemento final y llevar así a cabo las tareas que se le encomiendan.

Así mismo se optó por uniones de tipo rotacional en su totalidad, dando lugar a una configuración angular, debido a que se plantea un proceso de teleoperación para conseguir la movilidad del robot, siendo éste el esclavo y teniendo como maestro el propio brazo humano, de allí radica la importancia de contemplar una similitud entre los movimientos de ambos; de igual manera, este tipo de junta, nos garantiza lograr el alcance físico del extremo del autómatas para la manipulación de sustancias dentro del espacio de trabajo cumpliendo así requisitos tanto cinemáticos como de la aplicación para la que es diseñado.

El extremo del manipulador robótico estará conectado a una herramienta de sujeción que le permita cumplir con la labor encomendada, la cual consiste en la manipulación de instrumentos de laboratorio para investigación, es por ello que se vio por conveniente seleccionar una pinza de dos dedos como efector final.

Finalmente, en cuanto al sistema de transmisión de potencia, se decidió no emplear un reductor intermedio entre las articulaciones y los ejes de los actuadores, teniendo así en su totalidad mecanismos de accionamiento directo, ya que permiten un posicionamiento rápido y preciso además de la simplificación del sistema mecánico; sin embargo, tienen como desventaja el uso de actuadores de gran tamaño, inconveniente que no es tomado en cuenta durante el diseño de la tesis debido que el peso del robot y la carga a manipular por el mismo no es considerable, es por ello que los motores a emplearse serán de tamaño reducido pero con un torque adecuado para su correcta operación.

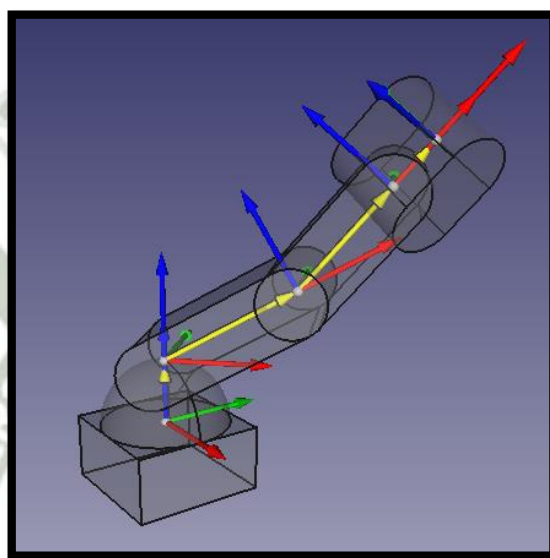


Figura 3.1: Estructura básica del brazo robótico

Fuente: Propia

3.1.2. Diseño Preliminar Manual:

Para comenzar a diseñar nuestro manipulador robótico había que tener en cuenta los requerimientos de diseño antes propuestos, relacionados a la aplicación, geometría y cinemática del brazo robótico.

Referente a la aplicación hay que tomar en cuenta que manipulará instrumentos de un laboratorio farmacéutico como por ejemplo probetas, pipetas, etc. Por lo cual se tiene que conocer las alturas y diámetros de los mismos y elegir algún rango de medidas aceptable para el prototipo a desarrollar.

Entre las medidas encontradas se determinó trabajar con las siguientes:

Tabla 3.2: Medidas a considerar

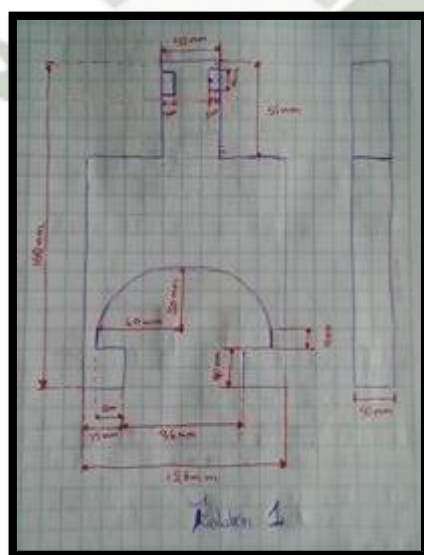
Rango de Diámetros (mm)	Rango de Alturas (mm)
11 - 65	160 - 320

Fuente: Propia

Teniendo en cuenta esta información previa se planteó la geometría del robot usando formas prismáticas para los eslabones del brazo robótico, excluyendo aún al efector final, los cuales se plasmaron en planos simples en una hoja cuadriculada como se puede ver a continuación en las siguientes figuras:

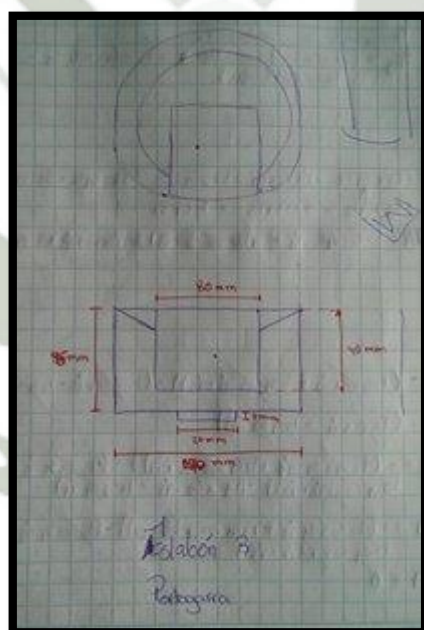


Fuente: Propia



Fuente: Propia

Fuente: Propia



Fuente: Propia

69

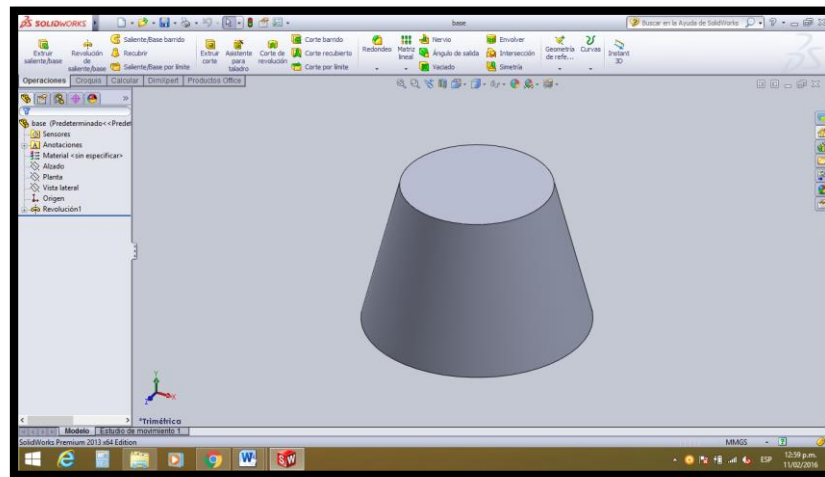


Figura 3.6: Base 1 solidworks
Fuente: Propia

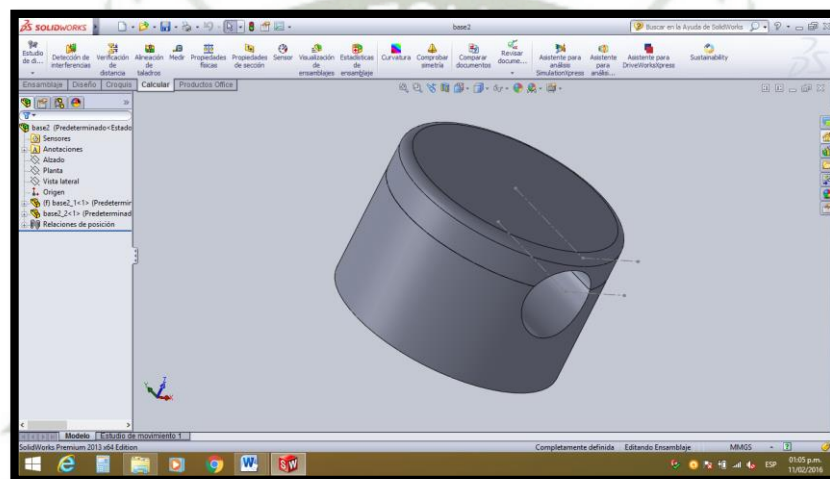


Figura 3.7: Base 2 solidworks
Fuente: Propia

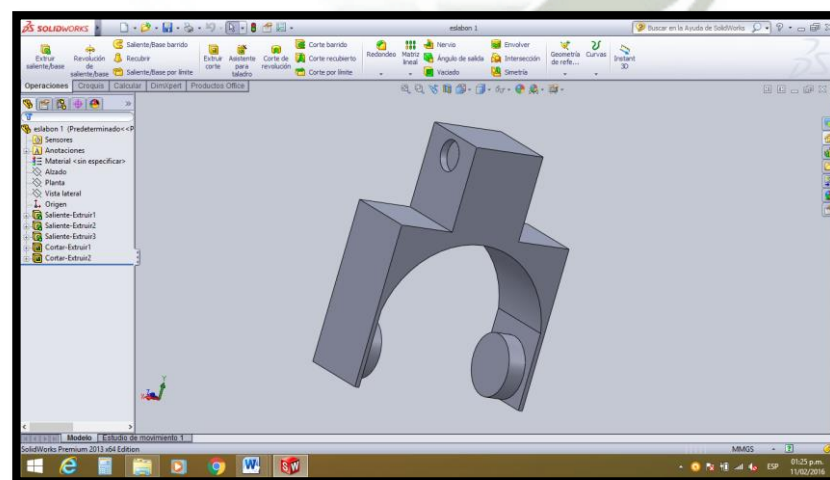
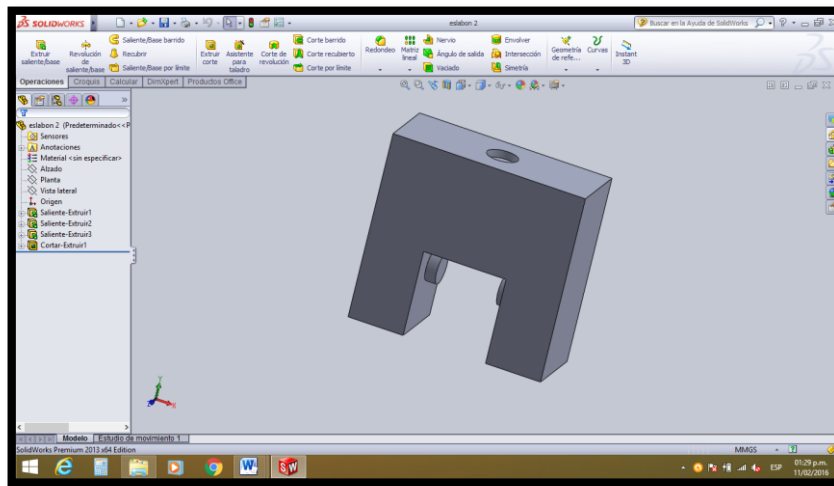
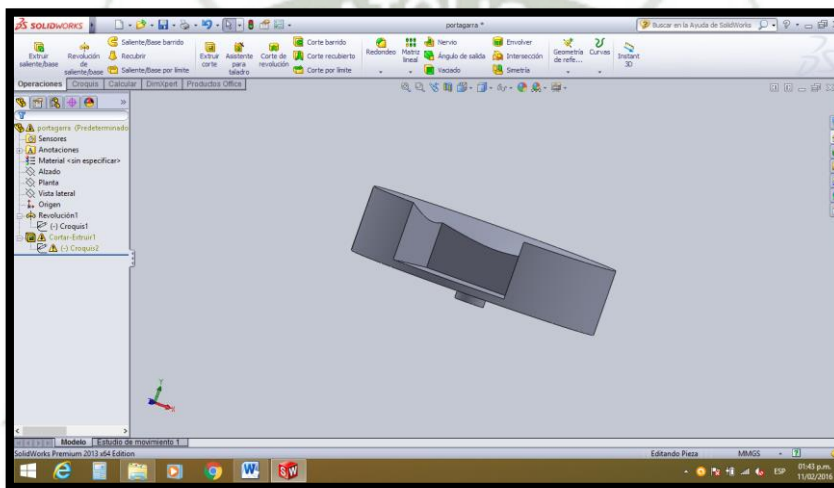


Figura 3.8: Eslabón 1 solidworks
Fuente: Propia


Figura 3.9: Eslabón Solidwork
Fuente: Propia

Figura 3.10: Eslabón 3 Solidwork
Fuente: Propia

3.1.3. Análisis Cinemático:

El estudio de la cinemática resulta de gran importancia en el proceso de diseño, puesto que nos permite predecir en qué punto específico del espacio se hallará un cuerpo, que velocidad tendrá en un tiempo determinado, o el lapso que demorará en llegar a su destino. Por esta razón es que, en el campo de la robótica, realizar un análisis cinemático es uno de los primeros pasos para el diseño de un robot, ya que nos posibilita estudiar la geometría del movimiento del manipulador robótico con respecto a un sistema de coordenadas fijo, sin tener en cuenta las fuerzas o momentos que lo originen. En esta sección se aplican las herramientas matemáticas aprendidas con anterioridad para hallar las ecuaciones que describen analíticamente el desplazamiento espacial del brazo robótico como función del tiempo.

3.1.3.1. Cinemática Directa:

El problema cinemático directo consiste en determinar las relaciones que permiten conocer la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas referencial ubicado de preferencia en la base, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del manipulador.

Existen diferentes procedimientos para llevar a cabo este cálculo, siendo el algoritmo de Denavith-Hartenberg el más utilizado. Se trata de un método matricial que permite establecer de manera sistemática un sistema de coordenadas (S_i) ligado a cada eslabón i de una cadena articulada, además plantea que, escogiendo adecuadamente los sistemas de coordenadas, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. Estas transformaciones constan de una serie de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i con el sistema del elemento $i-1$ y son las siguientes:

- Rotación alrededor del eje z_{i-1} un ángulo θ_i .
- Traslación a lo largo de z_{i-1} una distancia d_i ; vector $d_i(0,0,d_i)$.
- Traslación a lo largo de x_i una distancia a_i ; vector $a_i(0,0,a_i)$.
- Rotación alrededor del eje x_i , un ángulo α_i .

Debido a que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden dado, obteniendo así el siguiente resultado:

$${}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [3.1]$$

Donde θ_i , a_i , d_i , α_i , son los parámetros D-H del eslabón i .

De este modo, es que el algoritmo de Denavith-Hartenberg se reduce a obtener las matrices A y relacionar así todos y cada uno de los eslabones del robot, mediante la elección de manera apropiada de los sistemas $\{S_{i-1}\}$ y $\{S_i\}$ y la definición de los parámetros antes mencionados siguiendo determinadas normas, las mismas que se encuentran explícitas en la Tabla 3.3:

Tabla 3.3: Normas a seguir para la aplicación del algoritmo D-H

Norma	Descripción
D-H1	Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.
D-H2	Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.
D-H3	Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro.
D-H4	Para i de 0 a n-1, situar el eje z_i sobre el eje de la articulación i+1.
D-H5	Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0 . Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0 .
D-H6	Para i de 1 a n-1, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i . Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación i+1.
D-H7	Situar x_i en la línea normal común a z_{i-1} y z_i .
D-H8	Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i .
D-H9	Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n .
D-H10	Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.
D-H11	Obtener d_i como la distancia, medida a lo largo de z_{i-1} , que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.
D-H12	Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.
D-H13	Obtener α_i como el ángulo que habría que girar entorno a x_i (que ahora coincidiría con x_{i-1}), para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.
D-H14	Obtener las matrices de transformación ${}^{i-1}A_i$, definidas en [4.1].
D-H15	Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0A_1, {}^1A_2, \dots, {}^{n-1}A_n$.
D-H16	La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base en función de las n coordenadas articulares.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de Robótica".

Conocidos los pasos a seguir, es que se procede a la aplicación del Algoritmo, desarrollando primero las normas que nos permiten definir de manera correcta, los sistemas de coordenadas ligados a cada eslabón del manipulador de 4GDL a diseñar, tal y como se muestra en la Figura 3.11.

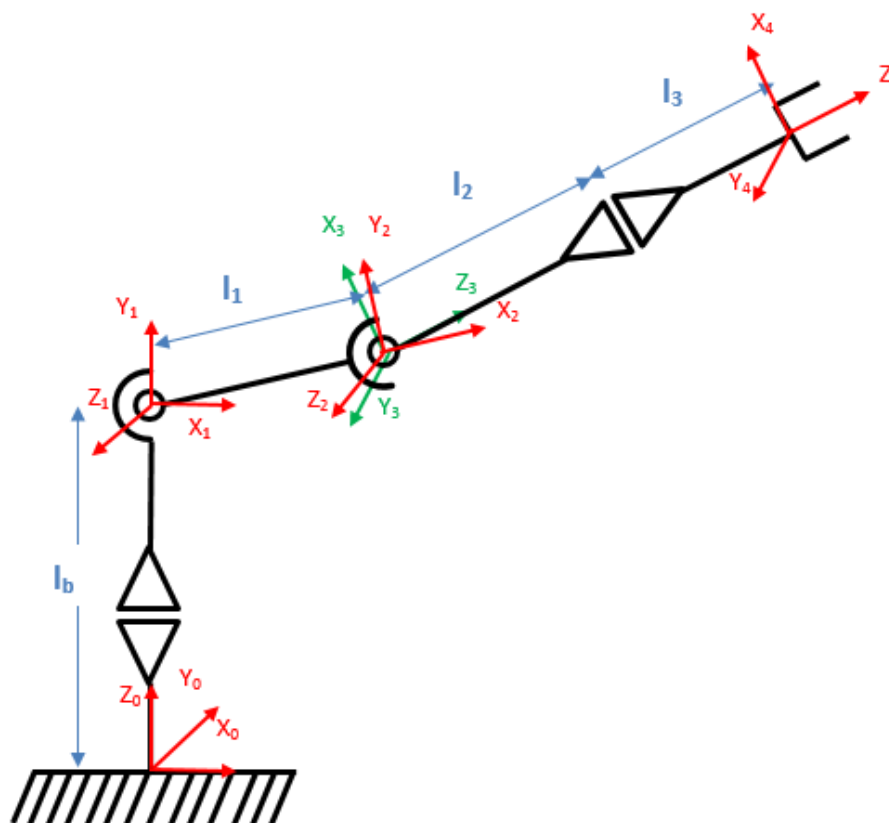


Figura 3.11: Representación esquemática del brazo robótico 4GDL
Fuente: Propia

Posteriormente, se determinan los parámetros de Denavith-Hartenberg, los mismos que se encuentran recopilados en la Tabla 3.4.

Tabla 3.4: Parámetros D-H del brazo robótico.

ARTICULACION	Θ	d	a	A
1	q_1	l_0	0	$\pi/2$
2	q_2	0	l_1	0
3	$q_3 + \pi/2$	0	0	$\pi/2$
4	q_4	$l_2 + l_3$	0	0

Fuente: Propia

Luego, se calculan las matrices A correspondientes, sustituyendo los parámetros obtenidos en la expresión general de la siguiente manera:

$${}^{i-1}A_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i) * \text{sen}(\theta_i) & \text{sen}(\alpha_i) * \text{sen}(\theta_i) & a_i * \cos(\theta_i) \\ \text{sen}(\theta_i) & \cos(\alpha_i) * \cos(\theta_i) & -\text{sen}(\alpha_i) * \cos(\theta_i) & a_i * \text{sen}(\theta_i) \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^0A_1 = \begin{bmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & l_b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1A_2 = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & l_1 \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & l_1 \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2A_3 = \begin{bmatrix} \cos\left(\frac{\pi}{2} + q_3\right) & 0 & \sin\left(\frac{\pi}{2} + q_3\right) & 0 \\ \sin\left(\frac{\pi}{2} + q_3\right) & 0 & -\cos\left(\frac{\pi}{2} + q_3\right) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^3A_4 = \begin{bmatrix} \cos(q_4) & -\sin(q_4) & 0 & 0 \\ \sin(q_4) & \cos(q_4) & 0 & 0 \\ 0 & 0 & 1 & l_2 + l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finalmente obtenemos la Matriz T, como resultado del producto de las matrices A, que indica la localización espacial del extremo del robot en función de las coordenadas articulares, con respecto al sistema referencial fijo de la base del mismo, dando así solución al problema cinemático directo:

$$T = {}^0A_1 {}^1A_2 {}^2A_3 {}^3A_4 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Debido a la extensión de cada uno de los términos de la matriz, es que se muestran de manera independiente:

$$n_x = \sin(q_1)\sin(q_4) - \cos(q_1)\cos(q_4) * (\cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2))$$

$$n_y = -\cos(q_1)\sin(q_4) - \sin(q_1)\cos(q_4) * (\cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2))$$

$$n_z = \cos(q_2 + q_3) * \cos(q_4)$$

$$o_x = \sin(q_1)\cos(q_4) + \cos(q_1)\sin(q_4) * (\cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2))$$

$$o_y = -\cos(q_1)\cos(q_4) + \sin(q_1)\sin(q_4) * (\cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2))$$

$$o_z = -\cos(q_2 + q_3) * \sin(q_4)$$

$$a_x = \cos(q_1) * \cos(q_2 + q_3)$$

$$a_y = \sin(q_1) * \cos(q_2 + q_3)$$

$$a_z = \sin(q_2 + q_3)$$

$$P_x = \cos(q_1) * (l_2 \cos(q_2 + q_3) + l_3 \cos(q_2 + q_3) + l_1 \cos(q_2))$$

$$P_y = \sin(q_1) * (l_2 \cos(q_2 + q_3) + l_3 \cos(q_2 + q_3) + l_1 \cos(q_2))$$

$$P_z = l_b + l_2 \sin(q_2 + q_3) + l_3 \sin(q_2 + q_3) + l_1 \sin(q_2)$$

3.1.3.2. Cinemática Inversa:

El problema cinemático inverso consiste en determinar los valores que deben adoptar las coordenadas articulares del robot, para una posición y orientación del extremo conocidas; además, a diferencia del problema cinemático directo, en este caso la resolución no es sistemática y depende de la configuración del manipulador robótico.

Se considera conveniente utilizar para esta etapa de diseño, el método de desacoplo cinemático como solución a la cinemática inversa, debido a que se trata de un robot de más de 3 GDL. Este procedimiento plantea la separación del problema en dos cuestiones: Posición y Orientación y realiza su resolución de manera independiente y por cualquier procedimiento.

En nuestro caso se tiene un manipulador robótico de 4GDL, en donde los tres primeros definen la posición del extremo final, y se optó por resolver su cinemática inversa a través de métodos geométricos, para ello se plantea el diagrama de la Figura 3.12.

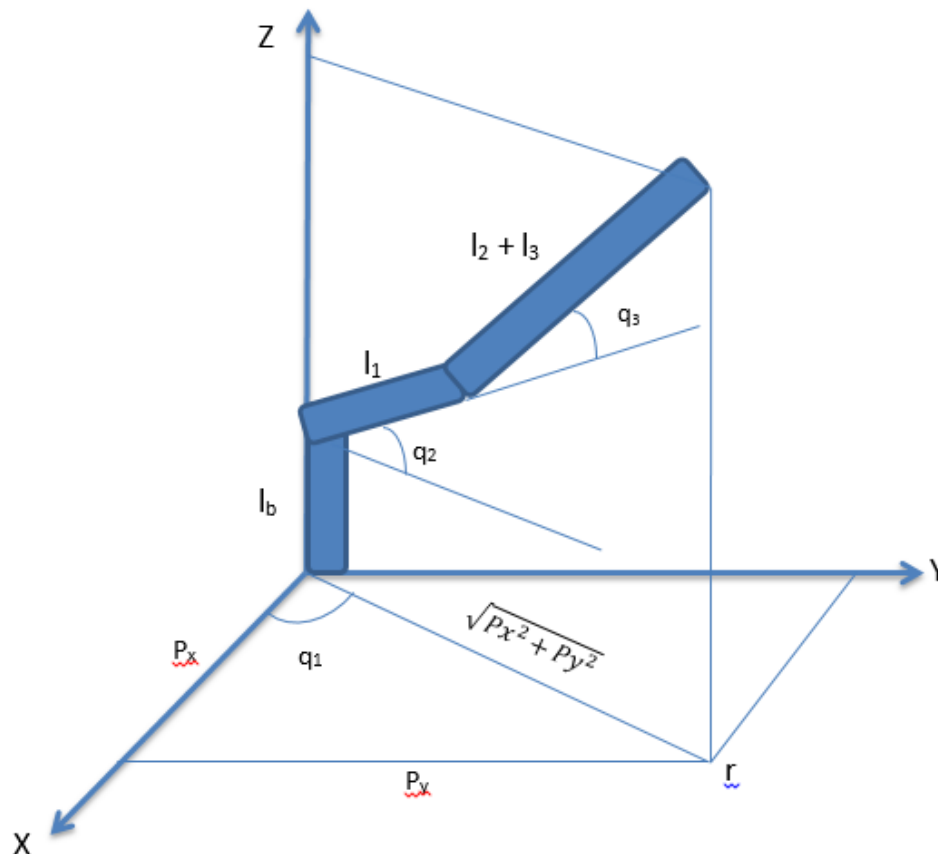


Figura 3.12: Primeros 3GDL del brazo robótico

Fuente: Propia

Haciendo referencia a la figura anterior, hallamos geoméricamente la ecuación que define la variable articular q_1 en función de los valores de la posición del extremo final del robot:

$$\tan q_1 = \frac{P_y}{P_x}$$

$$q_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right)$$

Posteriormente, para determinar la ecuación correspondiente a la articulación q_2 , se plantea el diagrama de la Figura 3.13:

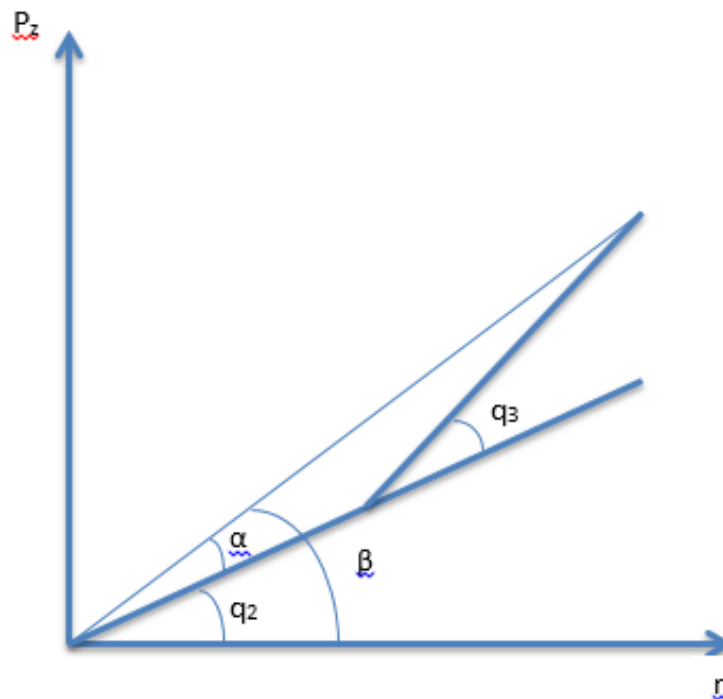


Figura 3.13: Diagrama para la obtención de la variable articular q_2

Fuente: Propia

$$\beta = \tan^{-1}\left(\frac{P_z}{r}\right) = \tan^{-1}\left(\frac{P_z}{\pm\sqrt{P_x^2 + P_y^2}}\right)$$

$$\alpha = \tan^{-1}\left(\frac{(l_2 + l_3) * \sin q_3}{l_1 + (l_2 + l_3) * \cos q_3}\right)$$

$$q_2 = \beta - \alpha$$

$$q_2 = \tan^{-1} \left(\frac{P_z}{\pm \sqrt{P_x^2 + P_y^2}} \right) - \tan^{-1} \left(\frac{(l_2 + l_3) * \text{sen } q_3}{l_1 + (l_2 + l_3) * \cos q_3} \right)$$

La Figura 3.14, es de gran ayuda para determinar la ecuación que define a la articulación q_3 , en función de las variables que describen la posición del extremo final del robot:

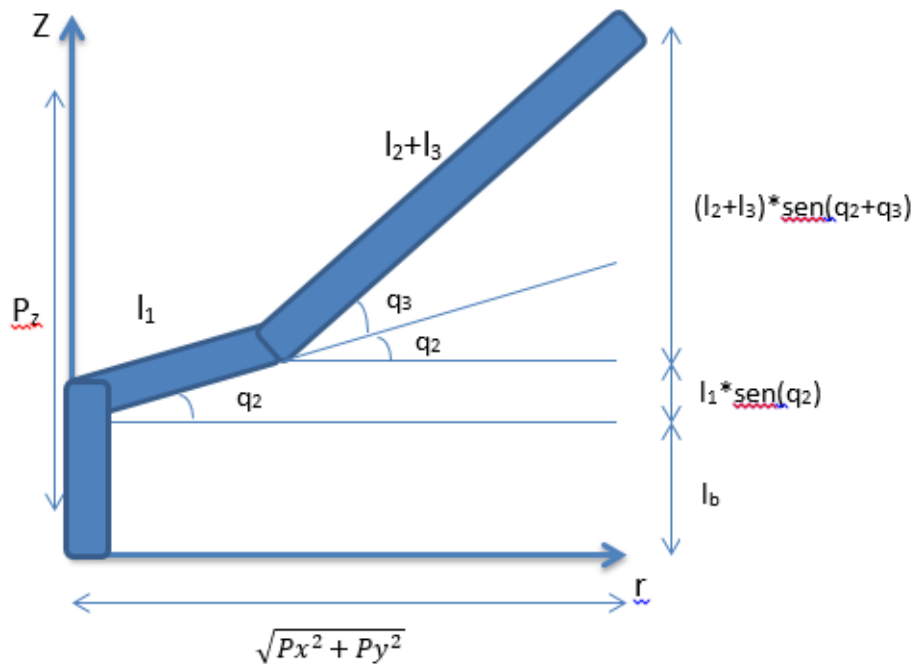


Figura 3.14: Diagrama para la obtención de la variable articular q_3

Fuente: Propia

$$\sqrt{P_x^2 + P_y^2} = l_1 * \cos q_2 + (l_2 + l_3) * \cos(q_2 + q_3)$$

$$r^2 + P_z^2 = l_1^2 + (l_2 + l_3)^2 + 2 * l_1 * (l_2 + l_3) * \cos q_3$$

$$P_x^2 + P_y^2 + P_z^2 = l_1^2 + (l_2 + l_3)^2 + 2 * l_1 * (l_2 + l_3) * \cos q_3$$

$$\cos q_3 = \frac{P_x^2 + P_y^2 + P_z^2 - l_1^2 - (l_2 + l_3)^2}{2 * l_1 * (l_2 + l_3)}$$

$$\text{sen } q_3 = \pm \sqrt{1 - \cos^2 q_3}$$

$$q_3 = \tan^{-1} \left(\frac{\pm \sqrt{1 - \cos^2 q_3}}{\cos q_3} \right)$$

Finalmente, el último grado de libertad correspondiente a la variable articular q_4 , define básicamente la orientación del extremo final del manipulador robótico y para hallar la ecuación que la determina, se utilizó la relación que existe entre la matriz de transformación homogénea, obtenida mediante el algoritmo de Denavith Hartenberg, y los ángulos de Euler WWV:

$$T = {}^0A_1 {}^1A_2 {}^2A_3 {}^3A_4 = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [3.2]$$

$$T = Rotz(\phi)Roty(\theta)Rotz(\psi) = \begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & C\phi C\theta & 0 \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\phi C\theta & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [3.3]$$

- 1) $\cos(\theta) = a_z$
 $\cos(\theta) = \sin(q_2 + q_3)$
 $\sin\left(\frac{\pi}{2} - \theta\right) = \sin(q_2 + q_3)$
 $\frac{\pi}{2} - \theta = q_2 + q_3$
 $\theta = \frac{\pi}{2} - (q_2 + q_3)$
- 2) $-\sin(\theta) * \cos(\psi) = n_z$
 $-\sin(\theta) * \cos(\psi) = \cos(q_2 + q_3) * \cos(q_4)$
 $(-\sin(\theta) * \cos(\psi))^2 = (\cos(q_2 + q_3) * \cos(q_4))^2$
 $\sin^2(\theta) * \cos^2(\psi) = \cos^2(q_2 + q_3) * \cos^2(q_4)$
 $\sin(\theta) * \sin(\psi) = o_z$
 $\sin(\theta) * \sin(\psi) = -\cos(q_2 + q_3) * \sin(q_4)$
 $(\sin(\theta) * \sin(\psi))^2 = (-\cos(q_2 + q_3) * \sin(q_4))^2$
 $\sin^2\theta * \sin^2\psi = \cos^2(q_2 + q_3) * \sin^2(q_4)$
 $\frac{\sin^2(\theta) * \sin^2(\psi)}{\sin^2(\theta) * \cos^2(\psi)} = \frac{\cos^2(q_2 + q_3) * \sin^2(q_4)}{\cos^2(q_2 + q_3) * \cos^2(q_4)}$
 $\frac{\sin^2(\psi)}{\cos^2(\psi)} = \frac{\sin^2(q_4)}{\cos^2(q_4)}$
 $tg^2(\psi) = tg^2(q_4)$
 $\psi = q_4$
- 3) $\sin(\phi) * \sin(\theta) = a_y$
 $\sin(\phi) * \sin\left(\frac{\pi}{2} - (q_2 + q_3)\right) = \sin(q_1) * \cos(q_2 + q_3)$
 $\sin(\phi) * \cos(q_2 + q_3) = \sin(q_1) * \cos(q_2 + q_3)$
 $\sin(\phi) = \sin(q_1)$
 $\phi = q_1$

3.1.3.3. Modelo Diferencial:

El modelo diferencial establece la relación entre las velocidades de las articulaciones y del extremo del robot; de tal manera, que el sistema de control puede determinar que velocidades debe imprimir a cada articulación, a través de sus actuadores, para conseguir que el extremo desarrolle una trayectoria temporal concreta. Dicha relación queda definida en la denominada matriz Jacobiana, la misma que permite también estudiar aspectos de evasión de singularidades.

- **Jacobiana Analítica Directa**

La matriz Jacobiana Analítica Directa nos permite obtener las velocidades de ubicación y orientación del efector final del robot a partir del vector de sus velocidades articulares.

Dada la resolución del problema cinemático directo, a través del algoritmo de Denavith Hartenberg, determinamos las ecuaciones que describen la posición del extremo final con respecto al sistema de referencia de la base del manipulador:

$$P_x = \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2)$$

$$P_y = \cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) + l_1 * \cos(q_2) * \sin(q_1)$$

$$P_z = l_b + l_2 * \sin(q_2 + q_3) + l_3 * \sin(q_2 + q_3) + l_1 * \sin(q_2)$$

Y haciendo uso de la relación que existe entre la matriz de transformación homogénea T y los ángulos de Euler WWV, se obtienen las ecuaciones que describen la orientación del efector final del robot:

$$\theta = \frac{\pi}{2} - (q_2 + q_3)$$

$$\psi = q_4$$

$$\phi = q_1$$

La matriz jacobiana analítica se obtiene por medio de derivación directa de cada elemento anterior con respecto a los valores articulares

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J_a \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} \quad [3.4]$$

$$J_a = \begin{bmatrix} \frac{\partial P_X}{\partial q_1} & \frac{\partial P_X}{\partial q_2} & \frac{\partial P_X}{\partial q_3} & \frac{\partial P_X}{\partial q_4} \\ \frac{\partial P_Y}{\partial q_1} & \frac{\partial P_Y}{\partial q_2} & \frac{\partial P_Y}{\partial q_3} & \frac{\partial P_Y}{\partial q_4} \\ \frac{\partial P_Z}{\partial q_1} & \frac{\partial P_Z}{\partial q_2} & \frac{\partial P_Z}{\partial q_3} & \frac{\partial P_Z}{\partial q_4} \\ \frac{\partial \phi}{\partial q_1} & \frac{\partial \phi}{\partial q_2} & \frac{\partial \phi}{\partial q_3} & \frac{\partial \phi}{\partial q_4} \\ \frac{\partial q_1}{\partial \theta} & \frac{\partial q_2}{\partial \theta} & \frac{\partial q_3}{\partial \theta} & \frac{\partial q_4}{\partial \theta} \\ \frac{\partial q_1}{\partial \psi} & \frac{\partial q_2}{\partial \psi} & \frac{\partial q_3}{\partial \psi} & \frac{\partial q_4}{\partial \psi} \end{bmatrix} \quad [3.5]$$

$$J_a = \begin{bmatrix} J_{a1} & J_{a3} & J_{a6} & 0 \\ J_{a2} & J_{a4} & J_{a7} & 0 \\ 0 & J_{a5} & J_{a8} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde:

$$J_{a1} = -\cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \cos(q_2) * \sin(q_1)$$

$$J_{a2} = \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2)$$

$$J_{a3} = -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) - l_1 * \cos(q_1) * \sin(q_2)$$

$$J_{a4} = -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \sin(q_1) * \sin(q_2)$$

$$J_{a5} = l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) + l_1 * \cos(q_2)$$

$$J_{a6} = -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3)$$

$$J_{a7} = -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3)$$

$$J_{a8} = l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3)$$

- **Jacobiana Analítica Inversa**

La matriz Jacobiana Analítica Inversa, es aquella que nos permite obtener las velocidades articulares del robot a partir de las velocidades de ubicación y orientación del extremo del mismo.

Se puede observar claramente que la Jacobiana obtenida en el punto anterior, no es una matriz cuadrada, debido a que el número de grados de libertad del robot no coincide con la dimensión del espacio de tarea, siendo este menor; de tal modo, que se tiene una matriz con mayor número de filas que de columnas, motivo por el cual, es necesario utilizar el método de la Jacobiana Pseudoinversa por la izquierda, el mismo que está definido por la siguiente operación matricial:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = J_a^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad [3.6]$$

$$J_a^{-1} = (J_a^T \cdot J_a)^{-1} \cdot J_a^T \quad [3.7]$$

$$J_a^{-1} = \begin{bmatrix} J_{a1}^{-1} & J_{a4}^{-1} & 0 & J_{a9}^{-1} & 0 & 0 \\ J_{a2}^{-1} & J_{a5}^{-1} & J_{a7}^{-1} & 0 & J_{a10}^{-1} & 0 \\ J_{a3}^{-1} & J_{a6}^{-1} & J_{a8}^{-1} & 0 & J_{a11}^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

Dada la extensión de algunos de los elementos correspondientes a la matriz, se optó por presentarlos de la siguiente forma:

$$J_{a1}^{-1} = -(2S_1 C_{23}(l_2 + l_3) + 2l_1 S_1 C_2) / (2l_2 l_3 + l_1^2 C(2q_2) + (l_2^2 + l_3^2) C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + (2l_1 l_2 + 2l_1 l_3) C_3 + (2l_1 l_2 + 2l_1 l_3) C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 2)$$

$$J_{a2}^{-1} = -(C_1(S_2 + ((l_2^2 + l_3^2) S_2) / 2 - ((l_2^2 + l_3^2) S(q_2 + 2q_3)) / 2 + l_2 l_3 (S_2 - S(q_2 + 2q_3)))) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{a3}^{-1} = (C_1(S_2 + ((l_2^2 + l_3^2) S_2) / 2 - ((l_2^2 + l_3^2) S(q_2 + 2q_3)) / 2 + l_2 l_3 S_2 + ((l_1 l_2 + l_1 l_3) S(q_2 - q_3)) / 2 - l_2 l_3 S(q_2 + 2q_3) - ((l_1 l_2 + l_1 l_3) S(q_2 + q_3)) / 2)) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{a4}^{-1} = (2C_1 C_{23}(l_2 + l_3) + 2l_1 C_1 C_2) / (2l_2 l_3 + l_1^2 C(2q_2) + (l_2^2 + l_3^2) C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + (2l_1 l_2 + 2l_1 l_3) C_3 + (2l_1 l_2 + 2l_1 l_3) C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 2)$$

$$J_{a5}^{-1} = -(S_1(S_2 + ((l_2^2 + l_3^2) S_2) / 2 - ((l_2^2 + l_3^2) S(q_2 + 2q_3)) / 2 + l_2 l_3 (S_2 - S(q_2 + 2q_3)))) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{a6}^{-1} = (S_1(S_2 + ((l_2^2 + l_3^2) S_2) / 2 - ((l_2^2 + l_3^2) S(q_2 + 2q_3)) / 2 + l_2 l_3 S_2 + ((l_1 l_2 + l_1 l_3) S(q_2 - q_3)) / 2 - l_2 l_3 S(q_2 + 2q_3) - ((l_1 l_2 + l_1 l_3) S(q_2 + q_3)) / 2)) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{a7}^{-1} = (2C_2 - (l_2^2 + l_3^2) C(q_2 + 2q_3) + (l_2^2 + l_3^2) C_2 + 2l_2 l_3 C_2 - 2l_2 l_3 C(q_2 + 2q_3)) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C(2q_3) + l_2^2 + l_3^2 - 2l_2 l_3 C(2q_3) + 2))$$

$$J_{a8}^{-1} = -(2C_2 - (l_2^2 + l_3^2) C(q_2 + 2q_3) + (l_2^2 + l_3^2) C_2 + 2l_2 l_3 C_2 + (l_1 l_2 + l_1 l_3) C(q_2 - q_3) - 2l_2 l_3 C(q_2 + 2q_3) - (l_1 l_2 + l_1 l_3) C(q_2 + q_3)) / (l_1(2l_2 l_3 - (l_2^2 + l_3^2) C(2q_3) + l_2^2 + l_3^2 + 2l_2 l_3 C(2q_3) + 2))$$

$$J_{a9}^{-1} = 1/(l_2 l_3 + (l_1^2 C(2q_2))/2 + ((l_2^2 + l_3^2)C(2q_2 + 2q_3))/2 + l_1^2/2 + l_2^2/2 + l_3^2/2 + (l_1 l_2 + l_1 l_3)C_3 + (l_1 l_2 + l_1 l_3)C(2q_2 + q_3) + l_2 l_3 C(2q_2 + 2q_3) + 1)$$

$$J_{a10}^{-1} = ((l_2 + l_3)C_3)/(l_1(2l_2 l_3 - (l_2^2 + l_3^2)C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{a11}^{-1} = -(l_1 + (l_2 + l_3)C_3)/(l_1(2l_2 l_3 - (l_2^2 + l_3^2)C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

- **Jacobiana Geométrica Directa**

La matriz Jacobiana Geométrica Directa nos permite obtener la velocidad lineal y angular del extremo del robot, con respecto a la base del mismo, a partir de las velocidades articulares.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_g \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} \quad [3.8]$$

Al igual que en la Jacobiana Analítica, partimos de la resolución del problema cinemático directo y determinamos las ecuaciones que describen la posición del extremo final:

$$P_x = \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2)$$

$$P_y = \cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) + l_1 * \cos(q_2) * \sin(q_1)$$

$$P_z = l_b + l_2 * \sin(q_2 + q_3) + l_3 * \sin(q_2 + q_3) + l_1 * \sin(q_2)$$

La matriz Jacobiana Geométrica se puede dividir en dos bloques, el primero referido a la velocidad lineal, la misma que se obtiene por medio de derivación directa de cada elemento anterior con respecto a los valores articulares:

$$J_{geo} = \begin{bmatrix} J_{gv} \\ J_{gw} \end{bmatrix}$$

$$J_{gv} = \begin{bmatrix} \frac{\partial P_x}{\partial q_1} & \frac{\partial P_x}{\partial q_2} & \frac{\partial P_x}{\partial q_3} & \frac{\partial P_x}{\partial q_4} \\ \frac{\partial P_y}{\partial q_1} & \frac{\partial P_y}{\partial q_2} & \frac{\partial P_y}{\partial q_3} & \frac{\partial P_y}{\partial q_4} \\ \frac{\partial P_z}{\partial q_1} & \frac{\partial P_z}{\partial q_2} & \frac{\partial P_z}{\partial q_3} & \frac{\partial P_z}{\partial q_4} \end{bmatrix} \quad [3.9]$$

$$J_{gv} = \begin{bmatrix} J_{gv1} & J_{gv4} & J_{gv7} & J_{gv10} \\ J_{gv2} & J_{gv5} & J_{gv8} & J_{gv11} \\ J_{gv3} & J_{gv6} & J_{gv9} & J_{gv12} \end{bmatrix}$$

$$J_{gv1} = -\cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \cos(q_2) * \sin(q_1)$$

$$\begin{aligned}
 J_{gv2} &= \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2) \\
 J_{gv3} &= 0 \\
 J_{gv4} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) - l_1 * \cos(q_1) * \sin(q_2) \\
 J_{gv5} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \sin(q_1) * \sin(q_2) \\
 J_{gv6} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) + l_1 * \cos(q_2) \\
 J_{gv7} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) \\
 J_{gv8} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) \\
 J_{gv9} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) \\
 J_{gv10} &= 0 \\
 J_{gv11} &= 0 \\
 J_{gv12} &= 0
 \end{aligned}$$

Y el segundo bloque, referido a la velocidad angular del extremo del robot, para su resolución se deben seguir los siguientes pasos:

Inicialmente hallamos las matrices de transformación homogénea, a través del algoritmo de Denavit Hartenberg, desde el punto base (A01, A02, A03):

$$A01 = \begin{bmatrix} \cos(q_1) & 0 & \sin(q_1) & 0 \\ \sin(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & l_b \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A02 = A01 * A12$$

$$A02 = \begin{bmatrix} \cos(q_1) * \cos(q_2) & -\cos(q_1) * \sin(q_2) & \sin(q_1) & l_1 * \cos(q_1) * \cos(q_2) \\ \cos(q_2) * \sin(q_1) & -\sin(q_1) * \sin(q_2) & -\cos(q_2) & l_1 * \cos(q_2) * \sin(q_1) \\ \sin(q_2) & \cos(q_2) & 0 & l_b + l_1 * \sin(q_2) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A03 = A01 * A12 * A23$$

$$A03 = \begin{bmatrix} A03_1 & A03_5 & A03_9 & A03_{13} \\ A03_2 & A03_6 & A03_{10} & A03_{14} \\ A03_3 & A03_7 & A03_{11} & A03_{15} \\ A03_4 & A03_8 & A03_{12} & A03_{16} \end{bmatrix}$$

$$\begin{aligned}
 A03_1 &= \cos(q_1) * \cos(q_2) * \cos\left(\frac{\pi}{2} + q_3\right) - \cos(q_1) * \sin(q_2) * \sin\left(\frac{\pi}{2} + q_3\right) \\
 A03_2 &= \cos(q_2) * \cos\left(\frac{\pi}{2} + q_3\right) * \sin(q_1) - \sin(q_1) * \sin(q_2) * \sin\left(\frac{\pi}{2} + q_3\right) \\
 A03_3 &= \cos(q_2) * \sin\left(\frac{\pi}{2} + q_3\right) + \cos\left(\frac{\pi}{2} + q_3\right) * \sin(q_2) \\
 A03_4 &= 0 \\
 A03_5 &= \sin(q_1) \\
 A03_6 &= -\cos(q_1)
 \end{aligned}$$

$$\begin{aligned}
 A03_7 &= 0 \\
 A03_8 &= 0 \\
 A03_9 &= \cos(q_1) * \cos(q_2) * \sin\left(\frac{\pi}{2} + q_3\right) + \cos(q_1) * \cos\left(\frac{\pi}{2} + q_3\right) * \sin(q_2) \\
 A03_{10} &= \cos(q_2) * \sin(q_1) * \sin\left(\frac{\pi}{2} + q_3\right) + \cos\left(\frac{\pi}{2} + q_3\right) * \sin(q_1) * \sin(q_2) \\
 A03_{11} &= \sin(q_2) * \sin\left(\frac{\pi}{2} + q_3\right) - \cos(q_2) * \cos\left(\frac{\pi}{2} + q_3\right) \\
 A03_{12} &= 0 \\
 A03_{13} &= l_1 * \cos(q_1) * \cos(q_2) \\
 A03_{14} &= l_1 * \cos(q_2) * \sin(q_1) \\
 A03_{15} &= l_b + l_1 * \sin(q_2) \\
 A03_{16} &= 1
 \end{aligned}$$

Posteriormente obtenemos los vectores Z0, Z01, Z02 y Z03 correspondientes a la componente en “z” de cada una de las matrices halladas en el paso anterior:

$$\begin{aligned}
 Z0 &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
 Z01 &= A01(1:3,3) = \begin{bmatrix} \sin(q_1) \\ -\cos(q_1) \\ 0 \end{bmatrix} \\
 Z02 &= A02(1:3,3) = \begin{bmatrix} \sin(q_1) \\ -\cos(q_1) \\ 0 \end{bmatrix} \\
 Z03 &= A03(1:3,3) \\
 Z03 &= \begin{bmatrix} -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \cos(q_1) \\ -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \sin(q_1) \\ \cos\left(q_2 - \frac{\pi}{2} + q_3\right) \end{bmatrix}
 \end{aligned}$$

Finalmente ordenamos los elementos del segundo bloque de la Matriz:

$$\begin{aligned}
 J_{gw} &= \begin{bmatrix} Z0(1,1) & Z01(1,1) & Z02(1,1) & Z03(1,1) \\ Z0(2,1) & Z01(2,1) & Z02(2,1) & Z03(2,1) \\ Z0(3,1) & Z01(3,1) & Z02(3,1) & Z03(3,1) \end{bmatrix} \\
 J_{gw} &= \begin{bmatrix} J_{gw1} & J_{gw4} & J_{gw7} & J_{gw10} \\ J_{gw2} & J_{gw5} & J_{gw8} & J_{gw11} \\ J_{gw3} & J_{gw6} & J_{gw9} & J_{gw12} \end{bmatrix}
 \end{aligned}$$

$$\begin{aligned}
 J_{gw1} &= 0 \\
 J_{gw2} &= 0 \\
 J_{gw3} &= 1 \\
 J_{gw4} &= \sin(q_1) \\
 J_{gw5} &= -\cos(q_1) \\
 J_{gw6} &= 0
 \end{aligned}$$

$$\begin{aligned}J_{gw7} &= \text{sen}(q_1) \\J_{gw8} &= -\text{cos}(q_1) \\J_{gw9} &= 0 \\J_{gw10} &= \text{cos}(q_2 + q_3) * \text{cos}(q_1) \\J_{gw11} &= \text{cos}(q_2 + q_3) * \text{sen}(q_1) \\J_{gw12} &= \text{sen}(q_2 + q_3)\end{aligned}$$

- **Jacobiana Geométrica Inversa**

La matriz Jacobiana Geométrica Inversa, es aquella que nos posibilita obtener las velocidades articulares del robot a partir de la velocidad lineal y angular del extremo del mismo, con respecto a su base.

Al igual que la Jacobiana Analítica, se puede observar que la Matriz Jacobiana Geométrica obtenida, no es cuadrada, teniendo así un mayor número de filas que de columnas, motivo por el cual, es necesario utilizar el método de la Jacobiana Pseudoinversa por la izquierda, definido por el siguiente producto matricial:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = J_{geo}^{-1} \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \quad [3.10]$$

$$J_{geo}^{-1} = (J_{geo}^T \cdot J_{geo})^{-1} \cdot J_{geo}^T \quad [3.11]$$

$$J_{geo}^{-1} = \begin{bmatrix} J_{g1}^{-1} & J_{g5}^{-1} & 0 & J_{g11}^{-1} & J_{g15}^{-1} & J_{g19}^{-1} \\ J_{g2}^{-1} & J_{g6}^{-1} & J_{g9}^{-1} & J_{g12}^{-1} & J_{g16}^{-1} & 0 \\ J_{g3}^{-1} & J_{g7}^{-1} & J_{g10}^{-1} & J_{g13}^{-1} & J_{g17}^{-1} & 0 \\ J_{g4}^{-1} & J_{g8}^{-1} & 0 & J_{g14}^{-1} & J_{g18}^{-1} & J_{g20}^{-1} \end{bmatrix}$$

Debido a la extensión de las componentes de la matriz, es que se decidió presentarlas de la siguiente manera:

$$\begin{aligned}J_{g1}^{-1} &= -(2S_1(l_2C(q_2 + q_3) + l_3C(q_2 + q_3) + l_1C_2))/(2C(q_2 + q_3) + 2l_2l_3 + l_1^2C(2q_2) + 2l_2^2C(q_2 + q_3) + 2l_3^2C(q_2 + q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1l_2C_3 + 2l_1l_3C_3 + 2l_1l_2C(2q_2 + q_3) + 2l_1l_3C(2q_2 + q_3) + 4l_2l_3C(q_2 + q_3) + 1)) \\J_{g2}^{-1} &= -(C_1(S_1 + l_2^2S_2)/2 + (l_3^2S_2)/2 - (l_2^2S(q_2 + 2q_3))/2 - (l_3^2S(q_2 + 2q_3))/2 + l_2l_3S_2 - l_2l_3S(q_2 + 2q_3))/(l_1(2l_2l_3 - l_2^2C_3^2 - l_3^2C_3^2 + l_2^2 + l_3^2 - 2l_2l_3C_3^2 + 1))\end{aligned}$$

$$J_{g3}^{-1} = (C_1(S_2 + l_2^2 S_2)/2 + ((l_3^2 S_2)/2 - (l_2^2 S(q_2 + 2q_3))/2 - (l_3^2 S(q_2 + 2q_3))/2 + l_2 l_3 S_2 + (l_1 l_2 S(q_2 - q_3))/2 + (l_1 l_3 S(q_2 - q_3))/2 - l_2 l_3 S(q_2 + 2q_3) - (l_1 l_2 S(q_2 + q_3))/2 - (l_1 l_3 S(q_2 + q_3))/2))/((l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1)))$$

$$J_{g4}^{-1} = (S_1(2l_2 S(q_2 + q_3) + 2l_3 S(q_2 + q_3) + l_1 S_3 + 2l_1 S(q_2 + q_3)))/(2C(q_2 + q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + 2l_2^2 C(q_2 + q_3) + 2l_3^2 C(q_2 + q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_3 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 4l_2 l_3 C(q_2 + q_3) + 1)$$

$$J_{g5}^{-1} = (2C_1(l_2 C(q_2 + q_3) + l_3 C(q_2 + q_3) + l_1 C_2))/(2C(q_2 + q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + 2l_2^2 C(q_2 + q_3) + 2l_3^2 C(q_2 + q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_3 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 4l_2 l_3 C(q_2 + q_3) + 1)$$

$$J_{g6}^{-1} = -(S_1(S_2 + l_2^2 S_2)/2 + (l_3^2 S_2)/2 - (l_2^2 S(q_2 + 2q_3))/2 - (l_3^2 S(q_2 + 2q_3))/2 + l_2 l_3 S_2 - l_2 l_3 S(q_2 + 2q_3))/((l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1)))$$

$$J_{g7}^{-1} = (S_1(S_2 + l_2^2 S_2)/2 + (l_3^2 S_2)/2 - (l_2^2 S(q_2 + 2q_3))/2 - (l_3^2 S(q_2 + 2q_3))/2 + l_2 l_3 S_2 + (l_1 l_2 S(q_2 - q_3))/2 + (l_1 l_3 S(q_2 - q_3))/2 - l_2 l_3 S(q_2 + 2q_3) - (l_1 l_2 S(q_2 + q_3))/2 - (l_1 l_3 S(q_2 + q_3))/2))/((l_1 * (l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1)))$$

$$J_{g8}^{-1} = -(C_1(l_2 S(2q_2 + 2q_3) + l_3 S(2q_2 + 2q_3) + l_1 S_3 + l_1 S(2q_2 + 2q_3)))/(C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_3 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1)$$

$$J_{g9}^{-1} = (2C_2 - l_2^2 C(q_2 + 2q_3) - l_3^2 C(q_2 + 2q_3) + l_2^2 C_2 + l_3^2 C_2 + 2l_2 l_3 C_2 - 2l_2 l_3 C(q_2 + 2q_3))/((l_1(2l_2 l_3 - l_2^2 C(2q_3) - l_3^2 C(2q_3) + l_2^2 + l_3^2 - 2l_2 l_3 C(2q_3) + 2))$$

$$J_{g10}^{-1} = -(2C_2 - l_2^2 C(q_2 + 2q_3) - l_3^2 C(q_2 + 2q_3) + l_2^2 C_2 + l_3^2 C_2 + 2l_2 l_3 C_2 + l_1 l_2 C(q_2 - q_3) + l_1 l_3 C(q_2 - q_3) - 2l_2 l_3 C(q_2 + 2q_3) - l_1 l_2 C(q_2 + q_3) - l_1 l_3 C(q_2 + q_3))/((l_1(2l_2 l_3 - l_2^2 C(2q_3) - l_3^2 C(2q_3) + l_2^2 + l_3^2 - 2l_2 l_3 C(2q_3) + 2))$$

$$J_{g11}^{-1} = (S(q_1 - 2q_2 - 2q_3)/2 - S(q_1 + 2q_2 + 2q_3)/2)/(C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C(q_3) + 2l_1 l_3 C(q_3) + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1)$$

$$J_{g12}^{-1} = -(C_3 S_1(l_2 + l_3))/((l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1)))$$

$$J_{g13}^{-1} = (S_1(l_1 + l_2 C_3 + l_3 C_3)) / (l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{g14}^{-1} = (C_1(4C(q_2 + q_3) + l_1^2 C(q_2 - q_3) + l_1^2 C(3q_2 + q_3) + l_2^2 C(3q_2 + 3q_3) + l_3^2 C(3q_2 + 3q_3) + 2l_1^2 C(q_2 + q_3) + 3l_2^2 C_2 + q_3) + 3l_3^2 C(q_2 + q_3) + 4l_1 l_2 C_2 + 4l_1 l_3 C_2 + 2l_1 l_2 C(q_2 + 2q_3) + 2l_1 l_3 C(q_2 + 2q_3) + 2l_1 l_2 C(3q_2 + 2q_3) + 2l_1 l_3 C(3q_2 + 2q_3) + 2l_2 l_3 C(3q_2 + 3q_3) + 26l_3 C(q_2 + q_3))) / (2(C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_2 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1))$$

$$J_{g15}^{-1} = -(C(q_1 - 2q_2 - 2q_3)/2 - C(q_1 + 2q_2 + 2q_3)/2) / (C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C(q_3) + 2l_1 l_3 C(q_3) + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1)$$

$$J_{g16}^{-1} = (C_1 C_3 (l_2 + l_3)) / (l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{g17}^{-1} = -(C_1(l_1 + l_2 C_3 + l_3 C_3)) / (l_1(2l_2 l_3 - l_2^2 C_3^2 - l_3^2 C_3^2 + l_2^2 + l_3^2 - 2l_2 l_3 C_3^2 + 1))$$

$$J_{g18}^{-1} = (S_1(4C(q_2 + q_3) + l_1^2 C(q_2 - q_3) + l_1^2 C(3q_2 + q_3) + l_2^2 C(3q_2 + 3q_3) + l_3^2 C(3q_2 + 3q_3) + 2l_1^2 C(q_2 + q_3) + 3l_2^2 C_2 + q_3) + 3l_3^2 C(q_2 + q_3) + 4l_1 l_2 C_2 + 4l_1 l_3 C_2 + 2l_1 l_2 C(q_2 + 2q_3) + 2l_1 l_3 C(q_2 + 2q_3) + 2l_1 l_2 C(3q_2 + 2q_3) + 2l_1 l_3 C(3q_2 + 2q_3) + 2l_2 l_3 C(3q_2 + 3q_3) + 6l_2 l_3 C(q_2 + q_3))) / (2(C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_2 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1))$$

$$J_{g19}^{-1} = (C(2q_2 + 2q_3) + 1) / (C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_2 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1)$$

$$J_{g20}^{-1} = (l_1^2 S(3q_2 + q_3) - l_1^2 S(q_2 - q_3) + l_2^2 S(3q_2 + 3q_3) + l_3^2 S(3q_2 + 3q_3) + 2l_1^2 S(q_2 + q_3) + l_2^2 S(q_2 + q_3) + l_3^2 S(q_2 + q_3) + 2l_1 l_2 S(q_2 + 2q_3) + 2l_1 l_3 S(q_2 + 2q_3) + 2l_1 l_2 S(3q_2 + 2q_3) + 2l_1 l_3 S(3q_2 + 2q_3) + 2l_2 l_3 S(3q_2 + 3q_3) + 2l_2 l_3 S(q_2 + q_3)) / (2(C(2q_2 + 2q_3) + 2l_2 l_3 + l_1^2 C(2q_2) + l_2^2 C(2q_2 + 2q_3) + l_3^2 C(2q_2 + 2q_3) + l_1^2 + l_2^2 + l_3^2 + 2l_1 l_2 C_3 + 2l_1 l_2 C_3 + 2l_1 l_2 C(2q_2 + q_3) + 2l_1 l_3 C(2q_2 + q_3) + 2l_2 l_3 C(2q_2 + 2q_3) + 1))$$

- **Relación entre la Matriz Jacobiana Analítica y la Matriz Jacobiana Geométrica**

A manera de comprobar los resultados obtenidos en los apartados anteriores, se utiliza la correspondencia existente entre ambas matrices Jacobiana, Geométrica y Analítica, la misma que viene representada por la siguiente ecuación:

$$J_{geo} = \begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix} * J_a \quad [3.12]$$

Siendo

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad [3.13]$$

$$Q = \begin{bmatrix} 0 & -\sin(\phi) & \sin(\theta) * \cos(\phi) \\ 0 & \cos(\phi) & \sin(\theta) * \sin(\phi) \\ 1 & 0 & \cos(\theta) \end{bmatrix} \quad [3.14]$$

Donde (ψ, ϕ, θ) son los ángulos de Euler WWV, hallados anteriormente

$$\theta = \frac{\pi}{2} - (q_2 + q_3)$$

$$\psi = q_4$$

$$\phi = q_1$$

Reemplazamos los ángulos de Euler en la Ecuación 3.14

$$Q = \begin{bmatrix} 0 & -\sin(q_1) & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \cos q_1 \\ 0 & \cos(q_1) & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \sin q_1 \\ 1 & 0 & \cos\left(q_2 - \frac{\pi}{2} + q_3\right) \end{bmatrix}$$

Finalmente sustituimos las matrices I, Q y J_a para obtener la Matriz Jacobiana Geométrica a partir de la Matriz Jacobiana Analítica

$$J_{geo} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\sin(q_1) & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \cos(q_1) \\ 0 & \cos(q_1) & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \sin(q_1) \\ 1 & 0 & \cos\left(q_2 - \frac{\pi}{2} + q_3\right) \end{bmatrix} * J_a$$

$$J_{geo} = \begin{bmatrix} J_{geo1} & J_{geo3} & J_{geo6} & 0 \\ J_{geo2} & J_{geo4} & J_{geo7} & 0 \\ 0 & J_{geo5} & J_{geo8} & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \cos(q_1) \\ 0 & \sin(q_1) & \sin(q_1) & -\sin\left(q_2 - \frac{\pi}{2} + q_3\right) * \sin(q_1) \\ 0 & -\cos(q_1) & -\cos(q_1) & \cos\left(q_2 - \frac{\pi}{2} + q_3\right) \\ 1 & 0 & 0 & \cos\left(q_2 - \frac{\pi}{2} + q_3\right) \end{bmatrix}$$

Donde:

$$\begin{aligned}
 J_{geo1} &= -\cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \cos(q_2) * \sin(q_1) \\
 J_{geo2} &= \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2) \\
 J_{geo3} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) - l_1 * \cos(q_1) * \sin(q_2) \\
 J_{geo4} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \sin(q_1) * \sin(q_2) \\
 J_{geo5} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) + l_1 * \cos(q_2) \\
 J_{geo6} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) \\
 J_{geo7} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) \\
 J_{geo8} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3)
 \end{aligned}$$

Al observar el resultado, vemos que la Matriz Jacobiana Geométrica hallada numéricamente y la obtenida mediante la relación con la Matriz Jacobiana Analítica es la misma, lo que nos lleva a concluir que el procedimiento realizado fue correcto.

• Configuraciones Singulares

Las configuraciones singulares son aquellos puntos en donde el determinante de la Matriz Jacobiana se anula. Básicamente es posible identificar dos tipos de singularidades, aquellas que se presentan en los límites del espacio de trabajo del robot cuando éste se encuentra completamente distendido o retraído, y aquellas que se encuentran en el interior del volumen de trabajo cuando ocurre el alineamiento de dos o más ejes de las articulaciones del manipulador, lo que se traduce en la pérdida efectiva de uno o más grados de libertad.

Es trascendental prestar especial atención al estudio y análisis de la localización de configuraciones singulares del manipulador para que sean tomadas en cuenta durante la realización del control, debido a que cuando el brazo robótico adopta una singularidad pierde su movilidad además de la posibilidad de provocar movimientos a velocidades inabordables y cambios bruscos en los actuadores; en este sentido, entre más distanciado esté el robot de estas configuraciones, podrá desplazarse mejor de manera uniforme y podrá aplicar fuerzas uniformemente en todas las direcciones.

Para calcular las singularidades del brazo robótico, se parte de la definición dada, tal como se muestra en la ecuación 3.15

$$DET(J) = 0 \quad [3.15]$$

Es importante considerar que para hallar la determinante de la Matriz Jacobiana, ésta debe ser cuadrada; teniendo así, en nuestro caso dos formas de trabajar:

1. Hallar la matriz Pseudoinversa
2. Perder algún grado de libertad según algunas suposiciones.

El robot por diseñar es de 4GDL, siendo el último referido a la orientación e independiente del resto de articulaciones, motivo por el cual se decidió trabajar

mediante el segundo método y eliminar así este grado de libertad, quedando la nueva Matriz Jacobiana de la siguiente manera:

$$J_{3 \times 3} = \begin{bmatrix} J_{3 \times 3.1} & J_{3 \times 3.4} & J_{3 \times 3.7} \\ J_{3 \times 3.2} & J_{3 \times 3.5} & J_{3 \times 3.8} \\ J_{3 \times 3.3} & J_{3 \times 3.6} & J_{3 \times 3.9} \end{bmatrix}$$

Donde:

$$\begin{aligned} J_{3 \times 3.1} &= -\cos(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \cos(q_2) * \sin(q_1) \\ J_{3 \times 3.2} &= \cos(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) + l_1 * \cos(q_1) * \cos(q_2) \\ J_{3 \times 3.3} &= 0 \\ J_{3 \times 3.4} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) - l_1 * \cos(q_1) * \sin(q_2) \\ J_{3 \times 3.5} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) - l_1 * \sin(q_1) * \sin(q_2) \\ J_{3 \times 3.6} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) + l_1 * \cos(q_2) \\ J_{3 \times 3.7} &= -\sin(q_2 + q_3) * \cos(q_1) * (l_2 + l_3) \\ J_{3 \times 3.8} &= -\sin(q_2 + q_3) * \sin(q_1) * (l_2 + l_3) \\ J_{3 \times 3.9} &= l_2 * \cos(q_2 + q_3) + l_3 * \cos(q_2 + q_3) \end{aligned}$$

Hallamos y simplificamos la determinante de la nueva matriz

$$\begin{aligned} DET(J_{3 \times 3}) &= -(l_1 * (l_2 + l_3) * (l_1 * \sin(q_2 + q_3) - l_2 * \sin(q_2) - l_3 * \sin(q_2) \\ &\quad - l_1 * \sin(q_2 - q_3) + l_2 * \sin(q_2 + 2q_3) + l_3 * \sin(q_2 + 2q_3))) / 2 \end{aligned}$$

Finalmente, según la ecuación 3.15, igualamos la determinante a 0 para obtener así las singularidades:

$$q_2 = \pm \pi / 2 \qquad q_3 = 0 \text{ ó } \pi$$

• Simulación Cinemática

Matlab es una herramienta de programación de alto nivel, orientada a llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y visualización gráfica de los mismos. En la actualidad dispone de una amplia gama de programas de apoyo especializados denominados Toolboxes, los que extienden el número de funciones incorporadas en el entorno principal, cubriendo casi todas las áreas de la ingeniería y la simulación.

Específicamente en el desarrollo de la presente tesis, se hizo uso del Robotics Toolbox for Matlab de Peter Corke versión 9.10, como herramienta para llevar a cabo la simulación de la cinemática directa e inversa del manipulador robótico y comprobar así los resultados obtenidos tras el análisis cinemático realizado, razón por la cual se realizó dos script, el primero denominado “petercorke”, en el que se empleó diferentes comandos propios del Toolbox para crear un entorno de

visualización agradable, donde es posible observar la animación del movimiento del robot a partir del desplazamiento de los sliders correspondientes a las coordenadas articulares, y el segundo denominado “cinematicadi”, que nos muestra en la ventana Command Window los valores de la posición y orientación del extremo final obtenidos mediante el reemplazo de los valores articulares en las ecuaciones propias de la cinemática. La Fig. 3.15 muestra diversas configuraciones angulares elegidas aleatoriamente.

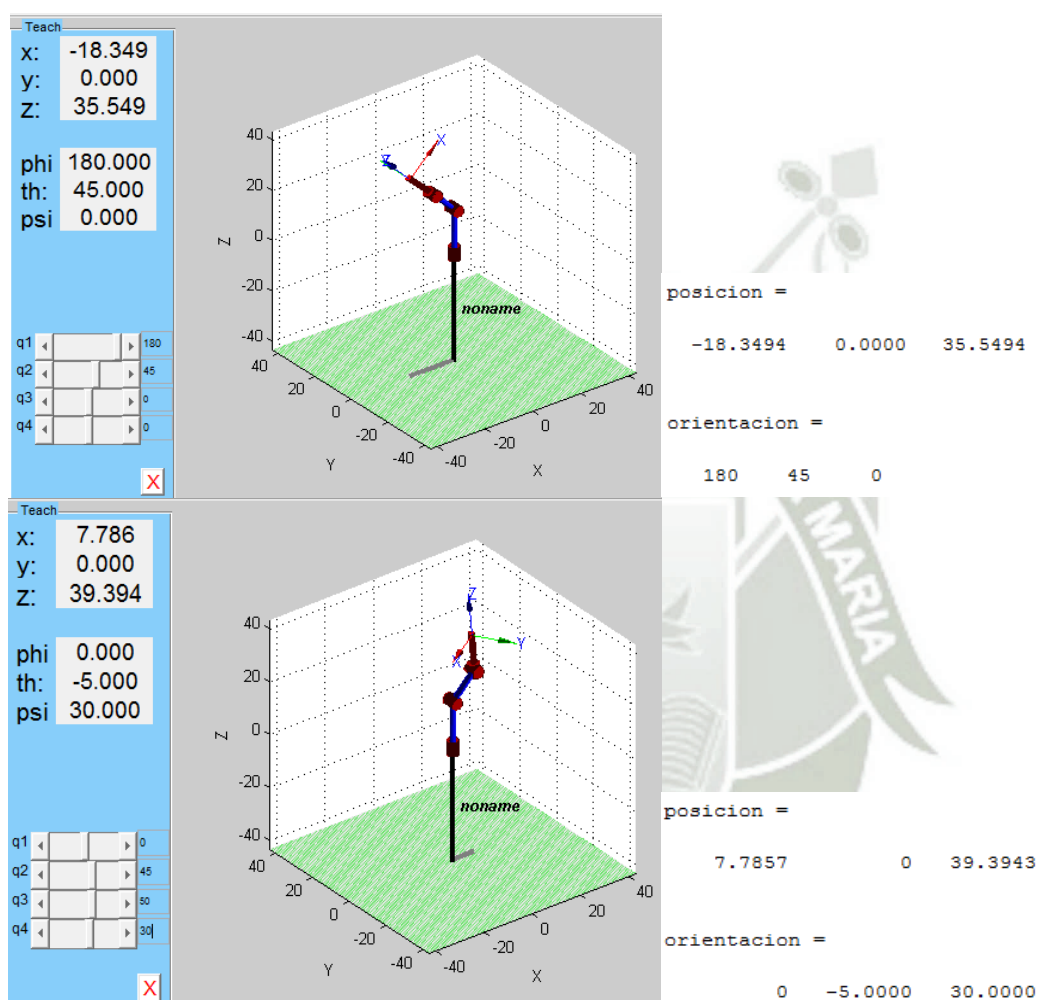


Figura 3.15: Configuraciones aleatorias aplicando el Robotics Toolbox para comprobar el análisis cinemático.

Fuente: Propia

3.1.4. Análisis Dinámico:

El estudio cinemático describe el movimiento del robot, mas no aporta información acerca de las causas que lo provocan, razón por la cual es necesario realizar un análisis dinámico. En el presente epígrafe, se realiza el modelo dinámico del robot, teniendo como objetivo principal el diseño y evaluación de la estructura mecánica del manipulador además del dimensionamiento de los actuadores, para lo cual se debe establecer las relaciones matemáticas entre las coordenadas articulares o del extremo

del robot, sus derivadas (velocidad y aceleración), los pares aplicados en las articulaciones o en el efector final y sus parámetros dimensionales como longitud, masas e inercias de sus elementos.

Existen principalmente dos métodos para obtener las ecuaciones dinámicas: el método de Newton-Euler y el método de Lagrange-Euler. Se optó por trabajar con la Formulación Lagrangiana, la misma que permite describir la dinámica del robot a partir de un balance de energías.

- **Formulación Lagrange-Euler**

Este procedimiento considera al robot como una caja negra, donde únicamente se tiene en cuenta la energía almacenada, que se expresa en términos de energía potencial y cinética; por lo tanto, el Lagrangiano es una función escalar que se define como la diferencia entre dichas energías en función de las variables articulares, tal como se expresa en la ecuación 3.16.

$$\mathcal{L}(q, \dot{q}) = K(q, \dot{q}) - U(q) \quad [3.16]$$

A partir de la última ecuación, se puede definir la relación que determina los pares aplicados en los actuadores del robot:

$$\tau = \frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{q}} \right) - \frac{\partial \mathcal{L}}{\partial q} \quad [3.17]$$

Dados los términos generales, iniciamos el análisis dinámico, hallando la energía cinética y potencial de cada uno de los eslabones del manipulador, tomando como referencia sus centros de masa, para luego obtener las energías totales del robot. En el desarrollo del estudio dinámico de cada elemento que conforma la cadena cinemática, se tendrá en cuenta para obtener la energía cinética, dos términos, el primero referido a la velocidad lineal y el segundo a la velocidad angular como se muestra en la ecuación 3.18, por esta razón se halla primero la matriz Jacobiana Geométrica correspondiente al eslabón en estudio. Así mismo en cuanto a la energía potencial se halla cada término de la ecuación 3.19, tomando en cuenta los gráficos realizados de cada uno de los elementos.

$$E_{C_i} = \frac{1}{2} m_i v_{C_i}^{C^T} v_{C_i}^C + \frac{1}{2} w_i^{C^T} I_i^C w_i^C \quad [3.18]$$

$$E_{P_i} = m_i g h_i \quad [3.19]$$

○ **Eslabón 1**

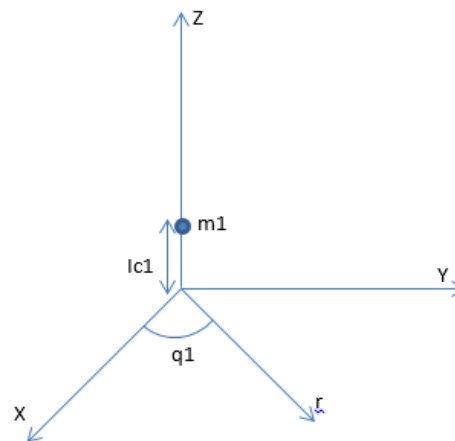


Figura 3.16: Representación del 1er eslabón para análisis dinámico

Fuente: Propia

Obtenemos la Matriz Jacobiana geométrica del primer eslabón, a partir de su definición vista anteriormente:

$$J_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Hallamos el valor de la velocidad lineal y angular multiplicando la matriz Jacobiana por los valores de las velocidades articulares:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dot{q}_1 \end{bmatrix}$$

Donde:

$$v_1 = \sqrt{v_x^2 + v_y^2 + v_z^2} = 0$$

$$w_1 = \sqrt{w_x^2 + w_y^2 + w_z^2} = \dot{q}_1^2$$

Reemplazamos las velocidades lineal y angular en la ecuación 3.18, para obtener la energía cinética del eslabón 1:

$$K_1 = \frac{I_1 \dot{q}_1^2}{2}$$

Tomando en cuenta la Figura 3.16 y la ecuación 3.19, encontramos la energía potencial del elemento en estudio:

$$U_1 = l_{c1} g m_1$$

○ Eslabón 2

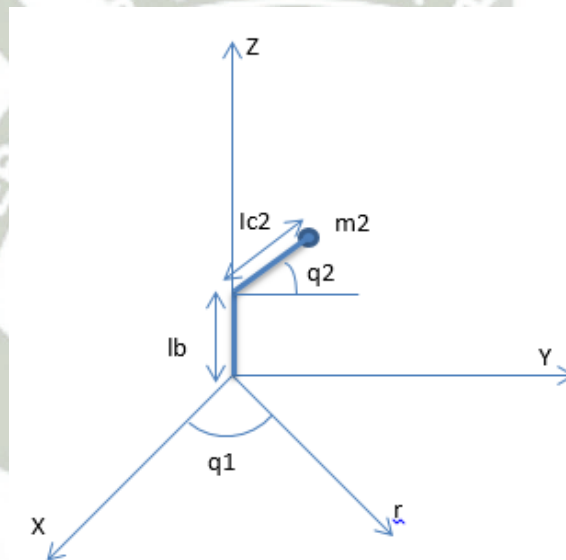


Figura 3.17: Representación del 2do eslabón para análisis dinámico

Fuente: Propia

Obtenemos la Matriz Jacobiana geométrica del segundo eslabón, a partir de su definición:

$$J_2 = \begin{bmatrix} J_{21} & J_{23} & 0 & 0 \\ J_{22} & J_{24} & 0 & 0 \\ 0 & J_{25} & 0 & 0 \\ 0 & J_{26} & 0 & 0 \\ 0 & J_{27} & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{21} = -l_{c2} C_2 S_1$$

$$J_{22} = l_{c2} C_1 C_2$$

$$J_{23} = -l_{c2} C_1 S_2$$

$$\begin{aligned} J_{24} &= -l_{c2}S_1S_2 \\ J_{25} &= l_{c2}C_2 \\ J_{26} &= S_1 \\ J_{27} &= -C_1 \end{aligned}$$

Hallamos el valor de la velocidad lineal y angular multiplicando la matriz Jacobiana por los valores de las velocidades articulares:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_2 * \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} v_{2x} \\ v_{2y} \\ v_{2z} \\ w_{2x} \\ w_{2y} \\ w_{2z} \end{bmatrix}$$

$$\begin{aligned} v_{2x} &= -l_{c2}\dot{q}_1C_2S_1 - l_{c2}\dot{q}_2C_1S_2 \\ v_{2y} &= l_{c2}\dot{q}_1C_1C_2 - l_{c2}\dot{q}_2S_1S_2 \\ v_{2z} &= l_{c2}\dot{q}_2C_2 \\ w_{2x} &= \dot{q}_2S_1 \\ w_{2y} &= -\dot{q}_2C_1 \\ w_{2z} &= \dot{q}_1 \end{aligned}$$

Donde:

$$v_2 = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

$$v_2 = \frac{l_{c2}^2 * (\dot{q}_1^2 \cos(2q_2) + \dot{q}_1^2 + 2\dot{q}_2^2)}{2}$$

$$w_2 = \sqrt{w_x^2 + w_y^2 + w_z^2}$$

$$w_2 = \dot{q}_1^2 + \dot{q}_2^2$$

Reemplazamos ambas velocidades en la ecuación 3.18, para obtener la energía cinética del eslabón 2:

$$K_2 = (m_2(\dot{q}_1^2 C(2q_2) + \dot{q}_1^2 + 2\dot{q}_2^2)l_{c2}^2)/4 + (I_2(\dot{q}_1^2 + \dot{q}_2^2))/2$$

Tomando en cuenta la Figura 3.17 y la ecuación 3.19, encontramos la energía potencial del elemento en estudio:

$$U_2 = gm_2(l_b + l_{c2}S_2)$$

○ **Eslabón 3**

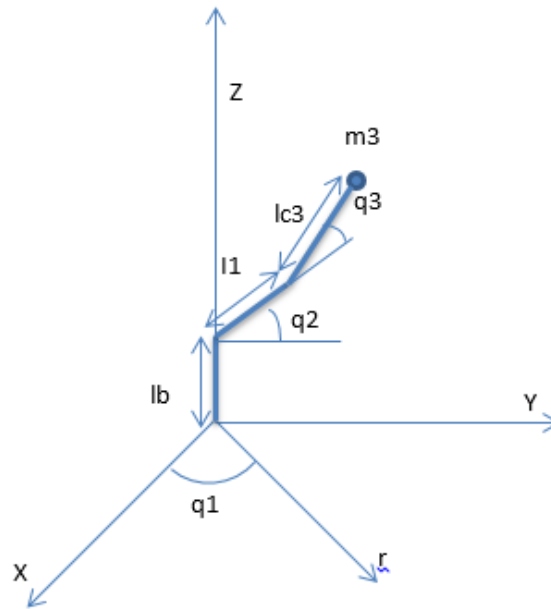


Figura 3.18: Representación del 3er eslabón para análisis dinámico

Fuente: Propia

Teniendo en cuenta la Figura 3.18, encontramos geoméricamente la posición del centro de masa del eslabón 3:

$$\begin{aligned} r &= l_1 C_2 + l_{c3} C(q_3 + q_2) \\ x &= r C_1 \\ y &= r S_1 \\ z &= l_b + l_1 S_2 + l_{c3} S(q_3 + q_2) \end{aligned}$$

Derivamos cada una de las ecuaciones obtenidas en función del tiempo para hallar luego la velocidad lineal del eslabón:

$$v_x = -l_{c3}S_{23}C_1(\dot{q}_2 + \dot{q}_3) - l_{c3}\dot{q}_1C_{23}S_1 - l_1\dot{q}_1C_2S_1 - l_1\dot{q}_2C_1S_2$$

$$v_y = -l_{c3}\dot{q}_1C_{23}C_1 - l_{c3}S_{23}S_1(\dot{q}_2 + \dot{q}_3) - l_1\dot{q}_2S_1S_2 - l_1\dot{q}_1C_1C_2$$

$$v_z = l_1\dot{q}_2C_2 + l_{c3}C_{23}(\dot{q}_2 + \dot{q}_3)$$

$$v_3 = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

$$v_3 = (l_1^2 \dot{q}_1^2)/2 + l_1^2 \dot{q}_2^2 + (l_{c3}^2 \dot{q}_1^2)/2 + l_{c3}^2 \dot{q}_2^2 + l_{c3}^2 \dot{q}_3^2 + 2l_{c3}^2 \dot{q}_2 \dot{q}_3 + (l_1^2 \dot{q}_1^2 C(2q_2))/2 + (l_{c3}^2 \dot{q}_1^2 C(2q_2 + 2q_3))/2 + l_1 l_{c3} \dot{q}_1^2 C(2q_2 + q_3) + l_1 l_{c3} \dot{q}_1^2 C_3 + 2l_1 l_{c3} \dot{q}_2^2 C_3 + 2l_1 l_{c3} \dot{q}_2 \dot{q}_3 C_3$$

Posteriormente, calculamos la matriz Jacobiana Geométrica del elemento en estudio:

$$J_3 = \begin{bmatrix} J_{31} & J_{33} & 0 & 0 \\ J_{32} & J_{34} & 0 & 0 \\ 0 & J_{35} & 0 & 0 \\ 0 & J_{36} & J_{38} & 0 \\ 0 & J_{37} & J_{39} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$J_{31} = -l_1 C_2 S_1$$

$$J_{32} = l_1 C_1 C_2$$

$$J_{33} = -l_1 C_1 S_2$$

$$J_{34} = -l_1 S_1 S_2$$

$$J_{35} = l_1 C_2$$

$$J_{36} = S_1$$

$$J_{37} = -C_1$$

$$J_{38} = S_1$$

$$J_{39} = -C_1$$

Multiplicamos la matriz Jacobiana por los valores de las velocidades articulares, para obtener así la velocidad angular del eslabón 3:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_3 * \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} v_{3x} \\ v_{3y} \\ v_{3z} \\ w_{3x} \\ w_{3y} \\ w_{3z} \end{bmatrix}$$

$$w_{3x} = \dot{q}_2 S_1 + \dot{q}_3 S_1$$

$$w_{3y} = -\dot{q}_2 C_1 - \dot{q}_3 C_1$$

$$w_{3z} = \dot{q}_1$$

$$w_3 = \sqrt{w_x^2 + w_y^2 + w_z^2}$$

$$w_3 = \dot{q}_1^2 + \dot{q}_2^2 + 2\dot{q}_2\dot{q}_3 + \dot{q}_3^2$$

Reemplazamos las velocidades lineal y angular del eslabón en la ecuación 3.18, para hallar la energía cinética:

$$K_3 = (m_3((l_1^2\dot{q}_1^2/2 + l_1^2\dot{q}_2^2 + (l_{c3}^2\dot{q}_1^2)/2 + l_{c3}^2\dot{q}_2^2 + l_{c3}^2\dot{q}_3^2 + 2l_{c3}^2\dot{q}_2\dot{q}_3 + (l_1^2\dot{q}_1^2C(2q_2))/2 + (l_{c3}^2\dot{q}_1^2C(2q_2 + 2q_3))/2 + l_1l_{c3}\dot{q}_1^2C(2q_2 + q_3) + l_1l_{c3}\dot{q}_1^2C_3 + 2l_1l_{c3}\dot{q}_2^2C_3 + 2l_1l_{c3}\dot{q}_2\dot{q}_3C_3))/2 + (I_3(\dot{q}_1^2 + \dot{q}_2^2 + 2\dot{q}_2\dot{q}_3 + \dot{q}_3^2))/2$$

Considerando la Figura 3.18 y la ecuación 3.19, obtenemos la energía potencial del elemento en estudio:

$$U_3 = gm_3(l_b + l_{c3}S(q_2 + q_3) + l_1S_2)$$

○ **Eslabón 4**

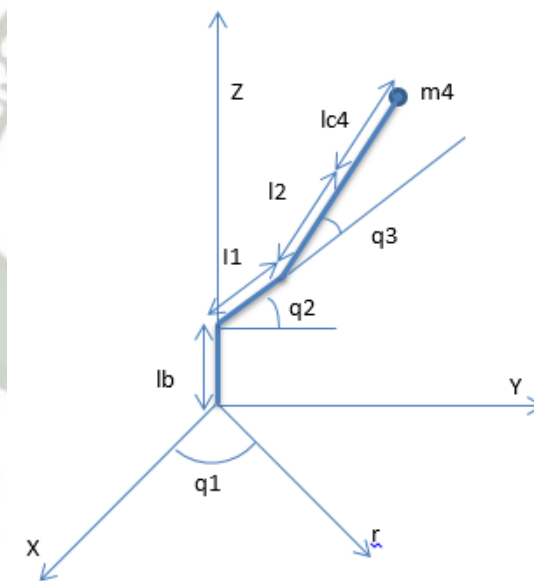


Figura 3.19: Representación del 4to eslabón para análisis dinámico

Fuente: Propia

Considerando la Figura 3.19, encontramos geométricamente la posición del centro de masa del eslabón 4:

$$\begin{aligned} r &= l_1C_2 + (l_2 + l_{c4})C(q_3 + q_2) \\ x &= rC_1 \\ y &= rS_1 \\ z &= l_b + l_1S_2 + (l_2 + l_{c4})S(q_3 + q_2) \end{aligned}$$

Derivamos cada una de las ecuaciones obtenidas en función del tiempo para hallar luego la velocidad lineal:

$$v_x = -l_2 S_{23} C_1 (\dot{q}_2 + \dot{q}_3) - l_{c4} S_{23} q_3 C_1 (\dot{q}_2 + \dot{q}_3) - l_2 \dot{q}_1 C_{23} S_1 - l_{c4} \dot{q}_1 C_{23} S_1 - l_1 \dot{q}_1 C_2 S_1 - l_1 \dot{q}_2 C_1 S_2$$

$$v_y = l_2 \dot{q}_1 C_{23} C_1 - l_2 S_{23} S_1 (\dot{q}_2 + \dot{q}_3) - l_{c4} S_{23} S_1 (\dot{q}_2 + \dot{q}_3) - l_1 \dot{q}_2 S_1 S_2 + l_{c4} \dot{q}_1 C_{23} C_1 + l_1 \dot{q}_1 C_1 C_2$$

$$v_z = l_1 \dot{q}_2 C_2 + C_{23} (l_2 + l_{c4}) (\dot{q}_2 + \dot{q}_3)$$

$$v_4 = \sqrt{v_x^2 + v_y^2 + v_z^2}$$

$$v_4 = (l_1^2 \dot{q}_1^2)/2 + l_1^2 \dot{q}_2^2 + (l_{c3}^2 \dot{q}_1^2)/2 + l_{c3}^2 \dot{q}_2^2 + l_{c3}^2 \dot{q}_3^2 + 2l_{c3}^2 \dot{q}_2 \dot{q}_3 + (l_1^2 \dot{q}_1^2 C(2q_2))/2 + (l_{c3}^2 \dot{q}_1^2 C(2q_2 + 2q_3))/2 + l_1 l_{c3} \dot{q}_1^2 C(2q_2 + q_3) + l_1 l_{c3} \dot{q}_1^2 C_3 + 2l_1 l_{c3} \dot{q}_2^2 C_3 + 2l_1 l_{c3} \dot{q}_2 \dot{q}_3 C_3$$

Posteriormente, calculamos la matriz Jacobiana Geométrica del elemento en estudio:

$$J_4 = \begin{bmatrix} J_{41} & J_{43} & J_{48} & 0 \\ J_{42} & J_{44} & J_{49} & 0 \\ 0 & J_{45} & J_{50} & 0 \\ 0 & J_{46} & J_{51} & J_{53} \\ 0 & J_{47} & J_{52} & J_{54} \\ 1 & 0 & 0 & J_{55} \end{bmatrix}$$

$$J_{41} = -S_1 (l_2 C_{23} + l_3 C_{23} + l_1 C_2)$$

$$J_{42} = C_1 (l_2 C_{23} + l_3 C_{23} + l_1 C_2)$$

$$J_{43} = -C_1 (l_2 S_{23} + l_3 S_{23} + l_1 S_2)$$

$$J_{44} = -S_1 (l_2 S_{23} + l_3 S_{23} + l_1 S_2)$$

$$J_{45} = l_2 C_{23} + l_3 C_{23} + l_1 C_2$$

$$J_{46} = S_1$$

$$J_{47} = -C_1$$

$$J_{48} = -S_{23} C_1 (l_2 + l_3)$$

$$J_{49} = -S_{23} S_1 (l_2 + l_3)$$

$$J_{50} = C_{23} (l_2 + l_3)$$

$$J_{51} = S_1$$

$$J_{52} = -C_1$$

$$J_{53} = C_{23} C_1$$

$$J_{54} = C_{23} S_1$$

$$J_{55} = S_{23}$$

Multiplicamos la matriz Jacobiana por los valores de las velocidades articulares, para obtener así la velocidad angular del eslabón 4:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_4 * \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} v_{4x} \\ v_{4y} \\ v_{4z} \\ w_{4x} \\ w_{4y} \\ w_{4z} \end{bmatrix}$$

$$w_{4x} = \dot{q}_2 S_1 + \dot{q}_3 S_1 + \dot{q}_4 C_{23} C_1$$

$$w_{4y} = \dot{q}_4 C_{23} S_1 - \dot{q}_3 C_1 - \dot{q}_2 C_1$$

$$w_{4z} = \dot{q}_1 + \dot{q}_4 S_{23}$$

$$w_4 = \sqrt{w_x^2 + w_y^2 + w_z^2}$$

$$w_4 = (\dot{q}_2 C_1 + \dot{q}_3 C_1 - \dot{q}_4 C_{23} S_1)^2 + (\dot{q}_2 S_1 + \dot{q}_3 S_1 + \dot{q}_4 C_{23} C_1)^2 + (\dot{q}_1 + \dot{q}_4 S_{23})^2$$

Reemplazamos las velocidades lineal y angular del eslabón en la ecuación 3.18, para hallar la energía cinética:

$$\begin{aligned} K_4 = & (I_4 \dot{q}_1^2)/2 + (I_4 \dot{q}_2^2)/2 + (I_4 \dot{q}_3^2)/2 + (I_4 \dot{q}_4^2)/2 + (l_1^2 m_4 \dot{q}_1^2)/4 + \\ & (l_1^2 m_4 \dot{q}_2^2)/2 + (l_{c3}^2 m_4 \dot{q}_1^2)/4 + (l_{c3}^2 m_4 \dot{q}_2^2)/2 + (l_{c3}^2 m_4 \dot{q}_3^2)/2 + \\ & I_4 \dot{q}_2 \dot{q}_3 + I_4 \dot{q}_1 \dot{q}_4 S_{23} + l_{c3}^2 m_4 \dot{q}_2 \dot{q}_3 + (l_1^2 m_4 \dot{q}_1^2 C(2q_2))/4 + \\ & (l_{c3}^2 m_4 \dot{q}_1^2 C(2q_2 + 2q_3))/4 + (l_1 l_{c3} m_4 \dot{q}_1^2 C_3)/2 + l_1 l_{c3} m_4 \dot{q}_2^2 C_3 + \\ & (l_1 l_{c3} m_4 \dot{q}_1^2 C(2q_2 + q_3))/2 + l_1 l_{c3} m_4 \dot{q}_2 \dot{q}_3 C_3 \end{aligned}$$

Tomando en cuenta la Fig. 3.19 y la ecuación 3.19, obtenemos la energía potencial:

$$U_4 = g m_4 (l_b + S_{23} (l_2 + l_{c4}) + l_1 S_2)$$

○ Energía Cinética

Seguidamente, obtenemos el valor de la Energía Cinética Total del manipulador robótico, como la suma de las energías cinéticas de los eslabones:

$$K = K_1 + K_2 + K_3 + K_4$$

$$\begin{aligned} K = & (I_1 \dot{q}_1^2)/2 + (I_2 \dot{q}_1^2)/2 + (I_2 \dot{q}_2^2)/2 + (I_3 \dot{q}_1^2)/2 + (I_3 \dot{q}_2^2)/2 + (I_4 \dot{q}_1^2)/ \\ & 2 + (I_3 \dot{q}_3^2)/2 + (I_4 \dot{q}_2^2)/2 + (I_4 \dot{q}_3^2)/2 + (I_4 \dot{q}_4^2)/2 + (l_1^2 m_3 \dot{q}_1^2)/4 + \end{aligned}$$

$$\begin{aligned}
 & (l_1^2 m_3 \dot{q}_2^2)/2 + (l_1^2 m_4 \dot{q}_1^2)/4 + (l_1^2 m_4 \dot{q}_2^2)/2 + (l_{c2}^2 m_2 \dot{q}_1^2)/4 + \\
 & (l_{c2}^2 m_2 \dot{q}_2^2)/2 + (l_{c3}^2 m_3 \dot{q}_1^2)/4 + (l_{c3}^2 m_3 \dot{q}_2^2)/2 + (l_{c3}^2 m_4 \dot{q}_1^2)/4 + \\
 & (l_{c3}^2 m_3 \dot{q}_3^2)/2 + (l_{c3}^2 m_4 \dot{q}_2^2)/2 + (l_{c3}^2 m_4 \dot{q}_3^2)/2 + I_3 \dot{q}_2 \dot{q}_3 + I_4 \dot{q}_2 \dot{q}_3 + \\
 & I_4 \dot{q}_1 \dot{q}_4 S_{23} + l_{c3}^2 m_3 \dot{q}_2 \dot{q}_3 + l_{c3}^2 m_4 \dot{q}_2 \dot{q}_3 + (l_1^2 m_3 \dot{q}_1^2 C(2q_2))/4 + \\
 & (l_1^2 m_4 \dot{q}_1^2 C(2q_2))/4 + (l_{c2}^2 m_2 \dot{q}_1^2 C(2q_2))/4 + (l_{c3}^2 m_3 \dot{q}_1^2 C(2q_2 + 2q_3))/4 + \\
 & (l_{c3}^2 m_4 \dot{q}_1^2 C(2q_2 + 2q_3))/4 + (l_1 l_{c3} m_3 \dot{q}_1^2 C_3)/2 + l_1 l_{c3} m_3 \dot{q}_2^2 C_3 + \\
 & (l_1 l_{c3} m_4 \dot{q}_1^2 C_3)/2 + l_1 l_{c3} m_4 \dot{q}_2^2 C_3 + (l_1 l_{c3} m_3 \dot{q}_1^2 C(2q_2 + q_3))/2 + \\
 & (l_1 l_{c3} m_4 \dot{q}_1^2 C(2q_2 + q_3))/2 + l_1 l_{c3} m_3 \dot{q}_2 \dot{q}_3 C_3 + l_1 l_{c3} m_4 \dot{q}_2 \dot{q}_3 C_3
 \end{aligned}$$

○ Energía Potencial

Hallamos el valor de la Energía Potencial Total del robot, como la suma de las energías potenciales de los eslabones:

$$U = U_1 + U_2 + U_3 + U_4$$

$$\begin{aligned}
 U = & gm_3(l_b + l_{c3}S_{23} + l_1S_2) + gl_{c1}m_1 + gm_2(l_b + l_{c2}S_2) + gm_4(l_b \\
 & + S_{23} * (l_2 + l_{c4}) + l_1S_2)
 \end{aligned}$$

○ Lagrangiano

Reemplazamos la energía cinética y potencial del manipulador en la ecuación 3.16, para obtener el Lagrangiano del modelo dinámico:

$$\begin{aligned}
 L = & (I_1 \dot{q}_1^2)/2 + (I_2 \dot{q}_1^2)/2 + (I_2 \dot{q}_2^2)/2 + (I_3 \dot{q}_1^2)/2 + (I_3 \dot{q}_2^2)/2 + (I_4 \dot{q}_1^2)/2 + \\
 & (I_3 \dot{q}_3^2)/2 + (I_4 \dot{q}_2^2)/2 + (I_4 \dot{q}_3^2)/2 + (I_4 \dot{q}_4^2)/2 - gm_3(l_b + l_{c3}S_{23} + \\
 & l_1S_2) + (l_1^2 m_3 \dot{q}_1^2)/4 + (l_1^2 m_3 \dot{q}_2^2)/2 + (l_1^2 m_4 \dot{q}_1^2)/4 + (l_1^2 m_4 \dot{q}_2^2)/2 + \\
 & (l_{c2}^2 m_2 \dot{q}_1^2)/4 + (l_{c2}^2 m_2 \dot{q}_2^2)/2 + (l_{c3}^2 m_3 \dot{q}_1^2)/4 + (l_{c3}^2 m_3 \dot{q}_2^2)/2 + \\
 & (l_{c3}^2 m_4 \dot{q}_1^2)/4 + (l_{c3}^2 m_3 \dot{q}_3^2)/2 + (l_{c3}^2 m_4 \dot{q}_2^2)/2 + (l_{c3}^2 m_4 \dot{q}_3^2)/2 + \\
 & I_3 \dot{q}_2 \dot{q}_3 + I_4 \dot{q}_2 \dot{q}_3 - gl_{c1}m_1 - gm_2(l_b + l_{c2}S_2) - gm_4(l_b + S_{23} * (l_2 + \\
 & l_{c4}) + l_1S_2) + I_4 \dot{q}_1 \dot{q}_4 S_{23} + l_{c3}^2 m_3 \dot{q}_2 \dot{q}_3 + l_{c3}^2 m_4 \dot{q}_2 \dot{q}_3 + \\
 & (l_1^2 m_3 \dot{q}_1^2 C(2q_2))/4 + (l_1^2 m_4 \dot{q}_1^2 C(2q_2))/4 + (l_{c2}^2 m_2 \dot{q}_1^2 C(2q_2))/4 + \\
 & (l_{c3}^2 m_3 \dot{q}_1^2 C(2q_2 + 2q_3))/4 + (l_{c3}^2 m_4 \dot{q}_1^2 C(2q_2 + 2q_3))/4 + \\
 & (l_1 l_{c3} m_3 \dot{q}_1^2 C_3)/2 + l_1 l_{c3} m_3 \dot{q}_2^2 C_3 + (l_1 l_{c3} m_4 \dot{q}_1^2 C_3)/2 + l_1 l_{c3} m_4 \dot{q}_2^2 C_3 + \\
 & (l_1 l_{c3} m_3 \dot{q}_1^2 C(2q_2 + q_3))/2 + (l_1 l_{c3} m_4 \dot{q}_1^2 C(2q_2 + q_3))/2 + \\
 & l_1 l_{c3} m_3 \dot{q}_2 \dot{q}_3 C_3 + l_1 l_{c3} m_4 \dot{q}_2 \dot{q}_3 C_3
 \end{aligned}$$

○ Torques

Considerando la ecuación 3.17, hallamos las ecuaciones que definen los torques que se deben aplicar a cada articulación del robot para producir su movimiento:

$$T_1 = \ddot{q}_1(I_1 + I_2 + I_3 + I_4 + (l_1^2 m_3)/2 + (l_1^2 m_4)/2 + (l_{c1}^2 m_2)/2 + (l_{c3}^2 m_3)/2 + (l_{c3}^2 m_4)/2) + I_4 \ddot{q}_4 S(q_2 + q_3) + I_4 \dot{q}_4 C(q_2 + q_3)(\dot{q}_2 + \dot{q}_3) + (l_1^2 \ddot{q}_1 C(2q_2)(m_3 + m_4))/2 + (l_{c3}^2 \ddot{q}_1 (2q_2 + 2q_3)(m_3 + m_4))/2 - (l_{c3}^2 \dot{q}_1 S(2q_2 + 2q_3)(2\dot{q}_2 + 2\dot{q}_3)(m_3 + m_4))/2 - l_1^2 \dot{q}_1 \dot{q}_2 S(2q_2)(m_3 + m_4) + l_1 l_{c3} \ddot{q}_1 C_3(m_3 + m_4) + l_1 l_{c3} \dot{q}_1 C(2q_2 + q_3)(m_3 + m_4) - l_1 l_{c3} \dot{q}_1 \dot{q}_3 S_3(m_3 + m_4) - l_1 l_{c3} \dot{q}_1 S(2q_2 + q_3)(m_3 + m_4)(2\dot{q}_2 + \dot{q}_3)$$

$$T_2 = \ddot{q}_2(I_2 + I_3 + I_4 + l_1^2 m_4 + l_2^2 m_3 + l_{c1}^2 m_2 + l_{c3}^2 m_3 + l_{c3}^2 m_4) + \ddot{q}_3(I_3 + I_4 + l_{c3}^2 m_3 + l_{c3}^2 m_4) + g m_4 (C(q_2 + q_3)(l_2 + l_{c4}) + l_1 C(2q_2)) + g m_3 (l_{c3} C(q_2 + q_3) + l_1 C(q_2)) - I_4 \dot{q}_1 \dot{q}_4 C(q_2 + q_3) + g l_{c2} m_2 C_2 + (l_1^2 m_3 \dot{q}_1^2 S(2q_2))/2 + (l_1^2 m_4 \dot{q}_1^2 S(2q_2))/2 + (l_{c1}^2 m_2 \dot{q}_1^2 S(2q_2))/2 + (l_{c3}^2 m_3 \dot{q}_1^2 S(2q_2 + 2q_3))/2 + (l_{c3}^2 m_4 \dot{q}_1^2 S(2q_2 + 2q_3))/2 - l_1 l_{c3} \dot{q}_3^2 S_3(m_3 + m_4) + l_1 l_{c3} m_3 \dot{q}_1^2 S(2q_2 + q_3) + l_1 l_{c3} m_4 \dot{q}_1^2 S(2q_2 + q_3) + 2 l_1 l_{c3} \dot{q}_2 C_3(m_3 + m_4) + l_1 l_{c3} \dot{q}_3 C_3(m_3 + m_4) - 2 l_1 l_{c3} \dot{q}_2 \dot{q}_3 S_3(m_3 + m_4)$$

$$T_3 = I_3 \ddot{q}_2 + I_3 \ddot{q}_3 + I_4 \ddot{q}_2 + I_4 \ddot{q}_3 + l_{c3}^2 m_3 \ddot{q}_2 + l_{c3}^2 m_3 \ddot{q}_3 + l_{c3}^2 m_4 \ddot{q}_2 + l_{c3}^2 m_4 \ddot{q}_3 - I_4 \dot{q}_1 \dot{q}_4 C(q_2 + q_3) + g l_2 m_4 C(q_2 + q_3) + g l_{c3} m_3 C(q_2 + q_3) + g l_{c4} m_4 C(q_2 + q_3) + (l_{c3}^2 m_3 \dot{q}_1^2 S(2q_2 + 2q_3))/2 + (l_{c3}^2 m_4 \dot{q}_1^2 S(2q_2 + 2q_3))/2 + (l_1 l_{c3} m_3 \dot{q}_1^2 S_3)/2 + l_1 l_{c3} m_3 \dot{q}_2^2 S_3 + (l_1 l_{c3} m_4 \dot{q}_1^2 S_3)/2 + l_1 l_{c3} m_4 \dot{q}_2^2 S_3 + (l_1 l_{c3} m_3 \dot{q}_1^2 S(2q_2 + q_3))/2 + (l_1 l_{c3} m_4 \dot{q}_1^2 S(2q_2 + q_3))/2 + l_1 l_{c3} m_3 \dot{q}_2 C_3 + l_1 l_{c3} m_4 \dot{q}_2 C_3$$

$$T_4 = I_4 \ddot{q}_4 + I_4 \dot{q}_1 S(q_2 + q_3) + I_4 \dot{q}_1 C(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)$$

○ **Modelo dinámico en términos de matrices**

$$T = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q)$$

Donde:

M: Matriz de Inercia, es simétrica y positiva definida.

C: Matriz de Fuerza de Coriolis.

G: Vector de Gravedad.

● **Matriz de Inercia:**

$$M = \begin{bmatrix} M_1 & M_5 & M_9 & M_{13} \\ M_2 & M_6 & M_{10} & M_{14} \\ M_3 & M_7 & M_{11} & M_{15} \\ M_4 & M_8 & M_{12} & M_{16} \end{bmatrix}$$

$$M_1 = I_1 + I_2 + I_3 + I_4 + (l_1^2 m_3)/2 + (l_1^2 m_4)/2 + (l_{c1}^2 m_2)/2 \\ + (l_{c3}^2 m_3)/2 + (l_{c3}^2 m_4)/2 + (l_1^2 C(2q_2)(m_3 + m_4))/2 \\ + (l_{c3}^2 (2q_2 + 2q_3)(m_3 + m_4))/2 + l_1 l_{c3} C_3(m_3 + m_4) \\ + l_1 l_{c3} C(2q_2 + q_3)(m_3 + m_4)$$

$$M_2 = 0$$

$$M_3 = 0$$

$$M_4 = I_4 S(q_2 + q_3)$$

$$M_5 = 0$$

$$M_6 = I_2 + I_3 + I_4 + l_1^2 m_4 + l_2^2 m_3 + l_{c1}^2 m_2 + l_{c3}^2 m_3 + l_{c3}^2 m_4 \\ + 2l_1 l_{c3} C_3(m_3 + m_4)$$

$$M_7 = I_3 + I_4 + l_{c3}^2 m_3 + l_{c3}^2 m_4 + l_1 l_{c3} C_3(m_3 + m_4)$$

$$M_8 = 0$$

$$M_9 = 0$$

$$M_{10} = I_3 + I_4 + l_{c3}^2 m_3 + l_{c3}^2 m_4 + l_1 l_{c3} m_3 C_3 + l_1 l_{c3} m_4 C_3$$

$$M_{11} = I_3 + I_4 + l_{c3}^2 m_3 + l_{c3}^2 m_4$$

$$M_{12} = 0$$

$$M_{13} = l_1 l_{c3} C_3(m_3 + m_4)$$

$$M_{14} = 0$$

$$M_{15} = 0$$

$$M_{16} = I_4$$

• **Matriz de fuerza de Coriolis:**

$$C = \begin{bmatrix} C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \\ C_4 & C_8 & C_{12} & C_{16} \end{bmatrix}$$

$$C_1 = -l_1^2 \dot{q}_2 S(2q_2)(m_3 + m_4) - (l_{c3}^2 S(2q_2 + 2q_3)(2\dot{q}_2 + 2\dot{q}_3)(m_3 \\ + m_4)) - l_1 l_{c3} S(2q_2 + q_3)(m_3 + m_4)(2\dot{q}_2 + \dot{q}_3) \\ - l_1 l_{c3} \dot{q}_3 S_3(m_3 + m_4)$$

$$C_2 = (l_1^2 m_3 \dot{q}_1 S(2q_2))/2 - I_4 \dot{q}_4 C(q_2 + q_3) + (l_1^2 m_4 \dot{q}_1 S(2q_2))/2 \\ + (l_{c1}^2 m_2 \dot{q}_1 S(2q_2))/2 + (l_{c3}^2 m_3 \dot{q}_1 S(2q_2 + 2q_3))/2 \\ + (l_{c3}^2 m_4 \dot{q}_1 S(2q_2 + 2q_3))/2 + l_1 l_{c3} m_3 \dot{q}_1 S(2q_2 + q_3) \\ + l_1 l_{c3} m_4 \dot{q}_1 S(2q_2 + q_3)$$

$$C_3 = (l_{c3}^2 m_3 \dot{q}_1 S(2q_2 + 2q_3))/2 - I_4 \dot{q}_4 C(q_2 + q_3) + (l_{c3}^2 m_4 \dot{q}_1 S(2q_2 + 2q_3))/2 + (l_1 l_{c3} m_3 \dot{q}_1 S(2q_2 + q_3))/2 + (l_1 l_{c3} m_4 \dot{q}_1 S(2q_2 + q_3))/2 + (l_1 l_{c3} m_3 \dot{q}_1 S_3)/2 + (l_1 l_{c3} m_4 \dot{q}_1 S_3)/2$$

$$C_4 = I_4 C(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)$$

$$C_5 = -l_1^2 \dot{q}_1 S(2q_2)(m_3 + m_4) - 2l_1 l_{c3} \dot{q}_1 S(2q_2 + q_3)(m_3 + m_4) - l_{c3}^2 \dot{q}_1 S(2q_2 + 2q_3)(m_3 + m_4 + I_4 \dot{q}_4 C(q_2 + q_3))$$

$$C_6 = -2l_1 l_{c3} \dot{q}_3 S_3(m_3 + m_4)$$

$$C_7 = l_1 l_{c3} m_3 \dot{q}_2 S_3 + l_1 l_{c3} m_4 \dot{q}_2 S_3$$

$$C_8 = I_4 \dot{q}_1 C(q_2 + q_3)$$

$$C_9 = I_4 \dot{q}_4 C(q_2 + q_3) - l_{c3}^2 \dot{q}_1 S(2q_2 + 2q_3)(m_3 + m_4) - l_1 l_{c3} \dot{q}_1 S_3(m_3 + m_4) - l_1 l_{c3} \dot{q}_1 S(2q_2 + q_3)(m_3 + m_4)$$

$$C_{10} = -2l_1 l_{c3} \dot{q}_2 S_3(m_3 + m_4) - l_1 l_{c3} \dot{q}_3 S_3(m_3 + m_4)$$

$$C_{11} = 0$$

$$C_{12} = I_4 \dot{q}_1 C(q_2 + q_3)$$

$$C_{13} = I_4 C(q_2 + q_3)(\dot{q}_2 + \dot{q}_3)$$

$$C_{14} = -I_4 \dot{q}_1 C(q_2 + q_3)$$

$$C_{15} = -I_4 \dot{q}_1 C(q_2 + q_3)$$

$$C_{16} = 0$$

• **Vector de Gravedad:**

$$G = \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$$G_1 = 0$$

$$G_2 = g m_4 (C(q_2 + q_3)(l_2 + l_{c4}) + l_1 C(2q_2)) + g m_3 (l_{c3} C(q_2 + q_3) + l_1 C(q_2)) + g l_{c2} m_2 C_2$$

$$G_3 = g l_2 m_4 C(q_2 + q_3) + g l_{c3} m_3 C(q_2 + q_3) + g l_{c4} m_4 C(q_2 + q_3)$$

$$G_4 = 0$$

3.1.5. Materiales:

Un compromiso entre “ligereza” y “resistencia” son algunas de las características que en la práctica se pretenden conseguir a la hora de seleccionar el material de fabricación de un sistema robótico. Además, es vital considerar que el robot cumpla con algunos objetivos con respecto a las etapas de diseño y fabricación, tales como gran rendimiento, que su estructura no afecte al movimiento, que tenga mínimos juegos u holguras y que sea capaz de soportar un funcionamiento continuo y elevado.

La diversidad en cuanto a los materiales que se utilizan durante la manufactura de un manipulador es muy amplia, va desde madera hasta metales estructurales como aluminio y acero, pasando por plásticos de ingeniería como ABS, DERLIN, ARNITE, Polietileno de alta densidad o PVC.

La selección del material a utilizarse durante la construcción del robot significa un proceso trascendente y fundamental, debido a que éste afecta a la inercia del sistema, a su respuesta dinámica, precisión y uso industrial. Es así, que durante el desarrollo de la presente tesis se vio por conveniente elegir al ABS como material para la fabricación del brazo robótico a diseñar, debido a que el acrílonitrilo butadieno estireno es un plástico duro y rígido con grandes características entre las que destaca su resistencia al impacto y su gran tenacidad, además de que es un material fácil de mecanizar, muy utilizado en impresiones 3D y permite buenos acabados. Así mismo, para la aplicación específica en laboratorios farmacéuticos, se decidió hacer uso de este polímero en color blanco para así satisfacer el requerimiento de higiene planteado. En cuanto al requisito referido a la resistencia se muestra la Tabla 3.5 con algunas propiedades extraídas del datasheet del material.

Tabla 3.5: Propiedades Mecánicas determinantes del ABS

Ensayo de Tracción	Valor Nominal
Límite elástico	54.0 MPa
Módulo elástico	2.20 Gpa

Fuente: Propia

3.1.6. Elección de Actuadores y Diseño Virtual en Solidwork:

El proceso de selección de actuadores es un paso fundamental durante el diseño mecánico de un robot, que tiene gran incidencia en el adecuado funcionamiento del mismo, por ello es por lo que se debe prestar especial atención en su desarrollo. Existen diferentes factores que es necesario considerar a la hora de una acertada elección de actuadores, tales como potencia, peso, volumen, precisión, velocidad, controlabilidad y torque; siendo el último, la característica de mayor interés durante este procedimiento; puesto que este parámetro determina la fuerza de giro que se requiere para el movimiento del robot en cada articulación.

Como se vio en el capítulo anterior, el objetivo fundamental de todo accionamiento es generar el movimiento de los diferentes mecanismos que conforman el robot para el desplazamiento de una carga a una posición específica con una determinada velocidad. En el campo de la robótica, existen diferentes tipos de actuadores clasificados según la fuente de energía que emplean para su funcionamiento, siendo los eléctricos los más utilizados para aplicaciones que requieran de exactitud y repetitividad como en este caso.

En el mercado actual, se tiene una amplia gama de motores eléctricos aptos para distintas aplicaciones, en donde se destacan los motores de corriente continua, paso a paso y servomotores como los de mayor incidencia en uso para robótica. Tras un análisis exhaustivo acerca de las características de cada tipo de motor anteriormente mencionado, se optó por elegir servomotores, por su capacidad de poder ser llevados a posiciones angulares específicas con una precisión de hasta centésimas de milímetros; así mismo, este tipo de elemento motriz se caracteriza por tener altos torques, bajos costos de operación y un control alto en comparación con los motores DC o paso a paso.

Por consiguiente, al realizarse una revisión minuciosa por diferentes catálogos de fabricantes en la web, se identificó que la marca Robotics Dynamixel Servos ofrece una serie de actuadores con muy altas prestaciones para robots totalmente programables. Dentro de sus características más destacables, podemos mencionar:

- Permite el uso de la topología daysi-chain⁶ para la conexión entre varios motores, lo que facilita el cableado.
- Son controlados por protocolos de comunicación digital, a través de paquetes de datos enviados por la red serial. Soportan comunicación TTL y el estándar RS-485, dependiendo del modelo.
- Alta eficiencia debido a la baja corriente de consumo
- Proporciona información adicional de retroalimentación en comparación con los servomotores convencionales, puesto que se puede leer además de la posición

⁶Método de conexión de periféricos agrupados en diversas ramas seriales correspondientes a un canal, donde cada uno de ellos, requiere de una sola interfaz con la unidad central.

angular actual del motor, la velocidad, la temperatura interna, el torque o la tensión de alimentación.

- Permiten leer en tiempo real la información captada por sus sensores embebidos.
- Posee una función de alarma, la misma que enciende un led o desconecta el actuador en caso se superen los márgenes especificados en su datasheet, ya sea de temperatura, torque, tensión de alimentación, etc.
- Algunos modelos, permiten configurar control PID para minimizar oscilaciones.

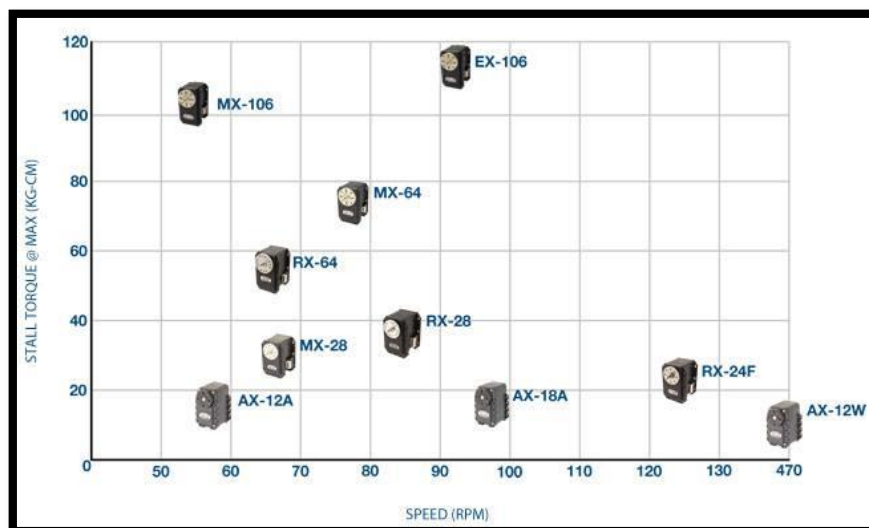


Figura 3.20: Torque(kg-cm) vs Velocidad(RPM) de Servomotores Dynamixel

Fuente: Propia

Considerando la información base antes descrita, procederemos a realizar un proceso iterativo partiendo de los bosquejos preliminares y de las relaciones halladas durante el análisis cinemático y dinámico del robot, para definir finalmente los actuadores a utilizar en cada una de las articulaciones del manipulador, así como para determinar la estructura física final del brazo robótico.

Iniciamos el procedimiento por el último eslabón de la cadena cinemática correspondiente a la garra, la misma que en su forma básica es sometida a dos tipos de ensayos mecánicos, con el objetivo de detallar su aspecto físico de acuerdo a una vida útil aceptable y resistencia del material de construcción apropiada para su funcionamiento. El primer ensayo llevado a cabo fue el ensayo estático, para el cual se graficó el bosquejo del efector final, haciendo uso del software de diseño mecánico en 3D, Solidwork, como se muestra en la Figura 3.21.

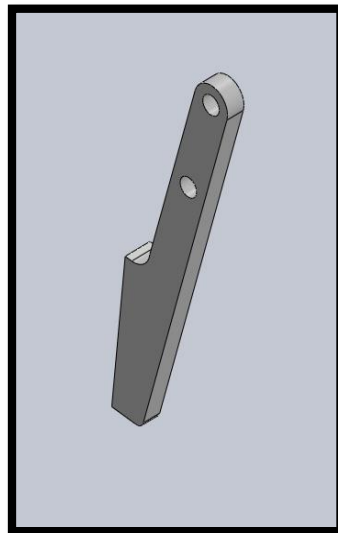


Figura 3.21: Bosquejo preliminar de la Garra

Fuente: Propia

Seguidamente, se creó el material de construcción elegido (ABS), tomando en cuenta esencialmente dos características definidas en su datasheet tales como límite de elasticidad, el mismo que especifica la tensión máxima que puede soportar un material sin sufrir deformaciones; y el límite de rotura, definido como la carga máxima que resiste el material antes de producirse su fractura.

Luego se realizó el ensayo estático propiamente dicho a través de un análisis de elementos finitos, para lo cual se agregó dos fuerzas de 10N en direcciones distintas; una simulando el peso de la carga máxima a soportar por el brazo robótico y la otra simulando la fuerza de reacción del instrumento a manipular; de igual manera, se tomó en cuenta la gravedad para que el estudio sea lo más real posible. (Fig. 3.22)



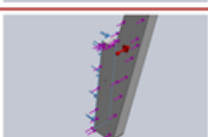
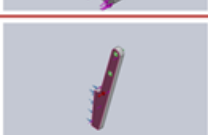
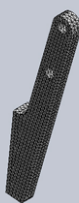
Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		Entidades: 1 cara(s) Tipo: Aplicar fuerza normal Valor: 10 N
Fuerza-2		Entidades: 1 cara(s) Tipo: Aplicar fuerza normal Valor: 10 N
Gravedad-1		Referencia: Carac 1 > Valores: 0 0 9.81 Unidades: SI

Figura 3.22: Fuerzas aplicadas a la garra

Fuente: Propia

A fin de asegurar la efectividad del ensayo, se debe además utilizar una malla con un tamaño mínimo de cada elemento de la mitad del espesor de la pieza sólida; por lo tanto, en nuestro caso teniendo 5 mm de espesor tendríamos un tamaño mínimo de malla de 2.5 mm, sin embargo, se utilizó una dimensión de 1.5 mm para hacer más exactos nuestros cálculos como se puede apreciar en la Figura 3.23.



Tipo de malla	Malla sólida
Mallador utilizado:	Malla estándar
Transición automática:	Desactivar
Incluir bucles automáticos de malla:	Desactivar
Puntos jacobianos	4 Puntos
Tamaño de elementos	1.5885 mm
Tolerancia	0.0794251 mm
Calidad de malla	Elementos cuadráticos de alto orden

Figura 3.23: Cálculos para malla

Fuente: Propia

Los resultados obtenidos, después de haber realizado el ensayo, son los siguientes:

- **Tensiones:** Nos muestra un gráfico, donde es posible analizar las tensiones que puede soportar el elemento como máximo, en función del límite elástico del material. Como se puede ver en la Fig. 3.24, la parte más crítica de la pieza es el segundo agujero, el mismo que posee una porción pigmentada en rojo, correspondiente a una tensión de 13MPa.

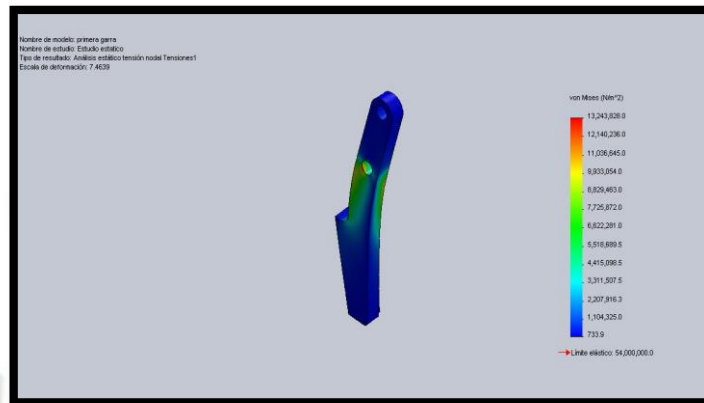


Figura 3.24: Gráfico de Tensiones
Fuente: Propia

- **Desplazamientos:** Nos muestra en milímetros, cual es el desplazamiento de la pieza respecto a las fuerzas soportadas, teniendo como mayor valor 1.49 mm.

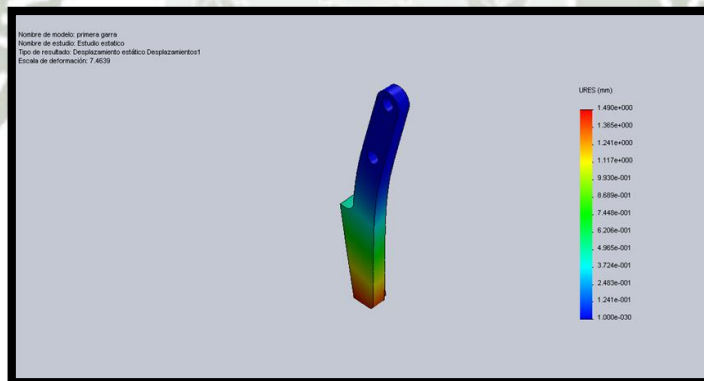


Figura 3.25: Gráfica de desplazamientos
Fuente: Propia

- **Factor de Seguridad:** Está determinado por el Solidwork y sirve para entender en base a los resultados anteriores cuantas veces puede soportar nuestra pieza, la carga aplicada. De esta manera, al observar la Fig. 3.26, podemos concluir que el componente es capaz de resistir 4.1 veces más carga que a la que fue sometida.

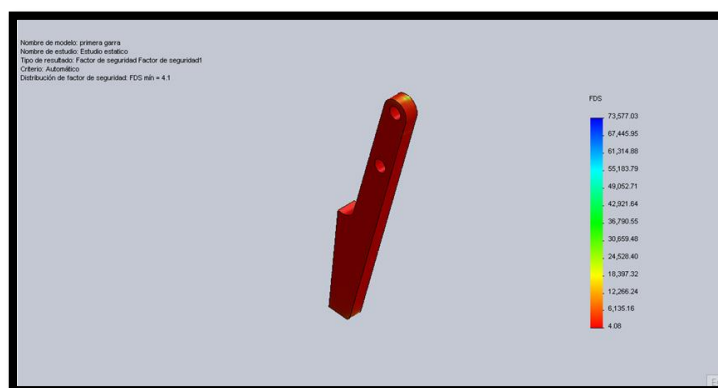


Figura 3.26: Gráfica de Factor de Seguridad

Fuente: Propia

Sin embargo, aunque los resultados alcanzados durante este estudio son positivos, confirmando la resistencia del elemento, no podemos confiar totalmente puesto que este componente no trabaja de forma estática; sino tendrá un movimiento repetido, soportará esfuerzos y manipulará carga, por lo tanto, un simple ensayo estático no es suficiente, se debe aplicar un segundo análisis, ensayo de fatiga.

El ensayo de fatiga nos permite definir el rango de repeticiones en el eslabón, para lo cual es esencial ingresar algunos datos al programa como por ejemplo definir cuantas repeticiones tiene que resistir este elemento, lo que implica la creación de un suceso considerando que 1000000 de repeticiones simula una vida infinita (Fig. 3.27).

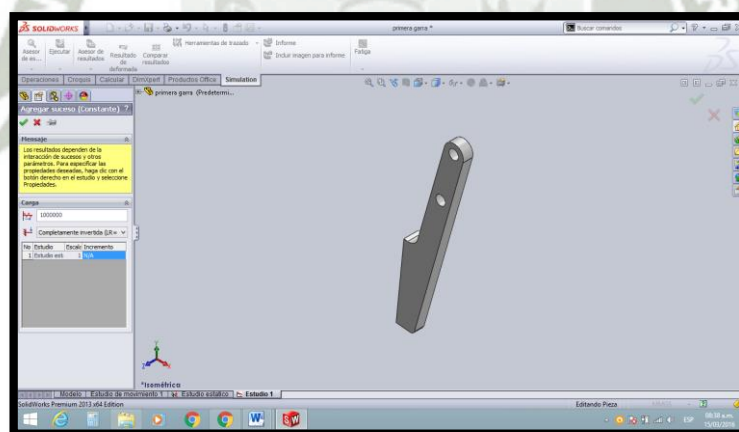


Figura 3.27: Definición de suceso

Fuente: Propia

El material establecido en el ensayo anterior, estima ahora un nuevo parámetro necesario llamado curva S – N, que define los valores de tensiones alternas sobre el número de ciclos requeridos por casos de ensayos de fatiga y es extraída mediante la realización de ensayos reales sobre componentes de dicho material. Esta característica debe ser agregada en el programa tal como se muestra en la Figura 3.28.

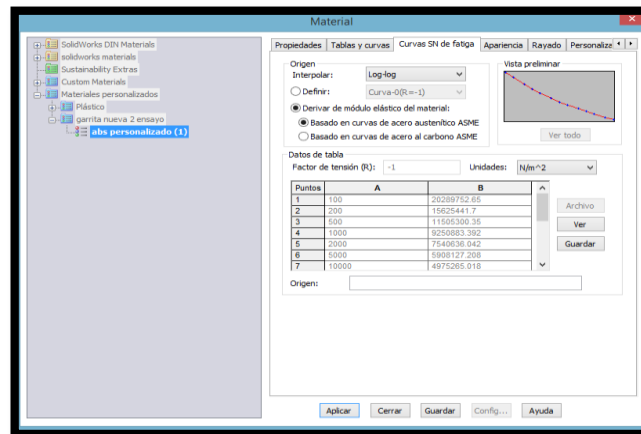


Figura 3.28: Definición de material

Fuente: Propia

Tras realizado el ensayo, adquirimos el siguiente resultado:

Vida total: Indica el número de repeticiones máximas que puede soportar la pieza en distintos puntos a lo largo de ella. En este caso podemos observar que el segmento de color rojo tiene tan solo 231 repeticiones antes de fallar lo cual no es óptimo.

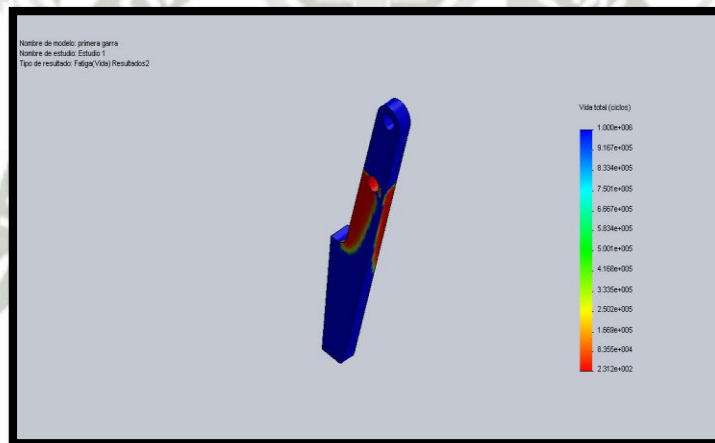


Figura 3.29: Gráfica de Vida Útil

Fuente: Propia

En consecuencia, de los ensayos efectuados, podemos concluir que el segundo agujero, es la parte de la pieza mayormente perjudicada; motivo por el cual es imprescindible llevar a cabo la modificación del diseño para su optimización.

De esta manera, se graficó un segundo bosquejo ensanchando la zona alrededor de los orificios para mejorar su resistencia y obtener una mayor vida útil tal como se muestra en la Fig. 3.30

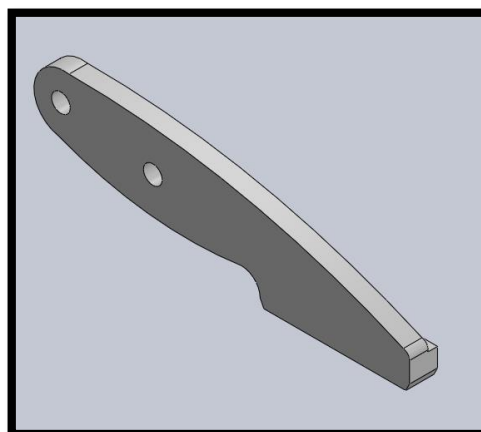


Figura 3.30: Segundo Bosquejo
Fuente: Propia

Nuevamente realizamos ambos ensayos especificados anteriormente:

- **Ensayo Estático:** Se llevó a cabo teniendo las mismas consideraciones que el estudio previo y se obtuvo los siguientes resultados:

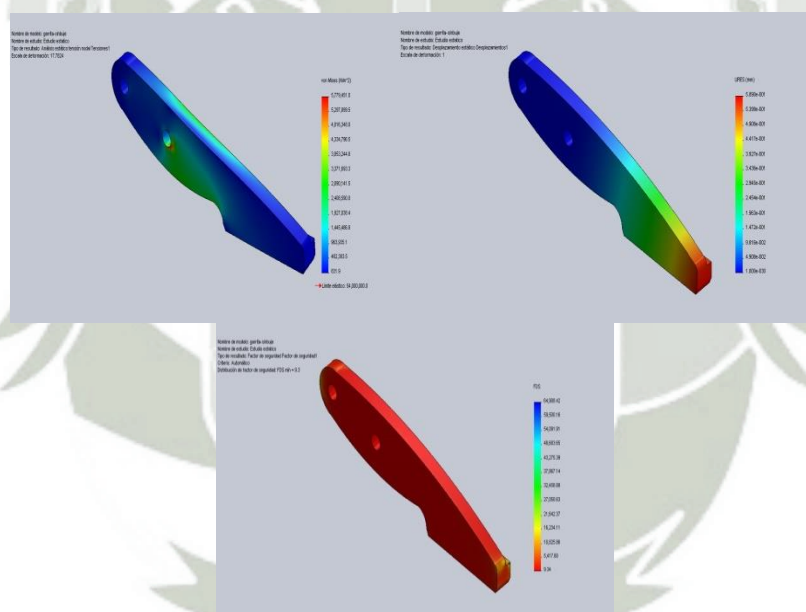


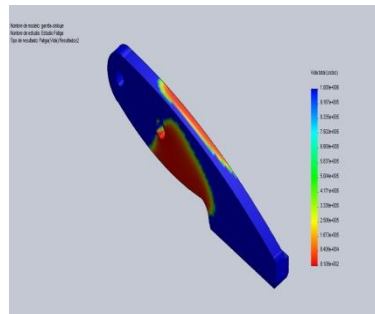
Figura 3.31: Gráficas de Ensayo Estático
Fuente: Propia

Tabla 3.6: Datos obtenidos de Ensayo Estático

Tensiones MPa	Desplazamientos mm	Factor de Seguridad
5.7	0.589	9.3

Fuente: Propia

- **Ensayo de Fatiga:** Al igual que el ensayo estático, se aplicó los mismos datos que el análisis de fatiga precedente para su simulación, obteniendo el siguiente gráfico:


Figura 3.32: Gráfica de Ensayo de Fatiga

Fuente: Propia
Tabla 3.7: Datos obtenidos de Ensayo Fatiga

Vida Útil (Valor mínimo)
810.8

Fuente: Propia

Ejecutados los estudios anteriores, podemos percibir que si bien es cierto mejoró la resistencia a la tensión y la vida útil de la pieza, todavía no es óptima puesto que se observa aún la zona del segundo agujero muy afectada por efecto de la carga.

Es así como se desarrolló un tercer análisis, modificando el contorno inferior del elemento, de tal manera que sea directo hasta el final del componente sin curvatura alguna, además se agregó un buje de bronce en cada agujero con la finalidad de mejorar la resistencia en este punto tal como se observa en la Figura 3.33.


Figura 3.33: Tercer Bosquejo

Fuente: Propia

En concordancia con el procedimiento efectuamos los ensayos requeridos, tanto el estático como el de fatiga.

- **Ensayo Estático:** Se desarrolló tomando en cuenta los mismos datos que el primer ensayo, para adquirir así los siguientes resultados:

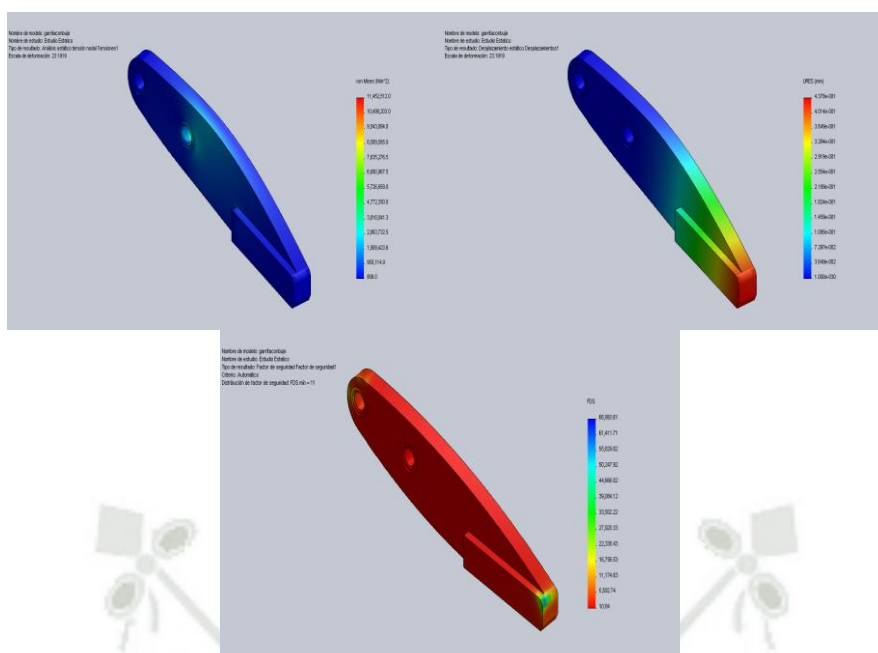


Figura 3.34: Gráficas de Ensayo Estático

Fuente: Propia

Tabla 3.8: Datos obtenidos de Ensayo Estático

Tensiones MPa	Desplazamientos mm	Factor de Seguridad
11.45	0.4378	11

Fuente: Propia

- **Ensayo de Fatiga:** Llevado a cabo mediante la aplicación de datos ya especificados en el primer estudio, permitiéndonos conseguir el siguiente gráfico presentado en la Fig. 3.35

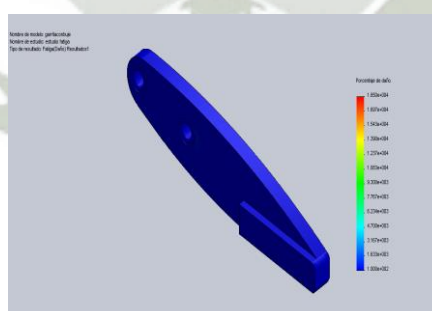


Figura 3.35: Gráfica de Ensayo de Fatiga

Fuente: Propia

Tabla 3.9: Datos obtenidos de Ensayo Fatiga

Vida Útil (Valor mínimo)
5405

Fuente: Propia

Analizando estos resultados podemos notar que ciertamente mejoró la resistencia y la vida útil de la garra, pero seguimos percibiendo aún un inconveniente en el segundo agujero aunque con menor intensidad, lo cual nos llevó a concluir la existencia de un fenómeno denominado aplastamiento, cuya ocurrencia depende del material de construcción mas no del diseño elaborado.

Entendiendo la razón del problema, pasamos a realizar una última variación en la cual no es necesario el uso de los bujes de bronce, es así como se obtuvo el diseño de la estructura final de la pieza, tal como se puede observar en la Figura 3.36.

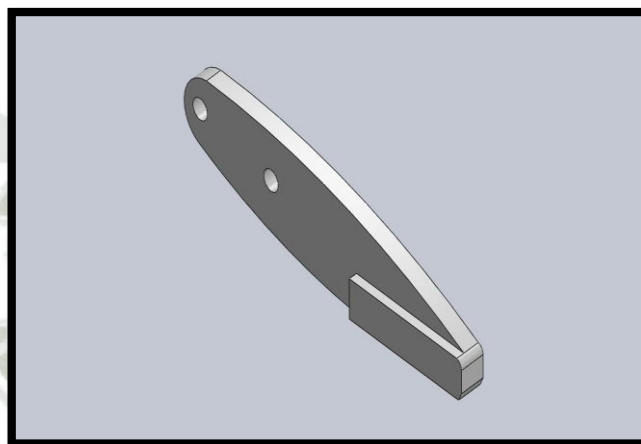


Figura 3.36: Bosquejo Final
Fuente: Propia

Sometemos la geometría final del elemento estudiado a los ensayos respectivos:

- **Ensayo Estático:** Tras su realización podemos lograr los siguientes resultados:

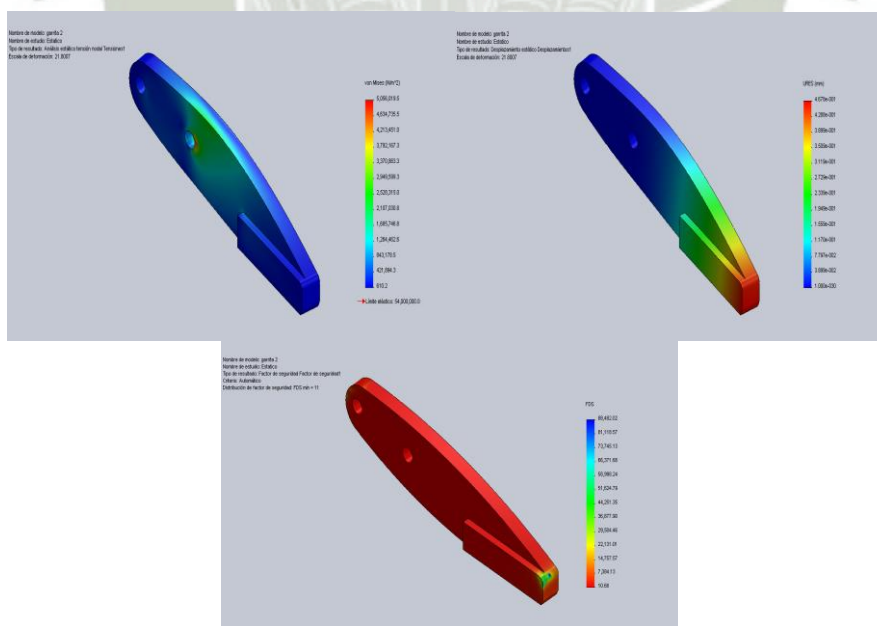


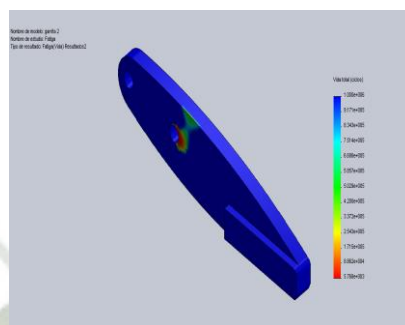
Figura 3.37: Gráficas de Ensayo Estático
Fuente: Propia

Tabla3.10: Datos obtenidos de Ensayo Estático

Tensiones MPa	Desplazamientos mm	Factor de Seguridad
5.06	0.4678	11

Fuente: Propia

- **Ensayo de Fatiga:** Se llevó a cabo utilizando las consideraciones anteriores y nos otorga el siguiente gráfico:


Figura 3.38: Gráfica de Ensayo de Fatiga

Fuente: Propia

Tabla 3.11: Datos obtenidos de Ensayo Fatiga

Vida Util (Valor mínimo)
5768

Fuente: Propia

Dado estos últimos resultados podemos asegurar que la pieza soportará el peso indicado y además tendrá una vida útil aceptable con un factor de seguridad de 11, lo cual está dentro de los estándares óptimos requeridos.

Posteriormente, se precisó detalles del portagarra a partir de su bosquejo preliminar, definiendo así la geometría de la pieza a través de la reducción de su peso; además se le adicionó un espacio adecuado para el soporte donde irá alojado el motor. Todos estos cambios se realizaron con el objetivo de obtener una pieza visiblemente estética, lo cual es posible observar en la figura 3.39.

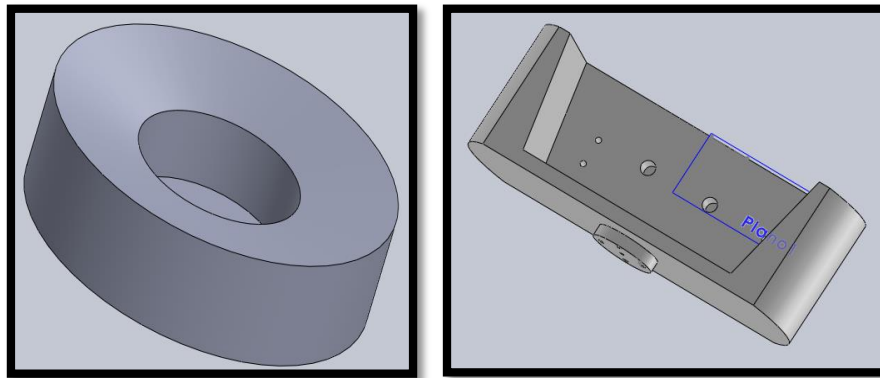


Figura 3.39: Izquierda: bosquejo inicial de portagarra; Derecha: bosquejo final de portagarra
Fuente: Propia

Seguidamente se realizó el ensamble del último eslabón de la cadena cinemática (Fig. 3.40).

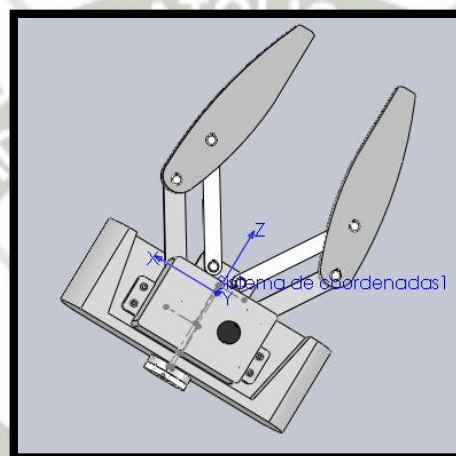


Figura 3.40: Ensamble de la garra
Fuente: Propia

Habiendo determinado la estructura física del efector final, se procedió a llevar a cabo el segundo paso del procedimiento, el mismo que tiene como finalidad la obtención del valor numérico de los torques necesarios en cada una de las articulaciones del manipulador robótico; para ello, se elaboró un script en Matlab, denominado “TRAYECTORIACIRC” (revisar Anexo A), a partir de las ecuaciones obtenidas en el análisis dinámico del robot, las mismas que claramente nos muestran que los pares de torsión quedan definidos en función de los valores angulares de las articulaciones, velocidades y aceleraciones articulares, momentos de inercia, masas, longitudes y fuerza de gravedad.

El programa creado consta básicamente de seis partes, donde las cuatro primeras están dirigidas a la obtención de los datos requeridos para calcular el valor máximo de cada uno de los momentos de torsión y las dos últimas, referidas a los torques específicamente y a generar entornos gráficos para un mayor análisis y comprensión. A

continuación, se describirá con mayor detalle cada una de las divisiones que componen el script realizado:

- **Matriz de Valores articulares:** Tiene como propósito, obtener una matriz que contenga los valores angulares de las articulaciones durante el desplazamiento del manipulador por una trayectoria específica; para ello se hizo uso inicialmente del análisis cinemático directo y del toolbox de PeterCorke, con la finalidad de generar una trayectoria circular compleja entre dos puntos específicos, siendo el primero el punto de partida correspondiente a la posición inicial del manipulador, donde los valores angulares son cero grados en su totalidad y el segundo, el punto de llegada. Posteriormente se empleó las relaciones obtenidas durante el análisis cinemático inverso, el mismo que junto a la matriz de los valores que indica la posición del efector final del robot en tiempos determinados de la trayectoria, nos permite lograr el objetivo planteado.
- **Matriz de Velocidades Articulares:** El programa “TRAYECTORIACIRC” calcula primero las velocidades lineales y angulares del efector final, a partir de la integración de sus aceleraciones, las mismas que son ingresadas por el usuario en forma de vector. Con este dato, el script construye un movimiento que parte del reposo y acelera en 0.1 segundo para alcanzar así una velocidad constante durante un tiempo específico y luego desacelerar de igual modo en 0.1 segundo y llegar nuevamente al estado de reposo. Seguidamente se hace uso del análisis cinemático diferencial que nos permite hallar la matriz jacobiana geométrica inversa, la misma que junto a la matriz de velocidades del efector final anteriormente calculada nos permite definir finalmente las velocidades articulares.
- **Matriz de Aceleraciones Articulares:** Para encontrar la matriz de aceleraciones articulares, el script realiza un proceso de derivación de la matriz de velocidades articulares obtenida en el punto anterior.
- **Momentos de Inercia, masa y longitudes:** El programa Solidwork, utilizado para graficar cada una de las partes que componen el brazo robótico, nos permite determinar datos como momentos de inercia y masas de acuerdo con el material elegido y a un sistema de coordenadas determinado, además de obtener longitudes exactas hacia el centro de masa de cada pieza. Estos datos son ingresados en el programa “TRAYECTORIACIRC”, para conseguir así los valores de los momentos de torsión del robot.
- **Torques:** Reemplazados todos los datos anteriormente calculados, es posible obtener cuatro vectores cuyos valores corresponden a los torques de cada articulación, en un tiempo determinado durante el movimiento del manipulador por la trayectoria dada; así mismo, el script “TRAYECTORIACIRC” nos proporciona como resultado el máximo valor de cada uno de los vectores.

- **Gráficos:** Conciene la última fase del programa, en donde se utilizan algunos comandos del Toolbox Robotics de Peter Corke para generar la primera gráfica en la que es posible ver el movimiento del robot siguiendo la trayectoria construida; de igual manera, el programa crea dos entornos gráficos adicionales, el primero muestra los torques vs tiempo en una misma figura de tal modo que es posible compararlos entre sí, y el segundo muestra también la evolución de los momentos de torsión en el tiempo pero en cuatro figuras distintas especificando su máximo valor con un punto.

Finalmente, con ayuda del script realizado en Matlab y las herramientas que nos ofrece el Solidwork, se desarrolló un proceso iterativo intercalando el uso de ambos softwares para obtener uno a uno los pares de torsión, concluyendo así en la elección de los actuadores que se utilizarán para dar movimiento a cada eslabón del manipulador robótico.

Para realizar dicho proceso, es conveniente empezar por el análisis del último elemento de la cadena cinemática, obteniendo así del Solidwork algunos datos esenciales tales como el momento de inercia, masa, longitud al centro de masa, y longitud total del portagarra (Fig. 3.41 y 3.42). Es importante considerar durante la adquisición del momento de inercia, la creación de un sistema de coordenadas auxiliar, cuya orientación de los ejes concuerde con el sistema previsto en el Denavith Hartenberg.

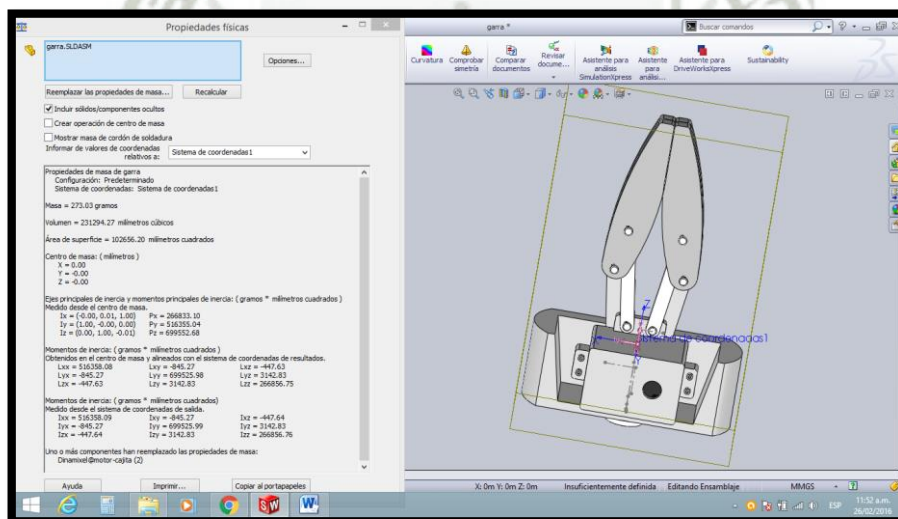


Figura 3.41: Momento de Inercia y masa obtenidos del Software Solidwork

Fuente: Propia

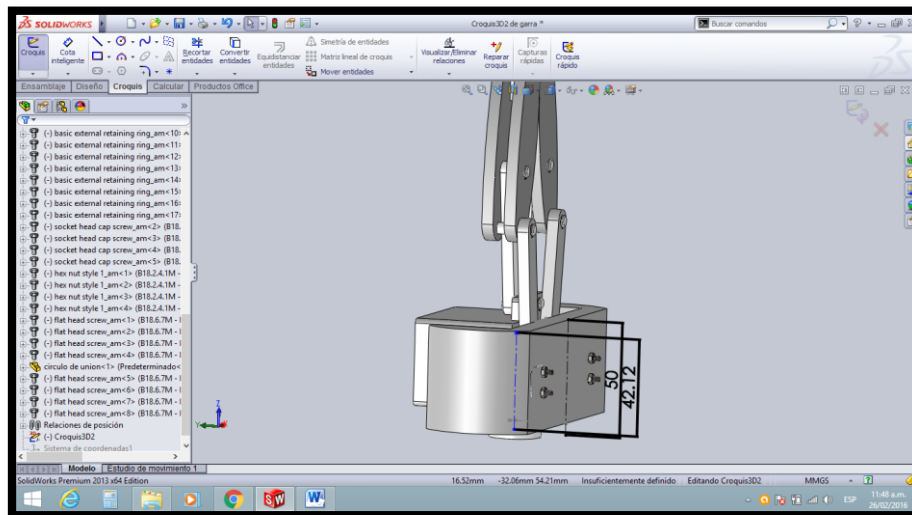


Figura 3.42: Longitud al centro de masa y Longitud total del portagarra obtenidas del software Solidwork
Fuente: Propia

Seguidamente ingresamos al programa elaborado en Matlab, “TRAYECTORIACIRC.m”, el mismo que requiere tanto los datos obtenidos previamente como el intervalo de tiempo, la orientación del último grado de libertad y el vector de aceleración lineal y angular del efector final.

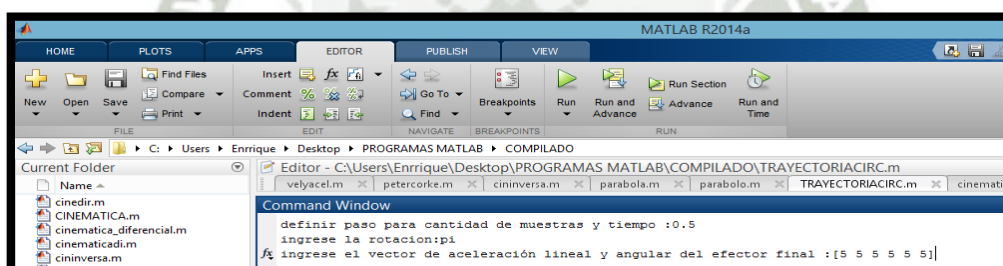


Figura 3.43: Aplicando programa “TRAYECTORIACIRC.m”
Fuente: Propia

Como se puede observar en la Figura 3.43, los valores tomados en cuenta son de 0.5 segundos en el caso del intervalo de tiempo, 180° en la orientación y 5 m/s^2 para la aceleración lineal y angular, la misma que determina una velocidad constante de 0.5 m/s, dato que es considerado adecuado experimentalmente para poder desplazar las sustancias líquidas tóxicas en cuestión, sin el riesgo de provocar algún derrame.

Ejecutado el script, adquirimos el resultado del par torsión correspondiente a la garra, el mismo que indica un torque máximo durante la trayectoria, de 0.0013 Nm (Fig. 3.44). Este dato define la elección del actuador que dará movimiento a la última articulación del brazo robótico, para ello se revisó el catálogo de servomotores Robotics Dynamixel en donde pudimos observar que el motor AX-12W, es el que ofrece menor torque, siendo este de 0.2 Nm; sin embargo, con el propósito de considerar un menor costo

durante el desarrollo de la tesis se prefirió el servomotor TowerPro de alta calidad, SG-5010, el mismo que nos entrega un torque de 0.52 Nm a 4.8 V.

$$T4 = \begin{bmatrix} 0 & 0.0013 & 0.0000 & 0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0001 \\ -0.0001 & -0.0000 & -0.0000 & -0.0000 & -0.0009 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}$$

$$T4m = 0.0013$$

Figura 3.44: Torque Máximo Garra
Fuente: Propia

Teniendo definido el torque máximo, así como el motor a utilizar se procedió a detallar el eslabón 2, que siendo una pieza compacta habría que convertirla en una pieza hueca para poder insertar el motor y hacer pasar todos los cables por dentro cumpliendo así con la estética, además se sustrajo material excesivo para controlar el peso y se le agregó dos tapas que permitirán la manipulación del cableado. En el transcurso de este diseño se realizó una variación resaltante en cuanto a la determinación de cambiar el servomotor Dynamixel AX-12W por un Tower Pro SG 5010 y debido a que las dimensiones de ambos motores son diferentes se tuvo que rediseñar el agujero y la forma de ubicación, por lo tanto, se agregó un soporte al lado izquierdo del elemento motriz para su sujeción; así mismo con el fin de evitar algún corte al operador se vio por conveniente redondear todas las aristas de la pieza. (Fig. 3.45)

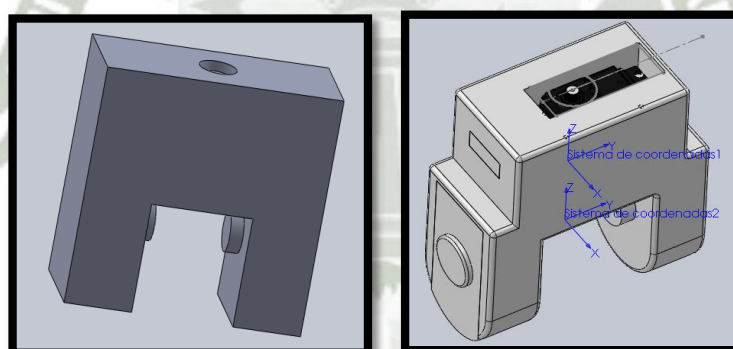


Figura 3.45: Izquierda: bosquejo inicial del eslabón 2; Derecha: bosquejo final del eslabón 2
Fuente: Propia

Posteriormente, repetimos el procedimiento anterior, pero esta vez tomando en cuenta la geometría modificada del eslabón 2 del manipulador robótico, adquiriendo así del Solidwork su masa, momento de inercia, longitud al centro de masa y dimensión entre articulaciones (Figuras 3.46 y 3.47).

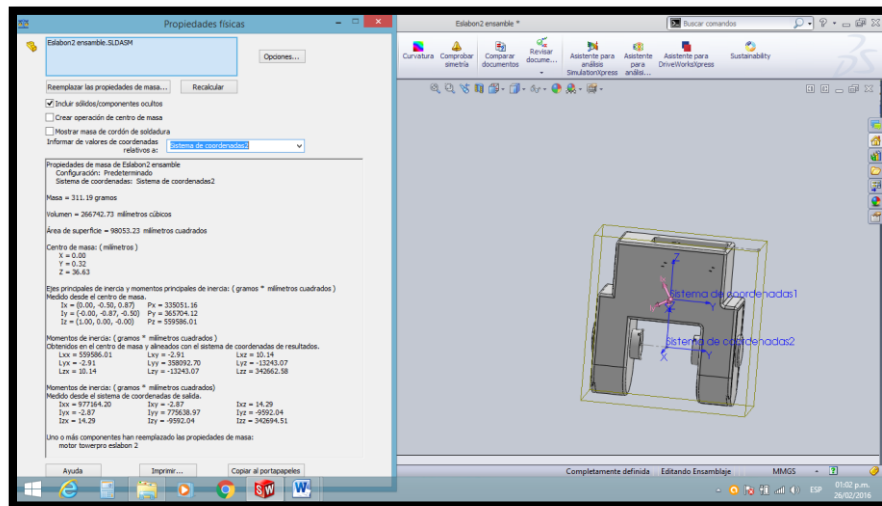


Figura 3.46: Momento de Inercia y masa obtenidos del Software Solidwork

Fuente: Propia

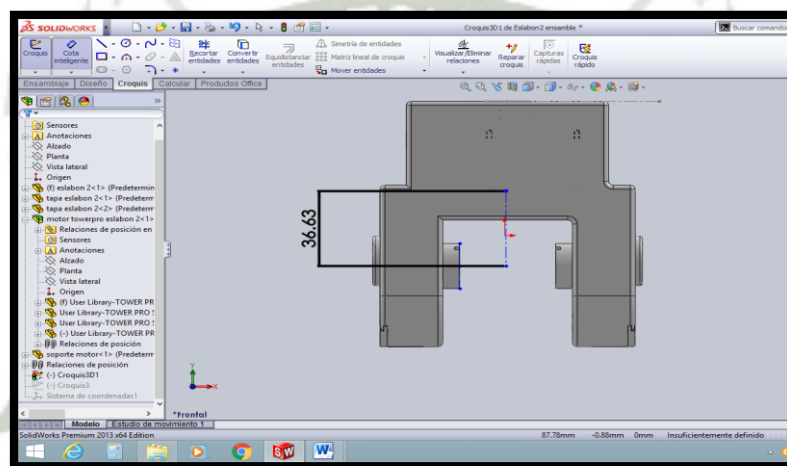


Figura 3.47: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software Solidwork

Fuente: Propia

De esta manera, ingresamos luego al programa creado en Matlab y lo ejecutamos considerando el ingreso de los nuevos datos del eslabón 2, adquiriendo así un momento de torsión máximo de 1.0708 N-m (Fig. 3.48), el mismo que nos permitió definir el uso del servomotor Dynamixel AX-12A, como el actuador adecuado para la articulación en estudio; puesto que ofrece un torque de 1.53 N-m.

$T_3 =$

1.0530	1.0708	1.0430	0.9854	0.8629	0.6844	0.4780
0.2880	0.1797	0.1865	0.2163	0.2231	0.2145	

$T_{3m} =$

1.0708

Figura 3.48: Torque Máximo Eslabón 2

Fuente: Propia

Seguidamente rectificamos el diseño del eslabón 1, siendo su desarrollo muy parecido al eslabón 2, realizando así una abertura interna de manera que permita el paso de los cables del motor y su estética no se vea afectada, además se creó un espacio conveniente para la ubicación del actuador, y se dejó solo unas barras donde ira atornillado el mismo. Al igual que la pieza anterior cuenta con dos tapas con el mismo funcionamiento. (Fig. 3.49).

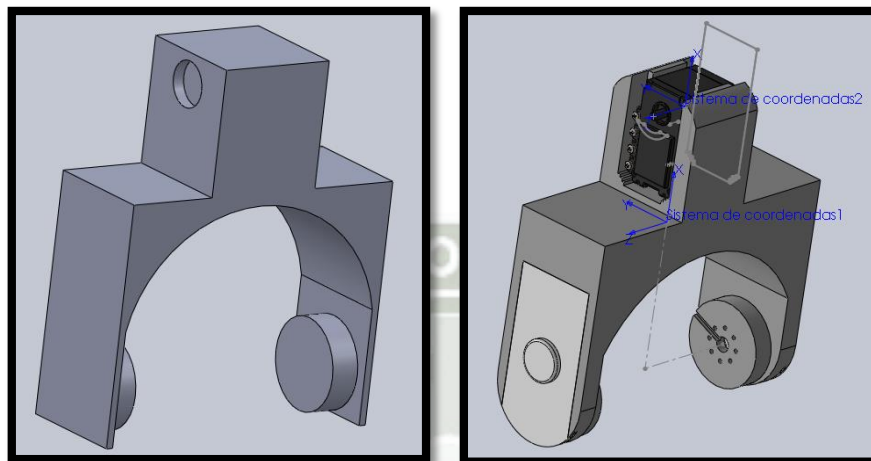


Figura 3.49: Izquierda: bosquejo inicial del eslabón 1; Derecha: bosquejo final del eslabón 1
Fuente: Propia

Tras haber modificado la estructura del eslabón 1, obtenemos al igual que en el caso anterior, su masa, momento de inercia, longitud al centro de masa y dimensión entre articulaciones, tal como se puede observar en las Figuras 3.50 y 3.51.

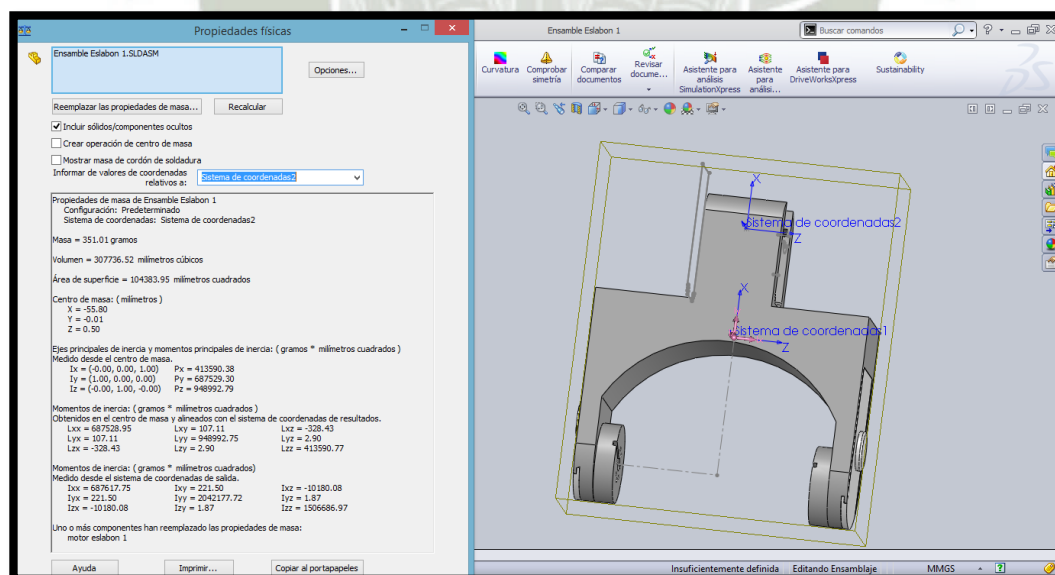


Figura 3.50: Momento de Inercia y masa obtenidos del Software Solidwork
Fuente: Propia

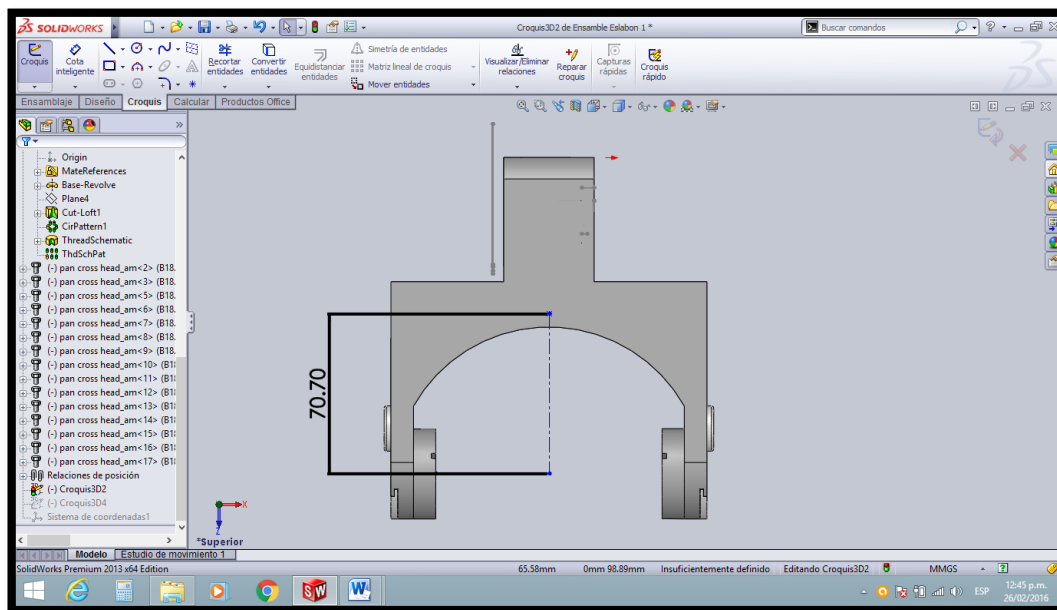


Figura 3.51: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software Solidwork

Fuente: Propia

De igual modo, con los nuevos datos, ejecutamos el script “TRAYECTORIACIRC” y obtenemos un par torsión máximo de 2.7558 N-m en la articulación 2 del manipulador (Fig. 3.52), siendo ésta considerada la más crítica del robot; puesto que es aquí donde el brazo realiza el mayor esfuerzo durante su movimiento por una trayectoria dada, por este motivo es que se vio por conveniente considerar el uso de dos servomotores Dynamixel RX- 24F, los mismos que nos otorgan un torque de 2.6 N-m cada uno, lo que en conjunto daría un momento de torsión de 5.2 N-m.

$T_2 =$

2.6419	2.7558	2.6165	2.4720	2.1649	1.7173	1.2004
0.7248	0.4516	0.4611	0.5388	0.5591	0.5061	

$|T_{2m} =$

2.7558

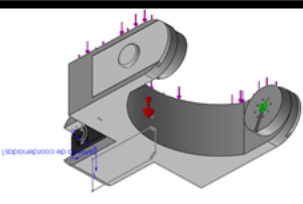
Figura 3.52: Torque Máximo Eslabón 1

Fuente: Propia

Tomando en cuenta que esta pieza es la que debe soportar el mayor torque de todo el manipulador vemos necesario realizar dos ensayos mecánicos para confirmar si el diseño es lo suficientemente resistente para estos esfuerzos y analizar el número máximo de ciclos de vida del eslabón antes de fallar.

Se realizó en primera instancia el ensayo estático para lo cual se agregó una fuerza de 11N simulando el peso de la carga máxima y además del peso de los eslabones a

soportar; de igual manera, se tomó en cuenta la gravedad para que el estudio sea lo más real posible. (Fig. 3.53)



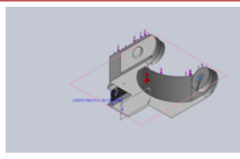
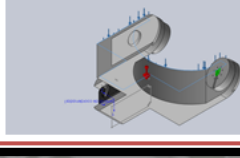
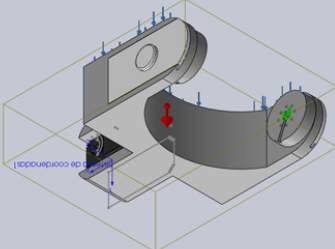
Nombre de carga	Cargar imagen	Detalles de carga
Gravedad-1		<u>Referencia:</u> Planta <u>Valores:</u> 0 0 9.81 <u>Unidades:</u> SI
Fuerza-1		<u>Entidades:</u> 1 cara(s) <u>Tipo:</u> Aplicar fuerza normal <u>Valor:</u> 11 N

Figura 3.53: Fuerzas aplicadas al eslabón 2
Fuente: Propia

Nuevamente se debe elegir un tamaño mínimo de cada elemento de malla en función del espesor de la pieza para tales fines nosotros tuvimos en este ensayo un tamaño mínimo de malla de 4.81 mm. (Fig. 3.54)



<u>Tipo de malla</u>	Malla sólida
<u>Mallador utilizado:</u>	Malla estándar
<u>Transición automática:</u>	Desactivar
<u>Incluir bucles automáticos de malla:</u>	Desactivar
<u>Puntos jacobianos</u>	4 Puntos
<u>Tamaño de elementos</u>	4.80806 mm
<u>Tolerancia</u>	0.240403 mm
<u>Calidad de malla</u>	Elementos cuadráticos de alto orden

Figura 3.54: Tamaño mínimo de malla
Fuente: Propia

Los resultados obtenidos, después de haber realizado el estudio, son los siguientes:

- **Tensiones:** Nos muestra un gráfico, donde es posible analizar las tensiones que puede soportar el elemento como máximo, en función del límite elástico del

material. Como se puede observar en la Figura 3.55, la parte más crítica de la pieza son los agujeros de sujeción en los laterales de la parte inferior, sin embargo, su máximo esfuerzo es de 2 MPa por lo que es aceptable.

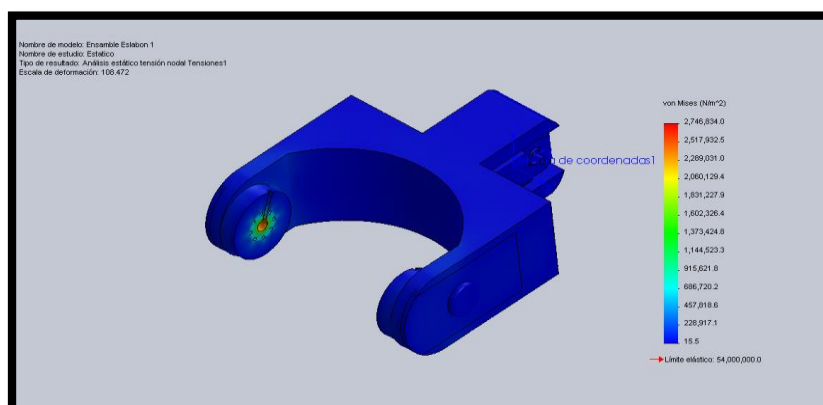


Figura 3.55: Gráfica de Tensiones

Fuente: Propia

- **Desplazamientos:** Nos muestra en milímetros cual es el desplazamiento de la pieza respecto a las fuerzas soportadas, teniendo como mayor valor 0.1484 mm.

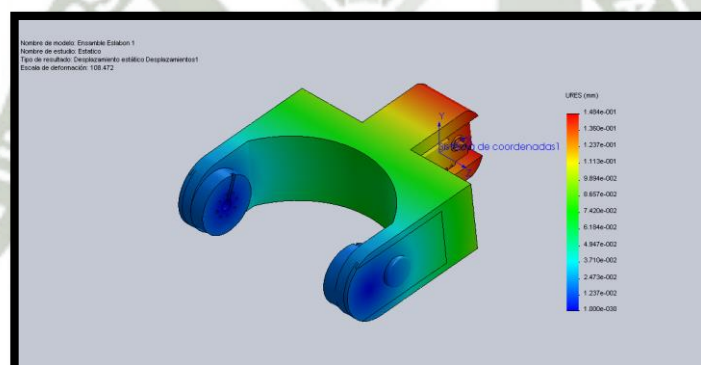


Figura 3.56: Gráfica de Desplazamientos

Fuente: Propia

- **Factor de Seguridad:** De este resultado podemos concluir que la pieza puede soportar 60 veces más carga que a la que fue sometida

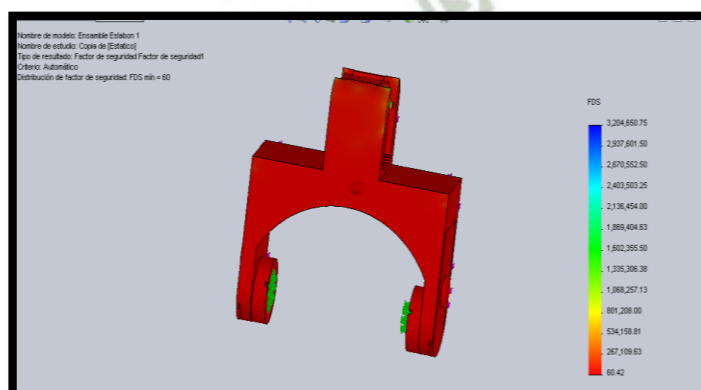


Figura 3.57: Gráfica de Factor de Seguridad

Fuente: Propia

Como ya se explicó en anteriores párrafos el ensayo estático no es suficiente para definir si una pieza está o no lista para su manufactura, se debe tener en cuenta un ensayo más como es el de fatiga que nos permitirá analizar el número de repeticiones que ha de soportar la pieza antes de fallar. Del mismo modo se creará un suceso de 1000000 de repeticiones para simular la vida infinita de eslabón (Fig. 3.58).

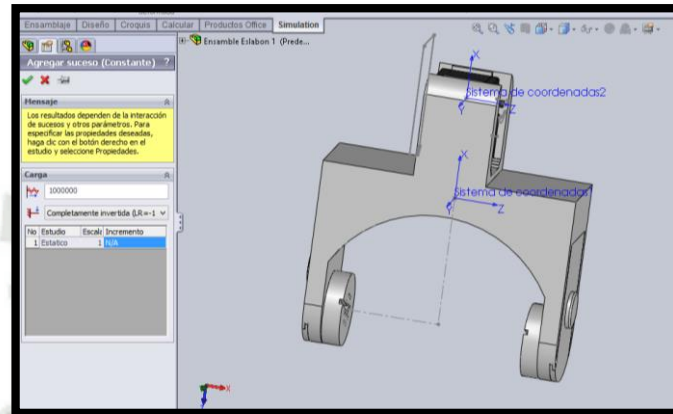


Figura 3.58: Definición de suceso

Fuente: Propia

Seguidamente se debe volver a definir la curva S – N para este eslabón, aunque ciertamente será la misma ya que ambos elementos usan el mismo material. (Fig. 3.59)

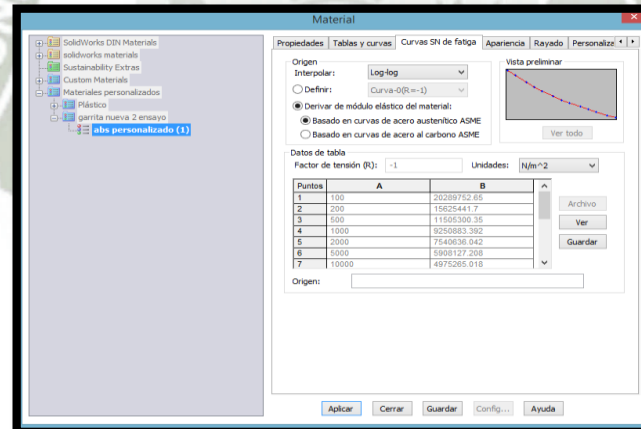
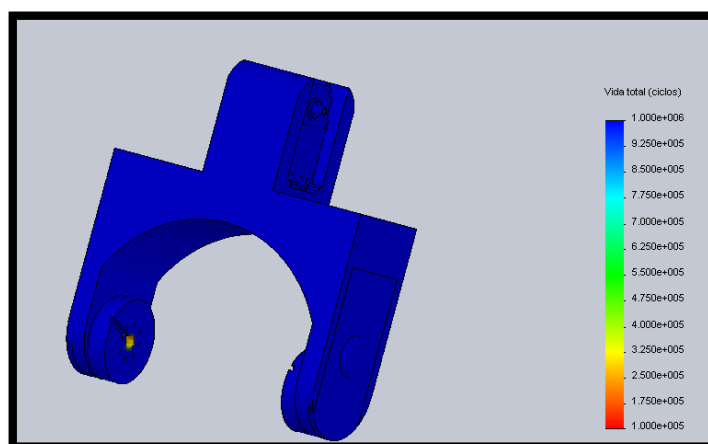


Figura 3.59: Definición de material

Fuente: Propia

Tras realizado el ensayo, adquirimos el siguiente resultado

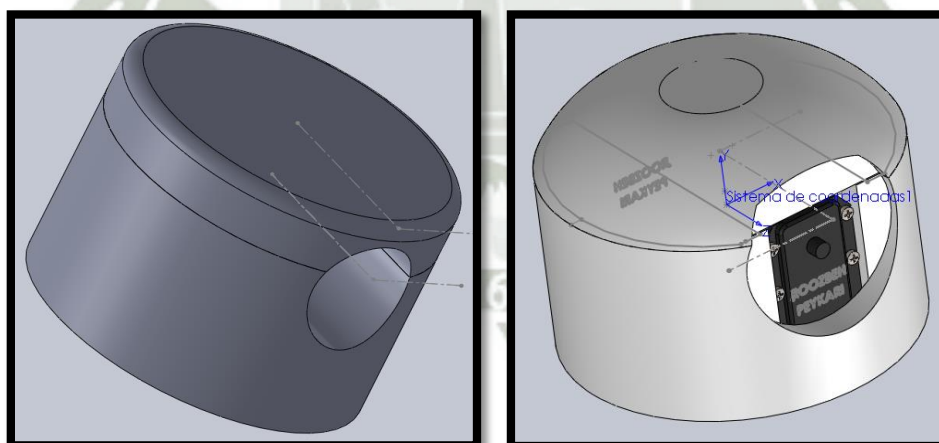
Vida total: Indica el número de repeticiones máximas que puede soportar la pieza en distintos puntos a lo largo de ella. En caso de este elemento podemos notar que lo mínimo a resistir son 100000 repeticiones en los agujeros siendo un número aceptable para un buen funcionamiento.


Figura 3.60: Gráfica de Vida Total

Fuente: Propia

En consecuencia, del ensayo estático y de fatiga se puede concluir que esta pieza se encuentra apta para trabajar de la forma más óptima en nuestro prototipo, con alta resistencia y una vida útil aceptable según la función a realizar.

A continuación, proseguimos con la base superior la cual consta de dos partes, una para el alojamiento de los motores y la otra que servirá como tapa para salvaguardar la estética. Así, durante el desarrollo de la estructura de estos elementos se suprimió abundante material a fin de reducir su peso y se diseñó el alojamiento para los dos motores (Fig. 3.61).


Figura 3.61: Izquierda: bosquejo inicial de la base superior; Derecha: bosquejo final de la base superior

Fuente: Propia

Calculamos el torque correspondiente a la articulación 1, para lo cual con ayuda del Solidwork, adquirimos la masa, momento de inercia, longitud al centro de masa y dimensión entre articulaciones de la base tal como se puede observar en las Figuras 3.62 y 3.63.

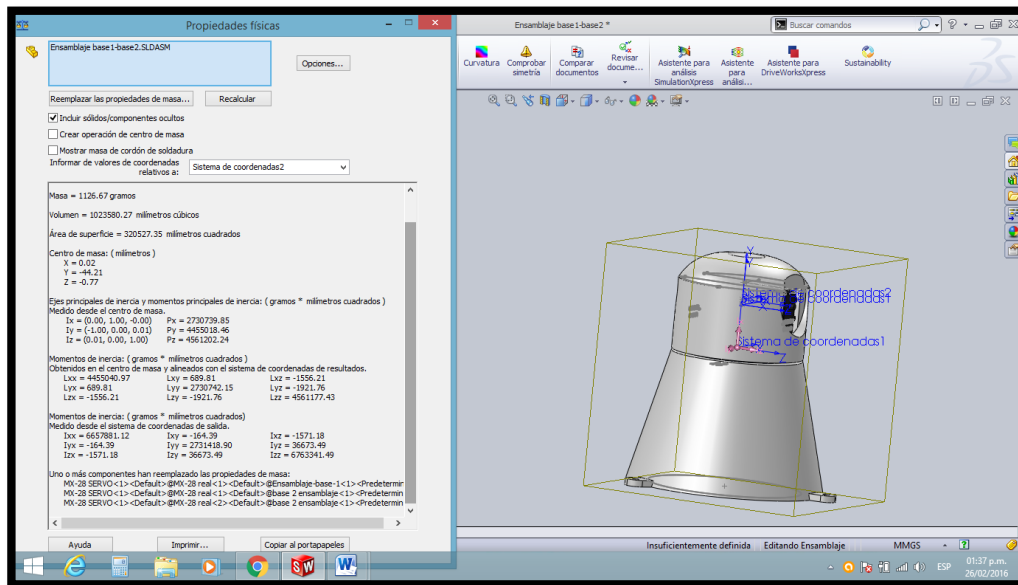


Figura 3.62: Momento de Inercia y masa obtenidos del Software Solidwork

Fuente: Propia

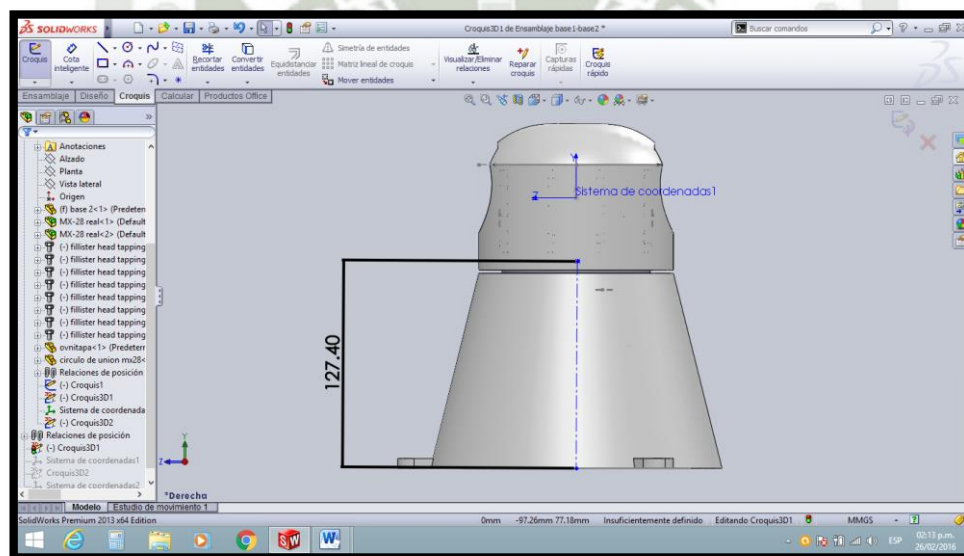


Figura 3.63: Longitud al centro de masa y dimensión entre articulaciones obtenidas del software

Solidwork

Fuente: Propia

Con los nuevos datos obtenidos, ingresamos al Matlab y corremos el programa creado para conseguir así el torque máximo perteneciente al primer actuador, siendo éste de 0.0081N-m (Fig. 3.64), el mismo que determina la selección del servomotor Dynamixel MX-28T como el elemento motriz apropiado para la articulación 1; puesto que, según sus especificaciones, éste es capaz de otorgar un momento de torsión de 2.5 N-m.

TOR1 =

0	0.0081	-0.0001	-0.0001	-0.0001	0.0000	0.0002	0.0006
0.0008	-0.0020	-0.0011	-0.0003	-0.0059			

T1m =

0.0081

Figura 3.64: Torque Máximo Base

Fuente: Propia

De esta manera, procedemos a modificar la geometría de la base inferior, siendo esta la pieza con mayor peso para mantener la estabilidad, sin embargo durante su corrección, dejó de ser compacta para permitir el alojamiento interior del motor y así lograr que sea más ligera; así mismo, se le agregó 3 pestañas en el contorno inferior a fin de obtener una sujeción completa a la mesa y se construyó un canal en la parte superior, el mismo que alojará un rodamiento axial con el objetivo de disminuir la fricción entre la base 1 y 2 durante el giro. (Fig. 3.65)

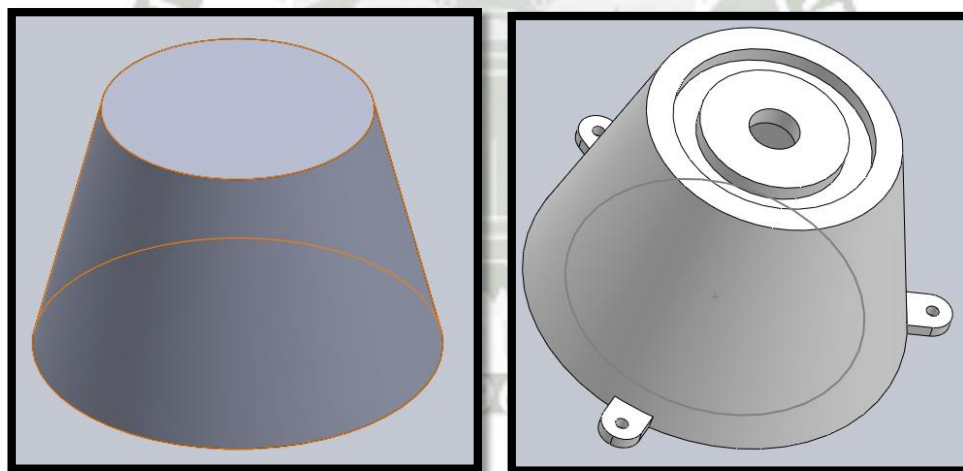


Figura 3.65: Izquierda: bosquejo inicial de la base inferior; Derecha: bosquejo final de la base inferior

Fuente: Propia

Los gráficos obtenidos tras la última ejecución del script “TRAYECTORIACIRC” son los mostrados en las Figuras 3.66, 3.67 y 3.68. En donde el primer diagrama muestra una representación del manipulador robótico de acuerdo a su configuración y la trayectoria tridimensional que sigue, el segundo esquema nos presenta la evolución de cada uno de los torques en función al tiempo considerado, siendo el par torsión de la segunda articulación mayor en comparación al resto, y la tercera figura nos muestra las tendencias de los momentos de torsión en el tiempo señalando en cada uno su valor máximo.

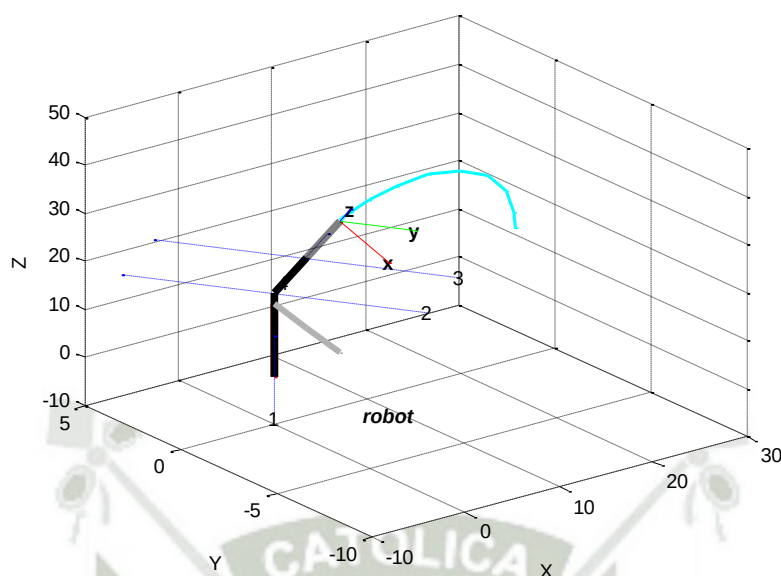


Figura 3.66: Representación del manipulador robótico de acuerdo a su configuración
Fuente: Propia

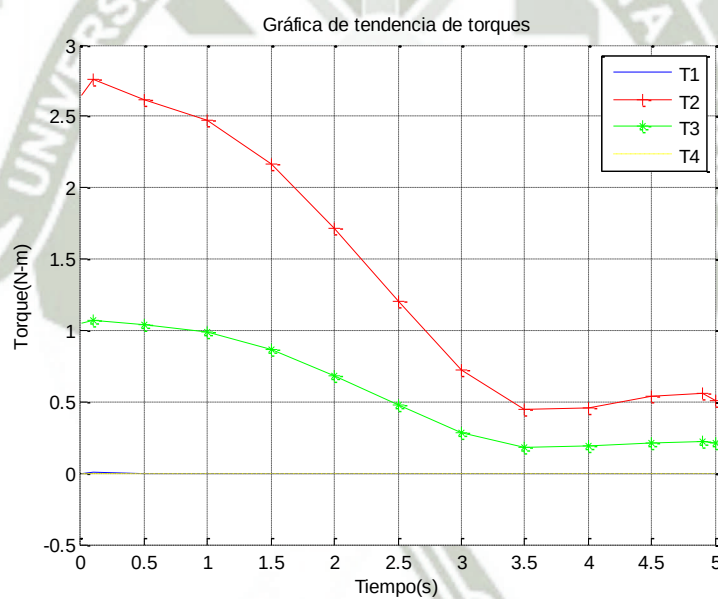


Figura 3.67: Evolución de cada uno de los torques en función al tiempo
Fuente: Propia

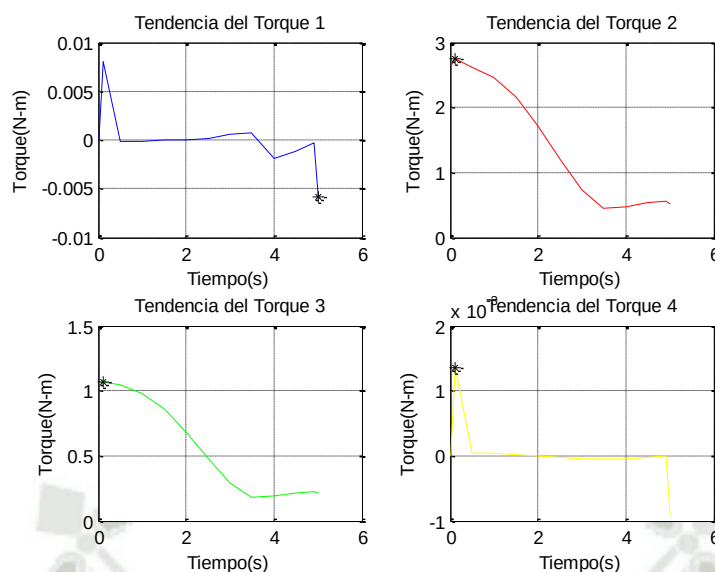


Figura 3.68: Tendencias de los momentos de torsión en el tiempo

Fuente: Propia

En la tabla 3.12 podemos observar los datos más importantes de cada eslabón necesarios para una elección adecuada de los actuadores, además también se muestra un resumen de los ensayos de las principales piezas.

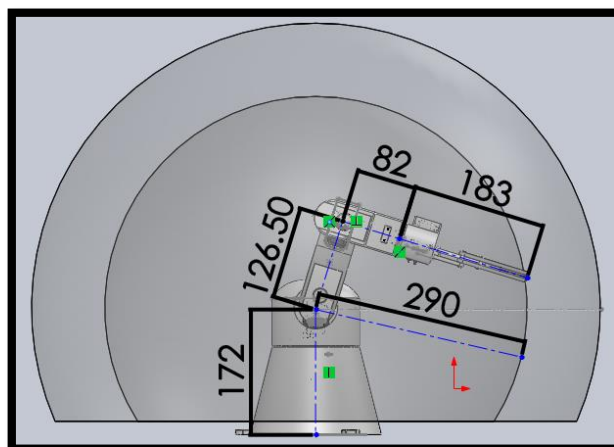
Tabla 3.12: Datos de cada Eslabón

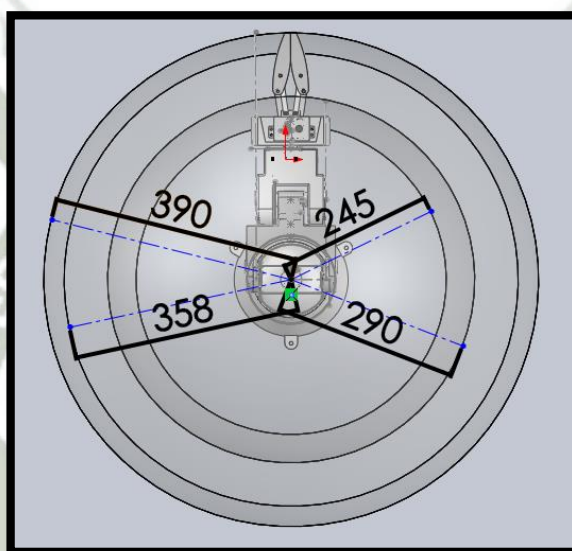
	Base	Eslabón 1	Eslabón 2	Garra
Masa(kg)	0.64932	0.35101	0.31119	0.2703
Izz(gm*mm^2)	6763341.49	1506686.97	342694.51	266856.76
Lc (mm)	127.4	70.7	36.63	42.12
L (mm)	172	126.5	82	50
Ensayo Estático				
Tensiones (MPa)	-----	2.74	-----	5.06
Desplazamiento (mm)	-----	0.1484	-----	0.4678
Factor de Seguridad	-----	60	-----	11
Ensayo de Fatiga				
Vida Útil (mín)	-----	100000	-----	5768

Fuente: Propia

3.1.7. Espacio de Trabajo:

El espacio de trabajo de un robot es grupo de puntos que puede ser alcanzado por su efector final, esto se ve determinado por las longitudes y ángulos de rotación de las articulaciones; teniendo en cuenta esto es que en las figuras 3.69 y 3.70 se muestra el alcance del brazo robótico, así como algunas dimensiones


Figura 3.69: Vista lateral del espacio de trabajo

Fuente: Propia

Figura 3.70: Vista superior del espacio de trabajo

Fuente: Propia
Tabla 3.13: Ángulos de rotación de cada articulación

Articulación	Rango de giro
Base	De 0° a 360°
Eslabón 1	De -10° a 190°
Eslabón 2	De -90° a 90°
Eslabón 3	De 0° a 180°

Fuente: Propia

En la tabla 3.13 se muestra los rangos de giro de cada articulación, siendo estos: base, eslabón 1, eslabón 2 y eslabón 3 o garra respectivamente.

3.2. Diseño Electrónico:

Una vez definida en su totalidad la estructura mecánica del prototipo, procedemos a llevar a cabo el diseño electrónico del mismo, en el que se pone a prueba nuestra

capacidad de ingenio, creatividad y conocimiento para determinar como punto inicial y bajo ciertos criterios, los componentes electrónicos necesarios y luego desarrollar así las conexiones entre ellos mediante un circuito, el mismo que posteriormente será sometido a distintos ensayos de funcionamiento.

3.2.1. Elección de Componentes Electrónicos:

3.2.1.1. Servomotores:

Los servomotores constituyen uno de los elementos electrónicos más importantes en el desarrollo de esta tesis, puesto que son los encargados de dar movimiento a cada una de las articulaciones del manipulador robótico; es por ello que en el apartado anterior se realizó un análisis exhaustivo acerca de la elección de los mismos, basándonos en el torque máximo que necesitan los actuadores durante una trayectoria específica. Es así como se tomó la decisión de emplear cinco servomotores Dynamixel y un Tower-Pro. (Figuras 3.71, 3.72.)



Figura 3.71: Servomotores Dynamixel

Fuente: Robotis e-Manual, 2010



Figura 3.72: Servomotor Tower Pro SG5010

Fuente: Propia

3.2.1.2. Fuente de Alimentación Externa de 10 Amp:

Una vez definidos los tipos de actuadores a emplear, determinamos los elementos necesarios para su funcionamiento, donde la fuente de alimentación requerida cumple un papel fundamental, pues es el dispositivo encargado de suministrar tensión para su operatividad.

Es importante tomar en cuenta algunos criterios a la hora de elegir la fuente de tensión, primando así la potencia que debe tener; para ello, consideramos una conexión en paralelo de los actuadores, los mismos que en función a su hoja de datos (Anexo B) deben ser alimentados por una tensión DC de 12 Voltios en el caso de los Dynamixel y 5 Voltios para el Tower Pro. En cuanto a la intensidad de corriente, se puede observar que varía de acuerdo al modelo del motor, siendo de 1.5 Amperios para los AX12, 2.4 Amperios para los RX24F, 1.4 Amperios para el MX28T y 1 Amperio para el TOWER PRO SG5010. Con estas especificaciones técnicas y de acuerdo a la configuración de conexión, concluimos en el uso de una fuente de corriente continua con dos salidas variables de 1.2 a 15 Voltios y una potencia de 120 VA, tal como se muestra en la Figura 3.73.



Figura 3.73: Fuente de Alimentación Regulada
Fuente: Propia

La fuente de alimentación elegida nos permite suministrar energía directamente a los actuadores Dynamixel; sin embargo, en el caso del servomotor Tower Pro, es necesario valorar un circuito adicional para obtener una tensión de 5 Voltios, la misma que es adecuada para este accionamiento según su datasheet; es por esta razón que se vio por conveniente el uso del regulador de voltaje LM7805 (Figura 3.74), dispositivo capaz de entregar una tensión de salida de 5V a partir de una entrada determinada, siendo en este caso 12V.

LM7805 PINOUT DIAGRAM

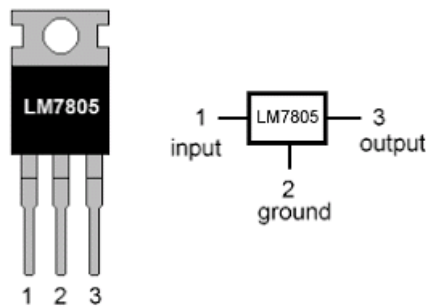


Figura 3.74: Regulador de Tensión LM7805
Fuente: "Componentes Útiles, LM7805", 2016

Para asegurar un buen funcionamiento del dispositivo mencionado en el párrafo anterior, tomamos en cuenta la disposición y conexión mostrada en el circuito de la Figura 3.75, donde es posible observar el uso de un diodo, el cual es opcional y cuya finalidad es la protección del regulador contra alguna regresión de voltaje o corto circuito.

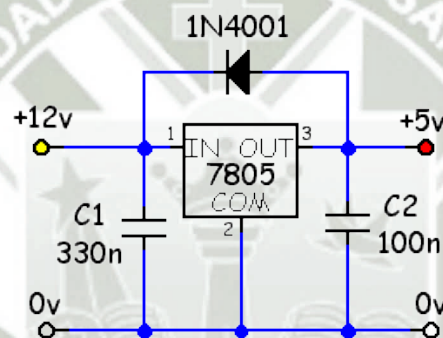


Figura 3.75: Circuito de Conexión LM7805
Fuente: "Fuente de 12V a 5V (7805)", Optimus Tronic, 2007

3.2.1.3. Procesador:

El procesador constituye el cerebro del proyecto, puesto que aquí se realizará toda la programación para el movimiento y control respectivo del brazo robótico; por lo tanto su elección es trascendental, de tal manera que satisfaga las necesidades y podamos cumplir así el objetivo planteado.

Se propone dos posibilidades de procesador para la aplicación; siendo el primero un DSP o procesador digital de señales y como segunda opción, la tarjeta Raspberry Pi.

El DSP es un sistema digital basado en un microprocesador que está específicamente diseñado para realizar procesamiento de datos en tiempo real. Entre los elementos básicos que lo componen tenemos: conversor analógico/digital a la entrada, conversor digital/analógico a la salida, memoria de datos, memoria de programa, multiplicadores y acumuladores, unidad Aritmético – Lógica y PWM. Así mismo, su lenguaje de

programación puede ser tanto en ensamblador como en C y su costo oscila entre 350 y 450 dólares. (Fig. 3.76)

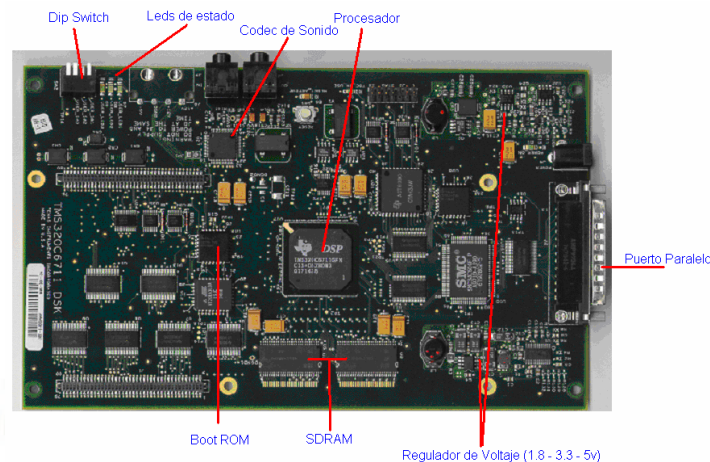


Figura 3.76: Procesador Digital de Señales
Fuente: "Digital Signal Processing", Alberto Cordero, 2009

La Raspberry Pi es un ordenador de placa única, cuyo tamaño es igual a la de una tarjeta de crédito. Posee varios puertos y entradas USB, Ethernet y salida HDMI, los mismos que permiten conectar distintos periféricos como teclado, mouse y pantalla LCD al igual que un ordenador común. Sin embargo; son los pines digitales GPIO los que diferencian a la Raspberry de un PC clásico, los mismos que se tratan de entradas y salidas de propósito general que hacen posible la interacción del dispositivo con el exterior. Este procesador admite varios sistemas operativos, la mayoría de ellos basados en Linux, y es posible programarlo en lenguaje Python. Su precio es bastante accesible pues oscila entre los 150 y 250 soles. (Fig. 3.77)



Figura 3.77: Raspberry Pi
Fuente: "Raspberry Pi, Reportajes", Alberto Castro, 2014

Dadas las características de ambos procesadores podemos notar que los dos dispositivos poseen especificaciones útiles para la aplicación de esta tesis, siendo el DSP más avanzado; sin embargo después de un análisis se concluyó que la tarjeta Raspberry Pi tiene la capacidad de poder llevar a cabo el control del prototipo y su precio es considerablemente menor en comparación a un procesador digital de señales; por esta

razón es que se optó por elegir una Raspberry Pi 2 model B como cerebro del proyecto. (Fig. 3.78)



Figura 3.78: Raspberry Pi2 Model B
Fuente: Propia

Habiendo elegido el procesador del proyecto es importante determinar la comunicación del mismo con los actuadores, la misma que se realizará mediante sus pines GPIO configurados como puerto serial, cuyo canal es full dúplex a diferencia de los motores Dynamixel que son half dúplex, por ello es necesario prever la compatibilidad entre ambos, la misma que será resuelta mediante buffer triestado 74ls241 para los servomotores AX12 Y MX28T y el circuito integrado MAX485 para los servomotores RX24F. Los circuitos se muestran en la figura 3.79.

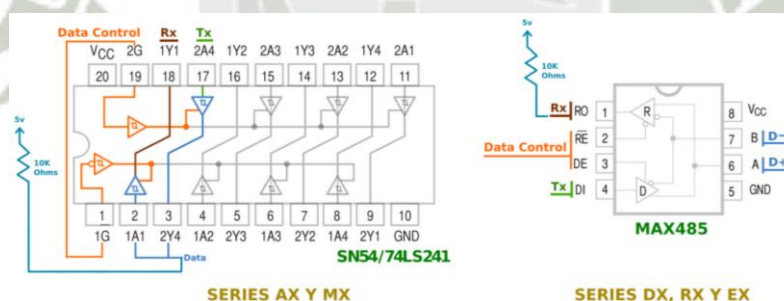


Figura 3.79: Circuito de conexión 74ls241 y MAX485
Fuente: "Savage Electronics", Josue Alejandro Savage, 2011

3.2.1.4. Pantalla LCD:

Habiendo elegido el procesador a utilizar correspondía ahora decidir que tipo pantalla LCD se iba a usar para visualizar los programas y operaciones de nuestro proyecto. En primera instancia teníamos como requerimientos que tuviera una buena resolución y un tamaño adecuado que nos permita poder observar cómodamente a una distancia máxima de 2 metros, por lo cual vimos por conveniente una resolución de 1366x768 y un tamaño de 10.1". Dadas estas características vimos en el mercado que una pantalla que satisfacía nuestras necesidades era LG Display LP101WH4-SLAF (Fig. 3.80).



Figura 3.80: Pantalla LCD LG Display LP101WH4-SLAF

Fuente: Propia

Además, se hacía necesario también adquirir un convertor VSLC Display (Fig. 3.81), para nuestro caso decidimos optar por uno simple ya que no requeríamos mayores especificaciones.



1.Power Supply(5~24V)	2.HDMI	3.VGA Connector	4.AV1
5.AV1+AV2	6.Keyboard	7.TTL Signal 50Pin	8.LVDS Signal 1/2 ch 6bit 8bit
9.12V Backlight	10.Inverter Connector		

Figura 3.81: Convertor VSLC Display con sus partes indicadas

Fuente: Propia

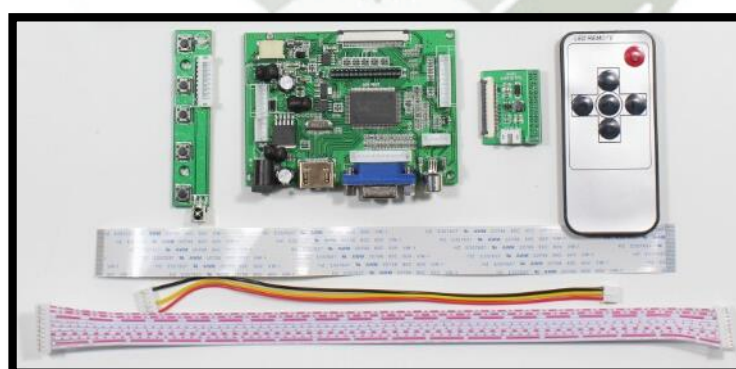


Figura 3.82: Elementos para conexión con la pantalla LCD

Fuente: Propia

De esta manera se definió el tipo de pantalla LCD y sus accesorios necesarios para su correcta operación.

3.2.1.5. Dispositivo para Visión Artificial:

Dado que tenemos como objetivo hacer uso de visión artificial para la teleoperación del brazo robótico, es que se debe elegir un equipo con características adecuadas; para ello, se indagó y se decidió finalmente el uso de un Kinect XBOX 360. (Figura 3.83)

El Kinect es un dispositivo que fue pensado inicialmente como un simple controlador de juego, que posee componentes como un sensor de profundidad, cámara RGB, array de micrófonos y sensor de infrarrojos, que lo hacen capaz de capturar el esqueleto humano, reconocerlo y posicionarlo en un plano.

Con el paso del tiempo este componente ha posibilitado a que los desarrolladores de software hagan uso de él para programar otra serie de aplicativos, gracias a la gran información que este dispositivo captura, siendo esta característica indispensable para el desarrollo de la presente tesis.



Figura 3.83: Kinect XBOX 360

Fuente: Propia

3.2.1.6. Cámara de Apoyo:

Conociéndose de antemano que nuestro manipulador será teleoperado se hizo imprescindible adquirir una cámara adicional que sirva de apoyo al operador al entregarle visión de los movimientos del brazo robótico. Por lo tanto, fue necesario encontrar una cámara que tuviera buena fluidez además de una resolución comparable con la pantalla LCD; basados en esto escogimos la cámara HD Pro Webcam c920 de la marca Logitech (Fig. 3.84) que como principal característica destaca su alta definición Full HD de 1080p que permite obtener un video fluido, imágenes nítidas y colores ricos.



Figura 3.84: Cámara HD Pro Webcam c920 - Logitech
Fuente: Propia

3.2.1.7. Sensor de Fuerza:

El manipulador robótico posee un gripper como efector final, el mismo que le permite sujetar un tubo de ensayo y transportarlo de un punto a otro. Esta operación implica la necesidad de conocer la cantidad de fuerza que debe ejercer la garra sobre el instrumento de laboratorio; es decir el nivel de presión; por este motivo, se vio por conveniente incluir un componente adicional en el proyecto, un sensor de fuerza.

Para llevar a cabo una elección adecuada del sensor, es necesario tomar en cuenta algunos requerimientos tales como: la aplicación, el tamaño y forma del dispositivo (encapsulado) y el sistema de control.

En cuanto a la aplicación se hizo un análisis de manera empírica con respecto a la fuerza máxima que requiere el gripper para obtener un agarre efectivo. En el resultado de algunas pruebas observamos que con 450gf aproximadamente se conseguía una sujeción adecuada del tubo de ensayo, de esta manera se obtuvo el primer parámetro para la elección del sensor.

Para el encapsulado del sensor se tomó en consideración que el área de contacto sea aproximadamente la misma que de la garra, siendo esta de 15mm x 15 mm, y además que su cableado se puede acoplar fácilmente al manipulador.

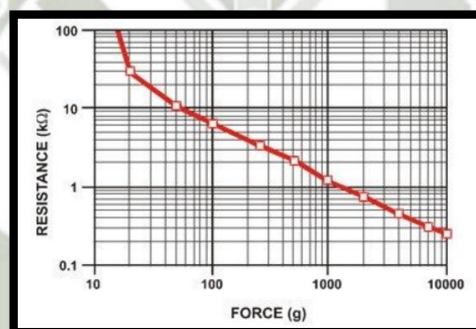
Y por último refiriéndonos al sistema de control, la selección se debe ajustar a los requerimientos del control del proceso, entonces dependerá del control la selección de la salida del sensor. Conociendo que nuestro procesador admite únicamente salidas y entradas digitales, nos veremos obligados a utilizar un conversor análogo digital.

Basados en estos 3 puntos vimos por conveniente elegir el sensor resistivo de fuerza MF01 (Fig. 3.85).


Figura 3.85: Sensor de fuerza MF01

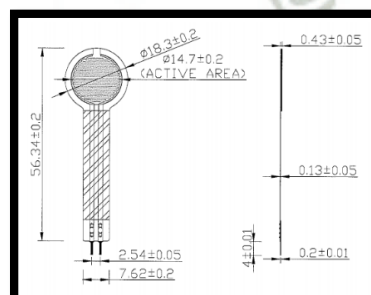
Fuente: "Sensor de Fuerza o Presión MF01", Carlos Nares, 2015

Referido al primer requisito que indica una fuerza máxima de agarre de 450gf, encontramos que este sensor tiene un comportamiento basado en una gráfica de resistencia vs fuerza (Fig. 3.86), donde nos muestra el valor de resistencia con el que se debe realizar un divisor de tensión y medir así el valor de fuerza requerido, siendo en nuestro caso de 3K Ω aproximadamente.


Figura 3.86: Gráfica de Resistencia (K Ω) vs Fuerza(g)

Fuente: "Sensor de Fuerza o Presión MF01", Carlos Nares, 2015

Tomando en consideración el segundo requerimiento observamos que las medidas del sensor son de 18.3 ± 0.2 de diámetro y 14.7 ± 0.2 de área activa (Fig. 3.87), además el cableado es flexible y se adapta fácilmente al manipulador; cumpliendo con lo establecido anteriormente.


Figura 3.87: Medidas del sensor de fuerza MF01

Fuente: "Sensor de Fuerza o Presión MF01", Carlos Nares, 2015

Como último requerimiento tenemos el sistema de control, el mismo que nos obliga a usar un conversor analógico digital; llevándonos a elegir el mcp3202 (Fig. 3.88) por tres razones principales; que su precio es económico y accesible, que tiene 12 bits de resolución y se alimenta con 3.3V lo que lo hace adecuado para el procesador y, por último, que posee dos canales analógicos, proporcionándonos de esta manera un canal adicional para la lectura del servomotor Tower Pro SG5010, ya que este no tiene integrada la opción de leer su posición como ocurre en el caso de los servomotores Dynamixel. Sus datos técnicos se encuentran detallados en el anexo C.

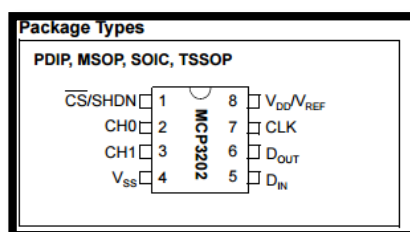


Figura 3.88: ADC MCP3202

Fuente: "Datasheet Mcp3202", MICROCHIP, 2006

Finalmente debido a que se tiene una fuente de alimentación regulada a 12V y el sensor MF01 utiliza 5V de tensión de entrada, optamos por usar el regulador de tensión LM7805 descrito precedentemente (Fig. 3.89).

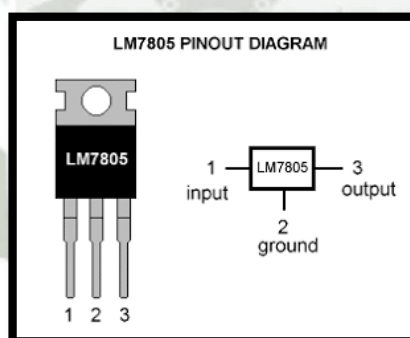


Figura 3.89: Regulador de tensión Lm7805

Fuente: "Componentes Útiles, LM7805", 2016

3.2.1.8. Supresor de picos y Estabilizador:

Durante este proceso de elección vimos por conveniente también elegir elementos que asegurarán la operación ante agentes externos como por ejemplo descargas eléctricas o variaciones de tensión, para tal caso es que optamos por incluir en nuestro proyecto los siguientes componentes:

-Supresor de Picos: tiene un parecido a una extensión con un switch, sirve para que en el caso que haya una descarga eléctrica este interrumpa la continuidad de energía

mediante un térmico o fusible. Evitando que salte o se queme los plomos de la casa o ambiente.



Figura 3.90: Supresor de picos de 6 salidas
Fuente: Propia

-Estabilizador: Un regulador de tensión o regulador de voltaje conocido comúnmente como estabilizador es un dispositivo electrónico diseñado para mantener un nivel de tensión constante, al igual que el supresor de picos también cuentan con un térmico o fusible.



Figura 3.91: Estabilizador de 8 salidas
Fuente: Propia

3.2.2. Diseño Preliminar del Circuito:

Para probar el funcionamiento de todos los componentes electrónicos en conjunto y mantener un orden visual en la circuitería, se consideró conveniente crear un diseño de la placa electrónica con el circuito final a utilizar; razón por la cual se realizó inicialmente en una hoja de papel (Fig. 3.92), teniendo en cuenta la representación gráfica de cada uno de los dispositivos con sus respectivos pines de conexión.

En el lado izquierdo del esquema se puede observar los actuadores Dynamixel en su totalidad alimentados por una fuente de 12V ubicada en la esquina superior derecha; así mismo, en cuanto a los pines de transmisión y recepción de datos apreciamos que se utiliza el integrado 74LS241 para los servomotores AX12 y MX28T y el integrado MAX485 para los RX24F con el fin de lograr la transferencia de paquetes entre los actuadores y la tarjeta Raspberry a través de su puerto serial (TX GPIO 8, Rx GPIO 10 y control GPIO 12, GPIO 18) como ya se explicó anteriormente.

Con respecto al motor Tower Pro SG5010, se emplea la misma fuente de poder con la variación del uso de un regulador de tensión, 7805, cuya finalidad consiste en proporcionar 5V para energizar el servomotor. El movimiento de este actuador se logra directamente a través del GPIO 11 de la Raspberry configurado por software como PWM; sin embargo, para la lectura de la posición se hace hincapié que a diferencia de los motores Dynamixel, el Tower Pro no posee dicha función integrada, por lo que es necesario adquirir esta información a través de una conexión adicional desde su potenciómetro interno hasta el procesador, por medio de uno de los canales del conversor analógico digital MCP3202.

Finalmente, vemos como último componente al sensor de fuerza MF01, el mismo que está dispuesto en un divisor de tensión junto a una resistencia de 3kΩ y es alimentado por el regulador de tensión 7805; además se puede apreciar que la salida está conectada al segundo canal del MCP3202 para así obtener la recepción de la fuerza ejercida por el manipulador.

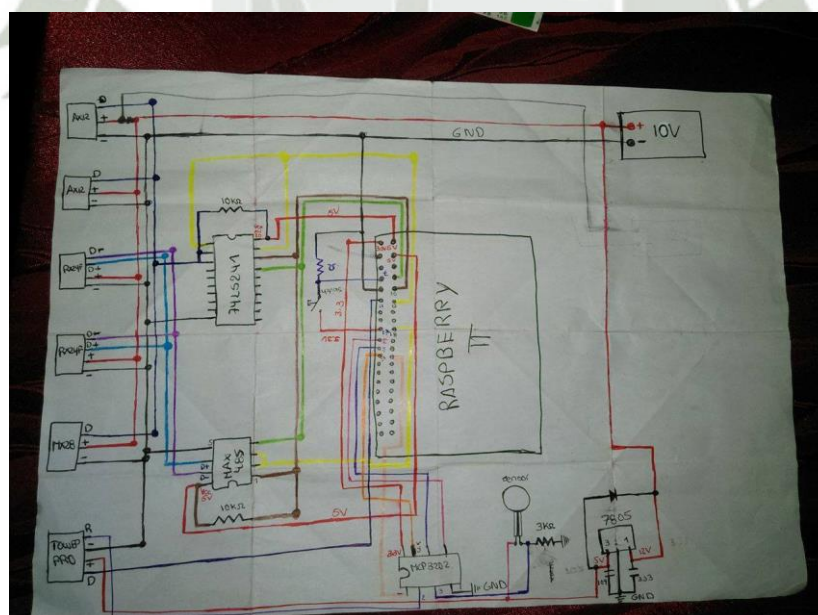


Figura 3.92: Diseño preliminar del circuito final

Fuente: Propia

3.3. Diseño y Construcción de Maqueta:

Habiendo realizado las pruebas necesarias comprobando así el buen funcionamiento del proyecto, procedemos a diseñar la maqueta que alojará todos los componentes usados en la aplicación de nuestro manipulador. Tendremos como requerimientos principales utilizar materiales livianos y de fácil montaje, mantener un orden visual en cada elemento; además de un acabado visiblemente estético al exterior.

Para los materiales en primer lugar vimos por conveniente utilizar melamina que es un tablero aglomerado de partículas, recubierto por ambos lados con películas decorativas impregnadas con resinas melamínicas, lo que le otorga una superficie totalmente cerrada, libre de poros, dura y resistente. Es más liviana que la madera común, no requiere muchos conocimientos de ebanistería¹ o bricolaje ni muchas herramientas para utilizarla, es capaz de soportar el calor, los líquidos agresivos de uso doméstico, no permite el desarrollo de microorganismos o parásitos y no requiere terminaciones adicionales. En segundo lugar, decidimos usar vidrio que es un material transparente que además de permitirnos tener visión de la operación del manipulador también lo protegerá contra el polvo lo cual es una característica a tener en cuenta en nuestro proyecto ya que nuestra aplicación se realizará en laboratorios farmacéuticos donde la limpieza e higiene es primordial.

Habiendo elegido que materiales se han de utilizar procedemos a diseñar nuestra maqueta basados en el espacio requerido para todos los componentes y analizando la mejor ubicación para cada elemento de tal manera que mantenga un orden interno y permite una correcta supervisión al momento de ejecutar nuestra aplicación. (Fig. 3.93)

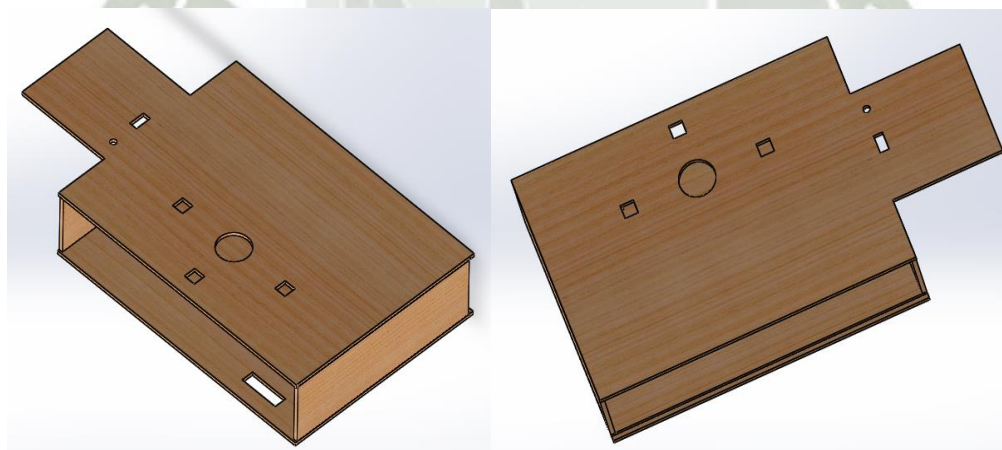


Figura 3.93: Diseño de la maqueta

Fuente: Propia

Ya con el diseño definido se utilizó los servicios de una empresa de corte de melamina en CNC, lo que nos permitió obtener un acabado fino, resaltando de esta forma la estética de nuestra maqueta. De igual modo se prestaron los servicios de una vidriería para adquirir la carcasa que recubrirá al brazo robótico, agregando un par de puertas para permitirnos realizar alguna modificación adicional si fuera requerida. (Fig. 3.94)



(a)



(b)

Figura 3.94: a) Parte superior hecha de melamina y b) Estructura de ensamblaje de la maqueta

Fuente: Propia

Para mejorar la estética y comodidad al transportar la maqueta se le agregó a la parte inferior de la base de la maqueta cuatro patas de aluminio que soportarán el peso de todos los componentes además que darán acceso al cable del supresor de picos que ira directamente al tomacorriente (Fig. 3.95); y en la parte superior de la base se hizo uso de algunos elementos de sujeción como topes triangulares diseñados especialmente para melamina, para fijar algunos componentes; también se agregó clips para la fijación de los cables lo cual permitirá una vía ordenada. (Fig. 3.96).


Figura 3.95: Patas de aluminio

Fuente: Propia

Figura 3.96: Topes triangulares para sujeción

Fuente: Propia

Seguidamente se agregó el manipulador robótico atornillándolo a la parte superior de la maqueta y asegurándonos de que sus conexiones tengan libertad de movimiento al

momento de manipularlo. Finalmente se colocó la carcasa de vidrio a modo de encapsular el brazo robótico. El proyecto montado en la maqueta se puede observar en la figura 3.97



Figura 3.97: Maqueta final

Fuente: Propia

CAPÍTULO IV: IMPLEMENTACIÓN

Implementar es una palabra que forma parte de nuestro lenguaje cotidiano y que entonces como tal la solemos emplear en diversos contextos y ámbitos. Tal es el caso del área de ingeniería en donde se emplea este término para designar la concreción de una individualización de tipo técnico. Buena parte de las implementaciones en este ámbito resultan en una nueva tecnología.

Toda invención en la vida se crea dos veces: la primera cuando se genera la idea visionando su forma y características; la segunda cuando se lleva a cabo y se transforma en algo tangible refiriéndose tanto al proceso de diseño e implementación. La palabra implementar permite expresar la acción de poner en práctica, medidas y métodos, para concretar alguna actividad, plan o misión en otras alternativas.

La realización de este capítulo abarca principalmente dos grandes temas, el primero referido a la construcción mecánica del manipulador robótico, y el segundo enfocado a la adición de elementos electrónicos y sus pruebas de funcionamiento. A continuación, se tocarán a detalle todas las herramientas, procesos y conceptos que se utilizaron para la realización del mismo.

4.1. Implementación Mecánica:

4.1.1. Herramientas usadas para la implementación mecánica del brazo robótico:

4.1.1.1. Herramientas Virtuales:

Entre las herramientas virtuales usadas para nuestro proyecto se encuentran:

- **Solidwork 2013:** Software que tuvo dos funciones muy importantes dentro de este proyecto: traspasar el bosquejo inicial a un diseño final mediante las herramientas propias del programa, para luego ser optimizado mediante la aplicación de estudios tanto estáticos como de fatiga sobre las piezas más críticas del manipulador robótico; y generar los archivos con formato. STL necesarios para la realización de la impresión 3D.
- **Repetier Host:** Software cuya finalidad es adquirir los archivos “. STL” provenientes del programa Solidwork, mediante comandos de posicionamiento y parámetros de impresión, dirige el proceso de conversión de un elemento virtual a una pieza real con un alto nivel de precisión mediante la impresora 3D.

4.1.1.2. Herramientas Mecánicas:

Entre las herramientas mecánicas usadas para nuestro proyecto se encuentran:

- **Impresora 3D:** es un dispositivo capaz de generar un objeto sólido desde cero mediante el incremento de material a partir de un diseño digital previo. La impresión 3D constituye parte de los procesos de fabricación aditiva y se ha convertido en una gran tendencia en el campo tecnológico que ha revolucionado los métodos de fabricación tradicionales. Los mismos que se caracterizan por ser sustractivos, pues generan una pieza con una forma o estructura determinada partiendo de un bloque de material sobre el que se realizan distintas operaciones eliminando una o varias capas hasta lograr el resultado que se quiere obtener. La impresión 3D permite también la creación de piezas de difícil manufactura ya sea por su forma o tamaño con una alta precisión y una baja pérdida de material. Todas estas ventajas nos llevaron a tomar la decisión de utilizar una impresora 3D para el proceso de fabricación del manipulador robótico. De entre tantas opciones se optó por adquirir un impresora 3D de marca PRUSA y modelo I3 tomándose en cuenta 3 factores principales para su elección: a) el tamaño máximo de las piezas, para lo cual las dimensiones del volumen de construcción (200mm x200mm x180mm) que ofrece esta impresora son ideales, b) la precisión requerida por el prototipo y que gracias a los dos extrusores (0.2mm y 0.4 mm) que llegan en la máquina se ve atendida, y c) el precio que comparado con otras alternativas en el mercado hace de esta impresora una opción accesible.
- **Herramientas para el ensamble:** Se utilizó llaves tipo Allen (1.5mm y 2.0mm) y desarmadores de precisión tanto tipo Estrella (3.0mm, 2.5mm y 1.5mm) como Planos (#1, #0, #00) para el montaje del manipulador robótico.
- **Herramientas de Ajuste Fino:** Se usó papel lijar N°600 y 400 para realizar algunos ajustes, a fin de lograr el encaje adecuado de las piezas.

4.1.1.3. Herramientas Eléctricas:

Entre las herramientas eléctricas usadas tenemos: una fuente de alimentación de corriente continua con dos salidas variables en paralelo de 1.2 a 15 voltios y una capacidad de entrega de corriente de hasta 10 amperios para la realización de pruebas en los servomotores, un multímetro digital para realizar todas las mediciones necesarias con el fin de asegurar el buen funcionamiento de los circuitos, una laptop para la creación y compilación de los programas de la placa Arduino Atmega, y este último para la configuración de movimiento de los actuadores.

4.1.2. Modelo Virtual del Brazo Robótico Armado:

Para realizar el modelo virtual del brazo robótico se procedió a ensamblar los eslabones uno a uno comenzando desde la Base 1 hasta la Garra como se observa en la Figura 4.1. Además, se especifica el orden y descripción de los eslabones en la tabla 4.1.

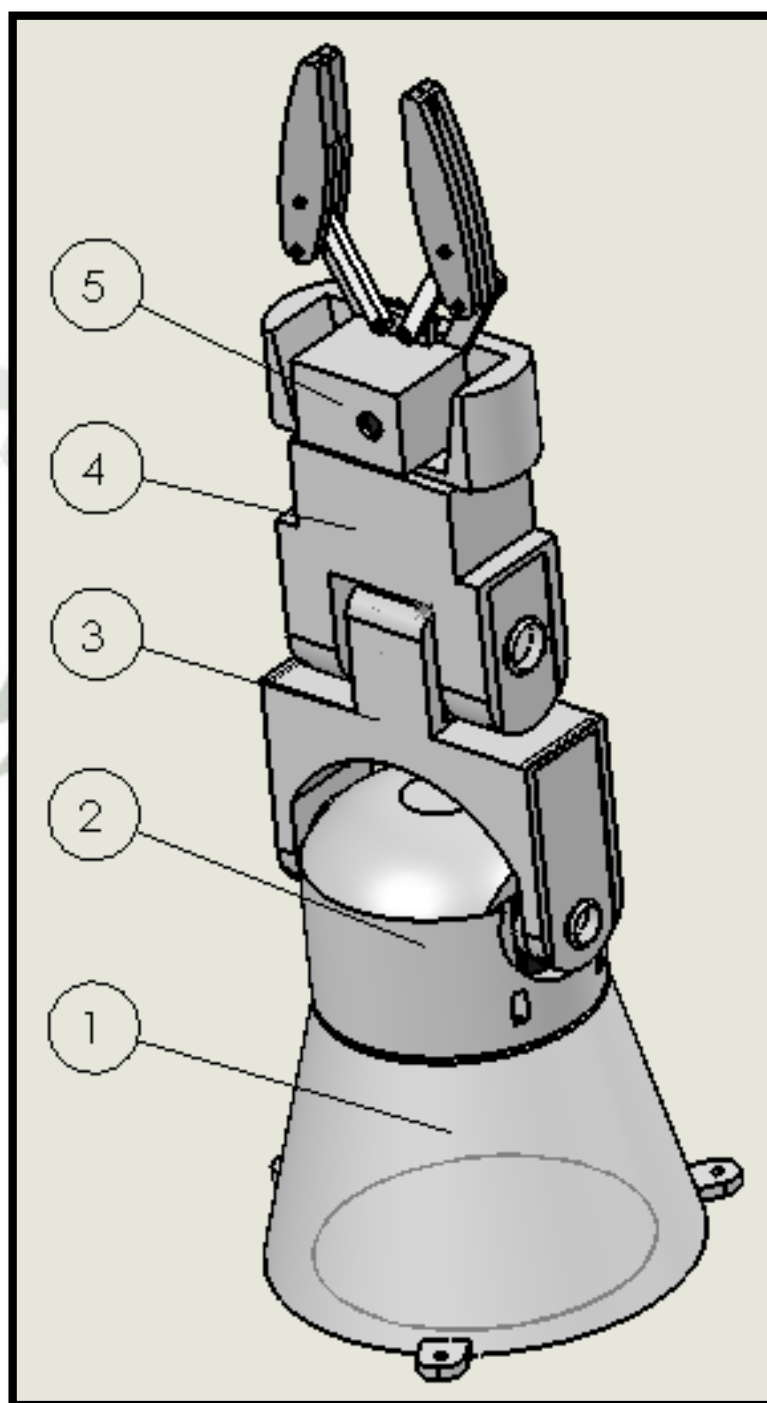


Figura 4.1: Brazo Robótico en subensambles

Fuente: Propia

Tabla 4.1: Orden y descripción de subensambles.

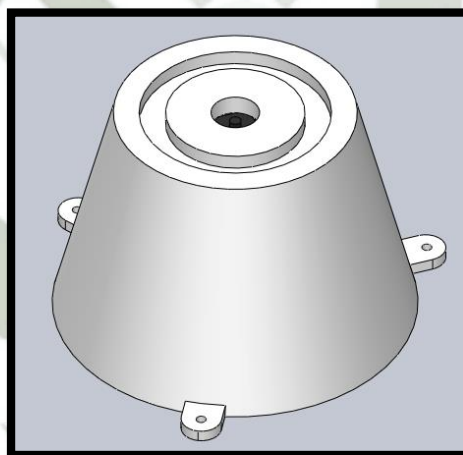
N° de elemento	N° de pieza	Descripción	Cantidad
1	Subensamblable 1	Base 1	1
2	Subensamblable 2	Base 2	1
3	Subensamblable 3	Eslabón 1	1
4	Subensamblable 4	Eslabón 2	1
5	Subensamblable 5	Garra	1

Fuente: Propia

4.1.3. Descripción de los Eslabones:

4.1.3.1. Subensamblable 1:

A este subensamblable lo denominamos Base 1, las dimensiones que le dan la robustez necesaria para soportar todo el peso del brazo robótico están descritas en la tabla 4.2, además cuenta con 3 puntos de unión a la mesa; todas estas características son necesarias para la sujeción y estabilidad del brazo robótico siendo esta la función principal de la pieza. (Figura 4.2)


Figura 4.2: Subensamblable 1: Base 1

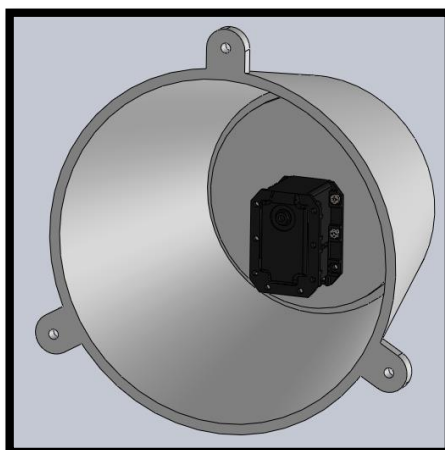
Fuente: Propia

Tabla 4.2: Subensamblable 1

Diámetro superior	120 mm
Diámetro inferior	180 mm
Altura	120 mm
Masa	480.19

Fuente: Propia

Cuenta con un servomotor MX-28 (Figura 4.3) de la nueva generación de los actuadores Robotics Dynamixel, cuyas características principales se dan en la Tabla 4.3.

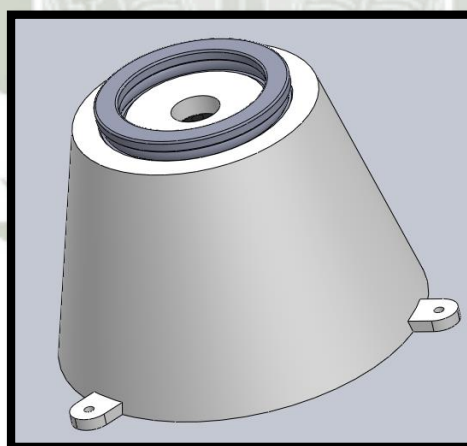

Figura 4.3: Motor MX-28

Fuente: Propia
Tabla 4.3: MX - 28

Dimensiones	35.6mm x 50.6mm x 35.5mm
Masa	72 gr
Tensión nominal	12 V
Torque	2.3 – 3.1 Nm
Corriente	1.3 – 1.7 A
Velocidad	50 – 67 rpm

Fuente: Propia

Además, en la parte superior va ubicado un rodamiento SKF 51114 sus dimensiones se muestran en la tabla 4.4; teniendo como función la unión con el subensamble 2 con un nivel despreciable de fricción. (Figura 4.4)


Figura 4.4: Rodamiento SKF 51114

Fuente: Propia
Tabla 4.4: Rodamiento SKF 51114

Diámetro interior	70 mm
Diámetro exterior	95 mm
Altura	18 mm
Masa	0.35 kg

Fuente: Propia

4.1.3.2. Subensamble 2:

Denominado Base 2, es el eslabón más pesado de todos y la razón de esto es porque dentro suyo resguarda dos piezas importantes de este diseño como son los dos servomotores RX-24f de la línea Robotics Dynamixel (tabla 4.5); motivo también por el cual su diseño tuvo que ser concebido como una pieza compacta ya que el movimiento generado en este elemento es el más crítico de todo el manipulador. Sus dimensiones generales se muestran en la tabla 4.6.

Además, cuenta con una canaleta circular en la parte inferior donde ira alojado el rodamiento SKF 51114 que es también compartido con la base 1, un círculo de unión entre los dos eslabones y dos canales por donde tendrán salida los cables de los motores. (Figura 4.5)

Tabla 4.5: RX – 24f

Dimensiones	35.6mm x 50.6mm x 35.5mm
Masa	67 gr
Tensión nominal	12 V
Torque	2.6 Nm
Corriente	2.4 A
Velocidad	126 rpm

Fuente: Propia

Tabla 4.6: Subensamble 2

Diámetro	120 mm
Altura	100 mm
Masa	613.95 gr

Fuente: Propia

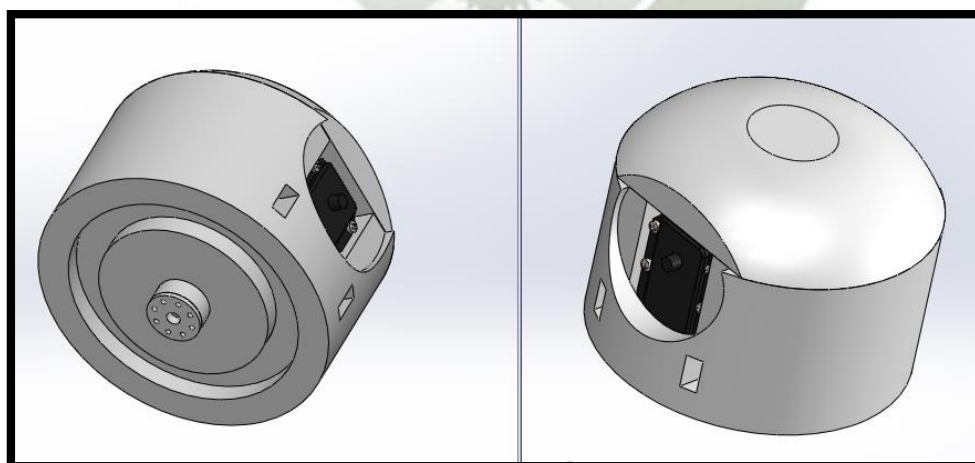


Figura 4.5: Subensamble 2: Base 2

Fuente: Propia

4.1.3.3. Subensamble 3:

Llamado también Eslabón 1, es una pieza relativamente ligera, cuenta con un servomotor Dynamixel AX-12 cuyos parámetros se detallan en la Tabla 4.7, la geometría de esta pieza permite alojar a este actuador y sus cables de alimentación y señal internamente lo que permite aligerar el componente y hacerlo más estético en el afán de simular un brazo humano (Figura 4.6). Sus dimensiones se observan en la tabla 4.8.

Tabla 4.7: AX - 12

Dimensiones	32mm * 50mm * 40mm
Masa	53.5 gr
Tensión nominal	12 V
Torque	1.53 Nm
Corriente	1.5 A
Velocidad	59 rpm

Fuente: Propia

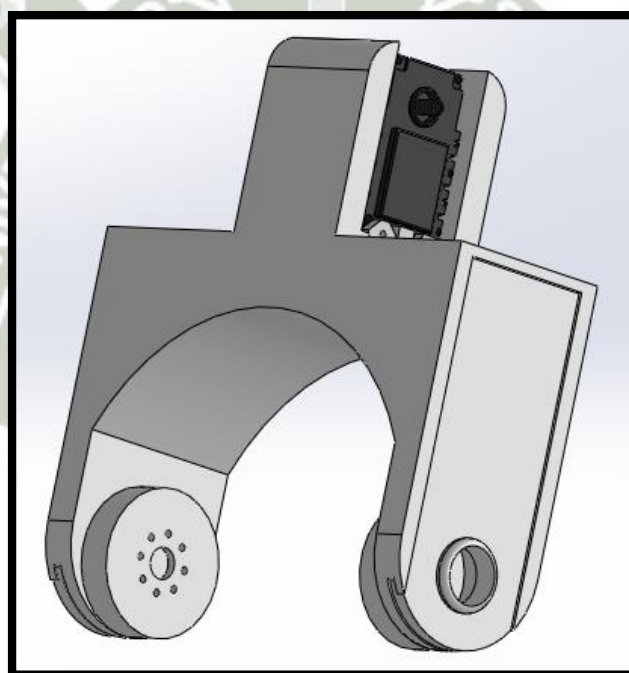


Figura 4.6: Subensamble 3: Eslabón 1

Fuente: Propia

Tabla 4.8: Subensamble 3

Largo	159 mm
Ancho	140 mm
Espesor	50 mm
Masa	332.56 gr

Fuente: Propia

4.1.3.4. Subensamble 4:

Esta pieza lleva de nombre Eslabón 2, sus propiedades físicas están descritas en la tabla 4.9, cuenta con un servomotor Tower Pro SG5010 cuyas características se ven en la tabla 4.10; al igual que el eslabón 1 es un componente ligero, compacto, estético y lo más importante tiene una ubicación óptima para el servomotor de manera que el espaciamento se ve bien aprovechado. (Figura 4.7)

Tabla 4.9: Subensamble 4

Largo	120 mm
Ancho	106 mm
Espesor	50 mm
Masa	310.26 gr

Fuente: Propia

Tabla 4.10: Tower Pro SG 5010

Dimensiones	40.6mm * 20.5mm * 38mm
Masa	47 gr
Tensión nominal	5 V
Torque	0.55 – 0.65 Nm
Velocidad	57 - 59 rpm

Fuente: Propia

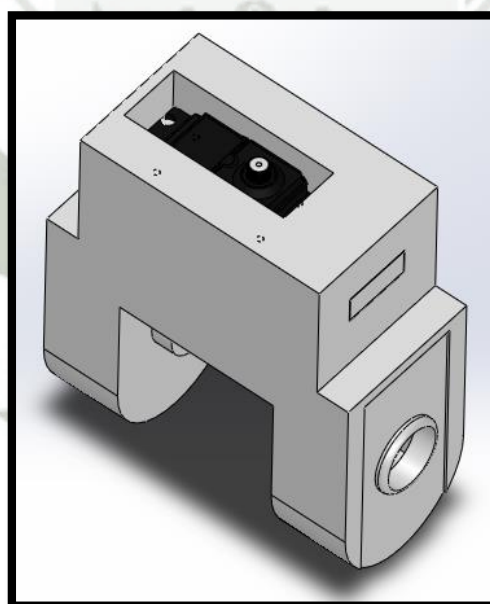


Figura 4.7: Subensamble 4: Eslabón 2

Fuente: Propia

4.1.3.5. Subensamble 5:

Es la última pieza ensamblada denominada también garra y tiene por función sujetar a presión el objeto para luego transportarlo de un punto a otro como se observa en la

figura 4.8; es el elemento con mayor cantidad de piezas, aunque todas de menor tamaño lo que la hace uno de los subensambles más ligeros. Las dimensiones del portagarra que es el componente más grande de este eslabón se observan en la tabla 4.11.

Tabla 4.11: Portagarra

Largo	45 mm
Ancho	122 mm
Espesor	45 mm
Masa garra total	265.45 gr

Fuente: Propia

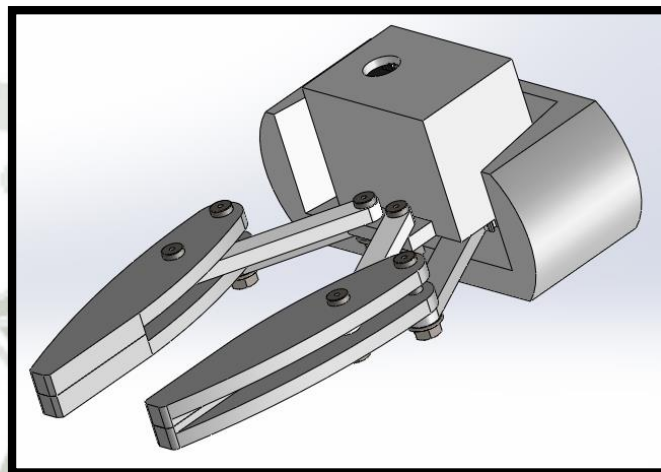


Figura 4.8: Subensamblado 5: Garra

Fuente: Propia

Cuenta con un servomotor Dynamixel AX-12 cuyas propiedades se observa en la tabla 4.7, que se encuentra encapsulado para su protección encima del portagarra y que le da el movimiento de apertura o cierre a los dedos del efector final. (Figura 4.9)

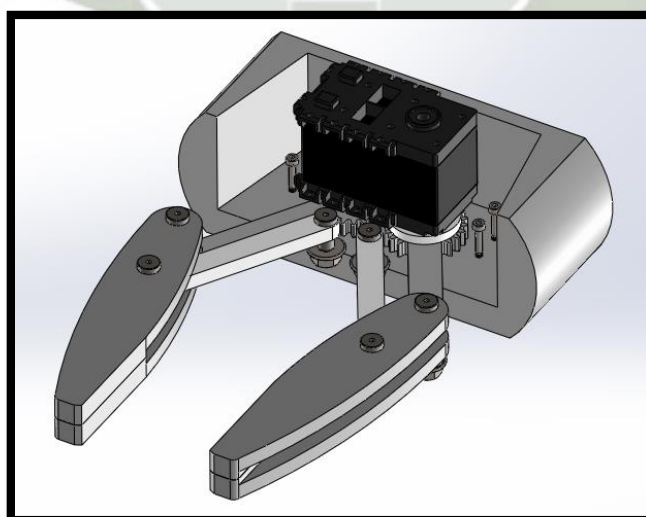


Figura 4.9: Motor AX-12

Fuente: Propia

4.1.4. Impresión 3D:

El proceso de fabricación del prototipo en cuestión mediante la tecnología de impresión 3D, inicia con la creación de archivos STL generados por el SolidWork a partir del diseño virtual de cada una de las piezas constituyentes del brazo robótico, los mismos que son luego importados por el Software Repetier Host para llevar a cabo el procedimiento de manufactura propiamente dicho.

Repetier Host es un soporte lógico que se inicia desde su respectivo icono de acceso directo ubicado en el escritorio de nuestro computador, para posteriormente establecer algunos parámetros de configuración de la máquina, con el propósito de asegurar su buen funcionamiento sin que se produzca algún conflicto o falla en el mismo. Dentro de dichos parámetros tenemos velocidad de desplazamiento, velocidad de avance en ejes, velocidad de extrusión, velocidad de retracción, número de extrusores, temperaturas máximas, entre otros; donde es imprescindible tomar especial atención a características tales como temperaturas de precalentamiento, siendo de 200°C en el caso del Extrusor y 90°C para la plataforma (Fig. 4.10), y diámetro del extrusor, que en nuestro caso puede ser de 0.2mm o 0.4mm dependiendo de la pieza (Fig 4.11).

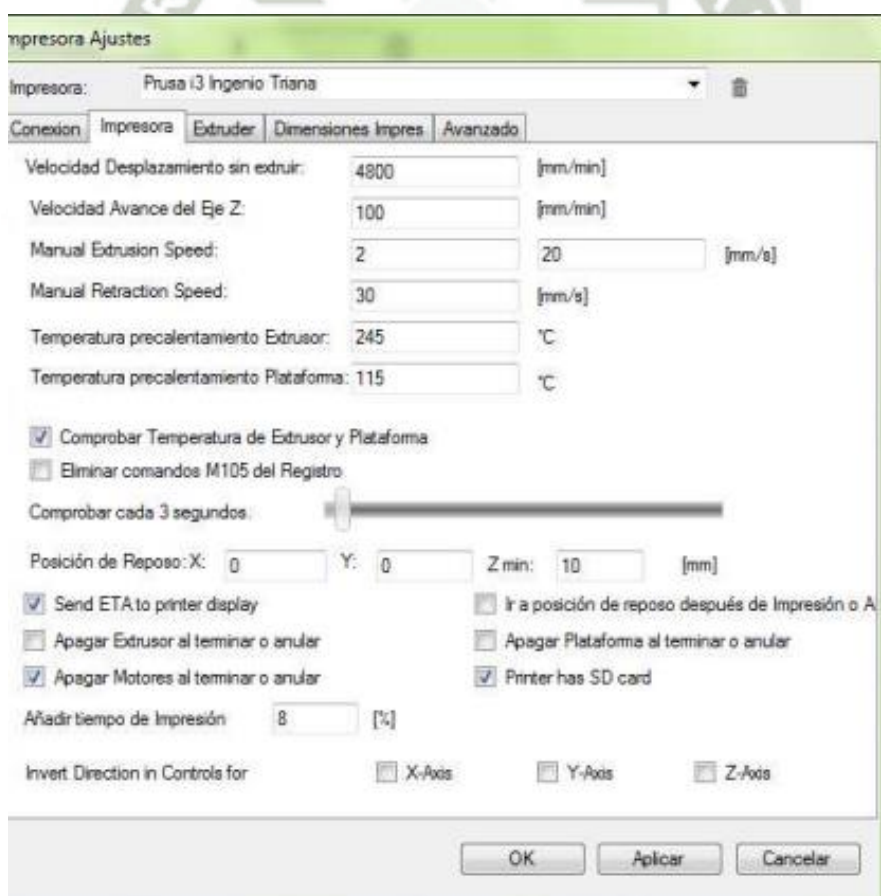


Figura 4.10: Parámetros de configuración de la impresora

Fuente: Software Repetier Host

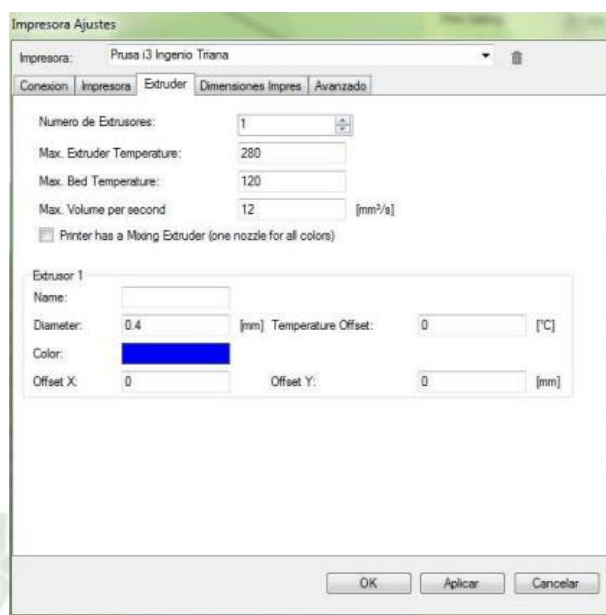


Figura 4.11: Parámetros de configuración del extrusor

Fuente: Software Repetier Host

Una vez determinadas las medidas necesarias, pulsamos el botón Conectar  para establecer comunicación entre la PC y la herramienta 3D a través de su controlador. De esta manera, observaremos que el botón cambia de color , siendo este un indicativo de una correcta conexión.

Posteriormente, cargamos en el software la pieza a imprimir, utilizando la opción “importar” desplegada de la pestaña “Archivo” en la barra de Menús, la misma que permitirá visualizar el objeto seleccionado en el cuadro de impresión (Figura 4.12).

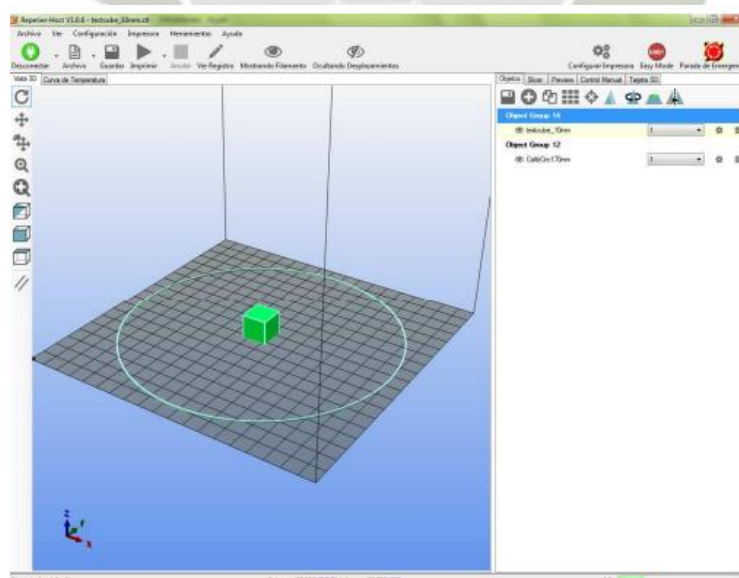


Figura 4.12: Visualización del elemento a imprimir

Fuente: Software Repetier Host

A continuación, nos ubicamos en la barra de herramientas lateral derecha del programa, donde disponemos de varias opciones de ajuste de la máquina. La ventana “Objetos”, nos permite posicionar la pieza en el centro de la plataforma y en caso sea necesario, la rotamos modificando los ángulos con respecto a cada eje para facilitar el proceso. (Figura 4.13)



Figura 4.13: Pestaña Objetos
Fuente: Software Repetier Host

La pestaña “Slicer” constituye la ventana de configuración principal, pues es aquí en donde el Repetier Host nos permite elegir entre dos programas para el laminado horizontal de la pieza, CuraEngine y Slic3r, siendo el primero un software básico que posibilita delimitar parámetros generales, a diferencia del segundo en donde podemos configurar características de impresión más avanzadas para objetos que necesiten mayor cuidado por su compleja geometría (Figura 4.14)

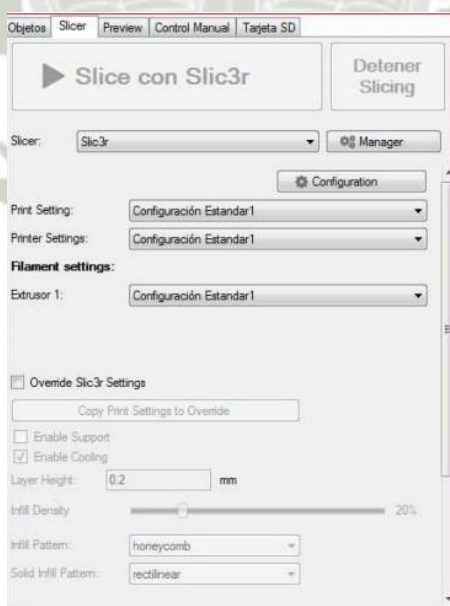


Figura 4.14: Pestaña Slicer
Fuente: Software Repetier Host

Para el desarrollo de la presente tesis, se vio por conveniente el uso del soporte lógico CuraEngine, como alternativa suficiente para llevar a cabo la fabricación de cada uno de los eslabones que componen al manipulador robótico, puesto que consideramos que su estructura física no resulta complicada; sin embargo, es necesario definir los valores correspondientes a ciertos parámetros, con el propósito de lograr un proceso de manufactura adecuado, para ello pulsamos el botón “Configuración” y procedemos a ingresar los siguientes datos: número de capas exteriores (3), número de capas superiores (3), número de capas inferiores (3), porcentaje de relleno interior (50%) y velocidades. Seguidamente presionamos “Slice con CuraEngine” para generar la trayectoria a seguir por el extrusor durante la impresión.

Luego abrimos la ventana “Control Manual” ubicada en la misma barra de herramientas, que nos permite manejar la impresora 3D por comandos o con el cuadro interactivo mediante pulsadores y flechas que indican la dirección de movimiento o temperatura (Fig. 4.15). Aquí activamos el precalentamiento tanto del extrusor como de la plataforma, y a la vez realizamos la calibración física de la “cama” para asegurar la adhesión de la primera capa de material de la pieza a imprimir; así mismo, es fundamental activar el extrusor para que expulse una pequeña cantidad de material, siempre y cuando haya llegado al límite de la temperatura de precalentamiento, con la finalidad de comprobar que la boquilla del mismo no se encuentre obstruida, y tenga así un buen funcionamiento. Seguidamente, hacemos click en el botón imprimir para dar paso al funcionamiento real de la impresora 3D Prusa I3.

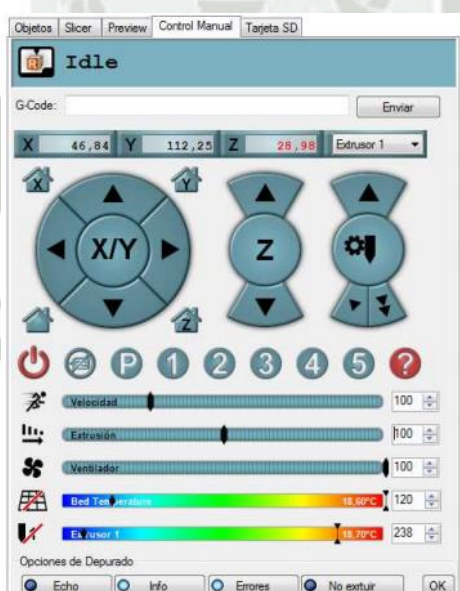


Figura 4.15: Pestaña Control Manual

Fuente: Software Repetier Host

Habiendo descrito el procedimiento repetitivo que se siguió para la fabricación de las piezas por tecnología de impresión 3D, nos enfocamos en definir detalles aún no tomados en cuenta. Tal como se especifica en el capítulo precedente, y considerando la máquina a utilizar, se determinó que el material a emplearse en la producción de cada

uno de los elementos es el filamento de plástico resistente al impacto, ABS de color blanco, el mismo que se adquiere por peso, usando así 2Kg.

Posteriormente, en cuanto a las piezas que constituyen el manipulador robótico, las clasificamos en dos grupos, siendo el primero compuesto por los elementos principales (Base 1, Base 2, Ovnitapa, Eslabón 1, Eslabón 2, Portagarra y Caja de motor), que se caracterizan por ser de mayor tamaño, motivo por el cual se usó el extrusor de 0.4mm para su impresión y el tiempo de demora de producción aproximado por cada objeto es de 24 horas (Fig. 4.16).



Figura 4.16: Impresión 3D de Portagarra
Fuente: Propia

El segundo grupo está conformado por 2 Engranajes, 2 Palitos, 4 Garritas, Soporte de Motor, 2 Tapas de Eslabon 1 y 2 Tapas de Eslabón 2, los mismos que tienen como características principales un tamaño considerablemente menor en comparación con el primer conjunto y la necesidad de una mayor precisión; es por ello que para su manufactura se utilizó el extrusor de 0.2mm de diámetro en la boquilla y el tiempo de demora de fabricación fue 1 hora aproximadamente por pieza (Fig. 4.17). Finalmente, durante el proceso de impresión, se pudo notar que es necesario considerar una tolerancia de 0.3mm en el diseño virtual de los objetos; para asegurar un adecuado ensamble del brazo robótico, puesto que esta tecnología de producción se basa en el calentamiento del material para fundirlo e ir adicionándolo capa por capa, el mismo que posteriormente es enfriado; por lo tanto, se deduce que el elemento pasa por procesos de dilatación y contracción.



Figura 4.17: Impresión 3D de Engranaje
Fuente: Propia

4.1.5. Pruebas individuales:

4.1.5.1. Pruebas Individuales de Motores:

En el capítulo de Diseño, específicamente en el apartado de elección de actuadores, se determinó el uso de servomotores Dynamixel y TowerPro como accionamientos para las articulaciones del brazo robótico. Dentro de ellos se seleccionaron los modelos Dynamixel AX-12, MX-28T, RX-24F y el modelo TowerPro SG5010, por tener las características adecuadas para esta aplicación. Una vez adquiridos los motores fueron puestos a prueba en vacío, para confirmar su buen funcionamiento, calcular sus límites de operación y ser luego acoplados en cada eslabón respectivamente.

- **TowerPro SG5010:** Actuador perteneciente a la cuarta articulación del manipulador robótico cuya función radica en darle movimiento rotativo de 0° a 180° al efector final. Para su puesta en marcha se hizo uso de una fuente de alimentación variable, regulada a 5VDC, siendo esta la tensión apropiada para su funcionamiento; así mismo se empleó un arduino MEGA 2560, encargado de enviar la señal PWM como data hacia el motor a través del pin 3; con este fin, es necesario la creación de un pequeño programa “towerpro” (ANEXO D), el mismo que utiliza la librería servo y en donde se especifican los pulsos mínimo y máximo en ms.

La disposición de los elementos y sus respectivas conexiones, se pueden observar en las figuras 4.18 y 4.19.

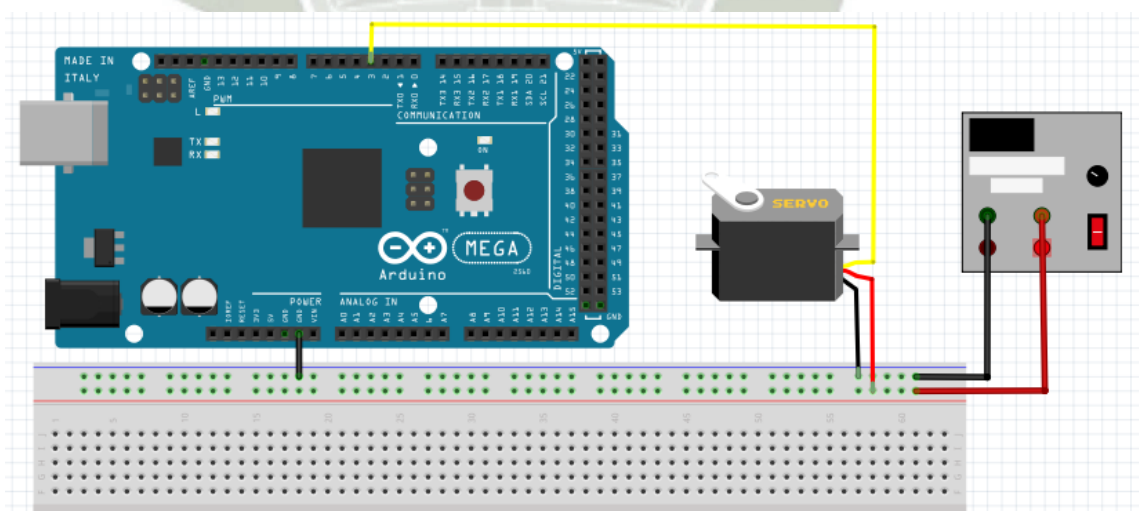


Figura 4.18: Conexión del motor TowerPRO SG5010 para puesta en funcionamiento

Fuente: Propia

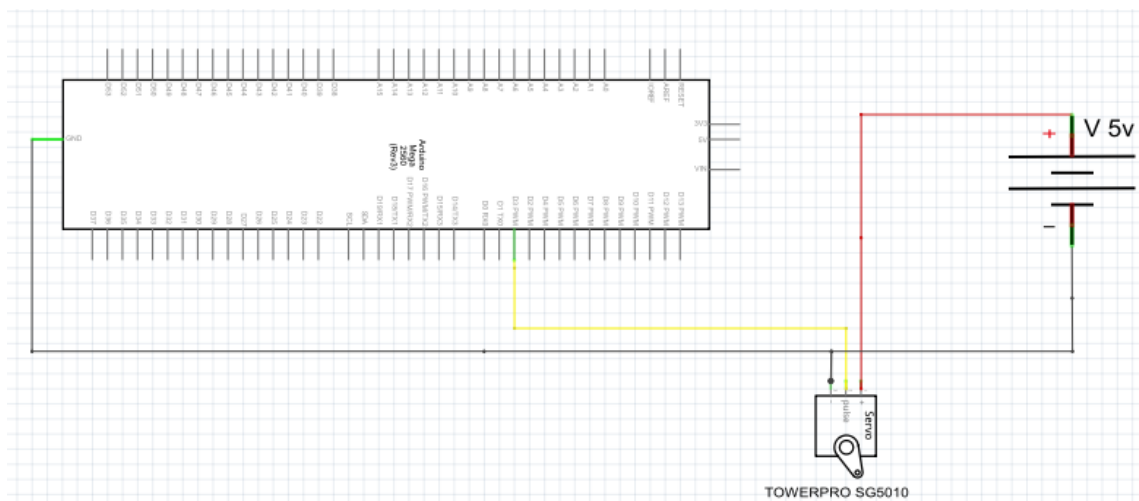


Figura 4.19: Esquemático eléctrico de conexión del motor TowerPRO SG5010

Fuente: Propia

La figura 4.20, nos muestra el ensayo experimental realizado.



Figura 4.20: Prueba de funcionamiento de motor TowerPRO SG5010

Fuente: Propia

Posteriormente, se procedió a calcular los límites de movimiento del servomotor acorde con nuestra aplicación, modificando los valores del ancho de pulso de manera iterativa hasta conseguir el objetivo deseado, determinando así que el valor de 545us corresponde a 0° y 2275us a 180°.

- **Dynamixel AX-12:** Se cuenta con dos accionamientos de este modelo, siendo el primero utilizado para dar movimiento de cierre y apertura a la garra del robot, y el segundo, para dar movimiento rotativo a la tercera articulación del brazo. Para la prueba de funcionamiento se conectaron ambos actuadores mediante una red daysi chain; además se utilizó una fuente de alimentación variable regulada a

12VDC, y se dispuso del arduino MEGA 2560, para transmitir la señal de datos a través del puerto serial 0 (Fig. 4.21 y 4.22).

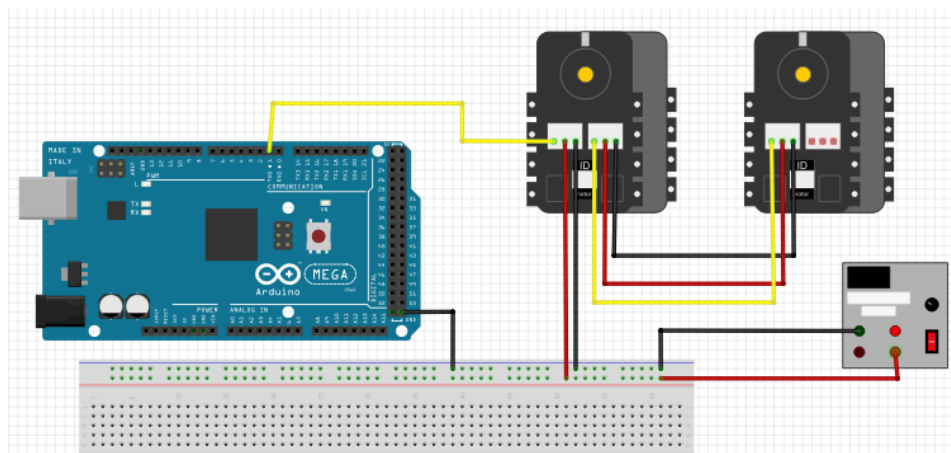


Figura 4.21: Conexión de actuadores Dynamixel AX-12 para puesta en funcionamiento
Fuente: Propia

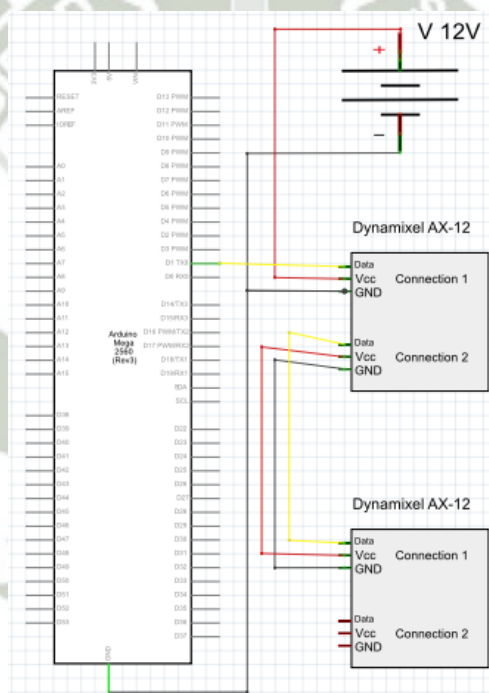


Figura 4.22: Esquemático eléctrico de conexión de motores Dynamixel AX-12
Fuente: Propia

En cuanto al software, inicialmente se realizó la configuración de los servomotores, cambiando el ID de uno de ellos a 2 y la velocidad de transmisión de ambos a 1Mbps, seguidamente se llevó a cabo un programa denominado “AX12” (Anexo D) que hace uso de la librería DynamixelSerial, la misma que permite leer y escribir data de forma sencilla.

En la figura 4.23, se puede observar la experimentación llevada a cabo.

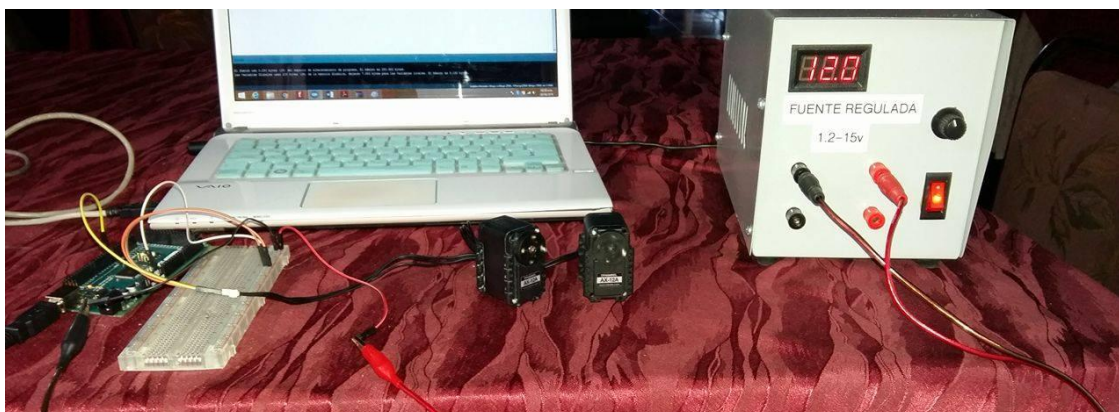


Figura 4.23: Prueba de funcionamiento de actuadores Dynamixel AX-12

Fuente: Propia

Para la definición de los límites de operación de ambos motores, se aplicó el código anteriormente mencionado, en el que se fue variando uno a uno los valores de resolución relacionados al movimiento angular de cada actuador. En el caso de la garra se obtuvo un intervalo de 10 a 170, que determina el rango de trabajo del efector final, mientras que para la tercera articulación se obtuvo un valor de 240 para 0° y 800 para 180°.

- **Dynamixel RX-24F:** Se hizo uso de dos actuadores de este tipo, a fin de generar rotación en la segunda articulación del brazo robótico. Del mismo modo que el caso previo, para otorgar movimiento a los motores, éstos fueron conectados mediante una red daisy chain; además se empleó una fuente de alimentación variable regulada a 12VDC, y se asignó la tarjeta arduino MEGA 2560, para transmitir la señal de datos a través del puerto serial 0 y el pin 2. Adicionalmente empleamos un circuito integrado MAX485, el mismo que nos permite adaptar una comunicación Full Duplex a Half Duplex. (Fig 4.24 y 4.25).

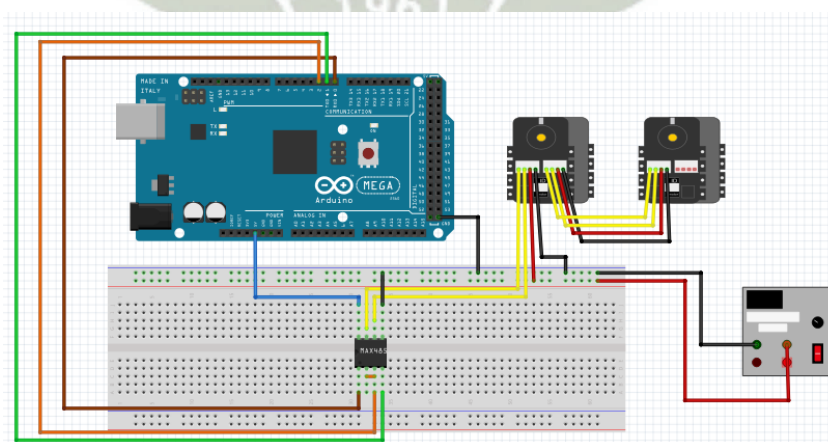


Figura 4.24: Conexión de actuadores Dynamixel RX-24F para puesta en funcionamiento

Fuente: Propia

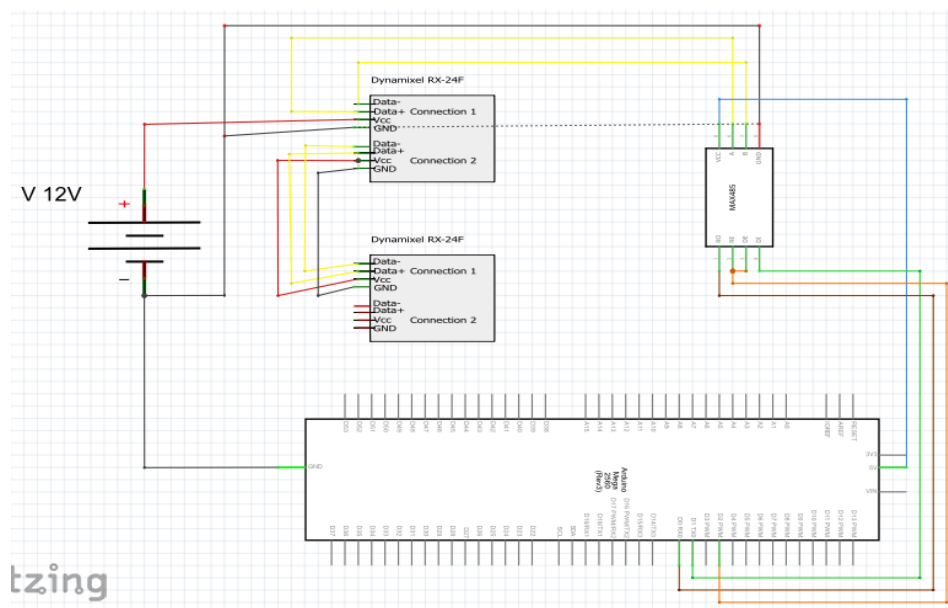


Figura 4.25: Esquemático eléctrico de conexión de motores Dynamixel RX-24F

Fuente: Propia

Haciendo referencia al software, se efectuó como primer paso, la configuración de los servomotores, cambiando su ID a 3 y 4 respectivamente; de igual manera se modificó la velocidad de transmisión de 57142 bps (default) a 1Mbps; posteriormente, se llevó a cabo un script denominado “RX-24F” (Anexo D) que hace uso de la librería DynamixelSerial para la lectura y escritura de datos.

La figura 4.26, nos muestra la prueba experimental ejecutada.

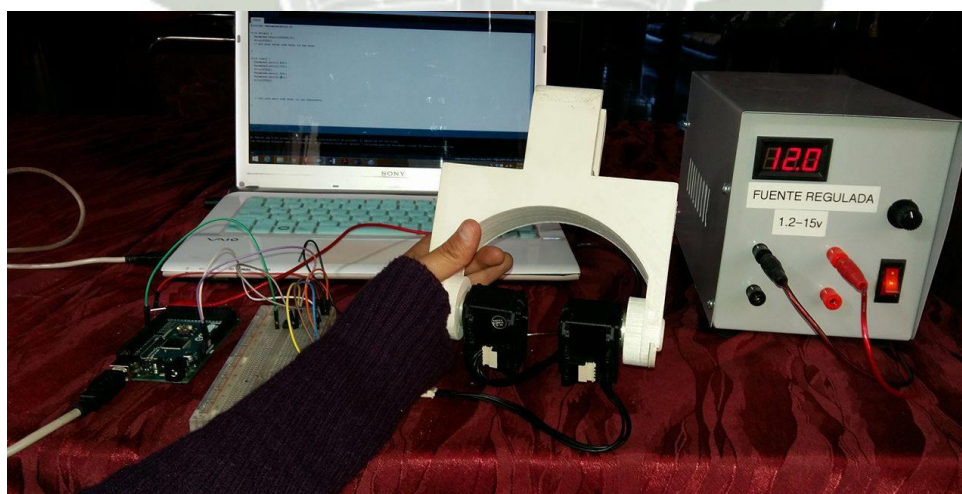


Figura 4.26: Prueba de funcionamiento de actuadores Dynamixel RX-24F

Fuente: Propia

Con respecto a la definición de los límites de funcionamiento en ambos motores, se empleó el programa ya creado, en el que se fue modificando uno a uno los valores

de resolución relacionados al movimiento angular de los accionamientos, teniendo en cuenta que, si uno de ellos tiene un giro horario, el otro debe imprimir un movimiento rotativo antihorario. De esta manera se logró el sincronismo en ambos actuadores, determinando un intervalo de resolución que va de 220 a 870 en el servomotor 3 y de 750 a 100 en el 4, lo que corresponde a una rotación de 0° a 180° en la articulación.

- **Dynamixel MX-28T:** Es utilizado en la primera articulación, cuya función radica en otorgar una rotación de 0° a 360° al manipulador robótico. Para la verificación de su operatividad, se hizo uso de una fuente de alimentación variable, regulada a 12VDC, siendo esta una tensión adecuada para su funcionamiento; así mismo se empleó un arduino MEGA 2560, encargado de enviar la señal de datos hacia el motor a través de su puerto serial 0 (Fig. 4.27 y 4.28).

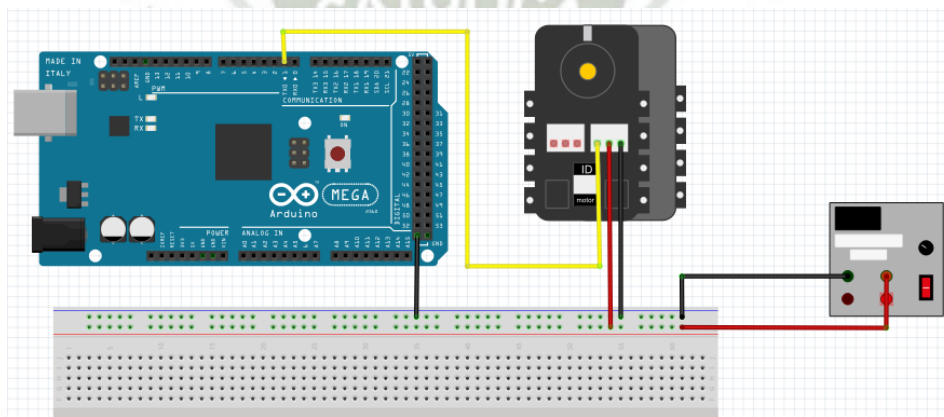


Figura 4.27: Conexión del motor Dynamixel MX-28T para puesta en funcionamiento
Fuente: Propia

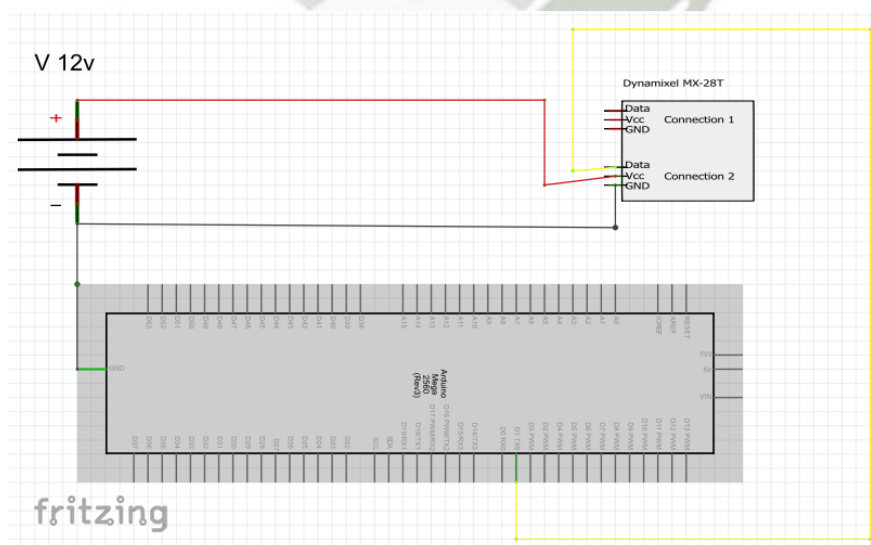


Figura 4.28: Esquemático eléctrico de conexión del motor Dynamixel MX-28T
Fuente: Propia

Referente al software, primero se realizó la configuración del servomotor, modificando su ID a 5 y su velocidad de transmisión de 57600 bps (default) a 1Mbps, consecutivamente se creó un script denominado “MX28T” (Anexo D) que al igual que los casos anteriores, incluye la librería Dynamixel Serial para leer y escribir data de forma simple.

En la figura 4.29, se puede observar el ensayo experimental para la puesta en marcha del actuador.

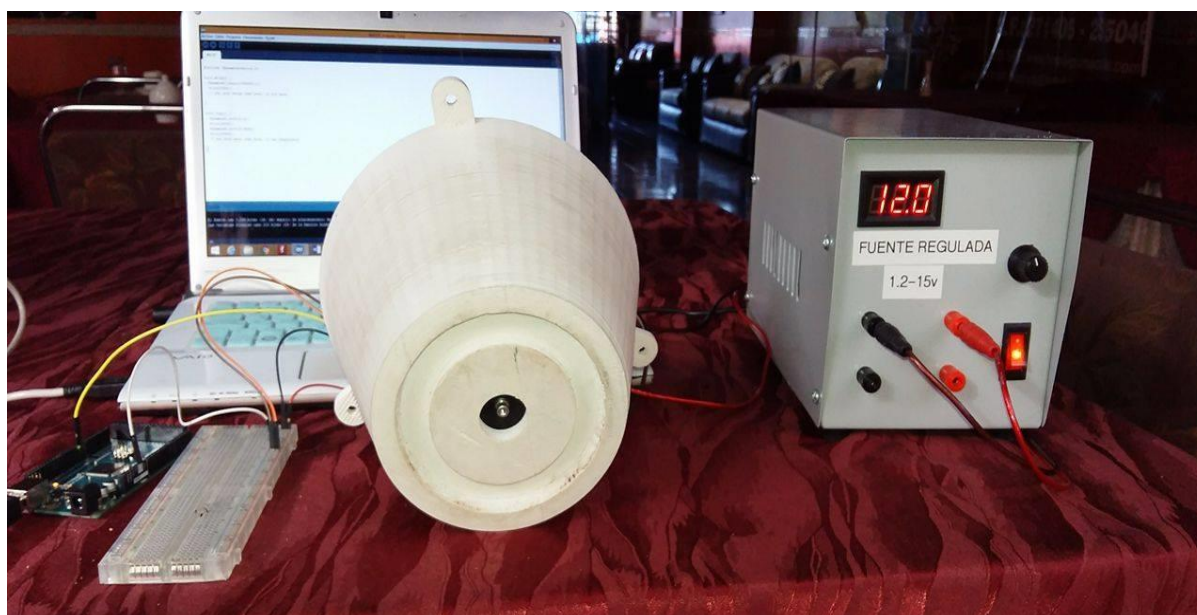


Figura 4.29: Prueba de funcionamiento de motor Dynamixel MX-28T

Fuente: Propia

Finalmente, se determinó que los límites de movimiento del accionamiento son los mismos que se especifican en su datasheet, teniendo así una resolución de 0 para 0° y 4095 para 360°.

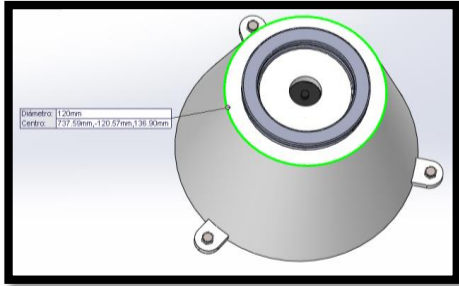
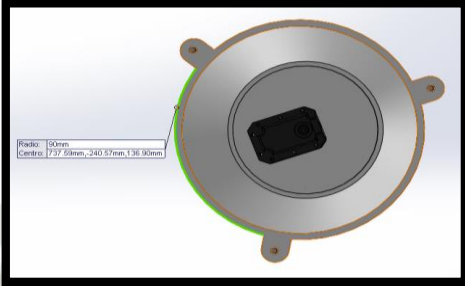
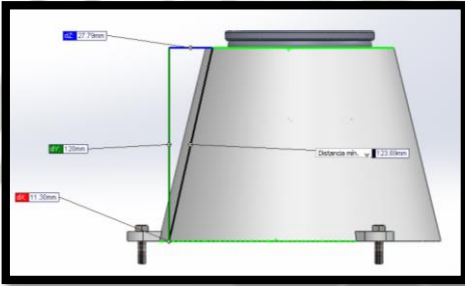
4.1.5.2. Pruebas Individuales de Medidas:

Realizada la impresión de los distintos eslabones se procedió a comparar las medidas generales de cada uno con sus diseños virtuales para corroborar el nivel de precisión en su fabricación.

- Subensamble 1:

Se tomó en cuenta para este eslabón el diámetro superior, el diámetro inferior y la altura; dando como resultado una alta precisión con un error máximo de ± 0.3 mm como se puede apreciar en la tabla 4.12.

Tabla 4.12: Comparación Subensamblado 1

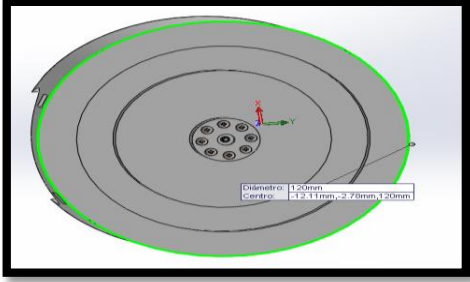
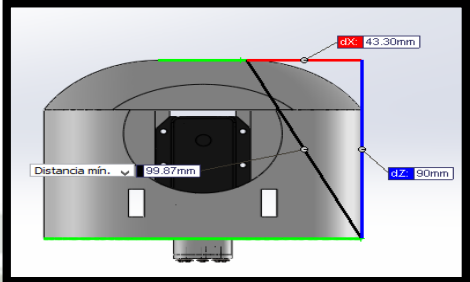
 Diámetro Superior: 120 mm	 Diámetro Superior: 119.6 mm
 Diámetro inferior: 180 mm	 Diámetro inferior: 180.3 mm
 Altura: 120 mm	 Altura: 119.8 mm

Fuente: Propia

- **Subensamblado 2:**

Para este eslabón se comparó el diámetro y la altura; dando como resultado una alta precisión con un error máximo de ± 0.8 mm como se puede apreciar en la tabla 4.13.

Tabla 4.13: Comparación Subensamble 2

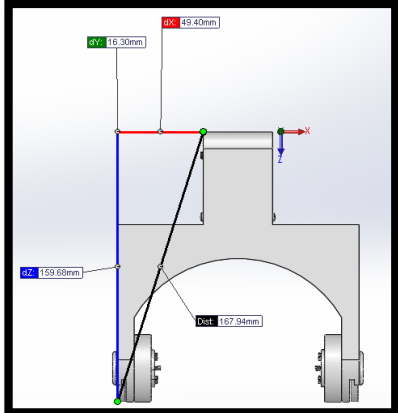
 Diámetro: 120 mm	 Diámetro: 120.2 mm
 Altura: 90 mm	 Altura: 90.8 mm

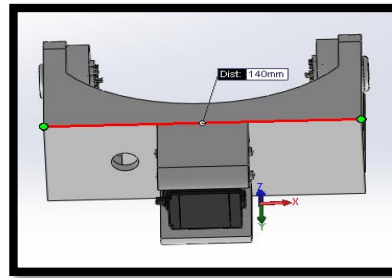
Fuente: Propia

- **Subensamble 3:**

Para este eslabón se comparó el largo, ancho y espesor; dando como resultado una alta precisión con un error máximo de ± 0.6 mm como se puede apreciar en la tabla 4.14.

Tabla 4.14: Comparación Subensamble 3

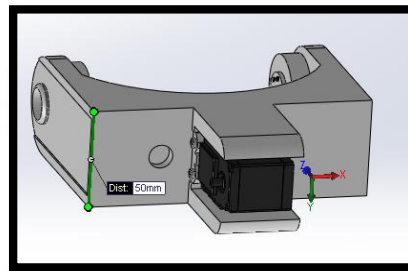
 Largo: 159.68 mm	 Largo: 159 mm
---	---



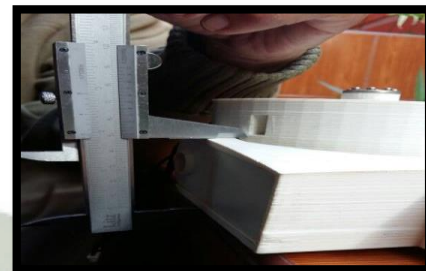
Ancho: 140 mm



Ancho: 140.25 mm



Espesor: 50 mm



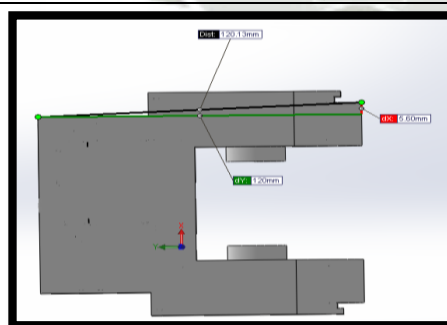
Espesor: 49.95 mm

Fuente: Propia

- **Subensamblado 4:**

Para este eslabón se comparó el largo, ancho y espesor; dando como resultado una alta precisión con un error máximo de ± 0.3 mm como se puede apreciar en la tabla 4.15.

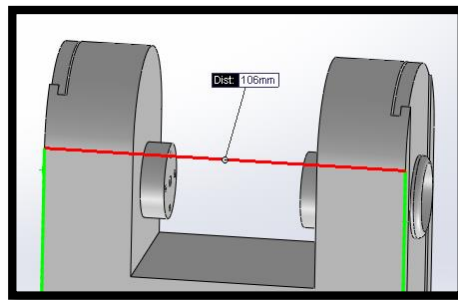
Tabla 4.15: Comparación Subensamblado 4



Largo: 120 mm



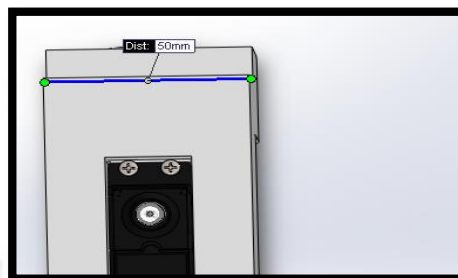
Largo: 119.7 mm



Ancho: 106 mm



Ancho: 105.8 mm



Espesor: 50 mm



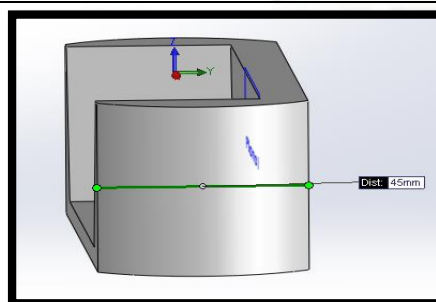
Espesor: 49.8 mm

Fuente: Propia

- Portagarra:

Para este eslabón se comparó el largo, ancho y espesor; dando como resultado una alta precisión con un error máximo de ± 0.8 mm como se puede apreciar en la tabla 4.16.

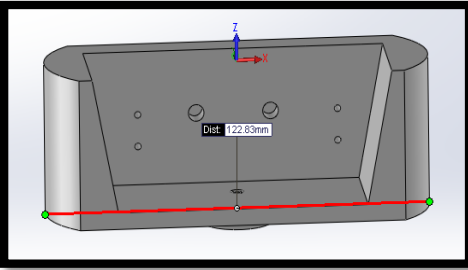
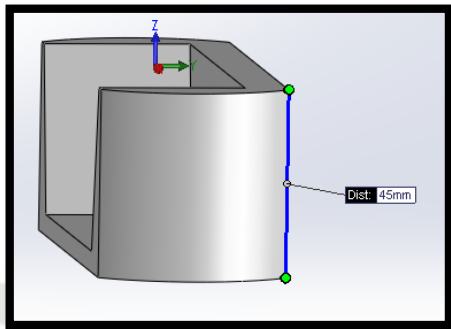
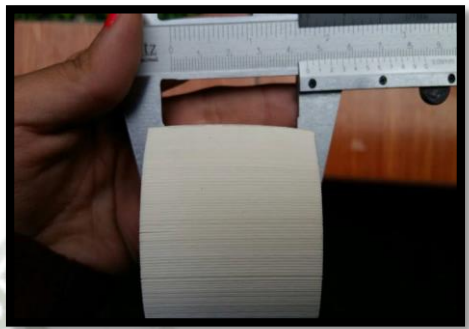
Tabla 4.16: Comparación Portagarra



Largo: 45 mm



Largo: 44.2 mm

 Ancho: 122.83 mm	 Ancho: 121.9 mm
 Espesor: 45 mm	 Espesor: 45.2 mm

Fuente: Propia

4.1.6. Orden de Montaje:

El orden durante el armado de un mecanismo indistintamente es un factor fundamental, ya que garantiza un ensamble acertado de la máquina para su correcto funcionamiento; por lo tanto, es importante establecer una serie de pasos que deberán seguirse durante el montaje de cada uno de los elementos que componen el aparato, siendo en este caso un brazo robótico.

Durante el desarrollo del diseño virtual del manipulador, se determinó que, para realizar el ensamble del prototipo en su totalidad, primero se debe llevar a cabo el armado de 4 subensamblés correspondientes específicamente a los primeros 4 eslabones:

- **Subensamble 1:**

Haciendo referencia al Anexo P que detalla el plano de conjunto de la Base 1, seguimos las siguientes pautas para el montaje del primer subensamble:

- Instalar el Servomotor Dynamixel MX28 por la parte inferior de la Base 1, cerciorándonos que el eje del actuador quede centrado. Asegurar el motor a la

estructura, atornillando 4 elementos de sujeción en sus respectivos agujeros.
(Figura 4.30a y 4.30b)

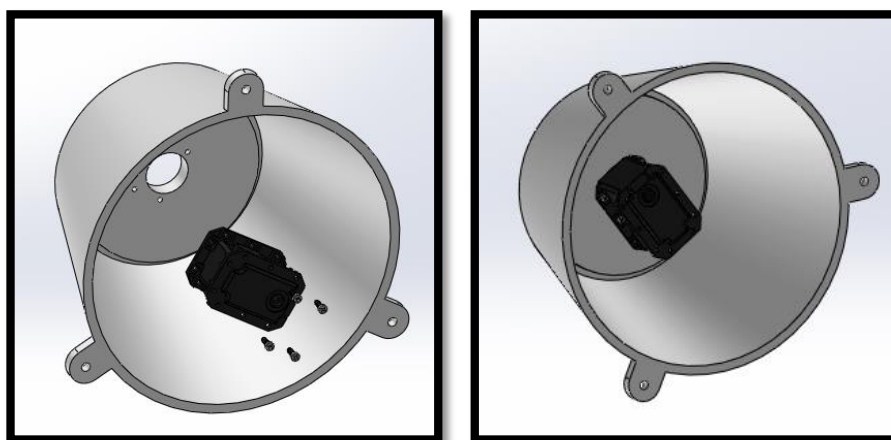


Figura 4.30: a) Disposición de piezas, b) Ajuste de Servomotor en Base 1

Fuente: Propia

- Colocar el rodamiento axial de bolas simple efecto en la cavidad superior de la Base tal como se muestra en la Figura 4.31.

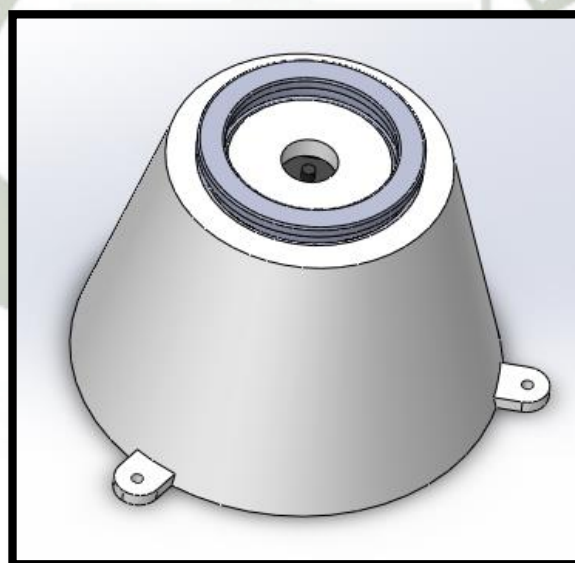


Figura 4.31: Montaje de Rodamiento en Base 1

Fuente: Propia

- **Subensamblado 2:**

Tomando en cuenta el Anexo P, el mismo que contiene el plano de conjunto perteneciente a la Base 2, tenemos:

- Empalmar el Círculo de Unión MX28 en el acople inferior de la Base 2, de tal manera que sus orificios coincidan con los agujeros de la estructura, para

posteriormente ajustar dicha pieza, mediante el uso de 8 tornillos M2-0.4 x 6. (Figura 4.32)

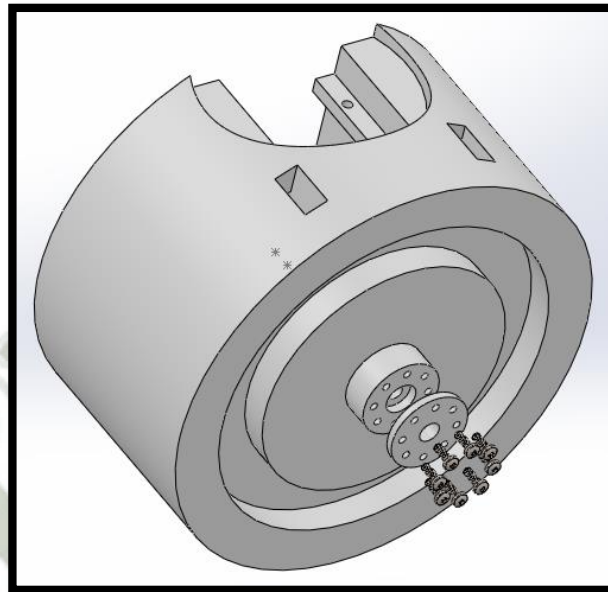


Figura 4.32: Instalación de Círculo de Unión MX28

Fuente: Propia

- Ubicar los dos motores Dynamixel RX-24f en el agujero interior central de la Base 2 y guiar sus cables de conexión de manera adecuada. (Figura 4.33)

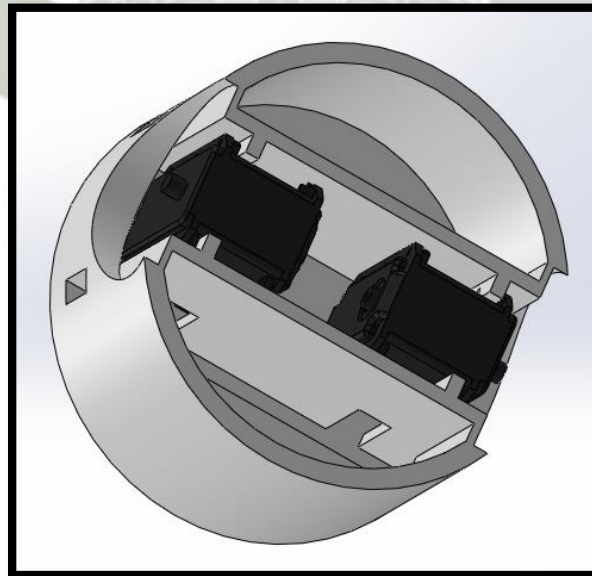


Figura 4.33: Montaje de Motores RX-24f

Fuente: Propia

- Encajar la ovnitapa en la parte superior de la Base 2 y presionarla a fin de que quede completamente sujeta. (Figura 4.34)

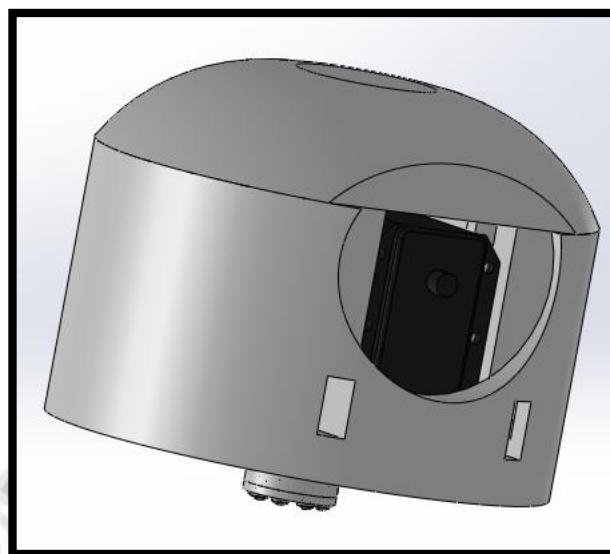


Figura 4.34: Montaje de Ovnitapa

Fuente: Propia

- **Subensamblado 3:**

En base al Anexo P, realizamos el armado correspondiente al Eslabón 1:

- Montar el Soporte Motor en la parte superior del Eslabón 1 y sujetarlo, atornillando 4 elementos de fijación. (Figura 4.35)

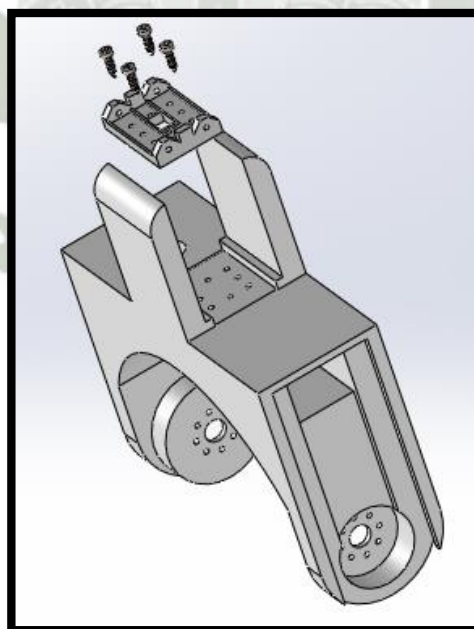


Figura 4.35: Ensamble de Soporte Motor

Fuente: Propia

- Instalar y atornillar el Servomotor Dynamixel Ax12 en el Soporte Motor, para rutear luego los cables de conexión por sus respectivas cavidades. (Figura 4.36)

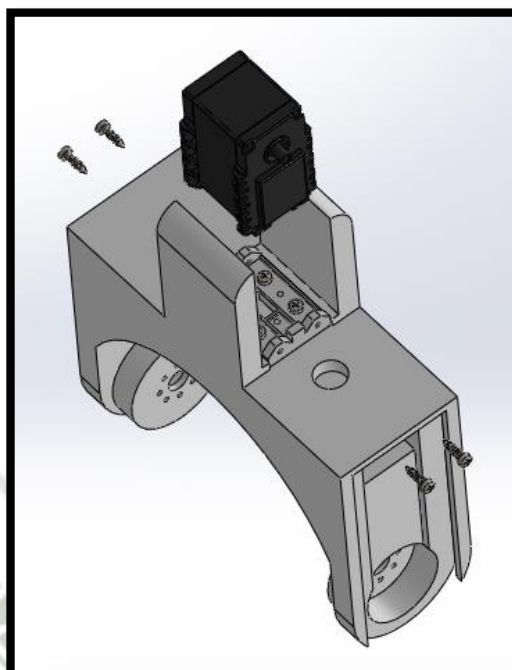


Figura 4.36: Instalación de Motor AX12

Fuente: Propia

- Situar ambos Círculos de Unión RX24f en los acoples laterales del Eslabón 1 y ajustarlos haciendo uso de 16 tornillos M2-0.4 x 6. (Figura 4.37)

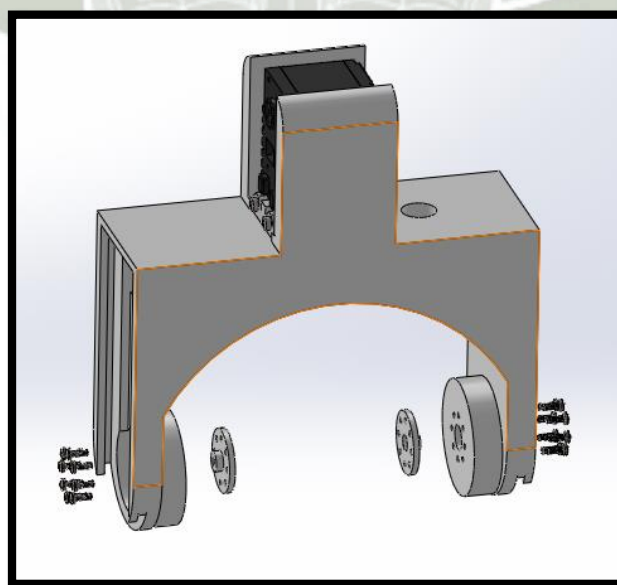


Figura 4.37: Montaje de Círculos de Unión RX24F

Fuente: Propia

- Colocar las dos tapas del Eslabón 1 en sus respectivos carriles, tal como se muestra en la Figura 4.38.

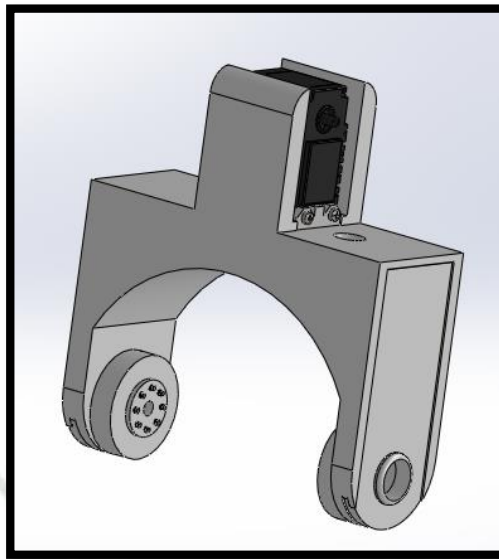


Figura 4.38: Montaje de Tapas laterales
Fuente: Propia

- **Subensamblaje 4:**

Realizamos el montaje del Eslabón 2, tomando como referencia el Anexo P, el mismo que contiene el plano de conjunto correspondiente:

- Ubicar el Servomotor TowerPro SG-5010 en la parte superior del Eslabón 2, ruteando los cables de conexión por el agujero conveniente. (Figura 4.39)

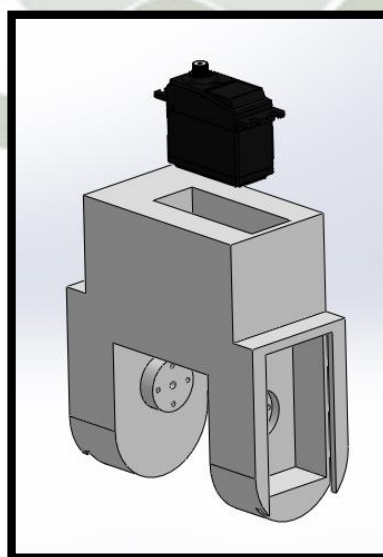


Figura 4.39: Instalación del Servomotor TowerPro SG-5010
Fuente: Propia

- Empalmar el Soporte Motor en la cavidad lateral del Eslabón 2, para seguidamente asegurar el actuador a la estructura mediante el uso de 4 tornillos M2-1x10. (Figura 4.40)

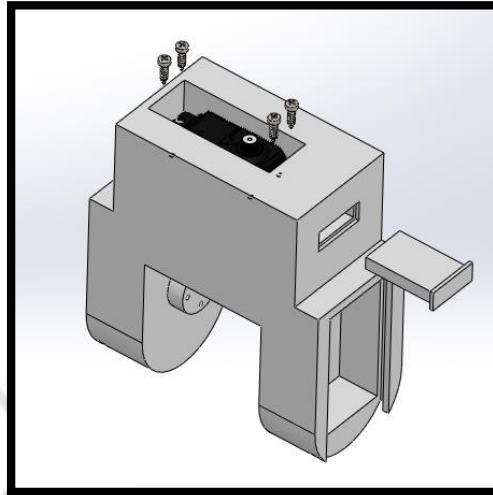


Figura 4.40: Montaje de Soporte Motor
Fuente: Propia

- Instalar las dos tapas laterales del Eslabón 2 en sus carriles determinados, tal como se muestra en la Figura 4.41.

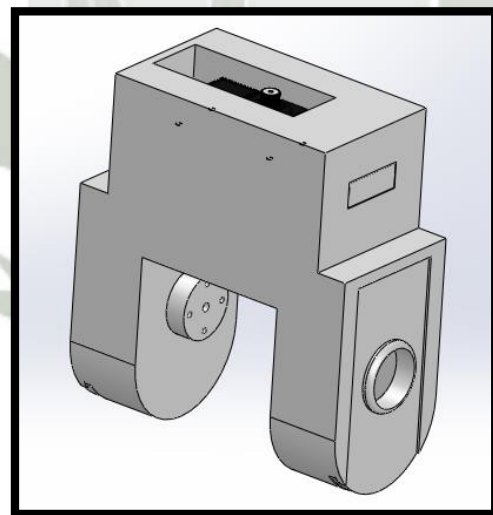


Figura 4.41: Montaje de Tapas laterales
Fuente: Propia

Como segundo paso a seguir realizamos el armado del brazo robótico en su totalidad, uniendo primeramente entre sí los subensambles, para concluir con el montaje del efector final en base al Anexo P:

- Retirar la ovnitapa del Subensamble 2. Acoplar la Base 2 a la Base 1, encajando el Circulo de Unión MX28 al eje del actuador para asegurarlo luego mediante un tornillo M2-0.4 x 20, instalado por la parte central de la estructura de la Base 2. Posteriormente reinstalar la ovnitapa. (Figura 4.42)

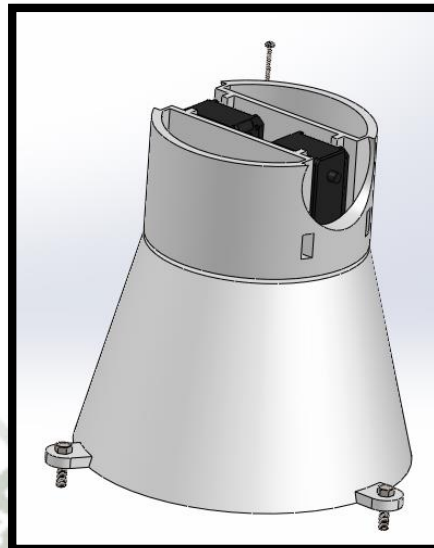


Figura 4.42: Ensamble Base1 y Base 2

Fuente: Propia

- Extraer las dos tapas laterales del Eslabón 1, a fin de efectuar la conexión del mismo con la Base del robot, situándolo de tal manera que los Círculos de Unión RX-24f coincidan con el eje de cada uno de los servomotores específicamente, para después sujetarlos con el uso de 2 tornillos Allen M2.5-0.5 x 8. Seguidamente reponer ambas Tapas. (Figura 4.43)

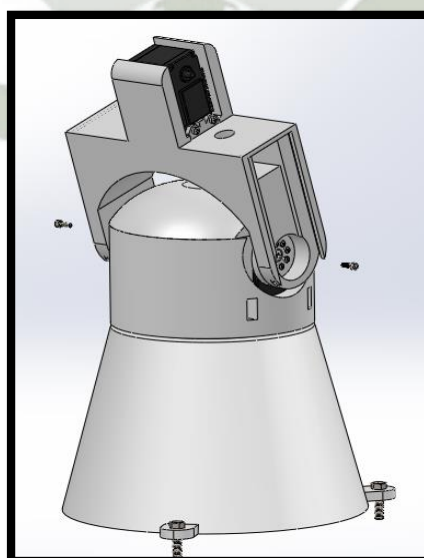


Figura 4.43: Montaje de Eslabón 1 a la Base del robot

Fuente: Propia

- Para realizar el empalme del Eslabón 2 en el Manipulador, desinstalar las tapas de los lados del Subensamble 4, para luego ubicarlo, asegurando el alineamiento de sus acoples laterales con el accionamiento AX-12 y posteriormente enroscar dos tornillos M2.5-0.45 x 13 y cuatro M2-0.4 x 6, que ajusten el motor a la estructura, como se puede ver en la Figura 4.44. Colocar nuevamente las tapas en su lugar.

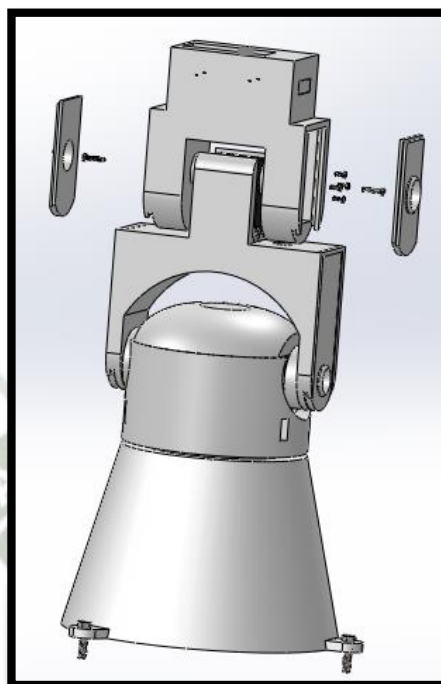


Figura 4.44: Montaje de Eslabón 2 al Brazo Robótico
Fuente: Propia

- Unir el Círculo de Unión TowerPro al acople del Portagarra, haciendo uso de 2 tornillos abrigados M2-0.4 x 10. (Figura 4.45)

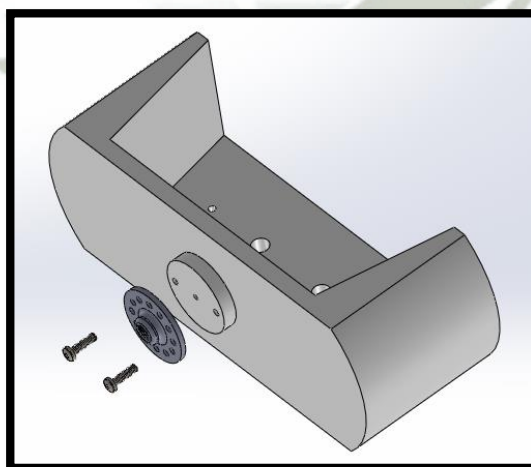


Figura 4.45: Ensamble de Círculo de Unión TowerPro
Fuente: Propia

- Instalar el portagarra en el Brazo Robótico, a través del empalme del Circulo de Unión con el eje del Actuador TowerPro y sujetarla por medio de un tornillo M2-0.4 x 20. (Figura 4.46)

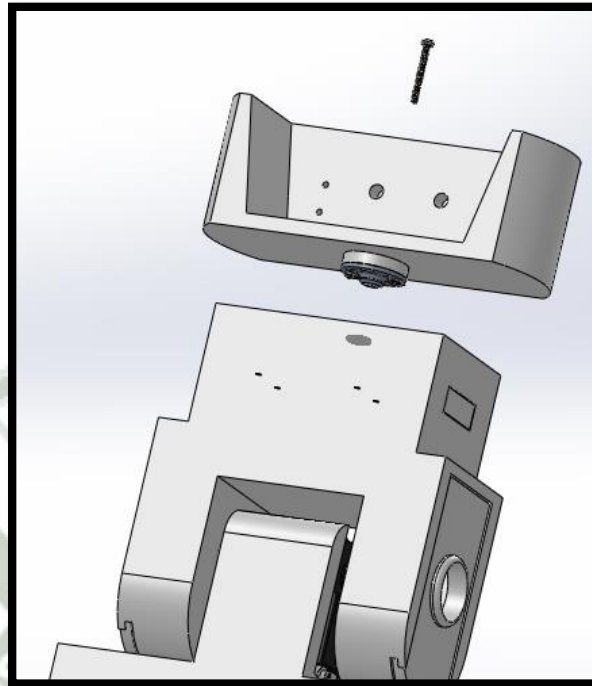


Figura 4.46: Montaje de Portagarra
Fuente: Propia

- Colocar el actuador AX-12 en el interior de la Caja de Motor, guiando los cables de conexión por el agujero respectivo. (Figura 4.47)

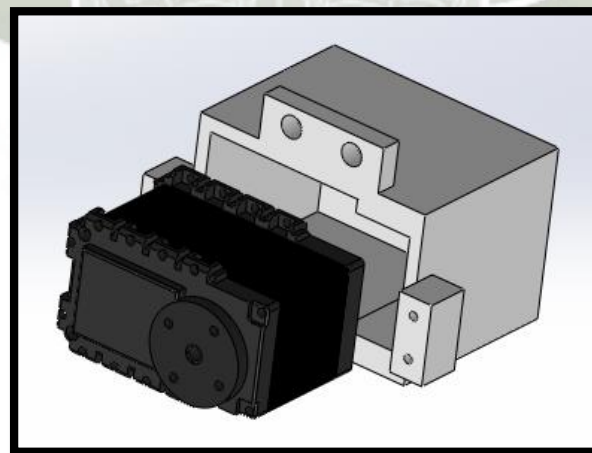


Figura 4.47: Instalación de Servomotor AX-12 de la Garra
Fuente: Propia

- Ensamblar el Engranaje Maestro al Servomotor AX-12, haciendo uso de 4 tornillos M2-0.4 x 6, como se puede observar en la Figura 4.48.

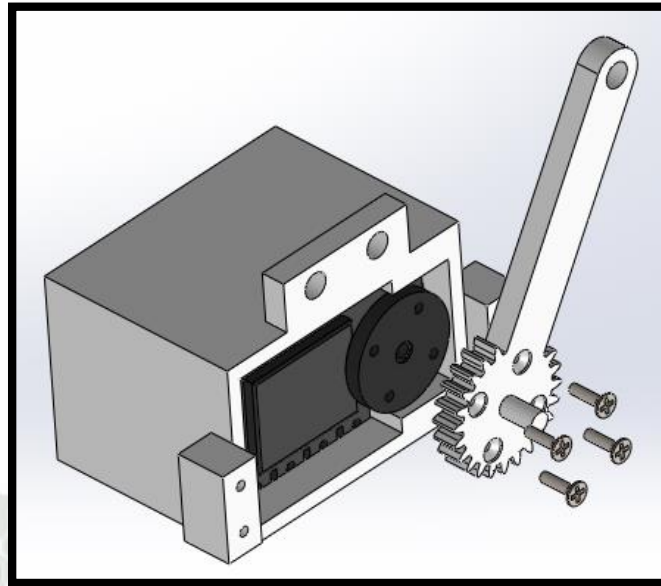


Figura 4.48: Ensamble de Engranaje Maestro

Fuente: Propia

- Encajar ambos Engranajes en el Portagarra, ubicando primero el Engranaje Esclavo en el orificio respectivo, para posteriormente situar el Engranaje Maestro, de tal manera que la garra quede completamente abierta, garantizando una correcta transmisión de movimiento. (Figura 4.49)

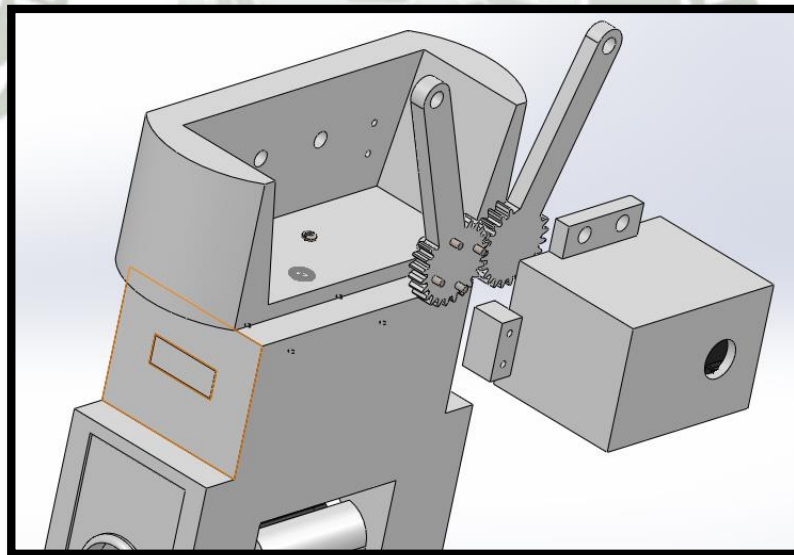


Figura 4.49: Montajes de Engranajes en el Portagarra

Fuente: Propia

- Asegurar la Caja de Motor en el Portagarra, empleando 4 pernos M2-1 x 16 y sus tuercas correspondientes. (Figura 4.50)

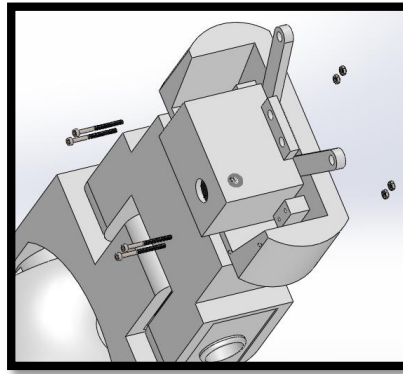


Figura 4.50: Sujeción de Caja Motor
Fuente: Propia

- Instalar los dos Palitos en los orificios sobresalientes de la Caja de Motor, para seguidamente ajustarlos con 2 pernos M5-1 x 25 y sus respectivas tuercas. (Figura 4.51)

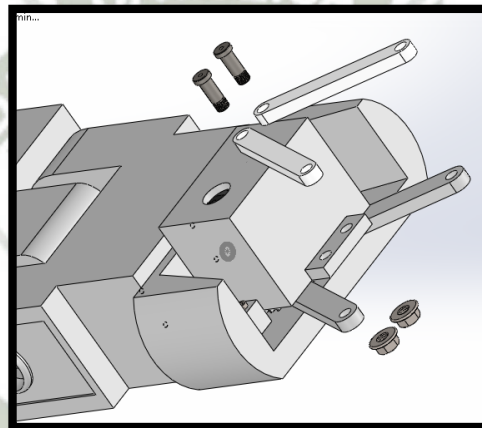


Figura 4.51: Montaje de Palitos
Fuente: Propia

- Finalmente, montar las 4 garras, que conforman los dedos del efector final, en los palitos y engranajes mediante el empleo de 4 pernos M5-1 x 25 y tuercas M5. (Figura 4.52)

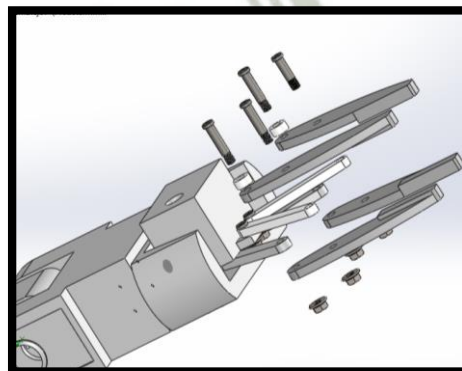


Figura 4.52: Montaje de Dedos de la Garra
Fuente: Propia

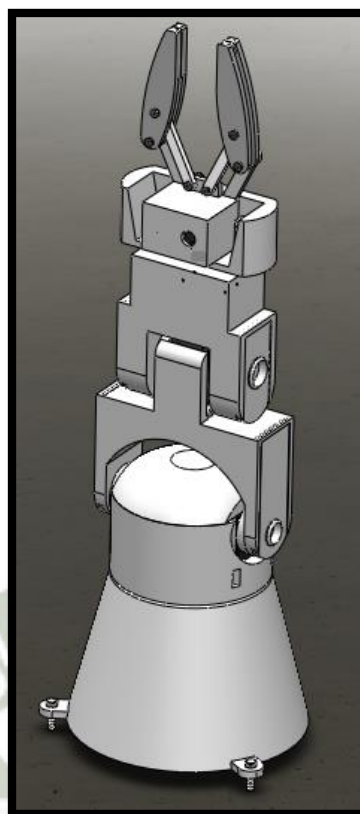


Figura 4.53: Brazo robótico ensamblado
Fuente: Propia

4.1.7. Montaje y Ajustes:

Teniendo impresas las piezas que componen al brazo robótico y siguiendo el orden de montaje predeterminado en el apartado anterior, se realizó el ensamble del manipulador en su totalidad, tal como se puede observar en los siguientes gráficos:



Figura 4.54: Subensamblado Base 1
Fuente: Propia



Figura 4.55: Subensamble Base 2
Fuente: Propia

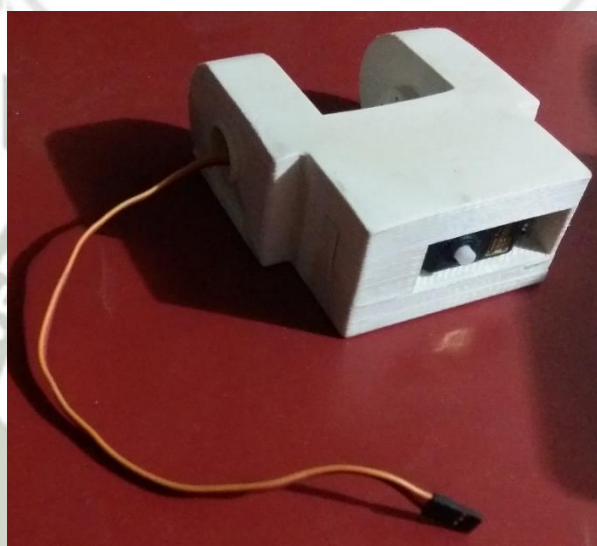


Figura 4.56: Subensamble Eslabón 2
Fuente: Propia



Figura 4.57: Subensamble Garra
Fuente: Propia

Sin embargo, durante el empalme del Eslabón 1 y la Base del Robot, se pudo identificar una forma más rápida y efectiva de realizar dicha unión, pues a diferencia de colocar los motores RX-24f en la Base 2, estos son situados en el Eslabón 1, asegurando la coincidencia de sus ejes con los Círculos de Unión respectivamente. Posteriormente se instala el Eslabón en el manipulador, guiando los cables de conexión de ambos actuadores por las cavidades correspondientes sin que sufran daño alguno. (Figura 4.58a y 4.58b)

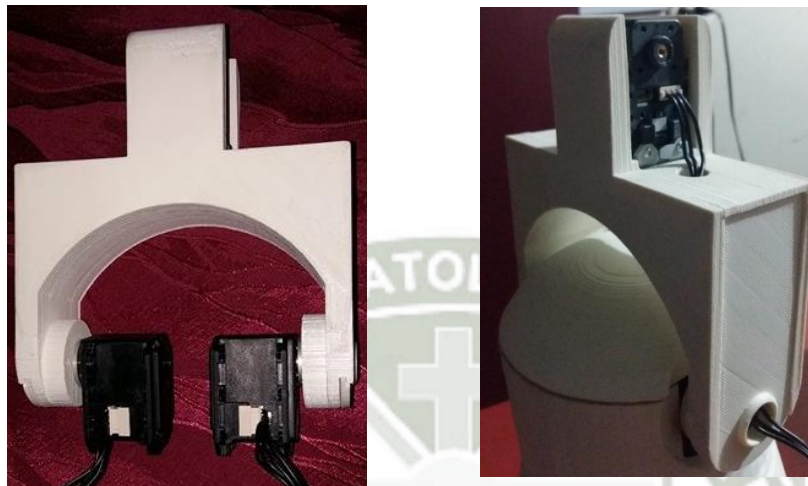


Figura 4.58: a) Montaje de servomotores RX-24f en Eslabón 1 b) Montaje del Eslabón 1 en la base del robot

Fuente: Propia

Así mismo, mientras se llevaba a cabo el desarrollo del armado del manipulador, pudimos notar que era necesario realizar algunos ajustes en los elementos de ensamble instalados a presión tales como tapas del Eslabón 1, tapas del Eslabón 2, Ovnitapa y Soporte Motor TowerPro. En todos los casos se utilizó papel lijar con el objetivo de disminuir algunos contornos de las piezas para que encajen de manera correcta. (Figura 4.59)



Figura 4.59: Ajuste de Piezas

Fuente: Propia

La figura 4.60 nos muestra el brazo robótico de 4 GDL completamente armado.



Figura 4.60: Brazo Robótico 4GDL

Fuente: Propia

4.1.8. Prueba de Funcionamiento del Manipulador:

Una vez ensamblado el brazo robótico en su totalidad, se procede a dar marcha a todos los actuadores en forma simultánea, con el fin de someterlos a una prueba de operatividad con carga y verificar de esta manera el correcto funcionamiento tanto eléctrico como mecánico del manipulador. Para ello, se utilizó una fuente de alimentación variable regulada a 12VDC, siendo esta la tensión adecuada para los servomotores Dynamixel, los mismos que fueron conectados en paralelo; así mismo se hizo uso de una batería de 9VDC, con el propósito de alimentar el motor TowerPRO SG5010; y se dispuso de un arduino MEGA 2560, encargado de enviar la señal de datos y control a los accionamientos. En el caso de los actuadores Dynamixel RX24F es necesario añadir el integrado MAX485 para la lograr una comunicación Half-Duplex.

En las figuras 4.61 y 4.62, se puede observar la distribución y la conexión de los dispositivos.

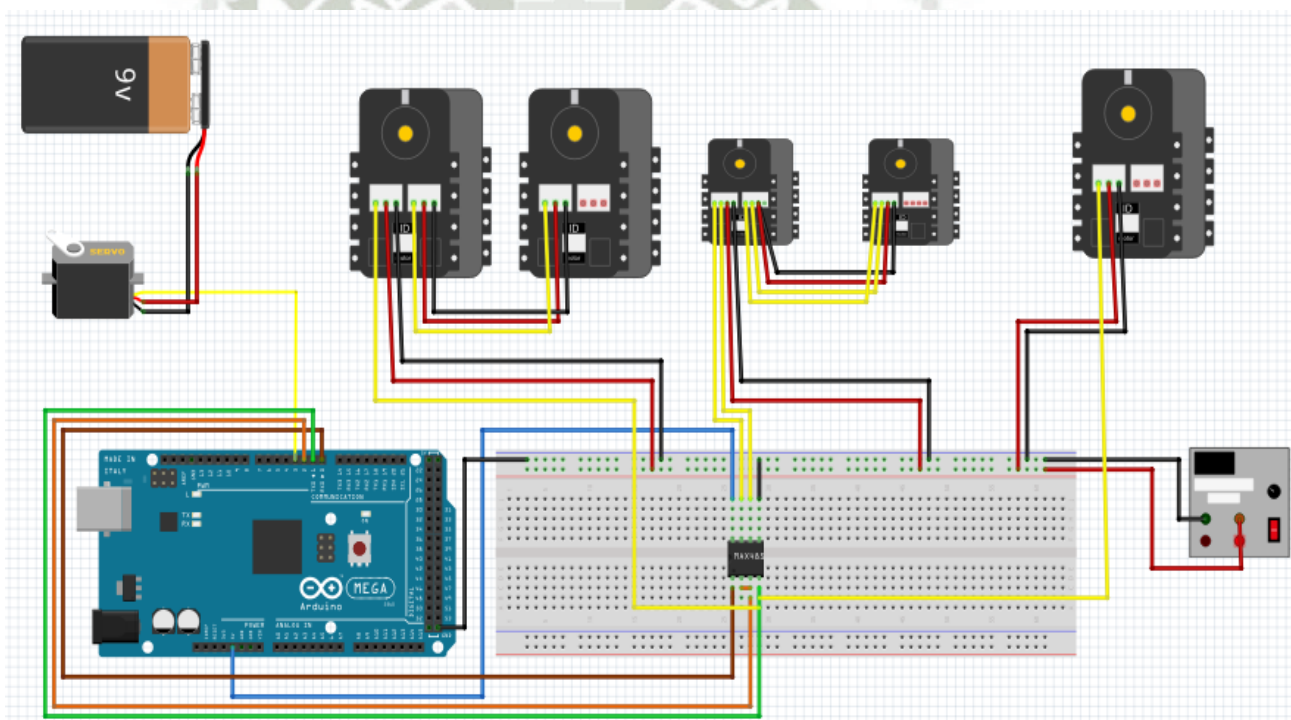


Figura 4.61: Conexión de los servomotores en simultáneo para puesta en funcionamiento

Fuente: Propia

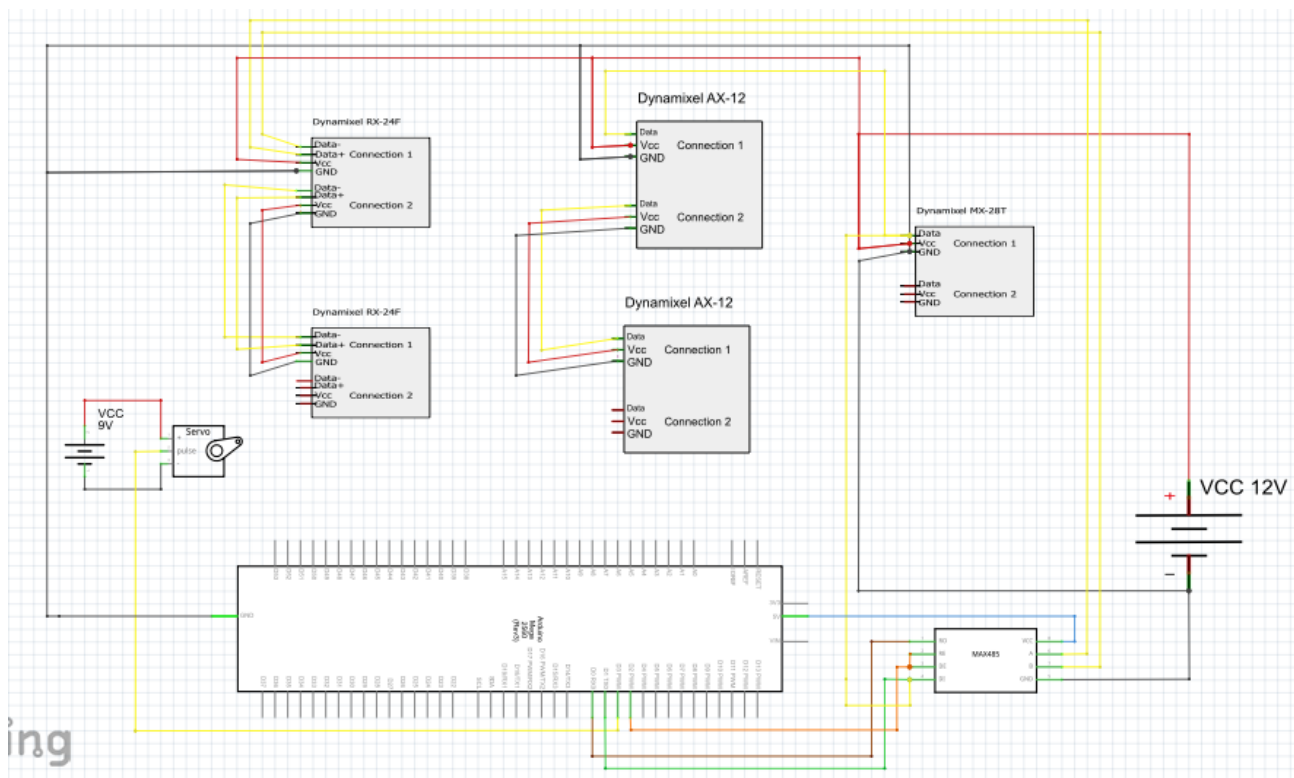


Figura 4.62: Esquema eléctrico de conexión de los servomotores en simultáneo

Fuente: Propia

Posteriormente, se elaboró un código lógico en Arduino IDE denominado “ARMADO” (Anexo D), el mismo que fusionó los antiguos scripts ya elaborados, generando así una secuencia de desplazamiento repetitivo.

Se compiló el software en la placa Arduino y se llevó a cabo el ensayo de forma experimental (Fig. 4.63), mediante el cual se pudo percibir que los actuadores realizaban el movimiento programado de forma correcta, respondiendo así de manera efectiva y confirmando el acierto en el cálculo de torques realizado. Sin embargo, durante el desarrollo de la prueba se observó la existencia de inestabilidad y vibración en la cuarta articulación del manipulador, inconveniente mecánico que fue necesario resolver. Dicha solución se encuentra descrita en el apartado posterior.

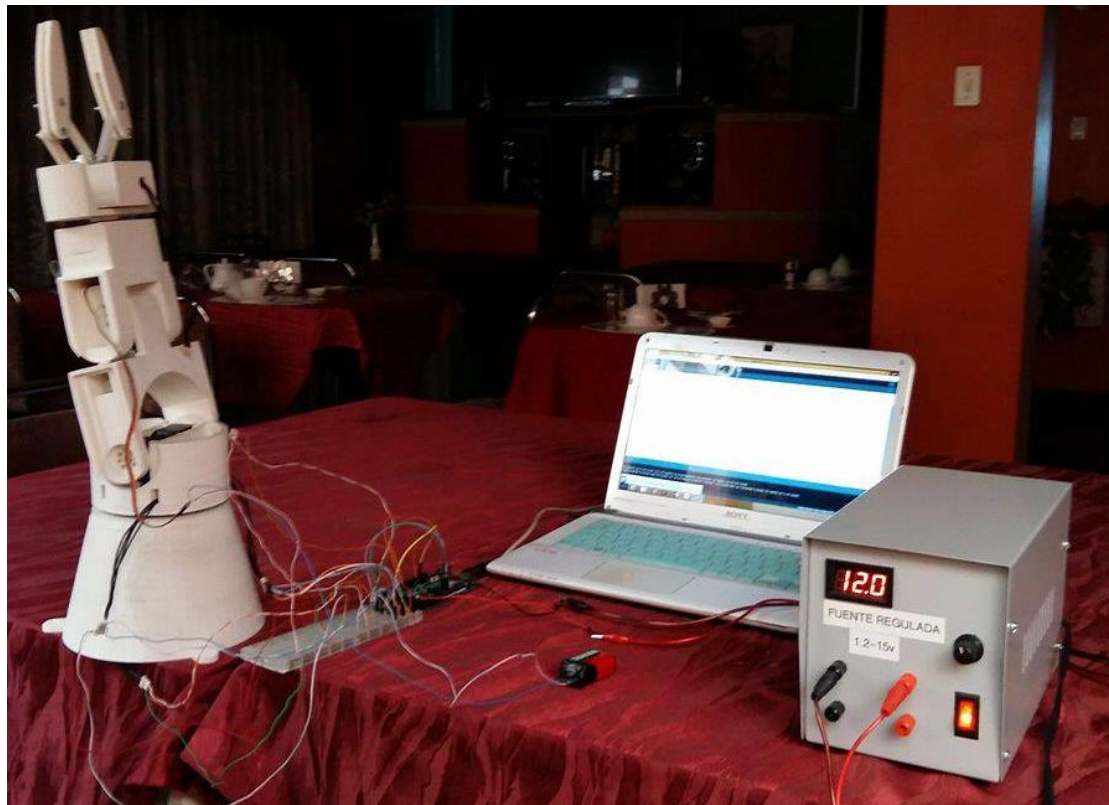


Figura 4.63: Prueba de funcionamiento del Manipulador Robótico
Fuente: Propia

Seguidamente se efectuó una prueba de funcionamiento mecánico y eléctrico en el efector final del brazo robótico, cuyo propósito es verificar que este ejerza una fuerza de agarre adecuada para que el objeto a manipular no resbale durante su desplazamiento de un punto a otro (Fig. 4.64). Para ello se realizaron los mismos pasos anteriormente descritos en el ensayo precedente, con la diferencia de llevar a cabo la aproximación de un elemento cualquiera hacia la garra de tal manera que simule el proceso de manipulación de un instrumento de laboratorio. Es así como se pudo observar claramente que el gripper no cumplía con el objetivo planteado, pues durante el movimiento del robot, el objeto que sostenía se deslizaba, razón por la cual es de vital importancia hallar una medida que corrija el problema dado, la misma que esta puntualizada en la siguiente sección.



Figura 4.64: Prueba de funcionamiento del Efector Final.
Fuente: Propia

4.1.9. Correcciones y/o Modificaciones al Ensamble Final:

Dada la exposición de los errores hallados, procedemos a llevar a cabo el desarrollo de las soluciones propuestas.

4.1.9.1. Diseño del soporte para la estabilidad del último eslabón:

Debido a la existencia de inestabilidad física en la última articulación del brazo robótico, es que se vio por conveniente diseñar una pieza que sirva de soporte (Fig. 4.65) al último eslabón del manipulador, de tal manera que éste tenga un apoyo donde asentarse evitando así vibraciones significativas en el efector final.

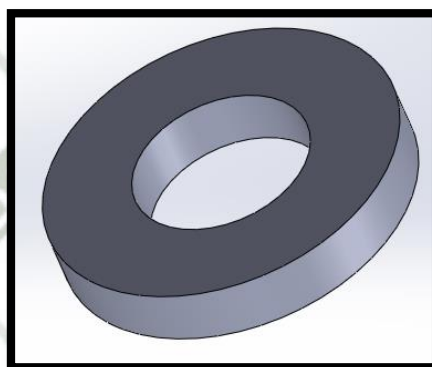


Figura 4.65: Soporte para estabilidad del Efector Final.

Fuente: Propia

La ubicación de esta pieza se puede apreciar en la siguiente figura:

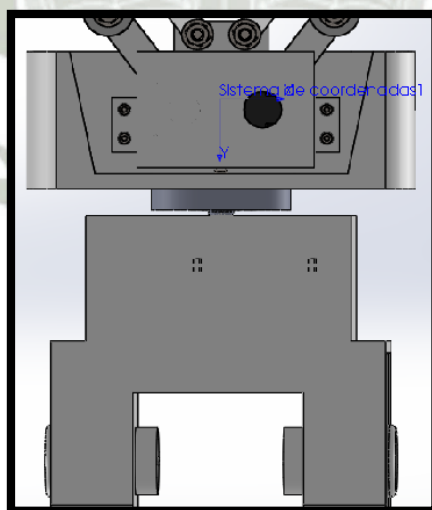


Figura 4.66: Soporte para estabilidad ubicado entre el eslabón 2 y la garra

Fuente: Propia

4.1.9.2. Modificaciones del efector final:

El efector final constituye una o la pieza más significativa que compone al robot, debido a que sin él no se lograría el objetivo principal de esta tesis, que está implícitamente relacionado con la aplicación del manipulador. Es por ello que resulta fundamental remediar el problema mecánico descrito anteriormente.

- **Mejoramiento en los puntos de sujeción de la garra**

Durante el desarrollo de la segunda prueba de funcionamiento del brazo robótico, se identificó que la fuerza en el punto de agarre del gripper no era suficiente para la manipulación de un instrumento de laboratorio; razón por la cual, se realizó un análisis del problema determinando así que el torque del motor transmitido desde su eje hacia el extremo de la garra, reduce su efecto en los puntos de sujeción de la misma, debido a que al llevar a cabo el ajuste de los pernos, las uniones móviles quedaban fijas.

De esta manera se planteó mejorar los puntos de unión del efector final, colocando bujes fabricados en poliamida y cambiando los pernos de 5mm de diámetro por otros de 1/8 de pulgada (3.175mm) con sus respectivas arandelas y tuercas; los mismos que son instalados por el interior de cada buje. (Fig. 4.67)



Figura 4.67: Instalación de bujes en los puntos de unión del Efector Final.

Fuente: Propia

Posteriormente, se efectuó una tercera prueba de funcionamiento del robot, para verificar si el inconveniente había sido resuelto; no obstante, se pudo percibir que

si bien el movimiento del gripper había mejorado manteniendo la estabilidad y fijación de sus piezas, no se alcanzaba aún la fuerza de agarre suficiente, pues los elementos a manipular continuaban deslizándose no cumpliendo así con su recorrido completo, es por ello que se optó por una segunda medida de solución, la misma que será detallada en el siguiente apartado.

- **Rediseño de gripper:**

Dado que la modificación en la sujeción no generó el agarre esperado, se procedió a rediseñar el efector final de la siguiente manera:

-En primera instancia se creó una nueva garra con una forma en L con cierto ángulo de inclinación para lograr que el punto de agarre esté a menor distancia del eje del motor donde se genera el momento de torsión, transmitiendo así mayor fuerza de sujeción, sin dejar de lado su resistencia y estética; además se cambió la geometría final de la garra de ser totalmente plana a una forma ovalada en el medio, para lograr una mayor superficie de contacto con el objeto a manipular, siendo este de estructura cilíndrica. En la Figura 4.68 se puede observar una comparación gráfica entre el diseño anterior y final de la pieza.

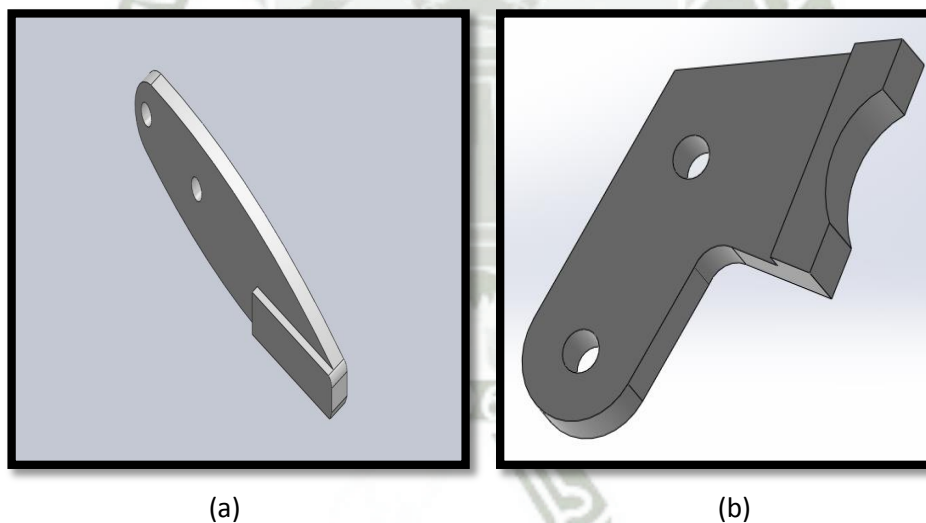


Figura 4.68: a) Diseño anterior de Garra y b) Diseño final de garra

Fuente: Propia

- Se modificó también los engranajes (Fig. 4.69) y soportes (Fig. 4.70) del efector final, reduciendo su tamaño para lograr el efecto antes mencionado.

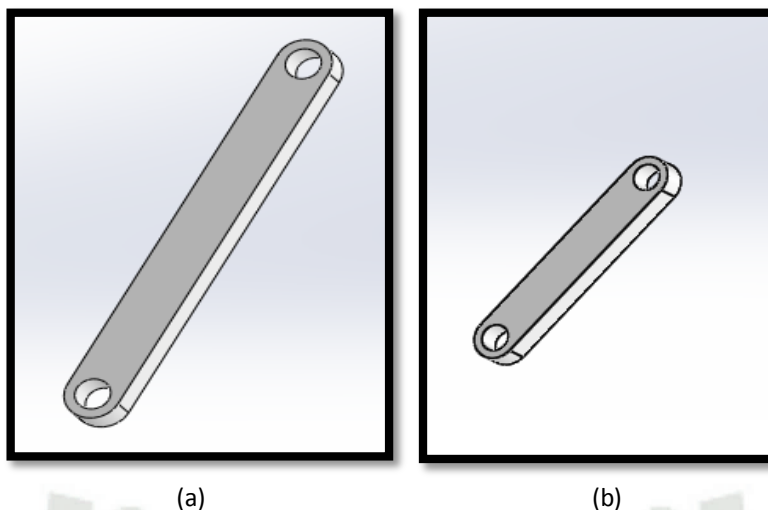


Figura 4.69: a) Diseño anterior de Soporte y b) Diseño final de Soporte
Fuente: Propia

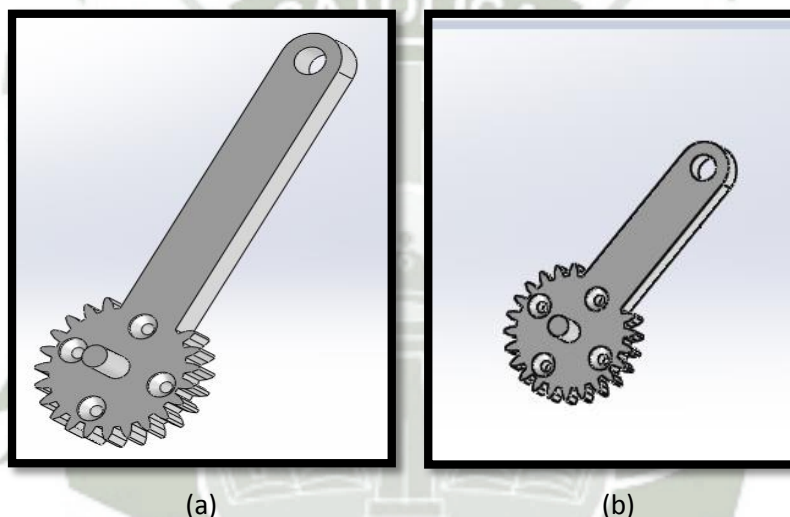


Figura 4.70: a) Diseño anterior de engranaje y b) Diseño final de Engranaje
Fuente: Propia

-Seguidamente se procedió a realizar los ensayos correspondientes para comprobar si la nueva garra cumple con las propiedades físicas de resistencia adecuadas.

- **Ensayo Estático:**

Se agregó dos fuerzas de 10N en direcciones distintas; una simulando el peso de la carga máxima a soportar por el brazo robótico y la otra simulando la fuerza de reacción del instrumento a manipular; de igual manera, se tomó en cuenta la gravedad para que el estudio sea lo más real posible. (Fig. 4.71)

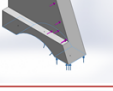
Nombre de carga	Cargar imagen	Detalles de carga
Fuerza-1		Entidades: 1 cara(s) Tipo: Aplicar fuerza normal Valor: 10 N
Fuerza-2		Entidades: 3 cara(s) Tipo: Aplicar fuerza normal Valor: 10 N
Gravedad-1		Referencia: Cara: 1 > Valores: 0 0 9,81 Unidades: SI

Figura 4.71: Tensiones de Prueba aplicadas a la garra con bujes

Fuente: Propia

Se obtuvo los siguientes resultados:

- **Tensiones:** Nos muestra un gráfico, donde es posible analizar las tensiones que puede soportar el elemento como máximo, en función del límite elástico del material. Como se puede ver en la figura 4.72 se tienen los valores máximo y mínimo de esfuerzo en el elemento, encontrándose el punto más crítico sometido a 1.4 MPa.

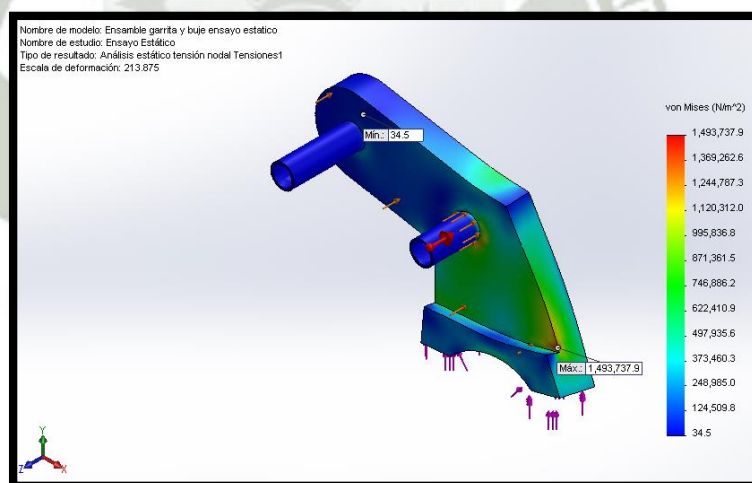


Figura 4.72: Resultado de Tensión

Fuente: Propia

- **Desplazamientos:** Nos muestra en milímetros, cual es el desplazamiento de la pieza respecto a las fuerzas soportadas, teniendo como mayor valor 0.03 mm. (Fig. 4.73)

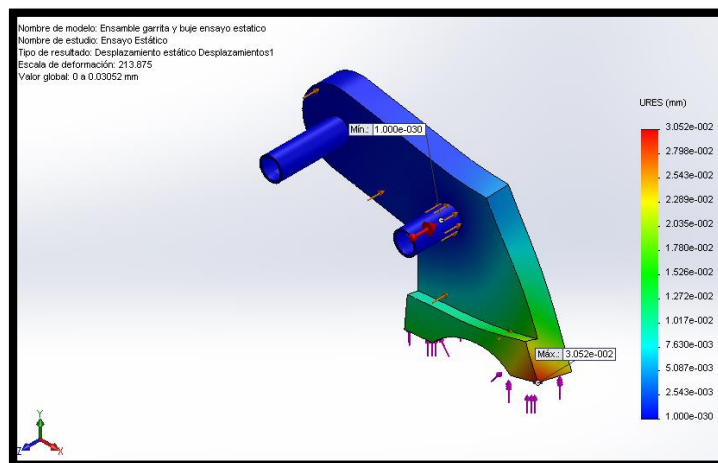


Figura 4.73: Resultado de Deformaciones

Fuente: Propia

- **Factor de Seguridad:** Está determinado por el Solidwork y sirve para entender en base a los resultados anteriores cuantas veces puede soportar nuestra pieza, la carga aplicada. De esta manera, al observar la figura 4.74, podemos concluir que el componente es capaz de resistir 29.13 veces más carga que a la que fue sometida.

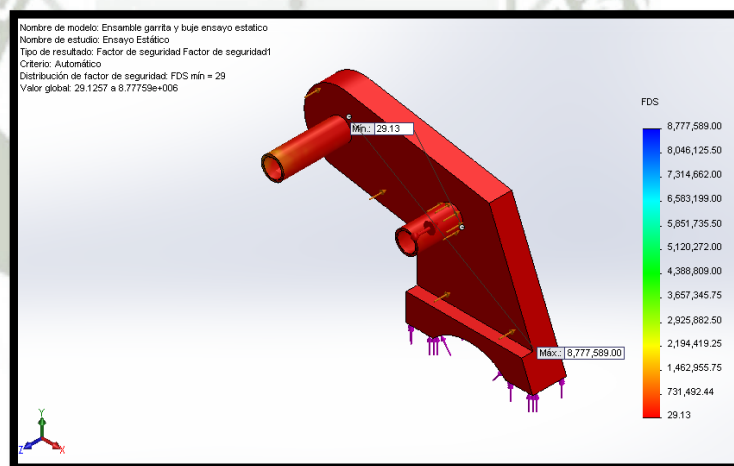


Figura 4.74: Resultado de Factor de Seguridad

Fuente: Propia

- **Ensayo de Fatiga:**

Para este ensayo se tomó en cuenta nuevamente 1000000 de repeticiones para simular la vida infinita (Fig. 4.75) y las curvas S-N de cada material, en este caso Nylon 6/10 para los bujes y ABS para la garrita. (Fig. 4.76)

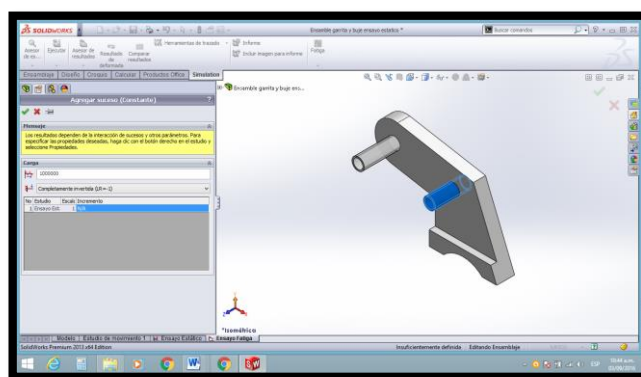


Figura 4.75: Suceso para ensayo de Fatiga
Fuente: Propia

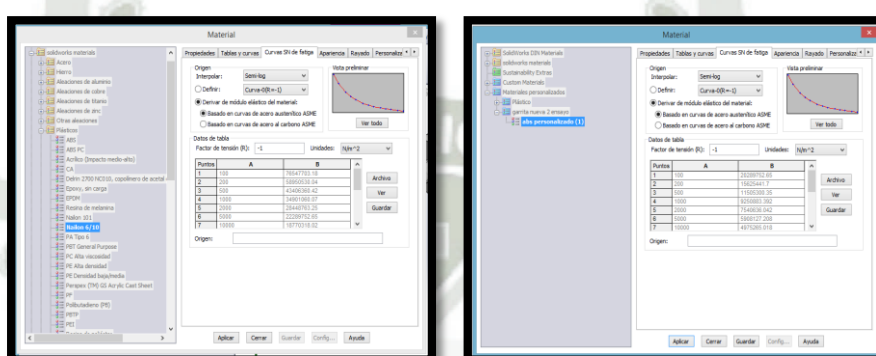


Figura 4.76: Curvas S – N de cada material
Fuente: Propia

Tras realizado el ensayo, adquirimos el siguiente resultado:

- **Vida total:** Indica el número de repeticiones máximas que puede soportar la pieza en distintos puntos a lo largo de ella. En este caso podemos observar que en toda la pieza se mantiene el 1000000 de repeticiones lo cual lo hace óptima en todo sentido. (Fig. 4.77)

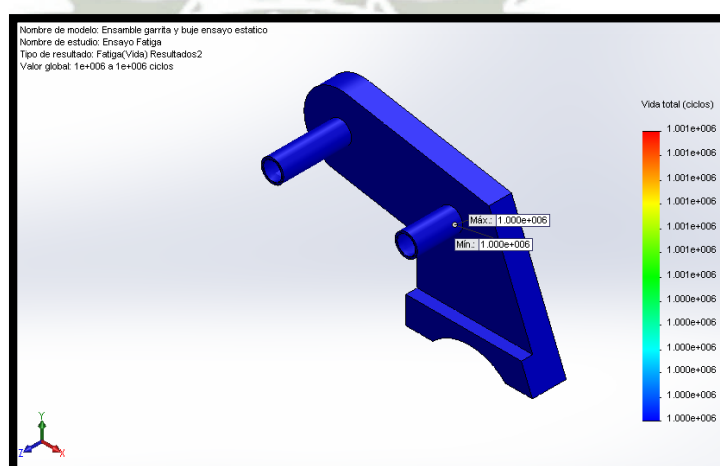
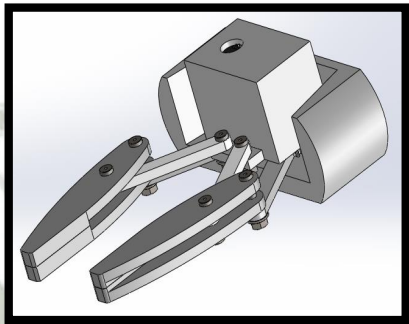
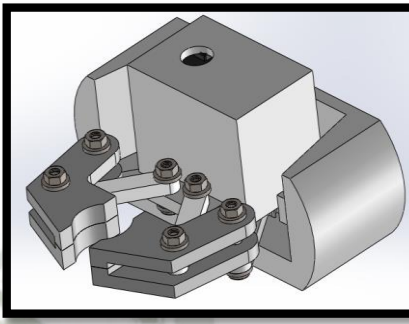


Figura 4.77: Gráfica de vida total
Fuente: Propia

Llevado a cabo ambos ensayos podemos concluir que el nuevo diseño propuesto cumple con los requerimientos de resistencia necesarios para nuestro brazo robótico.

-Posteriormente, se ejecutó una comparación de las propiedades físicas entre el subensamble del efector final precedente y el actual, con el objetivo de analizar si el motor AX-12 posee aún el momento torsor adecuado para el movimiento de este último eslabón. (Tabla 4.17)

Tabla 4.17: Comparación de propiedades físicas entre el efector final precedente y actual

		
	Garra Anterior	Garra Nueva
Masa(kg)	0.2703	0.25439
Izz(gm*mm^2)	263538.29	376524.86
Lc (mm)	42.12	34.94
L (mm)	50	50.50
Ensayo Estático de la garrita		
Tensiones (MPa)	5.06	1.4
Desplazamiento (mm)	0.4678	0.03
Factor de Seguridad	11	29.13
Ensayo de Fatiga de la garrita		
Vida Útil (mín)	5768	1000000

Fuente: Propia

Luego de analizar los valores comparados en la tabla anterior podemos concluir que el diseño propuesto para la nueva agarra es inclusive mucho más eficiente que la anterior, lo cual nos asegura que el motor AX-12 es el adecuada para dicho eslabón.

Con respecto al montaje del nuevo efector final, se considera el orden de armado del gripper anterior ya planteado, con la diferencia de la incorporación de bujes, arandelas y el cambio de diámetro en los pernos de 5mm a 1/8 de pulgada, tal como se puede observar en la figura 4.78.

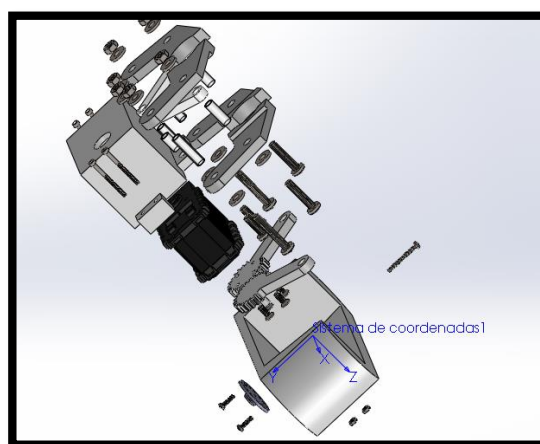


Figura 4.78: Montaje de Efector Final.

Fuente: Propia

De esta manera se obtiene un nuevo diseño del último eslabón, que dado el análisis comparativo antes mencionado cumple de sobremanera todos los requerimientos expuestos al inicio del proyecto.

Para finalizar ensamblamos este nuevo diseño al robot completo como se observa en la figura 4.79.

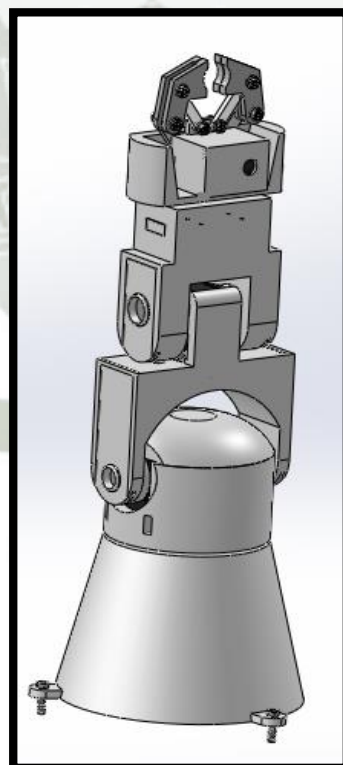


Figura 4.79: Ensamble final del brazo robótico

Fuente: Propia

4.1.10. Nuevo Espacio de Trabajo:

El espacio de trabajo de un robot es grupo de puntos que puede ser alcanzado por su efector final, esto se ve determinado por las longitudes y ángulos de rotación de las articulaciones; teniendo en cuenta esto es que se en las figuras 4.80 y 4.81 se muestra el alcance del brazo robótico, así como algunas dimensiones

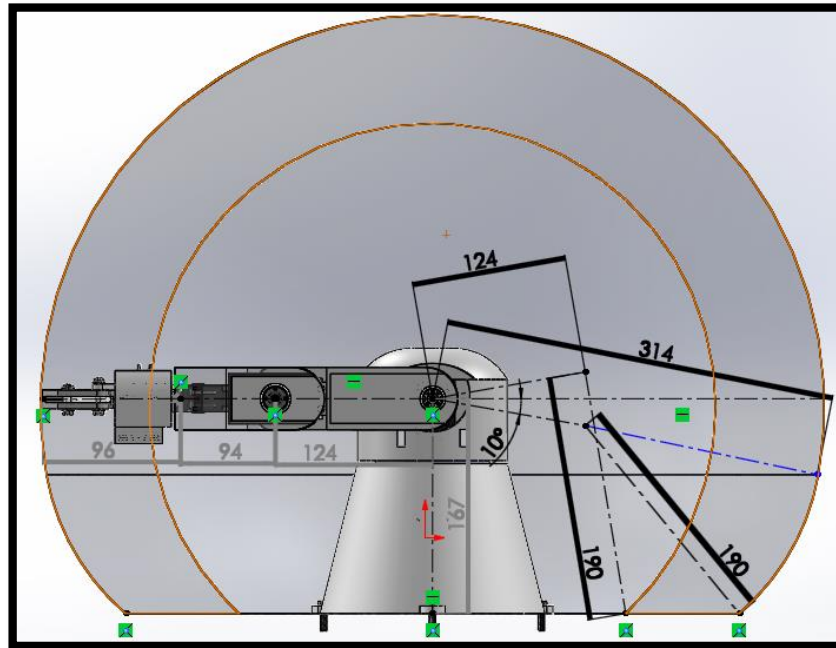


Figura 4.80: Vista lateral del espacio de Trabajo
Fuente: Propia

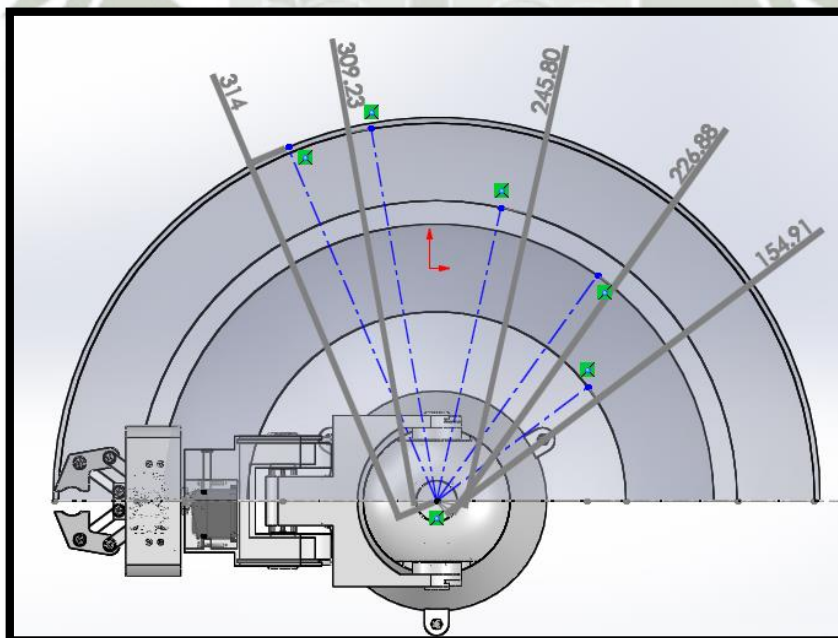


Figura 4.81: Vista superior del espacio de Trabajo
Fuente: Propia

En la tabla 4.18 se muestra los rangos de giro de cada articulación, siendo estos: base, eslabón 1, eslabón 2 y eslabón 3 o garra respectivamente.

Tabla 4.18: Ángulos de rotación de cada articulación

Articulación	Rango de giro
Base	De 0° a 180°
Eslabón 1	De -10° a 90°
Eslabón 2	De -90° a 90°
Eslabón 3	De 0° a 180°

Fuente: Propia

4.1.11. Pruebas Finales:

Realizado el armado del efector final y su respectivo montaje en el brazo robótico, se ejecuta una cuarta prueba de funcionamiento, cuya finalidad radica en comprobar si la fuerza en el punto de agarre del gripper incrementó, de tal manera que el robot desarrolle una apropiada manipulación de instrumentos de laboratorio.

Para ello, primero se llevó a cabo la conexión eléctrica de los actuadores a las fuentes de alimentación, a la plataforma software Arduino y al integrado MAX 485, como se puede observar en las Figuras 4.61 y 4.62. Seguidamente, se creó un código lógico en Arduino IDE, denominado “SECUENCIA” (Anexo E), el mismo que tiene como propósito simular la manipulación de un instrumento de laboratorio. Finalmente se compiló el script, dando marcha al proceso de desplazamiento repetitivo programado y utilizando un tubo de ensayo como objeto a desplazar.

La Figura 4.82, nos muestra los ocho movimientos realizados por el robot durante el ensayo.





Figura 4.82: Secuencia de desplazamiento del brazo robótico.

Fuente: Propia

Dada la obtención de resultados positivos a la prueba realizada, se puede concluir que los cambios llevados en cabo en el efector final son acertados, cumpliendo así con los requerimientos y objetivos propuestos.

4.2. Implementación Electrónica:

4.2.1. Herramientas usadas para la implementación mecánica del brazo robótico:

4.2.1.1. Herramientas Virtuales:

Entre las herramientas virtuales usadas en nuestro proyecto se encuentran:

- **Python:** Lenguaje con el cual se crearon los diferentes programas tanto de prueba de componentes como de uso final, se escribió dentro del procesador Raspberry Pi 2 model b+ y se ejecutó desde su consola de comandos.

Lo que nos llevó a utilizarlo fue que nos permitía escribir cada programa con sencillez y elegancia, utilizando menos líneas que con otros lenguajes y sin perder nada de claridad.

4.2.1.2. Herramientas Mecánicas:

Entre las herramientas mecánicas usadas para nuestro proyecto tenemos; un juego de 6 perilleros que debido a la cantidad de componentes electrónicos de pequeñas dimensiones se hizo imprescindible su uso para la comodidad y agilidad al momento de atornillar y desentornillar, un pelacables que se utilizó para el corte de cables de todo el proyecto gracias a que incorpora una serie de muescas de distintos calibres en la parte metálica que sirven para sujetar con facilidad la cobertura exterior de un cable, para luego hacer un pequeño corte y tirar del mismo para dejar los alambres de cobre a la vista para poder trabajar en la conexión; y un alicate de punta que uso en el proyecto para manipular elementos electrónicos de tamaño reducido tales como integrados, resistencias, capacitores entre otros ya que su forma alargada, y tener un mango (generalmente forrado de plástico) hacen que sea útil para alcanzar o colocar objetos en cavidades donde los cables u otro material se han atorado, son inalcanzables o imposibles de colocar para los dedos y otros medios.

4.2.1.3. Herramientas Eléctricas:

Entre las herramientas eléctricas usadas para nuestro proyecto tenemos: un soldador eléctrico o de estaño utilizado para todas las modificaciones o correcciones encontradas a lo largo del proyecto, una cinta aislante para prevenir algún corto circuito y velar por nuestra seguridad en el desarrollo de este trabajo, un multímetro necesario para mediciones y revisiones de los componentes electrónicos.

4.2.2. Instalación del Sistema Operativo del Procesador:

Inmediatamente luego de haber escogido como procesador el Raspberry Pi Model B+, era necesario decidir que sistema operativo y que versión del mismo íbamos a utilizar, conociéndose que en la actualidad son muchos los sistemas operativos, revisamos 5 de los más comunes:

- **Raspbian**, es una versión de Linux basada en Debian y especialmente desarrollada para Raspberry, y también una de las más populares. En realidad, es un poco el "sistema operativo oficial" de este pequeño ordenador. Viene preinstalado con software educativo, para programación y para uso general.
- **Fedora**, que ha creado una versión especial para Raspberry Pi 2. El proyecto cuenta con una amplia documentación online, y además para todos los niveles. Ésta no es la primera versión de Fedora para la Raspberry; la primera fue Fedora, que ya no se actualiza, pero todavía se puede descargar.
- **Arch Linux**, una de las opciones que viene incluida en el software de instalación principiante para Raspberry Pi. Es un sistema ligero, con un diseño simple lo hace ideal para un pequeño ordenador como la Raspberry.
- **Kano Os**, es el sistema operativo incluido en Kano Kit, el ordenador especialmente pensado para niños. Dado que este sistema operativo está pensado para niños, tiene una interfaz algo infantil.
- **Windows IoT Core**, se trata más bien de una plataforma de desarrollo para que los programadores experimenten con dispositivos conectados a Internet, necesita otro ordenador con Windows 10 y con Visual Studio instalado.

Aunque todos eran muy llamativos optamos por elegir Raspbian entendiendo que es la más usada y con mayores aplicaciones, sin embargo, también había que elegir la versión a instalar en nuestro Raspberry pi 2, lo que nos llevó a comparar todas sus versiones y elegir Debian 7 Wheezy basados en que a pesar de no ser la última es la versión más estable y la única que es compatible con Matlab, siendo necesaria dicha compatibilidad para poder enviar el programa convertido a lenguaje en C.

4.2.3. Pruebas Individuales:

4.2.3.1. Raspberry pi y Pantalla LCD:

Una vez instalado el sistema operativo en la memoria SD Card, procedemos a ubicar la tarjeta en la Raspberry Pi para comprobar su correcto funcionamiento; para ello es también necesario conectar la pantalla LCD, el teclado y un mouse con la finalidad de visualizar y operar diferentes funciones del sistema.

En el preciso momento en el que se encendió la Raspberry, se pudo notar que la pantalla LCD no tenía el mismo efecto (Fig. 4.83), lo que nos llevó a la realización de una modificación en las conexiones, la misma que será descrita a mayor detalle en el apartado de correcciones.



Figura 4.83: Prueba de funcionamiento de la Raspberry y la pantalla LCD
Fuente: Propia

4.2.3.2. Kinect:

Con la finalidad de verificar el buen funcionamiento del Kinect XBOX 360, es que se llevó a cabo una prueba de operatividad, la misma que fue realizada a través de la computadora portátil y el Matlab.

Para ello, inicialmente se ejecutó la instalación de los softwares OPENNI, Microsoft SDK, Nite 2 y Visual Basic 10 en la PC, a modo de que ésta pueda reconocer el dispositivo, y la librería Kinect for Windows Sensor Support Package en Matlab, con el fin de comenzar a programar (Fig. 4.84).

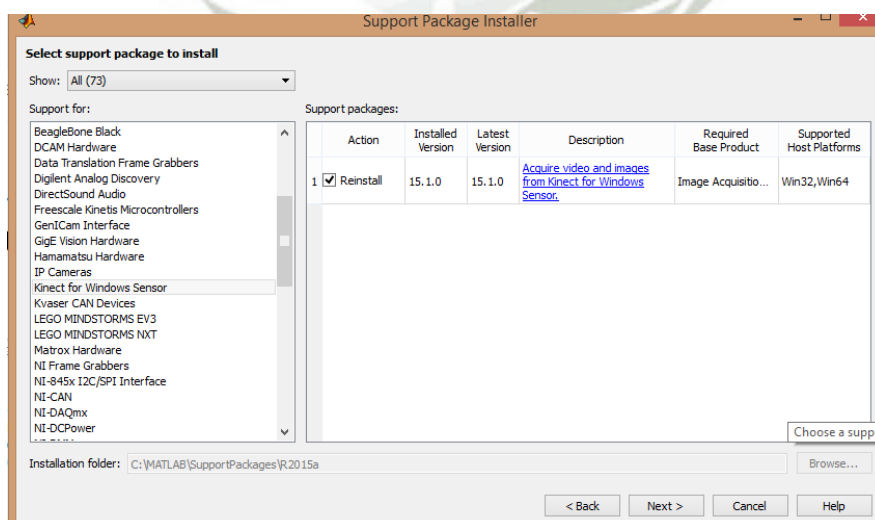


Figura 4.84: Instalación de la Librería “Kinect for Windows Sensor Support Package”
Fuente: Propia

Posteriormente, se realizó un script, “realtimehand” (Anexo F), en el que se sometió a prueba la función Skeleton de la Librería; por ser la de mayor incidencia en el desarrollo de la presente tesis. Este programa consiste en mostrar conjuntamente la imagen RGB de la persona captada por la cámara con un mapeo de las articulaciones y el esqueleto de su brazo derecho de forma consecutiva. Finalmente, como se puede observar en la figura 4.85, los resultados obtenidos fueron satisfactorios

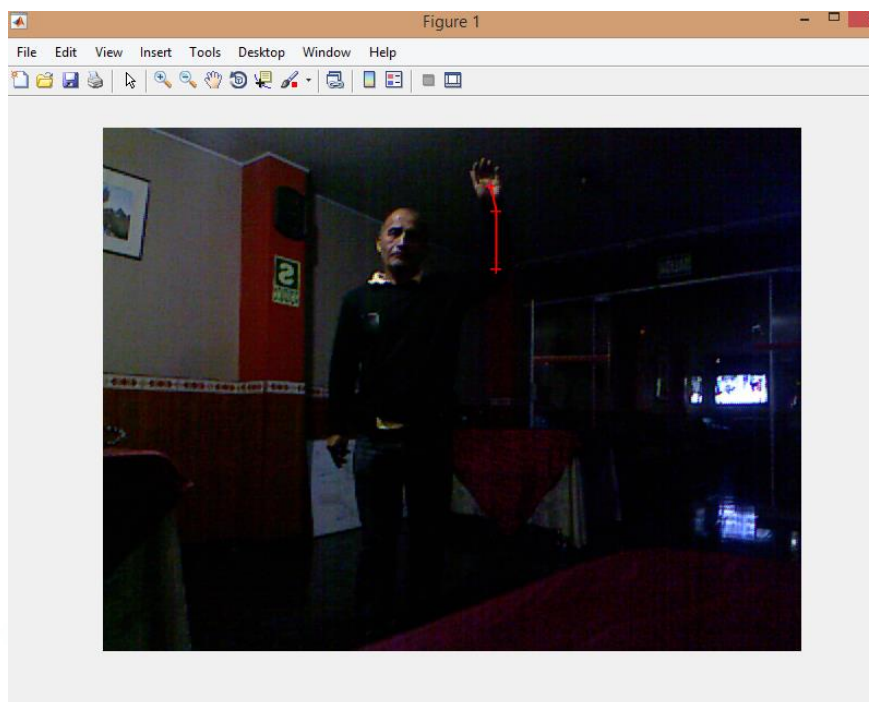


Figura 4.85: Imagen RGB y Skeleton captada por el sensor Kinect
Fuente: Propia

4.2.3.3. Logitech C920:

La cámara Logitech C920, es un dispositivo secundario que como ya se mencionó anteriormente, tiene por finalidad dar visibilidad al operador, acerca del espacio de trabajo del robot; lo que define la importancia de este elemento electrónico. Por este motivo, realizamos una prueba a fin de constatar su correcta operatividad, la misma que consiste en utilizar el puerto USB de la Laptop a modo que reconozca de forma automática el dispositivo en instale los drivers respectivos; para luego ejecutar la captura de una foto. Los resultados obtenidos fueron favorables. (Fig. 4.86)



Figura 4.86: Prueba de Funcionamiento de Logitech C920
Fuente: Propia

4.2.4. Correcciones y/o Modificaciones:

Dada la exposición del desacierto hallado en el punto 4.2.3.1, procedemos a llevar a cabo el desarrollo de la solución propuesta.

4.2.4.1. Incorporación de Hub:

Durante la realización de la prueba de operatividad de la pantalla LCD junto a la Raspberry Pi, se encontró un déficit de corriente por parte del procesador, provocando el no encendido del dispositivo de visualización; motivo por el cual se dispuso el uso de un concentrador o HUB de alimentación externa con 7 entradas USB para energizar la pantalla LCD sin afectar a la tarjeta Raspberry Pi; así mismo se aprovechan las entradas adicionales para alimentar también a los periféricos externos necesarios como el teclado y el mouse. (Fig. 4.87)



Figura 4.87: HUB
Fuente: Propia

Con el nuevo elemento, se llevó a cabo las conexiones respectivas a fin de verificar nuevamente el funcionamiento del procesador; para ello se inició sesión con el usuario PI y se creó un programa de prueba sencillo directamente en la interfaz de python, obteniendo así un resultado satisfactorio tal como se ve en la figura 4.88.

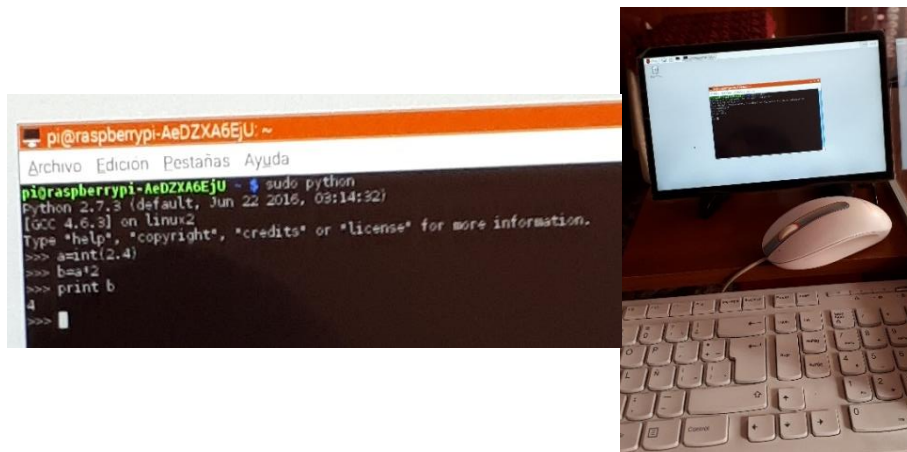


Figura 4.88: Prueba de funcionamiento Raspberry Pi y pantalla LCD

Fuente: Propia

Posteriormente y con el procesador en marcha, se conectó el sensor Kinect XBOX 360 al concentrador y de acuerdo a la información obtenida de la web se instaló a través de Python y el usuario root, el software OPEN CV en la tarjeta (Fig. 4.89), para luego intentar poner a prueba la cámara; sin embargo, el resultado obtenido fue desfavorable. Ante tal fallo se propone una nueva solución, la misma que es descrita en el apartado siguiente.



Figura 4.89: Instalación OPEN CV

Fuente: Propia

4.2.4.2. Cambio de Procesador:

Luego de varias pruebas e intentos realizados llegamos a la conclusión que la Raspberry Pi no tiene memoria suficiente para soportar el procesamiento de imágenes del sensor Kinect, siendo este de vital importancia para el proyecto; por esta razón optamos por usar una laptop con procesador Core i5, para utilizar así directamente el Matlab como software de procesamiento, y dejándole la función de tarjeta de adquisición de datos a la Raspberry pi.

- **Matlab**

Debido a que la instalación del sistema operativo de la Raspberry Pi se realizó desde Matlab es que se hizo más sencillo establecer comunicación entre ambos y adaptar el cambio descrito al proyecto; así mismo, ya que Matlab tiene incorporadas algunas

librerías en Simulink para la tarjeta Raspberry, se decidió modelar todo el sistema de control en dicha plataforma del software.

4.2.5. Armado Preliminar:

Establecido el diseño preliminar, se realizó el armado del circuito en un protoboard, teniendo en cuenta la continuidad de los cables a emplear y considerando el orden establecido tal como apreciamos en la figura 4.90.

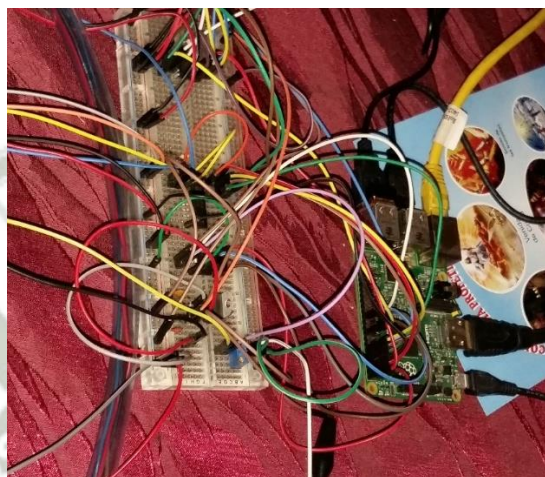


Figura 4.90: Armado preliminar del circuito

Fuente: Propia

4.2.6. Prueba de Funcionamiento del Circuito:

Se concretó la prueba de funcionamiento del circuito, conectando todos los dispositivos electrónicos que intervienen en la tesis a excepción del sensor Kinect y la cámara Logitech. (Fig. 4.91)

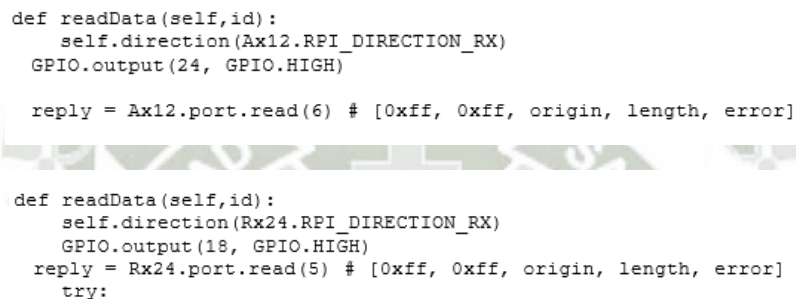


Figura 4.91: Conexión de elementos electrónicos para prueba de funcionamiento

Fuente: Propia

En cuanto al código de programación, se crearon cuatro softwares en Python, tres correspondientes a librerías, “ax12” (Anexo G), “rx24” (Anexo H), “ADC” (Anexo I) y el último al programa principal “SLIDERLEC” (Anexo J).

Ax12 y rx24 son códigos que albergan las funciones operativas en su totalidad de los motores dynamixel, fueron concebidos en base al manual dado por Robotis, el mismo que facilita todos los registros y la forma de envío e ingreso del paquete de datos. Ambos softwares son iguales con la única diferencia que en la configuración se detalla un pin diferente de la tarjeta Raspberry para que cumpla la labor de control en la transmisión y recepción de información, de tal manera que en el read data de ambos códigos los GPIO's varíen intermitentemente con la finalidad de evitar una confusión entre los paquetes ya que el funcionamiento de los motores será de forma simultánea. (Fig. 4.92)



```
def readData(self,id):
    self.direction(Ax12.RPI_DIRECTION_RX)
    GPIO.output(24, GPIO.HIGH)

    reply = Ax12.port.read(6) # [0xff, 0xff, origin, length, error]

def readData(self,id):
    self.direction(Rx24.RPI_DIRECTION_RX)
    GPIO.output(18, GPIO.HIGH)
    reply = Rx24.port.read(5) # [0xff, 0xff, origin, length, error]
    try:
```

Figura 4.92: Programación de los GPIO de Control de ambas librerías

Fuente: Propia

La librería ADC es un pequeño class, que tiene como tarea leer los bits enviados por el conversor MCP3202 correspondiente tanto a la posición del actuador Tower PRO como a la fuerza captada por el sensor de presión. Este código se inicializa con el canal a utilizar.

En cuanto al programa principal Sliderlec, podemos detallar que se trata de un código de programación que hace uso de las librerías anteriormente mencionadas para lograr su cometido, el mismo que consiste en dar movimiento por separado y a través de sliders, a cada uno de los servomotores que conforman el brazo robótico teniendo en cuenta sus límites de operación. Así mismo, este software nos permite no solo enviar datos sino también leer la posición de cada uno de los servomotores, por lo tanto, para ambas funciones fue necesario hacer una conversión de grados sexagesimales a resolución en el caso de los Dynamixel y a milisegundos en el caso del Tower Pro.

Definido tanto el hardware como el software, procedemos a llevar a cabo la prueba de funcionamiento propiamente dicha, en el que se obtuvo un buen resultado durante los primeros ensayos; sin embargo, momentos posteriores después de la realización de varios movimientos se pudo notar que los servomotores Rx24F a pesar de que desarrollaban su labor de manera correcta, emitían un pequeño ruido, es por esta razón que se vio por conveniente y se tomó la decisión de alimentar estos actuadores con una

fuelle regulada adicional de 0 a 12V y con la capacidad de entrega de corriente de hasta 5A. Finalmente, dicha medida nos permitió solucionar el conflicto. (Fig. 4.93)

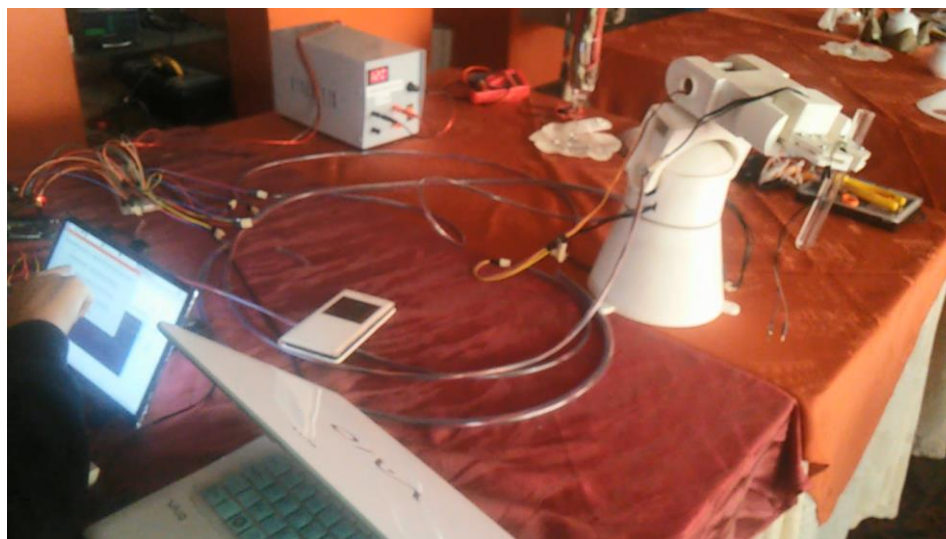


Figura 4.93: Prueba de Funcionamiento del Robot

Fuente: Propia

4.2.7. Diseño e impresión de Placa:

Habiéndose realizado todas las pruebas de los componentes tanto individual como en conjunto y analizado los posibles fallos con sus respectivas correcciones, se determinó elaborar el diseño final de la placa con los circuitos a utilizar en este proyecto; tarea que se realizó gracias al uso del software Eagle que nos permitió diseñar (Fig. 4.94) y quemar las pistas para luego soldar cada elemento a la placa (Fig. 4.95).

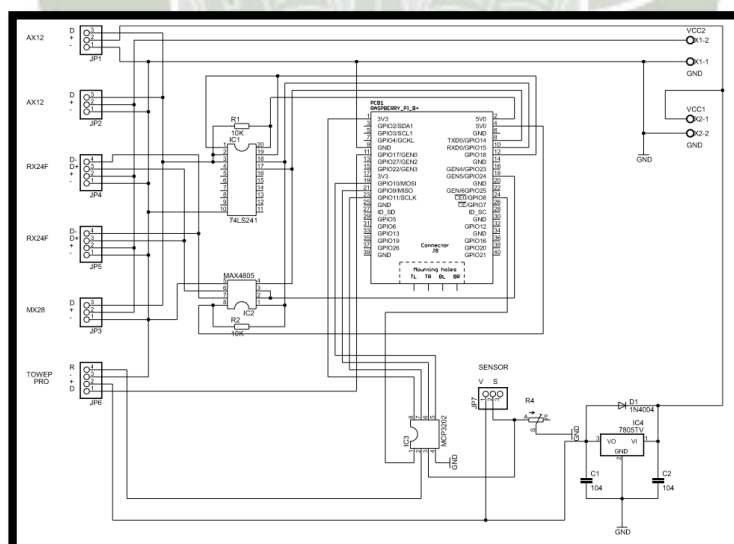


Figura 4.94: Diseño del circuito en Eagle

Fuente: Propia

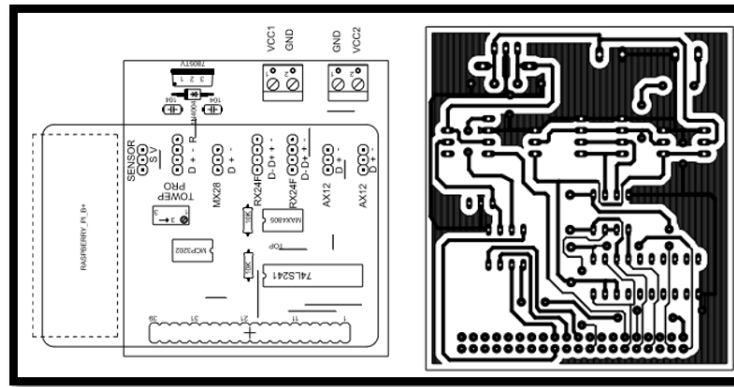


Figura 4.95: Diseño del circuito impreso y pistas en placa electrónica de cobre
Fuente: Propia

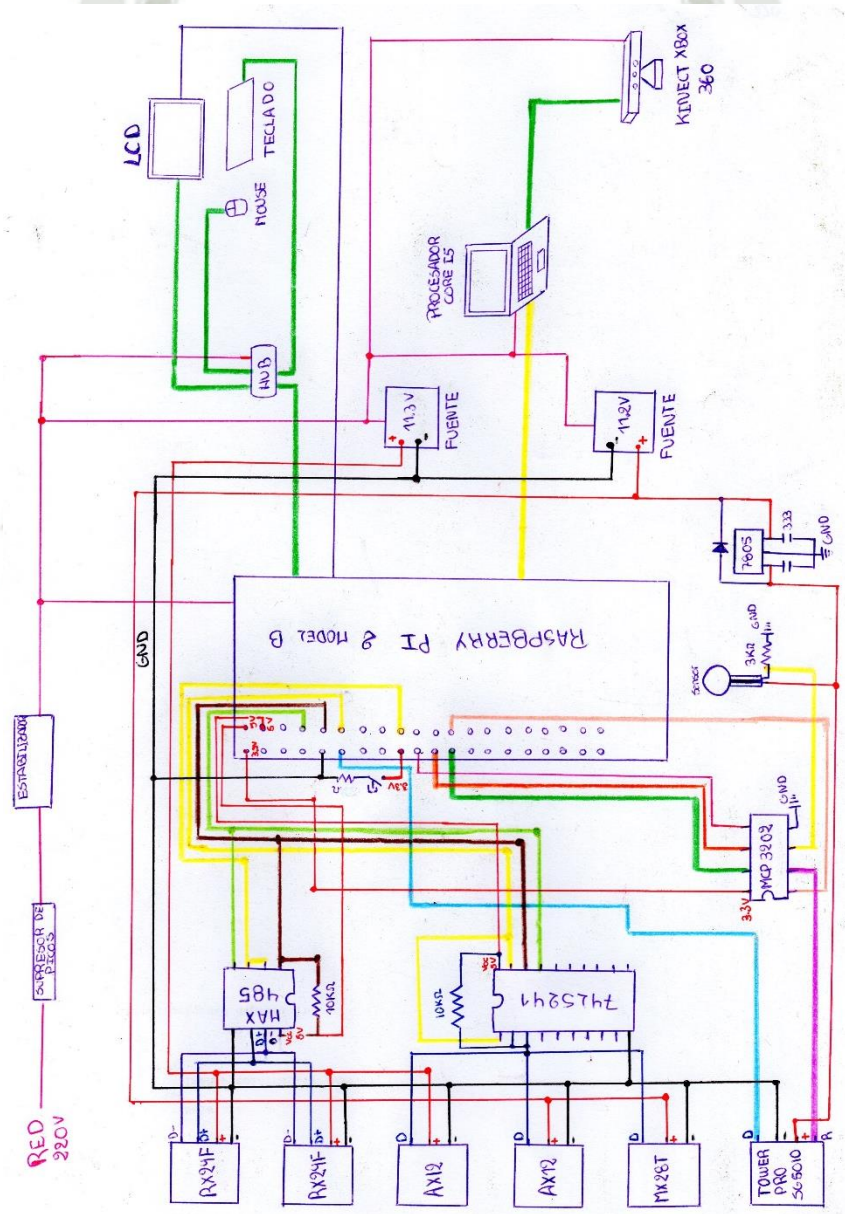


Figura 4.96: Conexión y Disposición de los Componentes Electrónicos
Fuente: Propia

CAPÍTULO V: CONTROL E INTERFAZ GRAFICA DE USUARIO

5.1. SimMechanics:

Es una herramienta que nos permite modelar y simular sistemas mecánicos de manera muy sencilla y eficiente en la plataforma de Simulink. El estudio de los sistemas mecánicos puede ser realizado desde dicha interfaz gráfica.

Para nuestro caso utilizaremos el diseño del manipulador robótico generado en el software Solidwork 2013 y lo guardaremos con la extensión .xml; de esta forma podremos usar el comando `mech import('')` en Matlab para importar nuestro diseño en bloques a la plataforma de Simulink (Fig. 5.1)

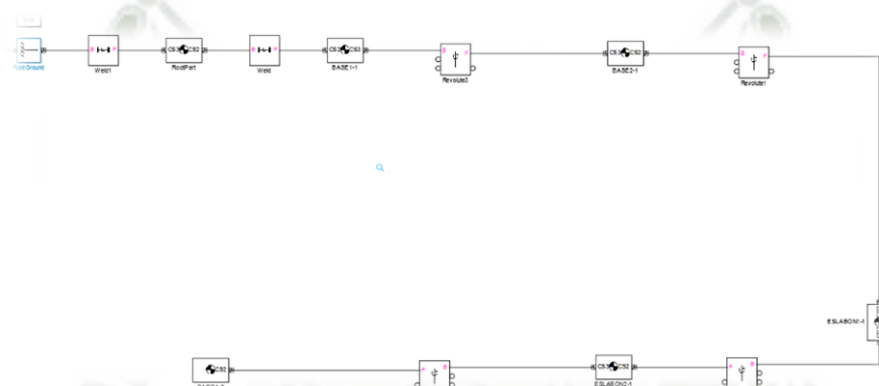


Figura 5.1: SimMechanics aplicado al manipulador robótico en simulink
Fuente: Propia

Seguidamente se agregó los bloques: “Joint Initial Condition”, “Joint Actuator” y “Joint Sensor”; para simular las condiciones iniciales, los actuadores y sensores del robot. (Fig. 5.2)

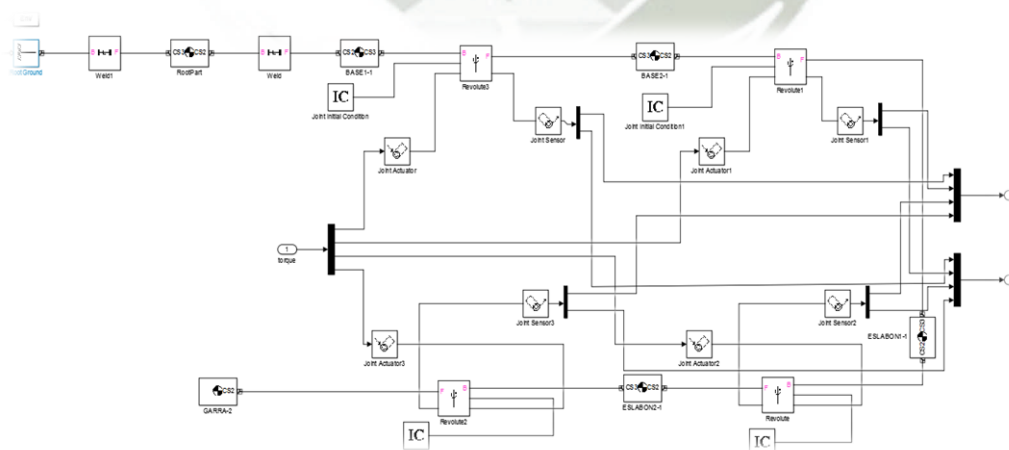
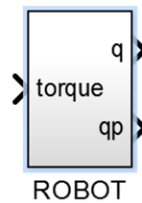


Figura 5.2: Diagrama completo del Robot simulado
Fuente: Propia

Por último, se encapsuló todo en un subsistema denominado “ROBOT”, tal como se puede ver en la figura 5.3, para usos posteriores.

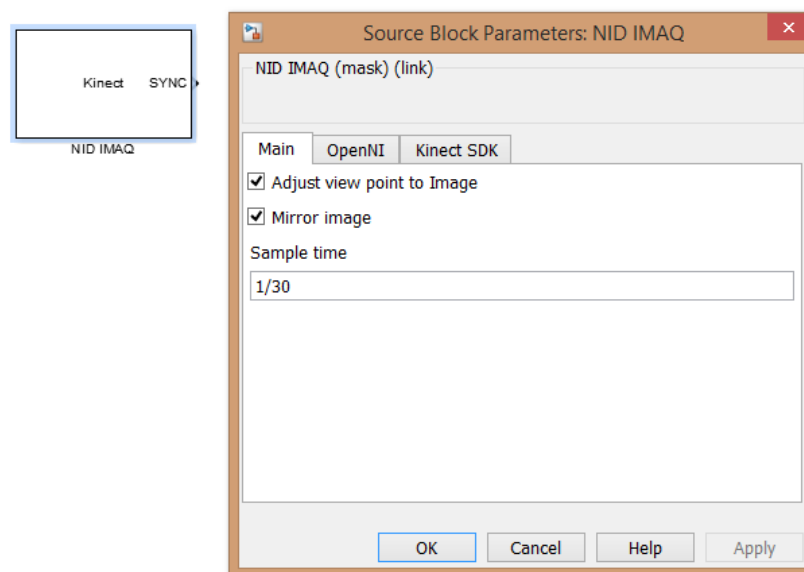
**Figura 5.3:** Subsistema ROBOT*Fuente: Propia*

5.2. Modelamiento Kinect:

El modelamiento Kinect tiene por finalidad determinar el set point en el plano cartesiano que se necesita como entrada para el control cinemático del manipulador robótico, a partir de la información del movimiento del brazo del operador captada por el sensor. Para ello, se crearon 7 funciones y se hizo uso de ciertos bloques de la librería proporcionada por la plataforma Simulink de Matlab, los mismos que son detallados a continuación:

- **NID IMAQ:**

Es el bloque de inicio que nos permite definir algunos parámetros de configuración del Sensor Kinect tales como: activación de la imagen espejo, tiempo de muestreo considerando un valor de 0.33s como el adecuado, la resolución de la imagen de 640 x 480 a 30 frames por segundo, activación de la opción “Inferir la posición de las articulaciones del esqueleto” y el ángulo del sensor, cuyo valor es de 4° considerando la ubicación del Kinect. (Fig. 5.4)

**Figura 5.4:** Bloque NID IMAQ y ventana de configuración*Fuente: Propia*

- **NID IMAGE Y VIDEO VIEWER:**

Bloques para determinar y mostrar la imagen percibida a color. (Fig. 5.5)

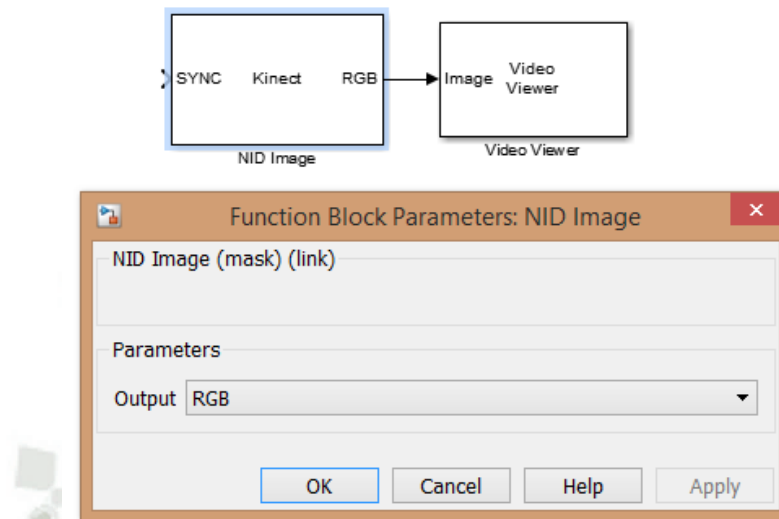


Figura 5.5: Bloque NID IMAGE, VIDEO VIEWER y ventana de configuración
Fuente: Propia

- **NID SKELETON:**

Constituye uno de los bloques principales, puesto que en él se especifica la cantidad de esqueletos a percibir y el sistema de coordenadas de las articulaciones. (Fig. 5.6)

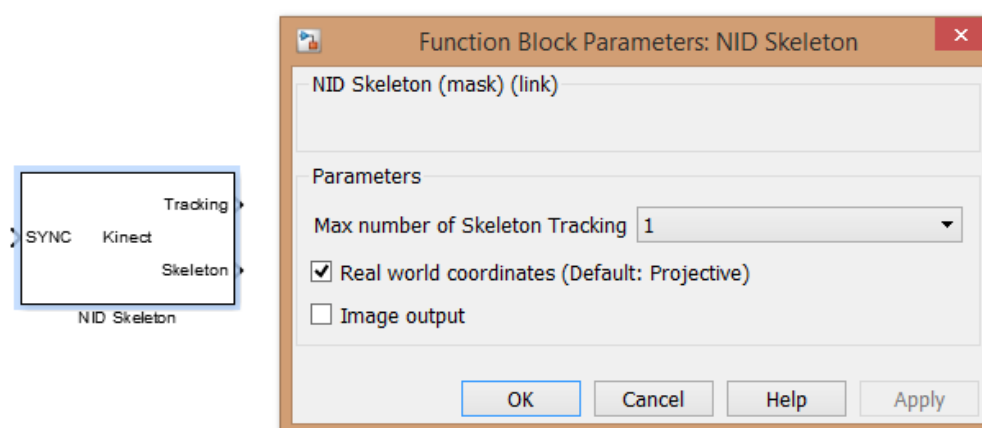


Figura 5.6: Bloque NID Skeleton y ventana de configuración
Fuente: Propia

- **Hand Viewer:**

Bloque que permite visualizar el esqueleto del operador y que fue modificado a fin de resaltar únicamente las extremidades superiores, las mismas que son utilizadas para dar movimiento al brazo robótico. (Fig. 5.7)



Figura 5.7: Bloque Hand Viewer
Fuente: Propia

- **Funciones para la adquisición de Coordenadas correspondientes a las articulaciones:**

Se crearon 6 bloques de funciones que como ya su nombre lo indica están destinadas a obtener las coordenadas de las articulaciones de las extremidades superiores del operador captado por el sensor, referidas específicamente al hombro derecho, codo derecho e izquierdo, muñeca derecha e izquierda, y mano izquierda. Todas ellas son ordenadas en una matriz 6x3 para su posterior procesamiento. (Fig. 5.8)

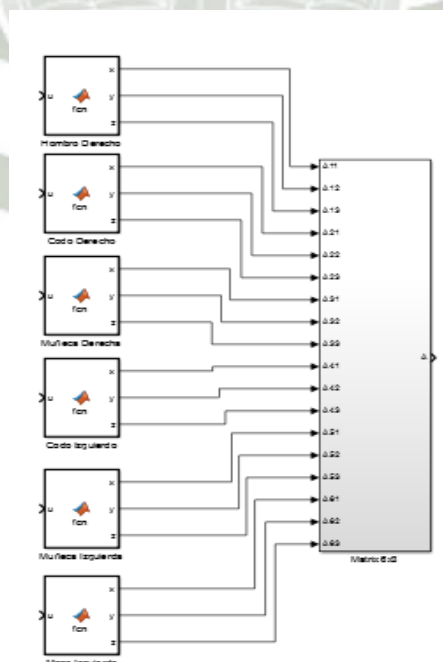


Figura 5.8: Funciones para la adquisición de Coordenadas articulares
Fuente: Propia

- **Función Transformar:**

Corresponde a uno de los bloques más importantes del modelamiento junto al NID Skeleton; ya que básicamente es aquí donde se realiza el procesamiento en su totalidad de las coordenadas obtenidas por el sensor para transformarlas en coordenadas cartesianas capaces de ser interpretadas por las redes neuronales en el control cinemático del manipulador robótico.

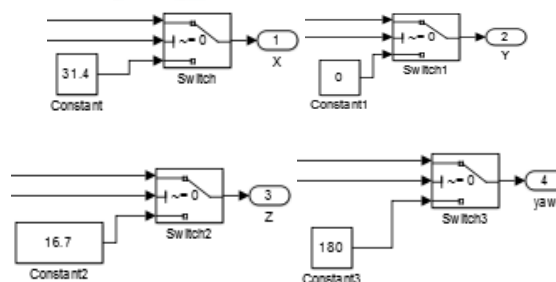
En este proceso se consideró la extremidad superior derecha para hallar la posición cartesiana del robot (x,y,z) y la extremidad superior izquierda para hallar la rotación y dar movimiento de apertura y cierre al efector final. (Fig. 5.9)



Figura 5.9: Funciones Transformar
Fuente: Propia

- **Bloques de posición Inicial:**

Nos permiten definir la posición inicial del brazo robótico en caso el sensor no capte operador alguno. Los valores seteados son x:31.4, y:0, z:16.7, yaw:180 y garra:0. (Fig. 5.10)



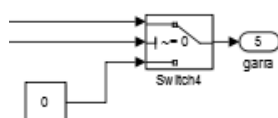


Figura 5.10: Bloques de posición Inicial
Fuente: Propia

Finalmente, las figuras 5.11 y 5.12 muestran la conexión de cada uno de los bloques ya detallados y la prueba de funcionamiento del subsistema, para posteriormente crear un solo bloque de aplicación posterior (Fig. 5.13).

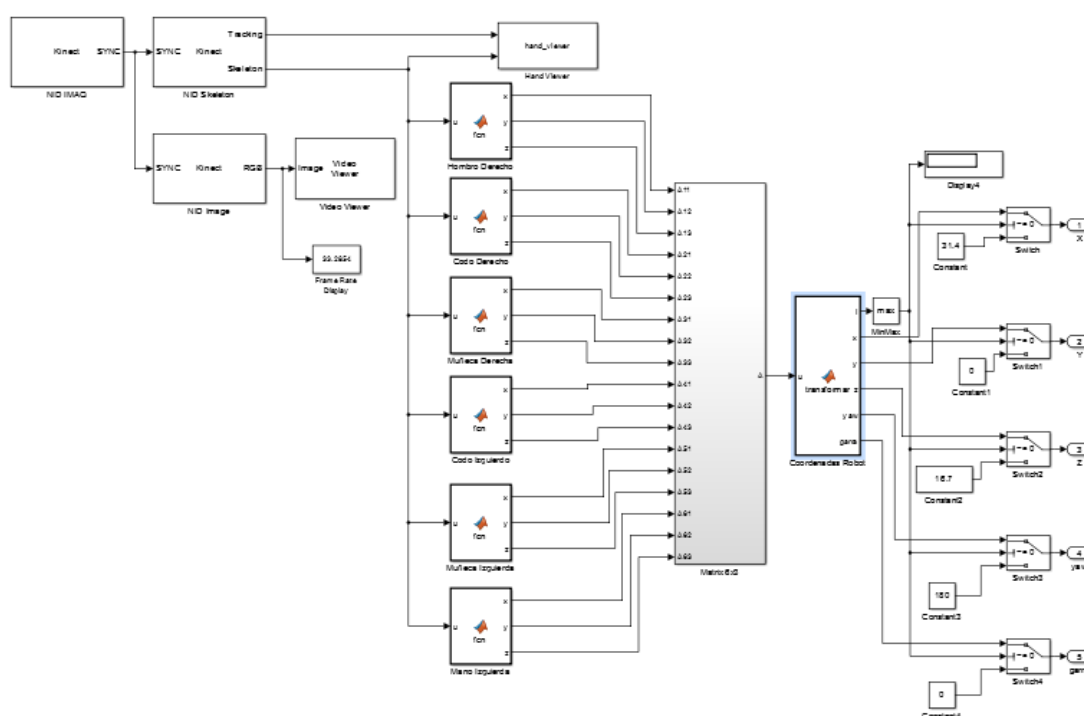
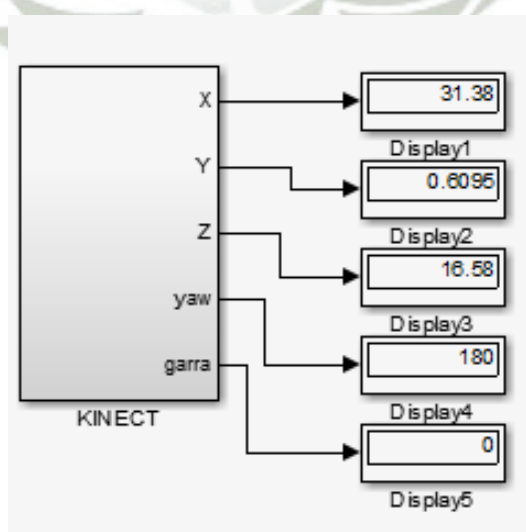
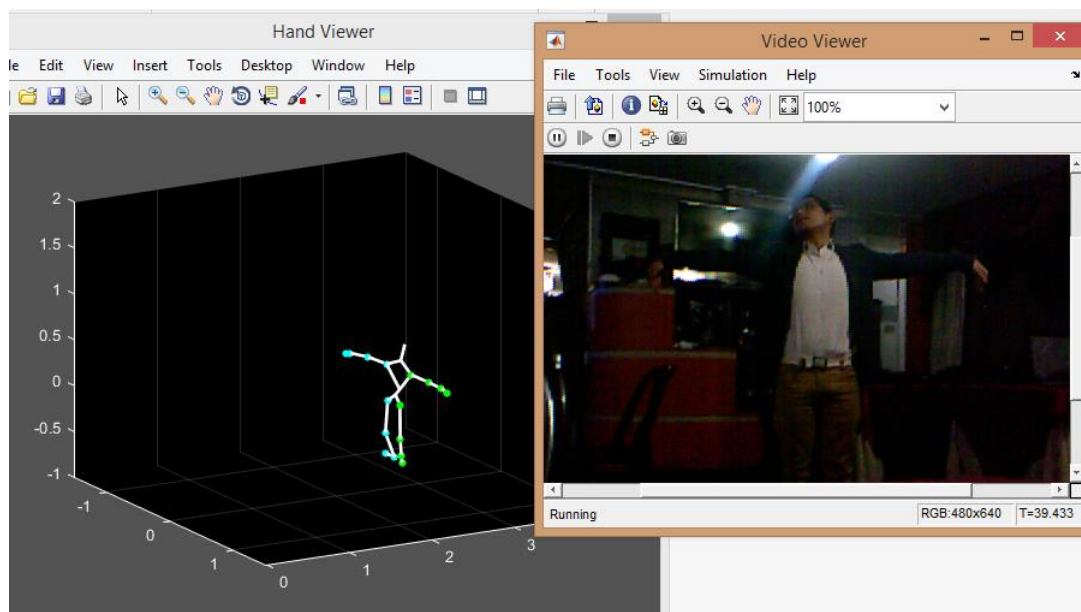


Figura 5.11: Subsistema Kinect
Fuente: Propia




Figura 5.12: Prueba de Funcionamiento

Fuente: Propia

Figura 5.13: Encapsulado

Fuente: Propia

5.3. Control Cinemático: Redes Neuronales:

En este proyecto las redes neuronales se usarán para aprender la cinemática inversa del manipulador robótico y poder sustituir dicha operación matemática, demostrando la efectividad de su uso ante modelos no lineales complejos. La manera más sencilla de abordar el aprendizaje de la cinemática inversa es aplicar el esquema de modelización inversa directa a una red multicapa que incorpore la regla de retropropagación del error.

Al ser esta una red supervisada, es necesario contar con un conjunto de entrenamiento, en dicho conjunto los puntos a utilizar deben ser representativos del problema que se intenta solucionar, ya que si se utilizar menos puntos de los necesarios, la red será incapaz de aprender. Y en el caso de contar con un conjunto de entrenamiento con demasiados puntos, la red necesitará un tiempo excesivo para su entrenamiento y en el peor de los casos, su entrenamiento no convergerá.

Teniendo esto en cuenta primero dividiremos el espacio de trabajo del robot en representaciones locales, que para nuestro caso vimos por conveniente que para los tres

primeros grados de libertad fueran nueve regiones: ocho zonas externas y una zona interna; y para el último grado de libertad solo sean cinco zonas externas tomando en cuenta que su movimiento no es tan variable como en el caso de los otros. Esta resolución se debe a que el alto nivel de no linealidad del problema impide resolverlo mediante una única red neuronal.

Dicho esto, se procedió a crear un script en Matlab para graficar y obtener una cantidad de puntos en cada zona tal que permita generar un entrenamiento adecuado buscando una eficiencia combinada de por lo menos 95%; para cada red se obtuvo una cantidad de 13026 datos teniendo como datos de entradas la posición y orientación del efector final y como datos de salida los valores articulares en radianes correspondientes a dichas posiciones.

Obtenidas y ordenadas las bases de datos se hizo uso de la plataforma de Matlab para redes neuronales llamada “Neural Network ToolBox” (Fig. 5.14), que nos permite crear una red neuronal con diferentes parámetros para luego entrenarla todo esto de manera intuitiva gracias a esta plataforma.

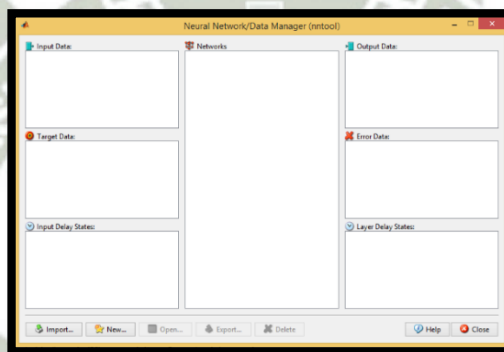


Figura 5.14: Neural Network ToolBox
Fuente: Propia

Para la creación y entrenamiento de cada red se siguió los siguientes pasos:

5.3.1. Ingreso de Datos:

Se agregó los datos de entrada y de salida para luego crear la red con la opción “New” tal como se puede apreciar en la siguiente figura.

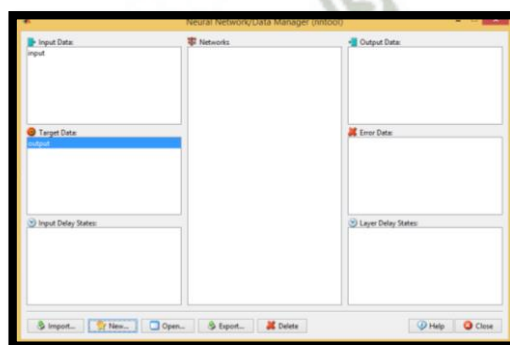


Figura 5.15: Asignación de datos
Fuente: Propia

5.3.2. Creación de la Red:

Nos aparecerá una ventana para configurar los parámetros de la red a crear, elegimos el nombre de la red en la opción “Name”, el tipo de red es “feed-forward backpropagation” que traducido al español es retropropagación del error, los valores de entrada y salida se obtendrá de la base de datos, la función de entrenamiento es “traingdm (Gradient descent with momentum backpropagation)” que traducido al español se denomina Descenso de gradiente de lotes por Momentum, tiene como característica principal una convergencia más rápida y la función de adaptación es LEARNGDM. La red consta de 3 capas ocultas con funciones de transferencia sigmoidales. La primera capa posee 35 neuronas, la siguiente 25 neuronas, luego una capa de 15 neuronas y por último una capa de salida de 3 neuronas. La red admite como entrada la posición y orientación deseada del manipulador (X , Y , Z , ψ) y entrega como salida cada uno de los ángulos requeridos en las articulaciones (q_1 , q_2 , q_3 , q_4) a efectos de alcanzar la posición deseada.

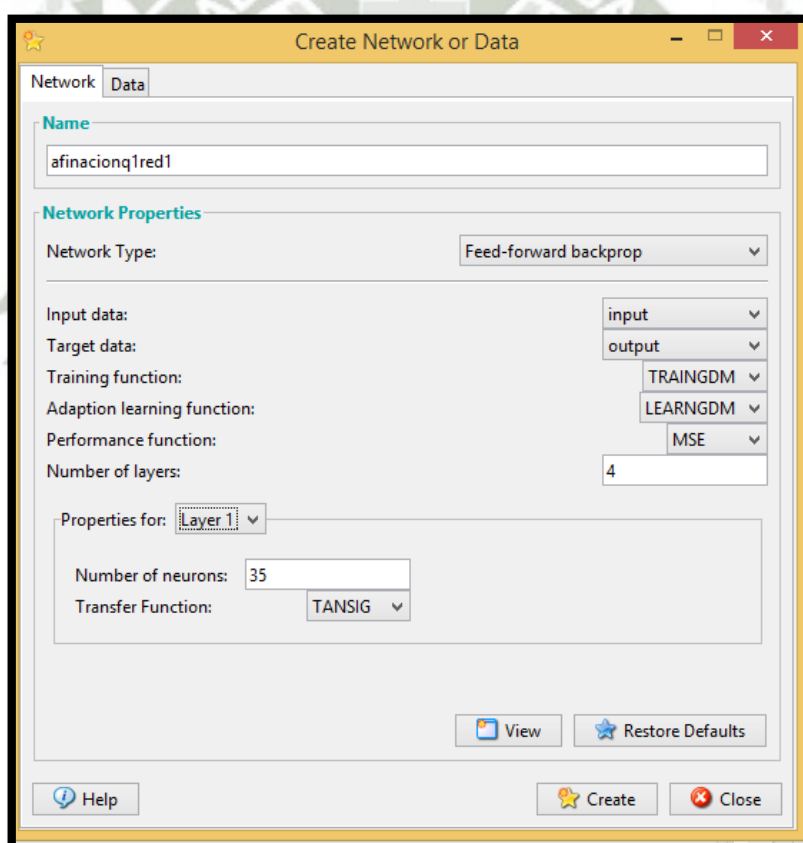


Figura 5.16: Asignación de datos

Fuente: Propia

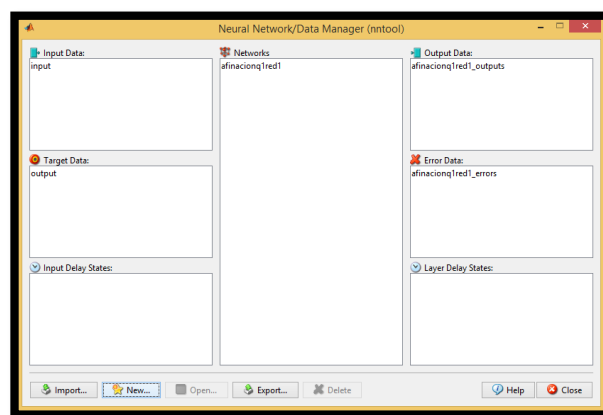


Figura 5.17: Creación de la Red Neuronal

Fuente: Propia

5.3.3. Entrenamiento de Red:

Para realizar el entrenamiento ingresamos a la red creada anteriormente y configuramos los parámetros, de los cuales los más importantes son los siguientes: el número de épocas, la tasa de aprendizaje (α) y la constante de momento (μ); datos que se obtienen experimentalmente y según sea el problema, sin embargo tenemos un rango de valores de donde uno puede partir para encontrar el valor que se adapte mejor a su aplicación (Fig. 5.18).

Mastering ANN Parameters		
	Typical	Range
learning rate - α	0.1	0.01 - 0.99
momentum - μ	0.8	0.1 - 0.9
weight-cost - λ	0.1	0.001 - 0.5
Fine tuning : - adjust individual parameters at each node and/or connection weight - automatic adjustment during training		

Figura 5.18: Rango de valores

Fuente: "Diseño de Redes Neuronales Multicapa y Entrenamiento", 2009

De esta manera y tras varias pruebas se encontró que los valores de α y μ que se ajustaban mejor para nuestro proyecto eran 0.1 y 0.65 respectivamente; y referente al número de épocas es 50000. (Fig. 5.19)

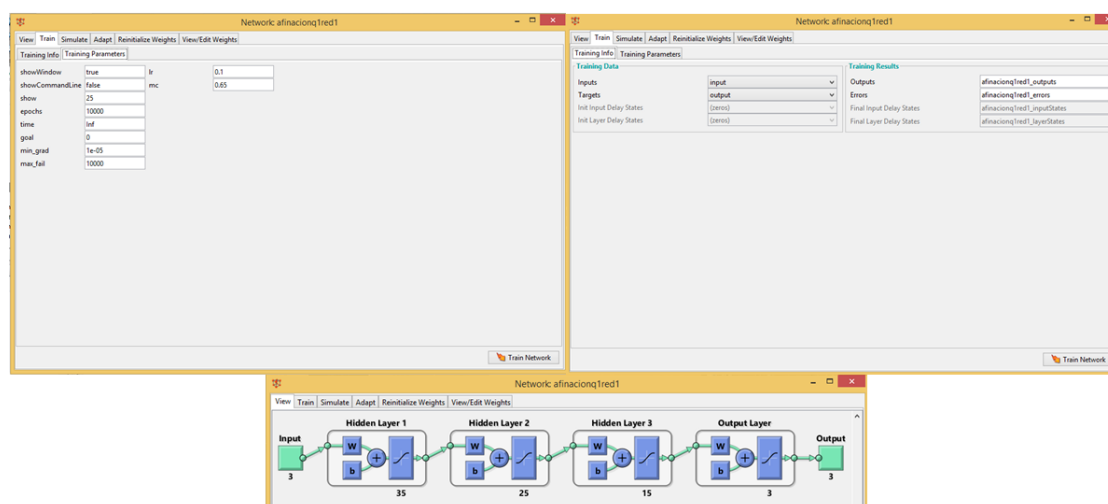


Figura 5.19: Parámetros de entrenamiento

Fuente: Propia

Finalizado esto se procede al entrenamiento mediante el uso del botón “Train Network” y comenzará el entrenamiento. (Fig. 5.20)

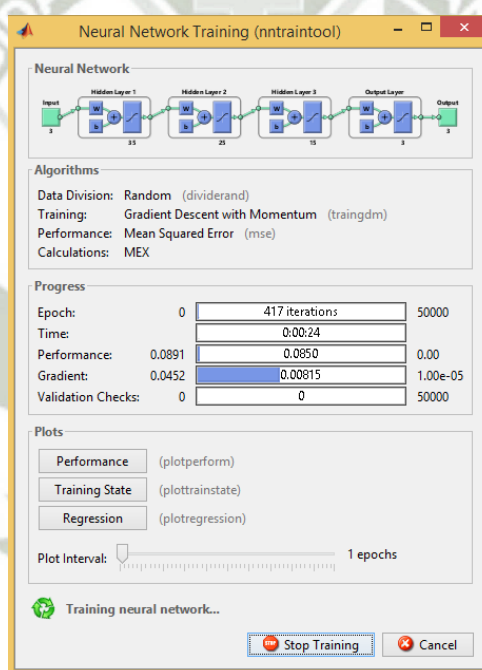


Figura 5.20: Proceso de entrenamiento

Fuente: Propia

5.3.4. Resultados del Entrenamiento:

Una vez iniciado el entrenamiento tomará un tiempo según el número de capas ocultas y épocas que se halla elegido, terminado este proceso tendremos acceso a tres ventanas para analizar los resultados: “Performance”, “Training State” y “Regression”

- **Performance:**

Traza el error vs la época para el entrenamiento, la validación, y el rendimiento de las pruebas del registro de entrenamiento (Fig. 5.21)

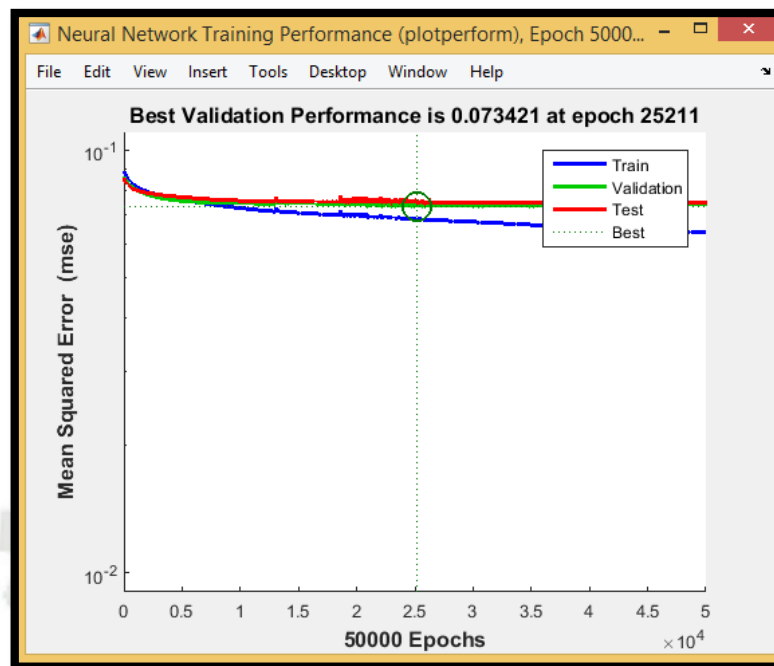


Figura 5.21: Performance: Error vs Épocas

Fuente: Propia

- **Training State:**

Traza el estado de entrenamiento de un registro de entrenamiento devuelto por el traingdm. (Fig. 5.22)

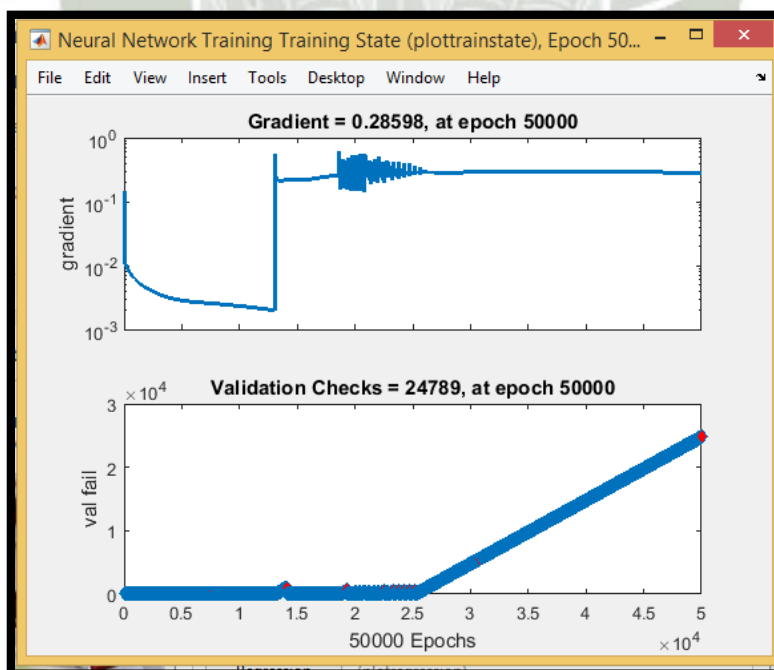


Figura 5.22: Estado de entrenamiento

Fuente: Propia

- **Regression:**

Traza la regresión lineal de los objetivos en relación con los resultados. (Fig. 5.23)

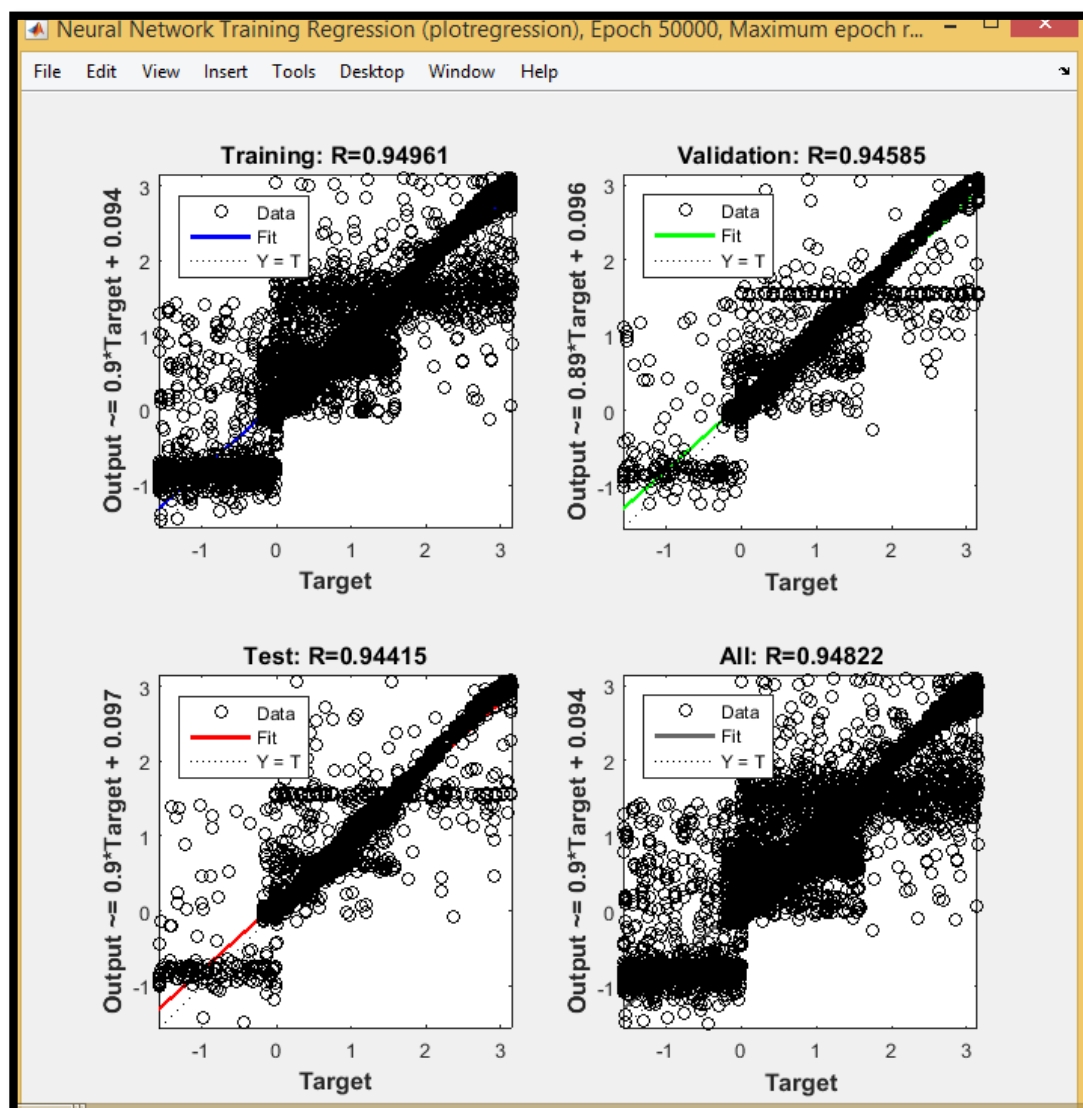


Figura 5.23: Regresión

Fuente: Propia

Los mismos pasos se realizaron para la creación de las demás redes para cada articulación de tal forma que se obtuvieron los resultados mostrados en la Tabla 5.1.

Tabla 5.1: Resumen de Entrenamiento de Redes

q1	estructura	Rend. (%)	lr	mc	Épocas	Cantidad Datos
Afinaciónq1red1	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red2	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red3	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red4	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red5	4-35-25-15-1	96.8	0.1	0.65	50000	13026

Afinaciónq1red6	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red7	4-35-25-15-1	96.8	0.1	0.65	50000	13026
Afinaciónq1red8	4-35-25-15-1	96.8	0.1	0.65	50000	13026
q2	Estructura	Rend. (%)	lr	mc	Épocas	Cantidad Datos
Afinaciónq2red1	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red2	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red3	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red4	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red5	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red6	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red7	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
Afinaciónq2red8	4-64-32-16-8-4-1	98.1	0.1	0.65	50000	13026
q3	estructura	Rend. (%)	lr	mc	Épocas	Cantidad Datos
Afinaciónq3red1	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red2	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red3	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red4	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red5	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red6	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red7	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
Afinaciónq3red8	4-64-32-16-8-4-1	97.8	0.1	0.65	50000	13026
q4	estructura	Rend. (%)	lr	mc	Épocas	Cantidad Datos
Afinaciónq4red1	4-35-25-15-1	99	0.1	0.65	50000	13026
Afinaciónq4red2	4-35-25-15-1	99	0.1	0.65	50000	13026
Afinaciónq4red3	4-35-25-15-1	99	0.1	0.65	50000	13026
Afinaciónq4red4	4-35-25-15-1	99	0.1	0.65	50000	13026
Afinaciónq4red5	4-35-25-15-1	99	0.1	0.65	50000	13026

Fuente: Propia

Con todas las redes entrenadas y guardadas en Matlab se procede a probarlas dentro de simulink para lo cual se usa el comando gensim que genera un bloque de simulink para la simulación de una red neural (Fig. 5.24); se añade un bloque seleccionador al programa que tiene como función elegir la red neuronal según la zona donde se encuentra el robot en ese instante. De esta manera se puede probar las redes creadas y observar su rendimiento (Fig. 5.25).

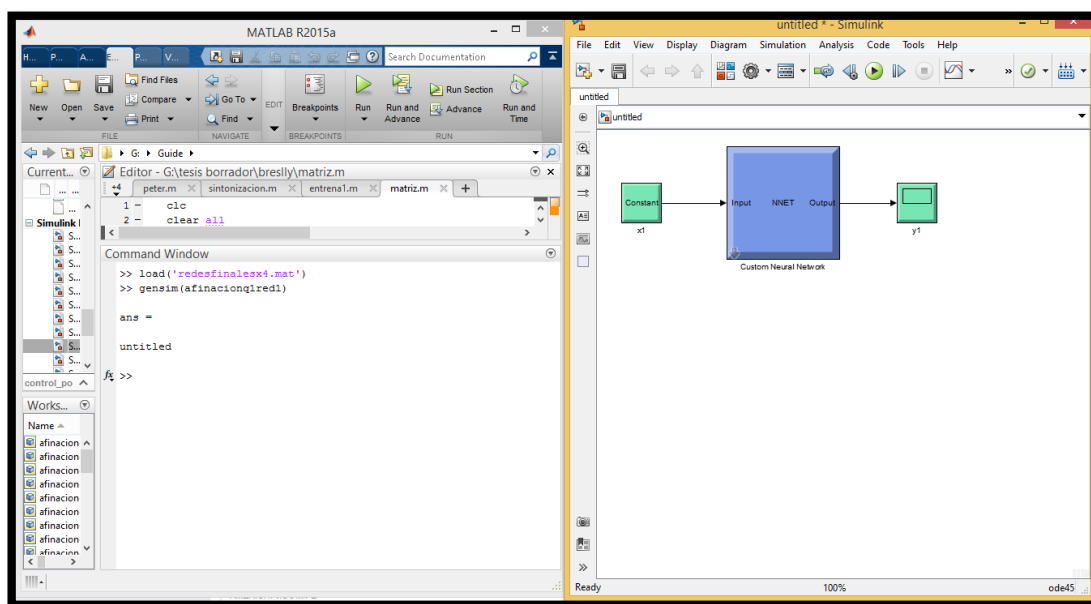


Figura 5.24: Uso del comando “Gensim”

Fuente: Propia

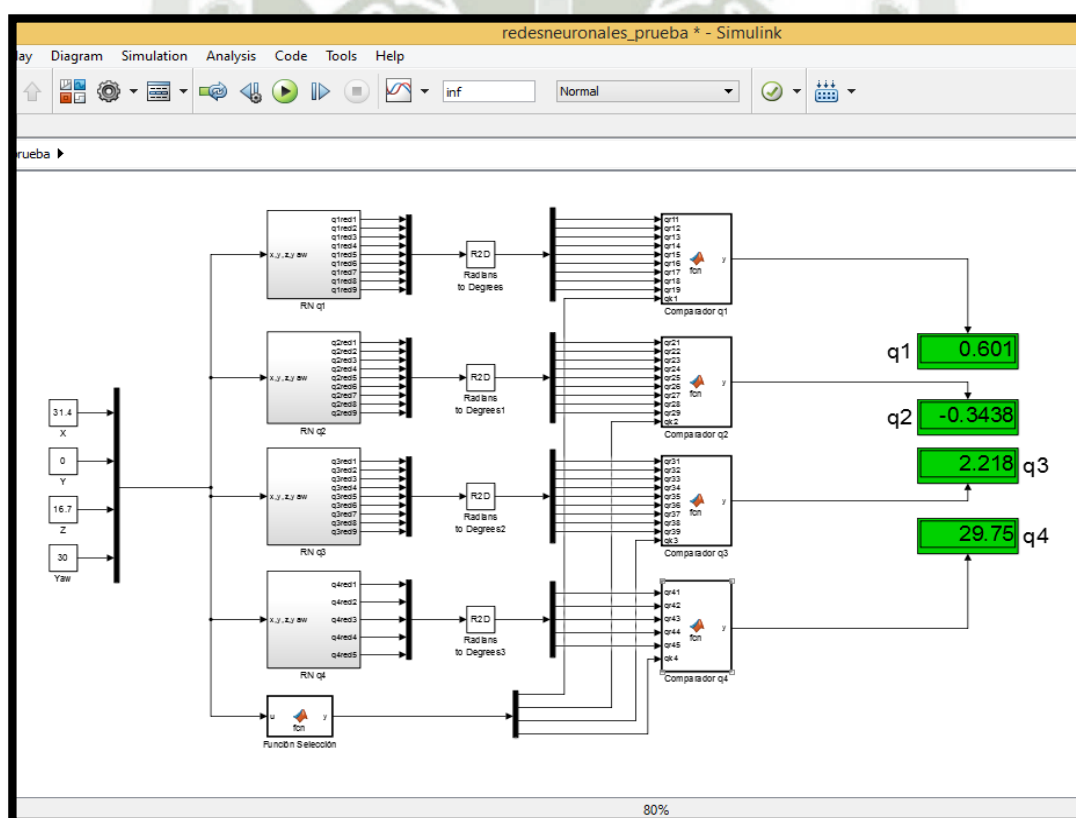


Figura 5.25: Programa de prueba de las Redes Neuronales

Fuente: Propia

Por último, se reduce estos bloques a un subsistema que le denominaremos RN q1, q2, q3, q4. (Fig. 5.26)

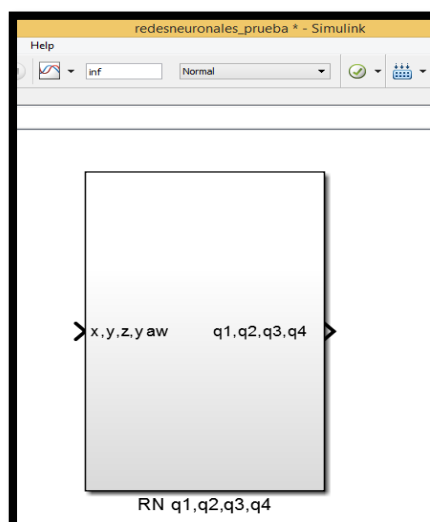


Figura 5.26: Subsistema RN q1, q2, q3, q4

Fuente: Propia

5.4. Comunicación entre Matlab y Raspberry Pi:

Una parte importante del proyecto es la comunicación entre el software Matlab y la tarjeta de adquisición de datos, Raspberry Pi, debido a que se tiene como finalidad que el manipulador robótico opere en un tiempo considerado real. Es por esta razón que se vio por conveniente usar los bloques “UDP Recieve” y “UDP Send” que nos ofrece la librería de Raspberry Pi en Matlab (Fig. 5.27)

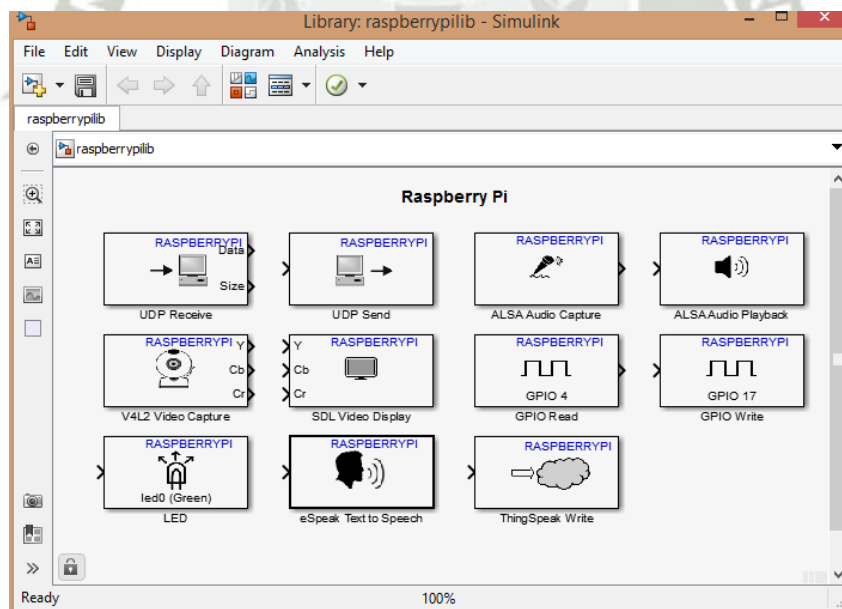


Figura 5.27: Librería Raspberry PI

Fuente: Propia

El protocolo UDP no requiere mucho procesamiento, proporciona una sencilla interfaz entre la capa de red y la capa de aplicación; pero la ventaja que nos llevó a elegir este protocolo es que tiene muchas aplicaciones en procesos en tiempo real, característica que se adecuaba a nuestro proyecto.

Para su funcionamiento tenemos que establecer la comunicación mediante un cable Utp entre la computadora y el Raspberry Pi (Fig. 5.28) y crear tanto el programa de envío y recepción entre dichos componentes.

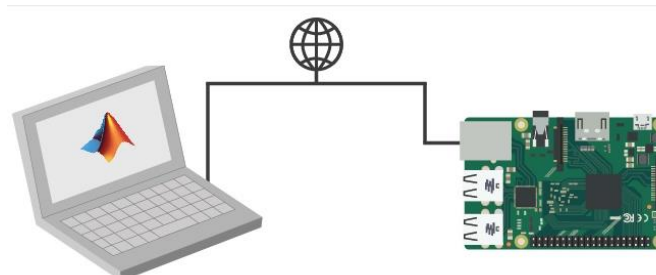


Figura 5.28: Comunicación entre la PC y Raspberry

Fuente: Propia

5.4.1. Computadora:

Para la comunicación utilizamos el software Matlab mediante su plataforma Simulink. Utilizamos los bloques ya mencionados en el apartado anterior y los configuraremos según la cantidad de datos que necesitemos enviar y recibir (Fig. 5.29)

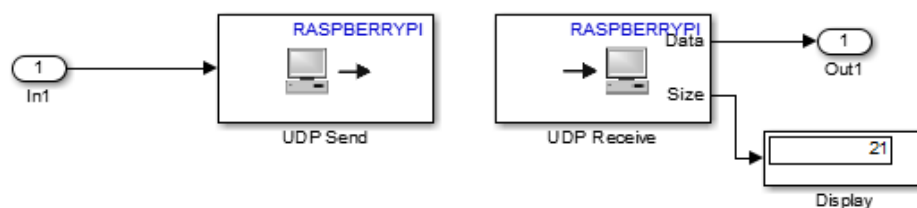


Figura 5.29: Bloques de envío y recepción de datos

Fuente: Propia

En el caso del bloque “UDP Send” tan solo hay que conocer la dirección IP remota que para nuestro caso por tener una única conexión LAN utilizaremos el Broadcast; y el puerto IP remoto, en este caso el del Raspberry Pi con el que deseamos comunicarnos.

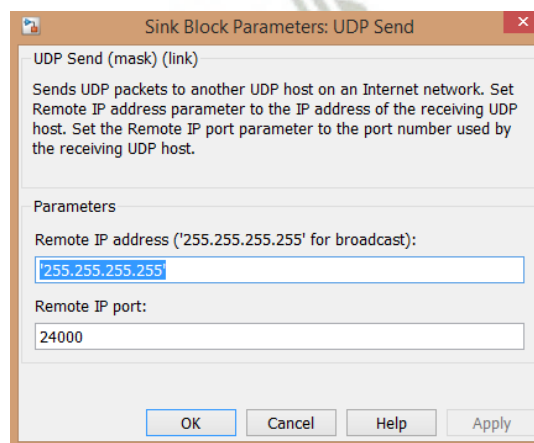


Figura 5.30: Parámetros de configuración del bloque UDP Send

Fuente: Propia

Para configurar el “UDP Receive tenemos que conocer nuestro propio puerto de salida, el tipo de dato que deseamos recibir que, para nuestro caso sería int8, es decir, enteros de 8 bits; el tamaño del dato, que serían 21 elementos que se recibirán desde el Raspberry Pi, 3 elementos por cada dato recibido. Esto es porque cada dato que recibiremos cuenta con 3 dígitos como por ejemplo si el valor articular q1 fuera 95, tendría que ser enviado como un conjunto de 3 elementos, es decir; “0”, “9” y “5”. Los datos que se recibirán son los valores articulares: q1, q2, q3, q4, garra, el valor del sensor de fuerza, y el estado del robot. Por último, tenemos que encontrar el valor más adecuado para el tiempo de muestreo, se debe tomar en cuenta que entre más tareas de procesamiento se agreguen este tiempo de muestreo se tendrá que volver a variar. El valor que nos permite una sincronización adecuada es 0.0075 segundos y fue hallado de manera empírica luego de varias pruebas.

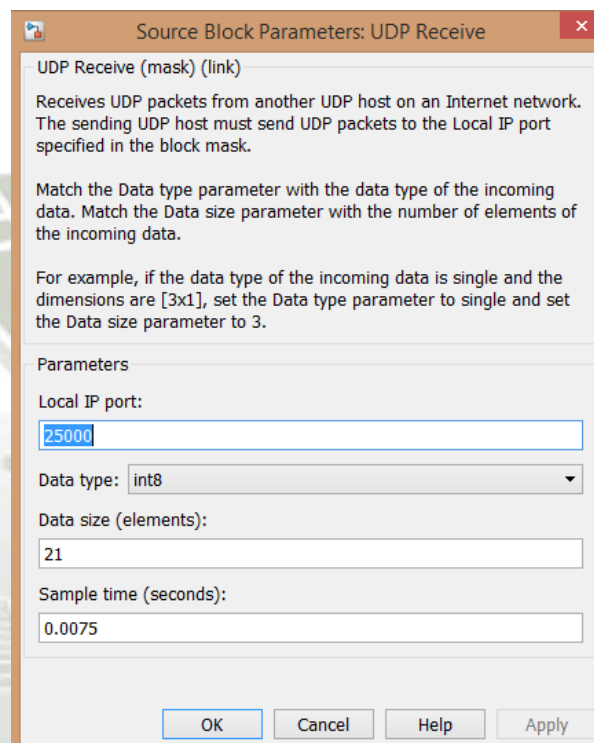


Figura 5.31: Parámetros de configuración del Bloque UDP Receive

Fuente: Propia

5.4.2. Raspberry Pi:

Así como se desarrolló en Simulink mediante bloques un programa de recepción y envío de datos, se creará en Python un programa que actúe de la misma manera, para esto haremos uso de la librería llamada “Socket” que tiene la misma función que los bloques UDP de Simulink.

Para el envío de datos utilizamos las siguientes líneas de código que al igual que en el bloque “UDP Send” únicamente tenemos que colocar la dirección IP remota y el puerto IP remoto, que en este caso sería de la computadora. (Fig. 5.32)

```

-----
#ENVIO DE DATOS DE LECTURAS A MATLAB DESDE RASPBERRY A PC
#INGRESAR EL DATO EN STRING
dato=c+e+h+j+l+n
PACKETDATA = dato
print dato

t = socket(AF_INET, SOCK_DGRAM)
t.connect(('169.254.139.155', 25000))

# enviar el paquete
t.send(PACKETDATA)

# cerrar socket
t.close()

```

Figura 5.32: Programación en Python de envío de datos hacia la PC

Fuente: Propia

Referente a la recepción de datos se programaron las siguientes líneas de código, tenemos un comando llamando “gethostbyname” que nos devuelve nuestro propio IP y el 24000 es nuestro Puerto IP local; en este caso los datos que son enviados desde Simulink hacia el Raspberry Pi vienen en paquetes, por lo cual utilizamos el comando “Struct.Struct(‘6d’)” para que cada paquete cuente con 6 datos y luego el comando “unpack(data)” para desempaquetar el paquete de datos y guardarlos en la variable “data”. Posteriormente dividiremos dicha variable en 6 partes que serán asignadas a las siguientes variables: “q1”, “q2”, “q3”, “q4”, “garra”, “error”.

```

#
#RECEPCION DE DATOS DE KINECT VIA UTP DESDE PC [5 DATOS]
#
#
#
#COMUNICACION
mySocket=socket(AF_INET,SOCK_DGRAM)
mySocket.bind((gethostbyname('0.0.0.0'),24000))
#-----
repeat=True
if repeat:|
    (data,addr)=mySocket.recvfrom(1024)
    s=struct.Struct('6d')
    data=s.unpack(data)
#-----

q1=data[0]
q2=data[1]
q3=data[2]
q4=data[3]
garra=data[4]
error=data[5]

```

Figura 5.33: Programación de datos en Python para la recepción de datos

Fuente: Propia

5.5. Variables de Estado:

Dado que el manipulador robótico no necesita tener todos sus movimientos activos en simultáneo a la vez se generó dos estados en el robot: el primero cuando la posición y la orientación del efector final tiene movimiento y la apertura de la garra que inmovilizada; y viceversa. Dichos estados dependen de tres variables: la fuerza máxima permitida para la garra, el error entre la posición deseada y obtenida, y un pulsador que se adicionó para el control del usuario al momento de la manipulación.

Estas variables determinarán un estado u otro basándose en la lógica de la tabla 5.2.

Tabla 5.2: Lógica de Estados

Número	Fuerza	Error	Pulsador	Estado
1	0	1	0	Movimiento del robot activo, movimiento de la garra desactivado
2	0	0	0	Transición (MR-RR)
3	0	0	1	Movimiento del robot desactivo, movimiento de la garra activado
4	1	0	1	Movimiento del robot desactivo, cierre de garra al máximo
5	1	0	0	Transición (RR-MR)
6	1	1	0	Movimiento del robot activo, movimiento de la garra desactivado, cierre de garra al máximo
7	1	0	0	Transición (MR-RR)
8	1	0	1	Movimiento del robot desactivo, movimiento de la garra activado, cierre de garra al máximo
9	0	0	1	Movimiento del robot desactivo, movimiento de la garra activado, garra abierta al máximo
10	0	0	0	Transición (RR-MR)
			MR	Movimiento del robot activo
			RR	Robot Rígido

Fuente: Propia

En el caso de la fuerza es 0 cuando el sensor de fuerza censa un valor menor o igual a 450 y es 1 cuando es mayor a 450. En el caso del error, se creó una función en Simulink que nos permite obtener un promedio de los errores por separado de cada articulación y en base a ese promedio se condiciona la salida del error de la siguiente manera, si es menor o igual a 3.5 el error en la tabla marca 0 y si es mayor a 3.5 marca 1.

De esta manera observamos que tenemos 2 estados que se repiten: “Transición (MR-RR)” en el número 2 y 7; y “Transición (RR-MR)” en el número 5 y 10; por lo cual tenemos que escoger el que mejor se adecue a nuestras condiciones de operación en cada caso. De esta forma descartaremos el número 2 y 7. (Tabla 5.3)

Tabla 5.3: Lógica de Estados Final

Número	Fuerza	Error	Pulsador	Estado
1	0	1	0	Movimiento del robot activo, movimiento de la garra desactivado
2	0	0	0	Transición (MR-RR)
3	0	0	1	Movimiento del robot desactivo, movimiento de la garra activado
4	1	0	1	Movimiento del robot desactivo, cierre de garra al máximo
5	1	0	0	Transición (RR-MR)
6	1	1	0	Movimiento del robot activo, movimiento de la garra desactivado, cierre de garra al máximo
7	1	0	0	Transición (MR-RR)
8	1	0	1	Movimiento del robot desactivo, movimiento de la garra activado, cierre de garra al máximo
9	0	0	1	Movimiento del robot desactivo, movimiento de la garra activado, garra abierta al máximo
10	0	0	0	Transición (RR-MR)
			MR	Movimiento del robot activo
			RR	Robot Rígido

Fuente: Propia

Teniendo nuestra tabla de estados armada procedemos a formar nuestra lógica combinacional teniendo como salida los dos estados antes mencionados. (Tabla 5.4)

Tabla 5.4: Lógica Combinacional

Fuerza	Error	Pulsador	Estado
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Fuente: Propia

Donde el estado 1 es el movimiento del robot activo, movimiento de la garra desactivado; y el estado 0 es movimiento del robot desactivo, movimiento de la garra activado.

Este análisis se agrega al programa en Python mediante el uso de condicionales “if” como se observa en la siguiente figura.

```
#LOGICA COMBINACIONAL
#0 es movimiento de garra y 1 es posicion y orientacion del robot

if fue==0 and error==0 and puls==0:
    condicion=0
elif fue==0 and error==0 and puls==1:
    condicion=1
elif fue==0 and error==1 and puls==0:
    condicion=0
elif fue==0 and error==1 and puls==1:
    condicion=1
elif fue==1 and error==0 and puls==0:
    condicion=0
elif fue==1 and error==0 and puls==1:
    condicion=1
elif fue==1 and error==1 and puls==0:
    condicion=0
elif fue==1 and error==1 and puls==1:
    condicion=1
#-----
print fue, error, puls, condicion
#=====
```

Figura 5.34: Programación en Python de la Lógica Combinacional

Fuente: Propia

5.6. Control Dinámico: Torque Computado:

Una vez definidas las etapas tanto de interpretación de información captada por el sensor Kinect XBOX 360 y la conversión de la posición cartesiana al espacio articular a través del control cinemático, se construyó el diagrama de bloques referente a la ley de control Torque Computado basándonos en la ecuación [2.40] y la figura 2.18, donde se puede observar claramente que es necesario definir la posición articular y velocidad articular deseada como valores de entrada.

Para ello se creó una etapa preliminar al control propiamente dicho, la misma que adquiere los datos articulares de la salida de las redes neuronales, ramificándose después en dos partes: la primera que ingresa directamente al bloque de control y la segunda que tiene el propósito de hallar la velocidad articular por medio del producto de la matriz Jacobiana Geométrica Inversa y la velocidad lineal y angular del robot. (Fig. 5.35)

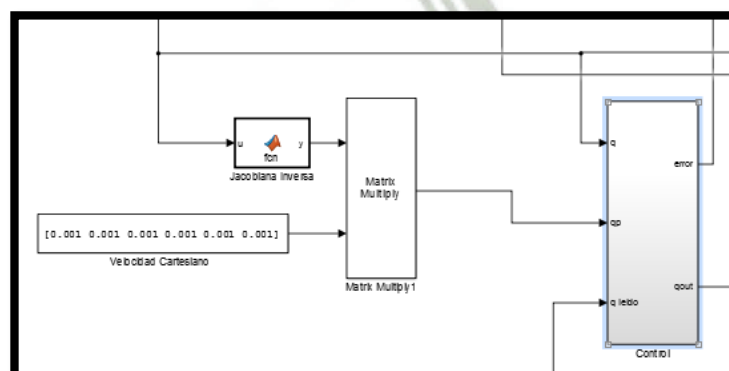


Figura 5.35: Definición de entradas al bloque de Control

Fuente: Propia

Se procedió a realizar el armado del bloque de control específicamente, tal como se puede observar en la figura 5.36 y en donde podemos identificar cuatro zonas principales:

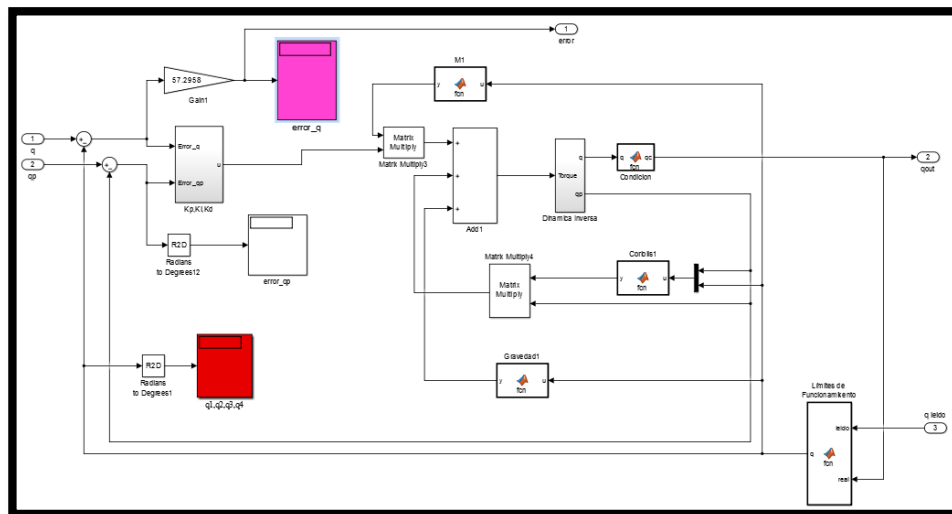


Figura 5.36: Desarrollo interno del Bloque de Control

Fuente: Propia

5.6.1. Regulador PID:

Consiste en un bloque, cuyos valores de entrada corresponden a los errores de posición y velocidad articulares, los mismos que se obtienen al restar la señal deseada o set point con la señal real o de realimentación; ésta última es obtenida directamente de los actuadores a través de la Tarjeta Raspberry Pi.

La figura 5.37 nos muestra el bloque PID encapsulado y su estructura interna, y en ella podemos observar también que la salida del bloque entrega la señal de control u .

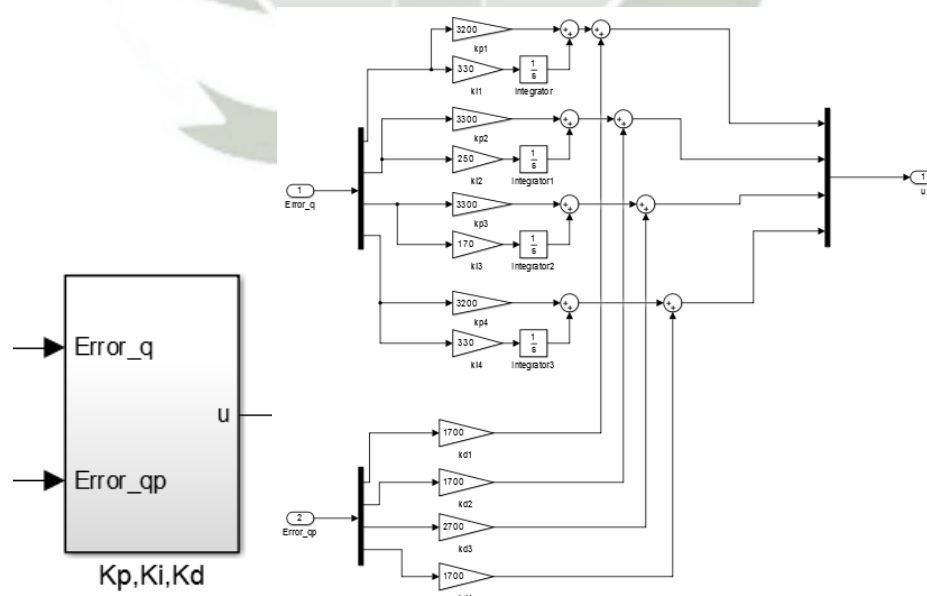


Figura 5.37: Bloque PID y estructura interna

Fuente: Propia

5.6.2. Dinámica Inversa:

Este bloque, está referido a la elaboración mediante un diagrama, de la dinámica inversa, la misma que fue calculada matemáticamente a través del algoritmo Lagrange Euler en el capítulo de diseño y que nos permitió obtener los valores de la Matriz de Inercias, Fuerza de Coriolis y el vector de Gravedad, en términos de variables.

Como se puede apreciar en la figura 5.38, la entrada de este subsistema es el par Motor y sus salidas son la posición y velocidad articulares estimadas.

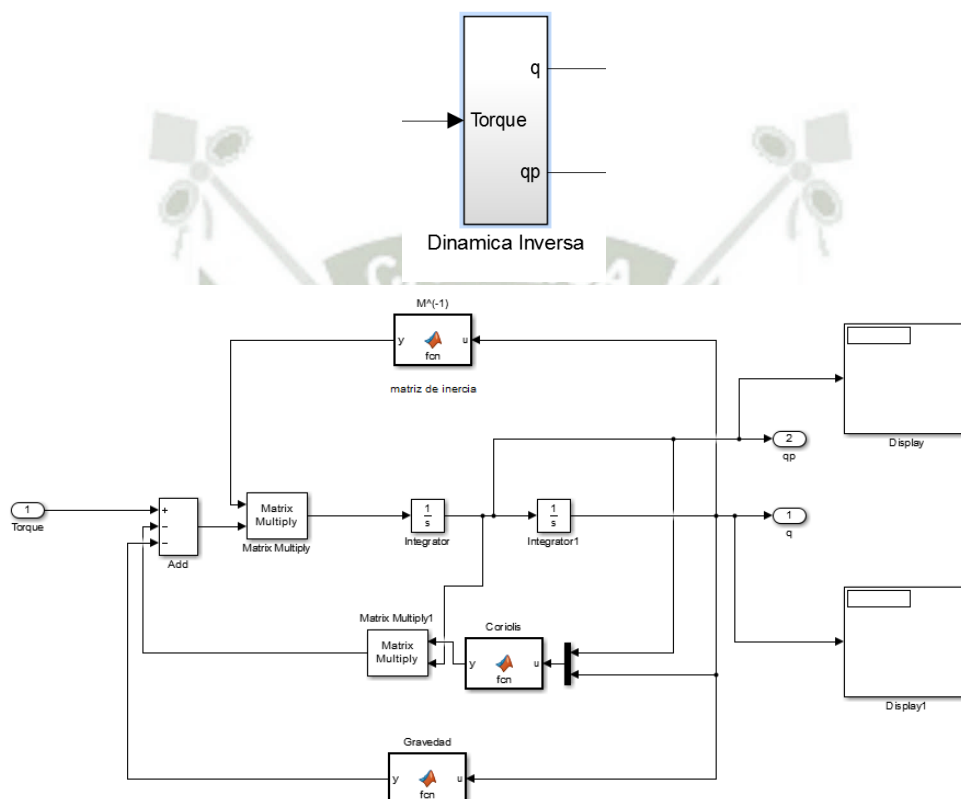


Figura 5.38: Bloque Dinámica Inversa y estructura interna

Fuente: Propia

5.6.3. Zona de Realimentación No - Lineal:

Se refiere al uso de la Matriz de Inercia, Fuerza de Coriolis y vector de gravedad, obtenidas en el diseño Mecánica de este proyecto, para la realimentación no lineal, con el fin de desacoplar las interacciones entre los eslabones del manipulador. Tiene como parámetros de entrada la señal de control “u”, obtenida del bloque PID y al momento de torsión como señal de salida, la misma que ingresará a la planta Dinámica del Robot.

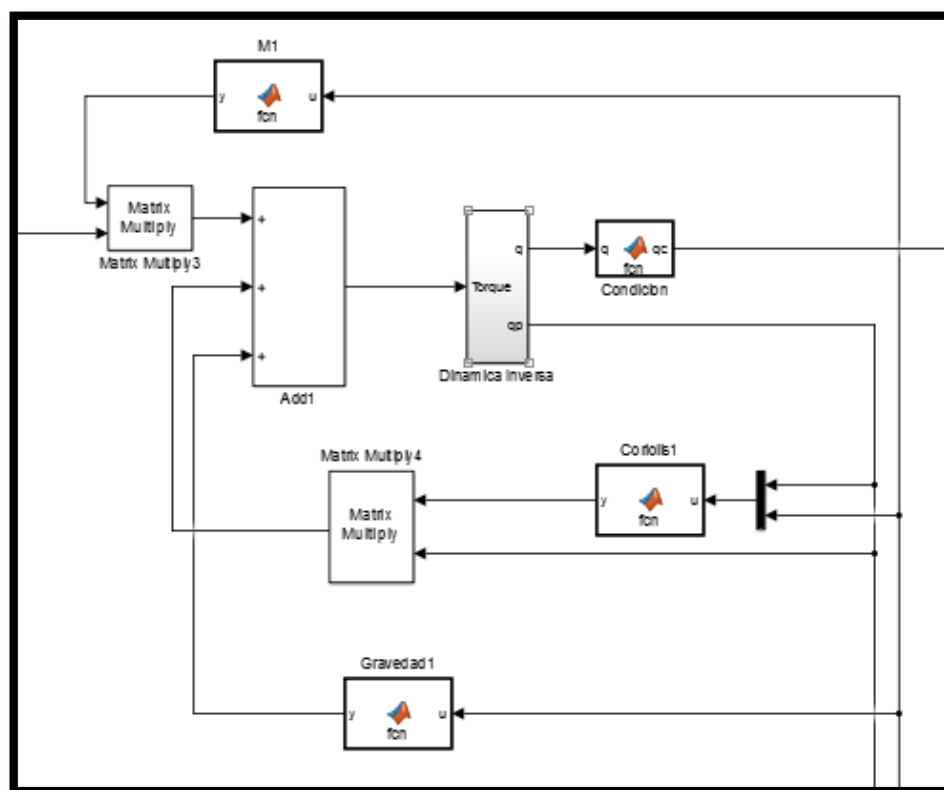


Figura 5.39: Zona de Realimentación No - Lineal

Fuente: Propia

5.6.4. Bloques de definición de Límites:

En la elaboración del Diagrama de Control se consideró, la adición de dos bloques que tienen como función la definición de los límites del espacio de trabajo del Manipulador Robótico en términos de los ángulos de cada servomotor, todo ello con la finalidad de proteger la estructura mecánica del robot de alguna ruptura, lo que a su vez provocaría sobrecarga y calentamiento en sus actuadores. (Fig. 5.40)

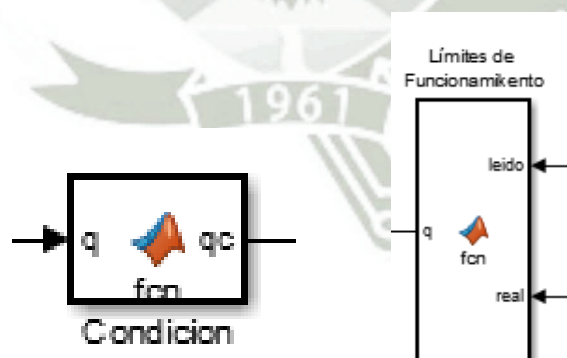


Figura 5.40: Bloques de Definición de Límites

Fuente: Propia

Es así que luego de realizar los enlaces respectivos entre cada uno de los bloques ya descritos, se encapsuló el diagrama en un subsistema denominado “Control”, que

cuenta con dos salidas, una correspondiente a la posición articular estimada transmitida al prototipo a través de la tarjeta Raspberry Pi y el error que es empleado como variable de estado para el buen funcionamiento del manipulador robótico.

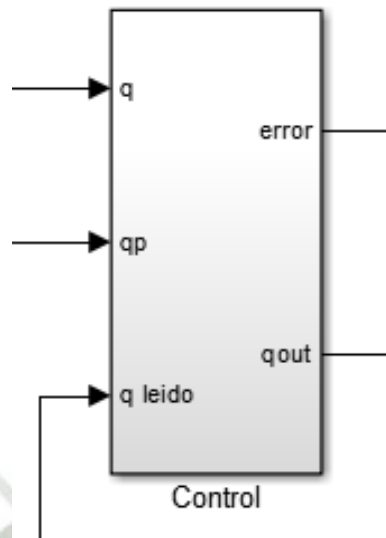


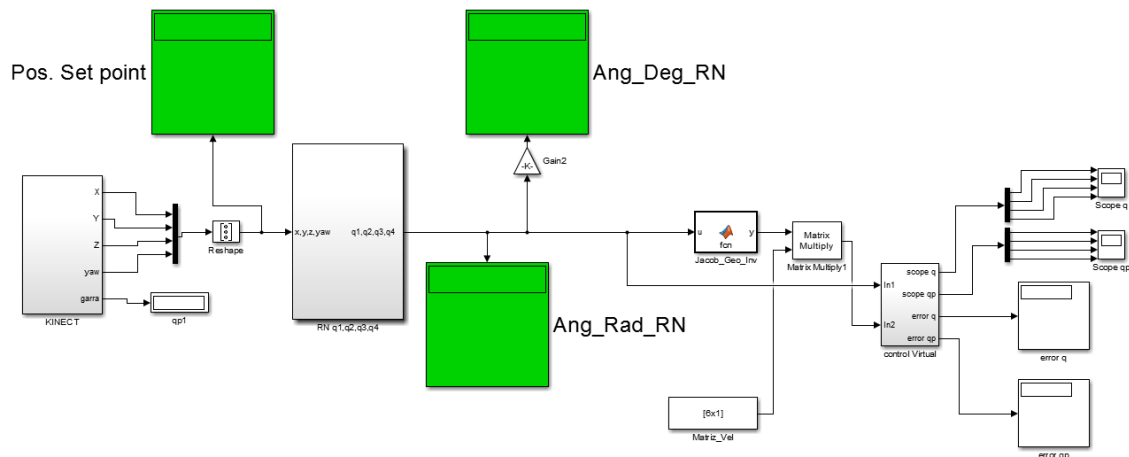
Figura 5.41: Subsistema Control
Fuente: Propia

5.7. Programación en Python

Una vez establecida la comunicación y teniendo definido todos los bloques en la Plataforma Simulink de Matlab, elaboramos un programa en lenguaje Python para la Tarjeta Raspberry Pi, “Dreamy” (Anexo K), el mismo que tiene como finalidad la adquisición del error y los valores articulares dados por el sistema de control desde la PC para su procesamiento y envío hacia los actuadores, logrando así el movimiento en el robot. Así mismo, este código se encarga de leer la posición actual del manipulador y exportarla hacia la PC para su posterior uso.

5.8. Prueba de Funcionamiento Virtual:

Previamente a realizar el ensayo de operatividad del sistema de control en el brazo robótico, se vio por conveniente realizar una prueba de funcionamiento virtual, con el propósito de evitar daños tanto en su estructura mecánica como en sus componentes electrónicos; para ello, se enlazó cada una de las etapas anteriormente descritas, a excepción de la comunicación y las variables de estado; así mismos, se reemplazó el bloque de la planta dinámica por el modelamiento virtual Simmechanics.


Figura 5.42: Control Virtual

Fuente: Propia

Posteriormente, se llevó a cabo una etapa de sintonización de las ganancias del regulador PID en forma experimental, basándonos en el conocimiento teórico acerca de la influencia que tiene cada una de ellas para la estabilización del movimiento del robot en la posición deseada.

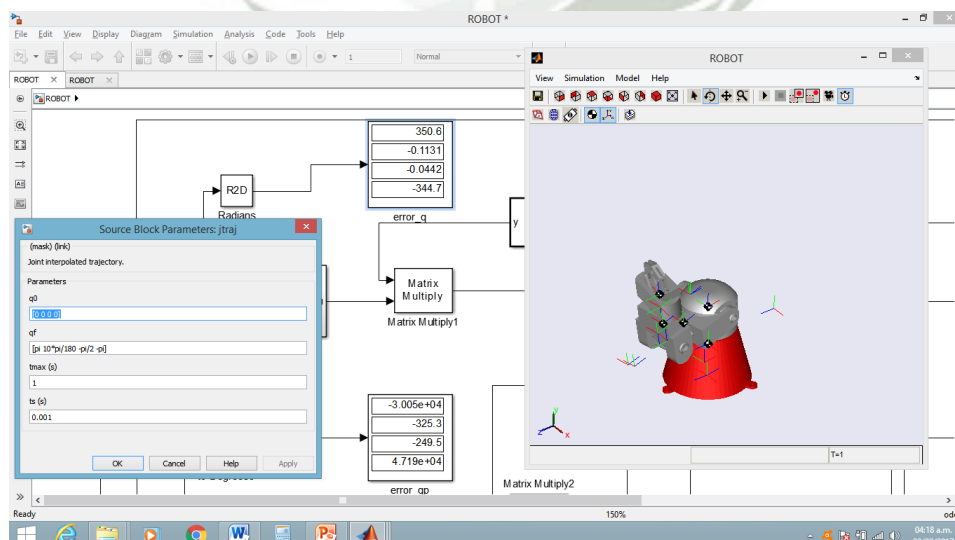
Dicha sintonización se inició considerando valores de ganancias generales para las cuatro articulaciones, obteniéndose como los más aceptables, los mostrados en la tabla 5.5.

Tabla 5.5: Valores de Ganancias de Sintonización Virtual

Kp	Ki	Kd
100000	0.1	600

Fuente: Propia

Para el ensayo, se utilizó una trayectoria punto a punto teniendo como posición inicial $[0;0;0;0]$ y posición final $[\pi; \frac{-10\pi}{180}; \frac{\pi}{3}; -\pi]$, durante un periodo de tiempo de 1 segundo y un tiempo de muestreo de 0.001 segundo. (Fig. 5.43)


Figura 5.43: Parámetros de prueba

Fuente: Propia

De esta manera se obtuvo la gráfica de la posición articular deseada vs obtenida, donde se puede ver que efectivamente las cuatro articulaciones estabilizan. (Fig. 5.44)

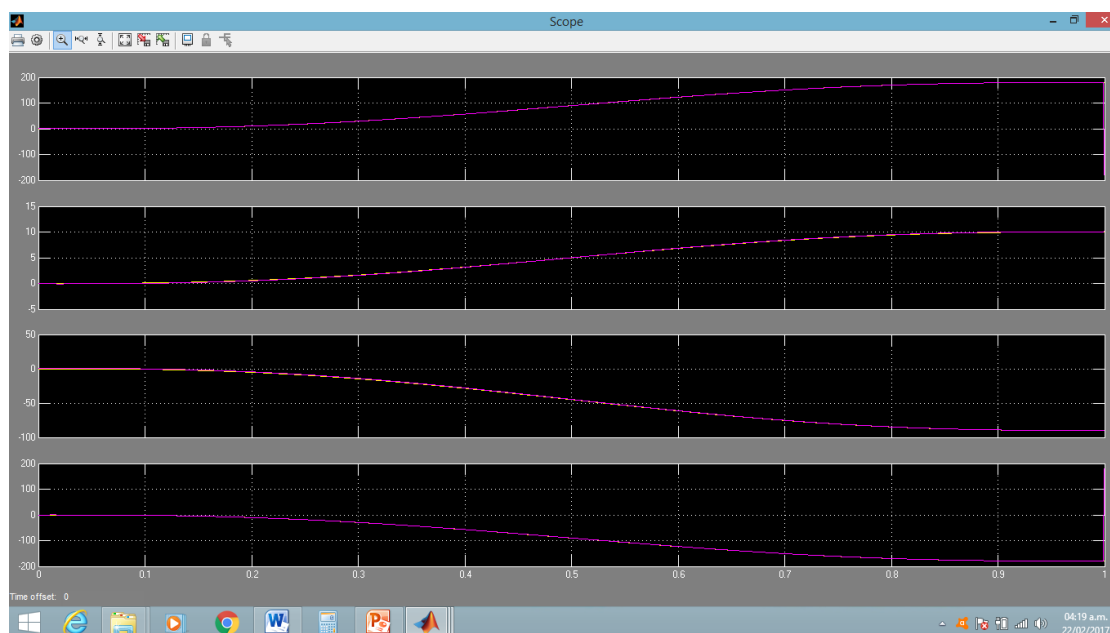


Figura 5.44: Gráfica de Posición articular deseada vs obtenida
Fuente: Propia

Sin embargo, al realizar un análisis minucioso del resultado y de las ganancias determinadas, surgió la incertidumbre de poner a prueba en el manipulador, el valor de 1000000 como K_p , razón por la cual, y después de realizados varios ensayos, se decidió considerar un regulador PID por articulación, cuyos valores óptimos están detallados en la tabla 5.6.

Tabla 5.6: Ganancias por Articulación

Ganancias	q1	q2	q3	q4
Kp	3200	3300	3300	3200
Ki	330	250	170	250
Kd	1700	1700	2700	1700

Fuente: Propia

Para la prueba de funcionamiento, se consideró la posición inicial del brazo robótico (0, 0, 0, 0) como set point, obteniendo así los resultados que se muestran en las siguientes figuras:

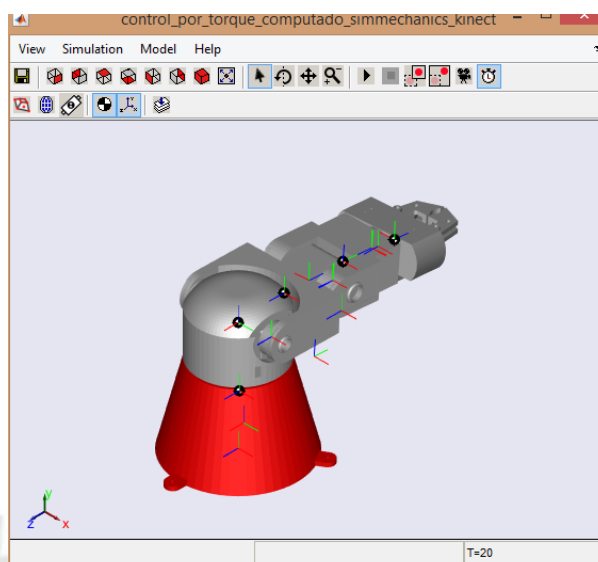


Figura 5.45: Simulación Simmechanics

Fuente: Propia

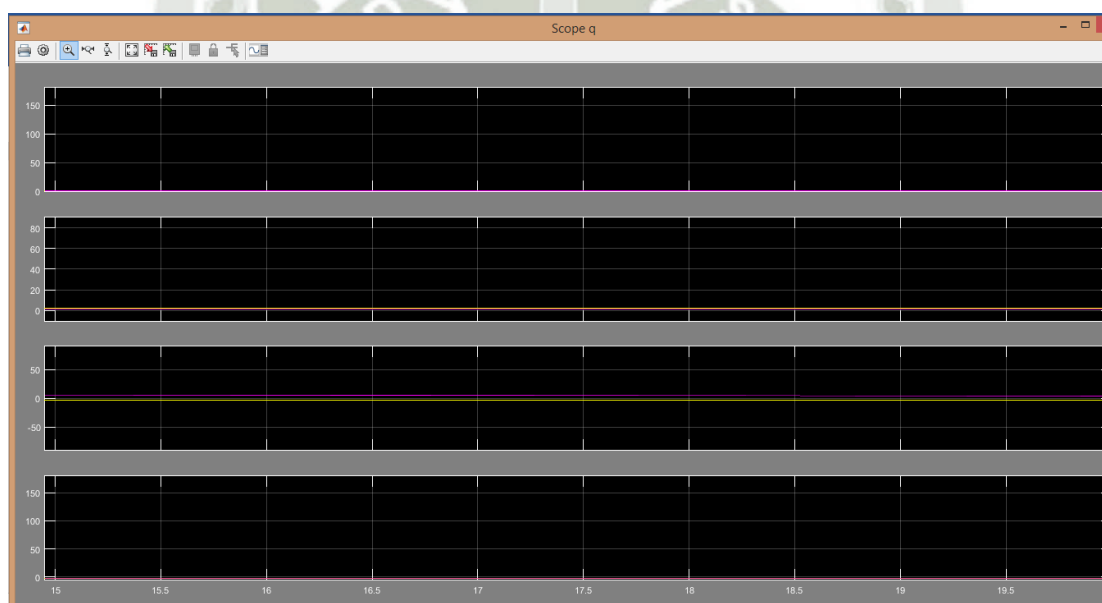
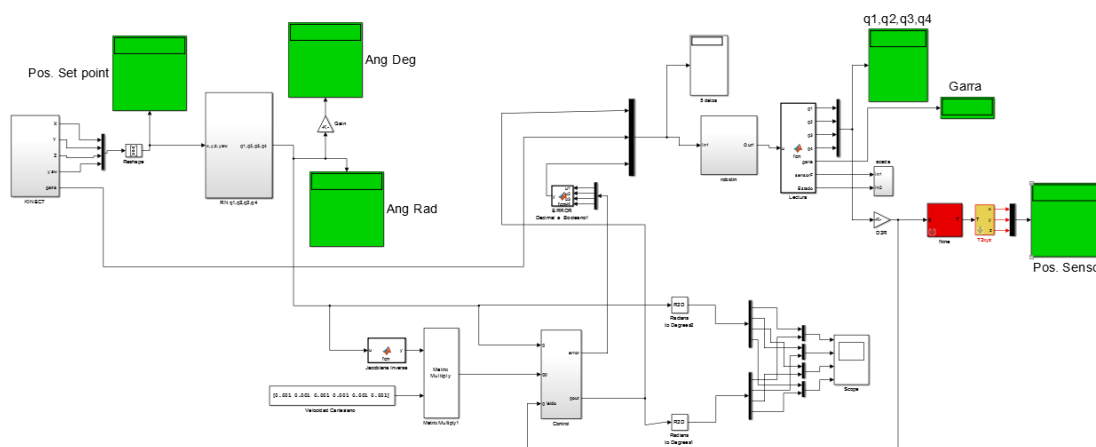


Figura 5.46 : Gráfica de Posición articular deseada vs obtenida

Fuente: Propia

Como podemos apreciar en el gráfico 5.46, los actuadores en su totalidad estabilizan alcanzando así la posición deseada; dado ello se procede con la construcción del sistema de control para la prueba de funcionamiento del brazo robótico, enlazando todas las etapas anteriormente descritas desde el sensor Kinect XBOX 360 hasta el movimiento controlado del robot, tal como se puede observar en la figura 5.47.


Figura 5.47: Sistema de Control Real

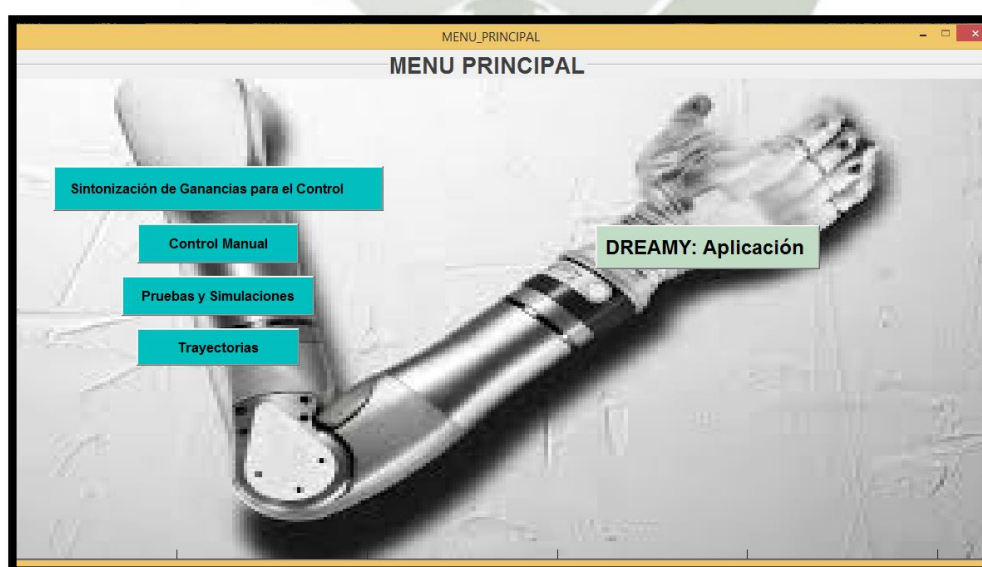
Fuente: Propia

5.9. Interfaz Gráfica de Usuario:

El gui o también conocido como interfaz gráfica de usuario nos permite un control sencillo del funcionamiento del manipulador robótico lo cual elimina la necesidad de aprender el lenguaje de programación y escribir comandos a fin de ejecutar su aplicación. Es por esto que se diseñó una interfaz intuitiva y fácil de manejar para el usuario final.

La interfaz gráfica se elaboró en la plataforma “Graphical User Interface” del Software Matlab ya que nos ofrece variadas opciones y nos permite enlazar programas en simulink, scripts y funciones.

Se diseñó ante todo una ventana de Menú Principal donde se encontrarán las opciones que se permitirán realizar al usuario y la aplicación final del manipulador (Fig. 5.48)


Figura 5.48: Ventana de Menú Principal

Fuente: Propia

Esta ventana cuenta con 5 opciones que se describirán a continuación:

5.9.1. Sintonización de Ganancias para el Control:

Se elaboró esta ventana con el fin que el usuario pueda probar diferentes valores de k_p , k_d y k_i para cada articulación (Fig. 5.49) y pueda probarlos mediante el programa “sintonización_de_ganancias.slx” creado en simulink, que consiste en la utilización del jtraj para generar la trayectoria, un subsistema donde se ingresará los valores de k_p , k_d y k_i ; continuando con el esquema de torque computado aplicado al ROBOT simulado por SimMechanics y terminando en dos scope, uno para medir la posición deseada y obtenida y otro para medir la velocidad también deseada y obtenida. (Fig. 5.50)

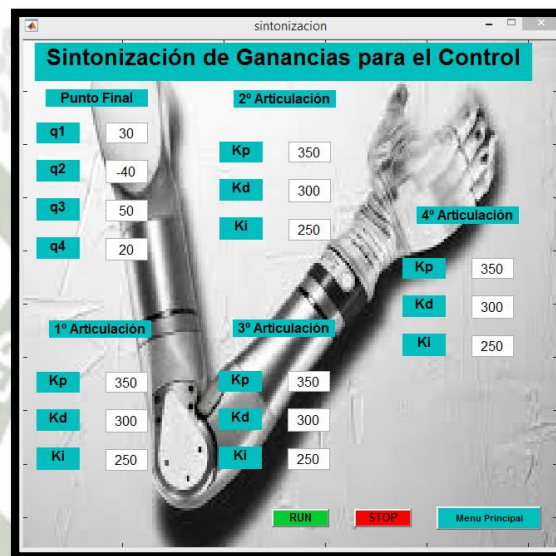


Figura 5.49: Ventana de Sintonización de ganancias
Fuente: Propia

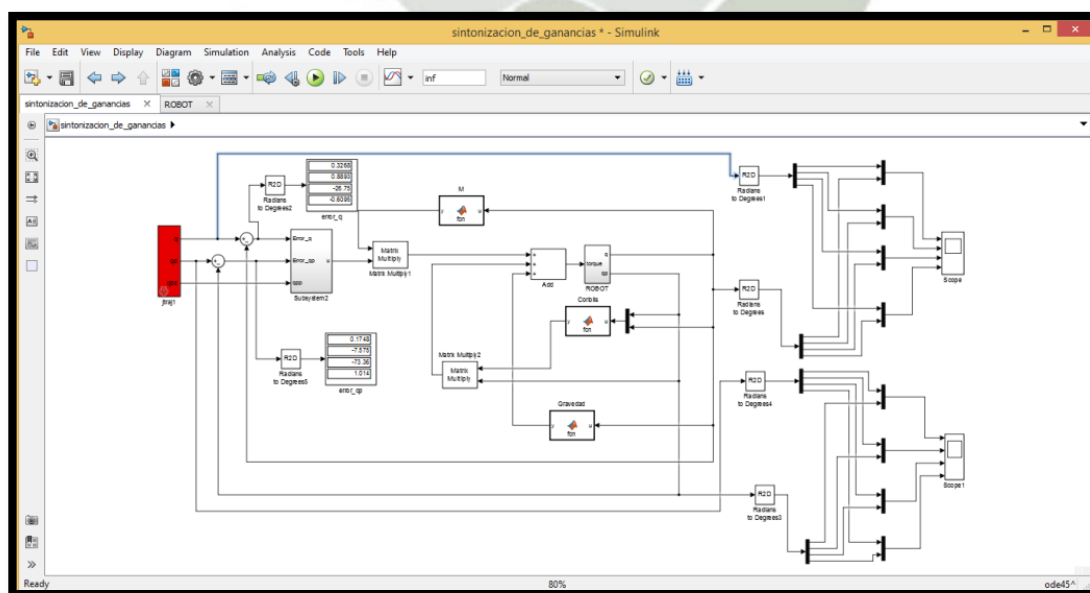


Figura 5.50: Ventana Sintonización_de_ganancias.slx
Fuente: Propia

Cuenta además con 3 botones: “RUN”, para correr el programa en función de los valores que se hallan colocado tanto en el punto final como en cada articulación; “STOP”, con el fin de detener el programa en cualquier momento y “Menú Principal”, para volver a la ventana de Menú Principal.

Una vez colocados los valores del punto final como de los k_p , k_d y k_i de cada articulación procedemos a presionar el botón RUN y nos abrirá dos ventanas adicionales: una ventana con el SimMechanics en movimiento y otra ventana con el Scope que muestra la posición deseada y la obtenida ambas ventanas durante un periodo de 0.2seg considerado como tiempo prudente para el análisis de las ganancias. (Fig. 5.51 y 5.52)

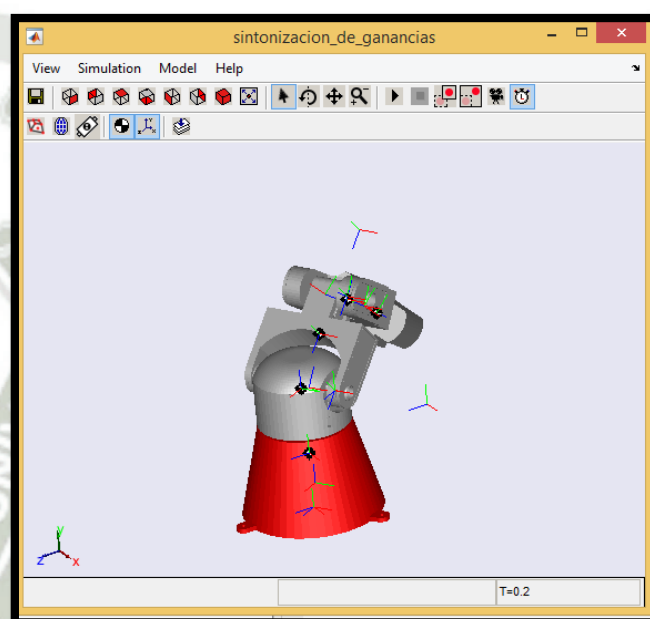


Figura 5.51: Ventana de SimMechanics
Fuente: Propia

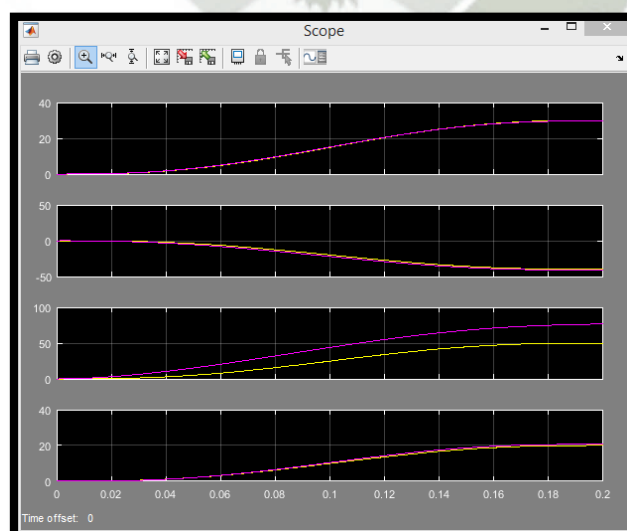


Figura 5.52: Scope de posición deseada y obtenida
Fuente: Propia

5.10. Control Manual:

Este botón nos abre la ventana “control_manual”, el cual nos permite mover el brazo robótico eslabón por eslabón enviando valores articulares q_1 , q_2 , q_3 , q_4 y de la garra hacia los servomotores de tal manera que podamos observar el movimiento de cada articulación.

Esta ventana nos permite variar cada articulación mediante un slider o tan solo colocar el valor articular en degrees; además cuenta con un graficador en 3D que se mueve en función de los valores articulares que se coloquen. Tenemos también 4 botones: “ENVIAR”, que recoge los valores articulares del Guide control_manual y los envía al programa en simulink Control_Manual.slx; “RUN”, este botón corre el simulink Control_Manual.slx y envía los valores al robot; “STOP”, detiene el proceso de envío al robot; “MENU PRINCIPAL”, nos devuelve a la ventana Menú Principal. (Fig. 5.53)

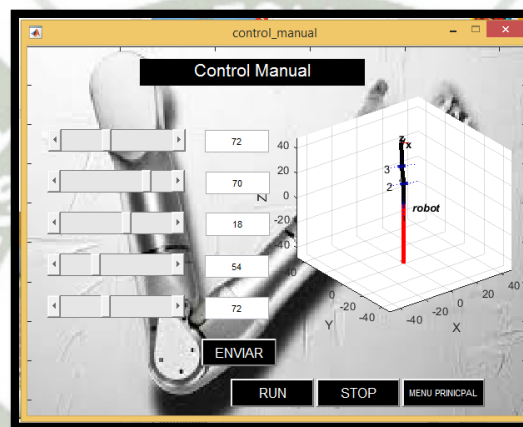


Figura 5.53: Ventana de Control Manual
Fuente: Propia

Esta ventana está enlazada al programa en simulink “Control_Manual.slx” que se encarga de enviar estos valores articulares al robot mediante el uso de un bloque de comunicación “UDP Send” que nos provee Matlab (Fig. 5.54).

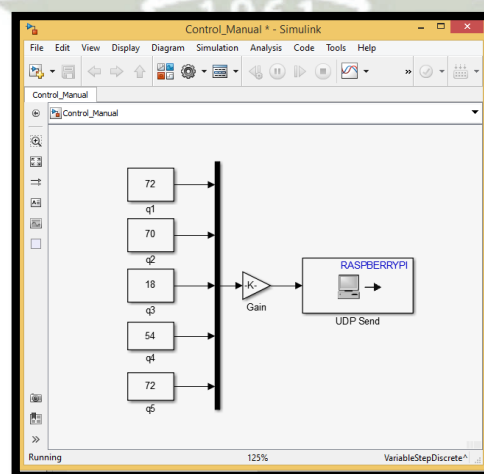


Figura 5.54: Ventana Control_Manual.slx
Fuente: Propia

5.11. Prueba y Simulaciones:

Esta ventana nos permite verificar la cinemática directa mediante las ecuaciones del robot, así como la cinemática inversa tanto con cálculos matemáticos como con redes neuronales de tal manera que se pueda observar los resultados en ambos casos.

Tenemos como primera parte de la interfaz la cinemática directa que nos permite ingresar valores articulares y obtener la posición y orientación del efector final mediante el uso del botón “ENVIAR”, el mismo que contiene las ecuaciones geométricas del robot. Además, está acompañado de un graficador que muestra la posición obtenida en la cinemática directa (Fig. 5.55).

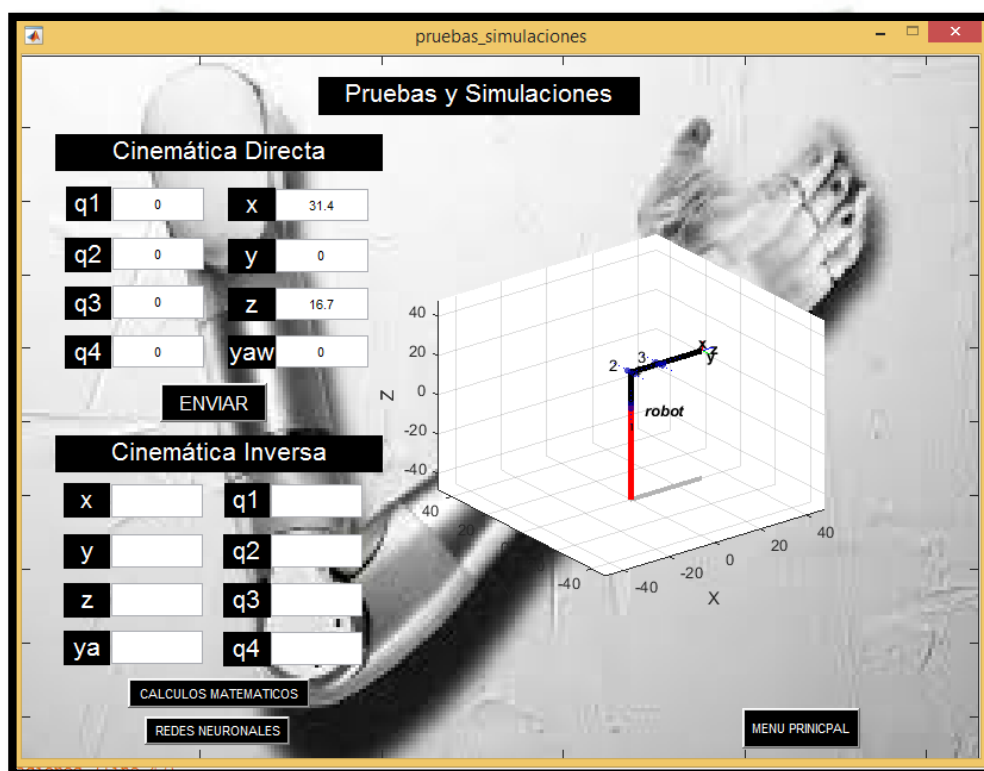


Figura 5.55: Ventana pruebas_simulaciones aplicando cinemática directa

Fuente: Propia

En la parte inferior de la interfaz se puede ingresar la posición y orientación del efector final y obtener su cinemática inversa. Para esto se crearon dos botones: “CALCULOS MATEMATICOS”, que nos permite obtener la cinemática inversa mediante ecuaciones matemáticas y “REDES NEURONALES”, que mediante el uso de redes multicapa permite aprender la cinemática inversa y obtener también dichos valores; ambos botones tienen la misma función pero lo resuelven de distinta forma de esta manera le da al usuario la oportunidad de poder comparar ambos métodos y ver cuál es más eficiente; del mismo modo que en la cinemática directa está acompañado de un graficador que nos permita observar los valores articulares obtenidos. (Fig. 5.56 Y 5.57)

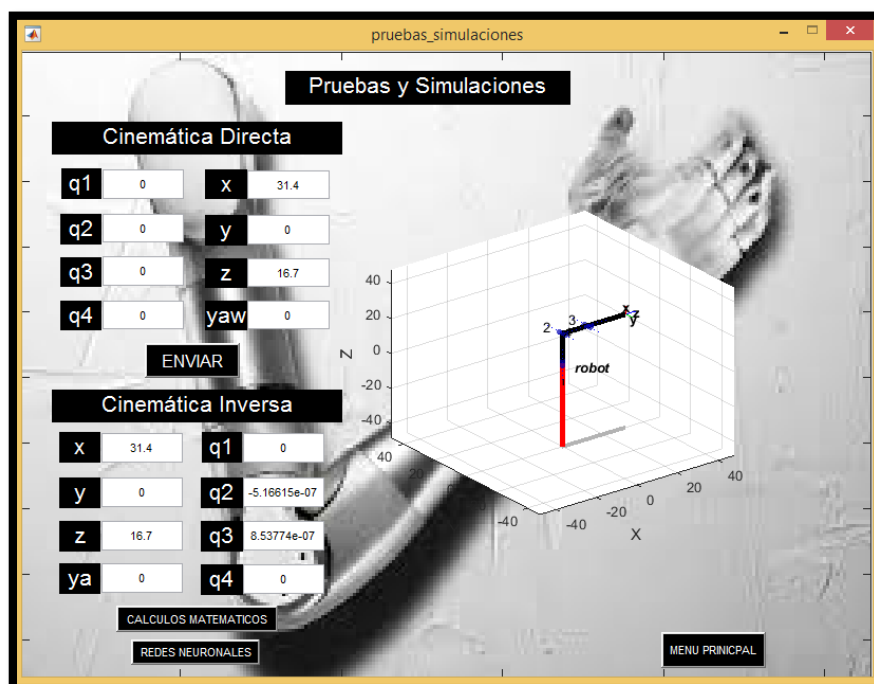


Figura 5.56: Ventana pruebas_simulaciones aplicando cinemática inversa mediante Cálculos Matemáticos
Fuente: Propia

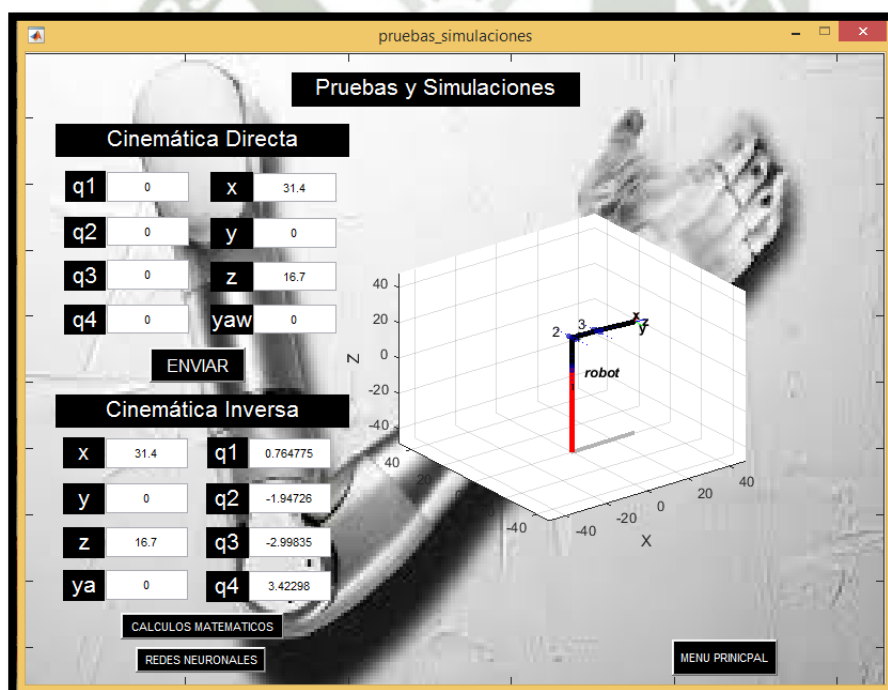
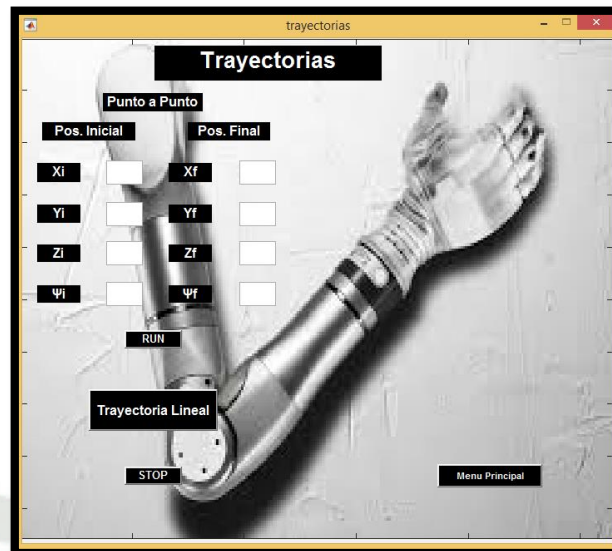


Figura 5.57: Ventana pruebas_simulaciones aplicando cinemática inversa mediante Redes Neuronales

Fuente: Propia

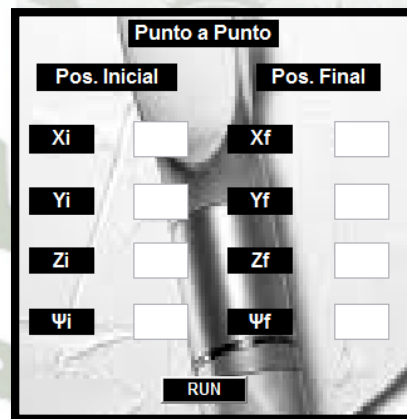
5.12. Trayectorias:

La ventana trayectorias tiene como función probar dos tipos de trayectorias y simularlas, con este propósito se dividió la misma en 3 partes (Fig. 5.58):

**Figura 5.58:** Ventana Trayectorias*Fuente: Propia*

- **Trayectoria Punto a Punto:**

Nos permite ingresar la posición y orientación cartesiana inicial y final. (Fig. 5.59) Seguidamente mediante el uso del botón “RUN” se correrá el programa en simulink llamado “trayectoria_punto_punto.slx con los valores de la posición inicial y final que se hallan ingresado.

**Figura 5.59:** Sección Trayectoria punto a punto*Fuente: Propia*

El programa en simulink está formado por un bloque Jtraj que tiene como función interpolar los puntos iniciales y finales ingresados, para de esta manera obtener una trayectoria con dichos puntos; cuenta también con un bloque de sintonización de ganancias llamado “Kp, Ki, Kd” y el esquema de torque computado aplicado al bloque ROBOT (simulación del robot en SimMechanics); se usó la combinación de bloques “Fkine” y “T2xyz” para mediante un Scope poder observar la posición cartesiana deseada y obtenida en cada punto interpolado; y por último dos Scope para visualizar la posición y velocidad articular deseada y obtenida en cada punto interpolado (Fig. 5.60).

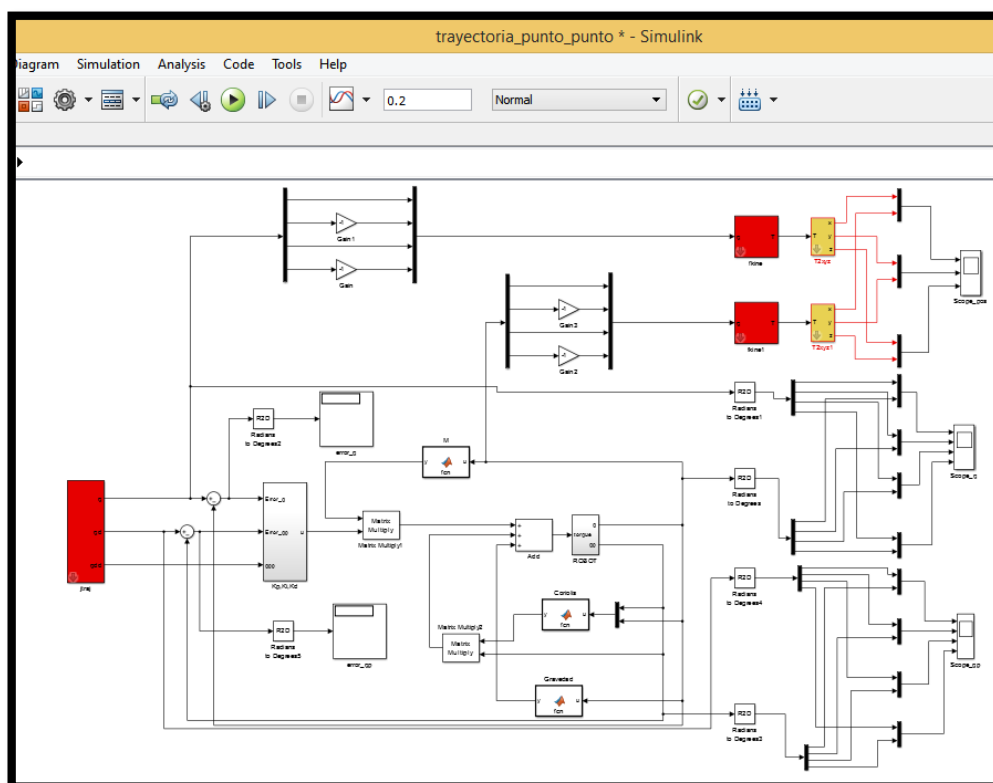


Figura 5.60: Programa en Simulink “trayectoria_punto_punto.slx”
Fuente: Propia

De esta forma se obtiene los siguientes resultados:

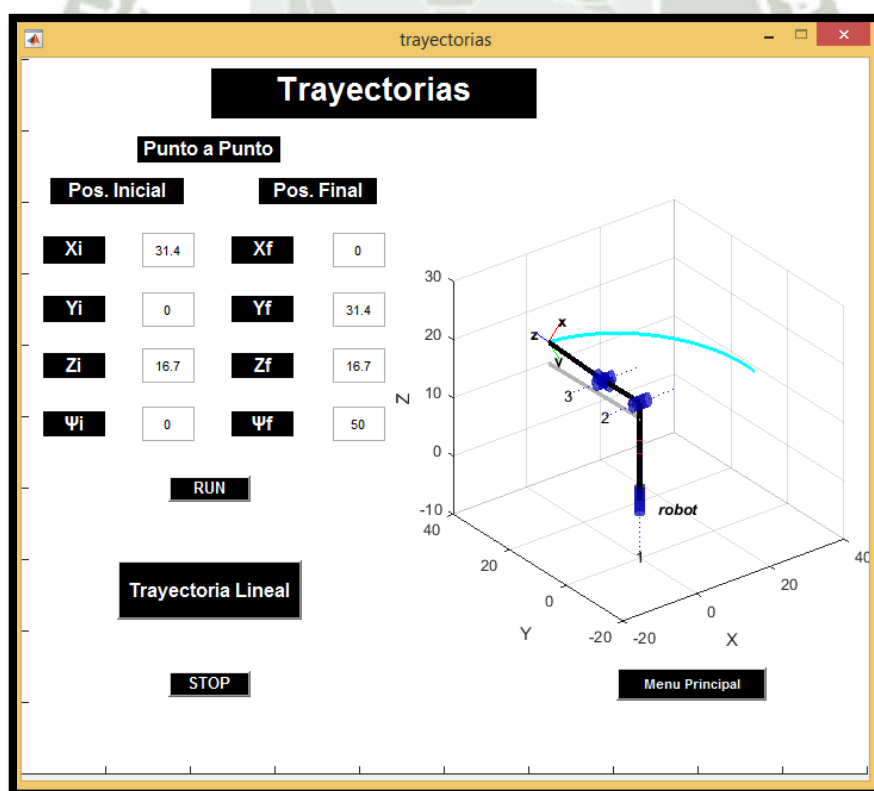


Figura 5.61: Prueba de trayectoria punto a punto
Fuente: Propia

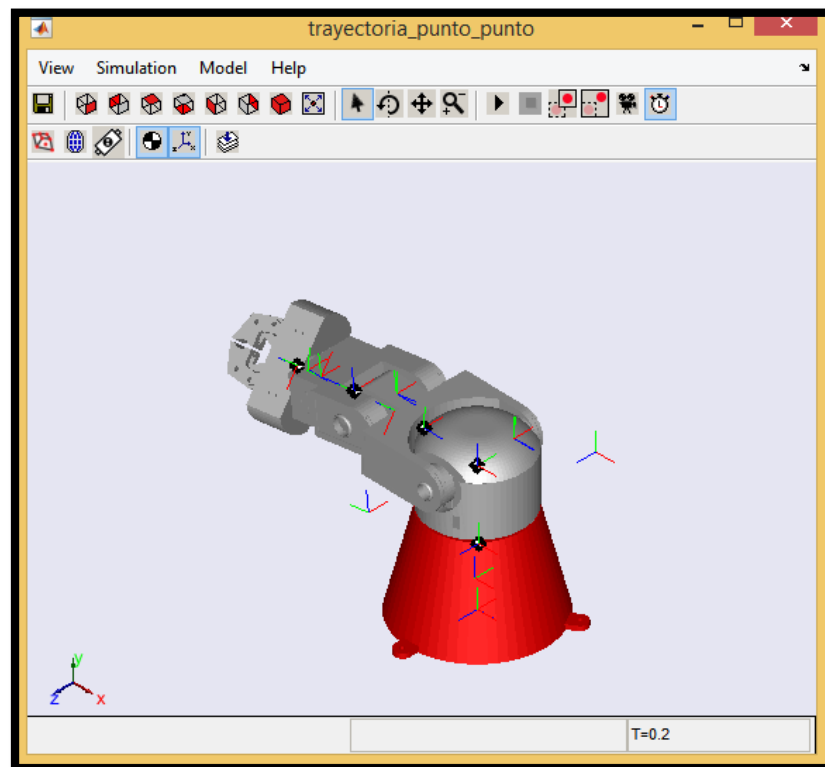


Figura 5.62: Prueba de trayectoria punto a punto observada en SimMechanics
Fuente: Propia

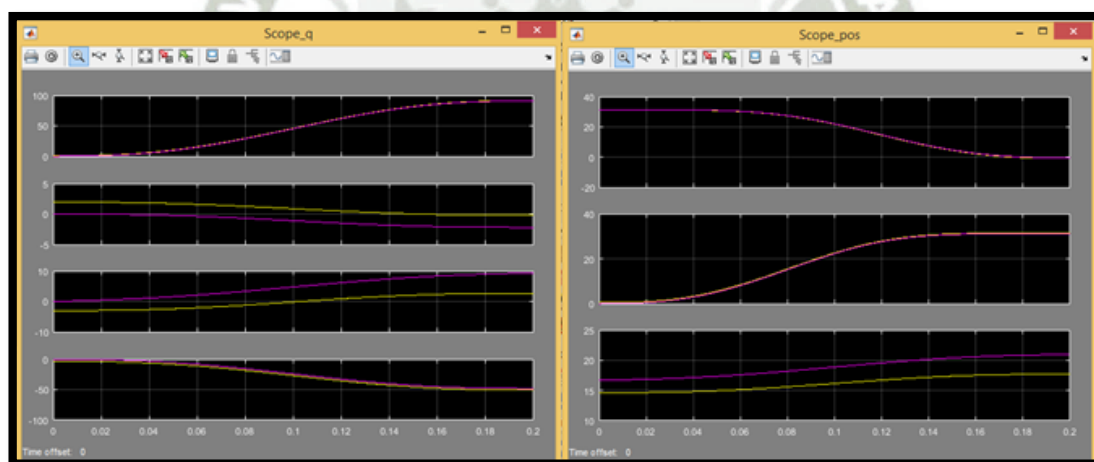


Figura 5.63: Gráfica de la posición articular (izquierda) y la posición cartesiana (derecha)
Fuente: Propia

- **Trayectoria Lineal:**

Para la trayectoria lineal se creó un botón llamado “Trayectoria Lineal” (Fig. 5.64) que nos permite correr el programa en simulink denominado “trayectoria_lineal.slx”.



Figura 5.64: Sección Trayectoria lineal
Fuente: Propia

El programa en Simulink es casi idéntico al programa en simulink “trayectoria_punto_punto.slx” con tan solo una diferencia, que el bloque “Jtraj” será reemplazado por un subsistema llamado “Trayectoria”

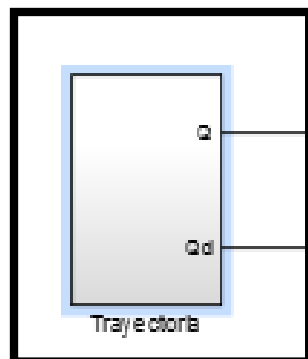


Figura 5.65: Subsistema Trayectoria

Fuente: Propia

Dicho subsistema cuenta con la función llamada “TRA_LINEAL_Q.m”, que define la trayectoria que seguirá el robot, además de un bloque “Clock” para el periodo de tiempo en que dicha trayectoria se irá formando; y para la obtención de la velocidad articular se hace uso de la Jacobiana Inversa multiplicada por una velocidad cartesiana, la misma que se elige según los requerimientos del usuario; de esta forma obtenemos un nuevo programa llamado “trayectoria_lineal”(Fig. 5.67).

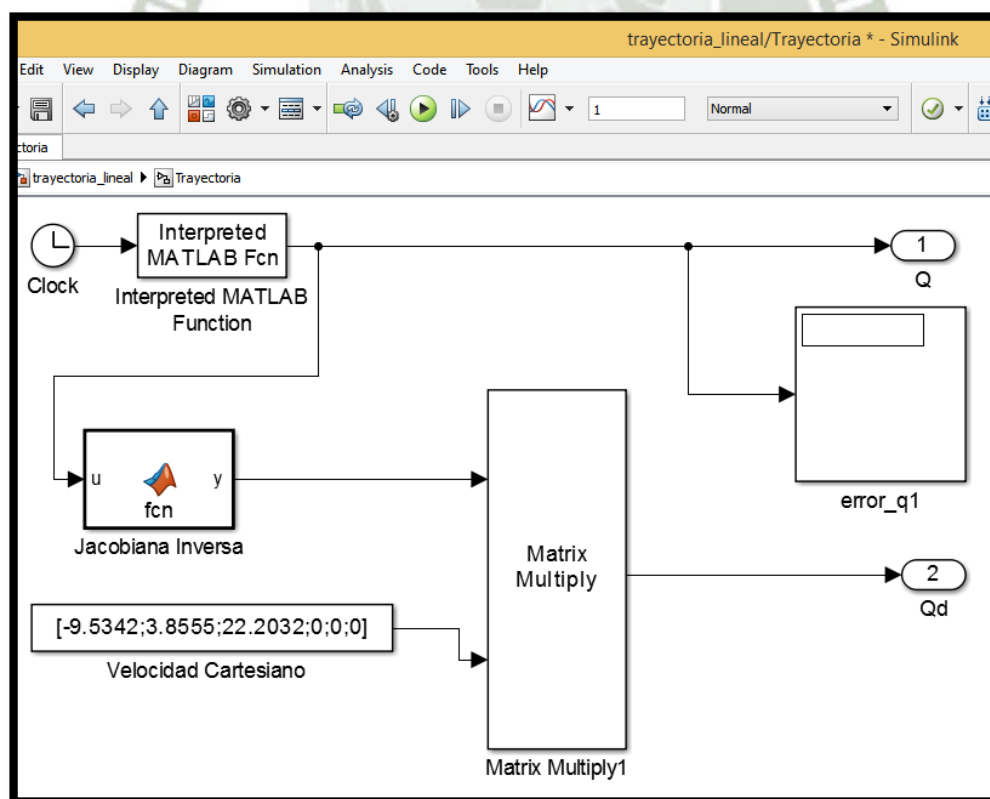


Figura 5.66: Subsistema “Trayectoria” internamente

Fuente: Propia

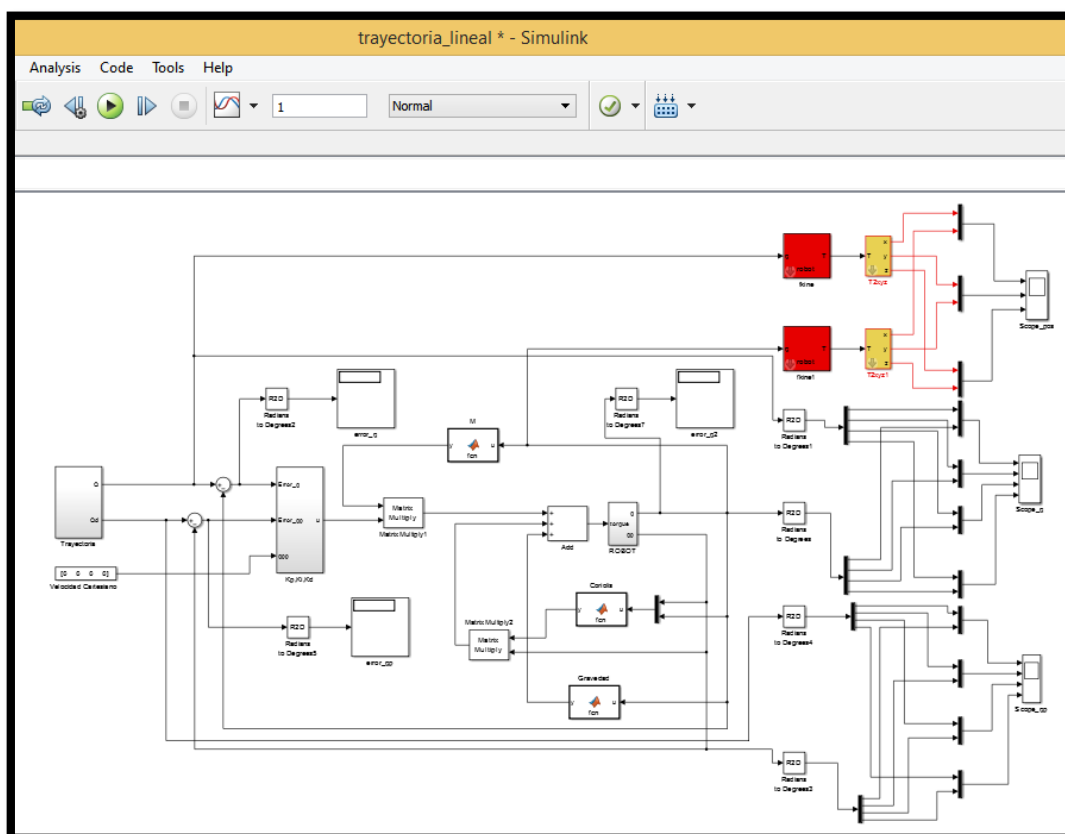


Figura 5.67: Programa en simulink “trayectoria_lineal.slx”

Fuente: Propia

De esta forma se obtiene los siguientes resultados:

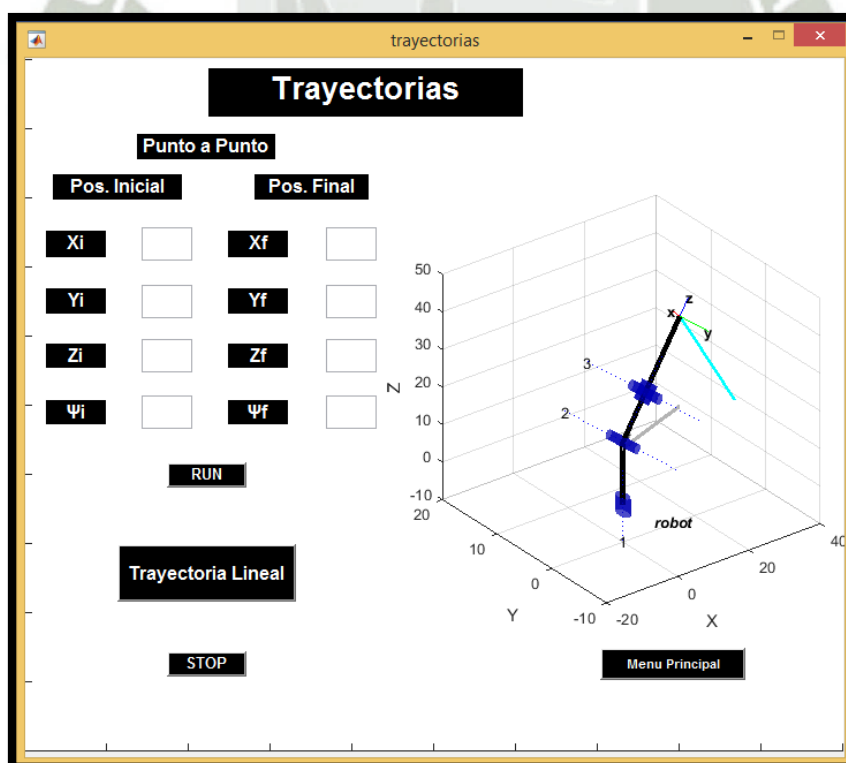


Figura 5.68: Prueba de trayectoria lineal

Fuente: Propia

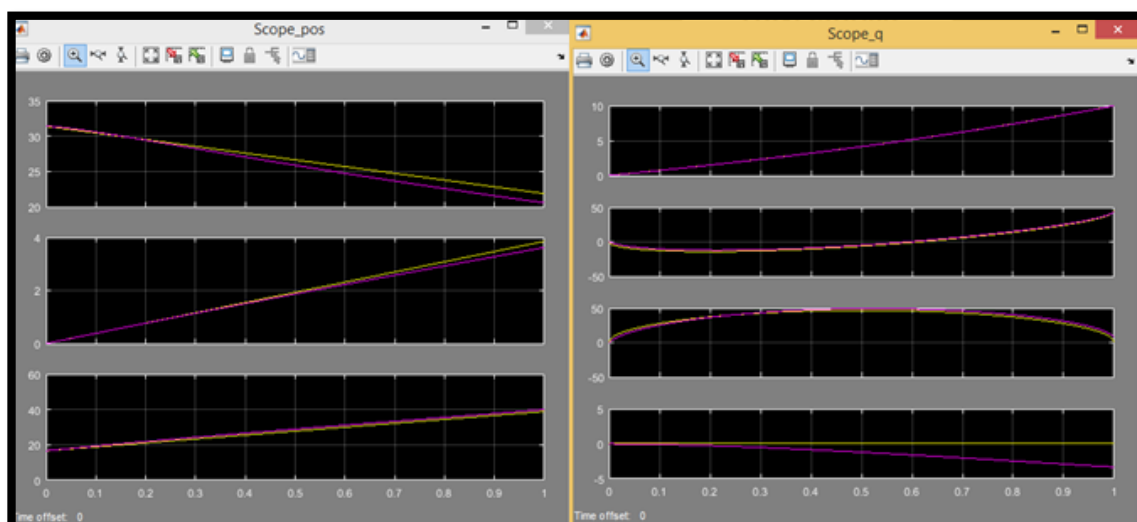


Figura 5.69: Gráfica de la posición articular (izquierda) y la posición cartesiana (derecha)

Fuente: Propia

5.13. Dreamy:

La última opción de nuestra Interfaz Gráfica se denomina “Dreamy” y tiene por finalidad iniciar la aplicación del proyecto, es decir correr el programa en Simulink “Dreamy.slx” (Fig. 5.70) y mostrarnos una ventana llamada SCADA donde se podrá visualizar dos botones, uno referente a la fuerza del efector final y otro para mostrarnos los estados del robot. (Fig. 5.71)

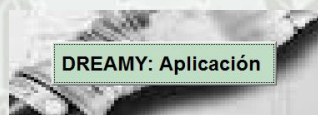


Figura 5.70: Botón “Dreamy”

Fuente: Propia

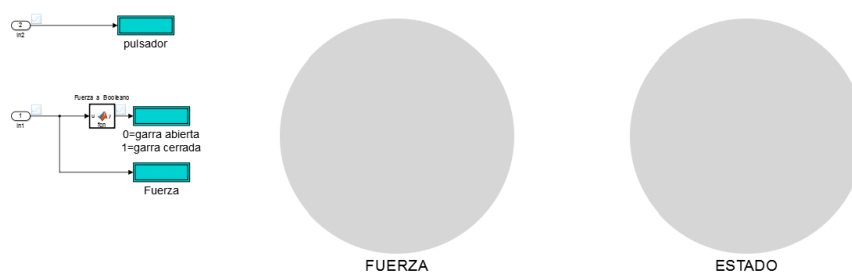


Figura 5.71: Ventana del bloque Scada llamado por el botón Dreamy

Fuente: Propia

Ambos botones servirán de guía para el usuario al momento de manipular el robot, por lo tanto, serán mostrados todo el tiempo que el brazo robótico este realizando su aplicación.

CAPÍTULO VI: PRUEBAS Y RESULTADOS

En este capítulo se mostrará los parámetros que se han de tener en cuenta para el correcto funcionamiento del robot, se visualizará también la posición adecuada del operador y los movimientos que debe realizar para la manipulación del brazo robótico.

Para iniciar el funcionamiento del robot se debe conectar el cable de alimentación del supresor de picos a la Red Eléctrica, y por seguridad siempre se está apagado el estabilizador hasta que se esté seguro de que todo está bien conectado, sin embargo, el sensor Kinect ya estará encendido pues este va directamente al supresor de picos por lo tanto nos permitirá verificar su conectividad con Matlab antes de encender todo lo demás. (Fig. 6.1)

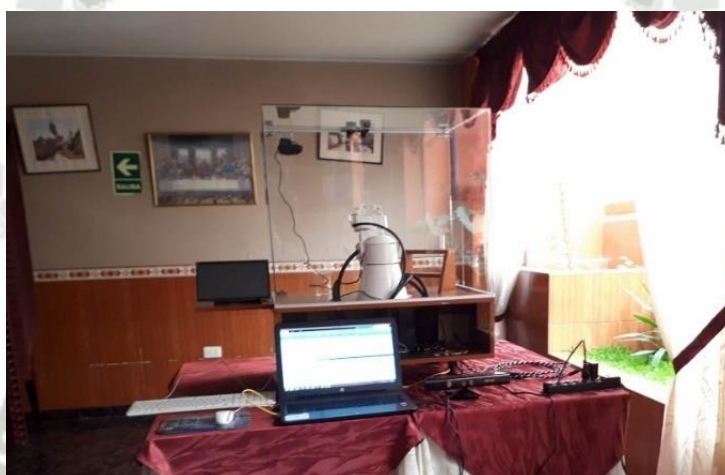


Figura 6.1: Conexión a la red eléctrica

Fuente: Propia

Teniendo el sensor Kinect encendido comprobamos su conectividad con Matlab mediante el comando “`imaghwinfo('kinect')`” que nos mostrará sus parámetros solo si se realizó una correcta conexión. (Fig. 6.2)

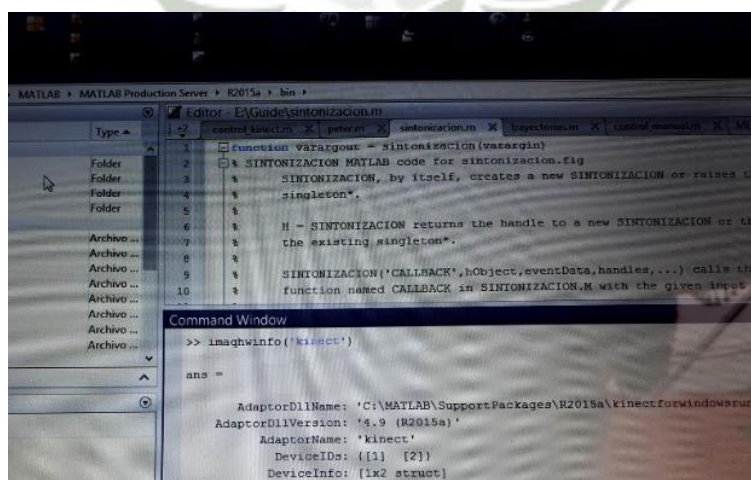


Figura 6.2: Comprobación de conectividad del Sensor Kinect

Fuente: Propia

Luego de verificar la conexión del sensor Kinect y haber revisado las conexiones de todos los circuitos procedemos a encender el estabilizador que a su vez distribuirá energía a las fuentes de alimentación, al Hub y a la tarjeta Raspberry Pi. (Fig. 6.3)



Figura 6.3: Encendido de todos los componentes electrónicos

Fuente: Propia

Según algunas pruebas que se realizaron previamente se observó que los motores trabajan sin esfuerzo a 11.3 V, lo cual nos hizo reconsiderar esta medida y establecer las fuentes a ese nivel de tensión.

Teniendo encendido el Raspberry Pi, tenemos que correr el programa “Dreamy” para lo cual utilizaremos la consola de comandos y digitaremos “sudo Python Dreamy.py” seguida de la tecla “intro”; lo que dejará a nuestra tarjeta a la espera del programa en Matlab. (Fig. 6.4)

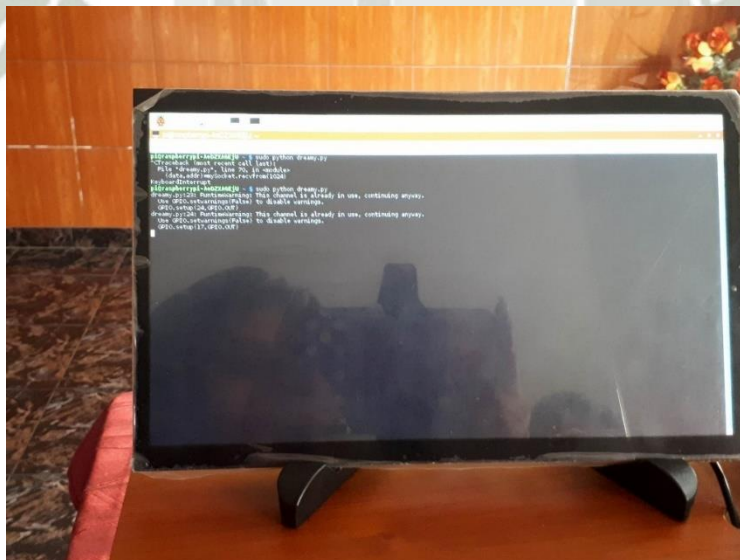


Figura 6.4: Programa “Dreamy” corriendo en Python

Fuente: Propia

Seguidamente corremos la interfaz gráfica en Matlab llamada “MENU_PRINCIPAL.m”, y nos dirigimos al botón “Dreamy”; y se empezará a compilar el programa en Simulink “Dreamy” (5 – 10 seg.). Terminada la compilación del programa se abrirá la ventana “Scada” con los dos leds “Fuerza” y “Estado”. Y se activará en el Python la cámara Logitech que le dará visión al operador de la mesa de trabajo. Si en la distancia de detección del sensor Kinect no hay ninguna persona simplemente el robot por defecto esperará en su posición inicial o Home. (Fig. 6.5)


Figura 6.5: Posición Inicial o Home

Fuente: Propia

El usuario deberá contar con el pulsador en su mano derecha iniciando en “1”, este le permitirá controlar el movimiento del robot referente a posición y orientación o la apertura y cierre de la garra. Cuando se ubique en la posición de detección del sensor Kinect, establecida en 1.8 m. la posición del brazo robótico empezará a ser dirigida por el brazo derecho del operador y la orientación por su muñeca izquierda. Y cuando se cambie el pulsador a “0” el robot se quedará rígido y se activará la garra, siendo su apertura o cierre controlado por el movimiento del antebrazo.

Cabe resaltar que en este proceso de traslado de una sustancia tóxica por medio de un tubo de ensayo hacia un vaso de precipitado se intercambiarán los dos estados antes descritos según la necesidad del usuario. De esta forma se puede observar el proceso completo en la figura 6.6.



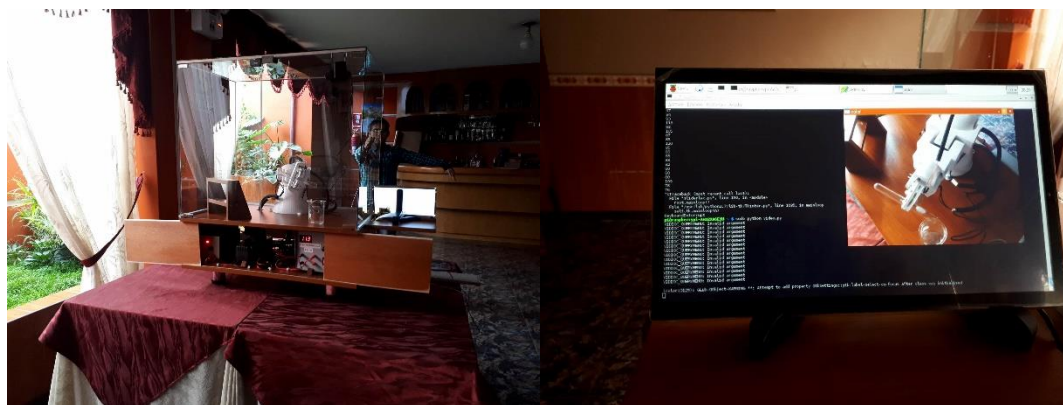


Figura 6.6: Proceso de traslado de sustancia tóxica

Fuente: Propia



CONCLUSIONES:

- Se diseñó e implementó un prototipo de un brazo robótico de 4 grados de libertad teleoperado para la manipulación de sustancias tóxicas asistido con visión artificial y redes neuronales para laboratorios farmacéuticos.
- Siguiendo la metodología propuesta, se logró realizar el diseño cinemático, dinámico y mecánico del prototipo, con el fin de asegurar el ensamble y el buen funcionamiento de cada pieza; lo que posibilitó elaborar planos y establecer especificaciones técnicas.
- Se consiguió implementar el prototipo, a partir de su previa fabricación mediante una impresora 3D y su posterior montaje, siguiendo un determinado procedimiento y agregando los componentes electrónicos necesarios para su operación.
- Se determinó algoritmos inteligentes para el procesamiento de imágenes, logrando transformar las coordenadas de las articulaciones correspondientes a las extremidades superiores del operador, en la posición y orientación del efector final del brazo robótico.
- Utilizando programación en bloques, se realizó el control cinemático del manipulador mediante redes neuronales multicapa en diferentes representaciones locales del espacio de trabajo.
- Gracias a la visión artificial y las redes neuronales que nos otorgan los valores articulares necesarios, fue posible la implementación de un sistema de control PID y Torque Computado con el propósito de lograr la teleoperación del sistema.
- Tomando en cuenta materiales de construcción y la distribución interna de los componentes del proyecto, se elaboró el diseño mecánico y la implementación del sistema de prueba; simulando así, un entorno similar a un ambiente en donde se desarrolla la aplicación planteada.

RECOMENDACIONES:

- Tomar en cuenta que los actuadores utilizados no poseen un freno mecánico, por lo que cuando no están alimentados eléctricamente, son susceptibles al efecto de la gravedad, lo que provoca que la estructura mecánica del prototipo ceda y se desplome, pudiendo ocasionar daños, principalmente al efector final; por lo tanto, se recomienda utilizar un Sistema de Potencia Ininterrumpida UPS.
- Realizar un entrenamiento previo sin carga en espacio abierto para la operación del manipulador, a fin de evitar movimientos erróneos que puedan llevar a la ruptura de los elementos a transportar o al daño colateral del robot.
- Para una buena adquisición de datos de la imagen captada por el sensor Kinect, se debe tener en cuenta la distancia del operador hacia la cámara, la iluminación del ambiente y la vestimenta a usar, la misma que debe ser pegada al cuerpo.
- Considerar el uso de un servomotor Dynamixel AX-12A como reemplazo del actuador Tower Pro SG5010, puesto que a pesar de su mayor costo se podría lograr una mejor precisión en la orientación del efector final.
- Para tener una mayor facilidad en la operación del prototipo, es recomendable el uso de un sensor Kinect for Windows, puesto que nos da la posibilidad de manejar un mayor número de puntos articulares en comparación al sensor Kinect XBOX 360, siendo de 24 frente a 20 respectivamente.
- Con el fin de mejorar la estabilidad en los movimientos del prototipo, se recomienda añadir una cantidad mayor de puntos de entrenamiento en los límites entre las zonas de cada red.

BIBLIOGRAFIA:

- [1] M. D. Sanchez García, “Estudio de la Situación de Riesgos Específicos en la industria farmacéutica”, *MAPFRE Seguridad*, vol. 70, pp. 3-13, Julio 1998.
- [2] J. Obiols Quinto, *NTP 721: Los fármacos en la industria farmacéutica (I): Exposición y riesgos para la salud*, España, 2000.
- [3] *Manual de Buenas Prácticas de Manufactura de Productos Farmacéuticos*, DIGEMID Perú, Feb. 1999.
- [4] O. Padilla Montiel, “Manipulador Teleoperado Inalámbicamente”, Ing. Mecatrónica, Departamento de Computación, Electrónica y Mecatrónica. Escuela de Ingeniería y Ciencias, Universidad de las Américas, Puebla, mayo 2008.
- [5] (2013) Yaskawa, Accedido en julio 2015. [Online]. Disponible en: http://www.motoman.es/es/noticias-y-eventos/archivo-de-noticias/news-view/?tx_ttnews%5Btt_news%5D=582&cHash=66bcec5e3e396cdd506a301b22f7cf9f
- [6] H. Zarzavilla, (2012, septiembre). Redes Neuronales, [Online], Disponible en: <http://juanitajuanabernal.blogspot.pe/p/ventajas-y-desventajas.html>
- [7] Proyecto de Grado, (2011, marzo). Métodos Deductivo e Inductivo, [Online], Disponible en: <https://proyectogrado.wordpress.com/2011/03/11/metodos-deductivo-e-inductivo/>
- [8] S. T. Alamilla Jiménez, (2014, agosto), Fundamentos de Investigación, [Online], Disponible en: <http://shounyalamilla.blogspot.pe/p/23-tipos-de-metodos-inductivo-deductivo.html>
- [9] C. Santaella, (2009, mayo), Método Inductivo, [Online], Disponible en: <http://www.monografias.com/trabajos87/metodo-inductivo/metodo-inductivo.shtml>
- [10] V. Ralac, (2014, enero), El Laboratorio Químico, [Online], Disponible en: <http://www.monografias.com/trabajos99/laboratorio-quimico/laboratorio-quimico.shtml>
- [11] Wikipedia, (2014, agosto), Laboratorio, [Online], Disponible en: <https://es.wikipedia.org/wiki/Laboratorio>
- [12] C. D. Luis, M.^a A. Camacho, G. Sánchez, G. J. Mingo, “Laboratorio farmacéutico. Materiales y equipos”, *Elaboración de preparados farmacéuticos y parafarmacéuticos*, McGraw Hill, Interamericana de España, SL, 2005
- [13] Accedido en marzo 2017, [Online], Disponible en: <http://www.lenntech.es/periodica/elementos/as.htm>

[14] Accedido en marzo 2017, [Online], Disponible en: <http://www.lenntech.es/periodica/elementos/se.htm>

[15] *Hoja Informativa sobre Sustancias Peligrosas*, Departamento de Salud y Servicios para Personas Mayores de New Jersey, Trenton, New Jersey, USA, 1998.

[16] Accedido en marzo 2017, [Online], Disponible en: <http://www.lenntech.es/periodica/elementos/hg.htm>

[17] Accedido en marzo 2016, *Manual Repetier Host/Repetier*, [Online], Disponible en: <https://www.repetier.com/>

[18] V. R. González, “Estructura de un robot industrial”, *Control y robótica*, Área de investigación, Universidad de Valladolid, 2002, [Online], Disponible en: http://platea.pntic.mec.es/vgonzalez/cyr_0204/ctrl_rob/robotica/sistema/morfologia.htm

[19] Accedido en agosto 2016, “Servomotores”, noviembre 2011, [Online], Disponible en: <https://www.slideshare.net/themigue10/>

[20] O. Sánchez García, A. Cruz Contreras, “En Aplicaciones Mecatrónicas ¿Motor o Servomotor?”, Instituto Politécnico Nacional, CIITEC, julio 2014, [Online], Disponible en: <http://www.boletin.upiita.ipn.mx/index.php/ciencia/577-cytnumero-44/1061-en-aplicaciones-mecatronicas-motor-o-servomotor>

[21] F. F. Valderrama Gutiérrez, L. Enrique Camargo, “Módulo de Robótica”, Escuela de Ciencias Básicas Tecnología e Ingeniería, Universidad Nacional Abierta y a Distancia CEAD DUITAMA, Julio 2008, [Online], Disponible en: <https://es.slideshare.net/conchaes/robotica-299011>

[22] F. Torres, J. Pomares, P. Gil, S. T. Puente, R. Aracil, “Robots y Sistemas Sensoriales”, Prentice Hall, Pearson Educación S.A., Madrid, 2002.

[23] A. Barrientos, L. F. Peñin, C. Balaguer, R. Aracil, “Fundamentos de Robótica”, McGraw Hill S.A., Interamericana de España, 2007

“Laboratorio farmacéutico. Materiales y equipos”, *Elaboración de preparados farmacéuticos y parafarmacéuticos*, McGraw Hill S.A., Interamericana de España, SL, 2005

[24] J. Guillermo, (2014, diciembre). Las redes neuronales: qué son y por qué están volviendo, [Online], Disponible en: <https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo>

[25] L. Alonso, Accedido en enero 2016, Redes Neuronales, [Online], Disponible en: avellano.usal.es/~lalonso/RNA/index.htm

[26] T. Carme, “Aplicaciones de las redes neuronales en robótica”, *Computación neuronal*, Edición a cargo de Senán Barro Ameneiro y José Mira: Cursos e Congresos da

Universidad de Santiago de Compostela nº 86 (1995): pp. 540. DL C-1718-1995. ISBN 84-8121-355-1. Paxs. 479-498.

[27] C. Platero Dueñas, "Introducción a la Visión Artificial", *Apuntes de Visión Artificial*, Departamento de Ingeniería Electrónica, Automática e Informática Industrial (ELAI), 2009, pp. 1-22.

[28] N. Fernández García, "Introducción a la Visión Artificial", *Visión Artificial Avanzada*, Departamento de Informática y Análisis Numérico, Escuela Politécnica Superior, Universidad De Córdoba, 2011, [Online], Disponible en: www.uco.es/users/ma1fegan/2011-2012/vision/Temas/Vision-artificial.pdf

[29] Centro Integrado Politécnico "ETI", Accedido en agosto 2016, Visión Artificial, *Célula de Fabricación Flexible ETI*, [Online], Disponible en: <http://www.etitudela.com/celula/downloads/visionartificial.pdf>

[30] E. Nuño Ortega, L. Basañez Villaluenga, "Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente, Institut d'Organització i Control de Sistemes Industrials, Barcelona, España, abril 2004.

[31] J. F. González de Diego, "Control Bilateral con Realimentación de esfuerzo aplicado a la cirugía cardíaca", Facultad de Informática, Universidad Politécnica de Madrid, Madrid, España, noviembre 2001.

[32] G. I. Solar González, "Control de Robots Industriales", *Aprendiendo con la robótica industrial entretenida*, Curso interactivo de robótica industrial, Santiago de Chile, Chile, 2005, [Online], Disponible en: <http://www.udesantiagoovirtual.cl/moodle2/mod/book/view.php?id=34409>



ANEXOS:



ANEXO A: "TRAYECTORIA CIRC"

```
%Limpiar el espacio de trabajo
clc
clear all
%Definir parámetros iniciales: longitudes
Lb=17.2;
L1=12.65;
L2=8.2;
L3=5;
%Definir el robot mediante sus parámetros Denavith Hartenberg
l=Link([0,Lb,0,pi/2,0]);
l(2)=Link([0,0,L1,0,0]);
l(3)=Link([0,0,0,pi/2,0]);
l(4)=Link([0,L2+L3,0,0,0]);
l(3).offset=pi/2;
R=SerialLink(l);
%Definir la posición inicial del efector final qi=0
T0 =transl([25.85 0 17.2]);
%Definir posición final del efector final
T1 =transl([-2.7893 -4.8312 41.8133]);
%Contrucción de la trayectoria
m=input('definir paso para cantidad de muestras y tiempo :');
t1=0.1;
t2=0:m:5;
s=length(t2);
t3=[t2(s)-0.1];
t=[t2(1),t1,t2(2):m:t2(s-1),t3,t2(s)];
r=jtraj(0,1,t);
TC = ctraj(T0, T1, r);
TC=transl(TC);
PX=TC(:,1);
PZ=17.2+sqrt((25.85.^2)-(PX.^2));
TC=[PX, TC(:,2), PZ];
k=length(TC);
yaw=input('ingrese la rotacion:');
d=yaw/k;
%Cinemática Inversa
q4=0;
for i=1:k
    if TC(i,1)==0 & TC(i,2)==0
        q1=0;
        cq3=((TC(i,1)).^2)+(TC(i,2)).^2+((TC(i,3)-17.2).^2)-((L2+L3).^2)-
        ((L1).^2))/(2.*(L1).(L2+L3));
        q3=real(atan(sqrt(1-((cq3).^2)))/(cq3));
        q2=real(atan((TC(i,3)-17.2)/sqrt((TC(i,1)).^2+(TC(i,2)).^2))-
        atan((sin(q3)*(L2+L3))/(L1+(cos(q3).(L2+L3)))));
        q4=real(q4+d);
    else
        q1=atan(TC(i,2)/TC(i,1));
        if TC(i,1)<0 & TC(i,2)>=0
            q1=q1+pi;
        elseif TC(i,1)<0 & TC(i,2)<0
            q1=q1-pi;
        else
            q1=q1;
        end
        cq3=((TC(i,1)).^2)+(TC(i,2)).^2+((TC(i,3)-17.2).^2)-((L2+L3).^2)-
        ((L1).^2))/(2.*(L1).(L2+L3));
        q3=real(atan(sqrt(1-((cq3).^2)))/(cq3));
        q2=real(atan((TC(i,3)-17.2)/sqrt((TC(i,1)).^2+(TC(i,2)).^2))-
        atan((sin(q3)*(L2+L3))/(L1+(cos(q3).(L2+L3)))));
        q4=real(q4+d);
    end
    QF(i,:)=[q1 q2 q3 q4];
end
%Obtener la matriz de aceleraciones del efector final por graficas
l=length(t);
a=input('ingrese el vector de aceleración lineal y angular del efector final
:');%aceleración 5m/s^2, q es una velocidad de 0.5m/s en 0.1 s
b=length(a);
for i=1:b
    for h=1:2
        ac(i,h)=a(i);
    end
end
end
```

```

for i=1:b
    for h=3:1-2
        ac(i,h)=0;
    end
end
for i=1:b
    for h=1-1:1
        ac(i,h)=-a(i);
    end
end
% Velocidades del fector final a través de la integración
for i=1:b
    for h=1:2
        v(i,h)=a(i)*t(h);
    end
end
for i=1:b
    for h=3:1-2
        v(i,h)=v(i,2);
    end
end
for i=1:b
    d2(i)=a(i)*t(1);
    for h=1-1:1
        v(i,h)=-a(i)*t(h)+d2(i);
    end
end
%Obtener las velocidades articulares atrvés del jacobiano
for f=1:1
    J=R.jacob0(QF(f,:));
    J0(:,f)=inv(transpose(J)*J)*transpose(J);
    qp(f,:)=J0(:,f)*v(:,f);
end
%Obtener las aceleraciones articulares a través de la derivada
for g=1:4
    qpp(:,g) = diff(transpose(qp(:,g)))./diff(t);
end
QF;
qp;
qpp;
figure(1)
plot3(TC(:,1), TC(:,2), TC(:,3), 'c', 'LineWidth', 2);
hold on
R.plot(QF)

%Hallamos los torques
%DATOS
%GARRA O ESLABON 4
m4=0.76545;%kg CONSIDERANDO CARGA DE 500 GR
I4=0.00026685676;%0.00038577349;%kg-m^2
lc4=0.04212;%0.03938;%m
l3=0.05;%m %18cm

%Eslabon 3
m3=0.31026;%kg
I3=0.00034269451;%0.00033528241;%kg-m^2
lc3=0.03663;%0.07495;%m
l2=0.082;%m

%Eslabon 2
m2=0.33256;%0.40089;%kg
I2=0.00150668697;%kg-m^2
lc2=0.0707;%m
l1=0.1265;%m

%Eslabon 1 o base
m1=0.64932;%0.59687;%kg
I1=0.00676334149;%kg-m^2
lc1=0.1274;%m
lb=0.172;%m

%Gravedad
g=9.81;%m/s^2

Qpp(1,:)=[0 0 0 0];
for i=2:1
    Qpp(i,:)=qpp(i-1,:);
end

```

```

end

%Torque de la garra
for i=1:l
    q1=QF(i,1);q2=QF(i,2);q3=QF(i,3);q4=QF(i,4);
    qp1=qp(i,1);qp2=qp(i,2);qp3=qp(i,3);qp4=qp(i,4);
    qpp1=Qpp(i,1);qpp2=Qpp(i,2);qpp3=Qpp(i,3);qpp4=Qpp(i,4);
    T4(i) = I4*qpp4 + I4*qpp1*sin(q2 + q3) + I4*qp1*cos(q2 + q3)*(qp2 + qp3);
end
T4
for i=1:l
    q1=QF(i,1);q2=QF(i,2);q3=QF(i,3);q4=QF(i,4);
    qp1=qp(i,1);qp2=qp(i,2);qp3=qp(i,3);qp4=qp(i,4);
    qpp1=Qpp(i,1);qpp2=Qpp(i,2);qpp3=Qpp(i,3);qpp4=Qpp(i,4);
    T3(i)=I3*qpp2 + I3*qpp3 + I4*qpp2 + I4*qpp3 + lc3^2*m3*qpp2 + lc3^2*m3*qpp3 +
    lc3^2*m4*qpp2 + lc3^2*m4*qpp3 - I4*qp1*qp4*cos(q2 + q3) + g*l2*m4*cos(q2 + q3) +
    g*lc3*m3*cos(q2 + q3) + g*lc4*m4*cos(q2 + q3) + (lc3^2*m3*qp1^2*sin(2*q2 + 2*q3))/2 +
    (lc3^2*m4*qp1^2*sin(2*q2 + 2*q3))/2 + (l1*lc3*m3*qp1^2*sin(q3))/2 +
    l1*lc3*m3*qp2^2*sin(q3) + (l1*lc3*m4*qp1^2*sin(q3))/2 + l1*lc3*m4*qp2^2*sin(q3) +
    (l1*lc3*m3*qp1^2*sin(2*q2 + q3))/2 + (l1*lc3*m4*qp1^2*sin(2*q2 + q3))/2 +
    l1*lc3*m3*qpp2*cos(q3) + l1*lc3*m4*qpp2*cos(q3);
end
T3
for i=1:l
    q1=QF(i,1);q2=QF(i,2);q3=QF(i,3);q4=QF(i,4);
    qp1=qp(i,1);qp2=qp(i,2);qp3=qp(i,3);qp4=qp(i,4);
    qpp1=Qpp(i,1);qpp2=Qpp(i,2);qpp3=Qpp(i,3);qpp4=Qpp(i,4);
    T2(i)=qpp2*(I2 + I3 + I4 + l1^2*m4 + l2^2*m3 + lc1^2*m2 + lc3^2*m3 + lc3^2*m4) +
    qpp3*(I3 + I4 + lc3^2*m3 + lc3^2*m4) + g*m4*(cos(q2 + q3)*(l2 + lc4) + l1*cos(q2)) +
    g*m3*(lc3*cos(q2 + q3) + l1*cos(q2)) - I4*qp1*qp4*cos(q2 + q3) + g*lc2*m2*cos(q2) +
    (l1^2*m3*qp1^2*sin(2*q2))/2 + (l1^2*m4*qp1^2*sin(2*q2))/2 + (lc2^2*m2*qp1^2*sin(2*q2))/2 +
    (lc3^2*m3*qp1^2*sin(2*q2 + 2*q3))/2 + (lc3^2*m4*qp1^2*sin(2*q2 + 2*q3))/2 -
    l1*lc3*qp3^2*sin(q3)*(m3 + m4) + l1*lc3*m3*qp1^2*sin(2*q2 + q3) +
    l1*lc3*m4*qp1^2*sin(2*q2 + q3) + 2*l1*lc3*qpp2*cos(q3)*(m3 + m4) +
    l1*lc3*qpp3*cos(q3)*(m3 + m4) - 2*l1*lc3*qp2*qp3*sin(q3)*(m3 + m4);
end
T2
for i=1:l
    q1=QF(i,1);q2=QF(i,2);q3=QF(i,3);q4=QF(i,4);
    qp1=qp(i,1);qp2=qp(i,2);qp3=qp(i,3);qp4=qp(i,4);
    qpp1=Qpp(i,1);qpp2=Qpp(i,2);qpp3=Qpp(i,3);qpp4=Qpp(i,4);
    TOR1(i)=qpp1*(I1 + I2 + I3 + I4 + (l1^2*m3)/2 + (l1^2*m4)/2 + (lc1^2*m2)/2 +
    (lc3^2*m3)/2 + (lc3^2*m4)/2) + I4*qpp4*sin(q2 + q3) + I4*qp4*cos(q2 + q3)*(qp2 + qp3) +
    (l1^2*qpp1*cos(2*q2)*(m3 + m4))/2 + (lc3^2*qpp1*cos(2*q2 + 2*q3)*(m3 + m4))/2 -
    (lc3^2*qp1*sin(2*q2 + 2*q3)*(2*qp2 + 2*qp3)*(m3 + m4))/2 - l1^2*qp1*qp2*sin(2*q2)*(m3 +
    m4) + l1*lc3*qpp1*cos(q3)*(m3 + m4) + l1*lc3*qpp1*cos(2*q2 + q3)*(m3 + m4) -
    l1*lc3*qp1*qp3*sin(q3)*(m3 + m4) - l1*lc3*qp1*sin(2*q2 + q3)*(m3 + m4)*(2*qp2 + qp3);
end
TOR1
T4m=max(abs(T4))
T3m=max(abs(T3))
T2m=max(abs(T2))
T1m=max(abs(TOR1))

%Buscamos la posición del mayor torque para las gráficas
for w=1:k
    if TOR1(w)==T1m
        w;
        break
    end
end
x1=t(w);
y1=TOR1(w);
for w=1:k
    if T2(w)==T2m
        w;
        break
    end
end
x2=t(w);
y2=T2(w);
for w=1:k
    if T3(w)==T3m
        w;
        break
    end
end
end
end

```



```
x3=t(w);
y3=T3(w);
for w=1:k
    if T4(w)==T4m
        w;
        break
    end
end
x4=t(w);
y4=T4(w);

figure(2)
hold on
grid on
plot(t,TOR1)
plot(t,T2,'+-r')
plot(t,T3,'*-g')
plot(t,T4,'--y')
legend('T1','T2','T3','T4')
title('Gráfica de tendencia de torques')
xlabel('Tiempo(s)')
ylabel('Torque(N-m)')

figure(3)
subplot(2,2,1)
plot(t,TOR1)
hold on
plot(x1,y1,'*k')
grid on
title('Tendencia del Torque 1')
xlabel('Tiempo(s)')
ylabel('Torque(N-m)')
subplot(2,2,2)
plot(t,T2,'r')
hold on
plot(x2,y2,'*k')
grid on
title('Tendencia del Torque 2')
xlabel('Tiempo(s)')
ylabel('Torque(N-m)')
subplot(2,2,3)
plot(t,T3,'g')
hold on
plot(x3,y3,'*k')
grid on
title('Tendencia del Torque 3')
xlabel('Tiempo(s)')
ylabel('Torque(N-m)')
subplot(2,2,4)
plot(t,T4,'y')
hold on
plot(x4,y4,'*k')
grid on
title('Tendencia del Torque 4')
xlabel('Tiempo(s)')
ylabel('Torque(N-m)')
```

ANEXO B: DATASHEET DE LOS ACTUADORES

ANEXO C: DATASHEET ADC MCP3202

ANEXO D: PRUEBA DE MOTORES

“TOWER PRO”

```
#include <Servo.h>
Servo towerpro;

const int pintower = 3;
const int pulsoMin = 620;
const int pulsoMax = 2365;

void setup(){
  towerpro.attach(pintower, pulsoMin, pulsoMax);
}

void loop() {

  towerpro.write(0);
  delay(2000);
  towerpro.write(180);
  delay(2000);
}
```

“AX12”

```
#include <DynamixelSerial.h>

void setup() {
  Dynamixel.begin(1000000,2);
  delay(1000); // put your setup code here, to run once:
}

void loop() {
  Dynamixel.moveSpeed(1,30,100);
  Dynamixel.move(2,240);
  delay(3000);
  Dynamixel.moveSpeed(1,140,100);
  Dynamixel.move(2,800);
  delay(3000);
}
```

“RX24F”

```
#include <DynamixelSerial.h>

void setup() {

  Dynamixel.begin(1000000,2);
  delay(1000);
}

void loop() {

  Dynamixel.moveSpeed(3,300,100); //
  Dynamixel.moveSpeed(4,670,100);
  delay(5000);
  Dynamixel.moveSpeed(3,500,100);
  Dynamixel.moveSpeed(4,470,100);
  delay(5000);
}
```

```
}
```

“MX28T”

```
#include <DynamixelSerial.h>
```

```
void setup() {  
  Dynamixel.begin(1000000);  
  delay(1000);
```

```
}  
void loop() {  
  Dynamixel.move(5,0);  
  delay(1000);  
  Dynamixel.move(5,4095);  
  delay(1000);  
}
```

“ARMADO”

```
#include <DynamixelSerial.h>  
#include <Servo.h>
```

```
Servo towerpro;
```

```
const int pintower = 3;  
const int pulsoMin = 620;  
const int pulsoMax = 2365;
```

```
void setup() {  
  Dynamixel.begin(1000000,2);  
  delay(1000);  
  towerpro.attach(pintower, pulsoMin, pulsoMax);
```

```
}
```

```
void loop() {  
  Dynamixel.moveSpeed(1,20,100);  
  Dynamixel.moveSpeed(2,500,100);  
  Dynamixel.moveSpeed(3,220,100);  
  Dynamixel.moveSpeed(4,750,100);  
  Dynamixel.moveSpeed(5,3200,100);  
  towerpro.write(0);  
  delay(5000);  
  Dynamixel.moveSpeed(1,170,100);  
  Dynamixel.moveSpeed(2,700,100);  
  Dynamixel.moveSpeed(3,870,100);  
  Dynamixel.moveSpeed(4,100,100);  
  Dynamixel.moveSpeed(5,2500,100);  
  towerpro.write(180);  
  delay(5000);
```

```
}
```


ANEXO E: PRUEBA ROBOT “SECUENCIA”

```
#include <DynamixelSerial.h>
#include <Servo.h>

Servo towerpro;

const int pintower = 3;
const int pulsoMin = 545;
const int pulsoMax = 2275;

void setup() {

    Dynamixel.begin(1000000,2);
    delay(1000);
    towerpro.attach(pintower, pulsoMin, pulsoMax);
    // put your setup code here, to run once:

}

void loop() {

    towerpro.write(180);
    Dynamixel.moveSpeed(1,10,100);//abierto
    Dynamixel.moveSpeed(2,520,100);//90 grados
    Dynamixel.moveSpeed(3,256,100);//10 grados
    Dynamixel.moveSpeed(4,714,100);//10 grados
    delay(2000);
    Dynamixel.moveSpeed(3,491,100);//75 grados
    Dynamixel.moveSpeed(4,479,100);//75 grados
    Dynamixel.moveSpeed(5,1024,100);//45 grados
    delay(5000);
    Dynamixel.moveSpeed(2,644,100);//130 grados
    Dynamixel.moveSpeed(3,383,100);//45 grados
    Dynamixel.moveSpeed(4,587,100);//45 grados
    delay(5000);
    Dynamixel.moveSpeed(1,140,100);//cerrado
    delay(2000);
    Dynamixel.moveSpeed(2,753,100);//130 grados
    Dynamixel.moveSpeed(3,491,100);//75 grados
    Dynamixel.moveSpeed(4,479,100);//75 grados
    delay(5000);
    Dynamixel.moveSpeed(5,2048,100);//45 grados
    delay(5000);
    towerpro.write(0);
    delay(3000);
    towerpro.write(180);
    // put your main code here, to run repeatedly:

}
```

ANEXO F: PRUEBA DE KINECT: "REALTIMEHAND"

```

clc; clear all; close all;
imaqreset;

%definimos las camaras y creamos dos objetos de video
kin=imqhwinfo('kinect');
color=imqhwinfo('kinect',1);
colorvid=videoinput('kinect',1);
depth=imqhwinfo('kinect',2);
depthvid=videoinput('kinect',2);

%configuracion de camara de color
colorsrc=getselectedsource(colorvid);

%configuracion modo skeleton de camara de profundidad
depthsrc=getselectedsource(depthvid);
depthsrc.TrackingMode='Skeleton';
depthsrc.BodyPosture='Standing';
depthsrc.CameraElevationAngle=4;

%configuracion de disparo
triggerconfig([colorvid depthvid],'manual');
colorvid.FramesPerTrigger=1;
colorvid.TriggerRepeat=inf;
depthvid.FramesPerTrigger=1;
depthvid.TriggerRepeat=inf;

%crear un mapa de conexion del esqueleto del brazo
SkeletonConnectionMap = [ %RIGTH arm
    [10 11];
    [11 12];
    ];

viewer=vision.DeployableVideoPlayer();

%iniciar la adquisicion de objetos
start([depthvid colorvid]);

himg = figure;

while ishandle(himg)
    trigger([colorvid depthvid]);
    [colormap] = getdata(colorvid);
    [depthmap,~,depthmetadata]=getdata(depthvid);

    if sum(depthmetadata.IsSkeletonTracked) > 0
        tracked=find(depthmetadata.IsSkeletonTracked);
        imshow(colormap,[0 4096])
        for i = 1:2
            skeletonjoints=depthmetadata.JointImageIndices(:, :, tracked);
            coordenadas=depthmetadata.JointWorldCoordinates(:, :, tracked);
            hold on
            X1 = [skeletonjoints(SkeletonConnectionMap(i,1),1,1)
skeletonjoints(SkeletonConnectionMap(i,2),1,1)];
            Y1 = [skeletonjoints(SkeletonConnectionMap(i,1),2,1)
skeletonjoints(SkeletonConnectionMap(i,2),2,1)];
            line(X1,Y1, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color','r');
            muneca=coordenadas(11,:)-coordenadas(10,:);
            lbrazo=sqrt(muneca(1).^2+muneca(2).^2+muneca(3).^2);
            mano=coordenadas(12,:)-coordenadas(11,:);
            lmano=sqrt(mano(1).^2+mano(2).^2+mano(3).^2);
            ptof=coordenadas(12,:)-coordenadas(10,:);
            theta=atan(ptof(2)/ptof(1));
            phi=asin(muneca(3)/lbrazo);
            beta=asin((ptof(3)-muneca(3))/lmano);
            l1=12.4;
            l2=19;
            r=l1*cos(phi)+l2*cos(beta);
            z=l1*sin(phi)+l2*sin(beta)+16.7;
            x=r*cos(-theta);
            y=r*sin(-theta);
            pos=[x y z]
            pause(0.3);
        end
    end
end

```

```
        hold off
    else
        imshow(colormap,[0 4096]);
    end

end

stop([colorvid depthvid]);
```



ANEXO G: LIBRERÍA PYTHON AX12

```
from time import sleep
from serial import Serial
import RPi.GPIO as GPIO

class Ax12:
    # important AX-12 constants
    AX_MODEL_NUMBER_L = 0
    AX_MODEL_NUMBER_H = 1
    AX_VERSION = 2
    AX_ID = 3
    AX_BAUD_RATE = 4
    AX_RETURN_DELAY_TIME = 5
    AX_CW_ANGLE_LIMIT_L = 6
    AX_CW_ANGLE_LIMIT_H = 7
    AX_CCW_ANGLE_LIMIT_L = 8
    AX_CCW_ANGLE_LIMIT_H = 9
    AX_SYSTEM_DATA2 = 10
    AX_LIMIT_TEMPERATURE = 11
    AX_DOWN_LIMIT_VOLTAGE = 12
    AX_UP_LIMIT_VOLTAGE = 13
    AX_MAX_TORQUE_L = 14
    AX_MAX_TORQUE_H = 15
    AX_RETURN_LEVEL = 16
    AX_ALARM_LED = 17
    AX_ALARM_SHUTDOWN = 18
    AX_OPERATING_MODE = 19
    AX_DOWN_CALIBRATION_L = 20
    AX_DOWN_CALIBRATION_H = 21
    AX_UP_CALIBRATION_L = 22
    AX_UP_CALIBRATION_H = 23
    AX_TORQUE_STATUS = 24
    AX_LED_STATUS = 25
    AX_CW_COMPLIANCE_MARGIN = 26
    AX_CCW_COMPLIANCE_MARGIN = 27
    AX_CW_COMPLIANCE_SLOPE = 28
    AX_CCW_COMPLIANCE_SLOPE = 29
    AX_GOAL_POSITION_L = 30
    AX_GOAL_POSITION_H = 31
    AX_GOAL_SPEED_L = 32
    AX_GOAL_SPEED_H = 33
    AX_TORQUE_LIMIT_L = 34
    AX_TORQUE_LIMIT_H = 35
    AX_PRESENT_POSITION_L = 36
    AX_PRESENT_POSITION_H = 37
    AX_PRESENT_SPEED_L = 38
    AX_PRESENT_SPEED_H = 39
    AX_PRESENT_LOAD_L = 40
    AX_PRESENT_LOAD_H = 41
    AX_PRESENT_VOLTAGE = 42
    AX_PRESENT_TEMPERATURE = 43
    AX_REGISTERED_INSTRUCTION = 44
    AX_PAUSE_TIME = 45
    AX_MOVING = 46
    AX_LOCK = 47
    AX_PUNCH_L = 48
    AX_PUNCH_H = 49

    Levels
    AX_RETURN_NONE = 0
    AX_RETURN_READ = 1
    AX_RETURN_ALL = 2
    AX_PING = 1
    AX_READ_DATA = 2
    AX_WRITE_DATA = 3
    AX_REG_WRITE = 4
    AX_ACTION = 5
    AX_RESET = 6
    AX_SYNC_WRITE = 131
    AX_RESET_LENGTH = 2
    AX_ACTION_LENGTH = 2
    AX_ID_LENGTH = 4
    AX_LR_LENGTH = 4
    AX_SRL_LENGTH = 4
    AX_RDT_LENGTH = 4
    AX_LEDALARM_LENGTH = 4
```

```

AX_SHUTDOWNALARM_LENGTH = 4
AX_TL_LENGTH = 4
AX_VL_LENGTH = 6
AX_AL_LENGTH = 7
AX_CM_LENGTH = 6
AX_CS_LENGTH = 5
AX_COMPLIANCE_LENGTH = 7
AX_CCW_CW_LENGTH = 8
AX_BD_LENGTH = 4
AX_TEM_LENGTH = 4
AX_MOVING_LENGTH = 4
AX_RWS_LENGTH = 4
AX_VOLT_LENGTH = 4
AX_LOAD_LENGTH = 4
AX_LED_LENGTH = 4
AX_TORQUE_LENGTH = 4
AX_POS_LENGTH = 4
AX_GOAL_LENGTH = 5
AX_MT_LENGTH = 5
AX_PUNCH_LENGTH = 5
AX_SPEED_LENGTH = 5
AX_GOAL_SP_LENGTH = 7
AX_BYTE_READ = 1
AX_INT_READ = 2
AX_ACTION_CHECKSUM = 250
AX_BROADCAST_ID = 254
AX_START = 255
AX_CCW_AL_L = 255
AX_CCW_AL_H = 3
AX_LOCK_VALUE = 1
LEFT = 0
RIGHT = 1
RX_TIME_OUT = 10
TX_DELAY_TIME = 0.00002
# Rpi constants
RPI_DIRECTION_PIN = 18
RPI_DIRECTION_TX = GPIO.HIGH
RPI_DIRECTION_RX = GPIO.LOW
RPI_DIRECTION_SWITCH_DELAY = 0.0001
# static variables
port = None
gpioSet = False
def __init__(self):
    if(Ax12.port == None):
        Ax12.port = Serial("/dev/ttyAMA0", baudrate=1000000, timeout=0.001)
    if(not Ax12.gpioSet):
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(Ax12.RPI_DIRECTION_PIN, GPIO.OUT)
        GPIO.setup(24,GPIO.OUT)
        Ax12.gpioSet = True
    self.direction(Ax12.RPI_DIRECTION_RX)
connectedServos = []
# Error lookup dictionary for bit masking
dictErrors = { 1 : "Input Voltage",
                2 : "Angle Limit",
                4 : "Overheating",
                8 : "Range",
                16 : "Checksum",
                32 : "Overload",
                64 : "Instruction"
              }

# Custom error class to report AX servo errors
class axError(Exception) : pass

# Servo timeout
class timeoutError(Exception) : pass

def direction(self,d):
    GPIO.output(Ax12.RPI_DIRECTION_PIN, d)
    sleep(Ax12.RPI_DIRECTION_SWITCH_DELAY)

def readData(self,id):
    self.direction(Ax12.RPI_DIRECTION_RX)
    GPIO.output(24, GPIO.HIGH)

```

```

reply = Ax12.port.read(6) # [0xff, 0xff, origin, length, error]
try:
    assert ord(reply[1]) == 0xFF
except:
    e="Timeout on servo" + str(id)
    Ax12.timeoutError(e)

try :
    length = ord(reply[4]) - 2
    error = ord(reply[5])

    if(error != 0):

        return -error
    # just reading error bit
    elif(length == 0):
        return error
    else:
        if(length > 1):
            reply = Ax12.port.read(2)
            returnValue = (ord(reply[1])<<8) + (ord(reply[0])<<0)
        else:
            reply = Ax12.port.read(1)
            returnValue = ord(reply[0])
        return returnValue

except Exception, detail:
    Ax12.axError(detail)

def move(self, id, position):
    self.direction(Ax12.RPI_DIRECTION_TX)
    Ax12.port.flushInput()
    p = [position&0xff, position>>8]
    checksum = (~(id + Ax12.AX_GOAL_LENGTH + Ax12.AX_WRITE_DATA +
Ax12.AX_GOAL_POSITION_L + p[0] + p[1]))&0xff
    outData = chr(Ax12.AX_START)
    outData += chr(Ax12.AX_START)
    outData += chr(id)
    outData += chr(Ax12.AX_GOAL_LENGTH)
    outData += chr(Ax12.AX_WRITE_DATA)
    outData += chr(Ax12.AX_GOAL_POSITION_L)
    outData += chr(p[0])
    outData += chr(p[1])
    outData += chr(checksum)
    Ax12.port.write(outData)
    sleep(Ax12.TX_DELAY_TIME)
    return self.readData(id)

def moveSpeed(self, id, position, speed):
    self.direction(Ax12.RPI_DIRECTION_TX)
    Ax12.port.flushInput()
    p = [position&0xff, position>>8]
    s = [speed&0xff, speed>>8]
    checksum = (~(id + Ax12.AX_GOAL_SP_LENGTH + Ax12.AX_WRITE_DATA +
Ax12.AX_GOAL_POSITION_L + p[0] + p[1] + s[0] + s[1]))&0xff
    outData = chr(Ax12.AX_START)
    outData += chr(Ax12.AX_START)
    outData += chr(id)
    outData += chr(Ax12.AX_GOAL_SP_LENGTH)
    outData += chr(Ax12.AX_WRITE_DATA)
    outData += chr(Ax12.AX_GOAL_POSITION_L)
    outData += chr(p[0])
    outData += chr(p[1])
    outData += chr(s[0])
    outData += chr(s[1])
    outData += chr(checksum)
    Ax12.port.write(outData)
    sleep(Ax12.TX_DELAY_TIME)
    return self.readData(id)

def readPosition(self, id):
    self.direction(Ax12.RPI_DIRECTION_TX)
    Ax12.port.flushInput()
    checksum = (~(id + Ax12.AX_POS_LENGTH + Ax12.AX_READ_DATA +
Ax12.AX_PRESENT_POSITION_L + Ax12.AX_INT_READ))&0xff
    outData = chr(Ax12.AX_START)
    outData += chr(Ax12.AX_START)

```



```
outData += chr(id)
outData += chr(Ax12.AX_POS_LENGTH)
outData += chr(Ax12.AX_READ_DATA)
outData += chr(Ax12.AX_PRESENT_POSITION_L)
outData += chr(Ax12.AX_INT_READ)
outData += chr(checksum)
Ax12.port.write(outData)
sleep(Ax12.TX_DELAY_TIME)
return self.readData(id)
```



ANEXO H: LIBRERÍA PYTHON RX24

```
from time import sleep
from serial import Serial
import RPi.GPIO as GPIO

class Rx24:
    AX_MODEL_NUMBER_L = 0
    AX_MODEL_NUMBER_H = 1
    AX_VERSION = 2
    AX_ID = 3
    AX_BAUD_RATE = 4
    AX_RETURN_DELAY_TIME = 5
    AX_CW_ANGLE_LIMIT_L = 6
    AX_CW_ANGLE_LIMIT_H = 7
    AX_CCW_ANGLE_LIMIT_L = 8
    AX_CCW_ANGLE_LIMIT_H = 9
    AX_SYSTEM_DATA2 = 10
    AX_LIMIT_TEMPERATURE = 11
    AX_DOWN_LIMIT_VOLTAGE = 12
    AX_UP_LIMIT_VOLTAGE = 13
    AX_MAX_TORQUE_L = 14
    AX_MAX_TORQUE_H = 15
    AX_RETURN_LEVEL = 16
    AX_ALARM_LED = 17
    AX_ALARM_SHUTDOWN = 18
    AX_OPERATING_MODE = 19
    AX_DOWN_CALIBRATION_L = 20
    AX_DOWN_CALIBRATION_H = 21
    AX_UP_CALIBRATION_L = 22
    AX_UP_CALIBRATION_H = 23
    AX_TORQUE_STATUS = 24
    AX_LED_STATUS = 25
    AX_CW_COMPLIANCE_MARGIN = 26
    AX_CCW_COMPLIANCE_MARGIN = 27
    AX_CW_COMPLIANCE_SLOPE = 28
    AX_CCW_COMPLIANCE_SLOPE = 29
    AX_GOAL_POSITION_L = 30
    AX_GOAL_POSITION_H = 31
    AX_GOAL_SPEED_L = 32
    AX_GOAL_SPEED_H = 33
    AX_TORQUE_LIMIT_L = 34
    AX_TORQUE_LIMIT_H = 35
    AX_PRESENT_POSITION_L = 36
    AX_PRESENT_POSITION_H = 37
    AX_PRESENT_SPEED_L = 38
    AX_PRESENT_SPEED_H = 39
    AX_PRESENT_LOAD_L = 40
    AX_PRESENT_LOAD_H = 41
    AX_PRESENT_VOLTAGE = 42
    AX_PRESENT_TEMPERATURE = 43
    AX_REGISTERED_INSTRUCTION = 44
    AX_PAUSE_TIME = 45
    AX_MOVING = 46
    AX_LOCK = 47
    AX_PUNCH_L = 48
    AX_PUNCH_H = 49

    Levels
    AX_RETURN_NONE = 0
    AX_RETURN_READ = 1
    AX_RETURN_ALL = 2
    AX_PING = 1
    AX_READ_DATA = 2
    AX_WRITE_DATA = 3
    AX_REG_WRITE = 4
    AX_ACTION = 5
    AX_RESET = 6
    AX_SYNC_WRITE = 131
    AX_RESET_LENGTH = 2
    AX_ACTION_LENGTH = 2
    AX_ID_LENGTH = 4
    AX_LR_LENGTH = 4
    AX_SRL_LENGTH = 4
    AX_RDT_LENGTH = 4
    AX_LEDALARM_LENGTH = 4
    AX_SHUTDOWNALARM_LENGTH = 4
```

```

AX_TL_LENGTH = 4
AX_VL_LENGTH = 6
AX_AL_LENGTH = 7
AX_CM_LENGTH = 6
AX_CS_LENGTH = 5
AX_COMPLIANCE_LENGTH = 7
AX_CCW_CW_LENGTH = 8
AX_BD_LENGTH = 4
AX_TEM_LENGTH = 4
AX_MOVING_LENGTH = 4
AX_RWS_LENGTH = 4
AX_VOLT_LENGTH = 4
AX_LOAD_LENGTH = 4
AX_LED_LENGTH = 4
AX_TORQUE_LENGTH = 4
AX_POS_LENGTH = 4
AX_GOAL_LENGTH = 5
AX_MT_LENGTH = 5
AX_PUNCH_LENGTH = 5
AX_SPEED_LENGTH = 5
AX_GOAL_SP_LENGTH = 7
AX_BYTE_READ = 1
AX_INT_READ = 2
AX_ACTION_CHECKSUM = 250
AX_BROADCAST_ID = 254
AX_START = 255
AX_CCW_AL_L = 255
AX_CCW_AL_H = 3
AX_LOCK_VALUE = 1
LEFT = 0
RIGHT = 1
RX_TIME_OUT = 10
TX_DELAY_TIME = 0.00002

# RPi constants
RPI_DIRECTION_PIN = 24
RPI_DIRECTION_TX = GPIO.HIGH
RPI_DIRECTION_RX = GPIO.LOW
RPI_DIRECTION_SWITCH_DELAY = 0.0001
# static variables
port = None
gpioSet = False

def __init__(self):
    if(Rx24.port == None):
        Rx24.port = Serial("/dev/ttyAMA0", baudrate=1000000, timeout=0.001)
    if(not Rx24.gpioSet):
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(Rx24.RPI_DIRECTION_PIN, GPIO.OUT)
        GPIO.setup(18,GPIO.OUT)
        Rx24.gpioSet = True
        self.direction(Rx24.RPI_DIRECTION_RX)
    connectedServos = []

# Error lookup dictionary for bit masking
dictErrors = { 1 : "Input Voltage",
                2 : "Angle Limit",
                4 : "Overheating",
                8 : "Range",
                16 : "Checksum",
                32 : "Overload",
                64 : "Instruction"
              }

# Custom error class to report AX servo errors
class rxError(Exception) : pass
# Servo timeout
class timeoutError(Exception) : pass

def direction(self,d):
    GPIO.output(Rx24.RPI_DIRECTION_PIN, d)
    sleep(Rx24.RPI_DIRECTION_SWITCH_DELAY)

def readData(self,id):
    self.direction(Rx24.RPI_DIRECTION_RX)
    GPIO.output(18, GPIO.HIGH)
    reply = Rx24.port.read(5) # [0xff, 0xff, origin, length, error]

```



```

try:
    assert ord(reply[0]) == 0xFF
except:
    e="Timeout on servo" + str(id)
    Rx24.timeoutError(e)
try :
    length = ord(reply[3]) - 2
    error = ord(reply[4])
    if(error != 0):
        print "Error from servo: " + Rx24.dictErrors[error] + ' (code ' +
hex(error) + ' )'
        return -error
    # just reading error bit
    elif(length == 0):
        return error
    else:
        if(length > 1):
            reply = Rx24.port.read(2)
            returnValue = (ord(reply[1])<<8) + (ord(reply[0])<<0)
        else:
            reply = Rx24.port.read(1)
            returnValue = ord(reply[0])
        return returnValue
except Exception, detail:
    Rx24.rxEError(detail)
self.direction(Rx24.RPI_DIRECTION_TX)

def move(self, id, position):
    self.direction(Rx24.RPI_DIRECTION_TX)
    Rx24.port.flushInput()
    p = [position&0xff, position>>8]
    checksum = (~(id + Rx24.AX_GOAL_LENGTH + Rx24.AX_WRITE_DATA +
Rx24.AX_GOAL_POSITION_L + p[0] + p[1]))&0xff
    outData = chr(Rx24.AX_START)
    outData += chr(Rx24.AX_START)
    outData += chr(id)
    outData += chr(Rx24.AX_GOAL_LENGTH)
    outData += chr(Rx24.AX_WRITE_DATA)
    outData += chr(Rx24.AX_GOAL_POSITION_L)
    outData += chr(p[0])
    outData += chr(p[1])
    outData += chr(checksum)
    Rx24.port.write(outData)
    sleep(Rx24.TX_DELAY_TIME)
    return self.readData(id)

def moveSpeed(self, id, position, speed):
    self.direction(Rx24.RPI_DIRECTION_TX)
    Rx24.port.flushInput()
    p = [position&0xff, position>>8]
    s = [speed&0xff, speed>>8]
    checksum = (~(id + Rx24.AX_GOAL_SP_LENGTH + Rx24.AX_WRITE_DATA +
Rx24.AX_GOAL_POSITION_L + p[0] + p[1] + s[0] + s[1]))&0xff
    outData = chr(Rx24.AX_START)
    outData += chr(Rx24.AX_START)
    outData += chr(id)
    outData += chr(Rx24.AX_GOAL_SP_LENGTH)
    outData += chr(Rx24.AX_WRITE_DATA)
    outData += chr(Rx24.AX_GOAL_POSITION_L)
    outData += chr(p[0])
    outData += chr(p[1])
    outData += chr(s[0])
    outData += chr(s[1])
    outData += chr(checksum)
    Rx24.port.write(outData)
    sleep(Rx24.TX_DELAY_TIME)
    return self.readData(id)

def readPosition(self, id):
    self.direction(Rx24.RPI_DIRECTION_TX)
    Rx24.port.flushInput()
    checksum = (~(id + Rx24.AX_POS_LENGTH + Rx24.AX_READ_DATA +
Rx24.AX_PRESENT_POSITION_L + Rx24.AX_INT_READ))&0xff
    outData = chr(Rx24.AX_START)
    outData += chr(Rx24.AX_START)
    outData += chr(id)

```

```
outData += chr(Rx24.AX_POS_LENGTH)
outData += chr(Rx24.AX_READ_DATA)
outData += chr(Rx24.AX_PRESENT_POSITION_L)
outData += chr(Rx24.AX_INT_READ)
outData += chr(checksum)
Rx24.port.write(outData)
sleep(Rx24.TX_DELAY_TIME)
return self.readData(id)
```



ANEXO I: LIBRERÍA PYTHON ADC

```
import spidev
import time
import RPi.GPIO as GPIO      #Importamos la libreria RPi.GPIO
spi=spidev.SpiDev()
spi.open(0,0)

class ADC:

    def __init__(self, channel):

        self.channel=channel

    def get_adc(self):
        if ((self.channel > 1) or (self.channel < 0)):
            return -1
        r=spi.xfer2([1, (2+self.channel)<<6,0])

        ret=((r[1]&0x0F) << 8)+(r[2])
        return ret
```



ANEXO J: PROGRAMACIÓN PYTHON “SLIDERLEC”

```
from Tkinter import *
import serial
from ax12 import Ax12
from rx24 import Rx24
from ADC import ADC
import time
import RPi.GPIO as GPIO
import spidev

import socket
# addressing information of target
IPADDR = '169.254.139.155'
PORTNUM = 25000

GPIO.setmode(GPIO.BCM)
GPIO.setup(24,GPIO.OUT)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
pwm = GPIO.PWM(17,50)
pwm.start(3)
port=serial.Serial("/dev/ttyAMA0",baudrate=1000000,timeout=3.0)
servos =Ax12()
x=Rx24()
adc=ADC(0)
DEBUG=0
spi=spidev.SpiDev()
spi.open(0,0)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=-10, to=190,
            orient=HORIZONTAL,length=600,width=20,tickinterval=10,label="Eslabon
1",command=self.update)
        scale.grid(row=0)

    def update(self,an):
        angle=int(round((3.25*(int(an)+10))+227))
        angle1=743-(angle-227)
        x.moveSpeed(3,angle,100)
        x.moveSpeed(4,angle1,100)
        p=x.readPosition(3)
        s=x.readPosition(4)
        print p, angle
        print s, angle1

class App1:

    def __init__(self, master1):
        frame1 = Frame(master1)
        frame1.pack()
        scale1 = Scale(frame1, from_=0, to=45,
            orient=HORIZONTAL,length=300,width=20,tickinterval=5,label="Garra
Apertura", command=self.update)
        scale1.grid(row=0)

    def update(self, an1):
        angle2=int(round((2.667*int(an1))+30))
        servos.moveSpeed(1,angle2,100)
        p1=servos.readPosition(1)
        print p1, angle2

class App2:

    def __init__(self, master2):
        frame2 = Frame(master2)
        frame2.pack()
        scale2 = Scale(frame2, from_=-90, to=90,
            orient=HORIZONTAL,length=600,width=20,tickinterval=10,label="Eslabon 2",
command=self.update)
        scale2.grid(row=0)
```

```

def update(self, an3):
    angle3=int(round((3.47*(int(an3)+90))+205))
    servos.moveSpeed(2,angle3,100)
    p2=servos.readPosition(2)
    print p2, angle3

class App3:

    def __init__(self, master3):
        frame3 = Frame(master3)
        frame3.pack()
        scale3 = Scale(frame3, from_=0, to=360,
                        orient=HORIZONTAL,length=800,width=20,tickinterval=20,label="Base",
command=self.update)
        scale3.grid(row=0)

    def update(self, an4):
        angle4=int(round(11.375*int(an4)))
        servos.moveSpeed(5,angle4,100)
        p3=servos.readPosition(5)
        print p3, angle4

class App4:

    def __init__(self, master4):
        frame4 = Frame(master4)
        frame4.pack()
        scale4 = Scale(frame4, from_=0, to=180,
orient=HORIZONTAL,length=600,width=20,label="Garra",tickinterval=10,command=self.update)
        scale4.grid(row=0)

    def update(self, angle5):

        duty = float(angle5) / 20.0 + 3
        pwm.ChangeDutyCycle(duty)

        tempval=0

        for x in range(0,9):
            tempval += adc.get_adc()

        promval =tempval/10

        V1=(adc.get_adc()*3.312)/4096

        Ang=int((V1-0.78)/((2.42-0.78)/180))

        # enter the data content of the UDP packet as hex
        data=str(Ang)
        PACKETDATA = data
        print PACKETDATA

        # initialize a socket, think of it as a cable
        # SOCK_DGRAM specifies that this is UDP
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, 0)

        # connect the socket, think of it as connecting the cable to the address location
        s.connect((IPADDR, PORTNUM))

        # send the command
        s.send(PACKETDATA)

        # close the socket
        s.close()

root = Tk()
root.wm_title('Servo Control')
app3= App3(root)
app = App(root)
app2= App2(root)
app4= App4(root)
app1= App1(root)
root.geometry("900x400+0+0")
root.mainloop()

```

ANEXO K: PROGRAMA PRINCIPAL EN PYTHON “DREAMY”

```
#
#IMPORTACION DE LIBRERIAS
import math
import numpy as np
import serial
import cv2
import sys, struct
from socket import*
from ax12 import Ax12
from rx24 import Rx24
import time
import RPi.GPIO as GPIO
import spidev
from ADC import ADC
#-----

#
#CONFIGURACIONES DE GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.IN)
GPIO.setup(24,GPIO.OUT)
GPIO.setup(17,GPIO.OUT)
pwm = GPIO.PWM(17,50)
pwm.start(3)
#-----

#
#SERIAL
DEBUG=0
port=serial.Serial("/dev/ttyAMA0",baudrate=1000000,timeout=3)
servos=Ax12()
x=Rx24()
adc=ADC(0)
adcl=ADC(1)
#-----

#
#LONGITUDES
Lb=16.7
L1=12.4
L2=9.4
L3=9.6
#-----
#=====
#CONDICION INICIAL
fuerza=0
#=====

#BUCLE DE MOTORES
while True:

    #
    #RECEPCION DE DATOS DE KINECT VIA UTP DESDE PC [6 DATOS]
    #
    #COMUNICACION
    mySocket=socket(AF_INET,SOCK_DGRAM)
    mySocket.bind((gethostbyname('0.0.0.0'),24000))
    #-----
    repeat=True
    if repeat:
        (data,addr)=mySocket.recvfrom(1024)
        s=struct.Struct('6d')
        data=s.unpack(data)
    #-----

    q1=data[0]
    q2=data[1]
    q3=data[2]
    q4=data[3]
    garra=data[4]
    error=data[5]
    error=int(error)
```



```
#
#LECTURA PULSADOR
#si es 1 movimiento del robot y si es 0 es movimiento de la garra

puls=GPIO.input(4)
if puls==0:
    ea='000'
else:
    ea='001'

#
#LECTURA SENSOR PRESION:
#si es 0 garra abierta, si es 1 garra cerrada
tempv=0
for y in range(0,9):
    tempv += adc1.get_adc()
Volt=(adc1.get_adc()*3.3)/4096
fuerza=int(round(Volt*400/2.60))
if fuerza<=450:
    fue=0
else:
    fue=1

#
#LOGICA COMBINACIONAL
#0 es movimiento de garra y 1 es posicion y orientacion del robot

if fue==0 and error==0 and puls==0:
    condicion=0
elif fue==0 and error==0 and puls==1:
    condicion=1
elif fue==0 and error==1 and puls==0:
    condicion=0
elif fue==0 and error==1 and puls==1:
    condicion=1
elif fue==1 and error==0 and puls==0:
    condicion=0
elif fue==1 and error==0 and puls==1:
    condicion=1
elif fue==1 and error==1 and puls==0:
    condicion=0
elif fue==1 and error==1 and puls==1:
    condicion=1
#-----
print fue, error, puls, condicion
#=====

#CONVERSION DE VALORES ANGULARES PARA ENVIO DE DATOS HACIA MOTORES
q1=int(round((q1*180)/math.pi))
q2=int(round((q2*180)/math.pi))
q3=int(round((q3*180)/math.pi))
q4=int(round((q4*180)/math.pi))
garra=int(round((garra*180)/math.pi))

#print 'q1: ',float(q1), 'q2: ',float(q2), 'q3: ',float(q3), 'q4: ',float(q4),
'garra: ',float(garra)
#-----
if condicion==1:
    #
    #BASE:
    if q1>=180:

        angle4=int(round(11.375*int(180)))
    elif q1<=0:
        angle4=int(round(11.375*int(0)))
    else:
        angle4=int(round(11.375*int(q1)))

    servos.moveSpeed(5,angle4,100)
    #-----
    #
    #ESLABON 1:

    if q2>=90:
        angle=int(round((3.25*(int(90)+10))+227))
        angle1=743-(angle-227)
    elif q2<=-10:
```

```

        angle=int(round((3.25*(int(-10)+10))+227))
        angle1=743-(angle-227)
    else:
        angle=int(round((3.25*(int(q2)+10))+227))
        angle1=743-(angle-227)
    x.moveSpeed(3,angle,100)
    x.moveSpeed(4,angle1,100)
    #-----
    #
    #ESLABON 2:

    if q3>=90:
        angle3=int(round((3.47*(int(90)+90))+205))
    elif q3<=-90:
        angle3=int(round((3.47*(int(-90)+90))+205))
    else:
        angle3=int(round((3.47*(int(q3)+90))+205))
    servos.moveSpeed(2,angle3,100)
    #-----
    #-----
    #
    #GARRA:

    if q4>=180:
        duty = float(180) / 20.0 + 3
    elif q4<=0:
        duty = float(0) / 20.0 + 3
    else:
        duty = float(q4) / 20.0 + 3
    pwm.ChangeDutyCycle(duty)
    #-----
    del q1
    del q2
    del q3
    del q4

elif condicion==0:

    #
    #APERTURA DE GARRA:

    if garra>=45:
        angle2=int(round((2.667*int(45))+30))
    elif garra<=0:
        angle2=int(round((2.667*int(0))+30))
    else:
        angle2=int(round((2.667*int(garra))+30))
    servos.moveSpeed(1,angle2,100)

    del q1
    del q2
    del q3
    del q4

    #
    #LECTURA APERTURA GARRA:
    gar=servos.readPosition(1)
    gar=angle2+1
    if gar == None:
        l='000'
    else:
        gar=int(round((gar-30)/2.44))
        gar=str(gar)
        k=len(gar)
        if k==3:
            gar1=str(0)
            gar2=str(gar[0])
            gar3=str(gar[1])
        elif k==2:
            gar1=str(0)
            gar2=str(gar[0])
            gar3=str(gar[1])
        elif k==1:
            gar1=str(0)
            gar2=str(0)
            gar3=str(gar[0])
        l=gar1+gar2+gar3

```

```
#-----
#=====
#LECTURA DE MOTORES
#
#LECTURA BASE:

rq1=servos.readPosition(5)
rq1=angle4+4
if rq1 == None:
    c='000'
else:
    rq1=int(round(rq1/11.375))
    rq1=str(rq1)
    b=len(rq1)
    if b==3:
        rq11=str(rq1[0])
        rq12=str(rq1[1])
        rq13=str(rq1[2])
    elif b==2:
        rq11=str(0)
        rq12=str(rq1[0])
        rq13=str(rq1[1])
    elif b==1:
        rq11=str(0)
        rq12=str(0)
        rq13=str(rq1[0])

    c=rq11+rq12+rq13

#-----
#
#LECTURA ESLABON 1:
rq2=x.readPosition(3)
if rq2 == None:
    e='000'
else:
    rq2=int(round(((rq2-227)/3.25)-10))
    rq2=str(rq2)
    d=len(rq2)
    if d==3:
        rq21=str(rq2[0])
        rq22=str(rq2[1])
        rq23=str(rq2[2])
    elif d==2:
        rq21=str(0)
        rq22=str(rq2[0])
        rq23=str(rq2[1])
    elif d==1:
        rq21=str(0)
        rq22=str(0)
        rq23=str(rq2[0])

    e=rq21+rq22+rq23

#-----
#
#LECTURA ESLABON 2:
rq3=servos.readPosition(2)
rq3=angle3+2
if rq3 == None:
    h='000'
else:
    rq3=int(round(((rq3-205)/3.47)-90))
    rq3=str(rq3)
    g=len(rq3)

    if g==3:
        rq31=str(rq3[0])
        rq32=str(rq3[1])
        rq33=str(rq3[2])
    elif g==2:
        rq31=str(0)
        rq32=str(rq3[0])
        rq33=str(rq3[1])
    elif g==1:
        rq31=str(0)
        rq32=str(0)
```



```

        rq33=str(rq3[0])

        h=rq31+rq32+rq33
#-----
#
#LECTURA GARRA:

tempval=0
for z in range(0,9):
    tempval += adc.get_adc()

promval =tempval/10

V1=(adc.get_adc()*3.312)/4096

Ang=int((V1-0.78)/((2.42-0.78)/180))
Ang=str(Ang)
i=len(Ang)

if i==3:
    Ang1=str(Ang[0])
    Ang2=str(Ang[1])
    Ang3=str(Ang[2])
elif i==2:
    Ang1=str(0)
    Ang2=str(Ang[0])
    Ang3=str(Ang[1])
elif i==1:
    Ang1=str(0)
    Ang2=str(0)
    Ang3=str(Ang[0])

j=(Ang1+Ang2+Ang3)

#-----
#
#LECTURA SENSOR PRESION:

fuerza=str(fuerza)
m=len(fuerza)
if m==3:
    fuerza1=str(fuerza[0])
    fuerza2=str(fuerza[1])
    fuerza3=str(fuerza[2])
elif m==2:
    fuerza1=str(0)
    fuerza2=str(fuerza[0])
    fuerza3=str(fuerza[1])
elif m==1:
    fuerza1=str(0)
    fuerza2=str(0)
    fuerza3=str(fuerza[0])

n=fuerza1+fuerza2+fuerza3
#-----
#=====
#ENVIO DE DATOS DE LECTURAS A MATLAB DESDE RASPBERRY A PC
#INGRESAR EL DATO EN STRING
estado=condicion
if estado==0:
    ae='000'
else:
    ae='001'
if condicion == 0:
    dato=c+e+h+j+l+n+ae
elif condicion ==1:
    dato=c+e+h+j+'000'+n+ae
PACKETDATA = dato
print dato

t = socket(AF_INET, SOCK_DGRAM)
t.connect(('169.254.37.116', 25000))

# enviar el paquete
t.send(PACKETDATA)
# cerrar socket
t.close()

```

ANEXO L: DATASHEET 74LS241

ANEXO M: DATASHEET MAX485

ANEXO N: DATASHEET SENSOR DE FUERZA MF01

ANEXO O: ESPECIFICACIONES TÉCNICAS DE ABS

ANEXO P: PLANOS DEL BRAZO ROBÓTICO



User's Manual 2006-06-14

Closer to Real, **ROBOTIS**

Dynamixel **AX-12**



Contents

1. Summary

1-1. Overview and Characteristics of AX-12 Page 2

1-2. Main Specifications Page 3

2. Dynamixel Operation

2-1. Mechanical Assembly Page 4

2-2. Connector Assembly Page 5

2-3. Dynamixel Wiring Page 6

3. Communication Protocol

3-1. Communication Overview Page 9

3-2. Instruction Packet Page 10

3-3. Status Packet Page 10

3-4. Control Table Page 12

4. Instruction Set and Examples

4-1. WRITE_DATA Page 19

4-2. READ_DATA Page 20

4-3. REG WRITE and ACTION Page 20

4-4. PING Page 21

4-5. RESET Page 22

4-6. SYNCWRITE Page 23

5. Example Page 24

Appendix Page 30

1. Dynamixel AX-12

1-1. Overview and Characteristics of AX-12

Dynamixel AX-12	The Dynamixel series robot actuator is a smart, modular actuator that incorporates a gear reducer, a precision DC motor and a control circuitry with networking functionality, all in a single package. Despite its compact size, it can produce high torque and is made with high quality materials to provide the necessary strength and structural resilience to withstand large external forces. It also has the ability to detect and act upon internal conditions such as changes in internal temperature or supply voltage. The Dynamixel series robot actuator has many advantages over similar products.
Precision Control	Position and speed can be controlled with a resolution of 1024 steps.
Compliance Driving	The degree of compliance can be adjusted and specified in controlling position.
Feedback	Feedback for angular position, angular velocity, and load torque are available.
Alarm System	The Dynamixel series robot actuator can alert the user when parameters deviate from user defined ranges (e.g. internal temperature, torque, voltage, etc) and can also handle the problem automatically (e.g. torque off)
Communication	Wiring is easy with daisy chain connection, and it support communication speeds up to 1M BPS.
Distributed Control	Position, velocity, compliance, and torque can be set with a single command packet, thus enabling the main processor to control many Dynamixel units even with very few resources.
Engineering Plastic	The main body of the unit is made with high quality engineering plastic which enables it to handle high torque loads.
Axis Bearing	A bearing is used at the final axis to ensure no efficiency degradation with high external loads.
Status LED	The LED can indicate the error status to the user.
Frames	A hinge frame and a side mount frame are included.

DYNAMIXEL

AX-12

ROBOTIS

1-2. Main Specifications

	AX-12	
Weight (g)	55	
Gear Reduction Ratio	1/254	
Input Voltage (V)	at 7V	at 10V
Final Max Holding Torque(kgf.cm)	12	16.5
Sec/60degree	0.269	0.196

Resolution	0.35°
Operating Angle	300°, Endless Turn
Voltage	7V~10V (Recommended voltage: 9.6V)
Max. Current	900mA
Operate Temperature	-5°C ~ +85°C
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Engineering Plastic

DYNAMIXEL

AX-12

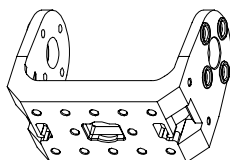
ROBOTIS

2. Dynamixel Operation

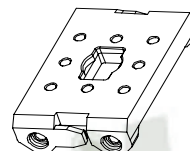
2-1. Mechanical Assembly

Frames Provided

The two frames provided with AX-12 are shown below.



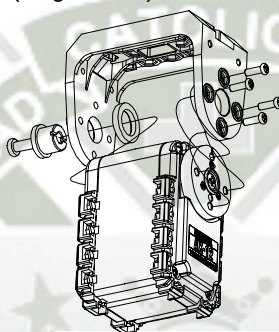
OF-12SH



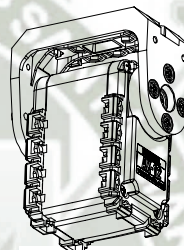
OF-12S

OF-12SH Installation

The OF-12SH (hinge frame) can be installed on the AX-12 as the following.



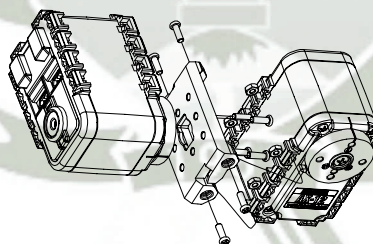
Exploded view



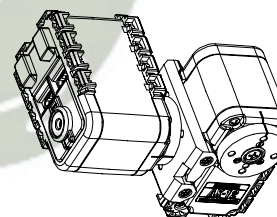
Assembled

OF-12S Installation

The OF-12S (side mount frame) can be installed on the AX-12 as the following. The OF-12S can be mounted on any of the three faces (left, right, or under side) of the AX-12 body as needed.

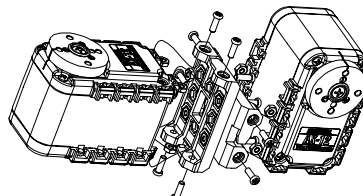


Exploded view

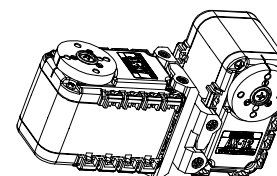


Assembled

Horn2Body

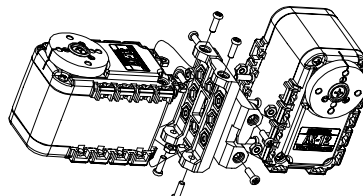


Exploded view

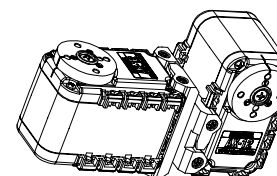


Assembled

Body2Body



Exploded view



Assembled

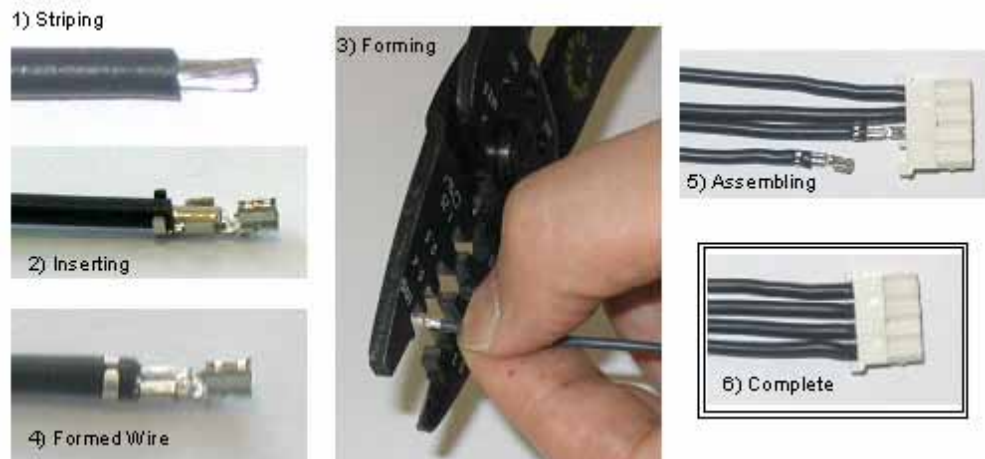
DYNAMIXEL

AX-12

ROBOTIS

2-2 . Connector Assembly

Assemble the connectors as shown below. Attach the wires to the terminals using the correct crimping tool. If you do not have access to a crimping tool, solder the terminals to the wires to ensure that they do not become loose during operation.



DYNAMIXEL

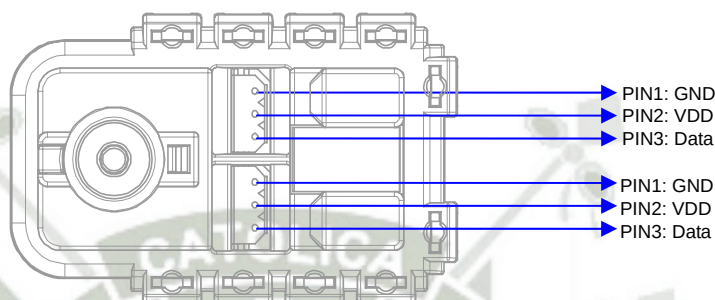
AX-12

ROBOTIS

2-3. Dynamixel Wiring

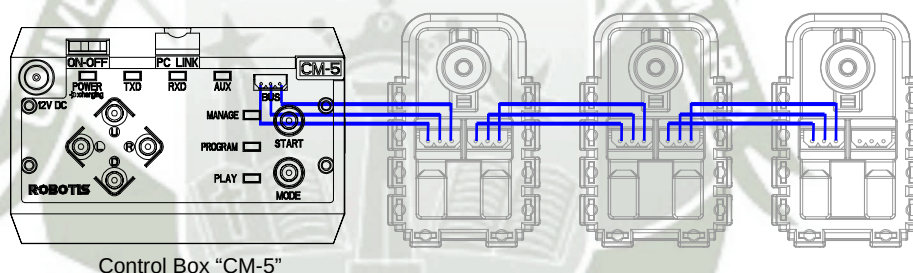
Pin Assignment

The connector pin assignments are as the following. The two connectors on the Dynamixel are connected pin to pin, thus the AX-12 can be operated with only one connector attached.



Wiring

Connect the AX-2 actuators pin to pin as shown below. Many AX-12 actuators can be controlled with a single bus in this manner.



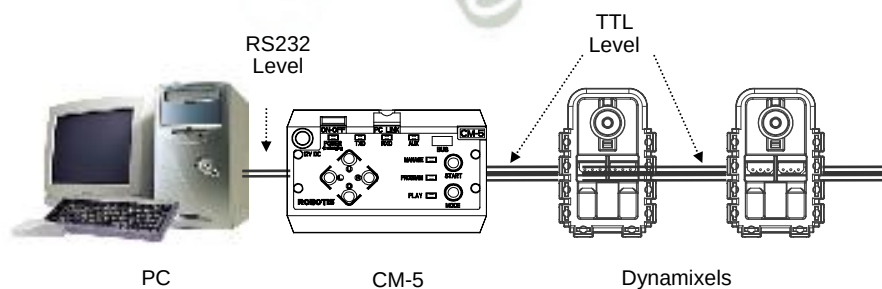
Control Box "CM-5"

Main Controller

To operate the Dynamixel actuators, the main controller must support TTL level half duplex UART. A proprietary controller can be used, but the use of the Dynamixel controller CM-5 is recommended.

PC LINK

A PC can be used to control the Dynamixel via the CM-5 controller.



DYNAMIXEL

AX-12

ROBOTIS

bioloid

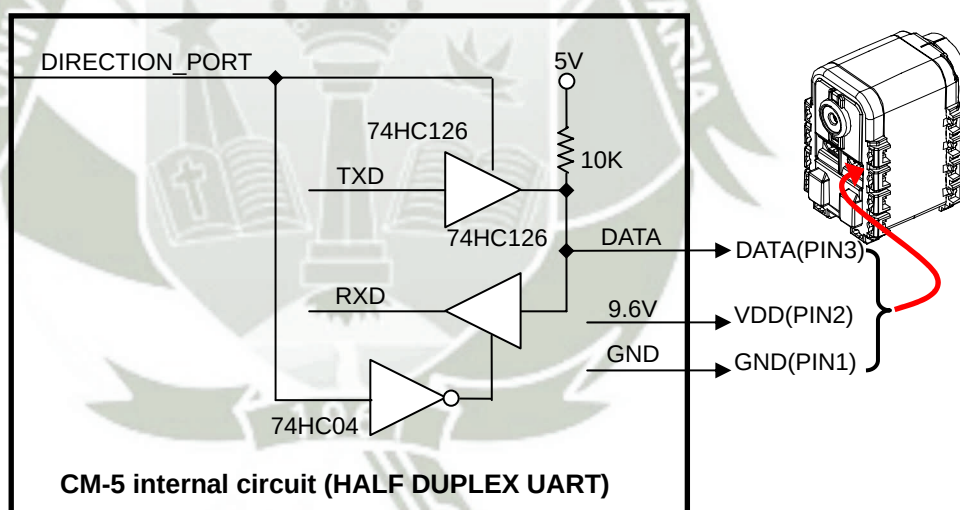
A robot can be built using only the CM-5 controller and a number of AX-12 actuators. An edutainment robotic kit named "Bioloid" is available which is based on the CM-5 controller and the AX-12 actuators.



An example of a robot built with Bioloid

For details, please refer to the Bioloid manual.

Connection to UART To control the Dynamixel actuators, the main controller needs to convert its UART signals to the half duplex type. The recommended circuit diagram for this is shown below.



The power is supplied to the Dynamixel actuator from the main controller through Pin 1 and Pin 2 of the Molex3P connector. (The circuit shown above is presented only to explain the use of half duplex UART. The CM-5 controller already has the above circuitry built in, thus the Dynamixel actuators can be directly connected to it)

The direction of data signals on the TTL level TxD and RxD depends on the DIRECTION_PORT level as the following.

DYNAMIXEL

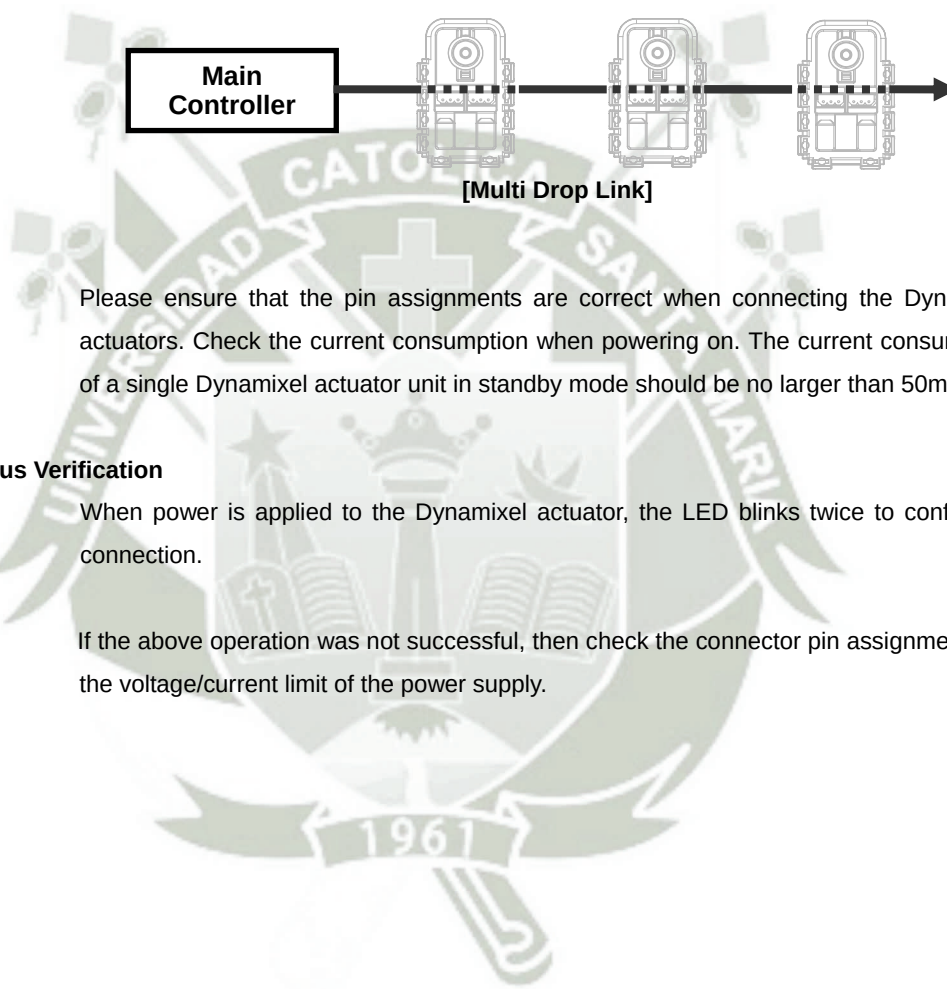
AX-12

ROBOTIS

- When the DIRECTION_PORT level is High: the signal TxD is output as Data
- When the DIRECTION_PORT level is Low: the signal Data is input as RxD

Half Duplex UART

A multi-drop method of connecting multiple Dynamixel actuators to a single node is possible by using the half duplex UART. Thus a protocol that does not allow multiple transmissions at the same time should be maintained when controlling the Dynamixel actuators.



Caution

Please ensure that the pin assignments are correct when connecting the Dynamixel actuators. Check the current consumption when powering on. The current consumption of a single Dynamixel actuator unit in standby mode should be no larger than 50mA

Connection Status Verification

When power is applied to the Dynamixel actuator, the LED blinks twice to confirm its connection.

Inspection

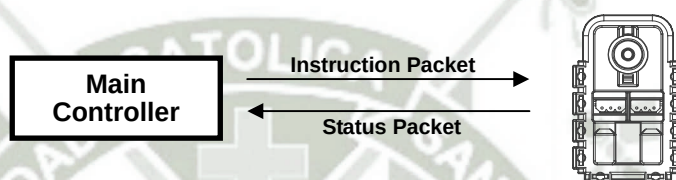
If the above operation was not successful, then check the connector pin assignment and the voltage/current limit of the power supply.

3. Communication Protocol

3-1. Communication Overview

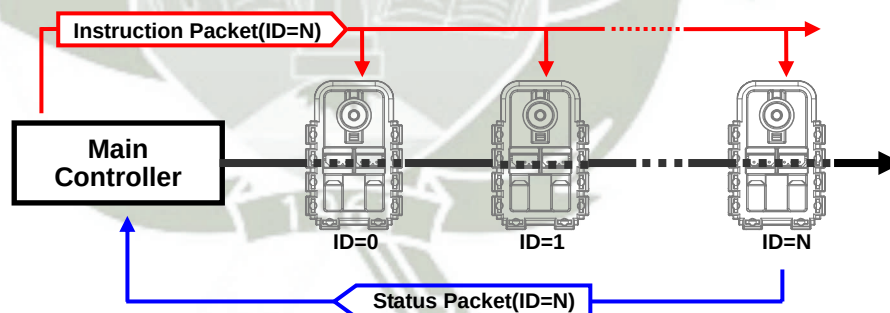
Packet

The main controller communicates with the Dynamixel units by sending and receiving data packets. There are two types of packets; the "Instruction Packet" (sent from the main controller to the Dynamixel actuators) and the "Status Packet" (sent from the Dynamixel actuators to the main controller.)



Communication

For the system connection below, if the main controller sends an instruction packet with the ID set to N, only the Dynamixel unit with this ID value will return its respective status packet and perform the required instruction.



Unique ID

If multiple Dynamixel units have the same ID value, multiple packets sent simultaneously collide, resulting in communication problems. Thus, it is imperative that no Dynamixel units share the same ID in a network node.

Protocol

The Dynamixel actuators communicate through asynchronous serial communication with 8 bit, 1 stop bit and no parity.

DYNAMIXEL

AX-12

ROBOTIS

3-2. Instruction Packet

The Instruction Packet is the packet sent by the main controller to the Dynamixel units to send commands. The structure of the Instruction Packet is as the following.

Instruction Packet 0XFF 0XFF ID LENGTH INSTRUCTION PARAMETER1 ... PARAMETER N CHECK SUM

The meanings of each packet byte definition are as the following.

0XFF 0XFF

The two 0XFF bytes indicate the start of an incoming packet.

ID

The unique ID of a Dynamixel unit. There are 254 available ID values, ranging from 0X00 to 0XFD.

Broadcasting ID

ID 0XFE is the Broadcasting ID which indicates all of the connected Dynamixel units. Packets sent with this ID apply to all Dynamixel units on the network. Thus packets sent with a broadcasting ID will not return any status packets.

LENGTH

The length of the packet where its value is "Number of parameters (N) + 2"

INSTRUCTION

The instruction for the Dynamixel actuator to perform.

PARAMETER0...N

Used if there is additional information needed to be sent other than the instruction itself.

CHECK SUM

The computation method for the 'Check Sum' is as the following.

Check Sum = $\sim (ID + Length + Instruction + Parameter1 + \dots + Parameter N)$

If the calculated value is larger than 255, the lower byte is defined as the checksum value.

$\sim$ represents the NOT logic operation.

3-3. Status Packet(Return Packet)

The Status Packet is the response packet from the Dynamixel units to the Main Controller after receiving an instruction packet. The structure of the status packet is as the following.

0XFF 0XFF ID LENGTH ERROR PARAMETER1 PARAMETER2... PARAMETER N CHECK SUM

DYNAMIXEL

AX-12

ROBOTIS

The meanings of each packet byte definition are as the following.

0XFF 0XFF

The two 0XFF bytes indicate the start of the packet.

ID

The unique ID of the Dynamixel unit returning the packet. The initial value is set to 1.

LENGTH

The length of the packet where its value is "Number of parameters (N) + 2"

ERROR

The byte representing errors sent from the Dynamixel unit. The meaning of each bit is as the following.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range.
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETER0...N

Used if additional information is needed.

CHECK SUM

The computation method for the 'Check Sum' is as the following.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)

If the calculated value is larger than 255, the lower byte is defined as the checksum value. ~ represents the NOT logic operation.

DYNAMIXEL

AX-12

ROBOTIS

3-4. Control
Table

EEPROM
Area

RAM
Area

Address	Item	Access	Initial Value
0(0X00)	Model Number(L)	RD	12(0x0C)
1(0X01)	Model Number(H)	RD	0(0x00)
2(0X02)	Version of Firmware	RD	?
3(0X03)	ID	RD,WR	1(0x01)
4(0X04)	Baud Rate	RD,WR	1(0x01)
5(0X05)	Return Delay Time	RD,WR	250(0xFA)
6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
10(0x0A)	(Reserved)	-	0(0x00)
11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
16(0X10)	Status Return Level	RD,WR	2(0x02)
17(0X11)	Alarm LED	RD,WR	4(0x04)
18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
19(0X13)	(Reserved)	RD,WR	0(0x00)
20(0X14)	Down Calibration(L)	RD	?
21(0X15)	Down Calibration(H)	RD	?
22(0X16)	Up Calibration(L)	RD	?
23(0X17)	Up Calibration(H)	RD	?
24(0X18)	Torque Enable	RD,WR	0(0x00)
25(0X19)	LED	RD,WR	0(0x00)
26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0
34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
36(0X24)	Present Position(L)	RD	?
37(0X25)	Present Position(H)	RD	?
38(0X26)	Present Speed(L)	RD	?
39(0X27)	Present Speed(H)	RD	?
40(0X28)	Present Load(L)	RD	?
41(0X29)	Present Load(H)	RD	?
42(0X2A)	Present Voltage	RD	?
43(0X2B)	Present Temperature	RD	?
44(0X2C)	Registered Instruction	RD,WR	0(0x00)
45(0X2D)	(Reserved)	-	0(0x00)
46(0x2E)	Moving	RD	0(0x00)
47(0x2F)	Lock	RD,WR	0(0x00)
48(0x30)	Punch(L)	RD,WR	32(0x20)
49(0x31)	Punch(H)	RD,WR	0(0x00)

DYNAMIXEL

AX-12

ROBOTIS

Control Table

The Control Table contains information on the status and operation of the Dynamixel actuator. The Dynamixel actuator is operated by writing values to its control table and its status is checked by reading values off its control table.

RAM and EEPROM

The data values for the RAM area will be set to the default initial values whenever the power is turned on. However, the data values for the EEPROM area are non-volatile and will still remain even after the power is turned off.

Initial Value

The Initial Value column on the right side of the control table shows the Factory Default Values for the case of EEPROM area data, and shows the initial value when the power is turned on for the case of RAM area data.

The following explains the meaning of data stored in each of the addresses in the control table.

Address 0x00,0x01

Model Number. For AX-12, this value is 0X000C (12).

Address 0x02

Firmware Version.

Address 0x03

ID. The unique ID number assigned to each Dynamixel actuators for identifying them. Different IDs are required for each Dynamixel actuators that are on the same network.

Address 0x04

Baud Rate. Determines the communication speed. The computation is done by the following formula.

$$\text{Speed (BPS)} = 2000000 / (\text{Address4} + 1)$$

Data Value for each Major Baud Rate

Adress4	Hex	Set BPS	Goal BPS	Error
1	0X01	1000000.0	1000000.0	0.000%
3	0X03	500000.0	500000.0	0.000%
4	0X04	400000.0	400000.0	0.000%
7	0X07	250000.0	250000.0	0.000%
9	0X09	200000.0	200000.0	0.000%
16	0X10	117647.1	115200.0	-2.124%
34	0X22	57142.9	57600.0	0.794%
103	0X67	19230.8	19200.0	-0.160%
207	0XCF	9615.4	9600.0	-0.160%

Note

A maximum Baud Rate error of 3% is within the tolerance of UART communication.

Caution

The initial value of Baudrate is set to 1(1000000bps)

DYNAMIXEL

AX-12

ROBOTIS

Address 0x05 **Return Delay Time.** The time it takes for the Status Packet to return after the Instruction Packet is sent. The delay time is given by $2\mu\text{Sec} * \text{Address5 value}$.

Address 0x06,0x07,0x08,0x09

Operating Angle Limit. Sets the Dynamixel actuator's operating angle range. The Goal Position needs to be within the range of: $\text{CW Angle Limit} \leq \text{Goal Position} \leq \text{CCW Angle Limit}$. An Angle Limit Error will occur if the Goal Position is set outside this range set by the operating angle limits.

Address 0x0B

the Highest Limit Temperature. The upper limit of the Dynamixel actuator's operating temperature. If the internal temperature of the Dynamixel actuator gets higher than this value, the Over Heating Error Bit (Bit 2 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are in Degrees Celsius.

Address 0x0C,0x0D

the Lowest (Highest) Limit Voltage. The upper and lower limits of the Dynamixel actuator's operating voltage. If the present voltage (Address 42) is out of the specified range, a Voltage Range Error Bit (Bit 0 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are 10 times the actual voltage value. For example, if the Address 12 value is 80, then the lower voltage limit is set to 8V.

Address 0x0E,0x0F, 0x22,0x23

Max Torque. The maximum torque output for the Dynamixel actuator. When this value is set to 0, the Dynamixel actuator enters the Free Run mode. There are two locations where this maximum torque limit is defined; in the EEPROM (Address 0x0E, 0x0F) and in the RAM (Address 0x22, 0x23). When the power is turned on, the maximum torque limit value defined in the EEPROM is copied to the location in the RAM. The torque of the Dynamixel actuator is limited by the values located in the RAM (Address 0x22, 0x23).

Address 0x10

Status Return Level. Determines whether the Dynamixel actuator will return a Status Packet after receiving an Instruction Packet.

Address16	Returning the Status Packet
0	Do not respond to any instructions
1	Respond only to READ_DATA instructions
2	Respond to all instructions

DYNAMIXEL

AX-12

ROBOTIS

In the case of an instruction which uses the Broadcast ID (0xFE) the Status Packet will not be returned regardless of the Address 0x10 value.

Address 0X11

Alarm LED. If the corresponding Bit is set to 1, the LED blinks when an Error occurs.

Bit	Function
Bit 7	0
Bit 6	If set to 1, the LED blinks when an Instruction Error occurs
Bit 5	If set to 1, the LED blinks when an Overload Error occurs
Bit 4	If set to 1, the LED blinks when a Checksum Error occurs
Bit 3	If set to 1, the LED blinks when a Range Error occurs
Bit 2	If set to 1, the LED blinks when an Overheating Error occurs
Bit 1	If set to 1, the LED blinks when an Angle Limit Error occurs
Bit 0	If set to 1, the LED blinks when an Input Voltage Error occurs

This function operates following the “OR” logical operation of all bits. For example, if the value is set to 0X05, the LED will blink when an Input Voltage Error occurs or when an Overheating Error occurs. Upon returning to a normal condition from an error state, the LED stops blinking after 2 seconds.

Address 0X12

Alarm Shutdown. If the corresponding Bit is set to a 1, the Dynamixel actuator’s torque will be turned off when an error occurs.

Bit	Function
Bit 7	0
Bit 6	If set to 1, torque off when an Instruction Error occurs
Bit 5	If set to 1, torque off when an Overload Error occurs
Bit 4	If set to 1, torque off when a Checksum Error occurs
Bit 3	If set to 1, torque off when a Range Error occurs
Bit 2	If set to 1, torque off when an Overheating Error occurs
Bit 1	If set to 1, torque off when an Angle Limit Error occurs
Bit 0	If set to 1, torque off when an Input Voltage Error occurs

This function operates following the “OR” logical operation of all bits. However, unlike the Alarm LED, after returning to a normal condition, it maintains the torque off status. To recover, the Torque Enable (Address 0X18) needs to be reset to 1.

Address 0x14~0x17

Calibration. Data used for compensating for the differences between the potentiometers used in the Dynamixel units. The user cannot change this data.

The following (from Address 0x18) is in the RAM area.

DYNAMIXEL

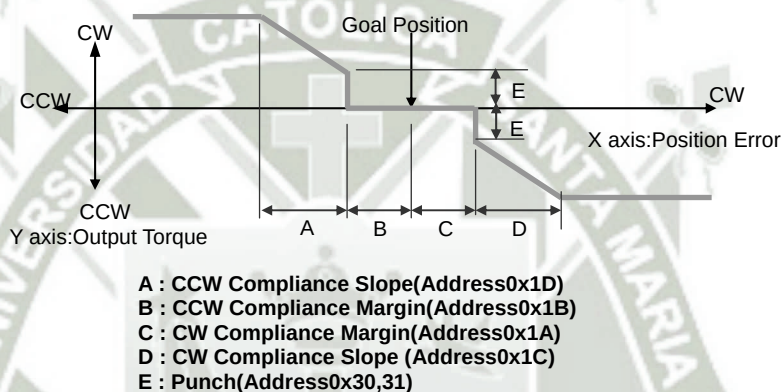
AX-12

ROBOTIS

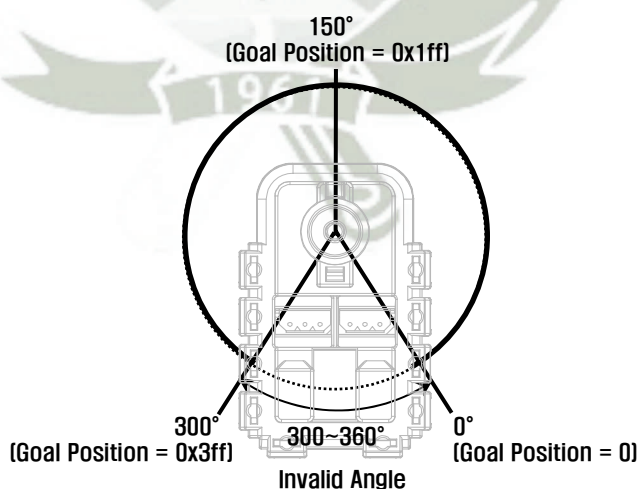
Address 0x18 **Torque Enable.** When the power is first turned on, the Dynamixel actuator enters the Torque Free Run condition (zero torque). Setting the value in Address 0x18 to 1 enables the torque.

Address 0x19 **LED.** The LED turns on when set to 1 and turns off if set to 0.

Address 0x1A~0x1D **Compliance Margin and Slope.** The compliance of the Dynamixel actuator is defined by setting the compliance Margin and Slope. This feature can be utilized for absorbing shocks at the output shaft. The following graph shows how each compliance value (length of A, B, C & D) is defined by the Position Error and applied torque.



Address 0x1E,0x1F **Goal Position** Requested angular position for the Dynamixel actuator output to move to. Setting this value to 0x3ff moves the output shaft to the position at 300°.



DYNAMIXEL

AX-12

ROBOTIS

Address 0x20,0x21 **Moving Speed.** Sets the angular velocity of the output moving to the Goal Position. Setting this value to its maximum value of 0x3ff moves the output with an angular velocity of 114 RPM, provided that there is enough power supplied (The lowest velocity is when this value is set to 1. When set to 0, the velocity is the largest possible for the supplied voltage, e.g. no velocity control is applied.)

Address 0x24,0x25 **Present Position.** Current angular position of the Dynamixel actuator output.

Address 0x26,0x27 **Present Speed.** Current angular velocity of the Dynamixel actuator output.

Address 0x28,0x29 **Present Load.** The magnitude of the load on the operating Dynamixel actuator. Bit 10 is the direction of the load.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Load Value									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Address 0x2A **Present Voltage.** The voltage currently applied to the Dynamixel actuator. The value is 10 times the actual voltage. For example, 10V is represented as 100 (0x64).

Address 0x2B **Present Temperature.** The internal temperature of the Dynamixel actuator in Degrees Celsius.

Address 0x2C **Registered Instruction.** Set to 1 when an instruction is assigned by the REG_WRITE command. Set to 0 after it completes the assigned instruction by the Action command.

Address 0x2E **Moving.** Set to 1 when the Dynamixel actuator is moving by its own power.

Address 0x2F **Lock.** If set to 1, only Address 0x18 to 0x23 can be written to and other areas cannot. Once locked, it can only be unlocked by turning the power off.

Address 0x30,0x31 **Punch.** The minimum current supplied to the motor during operation. The initial value is set to 0x20 and its maximum value is 0x3ff.

Endless Turn If both values for the CW Angle Limit and the CCW Angle Limit are set to 0, an Endless Turn mode can be implemented by setting the Goal Speed. This feature can be used for implementing a continuously rotating wheel.

DYNAMIXEL

AX-12

ROBOTIS

Goal Speed Setting

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Turn Direction	Speed Value									

Turn Direction = 0 : CCW Direction Turn, Load Direction = 1: CW Direction Turn

Range

Each data has a valid minimum and maximum values. Write instructions made outside of these valid ranges will return an error. The following table summarizes the data range for each register. 16 bit data registers are indicated with two bytes (L) and (H). Both bytes need to be written at the same time as one instruction packet.

Write Address	Writing Item	Length (bytes)	Min	Max
3(0X03)	ID	1	0	253(0xfd)
4(0X04)	Baud Rate	1	0	254(0xfe)
5(0X05)	Return Delay Time	1	0	254(0xfe)
6(0X06)	CW Angle Limit	2	0	1023(0x3ff)
8(0X08)	CCW Angle Limit	2	0	1023(0x3ff)
11(0X0B)	the Highest Limit Temperature	1	0	150(0x96)
12(0X0C)	the Lowest Limit Voltage	1	50(0x32)	250(0xfa)
13(0X0D)	the Highest Limit Voltage	1	50(0x32)	250(0xfa)
14(0X0E)	Max Torque	2	0	1023(0x3ff)
16(0X10)	Status Return Level	1	0	2
17(0X11)	Alarm LED	1	0	127(0x7f)
18(0X12)	Alarm Shutdown	1	0	127(0x7f)
19(0X13)	(Reserved)	1	0	1
24(0X18)	Torque Enable	1	0	1
25(0X19)	LED	1	0	1
26(0X1A)	CW Compliance Margin	1	0	254(0xfe)
27(0X1B)	CCW Compliance Margin	1	0	254(0xfe)
28(0X1C)	CW Compliance Slope	1	1	254(0xfe)
29(0X1D)	CCW Compliance Slope	1	1	254(0xfe)
30(0X1E)	Goal Position	2	0	1023(0x3ff)
32(0X20)	Moving Speed	2	0	1023(0x3ff)
34(0X22)	Torque Limit	2	0	1023(0x3ff)
44(0X2C)	Registered Instruction	1	0	1
47(0X2F)	Lock	1	1	1
48(0X30)	Punch	2	0	1023(0x3ff)

[Control Table Data Range and Length for Writing]

4. Instruction Set and Examples

The following Instructions are available.

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACTION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4~

4-1. WRITE_DATA

Function	To write data into the control table of the Dynamixel actuator
Length	N+3 (N is the number of data to be written)
Instruction	0X03
Parameter1	Starting address of the location where the data is to be written
Parameter2	1st data to be written
Parameter3	2nd data to be written
Parameter N+1	Nth data to be written

Example 1

Setting the ID of a connected Dynamixel actuator to 1

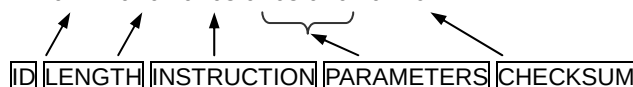
Write 1 to address 3 of the control table. The ID is transmitted using the Broadcasting ID (0xFE).

DYNAMIXEL

AX-12

ROBOTIS

Instruction Packet : 0XFF 0XFF 0XFE 0X04 0X03 0X03 0X01 0XF6`



Because it was transmitted with a Broadcast ID (0XFE), no status packets are returned.

4-2. READ_DATA

Function	Read data from the control table of a Dynamixel actuator
Length	0X04
Instruction	0X02
Parameter1	Starting address of the location where the data is to be read
Parameter2	Length of the data to be read

Example 2

Reading the internal temperature of the Dynamixel actuator with an ID of 1

Read 1 byte from address 0x2B of the control table.

Instruction Packet : 0XFF 0XFF 0X01 0X04 0X02 0X2B 0X01 0XCC`



The returned Status Packet will be as the following.

Status Packet : 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDB



The data read is 0x20. Thus the current internal temperature of the Dynamixel actuator is approximately 32°C (0X20).

4-3. REG_WRITE ACTION

4-3-1. REG_WRITE

Function	The REG_WRITE instruction is similar to the WRITE_DATA instruction, but the
-----------------	---

DYNAMIXEL

AX-12

ROBOTIS

execution timing is different. When the Instruction Packet is received the values are stored in the Buffer and the Write instruction is under a standby status. At this time, the Registered Instruction register (Address 0x2C) is set to 1. After the Action Instruction Packet is received, the registered Write instruction is finally executed.

Length	N+3 (N is the number of data to be written)
Instruction	0X04
Parameter1	Starting address of the location where the data is to be written
Parameter2	1st data to be written
Parameter3	2nd data to be written
Parameter N+1	Nth data to be written

4-3-2. ACTION

Function	Triggers the action registered by the REG_WRITE instruction
Length	0X02
Instruction	0X05
Parameter	NONE

The ACTION instruction is useful when multiple Dynamixel actuators need to move simultaneously. When controlling multiple Dynamixel actuator units, slight time delays can occur between the 1st and last units to receive an instruction. The Dynamixel actuator handles this problem by using the ACTION instruction.

Broadcasting	The Broadcast ID (0XFE) is used when sending ACTION instructions to more than two Dynamixel actuators. Note that no packets are returned by this operation.
---------------------	---

4-4. PING

Function	Does not command any operations. Used for requesting a status packet or to check the existence of a Dynamixel actuator with a specific ID.
Length	0X02
Instruction	0X01
Parameter	NONE

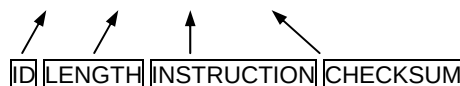
DYNAMIXEL

AX-12

ROBOTIS

Example 3
Obtaining the status packet of the Dynamixel actuator with an ID of 1

Instruction Packet : 0xFF 0xFF 0x01 0x02 0x01 0xFB`



The returned Status Packet is as the following

Status Packet : 0xFF 0xFF 0x01 0x02 0x00 0xFC



Regardless of whether the Broadcasting ID is used or the Status Return Level (Address 16) is 0, a Status Packet is always returned by the PING instruction.

4-5. RESET

Function

Changes the control table values of the Dynamixel actuator to the Factory Default Value settings

Length

0x02

Instruction

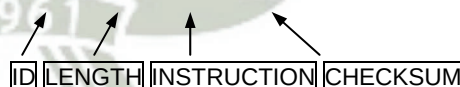
0x06

Parameter

NONE

Example 4
Resetting the Dynamixel actuator with an ID of 0

Instruction Packet : 0xFF 0xFF 0x00 0x02 0x06 0xF7`



The returned Status Packet is as the following

Status Packet : 0xFF 0xFF 0x00 0x02 0x00 0xFD



Note the ID of this Dynamixel actuator is now changed to 1 after the RESET instruction.

DYNAMIXEL

AX-12

ROBOTIS

4-6. SYNC WRITE

Function

Used for controlling many Dynamixel actuators at the same time. The communication time decreases by the Synch Write instruction since many instructions can be transmitted by a single instruction. However, you can use this instruction only when the lengths and addresses of the control table to be written to are the same. Also, the broadcasting ID needs to be used for transmitting.

ID

0xFE

Length

$(L + 1) * N + 4$ (L: Data length for each Dynamixel actuator, N: The number of Dynamixel actuators)

Instruction

0x83

Parameter1

Starting address of the location where the data is to be written

Parameter2

The length of the data to be written (L)

Parameter3

The ID of the 1st Dynamixel actuator

Parameter4

The 1st data for the 1st Dynamixel actuator

Parameter5

The 2nd data for the 1st Dynamixel actuator

...

Parameter L+3

The Lth data for the 1st Dynamixel actuator

Data for the 1st Dynamixel actuator

Parameter L+4

The ID of the 2nd Dynamixel actuator

Parameter L+5

The 1st data for the 2nd Dynamixel actuator

Parameter L+6

The 2nd data for the 2nd Dynamixel actuator

Data for the 2nd Dynamixel actuator

...

Parameter 2L+4

The Lth data for the 2nd Dynamixel actuator

....

Example 5

Setting the following positions and velocities for 4 Dynamixel actuators

Dynamixel actuator with an ID of 0: to position 0x010 with a speed of 0x150

Dynamixel actuator with an ID of 1: to position 0x220 with a speed of 0x360

Dynamixel actuator with an ID of 2: to position 0x030 with a speed of 0x170

Dynamixel actuator with an ID of 0: to position 0x220 with a speed of 0x380

Instruction Packet : 0xFF 0xFF 0xFE 0x18 0x83 0x1E 0x04 0x00 0x10 0x00 0x50
0x01 0x01 0x20 0x02 0x60 0x03 0x02 0x30 0x00 0x70 0x01 0x03 0x20 0x02 0x80
0x03 0x12

No status packets are returned since the Broadcasting ID was used.

5. Example

For the following examples, we assume a Dynamixel actuator with an ID of 1 in Reset status and that the Baud rate is 57142 BPS.

Example 6

Reading the Model Number and Firmware Version of the Dynamixel actuator with an ID of 1

Instruction Packet Instruction = READ_DATA, Address = 0x00, Length = 0x03

Communication ->[Dynamixel]:FF FF 01 04 02 00 03 F5 (LEN:008)
<-[Dynamixel]:FF FF 01 05 00 74 00 08 7D (LEN:009)

Status Packet Result Model Number = 116 (0x74) (for the case of DX-116) Firmware Version = 0x08

Example 7

Changing the ID to 0 for a Dynamixel actuator with an ID of 1

Instruction Packet Instruction = WRITE_DATA, Address = 0x03, DATA = 0x00

Communication ->[Dynamixel]:FF FF 01 04 03 03 00 F4 (LEN:008)
<-[Dynamixel]:FF FF 01 02 00 FC (LEN:006)

Status Packet Result NO ERROR

Example 8

Changing the Baud Rate of a Dynamixel actuator to 1M bps

Instruction Packet Instruction = WRITE_DATA, Address = 0x04, DATA = 0x01

Communication ->[Dynamixel]:FF FF 00 04 03 04 01 F3 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 9

Resetting the Return Delay Time to 4 uSec for a Dynamixel actuator with an ID of 0

A Return Delay Time Value of 1 corresponds to 2uSec.

DYNAMIXEL

AX-12

ROBOTIS

Instruction Packet Instruction = WRITE_DATA, Address = 0x05, DATA = 0x02

Communication ->[Dynamixel]:FF FF 00 04 03 05 02 F1 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

It is recommended to set the Return Delay Time to the minimum value allowed by the Main Controller.

Example 10

Limiting the operating angle range to 0°~150° for a Dynamixel actuator with an ID of 0

Since the CCW Angle Limit of 0x3ff corresponds to 300°, the angle 150° is represented by the value 0x1ff

Instruction Packet Instruction = WRITE_DATA, Address = 0x08, DATA = 0xff, 0x01

Communication ->[Dynamixel]:FF FF 00 05 03 08 FF 01 EF (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 11

Resetting the upper limit for the operating temperature to 80°C for a Dynamixel actuator with an ID of 0

Instruction Packet Instruction = WRITE_DATA, Address = 0x0B, DATA = 0x50

Communication ->[Dynamixel]:FF FF 00 04 03 0B 50 9D (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 12

Setting the operating voltage to 10V ~ 17V for a Dynamixel actuator with an ID of 0

10V is represented by 100 (0x64), and 17V by 170 (0xAA).

Instruction Packet Instruction = WRITE_DATA, Address = 0x0C, DATA = 0x64, 0xAA

Communication ->[Dynamixel]:FF FF 00 05 03 0C 64 AA DD (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

DYNAMIXEL

AX-12

ROBOTIS

Example 13

Setting the maximum torque to 50% of its maximum possible value for a Dynamixel actuator with an ID of 0

Set the MAX Torque value located in the ROM area to 0x1ff which is 50% of the maximum value 0x3ff.

Instruction Packet Instruction = WRITE_DATA, Address = 0x0E, DATA = 0xff, 0x01

Communication ->[Dynamixel]:FF FF 00 05 03 0E FF 01 E9 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR
To verify the effect of the adjusted Max Torque value, the power needs to be turned off and then on.

Example 14

Set the Dynamixel actuator with an ID of 0 to never return a Status Packet

Instruction Packet Instruction = WRITE_DATA, Address = 0x10, DATA = 0x00

Communication ->[Dynamixel]:FF FF 00 04 03 10 00 E8 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR
The Status Packet is not returned starting with the following instruction.

Example 15

Set the Alarm to blink the LED and Shutdown (Torque off) the actuator when the operating temperature goes over the set limit

Since the Overheating Error is Bit 2, set the Alarm value to 0x04.

Instruction Packet Instruction = WRITE_DATA, Address = 0x11, DATA = 0x04, 0x04

Communication ->[Dynamixel]:FF FF 00 05 03 11 04 04 DE (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 16

Turn on the LED and Enable Torque for a Dynamixel actuator with an ID of 0

Instruction Packet Instruction = WRITE_DATA, Address = 0x18, DATA = 0x01, 0x01

DYNAMIXEL

AX-12

ROBOTIS

Communication ->[Dynamixel]:FF FF 00 05 03 18 01_01 DD (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

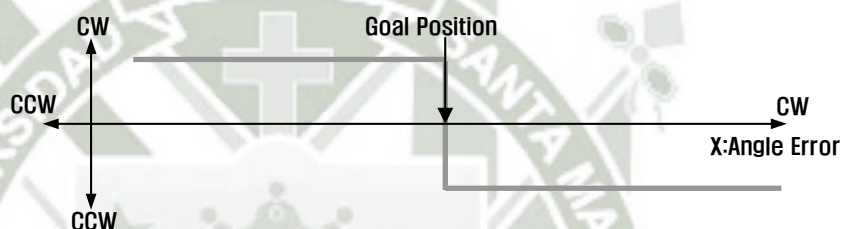
You can verify the Torque Enabled status by trying to move the output of the actuator by hand.

Example 17

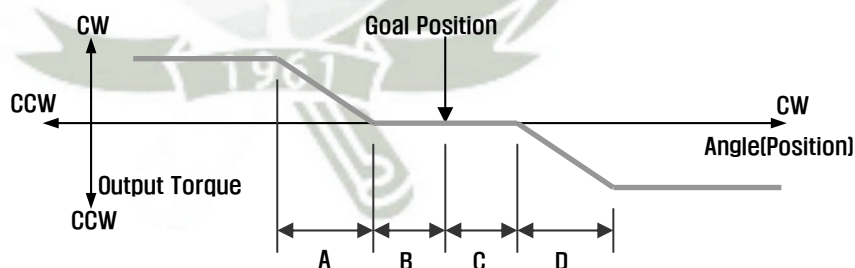
Setting the Compliance Margin to 1 and Compliance Slope to 0x40 for a Dynamixel actuator with an ID of 0

Compliance

The Angle Error and Torque Output can be represented with the following graph.



Even if the position deviates a little from the goal position in the CW direction, a large amount of torque is generated in the CCW direction to compensate for this. However, since inertia must be considered, a realistic implementation differs from this approach. Considering this, the given conditions can be represented by the following graph.



A : CCW Compliance Slope (Address0x1D) = 0x40 (about 18.8°)

B : CCW Compliance Margin (Address0x1B) = 0x01 (about 0.29°)

C : CW Compliance Margin (Address0x01A) = 0x01 (about 0.29°)

D : CW Compliance Slope (Address0x1C) = 0x40 (about 18.8°)

DYNAMIXEL

AX-12

ROBOTIS

Instruction Packet Instruction = WRITE_DATA, Address = 0x1A, DATA = 0x01, 0x01, 0x40, 0x40

Communication ->[Dynamixel]:FF FF 00 07 03 1A 01 01 40 40 59 (LEN:011)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR
The Compliance Slope takes effect with discrete steps of 2^n (n is integer). Thus any Compliance value between 0x11 and 0x20 has identical effects.

Example 18 Position the output of a Dynamixel actuator with an ID of 0 to 180° with an angular velocity of 057RPM

Set Address 0x1E (Goal Position) to 0x200 and Address 0x20 (Moving Speed) to 0x200.

Instruction Packet Instruction = WRITE_DATA, Address = 0x1E, DATA = 0x00, 0x02, 0x00, 0x02

Communication ->[Dynamixel]:FF FF 00 07 03 1E 00 02 00 02 D3 (LEN:011)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 19 Position the output of a Dynamixel actuator with an ID of 0 to 0° and Position the output of a Dynamixel actuator with an ID of 1 to 300°, and initiate the movement at the same time.

If the WRITE_DATA is used, the movement of the two actuators cannot be initiate at the same time, thus the REG_WRITE and ACTION instructions should be used instead.

Instruction Packet ID=0, Instruction = REG_WRITE, Address = 0x1E, DATA = 0x00, 0x00
ID=1, Instruction = REG_WRITE, Address = 0x1E, DATA = 0xff, 0x03
ID=0xfe(Broadcasting ID), Instruction = ACTION,

Communication ->[Dynamixel]:FF FF 00 05 04 1E 00 00 D8 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)
->[Dynamixel]:FF FF 01 05 04 1E FF 03 D5 (LEN:009)
<-[Dynamixel]:FF FF 01 02 00 FC (LEN:006)
->[Dynamixel]:FF FF FE 02 05 FA (LEN:006)
<-[Dynamixel]: //No return packet against broadcasting ID

Status Packet Result NO ERROR

DYNAMIXEL

AX-12

ROBOTIS

Example 20

Lock all addresses except for Address 0x18 ~ Address 0x23 for a Dynamixel actuator with an ID of 0

Set Address 0x2F (Lock) to 1.

Instruction Packet

Instruction = WRITE_DATA, Address = 0x2F, DATA = 0x01

Communication

->[Dynamixel]:FF FF 00 04 03 2F 01 C8 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result

NO ERROR

Once locked, the only way to unlock it is to remove the power.
If an attempt is made to access any locked data, an error is returned.

->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 08 F5 (LEN:006)

Range Error

Example 21

Set the minimum power (Punch) to 0x40 for a Dynamixel actuator with an ID of 0

Instruction Packet

Instruction = WRITE_DATA, Address = 0x30, DATA = 0x40, 0x00

Communication

->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

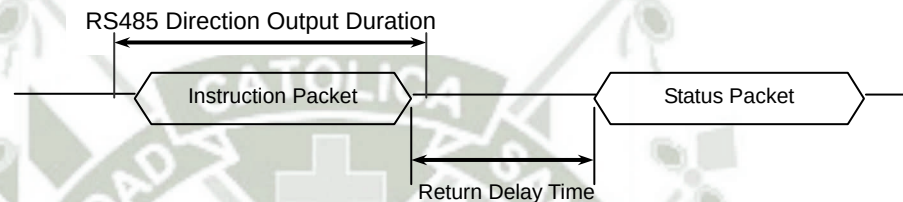
Status Packet Result

NO ERROR

Appendix

Half duplex UART

Half duplex UART is a serial communication protocol where both TxD and RxD cannot be used at the same time. This method is generally used when many devices need to be connected to a single bus. Since more than one device are connected to the same bus, all the other devices need to be in input mode while one device is transmitting. The Main Controller that controllers the Dynamixel actuators sets the communication direction to input mode, and only when it is transmitting an Instruction Packet, it changes the direction to output mode.



Return Delay Time

The time it takes for the Dynamixel actuator to return the Status Packet after receiving an Instruction Packet. The Default Value is 160 uSec and can be changed via the Control Table at Address 5. The Main Controller needs to change the Direction Port to input mode during the Return Delay Time after sending an instruction packet.

Tx,Rx Direction

For Half Duplex UART, the transmission ending timing is important to change the direction to receiving mode. The bit definitions within the register that indicates UART_STATUS are as the following

TXD_BUFFER_READY_BIT: Indicates that the transmission DATA can be loaded into the Buffer. Note that this only means that the SERIAL TX BUFFER is empty, and does not necessarily mean that the all the data transmitted before has left the CPU.

TXD_SHIFT_REGISTER_EMPTY_BIT: Set when all the Transmission Data has completed its transmission and left the CPU.

The TXD_BUFFER_READY_BIT is used when one byte is to be transmitted via the serial communication channel, and an example is shown below.

TxDByte(byte bData)

```
{
    while(!TXD_BUFFER_READY_BIT); //wait until data can be loaded.
    SerialTxDBuffer = bData;      //data load to TxD buffer
}
```


DYNAMIXEL

AX-12

ROBOTIS

When changing the direction, the TXD_SHIFT_REGISTER_EMPTY_BIT must be checked.

The following is an example program that sends an Instruction Packet.

```

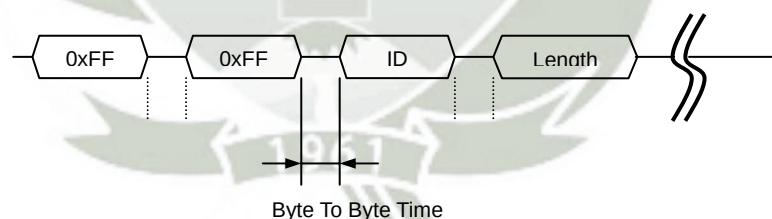
LINE 1      DIRECTION_PORT = TX_DIRECTION;
LINE 2      TxDByte(0xff);
LINE 3      TxDByte(0xff);
LINE 4      TxDByte(bID);
LINE 5      TxDByte(bLength);
LINE 6      TxDByte(bInstruction);
LINE 7      TxDByte(Parameter0); TxDByte(Parameter1); ...
LINE 8      DisableInterrupt(); // interrupt should be disable
LINE 9      TxDByte(Checksum); //last TxD
LINE 10     while(!TXD_SHIFT_REGISTER_EMPTY_BIT); //Wait till last data bit has been sent
LINE 11     DIRECTION_PORT = RX_DIRECTION; //Direction change to RXD
LINE 12     EnableInterrupt(); // enable interrupt again

```

Please note the important lines between LINE 8 and LINE 12. Line 8 is necessary since an interrupt here may cause a delay longer than the return delay time and corruption to the front of the status packet may occur.

Byte to Byte Time

The delay time between bytes when sending an instruction packet. If the delay time is over 100ms, then the Dynamixel actuator recognizes this as a communication problem and waits for the next header (0xff 0xff) of a packet again.



The following is the source code of a program (Example.c) that accesses the Dynamixel actuator using the Atmega 128.

DYNAMIXEL

AX-12

ROBOTIS

C Language Example : Dinamixel access with Atmega128

```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2005. 5. 11
 * Author : BS KIM
 */
```

```
/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
```

```
#define cbi(REG8, BITNUM) REG8 &= ~(1<BITNUM>)
#define sbi(REG8, BITNUM) REG8 |= 1<BITNUM>
```

```
typedef unsigned char byte;
typedef unsigned int word;
#define ON 1
#define OFF 0
#define _ON 0
#define _OFF 1
```

```
//--- Control Table Address ---
//EEPROM AREA
```

```
#define P_MODEL_NUMBER_L 0
#define P_MODEL_NUMBER_H 1
#define P_VERSION 2
#define P_ID 3
#define P_BAUD_RATE 4
#define P_RETURN_DELAY_TIME 5
#define P_CW_ANGLE_LIMIT_L 6
#define P_CW_ANGLE_LIMIT_H 7
#define P_CCW_ANGLE_LIMIT_L 8
#define P_CCW_ANGLE_LIMIT_H 9
#define P_SYSTEM_DATA2 10
#define P_LIMIT_TEMPERATURE 11
#define P_DOWN_LIMIT_VOLTAGE 12
#define P_UP_LIMIT_VOLTAGE 13
#define P_MAX_TORQUE_L 14
#define P_MAX_TORQUE_H 15
#define P_RETURN_LEVEL 16
#define P_ALARM_LED 17
#define P_ALARM_SHUTDOWN 18
#define P_OPERATING_MODE 19
#define P_DOWN_CALIBRATION_L 20
#define P_DOWN_CALIBRATION_H 21
#define P_UP_CALIBRATION_L 22
#define P_UP_CALIBRATION_H 23
```

```
#define P_TORQUE_ENABLE (24)
#define P_LED (25)
#define P_CW_COMPLIANCE_MARGIN (26)
#define P_CCW_COMPLIANCE_MARGIN (27)
#define P_CW_COMPLIANCE_SLOPE (28)
#define P_CCW_COMPLIANCE_SLOPE (29)
#define P_GOAL_POSITION_L (30)
#define P_GOAL_POSITION_H (31)
#define P_GOAL_SPEED_L (32)
#define P_GOAL_SPEED_H (33)
#define P_TORQUE_LIMIT_L (34)
#define P_TORQUE_LIMIT_H (35)
#define P_PRESENT_POSITION_L (36)
#define P_PRESENT_POSITION_H (37)
#define P_PRESENT_SPEED_L (38)
#define P_PRESENT_SPEED_H (39)
#define P_PRESENT_LOAD_L (40)
#define P_PRESENT_LOAD_H (41)
#define P_PRESENT_VOLTAGE (42)
#define P_PRESENT_TEMPERATURE (43)
```

```
#define P_REGISTERED_INSTRUCTION (44)
#define P_PAUSE_TIME (45)
#define P_MOVING (46)
#define P_LOCK (47)
#define P_PUNCH_L (48)
#define P_PUNCH_H (49)
```

```
//--- Instruction ---
```

```
#define INST_PING 0x01
#define INST_READ 0x02
#define INST_WRITE 0x03
#define INST_REG_WRITE 0x04
#define INST_ACTION 0x05
#define INST_RESET 0x06
#define INST_DIGITAL_RESET 0x07
#define INST_SYSTEM_READ 0x0C
#define INST_SYSTEM_WRITE 0x0D
#define INST_SYNC_WRITE 0x83
#define INST_SYNC_REG_WRITE 0x84
```

```
#define CLEAR_BUFFER gbRxBufferReadPointer = gbRxBufferWritePointer
#define DEFAULT_RETURN_PACKET_SIZE 6
#define BROADCASTING_ID 0xfe
```

```
#define TxD8 TxD81
#define RxD8 RxD81
```

```
//Hardware Dependent Item
#define DEFAULT_BAUD_RATE 34 //57600bps at 16MHz
```

```
////// For CM-5
#define RS485_TXD PORTE &= ~_BV(PE3), PORTE |= _BV(PE2)
//PORT_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~_BV(PE2), PORTE |= _BV(PE3)
//PORT_485_DIRECTION = 0
```

```
/*
////// For CM-2
#define RS485_TXD PORTE |= _BV(PE2); //PORT_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~_BV(PE2); //PORT_485_DIRECTION = 0
*/
//#define TXD0_FINISH UCSROA, 6 //This bit is for checking TxD Buffer
//in CPU is empty or not.
//#define TXD1_FINISH UCSRIA, 6
```

```
#define SET_TxD0_FINISH sbi(UCSROA, 6)
#define RESET_TxD0_FINISH cbi(UCSROA, 6)
#define CHECK_TxD0_FINISH bit_is_set(UCSROA, 6)
#define SET_TxD1_FINISH sbi(UCSRIA, 6)
#define RESET_TxD1_FINISH cbi(UCSRIA, 6)
#define CHECK_TxD1_FINISH bit_is_set(UCSRIA, 6)
```

```
#define RX_INTERRUPT 0x01
#define TX_INTERRUPT 0x02
#define OVERFLOW_INTERRUPT 0x01
#define SERIAL_PORT0 0
#define SERIAL_PORT1 1
#define BIT_RS485_DIRECTION0 0x08 //Port E
#define BIT_RS485_DIRECTION1 0x04 //Port E
```

```
#define BIT_ZIGBEE_RESET PD4 //out : default 1 //PORTD
#define BIT_ENABLE_RXD_LINK_PC PD5 //out : default 1
#define BIT_ENABLE_RXD_LINK_ZIGBEE PD6 //out : default 0
#define BIT_LINK_PLUGIN PD7 //in, no pull up
```

```
void TxD81(byte bTxDData);
void TxD80(byte bTxDData);
void TxDString(byte *bData);
void TxD8Hex(byte bSentData);
void TxD32Dec(long lLong);
byte RxD81(void);
void MilliSec(word wDelayTime);
void PortInitialize(void);
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt);
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength);
byte RxPacket(byte bRxLength);
void PrintBuffer(byte *bpPrintBuffer, byte bLength);
```

DYNAMIXEL

AX-12

ROBOTIS

```
// --- Global Variable Number ---
volatile byte gbpRxInterruptBuffer[256];
byte gbpParameter[128];
byte gbpRxBufferReadPointer;
byte gbpRxBuffer[128];
byte gbpTxBuffer[128];
volatile byte gbpRxBufferWritePointer;

int main(void)
{
    byte bCount, bID, bTxPacketLength, bRxPacketLength;

    PortInitialize(); //Port In/Out Direction Definition
    RS485_RXD; //Set RS485 Direction to Input State.
    SerialInitialize(SERIAL_PORT0, 1, RX_INTERRUPT); //RS485
        Initializing(RxInterrupt)
    SerialInitialize(SERIAL_PORT1, DEFAULT_BAUD_RATE, 0); //RS232
        Initializing(None Interrupt)

    gbpRxBufferReadPointer = gbpRxBufferWritePointer = 0; //RS485
        RxBuffer Clearing.

    sei(); //Enable Interrupt --- Compiler Function
    TxString("\r\n [The Example of Dynamixel Evaluation with
        ATmega128, GCC-AVR]");

    //Dynamixel Communication Function Execution Step.
    // Step 1. Parameter Setting (gbpParameter[]). In case of no parameter
    instruction (Ex. INST_PING), this step is not
    needed.
    // Step 2. TxPacket (ID, INSTRUCTION, LengthOfParameter); ---Total
    TxPacket Length is returned
    // Step 3. RxPacket (ExpectedReturnPacketLength); --- Real RxPacket
    Length is returned
    // Step 4 PrintBuffer (BufferStartPointer, LengthForPrinting);

    bID = 1;
    TxString("\r\n\r\n Example 1. Scanning Dynamixels (0~9). --- Any Key to
        Continue."); RxD8();
    for (bCount = 0; bCount < 0x0A; bCount++)
    {
        bTxPacketLength = TxPacket(bCount, INST_PING, 0);
        bRxPacketLength = RxPacket(255);
        TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
        TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);
        if (bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE)
        {
            TxString("\r\n Found!! ID:"); TxD8Hex(bCount);
            bID = bCount;
        }
    }

    TxString("\r\n\r\n Example 2. Read Firmware Version. --- Any Key to
        Continue."); RxD8();
    gbpParameter[0] = P_VERSION; //Address of Firmware Version
    gbpParameter[1] = 1; //Read Length
    bTxPacketLength = TxPacket(bID, INST_READ, 2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParameter
        [1]);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);
    if (bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
    {
        TxString("\r\n Return Error :"); TxD8Hex(gbpRxBuffer[4]);
        TxString("\r\n Firmware Version :"); TxD8Hex(gbpRxBuffer[5]);
    }

    TxString("\r\n\r\n Example 3. LED ON --- Any Key to Continue.");
        RxD8();
    gbpParameter[0] = P_LED; //Address of LED
    gbpParameter[1] = 1; //Writing Data
    bTxPacketLength = TxPacket(bID, INST_WRITE, 2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);
```

```
TxDString("\r\n\r\n Example 4. LED OFF --- Any Key to Continue.");
        RxD8();
    gbpParameter[0] = P_LED; //Address of LED
    gbpParameter[1] = 0; //Writing Data
    bTxPacketLength = TxPacket(bID, INST_WRITE, 2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

    TxString("\r\n\r\n Example 5. Read Control Table. --- Any Key to
        Continue."); RxD8();
    gbpParameter[0] = 0; //Reading Address
    gbpParameter[1] = 49; //Read Length
    bTxPacketLength = TxPacket(bID, INST_READ, 2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParameter
        [1]);

    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);
    if (bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
    {
        TxString("\r\n\r\n");
        for (bCount = 0; bCount < 49; bCount++)
        {
            TxD8(' '); TxD8Hex(bCount); TxString("\r\n");
            TxD8Hex(gbpRxBuffer[bCount+5]); TxD8(' ');
        }
    }

    TxString("\r\n\r\n Example 6. Go 0x200 with Speed 0x100 --- Any Key to
        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x02; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x00; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x01; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID, INST_WRITE, 5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

    TxString("\r\n\r\n Example 7. Go 0x00 with Speed 0x40 --- Any Key to
        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x00; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x40; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x00; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID, INST_WRITE, 5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

    TxString("\r\n\r\n Example 8. Go 0x3ff with Speed 0x3ff --- Any Key to
        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0xff; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x03; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0xff; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x03; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID, INST_WRITE, 5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

    TxString("\r\n\r\n Example 9. Torque Off --- Any Key to Continue.");
        RxD8();
    gbpParameter[0] = P_TORQUE_ENABLE; //Address of LED
    gbpParameter[1] = 0; //Writing Data
    bTxPacketLength = TxPacket(bID, INST_WRITE, 2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxString("\r\n TxID:"); PrintBuffer(gbpTxBuffer, bTxPacketLength);
    TxString("\r\n RxID:"); PrintBuffer(gbpRxBuffer, bRxPacketLength);

    TxString("\r\n\r\n End. Push reset button for repeat");
```


DYNAMIXEL

AX-12

ROBOTIS

```

while(1);
}

void PortInitialize(void)
{
    DDRA = DDRB = DDRC = DDRD = DDRE = DDRF = 0; //Set all port to
    input direction first.
    PORTB = PORTC = PORTD = PORTE = PORTF = PORTG = 0x00; //PortData
    initialize to 0
    cbi(SFIOA, 2); //All Port Pull Up ready
    DDRE |= (BIT_RS485_DIRECTION0|BIT_RS485_DIRECTION1); //set output
    the bit RS485direction

    DDRD
    |=
    (BIT_ZIGBEE_RESET|BIT_ENABLE_RXD_LINK_PC|BIT_ENA
    BLE_RXD_LINK_ZIGBEE);

    PORTD &= ~_BV(BIT_LINK_PLUGIN); // no pull up
    PORTD |= _BV(BIT_ZIGBEE_RESET);
    PORTD |= _BV(BIT_ENABLE_RXD_LINK_PC);
    PORTD |= _BV(BIT_ENABLE_RXD_LINK_ZIGBEE);
}

/*
TxPacket() send data to RS485.
TxPacket() needs 3 parameter: ID of Dynamixel, Instruction byte,
Length of parameters.
TxPacket() return length of Return packet from Dynamixel.
*/
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength)
{
    byte bCount, bChecksum, bPacketLength;

    gbpTxBuffer[0] = 0xff;
    gbpTxBuffer[1] = 0xff;
    gbpTxBuffer[2] = bID;
    gbpTxBuffer[3] =
    = bParameterLength+2;
    //Length(Paramter, Instruction, Checksum)
    gbpTxBuffer[4] = bInstruction;
    for(bCount = 0; bCount < bParameterLength; bCount++)
    {
        gbpTxBuffer[bCount+5] = gbpParameter[bCount];
    }
    bChecksum = 0;
    bPacketLength = bParameterLength+4+2;
    for(bCount = 2; bCount < bPacketLength-1; bCount++) //except
    0xff, checksum
    {
        bChecksum += gbpTxBuffer[bCount];
    }
    gbpTxBuffer[bCount] = ~bChecksum; //Writing Checksum with Bit
    Inversion

    RS485_TXD:
    for(bCount = 0; bCount < bPacketLength; bCount++)
    {
        sbi(UCSR0A, 6); //SET_TXD0_FINISH:
        TxD80(gbpTxBuffer[bCount]);
    }
    while(!CHECK_TXD0_FINISH); //Wait until TXD Shift register empty
    RS485_RXD:
    return(bPacketLength);
}

/*
RxPacket() read data from buffer.
RxPacket() need a Parameter: Total length of Return Packet.
RxPacket() return Length of Return Packet.
*/

byte RxPacket(byte bRxPacketLength)
{
    #define RX_TIMEOUT_COUNT2 3000L
    #define RX_TIMEOUT_COUNT1 (RX_TIMEOUT_COUNT2*10L)
    unsigned long ulCounter;
    byte bCount, bLength, bChecksum;
    byte bTimeout;

```

```

bTimeout = 0;
for(bCount = 0; bCount < bRxPacketLength; bCount++)
{
    ulCounter = 0;
    while(gbpRxBufferReadPointer == gbpRxBufferWritePointer)
    {
        if(ulCounter++ > RX_TIMEOUT_COUNT1)
        {
            bTimeout = 1;
            break;
        }
    }
    if(bTimeout) break;
    gbpRxBuffer[bCount]
    =
    gbpRxInterruptBuffer[gbpRxBufferReadPointer++];
}
bLength = bCount;
bChecksum = 0;

if(gbpTxBuffer[2] != BROADCASTING_ID)
{
    if(bTimeout && bRxPacketLength != 255)
    {
        TxDString("\r\n [Error:Rx Timeout]");
        CLEAR_BUFFER;
    }
}

if(bLength > 3) //checking is available.
{
    if(gbpRxBuffer[0] != 0xff || gbpRxBuffer[1] != 0xff )
    {
        TxDString("\r\n [Error:Wrong Header]");
        CLEAR_BUFFER;
        return 0;
    }
    if(gbpRxBuffer[2] != gbpTxBuffer[2] )
    {
        TxDString("\r\n [Error:TxID != RxID]");
        CLEAR_BUFFER;
        return 0;
    }
    if(gbpRxBuffer[3] != bLength-4)
    {
        TxDString("\r\n [Error:Wrong Length]");
        CLEAR_BUFFER;
        return 0;
    }
    for(bCount = 2; bCount < bLength; bCount++) bChecksum +=
    gbpRxBuffer[bCount];
    if(bChecksum != 0xff)
    {
        TxDString("\r\n [Error:Wrong CheckSum]");
        CLEAR_BUFFER;
        return 0;
    }
}
return bLength;
}

/*
PrintBuffer() print data in Hex code.
PrintBuffer() needs two parameter: name of Pointer(gbpTxBuffer,
gbpRxBuffer)
*/
void PrintBuffer(byte *bpPrintBuffer, byte bLength)
{
    byte bCount;
    for(bCount = 0; bCount < bLength; bCount++)
    {
        TxD8Hex(bpPrintBuffer[bCount]);
        TxD8(' ');
    }
    TxDString("\r\n (LEN:"); TxD8Hex(bLength); TxD8(' ');
}

```

DYNAMIXEL

AX-12

ROBOTIS

```

/*
Print value of Baud Rate.
*/
void PrintBaudrate(void)
{
    TxString("RS232:");TxD32Dec((1600000L/8L)/((long)UBRR1L+1
    L)); TxString(" BPS,");
    TxString(" RS485:");TxD32Dec((1600000L/8L)/((long)UBRR0L+1L) );
    TxString(" BPS");
}

/*Hardware Dependent Item*/
#define TXD1_READY bit_is_set(UCSR1A, 5)
// (UCSR1A_Bit5)
#define TXD1_DATA (UDR1)
#define RXD1_READY bit_is_set(UCSR1A, 7)
#define RXD1_DATA (UDR1)

#define TXD0_READY bit_is_set(UCSR0A, 5)
#define TXD0_DATA (UDR0)
#define RXD0_READY bit_is_set(UCSR0A, 7)
#define RXD0_DATA (UDR0)

/*
SerialInitialize() set Serial Port to initial state.
Vide Mega128 Data sheet about Setting bit of register.
SerialInitialize() needs port, Baud rate, Interrupt value.
*/
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt)
{
    if(bPort == SERIAL_PORT0)
    {
        UBRR0H = 0; UBRR0L = bBaudrate;
        UCSR0A = 0x02; UCSR0B = 0x18;
        if(bInterrupt&RX_INTERRUPT) sbi(UCSR0B, 7); // RxD interrupt enable
        UCSR0C = 0x06; UDR0 = 0xFF;
        sbi(UCSR0A, 6); //SET_TXD0_FINISH; // Note. set 1, then 0 is read
    }
    else if(bPort == SERIAL_PORT1)
    {
        UBRR1H = 0; UBRR1L = bBaudrate;
        UCSR1A = 0x02; UCSR1B = 0x18;
        if(bInterrupt&RX_INTERRUPT) sbi(UCSR1B, 7); // RxD interrupt enable
        UCSR1C = 0x06; UDR1 = 0xFF;
        sbi(UCSR1A, 6); //SET_TXD1_FINISH; // Note. set 1, then 0 is read
    }
}

/*
TxD8Hex() print data seperatly.
ex> 0x1a -> '1' 'a'.
*/
void TxD8Hex(byte bSentData)
{
    byte bTmp;

    bTmp = (byte) (bSentData>>4) & 0x0f + (byte)'0';
    if(bTmp > '9') bTmp += 7;
    TxD8(bTmp);
    bTmp = (byte) (bSentData & 0x0f) + (byte)'0';
    if(bTmp > '9') bTmp += 7;
    TxD8(bTmp);
}

/*
TxD80() send data to USART 0.
*/
void TxD80(byte bTxdData)
{
    while(!TXD0_READY);
    TXD0_DATA = bTxdData;
}

```

```

}

/*
TxD81() send data to USART 1.
*/
void TxD81(byte bTxdData)
{
    while(!TXD1_READY);
    TXD1_DATA = bTxdData;
}

/*
TxD32Dex() change data to decimal number system
*/
void TxD32Dec(long lLong)
{
    byte bCount, bPrinted;
    long lTmp, lDigit;
    bPrinted = 0;
    if(lLong < 0)
    {
        lLong = -lLong;
        Tx8('-');
    }
    lDigit = 1000000000L;
    for(bCount = 0; bCount < 9; bCount++)
    {
        lTmp = (byte) (lLong/lDigit);
        if(lTmp)
        {
            Tx8(((byte)lTmp)+'0');
            bPrinted = 1;
        }
        else if(bPrinted) Tx8(((byte)lTmp)+'0');
        lLong -= ((long)lTmp)*lDigit;
        lDigit = lDigit/10;
    }
    lTmp = (byte) (lLong/lDigit);
    /*if(lTmp)*/ Tx8(((byte)lTmp)+'0');
}

/*
TxDString() prints data in ACSII code.
*/
void TxDString(byte *bData)
{
    while(*bData)
    {
        Tx8(*bData++);
    }
}

/*
Rx81() read data from UART1.
Rx81() return Read data.
*/
byte Rx81(void)
{
    while(!RXD1_READY);
    return(RXD1_DATA);
}

/*
SIGNAL() UART0 Rx Interrupt - write data to buffer
*/
SIGNAL (SIG_UART0_RECV)
{
    gbpRxInterruptBuffer[(gbRxBufferWritePointer++)] = RXD0_DATA;
}

```

DYNAMIXEL

AX-12

ROBOTIS

Connector

Company Name : Molex

Pin Number: 4

Model Number

	Molex Part Number	Old Part Number
Male	22-03-5045	5267-03
Female	50-37-5043	5264-03

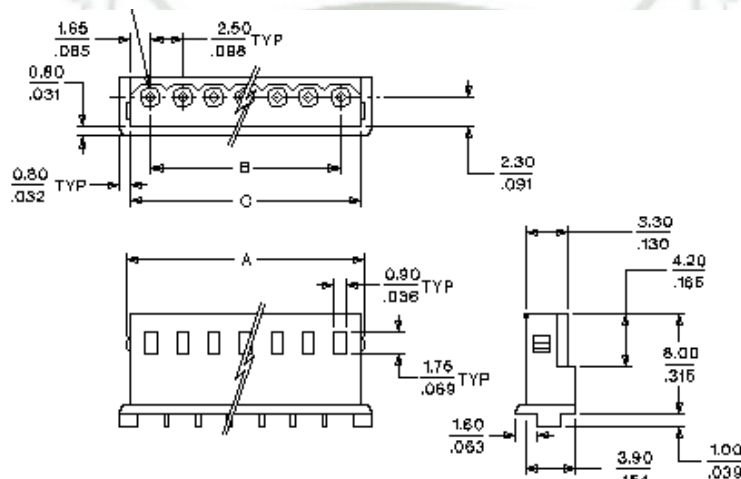
Temperature range : -40°C to +105°C

Contact Insertion Force-max : 14.7N (3.30 lb)

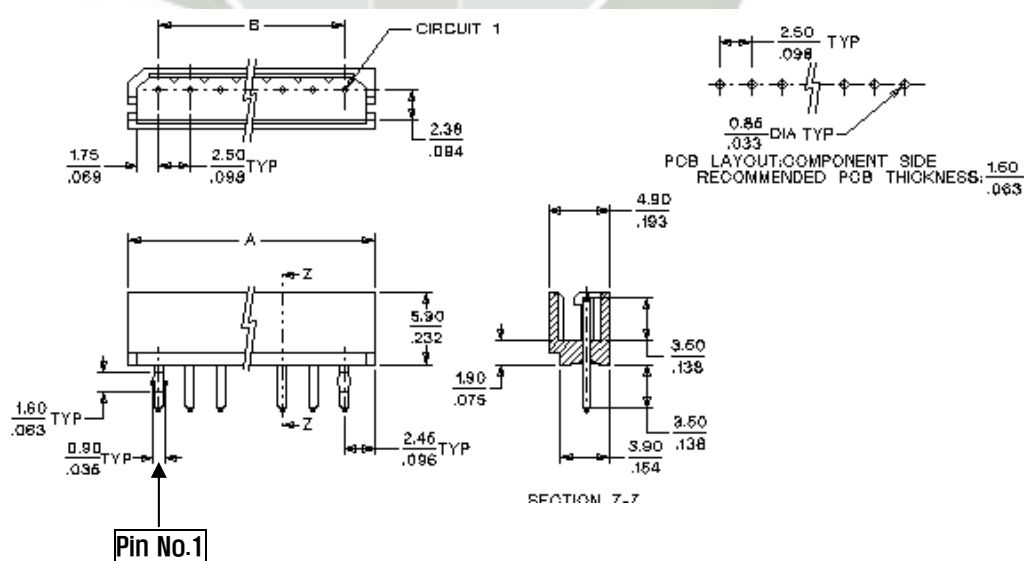
Contact Retention Force-min : 14.7N (3.30 lb)

www.molex.com or www.molex.co.jp for more detail information

Female Connector



Male Connector

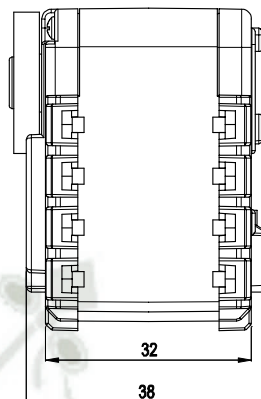
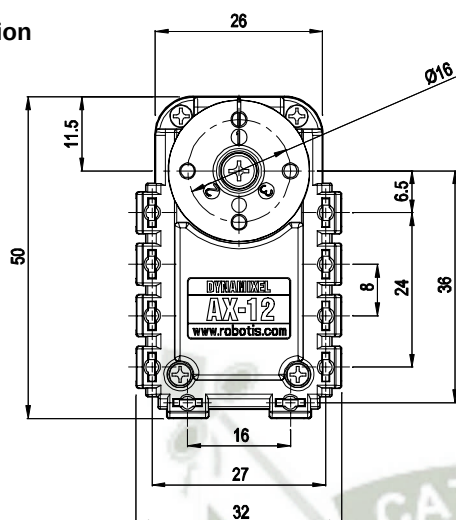


DYNAMIXEL

AX-12

ROBOTIS

Dimension



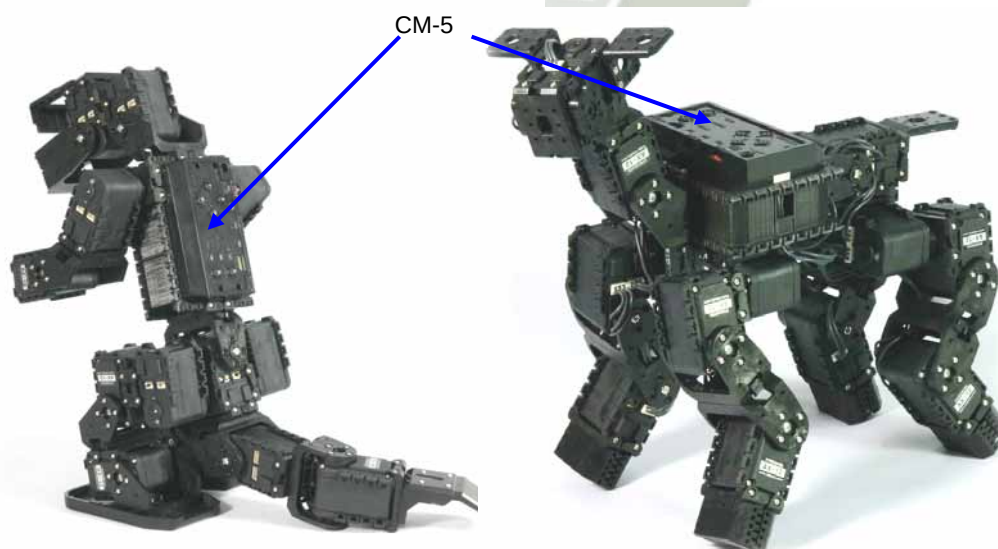
CM-5

Dedicated AX-12 control box. Able to control 30 AX-12 actuators.

6 push buttons (5 for selection, 1 for reset)

Optional installable wireless devices available

Battery compartment (AA x 8) with recharging capability (when connected to an external SMPS)



[Show](#)

ROBOTIS e-Manual v1.29.00

RX-24F

Part Photo



[RX-24F]

H/W Specification

- Weight : 67g
- Dimension : 35.6mm x 50.6mm x 35.5mm
- Resolution : 0.29°
- Gear Reduction Ratio : 193 : 1
- Stall Torque : 2.6N.m (at 12V, 2.4A)
- No load speed : 126rpm (at 12.V)
- Running Degree
 - 0° ~ 300°
 - Endless Turn
- Running Temperature : -5℃ ~ +80℃
- Voltage : 9V~12V (Recommended Voltage 11.1V)
- Command Signal : Digital Packet
- Protocol Type : RS485 Asynchronous Serial Communication (8bit,1stop, No Parity)
- Link (Physical) : RS485 Multi Drop Bus
- ID : 254 ID (0~253)
- Communication Speed : 7843bps ~ 1 Mbps
- Feedback : Position, Temperature, Load, Input Voltage, etc.
- Material : Full Metal Gear, Engineering Plastic Body
- Standby current : 50 mA

Stall torque is the maximum instantaneous and static torque
Stable motions are possible with robots designed for loads with 1/5 or less of the stall torque

Control Table

Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet.

EEPROM and RAM

Data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.

Address

It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.

Access

Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.

Initial Value

In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.

Highest/Lowest Byte

In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16bit, but it is divided into 8bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	24 (0X18)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	–
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	190 (0XBE)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
RAM	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	–
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	–
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	–
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	–
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	–
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	–
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	–
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	–
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	–
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	–
	42 (0X2A)	Present Voltage	Current Voltage	R	–
	43 (0X2B)	Present Temperature	Current Temperature	R	–
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Address Function Help

EEPROM Area

Model Number

It represents the Model Number.

Firmware Version

It represents the firmware version.

ID

It is a unique number to identify Dynamixel.

The range from 0 to 252 (0xFC) can be used, and, especially, 254(0xFE) is used as the Broadcast ID.

If the Broadcast ID is used to transmit Instruction Packet, we can command to all Dynamixels.

Please be cautious not to have the same IDs for the connected dynamixels. You may face communication issues or may not be able to search when IDs overlap.

Baud Rate

It represents the communication speed. 0 to 254 (0xFE) can be used for it.

This speed is calculated by using the below formula.

$$\text{Speed(BPS)} = 2000000 / (\text{Data} + 1)$$

Data	Set BPS	Target BPS	Tolerance
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

Note : Maximum Baud Rate error of 3% is within the tolerance of UART communication.

Return Delay Time

It is the delay time per data value that takes from the transmission of Instruction Packet until the return of Status Packet.

0 to 254 (0xFE) can be used, and the delay time per data value is 2 usec.

That is to say, if the data value is 10, 20 usec is delayed. The initial value is 250 (0xFA) (i.e., 0.5 msec).

CW/CCW Angle Limit

The angle limit allows the motion to be restrained.

The range and the unit of the value is the same as Goal Position(Address 30, 31).

- CW Angle Limit: the minimum value of Goal Position(Address 30, 31)
- CCW Angle Limit: the maximum value of Goal Position(Address 30, 31)

The following two modes can be set pursuant to the value of CW and CCW.

Operation Type	CW / CCW
Wheel Mode	both are 0
Joint Mode	neither at 0

The wheel mode can be used to wheel-type operation robots since motors of the robots spin infinitely.

The joint mode can be used to multi-joints robot since the robots can be controlled with specific angles.

The Highest Limit Temperature

Caution : Do not set the temperature lower/higher than the default value.

When the temperature alarm shutdown occurs, wait 20 minutes to cool the temperature before re-use.

Using the product when the temperature is high may and can cause damage.

The Lowest (Highest) Limit Voltage

It is the operation range of voltage.

50 to 250 (0x32 ~ 0x96) can be used. The unit is 0.1V.

For example, if the value is 80, it is 8V.

If Present Voltage (Address42) is out of the range, Voltage Range Error Bit (Bit0) of Status Packet is returned as '1' and Alarm is triggered as set in the addresses 17 and 18.

Max Torque

It is the torque value of maximum output. 0 to 1023 (0x3FF) can be used, and the unit is about 0.1%.

For example, Data 1023 (0x3FF) means that Dynamixel will use 100% of the maximum torque it can produce while Data 512 (0x200) means that Dynamixel will use 50% of the maximum torque. When the power is turned on, Torque Limit (Addresses 34 and 35) uses the value as the initial value.

Status Return Level

It decides how to return Status Packet. There are three ways like the below table.

Value	Return of Status Packet
0	No return against all commands (Except PING Command)
1	Return only for the READ command
2	Return for all commands

When Instruction Packet is Broadcast ID, Status Packet is not returned regardless of Status Return Level.

Alarm LED

Alarm Shutdown

Dynamixel can protect itself by detecting errors occur during the operation.
The errors can be set are as the table below.

Bit	Name	Contents
Bit 7	0	–
Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	Checksum Error	When the Checksum of the transmitted Instruction Packet is invalid
Bit 3	Range Error	When the command is given beyond the range of usage
Bit 2	OverHeating Error	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Error	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

It is possible to make duplicate set since the function of each bit is run by the logic of 'OR'. That is, if 0X05 (binary 00000101) is set, both Input Voltage Error and Overheating Error can be detected.
If errors occur, in case of Alarm LED, the LED blinks; in case of Alarm Shutdown, the motor output becomes 0 % by making the value of Torque Limit(Address 34, 35) as 0.

RAM Area

Torque Enable

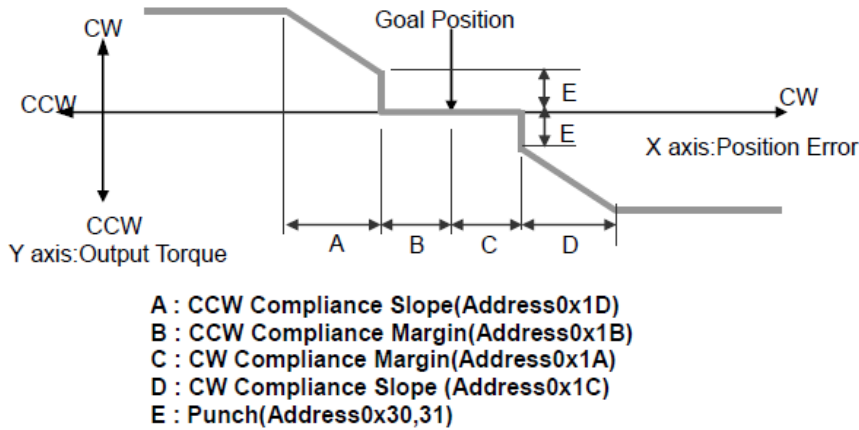
Value	Meaning
0	Keeps Torque from generating by interrupting the power of motor.
1	Generates Torque by impressing the power to the motor.

LED

Bit	Meaning
0	Turn OFF the LED
1	Turn ON the LED

Compliance

Compliance is to set the control flexibility of the motor.
The following diagram shows the relationship between output torque and position of the motor.



Compliance Margin

It exists in each direction of CW/CCW and means the error between goal position and present position.
The range of the value is 0~255, and the unit is the same as Goal Position.(Address 30,31)
The greater the value, the more difference occurs.

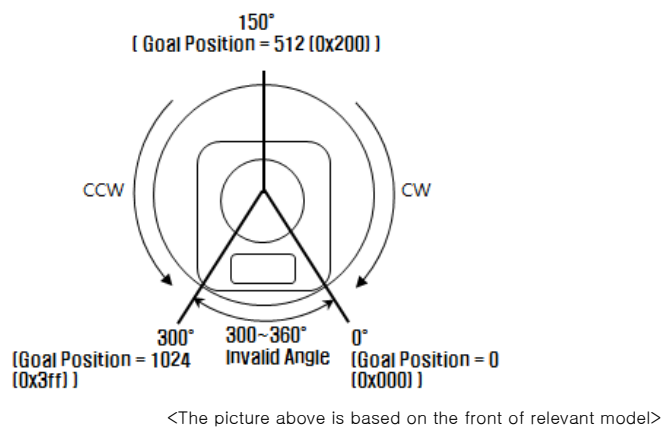
Compliance Slope

It exists in each direction of CW/CCW and sets the level of Torque near the goal position.
Compliance Slope is set in 7 steps, the higher the value, the more flexibility is obtained.
Data representative value is actually used value. That is, even if the value is set to 25, 16 is used internally as the representative value.

Step	Data Value	Data Representative Value
1	0 (0x00) ~ 3(0x03)	2 (0x02)
2	4(0x04) ~ 7(0x07)	4 (0x04)
3	8(0x08)~15(0x0F)	8 (0x08)
4	16(0x10)~31(0x1F)	16 (0x10)
5	32(0x20)~63(0x3F)	32 (0x20)
6	64(0x40)~127(0x7F)	64 (0x40)
7	128(0x80)~254(0xFE)	128 (0x80)

Goal Position

It is a position value of destination.
0 to 1023 (0x3FF) is available. The unit is 0.29 degree.
If Goal Position is out of the range, Angle Limit Error Bit (Bit1) of Status Packet is returned as '1' and Alarm is triggered as set in Alarm LED/Shutdown.



If it is set to Wheel Mode, this value is not used.

Moving Speed

It is a moving speed to Goal Position.
The range and the unit of the value may vary depending on the operation mode.

- Join Mode
0~1023 (0X3FF) can be used, and the unit is about 0.111rpm.
If it is set to 0, it means the maximum rpm of the motor is used without controlling the speed.
If it is 1023, it is about 114rpm.
For example, if it is set to 300, it is about 33.3 rpm.

Notes: Please check the maximum rpm of relevant model in Joint Mode. Even if the motor is set to more than maximum rpm, it cannot generate the torque more than the maximum rpm.

- Wheel Mode
0~2047(0X7FF) can be used, the unit is about 0.1%.
If a value in the range of 0~1023 is used, it is stopped by setting to 0 while rotating to CCW direction.
If a value in the range of 1024~2047 is used, it is stopped by setting to 1024 while rotating to CW direction.
That is, the 10th bit becomes the direction bit to control the direction.
In Wheel Mode, only the output control is possible, not speed.
For example, if it is set to 512, it means the output is controlled by 50% of the maximum output.

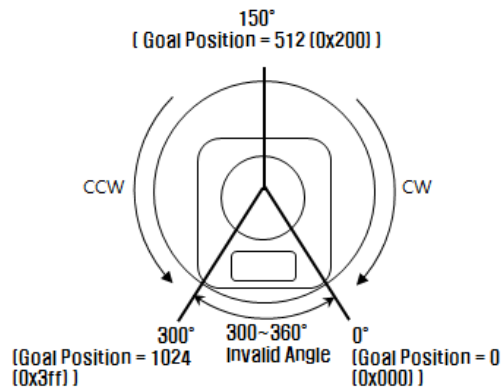
Torque Limit

It is the value of the maximum torque limit.
0 to 1023 (0x3FF) is available, and the unit is about 0.1%.
For example, if the value is 512, it is about 50%; that means only 50% of the maximum torque will be used.
If the power is turned on, the value of Max Torque (Address 14, 15) is used as the initial value.

Notes: If the function of Alarm Shutdown is triggered, the motor loses its torque because the value becomes 0. At this moment, if the value is changed to the value other than 0, the motor can be used again.

Present Position

It is the current position value of Dynamixel.
The range of the value is 0~1023 (0x3FF), and the unit is 0.29 degree.



<The picture above is based on the front of relevant model>

Caution: If it is set to Wheel Mode, the value cannot be used to measure the moving distance and the rotation frequency.

Present Speed

It is the current moving speed.
0~2047 (0X7FF) can be used.
If a value is in the rage of 0~1023, it means that the motor rotates to the CCW direction.
If a value is in the rage of 1024~2047, it means that the motor rotates to the CW direction.
That is, the 10th bit becomes the direction bit to control the direction, and 0 and 1024 are equal.
The unit of this value varies depending on operation mode.

- Joint Mode
The unit is about 0.111rpm.
For example, if it is set to 300, it means that the motor is moving to the CCW direction at a rate of about 33.3rpm.
- Wheel Mode
The unit is about 0.1%.
For example, if it is set to 512, it means that the torque is controlled by 50% of the maximum torque to the CCW direction.

Present Load

It means currently applied load.
The range of the value is 0~2047, and the unit is about 0.1%.
If the value is 0~1023, it means the load works to the CCW direction.
If the value is 1024~2047, it means the load works to the CW direction.
That is, the 10th bit becomes the direction bit to control the direction, and 1024 is equal to 0.
For example, the value is 512, it means the load is detected in the direction of CCW about 50% of the maximum torque.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Notes: Present load is an inferred value based on the internal output value; not a measured value using torque sensor, etc. Therefore, it may be inaccurate for measuring weight or torque. It is recommended to use it for predicting the direction and size of the force being applied to the joint.

Present Voltage

It is the size of the current voltage supplied.
This value is 10 times larger than the actual voltage. For example, when 10V is supplied, the data value is 100 (0x64)

Present Temperature

It is the internal temperature of Dynamixel in Celsius.
Data value is identical to the actual temperature in Celsius. For example, if the data value is 85 (0x55), the current internal temperature is 85℃.

Registered Instruction

Value	Meaning
0	There are no commands transmitted by REG_WRITE
1	There are commands transmitted by REG_WRITE.

Notes: If ACTION command is executed, the value is changed into 0.

Moving

Value	Meaning
0	Goal position command execution is completed.
1	Goal position command execution is in progress.

Lock

Value	Meaning
0	EEPROM area can be modified.
1	EEPROM area cannot be modified.

Caution: If Lock is set to 1, the power must be turned off and then turned on again to change into 0.

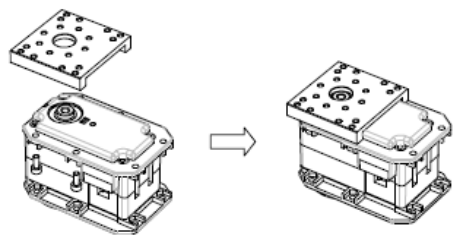
Punch

Current to drive motor is at minimum.
Can choose vales from 0x20 to 0x3FF.

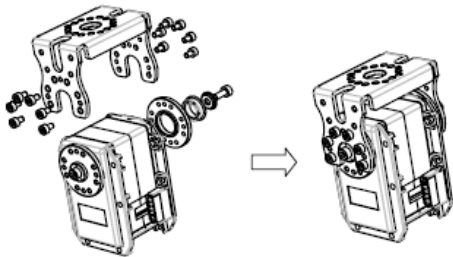
Option Frame (Old Model)

The types of RX-24F option frame are as follows.

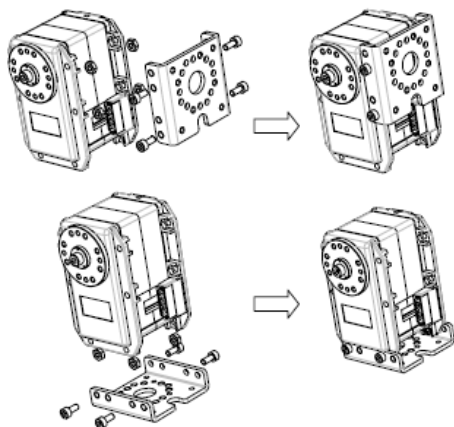
FR07-B1(OF-RX28B)



FR07-H1(OF-RX28H)



FR07-S1(OF-RX28S)



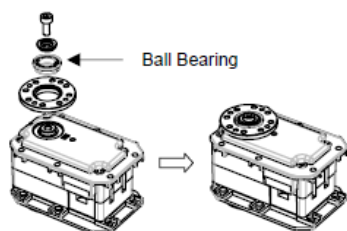
Horn (Old Model)

The types of RX-24F Horn are as follows.

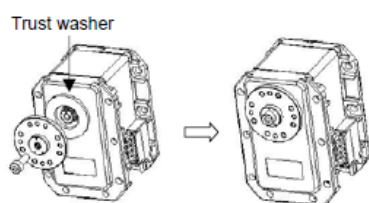
HN07-N1 (Horn RX-28)



HN07-I1 (Bearing Set RX-28H)

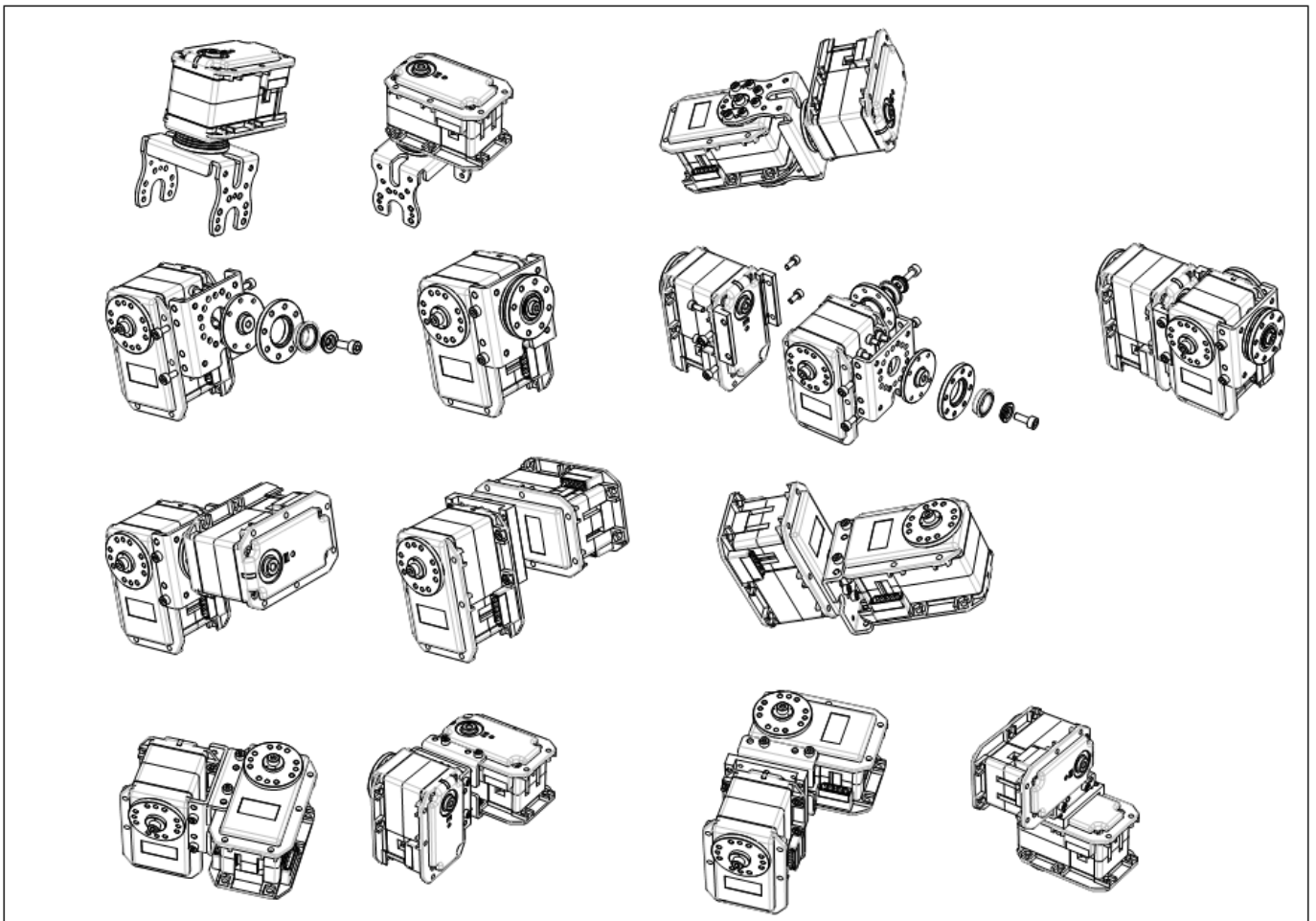


HN07-T1 (Horn 28T Set)



Combination

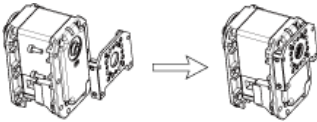
The following example shows the combination structure of option frames and horns.



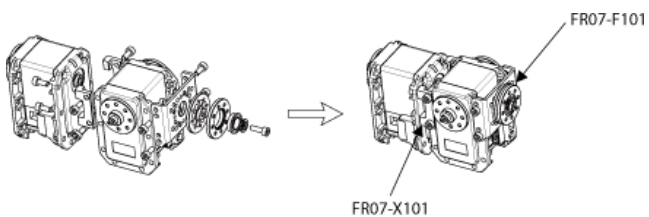
Option Frame (New Model)

The types of RX-24F option frames are as follows.

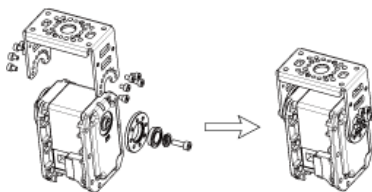
FR07-B101



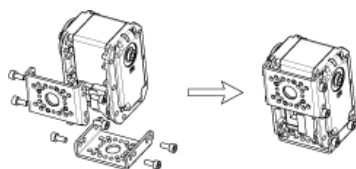
FR07-F101_FR07-X101



FR07-H101



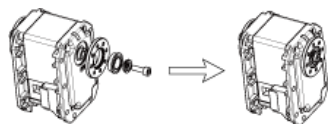
FR07-S101



Horn (New Model)

The types of RX-24F Horn are as follows.

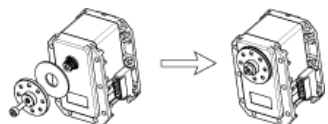
HN07-I101



HN07-N101



HN07-T101



Horn & Bearing Replacement

The horn is installed on the front wheel gear serration of the DYNAMIXEL whereas the bearing set is installed on the back.



Installing the Front Horn

Place the thrust horn washer into the actuator before inserting the horn
You must carefully align the horn to the wheel gear serration by aligning dots.



Once alignment is properly done, gently push the center of the horn toward the actuator.
Make sure that the horn washer is in place as you tighten the bolt.

Installing the Bearing Set

You may need to remove the bearing set from the previous actuator and reinstall it into the new actuator.

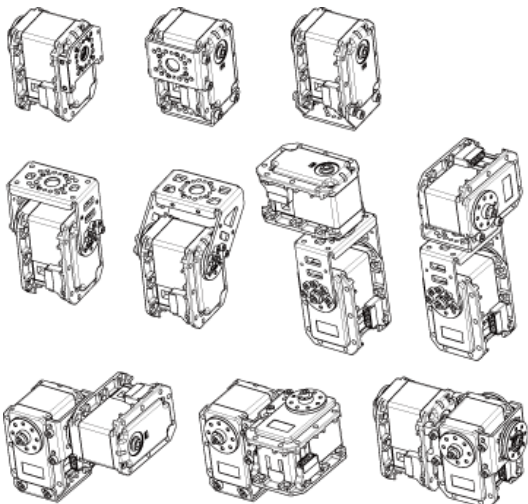
The bearing set can also be purchased separately.

As bearing set is rotating freely, therefore alignment is not required when assembling to DYNAMIXEL.



Combination (New Model)

The following example shows the combination structure of option frames and horns



Dimension

Drawing Information : [DOWNLOAD](#) RX24FDimension.pdf

Frame Compatibility Guide([LINK](#))

[Error Report](#)

Copyrights (c) 2010 [ROBOTIS](#) All rights reserved.

MX-28T / MX-28R

Parts Photo



※ Control Table's Compliance replaced by PID.

※ The control table's order for PID has changed to DIP from this version onwards. Please make reference of this change.

※ Although the MX-28T (TTL) and MX-28R (RS-485) differ in communications protocols both have the same features and perform equally. (TTL uses 3-pin connectors while RS-485 uses 4)

H/W Specification

- MCU : ST CORTEX-M3 (STM32F103C8 @ 72MHZ,32BIT)
- POSITION SENSOR : Contactless absolute encoder (12BIT,360 DEGREE)
- MOTOR : Maxon
- BAUD RATE : 8000 bps ~ 4.5 Mbps
- CONTROL ALGORITHM : PID CONTROL
- Resolution : 0.088°
- Running Degree
 - 0° ~ 360°
 - Endless Turn
- Weight : 72g
- Dimension : 35.6mm x 50.6mm x 35.5mm
- Gear Reduction Ratio : 193 : 1
- Stall Torque
 - 2.3N.m (at 11.1V, 1.3A),
 - 2.5N.m (at 12V, 1.4A)
 - 3.1N.m (at 14.8V, 1.7A)
- No load speed
 - 50rpm (at 11.1V)
 - 55rpm (at 12V)
 - 67rpm (at 14.8V)
- Running Temperature : -5℃ ~ +80℃
- **Voltage : 10 ~ 14.8V (Recommended Voltage 12V)**
- Command Signal : Digital Packet

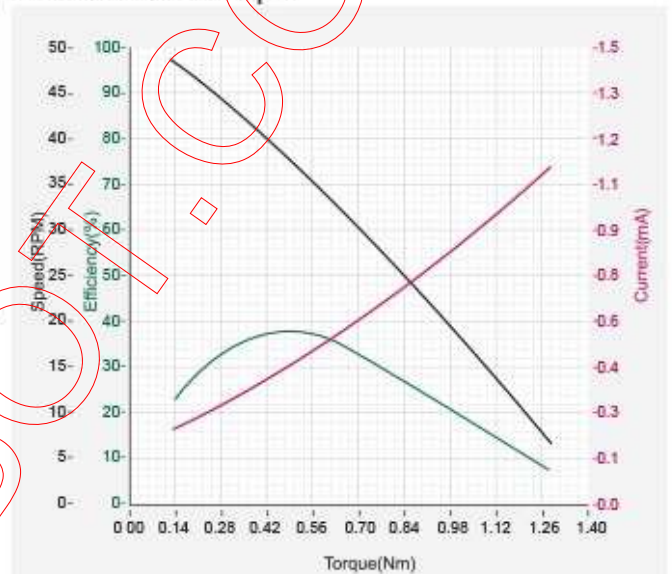
- Protocol Type
 - MX-28T (Half duplex Asynchronous Serial Communication (8bit,1stop, No Parity))
 - MX-28R (RS485 Asynchronous Serial Communication (8bit,1stop, No Parity))
- Link (Physical)
 - MX-28T (TTL Level Multi Drop Bus)
 - MX-28R (RS485 Multi Drop Bus)
- ID : 254 ID (0~253)
- Feedback : Position, Temperature, Load, Input Voltage, etc.
- Material : Full Metal Gear, Engineering Plastic Body
- Standby current : 100 mA

MX-28T/R

Data

	Unit	Data
Rated voltage	V	12
No load speed	RPM	55
No load current	mA	120
Resolution	Step/turn	4096
Gear ratio	-	193
Interface	-	RS-485 / TTL

Performance Graph



Precautions when connecting to power supply!

- For the stable power supply, we recommend using ROBOTIS controller or SMPS2Dynamixel.
- Connect your DYNAMIXEL to power supply while it's off and turn on/off with the power switch.

Control Table

Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet.

EEPROM and RAM

Data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.

Address

It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.

Access

Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.

Initial Value

In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In

case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.

Highest/Lowest Byte

In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16bit, but it is divided into 8bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	29 (0X1D)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	15 (0X0F)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	160 (0XA0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
	20 (0X14)	Multi Turn Offset(L)	multi-turn offset least significant byte (LSB)	RW	0 (0X00)
	21 (0X12)	Multi Turn Offset(H)	multi-turn offset most significant byte (MSB)	RW	0 (0X00)
	22 (0X12)	Resolution Divider	Resolution divider	RW	1 (0X01)
RAM	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	D Gain	Derivative Gain	RW	0 (0X00)
	27 (0X1B)	I Gain	Integral Gain	RW	0 (0X00)
	28 (0X1C)	P Gain	Proportional Gain	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-

39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	–
40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	–
41 (0X29)	Present Load(H)	Highest byte of Current Load	R	–
42 (0X2A)	Present Voltage	Current Voltage	R	–
43 (0X2B)	Present Temperature	Current Temperature	R	–
44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
48 (0X30)	Punch(L)	Lowest byte of Punch	RW	0 (0X00)
49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)
73 (0X49)	Goal Acceleration	Goal Acceleration	RW	0 (0X00)

Address Function Help

EEPROM Area

Model Number

It represents the Model Number.

Firmware Version

It represents the firmware version.

ID

It is a unique number to identify Dynamixel.

The range from 0 to 252 (0xFC) can be used, and, especially, 254(0xFE) is used as the Broadcast ID.

If the Broadcast ID is used to transmit Instruction Packet, we can command to all Dynamixels.

Please be careful not to duplicate the ID of connected Dynamixel.

Baud Rate

It is the baud rate to communicate with controller. It is available in between 0~254(0xFE).

If the data value is in between 0~249 :

$$\text{Baudrate(BPS)} = 2000000 / (\text{Data} + 1)$$

Data	Set BPS	Target BPS	Tolerance
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

If the data value is over the 250 :

Data	Set BPS	Target BPS	Tolerance
250	2250000.0	2250000.0	0.000 %
251	2500000.0	2500000.0	0.000 %
252	3000000.0	3000000.0	0.000 %

Note : Maximum Baud Rate error of 3% is within the tolerance of UART communication.

Return Delay Time

It is the delay time per data value that takes from the transmission of Instruction Packet until the return of Status Packet. 0 to 254 (0xFE) can be used, and the delay time per data value is 2 usec.

That is to say, if the data value is 10, 20 usec is delayed. The initial value is 250 (0xFA) (i.e., 0.5 msec).

CW/CCW Angle Limit

Sets allowable position values (angles) for Goal Position (address 30 & 31)

- CW Angle Limit: Goal Position(Address 30, 31) minimum value
- CCW Angle Limit: Goal Position(Address 30, 31) maximum value

The following sets 2 modes operation based on CW and CCW values

Operation Type	CW / CCW
Wheel Mode	both are 0
Joint Mode	neither at 0
Multi-turn Mode	both are 4095

Wheel mode allows the motor can have limitless revolutions.

Joint mode allows robot with multiple joints

Multi-turn mode allows joints have range of controllable position values from -28672 to 28672.

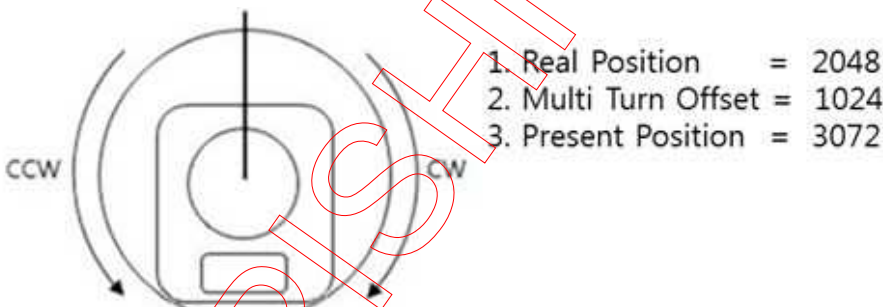
Multi Turn Offset

Adjusts position (zeroing). This value gets included in Present Position (36).

Present position + multi-turn offset.

Initial value is 0 and range is from -24576 to 24576

A Dynamixel with a position of 2048 with an applied offset of 1024 outputs a Present position of 3072.



Note: This feature is only applied in multi-turn mode and ignored in other modes.

Resolution Divider

It allows the user to change Dynamixel's resolution.

The default Resolution Divider Value is set as 1. (1 ~ 4 available)

When resolution is lowered, revolutions (in both directions) can be increased (up to 28 turns in each direction).

Present Position = Real Position / Resolution Divider

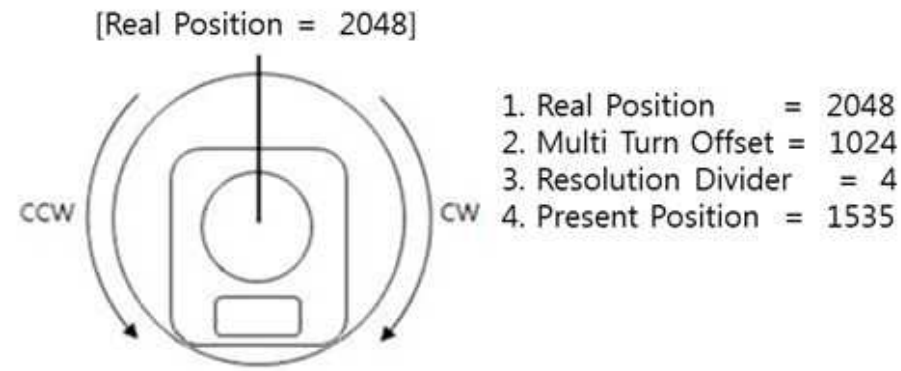
For example, a Real Position of 2048 with a Resolution Divider set as 2 will yield a Present Position value of 1024 ($2048/2 = 1024$). A

Dynamixel with Resolution Divider set as 2 will have a resolution 2048 for a single revolution.

The Present Position can be obtained while Multi-turn Offset and Resolution Divider are taken into account.

Present position = (Real Position / Resolution Divider) + Multi-turn Offset

For example, a Dynamixel with a Real Position of 2048 with a Resolution Divider set as 4 and Multi-turn Offset as 1024 will yield a Present Position of 1535 ((2048/4) + 1024 = 1535).



Note: This feature is only applied in multi-turn mode and ignored in other modes.

The Highest Limit Temperature

Caution : Do not set the temperature lower/higher than the default value.
When the temperature alarm shutdown occurs, wait 20 minutes to cool the temperature before re-use.
Using the product when the temperature is high may and can cause damage.

The Lowest (Highest) Limit Voltage

It is the operation range of voltage.
50 to 250 (0x32 ~ 0x96) can be used. The unit is 0.1V.
For example, if the value is 80, it is 8V.
If Present Voltage (Address42) is out of the range, Voltage Range Error Bit (Bit0) of Status Packet is returned as '1' and Alarm is triggered as set in the addresses 17 and 18.

Max Torque

It is the torque value of maximum output. 0 to 1023 (0x3FF) can be used, and the unit is about 0.1%.
For example, Data 1023 (0x3FF) means that Dynamixel will use 100% of the maximum torque it can produce while Data 512 (0x200) means that Dynamixel will use 50% of the maximum torque. When the power is turned on, Torque Limit (Addresses 34 and 35) uses the value as the initial value.

Status Return Level

It decides how to return Status Packet. There are three ways like the below table.

Value	Return of Status Packet
0	No return against all commands (Except PING Command)
1	Return only for the READ command
2	Return for all commands

When Instruction Packet is Broadcast ID, Status Packet is not returned regardless of Status Return Level.

Alarm LED

Alarm Shutdown

Dynamixel can protect itself by detecting errors occur during the operation.
The errors can be set are as the table below.

Bit	Name	Contents
Bit 7	0	-

Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	CheckSum Error	When the Checksum of the transmitted Instruction Packet is invalid
Bit 3	Range Error	When the command is given beyond the range of usage
Bit 2	OverHeating Error	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Error	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

It is possible to make duplicate set since the function of each bit is run by the logic of 'OR'. That is, if 0X05 (binary 00000101) is set, both Input Voltage Error and Overheating Error can be detected.

If errors occur, in case of Alarm LED, the LED blinks; in case of Alarm Shutdown, the motor output becomes 0 % by making the value of Torque Limit(Address 34, 35) as 0.

RAM Area

Torque Enable

Value	Meaning
0	Keeps Torque from generating by interrupting the power of motor.
1	Generates Torque by impressing the power to the motor.

LED

Bit	Meaning	Meaning
bit7		
bit6		
bit5		
bit4		
bit3		
bit2	BLUE LED	When Bit is set the blue LED turns on
bit1	GREEN LED	When Bit is set the green LED turns on
bit0	RED LED	When Bit is set the red LED turns on

PID Gain

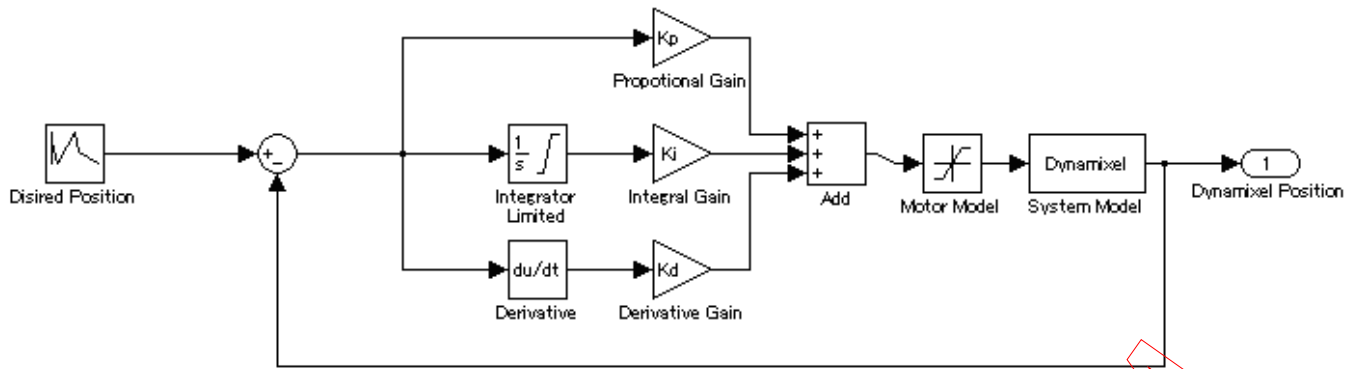
MX series will use the PID controller as a main control method.

P gain refers to the value of proportional band.

I gain refers to the value of integral action.

D Gain refers to the value of derivative action.

Gains values are in between 0~254.



$$K_p = \text{P Gain} / 8$$

$$K_i = \text{I Gain} * 1000 / 2048$$

$$K_d = \text{D Gain} * 4 / 1000$$

※ The relationship between Compliance Slope and PID

Slope	P Gain
8	128
16	64
32	32
64	16
128	8

The less the P gain, The larger the back lash, and the weaker the amount of output near goal position.

At some extent, it is like a combined concept of margin and slope.

It does not exactly match the previous concept of compliance. So it is obvious if you see the difference in terms of motion.

※ Explanation for PID required.

For the brief explanation about general PID, please refer to the website(link) below.

http://en.wikipedia.org/wiki/PID_controller

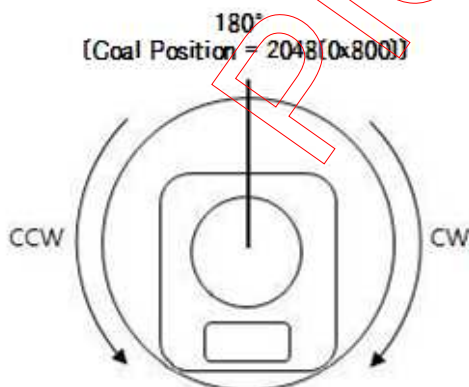
FYI, PID control theory is not only limited to the control of motor(actuator) but is a generic theory that can be applied to all kinds of control.

Goal Position

It is a position value of destination.

0 to 4095 (0xFFFF) is available. The unit is 0.088 degree.

If Goal Position is out of the range, Angle Limit Error Bit (Bit1) of Status Packet is returned as '1' and Alarm is triggered as set in Alarm LED/Shutdown.



Moving Speed

- Join Mode, Multi-Turn mode

It is a moving speed to Goal Position.

0~1023 (0X3FF) can be used, and the unit is about 0.114rpm.

If it is set to 0, it means the maximum rpm of the motor is used without controlling the speed.

If it is 1023, it is about 117.07rpm.

For example, if it is set to 300, it is about 34.33 rpm.

- Wheel Mode

It is a moving speed to Goal direction.

0~2047 (0X7FF) can be used, and the unit is about 0.114rpm.

If a value in the range of 0~1023 is used, it is stopped by setting to 0 while rotating to CCW direction.

If a value in the range of 1024~2047 is used, it is stopped by setting to 1024 while rotating to CW direction.

That is, the 10th bit becomes the direction bit to control the direction.

Note: This mode allows to check max rpm. Any values set higher than max rpm will not take effect.

Torque Limit

It is the value of the maximum torque limit.

0 to 1023 (0x3FF) is available, and the unit is about 0.1%.

For example, if the value is 512, it is about 50%; that means only 50% of the maximum torque will be used.

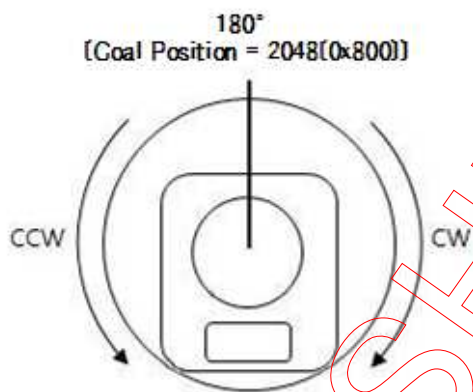
If the power is turned on, the value of Max Torque (Address 14, 15) is used as the initial value.

Notes: If the function of Alarm Shutdown is triggered, the motor loses its torque because the value becomes 0. At this moment, if the value is changed to the value other than 0, the motor can be used again.

Present Position

It is the current position value of Dynamixel.

The range of the value is 0~4095 (0xFFFF), and the unit is 0.088 degree.



In multi-turn mode range is from -28672 to 28672 with unit values dependent on Resolution Divider ($0.088 * \text{Resolution Divider}$)

Note: in multi-turn mode Present position depends on resolution divider and multi-turn offset For more information turn to the section on [Multi Turn offset](#) and [Resolution Divider](#)

Present Speed

Is the current moving speed.

0~2047 (0x000~0X7FF) can be used.

If a value is in the range of 0~1023 then the motor rotates to the CCW direction.

If a value is in the range of 1024~2047 then the motor rotates to the CW direction.

The 10th bit becomes the direction bit to control the direction; 0 and 1024 are equal.

The value unit is about 0.11rpm.

For example, if it is set to 300 then the motor is moving to the CCW direction at a rate of about 34.33rpm.

Present Load

It means currently applied load.

The range of the value is 0~2047, and the unit is about 0.1%.

If the value is 0~1023, it means the load works to the CCW direction.

If the value is 1024~2047, it means the load works to the CW direction.

That is, the 10th bit becomes the direction bit to control the direction, and 1024 is equal to 0.

For example, the value is 512, it means the load is detected in the direction of CCW about 50% of the maximum torque.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Notes: Current load is inferred from the internal torque value, not from Torque sensor etc.

For that reason, it cannot be used to measure weight or torque; however, it must be used only to detect which direction the force works.

Present Voltage

It is the size of the current voltage supplied.

This value is 10 times larger than the actual voltage. For example, when 10V is supplied, the data value is 100 (0x64).

Present Temperature

It is the internal temperature of Dynamixel in Celsius.

Data value is identical to the actual temperature in Celsius. For example, if the data value is 85 (0x55), the current internal temperature is 85°C.

Registered Instruction

Value	Meaning
0	There are no commands transmitted by REG_WRITE.
1	There are commands transmitted by REG_WRITE.

Notes: If ACTION command is executed, the value is changed into 0.

Moving

Value	Meaning
0	Goal position command execution is completed.
1	Goal position command execution is in progress.

Lock

Value	Meaning
0	EEPROM area can be modified.
1	EEPROM area cannot be modified.

Caution: If Lock is set to 1, the power must be turned off and then turned on again to change into 0.

Punch

Current to drive motor is at minimum.
Can choose vales from 0x00 to 0x3FF.

Goal Acceleration

This is Goal Acceleration value.

It can be used from 0~254(0xFE), and the unit is approximately 8.583 Degree / sec².

When it is set to 0, there is no control over acceleration and moves with the maximum acceleration of the motor.

When the goal speed is set to 0, there is no control over acceleration and moves with the maximum acceleration of the motor.

When it is set to 254, it becomes 2180 Degree / sec²

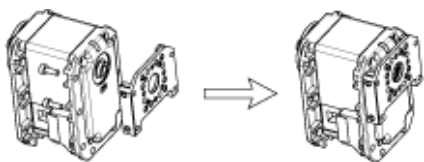
For example, the current speed of Dynamixel is 0, and Goal acceleration is 10,

The speed of Dynamixel after 1 second will be 14.3 RPM.

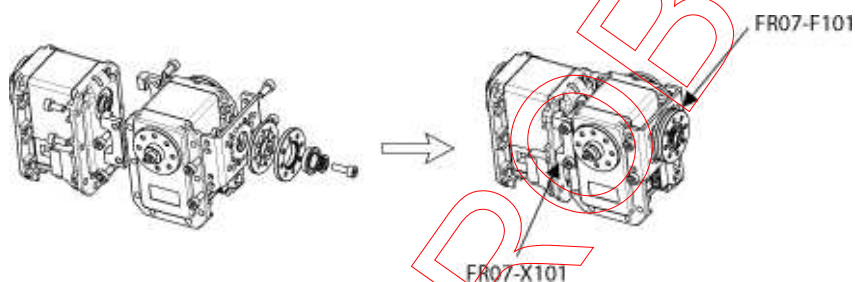
Option Frame

The types of MX-28 option frame are as follows.

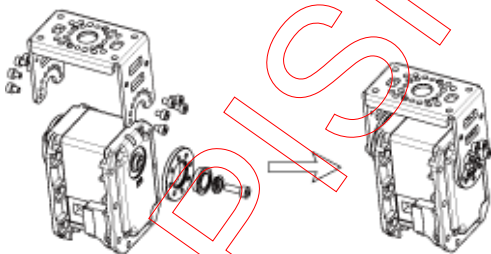
FR07-B101



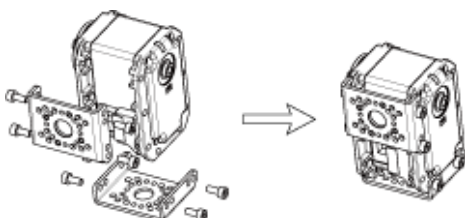
FR07-F101_FR07-X101



FR07-H101



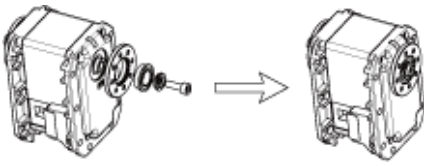
FR07-S101



Horn

The types of MX-28 Horns are as follows.

HN07-I101

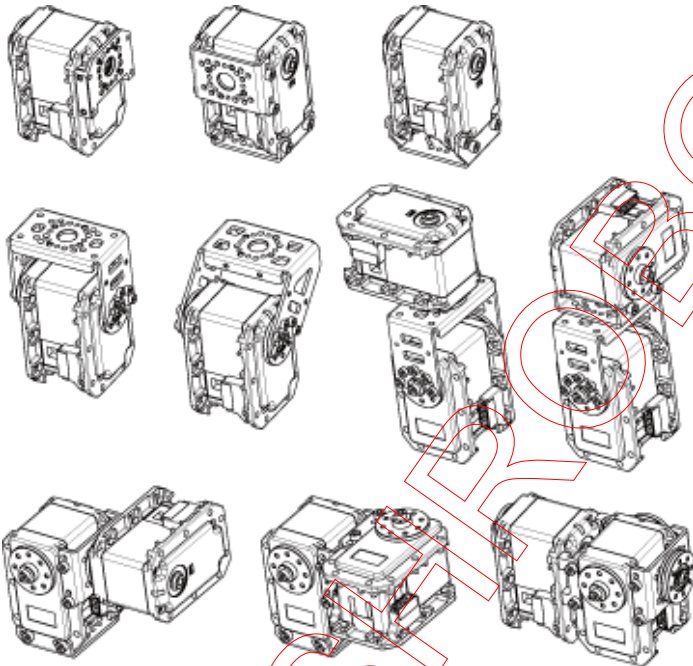


HN07-N101



Combination

The following example shows the combination structure of option frames and horns.



Dimension

Drawing Information : [DOWNLOAD](#) MX28Dimension.pdf

Videos

HOW TO REPLACE GEARS

2.7V Dual Channel 12-Bit A/D Converter with SPI[®] Serial Interface

FEATURES

- 12-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL (MCP3202-B)
- ± 2 LSB max INL (MCP3202-C)
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI[®] serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 100ksps max. sampling rate at $V_{DD} = 5V$
- 50ksps max. sampling rate at $V_{DD} = 2.7V$
- Low power CMOS technology
 - 500nA typical standby current, 5 μ A max.
 - 550 μ A max. active current at 5V
- Industrial temp range: -40°C to +85°C
- 8-pin PDIP SOIC and TSSOP packages

APPLICATIONS

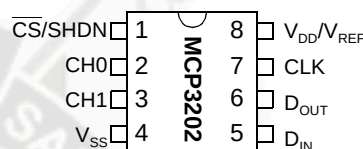
- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

DESCRIPTION

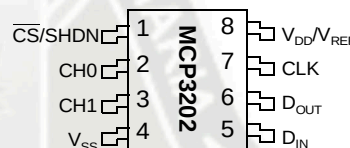
The Microchip Technology Inc. MCP3202 is a successive approximation 12-bit Analog-to-Digital (A/D) Converter with on-board sample and hold circuitry. The MCP3202 is programmable to provide a single pseudo-differential input pair or dual single-ended inputs. Differential Nonlinearity (DNL) is specified at ± 1 LSB, and Integral Nonlinearity (INL) is offered in ± 1 LSB (MCP3202-B) and ± 2 LSB (MCP3202-C) versions. Communication with the device is done using a simple serial interface compatible with the SPI protocol. The device is capable of conversion rates of up to 100ksps at 5V and 50ksps at 2.7V. The MCP3202 device operates over a broad voltage range (2.7V - 5.5V). Low current design permits operation with typical standby and active currents of only 500nA and 375 μ A, respectively. The MCP3202 is offered in 8-pin PDIP, TSSOP and 150mil SOIC packages.

PACKAGE TYPES

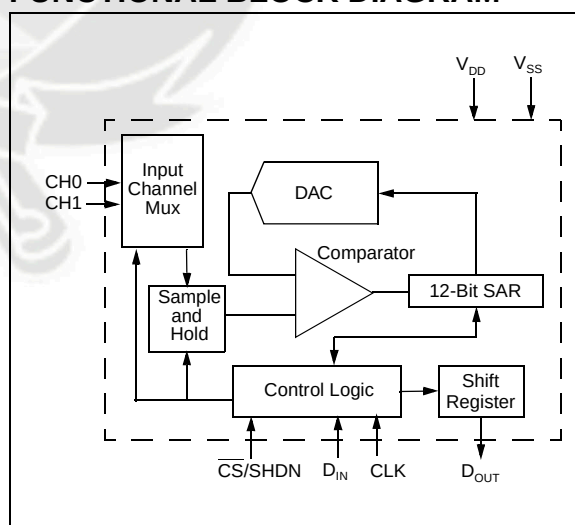
PDIP



SOIC, TSSOP



FUNCTIONAL BLOCK DIAGRAM



1.0 ELECTRICAL CHARACTERISTICS

1.1 Maximum Ratings*

V_{DD} 7.0V
 All inputs and outputs w.r.t. V_{SS} -0.6V to V_{DD} +0.6V
 Storage temperature -65°C to +150°C
 Ambient temp. with power applied -65°C to +125°C
 Soldering temperature of leads (10 seconds) .. +300°C
 ESD protection on all pins > 4kV

*Notice: Stresses above those listed under "Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

NAME	FUNCTION
V_{DD}/V_{REF}	+2.7V to 5.5V Power Supply and Reference Voltage Input
CH0	Channel 0 Analog Input
CH1	Channel 1 Analog Input
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}/SHDN$	Chip Select/Shutdown Input

ELECTRICAL CHARACTERISTICS

All parameters apply at $V_{DD} = 5.5V$, $V_{SS} = 0V$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$, $f_{SAMPLE} = 100ksps$ and $f_{CLK} = 18 \cdot f_{SAMPLE}$ unless otherwise noted.

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Conversion Rate						
Conversion Time	t_{CONV}			12	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}			100 50	ksps ksps	$V_{DD} = V_{REF} = 5V$ $V_{DD} = V_{REF} = 2.7V$
DC Accuracy						
Resolution			12		bits	
Integral Nonlinearity	INL		± 0.75 ± 1	± 1 ± 2	LSB LSB	MCP3202-B MCP3202-C
Differential Nonlinearity	DNL		± 0.5	± 1	LSB	No missing codes over temperature
Offset Error			± 1.25	± 3	LSB	
Gain Error			± 1.25	± 5	LSB	
Dynamic Performance						
Total Harmonic Distortion			-82		dB	$V_{IN} = 0.1V$ to $4.9V@1kHz$
Signal to Noise and Distortion (SINAD)			72		dB	$V_{IN} = 0.1V$ to $4.9V@1kHz$
Spurious Free Dynamic Range			86		dB	$V_{IN} = 0.1V$ to $4.9V@1kHz$
Analog Inputs						
Input Voltage Range for CH0 or CH1 in Single-Ended Mode		V_{SS}		V_{REF}	V	
Input Voltage Range for IN+ in Pseudo-Differential Mode		IN-		$V_{REF} + IN-$		See Sections 3.1 and 4.1
Input Voltage Range for IN- in Pseudo-Differential Mode		$V_{SS} - 100$		$V_{SS} + 100$	mV	See Sections 3.1 and 4.1
Leakage Current			.001	± 1	μA	
Switch Resistance	R_{SS}		1K		Ω	See Figure 4-1
Sample Capacitor	C_{SAMPLE}		20		pF	See Figure 4-1

ELECTRICAL CHARACTERISTICS (CONTINUED)

All parameters apply at $V_{DD} = 5.5V$, $V_{SS} = 0V$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$, $f_{SAMPLE} = 100kps$ and $f_{CLK} = 18 \cdot f_{SAMPLE}$ unless otherwise noted.						
PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Digital Input/Output						
Data Coding Format		Straight Binary				
High Level Input Voltage	V_{IH}	$0.7 V_{DD}$			V	
Low Level Input Voltage	V_{IL}			$0.3 V_{DD}$	V	
High Level Output Voltage	V_{OH}	4.1			V	$I_{OH} = -1mA$, $V_{DD} = 4.5V$
Low Level Output Voltage	V_{OL}			0.4	V	$I_{OL} = 1mA$, $V_{DD} = 4.5V$
Input Leakage Current	I_{LI}	-10		10	μA	$V_{IN} = V_{SS}$ or V_{DD}
Output Leakage Current	I_{LO}	-10		10	μA	$V_{OUT} = V_{SS}$ or V_{DD}
Pin Capacitance (All Inputs/Outputs)	C_{IN} , C_{OUT}			10	pF	$V_{DD} = 5.0V$ (Note 1) $T_{AMB} = 25^{\circ}C$, $f = 1 MHz$
Timing Parameters						
Clock Frequency	f_{CLK}			1.8 0.9	MHz MHz	$V_{DD} = 5V$ (Note 2) $V_{DD} = 2.7V$ (Note 2)
Clock High Time	t_{HI}	250			ns	
Clock Low Time	t_{LO}	250			ns	
CS Fall To First Rising CLK Edge	t_{SUCS}	100			ns	
Data Input Setup Time	t_{SU}			50	ns	
Data Input Hold Time	t_{HD}			50	ns	
CLK Fall To Output Data Valid	t_{DO}			200	ns	See Test Circuits, Figure 1-2
CLK Fall To Output Enable	t_{EN}			200	ns	See Test Circuits, Figure 1-2
CS Rise To Output Disable	t_{DIS}			100	ns	See Test Circuits, Figure 1-2 Note 1
CS Disable Time	t_{CSH}	500			ns	
D_{OUT} Rise Time	t_R			100	ns	See Test Circuits, Figure 1-2 Note 1
D_{OUT} Fall Time	t_F			100	ns	See Test Circuits, Figure 1-2 Note 1
Power Requirements						
Operating Voltage	V_{DD}	2.7		5.5	V	
Operating Current	I_{DD}		375	550	μA	$V_{DD} = 5.0V$, D_{OUT} unloaded
Standby Current	I_{DDS}		0.5	5	μA	$\overline{CS} = V_{DD} = 5.0V$

Note 1: This parameter is guaranteed by characterization and not 100% tested.

Note 2: Because the sample cap will eventually lose charge, effective clock rates below 10kHz can affect linearity performance, especially at elevated temperatures. See Section 6.2 for more information.

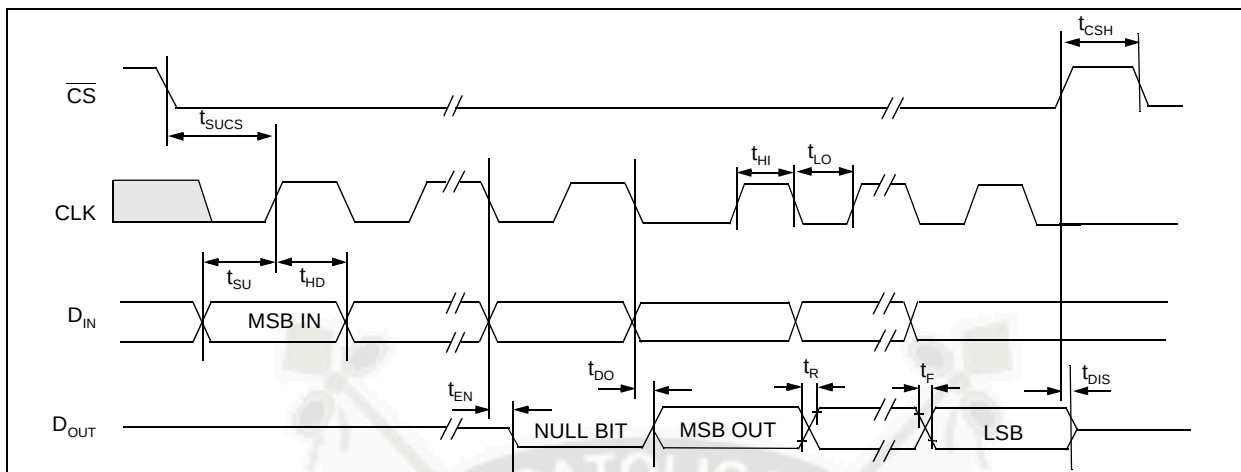


FIGURE 1-1: Serial Timing.

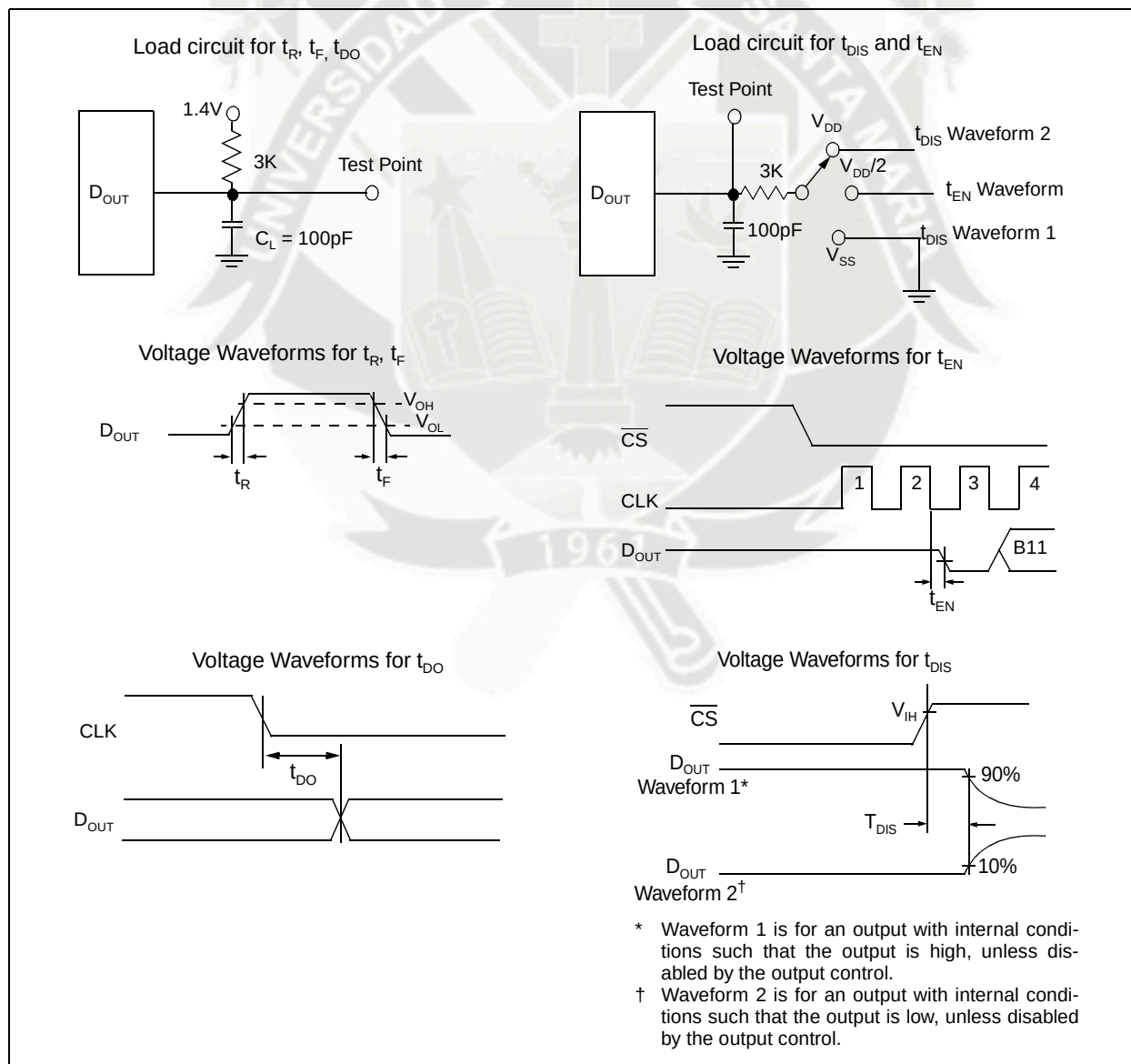


FIGURE 1-2: Test Circuits.

2.0 TYPICAL PERFORMANCE CHARACTERISTICS

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100\text{kpsps}$, $f_{CLK} = 18 \cdot f_{SAMPLE}$, $T_A = 25^\circ\text{C}$

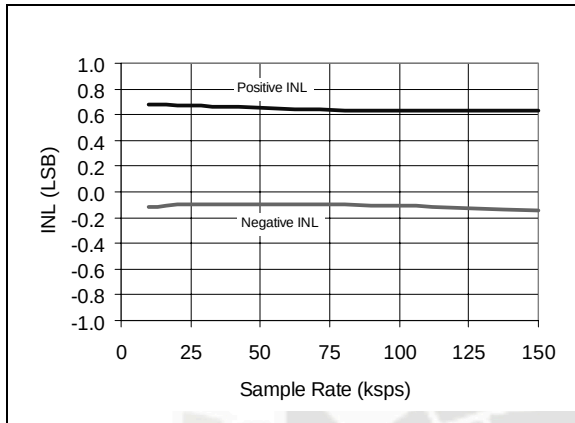


FIGURE 2-1: Integral Nonlinearity (INL) vs. Sample Rate.

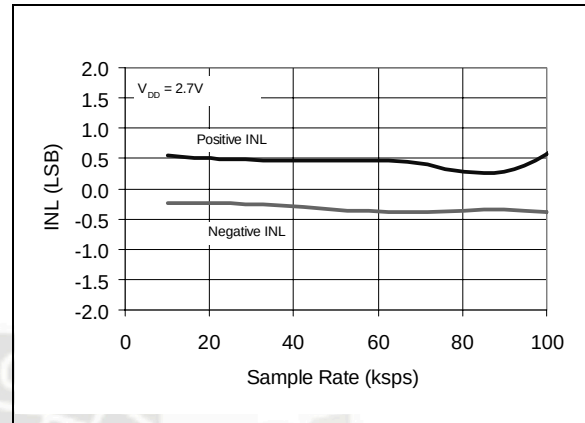


FIGURE 2-4: Integral Nonlinearity (INL) vs. Sample Rate ($V_{DD} = 2.7V$).

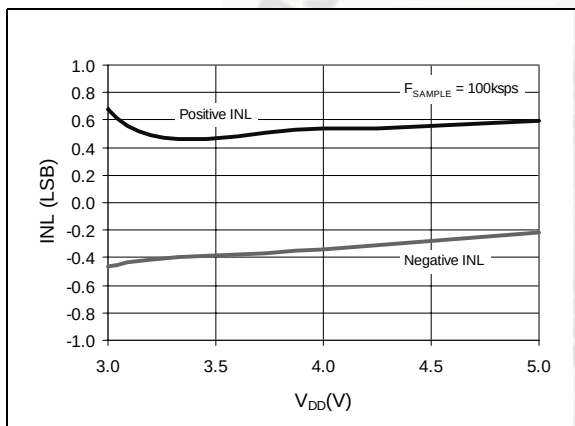


FIGURE 2-2: Integral Nonlinearity (INL) vs. V_{DD} .

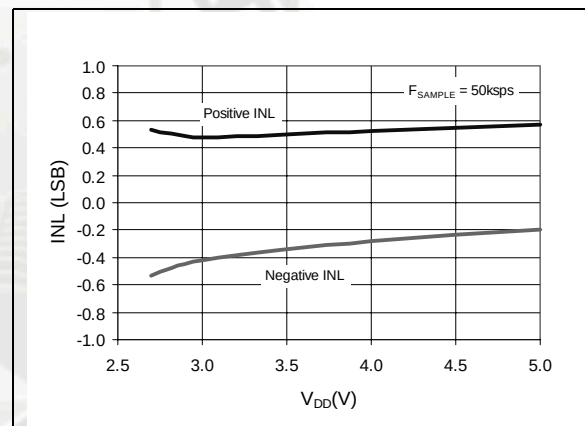


FIGURE 2-5: Integral Nonlinearity (INL) vs. V_{DD} .

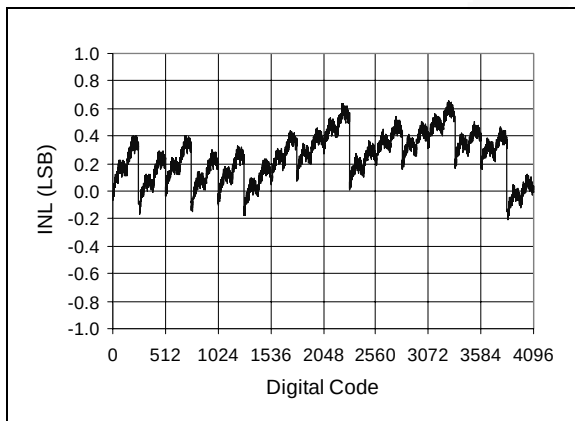


FIGURE 2-3: Integral Nonlinearity (INL) vs. Code (Representative Part).

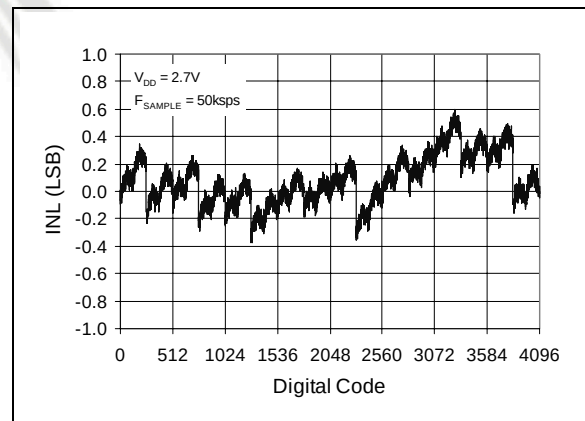


FIGURE 2-6: Integral Nonlinearity (INL) vs. Code (Representative Part, $V_{DD} = 2.7V$).

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100\text{kpsps}$, $f_{CLK} = 18 \cdot f_{SAMPLE}$, $T_A = 25^\circ\text{C}$

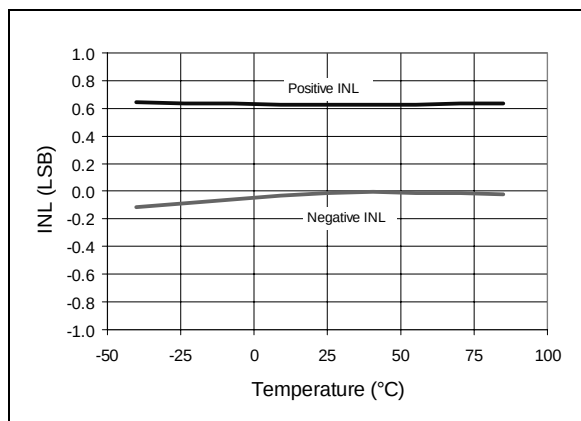


FIGURE 2-7: Integral Nonlinearity (INL) vs. Temperature.

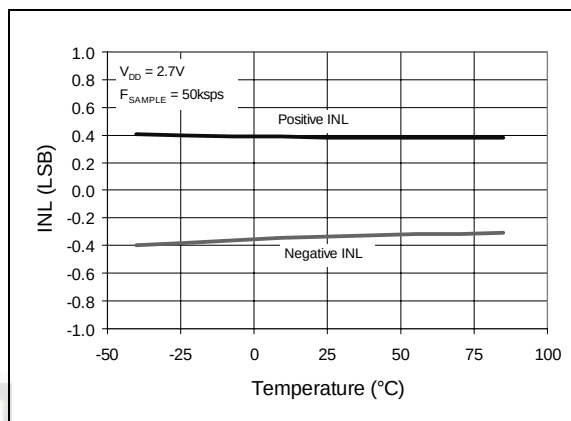


FIGURE 2-10: Integral Nonlinearity (INL) vs. Temperature ($V_{DD} = 2.7V$).

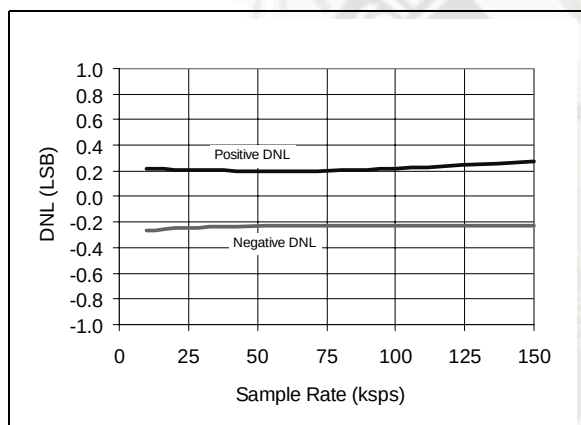


FIGURE 2-8: Differential Nonlinearity (DNL) vs. Sample Rate.

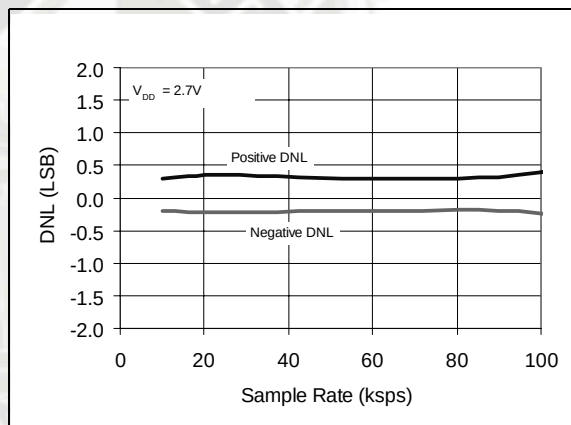


FIGURE 2-11: Differential Nonlinearity (DNL) vs. Sample Rate ($V_{DD} = 2.7V$).

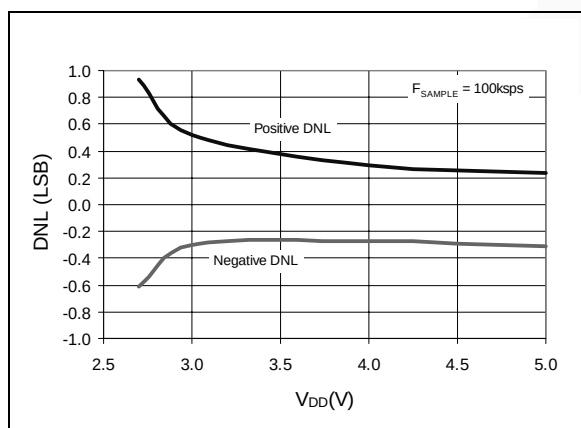


FIGURE 2-9: Differential Nonlinearity (DNL) vs. V_{DD} .

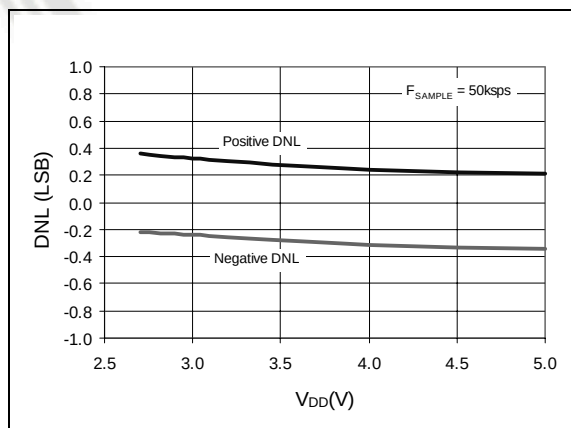


FIGURE 2-12: Differential Nonlinearity (DNL) vs. V_{DD} .

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100kps$, $f_{CLK} = 18 \cdot f_{SAMPLE}$, $T_A = 25^\circ C$

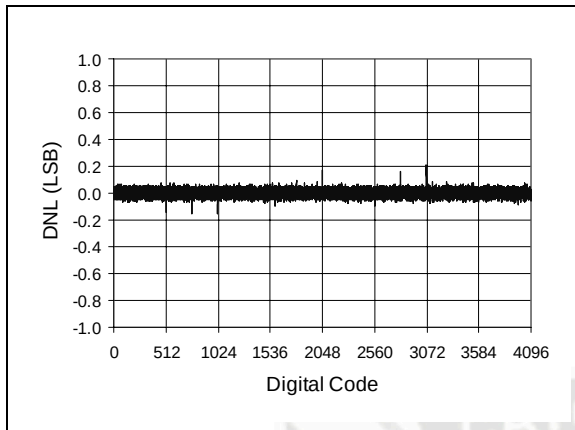


FIGURE 2-13: Differential Nonlinearity (DNL) vs. Code (Representative Part).

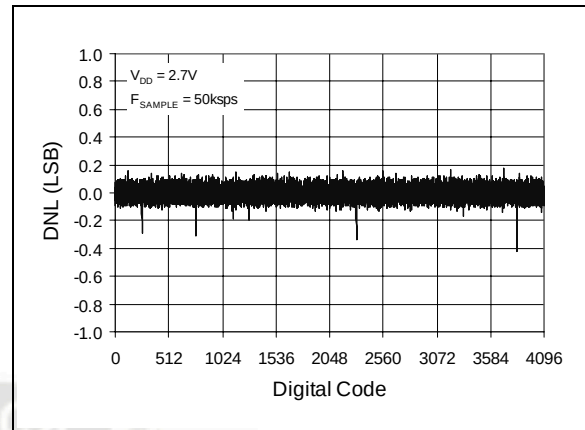


FIGURE 2-16: Differential Nonlinearity (DNL) vs. Code (Representative Part, $V_{DD} = 2.7V$).

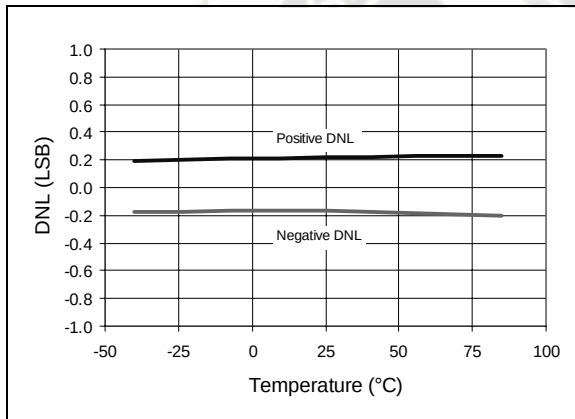


FIGURE 2-14: Differential Nonlinearity (DNL) vs. Temperature.

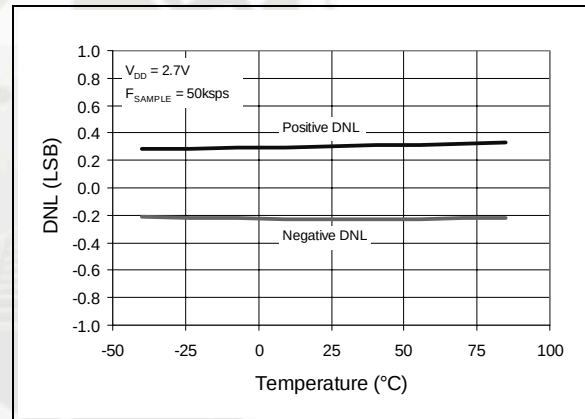


FIGURE 2-17: Differential Nonlinearity (DNL) vs. Temperature ($V_{DD} = 2.7V$).

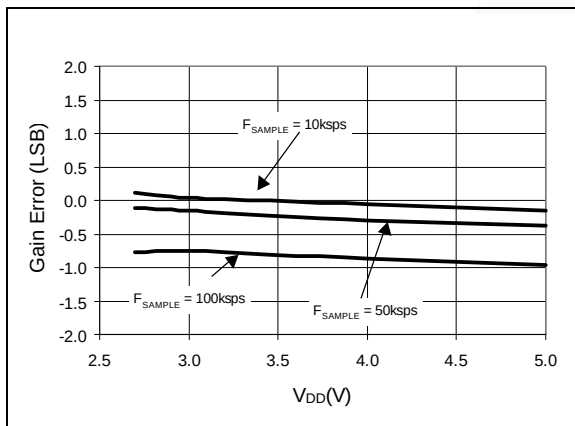


FIGURE 2-15: Gain Error vs. V_{DD} .

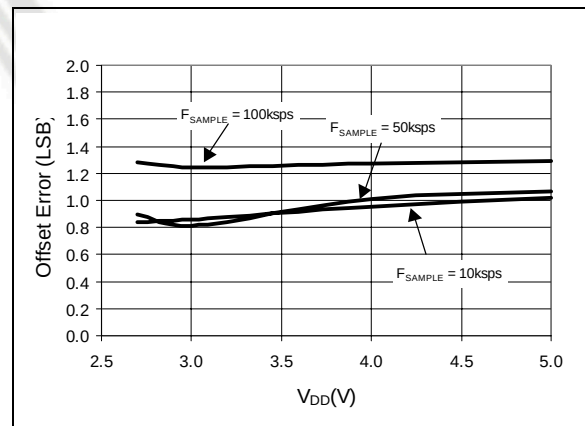


FIGURE 2-18: Offset Error vs. V_{DD} .

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100\text{kpsps}$, $f_{CLK} = 18 * f_{SAMPLE}$, $T_A = 25^\circ\text{C}$

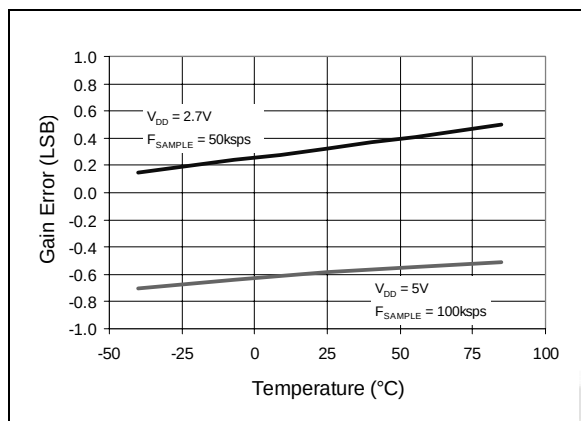


FIGURE 2-19: Gain Error vs. Temperature.

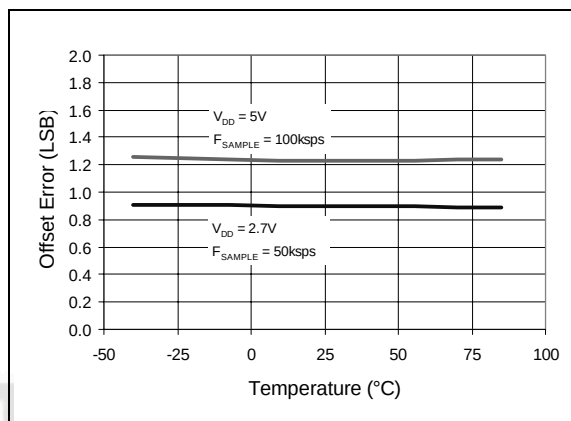


FIGURE 2-22: Offset Error vs. Temperature.

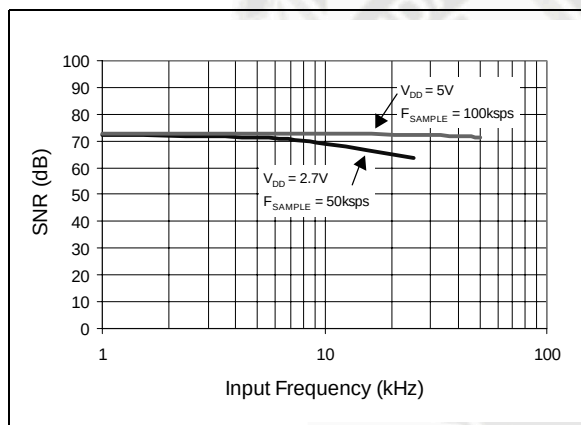


FIGURE 2-20: Signal to Noise Ratio (SNR) vs. Input Frequency.

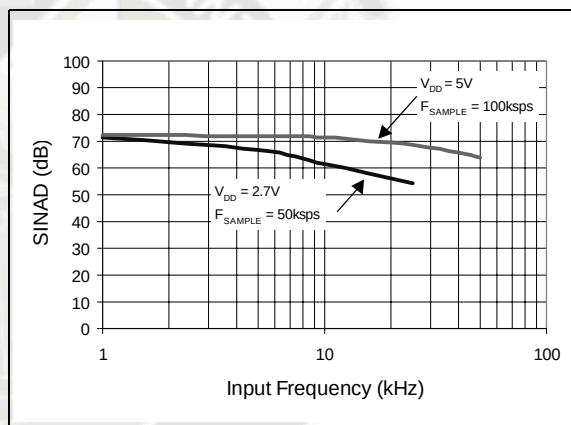


FIGURE 2-23: Signal to Noise and Distortion (SINAD) vs. Input Frequency.

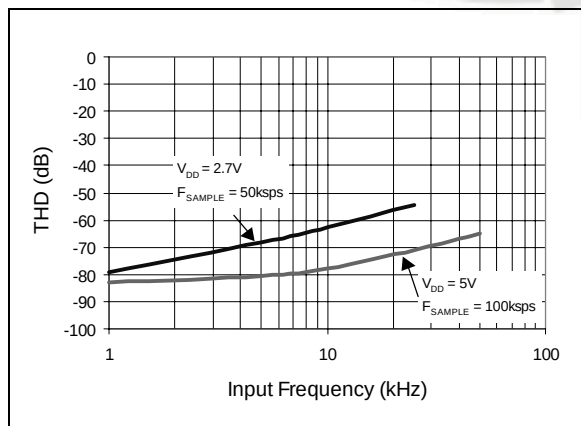


FIGURE 2-21: Total Harmonic Distortion (THD) vs. Input Frequency.

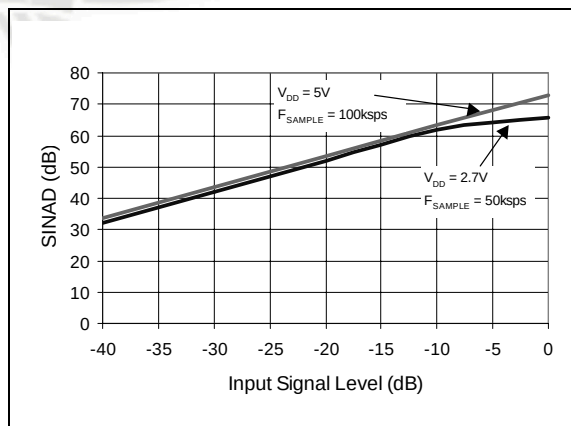


FIGURE 2-24: Signal to Noise and Distortion (SINAD) vs. Signal Level.

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100\text{ksps}$, $f_{CLK} = 18 \cdot f_{SAMPLE}$, $T_A = 25^\circ\text{C}$

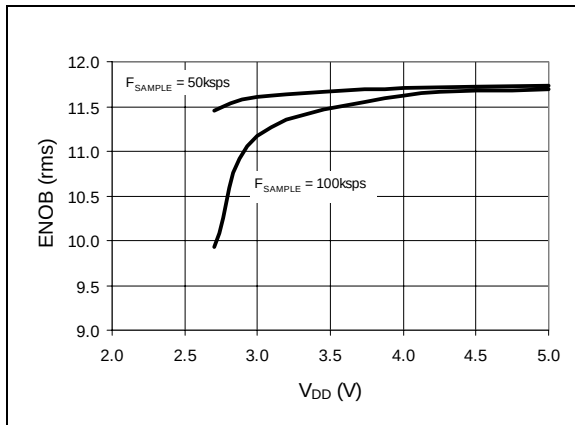


FIGURE 2-25: Effective number of bits (ENOB) vs. V_{DD} .

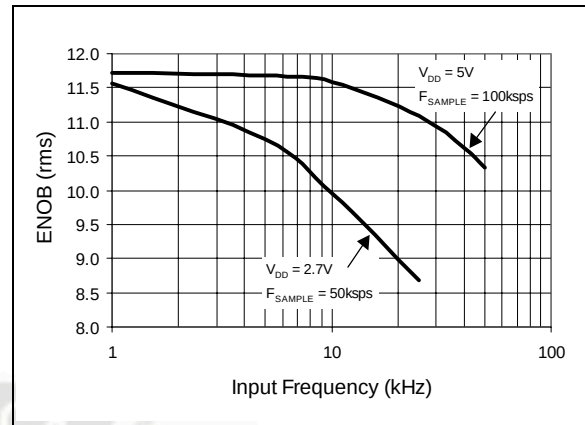


FIGURE 2-28: Effective Number of Bits (ENOB) vs. Input Frequency.

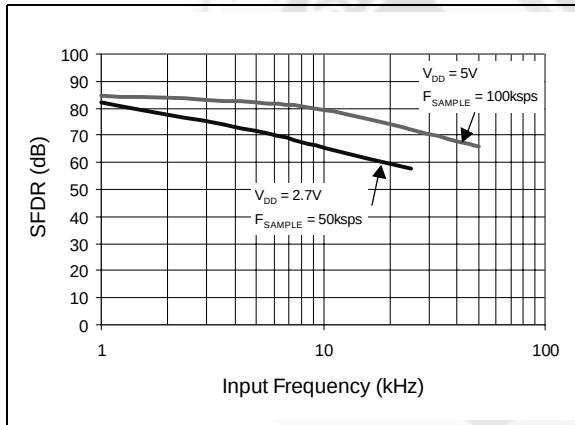


FIGURE 2-26: Spurious Free Dynamic Range (SFDR) vs. Input Frequency.

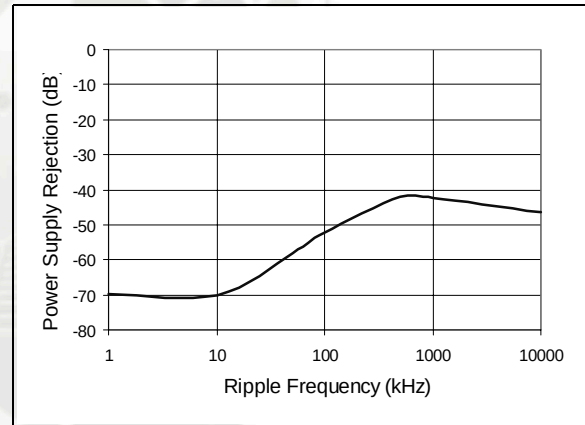


FIGURE 2-29: Power Supply Rejection (PSR) vs. Ripple Frequency.

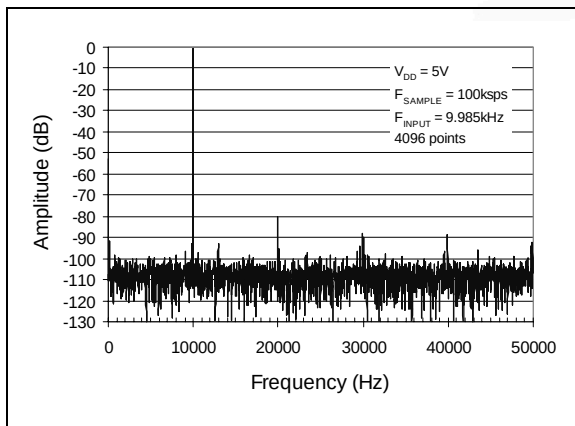


FIGURE 2-27: Frequency Spectrum of 10kHz input (Representative Part).

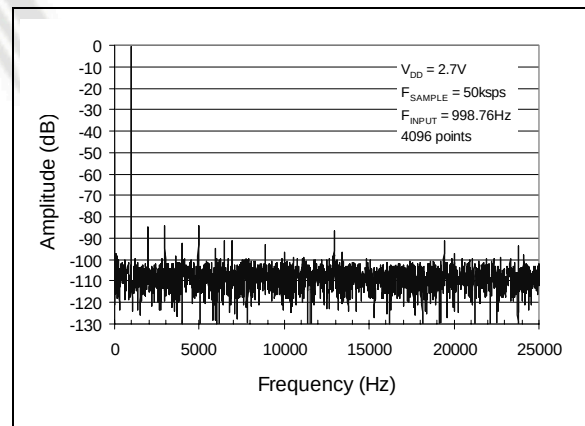


FIGURE 2-30: Frequency Spectrum of 1kHz input (Representative Part, $V_{DD} = 2.7V$).

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, $f_{SAMPLE} = 100ksps$, $f_{CLK} = 18 * f_{SAMPLE}$, $T_A = 25^{\circ}C$

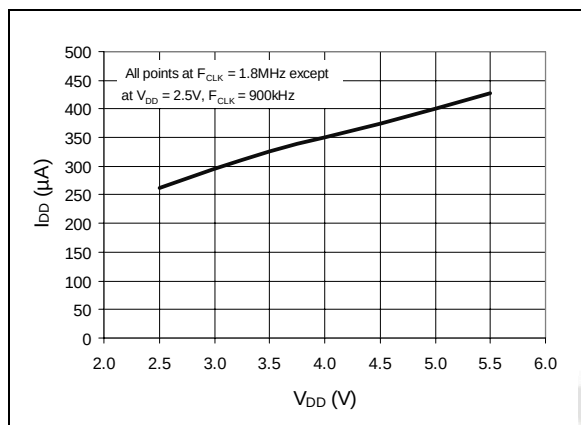


FIGURE 2-31: I_{DD} vs. V_{DD} .

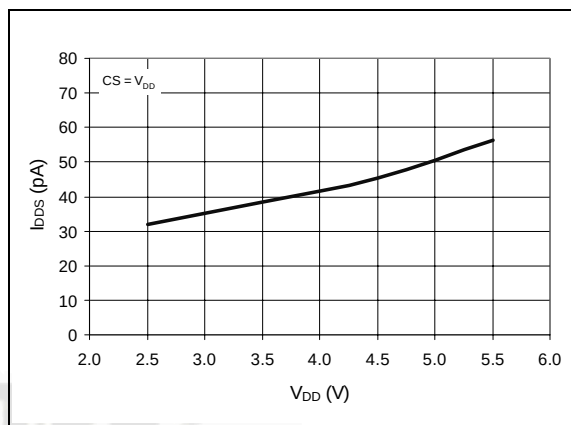


FIGURE 2-34: I_{DDS} vs. V_{DD} .

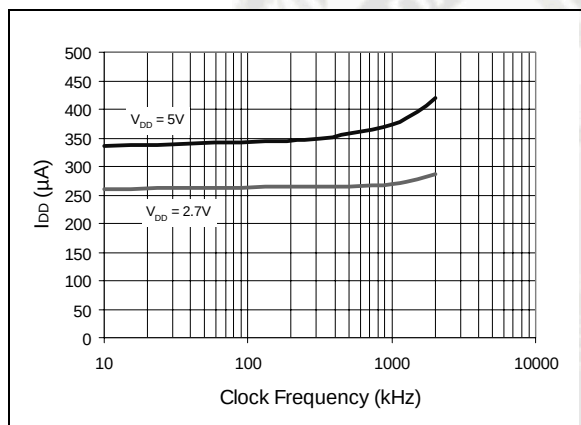


FIGURE 2-32: I_{DD} vs. Clock Frequency.

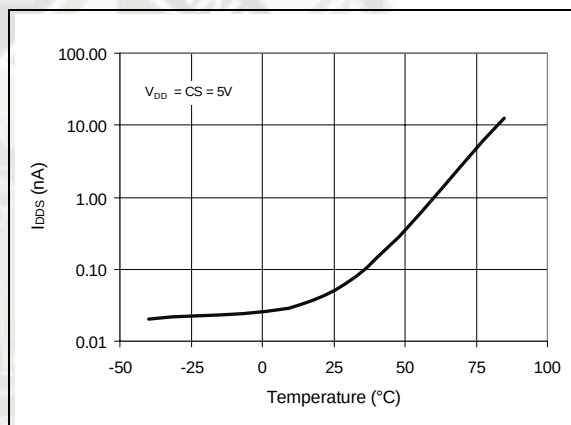


FIGURE 2-35: I_{DDS} vs. Temperature.

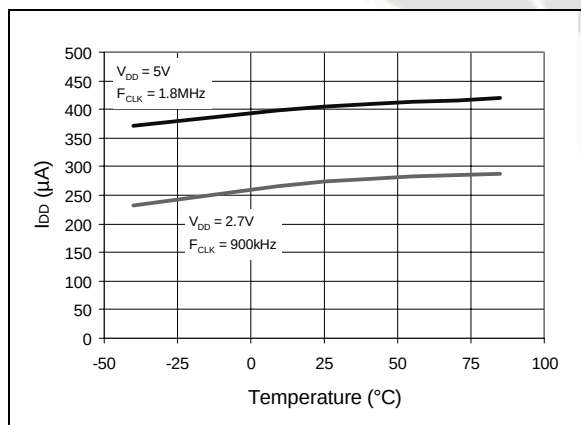


FIGURE 2-33: I_{DD} vs. Temperature.

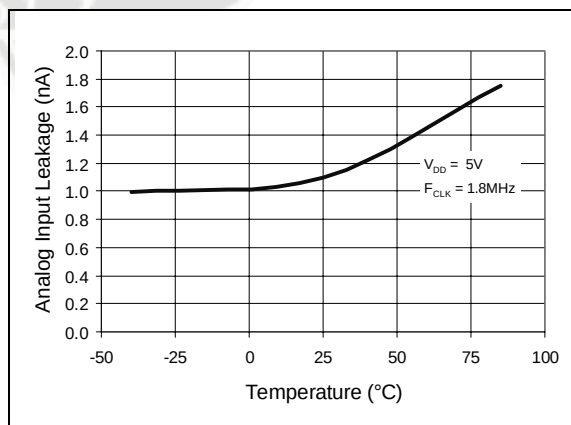


FIGURE 2-36: Analog Input leakage current vs. Temperature.

3.0 PIN DESCRIPTIONS

3.1 CH0/CH1

Analog inputs for channels 0 and 1 respectively. These channels can be programmed to be used as two independent channels in single ended-mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN-. See Section 5.0 for information on programming the channel configuration.

3.2 $\overline{\text{CS}}/\text{SHDN}$ (Chip Select/Shutdown)

The $\overline{\text{CS}}/\text{SHDN}$ pin is used to initiate communication with the device when pulled low and will end a conversion and put the device in low power standby when pulled high. The $\overline{\text{CS}}/\text{SHDN}$ pin must be pulled high between conversions.

3.3 CLK (Serial Clock)

The SPI clock pin is used to initiate a conversion and to clock out each bit of the conversion as it takes place. See Section 6.2 for constraints on clock speed.

3.4 DIN (Serial Data Input)

The SPI port serial data input pin is used to clock in input channel configuration data.

3.5 DOUT (Serial Data output)

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

4.0 DEVICE OPERATION

The MCP3202 A/D Converter employs a conventional SAR architecture. With this architecture, a sample is acquired on an internal sample/hold capacitor for 1.5 clock cycles starting on the second rising edge of the serial clock after the start bit has been received. Following this sample time, the input switch of the converter opens and the device uses the collected charge on the internal sample and hold capacitor to produce a serial 12-bit digital output code. Conversion rates of 100ksps are possible on the MCP3202. See Section 6.2 for information on minimum clock rates. Communication with the device is done using a 3-wire SPI-compatible interface.

4.1 Analog Inputs

The MCP3202 device offers the choice of using the analog input channels configured as two single-ended inputs or a single pseudo-differential input. Configuration is done as part of the serial command before each conversion begins. When used in the pseudo-differential mode, CH0 and CH1 are programmed as the IN+ and IN- inputs as part of the command string transmitted to the device. The IN+ input can range from IN- to V_{REF} ($V_{\text{REF}} + \text{IN-}$). The IN- input is limited to $\pm 100\text{mV}$ from the V_{SS} rail. The IN- input can be used to cancel small signal common-mode noise which is present on both the IN+ and IN- inputs.

For the A/D Converter to meet specification, the charge holding capacitor (C_{SAMPLE}) must be given enough time to acquire a 12-bit accurate voltage level during the 1.5 clock cycle sampling period. The analog input model is shown in Figure 4-1.

In this diagram, it is shown that the source impedance (R_{S}) adds to the internal sampling switch (R_{SS}) impedance, directly affecting the time that is required to charge the capacitor, C_{SAMPLE} . Consequently, larger source impedances increase the offset, gain, and integral linearity errors of the conversion.

Ideally, the impedance of the signal source should be near zero. This is achievable with an operational amplifier such as the MCP601 which has a closed loop output impedance of tens of ohms. The adverse affects of higher source impedances are shown in Figure 4-2.

When operating in the pseudo-differential mode, if the voltage level of IN+ is equal to or less than IN-, the resultant code will be 000h. If the voltage at IN+ is equal to or greater than $\{[V_{\text{REF}} + (\text{IN-})] - 1 \text{ LSB}\}$, then the output code will be FFFh. If the voltage level at IN- is more than 1 LSB below V_{SS} , then the voltage level at the IN+ input will have to go below V_{SS} to see the 000h output code. Conversely, if IN- is more than 1 LSB above V_{SS} , then the FFFh code will not be seen unless the IN+ input level goes above V_{REF} level.

4.2 Digital Output Code

The digital output code produced by an A/D Converter is a function of the input signal and the reference voltage. For the MCP3202, V_{DD} is used as the reference voltage. As the V_{DD} level is reduced, the LSB size is reduced accordingly. The theoretical digital output code produced by the A/D Converter is shown below.

$$\text{Digital Output Code} = \frac{4096 * V_{\text{IN}}}{V_{\text{DD}}}$$

where:

V_{IN} = analog input voltage

V_{DD} = supply voltage

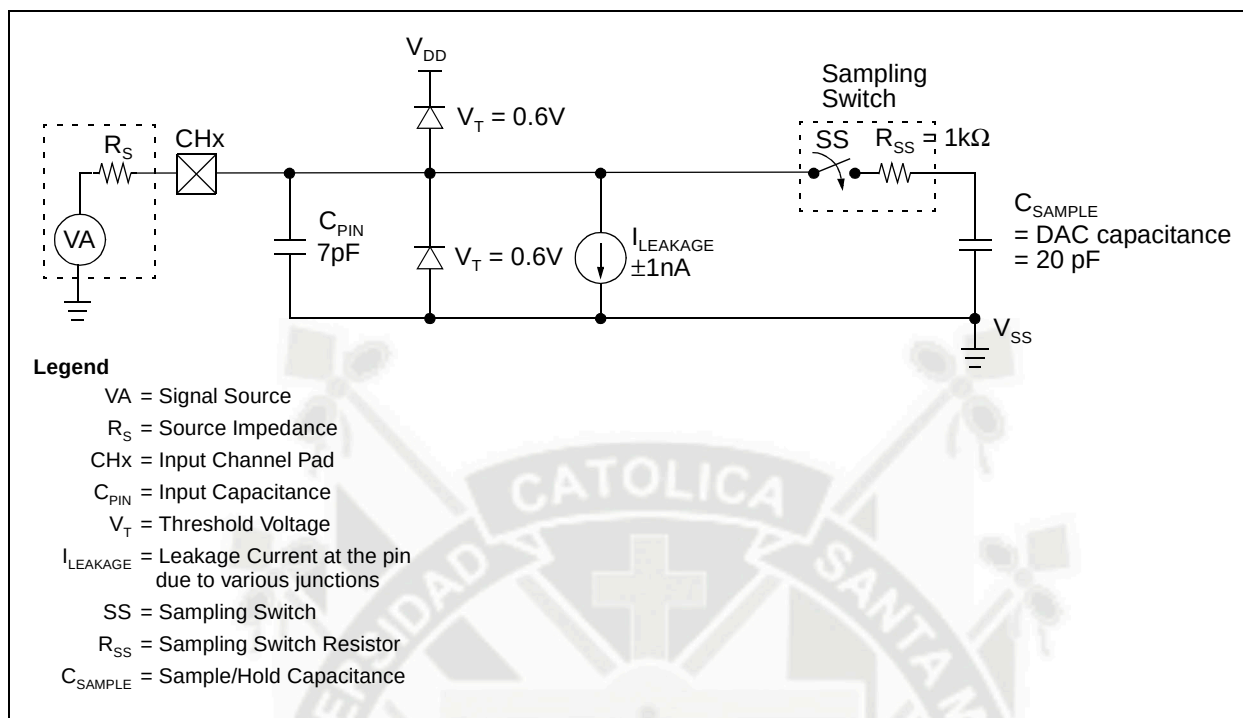


FIGURE 4-1: Analog Input Model.

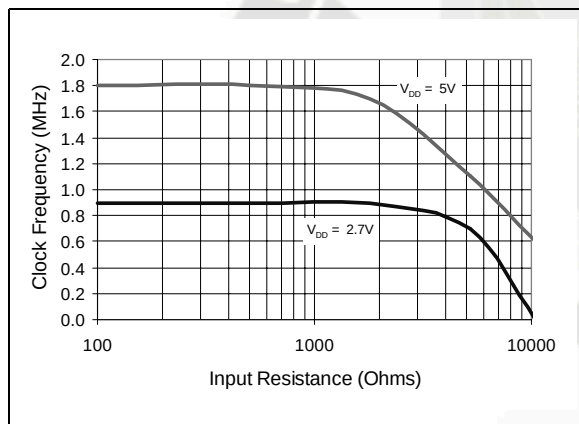


FIGURE 4-2: Maximum Clock Frequency vs. Input Resistance (R_S) to maintain less than a 0.1 LSB deviation in INL from nominal conditions.

5.0 SERIAL COMMUNICATIONS

5.1 Overview

Communication with the MCP3202 is done using a standard SPI-compatible serial interface. Initiating communication with the device is done by bringing the $\overline{CS}$ line low. See Figure 5-1. If the device was powered up with the $\overline{CS}$ pin low, it must be brought high and back low to initiate communication. The first clock received with $\overline{CS}$ low and D_{IN} high will constitute a start bit. The SGL/DIFF bit and the ODD/SIGN bit follow the start bit and are used to select the input channel configuration. The SGL/DIFF is used to select single ended or pseudo-differential mode. The ODD/SIGN bit selects which channel is used in single ended mode, and is used to determine polarity in pseudo-differential mode. Following the ODD/SIGN bit, the MSBF bit is transmitted to and is used to enable the LSB first format for the device. If the MSBF bit is low, then the data will come from the device in MSB first format and any further clocks with $\overline{CS}$ low will cause the device to output zeros. If the MSBF bit is high, then the device will output the converted word LSB first *after* the word has been transmitted in the MSB first format. See Figure 5-2. Table 5-1 shows the configuration bits for the MCP3202. The device will begin to sample the analog input on the second rising edge of the clock, after the start bit has been received. The sample period will end on the falling edge of the third clock following the start bit.

On the falling edge of the clock for the MSBF bit, the device will output a low null bit. The next sequential 12 clocks will output the result of the conversion with

MSB first as shown in Figure 5-1. Data is always output from the device on the falling edge of the clock. If all 12 data bits have been transmitted and the device continues to receive clocks while the $\overline{CS}$ is held low, (and MSBF = 1), the device will output the conversion result LSB first as shown in Figure 5-2. If more clocks are provided to the device while $\overline{CS}$ is still low (after the LSB first data has been transmitted), the device will clock out zeros indefinitely.

If necessary, it is possible to bring $\overline{CS}$ low and clock in leading zeros on the D_{IN} line before the start bit. This is often done when dealing with microcontroller-based SPI ports that must send 8 bits at a time. Refer to Section 6.1 for more details on using the MCP3202 devices with hardware SPI ports.

	CONFIG BITS		CHANNEL SELECTION		GND
	SGL/DIFF	ODD/SIGN	0	1	
SINGLE ENDED MODE	1	0	+		-
	1	1		+	-
PSEUDO-DIFFERENTIAL MODE	0	0	IN+	IN-	
	0	1	IN-	IN+	

TABLE 5-1: Configuration Bits for the MCP3202.

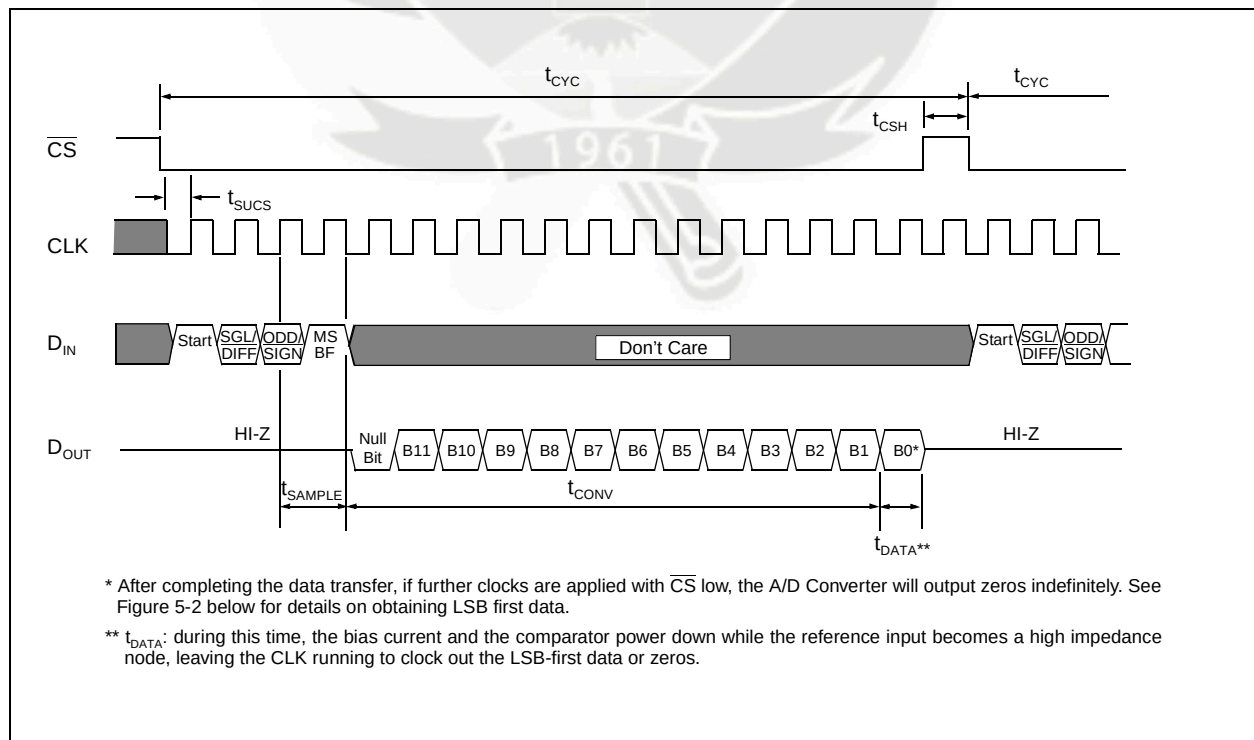


FIGURE 5-1: Communication with the MCP3202 using MSB first format only.

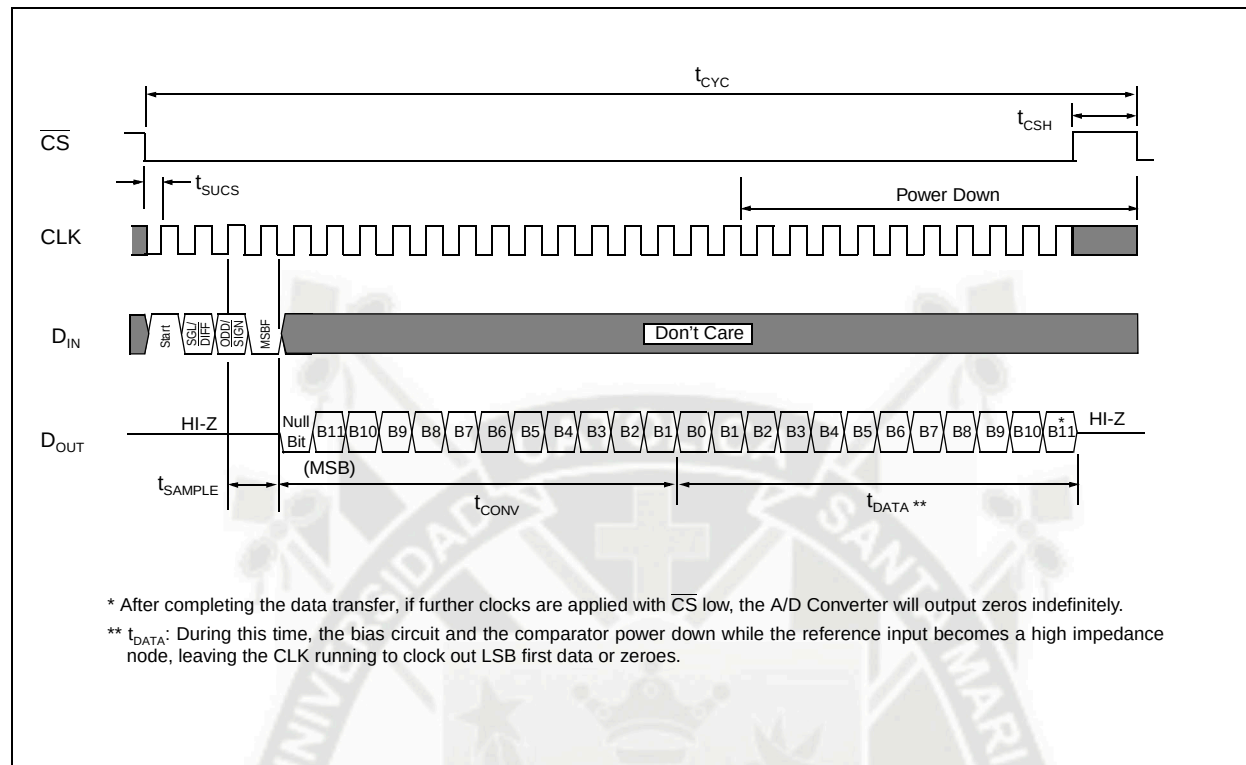


FIGURE 5-2: Communication with MCP3202 using LSB first format.

6.0 APPLICATIONS INFORMATION

6.1 Using the MCP3202 with Microcontroller (MCU) SPI Ports

With most microcontroller SPI ports, it is required to send groups of eight bits. It is also required that the microcontroller SPI port be configured to clock out data on the falling edge of clock and latch data in on the rising edge. Depending on how communication routines are used, it is very possible that the number of clocks required for communication will not be a multiple of eight. Therefore, it may be necessary for the MCU to send more clocks than are actually required. This is usually done by sending 'leading zeros' before the start bit, which are ignored by the device. As an example, Figure 6-1 and Figure 6-2 show how the MCP3202 can be interfaced to a MCU with a hardware SPI port. Figure 6-1 depicts the operation shown in SPI Mode 0,0,

which requires that the SCLK from the MCU idles in the 'low' state, while Figure 6-2 shows the similar case of SPI Mode 1,1 where the clock idles in the 'high' state.

As shown in Figure 6-1, the first byte transmitted to the A/D Converter contains seven leading zeros before the start bit. Arranging the leading zeros this way produces the output 12 bits to fall in positions easily manipulated by the MCU. The MSB is clocked out of the A/D Converter on the falling edge of clock number 12. After the second eight clocks have been sent to the device, the MCU receive buffer will contain three unknown bits (the output is at high impedance until the null bit is clocked out), the null bit and the highest order four bits of the conversion. After the third byte has been sent to the device, the receive register will contain the lowest order eight bits of the conversion results. Easier manipulation of the converted data can be obtained by using this method.

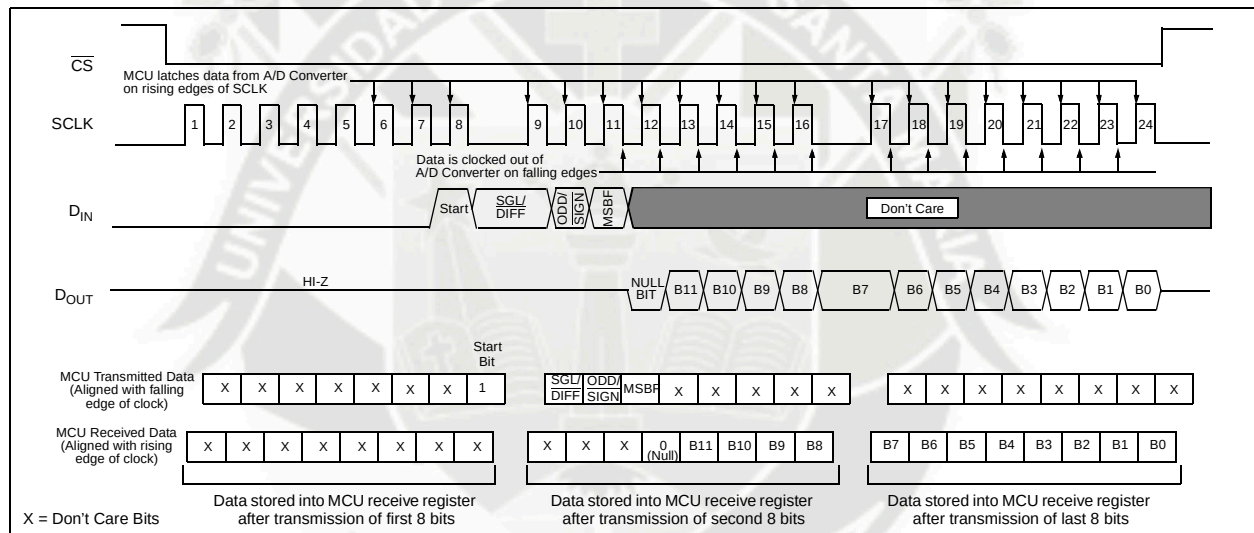


FIGURE 6-1: SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).

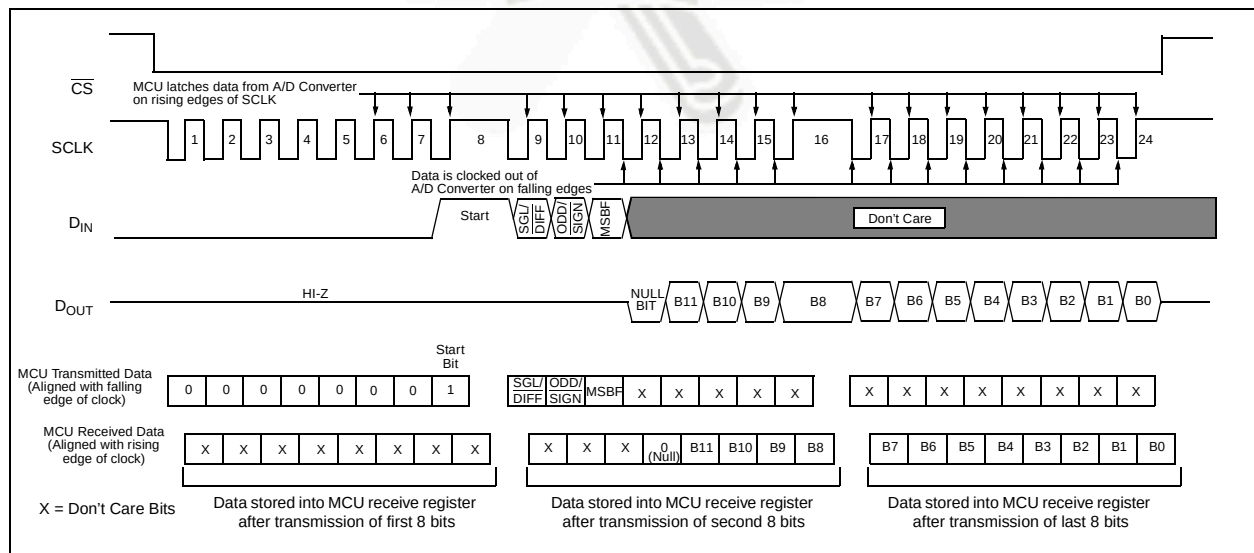


FIGURE 6-2: SPI Communication using 8-bit segments (Mode 1,1: SCLK idles high).

6.2 Maintaining Minimum Clock Speed

When the MCP3202 initiates the sample period, charge is stored on the sample capacitor. When the sample period is complete, the device converts one bit for each clock that is received. It is important for the user to note that a slow clock rate will allow charge to bleed off the sample cap while the conversion is taking place. At 85°C (worst case condition), the part will maintain proper charge on the sample capacitor for at least 1.2ms after the sample period has ended. This means that the time between the end of the sample period and the time that all 12 data bits have been clocked out must not exceed 1.2ms (effective clock frequency of 10kHz). Failure to meet this criteria may induce linearity errors into the conversion outside the rated specifications. It should be noted that during the entire conversion cycle, the A/D Converter does not require a constant clock speed or duty cycle, as long as all timing specifications are met.

6.3 Buffering/Filtering the Analog Inputs

If the signal source for the A/D Converter is not a low impedance source, it will have to be buffered or inaccurate conversion results may occur. It is also recommended that a filter be used to eliminate any signals that may be aliased back into the conversion results. This is illustrated in Figure 6-3 below where an op amp is used to drive the analog input of the MCP3202. This amplifier provides a low impedance output for the converter input and a low pass filter, which eliminates unwanted high frequency noise.

Low pass (anti-aliasing) filters can be designed using Microchip's interactive **FilterLab™** software. **FilterLab** will calculate capacitor and resistor values, as well as, determine the number of poles that are required for the application. For more information on filtering signals, see the application note AN699 "Anti-Aliasing Analog Filters for Data Acquisition Systems."

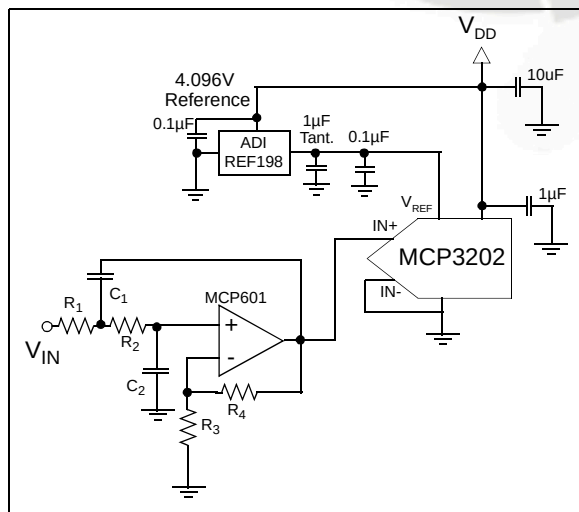


FIGURE 6-3: The MCP601 Operational Amplifier is used to implement a 2nd order anti-aliasing filter for the signal being converted by the MCP3202.

6.4 Layout Considerations

When laying out a printed circuit board for use with analog components, care should be taken to reduce noise wherever possible. A bypass capacitor should always be used with this device and should be placed as close as possible to the device pin. A bypass capacitor value of 1μF is recommended.

Digital and analog traces should be separated as much as possible on the board and no traces should run underneath the device or the bypass capacitor. Extra precautions should be taken to keep traces with high frequency signals (such as clock lines) as far as possible from analog traces.

Use of an analog ground plane is recommended in order to keep the ground potential the same for all devices on the board. Providing VDD connections to devices in a "star" configuration can also reduce noise by eliminating current return paths and associated errors. See Figure 6-4. For more information on layout tips when using A/D Converters, refer to AN688 "Layout Tips for 12-Bit A/D Converter Applications".

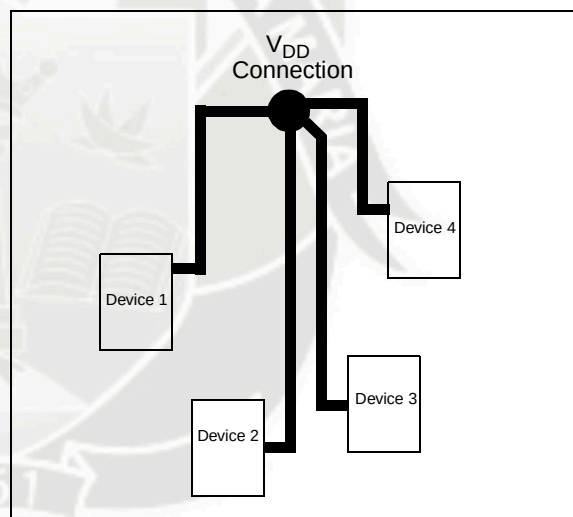
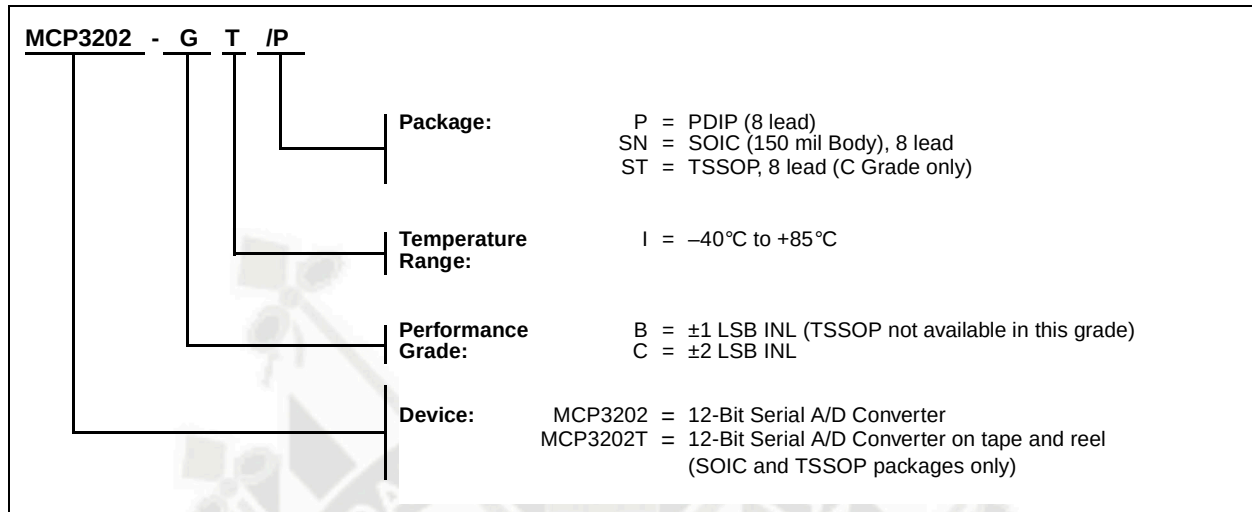


FIGURE 6-4: VDD traces arranged in a 'Star' configuration in order to reduce errors caused by current return paths.

FilterLab is a trademark of Microchip Technology Inc. in the U.S.A and other countries. All rights reserved.

MCP3202 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Sales and Support

Data Sheets

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (602) 786-7277. After September 1, 1999 (480) 786-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

NOTES:



NOTES:





WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-786-7200 Fax: 480-786-7277
Technical Support: 480-786-7627
Web Address: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
4570 Westgrove Drive, Suite 160
Addison, TX 75248
Tel: 972-818-7423 Fax: 972-818-2924

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit

Microchip Technology Inc.
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

AMERICAS (continued)

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
Unit 2101, Tower 2
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

Beijing

Microchip Technology, Beijing
Unit 915, 6 Chaoyangmen Bei Dajie
Dong Erhuan Road, Dongcheng District
New China Hong Kong Manhattan Building
Beijing 100027 PRC
Tel: 86-10-85282100 Fax: 86-10-85282104

India

Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Japan

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222-0033 Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700 Fax: 86 21-6275-5060

ASIA/PACIFIC (continued)

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5858 Fax: 44-118 921-5835

Denmark

Microchip Technology Denmark ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

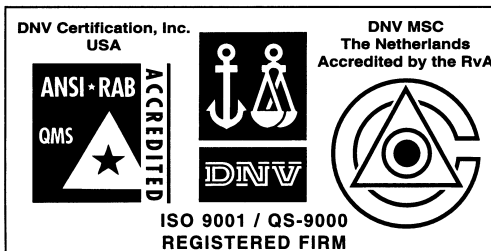
Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

11/15/99



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoc® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

All rights reserved. © 1999 Microchip Technology Incorporated. Printed in the USA. 11/99 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.

DM74LS240 • DM74LS241

Octal 3-STATE Buffer/Line Driver/Line Receiver

General Description

These buffers/line drivers are designed to improve both the performance and PC board density of 3-STATE buffers/drivers employed as memory-address drivers, clock drivers, and bus-oriented transmitters/receivers. Featuring 400 mV of hysteresis at each low current PNP data line input, they provide improved noise rejection and high fanout outputs and can be used to drive terminated lines down to 133Ω.

Features

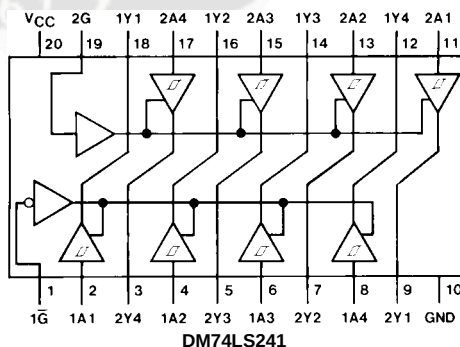
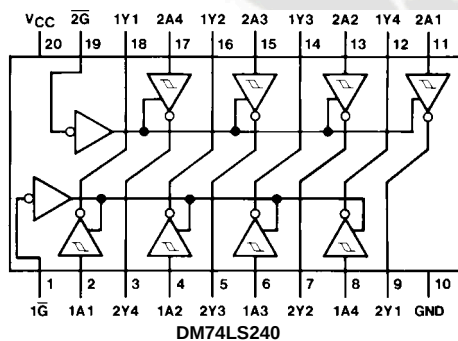
- 3-STATE outputs drive bus lines directly
- PNP inputs reduce DC loading on bus lines
- Hysteresis at data inputs improves noise margins
- Typical I_{OL} (sink current)
 - 24 mA
- Typical I_{OH} (source current)
 - 15 mA
- Typical propagation delay times
 - Inverting 10.5 ns
 - Noninverting 12 ns
- Typical enable/disable time 18 ns
- Typical power dissipation (enabled)
 - Inverting 130 mW
 - Noninverting 135 mW

Ordering Code:

Order Number	Package Number	Package Description
DM74LS240WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS240SJ	M20D	20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS240N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
DM74LS241WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
DM74LS241N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagrams



Function Tables

DM74LS240

Inputs		Output
$\bar{G}$	A	Y
L	L	H
L	H	L
H	X	Z

DM74LS241

Inputs				Outputs	
G	$\bar{G}$	1A	2A	1Y	2Y
X	L	L	X	L	
X	L	H	X	H	
X	H	X	X	Z	
H	X	X	L		L
H	X	X	H		H
L	X	X	X		Z

L = LOW Logic Level
 H = HIGH Logic Level
 X = Either LOW or HIGH Logic Level
 Z = High Impedance

Absolute Maximum Ratings(Note 1)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	–65°C to +150°C

Note 1: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			–15	mA
I_{OL}	LOW Level Output Current			24	mA
T_A	Free Air Operating Temperature	0		70	°C

Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
V_I	Input Clamp Voltage	$V_{CC} = \text{Min}, I_I = -18 \text{ mA}$			–1.5	V
HYS	Hysteresis ($V_{T+} - V_{T-}$) Data Inputs Only	$V_{CC} = \text{Min}$	0.2	0.4		V
V_{OH}	HIGH Level Output Voltage	$V_{CC} = \text{Min}, V_{IH} = \text{Min}$ $V_{IL} = \text{Max}, I_{OH} = -1 \text{ mA}$	2.7			V
		$V_{CC} = \text{Min}, V_{IH} = \text{Min}$ $V_{IL} = \text{Max}, I_{OH} = -3 \text{ mA}$	2.4	3.4		
		$V_{CC} = \text{Min}, V_{IH} = \text{Min}$ $V_{IL} = 0.5V, I_{OH} = \text{Max}$	2			
V_{OL}	LOW Level Output Voltage	$V_{CC} = \text{Min}$ $I_{OL} = 12 \text{ mA}$			0.4	V
		$V_{IL} = \text{Max}$ $I_{OL} = \text{Max}$			0.5	
		$V_{IH} = \text{Min}$				
I_{OZH}	Off-State Output Current, HIGH Level Voltage Applied	$V_{CC} = \text{Max}$ $V_{IL} = \text{Max}$	$V_O = 2.7V$		20	μA
I_{OZL}	Off-State Output Current, LOW Level Voltage Applied	$V_{IH} = \text{Min}$	$V_O = 0.4V$		–20	μA
I_I	Input Current at Maximum Input Voltage	$V_{CC} = \text{Max}$ $V_I = 7V$			0.1	mA
I_{IH}	HIGH Level Input Current	$V_{CC} = \text{Max}, V_I = 2.7V$			20	μA
I_{IL}	LOW Level Input Current	$V_{CC} = \text{Max}, V_I = 0.4V$			–0.2	mA
I_{OS}	Short Circuit Output Current	$V_{CC} = \text{Max}$ (Note 3)	–40		–225	mA
I_{CC}	Supply Current	$V_{CC} = \text{Max},$ Outputs OPEN	Outputs HIGH	13	23	mA
			Outputs LOW	26	44	
				27	46	
			Outputs Disabled	29	50	
				32	54	

Note 2: All typicals are at $V_{CC} = 5V, T_A = 25^\circ\text{C}$.

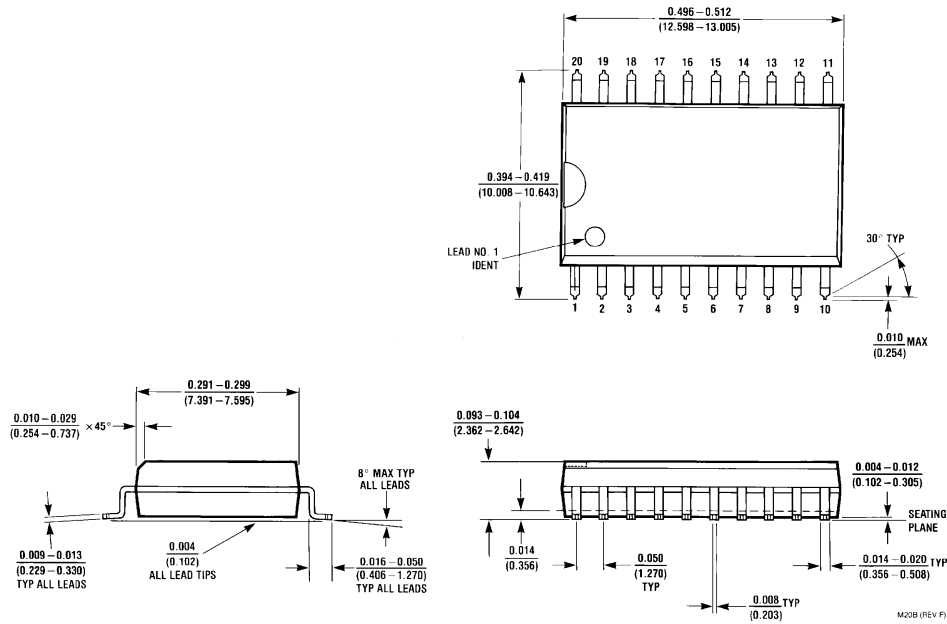
Note 3: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics

at $V_{CC} = 5V$ and $T_A = 25^\circ C$

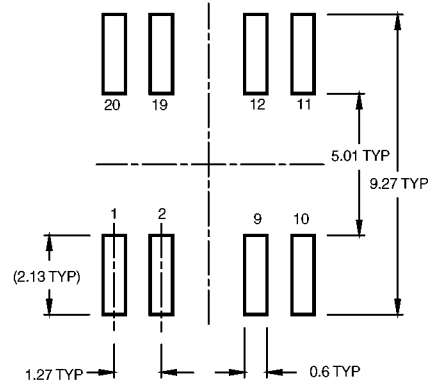
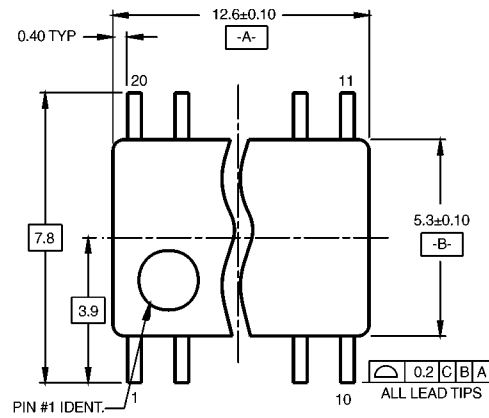
Symbol	Parameter	Conditions		Max	Units
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	$C_L = 45 \text{ pF}$	DM74LS240	14	ns
		$R_L = 667\Omega$	DM74LS241	18	
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	$C_L = 45 \text{ pF}$	DM74LS240	18	ns
		$R_L = 667\Omega$	DM74LS241	18	
t_{PZL}	Output Enable Time to LOW Level	$C_L = 45 \text{ pF}$	DM74LS240	30	ns
		$R_L = 667\Omega$	DM74LS241	30	
t_{PZH}	Output Enable Time to HIGH Level	$C_L = 45 \text{ pF}$	DM74LS240	23	ns
		$R_L = 667\Omega$	DM74LS241	23	
t_{PLZ}	Output Disable Time from LOW Level	$C_L = 5 \text{ pF}$	DM74LS240	25	ns
		$R_L = 667\Omega$	DM74LS241	25	
t_{PHZ}	Output Disable Time from HIGH Level	$C_L = 5 \text{ pF}$	DM74LS240	18	ns
		$R_L = 667\Omega$	DM74LS241	18	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	$C_L = 150 \text{ pF}$	DM74LS240	18	ns
		$R_L = 667\Omega$	DM74LS241	21	
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	$C_L = 150 \text{ pF}$	DM74LS240	22	ns
		$R_L = 667\Omega$	DM74LS241	22	
t_{PZL}	Output Enable Time to LOW Level	$C_L = 150 \text{ pF}$	DM74LS240	33	ns
		$R_L = 667\Omega$	DM74LS241	33	
t_{PZH}	Output Enable Time to HIGH Level	$C_L = 150 \text{ pF}$	DM74LS240	26	ns
		$R_L = 667\Omega$	DM74LS241	26	

Physical Dimensions inches (millimeters) unless otherwise noted

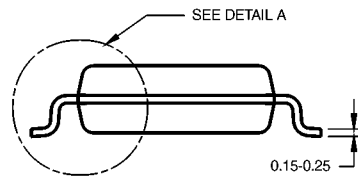
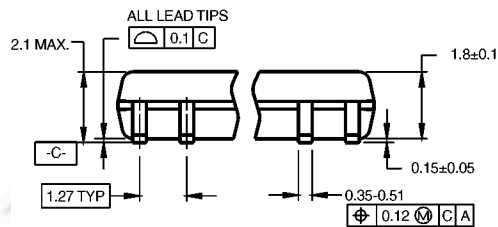


**20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300 Wide
Package Number M20B**

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



LAND PATTERN RECOMMENDATION

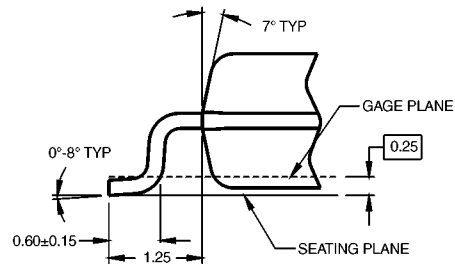


DIMENSIONS ARE IN MILLIMETERS

NOTES:

- A. CONFORMS TO EIAJ EDR-7320 REGISTRATION, ESTABLISHED IN DECEMBER, 1998.
- B. DIMENSIONS ARE IN MILLIMETERS.
- C. DIMENSIONS ARE EXCLUSIVE OF BURRS, MOLD FLASH, AND TIE BAR EXTRUSIONS.

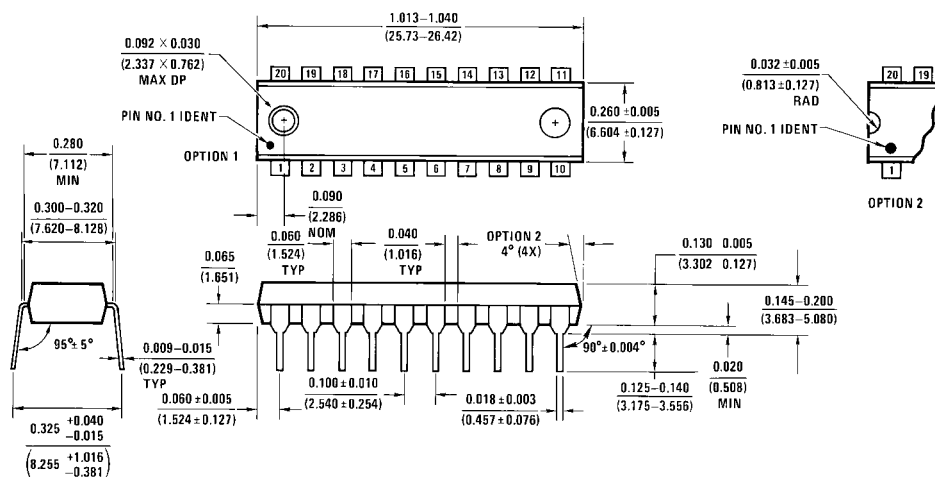
M20DRevB1



DETAIL A

20-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
Package Description M20D

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
Package Number N20A

Fairchild does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and Fairchild reserves the right at any time without notice to change said circuitry and specifications.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

www.fairchildsemi.com



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

General Description

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 are low-power transceivers for RS-485 and RS-422 communication. Each part contains one driver and one receiver. The MAX483, MAX487, MAX488, and MAX489 feature reduced slew-rate drivers that minimize EMI and reduce reflections caused by improperly terminated cables, thus allowing error-free data transmission up to 250kbps. The driver slew rates of the MAX481, MAX485, MAX490, MAX491, and MAX1487 are not limited, allowing them to transmit up to 2.5Mbps.

These transceivers draw between 120µA and 500µA of supply current when unloaded or fully loaded with disabled drivers. Additionally, the MAX481, MAX483, and MAX487 have a low-current shutdown mode in which they consume only 0.1µA. All parts operate from a single 5V supply.

Drivers are short-circuit current limited and are protected against excessive power dissipation by thermal shutdown circuitry that places the driver outputs into a high-impedance state. The receiver input has a fail-safe feature that guarantees a logic-high output if the input is open circuit.

The MAX487 and MAX1487 feature quarter-unit-load receiver input impedance, allowing up to 128 MAX487/MAX1487 transceivers on the bus. Full-duplex communications are obtained using the MAX488–MAX491, while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are designed for half-duplex applications.

Applications

Low-Power RS-485 Transceivers
Low-Power RS-422 Transceivers
Level Translators
Transceivers for EMI-Sensitive Applications
Industrial-Control Local Area Networks

Next Generation Device Features

- ◆ For Fault-Tolerant Applications
MAX3430: ±80V Fault-Protected, Fail-Safe, 1/4 Unit Load, +3.3V, RS-485 Transceiver
MAX3440E–MAX3444E: ±15kV ESD-Protected, ±60V Fault-Protected, 10Mbps, Fail-Safe, RS-485/J1708 Transceivers
- ◆ For Space-Constrained Applications
MAX3460–MAX3464: +5V, Fail-Safe, 20Mbps, Profibus RS-485/RS-422 Transceivers
MAX3362: +3.3V, High-Speed, RS-485/RS-422 Transceiver in a SOT23 Package
MAX3280E–MAX3284E: ±15kV ESD-Protected, 52Mbps, +3V to +5.5V, SOT23, RS-485/RS-422, True Fail-Safe Receivers
MAX3293/MAX3294/MAX3295: 20Mbps, +3.3V, SOT23, RS-855/RS-422 Transmitters
- ◆ For Multiple Transceiver Applications
MAX3030E–MAX3033E: ±15kV ESD-Protected, +3.3V, Quad RS-422 Transmitters
- ◆ For Fail-Safe Applications
MAX3080–MAX3089: Fail-Safe, High-Speed (10Mbps), Slew-Rate-Limited RS-485/RS-422 Transceivers
- ◆ For Low-Voltage Applications
MAX3483E/MAX3485E/MAX3486E/MAX3488E/MAX3490E/MAX3491E: +3.3V Powered, ±15kV ESD-Protected, 12Mbps, Slew-Rate-Limited, True RS-485/RS-422 Transceivers

Ordering Information appears at end of data sheet.

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF TRANSMITTERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC}) 12V
 Control Input Voltage ($\overline{RE}$, DE) -0.5V to ($V_{CC} + 0.5$ V)
 Driver Input Voltage (DI) -0.5V to ($V_{CC} + 0.5$ V)
 Driver Output Voltage (A, B) -8V to +12.5V
 Receiver Input Voltage (A, B) -8V to +12.5V
 Receiver Output Voltage (RO) -0.5V to ($V_{CC} + 0.5$ V)
 Continuous Power Dissipation ($T_A = +70^\circ\text{C}$)
 8-Pin Plastic DIP (derate 9.09mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 727mW
 14-Pin Plastic DIP (derate 10.00mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 800mW
 8-Pin SO (derate 5.88mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 471mW

14-Pin SO (derate 8.33mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 667mW
 8-Pin μMAX (derate 4.1mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 830mW
 8-Pin Cerdip (derate 8.00mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 640mW
 14-Pin Cerdip (derate 9.09mW/ $^\circ\text{C}$ above $+70^\circ\text{C}$) 727mW
 Operating Temperature Ranges
 MAX4_ _C_ /MAX1487C_ A 0°C to $+70^\circ\text{C}$
 MAX4_ _E_ /MAX1487E_ A -40°C to $+85^\circ\text{C}$
 MAX4_ _MJ_ /MAX1487MJA -55°C to $+125^\circ\text{C}$
 Storage Temperature Range -65°C to $+160^\circ\text{C}$
 Lead Temperature (soldering, 10sec) $+300^\circ\text{C}$

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL CHARACTERISTICS

($V_{CC} = 5V \pm 5\%$, $T_A = T_{MIN}$ to T_{MAX} , unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Differential Driver Output (no load)	V_{OD1}				5	V
Differential Driver Output (with load)	V_{OD2}	$R = 50\Omega$ (RS-422)	2			V
		$R = 27\Omega$ (RS-485), Figure 4	1.5		5	
Change in Magnitude of Driver Differential Output Voltage for Complementary Output States	ΔV_{OD}	$R = 27\Omega$ or 50Ω , Figure 4			0.2	V
Driver Common-Mode Output Voltage	V_{OC}	$R = 27\Omega$ or 50Ω , Figure 4			3	V
Change in Magnitude of Driver Common-Mode Output Voltage for Complementary Output States	ΔV_{OD}	$R = 27\Omega$ or 50Ω , Figure 4			0.2	V
Input High Voltage	V_{IH}	DE, DI, $\overline{RE}$	2.0			V
Input Low Voltage	V_{IL}	DE, DI, $\overline{RE}$			0.8	V
Input Current	I_{IN1}	DE, DI, $\overline{RE}$			± 2	μA
Input Current (A, B)	I_{IN2}	DE = 0V; $V_{CC} = 0\text{V}$ or 5.25V , all devices except MAX487/MAX1487	$V_{IN} = 12\text{V}$		1.0	mA
			$V_{IN} = -7\text{V}$		-0.8	
		MAX487/MAX1487, DE = 0V, $V_{CC} = 0\text{V}$ or 5.25V	$V_{IN} = 12\text{V}$		0.25	mA
			$V_{IN} = -7\text{V}$		-0.2	
Receiver Differential Threshold Voltage	V_{TH}	$-7\text{V} \leq V_{CM} \leq 12\text{V}$	-0.2		0.2	V
Receiver Input Hysteresis	ΔV_{TH}	$V_{CM} = 0\text{V}$		70		mV
Receiver Output High Voltage	V_{OH}	$I_O = -4\text{mA}$, $V_{ID} = 200\text{mV}$	3.5			V
Receiver Output Low Voltage	V_{OL}	$I_O = 4\text{mA}$, $V_{ID} = -200\text{mV}$			0.4	V
Three-State (high impedance) Output Current at Receiver	I_{OZR}	$0.4\text{V} \leq V_O \leq 2.4\text{V}$			± 1	μA
Receiver Input Resistance	R_{IN}	$-7\text{V} \leq V_{CM} \leq 12\text{V}$, all devices except MAX487/MAX1487	12			$k\Omega$
		$-7\text{V} \leq V_{CM} \leq 12\text{V}$, MAX487/MAX1487	48			$k\Omega$

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

DC ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
No-Load Supply Current (Note 3)	I _{CC}	MAX488/MAX489, DE, DI, RE = 0V or V _{CC}		120	250	μA
		MAX490/MAX491, DE, DI, RE = 0V or V _{CC}		300	500	
		MAX481/MAX485, RE = 0V or V _{CC}	DE = V _{CC}	500	900	
			DE = 0V	300	500	
		MAX1487, RE = 0V or V _{CC}	DE = V _{CC}	300	500	
			DE = 0V	230	400	
		MAX483/MAX487, RE = 0V or V _{CC}	DE = 5V	MAX483 350	MAX487 650	
			DE = 0V	250	400	
Supply Current in Shutdown	I _{SHDN}	MAX481/483/487, DE = 0V, RE = V _{CC}		0.1	10	μA
Driver Short-Circuit Current, V _O = High	I _{OSD1}	-7V ≤ V _O ≤ 12V (Note 4)	35		250	mA
Driver Short-Circuit Current, V _O = Low	I _{OSD2}	-7V ≤ V _O ≤ 12V (Note 4)	35		250	mA
Receiver Short-Circuit Current	I _{QSR}	0V ≤ V _O ≤ V _{CC}	7		95	mA

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Driver Input to Output	tPLH	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF		10	30	60	ns
	tPHL			10	30	60	
Driver Output Skew to Output	tSKEW	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF			5	10	ns
Driver Rise or Fall Time	t _R , t _F	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF	MAX481, MAX485, MAX1487	3	15	40	ns
			MAX490C/E, MAX491C/E	5	15	25	
			MAX490M, MAX491M	3	15	40	
Driver Enable to Output High	tZH	Figures 7 and 9, C _L = 100pF, S2 closed			40	70	ns
Driver Enable to Output Low	tZL	Figures 7 and 9, C _L = 100pF, S1 closed			40	70	ns
Driver Disable Time from Low	tLZ	Figures 7 and 9, C _L = 15pF, S1 closed			40	70	ns
Driver Disable Time from High	tHZ	Figures 7 and 9, C _L = 15pF, S2 closed			40	70	ns
Receiver Input to Output	tPLH, tPHL	Figures 6 and 10, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF	MAX481, MAX485, MAX1487	20	90	200	ns
			MAX490C/E, MAX491C/E	20	90	150	
			MAX490M, MAX491M	20	90	200	
tPLH - tPHL Differential Receiver Skew	tSKD	Figures 6 and 10, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF			13		ns
Receiver Enable to Output Low	tZL	Figures 5 and 11, C _R L = 15pF, S1 closed			20	50	ns
Receiver Enable to Output High	tZH	Figures 5 and 11, C _R L = 15pF, S2 closed			20	50	ns
Receiver Disable Time from Low	tLZ	Figures 5 and 11, C _R L = 15pF, S1 closed			20	50	ns
Receiver Disable Time from High	tHZ	Figures 5 and 11, C _R L = 15pF, S2 closed			20	50	ns
Maximum Data Rate	f _{MAX}			2.5			Mbps
Time to Shutdown	tSHDN	MAX481 (Note 5)		50	200	600	ns

MAX481/MAX483/MAX485/MAX487-MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

SWITCHING CHARACTERISTICS—MAX481/MAX485, MAX490/MAX491, MAX1487 (continued)

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Enable from Shutdown to Output High (MAX481)	t _{ZH} (SHDN)	Figures 7 and 9, C _L = 100pF, S2 closed		40	100	ns
Driver Enable from Shutdown to Output Low (MAX481)	t _{ZL} (SHDN)	Figures 7 and 9, C _L = 100pF, S1 closed		40	100	ns
Receiver Enable from Shutdown to Output High (MAX481)	t _{ZH} (SHDN)	Figures 5 and 11, C _L = 15pF, S2 closed, A - B = 2V		300	1000	ns
Receiver Enable from Shutdown to Output Low (MAX481)	t _{ZL} (SHDN)	Figures 5 and 11, C _L = 15pF, S1 closed, B - A = 2V		300	1000	ns

SWITCHING CHARACTERISTICS—MAX483, MAX487/MAX488/MAX489

(V_{CC} = 5V ±5%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Driver Input to Output	t _{PLH}	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF	250	800	2000	ns
	t _{PHL}		250	800	2000	
Driver Output Skew to Output	t _{SKEW}	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF		100	800	ns
Driver Rise or Fall Time	t _R , t _F	Figures 6 and 8, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF	250		2000	ns
Driver Enable to Output High	t _{ZH}	Figures 7 and 9, C _L = 100pF, S2 closed	250		2000	ns
Driver Enable to Output Low	t _{ZL}	Figures 7 and 9, C _L = 100pF, S1 closed	250		2000	ns
Driver Disable Time from Low	t _{LZ}	Figures 7 and 9, C _L = 15pF, S1 closed	300		3000	ns
Driver Disable Time from High	t _{HZ}	Figures 7 and 9, C _L = 15pF, S2 closed	300		3000	ns
Receiver Input to Output	t _{PLH}	Figures 6 and 10, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF	250		2000	ns
	t _{PHL}		250		2000	
t _{PLH} - t _{PHL} Differential Receiver Skew	t _{SKD}	Figures 6 and 10, R _{DIFF} = 54Ω, C _{L1} = C _{L2} = 100pF		100		ns
Receiver Enable to Output Low	t _{ZL}	Figures 5 and 11, C _R _L = 15pF, S1 closed		20	50	ns
Receiver Enable to Output High	t _{ZH}	Figures 5 and 11, C _R _L = 15pF, S2 closed		20	50	ns
Receiver Disable Time from Low	t _{LZ}	Figures 5 and 11, C _R _L = 15pF, S1 closed		20	50	ns
Receiver Disable Time from High	t _{HZ}	Figures 5 and 11, C _R _L = 15pF, S2 closed		20	50	ns
Maximum Data Rate	f _{MAX}	t _{PLH} , t _{PHL} < 50% of data period	250			kbps
Time to Shutdown	t _{SHDN}	MAX483/MAX487 (Note 5)	50	200	600	ns
Driver Enable from Shutdown to Output High	t _{ZH} (SHDN)	MAX483/MAX487, Figures 7 and 9, C _L = 100pF, S2 closed			2000	ns
Driver Enable from Shutdown to Output Low	t _{ZL} (SHDN)	MAX483/MAX487, Figures 7 and 9, C _L = 100pF, S1 closed			2000	ns
Receiver Enable from Shutdown to Output High	t _{ZH} (SHDN)	MAX483/MAX487, Figures 5 and 11, C _L = 15pF, S2 closed			2500	ns
Receiver Enable from Shutdown to Output Low	t _{ZL} (SHDN)	MAX483/MAX487, Figures 5 and 11, C _L = 15pF, S1 closed			2500	ns

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

NOTES FOR ELECTRICAL/SWITCHING CHARACTERISTICS

Note 1: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 2: All typical specifications are given for $V_{CC} = 5V$ and $T_A = +25^\circ C$.

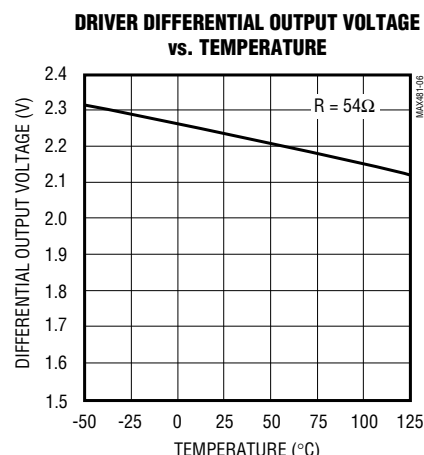
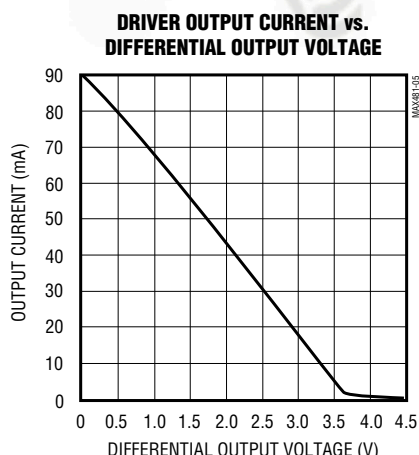
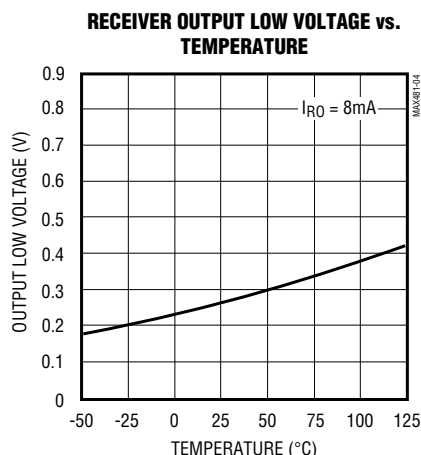
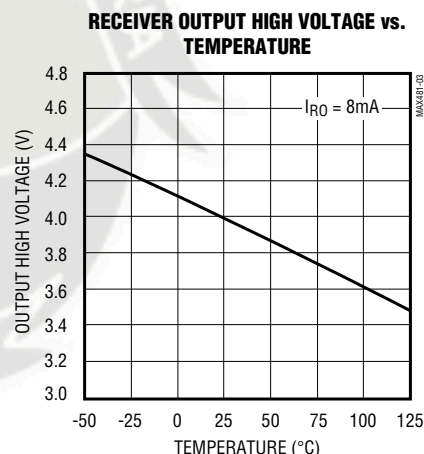
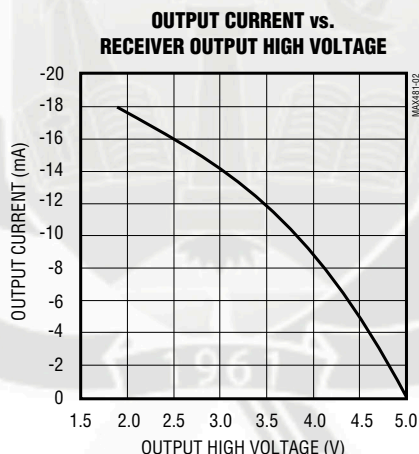
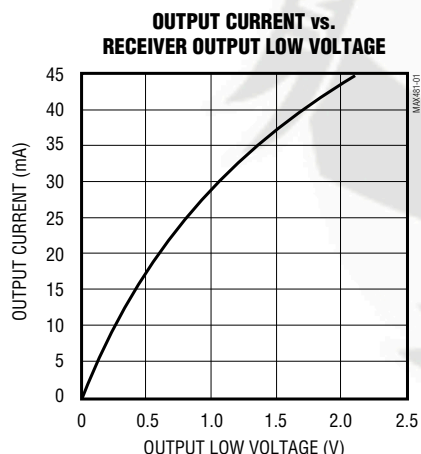
Note 3: Supply current specification is valid for loaded transmitters when $DE = 0V$.

Note 4: Applies to peak current. See *Typical Operating Characteristics*.

Note 5: The MAX481/MAX483/MAX487 are put into shutdown by bringing $\overline{RE}$ high and DE low. If the inputs are in this state for less than 50ns, the parts are guaranteed not to enter shutdown. If the inputs are in this state for at least 600ns, the parts are guaranteed to have entered shutdown. See *Low-Power Shutdown Mode* section.

Typical Operating Characteristics

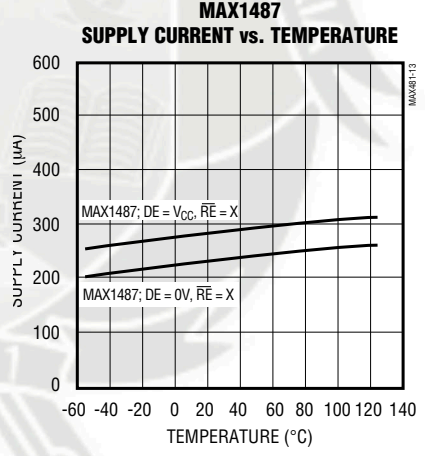
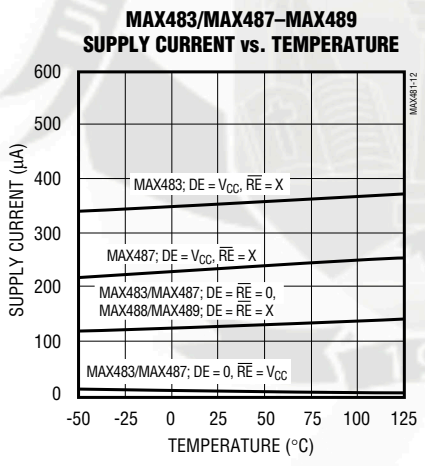
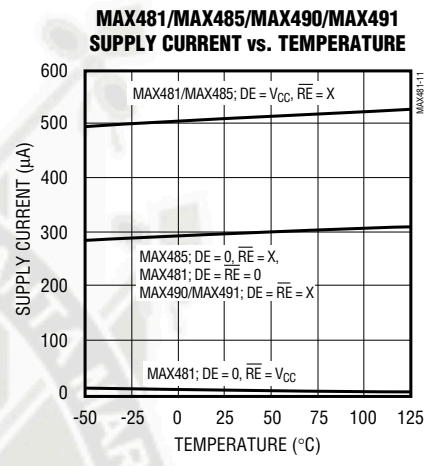
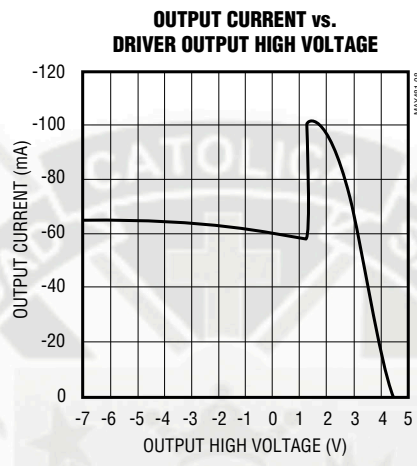
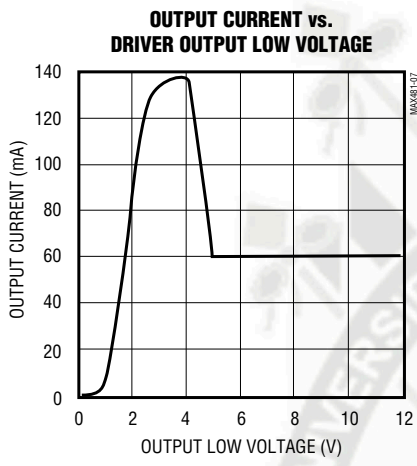
($V_{CC} = 5V$, $T_A = +25^\circ C$, unless otherwise noted.)



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Typical Operating Characteristics (continued)

(V_{CC} = 5V, T_A = +25°C, unless otherwise noted.)



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Pin Description

PIN					NAME	FUNCTION
MAX481/MAX483/ MAX485/MAX487/ MAX1487		MAX488/ MAX490		MAX489/ MAX491		
DIP/SO	μMAX	DIP/SO	μMAX	DIP/SO		
1	3	2	4	2	RO	Receiver Output: If A > B by 200mV, RO will be high; If A < B by 200mV, RO will be low.
2	4	—	—	3	$\overline{RE}$	Receiver Output Enable. RO is enabled when $\overline{RE}$ is low; RO is high impedance when $\overline{RE}$ is high.
3	5	—	—	4	DE	Driver Output Enable. The driver outputs, Y and Z, are enabled by bringing DE high. They are high impedance when DE is low. If the driver outputs are enabled, the parts function as line drivers. While they are high impedance, they function as line receivers if $\overline{RE}$ is low.
4	6	3	5	5	DI	Driver Input. A low on DI forces output Y low and output Z high. Similarly, a high on DI forces output Y high and output Z low.
5	7	4	6	6, 7	GND	Ground
—	—	5	7	9	Y	Noninverting Driver Output
—	—	6	8	10	Z	Inverting Driver Output
6	8	—	—	—	A	Noninverting Receiver Input and Noninverting Driver Output
—	—	8	2	12	A	Noninverting Receiver Input
7	1	—	—	—	B	Inverting Receiver Input and Inverting Driver Output
—	—	7	1	11	B	Inverting Receiver Input
8	2	1	3	14	VCC	Positive Supply: $4.75V \leq V_{CC} \leq 5.25V$
—	—	—	—	1, 8, 13	N.C.	No Connect—not internally connected

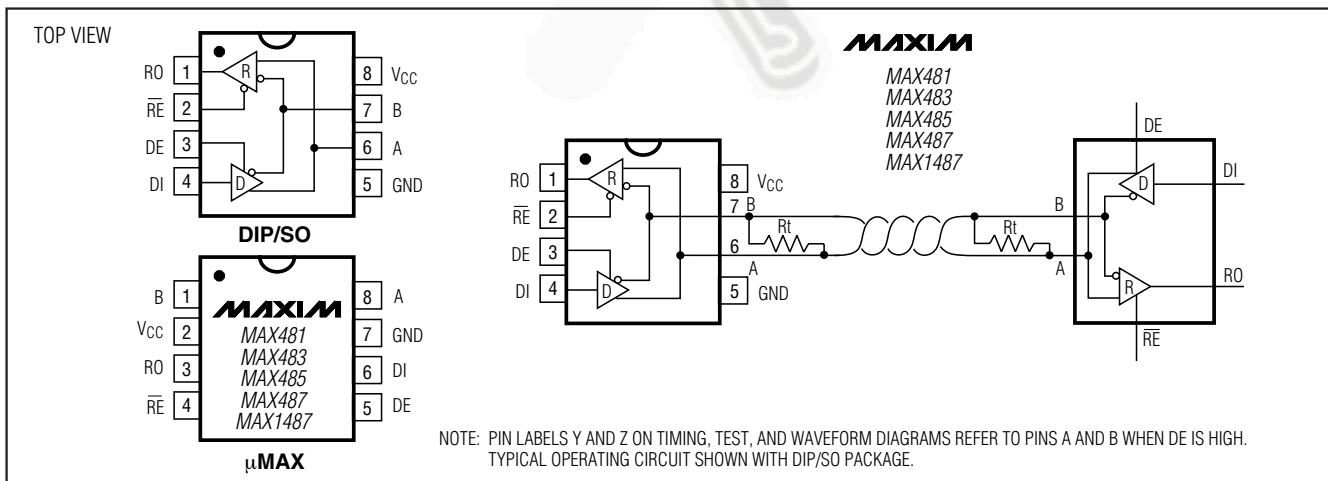


Figure 1. MAX481/MAX483/MAX485/MAX487/MAX1487 Pin Configuration and Typical Operating Circuit

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

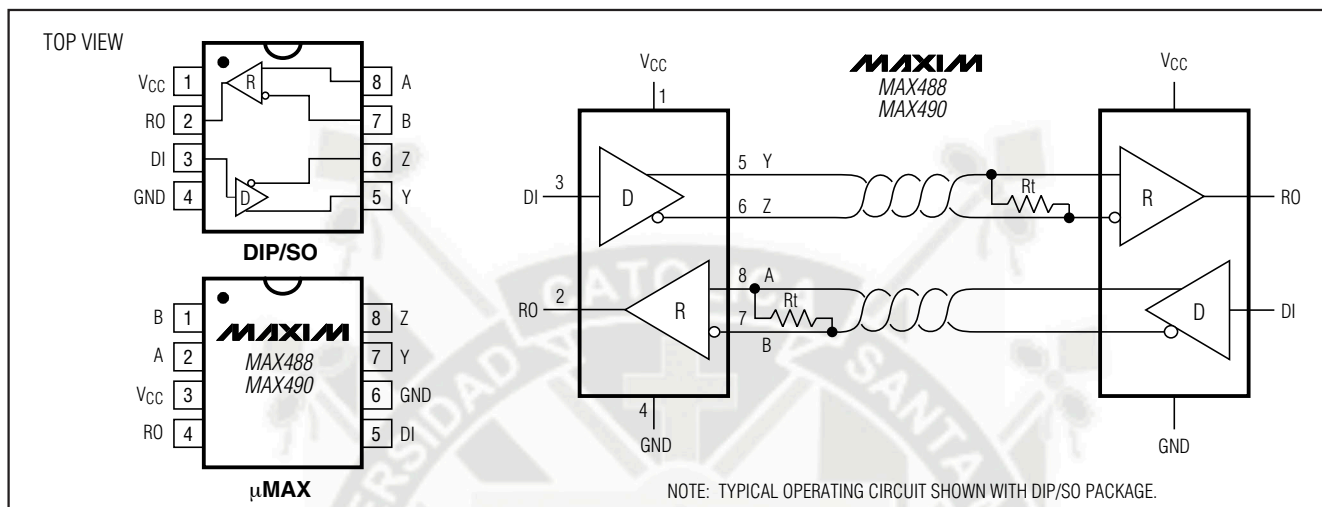


Figure 2. MAX488/MAX490 Pin Configuration and Typical Operating Circuit

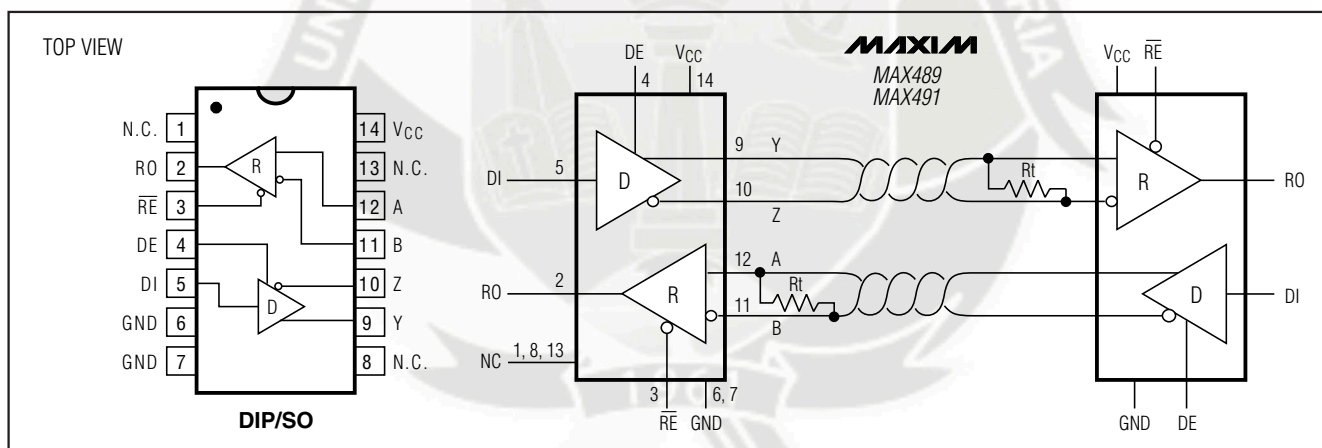


Figure 3. MAX489/MAX491 Pin Configuration and Typical Operating Circuit

Applications Information

The MAX481/MAX483/MAX485/MAX487-MAX491 and MAX1487 are low-power transceivers for RS-485 and RS-422 communications. The MAX481, MAX485, MAX490, MAX491, and MAX1487 can transmit and receive at data rates up to 2.5Mbps, while the MAX483, MAX487, MAX488, and MAX489 are specified for data rates up to 250kbps. The MAX488-MAX491 are full-duplex transceivers while the MAX481, MAX483, MAX485, MAX487, and MAX1487 are half-duplex. In addition, Driver Enable (DE) and Receiver Enable ($\overline{RE}$) pins are included on the MAX481, MAX483, MAX485, MAX487, MAX489, MAX491, and MAX1487. When disabled, the driver and receiver outputs are high impedance.

MAX487/MAX1487: 128 Transceivers on the Bus

The 48k Ω , 1/4-unit-load receiver input impedance of the MAX487 and MAX1487 allows up to 128 transceivers on a bus, compared to the 1-unit load (12k Ω input impedance) of standard RS-485 drivers (32 transceivers maximum). Any combination of MAX487/MAX1487 and other RS-485 transceivers with a total of 32 unit loads or less can be put on the bus. The MAX481/MAX483/MAX485 and MAX488-MAX491 have standard 12k Ω Receiver Input impedance.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Test Circuits

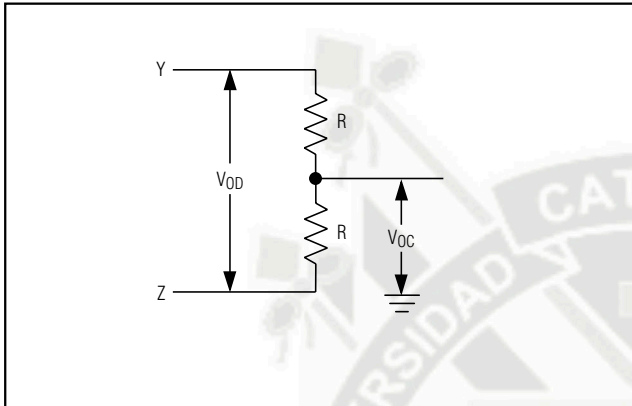


Figure 4. Driver DC Test Load

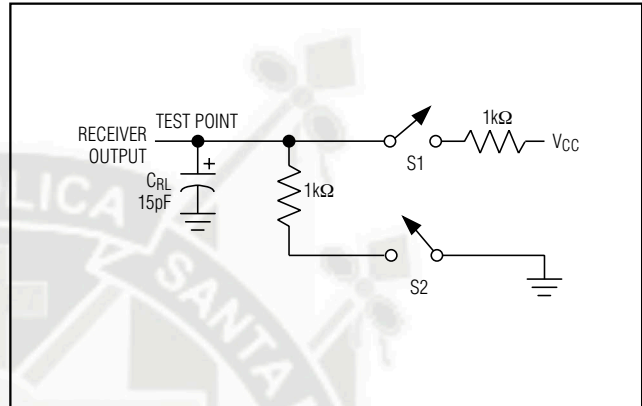


Figure 5. Receiver Timing Test Load

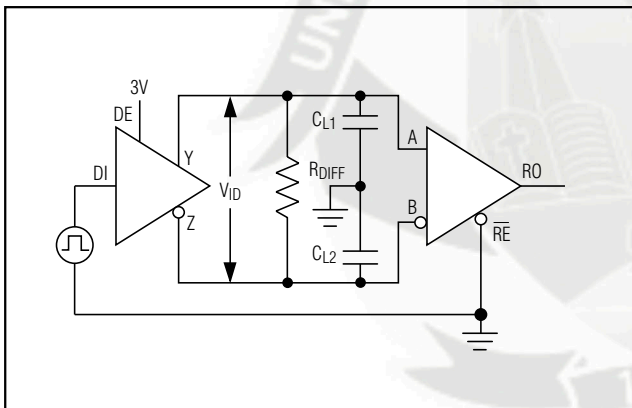


Figure 6. Driver/Receiver Timing Test Circuit

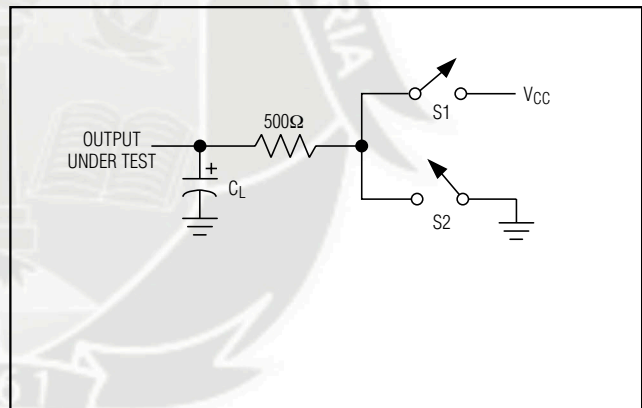


Figure 7. Driver Timing Test Load

MAX483/MAX487/MAX488/MAX489: Reduced EMI and Reflections

The MAX483 and MAX487–MAX489 are slew-rate limited, minimizing EMI and reducing reflections caused by improperly terminated cables. Figure 12 shows the driver output waveform and its Fourier analysis of a 150kHz signal transmitted by a MAX481, MAX485, MAX490, MAX491, or MAX1487. High-frequency har-

monics with large amplitudes are evident. Figure 13 shows the same information displayed for a MAX483, MAX487, MAX488, or MAX489 transmitting under the same conditions. Figure 13's high-frequency harmonics have much lower amplitudes, and the potential for EMI is significantly reduced.

MAX481/MAX483/MAX485/MAX487–MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Switching Waveforms

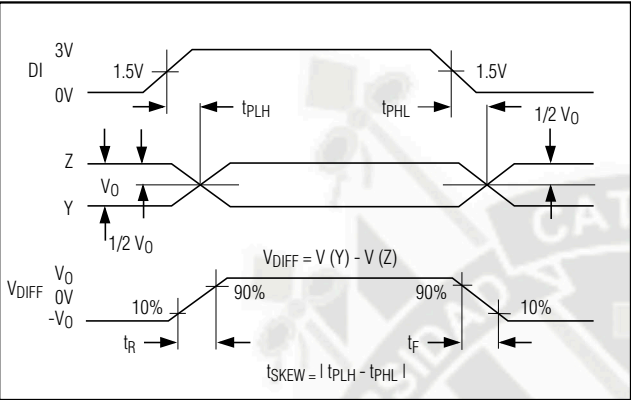


Figure 8. Driver Propagation Delays

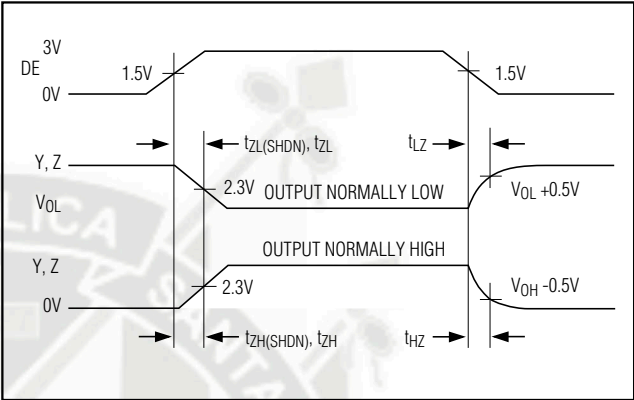


Figure 9. Driver Enable and Disable Times (except MAX488 and MAX490)

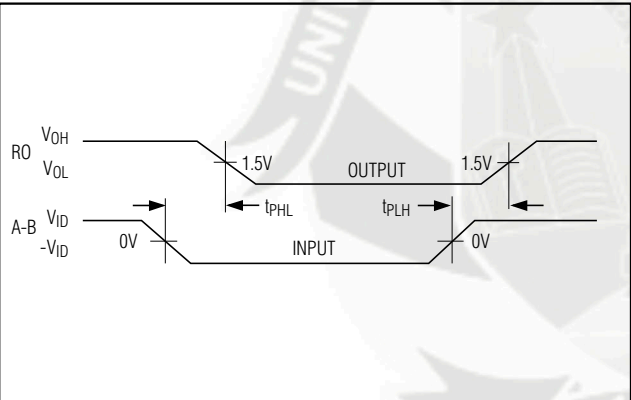


Figure 10. Receiver Propagation Delays

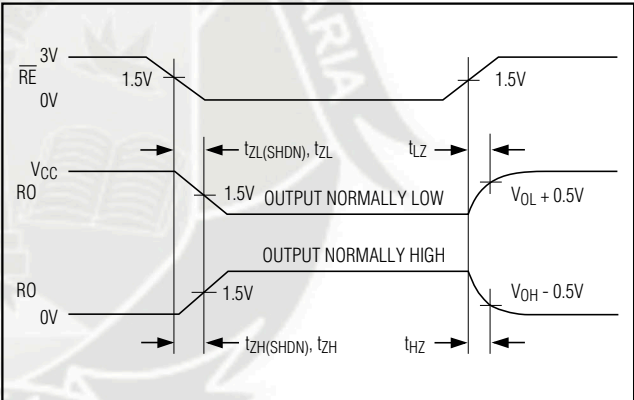


Figure 11. Receiver Enable and Disable Times (except MAX488 and MAX490)

Function Tables (MAX481/MAX483/MAX485/MAX487/MAX1487)

Table 1. Transmitting

INPUTS			OUTPUTS	
RE	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	High-Z	High-Z
1	0	X	High-Z*	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

Table 2. Receiving

INPUTS			OUTPUT
RE	DE	A-B	RO
0	0	$\geq +0.2V$	1
0	0	$\leq -0.2V$	0
0	0	Inputs open	1
1	0	X	High-Z*

X = Don't care
High-Z = High impedance
* Shutdown mode for MAX481/MAX483/MAX487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

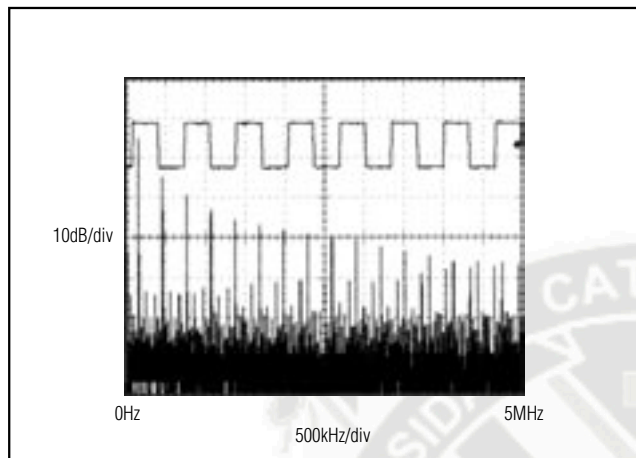


Figure 12. Driver Output Waveform and FFT Plot of MAX481/MAX485/MAX490/MAX491/MAX1487 Transmitting a 150kHz Signal

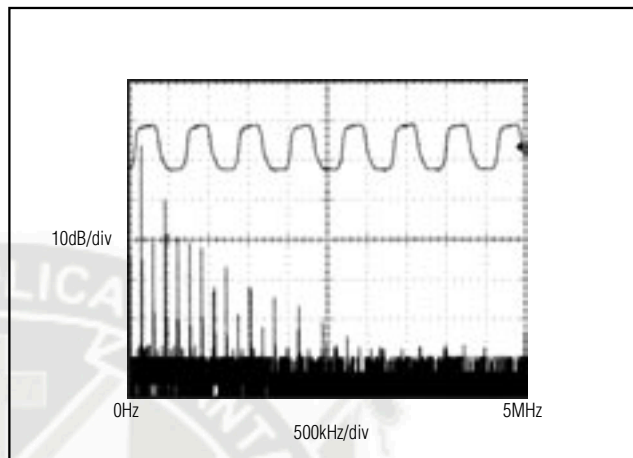


Figure 13. Driver Output Waveform and FFT Plot of MAX483/MAX487-MAX489 Transmitting a 150kHz Signal

Low-Power Shutdown Mode (MAX481/MAX483/MAX487)

A low-power shutdown mode is initiated by bringing both $\overline{\text{RE}}$ high and DE low. The devices will not shut down unless both the driver and receiver are disabled. In shutdown, the devices typically draw only 0.1μA of supply current.

$\overline{\text{RE}}$ and DE may be driven simultaneously; the parts are guaranteed not to enter shutdown if $\overline{\text{RE}}$ is high and DE is low for less than 50ns. If the inputs are in this state for at least 600ns, the parts are guaranteed to enter shutdown.

For the MAX481, MAX483, and MAX487, the t_{ZH} and t_{ZL} enable times assume the part was not in the low-power shutdown state (the MAX485/MAX488-MAX491 and MAX1487 can not be shut down). The t_{ZH}(SHDN) and t_{ZL}(SHDN) enable times assume the parts were shut down (see *Electrical Characteristics*).

It takes the drivers and receivers longer to become enabled from the low-power shutdown state (t_{ZH}(SHDN), t_{ZL}(SHDN)) than from the operating mode (t_{ZH}, t_{ZL}). (The parts are in operating mode if the $\overline{\text{RE}}$, DE inputs equal a logical 0, 1 or 1, 1 or 0, 0.)

Driver Output Protection

Excessive output current and power dissipation caused by faults or by bus contention are prevented by two mechanisms. A foldback current limit on the output stage provides immediate protection against short circuits over the whole common-mode voltage range (see *Typical Operating Characteristics*). In addition, a thermal shutdown circuit forces the driver outputs into a high-impedance state if the die temperature rises excessively.

Propagation Delay

Many digital encoding schemes depend on the difference between the driver and receiver propagation delay times. Typical propagation delays are shown in Figures 15–18 using Figure 14's test circuit.

The difference in receiver delay times, |t_{PLH} - t_{PHL}|, is typically under 13ns for the MAX481, MAX485, MAX490, MAX491, and MAX1487 and is typically less than 100ns for the MAX483 and MAX487-MAX489.

The driver skew times are typically 5ns (10ns max) for the MAX481, MAX485, MAX490, MAX491, and MAX1487, and are typically 100ns (800ns max) for the MAX483 and MAX487-MAX489.

MAX481/MAX483/MAX485/MAX487-MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

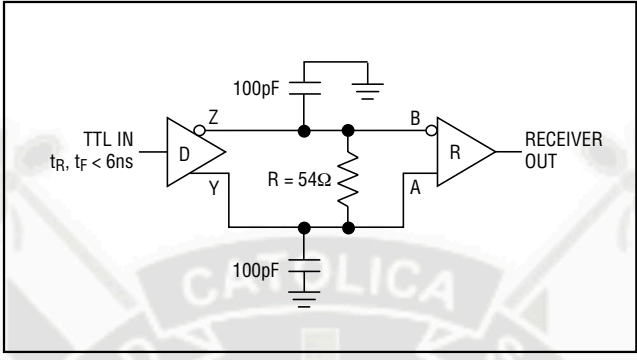


Figure 14. Receiver Propagation Delay Test Circuit

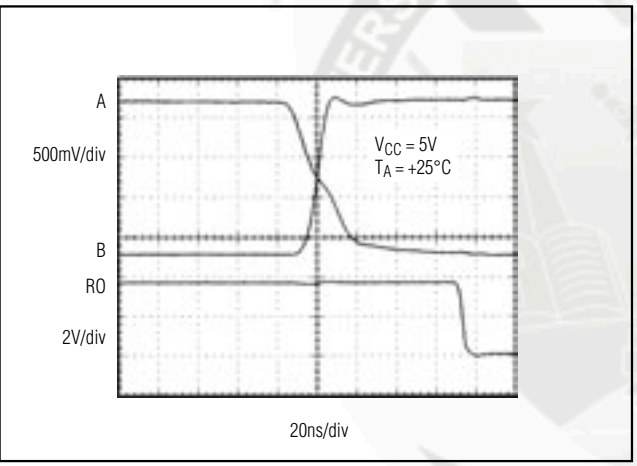


Figure 15. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver tPHL

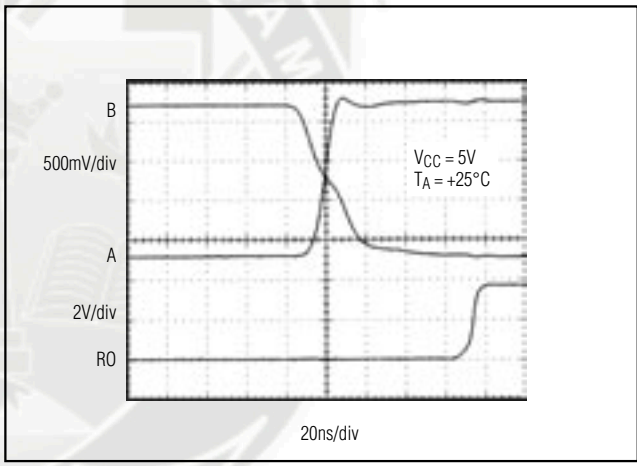


Figure 16. MAX481/MAX485/MAX490/MAX491/MAX1487 Receiver tPLH

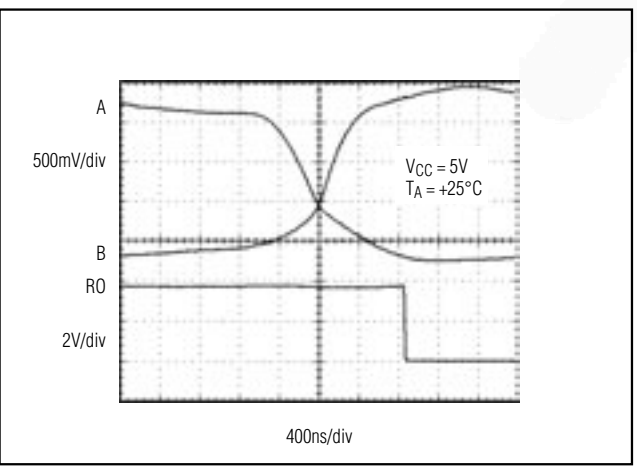


Figure 17. MAX483, MAX487-MAX489 Receiver tPHL

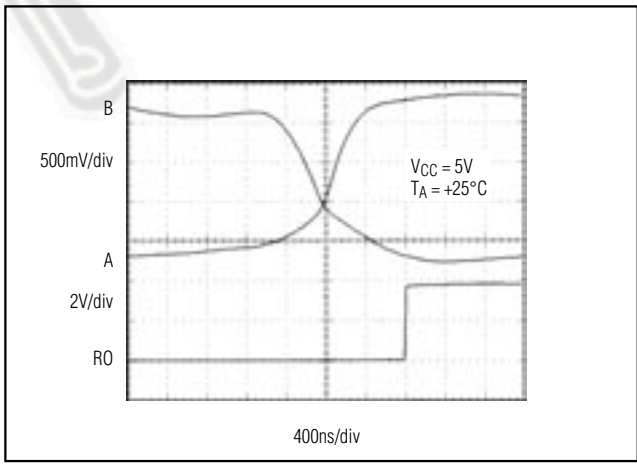


Figure 18. MAX483, MAX487-MAX489 Receiver tPLH

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Line Length vs. Data Rate

The RS-485/RS-422 standard covers line lengths up to 4000 feet. For line lengths greater than 4000 feet, see Figure 23.

Figures 19 and 20 show the system differential voltage for the parts driving 4000 feet of 26AWG twisted-pair wire at 110kHz into 120Ω loads.

Typical Applications

The MAX481, MAX483, MAX485, MAX487–MAX491, and MAX1487 transceivers are designed for bidirectional data communications on multipoint bus transmission lines.

Figures 21 and 22 show typical network applications circuits. These parts can also be used as line repeaters, with cable lengths longer than 4000 feet, as shown in Figure 23.

To minimize reflections, the line should be terminated at both ends in its characteristic impedance, and stub lengths off the main line should be kept as short as possible. The slew-rate-limited MAX483 and MAX487–MAX489 are more tolerant of imperfect termination.

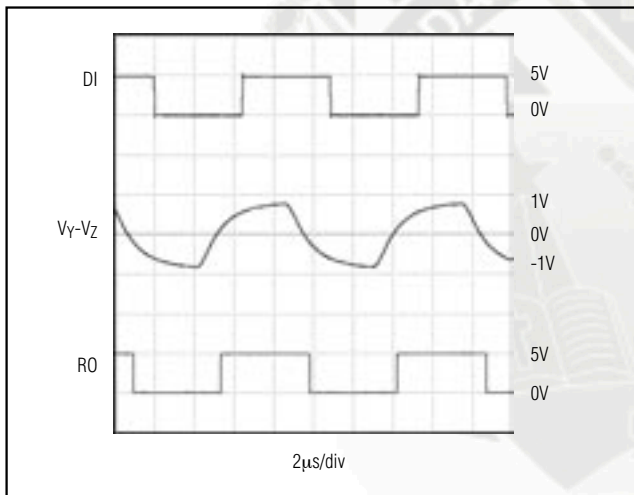


Figure 19. MAX481/MAX485/MAX490/MAX491/MAX1487 System Differential Voltage at 110kHz Driving 4000ft of Cable

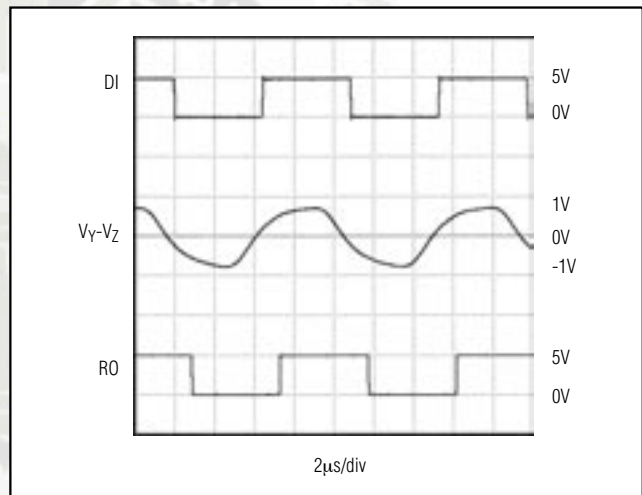


Figure 20. MAX483, MAX487–MAX489 System Differential Voltage at 110kHz Driving 4000ft of Cable

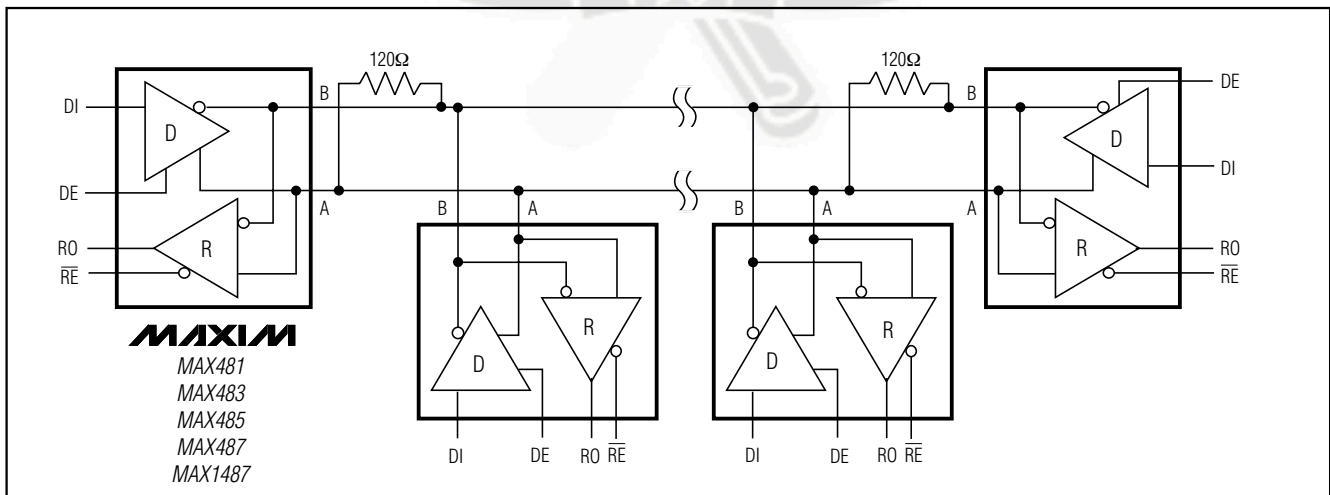


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network

MAX481/MAX483/MAX485/MAX487–MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

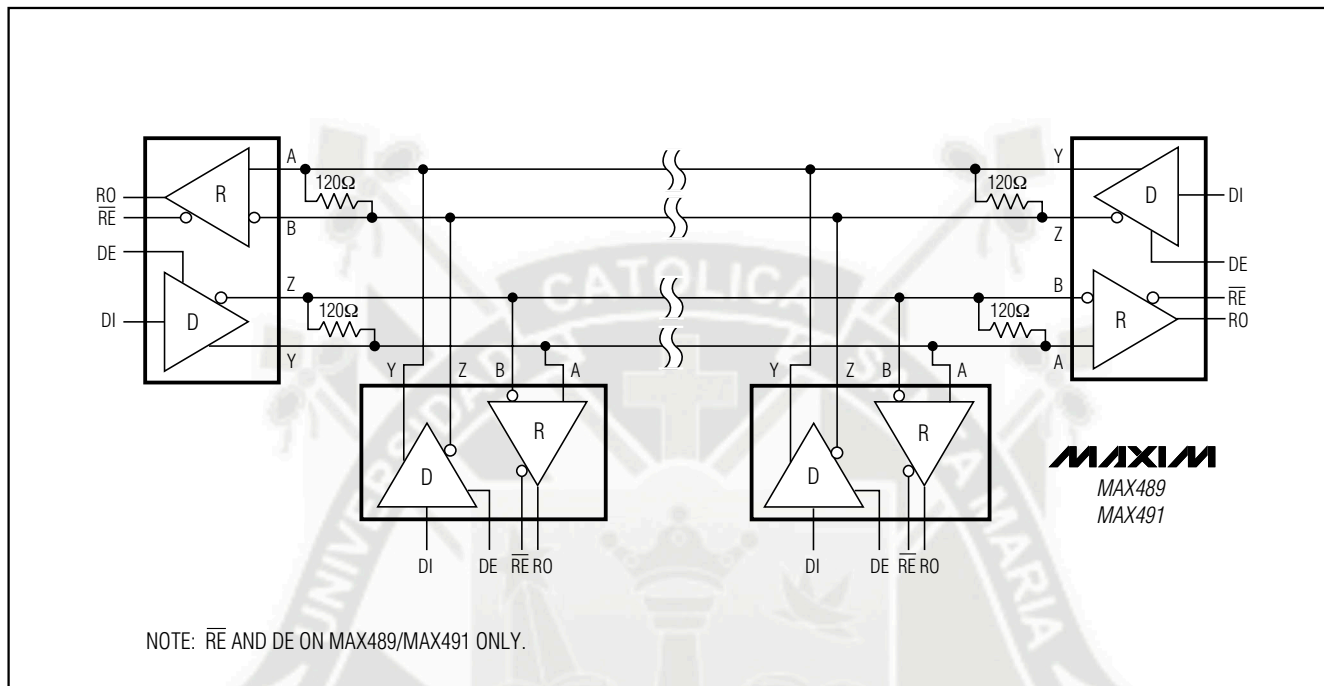


Figure 22. MAX488-MAX491 Full-Duplex RS-485 Network

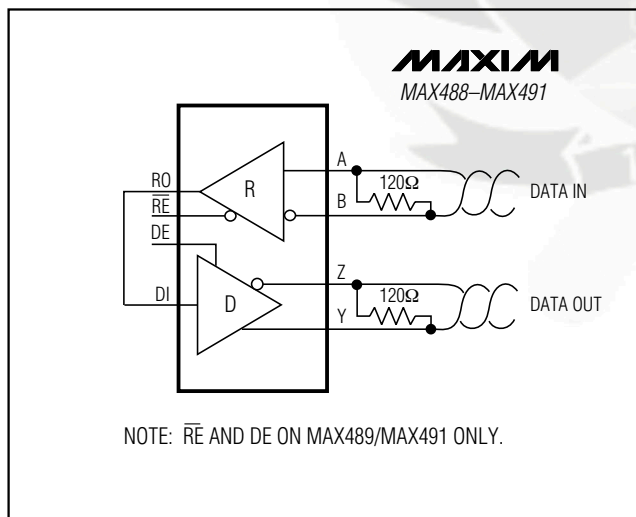


Figure 23. Line Repeater for MAX488-MAX491

Isolated RS-485

For isolated RS-485 applications, see the MAX253 and MAX1480 data sheets.

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX481 CPA	0°C to +70°C	8 Plastic DIP
MAX481CSA	0°C to +70°C	8 SO
MAX481CUA	0°C to +70°C	8 μ MAX
MAX481C/D	0°C to +70°C	Dice*
MAX481EPA	-40°C to +85°C	8 Plastic DIP
MAX481ESA	-40°C to +85°C	8 SO
MAX481MJA	-55°C to +125°C	8 CERDIP
MAX483 CPA	0°C to +70°C	8 Plastic DIP
MAX483CSA	0°C to +70°C	8 SO
MAX483CUA	0°C to +70°C	8 μ MAX
MAX483C/D	0°C to +70°C	Dice*
MAX483EPA	-40°C to +85°C	8 Plastic DIP
MAX483ESA	-40°C to +85°C	8 SO
MAX483MJA	-55°C to +125°C	8 CERDIP
MAX485 CPA	0°C to +70°C	8 Plastic DIP
MAX485CSA	0°C to +70°C	8 SO
MAX485CUA	0°C to +70°C	8 μ MAX
MAX485C/D	0°C to +70°C	Dice*
MAX485EPA	-40°C to +85°C	8 Plastic DIP
MAX485ESA	-40°C to +85°C	8 SO
MAX485MJA	-55°C to +125°C	8 CERDIP
MAX487 CPA	0°C to +70°C	8 Plastic DIP
MAX487CSA	0°C to +70°C	8 SO
MAX487CUA	0°C to +70°C	8 μ MAX
MAX487C/D	0°C to +70°C	Dice*
MAX487EPA	-40°C to +85°C	8 Plastic DIP
MAX487ESA	-40°C to +85°C	8 SO
MAX487MJA	-55°C to +125°C	8 CERDIP
MAX488 CPA	0°C to +70°C	8 Plastic DIP
MAX488CSA	0°C to +70°C	8 SO
MAX488CUA	0°C to +70°C	8 μ MAX
MAX488C/D	0°C to +70°C	Dice*
MAX488EPA	-40°C to +85°C	8 Plastic DIP
MAX488ESA	-40°C to +85°C	8 SO
MAX488MJA	-55°C to +125°C	8 CERDIP
MAX489 CPD	0°C to +70°C	14 Plastic DIP
MAX489CSD	0°C to +70°C	14 SO
MAX489C/D	0°C to +70°C	Dice*
MAX489EPD	-40°C to +85°C	14 Plastic DIP
MAX489ESD	-40°C to +85°C	14 SO
MAX489MJD	-55°C to +125°C	14 CERDIP

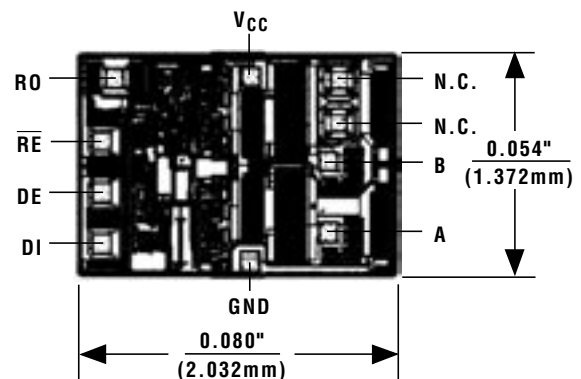
Ordering Information (continued)

PART	TEMP. RANGE	PIN-PACKAGE
MAX490 CPA	0°C to +70°C	8 Plastic DIP
MAX490CSA	0°C to +70°C	8 SO
MAX490CUA	0°C to +70°C	8 μ MAX
MAX490C/D	0°C to +70°C	Dice*
MAX490EPA	-40°C to +85°C	8 Plastic DIP
MAX490ESA	-40°C to +85°C	8 SO
MAX490MJA	-55°C to +125°C	8 CERDIP
MAX491 CPD	0°C to +70°C	14 Plastic DIP
MAX491CSD	0°C to +70°C	14 SO
MAX491C/D	0°C to +70°C	Dice*
MAX491EPD	-40°C to +85°C	14 Plastic DIP
MAX491ESD	-40°C to +85°C	14 SO
MAX491MJD	-55°C to +125°C	14 CERDIP
MAX1487 CPA	0°C to +70°C	8 Plastic DIP
MAX1487CSA	0°C to +70°C	8 SO
MAX1487CUA	0°C to +70°C	8 μ MAX
MAX1487C/D	0°C to +70°C	Dice*
MAX1487EPA	-40°C to +85°C	8 Plastic DIP
MAX1487ESA	-40°C to +85°C	8 SO
MAX1487MJA	-55°C to +125°C	8 CERDIP

* Contact factory for dice specifications.

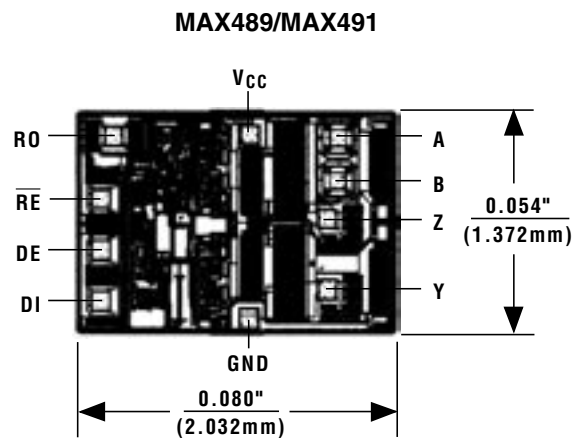
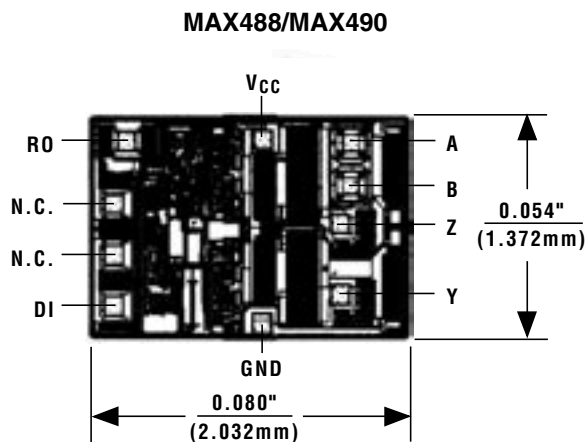
Chip Topographies

MAX481/MAX483/MAX485/MAX487/MAX1487



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Chip Topographies (continued)

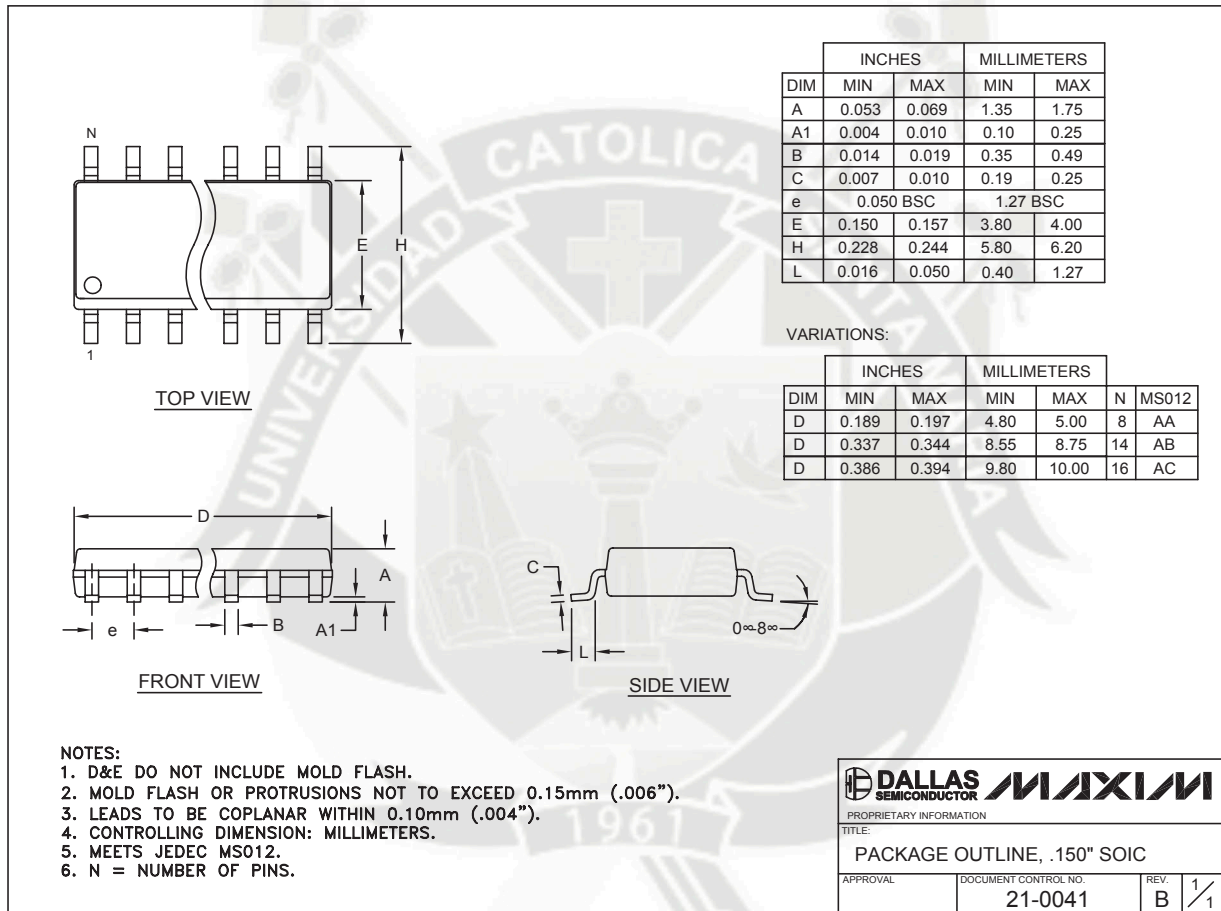


TRANSISTOR COUNT: 248
SUBSTRATE CONNECTED TO GND

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Package Information

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)



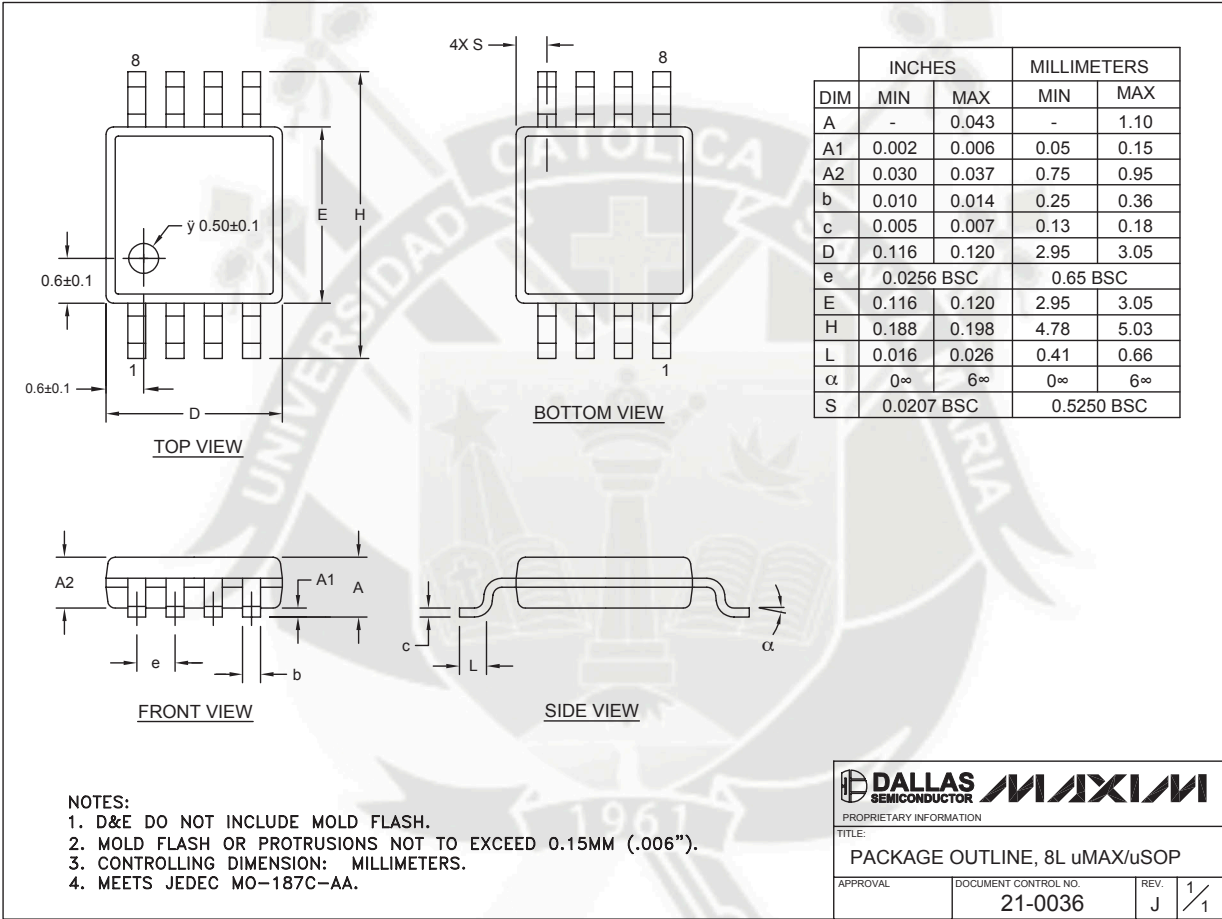
SOICN.EPS

MAX481/MAX483/MAX485/MAX487-MAX491/MAX1487

Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Package Information (continued)

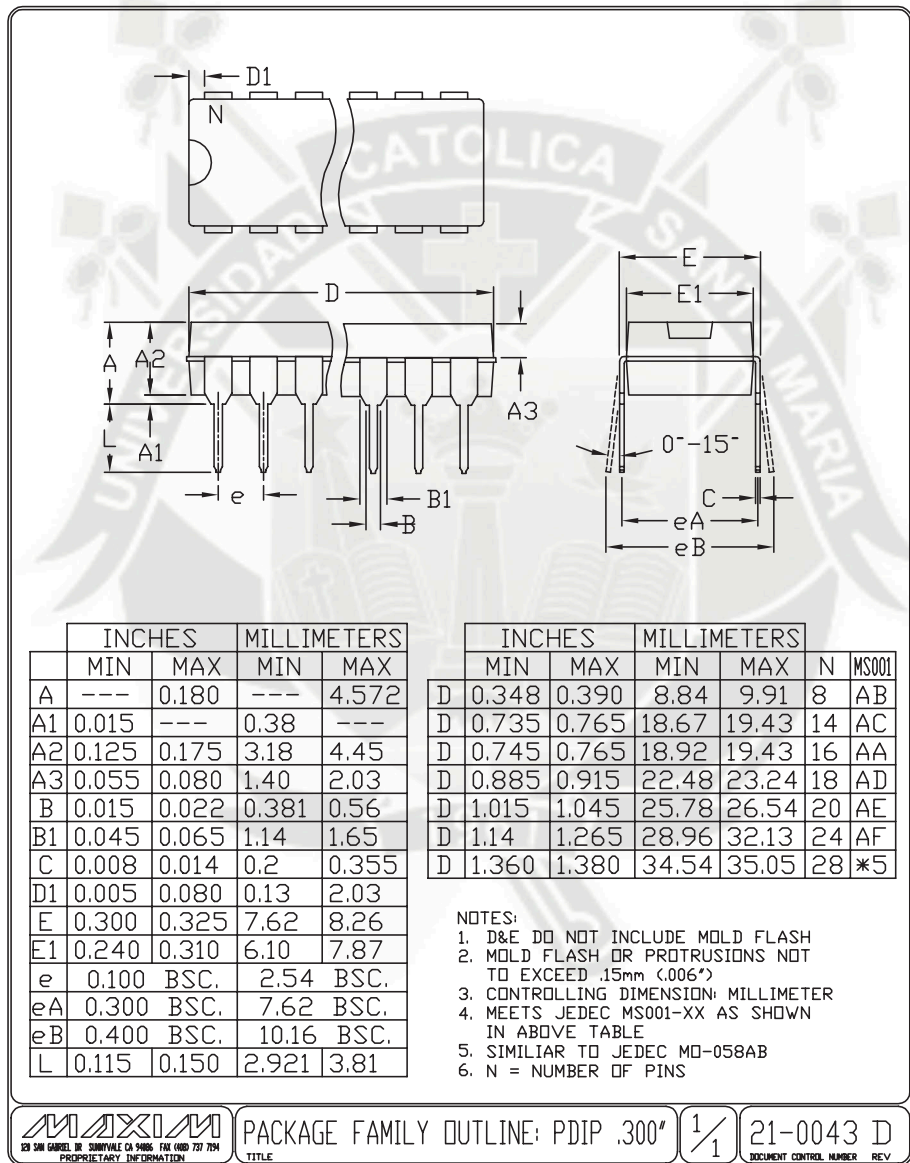
(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)



Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)



Maxim cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim product. No circuit patent licenses are implied. Maxim reserves the right to change the circuitry and specifications without notice at any time.

Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600 19



台灣艾華電子工業股份有限公司
ALPHA TAIWAN ALPHA ELECTRONIC CO., LTD.

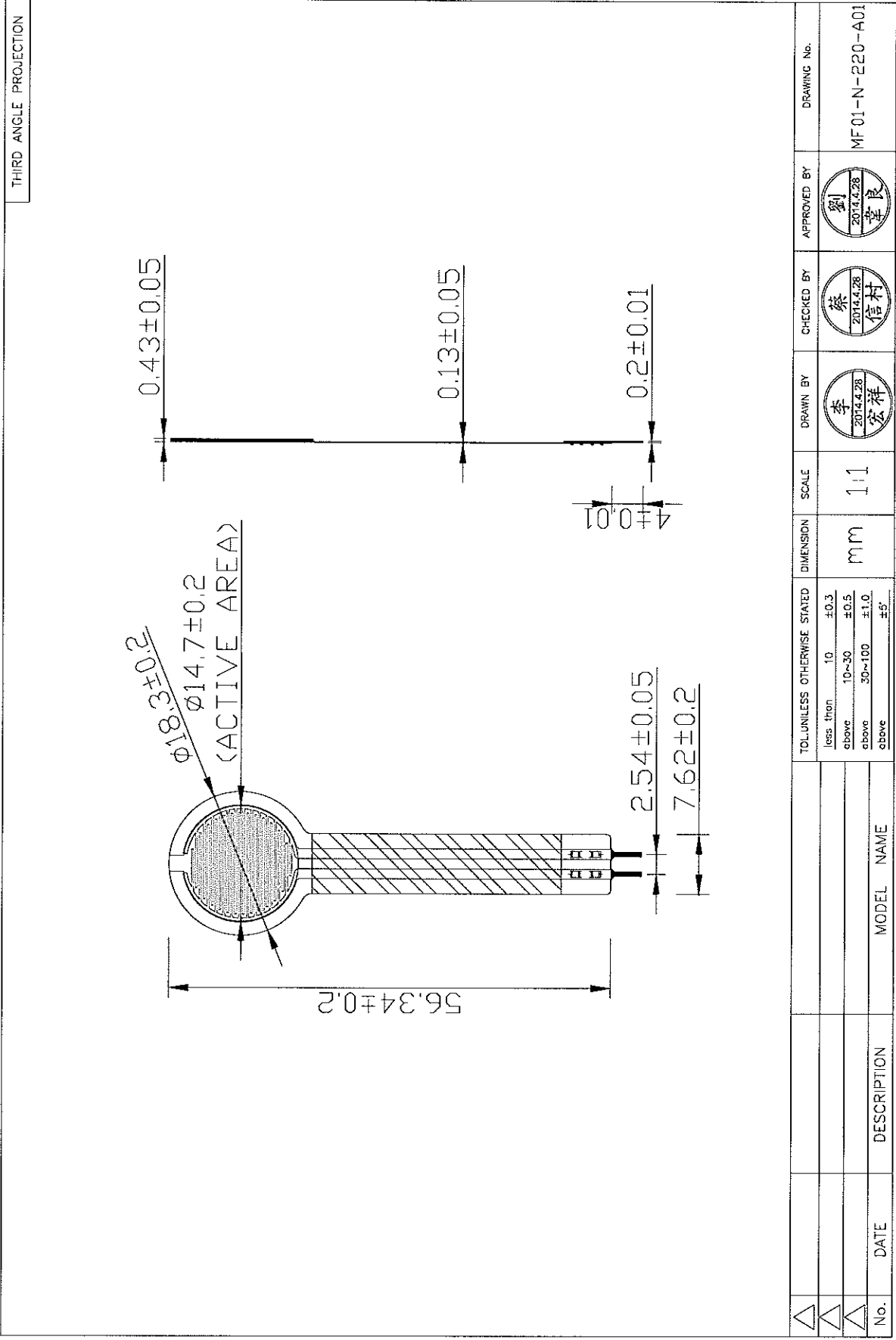
MF01-N-220-A01 規格書

SPECIFICATION

檢驗項目 Inspection Item	規格 SPEC.	備註 Notes
起始按壓力 Actuation Force	30g min.	
按壓力靈敏度範圍 Force Sensitivity Range	30~1000g	
再現性 Force repeatability (Single part)	±5%	
再現性 Force repeatability (Part to Part)	±20%	
絕緣阻抗 Ins. Resistance	>20MΩ	Without loading
遲滯 Hysteresis (Rf+)-(Rf-)/Rf+	10%	
反應時間 Response time	<1ms	
操作溫度 Operation Temp	-30~70°C	
使用壽命 Life Cycle	1 million	
厚度 Thickness	0.43±0.05mm	

本資料為台灣艾華電子工業股份有限公司之機密與專有財產，非經書面許可，不得對外透露或使用本文件，亦不得複印、複製或轉變成任何其他形式使用。

The Information is confidential property of Taiwan Alpha Electronic Co., LTD. Unauthorized distribution, reproduced or disclosed part without prior written permission of Taiwan Alpha Electronic Co., LTD. is strictly prohibited.



ABS FILAMENT DATASHEET

ABS is short for Acrylonitrile Butadiene Styrene plastic.

Diameter: 1.75mm / 3mm

Tolerance Roundness: ± 0.05

Melt Point: 230 °C

Print Temperature: 220 °C - 270 °C

Length per Roll (m): 275/150

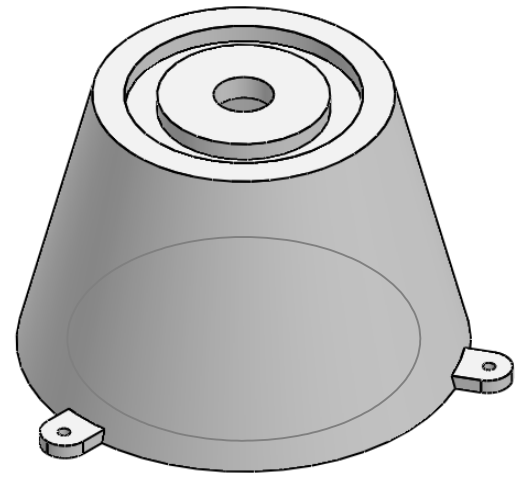
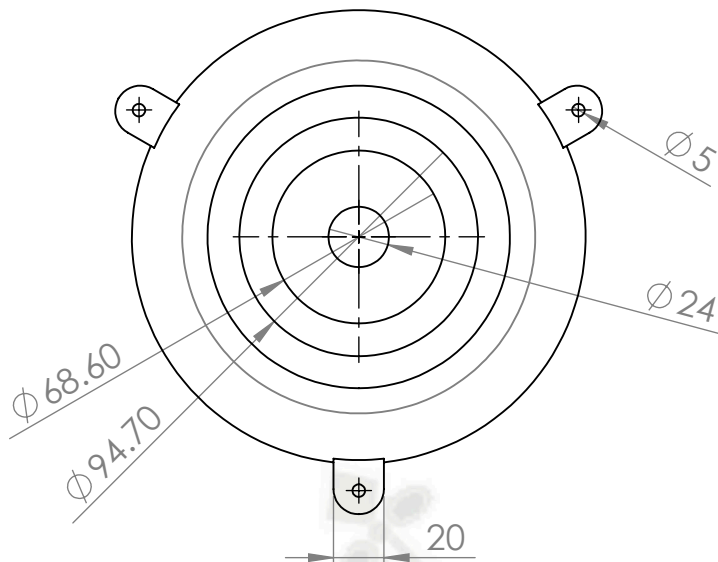
Net weight of Each roll: 1KG

Advantage: High strength, good toughness, stable in performance

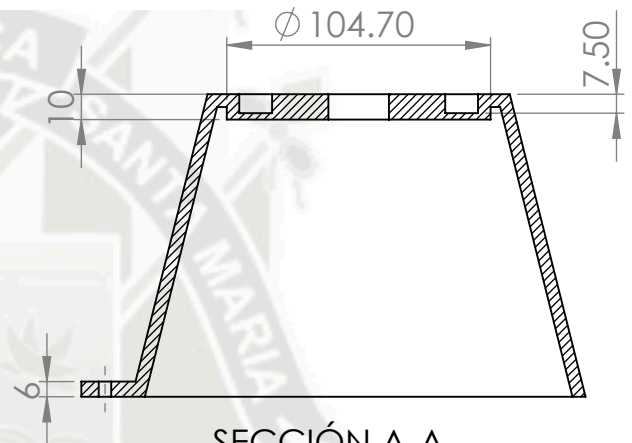
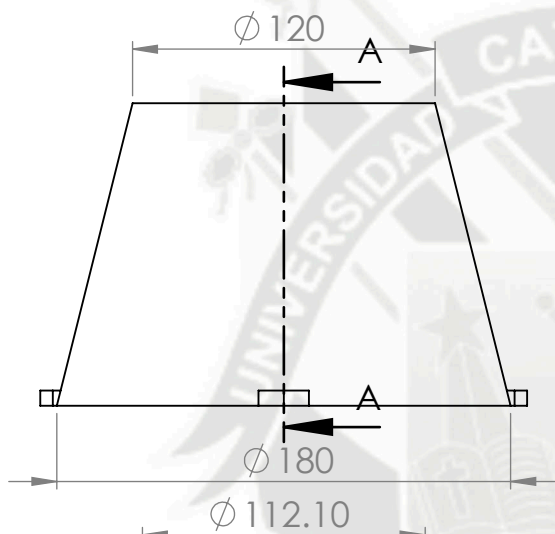
Suitable 3D printer: Makerbot, Up, RepRap, Cubify, Solidoodle, NesStarWay, Leapfrog, Afinia, Ultimaker, Mendel, Prusa I3 any other FDM 3D printer

Available colors: Transparent, Transparent-Yellow, Transparent-Green, Transparent-Orange, Transparent-Purple, Transparent-Red, Transparent- Blue, Fluo-Yellow, Fluo-Green, Fluo-Orange, Fluo-Red, Fluo-Blue, Natural, White, Yellow, Dark Yellow/107C, Green, Grass Green/G-13280, Dark Green/349C, Orange, Pink, Light Pink/107C, Dark Pink/P-13283, Purple, dark purple/V-13282, Red, Light Blue/291C, Blue, dark blue/B-13281, Aqua Blue, Golden, Golden Yellow, Coffee Brown, Brown, Gray, Silver, Black

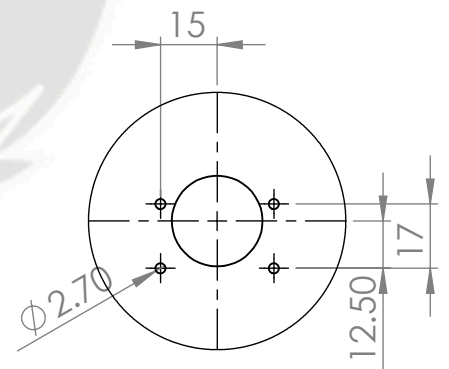
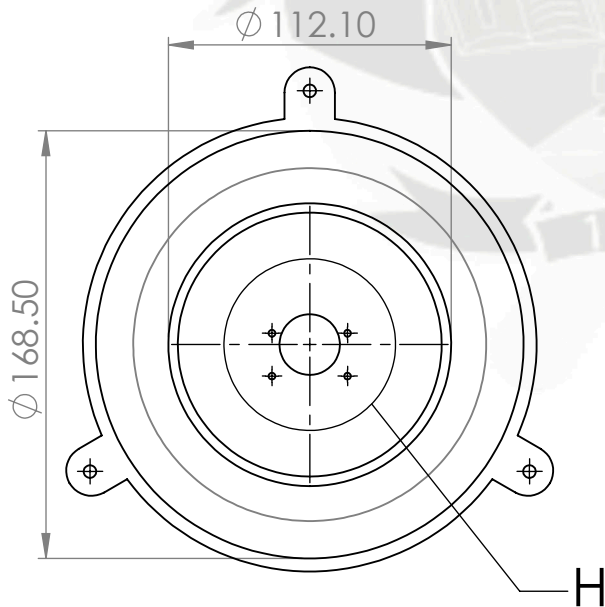
ABS Filament		
Physical	Nominal Value Unit	Test Method
Specific Gravity		
..	1.05g/cm ³	ASTM D792
23°C	1.05g/cm ³	ISO1183
Melt Mass-flow rate (MFR)		ASTM D1238
200°C/5.0KG	1.6g/10min	
220°C/10.0kg	22g/10min	
Melt Volume-Flow Rate(MVR")(220°C/10.0kg)	22cm ³ /10min	ISO1133
Mechanical	Nominal Value Unit	Test Method
Tensile Stress		
Yield	54.0Pa	ISO 527-2/50
Break	37.0MPa	ISO 527-2/50
3.00mm	45.1MPa	ASTM D638
Tensile Elongation		
Break,3.00mm	25%	ASTM D638
Break	20%	ISO527 -2/50
Flexural Modulus		
6.00mm	2650MPa	ASTM D790
	2200MPa	ISO 178
Flexural Strength		
6.00mm	77.5MPa	ASTM D790
	760MPa	ISO 178
Impact	Nominal	Test Method
Charpy Notched Impact Strength	20kj/m2	ISO179
Charpy Unnotched Impact Strength	No Break	ISO179
Notched Izod Impact		
23°C,3.00mm	210j/m	ASTM D256
23°C,6.00mm	180j/m	ASTM D256
—	18kj/m2	ISO 180/1A
Unnotched Izod Impact Strength	60kj/m2	ISO 180/1U
Hardness	Nominal Value Unit	Test Method
Rockwell Hardness(R-Scale)	116	ASTM D785
Ball Indentation Hardess(H 358/30)	110MPa	ISO 2093-1
Thermal	Nominal Value Unit	Test Method
Deflection Temperature Under Load		
1.8MPa,Unannesled	85°C	ASTM D648
1.8MPa,Unannesled	88.0°C	ISO 75-2/A
1.8MPa,Annealed	95.0°C	ASTM D648
1.8MPa,Annealed	98.0°C	ISO 75-2/A
Vicat Softening Tempersture		
—	105°C	ASTM D1527
—	104°C	ISO 306/A50
—	100°C	ISO 306/B50
Flammability	Nominal Value Unit	Test Method
Flame Rating (1.60mm)	HB	UL94
Processing Information	180to220°C	



VISTA ISOMETRICA
ESCALA 1 : 3



SECCIÓN A-A
ESCALA 1 : 3



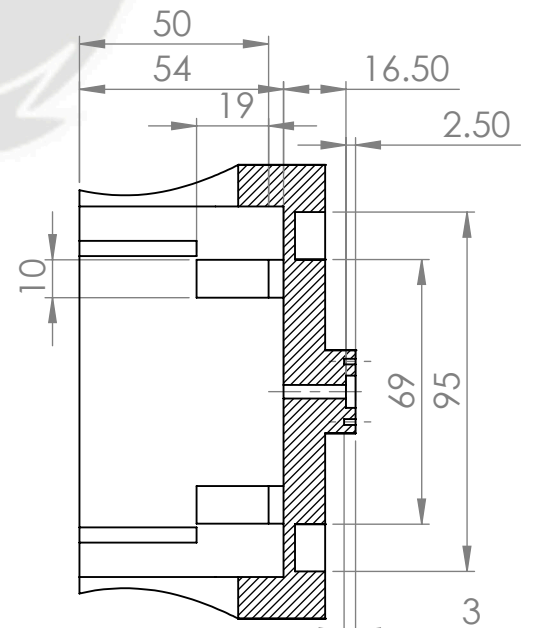
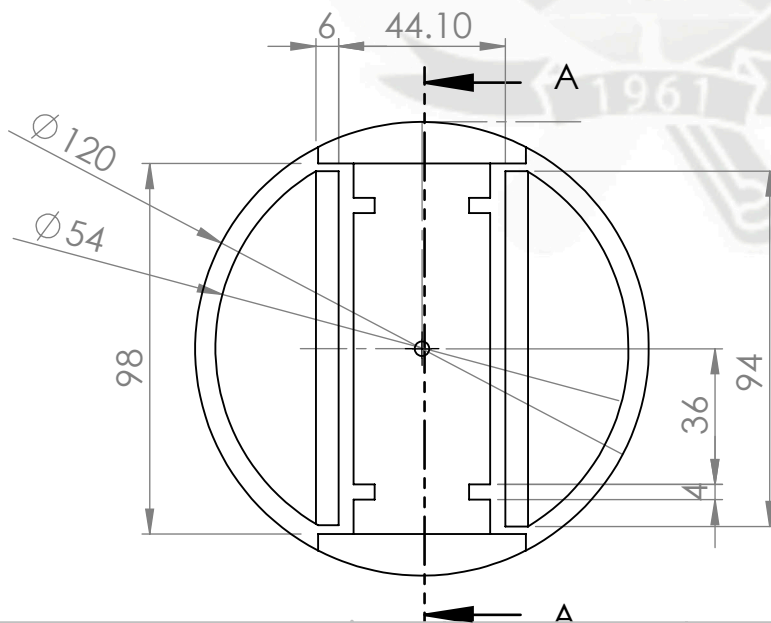
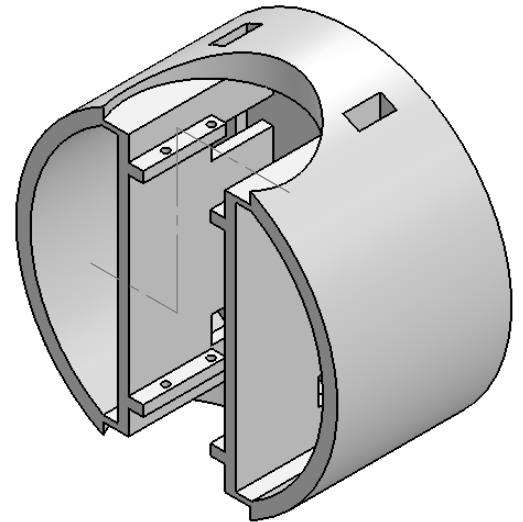
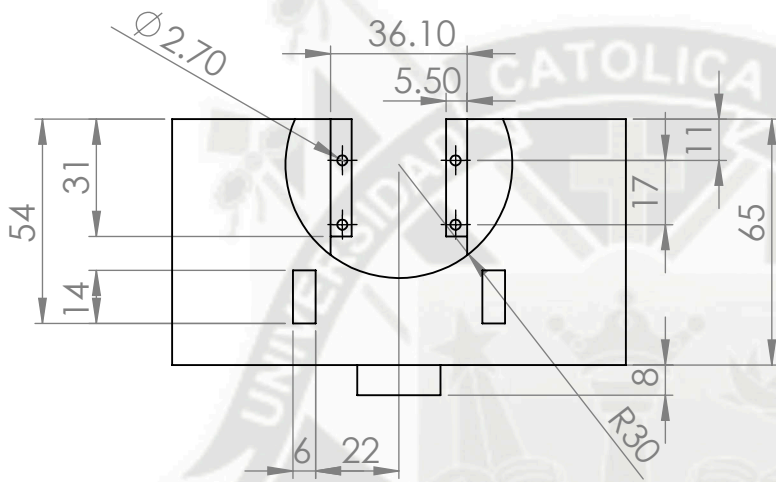
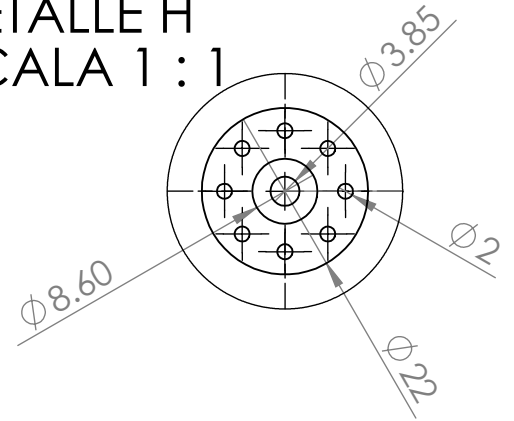
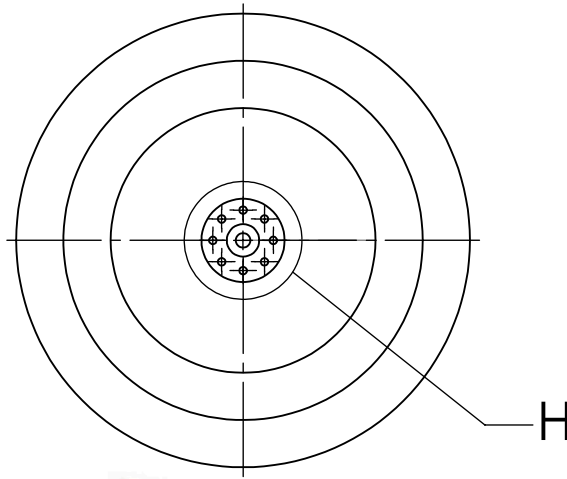
DETALLE H
ESCALA 1 : 2

	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016		
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	1 - 1
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm			PESO:	ESCALA:1:3	HOJA 1 DE 1

Base 1

A4

DETALLE H ESCALA 1 : 1

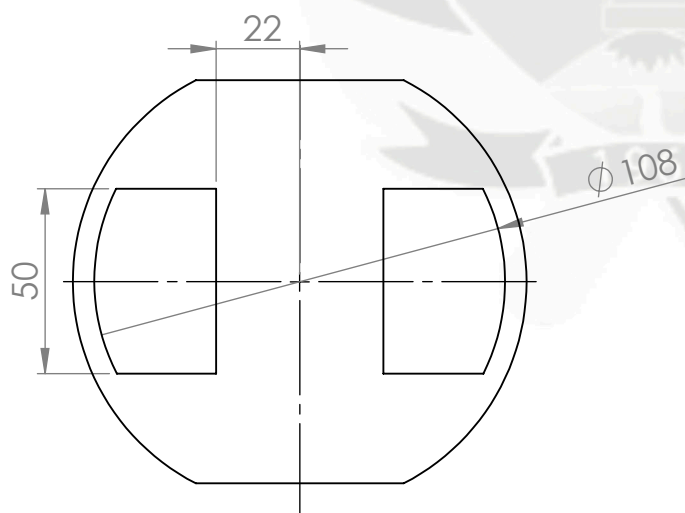
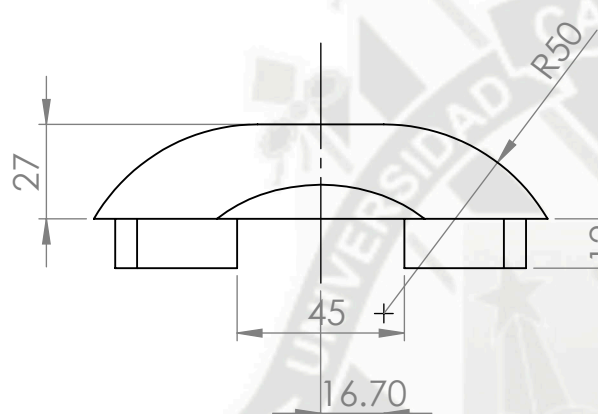
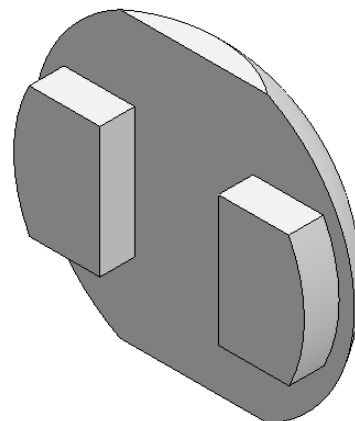
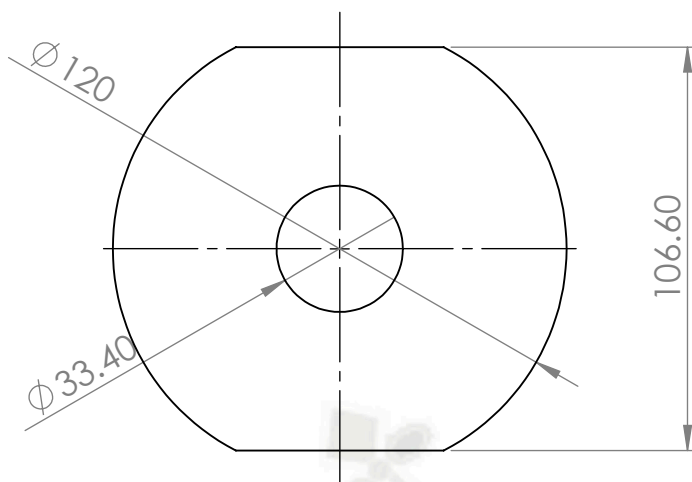


SECCIÓN A-A

	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016		
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	2 - 1
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm			PESO:	ESCALA: 1:2	HOJA 1 DE 1

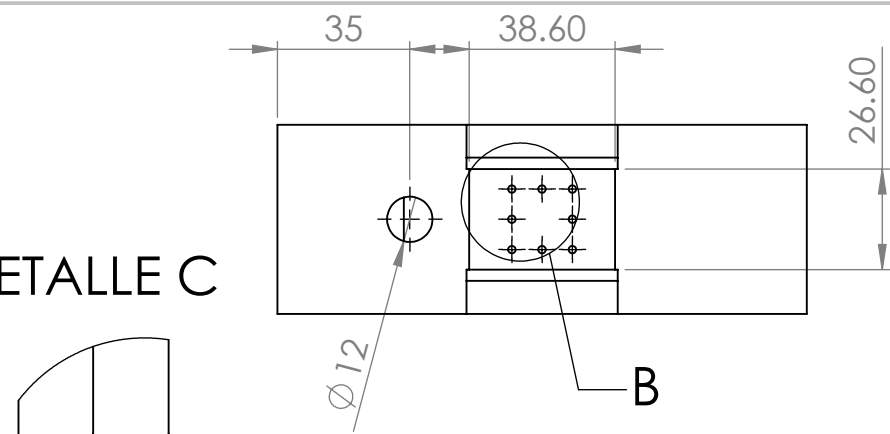
Base 2

A4

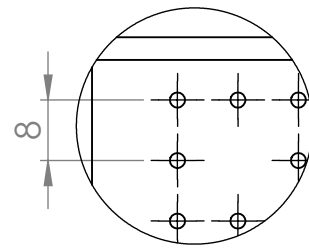


	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		19/04/2016	Tapa de la Base 2
VERIF.	BRAVO ROMERO, Bresly			
APROB.				
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:	2 - 2	A4
			ESCALA: 1:2	HOJA 1 DE 1

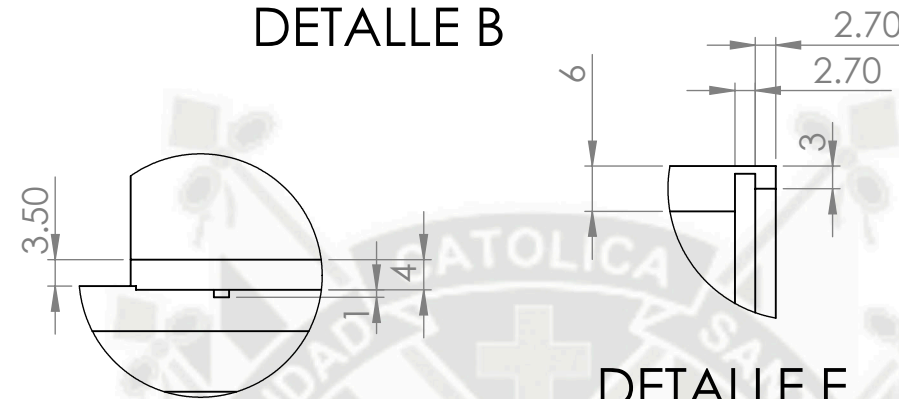
DETALLE C



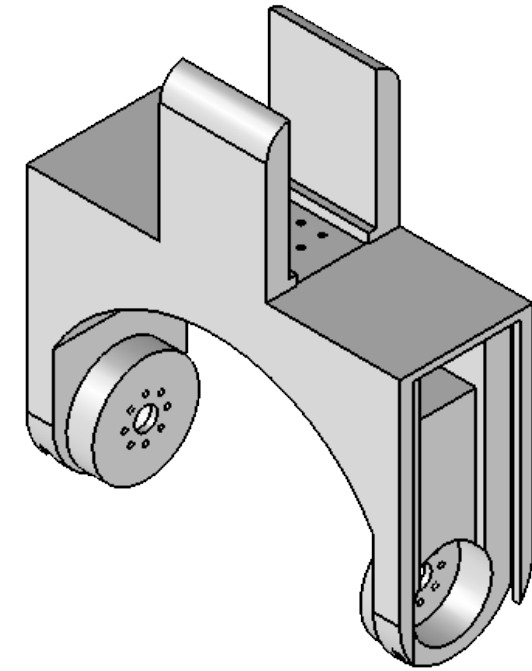
DETALLE B



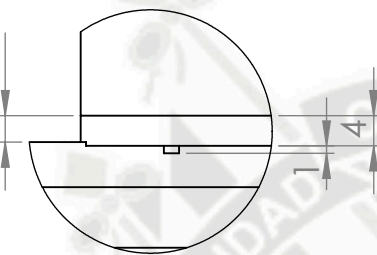
DETALLE E



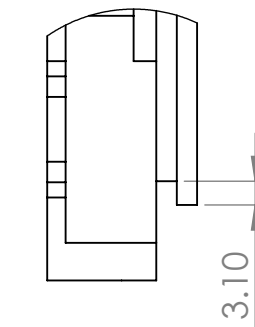
DETALLE F



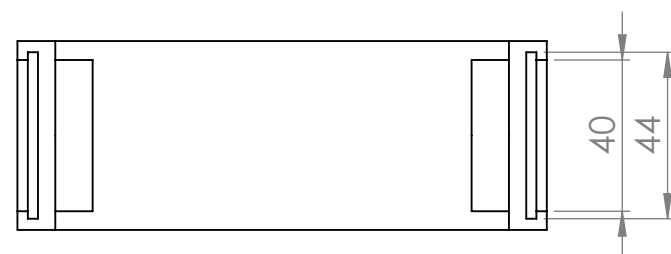
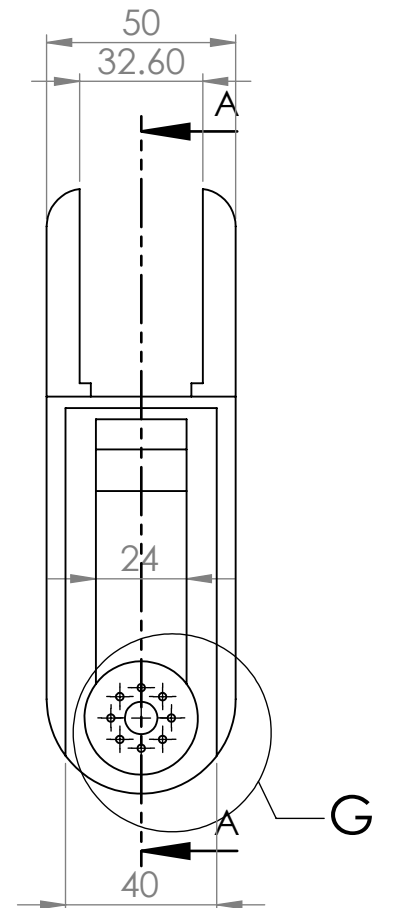
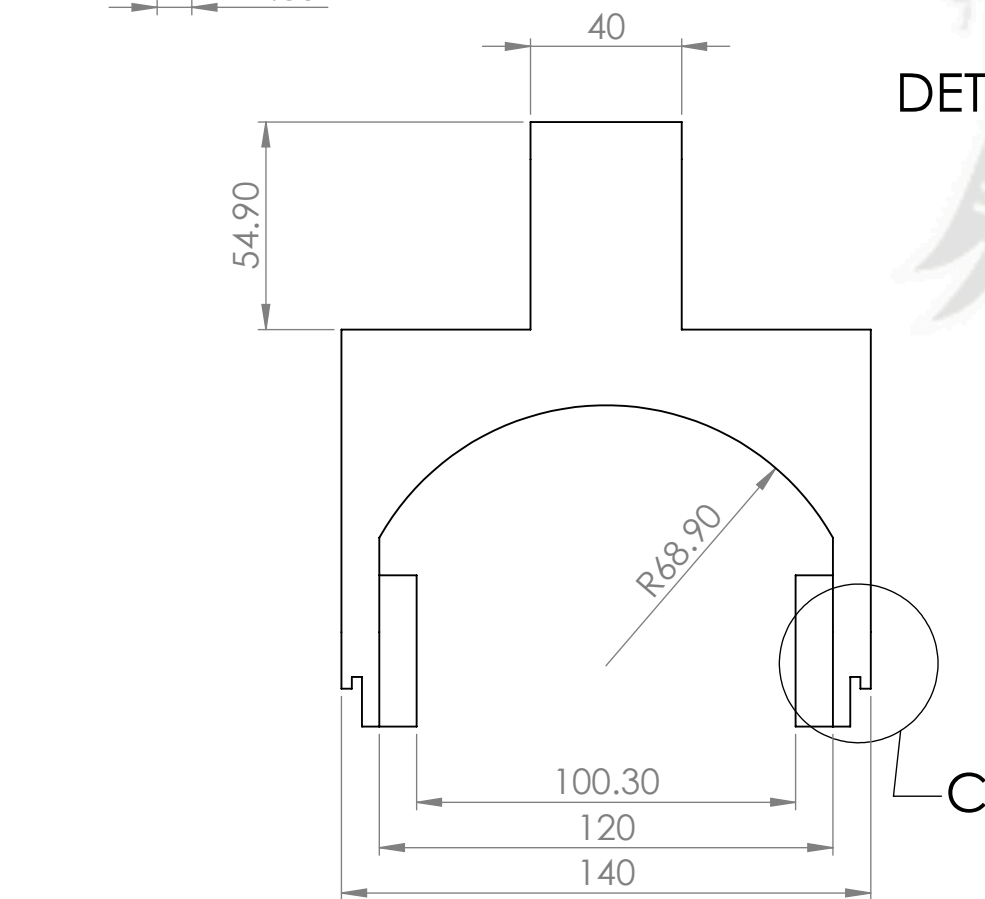
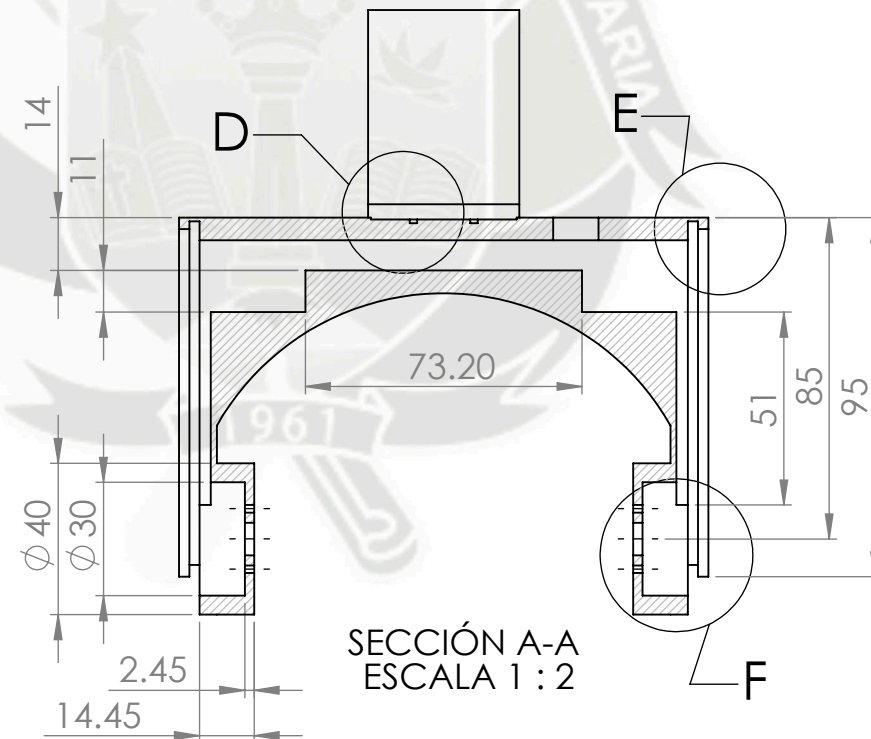
DETALLE D



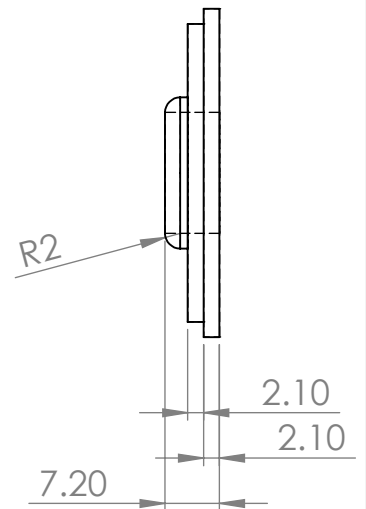
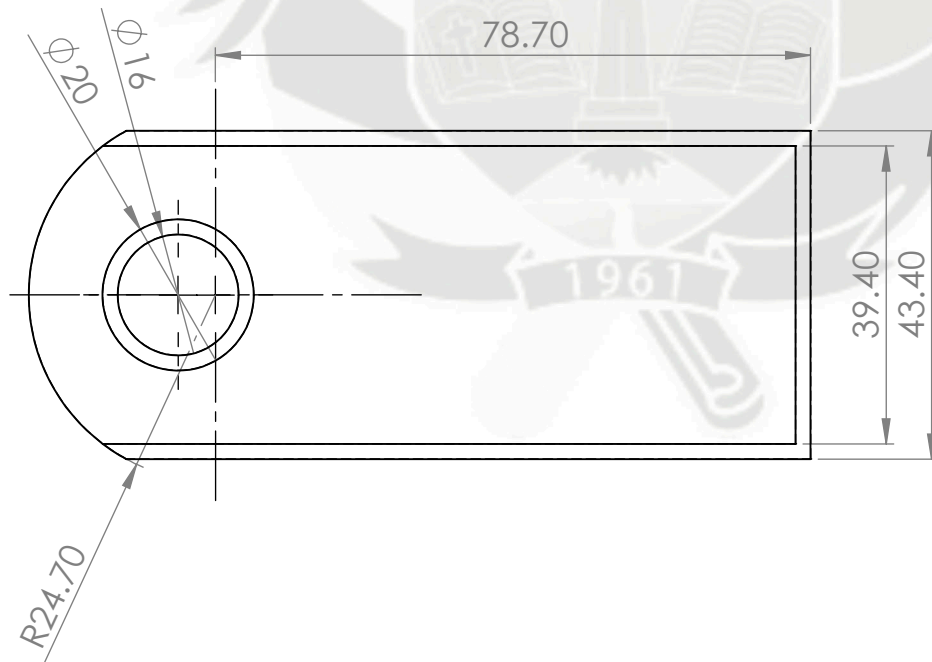
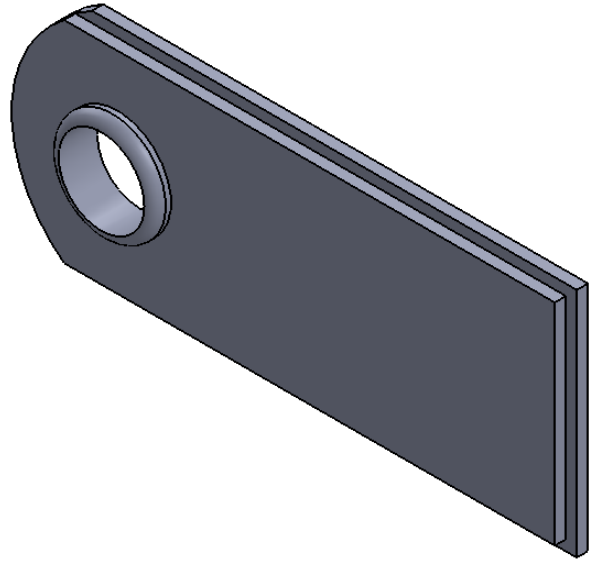
DETALLE G



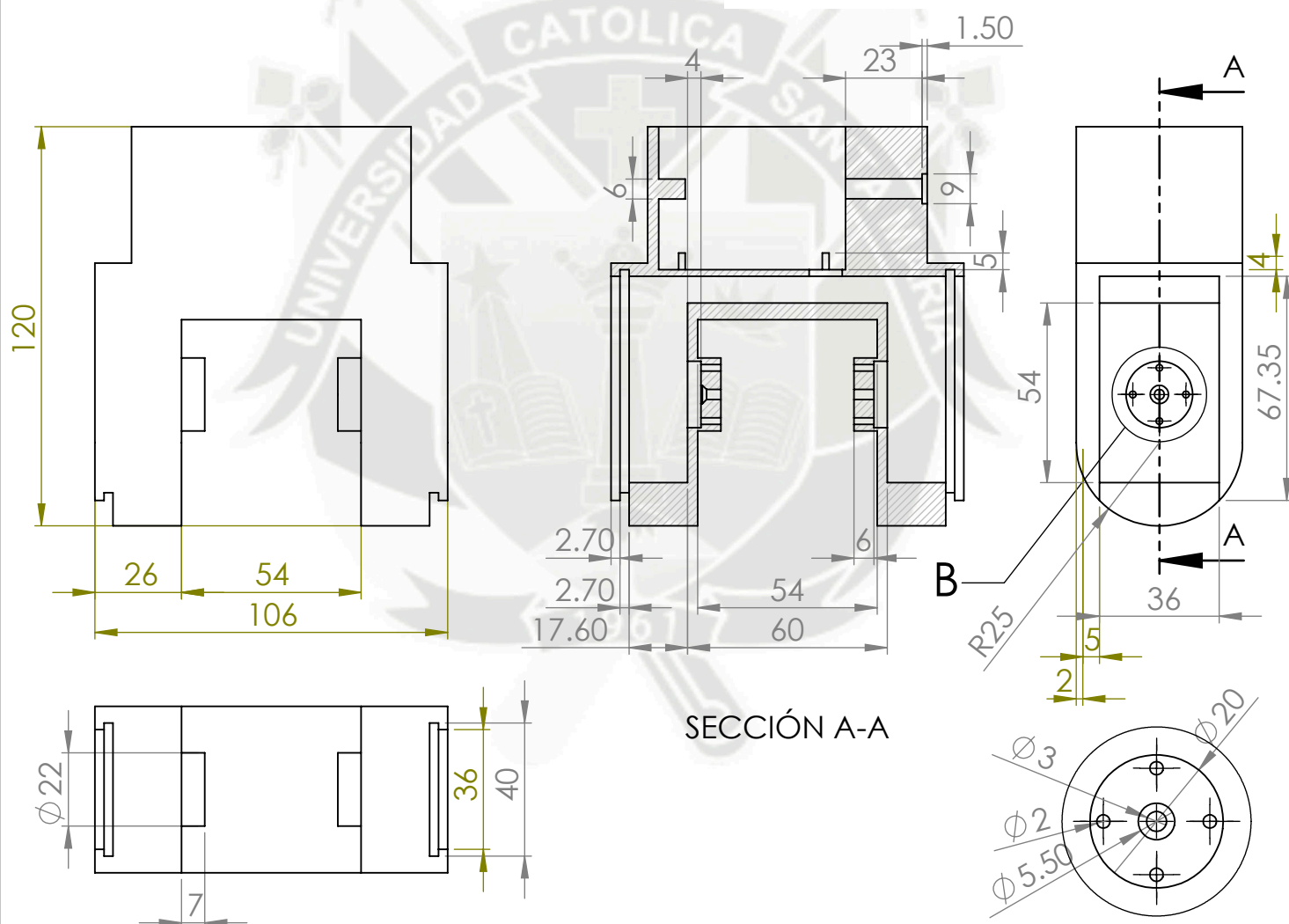
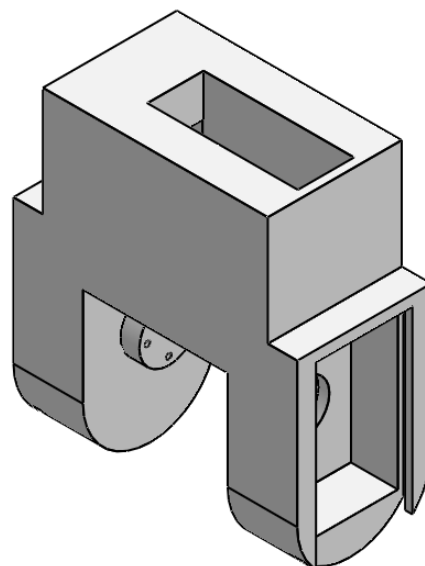
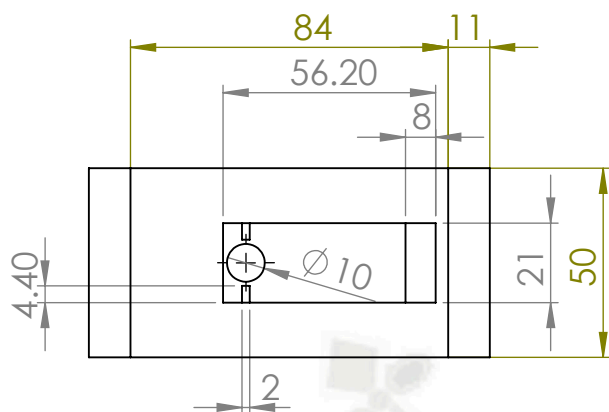
SECCIÓN A-A
ESCALA 1:2



NOMBRE		FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Eslabon 2	
	BRAVO ROMERO, Bresly				
VERIF.					
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	A3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm				3 - 1	
		PESO:	ESCALA:1:1	HOJA 1 DE 1	



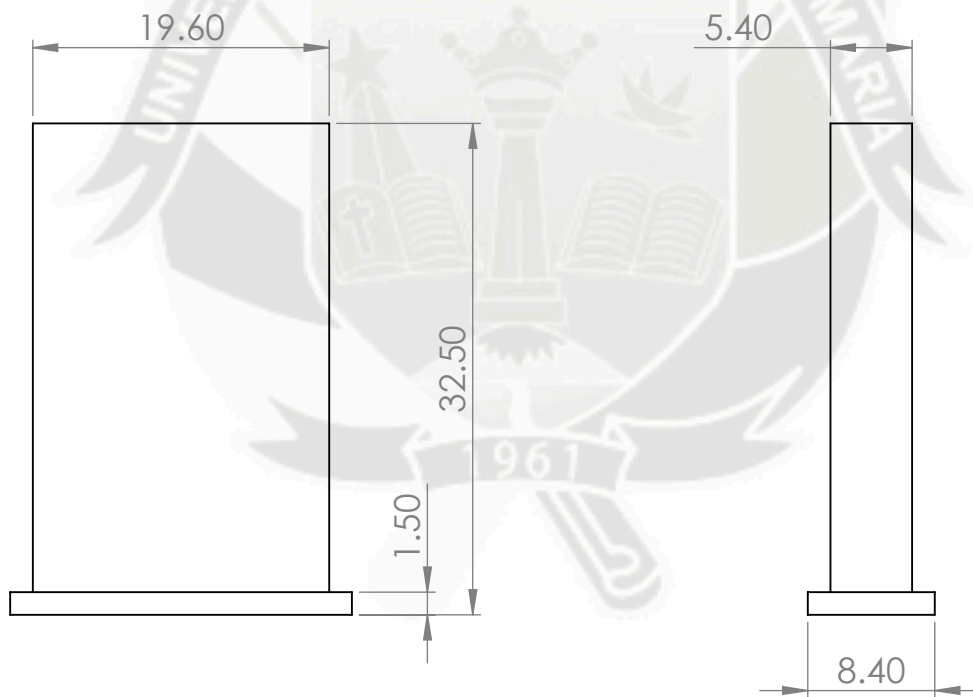
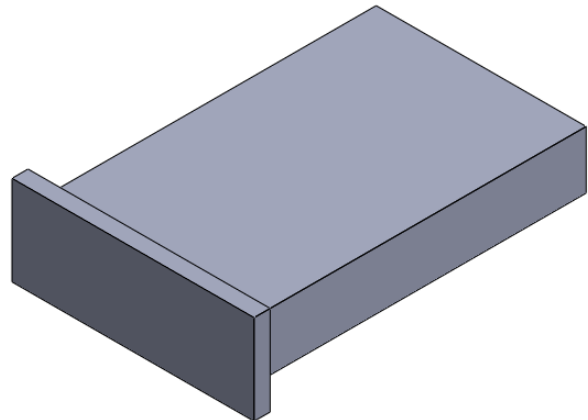
	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Tapa del Eslabón 1
VERIF.	BRAVO ROMERO, Bresly			
APROB.				
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		3 - 2
				A4
			ESCALA:1:1	HOJA 1 DE 1



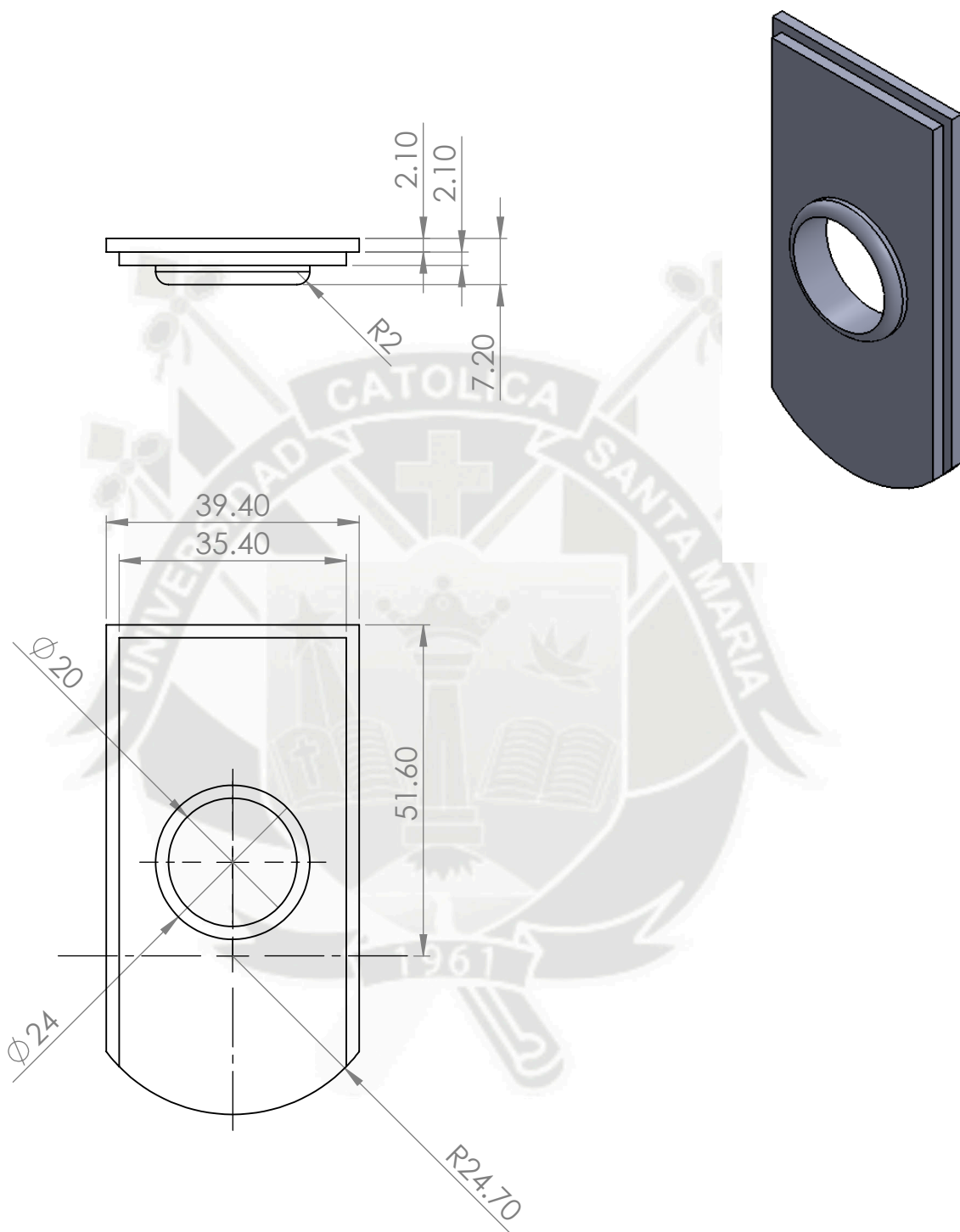
SECCIÓN A-A

DETALLE B
ESCALA 1 : 1

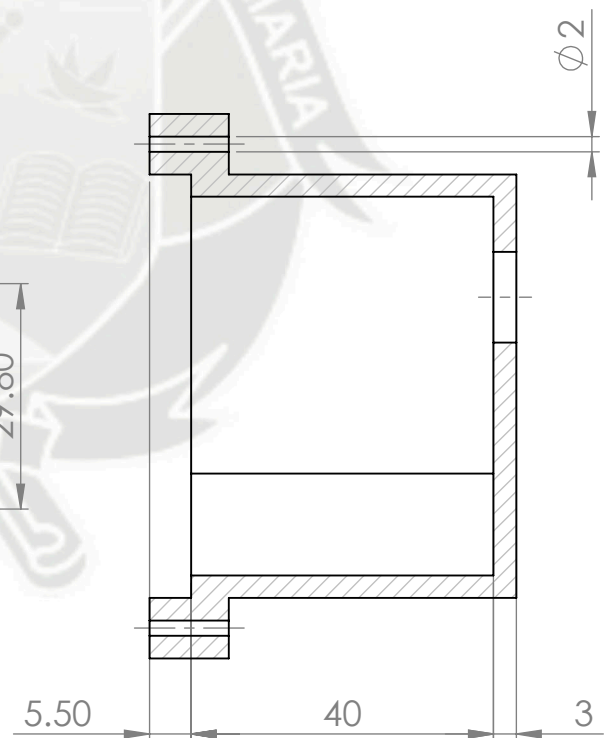
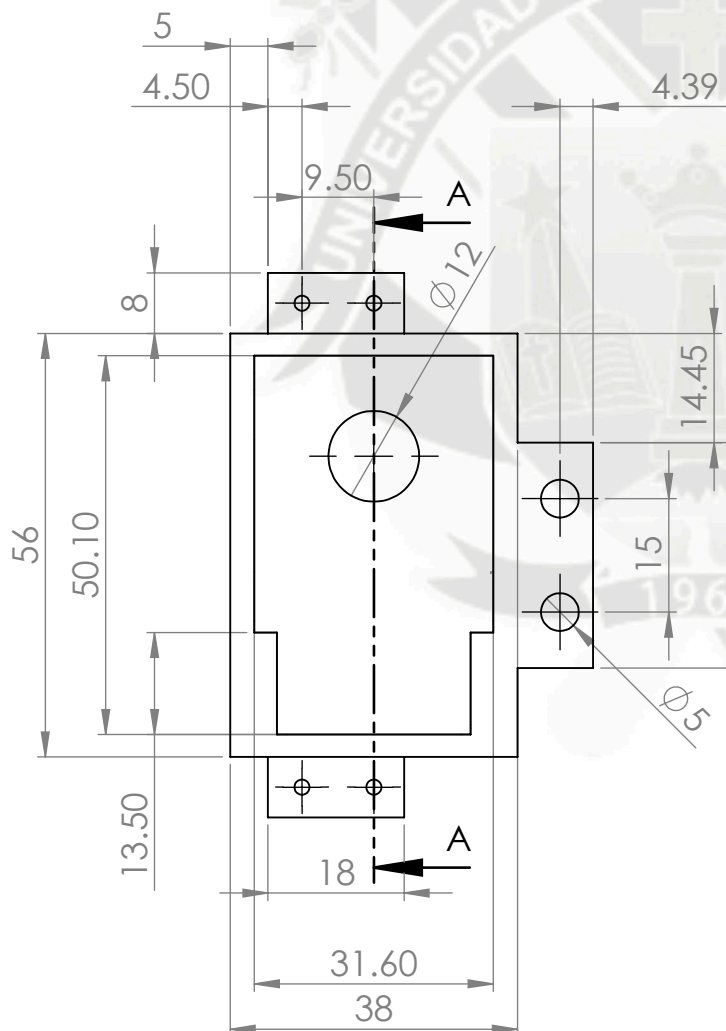
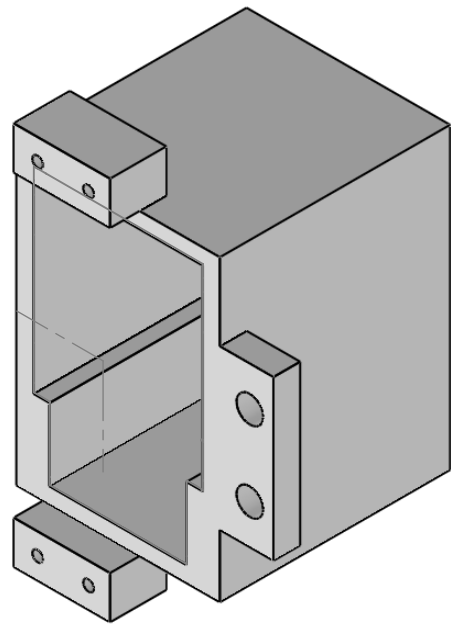
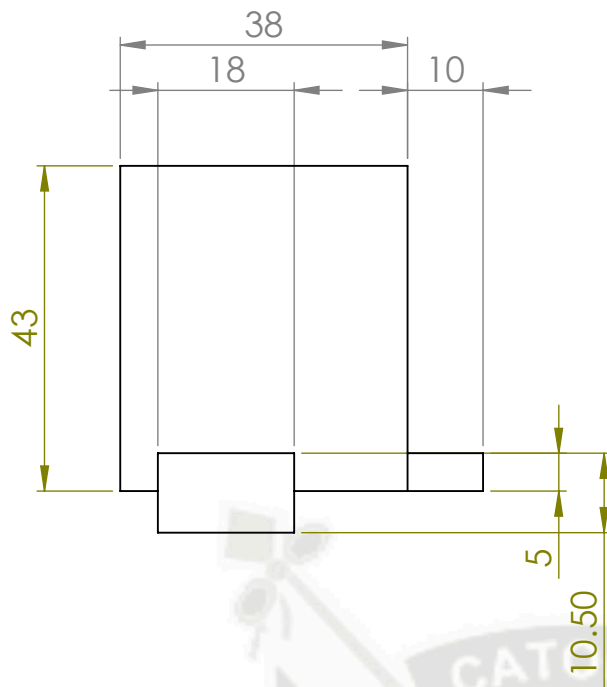
	NOMBRE	FIRMA	FECHA	TÍTULO:
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Eslabon 2
VERIF.	BRAVO ROMERO, Bresly			
APROB.				
FABR.	IMPRESION 3D			
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		MATERIAL:	ABS	N.º DE DIBUJO
		PESO:		4 - 1
				ESCALA: 1:2
				HOJA 1 DE 1



NOMBRE		FIRMA	FECHA	TÍTULO:		
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Soporte motor		
	BRAVO ROMERO, Breslly					
VERIF.						
APROB.						
FABR.	IMPRESION 3D	MATERIAL:		N.º DE DIBUJO		A4
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		ABS		4 - 2		
		PESO:		ESCALA:2:1		HOJA 1 DE 1

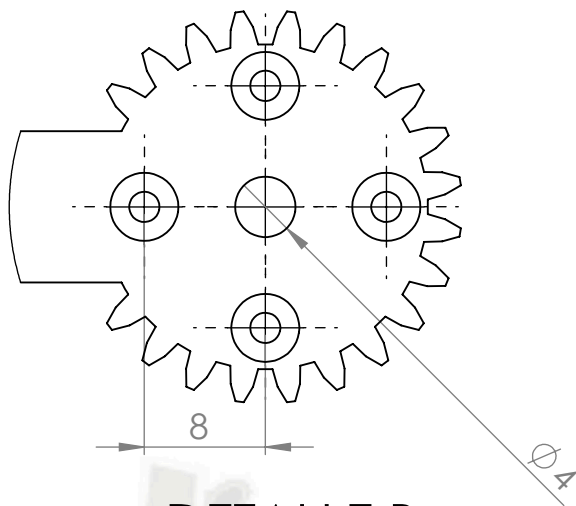


	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Tapa del Eslabón 2	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	4 - 3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA: 1:1	HOJA 1 DE 1
					A4

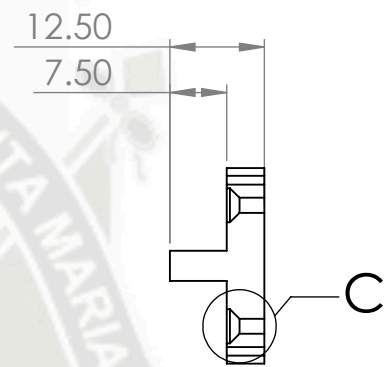
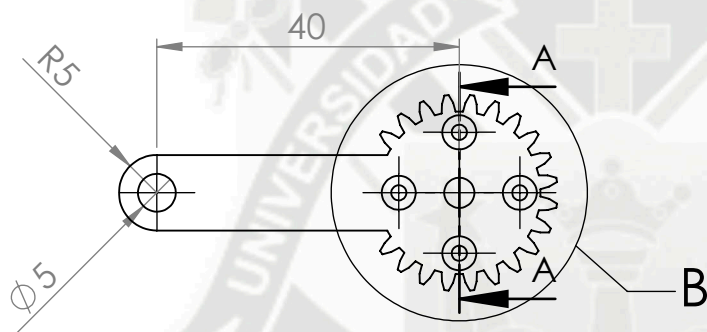
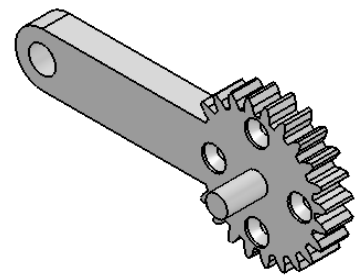


SECCIÓN A-A
ESCALA 1 : 1

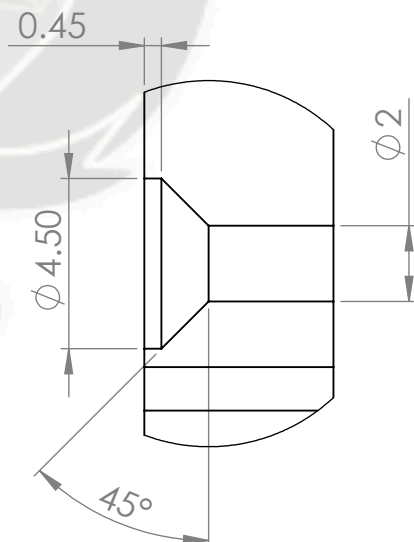
	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Caja motor	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	5 - 1
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA:1:1	HOJA 1 DE 1
					A4



DETALLE B
ESCALA 2 : 1

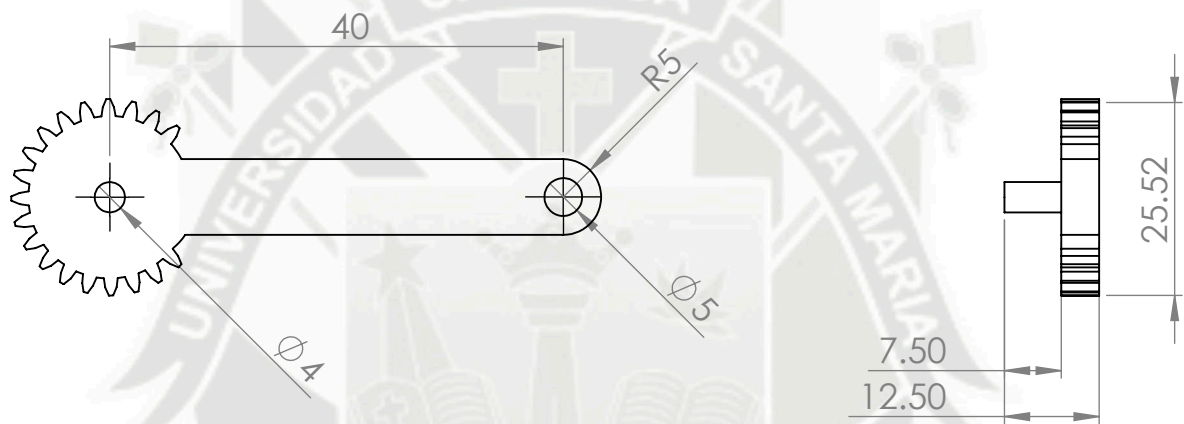
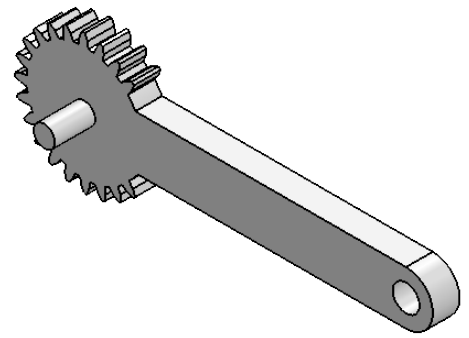


SECCIÓN A-A



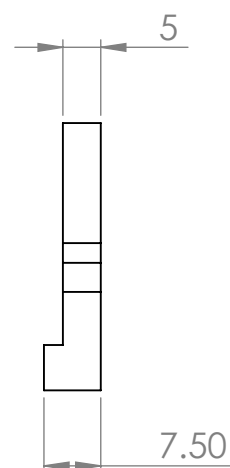
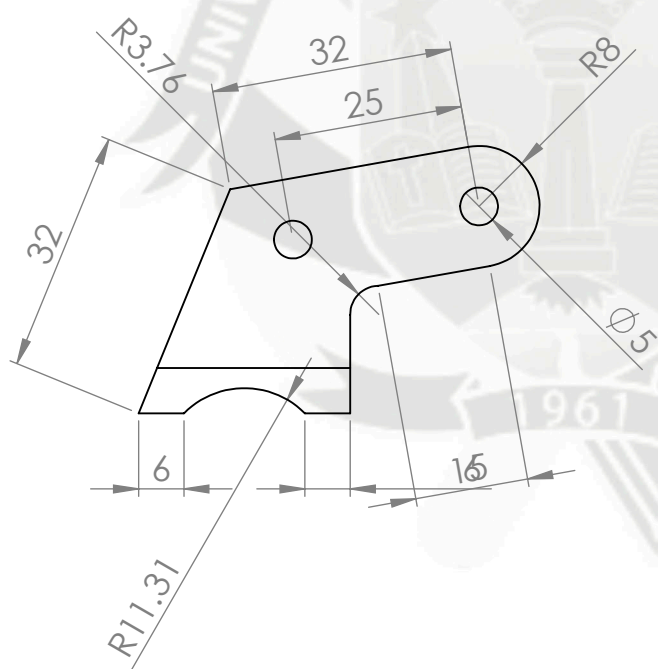
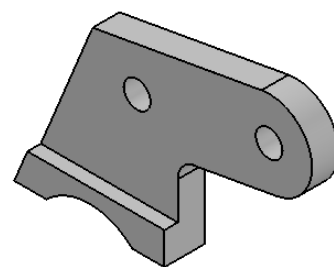
DETALLE C
ESCALA 5 : 1

	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Engranaje con Huequitos	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	5 - 2
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm			PESO:	ESCALA:1:1	HOJA 1 DE 1
					A4

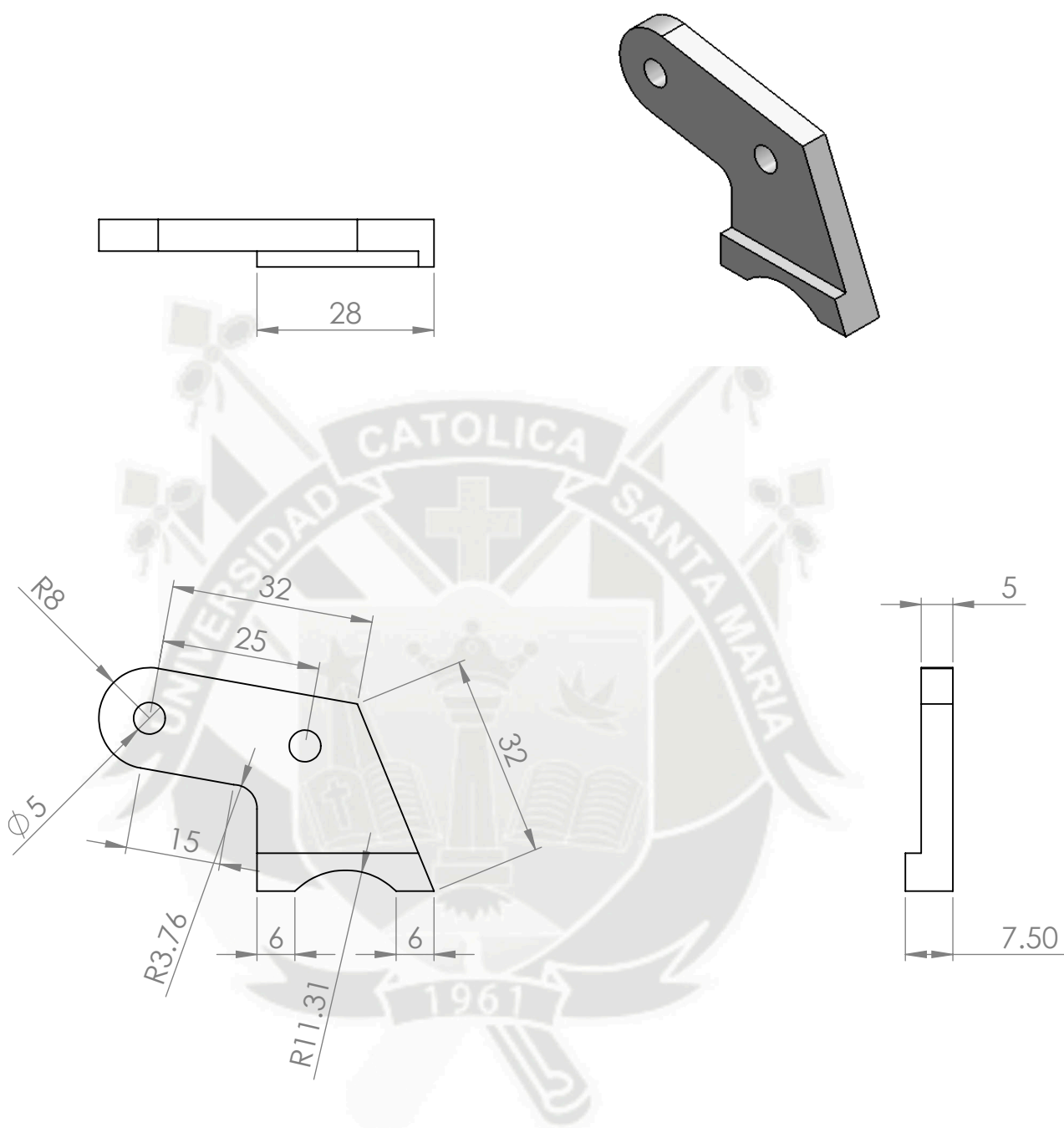


Módulo	1 mm
N° de Dientes	20 mm
Angulo de Presión	20 mm
Anchura de Cara	5 mm

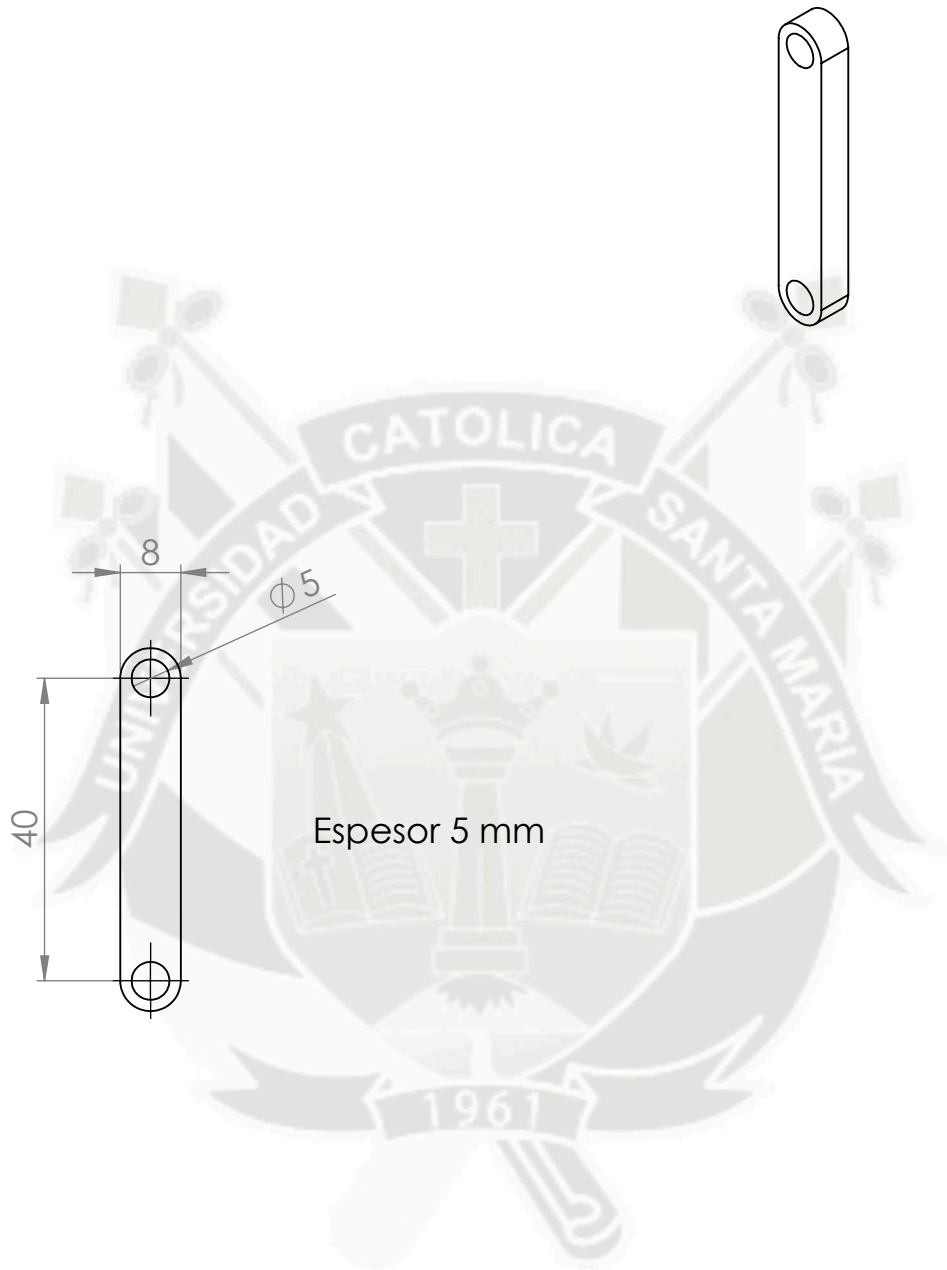
	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Engranaje sin Huequitos	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	5 - 3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA:1:1	HOJA 1 DE 1
					A4



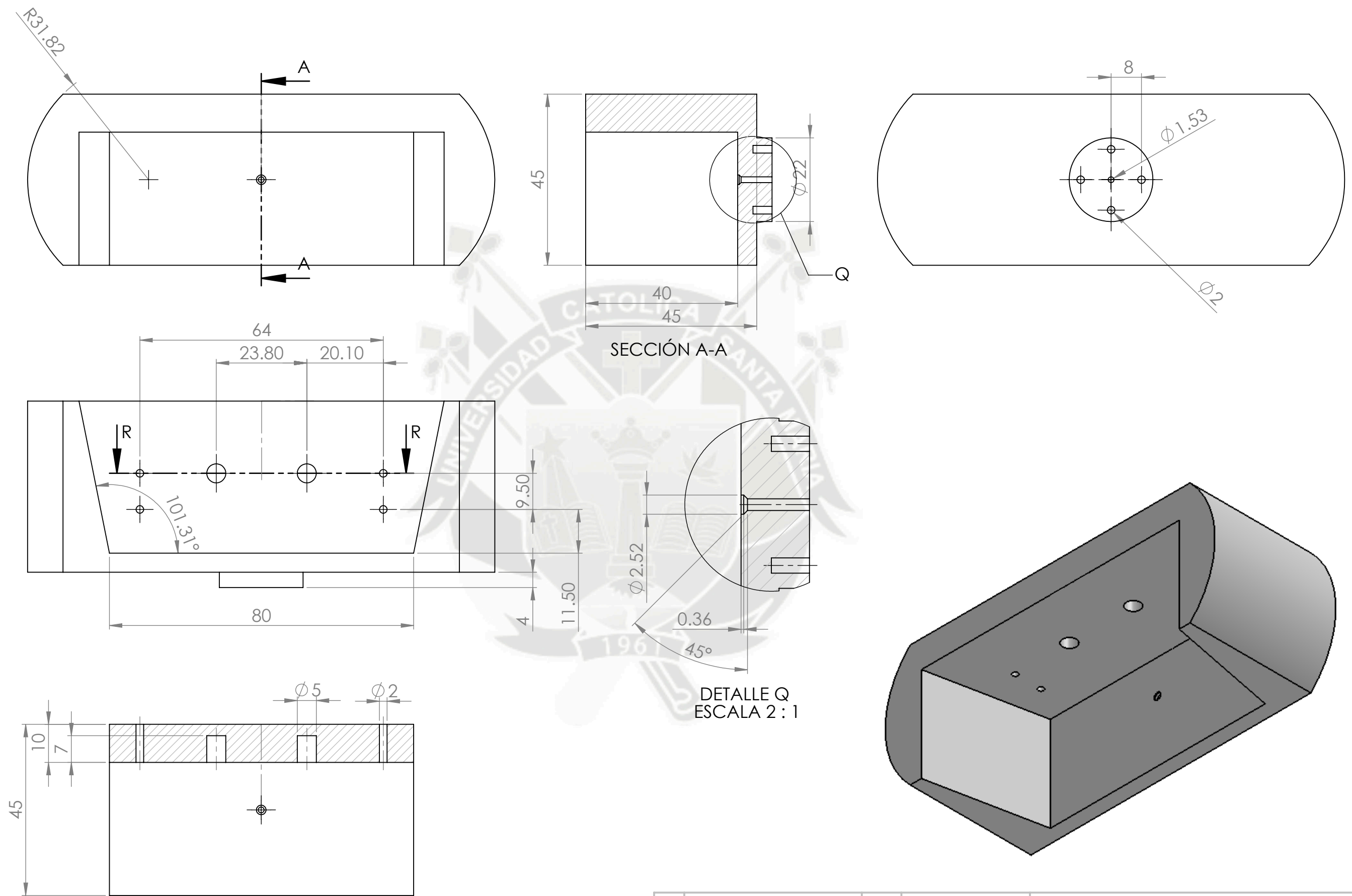
	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Garrita Final 1	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	N.º DE DIBUJO		A4
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		ABS	5 - 4		
		PESO:	ESCALA: 1:1	HOJA 1 DE 1	



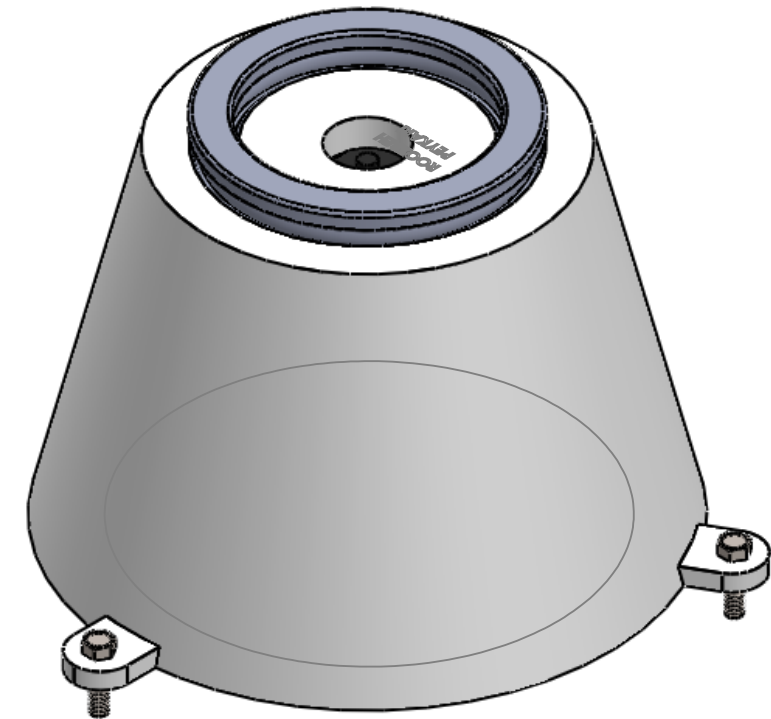
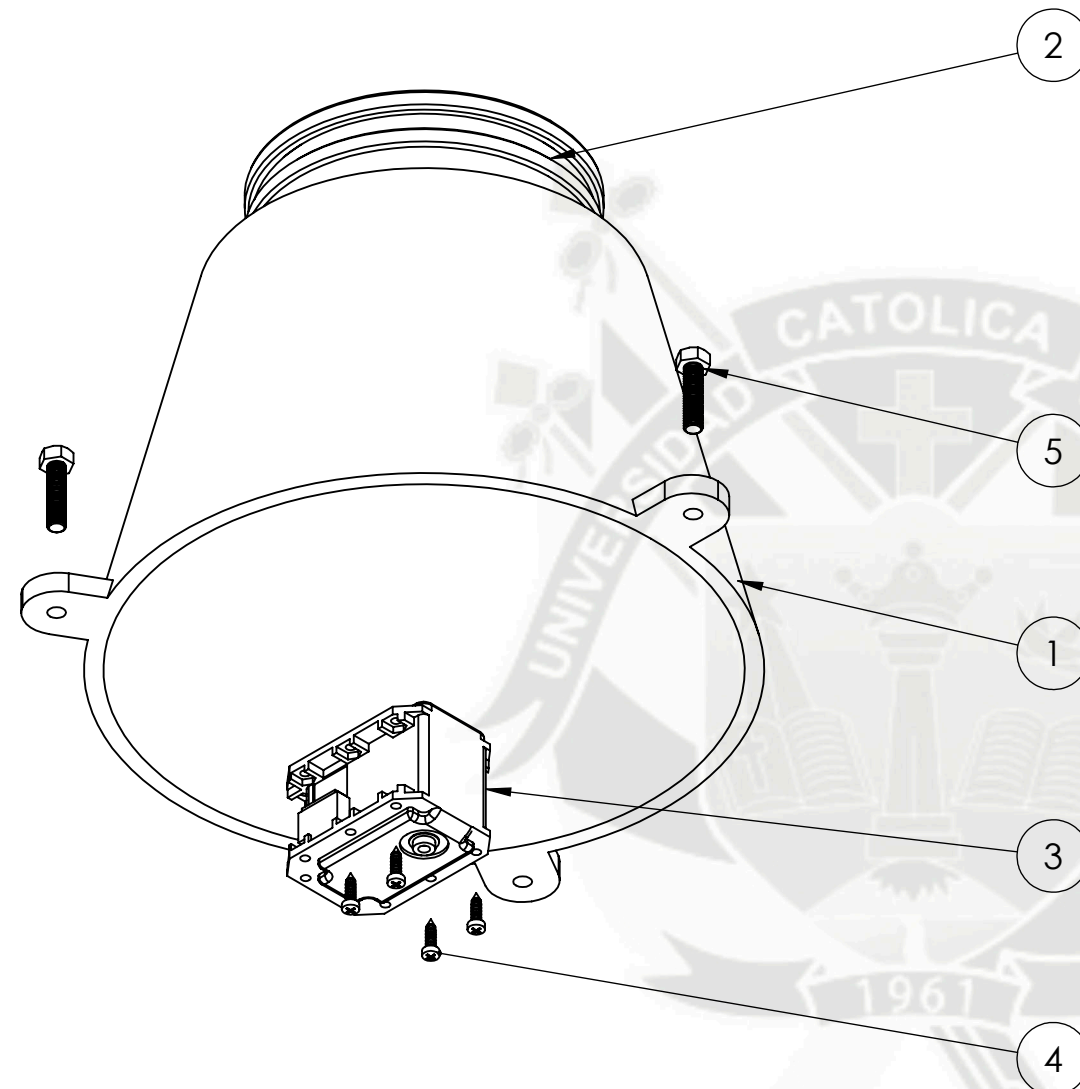
	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Garrita final 2	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	5 - 5
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA: 1:1	HOJA 1 DE 1



	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016	Palito de sujeción	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	N.º DE DIBUJO		A4
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		ABS	5 - 6		
		PESO:	ESCALA:1:1	HOJA 1 DE 1	

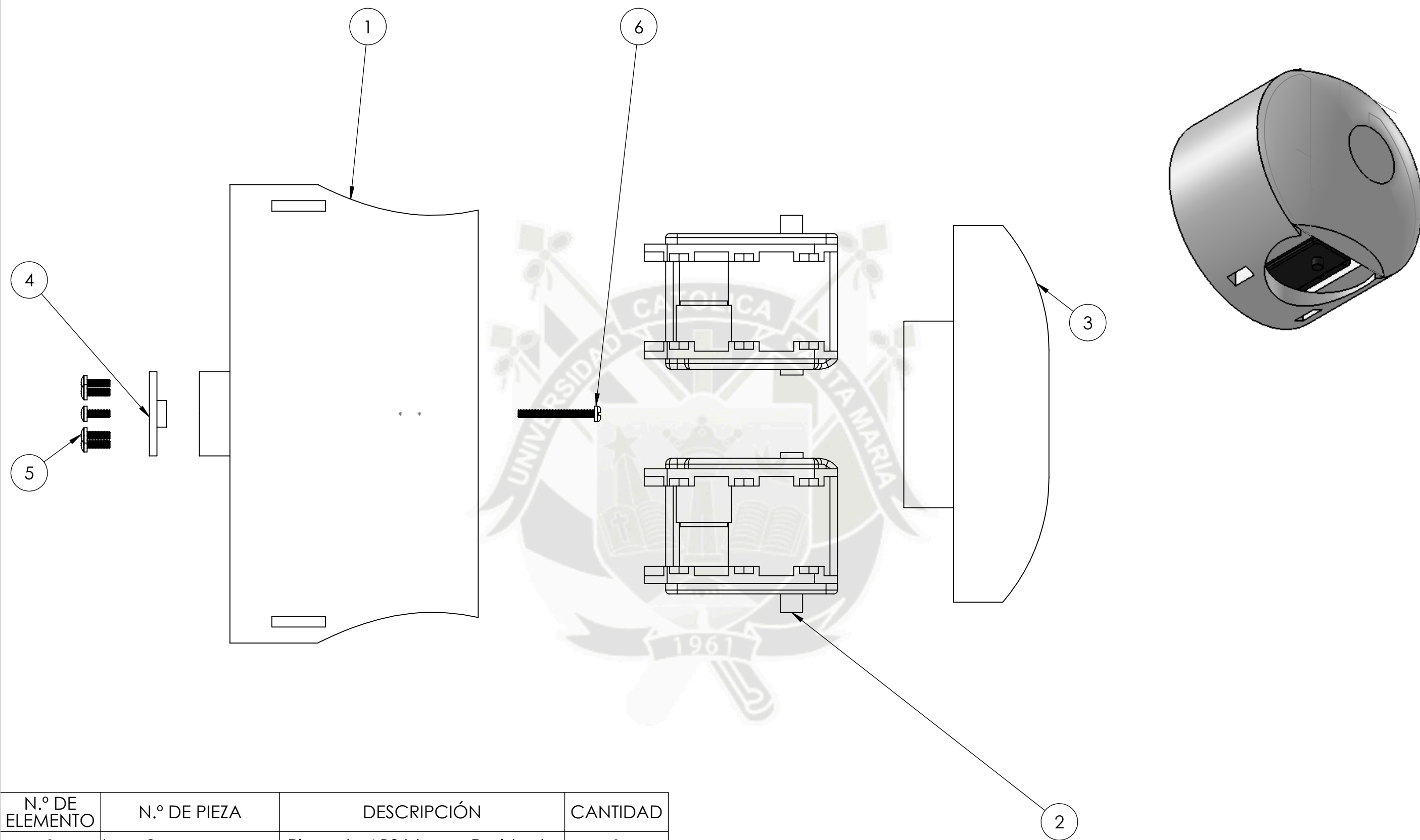


	NOMBRE	FIRMA	FECHA	TÍTULO:	Portagarra	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		23/04/2016			
VERIF.	BRAVO ROMERO, Bresly					
APROB.						
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	5 - 7	A4
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA:1:1	HOJA 1 DE 1	



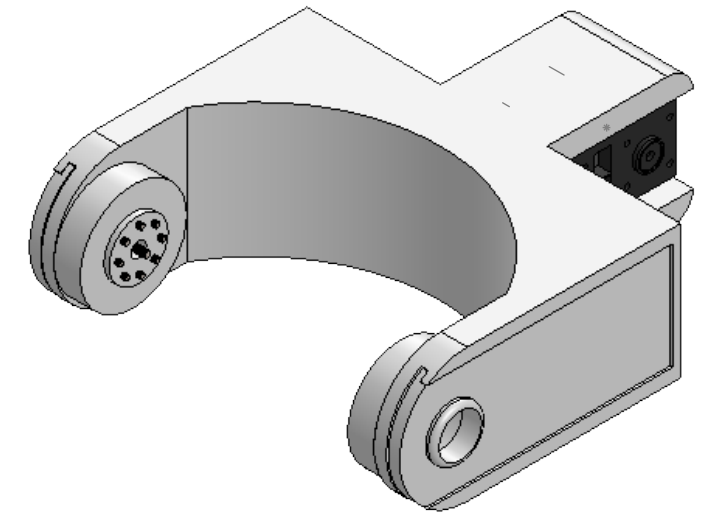
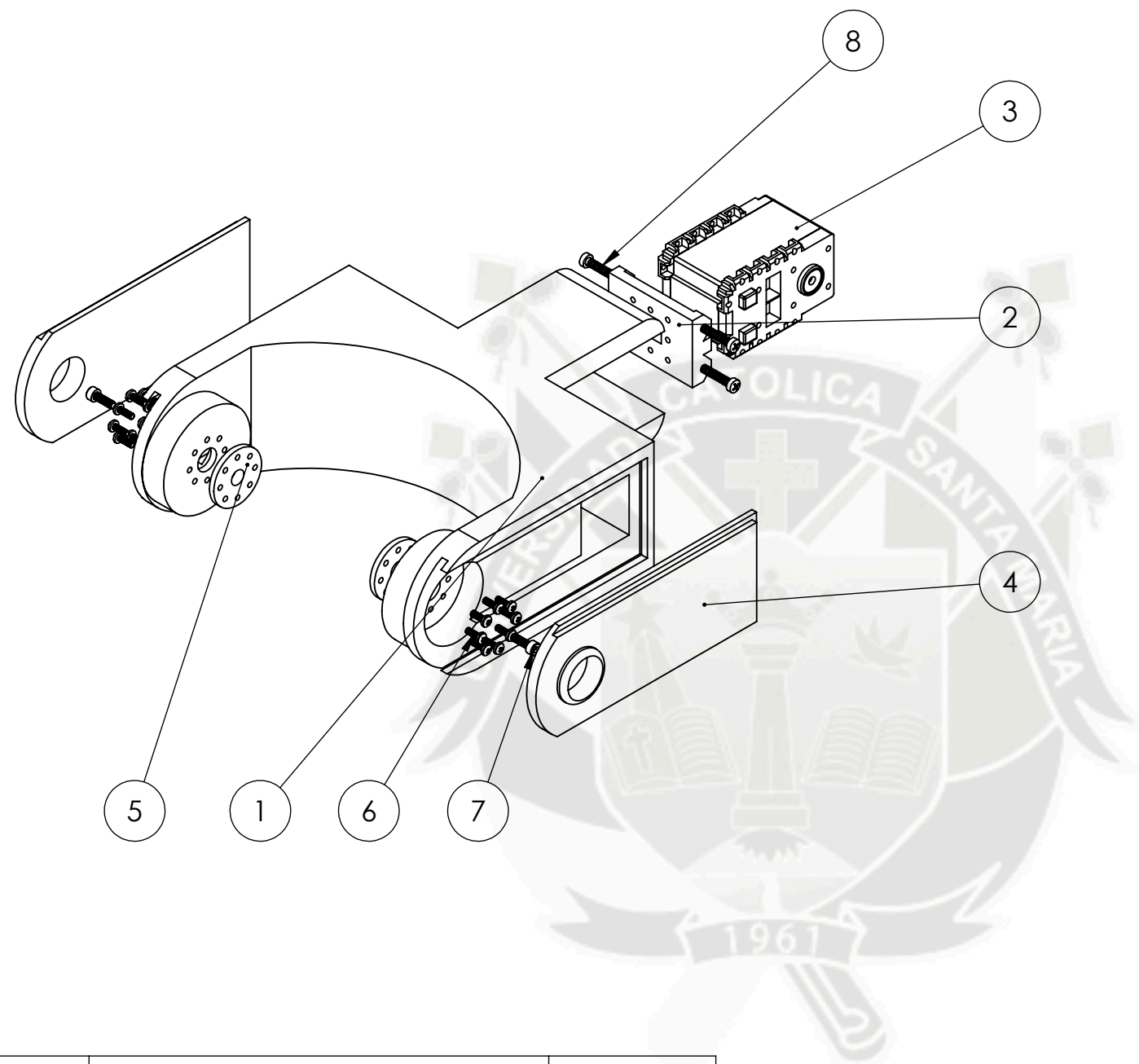
N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	base1	Pieza En Abs Blanco resistente	1
2	rodamiento SKF 51114_0	Rodamiento Axial de bolas	1
3	Dynamixel MX-28	Servomotor Robótico	1
4	M2 x 1x 10	tornillo del servomotor	4
5	M5 x 0.8 x 20	Perno para sujeción a base	3

NOMBRE	FIRMA	FECHA	TÍTULO:
VILLEGAS MEDINA, Jaime Enrique		15/05/2016	Subensamble 1
BRavo ROMERO, Bresly			
FABR. IMPRESION 3D	MATERIAL: ABS		N.º DE DIBUJO 6 - 1
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:	ESCALA:1:2
		HOJA 1 DE 1	



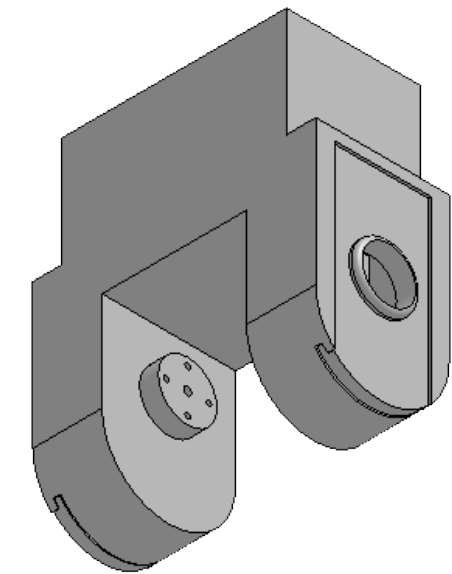
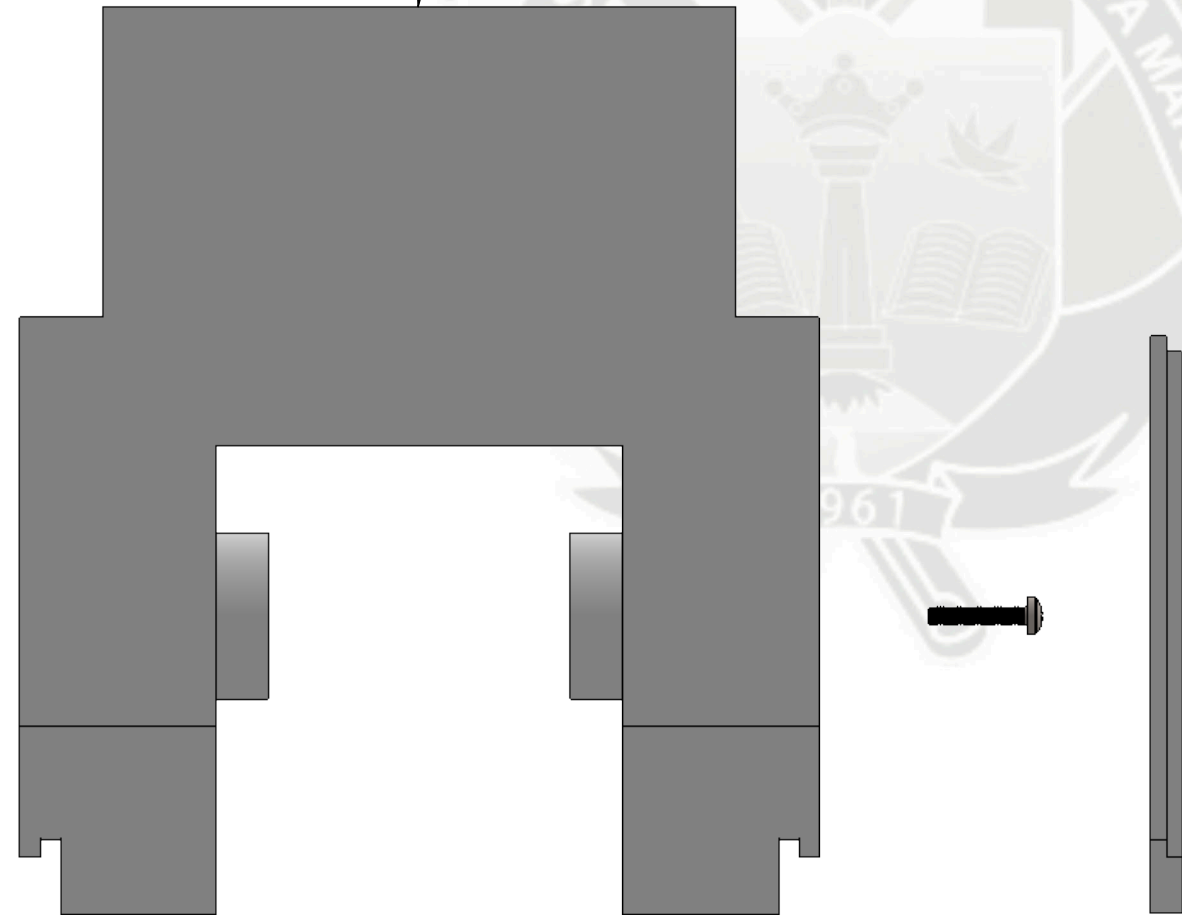
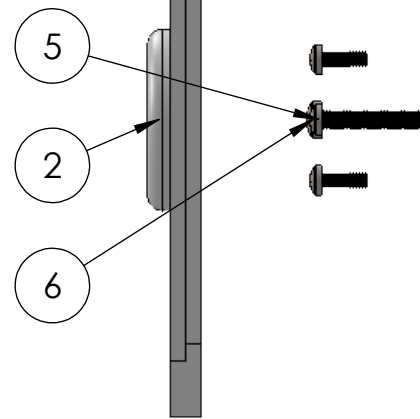
N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	base 2	Pieza de ABS blanco Resistente	1
2	RX-24f	Servomotor Dynamixel	2
3	ovnitapa	Tapa de protección de motores	1
4	circulo de union mx28	Pieza de Unión	1
5	B18.6.7M - M2 x 0.4 x 6 Type I Cross Recessed PHMS --6S	Tornillo de circulo de unión	8
6	B18.6.7M - M2 x 0.4 x 20 Slotted PHMS --20S	Tornillo de Unión	1

	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		16/05/2016	Subensamble 2	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D				
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm			MATERIAL:	ABS	
			PESO:		
				N.º DE DIBUJO	6 - 2
				ESCALA:1:2	HOJA 1 DE 1
					A3



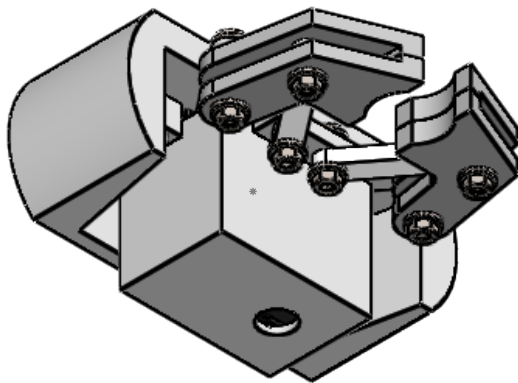
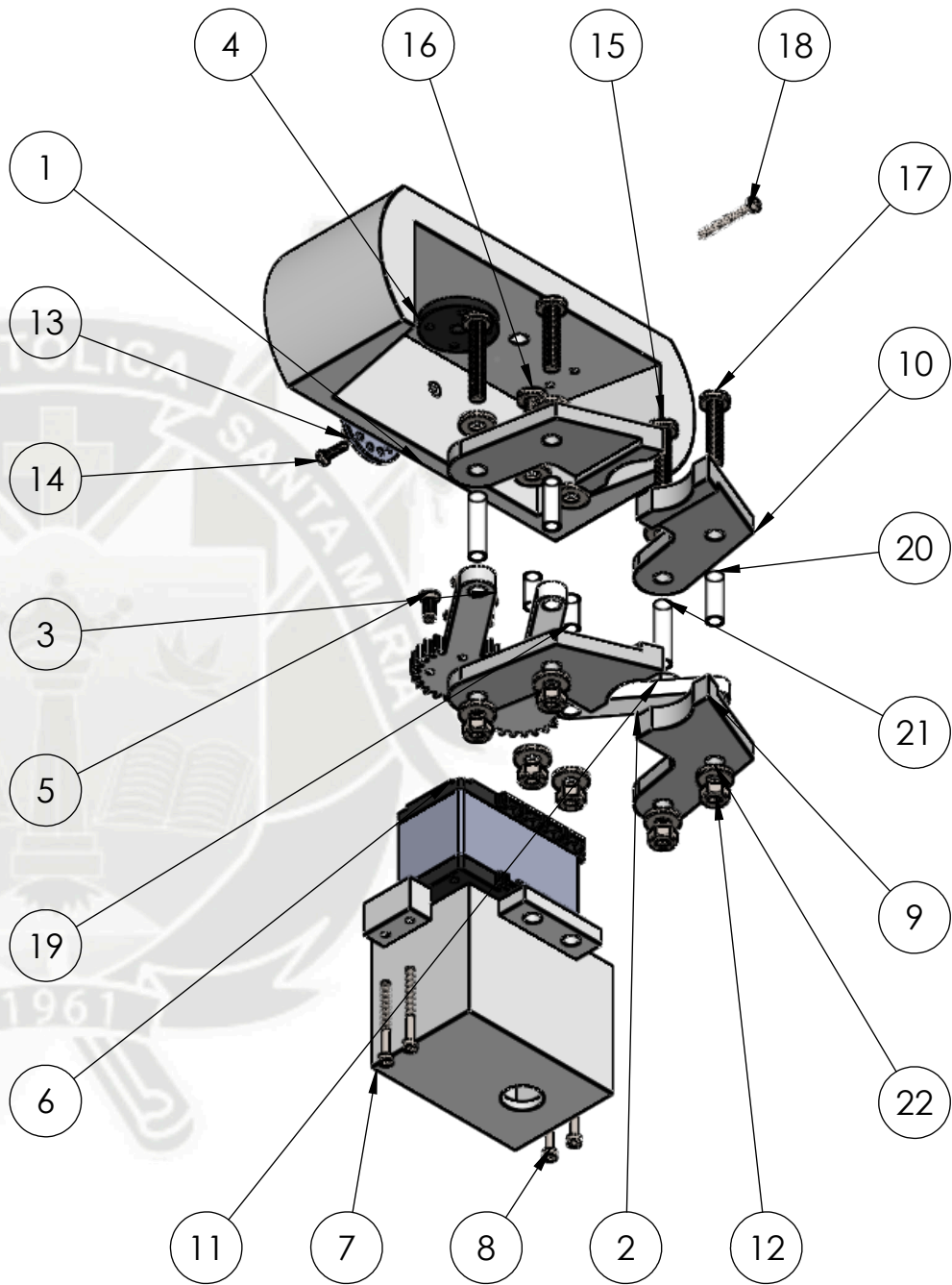
N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	eslabon1	Pieza de ABS blanco Resistente	1
2	soporte motor	Pieza de ABS blanco Resistente	1
3	motor eslabon 1	Servomotor Dynamixel	1
4	tapa de eslabon 1	Pieza de ABS blanco Resistente	2
5	circulo de union mx28	Pieza de Unión	2
6	B18.6.7M - M2 x 0.4 x 6 Type I Cross Recessed PHMS --6S	Tornillo de circulo de unión	16
7	B18.3.1M - 2.5 x 0.45 x 8 Hex SHCS -- 8SHX	Tornillo Allen	2
8	FHTS 0.112-24-type I- AB-0.8125-S	Tornillo de soporte de servomotor	8

	NOMBRE	FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		16/05/2016	Subensamble 3	
VERIF.	BRAVO ROMERO, Bresly				
APROB.					
FABR.	IMPRESION 3D	MATERIAL:		N.º DE DIBUJO	A3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		ABS		6 - 3	
PESO:		ESCALA:1:2		HOJA 1 DE 1	

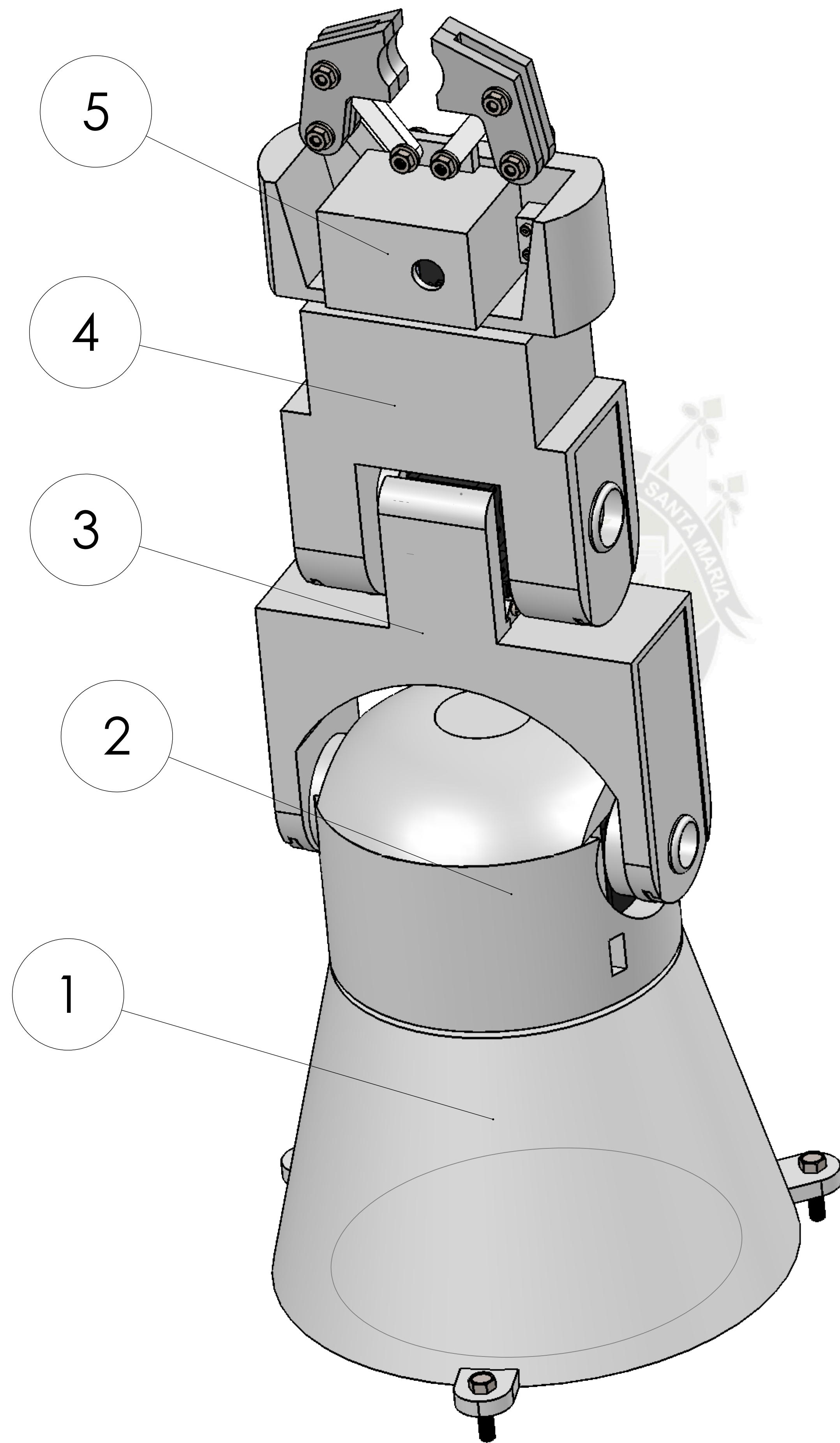


NOMBRE		FIRMA	FECHA	TÍTULO:	
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		16/05/2016	Subensamble 4	
	BRAVO ROMERO, Breslly				
VERIF.					
APROB.					
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	A3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm				6 - 4	
		PESO:	ESCALA:1:2	HOJA 1 DE 1	

	8	7	6	5		4	3	2	1
	N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD					
F	1	portagarra-casi-real (3)	Pieza de ABS blanco Resistente	1					
	2	engranaje-sin-huequitos-real	Pieza de ABS blanco Resistente	1					
	3	engranaje-con-huequitos-real	Pieza de ABS blanco Resistente	1					
	4	circulo-del-motor (3)	Pieza de ABS negro Resistente	1					
E	5	B18.6.7M - M3 x 0.5 x 8 Type I Cross Recessed FHMS --8S	Tornillo para sujección de engranaje	4					
	6	Ax - 12	Servomotor Dynamixel	1					
	7	cajita-del-motor (2)	Pieza de ABS blanco Resistente	1					
	8	B18.3.1M - 2 x 0.4 x 25 Hex SHCS -- 16SHX	Perno para sujeción de cajita	4					
	9	garrita final 2	Pieza de ABS blanco Resistente	2					
	10	garrita final 1	Pieza de ABS blanco Resistente	2					
	11	PALITO	Pieza de ABS blanco Resistente	2					
D	12	B18.2.4.1M - Hex nut, Style 1, M4 x 0.7 --D-S	tuerca para unión de garrita	6					
	13	circulo de union tower	Pieza de ABS blanco Resistente	1					
	14	tornillo 8 x 2	tornillo para círculo de TowerPro	2					
	15	B18.6.7M - M4 x 0.7 x 25 Type I Cross Recessed PHMS --25S	Tornillo para unión de garrita	2					
C	16	B18.6.7M - M4 x 0.7 x 13 Type I Cross Recessed PHMS --13S	Tornillo para unión de garrita	2					
	17	B18.6.7M - M4 x 0.7 x 20 Type I Cross Recessed PHMS --20S	Tornillo para unión de garrita	2					
	18	B18.6.7M - M2 x 0.4 x 20 Slotted PHMS --20S	tornillo para union entre eslabones	1					
	19	buje poliamida 10 mm	Pieza de Poliamida	2					
	20	buje poliamida 15 mm	Pieza de Poliamida	2					
B	21	buje poliamida 20 mm	Pieza de Poliamida	2					
	22	B18.22M - Plain washer, 3.5 mm, regular	Arandela para tornillos	12					



	NOMBRE	FIRMA	FECHA	TÍTULO:		
DIBUJ.	VILLEGAS MEDINA, Jaime Enrique		16/05/2016	Subensamble 5		
VERIF.	BRAVO ROMERO, Breslly					
APROB.						
FABR.	IMPRESION 3D	MATERIAL:	ABS	N.º DE DIBUJO	6 - 5	A3
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM TOLERANCIAS: 0.3 mm		PESO:		ESCALA:1:2	HOJA 1 DE 1	



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	Subensamble 1	Base 1	1
2	Subensamble 2	Base 2	1
3	Subensamble 3	Eslabón 1	1
4	Subensamble 4	Eslabón 2	1
5	Subensamble 5	Garra	1

NOVEMBRE	FEBRA	FECHA	TÍTULO
16/05/2016	Robot completo		
MATERIAL: ABS		N.º DE DIBUJO	A3
PESO:		ESCALA:1:2	HOJA 1 DE 1