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 ROBOT PARALELO TIPO DELTA DE 3 GRADOS DE LIBERTAD CONTROLADO POR MEDIO DE LÓGICA DIFUSA PARA LA CLASIFICACIÓN DE OBJETOS"

Tesis presentada por los Bachilleres:
Lazo Pinto, Arturo Alonso
Paz Pinto, José Augusto

para optar el Título Profesional de:
Ingeniero Mecatrónico

Asesor de Tesis:
Ing. Quispe Ccachuco, Marcelo

AREQUIPA – PERÚ

2018

Universidad Católica de Santa María

(51 54) 382038 Fax:(51 54) 251213 ucsm@ucsm.edu.pe http://www.ucsm.edu.pe Apartado:1350

AREQUIPA - PERU

ESCUELA PROFESIONAL DE INGENIERÍA MECANICA, MECÁNICA ELÉCTRICA Y MECATRÓNICA

INFORME DICTAMINATORIO

VISTO

EL BORRADOR DE TESIS TITULADO:

"DISEÑO E IMPLEMENTACION DE UN ROBOT PARALELO TIPO DELTA DE 3 GRADOS DE LIBERTAD CONTROLADO POR MEDIO DE LOGICA DIFUSA PARA LA CLASIFICACION DE OBJETOS"

Presentado por el Bachiller:

LAZO PINTO ARTURO ALONSO

PAZ PINTO JOSE AUGUSTO

Nuestro **DICTAMEN** es:

APROBADO

OBSERVACIONES: NINGUNA

Arequipa, 29 de agosto 2018

ING. SERGIO MESTAS RAMOS
1936

ING. MARCELO QUISPE CCACHUCO

"Tengan cuidado los que se sienten realizados, porque sin metas ni muros ya solo hay acantilados "

- ***Proof***

DEDICATORIA

A Mis padres Arturo e Himelda, por ser el pilar fundamental de lo que soy, por ser el ejemplo de perseverancia y responsabilidad que estoy orgulloso de seguir. A mis hermanos; Jennyfer, Fernando y Eduardo, que siempre han estado junto a mí brindándome su apoyo absoluto y demostrándome el amor incondicional a pesar de nuestras diferencias.

La memoria de mis abuelos, por haberme apoyado y corregido las veces que pudieron y por los consejos que me dejaron. Mi compañero de tesis, que pese a todas las dificultades por las que pasamos pudimos concluir con el presente proyecto de tesis. Mis amigos, David, Marcos, Christian, Italo, Johann, Gonzalo, Sergio entre otros por compartir tantos gratos momentos y por ser mi segunda familia.

Y Finalmente a todas esas personas especiales que me ayudaron a crecer como persona y que siempre serán importantes en mi vida.

Arturo Lazo Pinto

A mi madre Betty, por ser la persona que me supo apoyar en todo momento, por su constante motivación que me permitió mejorar para bien y por permitirme disfrutar de la música; el arte que más amo. A mi padre José, por sus ejemplos de perseverancia, sus grandes consejos sobre lo difícil que puede ser la vida y a enseñarme que nunca debo rendirme en todo lo que me propongo.

A mis hermanas Bettyna y Ximena, porque de alguna manera siempre me ayudaron y me aconsejaron sobre mis acciones y actitudes; a mi Tía Mery por siempre escucharme y estar pendiente de mí; a mí enamorada Abigail por acompañarme; escucharme, animarme en los momento más difíciles y por todos los buenos momentos; a mis amigos Dennis, Marco, D'angelo y Daniel por los más de 13 años de amistad y para que vean en mi un ejemplo a seguir. A mis amigas Claudia y Alejandra por siempre apoyarme, escucharme y por ser las personas que siempre estarán cuando las necesite; y a Arturo, con quien a pesar de nuestras diferencias, fue posible realizar este proyecto.

José Paz Pinto

AGRADECIMIENTOS

Agradezco a Dios por haberme dado la fortaleza necesaria para poder cumplir mis proyectos, por haberme dado la oportunidad de tener personas tan maravillosas en la vida.

Agradezco a mis padres Arturo e Himelda por brindarme su apoyo incondicionalmente, por su esfuerzo para poder tener educación superior de calidad, por su confianza en mí y sobre todo por los consejos que me han dado a lo largo de la vida.

Agradezco a mi Universidad y a todos mis docentes en especial a los Ingenieros Sergio Mestas Ramos, Marcelo Quispe Ccachuco y Juan Cuadros Machuca, que me dieron los conocimientos necesarios para poder realizar el presente proyecto de tesis.

Arturo Lazo Pinto

Agradezco a mis padres por todo el apoyo brindado a lo largo de la realización del presente proyecto de tesis y por permitirme alcanzar la oportunidad de obtener un título profesional; a la familia Lazo Pinto por toda su hospitalidad mostrada mientras me quedaba en su hogar.

Agradezco a todos los profesores por su conocimiento y experiencia que nos brindaron a lo largo de los 5 años de universidad.

Agradezco a Bettyna por las imágenes renderizadas del robot y a Abigail por la corrección de las normas.

Agradezco a mis compañeros Christian Pinto, Yonathan Vera y Marcos Onque por compartir gratos momentos; y a Arturo por permitirme tener el control total sobre la música que escuchábamos mientras realizábamos el proyecto.

José Paz Pinto

INTRODUCCIÓN

La robótica es una rama de la ciencia que de manera progresiva ha ido tomando mayor relevancia tanto en el campo académico como en el sector industrial, esto ha genero la creación de distintos tipos de robots y con ello se elevó los estándares de calidad de las distintas empresas que se vieron obligada a la automatización de sus procesos, ya que la idea de que un mecanismo realice diversas tareas de manera más precisa, eficiente y rápida que un ser humano, resulta más atractivo.

Un manipulador robótico puede ser usado para diversos procesos tales como transporte, clasificación de objetos, ensamblaje de piezas, trabajos de soldadura y muchos otros procesos que necesitan un estándar de calidad alto que cumpla con las expectativas de la industria.

Actualmente existe una gran variedad de robots o manipuladores robóticos capaces de realizar la tarea de clasificar diversos objetos dependiendo de sus configuraciones como por ejemplo un manipulador robótico tipo SCARA, robots cartesianos y robots articulados. Sin embargo, la mayoría de estos robots se basan en una topología similar a la de un brazo humano. Aunque sus diseños han ido mejorando con el paso del tiempo, estos cuentan con algunas desventajas relacionadas a su morfología que resulta inadecuados para esta aplicación, entre ellos la limitada capacidad de carga, su poca capacidad de compensar las perturbaciones y su limitada velocidad ya que si esta fuera elevada podría generar grandes inconvenientes en el área de trabajo además de un riesgo al personal.

Teniendo en cuenta estas desventajas se plantea una configuración que soporte dichos problemas mediante el uso de actuadores que trabajen en paralelo, de esta manera se puede compensar la perturbación, la limitada capacidad y velocidad, también se puede obtener un robot más liviano y mayor capacidad de cargar utilizando actuadores de menor potencia. Estos son los robots paralelos.

Uno de los aspectos más importantes en el diseño de un robot, es el tipo de control que se va a emplear, la mayoría de robots emplea un control convencional (PID), debido a que es la estrategia de control más usada y en la que su investigación se conoce con más detalle, pero a pesar de que es el más empleado cuenta con ciertos inconvenientes, uno de ellos y el más problemático es el modelamiento cinemático y dinámico del robot el cual se torna más complejo a medida que aumenten los grados de libertad. Considerando la complejidad que conlleva realizar el modelamiento cinemático y dinámico surgen otras estrategias de control en las que no se considera dicho modelamiento de manera analítica tal como el Control Difuso o Fuzzy Logic.

RESUMEN

El presente proyecto de tesis consiste en el diseño e implementación de un robot paralelo delta de 3 GDL que emula el movimiento básico de la clasificación de objetos, controlado por medio de una estrategia de control difuso.

Para llevar a cabo el proyecto se realizó una serie de análisis que contemplan estudios en la cinemática, dinámica, mecánica, electrónica, software y control relacionado al robot. Así mismo, el diseño y selección de un controlador difuso adecuado se basó en el comportamiento real del robot adquirido mediante diversas evaluaciones.

Se consiguió crear programas en lenguaje estructurado que permitan ejercer distintas funciones como ejecución de instrucciones de movimiento y adquisición de datos de manera continua.

Palabras Clave: Paralelo, Robot, Lógica Difusa, GDL, Control.

ABSTRACT

The present thesis project consists of the design and implementation of a 3 DOF delta parallel robot that emulates the basic movement of object classification, controlled by a fuzzy control strategy.

To carry out the project, a series of analyzes was carried out that contemplate studies in the kinematics, dynamics, mechanics, electronics, software and control related to the robot. Likewise, the design and selection of a suitable fuzzy controller was based on the actual behavior of the robot acquired through various evaluations.

It was possible to create programs in structured language that allow to exercise different functions such as execution of movement instructions and data acquisition in a continuous way.

Keywords: Parallel, Robot, Fuzzy Logic, DOF, Control.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I:

1. MARCO METODOLÓGICO

1.1. Objetivos

1.1.1. Objetivo Principal

Diseñar e implementar un robot paralelo de 3 Grados de Libertad tipo delta controlado mediante lógica difusa que cumpla con el principio básico de la clasificación de objetos.

1.1.2. Objetivos Específicos

- Diseñar la estructura mecánica del robot paralelo de manera que sea liviana, robusta, fácil de transportar y de instalar.
- Crear un sistema de control que cumpla con las instrucciones dadas al robot y que sea capaz de posicionarse sobre un punto en el espacio de la mejor manera posible.
- Implementar un efector final que cumpla con el propósito propuesto.
- Desarrollar una interfaz hombre-máquina que permite la interacción con el robot paralelo de manera continua.

1.2. Justificación

El desarrollo de un módulo experimental que involucre a un robot paralelo dentro de su arquitectura, es material de estudio en el área de control y de la robótica, esto se debe a la naturaleza no lineal, a su cinemática compleja y a su comportamiento dinámico.

Basándonos en clasificación de objetos existen diferentes tipos de configuraciones robóticas, pero la que más sobresale es la del robot paralelo; debido a que la poca masa de sus eslabones le permite realizar maniobras con gran velocidad y al ser de configuración paralela le permite

tener una mayor capacidad de carga, ya que se usan actuadores de menor potencia para cada articulación y esto resulta eficiente.

Con el fin de apoyar el desarrollo del control inteligente sobre los sistemas robóticos, el control difuso se presenta como una herramienta innovadora en sistemas robóticos, que permitiría el control en distintas aplicaciones. Este control se torna muy valioso, cuando el modelamiento dinámico se torna muy complejo.

1.3. Descripción del Problema

Hoy en día, la mayoría de los robots que se emplean en casi cualquier proceso están diseñados a partir de configuraciones convencionales, debido al amplio estudio que se tiene de estas, sin embargo, también se existe la configuración en paralelo, la cual no es tan estudiada a pesar de las ventajas que esta proporciona para ciertas aplicaciones.

Si se desea adquirir un robot paralelo resultaría dificultoso, ya que este tipo de robot no se vende en el mercado local y adquirir uno del extranjero resultaría ser una tarea tediosa, que conlleva tiempo.

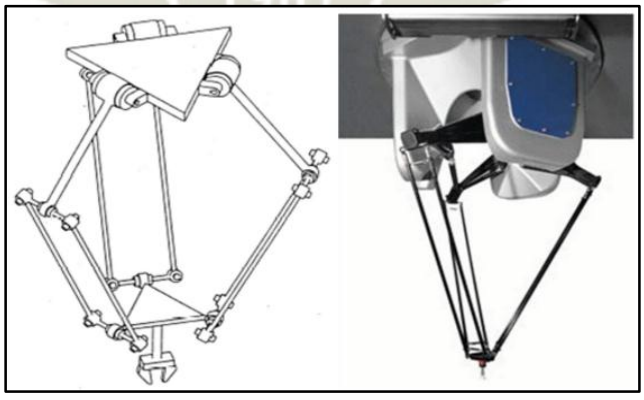


Figura 1.1. Estructura Delta propuesta por Clavel y su versión industrial
***Fuente**: (Ballané, 2017)*

Esto hace que diseñar e implementar un robot paralelo resulte ser la opción más adecuada, pero, aun así, el modelamiento y el control del robot paralelo

se tornan bastante complejo debido al comportamiento fuertemente no lineal que este conlleva, lo cual requeriría de mayor tiempo. Es por esto que se también se busca demostrar que es posible adecuar un robot paralelo y controlarlo con alguna otra estrategia que no resulte compleja, como lo es el control difuso, ya que, al tener un modelo dinámico complejo, usar una estrategia de control convencional resultaría dificultoso. En cambio, con un control difuso se puede dar solución a este gran problema ya que este no requiere de su modelo dinámico, pero sí de su comportamiento físico.

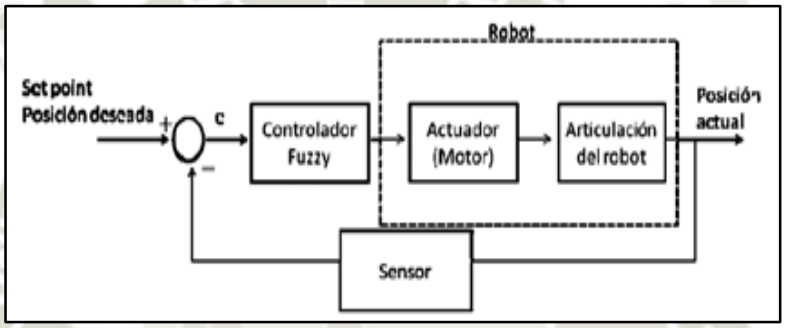


Figura 1.2. Diagrama de Bloques de un Controlador Difuso
Fuente: *(Tibaduiza, Amaya, Rodríguez, Mejia, & Flórez, 2011)*

1.4. Antecedentes

1.4.1. Antecedentes Locales

Ruiz Daza, Carlo Franco (Arequipa, 2014) “CONTROL KINECT DE UN BRAZO ROBÓTICO DE 5 GDL A TRAVÉS DE UNA INTERFAZ INTUITIVA”

El presente trabajo da una definición del robot clasificación de los robots, clasificación según la generación, clasificación según el área de aplicación, clasificación según el número de ejes, clasificación según la configuración y clasificación según el tipo de control morfología del robot. Diseñar la estructura mecánica de un robot. Seleccionar las transmisiones, actuadores, sensores internos, elementos terminales

métodos de localización espacial. Representación de la posición y orientación. Además de calcular las matrices de transformación homogénea, relación entre ángulos de Euler y matrices homogéneas, cinemática del robot, cinemática inversa, cinemática diferencial, dinámica del robot, control dinámico del robot. Implementar sensor Kinect y las interfaces naturales de usuario, diseño de la interfaz intuitiva pruebas, resultados y análisis.

1.4.2. Antecedentes Nacionales

Orihuela Rojas, Marcelo Miguel (Lima, 2015) "DISEÑO DE UN MECANISMO PARALELO TIPO PLATAFORMA DE SEIS GRADOS DE LIBERTAD DE APOYO MÓVIL PARA UN SIMULADOR DE ENTRENAMIENTO DE CONDUCTORES DE AUTOMÓVILES"

El presente trabajo muestra el diseño de un mecanismo paralelo del tipo plataforma de seis grados de libertad y apoyos móviles que puede ser utilizado en un simulador de automóviles para el entrenamiento de conductores en nuestro país.

El objetivo está centrado en lograr el diseño mecánico dejando pase a futuras investigaciones como el análisis de simulación mecánica, el diseño del sistema de control del mismo, entre otros que puedan complementar y optimizar el diseño de un simulador de manejo automatizado.

1.4.3. Antecedentes Internacionales

André Olsson (India, 2009) "MODELING AND CONTROL OF A DELTA-3 ROBOT"

El objetivo de la tesis es definir, analizar y verificar a través de simulaciones e implementaciones prácticas, robots paralelos con articulaciones no convencionales que les permitan ser subaccionados y / o reconfigurables. Veremos cómo, a través de ciertas transformaciones geométricas, algunas de las juntas estándar pueden ser reemplazadas por juntas bloqueables.

El beneficio esperado de estos nuevos diseños es obtener robots paralelos con: espacios de trabajo más grandes porque se reduce la posibilidad de colisiones entre piernas, y también se reduce el número de articulaciones (con sus limitaciones de rango intrínseco).

El objetivo de esta tesis es definir, analizar y verificar, mediante simulaciones e implementaciones prácticas, robots paralelos con articulaciones no convencionales con el fin de incorporar propiedades de sub-actuación y reconfigurabilidad.

1.5. Alcances y Limitaciones

El presente proyecto contempla únicamente el diseño y la implementación de un robot paralelo de 3 GDL, esto quiere decir que se realizaran estudios cinemáticos y dinámicos, como también se realizara un estudio a la estructura mecánica para así demostrar lo aprendido referente al área de diseño mecánico. También se enfocará en el desarrollo de un sistema de control difuso que permita realizar los movimientos y las acciones de robot de una manera rápida y eficaz basándose en el comportamiento de este.

El proyecto será donado a la Universidad Católica de Santa María, al laboratorio de Robótica de la escuela profesional de Ingeniería Mecánica,

Mecánica Eléctrica y Mecatrónica, el cual podrá ser usado con fines académicos y ser material de estudio.

Para darle un valor agregado al trabajo, se desarrollará una plataforma en la cual se demuestre el funcionamiento del robot para fines didácticos e industriales ya que portara una arquitectura abierta lo cual permitirá establecer nuevos esquemas de control o programas en torno a la aplicación.

El espacio de trabajo del robot paralelo delta estará limitado por las articulaciones mecánicas que serán usadas en su implementación.

Capitulo II:

2. MARCO TEÓRICO

2.1. Antecedentes Históricos

Desde el comienzo de la creación de nuevos dispositivos mecánicos, el hombre siempre se sintió encantado por uno que sea capaz de realizar acciones similares a las que un ser vivo es capaz de hacer. Podemos reconocer ciertos puntos a lo largo de la historia en los que se desarrolló de cierto modo la robótica hasta lo que conocemos actualmente, a continuación, los describiremos superficialmente.

Desde la antigua Grecia a estos mecanismos le denominaron autómatas, de donde deriva la palabra que ahora conocemos como autómata. Un autómata es un mecanismo que tiene un aspecto similar a la de un ser vivo y además este es capaz de realizar movimientos que son bastante parecidos a los que realiza el ser vivo. Recordamos así los mecanismos animados de Heron de Alejandría (85 d.C.) que se movían a través de dispositivos hidráulicos, poleas y palancas con fines totalmente lúdicos.

En la cultura árabe se heredaron los conocimientos griegos pero esta vez no solo fueron usados como mecanismos de diversión si no desde una aplicación práctica en la vida cotidiana de la realeza, donde se utilizaron sistemas automáticos de agua y lavaderos. En esta época hay referencias de autómatas, que, aunque no se tiene suficiente documentación, es sabido que por ejemplo el hombre de Hierro de Alberto Magno o la cabeza parlante de Roger Bacon fueron conocidos, así como el Gallo de Estamburgo que es el autómata del que tenemos referencia que aún se conserva en la actualidad, ya que formaba parte del reloj de la Catedral de Estamburgo donde al dar las horas movía las alas y el pico (Martín, y otros, 2007).

Durante el siglo XV y XVI Leonardo da Vinci construye el conocido León Mecánico para el rey Luis XII de Francia que abría su pecho con una garra y postraba el escudo de armas del rey. En España el Hombre de Palo construido por Juanelo Turriano en el siglo XVI para el emperador Carlos V era un autómata en forma de monje que andaba, movía los brazos, ojos y boca.

Ya en el siglo XVII y XVIII se empezaron a crear diseños mecánicos que tenían algunas características de los robots actuales. Estos dispositivos en su mayoría los crearon artesanos del gremio de la relojería, la misión que tenían estos era el de entretener en las ferias y la gente de la corte. Jacques Vaucanson también crearía el primer telar mecánico, con diversos muñecos animados, en los que destacan la recordada Flautista capaz de tocar varias melodías y el pato que comía, se movía e incluso evacuaba la comida. Pierre Jaquet Droz y sus hijos Henry-Luis y Jaquet construyeron varios muñecos capaces de escribir, dibujar y tocar diversas melodías en un órgano. Aún se encuentran en el museo de Arte e Historia de Neuchastel, Suiza. Casi contemporáneos con los anteriores muñecos mencionados, Henry Mailladert construyo una muñeca que era capaz de dibujar, que aún se conserva en Filadelfia.

A finales del siglo XVIII y comienzos del XIX se desarrollaron diversas invenciones mecánicas, utilizadas fundamentalmente en la industria textil, en la que destaca la hiladora giratoria de Hargraves, la hiladora mecánica de Crompton, el telar mecánico de Cartwrigth y el telar de Jacquard. Este último utilizaba una cinta de papel perforada como un programa para las acciones de la máquina. Es desde esta época en la que podemos decir que

se da el inicio de la automatización industrial. (Peñaranda, Prada, Calle, & Fernandez, 2015)

2.2. Definición de Robot

La palabra robot fue usada por primera vez en el año 1921 cuando el dramaturgo Karel Capek estreno en el teatro Nacional de Praga su obra Rossums Universal Robot. El origen de esta palabra eslava robota, se refiere al trabajo realizado de una manera forzada. (González V. , Introducción Robótica, 2002)

La definición más completa es la establecida por la Asociación Francesa de Normalización (AFNOR) donde Robot se define como 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 control incluye un dispositivo de memoria y ocasionalmente de percepción de su entorno. Normalmente su uso es el de realizar una tarea de manera cíclica, pudiéndose adaptar a otra sin cambios permanentes en su material. (González V. , Robots Industriales, 2002)

2.3. Clasificación de los Robots

Los robots pueden ser clasificados de distintas maneras teniendo distintos criterios de clasificación ya sea por la función que realizan, la arquitectura del mismo, a continuación de exponen algunas de ellas:

2.3.1. Clasificación según La Generación:

En la clasificación según la generación de un robot se toma en cuenta el momento tecnológico en el que el robot fue desarrollado, es por esto que cada vez que se tiene un hito significativo se puede decir que se tiene otra generación de robots.

La tabla 2.1 recopila una clasificación según la generación, la cual se divide hasta el momento en tres, se puede decir que la primera generación abarca desde el comienzo de la robótica hasta los años ochenta. La segunda generación se extiende desde los años ochenta hasta la actualidad y finalmente la tercera generación es la que se encuentra en pleno desarrollo. (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

Tabla 2.1 Clasificación de los Robots según la Generación

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: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

2.3.2. Clasificación según el Área de Aplicación

Para la clasificación del robot según el uso que se le dará nos basamos en la clasificación de la IFR (*International Federation of Robotics*), la cual se muestra en la tabla 2.2, la que mostramos a continuación.

Tabla 2.2 Clasificación según la IFR

Clasificación según la IFR
000 Sin especificar
110 Manipulación en fundición
130 Manipulación en moldeo de plásticos
140 Manipulación tratamientos térmicos
150 Manipulación en la forjado 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: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

2.3.3. Clasificación según su Configuración

La clasificación según su la configuración de un robot se determina por el tipo de movimientos permitidos que tendrán los dos o más eslabones continuos de la cadena.

- **Cartesiano:** Su 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**: Cuenta con dos articulaciones rotacionales y una lineal.
- Articular o antropomórfico: Cuenta con tres o más articulaciones rotacionales.
- **SCARA**: que posee varios tipos de articulaciones, combinaciones de las anteriores.
- **Paralelo**: Posee brazos con articulaciones prismáticas o rotacionales.

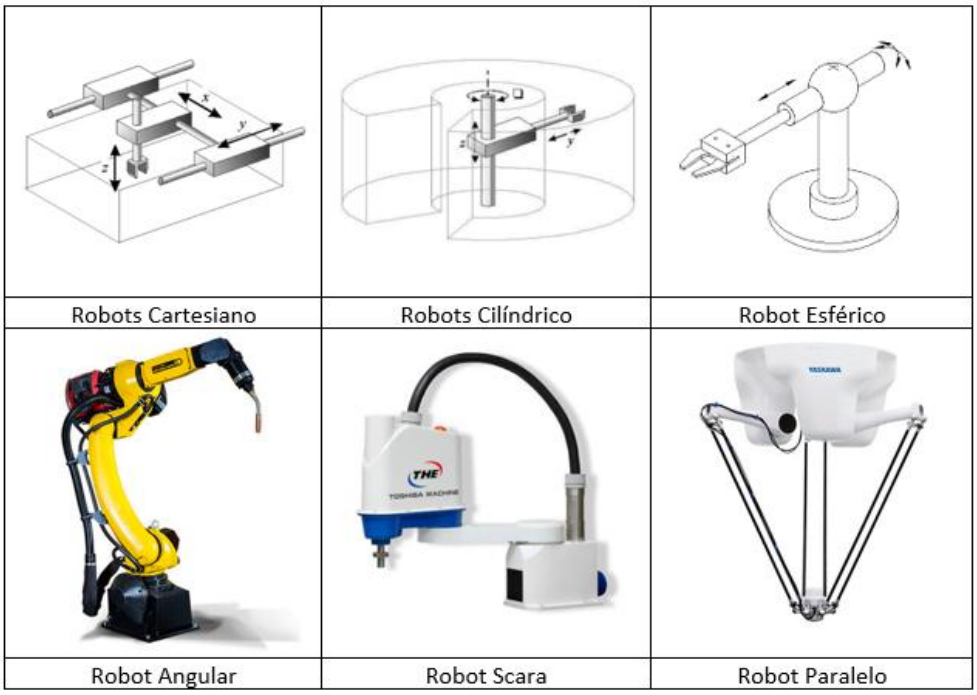


Figura 2.1. Clasificación de Robots según su Configuración.
Fuente: *Elaboración Propia.*

2.4. Morfología

Un robot está conformado por las siguientes partes: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de control y elementos terminales. La Figura 2.2 muestra la estructura mecánica de un robot.

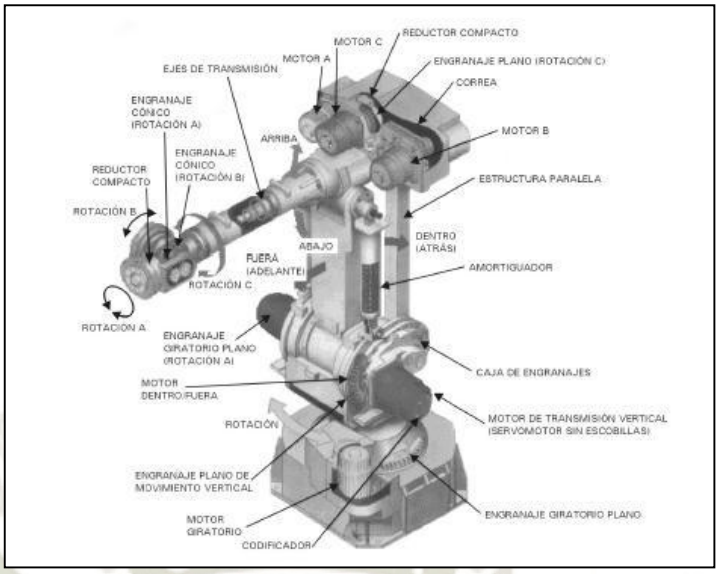


Figura 2.2. Estructura Mecánica de un Robot.
***Fuente**: (Ake, 2014)*

2.4.1. Estructura Mecánica

La estructura mecánica abarca una serie de eslabones unidos mediante articulaciones, las cuales permiten un movimiento de los eslabones. La mayoría de veces la unión de estos eslabones se asemeja bastante a la anatomía que posee un brazo humano, es por esto que se suele denominar a las articulaciones como: codos, muñecas, brazos. Existen distintos tipos de articulaciones, los que bien dados por el tipo de movimiento que estas ofrecen, se pueden distinguir 6 distintos tipos de articulaciones los que se muestran en la Figura 2.3.

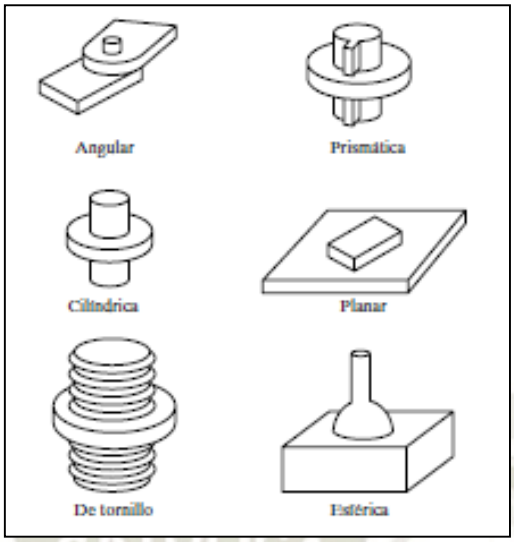


Figura 2.3. Tipos de Articulaciones.
Fuente: *(Craig, Cinemática de manipuladores, 2006)*

El conjunto de articulaciones con eslabones se denomina cadena cinemática, la cual puede ser cadena cinemática abierta o cadena cinemática cerrada, como se observa en la Figura 2.4. El empleo de las distintas combinaciones de articulaciones da como resultado distintas configuraciones.

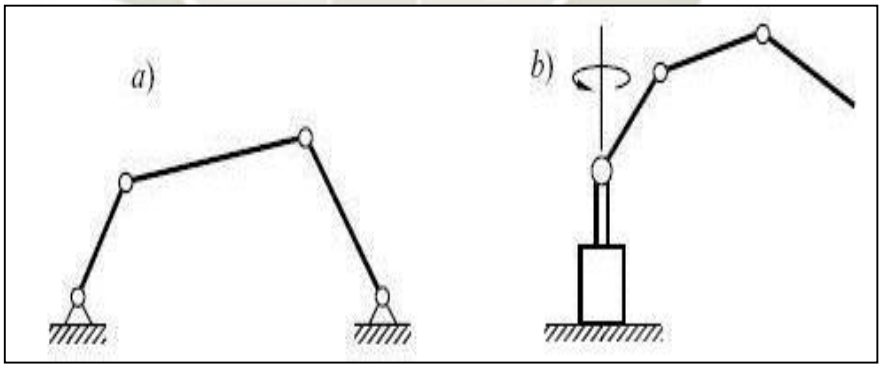


Figura 2.4 Cadenas Cinemáticas: a) Cerrada, b) Abierta.
Fuente: *(Mecánica, 2017)*

Cada movimiento independiente que una articulación pueda generar con respecto a la anterior se denomina grado de libertad. El número de GDL de un robot se determina por distintos métodos, por ejemplo el método de Gruebler.

2.4.2. Transmisiones

Las transmisiones son las encargadas de transmitir el movimiento de los actuadores hacia las articulaciones, un robot mueve muchas veces mueve el extremo a elevadas aceleraciones y también para realizar el movimiento se tiene que vencer pares estáticos, lo cual genera inercias, por esto las articulaciones tienen que soportar tal inercia. Las transmisiones también se utilizan para cambiar un movimiento circular a uno lineal o viceversa lo que a veces es de mucha utilidad. Un buen sistema de transmisión debe de cumplir algunos criterios; tener peso y tamaño reducido, debe evitar tener holguras considerables y finalmente que cuenten con un buen rendimiento. Si bien no existe un sistema de transmisión estandarizado para robots, si existen algunos que son utilizados con más frecuencia, estos se muestran en la tabla 2.3 la que muestra algunos tipos de transmisiones que se emplean además de las ventajas e inconvenientes que estas acarrean. (Mecatronica.net, Morfología del robot, 2017)

Tabla 2.3: Sistemas de Transmisión Para Robots.

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

Fuente: *(Ake Chuc, 2014)*

2.4.3. Actuadores

Los actuadores son los encargados de generar el movimiento que posteriormente será transmitido por las articulaciones, existen distintos tipos de actuadores estos pueden ser accionados por distintos tipos de energía como: neumática, eléctrica e hidráulica, cada uno de estos presenta sus propias características. La tabla 2.4 muestra las características que presentan los actuadores neumáticos, hidráulicos y eléctricos.

Tabla 2.4: Característica de Actuadores

	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	Motor CC Motor CA Motor paso a paso Servomotor
Ventajas	Baratos Rápidos Sencillos Robustos	Rápidos Alta relación potencia-peso Auto lubricantes 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: *(Ruiz, 2010)*

Según el tipo de energía eléctrica se presenta motores de corriente continua, motores de corriente alterna, motores paso a paso y finalmente servomotores, los que describiremos a continuación, una descripción del resto de actuadores se puede encontrar en la fuente de información.

Los servomotores Figura 2.5 son un tipo de actuadores que permiten ser operados de distintas maneras, como puede ser por la posición, velocidad o torque. Los servomotores se componen básicamente por un motor eléctrico, un sensor de posición un amplificador electrónico. Los

servomotores son accionados eléctricamente por una serie de pulsos denominada PWM (*pulse-width modulation)*

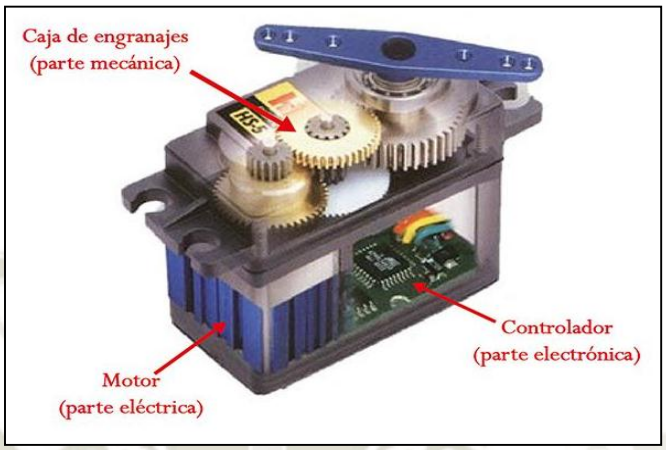


Figura 2.5. Servomotor.
Fuente: *(González A. G., 2016)*

2.4.4. Sensores

Para que el robot realice su tarea correctamente es necesario que este recopile la información del estado de su entorno, esta información se recopila con la ayuda de los sensores.

Esta información es enviada al sistema de control y se procesa para enviar las correcciones necesarias con la finalidad de que el robot realice correctamente la tarea que se le fue programada.

La tabla 2.5 muestra algunos de los sensores que se pueden emplear para obtener la posición del robot, presencia de un objeto y velocidad del robot en relación con su entorno.

Tabla 2.5: Sensores de Presencia Posición Y Velocidad.

Medida	Tipo
Presencia	Inductivo
	Capacitivo
	Efecto Hall
	Célula Reed
	Óptico
	Ultrasonido
	Contacto
Posición Analógicos	Potenciómetros
	Resolver
	Sincro
	Inductosyn
	LVDT
Posición Digitales	Digitales
	Encoders absolutos
	Encoders incrementales
	Regla óptica
Velocidad	Taco generatriz

Fuente: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

2.4.4.1. Sensor de Posición

Para identificar la posición angular de se emplea principalmente los encoders y resolvers, los encoders ópticos en particular consisten en una fuente de luz que apunta directamente hacia un plato, el cual presenta ranuras, las que permiten determinar la posición angular en la que se encuentra un actuador. La Figura 2.6 muestra un encoder óptico.

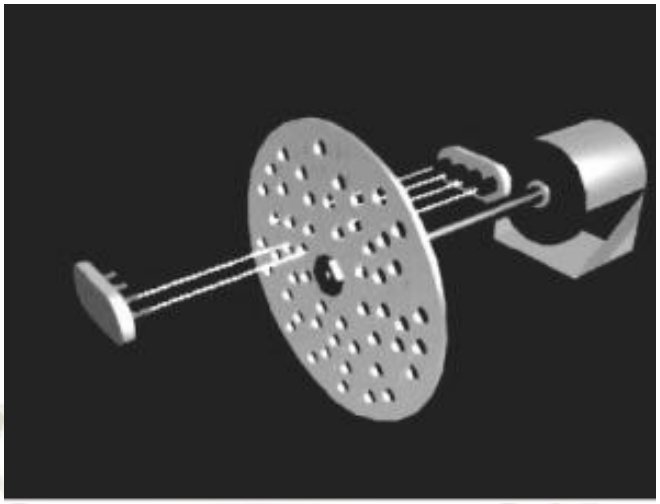


Figura 2.6. Encoder Óptico.
Fuente: *(Cortes, 2011)*

2.4.5. Elemento Terminal

También llamados efector final, es el que se encarga de interactuar directamente con el entorno de trabajo del robot, pueden ser elementos para manipulación o herramientas.

Figura 2.7. Ventosa Festo.
Fuente*: (FESTO, 2018)*

La tabla 2.6 muestra los diferentes sistemas que existen para la manipulación o sujeción de objetos.

Tabla 2.6: Sistemas de manipulación o sujeción.

Tipos de Sujeción	Accionamiento	Usos
Pinzas de presión de desplazamiento angular o 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
Electroimán	Eléctrico	Piezas ferromagnéticas

Fuente: *(Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

2.5. Robot Paralelo

Un robot paralelo (también llamado manipulador paralelo) es un mecanismo de cadena cinemática de lazo cerrado en el que el efector final se encuentra conectado a una base, básicamente es una plataforma móvil unida a una base fija por medio de brazos, cada brazo es controlado por un actuador independiente.

A diferencia de los robots en serie los robots paralelos tienden a poseer extremidades más simples y más cortas, por lo que son más rígidas y los motores son ubicados cerca de la base con el fin de reducir la masa en movimiento.

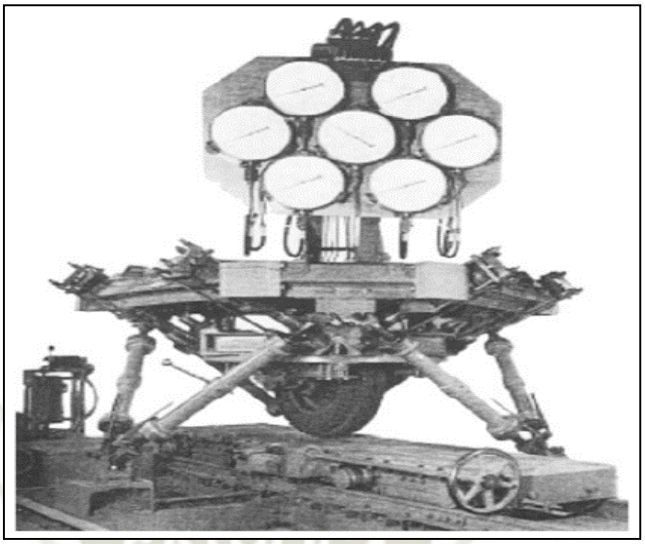


Figura 2.8. Plataforma de Gough.
Fuente: *(Andrew, 2015)*

Los robots paralelos fueron introducidos por Gough en el año 1947 con un manipulador paralelo que permite el posicionamiento y la orientación de una plataforma móvil para prueba el desgaste del neumático; la Figura 2.8 muestra la plataforma de Gough. En el año 1965 Stewart propuso el diseño de la plataforma Stewart con el fin de utilizarlo para simuladores de vuelo, la Figura 2.9 muestra la plataforma de Stewart.

Clavel en el año 1985 propuso el robot paralelo con configuración Delta, el robot paralelo con configuración delta es simétrico y este compuesto por tres brazos unidos a la base fija y la base móvil, cuenta con 3 grados de libertad y dependiendo de la tarea a realizar este podría tener un grado más si el actuador tuviera este grado, la Figura 2.10 muestra un robot paralelo con configuración delta.

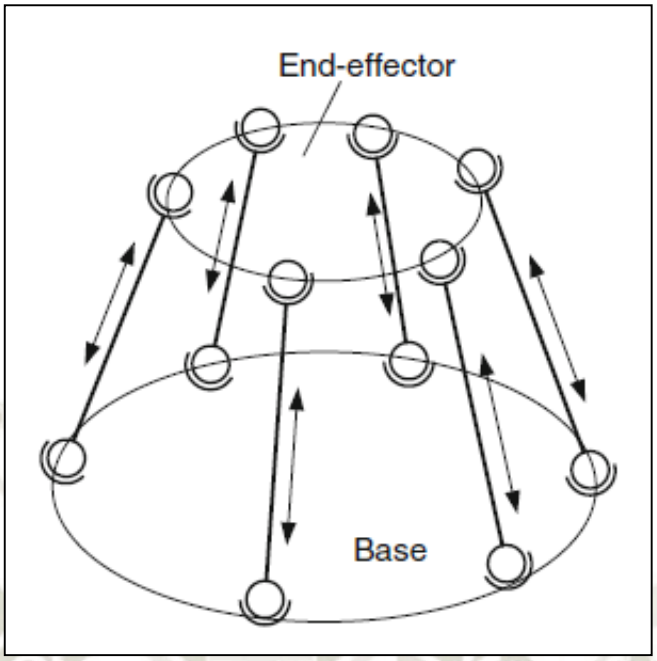


Figura 2.9. Plataforma Stewart.
Fuente: *(Andrew, 2015)*

Figura 2.10. Robot Paralelo con Configuración Delta.
Fuente: *(ABB, 2018)*

La Figura 2.11 muestra una cronología de las distintas patentes registradas a lo largo de los últimos años que contribuyeron con el desarrollo de los robots paralelos.

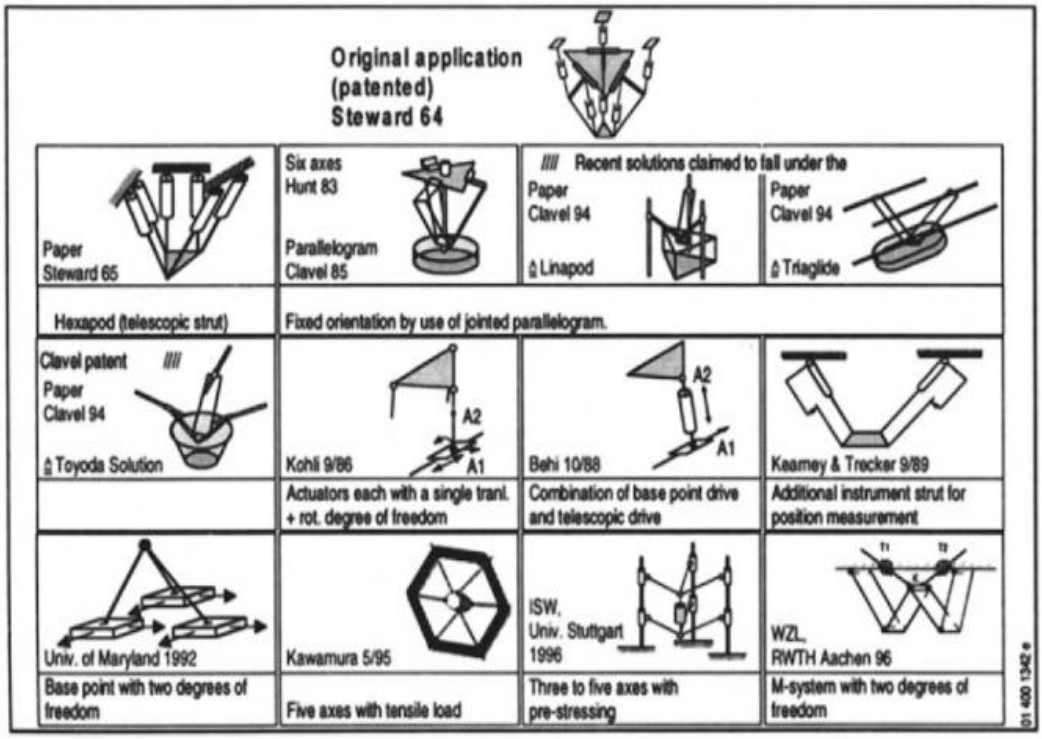


Figura 2.11. Robots Paralelos Patentados.
Fuente: *(Boer, Molinari-Tosatti, & Smith, 1999)*

2.5.1. Ventajas

- Cuentan con una alta rigidez debido a que la carga a soportar se comparte entre todos los brazos del robot.
- Gracias al bajo momento de inercia de la estructura móvil cuentan con una alta aceleración y alta velocidad.
- Los errores de ensamblaje se corrigen en lugar de acumularse en el efector final.
- Poseen alta capacidad de fuerza debido a que la fuerza de salida es proporcionada por varios actuadores.

2.5.2. Desventajas

- Básicamente la mayor desventaja reside en el volumen de su espacio de trabajo.

- Las singularidades presentes en el espacio de trabajo.
- La calibración de sus eslabones y articulaciones es sumamente complicada.

2.5.3. Aplicaciones

Debido a sus múltiples ventajas los robots paralelos tienen múltiples aplicaciones como, por ejemplo:

- Simuladores de vuelo, como se muestra en la Figura 2.12.
- Posicionamiento de alta precisión.
- Manipulación de pick-and-place.

Figura 2.12. Simuladores de Vuelo.
Fuente: *(Hispaviacion, 2018)*

2.5.4. Clasificación

Una breve clasificación basada en la clasificación del espacio, la dimensión y el grado de libertad se dan de la siguiente manera:

- Mecanismo paralelo plano de dos grados de libertad
- Mecanismo paralelo plano de tres grados de libertad
- Mecanismo espacial paralelo de tres grados de libertad (Robot paralelo delta)
- Mecanismo espacial paralelo de cuatro grados de libertad

- Mecanismo espacial de cinco grados de libertad
- Mecanismo espacial de seis grados de libertad: La plataforma Gough-Stewart es el original del paralelo espacial de seis grados de libertad.

2.5.5. Posibles Estructuras

Dependiendo de los grados de libertad que posee el robot existen distintas posibilidades que emplean distintos tipos de articulación como las siguientes: (Zhang, 2010)

S: Articulación esférica (con 3 GDL).

R: Articulación rotacional (con 1 GDL).

H: Articulación cardan (con 2 GDL).

P: Articulación prismática (con 1 GDL).

Se puede combinar estos para satisfacer los requisitos de cada GDL. Estas posibilidades se muestran en la Tabla 2.3

Tabla 2.3 Posibles Configuraciones

Number of possibilities	DOFs = 2	DOFs = 3	DOFs = 4	DOFs = 5	DOFs = 6
1	2R	1R2P	1S1P	1S2R	2S
2	2P	2R1P	1S1R	1S2P	1S1H1P
3	1R1P	3R	1R3P	1S1R1P	1S1H1R
4	1H	3P	2R2P	1S1H	1S3R
5		1H1R	3R1P	1H3R	1S3P
6		1H1P	4R	1H2R1P	1S2R1P
7		1S	4P	1H1R2P	1S1R2P
8			1H2R	1H3P	1H4R
9			1H2P	5R	1H3R1P
10			1H1R1P	4R1P	1H2R2P
11				3R2P	1H1R3P
12				2R3P	1H4P
13				1R4P	6R
14				5P	5R1P
15					4R2P
16					3R3P
17					2R4P
18					1R5P
19					6P

Fuente: *(Zhang, 2010)*

2.6. Métodos de Localización Espacial

La manipulación de objetos que lleva a cabo un robot implica el movimiento del mismo por el espacio, para que el robot pueda mover el objeto se tiene que saber la posición en que se encuentra respecto al robot, por tal motivo es necesaria la utilización de distintos métodos matemáticos o computacionales para conocer la orientación y ubicación del objeto.

2.6.1. Posición en el Sistema De Referencia

Para obtener la posición de un robot en el espacio se necesita tener una herramienta que calcule los puntos en el espacio; para obtener estos puntos se tiene que tener un sistema de referencial, como puede ser el sistema cartesiano, estos sistemas dependiendo de la cantidad de dimensiones puede emplear ya sea dos o tres ejes.

En el caso que sean tres dimensiones el sistema de referencia estaría dado por OXYZ que estaría compuesto por los vectores OX, OY y OZ como se observa en la Figura 2.13.

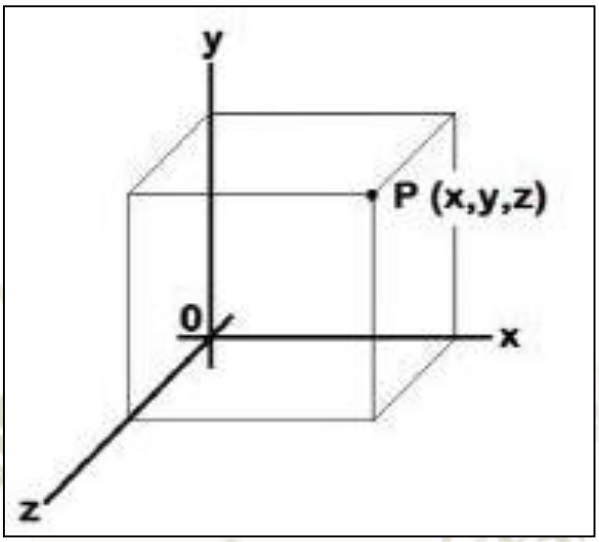


Figura 2.13. Sistema de coordenadas cartesiana
***Fuente**: (Inukai, 2011)*

2.6.2. Representación de la Orientación

En el espacio queda totalmente definido un punto con solo conocer su posición, pero en el caso de un sólido además de la posición se necesita conocer la orientación con respecto al sistema de referencia. En el caso de un robot y dependiendo de la aplicación que tendrá además de conocer la posición del extremo también se necesita conocer su orientación (Mecatronica.net, Herramientas Matemáticas Localización Espacial, 2017).

Por ejemplo, un robot que realizara una tarea de esculpido Figura 2.14 necesitaría conocer además de la posición del objeto se necesitaría conocer la orientación con la cual la herramienta entraría en contacto con el área de esculpido.

Figura 2.14. Robot Esculpiendo.
***Fuente**: (HD1080ide, 2013)*

2.6.2.1. Matrices de Rotación

El método más utilizado para determinar la orientación de un objeto son las matrices de rotación debido a lo cómodo que resulta utilizarlas.

Supongamos que se tiene los sistemas OXYZ y OUVW, 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 $\mathrm{i_x}, \mathrm{j}_y, \mathrm{k}_z$, mientras que los del OUVW serán $\mathrm{i_u}, \mathrm{j_v}, \mathrm{k_w}$ (Leyva, 2012).

Un vector del espacio, podrá ser referido a cualquiera de los sistemas de la siguiente manera:

$$puvw = [pu, pv, pw]\ T = pu.\ iu + pv.\ jv + pw.\ kw \quad (2.1)$$

$$pxyz = [px, py, pz]\ T = px.\ ix + py.\ jy + pz.\ kz \quad (2.2)$$

Realizando una serie de transformaciones se llega a la siguiente equivalencia.

$$\begin{bmatrix} px \\ py \\ py \end{bmatrix} = R \begin{bmatrix} pu \\ pv \\ pw \end{bmatrix} \quad (2.3)$$

Donde:

$$R = \begin{bmatrix} ixiu & ixjv & ixkw \\ jyiu & jyjv & jykw \\ kziu & kzjv & kzkw \end{bmatrix} \quad (2.3)$$

R es la matriz de rotación que define la orientación del sistema OUVW con respecto al sistema OXYZ. También tiene el nombre de matriz de cosenos directores y se trata de una matriz ortonormal, por lo que la inversa de la matriz R es igual a su traspuesta (Mecatronica.net, Herramientas Matemáticas Localización Espacial, 2017).

La razón principal de utilizar esta matriz de rotación es obtener la orientación del sistema girados únicamente sobre uno de los ejes del sistema de referencia.

2.6.2.2. Ángulos de Euler

En el caso de emplear matrices de rotación se necesita utilizar nueve elementos. Pero existe otro método llamado ángulos de Euler el cual solo emplea 3 elementos.

Todo sistema OUVW que sea solidario al cuerpo cuya orientación se desea obtener, se puede definir con respecto al sistema OXYZ con la ayuda de tres ángulos: φ, Θ, ψ, los cuales son llamados ángulos de Euler. Se gira uno por uno el sistema OXYZ sobre los ejes del triedro ortonormal los valores de φ, Θ, ψ Figura 2.15, y así

se obtiene el nuevo sistema OUVW. Por lo tanto, es necesario conocer además de los valores de los ángulos φ, Θ, ψ, cuáles son los ejes sobre los que se realizan los giros (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007).

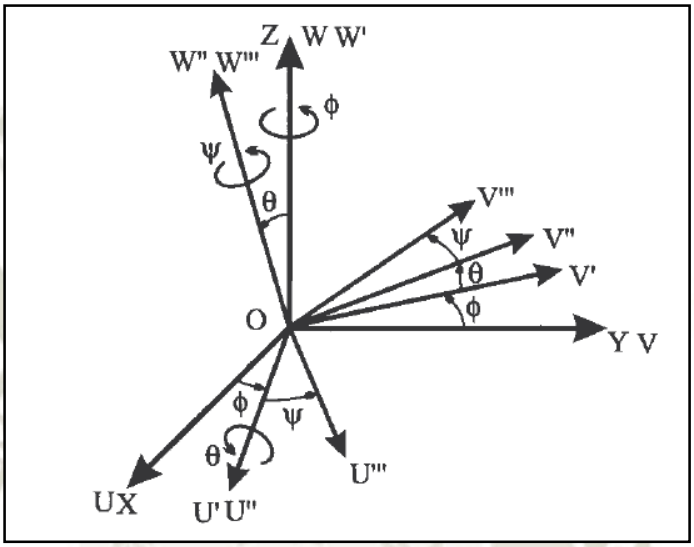


Figura 2.15. Ángulos de Euler
***Fuente**: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

2.6.2.3. Par de Rotación

La orientación del sistema OUVW con respecto al sistema de referencia OXYZ también se puede representar con la ayuda de un vector k (k_x, k_y, k_z) y un ángulo de giro θ, de modo que el sistema OUVW representa al sistema OXYZ girado un ángulo θ sobre un vector k Figura 2.16. Al par (k, θ) se le denomina par de rotación (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

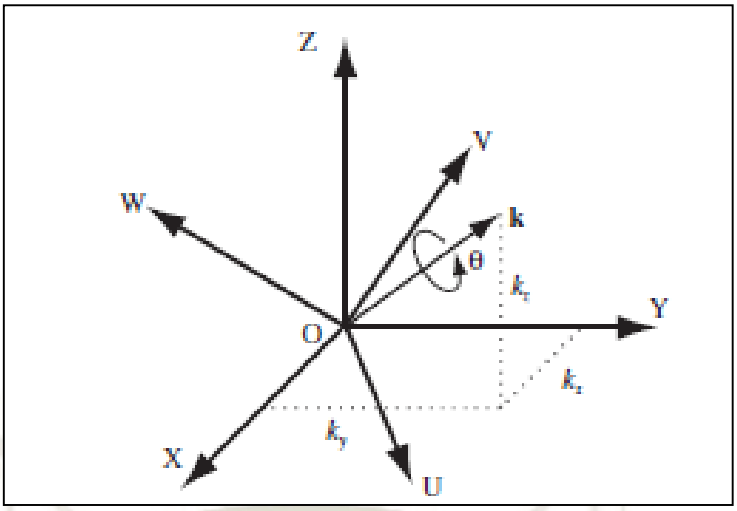


Figura 2.16. Representación de Orientación por Eje y Ángulo de Giro.
***Fuente**: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

2.6.2.4. Cuaternios

Un cuaternio Q se compone por cuatro componentes ($q_{0,}$ q_1, q_2, q_3) cada una de estos componentes representan las coordenadas del cuaternio en una base [e, i, j, k] donde es frecuente denominar a "e" como la parte escalar. Un cuaternio se representa de la siguiente manera:

$$Q = [q0, q1, q2, q3] = [s, v] \quad (2.4)$$

Donde "*s*" representa la parte escalar y "v" la parte vectorial.

2.6.3. Matrices de Transformación Homogénea

Las matrices de transformación homogénea permiten tener una representación conjunta de la posición y la orientación.

Un elemento de un espacio n-dimensional, se encuentra representado en coordenadas homogéneas por n+1 dimensiones de modo que un vector p (x, y, z) seria representado por p (wx, wy, wz, w), donde w es un valor escalar. De igual manera un vector $p = ai + bj + ck$, donde i, j, k son los vectores unitarios de los ejes OX, OY y

OZ del sistema de referencia OXYZ y se representa en coordenadas homogéneas de la siguiente manera: (Mecatronica.net, Herramientas Matemáticas Localización Espacial, 2017)

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} aw \\ bw \\ cw \\ w \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \quad (2.5)$$

Una matriz de transformación homogénea T se define como una matriz de dimensión 4x4 que representa las transformaciones de un vector de coordenadas homogéneas a otro. Una matriz homogénea está compuesta por cuatro submatrices: (Icaro, 2017)

$$T = \begin{bmatrix} R3x3 & p3x1 \\ f3x1 & w3x1 \end{bmatrix} = \begin{bmatrix} Rotacion & Traslación \\ Perspectiva & Escalado \end{bmatrix} \quad (2.6)$$

- R_{3x3}: Representa la matriz de rotación.
- p_{3x1}: Representa la traslación.
- f_{3x1}: Representa una transformación de perspectiva.
- w_{3x1}: Representa un escalado global.

En robótica generalmente solo se considera las matrices “R” y “p”, ya que no existe ninguna transformación de la perspectiva ni del escalado y el valor de “w” es uno y de “f” es cero.

$$T = \begin{bmatrix} R3x3 & p3x1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Rotacion & Traslación \\ 0 & 1 \end{bmatrix} \quad (2.7)$$

2.7. Cinemática del Robot

Como su nombre lo indica la cinemática del robot estudia el movimiento del mismo con respecto a su sistema de referencia, la cinemática describe analíticamente el movimiento espacial del robot en función del tiempo y en

particular la relación que existe entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Para resolver la cinemática se presentar dos dificultades, las cuales son, el problema cinemático directo, este consiste en poder determinar la posición y orientación del extremo final con respecto al sistema de coordenadas utilizado como referencia, el segundo es el problema cinemático inverso, el cual determina los ángulos que necesita moverse el robot paras alcanzar una posición y orientación conocidas, por lo que existe una relación entre estos dos Figura 2.17.

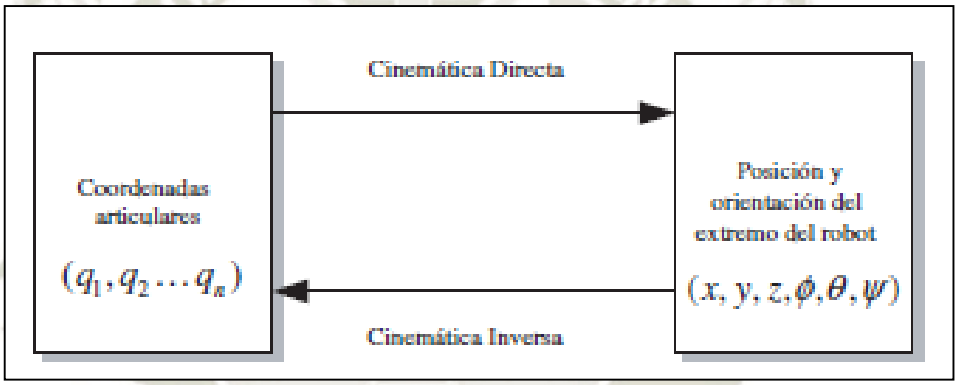


Figura 2.17. Relación entre Cinemática Directa e Inversa.
***Fuente**: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

2.7.1. Problema Cinemático Directo

Resolver el problema cinemático directo permite conocer la posición y la orientación que tendrá el extremo final del robot en cualquier momento conociendo los valores de las variables articulares. Los valores articulares pueden ser determinados fácilmente por medio de sensores. Para determinar el modelo cinemático directo se pueden utilizar dos métodos.

2.7.2. Métodos Geométricos

La resolución por este método consiste en determinar de manera algebraica la relación que permita conocer la posición y orientación del extremo del robot a partir de los valores articulares.

Por ejemplo, la trilateración es un método geométrico que se emplea para determinar las posiciones relativas de objetos usando la geometría de triángulos. La trilateración utiliza la ubicación ya conocida de dos o más puntos de referencia, y la distancia medida entre el punto deseado y cada punto de referencia. (Wikipedia, Trilateración, 2018).

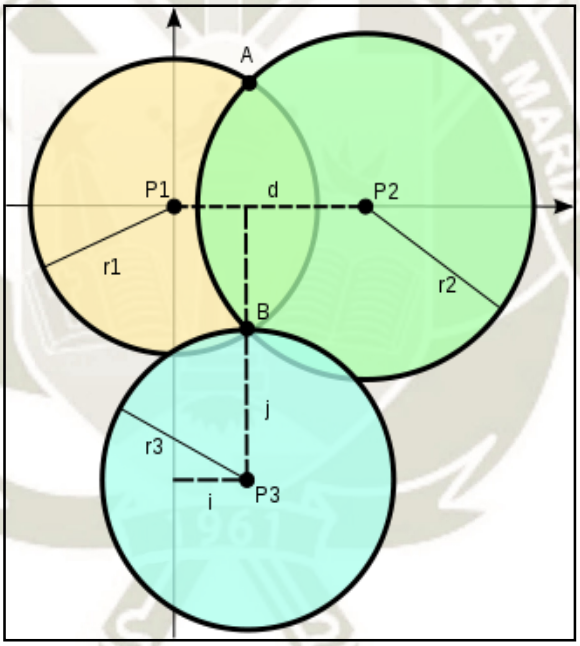


Figura 2.18. Trilateración.
***Fuente**: (Wikipedia, Trilateración, 2018)*

2.7.3. Por Matrices de Transformación Homogénea

Como ya se mencionó el problema cinemático directo consiste en tener una relación entre la posición y orientación con los valores articulares, una manera es tener una matriz de transformación homogénea que relacione estos valores, la cual estará en función de los valores

articulares. (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

Por lo general un robot de n GDL está formado por una serie de eslabones unidos por articulaciones, por lo tanto, se puede asignar un sistema de referencia solidario a cada eslabón, y utilizando las matrices de transformación homogénea se puede determinar la posición y orientación del extremo final del robot. La matriz de transformación homogénea que representa la orientación y posición respecto a los sistemas asociados consecutivos de denomina $^{i-1}A_i$. De tal modo que la matriz del primer eslabón seria $^{0}A_1$.

2.7.4. Por Algoritmo de Denavit Hartenberg para la Obtención del Modelo Cinemático Directo

En 1955 Denavit y Hartenberg propusieron un método matricial que permite determinar la localización que debe tener el extremo de un robot respecto a sus valores articulares. Para esto solo necesitan emplear 4 transformaciones básicas que consisten en una serie de rotaciones y traslaciones que finalmente permiten tener una relación entre el sistema de referencia i-1 con el sistema del elemento i. Las transformaciones se muestran en la Figura 2.19 y son las siguientes: (Aurova, 2017)

- 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 (a_i, 0, 0).
- Rotación alrededor del eje x_i un ángulo α_i.

Donde las transformaciones se refieren al sistema móvil.

Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$i-1Ai \;=\; Rotz\,(\theta i)\;T\,(0,0,di)\;T\,(ai\;0,0)\;Rotx\,(\alpha i.) \qquad (2.8)$$

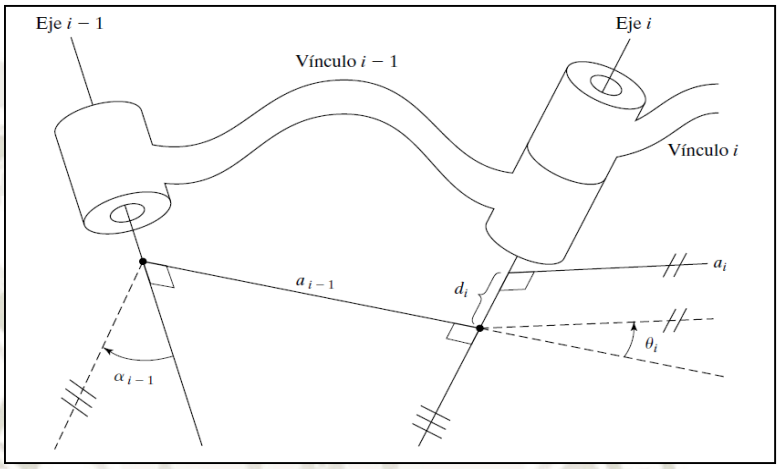


Figura 2.19. Parámetros de D-H para un Eslabón Giratorio
***Fuente**: (Craig, Cinemática de Manipuladores, 2006)*

2.8. Problema Cinemático Inverso

El problema cinemático inverso tiene como objetivo establecer los valores que deben de adoptar las articulaciones para que el extremo del robot pueda posicionarse en un punto en el cual la posición y la orientación son conocidas.

2.9. Modelo Diferencial

El modelo cinemático busca buscar una relación entre los valores articulares y la posición y orientación del extremo del robot, pero este no toma en cuenta las fuerzas o pares que actúan sobre el robot, sin embargo, si recurre a la cinemática del robot conocer la relación que existe entre la velocidad articulares y las de la posición y orientación del extremo del robot, esta relación queda definida por el modelo diferencial, el cual queda concretado

en la matriz Jacobiana. Similar que, con el modelo cinemático, existe una jacobiana directa y una jacobiana inversa. (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)

Existen dos métodos de determinar la jacobiana, la primera relaciona las velocidades articulares con otro vector de velocidades, esta relación viene dada por la jacobiana analítica.

La otra manera relaciona las velocidades articulares con los vectores de velocidad lineal y angular, esta relación viene dada por la matriz jacobiana geométrica

2.9.1. Jacobiana Analítica

La jacobiana analítica relaciona las velocidades articulares con las velocidades de localización (posición y orientación) del extremo del robot. Como se muestra en la Figura 2.20

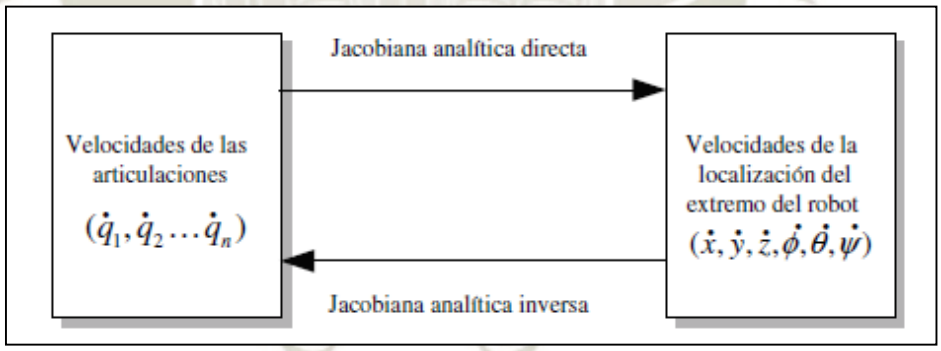


Figura 2.20. Jacobiana Analítica.
***Fuente**: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

La forma de obtener la jacobiana analítica es diferenciando la ecuación del modelo cinemático directo.

2.9.2. Jacobiana Geométrica

La jacobiana geométrica relaciona las velocidades articulares con las velocidades lineales y angulares del extremo del robot. Como se muestra en la Figura 2.21

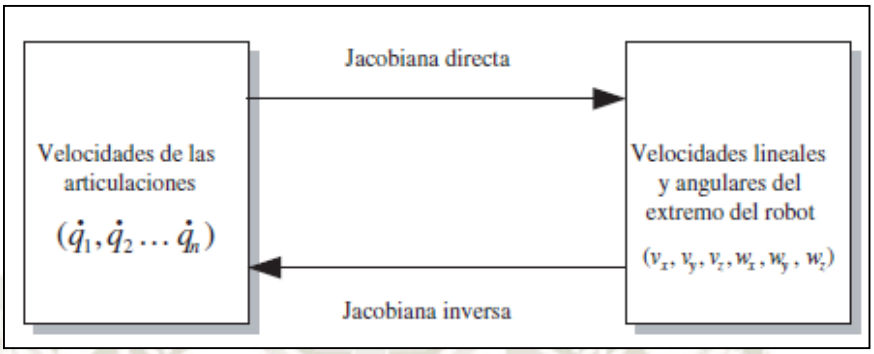


Figura 2.21. Jacobiana Geométrica.
***Fuente**: (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

Las velocidades lineales se obtienen diferenciando respecto al tiempo las posicionaron x, y, z que se obtienen en el modelo cinemático, por lo tanto, serán igual que las obtenidas en la jacobiana analítica, la velocidad angular se obtiene a partir de la matriz de rotación de la MTH.

2.9.3. Jacobiana Inversa

La jacobiana inversa permite obtener las velocidades articulares partiendo de las velocidades del extremo del robot.

Esta se puede obtener simplemente invirtiendo la matriz jacobiana, siendo una condición que la matriz jacobiana sea cuadrática.

2.10. Dinámica del Robot

El modelo dinámico se encarga de establecer la relación que existe entre las fuerzas aplicadas a un robot con el movimiento del mismo. El modelo dinámico se obtiene estableciendo las siguientes relaciones: (Nbio, 2017)

- La localización del robot con la velocidad y aceleración.

- Las fuerzas y pares aplicados en el extremo del robot o en las articulaciones.
- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

La obtención del modelo dinámico se torna complejo a medida que aumentan los GDL de un robot, por lo tanto el modelo dinamice tiene que ser resulto por algún método iterativo.

Los propósitos de establecer la relación entre el movimiento y las fuerzas implicadas en este son las siguientes:

- Obtener la simulación del robot.
- Diseñar y evaluar la estructura del robot.
- Dimensionar los actuadores.
- Diseñar y evaluar el control dinámico del robot.

2.10.1. Modelo Dinámico de la Estructura Mecánica de un Robot Rígido

La obtención del modelo dinámico de la estructura mecánica de un robot se basa en el equilibrio de fuerzas establecido en la segunda ley de newton, o su equivalente para movimientos de rotación, la ley de Euler.

$$\sum F = \frac{d}{dt}(mv) \qquad (2.9)$$

$$\sum T = \frac{d}{dt}(1\omega) = 1\omega + \omega * \dot{}(1\omega) \qquad (2.10)$$

Así como con el modelo cinemático se tiene un modelo cinemático directo y uno inverso de igual manera ocurre con el modelo dinámico,

en el que se tiene un modelo dinámico directo y un inverso (Sánchez & Saavedra, 2005).

- Modelo dinámico directo: permite obtener las coordenadas articulares del robot a partir de las fuerzas y torques que se le aplican al robot.
- Modelo dinámico inverso: con este modelo se determina las fuerzas y torques que se necesita para que el robot llegue a una coordenada articular ya establecida.

Plantear el equilibrio de fuerzas de un robot de 5 o 6 GDL, ya se torna bastante complicado, debido a la presencia de las fuerzas de centrípetas originadas por la rotación y fuerzas de Coriolis, causadas por el movimiento relativo que existe entre los elementos, las cuales dependen de la configuración que posea el robot en cada instante.

Por lo que un método alternativo que permite obtener el modelo dinámico es la formulación lagrangiana, debido a que es un método más sistemático que el anterior se facilita la formulación de un modelo tan complejo como el de un robot de 5 o más GDL. Este se basa en obtener el modelo dinámico considerando las energías presentes en el modelo. La formulación lagrangiana establece la siguiente ecuación.

$$L = Ec - Ep \tag{2.11}$$

$$\tau i = \frac{d}{dt}\frac{\delta L}{\delta \dot{q} i} - \frac{\delta L}{\delta q i} \tag{2.12}$$

Donde:

L = Función lagrangiana.

Ec = Energía cinética.

E = Energía potencial.

q_i = Coordenadas generalizadas (en este caso las articulares).

τi = Fuerza o pares aplicado sobre el grado de libertad q_i.

Cualquiera de los métodos utilizado para obtener el modelo dinámico tendrá la siguiente forma:

$$\boldsymbol{\tau_i = D\ddot{q} + H + C} \quad (2.13)$$

Donde:

q = Vector de coordenadas articulares.

τi = Vector de fuerzas o pares que se aplica a cada articulación.

D (q) = La matriz de inercias, de dimensión (n x n), cuyos elementos son función de q.

H (q, $\dot{q}$) = Matriz (n x 1) de fuerzas de Coriolis, dependiente de q y $\dot{q}$

C (q) = Matriz (n x 1) de fuerzas de gravedad, dependiente de q.

N = Número de grados de libertad del robot.

2.10.2. Modelo Dinámico en el Espacio de la Tarea

Para obtener el modelo dinámico se considerada la expresión de la jacobiana y la relación de la jacobiana geométrica.

$$\dot{v} = J\dot{q} \quad (2.14)$$

Derivando la expresión anterior con respecto al tiempo, se obtiene:

$$\ddot{v} = \dot{J}\dot{q} + J\ddot{q} \quad (2.15)$$

$$\ddot{q} = J^{-1}\ddot{v} - J^{-1}\dot{J}\dot{q} \quad (2.16)$$

Esta expresión relaciona las aceleraciones articulares y cartesianas de manera directa e inversa. Por otro lado, se sabe que la potencia consumida por el robot debe ser igual si se compara desde el espacio articular o el cartesiano.

$$Potencia = Par.Velocidad \rightarrow T^T \dot{v} = \tau^T \dot{q} \quad (2.17)$$

Donde T^Tes el vector de fierzas y pares ejecutados en el extremo del robot y τ^Tes el vector de pares y fuerzas articulares aplicados en las articulaciones. De la expresión [2.15] y [2.18] se obtiene:

$$T^T \dot{v} = \tau^T \dot{q} \rightarrow T^T J\dot{q} = \tau^T \dot{q} \rightarrow T^T J = \tau^T \rightarrow \tau = J^T \mathrm{T} \quad (2.18)$$

Sustituyendo [2.17] y [2.19] en [2.14] se obtiene:

$$\tau = D\ddot{q} + H + C$$

$$J^T \mathrm{T} = DJ^{-1}\ddot{v} - DJ^{-1}J\dot{q} + H + C$$

$$\mathrm{T} = (J^T)^{-1}DJ^{-1}\ddot{v} - (J^T)^{-1}DJ^{-1}J\dot{q} + (J^T)^{-1}H + (J^T)^{-1}C$$

$$T = D_j\ddot{v} + H_j + C_j \quad (2.19)$$

2.11. Inteligencia Artificial

El termino Inteligencia Artificial (IA) se le atribuye a Marvin Minsky del MIT (*Instituto Tecnológico de Massachusetts*) quien en 1961 escribió un artículo llamado “Hacia la inteligencia artificial”, pero lo que se conoce hoy como IA se debe a John McCarthy quien en 1960 desarrollo a LISP el primer lenguaje de investigación de la IA. (Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería, 2010)

La IA siempre ha utilizado como modelo las funcionalidades del hombre. La principal motivación para el desarrollo de la IA fue el deseo de crear una máquina que sea capaz de pensar como lo haría un ser humano, o una máquina que sea capaz de imitar alguna función y que esta demuestre cierta inteligencia. El estudio de la inteligencia es una de las disciplinas más antigua, además de que es un campo al que científicos de otras especialidades le gusta estudiar, como por ejemplo desde hace más de 2000 años los filósofos han buscado la explicación de cómo es que se aprende, se recuerda y se razona además de la manera en cómo estas tareas deben de realizarse. La IA está compuesta por varios elementos de las cuales se puede identificar 3 ramas: lógica difusa, Redes neuronales, algoritmos genéticos (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

2.11.1. Lógica Difusa

Las computadoras procesan una serie de datos complejos que básicamente están compuestos por unos y ceros y proposiciones que pueden ser solo o falsas o verdaderas, pero el cerebro humano es capaz de procesar información que posee cierta incertidumbre. La lógica difusa es una de las ramas de la IA que busca que una computadora sea capaz de trabajar con datos que se encuentran en una escala entre cierto o verdadero. (Banda, 2014)

Las bases para el desarrollo de la lógica difusa se dan hace más de 30 años por Lotfi Asker Zadeh. La lógica difusa se crea para imitar la lógica humana y tomar decisiones a pesar de la información, es una herramienta que está basada en reglas lingüísticas dictadas por

personas con cuentan con experiencia manipulando o tomando decisiones sobre un proceso es decir por expertos. Por ejemplo, la altura de una persona es una variable que puede tomar distintos valores lingüísticos, como “bajo”, “mediano” o “alto”. Estas variables ligústicas están determinadas por reglas que determinar la salida del sistema. (Cumpa, 2014)

La lógica difusa es un conjunto de principios matemáticos determinados en grados de pertenecía, cuya principal función es la de procesar la información. Este procesamiento emplea las reglas lingüísticas que aproximan una función mediante la relación de entradas y salidas del sistema. Esta lógica presenta rangos dentro de un intervalo que va entre cero y uno, a diferencia de la lógica convencía, la cual se limita solo a valores de cero y uno. (Morcillo, 2011)

2.11.2. Aplicaciones

La logita difusa tiene una gran utilidad y por tanto las aplicaciones que se le pueden dar se desarrollan en distintas áreas, las cuales van desde el desarrollo de programas que puedan tomar decisiones adecuadas hasta el desarrollo de tecnología en electrodomésticos a continuación se definen algunas: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

- **Controladores**: La lógica difusa se utiliza en el control de procesos como, por ejemplo, controlar la posición de un motor, corregir el error de un proceso.

- **Electrodomésticos**: La lógica difusa se emplea en electrodomésticos como dos posibles variables: hardware y software. En el caso de las aplicaciones que involucran hardware un ejemplo son las cámaras con enfoque automático del lente el cual se realiza por medio del uso de sensores.
- **Reconocimiento**: En procesos o actividades que requieran la identificación de un suceso por medio del procesamiento e interpretación de valores guardados en una base de datos ejemplo la identificación de actividad volcánica.

2.11.3. Uso de Lógica Difusa en Sistemas de Control

Una de las principales ventajas de emplear el control difuso es que este no necesita del modelado dinámico de la planta a controlar. El control difuso está basado en el conocimiento de un experto, es una estrategia de control automático. El control difuso tiene los siguientes objetivos:

- Mejorar la robustez que se obtiene con los métodos de control clásico.
- Contar con un diseño simplificado para modelos con alta complejidad.
- Ser autónomos.
- Ser adaptativos.
- No necesitar del modelamiento de la planta a controlar.

2.11.4. Conceptos de Lógica Difusa

Establecer una nomenclatura y terminología es indispensable. Cuando se trabaja con conjuntos nítidos la variable a usar es “X”. En conjuntos

difusos la función de pertenecía que se utiliza es “μ”. La que toma valores que están entre cero y uno. Los conjuntos difusos se pueden representar de dos maneras: de forma discreta o continua. (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

2.11.4.1. Lógica Simbólica

La lógica simbólica permite establecer un lenguaje artificial empleando símbolos. Lo primero es determinar las proposiciones, es decir oraciones verdaderas o falsas, es posible convertir estas oraciones en un lenguaje de símbolos y representaciones que permitan ejecutar operaciones o convertirlas nuevamente en proposiciones.

Una proposición puede ser simple o compuesta, dependiendo de los valores de verdad de los componentes simples conectados por operadores como y, o, no, por mencionar algunos. La conjunción está dada por el operador y, este se simboliza con “$\wedge$”. La disyunción está dado por el operador o, este se simboliza con “$\vee$”. La tabla 2.7 muestra los resultados de la con la conjunción y la disyunción (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

Tabla 2.6: Resultados de la con la conjunción y la disyunción

p	**q**	**p$\wedge$q**	**p$\vee$q**
V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

Fuente: (ISC, 2011)

Además de estas también existen proposiciones condicionales de la forma “si p, entonces q”; donde “p” viene a ser el antecedente y “q” el consecuente. Una forma de simbolizar esta proposición se muestra en la tabla 2.7.

Tabla 2.7: Tabla de verdad de una proposición condicional

p	Q	p→q
V	V	V
V	F	F
F	V	V
F	F	F

Fuente: (ISC, 2011)

2.11.5. Mecanismo de Inferencia

El mecanismo de inferencia forma parte de la estructura básica de un sistema difuso, debido a que este es el encargado de realizar la acción experta en base a las reglas establecidas en el controlador difuso. Existen diferentes métodos de inferencia, los más comunes son de Mamdani y Sugeno.

2.11.5.1. Método Mamdani

El tipo Mamdani emplea reglas del tipo si-entonces. Una base de reglas está compuesta por dos partes, un antecedente y una conclusión. En el caso de Mamdani el antecedente y la conclusión están dado por expresiones lingüísticas. (Córdova, 2011)

Figura 2.22. Regla tipo si- entonces Mamdani.
Fuente: *(Córdova, 2011)*

2.11.5.2. Método Takagi-Sugeno-Kang

En el caso de Sugeno el valor del consecuente de las reglas activadas ya son valores numéricos por lo que no necesitan una etapa de defuzzificación. Las reglas de la base de conocimiento de un sistema Sugeno son distintas a las de un Mamdani debido a que el consecuente ya no es una etiqueta lingüística, sino una función de la entrada que tenga el sistema en un momento dado. (Córdova, 2011)

Figura 2.23. Regla tipo si- entonces Sugeno.
Fuente: *(Córdova, 2011)*

2.11.6. Funciones de Membresía

Para poder representar los grados de pertenecía que posee un elemento respecto a un conjunto difuso se extraen los datos que representan y junto con ellos se define la forma de la función de membresía. Las funciones de membresía permiten realizar un mapeo de un universo nítido a un universo difuso.

2.11.6.1. Función Saturación

Esta función comienza con un valor de cero que avanza hasta cierto punto para posteriormente crecer con una pendiente constante hasta alcanzar el valor de uno. La Figura 2.24 muestra la función saturación.

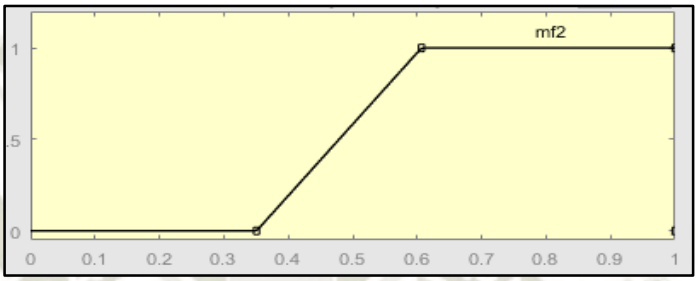


Figura 2.24. Función Saturación.
***Fuente**: Fuente Propia.*

2.11.6.2. Función Hombro

La función hombro realiza lo contrario de la función saturación. Esta función inicia con un valor de uno y desciende con una pendiente constantemente hasta alcanzar el valor de cero. La Figura 2.25 muestra la función hombro.

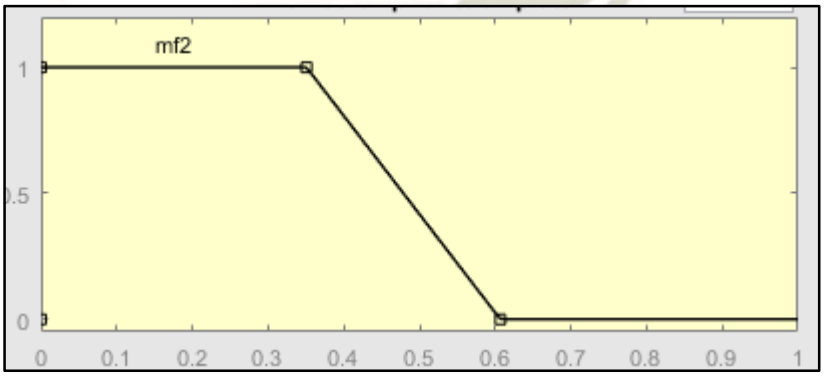


Figura 2.25. Función Hombro.
***Fuente**: Fuente Propia*

2.11.6.3. Función Triangular

La función triangular como su propio nombre lo indica tiene una forma triangular, cuenta con una parte con pendiente positiva

seguida de una endiente negativa. La Figura 2.26 muestra la función triangular.

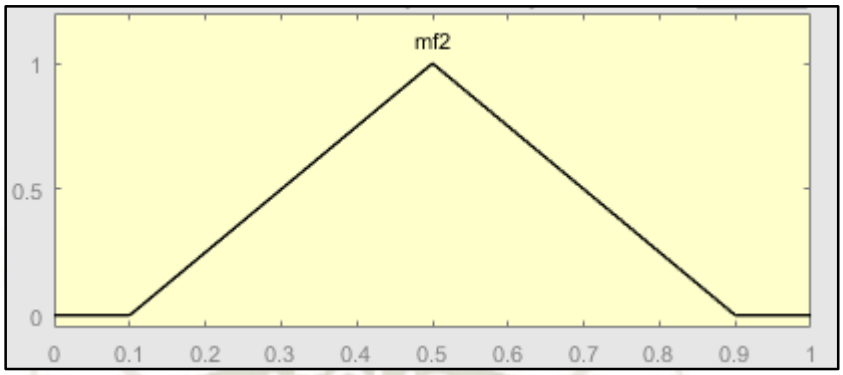


Figura 2.26. Función Triangular.
***Fuente**: Fuente Propia*

2.11.6.4. Función Trapecio o PI

Esta función es similar a la función triangular, con la diferencia de que esta tiene una franja de valores que tienen una pertenencia de 1 a diferencia de la función triangular que solo tiene un valor con pertenecía unitaria. La Figura 2.27 muestra la función PI.

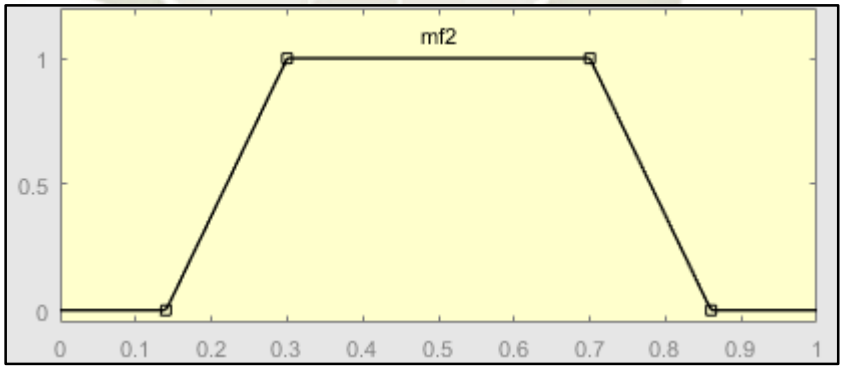


Figura 2.27. Función PI.
***Fuente**: Fuente Propia.*

2.11.6.5. Función Sigmoidal o S

La función sigmoidal tiene una forma similar a la función saturación, con la diferencia de que el cambio de grado de

pertenecía de esta no viene dado por una línea recta, sino una curva de segundo orden. La Figura 2.28 muestra la función sigmoidal.

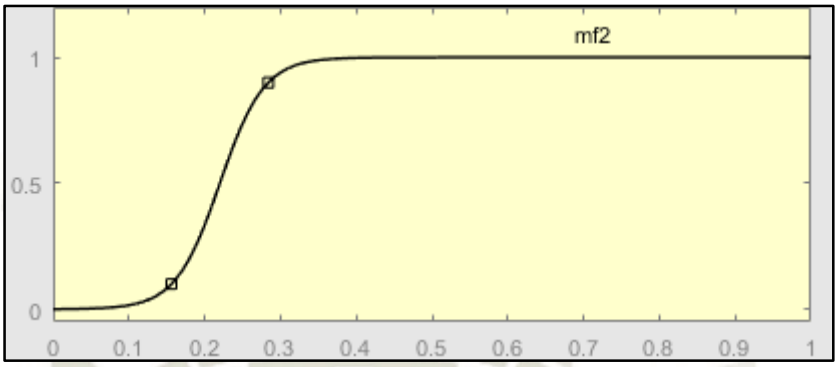


Figura 2.28. Función Sigmoidal.
***Fuente**: Fuente Propia*

2.11.7. Controladores Difusos

Un controlador difuso se compone por cuatro partes principales: interfaz de fuzzificación, base de conocimientos, lógica de decisiones e interfaz de defusificación, como se muestra en la Figura 2.29.

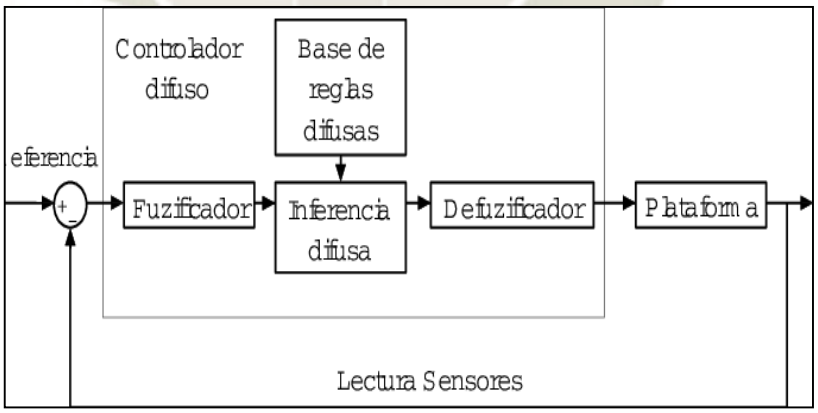


Figura 2.29. Estructurara de un Control Difuso.
***Fuente**: (Gómez, López, & Patiño, 2007)*

2.11.7.1. Interfaz de Fuzzificación

La interfaz de fuzzificación convierte lo valores de entrada a valores lingüísticos, por medio de un mapeo a escala que transfiere el rango de valores de las variables a un universo de

discurso difuso. La cantidad de conjuntos difusos determina la complejidad del controlador.

2.11.7.2. Base de Conocimiento

La base de conocimiento contiene la información del proceso que se va a controlar, también contiene las metas del controlador. Cuenta con una base de datos y una base de reglas lingüísticas que permiten controlar la variable. La base de datos otorga las definiciones que permiten establecer reglas y manipular los datos difusos. La base de reglas está dada por el criterio que utilizan los expertos para controlar un proceso.

2.11.7.3. Lógica de Decisiones

Es la que simula a la lógica que las personas utilizan para tomar decisiones, empleando las reglas dadas gracias a la base de conocimientos.

2.11.7.4. Interfaz de Defusificación

Se encarga del mapeo a escala que convierte las salidas a sus universos de discurso correspondiente. Algunos métodos se muestran a continuación.

2.11.7.4.1. Método de Centro de Área o Gravedad

Este método tiene una metodología sencilla, segmenta las funciones de membresía generando dos áreas. El área inferior que se forma es la que se toma para hacer el cálculo, se sobreponen todas estas áreas y se saca el centroide de la superposición, el cual indica la salida real. (Guzmán & Castaño, 2006)

2.11.7.4.2. Método de Centro Máximo

En primer lugar, se calcula el valor más común de cada etiqueta, esto se realiza respecto al máximo de la función de membresía. En el caso de funciones trapezoidales se escoge el medio de esta función Figura 2.30.

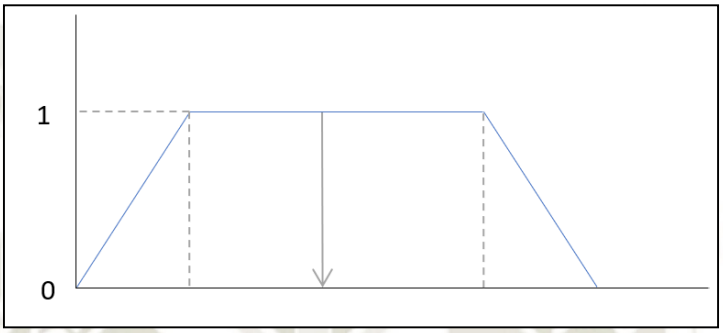


Figura 2.30. Método de Centro Máximo de una Función Trapezoidal.
***Fuente**: Elaboración Propia*

2.11.7.4.3. Método de la Izquierda Máxima

El valor de salida se obtiene similar que, con el método del centro máximo, pero en lugar del valor más común de la mitad se emplea el valor máximo izquierdo, como se muestra en la Figura 2.31.

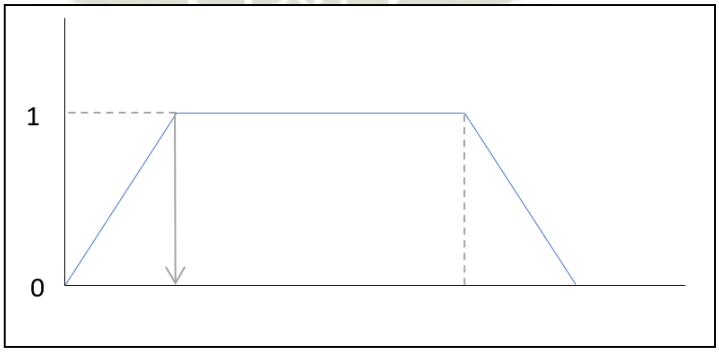


Figura 2.31. Método de Izquierda Máximo de una Función Trapezoidal
***Fuente**: Elaboración Propia*

2.11.7.4.3. Método de la Derecha Máxima

El valor de salida se obtiene similar que, con el método del centro máximo, pero en lugar del valor más común de la mitad se emplea el valor máximo izquierdo, como se muestra en la Figura 2.32.

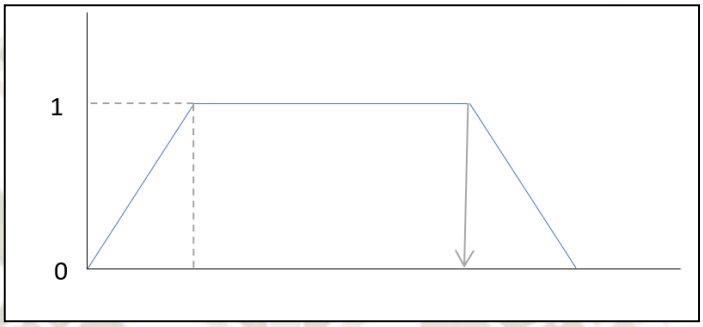


Figura 2.32. Método de Derecha Máximo de una Función Trapezoidal
***Fuente**: Elaboración Propia*

2.11.7.5. Control Difuso Clásico

Un control difuso clásico tiene un lazo de control como el que se muestra en la Figura 2.33, similar a los controladores clásicos.

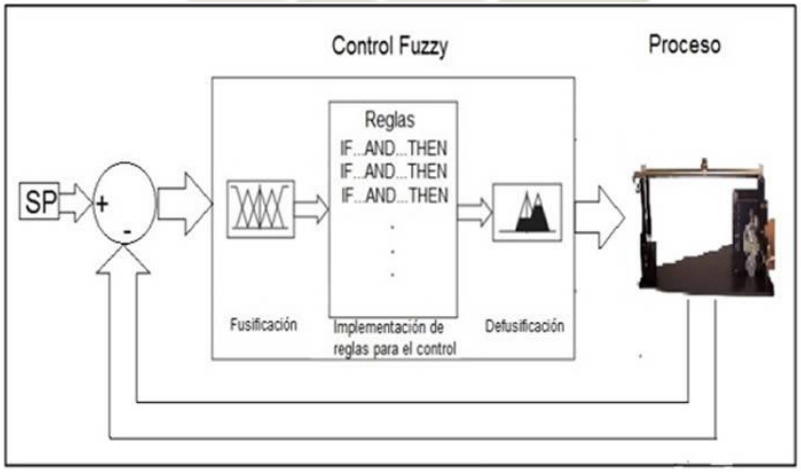


Figura 2.33. Lazo de un Controlador Difuso Clásico.
***Fuente**: (Charre-Ibarra, Velez-Díaz, Gudiño-Lau, Alcalá-Rodríguez, & Fuentes-Penna, 2016)*

Diseñar un controlador P, controlador PD, controlador PI y controlador PID difuso es posible a partir de las funciones de pertenencia para poder fuzzificar y defuzzificar.

2.11.7.5.1. Controlador P

Un controlador tipo P se define como u = KP ε, donde ε = r − y. Es a partir de esto y de la cantidad de funciones de pertenencia que se proponen las siguientes reglas, representadas de forma matricial en la tabla 2.8 y en la tabla 2.9, denominada matriz de asociación difusa o FAM. La Figura 2.34 y Figura 2.35 muestran las funciones de pertenecía de ambos controladores. (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

Tabla 2.8 Reglas de un controlador P con tres funciones de pertenencia

Entrada (error)	N	Z	P
Salida (corrección)	N	Z	P

Fuente: *(Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

Tabla 2.9 Reglas de un controlador P con cinco funciones de pertenencia

Entrada (error)	NB	NS	Z	PS	PB
Salida (corrección)	NB	NS	Z	PS	PB

Fuente: *(Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

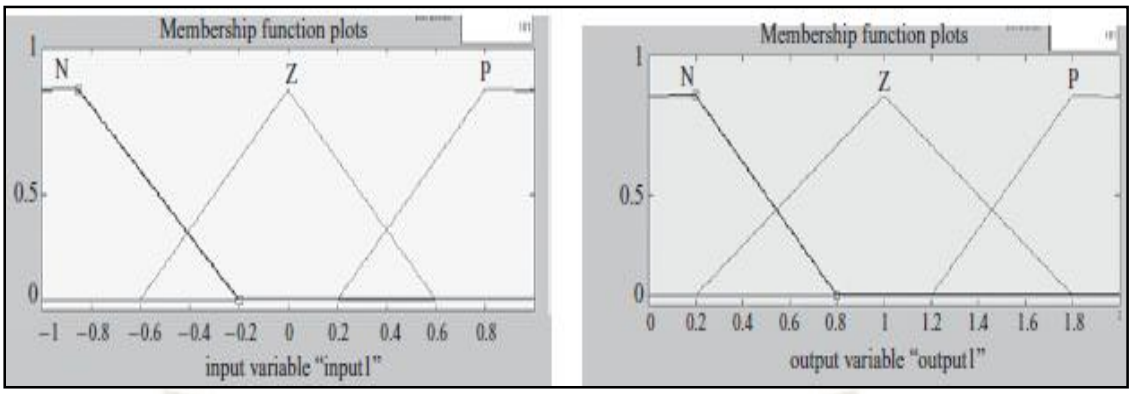


Figura 2.34. Controlador P con tres Funciones de Pertenencia
***Fuente**: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

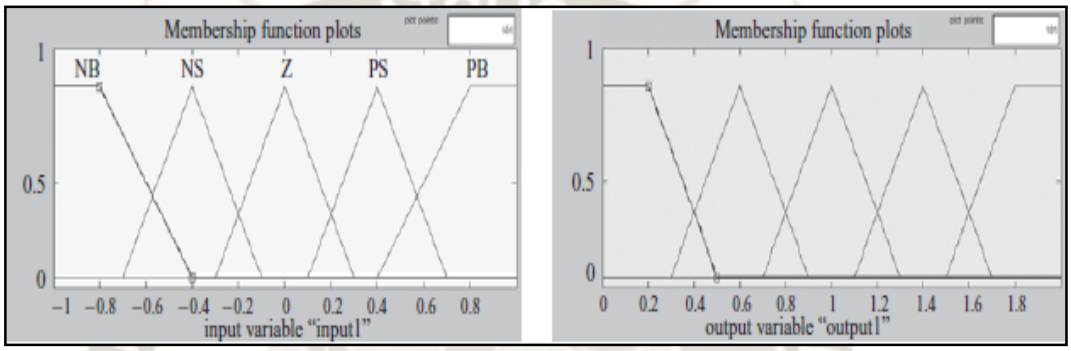


Figura 2.35. Controlador P con cinco Funciones de Pertenencia
***Fuente**: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

2.11.7.5.2. Controlador PD

Un controlador tipo PD se define como $u = KP\ \varepsilon + \mathrm{KD}\ \dot{\varepsilon}$, donde $\varepsilon = r - y$. Se propone dos controladores uno cuenta con tres funciones de pertenecía y el otro con cinco funciones de pertenencia. Las reglas se muestran en las Tablas 2.10 y en la tabla 2.11.

La Figura 2.36 y Figura 2.37 muestra las funciones de pertenecía de ambos controladores.

Tabla 2.10 Reglas de un controlador PD con tres Funciones de Pertenencia

		Error		
		N	**Z**	**P**
Derivada del error	**N**	N	Z	P
	Z	N	Z	P
	P	N	Z	P

Fuente: *(Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

Tabla 2.11 Reglas de un controlador PD con cinco Funciones de Pertenencia

		Error				
		NB	**NS**	**Z**	**PS**	**PB**
Derivada del error	**NB**	NB	NS	Z	PS	PB
	NS	NB	NS	Z	PS	PB
	Z	NB	NS	Z	PS	PB
	PS	NB	NS	Z	PS	PB
	PB	NB	NS	Z	PS	PB

Fuente: *(Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

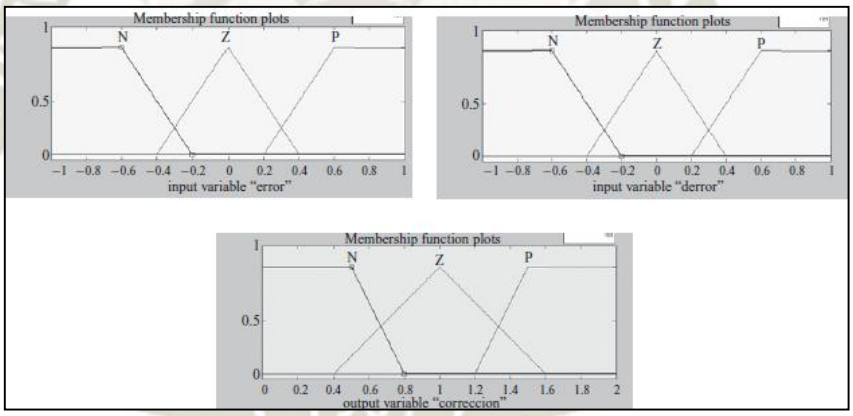


Figura 2.36. Controlador PD con tres Funciones de Pertenencia.
Fuente*: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

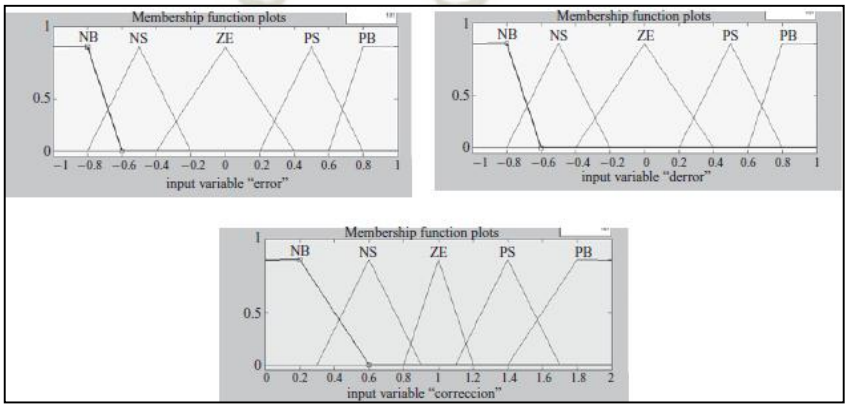


Figura 2.37. Controlador PD con tres Funciones de Pertenencia.
Fuente*: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

2.11.7.5.3. Controlador PI

Un controlador tipo PI se define como $u=K_P\,\varepsilon + K_I \int \varepsilon\, d\varepsilon$, donde $\varepsilon = r - y$. Siendo $\dot{u} = K_P\dot{\varepsilon} + K_I\varepsilon$ la derivada se propone obtener un controlador PI simplemente integrando la salida de un controlador PD. La Figura 2.38 muestra un control PI (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)

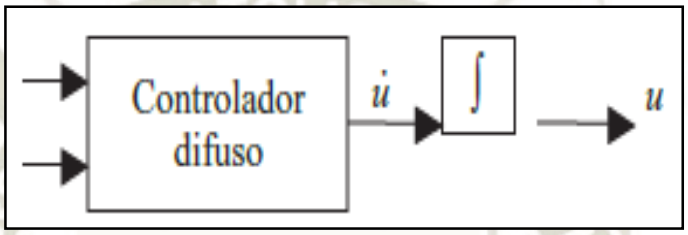


Figura 2.38. Obtención de un controlador PI difuso a partir de uno tipo PD.
***Fuente**: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

2.11.7.5.4. Controlador PID

Un controlador tipo PD se define como $u = KP\,\varepsilon + \mathrm{KD}\,\dot{\varepsilon} + \mathrm{KI} \int \varepsilon\, d\varepsilon$, donde $\varepsilon = r - y$. Se propone dos controladores uno cuenta con tres funciones de pertenecía y el otro con cinco funciones de pertenencia. Las reglas se muestran en la tabla 2.12; la Figura 2.39 muestra las funciones de pertenecía (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010).

Tabla 2.11 Reglas de un controlador PID con cinco funciones de pertenencia

Error	Derivada	Integral	Corrección
N	N	N	N
N	N	Z	N
N	N	P	N
N	Z	N	N
N	Z	Z	N
N	Z	P	N
N	P	N	N
N	P	Z	N
N	P	P	N
N	N	N	N
N	N	Z	N
N	N	P	N
N	Z	N	N
N	Z	Z	N
N	Z	P	N
N	P	N	N
N	P	Z	N
N	P	P	N
N	N	N	N
N	N	Z	N
N	N	P	N
N	Z	N	N
N	Z	Z	N
N	Z	P	N
N	P	N	N
N	P	Z	N
N	P	P	N

Fuente: *(Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

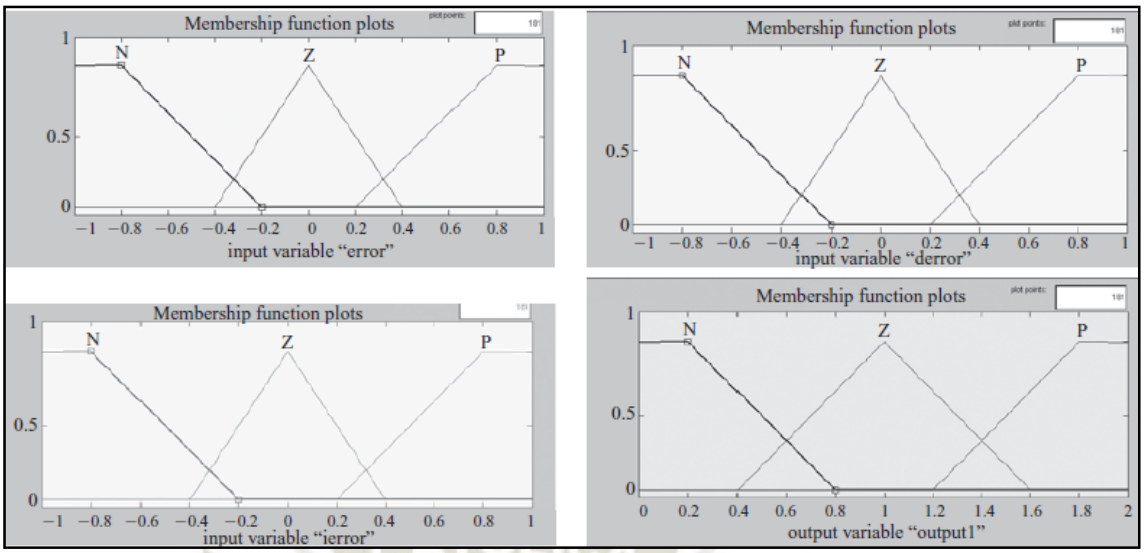


Figura 2.39. Controlador PID con tres funciones de pertenencia
***Fuente**: (Ponce Cruz, Inteligencia artificial con aplicaciones a la ingeniería, 2010)*

2.12. Programas a Emplear

2.12.1. Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo de Microsoft. Visual Studio soporta 36 distintos tipos de lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, etc. (Wikipedia, Microsoft Visual Studio, 2018)

Visual Studio incorpora un editor de programa que permite detectar errores de código de manera más sencilla y con esto se puede solucionar de manera más rápida tales errores, también incorpora una ventana que permite desarrollar un GUI, al cual se le puede incorporar distintas aplicaciones como cuadros de texto, sliders, tablas, imágenes, etc.

Visual Studio permite adquirir datos por medio de la comunicación serial de la computadora, esto permite la comunicación de distintos dispositivos por medio de la comunicación serial.

Figura 2.40. Visual Studio
***Fuente**: (Zahradník, 2017)*

2.12.2. MATLAB

MATLAB es una herramienta matemática utilizada a nivel mundial por su interfaz amigable ya que combina un entorno de escritorio para el análisis iterativo de datos con un lenguaje de programación que expresa problemas matemáticos.

También cuenta con la opción de generar interfaces de usuario (GUI) y la comunicación con programas que emplean otro lenguaje de comunicación, además de contar con una gran variedad de herramientas.

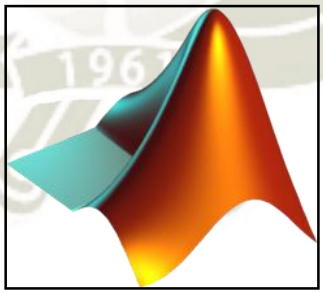


Figura 2.41. MATLAB.
***Fuente:** (Wikipedia, MATLAB, 2018)*

2.12.3. Arduino IDE

Arduino IDE es un software de código abierto con el que escribir un programa y subirlo a un controlador sea bastante sencillo, está escrito en lenguaje Java, pero también admite los lenguajes C y C++. El

software se puede emplear para cualquier placa Arduino. Este se puede instalar en cualquier sistema operativo y los requerimientos para poder instalarlo son insignificantes.

Figura 2.42. Arduino IDE.
Fuente: *(Arduino, 2018)*

2.12.4. Autodesk Inventor

El software Autodesk® Inventor® proporciona a los ingenieros y diseñadores una solución de grado profesional para el diseño, la simulación, la visualización y la documentación en mecánica 3D. Autodesk Inventor incluye poderosas herramientas de modelado, así como capacidades de traducción multi-CAD y dibujos DWG™ estándar de la industria (AUTODESK, 2018).

Figura 2.43. Autodesk Inventor.
Fuente*: (Autodesk, 2018)*

Capítulo III

3. DISEÑO

Antes de continuar, debemos empezar por preguntarnos ¿Qué es el diseño? Y ¿Por qué es tan importante? Diseñar es plantear una idea para satisfacer alguna necesidad específica o solucionar un problema, de esta manera si dicha idea se plasma en una creación físicamente tangible, deberá de cumplir con ciertos requerimientos o exigencias, tales como su confiabilidad, seguridad, funcionalidad, ergonomía, utilidad, operatividad, etc.

El diseño también es definido como un proceso iterativo, en donde la toma de decisiones es muy importante para poder consolidar un producto final. Estas decisiones nacen a partir del conocimiento, experiencia, habilidad para comunicarse y la destreza del diseñador para poder resolver problemas.

3.1. El Diseño en la Ingeniería Mecatrónica

Los ingenieros mecatrónicos cubren distintas áreas de la ciencia tales como la Mecánica, Computación, Control y la Electricidad y estas en conjunto dan origen a diferentes áreas en las cuales el ingeniero mecatrónico se puede enfocar tales como Sistemas de Control Digitales, Instrumentación, Automatización, Simulación de Procesos, etc. El diseño en la ingeniería mecatrónica involucra todas las áreas que componen esta disciplina.

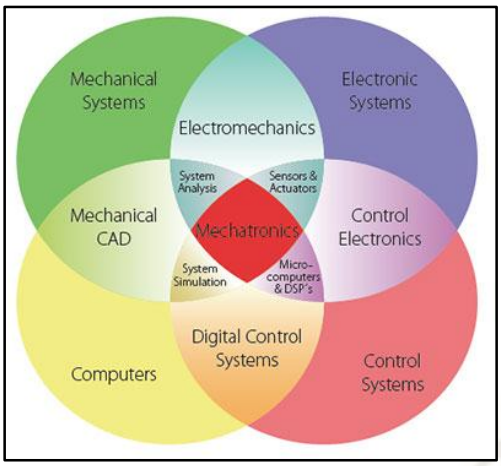


Figura 3.1. Conjunción de las disciplinas de la Mecatrónica
Fuente: *(SEAS, 2018)*

Los problemas reales a la hora del diseño se resisten a la especialización (Budynas & Nisbett, 2008). En otras palabras, se refiere a la dificultad que conlleva el diseño de un producto referido a la ingeniería. Por ejemplo, para poder llevar acabo el diseño de un drone involucran diferentes factores tales como el flujo del aire, el control de los motores, el tiempo de energía, el diseño de la estructura, las fuerzas involucradas en el movimiento de este, los procesos de manufactura y la codificación de un software capaz de interactuar con el drone.

El concepto de diseño en la ingeniería mecatrónica abarca gran cantidad de aspectos y eso se logra gracias a la conjunción de las diferentes áreas que integran la mecatrónica, es por eso que con estos conocimientos se puede tener un concepto más concreto de cómo se va a diseñar el producto, los factores que intervienen y como es que estos se relacionan entre sí. Es decir, al momento de diseñar el sistema mecánico del producto, se toma en cuenta cómo es que afectaría al sistema eléctrico y electrónico, esto le da un panorama más amplio al diseño en la ingeniería mecatrónica, pero como desventaja no puede profundizar a los mismos sistemas de manera más

compleja ya que no es un especialista, eh aquí donde la oración citada por (Budynas & Nisbett, 2008), toma más peso.

3.2. Fases del Proceso de Diseño

El proceso del diseño según (Budynas & Nisbett, 2008), parte en la identificación de la necesidad y luego de múltiples iteraciones finaliza con la presentación de la idea para satisfacer dicha necesidad.

Figura 3.2. Fases del proceso del diseño que reconoce múltiples iteraciones
Fuente: *(Budynas & Nisbett, 2008)"*

El análisis de las necesidades se refiere a encontrar déficits de una situación para así poder lograr dar una posible solución con el fin de mejorar un proceso, es un esfuerzo consecuente para identificar y resolver el problema. En cambio, la definición del problema es más exacto y debe de tomarse en cuenta todas las necesidades y especificaciones del producto a la hora de diseñarse.

Las especificaciones son las entradas y salidas del diseño como por ejemplo las características y las limitaciones. Las especificaciones nos dan un panorama más concreto de cómo se realizará el diseño y que se tiene que tener en cuenta como por ejemplo el limite económico, la facilidad de conseguir los componentes necesarios, la manufactura, etc. Luego de definir

correctamente el problema y tener en claro las especificaciones se pasan a uno de los procesos más importante que es, la síntesis, en la cual se debe proponer diferentes alternativas para la solución de la idea, deben investigarse, compararse y estar en constante análisis para saber si son las soluciones más optimas, es aquí donde podemos notar que la síntesis y el análisis están estrechamente unidos y en constante iteración.

Tanto en la síntesis como en el análisis se formulan modelos que describen la física del producto, estos modelos son de carácter matemático y suelen ser abstractos, cuando se crea dichos modelos matemáticos se espera que simule de manera correcta el producto en un ambiente físico real (Budynas & Nisbett, 2008).

Una vez finalizado todo el análisis llega la evaluación y esta representa la prueba final, esta prueba es la que nos brindara información si es que el diseño final satisface las necesidades y si es que no logra hacerlo se retrocede en el proceso para poder resolver el problema.

3.3. Diseño en la Robótica

Para llevar a cabo el diseño del robot paralelo delta de 3 GDL, se trabajará con las fases del proceso de Diseño descritas por (Budynas & Nisbett, 2008), pero antes de eso es necesario tener en cuenta la complejidad del caso. Los manipuladores robóticos son elementos de avanzada ingeniería mayormente diseñados con el fin de hacer tareas repetitivas y de avanzada precisión. El diseñar un manipulador robótico es una acción que cada día avanza más, esto se ve reflejada en la cantidad de proyectos e investigaciones y la cantidad de ingenieros que se enfocan en este tema. Sin embargo, para lograr un diseño que cumpla tan altas expectativas

pasaron años de investigación en los cuales estuvieron involucrados varios grupos de ingenieros y diseñadores para obtener tales resultados.

Hoy en día, los robots deben cumplir muchos requisitos para así ser llamados robots funcionales, deben ser ligeros, precisos, rápidos, tener un consumo moderado de energía y sobretodo compensar el gasto invertido. Viendo las especificaciones de esta manera, se recomienda empezar con el sistema más importante que es la estructura mecánica del robot. Esta estructura se compone de eslabones y articulaciones, mientras más articulaciones posea mayor será sus grados de libertad y esto conllevara a una dificultad mecánica ya que la precisión se verá afectada por cada error tratado en una articulación y este problema será resuelto con técnicas de control avanzadas.

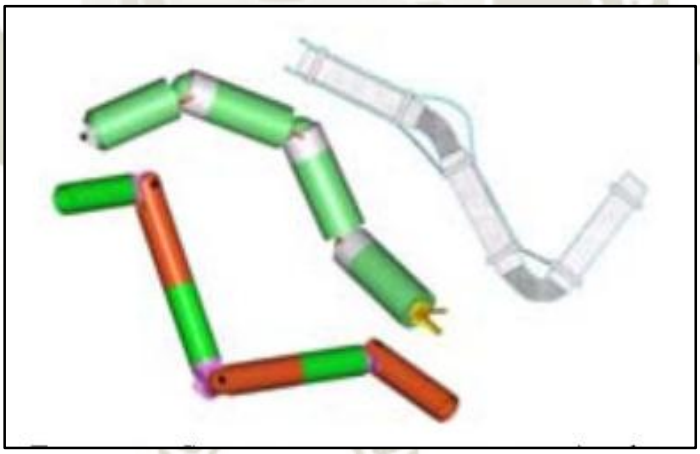


Figura 3.3. Estructura mecánica propuesta para robot quirúrgico
Fuente: *(Vilchis-González, y otros, 2014)*

Luego una vez planteada la estructura, se debe considerar las dimensiones de esta, pero dichas dimensiones deben de cumplir requisitos para así poder tener un espacio de trabajo bien definido y evitar problemas futuros. Una vez definida las dimensiones, se podrá trabajar con el espacio geométrico del robot, esta geometría estará definida por las dimensiones y los grados de libertad del robot, teniendo en cuenta esto se procede a analizar la

cinemática del robot. Esta es un parte crucial respecto al diseño de los robots ya que dicho análisis definirá el movimiento del robot de manera articular y cartesiana. Existen diversas investigaciones para poder resolver los modelos cinemáticos de los robots algunas complejas y otras más simples dependiendo de la configuración del robot.

La dinámica del robot es un área de suma complejidad, ya que en esta se aprecia el comportamiento del manipulador a lo largo del tiempo, como varia sus movimientos y las fuerzas que ejercen estos. De este análisis depende la correcta selección de actuadores y el método de control más óptimo que se adapte al diseño del robot para que pueda ejecutar las tareas encomendadas correctamente.

La técnica de control de un robot, es la parte más crucial ya que sin esta el robot carece de importancia debido a que sería una maquina inestable, el diseño de una técnica de control que se adapte al comportamiento del robot es un tema de extensa investigación que ha dado origines a nuevos métodos de control de robots, aun en la actualidad se siguen investigando métodos. En consecuencia, el diseño de la técnica de control conllevara a la elección de un procesador capaz de realizar consecutivamente y correctamente todos los análisis descritos anteriormente.

Finalmente, dependiendo de la tarea a realizar se escoge la herramienta de trabajo del robot y la manera en la que puede ser operado. Principalmente los robots, fueron diseñados para reemplazar al humano en la realización de tareas, esto conlleva a que se espera que el robot sea autónomo a la hora de realizar sus tareas. Sin embargo, es necesario que este no pierda la

esencia de la manipulación humana. Por consiguiente, es necesario una interfaz que permita interactuar al humano con el robot.

Figura 3.4. Manipulador Robótico con Interfaz Humano Maquina y Joystick
Fuente: *(Zacobria, 2018)*

3.4. Diseño Conceptual

Para poder llevar acabo el diseño del robot paralelo tipo delta de 3GDL es necesario dejar en claro las necesidades específicas que se quieren solucionar.

3.4.1. Determinación de las Necesidades del Robot

Primero, es necesario encontrar los déficits a la problemática del robot paralelo delta, tales como el peso que conlleva su estructura, el espacio de trabajo limitado, la complejidad del control al ser un robot con características fuertemente no lineales, la precisión y la repetitividad. Estas necesidades mencionadas son parte vital del proyecto, porque este nos servirá como punto de comparación de en donde se encuentre el diseño y adonde se planea llegar con él.

Respecto al déficit del peso, al tener una mayor masa implica complejidad en la parte dinámica del robot ya que se crean altas inercias,

también se vería afectada su portabilidad ya que no sería fácil transportar el manipulador robótico de un lugar a otro y en consecuencia esto reflejaría mayores gastos económicos.

El déficit del espacio de trabajo, es una parte en la que muchas industrias se ven afectadas, al adquirir un robot es necesario determinar el área que ocupara en el lugar de trabajo y a este añadirles mayor espacio debido a factores de seguridad, el tener un espacio de trabajo restringido conlleva problemas para el tránsito de los empleados y además puede crear distracciones.

El déficit del control, al ser un robot con una configuración compleja, las técnicas de control se ven más limitadas y esto crea problemas en su programación.

3.4.2. Definición del problema

A partir de dichas necesidades se desarrollará la definición del problema; con estas se obtendrá una perspectiva más concreta de adonde se quiere llegar con él. Luego, se analizará algunos aspectos globales del robot tales como el ámbito al que está dirigido y la tarea que realizará.

En robot paralelo delta que se diseñara, estará dirigido tanto al área de estudio como al área industrial, con esto se pretende que los estudiantes sean capaces de interactuar con el manipulador robotice y así comprender como funciona, esto implica que robot tendrá una arquitectura abierta, para que así a futuro se logren mejoras o se implementen más funciones. También, para el sector industrial se espera que realice la tarea de clasificación de objetos, es decir que se tiene que tener en cuenta las facilidades para que el robot pueda

clasificar los diferentes objetos que la empresa requiera. La tarea que va a desarrollar, conlleva que se pueda realizar automáticamente o con interacción del usuario.

El problema parte en cómo es que el diseño será capaz de satisfacer estos ámbitos, es aquí en donde a partir de las necesidades y teniendo en cuenta los sectores a los que está enfocado se puede representar las especificaciones correctas.

3.4.2.1. Análisis de Especificaciones

Con la información necesaria ya obtenida se procederá a plantear diferentes tipos de soluciones que sean capaces de cubrir con las necesidades. A continuación, se planteará todas las especificaciones que se requieran para llevar a cabo el diseño, es importante que en esta sección se dejen en claro las limitaciones del diseño y las diferentes funcionalidades, ya que si se piensa en un diseño ostentoso esto implicaría grandes costos económicos y si se piensa en un diseño simple este implicaría que los objetivos propuestos no sean logrados.

Tabla 3.1. Lista de Exigencias

Características	Deseos o Exigencias	Condiciones
Aplicación	E	El robot deberá ser diseñado de tal manera que pueda ser capaz de realizar movimientos capaces de clasificar objetos.
	D	Debe ser posible configurar al robot para que realice diferentes técnicas de clasificación dependiendo del objeto.
	E	Deberá contar con una arquitectura abierta para que así se puedan realizar mejoras.
Geometría	D	El espacio de trabajo del robot debe tener una altura mayor de 0.45 m y un radio mayor de 0.4 m.
	E	La estructura en la cual ira montada el robot debe ser mayor al espacio de trabajo del robot.
	E	Los componentes necesarios para la operatividad del robot no deben interrumpir el desplazamiento del robot.
Cinemática	E	El manipulador robótico deberá contar con 3 GDL.
	E	Las articulaciones deben ser rotacionales y esféricas.
	E	La velocidad debe ser variable dependiendo del tipo de acción a realizar y también debe ser manipulada por el usuario.
	E	La posición del efector final estará determinada por algoritmos de control.
Fuerzas	D	El robot debe ser estable y controlar las fuerzas de sus movimientos dependiendo de la aplicación.
	E	Debe contar con la suficiente fuerza para desplazar objetos con un peso máximo de 300g.
	D	Debe responder correctamente frente a las perturbaciones que se puedan presentar.
	E	El peso total del manipulador robótico, no deberá sobrepasar los 5 Kg.
Energía	E	La fuente de alimentación de manipulador deberá ser estable.
	E	Los actuadores del robot deberán contar con un suministro de energía que les permita realizar correctamente sus movimientos.
Señales	E	Las señales de control emitidas deben ser digitales.
	E	Deberá contar con indicadores visuales de los diferentes fenómenos físicos que se presenten en la operación del robot.
	D	Deberá poder integrarse más tipos de señales en caso se requiera alguna modificación.
Seguridad	E	La carga a soportar del manipulador no debe exceder los 300g para proteger a las actuadores de sobreesfuerzos.
	E	Los actuadores deben contar con grado de protección IP65, de protección contra polvo y humedad.
	D	El sistema de control debe de contar con alarmas visuales y sonoras capaces de identificar y dar a conocer las posibles fallas.
Ergonomía	E	El usuario que manipule el robot, debe de llevar a cabo dicha tarea en una posición adecuada.
	E	El tablero de operación deberá estar ubicado en un campo visual adecuado para el usuario.

Fabricación	E	El material a utilizar debe soportar las fuerzas generadas por el robot, ser ligero y fácil de maquinar.
	D	El proceso debe ser llevado acabo con máquinas que garanticen la exactitud y simetría del robot.
	D	La fabricación deberá lograrse respetando los márgenes de error permitidos según norma.
Control	E	El tipo de control debe ser en lazo cerrado y deberá ser lo más óptimo posible
	D	El robot deberá operar de manera automática como también manual.
	E	Se deberá usar micro controladores para así hacer más versátil el robot y no estar limitados al uso de Software especiales.
Montaje	E	El manipulador robótico debe ser armado según los planos de montaje.
	D	Los actuadores, articulas y demás no deberían presentar dificultades al momento de su instalación.
Transporte	E	Tanto como el manipulador robótico y la estructura a la cual estará montada deberá ser portátil y de fácil transporte.
Uso	E	El manipulador robótico deberá operar bajo condiciones normales
	E	Deberá de contar con colores que eviten la distracción del usuario.
	E	No deberá exceder los 55 dB, según la OMS.
Mantenimiento	E	Deberá asegurar un fácil mantenimiento mecánico y eléctrico.
Costos	E	Los costos de adquisición de materiales y manufactura deberán ser lo más bajo posibles sin restarle calidad.

Fuente: *Elaboración propia.*

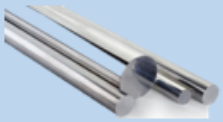
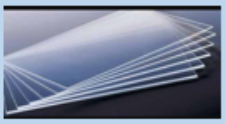
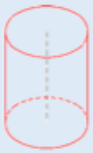
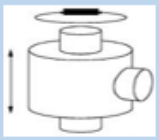
3.4.2.2. Estructura de Funciones

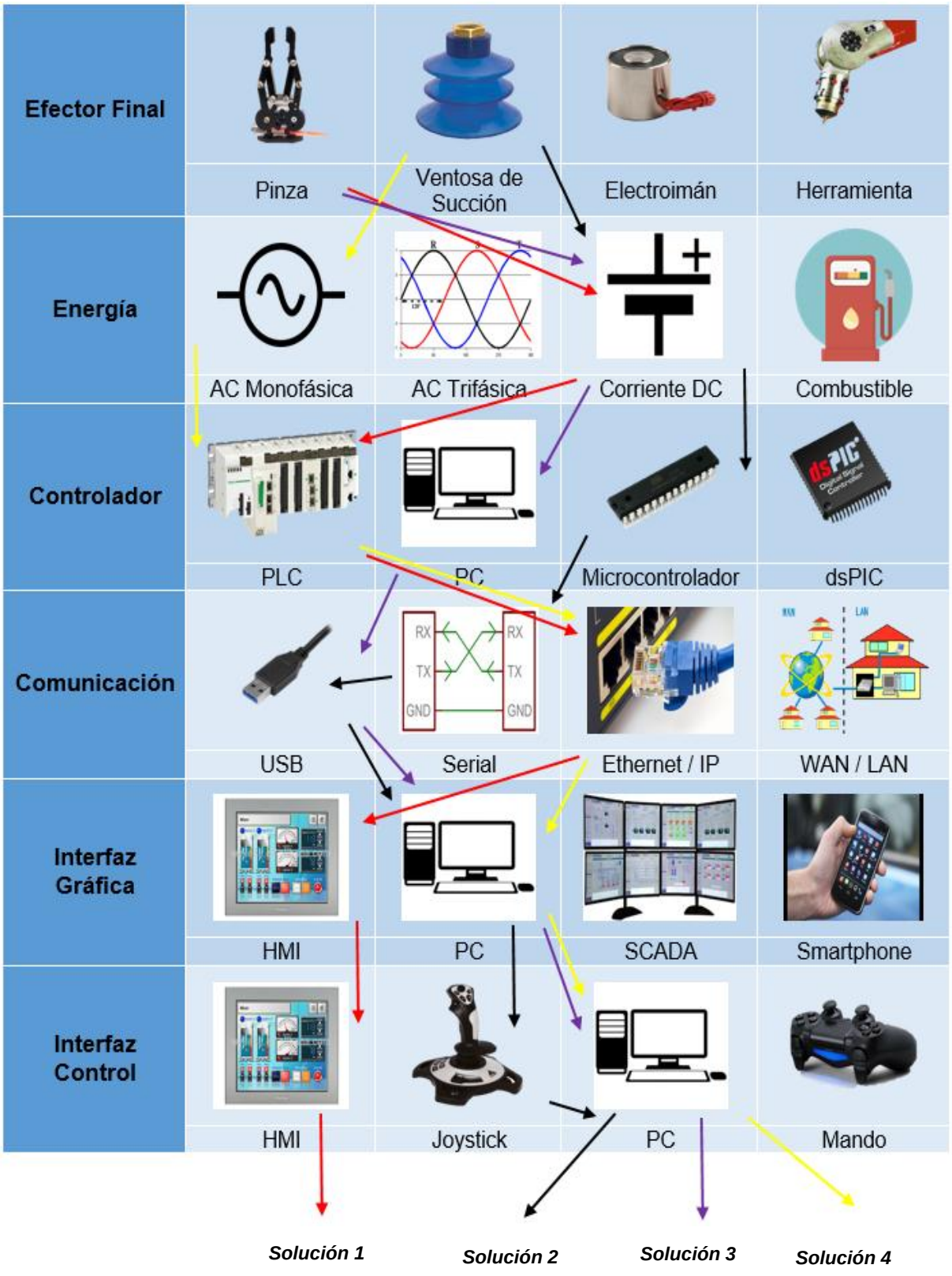
Una vez definida la lista de exigencias se determina funciones basadas en las características propuestas, para esto se dividirá en la mecánica, electrónica y control. Para entender cómo se relacionan estas funciones es necesario el uso de una matriz morfológica para poder plantear diferentes soluciones y así escoger la mejor.

3.4.2.2.1. Matriz Morfológica

En la siguiente matriz, se podrá apreciar las diferentes soluciones que se pueden plantear para el diseño, estas soluciones se analizaran una por una, se tomaran en cuenta

sus ventajas y desventajas y se procederá a escoger la solución más viable.

Funciones Parciales	Alternativa N°1	Alternativa N°2	Alternativa N°3	Alternativa N°4
Material	 Aluminio	 Policarbonato	 Plástico PLA o ABS	 Nylon
Forma	 Cuadrada	 Cilíndrica	 Rectangular	 Mixta
Articulación	 Rotacional	 Esférica	 Cilíndrica	 Tornillo
Actuador	 Motor DC	 Servomotor	 Motor Paso a Paso	 Pistones Hidráulicos
Transmisión	 Faja - Polea	 Engranajes	 Acople	 Tornillo sin Fin



Fuente: *Elaboración Propia.*

3.4.2.2.2. Evaluación de Conceptos de Soluciones

En esta parte del diseño se analizarán todas las soluciones que resultaron de la matriz morfológica, estas deben satisfacer de satisfacer tanto el criterio técnico como el criterio económico.

Tabla 3.2. Evaluación de Criterios Técnicos

Formato de Evaluación de Conceptos de Solución - Criterios Técnicos											
Valor Técnico		**Solución N°1**		**Solución N°2**		**Solución N°3**		**Solución N°4**		**Solución Ideal**	
Criterios	**g**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**
Función	5	3	15	4	20	1	5	3	15	5	25
Seguridad	3	3	9	3	9	3	9	2	6	5	15
Rapidez	2	2	4	4	8	2	4	2	4	5	10
Rigidez	3	4	12	3	9	3	9	3	9	5	15
Estabilidad	3	2	6	3	9	4	12	4	12	5	15
Manipulación	4	2	8	4	16	3	12	2	8	5	20
Portabilidad	2	3	6	5	10	2	4	1	2	5	10
Calidad	4	2	8	4	16	4	16	3	12	5	20
Fabricación	4	3	12	3	12	2	8	3	12	5	20
Complejidad	3	2	6	4	12	3	9	1	3	5	15
Confiabilidad	3	3	9	3	9	2	6	2	6	5	15
Puntaje Máximo	**36**	**29**	**95**	**40**	**130**	**29**	**94**	**26**	**89**	**55**	**180**
Valor Técnico		**0.53**		**0.72**		**0.52**		**0.49**		**1**	

Fuente: *Elaboración propia.*

Tabla 3.3. Evaluación de Criterios Económicos

Formato de Evaluación de Conceptos de Solución - Criterios Económicos											
Valor Técnico		**Solución N°1**		**Solución N°2**		**Solución N°3**		**Solución N°4**		**Solución Ideal**	
Criterios	**g**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**	**p**	**g*p**
Número de Piezas	3	3	9	4	12	2	6	3	9	5	15
Fabricación	5	3	15	3	15	2	10	4	20	5	25
Costo de Tecnología	5	5	25	3	15	3	15	1	5	5	25
Fácil Adquisición	3	4	12	2	6	4	12	2	6	5	15
Montaje	2	3	6	5	10	3	6	1	2	5	10
Mantenimiento	2	3	6	4	8	3	6	2	4	5	10
Puntaje Máximo	**20**	**21**	**73**	**21**	**66**	**17**	**55**	**13**	**46**	**30**	**100**
Valor Económico		**0.73**		**0.66**		**0.55**		**0.46**		**1**	

Fuente: *Elaboración propia.*

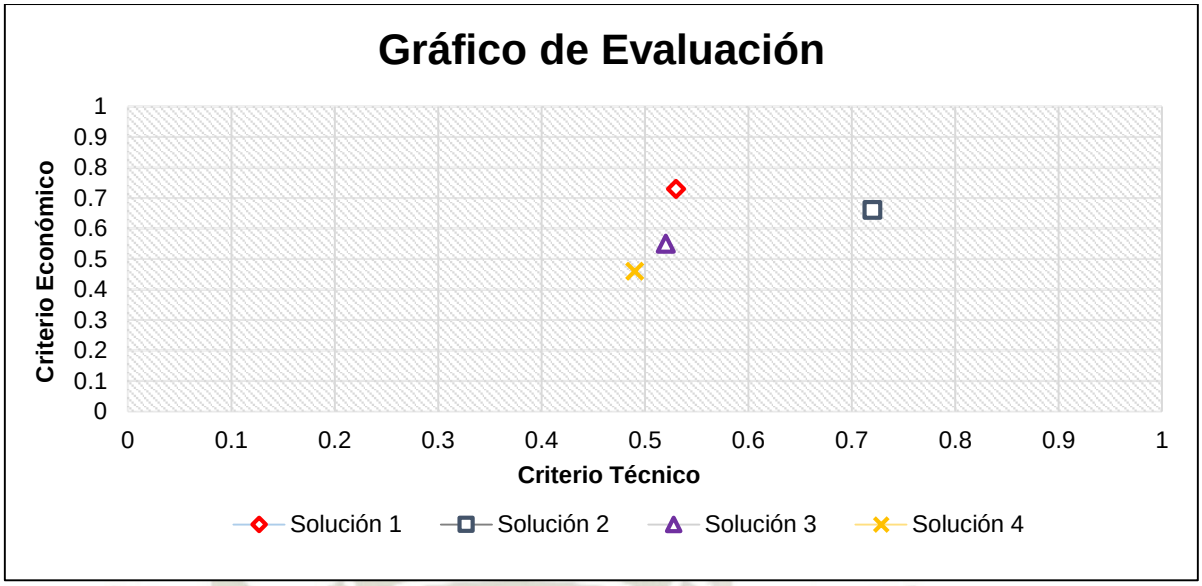


Figura 3.5. Diagrama de Evaluación Técnico – Económica
***Fuente**: Elaboración propia.*

De acuerdo al diagrama presentado, se denota que tanto la solución 1 y 2 son las más viables, por un lado, la solución 1 presenta un indicador bajo de criterio técnico, pero este es compensado en una mejor viabilidad económica. Sin embargo, la solución 2 presenta con grandes creces un mejor indicador en cuanto a la parte técnica, pero gracias a esto presenta un incremento en el costo.

Como ya se había mencionado anteriormente se necesita que el diseño se adapte mejor a las necesidades, pero sin perder la calidad de este, es por eso que se escogerá la solución 2, ya que a pesar de contar con un índice económico más elevado este se verá reflejado en la mejora técnica, de esta manera se obtiene un diseño que respetara más las exigencias.

3.4.3. Solución Óptima

La solución más óptima está compuesta de Plástico, ya que es fácil de maquinar y liviano, dependiendo del tipo de plástico, también tendrá una forma agradable al usuario y no típicas figuras geométricas. Para la función del movimiento, será con articulaciones rotacionales y esféricas ya que están son necesarias para la configuración del robot paralelo delta. El actuador que brindara el movimiento a los eslabones y articulaciones, serán servomotores ya que cuenta con engranajes en su interior y esto les permite tener un mayor torque, también son más accesible en cuanto a su control y dependiendo del tipo de servomotor se puede realimentar datos necesarios para el control del manipulador robótico.

El efector final, será una ventosa de succión, ya que este dispositivo es ligero y con mayor facilidad de ensamblar que una garra mecánica. Se resalta que el uso de servomotores y ventosa de succión requiere una fuente AC-DC debido a que el suministro estándar en la localidad es 220V a 60 Hz. También con este tipo de fuente se puede suministrar energía para el uso del controlador, en este caso se optó por un micro controlador ya que una computadora implicaría el uso de software especial y el diseño pierde peso al estar limitado a este tipo de Software.

La comunicación será de tipo serial ya que es la más elemental y más accesible para el micro-controlador, también se pueden encontrar en las PC lo cual facilitaría el uso para que esta sirva como una interfaz tanto grafica como de control.

3.4.3.1. Bosquejo de Mejor Solución

Habiendo planteado la solución más óptima se continuará con un bosquejo que nos ayude a plasmar mejor la idea de cómo debería verse el robot teniendo en cuenta las especificaciones dadas.

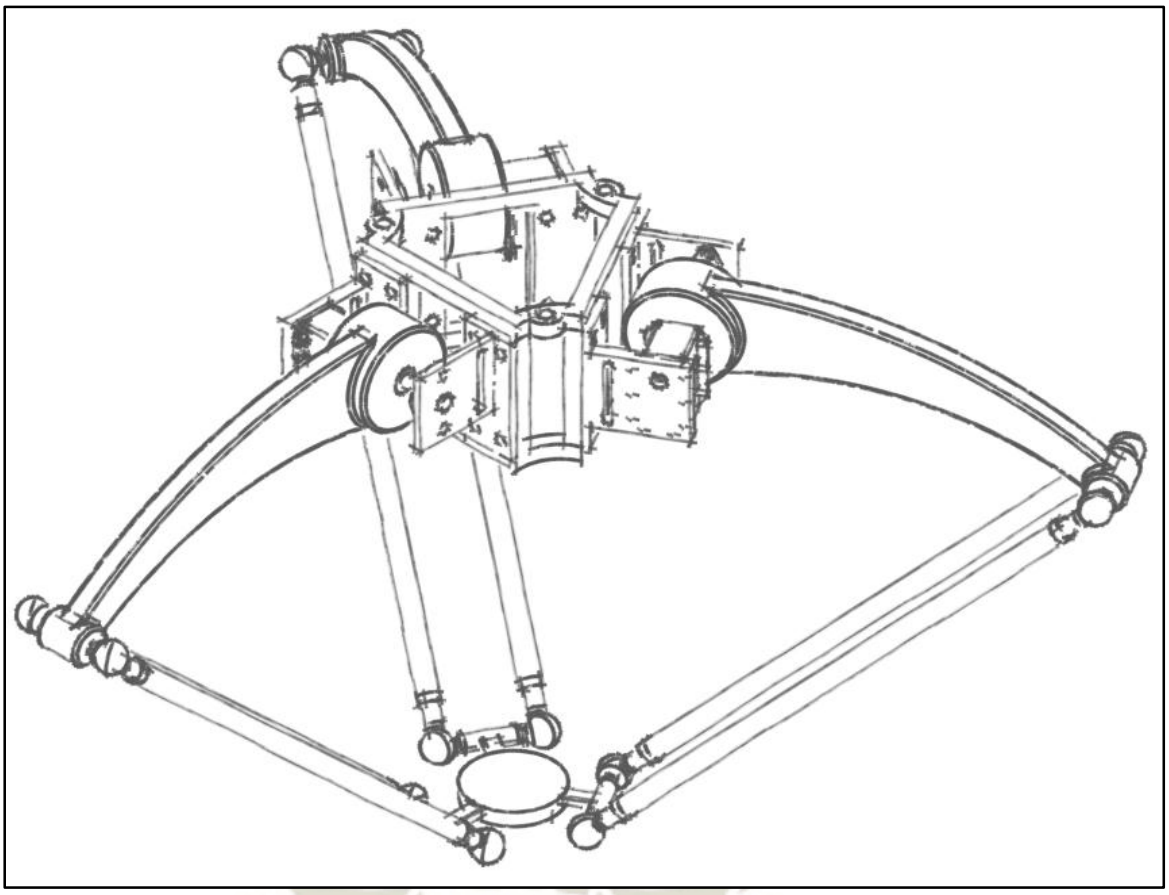


Figura 3.6. Bosquejo de Solución más Óptima
Fuente: *Elaboración propia*

3.4.4. Síntesis

Luego de un análisis exhaustivo sobre las diferentes particularidades del manipulador robótico, se procede a realizar la síntesis del diseño, en esta parte del diseño se busca representar el robot en modelos matemáticos para tener una idea como podría manifestarse el producto en un escenario real, es aquí donde se realizan diferentes pruebas y métodos que nos ayuden a describir el comportamiento del robot, en

esta fase del diseño se buscara definir la idea final de adonde se quiere llegar con el diseño.

Este parte del apartado se trabajará con el análisis cinemático, luego se procederá con el análisis dinámico. A partir, de estos análisis se obtiene los datos necesarios para el diseño mecánico. Teniendo en cuenta todos estos parámetros se continuará con un análisis eléctrico y finalmente la electrónica.

3.5. Análisis Cinemático

El análisis cinemático se divide en dos partes, cuando se conoce la posición cartesiana a la cual se quiere llevar al efector final y se necesita saber la posición que necesitaran los actuadores para llevar a cabo dicha operación (Cinemática Inversa). Cuando se sabe la posición en la cual se encuentran los actuadores y se necesita saber el lugar en el cual se encuentra el efector final (Cinemática Directa). Si se observa la Figura 3.7, este nos indicara tanto las posiciones angulares y las cartesianas necesarias para trabajar.

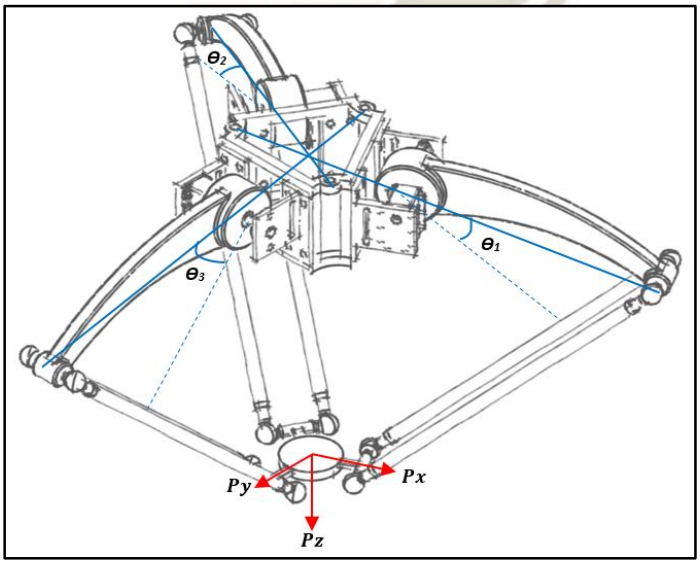


Figura 3.7. Parámetros de Posiciones Angulares y Cartesianas
Fuente: *Elaboración propia.*

3.5.1. Asunciones del Modelo

A diferencia de la mayoría de manipuladores, la configuración de un robot paralelo es más compleja ya que es una cadena cinemática cerrada, esto quiere decir que cualquier movimiento de un eslabón afectara a otro sin importar la magnitud de este.

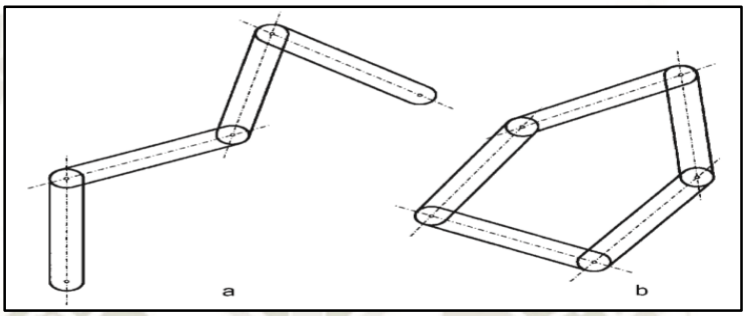


Figura 3.8. a) Cadena Cinemática Abierta b) Cadena Cinemática
Fuente: *(Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007)*

En la Figura 3.8, se aprecia con mayor claridad las diferencias entre un mecanismo con cadena cinemática abierta y otra con una cadena cinemática cerrada.

Debido a esta configuración se realizará algunas asunciones muy importantes que nos ayudaran de alguna manera a simplificar la dificultad del análisis y estas son:

- La base móvil deberá permanecer paralela a la base fija, por lo tanto, su orientación a los ejes perpendiculares es cero. De esta manera los antebrazos pueden ser reemplazados con líneas sin afectar la cinemática del robot.
- Para poder hallar las posiciones se asumirá que los antebrazos giran libremente respecto a las rotulas.

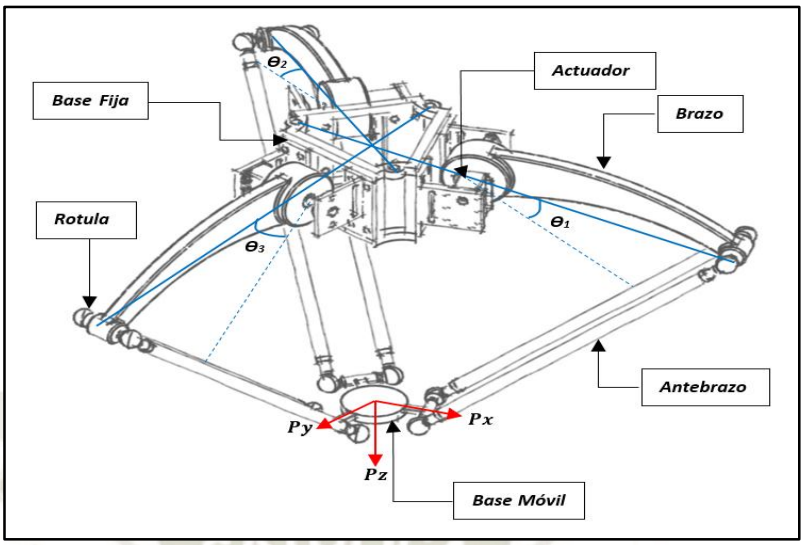


Figura 3.9. Partes principales del robot paralelo delta
Fuente: *Elaboración propia.*

3.5.2. Cinemática Directa

Para poder resolver el problema cinemático directo, existen diversos métodos matemáticos que se pueden usar como por el ejemplo, el uso de matrices de rotación o pares de rotación, que implican el uso de vectores o el ya conocido algoritmo de Denavit Hartenberg. Este algoritmo, permite encontrar de manera sistemática la posición del efector final, pero está restringido solo al uso de cadenas cinemáticas abiertas. Ya que la configuración del robot paralelo es de tipo cerrada, este algoritmo queda descartado por lo que se procederá con métodos vectoriales.

Primero es necesario conocer de qué manera están orientados los ejes cartesianos para así poder continuar con el análisis, esta orientación queda a libre decisión del diseñador.

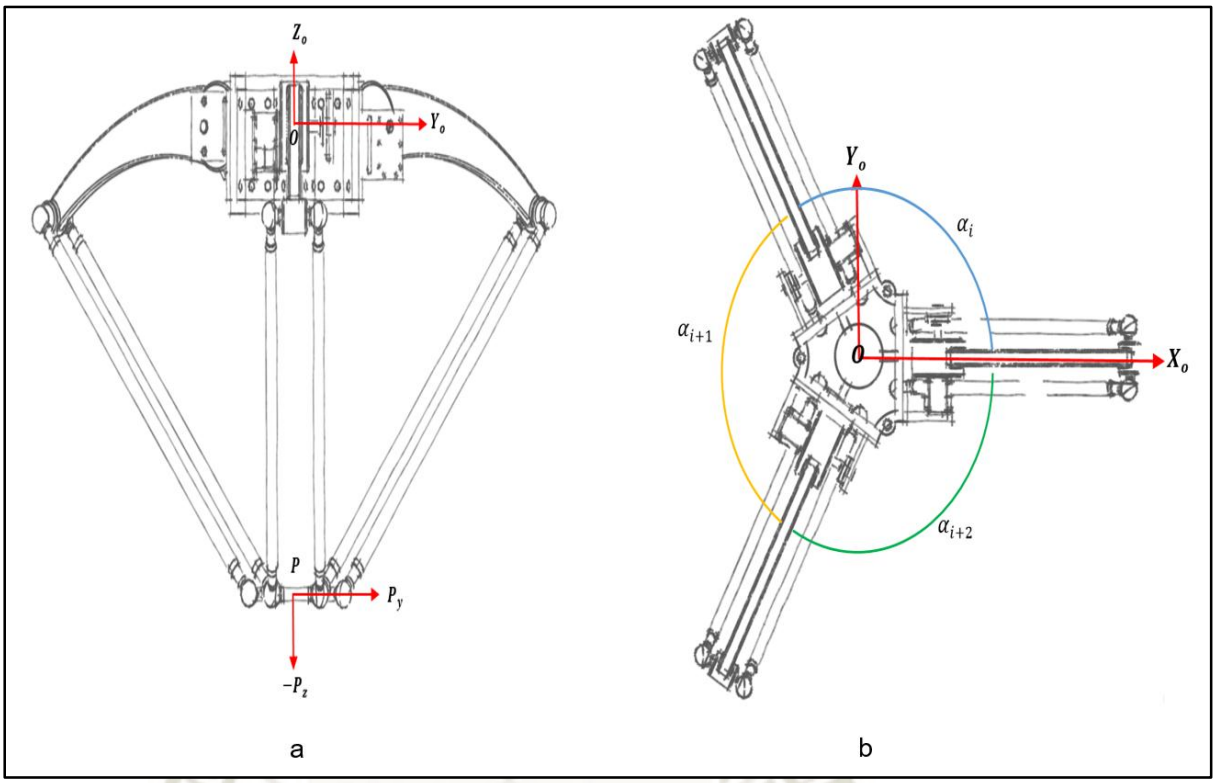


Figura 3.10. a) Vista Frontal b) Vista Superior
Fuente: *Elaboración propia.*

En la Figura 3.10 se puede apreciar el sentido de la posición de la base fija como también de la base móvil, también se observa el desfase simétrico de un ángulo α_i que se necesita para poder llegar a la configuración delta. Es importante plantear estas posiciones ya que es de esta manera que parte el análisis vectorial para poder encontrar la posición de P (Px, Py, Pz).

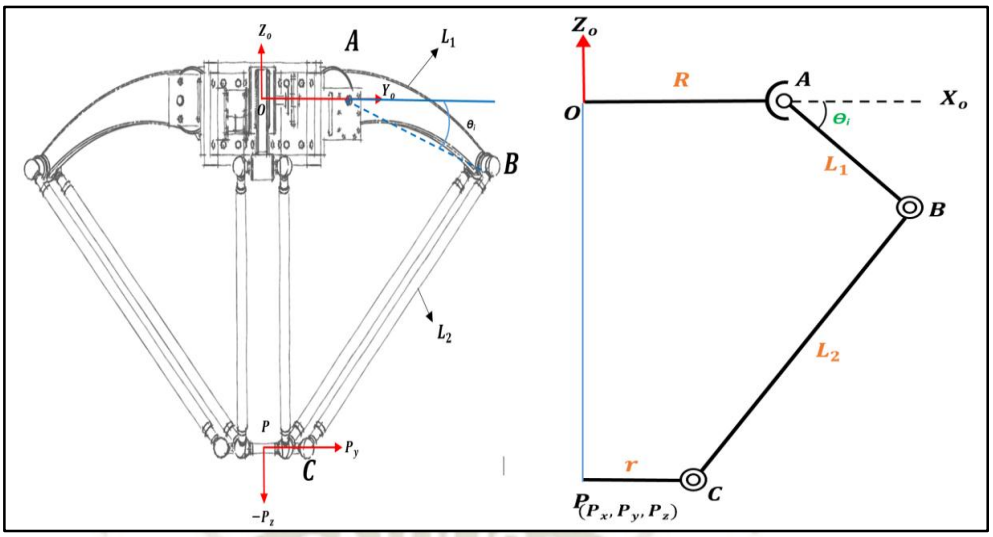


Figura 3.11. Representación esquemática del robot
Fuente: *Elaboración propia.*

En donde:

α_i: *Ángulo de rotación respecto al eje* Z_o

Θ_i: *Ángulo del actuador*

R: *Distancia del eje de la base fija a la articulación*

r: *Distancia del eje de la base móvil la articulación*

L_1: *Distancia del brazo*

L_2: *Distancia del antebrazo*

Una vez obtenido el esquema se procede con el análisis, de esta manera se obtiene lo siguiente:

$$\alpha_i = [0°, 120°, 240°] \quad (3.1)$$

A partir de la variable expresada en (3.1) y sabiendo que la configuración es paralela se entiende que las longitudes de todos los brazos y antebrazos son las mismas, esto también implica que la distancias R y r son las mismas, en todo momento, las longitudes tanto del brazo como el antebrazo son las mismas para todos los eslabones del robot. *En tanto,* Θ_i *será el ángulo del actuador*

asociado a cada brazo. A simple vista se llega a la siguiente conclusión.

$$R + L_1 + L_2 + r = P \tag{3.2}$$

Esto también se puede expresar como:

$$P - R - r - L_1 = L_2 \tag{3.3}$$

Se tomó en cuenta que la variable en cuestión sea L_2, ya que esta es el eslabón que se mueve con mayor libertad debido a sus articulaciones esféricas.

Expresado en forma vectorial se sabe que:

$$\overline{OP} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

$$\overline{OA} = R$$

$$\overline{AB} = L_1$$

$$\overline{BC} = L_2$$

$$\overline{CP} = r$$

Teniendo en cuenta su forma vectorial, se trabajará de la misma manera para el resto de los brazos, para esto hay que tener en cuenta la rotación del eje y considerar que el único vector igual para los tres brazos y antebrazos es el de la posición final.

$$\overline{OP} - \overline{OA} - \overline{AB} - \overline{CP} = \overline{BC} \tag{3.4}$$

$$\begin{aligned} {}_P T^O(p) - Rot(\alpha_i)({}_O T^A(R) - {}_A T^B(L_1) Rot(\theta_i) - {}_C T^P(r)) \\ = {}_C T^B(L_2) Rot(\beta, \gamma) \end{aligned} \tag{3.5}$$

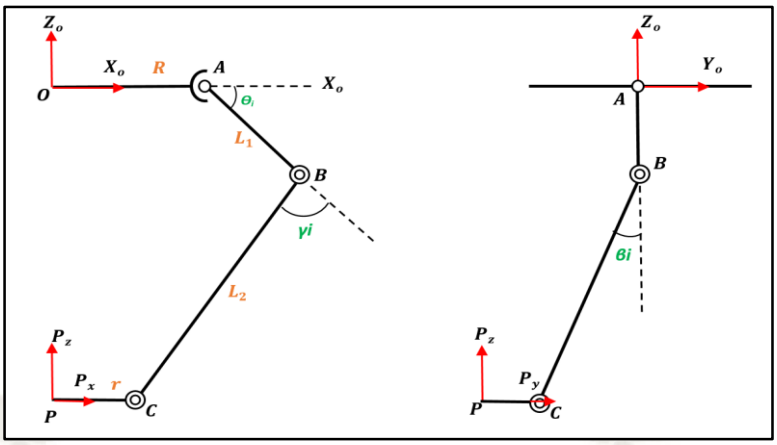


Figura 3.12. Representación de Vectores desde otro punto de vista
Fuente: *Elaboración propia.*

En la Figura 3.12 se ve desde otra perspectiva que el antebrazo al estar unido a una articulación esférica crea ángulos pasivos (β_i ; γ_i), ya que al principio del modelo se asumió que el antebrazo giraría libremente respecto a la rótula, estos ángulos pasivos serán obviados del modelo cinemático. Sin embargo, el antebrazo se verá afectado por los movimientos generados del TCP.

Es importante definir los datos de los vectores de cada punto del robot, teniendo en cuenta las representaciones dadas se obtiene lo siguiente:

$$\overline{OA} = \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$$

$$\overline{AB} = \begin{bmatrix} L_1 * \cos(\theta_i) \\ 0 \\ L_2 * \sin(\theta_i) \end{bmatrix}$$

$$\overline{CP} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}$$

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

Definir el vector BC, es el punto al cual se quiere llegar, por la cual hay diferentes maneras de hacerlo, en este caso nos basaremos en dos métodos, por un lado, usaremos la norma Euclidiana que precisamente trata de las longitudes de vectores en espacios normados y por otro lado la trilateración que nos permitirá encontrar el punto en común de las articulaciones del robot. Ambas son necesarias para posteriores análisis y serán de gran ayuda con las asunciones tomadas.

Según la norma Euclidiana la función:

$$x \to \|x\|_p, de\ \mathbb{K}^n\ en\ \mathbb{R}$$

Esto quiere decir que:

$$\|x\|_p = \left(\sum_{k=1}^{n} |x_k|^2\right)^{\frac{1}{2}}, \forall x = (x_1, \dots, x_n) \in\ \mathbb{K}^n$$

Siendo $\mathbb{K}^n$ tanto un valor que denotara tanto números reales ($\mathbb{R}$), como números complejos ($\mathbb{C}$). De la anterior expresión se puede asumir que:

$$\|\overline{BC}\|_2{}^2 = {L_2}^2 \qquad (3.6)$$

Al ya tener definido el vector de BC, es momento de relacionar cada brazo y antebrazo con un punto en común. Este punto serian las posiciónes que pueden alcanzar ya que la posición final es el medio en comun que une a los eslabones a una cadena cinematica cerrada. Como se asumio al principio del analisis y basandonos en

el principio de la trilateración. El antebrazo al estar unido a una articulación esferica, rotara libremente respecto del vector B, haciendo esto se obtiene una esfera en cada articulación conectada al brazo con un radio L_2

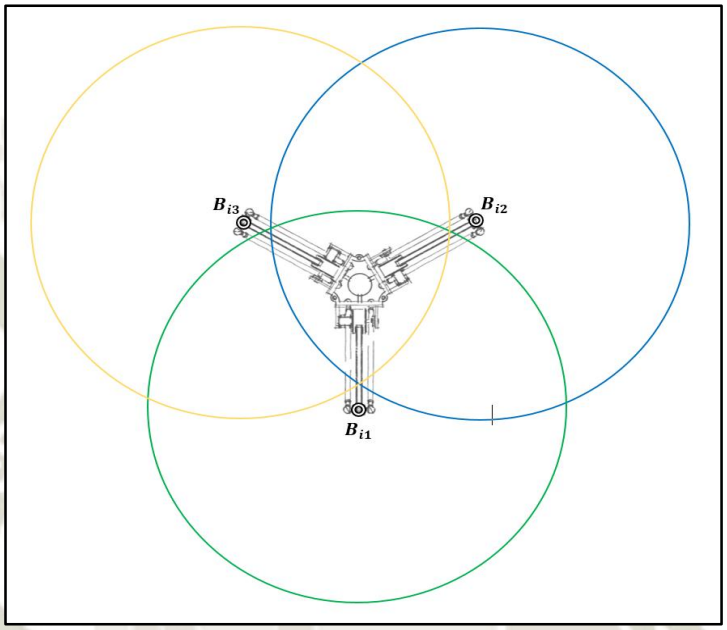


Figura 3.13. Intersección de esferas generadas a partir de las articulaciones esféricas
Fuente: *Elaboración propia.*

En la Figura 3.13 se observa que existe una zona donde las 3 esferas intersecan, esta intersección indica los puntos donde es posible que el efector final llegue. Teniendo en cuenta dicha intersección es posible plantear lo siguiente:

$$R - r = R_g$$

Esta expresión cobra sentido si se asume que estas distancias son al eje de la base móvil, al ser constantes se hace una simplificación que denota la distancia entre el punto P al punto A. Sabiendo esto se plantea que:

$$\overline{BC} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - Rot_z(\alpha_i)\left(\begin{bmatrix} R_g \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} L_1\cos(\theta_i) \\ 0 \\ -L_1\sin(\theta_i) \end{bmatrix}\right) \qquad i = 1,2,3 \tag{3.7}$$

Resolviendo la ecuación planteada se obtiene que el vector BC es:

$$\overline{BC} = \begin{bmatrix} P_x - \cos(\alpha_i) * (R_g + L_1\cos(\theta_i)) \\ P_y - \sin(\alpha_i) * (R_g + L_1\cos(\theta_i)) \\ P_z + L_1\sin(\theta_i) \end{bmatrix} \qquad i = 1,2,3 \tag{3.8}$$

Finalmente definida el vector BC es necesario encontrar una ecuación que defina la relación de las posiciones finales (P_x, P_y, P_z), con el resto de datos geométricos del robot. Usando la norma Euclidiana se define que la suma de cada elemento del vector elevada al cuadrado será la longitud del vector a encontrar, es decir:

$$(P_x - \cos(\alpha_i) * (R_g + L_1\cos(\theta_i)))^2 + (P_y - \sin(\alpha_i) * (R_g + L_1\cos(\theta_i)))^2 + (P_z + L_1\sin(\theta_i))^2 = {L_2}^2 \tag{3.9}$$

De la ecuación (3.9) se puede observar que el resultado de la magnitud es la longitud del antebrazo. Esto guarda relación con la ecuación de la esfera la cual es:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r^2 \tag{3.10}$$

Relacionando la ecuación (3.10) con la (3.9) se tiene:

$$x_i = \cos(\alpha_i) * \left(R_g + L_1\cos(\theta_i)\right) \qquad i = 1,2,3$$

$$y_i = \sin(\alpha_i) * (R_g + L_1\cos(\theta_i)) \qquad i = 1,2,3$$

$$z_i = L_1\sin(\theta_i) \qquad i = 1,2,3$$

$$r^2 = {L_2}^2$$

Planteando las 3 ecuaciones de las esferas:

$$(P_x - x_1)^2 + \left(P_y - y_1\right)^2 + (P_z - z_1)^2 = {L_2}^2 \quad (3.11)$$

$$(P_x - x_2)^2 + \left(P_y - y_2\right)^2 + (P_z - z_2)^2 = {L_2}^2 \quad (3.12)$$

$$(P_x - x_3)^2 + \left(P_y - y_3\right)^2 + (P_z - z_3)^2 = {L_2}^2 \quad (3.13)$$

Reemplazando la expresión anterior se obtendrá una expresión común:

$$\left(P_x - \cos(\alpha_1)\left(R_g + L_1\cos(\theta_1)\right)\right)^2 + \left(P_y - \sin(\alpha_1)\left(R_g + L_1\cos(\theta_1)\right)\right)^2 + (P_z - L_1\sin(\theta_1))^2 = \boldsymbol{L_2}^2$$

$$\left(P_x - \cos(\alpha_2)\left(R_g + L_1\cos(\theta_2)\right)\right)^2 + \left(P_y - \sin(\alpha_2)\left(R_g + L_1\cos(\theta_2)\right)\right)^2 + (P_z - L_1\sin(\theta_2))^2 = \boldsymbol{L_2}^2$$

$$\left(P_x - \cos(\alpha_3)\left(R_g + L_1\cos(\theta_3)\right)\right)^2 + \left(P_y - \sin(\alpha_3)\left(R_g + L_1\cos(\theta_3)\right)\right)^2 + (P_z - L_1\sin(\theta_3))^2 = \boldsymbol{L_2}^2$$

Al ser 3 ecuaciones con 3 variables, esta tiene solución, pero resultará muy engorrosa por eso se tratará de reducir lo máximo posible. De las ecuaciones (3.11, 3.12, 3.13) se obtiene que:

$$\left({P_x}^2 - 2P_x x_1 + {x_1}^2\right) + \left({P_y}^2 - 2P_y y_1 + {y_1}^2\right) + \left({P_z}^2 - 2P_z z_1 + z^2\right) = {L_2}^2$$

$$\left({P_x}^2 - 2P_x x_2 + {x_2}^2\right) + \left({P_y}^2 - 2P_y y_1 + {y_2}^2\right) + \left({P_z}^2 - 2P_z z_2 + z^2\right) = {L_2}^2$$

$$\left({P_x}^2 - 2P_x x_3 + {x_3}^2\right) + \left({P_y}^2 - 2P_y y_3 + {y_3}^2\right) + \left({P_z}^2 - 2P_z z_3 + z^2\right) = {L_2}^2$$

Reemplazando:

$${x_i}^2 + {y_i}^2 + {z_i}^2 = u_i$$

De esta manera:

$$\left({P_x}^2 - 2P_x x_1\right) + \left({P_y}^2 - 2P_y y_1\right) + \left({P_z}^2 - 2P_z z_1\right) = {L_2}^2 - u_1 \quad (3.14)$$

$$\left({P_x}^2 - 2P_x x_2\right) + \left({P_y}^2 - 2P_y y_2\right) + \left({P_z}^2 - 2P_z z_2\right) = {L_2}^2 - u_2 \quad (3.15)$$

$$\left({P_x}^2 - 2P_x x_3\right) + \left({P_y}^2 - 2P_y y_3\right) + \left({P_z}^2 - 2P_z z_3\right) = {L_2}^2 - u_3 \quad (3.16)$$

Reemplazando (3.14) en (3.15)

$$P_x(x_1-x_2) + P_y(y_1-y_2) + P_z(z_1-z_2) = \frac{(u_1 - u_2)}{2} \quad (3.17)$$

Reemplazando (3.14) en (3.16)

$$P_x(x_1-x_3) + P_y(y_1-y_3) + P_z(z_1-z_3) = \frac{(u_1 - u_3)}{2} \quad (3.18)$$

De la expresión (3.17) se tiene que:

$$P_y = \frac{\frac{(u_1 - u_2)}{2} - P_x(x_1-x_2) - P_z(z_1-z_2)}{(y_1-y_2)} \quad (3.19)$$

$$P_x = \frac{\frac{(u_1 - u_2)}{2} - P_y(y_1-y_2) - P_z(z_1-z_2)}{(x_1-x_2)} \quad (3.20)$$

Reemplazando (3.19) en (3.18)

$$P_x = P_z\frac{(z_1-z_2)(y_1-y_3) - (z_1-z_3)(y_1-y_2)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)} + \frac{1}{2}\left(\frac{(u_1-u_3)(y_1-y_2) - (u_1-u_2)(y_1-y_3)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)}\right) \quad (3.21)$$

Reemplazando (3.20) en (3.18)

$$P_y = P_z\frac{(z_1-z_3)(x_1-x_3) - (z_1-z_2)(x_1-x_3)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)} + \frac{1}{2}\left(\frac{(u_1-u_2)(x_1-x_3) - (u_1-u_3)(x_1-x_2)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)}\right) \quad (3.22)$$

Finalmente, al reemplazar las expresiones (3.21) y (3.22) en la ecuación (3.14) se obtiene la expresión para la posición en Z:

$$\left(P_z\frac{(z_1-z_2)(y_1-y_3) - (z_1-z_3)(y_1-y_2)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)} + \frac{1}{2}\left(\frac{(u_1-u_3)(y_1-y_2) - (u_1-u_2)(y_1-y_3)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)}\right)\right)^2$$

$$-$$

$$2\left(P_z\frac{(z_1-z_2)(y_1-y_3) - (z_1-z_3)(y_1-y_2)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)}\right.$$

$$\left. + \frac{1}{2}\left(\frac{(u_1-u_3)(y_1-y_2) - (u_1-u_2)(y_1-y_3)}{(x_1-x_3)(y_1-y_2) - (x_1-x_2)(y_1-y_3)}\right)\right)x_1 +$$

$$\left(P_z\frac{(z_1-z_3)(x_1-x_3)-(z_1-z_2)(x_1-x_3)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right.$$

$$\left.+\frac{1}{2}\left(\frac{(u_1-u_2)(x_1-x_3)-(u_1-u_3)(x_1-x_2)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right)\right)^2 \quad (3.23)$$

$$-2\left(P_z\frac{(z_1-z_3)(x_1-x_3)-(z_1-z_2)(x_1-x_3)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right.$$

$$\left.+\frac{1}{2}\left(\frac{(u_1-u_2)(x_1-x_3)-(u_1-u_3)(x_1-x_2)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right)\right)y_1$$

$$+{P_z}^2-2P_zz_1-{L_2}^2+u_1=0$$

Tanto las expresiones (3.21; 3.22; 3.23) requieren que se reemplace los términos de $x_i, y_i z_i$.Al hacer esto la ecuación matemática se tornara más grande de lo que ya es. Cabe resaltar que aun después de todos los arreglos matemáticos, la expresión (3.24) es de carácter no lineal, por lo que pueden existir dos soluciones.

Reemplazando:

$$d_1=\frac{(z_1-z_2)(y_1-y_3)-(z_1-z_3)(y_1-y_2)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}$$

$$e_1=\frac{1}{2}\left(\frac{(u_1-u_3)(y_1-y_2)-(u_1-u_2)(y_1-y_3)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right)$$

$$d_2=\frac{(z_1-z_3)(x_1-x_3)-(z_1-z_2)(x_1-x_3)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}$$

$$e_2=\frac{1}{2}\left(\frac{(u_1-u_2)(x_1-x_3)-(u_1-u_3)(x_1-x_2)}{(x_1-x_3)(y_1-y_2)-(x_1-x_2)(y_1-y_3)}\right)$$

Reemplazando las expresiones anteriores en la ecuación (3.23):

$$(P_zd_1+e_1)^2-2(P_zd_1+e_1)x_1+(P_zd_2+e_2)^2-2(P_zd_2+e_2)y_1+{P_z}^2-2P_zz_1$$

$$={L_2}^2-u_1$$

$$(P_z d_1)^2 + 2P_z d_1 e_1 + {e_1}^2 - 2P_z d_1 x_1 - 2e_1 x_1 + (P_z d_2)^2 + 2P_z d_2 e_2 + {e_2}^2 - 2P_z d_2 y_1 - 2e_2 y_1 + {P_z}^2 - 2P_z z_1 - {L_2}^2 + u_1 = 0$$

$$\boldsymbol{{P_z}^2({d_1}^2 + {d_2}^2 + 1) + 2P_z(d_1(e_1 - x_1) + d_2(e_2 - y_1) - z_1) + (e_1 - x_1)^2 + (e_2 - y_1)^2 - {L_2}^2 + {z_1}^2 = 0} \quad (3.24)$$

La expresión (3.24) es la misma que la (3.23) a diferencia que se realizó cambios de variables para que se note con mayor claridad que esta es una ecuación no lineal. Al obtener esta característica, implica que existan dos soluciones, escoger una de estas soluciones dependerá luego de los limites mecánicos o geométricos que posea el robot Aun así es recomendable calcular esta respuesta con ayuda de software matemáticos debido a su extenso análisis.

3.5.3. Cinemática Inversa

En este análisis se busca una solución en la cual partiendo de las posiciones cartesianas del efector final (P_x, P_y, P_z), se encuentren las posiciones angulares que requieren los actuadores para cumplir con las exigencias en el espacio cartesiano.

Al igual que en la cinemática directa, haremos uso de la ecuación que define el espacio de esfera. Este análisis se torna más simple ya que antes se tenía que resolver 3 ecuaciones para poder hallar la solución. Sin embargo, en este caso partiendo de una sola ecuación y cambiando los datos de rotación, es posible encontrar la solución al problema cinemático inverso. Referenciándonos de la expresión (3.9), podemos

observar que todos los términos son constantes a excepción del ángulo de los actuadores y las posiciones, de esta manera es posible:

$$R_g + {P_x}^2 + {P_y}^2 + {P_z}^2 + {L_1}^2 - \Big(P_x\cos(\alpha_i) + P_y\sin(\alpha_i)\Big)\big(2R_g + 2L_1\cos(\theta_i)\big) + 2R_gL_1\cos(\theta_i) - 2L_1\,P_z\sin(\theta_i) = {L_2}^2 \tag{3.25}$$

Al igual que las expresiones de la cinemática directa, son de carácter no lineal esto hace que su solución sea más compleja, para resolver la expresión (3.25) haremos uso de identidades trigonométricas de la tal manera que el resultado es el siguiente:

$$\theta_i = 2 * tan^{-1}\left(\frac{-2P_z \pm \sqrt{4{P_z}^2 + 4{R_g}^2 - {A_i}^2 + {B_i}^2\left(1 - \frac{{R_g}^2}{{L_1}^2}\right) + B_i\left(\frac{-2R_gA_i}{L_1} - 4R_g\right)}}{-2R_g - B_i\left(\frac{R_g}{L_1} - 1\right) - A_i}\right) \tag{3.26}$$

Donde:

$$A_i = \frac{1}{L_1}({L_2}^2 - {L_1}^2 - {R_g}^2 - {P_x}^2 - {P_y}^2 - {P_z}^2$$

$$B_i = 2P_x\cos(\alpha_i) + 2P_y\sin(\alpha_i) \qquad i = 1,2,3$$

Esta solución solo existirá si:

$$4{P_z}^2 + 4{R_g}^2 - {A_i}^2 + {B_i}^2\left(1 - \frac{{R_g}^2}{{L_1}^2}\right) + B_i\left(\frac{-2R_gA_i}{L_1} - 4R_g\right) \geq 0 \tag{3.27}$$

De la expresión (3.26) es posible obtener todos los ángulos necesarios de los actuadores para que así lleguen a la misma posición del efector final. También, se puede observar que en una parte de la ecuación se tiene la expresión ($\pm$) esto quiere decir que pueden existen diversas soluciones que satisfagan el mismo resultado.

En cuanto a la expresión (3.27), debe satisfacer la misma regla en todos los brazos, de no ser así la solución que se encuentre no será tomada. También, esta expresión puede ser usada para definir el espacio de trabajo del robot de una manera más exacta.

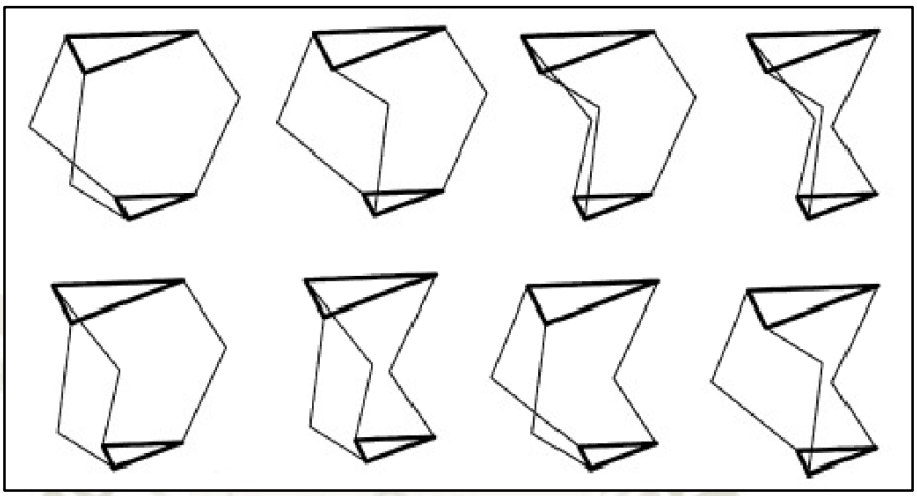


Figura 3.14. Las 8 soluciones de la cinemática inversa para un robot paralelo
Fuente: *(Liu & Wang, 2003)*

De la Figura 3.14, se observa que para una misma posición del TCP, el robot puede alcanzar distintas posiciones angulares. Sin embargo, algunas de estas soluciones quedan descartadas dependiendo de las limitaciones geométricas o mecánicas del robot que se verán más adelante.

3.5.4. Modelo Diferencial

El análisis diferencial permite establecer una relación entre las velocidades angulares de los actuadores con las velocidades cartesianas del TCP y viceversa. Esta relación es expresada por la matriz Jacobiana, la cual es una herramienta que nos permite hallar estas velocidades.

Por física sabemos que si derivamos la posición se obtiene la velocidad y si derivamos la velocidad se obtiene la aceleración. De esta manera si se deriva las posiciones cartesianas es posible hallar las velocidades cartesianas. Sin embargo, el problema parte de ahí, el robot paralelo delta al tener una cinemática directa compleja y de carácter no lineal hace que su análisis diferencial sea extenso y complejo. Aun si se derivan las expresiones (3.21; 3.22; 3.23) estas serían difíciles de analizar ya que al ser una cadena cinemática cerrada hace que aumente la complejidad del caso y que siempre se esté pendiente de las demás posiciones.

Para realizar un análisis diferencial en el robot paralelo delta partiremos de las expresiones vectoriales, para así poder hallar una relación entre las velocidades cartesianas con las velocidades articulares. Partiendo de la expresión (3.6):

$$\|\overline{BC}\|_2{}^2 - {L_2}^2 = 0 \quad i = 1,2,3 \tag{3.28}$$

La expresión (3.28) se usa como una restricción que vinculara las variables del espacio cartesiano con las variables articulares, reemplazando las variables conocidas da como resultado la expresión (3.6). Con el fin de simplificar el cálculo, asumiremos que el vector $\overline{BC}$ es igual a h_i. De esta manera resulta

$$h_i = \overline{OP} - \overline{OA} - \overline{AB} - \overline{CP} = \overline{BC} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - Rot_z(\alpha_i)\left(\begin{bmatrix} R_g \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} L_1\cos(\theta_i) \\ 0 \\ -L_1\sin(\theta_i) \end{bmatrix}\right) i \tag{3.29}$$

$$= 1,2,3$$

Debido a que se está trabajando en el espacio euclidiano se sabe que $\boldsymbol{h_i}$ es una matriz ortonormal y por ello:

$$h_i{}^T \cdot h_i = I \qquad i = 1,2,3 \tag{3.30}$$

Derivando respecto al tiempo la expresión (3.30) se tiene:

$$h_i{}^T * \dot{h}_i + \dot{h_i}{}^T * h_i = 0 \qquad i = 1,2,3 \tag{3.31}$$

Con la propiedad conmutativa del producto resulta:

$$h_i{}^T * \dot{h}_i = 0 \qquad i = 1,2,3 \tag{3.32}$$

De la expresión (3.32) reemplazando los datos de $\dot{h}_i$ ya derivados, se obtiene:

$$\begin{aligned} \dot{h}_i &= \begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{P}_z \end{bmatrix} - Rot_z(\alpha_i) \begin{bmatrix} -L_1 \sin(\theta_i) \\ 0 \\ L_1 \cos(\theta_i) \end{bmatrix} * \left(\dot{\theta}_i\right) \\ &\approx \dot{P}_n - b_i \dot{\theta}_i \end{aligned} \tag{3.33}$$

Entonces la ecuación (3.32) puede ser descrita por:

$$h_i{}^T \begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{P}_z \end{bmatrix} - h_i{}^T b_i \, \dot{\theta}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad i = 1,2,3 \tag{3.34}$$

Para todos los eslabones la expresión puede ser representada por:

$$\begin{bmatrix} h_1{}^T \\ h_2{}^T \\ h_3{}^T \end{bmatrix} \dot{P}_n - \begin{bmatrix} h_1{}^T b_1 & 0 & 0 \\ 0 & h_2{}^T b_2 & 0 \\ 0 & 0 & h_3{}^T b_3 \end{bmatrix} * \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.35}$$

Por teoría se sabe que la Jacobiana representa una relación entre las velocidades articulares con las velocidades angulares, entonces es posible decir que:

$$\dot{P}_n = J \, \dot{\theta}_i \qquad i = 1,2,3 \tag{3.36}$$

Despejando la expresión (3.35), da como resultado:

$$J = \begin{bmatrix} {h_1}^T \\ {h_2}^T \\ {h_3}^T \end{bmatrix}^{-1} \begin{bmatrix} {h_1}^T b_1 & 0 & 0 \\ 0 & {h_2}^T b_2 & 0 \\ 0 & 0 & {h_3}^T b_3 \end{bmatrix} \tag{3.37}$$

Al tener una cinemática compleja podemos notar que la matriz Jacobiana no depende solo del espacio articular θ_i, como es común en el caso de los robots con cinemática abierta, sino que también dependen las posiciones cartesianas del efector final, las cuales pueden ser calculadas a partir de la cinemática directa.

Reemplazando los valores de la Jacobiana se obtiene la primera matriz:

$$\begin{bmatrix} {h_1}^T \\ {h_2}^T \\ {h_3}^T \end{bmatrix}^{-1} = \begin{bmatrix} P_x - \cos(\alpha_1) * \left(R_g + L_1 \cos(\theta_1)\right) & P_y - \sin(\alpha_1) * \left(R_g + L_1 \cos(\theta_1)\right) & P_z - L_1 \sin(\theta_1) \\ P_x - \cos(\alpha_2) * \left(R_g + L_1 \cos(\theta_2)\right) & P_y - \sin(\alpha_2) * \left(R_g + L_1 \cos(\theta_2)\right) & P_z - L_1 \sin(\theta_2) \\ P_x - \cos(\alpha_3) * \left(R_g + L_1 \cos(\theta_3)\right) & P_y - \sin(\alpha_3) * \left(R_g + L_1 \cos(\theta_3)\right) & P_z - L_1 \sin(\theta_3) \end{bmatrix}^{-1}$$

Reemplazando los valores de la segunda matriz que conforma la Jacobiana

$${h_i}^T b_i = L_1 * \left(P_z \cos(\theta_i) + R_1 \sin(\theta_i) - P_x \cos(\alpha_i) \sin(\theta_i) - P_y \sin(\alpha_i) \sin(\theta_i)\right)$$

Como se especificó anteriormente, la Jacobiana de un robot de configuración paralela es compleja, por lo que tiene un costo computacional al momento de calcular las velocidades.

3.5.4.1. Jacobiana Inversa

A diferencia del modelo diferencial planteado anteriormente, la Jacobiana inversa representa la relación que se consigue al momento de multiplicar la inversa de la matriz Jacobiana con las velocidades cartesianas, de esta manera se obtiene la siguiente expresión.

$$\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} = \left(\begin{bmatrix}{h_1}^T\\ {h_2}^T\\ {h_3}^T\end{bmatrix}^{-1}\begin{bmatrix}{h_1}^T b_1 & 0 & 0\\ 0 & {h_2}^T b_2 & 0\\ 0 & 0 & {h_3}^T b_3\end{bmatrix}\right)^{-1}\begin{bmatrix}\dot{P}_x\\ \dot{P}_y\\ \dot{P}_z\end{bmatrix} \quad (3.38)$$

3.5.4.2. Configuraciones Singularidades

Luego de obtener la relación de las velocidades cartesianas y articulares, mediante la Jacobiana es importante saber si es que esta es invertible. Básicamente existen dos tipos de configuraciones singulares. Aquellas que se encuentran en las fronteras del espacio de trabajo, donde los eslabones del robot se encuentran completamente extendidos, y aquellas que se encuentran dentro del espacio de trabajo, cuando ocurre el alineamiento de uno o más ejes del robot, en otras palabras, cuando ocurre la pérdida de un grado de libertad.

La manera más fácil de hallar las singularidades de un robot es mediante la determinante de la matriz Jacobiana.

$$DET(J) = 0 \quad (3.39)$$

Se debe tener en cuenta que para hallar la determinante la matriz Jacobiana debe ser cuadrada, cosa que se cumple con la matriz Jacobiana del robot paralelo delta. Sin embargo, dicha matriz no puede ser planteada e igualada a cero con facilidad, ya que esta depende tanto como de los ángulos de las articulaciones como de las posiciones del efector final. Calcular la determinante resulta demasiado difícil ya que se tiene que analizar cada posición posible en el espacio cartesiano y luego analizar mediante si es que es invertible o no. Por lo tanto, el método de la determinante, resulta

tedioso y extenso, pero no imposible. No obstante, existe otra manera de saber las singularidades del robot paralelo delta y es a través de la cinemática inversa.

Cuando se calculó la cinemática inversa, se dio una expresión que indicaba si se podía resolver o no la ecuación. La expresión (3.27) indica cuando es o no posible llegar a una posición para cada brazo, esta expresión no limita a los demás brazos, esto quiere decir que puede que un brazo llegue a la posición y los demás no. Entonces cuando no sea posible llegar a la posición, aunque sea para un brazo se tomara como una configuración singular.

$$4{P_z}^2 + 4{R_g}^2 - {A_i}^2 + {B_i}^2\left(1 - \frac{{R_g}^2}{{L_1}^2}\right) + B_i\left(\frac{-2R_gA_i}{L_1} - 4R_g\right) \geq 0$$

Si la expresión (3.27) no se cumple, entonces existe una configuración singular. De igual manera, si se conoce la posición exacta del robot con anticipación es posible saber si existe una configuración singular con ayuda de la determinante.

3.5.5. Simulación Cinemática

Para contrastar los resultados obtenidos respecto al análisis cinemático es necesario definir las longitudes de los eslabones y demás partes del robot. De acuerdo con la lista de exigencias, se estableció que las medidas mostradas en la Figura 3.15 se adaptan mejor a las necesidades del robot.

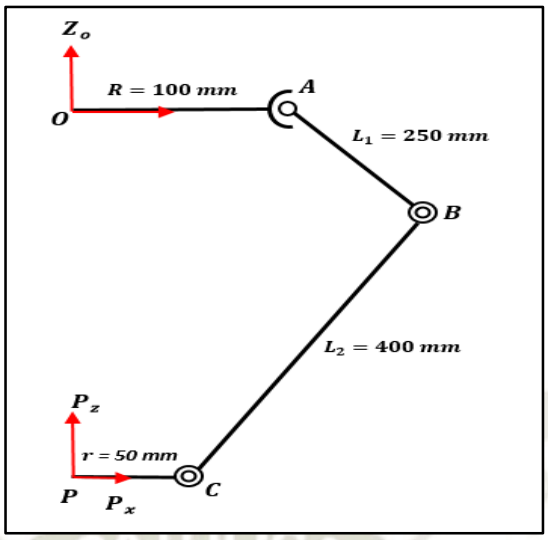


Figura 3.15. Medidas del Robot Paralelo
Fuente: *Elaboración propia.*

Una vez obtenido todos los parámetros necesarios es posible realizar una simulación con ayuda de Software matemáticos. Para ello se creó un programa en el Software de Matlab®. Se escogió este software matemático ya que cuenta con gran cantidad de herramientas para la representación gráfica del robot.

*(La programación que simula la cinemática inversa del robot se encuentra en el Anexo 01)

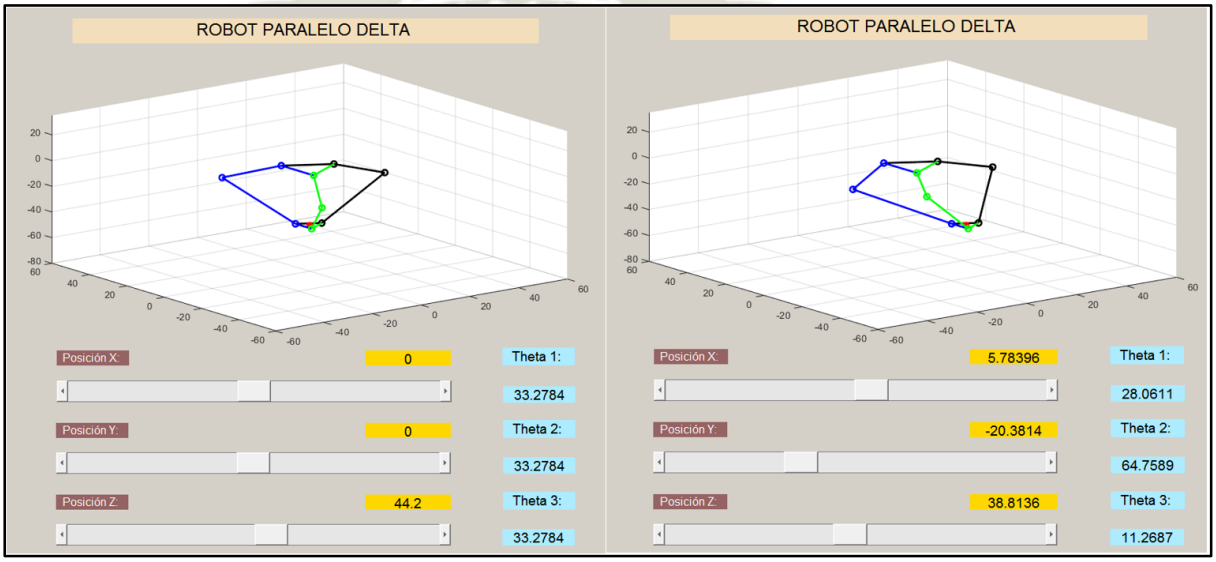


Figura 3.16. Simulación con datos aleatorios de cinemática
Fuente: *Elaboración propia.*

De la Figura 3.16 se puede observar cómo es que se resuelve el problema cinemático inverso, los brazos del robot paralelo delta se mueven solo en el eje Z, mientras tanto los antebrazos al estar unidos con articulaciones esféricas, se mueven en los 3 ejes cardinales. También se puede observar que existen barras deslizadoras que nos proporcionan el movimiento en los ejes cartesianos del robot y a la parte derecha se observa los ángulos necesarios que los actuadores deben alcanzar para llegar a la posición deseada.

Igualmente, a fin de validar los resultados obtenidos de la cinemática directa, se creó un programa capaz de resolver el problema cinemático directo, hay que tener en cuenta que este era muy extenso y sus ecuaciones eran no lineales por lo que al observar dichos resultados se optó por resolver esta ecuación con la herramienta *fsolve* del software Matlab®.

*(Programación de cinemática directa se encuentra en el Anexo 01)

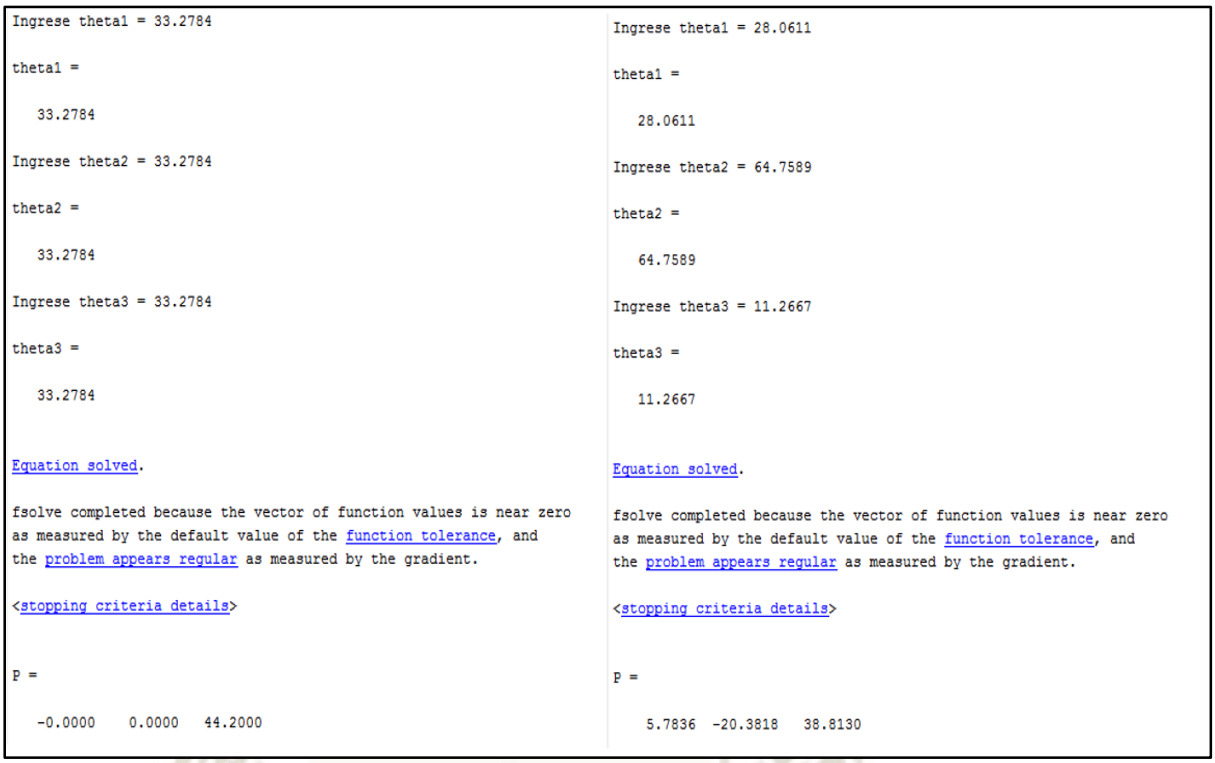


Figura 3.17. Simulación con datos cinemática
Fuente: *Elaboración propia.*

De la Figura 3.17, se obtuvo como resultado un vector llamado “P”, este vector de datos consta de las tres posiciones cartesianas siendo la posición X – Y – Z respectivamente. Los datos de los ángulos ingresados en la simulación anterior son los mismos datos obtenidos de la simulación de la cinemática inversa. Al comparar estas evidencias, se observa que los resultados coinciden con exactitud, esto quiere decir que las soluciones planteadas para el problema cinemático directo e inverso son correctas.

3.6. Análisis Dinámico

El estudio de la dinámica de un robot se basa en la relación que tiene el movimiento del robot con las fuerzas implicadas para realizar dicha acción. Es decir que a partir de las formulaciones cinemáticas se puede encontrar expresiones matemáticas que nos indiquen el comportamiento dinámico del

robot. Este análisis es de suma importancia para el dimensionamiento de los actuadores, evaluación de la estructura mecánica, evaluación y simulación del movimiento del robot.

Cabe recalcar que existen diferentes métodos que estudian y analizan el comportamiento dinámico del robot siendo los más conocidos la formulación de Newton – Euler y el método de Lagrange – Euler. Ambos métodos son bien estructurados y llegan a obtener las diversas fuerzas que se tienen en el movimiento del robot tales como inercia, Coriolis y gravedad. En ese sentido, ambos métodos tienen como conclusión la misma ecuación:

$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q}) \cdot \ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q}) \qquad (3.40)$$

Donde:

$\boldsymbol{q}$ = Vector de coordenadas articulares

$\boldsymbol{D}(\boldsymbol{q})$ = Matriz de Inercias en función de coordenadas articulares

$\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ = Matriz de Coriolis en función de las coordenadas y velocidades articulares

$\boldsymbol{C}(\boldsymbol{q})$ = Matriz de Gravedad en función de las coordenadas articulares

Al observar la expresión (3.40), se observa que depende de la primera y segunda derivada de las coordenadas articulares. Sin embargo, en el caso del robot paralelo delta obtener la Jacobiana fue un proceso complicado y obtener la aceleración articular será una variable más a calcular. Aun si se obtuviesen dichos datos, estos métodos están entrelazados con la metodología de Denavit – Hartenberg, la cual solo funciona con robots de cadena cinemática abierta. Cabe mencionar también que, si se requiere usar estos métodos, se necesitaría usar las ecuaciones de energía cinética y

energía potencial, lo cual implicaría conocer las derivadas del espacio cartesiano. Todas estas razones hacen que el proceso de obtener un modelo matemático sea más complicado y extenso. En ese sentido, se decide optar por otro método que atienda a las consideraciones necesarias y que se adapte a los modelos matemáticos ya obtenidos anteriormente.

En ocasiones, es conveniente tener el modelo dinámico expresado como una relación entre la trayectoria del extremo del robot y las fuerzas y pares que en él se aplican, referidos todos a un sistema de coordenadas cartesianas fijo del entorno de trabajo (Barrientos, Peñín, Balaguer, & Aracil, Fundamentos de Robotica, 2007).

El modelo dinámico en el espacio de la tarea o también conocido trabajo virtual es capaz de crear una relación entre estas fuerzas, esta se basa en el equilibrio dinámico que existe. En un sistema robótico, el trabajo virtual es el resultante de fuerzas virtuales que actúan por medio de desplazamientos reales o fuerzas reales. Este desplazamiento se ve reflejado en rotaciones y el término de fuerzas se ve reflejado en fuerzas o momentos. Este modelo puede ser considerado como una relación directa entre la cinemática y la dinámica.

Para obtener dicha relación entre fuerzas generalizadas, la igualdad de trabajo virtual se ve reflejado en el sistema de coordenadas cartesianas y el sistema de coordenadas articular. Partiendo de los supuestos anteriores se obtiene:

$$\boldsymbol{P = F \cdot V = \tau^T \cdot v = \ \tau^T \cdot \dot{q}} \tag{3.41}$$

Donde:

$\boldsymbol{P}$ = Potencia

$\boldsymbol{F}$ = Fuerza

$\boldsymbol{V}$ = Velocidad

$\boldsymbol{\tau^T}$ = Vector de fuerzas y pares ejercidos

$\boldsymbol{v}$ = Vector de velocidades cartesianas

$\dot{\boldsymbol{q}}$ = Vector de velocidades articulares

Sobre la base de las ideas expuestas, se escogió trabajar con la formulación del modelo dinámico en el espacio de la tarea, debido a que este método se acopla con mayor facilidad a nuestras expresiones matemáticas planteadas anteriormente.

3.6.1. Modelo Dinámico en el Espacio de la Tarea

Como se explicó anteriormente existe un equilibrio dinámico, en donde se toma en cuenta que la potencia consumida en el desplazamiento cartesiano es igual a la potencia consumida en el desplazamiento angular. Esto se ve expresado claramente en la expresión (3.41), pero aun así es necesario partir de esa expresión para hallar las demás fuerzas resultantes (Olsson, 2009). En consecuencia, de dicha expresión es posible indicar que para el robot paralelo delta se obtiene:

$$\boldsymbol{\tau_n^T \cdot \partial P_n} = \boldsymbol{\tau^T \cdot \partial \theta_i} \tag{3.42}$$

Donde:

$\boldsymbol{\tau_n^T}$ = Vector de fuerzas o torques que actúan en la base móvil

$\boldsymbol{\partial P_n}$ = Derivada de las posiciones cartesianas (velocidad cartesiana)

$\boldsymbol{\tau^T}$ = Vector de fuerzas o torques correspondiente al desplazamiento angular

$\boldsymbol{\partial \theta_i}$ = Derivada de las coordenadas articulares (velocidad articular)

Por teoría sabemos que la matriz Jacobiana representa una relación entre las velocidades cartesianas y articulares, entonces es posible decir que:

$$\partial P_n = J\, \partial \theta_i \tag{3.43}$$

Reemplazando la expresión (3.43) en la expresión (3.42):

$$\tau_n^T \cdot J\, \partial \theta_i = \tau^T \cdot \partial \theta_i$$

$$\tau^T = \tau_n^T \cdot J$$

$$\boldsymbol{\tau = J^T \cdot \tau_n} \tag{3.44}$$

En base a la expresión (3.44) es posible decir que el robot se puede reducir a 4 elementos, siendo estos los tres brazos y la base móvil. En consecuencia, el cálculo de fuerzas o torques que actúan sobre la base móvil se puede transferir a fuerzas en el espacio articular con la matriz Jacobiana. Visto que es posible hallar una expresión que relacione las torques ahora es necesario hallar las demás fuerzas o pares que nos den como resultado la expresión (3.40) en base a la expresión (3.44).

Está claro que existe una relación entre las torques o fuerzas articulares con las torques o fuerzas cartesianas. Sin embargo, es necesario hallar uno de estas variables primero para poder partir con el calcula de la otra.

El principio fundamental de la dinámica o comúnmente conocida como la segunda Ley de Newton nos indica que la fuerza que actúa sobre un cuerpo es directamente proporcional a su aceleración. Por consiguiente, es necesario hallar la aceleración que se ejerce en los desplazamientos tanto cartesiano como articular y esto se lograra con ayuda de la matriz Jacobiana.

Por teoría, es sabido que si se deriva la velocidad se obtiene la aceleración, entonces a partir de la expresión (3.35) es posible establecer una expresión que dé como resultado la aceleración tanto articular como cartesiana.

Derivando la expresión del modelo diferencial:

$$\dot{P}_n = J\,\dot{\theta}_\iota$$

$$\ddot{\boldsymbol{P}}_{\boldsymbol{n}} = \dot{\boldsymbol{J}}\,\dot{\boldsymbol{\theta}}_{\boldsymbol{\iota}} + \boldsymbol{J}\,\ddot{\boldsymbol{\theta}}_{\boldsymbol{\iota}} \tag{3.45}$$

De la expresión (3.45) se observa que se puede hallar la aceleración con ayuda de la Jacobiana, partiendo de este concepto y derivando la expresión (3.35) se obtiene que:

$$\begin{bmatrix} h_1^{\,T} \\ h_2^{\,T} \\ h_3^{\,T} \end{bmatrix} \ddot{P}_n + \begin{bmatrix} \dot{h}_1^{\,T} \\ \dot{h}_2^{\,T} \\ \dot{h}_3^{\,T} \end{bmatrix} \dot{P}_n - \begin{bmatrix} \dot{h}_1^{\,T} b_1 + h_1^{\,T} \dot{b}_1 & 0 & 0 \\ 0 & \dot{h}_2^{\,T} b_2 + h_2^{\,T} \dot{b}_2 & 0 \\ 0 & 0 & \dot{h}_3^{\,T} b_3 + h_3^{\,T} \dot{b}_3 \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{3.46}$$

$$- \begin{bmatrix} h_1^{\,T} b_1 & 0 & 0 \\ 0 & h_2^{\,T} b_2 & 0 \\ 0 & 0 & h_3^{\,T} b_3 \end{bmatrix} \cdot \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Sabiendo que la velocidad cartesiana es igual a:

$$\dot{P}_n = \left(\begin{bmatrix} h_1^{\,T} \\ h_2^{\,T} \\ h_3^{\,T} \end{bmatrix}^{-1} \begin{bmatrix} h_1^{\,T} b_1 & 0 & 0 \\ 0 & h_2^{\,T} b_2 & 0 \\ 0 & 0 & h_3^{\,T} b_3 \end{bmatrix} \right) \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

Reduciendo y reemplazando la velocidad cartesiana en la expresión (3.46) se tiene que:

$$\ddot{P}_n = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix}^{-1} \cdot \left(- \begin{bmatrix} \dot{h}_1^T \\ \dot{h}_2^T \\ \dot{h}_3^T \end{bmatrix} J + \begin{bmatrix} \dot{h}_1^T b_1 + h_1^T \dot{b}_1 & 0 & 0 \\ 0 & \dot{h}_2^T b_2 + h_2^T \dot{b}_2 & 0 \\ 0 & 0 & \dot{h}_3^T b_3 + h_3^T \dot{b}_3 \end{bmatrix} \right) \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} + J \cdot \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} \tag{3.47}$$

Como resultado de la derivación de las ecuaciones que indican el modélico diferencial, fue posible obtener la expresión (3.47), en la cual se aprecia la relación que existe entre la aceleración cartesiana con la aceleración angular.

Una vez obtenidas las aceleraciones, es importante asumir y especificar todos los parámetros que serán incluidos en el presente modelo. Ya que se sabe que para la creación del robot paralelo delta se usaran servomotores, hay que tomar en cuenta el funcionamiento de estos y como es que influyen en el modelo dinámico del robot.

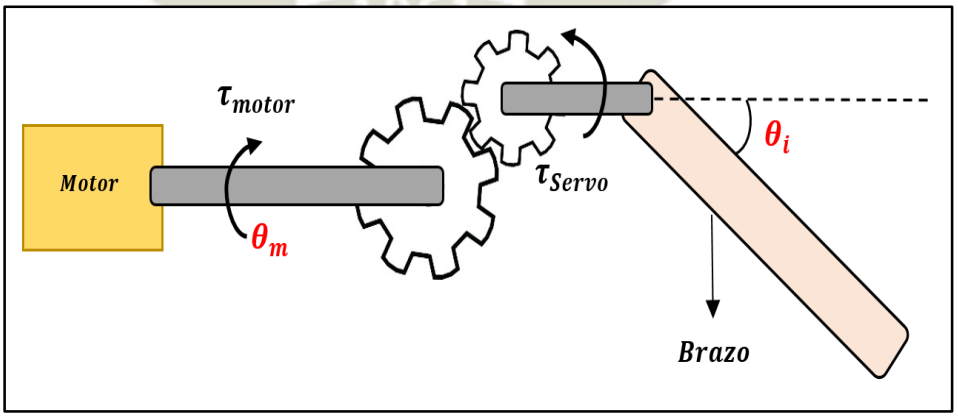


Figura 3.18. Brazo accionado con servomotor
Fuente: *Elaboración propia.*

De la Figura 3.18 se puede expresar una relación proporcional en la cual tomando en cuenta la transmisión de engranajes es posible decir que:

$$\boldsymbol{\theta_m = n_c \cdot \theta_i} \tag{3.48}$$

Donde:

$\boldsymbol{\theta_m}$ = Angulo de giro del motor

$\boldsymbol{\theta_i}$ = Angulo de giro de la articulación

$\boldsymbol{n_c}$ = Constante de reducción de engranajes

De acuerdo con la expresión (3.48) se asume que la transmisión es rígida y que la relación entre la velocidad de entrada y velocidad de salida son directamente proporcionales, a partir de esto es posible decir que:

$$\boldsymbol{\tau_{motor} = \frac{\tau_{servo}}{n_c}} \tag{3.49}$$

Donde:

$\boldsymbol{\tau_{motor}}$ = Torque producido por el motor

$\boldsymbol{\tau_{servo}}$ = Torque de carga necesario para mover el brazo y su extensión

$\boldsymbol{n_c}$ = Constante de reducción de engranajes

Partiendo de las expresiones anteriores, es posible continuar con la búsqueda de las demás expresiones que den como resultado fuerzas o torques. No obstante, para el robot paralelo delta la complejidad del modelo dinámico parte del movimiento del antebrazo, eso se ve reflejado en la expresión (3.47) donde se aprecia que el antebrazo no forma parte de las ecuaciones debido a las asunciones hechas al principio del epígrafe. Este problema se puede reducir si se obvia la inercia rotacional producida por el antebrazo, de esta manera las fuerzas generadas entre

el brazo y la base móvil están en una sola dirección dado por la orientación del brazo.

Partiendo de estas asunciones es posible decir que:

- Los momentos de inercial rotación son obviados solo para los antebrazos.
- Para considerar la masa del antebrazo se asumió que está separada en las extremidades del antebrazo, esto resulta a que una porción de la masa esta al final del brazo junto con la articulación y la otra porción de masa se encuentra iniciando la base móvil junto con la articulación.
- Partiendo del supuesto anterior y asumiendo que el antebrazo es una barra es posible decir que momento de inercia es el siguiente:

$$I = \frac{1}{3} \cdot m \cdot {L_2}^2 \tag{3.50}$$

Donde:

$\boldsymbol{I}$ = Momento de Inercia

$\boldsymbol{m}$ = Masa del eslabón

$\boldsymbol{L_2}$ = Longitud del antebrazo

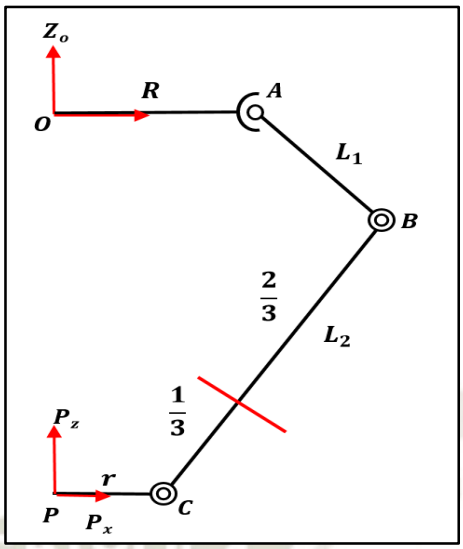


Figura 3.19. Proporcionalidad entre masas e Inercia
Fuente: *Elaboración propia.*

La Figura 3.19 muestra que se puede escoger un tercio de la masa de antebrazo en la articulación C y dos tercios del antebrazo en la articulación B.

A continuación, es necesario definir los parámetros de cada elemento del robot. En la base móvil se debe tener cuenta las diferentes cargas, sabiendo esto se obtiene:

$$m_{Total\ BM} = m_{base\ movil} + m_{carga} + m_{art} + 3 \cdot (1 - r) \cdot m_{L_2} \quad (3.51)$$

Donde:

$\boldsymbol{m_{Total\ BM}}$ = Masa total de todas las cargas que actúan en la base móvil

$\boldsymbol{m_{base\ movil}}$ = Masa de la base móvil

$\boldsymbol{m_{carga}}$ = Masa de la carga que el robot puede operar

$\boldsymbol{m_{art}}$ = Masa de la articulación implicada

$\boldsymbol{m_{L_2}}$ = Masa del antebrazo

$\boldsymbol{r}$ = Constante de relación de la masa entre eslabones

La expresión (3.51) implica la sumatoria de todas las masas que actúan sobre la base móvil, la variable “r” expresa la relación de la masa entre eslabones, como se explicó anteriormente, existe una proporción de masa en las que se reparte un tercio a la extremidad inferior y dos tercios a la extremidad superior en este caso “r” tomara el valor de 2/3.

Con relación al primer eslabón es decir el brazo del robot paralelo delta sus expresiones de masa y centro de masa son las siguientes:

$$cmb = L_1 \cdot \left(\frac{\frac{1}{2} \cdot m_{L_1} + m_{art} + r \cdot m_{L_2}}{m_{L_1} + m_{art} + m_{L_2}} \right) \tag{3.52}$$

Donde:

$\boldsymbol{cmb}$ = Centro de masa del brazo

$\boldsymbol{m_{L_1}}$ = Masa del brazo 1

La ubicación del centro de masa del brazo se halló con ayuda de la ecuación del centro de masa, debido a las asunciones hechas anteriormente la expresión (3.52) toma en cuenta la masa del antebrazo multiplicado con la proporción correspondiente y también la articulación que conecta estos eslabones.

En cuanto a la masa resultado del brazo se obtiene:

$$m_{Brazo} = m_{L_1} + m_{art} + r \cdot m_{L_2} \tag{3.53}$$

Respecto a la inercia generada en la articulación, es posible decir que en esta actúan las inercias implicadas en el actuador y en el brazo, esto se puede expresar en:

$$I_{bi} = J_{motor}.{n_c}^2 + I_{bt} \qquad i = 1,2,3 \tag{3.54}$$

Donde:

$\boldsymbol{I_{bi}}$ = Inercia resultante en la articulación

$\boldsymbol{J_{motor}}$ = Inercia del motor con reducción

$\boldsymbol{I_{bt}}$ = Inercia del brazo

En la expresión (3.54) se toma en cuenta la inercia generado por el motor y es llevada al espacio articular multiplicándola por el cuadrado de la constante de reducción. Asimismo, la inercia del brazo conlleva la inercia del antebrazo creado en el extremo inferior de este mismo, donde se tiene que:

$$I_{bt} = \frac{m_{L_1}}{3} \cdot {L_1}^2 + {L_1}^2 \cdot \left(m_{art} + r \cdot m_{L_2}\right) \tag{3.55}$$

De acuerdo con la expresión (3.55), se observa que interviene tanto las masas de las articulaciones y parte de la masa del antebrazo, cabe resalta que es multiplicada por toda la longitud del brazo ya que se consideró que esta proporción de la masa estaría en el extremo inferior del brazo.

En relación con las demás fuerzas, es momento de tomar en cuenta las fuerzas ocasionadas por la gravedad y por el movimiento mismo. Siendo este el caso, se considerará las fuerzas originadas por la base móvil y los brazos.

Para el caso de la base móvil, se definirán dos fuerzas siendo estas:

- **Fuerza Gravitacional de la Base Móvil**

$$G_{BM} = m_{Total\,BM} \cdot [0 \quad 0 \quad -g]^T \tag{3.56}$$

Donde:

$\boldsymbol{g}$ = Aceleración de la gravedad

- **Fuerzas Axiales**

$$F_n = m_{Total\,BM} \cdot \ddot{P}_n \tag{3.57}$$

De la expresión (3.56) y (3.57) reemplazando en la expresión (3.44) se obtiene que:

$$\boldsymbol{\tau = J^T \cdot F_n}$$

$$\tau_a = J^T \cdot m_{Total\,BM} \cdot \ddot{P}_n \tag{3.58}$$

$$\boldsymbol{\tau = J^T \cdot G_{BM}}$$

$$\tau_{GBM} = J^T \cdot m_{Total\,BM} \cdot [0 \quad 0 \quad -g]^T \tag{3.59 }$$

Donde:

$\boldsymbol{\tau_a}$ = Vector de fuerzas axiales generadas en la base móvil

$\boldsymbol{\tau_{GBM}}$ = Vector de fuerzas gravitacionales generadas en la base móvil

Por otra parte, al igual que en la base móvil existen dos tipos de fuerzas o torques que actual sobre cada brazo, las cuales son la fuerzas producida por la gravedad y la fuerza producida por la inercia del brazo.

- **Fuerza Gravitacional del Brazo**

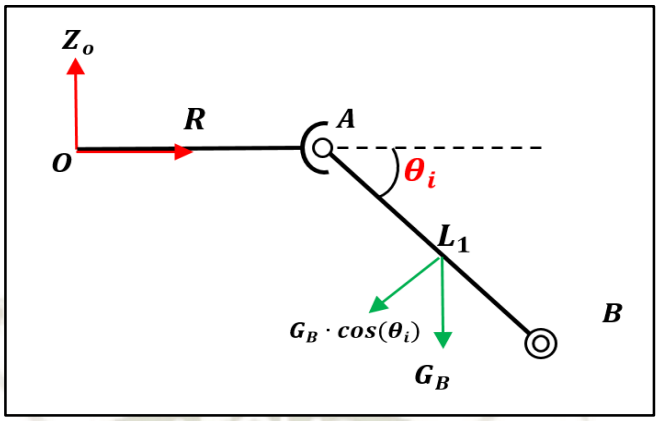


Figura 3.20. Fuerza Gravitacional actuado en el brazo
Fuente: *Elaboración propia.*

De la Figura 3.20 se observa que la fuerza gravitacional actúa de manera perpendicular a cada brazo, entonces se obtiene:

$$G_B = -m_{Brazo} \cdot g \tag{3.60}$$

$$\tau_{GB} = cmb \cdot G_B \cdot [\cos(\theta_1) \quad \cos(\theta_2) \quad \cos(\theta_3)] \tag{3.61}$$

Donde:

$\boldsymbol{G_B}$ = Fuerza Gravitatoria de cada brazo

$\boldsymbol{\tau_{GB}}$ = Vector de fuerzas gravitacional que actual en el centro de masa

- **Torques generados por la inercia de cada brazo**

Por física es sabido que existe una relación entre la inercia y el toque puesto que si se multiplica la velocidad angular con la inercia el resultado es torque, sabiendo esto es posible decir que:

$$\tau_{inerB} = I_{bi} \cdot \ddot{\theta}_i \tag{3.62}$$

De la expresión (3.62) se puede expresar I_{bi} en forma matricial, siendo esta:

$$I_{bi} = \begin{bmatrix} I_{b1} & 0 & 0 \\ 0 & I_{b2} & 0 \\ 0 & 0 & I_{b3} \end{bmatrix} \tag{3.63}$$

Reemplazando la expresión (3.63) en (3.62) se obtiene:

$$\tau_{InerB} = \begin{bmatrix} I_{b1} & 0 & 0 \\ 0 & I_{b2} & 0 \\ 0 & 0 & I_{b3} \end{bmatrix} \cdot \ddot{\theta}_i \tag{3.64}$$

Donde:

$\boldsymbol{\tau_{InerB}}$ = Vector de torques generadas por la inercia de cada brazo

$\boldsymbol{I_{bi}}$ = Inercia resultante en la articulación

De acuerdo con el principio de d'Alembert, quien propuso la teoría del equilibro dinámico, establece que la suma de fuerzas externas que actúan sobre un cuerpo y las fuerzas de inercia forma un sistema de equilibrio (d'Alembert, 2018)

Vinculado al concepto expuesto las fuerzas de inercia deben ser iguales a las fuerzas no inerciales. En efecto es posible decir que:

$$\boldsymbol{\tau_i + \tau_{GB} + \tau_{GBM} = \tau_{InerB} + \tau_a} \tag{3.65}$$

Donde:

$\boldsymbol{\tau_i}$ = Vector de torque correspondiente al actuador

$\boldsymbol{\tau_{GB}}$ = Vector de fuerzas gravitacional que actual en el centro de masa

$\boldsymbol{\tau_{GBM}}$ = Vector de fuerzas gravitacionales generadas en la base móvil

$\boldsymbol{\tau_{InerB}}$ = Vector de torques generadas por la inercia de cada brazo

$\boldsymbol{\tau_a}$ = Vector de fuerzas axiales generadas en la base móvil

Como resultado de la expresión (3.65) es posible hallar el torque necesario para ejecutar cada movimiento siendo este:

$$\boldsymbol{\tau_i} = \boldsymbol{\tau_{InerB}} + \boldsymbol{\tau_a} - \boldsymbol{\tau_{GB}} - \boldsymbol{\tau_{GBM}} \qquad i = 1,2,3 \tag{3.66}$$

Reemplazando las expresiones (3.58) y (3.64) en la expresión (3.66) de tal manera que, se puedan apreciar la matriz de inercia, coriolis y gravedad se obtiene que:

$$\tau_i = I_{bi} \cdot \ddot{\theta}_i + J^T \cdot m_{Total\,BM} \cdot \ddot{P}_n - \tau_{GB} - \tau_{GBM} \tag{3.67}$$

Reemplazando la expresión (3.45) en (3.67):

$$\tau_i = I_{bi} \cdot \ddot{\theta}_i + J^T \cdot m_{Total\,BM} \cdot \left(\dot{J}\,\dot{\theta}_\iota + J\,\ddot{\theta}_\iota\right) - \tau_{GB} - \tau_{GBM}$$

$$\tau_i = I_{bi} \cdot \ddot{\theta}_i + J^T \cdot m_{Total\,BM} \cdot \dot{J}\,\dot{\theta}_\iota + J^T \cdot m_{Total\,BM} \cdot J\,\ddot{\theta}_\iota - \tau_{GB} - \tau_{GBM}$$

$$\tau_i = (I_{bi} + m_{Total\,BM} \cdot J^T \cdot J) \cdot \ddot{\theta}_i + \left(m_{Total\,BM} \cdot J^T \cdot \dot{J}\right) \cdot \dot{\theta}_\iota - (\tau_{GB} + \tau_{GBM}) \tag{3.68}$$

En consecuencia, de la expresión (68), se obtuvo como resultado el torque necesario para que el robot paralelo delta pueda ejercer los movimientos y esta adopta la forma de la expresión planteada al principio del epígrafe:

$$\boldsymbol{\tau} = \boldsymbol{D}(\boldsymbol{q}) \cdot \ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{C}(\boldsymbol{q})$$

Siendo:

$$\boldsymbol{D}(\boldsymbol{q}) = (I_{bi} + m_{Total\,BM} \cdot J^T \cdot J)$$

$$\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \left(m_{Total\,BM} \cdot J^T \cdot \dot{J}\right)$$

$$\boldsymbol{C(q)} = \tau_{GB} + \tau_{GBM}$$

En conclusión, la expresión (3.68) indicará la evolución del torque respecto al tiempo necesario para cada actuador, esta expresión también será usada para el dimensionamiento necesario de los actuadores y material a usar del robot.

3.7. Selección de Material

Respecto al material a usar, según la tabla de exigencias se escogió entre dos tipos de plásticos siendo estos el plástico tipo PLA y el plástico tipo ABS. La razón por la cual se escogió estos es que ambos son usados en la impresión 3D. Y en el caso del robot paralelo delta y debido a la forma planteada que este tomara, es necesario el uso de una impresión 3D para partes como la base fija, base móvil y brazos.

En cuanto a que tipo de plástico a escoger, ambos son sumamente ligeros, pero en cuanto a resistencia y durabilidad el ABS es mucho mejor. No obstante, al ser un derivado del petróleo, este tipo de plástico produce gases nocivos al momento de su tratamiento. Al respecto del PLA, este material tiene una temperatura de trabajo menor y su uso en impresiones 3D es más común que el ABS. Sin embargo, es más difícil maquinar este material una vez que la pieza ya esté hecha.

Figura 3.21. Impresión 3D PLA (izquierda) - Impresión 3D ABS (derecha)
Fuente: *(Delta3D, 2017)*

A continuación, se muestra una tabla indicando las diferencias entre el plástico tipo PLA y el plástico tipo ABS.

Tabla 3.4. Diferencias entre plástico ABS y plástico PLA

	ABS	**PLA**
Resistencia a la Tracción	27 MPa	37 MPa
Módulo de Elasticidad	2.2 - 7.6 GPa	4 GPa
Densidad	1.0 - 1.4 g/cm^3	1.3 g/cm^3
Temperatura de Deformación	60°C	105°C
Precio	$20	$22

Fuente: *Elaboración propia.*

Según la Tabla 3.4 y tomando en cuenta las propiedades mecánicas se usará el plástico tipo ABS ya que es plástico comúnmente usada en la industria automotriz y su resistencia es mejor, también es fácil de mecanizar sobre este plástico y su precio es menor respecto al PLA. También, dependiendo del de tipo de plástico ABS se puede obtener piezas más ligeras.

3.8. Diseño Mecánico

El concepto de diseño mecánico hace referencia al diseño de objetos y sistemas de naturaleza mecánica, tales como estructuras, mecanismos, piezas, etc. Este se basa en analizar los límites de resistencia de las piezas a través de su límite de fluencia, los esfuerzos aplicado, etc.

En esta sección del capítulo, se realizará un estudio a las piezas o partes que serán usadas en el robot paralelo delta, estas partes serán los brazos, antebrazos, articulaciones base fija, base móvil y demás piezas necesarias. Este estudio nos indicara si las piezas cumplen con las condiciones necesarias para que el robot paralelo delta, pueda llevar acabo sus funciones. Así mismo todo este análisis puede ser contrastado con ayuda de Software tales como Inventor®, Solid Works®, AutoCAD®, etc.

Para empezar, es importante definir todas las partes que constituirán al robot paralelo delta. Si bien es cierto que se planteó un boceto indicando las partes principales como su brazo, ante-brazo, base fija y base móvil, no se tuvo en cuenta los detalles necesarios que serán usados para llevar acabo la fabricación del robot. Es por eso que en esta sección se mostrara y explicara todas las piezas que hacen posibles el movimiento del robot.

3.8.1. Partes Mecánicas del Robot Paralelo Delta

Para lograr un correcto análisis mecánico es necesario analizar todos los componentes de esta manera también podremos definir sus masas y sus dimensiones, estas serán necesarias para evaluar los análisis anteriores.

*(Para ver los planos de la estructura robótica referirse al Anexo 03)

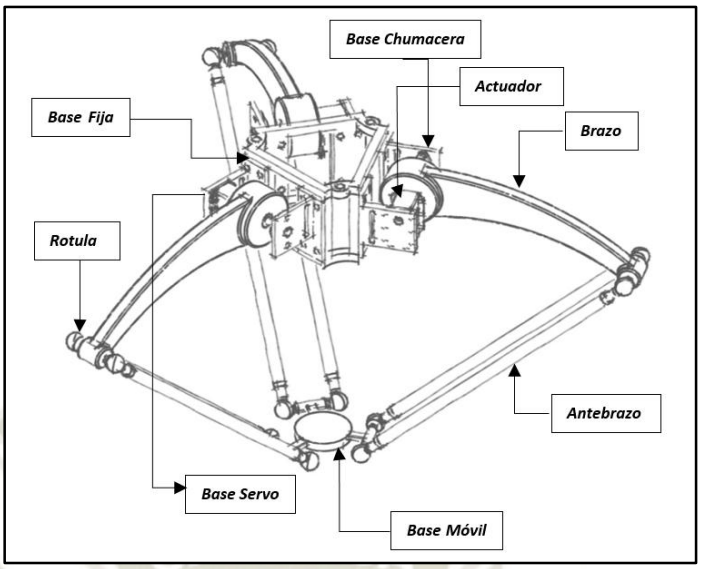


Figura 3.22. Componentes del robot paralelo delta
Fuente: *Elaboración propia.*

De la Figura 3.22 se observa que el robot paralelo delta posee más componentes que los estudiados anteriormente. A continuación, se analizará cada uno de ellos:

- **Base Fija**

  Este componente es el inicio del robot paralelo delta, ya que brinca la forma delta necesaria, separando cada brazo 120° sobre su eje. Este componente es estático y deberá aguantar el peso de todas las demás piezas del robot. También deberá estar sujeto a una superficie, para cumplir todos estos requisitos se propuso lo siguiente:

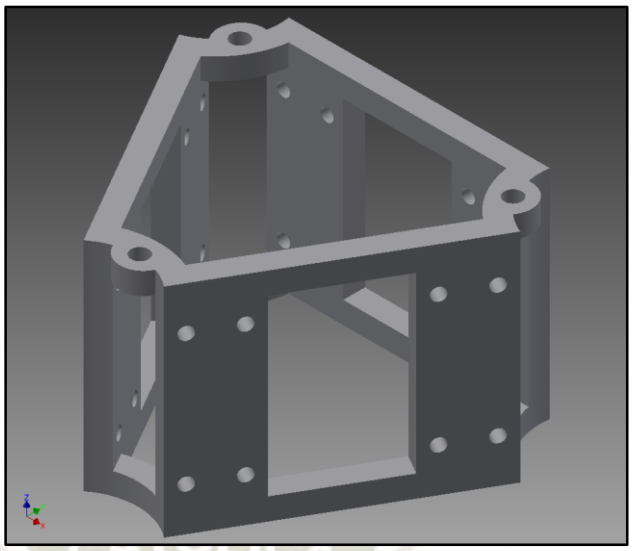


Figura 3.23. Base Fija
Fuente: *Elaboración propia.*

En la Figura 3.23 se observa que la base fija cuenta de una forma delta, esto refleja los 120° de separación necesarios de los brazos también posee 3 agujeros en la parte superior, estos servirán como sujeción al momento de montar el robot en alguna base. Los agujeros restantes servirán como un medio para soportar la base donde irán montados los actuadores y demás. La distancia del centro de toda la base a los extremos donde estarán montados los brazos es de 100mm.

- **Base Móvil**

  Esta pieza del robot es la encargada de brindar las posiciones cartesianas, al igual que la pieza anterior, este debe de tener una separación de 120 °, y debe ser lo suficientemente fuerte para soportar la carga que levantara. Debe tener una distancia desde el centro del eje a la articulación de 50mm.

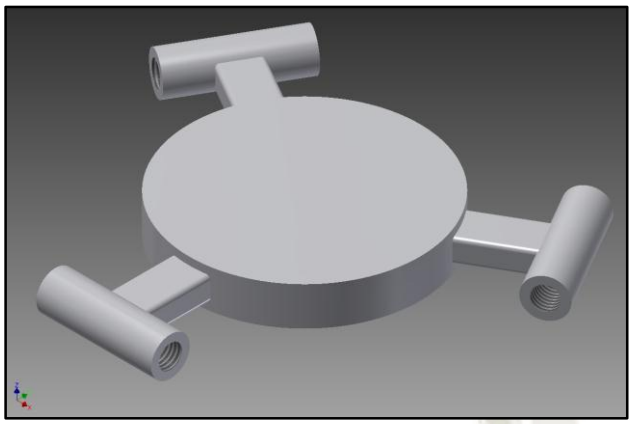


Figura 3.24. Base Móvil
Fuente: *Elaboración propia.*

Como se aprecia en la Figura 3.24, la base móvil cuenta con piezas que sobresalen de la circunferencia, esas piezas servirán como apoyo para conectar los eslabones con la base móvil. En esas piezas también se colocarán las articulaciones.

- **Brazo**

  Este componente es el primer eslabón del robot y es el más importante, en total se usarán 3 brazos. Este debe cumplir con la longitud de 250mm propuesta anteriormente. Debe ser liviana y soportar los esfuerzos cambiantes generados por el movimiento del robot. Así mismo, el brazo debe permitir un grado de rotación en la dirección del eje axial de la transmisión de potencia.

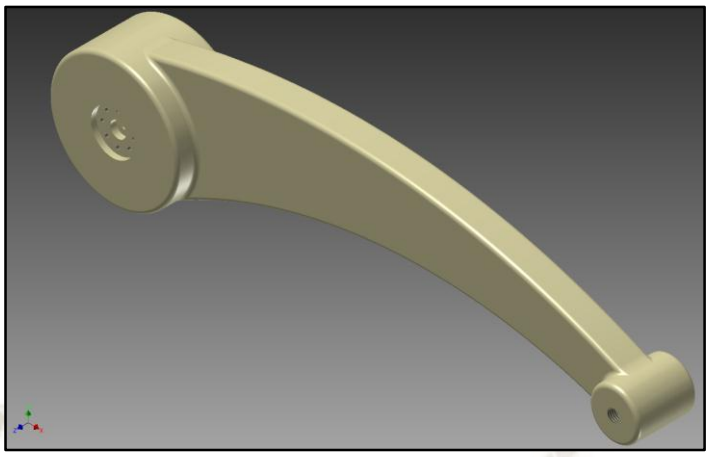


Figura 3.25. Brazo con sección transversal variable
Fuente: *Elaboración propia.*

De la Figura 3.25 se aprecia que el brazo cuenta con una sección transversal variable. Este diseño está enfocado, gracias a la recopilación de diversas fuentes, a soportar los esfuerzos cambiantes a lo largo de la geometría, la parte alta y ancha ayuda a soportar la flexión, mientras que la parte angosta ayuda a soportar mejor la torsión.

- **Antebrazo**

  Este elemento del robot, respecto a su forma, es el más simple ya que su función es transmitir el movimiento generado por el brazo mas no soportar cargas o torsiones significativas.

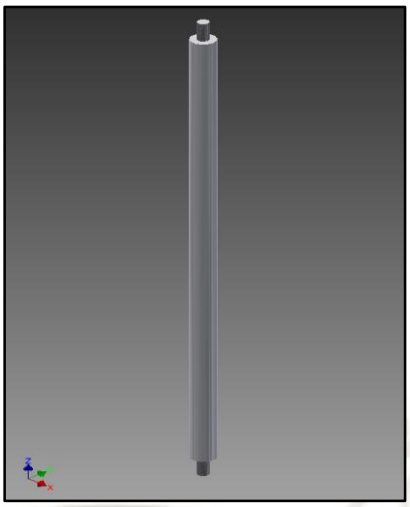


Figura 3.26. Antebrazo
Fuente: *Elaboración propia.*

Como se explicó anteriormente, la forma del antebrazo es simple y cuenta en las extremidades con hilos, estos servirán para acoplar las articulaciones. El robot constara de 6 antebrazos, dispuesto dos en cada brazo, de manera paralela y eso se hace con el objetivo de tener una estructura robótica más estable.

- **Articulación Esférica**

  Es un componente mecánico orientable que presenta dimensiones unificadas, permitiendo la transmisión de fuerzas estáticas y dinámicas. Este componente posee tres grados de libertad y su función con el robot es de suma importancia, ya que es el elemento que transforma las rotaciones ejercidas por los brazos en movimientos cartesianos.

  Existen diferentes tipos de articulaciones esféricas tales como:

  - Rótulas axiales
  - Rótulas con contacto angular
  - Rótulas con rodamiento integrado
  - Rótula con vástago

- Rótula de articulación angular
- Rótula de articulación recta

De todas las articulaciones expuestas, la que más se adapta al robot paralelo delta es la rótula de articulación angular, ya que posee una forma perpendicular que servirá para transferir las rotaciones del brazo a los antebrazos. Siendo el modelo que más se adapta a las necesidades del diseño se usara una rotula de articulación angular de la marca ISB modelo SQ 6CRS.

**(Para ver información referente a la rótula SQ 6CRS ir al Anexo 05)*

Figura 3.27. Rótula de articulación angular SQ 6CRS
Fuente: *(123Bearing, 2017)*

Ya que el robot posee 3 brazos y en cada brazo se montará 2 antebrazos hace necesario el uso de 4 articulaciones por brazo; esto significa que la carga que moverán los actuadores será mayor. Al mismo tiempo, el uso de las rotulas restringe parte del volumen del trabajo del robot, ya que estas articulaciones tienen un espacio de trabajo limitado dependiendo de la aplicación.

- **Base del Servomotor**

  Este componente servirá como apoyo del Servomotor, debe ser capaz de soportar el peso del actuador y también soportar la torsión generada por los movimientos del robot. También debe cumplir con las medidas planteadas anteriormente. Al igual que las demás piezas. Esta pieza al soportar un elemento eléctrico debe tener el espacio necesario para el cableado correspondiente y al igual que las demás piezas y debido a la simetría del robot, serán necesarios 3 bases de servomotor.

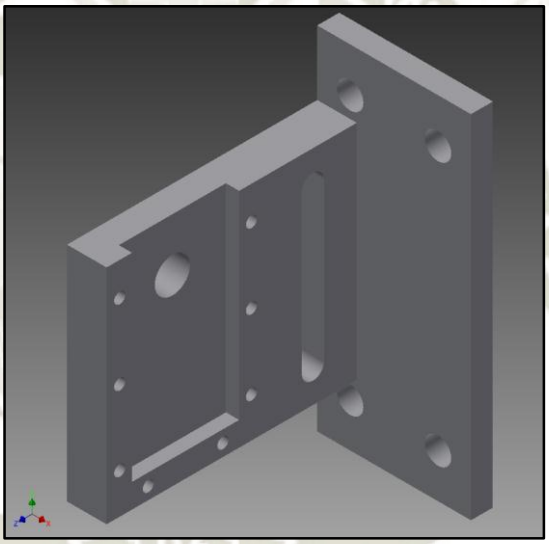


Figura 3.28. Base Servomotor
Fuente: *Elaboración propia.*

- **Eje y Rodamiento**

  Con el propósito de lograr que el servomotor brinde un torque libre al brazo, es necesario que el peso del brazo, antebrazo y base móvil, recaiga sobre otro elemento. Es por esto que se decidió usar un eje, para que sirva como mecanismo de apoyo de las cargas generados por los elementos ya antes mencionados. Este eje debe ser liviano, y capaz de soportar las cargas generadas, Teniendo en cuestas

consideraciones se optó por escoger un eje con diámetro de 8 mm, de acero inoxidable. Dicho elemento se encontrará apoyado en el interior del brazo del robot como fuera de este, para lograr el apoyo en el brazo se decidió usar un rodamiento lineal LM8UU. Tanto el eje y el rodamiento seleccionado son componentes muy usados en impresoras 3D, por lo que estos objetos se acoplan muy bien al propósito que se persigue.

(Para ver información sobre el rodamiento seleccionado *ir al Anexo 06)*

Figura 3.29. a) Eje con diámetro de 8mm b) Rodamiento lineal LM8UU
Fuente: *(Naylamp Mechatronics , 2017)*

- **Chumacera**

Una chumacera es un rodamiento montado que tiene como función dar apoyo a un eje de rotación. Como ya se mencionó antes, el peso de los brazos, antebrazos y base móvil recaerá sobre un eje, este eje debe de tener un mecanismo de apoyo, siendo este la chumacera. Este elemento también le proveerá al eje una libre moción alejado de fricciones.

Se ha decidido usar una chumacera de pared de modelo M8-KFL08, esta chumacera es muy usada en impresoras 3D, donde se tiene

revoluciones y cargas bajas. Por lo tanto, este modelo es conveniente para el propósito que persigue el robot paralelo. *(Para ver información sobre la Chumacera ir al Anexo 06)*

Figura 3.30. Chumacera de pared M8-KFL08
Fuente: *(Naylamp Mechatronics , 2017)*

- **Base de la Chumacera**

  Una vez definido el uso de un eje con rodamiento y una chumacera, es necesario crear un elemento capaz de sostener a la chumacera. Es aquí donde nace la idea dela base para la chumacera. Este elemento también debe de cumplir con los requisitos de carga, torsión y debe estar correctamente alineado para que de esta manera el eje coincida con el brazo y el servomotor.

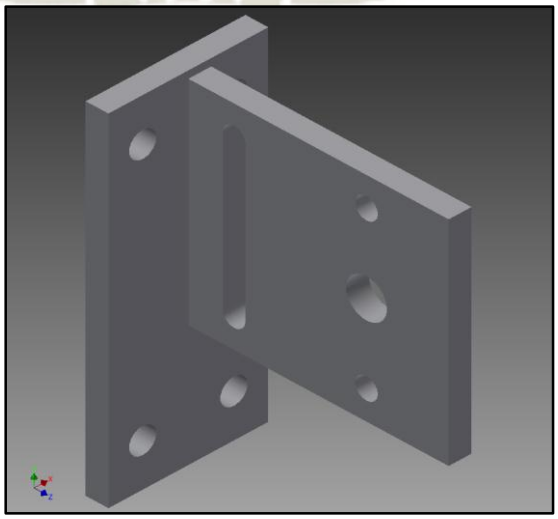


Figura 3.31. Base de la Chumacera
Fuente: *Elaboración propia.*

- **Jaula**

  A diferencia de los robots seriales, el robot párelo delta necesita que su base fija sea sostenida verticalmente. Atendiendo a estas necesidades, es necesario la creación de una estructura en la cual el robot pueda sostenerse y moverse sin colisionar, siendo necesario que se adapte a su forma delta se planteó lo siguiente.

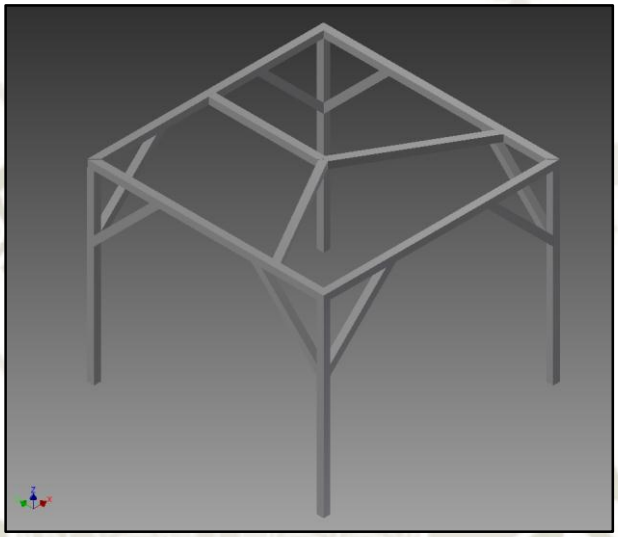


Figura 3.32. Jaula
Fuente: *Elaboración propia.*

3.8.2. Análisis de Esfuerzos

El objetivo principal de este análisis se centra en estudiar el comportamiento de los esfuerzos que actúan sobre los componentes y la deformación plástica obtenidos de aplicar diversas cargas en cada componente.

Para este estudió y simulación de esfuerzos se usará un Software capaz de generar y validad la información sobre el análisis de cada componente, siendo este el Software de diseño asistido por computadora Autodesk Inventor®. Este software cuenta con la opción de

elementos finitos y genera resultados con valores muy cercanos a los reales de esfuerzos, deformaciones y factores de seguridad.

A simple vista podemos afirmar que las cargas aplicadas a los componentes del robot no son significativamente grandes y al usar un material como el ABS, los resultados en cuanto en deformación serán minúsculos si es comparado con un robot industrial. En tal caso, este análisis y simulación de esfuerzos busca reforzar y corroborar lo ya sabido, así como obtener información complementaria de todos los componentes.

Para la simulación solo serán tomados en cuenta los componentes más propensos a fallar siendo estos la base fija, base móvil, base del servomotor, base de la chumacera y los brazos. Para la simulación se tomaron en cuenta ciertas condiciones:

- Todos los componentes a analizar cuentan con un cuerpo sólido y están hechos de plásticos ABS.
- Las restricciones de movimiento se dan en las áreas donde cada componente es apoyado.
- Las cargas que se tomaran en cuenta en la simulación, son las ejercidas por el peso de los componentes, la gravedad y la carga que moverá el robot. Las fuerzas originadas por la masa de los componentes se hallan en la Tabla 3.5.
- Se plantea una simulación en el supuesto caso más crítico, siendo esta cuando el brazo del robot paralelo se encuentra extendido a lo largo de su eje horizontal, ya que en esta posición se genera el mayor momento.

Tabla 3.5. Peso de los elementos del robot y fuerza que ejercen

Elemento	Cantidad	Masa (g)	Fuerza (N)
Brazo	3	260	2.55
Antebrazo	6	60	0.59
Base Servo	3	36	0.35
Base Chumacera	3	27	0.26
Base Fija	1	264	2.59
Base Móvil	1	40	0.39
Actuador*	3	100	0.98
Chumacera*	3	75	0.74
Eje*	3	25	0.25
Rodamiento*	3	10	0.10
Rotula*	12	40	0.39
Carga	1	300	2.94

Fuente: *Elaboración propia.* * *Aproximación*

En cuanto a los resultados ofrecidos de la simulación por elementos finitos se obtienen 3, siendo estos:

- **Tensión de Von Mises:** Indica el máximo esfuerzo aplicado al componente.
- **Desplazamiento:** Indica la cantidad de movimiento que ejerce la carga.
- **Factor de Seguridad:** Es el cociente entre el valor calculado de la capacidad máxima de carga y el valor del requerimiento esperado.

a) Resultado del análisis de esfuerzos – Brazo

En el análisis realizado a brazo se tomó en cuenta la carga que ejerce las rotulas, antebrazos, base móvil y carga. Luego de la simulación se obtienen los siguientes resultados.

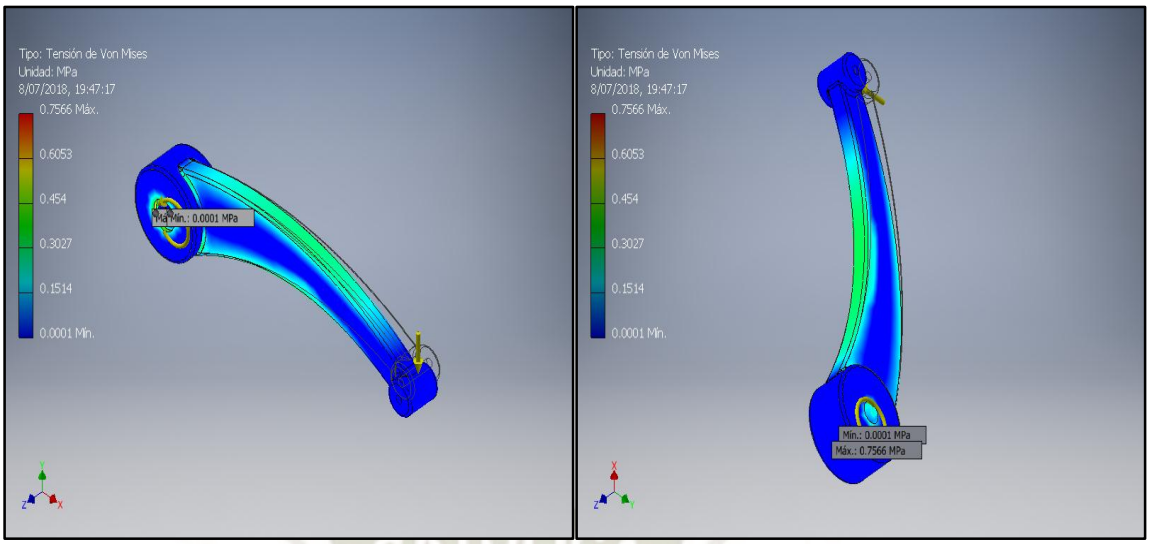


Figura 3.33. Esfuerzos de Von Mises hallados en el brazo
Fuente: *Elaboración propia.*

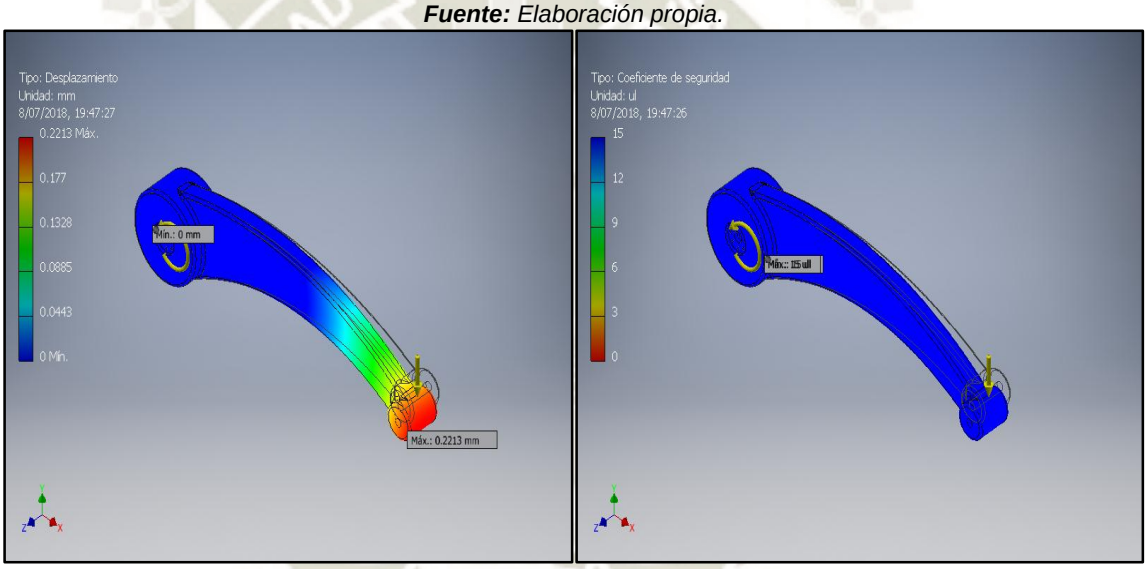


Figura 3.34. a) Desplazamiento del brazo b) Factor de Seguridad del brazo
Fuente: *Elaboración propia.*

Resumiendo, los datos obtenidos obtenemos:

- Máximo esfuerzo normal: 0.76 MPa
- Máximo desplazamiento: 0.2213 mm
- Máximo factor de seguridad: 15

Al comparar estas evidencias, se obtiene que los esfuerzos y desplazamientos son mínimos y por lo tanto despreciables, como

consecuencia el factor de seguridad es elevado, garantizando que el elemento no fallara.

b) Resultado del análisis de esfuerzos – Base Móvil

En el caso de la base móvil, la carga más importante que afecta a dicho elementos es el de la carga que el robot paralelo delta debe levantar. Basándonos en los supuestos anteriores se obtiene los siguientes resultados.

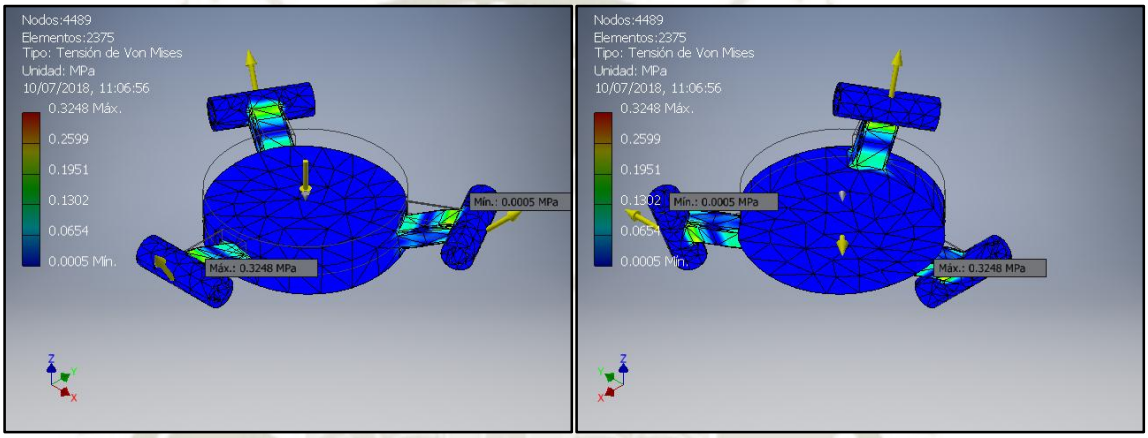


Figura 3.35. Esfuerzos de Von Mises hallados en la base móvil
Fuente: *Elaboración propia.*

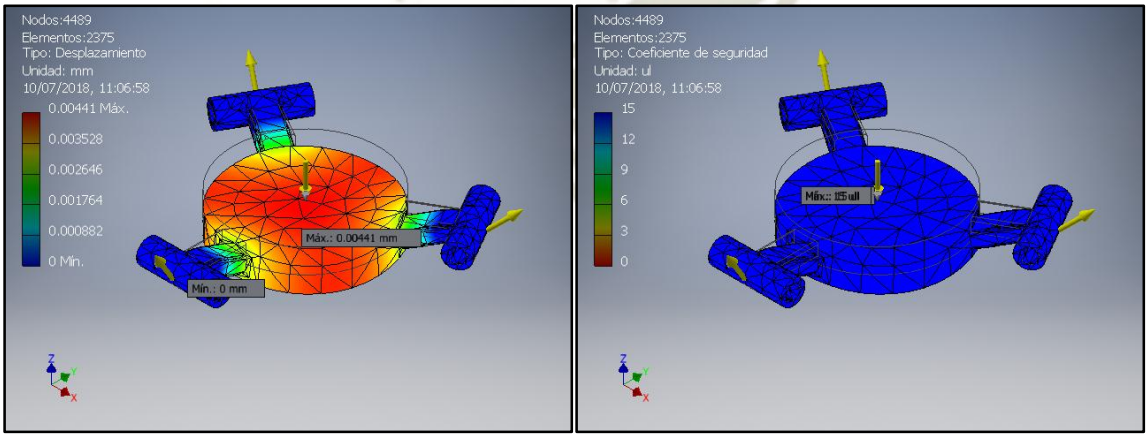


Figura 3.36. a) Desplazamiento de la base móvil b) Factor de Seguridad de la base móvil
Fuente: *Elaboración propia.*

Resumiendo, los datos obtenidos obtenemos:

- Máximo esfuerzo normal: 0.32348 MPa

- Máximo desplazamiento: 0.00441 mm
- Máximo factor de seguridad: 15

Al igual que el análisis realizado al brazo, se obtiene esfuerzos y desplazamientos mínimos que pueden ser considerados despreciables.

c) Resultado del análisis de esfuerzos – Base Fija

La base fija, es el componente del robot paralelo delta que más carga soportara, ya que sostendrás todos los componentes restantes, sin esta base no sería posible ningún movimiento del robot. También este elemento va empotrado en la jaula lo que significa que tendrá restricciones de movimiento en su zona superior, partiendo de estas consideraciones se obtiene lo siguiente.

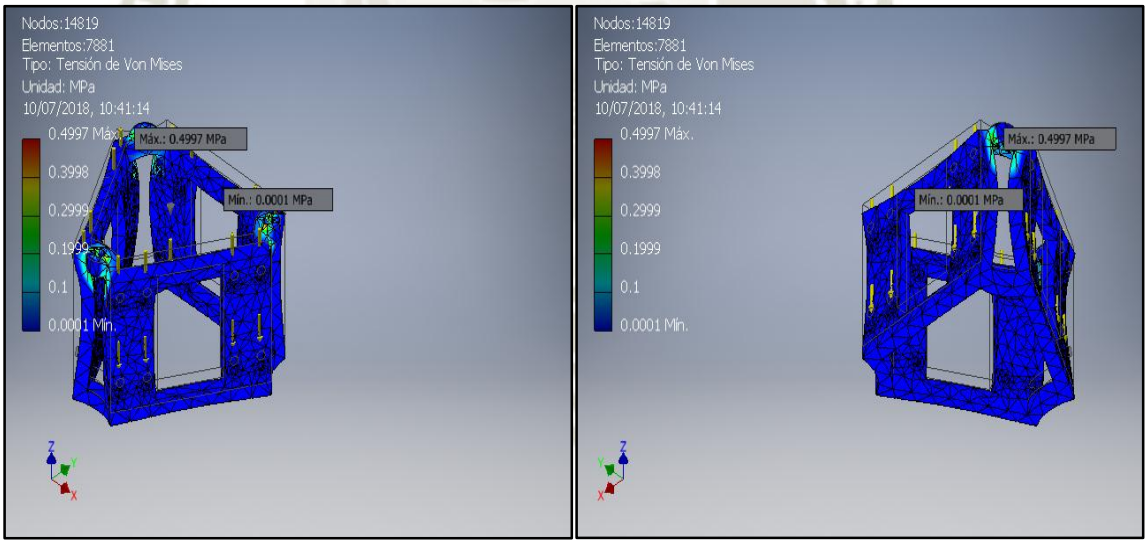


Figura 3.37. Esfuerzos de Von Mises hallados en la base fija
Fuente: *Elaboración propia.*

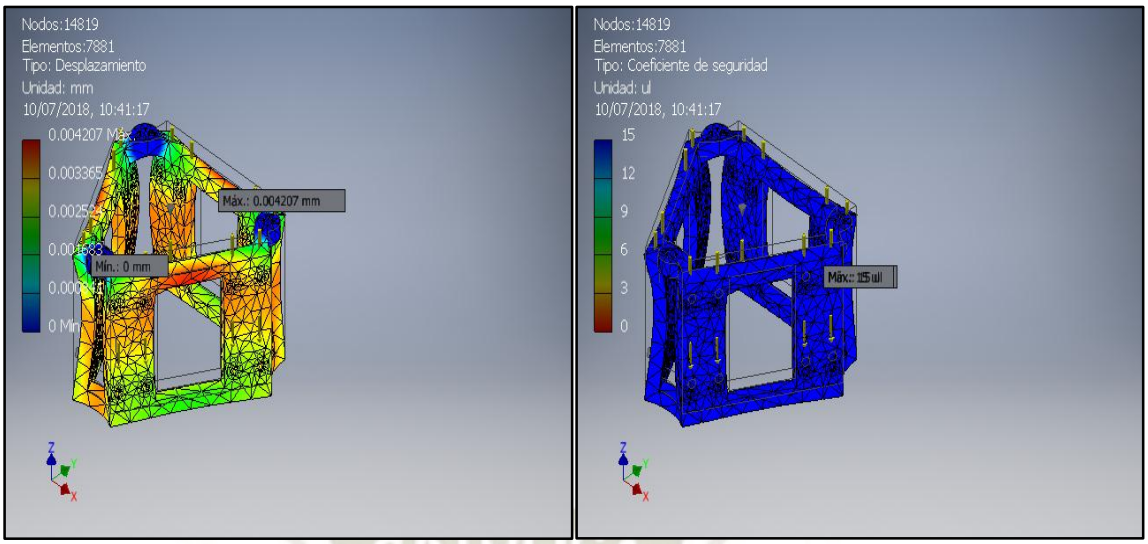


Figura 3.38. a) Desplazamiento de la base fija b) Factor de Seguridad de la base fija
Fuente: *Elaboración propia.*

Resumiendo, los datos obtenidos obtenemos:

- Máximo esfuerzo normal: 0.5 MPa
- Máximo desplazamiento: 0.004207 mm
- Máximo factor de seguridad: 15

Estos resultados indican que la base fija será capaz de soportar el peso de todo el robot con facilidad puesto a que su esfuerzo y desplazamiento son despreciables. También nos garantiza que este componente es capaz de aguantar 15 veces todas las cargas ejercidas.

d) Resultado del análisis de esfuerzos – Base del Servomotor

La base del servomotor es un componente relativamente pequeño y su función principal es la de brindar apoyo para sostener el servomotor. No obstante, esta pieza debe ser capaz de soportar el peso del actuador y la torsión que se genera. Dicho esto, se obtuvo los siguientes resultados.

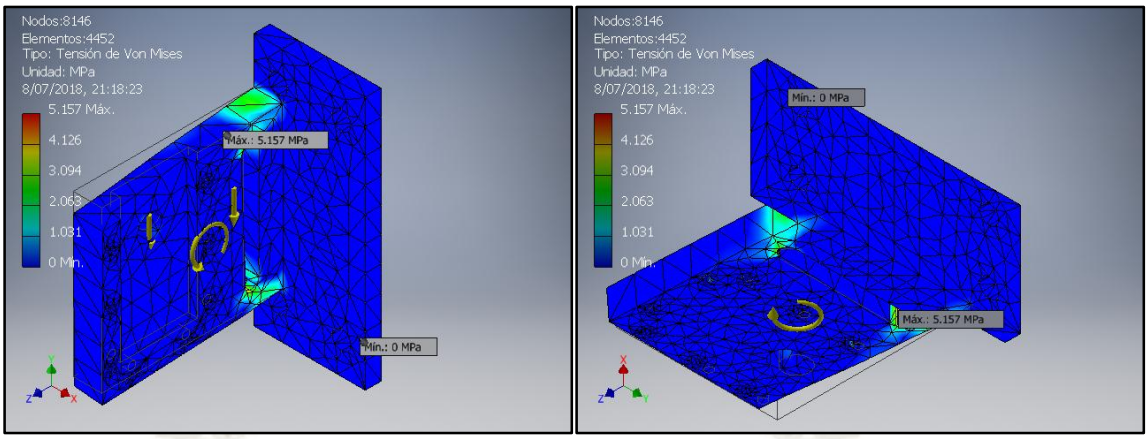


Figura 3.39. Esfuerzos de Von Mises hallados en la base del servomotor
Fuente: *Elaboración propia.*

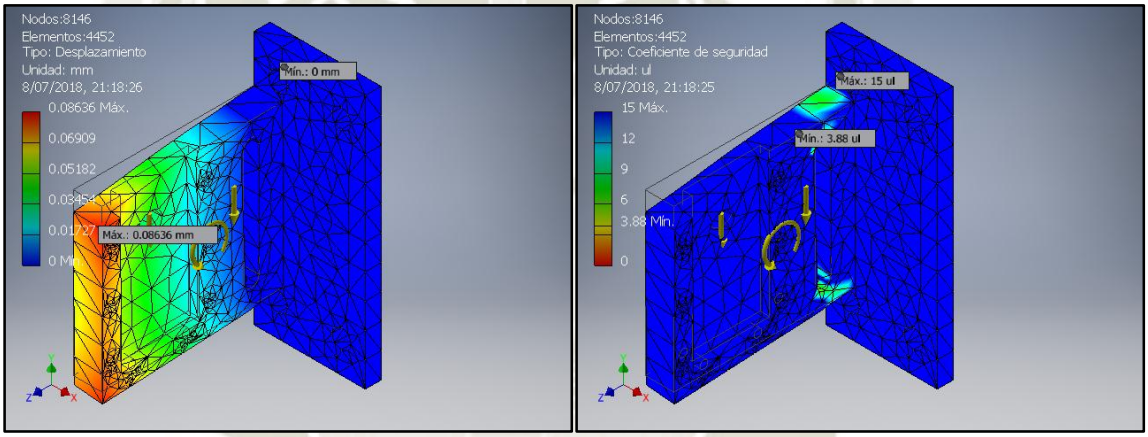


Figura 3.40. a) Desplazamiento de la base del servomotor b) Factor de Seguridad de la base del servomotor
Fuente: *Elaboración propia.*

Resumiendo, los datos obtenidos obtenemos:

- Máximo esfuerzo normal: 5.157 MPa
- Máximo desplazamiento: 0.08636 mm
- Máximo factor de seguridad: 15

A diferencia de los demás componentes del robot, la base del servomotor sufre un esfuerzo máximo mayor al resto, y esto cobra sentido sabiendo que es un elemento pequeño. También de la

Figura 3.40 se observa que tiene un factor de seguridad mínimo de 3.88, esto se debe a la torsión generada por el movimiento del robot.

e) Resultado del análisis de esfuerzos – Base de la chumacera

Por lo que se refiere a la base de la chumacera, esta pieza tiene como función principal servir como apoyo para la chumacera. Al estar enlazado con la chumacera, también tiene que resistir la carga generada por el peso del brazo y de los demás componentes del robot.

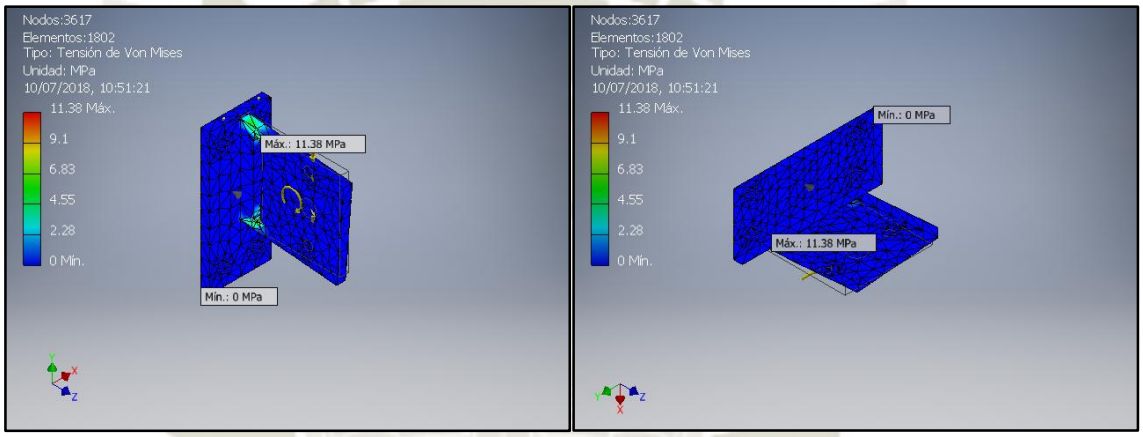


Figura 3.41. Esfuerzos de Von Mises hallados en la base de la chumacera
Fuente: *Elaboración propia.*

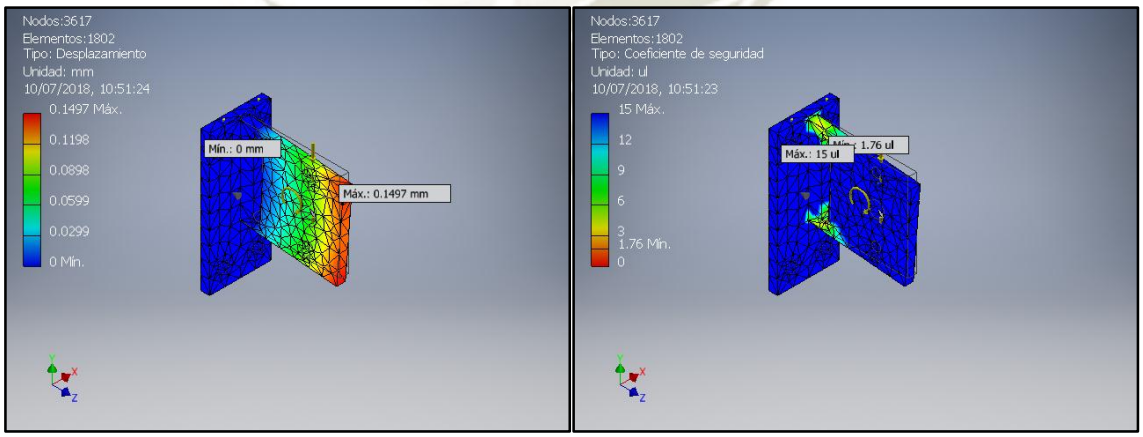


Figura 3.42. a) Desplazamiento de la base de la chumacera b) Factor de Seguridad de la base de la chumacera
Fuente: *Elaboración propia.*

Resumiendo, los datos obtenidos obtenemos:

- Máximo esfuerzo normal: 11.38 MPa
- Máximo desplazamiento: 0.1497 mm
- Máximo factor de seguridad: 15

Al ser un elemento pequeño y estar sometida a cargas mayores, es notable que existirá un incremento en los esfuerzos, también se puede observar en Figura 3.42 que el área más propensa a sufrir un fallo es la que une la zona de apoyo de la chumacera con la zona de apoyo de la base fija, esta área tiene un factor de seguridad mínimo de 1.76, por lo que es la zona del robot más propensa a fallar. Aun así, la base de la chumacera sigue cumpliendo su objetivo sin fallar.

3.8.3. Selección de Actuadores

Luego de haber visto gran parte del diseño del robot y ya haber definido longitudes, material y masas es posible realizar un análisis más completo para la selección de los actuadores. Es sabido que se estableció con anterioridad el uso de servomotores como actuadores, pero escoger un servomotor en especial requiere de varios factores tales como torque, velocidad, precisión, tamaño, volumen, peso y funcionalidad; siendo el torque el factor de más interés puesto a que este parámetro determina la fuerza de giro que se requiere para efectuar los movimientos del robot.

En el mercado actual los servomotores que predominan sobre los demás son los de corriente directa (DC) y esto se debe a que estos dispositivos de accionamiento rinden un mejor desempeño frente a otros dispositivos con convertidores de frecuencia, ya que estos no tienen un control

óptimo respecto a control de posición y resultan poco efectivos a bajas velocidades.

Antes de entrar en consideración, es necesario especificar las características necesarias que mejor se adapten al robot paralelo delta. Primero es necesario que exista una realimentación continua de las posiciones alcanzadas por el servomotor, este también debe tener un tamaño mínimo que se adapte a los componentes del robot, debe contar con una buena resolución para un control más óptimo. Respecto al sistema de control, este debe tener una velocidad de transmisión de datos óptima, capaz de enviar todos los datos necesarios al controlador sin ningún problema.

Los servomotores deberán contar con una velocidad de operación lo suficientemente grande como para ejercer los movimientos del robot en el tiempo estimado y lo más importante deben de tener un torque lo suficientemente grande, capaz de ejercer todos los movimientos del robot sin dificultad ni problema alguno.

Partiendo de las consideraciones anteriores y luego de una extensa revisión por diferentes catalogas de fabricantes de servomotores, se decidió optar por los servomotores Dynamixel de la marca coreana Robotis®.

Figura 3.43. Servomotores Dynamixel
Fuente: *(Pish Robot, 2018)*

Se optó por usar los Servomotores Dynamixel, porque:

- Poseen diferentes modos de control, siendo capaz de moveré libremente como una rueda si es necesario o modo articular limitando los movimientos de servo hasta donde sea necesario.
- Posee funciones de alarma que son activadas cuando el servomotor detecta valores máximos de corriente, temperatura, sobrecarga, etc.
- Son dispositivos que brindan información en tiempo real de la posición angular, velocidad, temperatura y voltaje.
- Gozan de una alta resolución lo que genera gran precisión a la hora de ejercer los movimientos del robot.
- Posee dos protocolos de comunicación siendo estos TTL y RS-485.
- Es posible conectar varios servomotores a partir de una misa red, de esta manera se tiene un solo bus de datos para comunicarse con todos los servomotores, reduciendo drásticamente el cableado.
- Posee gran torque y consta de una alta eficiencia.

Si bien es cierto que se escogió usar los servomotores Dynamixel, aún no se especificó qué modelo se usara puesto que no se sabe el torque

necesario para ejercer los movimientos del robot. Dicho esto, es necesario calcular los torques y esto se obtiene a partir del modelo dinámico.

Con el fin de obtener una correcta selección del actuador, se tiene que asegurar una velocidad y aceleración en cada movimiento del robot. Para tal efecto, se usará funciones sinusoidales como posiciones a alcanzar del servomotor, de esta manera se asegura las derivadas de la posición.

Como seguimiento de esta actividad se dispone una función de entrada:

$$q_i(t) = 15 + \left(\frac{4}{5} \cdot t \cdot sin\left(\frac{t}{5}\right)\right) \tag{3.69}$$

Siendo su gráfica:

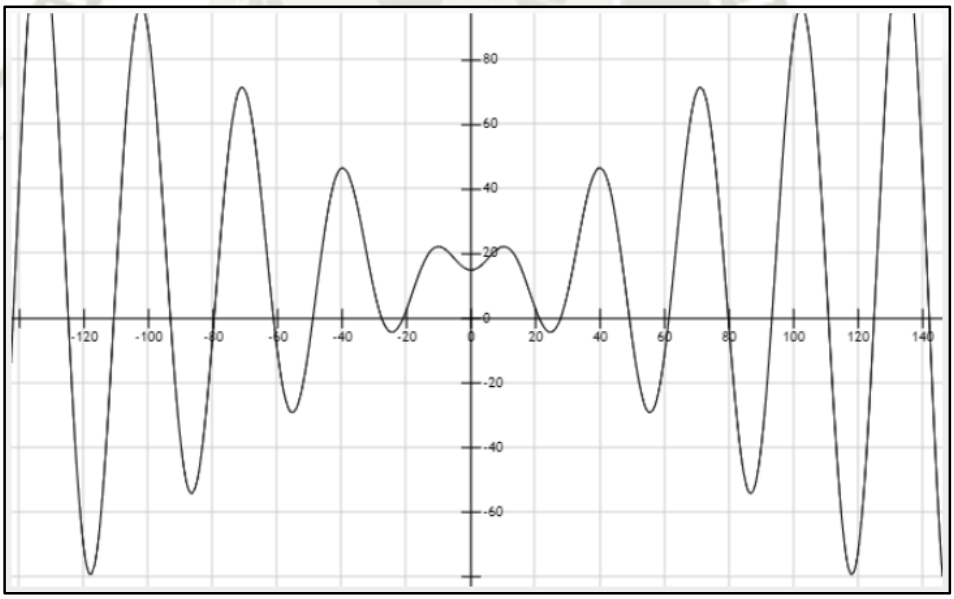


Figura 3.44. Función Sinusoidal de entrada
Fuente: *Elaboración propia.*

Para la realización de la selección de los actuadores se tomaron en cuenta las siguientes condiciones:

- La expresión (3.69) será la función de entrada para todos los actuadores, de esta manera se tiene un movimiento continuo en el eje cartesiano Z.
- Se tendrá en cuenta todas las masas que intervienen en el movimiento del robot, incluyendo una carga máxima de 300 gramos en el efector final.
- Con el fin de realizar un correcto dimensionamiento se creará un programa en el software de MatLab®, capaz de brindar el torque, los parámetros articulares y cartesianos; con la finalidad de entender a detalle la evolución del torque.

*(La programación en MATLAB se encuentre en el Anexo 02)

Atendiendo a estas consideraciones, se obtuvo los siguientes resultados:

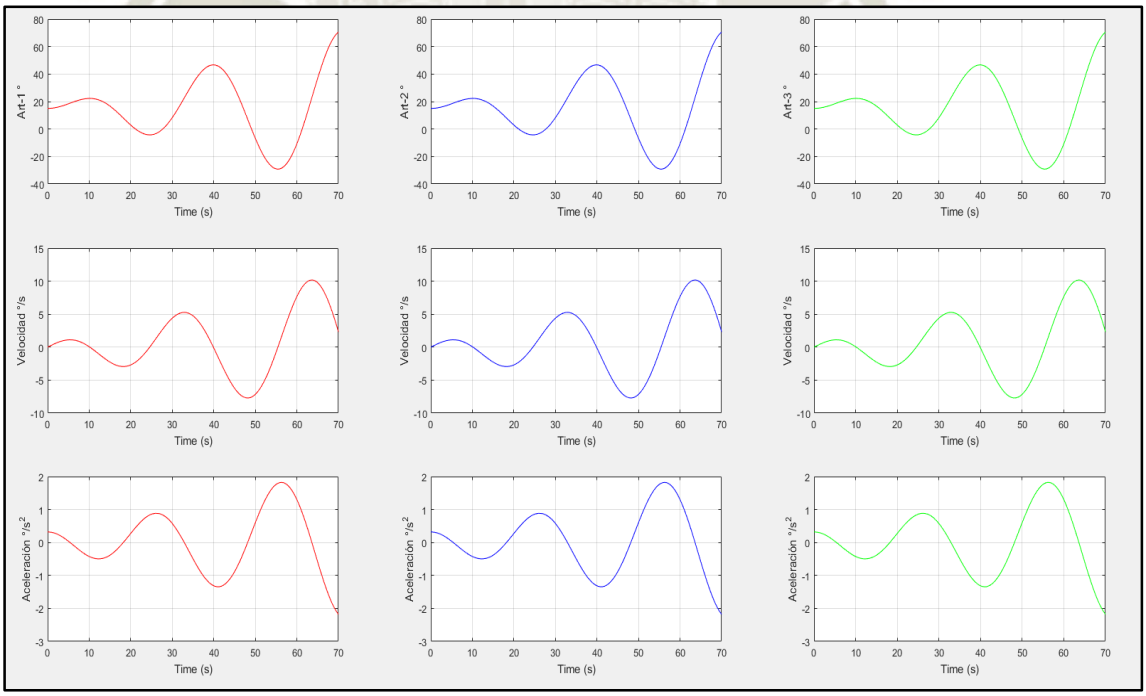


Figura 3.45. Posiciones, velocidades y aceleraciones angulares necesarias para cada actuador
Fuente: *Elaboración propia.*

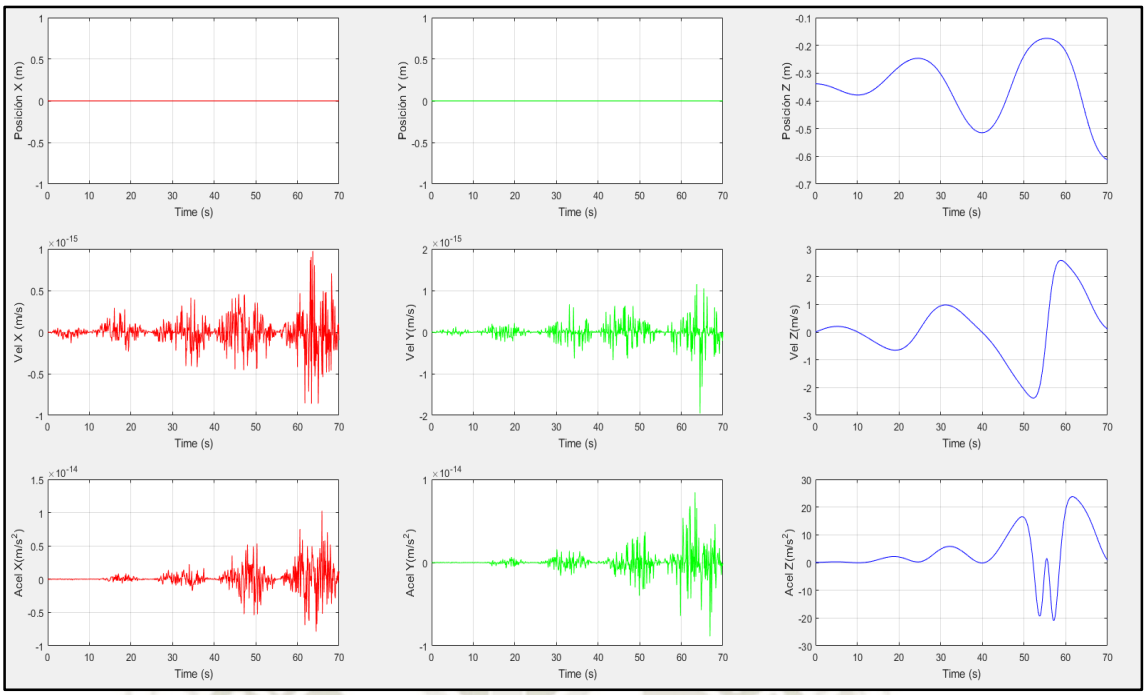


Figura 3.46. Posiciones, velocidades y aceleraciones cartesianas alcanzadas por el efector final
Fuente: *Elaboración propia.*

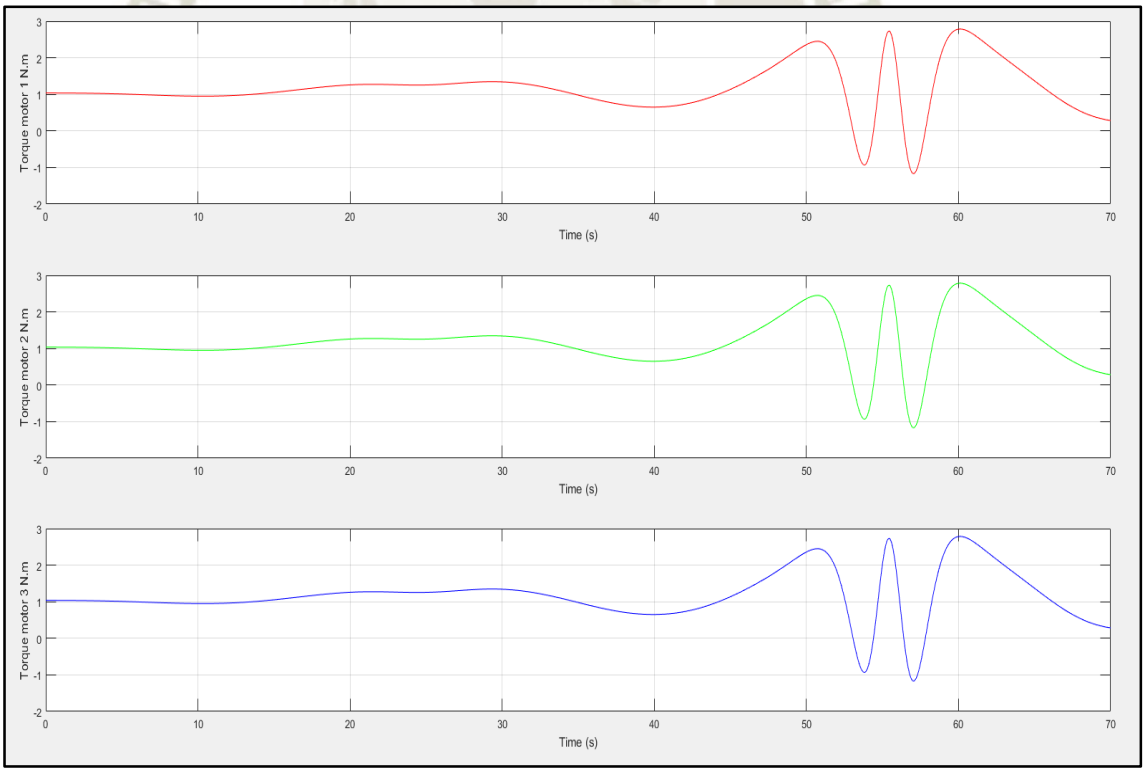


Figura 3.47. Torque calculado para cada articulación
Fuente: *Elaboración propia.*

De la Figura 3.47, se observa que desde el comienzo el robot paralelo delta necesita un torque mayor a 1N.m, a medida que la función de entrada impuesto avanza, su amplitud también. Sin embargo, el torque se mantiene relativamente constante pese a los cambios de amplitud, logrando así un torque máximo de 1.36 N.m en 40 segundos transcurridos. Por otra parte, desde los 45 segundos existe un cambio brusco y notable de torque y esto se debe a un incremento en la velocidad y la aceleración, obteniendo así un torque máximo de 2.734 N.m a los 55 segundos y un torque mínimo de -1.161 N.m a los 57 segundos, tal como muestra la Figura 3.48.

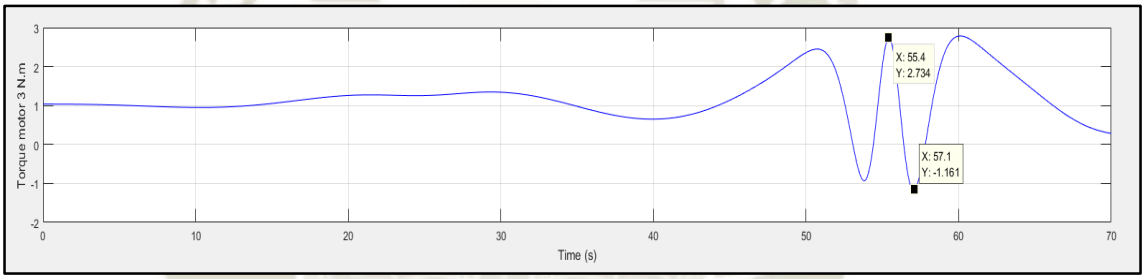


Figura 3.48. Torque máximo y mínimo de la articulación
Fuente: *Elaboración propia.*

De la Figura 3.46, se observa que a los 55 segundos se logra una posición en el eje Z de -0.175 metros, respecto a la base fija.

Comparando este resultado con los torques obtenidos es posible decir que ese punto se encuentra cerca de una singularidad, también que mientras más cerca se encuentre la base fija a la base móvil se requerirá de más torque. Esto se ve reflejado en la Figura 3.49, indicando la posición alcanzada a -0.175 metros en el eje Z.

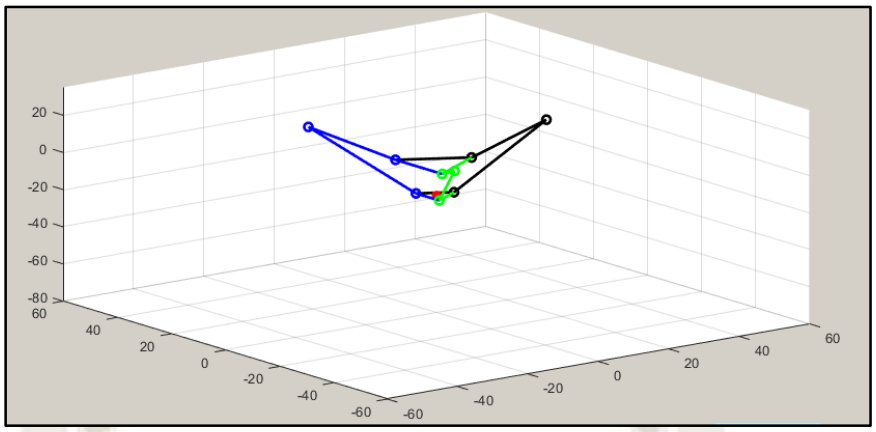


Figura 3.49. Posición alcanzada por los actuadores a Z = - 0.175 m
Fuente: *Elaboración propia.*

Al comparar estas evidencias, se optó por trabajar con el torque máximo obtenido aun cuando se tiene un torque promedio menor, es aconsejable trabajar siempre con los datos más críticos.

Como es sabido, el torque máximo se produjo a los 55 segundos obteniendo un valor de 2.734 N.m. Con la finalidad de evitar posibles problemas, se incluirá un factor de seguridad de 1.2, dando como resultado el torque requerido para el dimensionamiento del actuador.

$$\tau_{actuador} = \tau_{max} \cdot Factor\ de\ Seguridad \tag{3.70}$$

$$\tau_{actuador} = 2.734\ N \cdot m * 1.2$$

$$\boldsymbol{\tau_{actuador} = 3.2808\ N \cdot m}$$

En base a la expresión (3.70), fue posible hallar un valor de torque que se ajustan a las necesidades del robot paralelo delta. Una vez obtenido este valor, es posible proceder con la selección del servomotor correcto y esto se logrará recurriendo a los catálogos del fabricante. Cabe recalcar, que, por la simetría del robot y su configuración de cadena cinemática cerrada, el servomotor a escoger será el mismo para cada articulación.

Partiendo de los resultados anteriores, se optó por usar el servomotor Dynamixel XM430-W350-T, porque:

- Posee un torque de funcionamiento que varía entre 3.8 N.m a 4.8 N.m dependiendo del voltaje suministrado.
- Cuenta con un disipador de calor de carcasa de aluminio.
- Posee una resolución de 0.088˚ x 4096 bits.
- Puede realimentar datos de trayectoria, estado de movimiento (posición, error, seguimiento, etc.)
- Consta de una eficiencia energética (corriente standby de 40 mA).

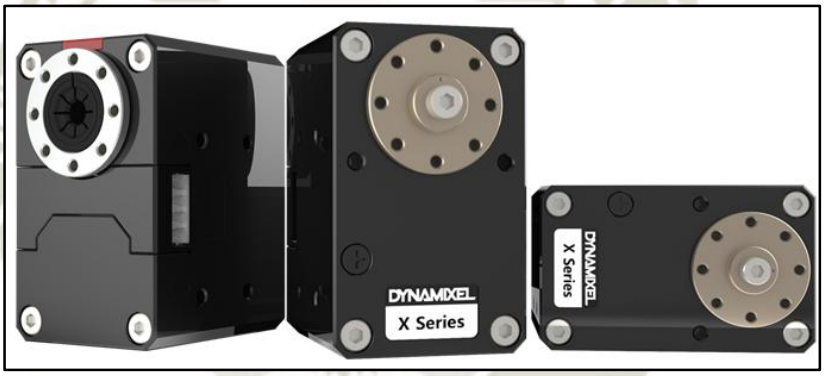


Figura 3.50. Servomotor Dynamixel XM430-W350-T
Fuente *(Robotis, 2017)*

3.8.4. Espacio de trabajo

El espacio de trabajo, define los puntos en el espacio que el robot es capaz de alcanzar, a diferencia de los robots seriales, el espacia de trabajo de un robot paralelo es limitado y hallarlo resulta en algunos casos complicado. No obstante, anteriormente a partir de la expresión resultante de la cinemática inversa nace una expresión que indica los límites del robot paralelo delta, dicha expresión puede ser usada para hallar todos los puntos que el robot es capaz de alcanzar.

Otra manera de identificar este espacio de trabajo nace del concepto de la cinemática directa, en la que se planteaba que los antebrazos del robot se podían mover libremente, logrando así 3 esferas. Partiendo del supuesto anterior, la intersección de esas 3 esferas indica el espacio de trabajo del robot paralelo delta.

Finalmente, y ya sabido el rango de giro del actuador, es posible hallar el espacio de trabajo del robot, esto se muestra en las siguientes figuras.

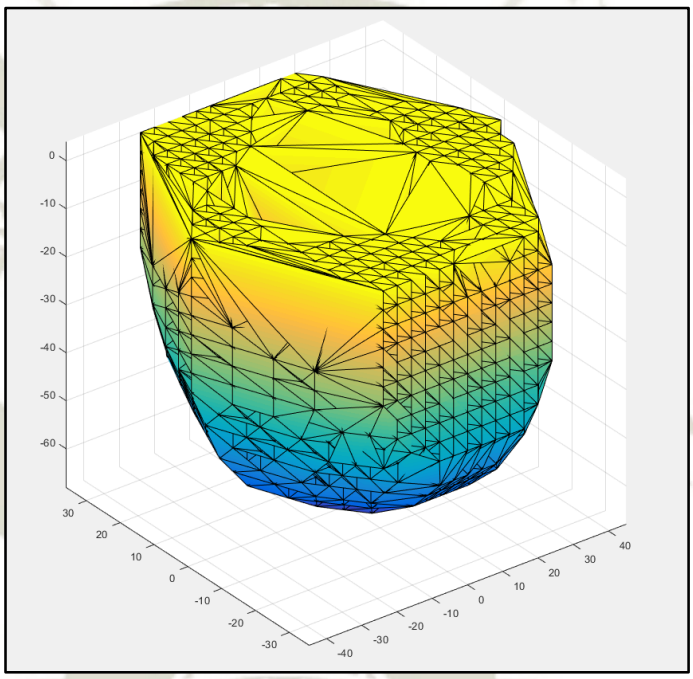


Figura 3.51. Espacio de trabajo del robot paralelo delta
Fuente: *Elaboración propia.*

De la Figura 3.51, se observa que el espacio de trabajo es complejo y tiene notables espacios vacíos. El método que se usó para la representación gráfica del espacio de trabajo es de la triangulación de Delaunay. A medida que los triángulos sean más grandes indica la existencia de espacios vacíos, esto se observa en el centro de la figura indicando que existe un gran espacio vacío, tal como muestra la Figura 3.52 tomando en cuenta solo dos ejes.

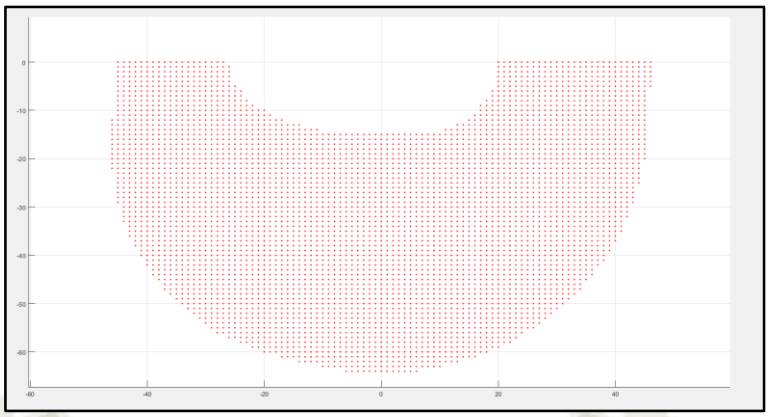


Figura 3.52. Espacio de trabajo del robot paralelo delta considerando los ejes X y Z
Fuente: *Elaboración propia.*

De acuerdo, con el espacio de trabajo visto, es necesario indicar una zona de trabajo exclusiva, que pertenezca al espacio visto y no se acerque a los límites del espacio de trabajo calculado.

**(La programación en MATLAB se encuentra en el Anexo 01)*

3.9. Diseño Electrónico

Luego de tener definido todos los componentes mecánicos, es posible realizar la electrónica del robot paralelo delta. En esta sección del apartado, se definirá todos los componentes electrónicos a usar y explicar el porqué de su uso, también se definirá un esquema detallando todas las conexiones necesarias.

Antes de entrar en consideración de que componentes electrónicos se usaran, es necesario indicar que este proyecto de tesis está enfocado en diseñar e implementar un robot capaz de ejercer diferentes funciones y que sea controlado con programas de uso libre, es decir, que no se desea estar limitado por licencias ni programas especiales que requieren de hardware especial, como por ejemplo; el uso del software de MATLAB®; es muy usado en el ambiente educativo, pero muy poco visto en las industrias, este software requiere de una PC o Laptop con requerimientos especiales tales

como la memoria RAM, tarjeta de video, procesadores avanzados, etc. Atendiendo estas consideraciones y como un medio para agregarle peso al presente proyecto, se decidió usar como controlador principal a un microcontrolador.

El hecho de usar un microcontrolador como dispositivo de control y no como medio para adquirir datos implica:

- Diseñar un programa capaz de cubrir las necesidades del robot.
- Trabajar bajo los requerimientos y limitantes del microprocesador tales como la memoria RAM, velocidad de transmisión, voltaje de operación, memoria RAM, etc.
- Establecer la electrónica en torno al microprocesador.

Finalmente, luego de definir las consideraciones del uso del microcontrolador es posible proceder con el diseño.

3.9.1. Elección de componentes electrónicos

a) Controlador

Es sabido que se usaran los servomotores Dynamixel como medio para ejercer el movimiento del robot. Sin embargo, el cómo será posible esto aún no se ha estudiado.

Robotis, fabricante de los servomotores Dynamixel, ofrece una amplia variedad de controladores compatibles con los servomotores, cada uno de estos controladores posee características únicas que lo hacen especial.

Sin embargo, estos controladores están orientados solo al uso de programas básicos sobre el control de los servomotores, agregar un programa para el control del robot y crear una interfaz luego

resultaría complicado y consumiría significantemente el uso de la memoria RAM. Es por eso que resulta necesario encontrar un controlador lo suficientemente potente, capaz de ejercer la función de manipular los servomotores mientras se realiza las funciones del robot o usar dos controladores capaces de ejercer independientemente las tareas asignadas.

El uso de un solo controlador implica una memoria con alta capacidad, velocidad de procesamiento alta y que sea robusto, sin embargo, un controlador con esas cualidades resulta de alto costo económico. Es por esto que se optó por usar dos controladores, uno capaz de ejercer todas las funciones del robot, tales como la cinemática inversa, modelo diferencial, trayectorias, funciones con la herramienta, etc. Y otro controlador capaz de realizar el control del actuador y realimentar información de este. Este método de comunicación entre controladores es llamado “Maestro – Esclavo”; donde el primero controlador, maestro, es el que toma las decisiones y las envía al segundo controlador, esclavo, que ejecuta las acciones tomadas por el primer controlador.

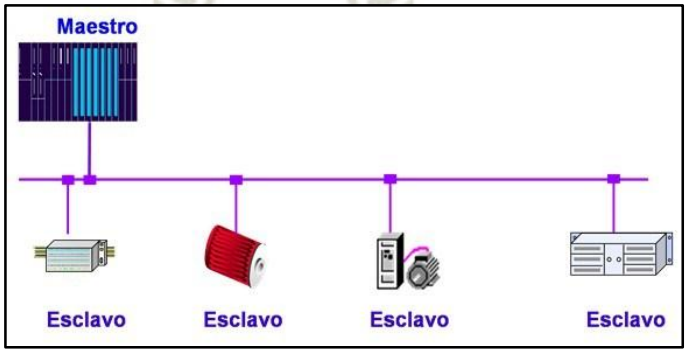


Figura 3.53. Diagrama de comunicación “Maestro – Esclavo”
Fuente: *(Suarez, 2016)*

Como seguimiento de esta actividad y ya definidos el uso de dos controladores, es necesario definir que controladores serán. En primer lugar, en cuanto al controlador maestro, se proponen dos opciones siendo estas la tarjeta Raspberry Pi 3B+ y Arduino Mega.

La tarjeta Raspberry Pi 3B+, es principalmente una computadora completamente funcional ya que cuenta con su propio sistema operativo, tiene una memoria relativamente alta y una velocidad de reloj de 700 MHz, cuenta con varios puertos y entradas USB. Este dispositivo es muy útil para cálculos complejos que requiere mayor velocidad y respuestas inmediatas. No obstante, el adicionar dispositivos es complicado ya que requiere de hardware especial que le permite acoplarse a este, también depende de librerías con términos necesarios para realizar un código, estos pequeños detalles hacen que el uso de una tarjeta Raspberry Pi 3B+ sea un poco tedioso.

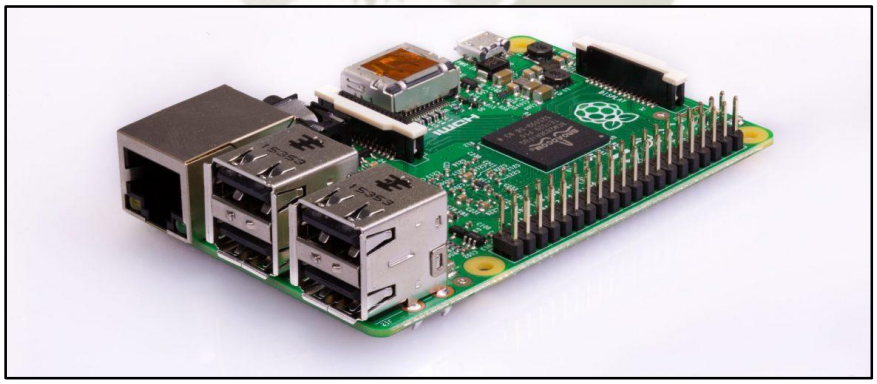


Figura 3.54. Raspberry Pi 3B+
Fuente: *(Raspberry , 2018)*

Por otra parte, la tarjeta de Arduino Mega es considerada un microcontrolador, cuenta con una entrada USB, se ve limitada por una memoria de 256 kB y una velocidad de reloj de 16 MHz. Sin

embargo, cuenta con una simplicidad que reduce la dificultad de la programación, se puede conectar fácilmente con cualquier dispositivo que tenga puertos seriales, su implementación con dispositivos electrónicos es más rápida ya que cuenta múltiples entradas y salidas tanto analógicas como digitales.

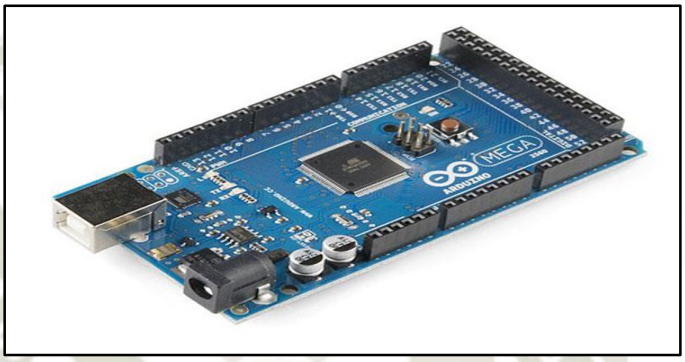


Figura 3.55. Arduino MEGA
Fuente: *(Arduino, 2017)*

Luego de analizar ambas opciones, se decidió usar el controlador Arduino MEGA como dispositivo maestro, ya que, a pesar de sus restricciones, es una plataforma ya conocida con la cual se cuenta con experiencia, tiene un lenguaje de programación C, la cual es compatible con los servomotores Dynamixel, es posible entablar una comunicación serial con el uso de solo dos cables. También, es posible importar librerías con usos específicos tal es el caso de librerías de control PID, control Fuzzy, control de motores, etc. Al mismo tiempo, es posible implementar hardware con facilidad y con códigos simples.

Entorno al controlador que será usado como dispositivo “esclavo”, debe cumplir con las condiciones impuestas por el servomotor, tales como puertos de comunicación TTL o RS-485, lenguaje de

programación compatible y velocidad de funcionamiento alta capaz de comunicarse con el servomotor de manera eficiente y sin pérdida de datos. Atendiendo estas consideraciones y luego de una búsqueda exhaustiva en los catálogos del fabricante se decidió usar la tarjeta OpenCM9.04.

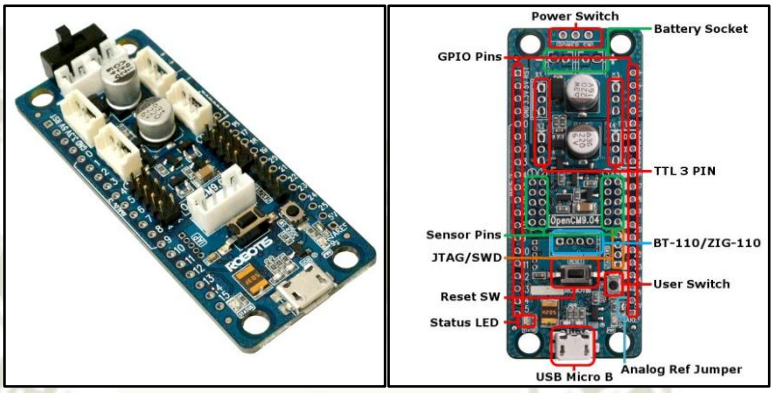


Figura 3.56. OpenCM9.04
Fuente: *(Robotis, 2010)*

Se escogió este dispositivo, porque:

- Cuenta con lenguaje de programación compatible con los servomotores Dynamixel
- Tiene 4 puertos de comunicación TTL o RS-485, esto simplifica la conexión de los servomotores drásticamente, también cuenta con 3 puertos de comunicación serial.
- Posee dos entradas de alimentación una para baterías y otra para fuentes de voltaje.
- Memoria RAM de 128 kB y clock de 72 MHz.

b) Fuente de Poder

A lo largo de los planteamientos hechos, se definió los actuadores y controladores a usar en el presente proyecto, esto significa que también será necesario definir una fuente de poder capaz de satisfacer las necesidades de los actuadores y demás componentes. La selección de la fuente de poder dependerá del voltaje de operación de los actuadores y la corriente máxima a la que pueden llegar. A este respecto, el voltaje de operación recomendado de los servomotores es de 12V DC con una corriente de 2.3 Amperios, ya que se usan 3 servomotores, se decidió usar una fuente de voltaje de 12V DC y 10 Amperios, de esta manera se cubre la carga consumida por los actuadores y también se tiene la energía suficiente para alimentar a la herramienta final.

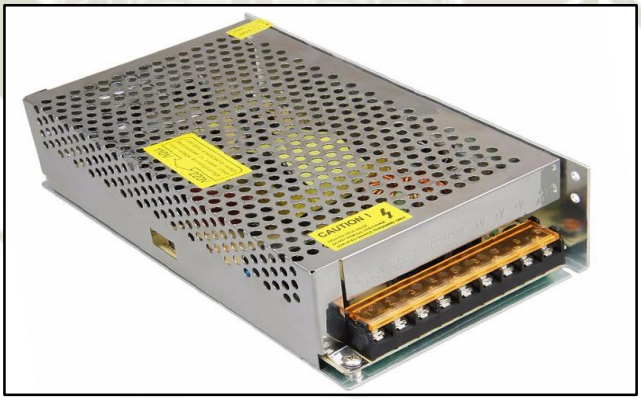


Figura 3.57. Fuente de Voltaje
Fuente: *(AFEL, 2018)*

c) Efector Final

Como ya es sabido, se usará una ventosa de succión como herramienta final del robot paralelo delta. No obstante, el uso de esta herramienta requiere el funcionamiento de una bomba de vacío.

Como ya es sabido que se usara una fuente de 12V DC, la bomba de vacío debe tener el mismo voltaje de operación para facilitar las conexiones. Dicho esto, también se necesita conmutar este componente de manera rápida y a través del controlador. Es por eso que se decidió usar una bomba de vacío conmutada con un transistor a su vez conmutada con una señal del controlador.

Luego de una búsqueda exhaustiva por diferentes catálogos de fabricantes, se escogió usar la bomba de vacío de la marca AIRPO®, porque funciona con un voltaje de alimentación de 12V, tiene una potencia de 12 Watts y un rango de vacío de 0 a 16” de Hg.

Figura 3.58. a) Bomba de vacío AIRPO b) Ventosa de Succión
Fuente: *(Brico Geek, 2017)*

En cuanto al transistor a usar, este debe ser capaz de soportar la corriente máxima de la bomba de vacío, puesto que uno de menor capacidad no sería factible, este también debe contar con un disipador de calor y un voltaje de emisor-colector mayor a 12V. Atendiendo a estos requerimientos se usará el transistor TIP31C, como medio para conmutar a la bomba de vació.

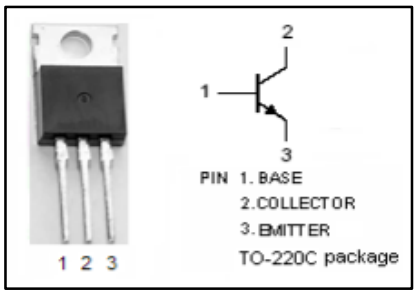


Figura 3.59. Transistor NPN TIP31C
Fuente: *(ArduShop, 2018)*

Puesto que la bomba de vacío funciona con un motor, se colocará un diodo en paralelo a este componente, con el fin de proteger al transistor y al controlador de la FEM que se produce cuando se activa la bomba de vació.

Respecto a la ventosa de succión, su elección depende de la potencia de la bomba y su fuerza de retención; luego de una búsqueda exhaustiva y con la facilidad de encontrarla en el mercado local se optó por escoger la ventosa de succión de la marca FESTO modelo VASB – 15 – 1/8 – PUR ; porque:

- Posee un rango de vacío de 0 a 22 “ de Hg
- Tiene una fuerza nominal de retención de 8.5 N equivalente a 870 gramos.
- Posee resistencia a la corrosión.
- Es liviana y sus dimensiones permiten acoplarlo fácilmente al robot.

**(Para ver información sobre la bomba de vacío y ventosa referirse al anexo 07)*

d) Joystick

Este componente, cumple la función de manipulación del robot paralelo delta, ya que nos permite controlar manualmente el movimiento del robot. A diferencia de los componentes anteriormente vistos, el joystick no se encontrará conectado directamente al controlador, en cambio esta ira conectado a la PC, que será usada como interfaz entre el robot y el humano. La selección de este componente dependerá solamente del precio, puesto que la mayoría de joystick en el mercado son confiables y todos cumplen la misma función, siendo así que se optó por escoger el joystick Metal Strike de la marca Genius®, por su bajo precio y la conectividad USB sin drivers que ofrece, haciendo a este equipo más versátil.

Figura 3.60. Joystick Metal Strike
Fuente: *(Lelong, 2010)*

3.9.2. Diseño Preliminar del Circuito

Una vez definidos todos los componentes necesarios, es posible realizar un esquema o diagrama que indique como es que se integran los componentes. Este diseño estará basado en los servomotores puesto que es el elemento que requiere más carga.

Los servomotores Dynamixel ofrecen una conexión en cadena, este tipo de conexión facilita la comunicación con todos los servomotores, una vez conectados en serie todos los servomotores bastan con conectar al puerto del controlador OpenCM9.04

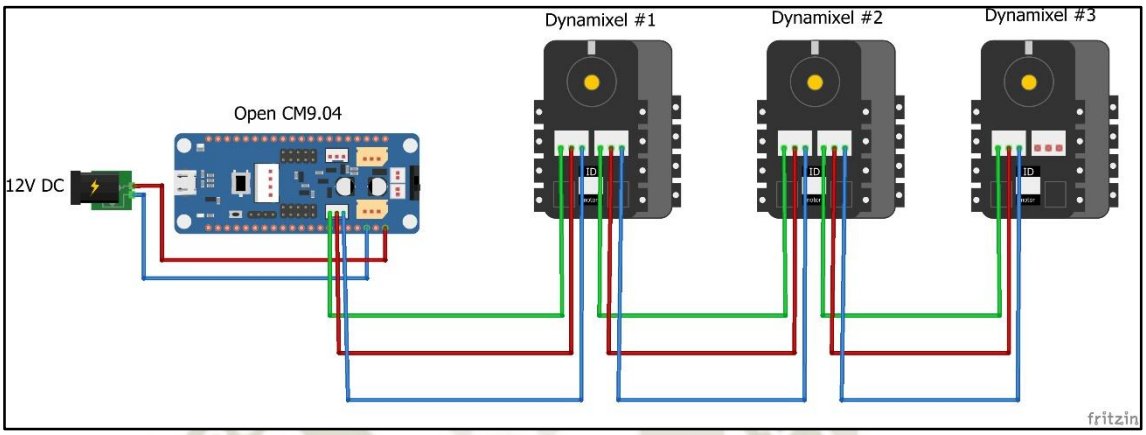


Figura 3.61. Servomotores Dynamixel conectados en serie
Fuente: *Elaboración propia.*

Sin embargo, al estar conectados en serie existe la posibilidad de que, si un servomotor llegase a fallar, haría que se pierde la comunicación y energía de los demás servomotores. También debido a esta conexión, el voltaje suministrado al dispositivo varía dependiendo de la carga consumida, no logrando un voltaje constante de 12V a lo largo del tiempo, sumándole a este toda la cantidad de cables, existe una caída de tensión considerable. Estos aspectos nos sirven para replantear si es reamente conveniente conectar en serie los servomotores.

Atendiendo a estas consideraciones, al conectar en paralelo los servomotores es posible solucionar el problema de la caída de tensión. Puesto que, una conexión en paralelo brinda la misma tensión a todos los actuadores. Con respecto a la comunicación, se optó por tener una topología tipo bus. Esto implica conectar todos los servomotores a una

red principal, logrando así reducir el cableado y solo conectar el cable de comunicación y tierra al controlador. Cabe recalcar, que los controladores serán alimentados mediante la PC.

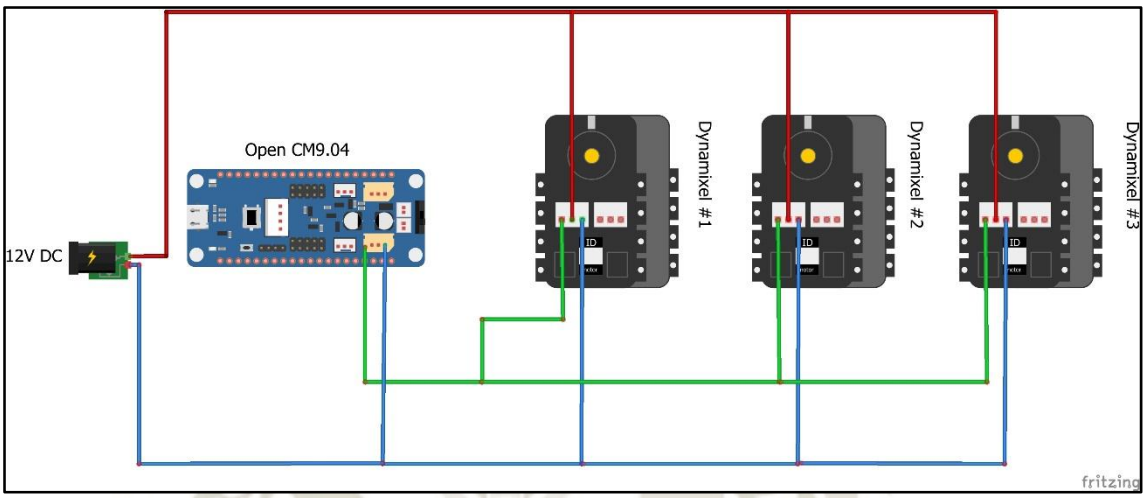


Figura 3.62. Servomotores Dynamixel conectados en paralelo
Fuente: *Elaboración propia.*

De la Figura 3.62 se observa que los servomotores Dynamixel se encuentran en paralelo, logrando así reducir el cableado. También, se indica la conexión de la comunicación mediante el cable de color verde, esto indica una topología tipo bus que incluye a los 3 actuadores. Los únicos cables que van conectados al controlador OpenCM9.04 son los de comunicación y tierra, esto reduce el riesgo de dañar la placa por sobre-corriente.

El siguiente punto a tratar es la comunicación entre el controlador OpenCM9.04 con el controlador Arduino MEGA, esto se logra básicamente conectando los cables de comunicación serial que poseen estos dos controladores, basta con conectar el cable de transmisión “TX” de un controlador con el de recepción “RX” del otro controlador y viceversa.

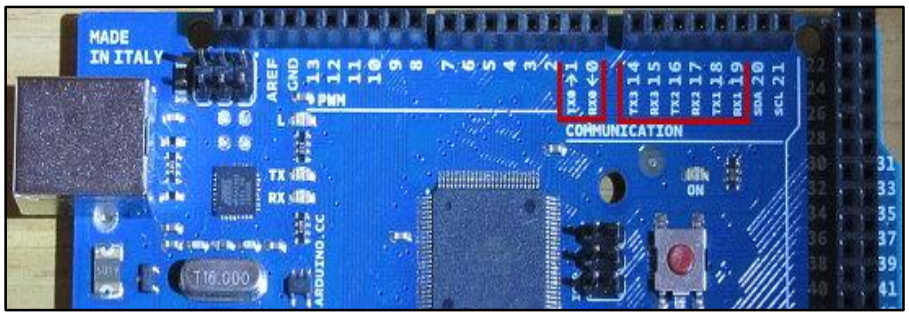


Figura 3.63. Puertos de comunicación serial del Arduino MEGA
Fuente *(Starting Electronics Blog, 2017)*

En cuanto al efector final, este será activado por una señal digital proveniente del controlador Arduino MEGA. Asimismo, esta señal conmutara a un transistor, para luego conmutar a la bomba de vacío.

**(Para visualizar el planos electro neumático referirse al Anexo 04)*

Capítulo IV

4. IMPLEMENTACIÓN

La implementación es un proceso continuo que incluye un conjunto de actividades diseñadas para poner en práctica un programa o actividad (que sabemos que funciona). Pasar del concepto del diseño a la implementación puede ser difícil; puesto que, en el proceso de hacer coincidir los hallazgos científicos con las necesidades específicas, suelen aparecer adversidades que no se tomaron en cuenta a la hora del diseño. En la implementación se busca plasmar en físico todas las ideas propuestas en el diseño y al mismo tiempo validarlas, en caso surgiesen problemas, se buscará dar una solución eficaz y que sea capaz de satisfacer las necesidades específicas planteadas del diseño.

Este capítulo abarcará tres temas, el primer tema estará referido a la implementación mecánica del robot paralelo delta, el segundo tema, estará enfocado a la implementación del sistema electrónico y finalmente el tercer tema estará enfocado a crear un programa capaz de controlar todas las funciones del robot.

4.1. Implementación Mecánica

La implementación mecánica está referida a la acción de plasmar en físico el diseño mecánico planteado. En esta sección del apartado, se indicará los procesos llevados a cabo para la obtención de las partes mecánicas del robot paralelo delta.

4.1.1. Herramientas para la Implementación Mecánica

Las herramientas usadas para llevar acabo la implementación mecánica fueron principalmente tres. La primera herramienta usada es el software de Autodesk Inventor®, con este programa fue posible diseñar al detalle

todos los componentes del robot paralelo delta, también se pueden crear los planos de detalle y planos de ensamble.

Este software abre paso a la segunda herramienta, la cual será una impresora 3D, esta máquina es capaz de imprimir figuras con volumen a partir de un diseño creado por computadora.

Básicamente, las impresoras 3D crean objetos con 3 dimensiones y esto se logra construyendo capas sucesivamente hasta obtener la pieza deseada. Estos dispositivos trabajan con diversos materiales tales como el ABS, PLA, Laybrick, Filaflex, etc.

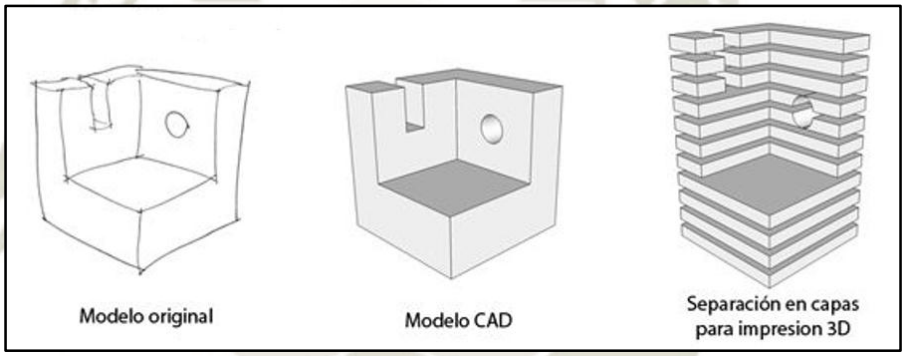


Figura 4.1. Proceso de la Impresión 3D
Fuente: *(3D Market, 2016)*

Para llevar a cabo la impresión 3D de las partes del robot paralelo delta, se usará una impresora 3D de la marca PRUSA modelo I3 para los componentes pequeños tales como la base móvil, base del servomotor y base de la chumacera. Asimismo, para las piezas más grandes como los brazos y la base fija se usará una impresora 3D de la marca Da Vinci. Antes de entrar en consideración, es necesario indicar que el costo de la impresión 3D depende de la calidad en la que se imprima. Esta calidad viene representada por la densidad del relleno, mientras más porcentaje

de relleno se obtenga, la pieza será más sólida, aumentando así su rigidez mecánica. Además de la densidad del relleno el costo también se ve afectado por el patrón de relleno, que proporciona propiedades mecánicas diferentes.

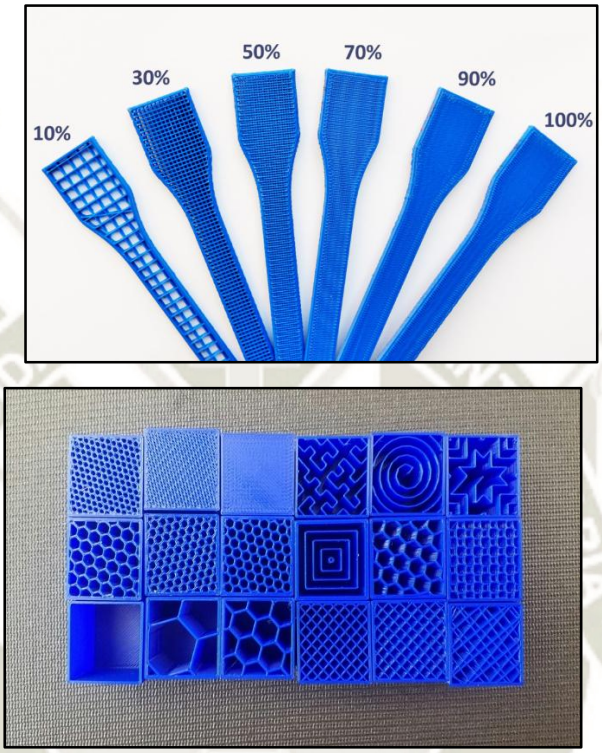


Figura 4.2. a) Densidad de relleno de la impresión 3D b) Patrón de relleno de la impresión 3D
Fuente: *(DYOR: Do Your Own Robot, 2016)*

Finalmente, la tercera herramienta se refiere a un torno, este es necesario para maquinar los antebrazos del robot. Si bien es cierto, que la impresión 3D es una herramienta capaz de brindar piezas de cualquier forma; esta no será necesaria para la fabricación de los antebrazos, ya que la forma simple que posee hace innecesaria el uso de esta. También, con el torno es posible lograr los hilos necesarios para ensamblar los antebrazos con las rotulas.

4.1.2. Modelo Virtual del Robot Paralelo Delta

Para la realización de la impresión 3D, es necesario contar con un modelo previo que indique como se verá el modelo una vez ensamblado.

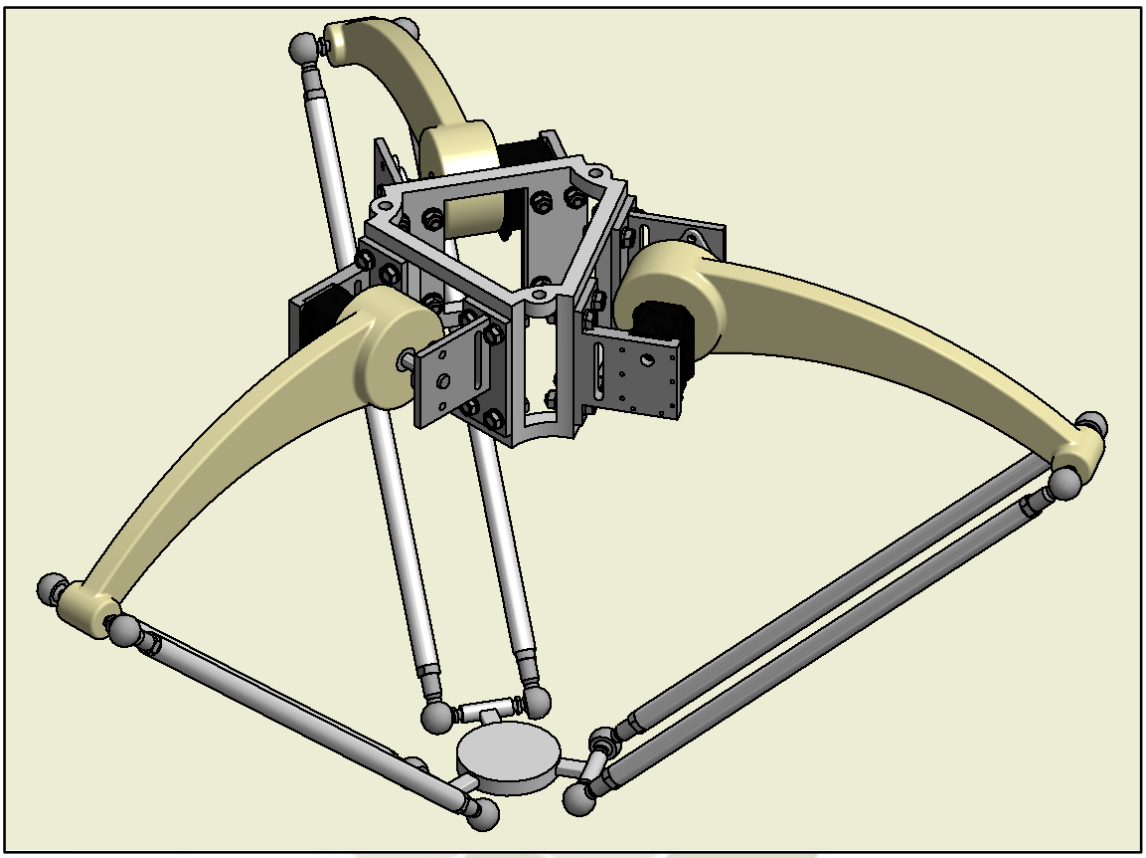


Figura 4.3. Modelo virtual del robot paralelo delta
Fuente: *Elaboración propia.*

Como ya es sabido, las piezas a imprimir en 3D, son los brazos, base fija, base móvil, base del servomotor y base de la chumacera. Los antebrazos al ser un elemento con sección transversal constante, usaran otro material, siendo este Nylon 66, por su baja densidad y facilidad de maquinar.

De la Figura 4.3 se observa que existe una simetría muy importante que se tiene que cumplir en la fabricación, los componentes deben estar correctamente alineados y coincidir en el momento del ensamble. Ya que el robot paralelo delta posee una cadena cinemática cerrada, si es

que hubiese un error en el ensamble este se apreciaría directamente, puesto que de alguna manera alteraría a los demás eslabones. Finalmente, con el objetivo de tener una idea más concisa de cómo lucirá el robot, una vez que se impriman y se ensamblen todos los componentes, al modelo obtenido de la figura anteriormente expuesta, se le realizo una renderización.

Figura 4.4. Vista isométrica del modelo renderizado del robot paralelo delta
Fuente: *Elaboración propia.*

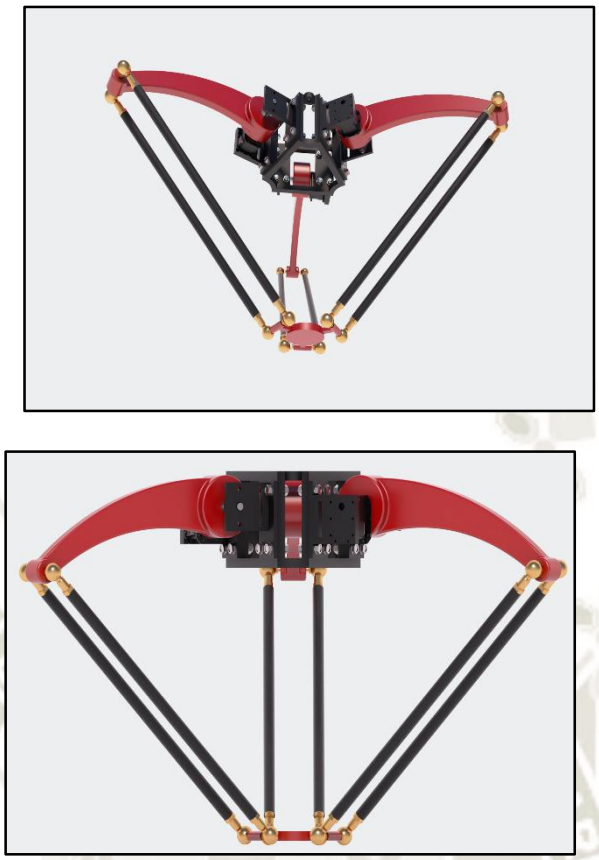


Figura 4.5. Vista isométrica y frontal del modelo renderizado del robot paralelo delta
Fuente: *Elaboración propia.*

En caso de las Figurar 4.4 y 4.5, se observa el robot paralelo delta renderizado, dándole un aspecto más real, para este modelo se tomó en cuenta el color, de los componentes del robot siendo este un color rojo y negro mate. También, se pueden apreciar las juntas esféricas de color dorado, y en la base fija las uniones empernadas. Este modelo contempla únicamente las piezas mecánicas, más no los componentes electrónicos, aun así, se espera que los componentes electrónicos se encuentren en un tablero alejado del robot.

Teniendo en cuenta el diseño planteado y ya visto un modelo que se asemeje más a la realidad, se procederá a implementar todos los componentes. Indicar, asimismo, que este es el modelo esperado del

robot paralelo delta y por lo tanto al finalizar la implementación, el robot paralelo delta deberá lucir exactamente igual.

4.1.3. Impresión 3D

Una vez obtenido todos los componentes del robot como un modelo CAD, es posible llevar este formato a la impresora 3D. Una vez que el modelo se encuentre listo, es importante definir la calidad de impresión que se usara, siendo estas calidad buena, intermedia y excelente. Como se planteó en el diseño conceptual, se espera tener componentes sólidos, es por eso que a primera instancia se planteó utilizar una calidad excelente logrando así casi un 100% de la densidad del relleno. No obstante, una calidad excelente conlleva más tiempo de impresión y los costos de impresión dependen del tiempo de impresión. Usar una calidad excelente en todos los componentes aumentaría los costos enormemente, es por esta razón que se decidió usar dos calidades de impresión.

Para las piezas más grandes, como los brazos y la base fija, se usará una calidad excelente, logrando así un 40% de la densidad de relleno. En cambio, para las piezas más pequeñas como la base móvil, base del servomotor y base de la chumacera se usará una calidad intermedia, puesta que estas piezas al tener menos volumen su tiempo de impresión es mucho menor comparado con el de las piezas grandes. Tomando en cuenta todas estas consideraciones se obtienen los siguientes parámetros dados por las impresoras 3D.

Tabla 4.1. Parámetros de la impresión 3D

Ítem	Cant.	Unid.	Descripción	
001	3	Pza.	**Brazo**	
			Material:	ABS
			Dimensión:	295mm x 77.45 mm x 35mm
			Impresora:	Prusa i3
			Calidad:	Buena
			Tiempo de Impresión:	13h 28 min c/u
			Peso:	111 g c/u
002	3	Pza.	**Base de la Chumacera**	
			Material	ABS
			Dimensión:	35 mmx 80 mm x 62 mm
			Impresora:	Da Vinci AIO 0.1 XYZ
			Calidad:	Intermedia
			Tiempo de Impresión:	8 h c/u
			Peso:	11 g c/u
003	3	Pza.	**Base del Servomotor**	
			Material	ABS
			Dimensión:	35 mmx 80 mm x 62 mm
			Impresora:	Da Vinci AIO 0.1 XYZ
			Calidad:	Intermedia
			Tiempo de Impresión:	8 h c/u
			Peso:	11 g c/u
004	1	Pza.	**Base Fija**	
			Material	ABS
			Dimensión:	156.99mm x 137.58 mm x 100mm
			Impresora:	Prusa i3
			Calidad:	Buena
			Tiempo de Impresión:	24 h 10 min c/u
			Peso:	100 g c/u
005	3	Pza.	**Base Móvil**	
			Material	ABS
			Dimensión:	110.22 mm x 95.46 mm x 10 mm
			Impresora:	Da Vinci AIO 0.1 XYZ
			Calidad:	Intermedia
			Tiempo de Impresión:	4 h 40 min c/u
			Peso:	11 g c/u

Fuente: *Elaboración propia.*

De la Tabla 4.1, se especifica todos los parámetros que se toman en cuenta al momento de la impresión. Siendo el parámetro más importante, el tiempo de impresión, este refleja que pese a tener una calidad de impresión baja, el tiempo es relativamente mayor comparado con la impresión en calidad intermedia de las piezas pequeñas. Habiendo descrito las características más relevantes se procede con la impresión de los componentes del robot paralelo delta.

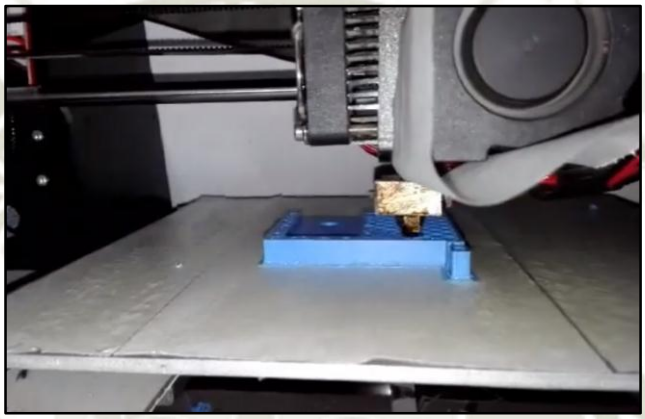


Figura 4.6. Impresión 3D de la base del servomotor
Fuente: *Elaboración propia.*

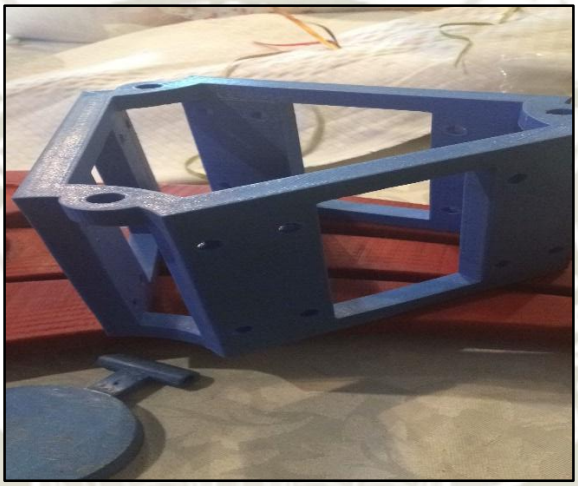


Figura 4.7. Brazos, base móvil y base fija impresos
Fuente: *Elaboración propia.*

Figura 4.8. Base de la chumacera y base del servomotor
Fuente: *Elaboración propia.*

De las Figuras 4.7 y 4.8 es posible apreciar que los componentes planteados en el modelo coinciden en forma con la impresión 3D realizada, también se observa las capas de impresión dadas. Estas capas pueden eliminarse con un proceso de lijado y pulido lo cual no sería problema alguno. Sin embargo, al manipular las piezas grandes se puedes escuchar varios crujidos provenientes de la parte interna de la pieza, indicando que es un elemento sensible.

Partiendo de la situación anterior cabría preguntarse, si las propiedades mecánicas se mantienen luego de tener los componentes impresos; obteniendo como respuesta un no. Al ser impresos mediante capas, con

un patrón de relleno y una calidad baja, no se obtiene una pieza totalmente sólida, esto complica totalmente el diseño planteado; ya que estos patrones de relleno no brindan las mismas características que una pieza sólida. En otras palabras, al imprimir las piezas con una densidad de relleno menor a la planteada, disminuye la rigidez y como consecuencia hace que la pieza sea más sensible frente a las torsiones generadas. Una manera de solucionar este problema es reforzando las piezas y esto se puede lograr con ayuda de la fibra de vidrio.

4.1.4. Reforzamiento con Fibra de Vidrio

La fibra de vidrio es un material que consta de numerosos hilos de vidrio extremadamente finos entrelazados en varias configuraciones o formas diferentes. Posee un costo bajo y su uso extendido se basa en las siguientes características:

- Baja tecnología de fabricación
- Durabilidad
- Alta tolerancia a la flexión
- Resistencia a la corrosión y a la humedad
- Ligero
- No es un conductor de electricidad

Este material es altamente usado en la industria automotriz como un refuerzo de las piezas de plástico de la carrocería. La resistencia del plástico reforzado dependerá del tipo de fibra de vidrio, la cantidad usada y el tipo de resina.

Figura 4.9. Fibra de vidrio
Fuente: *(PLAREMESA, 2018)*

Ya que los elementos del robot impresos no cumplen las condiciones mecánicas necesarias para ejercer los movimientos de robot, serán reforzados con fibra de vidrio. Este refuerzo permitirá a todas las piezas, cumplir con las condiciones mecánicas impuestas en el diseño conceptual. Sin embargo, al aplicar la fibra de vidrio los componentes sufrirán cambios en su volumen; aumentando el grosor algunos milímetros y alterando su rugosidad.

Para llevar a cabo el refuerzo de las piezas se usará fibra de vidrio tipo E, que cuenta con las siguientes características:

- Tenacidad (N/tex): 1.30
- Fuerza a la tracción (MPa): 3400
- Elongación hasta rotura (%): 4.5
- Conductividad Térmica (W/m. K): 1
- Factor de disipación dieléctrica: 0.0010 - 0.0018 a 106 Hz
- Resistencia a los disolventes: alta
- Resistencia a microorganismos: alta

Para la realización del refuerzo con fibra de vidrio a las piezas del robot se usarán dos capas de fibra de vidrio y resina de poliéster, otorgando así una alta resistencia y aumentando su grosor en 2mm. Con el fin de

lograr un acabado liso y estético, se aplicará un Gel-Coat a toda la superficie no sin antes haber lijado todas las imperfecciones.

Figura 4.10. Piezas del robot paralelo delta reforzadas con fibra de vidrio
Fuente: *Elaboración propia.*

De la Figura 4.10 se observa que a todas las piezas del robot fueron reforzadas con fibra de vidrio; a los brazos se les reforzó por todo su contorno por ser la pieza más crítica en el movimiento del robot. En cuanto a las piezas restantes estas fueron levemente reforzadas en algunas zonas. Por otra parte, el uso de la fibra de vidrio crea grumos en algunas zonas de la superficie de los componentes del robot. Estas imperfecciones deben ser removidas con una lija fina, para poder aplicar el Gel Coat.

Figura 4.11. Brazo recubierto con fibra de vidrio y Gel - Coat
Fuente: *Elaboración propia.*

Luego de aplicar Gel – Coat, las piezas del robot adquieren una superficie lisa, logrando así un buen acabado.

4.1.5. Ensamble y Ajustes

Una vez obtenidas todas las piezas y luego de ser reforzadas con fibra de vidrio, es posible realizar un pre-ensamble. En este se busca tener un armado preliminar de todos los componentes, es necesario indicar que, debido al reforzamiento con fibra de vidrio, se crearon algunas imperfecciones a lo largo de la superficie de las piezas. Una superficie irregular en piezas como la base del servomotor y la base de la chumacera, tiene como consecuencia brazos desalineados, ya que el eje que se proyecta del servomotor con el de la chumacera no coincide. Para solucionar este problema es necesario el uso de lainas, con estas herramientas es posible lograr coincidir los ejes proyectados, logrando así; simetría en cada uno de los lados.

Con el fin de lograr un ensamble correcto; se hace uso del Anexo 03. En este se encontrará todos los planos de las piezas y los procedimientos para el ensamble.

Figura 4.12. Ensamble preliminar de la base fija, base del servomotor y base chumacera
Fuente: *Elaboración propia.*

La Figura 4.12 refleja la simetría que debe portar cada lado de la base fija, también se aprecia que el eje del servomotor con el eje de acero, coinciden perpendicularmente visto desde una perspectiva superior. Realizar un ensamble preliminar de cada componente es sumamente importante, ya que de esta manera se evita problemas futuros. Aun así, es necesario verificar que todos los lados de la base fijan tengan la misma altura y coincidan con los ejes.

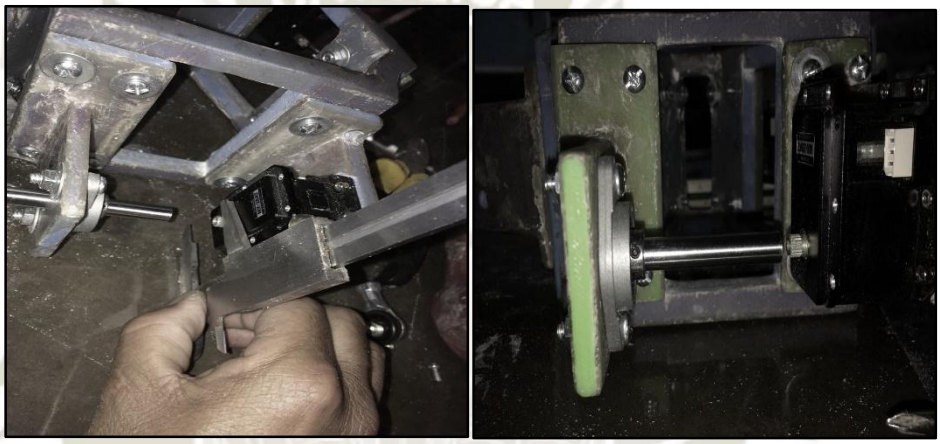


Figura 4.13. Alineamiento de los ejes con servomotor
Fuente: *Elaboración propia.*

De la Figura 4.13, se observa que el eje del servomotor Dynamixel se encuentra correctamente alineado con el eje que sobresale de la chumacera. Hay que hacer notar, que si los elementos no estarían alineados; las posiciones cartesianas a alcanzar del robot serian erróneas. Puesto que un leve cambio en su geometría, altera los 120° grados de desfase necesarios para lograr una forma delta.

Una vez montada la zona superior del robot, se procede con el montaje de los ante brazos, cabe recalcar que estos son los únicos componentes maquinados y que no poseen un reforzamiento con fibra de vidrio.

Figura 4.14. Antebrazos torneados
Fuente: *Elaboración propia.*

Luego de tener todos los ejes alienados se procede con un ensamble preliminar de todas las piezas.

Figura 4.15. Ensamble preliminar del robot paralelo delta
Fuente: *Elaboración propia.*

A continuación, luego de comprobar la movilidad del robot paralelo delta, se procede a pintar todos los componentes del robot, logrando así que el robot luzca como el modelo virtual planteado.

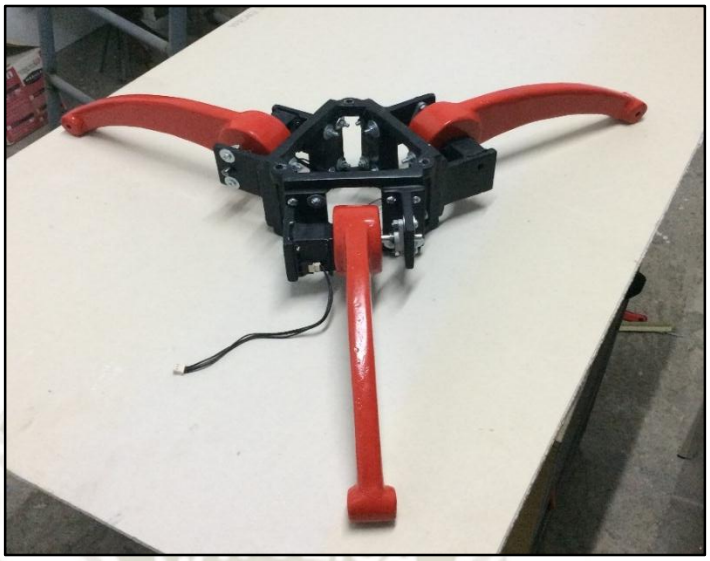


Figura 4.16. Parte superior del robot ensamblada y pintada
Fuente: *Elaboración propia.*

Al ya tener todas las piezas que componen al robot ensambladas y pintadas; es necesario proveer a este de su soporte. En el diseño conceptual, se planteó el uso de una jaula, esta estructura está hecha de acero con perfil cuadrado de 1” x 1” x 2mm. En cuanto a los planos de fabricación de la jaula, estos se encuentran en el anexo 03.

Figura 4.17. Jaula de Soporte
Fuente: *Elaboración propia.*

Finalmente, ya con la jaula de soporte fabricada, los componentes del robot pintados y ensamblados, es posible montar el robot paralelo delta

en su totalidad. Asimismo, es importante mencionar que la jaula solo cumple la función de sostener al robot.

Figura 4.18. Ensamble Final del Robot Paralelo Delta
Fuente: *Elaboración propia.*

4.2. Implementación Electrónica

En esta sección del apartado, se plasmará la idea del esquema electrónico planteado anteriormente en el diseño conceptual y además se detallará cada punto importante que implica el uso de los controladores.

4.2.1. Herramientas para la Implementación Electrónica

A primera vista, el circuito electrónico planteado en el diseño conceptual; no es complejo ya que, los servomotores Dynamixel ofrecen una configuración donde la electrónica es muy poca usada. Sin embargo, con el fin tener una mejor organización y reducir el cableado se implementará una placa electrónica.

Para llevar a cabo la realización de la placa electrónica fueron necesario dos herramientas, siendo la primera el software Eagle® para el diseño del diagrama y PCB del circuito preliminar planteado y la segunda

herramienta; un multímetro para la medición y revisión de los componentes electrónicos y la placa electrónica.

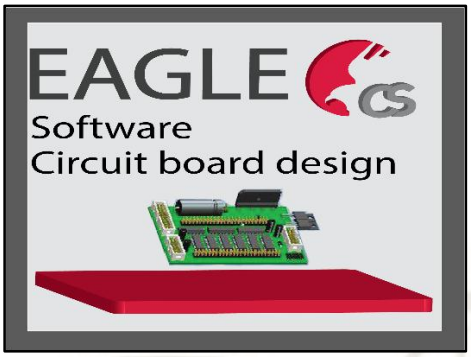
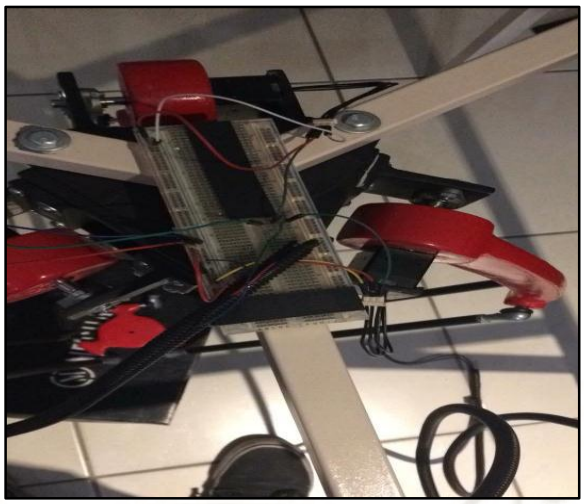


Figura 4.19. Software EAGLE
Fuente: *(Tindie, 2018)*

4.2.2. Armado Preliminar

Tomando en cuenta el circuito preliminar planteado en el diseño conceptual, se realizó el armado de dicho circuito en un protoboard, de esta manera se obtuvo lo siguiente.



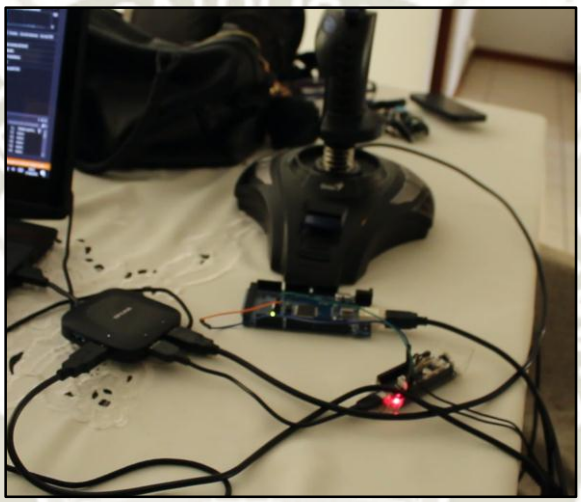


Figura 4.20. Conexión preliminar del circuito
Fuente: *Elaboración propia.*

De la Figura 4.20, se observa que los cables que salen del servomotor, son conectados al protoboard, y luego estos son llevados a los controladores. La conexión de la comunicación entre los controladores es simple y se logra solo con el uso de 3 cables, uno para la transmisión “Tx”, el segundo para la recepción “Rx” y finalmente el tercero para la tierra “GND”. Como se indicó en el diseño conceptual, tanto el controlador maestro como el controlador esclavo, serán alimentados por la computadora. De igual manera, el joystick también tendrá una

conexión con el ordenador, originando así ocupar la mayoría de los puertos USB de la computadora.

Con el fin de solucionar este problema, se sumó a los componentes electrónicos el uso de un Hub USB de la marca TP-Link, de esta manera se evita saturar los puertos USB que tiene una PC.

Figura 4.21. Hub TP-Link
Fuente *(TP-LINK, 2018)*

Dicho Hub, se puede observar en el lado derecho de la Figura 4.20, mediante este se conectó los dos controladores y el joystick. Algo muy importante que no se vio en el conexionado preliminar anterior, fue el circuito que da paso al uso del efector final. Con respecto a este conexionado, se implementó un circuito preliminar independiente del anterior ya visto, esto se hizo con el fin de realizar pruebas independientes de la bomba de vació. Aun así, su conexionado prevalece tal como se indicó en el diagrama de conexión planteado en el diseño conceptual, obteniendo como resultado lo siguiente.

**(Para visualizar los planos de conexión electro- neumática referirse al Anexo 04)*

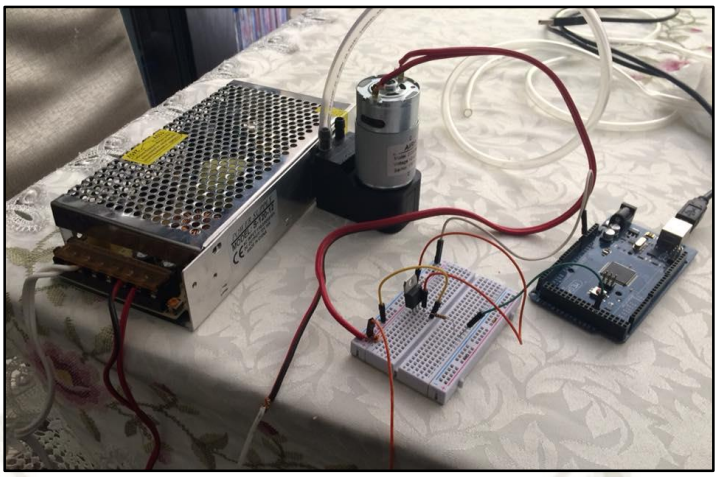


Figura 4.22. Conexión preliminar del efector final
Fuente: *Elaboración propia.*

De la Figura 4.22, se observa que el controlador Arduino Mega, está conectado al protoboard por dos cables, siendo el cable de tierra y el de la señal digital dirigida al transistor que conmutara a la bomba.

4.2.3. Prueba de Funcionamiento del circuito

Realizar una prueba de funcionamiento del circuito, implica tener un algoritmo predefinido capaz de ejecutar todas las acciones del robot. No obstante, este código aun no fue planteado; por lo que una prueba de funcionamiento completa requeriría el planteamiento e implementación de los algoritmos necesarios para llevar a cabo los movimientos del robot. Estos algoritmos serán planteados en la siguiente sección del apartado. Aun así, con el fin de proveer una prueba que valide el circuito planteado e implementado en un protoboard, se mandara instrucciones a los controladores que conlleven mover los servomotores a posiciones angulares aleatorias que se encuentre en el espacio de trabajo del robot paralelo delta.

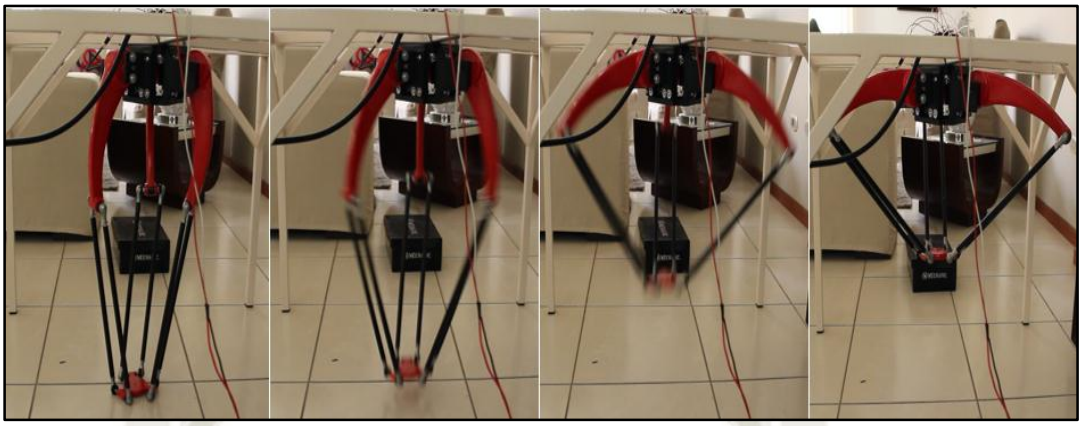


Figura 4.23. Prueba de funcionamiento del robot con el circuito preliminar
Fuente: *Elaboración propia.*

4.2.4. Impresión del Circuito

Habiéndose probado el funcionamiento del circuito planteado en el diseño conceptual, es posible plasmarlo en una placa electrónica; para esto se recurrió al diseñó planteado en el capítulo anterior.

**(Para visualizar los planos de conexión electro-neumática referirse al Anexo 04)*

El resultado obtenido es el siguiente:

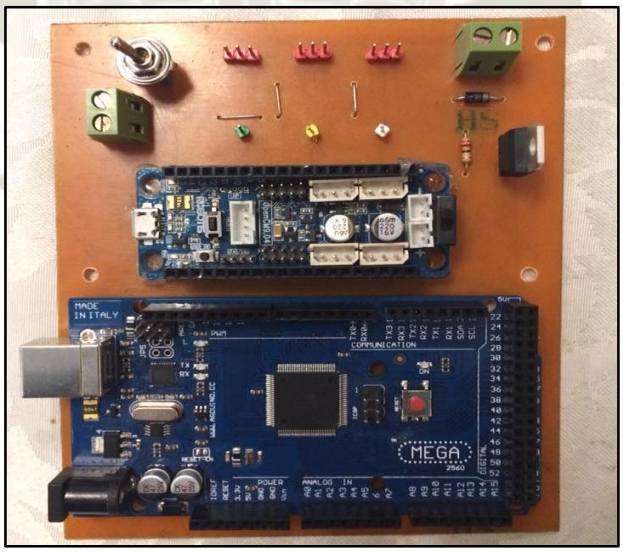


Figura 4.24. Placa electrónica implementada
Fuente: *Elaboración propia.*

Respecto a la Figura 4.24, se observa que es lo suficientemente grande, para que los dos controladores encajen correctamente, cabe resalar que

estos controladores estarán atornillados y correctamente aislados de la placa; esto se hace para evitar alguna falla eléctrica en los controladores y la placa impresa.

4.3. Implementación del Software

El propósito principal de la implementación de un software respecto al robot paralelo delta, es lograr ejecutar las acciones o tareas encomendadas hacia este. Sin un software, el robot no sería capaz de ejecutar ninguna acción y todo lo anteriormente estudiado y analizado no cobraría sentido.

No es objetivo del presente proyecto de tesis el implementar un software completo y multifuncional para el robot, ya que, al ser un prototipo, no requiere de un programa excesivamente complejo que brinde todas las características de un robot industrial. Por lo tanto, en este trabajo de tesis, se presentarán diagramas de flujo que indiquen el funcionamiento del controlador maestro, controlador esclavo y la interfaz. También se explicará los algoritmos computacionales más relevantes del proyecto, tales como la comunicación entre dispositivos, instrucciones importantes del controlador de los servomotores Dynamixel y consideraciones hechas sobre el controlador Arduino Mega.

4.3.1. Especificaciones del Programa

Antes de entrar en consideración, es necesario definir las bases del programa y especificar las funciones que podrá realizar. Para esto se plantea un diagrama que especifica el flujo de la comunicación de todos los dispositivos a usar.

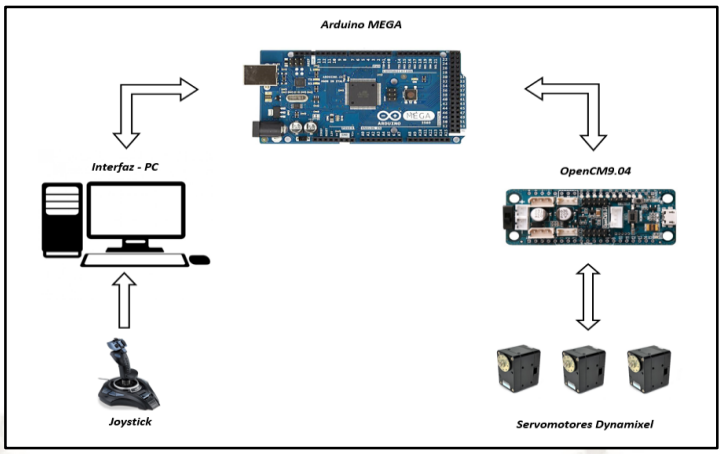


Figura 4.25. Diagrama de comunicación de dispositivos
Fuente: *Elaboración propia.*

La Figura 4.25 indica que existe una comunicación bidireccional entre el controlador Arduino MEGA con la PC y el controlador OpenCM9.04. Esto quiere decir que los datos recibidos por el controlador de los servomotores Dynamixel, pueden ser enviados al controlador maestro y estos a su vez mostrados en la PC. Respecto al joystick, este solo proporcionara sus datos a la PC; estos datos pueden ser enviados al controlador maestro y gracias a la comunicación bidireccional también puede ser enviado a los servomotores por medio de su controlador.

Con respecto al modo de comunicación, la tarjeta Arduino MEGA ofrece una comunicación serial de modo Half-Duplex, esto implica que no es posible una comunicación de manera simultánea, porque los datos tienen que turnarse. Así mismo, la tarjeta OpenCM9.04 también posee un modo de comunicación Half-Duplex tanto con el puerto serial y con el puerto TTL. Esto implica, que los datos transmitidos y recibidos deben poseer un medio que indique el inicio y el final de la cadena de caracteres enviados, para que no existan problemas en la comunicación.

Partiendo de las consideraciones anteriores, es posible plantear un programa que ejerza diferentes tareas y sea capaz de interactuar en tiempo real con el robot.

Primero, es necesario que existan maneras de confirmar que todos los dispositivos se encuentran conectados; segundo, el robot paralelo delta debe llegar a una posición preestablecida cada vez que inicie y cuando se desee; tercero, cuando se active una opción de manipulación el robot debe responder correctamente; cuarto, se debe visualizar los datos del robot en el momento que se use; quinto, el programa debe ofrecer diversas opciones y se debe tener un pleno control del robot.

Lo anteriormente expuesto, implica la creación de diversas subrutinas dentro de un programa principal y demanda un control pleno sobre los servomotores Dynamixel, ya que estos son los dispositivos que retroalimentaran toda la información.

Finalmente es importante decidir las acciones que llevara a cabo el controlador maestro, el controlador esclavo y la PC. Respecto a la interfaz esta debe ser capaz de brindar la visualización de los datos adquiridos por los actuadores y también debe brindar opciones para la manipulación del robot. El controlador maestro, Arduino Mega, debe realizar las operaciones matemáticas que involucren el movimiento de robot, también debe ofrecer una correcta recepción y transmisión de datos a la interfaz, controlar el uso del efector final y enviar datos de posición angular y velocidad al controlador de los servomotores. Finalmente, el controlador esclavo, OpenCM9.04, debe transmitir y recibir los datos sin problema alguno, lograr el control Fuzzy propuesto

como método de control en el presente proyecto de tesis y seguir las instrucciones dadas por el controlador maestro.

4.3.2. Diagrama de Flujo

Un diagrama de flujo es un esquema que representa de manera gráfica un algoritmo, se basa en el uso de diversos símbolos para representar operaciones específicas. Con los diagramas de flujo se busca expresar los algoritmos del robot paralelo delta, de manera simple. A continuación, se detallarán todos los algoritmos usados:

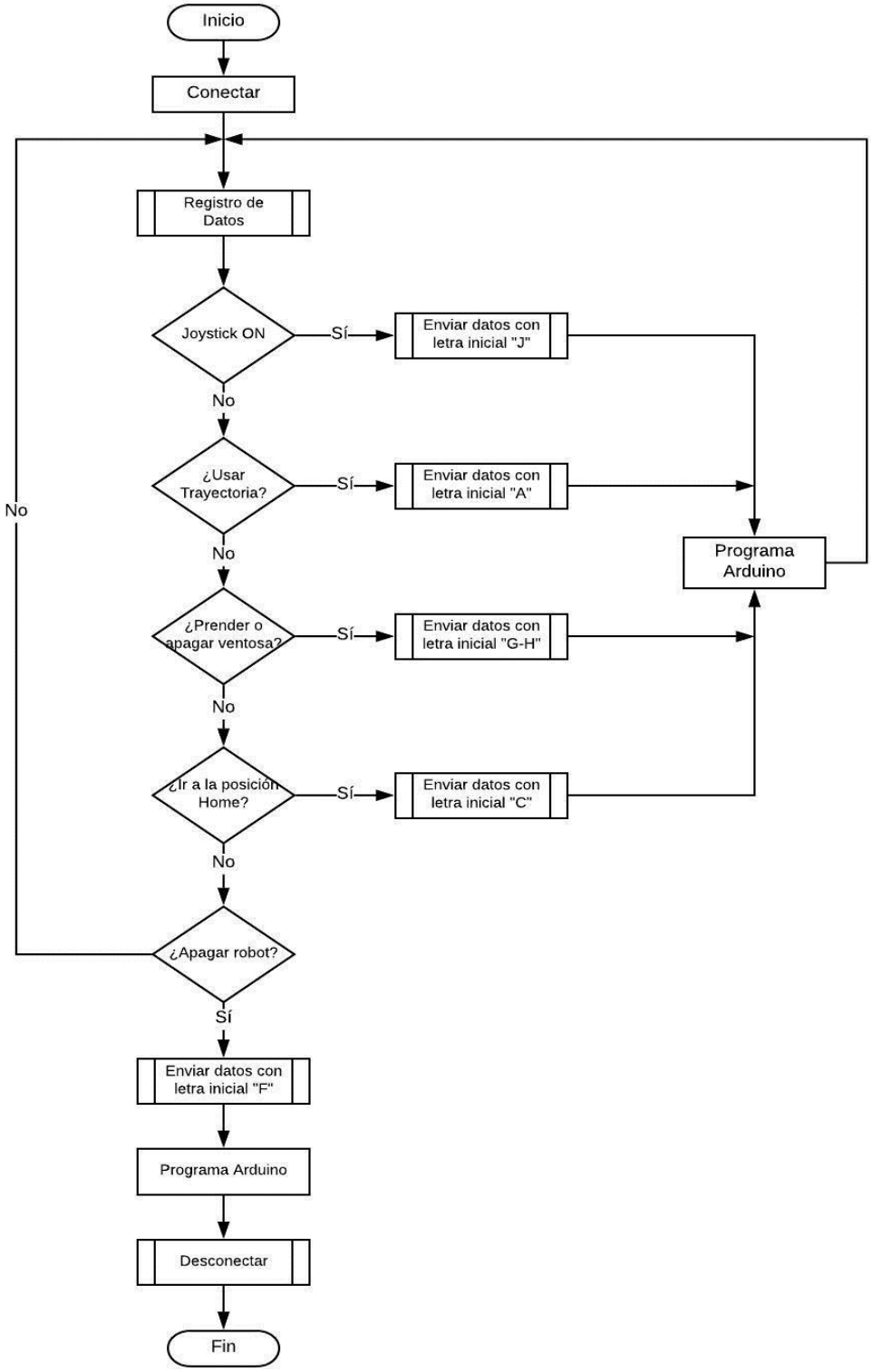


Figura 4.26. Diagrama de flujo del programa principal de la interfaz
Fuente: *Elaboración propia.*

En la Figura 4.26, se encuentra el diagrama de flujo del programa principal que usara la interfaz. Este programa consta de cinco opciones

que definen las tareas que el robot será capaz de ejercer, también muestro un continuo registro de datos indicando que se mostraran en todo momento que se use el robot; con el fin de transmitir las tareas a realizar por el robot, este envía un carácter en especial al inicio de la cadena de caracteres, de esta manera se espera que el controlador que reciba la información sepa correctamente cual es la función a realizar. Asimismo, se observa dos subrutinas siendo la primera la de conexión y la segunda el programa Arduino.

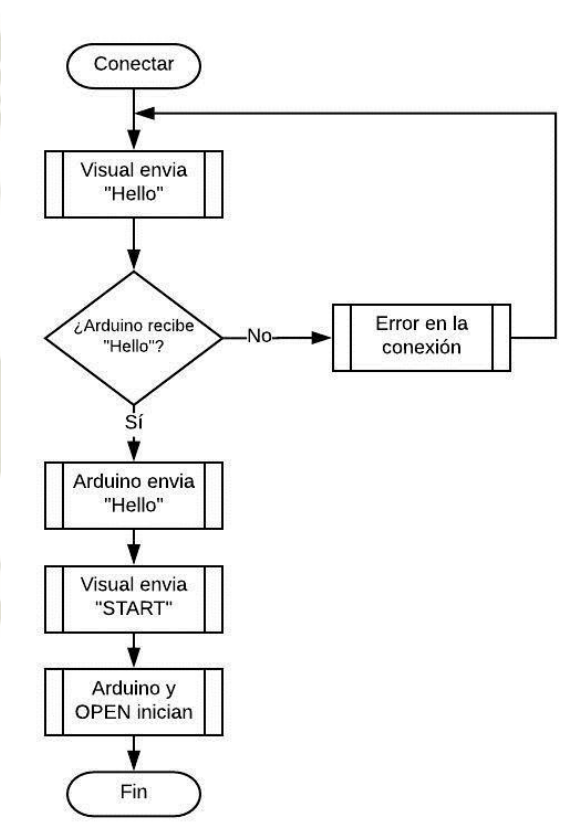


Figura 4.27. Diagrama de flujo subrutina Conectar
Fuente: *Elaboración propia.*

En la Figura 4.27, se detalla el proceso de conexión y comunicación entre los controladores y la interfaz; se aprecia que para que exista una comunicación tiene que haber una serie de confirmaciones, de no ser así se espera que la interfaz mande un aviso de error en la conexión.

La siguiente subrutina que se ve en la Figura 4.26, se refiere al programa del controlador Arduino. En este programa consiste de más subrutinas que contiene programas independientes sobre como manipular al robot, tal como se observa en la Figura 4.28.

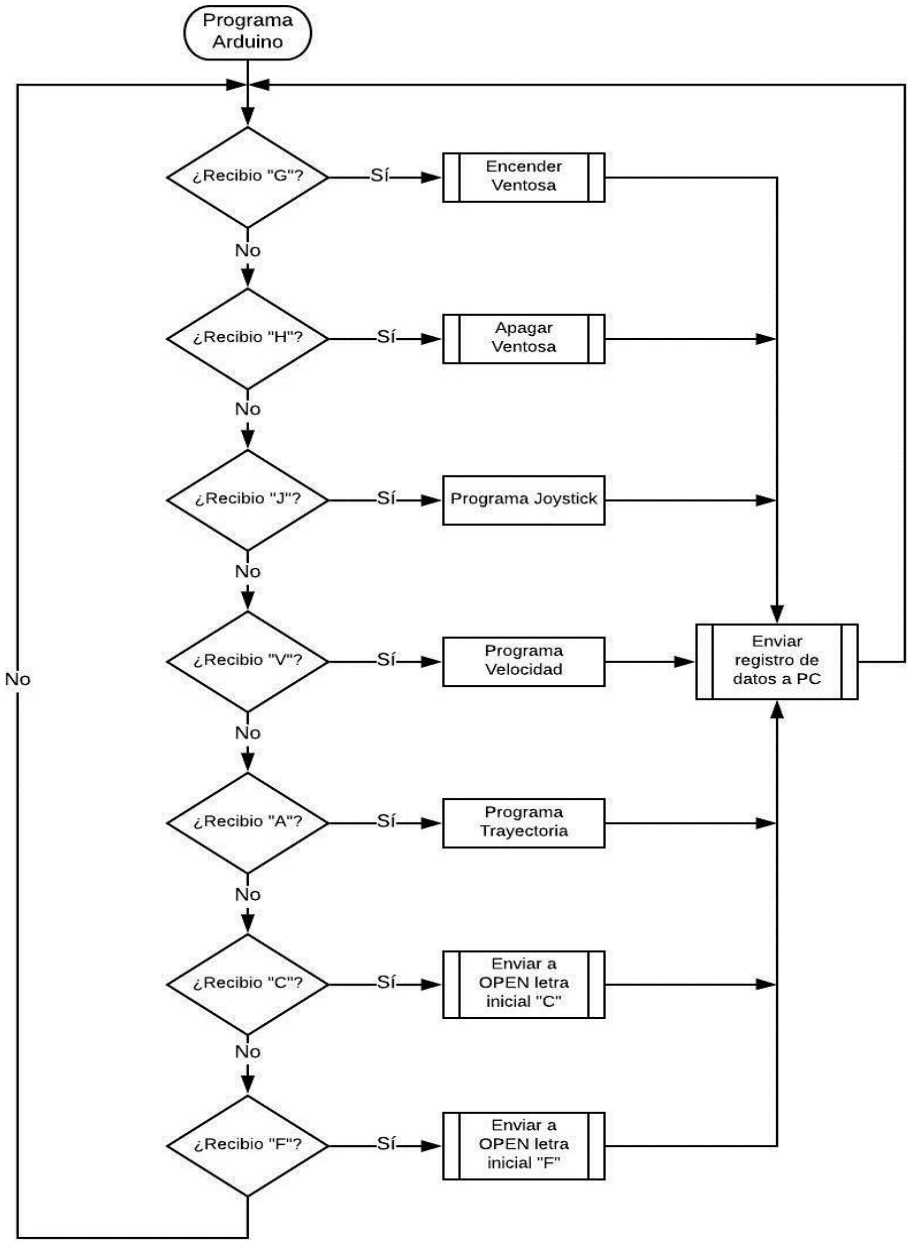


Figura 4.28. Diagrama de flujo subrutina Programa Arduino
Fuente: *Elaboración propia.*

En la Figura 4.28 se observa que el programa se encuentra en un bucle constante donde se esperar recibir el carácter inicial que define la tarea

específica a realizar y también el constante envió de los datos obtenidos del actuador hacia la PC. Este programa consta de procesos predefinidos donde las acciones a ejecutar son simples y subrutinas con programas más complejos para desarrollar la tarea. En este se observa tres subrutinas las cuales son, el programa del joystick, el programa de la trayectoria y el programa de la velocidad.

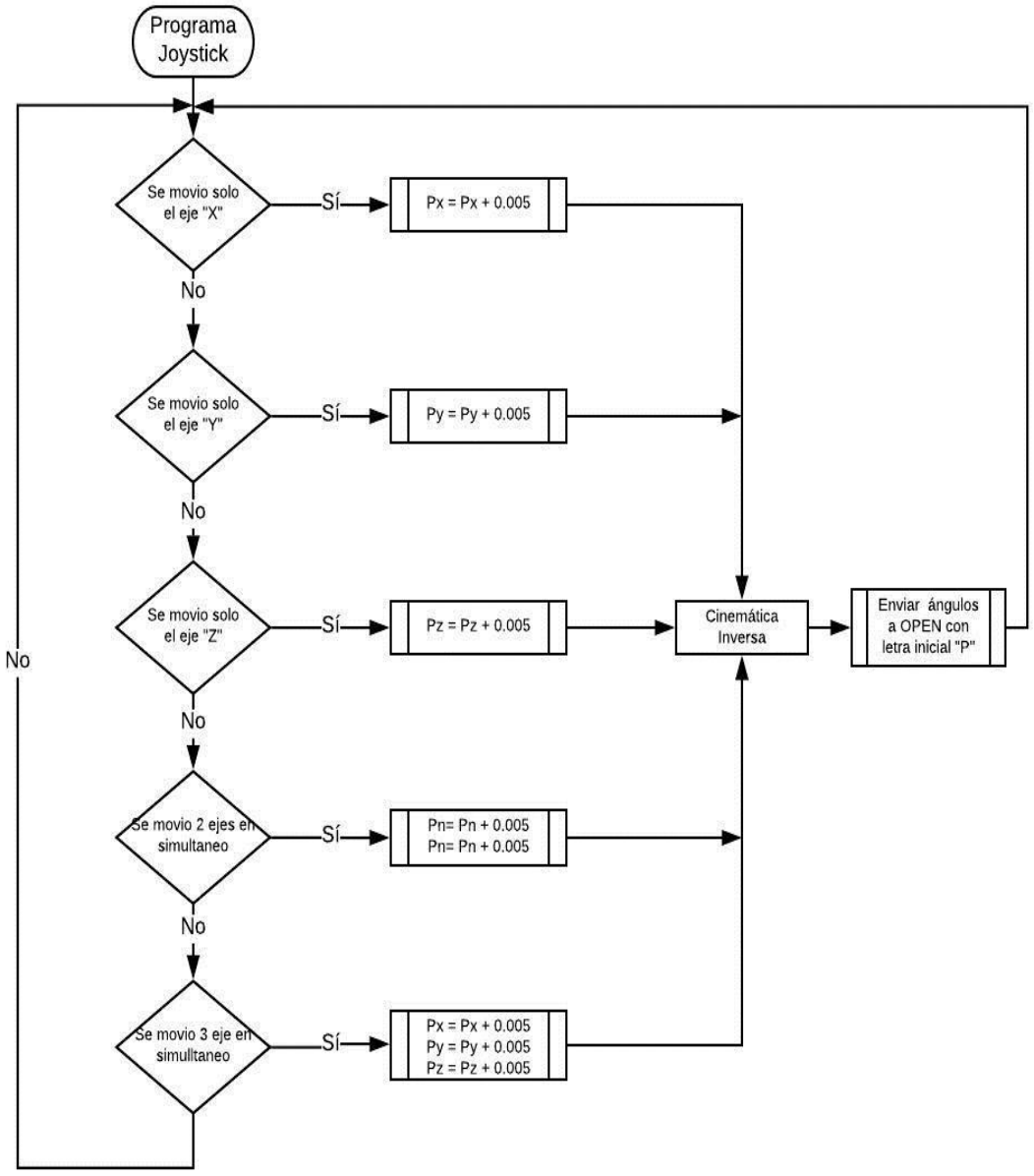


Figura 4.29. Diagrama de flujo subrutina Programa Joystick
Fuente: *Elaboración propia.*

En la Figura 4.29 se observa el programa que ejecutara el movimiento del robot a través de un joystick. En este se tiene 5 opciones que indican las posibilidades del movimiento cartesiano, si es que alguno se mueve se obtiene un incremento o decremento de 0.005m. Luego este dato pasa a una subrutina de cinemática inversa para ser finalmente enviadas al controlador esclavo como ángulos a alcanzar por los actuadores.

Respecto al número 0.005m, este es considerado como un avance en cada bucle, donde se espera que incremente o decrezca 5 milímetros cada periodo de tiempo. Dicho periodo de tiempo tiene que ser lo suficientemente constante para que el movimiento del robot evolucione con el tiempo de manera fluida sin que el robot vaya muy lento o muy rápido.

En relación a la subrutina del programa que regula la velocidad del robot, solo deberá funcionar cuando el robot este siendo manipulado por el joystick, de esta manera se puede tener un control completo al manipular manualmente el robot paralelo delta.

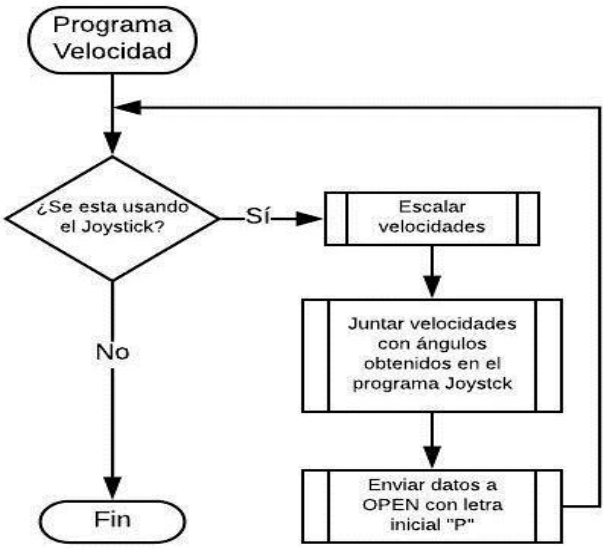


Figura 4.30. Diagrama de flujo subrutina Programa Velocidad
Fuente: *Elaboración propia.*

En la Figura 4.30 se encuentra el diagrama de flujo del programa de la velocidad, esta consta de una sola opción en la que indica que si se está usando el programa Joystick puede funcionar el programa de velocidad. Después de ya confirmada la decisión, las velocidades establecidas por la interfaz son enviadas hacia el controlador Arduino y luego se juntan en la misma cadena de caracteres que el programa Joystick, esto se hace con el fin de asegurar que solo se envíen cuando se usa el joystick. En consecuencia, la cadena de caracteres que se envía con la letra inicial “P” al controlador esclavo contiene datos de posición angular y velocidad que serán imprimidas hacia los actuadores.

Si bien es cierto que la velocidad es un factor que influye en la trayectoria generada del robot, el programa velocidad no debe formar parte de esta puesto que es una velocidad que el usuario decide a través de la interfaz. Sin embargo, una trayectoria implica una posición y velocidad pre-calculada en cada instante de tiempo que el robot se mueva. A continuación, se muestra el programa de la trayectoria y en esta se puede observar el momento en el que se calcula las velocidades.

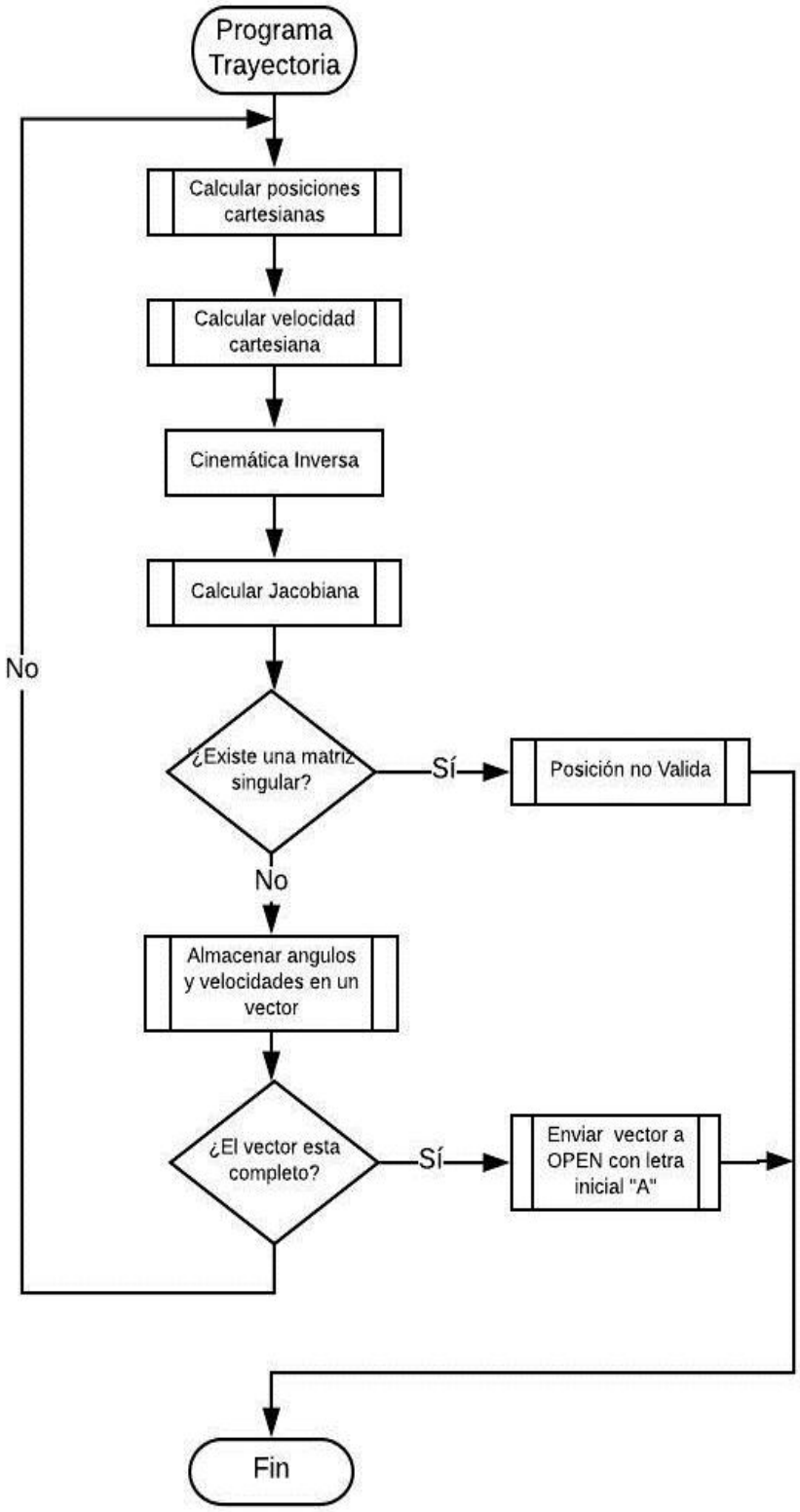


Figura 4.31. Diagrama de flujo subrutina Programa Trayectoria

Fuente: *Elaboración propia.*

De la Figura 4.31 se observa el diagrama de flujo del programa de la trayectoria, en la cual calcula posiciones cartesianas dependiendo de las

ingresadas en la interfaz. El funcionamiento de este programa es de cierta manera el más complejo, puesto que la trayectoria planteada es una trayectoria cartesiana, la cual involucra conocer sus posiciones en los ejes X-Y-Z en todo momento. Lograr esto requiere uso de la cinemática directa; no obstante, es sabido que la cinemática directa del robot paralelo delta es compleja y posee un alto coste computacional. Calcular la posición cartesiana en “n” instantes de tiempo resulta un problema muy grande para el controlador debido a su no linealidad y aun; si esta es calculada, el tiempo en el que lo haría sería relativamente grande. Estas razones hacen que el uso de la cinemática directa no sea considerado como una alternativa.

Con el objetivo de conocer las posiciones cartesianas en cada instante de tiempo, estas serán pre-calculadas, mediante una trayectoria predefinida. Una trayectoria que tenga el mismo comportamiento, pero a diferentes posiciones, de esta manera seria posible saber las posiciones cartesianas, velocidades cartesianas y así luego lograr conocer las velocidades angulares mediante el uso de la Jacobiana inversa.

Si bien es cierto que ya con los procedimientos anteriormente mencionados se obtiene la Jacobiana, es necesario verificar si es que existe una singularidad. Recordando los conceptos de singularidad del robot paralelo delta, al ser una cadena cinemática cerrada la Jacobiana depende de las posiciones cartesianas y posiciones angulares en “n” instante de tiempo, es por esto que calcular sus singularidades era sumamente complejo. Esta situación implica comprobar la no

singularidad de la matriz jacobiana, tal como se observa en la Figura 4.31, de existir una singularidad la trayectoria no será ejecutada; en caso no exista dicha singularidad, todos los parámetros de posición angular y velocidad angular serán guardados en un vector el cual al completarse será enviado al controlador esclavo.

Vinculando los diagramas de flujos presentados en la que su función consistía en mover el robot, se incluyó la subrutina de la cinemática inversa. Esta subrutina, se encarga de resolver el problema cinemático mediante operaciones matemáticas y asegurarse que exista una solución, puesto que en la solución del problema cinemático inverso se planteó que debe de cumplir requisitos para que exista una solución

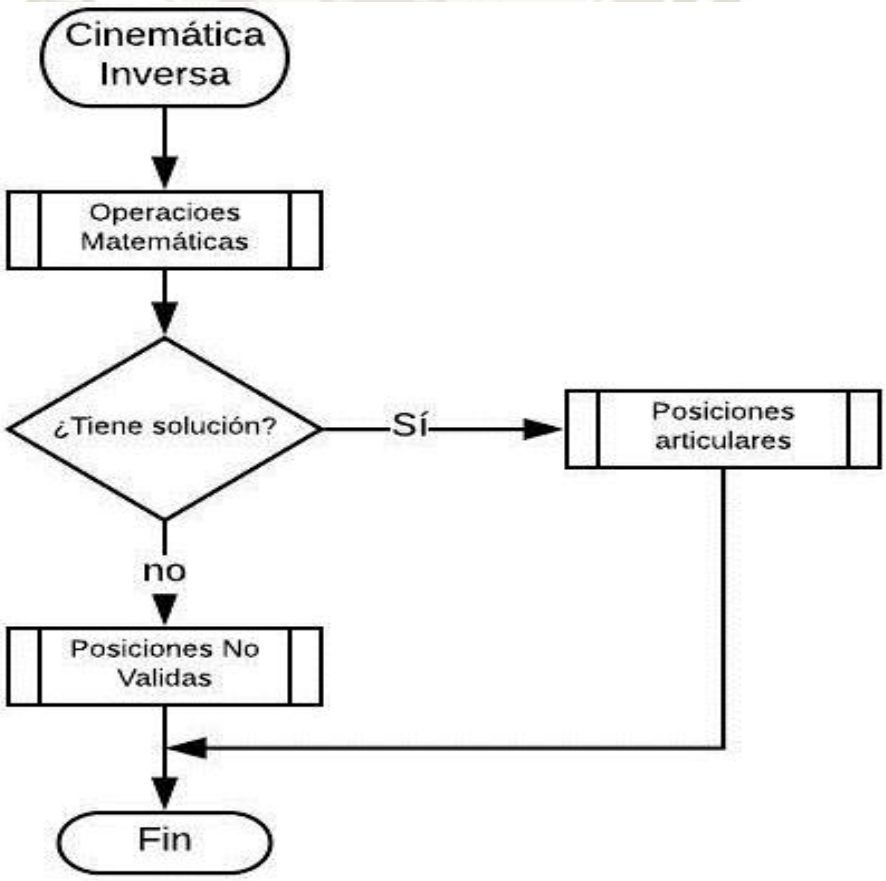


Figura 4.32. Diagrama de flujo subrutina Cinemática Inversa
Fuente: *Elaboración propia.*

Habiendo detallado los diagramas de flujo de la interfaz y el controlador Arduino, es turno del programa del controlador esclavo, OpenCM9.04.

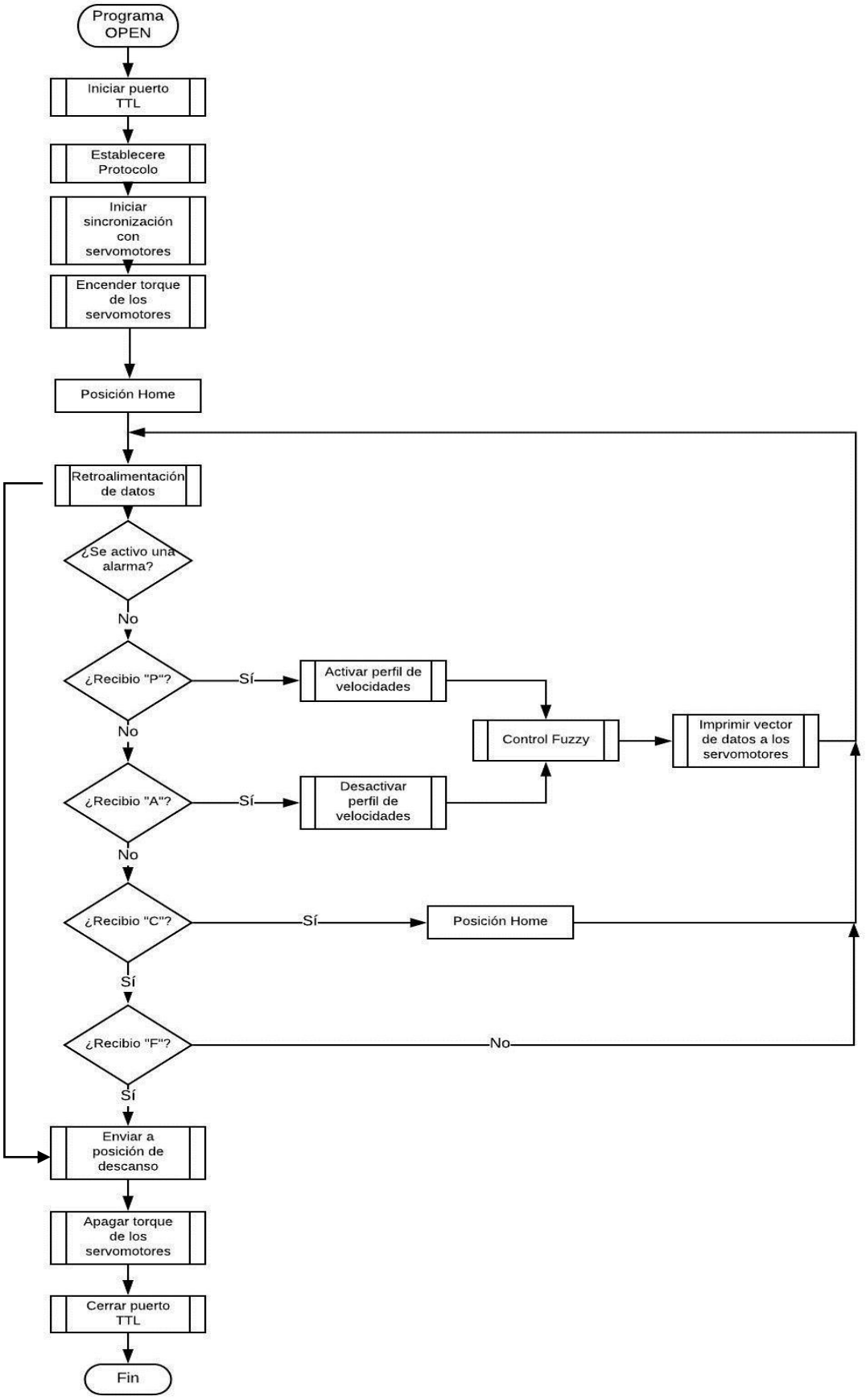


Figura 4.33. Diagrama de flujo Programa OpenCM9.04
Fuente: *Elaboración propia.*

En la Figura 4.33 se encuentra el diagrama de flujo que define el funcionamiento del controlador OpenCM9.04, a diferencia de los otros diagramas ya mencionados, este contiene una serie de procesos necesarios para inicializar. Una vez encendido el controlador, los puertos de comunicación tienen que ser iniciados para luego establecer la versión del protocolo de comunicación a usar; a continuación, se sincroniza la comunicación de todos los servomotores, esto se hace con el fin de que todos los actuadores ejerzan la función designada al mismo tiempo y no uno tras otro. Una vez configurada la sincronización, se enciende el torque de los servomotores y posteriormente se lleva al robot a una posición predefinida; después, se inicia la constante retroalimentación de los datos brindados por el actuador tales como posición, voltaje, velocidad, etc. Luego, se observa una serie de condiciones, la mayoría de estas están relacionadas a los programas dados por el controlador maestro, las cuales son:

- **Condición de alarmas:** Indica que, si alguna alarma del servomotor Dynamixel es activada, el robot deja de funcionar puesto que las torques de los servomotores son llevadas a 0, imposibilitando cualquier movimiento que el robot quiera ejercer.
- **Condición de letra inicial "P":** Esta condición está referida al programa del joystick; en la cual se reciben datos de posición angular y velocidades deseadas por el usuario; con el fin de lograr una manipulación estable donde las inercias creadas por el movimiento del robot sean suaves, se usan perfiles de velocidad dados por el mismo servomotor. Luego, de tener las variables deseadas se

procede con el control Fuzzy y finalmente las variables son enviadas a los actuadores.

- **Condición de letra inicial "A":** Esta condición está referida al programa de la trayectoria visto anteriormente, al igual que la condición anterior se tienen datos de posición y velocidad angular. No obstante, al tener una trayectoria predefinida con velocidades ya calculadas, el uso de los perfiles de velocidad se torna innecesario por lo que son desactivados; a continuación, estas variables pasan por el control Fuzzy para luego ser imprimidas a los servomotores.
- **Condición de letra inicial "C":** En esta condición se busca llevar al robot paralelo delta a una posición preestablecida.
- **Condición de letra inicial "F":** Esta última condición se encarga de apagar el robot cuando se desee; si es que es usada, el robot paralelo delta va a una posición final de descanso; luego, el torque de los servomotores es desactivado y finalmente el puerto TTL es cerrado, logrando así apagar completamente al robot.

Dentro de este marco se observó que la única subrutina dentro del programa del controlador OpenCM9.04 es la de posición Home, teniendo como diagrama de flujo lo siguiente.

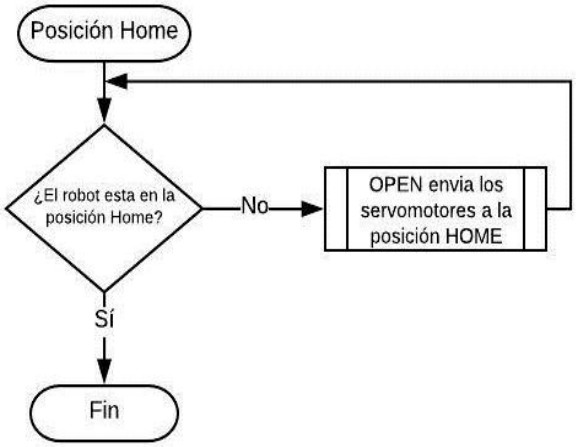


Figura 4.34. Diagrama de flujo subrutina posición home
Fuente: *Elaboración propia.*

Finalmente, en la Figura 4.34 se muestra el diagrama de flujo de la subrutina de la posición preestablecida; esta subrutina se creó con el fin de que cada vez que el robot se encienda o haya terminado una tarea, se mueva a una posición de reposo.

4.3.3. Algoritmo Computacional

En esta sección del apartado se busca plasmar los diagramas de flujo mediante una programación estructurada. Así mismo, solo se detallará las secciones del programa más importante tales como la comunicación entre los dispositivos, el programa de la trayectoria e instrucciones importantes de los controladores.

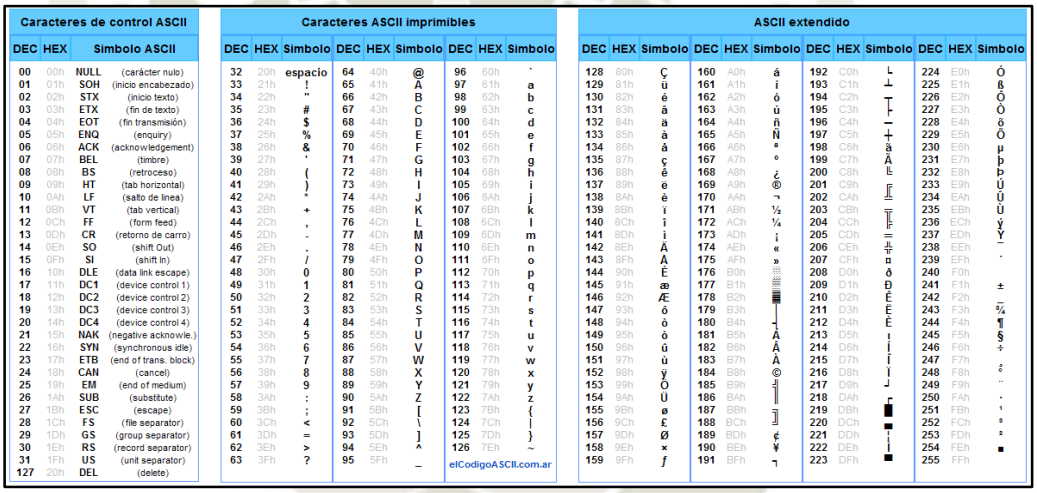
**(Los algoritmos en lenguaje estructurado de los controladores se encuentran en el Anexo 08)*

4.3.3.1. Comunicación entre Controladores e Interfaz

En cuanto a la comunicación serial que existirá entre los controladores y la interfaz, se indicó que el modo de comunicación usado es de Half-Duplex, esto quiere decir que la transmisión y recepción debe turnarse. Así mismo los datos enviados deben de

ser correctamente definidos y enviados, de tal manera que no exista colisión de datos ni problemas en la comunicación.

Por otra parte, los datos enviados tienen un formato ASCII (American Standard Code for Information Interchange); este formato ASCII se basa en un sistema de codificación que asigna a cada carácter alfanumérico (A-Z, a-z, 0-9) o de control un valor entre 0 y 255. De este modo al almacenar un texto o números utilizaremos un byte por carácter más algunos bytes de control.



Caracteres de control ASCII

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de línea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

elCodigoASCII.com.ar

ASCII extendido

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	á	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	é	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	è	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Figura 4.35. Codificación ASCII
Fuente *(El Código ASCII, 2016)*

El problema del formato ASCII surge a la hora de almacenar números, puesto que si se tienen decimales o signos negativos se debe reconocer en que digito se encuentra.

Respecto al entorno de programación del controlador maestro, Arduino Mega, y el controlador esclavo, OpenCM904, son capaces de ser programados bajo el mismo entorno, el cual es Arduino IDE y posee un lenguaje de programación C++.

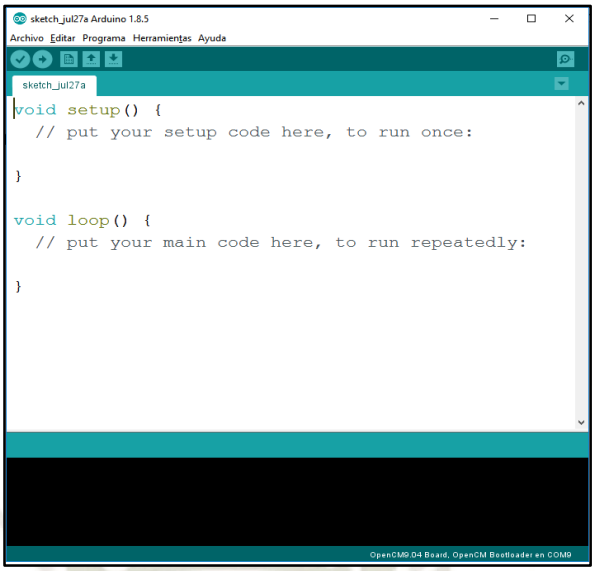


Figura 4.36. Entorno de programación Arduino IDE
Fuente: *Elaboración propia.*

El hecho de que los dos controladores posean el mismo entorno de programación hace posible usar las mismas librerías; Arduino IDE ofrece una librería de comunicación serial la cual facilita la transmisión y recepción de datos mediante diversas funciones. Entre ellas se encuentra la función “available”; el objetivo de esta función es obtener la cantidad de bytes disponibles para leer. Entonces sabiendo esto es posible decir que si la función “available” es mayor que 0; se está recibiendo datos.

Si bien es cierto que ya se sabe cuándo se reciben datos, aún se desconoce cómo reconocer los datos ASCII correctamente. Esta situación hace necesario reconocer el inició de una cadena de caracteres y por lo tanto su final.

Para lograr esto se plantea usar dos caracteres específicos siendo el símbolo menor “<” para el inicio de la data y el símbolo mayor “>” para el fin de la data. De esta manera se analizará byte por byte los

datos recibidos, comparando si es que estos poseen los términos de inicio o fin y los datos que se incluyen dentro de estos dos símbolos.

```
void LecturaDeData()
{
static boolean DataEnProgreso = false;
static byte numbyte = 0;
char iniciador = '<';
char terminador = '>';
char data;

if (Serial.available() > 0)
{
      data = Serial.read();

      if (DataEnProgreso == true)
      {
            if (data != terminador)
            {
                  CarecteresRecibidos[numbyte] = data;
                  numbyte++;
                  if (numbyte >= CantidadCaracteres) { numbyte = CantidadCaracteres - 1; }
            }
            else
            {
                  CarecteresRecibidos[numbyte] = '\0'; // terminate the string
                  DataEnProgreso = false;
                  numbyte = 0;
                  newData = true;
            }
      }

      else if (data == iniciador) { DataEnProgreso = true; }
}
}
```

Figura 4.37. Programación de la subrutina de comunicación
Fuente: *Elaboración propia.*

En la Figura 4.37 se observa un código estructurado que detalla el proceso de comunicación y como se establece la lectura de los paquetes de datos enviados.

Así mismo, al ya obtener un conjunto de datos es necesario convertirlos de formato ASCII a su formato original, para que se visualice y manipule los datos en las funciones necesarias. Para lograr esto, es necesario conocer siempre la cantidad de dígitos que son enviados por la interfaz; con el fin de estructurar y separar correctamente la data.

```
int ConvetirANumero( byte Digito)
{
  unsigned int val = 0;
  val = (CarecteresRecibidos[Digito]-48) * 1000;
  val = val + (CarecteresRecibidos[Digito+1]-48) * 100;
  val = val + (CarecteresRecibidos[Digito+2]-48) * 10;
  val = val + CarecteresRecibidos[Digito+3]-48;
  return val;
}
```

Figura 4.38. Programación de la instrucción para leer caracteres
Fuente: *Elaboración propia.*

De la Figura 4.38 se encuentra el código que define la lectura de los caracteres, en este se observa que se resta 48 unidades a cada byte y luego se multiplica por un factor dependiendo si es una unidad, decena, centena, etc.

4.3.3.2. Trayectoria Cartesiana

El programa trayectoria que se observó anteriormente en los diagramas de flujo, no indica la trayectoria específica a realizar por el robot paralelo delta; pero sí su comportamiento. Ya que es necesario conocer las posiciones y velocidades cartesianas, se plantea una trayectoria constante que sea capaz de simular el recojo de objetos y que no posea cambios bruscos que puedan afectar considerablemente la inercia de los elementos que componen la estructura del robot.

Atendiendo a estas consideraciones, se plantea usar trayectorias de línea recta que posean un punto intermedio, esto con el fin de simular una trayectoria “Pick and Place” y por la simplicidad de las funciones matemáticas que conlleva.

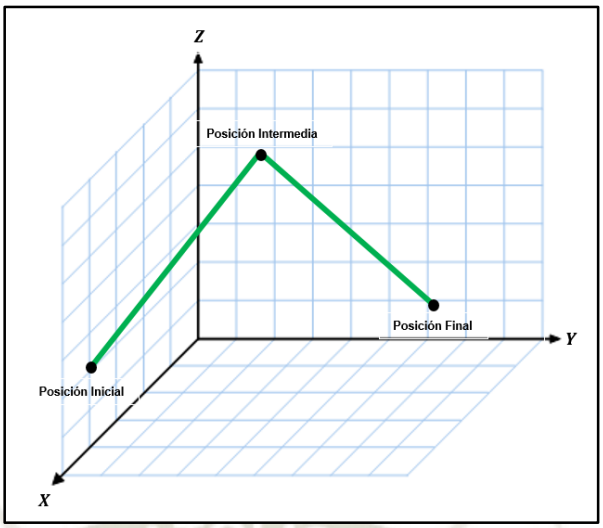


Figura 4.39. Trayectoria preestablecida que ejercerá el robot
Fuente: *Elaboración propia.*

En la Figura 4.39 se observa la trayectoria recta que dará paso al movimiento del robot; en esta se observa el recorrido que ejercerá a lo largo de tres puntos.

A primera instancia el tener un punto intermedio con una pronunciada cresta hace parecer que existirá discontinuidad en la velocidad y aceleración, creando así altas inercias. No obstante, para resolver este problema se tendrá en cuenta que cada punto de paso tendrá velocidad nula, de esta manera es posible regular la inercia.

En cuanto a la función matemática de la trayectoria se tiene lo siguiente:

$$\vec{r} = \vec{r}_o + t \cdot \vec{v} \qquad t \in \mathbb{R} \tag{4.1}$$

Donde:

$\vec{r}$ = Vector de punto arbitrario a través de la recta

$\vec{r}_o$ = Vector inicial de la recta

$\vec{v}$ = Vector dirección

t = Parámetro de la recta

La expresión 4.1 está referida a la ecuación vectorial de la recta, a simple vista es una ecuación simple que indica la posición de cada punto que pasa sobre la recta. Sin embargo, trabajar la ecuación de la recta con una variación lineal del parámetro "t" crea movimientos sin control sobre las velocidades o aceleraciones del robot. Estas razones hacen necesario encontrar una manera en la que el robot se pueda desplazar de manera suave y sin ejercer movimientos bruscos.

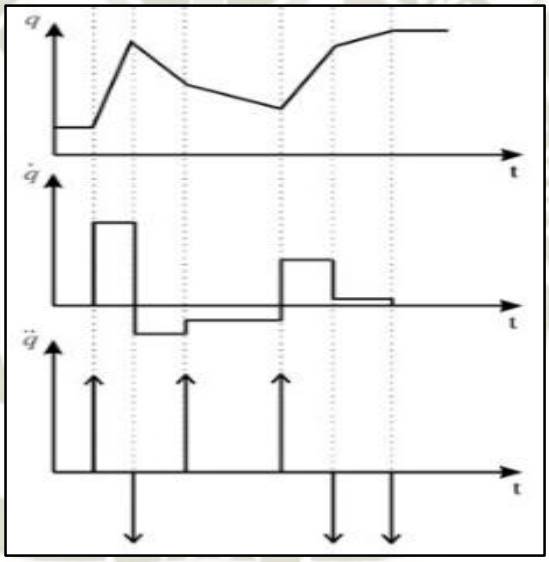


Figura 4.40. Interpolación Lineal
Fuente *(Universidad Tecnológica SLRC, 2015)*

Atendiendo a las consideraciones anteriores, se hará uso de un interpolador quintico para controlar la forma en la que evoluciona el parámetro "t", con la finalidad de obtener movimientos suaves en el espacio cartesiano y el espacio articular.

**(El programa en MATLAB de las trayectorias a usar se encuentra en el Anexo 09)*

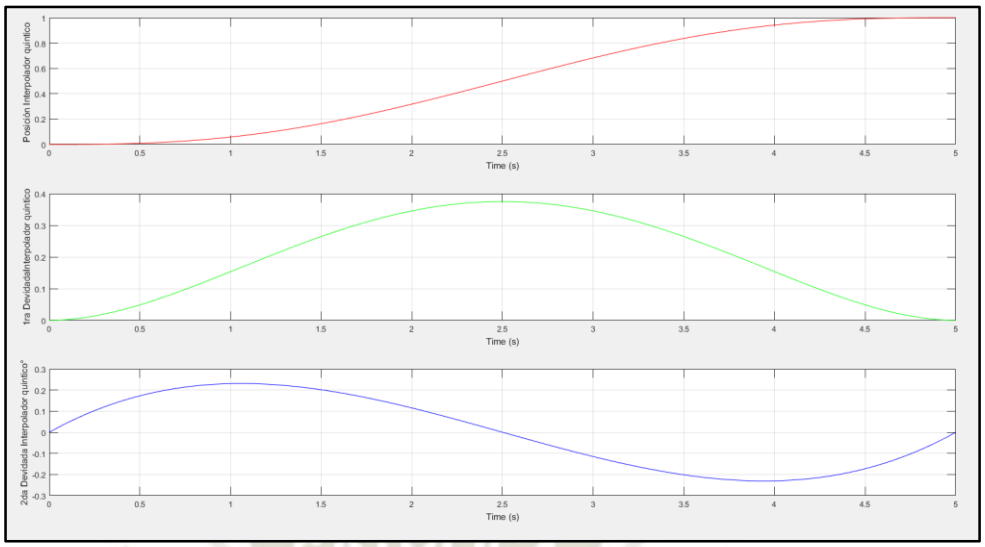


Figura 4.41. Interpolación Quintica del parámetro "t" en 4 segundos
Fuente: *Elaboración propia.*

De la Figura 4.41 se observa que la evolución del parámetro "t" con el interpolador quintico crea movimientos suaves, en consecuencia su primer y segunda derivada no contiene discontinuidades. Sobre la base del concepto expuesto, es posible generar la trayectoria planteada con movimientos suaves sin que creen se creen inercias altas en los componentes del robot; esto se demuestra en las siguientes figuras.

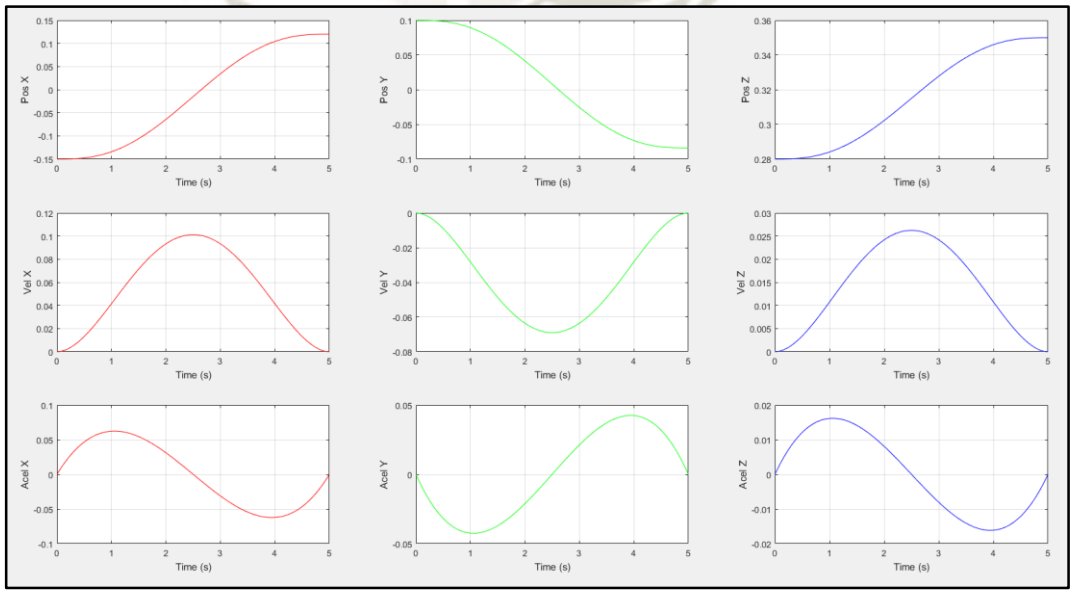


Figura 4.42. Posiciones, velocidades y aceleraciones cartesianas usando un interpolador quintico
Fuente: *Elaboración propia.*

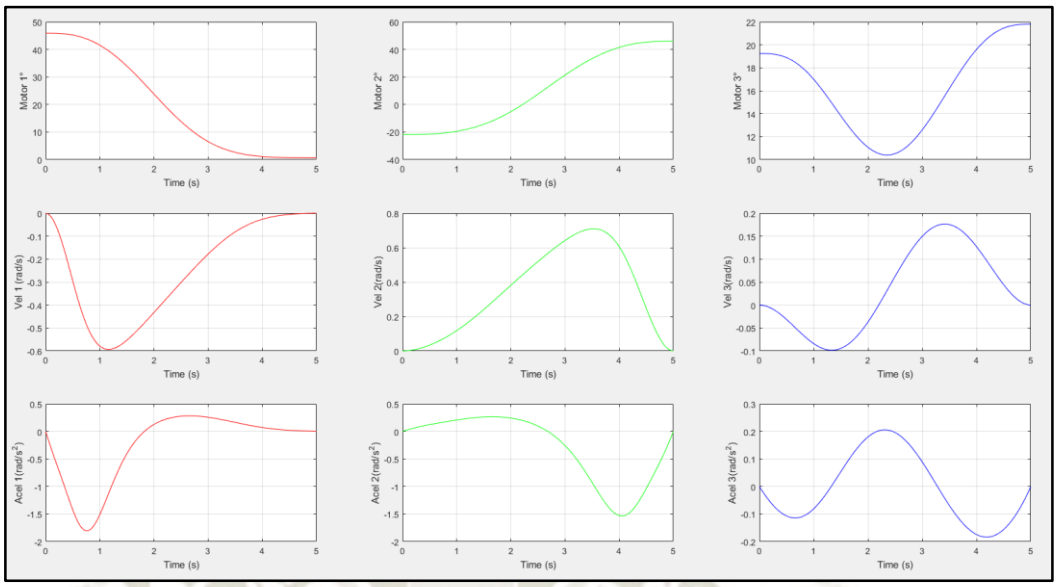


Figura 4.43. Posiciones, velocidades y aceleraciones angulares en base al interpolador quintico
Fuente: *Elaboración propia.*

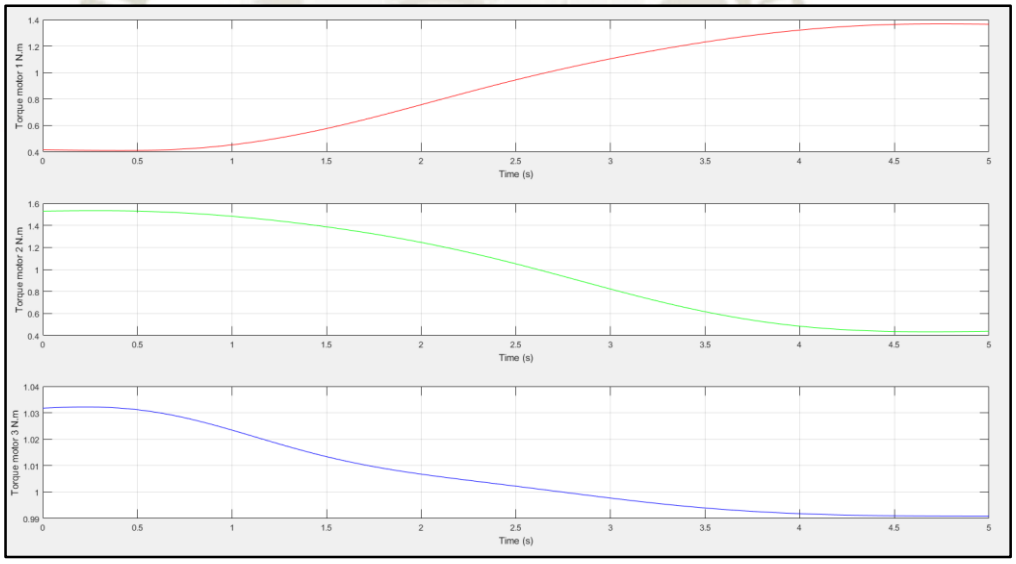


Figura 4.44. Torque de cada articulación obtenido en base al interpolador quintico
Fuente: *Elaboración propia.*

Las figuras anteriormente expuestas reflejan el comportamiento que se obtiene en la posición, velocidad y aceleración al usar un interpolador quintico; se observa continuidad en el espacio cartesiano y en el espacio angular. La Figura 4.44 demuestra que no se presentan cambios bruscos de torque en ninguna de las tres

articulaciones; así mismo, al comparar estas evidencias se resalta la efectividad del interpolador quintico.

Hasta el presente se ha analizado la trayectoria que ejercerá el robot paralelo delta, mas no el funcionamiento de esa plasmada en un lenguaje estructurado. Para lograr integrar la generación de la trayectoria planteada en el programa del controlador maestro, es necesario crear una función que calcule las posiciones y velocidades cartesianas constantemente en base a la posición ingresada por la interfaz. Respecto a las posiciones intermedia y final, estas serán variables constantes puesto que el robot siempre debe ir a una posición dada por la interfaz y llegar a una posición conocida. Finalmente, con el fin de evitar recalcular constantemente las variables del interpolador quintico en el parámetro "t", este será guardado en un vector, obteniendo así lo siguiente.

```
double MU[41]={0,0.0002,0.0012,0.0038,0.0086,0.0161,0.0266,0.0405,0.0579,0.0789,0.1035,0.1316,// Vector de posición Quintico
               0.1631,0.1977,0.2352,0.2752,0.3174,0.3615,0.4069,0.4532,0.5000,0.5468,0.5931,
               0.6385,0.6826,0.7248,0.7648,0.8023,0.8369,0.8684,0.8965,0.9211,0.9421,0.9595,
               0.9734,0.9839,0.9914,0.9962,0.9988,0.9998,1};
double MUD[41]={0,0.0089,0.0338,0.0722,0.1215,0.1794,0.2438,0.3127,0.3840,0.4561,0.5273,0.5963,// Derivada del interpolador Quintico
                0.6615,0.7219,0.7763,0.8240,0.8640,0.8958,0.9188,0.9328,0.9375,0.9328,0.9188,
                0.8958,0.8640,0.8240,0.7763,0.7219,0.6615,0.5963,0.5273,0.4561,0.3840,0.3127,
                0.2438,0.1794,0.1215,0.0722,0.0338,0.0089,0};

  if  ( CarecteresRecibidos[0] == 'A' )

      {
       float pxf = ConvetirANumero( 1 ); //PosX Deseada
       float pyf = ConvetirANumero( 5 ); //PosY Deseada
       float pzf = ConvetirANumero( 9 ); //PosZ Deseada
       pxf = pxf/1000; //Convertir a metros
       pyf = pyf/1000; //Convertir a metros
       pzf = pzf/1000; //Convertir a metros
       pxi =0;         //Posición X intermedia
       pyi =0;         //Posición Y intermedia
       pzi =.3;        //Posición Z intermedia

       vdx = pxf-pxi;vdy = pyf-pyi;vdz = pzf-pzi; //Vector dirección
       }
     {
     for (TT=0;TT<41;TT++)
                     {
                       Px = pxi + (MU[TT]*vdx);
                       Py = pyi + (MU[TT]*vdy);
                       Pz = pzi + (MU[TT]*vdz);
                       Vx = (MUD[TT]*vdx);
                       Vy = (MUD[TT]*vdy);
                       Vz = (MUD[TT]*vdz);

                       Vel[0][0]={{Vx}};
                       Vel[1][0]={{Vy}};
                       Vel[2][0]={{Vz}};}
```

Figura 4.45. Programación de la instrucción para crear trayectoria cartesiana
Fuente: *Elaboración propia.*

De la Figura 4.45 se observa que el programa cuenta con dos vectores predefinidos, estos representan al interpolador quintico y su derivada respectivamente a lo largo de un movimiento de 2 segundos; luego, se hace uso de la subrutina de la comunicación para leer los datos de posición deseada y finalmente ejecutar la trayectoria designada.

4.3.3.3. Instrucciones del Servomotor

El uso de los servomotores Dynamixel implica conocer el funcionamiento del protocolo de comunicación y las instrucciones de control, por ende, en esta sección del apartado se explicarán cómo es que funciona la comunicación y ejecución de las instrucciones creadas por el controlador OpenCM9.04.

Con respecto a la comunicación los servomotores Dynamixel poseen un enlace de comunicación TTL de modo Half-Duplex asíncrono; cada vez que estos son encendidos se espera una señal de inicio dada por el controlador; luego se estable el protocolo a usar; finalmente, se abre el puerto y se establece la velocidad de comunicación (baudrate).

```
dynamixel::PortHandler *portHandler = dynamixel::PortHandler::getPortHandler(DEVICENAME); // Initialize PortHandler instance
dynamixel::PacketHandler *packetHandler = dynamixel::PacketHandler::getPacketHandler(PROTOCOL_VERSION);  // Set the protocol version
// Open port
if (portHandler->openPort())
{
  Serial.print("Succeeded to open the port!\n");
}
else
{
  Serial.print("Failed to open the port!\n");
  Serial.print("Press any key to terminate...\n");
  return;
}

// Set port baudrate
if (portHandler->setBaudRate(BAUDRATE))
{
  Serial.print("Succeeded to change the baudrate!\n");
}
else
{
  Serial.print("Failed to change the baudrate!\n");
  Serial.print("Press any key to terminate...\n");
  return;
}
```

Figura 4.46. Programa para establecer comunicación con los servomotores
Fuente: *Elaboración Propia.*

Entorno a las instrucciones que darán paso a las funciones del servomotor, Robotis establece el uso de un protocolo que contiene un paquete de instrucciones; siendo la información que el controlador principal envía al dispositivo.

Tabla 4.2. Estructura del paquete de instrucciones

Header			Reserved	Packet ID	Packet Length		Instruction	Parameter			16bit CRC	
0xFF	0xFF	0xFD	0x00	ID	LEN_L	LEN_H	Instruction	Parameter1	...	ParameterN	CRC_L	CRC_H

Fuente: (Robotis, 2010)

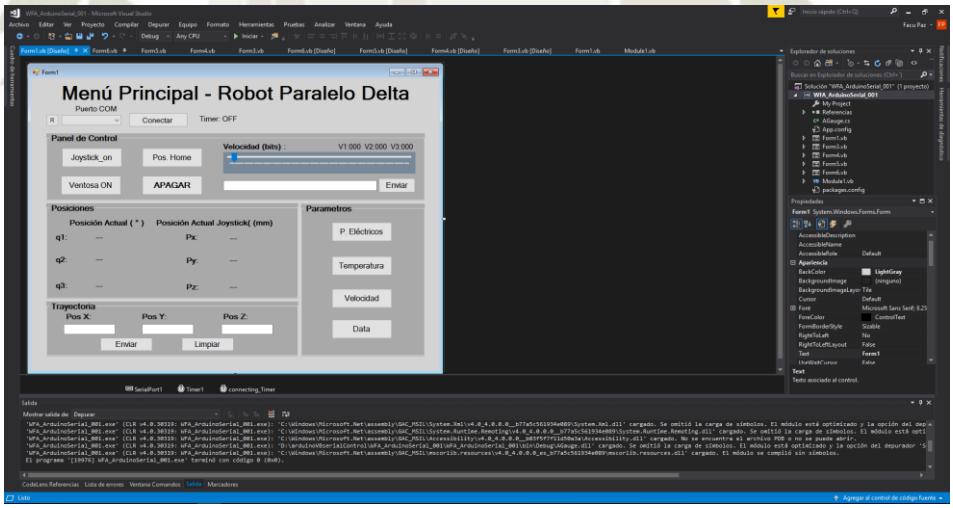
En la Tabla 4.2 se encuentre la estructura que compone el paquete de instrucciones enviado por el controlador, que cuenta con las siguientes partes:

- **Header:** Es el campo que indica el inicio del paquete.
- **Reserved:** Campo reservado.
- **Packet ID:** Este es el campo que indica la ID del dispositivo que debe recibir el paquete de instrucciones y procesarlo.
- **Packet Length:** La longitud del paquete después del Packet Length.

- **Instruction:** Este es el campo que define el propósito del paquete.
- **Parameter:** Campo de datos auxiliares.
- **16 Bit CRC:** Este es el campo que verifica si el paquete se ha dañado durante la comunicación.

4.3.4. Diseño y Desarrollo de la Interfaz Gráfica

El diseño y el desarrollo de la interfaz del robot paralelo delta, fueron programados en el software Microsoft Visual Studio 2017 usando el entorno de Visual Basic, se decidió usar este programa, porque posee un software libre y un entorno de programación amigable, aparte permite un acceso sin restricciones a las librerías que ofreceré la plataforma de sistema Windows.



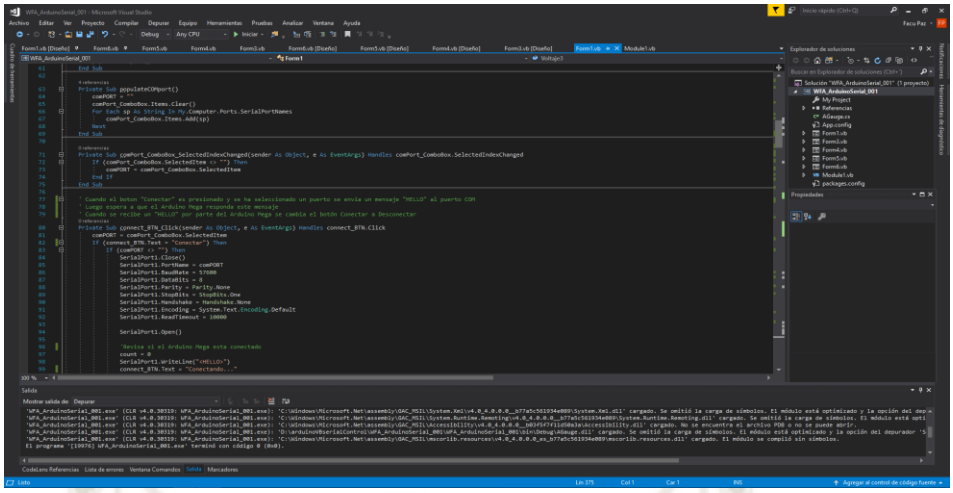


Figura 4.47. Diseño y programación de la interfaz gráfica
Fuente: *Elaboración Propia.*

En la Figura 4.47 se observa la ventana principal de la interfaz y parte de la programación necesaria para llevar su funcionamiento acabo. A continuación, se detallará el desarrollo y funcionamiento de la interfaz gráfica; para entrar al programa basta con ingresar desde el escritorio y darle clic al icono del programa.

Figura 4.48. Icono del Programa INTERFAZ Robot Paralelo Delta
Fuente: *Elaboración Propia.*

El diseño de la interfaz está compuesto por una ventana principal y cuatro ventanas segundaria, siendo estas las ventanas de los parámetros de los actuadores.

4.3.4.1. Ventana de Menú Principal

Esta ventana es la primera que aparecerá apenas sea iniciado el programa y consta de cinco partes las cuales son: comando de

conexión, barra de panel de control, barra de posiciones, barra de parámetros y barra de trayectoria; en esta ventana el usuario podrá: Establecer conexión con el controlador conectado a partir de la elección del puerto COM a la cual el dispositivo se encuentre conectado; de no encontrar un dispositivo el usuario puede resetear la búsqueda hasta que se encuentre algún dispositivo conectado.

- Entablar conexión con el joystick para luego usarlo como medio de manipulación con el robot paralelo delta.
- Visualizar la posición angular de los actuadores en todo momento y la posición cartesiana del robot paralelo delta cuando se esté haciendo uso del joystick.
- Cambiar la velocidad, esta acción solo está permitida cuando el usuario este haciendo uso del joystick, de esta manera podrá tener pleno control sobre los movimientos del robot.
- Encender y apagar la herramienta, el usuario podrá manipular en todo momento la ventosa que se encuentra en la base móvil del robot.
- Enviar al robot paralelo delta a una posición común; también, apagar el robot cuando el usuario lo desee.
- Ingresar posiciones cartesianas manualmente para luego hacer uso del programa trayectoria.
- En caso ocurra un problema el usuario puede enviar manualmente la instrucción al controlador ingresando la instrucción.

- Ingresar a los parámetros de electricidad, temperatura, velocidad y data que constantemente envían los actuadores como información.

  (Para ver la programación de la interfaz referirse al Anexo 10)

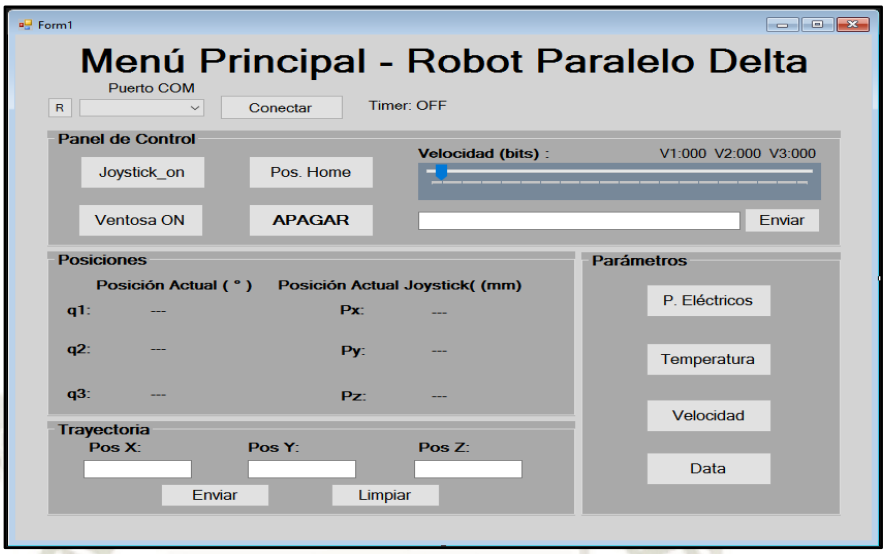


Figura 4.49. Ventana de Menú Principal
Fuente: *Elaboración Propia.*

4.3.4.2. Ventana de Parámetros Eléctricos

Esta ventana tiene como objetivo, visualizar los parámetros de voltaje, corriente y potencia dados por los servomotores Dynamixel. Cabe recalcar que los actuadores no proveen el dato de la potencia; no obstante, este parámetro fue obtenido indirectamente.

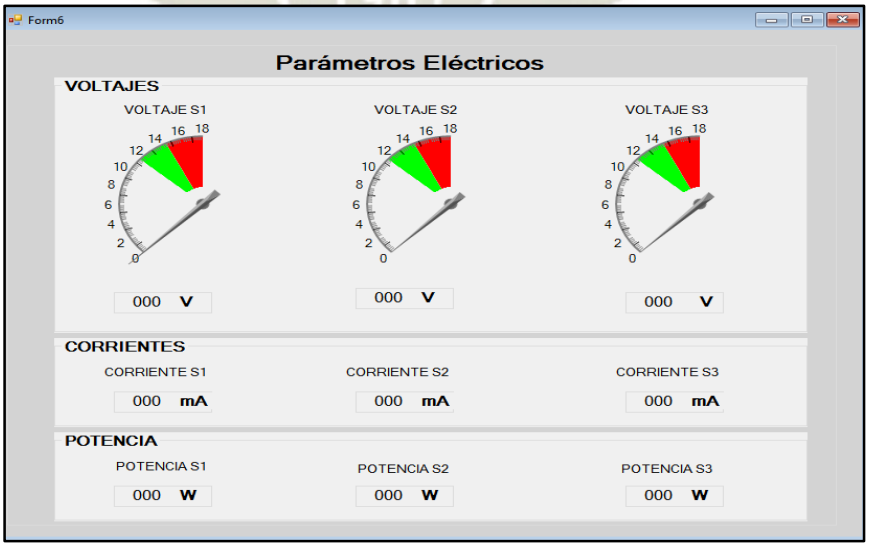


Figura 4.50. Ventana de Parámetros Eléctricos
Fuente: *Elaboración Propia.*

4.3.4.3. Ventana de Temperatura

En esta ventana se observa, la temperatura actual de los servomotores Dynamixel. Así mismo, esta ventana cuenta con medidores de distintos rangos de colores para avisar al usuario las altas temperaturas que puede llegar a alcanzar el actuador.

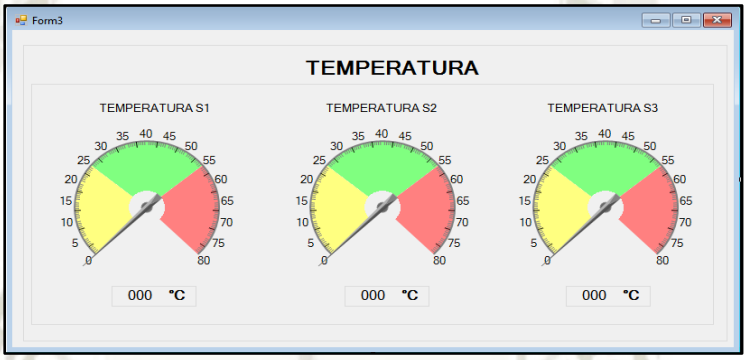


Figura 4.51. Venta de Temperatura
Fuente: *Elaboración Propia.*

4.3.4.4. Ventana de Velocidades

En esta ventana se visualiza la velocidad actual en rad/segundo de los 3 servomotores Dynamixel.

Figura 4.52. Ventana de Velocidades
Fuente: *Elaboración Propia*

4.3.4.5. Ventana de Data

En esta ventana el usuario podrá visualizar los paquetes de datos enviados y recibidos por el controlador Arduino Mega; así mismo, dicho paquete de datos contiene información sobre los parámetros de los servomotores Dynamixel.

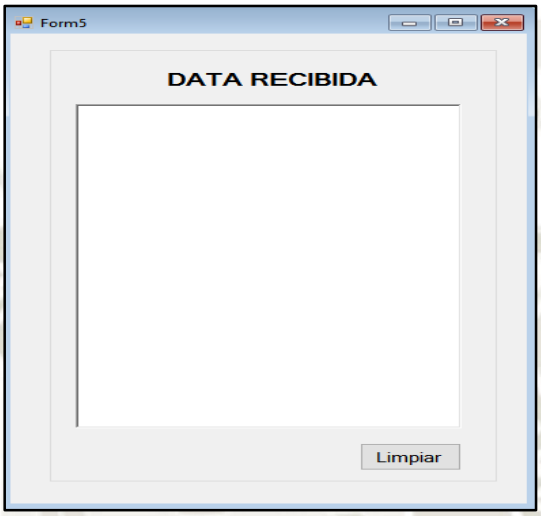


Figura 4.53. Ventana de Data Recibida
Fuente: *Elaboración Propia.*

Capítulo V

5. DISEÑO DEL CONTROLADOR DIFUSO

La etapa más compleja en el diseño de un manipulador robótico es el control del robot, ya que la dificultad radica en la configuración del robot y el número de grados de libertad que pueda tener. En la actualidad la estrategia de control más usada en el ámbito de la robótica es el control convencional PID con torque computado, debido a su gran robustez y a la gran allegada que tiene. No obstante, este tipo de control requiere de un modelo dinámico que se asemeje lo más posible a un escenario real; aun cuando la práctica demuestra que el modelo dinámico de un robot resulta no lineal, multivariable y de parámetros variables.

En el caso de robot paralelo delta; el tener una configuración de cadena cinemática cerrada implica un comportamiento dinámico fuertemente no lineal. En consecuencia, llevar a cabo una estrategia de control que dependa del modelo dinámico aun sabiendo todas las dificultades que acarrea, demanda mayor complejidad y eleva el coste computacional. Estas son las razones que hacen necesario plantear una estrategia de control que pueda sobreponerse a las limitaciones de un control convencional y al mismo tiempo se puedan obtener resultados óptimos.

El presente proyecto de tesis propone el uso de una estrategia de control basado en la lógica difusa, de esta manera se reemplaza el control dinámico del robot paralelo delta por el conocimiento de un experto, en este caso los autores de la tesis. Así mismo, se espera reducir la complejidad del control evitando el cálculo constante de las variables del modelo dinámico y al mismo tiempo disminuir el coste computacional.

Este capítulo se divide en cuatro partes, en la primera parte se presentara algunos sistemas de control difuso que pueden ser implementados en el robot paralelo delta; la segunda parte está referida a los conjunto difusos de entrada y salida, funciones de membresía y reglas de inferencia; la tercera parte está referida a la implementación del control difuso en el controlador del robot paralelo delta; finalmente, la cuarta parte está referida a la evaluación y selección de los distintos controles difusos propuestos.

5.1. Sistemas de Control Difuso

En el control difuso las variables de entrada y salida se procesan de forma cualitativa en vez de forma cuantitativa. En consecuencia, la acción de control debe medirse en términos de muy pequeño, pequeño, grande, demasiado grande, etc. De esta manera, un control difuso emula el proceso de inferencia humana con reglas lingüísticas como acciones de control.

Sobre la base de las ideas expuestas, es necesario conocer sobre que variables se trabajaran para así crear un sistema de control difuso que se adecue mejor al robot paralelo delta; para esto nos partiremos entorno al control convencional del robot, el cual está basado en el control del error y usa estrategias de control como el PI o el PID.

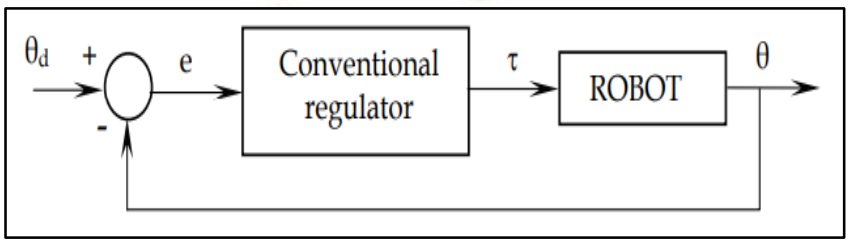


Figura 5.1. Sistema de control robótico convencional
Fuente: *(Robot Control by Fuzzy Logic, 2008)*

De la Figura 5.1 se observa que la entrada del sistema de control está referida a la posición articular deseada y la salida a la posición articular real; de esto también se obtiene que:

$$e(t) = \theta_d(t) - \theta(t) \quad (5.1)$$

Donde:

$e(t)$ = Error en el instante de tiempo

$\theta_d(t)$ = Posición articular deseada en función del tiempo

$\theta(t)$ = Posición articular real en función del tiempo

La estrategia de control de la Figura 5.1 se basa en determinar el torque necesario para que así el error converja a cero.

$$e_s = \lim_{t\to\infty} e(t) = 0 \quad (5.2)$$

De la expresión (5.2), se concluye que en el control convencional está basado en eliminar o disminuir considerablemente el error luego de una serie de procesos interno que conllevan tiempo y calculo. No obstante, se trabaja entorno del error y su variación respecto al tiempo, tomando en cuenta su derivada y su integral dependiendo del tipo de control. En ese sentido, el control difuso a plantear para el robot paralelo delta debe estar referido al error de la posición articular o posición cartesiana deseada.

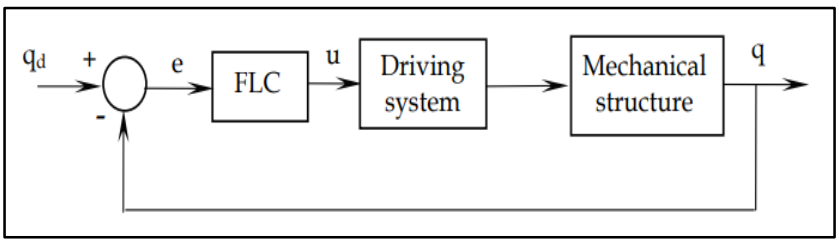


Figura 5.2. Estructura general de un control difuso
Fuente: *(Robot Control by Fuzzy Logic, 2008)*

En la Figura 5.2 se encuentra la estructura genera de un controlador difuso; no obstante, esta estructura puede variar según el proceso a controlar. Es evidente que una variable de entrada para el controlador difuso, es el error de la posición articular que se produce al ejercer movimientos sobre el robot; no obstante, el control difuso referido a una sola variable de entrada no crea la misma precisión ni robustez que crearía un controlador convencional. Para solucionar el problema anterior, se propone añadir una entrada más al controlador difuso, siendo esta la derivada del error; obteniendo así un controlador PD – Difuso; es importante saber que agregar una acción derivativa implica una señal de control proporcional a la derivada del error.

$$u(t) = K_p \frac{de(t)}{dt} \qquad (5.3)$$

En las estrategias de control convencionales el añadir una acción derivativa permite conocer la tendencia del error, por lo tanto, puede producir una corrección antes de que el error sea demasiado grande. Sin embargo, si el sistema se encuentra en estado estacionario y se tiene una perturbación repentina moderada (por ejemplo ruido), hace que la acción derivativa actúe tan bruscamente como rápido fuera el cambio en el error; este efecto de cambio también puede lograr que la magnitud de la señal de control sea tan alta que pueda llegar a saturar al actuador (Collado Oporto, 2016).

Por otra parte, el añadir una acción derivativa a un controlador difuso posee las mismas propiedades que en el controlador convencional a diferencia que el ruido no representa un problema ya que este es imperceptible para el sistema difuso, el error en estado estacionario puede ser tratado en las funciones de membresía de la lógica difusa dando un rango óptimo para que el sistema se considere estable; finalmente, es importante decidir entre el

comportamiento del controlador PD Difuso en régimen estacionario y en régimen transitorio.

5.1.1. Esquemas de Control Difuso

Luego de haber definido el uso de controlador PD Difuso, es necesario plantear el funcionamiento de este. Para esto se plantean dos esquemas en la cual ambos usan el error y su derivada como conjunto de entrada difusa; sin embargo, estos dos esquemas proponen funcionamientos diferentes en la cual se logra una señal de control multiplicando una ganancia a la referencia deseada o sumando una ganancia a la referencia deseada.

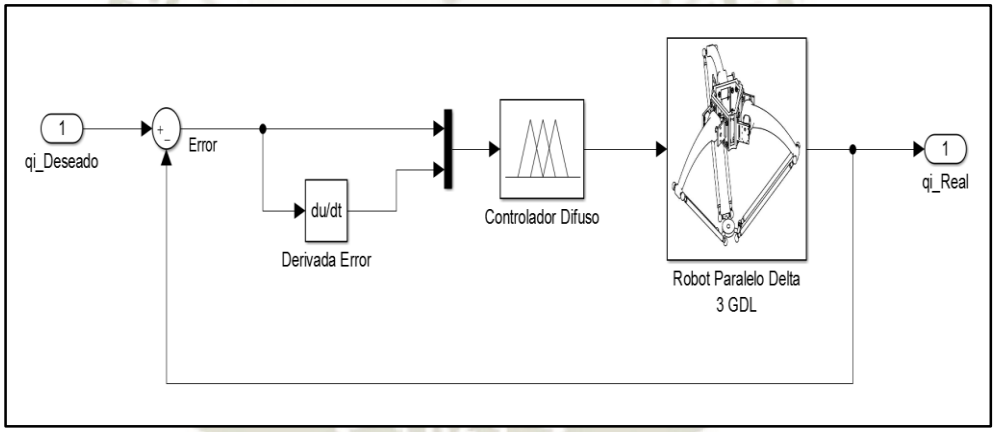


Figura 5.3. Diagrama de Bloques de control PD Difuso multiplicando una ganancia a la referencia
Fuente: *Elaboración Propia.*

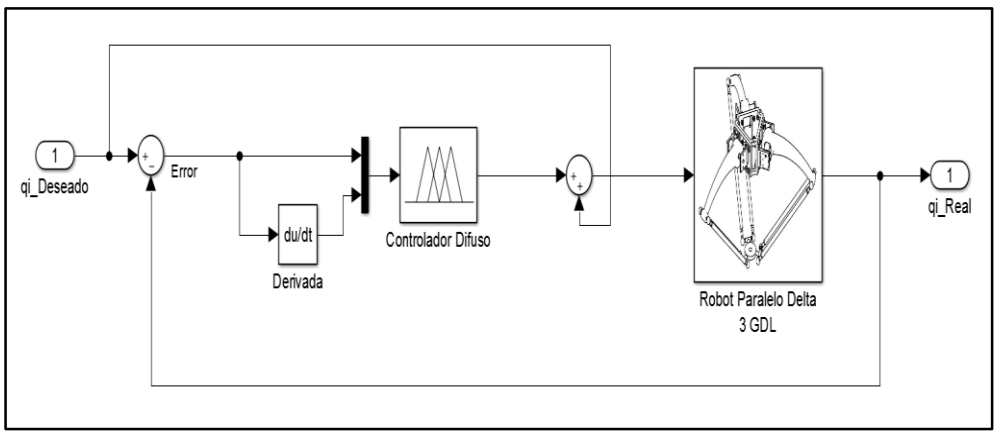


Figura 5.4. Diagrama de Bloques de control PD Difuso sumando una ganancia a la referencia
Fuente: *Elaboración Propia.*

En la Figura 5.3 se encuentra el diagrama de bloques del control PD Difuso multiplicando la señal de control a la referencia deseada, este esquema de control es el más usado en la mayoría de procesos por su alta allegada y porque se sumerge en los conceptos de control convencional. Por otro lado, en la Figura 5.4 se encuentra el diagrama de bloques de control PD Difuso que suma la señal de control difusa a la referencia deseada. La diferencia entre estos dos esquemas parte en que al multiplicar la señal de control con la señal de referencia el cambio se vuelve proporcional a la señal deseada; es decir que si se tiene una señal de control de 0.80 se pierde el 20 % de la referencia deseada indistintamente si el error era mínimo o máximo; en cambio, al sumar una señal de control a la referencia deseada no existe una proporcionalidad y a medida de que el de que error sea más grande la señal de referencia se vuelve menos significativa (Juárez Pérez, 2006).

Si bien es cierto que se plantean dos esquemas de control difusos, definir el cual será usado para el control del robot paralelo delta no resulta una decisión fácil, puesto que aún no se sabe cómo reaccionará el robot con estos esquemas. Como punto de partida, se evaluará los dos esquemas de control propuestos para luego tomar una decisión basado en los resultados más óptimos.

5.2. Diseño del Controlador PD Difuso

Antes de entrar en consideración de que esquema de control difuso se usara es necesario seguir ciertos pasos para el diseño, los cuales son; primero, determinar las entradas y salidas; segundo, determinar las funciones de

membresía; tercero, definir la base de reglas basados en el conocimiento experto; cuarto, probar el controlador con diferentes señales de referencia; quinto, ajustar el controlador difuso hasta que se logren resultados aceptables (C. Sousa & Kaymak, 2002).

De igual manera, es necesario definir el método de inferencia a usar en el control del robot paralelo delta; en este caso se usará el método de Mamdani ya que es más intuitivo y ciertamente útil para representar el conocimiento de expertos; sin embargo, tiene como principal desventaja un mediano coste computacional dependiendo de la cantidad de entradas y reglas de inferencia.

5.2.1. Entradas y Salidas

Como se definió anteriormente, la entrada del control difuso estará basada en el error y su derivada; sin embargo, derivar el error puede tornarse complicado si se busque que este se exprese como tal, es por esto que se utilizara la diferenciación del error para que se obtenga el cambio del error. De esta manera se obtiene:

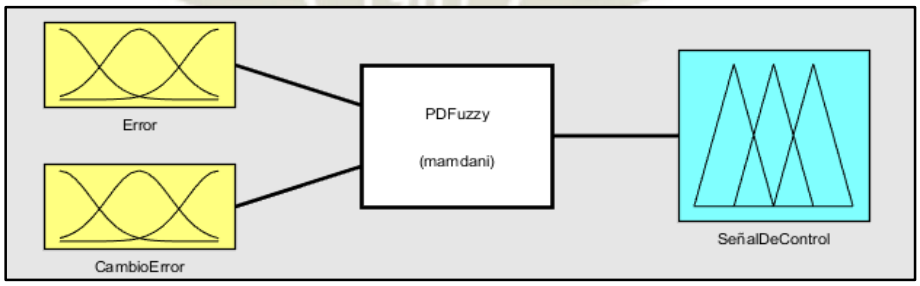


Figura 5.5. Entradas y Salidas del Control PD-Difuso
Fuente: *Elaboración Propia.*

En la Figura 5.5 se observa dos entradas; la primera es la señal de error generada a partir de la diferencia entre la posición articular deseada con

la posición articular real y la segunda está referida al cambio de error; es decir el error menos el error anterior.

5.2.2. Funciones de Membresía y Variables Lingüísticas

Las funciones de membresía representan una parte crucial en la lógica difusa, estas definen el grado de pertenencia de las variables lingüísticas. En el caso del robot paralelo delta, la representación del error, la diferencia del error y la señal de control estarán definidas por variables lingüísticas que van desde funciones que indique un gran cambio hasta un pequeño cambio.

Tomando en cuenta la representación del error, se definirá dos casos con diferentes variables lingüísticas que se pueden plantear para el control del robot paralelo delta siendo estos.

Tabla 5.1. Variables Lingüísticas opción 1

	ERROR	CAMBIO DE ERROR	SEÑAL DE CONTROL
CASO 1	Negativo Grande (NG)	Negativo Grande (NG)	Negativo Grande (NG)
	Negativo Pequeño (NP)	Negativo Pequeño (NP)	Negativo Pequeño (NP)
	Cero (C)	Cero (C)	Cero (C)
	Positivo Pequeño (PP)	Positivo Pequeño (PP)	Positivo Pequeño (PP)
	Positivo Grande (PG)	Positivo Grande (PG)	Positivo Grande (PG)

Fuente: *Elaboración Propia.*

Tabla 5.2. Variables Lingüísticas opción 2

	ERROR	CAMBIO DE ERROR	SEÑAL DE CONTROL
CASO 2	Negativo Grande (NG)	Negativo (N)	Negativo Grande (NG)
	Negativo Pequeño (NP)		Negativo Pequeño (NP)
	Cero (C)	Cero (C)	Cero (C)
	Positivo Pequeño (PP)	Positivo (P)	Positivo Pequeño (PP)
	Positivo Grande (PG)		Positivo Grande (PG)

Fuente: *Elaboración Propia.*

En la Tabla 5.1, se presentan cinco variables lingüísticas para cada entrada y salida, respecto al error y el cambio de error se considera que el error puede variar entre ser un valor negativo grande hasta un positivo grande; mientras tanto, la salida también refleja el mismo comportamiento; esto se hace con el fin de tener una mejor perspectiva a la hora de implementar el control. Por otro lado, la Tabla 5.2 muestra trece variables en donde el error toma un mayor peso sobre las decisiones del controlador; no obstante, el cambio de error se reduce a tres variables lingüísticas logrando así que la señal de control no sea afectada a grandes rasgos.

Si bien es cierto que se han planteado distintas variables lingüísticas y se ha reconocido que en algunos casos unas predominan más que otras, es necesario indicar los conjuntos de membresía a usar en el control difuso; a este respecto, se plantea iniciar con funciones de membresía lineales ya que reducen notablemente el costo computacional del controlador difuso y porque su interpretación simplifica las reglas que el experto configurara en el controlador. Las funciones de membresía que se usaran para el controlador serán dos; la primera, es la función triangular ya que es muy adecuada en las situaciones donde se tiene un valor optimo central; la segunda función que usara el controlador es la función trapezoidal debido a que presenta un rango constante que puede ser usado para indicar si un rango de valores es bueno o malo para el proceso a controlar.

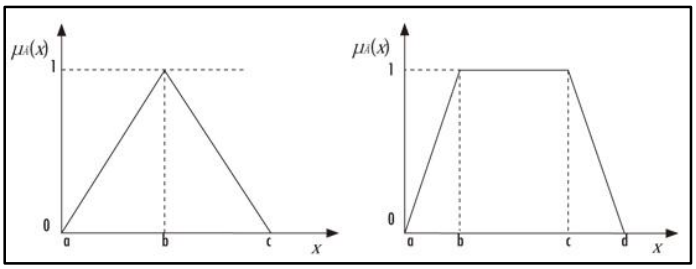


Figura 5.6. a) Función Triangular b) Función Trapezoidal
Fuente: *(Adarme Jaimes, Arango Serna, & Cogollo Flórez, 2012)*

5.2.3. Funcionamiento del Robot Paralelo Delta sin Control

Para continuar con el diseño del controlador PD – Difuso es imprescindible conocer cómo funciona el robot paralelo delta sin control alguno, esto también define un punto de partida para el usuario experto que configurara el universo de las entradas y salidas del controlador difuso, los rangos de las funciones de membresía y las reglas de inferencia. Para esto se plantea probar un movimiento variable del robot en los que se vean comprometidas variaciones en la posición articular y la velocidad angular.

En relación con las implicaciones se usará el programa trayectoria planteado en el capítulo anterior, para luego imprimir constantemente los datos de posición angular tomados por el controlador esclavo y llevarlos a una tabla en el software Excel para su respectivo análisis.

```
tiempofu = millis();

Serial.print("Tiempo =");Serial.print(tiempofu);
Serial.print("=Angulo_Q1 =");Serial.print(q1);Serial.print(" ");
Serial.print("=Angulo_Q2 =");Serial.print(q2);Serial.print(" ");
Serial.print("=Angulo_Q3 =");Serial.print(q3);Serial.print(" ");
Serial.print("=PosReal_1=");Serial.print(pos1);
Serial.print("=PosReal_2=");Serial.print(pos2);
Serial.print("=PosReal_3=");Serial.print(pos3);
Serial.print("=SETPOINT_Q1 =");Serial.print(Q1);Serial.print(" ");
Serial.print("=SETPOINT_Q2 =");Serial.print(Q2);Serial.print(" ");
Serial.print("=SETPOINT_Q3 =");Serial.print(Q3);Serial.print(" ");
Serial.print("=Vel_qdm1=");Serial.print(S1);Serial.print(" ");
Serial.print("=Vel_qdm2=");Serial.print(S2);Serial.print(" ");
Serial.print("=Vel_qdm3=");Serial.print(S3);Serial.println(" ");
```

Figura 5.7. Código estructurado para la recolección de datos
Fuente: *Elaboración Propia.*

Como seguimiento de las actividades se obtuvo los siguientes resultados:

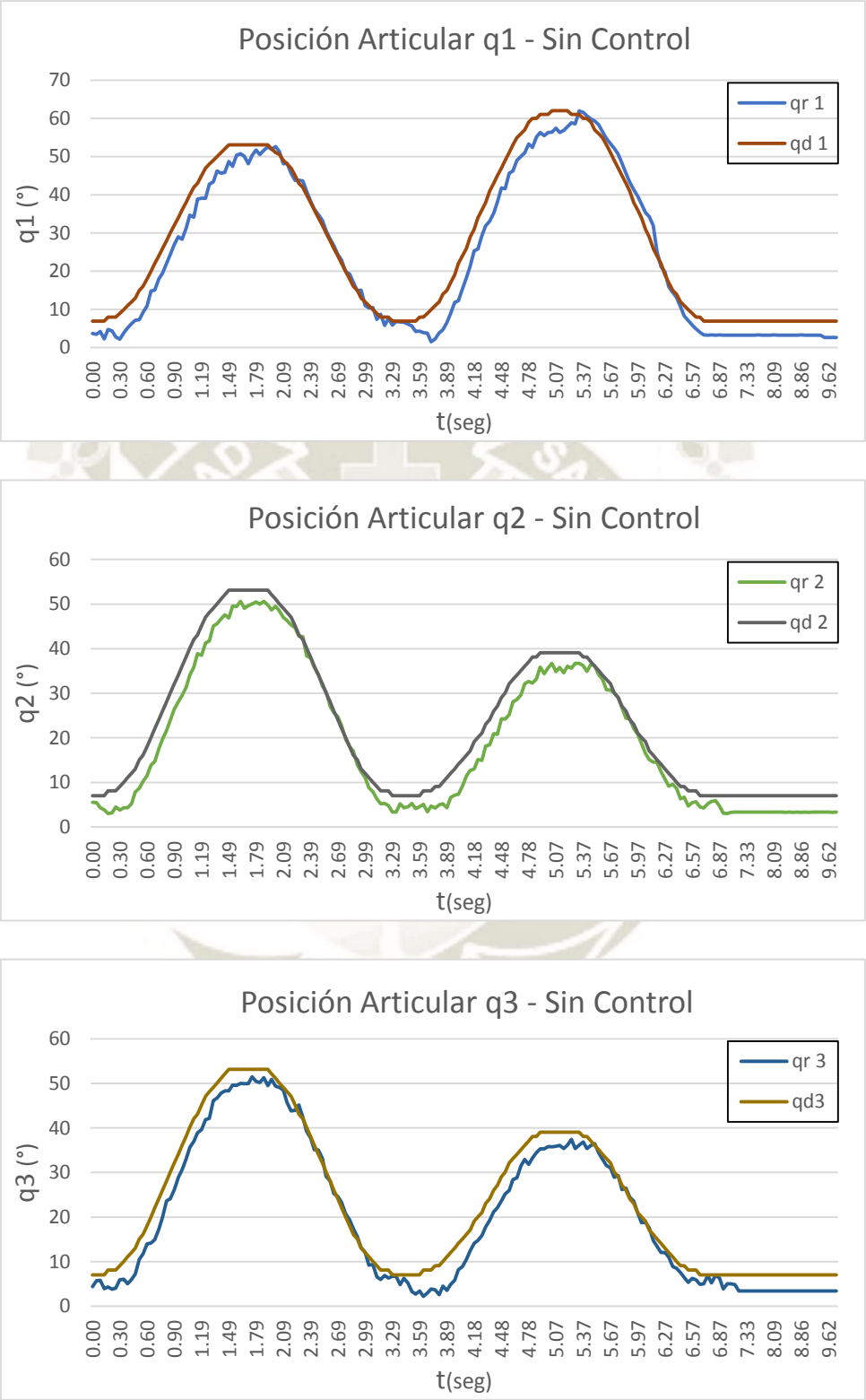


Figura 5.8. Trayectoria articular real (q1, q2, q3) vs Trayectoria articular deseada (q1, q2, q3)
Fuente: *Elaboración Propia*

En la Figura 5.8 se encuentran tres gráficos que muestran la evolución de la trayectoria del robot paralelo delta en el paso del tiempo; las líneas con nombre “qr” indican la trayectoria articular real obtenida de la lectura de los servomotores mientras que las líneas con nombre “qd” representa la trayectoria deseada impuesta por el programa trayectoria del controlador maestro. Antes de entrar en consideración de los resultados es necesario especificar que los actuadores Dynamixel cuenta con un control proporcional interno “P” con un valor $K_P = 6.25$; desafortunadamente la acción de control no basta para cumplir con las señales de referencias deseadas.

A simple vista, se observa que existe una notable diferencia entre las posiciones deseadas y las obtenidas, incluso existen momentos en los que se tienen picos muy altos de error. Al ser una cadena cinemática cerrada, el error que genera una articulación afecta enormemente a las demás articulaciones, por lo tanto, crea momentos de alta inestabilidad en el robot paralelo delta.

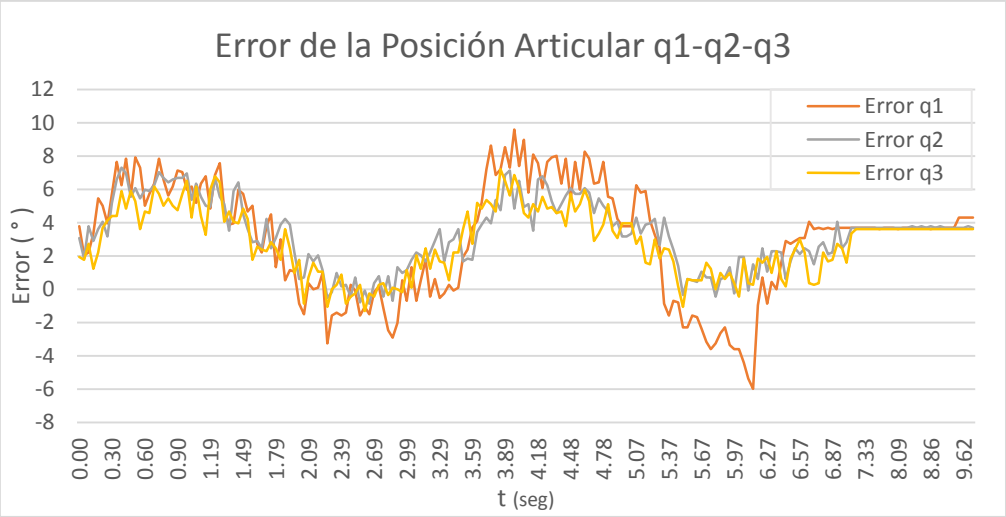


Figura 5.9. Error de la posición articular de todas las articulaciones
Fuente: *Elaboración Propia*

La Figura 5.9 muestra la evolución del error en cada articulación respecto al tiempo, en esta se ve con gran claridad los errores máximos que se producen; estos errores serán de gran importancia para la configuración de los parámetros del control difuso.

Tabla 5.3. Errores máximos y mínimos de cada articulación

	q1	**q2**	**q3**
Valor Máximo (°)	9.59	7.30	7.22
Valor Mínimo (°)	0	0.09	0

Fuente: *Elaboración Propia.*

Finalmente, con el fin de entender con mayor claridad cómo es que los errores presentados en las articulaciones afectan al movimiento cartesiano del robot, se presenta la Figura 5.10; en la cual se observa la trayectoria deseada y la trayectoria real obtenida a partir de la cinemática directa.

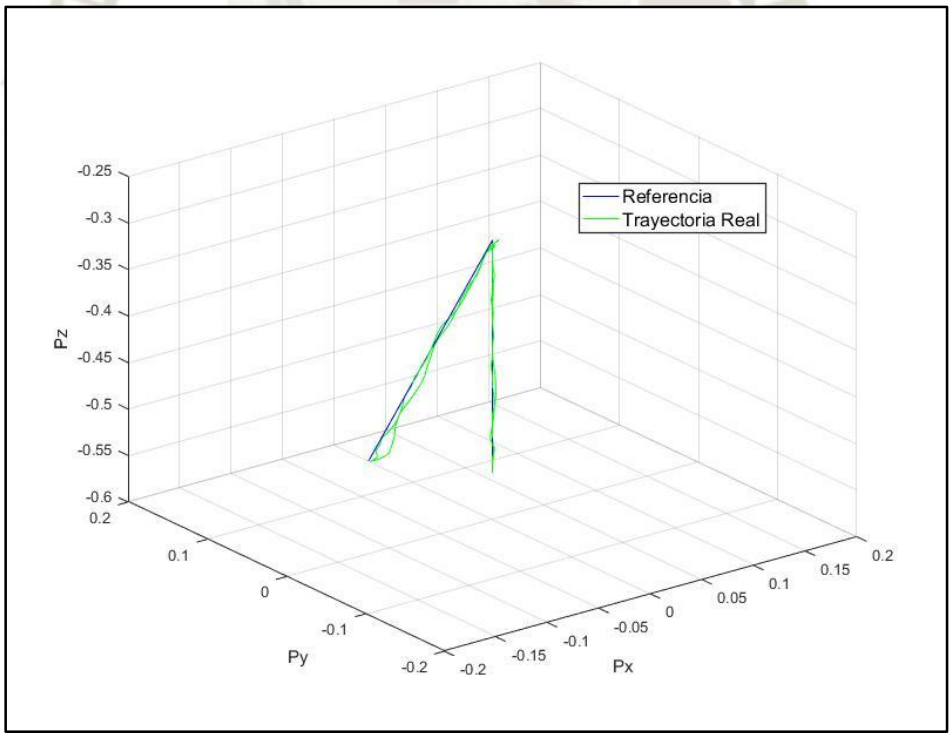


Figura 5.10. Trayectoria cartesiana deseada vs Trayectoria cartesiana real
Fuente: *Elaboración Propia.*

En la Figura 5.10 se observa una notable variación en los movimientos ejercidos por el robot paralelo delta, tanto al momento de ir a las posiciones iniciales y al regresar a la posición intermedia.

2.1.1. 5.2.4. Reglas de Inferencia y Parámetros del Controlador PD Difuso

Hasta el presente se planteó el uso de las funciones de membresía a ser usadas y las variables lingüísticas necesarias para llevar a cabo el control difuso; luego se observó el comportamiento del robot paralelo delta y la manera en la que este fallaba al ejercer una trayectoria predefinida. Partiendo de las consideraciones anteriores es posible definir los parámetros necesarios para lograr un buen control.

Respecto a las reglas de inferencia a usar se tomaron en cuenta tres modelos propuestos por diversos autores. El primer modelo está basado en la matriz de reglas difusas propuesta por (Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería, 2010) y se observa en las siguientes tablas.

Tabla 5.4. Modelo 1 Matriz de 25 Reglas Difusas para el controlador PD

		Error				
		NG	**NP**	**Z**	**PP**	**PG**
Derivada Error	**NG**	NG	NP	Z	PP	PG
	NP	NG	NP	Z	PP	PG
	Z	NG	NP	Z	PP	PG
	PP	NP	NP	Z	PP	PG
	PG	NG	NP	Z	PP	PG

Fuente: (Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería, 2010)

En la Tabla 5.4 se observa todas las variables lingüísticas que serán usadas en el controlador difuso, estas van desde variables negativas

grandes hasta positivas grandes. La matriz de reglas difusas propuesta por Ponce Cruz, llama la atención porque en esta se ve que se le da más prioridad al error que a la derivada del error.

La segunda tabla también concierne al modelo 1 y es una variante del modelo original propuesto por Ponce Cruz; en esta se observa una matriz de reglas difusas que contiene 15 resultados, esto debido a que se trabaja con trece variables lingüísticas tal como se propuso en la sección anterior.

Tabla 5.5. Modelo 2 – Matriz de 15 Reglas Difusas para el controlador PD

		Error				
		NG	**NP**	**Z**	**PP**	**PG**
Derivada Error	**N**	NG	NP	Z	PP	PG
	Z	NG	NP	Z	PP	PG
	P	NG	NP	Z	PP	PG

Fuente: *Elaboración Propia.*

El tercer modelo estaba basado en la matriz de reglas difusas propuesto por (C. Sousa & Kaymak, 2002), en esta se observa 25 reglas de inferencia, aunque, en este modelo se toma con mayor consideración la derivada del error.

Tabla 5.6. Modelo 3 – Matriz de 25 Reglas Difusas para el controlador PD

		Error				
		NG	**NP**	**Z**	**PP**	**PG**
Derivada Error	**NG**	NG	NG	NP	NP	Z
	NP	NG	NP	NP	Z	PP
	Z	NP	NP	Z	PP	PP
	PP	NP	Z	PP	PP	PG
	PG	Z	PP	PP	PG	PG

Fuente: (C. Sousa & Kaymak, 2002)

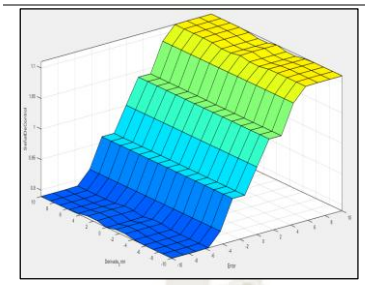
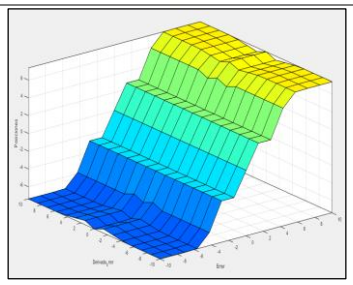
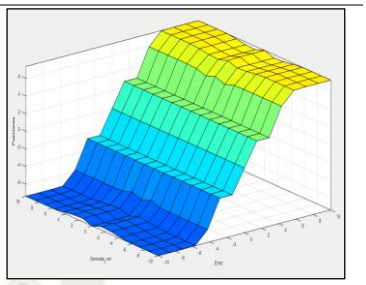
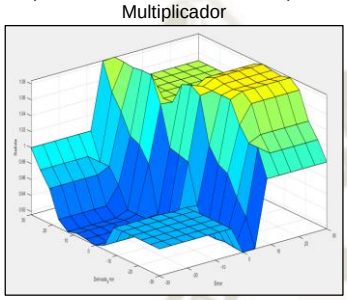
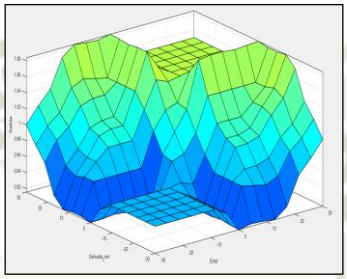
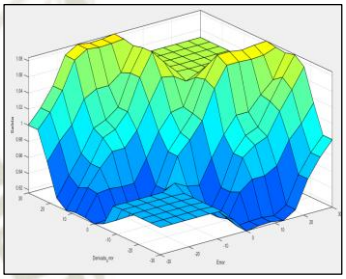
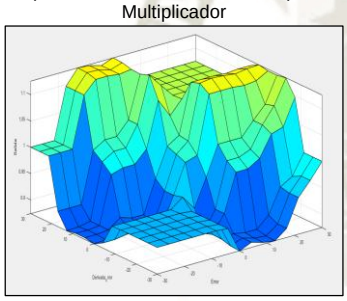
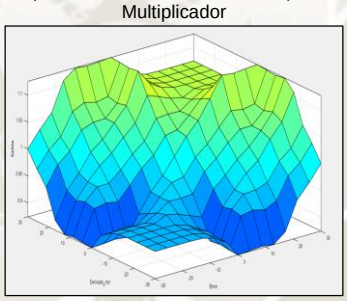
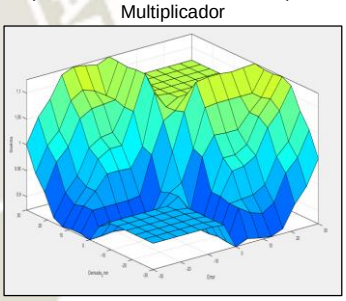
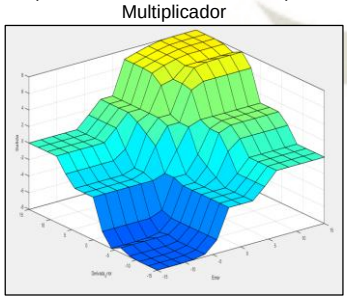
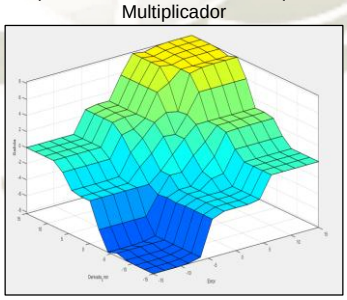
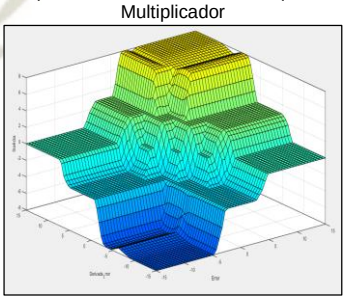
De las tablas mostradas anteriormente se tomará en cuenta que las reglas de inferencia difusas serán complementadas con la función AND;

por lo que se tiene que cumplir cada condición impuesta para que se dé un resultado diferente.

Hay que tomar en cuenta que para llegar a un resultado óptimo es necesario realizar diversas pruebas que evalúan diferentes rangos de las funciones de membresía. Para esto se propone probar controladores con los dos modelos de reglas difusas planteados y con los dos esquemas propuestos al inicio del presente capítulo.

El uso de la lógica difusa permite expresar el comportamiento del sistema en gráficos si es que se tienen dos o tres variables a manipular; en este caso al usar dos señales de entrada y una de salida es posible expresar el comportamiento del controlador difuso en un gráfico tridimensional. Ya sabido esto es posible observar los diferentes controladores a probar.

Tabla 5.7. Superficies Tridimensionales de los Diferentes Controladores Difusos Propuestos

SUPERFICIES DIFUSAS PROPUESTAS		
 Superficie N°1 – Modelo 01 – Esquema Multiplicador	 Superficie N°2 – Modelo 02 – Esquema Sumador	 Superficie N°3 – Modelo 02 – Esquema Sumador
 Superficie N°4 – Modelo 03 – Esquema Multiplicador	 Superficie N°5 – Modelo 03 – Esquema Multiplicador	 Superficie N°6 – Modelo 02 – Esquema Multiplicador
 Superficie N°7 – Modelo 03 - Esquema Multiplicador	 Superficie N°8 – Modelo 03 -Esquema Multiplicador	 Superficie N°9 – Modelo 03 -Esquema Multiplicador
 Superficie N°10 – Modelo 03 – Esquema Sumador	 Superficie N°11 – Modelo 03 – Esquema Sumador	 Superficie N°12 – Modelo 03 – Esquema Sumador

Fuente: *Elaboración Propia.*

De la Tabla 5.7 se observan múltiples superficies generadas a partir de los tres modelos de conjuntos de reglas difusas y diferentes rangos en las funciones de membresías. Entorno a las funciones de membresía, los rangos para las funciones de entrada varían dependiendo a los

errores máximos y mínimos vistos en las pruebas realizadas; mientras tanto los rangos para las funciones de salida varían dependiendo si el esquema a usar corresponde al diagrama de bloques multiplicador o al diagrama de bloques sumador, ya que un universo de salida muy alta generaría señales de control desproporcionadas a la salidas si es que el esquema corresponde a un sistema de control que multiplique la señal de control; por otro lado; un universo de salida muy reducido lograría que la señal de control no afecte lo suficiente al robot paralelo delta si es que se está trabajando con un esquema que corresponde al sistema de control que suma la señal de control a la señal de referencia.

Ya propuestas las diferentes superficies de control es necesario elaborar pruebas para analizar el desempeño de los controladores propuestos; no obstante, al principio del presente capítulo se indicó que al tener un modelo dinámico complejo se omitirá cualquier evaluación sobre este. Por lo tanto, todas las evaluaciones se harán de manera física, obteniendo resultados reales sobre los controladores planteados.

5.3. Implementación del Controlador PD Difuso

Una de las mayores desventajas al proponer una estrategia de control difusa es que la mayoría de controladores ofrece el uso de estrategias de control convencionales y si es que se planea usar una no convencional se vuelve complicado ya que en la mayoría de las veces se tiene que crear el algoritmo de control. En el ámbito educacional, la gran mayoría de proyectos que usan lógica difusa son elaborados a partir de la herramienta *Fuzzy* ofrecida por el software MATLAB; no obstante, al usar controladores de lenguaje

estructurado y con programas ya propuestos el uso de MATLAB resulta descartado.

Afortunadamente, gracias a la gran comunidad de programadores crecientes en los últimos años y con el apoyo de la comunidad de Arduino; se desarrolló una librería llamada "*eFLL (Embedded Fuzzy Logic Library)" creada por* Robotic Research Group (RRG) en la Universidad Estatal de Piauí (UESPI-Teresina).

La librería "eFLL (*Embedded Fuzzy Logic Library*) "es una biblioteca estándar para sistemas embebidos hacia sistemas difusos fáciles y eficientes. Está escrito en lenguaje C ++ / C, utiliza solo la biblioteca estándar de lenguaje C "stdlib.h", por lo que "eFLL" es una biblioteca diseñada no solo para Arduino, sino para cualquier sistema embebido que use comandos en lenguaje C. No tiene limitaciones explícitas sobre la cantidad de reglas difusas, entradas o salidas; estas capacidades de procesamiento y almacenamiento están limitadas al tipo de microcontrolador (Alves, Lemos, S. Kridi , & Leal, 2018).

La librería usa las operaciones de máximo y mínimo, el método de inferencia usado es el de Mamdani y el método para la defusificación es de centro de gravedad en un universo continuo.

Con el objetivo de entender correctamente el funcionamiento de la librería eFLL, se muestran las siguientes instrucciones ofrecidas por la librería:

- **Fuzzy:** Incluye todo el sistema Fuzzy, a través del cual se pueden manipular los conjuntos difusos, las reglas lingüísticas, las entradas y las salidas.

- **FuzzyInput:** Agrupa todas las entradas de conjuntos difusos que pertenecen al mismo dominio.
- **FuzzyOutput:** Agrupa todas las salidas de conjuntos difusos que pertenecen al mismo dominio.
- **FuzzySet:** Esta instrucción modifica los rangos de las funciones de membresía; por el momento solo admite funciones de pertenencia triangulares, trapezoidal y singleton, que se ensamblan en base a los puntos A, B, C y D.
- **FuzzyRule:** Sirve para dar inicio a una nueva regla.
- **FuzzyRuleAntecedent:** Es el responsable de ensamblar el antecedente de la expresión condicional de una regla difusa.
- **FuzzyRuleConsequente:** Ensamblar la expresión de salida de una regla difusa.

```
// Libreria eFLL
Fuzzy* fuzzy = new Fuzzy();
//Funciones de error
FuzzySet* NG = new FuzzySet(-20, -20, -12.5, -5.5);FuzzySet* NP = new FuzzySet(-12, -8, -8, -2);FuzzySet* CERO = new FuzzySet(-2.5, 0, 0, 2.5);
FuzzySet* PP = new FuzzySet(2, 8, 8, 12);FuzzySet* PG = new FuzzySet(5.5, 12.5, 20, 20);
//Funciones de derivada de error
FuzzySet* DNG = new FuzzySet(-20, -20, -12.5, -5.5);FuzzySet* DNP = new FuzzySet(-10, -5, -5, -1);FuzzySet* DZ = new FuzzySet(-4,0,0,4);
FuzzySet* DPP = new FuzzySet(1, 5, 5, 10);FuzzySet* DPG = new FuzzySet(5.5, 12.5, 20, 20);
// Fuzzy Output posicion
FuzzyOutput* posicion = new FuzzyOutput(1);
FuzzySet* NGO = new FuzzySet(0.9,0.9,0.9,0.95);
posicion->addFuzzySet(NGO);
FuzzySet* NPO = new FuzzySet(0.9,0.95,0.95,0.98);
posicion->addFuzzySet(NPO);
FuzzySet* ZO = new FuzzySet(0.97,1,1,1.03);
posicion->addFuzzySet(ZO);
FuzzySet* PPO = new FuzzySet(1.02,1.05,1.05,1.08);
posicion->addFuzzySet(PPO);
FuzzySet* PGO = new FuzzySet(1.05,1.1,1.1,1.1);
posicion->addFuzzySet(PGO);
```

```
// REGLAS DIFUSAS
 //REGLA1
 FuzzyRuleAntecedent* errorNGAndDeDNG = new FuzzyRuleAntecedent();
 errorNGAndDeDNG->joinWithAND(NG, DNG);
 FuzzyRuleConsequent* thenPosicionNPO1 = new FuzzyRuleConsequent();
 thenPosicionNPO1->addOutput(NPO);
 FuzzyRule* fuzzyRule1 = new FuzzyRule(1, errorNGAndDeDNG, thenPosicionNPO1);
 fuzzy->addFuzzyRule(fuzzyRule1);
 //REGLA2
 FuzzyRuleAntecedent* errorNGAndDeDNP = new FuzzyRuleAntecedent();
 errorNGAndDeDNP->joinWithAND(NG, DNP);
 FuzzyRuleConsequent* thenPosicionNPO2 = new FuzzyRuleConsequent();
 thenPosicionNPO2->addOutput(NPO);
 FuzzyRule* fuzzyRule2 = new FuzzyRule(2, errorNGAndDeDNP, thenPosicionNPO2);
 fuzzy->addFuzzyRule(fuzzyRule2);
 // Fusificación y Defusificación
 fuzzy->setInput(1, error1);
 fuzzy->setInput(2, de1);
 fuzzy->fuzzify();
 float o1 = fuzzy->defuzzify(1);
```

Figura 5.11. Ejemplo de programación de la librería eFLL
Fuente: *Elaboración Propia.*

Dentro de este marco, es necesario recalcar que el algoritmo de control difuso se encuentra en el controlador esclavo OpenCM9.04, ya que este es el controlador propio de los actuadores y la realimentación de datos es de manera directa hacia dicho controlador; dicho esto es posible realizar las pruebas necesarias para la correcta selección e implementación del control PD Difuso.

5.4. Evaluación y Selección del Controlador PD - Difuso

A lo largo del presente capítulo se propusieron dos esquemas de control, tres modelos de matrices de reglas difusas y se plantearon diferentes superficies que definen el comportamiento del controlador difuso; todo lo anterior expuesto se hizo con el fin de encontrar el mejor controlador difuso que se acople al comportamiento del robot paralelo delta.

Con el fin de presentar la mejor solución y de manera resumida, en esta sección del apartado se presentarán las superficies de control que más llamaron la atención ya sea por su comportamiento erróneo o por la aproximación al mejor resultado.

5.4.1. Superficie del Modelo Uno con Esquema Multiplicador

Esta evaluación pertenece a la superficie propuesta N° 1; en esta se obtiene un universo de salida de 0.9 a 1.1; esto con el fin de evitar que la señal de control sufra cambios de alta magnitud.

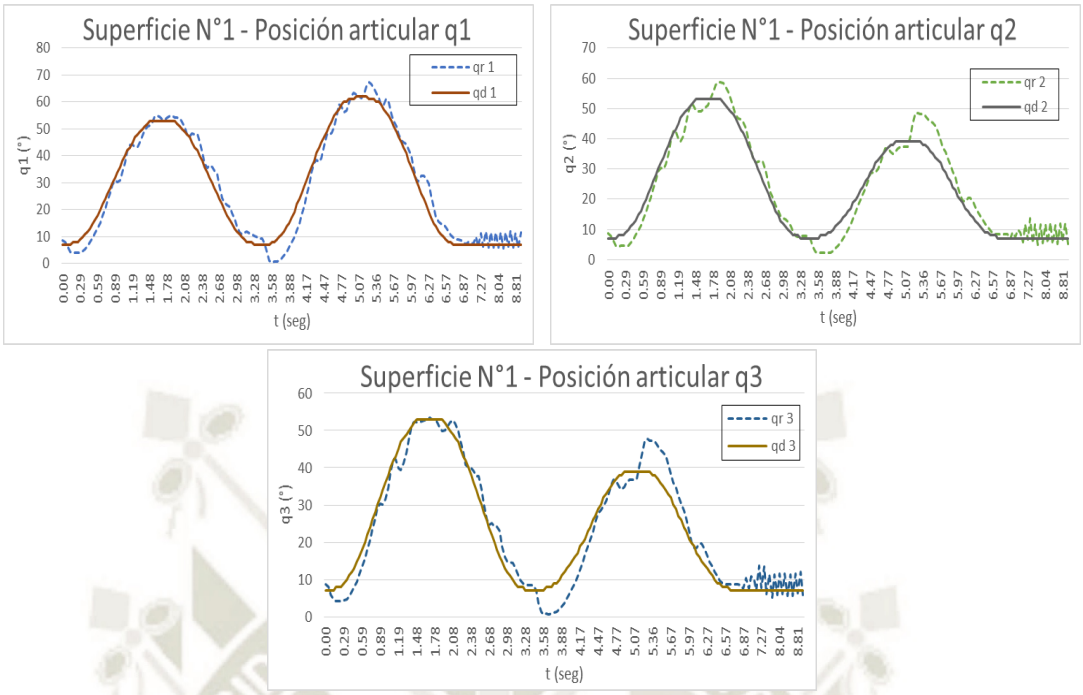


Figura 5.12. Evaluación de trayectorias articulares con la superficie N°1

Fuente: *Elaboración Propia*

La Figura 5.12 muestra que con los parámetros de la superficie N°1 compensa el error a lo largo de la trayectoria; sin embargo, existen momentos donde la señal de control envía valores que incrementa de manera desmedida el error y cuando la señal de control llega a estado estacionario, sufre un sobre impulso constante; logrando así que el robot paralelo delta se vuelva inestable. En resumen, la superficie N° 1 no cumple con las condiciones mínimas para ser tomado en cuenta como un buen controlador difuso.

5.4.2. Superficie de Modelo Dos con Esquema Sumador

La siguiente evaluación pertenece a las superficies dos y tres; no obstante, se mostrarán los gráficos de una ya que ambos resultarlos fueron muy parecidos. En este caso, el universo de salida se expandió de -10 a 10 debido a que se planea compensar los errores mediante una suma.

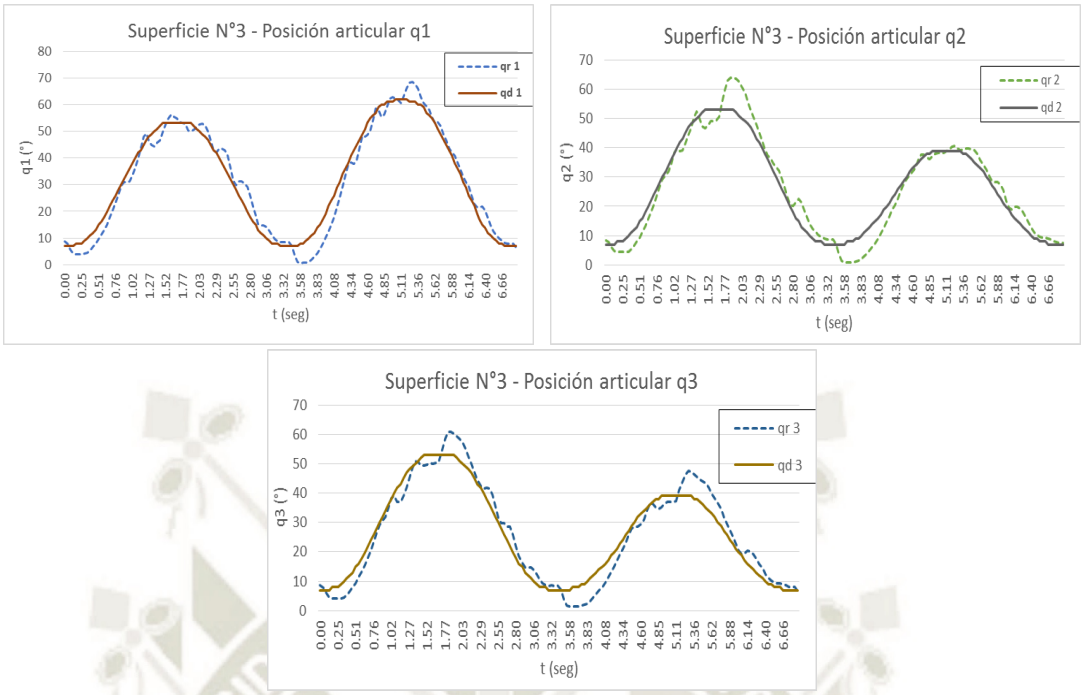


Figura 5.13. Evaluación de trayectorias articulares con la superficie N°3
Fuente: *Elaboración Propia*

En la Figura 5.13 se observa que existen sobre impulsos cada vez que el error se acerca a cero; esto podría deberse al reducido rango de la función de membresía que conlleva la variable lingüística cero. Al comparar estas evidencias, también se observa que la acción derivativa no afecta a grandes rasgos al controlador ya que en algunos momentos el error es demasiado grande; esto significa que no se conoce correctamente la tendencia del error. Por otro lado, a diferencia del esquema multiplicador, el usar un esquema que sume la señal de control a la referencia deseada evita cambios bruscos; no obstante, aún no se tienen resultados óptimos para el controlador PD Difuso.

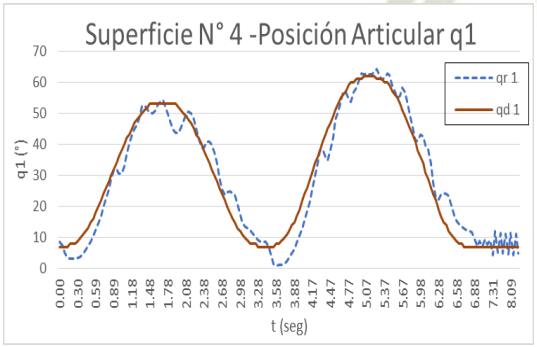
Hasta el momento se utilizó la matriz de reglas difusas propuesta por (Ponce Cruz, Inteligencia Artificial con Aplicaciones a la Ingeniería, 2010) y su respectiva variante; los resultados obtenidos por estas

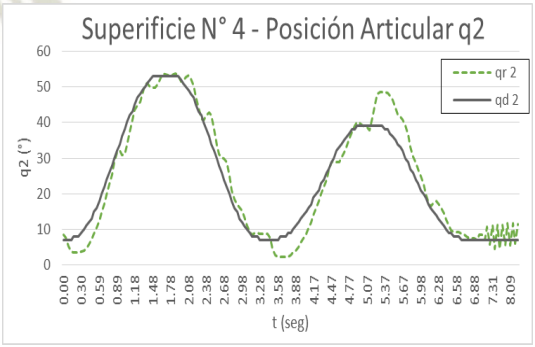
matrices no fueron favorables ya que no se reduce o elimina el error notablemente y en algunos momentos el robot pierde el control, creando movimientos erróneos en las articulaciones que a su vez se ven reflejados en el espacio cartesiano.

5.4.3. Superficie de Modelo Tres con Esquema Multiplicador

Las siguientes evaluaciones pertenecen a las superficies cuatro hasta la superficie nueve; estas también indican una evolución ya que poco a poco se va modificando sus universos de entradas y salidas y los rangos de las funciones de membresía. Cabe recalcar que este modelo toma en cuenta 25 reglas difusas donde se puede ver que tanto como el error y la derivada del error asumen un rol más importante que en los anteriores modelos vistos.

Inicialmente se trabajó con la superficie N° 4, en esta se puede observar una gran variación entre las variables negativas con las variables positivas; también, el universo de salida comprende un rango de valores de 0.92 a 1.08; al igual que los casos anteriores esto se hace con el fin de evitar salidas de gran magnitud.





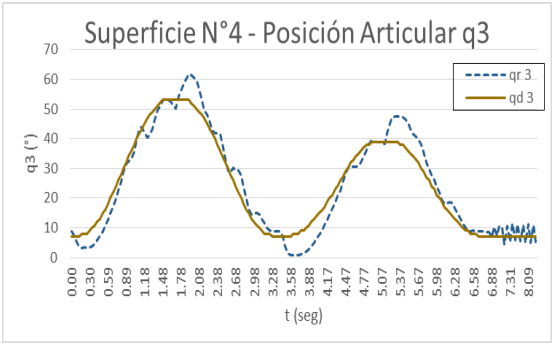


Figura 5.14. Evaluación de trayectorias articulares con la superficie N°4
Fuente: *Elaboración Propia*

En la Figura 5.14 se observa el comportamiento articular del robot paralelo delta frente al controlador difuso usando la superficie N°4; al igual que en el primer caso se observa que existe sobre impulsos cada vez que el error se torna más pequeño; también, cada vez que un valor cambio de mayor a menor o viceversa existe un gran pico de error; esto se puede deber a la acción derivativa, ya que intenta predecir el error y al encontrarse con un cambio que van en sentido opuesto a la tendencia original se magnifica la señal de control y por ende falla. Finalmente, al encontrarse en estado estacionario el robot paralelo delta se vuelve inestable y esto se puede apreciar en todos los cuadros mostrados a partir del séptimo segundo.

A pesar de haber reducido notablemente los rangos de errores pequeños y errores grandes, se puede apreciar que en algunos instantes de tiempos el error articular obtenido sobrepasa a los valores obtenidos de robot paralelo delta sin control, estas evidencias hacen pensar si es que realmente es aconsejable usar un esquema difuso que multiplique la señal de control, ya que este comportamiento solo se observa en los esquemas de multiplicación.

Con el fin de seguir probando el esquema de multiplicación, en las superficies 5 y 6 se redujo la función de membresía de la variable lingüística “Cero” para las entradas y salidas a valores que varían desde -5.2° a 5.2°, esto con el objetivo de evitar la inestabilidad del robot en estado estacionario; finalmente al probar una rango de valores más pequeño en las funciones de membresía “Error Negativo Pequeño y Error Positivo Pequeño” y aumentar los rangos de las funciones de membresía para la derivada del error se obtuvo la superficie N° 7 y por ende el siguiente resultado.

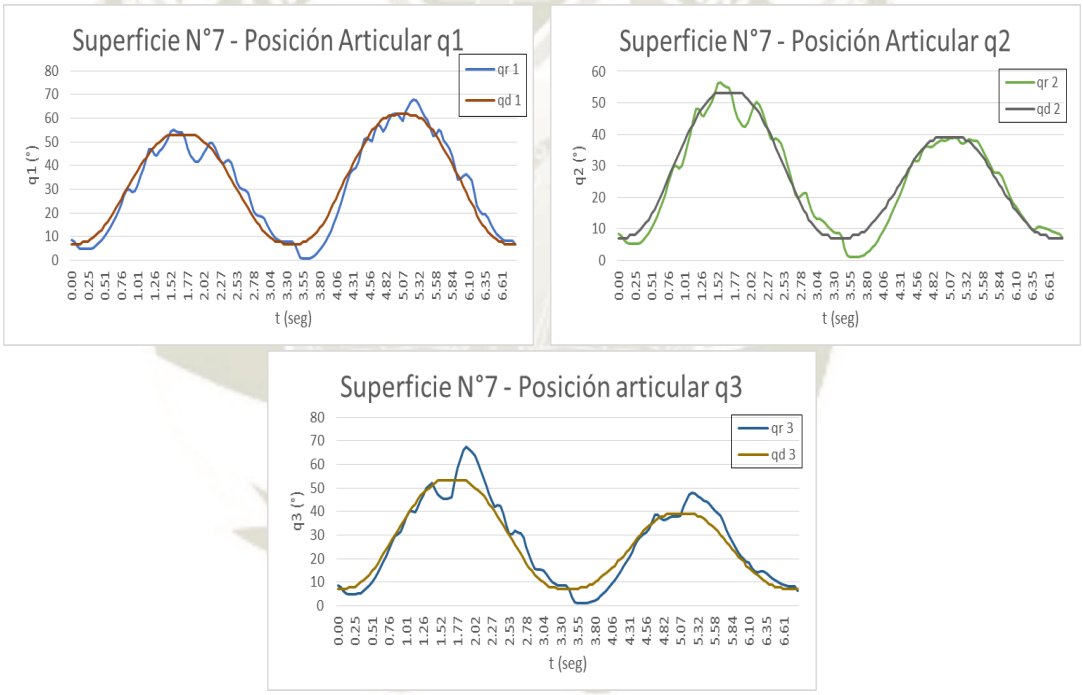


Figura 5.15. Evaluación de trayectorias articulares con la superficie N°7
Fuente: *Elaboración Propia*

En la Figura 5.15 se observa el mismo patrón de comportamiento de los resultados anteriores, ya que cada vez que el error tiende a ser cero, existe sobre impulsos que hacen que por momentos se pierda la estabilidad. Así mismo, los resultados de la superficie N°7 guarda mucha similitud con los de la superficie N°3; esto podría deberse a que redujo

notablemente los rangos de las funciones de membresía de la variable lingüística del error. A pesar del esfuerzo, las evaluaciones realizadas a la superficie N° 8 y N° 9 muestran los mismos resultados cuando el error tiende a cero.

Hasta el presente se han tratado distintas superficies en las que se puso a prueba el esquema, la matriz de reglas difusas y diferentes rangos para las funciones de membresía; obteniendo así distintos resultados que lejos de llegar a ser favores para el robot paralelo delta, nos sirvieron como guía para aprender el porqué de su comportamiento erróneo. En este punto del proyecto de tesis es necesario hacer énfasis que fueron probados muchísimas más superficies de control y que la tabla de superficies propuesta en la sección anterior es el resultado de la evolución de un control con resultados más favorables.

5.4.4. Superficie de Modelo Tres con Esquema Sumador

Las siguientes evaluaciones pertenecen a las superficies N°10, N° 11 y N° 12; como ya se mencionó antes luego de varias evaluaciones a distintas superficies se determinó que un esquema sumador se adapta con mejores resultados al robot paralelo delta; puesto que al no tener una salida proporcional la señal de control no sufre severos cambios de amplitud; no obstante, si el error es demasiado grande la señal de control no llegaría a eliminar correctamente el error.

Por una parte, tener un universo de entrada más limitado en las variables lingüísticas del error y la derivada del error logra que la señal de control obtenga resultados favorables, en la cual se evitan los grandes sobre impulsos. Por otro lado, para evitar el sobre impulso cada vez que el

error se vuelve mínimo, se dio el mismo rango a las variables lingüísticas “Cero”. Tomando en cuenta todas las consideraciones anteriores se obtuvo el siguiente resultado.

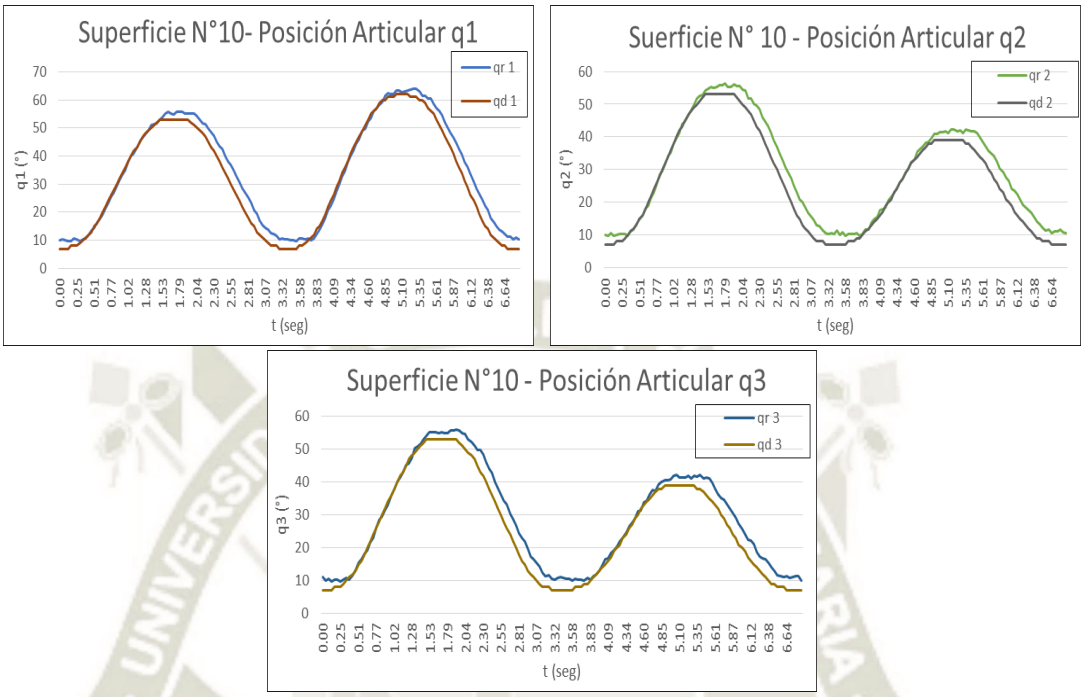


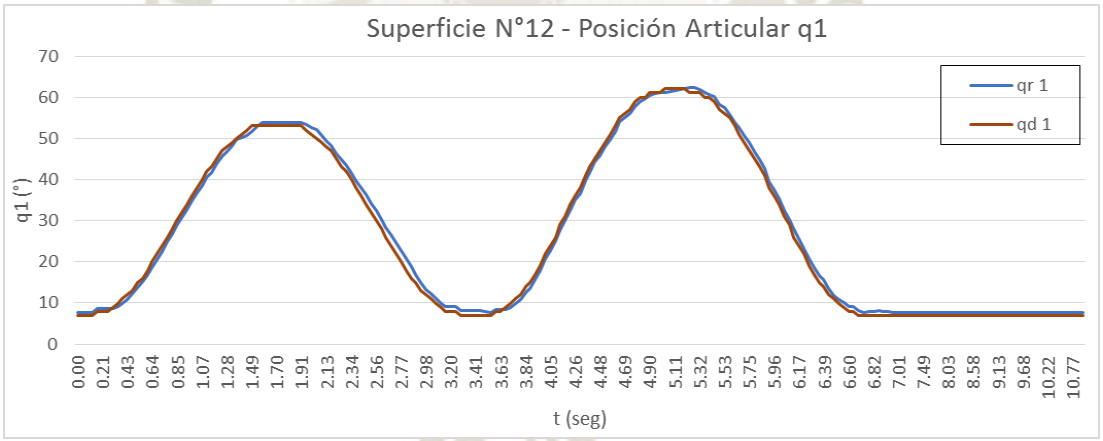
Figura 5.16. Evaluación de trayectorias articulares con la superficie N°10
Fuente: *Elaboración Propia*

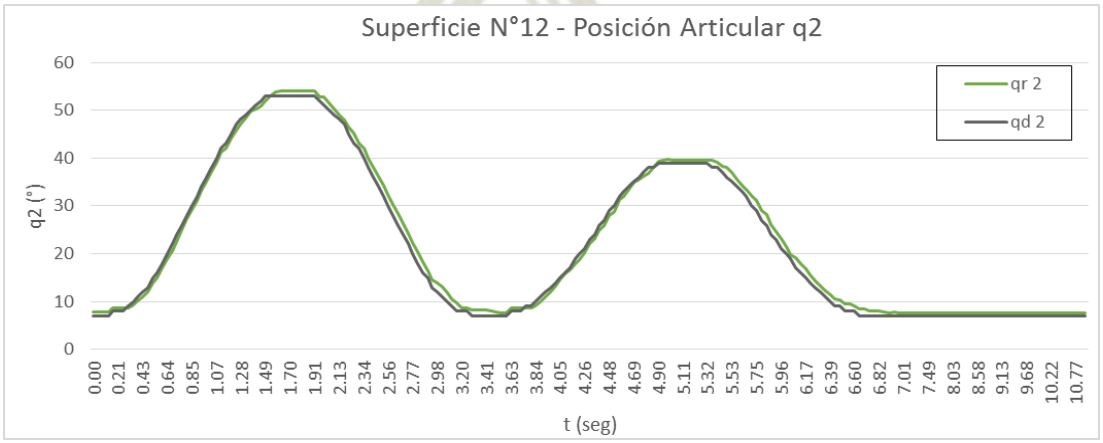
En la Figura 5.16 se observa la evolución de las trayectorias reales comparadas con las trayectorias deseadas; a diferencia de las superficies anteriormente expuestas, en esta se puede apreciar que ya no existen sobre impulsos; no obstante, cada vez que la pendiente de la trayectoria se vuelve negativa el error aumenta y se crea un desfase respecto de la trayectoria original.

Hasta el presente, la superficie que ha demostrado logar resultados favorables es la del esquema sumador con la matriz de 25 reglas difusas propuesta por (C. Sousa & Kaymak, 2002); no obstante, para llegar a tal resultado se tuvo que manipular constantemente los rangos de las funciones de membresía. Respecto a las funciones de membresía de las

variables lingüísticas del error, se consideró “Negativos Grandes” a valores que van desde -15 a -2.5 y en el caso de los “Positivos Grandes” se consideraron a valores que van desde 2.5 a 15; esto debido a que el rango máximo de falla oscila entre 5 a 10. Por otro lado, las variables lingüísticas de la derivada del error contemplan rangos más amplios si son comparadas con las variables lingüísticas del error; ya que si se trabaja con rangos de valores demasiado pequeños se puede crear inestabilidad en las articulaciones del robot.

Teniendo en cuenta todas las consideraciones anteriores y luego de varias pruebas fue posible encontrar una superficie que muestre resultados más que favorables y con una mínima cantidad de error.





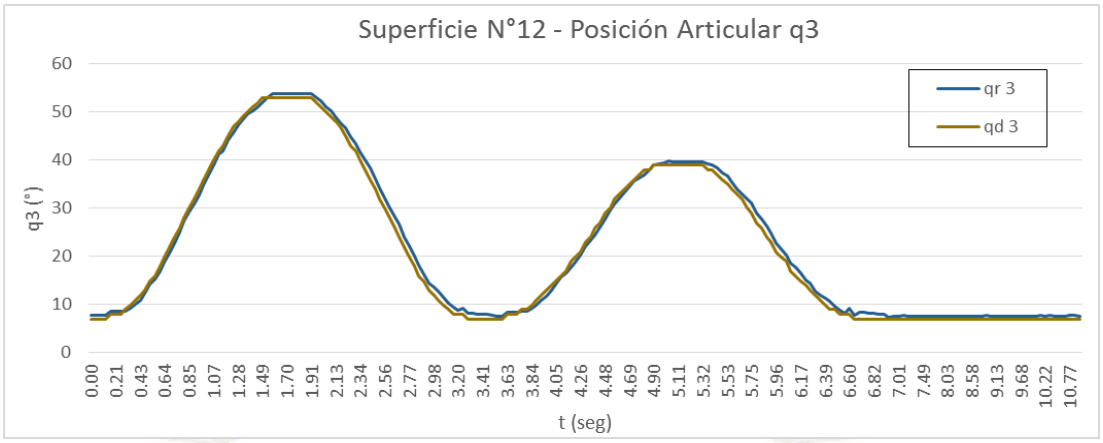


Figura 5.17. Evaluación de trayectorias articulares con la superficie N°12
Fuente: *Elaboración Propia*

En la Figura 5.17 se observa el comportamiento de las trayectorias frente al controlador resultante de la superficie N°12, en esta se puede observar que no existen sobre impulsos, los errores que se presentan a lo largo de la trayectoria se redujeron notablemente y no se pierde la estabilidad del robot; por otro lado, el error en estado estacionario es mínimo y ya no presenta el comportamiento de inestabilidad observado en las anteriores superficies.

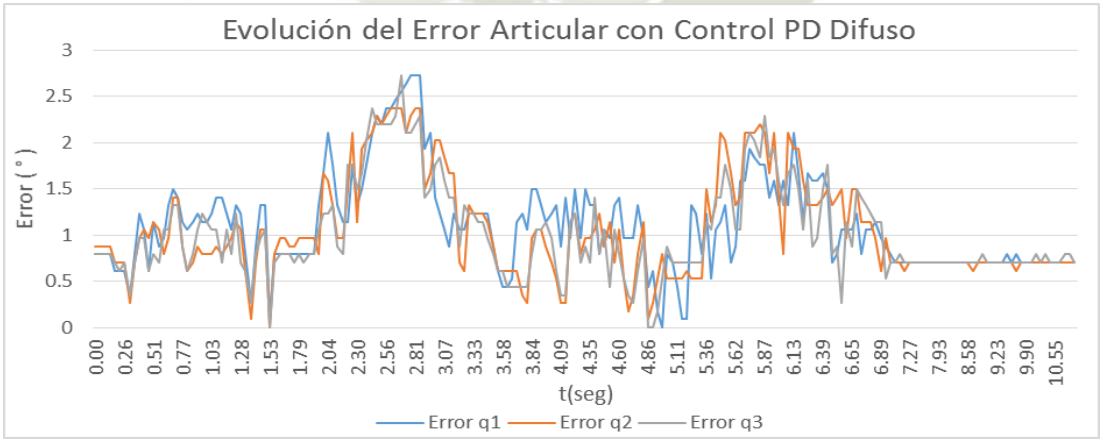


Figura 5.18. Evolución del error articular usando el controlador PD Difuso
Fuente: *Elaboración Propia*

En la Figura 5.18 se observa el paso del error articular frente a la trayectoria planteada, a simple vista hace parecer que se tiene respuestas erráticas; sin embargo, estos errores pueden ser

considerados como mínimos ya que solo varían desde 0 a 2,7° y cuenta un promedio de error de 1,06°. Al comparar estas evidencias con las del robot sin control alguno, se aprecia perfectamente que el error generado por los movimientos del robot paralelo delta ha sido disminuido notablemente y esto también se aprecia en el espacio cartesiano.

En la Figura 5.19 se encuentra la trayectoria cartesiana deseada comparada con la trayectoria real obtenida a partir de la cinemática directa; en esta se puede observar que con la superficie N°12 se mejoró notablemente los resultados si es comparado con la trayectoria cartesiana sin control.

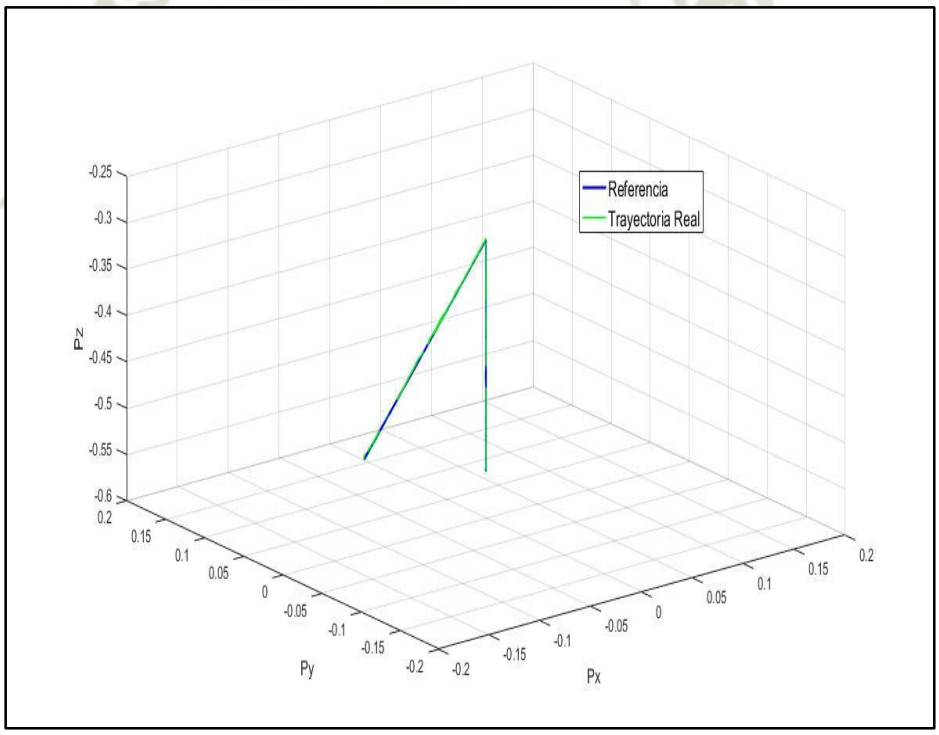


Figura 5.19. Comparación de trayectorias cartesianas usando control PD Difuso
Fuente: *Elaboración Propia.*

A este respecto los resultados que se muestran en la Figura 5.19, son más notables cuando los datos adquiridos de las articulaciones son

llevados mediante software a una gráfica; ya que en la realidad el error que se crea en el movimiento es imperceptible a la vista de los usuarios. Estas evidencias hacen que la superficie N°12 sea seleccionada como el controlador PD Difuso del robot paralelo delta, debido a su buena respuesta frente a la trayectoria planteada y al favorable comportamiento en estado estacionario una vez que el robot ha llegado a alguna posición. También, es necesario recalcar la eficiencia del esquema sumador frente al esquema multiplicador cuando es usado en el robot paralelo delta.

5.4.5. Parámetros del Controlador PD Difuso Seleccionado

En esta sección del capítulo se presentará todos los parámetros del controlador difuso seleccionado que hicieron posible lograr un control óptimo.

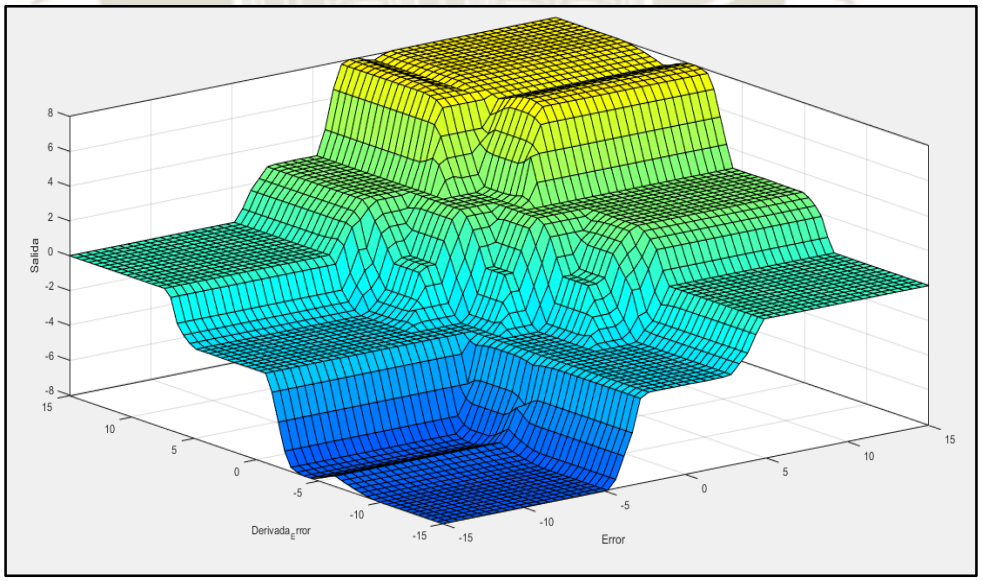


Figura 5.20. Superficie Tridimensional del Controlador PD Difuso elegido
Fuente: *Elaboración Propia.*

En la Figura 5.20 se encuentre la superficie del controlador PD Difuso; dicha superficie responde a la N° 12 de la tabla de superficies

propuestas. En esta se puede observar claramente la simetría que existe respecto a los valores negativos y los valores positivos. Por otro lado; los valores de las variables lingüísticas que se acercan a cero disponer de una forma especial en la cual existen zonas constante y zonas con pendientes positivas o negativas que van de -2 a 2 en la salida.

En cuanto a las funciones de membresía usadas en el controlador, se tiene lo siguiente:

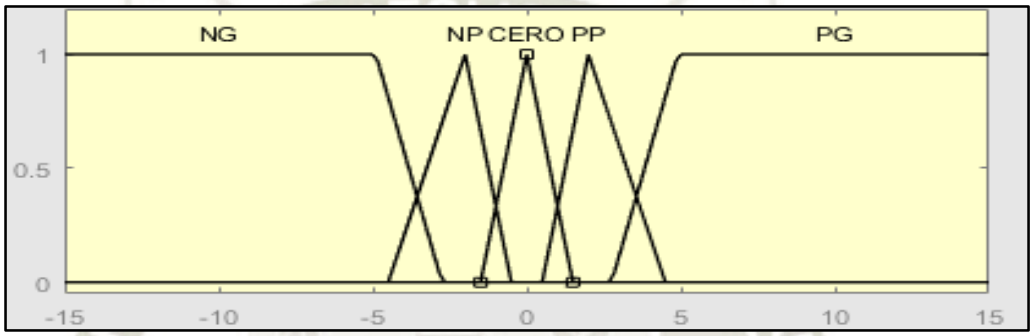


Figura 5.21. Funciones de membresía de la entrada "Error"
Fuente: *Elaboración Propia.*

El universo que presenta la Figura 5.21 representa los rangos de valores en los que existió un error cuando se evaluó el funcionamiento del robot paralelo delta sin control; cabe mencionar que este error está representado en grados y los rangos de cada función son los siguientes:

- **Negativo Grande "NG":** [-15; -15; -4.9; -2.75]
- **Negativo Pequeño "NP":** [-4.5; -2; -0.5]
- **Cero:** [-1.5; 0; 1.5]
- **Positivo Pequeño "PP":** [0.5; 2; 4.5]
- **Positivo Grande "PG":** [2.75; 4.9; 15; 15]

Los rangos de las funciones de membresía fueron elegidos gracias a la experiencia y evaluación de resultados obtenidos de las pruebas anteriores.

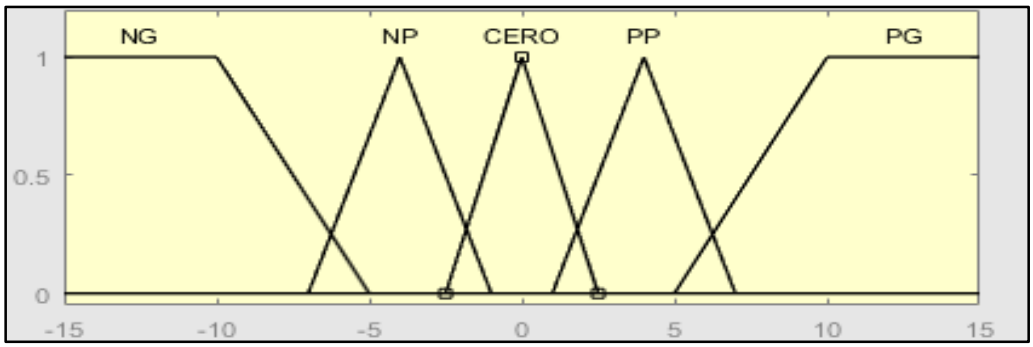


Figura 5.22. Funciones de membresía de la entrada "Derivada de Error"
Fuente: *Elaboración Propia.*

Al igual que en la entrada del error, el universo de la derivada del error contempla el mismo rango que al del error; no obstante, los rangos de las funciones de membresía varían. A lo largo de las pruebas realizadas, cada vez que se tenía rangos reducidos en las variables lingüísticas "Negativo Pequeño" y "Positivo Pequeño", se obtenía sobre impulsos en las trayectorias cuando la pendiente cambiaba de sentido; también, el rango de la variable lingüística "Cero" está obligado a tener zonas donde no se comprometan otras funciones de membresía; esto se hace con el fin de evitar que cada vez que el error se aproxime a cero no se generen sobre impulsos.

Los rangos de las funciones de membresía que contemplan la derivada del error son los siguientes:

- **Negativo Grande "NG":** [-15; -15; -10; -5]
- **Negativo Pequeño "NP":** [-7; -4; -1]
- **Cero:** [-2.5; 0; 2.5]
- **Positivo Pequeño "PP":** [1; 4; 7]
- **Positivo Grande "PG":** [5; 10; 15; 15]

Finalmente, se presentan las funciones de membresía que corresponden a la salida de la señal de control; al igual que las funciones

de membresía ya vistas, todas parten como resultado de diferentes pruebas realizados al robot.

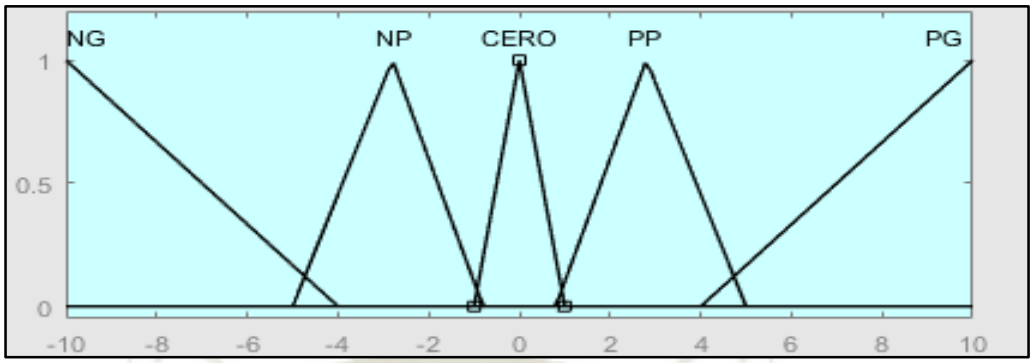


Figura 5.23. Funciones de membresía de la salida "Señal de Control"
Fuente: *Elaboración Propia.*

En la Figura 5.23 se encuentra las cinco funciones de membresía que contemplan la señal de control; a diferencia de las demás funciones ya vistas, la señal de control no cuenta con una función de membresía trapezoidal y cuenta con un universo más pequeño; esto se hace con el fin de evitar que la señal de control sea excesivamente grande y eleve la magnitud de la trayectoria. Respecto a función de membresía de la variable lingüística cero, se le dio un rango minúsculo con el fin de preservar la estabilidad del robot paralelo delta.

Los rangos de las funciones de membresía que contemplan la señal de control son los siguientes:

- **Negativo Grande "NG":** [-10; -10; -4]
- **Negativo Pequeño "NP":** [-5; -2.8; -0.8]
- **Cero:** [-1; 0; 1]
- **Positivo Pequeño "PP":** [0.8; 2.8; 5]
- **Positivo Grande "PG":** [4; 10; 10]

Respecto a las reglas difusas usadas en el controlador seleccionado, se usaron las reglas propuestas por (C. Sousa & Kaymak, 2002) y la implementación se dio tal como muestra la Figura 5.24.

1. If (Error is NG) and (Derivada_Error is NG) then (Salida is NG) (1)
2. If (Error is NG) and (Derivada_Error is NP) then (Salida is NG) (1)
3. If (Error is NG) and (Derivada_Error is CERO) then (Salida is NP) (1)
4. If (Error is NG) and (Derivada_Error is PP) then (Salida is NP) (1)
5. If (Error is NG) and (Derivada_Error is PG) then (Salida is CERO) (1)
6. If (Error is NP) and (Derivada_Error is NG) then (Salida is NP) (1)
7. If (Error is NP) and (Derivada_Error is NP) then (Salida is NP) (1)
8. If (Error is NP) and (Derivada_Error is CERO) then (Salida is NP) (1)
9. If (Error is NP) and (Derivada_Error is PP) then (Salida is CERO) (1)
10. If (Error is NP) and (Derivada_Error is PG) then (Salida is PP) (1)
11. If (Error is CERO) and (Derivada_Error is NG) then (Salida is NP) (1)
12. If (Error is CERO) and (Derivada_Error is NP) then (Salida is NP) (1)
13. If (Error is CERO) and (Derivada_Error is CERO) then (Salida is CERO) (1)
14. If (Error is CERO) and (Derivada_Error is PP) then (Salida is PP) (1)
15. If (Error is CERO) and (Derivada_Error is PG) then (Salida is PP) (1)
16. If (Error is PP) and (Derivada_Error is NG) then (Salida is NP) (1)
17. If (Error is PP) and (Derivada_Error is NP) then (Salida is CERO) (1)
18. If (Error is PP) and (Derivada_Error is CERO) then (Salida is PP) (1)
19. If (Error is PP) and (Derivada_Error is PP) then (Salida is PP) (1)
20. If (Error is PP) and (Derivada_Error is PG) then (Salida is PG) (1)
21. If (Error is PG) and (Derivada_Error is NG) then (Salida is CERO) (1)
22. If (Error is PG) and (Derivada_Error is NP) then (Salida is PP) (1)
23. If (Error is PG) and (Derivada_Error is CERO) then (Salida is PP) (1)
24. If (Error is PG) and (Derivada_Error is PP) then (Salida is PG) (1)
25. If (Error is PG) and (Derivada_Error is PG) then (Salida is PG) (1)

Figura 5.24. Reglas difusas impuestas al controlador
Fuente: *Elaboración Propia.*

Finalmente, con el fin de entender las respuestas dadas por el controlador PD Difuso se muestra las siguientes figuras.

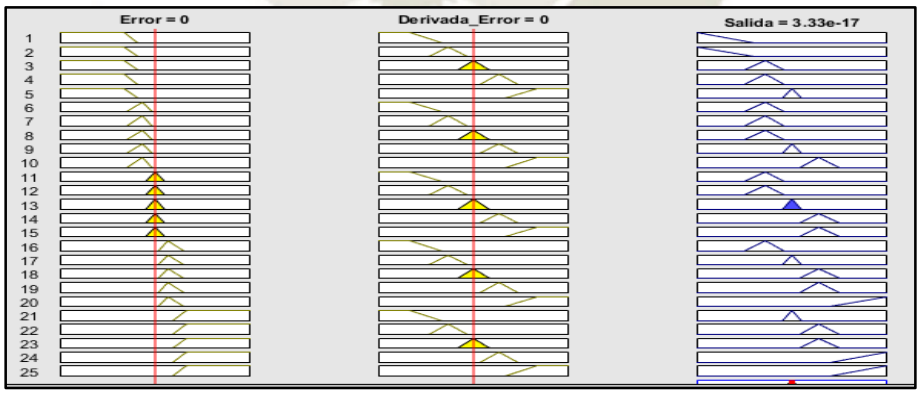


Figura 5.25. Ejemplo de Prueba N° 1 del Controlador PD Difuso
Fuente: *Elaboración Propia.*

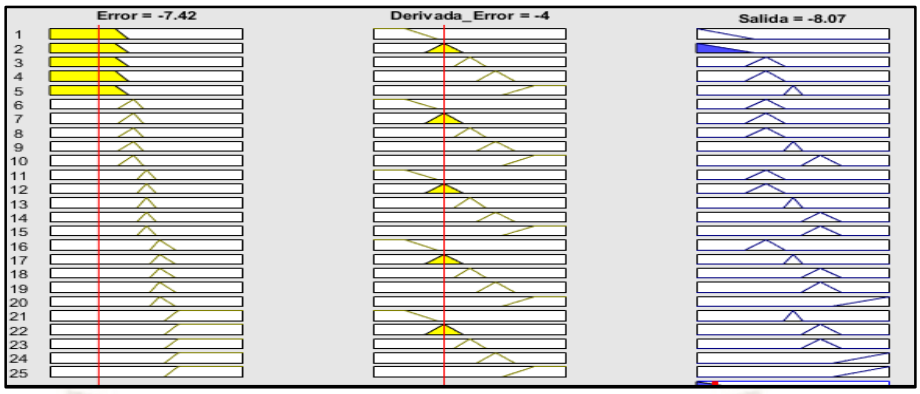


Figura 5.26. Ejemplo de Prueba N° 2 del Controlador PD Difuso
Fuente: *Elaboración Propia.*

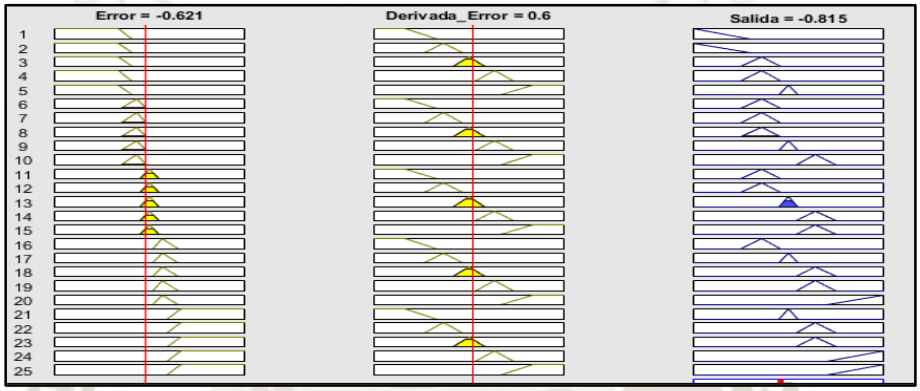


Figura 5.27. Ejemplo de Prueba N° 3 del Controlador PD Difuso
Fuente: *Elaboración Propia.*

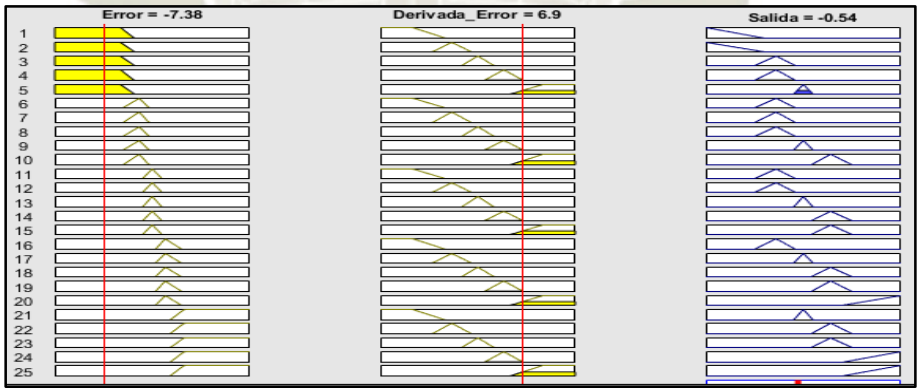


Figura 5.28. Ejemplo de Prueba N° 4 del Controlador PD Difuso
Fuente: *Elaboración Propia.*

Las figuras presentadas anteriormente demuestran el funcionamiento del controlador PD Difuso seleccionado frente a diferentes variables. En la Figura 5.25 se observa que cuando el error es 0 y la derivada del error es 0 la señal de control adquiere un valor tan mínimo que es considerado

cero; luego, en la Figura 5.26 se aprecia que al tener un error considerado como grande y con una derivada de error negativa pequeña el resultado de la señal de control es mayor a las anteriores con el fin de compensar el error creado; a continuación, en la Figura 5.27 se tiene un error muy pequeño con una derivada igual de pequeña pero de signo diferente logrando un resultado pequeño esto con el fin de no crear inestabilidades cada vez que el error se aproxima más a 0; finalmente en la Figura 5.28 se tiene un error considerado como negativo grande y una derivada de error positiva grande; al tener ambas entradas de diferentes signos el resultado que se obtiene es pequeño; a causa de que se conoce la tendencia del error y a fin de evitar sobre impulsos se obtiene un resultado pequeño y esto se demuestra en la matriz de reglas difusas.

Capítulo VI:

6. PRUEBAS Y RESULTADOS

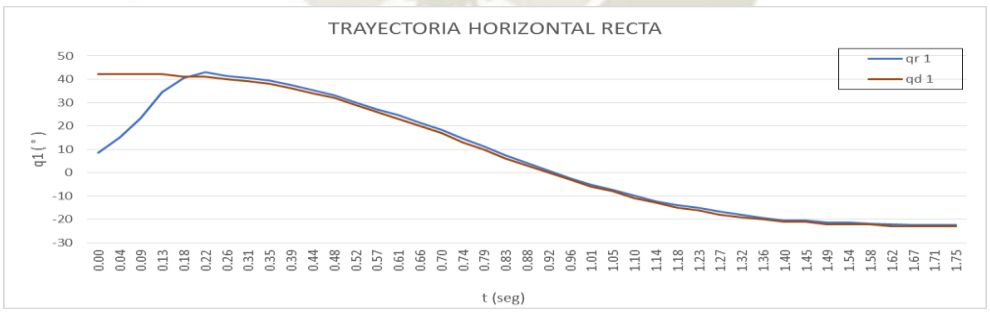
En el presente capítulo se demostrara el funcionamiento del robot paralelo delta con todas las funciones propuestas completas; es decir, comunicación entre dispositivos, programa de la trayectoria, uso manual con joystick, pruebas con el actuador final, movilidad del robot, control difuso e interfaz gráfica. Así mismo, se evaluara la respuesta del controlador PD difuso con mayor claridad frente a distintas señales de referencia.

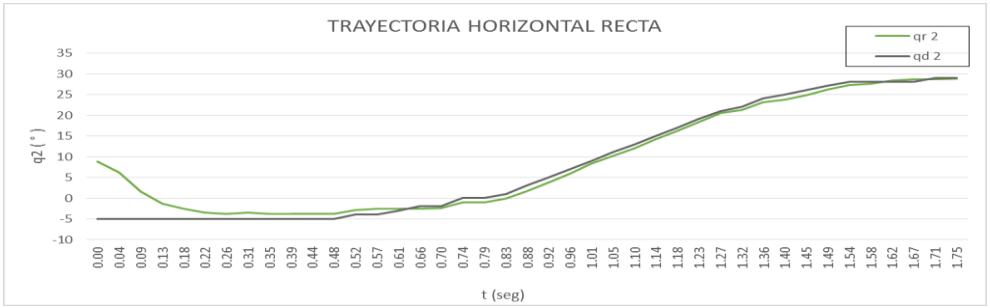
6.1. Evaluación de Trayectorias

6.1.1. Prueba Número Uno

En esta prueba se analizara el desempeño de controlador PD Difuso implementando en el capítulo anterior frente a una trayectoria en línea recta horizontal con valores que van desde P_i (-150 mm; 0 mm; - 280 mm) hasta P_F (150 mm; 0 mm; -280 mm).

Atendiendo a estas consideraciones se obtuvo los siguientes resultados:





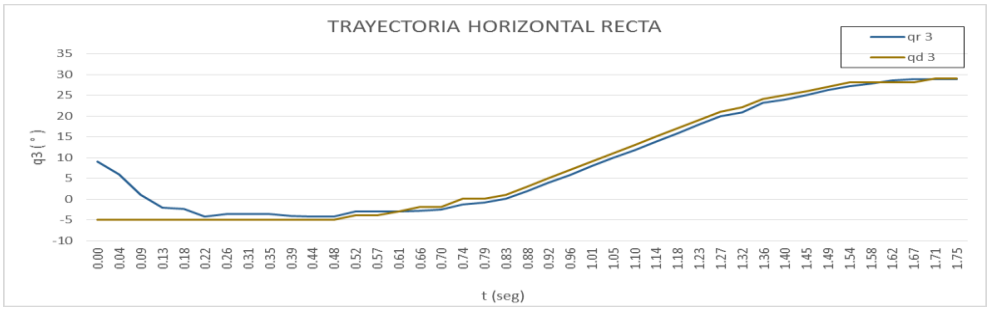


Figura 6.1. Trayectoria Articular Deseada vs Trayectoria Articular Real - Prueba N° 1
Fuente: *Elaboración Propia.*

En la Figura 6.1 se encuentra la evolución de los valores articular frente a una trayectoria planteada; al inicio de las grafica se aprecia una diferencia considerable entre las trayectorias mostradas, esto debido a que al iniciar la prueba el robot se encontraba en una posición de reposo; luego, el error se reduce notablemente ocasionando que los valores articulares reales lleguen a ser muy cercanos a los valores de la trayectoria deseada. De igual manera, siempre existe una diferencia entre las trayectorias por lo que el error nunca es cero.

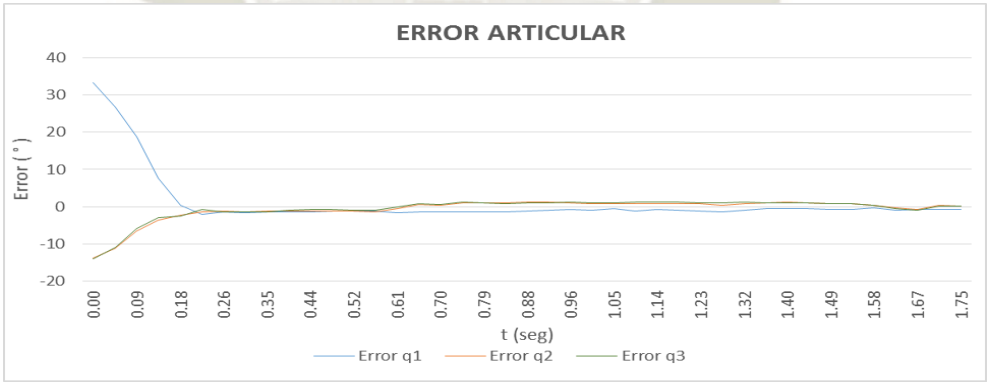


Figura 6.2. Evolución del error articular – Prueba N° 1
Fuente: *Elaboración Propia.*

Por otro lado, al usar resultados reales obtenidos del robot paralelo delta en físico es imprescindible dotar de señales reales al robot; dichas señales no son continuas puesto que los actuadores trabajan con señales discretas, de esta manera las trayectorias de referencia

enviadas al robot paralelo delta cuenta con un periodo de muestreo de 50 milisegundos.

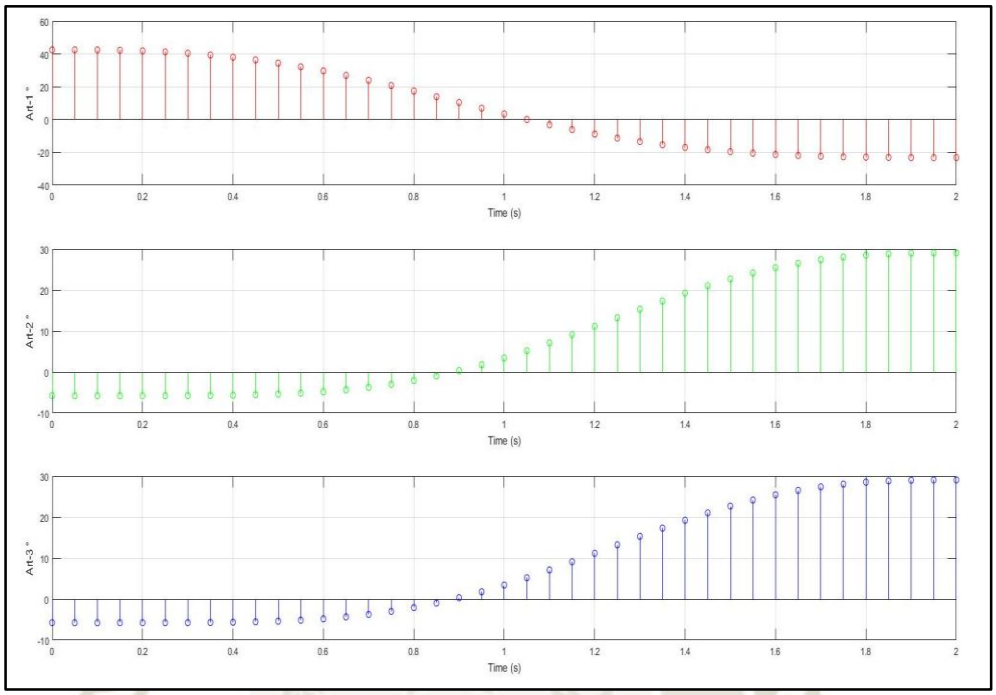


Figura 6.3. Trayectoria articular de referencia muestreada – Prueba N°1
Fuente: *Elaboración Propia.*

Finalmente, con el fin de evaluar el desempeño en el espacio cartesiano se evalúan las trayectorias articulares reales obtenidas con el uso de la cinemática directa, obteniendo como resultado la Figura 6.4.

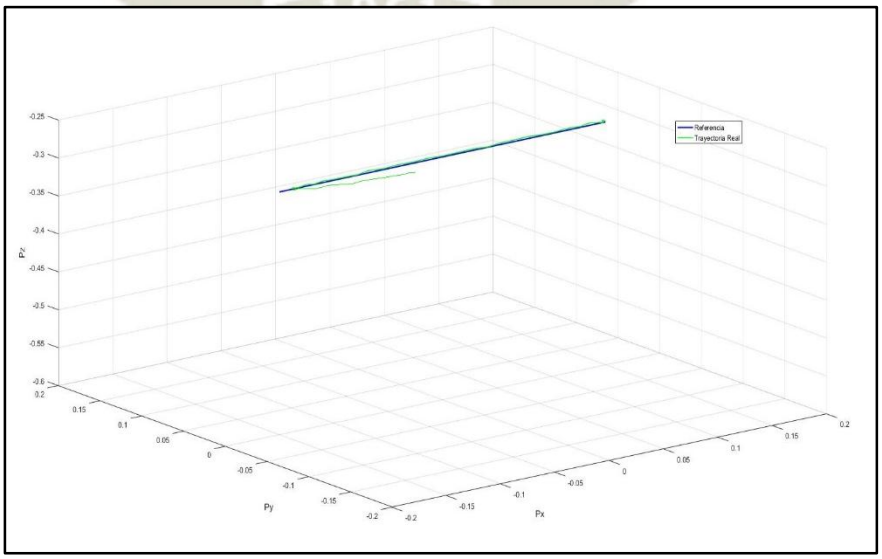


Figura 6.4. Trayectoria Cartesiana Deseada vs Trayectoria Cartesiana Real - Prueba N° 1
Fuente: *Elaboración Propia.*

6.1.2. Prueba Número Dos

En esta prueba se analizara el desempeño de los movimientos articulares y cartesianos del robot paralelo delta usando el controlador PD Difuso propuesto en el capítulo anterior frente a una trayectoria “Pick and Place” usualmente usada para la clasificación de objetos. Este constara de tres posiciones; P_i (150 mm; - 63 mm; - 465 mm), con una posición intermedia P_M (0 mm; 0 mm; -300 mm), hasta P_F (120 mm; 0 mm; -500 mm).

Atendiendo a estas consideraciones se obtuvo los siguientes resultados:

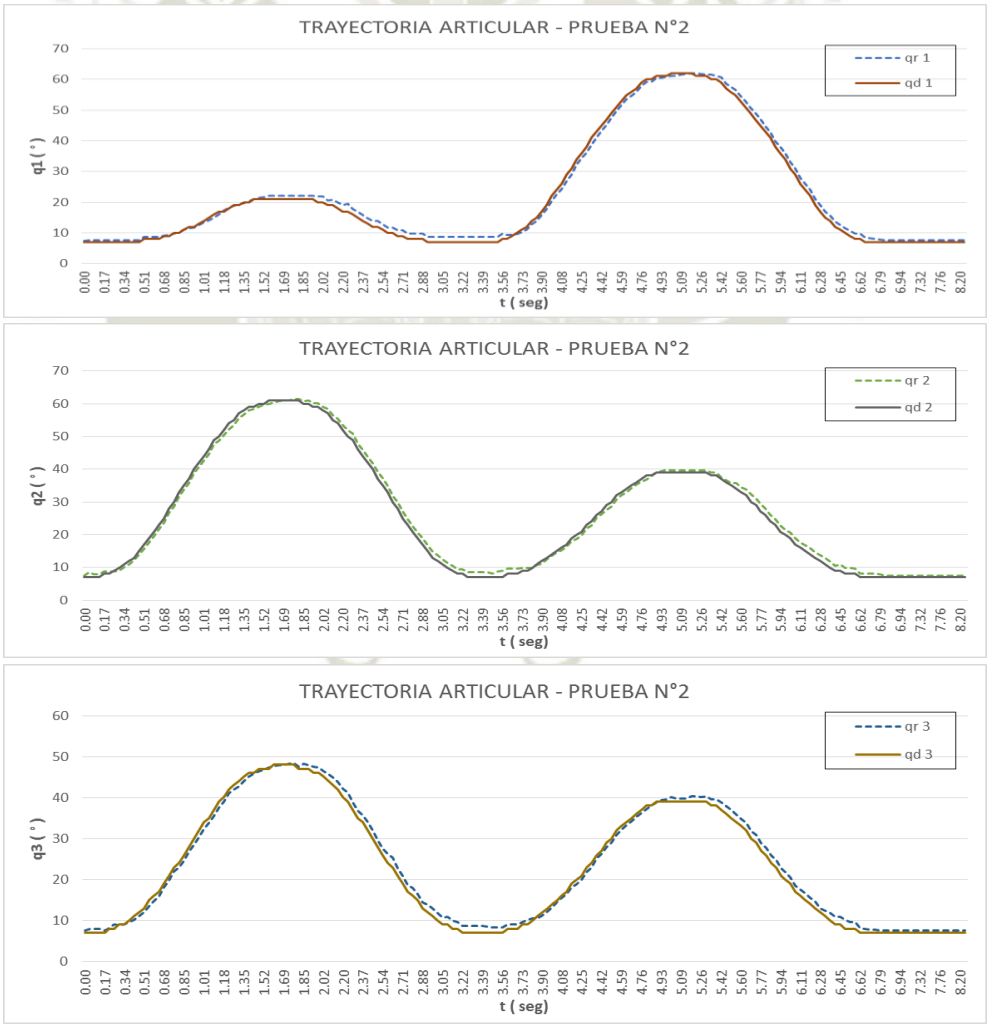


Figura 6.5. Trayectoria Articular Deseada vs Trayectoria Articular Real - Prueba N° 2
Fuente: *Elaboración Propia.*

En la Figura 6.5 se observa la evolución de la trayectoria articular real frente a la trayectoria de referencia; se aprecia que la diferencia entre la señal deseada y la señal de referencia es mínima, no existe sobre impulsos a lo largo de toda la señal; no obstante, a partir del segundo 1.69 el error empieza a incrementar y esto es causado a que cuando el robot paralelo delta se encuentra ejerciendo movimientos de subida requiere más torque; aun así el error generado es mínimo y no afecta en grandes rasgos a la trayectoria deseada.

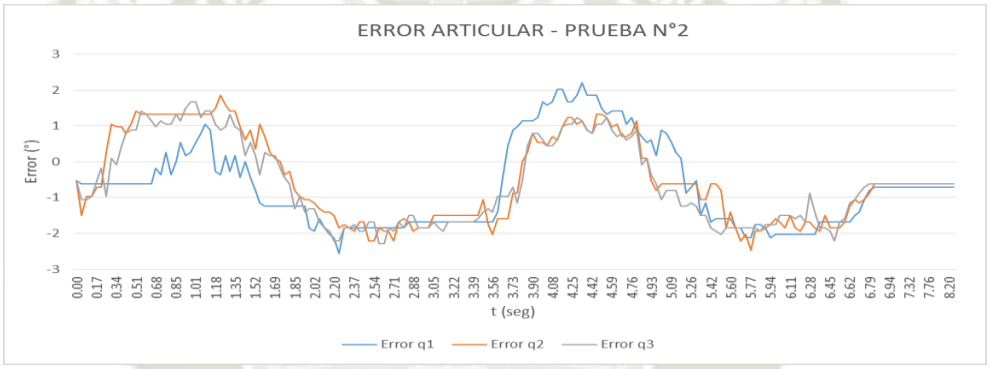


Figura 6.6. Evolución del error articular – Prueba N° 2
Fuente: *Elaboración Propia.*

La Figura 6.6 muestra la variación del error creador por la diferencia de trayectoria de referencia frente a la trayectoria real lo largo del tiempo, en esta se aprecia que alcanza picos de hasta 2.3°; así mismo, una vez que el robot se encuentra en estado de reposo el error se mantiene constante y es de -0.6°; también, se observa que por algunos momentos el error generado por la articulación 1 es mayor que las demás articulaciones.

Finalmente, se muestra en la Figura 6.7 la trayectoria “Pick and Place” de referencia frente a la trayectoria obtenida a partir de la cinemática directa.

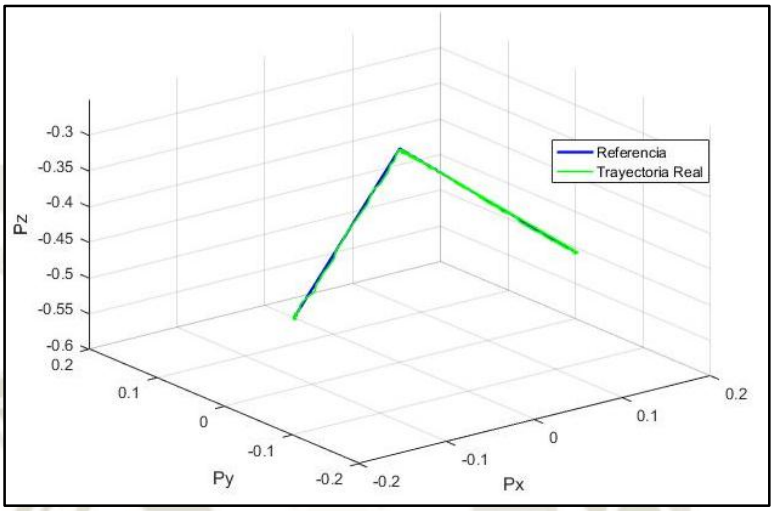
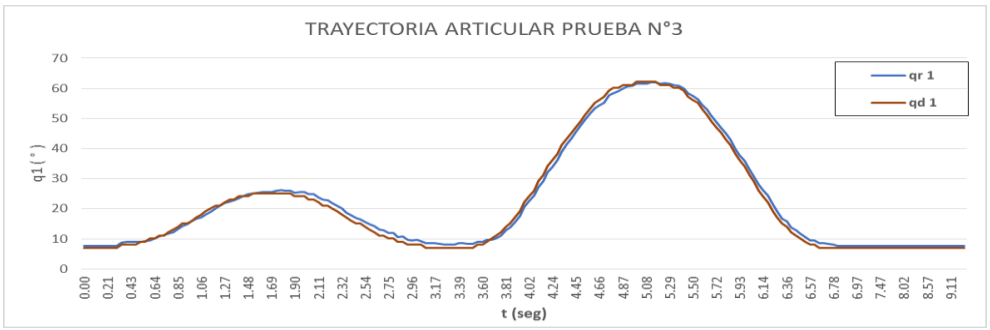


Figura 6.7. Trayectoria Cartesiana Deseada vs Trayectoria Cartesiana Real - Prueba N° 2
Fuente: *Elaboración Propia.*

6.1.3. Prueba Número Tres

A fin de seguir emulando la clasificación de objetos se analizara el desempeño de los movimientos articulares y cartesianos del robot paralelo delta con las siguientes posiciones; P_i (80 mm; 136 mm; - 428 mm), con una posición intermedia P_M (0 mm; 0 mm; -300 mm), hasta P_F (120 mm; 0 mm; -500 mm).

Atendiendo a estas consideraciones se obtuvo los siguientes resultados:



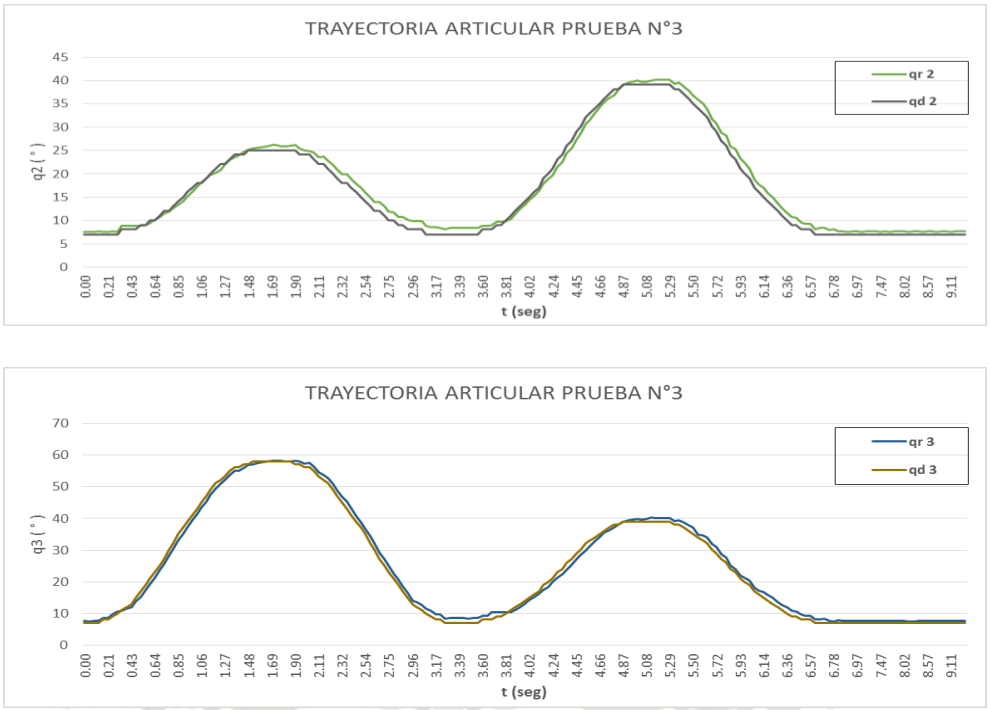


Figura 6.8. Trayectoria Articular Deseada vs Trayectoria Articular Real - Prueba N° 3
Fuente: *Elaboración Propia.*

En la Figura 6.8 se encuentran los gráficos que evalúan el desempeño de los movimientos del robot frente a una trayectoria “Pick and Place”; al igual que en los casos presentados anteriormente se observa que con el uso del controlador PD – Difuso propuesto en el capítulo anterior el error es disminuido notablemente y no se crean sobre impulsos; no obstante, la diferencia entre la trayectoria real frente a la trayectoria deseada aun es notable a lo largo de todo su recorrido puesto que el error nunca se elimina por completo, esto se muestra con más detalle en la Figura 6.9.

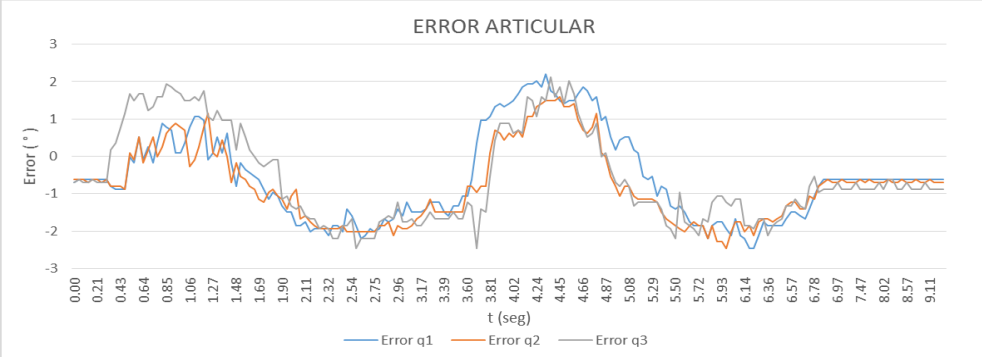


Figura 6.9. Evolución del error articular – Prueba N° 3
Fuente: *Elaboración Propia.*

En la Figura 6.9, se observa la evolución del error frente a la trayectoria planteada y en este se aprecia que el error nunca llega a ser cero; no obstante, el error es tan pequeño que en algunos momentos puede ser despreciado.

Finalmente, la Figura 6.10 muestra la trayectoria la cartesiana deseada frente a la trayectoria cartesiana real obtenida a partir de la cinemática directa.

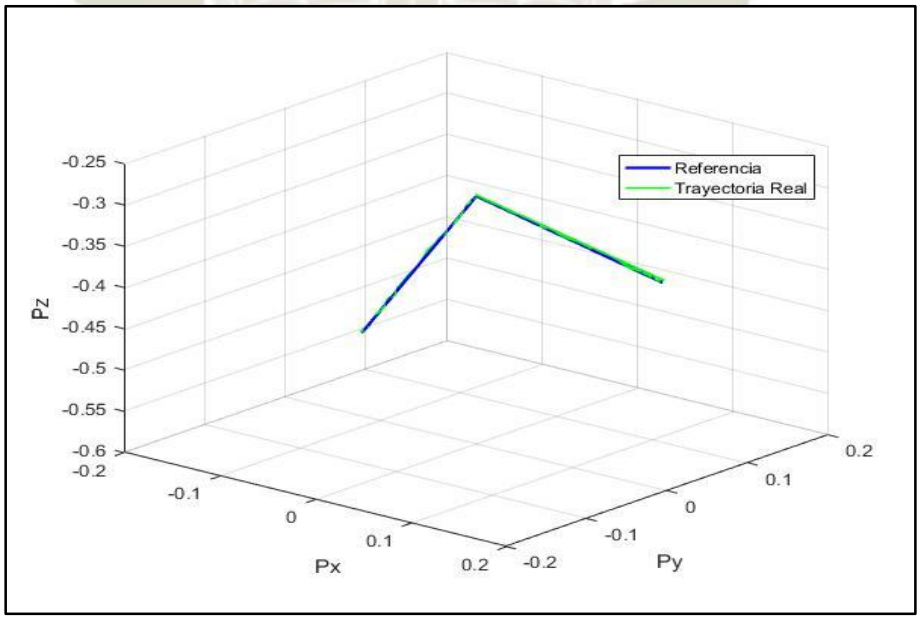
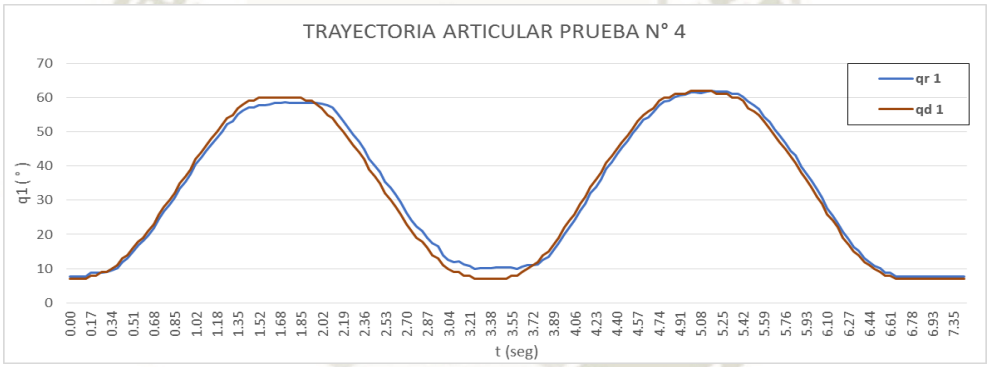


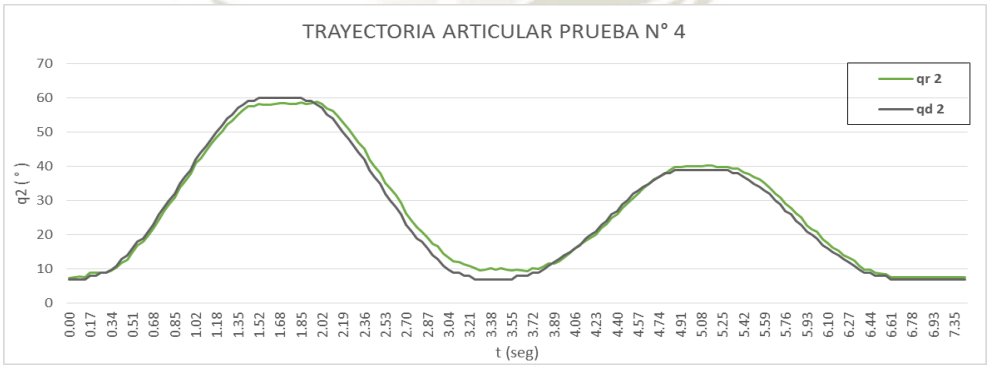
Figura 6.10. Trayectoria Cartesiana Deseada vs Trayectoria Cartesiana Real - Prueba N° 3
Fuente: *Elaboración Propia.*

6.1.3. Prueba Número Cuatro

Al momento de diseñar el robot paralelo se consideró que este podría ejercer movimientos con una masa en el efector final de 300 gramos; basándonos en lo anteriormente mencionado, esta prueba se realizada con un objeto de 300 gramos que será llevado de un lugar a otro con la trayectoria “Pick and Place”. Para realizar el movimiento se consideró las siguientes posiciones P_i (0 mm; 0 mm; - 580 mm), con una posición intermedia P_M (0 mm; 0 mm; -300 mm), hasta P_F (120 mm; 0 mm; -500 mm).

Atendiendo a estas consideraciones se obtuvo los siguientes resultados:





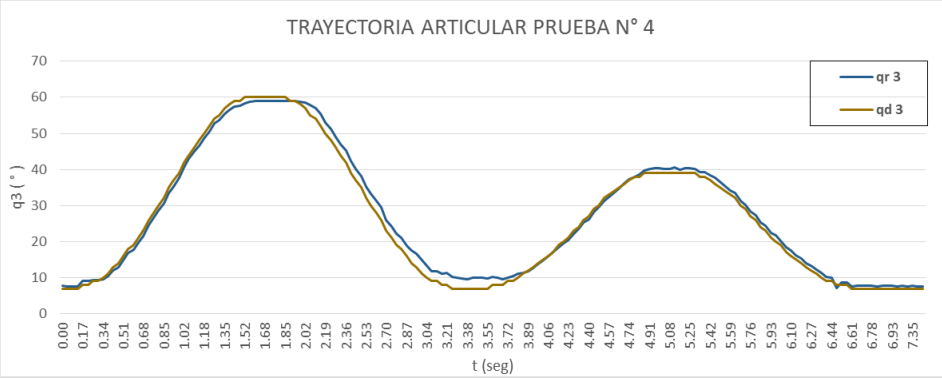


Figura 6.11. Trayectoria Articular Deseada vs Trayectoria Articular Real - Prueba N° 4
Fuente: *Elaboración Propia.*

En la Figura 6.11 se observa la evolución de la trayectoria articular real vs la trayectoria articular deseada; a diferencia de las otras pruebas realizadas anteriormente, la prueba cuatro presenta un error más pronunciado sobre todo en los movimientos generados a partir del segundo 1.35 hasta el 3.89. Esto se debe a que en esos instante de tiempo el robot paralelo delta levanta el objeto hasta la posición intermedia ocasionando que el error aumente considerablemente, aun así se cumplió el propósito de transportarlo hacia la posición final con un error mínimo.

Con el fin de comprender cuanto ha aumentado el error cuando se considera una carga de 300 gramos en el efector final, se presenta la Figura 6.12; en esta se puede observar que el error articular sufre un incremento de hasta casi 4°, aun así este resultado es mucho mejor si es comparado a los movimientos del robot cuando no posee control alguno.

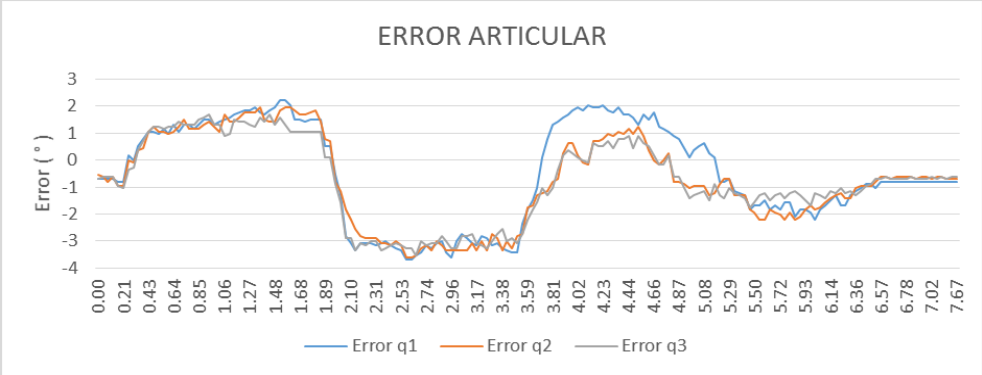


Figura 6.12. Evolución del error articular – Prueba N° 4
Fuente: *Elaboración Propia.*

Finalmente, la Figura 6.13 presenta una comparación entre la trayectoria cartesiana deseada frente a la trayectoria cartesiana real obtenida a partir de la cinemática directa.

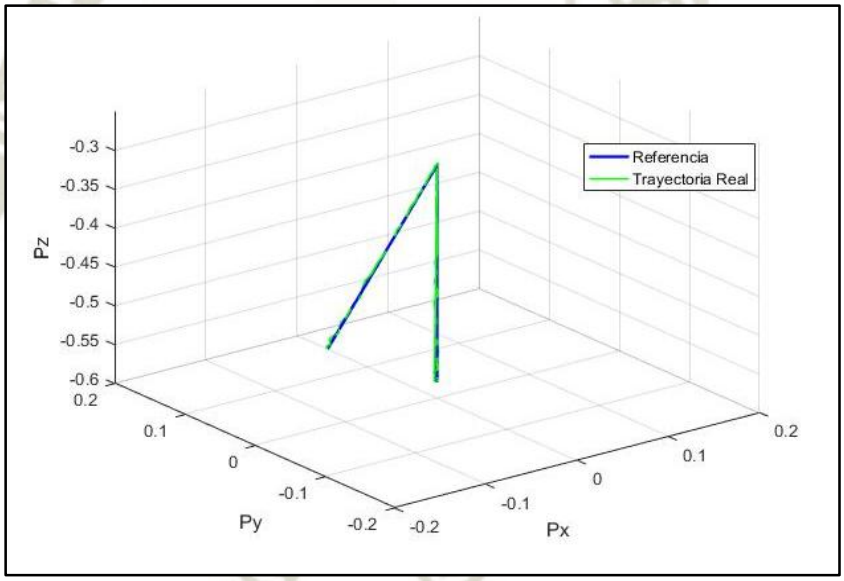


Figura 6.13. Evolución del error articular – Prueba N° 4
Fuente: *Elaboración Propia.*

En la Figura 6.13 se observa que el movimiento cartesiano del robot con una carga en el efector final crea algunas discontinuidades a lo largo de la trayectoria planteada; sin embargo, estas fluctuaciones son imperceptibles a simple vista.

6.2. Pruebas de la Interfaz Gráfica

En esta sección del apartado se pondrá a prueba la operatividad de interfaz gráfica creada a partir del software Visual Studio; así mismo, se evaluaran las distintas funciones incorporadas.

Antes de evidenciar las pruebas realizadas es necesario hacer énfasis que con el motivo de mejorar el aspecto visual y el orden del robot, se añadió un tablero de control que contiene todos los elementos eléctricos del robot tales como; fuente de alimentación, bomba de vació, HUB de conexión USB y la placa impresa.

Figura 6.14. Tablero de Control Implementado
Fuente: *Elaboración Propia.*

Así mismo, se añadió canaletas y mangueras corrugadas para evitar que el cableado proveniente del tablero de control quede expuesto al medio.

Figura 6.15. Canaletas para protección del cableado
Fuente: *Elaboración Propia.*

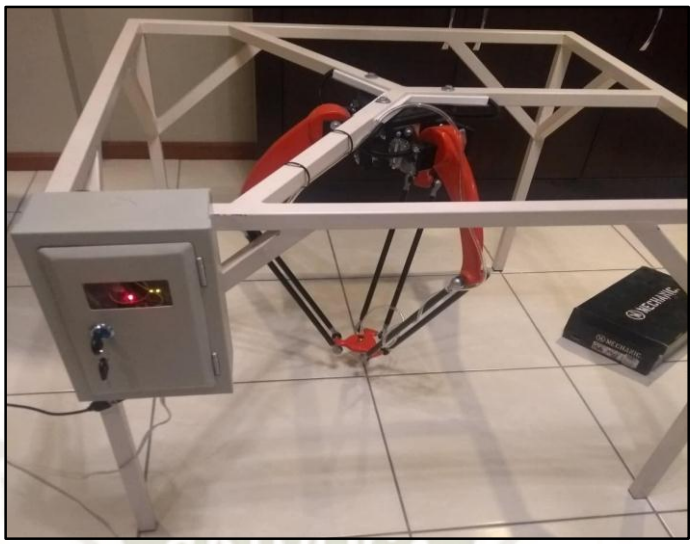


Figura 6.16. Robot paralelo delta implementado con tablero de control
Fuente: *Elaboración Propia.*

El propósito principal de la interfaz gráfica es el de manipular el robot y brindar los datos necesarios para que el usuario observe en todo momento los parámetros relacionados con el robot. Teniendo en cuenta lo anteriormente mencionado se presentan las siguientes figuras con el fin de evidenciar el funcionamiento de la interfaz.

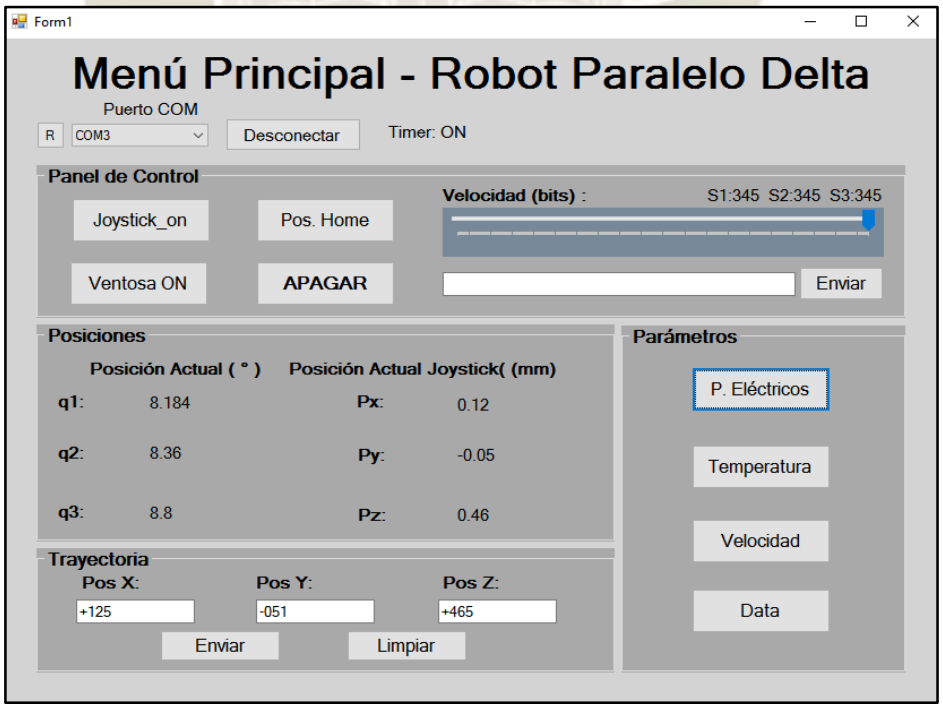


Figura 6.17. Interfaz Gráfica en funcionamiento
Fuente: *Elaboración Propia.*

En la Figura 6.17 se encuentra la interfaz gráfica del robot paralelo delta, en esta se observa la realimentación de datos de la posición articular actual del robot. Si se hiciera clic en alguna de las funciones dadas, el robot seguiría las instrucciones tal cual se hacen desde el menú.

Por otro lado, se presentan las evidencias de los datos adquiridos cuando se ingresa a la opción de parámetros.

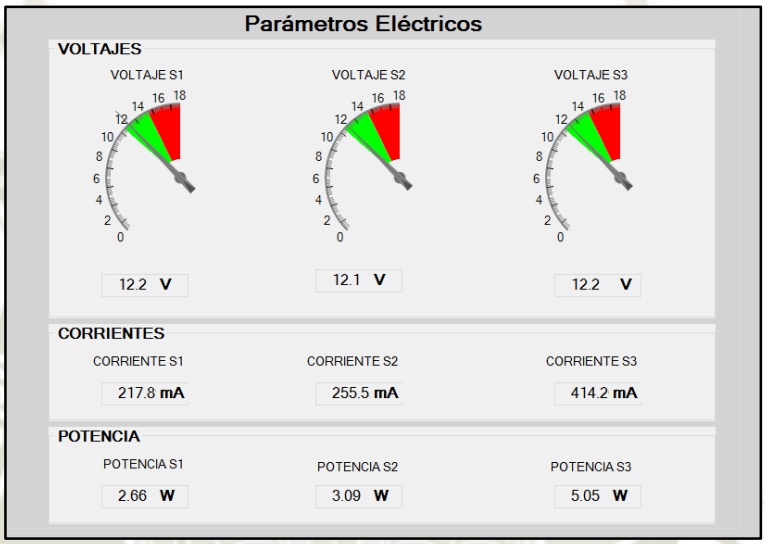


Figura 6.18. Menú de Parámetros Eléctricos
Fuente: *Elaboración Propia.*

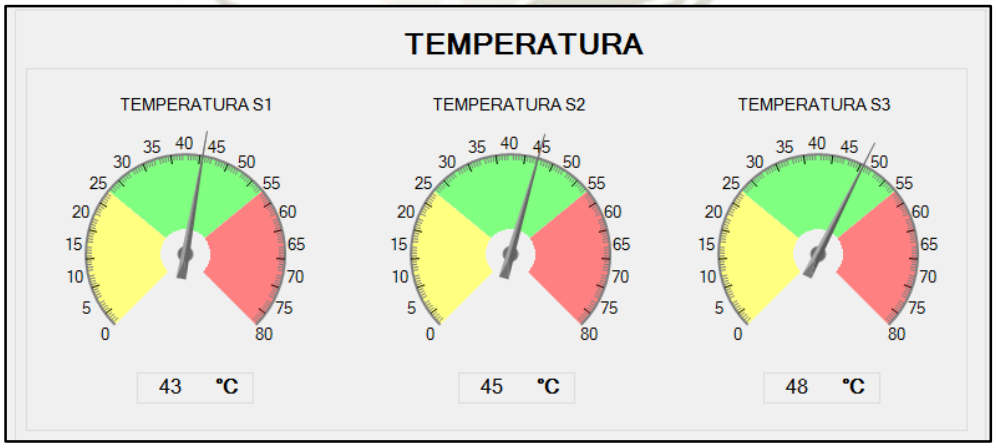


Figura 6.19. Menú de Parámetros de Temperatura
Fuente: *Elaboración Propia.*

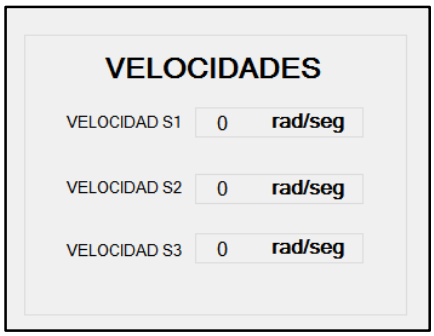


Figura 6.20. Menú de Parámetros de Velocidades
Fuente: *Elaboración Propia.*

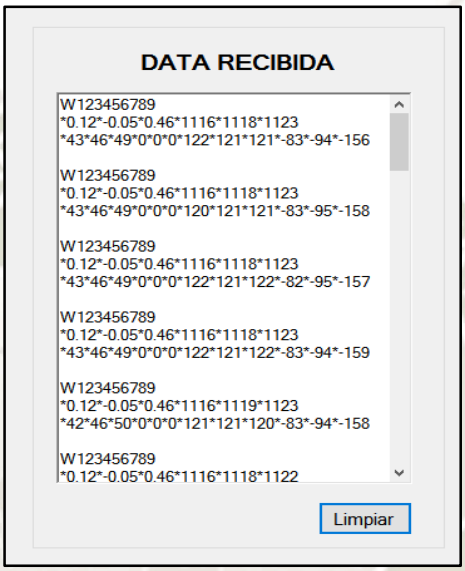


Figura 6.21. Menú de Datos Recibidos
Fuente: *Elaboración Propia.*

Respecto al uso de Joystick, se muestra las siguientes pruebas:

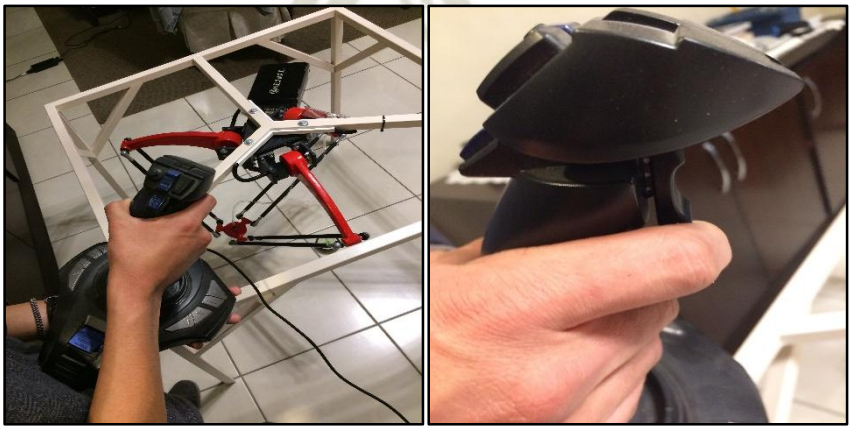


Figura 6.22. Manipulación con Joystick
Fuente: *Elaboración Propia.*

La Figura 6.22 muestra el funcionamiento del robot cuando se opera con el joystick; así mismo, para añadirle un valor agregado a la manipulación de objetos, cada vez que se presione el gatillo del joystick se encenderá la bomba de vació y por ende se podrá levantar objetos.

Por último, con el fin de evaluar la capacidad de succión de la bomba de vacío se ponen a prueba diferentes objetos que el robot tendría que levantar considerando los parámetros plantados en el diseño.

Atendiendo a estas consideraciones se obtuvo los siguientes resultados:

Figura 6.23. Pruebas de succión
Fuente: *Elaboración Propia.*

CONCLUSIONES

- Se diseñó e implemento un prototipo experimental de un manipulador robótico de configuración paralela delta de tres GDL que imita el movimiento necesario para ejecutar la acción de clasificar objetos, controlado con una estrategia de control difuso.
- Siguiendo el procedimiento de diseño propuesto, se obtuvo un análisis completo sobre la cinemática y dinámica del robot; así mismo, a partir de los análisis mencionados se sentó las bases para un diseño mecánico, diseño electrónico y diseño de software.
- Al llevar a cabo la fabricación estructural del robot paralelo delta y siguiendo las consideraciones del diseño mecánico se empleó la impresión 3D; desafortunadamente, al no obtener piezas completamente solidas se optó por reforzar las piezas con fibra de vidrio.
- Se desarrolló un controlador difuso tipo PD que cumple con el objetivo propuesto, obteniendo resultados sobresalientes frente a diferentes señales de referencia; sin embargo, se observó que siempre persiste un error y este puede llegar a ser muy cercano a cero pero nunca cero.
- Con el fin de manipular diversos objetos, se implementó una ventosa de succión como efector final.
- Se desarrolló una interfaz gráfica que permite la interacción con el robot paralelo delta en tiempo real, de una manera simple, agradable, rápida y que brinda los parámetros más resaltantes de los actuadores.
- Con el fin de añadirle peso al proyecto de tesis, todos los programas fueron creados a partir de lenguaje estructurado y se usaron dos controladores uno en configuración de maestro y el otro en configuración esclavo.

OBSERVACIONES

- Al tener una configuración robótica poco conocida, se tuvo que crear distintos programas que simulan el comportamiento cinemático y dinámico del robot paralelo delta en el software MATLAB, con el fin de diseñar y dimensionar los elementos que componen el robot; también, para evaluar el movimiento del robot.
- La bomba de vació usada en el presente proyecto de tesis no cuenta con la potencia necesaria para elevar objetos mayores a 300 gramos, pese a que el robot paralelo delta si cuenta con la capacidad para hacerlo.
- Al llevar a cabo el reforzamiento con fibra de vidrio de las piezas que componen el robot paralelo delta, se alteró ligeramente las dimensiones iniciales; ocasionado así que en algunas zonas se creen desalineamiento.
- La calibración de los elementos terminales del robot paralelo delta tales como los actuadores, ejes, brazos y efecto final se torna bastante compleja y tedioso debido a que se posee una cadena cinemática cerrada.
- La posición cartesiana brindada por la interfaz gráfica solo funciona cuando se hace uso del joystick y no en otros casos, esto se debe a que la cinemática directa del robot paralelo delta es muy compleja, de naturaleza no lineal y posee un elevado coste computacional.
- El control difuso implementado está basado en corregir las posiciones articulares del robot; por otro lado el control de velocidad es llevado acabo netamente por los servomotores Dynamixel; esto se debe a que emplear más controladores difusos en paralelo para otra función satura la memoria interna del controlador empleado.

- En la interfaz solo se ingresa la posición inicial del robot, mas no la posición final; si se desea cambiar la posición final a la cual se dirigirá el robot, se debe calcular una trayectoria con el programa del anexo 09.
- La sensibilidad que tiene el robot al utilizarlo con el modo joystick es alta. debido al incremento cartesiano que este modo de control posee, si se varía este parámetro el robot ya no podría ser controlado de manera adecuada.
- Para usar el joystick debe cerciorarse de que este se encuentre conectado a la PC; de no ser así, automáticamente se pierde el control del robot y puede ocasionar serios daños a la estructura y los actuadores.
- Las posiciones “Z” ingresadas en la interfaz respecto a la trayectoria deben ser consideradas positivas, en el caso de que no se pueda alcanzar la posición, se recibirá un mensaje indicando una posición no valida.
- Considerar un espacio de trabajo en forma cilíndrica ya que las rotulas limitan por mucho el movimiento del robot.

RECOMENDACIONES

- Se recomienda cambiar las articulaciones esféricas del robot paralelo delta ya que estas poseen un rango limitado de movimiento y conlleva que el espacio de trabajo del robot sea reducido notablemente.
- Con el fin de obtener mejores resultados del controlador PD Difuso se sugiere usar funciones de membresía no lineales como por ejemplo la función Gaussiana.
- Si se quiere lograr movimientos más óptimos y con más rapidez se recomienda cambiar el controlador Maestro por un controlador que posea una mayor memoria RAM, ya que los vectores dinámicos son muy limitados en los procesadores Arduino.
- Con el fin de movilizar elementos más pesados se recomienda cambiar la bomba de vacío por una que posea mayor potencia.

BIBLIOGRAFÍA

123Bearing. (2017). *123Bearing*. Obtenido de https://www.123bearing.com/rod-end-SQ06-C-RS.php

3D Market. (2016). Obtenido de https://www.3dmarket.mx/articulos/impresoras-3d-en-mexico/attachment/proceso-impresion-3d/

ABB. (2018). Obtenido de http://www.directindustry.es/prod/abb-robotics/product-30265-169123.html

Adarme Jaimes, W., Arango Serna, M., & Cogollo Flórez, J. (2012). Medición del desempeño para cadenas de abastecimiento en ambientes de imprecisión usando lógica difusa. *Scielo*.

AFEL. (2018). Obtenido de https://afel.cl/producto/fuente-de-poder-12v-20a-240w/

Ake Chuc, J. A. (2014). Estructura Mecanica de un Robot. *Estructura Mecanica de un Robot*.

Ake, J. A. (2014). Estructura Mecanica de un Robot. 3.

Alves, A., Lemos, M., S. Kridi , D., & Leal, K. (2018). *Github*. Obtenido de Github: https://github.com/zerokol/eFLL

Andrew, N. (2015). *Handbook of Manufacturing Engineering and Technology.* Springer.

Arduino. (2018). *Arduino .* Obtenido de https://www.arduino.cc/en/Main/Software

ttps://ardushop.ro/en/home/222-transistor-tip31c.html

Aurova. (2017). *Cinemática Directa Concepto teórico* . Obtenido de http://www.aurova.ua.es/robolab/EJS3/RRP_Intro_2.html

Autodesk. (2018). *Autodesk Inventor.* Obtenido de https://www.autodesk.com/education/free-software/inventor-professional

Ballané, V. (2017). Robotok Irányítása - Bevezető. *ANZDOC*.

Banda, H. (2014). *Inteligencia Artificial*. Obtenido de https://www.researchgate.net/publication/262487459_Inteligencia_Artificial_Principios_y_Aplicaciones

Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil, R. (2007). *Fundamentos de Robotica.* McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.

Boer, C., Molinari-Tosatti, L., & Smith, K. (1999). Parallel Kinematic Machines: Theoretical Aspects and Industrial Requirements.

Brico Geek. (2017). Obtenido de https://tienda.bricogeek.com/motores-dc/989-bomba-de-vacio-12v.html

Budynas, R. G., & Nisbett, J. K. (2008). Diseño en Ingeniería Mecánica de Shigley. Mc Graw-Hill.

C. Sousa, J., & Kaymak, U. (2002). *Fuzzy Decision Making in Modeling and Control.* World Scientific Publishing.

Charre-Ibarra, S., Velez-Díaz, D., Gudiño-Lau, J., Alcalá-Rodríguez, J., & Fuentes-Penna, A. (2016). *XIKUA*. Obtenido de https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n8/multimedia/a2/a2_4.jpg

Collado Oporto, C. (2016). *Acciones de Control.* Diapositivas de Power Point

Córdova, O. (2011). Métodos de Inferencia. Obtenido de https://prezi.com/5jju05ocq5xr/metodos-de-inferencia/

Cortes, F. R. (2011). Control de Robots Manipuladores. En *Robótica Control de Robots Manipuladores.* (pág. 17). Alfaomega.

Craig, J. (2006). Cinemática de manipuladores. En *Robótica* (pág. 63 - 66). Pearson Educación.

Cumpa, J. (2014). Lógica Difusa. Obtenido de https://es.slideshare.net/alexanderjamescumpamalca/lgica-difusa-38879988

d'Alembert, J. l. (2018). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Jean_le_Rond_d%27Alembert

Delta3D. (2017). *Delta3D*. Obtenido de https://delta3d.com.ar/guia-de-impresion-3d-filamento-2017-abs-vs-vs-pla-muchos-materiales/

DYOR: Do Your Own Robot. (2016). Obtenido de http://dyor.roboticafacil.es/criterios-de-diseno-y-fabricacion-mediante-impresion-3d/

FESTO. (2018). *FESTO.* Obtenido de https://www.festo.com/pe/es/npc/npc-config-ui/configimage/ESG_RUND_200.png

Gómez, A. A., López, A. L., & Patiño, C. R. (2007). *Scielo.* Obtenido de Control Difuso de una Plataforma Móvil Para el Seguimiento de Trayectorias: https://www.researchgate.net/publication/26544173_Control_difuso_de_una_plataforma_movil_para_el_seguimiento_de_trayectorias?_sg=Dnv2PgpGFoOCXnSCI4pJFX_OTR42CwZn_XWgzC0HQwyXUMn2UD76SrkQlYb1XQJtw0ftrDvnADGPZsmnWekGgenC8ocjkSEDsA

González, A. G. (2016). *Panamahitek.* Obtenido de http://panamahitek.com/wp-content/uploads/2016/11/servomotor.jpg

González, V. (2002). *Introducción Robótica.* Obtenido de http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/intro.htm

Guzmán, D., & Castaño, V. (2006). *La Lógica Difusa en Ingeniería.* Obtenido de https://revistas.ucr.ac.cr/index.php/cienciaytecnologia/article/download/2640/2591

HD1080ide. (2013). *YouTube.* Obtenido de Highlights der Hannover Messe : https://www.youtube.com/watch?v=epoqTYvAkH0

Hispaviacion. (2018). *Hispaviacion.* Obtenido de http://www.hispaviacion.es/wp-content/uploads/2018/01/gta1.jpg

Icaro. (2017). Coordenadas Homogéneas. Obtenido de http://icaro.eii.us.es/descargas/tema_4_parte_2.pdf

Inukai, Y. (2011). *Sistemas Tridimensionales*. Obtenido de http://sistematridimensional6a.blogspot.com/2011/02/sistemas-tridimensionales_28.html

ISC, I. M. (2011). *Matematicas Discretas*. Obtenido de Proposiciones compuestas (Disyunción, Conjunción, Negación, Condicional, Bicondicional) : https://matedisunidad3.wordpress.com/category/3-1-2-proposiciones-compuestas-disyuncion-conjuncion-negacion-condicional-bicondicional/

Juárez Pérez, P. (2006). *Implementación de un Control Supervisorio Difuso para el Control de Posición de un Robot Manipulador Delta.* Instituto Tecnológico y de Estudios Superiores de Monterrey .

Lelong. (2010). Obtenido de https://www.lelong.com.my/genius-metal-strike-3d-usb-joystick-pc-pclink-181655688-2017-08-Sale-P.htm

Leyva, H. R. (2012). Modelado Cinemática de Robots. Obtenido de http://www.utm.mx/~hugo/robot/Robot2.pdf

Liu, X.-J., & Wang, J. (2003). Analysis of a novel cylindrical 3-DoF parallel robot. *Semantic Scholar*. Obtenido de Analysis of a novel cylindrical 3-DoF parallel robot: https://www.semanticscholar.org/paper/Analysis-of-a-novel-cylindrical-3-DoF-parallel-Wang-Liu/ef89204911a30566327acd40bc103d5ca34b4bc5

Martín, S., Rodríguez, M., Bayarri, S., Redorta, P., Escovar, R., Fernández, E., & Mavrich, V. (2007). *Historia de la Robótica.* Obtenido de http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0210-48062007000200001

Mecánica, T. (Julio de 2017). *Sitenordeste*. Obtenido de http://www.sitenordeste.com/mecanica/maquinas_mecanismos.htm

Mecatronica.net. (2017). Herramientas Matemáticas Localización Espacial. Obtenido de http://fisica-mecatronica.net/robotica1/2a-Herramientas-localizacion-espacial.pdf

Morcillo, C. G. (2011). Lógica Difusa.

Naylamp Mechatronics . (2017). Obtenido de https://naylampmechatronics.com/

Nbio. (2017). *Dinámica de Robots*. Obtenido de http://nbio.umh.es/files/2012/04/practica3.pdf

Olsson, A. (2009). *Modeling and control of a Delta-3 robot.* Masther Thesis, Marktheidenfeld.

Peñaranda, S., Prada, L., Calle, I., & Fernandez, F. (2015). Robots Industriales. Obtenido de https://vdocuments.mx/historia-de-la-robotica-industrial.html

ido de http://www.pishrobot.com/en/products/dynamixel.htm

PLAREMESA. (2018). Obtenido de http://www.plaremesa.net/fibra-vidrio-precio-material-unico-ha-perdurado/

Ponce Cruz, P. (2010). Inteligencia artificial con aplicaciones a la ingeniería. Alfaomega Grupo Editor, S.A.

Raspberry . (2018). Obtenido de https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Robot Control by Fuzzy Logic. (2008). En M. I. Viorel Stoian, *Frontiers in Robotics, Automation and Control* (págs. 111-114). In-Tech.

Robotis. (2010). *ROBOTIS*. Obtenido de http://support.robotis.com/en/product/controller/opencm9.04.htm

Ruiz, J. (2010). *Robotica II Mecatrónica*. Obtenido de http://mecatroavila.blogspot.com/2010/08/caracteristicas-de-los-tipos-de.html

Sánchez, L. A., & Saavedra, M. S. (2005). *Matemáticas y Robótica*. Obtenido de https://imarrero.webs.ull.es/sctm05/modulo2lp/1/msigut.pdf

SEAS, B. (2018). *Blog SEAS*. Obtenido de https://www.seas.es/blog/diseno_mecanico/mecatronica-una-de-las-especializades-con-mas-salidas-laborales/

Suarez, D. (2016). *Ningenia*. Obtenido de http://www.ningenia.com/2016/03/17/profibus/

Tibaduiza, D., Amaya, I., Rodríguez, S., Mejia, N., & Flórez, M. (2011). Implementación de un Control Fuzzy para el Control Cinemático Directo en un Robot Manipulador. *Scielo*. Obtenido de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-33052011000300002

Tindie. (2018). Obtenido de https://www.tindie.com/products/CadSoft/eagle-pcb-software-license-and-raspberry-pi-2-bundle/

Vilchis-González, A., Ávila-Vilchis, J., Estrada-Flores, R., Martínez-Méndez, R., Portillo-Rodríguez, O., & Romero-Huertas, M. (2014). Robots modulares para cirugía mínimamente invasiva. *SCIELO*. Obtenido de http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322014000100008

Wikipedia. (2018). *MATLAB*. Obtenido de https://es.wikipedia.org/wiki/MATLAB

Wikipedia. (2018). *Trilateración*. Obtenido de https://es.wikipedia.org/wiki/Trilateraci%C3%B3n

Zacobria. (2018). *Zacobria*. Obtenido de http://www.zacobria.com/universal-robots-zacobria-3-dimension-robot-control-by-joystick-game-controller.html

Zahradník, R. (2017). *Radekzahradnik*. Obtenido de http://radekzahradnik.cz/wp-content/uploads/2016/09/visual_studio_purple-930x462.png

Zhang, D. (2010). *Tools Parallel Robotic Machine.* Springer.

ANEXOS

ANEXO 01

CÓDIGO EN MATLAB DE LA SIMULACIÓN DE LA CINEMÁTICA INVERSA DEL ROBOT PARALELO DELTA

SIMULACIÓN - CINEMÁTICA INVERSA

```
function varargout = PruebaN1(varargin)
% PRUEBAN1 MATLAB code for PruebaN1.fig
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PruebaN1_OpeningFcn, ...
                   'gui_OutputFcn',  @PruebaN1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PruebaN1 is made visible.
function PruebaN1_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = PruebaN1_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
VPX = get(handles.slider1,'Value');
set(handles.VPX,'String',VPX);
Px = get(handles.slider1,'Value');
Py = get(handles.slider2,'Value');
Pz = get(handles.slider3,'Value');
R=10;
r=5;
L1=25;
L2=40;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;

    axes(handles.axes1); set(handles.axes1,'Visible','on'); title('Gráfico
1');
    Si = (1/L1)*(-Px^2-Py^2-Pz^2+L2^2-L1^2-R1^2);
    %Angulo theta1
    Q1 = 2*Px*cosd(phi1)+2*Py*sind(phi1);
```

```
    D1= 4*Pz^2+4*R1^2-Si^2+Q1^2*(1-(R1^2/L1^2))+Q1*(((-2*R1*Si)/(L1))-
4*R1);
    E1= -2*R1-Q1*((R1/L1)-1)-Si;

    %Angulo theta2
    Q2 = 2*Px*cosd(phi2)+2*Py*sind(phi2);
    D2 = 4*Pz^2+4*R1^2-Si^2+Q2^2*(1-(R1^2/L1^2))+Q2*(((-2*R1*Si)/(L1))-
4*R1);
    E2 = -2*R1-Q2*((R1/L1)-1)-Si;

    %Angulo theta3
    Q3 = 2*Px*cosd(phi3)+2*Py*sind(phi3);
    D3 = 4*Pz^2+4*R1^2-Si^2+Q3^2*(1-(R1^2/L1^2))+Q3*(((-2*R1*Si)/(L1))-
4*R1);
    E3 = -2*R1-Q3*((R1/L1)-1)-Si;

    % RESULTADOS
    th1 = -2*atand(((-2*Pz)+sqrt(D1))/E1);
    th2 = -2*atand(((-2*Pz)+sqrt(D2))/E2);
    th3 = -2*atand(((-2*Pz)+sqrt(D3))/E3);

    the1 = th1;
    set(handles.the1,'String',the1);
    the2 = th2;
    set(handles.the2,'String',the2);
    the3 = th3;
    set(handles.the3,'String',the3);



    %Plot Base Fija
    plot3(Px,Py,-Pz,'red-','Linewidth',2,'Marker','*');hold on;
    plot3([ R*cosd(phi1) R*cosd(phi2)], [R*sind(phi1) R*sind(phi2)],[0
0],'black-','Linewidth',2,'Marker','o');hold on;
    grid on
    plot3([ R*cosd(phi2) R*cosd(phi3)], [R*sind(phi2) R*sind(phi3)],[0
0],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi1)], [R*sind(phi3) R*sind(phi1)],[0
0],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 1
    plot3([ R*cosd(phi1) R*cosd(phi1)+L1*cosd(phi1)*cosd(th1)], [
R*sind(phi1) R*sind(phi1)+L1*sind(phi1)*cosd(th1)],[0 -
L1*sind(th1)],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi2) R*cosd(phi2)+L1*cosd(phi2)*cosd(th2)], [
R*sind(phi2) R*sind(phi2)+L1*sind(phi2)*cosd(th2)],[0 -L1*sind(th2)],'blue-
','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi3)+L1*cosd(phi3)*cosd(th3)], [
R*sind(phi3) R*sind(phi3)+L1*sind(phi3)*cosd(th3)],[0 -
L1*sind(th3)],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Base Fija
    plot3([Px+r*cosd(phi1) Px+r*cosd(phi2)], [Py+r*sind(phi1)
Py+r*sind(phi2)],[-Pz -Pz],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi2) Px+r*cosd(phi3)], [Py+r*sind(phi2)
Py+r*sind(phi3)],[-Pz -Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi3) Px+r*cosd(phi1)], [Py+r*sind(phi3)
Py+r*sind(phi1)],[-Pz -Pz],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 2
    plot3([R*cosd(phi1)+L1*cosd(phi1)*cosd(th1) Px+r*cosd(phi1)],
[R*sind(phi1)+L1*sind(phi1)*cosd(th1) Py+r*sind(phi1)],[-L1*sind(th1) -
Pz],'black-','Linewidth',2,'Marker','o');hold on;
```

```
    plot3([R*cosd(phi2)+L1*cosd(phi2)*cosd(th2) Px+r*cosd(phi2)],
[R*sind(phi2)+L1*sind(phi2)*cosd(th2) Py+r*sind(phi2)],[-L1*sind(th2) -
Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([R*cosd(phi3)+L1*cosd(phi3)*cosd(th3) Px+r*cosd(phi3)],
[R*sind(phi3)+L1*sind(phi3)*cosd(th3) Py+r*sind(phi3)],[-L1*sind(th3) -
Pz],'green-','Linewidth',2,'Marker','o');hold off;

    axis([-60 60 -60 60 -80 35]);

% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
VPY = get(handles.slider2,'Value');
set(handles.VPY,'String',VPY);
Px = get(handles.slider1,'Value');
Py = get(handles.slider2,'Value');
Pz = get(handles.slider3,'Value');
R=10;
r=5;
L1=25;
L2=40;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;

    axes(handles.axes1); set(handles.axes1,'Visible','on'); title('Gráfico
1');
    Si = (1/L1)*(-Px^2-Py^2-Pz^2+L2^2-L1^2-R1^2);
    %Angulo theta1
    Q1 = 2*Px*cosd(phi1)+2*Py*sind(phi1);
    D1= 4*Pz^2+4*R1^2-Si^2+Q1^2*(1-(R1^2/L1^2))+Q1*(((-2*R1*Si)/(L1))-
4*R1);
    E1= -2*R1-Q1*((R1/L1)-1)-Si;

    %Angulo theta2
    Q2 = 2*Px*cosd(phi2)+2*Py*sind(phi2);
    D2 = 4*Pz^2+4*R1^2-Si^2+Q2^2*(1-(R1^2/L1^2))+Q2*(((-2*R1*Si)/(L1))-
4*R1);
```

```
    E2 = -2*R1-Q2*((R1/L1)-1)-Si;

    %Angulo theta3
    Q3 = 2*Px*cosd(phi3)+2*Py*sind(phi3);
    D3 = 4*Pz^2+4*R1^2-Si^2+Q3^2*(1-(R1^2/L1^2))+Q3*(((-2*R1*Si)/(L1))-
4*R1);
    E3 = -2*R1-Q3*((R1/L1)-1)-Si;

    % RESULTADOS
    th1 = -2*atand(((-2*Pz)+sqrt(D1))/E1);
    th2 = -2*atand(((-2*Pz)+sqrt(D2))/E2);
    th3 = -2*atand(((-2*Pz)+sqrt(D3))/E3);

    the1 = th1;
    set(handles.the1,'String',the1);
    the2 = th2;
    set(handles.the2,'String',the2);
    the3 = th3;
    set(handles.the3,'String',the3);

    %Plot Base Fija
    plot3(Px,Py,-Pz,'red-','Linewidth',2,'Marker','*');hold on;
    plot3([ R*cosd(phi1) R*cosd(phi2)], [R*sind(phi1) R*sind(phi2)],[0
0],'black-','Linewidth',2,'Marker','o');hold on;
    grid on
    plot3([ R*cosd(phi2) R*cosd(phi3)], [R*sind(phi2) R*sind(phi3)],[0
0],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi1)], [R*sind(phi3) R*sind(phi1)],[0
0],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 1
    plot3([ R*cosd(phi1) R*cosd(phi1)+L1*cosd(phi1)*cosd(th1)], [
R*sind(phi1) R*sind(phi1)+L1*sind(phi1)*cosd(th1)],[0 -
L1*sind(th1)],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi2) R*cosd(phi2)+L1*cosd(phi2)*cosd(th2)], [
R*sind(phi2) R*sind(phi2)+L1*sind(phi2)*cosd(th2)],[0 -L1*sind(th2)],'blue-
','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi3)+L1*cosd(phi3)*cosd(th3)], [
R*sind(phi3) R*sind(phi3)+L1*sind(phi3)*cosd(th3)],[0 -
L1*sind(th3)],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Base Fija
    plot3([Px+r*cosd(phi1) Px+r*cosd(phi2)], [Py+r*sind(phi1)
Py+r*sind(phi2)],[-Pz -Pz],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi2) Px+r*cosd(phi3)], [Py+r*sind(phi2)
Py+r*sind(phi3)],[-Pz -Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi3) Px+r*cosd(phi1)], [Py+r*sind(phi3)
Py+r*sind(phi1)],[-Pz -Pz],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 2
    plot3([R*cosd(phi1)+L1*cosd(phi1)*cosd(th1) Px+r*cosd(phi1)],
[R*sind(phi1)+L1*sind(phi1)*cosd(th1) Py+r*sind(phi1)],[-L1*sind(th1) -
Pz],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([R*cosd(phi2)+L1*cosd(phi2)*cosd(th2) Px+r*cosd(phi2)],
[R*sind(phi2)+L1*sind(phi2)*cosd(th2) Py+r*sind(phi2)],[-L1*sind(th2) -
Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([R*cosd(phi3)+L1*cosd(phi3)*cosd(th3) Px+r*cosd(phi3)],
[R*sind(phi3)+L1*sind(phi3)*cosd(th3) Py+r*sind(phi3)],[-L1*sind(th3) -
Pz],'green-','Linewidth',2,'Marker','o');hold off;
    axis([-60 60 -60 60 -80 35]);
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
slider
  % --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
VPZ = get(handles.slider3,'Value');
set(handles.VPZ,'String',VPZ);
Px = get(handles.slider1,'Value');
Py = get(handles.slider2,'Value');
Pz = get(handles.slider3,'Value');
R=10;
r=5;
L1=25;
L2=40;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;

    axes(handles.axes1); set(handles.axes1,'Visible','on'); title('Gráfico
1');
    Si = (1/L1)*(-Px^2-Py^2-Pz^2+L2^2-L1^2-R1^2);
    %Angulo theta1
    Q1 = 2*Px*cosd(phi1)+2*Py*sind(phi1);
    D1= 4*Pz^2+4*R1^2-Si^2+Q1^2*(1-(R1^2/L1^2))+Q1*(((-2*R1*Si)/(L1))-
4*R1);
    E1= -2*R1-Q1*((R1/L1)-1)-Si;

    %Angulo theta2
    Q2 = 2*Px*cosd(phi2)+2*Py*sind(phi2);
    D2 = 4*Pz^2+4*R1^2-Si^2+Q2^2*(1-(R1^2/L1^2))+Q2*(((-2*R1*Si)/(L1))-
4*R1);
    E2 = -2*R1-Q2*((R1/L1)-1)-Si;

    %Angulo theta3
    Q3 = 2*Px*cosd(phi3)+2*Py*sind(phi3);
    D3 = 4*Pz^2+4*R1^2-Si^2+Q3^2*(1-(R1^2/L1^2))+Q3*(((-2*R1*Si)/(L1))-
4*R1);
    E3 = -2*R1-Q3*((R1/L1)-1)-Si;
     % RESULTADOS
    th1 = -2*atand(((-2*Pz)+sqrt(D1))/E1);
    th2 = -2*atand(((-2*Pz)+sqrt(D2))/E2);
    th3 = -2*atand(((-2*Pz)+sqrt(D3))/E3);

    the1 = th1;
    set(handles.the1,'String',the1);
    the2 = th2;
```

```
    set(handles.the2,'String',the2);
    the3 = th3;
    set(handles.the3,'String',the3);

    %Plot Base Fija
    plot3(Px,Py,-Pz,'red-','Linewidth',2,'Marker','*');hold on;
    plot3([ R*cosd(phi1) R*cosd(phi2)], [R*sind(phi1) R*sind(phi2)],[0
0],'black-','Linewidth',2,'Marker','o');hold on;
    grid on
    plot3([ R*cosd(phi2) R*cosd(phi3)], [R*sind(phi2) R*sind(phi3)],[0
0],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi1)], [R*sind(phi3) R*sind(phi1)],[0
0],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 1
    plot3([ R*cosd(phi1) R*cosd(phi1)+L1*cosd(phi1)*cosd(th1)], [
R*sind(phi1) R*sind(phi1)+L1*sind(phi1)*cosd(th1)],[0 -
L1*sind(th1)],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi2) R*cosd(phi2)+L1*cosd(phi2)*cosd(th2)], [
R*sind(phi2) R*sind(phi2)+L1*sind(phi2)*cosd(th2)],[0 -L1*sind(th2)],'blue-
','Linewidth',2,'Marker','o');hold on;
    plot3([ R*cosd(phi3) R*cosd(phi3)+L1*cosd(phi3)*cosd(th3)], [
R*sind(phi3) R*sind(phi3)+L1*sind(phi3)*cosd(th3)],[0 -
L1*sind(th3)],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Base Fija
    plot3([Px+r*cosd(phi1) Px+r*cosd(phi2)], [Py+r*sind(phi1)
Py+r*sind(phi2)],[-Pz -Pz],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi2) Px+r*cosd(phi3)], [Py+r*sind(phi2)
Py+r*sind(phi3)],[-Pz -Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([Px+r*cosd(phi3) Px+r*cosd(phi1)], [Py+r*sind(phi3)
Py+r*sind(phi1)],[-Pz -Pz],'green-','Linewidth',2,'Marker','o');hold on;
    %Plot Eslabon 2
    plot3([R*cosd(phi1)+L1*cosd(phi1)*cosd(th1) Px+r*cosd(phi1)],
[R*sind(phi1)+L1*sind(phi1)*cosd(th1) Py+r*sind(phi1)],[-L1*sind(th1) -
Pz],'black-','Linewidth',2,'Marker','o');hold on;
    plot3([R*cosd(phi2)+L1*cosd(phi2)*cosd(th2) Px+r*cosd(phi2)],
[R*sind(phi2)+L1*sind(phi2)*cosd(th2) Py+r*sind(phi2)],[-L1*sind(th2) -
Pz],'blue-','Linewidth',2,'Marker','o');hold on;
    plot3([R*cosd(phi3)+L1*cosd(phi3)*cosd(th3) Px+r*cosd(phi3)],
[R*sind(phi3)+L1*sind(phi3)*cosd(th3) Py+r*sind(phi3)],[-L1*sind(th3) -
Pz],'green-','Linewidth',2,'Marker','o');hold off;

    axis([-60 60 -60 60 -80 35]);

% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

SIMULACIÓN - CINEMÁTICA DIRECTA

```
clear all
clc
global Bx1 Bx2 Bx3 By1 By2 By3 Bz1 Bz2 Bz3 R r L1 L2 phi1 phi2 phi3;

%Datos
R=10;
r=5;
phi1=0;
phi2=120;
phi3=240;
L1=25;
L2=40;
%Cinemática Directa

theta1=input('Ingrese theta1 = ')
theta2=input('Ingrese theta2 = ')
theta3=input('Ingrese theta3 = ')

Bx1=(R-r)*cosd(phi1)+L1*cosd(phi1)*cosd(-theta1);
By1=(R-r)*sind(phi1)+L1*sind(phi1)*cosd(-theta1);
Bz1=-L1*sind(-theta1);

Bx2=(R-r)*cosd(phi2)+L1*cosd(phi2)*cosd(-theta2);
By2=(R-r)*sind(phi2)+L1*sind(phi2)*cosd(-theta2);
Bz2=-L1*sind(-theta2);

Bx3=(R-r)*cosd(phi3)+L1*cosd(phi3)*cosd(-theta3);
By3=(R-r)*sind(phi3)+L1*sind(phi3)*cosd(-theta3);
Bz3=-L1*sind(-theta3);

 x0=[1000 1000 1000];
P=fsolve('myfun',x0)
```

Función "myfun"

```
function F=myfun(x);
global Bx1 Bx2 Bx3 By1 By2 By3 Bz1 Bz2 Bz3 L2;
F = [sqrt((x(1)-Bx1)^2+(x(2)-By1)^2+(x(3)-Bz1)^2)-L2;
     sqrt((x(1)-Bx2)^2+(x(2)-By2)^2+(x(3)-Bz2)^2)-L2;
     sqrt((x(1)-Bx3)^2+(x(2)-By3)^2+(x(3)-Bz3)^2)-L2];
end
```

SIMULACIÓN – ESPACIO DE TRABAJO

```
clear all
clc
close all
%Cinemática Inversa
syms X
syms Y
syms Z
R=10;
r=5;
L1=25;
L2=40;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;
Pz=0;
i=1;
m=4;
for Px =-60:m:60
   for Py=-60:m:60
        for Pz=0:4:70
            Si = (1/L1)*(-Px^2-Py^2-Pz^2+L2^2-L1^2-R1^2);
            %Angulo theta1
            Q1 = 2*Px*cosd(phi1)+2*Py*sind(phi1);
            D1= 4*Pz^2+4*R1^2-Si^2+Q1^2*(1-(R1^2/L1^2))+Q1*(((-
2*R1*Si)/(L1))-4*R1);

            %Angulo theta2
            Q2 = 2*Px*cosd(phi2)+2*Py*sind(phi2);
            D2 = 4*Pz^2+4*R1^2-Si^2+Q2^2*(1-(R1^2/L1^2))+Q2*(((-
2*R1*Si)/(L1))-4*R1);

            %Angulo theta3
            Q3 = 2*Px*cosd(phi3)+2*Py*sind(phi3);
            D3 = 4*Pz^2+4*R1^2-Si^2+Q3^2*(1-(R1^2/L1^2))+Q3*(((-
2*R1*Si)/(L1))-4*R1);

            %Plot Base Fija
               if D1>=0 && D2>=0 && D3>=0;
                    x(1,i)=Px;
                    y(1,i)=Py;
                    z(1,i)=-Pz-15;
                    i=i+1;

               end
        end
    end
end


tri = delaunay(x,y,z);
figure(1)
h = trisurf(tri, x, y, z,'FaceColor','interp','FaceLighting','phong');
axis vis3d
```

ANEXO 02

CÓDIGO EN MATLAB DE LA DEL DIMENSIONAMIENTO DE ACTUADORES

```
clear all
clc
close all
global Bx1 Bx2 Bx3 By1 By2 By3 Bz1 Bz2 Bz3 R r L1 L2 phi1 phi2 phi3;
R=0.1;
r=0.05;
L1=0.25;
L2=0.40;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;

%Variables Dinámicas
Kr=1/353.5;          % Parámetro de Reducción de Engranajes
mbm = 0.038;      % Masa Base Móvil en kg
mload = 0.300;    % Masa de la Carga
ml1 = 0.259;      % Masa del Brazo 1
ml2 = 0.060;      % Masa de las dos Piezas del Brazo 2
mrot = 0.039;     % Masa de la Rotula conectada al Brazo 2
re = 2/3;         % Relación de masa entre Eslabón N°1 y eslabón N°2
g = 9.81;         % Gravedad
mtotalbm = mbm + mload + 6*(1-re)*ml2+6*mrot;          % Masa total Base
Movil
rcm = L1*(((1/2)*ml1)+(mrot*4)+r*2*ml2)/(ml1+mrot+r*2*ml2);     %Centro de
Masa del Eslabón N°1
Jmotor = 0.0069;  % Momento de Inercia del Motor en Kg*cm^2 (Puede cambiar)

Ibc = L1^2*((ml1/3)+4*mrot+r*2*ml2);     % Suma de la Inercia creada por la
masa de L1 y su inercia del extremo
Ibi = Jmotor*Kr^2 + Ibc;
%% Inicio
T= input('ingrese el tiempo: ')
t=0:0.1:T;
q1= 15+(0.8*t.*sin(t/5));
%q2= cos(t/5);
q2= 15+(0.8*t.*sin(t/5));
q3= 15+(0.8*t.*sin(t/5));

dq1 = (4*sin(t/5))/5 + (4*t.*cos(t/5))/25;
dq2 = (4*sin(t/5))/5 + (4*t.*cos(t/5))/25;
dq3 = (4*sin(t/5))/5 + (4*t.*cos(t/5))/25;

ddq1= (8*cos(t/5))/25 - (4*t.*sin(t/5))/125;
ddq2= (8*cos(t/5))/25 - (4*t.*sin(t/5))/125;
ddq3= (8*cos(t/5))/25 - (4*t.*sin(t/5))/125;


tam1 = length(t);

figure(1)
     subplot(3,3,1); plot(t,q1,'r'); xlabel('Time (s)'); ylabel('Art-1
°');grid on;
     subplot(3,3,4); plot(t,dq1,'r'); xlabel('Time (s)'); ylabel('Velocidad
°/s');grid on;
     subplot(3,3,7); plot(t,ddq1,'r'); xlabel('Time (s)');
ylabel('Aceleración °/s^2');grid on;

     subplot(3,3,2); plot(t,q2,'b'); xlabel('Time (s)'); ylabel('Art-2
°');grid on;
```

```
     subplot(3,3,5); plot(t,dq2,'b'); xlabel('Time (s)'); ylabel('Velocidad
°/s');grid on;
     subplot(3,3,8); plot(t,ddq2,'b'); xlabel('Time (s)');
ylabel('Aceleración °/s^2');grid on;

     subplot(3,3,3); plot(t,q3,'g'); xlabel('Time (s)'); ylabel('Art-3
°');grid on;
     subplot(3,3,6); plot(t,dq3,'g'); xlabel('Time (s)'); ylabel('Velocidad
°/s');grid on;
     subplot(3,3,9); plot(t,ddq3,'g'); xlabel('Time (s)');
ylabel('Aceleración °/s^2');grid on;
%% Bucle de Cinemática directa
  for j=1:1:tam1;

theta1=-q1(1,j);
theta2=-q2(1,j);
theta3=-q3(1,j);

Bx1=(R-r)*cosd(phi1)+L1*cosd(phi1)*cosd(theta1);
By1=(R-r)*sind(phi1)+L1*sind(phi1)*cosd(theta1);
Bz1=-L1*sind(theta1);

Bx2=(R-r)*cosd(phi2)+L1*cosd(phi2)*cosd(theta2);
By2=(R-r)*sind(phi2)+L1*sind(phi2)*cosd(theta2);
Bz2=-L1*sind(theta2);

Bx3=(R-r)*cosd(phi3)+L1*cosd(phi3)*cosd(theta3);
By3=(R-r)*sind(phi3)+L1*sind(phi3)*cosd(theta3);
Bz3=-L1*sind(theta3);

x0=[1000 1000 1000];
P=fsolve('myfun',x0);
 if abs(P(1,1))<10^-3
     Pxx(1,j)=0;
 else
    Pxx(1,j) = P(1,1);
 end
 if abs(P(1,2))<10^-3
     Pyy(1,j)=0;
 else
    Pyy(1,j) = P(1,2);
 end
 if abs(P(1,3))<10^-3
     Pzz(1,j)=0;
 else
    Pzz(1,j) = P(1,3);
 end
  end
 figure(2)
plot3(Pxx,Pyy,-Pzz);xlabel('Px CD'); ylabel('Py CD '); zlabel('Pz CD');grid
on


%% Jacobiana
for j=1:1:tam1;

    Px = Pxx(1,j);
    Py = Pyy(1,j);
    Pz = Pzz(1,j);
    theta1=-q1(1,j);
```

```
    theta2=-q2(1,j);
    theta3=-q3(1,j);
    deq1 = dq1(1,j);
    deq2 = dq2(1,j);
    deq3 = dq3(1,j);
    ddeq1=ddq1(1,j);
    ddeq2=ddq2(1,j);
    ddeq3=ddq3(1,j);
    % Jacobiana
    s1T=[Px-cosd(phi1)*(R1 + L1*cosd(theta1)), Py - sind(phi1)*(R1 +
L1*cosd(theta1)), Pz - (L1*sind(theta1))];
    s2T=[Px-cosd(phi2)*(R1 + L1*cosd(theta2)), Py - sind(phi2)*(R1 +
L1*cosd(theta2)), Pz - (L1*sind(theta2))];
    s3T=[Px-cosd(phi3)*(R1 + L1*cosd(theta3)), Py - sind(phi3)*(R1 +
L1*cosd(theta3)), Pz - (L1*sind(theta3))];
    b1=[-L1*cosd(phi1)*sind(theta1);-L1*sind(phi1)*sind(theta1);
L1*cosd(theta1)];
    b2=[-L1*cosd(phi2)*sind(theta2);-L1*sind(phi2)*sind(theta2);
L1*cosd(theta2)];
    b3=[-L1*cosd(phi3)*sind(theta3);-L1*sind(phi3)*sind(theta3);
L1*cosd(theta3)];
    siTT=[s1T;s2T;s3T];
    u=j;
    K=[s1T*b1 0 0;0 s2T*b2 0;0 0 s3T*b3];

    Jacob = (siTT^-1)*K;
    Velc(:,:,j) = ((Jacob)*[deq1;deq2;deq3]);
    Vx(1,j) = Velc(1,:,j);
    Vy(1,j) = Velc(2,:,j);
    Vz(1,j) = Velc(3,:,j);

    %Aceleración
    s1dT=[Vx(1,j)+L1*cosd(phi1)*sind(theta1)*deq1 ,
Vy(1,j)+L1*sind(phi1)*sind(theta1)*deq1 , Vz(1,j)-L1*cosd(theta1)*deq1 ];
%Derivada de S1T
    s2dT=[Vx(1,j)+L1*cosd(phi2)*sind(theta2)*deq2 ,
Vy(1,j)+L1*sind(phi2)*sind(theta2)*deq2 , Vz(1,j)-L1*cosd(theta2)*deq2 ];
%Derivada de S2T
    s3dT=[Vx(1,j)+L1*cosd(phi3)*sind(theta3)*deq3 ,
Vy(1,j)+L1*sind(phi3)*sind(theta3)*deq3 , Vz(1,j)-L1*cosd(theta3)*deq3 ];
%Derivada de S3T
    b1d=[-L1*cosd(phi1)*cosd(theta1)*deq1 ;-L1*cosd(theta1)*sind(phi1)*deq1
;-L1*sind(theta1)*deq1 ];            %Derivada de b1
    b2d=[-L1*cosd(phi2)*cosd(theta2)*deq2 ;-L1*cosd(theta2)*sind(phi2)*deq2
;-L1*sind(theta2)*deq2 ];            %Derivada de b2
    b3d=[-L1*cosd(phi3)*cosd(theta3)*deq3 ;-L1*cosd(theta3)*sind(phi3)*deq3
;-L1*sind(theta3)*deq3 ];            %Derivada de b3
    sidT=[s1dT;s2dT;s3dT];
    OK = [s1dT*b1+s1T*b1d 0 0;0 s2dT*b2+s2T*b2d 0;0 0 s3dT*b3+s3T*b3d];
    Mn = (-sidT*Jacob+OK);
    Aart = [ddeq1;ddeq2;ddeq3];
    Acar(:,:,j)  = (siTT^-
1)*Mn*[deq1;deq2;deq3]+(Jacob*[ddeq1;ddeq2;ddeq3]);
    Acelcar=(siTT^-1)*Mn*[deq1;deq2;deq3]+(Jacob*[ddeq1;ddeq2;ddeq3]);

    Ax(1,j) = Acar(1,:,j);
    Ay(1,j) = Acar(2,:,j);
    Az(1,j) = Acar(3,:,j);

    %% Modelo Dinámico en Base al Trabajo Virtual
```

```
 % Para la base movil
 Gn = mtotalbm*[0 0 -g]';    % Fuerza Gravitacional
 Fn = mtotalbm*Acelcar;        % Vector de Fuerza Axial
 Tnn = Jacob'*Fn;            % Vector de Fuerza de Axial de la base móvil
 Tgn = Jacob'*Gn;            % Vector de Torques Gravitatorios

 % Para el Brazo 1
 Gb = -ml1*g;                  % Peso del Brazo 1
 Tgb = rcm*Gb*[cosd(theta1) cosd(theta2) cosd(theta3)]'; % Vector de
Torques Gravitacionales
 Ib = [Ibi 0 0;0 Ibi 0;0 0 Ibi];        %Matriz de Inercias de cada Brazo
 Tb = Ib*Aart;              % Vector de Torques de cada brazo

 %Dinámica del robot
  Tmotor(:,:,j) = Tb + Tnn - (Tgn + Tgb);
  Tmotor1(1,j)=Tmotor(1,:,j);
  Tmotor2(1,j)=Tmotor(2,:,j);
  Tmotor3(1,j)=Tmotor(3,:,j);

end

figure(3)
subplot(3,3,1); plot(t,Pxx,'r'); xlabel('Time (s)'); ylabel('Posición X
(m)');grid on;
subplot(3,3,4); plot(t,Vx,'r'); xlabel('Time (s)'); ylabel('Vel X
(m/s)');grid on;
subplot(3,3,7); plot(t,Ax,'r'); xlabel('Time (s)'); ylabel('Acel
X(m/s^2)');grid on;

subplot(3,3,2); plot(t,Pyy,'g'); xlabel('Time (s)'); ylabel('Posición Y
(m)');grid on;
subplot(3,3,5); plot(t,Vy,'g'); xlabel('Time (s)'); ylabel('Vel
Y(m/s)');grid on;
subplot(3,3,8); plot(t,Ay,'g'); xlabel('Time (s)'); ylabel('Acel
Y(m/s^2)');grid on;

subplot(3,3,3); plot(t,-Pzz,'b'); xlabel('Time (s)'); ylabel('Posición Z
(m)');grid on;
subplot(3,3,6); plot(t,Vz,'b'); xlabel('Time (s)'); ylabel('Vel
Z(m/s)');grid on;
subplot(3,3,9); plot(t,Az,'b'); xlabel('Time (s)'); ylabel('Acel
Z(m/s^2)');grid on;

figure(4)
subplot(3,1,1); plot(t,Tmotor1,'r'); xlabel('Time (s)'); ylabel('Torque
motor 1 N.m');grid on;
subplot(3,1,2); plot(t,Tmotor2,'g'); xlabel('Time (s)'); ylabel('Torque
motor 2 N.m');grid on;
subplot(3,1,3); plot(t,Tmotor3,'b'); xlabel('Time (s)'); ylabel('Torque
motor 3 N.m');grid on;
```

ANEXO 03

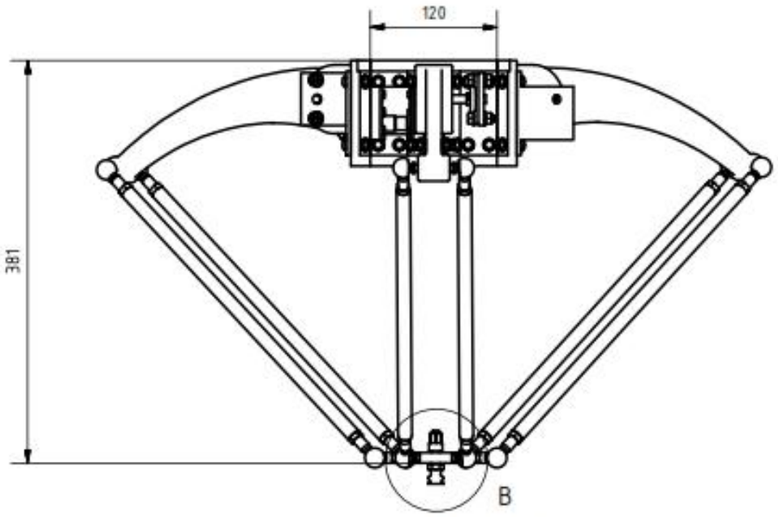
PLANOS DE DISEÑO Y ENSAMBLE DEL ROBOT PARALELO DELTA

ROBOT PARALELO DELTA DE 3 GDL

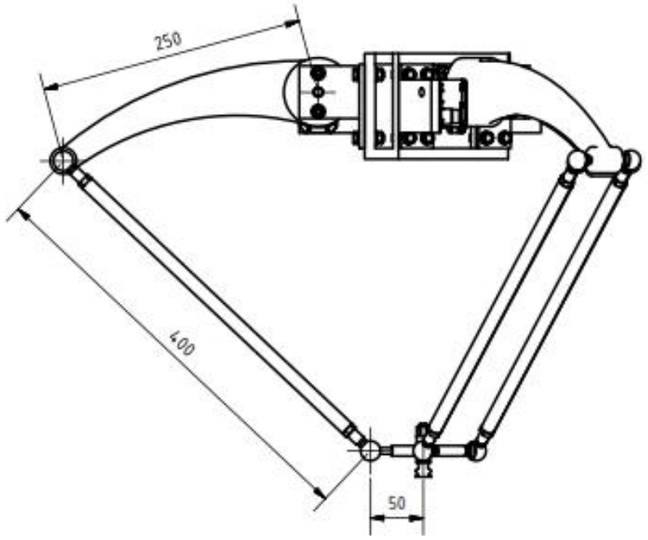
Vista Isométrica

Escala: 1 : 4

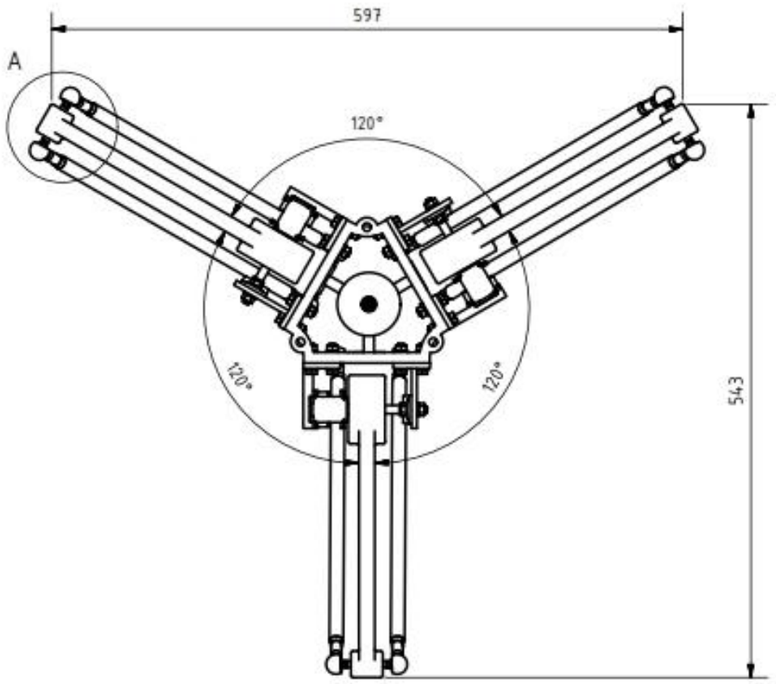
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	ROBOT PARALELO DELTA DE 3GDL			Nº PLANO: 01 - 016	REV: C



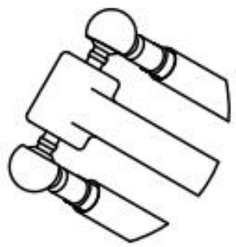
Vista Frontal
Escala: 1 : 5



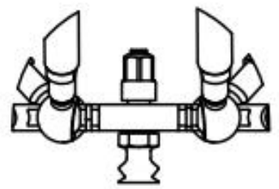
Vista Lateral Izquierda
Escala: 1 : 5



Vista Superior
Escala: 1 : 5



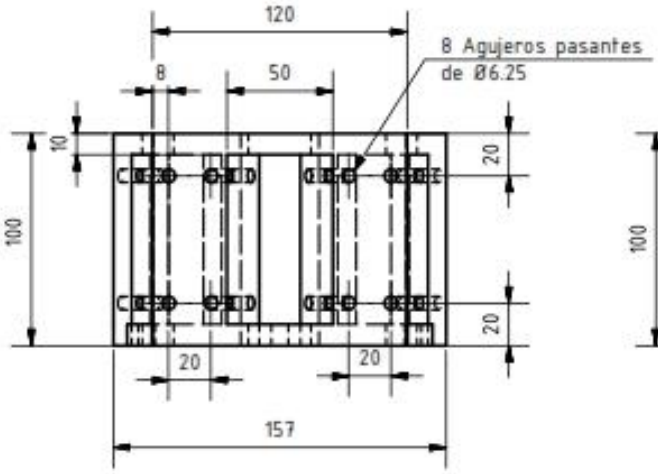
Detalle A
Escala: 1 : 2



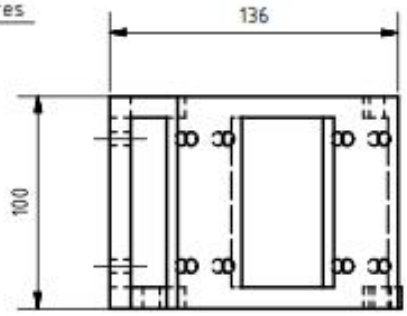
Detalle B
Escala: 1 : 2

NOTA:
- Las medidas de altura, ancho y alto mostrada son variables.
- El modelo presentendo es referencial.
- EL robot comparte una simetria de 120°.

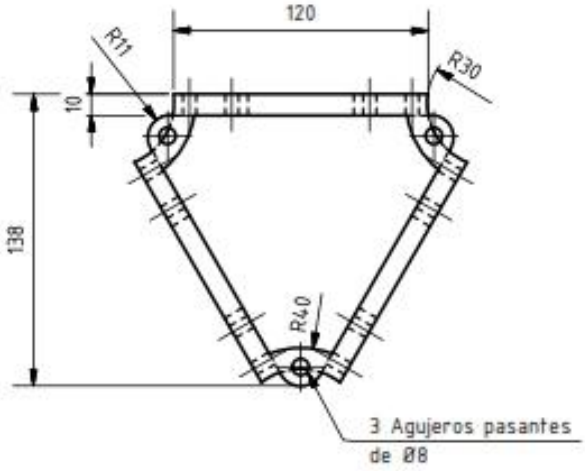
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A3
ESCALA: INDICADAS	ROBOT PARALELO DELTA DE 3GDL			Nº PLANO: 02 - 016	REV: C



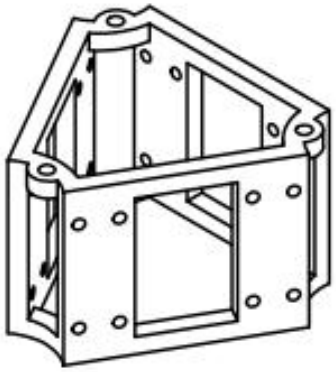
Vista Frontal
Escala: 1 : 3



Vista Lateral Izquierda
Escala: 1 : 3



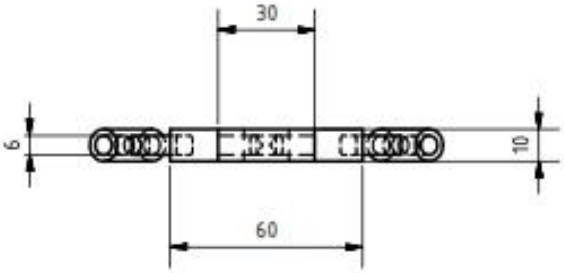
Vista Superior
Escala: 1 : 3



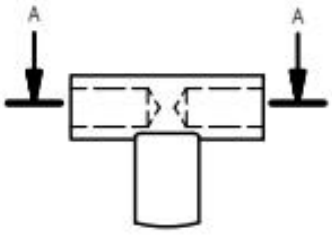
BASE FIJA
Isométrico
Escala: 1 : 3

NOTA: Todas las caras de la base fija son simetricas.

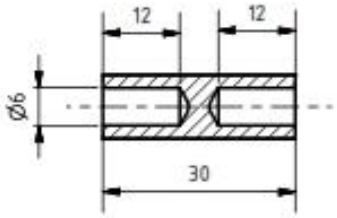
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	Base Fija			Nº PLANO: 03 - 016	REV: C



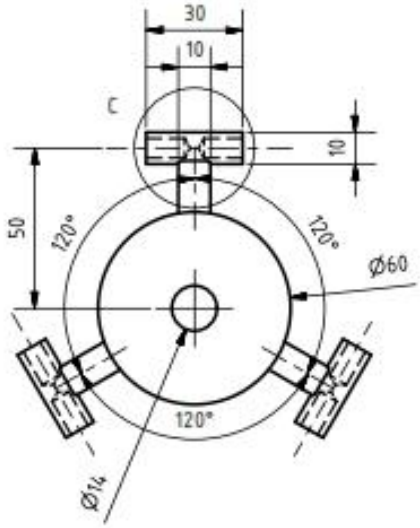
Vista Frontal
Escala: 1 : 2



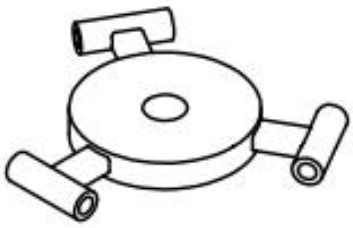
Detalle C
Escala: 1 : 1



Sección A-A
Escala: 1 : 1

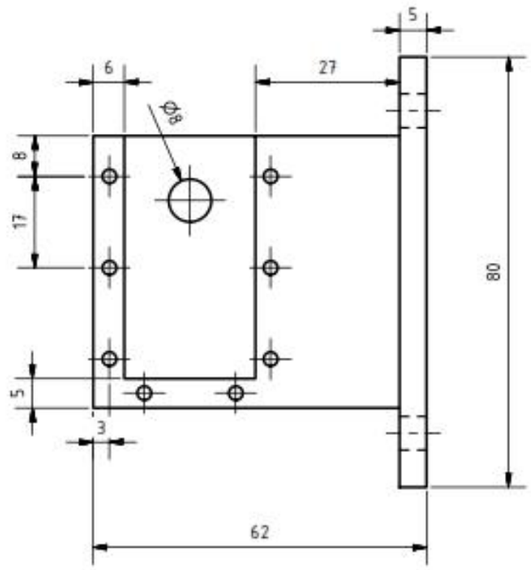


Vista Superior
Escala: 1 : 2

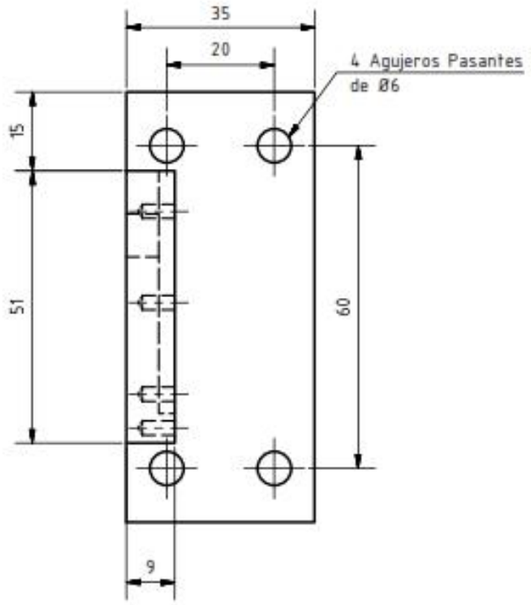


BASE MÓVIL
Vista Isométrica
Escala:1 : 2

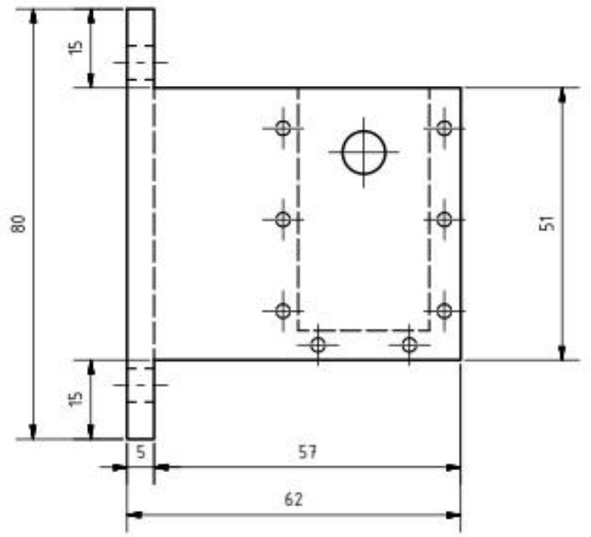
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	Base Móvil			Nº PLANO: 04 - 016	REV: C



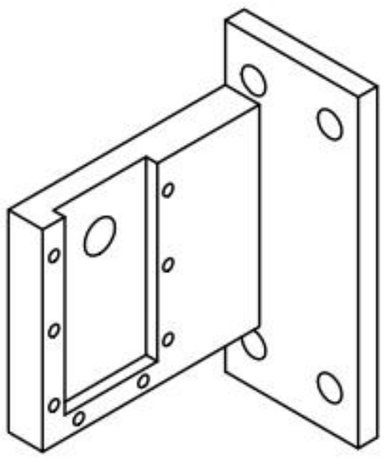
Vista Lateral Derecha
Escala: 1 : 1



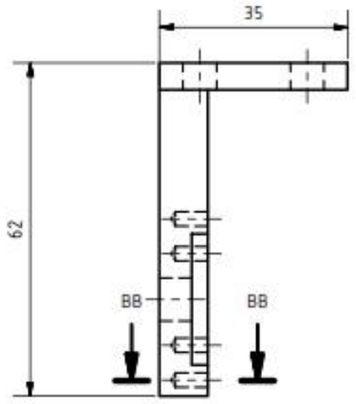
Vista Frontal
Escala: 1 : 1



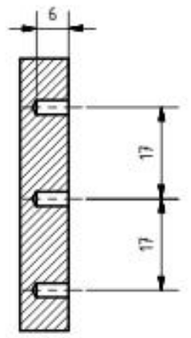
Vista Lateral Izquierda
Escala: 1 : 1



BASE DEL SERVOMOTOR
Vista Isométrica
Escala: 1 : 1
Cant. 03 Unidades

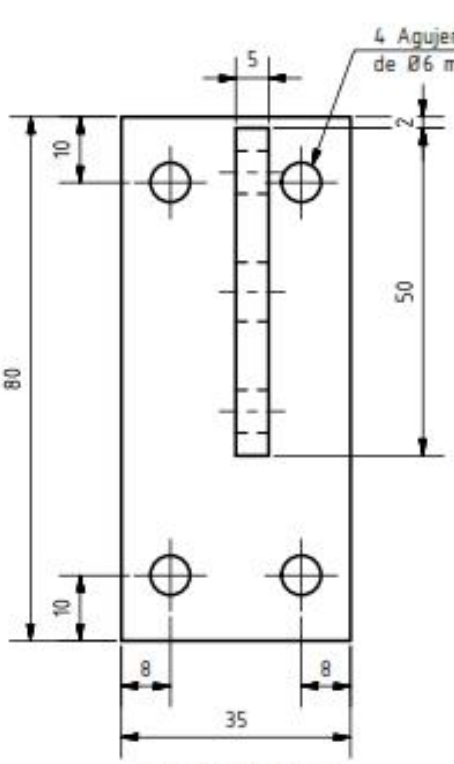


Vista Superior
Escala: 1 : 1

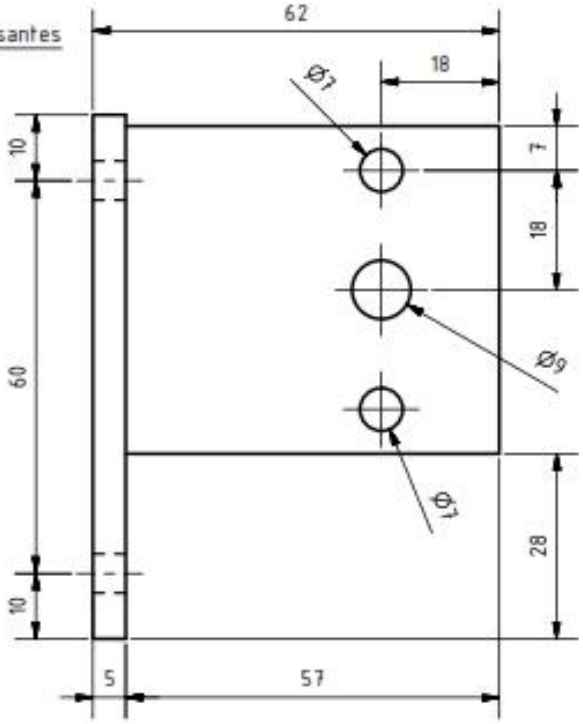


Sección BB - BB
Escala: 1 : 1

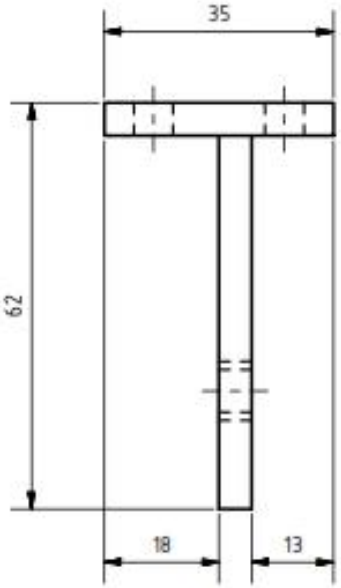
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018	
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA — HOJA: A3
ESCALA: INDICADAS	Base Servomotor			Nº PLANO: 05 - 016 — REV: C



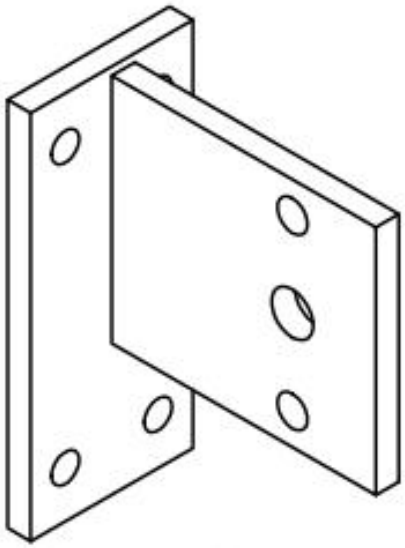
Vista Frontal
Escala: 1 : 1



Vista Lateral
Escala: 1 : 1

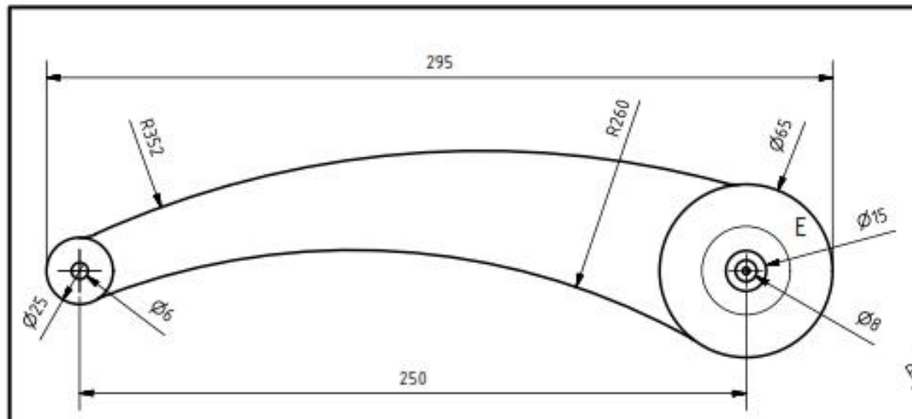


Vista Superior
Escala: 1 : 1

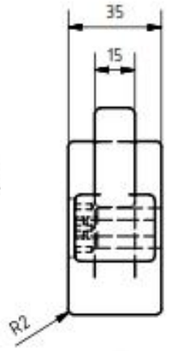


Base de la Chumacera
Vista Isométrica
Escala: 1 : 1
Cant. 03 Unidades

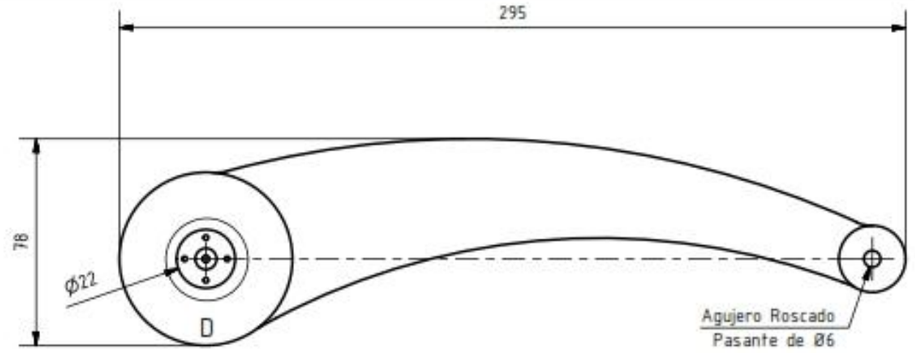
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	Base_Chumacera			Nº PLANO: 06 - 016	REV: C



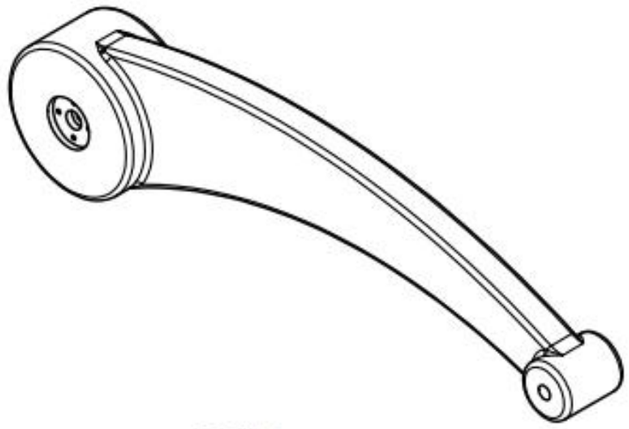
Vista Lateral Derecha
Escala: 1 : 2



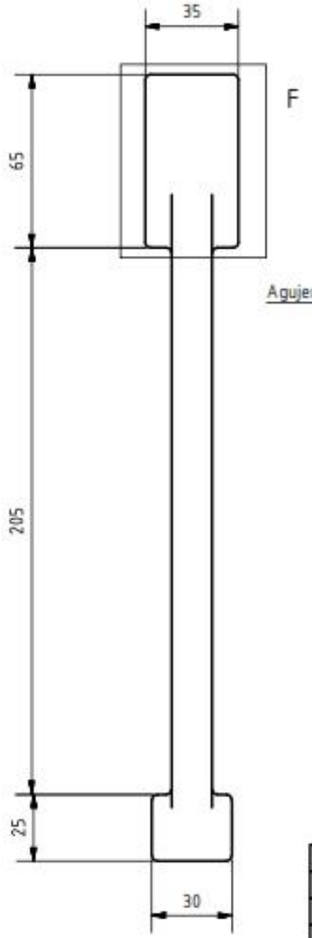
Vista Frontal
Escala: 1 : 2



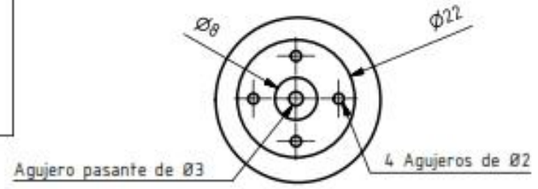
Vista Lateral Izquierda
Escala: 1 : 2



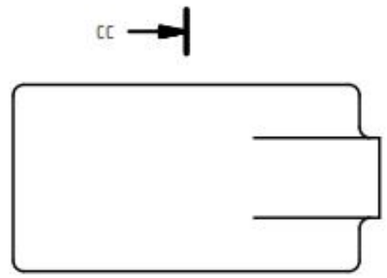
BRAZO
Vista Isométrica
Escala: 1 : 2
Cant. 03 Unidades



FG (1 : 2)

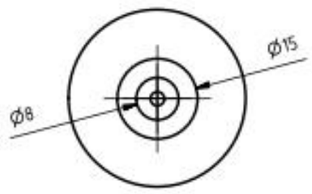


Detalle D
Escala: 1 : 1

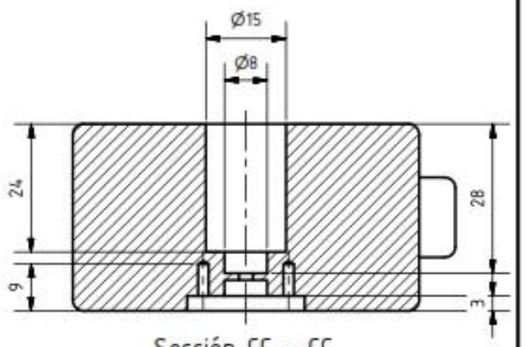


CC

Detalle F
Escala: 1 : 1

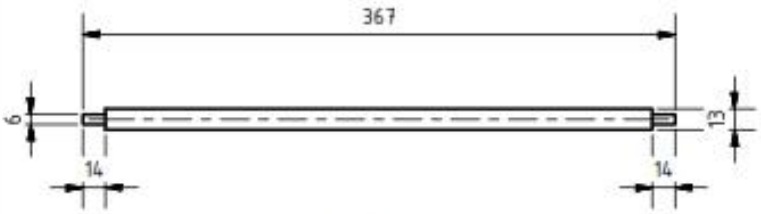


Detalle E
Escala: 1 : 1

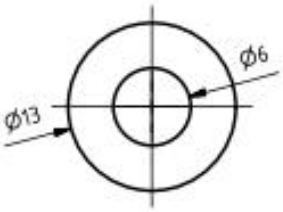


Sección CC – CC
Escala: 1 : 1

	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A3
ESCALA: INDICADAS	Brazo			Nº PLANO: 07 - 016	REV: C



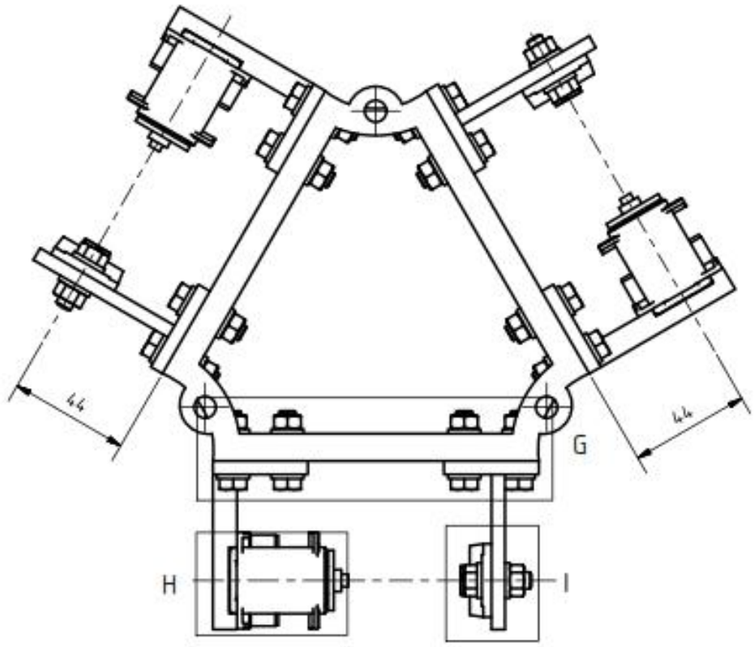
Vista Frontal
Escala: 1 : 4

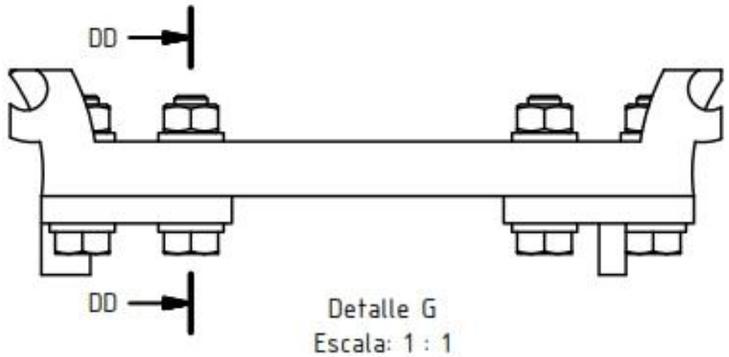


Vista Superior
Escala: 2 : 1

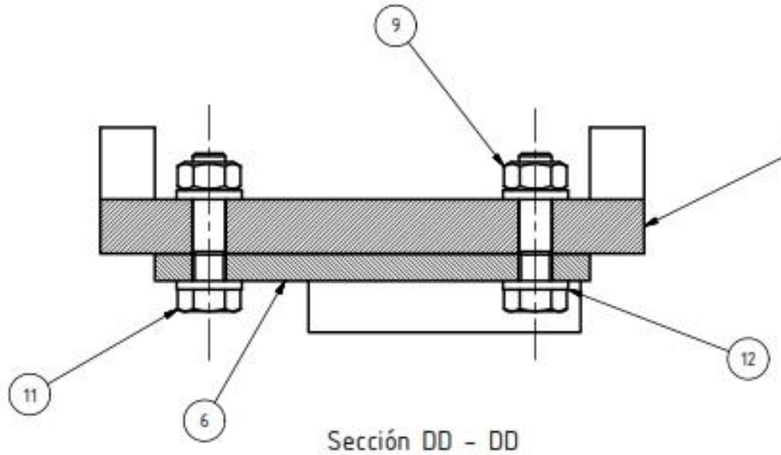
ANTEBRAZO
Vista Isométrica
Escala: 1 : 2
Cant. 06 Unidades

	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	Antebrazo 1			Nº PLANO: 08 - 016	REV: C

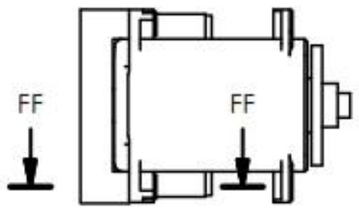




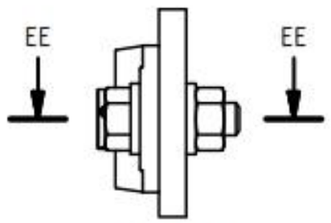
Detalle G
Escala: 1 : 1



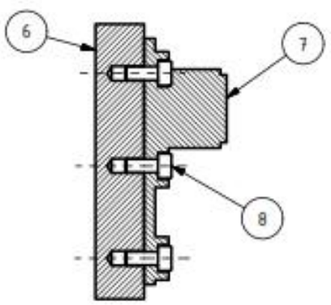
Sección DD - DD
Escala: 1 : 1



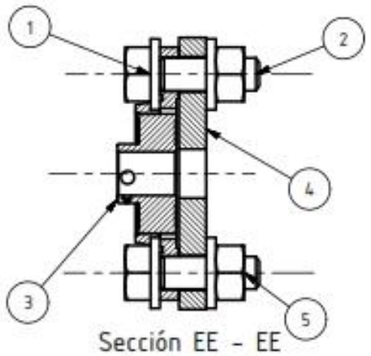
Detalle H
Escala: 1 : 1



Detalle I
Escala: 1 : 1



Sección FF - FF
Escala: 1 : 1



Sección EE - EE
Escala: 1 : 1

LISTA DE PARTES			
ITEM	CANT.	NOMBRE	DESCRIPCIÓM
1	12	Volanda PLana	ANSI B18.22M - 6M
2	6	Perno Hexagonal	ANSI B M6 x 1 x 20
3	3	Chumacera KFL08	
4	3	Base Chumacera	Plastico ABS
5	6	Tuerca Hexagonal	ANSI B M6 x 1
6	3	Base Servomotor	Plastico ABS
7	3	Dynamixel XM430 - W350 -T	
8	8	Tornillo Hexagonal de Cabeza Hueca	ISO 4762 - M2.5 x 6
9	24	Tuerca Hexagonal Grado A	ISO 4032 - M6
10	8	Base Fija	Plastico ABS
11	24	Tornillo Hexagonal Grado B	ISO 4015 - M6 x 25
12	48	Volanda Plana Gado A	ISO 7089 - 6 6 140 HV

	NOMBRES	FIRMA	FECHA
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018
REVISADO:			
ESCALA: INDICADAS	Ensamble Zona Superior		

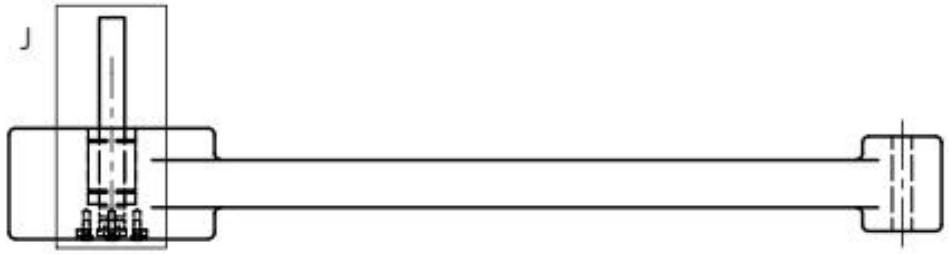
ROBOT PARALELO DELTA DE 3GDL

UNIVERSIDAD CATÓLICA DE SANTA MARIA

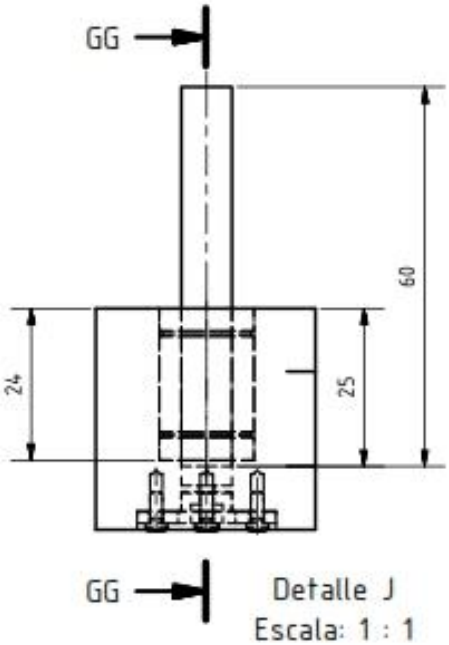
HOJA: A3

Nº PLANO: 09 - 016

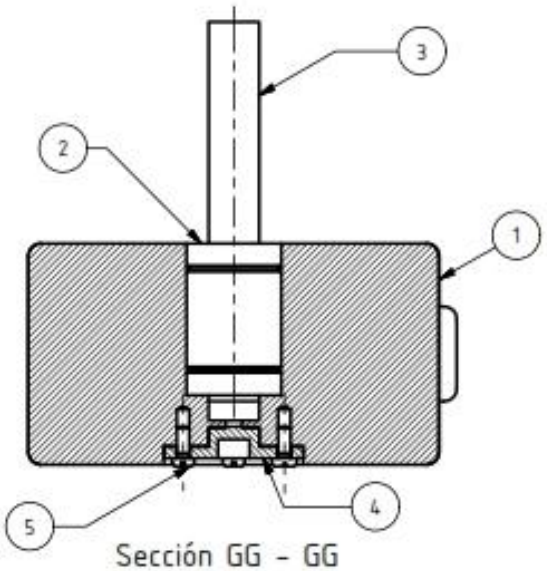
REV: C



Vista Superior
Escala: 1 : 2



Detalle J
Escala: 1 : 1



Sección GG - GG
Escala: 1 : 1

NOTA:
- Se realiza el mismo procedimiento para cada brazo.
- Insertar la junta en cada servomotor.

LISTA DE PARTES

ITEM	CANT.	Parte	DESCRIPCIÓN
1	3	Brazo	
2	3	Rodamiento 8 mm	
3	3	Eje Lineal	
4	3	Junta Servomotor	
5	12	JIS B 1101 - M2 x 5	Tornillo de Cabeza Ranurada

	NOMBRES	FIRMA	FECHA
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018
REVISADO:			

ROBOT PARALELO DELTA DE 3GDL

UNIVERSIDAD CATÓLICA DE SANTA MARIA

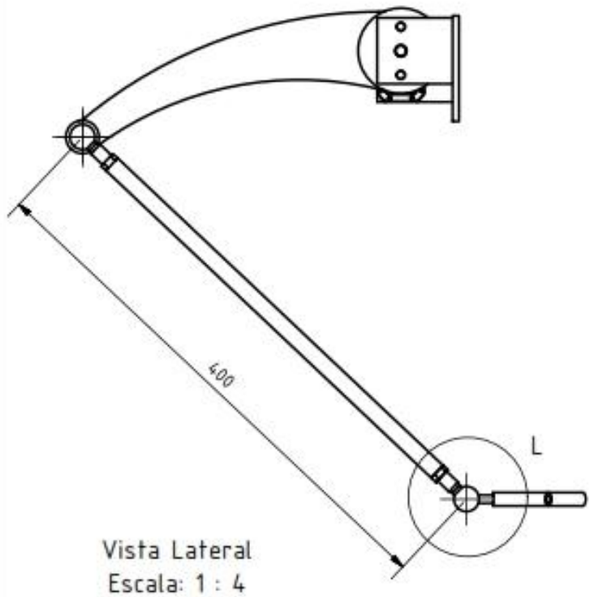
ESCALA: INDICADAS

Ensamble Servomotor a Brazo

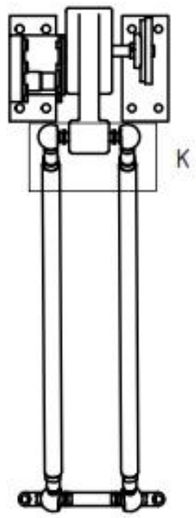
HOJA: A4

Nº PLANO: 010 - 016

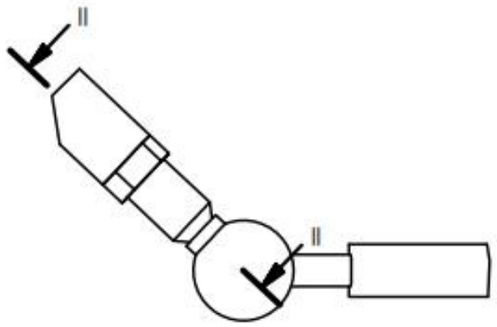
REV: C



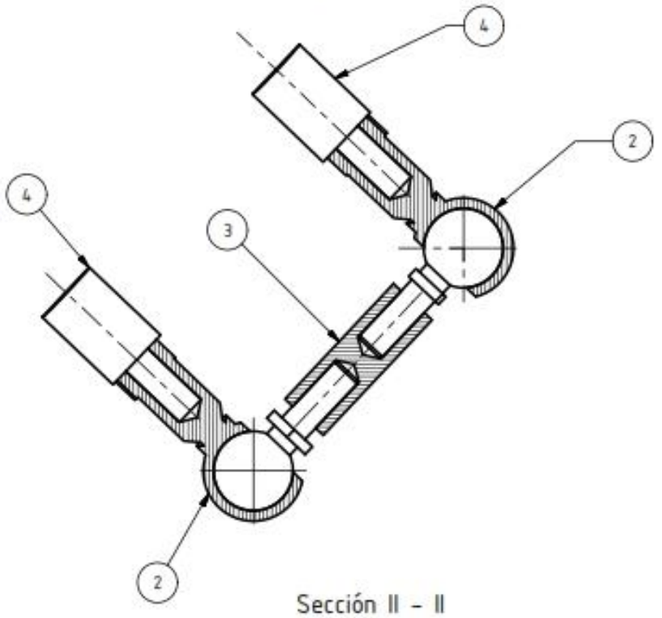
Vista Lateral
Escala: 1 : 4



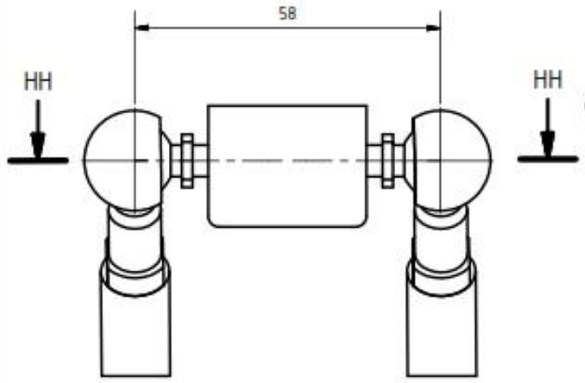
Vista Frontal
Escala: 1 : 4



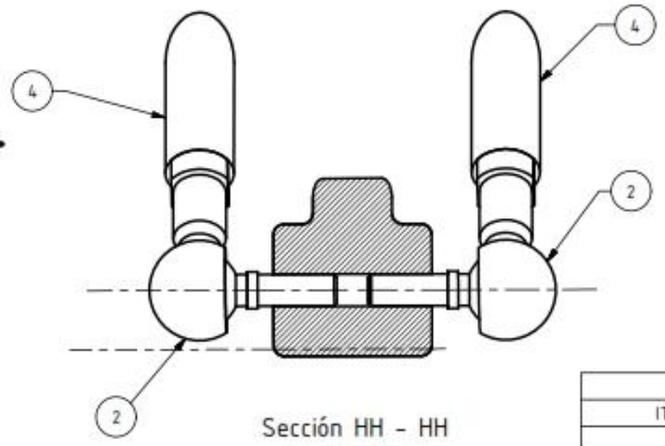
Detalle L
Escala: 1 : 1



Sección II – II
Escala: 1 : 1



Detalle K
Escala: 1 : 1

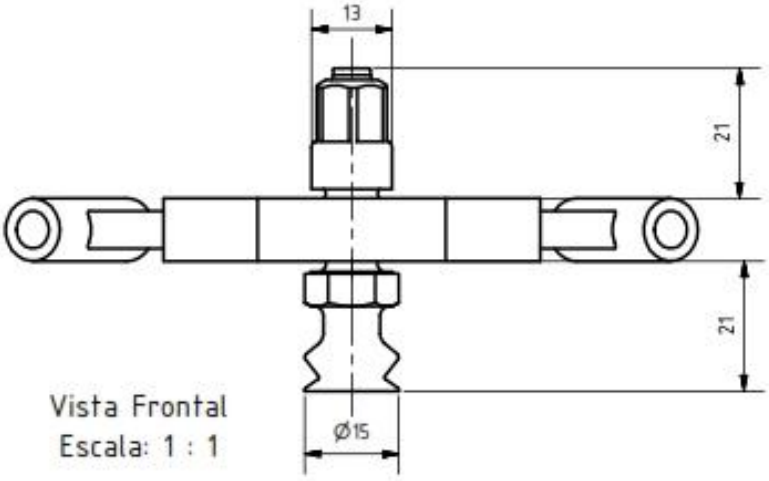


Sección HH – HH
Escala: 1 : 1

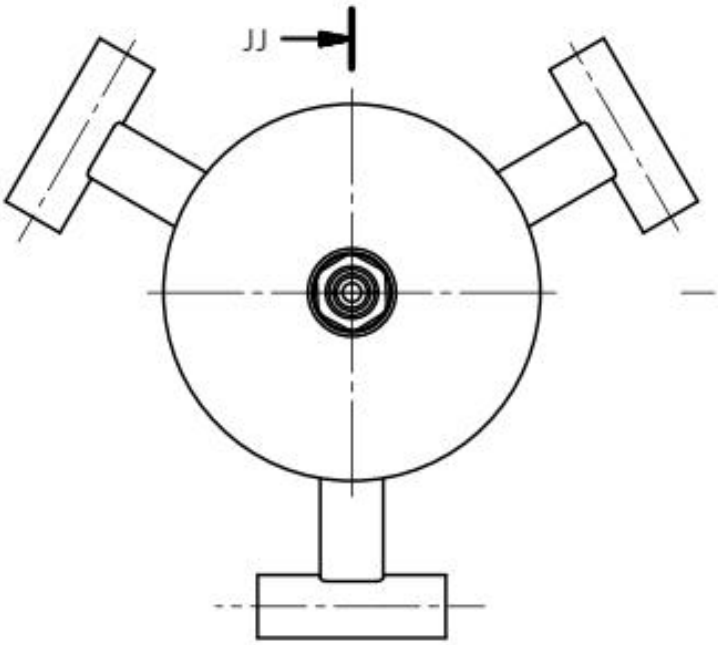
LISTA DE PARTES			
ITEM	CANT.	PARTE	DESCRIPCIÓN
1	3	Brazo	
2	12	Rótula Esférica SQ-6CRS	
3	1	Base Móvil	
4	6	Antebrazo 1	

NOTA:
-El ensamble de rotulas es igual para cada brazo y antebrazo.
- Insertar los antebrazos a las rotulas esféricas hasta que se alcance una distancia de 400 mm.
- La separación entre rotulas es de 58 mm para todos los eslabones.

	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018	
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA — HOJA: A3
ESCALA: INDICADAS	Ensamble de Rótulas			Nº PLANO: 011 - 016 — REV: C

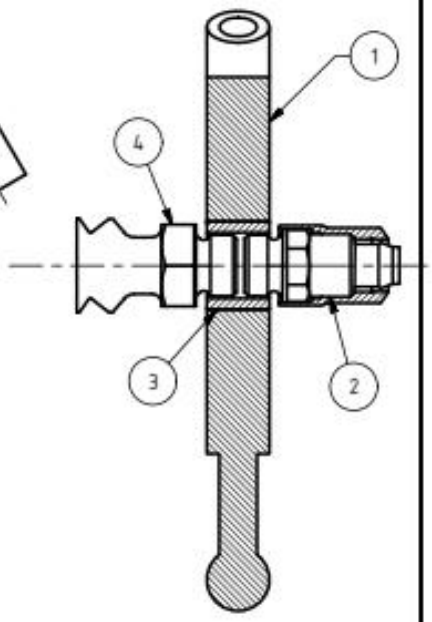


Vista Frontal
Escala: 1 : 1



JJ

Vista Superior
Escala: 1 : 1

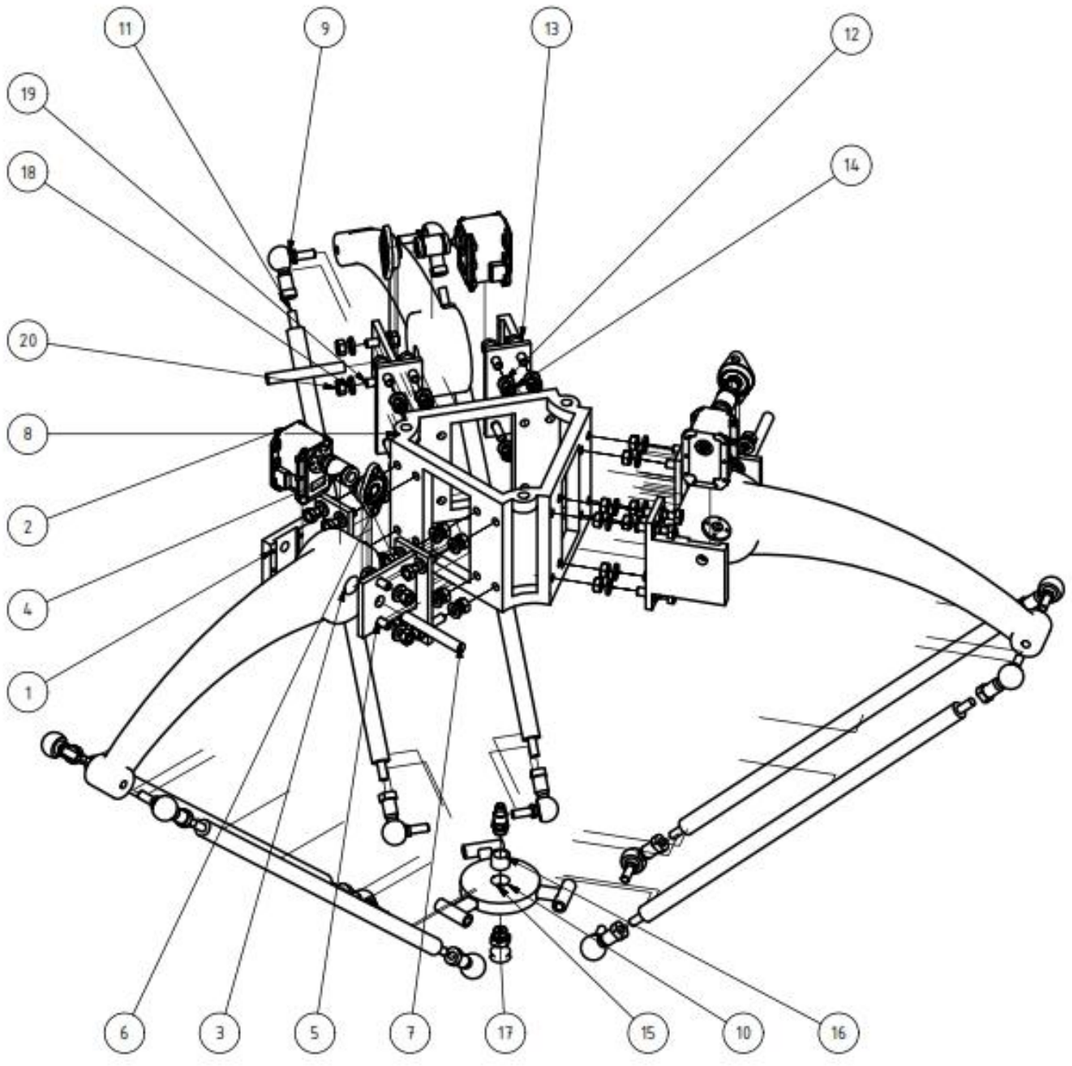


Sección JJ – JJ
Escala:1 : 1

LISTA DE PARTES

ITEM	CANT.	PARTE	DESCRIPCIÓN
1	1	Base Móvil	
2	1	Racor NPCK-C-D-G18-K8	Racor - Manguera 6 mm x 4 mm
3	1	Niple 14 mm x 10 mm x 10 mm	Niple Ventosa - Racor
4	1	VASB-15-1/8-SI-B	Ventosa de Succión VASB

	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018	
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA — HOJA: A4
ESCALA: INDICADAS	Ensamble Ventosa - Racor			Nº PLANO: 012 - 016 — REV: C



ROBOT PARALELO DELTA DE 3 GDL

Plano de Despiece

Escala: 1 : 4

LISTA DE PARTES

ITEM	CANT.	PARTES	DESCRIPCIÓN
1	3	Base Servomotor	
2	3	Dynamixel XM430-W350-T	
3	3	Brazo	
4	3	Rodamiento LM8UU	
5	3	Base_Chumacera	
6	3	Chumacera KFL08	
7	3	Eje Lineal	
8	1	Base Fija	
9	12	Rótula Esférica SQ-6CRS	
10	1	Base Móvil	
11	6	Antebrazo 1	
12	48	ISO 7089 - 6 - 140 HV	Volanda Plana Grado A
13	24	ISO 4015 - M6 x 25	Tornillo Hexagonal Grado B
14	24	ISO 4032 - M6	Tuerca Hexagonal Grado A
15	1	Racor NPCK-C-D-G18-K8	Racor - Manguera 6 mm x 4 mm
16	1	Niple 14 mm x 10 mm x 10 mm	Niple Ventosa - Racor
17	1	VASB-15-1/8-SI-B	Ventosa de Succión VASB
18	12	ANSI B18.22M - 6 N	Volanda Plana
19	6	ANSI B 18.6.7 M / IFI 513 - M6 x 1 x 20	Perno Hexagonal
20	6	ANSI B 18.2.4.1 M - M6 x 1	Tuerca Hexagonal

	NOMBRES	FIRMA	FECHA
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018
REVISADO:			
ESCALA: INDICADAS	ROBOT PARALELO DELTA DE 3GDL		

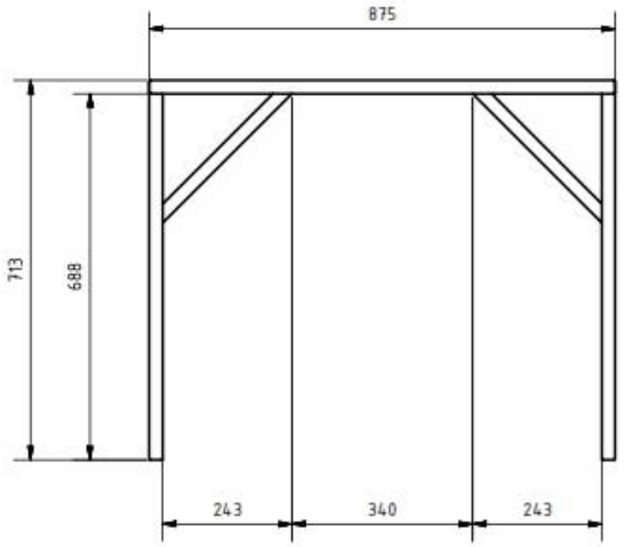
ROBOT PARALELO DELTA DE 3GDL

UNIVERSIDAD CATÓLICA DE SANTA MARIA

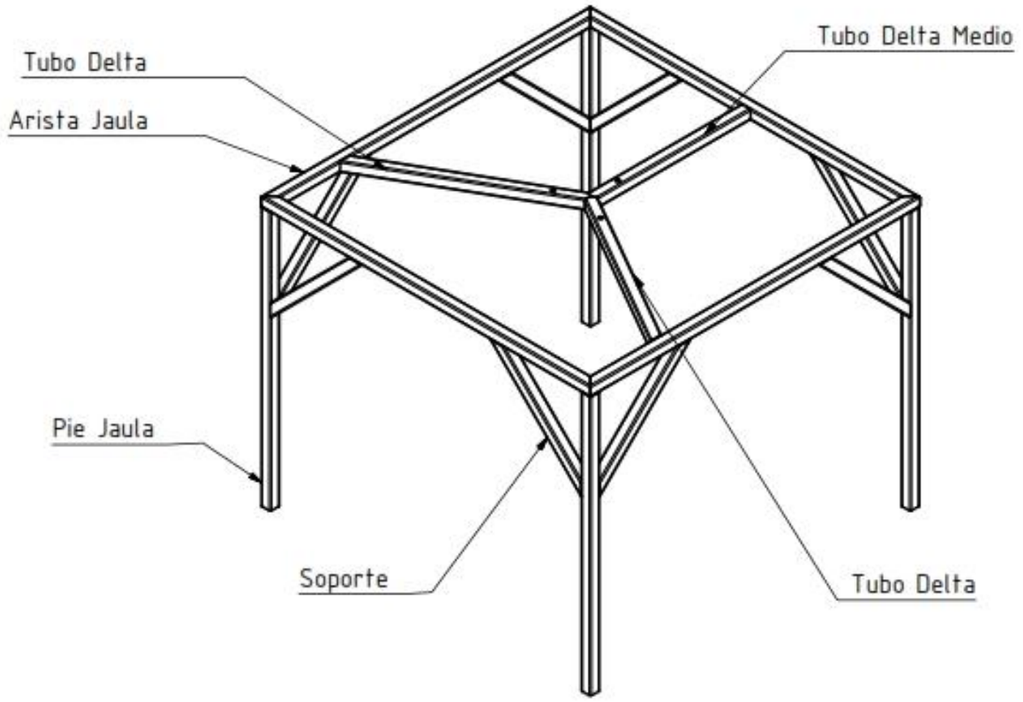
HOJA: A3

Nº PLANO: 013 - 016

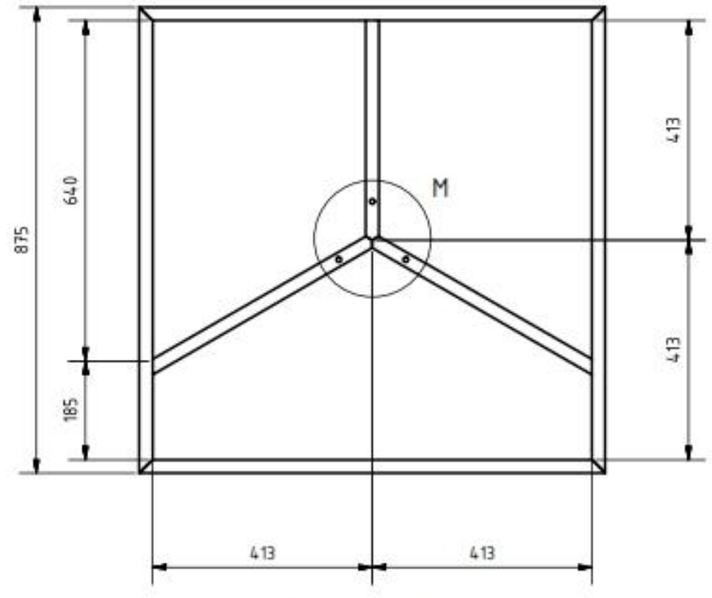
REV: C



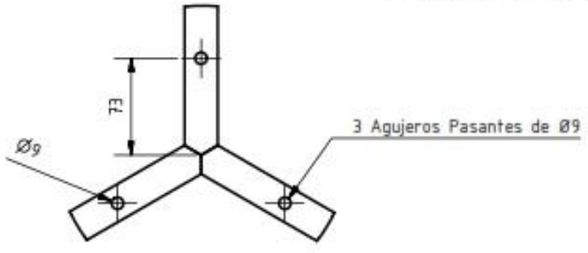
Vista Frontal
Escala: 1 : 10



JAULA DEL ROBOT PARALELO DELTA
Vista Isométrica
Escala: 1 : 10
Material: Acero HSS

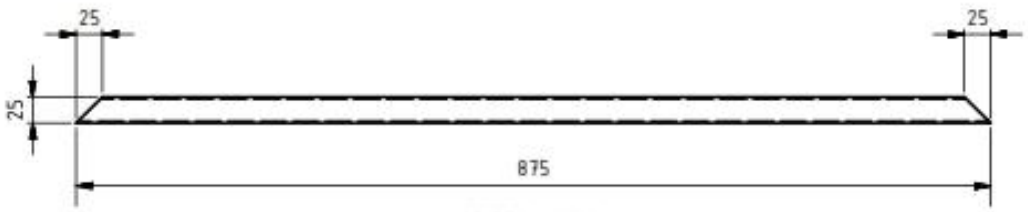


Vista Superior
Escala: 1 : 10

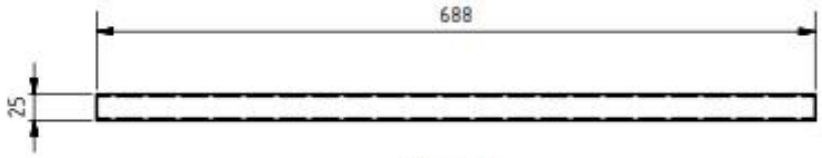


Detalle M
Escala: 1 : 4

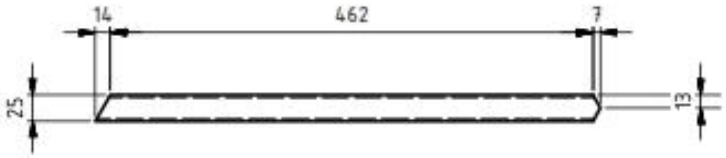
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A3
ESCALA: INDICADAS	Jaula_Robot			Nº PLANO: 014 - 016	REV: C



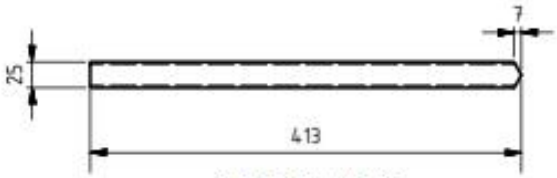
ARISTA JAULA
Escala: 1 : 5
Cant. 04 Unidades



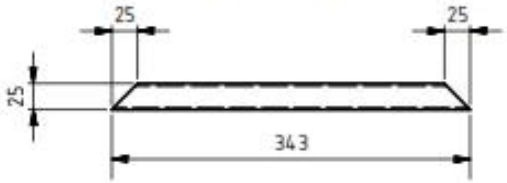
PIE JAULA
Escala: 1 : 5
Cant. 04 Unidades



TUBO DELTA
Escala: 1 : 5
Cant. 02 Unidades



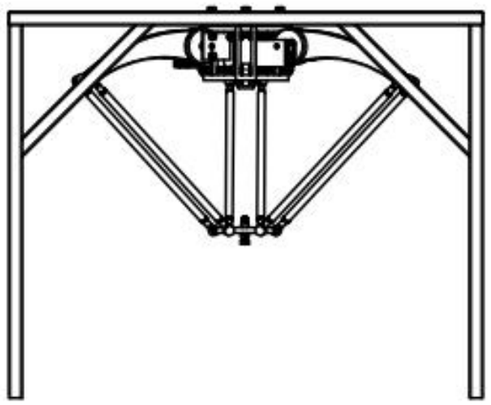
TUBO DELTA MEDIO
Escala: 1 : 5
Cant. 01 Unidad



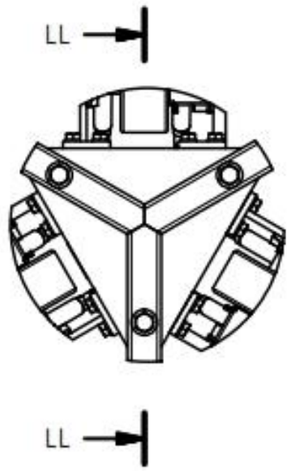
SOPORTE
Escala: 1 : 5
Cant. 08 Unidades

NOTA:
- Todas las piezas que componen la jaula del robot son de Tubo Cuadrado de Acero HSS 25 mm x 25 mm x 2 mm

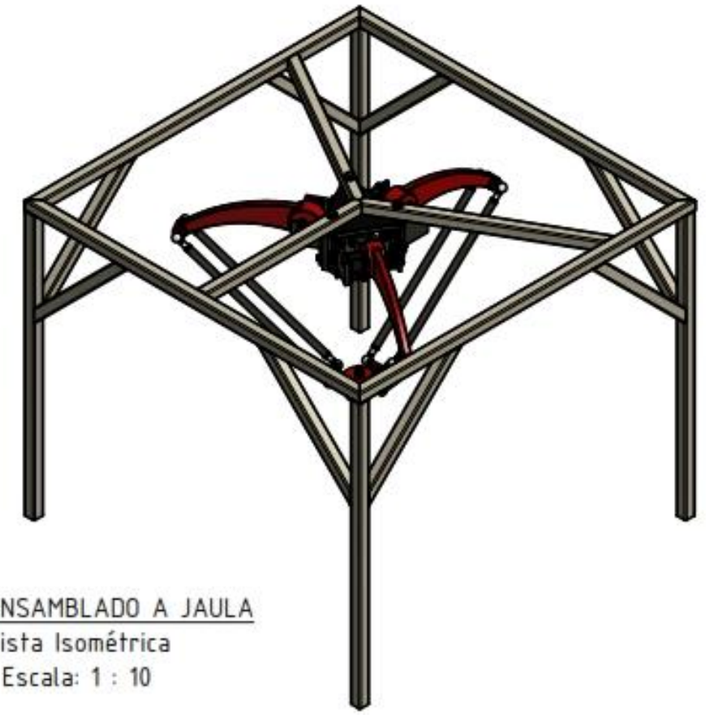
	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A4
ESCALA: INDICADAS	Jaula_Robot			Nº PLANO: 015 - 016	REV: C



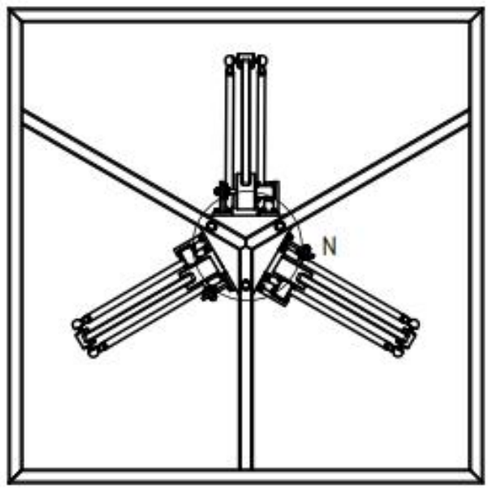
Vista Frontal
Escala: 1 : 10



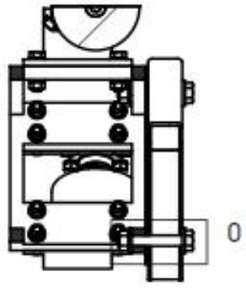
Detalle N
Escala: 1 : 4



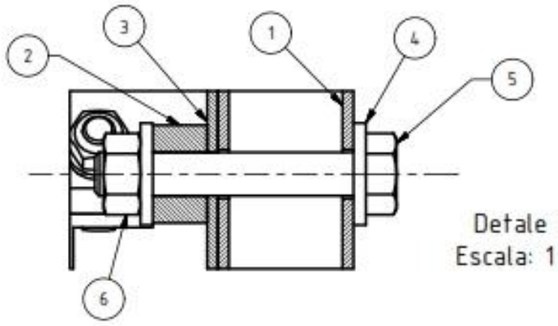
ROBOT ENSAMBLADO A JAULA
Vista Isométrica
Escala: 1 : 10



Vista Superior
Escala: 1 : 10



Sección LL - LL
Escala: 1 : 4



Detale 0
Escala: 1 : 1

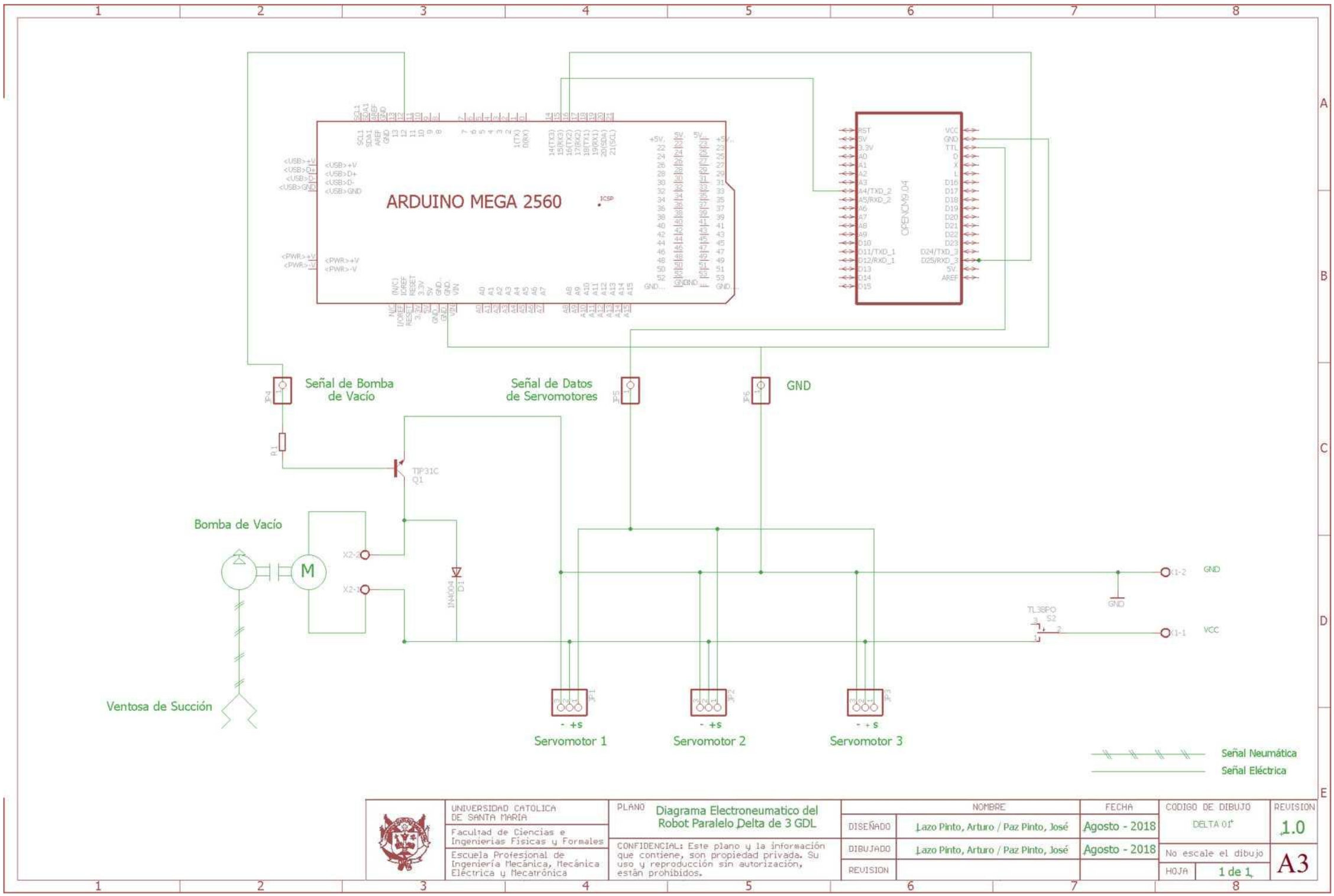
LISTA DE PARTES			
ITEM	CANT.	PARTE	DESCRIPCIÓN
1	-	Jaula del Robot	Tubo Cuadrado 25 x 25 x 2
2	1	Base Fija	
3	1	PLancha Jaula Triangular	
4	6	ANSI B18.22M - 8 N	Volanda Plana
5	3	ANSI B18.2.3.5M - M8 x 1.25 x 50	Perno Hexagonal
6	3	ANSI B 18.2.4.1 M - M8 x 1.25	Tuerca Hexagonal

NOTA:
- Repetir la mimsma conexión de pernos para cada esquina de la Base Fija.

	NOMBRES	FIRMA	FECHA	ROBOT PARALELO DELTA DE 3GDL	
DIBUJADO:	LAZO PINTO, Arturo Alonso / PAZ PINTO, José Augusto	A.L.P / J.P.P	6/08/2018		
REVISADO:				UNIVERSIDAD CATÓLICA DE SANTA MARIA	HOJA: A3
ESCALA: INDICADAS	Ensamble Jaula - Robot			Nº PLANO: 016 - 016	REV: C

ANEXO 04

PLANO ELECTRO NEUMÁTICO



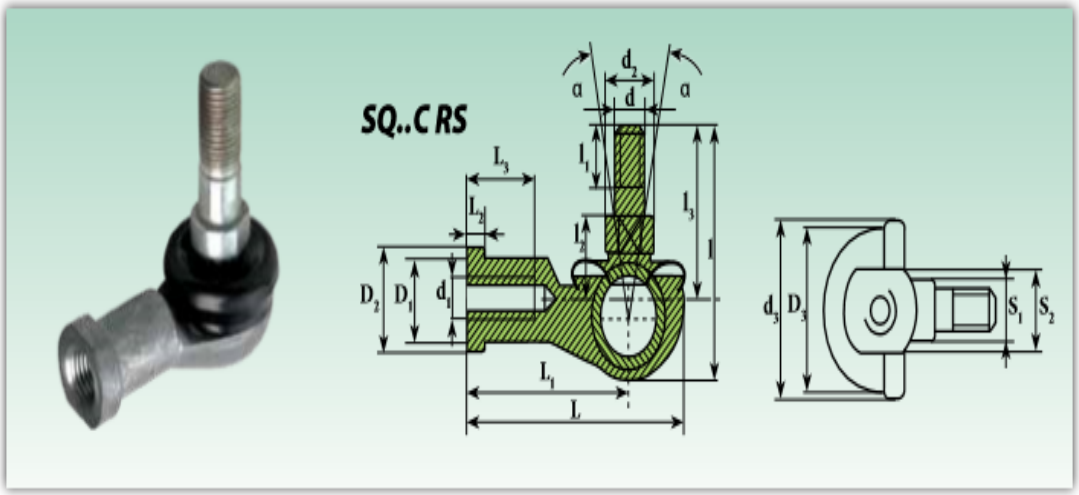
UNIVERSIDAD CATOLICA DE SANTA MARIA Facultad de Ciencias e Ingenierias Fisicas y Formales Escuela Profesional de Ingeniería Mecánica, Mecánica Eléctrica y Mecatrónica	PLANO: Diagrama Electroneumatico del Robot Paralelo Delta de 3 GDL	NOMBRE		FECHA	CODIGO DE DIBUJO: DELTA 01	REVISION: 1.0
		DISEÑADO	Lazo Pinto, Arturo / Paz Pinto, José	Agosto - 2018		
	CONFIDENCIAL: Este plano y la información que contiene, son propiedad privada. Su uso y reproducción sin autorización, están prohibidos.	DIBUJADO	Lazo Pinto, Arturo / Paz Pinto, José	Agosto - 2018	No escale el dibujo	A3
		REVISION			HOJA: 1 de 1	

ANEXO 05

INFORMACIÓN DE LAS ROTULAS SQ-6GRS

Cabezas de articulación angulares

Winding shape ball joint rod ends

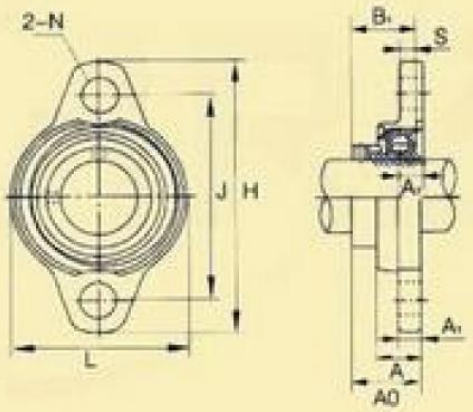


Sigla [1] *Designation* [1]	Dimensiones mm *Dimensions mm*																	Grados *Degrees*	Carga *Ratings load*		Peso *Weight*
con obturación *with seal*	d	d_1	d_2	d_3 *	D_1	D_2	D_3	l	l_1	l_2	l_3	L	L_1	L_2	L_3	S_1	S_2	≈ α	Dinámico C *Dinamyc C* KN	Estático C_0 *Static C_0* KN	≈ Kg
SQ 5 C RS	**5**	M 5x0,8	9	20	9	11	16	29	8	10	21	35	27	4	14	7	9	25	2,7	9,2	0,026
SQ 6 C RS	**6**	M 6x1	10	20	10	13	19	35,5	11	11	26	40	30	5	14	8	11	25	3,6	12	0,039
SQ 8 C RS	**8**	M 8x1,25	12	24	12,5	16	23	42,5	12	14	31	48	36	5	17	10	14	25	5,7	19	0,068
SQ 10 C RS	**10**	M 10x1,25	14	30	15	19	27	50,5	15	17	37	57	43	6,5	21	11	17	25	8,2	27	0,112
SQ 10 C RS-1	**10**	M 10x1,5	14	30	15	19	27	56,5	21	17	43	57	43	6,5	21	11	17	25	8,2	27	0,112
SQ 12 C RS	**12**	M 12x1,25	17	32	17,5	22	31	57,5	17	19	42	66	50	6,5	25	15	19	25	11	37	0,164
SQ 12 C RS-1	**12**	M 12x1,75	17	32	17,5	22	31	64,5	24	19	49	66	50	6,5	25	15	19	25	11	37	0,164
SQ 14 C RS	**14**	M 14x1,5	19	38	20	25	35	73,5	22	21,5	56	75	57	8	26	17	22	25	14	48	0,254
SQ 14 C RS-1	**14**	M 14x2	19	38	20	25	35	79,5	28	21,5	62	75	57	8	26	17	22	25	14	48	0,254
SQ 16 C RS	**16**	M 16x1,5	22	44	22	27	39	79,5	23	23,5	60	84	64	8	32	19	22	20	16	53	0,336
SQ 16 C RS-1	**16**	M 16x2	22	44	22	27	39	85,5	29	23,5	66	84	64	8	32	19	22	20	16	53	0,336
SQ 18 C RS	**18**	M 18x1,5	23	45	25	31	44	90	25	26,5	68	93	71	10	34	20	27	20	18	61	0,464
SQ 20 C RS	**20**	M 20x1,5	27	50	27,5	34	44	90	25	27	68	99	77	10	35	24	30	20	18	61	0,538
SQ 22 C RS	**22**	M 22x1,5	27	52	30	37	50	95	26	28	70	109	84	12	41	24	32	16	22	75	0,713

Bajo demanda, disponibles en acero inoxidable
Under request, stainless steel available

ANEXO 06

INFORMACIÓN DE LA CHUMACERA M8 – KFL08 Y RODAMIENTO LM8UU

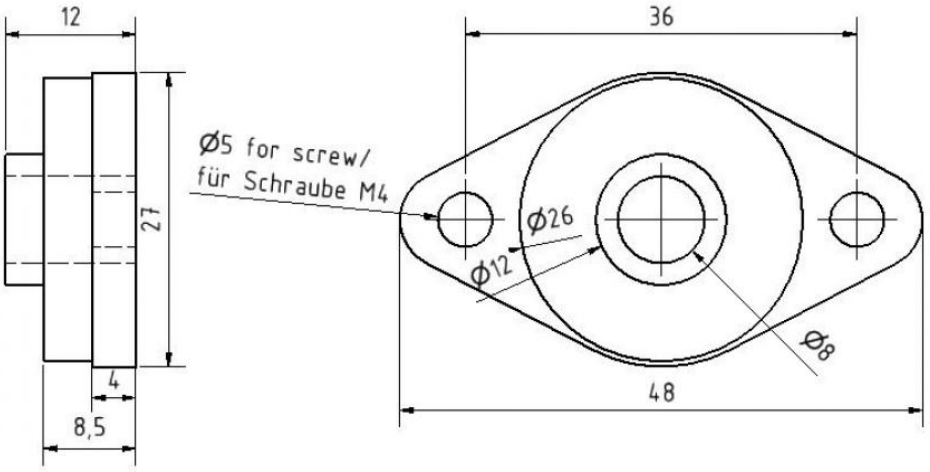


带座轴承代号 Bearing Unit No.	轴 承 Shaft Dia d	外形尺寸（毫米） Dimensions (mm)											
	(mm)	H	J	A1	A2	A	N	L	A0	B	S	C11	Zs
KFL08	8	48	36	4	4	8.5	5	27	12	11	3	4	12
KFL000	10	60	45	5.5	5.5	11.5	7	36	15.5	14	4	6	22
KFL001	12	63	48	5.5	5.5	11.5	7	38	16	14.5	4	6.5	22

Hinweis:
Alle Abmessungen in mm
Für die Richtigkeit der Zeichnung und CAD Dateien kann nicht garantiert werden (wegen möglichen Änderungen durch Lieferanten). Wir bemühen uns, diese zeitnah anzupassen.

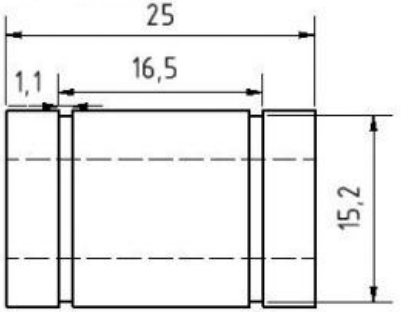
Note:
All dimensions in mm
We can't garantee the accuracy of the drawing and CAD files (because of changes by supplier). We try to adapt this realtime.

www.motedis.com



Note:
All dimensions in mm

Bemerkung:
Alle Abmessungen in mm



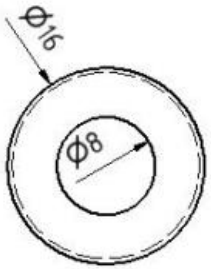


Abbildung ähnlich

Innendurchmesser: 8mm 8µ/-0µ (dr)
Außendurchmesser: 16mm 0µ/-9µ (D)
Länge: 25mm 0µ/-200µ (L)
Abstand der Nuten: 16,5mm 0µ/-200µ (B)
Nutenbreite: 1,1mm (W)
Durchmessser der Nuten: 15,2mm (D1)
Anzahl der Kugelreihen: 4
Die Lager sind beideitig abgedichtet.
Belastbarkeit:
dynamisch CoN : 265N
statisch CoN: 402N

Inner diameter: 8mm 8µ /-0µ (dr)
Outer diameter: 16mm 0µ /-9µ (D)
Length: 25mm 0µ /-200µ (L)
Spacing of the grooves: 16,5mm 0µ /-200µ (B)
Groove width: 1,1mm (W)
diametre of Slot: 15,2mm (D1)
Number of ball rows: 4
The bearings are sealed on both sides.
Resilience:
dynamically CoN: 265N
static CoN: 402N

Für die Richtigkeit der Zeichnung und CAD Dateien kann nicht garantiert werden (wegen möglichen Änderungen durch Lieferanten). Wir bemühen uns, diese zeitnah anzupassen.

We can`t garantee the accuracy of the drawing and CAD files (because of changes by supplier). We try to adapt this realtime.

ANEXO 07

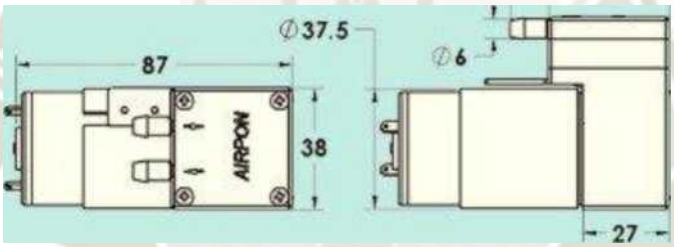
DATASHEET DE BOMBA DE VACÍO AIRPO Y VENTOSA DE SUCCIÓN FESTO VASB -15 -1/8

AIR PUMP SERIES FOREVER

D2028 Pump Specifications

Free Flow Range..... 12-15LPM
Vacuum Range.....0-16"TIg
Pressure Range 0-32PSI
Standard Motor Voltage Options 12VDC
Motor Construction Option.....Iron Core-Oil Bearing
Diaphragm Material Option.....EPDM
Operating Temperature Range.....32°-120°F (0°-50°)

12 ------ 38 -



Suction cup complete
VASB-15-1/8-PUR-B

Part number: 1395671

FESTO

Data sheet

Feature	Value
Suction cup height compensator	5.6 mm
Nominal size	3 mm
suction cup diameter	15 mm
suction cup volume	0.83 cm3
Effective suction diameter	12.4 mm
Position of connection	on top
Assembly position	Any
Suction cup shape	Round, bellows 1.5 conv
Working pressure	-0.95 ... 0 bar
Nominal working pressure	-0.7 bar
Operating medium	Atmospheric air based on ISO 8573-1:2010 [7:-:-]
Corrosion resistance classification CRC	2 - Moderate corrosion stress
Ambient temperature	-20 ... 60 °C
Retention force at nominal operating pressure	8.5 N
Product weight	11 g
Mounting type	Via vacuum port
Vacuum connection	G1/8
Color	blue
Shore hardness	60 +/- 5
Material of threaded plug	Brass
Materials note	Conforms to RoHS
Material suction cup	PUR

ANEXO 08

CODIGO ESTRUCTURADO DEL CONTROLADOR MAESTRO Y EL CONTROLADOR ESCLAVO

CONTROLADOR MAESTRO – ARDUINO MEGA

```
#include <MatrixMath.h>
    float Px;float Py;float Pz;
    float pxi ;float Vx;  float vdx;
    float pyi; float Vy;  float vdy;
    float pzi; float Vz;  float vdz;
    float pi=3.141592653;
    float th11;float th21;float th31;
    int a=0;int i=1;int m=0;float d=0.005;
    float R=0.1;float r=0.05;float R1=R-r;
    float L1=0.25;float L2=0.4;
    float phi1=0*(pi/180);float phi2=120*(pi/180);float phi3=240*(pi/180);
    int Q1; int Q2; int Q3;
    int xx; int x;int yy; int y;int zz; int z;
    boolean feedback = true;
    boolean debug = true;
    int POS;
    const byte numChars = 20; const byte posChars = 67;
    char receivedChars[numChars]; char receivedPos[posChars];
    int data_nueva = 1;
    int  vel3c=0; int vel3b=5; int vel3a=0; int vel2c=0; int vel2b=5; int vel2a=0; int
vel1c=0; int vel1b=5; int vel1a=0;
    double
MU[41]={0,0.0002,0.0012,0.0038,0.0086,0.0161,0.0266,0.0405,0.0579,0.0789,0.1
035,0.1316,0.1631,0.1977,0.2352,0.2752,0.3174,0.3615,0.4069,0.4532,0.5000,0.
5468,0.5931,0.6385,0.6826,0.7248,0.7648,0.8023,0.8369,0.8684,0.8965,0.9211,0
.9421,0.9595,0.9734,0.9839,0.9914,0.9962,0.9988,0.9998,1};
```

```
double MUD[41]={0,0.0089,0.0338,0.0722,0.1215,0.1794,0.2438,0.3127,0.3840,0.4561,0.5273,0.5963,0.6615,0.7219,0.7763,0.8240,0.8640,0.8958,0.9188,0.9328,0.9375,0.9328,0.9188,0.8958,0.8640,0.8240,0.7763,0.7219,0.6615,0.5963,0.5273,0.4561,0.3840,0.3127,0.2438,0.1794,0.1215,0.0722,0.0338,0.0089,0};
float QQ1f[1][41]={7.5781,7.5882,7.6559,7.8303,8.1514,8.6498,9.3472,10.2563,11.3812,12.7187,14.2593,15.9880,17.8859,19.9316,22.1021,24.3736,26.7228,29.1269,31.5639,34.0131,36.4549,38.8705,41.2423,43.5531,45.7864,47.9263,49.9572,51.8642,53.6328,55.2499,56.7039,57.9851,59.0869,60.0062,60.7443,61.3079,61.7093,61.9672,62.1071,62.1613,62.1694};
float QQ2f[1][41]={7.5781,7.5821,7.6091,7.6789,7.8077,8.0091,8.2936,8.6693,9.1419,9.7150,10.3901,11.1671,12.0442,13.0180,14.0837,15.2352,16.4651,17.7651,19.1256,20.5361,21.9850,23.4601,24.9483,26.4356,27.9074,29.3487,30.7441,32.0780,33.3353,34.5014,35.5628,36.5082,37.3285,38.0179,38.5748,39.0019,39.3070,39.5036,39.6103,39.6517,39.6579};
float QQ3f[1][41]={7.5781,7.5821,7.6091,7.6789,7.8077,8.0091,8.2936,8.6693,9.1419,9.7150,10.3901,11.1671,12.0442,13.0180,14.0837,15.2352,16.4651,17.7651,19.1256,20.5361,21.9850,23.4601,24.9483,26.4356,27.9074,29.3487,30.7441,32.0780,33.3353,34.5014,35.5628,36.5082,37.3285,38.0179,38.5748,39.0019,39.3070,39.5036,39.6103,39.6517,39.6579};
float dqm1f[1][41]={0,1,2,5,8,11,15,19,24,29,34,39,45,50,56,62,68,73,79,85,90,96,101,106,110,114,117,120,120,119,116,111,102,90,75,59,42,26,13,4,0};
float dqm2f[1][41]={0,1,1,3,4,6,8,10,12,14,17,19,22,25,27,30,32,34,36,38,39,40,40,40,40,39,38,36,34,31,28,25,21,18,14,10,7,5,2,1,0};
float dqm3f[1][41]={0,1,1,3,4,6,8,10,12,14,17,19,22,25,27,30,32,34,36,38,39,40,40,40,40,39,38,36,34,31,28,25,21,18,14,10,7,5,2,1,0};
int Q3d;int Q3c;int Q3b;int Q3a;
```

```
int Q2d;int Q2c;int Q2b;int Q2a;
int Q1d;int Q1c;int Q1b;int Q1a;
int V1c;int V1b;int V1a;
int V2c;int V2b;int V2a;
int V3c;int V3b;int V3a;
float k1[1][1];float k2[1][1];float k3[1][1]; float Jacob[3][3]; float Velq[3][1];float Vel[3][1]; float QQ1[41][1];float QQ2[41][1];float QQ3[41][1];
float Velq1[1][41];float Velq2[1][41];float Velq3[1][41];
int TT;
int TTT=0;
int cartesiano;
boolean newData = false;
boolean newPos = false;
int ventosa;
unsigned int posx;unsigned int posy;unsigned int posz;

byte UPpin = 16;byte DOWNpin = 15;byte LEFTpin = 14;byte RIGHTpin = 17;
byte OKpin = 18;
int luz = 13;
int val;
int b;

unsigned long tiempo = 0;
unsigned long t_actualizado = 0;
unsigned long t_delay = 40;

unsigned long tiempo2 = 0;
unsigned long t_actualizado2 = 0;
unsigned long t_delay2 = 50;

void setup()
```

```
{
    pinMode(13, OUTPUT);
     Serial.begin(57600);
    Serial2.begin(500000);
    Serial3.begin(57600);
    Pz = 0.3;
    Px=0;
    Py=0;
    sendStartMessage();
}

void loop()
{
    if (Serial.available() > 0)     {  recvWithStartEndMarkers();}
    if (newData) { parseData(); }
}
void sendStartMessage()
{
Serial.println(" ");
Serial.println("arduinoVBserialControl Ver 1.0");
Serial.println(" ");
Serial.println("DON = debug on");
Serial.println("DOF = debug off");
Serial.println("START to reset");
Serial.println(" ");
if (debug) { Serial.println("Debug is on");}
else { Serial.println("Debug is off"); }
Serial.println(" ");}

void parseData(){
  newData = false;
```

```
    newPos = false;
    if (debug) {
    //---------------ENVIAMOS DATOS A VISUAL -------------------
    Serial.println( receivedChars );
    Serial.print("*");Serial.print( Px);Serial.print("*");Serial.print( Py
);Serial.print("*");Serial.print( Pz );Serial.println( receivedPos );
    }
    if (strcmp(receivedChars, "HELLO")  == 0)
    {Serial.println("HELLO");
    digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,LOW);}
    if (strcmp(receivedChars, "START")  == 0)
    {sendStartMessage();}
    if  ( receivedChars[0] == 'G' )
       {
         digitalWrite(13,HIGH);
       }

    if  ( receivedChars[0] == 'H' )
       {
         digitalWrite(13,LOW);
       }
    if  ( receivedChars[0] == 'W' )
    {
    Serial2.print('<');Serial2.print("L123456789123456789123");Serial2.print('>');}
    //-----------------DATA JOYSTICK----------------------
    if  ( receivedChars[0] == 'J' )
    {
      x = convertToNumber( 1 );y = convertToNumber( 5 );z =  convertToNumber( 9
); ventosa=convertToButton(13);
```

```
if (ventosa==1){
  digitalWrite(12,HIGH);}
  else {
  digitalWrite(12,LOW);
}
if (x<2000){xx=1;}
else if (x>2096){xx=2;}
else if (2000<=x<=2096){xx=0;}
if (y<2000){yy=1;}
else if (y>2096){yy=2;}
else if (2000<=y<=2096){yy=0;}
if (z==0){zz=1;}
else if (z==4096){zz=2;}
else if (10<z<4090){zz=0;}
POS=xx+(10*yy)+(100*zz);
tiempo = millis();
if ( tiempo > t_actualizado + t_delay) {
//Se actualiza el tiempo que ha de transcurrir para el próximo delay.
t_actualizado = tiempo;
//---------------ACTUALIZAMOS POSICION ------------------
switch(POS){
case (1)://X=NEGATIVO
Px=Px-d;
break;
case (2)://X=POSITIVO
Px=Px+d;
break;
case (10)://Y=NEGATIVO
Py=Py-d;
break;
case (20)://Y=POSITIVO
```

```
Py=Py+d;
break;
case (100)://Z=NEGATIVO
Pz=Pz-d;
break;
case (200)://Z=POSITIVO
Pz=Pz+d;
break;
case (11):   //X=NEGATIVO && Y=NEGATIVO
Px=Px-d;Py=Py-d;
break;
case (21):   //X=NEGATIVO && Y=POSITIVO
Px=Px-d;Py=Py+d;
break;
case (101):  //X=NEGATIVO && Z=NEGATIVO
Px=Px-d;Pz=Pz-d;
break;
case (201):  //X=NEGATIVO && Z=POSITIVO
Px=Px-d;Pz=Pz+d;
break;
case (12):   //X=POSITIVO && Y=NEGATIVO
Px=Px+d;Py=Py-d;
break;
case (22):   //X=POSITIVO && Y=POSITIVO
Px=Px+d;Py=Py+d;
break;
case (102):  //X=POSITIVO && Z=NEGATIVO
Px=Px+d;Pz=Pz-d;
break;
case (202):  //X=POSITIVO && Z=POSITIVO
Px=Px+d;Pz=Pz+d;
```

```
break;
case (110):   //Y=NEGATIVO && Z=NEGATIVO
Py=Py-d;Pz=Pz-d;
break;
case (210):   //Y=NEGATIVO && Z=POSITIVO
Py=Py-d;Pz=Pz+d;
break;
case (120):  //Y=POSITIVO && Z=NEGATIVO
Py=Py+d;Pz=Pz-d;
break;
case (220):  //Y=POSITIVO && Z=POSITIVO
Py=Py+d;Pz=Pz+d;
break;
case (221):  //X=NEGATIVO && Y=POSITIVO && Z=POSITIVO
Px=Px-d;Py=Py+d;Pz=Pz+d;
break;
case (121):  //X=NEGATIVO && Y=POSITIVO && Z=NEGATIVO
Px=Px-d;Py=Py+d;Pz=Pz-d;
break;
case (211):  //X=NEGATIVO && Y=NEGATIVO && Z=POSITIVO
Px=Px-d;Py=Py-d;Pz=Pz+d;
break;
case (111):  //X=NEGATIVO && Y=NEGATIVO && Z=NEGATIVO
Px=Px-d;Py=Py-d;Pz=Pz-d;
break;
case (222):  //X=POSITIVO && Y=POSITIVO && Z=POSITIVO
Px=Px+d;Py=Py+d;Pz=Pz+d;
break;
case (122):  //X=POSITIVO && Y=POSITIVO && Z=NEGATIVO
Px=Px+d;Py=Py+d;Pz=Pz-d;
break;
```

```
case (212):  //X=POSITIVO && Y=NEGATIVO && Z=POSITIVO
Px=Px+d;Py=Py-d;Pz=Pz+d;
break;
case (112):  //X=POSITIVO && Y=NEGATIVO && Z=NEGATIVO
Px=Px+d;Py=Py-d;Pz=Pz-d;
break;
case (0):    //X=CENTRO && Y=CENTRO && Z=CENTRO
break;
}
if (Px>0.25){Px=0.25;}
else if (-0.25>Px){Px=-0.25;}
if (Py>0.3){Py=0.3;}
else if (-0.25>Py){Py=-0.25;}
if (Pz>0.6){Pz=0.6;}
else if (0>Pz){Pz=0;}
}
icinematica();//Enviar angulos validos
Q3d=Q3/1000;
Q3c=(Q3-(Q3d*1000))/100;
Q3b=(Q3- (Q3d*1000 + Q3c*100))/10;
Q3a=Q3-(Q3d*1000 + Q3c*100 + Q3b*10 );
Q2d=Q2/1000;
Q2c=(Q2-(Q2d*1000))/100;
Q2b=(Q2- (Q2d*1000 + Q2c*100))/10;
Q2a=Q2-(Q2d*1000 + Q2c*100 + Q2b*10 );
Q1d=Q1/1000;
Q1c=(Q1-(Q1d*1000))/100;
Q1b=(Q1- (Q1d*1000 + Q1c*100))/10;
Q1a=Q1-(Q1d*1000 + Q1c*100 + Q1b*10 );
//---------------ENVIAMOS DATOS A OPEN -------------------
Serial2.print('<');Serial2.print("P");
```

```
Serial2.print(Q1d);Serial2.print(Q1c);Serial2.print(Q1b);Serial2.print(Q1a);
Serial2.print(Q2d);Serial2.print(Q2c);Serial2.print(Q2b);Serial2.print(Q2a);
Serial2.print(Q3d);Serial2.print(Q3c);Serial2.print(Q3b);Serial2.print(Q3a);
Serial2.print(vel1c);Serial2.print(vel1b);Serial2.print(vel1a);
Serial2.print(vel2c);Serial2.print(vel2b);Serial2.print(vel2a);
Serial2.print(vel3c);Serial2.print(vel3b);Serial2.print(vel3a);
Serial2.println('>');
}
if  ( receivedChars[0] == 'V' )
{    int vel1;int vel2; int vel3;
vel1 = convertToNumber2( 1 );
vel2 = convertToNumber2( 4 );
vel3 = convertToNumber2( 7 );
vel3c=(vel3)/100;
vel3b=(vel3- (vel3c*100))/10;
vel3a=vel3-(vel3c*100 + vel3b*10 );
vel2c=(vel2)/100;
vel2b=(vel2- (vel2c*100))/10;
vel2a=vel2-(vel2c*100 + vel2b*10 );
vel1c=(vel1)/100;
vel1b=(vel1-(vel1c*100))/10;
vel1a=vel1-(vel1c*100 + vel1b*10 );
}
//-----------------FIN DATA JOYSTICK----------------------
//-----------------INICIO MODO CASA----------------------
if  ( receivedChars[0] == 'C' )
{
//---------------ENVIAMOS DATOS A OPEN -------------------
  Serial2.print('<');Serial2.print("C");
  Serial2.print("1110");Serial2.print("0020");
  Serial2.println('>');
```

```
}
//-------------------FIN MODO CASA-----------------------
//------------------INICIO MODO FIN----------------------
if  ( receivedChars[0] == 'F' )
{
//---------------ENVIAMOS DATOS A OPEN -------------------
  Serial2.print('<');Serial2.print("F");
  Serial2.print("1779");Serial2.print("0020");
  Serial2.println('>');
}
//-------------------FIN MODO FIN-----------------------
//---------------INICIO MODO ARTICULAR-------------------
if  ( receivedChars[0] == 'A' )
{
data_nueva=1;
if (data_nueva==1)
   {
float pxf = convertToPosCar( 1 ); //Pos1 es Px
float pyf = convertToPosCar( 5 ); //Pos1 es Px
float pzf = convertToPosCar( 9 ); //Pos1 es Px
pxf = pxf/1000; //Pos1 es Px
pyf = pyf/1000; //Pos1 es Px
pzf = pzf/1000; //Pos1 es Px
pxi =0;
pyi =0;
pzi =.3;
data_nueva=2;
vdx = pxf-pxi;
vdy = pyf-pyi;
vdz = pzf-pzi;
Serial.print("pxf="); Serial.println(vdx,5);
```

```
Serial.print("pyf="); Serial.println(vdy,5);
Serial.print("pzf="); Serial.println(vdz,5);    }
if (data_nueva==2)
{
for (TT=0;TT<41;TT++)
{
Px = pxi + (MU[TT]*vdx);
Py = pyi + (MU[TT]*vdy);
Pz = pzi + (MU[TT]*vdz);
Vx = (MUD[TT]*vdx);
Vy = (MUD[TT]*vdy);
Vz = (MUD[TT]*vdz);

Vel[0][0]={{Vx}};
Vel[1][0]={{Vy}};
Vel[2][0]={{Vz}};


icinematica();
float theta1=-th11*(pi/180);
float theta2=-th21*(pi/180);
float theta3=-th31*(pi/180);
float M1 = Px-(cos(phi1)*(R1 + L1*cos(theta1)));
float M2 = Py-(sin(phi1)*(R1 + L1*cos(theta1)));
float M3 = Pz - (L1*sin(theta1));
float M4 = Px-(cos(phi2)*(R1 + L1*cos(theta2)));
float M5 = Py-(sin(phi2)*(R1 + L1*cos(theta2)));
float M6 = Pz - (L1*sin(theta2));
float M7 = Px-(cos(phi3)*(R1 + L1*cos(theta3)));
float M8 = Py-(sin(phi3)*(R1 + L1*cos(theta3)));
float M9 = Pz - (L1*sin(theta3));
```

```
    float s1T[1][3]= {M1,M2,M3};
    float s2T[1][3]= {M4,M5,M6};
    float s3T[1][3]= {M7,M8,M9};
    float b1[3][1]={{-L1*cos(phi1)*sin(theta1)},{-L1*sin(phi1)*sin(theta1)},
{L1*cos(theta1)}};
    float b2[3][1]={{-L1*cos(phi2)*sin(theta2)},{-L1*sin(phi2)*sin(theta2)},
{L1*cos(theta2)}};
    float b3[3][1]={{-L1*cos(phi3)*sin(theta3)},{-L1*sin(phi3)*sin(theta3)},{
L1*cos(theta3)}};
    float siTT[3][3]={{M1,M2,M3},
                      {M4,M5,M6},
                      {M7,M8,M9}};

    Matrix.Multiply((float*)s1T, (float*)b1, 1, 3, 1, (float*)k1);
    Matrix.Multiply((float*)s2T, (float*)b2, 1, 3, 1, (float*)k2);
    Matrix.Multiply((float*)s3T, (float*)b3, 1, 3, 1, (float*)k3);

    float K[3][3]={{k1[0][0], 0 ,0},{0 ,k2[0][0] ,0},{0, 0, k3[0][0]}};

    Matrix.Invert((float*)siTT, 3);
    Matrix.Multiply((float*)siTT, (float*)K, 3, 3, 3, (float*)Jacob);
    Matrix.Invert((float*)Jacob, 3);
    Matrix.Multiply((float*)Jacob, (float*)Vel, 3, 3, 1, (float*)Velq);

    Velq1[0][TT]=ceil(abs((Velq[0][0]*60/(2*pi))/0.229));
    Velq2[0][TT]=ceil(abs((Velq[1][0]*60/(2*pi))/0.229));
    Velq3[0][TT]=ceil(abs((Velq[2][0]*60/(2*pi))/0.229));

    QQ1[TT][0]=Q1;
    QQ2[TT][0]=Q2;
```

```
QQ3[TT][0]=Q3;


}
cartesiano=1;
data_nueva=3;
}

while (data_nueva==3)
{ recvWithStartEndMarkers();
tiempo = millis();
if ( tiempo > t_actualizado + t_delay)  //Se actualiza el tiempo que ha de transcurrir para el próximo delay.
{
  t_actualizado = tiempo;
  if(cartesiano==1){//for (TT=0;TT<81;TT++){                {


  Q3d=QQ3[TTT][0]/1000;
  Q3c=(QQ3[TTT][0]-(Q3d*1000))/100;
  Q3b=(QQ3[TTT][0]- (Q3d*1000 + Q3c*100))/10;
  Q3a=QQ3[TTT][0]-(Q3d*1000 + Q3c*100 + Q3b*10 );
  Q2d=QQ2[TTT][0]/1000;
  Q2c=(QQ2[TTT][0]-(Q2d*1000))/100;
  Q2b=(QQ2[TTT][0]- (Q2d*1000 + Q2c*100))/10;
  Q2a=QQ2[TTT][0]-(Q2d*1000 + Q2c*100 + Q2b*10 );
  Q1d=QQ1[TTT][0]/1000;
  Q1c=(QQ1[TTT][0]-(Q1d*1000))/100;
  Q1b=(QQ1[TTT][0]- (Q1d*1000 + Q1c*100))/10;
  Q1a=QQ1[TTT][0]-(Q1d*1000 + Q1c*100 + Q1b*10 );
  //Enviar velocidades validos
  V1c=Velq1[0][TTT]/100;
  V1b=(Velq1[0][TTT]-(V1c*100))/10;
```

```
  V1a=(Velq1[0][TTT]- (V1c*100 + V1b*10));
  V2c=Velq2[0][TTT]/100;
  V2b=(Velq2[0][TTT]-(V2c*100))/10;
  V2a=(Velq2[0][TTT]- (V2c*100 + V2b*10));
  V3c=Velq3[0][TTT]/100;
  V3b=(Velq3[0][TTT]-(V3c*100))/10;
  V3a=(Velq3[0][TTT]- (V3c*100 + V3b*10));
  if (TTT==40){digitalWrite(12,HIGH);
  cartesiano=2;
  Serial.println("SENTIDO CONTRARIO :V");
  TTT=39;}
  TTT=TTT+1;
}
if(cartesiano==2)
{
  Q3d=QQ3[TTT][0]/1000;
  Q3c=(QQ3[TTT][0]-(Q3d*1000))/100;
  Q3b=(QQ3[TTT][0]- (Q3d*1000 + Q3c*100))/10;
  Q3a=QQ3[TTT][0]-(Q3d*1000 + Q3c*100 + Q3b*10 );
  Q2d=QQ2[TTT][0]/1000;
  Q2c=(QQ2[TTT][0]-(Q2d*1000))/100;
  Q2b=(QQ2[TTT][0]- (Q2d*1000 + Q2c*100))/10;
  Q2a=QQ2[TTT][0]-(Q2d*1000 + Q2c*100 + Q2b*10 );
  Q1d=QQ1[TTT][0]/1000;
  Q1c=(QQ1[TTT][0]-(Q1d*1000))/100;
  Q1b=(QQ1[TTT][0]- (Q1d*1000 + Q1c*100))/10;
  Q1a=QQ1[TTT][0]-(Q1d*1000 + Q1c*100 + Q1b*10 );
  //Enviar velocidades validos
  V1c=Velq1[0][TTT]/100;
  V1b=(Velq1[0][TTT]-(V1c*100))/10;
  V1a=(Velq1[0][TTT]- (V1c*100 + V1b*10));
```

```
  V2c=Velq2[0][TTT]/100;
  V2b=(Velq2[0][TTT]-(V2c*100))/10;
  V2a=(Velq2[0][TTT]- (V2c*100 + V2b*10));
  V3c=Velq3[0][TTT]/100;
  V3b=(Velq3[0][TTT]-(V3c*100))/10;
  V3a=(Velq3[0][TTT]- (V3c*100 + V3b*10));
  if (TTT==0){cartesiano=3;
  Serial.println("Posicion Meta");TTT=1;}
  TTT=TTT-1;
}
if(cartesiano==3)
{
  int Q3f=map(QQ3f[0][TTT], -90, 0, 0, 1024);
  int Q2f=map(QQ2f[0][TTT], -90, 0, 0, 1024);
  int Q1f=map(QQ1f[0][TTT], -90, 0, 0, 1024);
  //Enviar angulos validos
  Q3d=Q3f/1000;
  Q3c=(Q3f-(Q3d*1000))/100;
  Q3b=(Q3f- (Q3d*1000 + Q3c*100))/10;
  Q3a=Q3f-(Q3d*1000 + Q3c*100 + Q3b*10 );
  Q2d=Q2f/1000;
  Q2c=(Q2f-(Q2d*1000))/100;
  Q2b=(Q2f- (Q2d*1000 + Q2c*100))/10;
  Q2a=Q2f-(Q2d*1000 + Q2c*100 + Q2b*10 );
  Q1d=Q1f/1000;
  Q1c=(Q1f-(Q1d*1000))/100;
  Q1b=(Q1f- (Q1d*1000 + Q1c*100))/10;
  Q1a=Q1f-(Q1d*1000 + Q1c*100 + Q1b*10 );
  //Enviar velocidades validos
  V1c=dqm1f[0][TTT]/100;
  V1b=(dqm1f[0][TTT]-(V1c*100))/10;
```

```
      V1a=(dqm1f[0][TTT]- (V1c*100 + V1b*10));
      V2c=dqm2f[0][TTT]/100;
      V2b=(dqm2f[0][TTT]-(V2c*100))/10;
      V2a=(dqm2f[0][TTT]- (V2c*100 + V2b*10));
      V3c=dqm3f[0][TTT]/100;
      V3b=(dqm3f[0][TTT]-(V3c*100))/10;
      V3a=(dqm3f[0][TTT]- (V3c*100 + V3b*10));
      if   (TTT==40){cartesiano=4;                    digitalWrite(12,LOW);     TTT=39;
Serial.println("REGRESO DE POSICION META");
      }TTT=TTT+1;
    }
    if(cartesiano==4)
    {
      int Q3f=map(QQ3f[0][TTT], -90, 0, 0, 1024);
      int Q2f=map(QQ2f[0][TTT], -90, 0, 0, 1024);
      int Q1f=map(QQ1f[0][TTT], -90, 0, 0, 1024);
      //Enviar angulos validos
      Q3d=Q3f/1000;
      Q3c=(Q3f-(Q3d*1000))/100;
      Q3b=(Q3f- (Q3d*1000 + Q3c*100))/10;
      Q3a=Q3f-(Q3d*1000 + Q3c*100 + Q3b*10 );
      Q2d=Q2f/1000;
      Q2c=(Q2f-(Q2d*1000))/100;
      Q2b=(Q2f- (Q2d*1000 + Q2c*100))/10;
      Q2a=Q2f-(Q2d*1000 + Q2c*100 + Q2b*10 );
      Q1d=Q1f/1000;
      Q1c=(Q1f-(Q1d*1000))/100;
      Q1b=(Q1f- (Q1d*1000 + Q1c*100))/10;
      Q1a=Q1f-(Q1d*1000 + Q1c*100 + Q1b*10 );
      //Enviar velocidades validos
      V1c=dqm1f[0][TTT]/100;
```

```
    V1b=(dqm1f[0][TTT]-(V1c*100))/10;
    V1a=(dqm1f[0][TTT]- (V1c*100 + V1b*10));
    V2c=dqm2f[0][TTT]/100;
    V2b=(dqm2f[0][TTT]-(V2c*100))/10;
    V2a=(dqm2f[0][TTT]- (V2c*100 + V2b*10));
    V3c=dqm3f[0][TTT]/100;
    V3b=(dqm3f[0][TTT]-(V3c*100))/10;
    V3a=(dqm3f[0][TTT]- (V3c*100 + V3b*10));
    if (TTT==0){cartesiano=5;
    data_nueva=4;TTT=1;}
    TTT=TTT-1;
  }
  //---------------ENVIAMOS DATOS A OPEN -------------------
  Serial2.print('<');Serial2.print("A");
  Serial2.print(Q1d);Serial2.print(Q1c);Serial2.print(Q1b);Serial2.print(Q1a);
  Serial2.print(Q2d);Serial2.print(Q2c);Serial2.print(Q2b);Serial2.print(Q2a);
  Serial2.print(Q3d);Serial2.print(Q3c);Serial2.print(Q3b);Serial2.print(Q3a);
  Serial2.print(V1c);Serial2.print(V1b);Serial2.print(V1a);
  Serial2.print(V2c);Serial2.print(V2b);Serial2.print(V2a);
  Serial2.print(V3c);Serial2.print(V3b);Serial2.print(V3a);
  Serial2.println('>');
  Serial.print("*");Serial.print(          Px);Serial.print("*");Serial.print(          Py
);Serial.print("*");Serial.print( Pz );Serial.println( receivedPos );
  }
  //---------------FIN MODO ARTICULAR-------------------
  }
  }
  }

  void recvWithStartEndMarkers()
  {
```

```
static boolean recvInProgress = false;
static boolean recvPosInProgress = false;

static byte ndx = 0;
char startMarker = '<';
char endMarker = '>';
char rc;

char startPosi = '<';
char endPos = '>';
static byte ndxx = 0;
char rcc;

if (Serial.available() > 0)
{
  rc = Serial.read();
  if (recvInProgress == true)
  {
    if (rc != endMarker)
    {
    receivedChars[ndx] = rc;
    ndx++;
    if (ndx >= numChars) { ndx = numChars - 1; }
    }
    else
    {
       receivedChars[ndx] = '\0'; // terminate the string
       recvInProgress = false;
       ndx = 0;
       newData = true;
     }
```

```
    }
    else if (rc == startMarker) { recvInProgress = true; }
  }
// Data recibida de Open
  if (Serial3.available() > 0)
  {
    rcc = Serial3.read();
    if (recvPosInProgress == true)
    {
      if (rcc != endPos)
      {
        receivedPos[ndxx] = rcc;
        ndxx++;
        if (ndxx >= posChars) { ndxx = posChars - 1; }
      }
      else
      {
        receivedPos[ndxx] = '\0'; // Terminar el string
        recvPosInProgress = false;
        ndxx = 0;
        newPos = true;
      }
    }
    else if (rcc == startPosi) {recvPosInProgress = true; }
  }
}
//---------------CONVERTIMOS NUMERO A ENTERO --------------------
int convertToNumber( byte startPos)
{
  unsigned int tmp = 0;
  tmp = (receivedChars[startPos]-48) * 1000;
```

```
  tmp = tmp + (receivedChars[startPos+1]-48) * 100;
  tmp = tmp + (receivedChars[startPos+2]-48) * 10;
  tmp = tmp + receivedChars[startPos+3]-48;
  return tmp;
}
//---------------CONVERTIMOS NUMERO A ENTERO -------------------
int convertToNumber2( byte startPos)
{
  unsigned int tmp = 0;
  tmp = tmp + (receivedChars[startPos]-48) * 100;
  tmp = tmp + (receivedChars[startPos+1]-48) * 10;
  tmp = tmp + receivedChars[startPos+2]-48;
  return tmp;
}
//---------------CONVERTIMOS NUMERO A ENTERO -------------------
int convertToPosCar( byte startPosCar)
{
  unsigned int tmp = 0;
  unsigned int signo = 0;
  signo = (receivedChars[startPosCar]);
  if (signo==45){
  tmp = tmp + (receivedChars[startPosCar+1]-48) * 100;
  tmp = tmp + (receivedChars[startPosCar+2]-48) * 10;
  tmp = tmp + receivedChars[startPosCar+3]-48;
  tmp = tmp*(-1);
  }
  else if (signo==43){
  tmp = tmp + (receivedChars[startPosCar+1]-48) * 100;
  tmp = tmp + (receivedChars[startPosCar+2]-48) * 10;
  tmp = tmp + receivedChars[startPosCar+3]-48; }
Serial.println(tmp);
```

```
  return tmp;
}

//---------------CONVERTIMOS NUMERO A ENTERO -------------------
int convertToPos( byte startP)
{
  unsigned int tmpos = 0;
  tmpos = (receivedPos[startP]-48) * 1000;
  tmpos = tmpos + (receivedPos[startP+1]-48) * 100;
  tmpos = tmpos + (receivedPos[startP+2]-48) * 10;
  tmpos = tmpos + receivedPos[startP+3]-48;
  return tmpos;
}
//---------------CONVERTIMOS NUMERO A ENTERO -------------------
int convertToButton( byte startPos)
{
  unsigned int suc = 0;
  suc = (receivedChars[startPos]-48);
  return suc;
}
//---------------FUNCION ICINEMATICA -------------------
void icinematica ()
{
  float Si1 = (1/L1)*(-(Px*Px)-(Py*Py)-(Pz*Pz)+(L2*L2)-(L1*L1)-(R1*R1));
  //Angulo theta1
    float Q11 = 2*Px*cos(phi1)+2*Py*sin(phi1);
    float D11 = 4*(Pz*Pz)+4*(R1*R1)-(Si1*Si1)+(Q11*Q11)*(1-
((R1*R1)/(L1*L1)))+Q11*(((-2*R1*Si1)/(L1))-4*R1);
    float E11 = -2*R1-Q11*((R1/L1)-1)-Si1;
    //Angulo theta2
    float Q21 = 2*Px*cos(phi2)+2*Py*sin(phi2);
```

```
float D21 = 4*(Pz*Pz)+4*(R1*R1)-(Si1*Si1)+(Q21*Q21)*(1-((R1*R1)/(L1*L1)))+Q21*(((-2*R1*Si1)/(L1))-4*R1);
float E21 = -2*R1-Q21*((R1/L1)-1)-Si1;
// Angulo theta3
float Q31 = 2*Px*cos(phi3)+2*Py*sin(phi3);
float D31 = 4*(Pz*Pz)+4*(R1*R1)-(Si1*Si1)+(Q31*Q31)*(1-((R1*R1)/(L1*L1)))+Q31*(((-2*R1*Si1)/(L1))-4*R1);
float E31 = -2*R1-Q31*((R1/L1)-1)-Si1;

 if(D11>=0 || D21>=0 || D31>=0)  //---------------DETERMINAR SI POSICION ES VALIDA ------------------
{
th11 = -2*atan(((-2*Pz)+sqrt(D11))/E11);
th21 = -2*atan(((-2*Pz)+sqrt(D21))/E21);
th31 = -2*atan(((-2*Pz)+sqrt(D31))/E31);
th11=th11*(180/pi);
th21=th21*(180/pi);
th31=th31*(180/pi);

 Q1 = map(th11, -90, 0, 0, 1024);
 Q2 = map(th21, -90, 0, 0, 1024);
 Q3 = map(th31, -90, 0, 0, 1024);
}
else{Serial.print(" POSICION INVALIDA");}

}
void sendOK(int val)
{
  Serial.print("OK");Serial.println(val);
}
```

CONTROLADOR ESCLAVO – OPENCM9.04

```
#include <DynamixelSDK.h>
#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>

// Tabla de Control Servos Dynamixel XM-430 W-350 T
#define ADDR_PRO_TORQUE_ENABLE          64
#define ADDR_PRO_GOAL_POSITION          116
#define ADDR_PRO_PRESENT_POSITION       132
#define ADDR_TEMPERATURA                146
#define ADDR_VOLTAJE                    144
#define ADDR_TJVELOCIDAD                136
#define ADDR_APROF                      108
#define ADDR_VPROF                      112
#define ADDR_ACELERACION                108

// Seleccionamos la version del protocolo
#define PROTOCOL_VERSION                2.0
#define LEN_PRO_GOAL_POSITION           8
#define SERVO1 1
#define SERVO2 2
```

```
#define SERVO3 3
// Seleccionamos la velocidad y puerto COM empleado
#define BAUDRATE                    1000000
#define DEVICENAME                   "1"

// Variables de Proceso
#define TORQUE_ENABLE                1
#define TORQUE_DISABLE               0
#define DXL_MOVING_STATUS_THRESHOLD     20
#define ACEL_OFF                     0
#define ACEL_ON                      20
#define ESC_ASCII_VALUE              0x1b
#define x

// Variables del robot
boolean debug = true;
int Q1;int Q2;int Q3;
int poshome; int velhome;
int posfin; int velfin;
int V1;int V2;int V3;
int S1;int S2;int S3;
int JJ;int AA=0;int CC;int FF;
unsigned long tiempo = 0;
unsigned long t_actualizado = 0;
unsigned long t_delay = 40;
unsigned long tiempofu = 0;
const byte numChars = 40;
const byte ptotal = 40;
char receivedChars[numChars];
char postotal[ptotal];
boolean newData = false;
```

```
bool dxl_addparam_result = false;

boolean feedback = true;
int yy; int y;
int zz; int z;

// Libreria eFLL
Fuzzy* fuzzy = new Fuzzy();

//Funciones de error
FuzzySet* NG = new FuzzySet (-15, -15, -4.9, -2.75);
FuzzySet* NP = new FuzzySet(-4.5, -2,-2, -0.5);
FuzzySet* CERO = new FuzzySet(-1.5, 0,0, 1.5);
FuzzySet* PP = new FuzzySet(0.5, 2,2, 4.5);
FuzzySet* PG = new FuzzySet(2.75, 4.9, 15, 15);

//Funciones de derivada de error
FuzzySet* DNG = new FuzzySet(-15, -15, -10, -5);
FuzzySet* DNP = new FuzzySet(-7, -4,-4, -1);
FuzzySet* DZ = new FuzzySet(-2.5, 0,0, 2.5);
FuzzySet* DPP = new FuzzySet(1, 4,4, 7);
FuzzySet* DPG = new FuzzySet(5, 10, 15, 15);

void setup()
{
  Serial.begin(115200);
  Serial3.begin(500000);
  Serial2.begin(57600);

  // Definimos error como entrada
  FuzzyInput* error = new FuzzyInput(1);
```

```
error->addFuzzySet(NG);
error->addFuzzySet(NP);
error->addFuzzySet(CERO);
error->addFuzzySet(PP);
error->addFuzzySet(PG);
fuzzy->addFuzzyInput(error);

// Definimos derivada de error como entrada
FuzzyInput* De = new FuzzyInput(2);
De->addFuzzySet(DNG);
De->addFuzzySet(DNP);
De->addFuzzySet(DZ);
De->addFuzzySet(DPP);
De->addFuzzySet(DPG);
fuzzy->addFuzzyInput(De);

// Definimos posicion como salida
FuzzyOutput* posicion = new FuzzyOutput(1);
FuzzySet* NGO = new FuzzySet(-10, -10,-10, -4);
posicion->addFuzzySet(NGO);
FuzzySet* NPO = new FuzzySet(-5, -2.8,-2.8, -0.8);
posicion->addFuzzySet(NPO);
FuzzySet* ZO = new FuzzySet(-1, 0,0, 1);
posicion->addFuzzySet(ZO);
FuzzySet* PPO = new FuzzySet(0.8, 2.8,2.8,5);
posicion->addFuzzySet(PPO);
FuzzySet* PGO = new FuzzySet(4, 10,10, 10);
posicion->addFuzzySet(PGO);
fuzzy->addFuzzyOutput(posicion);

// Reglas difusas
```

```
//REGLA1
FuzzyRuleAntecedent* errorNGAndDeDNG = new FuzzyRuleAntecedent();
errorNGAndDeDNG->joinWithAND(NG, DNG);
FuzzyRuleConsequent* thenPosicionNGO1 = new FuzzyRuleConsequent();
thenPosicionNGO1->addOutput(NGO);
FuzzyRule* fuzzyRule1 = new FuzzyRule(1, errorNGAndDeDNG, thenPosicionNGO1);
fuzzy->addFuzzyRule(fuzzyRule1);
//REGLA2
FuzzyRuleAntecedent* errorNGAndDeDNP = new FuzzyRuleAntecedent();
errorNGAndDeDNP->joinWithAND(NG, DNP);
FuzzyRuleConsequent* thenPosicionNGO2 = new FuzzyRuleConsequent();
thenPosicionNGO2->addOutput(NGO);
FuzzyRule* fuzzyRule2 = new FuzzyRule(2, errorNGAndDeDNP, thenPosicionNGO2);
fuzzy->addFuzzyRule(fuzzyRule2);
//REGLA3
FuzzyRuleAntecedent* errorNGAndDeDZ = new FuzzyRuleAntecedent();
errorNGAndDeDZ->joinWithAND(NG, DZ);
FuzzyRuleConsequent* thenPosicionNPO1 = new FuzzyRuleConsequent();
thenPosicionNPO1->addOutput(NPO);
FuzzyRule* fuzzyRule3 = new FuzzyRule(3, errorNGAndDeDZ, thenPosicionNPO1);
fuzzy->addFuzzyRule(fuzzyRule3);
//REGLA4
FuzzyRuleAntecedent* errorNGAndDeDPP = new FuzzyRuleAntecedent();
errorNGAndDeDPP->joinWithAND(NG, DPP);
FuzzyRuleConsequent* thenPosicionNPO2 = new FuzzyRuleConsequent();
thenPosicionNPO2->addOutput(NPO);
FuzzyRule* fuzzyRule4 = new FuzzyRule(4, errorNGAndDeDPP, thenPosicionNPO2);
```

```
fuzzy->addFuzzyRule(fuzzyRule4);
//REGLA5
FuzzyRuleAntecedent* errorNGAndDeDPG = new FuzzyRuleAntecedent();
errorNGAndDeDPG->joinWithAND(NG, DPG);
FuzzyRuleConsequent* thenPosicionZO1 = new FuzzyRuleConsequent();
thenPosicionZO1->addOutput(ZO);
FuzzyRule* fuzzyRule5 = new FuzzyRule(5, errorNGAndDeDPG, thenPosicionZO1);
fuzzy->addFuzzyRule(fuzzyRule5);
//REGLA6
FuzzyRuleAntecedent* errorNPAndDeDNG = new FuzzyRuleAntecedent();
errorNPAndDeDNG->joinWithAND(NP, DNG);
FuzzyRuleConsequent* thenPosicionNGO3 = new FuzzyRuleConsequent();
thenPosicionNGO3->addOutput(NGO);
FuzzyRule* fuzzyRule6 = new FuzzyRule(6, errorNPAndDeDNG, thenPosicionNGO3);
fuzzy->addFuzzyRule(fuzzyRule6);
//REGLA7
FuzzyRuleAntecedent* errorNPAndDeDNP = new FuzzyRuleAntecedent();
errorNPAndDeDNP->joinWithAND(NP, DNP);
FuzzyRuleConsequent* thenPosicionNPO3 = new FuzzyRuleConsequent();
thenPosicionNPO3->addOutput(NPO);
FuzzyRule* fuzzyRule7 = new FuzzyRule(7, errorNPAndDeDNP, thenPosicionNPO3);
fuzzy->addFuzzyRule(fuzzyRule7);
//REGLA8
FuzzyRuleAntecedent* errorNPAndDeDZ = new FuzzyRuleAntecedent();
errorNPAndDeDZ->joinWithAND(NP, DZ);
FuzzyRuleConsequent* thenPosicionNPO4 = new FuzzyRuleConsequent();
thenPosicionNPO4->addOutput(NPO);
```

```
FuzzyRule* fuzzyRule8 = new FuzzyRule(8, errorNPAndDeDZ, thenPosicionNPO4);
fuzzy->addFuzzyRule(fuzzyRule8);
//REGLA9
FuzzyRuleAntecedent* errorNPAndDeDPP = new FuzzyRuleAntecedent();
errorNPAndDeDPP->joinWithAND(NP, DPP);
FuzzyRuleConsequent* thenPosicionZO2 = new FuzzyRuleConsequent();
thenPosicionZO2->addOutput(ZO);
FuzzyRule* fuzzyRule9 = new FuzzyRule(9, errorNPAndDeDPP, thenPosicionZO2);
fuzzy->addFuzzyRule(fuzzyRule9);
//REGLA10
FuzzyRuleAntecedent* errorNPAndDeDPG = new FuzzyRuleAntecedent();
errorNPAndDeDPG->joinWithAND(NP, DPG);
FuzzyRuleConsequent* thenPosicionPPO1 = new FuzzyRuleConsequent();
thenPosicionPPO1->addOutput(PPO);
FuzzyRule* fuzzyRule10 = new FuzzyRule(10, errorNPAndDeDPG, thenPosicionPPO1);
fuzzy->addFuzzyRule(fuzzyRule10);
//REGLA11
FuzzyRuleAntecedent* errorCEROAndDeDNG = new FuzzyRuleAntecedent();
errorCEROAndDeDNG->joinWithAND(CERO, DNG);
FuzzyRuleConsequent* thenPosicionNPO5 = new FuzzyRuleConsequent();
thenPosicionNPO5->addOutput(NPO);
FuzzyRule* fuzzyRule11 = new FuzzyRule(11, errorCEROAndDeDNG, thenPosicionNPO5);
fuzzy->addFuzzyRule(fuzzyRule11);
//REGLA12
FuzzyRuleAntecedent* errorCEROAndDeDNP = new FuzzyRuleAntecedent();
errorCEROAndDeDNP->joinWithAND(CERO, DNP);
FuzzyRuleConsequent* thenPosicionNPO6 = new FuzzyRuleConsequent();
```

```
thenPosicionNPO6->addOutput(NPO);
FuzzyRule* fuzzyRule12 = new FuzzyRule(12, errorCEROAndDeDNP, thenPosicionNPO6);
fuzzy->addFuzzyRule(fuzzyRule12);
//REGLA13
FuzzyRuleAntecedent* errorCEROAndDeDZ = new FuzzyRuleAntecedent();
errorCEROAndDeDZ->joinWithAND(CERO, DZ);
FuzzyRuleConsequent* thenPosicionZO3 = new FuzzyRuleConsequent();
thenPosicionZO3->addOutput(ZO);
FuzzyRule* fuzzyRule13 = new FuzzyRule(13, errorCEROAndDeDZ, thenPosicionZO3);
fuzzy->addFuzzyRule(fuzzyRule13);
//REGLA14
FuzzyRuleAntecedent* errorCEROAndDeDPP = new FuzzyRuleAntecedent();
errorCEROAndDeDPP->joinWithAND(CERO, DPP);
FuzzyRuleConsequent* thenPosicionPPO2 = new FuzzyRuleConsequent();
thenPosicionPPO2->addOutput(PPO);
FuzzyRule* fuzzyRule14 = new FuzzyRule(14, errorCEROAndDeDPP, thenPosicionPPO2);
fuzzy->addFuzzyRule(fuzzyRule14);
//REGLA15
FuzzyRuleAntecedent* errorCEROAndDeDPG = new FuzzyRuleAntecedent();
errorCEROAndDeDPG->joinWithAND(CERO, DPG);
FuzzyRuleConsequent* thenPosicionPPO3 = new FuzzyRuleConsequent();
thenPosicionPPO3->addOutput(PPO);
FuzzyRule* fuzzyRule15 = new FuzzyRule(15, errorCEROAndDeDPG, thenPosicionPPO3);
fuzzy->addFuzzyRule(fuzzyRule15);
//REGLA16
FuzzyRuleAntecedent* errorPPAndDeDNG = new FuzzyRuleAntecedent();
errorPPAndDeDNG->joinWithAND(PP, DNG);
```

```
FuzzyRuleConsequent* thenPosicionNPO7 = new FuzzyRuleConsequent();
thenPosicionNPO7->addOutput(NPO);
FuzzyRule* fuzzyRule16 = new FuzzyRule(16, errorPPAndDeDNG, thenPosicionNPO7);
fuzzy->addFuzzyRule(fuzzyRule16);
//REGLA17
FuzzyRuleAntecedent* errorPPAndDeDNP = new FuzzyRuleAntecedent();
errorPPAndDeDNP->joinWithAND(PP, DNP);
FuzzyRuleConsequent* thenPosicionZO4 = new FuzzyRuleConsequent();
thenPosicionZO4->addOutput(ZO);
FuzzyRule* fuzzyRule17 = new FuzzyRule(17, errorPPAndDeDNP, thenPosicionZO4);
fuzzy->addFuzzyRule(fuzzyRule17);
//REGLA18
FuzzyRuleAntecedent* errorPPAndDeDZ = new FuzzyRuleAntecedent();
errorPPAndDeDZ->joinWithAND(PP, DZ);
FuzzyRuleConsequent* thenPosicionPPO4 = new FuzzyRuleConsequent();
thenPosicionPPO4->addOutput(PPO);
FuzzyRule* fuzzyRule18 = new FuzzyRule(18, errorPPAndDeDZ, thenPosicionPPO4);
fuzzy->addFuzzyRule(fuzzyRule18);
//REGLA19
FuzzyRuleAntecedent* errorPPAndDeDPP = new FuzzyRuleAntecedent();
errorPPAndDeDPP->joinWithAND(PP, DPP);
FuzzyRuleConsequent* thenPosicionPPO5 = new FuzzyRuleConsequent();
thenPosicionPPO5->addOutput(PPO);
FuzzyRule* fuzzyRule19 = new FuzzyRule(19, errorPPAndDeDPP, thenPosicionPPO5);
fuzzy->addFuzzyRule(fuzzyRule19);
//REGLA20
FuzzyRuleAntecedent* errorPPAndDeDPG = new FuzzyRuleAntecedent();
```

```
errorPPAndDeDPG->joinWithAND(PP, DPG);
FuzzyRuleConsequent* thenPosicionPGO1 = new FuzzyRuleConsequent();
thenPosicionPGO1->addOutput(PGO);
FuzzyRule* fuzzyRule20 = new FuzzyRule(20, errorPPAndDeDPG, thenPosicionPGO1);
fuzzy->addFuzzyRule(fuzzyRule20);
//REGLA21
FuzzyRuleAntecedent* errorPGAndDeDNG = new FuzzyRuleAntecedent();
errorPGAndDeDNG->joinWithAND(PG, DNG);
FuzzyRuleConsequent* thenPosicionZO5 = new FuzzyRuleConsequent();
thenPosicionZO5->addOutput(ZO);
FuzzyRule* fuzzyRule21 = new FuzzyRule(21, errorPGAndDeDNG, thenPosicionZO5);
fuzzy->addFuzzyRule(fuzzyRule21);
//REGLA22
FuzzyRuleAntecedent* errorPGAndDeDNP = new FuzzyRuleAntecedent();
errorPGAndDeDNP->joinWithAND(PG, DNP);
FuzzyRuleConsequent* thenPosicionPPO6 = new FuzzyRuleConsequent();
thenPosicionPPO6->addOutput(PPO);
FuzzyRule* fuzzyRule22 = new FuzzyRule(22, errorPGAndDeDNP, thenPosicionPPO6);
fuzzy->addFuzzyRule(fuzzyRule22);
//REGLA23
FuzzyRuleAntecedent* errorPGAndDeDZ = new FuzzyRuleAntecedent();
errorPGAndDeDZ->joinWithAND(PG, DZ);
FuzzyRuleConsequent* thenPosicionPPO7 = new FuzzyRuleConsequent();
thenPosicionPPO7->addOutput(PGO);
FuzzyRule* fuzzyRule23 = new FuzzyRule(23, errorPGAndDeDZ, thenPosicionPPO7);
fuzzy->addFuzzyRule(fuzzyRule23);
//REGLA24
```

```
    FuzzyRuleAntecedent* errorPGAndDeDPP = new FuzzyRuleAntecedent();
    errorPGAndDeDPP->joinWithAND(PG, DPP);
    FuzzyRuleConsequent* thenPosicionPGO2 = new FuzzyRuleConsequent();
    thenPosicionPGO2->addOutput(PGO);
    FuzzyRule* fuzzyRule24 = new FuzzyRule(24, errorPGAndDeDPP, thenPosicionPGO2);
    fuzzy->addFuzzyRule(fuzzyRule24);
    //REGLA25
    FuzzyRuleAntecedent* errorPGAndDeDPG = new FuzzyRuleAntecedent();
    errorPGAndDeDPG->joinWithAND(PG, DPG);
    FuzzyRuleConsequent* thenPosicionPGO3 = new FuzzyRuleConsequent();
    thenPosicionPGO3->addOutput(PGO);
    FuzzyRule* fuzzyRule25 = new FuzzyRule(25, errorPGAndDeDPG, thenPosicionPGO3);
    fuzzy->addFuzzyRule(fuzzyRule25);
    //-------------------------------------INICIO----------------------------//
  }
  void loop()
  {
    DynamixelM();  //Nos envia a la funcion DynamixelM()
  }
  void recvWithStartEndMarkers()
  {
    static boolean recvInProgress = false;
    static byte ndx = 0;
    char startMarker = '<';
    char endMarker = '>';
    char rc;
    if (Serial3.available() > 0)
    {
      rc = Serial3.read();
```

```
    if (recvInProgress == true)
    {
      if (rc != endMarker)
      {
        receivedChars[ndx] = rc;
        ndx++;
        if (ndx >= numChars)
        {ndx = numChars - 1;}
       }
       else
       {
        receivedChars[ndx] = '\0'; // Terminamos el string
        recvInProgress = false;
        ndx = 0;
        newData = true;
        }
     }
     else if (rc == startMarker) { recvInProgress = true; }
  }
}

//Convertimos a entero
int convertToNumber( byte startPos)
{
  unsigned int tmp = 0;
  tmp = (receivedChars[startPos]-48) * 1000;
  tmp = tmp + (receivedChars[startPos+1]-48) * 100;
  tmp = tmp + (receivedChars[startPos+2]-48) * 10;
  tmp = tmp + receivedChars[startPos+3]-48;
  return tmp;
}
```

```
//Convertimos a entero
int convertToNumber2( byte startPos2)
{
  unsigned int cmp = 0;
  cmp = (receivedChars[startPos2]-48) * 100;
  cmp = cmp + (receivedChars[startPos2+1]-48) * 10;
  cmp = cmp + (receivedChars[startPos2+2]-48);
  return cmp;
}

void DynamixelM()
{
  // Inicializamos los servomotores
  dynamixel::PortHandler *portHandler = dynamixel::PortHandler::getPortHandler(DEVICENAME);
  dynamixel::PacketHandler *packetHandler = dynamixel::PacketHandler::getPacketHandler(PROTOCOL_VERSION);
  dynamixel::GroupSyncWrite groupSyncWrite(portHandler, packetHandler, ADDR_VPROF, LEN_PRO_GOAL_POSITION);

  int index = 0;
  int dxl_comm_result = COMM_TX_FAIL;
  int dxl_goal_position[2] = {DXL_MINIMUM_POSITION_VALUE, DXL_MAXIMUM_POSITION_VALUE};
  uint8_t param_goal_position_1[8];
  uint8_t param_goal_position_2[8];
  uint8_t param_goal_position_3[8];

  uint8_t param_pos_vel_1[8];
  uint8_t param_pos_vel_2[8];
```

```
uint8_t param_pos_vel_3[8];

uint8_t dxl_error = 0;
int32_t pos1 = 0;
int8_t tem1=0  ;
int16_t nVolt1=0;
int32_t velocidad1=0;
int16_t carga1=0;

int32_t pos2 = 0;
int8_t tem2=0  ;
int16_t nVolt2=0;
int32_t velocidad2=0;
int16_t carga2=0;

int32_t pos3 = 0;
int8_t tem3=0  ;
int16_t nVolt3=0;
int32_t velocidad3=0;
int16_t carga3=0;

int errora1;
int errora2;
int errora3;
// Abrimos puertos
if (portHandler->openPort())
{
  Serial.print("Exito para abrir el puerto!\n");
}
else
{
```

```
    Serial.print("Error al abrir el puerto!\n");
    return;
  }
  // Establecer velocidad del puerto
  if (portHandler->setBaudRate(BAUDRATE))
  {
    Serial.print("Tuvo éxito para cambiar velocidad del puerto!\n");
  }
  else
  {
    Serial.print("Error al cambiar velocidad del puerto!\n");
    return;
  }
 //----------------------------------------HABILITAR TORQUES PARA CADA MOTOR----------------------------------------//
  // Habilitar Torque Dynamixel 1
  dxl_comm_result = packetHandler->write1ByteTxRx(portHandler, SERVO1, ADDR_PRO_TORQUE_ENABLE, TORQUE_ENABLE, &dxl_error);
  if (dxl_comm_result != COMM_SUCCESS)
  {
   packetHandler->printTxRxResult(dxl_comm_result);
  }
  else if (dxl_error != 0)
  {
    packetHandler->printRxPacketError(dxl_error);
  }
  else
  {
    Serial.print("Dynamixel se ha conectado con éxito \n");
  }
 // Habilitar Torque Dynamixel 2
```

```
    dxl_comm_result = packetHandler->write1ByteTxRx(portHandler, SERVO2,
ADDR_PRO_TORQUE_ENABLE, TORQUE_ENABLE, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
     packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    else
    {
      Serial.print("Dynamixel se ha conectado con éxito \n");
    }

   // Habilitar Torque Dynamixel 3
    dxl_comm_result = packetHandler->write1ByteTxRx(portHandler, SERVO3,
ADDR_PRO_TORQUE_ENABLE, TORQUE_ENABLE, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
     packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    else
    {
      Serial.print("Dynamixel se ha conectado con éxito \n");
    }
```

```
//------------------------------FIN HABILITAR TORQUES PARA CADA MOTOR-------
---------------------------------------//

//------------------------------HABILITAR ACELERACION PARA CADA MOTOR------
-----------------------------------------//
// Habilitar  Aceleración Dynamixel 1
 dxl_comm_result  =  packetHandler->write4ByteTxRx(portHandler,  SERVO1,
116, 1103, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
 {
  packetHandler->printTxRxResult(dxl_comm_result);
 }
 else if (dxl_error != 0)
{
   packetHandler->printRxPacketError(dxl_error);
 }

 //  Habilitar  Aceleración Dynamixel 2
 dxl_comm_result  =  packetHandler->write4ByteTxRx(portHandler,  SERVO2,
116, 1103, &dxl_error);
 if (dxl_comm_result != COMM_SUCCESS)
 {
  packetHandler->printTxRxResult(dxl_comm_result);
 }
 else if (dxl_error != 0)
{
  packetHandler->printRxPacketError(dxl_error);
 }
 // Habilitar  Aceleración Dynamixel 3
dxl_comm_result  =  packetHandler->write4ByteTxRx(portHandler,  SERVO3,
116, 1103, &dxl_error);
```

```
 if (dxl_comm_result != COMM_SUCCESS)
 {
  packetHandler->printTxRxResult(dxl_comm_result);
 }
  else if (dxl_error != 0)
 {
  packetHandler->printRxPacketError(dxl_error);
  }
 //------------------------------FIN ACELERACION PARA CADA MOTOR-----------------
----------------------------//
 while(1)
 {
  if (Serial3.available() > 0)    {  recvWithStartEndMarkers();}
  if (newData) {
    newData = false;
 //---------------------------------------LEER POSICION-------------------------------------
-----------//
    // Leer posición actual SERVO 1
    dxl_comm_result  =  packetHandler->read4ByteTxRx(portHandler,  SERVO1,
ADDR_PRO_PRESENT_POSITION, (uint32_t*)&pos1, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Leer posición actual SERVO 2
    dxl_comm_result  =  packetHandler->read4ByteTxRx(portHandler,  SERVO2,
ADDR_PRO_PRESENT_POSITION, (uint32_t*)&pos2, &dxl_error);
```

```
        if (dxl_comm_result != COMM_SUCCESS)
        {
          packetHandler->printTxRxResult(dxl_comm_result);
        }
          else if (dxl_error != 0)
        {
          packetHandler->printRxPacketError(dxl_error);
        }
        // Leer posición actual SERVO 3
        dxl_comm_result  =  packetHandler->read4ByteTxRx(portHandler,  SERVO3,
ADDR_PRO_PRESENT_POSITION, (uint32_t*)&pos3, &dxl_error);
        if (dxl_comm_result != COMM_SUCCESS)
        {
          packetHandler->printTxRxResult(dxl_comm_result);
        }
          else if (dxl_error != 0)
        {
          packetHandler->printRxPacketError(dxl_error);
        }
    //-------------------------------------FIN LEER POSICION---------------------------------
-------------//

    //-----------------------------------INICIO PRE MODO JOYSTICK-------------------------
-------------------//
    if  ( receivedChars[0] == 'P' )
    {
      JJ=1;AA=0;CC=0;FF=0;
      Q1 = convertToNumber( 1 );
      Q2 = convertToNumber( 5 );
      Q3 = convertToNumber( 9 );
      V1 = convertToNumber2( 13 );
```

```
    V2 = convertToNumber2( 16 );
    V3 = convertToNumber2( 19 );
    // Habilitar Dynamixel Aceleración 1
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO1,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    //  Habilitar Dynamixel Aceleración 2
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO2,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Habilitar Dynamixel Aceleración 3
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO3,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
```

```
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
  }
  //-------------------------------------FIN PRE MODO JOYSTICK-----------------------------
-----------------//
  //-------------------------------------INICIO MODO JOYSTICK-----------------------------
----------------//
  if (JJ==1)
  {
    int  error1 = Q1-pos1;
    int  error2 = Q2-pos2;
    int  error3 = Q3-pos3;
    int  de1 = error1 - errora1;
    int  de2 = error2 - errora2;
    int  de3 = error3 - errora3;
    errora1 = error1;
    errora2 = error2;
    errora3 = error3;

    fuzzy->setInput(1, error1);
    fuzzy->setInput(2, de1);
    fuzzy->fuzzify();
    float o1 = fuzzy->defuzzify(1);
    int q1 = round(Q1+o1);

    fuzzy->setInput(1, error2);
    fuzzy->setInput(2, de2);
    fuzzy->fuzzify();
    float o2 = fuzzy->defuzzify(1);
```

```
int q2 = round(Q2+o2);

fuzzy->setInput(1, error3);
fuzzy->setInput(2, de3);
fuzzy->fuzzify();
float o3 = fuzzy->defuzzify(1);
int q3 = round(Q3+o3);

// Asignar valores en matriz de bytes
param_goal_position_1[0] = DXL_LOBYTE(DXL_LOWORD(V1));
param_goal_position_1[1] = DXL_HIBYTE(DXL_LOWORD(V1));
param_goal_position_1[2] = DXL_LOBYTE(DXL_HIWORD(V1));
param_goal_position_1[3] = DXL_HIBYTE(DXL_HIWORD(V1));
param_goal_position_1[4] = DXL_LOBYTE(DXL_LOWORD(q1));
param_goal_position_1[5] = DXL_HIBYTE(DXL_LOWORD(q1));
param_goal_position_1[6] = DXL_LOBYTE(DXL_HIWORD(q1));
param_goal_position_1[7] = DXL_HIBYTE(DXL_HIWORD(q1));

param_goal_position_2[0] = DXL_LOBYTE(DXL_LOWORD(V2));
param_goal_position_2[1] = DXL_HIBYTE(DXL_LOWORD(V2));
param_goal_position_2[2] = DXL_LOBYTE(DXL_HIWORD(V2));
param_goal_position_2[3] = DXL_HIBYTE(DXL_HIWORD(V2));
param_goal_position_2[4] = DXL_LOBYTE(DXL_LOWORD(q2));
param_goal_position_2[5] = DXL_HIBYTE(DXL_LOWORD(q2));
param_goal_position_2[6] = DXL_LOBYTE(DXL_HIWORD(q2));
param_goal_position_2[7] = DXL_HIBYTE(DXL_HIWORD(q2));

param_goal_position_3[0] = DXL_LOBYTE(DXL_LOWORD(V3));
param_goal_position_3[1] = DXL_HIBYTE(DXL_LOWORD(V3));
param_goal_position_3[2] = DXL_LOBYTE(DXL_HIWORD(V3));
param_goal_position_3[3] = DXL_HIBYTE(DXL_HIWORD(V3));
```

```
      param_goal_position_3[4] = DXL_LOBYTE(DXL_LOWORD(q3));
      param_goal_position_3[5] = DXL_HIBYTE(DXL_LOWORD(q3));
      param_goal_position_3[6] = DXL_LOBYTE(DXL_HIWORD(q3));
      param_goal_position_3[7] = DXL_HIBYTE(DXL_HIWORD(q3));

      // Asignar el valor a Dynamixel 1 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO1,
param_goal_position_1);
      // Asignar el valor a Dynamixel 2 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO2,
param_goal_position_2);
      // Asignar el valor a Dynamixel 3 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO3,
param_goal_position_3);

      //IMPRIMIR POSICIONES AL MOTOR DYNAMIXEL XM-430 W350 T //
      dxl_comm_result = groupSyncWrite.txPacket();
      if (dxl_comm_result != COMM_SUCCESS) packetHandler-
>printTxRxResult(dxl_comm_result);
      // Borrar el almacenamiento del parámetro syncwrite
      groupSyncWrite.clearParam();
    }
    //-------------------------------------FIN MODO JOYSTICK-----------------------------
--------------//
    //-------------------------------------INICIO PRE MODO ARTICULAR---------------------
--------------------//
    if  ( receivedChars[0] == 'A' )
    {
      JJ=0;CC=0;FF=0;AA=1;
      Q1 = convertToNumber( 1 );
      Q2 = convertToNumber( 5 );
```

```
Q3 = convertToNumber( 9 );
S1 = convertToNumber2( 13);
S2 = convertToNumber2( 16);
S3 = convertToNumber2( 19);
// Habilitar Dynamixel Aceleración 1
dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO1, ADDR_ACELERACION, ACEL_OFF, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
  packetHandler->printTxRxResult(dxl_comm_result);
}
else if (dxl_error != 0)
{
  packetHandler->printRxPacketError(dxl_error);
}
//  Habilitar Dynamixel Aceleración 2
dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO2, ADDR_ACELERACION, ACEL_OFF, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
  packetHandler->printTxRxResult(dxl_comm_result);
}
else if (dxl_error != 0)
{
  packetHandler->printRxPacketError(dxl_error);
}
// Habilitar Dynamixel Aceleración 3
dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO3, ADDR_ACELERACION, ACEL_OFF, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
```

```
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
  }
  //-------------------------------------FIN PRE MODO ARTICULAR----------------------
----------------------//
  //-------------------------------------INICIO  MODO ARTICULAR--------------------------
--------------------//
  if (AA==1)
  {
    int  error1 = Q1-pos1;
    int    error2 = Q2-pos2;
    int error3 = Q3-pos3;
    int  de1 = error1 - errora1;
    int  de2 = error2 - errora2;
    int  de3 = error3 - errora3;
    errora1 = error1;
    errora2 = error2;
    errora3 = error3;
    fuzzy->setInput(1, error1);
    fuzzy->setInput(2, de1);
    fuzzy->fuzzify();
    float o1 = fuzzy->defuzzify(1);
    int q1 = round(Q1+o1);
    fuzzy->setInput(1, error2);
    fuzzy->setInput(2, de2);
    fuzzy->fuzzify();
    float o2 = fuzzy->defuzzify(1);
```

```
int q2 = round(Q2+o2);
fuzzy->setInput(1, error3);
fuzzy->setInput(2, de3);
fuzzy->fuzzify();
float o3 = fuzzy->defuzzify(1);
int q3 = round(Q3+o3);
// Vector de Velocidades y Posiciones Angulares - Motor 01
param_pos_vel_1[0] = DXL_LOBYTE(DXL_LOWORD(S1));
param_pos_vel_1[1] = DXL_HIBYTE(DXL_LOWORD(S1));
param_pos_vel_1[2] = DXL_LOBYTE(DXL_HIWORD(S1));
param_pos_vel_1[3] = DXL_HIBYTE(DXL_HIWORD(S1));
param_pos_vel_1[4] = DXL_LOBYTE(DXL_LOWORD(q1));
param_pos_vel_1[5] = DXL_HIBYTE(DXL_LOWORD(q1));
param_pos_vel_1[6] = DXL_LOBYTE(DXL_HIWORD(q1));
param_pos_vel_1[7] = DXL_HIBYTE(DXL_HIWORD(q1));

// Vector de Velocidades y Posiciones Angulares - Motor 02
param_pos_vel_2[0] = DXL_LOBYTE(DXL_LOWORD(S2));
param_pos_vel_2[1] = DXL_HIBYTE(DXL_LOWORD(S2));
param_pos_vel_2[2] = DXL_LOBYTE(DXL_HIWORD(S2));
param_pos_vel_2[3] = DXL_HIBYTE(DXL_HIWORD(S2));
param_pos_vel_2[4] = DXL_LOBYTE(DXL_LOWORD(q2));
param_pos_vel_2[5] = DXL_HIBYTE(DXL_LOWORD(q2));
param_pos_vel_2[6] = DXL_LOBYTE(DXL_HIWORD(q2));
param_pos_vel_2[7] = DXL_HIBYTE(DXL_HIWORD(q2));
// Vector de Velocidades y Posiciones Angulares - Motor 03
param_pos_vel_3[0] = DXL_LOBYTE(DXL_LOWORD(S3));
param_pos_vel_3[1] = DXL_HIBYTE(DXL_LOWORD(S3));
param_pos_vel_3[2] = DXL_LOBYTE(DXL_HIWORD(S3));
param_pos_vel_3[3] = DXL_HIBYTE(DXL_HIWORD(S3));
param_pos_vel_3[4] = DXL_LOBYTE(DXL_LOWORD(q3));
```

```
      param_pos_vel_3[5] = DXL_HIBYTE(DXL_LOWORD(q3));
      param_pos_vel_3[6] = DXL_LOBYTE(DXL_HIWORD(q3));
      param_pos_vel_3[7] = DXL_HIBYTE(DXL_HIWORD(q3));

      // Asignar el valor a Dynamixel 1 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO1,
param_pos_vel_1);
      // Asignar el valor a Dynamixel 2 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO2,
param_pos_vel_2);
      // Asignar el valor a Dynamixel 3 al almacenamiento Syncwrite
      dxl_addparam_result = groupSyncWrite.addParam(SERVO3,
param_pos_vel_3);

      //IMPRIMIR POSICIONES AL MOTOR DYNAMIXEL XM-430 W350 T //
      dxl_comm_result = groupSyncWrite.txPacket();
      if (dxl_comm_result != COMM_SUCCESS) packetHandler-
>printTxRxResult(dxl_comm_result);

      // Borrar el almacenamiento del parámetro syncwrite
      groupSyncWrite.clearParam();
    }
    //-------------------------------------FIN  MODO ARTICULAR-----------------------------
---------------//
    //-------------------------------------INICIO PRE MODO CASA----------------------------
----------------//
    if  ( receivedChars[0] == 'C' )
    {
     AA=0;JJ=0;FF=0;CC=1;
     poshome = 1110;
     velhome = 20;
```

```
    // Habilitar Dynamixel Aceleración 1
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO1,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    //  Habilitar Dynamixel Aceleración 2
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO2,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Habilitar Dynamixel Aceleración 3
    dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO3,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
```

```
      packetHandler->printRxPacketError(dxl_error);
    }
   }
   //--------------------------------------FIN PRE MODO CASA-------------------------------
----------//
   //--------------------------------------INICIO MODO CASA--------------------------------
---------//
   if (CC==1)
   {
     int error1 = poshome-pos1;
     int error2 = poshome-pos2;
     int error3 = poshome-pos3;
     int  de1 = error1 - errora1;
     int  de2 = error2 - errora2;
     int  de3 = error3 - errora3;
     errora1 = error1;
     errora2 = error2;
     errora3 = error3;

     fuzzy->setInput(1, error1);
     fuzzy->setInput(2, de1);
     fuzzy->fuzzify();
     float o1 = fuzzy->defuzzify(1);
     int q1 = round(poshome+o1);

     fuzzy->setInput(1, error2);
     fuzzy->setInput(2, de2);
     fuzzy->fuzzify();
     float o2 = fuzzy->defuzzify(1);
     int q2 = round(poshome+o2);
```

```
fuzzy->setInput(1, error3);
fuzzy->setInput(2, de3);
fuzzy->fuzzify();
float o3 = fuzzy->defuzzify(1);
int q3 = round(poshome+o3);

// Vector de Velocidades y Posiciones Angulares - Motor 01
param_pos_vel_1[0] = DXL_LOBYTE(DXL_LOWORD(velhome));
param_pos_vel_1[1] = DXL_HIBYTE(DXL_LOWORD(velhome));
param_pos_vel_1[2] = DXL_LOBYTE(DXL_HIWORD(velhome));
param_pos_vel_1[3] = DXL_HIBYTE(DXL_HIWORD(velhome));
param_pos_vel_1[4] = DXL_LOBYTE(DXL_LOWORD(q1));
param_pos_vel_1[5] = DXL_HIBYTE(DXL_LOWORD(q1));
param_pos_vel_1[6] = DXL_LOBYTE(DXL_HIWORD(q1));
param_pos_vel_1[7] = DXL_HIBYTE(DXL_HIWORD(q1));

// Vector de Velocidades y Posiciones Angulares - Motor 02
param_pos_vel_2[0] = DXL_LOBYTE(DXL_LOWORD(velhome));
param_pos_vel_2[1] = DXL_HIBYTE(DXL_LOWORD(velhome));
param_pos_vel_2[2] = DXL_LOBYTE(DXL_HIWORD(velhome));
param_pos_vel_2[3] = DXL_HIBYTE(DXL_HIWORD(velhome));
param_pos_vel_2[4] = DXL_LOBYTE(DXL_LOWORD(q2));
param_pos_vel_2[5] = DXL_HIBYTE(DXL_LOWORD(q2));
param_pos_vel_2[6] = DXL_LOBYTE(DXL_HIWORD(q2));
param_pos_vel_2[7] = DXL_HIBYTE(DXL_HIWORD(q2));

// Vector de Velocidades y Posiciones Angulares - Motor 03
param_pos_vel_3[0] = DXL_LOBYTE(DXL_LOWORD(velhome));
param_pos_vel_3[1] = DXL_HIBYTE(DXL_LOWORD(velhome));
param_pos_vel_3[2] = DXL_LOBYTE(DXL_HIWORD(velhome));
param_pos_vel_3[3] = DXL_HIBYTE(DXL_HIWORD(velhome));
```

```
    param_pos_vel_3[4] = DXL_LOBYTE(DXL_LOWORD(q3));
    param_pos_vel_3[5] = DXL_HIBYTE(DXL_LOWORD(q3));
    param_pos_vel_3[6] = DXL_LOBYTE(DXL_HIWORD(q3));
    param_pos_vel_3[7] = DXL_HIBYTE(DXL_HIWORD(q3));


    // Add Dynamixel#1 goal position value to the Syncwrite storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO1,
param_pos_vel_1);

    // Add Dynamixel#2 goal position value to the Syncwrite parameter storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO2,
param_pos_vel_2);

    // Add Dynamixel#3 goal position value to the Syncwrite parameter storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO3,
param_pos_vel_3);

    //IMPRIMIR POSICIONES AL MOTOR DYNAMIXEL XM-430 W350 T //
    dxl_comm_result = groupSyncWrite.txPacket();
    if (dxl_comm_result != COMM_SUCCESS) packetHandler-
>printTxRxResult(dxl_comm_result);

    // Clear syncwrite parameter storage
    groupSyncWrite.clearParam();
  }
  //-------------------------------------FIN  MODO CASA-------------------------------------
---------//
  //-----------------------------------INICIO PRE MODO CASA-----------------------------
-------------//
  if  ( receivedChars[0] == 'F' )
```

```
    {
      AA=0;JJ=0;CC=0;FF=1;
      posfin = 1800;
      velfin = 20;
      // Enable Dynamixel Aceleración 1
      dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO1,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
      if (dxl_comm_result != COMM_SUCCESS)
      {
        packetHandler->printTxRxResult(dxl_comm_result);
      }
      else if (dxl_error != 0)
      {
        packetHandler->printRxPacketError(dxl_error);
      }
      //  Enable Dynamixel Aceleración 2
      dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO2,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
      if (dxl_comm_result != COMM_SUCCESS)
      {
        packetHandler->printTxRxResult(dxl_comm_result);
      }
      else if (dxl_error != 0)
      {
        packetHandler->printRxPacketError(dxl_error);
      }
      // Enable Dynamixel Aceleración 3
      dxl_comm_result = packetHandler->write4ByteTxRx(portHandler, SERVO3,
ADDR_ACELERACION, ACEL_ON, &dxl_error);
      if (dxl_comm_result != COMM_SUCCESS)
      {
```

```
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
  }
  //---------------------------------------FIN PRE MODO FIN-----------------------------------
-------//
  //---------------------------------------INICIO MODO FIN-----------------------------------
------//
  if (FF==1)
  {
    Serial.print("POSICION FINAL = ");Serial.print(posfin);Serial.println(" ");
    Serial.print("Velocidad Final = ");Serial.print(velfin);Serial.println(" ");
    // Vector de Velocidades y Posiciones Angulares - Motor 01
    param_pos_vel_1[0] = DXL_LOBYTE(DXL_LOWORD(velfin));
    param_pos_vel_1[1] = DXL_HIBYTE(DXL_LOWORD(velfin));
    param_pos_vel_1[2] = DXL_LOBYTE(DXL_HIWORD(velfin));
    param_pos_vel_1[3] = DXL_HIBYTE(DXL_HIWORD(velfin));
    param_pos_vel_1[4] = DXL_LOBYTE(DXL_LOWORD(posfin));
    param_pos_vel_1[5] = DXL_HIBYTE(DXL_LOWORD(posfin));
    param_pos_vel_1[6] = DXL_LOBYTE(DXL_HIWORD(posfin));
    param_pos_vel_1[7] = DXL_HIBYTE(DXL_HIWORD(posfin));
    // Vector de Velocidades y Posiciones Angulares - Motor 02
    param_pos_vel_2[0] = DXL_LOBYTE(DXL_LOWORD(velfin));
    param_pos_vel_2[1] = DXL_HIBYTE(DXL_LOWORD(velfin));
    param_pos_vel_2[2] = DXL_LOBYTE(DXL_HIWORD(velfin));
    param_pos_vel_2[3] = DXL_HIBYTE(DXL_HIWORD(velfin));
    param_pos_vel_2[4] = DXL_LOBYTE(DXL_LOWORD(posfin));
    param_pos_vel_2[5] = DXL_HIBYTE(DXL_LOWORD(posfin));
```

```
    param_pos_vel_2[6] = DXL_LOBYTE(DXL_HIWORD(posfin));
    param_pos_vel_2[7] = DXL_HIBYTE(DXL_HIWORD(posfin));
    // Vector de Velocidades y Posiciones Angulares - Motor 03
    param_pos_vel_3[0] = DXL_LOBYTE(DXL_LOWORD(velfin));
    param_pos_vel_3[1] = DXL_HIBYTE(DXL_LOWORD(velfin));
    param_pos_vel_3[2] = DXL_LOBYTE(DXL_HIWORD(velfin));
    param_pos_vel_3[3] = DXL_HIBYTE(DXL_HIWORD(velfin));
    param_pos_vel_3[4] = DXL_LOBYTE(DXL_LOWORD(posfin));
    param_pos_vel_3[5] = DXL_HIBYTE(DXL_LOWORD(posfin));
    param_pos_vel_3[6] = DXL_LOBYTE(DXL_HIWORD(posfin));
    param_pos_vel_3[7] = DXL_HIBYTE(DXL_HIWORD(posfin));
    // Add Dynamixel#1 goal position value to the Syncwrite storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO1,
param_pos_vel_1);
    // Add Dynamixel#2 goal position value to the Syncwrite parameter storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO2,
param_pos_vel_2);
    // Add Dynamixel#3 goal position value to the Syncwrite parameter storage
    dxl_addparam_result = groupSyncWrite.addParam(SERVO3,
param_pos_vel_3);
    //IMPRIMIR POSICIONES AL MOTOR DYNAMIXEL XM-430 W350 T //
    dxl_comm_result = groupSyncWrite.txPacket();
    if (dxl_comm_result != COMM_SUCCESS) packetHandler-
>printTxRxResult(dxl_comm_result);
    // Clear syncwrite parameter storage
    groupSyncWrite.clearParam();
  }
  //--------------------------------------FIN PRE MODO FIN-----------------------------------
-------//
```

```
    if  ( receivedChars[0] == 'L' )
            { JJ=0;


          }
     tiempo = millis();
    if ( tiempo > t_actualizado + t_delay)
    {
    //----------------------------------------LEER DATOS----------------------------------------//
      // Leeer la temperatura
      dxl_comm_result  =  packetHandler->read1ByteTxRx(portHandler,  SERVO1, ADDR_TEMPERATURA,  (uint8_t*)&tem1, &dxl_error);
      if (dxl_comm_result != COMM_SUCCESS)
      {
        packetHandler->printTxRxResult(dxl_comm_result);
      }
      else if (dxl_error != 0)
      {
        packetHandler->printRxPacketError(dxl_error);
      }
      // Leeer Voltaje
        dxl_comm_result  =  packetHandler->read2ByteTxRx(portHandler,  SERVO1, ADDR_VOLTAJE,  (uint16_t*)&nVolt1, &dxl_error);
      if (dxl_comm_result != COMM_SUCCESS)
      {
        packetHandler->printTxRxResult(dxl_comm_result);
      }
      else if (dxl_error != 0)
      {
        packetHandler->printRxPacketError(dxl_error);
      }
```

```
// Leeer Velocidad
dxl_comm_result = packetHandler->read4ByteTxRx(portHandler, SERVO1, ADDR_TJVELOCIDAD, (uint32_t*)&velocidad1, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
  packetHandler->printTxRxResult(dxl_comm_result);
}
else if (dxl_error != 0)
{
  packetHandler->printRxPacketError(dxl_error);
}
// Leeer Carga
dxl_comm_result = packetHandler->read2ByteTxRx(portHandler, SERVO1, 126 , (uint16_t*)&carga1, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
packetHandler->printTxRxResult(dxl_comm_result);
}
else if (dxl_error != 0)
{
  packetHandler->printRxPacketError(dxl_error);
}
// Leeer la temperatura
dxl_comm_result = packetHandler->read1ByteTxRx(portHandler, SERVO2, ADDR_TEMPERATURA, (uint8_t*)&tem2, &dxl_error);
if (dxl_comm_result != COMM_SUCCESS)
{
  packetHandler->printTxRxResult(dxl_comm_result);
}
else if (dxl_error != 0)
{
```

```
    packetHandler->printRxPacketError(dxl_error);
  }
  // Leeer Voltaje
  dxl_comm_result = packetHandler->read2ByteTxRx(portHandler, SERVO2,
ADDR_VOLTAJE, (uint16_t*)&nVolt2, &dxl_error);
  if (dxl_comm_result != COMM_SUCCESS)
  {
  packetHandler->printTxRxResult(dxl_comm_result);
  }
  else if (dxl_error != 0)
  {
    packetHandler->printRxPacketError(dxl_error);
  }
  // Leeer Velocidad
  dxl_comm_result = packetHandler->read4ByteTxRx(portHandler, SERVO2,
ADDR_TJVELOCIDAD, (uint32_t*)&velocidad2, &dxl_error);
  if (dxl_comm_result != COMM_SUCCESS)
  {
  packetHandler->printTxRxResult(dxl_comm_result);
  }
  else if (dxl_error != 0)
  {
    packetHandler->printRxPacketError(dxl_error);
  }
  // Leeer Carga
  dxl_comm_result = packetHandler->read2ByteTxRx(portHandler, SERVO2,
126, (uint16_t*)&carga2, &dxl_error);
  if (dxl_comm_result != COMM_SUCCESS)
  {
    packetHandler->printTxRxResult(dxl_comm_result);
  }
```

```
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Leeer la temperatura
    dxl_comm_result = packetHandler->read1ByteTxRx(portHandler, SERVO3,
ADDR_TEMPERATURA, (uint8_t*)&tem3, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Leeer Voltaje
    dxl_comm_result = packetHandler->read2ByteTxRx(portHandler, SERVO3,
ADDR_VOLTAJE, (uint16_t*)&nVolt3, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // Leeer Velocidad
    dxl_comm_result = packetHandler->read4ByteTxRx(portHandler, SERVO3,
ADDR_TJVELOCIDAD, (uint32_t*)&velocidad3, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
```

```
    packetHandler->printTxRxResult(dxl_comm_result);
  }
  else if (dxl_error != 0)
  {
    packetHandler->printRxPacketError(dxl_error);
  }
  // Leeer Carga
  dxl_comm_result = packetHandler->read2ByteTxRx(portHandler, SERVO3,
126, (uint16_t*)&carga3, &dxl_error);
  if (dxl_comm_result != COMM_SUCCESS)
  {
    packetHandler->printTxRxResult(dxl_comm_result);
  }
  else if (dxl_error != 0)
  {
    packetHandler->printRxPacketError(dxl_error);
  }
 Serial.print("[ID: ");          Serial.print(SERVO1);
 Serial.print(" PresPos: ");     Serial.print(pos1);
 Serial.print(" PresPos2: ");    Serial.print(pos2);
 Serial.print(" PresPos3: ");    Serial.print(pos3);
 Serial.print(" Temperatura: "); Serial.print(tem1);
 Serial.print(" Voltaje: ");     Serial.print(nVolt1);
 Serial.print(" Velocidad: "); Serial.print(velocidad1);
 Serial.println(" ");

 int pos3d=pos3/1000;
 int pos3c=(pos3-(pos3d*1000))/100;
 int pos3b=(pos3- (pos3d*1000 + pos3c*100))/10;
 int pos3a=pos3-(pos3d*1000 + pos3c*100 + pos3b*10 );
 int pos2d=pos2/1000;
```

```
    int pos2c=(pos2-(pos2d*1000))/100;
    int pos2b=(pos2- (pos2d*1000 + pos2c*100))/10;
    int pos2a=pos2-(pos2d*1000 + pos2c*100 + pos2b*10 );
    int pos1d=pos1/1000;
    int pos1c=(pos1-(pos1d*1000))/100;
    int pos1b=(pos1- (pos1d*1000 + pos1c*100))/10;
    int pos1a=pos1-(pos1d*1000 + pos1c*100 + pos1b*10 );
    //Serial.print(pos3);Serial.print('-');Serial.print(pos2);Serial.print('-
');Serial.println(pos1);
    Serial2.print('<');Serial2.print('*');//Serial2.print("-");
    Serial2.print(pos1d);Serial2.print(pos1c);Serial2.print(pos1b);Serial2.print(pos1
a);Serial2.print('*');
    //Seri2al2.print("7777");Serial2.print('-');
    Serial2.print(pos2d);Serial2.print(pos2c);Serial2.print(pos2b);Serial2.print(pos2
a);Serial2.print('*');
    Serial2.print(pos3d);Serial2.print(pos3c);Serial2.print(pos3b);Serial2.println(pos
3a);Serial2.print("*");
    Serial2.print(tem1);Serial2.print('*');Serial2.print(tem2);Serial2.print('*');Serial2.p
rint(tem3);Serial2.print("*");
    Serial2.print(velocidad1);Serial2.print('*');Serial2.print(velocidad2);Serial2.print('
*');Serial2.print(velocidad3);;Serial2.print("*");
    Serial2.print(nVolt1);Serial2.print('*');Serial2.print(nVolt2);Serial2.print('*');Serial
2.print(nVolt3);Serial2.print("*");
    Serial2.print(carga1);Serial2.print('*');Serial2.print(carga2);Serial2.print('*');Serial
2.println(carga3);
    Serial2.print('>');
    //-----------------------------------------FIN LEER DATOS-----------------------------------
---------------//
    }
    }
    }
```

```
    dxl_comm_result = packetHandler->write1ByteTxRx(portHandler, SERVO1,
ADDR_PRO_TORQUE_ENABLE, TORQUE_DISABLE, &dxl_error);
    if (dxl_comm_result != COMM_SUCCESS)
    {
      packetHandler->printTxRxResult(dxl_comm_result);
    }
    else if (dxl_error != 0)
    {
      packetHandler->printRxPacketError(dxl_error);
    }
    // cerramos puerto
      portHandler->closePort();
    }
```

ANEXO 09

CÓDIGO EN MATLAB DE LA TRAYECTORIA “PICK AND PLACE”

```
clear all
clc
close all
global Bx1 Bx2 Bx3 By1 By2 By3 Bz1 Bz2 Bz3 R r L1 L2 phi1 phi2 phi3;
R=0.100;
r=0.050;
L1=0.250;
L2=0.400;
phi1=0;
phi2=120;
phi3=240;
R1 = R-r;
fm=0.050;
conv = (60/(2*pi))/0.229;

%Variables Dinámicas
Kr=1/353.5;          % Parámetro de Reducción de Engranajes
mbm = 0.038;      % Masa Base Móvil en kg
mload = 0.300;    % Masa de la Carga
ml1 = 0.259;      % Masa del Brazo 1
ml2 = 0.248;      % Masa de las dos Piezas del Brazo 2
mrot = 0.039;     % Masa de la Rotula conectada al Brazo 2
re = 2/3;         % Relacion de masa entre Eslaboón N°1 y eslabon N°2
g = 9.81;         % Gravedad
mtotalbm = mbm + mload + 6*(1-re)*ml2;          % Masa total Base Movil
rcm = L1*(((1/2)*ml1)+(mrot*4)+r*ml2)/(ml1+mrot+r*ml2);      %Centro de
Masa del Eslabón N°1
Jmotor = 0.0069;  % Momento de Inercia del Motor en Kg*cm^2 (Puede
cambiar)

Ibc = L1^2*((ml1/3)+mrot+r*ml2);      % Suma de la Inercia creada por la
masa de L1 y su inercia del extremo
Ibi = Jmotor*Kr^2 + Ibc;

%% Inicio del Programa
Px1=input('Ingrese Posciocion X inicial : ');
Py1=input('Ingrese Posciocion Y inicial : ');
Pz1=input('Ingrese Posciocion Z inicial : ');
Px2=input('Ingrese Posciocion X final : ');
Py2=input('Ingrese Posciocion Y final : ');
Pz2=input('Ingrese Posciocion Z final : ');
t=input('Ingrese el tiempo : ');
% Parametrización del vector
vdx = Px2-Px1;  % Vector Dirección X
vdy = Py2-Py1;  % Vector Dirección Y
vdz = Pz2-Pz1;  % Vector Dirección Z

mu=[0:fm:t];
[MU,MUD,MUDD] = jtraj(0,1,mu);
MU = MU';
MUD = MUD';
MUDD = MUDD';
%Posiciones Cartesianas
Pxin = Px1 + (MU*vdx);
Pyin = Py1 + (MU*vdy);
Pzin = Pz1 + (MU*vdz);

%Velocidades Cartesianas
Pxdin = (MUD*vdx);
Pydin = (MUD*vdy);
```

```
Pzdin = (MUD*vdz);

%Aceleraciones Cartesianas
Pxddin = (MUDD*vdx);
Pyddin = (MUDD*vdy);
Pzddin = (MUDD*vdz);

tamu= length(MU);
figure(1)
plot3(Pxin,Pyin,Pzin);xlabel('Px'); ylabel('Py'); zlabel('Pz');
grid on

figure(2)
subplot(3,3,1); plot(mu,Pxin,'r'); xlabel('Time (s)'); ylabel('Pos
X');grid on;
subplot(3,3,4); plot(mu,Pxdin,'r'); xlabel('Time (s)'); ylabel('Vel
X');grid on;
subplot(3,3,7); plot(mu,Pxddin,'r'); xlabel('Time (s)'); ylabel('Acel
X');grid on;

subplot(3,3,2); plot(mu,Pyin,'g'); xlabel('Time (s)'); ylabel('Pos
Y');grid on;
subplot(3,3,5); plot(mu,Pydin,'g'); xlabel('Time (s)'); ylabel('Vel
Y');grid on;
subplot(3,3,8); plot(mu,Pyddin,'g'); xlabel('Time (s)'); ylabel('Acel
Y');grid on;

subplot(3,3,3); plot(mu,Pzin,'b'); xlabel('Time (s)'); ylabel('Pos
Z');grid on;
subplot(3,3,6); plot(mu,Pzdin,'b'); xlabel('Time (s)'); ylabel('Vel
Z');grid on;
subplot(3,3,9); plot(mu,Pzddin,'b'); xlabel('Time (s)'); ylabel('Acel
Z');grid on;


%% Bucle de Cinemática Inversa

for j=1:1:tamu

Pnx = Pxin(1,j);
Pny = Pyin(1,j);
Pnz = Pzin(1,j);
% Posicion inicial
Si1 = (1/L1)*(-Pnx^2-Pny^2-Pnz^2+L2^2-L1^2-R1^2);
%Angulo theta1
Q11 = 2*Pnx*cosd(phi1)+2*Pny*sind(phi1);
D11= 4*Pnz^2+4*R1^2-Si1^2+Q11^2*(1-(R1^2/L1^2))+Q11*(((-2*R1*Si1)/(L1))-
4*R1);
E11= -2*R1-Q11*((R1/L1)-1)-Si1;
%Angulo theta2
Q21 = 2*Pnx*cosd(phi2)+2*Pny*sind(phi2);
D21 = 4*Pnz^2+4*R1^2-Si1^2+Q21^2*(1-(R1^2/L1^2))+Q21*(((-2*R1*Si1)/(L1))-
4*R1);
E21 = -2*R1-Q21*((R1/L1)-1)-Si1;
% Angulo theta3
Q31 = 2*Pnx*cosd(phi3)+2*Pny*sind(phi3);
D31 = 4*Pnz^2+4*R1^2-Si1^2+Q31^2*(1-(R1^2/L1^2))+Q31*(((-2*R1*Si1)/(L1))-
4*R1);
E31 = -2*R1-Q31*((R1/L1)-1)-Si1;
```

```
% RESULTADOS
th11 = (-2*atand(((-2*Pnz)+sqrt(D11))/E11));
th21 = (-2*atand(((-2*Pnz)+sqrt(D21))/E21));
th31 = (-2*atand(((-2*Pnz)+sqrt(D31))/E31));

q1(1,j)=th11;
q2(1,j)=th21;
q3(1,j)=th31;
end

figure(3)
subplot(3,1,1); stem(mu,q1,'r'); xlabel('Time (s)'); ylabel('Art-1
°');grid on;
subplot(3,1,2); stem(mu,q2,'g'); xlabel('Time (s)'); ylabel('Art-2
°');grid on;
subplot(3,1,3); stem(mu,q3,'b'); xlabel('Time (s)'); ylabel('Art-3
°');grid on;

%% Bucle de Cinemática directa
  for j=1:1:tamu;

theta1=-q1(1,j);
theta2=-q2(1,j);
theta3=-q3(1,j);

Bx1=(R-r)*cosd(phi1)+L1*cosd(phi1)*cosd(theta1);
By1=(R-r)*sind(phi1)+L1*sind(phi1)*cosd(theta1);
Bz1=-L1*sind(theta1);

Bx2=(R-r)*cosd(phi2)+L1*cosd(phi2)*cosd(theta2);
By2=(R-r)*sind(phi2)+L1*sind(phi2)*cosd(theta2);
Bz2=-L1*sind(theta2);

Bx3=(R-r)*cosd(phi3)+L1*cosd(phi3)*cosd(theta3);
By3=(R-r)*sind(phi3)+L1*sind(phi3)*cosd(theta3);
Bz3=-L1*sind(theta3);

x0=[1000 1000 1000];
P=fsolve('myfun',x0);
 if abs(P(1,1))<10^-3
     Pxx(1,j)=0;
 else
    Pxx(1,j) = P(1,1);
 end
 if abs(P(1,2))<10^-3
     Pyy(1,j)=0;
 else
    Pyy(1,j) = P(1,2);
 end
 if abs(P(1,3))<10^-3
     Pzz(1,j)=0;
 else
    Pzz(1,j) = P(1,3);
 end
  end
 figure(4)
plot3(Pxx,Pyy,Pzz);xlabel('Px CD'); ylabel('Py CD '); zlabel('Pz
CD');grid on
axis([-0.3 0.3 -0.3 0.3 -0.1 0.7])
```

```
theta1a=-q1(1,1);
%% JacobianA

for j=1:1:tamu;

    Px = Pxin(1,j);
    Py = Pyin(1,j);
    Pz = Pzin(1,j);
    theta1=-q1(1,j);
    theta2=-q2(1,j);
    theta3=-q3(1,j);
    Vx = Pxdin(1,j);
    Vy = Pydin(1,j);
    Vz = Pzdin(1,j);
    Ax=Pxddin(1,j);
    Ay=Pyddin(1,j);
    Az=Pzddin(1,j);
    % Jacobiana
    s1T=[Px-cosd(phi1)*(R1 + L1*cosd(theta1)), Py - sind(phi1)*(R1 +
L1*cosd(theta1)), Pz - (L1*sind(theta1))];
    s2T=[Px-cosd(phi2)*(R1 + L1*cosd(theta2)), Py - sind(phi2)*(R1 +
L1*cosd(theta2)), Pz - (L1*sind(theta2))];
    s3T=[Px-cosd(phi3)*(R1 + L1*cosd(theta3)), Py - sind(phi3)*(R1 +
L1*cosd(theta3)), Pz - (L1*sind(theta3))];
    b1=[-L1*cosd(phi1)*sind(theta1);-L1*sind(phi1)*sind(theta1);
L1*cosd(theta1)];
    b2=[-L1*cosd(phi2)*sind(theta2);-L1*sind(phi2)*sind(theta2);
L1*cosd(theta2)];
    b3=[-L1*cosd(phi3)*sind(theta3);-L1*sind(phi3)*sind(theta3);
L1*cosd(theta3)];
    siTT=[s1T;s2T;s3T]
    u=j
    K=[s1T*b1 0 0;0 s2T*b2 0;0 0 s3T*b3]

    Jacob = (siTT^-1)*K
    dq(:,:,j)    = ((Jacob^-1)*[Vx;Vy;Vz]);%*(60/(2*pi));     % Conversio
en RPM
    dq1(1,j) = dq(1,:,j);
    dq2(1,j) = dq(2,:,j);
    dq3(1,j) = dq(3,:,j);


    %Variables a Imprimir
    dqm1(1,j) = ceil(abs(dq(1,:,j)*conv));
    dqm2(1,j) = ceil(abs(dq(2,:,j)*conv));
    dqm3(1,j) = ceil(abs(dq(3,:,j)*conv));


    %Aceleración
    s1dT=[Vx+L1*cosd(phi1)*sind(theta1)*dq1(1,j) ,
Vy+L1*sind(phi1)*sind(theta1)*dq1(1,j) , Vz-L1*cosd(theta1)*dq1(1,j) ];
%Derivada de S1T
    s2dT=[Vx+L1*cosd(phi2)*sind(theta2)*dq2(1,j) ,
Vy+L1*sind(phi2)*sind(theta2)*dq2(1,j) , Vz-L1*cosd(theta2)*dq2(1,j) ];
%Derivada de S2T
    s3dT=[Vx+L1*cosd(phi3)*sind(theta3)*dq3(1,j) ,
Vy+L1*sind(phi3)*sind(theta3)*dq3(1,j) , Vz-L1*cosd(theta3)*dq3(1,j) ];
%Derivada de S3T
```

```
    b1d=[-L1*cosd(phi1)*cosd(theta1)*dq1(1,j) ;-
L1*cosd(theta1)*sind(phi1)*dq1(1,j) ;-L1*sind(theta1)*dq1(1,j) ];
%Derivada de b1
    b2d=[-L1*cosd(phi2)*cosd(theta2)*dq2(1,j) ;-
L1*cosd(theta2)*sind(phi2)*dq2(1,j) ;-L1*sind(theta2)*dq2(1,j) ];
%Derivada de b2
    b3d=[-L1*cosd(phi3)*cosd(theta3)*dq3(1,j) ;-
L1*cosd(theta3)*sind(phi3)*dq3(1,j) ;-L1*sind(theta3)*dq3(1,j) ];
%Derivada de b3
    sidT=[s1dT;s2dT;s3dT];
    OK = [s1dT*b1+s1T*b1d 0 0;0 s2dT*b2+s2T*b2d 0;0 0 s3dT*b3+s3T*b3d];
    Mn = (-sidT*Jacob+OK);
    Acar = [Ax;Ay;Az];

     ddq(:,:,j)  = (Jacob^-1)*(Acar-((siTT^-
1)*Mn*[dq1(1,j);dq2(1,j);dq3(1,j)]));
     Acelq       = (Jacob^-1)*(Acar-((siTT^-
1)*Mn*[dq1(1,j);dq2(1,j);dq3(1,j)]));

    ddq1(1,j) = ddq(1,:,j);
    ddq2(1,j) = ddq(2,:,j);
    ddq3(1,j) = ddq(3,:,j);


    %% Modelo Dinámico en Base al Trabajo Virtual

 % Para la base móvil
 Gn = mtotalbm*[0 0 -g]';    % Fuerza Gravitacional
 Fn = mtotalbm*Acar;         % Vector de Fuerza Axial
 Tnn = Jacob'*Fn;            % Vector de Fuerza de AXial de la base movil
 Tgn = Jacob'*Gn;            % Vector de Torques Gravitatorios

 % Para el Brazo 1
 Gb = -ml1*g;                 % Peso del Brazo 1
 Tgb = rcm*Gb*[cosd(theta1) cosd(theta2) cosd(theta3)]'; % Vector de
Torques Gravitacionales
 Ib = [Ibi 0 0;0 Ibi 0;0 0 Ibi];        %Matriz de Inercias de cada Brazo
 Tb = Ib*Acelq;              % Vector de Torques de cada brazo

 %Dinámica del robot
  Tmotor(:,:,j) = Tb + Tnn - (Tgn + Tgb);
  Tmotor1(1,j)=Tmotor(1,:,j);
  Tmotor2(1,j)=Tmotor(2,:,j);
  Tmotor3(1,j)=Tmotor(3,:,j);
end

figure(5)
subplot(3,3,1); plot(mu,q1,'r'); xlabel('Time (s)'); ylabel('Motor
1°');grid on;
subplot(3,3,4); plot(mu,dq1,'r'); xlabel('Time (s)'); ylabel('Vel 1
(rad/s)');grid on;
subplot(3,3,7); plot(mu,ddq1,'r'); xlabel('Time (s)'); ylabel('Acel
1(rad/s^2)');grid on;

subplot(3,3,2); plot(mu,q2,'g'); xlabel('Time (s)'); ylabel('Motor
2°');grid on;
subplot(3,3,5); plot(mu,dq2,'g'); xlabel('Time (s)'); ylabel('Vel
2(rad/s)');grid on;
```

```
subplot(3,3,8); plot(mu,ddq2,'g'); xlabel('Time (s)'); ylabel('Acel
2(rad/s^2)');grid on;

subplot(3,3,3); plot(mu,q3,'b'); xlabel('Time (s)'); ylabel('Motor
3°');grid on;
subplot(3,3,6); plot(mu,dq3,'b'); xlabel('Time (s)'); ylabel('Vel
3(rad/s)');grid on;
subplot(3,3,9); plot(mu,ddq3,'b'); xlabel('Time (s)'); ylabel('Acel
3(rad/s^2)');grid on;

figure(6)
subplot(3,1,1); plot(mu,Tmotor1,'r'); xlabel('Time (s)'); ylabel('Torque
motor 1 N.m');grid on;
subplot(3,1,2); plot(mu,Tmotor2,'g'); xlabel('Time (s)'); ylabel('Torque
motor 2 N.m');grid on;
subplot(3,1,3); plot(mu,Tmotor3,'b'); xlabel('Time (s)'); ylabel('Torque
motor 3 N.m');grid on;


figure(7)
subplot(3,1,1); plot(mu,MU,'r'); xlabel('Time (s)'); ylabel('Posición
Interpolador quintico ');grid on;
subplot(3,1,2); plot(mu,MUD,'g'); xlabel('Time (s)'); ylabel('1ra
DevidadaInterpolador quintico');grid on;
subplot(3,1,3); plot(mu,MUDD,'b'); xlabel('Time (s)'); ylabel('2da
Devidada Interpolador quintico°');grid on;
```

ANEXO 10

CODIGO ESTRUCTURADO DE LA INTERFAZ GRÁFICA – VISUAL STUDIO 2017

CÓDIGO MENÚ PRINCIPAL

```
' --------------------Codigo de la Interfaz Principal ROBOT PARALELO DELTA---------
--------
'---Autores:-----------------------------José Paz Pinto - Arturo Lazo Pinto-------
--------
Imports System
Imports System.IO.Ports
Imports System.Runtime.InteropServices 'Bibiioteca para detectar unidades USB



Public Class Form1
    ' Variables globales
    Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As
Integer, ByVal wMsg As Integer, ByVal wParam As Integer,
                                                                    ByVal lParam As
Integer) As Integer
    Declare Function joyGetPosEx Lib "winmm.dll" (ByVal uJoyID As Integer, ByRef
pji As JOYINFOEX) As Integer
    Dim comPORT As String
    Dim receivedData As String = ""
    Dim connected As Boolean = False
    Dim count = 0
    Public T1 As String = ""
    Public T2 As String = ""
    Public T3 As String = ""
    Public Vel1 As String = ""
    Public Vel2 As String = ""
    Public Vel3 As String = ""
    Public Voltaje1 As String = ""
    Public Voltaje2 As String = ""
    Public Voltaje3 As String = ""
    Public Carga1 As String = ""
    Public Carga2 As String = ""
    Public Carga3 As String = ""

<StructLayout(LayoutKind.Sequential)>
    Public Structure JOYINFOEX  ' Variables Joystick
        Public dwSize As Integer 'Size, in bytes, of this structure.
        Public dwFlags As Integer 'Flags indicating the valid information returned
in this structure. Members that do not contain valid information are set to zero.
The following flags are defined:VER PAGINA WEB
        Public dwXpos As Integer 'Current X-coordinate.
        Public dwYpos As Integer 'Current Y-coordinate.
        Public dwZpos As Integer 'Current Z-coordinate.
        Public dwRpos As Integer 'Current position of the rudder or fourth joystick
axis.
        Public dwUpos As Integer 'Current fifth axis position.
        Public dwVpos As Integer 'Current sixth axis position.
        Public dwButtons As Integer 'Current state of the 32 joystick buttons. The
value of this member can be set to any combination of JOY_BUTTONn flags, where n is
a value in the range of 1 through 32 corresponding to the button that is pressed.
        Public dwButtonNumber As Integer 'Current button number that is pressed.
        Public dwPOV As Integer 'Current position of the point-of-view control.
Values for this member are in the range 0 through 35,900. These values represent
the angle, in degrees, of each view multiplied by 100.
        Public dwReserved1 As Integer 'Reserved; do not use.
        Public dwReserved2 As Integer 'Reserved; do not use.
    End Structure
    Dim myjoyEX As JOYINFOEX
```

```
    ' Cuando se inicie el programa se asegura que el temporizador este apagadoy que
se abran los puertos seriales
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Timer1.Enabled = False
        populateCOMport()
    End Sub

    'Actualiza los puertos COM
    Private Sub refreshCOM_BTN_Click(sender As Object, e As EventArgs) Handles
refreshCOM_CB_BTN.Click
        SerialPort1.Close()
        populateCOMport()
    End Sub

    Private Sub populateCOMport()
        comPORT = ""
        comPort_ComboBox.Items.Clear()
        For Each sp As String In My.Computer.Ports.SerialPortNames
            comPort_ComboBox.Items.Add(sp)
        Next
    End Sub

    Private Sub comPort_ComboBox_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles comPort_ComboBox.SelectedIndexChanged
        If (comPort_ComboBox.SelectedItem <> "") Then
            comPORT = comPort_ComboBox.SelectedItem
        End If
    End Sub

    ' Cuando el boton "Conectar" es presionado y se ha seleccionado un puerto se
envia un mensaje "HELLO" al puerto COM
    ' Luego espera a que el Arduino Mega responda este mensaje
    ' Cuando se recibe un "HELLO" por parte del Arduino Mega se cambia el botón
Conectar a Desconectar
    Private Sub connect_BTN_Click(sender As Object, e As EventArgs) Handles
connect_BTN.Click
        comPORT = comPort_ComboBox.SelectedItem
        If (connect_BTN.Text = "Conectar") Then
            If (comPORT <> "") Then
                SerialPort1.Close()
                SerialPort1.PortName = comPORT
                SerialPort1.BaudRate = 57600
                SerialPort1.DataBits = 8
                SerialPort1.Parity = Parity.None
                SerialPort1.StopBits = StopBits.One
                SerialPort1.Handshake = Handshake.None
                SerialPort1.Encoding = System.Text.Encoding.Default
                SerialPort1.ReadTimeout = 10000

                SerialPort1.Open()

                'Revisa si el Arduino Mega esta conectado
                count = 0
                SerialPort1.WriteLine("<HELLO>")
                connect_BTN.Text = "Conectando..."
                connecting_Timer.Enabled = True
Else
                MsgBox("Seleccione un puerto COM")
            End If
        Else
```

```
            'Cierra la conexión
            Timer1.Enabled = False
            Timer_LBL.Text = "Timer: OFF"
            SerialPort1.Close()
            connected = False
            connect_BTN.Text = "Conectar"
            populateCOMport()
        End If


    End Sub

    'El temporizador de conexión espera el mensaje del Arduino Mega
    ' Si no se recibe un "HELLO" por parte del arduino aparecera un mensaje de
error en la conexión
    ' El temporizador de conexión se usa solo para conectar
    Private Sub connecting_Timer_Tick(sender As Object, e As EventArgs) Handles
connecting_Timer.Tick
        connecting_Timer.Enabled = False
        count = count + 1

        If (count <= 8) Then
            receivedData = receivedData & ReceiveSerialData()

            If (Microsoft.VisualBasic.Left(receivedData, 5) = "HELLO") Then
                ' Si se recibe un "HELLO" por parte del Arduino Mega signica que
existe una conexión
                connected = True
                connect_BTN.Text = "Desconectar"
                Timer1.Enabled = True
                Timer_LBL.Text = "Timer: ON"
                receivedData = ReceiveSerialData()
                receivedData = ""
                SerialPort1.WriteLine("<START>")

            Else
                'Arranca el temporizador otra vez y sigue esperando por una señal
del Arduino Mega
                connecting_Timer.Enabled = True
            End If


        Else

            connect_BTN.Text = "Conectar"
            populateCOMport()
        End If



    End Sub


'Luego que se registra una conexión satisfactoria se espera la dara recibida por el
Arduino Mega constantemente

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Dim Px As String
        Dim Py As String
        Dim Pz As String
        Dim A1 As String
        Dim A2 As String
```

```
        Dim A3 As String

        Dim A1d As Double
        Dim A2d As Double
        Dim A3d As Double
        Dim Pox As Double
        Dim Poy As Double
        Dim Poz As Double


        joyGetPosEx(0, myjoyEX)
        receivedData = ReceiveSerialData()

        Dim split = receivedData.Split("*"c) ' Se separa la data recibida
        If (split.Count = 19) Then
            Px = split(1).ToString
            Py = split(2).ToString
            Pz = split(3).ToString
            A1 = split(4).ToString
            A2 = split(5).ToString
            A3 = split(6).ToString
            T1 = split(7).ToString
            T2 = split(8).ToString
            T3 = split(9).ToString
            Vel1 = split(10).ToString
            Vel2 = split(11).ToString
            Vel3 = split(12).ToString
            Voltaje1 = split(13).ToString
            Voltaje2 = split(14).ToString
            Voltaje3 = split(15).ToString
            Carga1 = split(16).ToString
            Carga2 = split(17).ToString
            Carga3 = split(18).ToString


        End If

        Dim PPX As Double
        Dim PPY As Double
        Dim PPZ As Double
        Dim AA1 As Double
        Dim AA2 As Double
        Dim AA3 As Double
If (Decimal.TryParse(Px, PPX)) Then
            PosiX.Text = Px
        End If
        If (Decimal.TryParse(Py, PPY)) Then
            PosiY.Text = Py
        End If
        If (Decimal.TryParse(Pz, PPZ)) Then
            PosiZ.Text = Pz
        End If

        If (Int32.TryParse(A1, AA1)) Then
            AA1 = (AA1 - 1023) * 0.088
            Ang1.Text = Format(AA1)
        End If
        If (Int32.TryParse(A2, AA2)) Then
            AA2 = (AA2 - 1023) * 0.088
            Ang2.Text = Format(AA2)
        End If
        If (Int32.TryParse(A3, AA3)) Then
```

```
            AA3 = (AA3 - 1023) * 0.088
            Ang3.Text = Format(AA3)
        End If

        ' El joystick necesita actualizar constantemente sus datos es por esto que
se encuentra en el timer
        ' Si el boton de Joystick se encuentre encendido se se envia la data del
Joystick; si se encuentra apagado recibe data constantemente del Arduino Mega
        If (BJoystick.Text = "Joystick_off") Then
            Dim datas As String
            datas = "<J" & Format(Math.Round(myjoyEX.dwXpos / 16), "0000") &
Format(Math.Round(myjoyEX.dwYpos / 16), "0000") & Format(Math.Round(myjoyEX.dwZpos
/ 16), "0000") & Format(Math.Round(myjoyEX.dwButtons / 1), "0") & "
>"
            SerialPort1.WriteLine(datas)

        ElseIf (BJoystick.Text = "Joystick_on") Then
            SerialPort1.WriteLine("<W123456789
>")
        End If



        Dim tmp As String
        tmp = Microsoft.VisualBasic.Left(receivedData, 3)

        ' Se secciona la data a ser visualizada en otras ventanas
        Module1.Data = receivedData
        Module1.Volts1 = Voltaje1
        Module1.Volts2 = Voltaje2
        Module1.Volts3 = Voltaje3
        Module1.Corriente1 = Carga1
        Module1.Corriente2 = Carga2
        Module1.Corriente3 = Carga3
        Module1.Velo1 = Vel1
        Module1.Velo2 = Vel2
        Module1.Velo3 = Vel3
        Module1.Temp1 = T1
        Module1.Temp2 = T2
        Module1.Temp3 = T3

    End Sub

    Function ReceiveSerialData() As String
        Dim Incoming As String
        Try

            Incoming = SerialPort1.ReadExisting()
            If Incoming Is Nothing Then
                Return "nothing" & vbCrLf
            Else
                Return Incoming
            End If
        Catch ex As TimeoutException
            Return "Error: Serial Port read timed out."
        End Try

    End Function


    ' Slider de velocidad
```

```
    Private Sub red_TrackBar_Scroll(sender As Object, e As EventArgs) Handles
red_TrackBar.Scroll
        updateRGBlbl()
    End Sub



    ' Actualización constante del Slider de Velocidad
    Private Sub updateRGBlbl()
        slider_LBL.Text = "S1:" & Format(red_TrackBar.Value, "000") & "  S2:" &
Format(red_TrackBar.Value, "000") & "  S3:" & Format(red_TrackBar.Value, "000")
        If (connected) Then
            Dim data As String
            data = "<V" & Format(red_TrackBar.Value, "000") &
Format(red_TrackBar.Value, "000") & Format(red_TrackBar.Value, "000") & ">"
            SerialPort1.WriteLine(data)
        End If

    End Sub
'Se añade los iniciadores y terminadores y se envia data manualmente
    Private Sub send_BTN_Click(sender As Object, e As EventArgs) Handles
send_BTN.Click
        If (connected) Then
            Dim tmp As String
            tmp = "<" & send_TB.Text & ">"
            SerialPort1.WriteLine(tmp)
        End If
    End Sub



    ' Si se hace click al boton de Joystick inicializa los comandos del Joystick
    Private Sub BJoystick_Click(sender As Object, e As EventArgs) Handles
BJoystick.Click
        If (BJoystick.Text = "Joystick_on") Then
            myjoyEX.dwSize = 64
            myjoyEX.dwFlags = &HFF
            BJoystick.Text = "Joystick_off"
        ElseIf (BJoystick.Text = "Joystick_off") Then
            myjoyEX.dwSize = 0
            myjoyEX.dwFlags = &HFF
            BJoystick.Text = "Joystick_on"
        End If
    End Sub
    'Boton de Enviar Posiciones Deseadas
    Private Sub Benviar_Click(sender As Object, e As EventArgs) Handles
Benviar.Click
        If (connected) Then
            Dim datapos As String
            datapos = "<A" & Pdx.Text & Pdy.Text & Pdz.Text & ">"
            SerialPort1.WriteLine(datapos)
        End If
    End Sub

    'Botón Para Limpiar los cuadros de texto de las posiciones deseadas
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        Pdx.Text = ""
        Pdy.Text = ""
        Pdz.Text = ""
    End Sub
    ' Botones para encender o apagar la ventosa
```

```
    Private Sub BotonVentosa_Click(sender As Object, e As EventArgs) Handles
BotonVentosa.Click
        If (BotonVentosa.Text = "Ventosa ON") Then
            If (connected) Then
                SerialPort1.WriteLine("<G>")
            End If
            BotonVentosa.Text = "Ventosa OFF"
        ElseIf (BotonVentosa.Text = "Ventosa OFF") Then
            If (connected) Then
                SerialPort1.WriteLine("<H>")
            End If
            BotonVentosa.Text = "Ventosa ON"
        End If
    End Sub
    ' Botón para enviar el robot paralelo delta a una posición preestablecida
    Private Sub BotonHome_Click(sender As Object, e As EventArgs) Handles
BotonHome.Click
        If (connected) Then
            SerialPort1.WriteLine("<C>")
        End If
    End Sub
    ' Botón para APAGAR el robot paralelo delta
    Private Sub ButtonOFF_Click(sender As Object, e As EventArgs) Handles
ButtonOFF.Click
        If (connected) Then
            SerialPort1.WriteLine("<F>")
        End If
    End Sub
    '  Botones para inicializar las ventanas de parametros
    Private Sub pruebaven_Click(sender As Object, e As EventArgs) Handles
pruebaven.Click
        ' Module1.nombre = slider_XYZ.Text
        Dim f6 As New Form6()
        f6.Show()
    End Sub

    Private Sub venTemp_Click(sender As Object, e As EventArgs) Handles
venTemp.Click
        Dim f3 As New Form3()
        f3.Show()
    End Sub

    Private Sub venVel_Click(sender As Object, e As EventArgs) Handles venVel.Click
        Dim f4 As New Form4()
        f4.Show()
    End Sub

    Private Sub venData_Click(sender As Object, e As EventArgs) Handles
venData.Click
        Dim f5 As New Form5()
        f5.Show()
    End Sub


End Class
```

CÓDIGO MENÚ DE PARÁMETROS ELÉCTRICOS

```
Public Class Form6
    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Dim valor As Integer = 50
        Dim vs1 As Integer
        Dim vs2 As Integer
        Dim vs3 As Integer
        Dim v1 As Double
        Dim v2 As Double
        Dim v3 As Double
        Dim cs1 As Integer
        Dim Is1 As Double
        Dim cs2 As Integer
        Dim Is2 As Double
        Dim cs3 As Integer
        Dim Is3 As Double

        If (Int32.TryParse(Module1.Volts1, vs1)) Then
            v1 = vs1 / 10
            AGauge1.Value = v1
            AGauge1.Refresh()
        End If
        If (Int32.TryParse(Module1.Volts2, vs2)) Then
            v2 = vs2 / 10
            AGauge2.Value = v2
            AGauge2.Refresh()
        End If
        If (Int32.TryParse(Module1.Volts3, vs3)) Then
            v3 = vs3 / 10
            AGauge3.Value = v3
            AGauge3.Refresh()
        End If

        If (Int32.TryParse(Module1.Corriente1, cs1)) Then
            Is1 = Math.Abs(cs1 * 0.00269) * 1000
        End If
        If (Int32.TryParse(Module1.Corriente2, cs2)) Then
            Is2 = Math.Abs(cs2 * 0.00269) * 1000
        End If
        If (Int32.TryParse(Module1.Corriente3, cs3)) Then
            Is3 = Math.Abs(cs3 * 0.00269) * 1000
        End If
        LabelVolt1.Text = Format(v1)
        LabelVolt2.Text = Format(v2)
        LabelVolt3.Text = Format(v3)
        LabelC1.Text = Format(Is1)
        LabelC2.Text = Format(Is2)
        LabelC3.Text = Format(Is3)
        LabelW1.Text = Format(Math.Round(((v1 * Is1) / 1000), 2))
        LabelW2.Text = Format(Math.Round(((v2 * Is2) / 1000), 2))
        LabelW3.Text = Format(Math.Round(((v3 * Is3) / 1000), 2))

    End Sub
```

```
End Class
```

CÓDIGO MENÚ DE PARÁMETROS DE TEMPERATURA

```
Public Class Form3
    Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

        Dim Ts1 As Double
        Dim Ts2 As Double
        Dim Ts3 As Double


        If (Int32.TryParse(Module1.Temp1, Ts1)) Then
            AGauge1.Value = Ts1
            AGauge1.Refresh()
        End If
        If (Int32.TryParse(Module1.Temp2, Ts2)) Then
            AGauge2.Value = Ts2
            AGauge2.Refresh()
        End If
        If (Int32.TryParse(Module1.Temp3, Ts3)) Then
            AGauge3.Value = Ts3
            AGauge3.Refresh()
        End If


        LabelT1.Text = Format(Ts1)
        LabelT2.Text = Format(Ts2)
        LabelT3.Text = Format(Ts3)


    End Sub

                                End Class
```

CÓDIGO MENÚ DE PARÁMETROS DE VELOCIDADES

```
Public Class Form4
    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        Dim Vels1 As Double
        Dim Vels2 As Double
        Dim Vels3 As Double

        Int32.TryParse(Module1.Velo1, Vels1)
        Int32.TryParse(Module1.Velo2, Vels2)
        Int32.TryParse(Module1.Velo3, Vels3)

        LabelVel1.Text = Format(Vels1)
        LabelVel2.Text = Format(Vels2)
        LabelVel3.Text = Format(Vels3)
    End Sub
                                    End Class
```

CÓDIGO MENÚ DE PARÁMETROS DE DATA

```
blic Class Form5
    Private Sub Form5_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
    End Sub
    Private Sub clear_BTN_Click(sender As Object, e As EventArgs) Handles
clear_BTN.Click
        RichTextBox1.Text = ""
    End Sub

    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        RichTextBox1.Text &= Module1.Data
    End Sub
End Class
```

CÓDIGO MODULO

```
Module Module1

    Public Data As String
    Public Volts1 As String
    Public Volts2 As String
    Public Volts3 As String
    Public Corriente1 As String
    Public Corriente2 As String
    Public Corriente3 As String
    Public Velo1 As String
    Public Velo2 As String
    Public Velo3 As String
    Public Temp1 As String
    Public Temp2 As String
    Public Temp3 As String

End Module
```

ANEXO 11

COSTOS DE PROYECTO

Dispositivo o Proceso	Cant.	Unidad	Precio Unitario ($)	Precio Total($)
Dynamixel XM-430-W350 - T	3	UND	229.9	689.7
OpenCM9.04	1	UND	15	15
Arduino MEGA	1	UND	20	20
Bomba de Vacío AIRPO	1	UND	27	27
Ventosa FESTO VASB-15-1/8	1	UND	23	23
Placa Impresa con Componentes	1	UND	7.6	7.6
HUB de USB Genius	1	UND	9.8	9.8
Joystick Metal Strike	1	UND	29	29
Chumacera M8-KFL08	3	UND	5	15
Rodamiento Lineal LM8UU	3	UND	1.6	4.8
Eje Lineal INOX 8mm	1	UND	9.5	9.5
Rotulas Esféricas SQ-6CRS	12	UND	6.75	81
Nylon 66	2	Barras	9.2	18.4
Fuente de Voltaje 12V - 10 Amp	1	UND	14	14
Tubo Cuadrado de Acero HSS 1" x 1" x 2mm	2	Tubos	8	16
Pintura	1/4	Galón	60	15
Cable USB	3	UND	3	9
Cableado	5	METROS	1	5
Manguera Neumática	3	METROS	1.1	3.3
Componentes Neumáticos	1	UND	3	3
Tablero de Control	1	UND	10	10
Manguera Corrugada	1	METROS	1.1	1.1
Canaleta	1	UND	3	3
Empernado	40	UND	0.5	20
Impresión 3D	1	---	325	325
Reforzamiento en Fibra de Vidrio	1	---	340.5	340.5
Mano de Obra	1	---	137	137
Gastos de Importación	1	---	250	250
Otros	1	---	450	450
TOTAL				$ 2551.70

Tipo de Cambio Considerado S/. 3.27