UNIVERSIDAD CATÓLICA DE SANTA MARÍA

Facultad de Ciencias e Ingenierías Físicas y Formales

Programa Profesional de Ingeniería Mecánica, Mecánica Eléctrica y Mecatrónica

Control Kinect de un brazo robótico de 5 GDL a través de una interfaz intuitiva

TESIS PRESENTADA POR EL BACHILLER:

Carlofranco Ruiz Daza

Para optar el título profesional de:

INGENIERO MECATRÓNICO

Arequipa-Perú

2014

A mis padres,
por enseñarme a navegar
en las mareas del tiempo.

C.R.D.

Agradecimientos

Quiero agradecer en primer lugar a Dios, por el regalo más hermoso que a un hijo Él puede dar: sus padres y el milagro de su amor.

A ti Virgen María, Madre mía Inmaculada, a quien aprendí a amar y defender desde el primer momento que te conocí, y a quien nunca podré olvidar, porque te convertiste en el lucero que alumbra y protege cada uno de mis días.

A mis padres, por darme junto al Señor la vida, por ser el ejemplo impecable de mi existencia, por enseñarme más de lo que un hijo desearía aprender, por transmitirme ese espíritu a prueba de debilidades, por imprimir en mí esa personalidad sólida y arrolladora que no claudica ante las adversidades, por fortalecer mi alma, esa que nunca se vende ni se rinde ante nada y ante nadie, y que posee el fuego inacabable de la perseverancia. Muchas gracias por estar siempre junto a mí, por su apoyo incondicional, por ese aliento de su parte que nunca estuvo ausente, y que me motivaba a luchar por mis sueños e ideales, por sus sabios consejos que nunca fueron inoportunos cuando la duda y la indecisión me invadía. Gracias también por acompañarme en cada paso de mi vida, en cada proyecto que iniciaba, en esas largas horas de estudio e investigación, por sacrificarlo todo desinteresadamente, y en definitiva, por hacer de mí un hombre de bien, un hombre que espera colmarlos siempre de inmensas alegrías y del cual se sientan eternamente orgullosos. Gracias papás. Por todo esto y mucho más.

A mi querido colegio San José, donde pasé los años más hermosos de mi vida, donde mi espíritu se hizo jesuita, donde bajo el manto sagrado de la Virgen Inmaculada mi madre me entregó y en cuyos brazos se pasó como un sueño mi niñez. Gracias por enseñarme a ser más para servir mejor, a que todo lo que haga siempre sea a mayor gloria de Dios, y principalmente, en todo amar y servir.

Quiero agradecer también a todas aquellas personas que contribuyeron con su experiencia y conocimientos en hacer de esta investigación, una realidad.

A los asesores de mi *alma mater*, la Universidad Católica de Santa María, el Ing. Sergio Mestas Ramos y el Ing. Juan Carlos Cuadros Machuca, por contribuir a mi formación profesional durante mis estudios de pregrado, por transmitirme sus experiencias y conocimientos académicos no solo en las horas de teoría sino también en las de práctica, por su asesoramiento y consejos en cada revisión de mis avances de tesis, que siempre fueron un aporte importante para la investigación, por su sincera amistad, por atender mis llamadas cada vez que los necesitaba, y sobre todo, por ser los primeros en enseñarme ese mundo fascinante de la Mecatrónica, y transformarse en los gestores de la

decisión de pertenecer perpetuamente a ella. Ha sido para mí un honor conocerlos, ser su alumno y compartir junto a ustedes numerosos momentos de cátedra.

Al Ing. Seki Kenzaburo, por haber logrado convencerme de que la idea de utilizar el sensor Kinect fuera de los salones de videojuegos no era un disparate, sino una genialidad, y porque en uno de tus viajes, te diste la molestia de adquirirlo por mí en el Japón. Donde te encuentres, estarás siempre en mi memoria Seki-Sama.

Al Ing. Nilton Anchayhua Aréstegui de TECSUP, por su asesoramiento incondicional, por ser mi guía a lo largo de toda la investigación, por compartir su valioso tiempo en revisar todos mis avances y en resolver mis dudas, por impulsarme a dar lo mejor de mí y nunca rendirme cuando el camino se ponía cuesta arriba, y por enseñarme los secretos de la Robótica y la Ingeniería de Control, incrementando así mis conocimientos. Gracias por convertirse en el referente de Ingeniero Mecatrónico que deseo imitar, por toda su amabilidad y ayuda desinteresada, por permitirme trabajar junto a usted, lo que me hace sentir muy privilegiado y eternamente agradecido.

A HICOSER S.R.L. por proveer la materia prima para fabricar la estructura del brazo robótico. Al Lic. Miguel Ángel Peralta Barriga de Representaciones Panchito E.I.R.L., por su cordial apoyo y servicio al momento de adquirir las uniones roscadas que necesitaba. Al Maestro Edmundo Talavera Sánchez y su hijo Darwin, en cuya maestranza se llevó a cabo la mecanización y ensamblaje de todas y cada una de las piezas. Muchas gracias por su talento y pericia, por sus valiosos consejos y don de gente, y sobre todo, por esa infinita paciencia que les permitió comprender mis exigencias y "caprichos" estructurales. A Trossen Robotics, por ayudarme a adquirir e importar los accesorios y actuadores que dan vida al robot.

Finalmente, quiero agradecer a mi familia, a todos mis amigos y amigas, y a todas las personas que han formado parte de mi vida, por su amistad, consejos, apoyo, ánimo y compañía, tanto en los momentos buenos como en los malos. Algunas están aquí conmigo y otras, en mis recuerdos y en mi corazón. Sin importar en donde estén, quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todas sus bendiciones. En especial quiero agradecerte a ti Mayte, por completar la dosis de equilibrio que le faltaban a mis emociones y sentimientos, por tus consejos, apoyo y compañía incondicional a lo largo de esta investigación, por impulsarme a trazar metas cada vez más ambiciosas, y sobre todo, por darle sentido al amor, ese amor de mujer que todo hombre necesita.

Para todos ustedes: muchas gracias y que Dios los bendiga.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Resumen

La presente investigación despliega la creación de un sistema de control gestual, que permite interactuar con un brazo robótico a través de gestos y movimientos corporales captados desde un sensor Kinect. Para poder implementar dicho control, se desarrolla una plataforma experimental científica denominada Darko, la cual alberga dentro de su equipamiento: un robot de 5 grados de libertad (GDL), un sensor Kinect, conocido por su capacidad de reconocimiento corporal, y que junto a un ordenador portátil, permiten crear una interfaz natural de usuario, a través de la cual el ser humano se convierte en el instrumento que "mueve" al robot, cuyo diseño, simulación y puesta en marcha, se incluyen sin dejar de lado los detalles pertinentes.

El contenido de la tesis está organizado en los siguientes capítulos:

El **Capítulo I Planteamiento Teórico** presenta el soporte científico sobre el que se construye toda la investigación, el cual se inicia con la identificación y descripción del problema de estudio; luego se revisa, de manera sucinta, los antecedentes más relacionados con la tesis y de mayor rigor científico. Seguidamente, se trazan los objetivos que se desean alcanzar al finalizar la labor investigativa, se formula la hipótesis y se identifican las variables pertinentes. Finalmente, se realiza la justificación de la investigación, se detallan los alcances y limitaciones correspondientes, concluyendo con la descripción de la metodología a utilizar, la cual está basada en el método científico.

El **Capítulo II Marco Teórico** recopila, de manera sintetizada, todo el fundamento teórico necesario para entender paso a paso y con plenitud, el desarrollo íntegro de la investigación. Claramente pueden identificarse tres secciones de teoría: la primera relacionada netamente a la robótica (con todo su tratamiento matemático), la segunda vinculada al sensor Kinect, y la última referida a las Interfaces Naturales de Usuario (NUI).

El **Capítulo III Diseño del brazo robótico** considera todo el proceso de diseño aplicado del robot, desde su etapa conceptual, pasando por el análisis cinemático (donde se estudia la geometría del movimiento) y dinámico (que incluye selección de materiales estructurales y actuadores), hasta llegar al entorno computacional avanzado: el Diseño e Ingeniería Asistida por Computadora (CAX), elaborado en Autodesk® Inventor® Professional.

El **Capítulo IV Control Dinámico del brazo robótico** contiene el desarrollo de la estrategia de control conocida como Control PID con compensación de gravedad, que es la encargada de gobernar el movimiento automático del robot. Al ser un requisito preliminar para alcanzar el Control Kinect, se procedió a dividir el capítulo en dos etapas.

La primera, compuesta por el control de los 3 primeros GDL (grados de posición), y la segunda, conformada por el control de los 5 GDL, todo ello bajo la interfaz de Matlab®.

El **Capítulo V Control Kinect. Diseño de la interfaz intuitiva** detalla el diseño de la interfaz intuitiva a través de Visual Studio Express 2012, y muestra cómo, a partir de los gestos y movimientos corporales del usuario, pueden accionarse los actuadores del robot para controlar su movimiento.

El **Capítulo VI Pruebas, Resultados y Análisis** es una colección de los ensayos realizados del Control Kinect, cuyos resultados obtenidos son analizados y discutidos a fin de comprobar, no solo si los objetivos fueron alcanzados o si fue posible validar la hipótesis, sino también si el conocimiento científico condensado en el marco teórico, se demuestra verdaderamente a través de la práctica.

La presente investigación evidencia que todos los esfuerzos se han fusionado y conducido con intenso esmero, para incorporar al mundo de la robótica, una estrategia de control muy particular e innovadora, que vaya en sintonía con la última tendencia tecnológica de interfaces de usuario.

Introducción

Cuando Microsoft® presentó en el E3 (Electronic Entertainment Expo) del 2010 su nuevo dispositivo Kinect, todos los "gamers" del mundo cambiaron su forma de percibir los videojuegos. Sin embargo, en las mentes de ingeniería nació la idea de que este periférico podría ser utilizado en otras aplicaciones, como en proyectos de investigación, cuyo impacto tecnológico transforme ámbitos tan variados como la educación, el comercio, la industria, el arte y la medicina. Este proyecto, es uno de ellos.

Actualmente, existen diversas formas de interactuar directamente con los dispositivos tecnológicos. Sin embargo, cada día se busca la forma de que dicha interacción se realice de la manera más natural posible, es decir, sin la necesidad de usar herramientas especiales para manipular la información que se nos presente. Así, dispositivos como el mouse, el teclado o el joystick, se han venido reemplazando paulatinamente por las pantallas táctiles, una opción de contacto más directo. En el mercado del ocio y entretenimiento, se están normalizando aquellas interfaces que nos permiten trabajar con gestos o ademanes corporales: hablamos de las interfaces naturales, caracterizadas por ofrecer un control gestual del contexto que se afronta.

Con la finalidad de trasladar este tipo de interfaces al mundo de la Ingeniería Mecatrónica, el Control Kinect de un brazo robótico de 5 GDL a través de una interfaz intuitiva, pretende ser la llave que apertura el camino hacia el desarrollo de una nueva tecnología robótica, cuyo impacto aporte soluciones innovadoras que optimicen el control, garanticen una rápida y segura interacción hombre-máquina, y trasciendan más allá de los límites de la ingeniería.

En las dos últimas décadas, la robótica se ha convertido en un área estratégica y vital para todo país en desarrollo. Hablar de ella es sinónimo de automatización, modernización y precisión. Cuando se inicia su estudio en la universidad, el gusto por ella a través del tiempo hace que se transforme en pasión. Una pasión que te envuelve y te conduce a sentirte comprometido con el desarrollo tecnológico de tu nación. Ese compromiso se ve materializado en la construcción y puesta en marcha de la plataforma científica Darko, fruto del conocimiento y el arte mecatrónico, que cobija un brazo robótico capaz de ser controlado por el lenguaje natural de los gestos.

Que este aporte tecnológico y científico sea el inicio de una nueva era en la robótica, porque la robótica no sólo es tecnología y conocimientos, sino también arte y cultura.

Capítulo I

Planteamiento Teórico

1.1. Título

"Control Kinect de un brazo robótico de 5 GDL a través de una interfaz intuitiva".

1.2. Identificación y descripción del problema

La naturaleza multidisciplinaria de la robótica involucra una gran cantidad de áreas del conocimiento (matemáticas, física, electrónica, computación, visión e inteligencia artificial, entre otras). Por otro lado, si bien la robótica es un área inminentemente práctica y experimental, todos sus resultados están sustentados con estricto rigor científico. Es por ello que, para obtener una comprensión sistémica y completa del diseño y control de un brazo robótico, es preciso enlazar la teoría con la práctica; enlace que no siempre se da en la dimensión adecuada, debido a que en la actualidad, la gran mayoría de universidades y centros de investigación carecen de infraestructura adecuada en materia de robótica. De ahí que los trabajos de tesis, reportes técnicos y publicaciones científicas presentan resultados sólo de simulación, y muy pocos incluyen evaluación experimental.

Las simulaciones son importantes durante las primeras etapas del diseño de algoritmos de control. Sin embargo, los estudios de simulación generalmente son incompletos debido a que desprecian aspectos prácticos. Tal es el caso en simulaciones de sistemas de control donde la dinámica del robot no contempla el fenómeno de fricción ni las limitaciones de los actuadores del robot, y usualmente se ignora el ruido de los sensores; de ahí que los resultados de simulación sean de valor limitado. Por otro lado, la validación experimental de algoritmos de control, asegura su éxito potencial en el mundo real de las aplicaciones.

Dentro del contexto nacional, la investigación científica orientada a desarrollar nueva tecnología no evoluciona al ritmo que se necesita, y es claramente perceptible la ausencia de productos tecnológicos que hayan sido concebidos, diseñados y producidos dentro del país, sea con fines industriales o netamente didácticos. Ésta realidad es aún más crítica en el mundo de la robótica, cuya presencia en los laboratorios mecatrónicos es aún tenue, limitada e incompleta, debido a que su estudio alcanza poca profundidad científica y posee como tope

máximo la animación computacional del modelamiento y programación de un robot.[1]

Por otro lado, en el contexto mundial de las interfaces de usuario para sistemas robóticos, existe actualmente la necesidad de desarrollar interfaces que ofrezcan una experiencia más inmersiva y natural entre el usuario y el robot, que liberen al primero del contacto con superficies o del uso de dispositivos específicos, cuyas rutinas de operación son poco flexibles y conocidas de antemano.

Las interfaces que reúnen todos estos requerimientos son las denominadas interfaces naturales, cuya característica principal radica en su capacidad de reconocer al usuario, para que a través de la voz, gestos o movimientos corporales, pueda controlar un determinado dispositivo. Estas interfaces pretenden convertirse en la nueva herramienta que potencie y posibilite las comunicaciones con distintos dispositivos, al extremo de transformarse en un accesorio adicional para diversas actividades industriales, e incluso, para determinadas operaciones médicas.

Bajo estas circunstancias, es evidente la necesidad de dar solución a la problemática planteada a través de la puesta en marcha de una plataforma experimental científica, que supere los inconvenientes de infraestructura robótica, y que contribuya, entre otras cosas, al desarrollo de tecnología nacional, mejore la eficiencia terminal de los programas de ingeniería y posgrado, genere el desarrollo de tesis al nivel de pregrado y posgrado, y consolide los conocimientos de la robótica aplicados a problemas reales. Además, sin duda alguna, el principal valor agregado consiste en fortalecer la formación profesional de recursos humanos, ya que capacita para innovar, desarrollar y adecuar conocimientos y tecnología en beneficio de la sociedad.

1.2.1. Campo, Área y Línea de Investigación

- Campo : Ingeniería
- Área : Ingeniería Mecatrónica
- Línea : Robótica e interfaces de usuario

1.3. Antecedentes

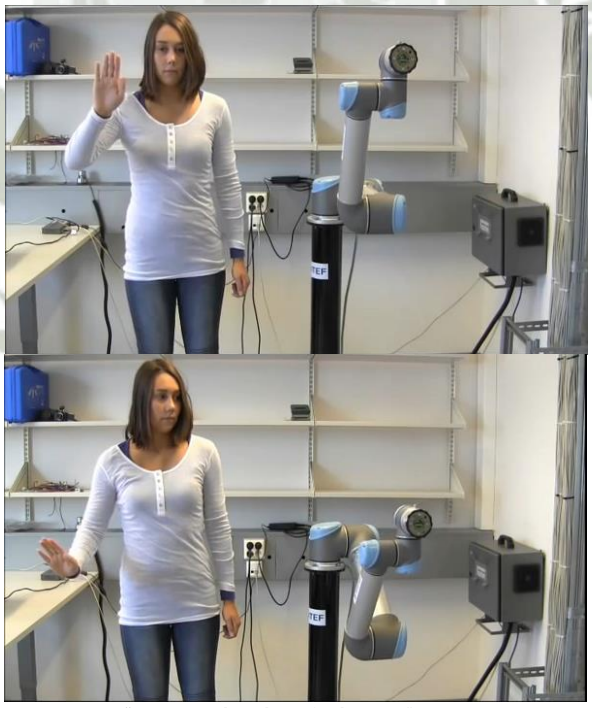
El universo de investigaciones precedentes, que posean una relación directa con el objeto de estudio de la presente investigación es reducido, y se encuentra

[1] Es importante no confundir la animación con simulación, ya que son procesos diferentes: la animación no requiere incorporar efectos dinámicos en el movimiento del robot, lo que impide reproducir todos los fenómenos físicos que un robot real posee y genera durante su operación.

enmarcado exclusivamente en el ámbito internacional. Sin embargo, de ese número reducido de antecedentes, se han rescatado las investigaciones más sólidas y de mayor rigor científico, que orientan la labor académica y constituyen una referencia útil en el ámbito donde fueron desarrolladas.

El primer antecedente científico más cercano a un control gestual utilizando el sensor Kinect se da en Noruega, a través de la experiencia "*Kinect Hand Guiding of Robot Arm*"[2] (*"Guiado manual Kinect de un brazo robótico"*) desarrollada por SINTEF, la mayor organización independiente de investigación en los países escandinavos, como parte de uno de sus proyectos de investigación más ambiciosos llamado *"Next Generation Robotics for Norwegian Industry"* (Fig. 1.1.).

El proyecto es liderado por SINTEF TIC, y cuenta con la cofinanciación del Consejo de Investigación de Noruega y de 8 socios industriales (SINTEF, NTNU, Statoil, Hydro, Tronrud Engineering, Glen Dimplex Nordic, SbSeating y RobotNorge). Su objetivo principal es el desarrollo de una nueva tecnología para la robótica de última generación en la industria noruega. Tiene una duración de 5 años, desde 2009 hasta 2014, y el presupuesto es de 36 millones de coronas noruegas (aproximadamente casi 6 millones de dólares americanos), donde 4 estudiantes de doctorado están siendo educados durante este proyecto.



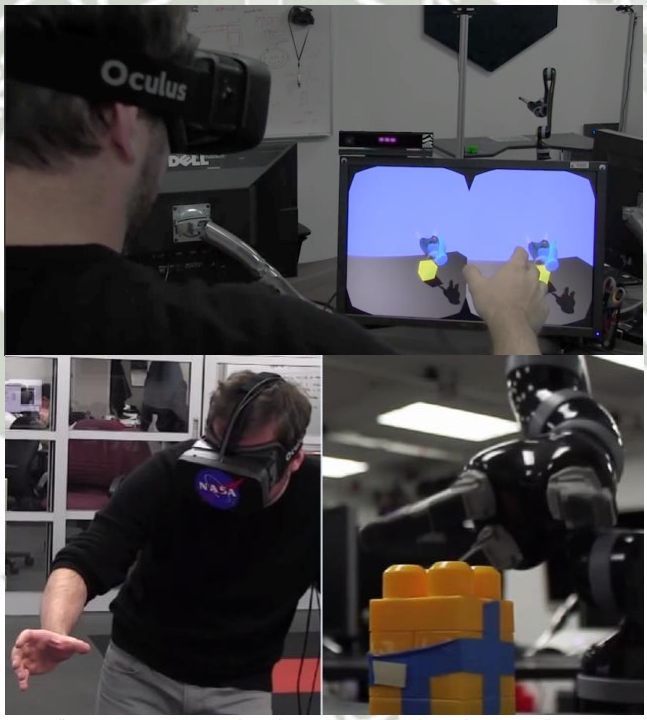
Fuente: "Kinect Hand Guiding of Robot Arm", SINTEF, 2012.

Fig. 1.1. Guiado manual Kinect de un brazo robótico.

[2] En esta experiencia, el usuario se enfrenta a una cámara Kinect que rastrea la posición de su mano. El seguimiento se inicia con un gesto de enfoque. La posición de la mano es actualizada continuamente y enviada al robot, que se mueve a la ubicación indicada.

El segundo antecedente tiene como protagonista a la NASA, cuyos investigadores del Laboratorio de Propulsión Jet están experimentando con robots usando el sensor Kinect de la consola Xbox One y Oculus Rift[3].

El equipo ha estado usando la tecnología de videojuegos de la consola Xbox, para controlar un brazo robótico llamado JACO en tiempo real. Según la NASA, la forma en que funciona es mediante la combinación del seguimiento de posición del Kinect y el seguimiento de rotación del Oculus Rift, para crear una experiencia totalmente inmersiva que permita la visibilidad en primera persona del operador, cuando éste manipula algún objeto de forma remota. Una vez que el operador está establecido, entonces puede realizar tareas simples como recoger pequeños bloques (Fig. 1.2.). Si las pruebas tienen éxito, las futuras aplicaciones podrían incluir el uso de esta tecnología en el humanoide Robonaut 2 de la Estación Espacial Internacional.



Fuente: "NASA Uses Kinect And Oculus Rift To Control A Robotic Arm", IGN[US], 2013.

Fig. 1.2. Operación remota con Kinect de Xbox One y Oculus Rift.

Esta no es la primera vez que la NASA ha utilizado la tecnología de juegos. A principios de 2013, el Laboratorio de Propulsión Jet utilizó el Oculus Rift emparejado con una cinta de correr Omni, para recrear las sensaciones de caminar sobre la superficie de Marte.

[3] Oculus Rift es un *head-mounted display* de realidad virtual que está siendo desarrollado por Oculus VR. Durante su periodo como compañía independiente, Oculus VR ha invertido 91 millones de dólares para el desarrollo de Oculus Rift. Actualmente, la versión para el consumidor está prevista para finales del 2014 o principios del 2015.

1.4. Objetivos

1.4.1. Objetivo principal

Diseñar e implementar una plataforma experimental científica dotada de un control gestual que permita la interacción intuitiva con un brazo robótico de 5 GDL a través del sensor Kinect.

1.4.2. Objetivos secundarios

- Diseñar una estructura robótica robusta, segura, ligera y transportable, cuya cinemática sea solucionable.
- Elaborar un sistema de control que sitúe el cuerpo del usuario como un mando regulador del movimiento del robot.
- Desarrollar una interfaz de usuario natural que permita la interacción hombre-máquina de una manera sencilla, cómoda, rápida e intuitiva, a través de gestos y movimientos corporales.

1.5. Hipótesis

Dado que: El sensor Kinect tiene capacidades de sensado 3D, que permiten sustituir el control a través de "clicks" o pulsaciones táctiles, por aquel basado en gestos y movimientos naturales del cuerpo.

Es posible: Controlar gestualmente un brazo robótico de 5 GDL a través de una interfaz intuitiva, que permita la interacción natural hombre-máquina, sin la necesidad de aprender movimientos específicos, tocar alguna superficie o utilizar mandos determinados.

1.6. Variables

1.6.1. Variable dependiente

Diseño del Control Kinect y la interfaz intuitiva.

1.6.2. Variable independiente

Diseño y modelamiento de un brazo robótico de 5 GDL.

1.7. Justificación de la investigación

El desarrollo de una plataforma experimental que involucre a un robot manipulador dentro de su arquitectura, es objeto de estudio en control automático, y ofrece un amplio espectro para la formulación de problemas de carácter teórico y

práctico; esto se debe a la naturaleza no lineal, acoplada y multivariable de su comportamiento dinámico.

Por otro lado, esta investigación también busca ampliar el campo de acción de las interfaces intuitivas y trasladarlas hacia la ingeniería, demostrando que la utilidad del sensor Kinect no es una exclusividad lúdica, y cuya trascendencia puede marcar un antes y un después, tanto en el diseño de interfaces de usuario como en el control basado en técnicas de visión artificial.

Con el fin de apoyar la investigación sobre los sistemas robóticos, el control gestual se presenta como una herramienta innovadora en la operación remota de brazos robóticos, que permitiría la supervisión y control a distancia. Estas cualidades se tornan muy valiosas, cuando el robot debe operar en situaciones extremas como: manipulación de sustancias tóxicas u objetos explosivos, aplicaciones nocivas como la soldadura, cuyos gases residuales constituyen un potencial agente cancerígeno, casos de emergencia derivados por colisiones o maniobras temerarias, entre otras, donde no es recomendable la presencia del ser humano o su cercanía al robot, pues estos contextos representan un alto riesgo para su salud e integridad física.

Finalmente, el control gestual tiene el poder facultativo de equipar a cualquier brazo robótico, sea didáctico o industrial, con una función de imitación, la cual detecta y almacena los movimientos y gestos humanos, para así llevar a cabo una enseñanza intuitiva que facilite la programación por aprendizaje, lo que permitiría: reducir del tiempo de programación, guiar manualmente el robot en cualquier dirección, grabar fácilmente trayectorias sin conocimientos profundos de programación, interactuar con una interfaz intuitiva para almacenar puntos de localización y establecer parámetros específicos, ajustar el acceso para mantener movimientos de traslación y rotación, y combinar métodos de automatización.

Todas estas ventajas representan la solución a las deficiencias claramente perceptibles de la programación por aprendizaje convencional: su excesivo tiempo de programación, y la complejidad que representa la localización precisa y exacta del efector final del robot, a través de botones accionados desde la paleta de control a una velocidad reducida.

1.8. Alcances y limitaciones

El tema de estudio de la presente investigación, se enfoca en el desarrollo de un sistema de control gestual que permita interactuar con un brazo robótico de 5 GDL, de una manera sencilla, cómoda, rápida e intuitiva, a través de gestos y movimientos corporales captados desde un sensor Kinect.

Para imprimirle un valor agregado al trabajo, se desarrollará el brazo robótico como una creación personal, que servirá no solo de voluntario para realizar la demostración del control planteado, sino también para identificar el campo hacia donde va dirigida la contribución científica, que es el de la robótica. Todo ello sin profundizar en detalles sobre el diseño estructural y modelamiento del mismo, debido a que estos aspectos no corresponden con el tema central de la investigación.

Así, como aporte experimental a la robótica, ésta investigación busca dar solución a la problemática planteada a inicios del capítulo, a través de la creación de una plataforma experimental científica diseñada para realizar investigación, desarrollar aplicaciones de teleoperación y al mismo tiempo, jugar el rol de herramienta indispensable en docencia para robótica y mecatrónica. La plataforma experimental denominada Darko consiste en un robot de 5 GDL, que puede ser controlado gestualmente a través de una interfaz natural de usuario, gracias a la presencia de un sensor Kinect dentro de su equipamiento, el cual permite el reconocimiento de gestos y movimientos del ser humano.

El diseño y la construcción del robot fueron concebidos para fines exclusivamente didácticos y no industriales, y se caracteriza por tener arquitectura abierta[4]. En contraste, los robots industriales tienen arquitectura cerrada, por lo que no permiten programar nuevos esquemas de control y quedan confinados a lo que permita realizar su sistema operativo. Generalmente, un robot comercial no satisface todos los requerimientos de una institución de educación superior, por lo que en poco tiempo resulta una mala inversión y se convierte en una pieza decorativa.

1.9. Metodología de la investigación

La metodología a utilizar en la presente investigación científica, se basa en la vertiente experimental o inductiva del método científico, cuyos pasos se detallan en la Fig. 1.3.

Gracias al método científico y su rigurosidad, los resultados que se obtienen después de realizada una investigación ganan credibilidad. Así, se construyen no solamente conocimientos sólidos, válidos y correctamente fundamentados, sino también se hacen realidad nuevos descubrimientos científicos para el beneficio de toda la humanidad.

[4] Un robot que posee una arquitectura abierta, significa que es posible evaluar experimentalmente las diversas estrategias de control que se diseñe.

Fuente propia.

Fig. 1.3. Pasos del Método Científico.

Capítulo II

Marco Teórico

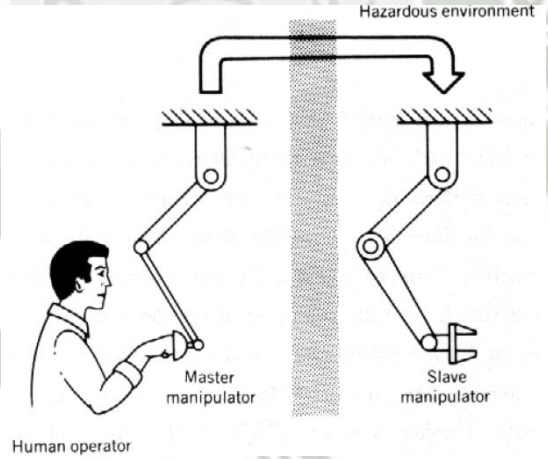
2.1. El sentido de la historia

Muchas definiciones han sido sugeridas para definir lo que es un robot. Ésta palabra puede evocar distintos niveles de sofisticación tecnológica, que van desde un simple dispositivo de manipulación de materiales hasta la máquina antropomórfica avanzada de la ciencia ficción. La imagen de los robots varía ampliamente según los investigadores, ingenieros, fabricantes de robots y países. Sin embargo, es ampliamente aceptado que los robots industriales actuales se originaron con la invención de un dispositivo de manipulación de material programado por George C. Devol. En 1954, Devol presentó una patente en Estados Unidos acerca de una nueva máquina de transferencia de partes, y reivindicó el concepto básico de enseñanza/réplica (métodos de programación en línea) para controlar el dispositivo. Este esquema se utiliza ampliamente en la mayoría de los robots industriales de hoy en día.

Los robots industriales de Devol tienen sus orígenes en dos tecnologías anteriores: el control numérico de máquinas herramientas, y la manipulación remota. El control numérico es un esquema para generar acciones de control sobre la base de los datos almacenados. En los datos almacenados se pueden incluir datos de las coordenadas de los puntos a los que la máquina se va a mover, señales de reloj para iniciar y detener las operaciones, y declaraciones lógicas para realizar ramas secuenciales de control. Toda la secuencia de operaciones y sus variaciones se prescriben y se almacenan en una forma de memoria, de modo que las tareas diferentes pueden ser realizadas sin necesidad de cambios importantes de hardware. Los sistemas de fabricación modernos deben producir una variedad de productos en pequeñas jornadas, en lugar de un gran número de los mismos productos durante un período prolongado de tiempo; los frecuentes cambios de modelos de productos y programas de producción, requieren flexibilidad en el sistema de fabricación. El enfoque de la línea de transferencia, que es más eficaz para la producción en masa, no es apropiado cuando se necesita tal flexibilidad. Cuando se requieren cambios en los productos, una línea de producción para fines especiales se vuelve inútil, y a menudo termina siendo abandonada, a pesar de la gran inversión de capital que implicó originalmente. La automatización flexible ha sido un tema central en la innovación de fabricación de hace unas pocas décadas, y

el control numérico ha jugado un papel central en el aumento de la flexibilidad del sistema. Los robots industriales contemporáneos son máquinas programables que pueden realizar diferentes operaciones con tan solo modificar los datos almacenados, una característica que ha evolucionado a partir de la aplicación del control numérico.

Otro origen de los robots industriales de hoy en día se puede encontrar en manipuladores remotos. Un manipulador remoto es un dispositivo que realiza una tarea a distancia. Se puede utilizar en ambientes donde los trabajadores humanos no pueden acceder con facilidad o con seguridad, por ejemplo, para la manipulación de materiales radiactivos, o en algunas aplicaciones marinas y espaciales. El primer sistema manipulador maestro-esclavo fue desarrollado en 1948. El concepto implica un brazo mecánico eléctrico instalado en el lugar de operación, y una palanca de mando de control de geometría similar a la del brazo mecánico (Fig. 2.1.). La palanca de mando tiene transductores de posición en las articulaciones individuales, que miden el movimiento del operador humano mientras se mueve la palanca de mando. Por lo tanto, el movimiento del operador se transforma en señales eléctricas, que son transmitidas al brazo mecánico y hacen el mismo movimiento que realiza el operador humano.



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 2.1. Manipulador maestro-esclavo.

La palanca de mando que maneja el operador se llama el maestro de la manipulación, mientras que el brazo mecánico se llama el manipulador esclavo, ya que su movimiento es idealmente la réplica de movimiento ordenado del operador. Un manipulador maestro-esclavo tiene normalmente seis grados de libertad para permitir que la pinza pueda localizar un objeto en una posición y orientación arbitraria. La mayoría de las articulaciones son de revolución, y toda la construcción mecánica es similar a la del brazo humano. Esta analogía con el brazo humano resulta de la necesidad de replicar los movimientos humanos. Además, esta estructura permite movimientos diestros en una amplia gama de espacio de

trabajo, lo cual es deseable para las operaciones en los sistemas de fabricación modernos.

Los robots industriales contemporáneos conservan cierta similitud en la geometría tanto con el brazo humano como con los manipuladores remotos. Además, sus conceptos básicos han evolucionado a partir de los del control numérico y la manipulación a distancia. Por lo tanto, una definición ampliamente aceptada de robot industrial hoy en día, es el de un manipulador de control numérico, donde el operador humano y el maestro de la manipulación se sustituyen por un controlador numérico.

La fusión del control numérico y la manipulación remota crea un nuevo campo de la ingeniería que se puede denominar como la robótica, y con ella una serie de temas científicos en el diseño y control, que son sustancialmente diferentes de los de las tecnologías originales.

2.2. Definición del Robot

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

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

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

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

Una definición que complementa a las anteriores, es la establecida por la Asociación Francesa de Normalización (AFNOR) que define primero el manipulador y, basándose en ella, el robot:

"Un manipulador es un mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado a la captura y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o mediante dispositivo lógico".

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

2.3. Clasificación de los Robots

Un robot puede ser clasificado atendiendo a diferentes criterios y características. Algunas de éstas serán dependientes de su propia esencia, otras de la aplicación o tarea a que se destinan. A continuación se exponen aquellas relacionadas al prototipo en estudio.

2.3.1. Clasificación según la Generación

La generación de un robot hace referencia al momento tecnológico en que éste aparece. De este modo se puede considerar que se pasa de una generación a la siguiente cuando se da un hito que supone un avance significativo en las capacidades de los robots.

Aun siendo ésta una división subjetiva, es interesante, pues permite hacerse una idea de cuán avanzado es un robot. La Tabla 2.1 recoge una posible clasificación en generaciones. Cronológicamente podría decirse que la primera generación se extiende desde el comienzo de la robótica hasta los años ochenta. La segunda generación se desarrolla en los años ochenta y es la que mayoritariamente se puede encontrar hoy en día en las industrias. La tercera generación está desarrollándose en estos días, siendo, por tanto, objeto de un futuro cercano.

Tabla 2.1. Clasificación de los robots según generaciones.

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

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

2.3.2. Clasificación según el Área de Aplicación

Desde el punto de vista del uso que se da al robot, es posible clasificarlos bien en base al sector económico en el que se encuentran trabajando o bien en base al tipo de aplicación o tarea que desarrollan, independientemente de en qué sector económico trabajen. La clasificación más práctica es la que hace la IFR (*International Federation of Robotics*) en base al tipo de aplicación. En ésta se detallan las aplicaciones más frecuentes a las que se dedican los robots en el ámbito industrial, profesional e investigativo.

Tabla 2.2. Clasificación de las aplicaciones de los robots industriales según 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: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

2.3.3. Clasificación según el Número de Ejes

Esta característica es aplicable a los robots o telerobot con cadena cinemática (es decir, sería aplicable a los robots manipuladores, pero no lo sería, por ejemplo, a los robots móviles).

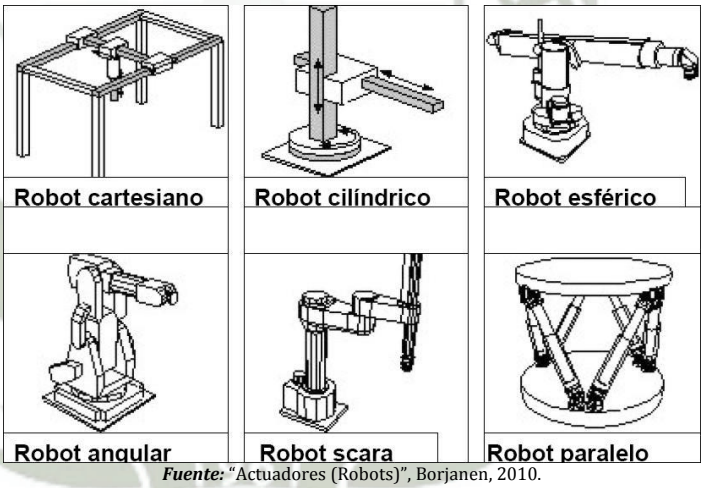
Se entiende por un eje cada uno de los movimientos independientes con que está dotado del robot. Puesto que de acuerdo a la definición ISO el robot manipulador industrial debe tener al menos 3 ejes y extendiendo esta condición a los robots de servicio manipuladores, se podrán encontrar robots de cualquier número de ejes superior o igual a 3.

En la práctica, la mayor parte de los robots tienen 6 ejes, seguidos por los de 4. Los robots con más de seis ejes son poco frecuentes, estando justificado este número para aumentar la capacidad de maniobra del robot, y siendo en muchas ocasiones telerobots.

2.3.4. Clasificación según la Configuración

Como ocurre en el caso anterior, esta clasificación es sólo aplicable a robots o telerobots con cadena cinemática.

- **Cartesiano:** cuyo posicionamiento en el espacio se lleva a cabo mediante articulaciones lineales.
- **Cilíndrico:** con una articulación rotacional sobre una base y articulaciones lineales para el movimiento en altura y en radio.
- **Polar o esférico:** que cuenta con dos articulaciones rotacionales y una lineal.
- **Articular o antropomórfico:** 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 concurrentes.



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

Fig. 2.2. Configuraciones de robots industriales.

2.3.5. Clasificación según el Tipo de Control

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

- **Robot secuencial:** *Robot con un sistema de control en el que un conjunto de movimientos se efectúa eje a eje en un orden dado, de tal forma que la finalización de un movimiento inicia el siguiente.*

  En este tipo de robots, sólo es posible controlar una serie de puntos de parada, resultando un movimiento punto a punto. Un ejemplo de ellos son los manipuladores neumáticos.

- **Robot controlado por trayectoria:** *Robot que ejecuta un procedimiento controlado por el cual los movimientos de tres o más ejes controlados, se desarrollan según instrucciones que especifican en el tiempo la trayectoria requerida para alcanzar la siguiente posición (obtenida normalmente por interpolación).*

  Los robots controlados por trayectoria permiten la realización de movimientos en los que puede ser especificado toda la trayectoria de manera continua.

- **Robot adaptativo:** *Robot que tiene funciones de control con sensores, control adaptativo o funciones de control de aprendizaje.*

  De este modo el robot puede modificar su tarea de acuerdo a la información captada del entorno, por ejemplo, a través de un sistema de visión por computador o por sensores de fuerza o contacto.

- **Robot teleoperado:** *Un robot que puede ser controlado remotamente por un operador humano, extendiendo las capacidades sensoriales y motoras de éste a localizaciones remotas.*

2.4. Morfología del robot

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de potencia y control, y elementos terminales.

En esta sección se examinan estos elementos constitutivos de un robot. Se estudia primero la estructura mecánica, haciendo referencia a los elementos que constituyen un robot y a los distintos tipos de articulaciones posibles entre los eslabones consecutivos. Seguidamente, se enumeran los principales sistemas de transmisión, necesarios para transmitir a cada una de las articulaciones el movimiento generado por los actuadores, y se analiza el caso particular de la transmisión directa.

Se repasan después los denominados sensores internos, necesarios para proporcionar al sistema que controla los movimientos del robot, información relativa a la localización del mismo. Finalmente, el último apartado se ocupa de los elementos terminales (piezas, herramientas, dispositivos de sujeción, etc.), que situados generalmente en el extremo del robot, sirven para que éste pueda interaccionar con el mundo exterior, realizando las tareas que le han sido asignadas.

Aunque los elementos empleados en los robots no son exclusivos de estos (máquinas herramientas y otras muchas máquinas emplean tecnologías semejantes), las altas prestaciones que se exigen a los robots han motivado que, en algunos casos, se empleen en ellos elementos con características específicas.

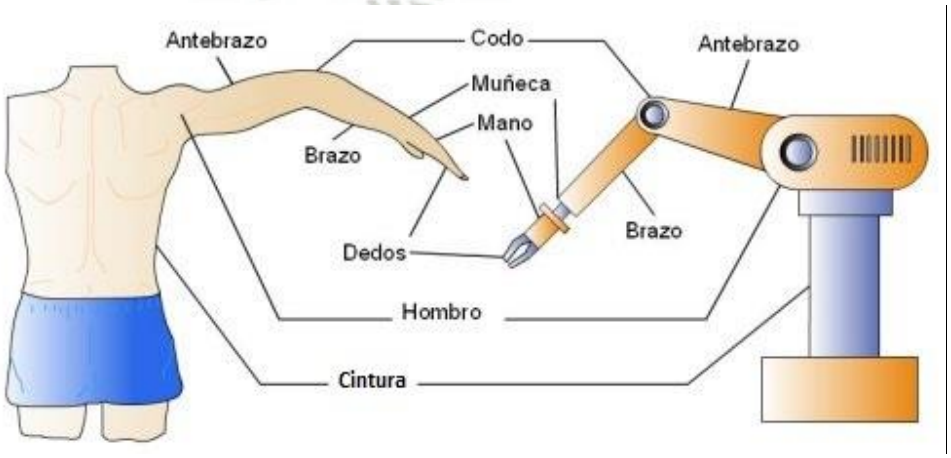
2.4.1. Estructura Mecánica de un robot

Mecánicamente, un manipulador robótico consta de una secuencia de elementos estructurales rígidos, denominados enlaces o eslabones, conectados entre sí mediante juntas o articulaciones, que permiten el movimiento relativo entre cada dos eslabones consecutivos.

Fuente: "Control y Robótica – Curso en Línea", Víctor R. González.

Fig. 2.3. Elementos estructurales de un robot industrial.

La constitución física de la mayor parte de los robots manipuladores industriales, guarda cierta similitud con la anatomía del brazo humano (es por eso que se les suele denominar también brazos robóticos), por lo que en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cintura, hombro, brazo, codo y muñeca.

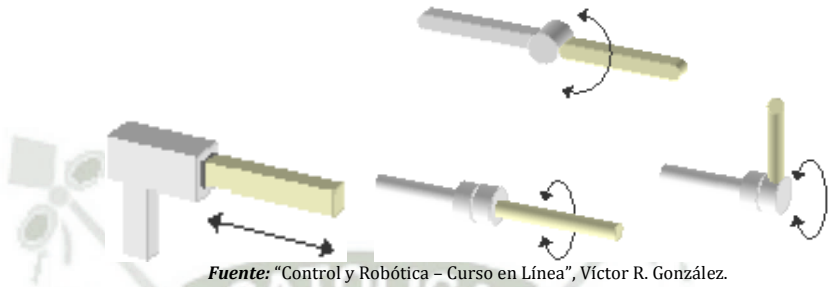


Fuente: "Control y Robótica", Antonio Bueno.

Fig. 2.4. Semejanza de un manipulador con la anatomía humana.

Una articulación puede ser:

- **Lineal** (traslacional o prismática), si un eslabón desliza sobre un eje solidario al eslabón anterior.
- **Rotacional,** en caso de que un eslabón gire en torno a un eje solidario al eslabón anterior.



Fuente: "Control y Robótica – Curso en Línea", Víctor R. González.

a) **b)**

Fig. 2.5. Distintos tipos de articulaciones de un robot: a) lineal, b) rotacionales.

Se denomina grado de libertad (GDL) a cada una de las coordenadas independientes que son necesarias para describir el estado del sistema mecánico del robot (posición y orientación en el espacio de sus elementos). Normalmente, en cadenas cinemáticas abiertas, cada par eslabón-articulación tiene un solo grado de libertad, ya sea de rotación o de traslación, pero una articulación podría tener dos o más GDL que operan sobre ejes que se cortan entre sí.

Fuente: "Control y Robótica – Curso en Línea", Víctor R. González.

Fig. 2.6. Distintos grados de libertad de un brazo robótico.

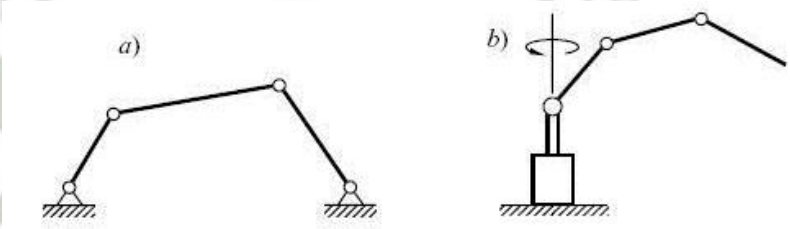
El número de grados de libertad de una cadena cinemática puede ser obtenido mediante la fórmula de Grübler, según la cual:

$$\#\ GDL = \lambda(n - j - 1) + \sum_{i=1}^{j} f_i \qquad [2.1]$$

Donde:

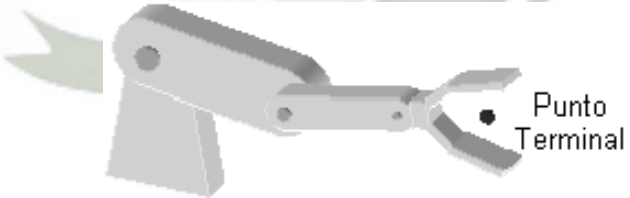
$\boldsymbol{\lambda}$: GDL del espacio de trabajo (Típicamente tres en el plano, seis en el espacio).
$\boldsymbol{n}$: Número de eslabones (debe incluirse el esclavo fijo o base).
$\boldsymbol{j}$: Número de articulaciones.
$\boldsymbol{f_i}$: Grados de libertad permitidos a la articulación *i*.

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



Fuente: "Máquinas y mecanismos", SiteNorDeste, 2014.
Fig. 2.7. Tipos de Cadenas Cinemáticas: a) Cerrada, b) Abierta.

Al extremo final se puede conectar un elemento terminal o actuador final, que es una herramienta especial, que permite al robot realizar una aplicación particular, la cual debe diseñarse específicamente para dicha aplicación: una herramienta de sujeción, de soldadura, de pintura, etc. El punto más significativo del elemento terminal se denomina punto terminal (PT). En el caso de una pinza, el punto terminal es el centro de sujeción de la misma.



Fuente: "Control y Robótica – Curso en Línea", Víctor R. González.
Fig. 2.8. Punto terminal de un manipulador.

Para describir y controlar el estado de un brazo de robot es preciso determinar:

- La posición del punto terminal (o de cualquier otro punto) respecto de un sistema de coordenadas externo y fijo, denominado el sistema mundo.
- El movimiento del brazo cuando los elementos actuadores aplican sus fuerzas y momentos.

El análisis desde el punto de vista mecánico de un robot se puede efectuar atendiendo exclusivamente a sus movimientos (estudio cinemático) o atendiendo además a las fuerzas y momentos que actúan sobre sus partes (estudio dinámico) debidas a los elementos actuadores y a la carga transportada por el elemento terminal.

2.4.2. Transmisiones

Las transmisiones son los elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones. Generalmente se incluyen, junto a las transmisiones, a los reductores, encargados de adaptar el par y la velocidad de la salida del actuador, a los valores adecuados para el movimiento de los elementos del robot.

Es de esperar que un buen sistema de transmisión cumpla una serie de características básicas: debe tener un tamaño y peso reducido, se ha de evitar que presente juegos u holguras considerables y se deben buscar transmisiones con gran rendimiento.

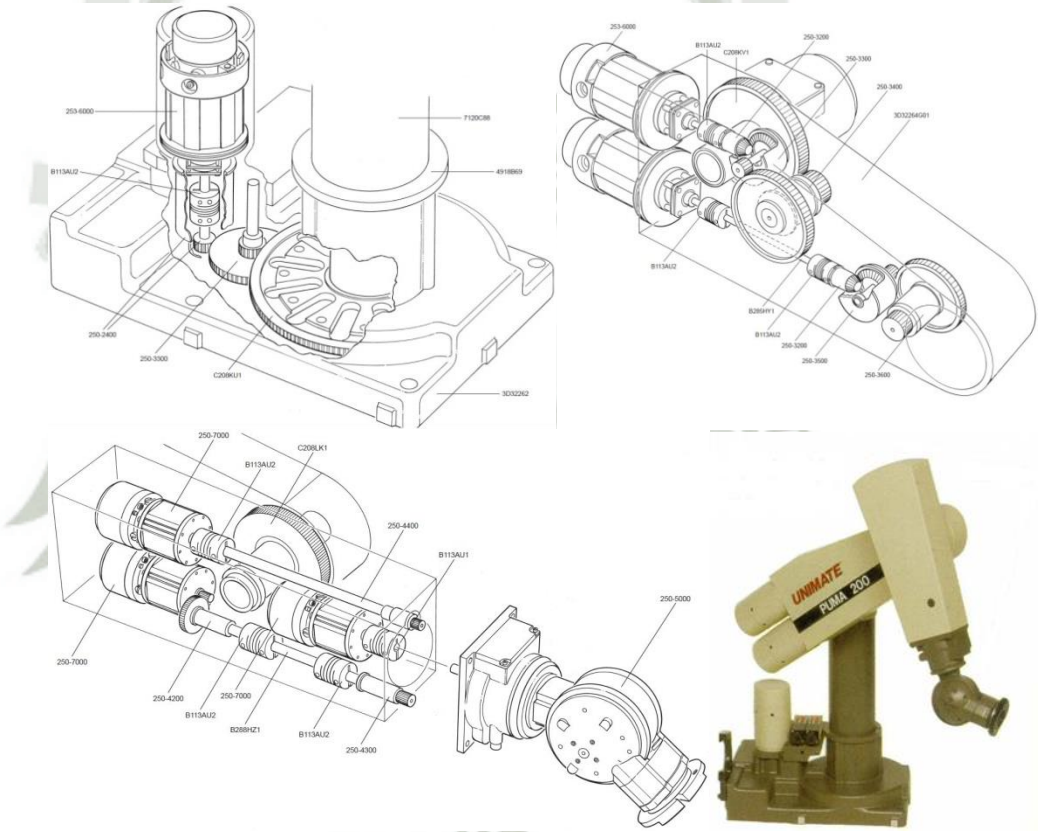
Aunque no existe un sistema de transmisión específico para robots, sí existen algunos usados con mayor frecuencia y que se recogen clasificados en la Tabla 2.3. La clasificación se ha realizado en base al tipo de movimiento posible en la entrada y salida.: lineal o circular. En la citada tabla también quedan reflejados algunas ventajas e inconvenientes propios de algunos sistemas de transmisión. Entre ellos cabe destacar la holgura y el juego.

Tabla 2.3. Sistemas de transmisión para robots.

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

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Es muy importante que el sistema de transmisión a utilizar no afecte al movimiento que transmite, ya sea por el rozamiento inherente a su funcionamiento o por las holguras que su desgaste pueda introducir. También se debe tener en cuenta que el sistema de transmisión, sea capaz de soportar un funcionamiento continuo a un par elevado, y a ser posible, entre grandes distancias. Las transmisiones más habituales son aquellas que cuentan con movimiento circular tanto a la entrada como a la salida. Incluidas en estas se hallan los engranajes, las correas de entradas y las cadenas.



Fuente: "PUMA 200 Robot Parts", RP Automation, 2011.

Fig. 2.9. Sistema de engranajes del robot PUMA 200.

Transmisión Directa

Desde hace unos años, existen en el mercado robots que poseen otro tipo de tecnología denominada transmisión directa (*Direct Drive*), la cual constituye la nueva generación de robots cuya tecnología irá desplazando paulatinamente a los robots tradicionales. Transmisión directa o accionamiento directo significa, que el eje del actuador se conecta directamente a la carga o articulación, sin la utilización de un reductor

intermedio[5], eliminando el sistema de engranaje tradicional. Este concepto fue establecido por Haruiko Asada en 1980.

La tecnología de transmisión directa elimina el cascabeleo o falta de movimiento y reduce significativamente el fenómeno de fricción comparada con los robots convencionales; la fricción no se elimina completamente, su magnitud se reduce debido a que el motor no tiene escobillas (*brushless*). Por otra parte, los materiales de construcción hacen que el rotor y el estator se encuentren levitando entre sí. Además, la construcción mecánica del robot es mucho más simple y la exactitud en el posicionamiento del extremo final del robot es mejorada. Una característica importante de este tipo de tecnología es que la electrónica asociada del motor lo hace funcionar como fuente ideal de par aplicado, lo cual significa que independientemente de la carga mecánica, mantiene constante el par solicitado en cada periodo de muestreo. Sin embargo, en la práctica está restringido por los límites físicos en los servo actuadores.

Este tipo de tecnología aparece a raíz de la necesidad de utilizar robots en aplicaciones que exigen combinar gran precisión con alta velocidad. Los reductores introducen una serie de efectos negativos, como son juego angular, rozamiento o disminución de la rigidez del accionador, que pueden impedir alcanzar los valores de precisión y velocidad requeridos.

Las principales ventajas que se derivan de la utilización de accionamientos directos son las siguientes:

- Posicionamiento rápido y preciso, pues se evitan los rozamientos y juegos de las transmisiones y reductores.
- Aumento de las posibilidades de controlabilidad del sistema a costa de una mayor complejidad.
- Simplificación del sistema mecánico al eliminarse el reductor.

El principal problema que existe para la aplicación práctica de un accionamientos directo radicada en el motor a emplear. Debe tratarse de motores que proporcionen un par elevado (unas 50-100 veces mayor que con reductor) a bajas revoluciones (más de movimiento de la articulación) manteniendo la máxima rigidez posible.

Entre los motores empleados para accionamientos directo y que cumplan estas características, se encuentran los motores síncronos y de continua sin

[5] Montano, L., Tardós, J. D., y Saghés, C., "Accionamientos directos para robots", *Automática e Instrumentación*, 179, pp. 251-261, 1988.

escobillas (*brushless*), ambos con imanes permanentes fabricados con materiales especiales (samario-cobalto).

También se utilizan motores de inducción de reluctancia variable y servomotores. La necesaria utilización de este tipo de motores encarece notablemente el sistema de accionamiento.

Otra cuestión importante a tener en cuenta en el empleo de accionamientos directos, es la propia cinemática del robot. Colocar motores, generalmente pesados y voluminosos, junto a las articulaciones, no es factible para todas las configuraciones del robot debido a los pares que se generan. El estudio de la cinemática con la que se diseña el robot ha de tener en cuenta estos parámetros, estando la estructura final elegida altamente condicionada por ellos. Al eliminar un reductor, también se disminuye de forma considerable la resolución real del codificador de posición acoplado al eje. Esto lleva a la utilización, en este tipo de robots, de codificadores de posición de muy alta resolución.

El primer robot comercial con accionamiento directo se presentó en 1984. Se trataba de un robot SCARA denominado AdeptOne, de la compañía norteamericana *Adept Technology*. A partir de entonces, este tipo de robots se ha hecho popular para aplicaciones que requieran robots con altas prestaciones en velocidad y posicionamiento (montaje microelectrónico, corte de metal por láser, etc.). La Fig. 2.10. muestra una fotografía del robot AdeptOne. En la actualidad, un robot con accionamiento directo puede llegar a aumentar tanto la velocidad como la precisión de manera significativa con respecto a aquellos robots de accionamiento tradicional.

Fuente: "Robot History", Robot System, 2012.

Fig. 2.10. Robot AdeptOne de accionamiento directo.

En la Fig. 2.11. se muestra un robot prototipo de transmisión directa de 3 GDL, diseñado y construido de manera específica para realizar investigación científica en robótica. Dicho prototipo fue construido en 1998 en el Laboratorio de Robótica de la Facultad de Ciencias de la Electrónica, de la Benemérita Universidad Autónoma de Puebla (BUAP).

Fuente: "Robot Rotradi", BUAP, 2009.

Fig. 2.11. Robot de transmisión directa de la BUAP.

2.4.3. Actuadores

Los actuadores tienen por misión generar el movimiento de los elementos del robot según las órdenes dadas por la unidad de control. Los primeros robots industriales utilizaban actuadores hidráulicos para mover sus elementos. Las ventajas prácticas del uso de la electricidad como fuente de energía, han motivado que en la actualidad, posiblemente la totalidad de los robots industriales existentes utilizan esta opción. No obstante, en determinadas situaciones deben ser considerados otros tipos de actuadores. De manera general, los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica. Cada uno de estos sistemas presenta características diferentes, de entre las que se pueden considerar las siguientes: potencia, controlabilidad, peso y volumen, precisión, velocidad, mantenimiento y coste.

En este apartado se revisa brevemente un tipo particular de actuadores eléctricos: los servomotores, y se les compara en cuanto a las características anteriores. Un estudio más detallado del resto de actuadores eléctricos y los de género neumático e hidráulico, puede encontrarse en bibliografía específica.

Servomotores

Los servomotores son sistemas electromecánicos que forman las uniones o articulaciones del robot mediante las cuales éste se mueve, motivo por el cual al movimiento del robot se le denomina desplazamiento articular. La tecnología asociada a los servomotores es complicada, ya que su electrónica está basada en microprocesadores avanzados de procesamiento como los DSP's (*Digital Signal Processors*) para llevar a cabo el análisis y control de las diversas variables que determinan su modo de operación.

Un servomotor está compuesto principalmente por tres elementos: motor eléctrico, sensor de posición para medir el desplazamiento articular (rotacional o lineal) y el amplificador electrónico (*electronic driver*) o servo amplificador constituido por un conjunto que microprocesadores y electrónica de potencia, que se encarga de acoplar y acondicionar al motor la impedancia y señal de voltaje de baja potencia que provienen de la computadora o de un sistema mínimo digital. Evidentemente, un motor requiere mucha mayor potencia para su funcionamiento.

Los servomotores tienen tres modos de operación: posición, velocidad y par. El modo posición permite mover al motor a una posición preestablecida, también conocida como *set point*. Sin embargo, no puede desplazar cargas o aplicar una fuerza determinada. En este modo se emplean reguladores simples como el proporcional derivativo (PD) y proporcional integral derivativo (PID). El modo velocidad controla el movimiento del motor sobre una velocidad deseada. Tampoco puede ejercer fuerza como en el caso del modo de posición. Estos modos tienen la característica de ser arquitectura cerrada, es decir, no permiten programar otro tipo de controladores, por lo que sus aplicaciones en robótica se encuentran limitadas.

El modo par es el que se emplea en robótica y su principal característica es la arquitectura abierta, lo que hace posible evaluar experimentalmente el desempeño y robustez de cualquier estrategia de control. De esta forma, las aplicaciones en robótica se incrementan. Asimismo, permite la interacción dinámica con el sistema mecánico del robot, por lo que es posible compensar los efectos dinámicos del robot y en consecuencia controlar la posición o desplazamiento articular. El modo par también permite el control de la trayectoria y aplicaciones más complicadas como el control de impedancia, *visual-servoing*, teleoperación y control de fuerza.

Las ecuaciones de Maxwell permiten deducir una ley fundamental para servomotores que determina la relación entre el campo electromagnético y el par (torque) aplicado. En general, la generación de movimiento rotacional con respecto a su eje de giro es producido por el par aplicado, esta simple ley es la base del desarrollo de la robótica y está planteada en la siguiente ecuación:

$$\tau = kv \qquad [2.2]$$

En donde τ representa el par aplicado al servomotor y sus unidades de medición son N.m, k es una constante que representa la ganancia del amplificador electrónico y tiene unidades de N.m/V, y la variable v es el voltaje que proviene de la computadora y representa el comando o ley de control que programa el usuario para que el robot lleve a cabo una aplicación específica. No todos los motores cumplen con la ecuación, como son los casos de algunos motores de corriente alterna y motores de pasos, y por esta razón no forman parte de la estructura del sistema mecánico del robot. Pueden ser empleados en la construcción de garras mecánicas o de herramientas específicas que se colocan en el robot para diversas aplicaciones.

Fuente: "Electromechanical Positioning Systems", Parkermotion, 2005.
Fig. 2.12. Servomotores de accionamiento directo.

2.4.4. Sensores internos

Para conseguir que un robot realice una tarea con adecuada precisión, velocidad e inteligencia, será preciso que tenga conocimiento tanto de su propio estado como del estado de su entorno. La información relacionada con su estado (fundamentalmente la posición de sus articulaciones) la consigue con los denominados sensores internos, mientras que la que se refiere al estado de su entorno, se adquiere con los sensores externos. En este epígrafe se tratará únicamente, a modo de resumen, los sensores internos. Un estudio más detallado de estos y de los sensores externos puede encontrarse en bibliografía específica.

La información que la unidad de control del robot puede obtener sobre el estado de su estructura mecánica es, fundamentalmente, la relativa a su posición y velocidad. En la Tabla 2.4 se resumen los sensores más comúnmente empleados para obtener información de presencia, posición y velocidad en robots industriales.

Tabla 2.4. Tipos de sensores propioceptivos para robot.

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

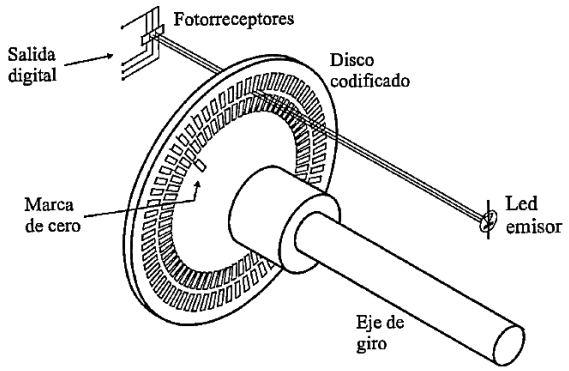
Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Sensores de Posición

Para el control de posición angular se emplean, fundamentalmente, los denominados encoders y resolvers. Los potenciómetros dan bajas prestaciones, por lo que no se emplean salvo en contadas ocasiones (robots didácticos, ejes de poca importancia)

Codificadores angulares de posición (encoders)

Los codificadores ópticos o encoders incrementales constan, en su forma más simple, de un disco transparente con una serie de marcas opacas colocadas radialmente y equidistantes entre sí; de un sistema de iluminación en el que la luz es colimada de forma correcta, y de un elemento fotorreceptor (Fig. 2.13.). El eje, cuya posición se desea medir, va acoplado al disco transparente. Con esta disposición, a medida que el eje gire se irán generando pulsos en el receptor cada vez que la luz atraviese cada marca. Llevando la cuenta de estos pulsos, será posible conocer la posición del eje.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

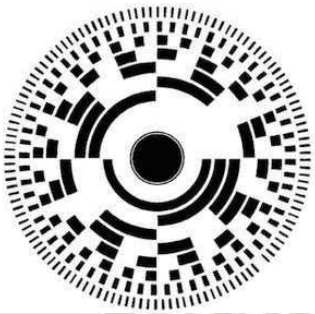
Fig. 2.13. Disposición de un codificador óptico (encoder) incremental.

Los pulsos que se originan como consecuencia del giro, no informan por si mismos del sentido de éste. Al objeto de poder conocer esta información, se dispone de una segunda franja de marcas, desplazada de la anterior, de manera que el tren de pulsos que con ella se genere, esté desfasado 90° con respecto al generado por la primera franja. De esta manera, mediante el uso de un biestable del tipo D, en el que la salida mantiene el valor de la entrada leída durante el flanco positivo del reloj, es posible obtener una señal adicional que indique cuál es el sentido de giro, y que actúe sobre el contador correspondiente indicando que incremente o decremente la cuenta que se esté realizando. Dado que el contador es incremental, es preciso establecer cuál es el cero, para lo cual se incorpora una marca de referencia adicional única en todo el trazado del disco, que indique que se ha dado una vuelta completa y que, por tanto, se ha de reinicializar el contador.

La resolución de este tipo de sensores depende directamente del número de marcas que se pueden poner físicamente en el disco. Un método relativamente sencillo para aumentar esta resolución es, no solamente contabilizar los flancos de subida de los trenes de pulsos, sino contabilizar también los de bajada, resultando que el captador sea capaz de distinguir un número de posiciones igual a cuatro veces el número de marcas por canal. Se puede llegar de esta manera valores de 100 000 pulsos por vuelta. Este valor puede ser todavía aumentado mediante el uso de filtros que modulan el nivel de luz que llega al receptor, lo que, tras una umbralización del mismo, permiten distinguir un número de posiciones muy elevado.

El funcionamiento básico de los codificadores o encoders absolutos es similar al de los incrementales. Se tiene una fuente de luz con las fuentes de adaptación correspondientes, un disco graduado y unos fotorreceptores. En este caso, el disco transparente se divide en un número determinado de sectores (potencia de 2), codificándose cada uno de ellos según un código

binario cíclico (normalmente código Gray) que queda representado por zonas transparentes y opacas dispuestas radialmente (Fig. 2.14.). No es necesario en este caso, ningún contador o electrónica adicional para detectar el sentido de giro, pues cada posición (sector) es codificada de forma absoluta. Su resolución es fija, y vendrá dada por el número de anillos que posea el disco graduado. Resoluciones habituales van desde 2^8 a 2^{19} bits (desde 256 a 524 288 posiciones distintas).



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.14. Franjas en un captador óptico absoluto.

Normalmente, los sensores de posición se acoplan al eje del motor. Considerando que la mayor parte de los casos, entre el eje del motor y el de la articulación se sitúa un reductor de relación N, cada movimiento de la articulación se verá multiplicado por N al ser medido por el sensor. Éste aumenta así su resolución multiplicándola por N.

En el caso de los codificadores absolutos, este efecto origina que una misma lectura del captador, pueda corresponder a N posiciones distintas de la articulación. Para diferenciarlas, se puede utilizar un encoder adicional, conectado por un engranaje reductor al principal, de manera que cuando éste gire una vuelta completa, el codificado adicional avanza una posición. El conjunto se denomina encoders absolutos multivuelta.

Esta misma circunstancia (presencia del reductor) originará que en el caso de los codificadores incrementales, la señal de referencia o marca de cero sea insuficiente para detectar el punto origen para la cuenta de pulsos, pues habrá N posibles puntos de referencia para un giro completo de la articulación. Para distinguir cuál de ellos es el correcto, se suele utilizar un detector de presencia denominado de sincronismo, acoplado directamente al eslabón del robot que se considere. Cuando se conecta al robot desde una situación de apagado, es preciso ejecutar un procedimiento de búsqueda de referencias para los sensores (sincronizado). Durante su ejecución, se leen los detectores de sincronismo que detectan la presencia o ausencia del eslabón del robot. Cuando se detecta la conmutación de presencia o

ausencia de pieza, o viceversa, se atiende al encoder incremental, tomándose como position de origen la correspondiente al primer pulso de marca de cero que aquél genere.

Los encoders pueden presentar problemas mecánicos debido a la gran precisión que se debe tener en su fabricación. La contaminación ambiental puede ser una fuente de interferencias en la transmisión óptica. Son dispositivos particularmente sensibles a golpes y vibraciones, estando su margen de temperatura de trabajo limitado por la presencia de componentes electrónicos.

Sensores de Velocidad

La captación de la velocidad se hace necesaria para mejorar el comportamiento dinámico de los actuadores del robot. La velocidad de movimiento de cada actuador (que tras el reductor es la velocidad de la articulación) se realimenta normalmente a un bucle de control analógico implementado en el propio accionador el elemento motor. No obstante, en ocasiones en las que el sistema de control del robot lo exija, la velocidad de giro de cada actuador es llevada hasta la unidad de control del robot.

Normalmente, y puesto que el bucle de control de velocidad es analógico, el captador usado es una tacogeneratriz que proporciona una tensión proporcional a la velocidad de giro de su eje (valores típicos pueden ser 10 mV por rpm). Otra posibilidad, usada para el caso de que la unidad de control del robot precise valorar la velocidad de giro de las articulaciones, consiste en derivar la información de posición que ésta posee.

Sensores de Presencia

Este tipo de sensor es capaz de detectar la presencia de un objeto dentro de un radio de acción determinado. Esta detección puede hacerse con o sin contacto con el objeto. En el segundo caso, se utilizan diferentes principios físicos para detectar la presencia, dando lugar a los diferentes tipos de captadores (véase Tabla 2.4.). En el caso de detección con contacto, se trata siempre de un interruptor, normalmente abierto o normalmente cerrado según interese, actuado mecánicamente a través de un vástago u otro dispositivo.

Los detectores de presencia se utilizan en robótica principalmente como auxiliares de los detectores de posición, para indicar los límites de movimiento de las articulaciones y permiten localizar la posición de referencia de cero de éstos, en el caso de que sean incrementales.

Además de esta aplicación, los sensores de presencia se usan como sensores externos, siendo muy sencillos de incorporar al robot por su carácter binario y su costo reducido. Los detectores inductivos permiten detectar la presencia o contar el número de objetos metálicos sin necesidad de contacto. Presentan el inconveniente de distinto comportamiento según el tipo de metal del que se trate. El mismo tipo de aplicación tiene los detectores capacitivos, más voluminosos, aunque en este caso los objetos a detectar no precisan ser metálicos. En cambio, presentan problemas de trabajo en condiciones húmedas y con puestas a tierra defectuosas. Los sensores basados en el efecto Hall detectan la presencia de objetos ferromagnéticos, por la deformación que éstos provocan sobre un campo magnético. Los captadores ópticos, sin embargo, pueden detectar la reflexión del rayo de luz procedente del emisor sobre el objeto.

2.4.5. Elementos terminales

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

Elementos de sujeción

Los elementos de sujeción se pueden clasificar según el sistema de sujeción empleado. En la Tabla 2.5. se representan estas opciones, así como los usos más frecuentes.

Tabla 2.5. Sistemas de sujeción para robots.

Tipos de sujeción	Accionamiento	Uso
Pinzas de presión de desplazamiento angular o lineal	Neumático o eléctrico	Transporte y manipulación de piezas sobre las que no importa presionar.
Pinza de enganche	Neumático o eléctrico	Piezas de grandes dimensiones o sobre las que no se puede ejercer presión.
Ventosas de vacío	Neumático	Cuerpos con superficie lisa poco porosa (cristal, plástico, etc.)
Electroimán	Eléctrico	Piezas ferromagnéticas

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Los elementos de sujeción más comunes son las denominadas pinzas o garras. Habitualmente, utilizan accionamiento neumático para sujetar las

piezas por presión (Fig. 2.15.). En el mercado se pueden encontrar pinzas neumáticas de diversas características, debiéndose atender en la selección al tipo de movimientos de los dedos (linear o angular), el recorrido de éstos, la fuerza que ejercen, el número de dedos (por lo general 2 o 3), si se precisa ejercer fuerza tanto en apertura como en cierre y al tiempo de respuesta.

En el cálculo de la fuerza de agarre, debe considerarse no sólo el peso de la pieza a transportar, sino también su forma; el material de que está hecha, que afectará al valor de la fuerza de rozamiento con la superficie de los dedos de la pinza; y las aceleraciones con que se pretende mover a la pieza.

Dada la importancia de conseguir la mayor superficie de contacto entre los dedos de la pinza y la pieza, suele ser necesario el diseñar unos dedos a medida para la pieza manipular. Estos son fijados a los elementos móviles que incorporarán las pinzas.

Fuente: "Clamping Technology and gripping systems", Schunk, 2014.

Fig. 2.15. Pinzas neumáticas para robots de dos y tres dedos.

En ocasiones, la tarea encomendada al robot precisa la manipulación de piezas de diferentes características, para cada una de las cuales es necesario el uso de una pinza diferente. En estos casos hay dos opciones posibles. En la primera, el robot porta un sistema multipinza, cada una de las cuales está preparada para la manipulación de una pieza en concreto. En la segunda, el robot porta un sistema que permite el cambio automático de la pinza. Estas se encuentran preparadas para ser fijadas, de manera automática, al acoplamiento transportado por el robot, facilitando la conexión mecánica, neumática y en su caso eléctrica (señales procedentes de sensores incorporados a la pinza).

Una opción usada frecuentemente para la manipulación de piezas en tareas de coger y dejar (*pick and place*), es la sujeción mediante succión o vacío. Se emplean para ello ventosas de diferentes materiales (caucho, silicona, etc.)

sobre las que, una vez estando en contacto con la pieza, se hace el vacío. Este se consigue mediante el efecto Venturi que un caudal de aire a presión consigue sobre una tobera. El sistema de vacío por Venturi y la ventosa, constituyen una unidad compacta que es transportada por el robot. Lógicamente, este método de sujeción es sólo aplicable a materiales que permitan la estanqueidad. Ejemplos de manipulación por vacío son superficies planas de plástico, vidrio, papel o metal.

Fuente: "Sistemas de ventosas de vacío", Schmalz, 2014.

Fig. 2.16. Sistema de ventosas para la manipulación por vacío.

Herramientas terminales

En muchas aplicaciones el robot ha de realizar operaciones que no consiste en manipular objetos, sino que implican el uso de una herramienta. En general, esta herramienta debe ser construida o adaptada de manera específica para el robot, pero dado que hay aplicaciones ampliamente robotizadas, se comercializan herramientas específicas para su uso por robots. Aplicaciones como la soldadura por puntos, por arco o la pintura son algunas de ellas.

Fuente: "Spot Welding Robots", ABB, 2010.

Fig. 2.17. Robot IRB 6000ID con pinza de soldadura.

La Fig. 2.17. muestra un robot equipado con una pinza de soldadura por puntos. Esto puede incluir los actuadores para cerrar los electrodos sobre las piezas con la adecuada precisión, el transformador de soldadura y el sistema de refrigeración de los electrodos.

En la Fig. 2.18. se muestra una antorcha de soldadura al arco, mientras que la Fig. 2.19., muestra una pistola de pulverización de pintura.

Fuente: "Welding Robots", ABB, 2010.

Fig. 2.18. Robot con antorcha de soldadura al arco.

Fuente: "Painting Robots", ABB, 2010.

Fig. 2.19. Robot con pistola de pulverización de pintura.

Normalmente, la herramienta (o la pinza en su caso) está fijada rígidamente al extremo del robot, aunque en ocasiones se dota a este de un dispositivo que, mediante cierto grado de flexibilidad permite la modificación de su posición ante la presencia de esfuerzos exteriores, facilitando así tareas de contacto, como el ensamblado o el desbaste de material (pulido, desbarbado, etc.).

La Tabla 2.6. enumera algunas de las herramientas terminales más frecuentes utilizadas en la industria.

Tabla 2.6. Herramientas terminales para robots.

Tipo de herramienta	Comentarios
Pinza soldadura por puntos	Dos electrodos que se cierran sobre la pieza a soldar.
Soplete soldadura al arco	Aportan el flujo de electrodo que se funde.
Cucharón para colada	Para trabajos de fundición.
Atornillador	Suelen incluir la alimentación de tornillos.
Fresa-lija	Para perfilar, eliminar la rebabas, pulir, etc.
Pistola de pintura	Por pulverización de la pintura.
Cañón láser	Para corte de materiales, soldadura o inspección.
Cañón de agua a presión	Para corte de materiales.

Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

2.5. Métodos de Localización Espacial

Para que el robot pueda realizar las tareas de manipulación que le son encomendadas, es necesario que conozca la posición y orientación de los elementos a manipular con respecto a la base del robot. Se entiende entonces, la necesidad de contar con una serie de herramientas matemáticas que permitan especificar la posición y orientación en el espacio de piezas, herramientas y, en general, de cualquier objeto.

Estas herramientas han de ser lo suficientemente potentes como para permitir obtener de forma sencilla, relaciones espaciales entre distintos objetos y en especial, entre éstos y el manipulador. Los siguientes epígrafes introducen de forma progresiva estas herramientas, de especial importancia para la adecuada comprensión de desarrollos que aparecerán en capítulos posteriores. Sin embargo, es necesario resaltar que éstas son de aplicación general para el tratamiento de problemas de localización espacial y que, por tanto, no son de aplicación exclusiva en el campo de la robótica.

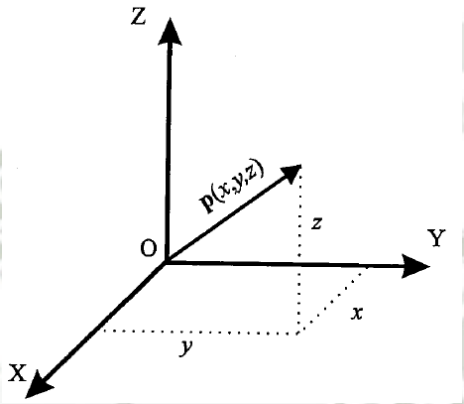
El primer apartado presenta los distintos métodos existentes para la representación de la posición y orientación espacial de un cuerpo rígido. En el siguiente epígrafe se introduce el concepto de matriz de transformación homogénea (MTH), necesario para la representación conjunta de posición y orientación. Se trata de una herramienta muy útil para representar transformaciones espaciales, estando su uso ampliamente extendido en diversos campos además del de la robótica, como por ejemplo, en el de gráficos por computador.

2.5.1. Representación de la posición y orientación

La localización de un cuerpo rígido en el espacio precisa especificar tanto su posición como su orientación. Ambas deben ser establecidas en relación al sistema de referencia definido, pudiéndose hacer uso de diferentes modos o

herramientas para especificar la relación entre la posición y orientación del cuerpo rígido y los sistemas de referencia.

Normalmente, los sistemas de referencia se definen mediante ejes perpendiculares entre sí con un origen definido. Éstos se denominan sistemas cartesianos, y en el caso de trabajar en el espacio (tres dimensiones), el sistema cartesiano OXYZ estará compuesto por una terna ortonormal de vectores unitarios OX, OY y OZ, como se ve en la Fig. 2.20.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.20. Representación de un vector en coordenadas cartesianas en 3 dimensiones

Un punto queda totalmente definido en el espacio través de los datos de su posición. Sin embargo, para el caso de un sólido rígido, es necesario además definir cuál es su orientación con respecto a un sistema de referencia.

En el caso de un robot, no es suficiente con especificar cuál debe ser la posición de su extremo, sino que, en general, es también necesario indicar su orientación. Por ejemplo, en el caso de un robot que tenga que realizar sobre una pieza curva uno operación del pulido, no bastaría con especificar los puntos de la superficie para situar adecuadamente la herramienta, sino que será necesario también conocer la orientación con que la herramienta ha de realizar la operación.

Una orientación en el espacio tridimensional viene definida por tres grados de libertad o tres componentes linealmente independientes. Para poder escribir de forma sencilla la orientación de un objeto respecto a un sistema de referencia, es habitual asignar solidariamente al objeto un nuevo sistema, y después estudiar la relación espacial existente entre los dos sistemas[6]. De forma general, esta relación vendrá dada por la posición y orientación del sistema asociado al objeto respecto al de referencia. Para el análisis de los distintos métodos de representar orientaciones, se supondrá

[6] Barrientos, A., Peñín, L. F., Balaguer, C., y Aracil, R., "Fundamentos de robótica", pp. 65-74, 2007.

que ambos sistemas coinciden en el origen, y que por tanto, no existe cambio alguno de posición entre ellos.

Matrices de Rotación

Las matrices de rotación son el método más extendido para la descripción de orientaciones, debido principalmente a la comodidad que proporciona el uso del álgebra matricial.

Supóngase los sistemas OXYZ y OUVW, coincidentes en el origen, siendo el OXYZ el sistema de referencia fijo, y el OUVW el solidario al objeto cuya orientación se desea definir. Los vectores unitarios del sistema OXYZ serán i_x, j_y, k_z, mientras que los del OUVW serán i_u, j_v, k_w. Un vector p del espacio, podrá ser referido a cualquiera de los sistemas de la siguiente manera:

$$p_{uvw} = [p_u, p_v, p_w]^T = p_u \cdot i_u + p_v \cdot j_v + p_w \cdot k_w \qquad [2.3]$$

$$p_{xyz} = [p_x, p_y, p_z]^T = p_x \cdot i_x + p_y \cdot j_y + p_z \cdot k_z \qquad [2.4]$$

Así, utilizando la definición del producto escalar, se puede obtener la siguiente equivalencia:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} \qquad [2.5]$$

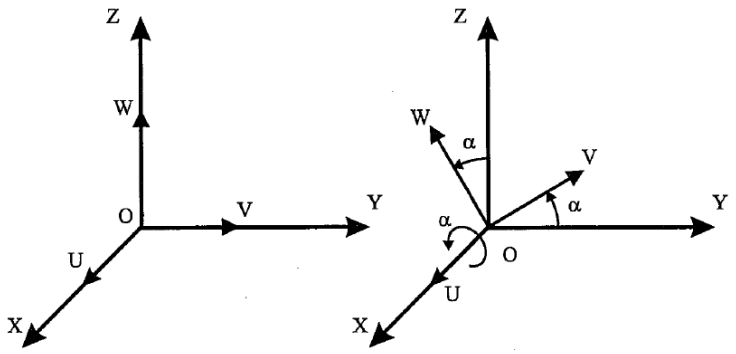
Donde:

$$\mathbf{R} = \begin{bmatrix} i_x i_u & i_x j_v & i_x k_w \\ j_y i_u & j_y j_v & j_y k_w \\ k_z i_u & k_z j_v & k_z k_w \end{bmatrix} \qquad [2.6]$$

es la matriz de rotación que define la orientación del sistema OUVW con respecto al sistema OXYZ. Recibe el nombre de matriz de cosenos directores y se trata de una matriz ortonormal, tal que la inversa de la matriz R es igual a su traspuesta: $\mathrm{R}^{-1} = \mathrm{R}^T$.

Es especialmente útil el establecer la expresión de la matriz de rotación correspondiente a sistemas girados únicamente sobre uno de los ejes del sistema de referencia. En la Fig. 2.21b, la orientación del sistema OUVW, con el eje OU coincidente con el eje OX, vendrá representada mediante la matriz:

$$\mathbf{Rot\ x}(\boldsymbol{\alpha}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \qquad [2.7]$$



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

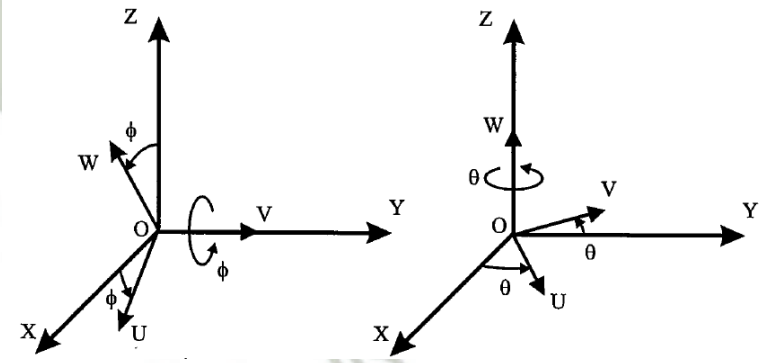
Fig. 2.21. Sistema de referencia OXYZ y rotación del sistema OUVW respecto al eje OX.

En la Fig. 2.22a, la orientación del sistema OUVW, con el eje OV coincidente con el eje OY, vendrá representada mediante la matriz:

$$\mathbf{Rot\ y}(\boldsymbol{\phi}) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \qquad [2.8]$$

En la Fig. 2.22b, la orientación del sistema OUVW, con el eje OW coincidente con el eje OZ, vendrá representada mediante la matriz:

$$\mathbf{Rot\ z}(\boldsymbol{\theta}) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [2.9]$$



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.22. Rotación del sistema OUVW respecto a los ejes OY y OZ.

Estas tres matrices, se denominan matrices básicas de rotación de un sistema espacial de tres dimensiones.

Ángulos de Euler

Para la representación de orientación en un espacio tridimensional mediante una matriz de rotación, es necesario definir nueve elementos. Aunque la utilización de las matrices de rotación presente múltiples ventajas, como se verá en el siguiente epígrafe, existen otros métodos de

definición de orientación, que hacen únicamente uso de tres componentes para su descripción. Éste es el caso de los llamados ángulos de Euler.

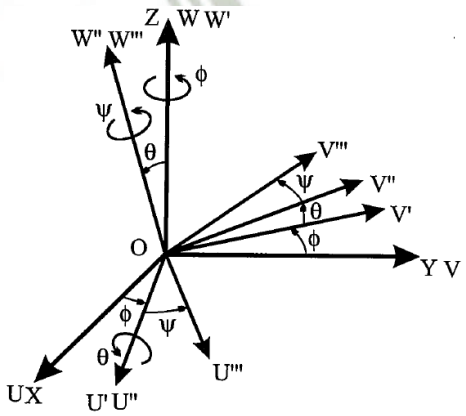
Todo sistema OUVW solidario al cuerpo cuya orientación se requiere describir, puede definirse con respecto al sistema OXYZ mediante tres ángulos: ϕ, θ, ψ, denominados ángulos de Euler, que representan los valores de los giros a realizar sobre tres ejes ortogonales entre sí, de modo que, girando sucesivamente el sistema OXYZ sobre estos ejes los valores de ϕ, θ, ψ, se obtendrá el sistema OUVW. Es necesario, por tanto, conocer además de los valores de los ángulos, cuáles son los ejes sobre los que se realizan los giros. Existen diversas posibilidades, siendo las tres más usuales las que se muestran a continuación:

Ángulos de Euler WUW

Es una de las representaciones más habituales entre las que realizan los giros sobre ejes previamente girados. Se le suele asociar con los movimientos básicos de un giroscopio. Si se parte de los sistemas OXYZ y OUVW. Inicialmente coincidentes, se puede colocar al sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Fig. 2.23.):

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ, convirtiéndose así en el OU'V'W'.
2. Girar el sistema OU'V'W' un ángulo θ con respecto al eje OU', convirtiéndose así en el OU''V''W''.
3. Girar el sistema OU''V''W'' un ángulo ψ con respecto al eje OW'' convirtiéndose finalmente en el OU'''V'''W'''.

Es importante que estas operaciones se realicen en las secuencias especificadas, pues las operaciones de giros consecutivos sobre ejes, no son conmutativas.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

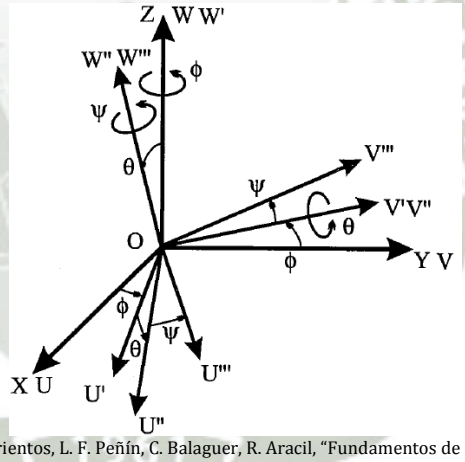
Fig. 2.23. Ángulos de Euler WUW.

Ángulos de Euler WVW

Es otra de las representaciones más habituales entre las que realizan los giros sobre ejes previamente girados. Sólo se diferencia del anterior, en la elección del eje sobre el que se realiza el segundo giro. Si se parte de los sistemas OXYZ y OUVW, inicialmente coincidentes, se puede colocar al sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Fig. 2.24):

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ, convirtiéndose así en el OU'V'W'.
2. Girar el sistema OU'V'W' un ángulo θ con respecto al eje OV', convirtiéndose así en el OU''V''W''.
3. Girar el sistema OU''V''W'' un ángulo ψ con respecto al eje OW'' convirtiéndose finalmente en el OU'''V'''W'''.

Como antes, es preciso considerar que el orden de los giros no es conmutativo.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.24. Ángulos de Euler WVW.

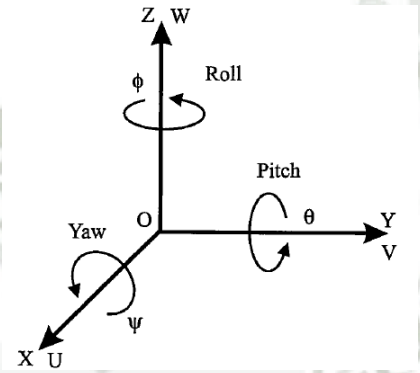
Ángulos de Euler XYZ

Estos giros sobre los ejes fijos se denominan guiñada, cabeceo y alabeo (Yaw, Pitch y Roll). Se trata de la representación utilizada generalmente en aeronáutica. Es también la más habitual de entre las que se aplican a los giros sobre los ejes del sistema fijo. Si se parte de los sistemas OXYZ y OUVW, al igual que en el caso anterior, se puede colocar al sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Fig. 2.25.):

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OX, Es el denominado Yaw o guiñada.

2. Girar el sistema OUVW un ángulo θ con respecto al eje OY. Es el denominado Pitch o cabeceo.
3. Girar el sistema OUVW un ángulo ψ con respecto al eje OZ. Es el denominado Roll o alabeo.

Al igual que en los casos anteriores, y en general siempre que se concatenan varios giros seguidos, es necesario considerar que no se trata de una transformación conmutativa, debiéndose seguir una secuencia determinada de aplicación de los mismos.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.25. Ángulos de Euler XYZ (Yaw, Pitch y Roll).

2.5.2. Matrices de Transformación Homogénea

En el epígrafe anterior, se han estudiado distintos métodos de representar la posición o la orientación de un sólido en el espacio. Pero ninguno de estos métodos por sí solo permite una representación conjunta de la posición y de la orientación (localización). Las matrices de transformación homogénea, permiten esta representación conjunta, facilitando su uso mediante el álgebra matricial.

Al objeto de poder representar y tratar conjuntamente la posición y orientación de un sólido, se introducen las coordenadas homogéneas. Un elemento de un espacio n-dimensional, se encuentra representado en coordenadas homogéneas por $n+1$ dimensiones, de tal forma que un vector $p(x, y, z)$ vendrá representado por $p(wx, wy, wz)$, donde w tiene un valor arbitrario y representa un factor de escala. De forma general, un vector $p = a\mathrm{i} + b\mathrm{j} + c\mathrm{k}$, donde i, j y k son los vectores unitarios de los ejes OX, OY y OZ del sistema de referencia OXYZ, se representa en coordenadas homogéneas mediante el vector columna:

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} aw \\ bw \\ cw \\ w \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix} \qquad [2.10]$$

Por ejemplo, el vector $2\mathrm{i} + 3\mathrm{j} + 4\mathrm{k}$ se puede representar en coordenadas homogéneas como $[2,3,4,1]^T$, o como $[4,6,8,2]^T$ o también como $[-6,-9,-12,-3]^T$, etc. Los vectores nulo se representan como $[0,0,0,n]^T$, donde n es no nulo. Los vectores de la forma $[a,b,c,0]^T$ sirven para representar direcciones, pues representan vectores de longitud infinita.

A partir de la definición de las coordenadas homogéneas, surge inmediatamente el concepto de matriz de transformación homogénea. Se define como matriz de transformación homogénea T, a una matriz de dimensión 4x4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{p}_{3x1} \\ \mathbf{f}_{1x3} & \mathbf{w}_{1x1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} \qquad [2.11]$$

Se puede considerar que una matriz homogénea se haya compuesta por cuatro submatrices de distinto tamaño: una submatriz $\mathbf{R}_{3x3}$ que corresponde a una matriz de rotación; una submatriz $\mathbf{p}_{3x1}$ que corresponde al vector de traslación; una submatriz $\mathbf{f}_{1x3}$ que representa una transformación de perspectiva, y una submatriz $\mathbf{w}_{1x1}$ que representa un escalado global. En robótica, generalmente sólo interesa conocer el valor de $\mathbf{R}_{3x3}$ y de $\mathbf{p}_{3x1}$, considerándose las componentes de $\mathbf{f}_{1x3}$ nulas (se supone que no existe ninguna transformación de perspectiva), y la de $\mathbf{w}_{1x1}$ la unidad. Al tratarse de una matriz 4x4, los vectores sobre los que se aplique deberán contar con 4 dimensiones, que serán las coordenadas homogéneas del vector tridimensional de que se trate.

Una matriz homogénea sirve para transformar un vector expresado en coordenadas homogéneas con respecto a un sistema O'UVW, a su expresión en las coordenadas del sistema de referencia OXYZ. También se puede utilizar para rotar y girar un vector referido a un sistema de referencia fijo, y en definitiva sirve para expresar la orientación y posición de un sistema de referencia O'UVW con respecto a otro fijo OXYZ. La matriz T de transformación se suele escribir de la siguiente forma:

$$\mathrm{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} & \boldsymbol{o} & \boldsymbol{a} & \boldsymbol{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.12]$$

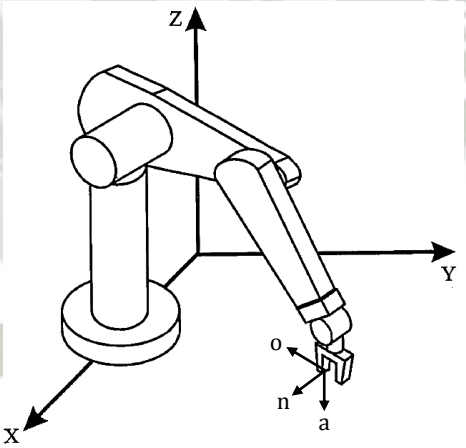
Donde n, o, a es una terna ortonormal que representa la orientación y p es un vector que representa la posición. Es decir, el vector columna n representa las coordenadas del eje O'U del sistema O'UVW con respecto del sistema OXYZ. De igual forma, el vector columna o representa las

coordenadas del eje O'V del sistema O'UVW con respecto del sistema OXYZ, y el vector columna a representa las coordenadas del eje O'W del sistema O'UVW con respecto del sistema OXYZ.

Si se aplica lo anteriormente indicado a un robot, la matriz de transformación homogénea permite describir la localización (posición y orientación) de su extremo con respecto a su base. Así, asociando a la base del robot un sistema de referencia fijo (OXYZ) y al extremo un sistema de referencia que se mueva con él, cuyo origen se encuentren en el punto p y los vectores directores son n, o y a, escogidos de modo que (Fig. 2.26.):

- ***a*:** sea un vector en la dirección de aproximación del extremo del robot a su destino.
- ***o*:** sea un vector perpendicular a a en el plano definido por la pinza del robot.
- ***n*:** sea un vector que forme una terna ortogonal con los dos anteriores.

se tendrá que el extremo del robot quede perfectamente localizado por la matriz de transformación homogénea dada por la matriz T.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.26. Definición de la matriz [noap] de la localización del extremo del robot.

2.5.3. Relación entre Ángulos de Euler y Matrices Homogéneas

Debido a que los métodos vistos para la representación espacial son equivalentes, es decir, expresan lo mismo de forma distinta, deberá existir un modo de pasar de un tipo de representación a otro. A continuación, se muestran las relaciones de paso que se utilizan más frecuentemente. A través de ellas es posible pasar de una representación a cualquier otra, aunque en algunos casos sea más cómodo utilizar una representación auxiliar intermedia.

Los ángulos de Euler sólo son capaces de realizar una representación de la orientación. Por ello, a la hora de obtener la matriz homogénea equivalente a un conjunto de ángulos de Euler dados, únicamente quedará definida la submatriz de rotación $\mathbf{R}_{3x3}$. La obtención de la matriz homogénea correspondiente a cada conjunto de ángulos de Euler es inmediata; bastará con componer las matrices que representan las rotaciones que definen los propios ángulos:

Sistema WUW

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

$$\mathrm{T}_{ZXZ} = \mathrm{Rot\,z(f)Rot\,x(q)Rot\,z}(y) \qquad [2.13]$$

Que desarrollado en forma matricial:

$$\mathrm{T}_{ZXZ} = \begin{bmatrix} C\phi C\psi - S\phi C\theta S\psi & -C\phi S\psi - S\phi C\theta C\psi & S\phi S\theta & 0 \\ S\phi C\psi + C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & -C\phi S\theta & 0 \\ S\theta S\psi & S\theta C\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.14]$$

Sistema WVW

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

$$\mathrm{T}_{ZYZ} = \mathrm{Rot\,z(f)Rot\,y(q)Rot\,z}(y) \qquad [2.15]$$

Que desarrollado en forma matricial:

$$\mathrm{T}_{ZYZ} = \begin{bmatrix} C\phi C\theta C\psi - S\phi S\psi & -C\phi C\theta S\psi - S\phi C\psi & C\phi S\theta & 0 \\ S\phi C\theta C\psi + C\phi S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\phi S\theta & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.16]$$

Sistema XYZ (Yaw – Pitch - Roll)

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

$$\mathrm{T}_{ZYZ} = \mathrm{Rot\,z(f)Rot\,y(q)Rot\,x}(y) \qquad [2.17]$$

y de forma matricial:

$$\mathrm{T}_{ZYZ} = \begin{bmatrix} C\phi C\theta & C\phi S\theta S\psi - S\phi C\psi & C\phi S\theta C\psi + S\phi S\psi & 0 \\ S\phi C\theta & S\phi S\theta S\psi + C\phi C\psi & S\phi S\theta C\psi - C\phi S\psi & 0 \\ -S\theta & C\theta S\psi & C\theta C\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.18]$$

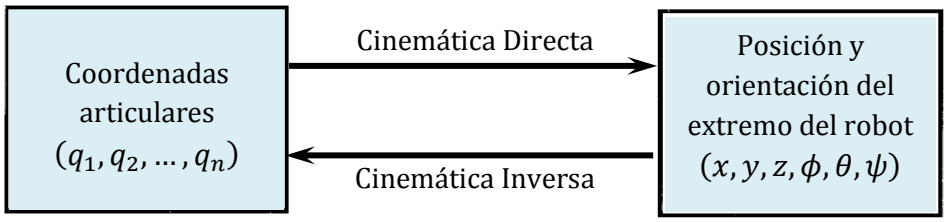
2.6. Cinemática del robot

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia, sin considerar las fuerzas que intervienen. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

Existen tres aspectos fundamentales a resolver en la cinemática del robot; el primero de ellos se conoce como la cinemática directa, y consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; el segundo, denominado cinemática inversa, resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas.

Denavit y Hartenberg propusieron un método sistemático para describir y representar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea 4x4, que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base.

Por otra parte, la cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo. Esta relación viene dada por la cinemática diferencial mediante la matriz Jacobiana.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.
Fig. 2.27. Diagrama de relación entre cinemática directa e inversa.

2.6.1. Cinemática Directa

La resolución de la cinemática directa permite conocer cuál es la posición y orientación que adopta el extremo del robot, cuando cada una de las

variables que fijan la posición u orientación de sus articulaciones toman valores determinados.

Dado que son las variables articulares las que pueden ser leídas directamente de los correspondientes sensores por la unidad de control del robot, el modelo cinemático directo será utilizado por éste, entre otros fines, para presentar al usuario información relativa a la localización del extremo del robot.

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

$$x = f_x(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.19]$$

$$y = f_y(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.20]$$

$$z = f_z(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.21]$$

$$\phi = f_\phi(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.22]$$

$$\theta = f_\theta(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.23]$$

$$\psi = f_\psi(q_1, q_2, q_3, q_4, q_5, q_6) \quad [2.24]$$

La obtención del modelo cinemático directo puede ser abordado mediante dos enfoques diferentes denominados métodos geométricos y métodos basados en cambios de sistemas de referencia.

Los primeros son adecuados para casos simples, pero al no ser sistemáticos, su aplicación queda limitada a robots con pocos grados de libertad; por tal motivo, no se desarrollará su estudio, sin embargo, el detalle del mismo puede encontrarse en [1].

Los métodos basados en cambio de sistemas de referencia, permiten de una manera sistemática, abordar la obtención del modelo cinemático directo del robot para robots de n grados de libertad, siendo éstos, por tanto, los más frecuentemente utilizados, en particular los que usan las matrices de transformación homogénea, como el algoritmo de Denavit-Hartenberg.

Algoritmo de Denavit-Hartenberg

Aunque para describir la relación que existe entre dos elementos contiguos se puede hacer uso de cualquier sistema de referencia ligado a cada

elemento, la forma habitual que se suele utilizar en robótica es la representación de Denavit-Hartenberg (D-H). Denavit y Hartenberg propusieron en 1955, un método matricial que establece la localización que debe tomar cada sistema de coordenadas $\{S_i\}$ ligado a cada eslabón i de una cadena articulada, para poder sistematizar la obtención de las ecuaciones cinemáticas de la cadena completa.

Escogiendo los sistemas de coordenadas asociados a cada eslabón según la representación propuesta por D-H, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Hay que hacer notar que si bien en general, una matriz de transformación homogénea queda definida por 6 grados de libertad, el método de Denavit-Hartenberg, permite, en eslabones rígidos, reducir éste a 4, con la correcta elección de los sistemas de coordenadas.

Estas 4 transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento $i-1$, con el sistema del elemento i. Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema $\{S_{i-1}\}$ al $\{S_i\}$ mediante estas 4 transformaciones, está garantizado sólo si los sistemas $\{S_{i-1}\}$ y $\{S_i\}$ han sido definidos de acuerdo a unas normas determinadas que se expondrán posteriormente):

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

Donde todas 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 anteriormente. De este modo se tiene que:

$$^{i-1}\mathrm{A_i} = \mathbf{Rotz}(\theta_i)\mathbf{T}(0,0,d_i)\mathbf{T}(a_i,0,0)\mathbf{Rotx}(\alpha_i) \qquad [2.25]$$

Y realizando el producto entre matrices se obtiene que:

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

$$ {}^{i-1}A_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.26] $$

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

Como se ha indicado, para que la matriz ${}^{i-1}A_i$, definida en [2.26], relacione los sistemas $\{S_{i-1}\}$ y $\{S_i\}$, es necesario que los sistemas se hayan escogido de acuerdo a unas determinadas normas. Éstas, junto con la definición de los 4 parámetros de Denavit-Hartenberg, conforman el siguiente algoritmo para la resolución del problema cinemático directo:

DH1 Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

DH2 Numerar cada articulación comenzando por 1 (la correspondiente al primer GDL) y acabando en n.

DH3 Localizar el eje de la articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

DH4 Para i de 0 a $n-1$, situar el eje $\boldsymbol{z_i}$ sobre el eje de la articulación $i+1$.

DH5 Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje $\boldsymbol{z_0}$. Los ejes $\boldsymbol{x_0}$ e $\boldsymbol{y_0}$ se situarán de modo que formen un sistema dextrógiro con $\boldsymbol{z_0}$.

DH6 Para i de 1 a $n-1$, situar el origen del sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje $\boldsymbol{z_i}$ con la línea normal común a $\boldsymbol{z_{i-1}}$ y $\boldsymbol{z_i}$. Si ambos ejes se cortasen, se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos, $\{S_i\}$ se situaría en la articulación $i+1$.

DH7 Situar $\boldsymbol{x_i}$ en la línea normal común a $\boldsymbol{z_{i-1}}$ y $\boldsymbol{z_i}$.

DH8 Situar $\boldsymbol{y_i}$ para que forme un sistema dextrógiro con $\boldsymbol{x_i}$ y $\boldsymbol{z_i}$.

DH9 Situar el sistema $\{S_n\}$ en el extremo, del robot de modo que $\boldsymbol{z_n}$ coincida con la dirección de $\boldsymbol{z_{n-1}}$ y $\boldsymbol{x_n}$ sea normal a $\boldsymbol{z_{n-1}}$ y $\boldsymbol{z_n}$.

DH10 Obtener θ_i como el ángulo que hay que girar en torno a $\boldsymbol{z_{i-1}}$ para que $\boldsymbol{x_{i-1}}$ y $\boldsymbol{x_i}$ queden paralelos.

DH11 Obtener d_i como la distancia medida a lo largo de $\boldsymbol{z_{i-1}}$, que habría que desplazar $\{S_{i-1}\}$ para que $\boldsymbol{x_i}$ y $\boldsymbol{x_{i-1}}$ quedasen alineados.

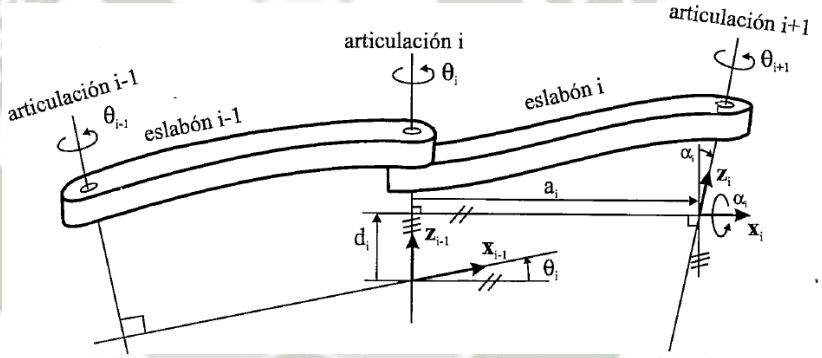
DH12 Obtener a_i como la distancia medida a lo largo de $\boldsymbol{x_i}$ (que ahora coincidiría con $\boldsymbol{x_{i-1}}$) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

DH13 Obtener α_i como el ángulo que habría que girar en torno a $\boldsymbol{x_i}$, para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

DH14 Obtener las matrices de transformación ${}^{i-1}\mathrm{A}_i$ definidas en [2.26].

DH15 Obtener la matriz de transformación homogénea que relaciona el sistema de la base con el del extremo del robot $\mathbf{T} = {}^{0}\mathbf{A}_1\,{}^{1}\mathbf{A}_2\,{}^{n-1}\mathbf{A}_n$.

DH16 La matriz **T** define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base, en función de las n coordenadas articulares.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.28. Parámetros de D-H para un eslabón giratorio.

Los cuatro parámetros de D-H $(\theta_i, d_i, a_i, \alpha_i)$ dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente. En concreto, ellos representan (Fig. 2.28.):

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

d_i Es la distancia a lo largo del eje $\boldsymbol{z_{i-1}}$, que desde el origen del sistema de coordenadas $(i-1)$-ésimo hasta la intersección del eje $\boldsymbol{z_{i-1}}$ con el eje $\boldsymbol{x_i}$. Se trata de un parámetro variable de articulaciones prismáticas.

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

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

Una vez obtenidos los parámetros D-H, el cálculo de las relaciones entre los eslabones consecutivos del robot es inmediato, ya que vienen dadas por las matrices ${}^{i-1}A_i$, que se calculan según la expresión general [2.26]. Las relaciones entre varios eslabones consecutivos dos a dos, vienen dadas por las matrices T que, como ya se comentó anteriormente, se obtienen como producto de un conjunto de matrices A.

Obtenida la matriz T, ésta expresará la orientación (submatriz (3x3) de rotación) y posición (submatriz (3x1) de traslación) del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

2.6.2. Cinemática Inversa

El objetivo de la cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = [q_1, q_2, \dots, q_n]^T$ para que su extremo se posicione y oriente según una determinada localización espacial (p, [n, o, a]).

Así como es posible abordar el problema cinemático directo de una manera sistemática a partir de la utilización de matrices de transformación homogénea, e independientemente de la configuración del robot, no ocurre lo mismo con el problema cinemático inverso, siendo el procedimiento de obtención de las ecuaciones fuertemente dependiente de la configuración del robot.

Se han desarrollado algunos procedimientos genéricos susceptibles de ser programados, de modo que un computador pueda, a partir del reconocimiento de la cinemática del robot (con sus parámetros de Denavit-Hartenberg, por ejemplo) obtener la n-cupla de valores articulares que posicionan y orientan su extremo. El inconveniente de estos procedimientos es que se trata de métodos numéricos iterativos, cuya velocidad de convergencia, e incluso su convergencia en sí, no está siempre garantizada.

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

$$q_k = f_k(x, y, z, \phi, \theta, \psi) \qquad k = 1 \dots n \ (GDL) \qquad [2.27]$$

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

- En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real (por ejemplo, en el seguimiento de una determinada trayectoria). Una solución de tipo iterativo no garantiza tener la solución en el momento adecuado.

- Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única, existiendo diferentes n-cuplas $[q_1, \dots, q_n]^T$ que posicionan y orientan el extremo del robot del mismo modo. En estos casos, una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada de entre las posibles (por ejemplo, límites en los recorridos articulares).

No obstante, a pesar de las dificultades comentadas, la mayor parte de los robots poseen cinemáticas relativamente simples, que facilitan en cierta medida la resolución de su problema cinemático inverso. Por ejemplo, si se consideran sólo los tres primeros grados de libertad de muchos robots, éstos tiene una estructura planar, esto es, los tres primeros elementos quedan contenidos en un plano. Esta circunstancia facilita la resolución del problema. Asimismo, en muchos robots se da la circunstancia de que los tres grados de libertad últimos, dedicados fundamentalmente a orientar el extremo del robot, corresponden a giros sobre ejes que se cortan en un punto. De nuevo esta situación facilita el cálculo de la n-cupla $[q_1, \dots, q_n]^T$ correspondiente a la posición y orientación deseadas. Por tanto, para los casos citados y otros, es posible establecer ciertas pautas generales que permitan plantear y resolver el problema cinemático inverso de una manera sistemática.

Los métodos geométricos permiten, normalmente, obtener los valores de las primeras variables articulares, que son las que consiguen posicionar el robot (prescindiendo de la orientación de su extremo). Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot.

Como alternativa para resolver el mismo problema, se puede recurrir a manipular directamente las ecuaciones correspondientes al problema cinemático directo, es decir, puesto que éste establece la relación:

$$\begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = [t_{ij}] \qquad [2.28]$$

Donde los elementos t_{ij} son función de las coordenadas articulares $[q_1, \dots, q_n]^T$; es posible pensar que mediante ciertas combinaciones de las 12 ecuaciones planteadas en [2.26], se pueda despejar las n variables articulares q_i en función de las componentes de los vectores n, o, a y p. Debe considerarse en este caso, que en general las doce ecuaciones responden a ecuaciones trigonométricas acopladas cuya resolución no es trivial. Para facilitar esta solución, es posible proceder de manera ordenada, despejando sucesivamente los grados de libertad.

Por último, si se consideran robots con capacidad de posicionar y orientar su extremo en el espacio, esto es, robots con 6 GDL, el método de desacoplamiento cinemático permite, para determinados tipos de robots, resolver los primeros grados de libertad dedicados al posicionamiento, de manera independiente a la resolución de los últimos grados de libertad, dedicados a la orientación. Cada uno de estos dos problemas más simples podrá ser tratado y resuelto por cualquiera de los procedimientos anteriores. Por ello, en el presente apartado se analiza el método de desacoplo cinemático.

Desacoplo Cinemático

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

Ahora bien, como es sabido, en general no basta con posicionar el extremo del robot en un punto del espacio, sino que casi siempre es preciso también conseguir que la herramienta que aquél porta, se oriente de una manera determinada. Para ello, los robots cuentan con otros tres grados de libertad, situados al final de la cadena cinemática y cuyos ejes, con frecuencia, se cortan en un punto, que informalmente se denominará muñeca del robot. Si bien la variación de estos tres últimos grados de libertad origina un cambio

en la posición final del extremo real del robot, su verdadero objetivo es poder orientar la herramienta del robot libremente en el espacio.

El método de desacoplo cinemático es aplicable a aquellos robots cuyos tres últimos grados de libertad se cortan en un punto, sacando partido de este hecho, separando los problemas de obtención del modelo cinemático inverso de posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los 3 últimos ejes (muñeca del robot) calculándose los valores de las tres primeras variables articulares (q_1, q_2, q_3) que consiguen posicionar este punto. A continuación, a partir de los datos de orientación deseada para el extremo del robot y de los ya calculados, se obtienen los valores del resto de las variables articulares. El detalle aplicado del desacoplo cinemático se desarrolla en el Capítulo III.

2.6.3. Cinemática Diferencial

El modelado cinemático de un robot busca las relaciones entre las variables articulares y la posición (expresadas normalmente en forma de coordenadas cartesianas) y orientación del extremo del robot (expresada con matrices de rotación, ángulos de Euler o algún otro método establecido). En esta relación no se tienen en cuenta las fuerzas o pares que actúan sobre el robot (actuadores, cargas, fricciones, etc.) y que pueden originar el movimiento del mismo. Sin embargo, sí incumbe a la cinemática del robot conocer la relación entre las velocidades de las coordenadas articulares y las de la posición y orientación del extremo, con lo que es equivalente, el efecto que un movimiento diferencial de las variables articulares tiene sobre las variables en el espacio de la tarea. Esta relación queda definida por la cinemática diferencial. Mediante ella, el sistema de control del robot puede establecer que velocidades debe imprimir a cada articulación (a través de sus respectivos actuadores) para conseguir que el extremo desarrolle una trayectoria temporal concreta, por ejemplo, una línea recta a velocidad constante.

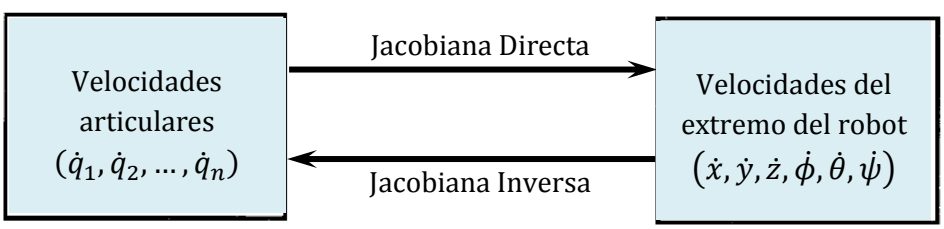
La cinemática diferencial queda concretada en la denominada matriz Jacobiana. En general, la matriz Jacobiana de un robot relaciona el vector de velocidades articulares $(\dot{q_1}, \dot{q_2}, \dot{q_n})$ con otro vector de velocidades expresado en un espacio distinto. Existen diferentes posibilidades a la hora de seleccionar este espacio. Una primera elección es la de considerar la relación con las velocidades de localización del extremo del robot, siendo ésta la posición y orientación expresada en base a sus coordenadas cartesianas y sus ángulos de Euler $(\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$ (otras representaciones

de la orientación pueden ser consideradas). Esta relación viene dada por la denominada Jacobiana analítica del manipulador.

Una segunda elección es relacionar las velocidades articulares con los vectores de velocidad lineal y angular $(v_x, v_y, v_z, w_x, w_y, w_z)$ con que se mueve el extremo del robot, expresados en un sistema de referencia determinado, por ejemplo, el del origen. La relación entre ambas velocidades (articulares y lineal-angular del extremo) se obtiene a través de la denominada matriz Jacobiana geométrica o simplemente Jacobiana del manipulador. En ambos casos, la matriz Jacobiana directa permite conocer una expresión de las velocidades del extremo del robot a partir de los valores de las velocidades de cada articulación. Por su parte, la matriz Jacobiana inversa permitirá conocer las velocidades articulares necesarias para obtener un vector concreto de velocidades del extremo.

Jacobiana Analítica

Supóngase conocida la posición (x, y, z) del extremo del robot así como su orientación, definida por los ángulos de Euler WVW (ϕ, θ, ψ), por ejemplo. La Jacobiana analítica relaciona las velocidades articulares $(\dot{q}_1, \dot{q}_2, \dots, \dot{q}_n)$ con las velocidades de localización (posición y orientación) del extremo del robot $(\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 2.29. Jacobiana analítica directa e inversa.

El método más directo para obtener la relación entre velocidades articulares y del extremo del robot, consiste en diferenciar las ecuaciones correspondientes al modelo cinemático directo. Así, supónganse conocidas las ecuaciones que resuelven el problema cinemático directo de un robot de n GDL:

$$x = f_x(q_1, \dots, q_n) \quad y = f_y(q_1, \dots, q_n) \quad z = f_z(q_1, \dots, q_n)$$

$$\phi = f_\phi(q_1, \dots, q_n) \quad \theta = f_\theta(q_1, \dots, q_n) \quad \psi = f_\psi(q_1, \dots, q_n)$$

Si se derivan con respecto al tiempo ambos miembros del conjunto de ecuaciones anteriores, se tendrá:

$$\dot{x} = \sum_{1}^{n} \frac{\partial f_x}{\partial q_i} \dot{q}_\iota \qquad \dot{y} = \sum_{1}^{n} \frac{\partial f_y}{\partial q_i} \dot{q}_\iota \qquad \dot{z} = \sum_{1}^{n} \frac{\partial f_z}{\partial q_i} \dot{q}_\iota$$

$$\dot{\phi} = \sum_{1}^{n} \frac{\partial f_\phi}{\partial q_i} \dot{q}_\iota \qquad \dot{\theta} = \sum_{1}^{n} \frac{\partial f_\theta}{\partial q_i} \dot{q}_\iota \qquad \dot{\psi} = \sum_{1}^{n} \frac{\partial f_\psi}{\partial q_i} \dot{q}_\iota$$

O expresado en forma matricial:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J_a \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \qquad con \qquad J_a = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\psi}{\partial q_1} & \cdots & \frac{\partial f_\psi}{\partial q_n} \end{bmatrix} \qquad [2.29]$$

La matriz J_a se denomina matriz Jacobiana analítica. Puesto que el valor numérico de cada uno de los elementos $[j_{pq}]$ de la Jacobiana dependerá de los valores de las coordenadas articulares q_i, el valor de la jacobiana será diferente en cada uno de los puntos del espacio articular.

Jacobiana Geométrica

La Jacobiana analítica presentada en el epígrafe anterior, relaciona las velocidades de las articulaciones con las velocidades de variación de la posición y orientación del extremo del robot. Otra posible relación de interés, es la que se establece entre las velocidades articulares y la velocidad lineal (v) y angular (w) del extremo del robot expresadas habitualmente en el sistema de referencia de la base del robot $\{S_0\}$.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} = J \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \qquad [2.30]$$

Para diferenciarla de la relación anterior, se denominará a ésta Jacobiana geométrica o simplemente Jacobiana. La deducción de esta matriz Jacobiana es menos directa que la Jacobiana analítica, precisando su obtención algunas consideraciones.

Existen diferentes procedimientos que permiten la obtención numérica de la Jacobiana a partir de la información contenida en las matrices ${}^{i-1}A_i$ que definen el modelo cinemático.

Debe considerarse que puesto que las matrices ${}^{i-1}\mathrm{A}_{\mathrm{i}}$ tienen, para un robot determinado, una expresión genérica en función de q_i (tomando ${}^{i-1}\mathrm{A}_{\mathrm{i}}$ un valor numérico concreto para un valor numérico de q_i) estos procedimientos pueden ser aplicados tanto de manera analítica, para obtener la expresión general de la Jacobiana, como numérica, para la obtención del valor instantáneo de la Jacobiana en una posición concreta del robot.

El siguiente procedimiento de obtención de la Jacobiana, está basado en la propagación de las velocidades. Este método permite obtener las columnas de la matriz Jacobiana geométrica, que relaciona las velocidades articulares con las velocidades lineales y angulares del extremo del robot, medidas con respecto del sistema de base, a partir de las matrices ${}^{i-1}\mathrm{A}_{\mathrm{i}}$.

Se denomina ${}^{0}\mathrm{z}_i$ al vector unitario orientado según el eje de la articulación $i+1$, definido en el sistema de coordenadas de la base del robot $\{S_0\}$ (tal como se definió en las reglas DH3 y DH4 del Algoritmo de Denavit-Hartenberg).

Las matrices ${}^{i-1}\mathrm{A}_i = \begin{bmatrix} {}^{i-1}n_i & {}^{i-1}o_i & {}^{i-1}a_i & {}^{i-1}p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$ contienen la información de los vectores directores y origen del sistema $\{S_i\}$ en la base $\{S_{i-1}\}$. Por tanto, la matriz ${}^{0}\mathrm{A}_i = \begin{bmatrix} {}^{0}n_i & {}^{0}o_i & {}^{0}a_i & {}^{0}p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$ contendrá la información de los vectores directores y origen del sistema $\{S_i\}$ (solidario al esclavo i y con su eje z_i en el eje de la articulación $i+1$) en la base $\{S_0\}$. De modo que ${}^{0}\mathrm{z}_i$ estará definido por los tres primeros elementos de la tercera columna de ${}^{0}\mathrm{A}_i$. (Al ser ${}^{0}\mathrm{A}_0$ la matriz identidad ${}^{0}\mathrm{z}_0$ será el vector (0,0,1)).

$$ {}^{0}\mathrm{z}_i = {}^{0}\mathrm{A}_i(1{:}3,3) \qquad [2.31] $$

Donde la notación (i, j, k) indica los elementos i a j de la columna k.

Se denominará ${}^{i}\mathrm{p}_n$ al vector que va desde el origen del sistema $\{S_i\}$ hasta el extremo del robot (origen del sistema $\{S_n\}$) expresado en el sistema de la base del robot $\{S_0\}$.

Puesto que la cuarta columna de ${}^{0}\mathrm{A}_n$ contiene las coordenadas del extremo del robot en el sistema $\{S_0\}$ y la cuarta columna del ${}^{0}\mathrm{A}_i$ contiene las coordenadas del origen del sistema $\{S_i\}$ en el sistema $\{S_0\}$. ${}^{i}\mathrm{p}_n$ se obtendrá restando las cuartas columnas de ${}^{0}\mathrm{A}_n$ y ${}^{0}\mathrm{A}_i$.

$$ {}^{i}\mathrm{p}_n = {}^{0}\mathrm{A}_n(1{:}3,4) - {}^{0}\mathrm{A}_i(1{:}3,4) \qquad [2.32] $$

Definidos los vectores ${}^{0}z_i$ y ${}^{i}p_n$, la matriz Jacobiana que relaciona las velocidades articulares con las velocidades de traslación y rotación del extremo del robot, expresadas en el sistema de coordenadas de la base (relación [2.30]), se puede obtener como una matriz $6 \times n$ (n = número de grados de libertad) expresada por columnas como:

$$J = [J_1 \quad J_2 \ldots \quad J_n] \qquad [2.33]$$

Donde:

$$J_i = \begin{Bmatrix} \begin{bmatrix} {}^{0}z_{i-1} \times {}^{i-1}p_n \\ {}^{0}z_{i-1} \end{bmatrix} & \text{Si el eslabón } i \text{ es de rotación} \\ \begin{bmatrix} {}^{0}z_{i-1} \\ 0 \end{bmatrix} & \text{Si el eslabón } i \text{ es de traslación} \end{Bmatrix} \qquad [2.34]$$

Se va a presentar a continuación, el modo en que están relacionados los dos conceptos presentados anteriormente de la Jacobiana, esto es, la matriz J_a que relaciona las velocidades articulares con las velocidades de la localización del extremo del robot $(\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$, y la matriz J que relaciona las velocidades articulares con el vector de velocidad lineal y angular del extremo[7].

La relación entre ambas viene dada por la expresión:

$$J = \begin{bmatrix} \mathrm{I} & 0 \\ 0 & \mathrm{Q} \end{bmatrix} J_a \qquad [2.35]$$

Donde las matrices I y 0, son las matrices identidad y nula de dimensión (3x3) respectivamente, y la matriz Q, viene definida por la expresión:

$$Q = \begin{bmatrix} 0 & -S_\phi & C_\phi S_\theta \\ 0 & C_\phi & S_\phi S_\theta \\ 1 & 0 & C_\theta \end{bmatrix} \qquad [2.36]$$

Donde (ϕ, θ, ψ) son los ángulos de Euler WVW asociados a la submatriz de rotación $\mathrm{R} = [n \quad o \quad a]$ y que pueden ser obtenidos de ésta por comparación con la caja de rotación de la expresión [2.16].

En el caso de que se pretenda obtener la Jacobiana analítica J_a a partir de la Jacobiana J, se invertirá la matriz que la relaciona, resultando:

[7] La deducción de esta relación puede encontrarse en: Mark W. Spong, Seth Hutchinson, M. Vidyasagar, "Robot Modeling and Control", John Wiley & Sons,Ltd, 2006.

$$J_a = \begin{bmatrix} \mathrm{I} & 0 \\ 0 & \mathrm{Q}^{-1} \end{bmatrix} J \qquad [2.37]$$

Es preciso hacer la salvedad, que en el caso de que $\theta = 0$ o $\theta = \pi$, resultan indeterminados los valores de ϕ y ψ, y por tanto, no es posible obtener la matriz Q, no pudiéndose aplicar la relación anterior.

Jacobiana Inversa

Del mismo modo que se ha obtenido la relación directa, que permite obtener las velocidades del extremo a partir de las velocidades articulares, puede obtenerse la relación inversa que permite calcular las velocidades articulares partiendo de las del extremo. En la obtención de la relación inversa pueden emplearse diferentes procedimientos.

En primer lugar, conocida la relación directa, dada por la matriz Jacobiana [2.29] o [2.30], se puede obtener la relación inversa invirtiendo simbólicamente la matriz.

$$\begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = {J_a}^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad ; \quad \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J^{-1} \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix} \qquad [2.38]$$

Esta alternativa de planteamiento sencillo, es en la práctica de difícil realización. Suponiendo que la matriz Jacobiana sea cuadrada, la inversión simbólica de una matriz 6x6, cuyos elementos son funciones trigonométricas, es de gran complejidad, siendo este procedimiento inviable.

Como segunda alternativa, puede plantearse la evaluación numérica de la matriz Jacobiana para una configuración (q_i) concreta del robot, e invirtiendo numéricamente está matriz, encontrar la relación inversa válida para esa configuración. En este caso, hay que considerar, en primer lugar, que el valor numérico de la Jacobiana va cambiando a medida que el robot se mueve y, por tanto, la Jacobiana inversa ha de ser recalculada constantemente. Además, pueden existir n-cuplas $(q_1, \dots, q_n)$ para las cuales la matriz Jacobiana no se ha invertible por ser su determinante, denominado Jacobiano, nulo. Estas configuraciones del robot en las que el Jacobiano se anula se denominan configuraciones singulares y serán tratadas más adelante. Una tercera dificultad que pueden surgir con éste y

otros procedimientos de cómputo de la matriz Jacobiana inversa, se deriva de la circunstancia de que la matriz Jacobiana no sea cuadrada. Esto ocurre cuando el número de grados de libertad del robot no coincide con la dimensión del espacio de la tarea (normalmente seis). En el caso de que el número de grados de libertad sea inferior, la matriz Jacobiana tendrá más filas que columnas. Esto quiere decir, que el movimiento del robot está sometido a ciertas restricciones (por ejemplo, no se puede alcanzar cualquier orientación). En ocasiones, esto ocurre en casos en los que esta restricción no tiene importancia, como en robots dedicados a tareas como soldaduras por arco, en las que la orientación de la herramienta en cuanto su giro en torno al vector a es indiferente, o en algunas tareas de coger y dejar en las que el vector a siempre toma la dirección vertical. En estos casos, se puede eliminar algún grado de libertad del espacio de la tarea, quedando una nueva matriz Jacobiana cuadrada.

En los casos en que el robot sea redundante (más de 6 GDL o más columnas que filas en la matriz Jacobiana) existirán grados de libertad articulares innecesarios, es decir, que no será preciso mover para alcanzar las nuevas posiciones y orientaciones del extremo requeridas. Por ello, la correspondiente velocidad articular podrá ser tomada como cero, o si fuera útil, como un valor constante.

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

Jacobiana Pseudoinversa

En general, en el caso de que la Jacobiana tenga más filas que columnas, es decir, el número de ejes del robot sea inferior a la dimensión del espacio de la tarea, se tendrá que en las expresiones [2.29] y [2.30] habrá más ecuaciones que incógnitas, es decir, no existiera un vector de velocidades $\dot{q}$ que satisfaga simultáneamente todas estas las ecuaciones.

El vector $\dot{q}$ que consigue ajustar de la mejor manera posible (con un criterio matemático de mínimos cuadrados) todas las ecuaciones [2.29] y [2.30], se puede encontrar haciendo uso de la pseudoinversa por la izquierda, definida por:

$$J_i^+ = (J^T * J)^{-1} * J^T \qquad J_i^+ * J = \mathrm{I} \qquad [2.39]$$

Debe entenderse que, en este caso, las velocidades en el espacio de la tarea obtenidas de [2.29] y [2.30] no serán exactamente las deseadas, pero serán las que más se aproximan a ellas, dado lugar a un error cuadrático mínimo.

En el caso opuesto, esto es, si la matriz Jacobiana tiene más columnas que filas, se estará ante un robot redundante, con más ejes que la dimensión del espacio de la tarea. En este caso, habrá infinitos vectores de velocidades articulares $\dot{q}$ que satisfagan las expresiones [2.29] o [2.30] para unas velocidades definidas en el espacio de la tarea. De todas ellas, la que minimiza la norma del vector de velocidades $\dot{q}$, puede obtenerse mediante la pseudoinversa por la derecha, definida como:

$$J_i^+ = J^T * (J * J^T)^{-1} \qquad J_i^+ * J = \mathrm{I} \qquad [2.40]$$

Configuraciones singulares

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

- **Singularidades en los límites del espacio de trabajo del robot**. Se presentan cuando el extremo del robot está en algún punto del límite de trabajo interior o exterior. En esta situación, resulta obvio que el robot no podrá desplazarse en las direcciones que lo alejan de este espacio de trabajo.
- **Singularidades en el interior del espacio de trabajo del robot**. Ocurren dentro de la zona de trabajo y se producen, generalmente, por el alineamiento de dos o más ejes de las articulaciones del robot.

2.7. Dinámica del robot

La dinámica se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. Por tanto, el modelo dinámico de un robot tiene

por objetivo la relación entre el movimiento del robot y las fuerzas implicadas en el mismo.

Esta relación se obtiene mediante el denominado modelo dinámico, que establece la relación matemática entre:

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

La obtención de este modelo para mecanismos de uno o dos grados de libertad no es excesivamente compleja, pero a medida que el número de grados de libertad aumenta, el planteamiento y obtención del modelo dinámico se complica enormemente. Por este motivo, no siempre es posible obtener un modelo dinámico expresado de una forma cerrada, esto es, mediante una serie de ecuaciones, normalmente de tipo diferencial de segundo orden, cuya integración permita conocer qué movimiento surge al aplicar unas fuerzas o qué fuerzas hay que aplicar para obtener un movimiento determinado. El modelo dinámico debe ser resuelto entonces de manera iterativa, mediante la utilización de un procedimiento numérico.

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

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

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

Es importante hacer notar, que el modelo dinámico completo de un robot debe incluir no sólo la dinámica de sus elementos (barras o eslabones), sino también la propia de sus sistemas de transmisión, de los actuadores y sus equipos electrónicos de mando. Estos elementos incorporan al modelo dinámico nuevas inercias, rozamientos, saturaciones de los circuitos electrónicos, etc., aumentando aún más su complejidad.

Por último, es preciso señalar que si bien en la mayor parte de las aplicaciones reales de la robótica, las cargas e inercias manejadas no son suficientes como para originar deformaciones en los eslabones del robot, en determinadas ocasiones no ocurre así, siendo preciso considerar al robot como un conjunto de eslabones no rígidos. Aplicaciones de este tipo pueden encontrarse en la robótica espacial o en robots de grandes dimensiones, entre otras.

Este apartado presta atención a la obtención del modelo dinámico de robots de eslabones rígidos, debido a que la aplicación para la cual es diseñado el brazo robótico es netamente didáctica, por lo tanto, sólo se manipularán cargas ligeras, que no originen deformaciones en ninguna parte de la estructura mecánica.

El modelo dinámico de un robot se puede obtener a partir de leyes físicas conocidas, tales como las leyes de la mecánica newtoniana y lagrangiana. Esto conduce al desarrollo de las ecuaciones de movimiento dinámico para las diversas articulaciones del manipulador, en términos de los parámetros geométricos e inerciales de los elementos. Métodos convencionales como las formulaciones de Lagrange-Euler (L-E) y Newton-Euler (N-E) se pueden aplicar entonces sistemáticamente para desarrollar las ecuaciones de movimiento del robot. De estas dos formulaciones se obtienen diferentes formas de describir la dinámica del robot, tales como las ecuaciones de Lagrange-Euler de Uicker, las ecuaciones recursivas de Lagrange de Hollerbach, las ecuaciones de Newton-Euler de Luh y las ecuaciones generalizadas de D'Alembert y Lee. Estas ecuaciones de movimiento son equivalentes unas a otras, en el sentido de que describen la conducta dinámica del mismo robot físico. Sin embargo, sus estructuras pueden diferir porque se obtienen por diversas razones y objetivos. Algunas se obtienen para lograr tiempos de cálculo rápido, en la evaluación de los pares de las articulaciones nominales para controlar el manipulador; otras se obtienen para facilitar el análisis y la síntesis de control, y todavía otras se obtienen para mejorar la simulación en una computadora del movimiento del robot.

2.7.1. Formulación de Lagrange-Euler

Uicker en 1965 utilizó la representación de Denavit-Hartenberg basada en las matrices de transformación homogénea, para formular el modelo

dinámico de un robot mediante la ecuación de Lagrange, la cual originalmente establece:

$$L = E_c - E_p \qquad \tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} \qquad [2.41]$$

Donde:

L: Función Lagrangiana.
E_c: Energía Cinética.
E_p: Energía Potencial.
q_i: Coordenadas generalizadas (en este caso las articulares).
τ_i: Fuerza o pares aplicados sobre el grado de libertad q_i.

Este planteamiento utiliza, por tanto, las matrices ${}^{i-1}\mathrm{A}_i$ que relacionan el sistema de coordenadas de referencia del elemento $i-1$. Se realizan en este caso, operaciones de producto y suma innecesarias (recuérdese la información redundante contenida en las matrices ${}^{i-1}\mathrm{A}_i$ debido a la ortonormalidad de la submatriz de rotación), tratándose por ello de un procedimiento ineficiente desde el punto de vista computacional. Puede comprobarse que el algoritmo es de un orden de complejidad computacional $O(n^4)$, es decir, el número de operaciones a realizar crece con la potencia 4 del número de grados de libertad. Sin embargo, conduce a unas ecuaciones finales bien estructuradas, donde aparecen de manera clara todos los fenómenos físicos que se encuentran en la estructura mecánica del robot y que intervienen en su movimiento, tales como efectos inerciales, fuerzas centrípetas y de Coriolis, par gravitacional y fricción.

Se presenta a continuación, el algoritmo a seguir para obtener el modelo dinámico del robot por el método de Lagrange-Euler (L-E).

Algoritmo computacional de Lagrange-Euler para el modelado dinámico de un robot

L-E1 Asignar a cada eslabón un sistema de referencia de acuerdo a las normas de D-H.

L-E2 Obtener las matrices de transformación ${}^{0}\mathrm{A}_i$ para cada elemento i.

L-E3 Obtener las matrices $\mathbf{U}_{ij}$ definidas por:

$$\mathbf{U}_{ij} = \frac{\partial\, {}^{0}\mathbf{A}_i}{\partial q_j} \qquad \text{(véase nota 1)} \qquad [2.42]$$

L-E4 Obtener las matrices $\mathbf{U}_{ijk}$ definidas por:

$$\mathbf{U}_{ijk} = \frac{\partial \mathbf{U}_{ij}}{\partial q_k} \qquad \text{(veáse nota 2)} \qquad [2.43]$$

L-E5 Obtener las matrices de pseudoinercias $\mathbf{J}_i$ para cada elemento, que vienen definidas por:

$$\mathbf{J}_i = \begin{bmatrix} \int_i {x_i}^2 \mathrm{dm} & \int_i x_i y_i \mathrm{dm} & \int_i {x_i}^2 \mathrm{dm} & \int_i x_i \mathrm{dm} \\ \int_i y_i x_i \mathrm{dm} & \int_i {y_i}^2 \mathrm{dm} & \int_i {x_i}^2 \mathrm{dm} & \int_i y_i \mathrm{dm} \\ \int_i z_i x_i \mathrm{dm} & \int_i z_i y_i \mathrm{dm} & \int_i {x_i}^2 \mathrm{dm} & \int_i z_i \mathrm{dm} \\ \int_i x_i \mathrm{dm} & \int_i y_i \mathrm{dm} & \int_i z_i \mathrm{dm} & \int_i \mathrm{dm} \end{bmatrix} \qquad [2.44]$$

Donde las integrales están extendidas al elemento i considerado, y $(x_i\ y_i\ z_i)$ son las coordenadas del diferencial de masa dm respecto al sistema de coordenadas del elemento.

L-E6 Obtener la matriz de inercias $\mathbf{H} = [h_{ij}]$ cuyos elementos vienen definidos por:

$$h_{ij} = \sum_{k=(\max i,j)}^{n} \mathrm{Traza}\left(\mathbf{U}_{kj}\mathbf{J}_k{\mathbf{U}_{ki}}^T\right) \qquad [2.45]$$

Con $i, j = 1,2, \dots, n$, donde n: números de grados de libertad.

L-E7 Obtener los términos d_{ikm} definidos por:

$$d_{ikm} = \sum_{j=(\max i,k,m)}^{n} \mathrm{Traza}\left(\mathbf{U}_{jkm}\mathbf{J}_j{\mathbf{U}_{ji}}^T\right) \qquad [2.46]$$

Con $i, k, m = 1,2, \dots, n$

L-E8 Obtener la matriz columna de fuerzas de Coriolis y centrípeta $\mathbf{C} = [c_i]^T$ cuyos elementos vienen definidos por:

$$c_i = \sum_{k=1}^{n}\sum_{m=1}^{n} d_{ikm}\dot{q}_k\dot{q_m} \qquad [2.47]$$

L-E9 Obtener la matriz columna de fuerzas de gravedad $\mathbf{G} = [g_i]^T$ cuyos elementos están definidos por:

$$g_i = \sum_{j=1}^{n} \left(-m_j \mathrm{g} \mathbf{U}_{ji}\ {}^{j}\mathrm{r}_j\right) \qquad [2.48]$$

Con $i = 1,2,\ldots,n$, donde:

g: es el vector de gravedad expresado en el sistema de la base $\{S_0\}$ y expresado por $(\mathrm{g}_{x0}, \mathrm{g}_{y0}, \mathrm{g}_{z0}, 0)$.

${}^{j}\mathrm{r}_j$: es el vector de coordenadas homogéneas del centro de masas del elemento j expresado en el sistema de referencia del elemento i.

L-E10 La ecuación dinámica del sistema será:

$$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{G}(\boldsymbol{q}) \qquad [2.49]$$

Donde τ es el vector de fuerzas y pares motores efectivos sobre cada coordenada q_i.

Notas:

I. La derivada de la matriz de D-H ${}^{0}\mathrm{A}_i$ respecto de la coordenada q_j puede obtenerse fácilmente de manera computacional, mediante la expresión:

$$\frac{\partial\ {}^{0}\mathrm{A}_i}{\partial q_j} = \begin{Bmatrix} {}^{0}\mathbf{A}_{j-1}\mathbf{Q}_j\ {}^{j-1}\mathbf{A}_i & \text{si } j \le i \\ 0 & \text{si } j > i \end{Bmatrix} \qquad [2.50]$$

Con:

$$\mathbf{Q}_i = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{si la articulación } i \text{ es de rotación}$$

$$\mathbf{Q}_i = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{si la articulación } i \text{ es de traslación}$$

II. Análogamente:

$$\frac{\partial \mathbf{U}_{ij}}{\partial q_k} = \frac{\partial}{\partial q_k}\left(\frac{\partial\ {}^{0}\mathrm{A}_i}{\partial q_j}\right) \begin{Bmatrix} {}^{0}\mathbf{A}_{j-1}\mathbf{Q}_j\ {}^{j-1}\mathbf{A}_{k-1}\mathbf{Q}_k\ {}^{k-1}\mathbf{A}_i & \text{si } i \ge k \ge j \\ {}^{0}\mathbf{A}_{k-1}\mathbf{Q}_k\ {}^{k-1}\mathbf{A}_{j-1}\mathbf{Q}_j\ {}^{j-1}\mathbf{A}_i & \text{si } i \ge j \ge k \\ 0 & \text{si } i < j \text{ o } i < k \end{Bmatrix} \qquad [2.51]$$

III. Las matrices $\mathbf{J}_i$ y $\mathbf{H}$ son simétricas y semidefinidas positivas.

IV. El término d_{ikm} representa el efecto, en cuanto a fuerza o par, generado sobre el eslabón i como consecuencia del movimiento relativo entre los eslabones k y m. Se cumple que $d_{ikm} = d_{imk}$ y que $d_{iii} = 0$.

V. En la obtención de las matrices de pseudoinercia $\mathbf{J}_i$, las integrales están extendidas al elemento i, de modo que ésta se evalúa para cada punto del elemento de masa dm y coordenadas $(x_i\ y_i\ z_i)$ referidas al sistema de coordenadas del elemento.

VI. La expresión de la matriz de pseudoinercias $\mathbf{J}_i$ equivale a:

$$\mathbf{J}_i = \begin{bmatrix} \frac{1}{2}(-Ix_i + Iy_i + Iz_i) & Ix_i y_i & Ix_i z_i & \int_i m_i x_i \\ Ix_i y_i & \frac{1}{2}(Ix_i - Iy_i + Iz_i) & Iy_i z_i & \int_i m_i y_i \\ Ix_i z_i & Iy_i z_i & \frac{1}{2}(Ix_i + Iy_i - Iz_i) & \int_i m_i z_i \\ \int_i m_i x_i & \int_i m_i y_i & \int_i m_i z_i & \int_i m_i \end{bmatrix} \quad [2.52]$$

Con:

$$Ix_i = \int_i ({y_i}^2 + {z_i}^2)dm \quad Iy_i = \int_i ({x_i}^2 + {z_i}^2)dm \quad Iz_i = \int_i ({x_i}^2 + {y_i}^2)dm$$

Tabla 2.7. Complejidad computacional de las ecuaciones de movimiento de L-E.

Formulación de L-E	Multiplicaciones	Adiciones
${}^{j-1}\mathbf{A}_i$	$32n^*(n-1)$	$24n(n-1)$
$-m_j \mathbf{g}\mathbf{U}_{ji}\ {}^j\mathrm{r}_j$	$4n(9n-7)$	$n\frac{51n-45}{2}$
$\sum_{j=1}^{n}(-m_j \mathbf{g}\mathbf{U}_{ji}\ {}^j\mathrm{r}_j)$	0	${}^1/_2\, n(n-1)$
$\mathrm{Traza}(\mathbf{U}_{kj}\mathbf{J}_k\mathbf{U}_{ki}{}^T)$	${}^{128}/_3\, n(n+1)(n+2)$	${}^{65}/_2\, n(n+1)(n+2)$
$\sum_{k=(\max i,j)}^{n} \mathrm{Traza}(\mathbf{U}_{kj}\mathbf{J}_k\mathbf{U}_{ki}{}^T)$	0	${}^1/_6\, n(n-1)(n+1)$
$\mathrm{Traza}(\mathbf{U}_{jkm}\mathbf{J}_j\mathbf{U}_{ji}{}^T)$	${}^{128}/_3\, n^2(n+1)(n+2)$	${}^{65}/_2\, n^2(n+1)(n+2)$
$\sum_{j=(\max i,k,m)}^{n} \mathrm{Traza}(\mathbf{U}_{jkm}\mathbf{J}_j\mathbf{U}_{ji}{}^T)$	0	${}^1/_6\, n^2(n-1)(n+1)$
$\boldsymbol{\tau} = \mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{G}(\boldsymbol{q})$	${}^{128}/_3\, n^4 + {}^{512}/_3\, n^3 + {}^{844}/_3\, n^2 + {}^{76}/_3\, n$	${}^{98}/_3\, n^4 + {}^{781}/_6\, n^3 + {}^{637}/_3\, n^2 + {}^{107}/_6\, n$

Fuente: K. S. Fu, R. C. Gonzalez, C. S. G. Lee, "Robótica: Control, detección, visión e inteligencia", 1988.

$* n =$ números de GDL del brazo.

2.7.2. Formulación de Newton-Euler

La obtención del modelo dinámico de un robot a partir de la formulación Lagrangiana conduce a un algoritmo con un coste computacional de orden O(n^4). En el caso habitual de robots de 6 grados de libertad, este número de operaciones hace al algoritmo presentado en el epígrafe anterior materialmente inutilizable para ser utilizado en tiempo real. La formulación de Newton-Euler (N-E) parte del equilibrio de fuerzas o pares para cada elemento:

$$\sum F_i = \frac{d}{dt}(m_i v_i) = m_i \dot{v}_i$$
$$\sum T_i = \frac{d}{dt}(I_i \omega_i) = I_i \dot{\omega}_\iota + \omega_i \times (I_i \omega_i) \qquad [2.53]$$

Donde:

F_i: son las fuerzas ejercidas sobre la barra i.

T_i: son los pares ejercidos sobre la barra i en torno a su centro de masas.

m_i: es la masa de la barra i.

I_i: es el tensor de inercia de la barra i en torno a su centro de masas, expresado en el sistema de referencia $\{S_i\}$.

$v_i, \dot{v}_i$ son la velocidad y aceleración lineal del centro de masas de la articulación i.

$\omega_i, \dot{\omega}_\iota$ son la velocidad y aceleración angular de la articulación i.

Un adecuado desarrollo de estas ecuaciones, conduce a una formulación recursiva en la que se obtienen la posición, velocidad y aceleración del eslabón i referidos a la base del robot a partir de los correspondientes del eslabón $i-1$ y del movimiento relativo de la articulación i. De este modo, partiendo del eslabón 1 se llega el eslabón n. Con estos datos se procede a obtener las fuerzas y pares actuantes sobre el eslabón i, referidos a la base del robot, a partir de los correspondientes al eslabón $i+1$, recorriéndose de esta forma a todos los eslabones desde el eslabón n al eslabón 1.

El algoritmo se basa en operaciones vectoriales, siendo más eficiente en comparación con las operaciones matriciales asociadas con la formulación de L-E. De hecho, el orden de complejidad computacional de la formulación recursiva de N-E es O(n), lo que indica que depende directamente del número de grados de libertad. El algoritmo se desarrolla en los siguientes pasos.

Algoritmo computacional de Newton-Euler para el modelado dinámico de un robot

N-E1 Asignar a cada eslabón un sistema de referencia de acuerdo con las normas de D-H.

N-E2 Establecer las condiciones iniciales. Para el sistema de la base $\{S_0\}$:

$^0\boldsymbol{\omega}_0$: velocidad angular = $[0, 0, 0]^T$
$^0\dot{\boldsymbol{\omega}}_0$: aceleración angular = $[0, 0, 0]^T$
$^0\boldsymbol{v}_0$: velocidad lineal= $[0, 0, 0]^T$
$^0\dot{\boldsymbol{v}}_0$: aceleración lineal = $-[g_{x0}, g_{y0}, g_{z0}]^T$

$^0\boldsymbol{\omega}_0$, $^0\dot{\boldsymbol{\omega}}_0$, $^0\boldsymbol{v}_0$, son típicamente nulos salvo que la base del robot esté en movimiento.

$[g_{x0}, g_{y0}, g_{z0}]$ es el vector de gravedad expresado en el sistema $\{S_0\}$ (habitualmente toma el valor $[0, 0, -9.81]$ pues z_0 se sitúa vertical hacia arriba).

Para el extremo del robot se conocerá la fuerza y el par ejercidos externamente $^{n+1}\mathbf{f}_{n+1}$ y $^{n+1}\mathbf{n}_{n+1}$.

$\boldsymbol{z}_0 = [0, 0, 1]^T$
$^i\mathbf{p}_i =$ Vector que une el origen $\{S_{i-1}\}$ con el eje $\{S_i\}$ expresadas en $\{S_i\} = [a_i,\ d_i \sin\alpha_i,\ d_i \cos\alpha_i]$
$^i\mathbf{s}_i =$ Coordenadas del centro de masas del eslabón i respecto del sistema $\{S_i\}$.
$^i\mathbf{I}_i =$ Matriz de inercia del eslabón i expresado en un sistema paralelo al $\{S_i\}$ y con el origen en el centro de masas del eslabón.

N-E3 Obtener las matrices de rotación $^{i-1}\mathbf{R}_i$ y sus inversas $^i\mathbf{R}_{i-1}=\left(^{i-1}\mathbf{R}_i\right)^{-1}=\left(^{i-1}\mathbf{R}_i\right)^T$, siendo:

$$^{i-1}\mathbf{R}_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i \\ 0 & S\alpha_i & C\alpha_i \end{bmatrix} \quad [2.54]$$

Para $i = 1 \ldots n$ realizar los pasos 4 a 7:

N-E4 Obtener la velocidad angular del sistema $\{S_i\}$:

$$^i\boldsymbol{\omega}_i = \begin{Bmatrix} ^i\mathbf{R}_{i-1}\left(^{i-1}\boldsymbol{\omega}_{i-1} + \boldsymbol{z}_0\dot{\boldsymbol{q}}_i\right) & \text{si el eslabón } i \text{ es de rotación} \\ ^i\mathbf{R}_{i-1}\ ^{i-1}\boldsymbol{\omega}_{i-1} & \text{si el eslabón } i \text{ es de traslación} \end{Bmatrix} \quad [2.55]$$

N-E5 Obtener la aceleración angular del sistema $\{S_i\}$:

$$ {}^{i}\dot{\boldsymbol{\omega}}_i = \begin{Bmatrix} {}^{i}\mathbf{R}_{i-1}({}^{i-1}\dot{\boldsymbol{\omega}}_{i-1} + \boldsymbol{z}_0\ddot{\boldsymbol{q}}_i) + {}^{i-1}\boldsymbol{\omega}_{i-1} \times \boldsymbol{z}_0\dot{\boldsymbol{q}}_i & \text{si el eslabón } i \text{ es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\dot{\boldsymbol{\omega}}_{i-1} & \text{si el eslabón } i \text{ es de traslación} \end{Bmatrix} \quad [2.56] $$

N-E6 Obtener la aceleración lineal del sistema i:

$$ {}^{i}\dot{\boldsymbol{v}}_i = \begin{Bmatrix} {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{p}_i + {}^{i}\boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_i) + {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\dot{\boldsymbol{v}}_{i-1} & \text{si el eslabón } i \text{ es de rotación} \\ {}^{i}\mathbf{R}_{i-1}(\boldsymbol{z}_0\ddot{\boldsymbol{q}}_i + {}^{i-1}\dot{\boldsymbol{v}}_{i-1}) + {}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_i + & \\ 2\,{}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{R}_{i-1}\boldsymbol{z}_0\dot{q}_i + {}^{i}\boldsymbol{\omega}_i \times ({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{p}_i) & \text{si el eslabón } i \text{ es de traslación} \end{Bmatrix} \quad [2.57] $$

N-E7 Obtener la aceleración lineal del centro de gravedad del eslabón i:

$$ {}^{i}\mathbf{a}_i = {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{s}_i + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{s}_i\right) + {}^{i}\dot{\boldsymbol{v}}_i \quad [2.58] $$

Para $i = n \ldots 1$ realizar los pasos 8 a 10:

N-E8 Obtener la fuerza ejercida sobre el eslabón i:

$$ {}^{i}\mathbf{f}_i = {}^{i}\mathbf{R}_{i+1}\,{}^{i+1}\mathbf{f}_{i+1} + m_i\,{}^{i}\mathbf{a}_i \quad [2.59] $$

N-E9 Obtener el par ejercido sobre el eslabón i:

$$ {}^{i}\mathbf{n}_i = {}^{i}\mathbf{R}_{i+1}\left[{}^{i+1}\mathbf{n}_{i+1} + \left({}^{i+1}\mathbf{R}_i\,{}^{i}\mathbf{p}_i\right) \times {}^{i+1}\mathbf{f}_{i+1}\right] + \left({}^{i}\mathbf{p}_i + {}^{i}\mathbf{s}_i\right) \times m_i\,{}^{i}\mathbf{a}_i + {}^{i}\mathbf{I}_i\,{}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i}\boldsymbol{\omega}_i \times \left({}^{i}\mathbf{I}_i\,{}^{i}\boldsymbol{\omega}_i\right) \quad [2.60] $$

N-E10 Obtener la fuerza o par aplicado a la articulación i:

$$ \boldsymbol{\tau}_i = \begin{Bmatrix} {}^{i}\mathbf{n}_i^{T}\,{}^{i}\mathbf{R}_{i-1}\boldsymbol{z}_0 & \text{si el eslabón } i \text{ es de rotación} \\ {}^{i}\mathbf{f}_i^{T}\,{}^{i}\mathbf{R}_{i-1}\boldsymbol{z}_0 & \text{si el eslabón } i \text{ es de traslación} \end{Bmatrix} \quad [2.61] $$

Donde τ es el par o fuerza efectivo (par motor menos pares de rozamiento o perturbación).

Tabla 2.8. Complejidad computacional de las ecuaciones de movimiento de N-E.

Formulación de N-E	Multiplicaciones	Adiciones
${}^{i}\mathbf{R}_0\boldsymbol{\omega}_i$	$9n^*$	$7n$
${}^{i}\mathbf{R}_0\dot{\boldsymbol{\omega}}_i$	$9n$	$9n$
${}^{i}\mathbf{R}_0\dot{\boldsymbol{v}}_i$	$27n$	$22n$
${}^{i}\mathbf{R}_0\boldsymbol{a}_i$	$15n$	$14n$
${}^{i}\mathbf{R}_0\mathbf{F}_i$	$3n$	0
${}^{i}\mathbf{R}_0\mathbf{f}_i$	$9(n-1)$	$9n-6$
${}^{i}\mathbf{R}_0\mathbf{N}_i$	$24n$	$18n$
${}^{i}\mathbf{R}_0\mathbf{n}_i$	$21n-15$	$24n-15$
Total de operaciones matemáticas	$117n-24$	$103n-21$

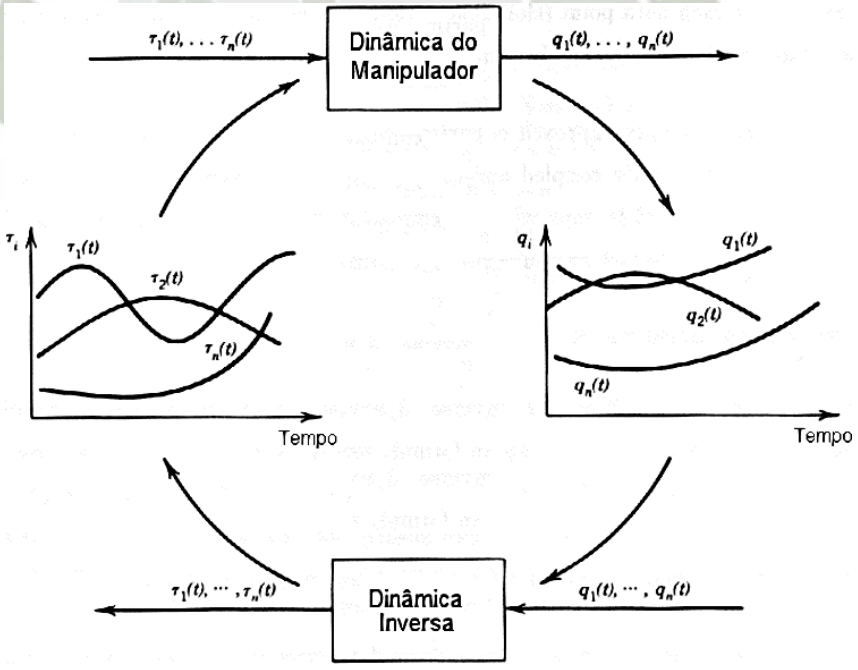
Fuente: K. S. Fu, R. C. Gonzalez, C. S. G. Lee, "Robótica: Control, detección, visión e inteligencia", 1988.

$* n =$ números de GDL del brazo.

2.7.3. Dinámica Inversa

Las ecuaciones dinámicas de forma cerrada derivadas en los apartados anteriores, gobiernan la respuesta dinámica de un brazo manipulador a través de los pares articulares de entrada generados por los actuadores. Este proceso dinámico puede ser ilustrado por el diagrama de bloques de la Fig. 2.30., donde las entradas son los pares articulares $\tau_1(t), \dots, \tau_n(t)$, y las salidas son coordenadas generalizadas, desplazamientos típicamente articulares $q_1(t), \dots, q_n(t)$. Como se ha expuesto en los epígrafes anteriores, los problemas inversos son importantes para el control y programación de robots, ya que permiten encontrar las entradas apropiadas y necesarias para producir las salidas o resultados deseados.

Los algoritmos computacionales analizados anteriormente, permiten modelar el comportamiento dinámico del robot con una buena aproximación a la realidad. Sin embargo, las ecuaciones de L-E son muy difíciles de utilizar con fines de control en tiempo real, a menos que se simplifiquen considerablemente, debido a que es necesario calcular los coeficientes dinámicos H, C y G para cada valor de q_i y $\dot{q}_i$, lo que lamentablemente requiere una extensa cantidad de operaciones aritméticas.



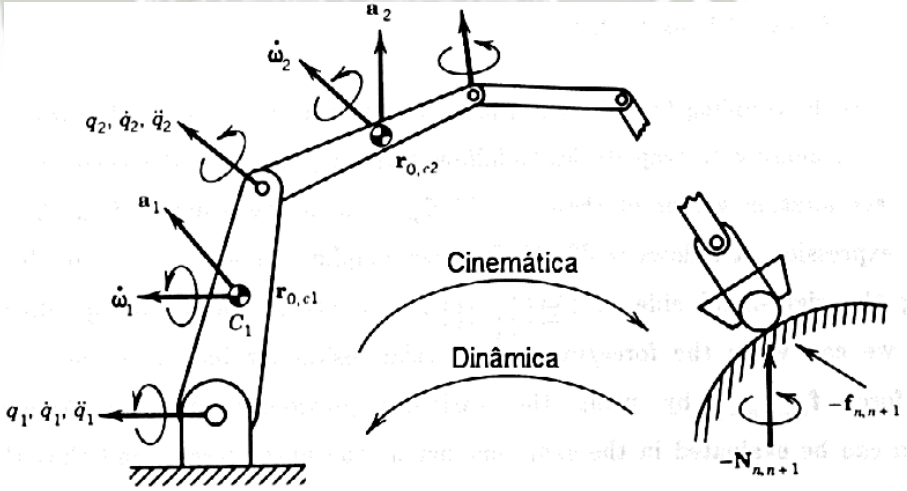
Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 2.30. Dinámica Inversa.

Como alternativa para derivar ecuaciones de movimientos más eficientes, se dirigió la atención a desarrollar algoritmos para calcular las fuerzas/pares

generalizados, basados en las ecuaciones de N-E. Uno de estos algoritmos es el de Luh-Walker-Paul, considerado el algoritmo recursivo más eficiente para robots manipuladores de hasta 6 GDL.

Como se verá a continuación, su obtención es simple, implica términos de producto vectorial (en lugar de matrices homogéneas 4x4) y sus ecuaciones dinámicas resultantes, excluyendo la dinámica del dispositivo de control, holguras y rozamiento de engranajes, son un conjunto de ecuaciones recursivas hacia adelante y hacia atrás. Este conjunto de ecuaciones se puede aplicar secuencialmente a los elementos del robot. La recursión hacia delante, propaga la información cinemática (tal como velocidades lineales, velocidades angulares, aceleraciones angulares y aceleraciones lineales del centro de masa de cada elemento) desde el sistema de coordenadas inercial hasta el sistema de coordenadas de la mano. La recursión hacia atrás, propaga las fuerzas y momentos ejercidos sobre cada elemento, desde el efector final del manipulador hasta el sistema de referencia de la base. El resultado más significativo de esta formulación, es que el tiempo de cálculo de las fuerzas/pares generalizados se encuentra que es linealmente proporcional al número de articulaciones del brazo e independientemente de la configuración del mismo. Con este algoritmo, se puede realizar el control en tiempo real simple del robot, en el espacio de las variables de articulación.



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 2.31. Cálculo recursivo de ecuaciones cinemáticas y dinámicas.

Algoritmo Luh-Walker-Paul

La primera fase de este algoritmo la constituyen los cálculos cinemáticos. Se derivan entonces, diferentes ecuaciones recursivas dependiendo del tipo de articulación (prismática o de revolución). Cuando la articulación $i+1$ es

prismática, la velocidad y aceleración angular del eslabón $i+1$ son las mismas que las del eslabón anterior:

$$\omega_{i+1} = \omega_i \qquad \dot{\omega}_{i+1} = \dot{\omega}_i \tag{2.62}$$

Por otro lado, si la articulación $i+1$ es de revolución, el marco de referencia $i+1$ se hace girar a una velocidad angular $\dot{q}_{i+1}b_i$ y con una aceleración angular $\ddot{q}_{i+1}b_i$ alrededor del eje z_i del marco de coordenadas de referencia unido al eslabón i . La velocidad angular del eslabón $i+1$ referido al marco de referencia de la base, está dada por:

$$\omega_{i+1} = \omega_i + \dot{q}_{i+1}b_i \tag{2.63}$$

Donde b_i es el vector unitario que apunta a lo largo de la dirección del eje de la articulación i.

La ecuación recursiva para la aceleración angular se puede obtener, simplemente tomando la derivada de tiempo de ambos lados. Se debe tener en cuenta, sin embargo, que el segundo término se define como un vector en relación con el marco de coordenadas en movimiento. Así, la aceleración angular queda expresada por:

$$\dot{\omega}_{i+1} = \dot{\omega}_i + \ddot{q}_{i+1}b_i + \omega_i \times \dot{q}_i b_i \tag{2.64}$$

Las ecuaciones recursivas para las velocidades y aceleraciones lineales, si la articulación $i+1$ es prismática, son:

$$v_{i+1} = v_i + \dot{q}_{i+1}b_i + \omega_i \times r_{i.i+1} \tag{2.65}$$

$$a_{i+1} = a_i + \ddot{q}_{i+1}b_i + \dot{\omega}_i \times r_{i.i+1} + 2\omega_i \times \dot{q}_{i+1}b_i + \omega_i \times (\omega_i \times r_{i.i+1}) \tag{2.66}$$

Donde $r_{i.i+1}$ es el vector de posición desde el origen del marco de referencia de la articulación i al origen del marco de referencia de la articulación $i+1$. Si la articulación $i+1$ es de revolución, se tiene:

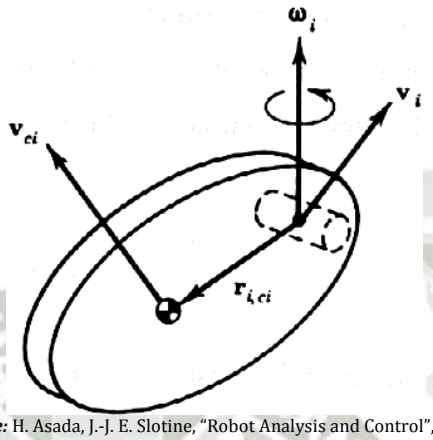
$$v_{i+1} = v_i + \omega_{i+1} \times r_{i.i+1} \tag{2.67}$$

$$a_{i+1} = a_i + \dot{\omega}_{i+1} \times r_{i.i+1} + \omega_{i+1} \times (\omega_{i+1} \times r_{i.i+1}) \tag{2.68}$$

Las ecuaciones de N-E se expresan en términos de aceleraciones centroidales, mientras que la formulación recursiva se expresa con respecto al origen del marco de coordenadas asociado a cada eslabón. Por lo tanto, es necesario transformar todas las variables a variables centroidales. Esto se ilustra en la Fig. 2.32., donde v_i y ω_i son, respectivamente, la velocidad en el origen del marco de coordenadas asociado al eslabón i, y la velocidad angular del mismo. La velocidad centroidal entonces, está dada por:

$$v_{ci} = v_i + \omega_i \times r_{i.ci} \qquad [2.69]$$

Donde $r_{i.ci}$ representa el vector de posición desde el origen del marco de referencia de la articulación i al centroide del eslabón i.



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 2.32. Velocidad centroidal y velocidad articular.

Del mismo modo, la aceleración centroidal está descrita por:

$$a_{ci} = a_i + \dot{\omega}_i \times r_{i.ci} + \omega_i \times (\omega_i \times r_{i.ci}) \qquad [2.70]$$

Finalmente, se discute el momento angular involucrado en la ecuación [2.73]. Como se mencionó antes, el tensor de inercia I_i varía dependiendo de la orientación del eslabón. Si ${}^0\mathrm{R}_\mathrm{i}$ es la matriz de rotación 3x3 asociada con la transformación de coordenadas del marco i de la base, y $\overline{I}_i$ es el tensor de inercia expresado en el marco de coordenadas fijo para el propio eslabón, el tensor de inercia I_i está dado por:

$$I_i = {}^0\mathrm{R}_\mathrm{i}\overline{I}_i\,{}^0\mathrm{R}_\mathrm{i}{}^T \qquad [2.71]$$

El tensor de inercia $\overline{I}_i$ es invariante, ya que sólo depende de la distribución de la masa del mismo eslabón y debe ser obtenido para cada configuración de brazo. Esto requiere tiempo de cálculo adicional. En el algoritmo de Luh-Walker-Paul, todas las variables y parámetros se expresan en el marco de coordenadas del eslabón, de manera que el cálculo adicional puede ser eliminado. Es decir, en lugar de representar vectores tales como v_i, ω_i y a_i con referencia al marco de coordenadas de la base, se expresa con referencia a las coordenadas del marco fijo para cada eslabón, es decir, en coordenadas de eslabón. Para expresar las ecuaciones en coordenadas de eslabón, simplemente se reemplaza v_i, ω_i y las otras variables, por las que se hace referencia a ese marco de coordenadas del eslabón. Además, cuando una variable se refiere al marco i que está implicado en una ecuación que

hace referencia a la marco $i+1$, se debe pre-multiplicar primero por la matriz de rotación ${}^{i-1}R_i$, de manera que todas las variables se expresan con referencia al marco $i+1$. En coordenadas de eslabón, los vectores b_i y $r_{i.ci}$ son constantes, ya que están fijados al cuerpo del eslabón. Además, si la articulación i es de revolución, el vector $r_{i.i+1}$ es constante también.

La segunda fase de este algoritmo la constituyen los cálculos dinámicos, que permiten evaluar los cálculos cinemáticos obtenidos para determinar los pares articulares. Ahora se procede con el cálculo recursivo en sentido contrario al realizado en la fase cinemática, es decir, a partir del último eslabón hacia atrás. Las relaciones de la fuerza de acoplamiento y par de acoplamiento entre eslabones adyacentes están dadas por:

$$f_{i-1,i} = f_{i,i+1} - m_i g + m_i a_{ci} \qquad [2.72]$$

$$N_{i-1,i} = N_{i,i+1} - r_{i.ci} \times f_{i,i+1} + r_{i-1.ci} \times f_{i-1,i} + I_i \dot{\omega}_\iota + \omega_i \times (I_i \omega_i) \qquad [2.73]$$

Las ecuaciones [2.72] y [2.73] gobiernan el comportamiento dinámico de un eslabón individual del robot. El conjunto completo de ecuaciones para todo el brazo manipulador se obtiene evaluando ambas ecuaciones para todos los eslabones.

Finalmente, una vez que la fuerza y el momento de acoplamiento de cada articulación han sido determinados, el par de torsión articular, si la articulación i es prismática, se puede calcular a partir de:

$$\tau_i = {b_{i-1}}^T \cdot f_{i-1,i} \qquad [2.74]$$

Si la articulación i es de revolución:

$$\tau_i = {b_{i-1}}^T \cdot N_{i-1,i} \qquad [2.75]$$

Un estudio completo de la obtención del algoritmo Luh-Walker-Paul y la demostración de sus ecuaciones recursivas, puede encontrarse en [5].

2.8. Control Dinámico del robot

Dadas las ecuaciones de movimiento dinámico de un manipulador, el objetivo del control dinámico del robot es mantener la respuesta dinámica del mismo de acuerdo con un criterio de funcionamiento preespecificado, por ejemplo, el seguimiento de una determinada trayectoria, que involucra características de velocidad y precisión. Aunque el problema de control puede parecer tan simple, su solución se complica por las fuerzas inerciales, las fuerzas de reacción en los

acoplos y la carga de la gravedad sobre los elementos, lo que hace imposible la coincidencia plena entre la trayectoria deseada y la real.

En general, el problema de control consiste en: 1) obtener modelos dinámicos del manipulador, y 2) utilizando estos modelos, determinar leyes o estrategias de control para conseguir la respuesta y funcionamiento deseado del sistema, es decir, procurar que las trayectorias realmente seguidas por el robot sean lo más parecidas posibles a las propuestas.

La primera parte fue estudiada anteriormente, por lo que el presente apartado se concentra en examinar las dos técnicas de control dinámico más utilizadas en robótica, cuyo carácter multiarticular permite considerar al robot como el sistema multivariable que realmente es. Las fases del control se dividen en dos: por un lado, el control de posición, en el que se utiliza la técnica de Control Proporcional-Integral-Derivativo (PID) con compensación de gravedad, y por el otro, el control de movimiento o trayectoria, donde la técnica de Control Torque Computado cobra protagonismo.

2.8.1. Control de posición

El control de posición o regulación de control de robots manipuladores, es un caso particular de control de movimiento, en el cual no hay una referencia variante en el tiempo que induzca a que el robot haga seguimiento, como en el caso de control de trayectoria. Por el contrario, es un punto constante en el tiempo, al que se le denomina posición deseada o *set point*. El objetivo de control es posicionar el extremo final del robot, desde cualquier posición inicial hacia una posición deseada, y que permanezca ahí de manera indefinida.

Por supuesto, para propósitos prácticos industriales, una vez que el extremo final del robot alcanza el punto deseado, deberá pasar uno o más periodos de muestreo para cambiar dicho punto; entonces, el actual punto deseado tomará el papel de condición inicial y el extremo final del robot se moverá nuevo punto deseado, y así sucesivamente.

Este concepto da la posibilidad de interpolar curvas, para que el robot pueda seguir la trayectoria a través de un esquema de control de posición con puntos cercanos entre sí. En control automático se le denomina control punto a punto. Para realizar esta aplicación, es necesario que el esquema de control que forma parte de la ecuación en lazo cerrado, genere un punto de equilibrio asintóticamente estable, ya que no dependería de las condiciones iniciales.

Es importante aclarar que en el control punto a punto, no se controla la velocidad de movimiento, como en el caso de control de trayectoria, donde el error de posición y el de velocidad son controlados simultáneamente.

Formalmente, la misión del control de posición es encontrar una ley de control τ que proporcione los pares aplicados a las articulaciones, de tal forma que la posición actual del robot $q(t)$ y la velocidad articular de movimiento $\dot{q}(t)$, tiendan asintóticamente hacia la posición deseada q_d y velocidad cero, respectivamente, sin importar las condiciones iniciales.

Control PID con compensación de gravedad

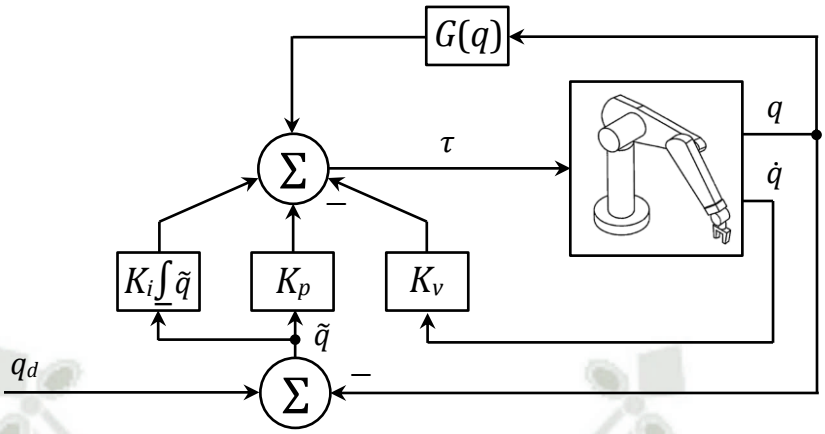
El control de posición pura de robots manipuladores puede realizarse potencialmente mediante la técnica de control PID. Sin embargo, dicho esquema de control posee ciertas restricciones que pueden limitar su uso. Efectivamente, el controlador PID garantiza el cumplimiento del objetivo de control de posición en forma global para robots cuyos modelos dinámicos no poseen el vector de pares gravitacionales $\mathbf{G}(\boldsymbol{q})$. En este caso, la sintonía de este controlador es trivial, ya que es suficiente con seleccionar las matrices de diseño K_p, K_i y K_v como simétricas y definidas positivas. No obstante, el control PID no garantiza el cumplimiento del objetivo de control de posición pura de manipuladores cuyos modelos dinámicos contienen el término de pares gravitacionales $\mathbf{G}(\boldsymbol{q})$, a menos que la posición deseada q_d sea tal que $\mathrm{G}(q_d) = 0$.

El control PID con compensación de gravedad es capaz de satisfacer el objetivo de control de posición pura en forma global para robots de n GDL. En la ley de control se requiere el conocimiento previo de una parte del modelo dinámico del robot a ser controlado, puesto que utiliza el vector de pares gravitacionales $\mathbf{G}(\boldsymbol{q})$.

La ley de control PID con compensación de gravedad está representada por la siguiente ecuación:

$$\boldsymbol{\tau} = \boldsymbol{K_p}\tilde{\boldsymbol{q}} + \boldsymbol{K_v}\dot{\boldsymbol{q}} + \boldsymbol{K_i}\int_0^t \tilde{\boldsymbol{q}}(\boldsymbol{t})\boldsymbol{dt} + \mathbf{G}(\boldsymbol{q}) \qquad [2.76]$$

Donde K_p, K_i y $K_v \in \mathbb{R}^{n\times n}$ son las matrices simétricas definidas positivas de las ganancias proporcional, integral y derivativa, respectivamente, y $\tilde{q} \in \mathbb{R}^n$ es el vector de error de posicionamiento. La Fig. 2.33. muestra el diagrama de bloques correspondiente al control PID con compensación de gravedad de robots.



Fuente propia.

Fig. 2.33. Diagrama de bloques del Control PID con compensación de gravedad.

Nótese que la única variación respecto al control PID tradicional es el término aditivo G(q). A diferencia del control PID, que no requiere conocimiento alguno sobre la estructura del modelo del robot, el controlador de la ecuación [2.76] hace uso explícito del conocimiento parcial del modelo del manipulador, específicamente de G(q). Sin embargo, es importante observar que para un robot dado, el vector de pares gravitacionales G(q) puede obtenerse con relativa facilidad, puesto que para este fin basta con calcular sólo la expresión correspondiente a la energía potencial del robot. La velocidad de movimiento $\dot{q}$ se emplea para inyección de amortiguamiento. Obsérvese que el signo menos de la acción de control derivativa sirve para contrarrestar la energía al control proporcional.

La ley de control [2.76] requiere información sobre la posición deseada q_d, así como medición de la posición q a cada instante.[8]

2.8.2. Control de movimiento

El control de movimiento o control de trayectoria, consiste en determinar los pares aplicados a los actuadores que forman las articulaciones, de tal forma que las posiciones $q(t)$ y velocidades $\dot{q}(t)$ asociadas a las coordenadas articulares del robot, sigan con exactitud a la posición y velocidad deseadas $q_d(t)$ y $\dot{q}_d(t)$ respectivamente, variantes en el tiempo.

En contraste con el algoritmo de control de posición, el cual se define en términos del error de posición $\tilde{q}$ y la velocidad articular $\dot{q}$ (ésta última para propósitos de inyección de amortiguamiento), y que únicamente requiere el conocimiento parcial de la dinámica del robot, los algoritmos del control de

[8] Kelly, R., y Santibáñez, V., "Control de Movimiento de Robots Manipuladores", pp. 143-149, 2003.

trayectoria incluyen el control del error de posición $\tilde{q}$ y el de velocidad $\dot{\tilde{q}}$, además de la dinámica completa del robot manipulador en la estructura matemática del controlador, es decir, se basan en el modelo dinámico del robot. La exactitud, desempeño y robustez de esos controladores, dependen del grado de precisión con que se conozcan los parámetros dinámicos que describen el modelo.

Control Torque Computado

El modelo dinámico que caracteriza el comportamiento de los robots manipuladores es generalmente no lineal en términos de las variables de estado (posiciones y velocidades articulares). Esta peculiaridad del modelo dinámico podría hacer pensar que dado cualquier controlador, la ecuación diferencial que gobierna al sistema de control en malla cerrada debería ser también no lineal en las variables de estado correspondientes. Esta idea intuitiva se confirma para el resto de controladores clásicos utilizados en robótica.

Por otro lado, el controlador para posición presentado en el apartado anterior, tiene en su ley de control la presencia explícita de un controlador lineal del tipo PID. Siendo la excepción a esta regla, en este epígrafe se estudia un controlador de movimiento cuya ley de control no presenta explícitamente el término lineal PID. Específicamente se trata del Control Torque Computado o Par Calculado.

Otras peculiaridades de este controlador es que emplea la dinámica de compensación en el lazo de realimentación, para linealizar y desacoplar la dinámica no lineal del robot, lo que permite obtener una ecuación de malla cerrada lineal en términos de las variables de estado. Este hecho no tiene precedentes en el estudio de controladores típicos que se aborda en las asignaturas de control. Además, este tipo de controlador satisface el objetivo de control de movimiento con una selección adecuada de sus parámetros de diseño. La ecuación correspondiente al control Torque Computado viene dada por:

$$\boldsymbol{\tau = H(q)\left[\ddot{q}_d + K_p\tilde{q} + K_v\dot{\tilde{q}}\right] + C(q,\dot{q})\dot{q} + G(q)} \qquad [2.77]$$

Donde $\dot{\tilde{q}} \in \mathbb{R}^n$ es el vector de error de velocidad.

A pesar de la presencia del término $K_p\tilde{q} + K_v\dot{\tilde{q}}$ en la ley de control [2.77], éstos son en realidad multiplicados por la matriz de inercia $\mathrm{H}(q_d - \tilde{q})$. Este hecho tiene como resultado que, efectivamente la ley de control cuente con

un término del tipo PD, pero éste no es un controlador lineal, ya que las ganancias de posición y de velocidad no son constantes, sino que dependen explícitamente del error de posición $\tilde{q}$. Esto puede verse claramente, expresando la ley de control Torque Computado de la siguiente manera:

$$\boldsymbol{\tau = H(q_d - \tilde{q})K_p\tilde{q} + H(q_d - \tilde{q})K_v\dot{\tilde{q}} + H(q_d - \tilde{q})\ddot{q}_d + C(q, \dot{q})\dot{q} + G(q)}$$

El control Torque Computado fue una de las primeras estructuras de control de movimiento basadas en el modelo del manipulador a ser controlado, es decir, que hace uso explícito del conocimiento de las matrices $H(q)$, $C(q, \dot{q})$ y $G(q)$. Obsérvese que la trayectoria de movimiento deseada: $q_d(t)$, $\dot{q}_d(t)$ y $\ddot{q}_d$, así como la medición de la posición $q(t)$ y de la velocidad $\dot{q}(t)$, se emplean para realizar el cálculo de la acción de control. El diagrama de bloques correspondiente al control Torque Computado de robots manipuladores se puede ver en la Fig. 2.34.

La ecuación de malla cerrada se obtiene sustituyendo la acción de control τ de la ley de control [2.77] en la ecuación del modelo del robot:

$$\boldsymbol{H(q)\ddot{q} = H(q)\left[\ddot{q}_d + K_p\tilde{q} + K_v\dot{\tilde{q}}\right]} \qquad [2.78]$$

Dado que $H(q)$ es una matriz definida positiva y por lo tanto también invertible, y asumiendo que existe una cancelación exacta del modelo dinámico que interviene en el lazo de realimentación, entonces el control Torque Computado realiza el desacoplamiento dinámico en todas las articulaciones del robot, resultando el siguiente sistema lineal:

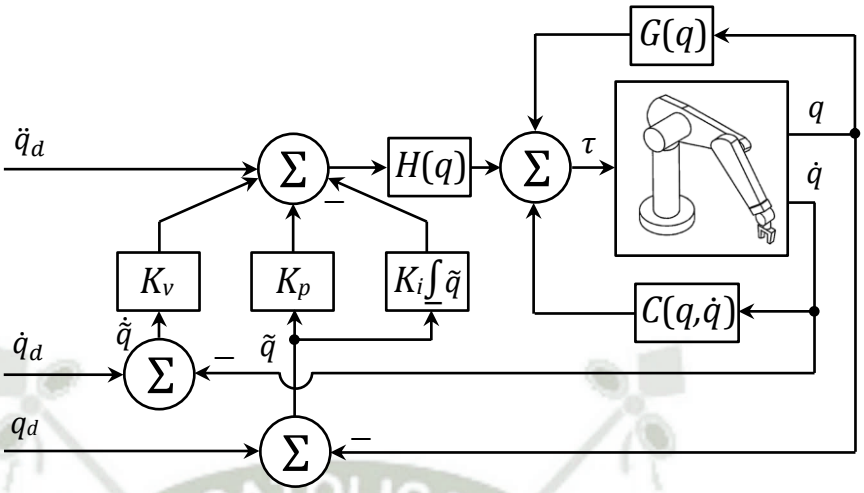
$$\boldsymbol{\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} = 0} \qquad [2.79]$$

El cual puede escribirse en términos del vector de estado $\left[\tilde{q}^T\ \dot{\tilde{q}}^T\right]^T \in \mathbb{R}^{2n}$ de la siguiente forma:

$$\frac{d}{dt}\begin{bmatrix}\tilde{q}\\ \dot{\tilde{q}}\end{bmatrix} = \begin{bmatrix}\dot{\tilde{q}}\\ -K_p\tilde{q} - K_v\dot{\tilde{q}}\end{bmatrix} = \begin{bmatrix}0 & I\\ -K_p & -K_v\end{bmatrix}\begin{bmatrix}\tilde{q}\\ \dot{\tilde{q}}\end{bmatrix} \qquad [2.80]$$

Donde $I \in \mathbb{R}^{n\times n}$ es la matriz identidad.

Es importante observar que la ecuación de malla cerrada [2.80] representa una ecuación diferencial lineal y autónoma, cuyo único estado de equilibrio es $\left[\tilde{q}^T\ \dot{\tilde{q}}^T\right]^T = 0 \in \mathbb{R}^{2n}$. La unicidad del equilibrio se debe a que la matriz K_p es por diseño definida positiva y por lo tanto no singular. Como la ecuación [2.80] es lineal y autónoma, las soluciones de ésta pueden obtenerse en forma cerrada.



Fuente propia.

Fig. 2.34. Diagrama de bloques del Control Torque Computado.

2.9. Sensor Kinect

Kinect es un sensor 3D de movimiento desarrollado por Microsoft®, fruto de veinte largos años de investigación para la creación de nuevas tecnologías de información y entretenimiento.

Inicialmente fue creado para la consola de videojuegos Xbox 360, cuyo lanzamiento oficial se dio en noviembre del 2010. Sin embargo, debido a su gran acogida[9], y como parte de la iniciativa del gigante informático por ampliar las fronteras del Kinect más allá de los salones de entretenimiento, en febrero del 2012 se lanza al mercado mundial una segunda versión de este periférico para Windows (Fig. 2.35.), con mejoradas prestaciones que su antecesor, y exclusivamente orientado para aplicaciones en PC. Ésta versión del Kinect es la que se utiliza en la presente investigación.

Fuente: "Con Kinect, Tú eres el mando", AnaSoft Solutions, 2011.

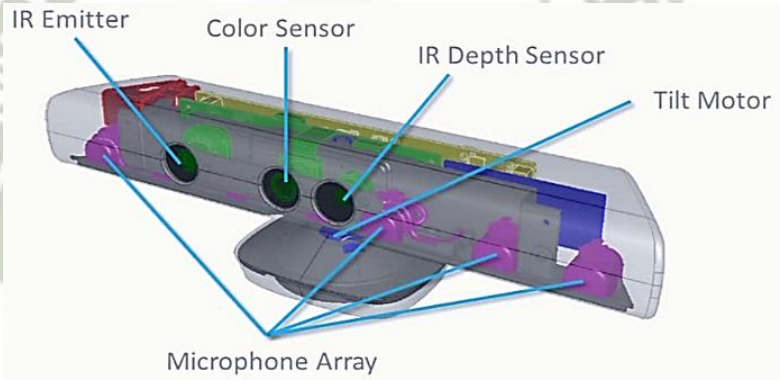
Fig. 2.35. Sensor Kinect para Windows.

[9] En enero del 2011, Microsoft® publicó las cifras de ventas de Kinect y estas ascendían a más de 8.5 millones de unidades, lo que sirvió para reclamar el Récord Mundial Guinness de ser el "dispositivo de electrónica de consumo de venta más rápida". 24 millones de unidades del sensor Kinect se habían enviado en enero de 2012.

En junio del 2011, Microsoft® anuncia el lanzamiento del Kit de Desarrollo de Software (SDK) de Kinect para Windows. A principios del 2012, Microsoft envía la versión comercial del SDK, que permite a los desarrolladores de todo el mundo, utilizar el poder del sensor Kinect en sus propias aplicaciones bajo diversos lenguajes de programación tales como C, C++, C# o Visual Basic .NET. Unos meses después, en junio del 2012, el Microsoft Kinect SDK v1.5 se encuentra disponible y listo para ser explorado.

El Kinect alberga un conjunto de sensores, dispositivos y circuitos embebidos conectados a una base motorizada. La Fig. 2.36. muestra los componentes internos principales de este periférico, los cuales son:

- cámara RGB.
- emisor de luz infrarroja (IR).
- sensor de profundidad IR.
- matriz de micrófonos.
- motor de inclinación.



Fuente: "Inside the Newest Kinect for Windows SDK—Infrared Control", KINECT for Windows Blog, 2012.

Fig. 2.36. Componentes internos del sensor Kinect.

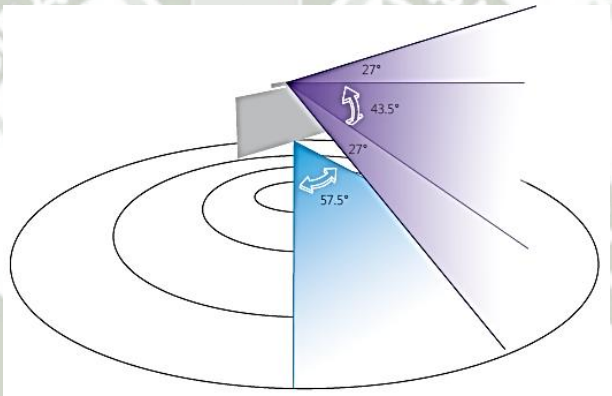
Además de los componentes mencionados anteriormente, el dispositivo Kinect también tiene un adaptador de corriente para la fuente de alimentación externa (el puerto USB no puede aportar directamente el consumo de energía que el sensor necesita), un adaptador de puerto USB 2.0 para comunicarse con la PC, un LED de color verde, situado entre la cámara RBG y el emisor de luz IR, que indica si la comunicación Kinect-PC ha sido exitosa, y un procesador digital de señal (DSP), que se utiliza para procesar la señal de los micrófonos.

Los controladores, software y el SDK del Kinect se pueden instalar en Windows 7, 8 y 8.1, con un procesador de 32 o 64 bits. Se necesita un mínimo de 2 GB de RAM, 190 MB de espacio libre en el disco duro y un procesador de doble núcleo con una velocidad de 2,66 GHz o más, para un procesamiento de señales y datos exitoso con el sensor.

Cámara RGB

La cámara RGB es similar a una cámara web de color común, pero a diferencia de ésta, la cámara RGB no tiene un filtro de corte IR. Esta cámara es responsable de la captura y transmisión de los datos de video a color. Su función es detectar los colores rojo, azul y verde de la fuente. La corriente de los datos devueltos por la cámara es una sucesión de cuadros de imágenes fijas. La corriente de color del Kinect es compatible con una velocidad de 30 fotogramas por segundo (FPS) a una resolución de 640 x 480 píxeles, y una resolución máxima de 1280 x 960 píxeles de hasta 12 FPS. El valor de fotogramas por segundo puede variar dependiendo de la resolución utilizada para el marco de imagen.

El rango visible para la cámara del Kinect es de 43.5 grados verticales por 57.5 grados horizontales. La Fig. 2.37. es un ejemplo del campo visible de la cámara Kinect.



Fuente: "Human Interface Guidelines", Microsoft Corporation, 2012.

Fig. 2.37. Campo visible del sensor Kinect.

La siguiente figura muestra una imagen de color que se capturó utilizando la cámara RGB del Kinect, con una resolución de 640 x 480 píxeles:

Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.

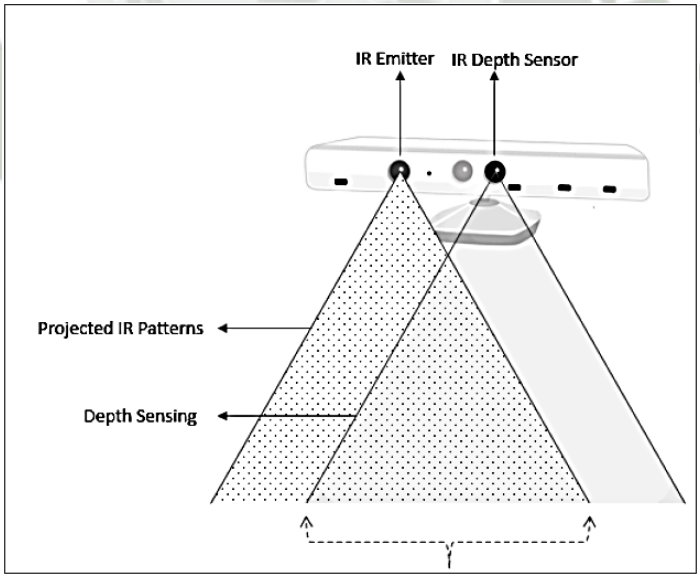
Fig. 2.38. Imagen de color capturada con la cámara RGB del Kinect.

Esta cámara RGB es capaz de realizar acciones que evitan el parpadeo de color, operaciones de saturación de color y equilibrio de blancos automático.

Emisor de luz infrarroja (IR)

El sistema sensorizado de profundidad del Kinect, consiste en un emisor de luz infrarroja y un sensor de profundidad IR. Ambos trabajan juntos para el cálculo de los datos de profundidad. El emisor de infrarrojos puede parecer una cámara desde el exterior, pero es un proyector de infrarrojos que emite constantemente luz infrarroja en un patrón de "puntos pseudo-aleatorios", estructurados en una longitud de onda alrededor de 830 nm, sobre todo lo que esté en frente de él. Estos puntos son normalmente invisibles para el ser humano (que por lo general responden a longitudes de onda de aproximadamente 390 nm a 750 nm) a excepción de un brillante punto rojo en el centro del emisor, pero es posible capturar su información de profundidad utilizando un sensor de profundidad IR.

El patrón se compone de 3 x 3 sub-patrones de 211 x 165 puntos (para un total de 633 x 495 puntos). En cada sub-patrón, un punto es mucho más brillante que todos los demás. La luz de puntos se refleja en diferentes objetos, el patrón se distorsiona y el sensor de profundidad IR lee los patrones de los objetos y los convierte en información de profundidad mediante la medición de la distancia entre el sensor y el objeto desde donde se leyó el punto IR. En la Fig. 2.39. se grafica cómo se ve la detección global de profundidad.



Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.

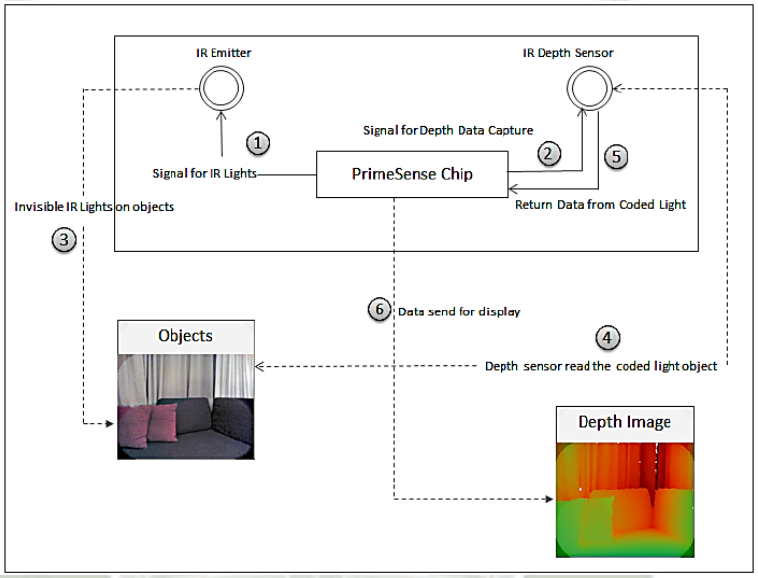
Fig. 2.39. Detección global de profundidad.

Sensor de profundidad IR

El sensor de profundidad es un sensor CMOS (semiconductor de óxido-metálico complementario) monocromo de infrarrojos, que no es más que un circuito integrado que contiene una matriz de fotodetectores que actúan como un sensor de imagen de infrarrojos. Este dispositivo también se conoce como cámara de

infrarrojos, sensor de IR, CMOS de imagen de profundidad, o sensor CMOS, dependiendo de la fuente. Está equipado con un filtro *IR-pass* (que bloquea la luz visible). Éste sensor es el encargado de capturar los datos de profundidad, que devuelven las coordenadas 3D (x, y, z) de la escena como una corriente.

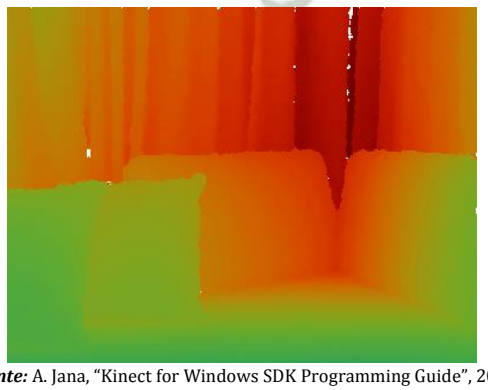
El sensor de IR capta la luz estructurada emitida por el proyector IR y la luz reflejada desde los objetos dentro de la escena. Todos estos datos se convierten en una corriente de tramas. Cada trama individual es procesada por el chip PrimeSense del Kinect, que produce un flujo de salida de fotogramas. El diagrama de la Fig. 2.40. grafica como funciona esto.



Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.

Fig. 2.40. Proceso de captura de datos de profundidad.

El flujo de datos de profundidad permite una resolución de 640 x 480 píxeles, 320 x 240 píxeles, y 80 x 60 píxeles. Cada pixel, basado en 11 bits, puede representar 2048 niveles de profundidad. El rango visible del sensor sigue siendo el mismo que la cámara RGB. La siguiente figura muestra la corriente de imágenes de profundidad que se capturan desde el sensor Kinect.



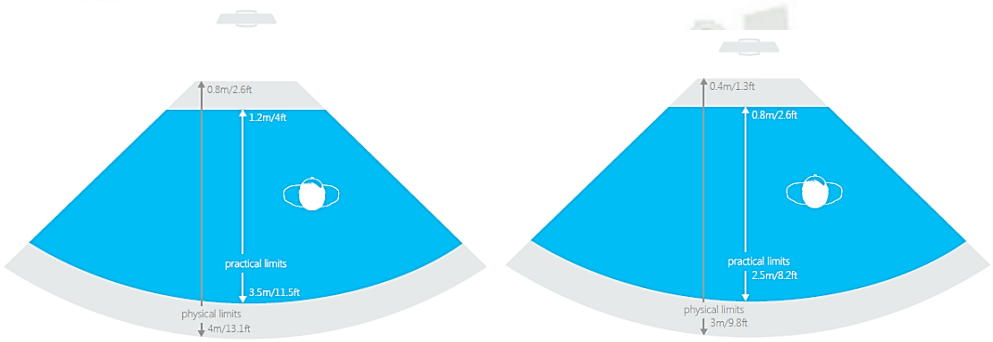
Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.

Fig. 2.41. Imagen de profundidad capturada con del Kinect.

La Tabla 2.9. muestra los rangos de distancia de profundidad para los dos modos de trabajo del sensor: el modo normal y el modo cercano, mientras que la Fig. 2.42. los grafica en el espacio de trabajo.

Tabla 2.9. Rangos de distancia de profundidad.

Modo	Límites físicos	Límites prácticos
Normal	0.8 a 4 m (2.6 a 13.1 ft)	1.2 a 3.5 m (4 a 11.5 ft)
Cercano	0.4 a 3 m (1.3 a 9.8 ft)	0.8 a 2.5 m (2.6 a 8.2 ft)



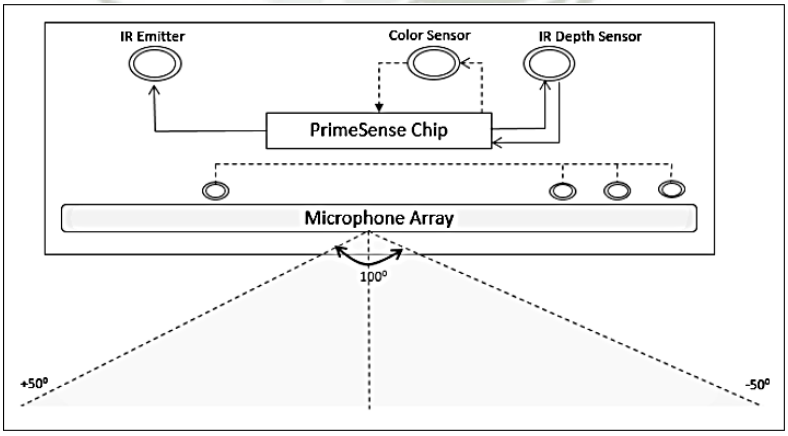
Fuente: "Human Interface Guidelines", Microsoft Corporation, 2012.

Modo Normal **Modo Cercano**

Fig. 2.42. Rangos de distancia para ambos modos.

Matriz de micrófonos

La matriz de micrófonos del sensor de Kinect consta de cuatro micrófonos que se encuentran en un patrón lineal en la parte inferior del dispositivo, con un convertidor analógico-digital (ADC) de 24-bits. El audio capturado se codifica mediante *Pulse Code Modulation* (PCM) con una velocidad de muestreo de 16 KHz y una profundidad de 16 bits.



Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.

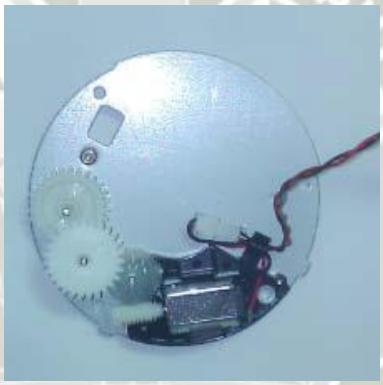
Fig. 2.43. Rango de captación de la fuente acústica.

El propósito de la matriz de micrófonos no es sólo para permitir que el Kinect capture el sonido, sino también para determinar la ubicación y dirección de la

onda de audio. Las principales ventajas de tener una matriz de micrófonos sobre un solo micrófono son que la captura y el reconocimiento de la voz se hacen más eficaces con la supresión mejorada del ruido ambiental, cancelación de eco (AEC), y la tecnología de formación de haz. Esto le permite al Kinect ser un micrófono de alta bidireccional que puede identificar la fuente del sonido y reconocer la voz independientemente del ruido y el eco presente en el medio ambiente.

Motor de inclinación

La base y el cuerpo del sensor están conectados por un motor pequeño. Éste es usado para ajustar la inclinación vertical del sensor, con la finalidad de aumentar el espacio de interacción. De ésta manera, el motor de inclinación puede cambiar el ángulo de la cabeza del Kinect hacia arriba o hacia abajo en 27 grados, respectivamente (Fig. 2.37.). La Fig. 2.44. muestra el motor junto con tres engranajes que permiten que el sensor se incline gradualmente.



Fuente: A. Jana, "Kinect for Windows SDK Programming Guide", 2012.
Fig. 2.44. Motor de inclinación del Kinect.

El Kinect también contiene un acelerómetro de tres ejes configurado para un rango de $2g$ (g es la aceleración de la gravedad) con una precisión de 1 a 3 grados. Es posible conocer la orientación del dispositivo con respecto a la gravedad de leer los datos del acelerómetro.

Así, mediante la combinación de la salida de todos estos sensores, un programa basado en el SDK del Kinect puede capturar el movimiento de todo el cuerpo humano en 3D, realizar un reconocimiento facial, y proporcionar capacidades de reconocimiento de voz. En otras palabras, les permite a los usuarios controlar e interactuar con una situación determinada, sin la necesidad de tocar un dispositivo físico, a través de una interfaz natural de usuario usando gestos y comandos de voz.

La capacidad del sensor de Kinect para registrar la información 3D es increíble, pero a medida que usted vaya explorando esta investigación, descubrirá que es mucho más que una cámara 3D.

2.10. Interfaces Naturales de Usuario

Una interfaz de usuario es el medio a través del cual el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar, aunque en el ámbito de la informática es preferible referirse a que suelen ser "amigables e intuitivos" pues es muy complejo y subjetivo decir que algo es "fácil".

La mejor interacción humano-máquina a través de una adecuada interfaz de usuario, es aquella que le brinde tanto comodidad como eficiencia. Según la forma de interactuar del usuario, se pueden distinguir los siguientes tipos:

- **Interfaces de línea de comandos (CLI)** o alfanuméricas que solo presentan texto simple.
- **Interfaces gráficas de usuario (GUI)**, las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.
- **Interfaces táctiles de usuario (TUI)**, que representan gráficamente un panel de control en una pantalla sensible, que permite interactuar con el dedo de forma similar a si se accionara un control físico.

Sin embargo**,** debido al avance tecnológico de los últimos años, ha surgido una nueva tendencia informática conocida como Informática de Control de Movimiento, que es la disciplina que procesa, digitaliza, y detecta la posición y/o velocidad de personas y objetos con el fin de interactuar con los sistemas de software. Ésta disciplina se viene consolidando como una de las técnicas más relevantes para el diseño e implementación de la nueva categoría de interfaces: la **Interfaz de Usuario Natural (NUI)**, caracterizada por brindar una experiencia más inmersiva e intuitiva para el usuario.

Una NUI es aquella en la que se interactúa con un sistema, aplicación, etc. sin utilizar sistemas de mando o dispositivos de entrada de las GUI (como por ejemplo un ratón, teclado, lápiz óptico, pantalla táctil y joystick); en lugar de ello, se hace uso de movimientos gestuales (con las manos o el rostro) corporales y/o la voz (interacción multimodal), situando el cuerpo del usuario como el propio mando de control.

La Fig. 2.45. es un ejemplo de una NUI, donde se puede apreciar a un cirujano durante una delicada operación, que se encuentra analizando imágenes radiológicas del cerebro de un paciente, sin tener contacto con ningún objeto para no contaminar sus guantes quirúrgicos.

Fuente: "Kinect de Xbox en tu ordenador y en tu clínica", Blog de rehabilitación que mira al futuro, 2011.

Fig. 2.45. NUI en un quirófano de neurología.

Para que una interfaz sea considerada natural, debe cumplir con dos principios fundamentales:

- La NUI tiene que ser imperceptible, gracias a sus características intuitivas: (un sensor capaz de capturar los gestos, un micrófono capaz de captar la voz, y una pantalla táctil o gráfica capaz de mostrar el movimiento de las manos). Todas estas interfaces son imperceptibles para el ser humano, ya que su uso es intuitivo. La interfaz no distrae al usuario de las funcionalidades básicas del sistema de software.
- La NUI se basa en la naturaleza o en elementos naturales (el gesto de deslizamiento, el tacto, los movimientos del cuerpo, los comandos de voz, todas estas acciones son naturales y no se desvían del comportamiento normal de una persona).

Las NUIs se están convirtiendo en elementos cruciales para el aumento y la mejora de la accesibilidad de los usuarios para la solución de software. Programar una NUI es muy importante hoy en día y seguirá evolucionando en el futuro, porque crea una experiencia que da la sensación de ser una extensión del cuerpo, natural tanto a usuarios expertos como a los usuarios nuevos.

Kinect abraza el principio de una NUI y proporciona una potente interfaz multimodal para el usuario. Es posible interactuar con aplicaciones de software complejas y/o en los videojuegos simplemente usando la voz y los gestos naturales. El Kinect puede detectar la posición del cuerpo humano, la velocidad de sus movimientos y los comandos de voz. También, es capaz de detectar la posición de objetos.

El contenido de este capítulo se ha obtenido de las referencias citadas al final de la tesis. El lector interesado en profundizar en el material está invitado a consultarlas.

Capítulo III

Diseño del brazo robótico

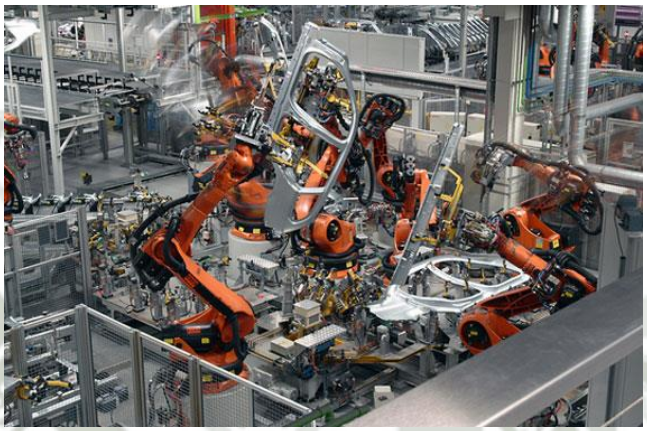
3.1. El sentido del diseño

Los robots son exigidos a tener mucha mayor movilidad y destreza que las máquinas herramientas tradicionales. Ellos deben ser capaces de trabajar en un gran rango alcanzable, acceder a lugares con poca facilidad de movimiento, manejar una variedad de piezas de trabajo y realizar tareas flexibles. La estructura mecánica única del robot, que es paralela a la del brazo humano, resulta de estos requisitos de alta movilidad y destreza. Esta estructura, sin embargo, se aparta significativamente del diseño de las máquinas tradicionales. Una estructura mecánica del robot se compone básicamente de eslabones, formando una secuencia de enlaces de brazos conectados por articulaciones. Tal estructura tiene inherentemente pobre rigidez y la precisión mecánica, por lo tanto, no es apropiada para el trabajo pesado ni para aplicaciones de alta precisión requeridas en máquinas herramientas. Además, también implica una secuencia en serie de las articulaciones con servomotores, cuyos errores se acumulan a lo largo de los acoplamientos. Con el fin de aprovechar la gran movilidad y destreza ofrecida únicamente por el acoplamiento en serie, estas dificultades deben ser superadas por medio de técnicas de diseño y control avanzados.

La geometría de acoplamiento en serie de los brazos manipuladores está descrita por ecuaciones no lineales complejas. Herramientas analíticas eficaces son necesarias para comprender el comportamiento geométrico y cinemático del manipulador, globalmente referido como la cinemática del manipulador. Esto representa un área importante y única de la investigación robótica, dado que la investigación en la cinemática y diseño se ha centrado tradicionalmente en los mecanismos de una sola entrada, con actuadores simples moviéndose a velocidades constantes, mientras que los robots son mecanismos espaciales multientradas que requieren de herramientas analíticas más sofisticadas.

El comportamiento dinámico de un robot manipulador también es complejo, dado que la dinámica de acoplamientos espaciales multientrada está altamente acoplada y no lineal. El movimiento de cada articulación se ve afectado de manera significativa por los movimientos de todas las otras articulaciones. La carga inercial impuesta a cada articulación es muy variable dependiendo de la

configuración del brazo manipulador. Los efectos de Coriolis y centrífugos son prominentes cuando el brazo manipulador se mueve a altas velocidades. Las complejidades cinemáticas y dinámicas crean problemas de control únicos que no es posible solucionar adecuadamente a través de técnicas de control lineal estándar, y por lo tanto, hacen que el diseño de un sistema de control eficaz se convierta en un tema crítico dentro de la robótica.



Fuente: "Visitamos la Planta BMW en Munich", BMW Group, 2012.

Fig. 3.1. Robots KUKA en la sección de montaje.

Por último, un robot es exigido a interactuar más estrechamente con los dispositivos periféricos que las máquinas herramientas tradicionales controladas numéricamente. Las máquinas herramienta son esencialmente sistemas autónomos que se encargan de las piezas de trabajo en lugares bien definidos. Por el contrario, el entorno en el que se utilizan los robots es a menudo variable, y medios eficaces deben ser desarrollados para identificar la localización de las piezas de trabajo, así como establecer comunicaciones adecuadas con los dispositivos periféricos y el resto de máquinas de una manera coordinada.

Los robots también son críticamente diferentes de los manipuladores maestro-esclavo, en el sentido de que los primeros son sistemas autónomos. Manipuladores maestro-esclavo son sistemas esencialmente controlados manualmente, donde el operador humano toma las decisiones y aplica acciones de control. El operador interpreta una determinada tarea, encuentra una estrategia adecuada para llevar a cabo la tarea, y tiene previsto el procedimiento de las operaciones. Él inventa una forma eficaz de lograr el objetivo sobre la base de su experiencia y conocimiento acerca de la tarea. Sus decisiones se transfieren a continuación al manipulador esclavo, generalmente a través de una palanca de mando. El movimiento resultante del manipulador esclavo se controla por el operador, y los ajustes o modificaciones necesarias de las acciones de control se proporcionan cuando el movimiento resultante no es adecuado, o cuando se producen eventos inesperados durante la operación. Así, el humano es, por lo tanto, una parte esencial del bucle de control.

Cuando el operador se elimina del sistema de control, toda la planificación y control de comandos deben ser generados por la propia máquina. El procedimiento detallado de las operaciones se debe configurar de antemano, y cada paso de mando de movimiento debe ser generado y codificado en una forma apropiada para que el robot pueda interpretarlo y ejecutarlo con precisión. También se necesitan medios eficaces para almacenar los comandos y gestionar el archivo de datos. Por lo tanto, la programación y la generación de comandos son los temas críticos en la robótica. Además, el robot debe ser capaz de controlar totalmente su propio movimiento.

Fuente: "Da Vinci Robotic Surgery getting popular despite costs", Arabian Gazzete, 2012.

Fig. 3.2. Robot da Vinci realizando una cirugía.

Con el fin de adaptarse a las perturbaciones y los cambios impredecibles en el ambiente de trabajo, el robot necesita una variedad de sensores, a fin de obtener información tanto sobre el medio ambiente (uso de sensores externos, como cámaras o sensores táctiles) y sobre sí mismo (mediante sensores internos, tales como encoders o sensores de torque en las articulaciones). Estrategias basadas en sensores eficaces que incorporen esta información, requieren algoritmos de control avanzados. Pero también implican una comprensión detallada de la tarea.

3.2. Diseño Conceptual

Para llevar a cabo el diseño de cualquier producto, y en concreto el de un sistema robótico, es muy importante que el grupo de diseño focalice sus esfuerzos en precisar y dejar bien claros los objetivos que se persiguen. Sin embargo, aparecen continuamente diseños de baja calidad que fracasan al pretender elaborar un sistema robótico de propósito muy general y lo suficientemente versátil sin centrarse en los objetivos (por ello, muchos de estos sistemas no son capaces de ejecutar completamente las tareas requeridas). Comúnmente, este fracaso se origina debido a que no se ha invertido el esfuerzo necesario en obtener unas especificaciones de diseño adecuadas.

Un sistema robótico, y en general cualquier producto, será un objeto de alta calidad si satisfacen ampliamente demandas o requisitos establecidos. Efectivamente, el diseño desarrollado debe partir de un profundo conocimiento de las especificaciones y requisitos que han de satisfacerse. Además, el diseño debe contemplar cómo estos requisitos afectan a cada uno de los componentes que integran el robot. Dada la complejidad de la tarea que supone recopilar las especificaciones de una aplicación en general, se recomienda al equipo de diseño basarse en la metodología de diseño de una nueva disciplina de la Ingeniería de Diseño, llamada Ingeniería de las Especificaciones.

Algunas de las primeras cuestiones que un diseñador debe plantearse para el diseño de un sistema robótico (dada una aplicación) son:

- ¿Cómo es posible conseguir las especificaciones del diseño?
- ¿Qué agente es responsable de que se originen dichas especificaciones?
- ¿Cuál es la precisión que debe demandar el responsable de diseño?
- ¿Qué tipo de pasos hay que realizar para abordar una aplicación sin unas especificaciones completamente definidas?

Respecto al diseño de sistemas robóticos, la cuestión reside en cómo plasmar las necesidades del usuario en las especificaciones técnicas, para que el sistema robótico presente la funcionalidad requerida. El diseñador debe sopesar todas las consideraciones que puntualicen los usuarios; también es preciso que el responsable de diseño justifique el rechazo de ciertas especificaciones por no ser técnica o económicamente viables. Por tanto, es preciso un entendimiento común de todas las partes para una adecuada motivación y consecución de los objetivos comunes.

En el presente apartado se desarrollará la primera de las tres etapas que forman parte del proceso de diseño de un sistema robótico y que se ha denominado Diseño Conceptual. Así, la primera parte del mismo tratará de la identificación del problema de diseño, mediante una descripción de cómo nace la idea de diseño del robot, basándose en tres agentes relacionados con la aplicación: entorno, usuario y tarea.

A partir de la información procedente del análisis anterior, se podrán establecer las necesidades que deben ser cubiertas en la aplicación y, de este modo, será posible definir posibles alternativas para resolver el problema que satisfaga dichas necesidades y requisitos.

Posteriormente, se llevará a cabo el análisis de una serie de conceptos relacionados con las especificaciones técnicas, económicas y normativas. Para llevar a cabo este

análisis, será necesario hacer un estudio profundo tanto desde un punto general de las características funcionales y especificaciones técnicas donde se analice el escenario, usuario, tarea y seguridad, como de las especificaciones concretas del sistema robótico donde se estudien características como masa, destreza, precisión, entre otras.

Una vez establecidas las características que presenta el sistema robótico, se llevará a cabo un proceso de síntesis estructural de modo que se establezca el número de grados de libertad, disposición de articulaciones, el tipo de configuración, tipo de juntas, etc. Al finalizar todas estas etapas, será posible definir el concepto preliminar del robot, el cual esboza la tendencia del diseño definitivo.

3.2.1. Identificación del problema de diseño

En esta etapa, es necesario analizar, comprender y especificar el contexto de uso del robot. Para ello, se deben identificar el entorno en el cual el sistema va a ser utilizado, las características de los usuarios potenciales, y las tareas que estos van a desarrollar.

El entorno deseado del robot, y en general, de la plataforma experimental científica, es el didáctico, el cual está caracterizado por contener todos los elementos que son necesarios para el aprendizaje de conceptos y destrezas al ritmo del estudiante, con o sin el elemento presencial continuo del profesor. De ahí que el módulo didáctico tenga arquitectura abierta (como se detalló en el Capítulo I). Sin ella, el usuario no podría adentrarse al "corazón" del mismo.

Se suponen usuarios familiarizados con los conceptos básicos de robótica industrial, electricidad, programación y teoría de control. Estas exigencias son necesarias no solo para evitar daños (a nivel hardware o software) en la plataforma experimental, sino principalmente para proteger la integridad del usuario, el cual puede verse tentado a operar el módulo sin primero estar informado acerca de su funcionamiento y modo de interacción confiable.

Las tareas que los usuarios van a desarrollar, involucran interacciones con dos interfaces distintas: la interface física que ofrece cualquier computadora portátil, y la interface natural de usuario (basada en capacidades intuitivas como gestos o movimientos corporales) que todo sensor Kinect exige. Esta última forma de interacción, obliga a considerar un área segura dentro del entorno didáctico, que permita realizar gestos o movimientos libres de colisiones.

3.2.2. Análisis de especificaciones técnicas, económicas y normativas

El objetivo que se persigue en esta sección, es el de identificar los aspectos de diseño más importantes que se deben tener en cuenta, de modo que sea posible idear adecuadamente el sistema robótico.

Para definir las especificaciones del sistema de forma correcta, es preciso estudiar la funcionalidad que debe tener el sistema robótico, para que éste pueda alcanzar los objetivos marcados en la aplicación (partiendo de la descripción funcional de lo que será solicitado).

El sistema robótico, para ser viable, además de desarrollar las secuencias de movimientos requeridas, tendrá que cumplir una serie de requisitos económicos y deberá estar correctamente definido en cuenta a derechos de explotación del producto obtenido.

Es evidente que lo expuesto anteriormente, supone manejar muchos factores y variables que se obstaculizan unas con otras, por lo que es necesario tener claras las restricciones de la aplicación y la repercusión de unos factores sobre otros.

La Tabla 3.1. presenta una lista de especificaciones técnicas, económicas y normativas, que dependiendo si son un deseo o una exigencia, deben ser cubiertas al finalizar el proceso de diseño.

Tabla 3.1. Lista de especificaciones técnicas, económicas y normativas.

Características	Deseos o Exigencias	Condiciones
Geometría	E	Las dimensiones aproximadas del volumen de trabajo del robot deben ser las siguientes: altura (0.45 m) y radio (0.4 m).
	E	La plataforma científica debe estar instalada en un área cuadrada mínima de 16 m^2 y 2.5 m de altura.
	E	La disposición del equipamiento necesario para su funcionamiento (ordenador, sistemas de control, etc.) debe ser tal, que no interrumpa ni obstaculice el accionar del manipulador ni el desplazamiento de los operadores.
Cinemática	E	El robot debe poseer articulaciones del tipo rotacional, y su cinemática debe ser solucionable.
	E	La posición y orientación del efector final del robot estarán determinadas por algoritmos de control.
	E	La velocidad de operación debe ser regulada según la tarea que se desea realizar, no superando las 6 RPM ($\pi/5$ rad/s).
	E	El robot estará dotado de 5 GDL.

Fuerzas	D	El peso total del brazo robótico no debe sobrepasar los 3 kg. aproximadamente.
	D	En todo momento de su accionar, el manipulador deberá mostrar estabilidad, versatilidad y robustez en toda su estructura.
Energía	E	El robot debe estar alimentado por una fuente de poder no regulada que suministre aproximadamente 12 VCC @ 5 A.
	D	La selección de los actuadores debe realizarse de acuerdo a parámetros característicos tales como: configuración, voltaje y corriente de operación, resolución, torque, relación peso-potencia, pérdidas eléctricas entre otros, buscando en lo posible reunir óptimas prestaciones en el actuador seleccionado.
	E	La temperatura de operación permitida de los motores debe mantenerse entre -5°C y 70°C, como lo indica el fabricante de los actuadores, en los datasheets correspondientes, los cuales pueden ser consultados en la sección de Anexos.
Señales	E	El manipulador deberá contar con indicadores visuales de alimentación, corte del flujo de energía y funcionamiento en su sistema de control, fácilmente reconocibles por el ser humano.
	E	La naturaleza de la señal de control debe ser del tipo digital.
	D	El diámetro que el robot desarrolla cuando se encuentra en su posición más crítica (Fig. 3.7.), debe estar señalizado con claridad en el suelo donde sea instalado.
Seguridad	E	Las advertencias sobre peligros mecánicos, eléctricos, térmicos, entre otros, deben estar señalizadas correctamente en la base de sustentación del robot, según la norma NTP 399.010-1.
	E	El grado de protección contra el contacto, penetración de agua y suciedad de los actuadores, debe ser, en lo posible, IP 65.
	E	Con el objetivo de proteger los actuadores, la capacidad de manipulación de cargas del robot no debe exceder los 0.2 kg.
Ergonomía	E	Las dimensiones de los eslabones del robot, deben tomarse como referencia de las longitudes promedio, que un brazo humano adulto posee.
	E	El ambiente donde sea instalado el robot, debe poseer las características respectivas que se mencionan en la sección de Recomendaciones.
Fabricación	E	Las piezas de la estructura robótica deben ser fabricadas a través de métodos de mecanizado utilizados en máquinas-herramientas.

	D	Durante el mecanizado, deben evitarse grandes cuotas de desperdicio de material.
	E	En los planos de despiece debe detallarse, entre otras cosas, las tolerancias dimensionales y geométricas, y acabados superficiales correspondientes.
Control	E	El tipo de control aplicado puede ser: secuencial, por trayectoria, adaptativo o teleoperado, según la norma ISO 8373.
	D	El control que gobierne el funcionamiento del robot debe ser reprogramable y que permita un movimiento fluido y continuo.
	D	Crear un modelamiento computacional (a través de MATLAB, por ejemplo), que permita realizar animaciones de la estrategia de control utilizada, a fin de predecir el comportamiento del robot.
Montaje	E	La instalación y ensamblaje del manipulador robótico debe realizarse bajo la guía que ofrecen los planos de ensamble adjuntos en la sección de Anexos.
	E	Los actuadores deben alojarse de tal manera que se tenga fácil acceso para realizar el mantenimiento respectivo sin dificultades (por ejemplo, pueden instalarse directamente en cada articulación, expuestos al ambiente).
Transporte	E	El robot debe ser fácilmente transportable al lugar de ensamblaje e instalación.
Uso	E	Capacidad de operación bajo condiciones atmosféricas normales (1 atm y 15 °C).
	E	El nivel de ruido permitido no debe sobrepasar los 55 dB, como recomienda la OMS.
Mantenimiento	D	Los motores y articulaciones deben tener un mantenimiento periódico de acuerdo al régimen de uso, para detectar a tiempo posibles fallas y evitar desastres.
	D	Las piezas normalizadas y dispositivos a utilizar, deben ser comerciales y estar disponibles en el mercado para su cambio o fabricación.
Costos	D	Los costos de fabricación deben reducirse en lo posible sin sacrificar la calidad de los dispositivos y piezas, para no elevar el valor del producto terminado.

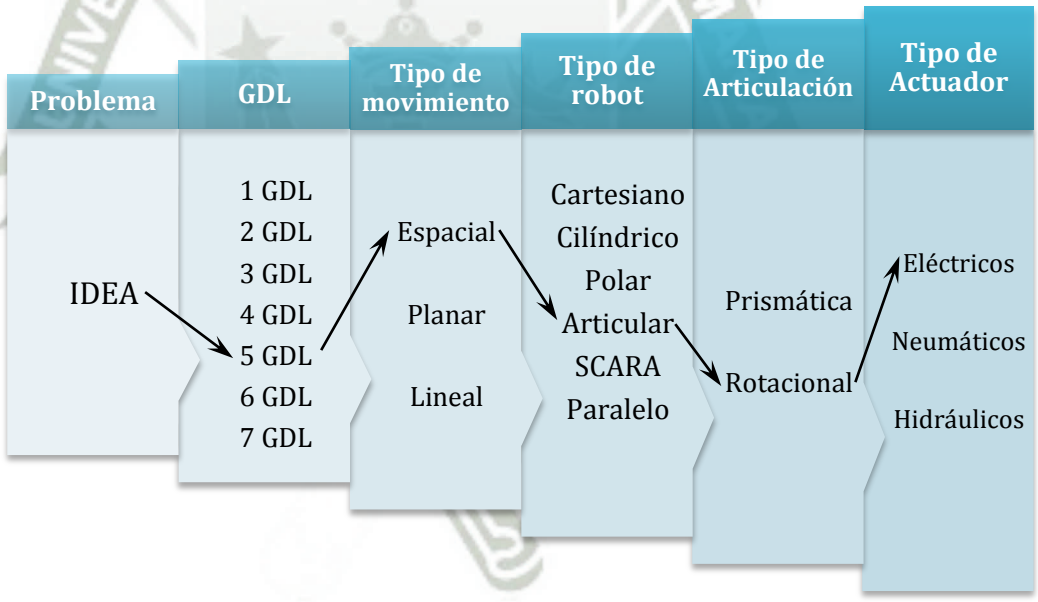
Fuente propia.

3.2.3. Síntesis Estructural

Generalmente, los aspectos considerados en la síntesis estructural, y que están estrechamente relacionados entre sí a lo largo del proceso de diseño, son clasificados en:

- **Síntesis de Número:** Dado un mecanismo específico, este tipo de síntesis se centra en determinar el número de cuerpos. Además, establece el más apropiado número y clase de juntas cinemáticas que el robot debe incluir, para disponer de un número de grados de libertad.
- **Síntesis de Tipo:** Ésta incluye aquellas consideraciones relativas a la selección del tipo de mecanismo que mejor cumple con los requisitos de diseño. En definitiva, con esta síntesis se pretende determinar la clase de eslabones a emplear.
- **Síntesis Dimensional:** Ésta estudia los valores más apropiados para las dimensiones y parámetros geométricos del robot, con el fin de cumplir los requerimientos impuestos por la aplicación.

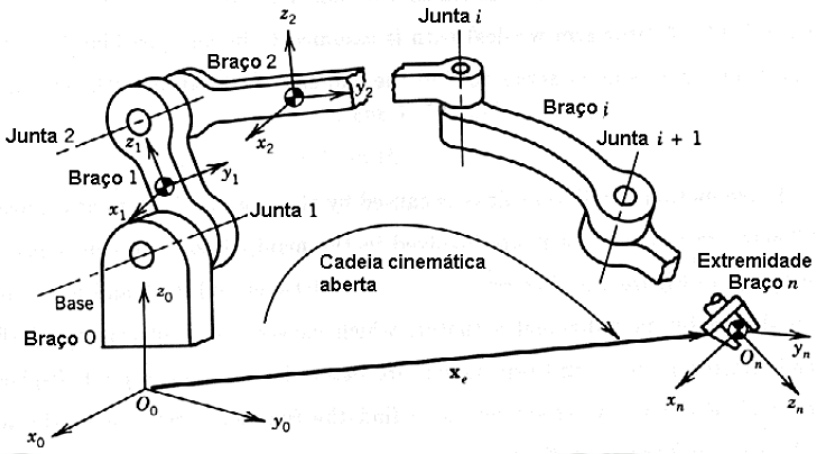
Las dos primeras categorías es posible unificarlas en la síntesis estructural, siendo ésta el primer paso que se debe dar para iniciar el análisis. Por otro lado, la última categoría será analizada utilizando conceptos desarrollados en los dos siguientes apartados.



Fuente propia.

Fig. 3.3. Síntesis Estructural del robot Darko.

En la Fig. 3.3. se muestra la síntesis estructural del robot Darko, donde se han seleccionado 5 GDL (Síntesis de Número), una tipología espacial, un robot articular o antropomórfico, con articulaciones rotacionales (Síntesis de Tipo) y actuadores eléctricos. La Fig. 3.4. muestra el concepto preliminar del robot, generado a partir de la síntesis estructural. Este boceto permite sentar las bases de lo que más adelante se convertirá en el modelo definitivo.



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 3.4. Concepto preliminar del brazo robótico.

3.3. Análisis Cinemático

La cinemática del robot es un estudio analítico de la geometría de su movimiento, sin considerar las fuerzas y momentos que lo originan, cuya importancia la ha convertido en una herramienta fundamental para el diseño y control de robots manipuladores. En este epígrafe, se aplican las herramientas matemáticas estudiadas en el capítulo anterior, necesarias para describir el comportamiento cinemático del robot en estudio, y se discuten sus soluciones.

3.3.1. Descripción de la geometría del mecanismo

Una vez determinado el diseño previo, obtenido durante la etapa de diseño conceptual, se dispone de una serie de información relativa al modelo virtual del robot, aún sin detalles, y principalmente, de una configuración preliminar que identifica, entre otras características, el número de grados de libertad que establecerán el volumen de trabajo del robot. Sin embargo, antes de llevar acabo cualquier análisis matemático, es necesario definir geométricamente el mecanismo.

La descripción geométrica del mecanismo permite tener una visión global del sistema, y debe ser capaz de describir y definir completamente la estructura mecánica del robot. Para ello, se necesita establecer el tipo de eslabones y articulaciones, que en conjunto determinarán la clase de cadena cinemática a tratar.

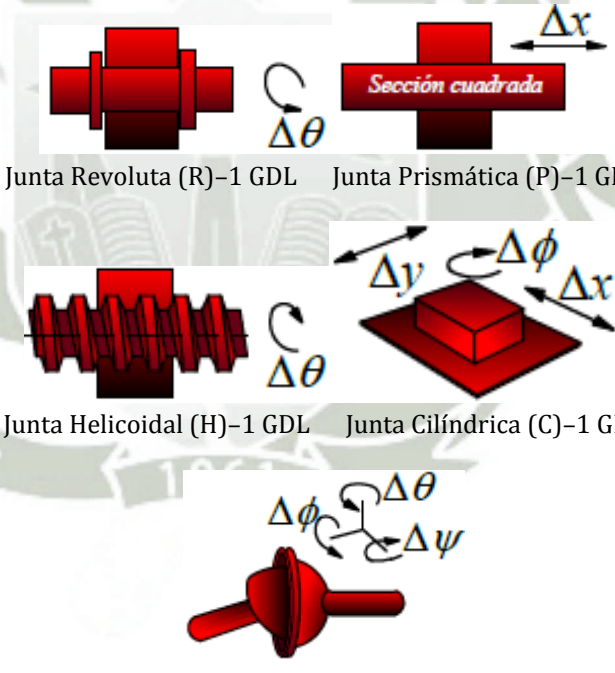
Un eslabón puede ser definido de manera generalizada, como un cuerpo rígido que posee, por lo menos dos nodos, que son puntos de unión con otros eslabones. Se clasifican, generalmente, en función del número de nodos que éste dispone, o lo que es lo mismo, por su grado de conexión. Así se tienen eslabones binarios, terciarios, cuaternarios, en adelante.

Fuente: J. Martínez Verdú, J. M. Sabater Navarro, "Guía docente para el diseño de robots de servicio", 2012.

Fig. 3.5. Eslabones de varios grados de conexión.

Por otro lado, una articulación es una conexión entre dos o más eslabones mediante sus nodos, la cual permite un movimiento relativo entre los eslabones conectados. Se clasifican, principalmente, por el número de grados de libertad que permite y por el tipo de movimiento que genera: rotacional, prismática o deslizante y de movimiento complejo, que resulta de una combinación de los dos anteriores.

Para el robot diseñado, se seleccionaron eslabones binarios (cuyas dimensiones y geometría se detallan en los planos adjuntos en la sección de anexos) y articulaciones rotacionales de 1 GDL.



Fuente: J. Martínez Verdú, J. M. Sabater Navarro, "Guía docente para el diseño de robots de servicio", 2012.

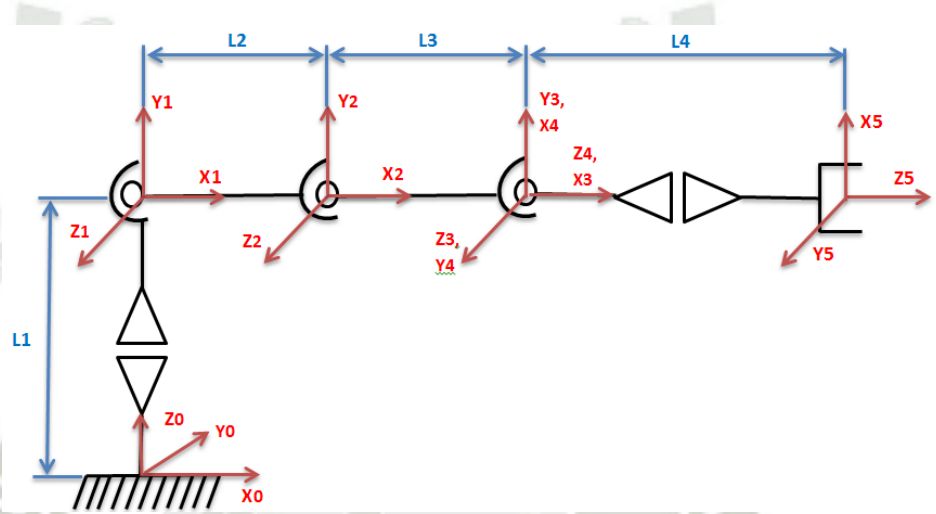
Fig. 3.6. Juntas con contacto superficial (pares inferiores).

3.3.2. Solución del Problema Cinemático Directo

El problema cinemático directo consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas referencial, conocidos los valores de las articulaciones en el tiempo y los parámetros geométricos de los elementos del robot.

Para encontrar dicha relación, se aplica el algoritmo de Denavit-Hartenberg, porque permite abordar de forma sistemática, la obtención del modelo cinemático directo para robots de 5 GDL, como el diseñado en esta investigación.

Se utiliza la notación esquemática de robots, para representar el brazo robótico (Fig. 3.7.), detallando los sistemas de referencia en cada trama, y siguiendo paso a paso las instrucciones del algoritmo, se determinan sus parámetros de D-H, con los que se construye la Tabla 3.2.



Fuente propia.

Fig. 3.7. Representación esquemática del brazo robótico Darko.

Tabla 3.2. Parámetros de D-H del robot.

Articulación	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
1	θ_1	L_1	0	90°
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0
4	$\theta_4 + 90°$	0	0	90°
5	θ_5	L_4	0	0

Fuente propia.

Se calculan las matrices A, sustituyendo en la expresión general de la siguiente manera:

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

$$ {}^{0}A_1 = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{1}A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & L_2C_2 \\ S_2 & C_2 & 0 & L_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

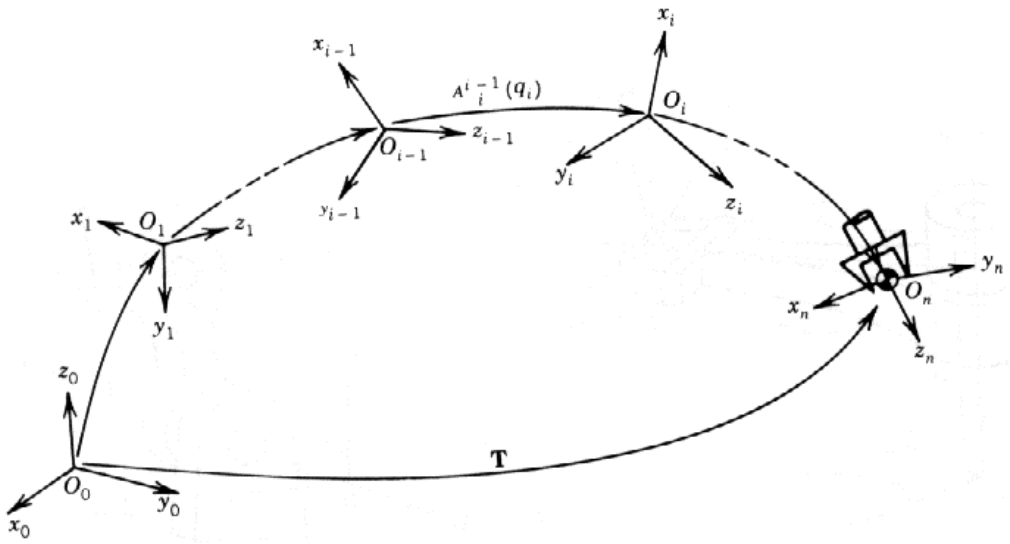
$$^{2}\mathrm{A}_{3} = \begin{bmatrix} C_3 & -S_3 & 0 & L_3C_3 \\ S_3 & C_3 & 0 & L_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}\mathrm{A}_{4} = \begin{bmatrix} C(\theta_4+90) & 0 & S(\theta_4+90) & 0 \\ S(\theta_4+90) & 0 & -C(\theta_4+90) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -S_4 & 0 & C_4 & 0 \\ C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}\mathrm{A}_{5} = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Así, se puede calcular la matriz T que indica la localización del sistema asociado al extremo del robot con respecto al sistema de referencia de la base del mismo:

$$T = {}^{0}\mathrm{A}_{1}\,{}^{1}\mathrm{A}_{2}\,{}^{2}\mathrm{A}_{3}\,{}^{3}\mathrm{A}_{4}\,{}^{4}\mathrm{A}_{5} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 3.8. Representación de la ubicación del efector final por la matriz T.

Dada su extensión, se desarrollan los términos de la matriz T:

$$n_x = S\theta_1 S\theta_5 - C\theta_5[C\theta_4(C\theta_1 C\theta_2 S\theta_3 + C\theta_1 S\theta_2 C\theta_3) + S\theta_4(C\theta_1 C\theta_2 C\theta_3 - C\theta_1 S\theta_2 S\theta_3)]$$

$$n_y = -C\theta_1 S\theta_5 - C\theta_5[C\theta_4(S\theta_1 C\theta_2 S\theta_3 + S\theta_1 S\theta_2 C\theta_3) - S\theta_4(S\theta_1 S\theta_2 S\theta_3 - S\theta_1 C\theta_2 C\theta_3)]$$

$$n_z = C\theta_5 C(\theta_2 + \theta_3 + \theta_4)$$

$$o_x = S\theta_5[C\theta_4(C\theta_1C\theta_2S\theta_3 + C\theta_1S\theta_2C\theta_3) + S\theta_4(C\theta_1C\theta_2C\theta_3 - C\theta_1S\theta_2S\theta_3)] + S\theta_1C\theta_5$$

$$o_y = S\theta_5[C\theta_4(S\theta_1C\theta_2S\theta_3 + S\theta_1S\theta_2C\theta_3) - S\theta_4(S\theta_1S\theta_2S\theta_3 - S\theta_1C\theta_2C\theta_3)] - C\theta_1C\theta_5$$

$$o_z = -S\theta_5C(\theta_2 + \theta_3 + \theta_4)$$

$$a_x = C\theta_1C(\theta_2 + \theta_3 + \theta_4)$$

$$a_y = S\theta_1C(\theta_2 + \theta_3 + \theta_4)$$

$$a_z = S(\theta_2 + \theta_3 + \theta_4)$$

$$p_x = C\theta_1[L_2C\theta_2 + L_3C(\theta_2 + \theta_3) + L_4C(\theta_2 + \theta_3 + \theta_4)]$$

$$p_y = S\theta_1[L_2C\theta_2 + L_3C(\theta_2 + \theta_3) + L_4C(\theta_2 + \theta_3 + \theta_4)]$$

$$p_z = L_1 + L_2S\theta_2 + L_3S(\theta_2 + \theta_3) + L_4S(\theta_2 + \theta_3 + \theta_4)$$

Como se percibe, en estas ecuaciones queda reflejado el valor de la posición (p_x, p_y, p_z) y orientación (n, o, a) del extremo del robot en función de las coordenadas articulares $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$.

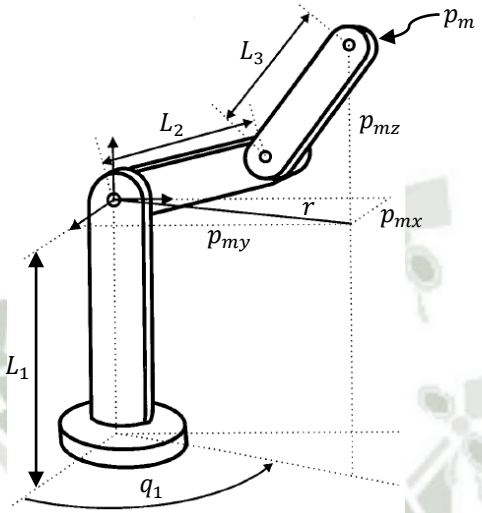
3.3.3. Solución del Problema Cinemático Inverso

El problema cinemático inverso consiste en encontrar los valores que deben adoptar las articulaciones, a partir de una posición y orientación del extremo del robot conocida. Por la complejidad de la matriz de transformación homogénea, y tratándose de un robot de más de 3 GDL, se utilizó el método de desacoplo cinemático para obtener la cinemática inversa del robot. Para ello, se asume un robot de 6 GDL cuyos últimos 3 GDL se cortan en un punto, pero que ha perdido 1 GDL, teniéndose así un robot de 5 GDL. El desacoplo cinemático permite posicionar la muñeca del robot mediante los 3 primeros GDL, y posteriormente encontrar el resto de GDL en función de la orientación deseada del extremo.

Ya obtenidos los parámetros de D-H, se determina la ubicación de la muñeca p_m(punto de corte de los ejes z de los 3 últimos GDL), que coincide en este caso con el origen del sistema $\{S_4\}$. Es importante observar que el movimiento de q_4 y q_5 no afecta a la posición de p_m. La posición del extremo del robot p, se puede obtener trasladando el centro de la muñeca p_m una distancia L_4 a lo largo del eje z_4 que coincide con z_5. Es decir:

$$p_m = p - L_4 * z_5 \rightarrow \begin{bmatrix} p_{mx} \\ p_{my} \\ p_{mz} \end{bmatrix} = \begin{bmatrix} p_x - L_4 * a_x \\ p_y - L_4 * a_y \\ p_z - L_4 * a_z \end{bmatrix}$$

Por tanto, es posible conocer p_m en función de los vectores p y a. Se trata ahora de encontrar los valores de q_1, q_2 y q_3, que consiguen posiciones la muñeca en p_m. Para ello, se considera el robot mostrado en la Fig. 3.9., formado únicamente por los tres primeros grados de libertad.



Fuente: A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, "Fundamentos de robótica", 2007.

Fig. 3.9. Primeros 3 GDL del robot.

La cinemática de este robot puede resolverse por métodos geométricos de la siguiente manera:

$$S\theta_1 p_{m_x} - C\theta_1 p_{m_y} = 0 \rightarrow \boldsymbol{\theta_1 = \tan^{-1}\left(\frac{P_{my}}{P_{mx}}\right)}$$

$$r^2 = {p_{m_x}}^2 + {p_{m_y}}^2$$

$$r^2 + \left(p_{m_z} - L_1\right)^2 = {L_2}^2 + {L_3}^2 + 2L_2L_3\cos\theta_3$$

$$\cos\theta_3 = \frac{{p_{m_x}}^2 + {p_{m_y}}^2 + (p_{m_z} - L_1)^2 - {L_2}^2 - {L_3}^2}{2L_2L_3}$$

Si $\sin\theta_3 = \pm\sqrt{1 - \cos^2\theta_3}$

$$\boldsymbol{\theta_3 = \tan^{-1}\left(\frac{\pm\sqrt{1 - \cos^2\theta_3}}{\cos\theta_3}\right)}$$

$$\boldsymbol{\theta_2 = \tan^{-1}\left(\frac{p_{m_z} - L_1}{\pm\sqrt{{p_{m_x}}^2 + {p_{m_y}}^2}}\right) - \tan^{-1}\left(\frac{L_3 * \sin\theta_3}{L_2 + L_3\cos\theta_3}\right)}$$

Resueltos los 3 primeros GDL, se resolverán los siguientes. Para ello, se debe considerar que las orientaciones de los sistemas $\{S_6\}$ y $\{S_3\}$ vienen relacionados por:

$$^0R_6 = {}^0R_3\,{}^3R_6$$

$$^3R_6 = (^0R_3)^{-1}\,{}^0R_6 = (^0R_3)^T\,{}^0R_6$$

$$^0R_3 = {}^0R_1\,{}^1R_2\,{}^2R_3$$

$$(^0R_3)^T = \begin{bmatrix} C(\theta_2+\theta_3)C\theta_1 & C(\theta_2+\theta_3)S\theta_1 & S(\theta_2+\theta_3) \\ -S(\theta_2+\theta_3)C\theta_1 & -S(\theta_2+\theta_3)S\theta_1 & C(\theta_2+\theta_3) \\ S\theta_1 & -C\theta_1 & 0 \end{bmatrix}$$

$$^3R_6 = {}^3R_4\,{}^4R_5.\mathrm{I}$$

$$^3R_6 = \begin{bmatrix} -C\theta_5S\theta_4 & S\theta_4S\theta_5 & C\theta_4 \\ C\theta_4C\theta_5 & -C\theta_4S\theta_5 & S\theta_4 \\ S\theta_5 & C\theta_5 & 0 \end{bmatrix}$$

Obtenidas las expresiones $(^0R_3)^T$ y $^0R_6$, como datos conocidos y la de $^3R_6$ en función de q_4 y q_5, se tendrá la siguiente expresión que proporciona 9 ecuaciones, con 2 incógnitas:

$$(^0R_3)^T\,{}^0R_6 = {}^3R_6$$

$$\begin{bmatrix} C(\theta_2+\theta_3)C\theta_1 & C(\theta_2+\theta_3)S\theta_1 & S(\theta_2+\theta_3) \\ -S(\theta_2+\theta_3)C\theta_1 & -S(\theta_2+\theta_3)S\theta_1 & C(\theta_2+\theta_3) \\ S\theta_1 & -C\theta_1 & 0 \end{bmatrix}\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} -C\theta_5S\theta_4 & S\theta_4S\theta_5 & C\theta_4 \\ C\theta_4C\theta_5 & -C\theta_4S\theta_5 & S\theta_4 \\ S\theta_5 & C\theta_5 & 0 \end{bmatrix}$$

Tomando los elementos (3,1) y (3,2) y dividiéndolos entre sí:

$$S\theta_1n_x - C\theta_1n_y = S\theta_5 \qquad S\theta_1o_x - C\theta_1o_y = C\theta_5$$

$$\frac{S\theta_1n_x - C\theta_1n_y}{S\theta_1o_x - C\theta_1o_y} = \frac{S\theta_5}{C\theta_5} \rightarrow \boldsymbol{\theta_5 = \tan^{-1}\left(\frac{S\theta_1n_x - C\theta_1n_y}{S\theta_1o_x - C\theta_1o_y}\right)}$$

Tomando el elemento (1,3):

$$[C(\theta_2+\theta_3)C\theta_1]a_x + [C(\theta_2+\theta_3)S\theta_1]a_y + S(\theta_2+\theta_3)a_z = C\theta_4$$

$$\boldsymbol{\theta_4 = \tan^{-1}\left(\frac{\pm\sqrt{1-\cos^2\theta_4}}{\cos\theta_4}\right)}$$

Se hallaron ecuaciones donde queda reflejado el valor de las coordenadas articulares $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ en función de la posición (p_x, p_y, p_z) y orientación (n, o, a) del extremo del robot, lo que significa que fue posible alcanzar una solución cerrada. Sin embargo, y como se verá más adelante, es posible encontrar relaciones más sencillas para q_4 y q_5, que permitan reducir el tiempo de cálculo computacional, considerando que la cinemática inversa debe resolverse casi en tiempo real.

3.3.4. Solución del Problema Cinemático Diferencial

El problema cinemático diferencial busca la relación entre las velocidades de las articulaciones y las del extremo final del robot. Esta relación viene expresada mediante la matriz Jacobiana. A partir de esta matriz será posible también estudiar aspectos de evasión de singularidades, además de evaluar el espacio de trabajo resultante.

Jacobiana Analítica

De la matriz T, obtenida en la solución del problema cinemático directo, se determinan las ecuaciones que describen la posición del efector final con respecto al sistema de referencia de la base del robot:

$$x = C\theta_1[L_2C\theta_2 + L_3C(\theta_2 + \theta_3) + L_4C(\theta_2 + \theta_3 + \theta_4)]$$

$$y = S\theta_1[L_2C\theta_2 + L_3C(\theta_2 + \theta_3) + L_4C(\theta_2 + \theta_3 + \theta_4)]$$

$$z = L_1 + L_2S\theta_2 + L_3S(\theta_2 + \theta_3) + L_4S(\theta_2 + \theta_3 + \theta_4)$$

Aplicando la relación que existe entre los ángulos de Euler WVW y la matriz de transformación homogénea T, se obtienen fácilmente las ecuaciones que describen la orientación del efector final:

$$\phi = \theta_1 \qquad \theta = 90 - (\theta_2 + \theta_3 + \theta_4) \qquad \psi = \theta_5$$

La Jacobiana analítica se obtiene por derivación directa de estas relaciones, resultando:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J_a \begin{bmatrix} \dot{q_1} \\ \vdots \\ \vdots \\ \vdots \\ \dot{q_n} \end{bmatrix} \qquad con \qquad J_a = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \cdots & \frac{\partial f_x}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_\psi}{\partial q_1} & \cdots & \frac{\partial f_\psi}{\partial q_n} \end{bmatrix}$$

$$\frac{\partial f_x}{\partial q_1} = -S\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_x}{\partial q_2} = -C\theta_1[L_2S\theta_2 + L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_x}{\partial q_3} = -C\theta_1[L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_x}{\partial q_4} = -L_4C\theta_1S(\theta_2+\theta_3+\theta_4) \qquad \frac{\partial f_x}{\partial q_5} = 0$$

$$\frac{\partial f_y}{\partial q_1} = C\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_y}{\partial q_2} = -S\theta_1[L_2S\theta_2 + L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_y}{\partial q_3} = -S\theta_1[L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4)]$$

$$\frac{\partial f_y}{\partial q_4} = -L_4S\theta_1S(\theta_2+\theta_3+\theta_4) \qquad \frac{\partial f_y}{\partial q_5} = 0$$

$$\frac{\partial f_z}{\partial q_2} = L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)$$

$$\frac{\partial f_z}{\partial q_3} = L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)$$

$$\frac{\partial f_z}{\partial q_4} = L_4C(\theta_2+\theta_3+\theta_4) \qquad \frac{\partial f_z}{\partial q_5} = \frac{\partial f_z}{\partial q_1} = 0$$

$$\frac{\partial f_\phi}{\partial q_1} = 1 \qquad \frac{\partial f_\phi}{\partial q_2} = \frac{\partial f_\phi}{\partial q_3} = \frac{\partial f_\phi}{\partial q_4} = \frac{\partial f_\phi}{\partial q_5} = 0$$

$$\frac{\partial f_\theta}{\partial q_1} = \frac{\partial f_\theta}{\partial q_5} = 0 \qquad \frac{\partial f_\theta}{\partial q_2} = \frac{\partial f_\theta}{\partial q_3} = \frac{\partial f_\theta}{\partial q_4} = -1$$

$$\frac{\partial f_\psi}{\partial q_1} = \frac{\partial f_\psi}{\partial q_2} = \frac{\partial f_\psi}{\partial q_3} = \frac{\partial f_\psi}{\partial q_4} = 0 \qquad \frac{\partial f_\psi}{\partial q_5} = 1$$

Así, la Jacobiana queda definida en función de la variables articulares, lo que significa que será diferente en cada uno de los puntos del espacio articular. Basta con definir las velocidades articulares, para que a través de la Jacobina se pueda conocer las velocidades del extremo del robot.

Jacobiana Geométrica

Aplicando el método de propagación de velocidades para la obtención de la Jacobiana geométrica, y a partir de las matrices ${}^{i-1}A_i$, se tiene:

$$ {}^0z_0 = {}^0A_0(1:3,3) = (0,0,1)^T $$

$$ {}^0z_1 = {}^0A_1(1:3,3) = (S\theta_1, -C\theta_1, 0)^T $$

$$ {}^0z_2 = {}^0A_2(1:3,3) = (S\theta_1, -C\theta_1, 0)^T $$

$$ {}^0z_3 = {}^0A_3(1:3,3) = (S\theta_1, -C\theta_1, 0)^T $$

$$ {}^0z_4 = {}^0A_4(1:3,3) = \begin{bmatrix} C(\theta_2+\theta_3+\theta_4)C\theta_1 \\ C(\theta_2+\theta_3+\theta_4)S\theta_1 \\ S(\theta_2+\theta_3+\theta_4) \end{bmatrix} $$

$$ {}^0z_5 = {}^0A_5(1:3,3) = \begin{bmatrix} C(\theta_2+\theta_3+\theta_4)C\theta_1 \\ C(\theta_2+\theta_3+\theta_4)S\theta_1 \\ S(\theta_2+\theta_3+\theta_4) \end{bmatrix} $$

$$ {}^0p_5 = {}^0A_5(1:3,4) - {}^0A_0(1:3,4) $$

$$ = \begin{bmatrix} C\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ S\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ L_1 + L_2S\theta_2 + L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4) \end{bmatrix} $$

$$ {}^1p_5 = {}^0A_5(1:3,4) - {}^0A_1(1:3,4) $$

$$ = \begin{bmatrix} C\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ S\theta_1[L_2C\theta_2 + L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ L_2S\theta_2 + L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4+90) \end{bmatrix} $$

$$ {}^2p_5 = {}^0A_5(1:3,4) - {}^0A_2(1:3,4) $$

$$ = \begin{bmatrix} C\theta_1[L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ S\theta_1[L_3C(\theta_2+\theta_3) + L_4C(\theta_2+\theta_3+\theta_4)] \\ L_3S(\theta_2+\theta_3) + L_4S(\theta_2+\theta_3+\theta_4) \end{bmatrix} $$

$$ {}^3p_5 = {}^0A_5(1:3,4) - {}^0A_3(1:3,4) $$

$$ = \begin{bmatrix} L_4[C(\theta_1+\theta_2+\theta_3+\theta_4) + C(\theta_1-\theta_2-\theta_3-\theta_4)]/2 \\ L_4C(\theta_2+\theta_3+\theta_4)S\theta_1 \\ L_4S(\theta_2+\theta_3+\theta_4) \end{bmatrix} $$

$$ {}^4p_5 = {}^0A_5(1:3,4) - {}^0A_4(1:3,4) $$

$$= \begin{bmatrix} L_4[C(\theta_1+\theta_2+\theta_3+\theta_4)+C(\theta_1-\theta_2-\theta_3-\theta_4)]/2 \\ L_4C(\theta_2+\theta_3+\theta_4)S\theta_1 \\ L_4S(\theta_2+\theta_3+\theta_4) \end{bmatrix}$$

La jacobiana geométrica está conformada por:

$$J_1 = \begin{bmatrix} {}^{.0}z_0 \times {}^{.0}p_5 \\ {}^{.0}z_0 \end{bmatrix} \qquad J_2 = \begin{bmatrix} {}^{.0}z_1 \times {}^{.1}p_5 \\ {}^{.0}z_1 \end{bmatrix} \qquad J_3 = \begin{bmatrix} {}^{.0}z_2 \times {}^{.2}p_5 \\ {}^{.0}z_2 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} {}^{.0}z_3 \times {}^{.3}p_5 \\ {}^{.0}z_3 \end{bmatrix} \qquad J_5 = \begin{bmatrix} {}^{.0}z_4 \times {}^{.4}p_5 \\ {}^{.0}z_4 \end{bmatrix}$$

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} = [J_1 \quad J_2 \quad J_3 \quad J_4 \quad J_5]$$

$$J_1 = \begin{bmatrix} -S\theta_1[L_2C\theta_2+L_3C(\theta_2+\theta_3)+L_4C(\theta_2+\theta_3+\theta_4)] \\ C\theta_1[L_2C\theta_2+L_3C(\theta_2+\theta_3)+L_4C(\theta_2+\theta_3+\theta_4)] \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_2 = \begin{bmatrix} -C\theta_1[L_2S\theta_2+L_3S(\theta_2+\theta_3)+L_4S(\theta_2+\theta_3+\theta_4)] \\ -S\theta_1[L_2S\theta_2+L_3S(\theta_2+\theta_3)+L_4S(\theta_2+\theta_3+\theta_4)] \\ L_2C\theta_2+L_3C(\theta_2+\theta_3)+L_4C(\theta_2+\theta_3+\theta_4) \\ S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} -C\theta_1[L_3S(\theta_2+\theta_3)+L_4S(\theta_2+\theta_3+\theta_4)] \\ -S\theta_1[L_3S(\theta_2+\theta_3)+L_4S(\theta_2+\theta_3+\theta_4)] \\ L_3C(\theta_2+\theta_3)+L_4C(\theta_2+\theta_3+\theta_4) \\ S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} -L_4C\theta_1S(\theta_2+\theta_3+\theta_4) \\ -L_4S\theta_1S(\theta_2+\theta_3+\theta_4) \\ L_4C(\theta_2+\theta_3+\theta_4) \\ S\theta_1 \\ -C\theta_1 \\ 0 \end{bmatrix} \qquad J_5 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ C\theta_1C(\theta_2+\theta_3+\theta_4) \\ S\theta_1C(\theta_2+\theta_3+\theta_4) \\ S(\theta_2+\theta_3+\theta_4) \end{bmatrix}$$

Así, la matriz Jacobiana geométrica relaciona las velocidades articulares con las velocidades lineales y angulares del extremo del robot, medidas con respecto del sistema de base, y al igual que la Jacobiana analítica, es diferente en cada uno de los puntos del espacio articular. El lector está

invitado a realizar la comprobación del cálculo realizado, hallando la Jacobiana geométrica a partir de la Jacobiana analítica a través de la relación [2.35].

Jacobiana inversa

Como la matriz jacobiana posee más filas que columnas se utiliza la jacobiana pseudoinversa por la izquierda, bajo el siguiente criterio:

$$J_i^+ = (J^T * J)^{-1} * J^T$$

Debido a su gran extensión y complejidad (matriz simbólica), no se detalla su cálculo en el presente epígrafe. Sin embargo, la obtención de la jacobiana pseudoinversa es susceptible a ser verificada, por ejemplo, a través de la elaboración de un script sencillo en Matlab®, que facilite la compleja tarea del producto de matrices descritas en términos analíticos.

Para verificar si el cálculo ha sido correcto, se multiplica la jacobiana pseudoinversa con la jacobiana y debe obtenerse la matriz identidad.

3.3.5. Parámetros de funcionalidad del robot

Los parámetros de funcionalidad son un conjunto de detalles adicionales a tener en cuenta en el diseño de un brazo robótico, y constituyen una gran ayuda para su optimización.

Solvability

En el apartado anterior se ha resuelto al detalle toda la cinemática de un robot de 5 GDL, y la característica más trascendente es que se obtuvieron soluciones de forma cerrada. La información cinemática que se desee adquirir ya sea en el contexto cartesiano, articular o diferencial, puede obtenerse simplemente de la evaluación de estas ecuaciones, asignando las variables de entrada necesarias de forma correcta.

Por otra parte, si esas soluciones cerradas no se hubieran podido obtener, no sería posible encontrar desplazamientos ni velocidades conjuntas de manera analítica. Esto obligaría a utilizar soluciones de forma abierta, cuyo método iterativo de solución incrementa la cantidad de cálculos, y crea una complejidad computacional considerable, que resta practicidad y rapidez a la solución de los diferentes problemas cinemáticos, sin garantizar una solución cerrada. En particular, el tiempo de cálculo es crucial si las transformaciones deben llevarse a cabo en tiempo real.

La existencia de una solución de forma cerrada depende de la estructura cinemática del brazo robótico. Para una estructura apropiada, tal como la del brazo manipulador propuesto en esta investigación, es posible conseguir una solución de forma cerrada. Una estructura cinemática para la que existe una solución de forma cerrada se conoce como una estructura solucionable, es decir, que posee la característica denominada *solvability*.

La estructura cinemática de un brazo manipulador está a menudo diseñada de modo que su cinemática sea solucionable, con el fin de evitar la complejidad computacional. La mayoría de los robots industriales tienen estructuras con *solvability*, es decir, con solución. Por lo tanto, es importante encontrar qué hace a una estructura cinemática solucionable. Pieper muestra que la condición suficiente para que la estructura cinemática de un brazo manipulador de 6 GDL sea solucionable, es que los ejes de tres articulaciones angulares consecutivas se corten en un punto único para todas las configuraciones de brazos. El robot en discusión satisface dicha condición. De ahí que el método de desacoplo cinemático para la cinemática inversa sea efectivo y tenga validez científica. Se debe tener en cuenta, sin embargo, que la condición de *solvability* dada por Pieper no es indispensable, pero sí suficiente. Así, es posible que otros tipos de estructuras cinemáticas también sean solucionables.

Configuraciones Singulares

Aun teniendo analíticamente una solución al conjunto de ecuaciones cinemáticas, es muy probable que la solución se encuentre fuera del espacio de trabajo del robot o simplemente, no satisfaga el rango de movimiento de las articulaciones. En estos casos se dice que existe una configuración singular de límite o interna. En efecto, cerca de una singularidad, un incremento infinitesimal en las coordenadas cartesianas implica un incremento infinito en el espacio articular.

El Jacobiano es la herramienta principal para determinar las singularidades, pues éstas corresponden con aquellos valores articulares que anulan su determinante. Sin embargo, la Jacobiana hallada anteriormente (de orden 6x5) no es cuadrada[10]. Para solucionar este inconveniente, se prescinde del ángulo de orientación ψ, ya que este es independiente del resto de articulaciones. Así se obtiene una matriz cuadrada de 5x5, cuyo determinante es:

$$|J_s| = -L_2 L_3 S\theta_3 C(\theta_2 + \theta_3 + \theta_4)[L_2 C\theta_2 + L_3 C(\theta_2 + \theta_3) + L_4 C(\theta_2 + \theta_3 + \theta_4)]$$

[10] Por propiedades de matrices, solo las matrices cuadradas, es decir, de orden $n \times n$, poseen determinante.

Que se anula para:

$$L_2C\theta_2 + L_3C(\theta_2 + \theta_3) + L_4C(\theta_2 + \theta_3 + \theta_4) = 0$$

La relación se cumple para:

$$\boldsymbol{\theta_2 = \pm \pi/2} \qquad \boldsymbol{\theta_3 = \theta_4 = 0} \text{ ó } \boldsymbol{\pi}$$

Como se verá en el siguiente apartado, se presta especial atención a la localización de las configuraciones singulares del robot, para que sean consideradas tanto en la descripción del espacio de trabajo como en su control, evitándose solicitar a los actuadores movimientos a velocidades inabordable o cambios bruscos de las mismas.

Espacio de Trabajo

Al diseñar un robot, e independientemente del tipo, se debe definir su espacio de trabajo, el cual está limitado por el rango de valores angulares que cada articulación puede alcanzar y por las dimensiones de los eslabones. En este apartado se definen los límites angulares de cada articulación, dejando el detalle dimensional para el análisis dinámico, debido a que la estimación de pesos va de la mano con las dimensiones de cada eslabón.

Lo primero que debe tenerse en cuenta para definir un espacio de trabajo accesible, es evitar configuraciones angulares que produzcan colisiones o interferencias entre eslabones. Lo segundo que debe evitarse, en lo posible, son la aparición de configuraciones singulares. Para ello, es necesario imponer restricciones al movimiento articular, de modo que los valores angulares que producen una singularidad no se encuentren dentro del rango de movimiento correspondiente a cada articulación. La combinación de ambos criterios permite definir el recorrido máximo y mínimo que cada articulación puede alcanzar y el espacio de trabajo del robot. La Tabla 3.3. muestra el rango de movimiento estimado de cada articulación del robot.

Tabla 3.3. Rango de movimiento estimado para cada eje del robot.

Articulación	Rango de movimiento estimado
1	$\pm 120°$
2	$-7°\ a + 188°$
3	$-118°\ a + 114°$
4	$\pm 102°$
5	$\pm 150°$

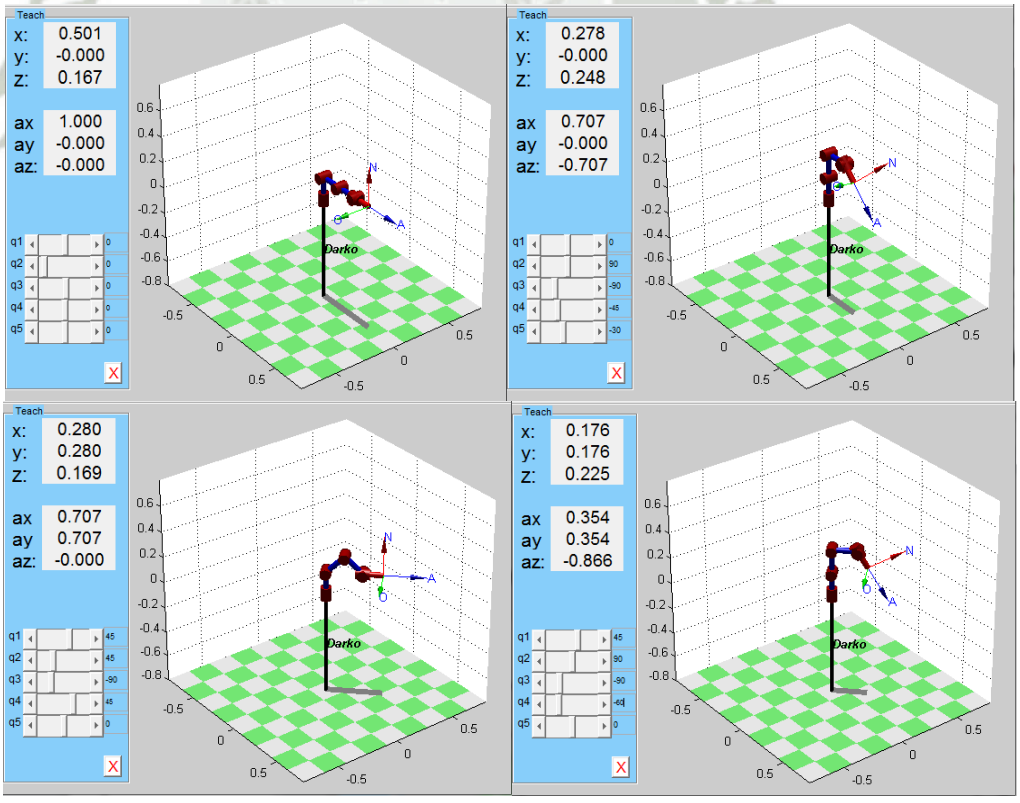
Fuente propia.

Como se puede apreciar, el rango de movimiento estimado para la articulación 2 tiene como límite inferior los -7°, excluyendo la configuración angular de -90°. De la misma manera, para la articulación 3 y 4 se excluye el valor de 180°. Así, se logran reducir las probabilidades de aparición de singularidades, beneficio que se debe aprovechar durante el diseño de la estrategia de control.

3.3.6. Animación Cinemática Virtual

Tras el análisis cinemático, es posible realizar una verificación utilizando herramientas informáticas para contrastar las soluciones de los problemas cinemáticos (directo e inverso) de forma rápida y sencilla. Para ello, se utiliza la versión 9.9 del Robotics Toolbox for Matlab® de Peter Corke, última actualización lanzada a fines de Abril de este año.

El script *Darkobot* elaborado, utilizando los comandos del Robotics Toolbox, permite comprobar la cinemática directa del robot en estudio. Al ejecutarlo, se crea un escenario animado donde es posible accionar el robot a partir de los sliders de coordenadas articulares. La Fig. 3.10. muestra diversas configuraciones elegidas aleatoriamente.



Fuente propia.

Fig. 3.10. Configuraciones aleatorias aplicando el Robotics Toolbox de P. Corke.

3.4. Análisis Dinámico

El análisis dinámico trata con las formulaciones matemáticas de las ecuaciones de movimiento del brazo robótico, que describen su conducta dinámica. A diferencia del análisis cinemático, aquí si se estudian las fuerzas y momentos que causan el movimiento del robot, y se definen diversos aspectos estructurales que involucran la selección de materiales y actuadores, geometría de eslabones y juntas, estimación de pesos, que en definitiva, permiten diseñar una estructura robótica óptima y definitiva.

3.4.1. Selección de Materiales

Tradicionalmente, en el diseño de sistemas robóticos y como herencia de la robótica industrial, suelen seleccionarse materiales metálicos, esencialmente aceros estructurales con alto contenido en carbono, debido a que presentan una gran resistencia. Mediante la incorporación de aleantes, tanto en aceros como en aluminios, se han conseguido materiales con elevadas resistencias mecánicas, del orden de varios miles de MPa.

Sin embargo, cada vez es más frecuente la utilización de materiales más ligeros como los cerámicos o los termoplásticos. Si bien los primeros presentan dificultades en los procesos de conformación, que los hacen inadecuados en la fabricación tradicional por arranque de viruta, los materiales plásticos y, en especial, los compuestos con matriz plástica, están siendo ampliamente utilizados para sustituir diversas piezas metálicas que están sometidas a esfuerzos exigentes.

Normalmente, el punto de partida para la selección de los materiales a utilizar en un robot es muy poco concreto y sin embargo, afectan muy significativamente al coste pues determinan las técnicas de fabricación. Sin embargo, del diseño conceptual elaborado anteriormente, se tienen un conjunto de especificaciones técnicas, económicas y normativas, que permiten establecer las exigencias a las que debe sujetarse la estructura robótica, y marcan un buen punto de inicio en la búsqueda por encontrar el material más indicado.

Es por ello que en el proceso de selección, no se trató al material como un elemento individual, sino como un conjunto de atributos físicos y económicos, cuyo perfil debía ajustarse a unas determinadas especificaciones. Los atributos determinantes fueron: baja densidad (de forma que se minimice el peso del robot), alta resistencia mecánica (para que soporte impactos y aporte estabilidad dimensional a lo largo del proceso de fabricación), reciclable (porque todo diseño debe proteger el

medio ambiente), buena facilidad de mecanizado (para reducir los costos de fabricación), y adquisición accesible en el mercado local.

Después de una exploración por el mundo de los polímeros de ingeniería, se eligió al Nylon como el material ideal para la estructura mecánica del robot.

El Nylon es un termoplástico obtenido a partir de la poliamida 6, que difiere de los plásticos de uso corriente por su magnífico cuadro de propiedades mecánicas, eléctricas, térmicas, químicas, y la posibilidad de ser modificado con aditivos (MoS_2), para una mayor resistencia al desgaste.

Se pueden distinguir varios tipos de Nylon, siendo los más comunes el Nylon 6 (PA 6, por su designación química), poliamida estructurada a partir de un solo material de partida (la caprolactama, molécula clave en la síntesis del Nylon), y el Nylon 66 (PA 66), poliamida estructurada a partir de 2 materiales de partida (ácido adípico y hexametilendiamina).

Para fabricar la estructura robótica, se utilizó el Nylon 6 (Fig. 3.11.), conocido comercialmente como Sustamid 6G, Ertalón 6, Nitanyl, Grilón, Sneamid, Ultramid B, Akulón F, Durethan B, entre los principales. En el Perú, se comercializa en tres presentaciones:

- **Barras Redondas**: De 1 m. de longitud y diámetros de 6 a 250 mm.
- **Barras Cuadradas:** De 1 m. de longitud y superficie de 15x15 a 80x80 mm.
- **Planchas:** De espesores que van de 4 a 64 mm. y secciones (largo x ancho) de 2x1 m.[11]

Fuente: "Nylon 6 MGS", Corporación Jebeemsa S.A.C., 2014.

Fig. 3.11. Barras cilíndricas y cuadradas de Nylon 6.

La Tabla 3.4. recopila las propiedades físicas más básicas del Nylon 6. Se invita al lector visitar la sección de anexos, donde se adjunta una tabla más detallada.

[11] Las dimensiones de las barras y planchas pueden variar dependiendo del proveedor. Las dimensiones aquí detalladas fueron extraídas de http://www.hynempaquetaduras.com/nylon.html.

Tabla 3.4. Propiedades físicas básicas del Nylon 6.

Propiedad Física	Unidad	Valor
Densidad	gr/cm^3	1.15
Temperatura máxima de utilización continua	°C	90
Absorción de agua	100%HR	8.9
Coeficiente de rozamiento	μ	0.3-0.4
Resistencia a la tracción	Mpa	65-75
Resistencia al impacto	Kj/cm^2	3.5-4.5
Resistencia a la flexión	Mpa	70-80

Fuente: "Nylon 6 MGS", Corporación Jebeemsa S.A.C., 2014.

3.4.2. Modelo dinámico

El modelo dinámico de un robot manipulador de n GDL, en su forma compacta y más utilizada en robótica, está dada por la siguiente ecuación

$$\boldsymbol{\tau(t) = \mathbf{H}\big(q(t)\big)\ddot{q}(t) + \mathbf{C}\big(q(t), \dot{q}(t)\big) + \mathbf{G}\big(q(t)\big) + F_f(\dot{q}(t), f_e)}$$

Donde:

$\tau(t) = n \times 1$ vector de par generalizado aplicado en las articulaciones $i = 1,2 \dots, n$, esto es :

$$\tau(t) = [\tau_1(t), \tau_2(t), \dots, \tau_n(t)]^T$$

$q(t) =$ un vector $n \times 1$ de las variables de articulación del brazo y se puede expresar como:

$$q(t) = [q_1(t), q_2(t), \dots, q_n(t)]^T$$

$\dot{q}(t) =$ un vector $n \times 1$ de la velocidad de las articulaciones del brazo y se puede expresar como:

$$\dot{q}(t) = [\dot{q}_1(t), \dot{q}_2(t), \dots, \dot{q}_n(t)]^T$$

$\ddot{q}(t) =$ un vector $n \times 1$ de la aceleración de las variables de articulación $q(t)$ y se puede expresar como:

$$\ddot{q}(t) = [\ddot{q}_1(t), \ddot{q}_2(t), \dots, \ddot{q}_n(t)]^T$$

$\mathrm{H}(q) =$ una matriz simétrica de efectos inerciales relacionada con la aceleración $n \times n$ cuyos elementos son :

$$\mathrm{H}_{ij} = \sum_{k=(\max i,j)}^{n} \mathrm{Traza}\big(\mathbf{U}_{kj}\mathbf{J}_k\mathbf{U}_{ki}{}^T\big) \qquad i,j = 1,2, \dots, n$$

$\mathrm{C}(q, \dot{q}) =$ un vector de fuerzas centrípetas y de Coriolis no lineal $n \times 1$ cuyos elementos son:

$$\mathrm{C}(q, \dot{q}) = [c_1, c_2, \dots, c_n]^T$$

Donde:

$$c_i = \sum_{k=1}^{n} \sum_{m=1}^{n} d_{ikm} \dot{q_k} \dot{q_m} \qquad i = 1,2, \dots, n$$

Y:

$$d_{ikm} = \sum_{j=(\max i,k,m)}^{n} \mathrm{Traza}\left(\mathbf{U}_{jkm} \mathbf{J}_j \mathbf{U}_{ji}{}^T\right) \qquad i, k, m = 1,2, \dots, n$$

$\mathrm{G}(q) =$ un vector de pares gravitatorios $n \times 1$ cuyos elementos son:

$$\mathrm{G}(q) = [g_1, g_2, \dots, g_n]^T$$

Donde:

$$g_i = \sum_{j=1}^{n} \left(-m_j \mathbf{g} \mathbf{U}_{ji}{}^j \mathrm{r}_j\right) \qquad i = 1,2, \dots, n$$

Se ha incluido también el vector de pares de fricción que incluye la fricción viscosa, de Coulomb y estática ($\boldsymbol{f_e}$) de cada articulación del robot.

La trascendencia del modelo dinámico radica en lo siguiente: representa la base matemática para llevar a cabo el análisis y estudio de los fenómenos físicos presentes en la estructura mecánica de un robot manipulador de n GDL en cadena cinemática abierta, con eslabones rígidos, en su rango de operación nominal. Para realizar dicho estudio, es necesario hallar los coeficientes dinámicos H, C y G para cada valor de q_i y $\dot{q}_i$. El método más óptimo para lograrlo es a través de la formulación de L-E.

Formulación de Lagrange-Euler

La razón por la que se utiliza la formulación de L-E, es porque permite obtener ecuaciones finales bien estructuradas, donde aparecen de manera clara todos los fenómenos físicos que se encuentran en la estructura mecánica del robot y que intervienen en su movimiento, tales como efectos inerciales, fuerzas centrípetas y de Coriolis, par gravitacional y fricción. Esta ventaja es explotada en el siguiente capítulo, para facilitar el diseño y

análisis de la estrategia de control, así como para mejorar la simulación virtual del movimiento del robot.

Por tratarse de un robot de 5 GDL, transcribir paso a paso el detalle del algoritmo computacional de L-E y sus resultados, resulta demasiado extenso, complejo y poco didáctico. Sin embargo, el procedimiento que fundamenta la obtención de los coeficientes dinámicos H, C y G expresados en función de q_i y $\dot{q}_i$, está plasmado en el script de Matlab® titulado "Dinamica_Lagrange". Ejecutando este script, pueden obtenerse las matrices de efectos inerciales H(q) de dimensión 5x5, de fuerzas centrípetas y de Coriolis C($q, \dot{q}$), y de pares gravitatorios G(q), estas últimas del orden de 5x1, que describen el modelo dinámico del robot.

3.4.3. Solución del Problema Dinámico Inverso

El resumen teórico sobre la dinámica del robot, visto en el capítulo anterior, deja un mensaje claro: las formulaciones clásicas de L-E y N-E, a pesar de modelar el comportamiento dinámico del robot con una buena aproximación a la realidad, no se pueden utilizar con fines de control en tiempo real, donde se deben alcanzar tiempos de cálculo rápido en la evaluación de los pares de las articulaciones nominales.

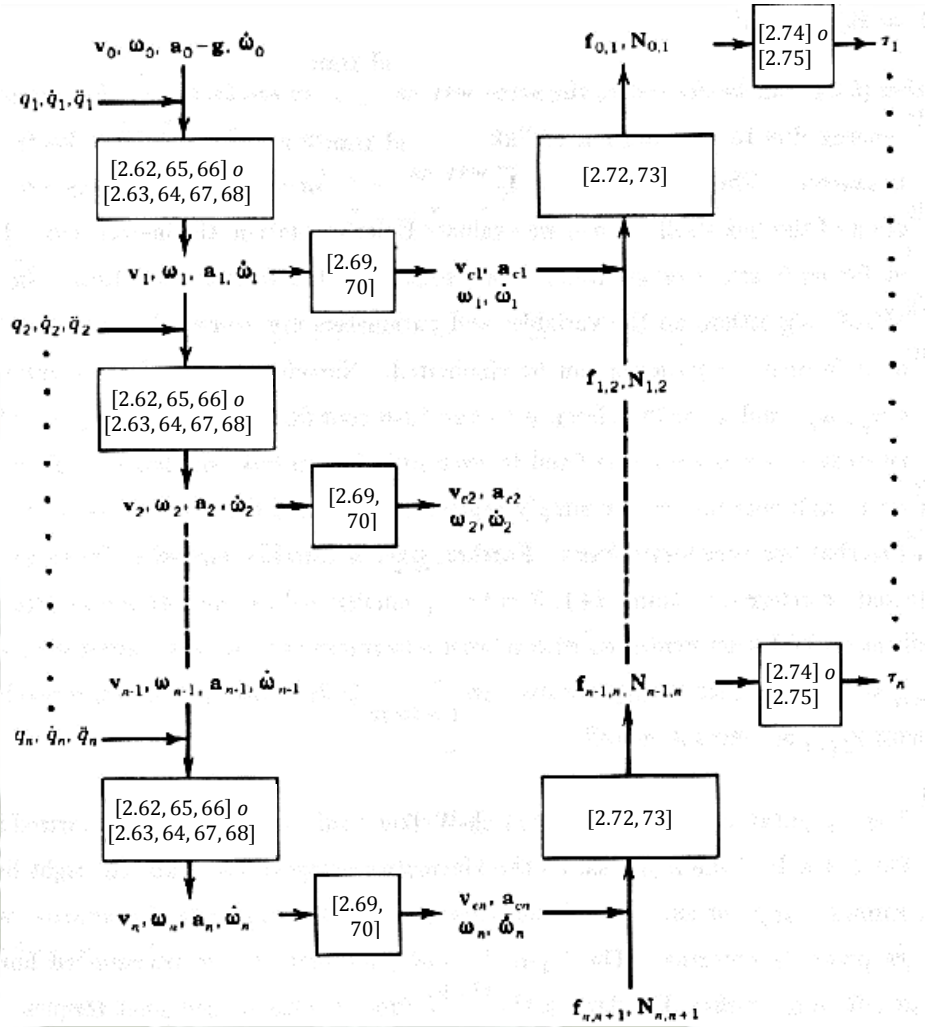
Para satisfacer está necesidad de ecuaciones de movimiento más eficientes, se recurre al algoritmo recursivo de Luh-Walker-Paul, por ser el más eficiente computacionalmente, y cuya complejidad depende directamente del número de grados de libertad.

Algoritmo Luh-Walker-Paul

Conocidos todos los pasos de las dos fases del algoritmo Luh-Walker-Paul (cálculo cinemático "hacia adelante" y cálculo dinámico "hacia atrás"), es necesario esbozar una estructura recursiva que permita su implementación computacional a través de un script.

La Fig. 3.12. muestra la estructura computacional de este algoritmo, cuyo desarrollo va de izquierda a derecha. Antes de iniciar el cálculo cinemático, es necesario definir las condiciones del eslabón cero o base ($v_0, \omega_0, a_0, g, \dot{\omega}_0$), las condiciones iniciales de fuerza $f_{5,6}$ y par $N_{5,6}$ en el efector final, y los vectores b_i, $r_{i.i+1}$, y $r_{i.ci}$ para cada articulación.

Para el cálculo dinámico, es necesario definir, de forma numérica o analítica, los tensores de inercia correspondientes a cada eslabón. De ser necesario, considerar el tensor de la carga a manipular.



Fuente: H. Asada, J.-J. E. Slotine, "Robot Analysis and Control", 1986.

Fig. 3.12. Estructura computacional del Algoritmo Luh-Walker-Paul.

Finalizada la fase de cálculo dinámico, es posible obtener los pares de torsión articulares que deben aplicarse a cada actuador, para que el robot realice, por ejemplo, una determinada trayectoria.

3.4.4. Selección de la Motorización

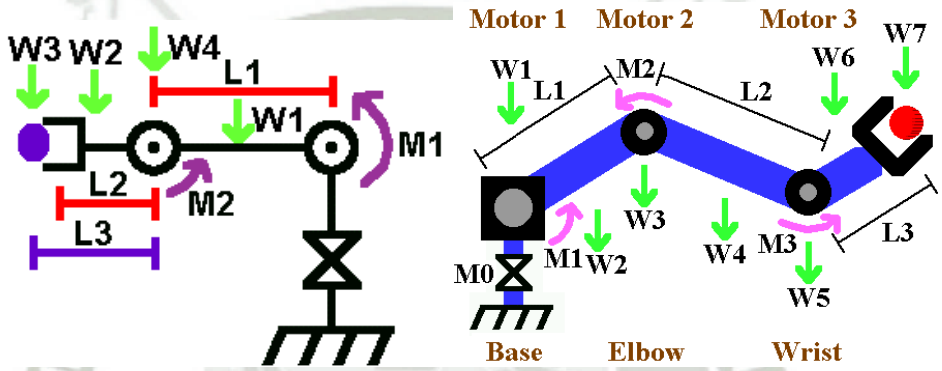
A partir de la solución de dinámica inversa, es posible obtener información acerca de fuerzas y torques que aparecen en las articulaciones como resultado del movimiento del robot. Esto, unido a un cálculo preliminar de momentos, permiten dimensionar los actuadores en función de los torques solicitados.

La selección de los actuadores y el modelo de los mismos en base a la dinámica inversa y catálogos comerciales de motores, es un paso clave en el proceso de diseño de un robot. El presente epígrafe pretende mostrar los pasos a seguir para una correcta elección de actuadores.

Cálculo preliminar de momentos

El cálculo preliminar de momentos no es más que un balance estático de los mismos. El primer paso es elaborar el diagrama de cuerpo libre (DCL) del brazo robótico, extendido en su máxima longitud (Fig. 3.13.), la cual corresponde con la configuración más crítica que puede adoptar. A continuación, se determinan los siguientes parámetros:

- Peso w_n y longitud L_n de cada eslabón.
- Peso w_{na} de cada articulación (considerando los actuadores).
- Peso del objeto a manipular w_{carga}.[12]



Fuente: "Robot Arm Tutorial", Society of Robots, 2005.

Fig. 3.13. DCL del robot para el balance estático de momentos.

Finalmente, se realiza el cálculo de momentos bajo el concepto sencillo de multiplicar la fuerza por la distancia. Para ello, primero se determina como punto de balance el centro de cada articulación, y luego se procede a calcular, para cada actuador y desde el extremo de la muñeca del robot hacia la base, bajo la siguiente convención de signos: los momentos son positivos si se aplican en sentido antihorario, y lo son negativos en sentido horario.

A continuación, se muestra el procedimiento del balance de momentos realizado para el robot Darko, en el que se asume que el centro de masa de cada eslabón está situado en la mitad de su longitud. Así, para cada articulación se tiene:

$$\tau_5 = w_{carga} * l = (0.3\ kg)\left(9.81\frac{m}{s^2}\right)(10.6 * 10^{-3}\ m) \rightarrow \boldsymbol{\tau_5 = 31.2\ mN.m}$$

$$\tau_4 = w_4\left(\frac{L_4}{2}\right) + w_{carga}L_4$$

[12] Es recomendable asegurarse de que el motor que se elija, no sólo pueda soportar el peso de la sección del robot que le corresponde, sino también la carga a manipular, aunque ésta sea netamente simbólica.

$$\tau_4 = (9.81)\left[0.142\left(\frac{0.392}{2} + 0.3\right)\right] \rightarrow \boldsymbol{\tau_4 = 0.691\ \text{N.m}}$$

$$\tau_3 = w_3\left(\frac{L_3}{2}\right) + w_{4a}L_3 + w_4\left(L_3 + \frac{L_4}{2}\right) + w_{carga}(L_3 + L_4)$$

$$= (9.81)\left[0.178\left(\frac{0.218}{2} + 0.17\right) + 0.392\left(0.178 + \frac{0.142}{2}\right) + 0.3(0.178 + 0.142)\right]$$

$$\boldsymbol{\tau_3 = 2.386\ \text{N.m}}$$

$$\tau_2 = w_2\left(\frac{L_2}{2}\right) + w_{3a}L_2 + w_3\left(L_2 + \frac{L_3}{2}\right) + w_{4a}(L_2 + L_3) + w_4\left(L_2 + L_3 + \frac{L_4}{2}\right) + w_{carga}(L_2 + L_3 + L_4)$$

$$= (9.81)\left[0.181\left(\frac{0.265}{2} + 0.187\right) + 0.218\left(0.181 + \frac{0.17}{2}\right) + 0.17(0.181 + 0.178) + 0.392\left(0.181 + 0.178 + \frac{0.142}{2}\right) + 0.3(0.181 + 0.178 + 0.142)\right]$$

$$\boldsymbol{\tau_2 = 4.863\ \text{N.m}}$$

$$\tau_1 = \text{w}\left(\frac{L_2 + L_3 + L_4}{2}\right) = (2)(9.81)\left(\frac{0.501}{2}\right) \rightarrow \boldsymbol{\tau_1 = 4.915\ \text{N.m}}$$

Los torques calculados son los máximos requeridos por el robot, y fueron obtenidos a partir de un equilibrio de momentos estático, lo que significa que, teóricamente, deben imprimirse dichos valores a cada actuador para que el robot pueda mantener esa configuración crítica sin desplomarse.

Selección de Actuadores

El diseñador siempre deberá llevar a cabo una selección de los actuadores, ya que suele representar en torno a la mitad del peso del sistema robótico. Una concienzuda selección de actuadores, permitirá minimizar la masa que aporta cada uno de ellos.

Como se recuerda, el objetivo fundamental de todo accionamiento es mover una carga a una determinada velocidad y posicionarla donde sea preciso. El servoaccionamiento más utilizado en robótica es el eléctrico y este tipo de actuadores supone aproximadamente un 50% del peso total del robot.

Para que el motor cumpla técnicamente, no sólo es necesario que tenga la capacidad de acelerar la carga requerida por la aplicación para moverla en un tiempo preestablecido, sino que éste no debe sobredimensionarse innecesariamente. Además, el actuador no deberá sufrir sobrecalentamiento en operación normal, ni causar perturbaciones electromagnéticas que puedan afectar a otros equipos o causar malestar al usuario, debido al ruido producido por la utilización del mismo. Éstas son algunas de las especificaciones que, junto a las especificaciones cinemáticas y dinámicas que están consignadas en el diseño conceptual, permiten construir un criterio sólido para seleccionar el tipo de actuador que dará vida al robot. El objetivo es claro: elegir un actuador que reúna óptimas prestaciones.

Luego de una búsqueda exhaustiva a través de diversos catálogos de fabricantes, se optó por utilizar servomotores de corriente continua, entre otras cosas, porque son ideales cuando se desea girar un eje y detenerlo en cualquier posición dentro de su rango de operación, con una precisión de hasta centésimas de milímetros. Esta característica está en armonía con la aplicación del robot y es difícil de lograr con los motores DC, en los que el rotor continúa girando inercialmente hasta una posición casual.

Se eligieron los Dynamixel *Smart Servos* (Fig. 3.14.), del fabricante coreano ROBOTIS, porque:

- Enriquecen la precisión de la cadena cinemática gracias a su alta resolución, amplio rango de trabajo[13] y velocidad.
- Son pequeños, ligeros, fáciles de montar en la estructura robótica, y poseen excelente relación torque-velocidad.
- Permiten la topología de conexión en cadena (*daisy chain*), lo que facilita el cableado.
- Son silenciosos y de alta eficiencia debido a la baja corriente de consumo.
- Gracias al buen soporte técnico que ofrece el fabricante, su instalación y puesta en marcha es sencilla, poseen repuestos y son compatibles con diversos software de control.
- Están equipados con un sistema de realimentación interna de posición, velocidad, temperatura de operación, carga y tensión de alimentación, cuya información captada por sus sensores embebidos, puede leerse en tiempo real.

[13] Los actuadores Dynamixel *Smart Servos* de la serie AX, tienen un rango de operación de 300°; los de la serie MX, pueden alcanzar un desplazamiento angular de hasta 360°.

- Son controlados por protocolos de comunicación digital, a través de paquetes de datos enviados por la red serial. Soportan comunicación TTL y el estándar RS-485 (dependiendo del modelo).
- En los modelos más avanzados, es posible configurar un control PID para minimizar oscilaciones.

Fuente: "Smart Servo Dynamixel", tdrobótica.co, 2011.

Fig. 3.14. Dynamixel *Smart Servos*.

Dynamixel son los actuadores más avanzados, con muy altas prestaciones a nivel de robótica personal. Estos actuadores robóticos superan con creces al típico servomotor RC de hobby, que sólo entiende la orden "ángulo objetivo" (dada por una señal PWM), y posee un rango de trabajo limitado que va de 0 a 180°. La Tabla 3.5. muestra los modelos seleccionados de servomotores, junto a sus características más representativas. En la sección de Anexos es posible encontrar los *datasheets* de cada uno de ellos, con información técnica más detallada.

Tabla 3.5. Modelos de servomotores Dynamixel seleccionados.

Articulación	Servomotor	Voltaje (V)	Torque (N.m)	Resolución (°)	Masa (g)
1	Dynamixel MX-64T	12	6	0.088	126
2	Dynamixel MX-64T	12	6	0.088	126
3	Dynamixel MX-28T	12	2.5	0.088	72
4	Dynamixel AX-12A	12	1.52	0.29	55
5	Dynamixel AX-12W	12	0.2	0.29	53
Gripper	Tower Pro SG-5010	6	0.51	1.2	41

Fuente propia.

Dejando de lado sus óptimas prestaciones técnicas, lo más resaltante de estos servomotores es su excelente relación peso-torque, debido a que normalmente éste último es directamente proporcional al primero: un actuador con mayor torque es siempre más voluminoso y por lo tanto más pesado, lo que constituye una gran desventaja estructural que impone restricciones importantes (puede llegarse al extremo de que esto genere reconsiderar todo el proceso de selección del actuador).

El fabricante ROBOTIS, ofrece el software RoboPlus que permite, entre otras cosas, editar parámetros importantes de cada actuador (ID, rango de movimiento, habilitación de par, etc.), pero que es muy limitado para las perspectivas ambiciosas de esta investigación, en lo que a control respecta. Sin embargo, estos *"Smart Servos"* poseen tal grado de versatilidad, que gracias a sus librerías múltiples integradas en el Dynamixel SDK, pueden ser controlados a través de diversos software, entre los que se encuentran: Matlab®, Python™, Microsoft® Visual Studio, y LabVIEW™.[14]

Velocidad de operación

La velocidad de operación de las articulaciones de un robot, viene limitada principalmente por la matriz de fuerzas centrípetas y de Coriolis. Como se recuerda, ésta matriz se encuentra expresada en función de la posición angular q y de la velocidad angular $\dot{q}$. Para tener mayor éxito en el control dinámico de un robot, garantizar un movimiento seguro y suave, evitar la aparición de grandes fuerzas y pares de acoplamiento que la estructura robótica no pueda soportar, entre otras finalidades, se busca reducir al máximo su valor numérico de dicha matriz, a fin de minimizar todos los efectos negativos que la matriz C genera.

Bajo estas consideraciones y luego de algunas experimentaciones, se establece que la velocidad máxima para las articulaciones será de 6 RPM (aproximadamente $\pi/5$ rad/s). Este valor es congruente no solo con el carácter didáctico de la plataforma científica, sino también con los objetivos de la investigación.

Por otro lado, es importante destacar que en función del tipo de articulación se considerará la velocidad máxima positiva o negativa. En el caso de articulaciones rotacionales, es independiente considerar la velocidad máxima positiva o negativa, ya que a efectos de requerimientos de par, no hay diferencia en girar la articulación en un sentido o en otro.

[14] Para mayor información técnica detallada sobre los actuadores Dynamixel y sus librerías, visitar http://support.robotis.com/en/.

3.5. Diseño Mecánico Avanzado

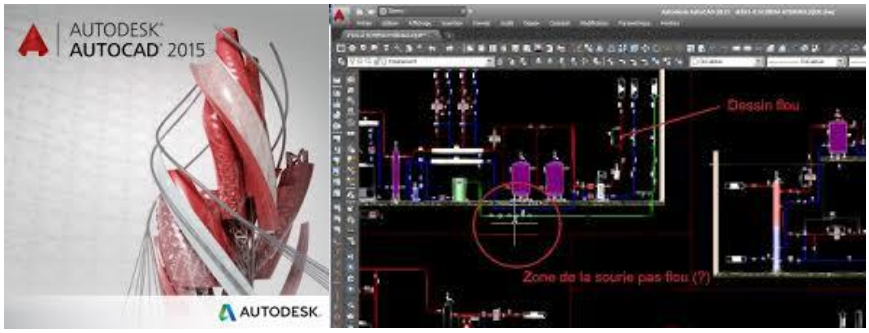
La gran mayoría de sistemas robotizados están conformados por plataformas mecánicas muy complejas, derivando en análisis muy complicados desde un punto de vista teórico. En la actualidad, gracias a los avances tecnológicos, se dispone de diferentes soluciones informáticas que, de modo rápido y sencillo, pueden modelar y simular sistemas mecánicos muy complejos. Este tipo de software es denominado de forma general como Herramienta Informática de Diseño Mecánico Avanzado (HI-DMA).

La utilización de una computadora como elemento de ayuda para las diferentes actividades de diseño, ha cobrado tal importancia que hoy en día resulta prácticamente inconcebible subsistir en un mundo tan competitivo sin su utilización. Por ello, en el presente epígrafe se abordará el tema del software como una valiosa herramienta de apoyo al diseñador para el desarrollo de robots.

Las HI-DMA no se utilizan únicamente para validar los resultados teóricos procedentes de las etapas previas de análisis, sino que mantienen relaciones de realimentación a lo largo de todo el proceso de diseño.

3.5.1. Diseño e Ingeniería Asistida por Computadora (CAX)

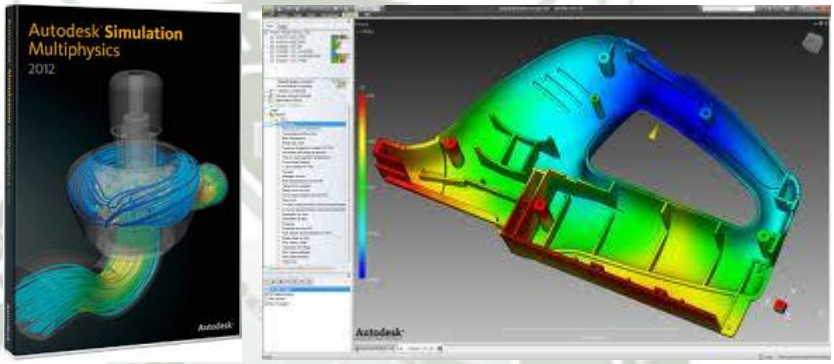
Existen herramientas de Diseño Asistido por Computadora (CAD), como el Autodesk® AutoCAD®, que permiten desarrollar bocetos de componente o piezas, ya sea en 2D o 3D, los cuales se basan en el trazado de figuras geométricas (puntos, líneas, circunferencias, arcos, etc.) para generar sólidos y superficies. En general, estos sistemas incluyen diversos asistentes para la elaboración de agujeros, roscas, avellanados, chaveteros, entre otros, todo ello cumpliendo con la normativa aplicable a dibujo técnico, lo cual facilita el diseño de piezas mecánicas. A su vez, estos sistemas contienen librerías de piezas normalizas y admiten realizar ensamblajes de piezas, presentaciones fotorrealistas, planos y traslación a un código compatible con sistemas de manufactura.



Fuente: "AUTOCAD 2015", Autodesk Community, 2014.

Fig. 3.15. CAD a través de Autodesk® AutoCAD® 2015.

También existen sistemas de Ingeniería Asistida por Computadora (CAE), como Autodesk® Simulation Multiphysics (Algor) o Ansys®, que ofrecen una serie de herramientas de análisis y simulación para ayudar a los diseñadores a tomar decisiones respecto a sus características y viabilidad. Su finalidad es optimizar el desarrollo del producto, minimizar los consecuentes costes de fabricación y reducir al máximo las pruebas. La mayoría de las herramientas CAE, que generalmente se presentan como aplicaciones independientes de los sistemas CAD, están formadas por los siguientes módulos de análisis: estructural, de movimiento, térmico, de fluidos y electromagnético (pudiendo estudiar la combinación simultánea de varios fenómenos).



Fuente: "Autodesk Simulation", Cadac Group, 2013.

Fig. 3.16. Análisis de Elementos Finitos (FEA) con Algor.

Sin embargo, últimamente se acuña el término de Diseño e Ingeniería Asistida por Computadora (CAX), para referirse a aplicaciones informáticas que integran filosofías CAD y CAE (e incluso otras adicionales, como por ejemplo las de Manufactura Asistida por Computadora (CAM)), y cuya tendencia va tomando protagonismo en el Diseño Mecánico Avanzado. Muestra de ello son los software SolidWorks® y Autodesk® Inventor® Professional.

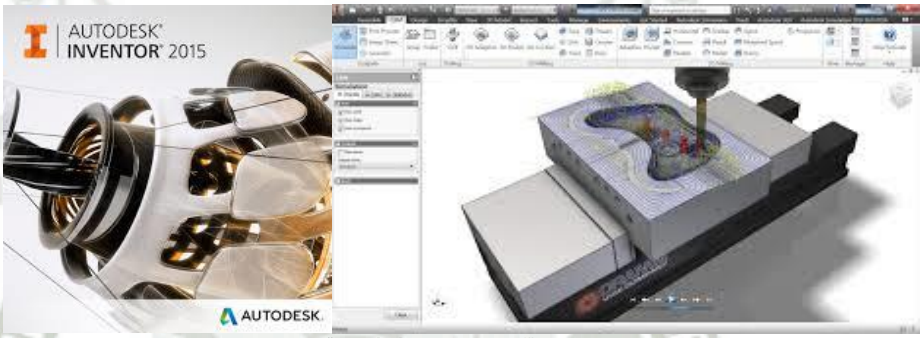
Diseño e Ingeniería Asistida con Autodesk® Inventor® Professional

Autodesk® Inventor® Professional generalmente se emplea para construir diseños de sistemas mecánicos tridimensionales, utilizando para ello tecnología adaptativa. Permite establecer modelos virtuales con sus respectivos planos de fabricación en dos dimensiones, y con su capacidad de adaptación, se consigue realizar ensamblajes de los diferentes modelos creados por separado para así crear máquinas muy complejas.

Con este software, es posible crear piezas y ensambles basados en parámetros de diseño reales como velocidad, potencia, fuerza, propiedades del material, etc., que son de gran importancia para la elección de

actuadores y el control del sistema. A partir de este tipo de información se podrán crear componentes de conexión como tornillos, ejes, engranes, cadenas, correas, tornillos de transmisión y muelles, y así poder desarrollar conjuntos mecánicos en 3D para la representación de los sistemas.

A su vez, con esta herramienta se podrán crear planos automáticos, reduciendo el tiempo de diseño del dibujo mediante la generación automática de vistas frontales, laterales, isométricas, seccionadas y auxiliares de las piezas, así como vistas para ensamble de componentes mecánicos, acotaciones, anotaciones y símbolos basados en normas de dibujo.



Fuente: "Autodesk Inventor 2015", Synergis, 2014.

Fig. 3.17. CAM integrado en Inventor® Professional.

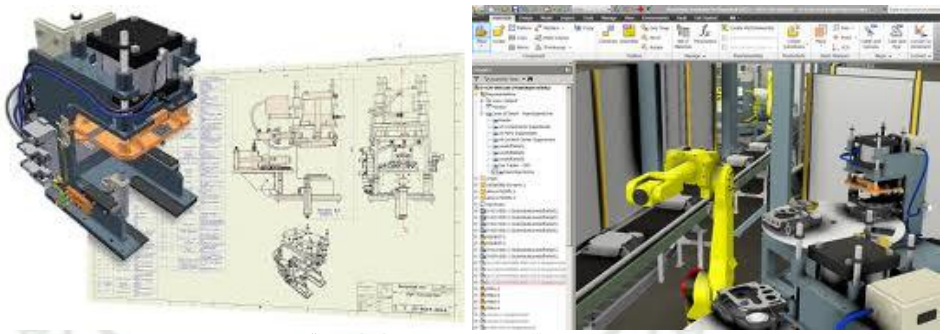
Los modelos virtuales son paramétricos, pues en el caso de haber un cambio en los bocetos se actualizan los datos, el cual se produce en todos los planos generados. Se incluye la lista de materiales del sistema para una mejor presentación del despiece.

Una vez finalizada la etapa de diseño, será posible simular y analizar el sistema para conocer su comportamiento en equilibrio estático, determinando así tensiones y flexiones bajo cargas, frecuencias de resonancia, entre otras. De este modo es posible minimizar los esfuerzos y reducir los costes del material, sin perjudicar el rendimiento de la máquina.

A su vez, empleando el módulo de análisis dinámico de movimiento, se podrá predecir la cinemática y dinámica de los cuerpos en movimiento. De hecho, será posible obtener resultados de fuerzas/torques y aceleraciones lineales/angulares, entre otras variables, que experimenta cada uno de los componentes ante la presencia de cargas variables y componentes de fricción como muelles y amortiguadores.

Además, se puede definir una envolvente de referencia para el desplazamiento del sistema y grabar las animaciones. La aplicación de este conjunto de conocimientos y técnicas indicados posteriormente llevará a

conseguir una mejora del ciclo de desarrollo y fabricación del sistema robótico. Autodesk® Inventor® Professional fue el software elegido para el CAX del robot.



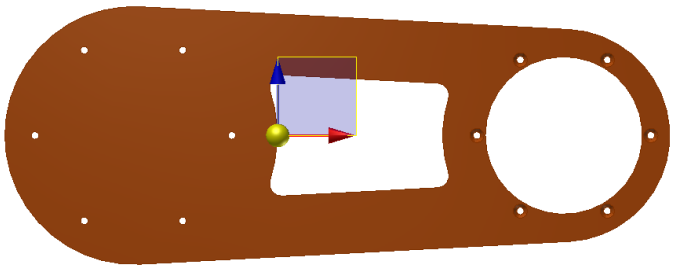
Fuente: "Autodesk inventor LT 2015", Get Into PC, 2014.

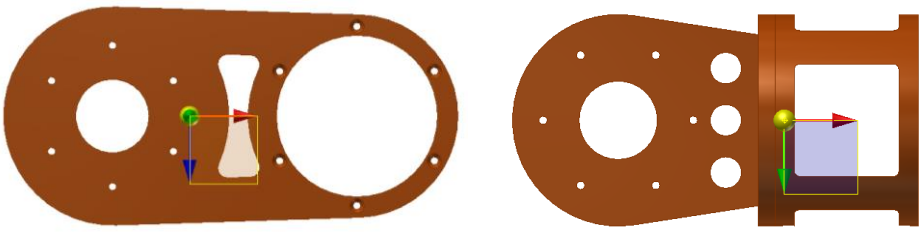
Fig. 3.18. Planos y animación de un sistema robotizado.

Geometría de eslabones y juntas

El objetivo de obtener toda la información desarrollada empleando los procedimientos cinemáticos antes detallados, no sólo sirve para tener el sistema robótico definido en cuanto a su movimiento, sino que es necesario emplearla para definir las dimensiones óptimas del robot, cuya estructura mecánica está compuesta por eslabones y juntas. La geometría de ambos se utiliza para completar la síntesis cinemática y determinar qué valores deben asignarse a la longitud de cada eslabón y cuál es la disposición y recorrido de las coordenadas articulares. Determinando dichos parámetros, será posible definir las trayectorias que podrán realizarse y el volumen de trabajo accesible en el espacio 3D.

La Fig. 3.19. muestra los parámetros cinemáticos de los eslabones 2, 3 y 4 del robot. Para su diseño se tomaron las siguientes consideraciones: minimizar el peso en lo posible, priorizar una geometría simétrica que facilite su fabricación, eliminar bordes y terminaciones en ángulo recto, y localizar el centro de gravedad cerca al extremo izquierdo, que corresponde a la zona de montaje de cada eslabón. Para completar la alegoría entre el brazo humano y el brazo robótico, las longitudes de los eslabones son equivalentes a las de un brazo humano promedio.

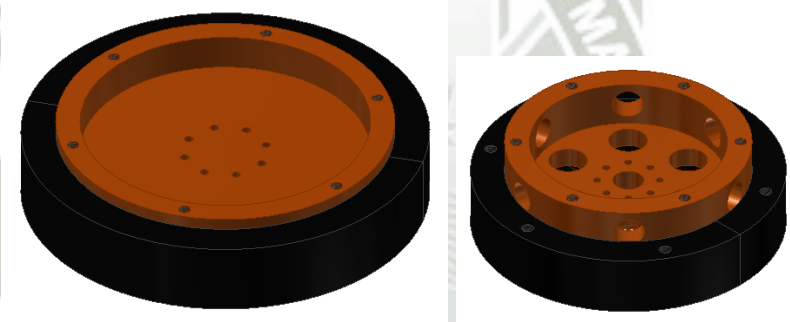




Fuente propia.

Fig. 3.19. Parámetros cinemáticos de los eslabones 2, 3 y 4, respectivamente.

La Fig. 3.20. muestra los dos tipos de juntas rotativas diseñadas, donde se instalarán los actuadores. La unión junta-actuador constituyen las articulaciones físicas del brazo robótico. Cada junta está compuesta por una carcasa (sección negra) que alberga un disco, permitiendo la rotación de este último. El disco es acoplado al eje del actuador, el cual aporta el movimiento rotacional deseado. Su diseño, al igual que en el caso de los eslabones, pretende minimizar el peso, optando por una geometría de revolución (muy útil para el mecanizado) que sea capaz de albergar al actuador correspondiente.



Fuente propia.

Fig. 3.20. Tipos de juntas rotativas diseñadas.

El detalle dimensional de todas las piezas del robot ha sido plasmado en los planos correspondientes, bajo el rigor de la normativa del dibujo técnico, los cuales pueden encontrarse en la sección de Anexos.

Creación Conceptual

Una vez finalizadas las etapas de diseño conceptual y análisis cinemático y dinámico, se hace uso de toda esta información para realizar la creación conceptual del robot, es decir, definir al detalle el modelo final óptimo de la estructura robótica que será fabricada, instalada y puesta en funcionamiento. A este nivel de análisis, se está en la capacidad de generar esquemas y bocetos descriptivos del sistema robótico, que progresivamente fueron perfeccionados hasta alcanzar el diseño óptimo. Dichos bocetos son obtenidos mediante un diseño conceptual vía HI-DMA, a través de Autodesk® Inventor® Professional. La Fig. 3.21. expone el diseño definitivo

del robot Darko, con los actuadores instalados en cada articulación. Se eligieron uniones roscadas normalizadas (al no ser permanentes, facilitan su despiece para el mantenimiento) del tipo tornillo metálico con rosca métrica ISO.[15]

Fuente propia.

Fig. 3.21. Modelo definitivo del robot Darko.

Estimación de masas e inercias

Inventor® permite seleccionar el material de fabricación y acabado superficial de cada pieza. Definida la geometría y el material, el *iProperties* de Autodesk® calcula una serie de parámetros físicos, muy valiosos no solo para la optimización de la creación conceptual, sino también para la el modelamiento dinámico del robot. Estos parámetros se dividen en: Propiedades Generales y Propiedades Inerciales.

Dentro de las propiedades generales se detallan los valores de masa, volumen, área y centro de gravedad, mientras que las propiedades inerciales albergan los valores de los momentos de inercia principales, calculados respecto a cada eje cartesiano, y el tensor de inercia, calculado tanto respecto al origen de coordenadas como al centro de gravedad. La Tabla 3.6. recopila la estimación de masas y detalla los materiales a utilizar para la fabricación de cada pieza, mientras que en la Tabla 3.7. se brinda información sobre los parámetros físicos de los eslabones y sus respectivos tensores de inercia.

[15] Se utilizaron dos tipos de tornillos: Tornillos Socket Allen ISO 4762 (cabeza cilíndrica) y Tornillos Flat Allen ISO 10642 (cabeza avellanada).

Tabla 3.6. Estimación de masas y materia prima de las piezas del robot.

Pieza o dispositivo	Material	Cantidad	Masa (kg)	Materia Prima
Dynamixel MX-64T	Plástico	1	0.125	
Arm Base	Nylon 6	1	0.210	Barra de Ø150 mm x 80 mm
Casing	Nylon 6	4	0.192	Barra de Ø110 mm x 45 mm
Ring	Nylon 6	2	0.172	Barra de Ø110 mm x 50 mm
Base Joint	Nylon 6	1	0.106	Barra de Ø120 mm x 28 mm
Dynamixel MX-64T	Plástico	1	0.125	
Shoulder Mount	Nylon 6	1	0.098	Barra de Ø120 mm x 65 mm
Shoulder Elbow	Nylon 6	1	0.167	Plancha de 6 mm x 350 x 115
Ring 1	Nylon 6	2	0.092	Barra de Ø80 mm x 50 mm
Casing 3	Nylon 6	1	0.027	Barra de Ø85 mm x 25 mm
Casing 4	Nylon 6	1	0.025	Barra de Ø85 mmx 25 mm
Dynamixel MX-28T	Plástico	1	0.072	
Elbow Wrist	Nylon 6	1	0.086	Plancha de 6 mm x 350 x 95
Casing 1	Nylon 6	1	0.026	Barra de Ø85 mm x 25 mm
Casing 2	Nylon 6	1	0.026	Barra de Ø85 mm x 25 mm
Dynamixel AX-12A	Plástico	1	0.055	
Wrist Hand	Nylon 6	1	0.104	Barra de Ø85 mm x 130 mm
Wrist Case	Nylon 6	1	0.078	Barra de Ø85 mm x 75 mm
Dynamixel AX-12W	Plástico	1	0.053	
Casing 5	Nylon 6	2	0.034	Barra de Ø85 mm x 25 mm
Ring 2	Nylon 6	1	0.047	Barra de Ø80 mm x 25 mm
Tower Pro SG 5010	Plástico	1	0.053	
Shoulder Elbow Case	Acrílico	1	0.098	Plástico inyectado
Elbow Wrist Case	Acrílico	1	0.053	Plástico Inyectado
Gripper	Plástico	1	0.045	Plástico Inyectado
TOTAL		**31**	**2.169**	

Fuente propia.

Tabla 3.7. Parámetros físicos de los eslabones.

Eslabón	Masa (kg)	Longitud (m)	Centro de gravedad	Tensor de Inercia (*10^{-3} kg.m^2)
1	0.624	0.167	(0,-0.101,0)	$\begin{bmatrix} 1.243 & -0.015 & 0 \\ -0.015 & 1.272 & 0 \\ 0 & 0 & 1.268 \end{bmatrix}$
2	0.759	0.181	(-0.129,0,0)	$\begin{bmatrix} 5.459 & 0 & 0.004 \\ 0 & 5.723 & -0.145 \\ 0.004 & -0.145 & -0.659 \end{bmatrix}$
3	0.303	0.100	(-0.053,0,0)	$\begin{bmatrix} 1.644 & 0 & 0 \\ 0 & 0.214 & -0.137 \\ 0 & -0.137 & 1.747 \end{bmatrix}$
4	0.392	0.133	(0,0,0.081)	$\begin{bmatrix} 0.318 & 0.021 & 0.005 \\ 0.021 & 0.95 & 0 \\ 0.005 & 0 & 0.908 \end{bmatrix}$

Fuente propia.

Capítulo IV

Control Dinámico del brazo robótico

4.1. Control PID con compensación de gravedad

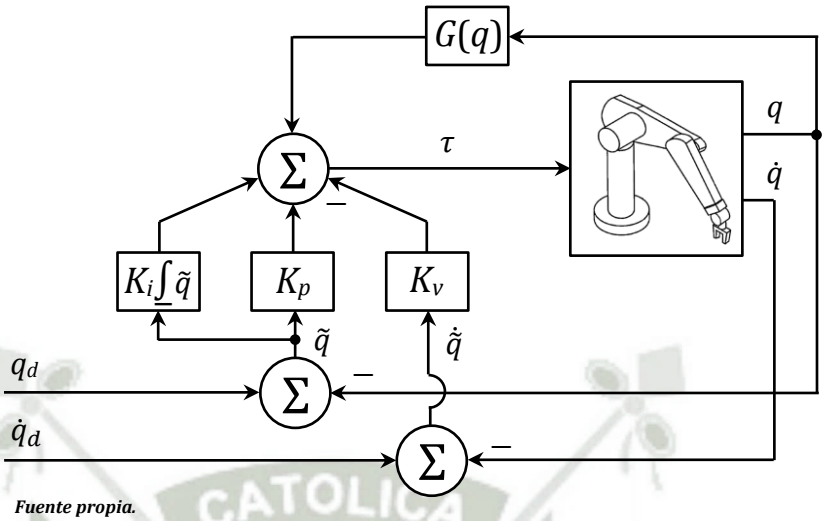
Una vez obtenidos los modelos cinemáticos y dinámicos, es posible abordar la etapa más compleja de toda la investigación: el control del robot. Dicha etapa está dividida en dos fases. La primera compuesta por el Control Dinámico, y la segunda por el tema cumbre de la investigación: el Control Kinect.

En el presente capítulo se desarrolla el Control Dinámico del brazo robótico, como un requisito preliminar para alcanzar el control de la segunda fase. Es preciso recordar que el objetivo de llevar a cabo este tipo de control, no es alcanzar una amplia versatilidad y exigencia de movimientos que conduzcan hacia una etapa posterior de programación (como si sucede en los robots industriales comerciales). Por ello, se realizará un estudio con el nivel de profundidad necesario y suficiente para poder alcanzar un control aceptable.

De aquí en adelante, se denomina Control Dinámico a aquel basado en el modelo dinámico del robot. Busca definir el movimiento del robot, de manera que siga un camino planificado. Así, el objetivo es establecer cuáles son las trayectorias que debe seguir cada articulación del robot a lo largo del tiempo, para conseguir los objetivos fijados, a la vez que se exige cumplir una serie de restricciones físicas impuestas por los actuadores, y de calidad de la trayectoria, como son suavidad, precisión, entre otras.

Se utiliza la técnica de control PID con compensación de gravedad (PID + G). Como se vio en el Capítulo II, ésta técnica es utilizada, en su forma más genérica, para el control de posición. Sin embargo, posee una variante para el control de movimiento, muy poderosa y efectiva, la cual se va a utilizar para materializar el Control Dinámico del robot Darko. El diagrama de bloques de este tipo de control se representa en la Fig. 4.1.

Debido a la alta complejidad que involucra el control de un robot de 5 GDL, y con la finalidad de ser más didácticos, se procedió a dividir el proceso de control en dos etapas. Una primera etapa, compuesta por el control de los 3 primeros GDL (grados de posición), y la segunda, conformada por el control de los 5 GDL (aquí se añaden a los anteriores, los grados de orientación).

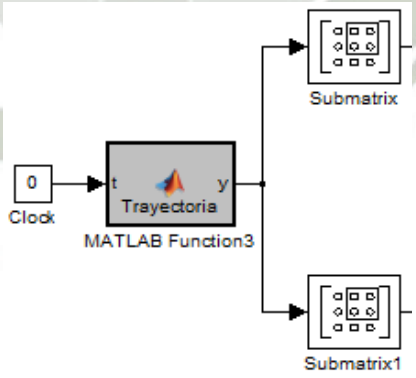


Fuente propia.

Fig. 4.1. Diagrama de bloques del control de movimiento PID+G.

Control de los 3 GDL

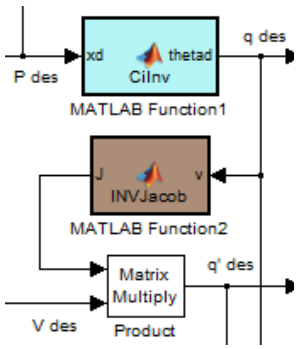
En el control de los primeros 3 GDL, se trabaja con los grados que únicamente posicionan el extremo del robot en un punto del espacio determinado. Este detalle permite que, las trayectorias que se vayan a trazar para evaluar la calidad del control, sean del tipo paramétrico y en el espacio cartesiano, características que facilitan el control y convierten la presencia obligatoria del bloque interpolador de trayectorias articulares, en opcional. Así, como primer bloque del esquema de control, se tiene un planificador de trayectorias paramétricas (Fig. 4.2), que generará curvas de posición (x, y, z) y velocidad (v_x, v_y, v_z).



Fuente propia.

Fig. 4.2. Bloque planificador de trayectorias paramétricas.

El esquema de control de la Fig. 4.1. exige como datos de entrada, la posición q_d y velocidad $\dot{q_d}$ articulares deseadas. Como el planificador de trayectorias entrega posición y velocidad del extremo del robot en el espacio cartesiano, es necesario llevarlas al espacio articular. Para lograrlo, se utilizan los bloques de Cinemática Inversa y Jacobiana Inversa, respectivamente (Fig. 4.3.).

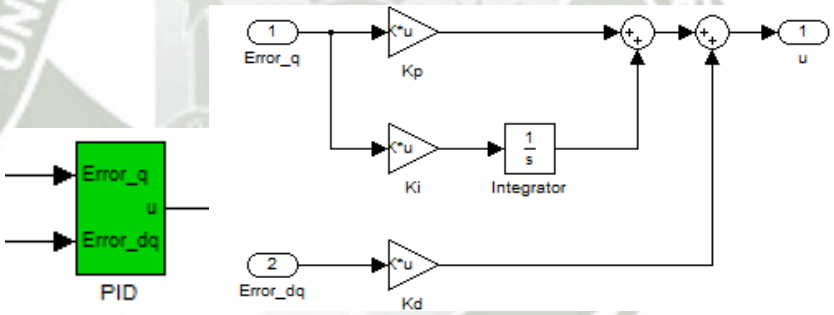


Fuente propia.

Fig. 4.3. Bloques de Cinemática Inversa y Jacobiana Inversa.

No debe olvidarse que la Jacobiana es única para cada configuración articular, por lo que tiene como entrada la posición articular deseada. Multiplicándola con las velocidades del extremo, se obtienen las velocidades articulares deseadas.

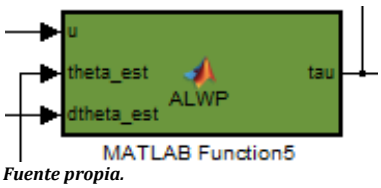
Obtenidas la posición y velocidad articulares deseadas, es posible construir el controlador de la Fig. 4.1. Así, se elabora el bloque PID, cuyas entradas son los errores de posición y velocidad articulares. El error se obtiene de restar la señal deseada con la señal estimada, real o de realimentación. La Fig. 4.4. muestra el bloque PID diseñado y su estructura interna.



Fuente propia.

Fig. 4.4. Bloque de control PID.

La salida del bloque PID entrega una señal de control u. Sin embargo, la planta a controlar, es decir el robot, admite solo señales de torque. Es aquí donde el Algoritmo Luh-Walker-Paul toma protagonismo, debido a que permite calcular de forma muy eficiente y rápida los torque necesarios para cada actuador. Siguiendo paso a paso el algoritmo recursivo, se elabora el bloque ALWP (Fig. 4.5.), cuyas señales de entrada son la señal de control u, y las señales articulares de posición y velocidad estimadas.



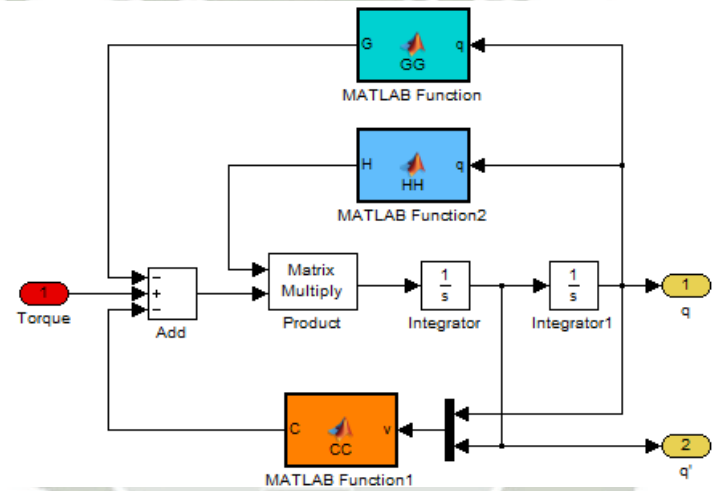
Fuente propia.

Fig. 4.5. Bloque del Algoritmo Luh-Walker-Paul.

A continuación, se procede a implementar matricialmente el robot, que como se dijo, es la planta a controlar. Para lograrlo, es necesario despejar la aceleración $\ddot{q}$ de la ecuación que describe el modelo dinámico, de la siguiente manera:

$$\ddot{\boldsymbol{q}} = \mathbf{H}(\boldsymbol{q})^{-1}[\boldsymbol{\tau} - \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathbf{G}(\boldsymbol{q})]$$

Las matrices H, C y G ya se obtuvieron en el capítulo anterior, gracias al algoritmo computacional de Lagrange-Euler. Así, y gracias al despeje realizado, es posible integrar la aceleración articular $\ddot{q}$ para obtener la velocidad articular estimada, y si se integra nuevamente, se tiene la posición articular estimada. Ambas señales son importantes, pues el éxito del control depende del grado de similitud que tengan con sus correspondientes señales deseadas. En la Fig. 4.6. se muestra la planta dinámica, que constituye el modelamiento ideal de un robot (bloque Robot Sims).



Fuente propia.

Fig. 4.6. Planta dinámica del robot.

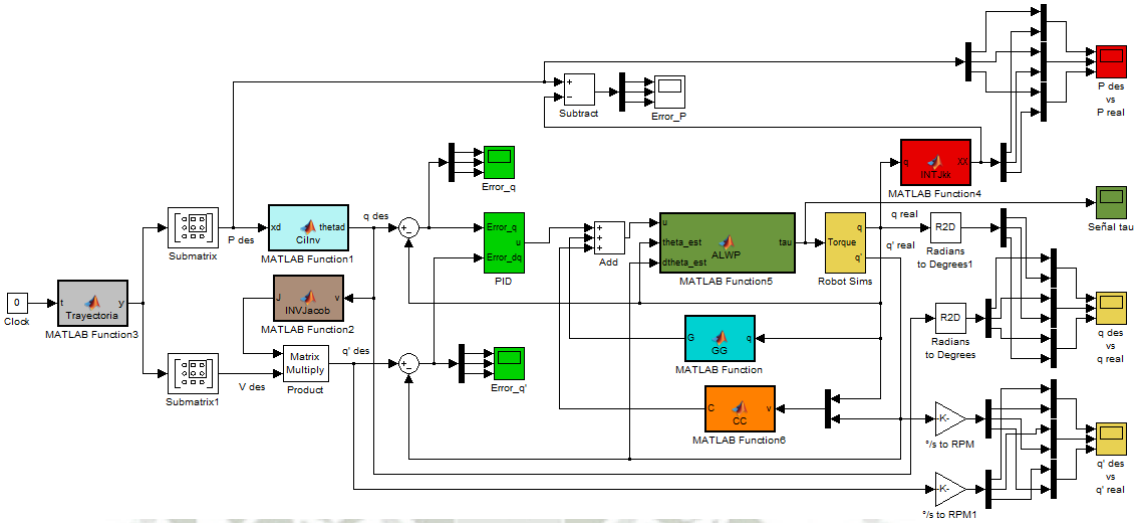
Finalmente, para implementar correctamente el control PID+G, se extrae la matriz de pares gravitacionales G, y se realimenta de tal manera que la señal de control u final quede descrita de la siguiente forma:

$$\boldsymbol{u} = \boldsymbol{K_p}\tilde{\boldsymbol{q}} + \boldsymbol{K_v}\dot{\boldsymbol{q}} + \boldsymbol{K_i}\int_0^t \tilde{\boldsymbol{q}}(\boldsymbol{t})\boldsymbol{dt} + \mathbf{G}(\boldsymbol{q})$$

Todo el Control Dinámico descrito para los 3 primeros GDL se implementó en la herramienta Simulink® de Matlab®, debido a que su lenguaje de programación por bloques permite una explicación más didáctica de lo realizado; así, el lector puede comprender de forma integral todo el proceso.

El esquema de control PID+G final se expone en la Fig. 4.7. La sintonización de las ganancias proporcionales, integrales y derivativas se realizaron por métodos experimentales. A través del *scope* es posible visualizar diversas señales

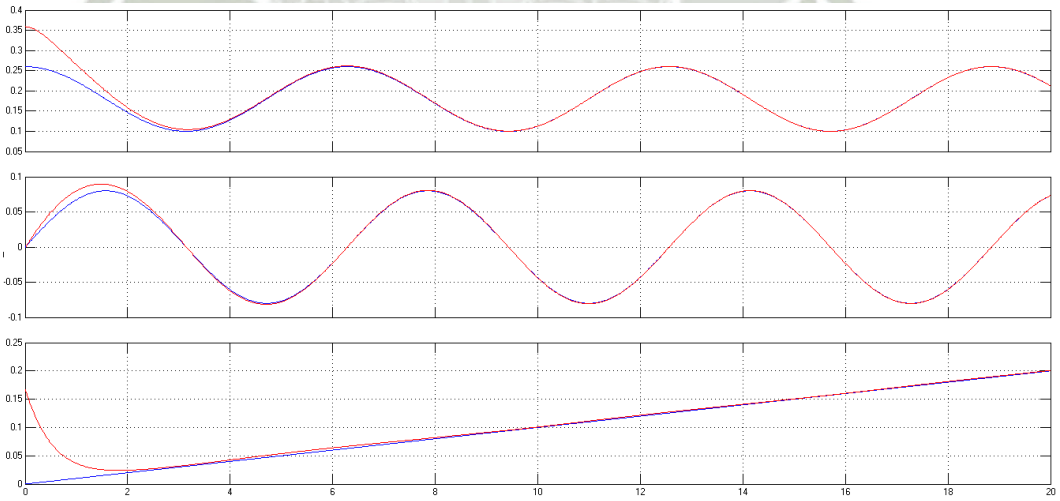
trascendentes descritas en el tiempo, cuya comparación permite verificar la calidad del control realizado.



Fuente propia.

Fig. 4.7. Control PID+G para 3 GDL.

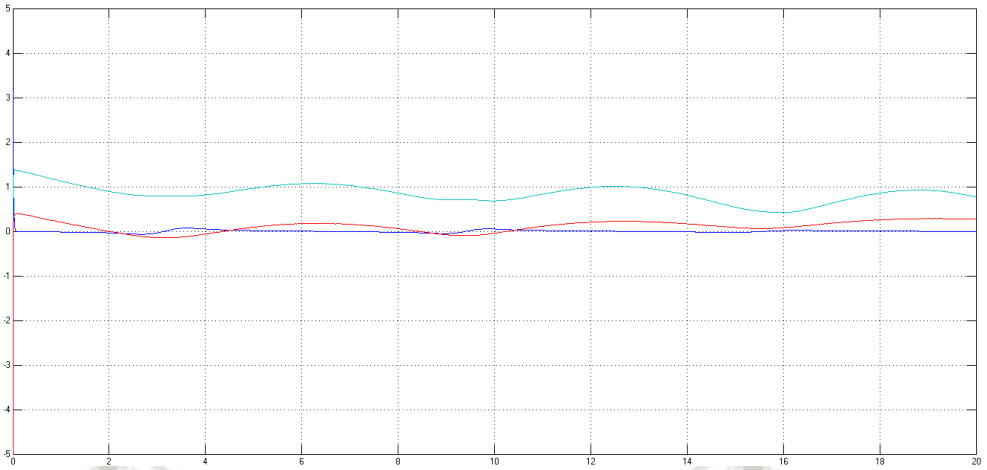
Seleccionando como trayectoria una curva helicoidal "tipo horquilla" ascendente en el eje z, se ejecuta el programa *Darko.mdl* que alberga todo el control PID+G de los tres primeros grados de libertad del robot.. La Fig. 4.8. captura las gráficas del *scope* rojo, el cual compara la posición del extremo deseada con la real, ambas en metros.



Fuente propia.

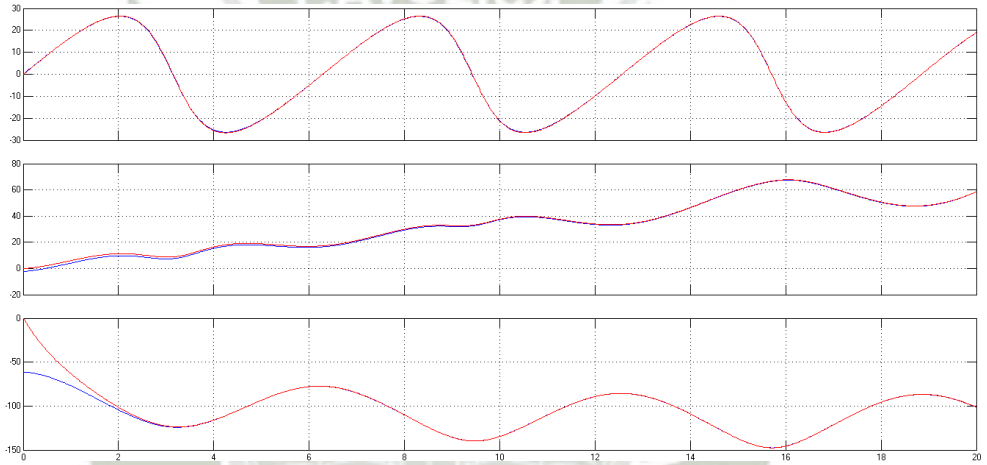
Fig. 4.8. Posición del extremo deseada (azul) vs. Posición del extremo real (roja).

La Fig. 4.9. muestra las tres señales de torque (en N.m) correspondientes a cada GDL, obtenidas después de aplicar el Algoritmo Luh-Walker-Paul (*scope* verde oscuro), mientras que en las Fig. 4.10. y 4.11. se comparan las señales de posición y velocidad articulares deseadas con las reales, respectivamente (*scopes* amarillos). Todas las señales en el espacio articular se encuentran en grados sexagesimales.



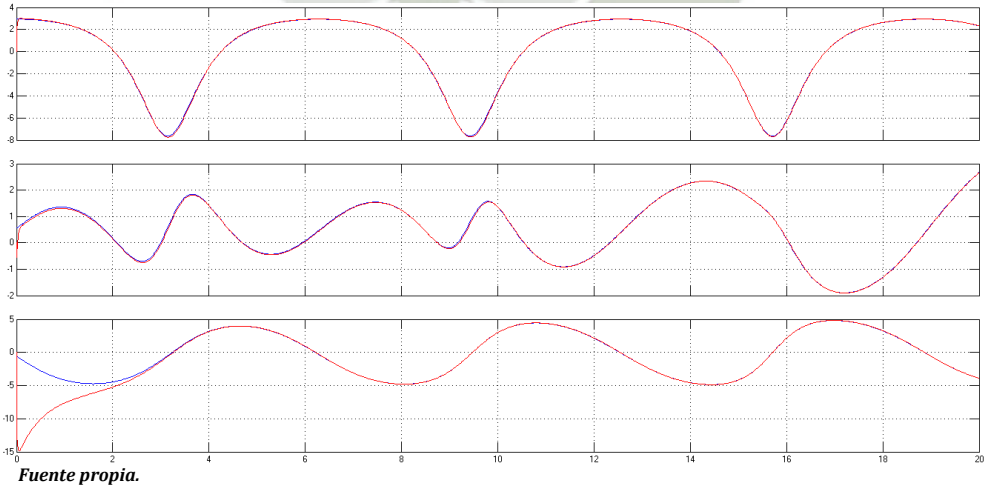
Fuente propia.

Fig. 4.9. Señales de torques aplicados en cada GDL.



Fuente propia.

Fig. 4.10. Posición articular deseada (azul) vs. Posición articular real (roja).

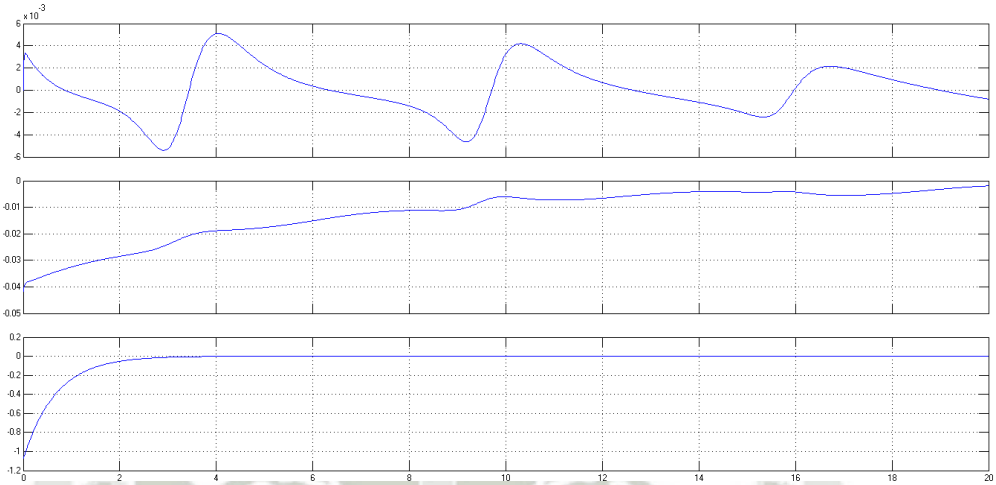


Fuente propia.

Fig. 4.11. Velocidad articular deseada (azul) vs. Velocidad articular real (roja).

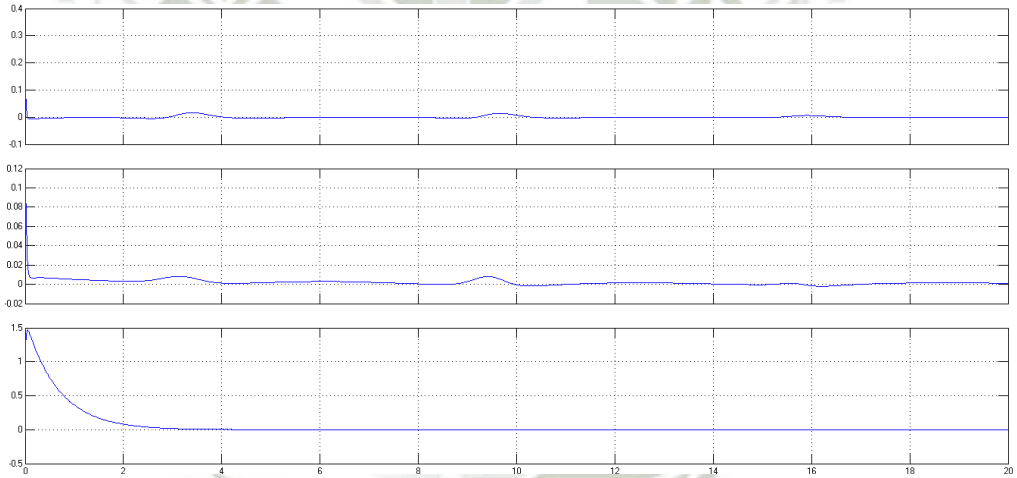
Por otro lado, las Fig. 4.12., 4.13. y 4.14. muestran los errores de posición articular, velocidad articular y posición del extremo del robot, respectivamente. En todos los casos, los errores poseen valores muy pequeños, y sus señales presentan una clara

tendencia de aproximarse siempre a cero, que es lo ideal, ya que en toda estrategia de control se busca reducir al máximo la magnitud del error.



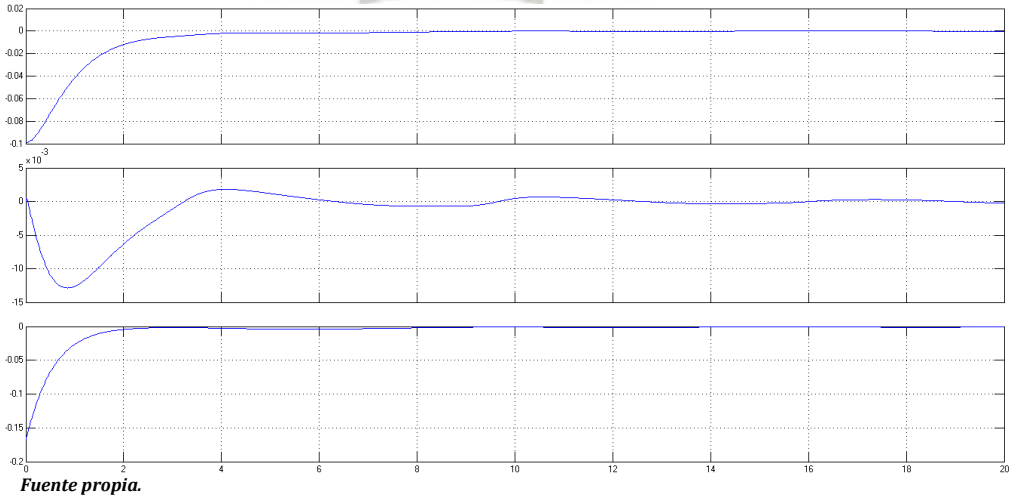
Fuente propia.

Fig. 4.12. Error de la posición articular.



Fuente propia.

Fig. 4.13. Error de la velocidad articular.



Fuente propia.

Fig. 4.14. Error de la posición del extremo del robot.

Para todas las señales mostradas, la curva azul es la señal deseada y la curva roja es la real, la obtenida a la salida del modelo dinámico del robot, y como se puede apreciar, ambas se aproximan muy de cerca, y en algunos casos prácticamente se superponen, evidencia contundente de que se logró un excelente control.

Si bien, a través de Simulink® se puede apreciar y entender mejor la construcción del esquema de control del robot, no es muy apropiado para la implementación del control real, principalmente porque su velocidad de cálculo y procesamiento de datos no es la más recomendable.

Para solucionar este inconveniente, se crea un *M file* (*script* típico de Matlab®), el cual permite llevar a líneas de código programado todo lo elaborado en Simulink®. La programación en código utilizando el lenguaje de Matlab®, permite alcanzar velocidades de cálculo y procesamiento más rápidas que en Simulink® y, entre otras múltiples ventajas, posee una interfaz tan flexible y versátil que es posible llevar a cabo una programación muy personalizada y completa, que satisfaga los objetivos del propio diseñador.

Así, se elaboró el script *Robot_PID_G* que cobija todo el control descrito líneas arriba en el formato de código programado. Dicho script incluye una animación virtual 3D del robot, en la cual es posible ver el desarrollo de la trayectoria seleccionada en toda su magnitud. Ésta animación 3D permite evaluar la calidad del control propuesto, y aproximar muy de cerca lo que sería el comportamiento ideal del robot en el espacio de trabajo real.

Se trazaron 5 tipos de trayectorias, para poner a prueba la efectividad del control de los 3 GDL. En todas ellas, se detallan los valores articulares máximos y mínimos alcanzados por cada articulación, la integral del módulo del vector de error (indicador importante para saber qué tan bueno es el control), y se muestran 5 ventanas que grafican: la trayectoria cartesiana deseada o de referencia con la real trazada por el robot, los torques de control calculados para cada articulación, las trayectorias de las posiciones y velocidades articulares de cada GDL, el módulo del vector error en el espacio de trabajo y la animación del movimiento en 3D.

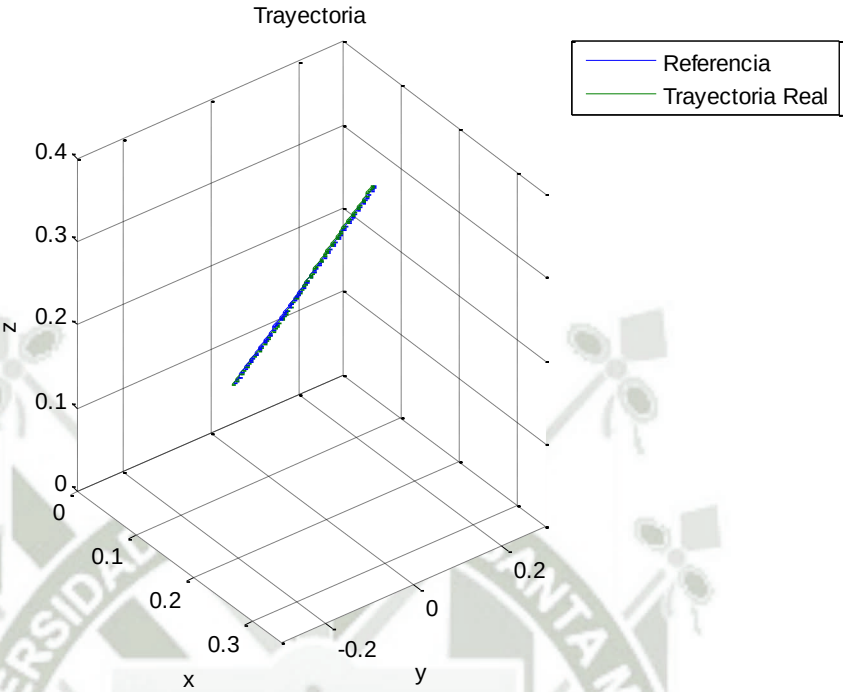
Los resultados para cada trayectoria, se muestran a continuación:

- **Trayectoria en Línea Recta**

  Los valores articulares máximos y mínimos son:

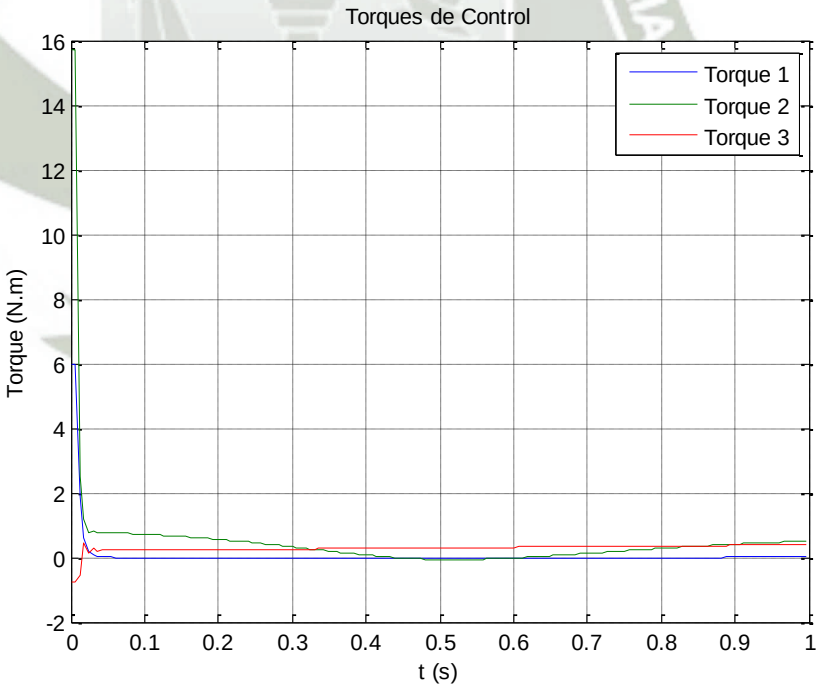
	1er GDL	**2do GDL**	**3er GDL**
Valor máximo	56.4597	91.7676	-77.9536
Valor mínimo	-45.0000	53.9344	-128.9613

La integral del módulo del vector error es: 0.0023.



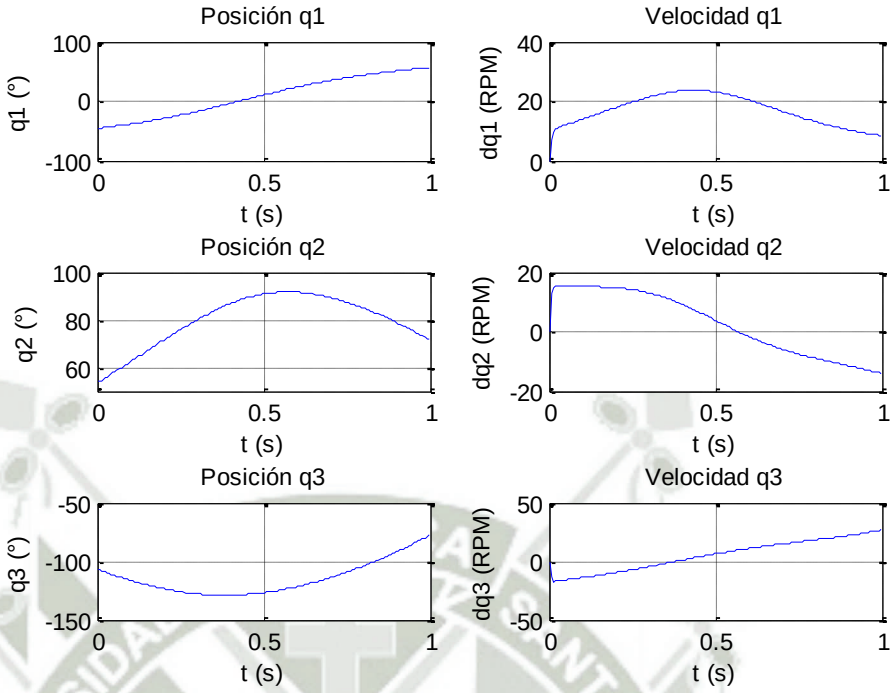
Fuente propia.

Fig. 4.15. Trayectoria en línea recta.



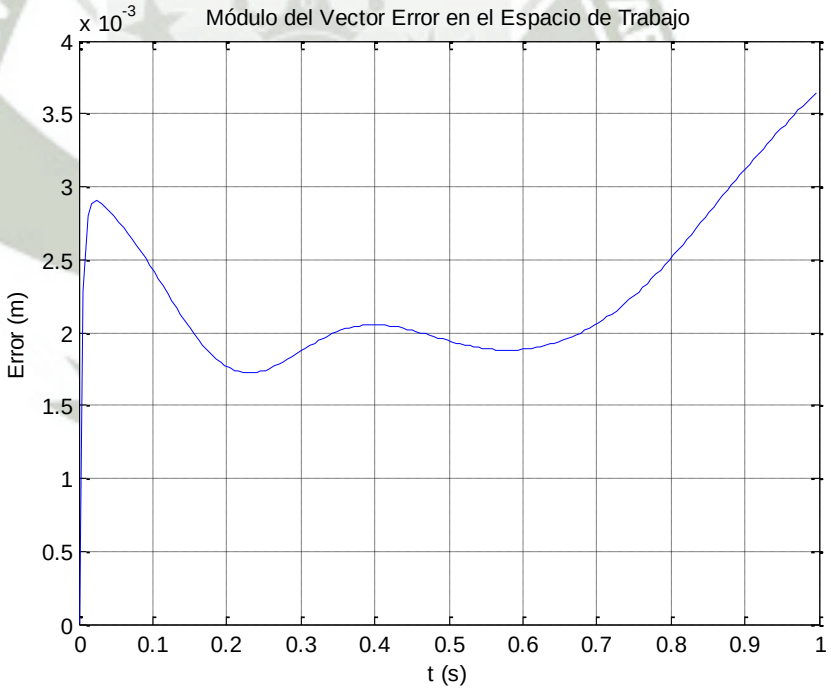
Fuente propia.

Fig. 4.16. Torques de control calculados para cada articulación.



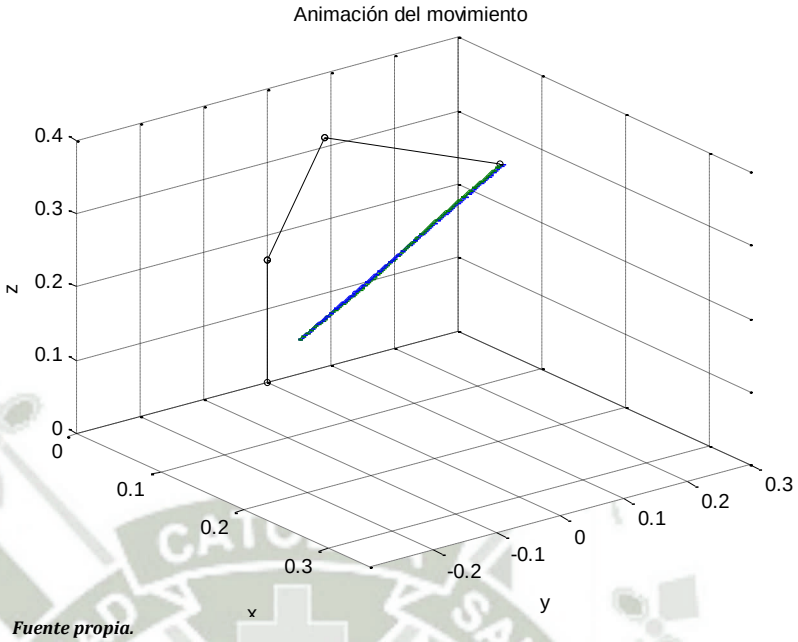
Fuente propia.

Fig. 4.17. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.18. Módulo del vector error en el espacio de trabajo.



Fuente propia.

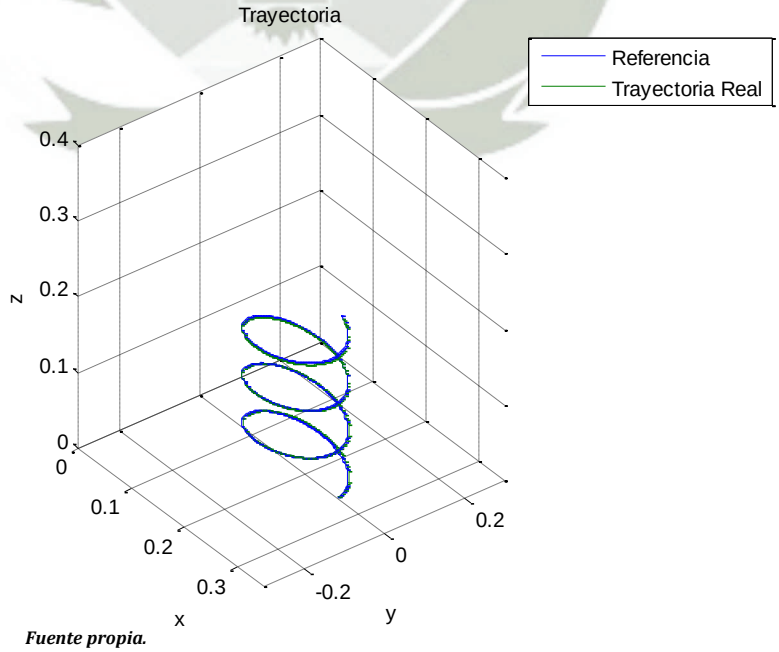
Fig. 4.19. Animación del movimiento.

- **Trayectoria helicoidal "horquilla" ascendente en el eje z**

Los valores articulares máximos y mínimos son:

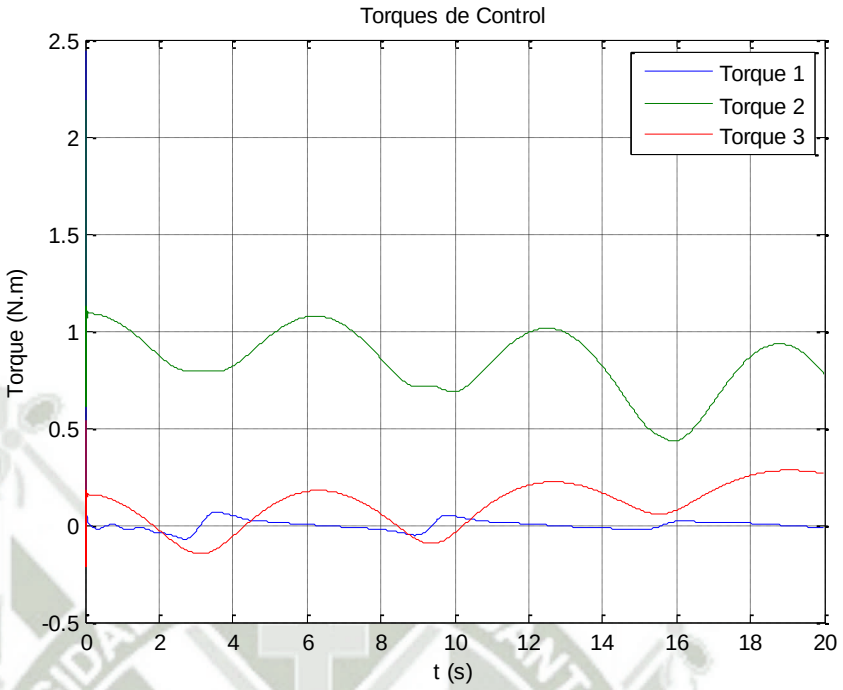
	1er GDL	2do GDL	3er GDL
Valor máximo	26.4643	66.3675	-61.1773
Valor mínimo	-26.3966	-2.3970	-147.4538

La integral del módulo del vector error es: 0.0431.



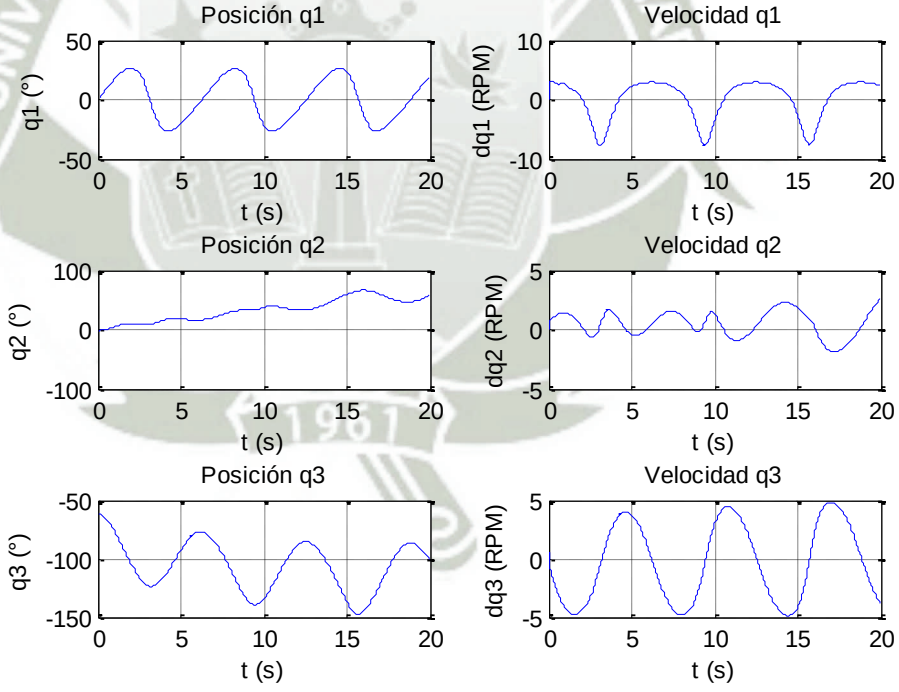
Fuente propia.

Fig. 4.20. Trayectoria helicoidal "horquilla" ascendente en el eje z.



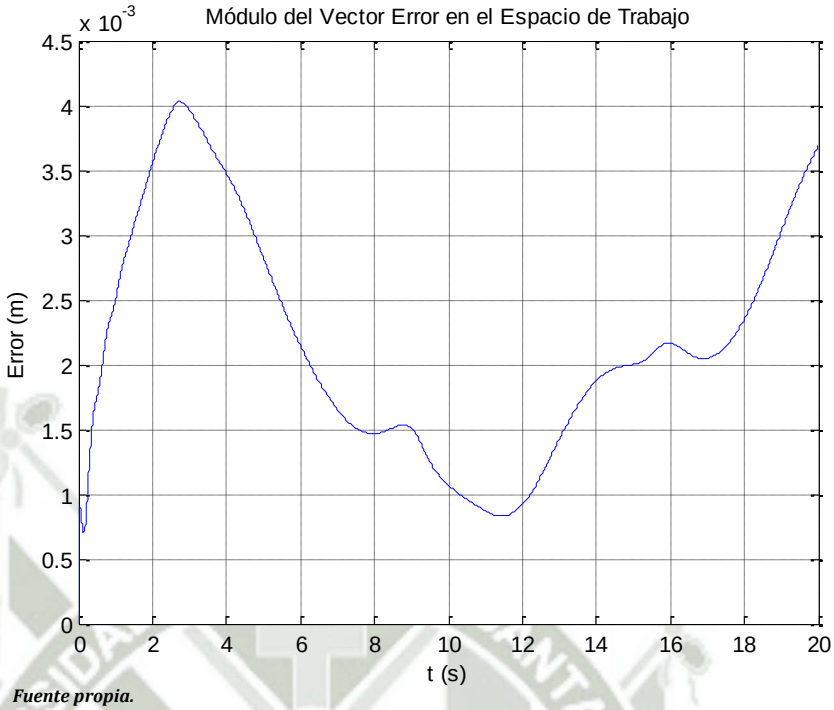
Fuente propia.

Fig. 4.21. Torques de control calculados para cada articulación.



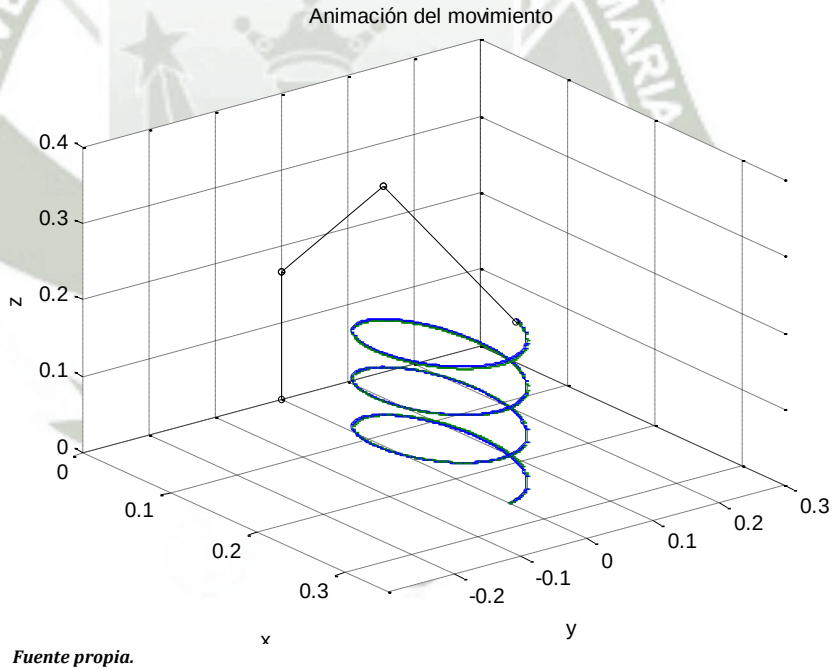
Fuente propia.

Fig. 4.22. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.23. Módulo del vector error en el espacio de trabajo.



Fuente propia.

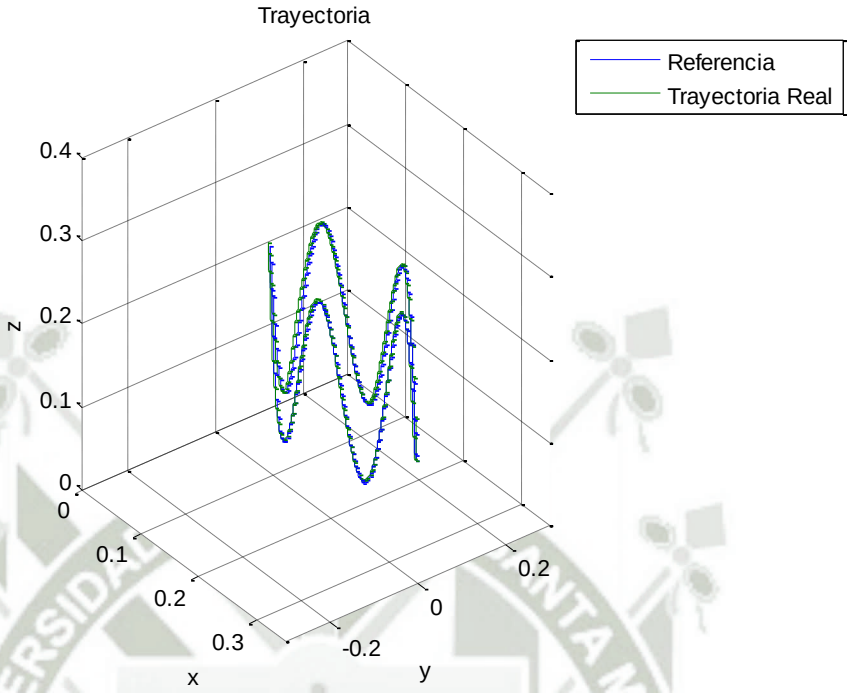
Fig. 4.24. Animación del movimiento.

- **Trayectoria helicoidal senoidal en el eje z**

  Los valores articulares máximos y mínimos son:

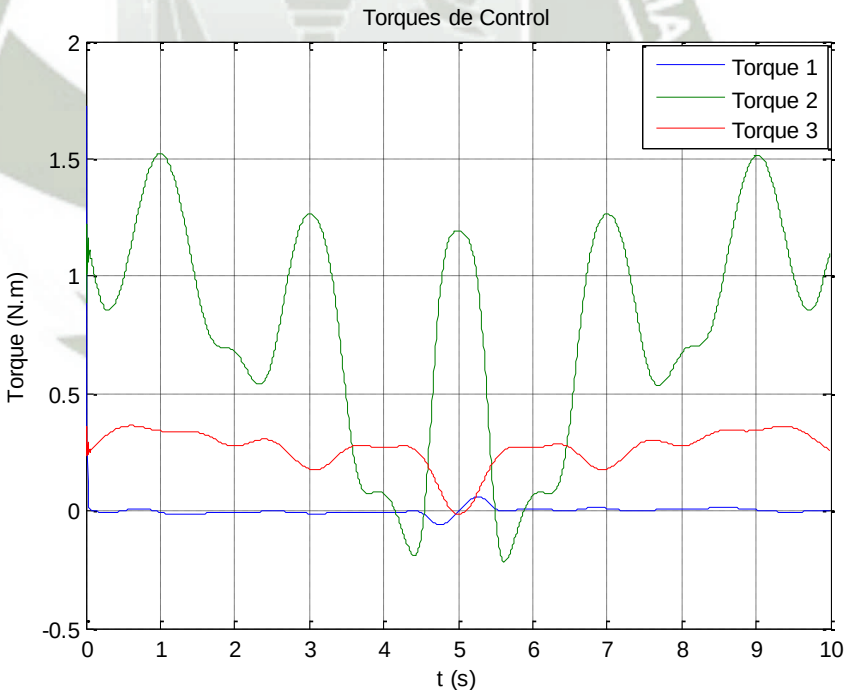
	1er GDL	2do GDL	3er GDL
Valor máximo	27.0293	100.3620	-24.9892
Valor mínimo	-27.0800	18.9051	-141.0374

La integral del módulo del vector error es: 0.0249.



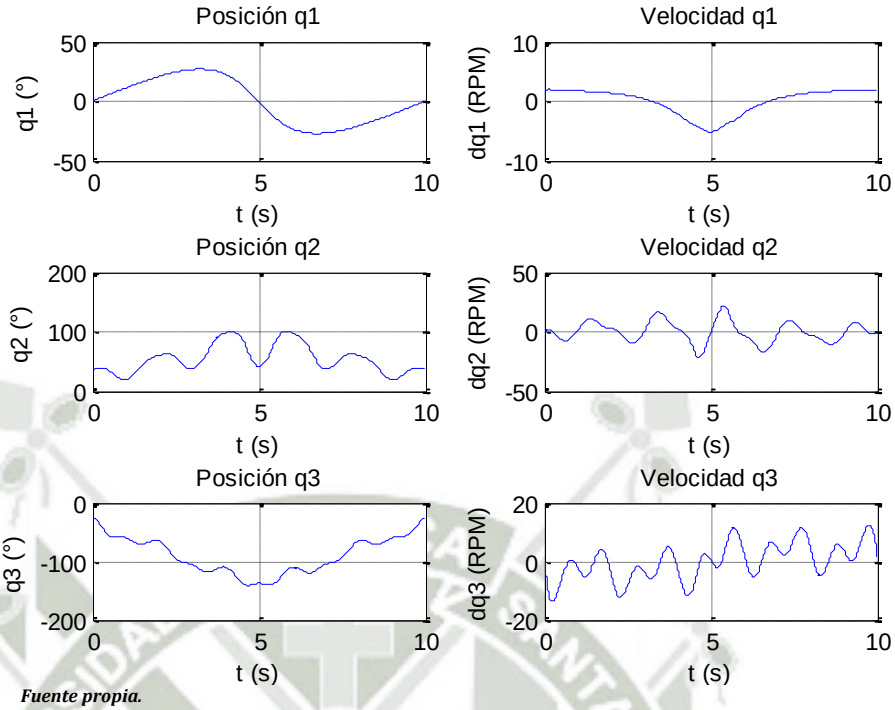
Fuente propia.

Fig. 4.25. Trayectoria helicoidal senoidal en el eje z.



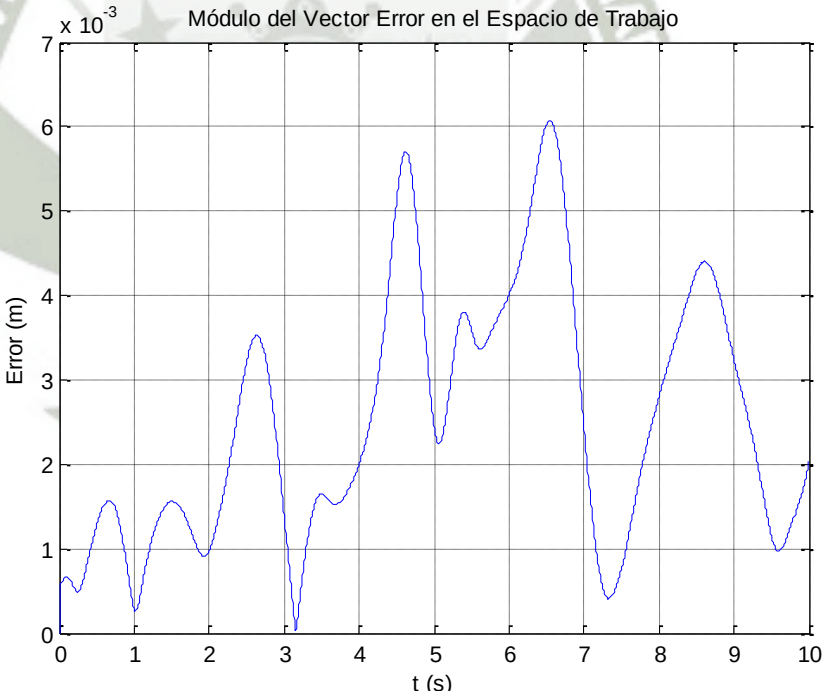
Fuente propia.

Fig. 4.26. Torques de control calculados para cada articulación.



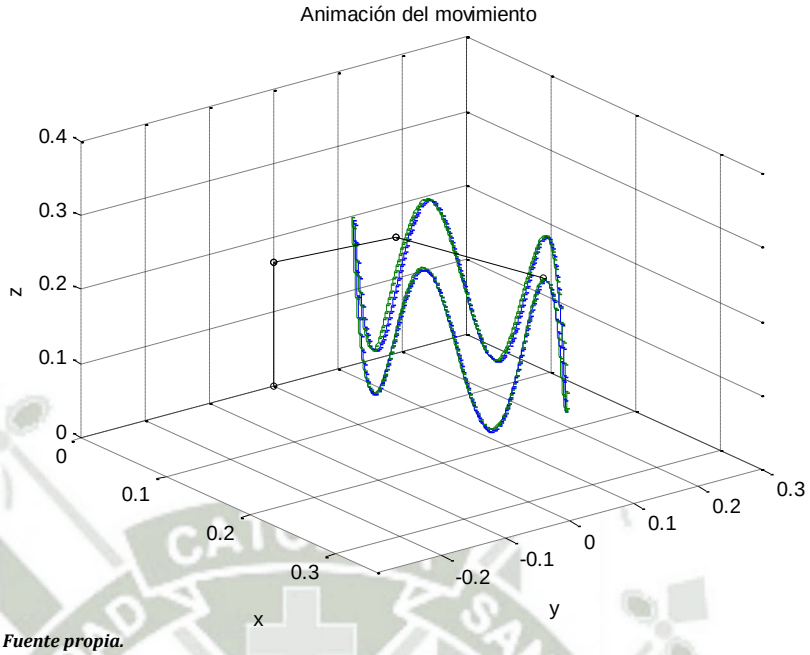
Fuente propia.

Fig. 4.27. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.28. Módulo del vector error en el espacio de trabajo.



Fuente propia.

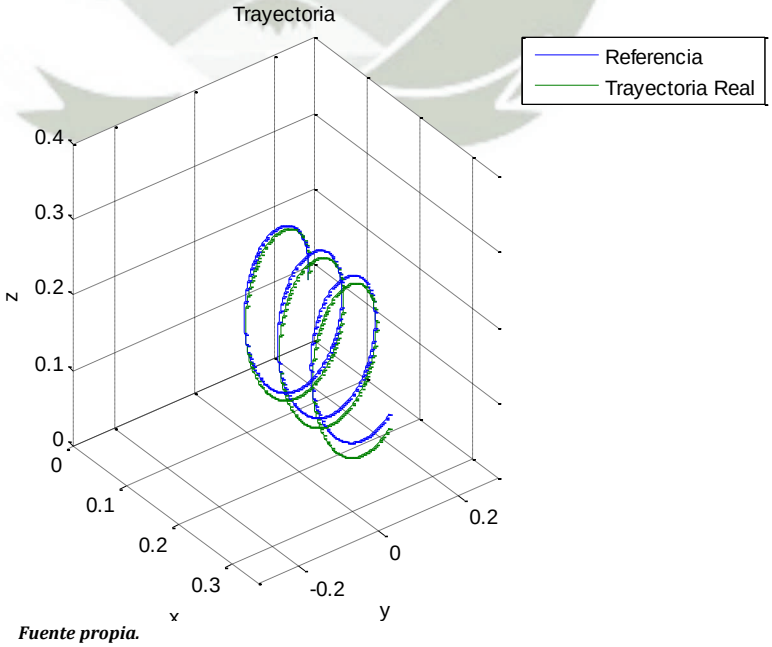
Fig. 4.29. Animación del movimiento.

- **Trayectoria helicoidal "horquilla" ascendente en el eje x**

  Los valores articulares máximos y mínimos son:

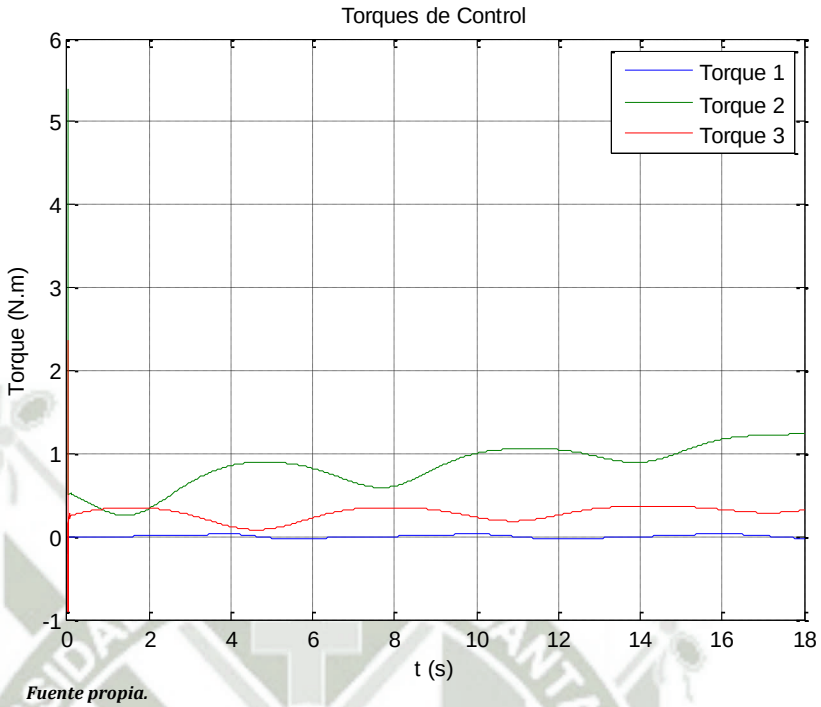
	1er GDL	2do GDL	3er GDL
Valor máximo	35.5377	93.7373	-44.9378
Valor mínimo	-30.3087	9.2638	-121.5921

  La integral del módulo del vector error es: 0.1894.



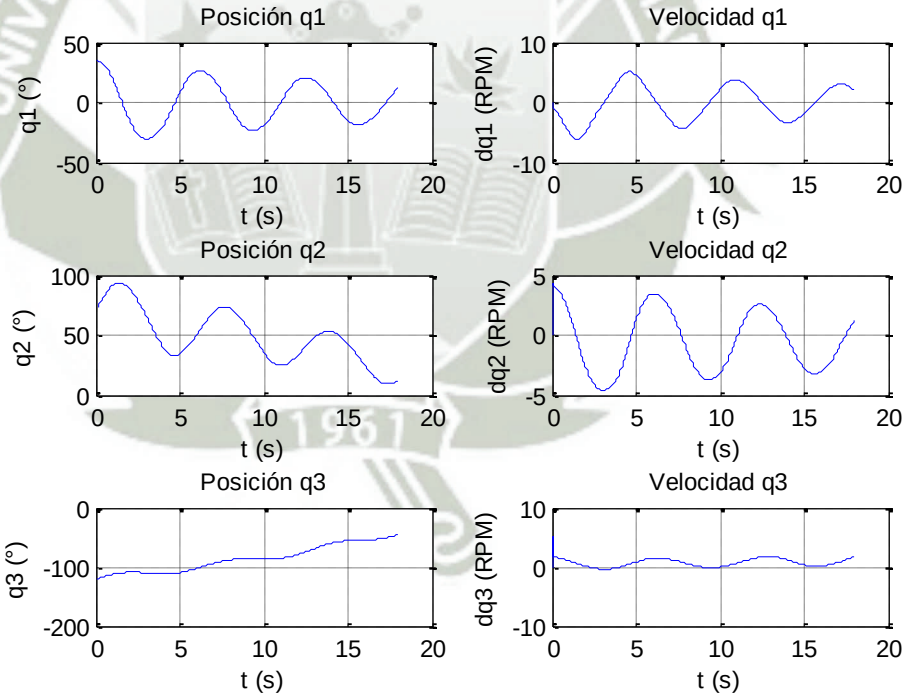
Fuente propia.

Fig. 4.30. Trayectoria helicoidal "horquilla" ascendente en el eje x.



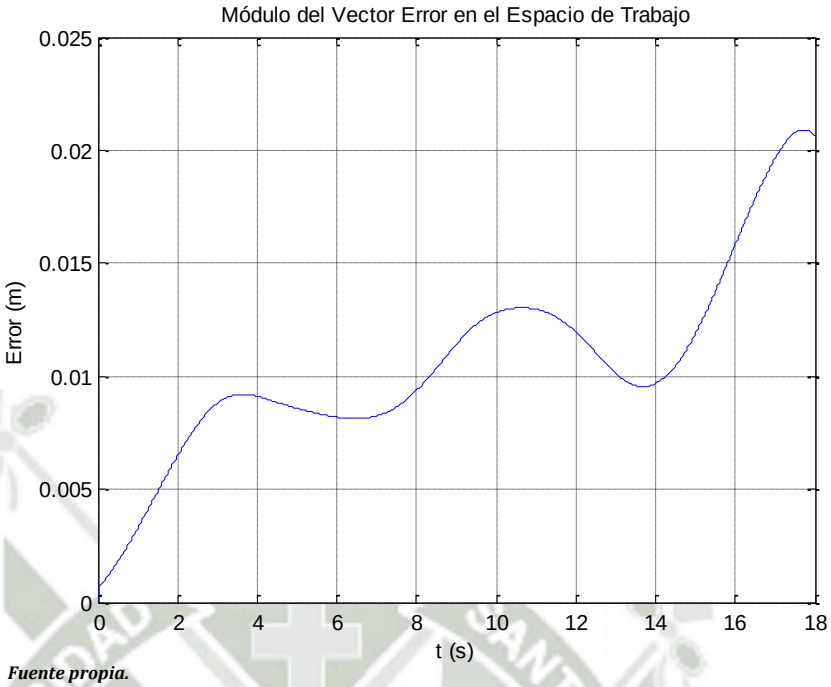
Fuente propia.

Fig. 4.31. Torques de control calculados para cada articulación.



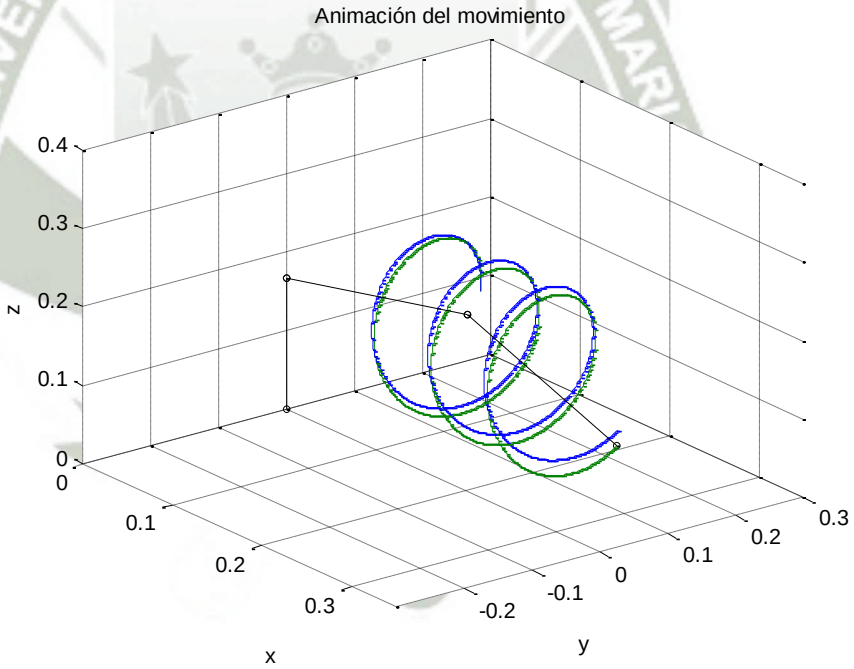
Fuente propia.

Fig. 4.32. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.33. Módulo del vector error en el espacio de trabajo.



Fuente propia.

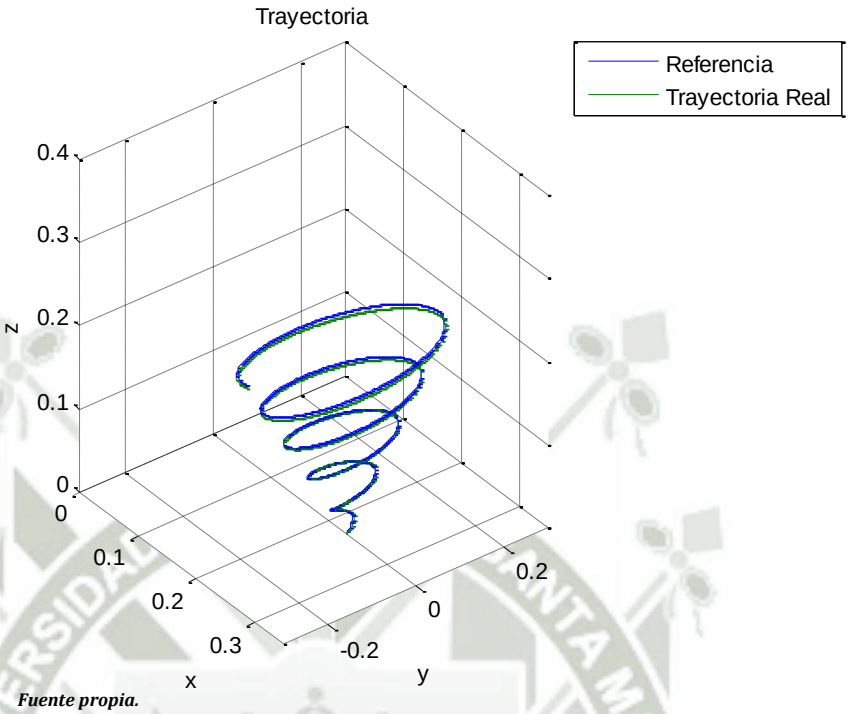
Fig. 4.34. Animación del movimiento.

- **Trayectoria helicoidal cónica en el eje z**

  Los valores articulares máximos y mínimos son:

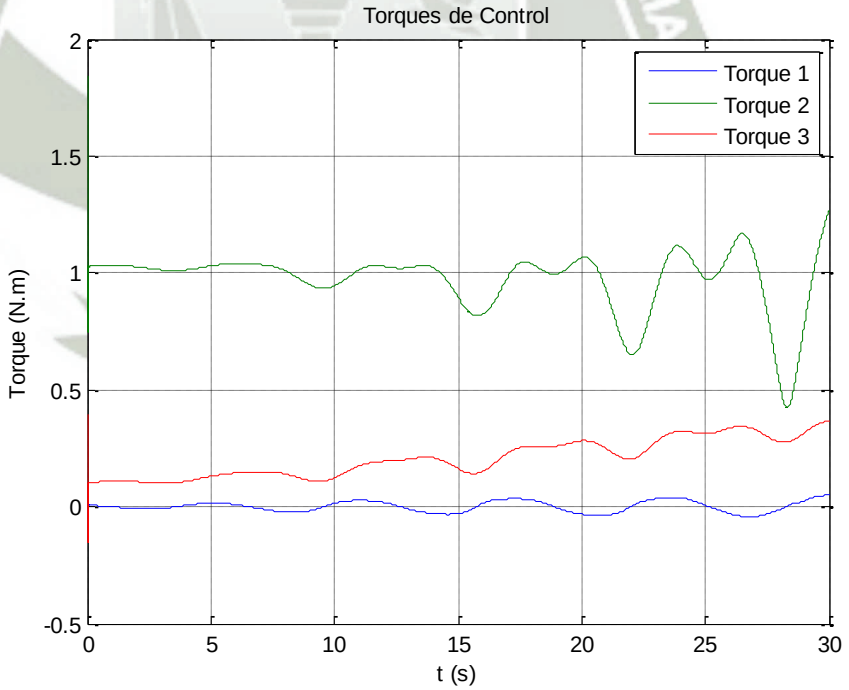
	1er GDL	2do GDL	3er GDL
Valor máximo	43.9708	76.1942	-32.5432
Valor mínimo	-46.8070	1.3009	-119.0821

La integral del módulo del vector error es: 0.0796.



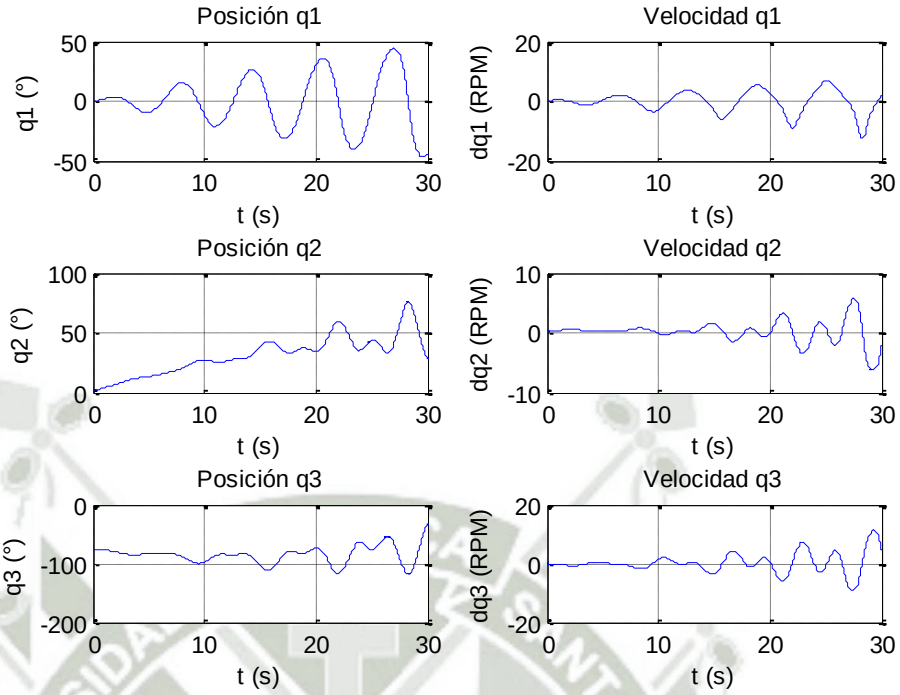
Fuente propia.

Fig. 4.35. Trayectoria helicoidal cónica en el eje Z.



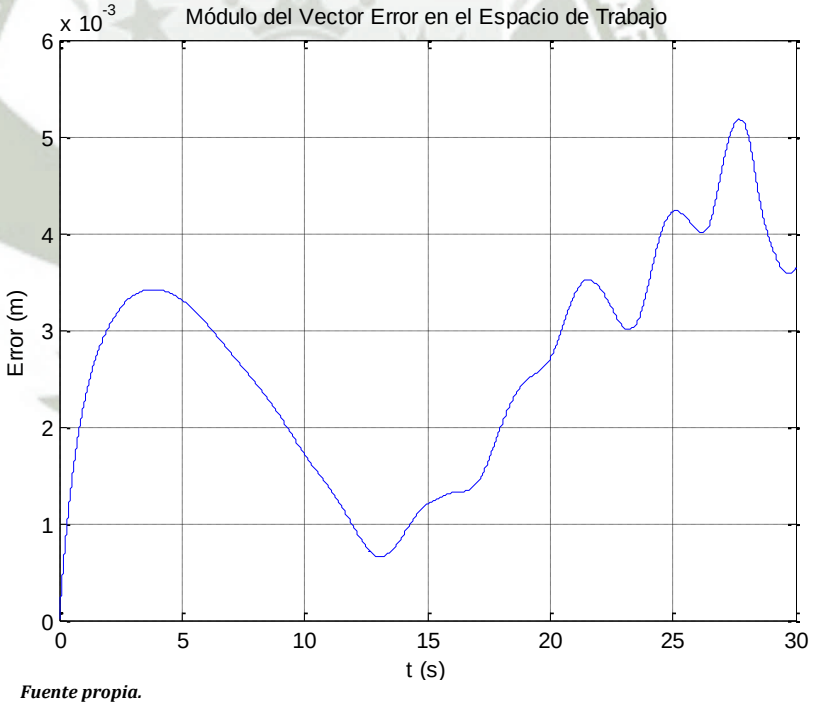
Fuente propia.

Fig. 4.36. Torques de control calculados para cada articulación.



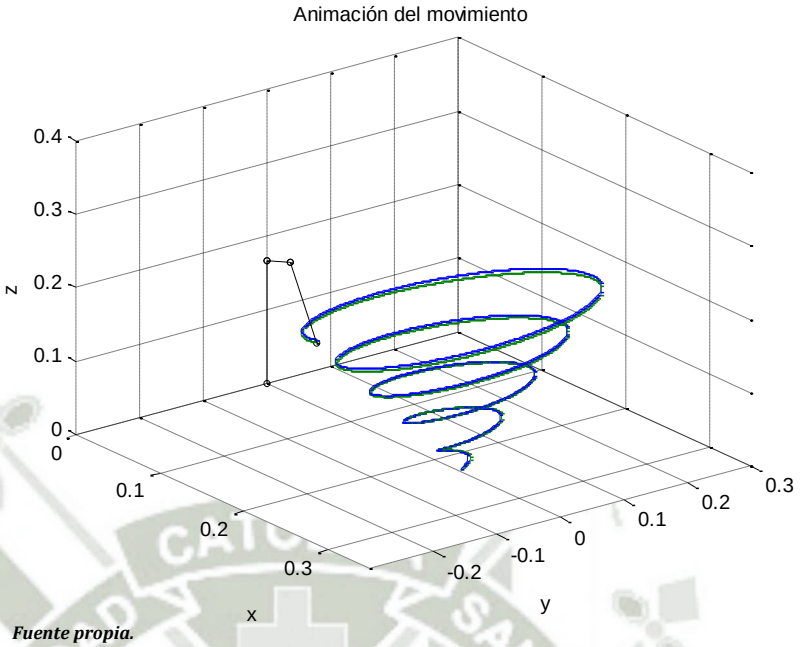
Fuente propia.

Fig. 4.37. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.38. Módulo del vector error en el espacio de trabajo.



Fuente propia.

Fig. 4.39. Animación del movimiento.

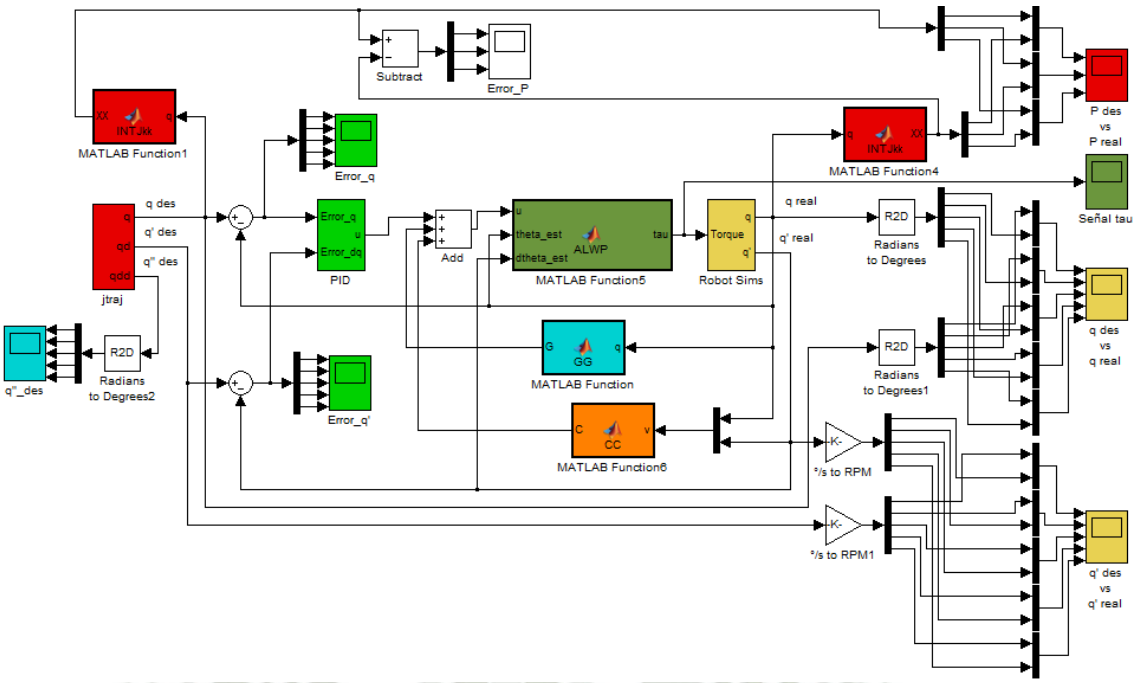
Como puede apreciarse en todos los casos, el módulo del error es muy pequeño, las trayectorias se encuentran dentro del espacio de trabajo del robot, y en todo instante del movimiento, la trayectoria real tiende a aproximarme muy de cerca a la trayectoria deseada, evidencia nítida de un control exitoso. Un dato interesante: no hubo necesidad de cambiar la sintonización del PID para cada trayectoria.

Control de los 5 GDL

Llevado a cabo el control de los 3 primeros GDL, se está en condiciones de agregar los 2 GDL faltantes, que permiten orientar el extremo del robot en dos direcciones referenciales.

Para elaborar el control de los 5 GDL, se sigue un procedimiento muy similar al realizado para los 3 GDL, con la única diferencia que el generador de trayectorias en lugar de ser del tipo cartesiano, ahora será del tipo articular. Esto significa que creará trayectorias en el espacio articular para cada articulación, dejando de lado las transformaciones de coordenadas cartesianas a articulares.

La Fig. 4.40. muestra el esquema de control PID+G para los 5 GDL. El bloque *jtraj* fue tomado del Robotics Toolbox de Corke, y permite generar trayectorias articulares interpoladas, indicados los puntos de inicio y fin. Este tipo de generador de trayectorias utiliza polinomios de orden 5 para la interpolación, considerando condiciones de contorno cero por defecto, tanto para la velocidad como para la aceleración, y sigue el criterio de que en un brazo robótico, la forma más rápida entre dos puntos a menudo no es una línea recta. De ahí que las trayectorias posean un perfil “parabólico”.



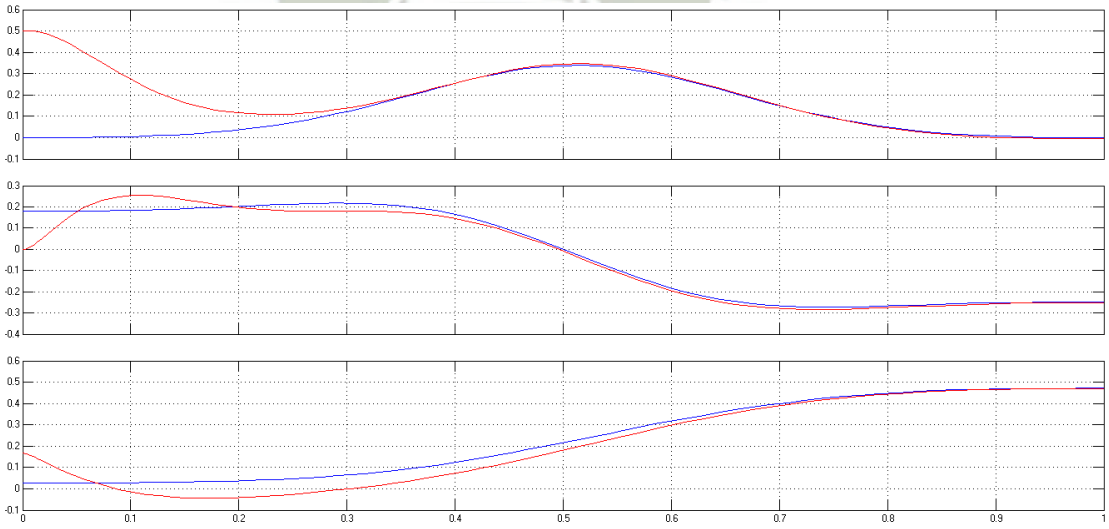
Fuente propia.

Fig. 4.40. Control PID+G para 5 GDL.

Para poder evaluar el esquema de control de la Fig. 4.40. que constituye el control final diseñado para 5 GDL, se genera una trayectoria articular interpolada, para la cual se seleccionan como puntos de inicio q_0 y final q_1 las siguientes coordenadas articulares:

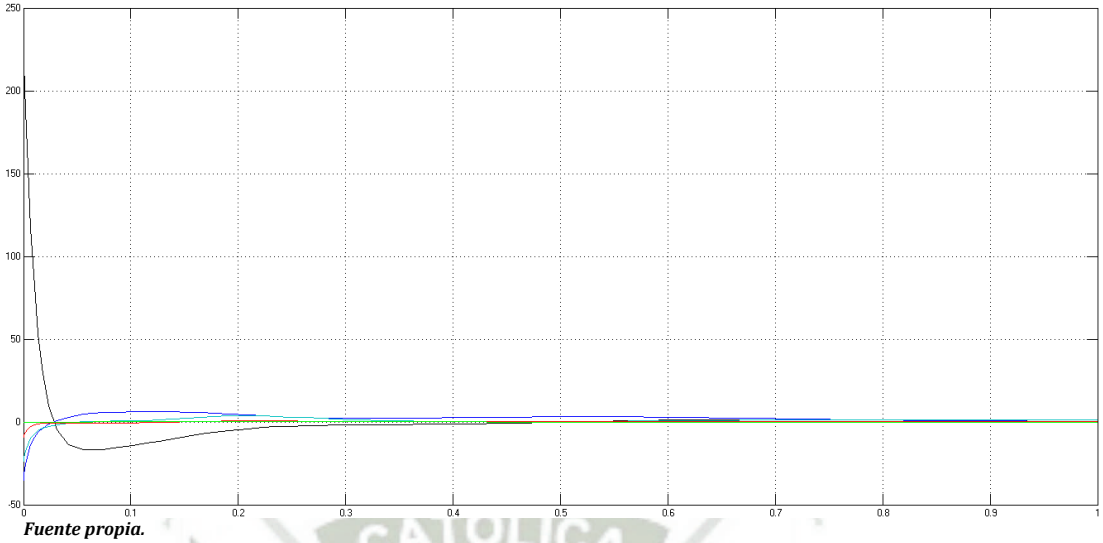
$$q_0 = \left[\frac{\pi}{2} \quad \frac{\pi}{3} \quad -\frac{2\pi}{3} \quad -\frac{\pi}{6} \quad -\frac{\pi}{2}\right] \qquad q_1 = \left[-\frac{\pi}{2} \quad \frac{\pi}{2} \quad -\frac{\pi}{2} \quad \frac{\pi}{3} \quad \frac{2\pi}{3}\right]$$

Al igual que en los 3 GDL, se muestran a continuación, todas las gráficas obtenidas y de manera comparativa, que permiten evaluar la estrategia de control diseñada.



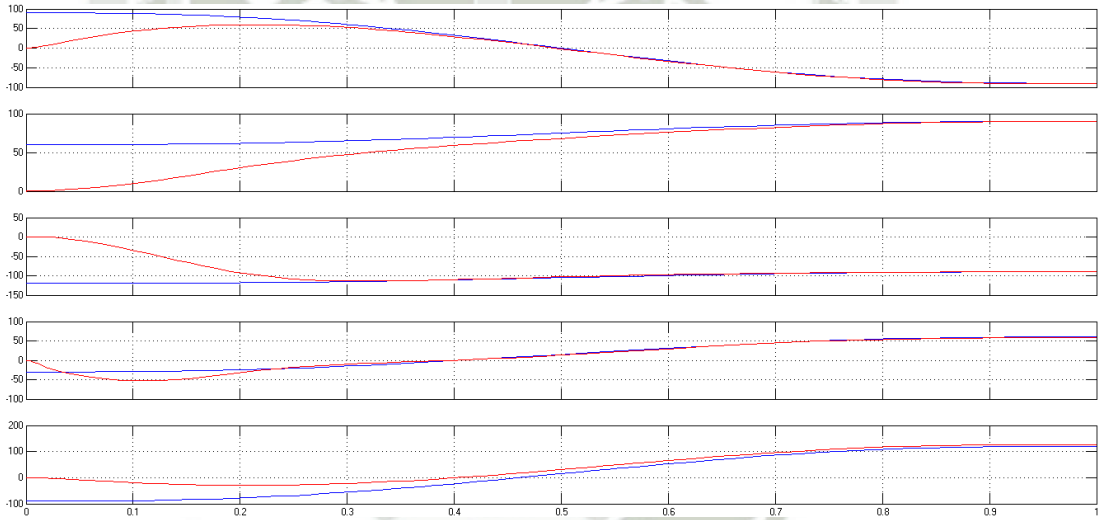
Fuente propia.

Fig. 4.41. Posición del extremo deseada (azul) vs. Posición del extremo real (roja).



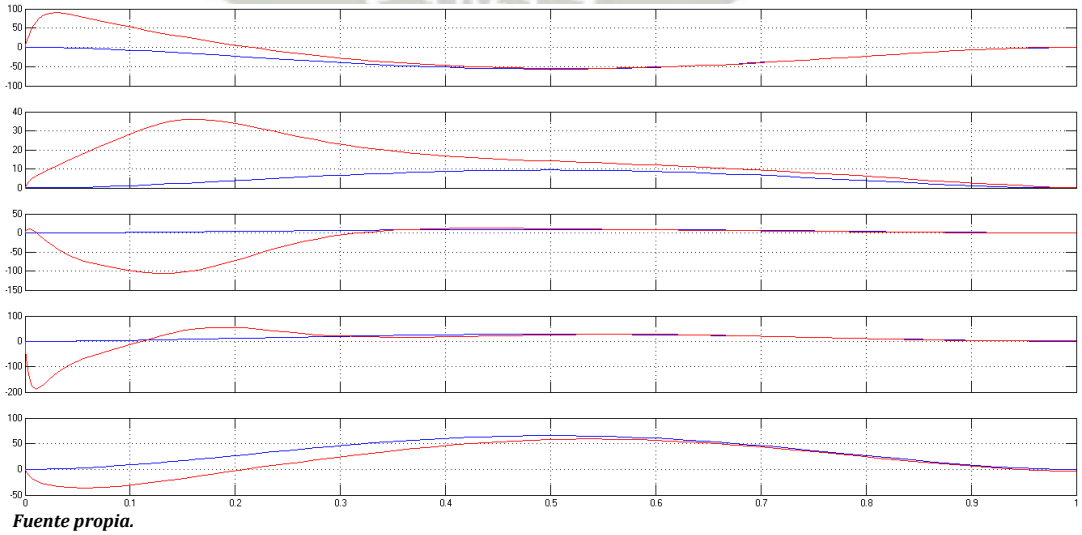
Fuente propia.

Fig. 4.42. Señales de torques aplicados en cada GDL.



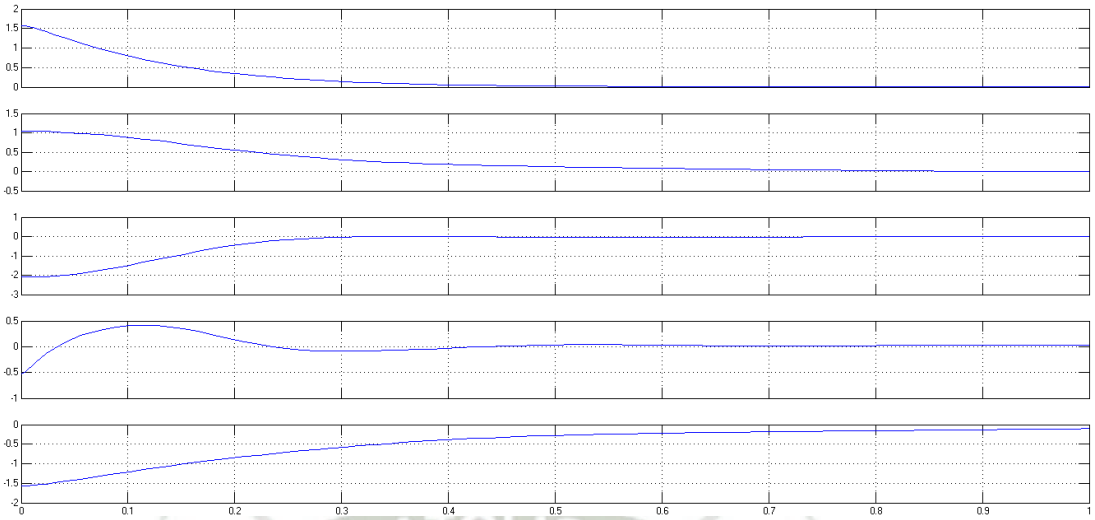
Fuente propia.

Fig. 4.43. Posición articular deseada (azul) vs. Posición articular real (roja).



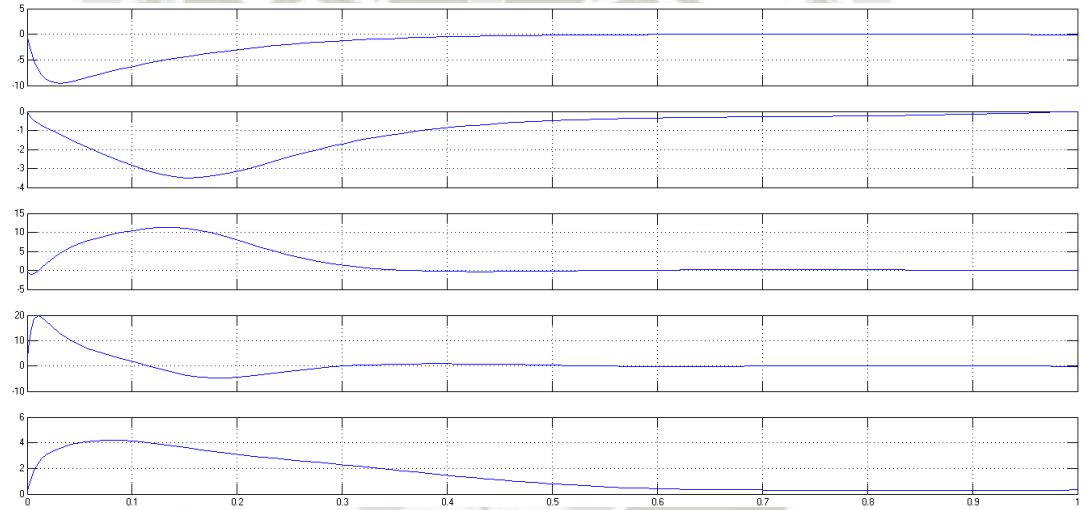
Fuente propia.

Fig. 4.44. Velocidad articular deseada (azul) vs. Velocidad articular real (roja).



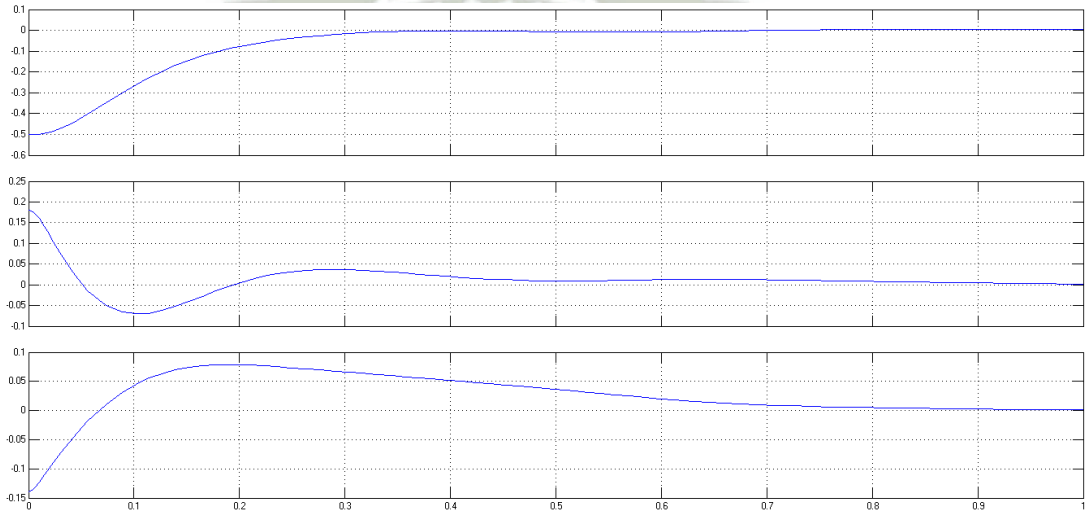
Fuente propia.

Fig. 4.45. Error de la posición articular.



Fuente propia.

Fig. 4.46. Error de la velocidad articular.



Fuente propia.

Fig. 4.47. Error de la posición del extremo del robot.

Así mismo, se elaboró el script *Darko_PID_G*, que cobija todo el control desarrollado anteriormente en Simulink®, traducido a código programado, incluyendo también una animación virtual 3D del robot completo, es decir, con los 5 GDL.

Se trazaron 3 tipos de trayectorias, para poner a prueba la efectividad del control de los 5 GDL. En todas ellas, se detallan los valores articulares máximos y mínimos alcanzados por cada articulación, la integral del módulo del vector de error, y se muestran 5 ventanas que grafican: la trayectoria cartesiana deseada o de referencia con la real trazada por el robot, los torques de control calculados para cada articulación, las trayectorias de las posiciones y velocidades articulares de cada GDL, el módulo del vector error en el espacio de trabajo y la animación del movimiento en 3D.

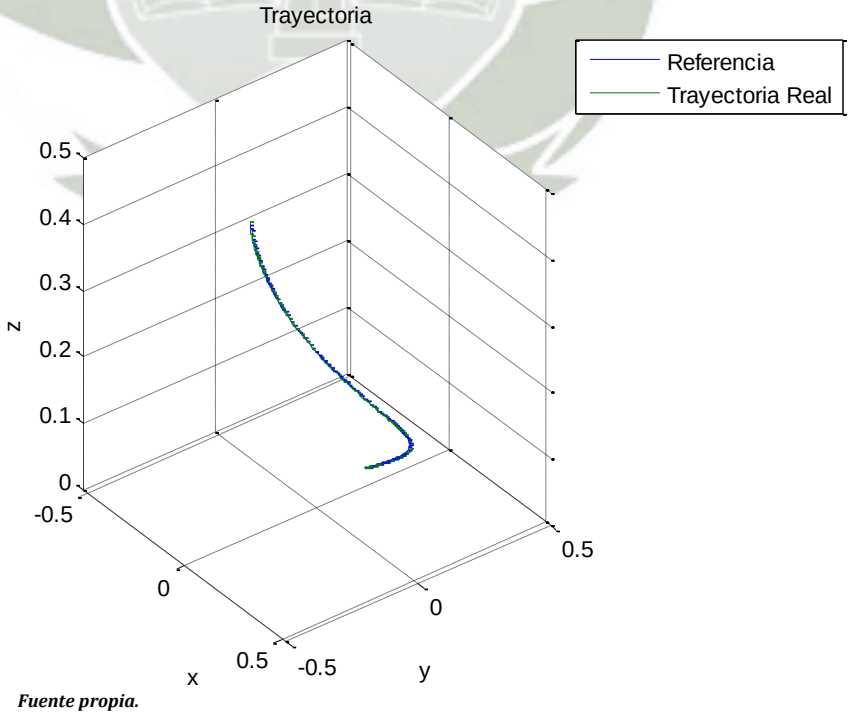
Los resultados para cada trayectoria, se muestran a continuación:

- **Trayectoria 1**

  Los valores articulares máximos y mínimos son:

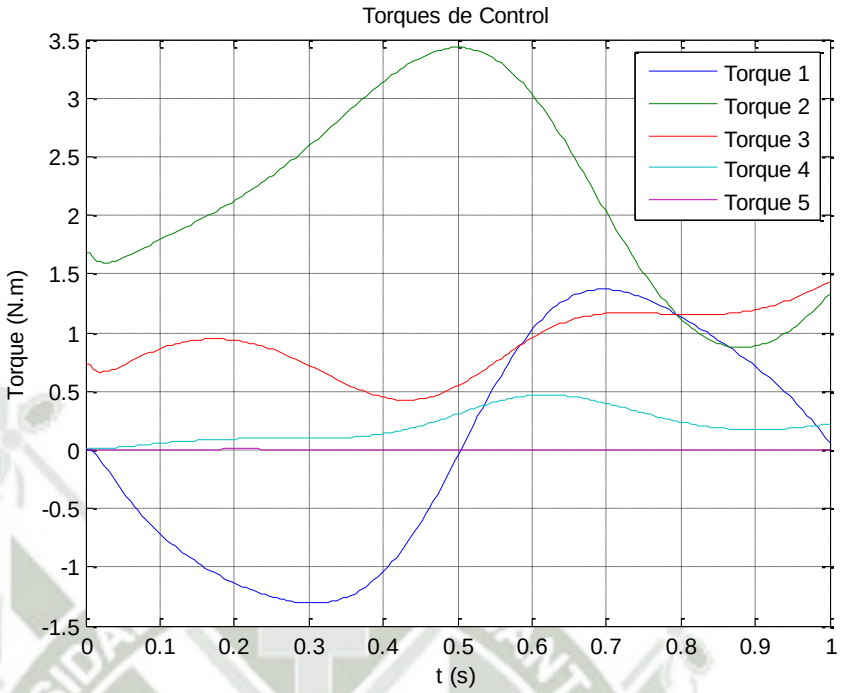
	1er GDL	2do GDL	3er GDL	4to GDL	5to GDL
Valor máximo	90.0000	90.6554	-89.7811	58.6812	122.8907
Valor mínimo	-91.0235	60.0000	-120.0000	-30.0209	-90.0000

  La integral del módulo del vector error es: 0.0048.



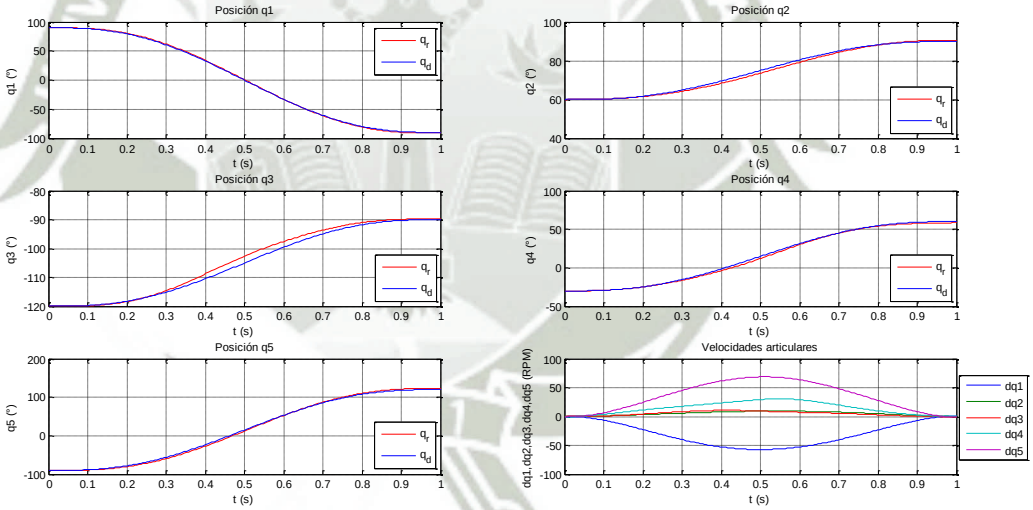
Fuente propia.

Fig. 4.48. Trayectoria 1.



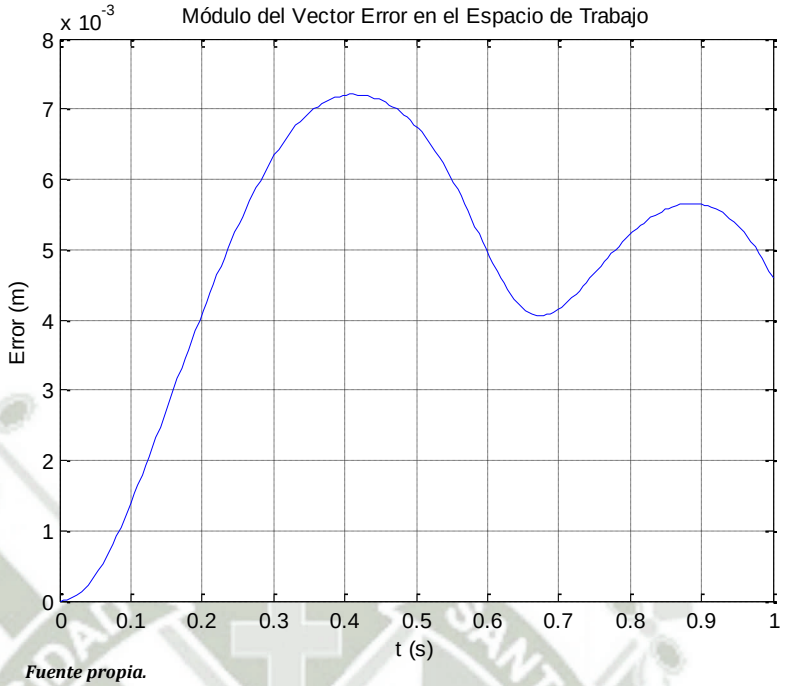
Fuente propia.

Fig. 4.49. Torques de control calculados para cada articulación.



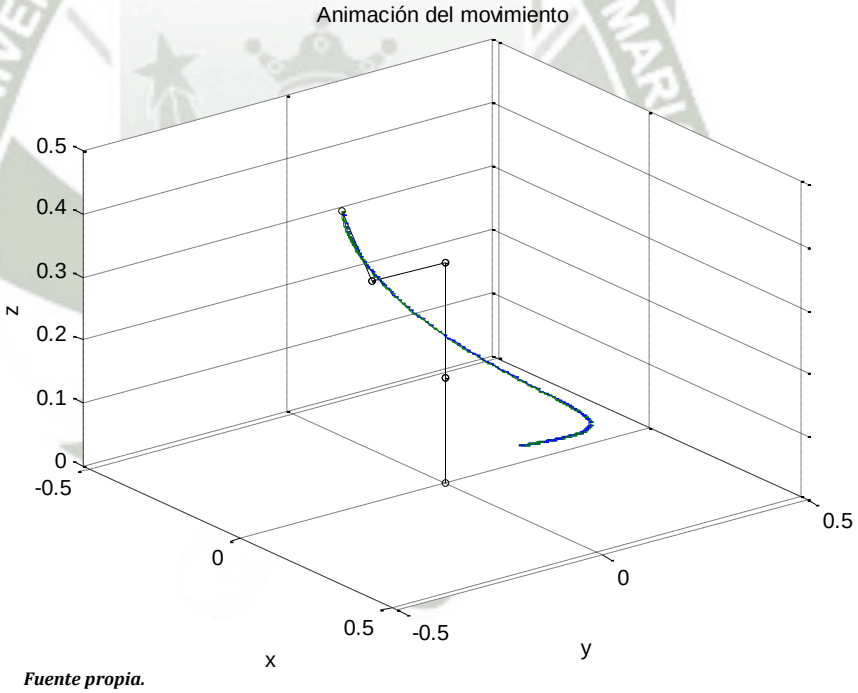
Fuente propia.

Fig. 4.50. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.51. Módulo del vector error en el espacio de trabajo.



Fuente propia.

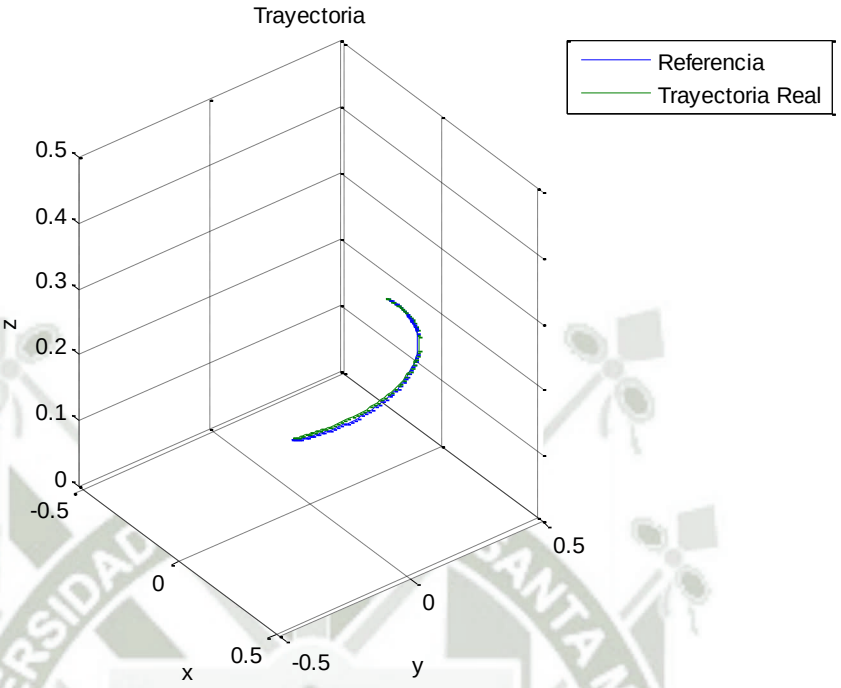
Fig. 4.52. Animación del movimiento.

- **Trayectoria 2**

Los valores articulares máximos y mínimos son:

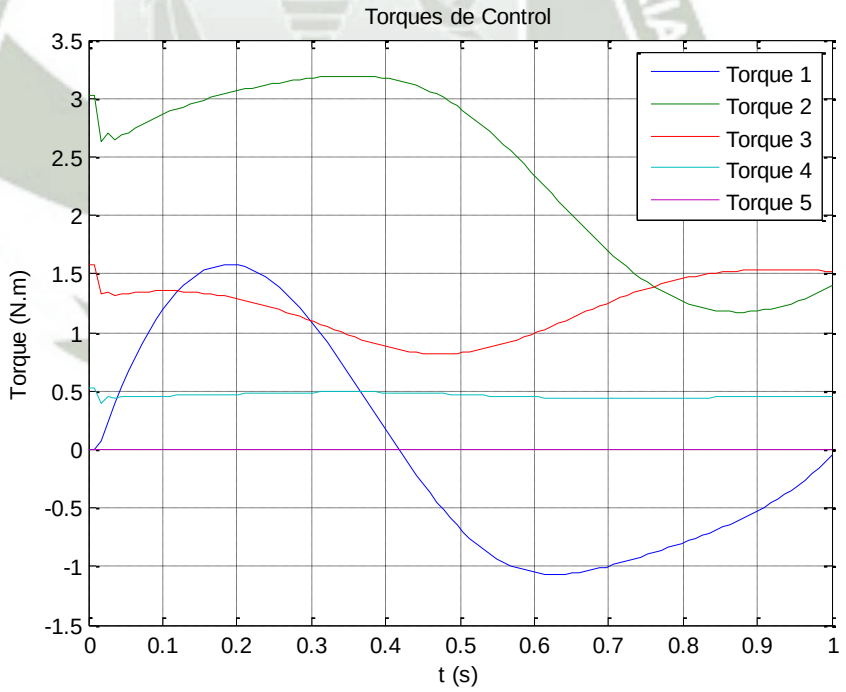
	1er GDL	2do GDL	3er GDL	4to GDL	5to GDL
Valor máximo	90.8918	91.0385	-89.9246	60.2048	62.4001
Valor mínimo	-45.0000	44.9985	-120.0595	28.2869	-90.0000

La integral del módulo del vector error es: 0.0056.



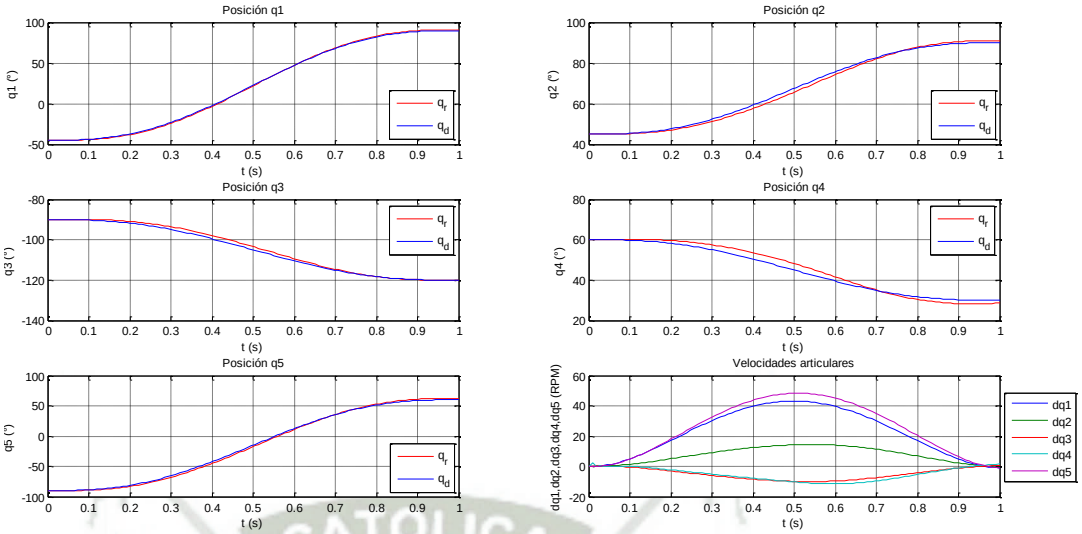
Fuente propia.

Fig. 4.53. Trayectoria 2.



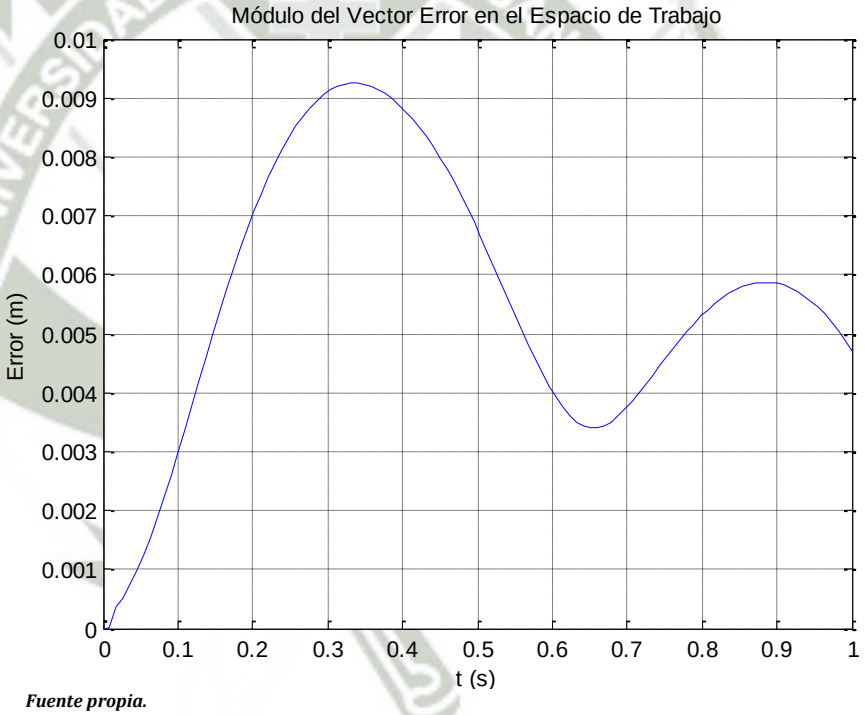
Fuente propia.

Fig. 4.54. Torques de control calculados para cada articulación.



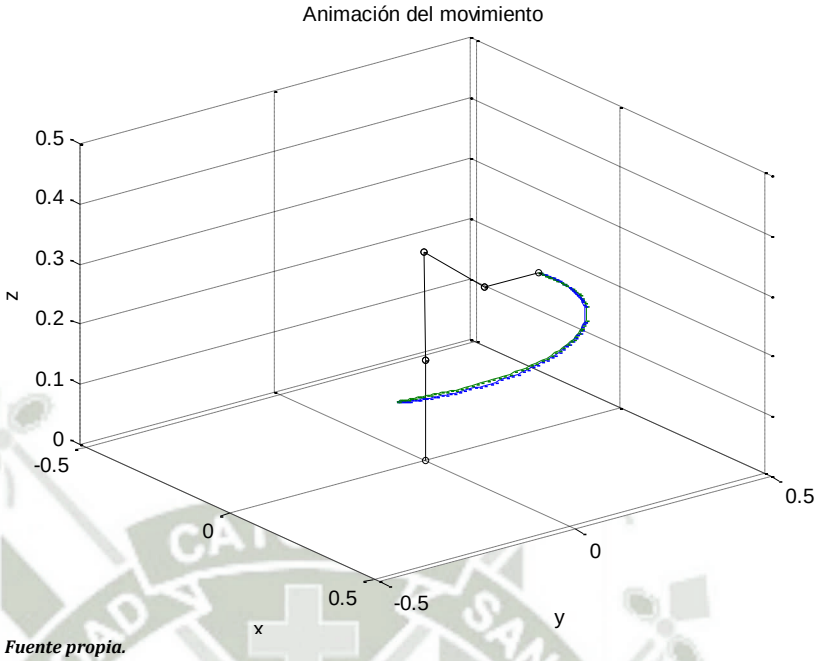
Fuente propia.

Fig. 4.55. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.56. Módulo del vector error en el espacio de trabajo.



Fuente propia.

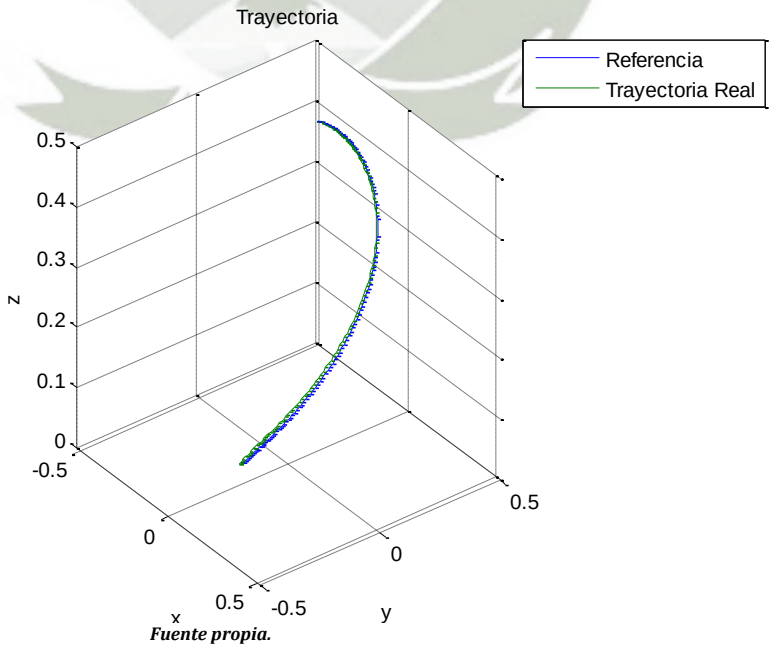
Fig. 4.57. Animación del movimiento.

- **Trayectoria 3**

  Los valores articulares máximos y mínimos son:

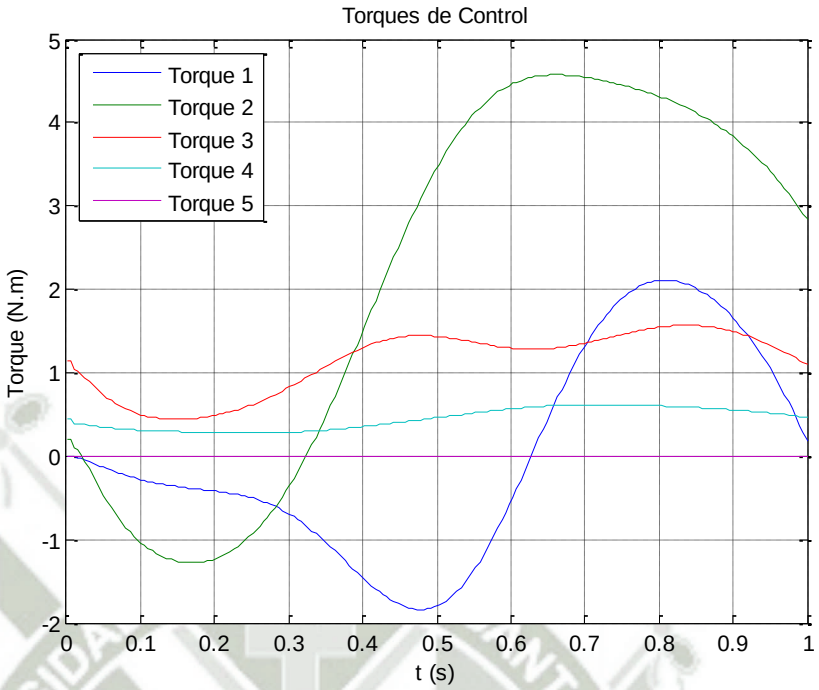
	1er GDL	2do GDL	3er GDL	4to GDL	5to GDL
Valor máximo	90.0000	120.0000	-59.9572	64.7836	120.0000
Valor mínimo	-60.9702	27.3934	-89.0951	-30.0000	-32.2583

  La integral del módulo del vector error es: 0.0074.



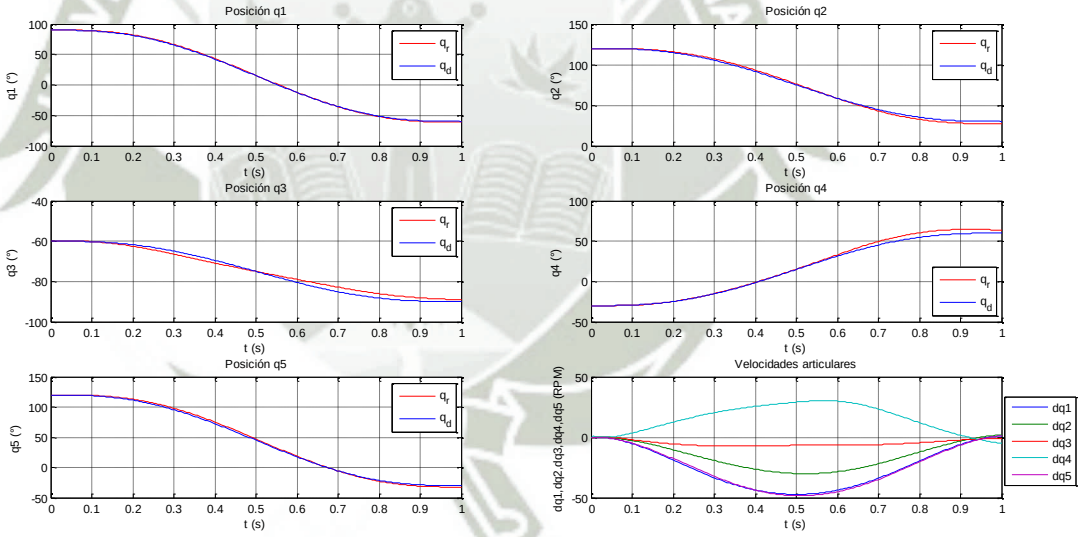
Fuente propia.

Fig. 4.58. Trayectoria 3.



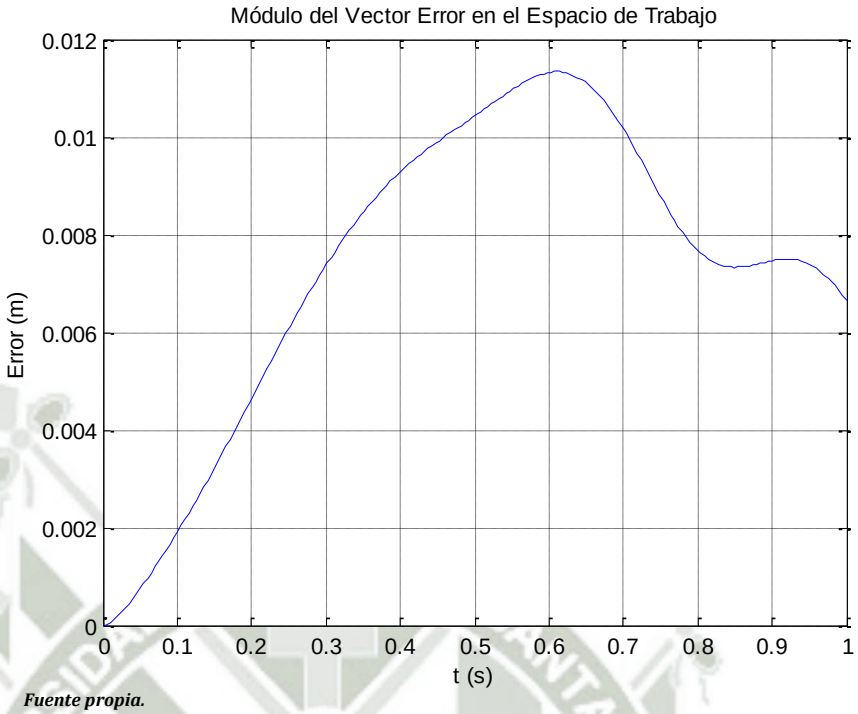
Fuente propia.

Fig. 4.59. Torques de control calculados para cada articulación.



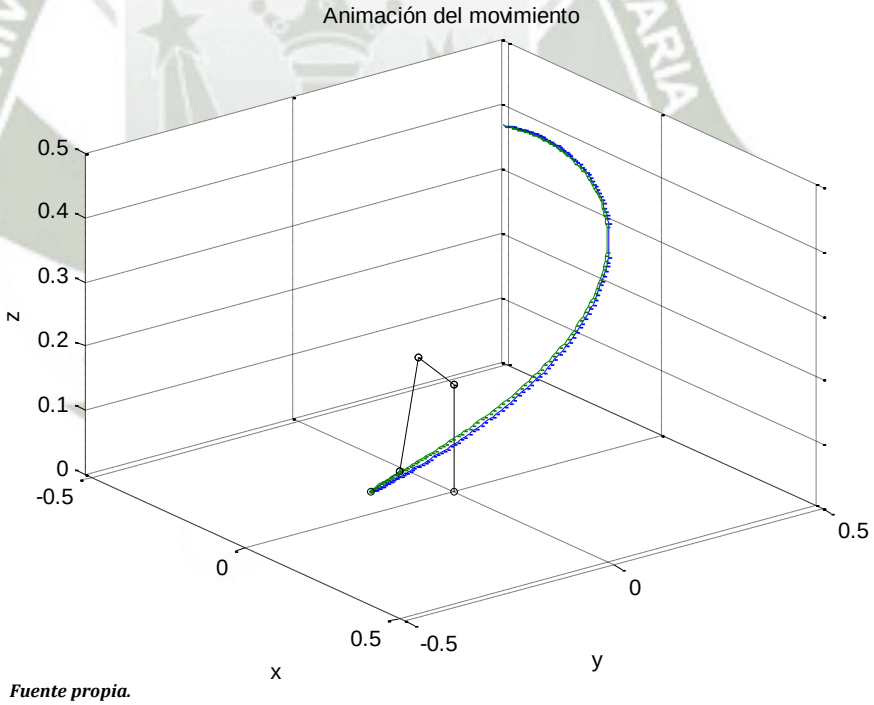
Fuente propia.

Fig. 4.60. Trayectorias articulares de posición y velocidad de cada articulación.



Fuente propia.

Fig. 4.61. Módulo del vector error en el espacio de trabajo.



Fuente propia.

Fig. 4.62. Animación del movimiento.

Como puede apreciarse en todos los casos, el módulo del error es muy pequeño, las trayectorias se encuentran dentro del espacio de trabajo del robot, y en todo instante del movimiento, la trayectoria real tiende a aproximarme muy de cerca a la trayectoria deseada, evidencia nítida de un control exitoso. Y nuevamente: no hubo necesidad de cambiar la sintonización del PID para cada trayectoria.

4.2. Simulación Controlada de Realidad Virtual

La simulación controlada de realidad virtual es un estudio por computador, dinámico y tridimensional, dirigido a la visualización y análisis de un sistema complejo donde se somete dicho sistema a un control. Existen diversas aplicaciones, pero en esta investigación se utiliza una de las más extendidas: el SimMechanics™ de Matlab®.

Simulación Controlada con SimMechanics™ de Matlab®

SimMechanics™ es una herramienta para modelar sistemas mecánicos a través del Simulink™ de Matlab®, y su utilidad es el diseño, control y simulación de cuerpos rígidos conectados mediante articulaciones y sometidos a fuerzas y pares. Los cuerpos pueden ser descritos a través de sus propiedades de masa e inercia y conectados mediante juntas actuadas por fuerza o movimiento (a través de librerías de diagramas de bloques o código generado); además, se pueden utilizar sensores para medir diversos parámetros que se producen durante la simulación y emplearlos para el control, por ejemplo. Esta aplicación se basa en mecánica newtoniana, para obtener el movimiento que resulta de aplicar una fuerza en un sistema mecánico y viceversa.

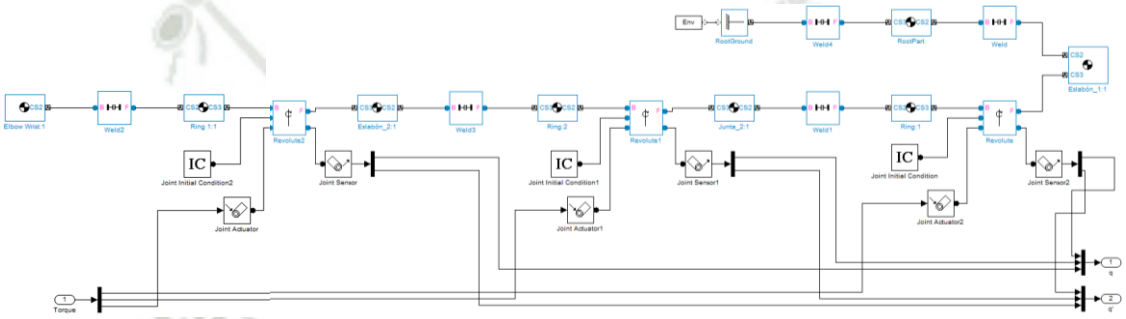
Un brazo robótico, como el diseñado en esta investigación, es un mecanismo de cuerpos rígidos, y como todo cuerpo rígido, puede ser modelado en SimMechanics™ de dos maneras: por diseño individual bloque a bloque (a través de la interface gráfica de Simulink™) y por traducción del sistema CAD utilizando SimMechanics Link™. Con la finalidad de obtener un modelamiento más detallado y cercano a la realidad, se utiliza el método por traducción del sistema CAD.

SimMechanics Link™, mediante el concepto *CAD Translation*, permite convertir ensamblajes o piezas de un sistema CAD, en ficheros de modelado físico *.xml que son portables e independientes de SimMechanics™. Empleando este *add-in*, será posible evitar todos los pasos de creación de cuerpos y articulaciones (que el método por diseño individual bloque a bloque exige), puesto que serán generados automáticamente en función de la configuración realizada del sistema robótico en el software de CAD. Para llevar a cabo una exportación de formato CAD a un formato reconocible por SimMechanics™, se deben seguir los siguientes pasos:

- Instalar el SimMechanics Link™ según la plataforma CAD que se tenga (Autodesk® Inventor® Professional, en este caso).
- Exportar el ensamblaje o pieza desde la plataforma CAD en base a las opciones de conversión deseadas (tendrá una estructura jerárquica de raíces, conjuntos y subconjuntos mecánicos).

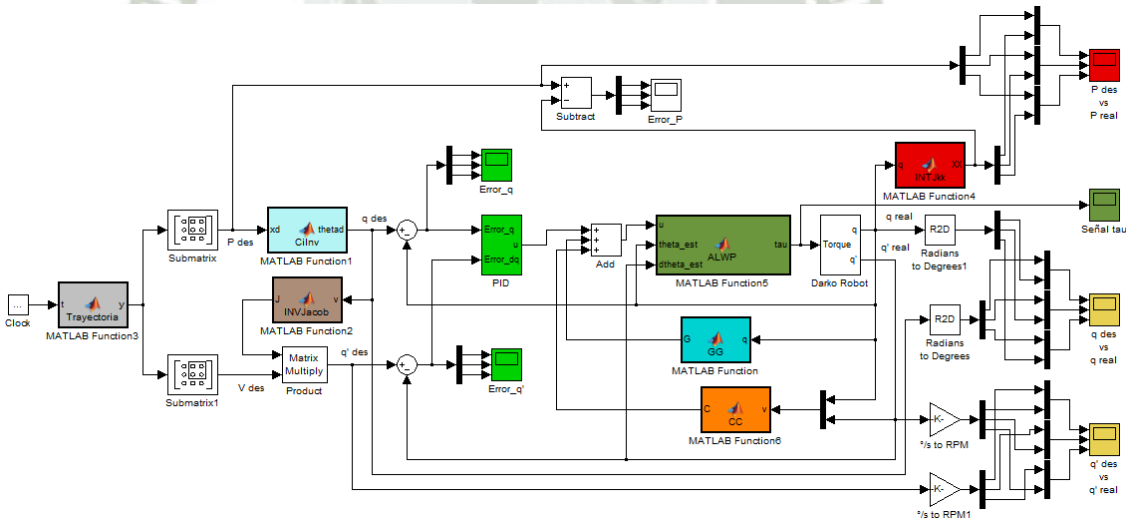
- Importar el fichero **.xml* generado en el paso anterior de modelo físico a SimMechanics™, empleando para ello el comando *mech_import('Name.xml')* desde Matlab®, donde *Name* es el nombre del archivo **.xml*.
- Una vez obtenido el archivo **.mdl* que contiene el modelo del robot en SimMechanicsTM, se añaden actuadores, sensores y otros bloques de interés a cada articulación.[16]

Finalmente, se enmascara todo este modelamiento bajo el bloque llamado *Darko Robot* (Fig. 4.63.), y se inserta en el diagrama de control PID+G, el cual se muestra en la Fig. 4.64.



Fuente propia.

Fig. 4.63. Bloque *Darko Robot*.



Fuente propia.

Fig. 4.64. Control PID+G de la planta obtenida con SimMechanics™.

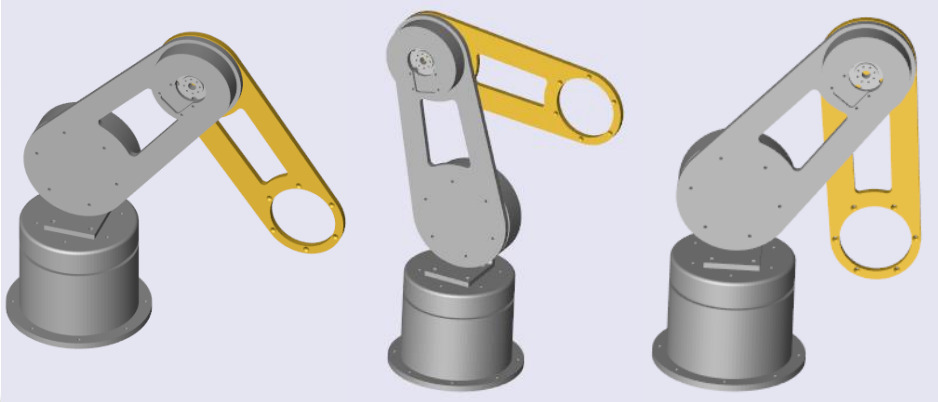
A diferencia del control de la Fig. 4.7., donde la planta fue obtenida a través de la Formulación L-E, esta vez la planta es modelada en SimMechanics™. Sin duda, existen diferencias entre ambas, y si se desea profundizar en su estudio (alcance que se encuentra fuera de los objetivos de esta tesis), se invita al lector a realizar, a

[16] Para obtener el detalle de cómo realizar la traducción del sistema CAD utilizando el SimMechanics Link™, visitar el siguiente link: http://www.mathworks.com/help/physmod/smlink/ug/installing-and-linking-simmechanics-link-software.html

modo de investigación extensiva, todas las comparaciones que cree pertinentes entre ambas plantas, con la finalidad principal de encontrar el grado de similitud entre ellas.

Al ejecutar el programa *Robot_3GDL_1*, que contiene el esquema de la Fig. 4.64., se apertura una ventana donde se muestra en realidad virtual, el movimiento controlado del modelo CAD del robot. La Fig. 4.65. captura algunos instantes de dicho movimiento.

La simulación del comportamiento del robot, bajo varias condiciones de funcionamiento, hace posible predecir cómo se comportará el sistema robótico una vez éste sea construido y se ponga en funcionamiento.



Fuente propia.

Fig. 4.65. Simulación controlada en realidad virtual con SimMechanics™.

Capítulo V

Control Kinect. Diseño de la interfaz intuitiva.

5.1. Diseño de la interfaz intuitiva

Para llevar a cabo el Control Kinect del brazo robótico, es necesario diseñar todo un entorno de desarrollo a nivel de software, que permita crear el ambiente adecuado para que el usuario perciba que se encuentra dentro de toda una NUI y que dentro de ella, él es el "mando" que tiene el control del robot.

A continuación, se detallan los pasos a seguir para construir el Control Kinect que ésta investigación propone.

Paso 1: Instalación del Microsoft® Visual Studio Express 2012

A través de Visual Studio Express 2012[17], el Entorno de Desarrollo Integrado (IDE) más conocido de Microsoft® para crear aplicaciones en sistemas operativos Windows, es posible lograr no solo la comunicación y programación del Kinect (teniendo acceso a todas sus prestaciones y diversos lenguajes de alto nivel tales como Visual Basic .NET, C++, C#, entre los más conocidos), sino también de los *Smart Servos Dynamixel* seleccionados anteriormente.

Fuente: "Logo Visual Studio Express 2012", Microsoft Corporation, 2012.

Fig. 5.1. Visual Studio Express 2012.

Paso 2: Instalación del Microsoft® Kinect SDK v1.5

El Kinect para Windows SDK proporciona las herramientas e Interfaces de Programación de Aplicaciones (APIs), tanto nativas y administradas, que se necesitan para desarrollar aplicaciones compatibles con Kinect para Windows. Además, el SDK también contiene todos los controladores USB para los distintos

[17] Puede descargar el Visual Studio Express 2012 del siguiente link: http://www.microsoft.com/en-us/download/details.aspx?id=34673. Es necesario tener conexión a Internet durante la instalación.

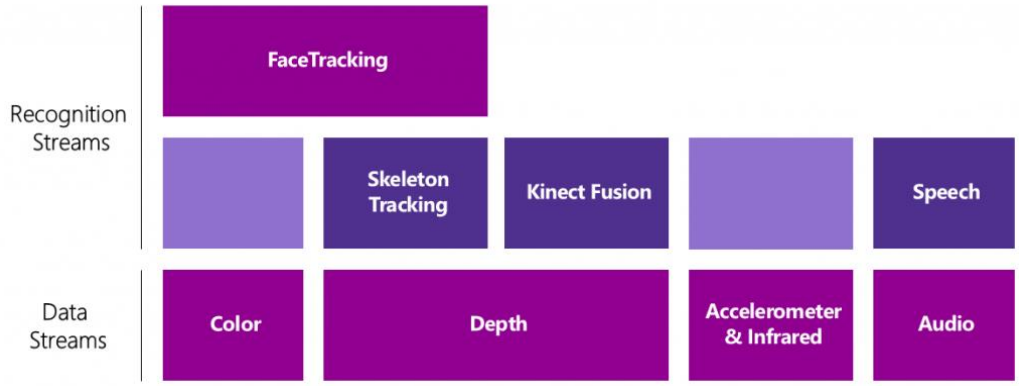
elementos internos del sensor. Su descarga es gratuita[18] y está compuesto, principalmente, por lo siguiente:

Bloques de construcción

Antes de programar el Kinect, es importante saber a qué tipos de datos es posible acceder a través de las funciones que brinda el SDK. Como puede apreciarse en la Fig. 5.2., el SDK se divide en dos corrientes de datos: los Flujos de Datos (*Data Streams*) y los Flujos de Reconocimiento (*Recognition Streams*).

Los Flujos de Datos son básicamente los flujos que el sensor puede capturar a través de sus sensores como el color con la cámara RGB, la profundidad con el emisor de luz IR y el sensor de profundidad IR, y los comandos de voz y audio a través de la matriz de micrófonos.

Los Flujos de Reconocimiento, son aquellos que dependen de los Flujos de datos, debido a que se encargan de procesar la información que estos últimos capturan, como por ejemplo, si desea utilizar el seguimiento de rostro (*Face Tracking*), es necesario habilitar los flujos de color, profundidad y seguimiento de esqueleto (*Skeleton Tracking*) con el fin de permitir un exitoso seguimiento de rostro.



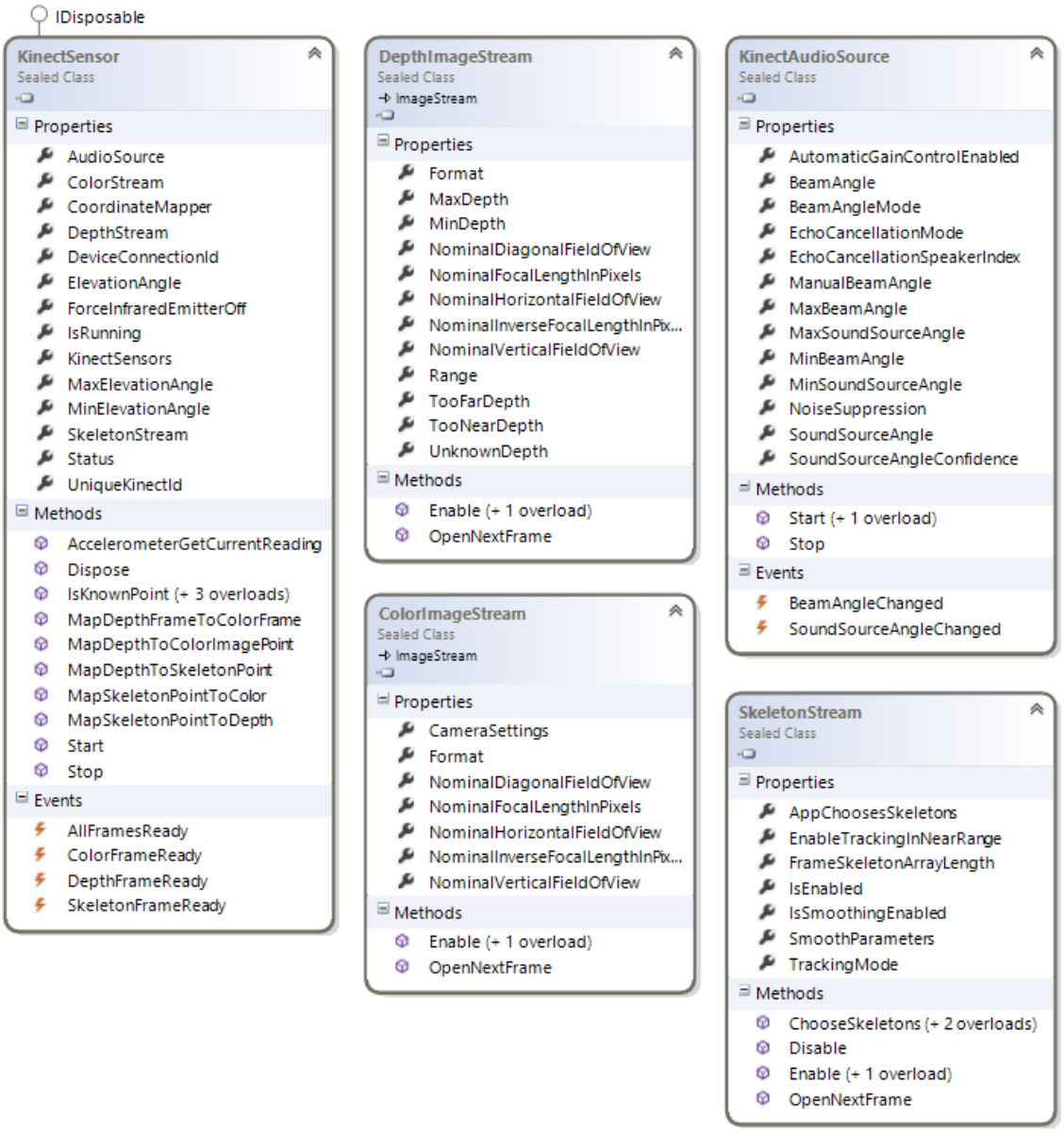
Fuente: "Introduction to Kinect for Windows SDK", Kinecting for Windows, 2013.

Fig. 5.2. Bloques de Construcción del Kinect SDK.

Clases y objetos

El corazón del SDK son las clases. Una clase es una construcción que permite crear tipos personalizados propios mediante la agrupación de variables de otros tipos, métodos y eventos. Una clase es como un plano. Define los datos y el comportamiento de un tipo. La Fig. 5.3. muestra las principales clases que se utilizan al programar con Kinect.

[18] Puede descargar el Kinect para Windows SDK del siguiente link: http://www.microsoft.com/en-us/kinectforwindows/. Si desea mayor información sobre su instalación y contenido, visite http://msdn.microsoft.com/en-us/library/hh855354.aspx.



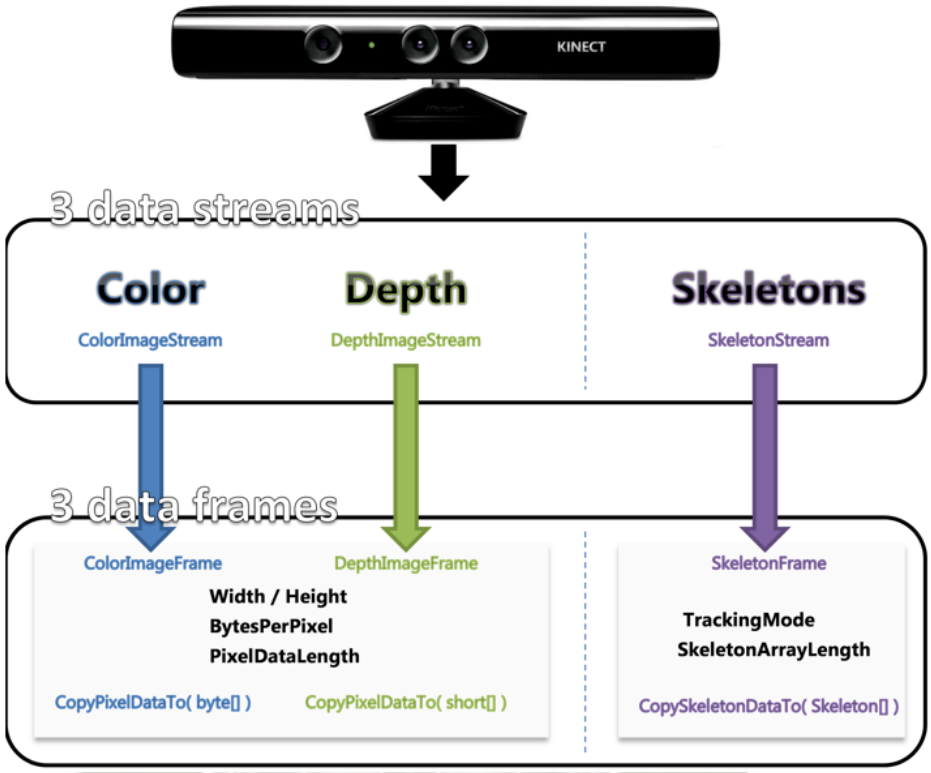
Fuente: "Introduction to Kinect for Windows SDK", Kinecting for Windows, 2013.

Fig. 5.3. Clases disponibles del Kinect SDK.

Un objeto es básicamente un bloque de memoria que se ha asignado y configurado en función del plano. Un programa puede crear muchos objetos de la misma clase. Los objetos también se denominan instancias y se pueden almacenar en una variable con nombre o en una matriz o colección.

Uno de los objetos principales que el SDK posee es el llamado **KinectSensor**. Este objeto representa a un sensor Kinect dentro del IDE. A través de él, es posible activar los flujos de color, profundidad y seguimiento del esqueleto, cambiar el ángulo de elevación del sensor, comprobar su estado de funcionamiento y si está en ejecución, entre otras funciones.

Como puede apreciarse en la Fig. 5.4., cada sensor Kinect tiene, como mínimo, tres flujos (Color, Profundidad y Esqueleto). Cada uno está representado por un objeto (*ColorImageStream*, *DepthImageStream* y *SkeletonStream*, respectivamente) que le darán marcos (*Frames*) de ejecución a los datos, cuando se invoque al método *OpenNextFrame*.



Fuente: "Introduction to Kinect for Windows SDK", Kinecting for Windows, 2013.

Fig. 5.4. Flujos y Marcos principales del Kinect.

Paso 3: Instalación del Dynamixel SDK

Los *Smart Servos Dynamixel* también poseen un SDK que permite interactuar con ellos a través de diferentes IDEs (Matlab®, Python™, Microsoft® Visual Studio, y LabVIEW™.). El Dynamixel SDK es una biblioteca de programación estándar, creada exclusivamente para controlar los actuadores Dynamixel[19]. Sus características son las siguientes:

- Tiene portabilidad excepcional en cada plataforma ya escrita en lenguaje C.
- Es fácil de realizar una portabilidad de plataforma ya que está escrito en ambas plataformas: fuentes independientes y dependientes.
- La interfaz es estándar; por lo tanto, el software desarrollado puede ser utilizado siempre, incluso si el controlador es diferente.

[19] Para descargar el Dynamixel SDK v1.02, visite la dirección web http://support.robotis.com/en/. Ubique el Menú Principal y siga la siguiente ruta: Software Help > SDK > Dynamixel SDK > USB2Dynamixel > Windows.

Paso 4: Creación del proyecto Windows Presentation Foundation (WPF)

La instalación del Kinect SDK no añade funciones adicionales a la copia de Visual Studio Express 2012 en el equipo. En su lugar, una biblioteca de clases se copia en su PC de Windows. Estas clases contienen el código del programa que se comunica a través de los controladores USB con el sensor de Kinect. Cuando desee crear un programa que utiliza el sensor Kinect, debe agregar este archivo de biblioteca a su proyecto.

Visual Studio ofrece tres clases de aplicaciones fundamentales: aplicaciones de Consola, aplicaciones Windows Forms y aplicaciones Windows Presentation Foundation. Las WPF son las más apropiadas para interactuar con el Kinect, porque son el enfoque más reciente y avanzado de Microsoft® a un marco de GUI, permiten crear aplicaciones con una amplia gama de elementos gráficos, cuyo nivel de interacción conjuga son el sensado en tiempo real del sensor 3D, y brindan una flexibilidad de trabajo muy confortable, lo que permite a los desarrolladores centrarse en construir aplicaciones muy pulidas.

Fuente: "Current Version Plugin Windows Presentation Foundation", Version Plugins, 2013.

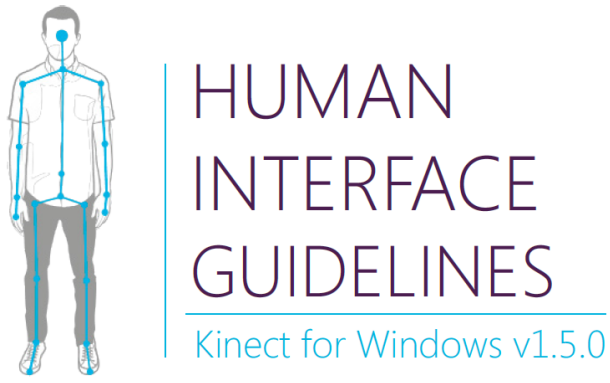
Fig. 5.5. Windows Presentation Foundation.

Las aplicaciones WPF utilizan un lenguaje especial para su diseño gráfico, conocido como XAML, que significa *eXtensible Application Markup Language* (Lenguaje Extensible de Formato para Aplicaciones), es la variante de Microsoft® de XML para describir una GUI. En Windows Forms, los elementos gráficos se crean en el mismo idioma en que se usa para interactuar con la interfaz, por ejemplo, C # o VB.NET, pero con XAML, Microsoft® va por otro camino. Al igual que con HTML, que son capaces de escribir y editar una GUI fácilmente.

Paso 5: Configuración y programación del Kinect

Hasta aquí, se tiene todo el software necesario para comenzar el desarrollo de la interfaz intuitiva basada en los lenguajes C# y XAML. No debe olvidarse que el objetivo es lograr no solo una configuración adecuada del Kinect, sino también un nivel de entendimiento Usuario-Kinect-Robot, para crear todo ese entorno de control del que se habló a inicios del capítulo[20].

[20] Para proveer conocimientos al desarrollador de NUIs, sobre como interactuar con Kinect e inspirarlo a crear una interfaz interesante y fácil de usar, se recomienda descargar la guía "*Human Interface Guidelines*" del siguiente link: http://es.scribd.com/doc/96708567/Human-Interface-Guidelines-v1-5-0.



Fuente: "Human Interface Guidelines", Microsoft Corporation, 2012.

Fig. 5.6. Portada del manual del Kinect para Windows SDK v1.5.

Antes de iniciar con la programación, es necesario agregar (desde la ventana de Solución de Explorador) la referencia *Microsoft.Kinect* y la librería *dynamixel.cs*, para poder utilizar los comandos del Kinect SDK y el Dynamixel SDK, respectivamente. Así también, para habilitarlas en la aplicación, deben ser añadidas a través de la función *using* en la cabecera del programa, junto a las demás librerías que por defecto se generan al crear este tipo de aplicación.

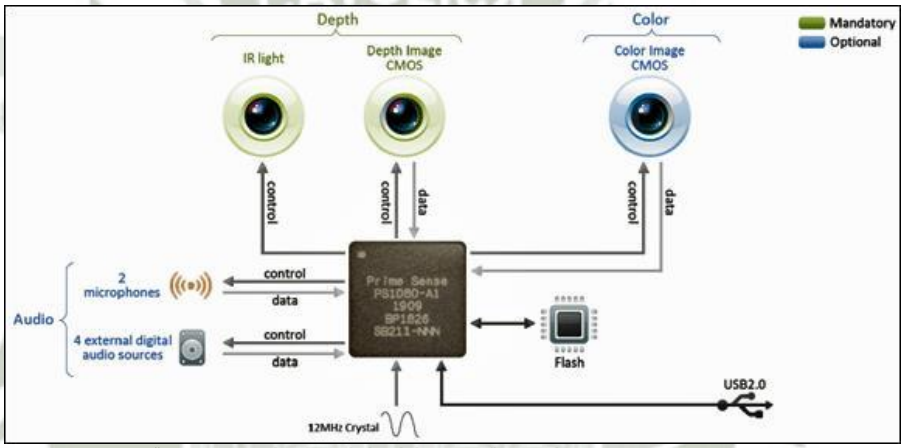
El código realizado en WPF de Visual Studio, está dividido en cuatro regiones: Kinect Setup, Skeleton Setup, Video Camara Setup y Elevation angle. Por su gran extensión y complejidad, se procede a resumir lo elaborado en cada sección. El detalle del código por extenso puede ser consultado en la sección de Anexos.

```
 9  using System.Windows.Data;
10  using System.Windows.Documents;
11  using System.Windows.Input;
12  using System.Windows.Media;
13  using System.Windows.Media.Imaging;
14  using System.Windows.Navigation;
15  using System.Windows.Shapes;
16  using Microsoft.Kinect;
17  using ROBOTIS;
18
19  namespace ControlKinect
20  {
21      /// <summary>
22      /// Interaction logic for MainWindow.xaml
23      /// </summary>
24      public partial class MainWindow : Window
25      {
26          // Dynamixel Setup
27          public const int DEFAULT_PORTNUN = 4; // COM 4
28          public const int DEFAULT_BAUDNUN = 1; // 1 Mbps
29
30          public MainWindow()
31          {
32              InitializeComponent();
33          }
34
35          Kinect Setup
76
77          Skeleton Setup
314
315         Video Camara setup
352
353         Elevation angle
369
370         private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
371         {
372             shutdownKinect();
373         }
374     }
375 }
```

Fuente propia.

Fig. 5.7. Regiones de la aplicación WPF.

La región **Kinect Setup** alberga un evento llamado *Window_Loaded*, que contiene la configuración básica del sensor, la cual será ejecutada apenas se apertura la ventana principal de la aplicación. Dicha configuración crea un objeto del tipo *KinectSensor* llamado *myKinect*, comprueba si existe algún Kinect disponible (de lo contrario envía un mensaje de que no detectó ningún Kinect), lo reconoce, habilita los flujos de color y esqueleto e inicializa el sensor. De no llevarse a cabo una inicialización correcta, la aplicación enviará una mensaje al respecto. Luego se crean los eventos correspondientes a cada flujo (*ColorFrameReady* y *SkeletonFrameReady*), a través de los cuáles es posible actualizar las funciones de los marcos de color y esqueleto cada vez que se encuentren disponibles. Finalmente, se crea el método *shutdownKinect* que permite detener el sensor de una forma segura.



Fuente: "PrimeSense: The magic behind the Kinect", New Florence. New Renaissance., 2010.

Fig. 5.8. ¿Cómo funciona Kinect internamente?

La región **Skeleton Setup**, inicia con la creación de los métodos *addLine* y *drawPoint*, los cuales permiten dibujar los huesos y articulaciones del esqueleto del usuario, respectivamente. El método *jointDistance* permite medir la distancia entre 2 articulaciones (posteriormente se verá que si la distancia entre las manos del usuario es menor a 0.65 m, se activan los actuadores y el sensado de la orientación). En el evento *SkeletonFrameReady* se realiza toda la configuración para escanear el esqueleto del usuario, obtener las coordenadas de posición XYZ de la mano derecha (la cual controla los grados destinados a posicionar el extremo final del robot), obtener los datos de orientación de la muñeca izquierda (la cual controla los grados destinados a orientar el extremo final del robot). También se crea la condición *ClippedEdges* que permite determinar la calidad del escaneo, lo que le permite al usuario saber a qué distancia debe colocarse del sensor. Finalmente, se añade la cinemática inversa del robot, y se establece el envío de las coordenadas articulares (transformadas en una cadena de bits) hacia los actuadores, a través de la función *dynamixel.dxl_write_word*.

La región **Video Camara Setup**, configura la cámara RBG, para que pueda funcionar como una cámara web, a través de la cual se pueda visualizar el entorno que rodea al usuario. Esta sección contiene el evento *ColorFrameReady* donde se configura el flujo de color y los parámetros de imagen (longitud de los datos de pixeles, resolución, formatos, entre otros).

Por último, la región **Elevation angle** permite ajustar, a través de un slider, el ángulo de elevación del Kinect entre ± 27°. El evento *Window_Closing* permite ejecutar el método de detención del Kinect cuando se cierra la ventana de la aplicación.

Finalizada toda la programación en C# comentada líneas arriba, es necesario visualizar su contenido a través de un entorno gráfico. Es aquí donde el lenguaje XAML toma protagonismo, porque permite crear todos los elementos gráficos necesarios como ventanas de visualización, cajetines de texto y numéricos, *sliders*, etiquetas, entre otros. La Fig. 5.9. muestra el entorno gráfico creado a través de XAML. La aplicación está dividida entre 3 secciones. La primera sección, encabezada por la etiqueta "Escaneo del usuario", contiene una ventana de visualización, donde se presentarán las imágenes que captura la cámara RBG y se dibujará el esqueleto escaneado del usuario. La segunda sección, ubicada en la parte inferior, muestra el estado del esqueleto (una vez escaneado, muestra las coordenadas XYZ de la mano izquierda), la calidad del escaneo (que le indica al usuario a que distancia debe ubicarse el usuario una vez que se encuentre en frente del sensor) y la distancia entre ambas manos. La última sección (superior derecha) está conformada por las etiquetas "Ángulos de orientación" y "Ángulo Kinect". Bajo la primera, es posible visualizar los valores numéricos de la orientación, representada en ángulos de Euler, mientras que la segunda alberga un slider que permite ajustar el ángulo de inclinación del sensor.

Fuente propia.

Fig. 5.9. Entorno gráfico creado en XAML.

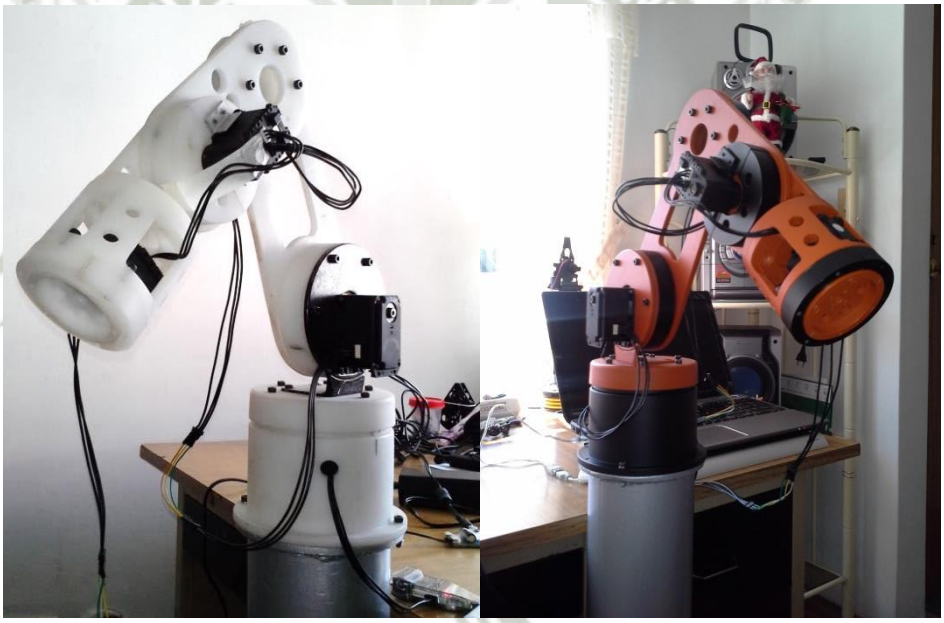
Debajo de la ventana que contiene el entorno gráfico descrito, se encuentra una sección de líneas de código. Es allí donde se escribe el código XAML, cuyo detalle también puede ser encontrado en la sección de Anexos. Una vez creados los componentes gráficos, es posible editar sus dimensiones, ubicación en el entorno gráfico, tipo y color de letra, entre múltiples opciones de formato. Se añade una pequeña imagen del sensor Kinect muy cerca del slider, a modo de guiar al usuario sobre cómo debe ajustar el ángulo de elevación.

Capítulo VI

Pruebas, Resultados y Análisis

6.1. Fabricación del brazo robótico

En el Capítulo III se realizó el diseño completo de brazo robótico, que concluyó con la elaboración de los planos correspondientes de todas las piezas de su estructura. Basados en ellos, y gracias a la maestranza "Edmundo Talavera Sánchez" pudo fabricarse todas y cada una de dichas piezas; así también, se realizó su ensamblado y montaje. La Fig. 6.1. presenta el robot finalizado, con los actuadores instalados y debidamente conectados.



Fuente propia.

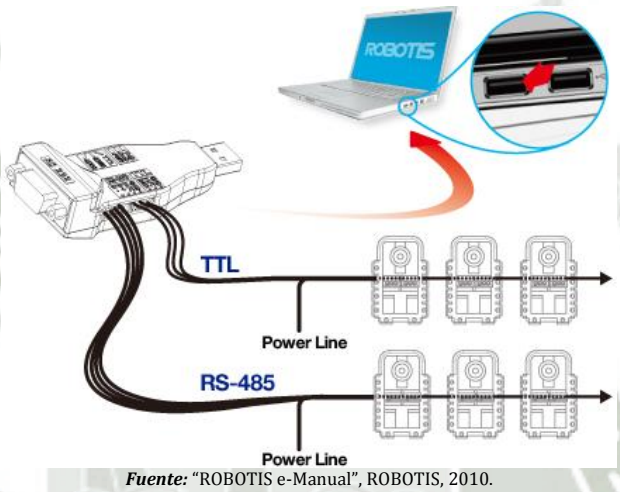
Fig. 6.1. Brazo robótico Darko fabricado.

Fuente propia.

Fig. 6.2. Maestranza "Edmundo Talavera Sánchez" y pieza en fabricación.

6.2. Pruebas y resultados del Control Kinect

Realizada la aplicación de la interfaz intuitiva, se está en condiciones de evaluar el comportamiento del Control Kinect. Para ello, primero se deben conectar en cadena todos los actuadores, de manera que se tenga un cable de inicio que se conecta, por un extremo con el controlador USB2Dynamixel y por el otro al actuador de la base. Luego se procede a conectar uno detrás de otro los actuadores, hasta que se tiene un último cable, el cual irá a un adaptador (SMPS2Dynamixel) que lo une a la fuente de alimentación. Todo este procedimiento se encuentra graficado en la Fig. 6.3. La serie de TTL (3 líneas) es la utilizada.



Fuente: "ROBOTIS e-Manual", ROBOTIS, 2010.

Fig. 6.3. Conexión USB2Dynamixel y actuadores.

Fuente: "ROBOTIS e-Manual", ROBOTIS, 2010.

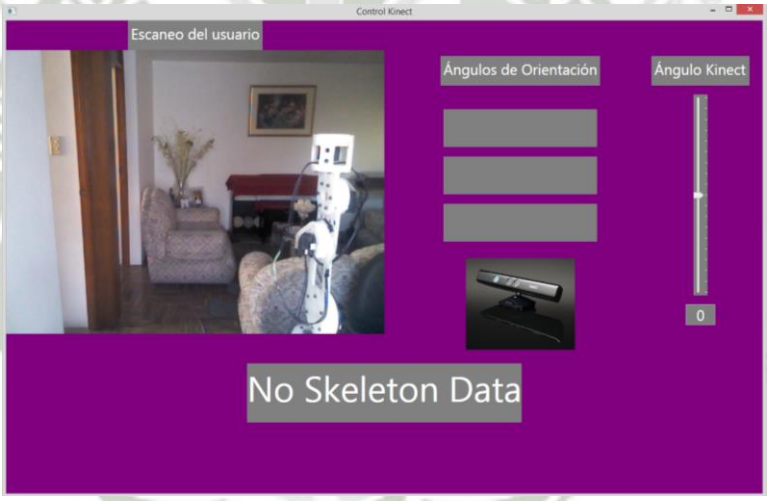
Fig. 6.4. Adaptador SMPS2Dynamixel.

La segunda conexión que se debe realizar, es la del Kinect con la PC. El Kinect para Windows utilizado, incorpora consigo un adaptador, que permite convertir el conector Kinect (único medio de comunicación del sensor) en un conector USB. Esto debido a que, como se recuerda, el Kinect fue creado inicialmente para la consola Xbox 360, la cual posee un puerto de color naranja, muy similar al típico puerto USB que se conoce, pero en definitiva no es un puerto USB estándar. Microsoft® no cambió el puerto Kinect a USB en la versión del periférico para PC, probablemente para no restarle versatilidad al producto. No se debe intentar conectar el sensor a la PC sin el adaptador mostrado en la Fig. 6.5., ya que de forzar dicha conexión, es posible dañar permanentemente el puerto USB de la PC, y el conector propio del Kinect.

Fuente: "Kinect PC Adapter", Virtual Sensei Lite, 2011.

Fig. 6.5. Vista posterior del Kinect y adaptador para PC.

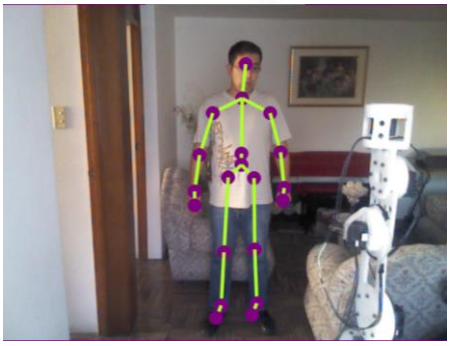
Una vez que se tiene todo conectado, se procede a ejecutar la aplicación WPF creada, la cual permite realizar el Control Kinect a través de una interfaz natural de usuario. La Fig. 6.6. captura el primer instante de ejecución del Control Kinect. Se visualiza todo un entorno, en el que se encuentra el robot en su posición de reposo, y se muestra un mensaje de que no existen datos de algún esqueleto humano (*No Skeleton Data*), debido a que no se tiene la presencia de ningún usuario.



Fuente propia.

Fig. 6.6. Primer instante de ejecución del Control Kinect.

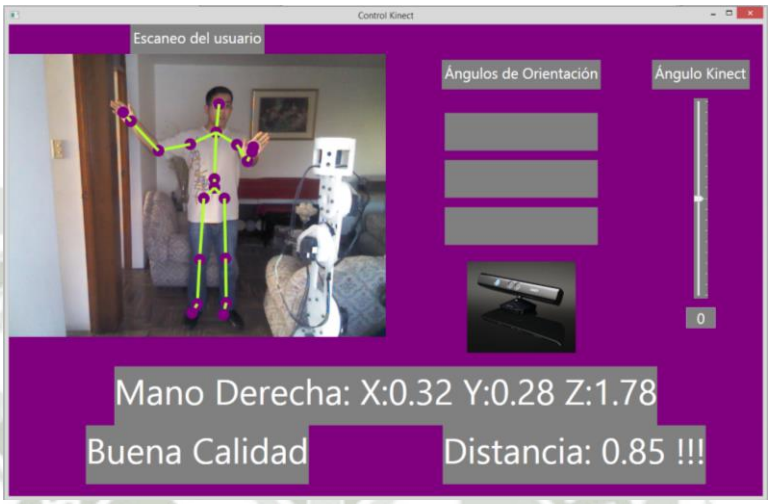
Cuando el usuario se sitúa en la zona efectiva de escaneo, el Kinect procede a identificarlo y graficar su esqueleto por completo, como se muestra en la Fig.6.7.



Fuente propia.

Fig. 6.7. Escaneo del esqueleto del usuario.

Para iniciar la comunicación entre el usuario y el robot (esto implica habilitar la adquisición de los datos de posición y orientación y activar los actuadores.), debe aproximarse ambas manos a una distancia menor a 0.65 m[21]. De lo contrario no se podrá iniciar el control (Fig. 6.8.).



Fuente propia.

Fig. 6.8. Posiciones para iniciar el Control Kinect.

Una vez que el sensor detecta el esqueleto humano y se cumple la condición de distancia entre manos (Fig. 6.9. y Fig. 6.10.), es capaz de facilitar información detallada de la posición exacta (coordenadas X, Y, Z) de la mano derecha, y de la orientación en el espacio de la muñeca de la mano izquierda. Es gracias a esa información lo que hace posible desarrollar el Control Kinect, que permite que la muñeca del robot se posicione según las coordenadas de la mano derecha, y el extremo final se oriente según los ángulos de orientación de la muñeca izquierda.

Fuente propia.

Fig. 6.9. Control Kinect en tiempo real.

[21] Los signos de admiración son un indicador adicional que denotan que la distancia entre ambas manos es mayor a 0.65m. Valores inferiores a este límite son indicados a través de un "Ok".

Fuente propia.

Fig. 6.10. Control Kinect en tiempo real II.

Se optó por separar el sensado de posición (mano derecha) y orientación (muñeca izquierda), por fines didácticos (para que el usuario posea una idea más clara de las diferencias que existen entre posicionar y orientar) y para brindar una experiencia más inmersiva y natural (mientras más partes del cuerpo humano intervengan, mayor será el efecto de interacción usuario-robot).

Como puede apreciarse en las imágenes presentadas anteriormente, los resultados son favorables. El usuario posiciona la muñeca del robot al mover su mano derecha en los 3 eje coordenados, y orienta el extremo del robot al girar su muñeca izquierda. Lo más destacable, es que todo ello se realiza en tiempo real, es decir que no existen retardos de tiempo entre el instante que se mueve alguna articulación y el instante en el que se envían los datos sensados a los actuadores. En otras palabras, el Control Kinect es un control en tiempo real que permite una interacción sencilla, rápida y efectiva con el robot.

6.3. Análisis comparativo

Hasta aquí, se ha llevado acabo un desarrollo progresivo de cada etapa que forma parte de la investigación, direccionando en un solo sentido todos los esfuerzos posibles para obtener una plataforma científica que conjugue con los objetivos trazados y cuyo funcionamiento ha sido evaluado al realizar diferentes pruebas o ensayos, los cuales permitieron obtener los resultados esperados.

Sin embargo, a este nivel de desarrollo de la tesis, es posible que surjan algunas preguntas o cuestionamientos sobre lo realizado, pertinentes tanto a la parte teórica (conocimiento científico utilizado) como práctica (pruebas realizadas y resultados obtenidos). Esta sección intenta responder dichas interrogantes a través de un análisis comparativo bajo el perfil del ¿por qué se optó por un método

de diseño, parámetro físico, estrategia de control, algoritmo computacional, software de desarrollo, material estructural, dispositivo determinado, y no por otro diferente? Además, brinda un breve análisis de los resultados obtenidos, a modo de verificar tanto si los objetivos inicialmente planteados fueron alcanzados, como la hipótesis validada.

Para la representación de la orientación, ¿por qué se escogieron los Ángulos de Euler y no las Matrices de rotación, o el Par de rotación o los Cuaternios?

Los ángulos de Euler, a pesar de no ofrecer una visualización sencilla de la orientación, en cualquiera de sus modalidades, permiten una notación compacta de la orientación (sólo es necesario definir tres números reales para su descripción) y facilitan la aplicación del método de desacoplo cinemático para la adquisición de la cinemática inversa de robots manipuladores con 5 o 6 GDL, como se demostró en el Capítulo III.

Es sabido que las matrices de rotación son el método más extendido para la descripción de orientaciones, pero tienen como principal inconveniente su extensión, ya que es necesario definir nueve elementos para describir una orientación, lo que crea dificultades para su implementación computacional en tiempo real.

Por otro lado, el par de rotación utiliza 4 parámetros para la definición de orientación (tres componentes cartesianas el vector k y un ángulo de rotación θ) y la composición de rotaciones presenta una expresión complicada, lo que limita su utilización práctica en algunas aplicaciones.

Finalmente, lo cuaternios sólo son capaces de representar la orientación relativa de un sistema con respecto a otro, a través del uso de cuatro componentes, y su uso es recomendado para el tratamiento de giros y cambios de orientación complejos, como es el caso de robots industriales de 6 o 7 GDL.

Para la obtención de la cinemática inversa, ¿por qué se eligió el método de desacoplo cinemático y no métodos geométricos o aquellos basados en MTH?

Los métodos geométricos o aquellos basados en MTH son útiles y recomendables para robots de hasta 3 GDL, porque permiten obtener con facilidad, los valores de las tres primeras variables articulares del robot, aquellas que posicionan su extremo en unas coordenadas determinadas. De ser aplicados a robots de 5 o 6 GDL, su complejidad aumenta exponencialmente, lo cual los hace inadecuados para su implementación computacional.

Como el robot diseñado es de 5 GDL (tres grados dedicados a posicionar el extremo y dos a orientarlo), los métodos anteriores quedan descartados, y el método de desacoplamiento cinemático adquiere protagonismo, porque permite resolver los primeros grados de libertad dedicados al posicionamiento, de manera independiente a la resolución de los últimos grados de libertad, dedicados a la orientación. Cada uno de estos dos problemas más simples puede ser tratado y resuelto por cualquiera de los procedimientos anteriores, lo cual facilita el cálculo matemático y su implementación por medio de software.

Para la obtención de la dinámica inversa, ¿por qué se utiliza el algoritmo Luh-Walker y no los algoritmos de L-E y N-E en su forma clásica?

Los algoritmos computacionales analizados en el Capítulo III, permiten modelar el comportamiento dinámico del robot con una buena aproximación a la realidad. Sin embargo, las ecuaciones de L-E son muy difíciles de utilizar con fines de control en tiempo real, a menos que se simplifiquen considerablemente, debido a que es necesario calcular los coeficientes dinámicos H, C y G para cada valor de q_i y $\dot{q}_i$, lo que lamentablemente requiere una extensa cantidad de operaciones aritméticas. La Tabla 6.1. muestra las complejidades computacionales de los métodos clásicos para calcular la dinámica del robot.

Tabla 6.1. Complejidades computacionales de la dinámica del robot.

Método	Lagrange-Euler	Newton-Euler	D'Alembert generalizado
Multiplicaciones	$^{128}/_{3}\,n^4 + {}^{512}/_{3}\,n^3 + {}^{844}/_{3}\,n^2 + {}^{76}/_{3}\,n$	$132n$	$^{13}/_{6}\,n^3 + {}^{105}/_{2}\,n^2 + {}^{268}/_{3}\,n + 69$
Adiciones	$^{98}/_{3}\,n^4 + {}^{781}/_{6}\,n^3 + {}^{637}/_{3}\,n^2 + {}^{107}/_{6}\,n$	$111n - 4$	$^{4}/_{3}\,n^3 + 44n^2 + {}^{146}/_{3}\,n + 45$
Representación cinemática	Matrices homogéneas 4x4	Matrices de rotación y vectores de posición	Matrices de rotación y vectores de posición
Ecuaciones de movimiento	Ecuaciones diferenciales en forma cerrada	Ecuaciones recursivas	Ecuaciones diferenciales en forma cerrada

Fuente: K. S. Fu, R. C. Gonzalez, C. S. G. Lee, "Robótica: Control, detección, visión e inteligencia", 1988.

Como alternativa para derivar ecuaciones de movimientos más eficientes, se desarrollaron algoritmos para calcular las fuerzas/pares generalizados, basados en las ecuaciones de N-E. Uno de estos algoritmos es el de Luh-Walker-Paul, considerado el algoritmo recursivo más eficiente para robots manipuladores de hasta 6 GDL.

Las razones que llevaron a escoger este tipo de algoritmo radican en lo siguiente: su obtención es simple, implica términos de producto vectorial (en lugar de matrices homogéneas 4x4) y sus ecuaciones dinámicas resultantes, excluyendo la dinámica del dispositivo de control, holguras y rozamiento de engranajes, son un conjunto de ecuaciones recursivas hacia adelante y hacia atrás. Este conjunto de ecuaciones se puede aplicar secuencialmente a los elementos del robot.

El resultado más significativo de esta formulación, es que el tiempo de cálculo de las fuerzas/pares generalizados se encuentra que es linealmente proporcional al número de articulaciones del brazo e independientemente de la configuración del mismo. Con este algoritmo, se puede realizar el control en tiempo real simple del robot, en el espacio de las variables de articulación.

Al definir el material de la estructura mecánica del robot, ¿por qué se optó por un plástico (Nylon 6) y no por un metal?

Como plantea uno de los objetivos secundarios, se desea diseñar una estructura robótica robusta, segura, ligera y transportable. Bajo esta premisa, se consideró que, para que un material aporte estas características al cuerpo del robot, debe tener baja densidad (de forma que se minimice el peso del robot), alta resistencia mecánica (para que soporte impactos y aporte estabilidad dimensional a lo largo del proceso de fabricación), reciclable (porque todo diseño debe proteger el medio ambiente), buena facilidad de mecanizado (para reducir los costos de fabricación), y adquisición accesible en el mercado local.

Luego de una exploración por el mundo de los materiales, se eligió al Nylon 6 como el material ideal para la estructura mecánica del robot. La Tabla 6.2. establece una comparación genérica de las propiedades principales del plástico y el metal, lo que refuerza la decisión tomada.

Tabla 6.2. Comparación general Plástico vs. Metal.

Características	Plástico	Metal
Densidad	Baja	Alta
Precio	Bajo	Alto
Procesamiento	Fácil	Difícil
Corrosión	----	Alta
Consumo de Energía	Bajo	Alto
Flexibilidad	Alta y Baja	Rígido
Transparencia	Alta y Baja	Opaco
Conducción de electricidad	Aislante	Alta

Fuente propia.

Las piezas del robot se obtuvieron a través de diversos procesos de mecanizado. De tener una impresora 3D, también pudo haberse utilizado este material.

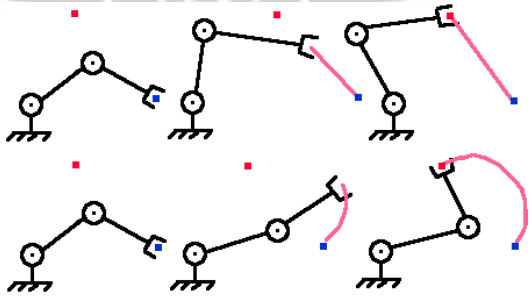
¿Por qué trayectorias articulares y no cartesianas?

En un brazo robótico, la forma más rápida de realizar una trayectoria entre dos puntos, a menudo no es una línea recta. Si dos articulaciones tienen dos motores diferentes o llevan diferentes cargas, la velocidad máxima puede variar entre ellas. Cuando se le indica al efector final para ir de un punto a otro, se tienen dos opciones: la primera, que trace una línea recta entre dos puntos, o la segunda, indicarle a todas las articulaciones que deben ir lo más rápido posible hacia el punto final, dejando el efector final posiblemente oscilando bruscamente entre esos puntos.

La Fig.6.11. grafica el comportamiento del extremo del brazo robótico desde una posición inicial (punto azul) hasta una posición final (punto rojo), cuando se describe una trayectoria cartesiana y una articular, respectivamente.

En la secuencia superior, el extremo se desplaza en línea recta. Este es el único movimiento posible que este brazo puede realizar al viajar en línea recta. En la secuencia inferior, se le indica al brazo que debe llegar al punto rojo lo más rápido posible. Dado que existen muchas trayectorias diferentes entre ambos puntos, el brazo evalúa el método que le permite a las articulaciones girar lo más rápido posible.

¿Qué método es mejor? Hay muchos factores que intervienen para emitir una respuesta. Por lo general, se desea las trayectorias en línea recta cuando la carga a transportar por el brazo es muy pesada, ya que requiere el cambio de momento para el movimiento (impulso = masa * velocidad). Pero para la velocidad máxima (tal vez el brazo no se lleva nada, o sólo objetos de luz), es recomendable trayectorias en el espacio articular, debido a que se desprecia el uso de la cinemática inversa.



Fuente: "Robot Arm Tutorial", Society of Robots, 2005.

Fig. 6.11. Comparación de trayectorias.

¿Por qué Kinect para Windows y no Kinect para Xbox 360?

Para conocer qué diferencia al Kinect para Windows del Kinect para Xbox 360, la Tabla 6.3. compara sus principales prestaciones.

Tabla 6.3. Kinect para Windows vs. Kinect para Xbox 360.

	KINECT FOR WINDOWS	KINECT FOR XBOX 360
Cámara Profundidad	– 320x240 640x480 – 8m, 4m – 5m, 3m (Near Mode)	– 320x240, 640x480 – 8m, 4m
Cámara RGB	– 320x240, 640x480, 1024x768	– 320x240, 640x480
Inclinación	– Desde -27 a 27 grados	– Desde -27 a 27 grados
Audio	– 4 micrófonos – Audio direccional – Detección de audio	– 4 micrófonos – Audio direccional – Detección de audio
Características	– Características adicionales como Near-Mode (Modo cercano), seated mode (modo sentado, USB cable corto, cámara con más resolución.	– Características Básicas
Precio	– Oscila entre 200$ a 250 $ USD – SDK gratuito	– Oscilan entre 100 $ a 150$ USD – SDK gratuito
Licencia	– Para desarrolladores y uso publico	– Para desarrolladores y uso en el XBOX 360 – No es para uso publico

Fuente: "What is the difference between Kinect for Windows & Kinect for Xbox360?", Kinecting for Windows, 2012.

De la comparación realizada, es evidente que el Kinect para Windows ofrece varias características que no están habilitadas cuando se utiliza un Kinect para Xbox 360, las cuales son:

1. **Modo cercano**
   Permite a la cámara ver objetos a distancias de hasta 40 centímetros en frente del dispositivo sin perder exactitud o precisión, con una degradación elegante a 3 metros.

2. **Modo sentado con "10 articulaciones"**
   Seguimiento del esqueleto que proporciona la capacidad de realizar un seguimiento de la cabeza, el cuello y los brazos de un usuario, ya sea sentado o de pie.

3. **Cable adaptador de USB incorporado**
   Asegura la fiabilidad en una amplia gama de equipos y mejora la convivencia con otros periféricos USB.

4. **Ajustes extendidos de la cámara**
   Proporciona ajustes adicionales tales como el brillo, exposición, etc. para que pueda sintonizar aún más.

5. **Kinect Fusion**
   Mapea el entorno en 3D sobre la marcha o le permite utilizar la opción de reemplazo de objetos.

6. ***Handgrip***
   La detección de la mano permite implementar gestos como pellizcar para hacer zoom, agarrar, etc., para mejorar sus aplicaciones y construir un nuevo tipo de aplicaciones en conjunto.

En cuanto a su precio oficial, el Kinect para Windows cuesta alrededor de $ 250 mientras que el Kinect para Xbox 360 es más económico y bordea los $ 150. Sin embargo, para este último, es posible que usted se vea obligado a comprar una fuente de alimentación por separado para que pueda conectarlo a su PC, lo cual incrementa su costo original en unos $ 35. El Kinect para Windows SDK es de uso gratuito y se encuentra disponible para ambos.

Si se explora el Microsoft FAQ (Frequently Asked Questions) acerca de las licencias que se incluyen en la compra de un Kinect para Windows, se obtendrá lo siguiente:

"La licencia comercial autoriza el desarrollo y distribución de aplicaciones comerciales. El SDK beta era sólo para la investigación, pruebas y experimentación, y no era adecuado para su uso con productos comerciales finales. En la nueva licencia que está disponible con el Kinect para Windows versión 1.0 y la versión 1.5 del software, permite a los desarrolladores crear y vender sus aplicaciones de Kinect para Windows a los clientes que utilizan dichas aplicaciones en las plataformas de Windows".

Esto significa que cuando se quiere hacer pública su aplicación, usted tendrá que usar un Kinect para Windows, porque el Kinect para Xbox 360 no es legal. Eso también explica el alto precio del sensor Kinect para Windows, ya que en él también está incluido el pago de las licencias originales, que permiten realizar investigaciones y publicaciones científicas legales y respaldadas por Microsoft®.[22]

Fuente: "What is the difference between Kinect for Windows & Kinect for Xbox360?", Kinecting for Windows, 2012.
Fig. 6.12. Alegoría de ambas versiones del Kinect.

[22] Si desea saber más acerca de la concesión de licencias, puede visitar la siguiente dirección web: http://www.microsoft.com/en-us/kinectforwindows/develop/sdk-eula.aspx

¿Las NUIs poseen algún punto débil?

Una NUI es activada con los gestos, con la voz, por los sentidos del ser humano. Son nuestros prejuicios y la incapacidad de aprender la delicada naturaleza de las interacciones humanas, el punto débil de una NUI. Sin embargo, cada vez más seres humanos están aprendiendo a superar sus propios paradigmas y limitaciones, y pronto llegará el día en que el gesto de un corazón con sus manos tenga el mismo significado tanto para usted como para su computadora.

Afortunadamente la tecnología ha avanzado en más de un área y la fusión de la realidad aumentada y las NUIs son un éxito. Esto es aún más apreciable cuando se ven los videos de productos conceptuales y los testimonios de historias de éxito con Kinect.

Por otro lado, crear una interfaz de usuario que considere el contexto, que tome en cuenta las indicaciones visuales correctas, realimentación y métodos de entrada y salida, y evite caer en los paradigmas de las interfaces de usuario existentes, es todo un reto de en el pasado era inimaginable, pero que ahora es una realidad.

Conclusiones

- ✓ Se diseñó e implementó una plataforma experimental científica llamada Darko, la cual posee un control gestual que permite la interacción intuitiva del usuario con el brazo robótico de 5 GDL a través del sensor Kinect.
- ✓ Gracias a la metodología de diseño empleada, fue posible diseñar una estructura robótica robusta, segura, ligera y transportable, así como validar la funcionalidad de cada una de sus piezas. Por otro lado, a través del criterio de *solvability*, pudo cumplirse el requisito de que su cinemática sea solucionable.
- ✓ El sistema de control Kinect diseñado permite alcanzar un buen nivel de entendimiento Usuario-Kinect-Robot en tiempo real, y convierte el cuerpo humano en el mando regulador de movimiento siguiendo la filosofía "nada se interpone entre el robot y yo", es decir, prescindir de los típicos dispositivos y superficies físicas de control.
- ✓ Se desarrolló una interfaz de usuario natural basada en un lenguaje de programación de alto nivel (C#), que permite la interacción en tiempo real hombre-máquina, de una manera sencilla, cómoda, rápida e intuitiva, a través de los gestos y movimientos que el usuario realiza con sus brazos.
- ✓ Un aporte adicional de esta tesis en el área de robótica, es la aplicación del Algoritmo Luh-Walker-Paul, considerado el algoritmo recursivo más eficiente para el cálculo de torques instantáneos en robots manipuladores de hasta 6 GDL. Esto permitió alcanzar tal nivel de profundidad en la investigación, que fue posible aproximarse al saber del cómo funcionan verdaderamente las cosas.
- ✓ Los *Smart Servos* de Dynamixel, superan con creces a los típicos servomotores RC de *hobby*, debido a que poseen una tecnología avanzada, con muy altas prestaciones a nivel de robótica personal. Su utilización constituye una innovación en el campo de los actuadores, pues son escasamente conocidos en nuestro medio.
- ✓ La técnica de control PID con compensación de gravedad utilizada en su variante de control de movimiento, es muy poderosa y efectiva, porque permite realizar un control dinámico del robot eficiente, versátil para diferentes trayectorias y muy aproximado a la realidad.

Observaciones

- ✓ Aunque no se dispuso de un *toolbox* de robótica que se ajuste exactamente a las exigencias particulares del análisis dinámico y control del robot presentado, fue posible implementar en Matlab® un código programado, propio y lo suficientemente personalizado, que permitió realizar un modelamiento detallado y efectivo de la robótica tratada en la investigación.
- ✓ El Control Kinect que propone la presente investigación, es una muestra clara de investigación, desarrollo e innovación (I+D+I). Además, dotar al brazo robótico de una función de imitación, es una evidencia clara de que las funciones de reconocimiento de gestos del sensor Kinect son adaptables a plataformas de ingeniería, más allá de los simples salones de videojuegos.
- ✓ Las interfaces naturales de usuario reducen significativamente la curva de aprendizaje y permiten que usuarios sin experiencia y sin conocimientos técnicos, puedan interactuar con su entorno obteniendo resultados notables.
- ✓ La tendencia a futuro vislumbra una importante separación entre interfaces para la creación de contenidos (códigos QR, sistemas de acceso RFID y redes sociales) e interfaces para el consumo de contenidos (presentes en dispositivos *Smart* y plataformas de investigación con NUIs).

Recomendaciones

- ✓ Los motores no poseen un freno mecánico, por lo que cuando están sin alimentación eléctrica, no se encuentra habilitado la opción de par. Esto significa que en estas condiciones, son susceptibles al efecto de la gravedad, que hace que la estructura se desplome y no conserve su configuración.
- ✓ Como una extensión a la presente investigación, se recomienda realizar un análisis comparativo entre una planta obtenida por algoritmos computacionales y aquella obtenida en SimMechanics.
- ✓ Para tener una experiencia exitosa con Kinect, no coloque el sensor delante de un altavoz o en una superficie que vibra o en lugar ruidoso, manténgalo fuera de la luz solar directa, no lo utilice cerca de fuentes de calor.
- ✓ No es posible utilizar el sensor Kinect bajo una máquina virtual, debido a que los controladores disponibles aún no soportan un entorno virtualizado de programación.
- ✓ No fuerce físicamente el dispositivo en un ángulo específico. El Kinect para Windows SDK tiene algunas APIs específicas que pueden ayudar a controlar la inclinación del motor del sensor. No incline el motor del Kinect con frecuencia; úselo pocas veces como sea posible y sólo cuando se requiera.
- ✓ Al desconectar el Kinect de la PC, asegúrese de finalizar completamente la ejecución de la aplicación y que el emisor de luz IR esté apagado (no emita ningún haz de luz roja). Luego proceda que interrumpir la alimentación eléctrica. Si interrumpe la comunicación del sensor con la PC arbitrariamente, puede dañarlo.
- ✓ No fuerce físicamente los actuadores, cualquiera que sea su estado de funcionamiento. Podría fatigar el eje, dañar los engranajes o descalibrar el encoder interno que posee cada uno de ellos.

Bibliografía

[1] A. Barrientos, L. F. Peñín, C. Balaguer, R. Aracil, *"Fundamentos de robótica"*, Ed. McGraw-Hill, 2007.

[2] K. S. Fu, R. C. Gonzalez, C. S. G. Lee, *"Robótica: Control, detección, visión e inteligencia"*, Ed. McGraw-Hill, 1988.

[3] J. J. Craig, *"Robótica"*, Ed. Pearson Prentice Hall, 2006.

[4] F. Reyes Cortés, *"Robótica. Control de Robots Manipuladores"*, Ed. Alfaomega, 2011.

[5] H. Asada, J.-J. E. Slotine, *"Robot Analysis and Control"*, Massachusetts Institute of Technology, John Wiley & Sons, Inc., 1986.

[6] J. Y. S. Luh, M. W. Walker, R. P. C. Paul, (1980), "*On-Line Computational Scheme for Mechanical Manipulators"*, School of Electrical Engineering, Purdue University, Vol. 102

[7] R. Kelly, V. Santibáñez, *"Control de Movimiento de Robots Manipuladores"*, Ed. Pearson Prentice Hall, 2003.

[8] R. Kelly, V. Santibáñez, A. Loría, *"Control of Robot Manipulators in Joint Space"*, Springer, 2005.

[9] L. Sciavicco, B. Siciliano, *"Modeling and Control of Robot Manipulators"*, Springer-Verlag London Limited, 2000

[10] F. L. Lewis, D. M. Dawson, C. T. Abdallah, *"Robot Manipulator Control – Theory and Practice"*, Marcel Dekker, 2nd edition, 2004

[11] J. Martínez Verdú, J. M. Sabater Navarro, *"Guía docente para el diseño de robots de servicio"*, AIDICO, Universidad Miguel Hernández de Elche, 2012.

[12] J. Félez Mindán, M.ª L. Martínez Muneta, *"Ingeniería gráfica y diseño"*, Ed. Síntesis S. A., 2008.

[13] P. Deitel, H. Deitel, *"Visual C#® 2012 How to Program"*, Deitel®, Ed. Pearson Prentice Hall, 5th edition, 2011.

[14] J. Sharp, *"Microsoft® Visual C#® 2012 Step by Step"*, Microsoft® Corporation, 2012.

[15] R. Miles, *"Start Here!™ Learn the Kinect™ API"*, Microsoft® Corporation, 2012.

[16] A. Jana, *"Kinect for Windows SDK Programming Guide"*, Packt Publishing, 2012.

[17] C. Giorio, M. Fascinari, *"Kinect in Motion – Audio and Visual Tracking by Example"*, Packt Publishing, 2013.

[18] M. Tamayo y Tamayo, *"Metodología Formal de la Investigación Científica"*, Ed. Limusa Noriega Editores, 1999.

Direcciones Web

- **SINTEF**. Next Generation Robotics for Norwegian Industry. 4 de Octubre 2013. <http://www.sintef.no/home/Information-and-Communication-Technology-ICT-old/Applied-Cybernetics/Projects/NexGenRob/>.
- **IGN Entertainment, Inc**. NASA Uses Kinect and Oculus Rift to Control a Robotic Arm [en línea]. [Fecha de consulta: 10/06/2014]. Disponible en: <http://www.ign.com/articles/2013/12/31/nasa-uses-kinect-and-oculus-rift-to-control-a-robotic-arm>.
- **Wikipedia.** Oculus Rift. [en línea]. [Fecha de consulta: 10/06/2014]. Disponible en: <http://es.wikipedia.org/wiki/Oculus_Rift>.
- **Ensinger, S.A**. Plásticos de Ingeniería. Poliamida. <http://www.ensinger.es/es/materiales/plasticos-de-ingenieria/poliamida/>.
- **Dotmar Universal Plastics**. Nylon. Sustamid 6G. <http://www.dotmar.co.nz/sustamid-6g/sustamid-6g-cast-nylon.html>.
- **Robotics Toolbox**. 28 de Abril 2014. <http://petercorke.com/Robotics_Toolbox.html>.
- **Society of Robots.** Robot Arm Tutorial [en línea]. [Fecha de consulta: 5/04/2014]. Disponible en: <http://www.societyofrobots.com/robot_arm_tutorial.shtml>.
- **ROBOTIS**. e-Manual [en línea]. [Fecha de consulta: 10/04/2014]. Disponible en: <http://support.robotis.com/en/>.
- **MathWorks® Documentation.** Install and Register SimMechanics Link Software [en línea]. [Fecha de consulta: 22/05/2014]. Disponible en: <http://www.mathworks.com/help/physmod/sm/ug/install-and-register-simmechanics-link-software.html>.
- **E-Centro.** Kinect [en línea]. [Fecha de consulta: 17/09/2014]. Disponible en:< http://centrodeartigo.com/articulos-enciclopedicos/article_96478.html>
- **Kinecting for Windows.** Introduction to Kinect for Windows SDK [en línea]. [Fecha de consulta: 24/09/2014]. Disponible en: <http://www.kinectingforwindows.com/>.

- **Kinect for Developers.** ¿Qué es el SDK para el dispositivo Kinect de Microsoft? [en línea]. [Fecha de consulta: 06/10/2014]. Disponible en: <http://www.kinectfordevelopers.com/es/2012/11/06/que-es-el-sdk-de-microsoft/>.

- **Microsoft Developer Network**. Kinect for Windows SDK. Getting Started [en línea]. [Fecha de consulta: 08/10/2014]. Disponible en: <http://msdn.microsoft.com/en-us/library/hh855354.aspx>.

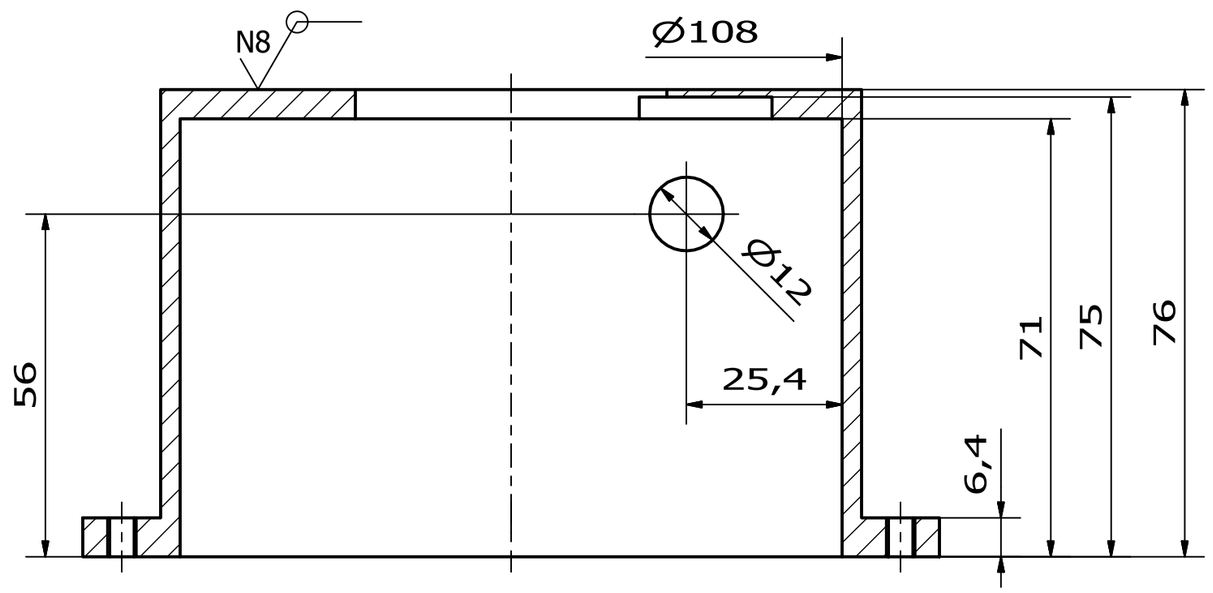
- **WPF Tutorial.** What is WPF? [en línea]. [Fecha de consulta: 11/10/2014]. Disponible en: <http://www.wpf-tutorial.com/about-wpf/what-is-wpf/>.

- **Wikipedia.** Interfaz de Usuario [en línea]. [Fecha de consulta: 15/10/2014]. Disponible en: <http://es.wikipedia.org/wiki/Interfaz_de_usuario>.

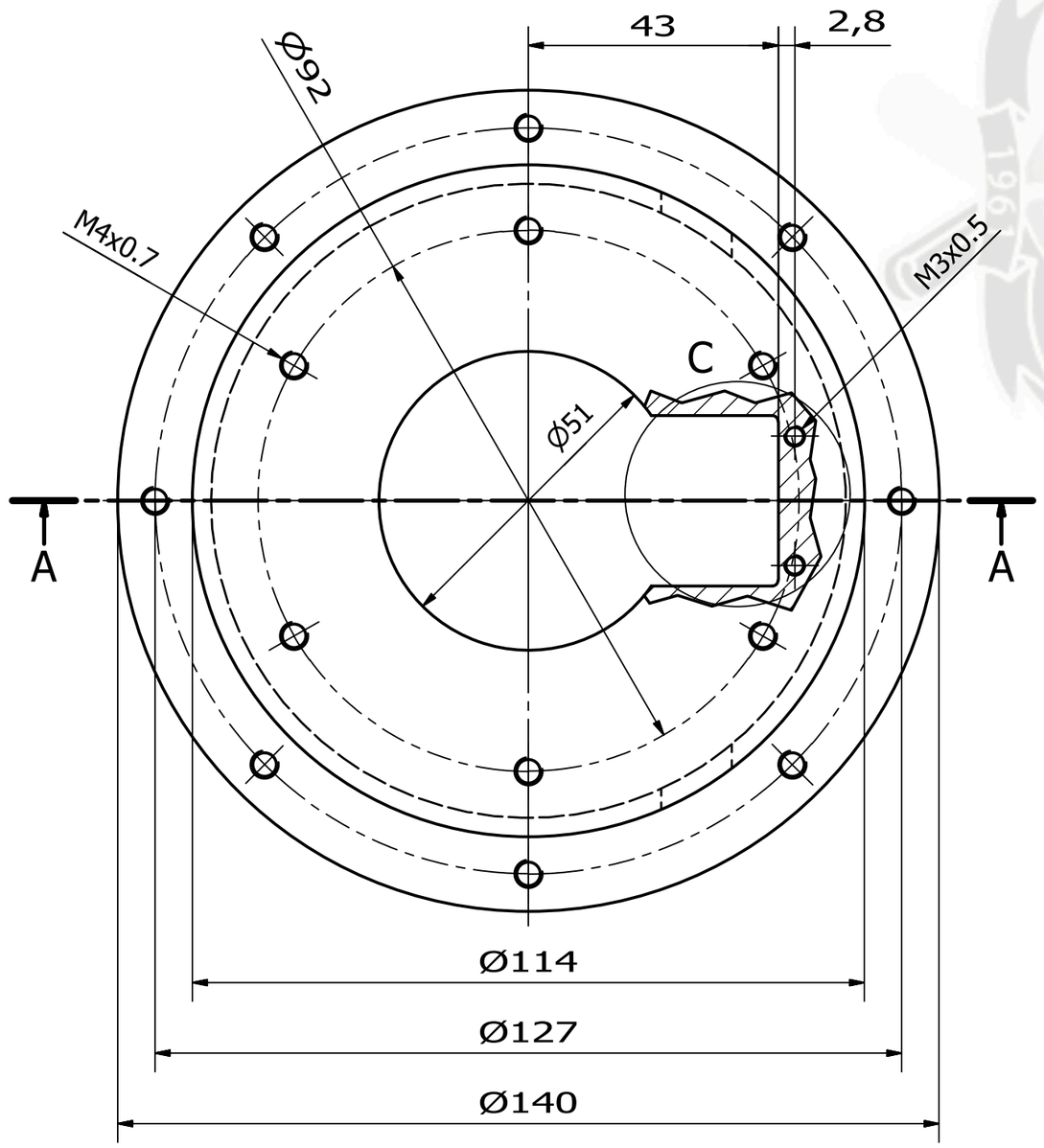
- **Wikipedia.** Interfaz Natural de Usuario [en línea]. [Fecha de consulta: 15/10/2014]. Disponible en: <http://es.wikipedia.org/wiki/Interfaz_natural_de_usuario>.

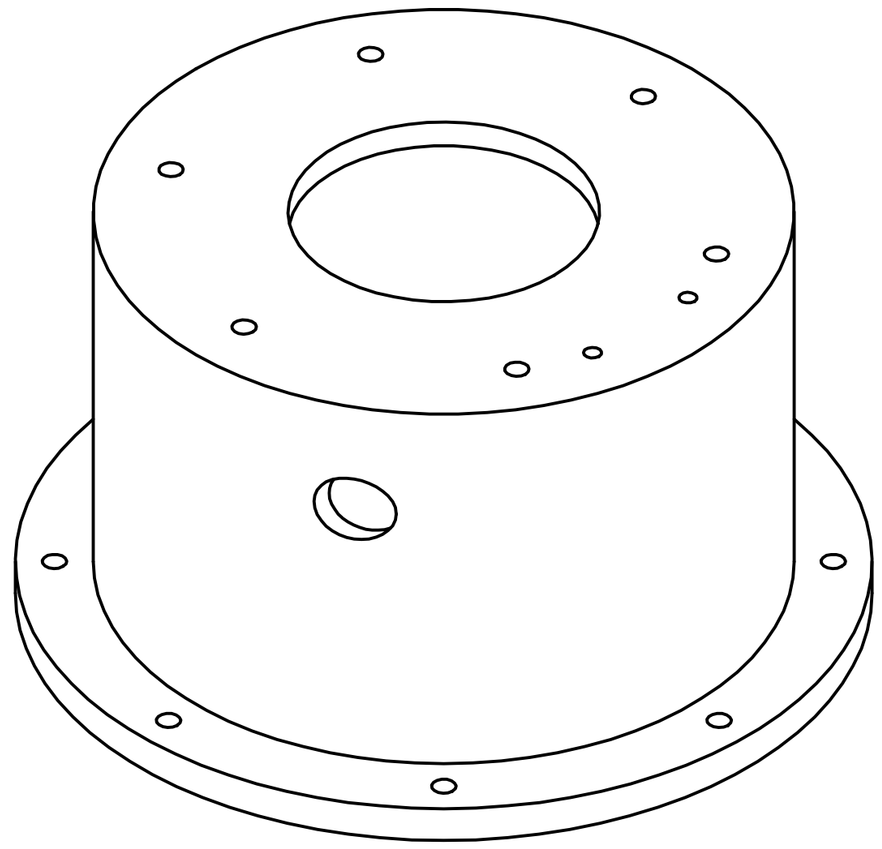
Anexos

Anexo A: Planos del robot

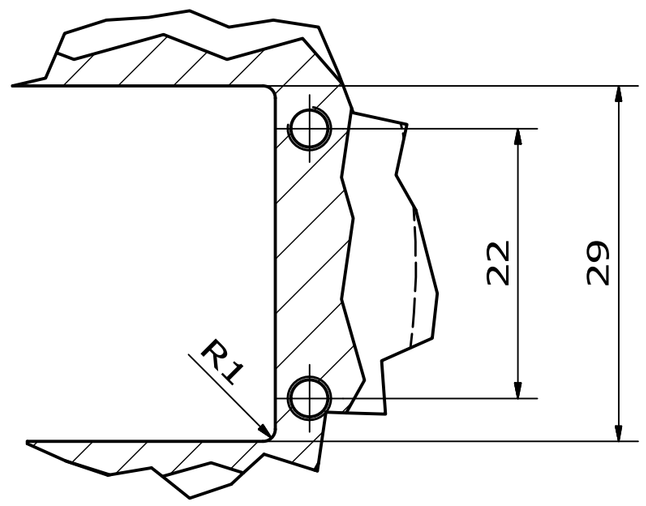


A-A (1 : 1)

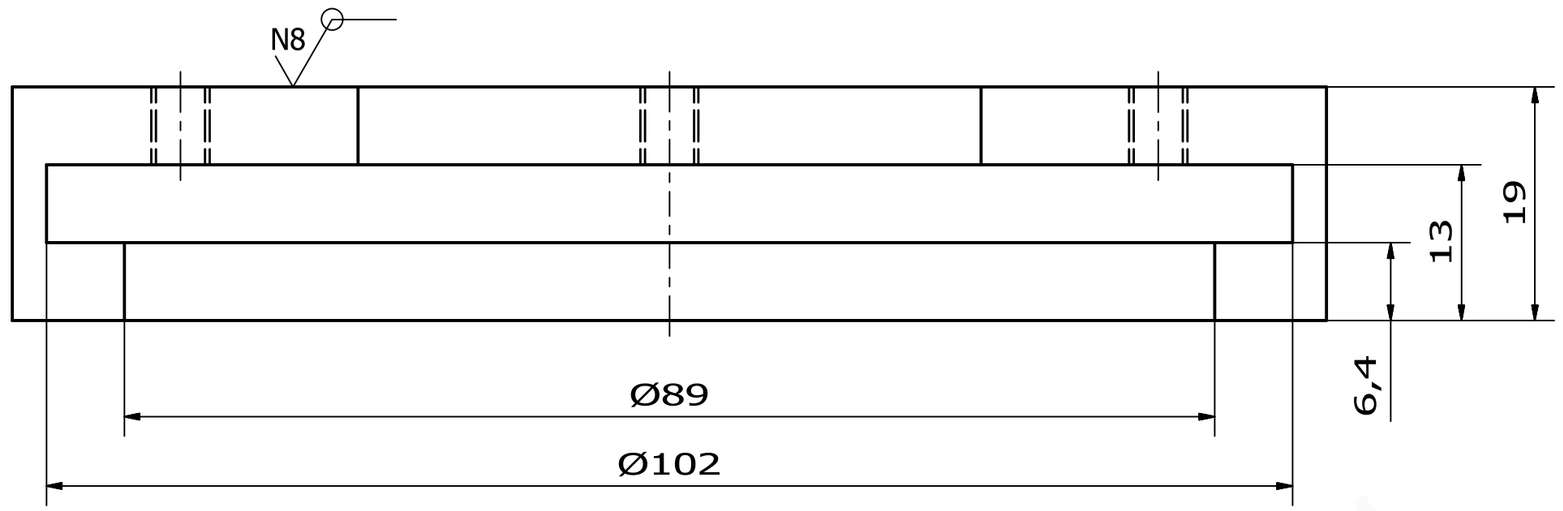


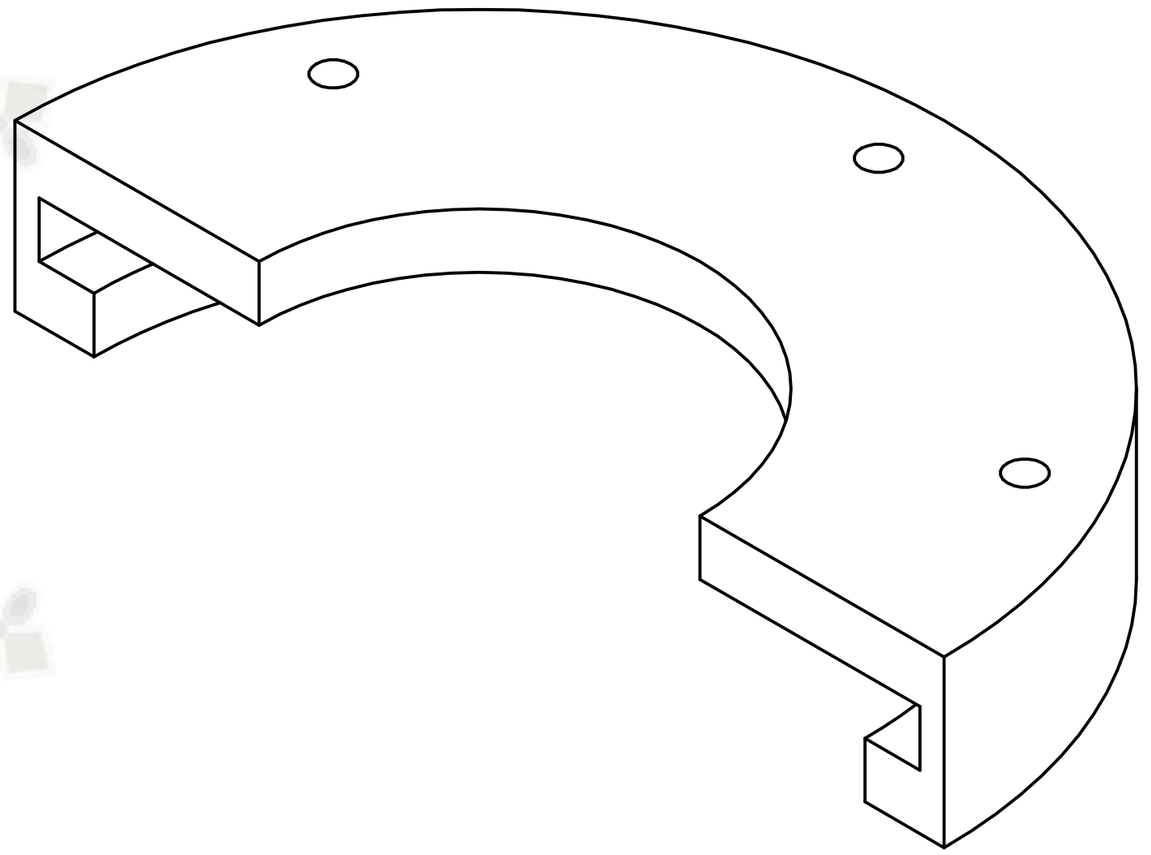


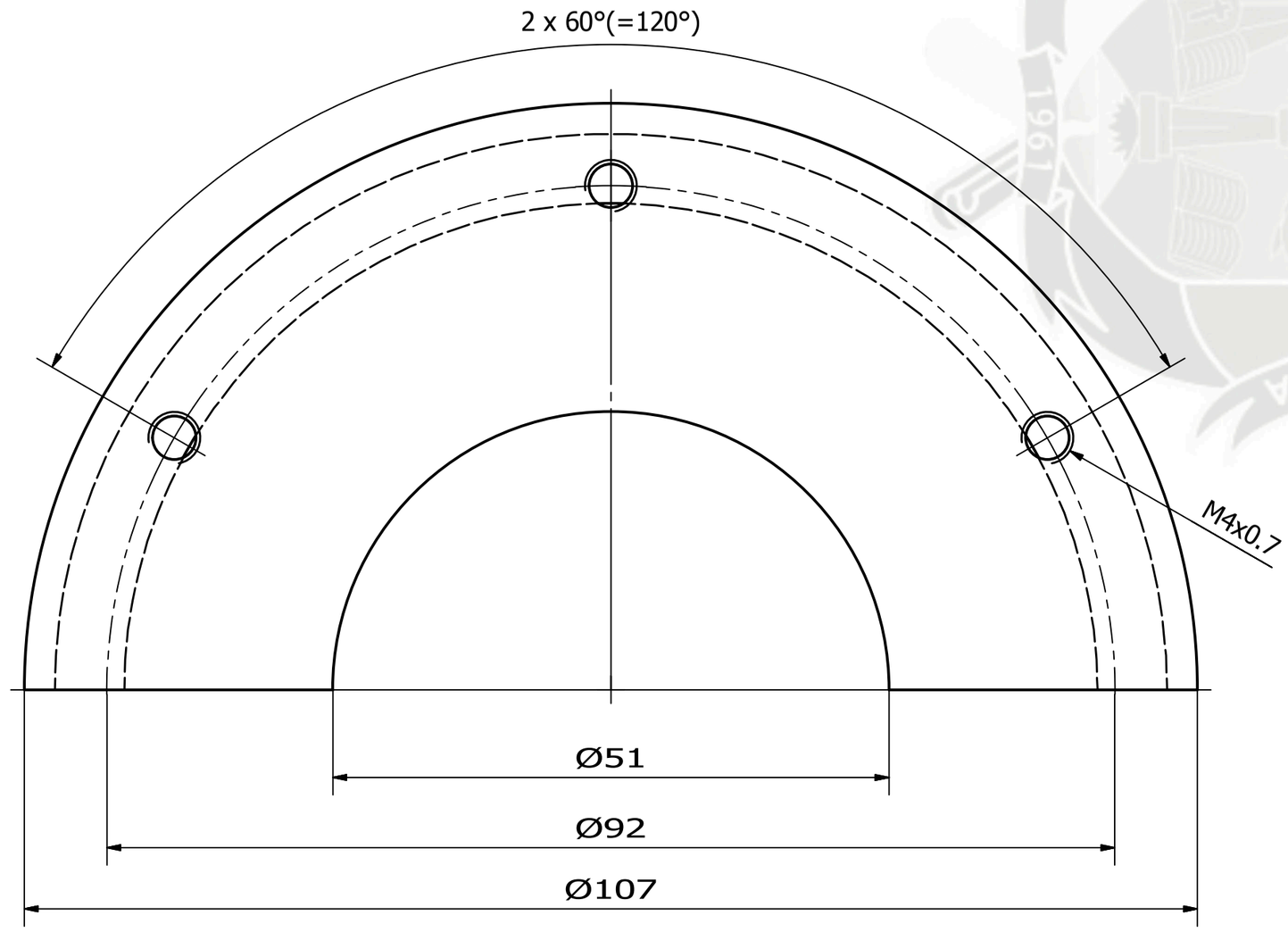
C (2 : 1)



Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 1 / 18

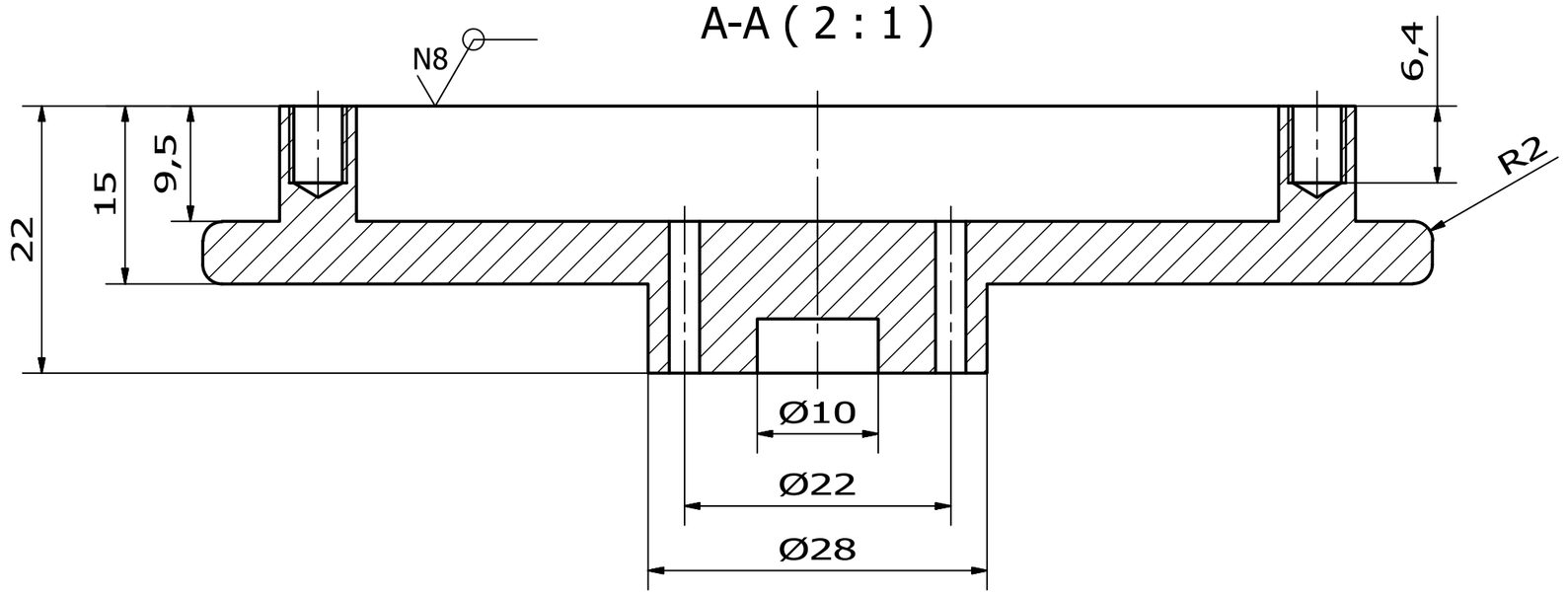


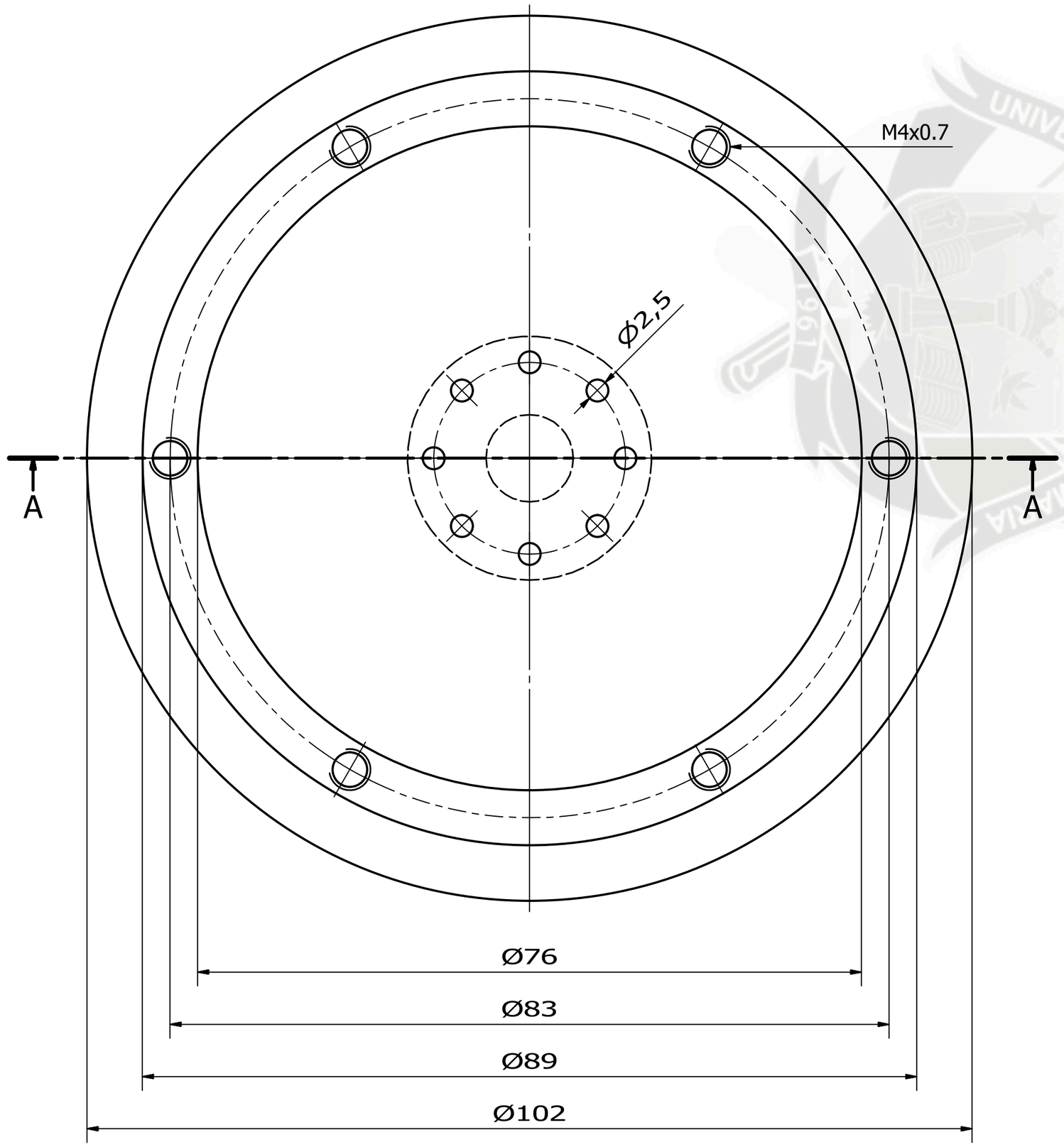


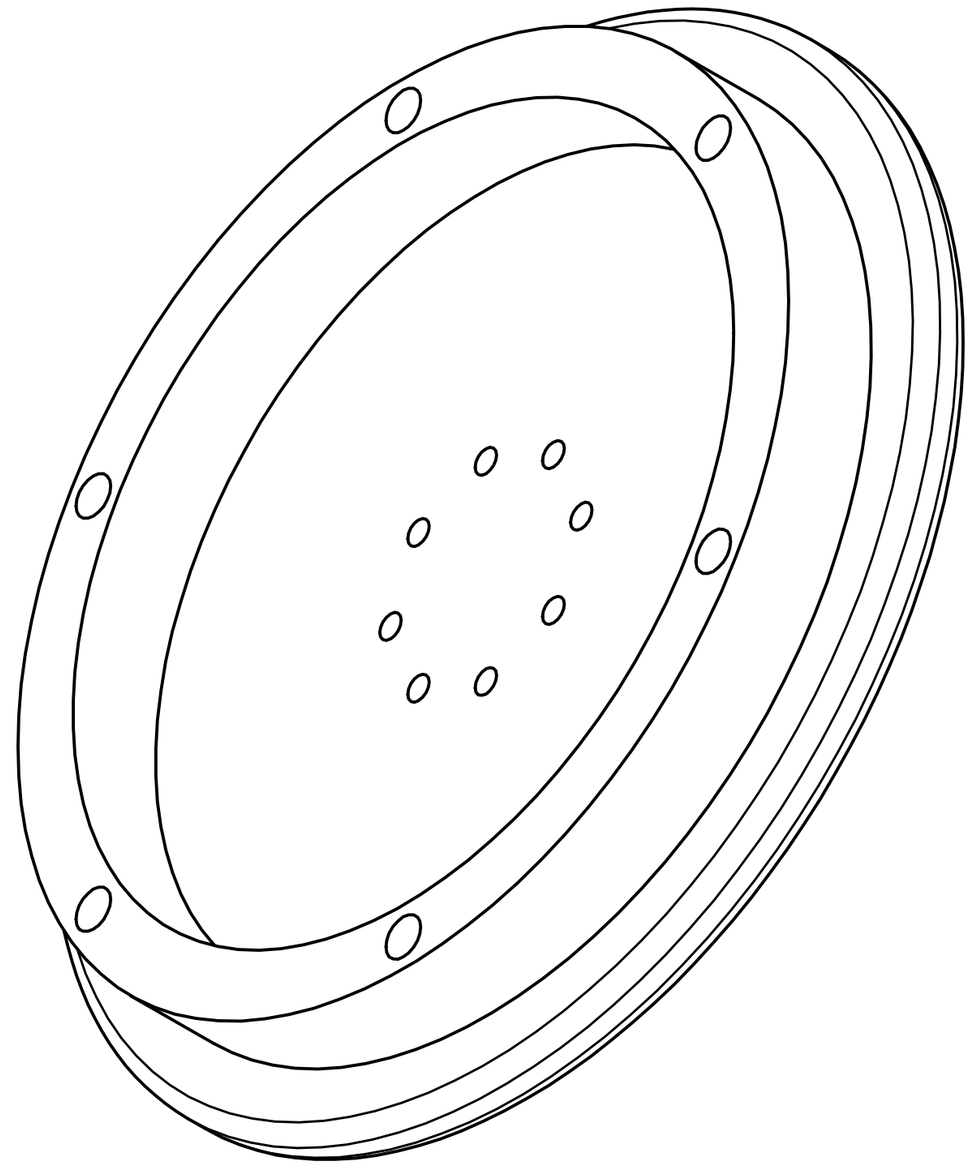


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 2 / 18

A-A (2 : 1)

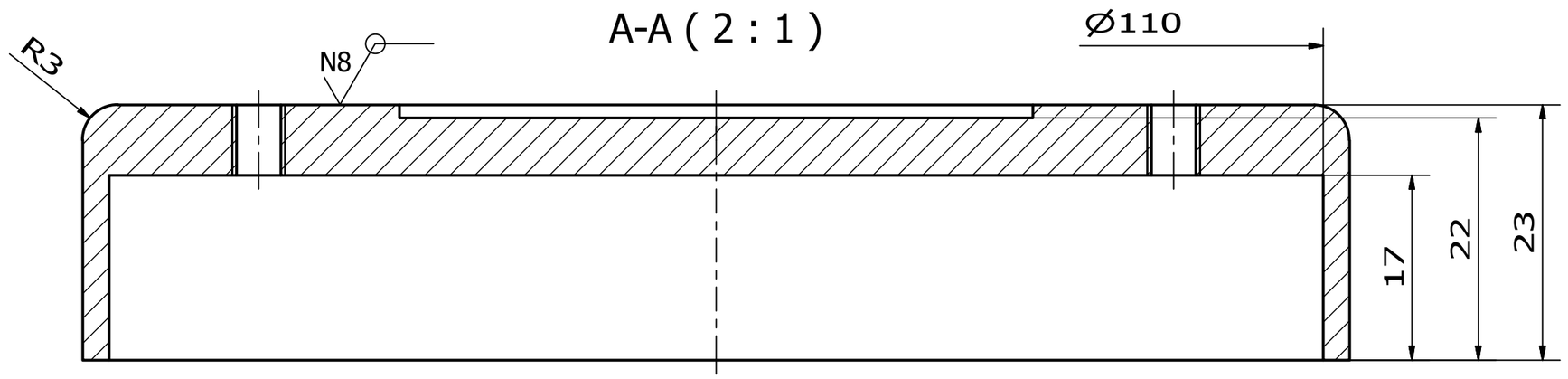


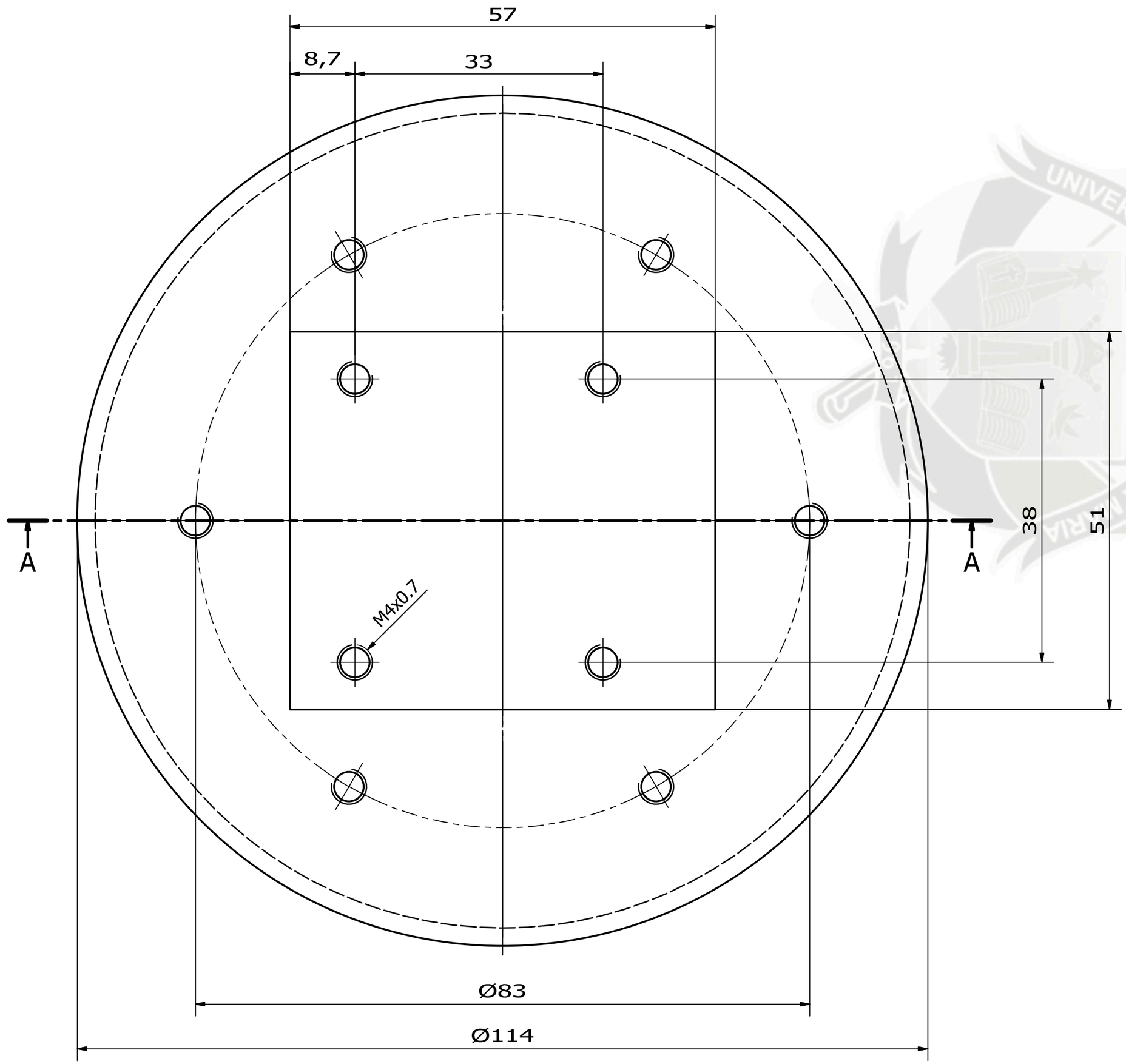


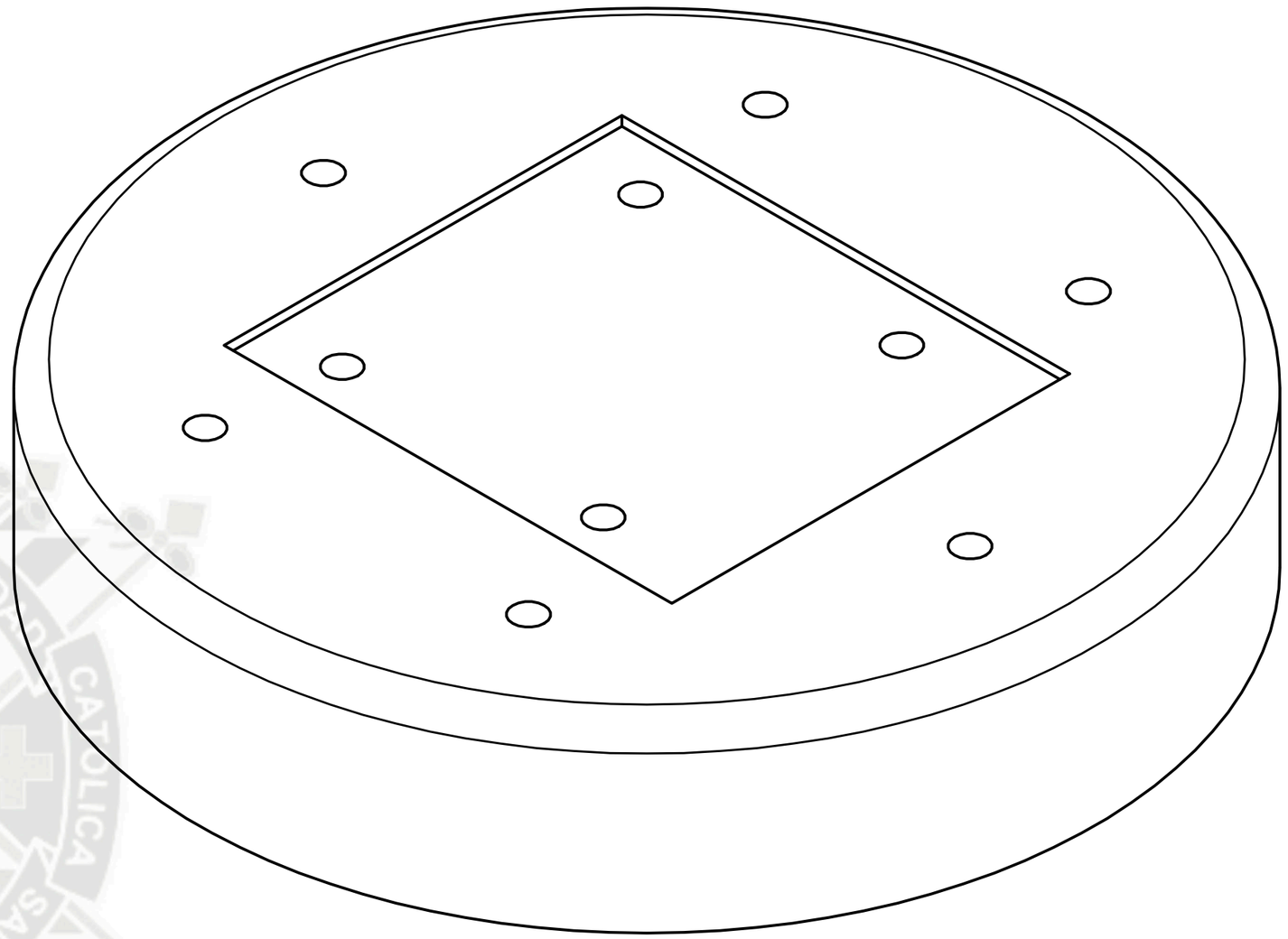


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 3 / 18

A-A (2 : 1)

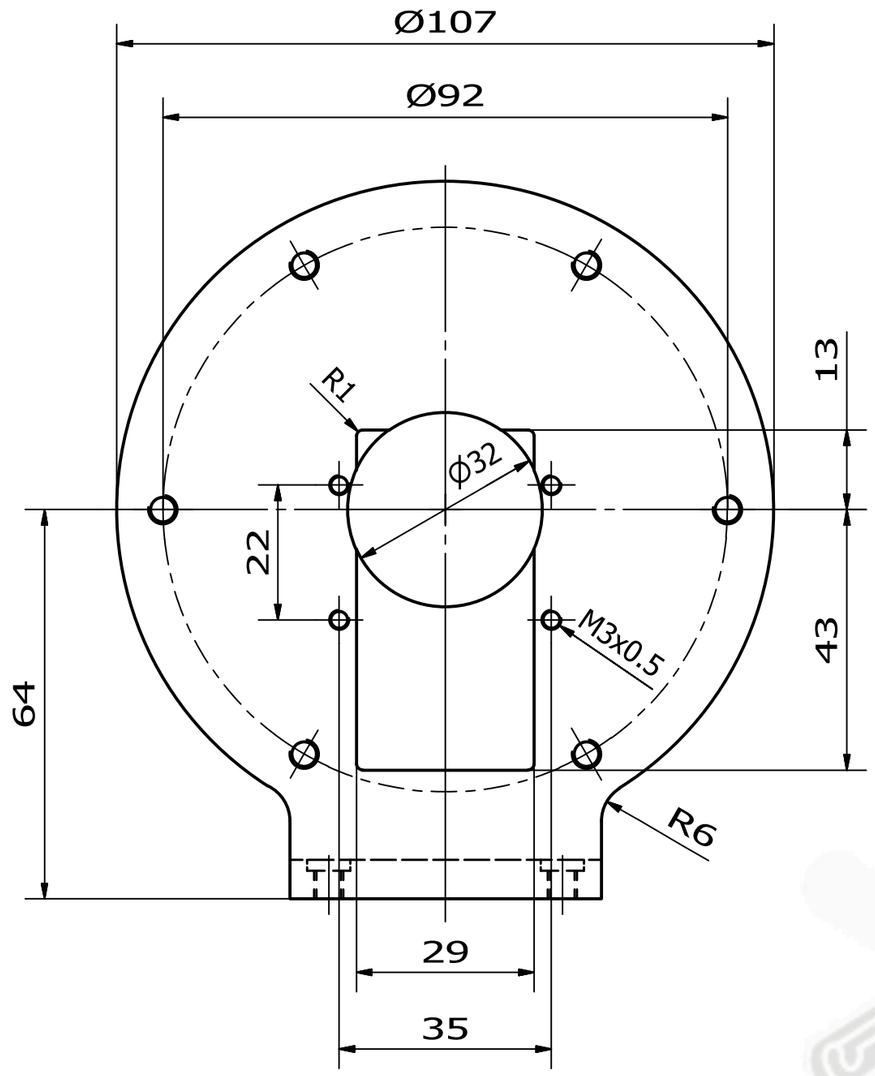


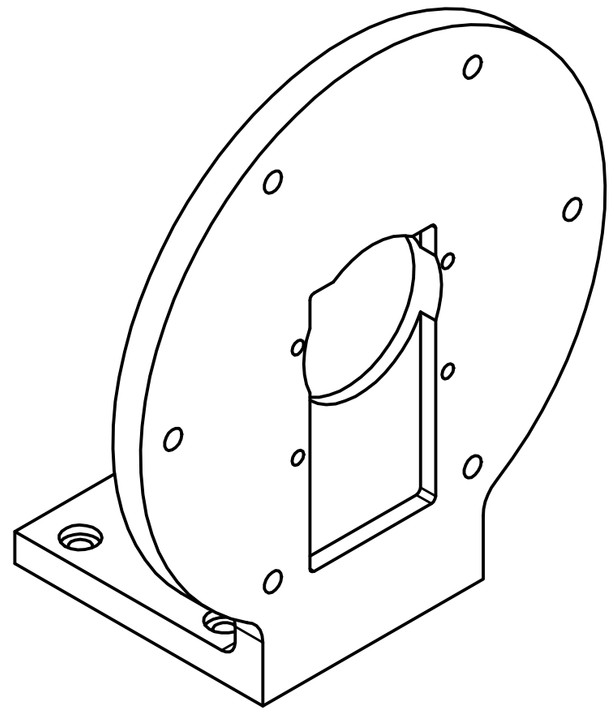


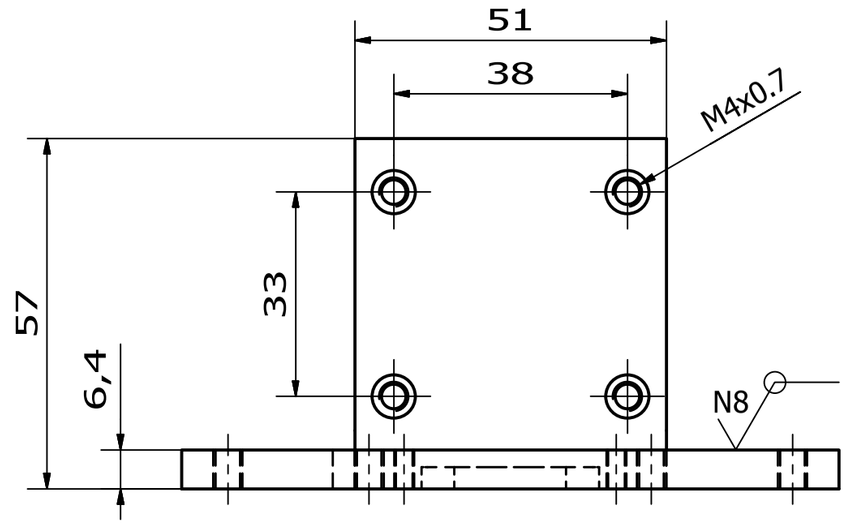


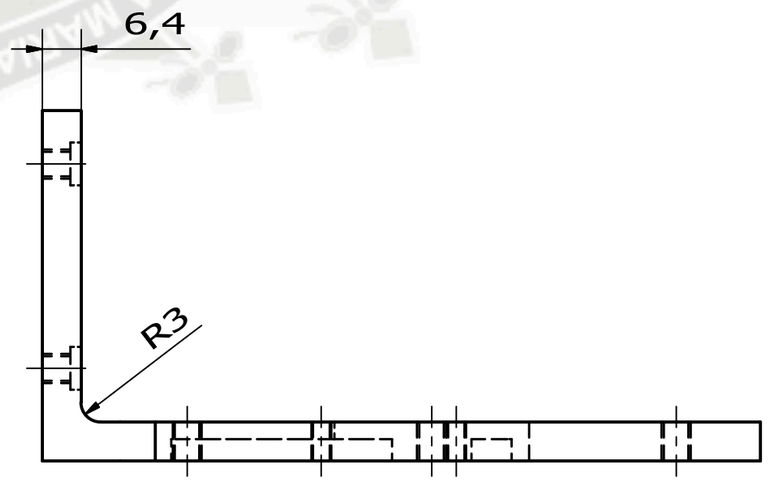
Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	

Universidad Católica de Santa María	Robotic Arm		
		Edition: 1	Sheet: 4 / 18

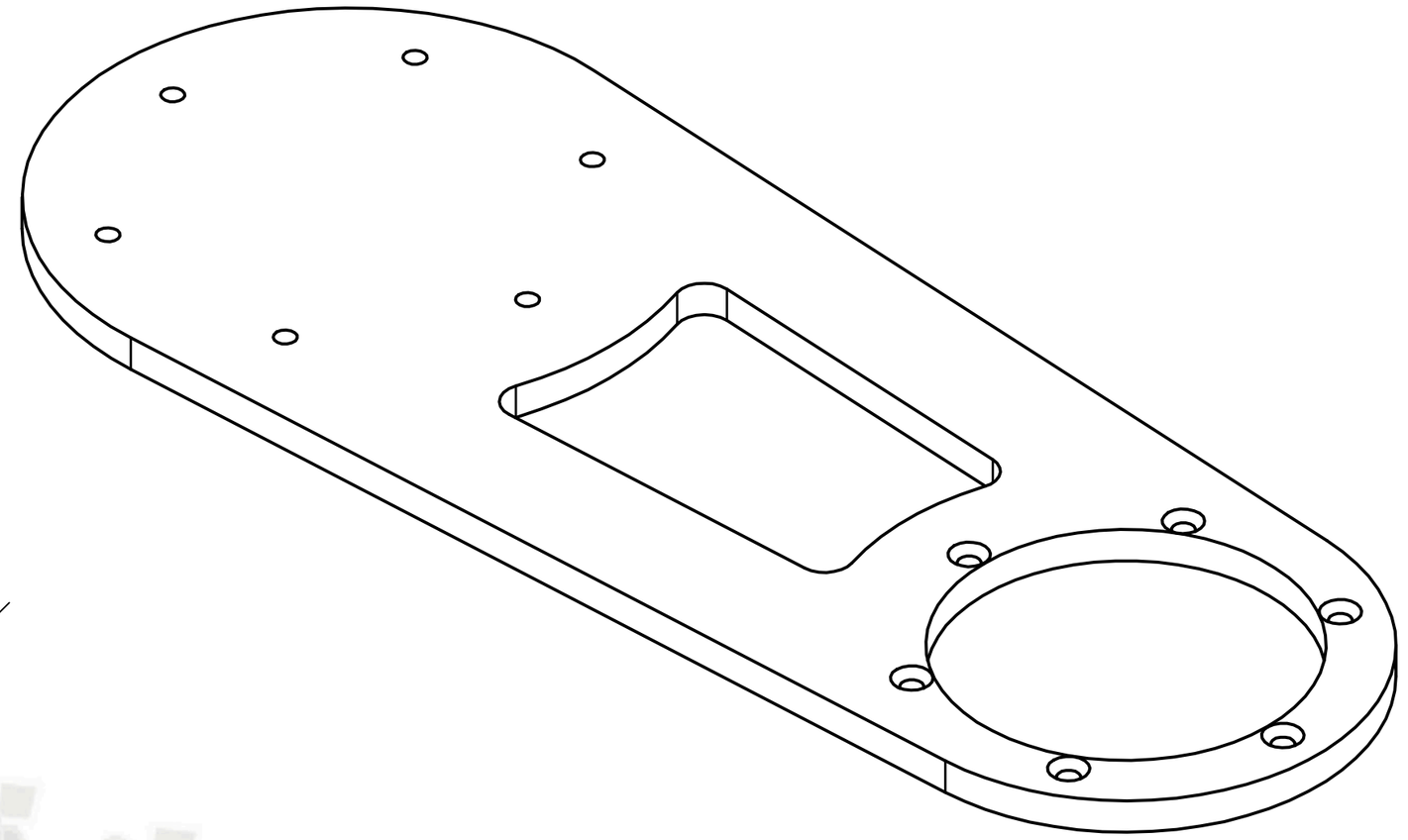


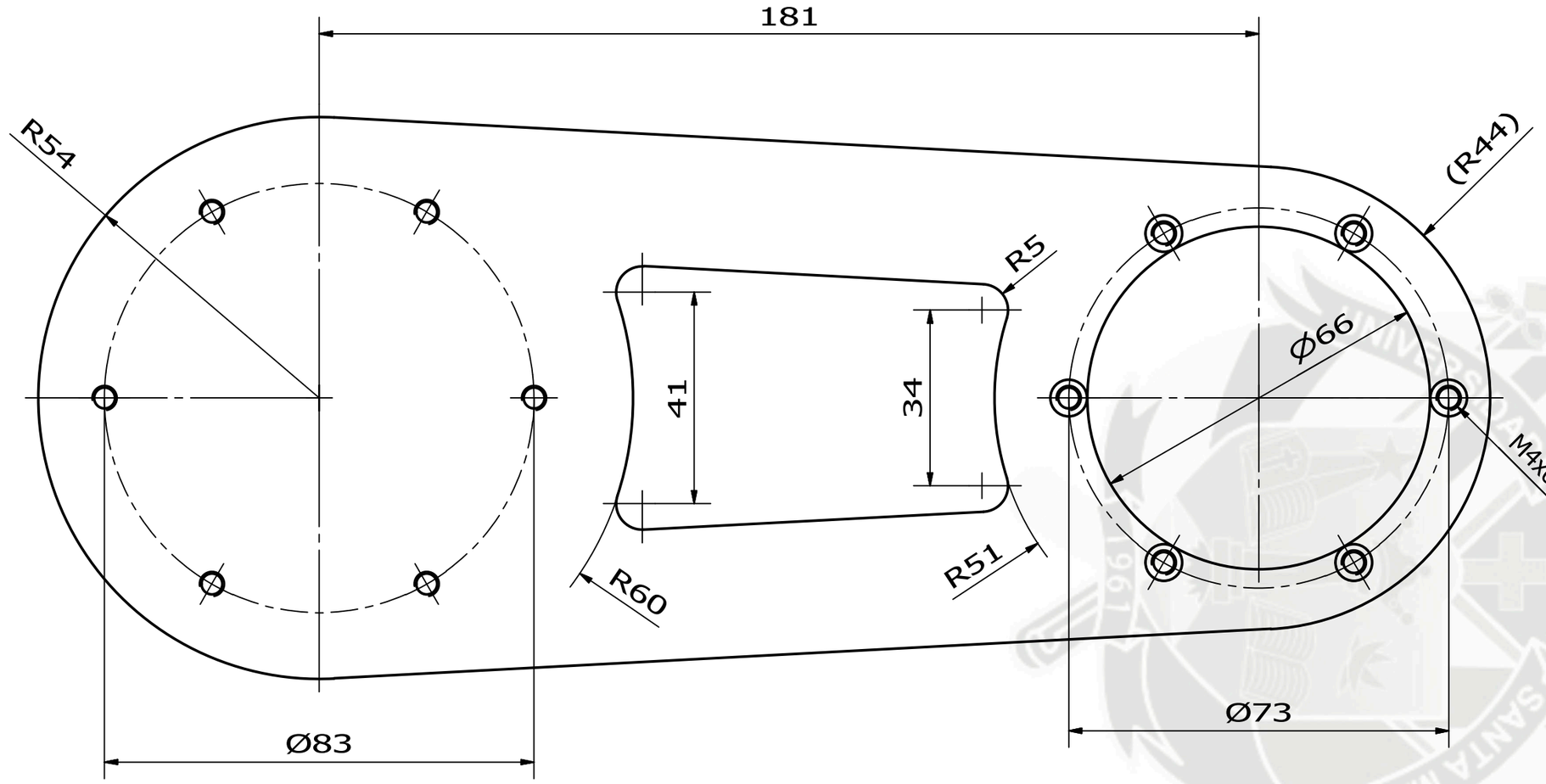


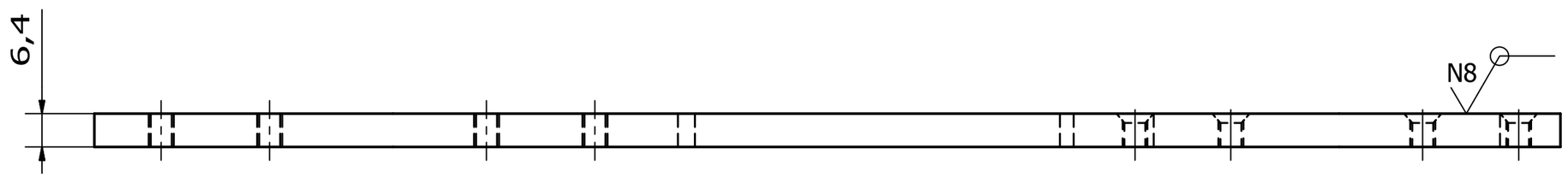




Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 5 / 18

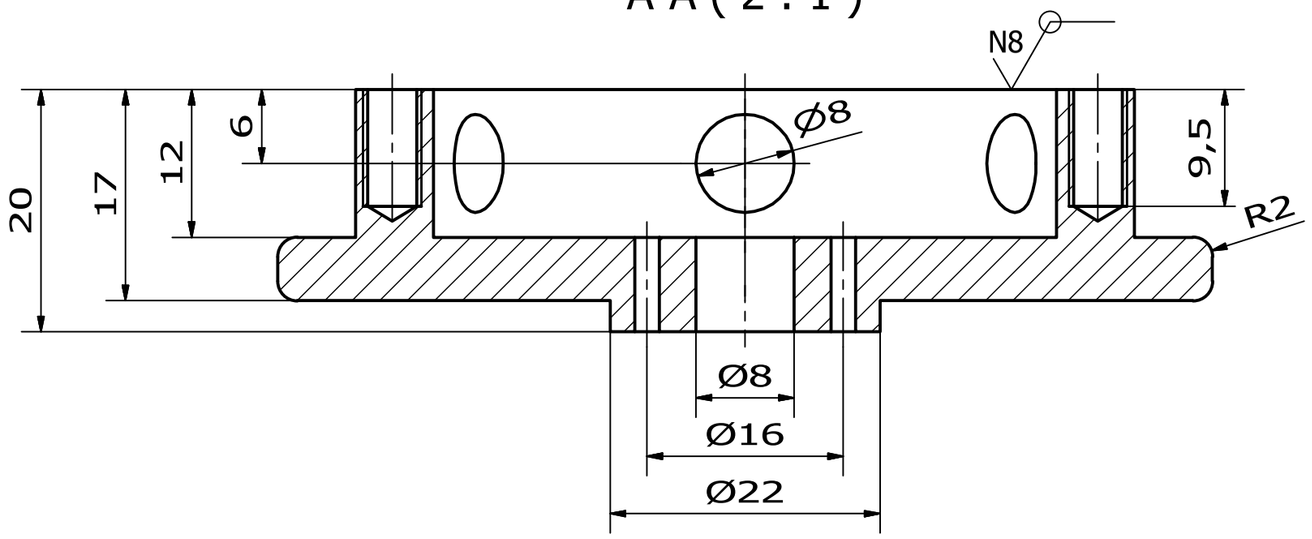


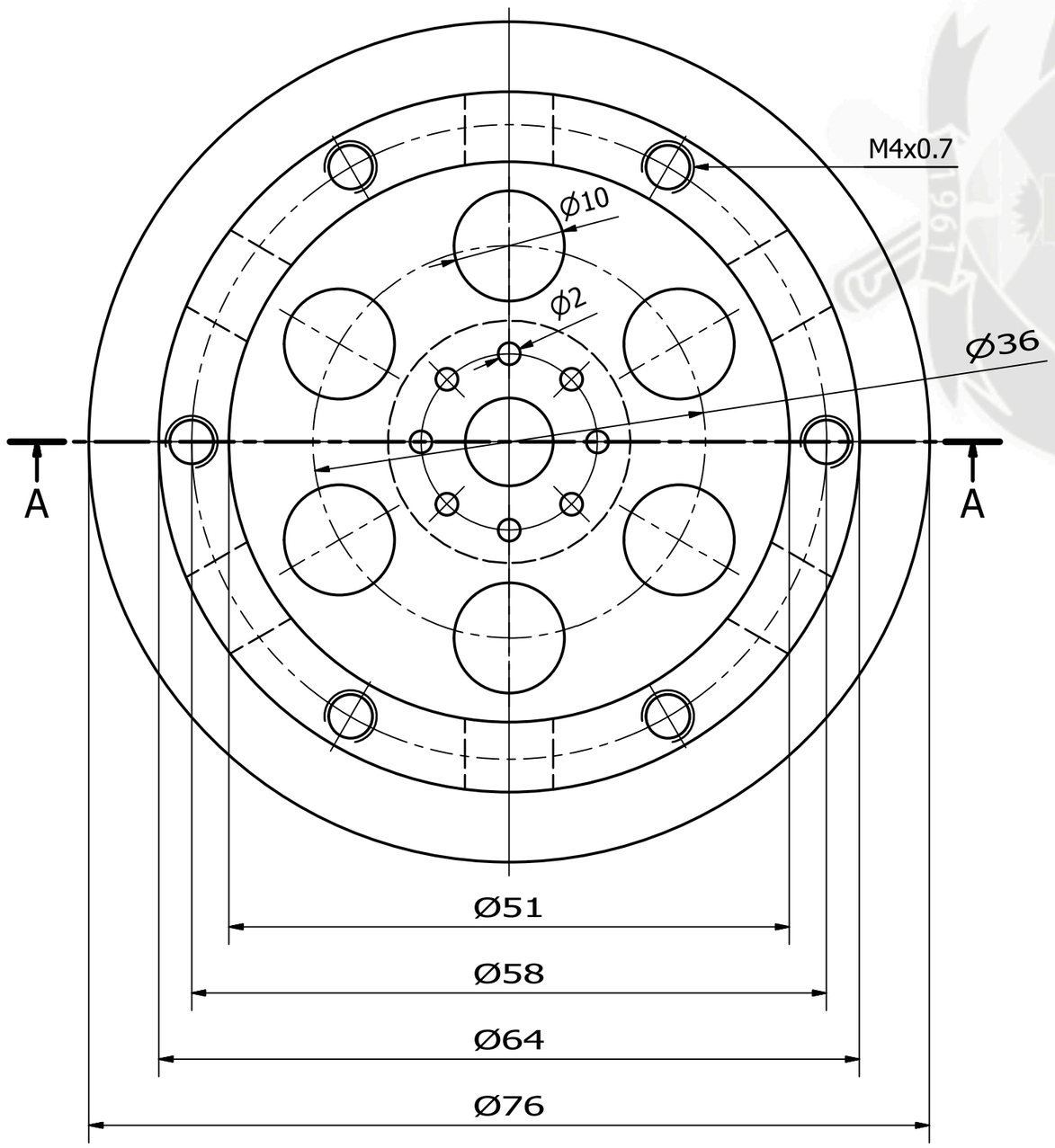


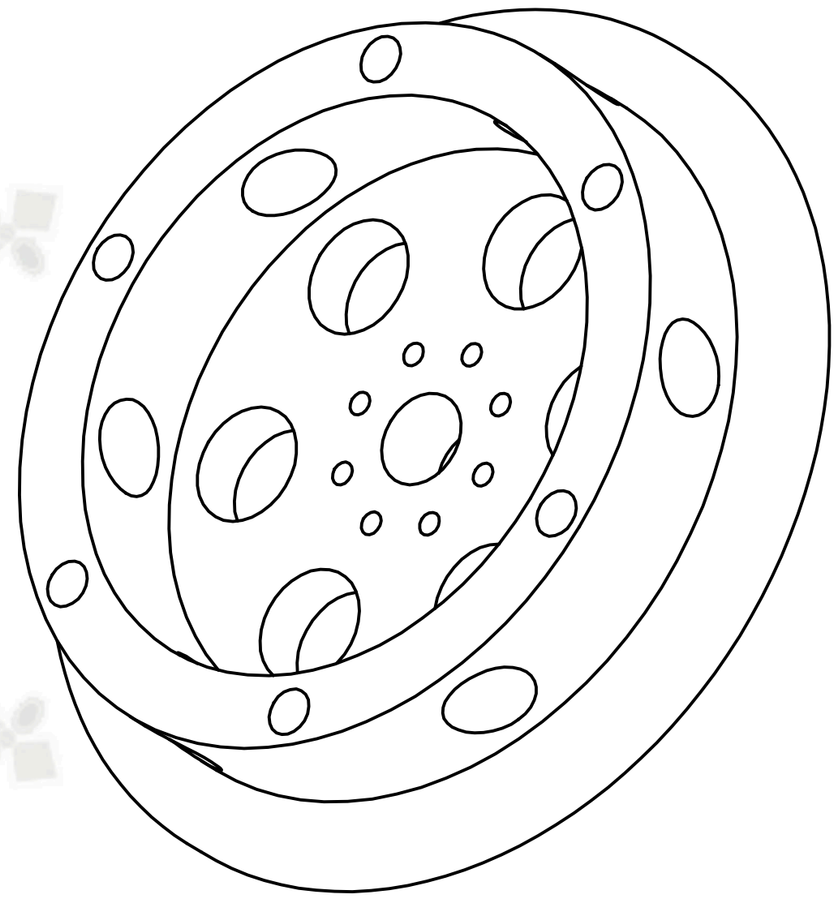


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 6 / 18

A-A (2 : 1)

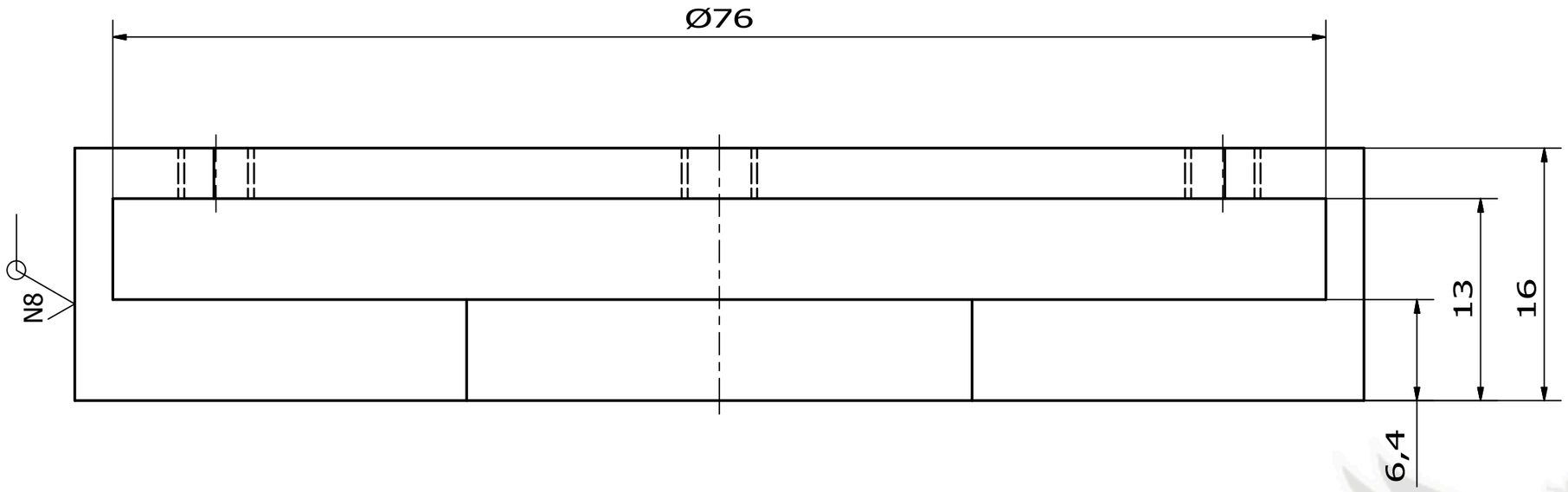


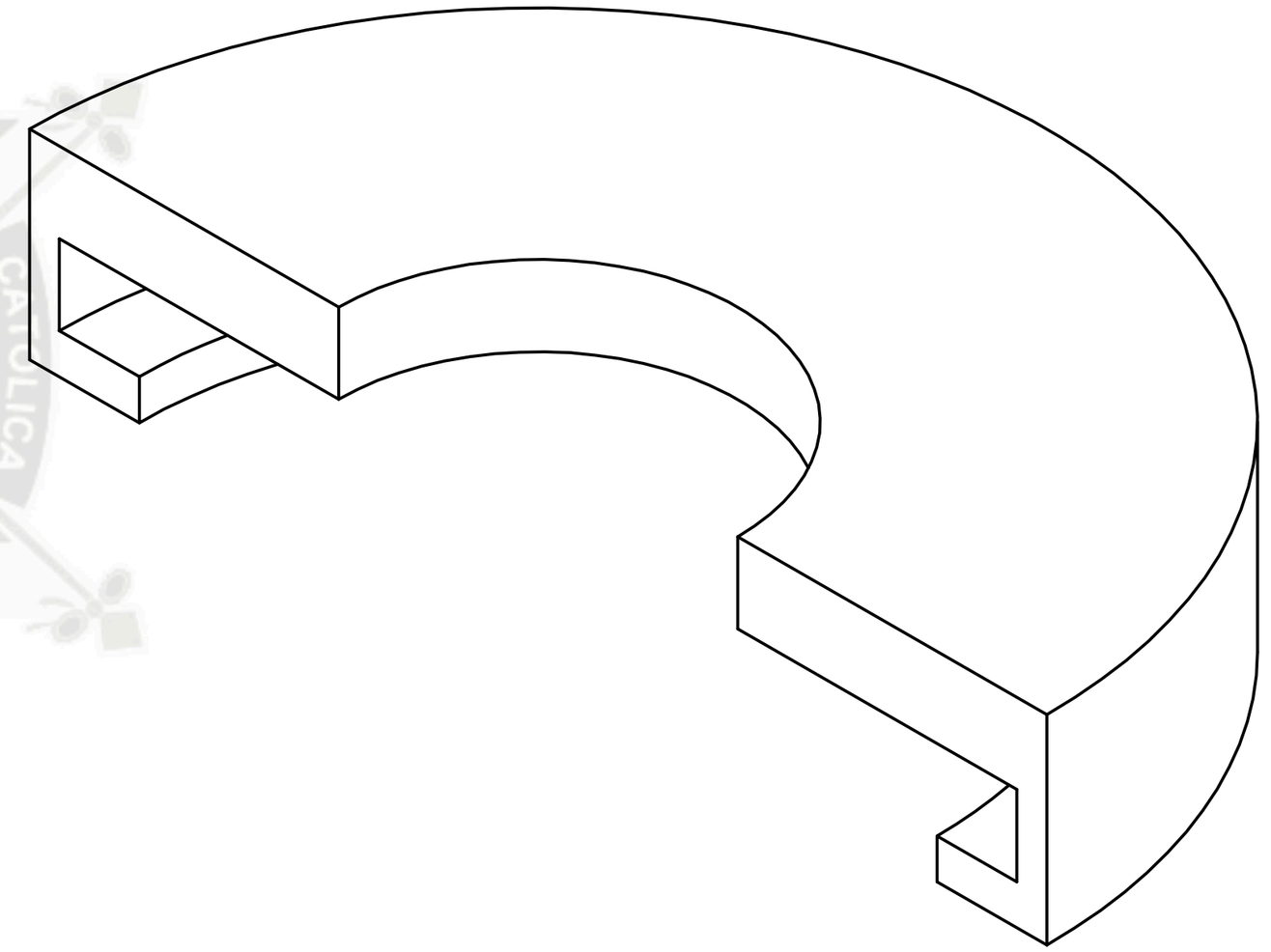


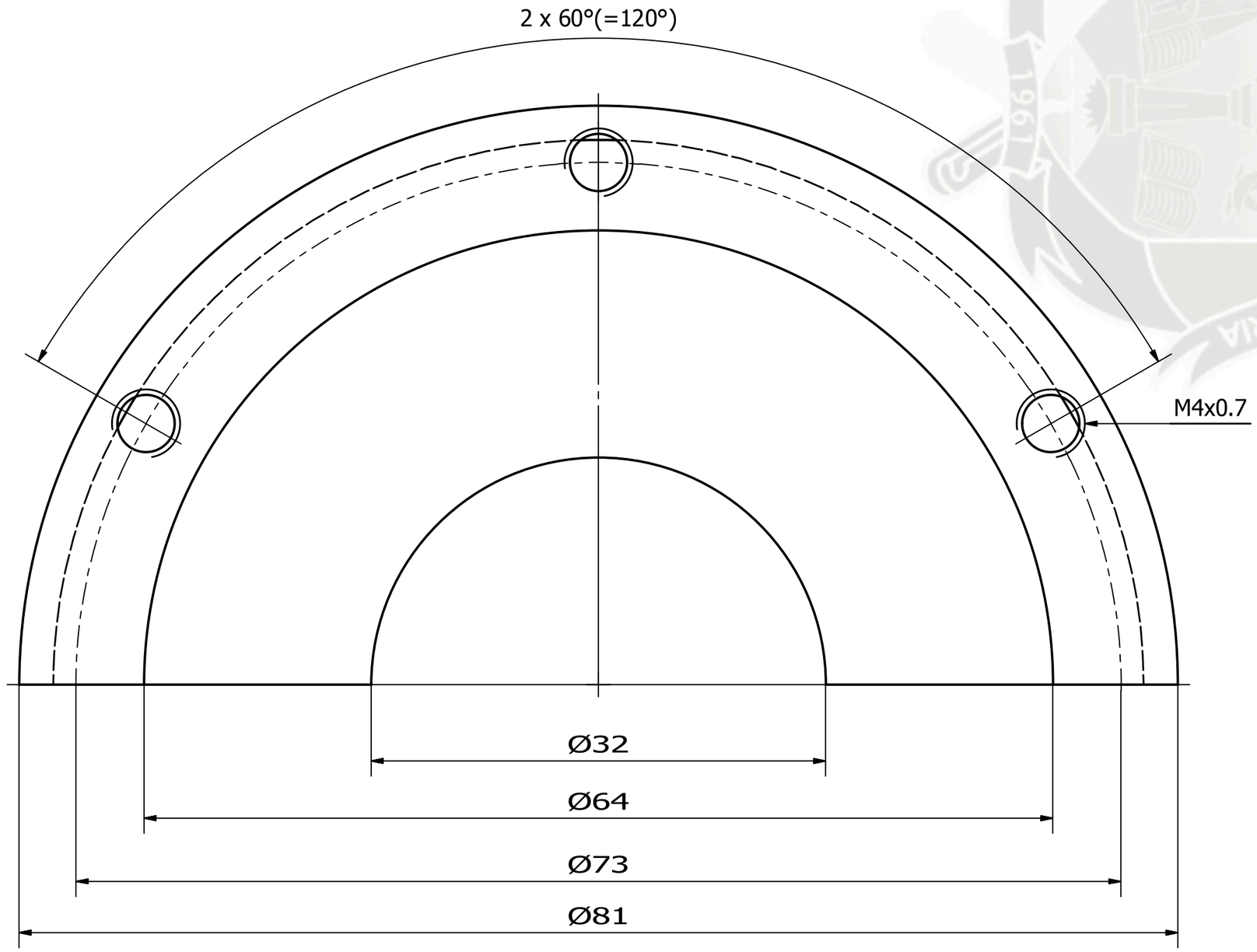


Designed by	Checked by	Approved by	Date	Date	
Carlofranco Ruiz D.		Nylon 6		17/03/2014	

Universidad Católica de Santa María	Robotic Arm		
		Edition 1	Sheet 7 / 18

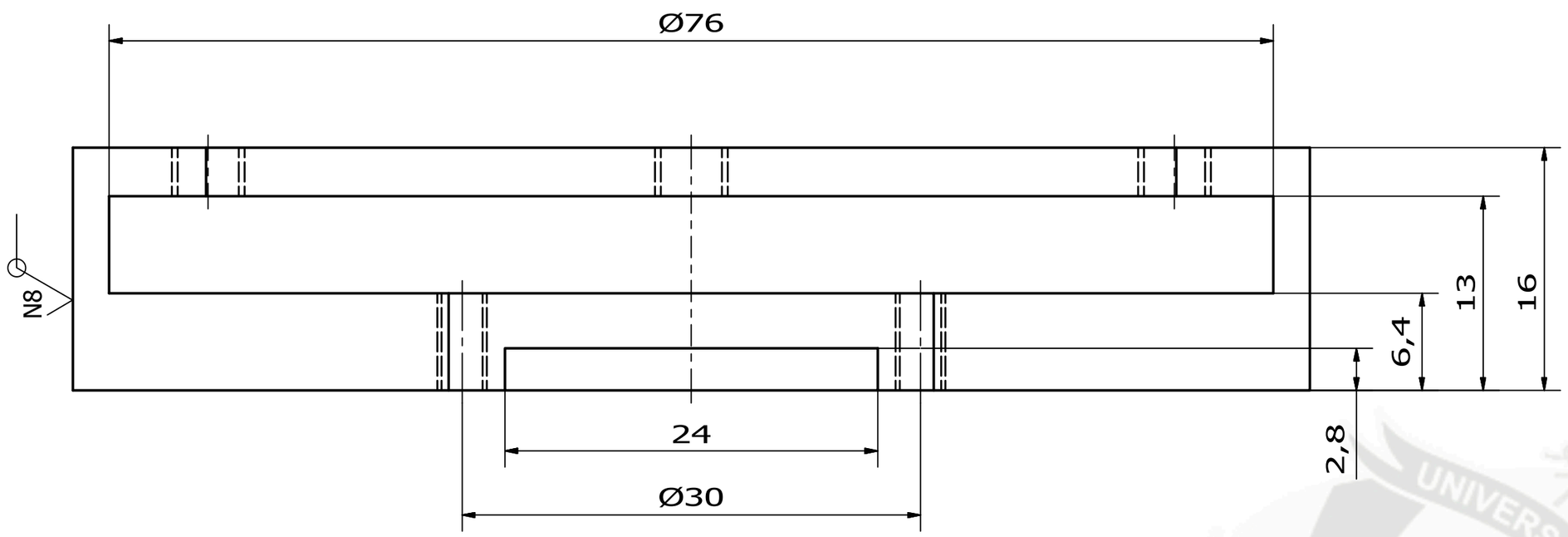


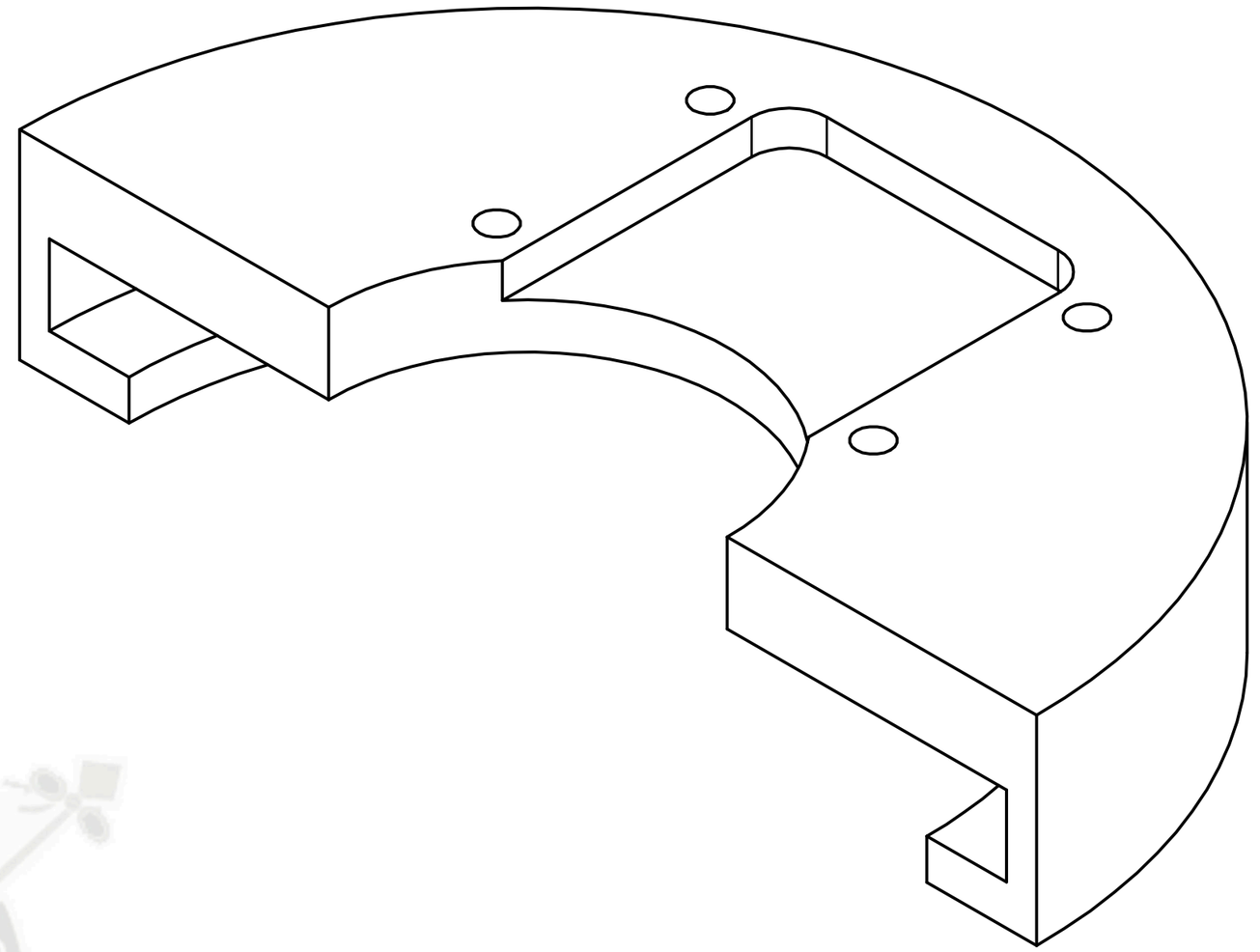


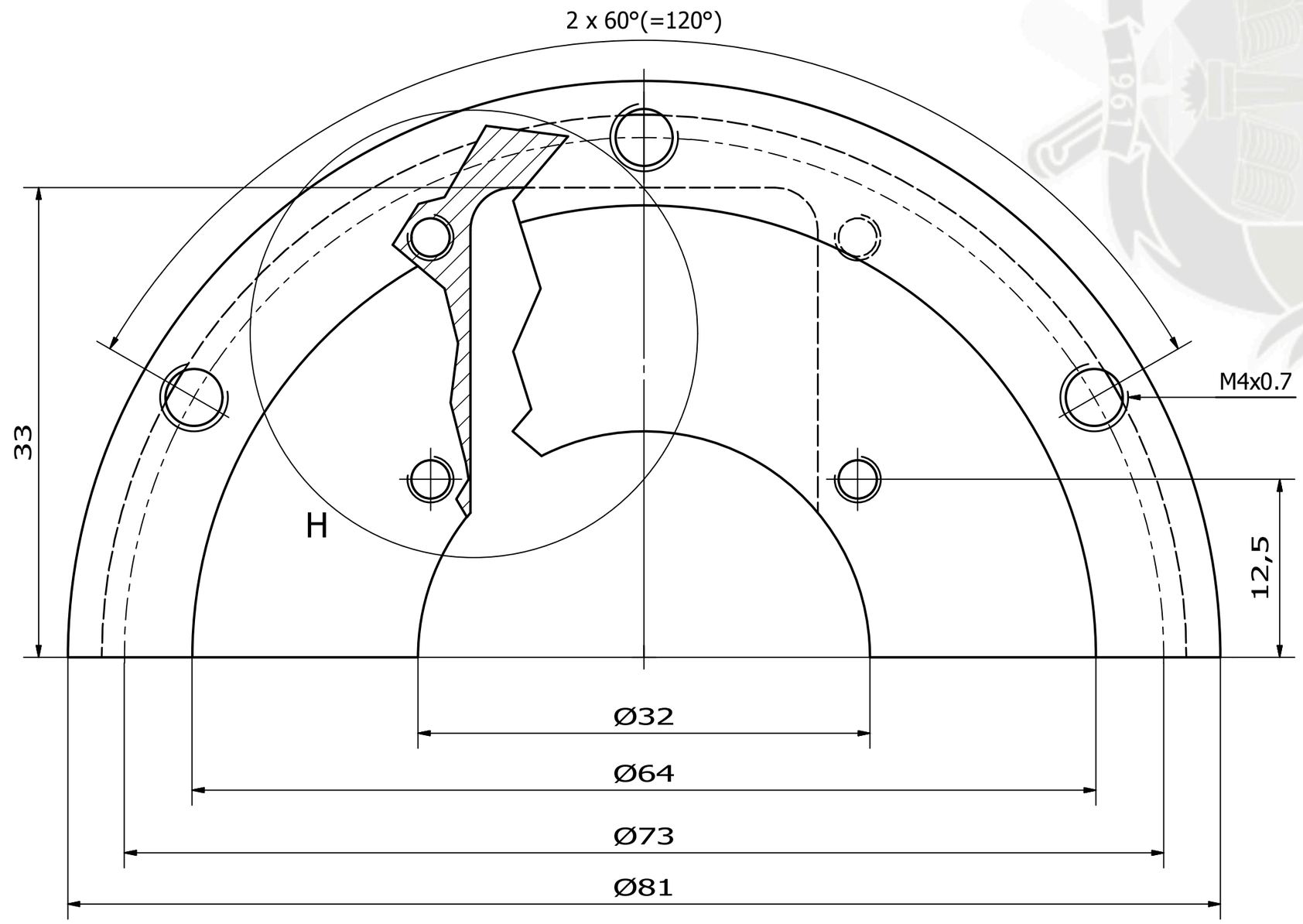


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	

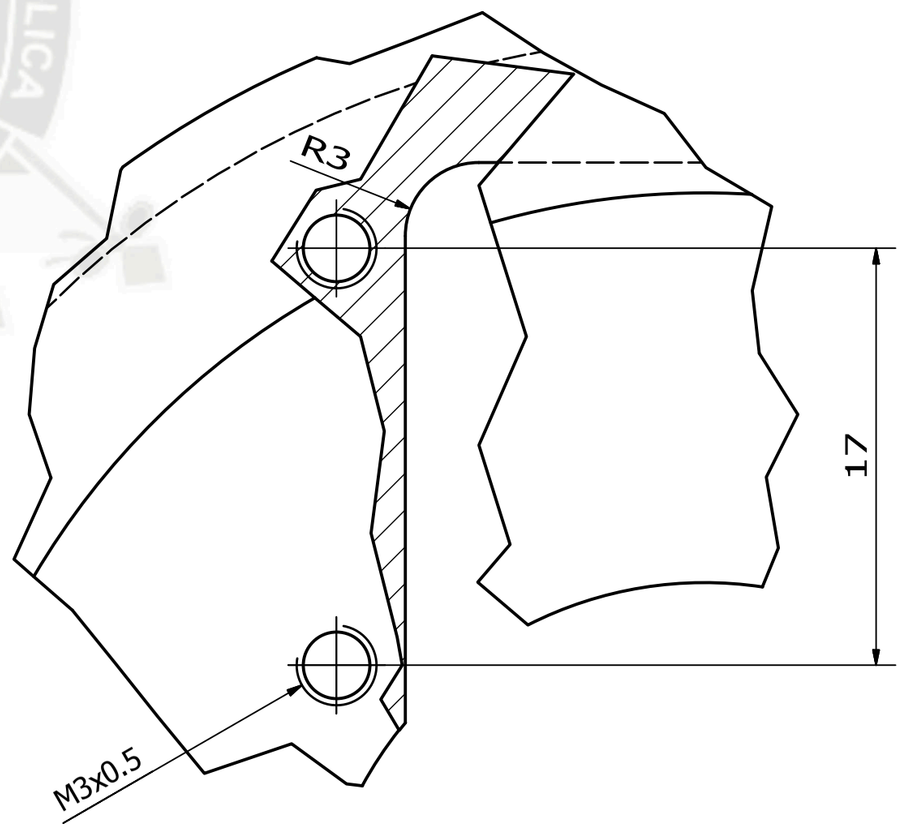
Universidad Católica de Santa María	Robotic Arm	
	Edition: 1	Sheet: 8 / 18



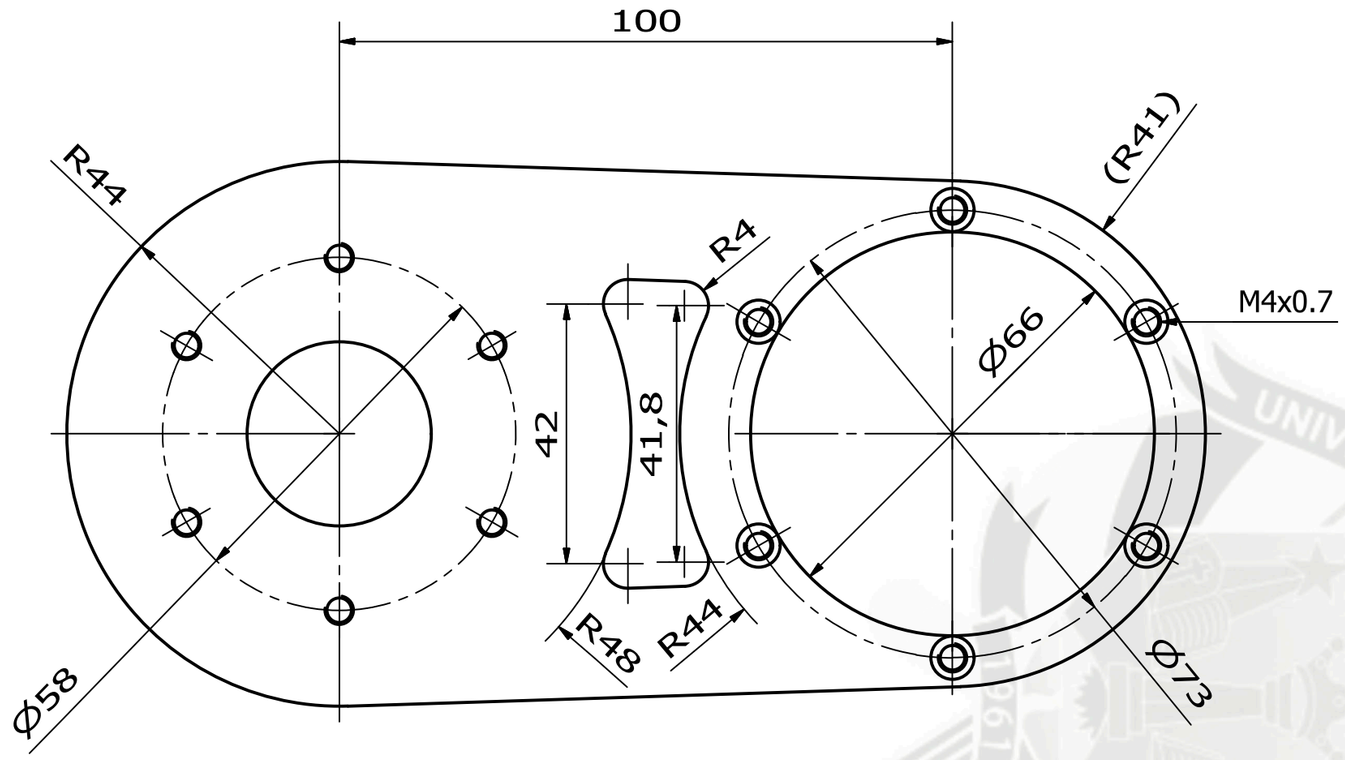


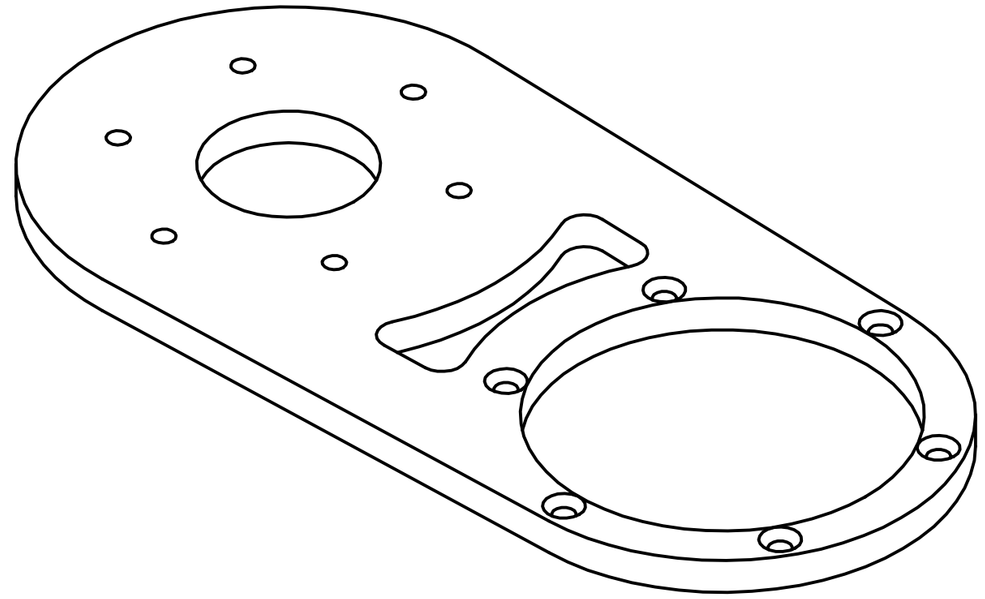


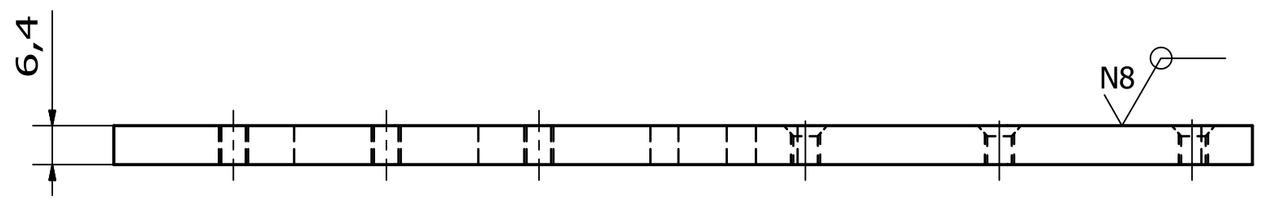
H (4 : 1)



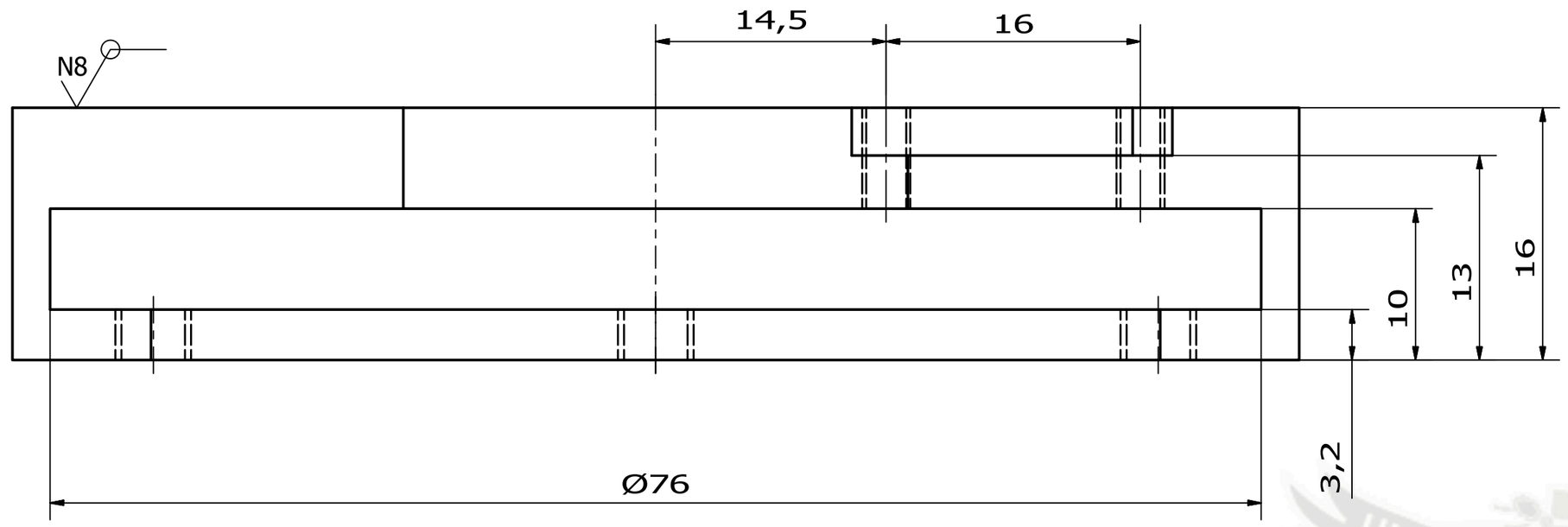
Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 9 / 18

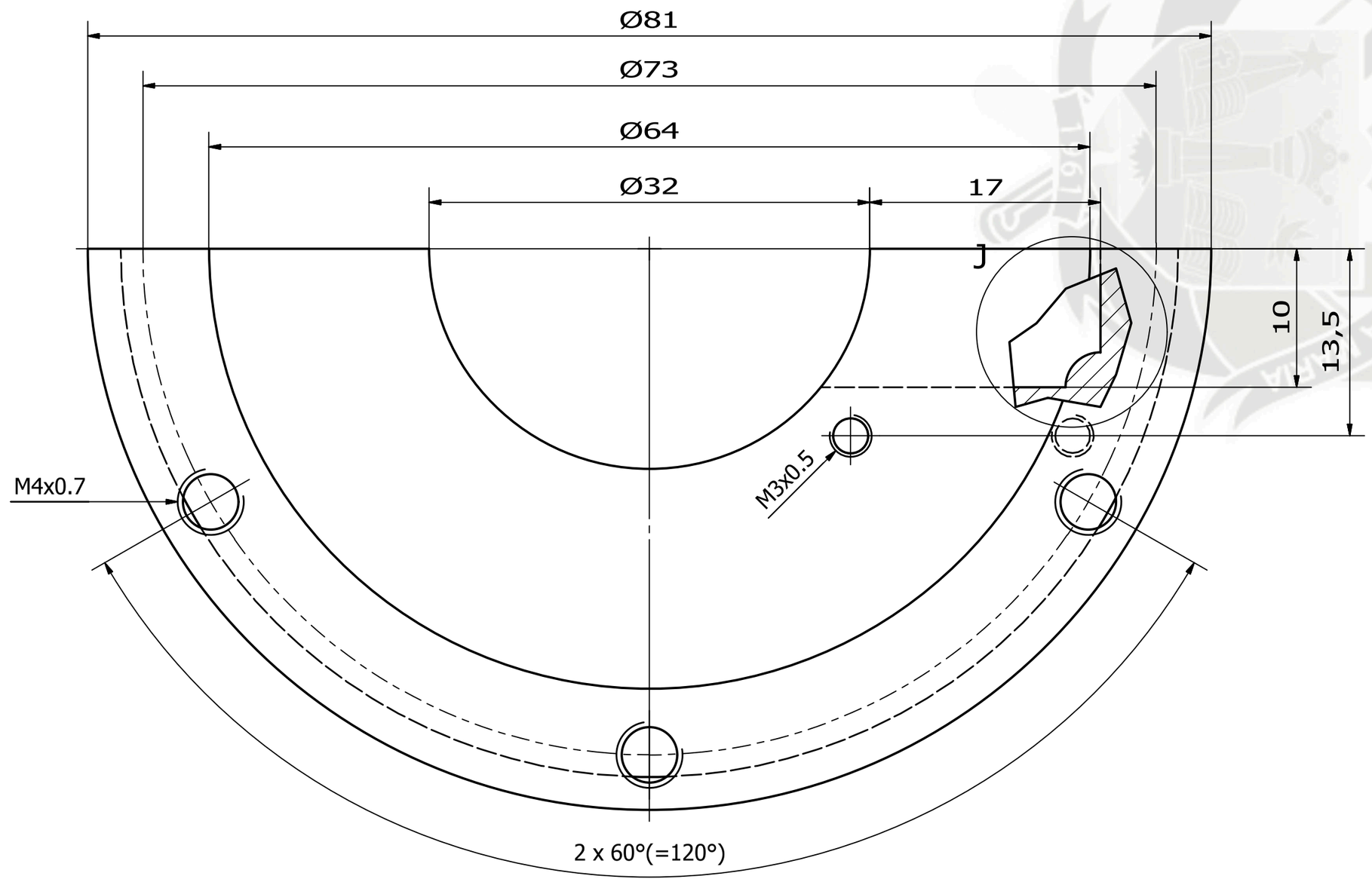


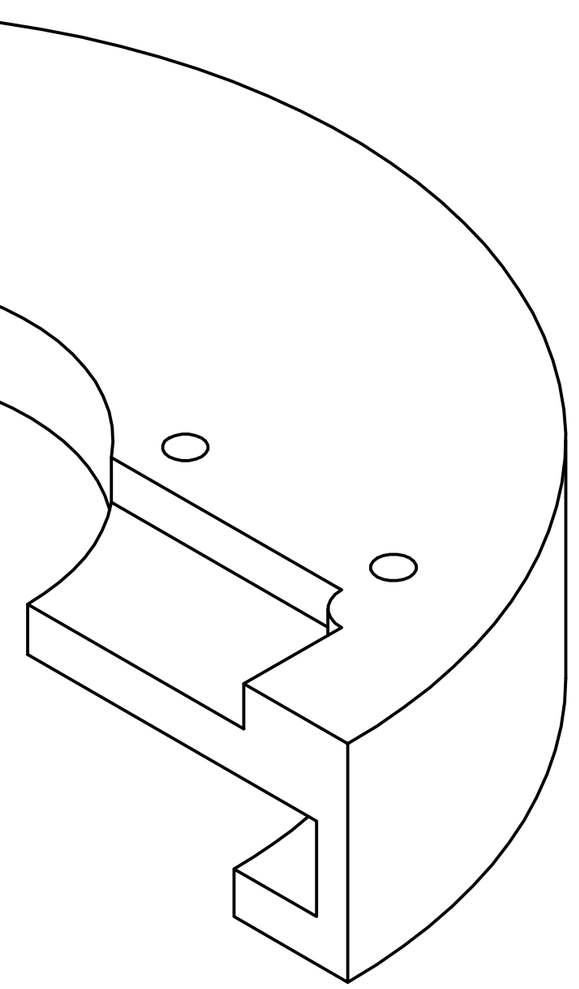




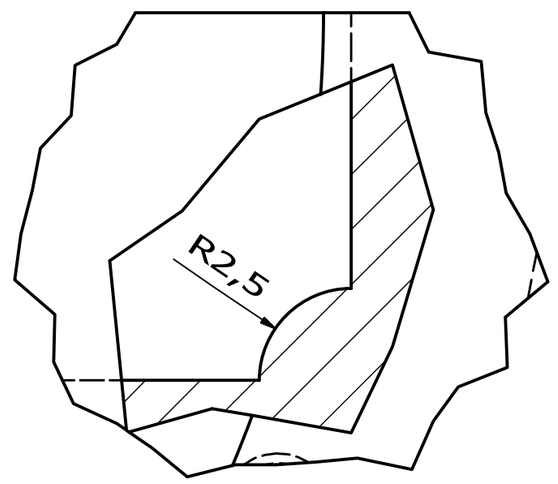
Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 10 / 18



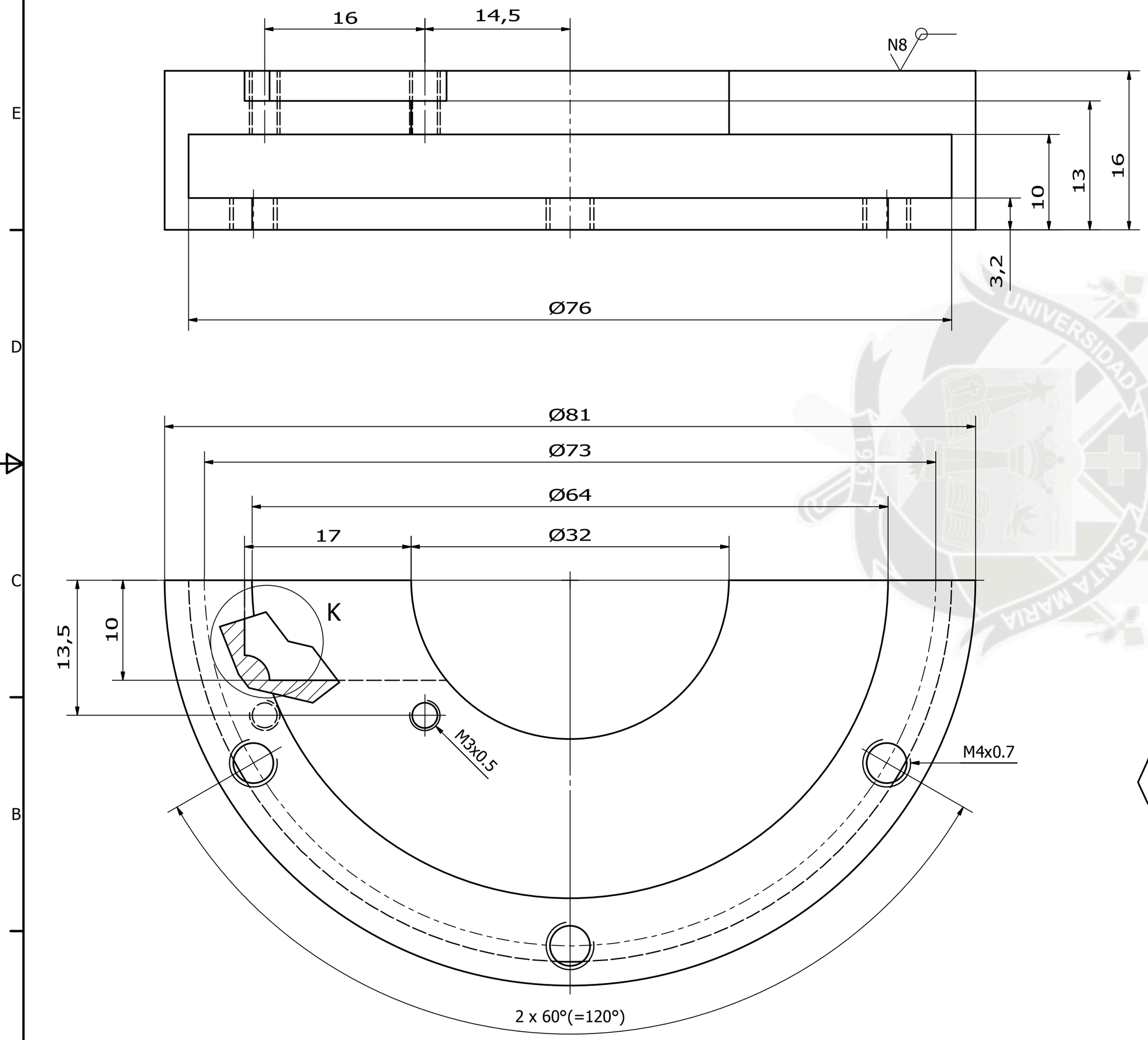


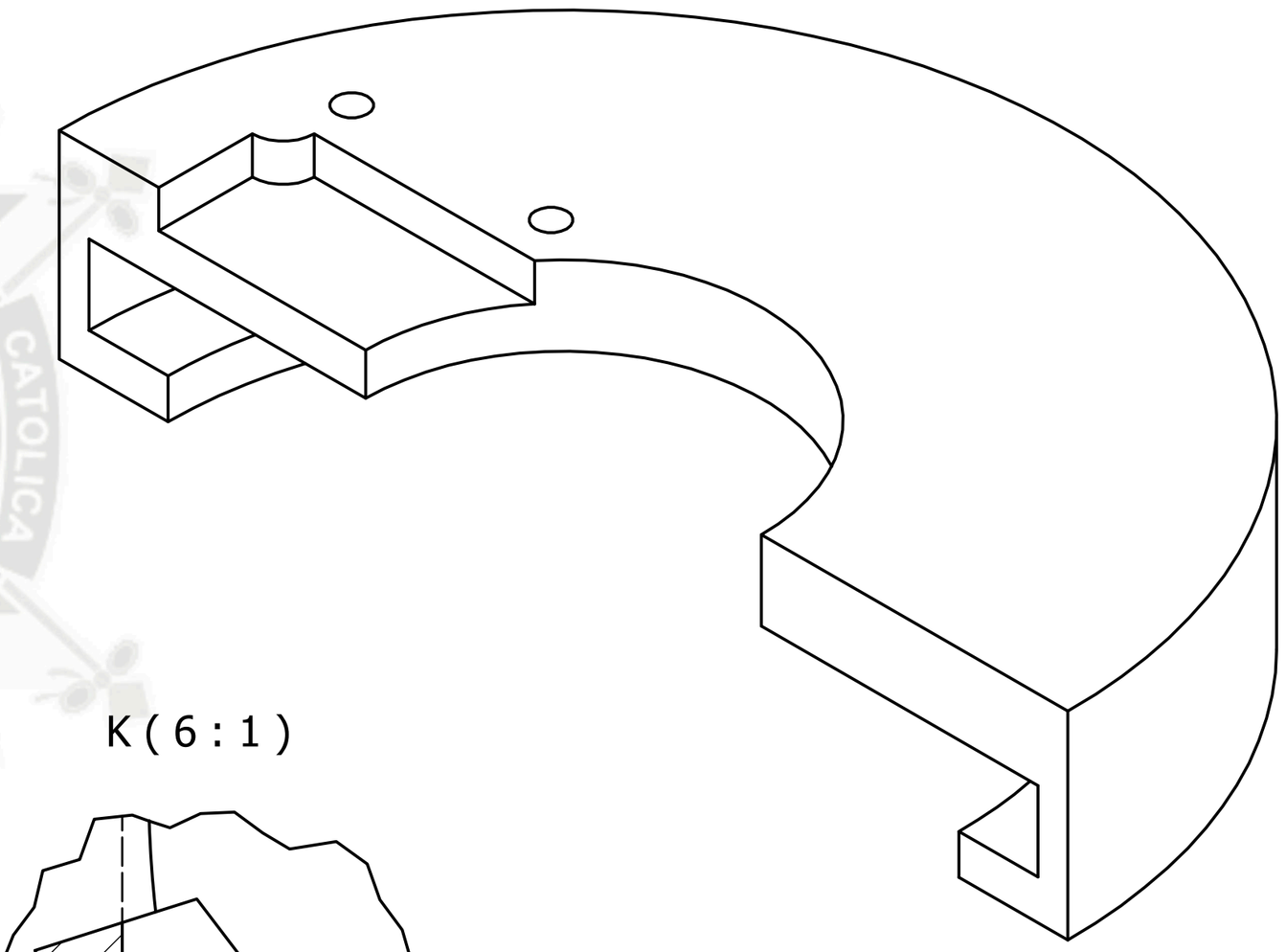


J (6 : 1)

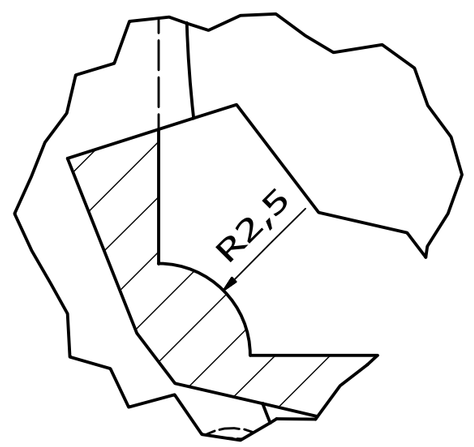


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 11 / 18

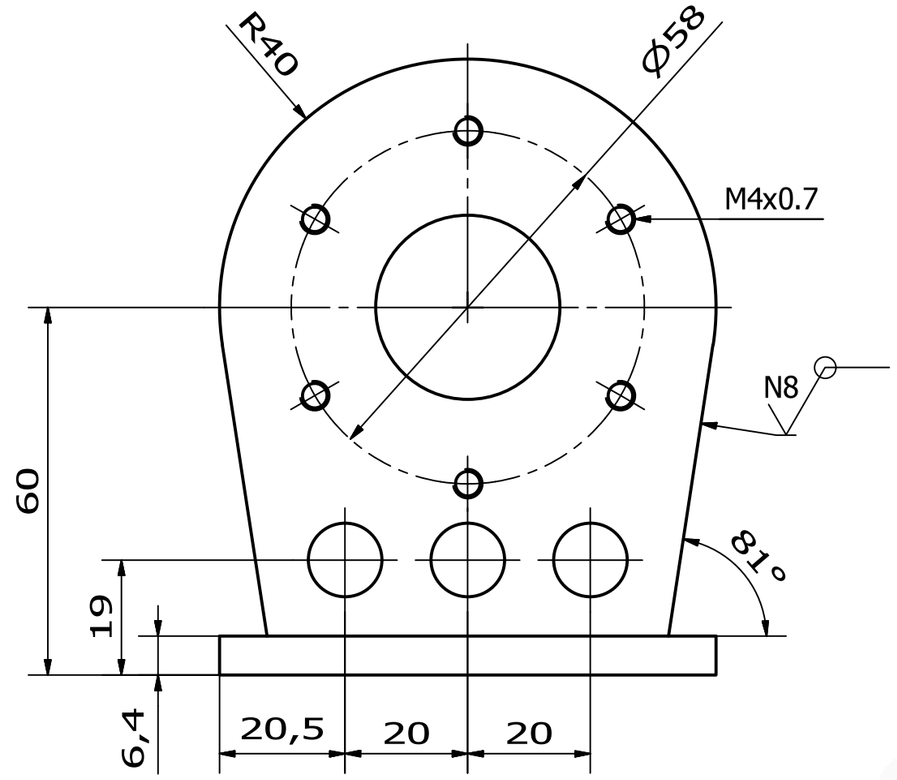


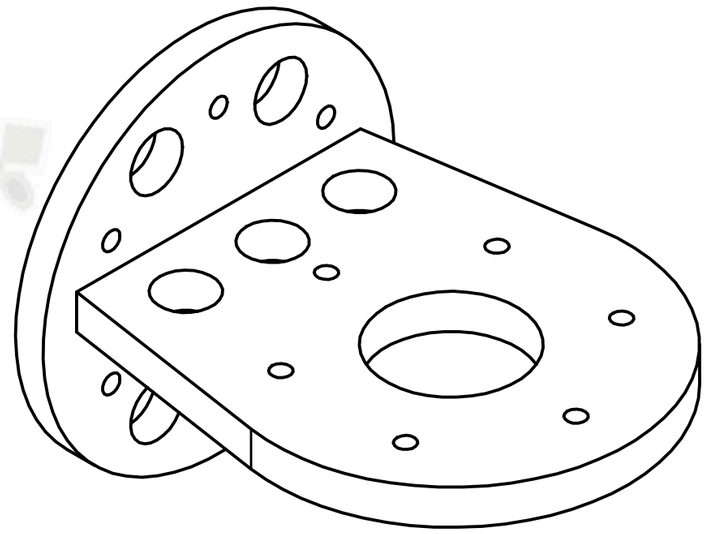


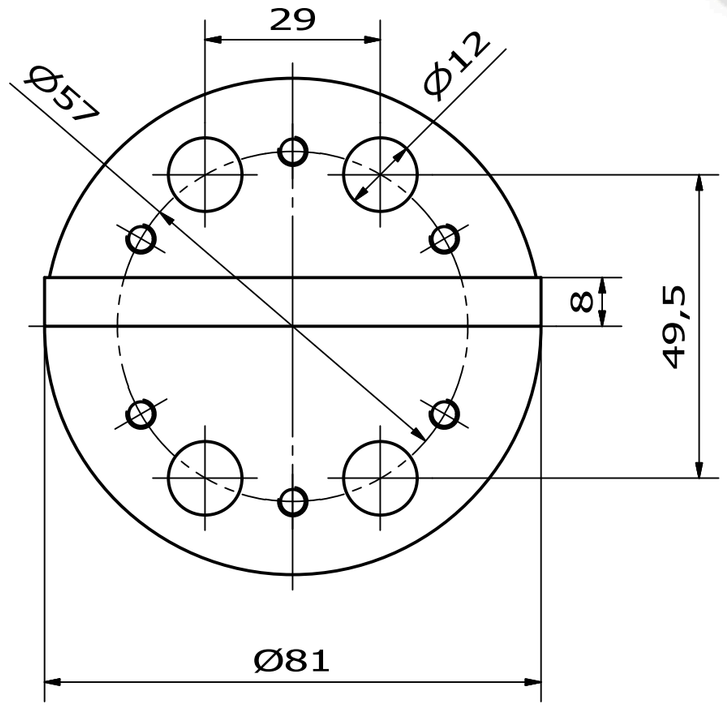
K (6 : 1)



Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 12 / 18

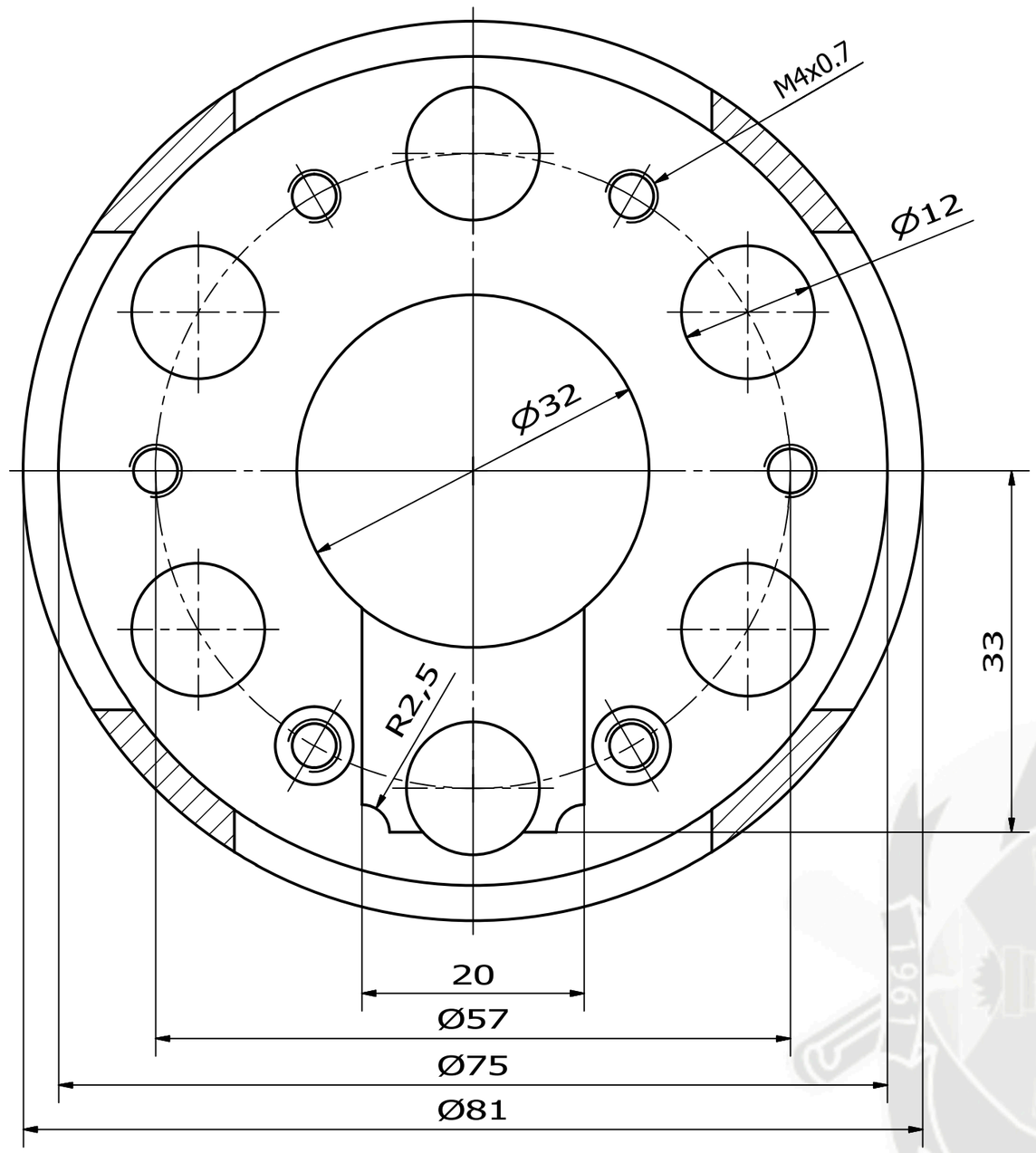


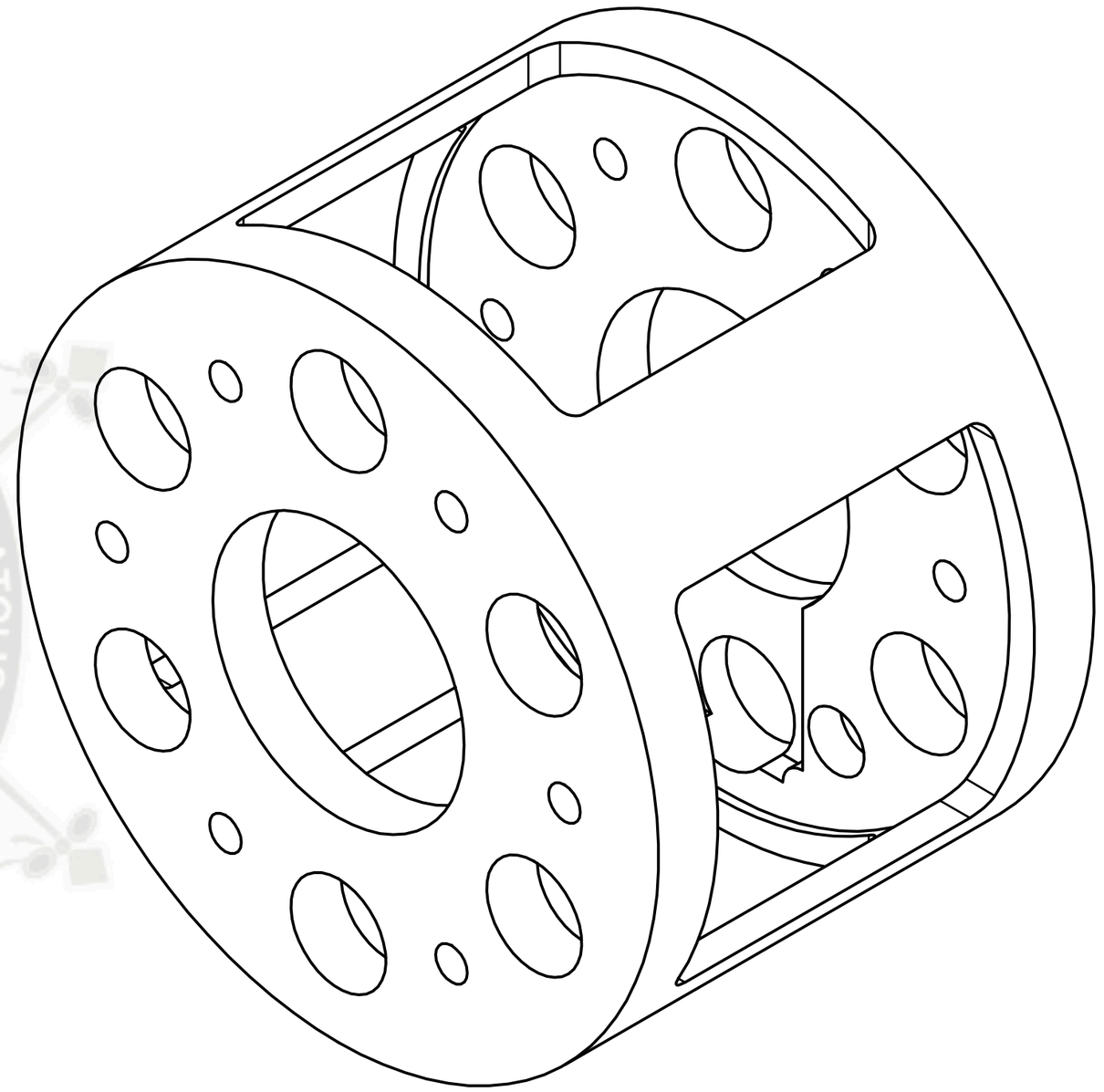


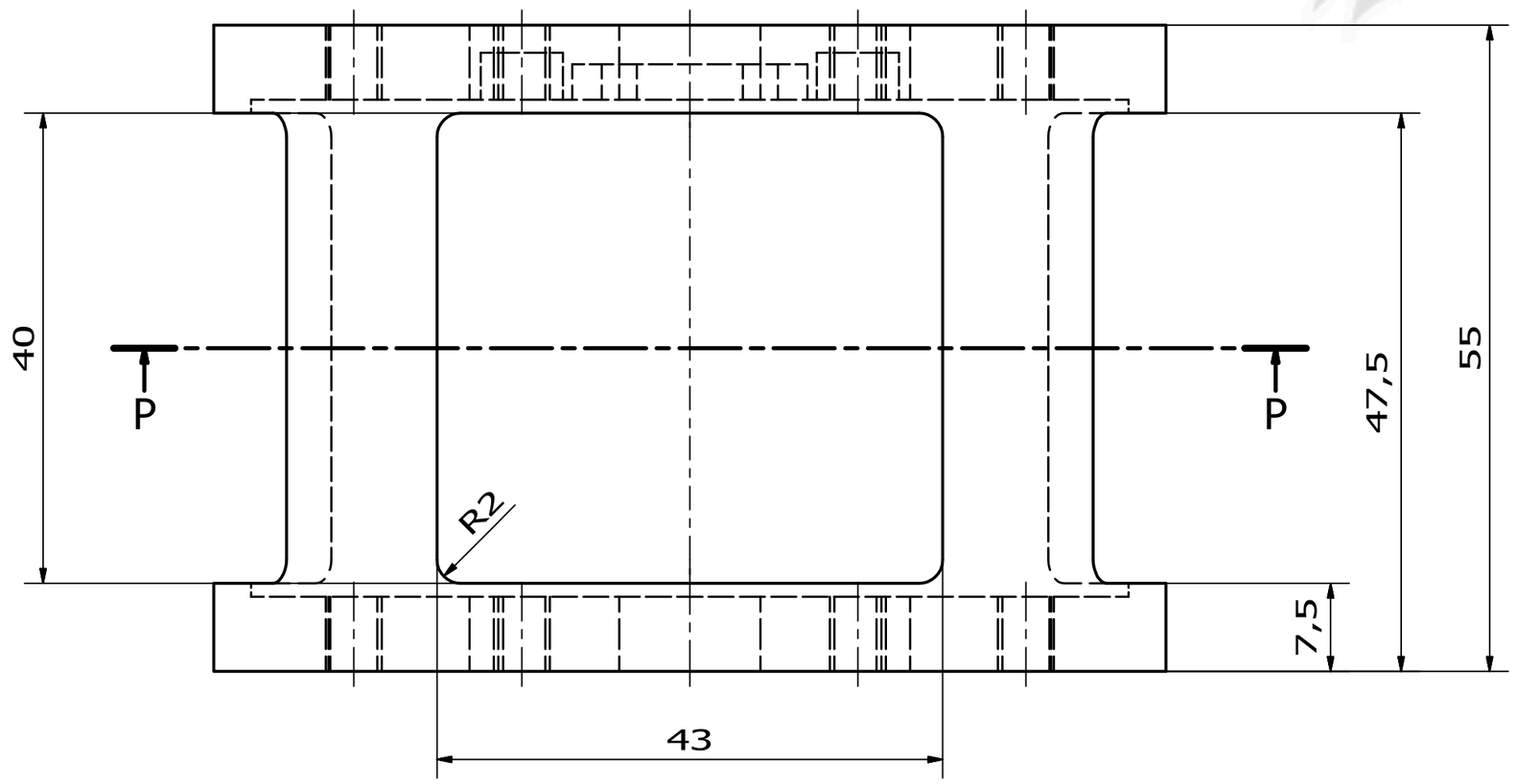


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 13 / 18

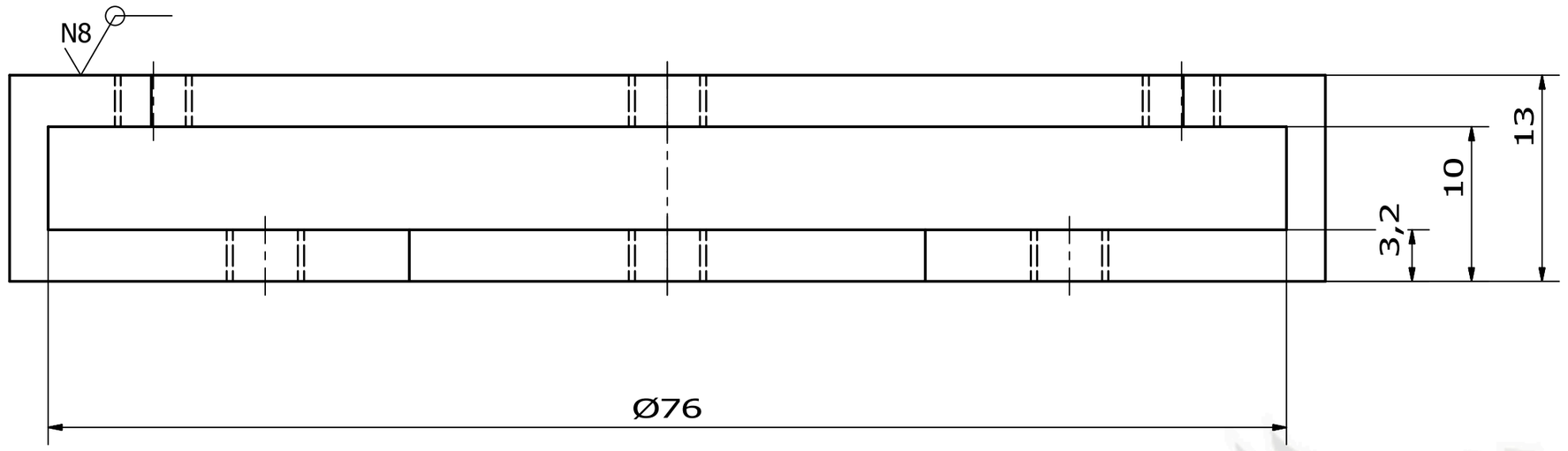
P-P (2 : 1)

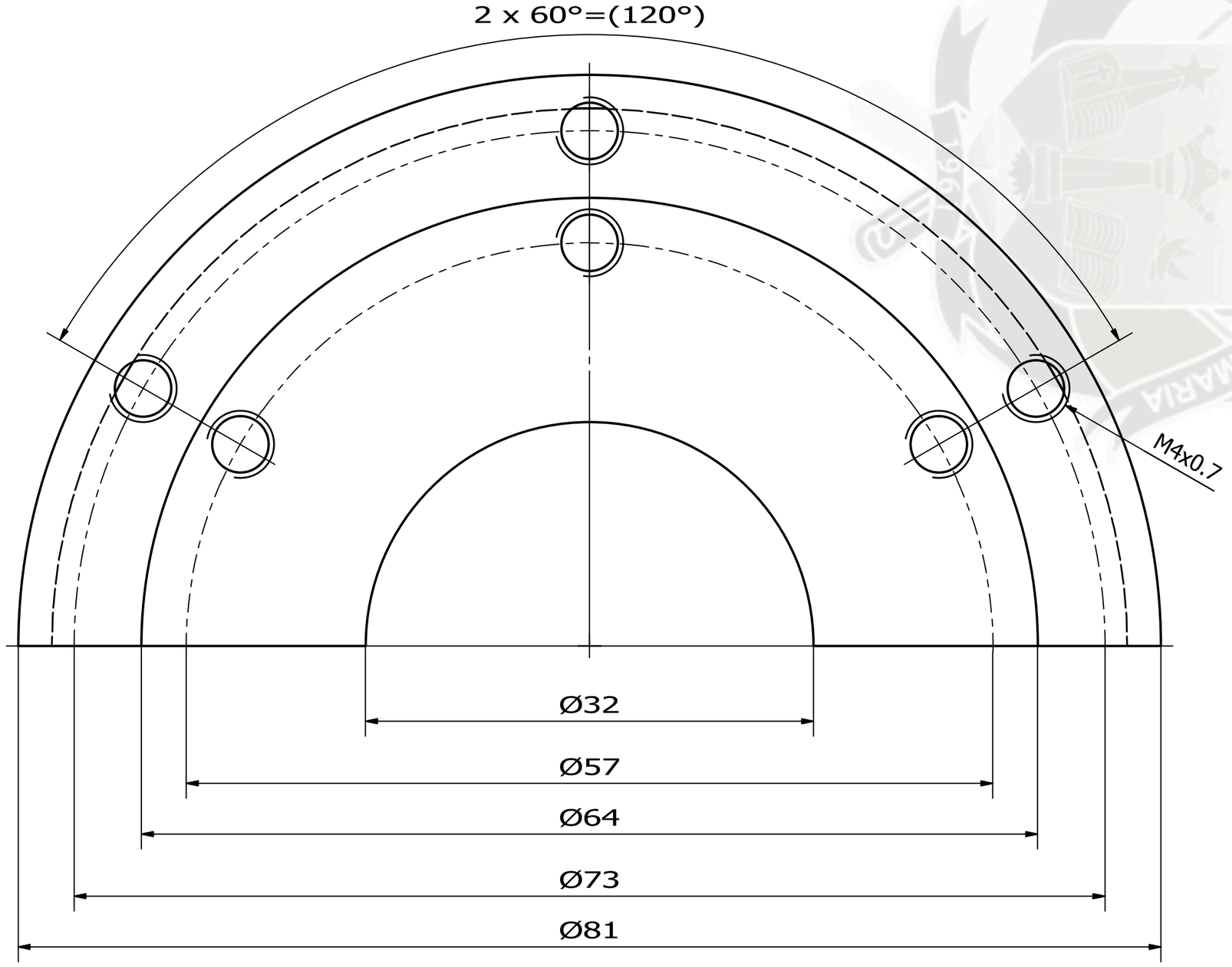


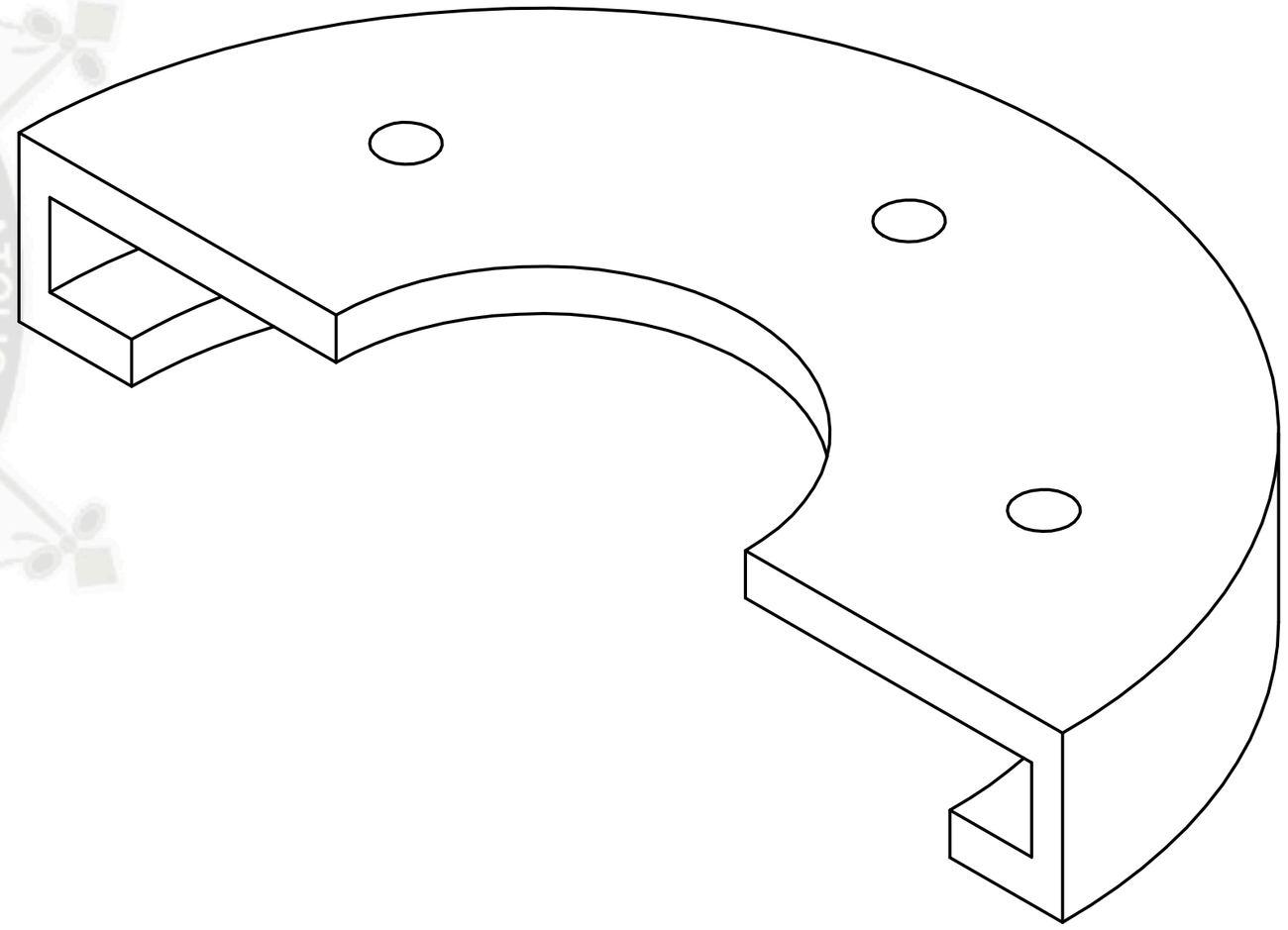




Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 14 / 18



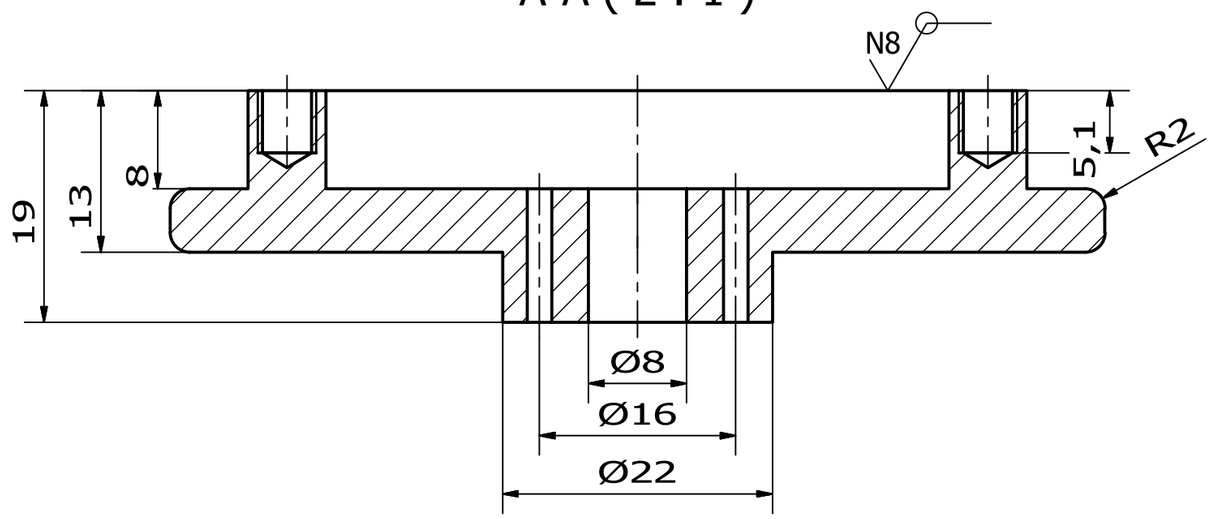


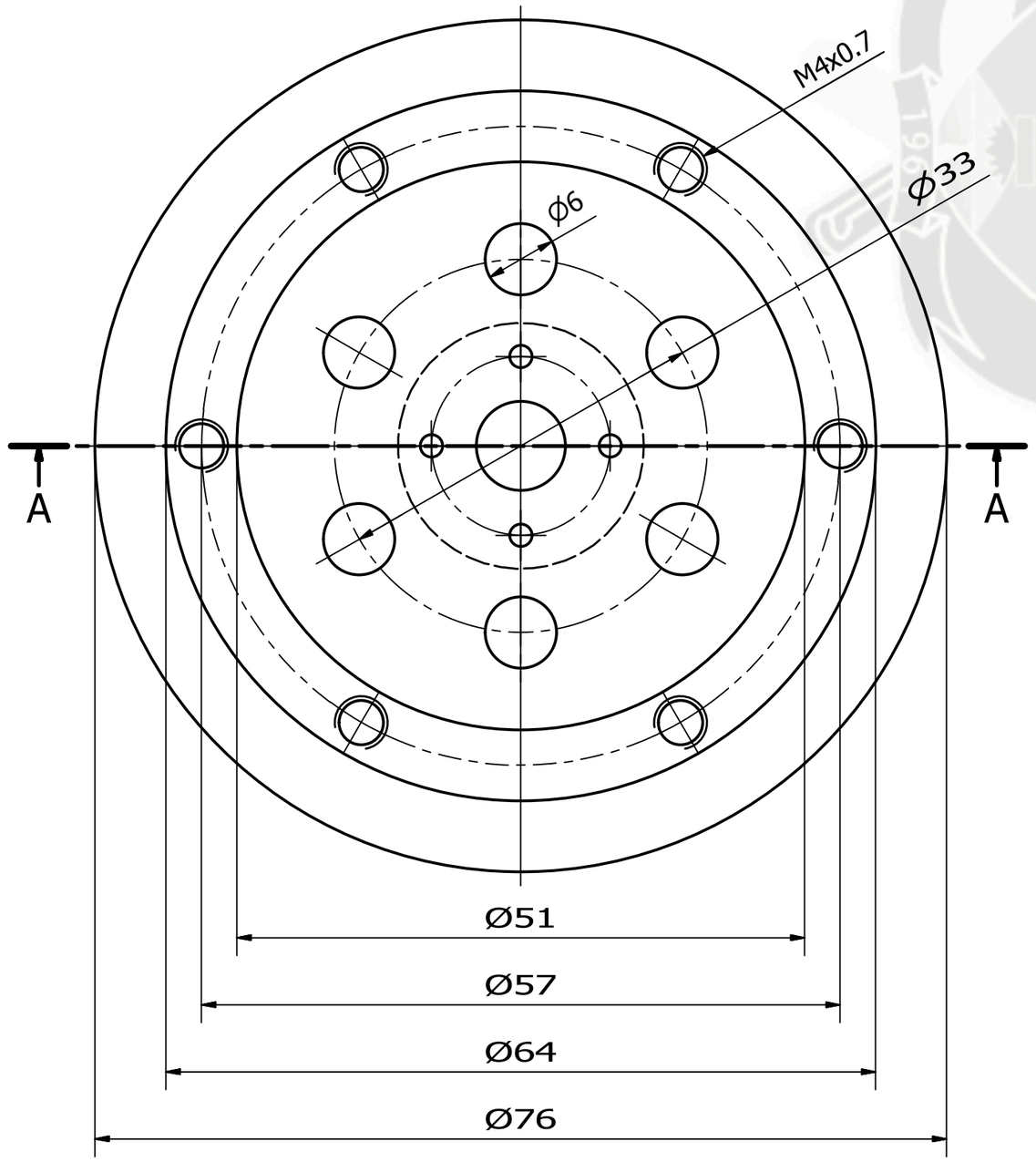


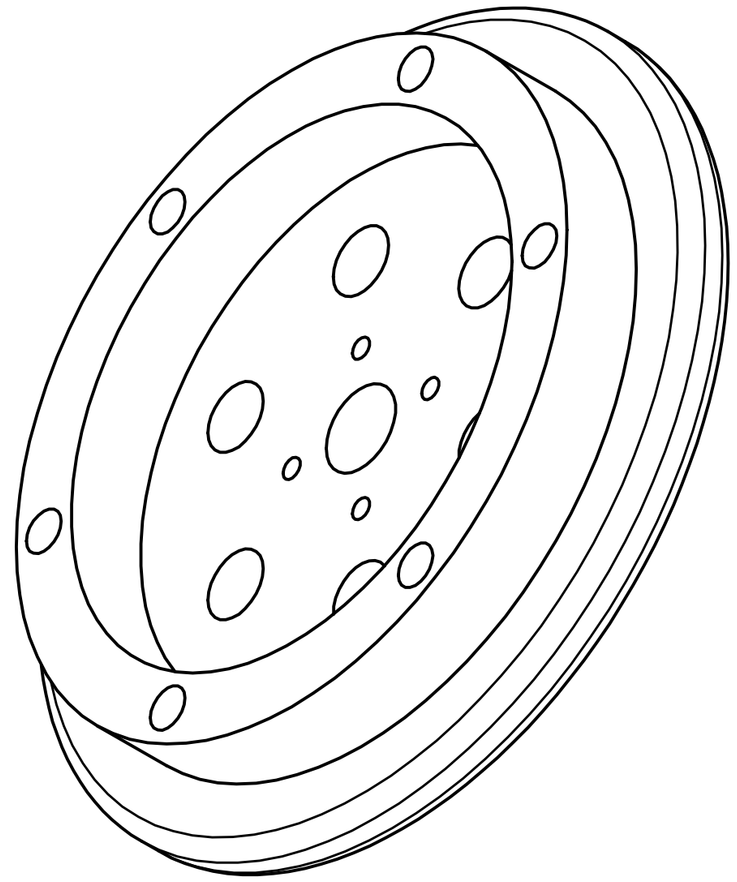
Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	

Universidad Católica de Santa María	Robotic Arm	
	Edition: 1	Sheet: 15 / 18

A-A (2 : 1)

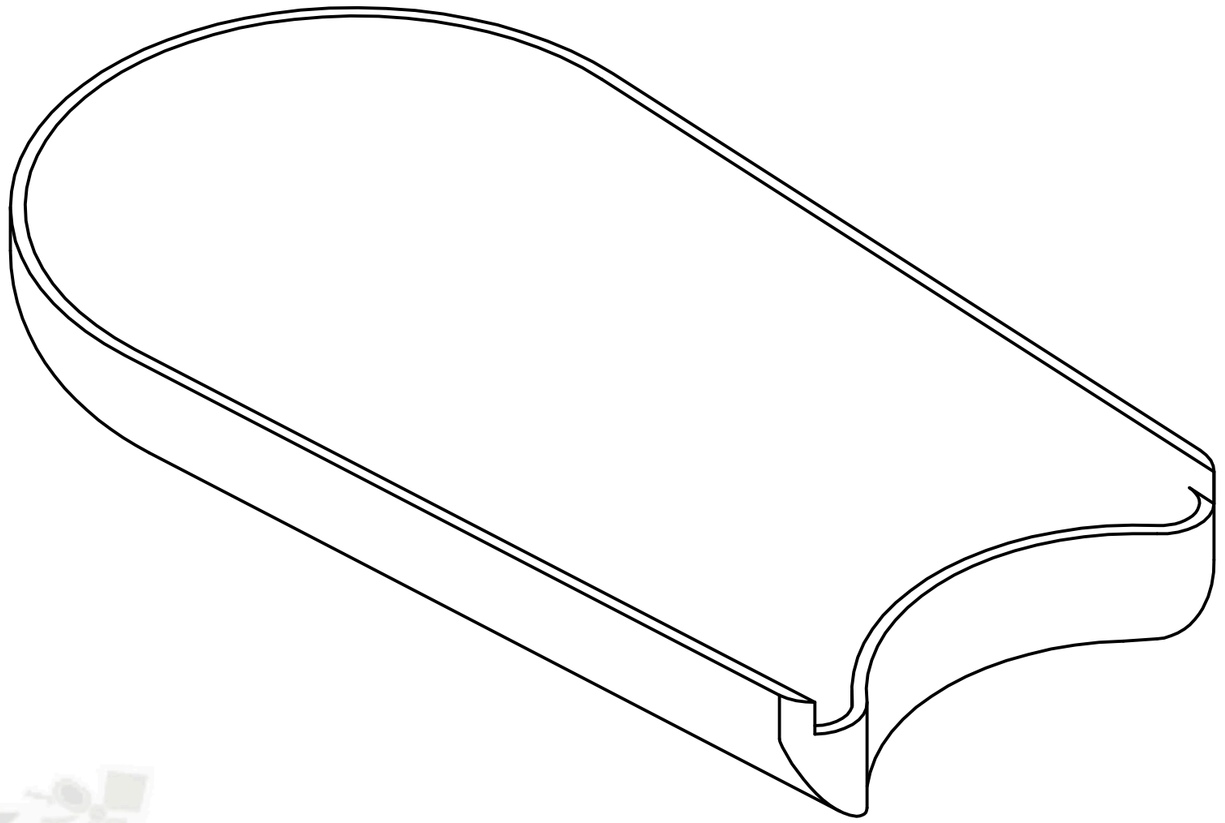


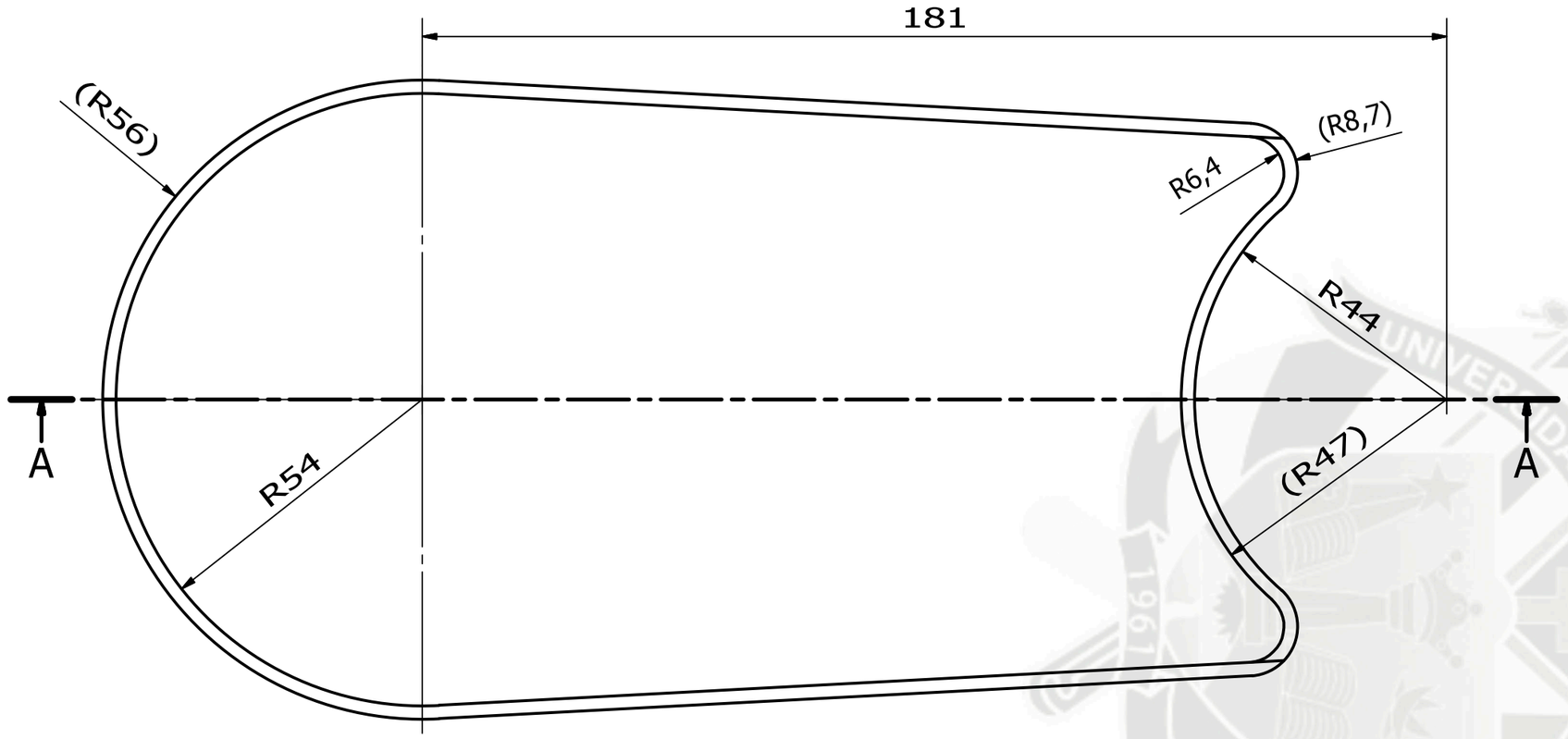


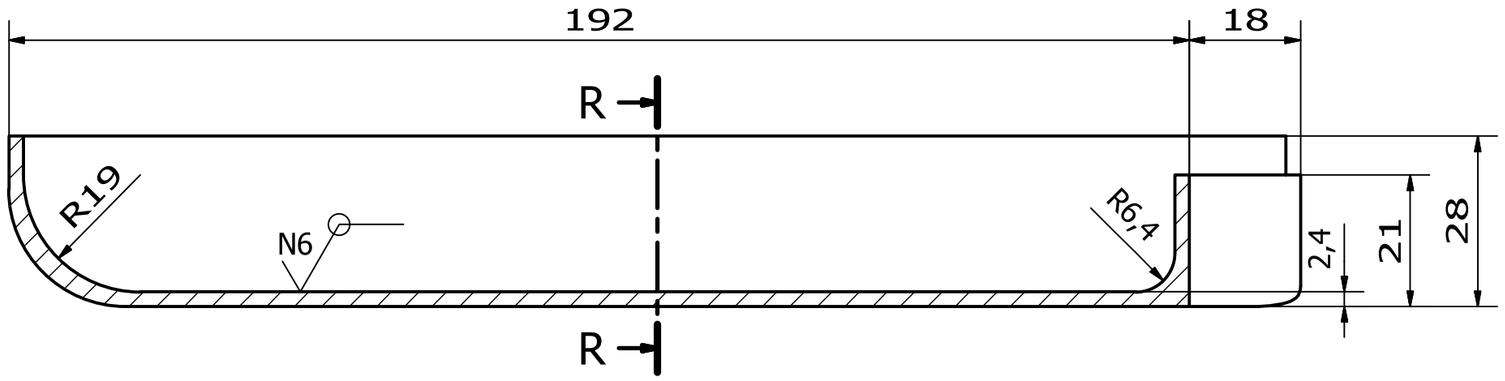


Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	

Universidad Católica de Santa María	Robotic Arm		
		Edition: 1	Sheet: 16 / 18

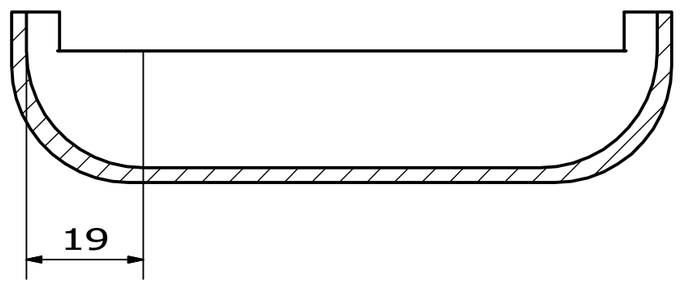






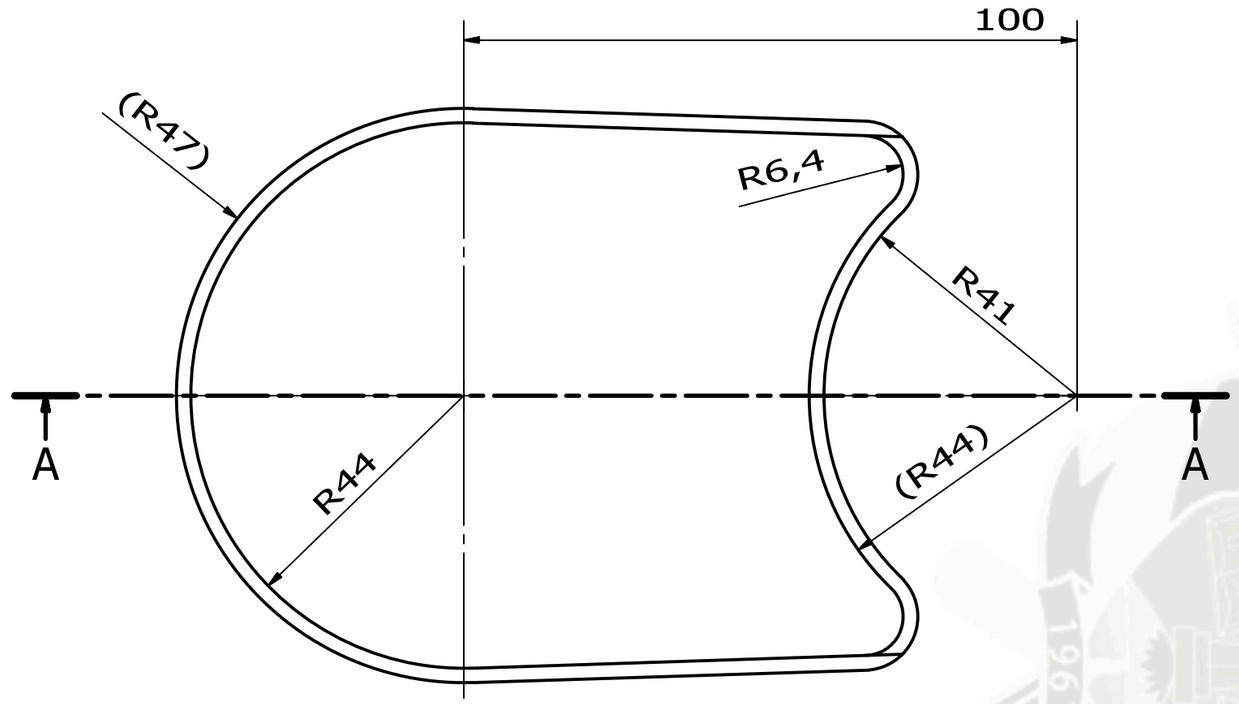
A-A (1 : 1)

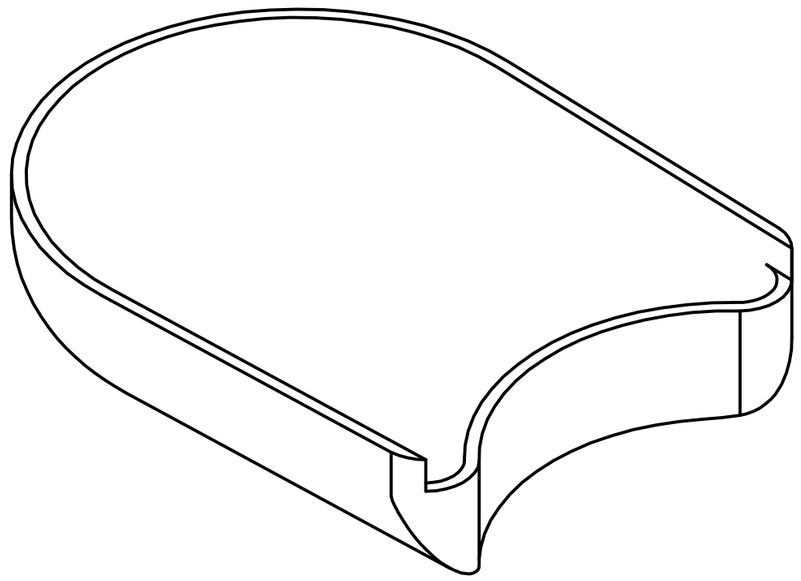
R-R (1 : 1)

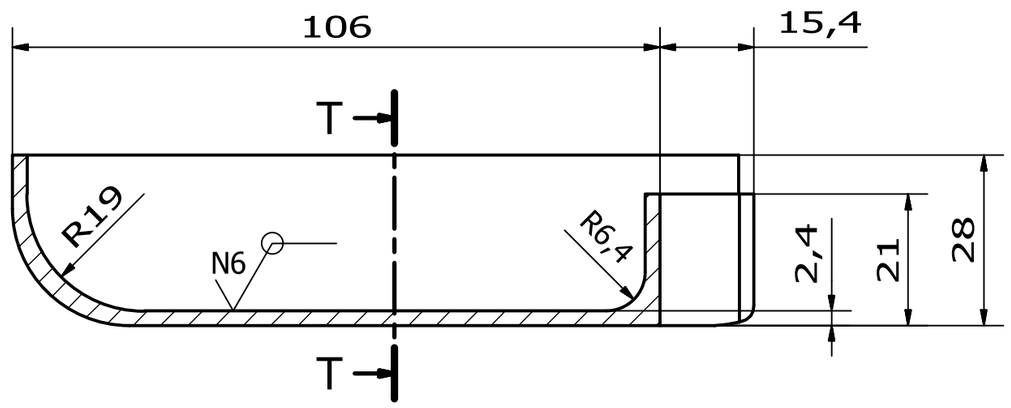


Designed by	Checked by	Approved by	Date	Date	
Carlofranco Ruiz D.		Nylon 6		17/03/2014	

Universidad Católica de Santa María	Robotic Arm		
		Edition 1	Sheet 17 / 18

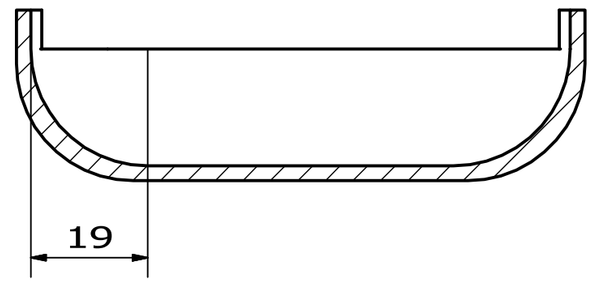




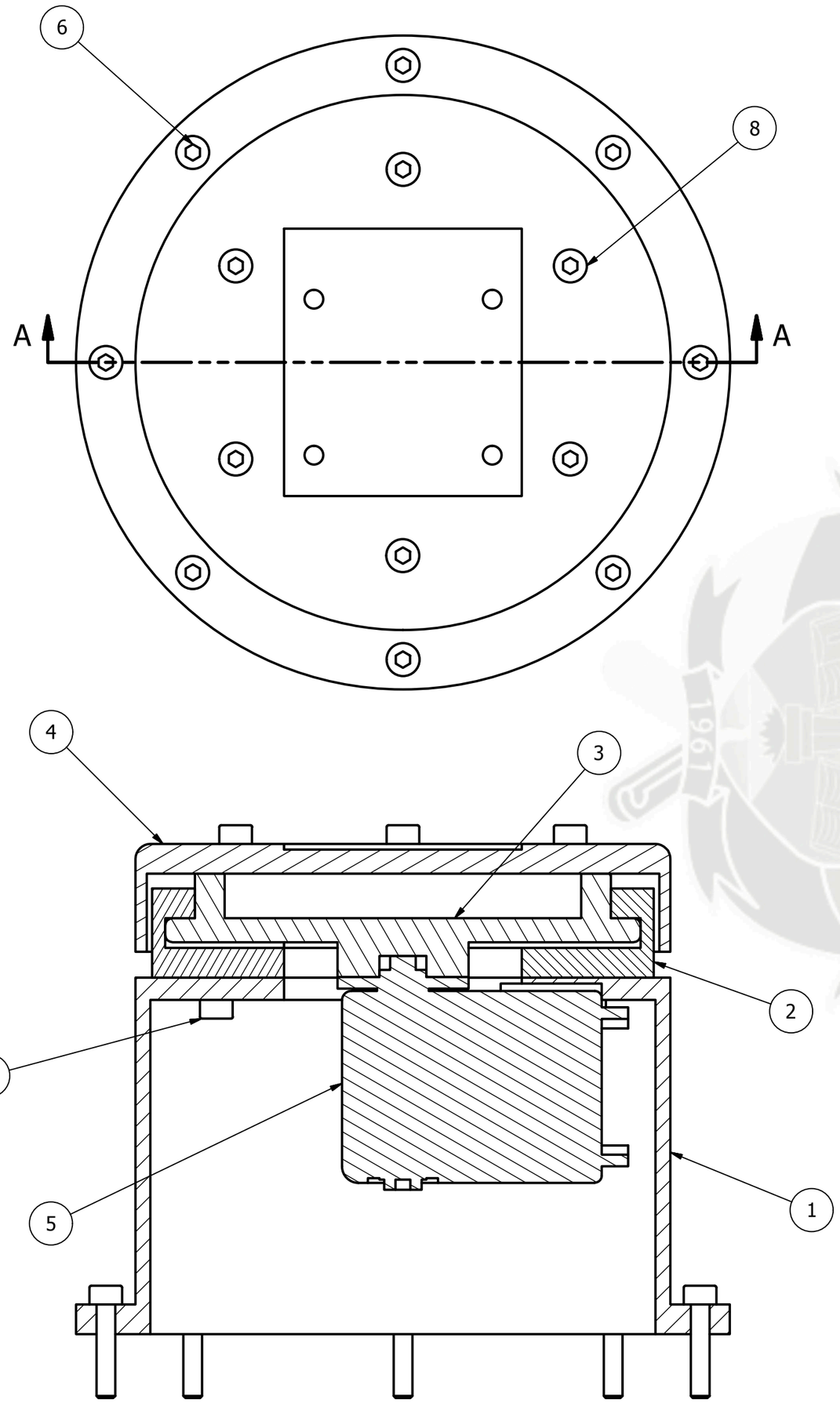


A-A (1 : 1)

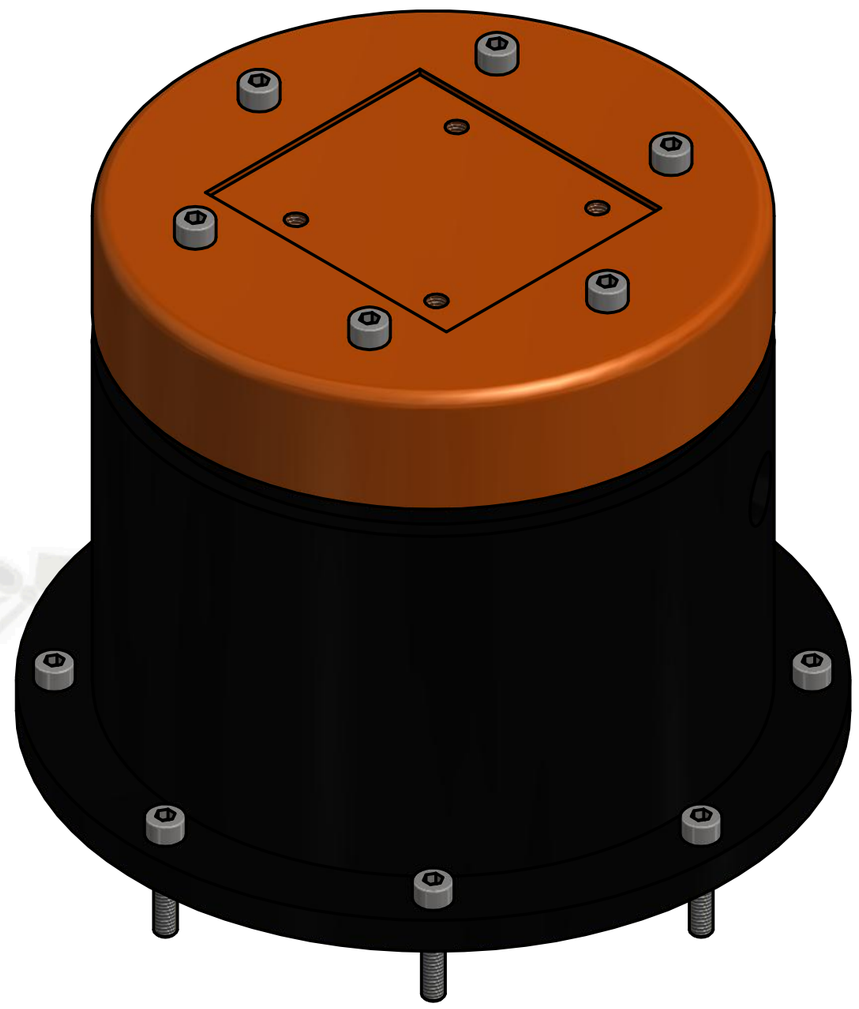
T-T (1 : 1)



Designed by: Carlofranco Ruiz D.	Checked by:	Approved by: Nylon 6	Date:	Date: 17/03/2014	
Universidad Católica de Santa María		Robotic Arm			
				Edition: 1	Sheet: 18 / 18

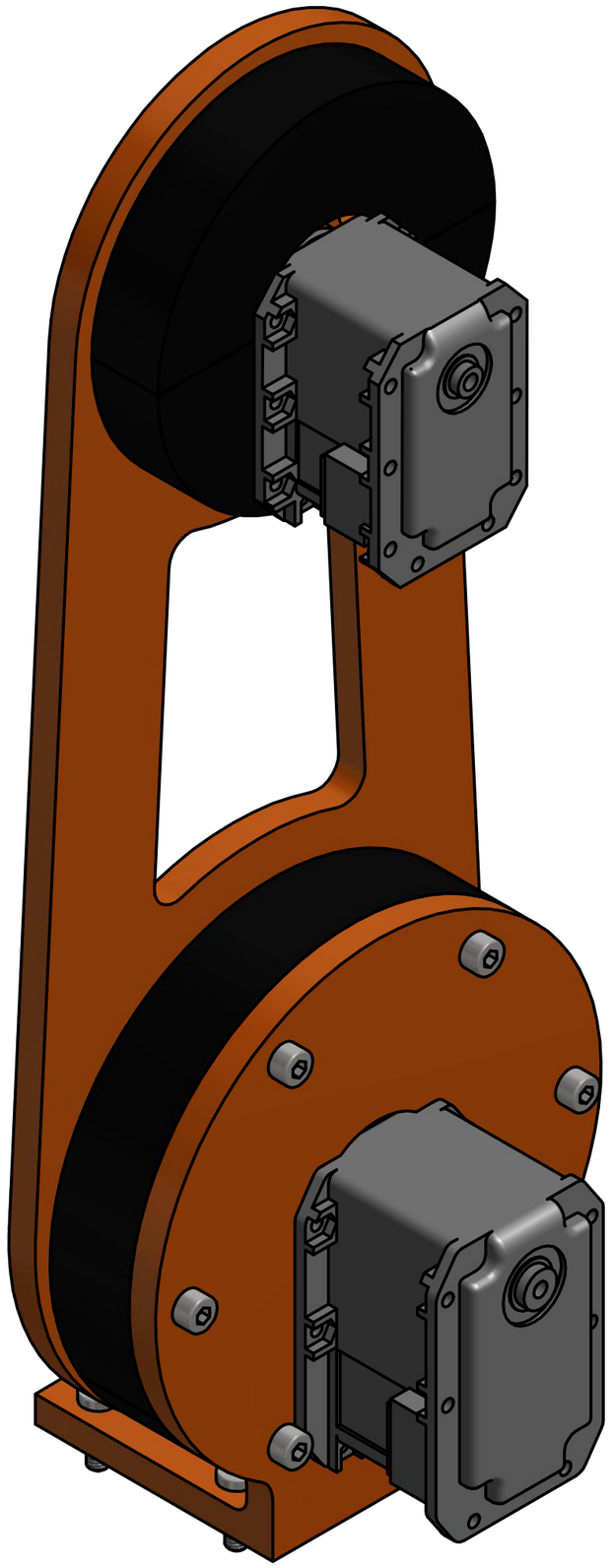


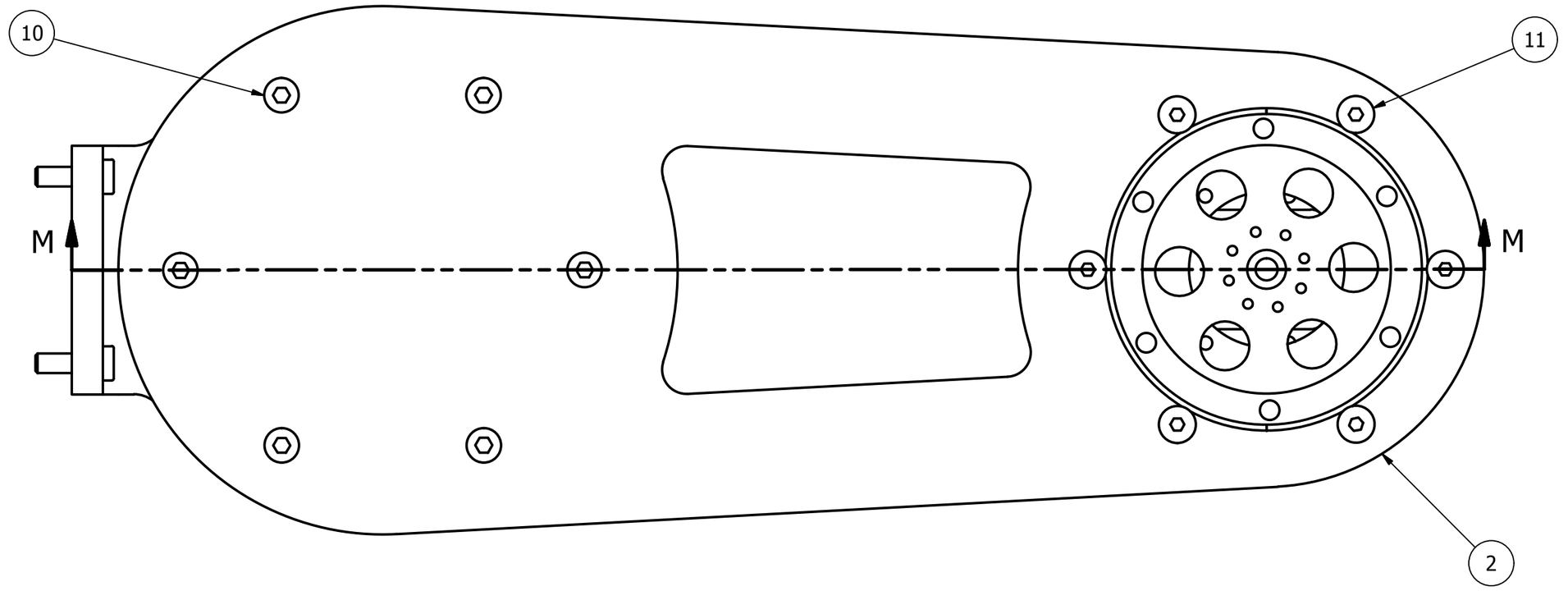
SECTION A-A
SCALE 1 : 1

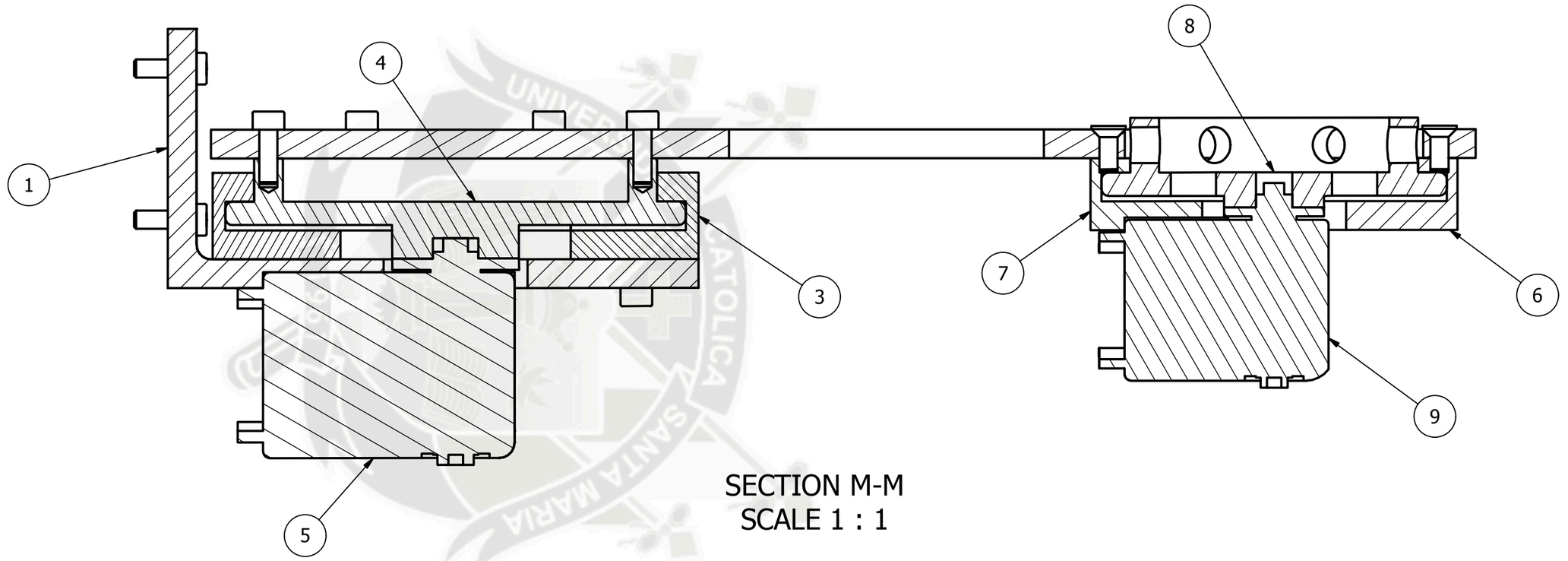


PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	Arm Base		Acetal Resin, White
2	2	Casing		Acetal Resin, White
3	1	Ring		Acetal Resin, White
4	1	Base Joint		Acetal Resin, White
5	1	rx64N		Generic
6	8	ISO 4762 - M4 x 20	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
7	6	ISO 4762 - M4 x 10	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
8	6	ISO 4762 - M4 x 12	Hexagon Socket Head Cap Screw	Stainless Steel, 440C

DRAWN Carlofranco Ruiz Daza	19/03/2014	Universidad Católica de Santa María
CHECKED		TITLE: Robotic Arm
QA		
MFG		
APPROVED		
		SIZE C / DWG NO Articulación 1 / REV 1
		SCALE / SHEET 1 OF 1





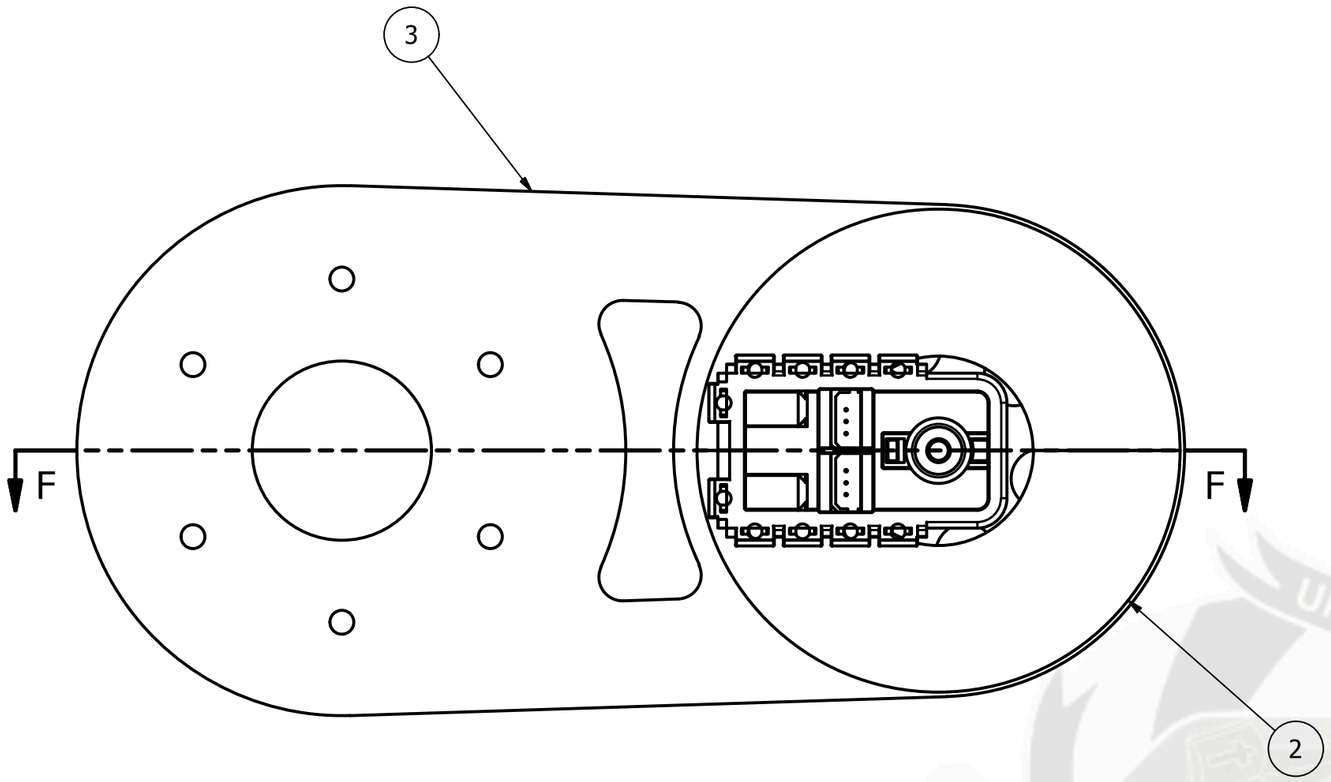


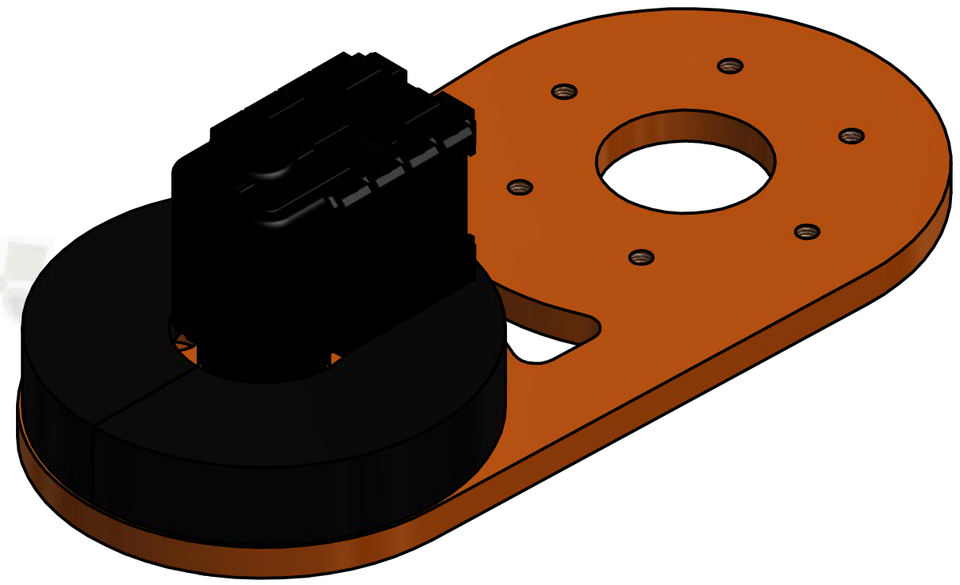
SECTION M-M
SCALE 1 : 1

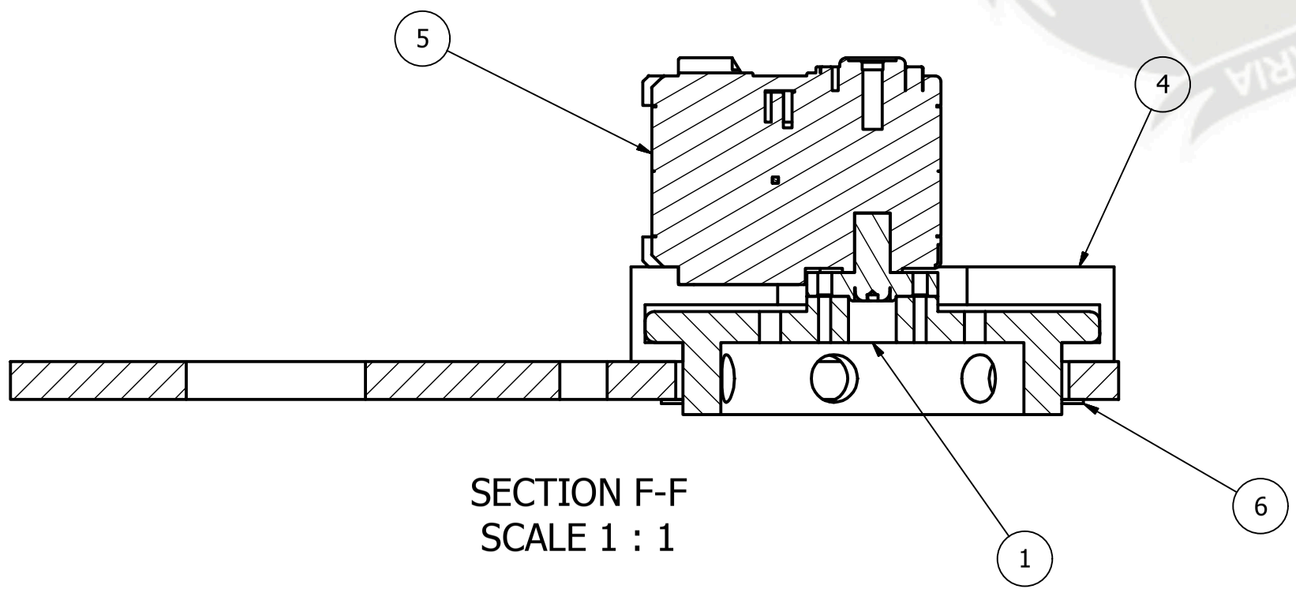
PARTS LIST

ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	Shoulder mount		Acetal Resin, White
2	1	Shoulder-Elbow		Acetal Resin, White
3	2	Casing		Acetal Resin, White
4	1	Ring		Acetal Resin, White
5	1	rx64N		Generic
6	1	Casing 3		Acetal Resin, Black
7	1	Casing 4		Acetal Resin, Black
8	1	Ring 1		Acetal Resin, White
9	1	rx28_n		Generic
10	16	ISO 4762 - M4 x 12	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
11	6	ISO 10642 - M4 x 10	Hexagon Socket Countersunk Head Screw-1 - Product grade A	Steel

DRAWN Carlofranco Ruiz Daza	19/03/2014	Universidad Católica de Santa María
CHECKED		TITLE: Robotic Arm
QA		
MFG		
APPROVED		
		SIZE C | DWG NO Articulación 2 | REV 1
		SCALE | SHEET 1 OF 1



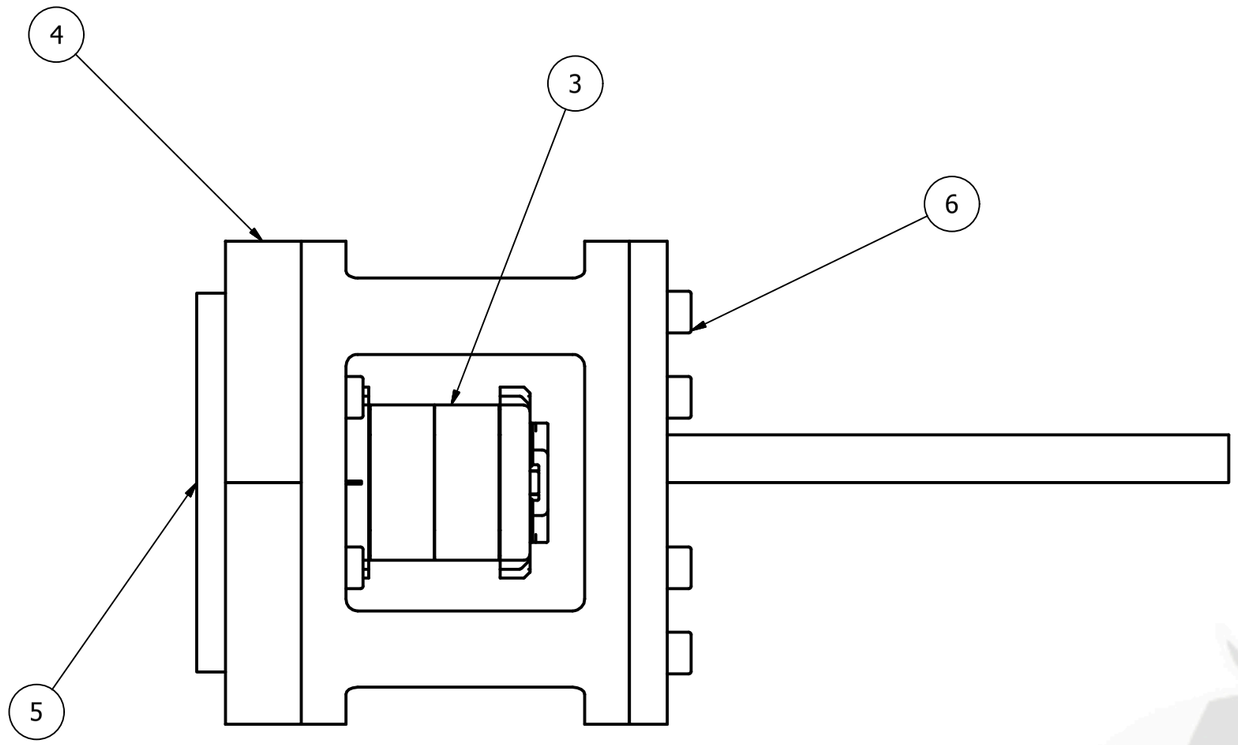


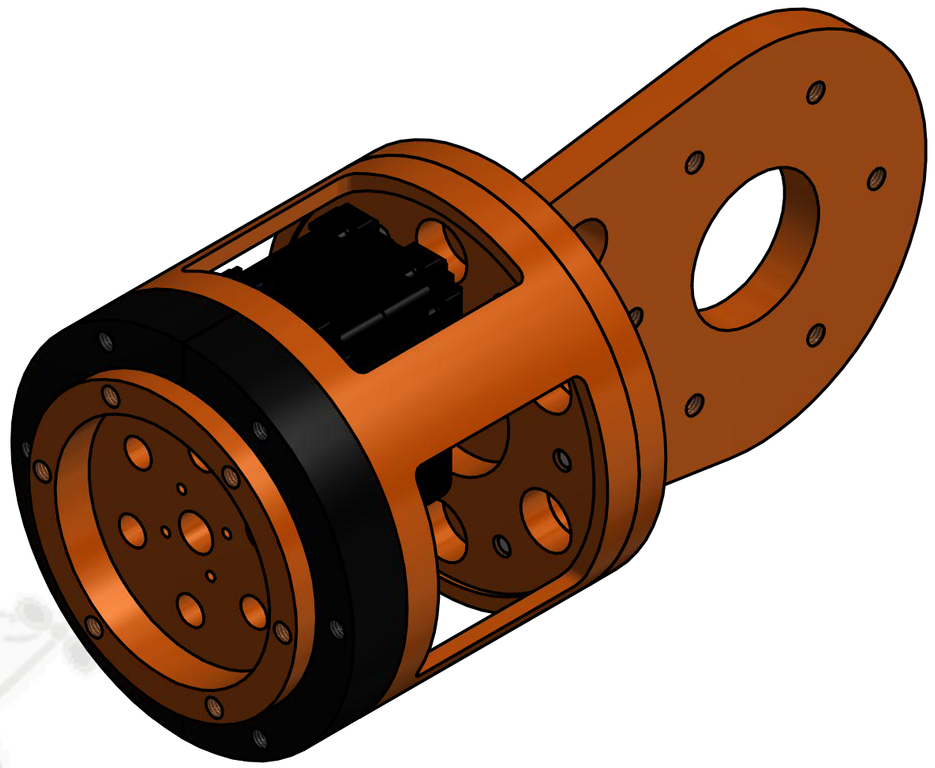


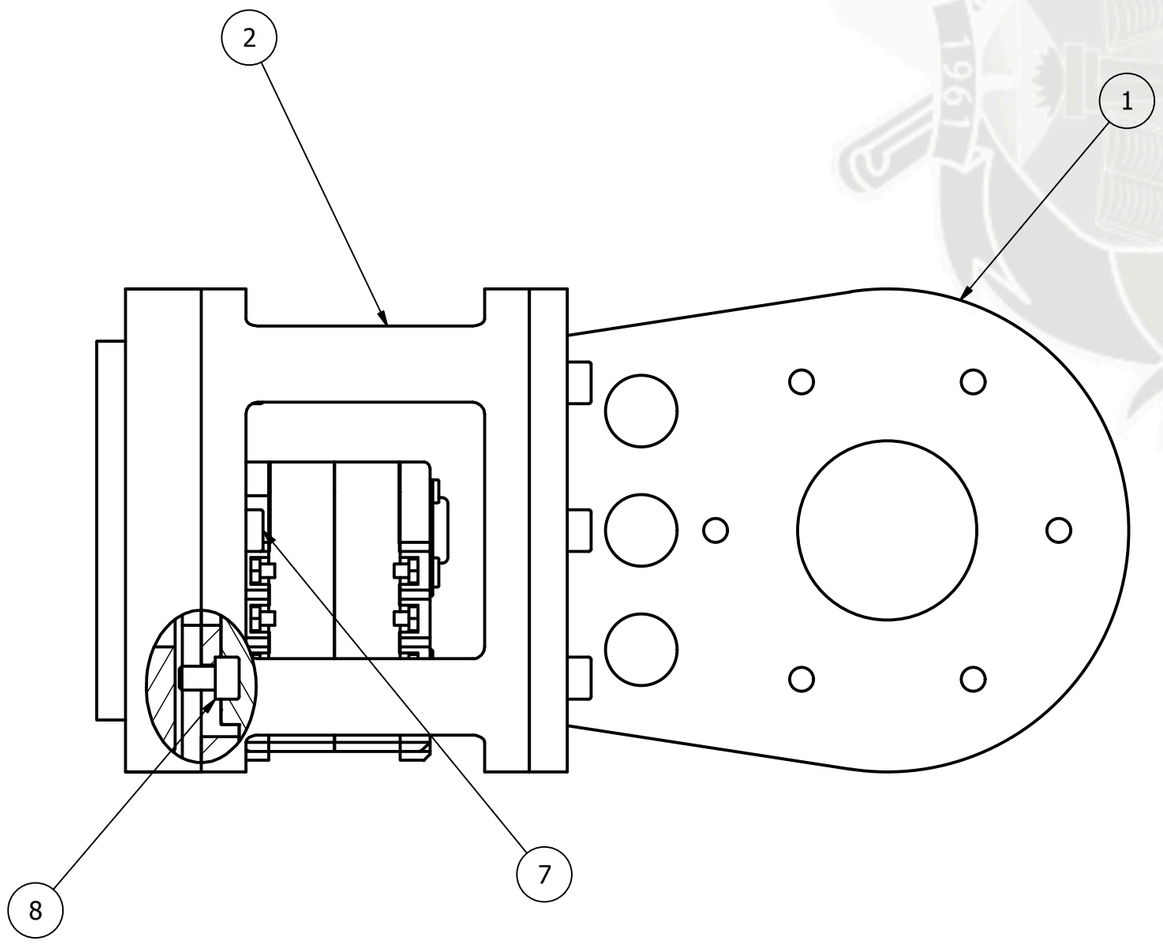
SECTION F-F
SCALE 1 : 1

PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	Ring 1		Acetal Resin, White
2	1	Casing 1		Acetal Resin, White
3	1	Elbow Wrist		Acetal Resin, White
4	1	Casing 2		Acetal Resin, Black
5	1	ax12_box		Generic
6	6	ISO 10642 - M4 x 10	Hexagon Socket Countersunk Head Screw-1 - Product grade A	Steel

DRAWN Carlofranco Ruiz Daza	19/03/2014	Universidad Católica de Santa María
CHECKED		TITLE: Robotic Arm
QA		
MFG		
APPROVED		
		SIZE C | DWG NO Articulación 3 y 4 | REV 1
		SCALE | SHEET 1 OF 1

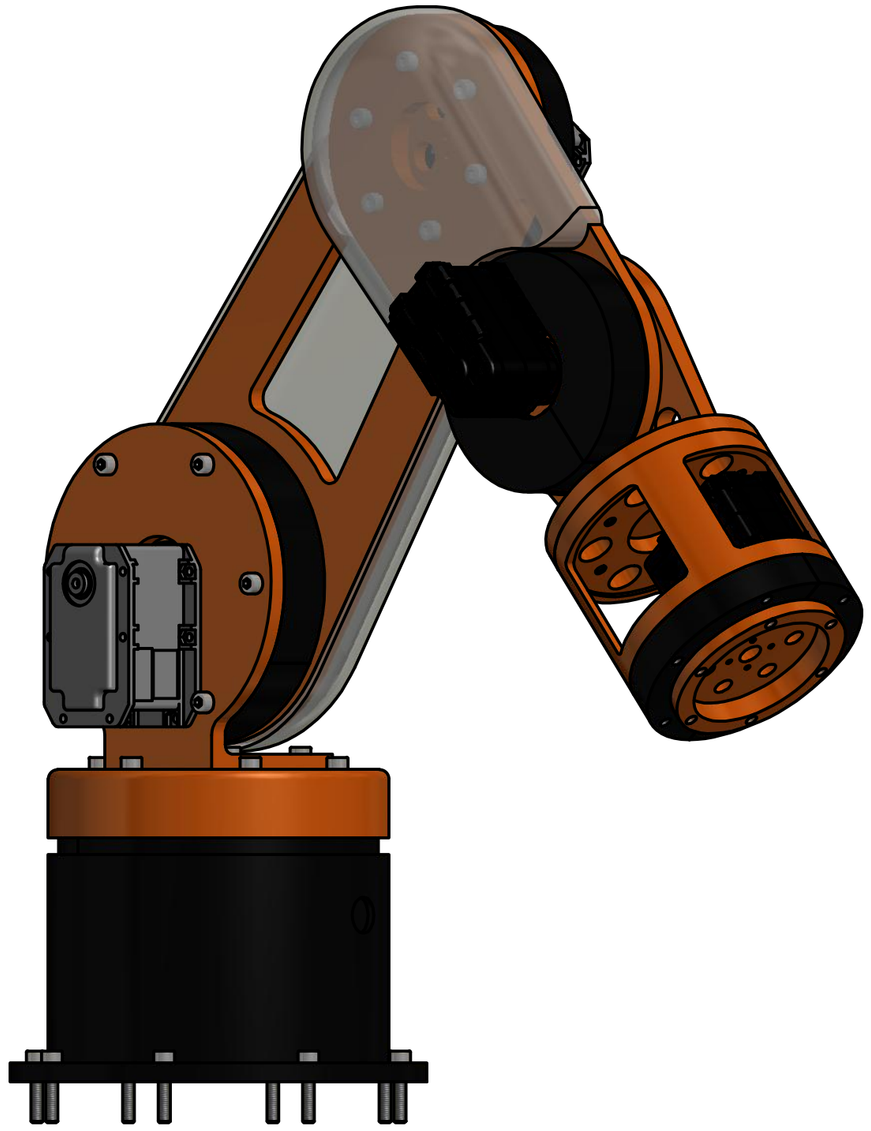


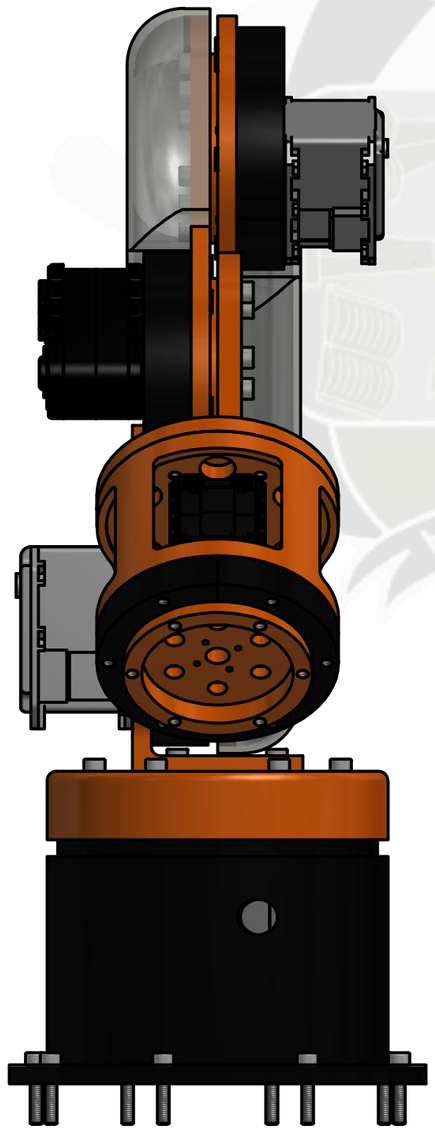




PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	Wrist Hand		Acetal Resin, White
2	1	Wrist Case		Acetal Resin, White
3	1	AX12		
4	2	Casing 5		Acetal Resin, White
5	1	Ring 2		Acetal Resin, White
6	6	ISO 4762 - M4 x 12	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
7	4	ISO 4762 - M4 x 10	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
8	2	ISO 4762 - M4 x 6	Hexagon Socket Head Cap Screw	Stainless Steel, 440C

DRAWN Carlofranco Ruiz Daza	19/03/2014	Universidad Católica de Santa María
CHECKED		TITLE: Robotic Arm
QA		
MFG		
APPROVED		
		SIZE C / DWG NO Articulación 5 / REV 1
		SCALE / SHEET 1 OF 1





PARTS LIST				
ITEM	QTY	PART NUMBER	DESCRIPTION	MATERIAL
1	1	Arm Base		Acetal Resin, White
2	4	Casing		Acetal Resin, White
3	2	Ring		Acetal Resin, White
4	1	Base Joint		Acetal Resin, White
5	1	Shoulder mount		Acetal Resin, White
6	1	Shoulder-Elbow		Acetal Resin, White
7	2	Ring 1		Acetal Resin, White
8	1	Casing 3		Acetal Resin, Black
9	1	Casing 4		Acetal Resin, Black
10	1	Elbow Wrist		Acetal Resin, White
11	1	Casing 1		Acetal Resin, White
12	1	Casing 2		Acetal Resin, Black
13	1	Wrist Hand		Acetal Resin, White
14	1	Wrist Case		Acetal Resin, White
15	2	Casing 5		Acetal Resin, White
16	1	Ring 2		Acetal Resin, White
17	1	Shoulder-Elbow Case		Polycarbonate, Clear
18	1	Elbow Wrist Case		Polycarbonate, Clear
19	2	ax12_box		Generic
20	2	rx64N		Generic
21	1	rx28_n		Generic
22	8	ISO 4762 - M4 x 20	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
23	10	ISO 4762 - M4 x 10	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
24	40	ISO 4762 - M4 x 12	Hexagon Socket Head Cap Screw	Stainless Steel, 440C
25	12	ISO 10642 - M4 x 10	Hexagon Socket Countersunk Head Screw-1 - Product grade A	Steel
26	2	ISO 4762 - M4 x 6	Hexagon Socket Head Cap Screw	Stainless Steel, 440C

DRAWN Carlofranco Ruiz Daza	28/05/2013	Universidad Católica de Santa María
CHECKED Ph.D.Hermann Alcázar	29/05/2013	TITLE: Robotic Arm
QA		
MFG		
APPROVED		
		SIZE C / DWG NO Robotic Arm (Conjunto) / REV
		SCALE / SHEET 1 OF 1

Anexo B: Especificaciones del Nylon 6

RÖCHLING SUSTAPLAST KG

Sustaplast Straße 1
D-56112 Lahnstein/Germany

Tel. 02621/693-0
Fax 02621/693170
Internet: www.sustaplast.de

RÖCHLING Sustaplast

Material Data Sheet

Trade name	SUSTAMID 6 G
DIN EN ISO 1043 designation	PA 6
Modification:	none

Properties	*Unit*	*Test method*	*Value*
General Properties			
Density	g/cm³	DIN EN ISO 1183-1	1,15
Moisture absoprtion			
Saturation in air of 23°C/50% RH	%	DIN EN ISO 62	2,5
Flammability acc.to UL 94 (Thickn. 3mm/6mm)		ISO 1210 (UL 94)	HB / V2
Mechanical Properties			*Test specimen "dry"*
Yield point	MPa	DIN EN ISO 527	75
Elongation at break	%	DIN EN ISO 527	>45
Tensile modulus of elasticity	MPa	DIN EN ISO 527	3.400
Notched impact strength (Charpy)	kJ/m²	ISO 179/1eA/Pendel 1J	>3
Ball indentation hardness	N/mm²	DIN EN ISO 2039-1	180
Shore - Hardness	Skala D	DIN 53505	83
Thermal Properties			
Melting temperature	°C	ISO 11357	216
Thermal conductivity	W/(mK)	DIN 52612	0,25
Specific thermal capacity	kJ/(kgK)	DIN 52612	1,7
Coefficient of linear thermal expansion	10^{-6} K^{-1}	Average betw.20°C-60°C	80
Service temperature - long-term	°C		- 40 up to 110
Service temperture - short-term, max.	°C		170
Heat deflection temperature, Method A:1,8 MPa	°C	DIN EN ISO 75	95
Electrical Properties			
Dielectric constant, 50 Hz		IEC 60250	3,7
Dielectric dissipation factor, 50 Hz		IEC 60250	0,02
Volume resistivity	Ohm cm	IEC 60093	10^{15}
Surface resistivity	Ohm	IEC 60093	10^{13}
Comperative tracking index CTI, Sol. A		IEC 60112	600
Dielectric strength	kV/mm	IEC 60243	20

Remarks:

The following applies to Polyamides:
Under the influence of moisture absorption, the mechanical properties change. The material becomes tougher and more resistant to impact, the modulus of elasticity declines. Depending on the environmental atmosphere, the temperature and the period of moisture absorption, only the surface layer ist affected by alterations of property to a certain depth. On thick-walled parts, the center area remains unaffected.

The short-term maximum application temperature only applies to very low mechanical stress for a few hours.
The long-term maximum application temperature is based on the thermal ageing of plastics by oxidation, resulting in a decrease of the mechanical properties. This applies to an exposure to temperatures for at least 5.000 hours causing a 50% loss of the tensile strength from the original value (measured at room temperature). This value says nothing about the mechanical strength of the material at high application temperatures. In case of thick-walled parts, only the surface layer ist affected by oxidation from high temperatures. With the addition of antioxidants, a better protection of the surface layer is achieved. In any case, the center area of the material remains unaffected.
The minimum application temperature ist basically influenced by possible stress factors like impact and/or shock under application. The values stated refer to an minimum degree of impact stress.
The electrical properties as stated result from measurements on natural, dry material. With other colours (in particular black) or saturated material, there may be clear differences in the electrical properties.
The values indicated result from numerous individual measurements for an approximation of the values and are to our today's knowledge. They serve as information about our products and are presented as a guide to choose from our range of materials. This, however, does not include an assurance of specific properties or the suitability for particular application purposes that are legally binding. Since the properties also depend on the dimension of the semi-finished products and the degree of crystallisation (e.g. nucleating by pigments), the actual values of the properties of a particular product may differ from the indicated values.
* The mechanical properties of fibre reinforced material were measured on injection molded samples, parallel to fibre direction.
Special construction details of further material specifications on request.

Anexo C: Datasheet de los actuadores

AX-12W/AX-12A

Parts Photo

[AX-12W] **[AX-12A]**

Hardware Specifications

- Weight: 52.9 g (AX-12W), 54.6 g (AX-12A)
- Dimension: 32 mm * 50 mm * 40 mm
- Resolution: 0.29°
- Gear Reduction Ratio: 32 : 1 (AX-12W), 254 : 1 (AX-12A)
- Stall Torque: 0.2 N.m (AX-12W), 1.5 N.m (AX-12A) (at 12.0 V, 1.5 A)
- No load speed: 470 RPM (at 12 V, wheel mode), 54-59 RPM (at 12 V, joint mode)
- Running Degree:
  - 0° ~ 300°
  - Endless Turn
- Running Temperature: -5 ℃ ~ +70 ℃
- Voltage: 9 ~ 12 V (Recommended Voltage 11.1 V)
- Command Signal: Digital Packet
- Protocol Type: Half duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
- Link (Physical): TTL Level Multi Drop (daisy chain type Connector)
- ID: 254 ID (0~253)
- Communication Speed: 7843bps ~ 1 Mbps
- Feedback: Position, Temperature, Load, Input Voltage, etc.
- Material: Engineering Plastic

Control Table

Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet.

EEPROM and RAM

Data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.

Address

It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.

Access

Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.

Initial Value

In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.

Highest/Lowest Byte

In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16bit, but it is divided into 8bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	44 (0X2C)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	1 (0X01)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	1 (0X01)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	70 (0X46)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	140 (0XBE)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)

	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36(0x24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36(0x24)
RAM	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	4 (0X04)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	4 (0X04)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	64 (0X40)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	64 (0X40)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Address Function Help

EEPROM Area

Model Number

It represents the Model Number.

Firmware Version

It represents the firmware version.

ID

It is a unique number to identify Dynamixel. The range from 0 to 252 (0xFC) can be used, and, especially, 254(0xFE) is used as the Broadcast ID. If the Broadcast ID is used to transmit Instruction Packet, we can command to all Dynamixels. Please be careful not to duplicate the ID of connected Dynamixel.

Baud Rate

It represents the communication speed. 0 to 254 (0xFE) can be used for it. This speed is calculated by using the below formula:

Speed(BPS) = 2000000/(Data+1)

Data	Set BPS	Target BPS	Tolerance
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

Note: Maximum Baud Rate error of 3% is within the tolerance of UART communication.

Return Delay Time

It is the delay time per data value that takes from the transmission of Instruction Packet until the return of Status Packet. 0 to 254 (0xFE) can be used, and the delay time per data value is 2 µs. That is to say, if the data value is 10, 20 µs is delayed. The initial value is 250 (0xFA) (i.e., 0.5 ms).

CW/CCW Angle Limit

The angle limit allows the motion to be restrained. The range and the unit of the value is the same as Goal Position (Address 30, 31).

- **CW Angle Limit:** the minimum value of Goal Position (Address 30, 31).
- **CCW Angle Limit:** the maximum value of Goal Position (Address 30, 31).

The following two modes can be set pursuant to the value of CW and CCW.

Operation Type	CW / CCW
Wheel Mode	the value of the both are 0
Joint Mode	the value of the both are not 0

The wheel mode can be used to wheel-type operation robots since motors of the robots spin infinitely. The joint mode can be used to multi-joints robot since the robots can be controlled with specific angles.

The Highest Limit Temperature

Caution: Do not set the temperature lower/higher than the default value.

When the temperature alarm shutdown occurs, wait 20 minutes to cool the temperature before re-use. Using the product when the temperature is high may and can cause damage.

The Lowest (Highest) Limit Voltage

It is the operation range of voltage. 50 to 250 (0x32 ~ 0x96) can be used. The unit is 0.1 V.

For example, if the value is 80, it is 8 V. If Present Voltage (Address42) is out of the range, Voltage Range Error Bit (Bit0) of Status Packet is returned as '1' and Alarm is triggered as set in the addresses 17 and 18.

Max Torque

It is the torque value of maximum output. 0 to 1023 (0x3FF) can be used, and the unit is about 0.1%. For example, Data 1023 (0x3FF) means that Dynamixel will use 100% of the maximum torque it can produce while Data 512 (0x200) means that Dynamixel will use 50% of the maximum torque. When the power is turned on, Torque Limit (Addresses 34 and 35) uses the value as the initial value.

Status Return Level

It decides how to return Status Packet. There are three ways like the below table.

Value	Return of Status Packet
0	No return against all commands (Except PING Command)
1	Return only for the READ command
2	Return for all commands

When Instruction Packet is Broadcast ID, Status Packet is not returned regardless of Status Return Level.

Alarm LED

Alarm Shutdown

Dynamixel can protect itself by detecting errors occur during the operation. The errors can be set are as the table below.

Bit	Name	Contents
Bit 7	0	-
Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	CheckSum Error	When the Checksum of the transmitted Instruction Packet is invalid
Bit 3	Range Error	When the command is given beyond the range of usage
Bit 2	OverHeating Error	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Error	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

It is possible to make duplicate set since the function of each bit is run by the logic of 'OR'. That is, if 0X05 (binary 00000101) is set, both Input Voltage Error and Overheating Error can be detected. If errors occur, in case of Alarm LED, the LED blinks; in case of Alarm Shutdown, the motor output becomes 0 % by making the value of Torque Limit (Address 34, 35) as 0.

RAM Area

Torque Enable

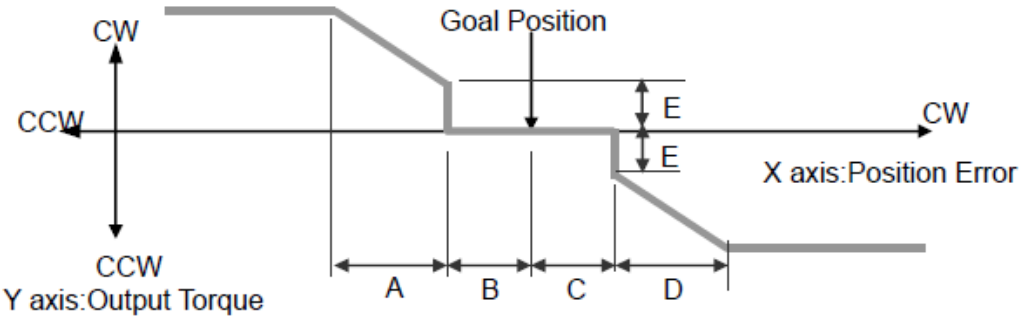
Value	Meaning
0	Keeps Torque from generating by interrupting the power of motor.
1	Generates Torque by impressing the power to the motor.

LED

Bit	Meaning	Meaning
bit7		
bit6		
bit5		
bit4		
bit3		
bit2	BLUE LED	When *Bit* is set the blue LED turns on
bit1	GREEN LED	When *Bit* is set the green LED turns on
bit0	RED LED	When *Bit* is set the red LED turns on

Compliance

Compliance is to set the control flexibility of the motor. The following diagram shows the relationship between output torque and position of the motor.



A : CCW Compliance Slope(Address0x1D)
B : CCW Compliance Margin(Address0x1B)
C : CW Compliance Margin(Address0x1A)
D : CW Compliance Slope (Address0x1C)
E : Punch(Address0x30,31)

Compliance Margin

It exists in each direction of CW/CCW and means the error between goal position and present position. The range of the value is 0~255 and the unit is the same as Goal Position (Address 30, 31). The greater the value, the more difference occurs.

Compliance Slope

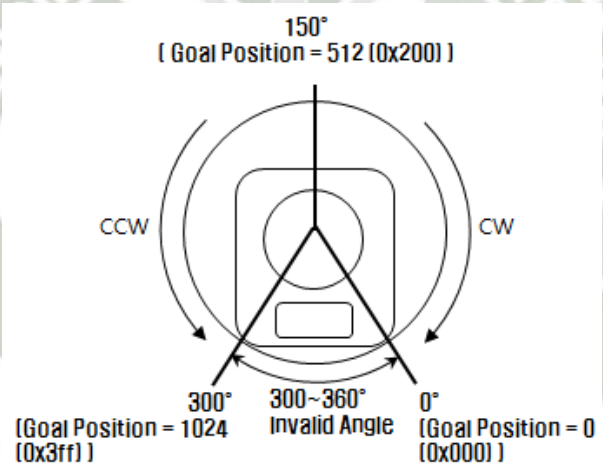
It exists in each direction of CW/CCW and sets the level of Torque near the goal position. Compliance Slope is set in 7 steps, the higher the value, the more flexibility is obtained.

Data representative value is actually used value. That is, even if the value is set to 25, 16 is used internally as the representative value.

Step	Data Value	Data Representative Value
1	0 (0x00) ~ 3(0x03)	2 (0x02)
2	4(0x04) ~ 7(0x07)	4 (0x04)
3	8(0x08)~15(0x0F)	8 (0x08)
4	16(0x10)~31(0x1F)	16 (0x10)
5	32(0x20)~63(0x3F)	32 (0x20)
6	64(0x40)~127(0x7F)	64 (0x40)
7	128(0x80)~254(0xFE)	128 (0x80)

Goal Position

It is a position value of destination. 0 to 1023 (0x3FF) is available. The unit is 0.29 degree. If Goal Position is out of the range, Angle Limit Error Bit (Bit1) of Status Packet is returned as '1' and Alarm is triggered as set in Alarm LED/Shutdown. If it is set to Wheel Mode, this value is not used.



Note: The picture above is based on the front of relevant model.

Moving Speed

It is a moving speed to Goal Position. The range and the unit of the value may vary depending on the operation mode.

- **Join Mode**
  0~1023 (0X3FF) can be used, and the unit is about 0.111rpm. If it is set to 0, it means the maximum rpm of the motor is used without controlling the speed. If it is 1023, it is about 114 RPM. For example, if it is set to 300, it is about 33.3 rpm.
  Notes: Please check the maximum rpm of relevant model in Joint Mode. Even if the motor is set to more than maximum rpm, it cannot generate the torque more than the maximum rpm.

- **Wheel Mode**
  0~2047(0X7FF) can be used, the unit is about 0.1%. If a value in the range of 0~1023 is used, it is stopped by setting to 0 while rotating to CCW direction. If a value in the range of 1024~2047 is used, it is stopped by setting to 1024 while rotating to CW direction. That is, the 10th bit becomes the direction bit to control the direction. In Wheel Mode, only the output control is possible, not speed. For example, if it is set to 512, it means the output is controlled by 50% of the maximum output.

Torque Limit
It is the value of the maximum torque limit. 0 to 1023 (0x3FF) is available, and the unit is about 0.1%. For example, if the value is 512, it is about 50%; that means only 50% of the maximum torque will be used. If the power is turned on, the value of Max Torque (Address 14, 15) is used as the initial value.

Notes: If the function of Alarm Shutdown is triggered, the motor loses its torque because the value becomes 0. At this moment, if the value is changed to the value other than 0, the motor can be used again.

Present Position
It is the current position value of Dynamixel. The range of the value is 0~1023 (0x3FF), and the unit is 0.29 degree. Caution: If it is set to Wheel Mode, the value cannot be used to measure the moving distance and the rotation frequency.

Present Speed
It is the current moving speed. 0~2047 (0X7FF) can be used. If a value is in the rage of 0~1023, it means that the motor rotates to the CCW direction. If a value is in the rage of 1024~2047, it means that the motor rotates to the CW direction. That is, the 10th bit becomes the direction bit to control the direction, and 0 and 1024 are equal.

The unit of this value varies depending on operation mode.

- **Joint Mode**
  The unit is about 0.111rpm. For example, if it is set to 300, it means that the motor is moving to the CCW direction at a rate of about 33.3rpm.

- **Wheel Mode**
  The unit is about 0.1%. For example, if it is set to 512, it means that the torque is controlled by 50% of the maximum torque to the CCW direction.

Present Load
It means currently applied load. The range of the value is 0~2047, and the unit is about 0.1%. If the value is 0~1023, it means the load works to the CCW direction. If the value is

1024~2047, it means the load works to the CW direction. That is, the 10th bit becomes the direction bit to control the direction, and 1024 is equal to 0. For example, the value is 512, it means the load is detected in the direction of CCW about 50% of the maximum torque.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Present Voltage

It is the size of the current voltage supplied. This value is 10 times larger than the actual voltage. For example, when 10V is supplied, the data value is 100 (0x64).

Present Temperature

It is the internal temperature of Dynamixel in Celsius. Data value is identical to the actual temperature in Celsius. For example, if the data value is 85 (0x55), the current internal temperature is 85°C.

Registered Instruction

Value	Meaning
0	There are no commands transmitted by REG_WRITE
1	There are commands transmitted by REG_WRITE.

Notes: If ACTION command is executed, the value is changed into 0.

Moving

Value	Meaning
0	Goal position command execution is completed.
1	Goal position command execution is in progress.

Lock

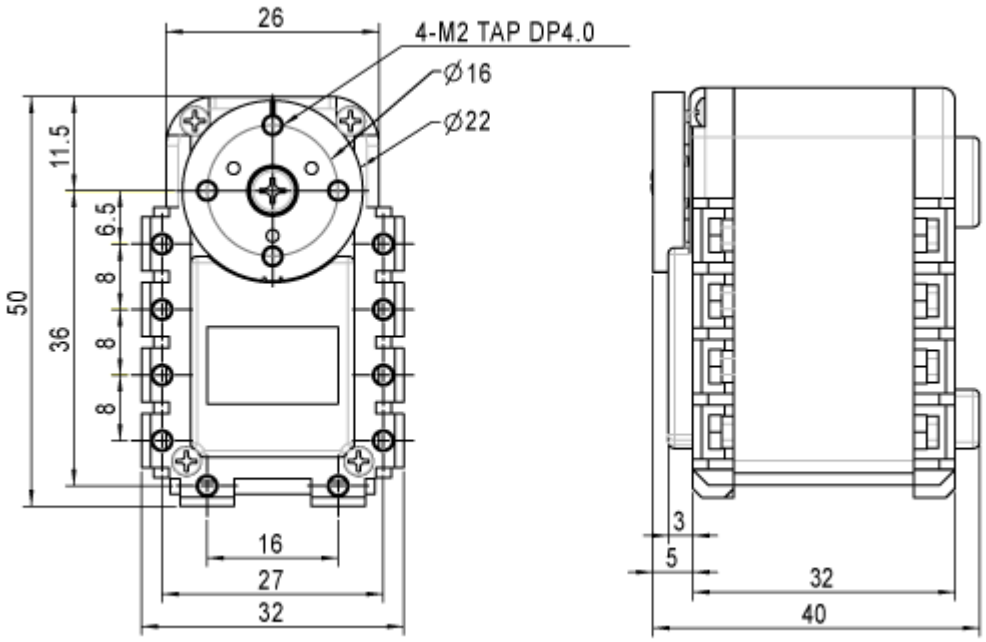
Value	Meaning
0	EEPROM area can be modified.
1	EEPROM area cannot be modified.

Caution: If Lock is set to 1, the power must be turned off and then turned on again to change into 0.

Punch

Current to drive motor is at minimum. Can choose vales from 0x20 to 0x3FF.

Dimension



MX-28T/MX-64T

Parts Photo

[MX-28T] **[MX-64T]**

Hardware Specifications

- MCU : ST CORTEX-M3 (STM32F103C8 @ 72MHZ,32BIT)
- POSITION SENSOR : Contactless absolute encoder (12BIT,360 DEGREE)
- MOTOR : Maxon
- BAUD RATE : 8000 bps ~ 4.5 Mbps
- CONTROL ALGORITHM : PID CONTROL
- Resolution : 0.088°
- Running Degree :
  - 0° ~ 360°
  - Endless Turn
- Weight : 72g (MX-28T), 126g (MX-64T)
- Dimension : 35.6mm x 50.6mm x 35.5mm (MX-28T), 40.2mm x 61.1mm x 41mm (MX-64T)
- Gear Reduction Ratio : 193 : 1 (MX-28T), 200:1 (MX-64T)
- Stall Torque
  - 2.3N.m at 11.1V, 1.3A (MX-28T), 5.5N.m at 11.1V, 3.9A (MX-64T).
  - 2.5N.m at 12V, 1.4A (MX-28T), 6.0N.m at 12V, 4.1A (MX-64T).
  - 3.1N.m at 14.8V, 1.7A (MX-28T), 7.3N.m at 14.8V, 5.2A (MX-64T).
- No load speed
  - 50rpm at 11.1V (MX-28T), 58rpm at 11.1V (MX-64T),
  - 55rpm at 12V (MX-28T), 63rpm at 12V (MX-64T),
  - 67rpm at 14.8V (MX-28T), 78rpm at 14.8V (MX-64T),
- Running Temperature : -5℃ ~ +80℃
- Voltage : 10 ~ 14.8V (Recommended Voltage 12V)
- Command Signal : Digital Packet

- Protocol Type: Half duplex Asynchronous Serial Communication (8bit,1stop, No Parity)
- Link (Physical): TTL Level Multi Drop Bus
- ID : 254 ID (0~253)
- Feedback: Position, Temperature, Load, Input Voltage, etc.
- Material : Full Metal Gear, Engineering Plastic Body
- Standby current : 100 mA

Stall torque is the maximum instantaneous and static torque. Stable motions are possible with robots designed for loads with 1/5 or less of the stall torque.

Precautions when connecting to power supply!

- For the stable power supply, we recommend using ROBOTIS controller or SMPS2Dynamixel.
- Connect your DYNAMIXEL to power supply while it's off and turn on/off with the power switch.

Control Table

Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet.

EEPROM and RAM
Data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.

Address
It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.

Access
Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.

Initial Value
In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.

Highest/Lowest Byte

In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16bit, but it is divided into 8bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	29 (0X1D)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	34 (0X22)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	15 (0X0F)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	160 (0XA0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36 (0X24)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
	20 (0X14)	Multi Turn Offset(L)	multi-turn offset least significant byte (LSB)	RW	0 (0X00)
	21 (0X12)	Multi Turn Offset(H)	multi-turn offset most significant byte (MSB)	RW	0 (0X00)
	22 (0X12)	Resolution Divider	Resolution divider	RW	1 (0X01)
	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	D Gain	Derivative Gain	RW	0 (0X00)
	27 (0X1B)	I Gain	Integral Gain	RW	0 (0X00)

RAM	28 (0X1C)	P Gain	Proportional Gain	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	0 (0X00)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)
	68 (0X44)*	Current(L)	Lowest byte of Consuming Current	RW	0 (0X00)
	69 (0X45)*	Current(H)	Highest byte of Consuming Current	RW	0 (0X00)
	70 (0X46)*	Torque Control Mode Enable	Torque control mode on/off	RW	0 (0X00)
	71 (0X47)*	Goal Torque(L)	Lowest byte of goal torque value	RW	0 (0X00)
	72 (0X48)*	Goal Torque(H)	Highest byte of goal torque value	RW	0 (0X00)
	73 (0X49)	Goal Acceleration	Goal Acceleration	RW	0 (0X00)

* Only with MX-64T.

Address Function Help

EEPROM Area

Model Number

It represents the Model Number

Firmware Version

It represents the firmware version

ID

It is a unique number to identify Dynamixel. The range from 0 to 252 (0xFC) can be used, and, especially, 254(0xFE) is used as the Broadcast ID. If the Broadcast ID is used to transmit Instruction Packet, we can command to all Dynamixels. Please be careful not to duplicate the ID of connected Dynamixel.

Baud Rate

It is the baud rate to communicate with controller. It is available in between 0~254(0XFE). If the data value is in between 0~249:

Baud rate(BPS) = 2000000 / (Data + 1)

Data	Set BPS	Target BPS	Tolerance
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

If the date value is over the 250:

Data	Set BPS	Target BPS	Tolerance
250	2250000.0	2250000.0	0.000 %
251	2500000.0	2500000.0	0.000 %
252	3000000.0	3000000.0	0.000 %

Note: Maximum Baud Rate error of 3% is within the tolerance of UART communication.

Return Delay Time

It is the delay time per data value that takes from the transmission of Instruction Packet until the return of Status Packet. 0 to 254 (0xFE) can be used, and the delay time per data value is 2 µs. That is to say, if the data value is 10, 20 µs is delayed. The initial value is 250 (0xFA) (i.e., 0.5 msec).

CW/CCW Angle Limit

Sets allowable position values (angles) for Goal Position (address 30 & 31)

- **CW Angle Limit:** Goal Position(Address 30, 31) minimum value.
- **CCW Angle Limit:** Goal Position(Address 30, 31) maximum value.

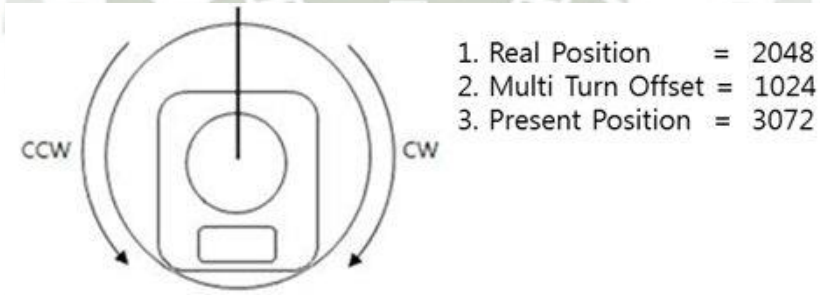
The following sets 2 modes operation based on CW and CCW values:

Operation Type	CW / CCW
Wheel Mode	both are 0
Joint Mode	neither at 0
Multi-turn Mode	both are 4095

Wheel mode allows the motor can have limitless revolutions. Joint mode allows robot with multiple joints. Multi-turn mode allows joints have range of controllable position values from -28672 to 28672.

Multi Turn Offset

Adjusts position (zeroing). This value gets included in Present Position (36). Present position + multi-turn offset. Initial value is 0 and range is from -24576 to 24576. A Dynamixel with a position of 2048 with an applied offset of 1024 outputs a Present position of 3072.



Note: This feature is only applied in multi-turn mode and ignored in other modes.

Resolution Divider

It allows the user to change Dynamixel's resolution. The default Resolution Divider Value is set as 1. (1 ~ 4 available). When resolution is lowered, revolutions (in both directions) can be increased (up to 28 turns in each direction).

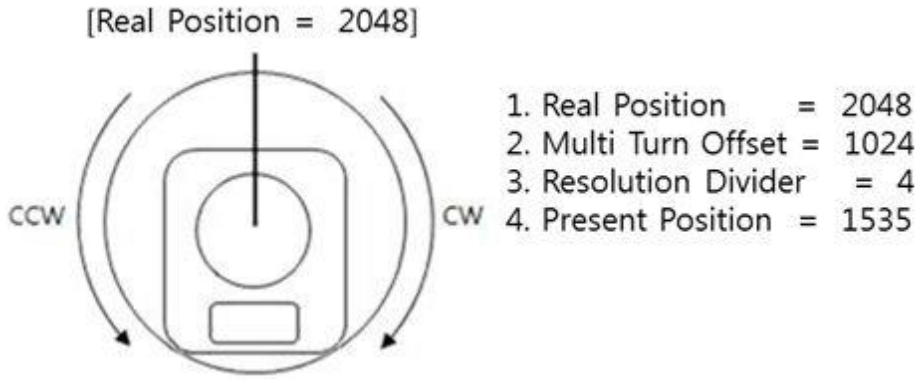
Present Position = Real Position / Resolution Divider

For example, a Real Position of 2048 with a Resolution Divider set as 2 will yield a Present Position value of 1024 (2048/2 = 1024). A Dynamixel with Resolution Divider set as 2 will have a resolution 2048 for a single revolution.

The Present Position can be obtained while Multi-turn Offset and Resolution Divider are taken into account.

Present position = (Real Position / Resolution Divider) + Multi-turn Offset

For example, a Dynamixel with a Real Position of 2048 with a Resolution Divider set as 4 and Multi-turn Offset as 1024 will yield a Present Position of 1535 ((2048/4) + 1024 = 1535).



Note: This feature is only applied in multi-turn mode and ignored in other modes.

The Highest Limit Temperature

Caution: Do not set the temperature lower/higher than the default value. When the temperature alarm shutdown occurs, wait 20 minutes to cool the temperature before re-use. Using the product when the temperature is high may and can cause damage.

The Lowest (Highest) Limit Voltage

It is the operation range of voltage. 50 to 250 (0x32 ~ 0x96) can be used. The unit is 0.1V. For example, if the value is 80, it is 8V. If Present Voltage (Address42) is out of the range, Voltage Range Error Bit (Bit0) of Status Packet is returned as '1' and Alarm is triggered as set in the addresses 17 and 18.

Max Torque

It is the torque value of maximum output. 0 to 1023 (0x3FF) can be used, and the unit is about 0.1%. For example, Data 1023 (0x3FF) means that Dynamixel will use 100% of the maximum torque it can produce while Data 512 (0x200) means that Dynamixel will use 50% of the maximum torque. When the power is turned on, Torque Limit (Addresses 34 and 35) uses the value as the initial value.

Status Return Level

It decides how to return Status Packet. There are three ways like the below table.

Value	Return of Status Packet
0	No return against all commands (Except PING Command)
1	Return only for the READ command
2	Return for all commands

When Instruction Packet is Broadcast ID, Status Packet is not returned regardless of Status Return Level.

Alarm LED

Alarm Shutdown

Dynamixel can protect itself by detecting errors occur during the operation. The errors can be set are as the table below.

Bit	Name	Contents
Bit 7	0	-
Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	CheckSum Error	When the Checksum of the transmitted Instruction Packet is invalid
Bit 3	Range Error	When the command is given beyond the range of usage
Bit 2	OverHeating Error	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Error	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

It is possible to make duplicate set since the function of each bit is run by the logic of 'OR'. That is, if 0X05 (binary 00000101) is set, both Input Voltage Error and Overheating Error can be detected. If errors occur, in case of Alarm LED, the LED blinks; in case of Alarm Shutdown, the motor output becomes 0 % by making the value of Torque Limit(Address 34, 35) as 0.

RAM Area

Torque Enable

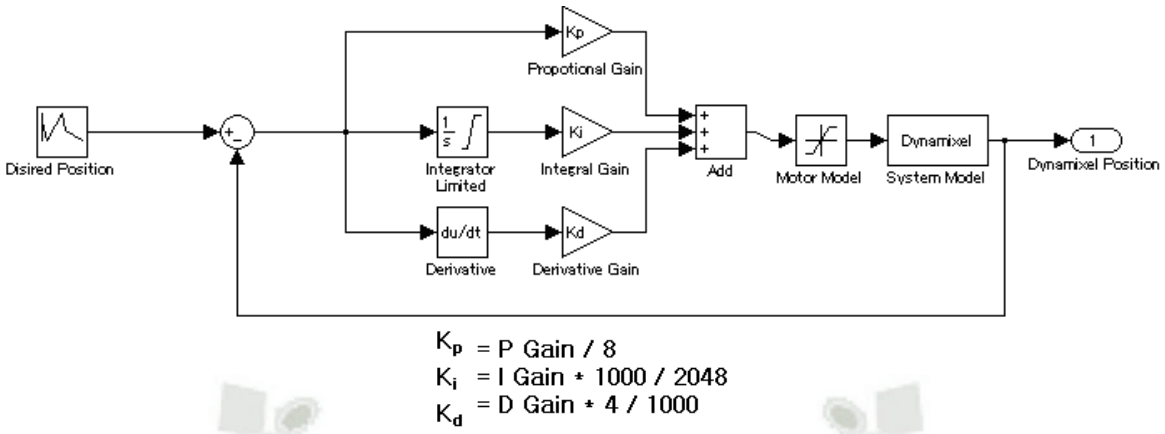
Value	Meaning
0	Keeps Torque from generating by interrupting the power of motor.
1	Generates Torque by impressing the power to the motor.

LED

Bit	Meaning	Meaning
bit7		
bit6		
bit5		
bit4		
bit3		
bit2	BLUE LED	When *Bit* is set the blue LED turns on
bit1	GREEN LED	When *Bit* is set the green LED turns on
bit0	RED LED	When *Bit* is set the red LED turns on

PID Gain

MX series will use the PID controller as a main control method. P gain refers to the value of proportional band. I gain refers to the value of integral action. D Gain refers to the value of derivative action. Gains values are in between 0~254.



K_p = P Gain / 8
K_i = I Gain * 1000 / 2048
K_d = D Gain * 4 / 1000

The relationship between Compliance Slop and PID

Slope	P Gain
8	128
16	64
32	32
64	16
128	8

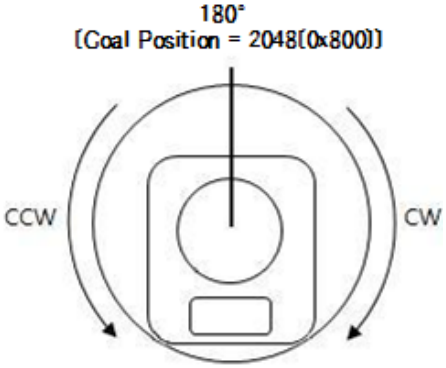
The less the P gain, the larger the back lash, and the weaker the amount of output near goal position. At some extent, it is like a combined concept of margin and slope. It does not exactly match the previous concept of compliance. So it is obvious if you see the difference in terms of motion.

Explanation for PID required

For the brief explanation about general PID, please refer to the website(link) below. http://en.wikipedia.org/wiki/PID_controller. FYI, PID control theory is not only limited to the control of motor(actuator) but is a generic theory that can be applied to all kinds of control.

Goal Position

It is a position value of destination. 0 to 4095 (0xFFF) is available. The unit is 0.088 degree. If Goal Position is out of the range, Angle Limit Error Bit (Bit1) of Status Packet is returned as '1' and Alarm is triggered as set in Alarm LED/Shutdown.



Moving Speed

- **Join Mode (Multi-Turn mode)**
  It is a moving speed to Goal Position. 0~1023 (0X3FF) can be used, and the unit is about 0.114rpm. If it is set to 0, it means the maximum rpm of the motor is used without controlling the speed. If it is 1023, it is about 117.07rpm. For example, if it is set to 300, it is about 34.33 rpm.

- **Wheel Mode**
  It is a moving speed to Goal direction. 0~2047 (0X7FF) can be used, and the unit is about 0.114rpm. If a value in the range of 0~1023 is used, it is stopped by setting to 0 while rotating to CCW direction. If a value in the range of 1024~2047 is used, it is stopped by setting to 1024 while rotating to CW direction. That is, the 10th bit becomes the direction bit to control the direction.

Note: This mode allows to check max rpm. Any values set higher than max rpm will not take effect.

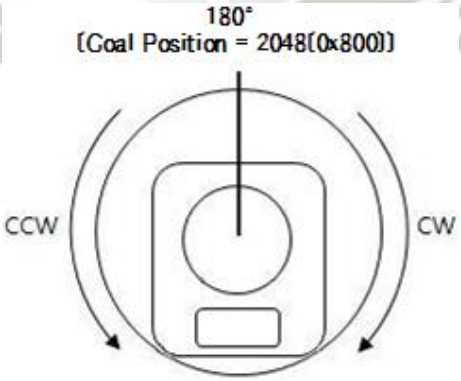
Torque Limit

It is the value of the maximum torque limit. 0 to 1023 (0x3FF) is available, and the unit is about 0.1%. For example, if the value is 512, it is about 50%; that means only 50% of the maximum torque will be used. If the power is turned on, the value of Max Torque (Address 14, 15) is used as the initial value.

Notes: If the function of Alarm Shutdown is triggered, the motor loses its torque because the value becomes 0. At this moment, if the value is changed to the value other than 0, the motor can be used again.

Present Position

It is the current position value of Dynamixel. The range of the value is 0~4095 (0xFFF), and the unit is 0.088 degree.



In multi-turn mode range is from -28672 to 28672 with unit values dependent on Resolution Divider (0.088 * Resolution Divider)

Note: in multi-turn mode, Present position depends on resolution divider and multi-turn offset. For more information turn to the section on Multi Turn offset and Resolution Divider

Present Speed

Is the current moving speed. 0~2047 (0x000~0X7FF) can be used. If a value is in the rage of 0~1023 then the motor rotates to the CCW direction. If a value is in the rage of 1024~2047 then the motor rotates to the CW direction. The 10th bit becomes the direction bit to control the direction; 0 and 1024 are equal. The value unit is about 0.11rpm. For example, if it is set to 300 then the motor is moving to the CCW direction at a rate of about 34.33rpm.

Present Load

It means currently applied load. The range of the value is 0~2047, and the unit is about 0.1%. If the value is 0~1023, it means the load works to the CCW direction. If the value is 1024~2047, it means the load works to the CW direction. That is, the 10th bit becomes the direction bit to control the direction, and 1024 is equal to 0. For example, the value is 512, it means the load is detected in the direction of CCW about 50% of the maximum torque.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Notes: Current load is inferred from the internal torque value, not from Torque sensor etc. For that reason, it cannot be used to measure weight or torque; however, it must be used only to detect which direction the force works.

Present Voltage

It is the size of the current voltage supplied. This value is 10 times larger than the actual voltage. For example, when 10V is supplied, the data value is 100 (0x64)

Present Temperature

It is the internal temperature of Dynamixel in Celsius. Data value is identical to the actual temperature in Celsius. For example, if the data value is 85 (0x55), the current internal temperature is 85℃.

Registered Instruction

Value	Meaning
0	There are no commands transmitted by REG_WRITE
1	There are commands transmitted by REG_WRITE.

Notes: If ACTION command is executed, the value is changed into 0.

Moving

Value	Meaning
0	Goal position command execution is completed.
1	Goal position command execution is in progress.

Lock

Value	Meaning
0	EEPROM area can be modified.
1	EEPROM area cannot be modified.

Caution: If Lock is set to 1, the power must be turned off and then turned on again to change into 0.

Punch

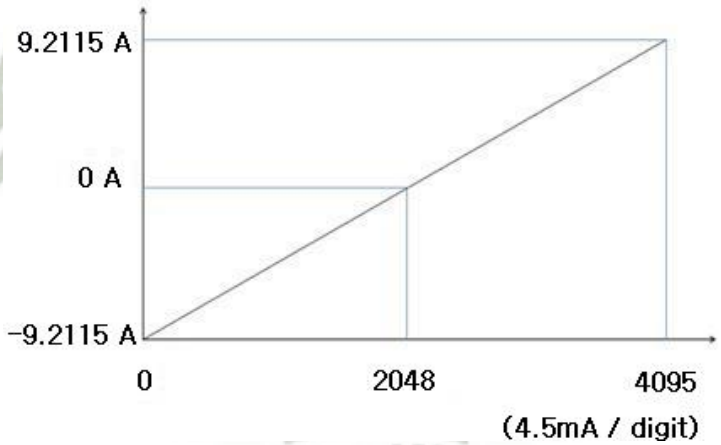
Current to drive motor is at minimum. Can choose vales from 0x00 to 0x3FF.

Current

Value at 2048(0x800) when current is consumption is idle. Values higher than 2048(0x800) during positive current flow. Values lower than 2048(0x800) during negative current flow. The following is a method to calculate current flow:

I = (4.5mA) * (CURRENT – 2048) in amps unit (A).

For example, 68 gives a value of 2148, which corresponds to 450mA of current flow.



Torque Control Mode Enable

Value	Meaning
0	Turn off the torque mode. Executes Joint mode or Wheel mode.
1	Turn on the torque mode. Cannot control the position or moving speed but only Torque.

When Torque Control Mode Enable is at 1, DYNAMIXEL behaves like the followings:

1. DYNAMIXEL does not control the position or the moving speed.
2. DYNAMIXEL controls with goal torque value.
3. DYNAMIXEL does not react to whatever value in Goal position and Goal speed.

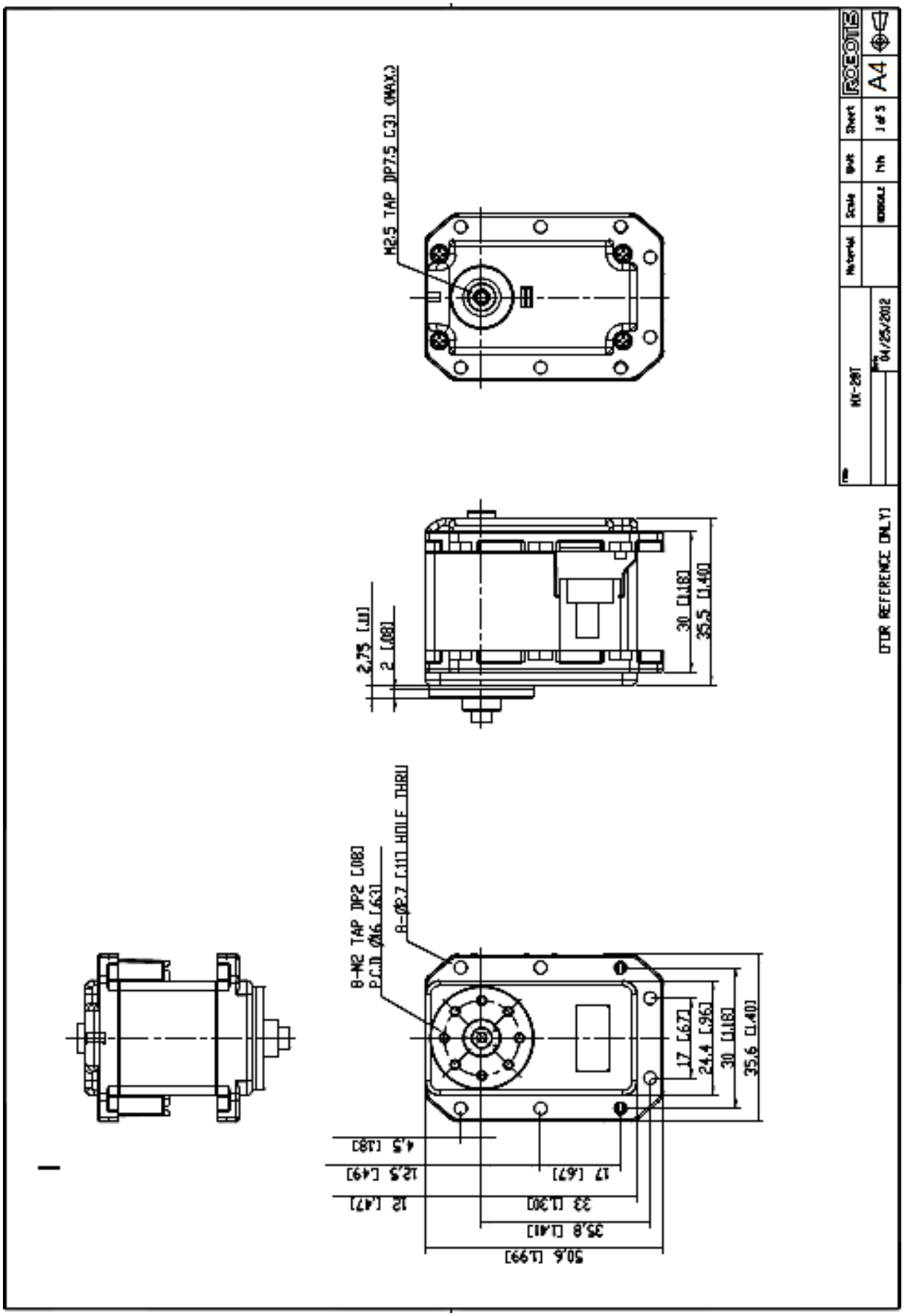
4. Because position/moving speed is not controller, DYNAMIXEL behaves as if it is in the wheel mode.

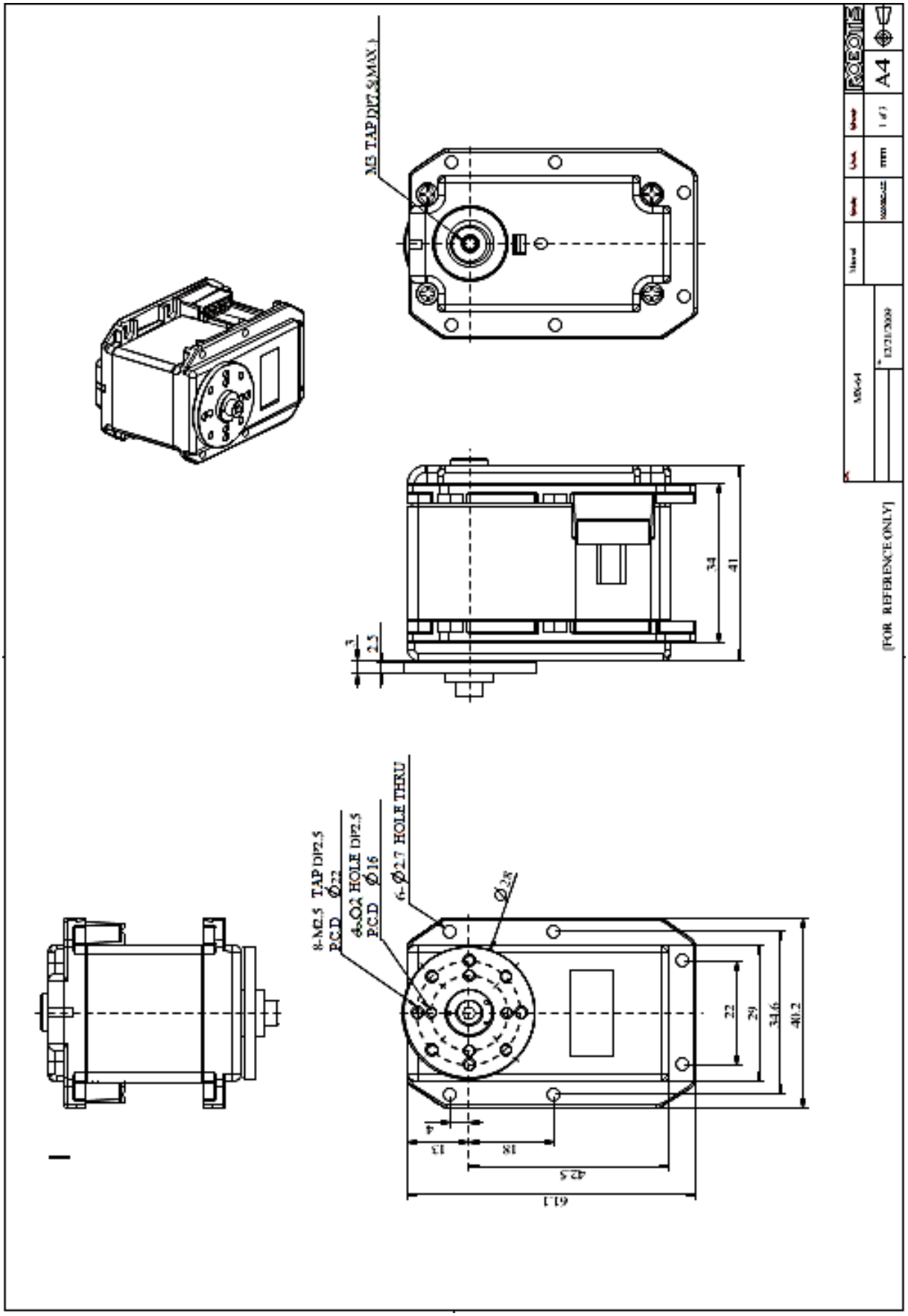
Goal Torque

These are the goal torque value. You can use 0 ~ 2047 (0x7FF), and the unit is 4.5mA (torque is directly proportional to the current value). If you use from 0~1023, torque is on toward CCW, and when you set it to 0, it stops. If you use from 1024~2047, torque is on toward CW, and when you set it to 1024, it stops. That means, 10^{th} bit becomes the direction bit, which controls direction. Goal Torque cannot be bigger than Torque Limit(34,35).

Goal Acceleration

This is Goal Acceleration value. It can be used from 0~254(0XFE), and the unit is approximately 8.583 Degree / sec^2. When it is set to 0, there is no control over acceleration and moves with the maximum acceleration of the motor. When the goal speed is set to 0, there is no control over acceleration and moves with the maximum acceleration of the motor. When it is set to 254, it becomes 2180 Degree / sec^2. For example, the current speed of Dynamixel is 0, and Goal acceleration is 10. The speed of Dynamixel after 1 second will be 14.3 RPM.





Anexo D: Datasheet Controlador USB2Dynamixel

Part Photo

[USB2Dynamixel]

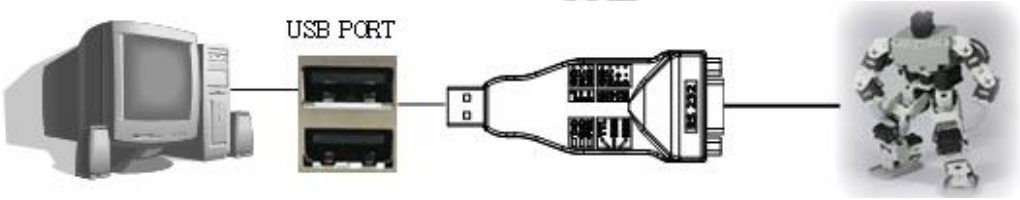
Product Usage

USB2Dynamixel is a device used to operate Dynamixel directly from PC. USB2Dynamixel is connected to USB port of PC, and 3P and 4P connectors are installed so that various Dynamixels can be connected.

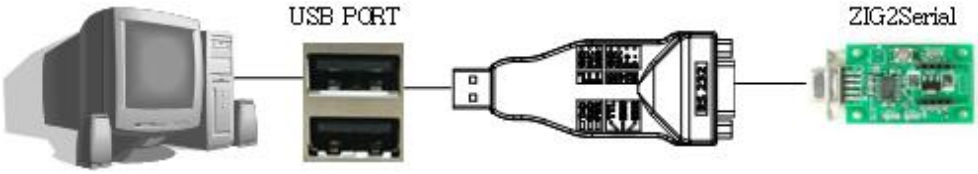
Also, USB2Dynamixel can be used to change from USB port to Serial port on the PC without serial port such as notebook computer, etc. The function is very useful in the cases when the Dynamixel exclusive controllers such as CM-2, CM-2+, CM-5, and CM-510 are connected to USB Port, or when ZIG2Serial is connected to USB port to control robots wirelessly.

- **Dynamixel Control Using PC**

- **Changing Serial Port**

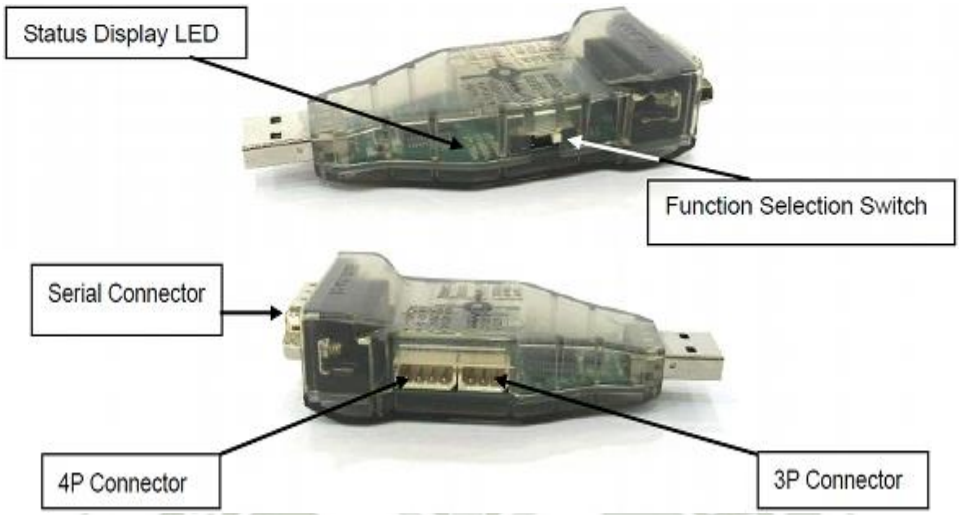


- **Wireless Communication**



Difference in voltage may cause unstable USB2DXL connections. Ensure that both connecting equipment and PC are properly grounded.

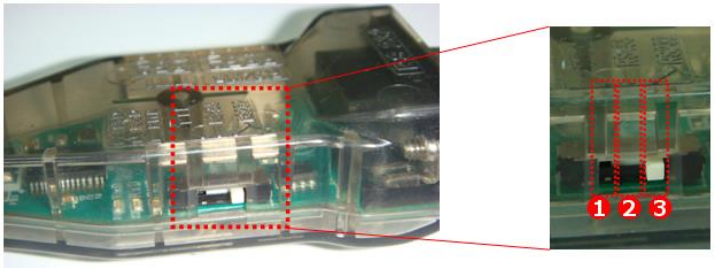
Name and Usage of Each Part



Name	Description
Status Display LED	Display power supply, TXD(data writing), and RXD(data reading) status.
Function Selection Switch	Select the communication method of TTL, RS-485, and RS-232.
3P Connector	Connect Dynamixels of AX Series through TTL communication.
4P Connector	Connect Dynamixels of DX, RX Series through RS-485 communication.
Serial Connector	Change from USB port to Serial port through RS-232 communication.

How to Select the Communication Mode

The communication mode can be selected by changing the switch of USB2Dynamixel as below.



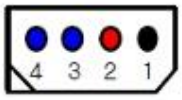
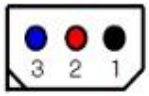
1. TTL Communication: Dynamixels using 3-pin port such as AX Series, AX-S1 etc.
2. RS485 Communication: Dynamixels using 4-pin port such as DX Series, RX Series, EX Series etc.
3. RS232 Communication: Controllers using serial cable such as CM-5, CM-510 etc.

Usage

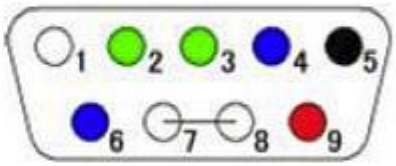
Dynamixel can be controlled directly by C language using USB2Dynamixel SDK.

PIN Figure

The following pictures show the usage of connector pins used by USB2Dynamixel. If you want to use each pin for your own purpose, please use them after you learn the usage of each pin.

4 Pin Cable			3 Pin Cable		
Pin No.	Signal	Pin Figure	Pin No.	Signal	Pin Figure
1	GND		1	GND	
2	NOT Connected		2	NOT Connected	
3	DATA + (RS-485)		3	DATA (TTL)	
4	DATA - (RS-485)				

[PIN Figure of 4P / 3P Cable Connectors]

Pin No.	Signal	Pin Figure
1	Data (TTL)	
2	RXD (RS-232)	
3	TXD (RS-232)	
4	D+ (RS-485)	
5	GND	
6	D- (RS-485)	
7	Short with No. 8	
8	Short with No. 7	
9	USB power (5V)	

[PIN Figure of Serial Connector]

How to Supply Power

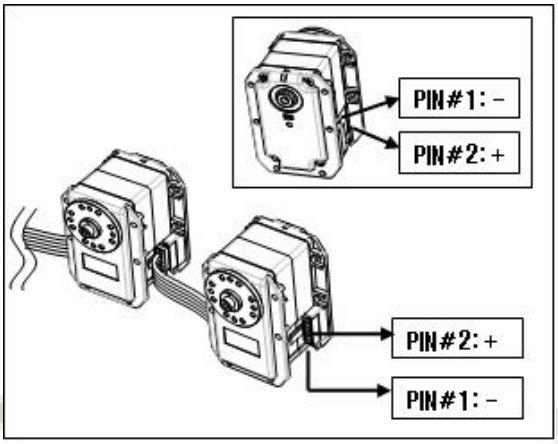
USB2Dynamixel does not supply power to Dynamixel. Therefore, the power must be supplied separately to operate Dynamixel as the following pictures. Please refer to the explanation page of each Dynamixel regarding proper voltages for each Dynamixel mode.

Precautions when connecting to power supply!

- For the stable power supply, we recommend using ROBOTIS controller or SMPS2Dynamixel.
- Connect your DYNAMIXEL to power supply while it's off and turn on/off with the power switch

Referring to PIN Figure, apply positive (+) voltage to the #2 PIN of the connector, and negative (-) voltage on #1 PIN of the connector.

Since the 2 connectors of Dynamixels are equal, power can be applied at any of.

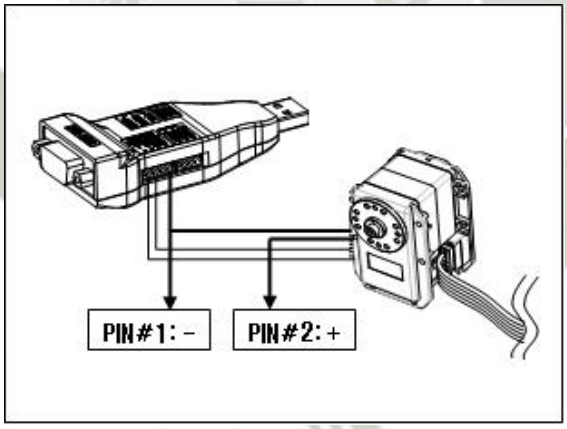


[Applying Power to the Dynamixel]

If power cannot be applied as above, apply the power between USB2Dynamixel and Dynamixel as below.

Separate the power cable on the #2 PIN of USB2Dynamixel connector, and then apply positive (+) voltage.

Connect additionally Y-cable to the power cable on the #1 PIN of USB2Dynamixel connector, and then apply negative (-) voltage.



[Applying power between USB2Dynamixel and Dynamixel]

Anexo E: Scripts elaborados

```
clear all; close all; clc
%% Inicializacion
global a1 a2 a3 m1 m2 m3 g yc1 xc2 xc3
global a1_est a2_est a3_est m1_est m2_est m3_est
global kp1 kp2 kp3 ki1 ki2 ki3 kd1 kd2 kd3
%Parámetros del robot
a1=0.167;        %Longitud Eslabón 1
a2=0.181;        %Longitud Eslabón 2
a3=0.178;        %Longitud Eslabón 3
m1=0.624;        %Masa Eslabón 1
m2=0.759;        %Masa Eslabón 2
m3=0.303;        %Masa Eslabón 3
g=9.81;          %gravedad
yc1=0.101;       %Distancia al C.G. del Eslabón 1
xc2=0.129;       %Distancia al C.G. del Eslabón 2
xc3=0.053;       %Distancia al C.G. del Eslabón 3

%Parámetros estimados
a1_est=0.17;         %Longitud Eslabón 1
a2_est=0.181;        %Longitud Eslabón 2
a3_est=0.178;        %Longitud Eslabón 3
m1_est=0.624;        %Masa Eslabón 1
m2_est=0.78;         %Masa Eslabón 2
m3_est=0.33;         %Masa Eslabón 3

%Momentos de inercia calculados en el controlador
I1=[1/12*m1_est*a1_est^2 0 0;
    0 1/12*m1_est*a1_est^2 0;
    0 0 1/12*m1_est*a1_est^2];   %Tensor de Inercia 1
I2=[2/9*m2_est*a2_est^2 0 0;
    0 2/9*m2_est*a2_est^2 0;
    0 0 -1/34*m2_est*a2_est^2];  %Tensor de Inercia 2
I3=[1/5*m3_est*a3_est^2 0 0;
    0 1/44*m3_est*a3_est^2 0;
    0 0 1/5*m3_est*a3_est^2];    %Tensor de Inercia 3

% Sintonizacion de ganancias PID
kp1=580; kp2=165; kp3=900;       %Ganancias proporcionales
ki1=100; ki2=0.1; ki3=0.1;       %Ganancias integrales
kd1=270; kd2=850; kd3=600;       %Ganancias derivativas

%Condiciones del eslabón cero
w_0=[0 0 0]'; w_prime_0=[0 0 0]'; v_0=[0 0 0]'; a_0=[0 0 0]';
%Condiciones iniciales de fuerza en el efector final
f_34=[0 0 0]'; N_34=[0 0 0]'; g_v=[0 0 -g]';

%Estado inicial
p_0=Trayectoria(0);
%Cinemática inversa:
q_0=CiInv(p_0(:,1));
q0 = [q_0' 0 0 0]';

%Tiempo de simulación, vectores
tf = 1; dt=0.006; t=0:dt:tf;
```

```
q=zeros(length(t),6); q(1,:)=q0';
Tau_Control=zeros(length(t)-1,3);

%Inicialización de matrices de ganancias PID y error
Kp=diag([kp1 kp2 kp3]);
Ki=diag([ki1 ki2 ki3]);
Kd=diag([kd1 kd2 kd3]);
e1=zeros(3,1);
de1=zeros(3,1);
ie1=zeros(3,1);

%% Simulación
for i=2:length(t)
    %Grados de libertad y derivadas
    %Vectores de Ángulos
    noise=0;
    theta=q(i-1,1:3)';
    dtheta=q(i-1,4:6)';
    theta_est=q(i-1,1:3)'+noise*randn(3,1);
    dtheta_est=q(i-1,4:6)'+noise*randn(3,1);

    %Matrices del robot
    H=HH(theta);
    C=CC([theta;dtheta]);
    G=GG(theta);

    %Cálculos que deben efectuarse en el controlador
    %================================================
    %Cinemática directa:
    p=INTJkk_est(theta_est);

    %Valores deseados
    %Trayectoria deseada:
    p_d=Trayectoria(t(i-1)); v_d=p_d(:,2); p_d=p_d(:,1);
    %Cinemática inversa:
    theta_d=CiInv_est(p_d);
    %Inversa del Jacobiano
    J_inv=INVJacob_est(theta_est);
    dtheta_d=J_inv*v_d;

    % Control PID+G
    e1=theta_d-theta_est;                %Error proporcional
    ie1=ie1+e1;                          %Error integrativo
    de1=dtheta_d-dtheta_est;             %Error derivativo
    u=Kp*e1+Ki*ie1+Kd*de1+GG(theta_est);  %Control PID+G

    %Cinemática
    T01=[cos(theta_est(1)) 0 sin(theta_est(1)) 0;
         sin(theta_est(1)) 0 -cos(theta_est(1)) 0;
         0 1 0 a1_est;
         0 0 0 1];
    T12=[cos(theta_est(2)) -sin(theta_est(2)) 0 a2_est*cos(theta_est(2));
         sin(theta_est(2)) cos(theta_est(2)) 0 a2_est*sin(theta_est(2));
         0 0 1 0;
         0 0 0 1];
    T23=[cos(theta_est(3)) -sin(theta_est(3)) 0 a3_est*cos(theta_est(3));
```

```
        sin(theta_est(3)) cos(theta_est(3)) 0 a3_est*sin(theta_est(3));
        0 0 1 0;
        0 0 0 1];
    b_0=[0 0 1]';
    b_1=T01(1:3,3);
    b_2=(T01*T12); b_2=b_2(1:3,3);
    r_01=T01(1:3,4);
    r_12=T01(1:3,1:3)*T12(1:3,4);
    r_23=T01(1:3,1:3)*T12(1:3,1:3)*T23(1:3,4);
    r_c1=[0 -yc1 0]'; r_c1=T01(1:3,1:3)*r_c1;
    r_c2=[-xc2 0 0]'; r_c2=T01(1:3,1:3)*T12(1:3,1:3)*r_c2;
    r_c3=[-xc3 0 0]'; r_c3=T01(1:3,1:3)*T12(1:3,1:3)*T23(1:3,1:3)*r_c3;

    %Cinemática
    %Primer eslabón:
    w_1=w_0+dtheta_est(1)*b_0;
    w_prime_1=w_prime_0+u(1)*b_0+cross(w_0,dtheta_est(1)*b_0);
    v_1=v_0+cross(w_1,r_01);
    a_1=a_0+cross(w_prime_1,r_01)+cross(w_1,cross(w_1,r_01));
    v_c1=v_0+cross(w_1,r_01+r_c1);
    a_c1=a_0+cross(w_prime_1,r_01+r_c1)+cross(w_1,cross(w_1,r_01+r_c1));
    %Segundo eslabón:
    w_2=w_1+dtheta_est(2)*b_1;
    w_prime_2=w_prime_1+u(2)*b_1+cross(w_1,dtheta_est(2)*b_1);
    v_2=v_1+cross(w_2,r_12);
    a_2=a_1+cross(w_prime_2,r_12)+cross(w_2,cross(w_2,r_12));
    v_c2=v_1+cross(w_2,r_12+r_c2);
    a_c2=a_1+cross(w_prime_2,r_12+r_c2)+cross(w_2,cross(w_2,r_12+r_c2));
    %Tercer eslabón:
    w_3=w_2+dtheta_est(3)*b_2;
    w_prime_3=w_prime_2+u(3)*b_2+cross(w_2,dtheta_est(3)*b_2);
    v_3=v_2+cross(w_3,r_23);
    a_3=a_2+cross(w_prime_3,r_23)+cross(w_3,cross(w_3,r_23));
    v_c3=v_2+cross(w_3,r_23+r_c3);
    a_c3=a_2+cross(w_prime_3,r_23+r_c3)+cross(w_3,cross(w_3,r_23+r_c3));

    %Dinámica
    %Tercer eslabón

I33=(T01(1:3,1:3)*T12(1:3,1:3)*T23(1:3,1:3))*I3*(T01(1:3,1:3)*T12(1:3,1:3)*T
23(1:3,1:3))';
    f_23=f_34-m3_est*g_v+m3_est*a_c3;
    N_23=N_34-
cross(r_c3,f_34)+cross(r_23+r_c3,f_23)+I33*w_prime_3+cross(w_3,I33*w_3);
    %Segundo eslabón
    I22=(T01(1:3,1:3)*T12(1:3,1:3))*I2*(T01(1:3,1:3)*T12(1:3,1:3))';
    f_12=f_23-m2_est*g_v+m2_est*a_c2;
    N_12=N_23-
cross(r_c2,f_23)+cross(r_12+r_c2,f_12)+I22*w_prime_2+cross(w_2,I22*w_2);
    %Primer eslabón
    I11=(T01(1:3,1:3))*I1*(T01(1:3,1:3))';
    f_01=f_12-m1_est*g_v+m1_est*a_c1;
    N_01=N_12-
cross(r_c1,f_12)+cross(r_01+r_c1,f_01)+I11*w_prime_1+cross(w_1,I11*w_1);

    %Torques
    tau=zeros(3,1);
```

```
    tau(1)=dot(b_0,N_01);
    tau(2)=dot(b_1,N_12);
    tau(3)=dot(b_2,N_23);

    %Fin del PID con Compensación de Gravedad
    %=========================================

    %Simulación de dinámica
    d2q=H^-1*(tau-C-G);
    qpunto=[dtheta;d2q];
    q(i,:)=q(i-1,:)+(qpunto*(t(i)-t(i-1)))';
    Tau_Control(i-1,:)=tau';
end

%% Presentación de Resultados
%Posiciones del robot
%Referencia
xref=zeros(size(q(:,1:3)));
for i=1:length(t)
    x_i=Trayectoria(t(i));
    xref(i,:)=x_i(:,1)';
end
%Trayectoria
xreal=zeros(size(q(:,1:3)));
for i=1:length(t)
    xreal(i,:)=INTJkk(q(i,1:3));
end

%Gráfico 1: Trayectoria
figure,
plot3(xref(:,1),xref(:,2),xref(:,3),xreal(:,1),xreal(:,2),xreal(:,3))
grid, axis([0 0.35 -0.3 0.3 0 0.4]), view(52.5, 30)
legend('Referencia', 'Trayectoria Real')
xlabel('x'), ylabel('y'), zlabel('z')
title('Trayectoria')
%Gráfico 2: Torque de Control
Tau_Control=[Tau_Control(1,:);Tau_Control];
figure, plot(t,Tau_Control), legend('Torque 1','Torque 2','Torque 3')
grid, xlabel('t (s)'), ylabel('Torque (N.m)'), title('Torques de Control')
%Grafico 3: Trayectorias de cada articulación
figure, subplot(3,2,1), plot(t,q(:,1).*180/pi)
grid, xlabel('t (s)'), ylabel('q1 (°)'), title('Posición q1')
subplot(3,2,2), plot(t,q(:,4).*(180/(6.*pi)))
grid, xlabel('t (s)'), ylabel('dq1 (RPM)'), title('Velocidad q1')
subplot(3,2,3), plot(t,q(:,2).*180/pi)
grid, xlabel('t (s)'), ylabel('q2 (°)'), title('Posición q2')
subplot(3,2,4), plot(t,q(:,5).*(180/(6.*pi)))
grid, xlabel('t (s)'), ylabel('dq2 (RPM)'), title('Velocidad q2')
subplot(3,2,5), plot(t,q(:,3).*180/pi)
grid, xlabel('t (s)'), ylabel('q3 (°)'), title('Posición q3')
subplot(3,2,6), plot(t,q(:,6).*(180/(6.*pi)))
grid, xlabel('t (s)'), ylabel('dq3 (RPM)'), title('Velocidad q3')
q_max=max(q(:,1:3));
q_min=min(q(:,1:3));
disp('Los valores articulares máximos y mínimos son:'), disp(q_max.*180/pi),
disp(q_min.*180/pi)
%Gráfico 4: Error y error en el espacio de trabajo
```

```
mError=sqrt((xref(:,1)-xreal(:,1)).^2+(xref(:,2)-xreal(:,2)).^2+...
    (xref(:,3)-xreal(:,3)).^2);
figure
plot(t,mError), xlabel('t (s)'), ylabel('Error (m)')
grid, title('Módulo del Vector Error en el Espacio de Trabajo')
SError=0;
for i=1:length(t)-1
    SError=SError+(mError(i)+mError(i+1))/2*(t(i+1)-t(i));
end
disp('La integral del módulo del vector error es:'), disp(SError)
%Gráfico 5: Animación del movimiento
V0=zeros(length(t),3);
V1=[zeros(length(t),1), zeros(length(t),1), a1*ones(length(t),1)];
V2=[cos(q(:,1))*a2.*cos(q(:,2)),                sin(q(:,1))*a2.*cos(q(:,2)),
a2*sin(q(:,2))+a1];
V3=[cos(q(:,1)).*(a3*cos(q(:,2)+q(:,3))+a2*cos(q(:,2))),
sin(q(:,1)).*(a3*cos(q(:,2)+q(:,3))+a2*cos(q(:,2))),
a3*sin(q(:,2)+q(:,3))+a2*sin(q(:,2))+a1];
figure
title('Animación del movimiento')
for i=1:length(t)
    data_1=[V0(i,:)' V1(i,:)' V2(i,:)'];
    data_2=[V1(i,:)' V2(i,:)' V3(i,:)'];
    h1=line([data_1(1,:);data_2(1,:)],[data_1(2,:);data_2(2,:)],...

[data_1(3,:);data_2(3,:)],'Color','k','Marker','o','MarkerSize',3);
    hold on
    h2=plot3(xref(1:i,1),xref(1:i,2),xref(1:i,3),...
             xreal(1:i,1),xreal(1:i,2),xreal(1:i,3));
    grid on, axis([0 0.35 -0.3 0.3 0 0.4]), view(52.5, 30)
    xlabel('x'), ylabel('y'), zlabel('z')
    Robot_Movie(i)=getframe(gcf);
    if i~=length(t)
        delete(h1), delete(h2)
    end
end
```

Anexo F: Costos de la investigación

Dispositivo o proceso		Cantidad	Precio Unitario	Precio Total
Actuadores Dynamixel	MX-64T	2 unid.	$ 299.90	$ 599.80
	MX-28T	1 unid.	$ 219.90	$ 219.90
	AX-12A	1 unid.	$ 44.90	$ 44.90
	AX-12W	1 unid.	$ 44.90	$ 44.90
Adaptador USB2Dynamixel		1 unid.	$ 49.90	$ 49.90
Adaptador SMPS2Dynamixel		1 unid.	$ 4.90	$ 4.90
Cables compatibles 250mm 3 Pin		1 paq.	$ 14.90	$ 14.90
6-Port Cable Hub		2 unid	$ 4.95	$ 9.90
Fuente de poder 12V - 5A (2.1mm Jack)		1 unid.	$ 19.95	$ 19.95
Impuestos de importación			$ 400	$ 400
Nylon 6			S/. 400	$ 136.5
Tornillos normalizados			S/. 70	$ 23.90
Pedestal del brazo robótico		1 unid.	S/. 200	$ 68.30
Fabricación de piezas			S/. 950	$ 324.20
Pintado de piezas			S/. 150	$ 51.20
Sensor Kinect		1 unid.	$ 280	$ 280
Trípode para sensor Kinect		1 unid.	$ 20	$ 20
TOTAL				$ 2313.15

Tipo de cambio considerado: S/. 2.93.