ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE COMUNICACIÓN PARA EL CONTROL DE POSICIÓN DE UN BRAZO ROBÓTICO DE FORMA LOCAL Y REMOTA.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN ELECTRÓNICA Y AUTOMATIZACIÓN, MENCIÓN REDES INDUSTRIALES

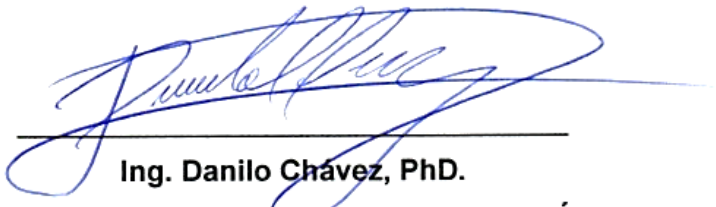
GABRIELA ALEXANDRA YERBABUENA TORRES

DIRECTOR: ING. GEOVANNY DANILO CHÁVEZ GARCÍA, PhD.

Quito, marzo 2024

AVAL

Certifico que el presente trabajo fue desarrollado por Gabriela Alexandra Yerbabuena Torres, bajo mi supervisión.



Ing. Danilo Chávez, PhD.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Gabriela Alexandra Yerbabuena Torres, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Gabriela Alexandra Yerbabuena Torres

DEDICATORIA

A mi familia, especialmente a mis padres por apoyarme incondicionalmente toda mi vida.

Gabriela

AGRADECIMIENTO

Les agradezco a mis padres Inés y Carlos, por haberme dado la vida, por acompañarme, guiarme y apoyarme en cada etapa de mi vida. Le agradezco a mi abuela Ermelinda por siempre estar presente y pendiente de sus nietos, siendo mi ejemplo de mujer valiente.

Gracias a los docentes de la Escuela Politécnica Nacional por impartir sus conocimientos a las nuevas generaciones.

En fin, gracias a todas las personas que me han impulsado y ayudado a superarme personal y profesionalmente.

Gabriela

ÍNDICE DE CONTENIDO

RESUMEN

Con la creciente demanda en la industria manufacturera, se ve la necesidad que las empresas automaticen sus procesos productivos con la implementación de brazos robóticos, sin embargo, esto puede significar costos elevados. Si existiese la posibilidad de programar el robot de forma independiente, sería de gran ayuda porque se podría codificar al autómata según requerimiento específicos, y mejor aún si se lo hace con un lenguaje de programación de código abierto que no implique costos adicionales. Haciendo hincapié a lo mencionado anteriormente, se plantea el presente trabajo cuyo enfoque se centra en el diseño e implementación de una interfaz de comunicación aplicando la cinemática directa e inversa para el control de movimientos del brazo robótico PincherX100 con 4 grados de libertad (GDL), mediante comunicación directa, local y remota, para lo cual se han diseñado tres interfaces de usuario respectivamente según el tipo de comunicación. El computador host es el que se comunica directamente al brazo robótico, mientras que los ordenadores local y remoto se comunicaran con el host para que este a su vez interactúe con el manipulador robótico.

Para desarrollar la interfaz de comunicación e interfaz gráfica de usuario se ha utilizado Python y el paquete de desarrollo DYNAMIXEL SDK, los cuales son compatibles entre sí y con los servomotores del autómata en mención. La cinemática directa fue resuelta a través del método geométrico, así como también mediante el algoritmo de Denavit Hartenberg, esto con el propósito de definir y comprobar las ecuaciones que se rigen al modelo analizado. A partir de dichas ecuaciones, y con herramientas propias de Python que permiten optimizar resultados, se solucionó la cinemática inversa. La comunicación local se realizó mediante la configuración de una red LAN entre los ordenadores que están conectados a un mismo router. Por su parte, la comunicación remota fue implementada mediante el uso de la versión gratuita del software Hamachi, mismo que permite la creación de escritorios remotos que se conectan mediante una red VPN.

El control de movimientos del brazo robótico se lo hizo utilizando softwares libres y de código abierto, demostrando que si es posible reducir los costos de implementación y deslindarse de softwares del fabricante. Sin embargo, se notó que la comunicación de forma remota demora seis veces más que la comunicación local, esto se debe a varios factores como la distancia entre dispositivos, ancho de banda o interferencias en la red.

PALABRAS CLAVE: Brazo robótico, 4GDL, cinemática directa, cinemática inversa, Python.

ABSTRACT

With the growing demand in the manufacturing industry, there is a need for companies to automate their production processes with the implementation of robotic arms, however, this can mean high costs. If there is the possibility of programming the robot independently, it would be of great help because the robot could be coded according to a specific requirement, and even better if it is done with an open-source programming language that does not involve additional costs. Emphasizing the, the present work is proposed, the focus of which is focused on the design and implementation of a communication interface applying direct and inverse kinematics for the control of movements of the PincherX100 robotic arm with 4 degrees of freedom (DOF), through direct, local, and remote communication, for which three user interfaces have been designed respectively according to the type of communication. The host computer is the one that communicates directly to the robotic arm, while the local and remote computers communicate with the host so that it in turn interacts with the robotic manipulator.

To develop the communication interface and graphical user interface, Python and the DYNAMIXEL SDK development package were used, which are compatible with each other and with the servomotors of the automaton in question. The direct kinematics was obtained through the geometric method, as well as through the Denavit Hartenberg algorithm, with the purpose of defining and verifying the equations that govern the analyzed model. From these equations, and with Python tools that allow optimizing results, the inverse kinematics was solved. Local communication was carried out by configuring a LAN network between computers that are connected to the same router. For its part, remote communication was implemented using the free version of the Hamachi software, which allows the creation of remote desktops that are connected through a VPN network.

The control of movements of the robotic arm was done using free and open-source software, demonstrating that it is possible to reduce implementation costs and distance yourself from manufacturer's software. However, it was noted that remote communication takes six times longer than local communication, this is due to several factors such as the distance between devices, bandwidth, or interference in the network.

KEYWORDS: Robotic arm, 4GDL, forward kinematics, inverse kinematics, Python.

1. INTRODUCCIÓN

En el contexto de la industria manufacturera en Ecuador, la creciente demanda de producción impone la necesidad imperante de evitar interrupciones o retrasos en las líneas de producción para satisfacer dicha demanda [1]. En los últimos años, se ha presenciado un marcado avance tecnológico que ha facilitado la incorporación de brazos robóticos en las industrias, tanto en las líneas de producción como en la logística de almacenes [2].

Actualmente, los brazos robóticos encuentran aplicaciones principalmente en la industria, la educación y la investigación. El costo de estos autómatas varía según su tamaño, número de articulaciones, tipo de efecto final y software [3]. Este último generalmente es propio del fabricante, por tanto, los costos se pueden elevar según las funcionalidades del programa, actualizaciones o mantenimiento.

No obstante, en el ámbito académico e investigativo, surge un inconveniente, ya que los brazos robóticos suelen estar preconfigurados para su utilización inmediata una vez adquiridos, lo que dificulta la realización de pruebas de nuevos esquemas de control y estudios de la dinámica propia del robot, entre otras investigaciones.

En muchos procesos industriales, se produce una interacción directa entre humanos y robots, lo que incrementa la posibilidad de fallos por parte del operador y puede causar daños a la infraestructura, las máquinas y al personal. Para afrontar este problema, las tendencias tecnológicas permiten controlar los procesos industriales de forma remota mediante la comunicación telemática. Con la implementación del control remoto, se pueden enviar datos a dispositivos distantes, lo que posibilita la automatización de tareas rutinarias y libera el tiempo de los operadores para actividades más críticas [4]. Además, se logra reducir las condiciones inseguras o peligrosas para los trabajadores.

Este estudio abordará la problemática a través de la programación e implementación de una interfaz de comunicación que permita controlar, de manera directa, local y remota, los movimientos del brazo robótico PincherX 100. La elección de este modelo de brazo robótico se fundamenta en sus características, tales como eficiente disipación de calor, alta durabilidad, tamaño compacto, monitoreo de temperatura, retroalimentación posicional, entre otros. Además, su costo es accesible y ha sido específicamente diseñado para su uso en la educación e investigación, con 4 grados de libertad, una rotación completa de 360 grados y una capacidad de carga de hasta 50 gramos [5].

La implementación de esta interfaz de comunicación posibilitará el control autónomo e independiente de los movimientos del brazo robótico, desligándolo del software del fabricante y eliminando los costos asociados a software comerciales y sus actualizaciones. Además, al lograr el control remoto de los movimientos, se mejora la seguridad del personal operativo y se abre la posibilidad de sincronizar los datos de diferentes robots para coordinar sus movimientos.

Para llevar a cabo este trabajo, inicialmente se ha realizado una revisión bibliográfica exhaustiva para comprender los tipos, características, funcionamiento y terminología referente a los brazos robóticos. Posteriormente, se ha recopilado información específica acerca del brazo robótico PincherX 100, así como también de los softwares libres o plataformas gratuitas compatibles con el manipulador, que además de crear el código, permitan la comunicación local y remota entre el usuario y el brazo robótico.

La metodología se basa en varias etapas, partiendo desde la definición de requerimientos a partir de la fase teórica, para seguidamente diseñar o programar el código con el que se controlará los movimientos del brazo robótico a través de una interfaz gráfica de usuario (GUI), considerando la aplicación de la cinemática directa e inversa mediante la utilización de métodos geométricos, algoritmos matemáticos y herramientas propias de Python.

Para la comunicación entre dispositivos se utiliza un enfoque cliente servidor, existiendo una comunicación bidireccional para escribir y leer datos entre el computador y el brazo robótico, esto mediante una conexión directa, local y remota. Al hablar de comunicación remota se puede hacer énfasis en las redes industriales que permiten conectar dispositivos desde diferentes zonas geográficas a través del internet, lo cual permite un intercambio de datos eficiente. Una vez creada la comunicación, las pruebas de simulación serán fundamentales para determinar el comportamiento y tiempos de respuesta según el tipo de conexión.

Por tanto, el problema del presente proyecto se centra en la necesidad de controlar de manera autónoma los movimientos de los brazos robóticos utilizados en la industria manufacturera de Ecuador, como son Cervecería Nacional, Ecuacerámica, por mencionar algunas; ya sea a través de comunicación local o remota. Esto permitirá superar las limitaciones actuales en términos de flexibilidad y dependencia de terceros.

1.1. Objetivo General

Diseñar e implementar una interfaz de comunicación para el control de posición de un brazo robótico de forma local y remota.

1.2. Objetivos Específicos

- Realizar una revisión bibliográfica sobre el control de movimientos de robots de forma directa y remota.
- Obtener los modelos cinemáticos directo e inverso del brazo robótico a fin de determinar las ecuaciones de movimientos.
- Diseñar e implementar las interfaces de comunicación para el control de movimientos de forma local y remota.
- Realizar el control de posición en lazo cerrado.
- Realizar pruebas de funcionamiento y análisis de resultados.

1.3. Alcance

Se realizará el diseño e implementación de dos interfaces de comunicación, que permitan controlar de manera directa y remota los movimientos de un brazo robótico PincherX 100 con 4GDL (4 grados de libertad GDL). Para la comunicación directa, se investigará acerca de softwares libres y de código abierto compatibles con la tarjeta Robotis DYNAMIXEL U2D2, mientras que para la comunicación remota se buscará una plataforma online que permita utilizar la nube para el almacenamiento y procesamiento de datos.

Se revisarán los datos del fabricante del brazo robótico PincherX 100 4GDL con el fin de entender el funcionamiento y control de este, y en base a estos datos se determinará los requerimientos tanto en hardware como en software que se necesitan para crear la interfaz de comunicación directa y remota.

En base a la cinemática directa e inversa se realizará la programación de los algoritmos para el control del posicionamiento del brazo robótico en lazo cerrado, esto es posible mediante la retroalimentación de la posición y/o la velocidad de los servomotores dinámicos a la tarjeta Robotis DYNAMIXEL U2D2, que a su vez envía los datos al ordenador. Para esto se hará una revisión bibliográfica del esquema estructural del brazo robótico y de los modelos matemáticos que permitan deducir las ecuaciones cinemáticas

de acuerdo con los grados de libertad existentes, todo esto a fin de conocer los rangos de movimiento de cada eslabón (base, brazo, antebrazo, muñeca).

Se analizará cual es el mejor programa de computador de acuerdo con sus funcionalidades y librerías, que permita realizar la interfaz para el control de los movimientos del brazo robótico, considerando además que el software libre seleccionado debe permitir el control tanto de forma directa como de forma remota a través de la nube.

Para la conexión entre la interfaz y el brazo robótico mediante la nube, se estudiarán alternativas como el uso de Ethernet industrial, tecnologías inalámbricas o únicamente el uso de librerías que permitan conseguir una comunicación. De ser necesario se considerará la implementación de tarjetas adicionales que permitan cumplir con los objetivos planteados.

Previo a la puesta en marcha del robot, se calibrarán los servomotores para indicar su posición inicial. Antes, durante y después a la ejecución del algoritmo programado, mediante el software DINAMIXEL propio de la tarjeta Robotis DYNAMIXEL U2D2, se verificará la respuesta de los servomotores mediante gráficos proporcionados por dicho software, debido a que estos muestran el comportamiento de los servomotores y facilitan el monitoreo de los mismos, esto es posible gracias a la comunicación bidireccional de la tarjeta de comunicación.

Para la interacción entre el usuario y el brazo robot, se implementará una interfaz en la que el usuario podrá dar órdenes al robot de dos maneras diferentes:

- El usuario podrá ingresar los ángulos deseados de cada articulación y el sistema mediante la cinemática directa determinará la posición en el espacio del efector final.
- El usuario podrá ingresar la posición deseada del efector final y el sistema calculará mediante la cinemática inversa los ángulos de cada articulación que permitan llegar a dicha posición.

Modificaciones

Durante la ejecución del proyecto, se realizaron algunas modificaciones en relación con el alcance inicialmente propuesto. Originalmente, se había contemplado la implementación de la cinemática inversa en base a modelos matemáticos desarrollados desde cero, sin embargo, tras una evaluación exhaustiva de las opciones disponibles, se optó por aprovechar las capacidades de librerías específicas de Python. Esta decisión se basó en

la eficacia demostrada por dichas librerías, lo que permitió una implementación más eficiente y rápida del componente de cinemática inversa del brazo robótico de 4GDL.

Adicionalmente, en lugar de realizar el control remoto utilizando la nube, se eligió implementar una solución de control remoto desde un computador remoto. De tal forma que computador remoto, envié la posición deseada al computador principal o host, y este a su vez al estar conectado directamente al brazo robótico, realicé el envió de datos al autómata. Esta elección se fundamentó en que la implementación del proyecto busca independencia, autonomía, seguridad y reducción de costos, por lo que el uso de plataformas en la nube no favorece a solucionar el problema planteado inicialmente.

Estas modificaciones se realizaron con la intención de mejorar la eficiencia y la efectividad del proyecto. Cabe mencionar que, a pesar de estas adaptaciones, los resultados obtenidos demostraron el éxito en el diseño e implementación de las interfaces de comunicación para el control del brazo robótico PincherX 100 con 4GDL.

1.4. Marco Teórico

1.4.1. Robot

Un robot es un dispositivo servocontrolado que realiza manipulación automática, siendo reprogramable y versátil. Su función principal consiste en posicionar y orientar piezas, herramientas o dispositivos especiales, siguiendo trayectorias variables que pueden ser reprogramadas, por lo que su uso habitual implica la realización de tareas repetitivas, con la capacidad de adaptarse a otra sin requerir cambios permanentes en su estructura física. Generalmente, adoptan la forma de brazo que culmina en una muleca o pinza, esto con el fin de lograr sujetar objetos para poder ser transportados de un lugar a otro [6].

1.4.1.1. Clasificación de los robots

Existen varios autores que clasifican a los robots según su generación, área de aplicación, tipo de actuadores, numero de ejes, configuración, tipo de control. En la presente investigación para categorizar a los robots se considerará su morfología, obteniendo los siguientes tipos [7]:

- Robot Industrial o Manipulador
- Robots Móviles
- Androide o humanoide

- Zoomórficos

1.4.2. Robot Industrial o Manipulador

Los robots industriales son dispositivos avanzados que combinan componentes mecánicos y electrónicos para realizar tareas específicas en procesos de fabricación o manipulación. Estos robots se utilizan principalmente en entornos industriales para diversos fines. Los robots industriales han ganado una popularidad significativa y se usan ampliamente en todo el mundo [7].

Manipulador

Es un mecanismo compuesto típicamente por componentes conectados en serie y articulados entre sí, cuya función principal es sujetar y mover objetos mediante el control de posición del efector final. Posee una versatilidad en sus funciones y puede ser controlado tanto manualmente por un operador humano como a través de un dispositivo lógico, ver Figura 1.1. Sus servomotores están compuestos por engranajes y microcontroladores, capaces de mover y soportar el propio peso de la estructura externa o eslabones, garantizando el posicionamiento deseado en cada uno de los elementos del autómata [8].

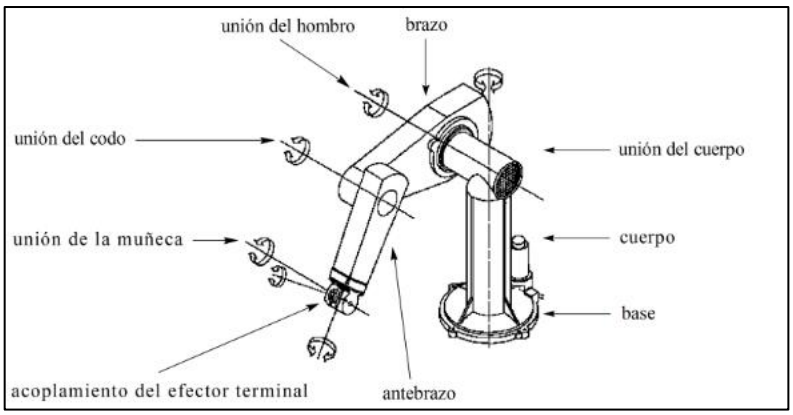


Figura 1.1. Estructura de un Manipulador [8].

1.4.3. Morfología de los robots

La estructura y funcionamiento de un robot móvil pueden ser asociados a un ser vivo, como se muestra en la Figura 1.2., puesto que la morfología de los robots hace referencia a los elementos que conforman el mismo, siendo estos: estructura mecánica, sensores y actuadores, y el sistema inteligente [9].

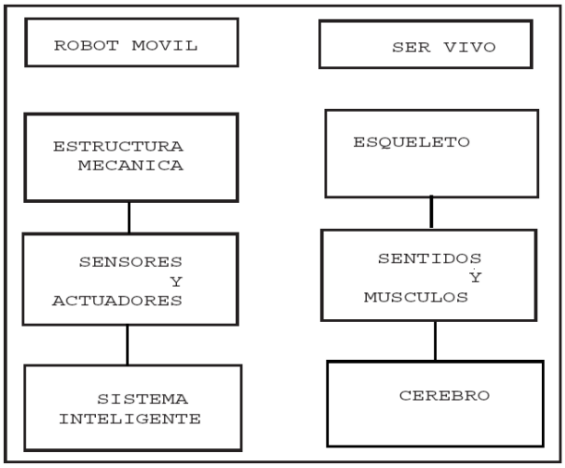


Figura 1.2. Estructura de un robot móvil [9].

1.4.3.1. Estructura Mecánica

Para comprender lo que es la estructura mecánica es importante tener claros dos conceptos, como son eslabón y grados de libertad.

- **Eslabón:** es una pieza que forma parte de la estructura articulada de un robot y permite el movimiento relativo con otros eslabones.
- **Grados de libertad:** son el número de movimientos independientes que pueden realizar un robot en su estructura o articulaciones. Los tres primeros grados de libertad correspondientes a la base, brazo y antebrazo, son fundamentales puesto que permiten definir la posición del efector final, mientras que el resto o demás GDL son utilizados para determinar la orientación de la pinza o gripper.

La estructura mecánica en robótica consta de elementos o eslabones que están conectados por articulaciones, esto les permite moverse consecutivamente de par en par. Los robots industriales a menudo se asemejan al brazo humano en su construcción física, por lo que se suele utilizar términos como cuerpo, brazo, codo y muñeca para describir las diferentes partes del robot. Cada articulación puede moverse en un desplazamiento, rotación o una combinación de ambos, lo que da como resultado seis tipos diferentes de articulación, ver Figura 1.3. En la práctica se emplea generalmente la articulación rotacional y la prismática.

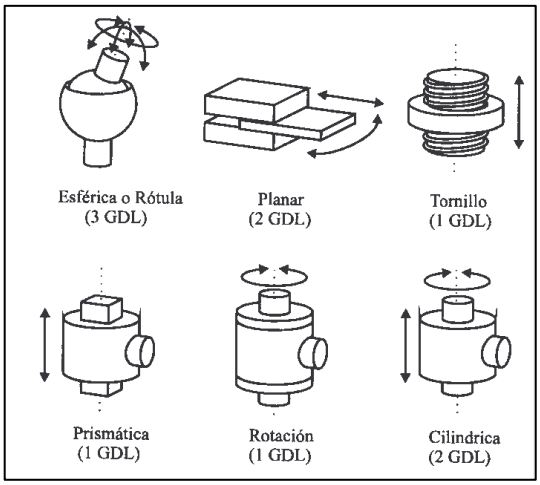


Figura 1.3. Distintos tipos de articulaciones para robots [6].

Los grados de libertad de un robot se define a partir de la suma de los grados de libertad en cada articulación que este posea.

A partir de la combinación de los diferentes tipos de articulaciones, surgen nuevas configuraciones para el diseño, construcción y aplicación de robots, entre los arreglos más frecuentes se muestra en la Figura 1.4., donde se observa que las tres articulaciones iniciales del robot, que son las que permiten posicionar el extremo de este en un punto definido del espacio [6].

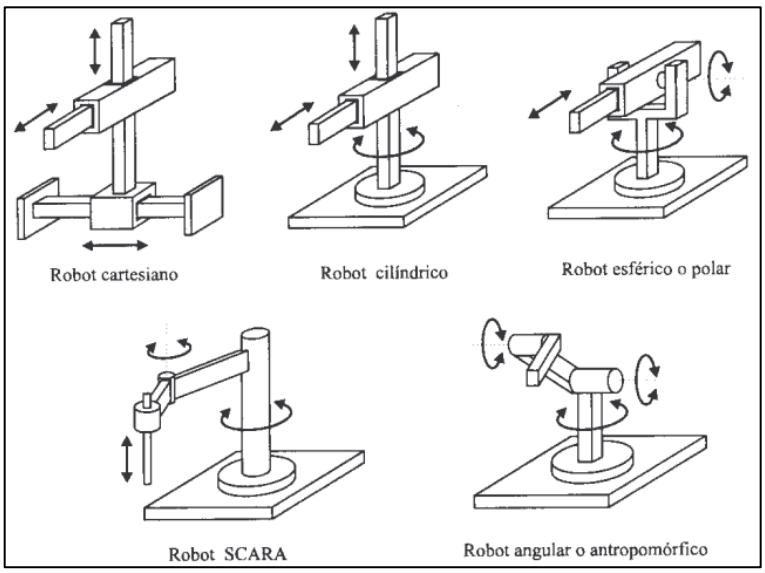


Figura 1.4. Configuraciones más frecuentes en robots industriales [6].

1.4.3.2. Sensores y Actuadores

Para que un robot realice su tarea de manera efectiva, debe ser consciente tanto de su propia condición como del entorno que lo rodea. Los sensores internos brindan información sobre el estado del robot, incluida la posición de sus articulaciones; por su parte los sensores externos recopilan datos sobre el espacio físico que lo rodea. Además, la unidad de control del robot puede obtener información sobre su estructura mecánica, como la posición y la velocidad. La Tabla 1.1. describe los sensores comúnmente utilizados en robots industriales para obtener información de presencia, posición y velocidad.

Los actuadores en robótica tienen la tarea de generar movimiento en los componentes del robot según las instrucciones de la unidad de control. Inicialmente, los actuadores hidráulicos se usaban en robots industriales, pero las ventajas prácticas de la electricidad como fuente de energía la han convertido en la opción preferida para la mayoría de los robots en la actualidad. Sin embargo, hay situaciones en las que se deben considerar otros tipos de actuadores. En robótica, los actuadores pueden utilizar fuentes de energía neumática, hidráulica o eléctrica. Cada sistema tiene sus propias características que incluyen potencia, capacidad de control, peso y volumen, precisión, velocidad, requisitos de mantenimiento y consideraciones de costos [6].

Tabla 1.1. Tipos de sensores propioceptivos para robots [6].

Medida	Tipo
Presencia	• Inductivo • Capacitivo • Efecto hall • Óptico • Ultrasónico • Contacto
Posición Analógicos	• Potenciómetros • Resolver • Sincro
Posición Digitales	• Digitales • Encoders absolutos • Encoders incrementales • Regla óptica
Velocidad	• Tacogeneratriz

1.4.3.3. Sistema Inteligente o programación de algoritmos

El sistema inteligente hace referencia a los algoritmos y programación, puesto que estos son los que permiten el desarrollo de sistemas inteligentes mediante la definición y ejecución de secuencias lógicas de instrucciones. Al crear algoritmos eficientes e implementarlos a través de la programación, es posible dotar a los robots con capacidades de procesamiento de datos, la toma de decisiones y la resolución de problemas complejos de forma automatizada. Estas herramientas permitirán que el robot pueda interactuar con su entorno en base a la señal de los sensores o a las señales eléctricas enviadas por un operador [9].

1.4.4. Descripción de la plataforma a utilizarse

En el mercado existen una gran variedad de brazos robóticos, los cuales varían su costo de acuerdo con sus características, como grados de libertad, tipo de servomotores, material de la estructura, entre otros. Considerando precio y funcionalidad, el proyecto se trabajará con el brazo robótico PincherX 100.

1.4.4.1. Brazo Robótico PincherX 100

El PincherX-100 es un brazo robótico que pertenece a la familia de la serie X de Interbotix, diseñado para la investigación y educación. Está equipado con servomotores inteligentes DYNAMIXEL X-Series, que proporcionan un alto par, ver Figura 1.5. El PincherX 100 cuenta con un convertidor de comunicación USB llamado Robotis DYNAMIXEL U2D2, este permite que el computador pueda conectarse con los servomotores. Proporciona una configuración con cuatro grados de libertad (4 GDL) y puede soportar una carga de hasta 50 gramos [10].

PincherX-100	
Grados de libertad	4
Alcanzar	300mm
Envergadura total	600mm
Repetibilidad	5mm
Exactitud	8mm
Carga útil de trabajo	50g*
Total Servos	5
Rotación de muñeca	No

Figura 1.5. Brazo Robótico PincherX 100 [10].

1.4.5. Componentes del Brazo Robótico PincherX 100

1.4.5.1. Estructura

El brazo PincherX 100 este fabricado en acrílico rígido P1 de 8/95, por lo que tiene alta resistencia al calor. El brazo se levanta sobre un rodamiento de rodillos de aguja de 7/8" para garantizar su estabilidad. Por su parte, la electrónica está protegida por un escudo acrílico transparente, que evita la acumulación de residuos [5].

El brazo robótico está conformado de la base fija, cintura (base giratoria), brazo (eslabón 1), antebrazo (eslabón 2), muñeca (eslabón 3), gripper (o pinza) y 5 servomotores DYNAMIXEL XL430-W250-T. Ver Figura 1.6.

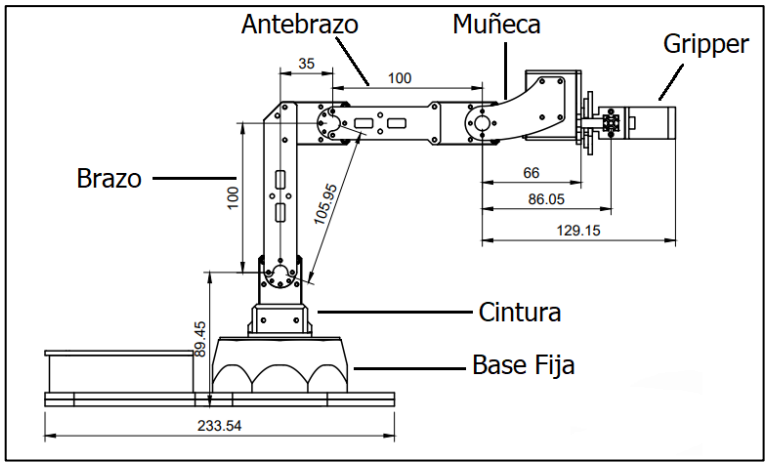


Figura 1.6. Estructura y Medidas del Brazo Robótico PincherX 100 [10].

El brazo tiene un alcance total de 600 mm, sin embargo, el fabricante recomienda extender máximo el 70% de su extensión total admisible [10], ver Tabla 1.2. En la Figura 1.7. se observa la vista superior de la extensión del brazo robótico, el circulo verde indica el espacio de trabajo recomendado, en donde el brazo puede extenderse un máximo de 420 mm.

Tabla 1.2. Espacio de trabajo y alcance total del brazo (mm) [11].

Brazo	Espacio de trabajo recomendado (mm)	Alcance total (mm)
PincherX 100	420	600

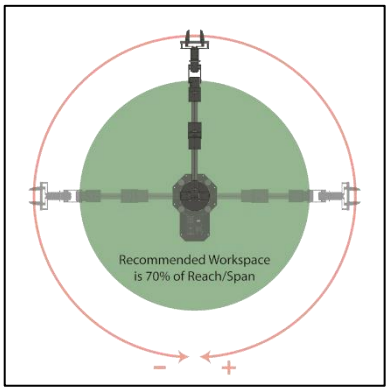


Figura 1.7. Espacio de trabajo recomendado [11].

La apertura máxima del gripper o pinza es de 74 mm, y un cierre de hasta 30 mm [11]. Ver Figura 1.8.

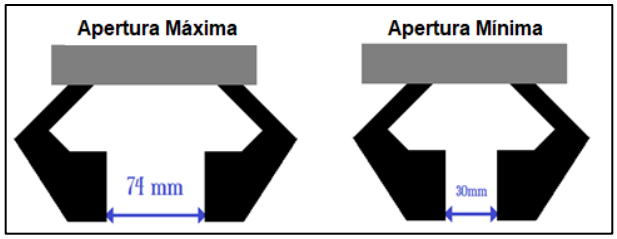


Figura 1.8. Límites de apertura del gripper.

1.4.5.2. Servomotores DYNAMIXEL XL430-W250-T

El brazo robótico PincherX 100 cuenta con 5 servomotores DYNAMIXEL XL430-W250-T, ubicados en la cintura (servomotor1/ S1), brazo (servomotor2/ S2), antebrazo (servomotor3/ S3), muñeca (sevomotor4/ S4) y pinza (servomotor5/ S5); siendo los cuatro primeros los que permiten posicionar el efector final, resultado el brazo robótico con 4GDL, ver Figura 1.9.

Figura 1.9. Ubicación de los servomotores [10].

Los servomotores DYNAMIXEL XL430-W250-T son actuadores robóticos avanzados que se utilizan generalmente en proyectos robóticos y aplicaciones de automatización. Estos servomotores permiten un control preciso de posición, torque y velocidad, retroalimentación de posición y cuenta con una interfaz de comunicación robusta, considerando además que tienen un rango de giro de 360° [12], ver Figura 1.10. En la Tabla 1.3., se muestran las características principales de dichos actuadores [13].

Figura 1.10. Servomotor DYNAMIXEL XL430-W250-T [12].

Tabla 1.3. Características del Servomotor DYNAMIXEL XL430-W250-T [13].

Característica	Especificación
Peso	57.2 gramos
Dimensiones (An. x Al. x Pr.)	28,5 x 46,5 x 34 [mm]
Algoritmo de control	Control PID
Voltaje de entrada	6,5 ~ 12,0 [V] (recomendado: 11,1 [V])
Material de la Caja y Engranajes	Plástico de ingeniería

Los actuadores DYNAMIXEL XL430-W250-T tienen integrado un circuito de control de posición, ver Figura 1.11. [13]. Consiguiendo ampliar significativamente las capacidades y la precisión del sistema. Estos servomotores avanzados, al poseer controladores de posición incorporados, permiten operar en lazo cerrado, lo que significa que el sistema recibe retroalimentación en tiempo real de la posición actual de cada articulación.

Esta funcionalidad resulta esencial en el contexto del proyecto, ya que posibilita un control más preciso y reactivo de los movimientos del brazo robótico. El control de posición en lazo cerrado garantiza que cada articulación alcance y mantenga la posición deseada con una alta exactitud.

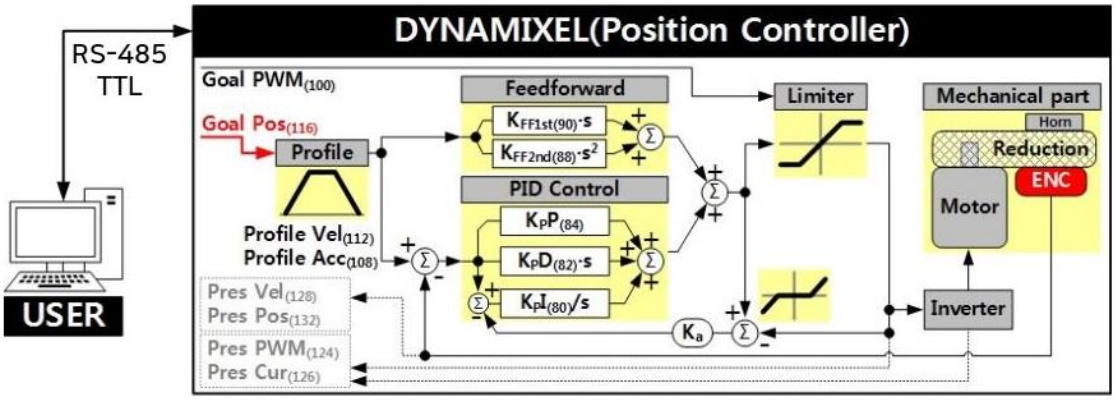


Figura 1.11. Circuito de control de posición [13].

1.4.5.3. Robotis DYNAMIXEL U2D2

DYNAMIXEL U2D2 es un adaptador de comunicación desarrollado por ROBOTIS, una empresa especializada en robótica y automatización. En la Figura 1.12. se muestra este adaptador que permite la comunicación entre un ordenador o una plataforma de control con los servomotores DYNAMIXEL.

El DYNAMIXEL U2D2 se utiliza en proyectos donde se requiere una interfaz de comunicación para programar, configurar y controlar los servomotores DYNAMIXEL de manera más sencilla y efectiva. Este adaptador es especialmente útil en aplicaciones de robótica, automatización industrial y sistemas de educación en robótica.

Para el funcionamiento del convertidor U2D2 y de los servomotores se necesita que estos se conecten a una fuente externa que les proporcione energía, ver Figura 1.13., es importante conocer el voltaje que necesitan estos dos elementos para su correcto funcionamiento, puesto que de lo contrario pueden existir fallos y daños en los componentes [14].

Figura 1.12. DYNAMIXEL U2D2 [14].

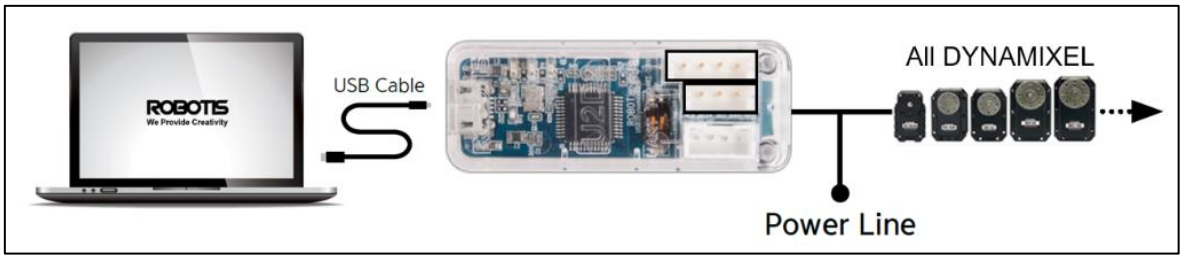


Figura 1.13. Modo de conexión de DYNAMIXEL U2D2 [14].

Al momento de realizar la programación, es importante que los servomotores a utilizarse tengan un ID único, para que de esta forma se evite que los datos se pierdan durante su transmisión.

U2D2 es capaz de trabajar con diferentes softwares como RoboPlus, DYNAMIXEL Workbench y DYNAMIXEL SDK, siendo este último con el que se trabajara en el presente proyecto, debido a que es compatible con diferentes lenguajes de programación: C, C++, C#, MATLAB, LabVIEW, Python y Java [14].

1.4.6. Cinemática

El estudio de la cinemática de robots se centra en analizar el movimiento de un robot en relación con un sistema de referencia, sin considerar las fuerzas involucradas. Este campo se ocupa principalmente de describir el movimiento espacial del robot a lo largo del tiempo y comprender la relación entre sus coordenadas conjuntas y la posición y orientación de su punto final.

Hay dos problemas principales en la cinemática de los robots. El primer problema se conoce como el problema cinemático directo, que consiste en determinar la posición y orientación del punto final del robot en relación con un sistema de coordenadas determinado, en función de los valores de sus articulaciones y parámetros geométricos. El segundo problema se denomina problema de cinemática inversa, que consiste en encontrar la configuración que debe adoptar el robot en sus articulaciones para lograr que el punto final se coloque en la posición deseada. Ver Figura 1.14.

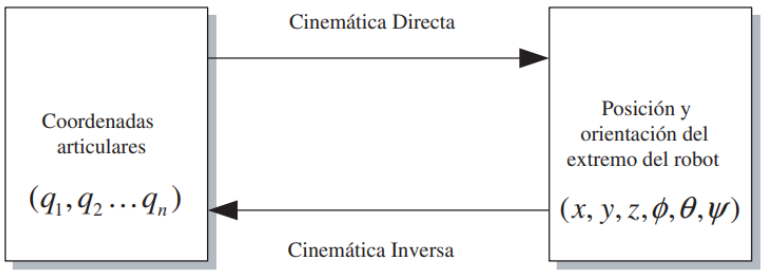


Figura 1.14. Diagrama de relación entre cinemática directa e inversa [6].

En donde q_1, q_2, q_3, …, q_n, corresponden a los ángulos o coordenadas articulares en cada nodo. Mientras que x, y, z representan la traslación a lo largo de estos ejes; de la misma forma Φ, θ, Ψ simbolizan la rotación en x, y, z respectivamente [6].

1.4.6.1. Cinemática Directa

La resolución de problemas de cinemática directa ayuda a determinar la posición y orientación del extremo de un robot cuando se asignan valores específicos a sus variables conjuntas o coordenadas articulares. Esta información es crucial para la unidad de control del robot, puesto que puede leer directamente estas variables de los sensores de posición y de esta forma proporcionar a los usuarios información sobre el efector final del robot.

Existen dos enfoques para obtener el modelo cinemático directo: métodos geométricos y métodos basados en cambios en los sistemas de referencia. Los métodos geométricos son adecuados para casos simples, es decir para manipuladores con pocos grados de libertad. Por otro lado, los métodos basados en cambios en los sistemas de referencia ofrecen un enfoque sistemático para obtener el modelo cinemático directo para robots con cualquier número de grados de libertad, siendo los más utilizados, especialmente aquellos que utilizan matrices de transformación homogéneas [6].

Algoritmo de Denavit Hartenberg

El Algoritmo de Denavit Hartenberg es un método matricial crucial en la cinemática de robots manipuladores articulados, presentado por primera vez por Denavit y Hartenberg en 1955. Este enfoque brinda una estructura sistemática para describir y modelar los movimientos de los autómatas [6].

El principio central de este algoritmo radica en la creación de un sistema de coordenadas en cada eslabón de la cadena articulada del robot. Consiste en establecer y relacionar transformaciones entre los sistemas de referencia de cada eslabón, permitiendo comprender las rotaciones y traslaciones entre estos sistemas.

Las transformaciones clave en el método de Denavit Hartenberg se basan en cuatro parámetros fundamentales:

- $\boldsymbol{\theta}$*:* ángulo entre x_{i-1} y x_i en torno al del eje z_{i-1}.
- $\boldsymbol{d}\ (\mathbf{0},\mathbf{0},\boldsymbol{d})$: distancia de traslación entre x_{i-1} y x_i a lo largo del eje z_{i-1}.
- $\boldsymbol{a}\ (\boldsymbol{a},\mathbf{0},\mathbf{0})$: distancia de traslación entre z_{i-1} y z_i a lo largo del eje x_i.
- $\boldsymbol{\alpha}$*:* ángulo entre z_{i-1} y z_i en torno al del eje x_i.

Con estos parámetros definidos, podemos calcular las variables θ, d, a y α para cada eslabón del robot y, a partir de estos datos, determinar los ángulos y distancias que describen la posición y orientación del extremo del brazo robótico en función de su estructura misma. Para relacionar estos parámetros, se emplean fórmulas y relaciones angulares entre los ejes y planos en cada eslabón. Por ejemplo, $\boldsymbol{\theta}_i$ corresponde al ángulo entre X_{i-1} y X_i alrededor del eje Z_{i-1}, $\boldsymbol{d}_i$ es la distancia entre X_{i-1} y X_i a lo largo de Z_{i-1}, $\boldsymbol{a}_i$ es la distancia entre Z_{i-1} y Z_i a lo largo de X_i, y $\boldsymbol{\alpha}_i$ es el ángulo entre Z_{i-1} y Z_i alrededor del eje X_i.

De tal manera, el Algoritmo de Denavit Hartenberg proporciona una metodología efectiva y sistemática para describir la cinemática de robots articulados, utilizando parámetros clave para definir las relaciones espaciales entre los eslabones y, por ende, determinar la posición y orientación del efector final del brazo robótico en un espacio tridimensional.

Por tanto, la transformación del sistema móvil queda expresado de la siguiente manera:

$$\boldsymbol{A_i^{i-1} = RotZ(\theta_i)\, T(O,O,d_i)\, T(a_i,O,O) RotX(\alpha_i)} \qquad (1.1)$$

De la multiplicación de matrices se determina la matriz A_i^{i-1}, que relaciona a todos los eslabones entre sí, misma que se muestra a continuación:

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

$$\boldsymbol{A_i^{i-1} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}} \qquad (1.2)$$

Para definir la matriz $\boldsymbol{A_i^{i-1}}$ basta con determinar los parámetros θ, d, a, α. Esta matriz se aplicará a cualquier modelo de manipulador robótico, lo que variará son los valores que tomen los cuatro parámetros antes mencionados y la posición de los sistemas de referencia en cada nodo, esto dependerá del tipo de articulación que se presente ya sea prismática, rotacional, planar, etc. Para establecer los valores de θ, d, a, α se realizará un análisis de par en par entre nodos consecutivos, esto debido a que la conexión es serial.

En la sección 2.2.6.2. se detalla la aplicación del método de Denavit Hartenberg para el brazo robótico PincherX 100.

1.4.6.2. Cinemática Inversa

La cinemática inversa tiene como propósito determinar las coordenadas de articulación de un robot para que su efector final se posicione en las coordenadas espaciales definidas por el usuario.

Se han desarrollado varios procedimientos para resolver el problema de la cinemática inversa mediante programación, utilizando información sobre la cinemática del robot. Sin embargo, estos procedimientos son métodos numéricos iterativos que debido a su complejidad pueden llegar a presentar demoras o retrasos en el procesamiento de datos, por lo que no es garantizado.

A diferencia del problema de la cinemática directa, es posible que no haya una solución única para el problema de la cinemática inversa. Puede haber múltiples conjuntos de coordenadas conjuntas que posicionan y orientan el efector final del robot de la misma manera. En tales casos, una solución cerrada permite incorporar reglas o restricciones para asegurar que la solución obtenida sea la más adecuada (por ejemplo, considerando rangos de alcance) [6].

1.4.7. Tipos de conexión a la plataforma robótica

1.4.7.1. Directa

Una conexión directa se refiere a un enlace físico establecido entre dos dispositivos o nodos sin la participación de ningún otro dispositivo de red. Este tipo de conexión permite una transferencia de datos eficiente ya que los dispositivos están conectados directamente entre sí a través de un medio de transmisión físico como un cable. Los ejemplos de conexiones directas incluyen conexiones Ethernet, seriales o USB.

1.4.7.2. Local

Una conexión local se refiere a la comunicación y el acceso a recursos dentro de una red específica o un área limitada, como una red de área local (LAN). En una conexión local, los dispositivos se encuentran muy cerca unos de otros, lo que permite una transferencia de datos rápida y una latencia mínima. Un escenario típico es cuando varios dispositivos como computadoras, impresoras y servidores están conectados a un enrutador o conmutador dentro de una oficina o un hogar [15].

1.4.7.3. Remota

Una conexión remota implica la comunicación entre dos dispositivos que están situados en redes diferentes, a menudo en ubicaciones geográficamente distantes. El intercambio de información tiene lugar a través de tecnologías de red como Internet. Por ejemplo, el acceso remoto a una red corporativa desde una ubicación externa se puede lograr a través de una conexión VPN (red privada virtual). Del mismo modo, acceder a un servidor web alojado en otro país también constituye una conexión remota [16].

1.4.8. Escritorio Remoto

Gracias a los avances tecnológicos, se ha logrado controlar o interactuar con otros ordenadores sin estar en el mismo la misma área geográfica, esto es posible gracias a la implementación de escritorios remotos, estos surgieron a partir de la necesidad de acceder a un ordenador y poder controlar su sistema operativo, o como puede ser también para ayudar a arreglar algún servidor. Esta tecnología es útil a la hora de querer tener el control de una base de datos en diferentes ordenados, debido a que el escritorio remoto puede acceder y controlar varios dispositivos, solo se necesita vincular los dispositivos remotos con el computador principal, dicha conexión se la puede hacer a través de internet, bluetooth, otros.

Al igual que todas las aplicaciones, tiene su propio lenguaje informático, en este caso vemos que el usuario o cliente es denominado Host. Este tipo de aplicaciones permiten acceder al software del computador, consiguiendo visualizar y manipular, archivos, imágenes, aplicaciones [17].

1.4.8.1. Hamachi by LogMeIn

Figura 0.15. Software de escritorio remoto Hamachi [18].

LogMeIn Hamachi es un escritorio remoto, consiste en una aplicación que permite crear una red privada virtual (VPN) segura a través de Internet, es decir la comunicación es inalámbrica y se encuentra encriptada de extremo a extremo. Esta red VPN permite que los dispositivos conectados se comuniquen entre sí como si estuviesen en una red LAN física, sin importar que los dispositivos interconectados estén en diferentes ubicaciones geográficas. Hamachi es muy útil al momento de requerir una conexión segura que permita compartir recursos de forma privada entre dispositivos remotos.

Hamachi al crear redes VPN, puede que en ciertos países no pueda ejecutarse puesto que este tipo de red está restringida en varias naciones por motivos de seguridad.

Para que la conexión e intercambio de datos mediante Hamachi sea de forma fluida se debe verificar que la conexión a internet sea buena, ya sea mediante Wi-Fi o a través de cable ethernet.

Actualmente esta aplicación es propiedad de LogMeIn, su servicio posee dos modalidades, una versión gratuita y otra pagada, es notorio las diferencias, ya que, por ser de paga, se puede obtener más opciones que hará la experiencia más agradable, sin embargo, la versión gratuita realiza perfectamente su objetivo que es, crear y mantener una conexión privada, admitiendo un máximo de cinco servidores o dispositivos.

Es importante considerar los riegos que toda conexión inalámbrica puede presentar, en donde la privacidad de datos se puede ver expuesta ya que existe en la nube miles de virus que intentan robar datos de los cibernautas, por ello es recomendable usar contraseñas seguras para conectar los dispositivos remotos [18].

1.4.9. Interfaz de Comunicación

Las interfaces de comunicación son sistemas, protocolos, dispositivos y mecanismos que permiten el intercambio de información y comunicación efectiva entre dos o más sistemas informáticos o componentes electrónicos. La comunicación puede ocurrir a nivel local a

través de múltiples dispositivos en la misma red o extenderse a redes más grandes como Internet.

1.4.10. Interfaz Gráfica de Usuario (GUI)

Una interfaz gráfica de usuario (GUI) es un entorno virtual y visual que permite a los usuarios comunicarse con uno o más dispositivo sin la necesidad de usar comandos de texto en una consola. Proporciona un entorno intuitivo de usar, en donde las personas pueden realizar diversas acciones sin necesidad de tener conocimientos previos de programación [19].

1.4.11. Lenguaje de Programación

Un lenguaje de programación permite enviar instrucciones que un sistema puede interpretar y ejecutar, esto se cumple en base al uso de símbolos, comandos, algoritmos o reglas. Se lo realiza de manera sistematizada y sincronizada, lo que permite que los dispositivos electrónicos como: smartphone, pc, laptops, entre otros, realicen misiones o ejecuten comandos que el usuario necesite. Los lenguajes de programación son esenciales para el desarrollo de software y la creación de aplicaciones, programas y sistemas informáticos. Permiten a los programadores expresar sus ideas y algoritmos de manera estructurada, y luego convertir esas ideas en código para que las computadoras puedan comprender y ejecutar las instrucciones indicadas por el usuario [20].

1.4.11.1. Python

Figura 0.16. Lenguaje de programación Python [21].

En los últimos años la comunidad de Python ha crecido significativamente, por lo que existen muchos usuarios que contribuyen a la biblioteca DreamWorks y a los recursos educativos, lo cual resulta de gran ayuda al momento de resolver algún problema o de obtener algún código o ejercicio.

Python es un lenguaje de programación de alto nivel puesto que su estructura y código es de fácil legibilidad e interpretación. Es decir, es sencillo, claro y preciso, para que los dispositivos electrónicos puedan ejecutar las ordenes sin dificultas, y, además, que el programador no utilice códigos complejos de constituir o de comprender.

Python es ampliamente reconocido por ser un lenguaje versátil y poderoso, adecuado para una amplia variedad de proyectos y aplicaciones, desde pequeños scripts hasta aplicaciones empresariales complejas que involucren la comunicación a través de la red [21].

1.4.11.2. Librerías de Python

Python cuenta con una amplia variedad de bibliotecas (también conocidas como módulos o paquetes) que proporcionan funcionalidades adicionales para diferentes propósitos. Estas bibliotecas son creadas y mantenidas por la comunidad de desarrolladores y amplían las capacidades del lenguaje en diversas áreas.

A continuación, se muestras algunas de las muchas bibliotecas disponibles en Python, las que se mencionan son las más relevantes para el desarrollo de la presente investigación.

- **Pygame:** es especialmente popular entre los principiantes y los desarrolladores que desean crear juegos 2D de manera rápida y sencilla. Aunque no es tan potente como algunos motores de juegos más avanzados, Pygame es una excelente opción para proyectos más pequeños y aprendizaje práctico sobre programación de juegos y aplicaciones multimedia [22].

Figura 0.17. Librería Pygame de Python [22].

- **Math:** Es un módulo incorporado en la biblioteca estándar de Python que proporciona funciones y constantes matemáticas. Este módulo permite a los programadores realizar cálculos matemáticos más avanzados y operaciones numéricas complejas en sus programas [23].

- **Openpyxl:** Permite trabajar con archivos de Excel en el formato .xlsx. Esta biblioteca facilita la creación, manipulación y lectura de hojas de cálculo y libros de Excel utilizando el formato moderno de archivo introducido por Microsoft Office 2007 y versiones posteriores [24].

- **Time:** Es un módulo que esta incorporado en la biblioteca estándar de Python, proporciona funciones para trabajar con medidas de tiempo y realizar operaciones relacionadas con el tiempo. Este módulo permite a los programadores trabajar con

el tiempo actual, medir intervalos de tiempo, pausar la ejecución de un programa y más [25] .

1.4.12. Entorno al desarrollo integrado (IDE)

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es una aplicación de software que proporciona un conjunto completo de herramientas y funciones que permiten crear y escribir códigos de computadora para el desarrollo de softwares o aplicaciones.

Los IDEs a menudo ofrecen generadores de código, plantillas, bibliotecas, lo que producen un ahorro de tiempo al momento de escribir códigos repetitivos o estructuras comunes, por lo que el proceso de desarrollo del código es más eficiente y ordenado, y menos propenso a errores [26].

1.4.13. Registro de datos en Excel

El almacenar, preservar y analizar un determinado conjunto de datos obtenidos de cierta actividad permite tener un mayor control de dicha actividad. Existen varios softwares que permiten el registro y análisis de datos, entre ellos el más conocido es Excel.

La base de datos que se realiza en Excel se lo puede definir como una tabla en la cual se plasman datos de entrada y salida, en donde estos serán clasificados por categorías y mediante lenguaje numérico nos dará los resultados deseados.

Excel funciona, a partir de una hoja de cálculo, lo cual por medio de celdas representa un registro de entrada de varios datos. Existe celdas de forma en fila, en esta sección representa los datos primarios, en la celda o columna representa datos de complementación o de refuerzo.

1.4.14. Trayectoria

La trayectoria se refiere a la ruta espacial que sigue el extremo del brazo robótico (la herramienta o efector final) durante un determinado movimiento. Esta ruta se representa en un espacio tridimensional y describe la posición y orientación del extremo del brazo en función del tiempo o de otro parámetro, como el número de pasos.

En un brazo robótico es fundamental conocer su trayectoria para planificar y controlar sus movimientos de manera precisa y eficiente. Puede ser una trayectoria simple, como un movimiento rectilíneo desde un punto A hacia un punto B, o una trayectoria compleja, como una serie de movimientos curvilíneos o articulados para completar una tarea específica.

La trayectoria puede ser predefinida y almacenada en el controlador del robot antes de la ejecución (trayectoria planificada), o puede ser generada en tiempo real mediante algoritmos de control basados en la retroalimentación sensorial (trayectoria en tiempo real). Además, en el contexto de control local y remoto, la trayectoria puede ser generada y controlada desde un dispositivo local (como un ordenador cercano al robot) o de forma remota a través de una conexión telemática.

Es decir, la trayectoria se define como el camino parametrizado en el tiempo que sigue el extremo del brazo robótico en el espacio tridimensional. Su correcta definición y control son esenciales para lograr movimientos precisos y coordinados del robot en el cumplimiento de diversas tareas [27].

2. METODOLOGÍA

El desarrollo del presente trabajo se basará en un enfoque investigativo y experimental que busca implementar tres interfaces de comunicación que permitan controlar los movimientos del brazo robótico PincherX 100; la interfaz principal utilizará una comunicación directa entre el brazo robótico y el computador anfitrión o host, mientras que las interfaces para el control de movimientos de forma local y remota se conectaran al host para que este a su vez interactúe con el robot y se consiga el movimiento deseado. Para alcanzar este objetivo, se llevará a cabo una revisión bibliográfica exhaustiva para identificar las características de los software y lenguajes de programación compatibles con la tarjeta Robotis DYNAMIXEL U2D2. Se realizará un análisis comparativo entre las opciones disponibles, priorizando aquellos que sean de código abierto y se ajusten a los requerimientos específicos del trabajo.

En el análisis de la cinemática directa e inversa, se estudiarán distintos enfoques, como métodos geométricos y algoritmos en base a matrices de transformación homogénea. Este análisis permitirá comprender los límites y alcances del brazo robótico, es decir, los rangos admisibles de movimiento y la capacidad de posicionar el efector final según las indicaciones del usuario.

El proceso experimental será fundamental para verificar y validar el código programado y la respuesta de movimiento de cada servomotor. Se procederá a realizar conjuntamente la programación del sistema y desarrollando la interfaz gráfica de usuario, ya que el brazo robótico carece de este tipo de interfaz. Todo el proceso se realizará mediante una conexión directa entre la PC que actuará como cliente y el brazo robótico que será el servidor.

Por otro lado, para lograr el control de movimientos del brazo robótico de forma local y remota, se investigará sobre aplicaciones compatibles con el sistema operativo Windows, puesto que es el que se utilizará.

La combinación del enfoque investigativo con el desarrollo experimental permitirá alcanzar los objetivos propuestos en este trabajo, logrando una interfaz de comunicación efectiva y segura que facilite el control directo, local y remoto del brazo robótico PincherX 100. En la Figura 2.1, se muestra el diagrama de flujo que describe el proceso metodológico para el desarrollo de la presente investigación.

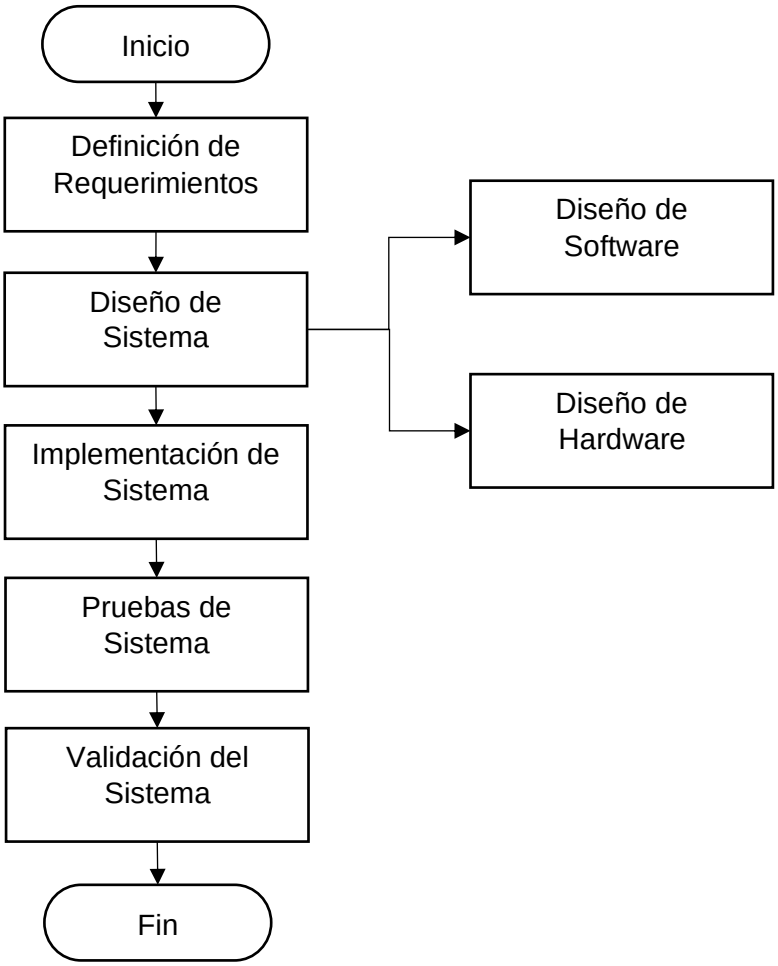


Figura 2.1. Diagrama de flujo – Metodología.

2.1. Definición de requerimientos

La presente investigación tiene como objetivo principal crear una interfaz de comunicación y una interfaz gráfica de usuario mediante el uso de softwares libres y de código abierto que permita controlar los movimientos de un brazo robótico de 4GDL, a fin de evitar incremento de costo de mantenimiento de la plataforma y el deslindamiento del software del fabricante. Para esto, se debe diseñar una interfaz gráfica de usuario intuitiva, en donde el operador no necesite de conocimientos profundos sobre robots o programación para poder mover el autómata, dicha interfaz constará con los modos de cinemática directa y cinemática inversas, y a su vez la comunicación podrá ser de forma local o remota, considerando que para ello la interfaz del computador principal o host deberá estar activada y conectada al brazo robótico.

2.1.1. Requerimientos de Software

Para el desarrollo de la interfaz de comunicación se requerirán varios softwares que servirán para crear el código en sí, y otras herramientas que permitirán establecer la comunicación entre los dispositivos. Mismos que se detallan a continuación:

- Sistema operativo compatible con las bibliotecas y herramientas necesarias para el desarrollo de la interfaz de comunicación.
- Software compatible con los servomotores DINAMIXEL para la calibración y monitorización de servomotores.
- Lenguaje de programación libre y código abierto.
- Plataforma de desarrollo que admita la implementación de algoritmos de cinemática directa e inversa.
- Software de comunicación remota que garantice la seguridad y la eficiencia en la transmisión de datos.

2.1.1.1. DYNAMIXEL Wizard 2.0

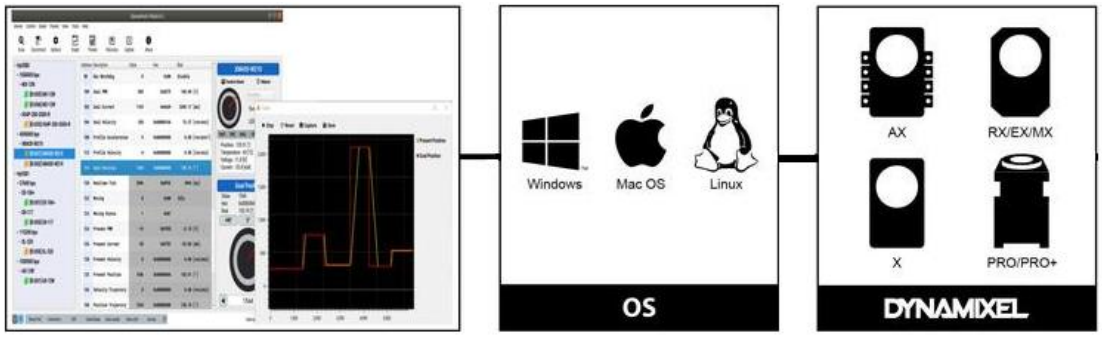


Figura 2.2. Software DYNAMIXEL Wizard 2.0 [28].

DYNAMIXEL Wizard 2.0 es un software que permite conocer y modificar las características de los actuadores de la serie DYNAMIXEL, es decir, permite conocer la identificación (ID), velocidad de giro, posición, graficas de variables, etc., de los servomotores. Es compatible con Windows, Linux y Macos [28].

Esta herramienta sirve principalmente para realizar un diagnóstico previo de la funcionalidad de los servomotores a utilizarse. Admite los protocolos denominados Protocolo 1.0 y Protocolo 2.0, propios de kit de DYNAMIXEL. Puesto que los servomotores con los que se trabajaran son específicamente de la serie X, ver Tabla 2.1., se aplicará el Protocolo 2.0.

Tabla 2.1. Protocolos de DYNAMIXEL [28].

Valor	Descripción	Compatible DINAMIXEL
1	Protocolo DYNAMIXEL 1.0	Serie AX, Serie DX, Serie RX, Serie EX, Serie MX con firmware inferior a v39
2 (predeterminado)	Protocolo DYNAMIXEL 2.0	MX-28/64/106 con firmware v39 o superior, Serie X, Serie PRO

2.1.1.2. DYNAMIXEL SDK

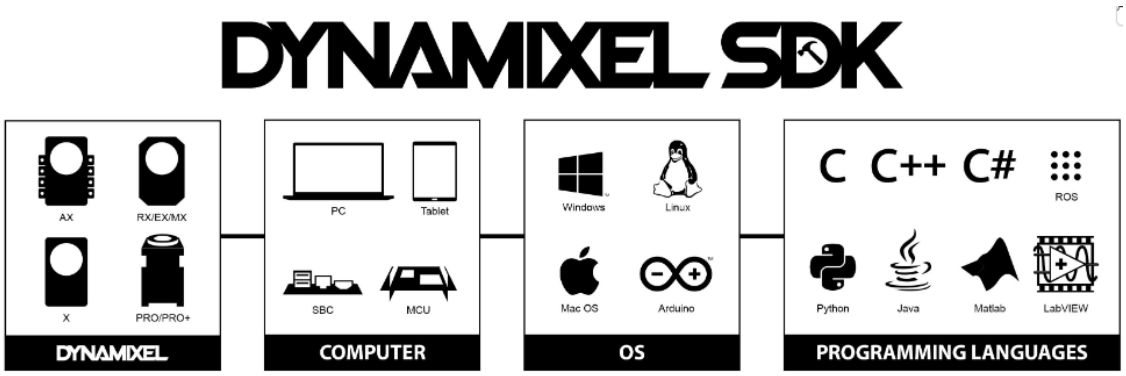


Figura 2.3. Kit DYNAMIXEL SDK [29].

DYNAMIXEL SDK es un paquete de desarrollo que se complementa tanto con el convertidor de comunicación DYNAMIXEL U2D2, así como también con los actuadores de la serie DYNAMIXEL.

Este paquete es compatible con varios sistemas operativos como son: Windows, Linux y Macos. Además, admite diferentes lenguajes de programación como: C, C++, C#, Python, Java, MATLAB y LabVIEW [29].

Esta herramienta permite realizar las mismas funciones que el software DYNAMIXEL Wizard 2.0 pero de forma independiente, es decir que se puede usar varias o todas sus funciones dependiendo del proyecto que se esté desarrollando, en si es una librería completa que permite controlar a los actuadores de la marca DYNAMIXEL o plataformas de dicha marca.

2.1.1.3. Sistema Operativo Windows

La elección de utilizar el sistema operativo Windows para este proyecto se fundamenta en su compatibilidad integral con el kit de desarrollo DYNAMIXEL SDK y el software DYNAMIXEL Wizard 2.0. Ambas herramientas son fundamentales para el control y la programación efectiva del brazo robótico. Windows ofrece un entorno de desarrollo

ampliamente utilizado, lo que facilita la obtención de soporte técnico y la resolución eficiente de problemas. Además, al ser compatible con las tecnologías y controladores específicos proporcionados por DYNAMIXEL, se garantiza una integración sin problemas entre el sistema operativo y el hardware, permitiendo una comunicación fluida y eficaz con los servomotores DYNAMIXEL.

2.1.1.4. Lenguaje de Programación, Python.

Los dispositivos de DINAMIXEL cuentan con su propia herramienta que es DYNAMIXEL SDK, admite diferentes lenguajes de programación que pueden ser pagados o de libre acceso. Los de código abierto son: C, C++, Python y Java.

Para definir el lenguaje de programación a utilizarse, se ha hecho una tabla de ponderaciones en base a seis características, mismas que son las necesarias y fundamentales para que se desarrolle y ejecute el proyecto en forma satisfactoria, las puntuaciones se muestran en la Tabla 2.2.

Tabla 2.2. Softwares de Código Abierto.

N°	Características	C (1-5)	C++ (1-5)	Python (1-5)	Java (1-5)
1	Abstracción y complejidad.	2	3	5	3
2	Uso en la automatización.	4	4	5	2
3	Facilidad de aprendizaje.	3	3	5	4
4	Gestión de la memoria.	2	3	5	4
5	Bibliotecas/Frameworks	3	4	5	4
6	Comunidad/ Soporte	4	4	5	4
	Total	**18**	**21**	**30**	**21**

En la Tabla 2.2., se puede observar que el lenguaje de programación que cumple con las características requeridas para el desarrollo del código es Python, debido a que presenta un alto nivel de abstracción y complejidad por lo que permite a los programadores trabajar en niveles altos sin la necesidad de enforcarse en los detalles internos. Así también tiene un buen desenvolvimiento al desarrollar plataformas de automatización, puesto que posee una gran variedad de bibliotecas y Frameworks. Al hablar de la facilidad de aprendizaje se hace referencia a que su código presenta una estructura de fácil comprensión. En los últimos años Python ha crecido considerablemente por lo que su comunidad de usuarios se ha extendido y a la vez han desarrollado soluciones a diferentes problemas o necesidades comunes entre los consumidores.

Por tanto, el lenguaje de programación a utilizar para el desarrollo de la presente investigación será Python, este permitirá escribir el algoritmo que permitirá controlar los movimientos del brazo robótico mediante la cinemática directa e inversa, así como también servirá para la creación de interfaz gráfica de usuario a través de códigos.

Python debe ser descargado desde el sitio oficial y en lo posible la última versión, puesto que las nuevas versiones presentan mejoras.

Es importante considerar las librerías de Python que se utilizarán, ya que estas deben estar instaladas previamente para el desarrollo del código. Mismas que se muestran a continuación:

- import pygame
- import math
- import openpyxl
- import time
- from dynamixel_sdk import *

A partir de las diferentes librerías se puede utilizar diferentes funciones específicas, como por ejemplo del paquete math, se puede usar las funciones de pi, cos, sin, entre otras.

2.1.1.5. Visual Studio Code

Figura 2.4. IDE Visual Studio Code [30].

Como ambiente de desarrollo integrado (IDE) se utilizará el software Visual Studio Code, este es un editor de código creado por Microsoft y está disponible para diferentes sistemas operativos como Windows, Linux y MacOS; y sobre todo permite trabajar con Python como lenguaje de programación [30].

2.1.1.6. Escritorio Remoto – Hamachi by LogMeIn

En cuanto a la comunicación de forma remota para el control de los movimientos del brazo robótico se han analizados las características fundamentales de diferentes softwares de escritorio remoto, mismo que se muestra en la Tabla 2.3.

Tabla 2.3. Softwares de Escritorio Remoto.

Software	Sistemas Operativos	Característica Principal	Limitación	Versión	Fuente
AnyDesk	Windows, macOS, Linux, Android, iOS.	Alta calidad de imagen, rendimiento fluido, fácil de usar, latencia mínima.	Tiempo limitado de sesión, número limitado de dispositivos.	Versión gratuita.	www.anydesk.com/es
TeamViewer	Windows, macOS, Linux, Android, iOS and iPadOS, SO Chrome, Web Services.	Amplia adopción, transferencia de archivos, reuniones en línea.	Funciones de seguridad insuficientes.	Versión gratuita.	www.teamviewer.com
Chrome Remote Desktop	Windows, Linux, macOS o Chrome OS.	Integrado con Google Chrome, fácil de configurar.	Requiere instalación de navegador Chrome.	Versión gratuita.	www.remotedesktop.google.com/
Remote Desktop Manager	Windows, macOS, Linux.	Gestión centralizada de conexiones remotas, contraseñas y credenciales.	Características avanzadas como base de datos compartidas requiere de pagos.	Versión gratuita.	www.devolutions.net/remote-desktop-manager/
Hamachi by LogMeIn	Windows, macOS, Linux.	Creación de VPN para conexiones seguras.	Limitado número de dispositivos en red VPN.	Versión gratuita.	www.logmein.com/es

Una vez analizadas las diferentes características de varios softwares de escritorio remoto, considerando que deben ser de uso gratuito, es decir debe permitir el uso de versiones gratuitas. Se ha elegido al programa para computador Hamachi by LogMeIn puesto que, a diferencia de los otros, este ofrece una alta seguridad debido a que trabaja con una red VPN, es decir los datos están encriptados de extremo a extremo y la comunicación de forma remota no podrá ser filtrada o interferida.

2.1.2. Requerimiento de Hardware

A continuación, se detalla los requerimientos específicos en lo referente a hardware:

- Tarjeta Robotis DYNAMIXEL U2D2 para la comunicación entre el computador principal o host con los servomotores.
- Brazo robótico PincherX 100 con 4GDL y servomotores inteligentes.
- Computadora con procesador Core i7 de 64 bits, memoria RAM de al menos 8GB y un disco duro sólido, para garantizar la fluides de los datos y control en tiempo real. Además, debe contar necesariamente con los puertos Ethernet y USB; y, tecnología WiFi.
- Fuente de alimentación con capacidad adecuada para soportar el funcionamiento continuo de los servomotores. Y cables en estado óptimo para su correcto funcionamiento.

2.1.2.1. Brazo Robótico PincherX 100

En esta investigación se utilizará un brazo robótico PincherX 100, puesto que es didáctico y muy utilizado en los ámbitos académico e investigativo, considerando además la relación entre costo y funcionalidad. Su material estructural es acrílico, cuenta con 5 servomotores DYNAMIXEL XL430-W250-T de los cuales 4 están posicionados en las articulaciones y el quinto sirve para la apertura o cierre de la pinza, estos poseen su fuente de alimentación independiente de 12 volteos. El convertidor U2D2, permite la comunicación entre los servomotores y el ordenador.

Control en lazo cerrado

Los servomotores DYNAMIXEL XL430-W250-T, son actuadores inteligentes que tienen integrado un controlador PID, este realiza una comparación constante entre la posición real y la posición deseada, esto con el fin de ajustar la señal de control y llegar a la posición requerida o enviada por el usuario. Considerando que, además proporciona una respuesta rápida y estable frente a cambios o perturbaciones del sistema. Por tanto, permite tener un sistema preciso y eficiente en lazo cerrado, en donde el operador envía la posición requerida a los actuadores, y el controlador analiza la posición actual con la requerida de forma cíclica a fin de conseguir la posición enviada inicialmente.

2.1.2.2. Computador

Con respecto al ordenador o computador, debe poseer una capacidad de procesamiento óptima y un espacio suficiente en la memoria para ejecutar eficientemente el código en

Python y los softwares necesarios para el control del brazo robótico. Es importante tener en cuenta que Python, el lenguaje de programación utilizado, no demanda grandes cantidades de memoria en el computador.

Dado que se utilizarán diferentes tipos de conexión con el brazo robótico, el computador debe estar equipado con una variedad de puertos y capacidades de comunicación. Se sugiere que el computador cuente con los siguientes elementos para complementar las características de acuerdo con las necesidades del proyecto:

- **Puerto Ethernet:** Permite la conexión por cable, proporcionando una comunicación estable y de alta velocidad, lo cual es fundamental para la transmisión eficiente de datos al brazo robótico, especialmente en aplicaciones donde la latencia es crítica.
- **Wi-Fi (Conexión inalámbrica):** Permite la comunicación inalámbrica entre el computador y el brazo robótico. Esta opción es útil para facilitar la movilidad y flexibilidad en la ubicación del computador en relación con el robot.
- **Puertos USB:** Deben incluir múltiples puertos USB para conectar periféricos y dispositivos de entrada/salidas adicionales que puedan ser necesarios para el proyecto, como cámaras, sensores u otros componentes.
- **Memoria RAM adecuada:** Una cantidad suficiente de RAM es esencial para el funcionamiento fluido de las aplicaciones y el manejo eficiente de los datos que se procesan durante el control del brazo robótico.
- **Unidad de almacenamiento espaciosa:** Se requiere un disco duro o unidad de almacenamiento con suficiente espacio para guardar el software, los datos y el código necesario para el control del brazo robótico, así como para almacenar los registros y resultados de las pruebas.
- **Procesador potente:** Un procesador con múltiples núcleos y una velocidad de reloj adecuada para garantizar un rendimiento eficiente durante la ejecución de aplicaciones y algoritmos complejos de control.
- **Sistema operativo compatible:** Un sistema operativo que sea compatible con el software de control y las aplicaciones requeridas para el brazo robótico.

Al tener en cuenta y satisfacer estas características y elementos en el computador, se garantiza que este sea capaz de procesar los datos y ejecutar las operaciones necesarias para controlar el brazo robótico de forma local y remota de manera efectiva y eficiente.

A continuación, en la Tabla 1.4., se muestra las características del computador principal utilizado:

Tabla 0.4. Características del computador principal o host.

Tipo de computador	Computar de escritorio
Procesador del Sistema	Intel(R) Core (TM) i7
Tipo de Sistema	64 bits
Sistema Operativo	Windows 10 Home
Memoria RAM	8,0 GB
Resolución	Alta definición (HD) 1920 x 1080

2.1.2.3. Cables

Para la puesta en marcha del presente proyecto, se requerirán varios tipos de cables estos deben estar en buenas condiciones para evitar fallas al momento de realizar las pruebas de funcionamiento. Los cables que se utilizarán son: cables Ethernet para la conexión local y remota, así como también el cable USB para conectar el convertidor DYNAMIXEL U2D2 con el computador.

2.2. Diseño del sistema

En la Figura 2.5. se observa la consolidación general del sistema, en donde se puede diferenciar las diferentes redes a utilizarse según el tipo de comunicación.

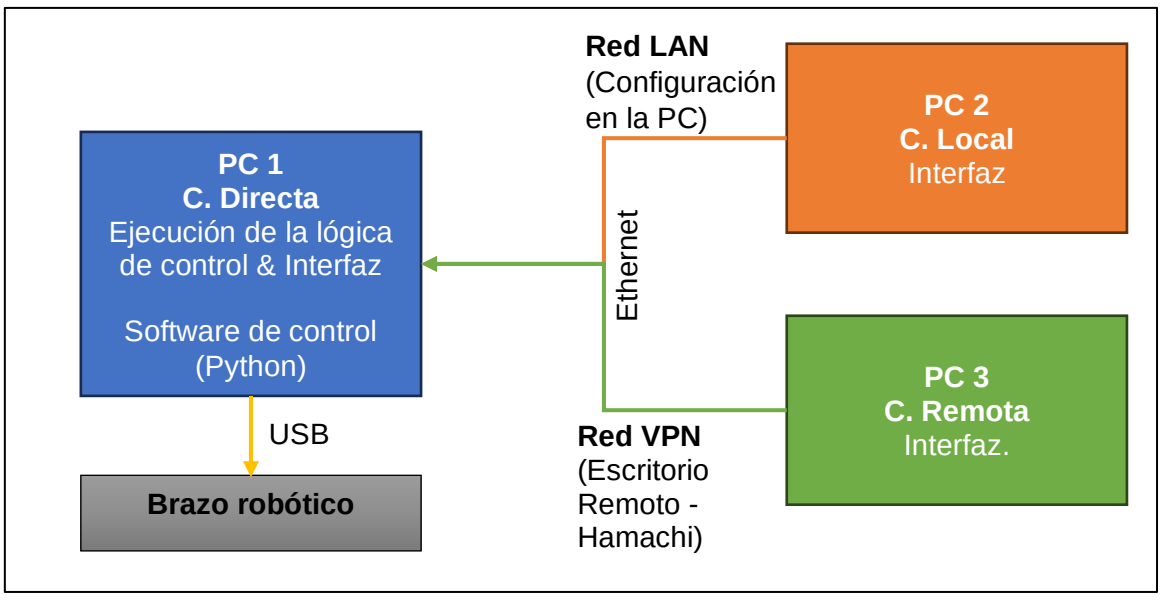


Figura 2.5. Diseño del Sistema.

2.2.1. Nivel de control

En esta capa se encuentra instalado el software para el control del brazo robótico. Incluye el desarrollo de la lógica de control, algoritmos de la cinemática directa e inversa, así como también las instrucciones de movimiento. La programación se realizará en Python, esto permitirá manejar la comunicación directa con el brazo robótico y desarrollar la interfaz gráfica de usuario.

2.2.2. Nivel de Comunicación

- **Comunicación Directa**

En este nivel se establecerá la comunicación directa con el brazo robótico a través del convertidor USB DYNAMIXEL U2D2 que permite comunicar a los servomotores con el computador.

- **Comunicación Local**

En este nivel se realizará la comunicación local con el brazo robótico a través de una red LAN, esta red permite la comunicación entre dispositivos que están próximos o ubicados en un mismo espacio geográfico puesto que están en la misma red. La formación de la red LAN se lo hará mediante el uso del protocolo Ethernet o mediante la tecnología Wifi, de ser necesario se utilizará un conmutador de red ethernet (switch).

- **Comunicación Remota**

Este nivel será responsable de permitir la comunicación remota del brazo robótico a través de la red. Aquí es donde se utilizará una red VPN creada por Hamachi. La VPN garantizará la seguridad de la conexión y permitirá la comunicación segura entre los dispositivos en diferentes ubicaciones geográficas.

2.2.3. Nivel de Interfaz de Usuario

Este nivel contiene a la interfaz gráfica de usuario que será utilizada para controlar el brazo robótico, misma que se desarrollará utilizando la librería pygame propia de Python. La interfaz permitirá la selección de movimientos, configuración de parámetros y visualización en tiempo real de la posición del brazo. Esta interfaz será accesible en el computador principal donde estará funcionando la lógica de control.

2.2.4. Nivel de Comunicación entre Computadores

La arquitectura del sistema se alinea con los principios fundamentales de las redes, puesto que los computadores deben comunicarse entre sí para transmitir señales de la posición actual del robot, manteniendo un control remoto seguro y una gestión centralizada, este último debido a que simultáneamente se pueden controlar varios sistemas de forma remota, estando en espacios geográficos totalmente distanciados. Estos aspectos no solo optimizan el rendimiento del sistema, sino que también sientan las bases para la evolución futura de soluciones automatizadas en entornos industriales exigentes. La implementación de estas redes representa un paso significativo hacia la eficiencia, la seguridad y la versatilidad en el ámbito de la automatización industrial. La comunicación será gestionada a través de una red LAN y una red VPN, para la comunicación local y remota respectivamente.

2.2.5. Plataforma Robótica

Es el brazo robótico que puede ser controlado desde diferentes computadores, según el tipo de conexión, ya sea directo, local o remoto. Este recibe comandos y proporciona datos de posición a través de las conexiones establecidas. La trasmisión y recepción de datos, o comunicación bidireccional entre el computador y la plataforma robótica, es posible gracias a los servomotores que posee el autómata, puesto que son actuadores inteligentes que poseen sensores embebidos que permiten controlar la posición, velocidad, torque y temperatura interna.

2.2.5.1. Especificación de eslabones y ángulos del brazo robótico PincherX 100.

En la Figura 2.6. Se muestra de color azul los cuatro eslabones con los que se debería trabajar el análisis cinemático considerando 4 GDL. Sin embargo, como se puede observar, el eslabón L1 no pertenece a una pieza física, sino más bien a la hipotenusa de dos elementos que están ensamblados de forma fija, por lo que el triángulo rectángulo (marcado en color rojo) formado entre dichos elementos siempre mantendrán el ángulo recto de 90°.

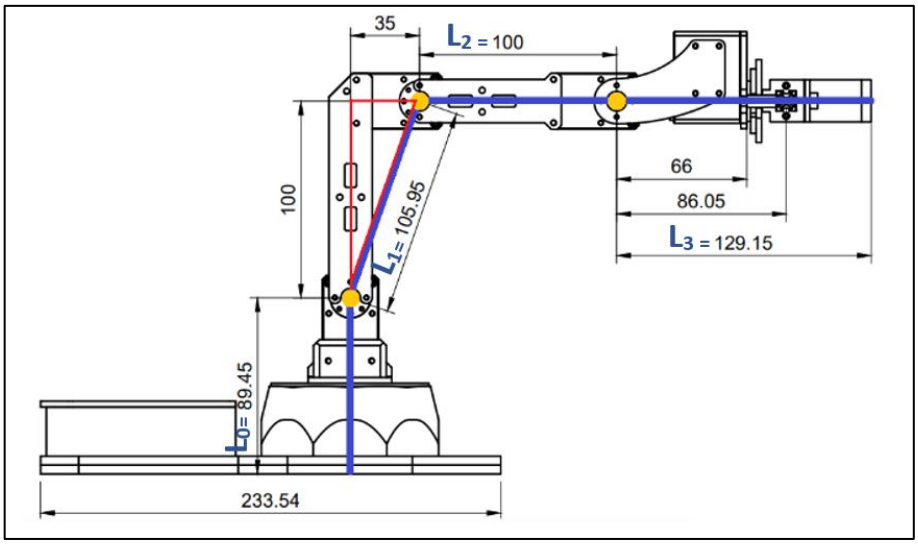


Figura 2.6. Eslabones del brazo robótico considerando 4GDL.

Para trabajar con el eslabón L1 de la Figura 2.6., es necesario considerar las variaciones angulares generadas por el ensamble fijo entre eslabones. Por tanto, es necesario determinar los ángulos θ y α, ver Figura 2.7, puesto que en relación con estos será el movimiento angular real de los elementos físicos y a su vez de los servomotores.

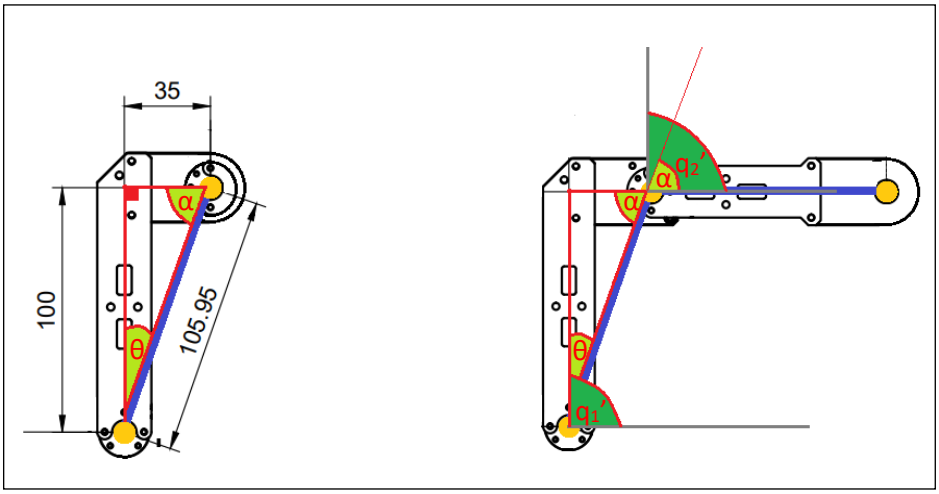


Figura 2.7. Ángulos auxiliares.

Por tanto, para omitir el uso de angulos auxiliares θ y α, se trabajará con las medidas reales de la plataforma robótica, ver Figura 2.8, es decir con las medidas propias de los 5 eslabones, esto ocasiona que se genere un grado de libertad adicional, mismo que corresponderá al ángulo formado entre las piezas ensambladas de forma fija, es decir a 90°.

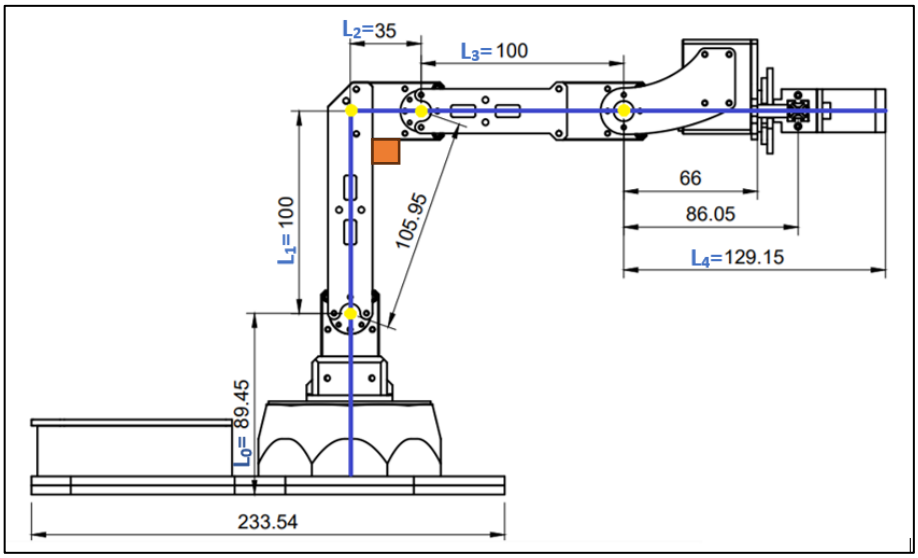


Figura 2.8. Eslabones del brazo robótico considerando 5GDL.

2.2.5.2. Trayectoria del brazo robótico

Para describir la trayectoria del brazo robótico en función de las medidas de los eslabones y las coordenadas angulares permisibles, es necesario entender cómo influyen estas variables en la posición y orientación del extremo del brazo.

- **Medidas de los eslabones (longitudes):** Las longitudes de los eslabones son parámetros clave que determinan la amplitud de movimiento y la extensión máxima del brazo robótico. Cada eslabón, con su longitud específica, contribuirá a la longitud total del brazo. Cuanto más largo sea un eslabón, mayor será el alcance del brazo en esa dirección. Sin embargo, se debe considerar que no es recomendable extender el 100% de la extensión admitida, puesto que, debido al propio peso de la estructura la plataforma podría volcarse al momento que esta esté en movimiento.
- **Coordenadas angulares permisibles:** Las coordenadas angulares, representadas por los ángulos de articulación de cada eslabón, influyen en la dirección y el ángulo de inclinación del brazo robótico en el espacio tridimensional. Las variaciones en estos ángulos modificarán la posición y orientación del extremo del brazo. Los límites de estos ángulos deben respetarse para evitar movimientos fuera del rango seguro y eficiente.

En la Tabla 2.5., se indica las medidas de los eslabones y los ángulos utilizados en el desarrollo del presente proyecto. En la Figura 2.9. se muestra el sistema de referencia

considerado, puesto que los valores van desde 180° hasta -180°, mas no considerando un intervalo de 0° a 360°.

Tabla 2.5. Medida de eslabones y rangos angulares.

Articulación	Eslabón (mm)	Ángulo mínimo (°)	Ángulo Máximo (°)
Cintura	89.45	-180	180
Brazo	100	17.4	180
Antebrazo	100	-149	0
Muñeca	129.15	-102.9	74.4

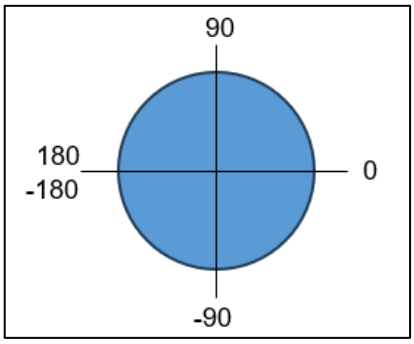


Figura 2.9. Sistema de referencia angular.

La trayectoria del brazo robótico se compone de una serie de puntos en el espacio tridimensional, donde cada punto representa la posición y orientación del extremo del brazo en un momento específico. Estos puntos forman una curva o línea en el espacio tridimensional que representa la trayectoria; dicho conjunto de puntos forma el espacio de trabajo, mismo que consiste en todas las posibles posiciones que puede adoptar el brazo robótico considerando los ángulos admisibles para cada nodo, en la Figura 2.10. se muestra es espacio de trabajo para la plataforma robótica PincherX 100.

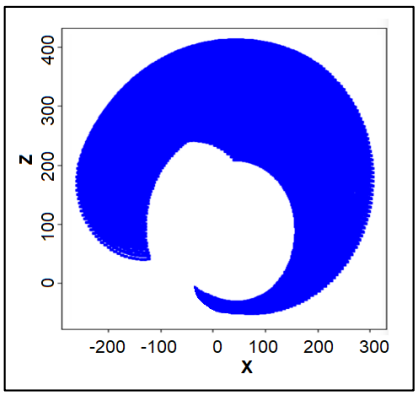


Figura 2.10. Espacio de Trabajo – Brazo Robótico PincherX 100.

- **Movimiento del brazo en el plano XY:** Cambiar las coordenadas angulares en los planos XY afectará la posición del extremo del brazo en ese plano. Ajustar los ángulos de articulación permitirá al brazo moverse en el plano XY a lo largo de una trayectoria específica, variando las longitudes de los eslabones también afectará la trayectoria.
- **Movimiento en la coordenada Z:** Cambiar las coordenadas angulares que influyen en la coordenada Z afectará la altura a la que el brazo robótico puede llegar. Aquí, nuevamente, las longitudes de los eslabones también influirán en esta trayectoria en la dimensión Z.

2.2.6. Cinemática Directa

La cinemática directa permite conocer la posición del efector final o extremo del brazo robótico tomando como referencia las posiciones angulares de cada una de sus articulaciones giratorias. Existen varias formas para resolver este tipo de problemas, a continuación, analizaremos específicamente el método geométrico y el algoritmo de Denavit Hartenberg puesto que son los más utilizados.

2.2.6.1. Método geométrico

Este método utiliza relaciones geométricas entre los eslabones y la posición de estos para determinar los comportamientos angulares, haciendo uso de las funciones trigonométricas. Este método generalmente se aplica en manipuladores que tengan pocos grados de libertad, puesto que es sencillo entender y deducir las relaciones geométricas y trigonométricas existentes.

Para el caso de estudio, se analizará al brazo robótico PincherX 100 de 4 GDL, considerando el grado adicional antes mencionado que será denotado como q_x y tiene el valor angular de -90°, ver Figura 2.11. Inicialmente se hace un análisis de la posición de los eslabones en un plano **r** y en el eje Z, para después proyectar dicho plano a los ejes reales de trabajo, es decir a los ejes X y Y.

En la Figura 2.11. se observa que, para conseguir la proyección en r del primer eslabón L_1 basta con conocer el ángulo articular q_1. Por su parte, para conocer la proyección del segundo eslabón L_2 se trabajará con la suma de los ángulos q_1 y q_x, puesto que al estar conectados en serie los componentes dependerán uno del otro consecutivamente. Dicho esto, de la misma forma, para la proyección en r del tercer eslabón L_3 se utilizará la suma de los tres ángulos q_1, q_2 y q_x, de la misma forma para el eslabón L_4 se trabajará con la

suma de los ángulos q_1, q_2, q_3 y q_x. Una vez conocida la proyección de mi resultante r en el plano r, esta se proyectará a los ejes X y Y utilizando únicamente el ángulo q0, como se muestra en la Figura 2.12.

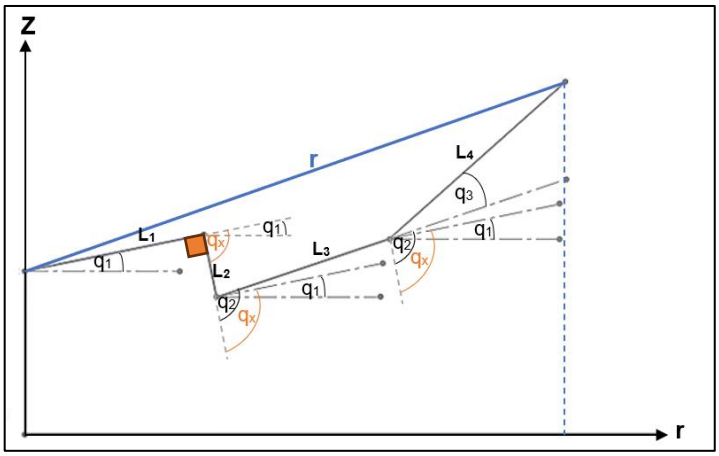


Figura 2.11. Proyección en el plano r.

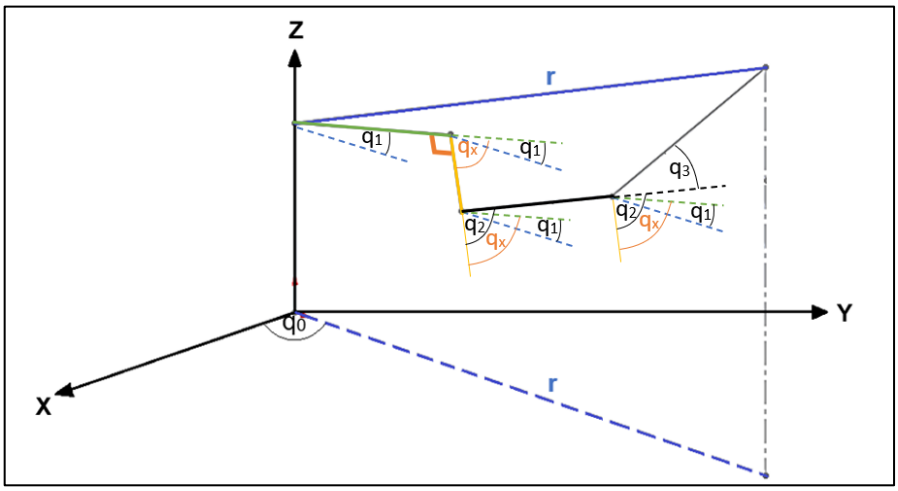


Figura 2.12. Proyección de la resultante r en los tres ejes.

Una vez analizada la geometría y comportamiento de los eslabones y ángulos en los diferentes planos, las ecuaciones que rigen la cinemática directa para el modelo estudiado son:

$$R = L_1.\cos(q_1) + L_2.\cos(q_1 + q_x) + L_3.\cos(q_1 + q_x + q_2) + L_4.\cos(q_1 + q_2 + q_3 + q_x) \quad (1.3)$$

$$Z = L_0 + L_1.\sin(q_1) + L_2.\sin(q_1 + q_x) + L_3.\sin(q_1 + q_x + q_2) + L_4.\sin(q_1 + q_2 + q_3 + q_x) \quad (1.4)$$

$$X = R * \cos(q_0) \quad (1.5)$$

$$Y = R * \sin(q_0) \quad (1.6)$$

En el Anexo A se muestra el código en Python en donde se aplica el método geométrico para resolver la cinemática directa.

2.2.6.2. Algoritmo de Denavit Hartenberg

El análisis del brazo robótico PincherX 100 con 4GDL, considerando además la constante angular adicional q_x, se muestra su análisis estructural y angular en la Figura 2.13, se puede observar la posición y orientación de los sistemas de referencia en cada nodo, así como también las variables correspondientes a las coordenadas angulares $(q0, q1, q2, q3)$, las constantes pertenecientes a las medidas de cada eslabón $(L0, L1, L2, L3)$ y la constante angular qx.

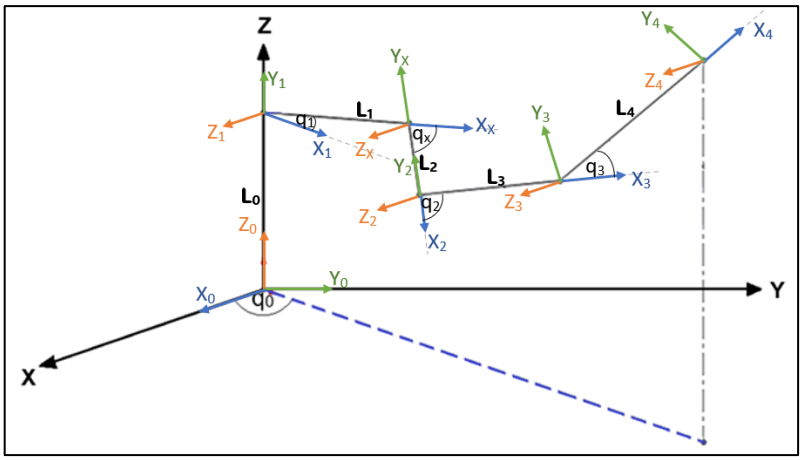


Figura 2.13. Sistemas de referencia en cada nodo.

Para definir el sentido de los ejes en los sistemas de referencia mostrados en la Figura 2.13., primero se debe colocar al eje Z en dirección del eje de giro. A continuación, se definirá el sentido del eje X en base a los casos que se muestran a continuación.

Caso 1: X_i será perpendicular al plano formado por Z_i y Z_{i-1}.

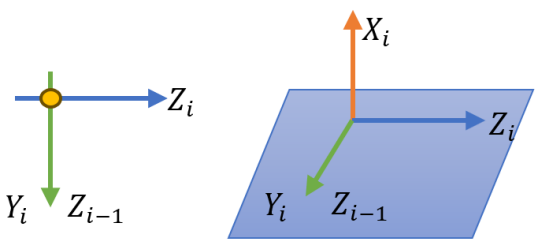


Figura 2.14. Condición 1 - Sistemas de referencia.

Caso 2: X_i será perpendicular a Z_i y a Z_{i-1}.

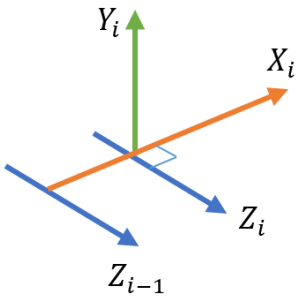


Figura 2.15. Condición 2 - Sistemas de referencia.

El caso uno aplica en el nodo 0, el resto de los nodos corresponden al caso 2.

Mientras que para definir el sentido del eje Y en cada nodo, se debe aplicar la regla de la mano derecha.

Una vez establecidos los sistemas de referencia en cada nodo, ya es posible determinar los parámetros Denavit Hartenberg θ, d, a y α , los cuales se muestran en la Tabla 2.6.

Tabla 2.6. Parámetros Denavit Hartenberg.

N°	Articulación	$\boldsymbol{\theta}$	$\boldsymbol{d}$	$\boldsymbol{a}$	$\boldsymbol{\alpha}$
1	0	q_0	L_0	0	90°
2	1	q_1	0	L_1	0°
3	X	$q_x = -90°$	0	L_2	0°
4	2	q_2	0	L_3	0°
5	3	q_3	0	L_4	0°

El nodo X corresponde al acople fijo entre los eslabones L1 y L2, cuyo ángulo constante será igual a -90° y está denotado como q_x, ver Figura 2.13.

Para definir las matrices de transformación individual que rigen en cada articulación de la plataforma robótica estudiada, se debe reemplazar los parámetros obtenidos en la Tabla 2.6. en la expresión (1.2) mostrado en la sección 1.4.6.1; es decir que con los datos de cada fila se formará una matriz de tipo A_i^{i-1}.

Por tanto, para el caso de estudio serán un total de cinco matrices de transformación individual, dicho valor corresponde a la cantidad de grados de libertad de la plataforma

robótica analizada. Siendo 4 GDL propios del autómata y 1GDL adicional debido al ángulo fijo q_x, quedando estás de la siguiente manera:

$$A_1^0 = \begin{bmatrix} C_0 & 0 & S_0 & 0 \\ S_0 & 0 & -C_0 & 0 \\ 0 & 1 & 0 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_x^1 = \begin{bmatrix} C_1 & -S_1 & 0 & L_1C_1 \\ S_1 & C_1 & 0 & L_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^x = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -L_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.7)$$

$$A_3^2 = \begin{bmatrix} C_2 & -S_2 & 0 & L_3C_2 \\ S_2 & C_2 & 0 & L_3S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4^3 = \begin{bmatrix} C_3 & -S_3 & 0 & L_4C_3 \\ S_3 & C_3 & 0 & L_4S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Seguidamente, se calcula la matriz de transformación total T o también denominada matriz de transformación homogénea, esta permite definir la posición del efector final con respecto al sistema de referencia del nodo inicial ubicado en la base fija del manipulador. La matriz T se consigue de la multiplicación de las matrices de transformación individual mostradas anteriormente, quedando expresada de la siguiente forma:

$$\boldsymbol{T = A_1^0\ A_x^1\ A_2^x\ A_3^2\ A_4^3}$$

Las tres primeras filas de la matriz T permiten definir la rotación y posición del efector final, siendo la última columna la que mostrará las coordenadas cartesianas de la pinza: $\boldsymbol{P_x, P_y}$ y $\boldsymbol{P_z}$, como se muestra en la ecuación 1.7. Mientras que, la última fila se conservará como [0 0 0 1], en donde los ceros representan la perspectiva y el uno corresponde a la escala, consiguiendo así que sea factible realizar la multiplicación entre matrices de tamaño 4x4 [31].

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{s_x} & \boldsymbol{a_x} & \boldsymbol{P_x} \\ \boldsymbol{n_y} & \boldsymbol{s_y} & \boldsymbol{a_y} & \boldsymbol{P_y} \\ \boldsymbol{n_z} & \boldsymbol{s_z} & \boldsymbol{a_z} & \boldsymbol{P_z} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \quad (1.8)$$

A continuación, se muestran las expresiones que corresponden a los elementos $\boldsymbol{P_x}, \boldsymbol{P_y}$ y $\boldsymbol{P_z}$ de la matriz T del brazo robótico PincherX 100.

$$P_X = L1.C_0.C_1 + L2.S_1.C_0 + L3.S_1.C_0.C_2 + L3.S_2.C_0.C_1 + L4.S_3(-S_1.S_2.C_0 + C_0.C_1.C_2) + L4.C_3(S_1.C_0.C_2 + S_2.C_0.C_1) \quad (1.9)$$

$$P_Y = L1.S_0.C_1 + L2.S_0.S_1 + L3.S_0.S_1.C_2 + L3.S_0.S_2.C_1 + L4.S_3(-S_0.S_1.S_2 + S_0.C_1.C_2) + L4.C_3(S_0.S_1.C_2 + S_0.S_2.C_1) \quad (1.1)]$$

$$P_Z = L0 + L1.S_1 - L2.C_1 + L3.S_1.S_2 - L3.C_1.C_2 + L4.C_3(S_1.S_2 - C_1.C_2) + L4.S_3(S_1.C_2 + S_2.C_1) \quad (1.11)$$

Se hace la aclaración que la letra S representa la función de $seno$ y las letras C corresponde a la función de $coseno$. Así mismo los subíndices ubicados junto a estas letras representan los valores de θ indicados en la Tabla 2.6. Por ejemplo, la expresión S_0 se refiere al seno del ángulo q_0.

Por tanto, las expresiones de los elementos Px, Py y Pz de la matriz de transformación total T son las que permiten determinar la posición del efector final en los ejes X, Y y Z respectivamente.

En el Anexo B, se muestra el código en Python para calcular la cinemática directa mediante el algoritmo de Denavit Hartenberg. Y, en el Anexo C, se pueden observar todos los elementos de la matriz de transformación total T.

2.2.7. Cinemática Inversa

La cinemática inversa, en contraposición a la directa, es un proceso que permite calcular las posiciones angulares de cada articulación en función de las coordenadas del efector final del robot. Es esencial comprender que las primeras tres variables angulares determinan la posición espacial del extremo del robot, mientras que las variables restantes se enfocan en definir la orientación o la rotación de dicho extremo.

En muchas de las ocasiones, al aplicar métodos geométricos, matriciales o algoritmos, no se logra llegar a una solución única, e incluso, en ocasiones, no hay una respuesta posible. En tales circunstancias, es necesario recurrir a métodos numéricos que, aunque son efectivos, consumen más tiempo para procesar los datos y alcanzar una solución.

Para el caso de la plataforma robótica PincherX 100, dado que cuenta con 4GDL propios de la plataforma y 1GDL adicional debido al ensamble fijo entre estabones, es decir un total de 5GDL, se vuelve complejo aplicar métodos geométricos, matriciales o algoritmos con

los que probablemente no se llegue a una solución. Por tal razón, se utilizará herramientas propias de Python, puesto que permiten llegar a una solución óptima y aprovechar los recursos existentes y creados por los usuarios que utilizan dicho lenguaje de programación.

Las herramientas utilizadas son:

- **scipy. optimize:** es una biblioteca propia de Python que permite resolver diferentes problemas matemáticos de carácter científico como: ecuaciones lineales, no lineales, mínimos, cuadrados, regresiones, etc., permitiendo específicamente hallar o definir la solución más optima, lo cual es de gran ayuda considerando que al trabajar en la cinemática inversa pueden existir varias soluciones, esta función es clave para definir el mejor resultado. A su vez, scipy. optimize presenta la función **fsolve,** misma que es utilizada para encontrar las raíces de un sistema de ecuaciones [32].
- **envolverTo2pi:** esta función permite que los ángulos oscilen entre - π y π, esto con el fin de evitar errores o golpes al momento de enviar datos al brazo robótico [33].

La resolución de la cinemática inversa se lo hace a partir de las ecuaciones definidas en la cinemática directa, y mediante las herramientas de Python se realiza la resolución de ecuaciones en forma de regresión mediante el uso de la función **fsolve**.

El código correspondiente a este apartado se muestra en el Anexo D.

2.2.8. Interfaz Gráfica de Usuario

La creación de la interfaz gráfica de usuario es fundamental en la presente investigación puesto que es el medio tangible a través de la cual los operadores pueden controlar los movimientos de brazo robótico de forma intuitiva, sin necesidad de inmiscuirse o modificar el código desarrollado en Python.

La interfaz gráfica de usuario representa la complejidad de toda la programación de algoritmos, comandos, formulas, etc., mostradas de forma sencilla en una sola ventana, en donde el usuario mediante elementos visuales como botones, indicadores, gráficos y representaciones en tiempo real, es capaz de enviar datos al brazo robótico y poder observar su posición y trayectoria antes, durante y después del envió de dichos comandos.

Al diseñar y describir esta interfaz, se busca no solo la funcionalidad y eficiencia, sino también la usabilidad e intuición del usuario. La disposición lógica de los elementos, el uso de colores, la claridad de las etiquetas y la capacidad de respuesta son elementos clave

en la creación de una interfaz gráfica que permita a los usuarios interactuar de manera fluida y efectiva con el brazo robótico, independientemente de su nivel de experiencia.

A continuación, se explorará con mayor detalle la estructura, funciones y elementos que componen la interfaz gráfica de usuario a utilizarse en la presente investigación, mostrando cómo se convierte la complejidad del control de un brazo robótico en una experiencia visualmente coherente y comprensible para los usuarios.

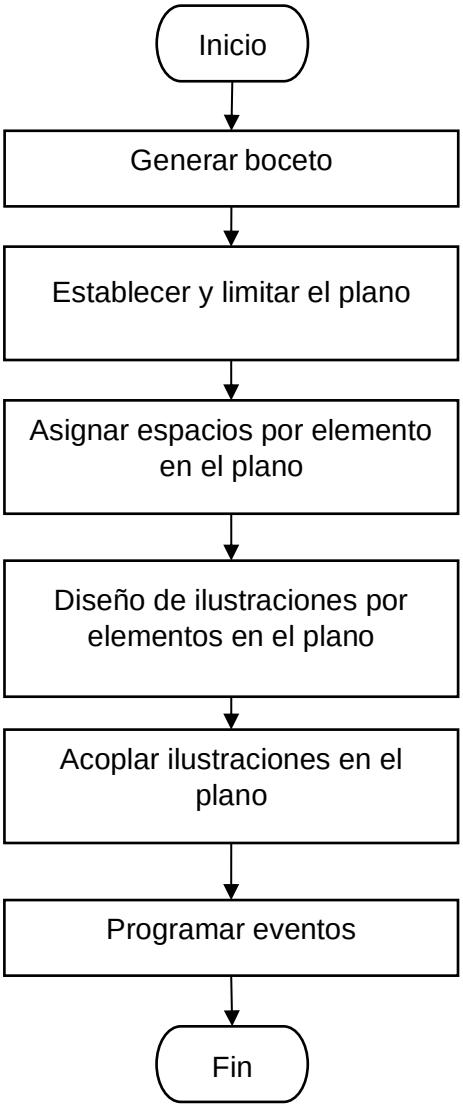


Figura 2.16. Diagrama de flujo creación y diseño interfaz gráfica de usuario.

2.2.8.1. Generar boceto

La creación del boceto permite definir los elementos necesarios para una manipulación adecuada de la interfaz gráfica de usuario, así como también ayuda a precisar la proporcionalidad y funcionalidad de dichos elementos, de tal forma que los iconos e imágenes sean posicionados y agrupados en forma ordenada y coherente.

La ventana principal estará dividida en dos partes principales, como se muestra en la Figura 2.17. En la primera parte se mostrará la vista isométrica del brazo robótico PincherX 100, esto con el objetivo de que se visualicen los cinco servomotores que serán controlados por el usuario, además, el ángulo de giro para cada uno de ellos se lo podrá modificar de forma gráfica y a través Spin Box. En la segunda mitad se visualizará la vista en 3D de cada uno de los eslabones, así como también la vista lateral y superior de los mismos, además se mostrará la apertura o cierre de la pinza, todo esto con el objetivo de apreciar claramente el comportamiento y movimientos que realiza el brazo robótico durante su operación.

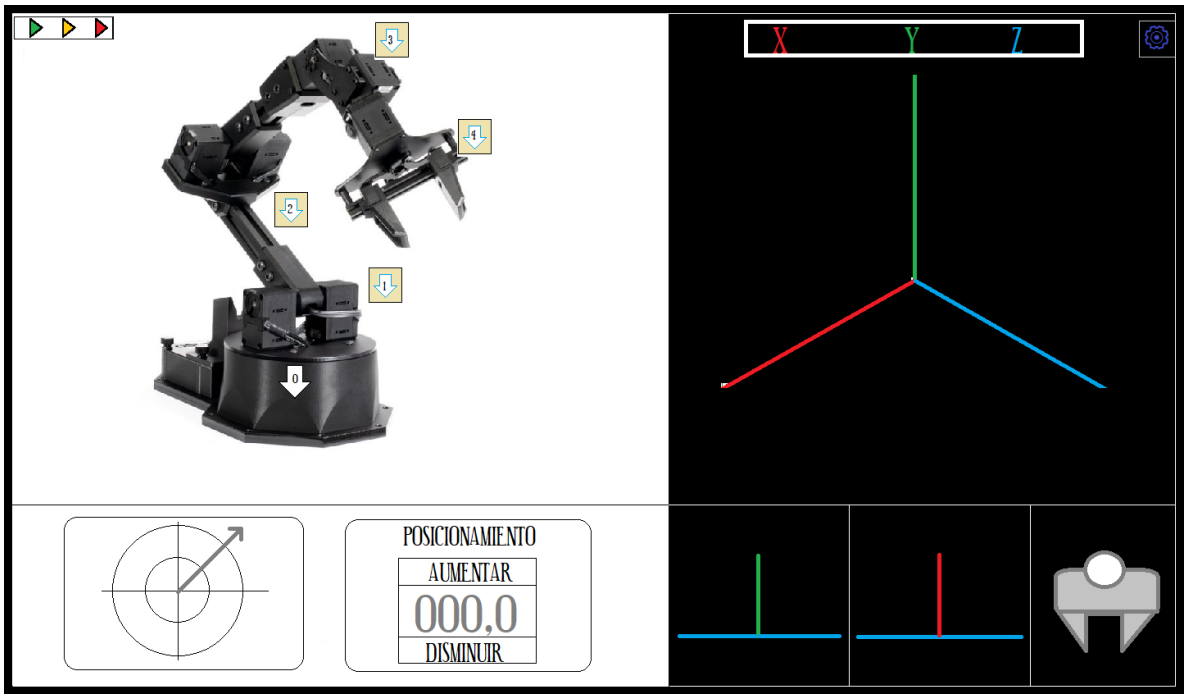


Figura 2.17. Generar boceto.

2.2.8.2. Establecer y limitar el plano

En lo referente a monitores para PC se los puede diferenciar principalmente por su resolución, misma que se mide en pixeles, de esto depende la nitidez y claridad de las imágenes que se observen [34], en la Tabla 2.7. se muestra las resoluciones más utilizadas.

Tabla 2.7. Resolución de Monitores.

Denominación	Resolución (pixeles)
Matriz de gráficos super extendida (SXGA)	1280 x 1024
Alta definición (HD)	1366 x 768
Alta definición Plus (HD+)	1600 x 900
Alta definición Total- Full HD- (FHD)	1920 x 1080

La resolución que tiene el computador a utilizarse para el desarrollo de la interfaz gráfica de usuario tiene 1366 x 768 pixeles, considerando además que este formato HD es uno de los más utilizados por los usuarios. En consecuencia, el espacio de trabajo no debe exceder el tamaño antes mencionado a fin de garantizar la visualización completa de la interfaz gráfica. Por tanto, se ha considerado 1366 pixeles de ancho y 768 pixeles de alto para el área de trabajo, esta será repartida en forma proporcional según la dimensión de los elementos a fin de conseguir que se visualicen todos los iconos, imágenes y animaciones de forma ordenada y entendible para el usuario. Los espacios para cada elemento se distribuirán como se indica en la Figura 2.18.

Figura 2.18. Establecer y limitar el plano.

2.2.8.3. Asignar espacios por elemento en el plano

Una vez definidos los elementos principales a utilizarse y las dimensiones del área de trabajo, se procede a establecer la posición y el espacio que ocuparan cada uno de los elementos en base al boceto inicial. Como se puede observar en la Figura 2.19., sobre la vista isométrica del brazo robótico se colocará animaciones que permitan controlar los ángulos de cada servomotor; por su parte en la segunda mitad se visualizará la vista en 3D del comportamiento del brazo robótico en tiempo real, así como también las coordenadas cartesianas del efector final, y las vistas superior y lateral de todo el conjunto.

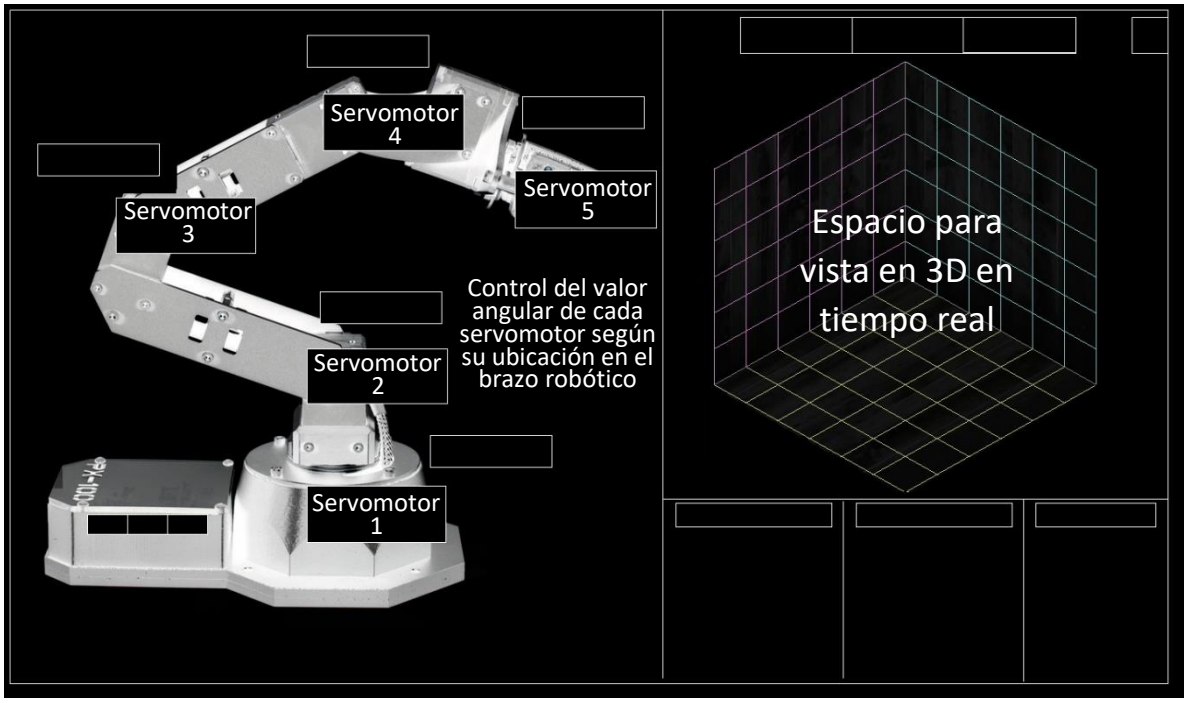
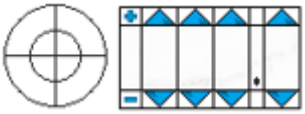
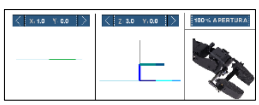
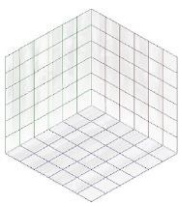


Figura 2.19. Asignar espacios por elemento en el plano.

2.2.8.4. Diseño de ilustraciones por elementos en el plano

Las ilustraciones utilizadas son de fácil interpretación a fin de mejorar la experiencia del usuario mientras usa la interfaz gráfica. En la Tabla 2.8. se detalla la función de cada elemento.

Tabla 2.8. Elementos de la Interfaz Gráfica de Usuario.

Ilustración	Función
	Permite al usuario desplazarse sobre el área de trabajo.
	Se muestra cuando existe opciones modificables debajo del espacio que ocupa.
	Permite modificar el ángulo de giro de cada servomotor de forma gráfica en un sistema de referencia en 2D o mediante el incremento/ disminución de valores numéricos.
	Una vez aplicada la cinemática directa, en estos espacios se mostrará la posición del efector final en cada uno de los ejes.
	Permite realizar el envió de datos (coordenadas angulares) al brazo robótico.
	Permite trabajar a los servomotores en modo de lectura o escritura. Desactivado: lectura y activado: escritura.
	Indica que se está trabajando la cinemática directa.
	Indica que se es trabajando la cinemática inversa.
	Indica desde donde o quien está conectado al brazo robótico.
	Muestras las vistas superior y lateral de la posición de los eslabones. Además, permite visualizar la apertura o cierre de la pinza.
	Es un plano en 3D en donde se mostrará el movimiento del brazo robótico.

2.2.8.5. Acoplar ilustraciones en el plano

Una vez establecido los espacios de trabajo y los elementos para utilizarse en la interfaz gráfica de usuario, se procede a colocarlos en su respectiva ubicación, quedando como se muestra en la Figura 2.20.

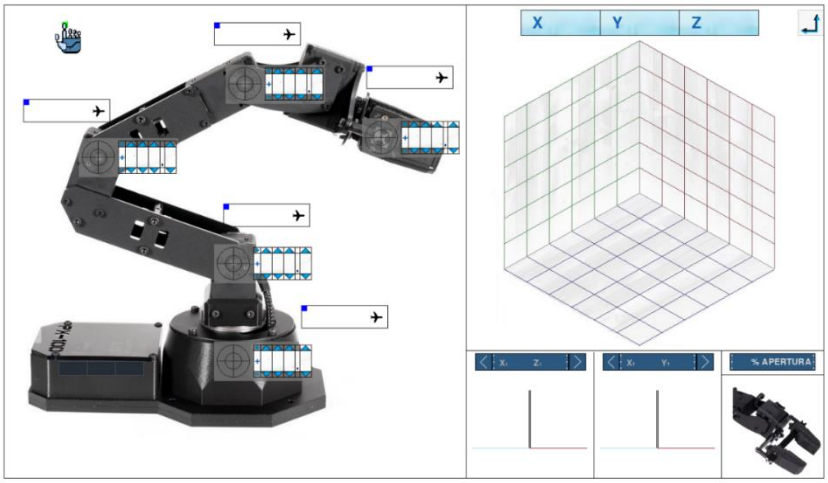


Figura 2.20. Acoplar ilustraciones en el plano.

2.2.8.6. Programar eventos

Una vez añadidos los elementos visuales y definida su funcionalidad para el control de posición del brazo robótico mediante la cinemática directa e inversa, el siguiente paso es establecer la lógica de funcionamiento de la interfaz gráfica de usuario. Para ello se hace uso de la librería Pygame propia de Python, esta ofrece herramientas que permiten desarrollar diferentes aplicaciones, como asignar funciones a los elementos, crear objetos, leer eventos.

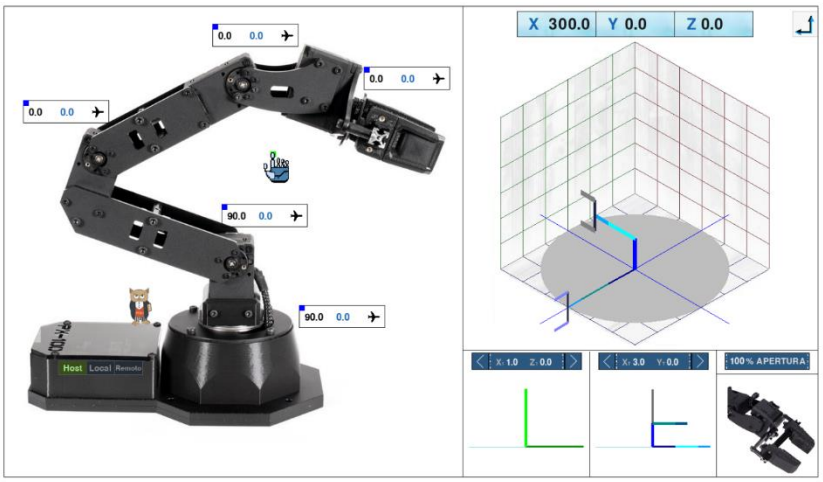


Figura 2.21. Programar eventos.

2.2.9. Registro de ejecución

La obtención de un registro de actividad sobre el actuador representado por el Brazo PincherX 100 por parte del usuario permite conocer el comportamiento o posición del efector final al modificar el ángulo únicamente de un servomotor, es decir que el cambio de posición en un eslabón afecta a toda la cadena articulada puesto que se encuentra constituida de forma serial. Esto con la finalidad de identificar a detalle el comportamiento del manipulador al modificar sus coordenadas articulares.

El registro de ejecución guarda la información sobre el tipo de usuario, la fecha que se ejecutó la acción, la posición angular inicial y actual de los cinco servomotores, y, las coordenadas cartesianas del efector final antes y después de la operación.

- El tipo de usuario se refiere a quien o desde donde se está controlando los movimientos de la plataforma robótica. Puesto que puede ser maniobrada por comunicación directa (host), local y remota.
- El formato de fecha contiene el año, mes, día, hora, minuto y segundo en que se realizó el cambio de posición del manipulador robótico.

En caso de fallos, tanto el registro del tipo de usuario como los datos de fechas, ayudarían a determinar las posibles causas de lo sucedido.

- El registro angular de los actuadores y de la posición cartesiana del extremo final del robot, permiten visualizar de forma cuantitativa el comportamiento de este.

La aplicación encargada del registro de información es Microsoft Excel debido a que permite realizar arreglos numéricos entre filas y columnas. La información se guardará de acuerdo con la estructura de la Tabla 2.9. Cada registro se almacena un total de 23 celdas, como se muestra en la Tabla 2.10.

Tabla 2.9. Formato de las variables almacenadas.

Usuario	Fecha						Posición angular inicial (°)					Posición angular actual (°)					Cinemática inicial (mm)			Cinemática actual (mm)		
	Año	Mes	Dia	Hora	Minuto	Segundo	Servomotor 1	Servomotor 2	Servomotor 3	Servomotor 4	Servomotor 5	Servomotor 1	Servomotor 2	Servomotor 3	Servomotor 4	Servomotor 5	X	Y	Z	X	Y	X

A continuación, se muestra en ejemplo guía de registro y almacenamiento de datos, e ilustración grafica en el proceso de posicionamiento del brazo robótico.

Tabla 2.10. Ejemplo de almacenamiento de datos.

Usuario	Fecha						Posición angular inicial (°)					Posición angular actual (°)					Cinemática inicial (mm)			Cinemática actual (mm)		
	Año	Mes	Dia	Hora	Minuto	Segundo	Servomotor 1	Servomotor 2	Servomotor 3	Servomotor 4	Servomotor 5	Servomotor 1	Servomotor 2	Servomotor 3	Servomotor 4	Servomotor 5	X	Y	Z	X	Y	X
Host	2023	07	31	16	40	36	90	45	145	180	45	45	45	145	180	45	-111.2	-100	-128	-189.81	-70.7	-128

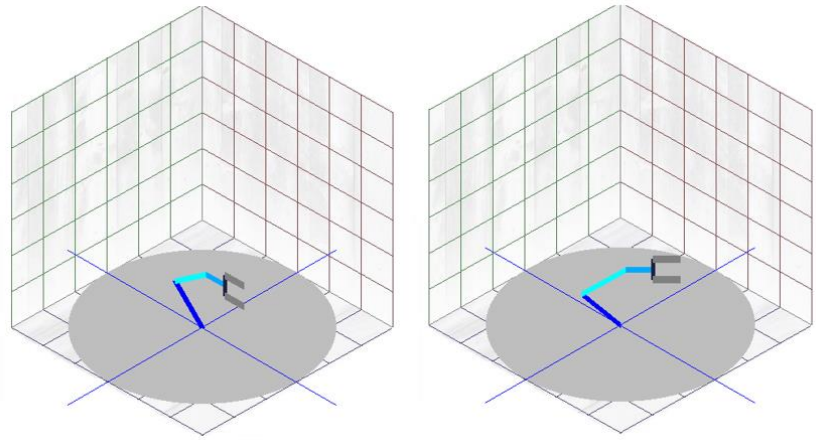


Figura 2.22. Ilustración gráfica registro de posición.

2.2.10. Configuración Red local

Para configurar un Red Local en Windows se necesita que los dispositivos a utilizarse estén en red, es decir que se comuniquen entre ellos ya sea mediante ethernet o por Wifi. En la investigación se ha utilizado dos computadores con sistema operativo Windows 10 y Windows 11 para que uno sea el host y el otro cumpla la función de servidor, respectivamente. Por tanto, para compartir y acceder a archivos entre los dispositivos

conectados a una red LAN se debe realizar la siguiente configuración en Windows en todos los dispositivos que vayan a interconectarse entre sí .

Paso 1: Asignar un mismo nombre de grupo a todos los dispositivos.

- Presionar la tecla Windows + R, y en la ventana que se abre escribir el comando **sisdm.cpl.**

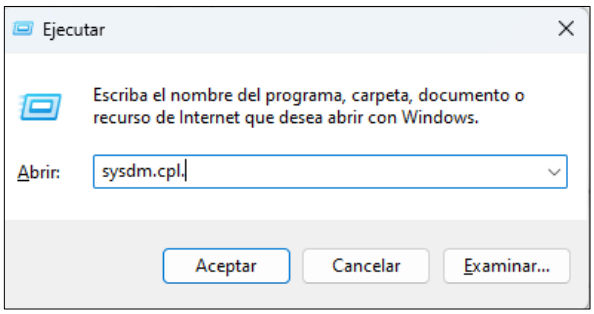


Figura 2.23. Acceso a la configuración de nombre del grupo.

- En la ventana propiedades del sistema, hacer clic en la opción → Nombre del Equipo → Cambiar.

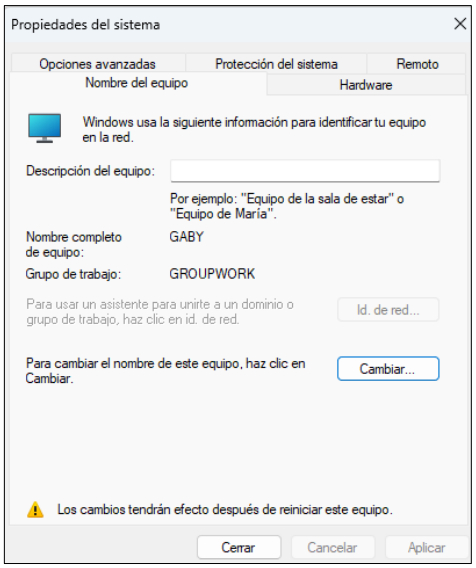


Figura 2.24. Acceso a propiedades del sistema.

- Cambiar el nombre del grupo de trabajo a todas las maquinas que van a estar interconectadas entre sí, estas deben tener exactamente la misma identificación.

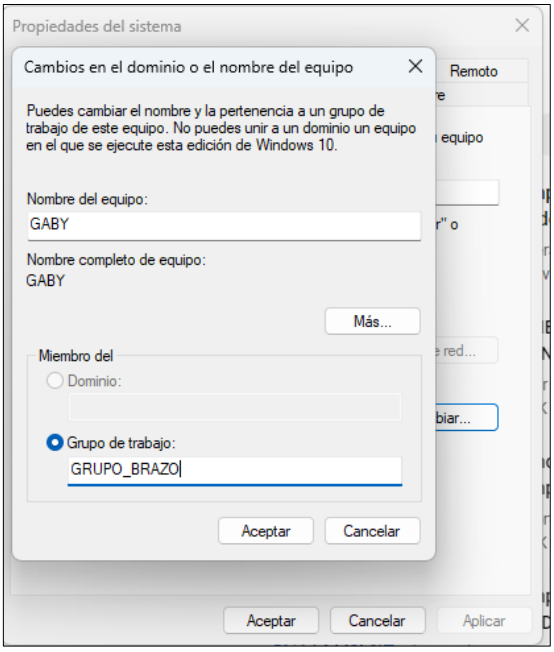


Figura 2.25. Cambio de nombre del grupo de trabajo.

Seguidamente hacer clic en →Aceptar, y reiniciar el computador para que los cambios se actualicen y se guarden correctamente.

Paso 2: Configurar el acceso a la red.

- En la parte inferior derecha, acceder al icono de Wifi en donde se visualizan todas las redes disponibles. Dar clic derecho sobre la red conectada, y seleccionar la opción →Propiedades.

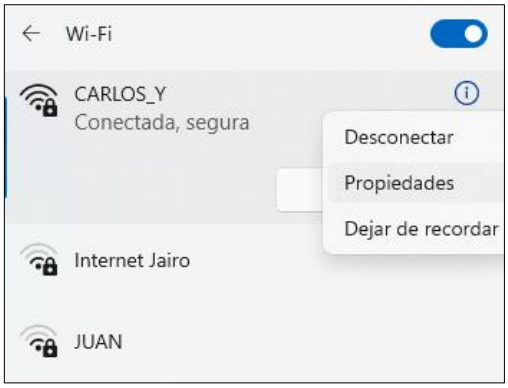


Figura 2.26. Acceso a la configuración de la red.

- Verificar que el tipo de red sea →Privada. Esta configuración debe hacerse en todos los equipos.

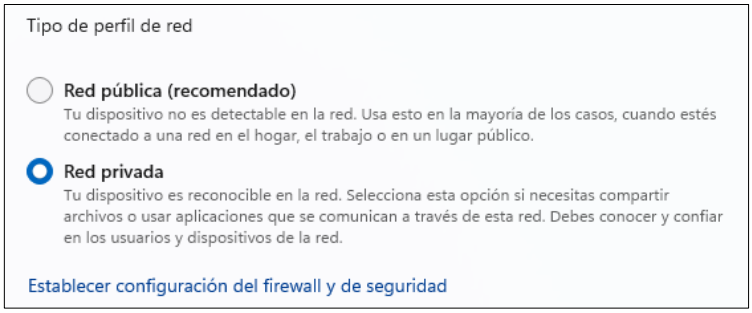


Figura 2.27. Asignar red privada.

Paso 3: Habilitar la detección de redes y el uso compartido de archivos.

- Acceder al panel de control y dar clic en la opción → Centro de redes y recursos compartidos.

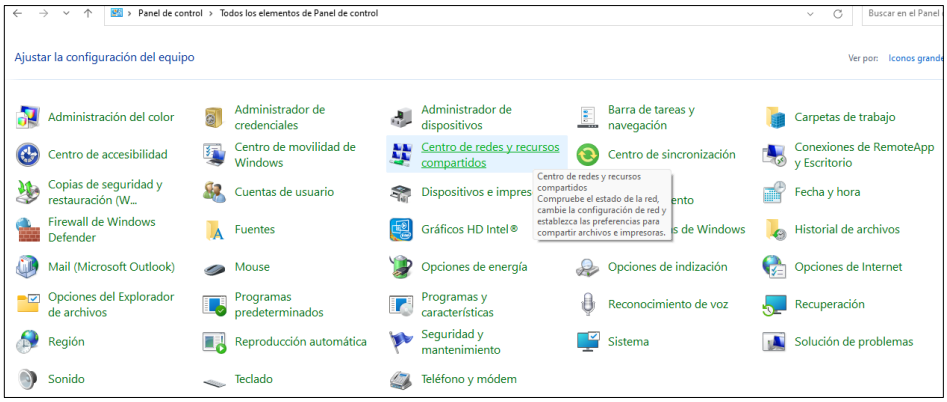


Figura 2.28. Opción de centro de redes y recursos compartidos.

En caso de que no se muestre todos los iconos, hacer clic en → Categoría →Iconos grandes.

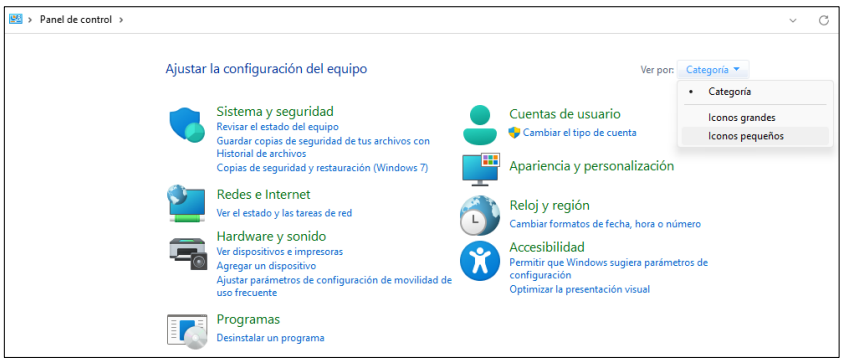


Figura 2.29. Acceso al panel de control.

- En la ventana de Centro de redes y recursos compartidos, hacer clic en la opción → Cambiar configuración de uso compartido avanzado.

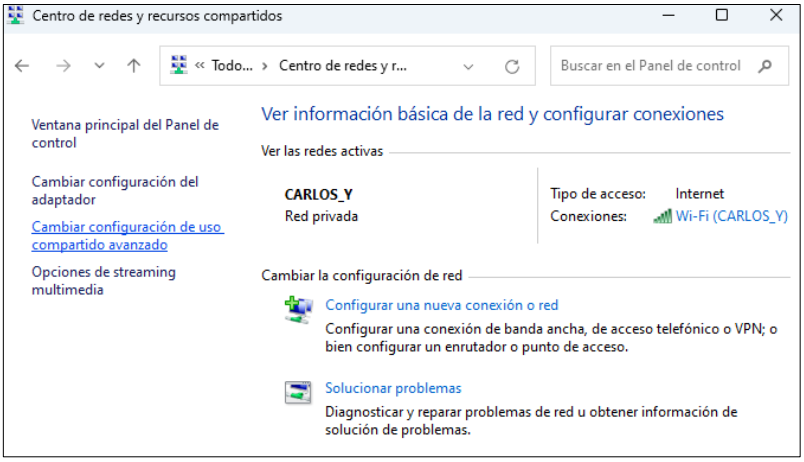


Figura 2.30. Configurar el uso compartido.

- En la ventana de Configuración avanzada de uso compartido se puede visualizar tres tipos opciones: red privada, red pública y todas las redes.

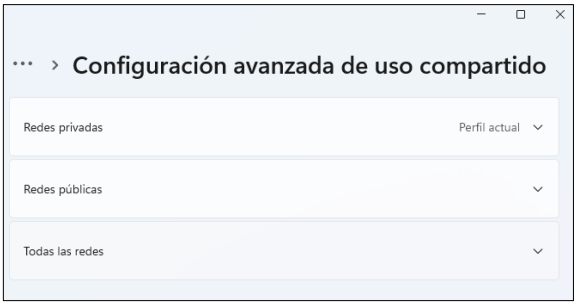


Figura 2.31. Redes existentes en el computador.

En las redes pública y privada se debe activar detección de redes y el uso compartido de archivos e impresoras. Por su parte en la opción de todas las redes se debe activar el uso compartido de carpetas públicas, y desactivar el uso compartido protegido con contraseña.

Figura 2.32. Habilitar opciones en la red privada.

Figura 2.33. Habilitar opciones en la red pública.

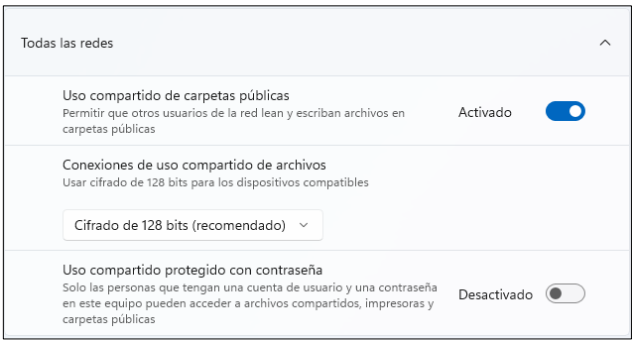


Figura 2.34. Configuración en la opción todas las redes.

- Finalmente, al hacer clic en la opción de →Red, ya se mostrará que el equipo ya está en red, es decir ya puede compartir y acceder a los archivos de otros computadores que pertenezcan al mismo grupo de trabajo [35].

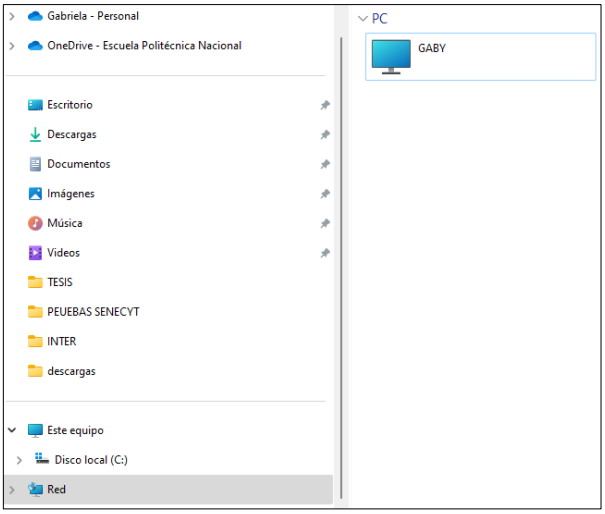


Figura 2.35. Visualización de equipos en red.

2.2.11. Configuración Red remota

Hamachi es un software desarrollado por LogMeIn, permite crear escritorios remotos que se comunican mediante una red VPN, simulando una red LAN a partir de una res WAN.

A continuación, se muestra la imitación y configuración del software Hamachi.

Paso 1: Descarga e instalación de Hamachi by LogMeIn.

- En el navegador escribir **vpn.net**, para descargar el software. Damos clic en →Download now.

Figura 2.36. Página web oficial de Hamachi by LogMeIn

- Una vez descargado el programa, se procede a la instalación de este. Para esto se debe abrir el archivo descargado y dar clic en →Ejecutar.

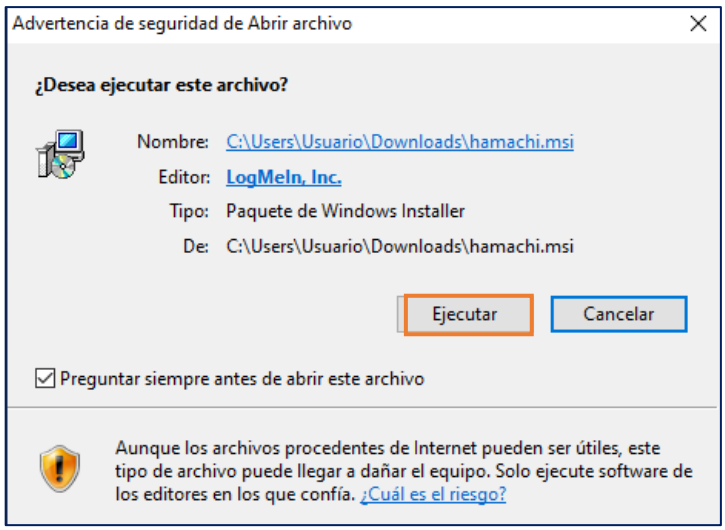


Figura 2.37. Ejecución del software Hamachi by LogMeIn.

- El siguiente paso es escoger el idioma del programa y damos clic en el botón → Next.

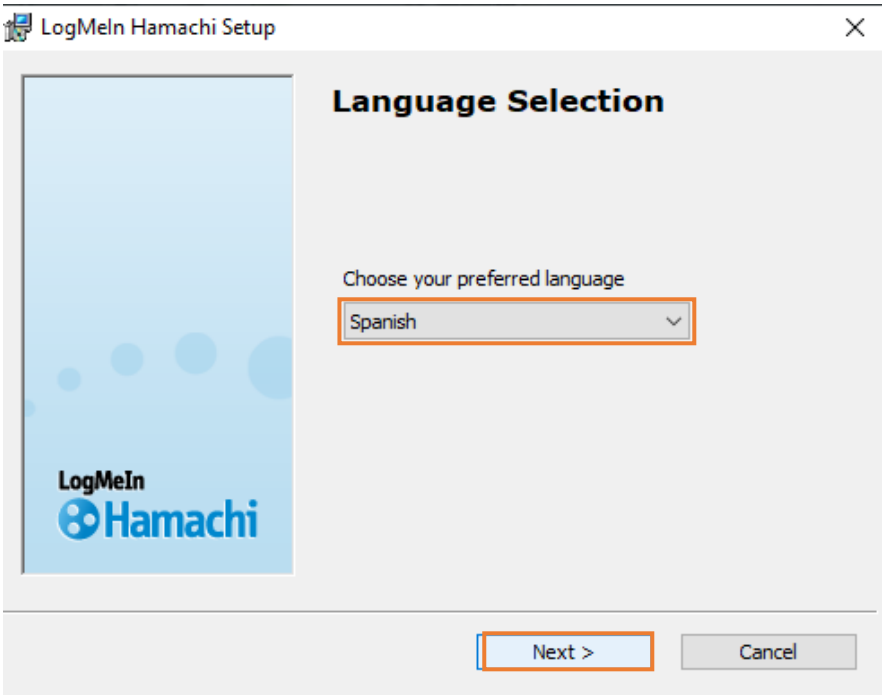


Figura 2.38. Selección de idioma.

- Aceptar los términos y condiciones, hacer clin en →Siguiente. Seguidamente se abrirá una ventana para escoger la ubicación del programa y crear su acceso directo en el escritorio.

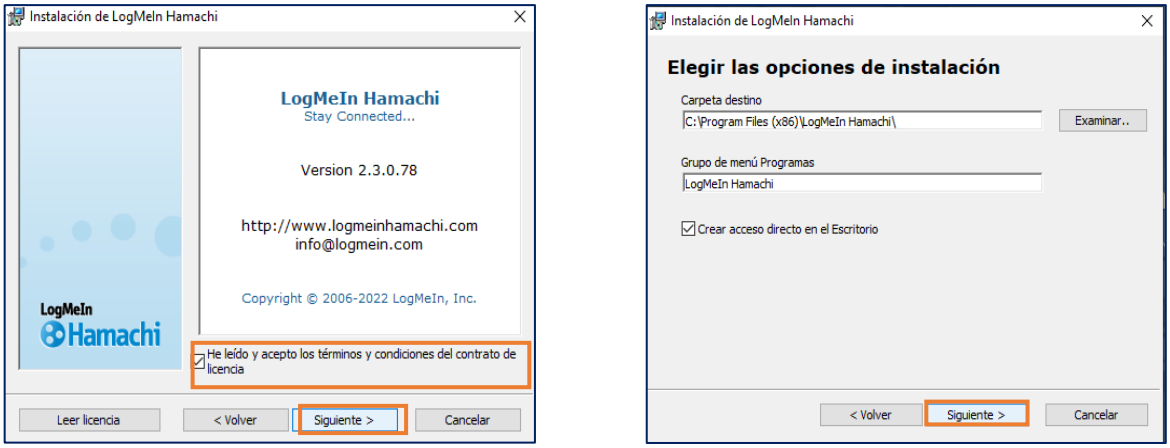


Figura 2.39. Opciones de instalación del software.

- A continuación, se mostrará la ventana de la Figura 2.40., dar clic en el botón →Instalar, y esperar un momento; una vez finalizado dar clic en el botón →Terminar.

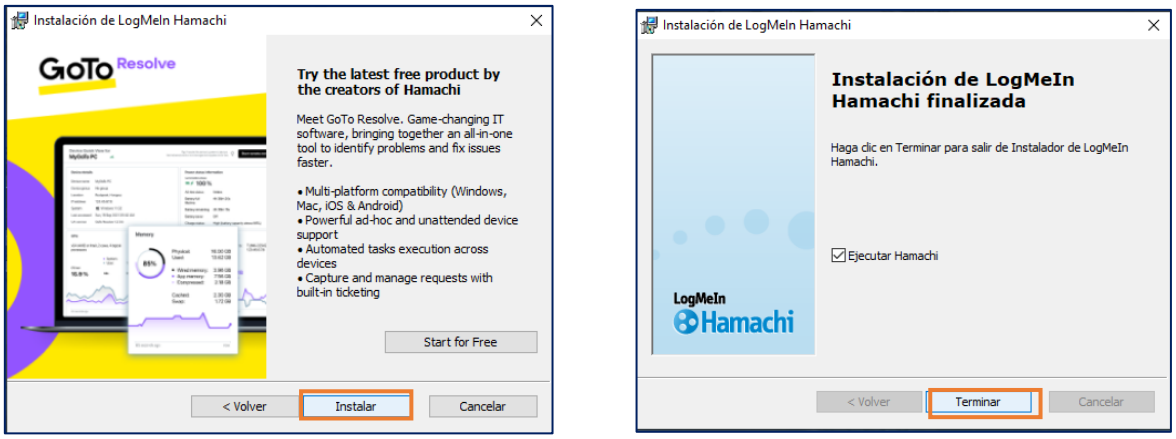


Figura 2.40. Instalación de Hamachi.

Paso 2: Creación de una nueva red en Hamachi

- Una vez que se abra el programa, se deberá crear una red. Para esto se debe tener o crear una cuenta de usuario. Para registrarse solamente se debe colar un correo electrónico y una contraseña, y seguidamente hacer clic en → Crear cuenta.

Figura 2.41. Crear cuenta.

- Ya creada la cuenta, se puede acceder al software. Para activarlo se debe hacer clic en el →Icono de encender.

Figura 2.42. Habilitar conexión.

- Seguidamente se debe crear una red, para esto se debe hacer clic en la opción →Red → Crear una nueva red.

Un dato importante es que en la parte superior podemos encontrar la dirección IP con la que se identificara el computador al usar el software Hamachi y el número **2620** que indica el número de puerto.

Figura 2.43. Crear una nueva red.

- En la nueva ventana se asignará un ID de red y una contraseña**,** estos datos permitirán a los otros usuarios conectarse a la red creada.

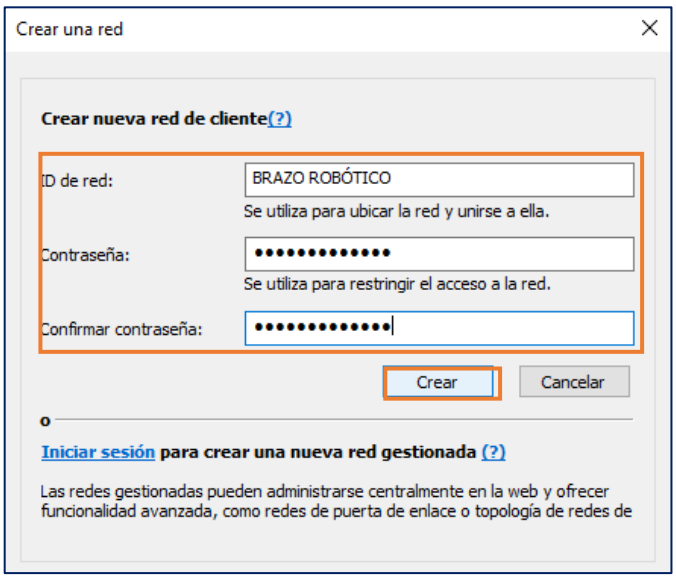


Figura 2.44. Asignar credenciales a la nueva red.

- Se podrá ver la red creada, pero de momento solo habrá un dispositivo conectado. Todos los usuarios que vayan a interconectarse por medio de la red VPN creada por Hamachi, deberán realizar la descarga e instalación del programa en mención.

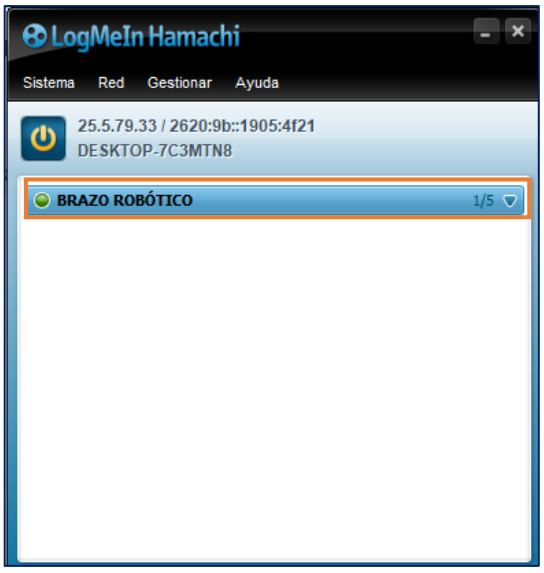


Figura 2.45. Visualización de la red creada.

Paso 3: Unirse a una red existente

- Para acceder a una red creada en Hamachi se debe conocer el ID de la red y la contraseña. Ahora se dará clic en la opción →Red → Unirse a una red existente.

Figura 2.46. Unirse a una red existente.

- Colocar el ID de la red y la contraseña de la red a la cual se requiera hacer la conexión.

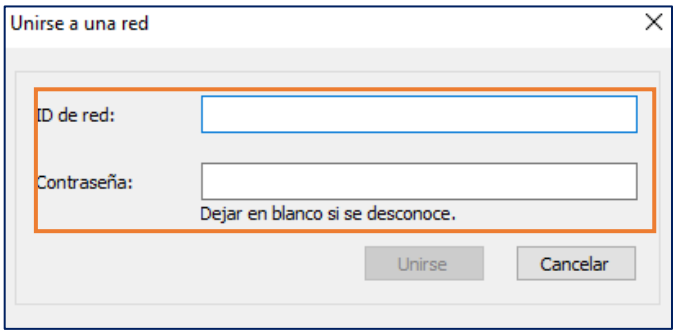


Figura 2.47. Acceder a una red existente a través de credenciales.

Cabe recalcar que el número de integrantes es limitado (5 dispositivos) debido a que se está utilizando la versión gratuita [36]. Una vez conectado los usuarios a la red, se podrá visualizar el nombre del equipo conectado.

En la Figura 2.48. se puede observar cómo dos equipos ya están conectados a la misma red denominada Brazo Robótico.

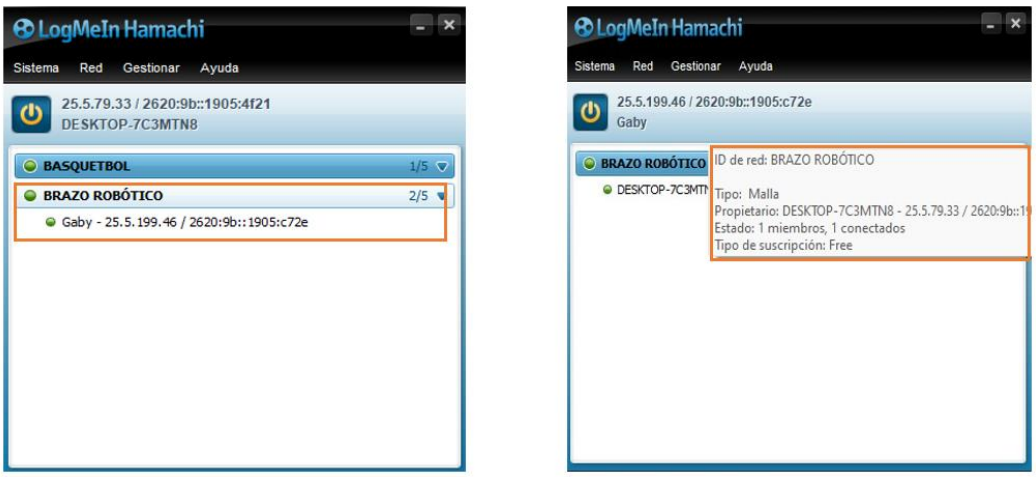


Figura 2. 48. Conexión de dos computadores a una misma red a través de Hamachi.

2.3. Implementación del sistema.

2.3.1. Red de comunicación

En la Figura 2.49. se observa la consolidación general del sistema relacionado a la red de comunicación y como esta se comunica con el brazo robótico.

El sistema considera tres tipos de usuario:

- El anfitrión o host con la característica de ser el único independiente y tener comunicación directa con el Brazo PincherX 100 por medio del convertidor USB DYNAMIXEL U2D2.
- Acceso Local se lo hace entre dispositivos que estén conectados a una misma red ya sea mediante el uso de cable Ethernet o Wifi. Los computadores utilizados cuentan con el sistema operativo Windows 10 y 11, por lo que para que tengan acceso a los datos uno del otro, basta con configurar la red entre ordenadores, tal como se mostró en el apartado 2.2.10.
- Acceso Remoto se realiza a través del software Hamachi by LogMeIn, este debe estar instalado en los dos computadores que vayan a conectarse y deben ser configurados como se indicó en el numeral 2.2.11. De esta forma se consigue el control del brazo de forma remota y segura.

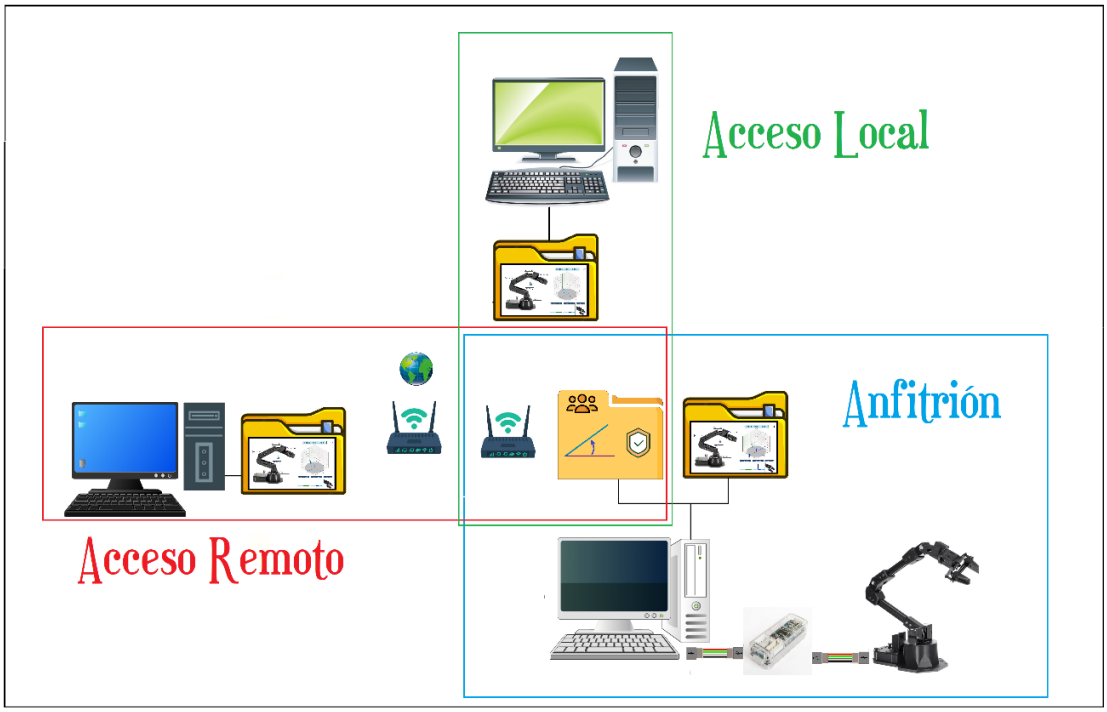


Figura 2.49. Red de Comunicación.

Cada uno de los tipos de usuario posee su propia aplicación con su identificador correspondiente y se comunica al sistema anfitrión mediante un sistema de red de comunicación. Es decir, se diseñaron e implementaron tres interfaces de comunicación y de usuario para la comunicación directa, local y remora, respectivamente.

Los archivos que comparte el anfitrión corresponden al posicionamiento actual del brazo y el estado de conexión de cada usuario, la información sobre el estado de conexión es presentada en cada una de las aplicaciones mediante un sistema de colores que se muestra en la Tabla 2.11.

Tabla 2.11. Estado de conexión.

N°	Estado	Color
0	Desconectado	Negro
1	Conectado	Azul
3	Ejecución	Verde
4	Host desconectado	Rojo

En el siguiente diagrama de flujo se observa el proceso de solicitud de acceso que realiza el usuario para poder controlar y posicionar los actuadores.

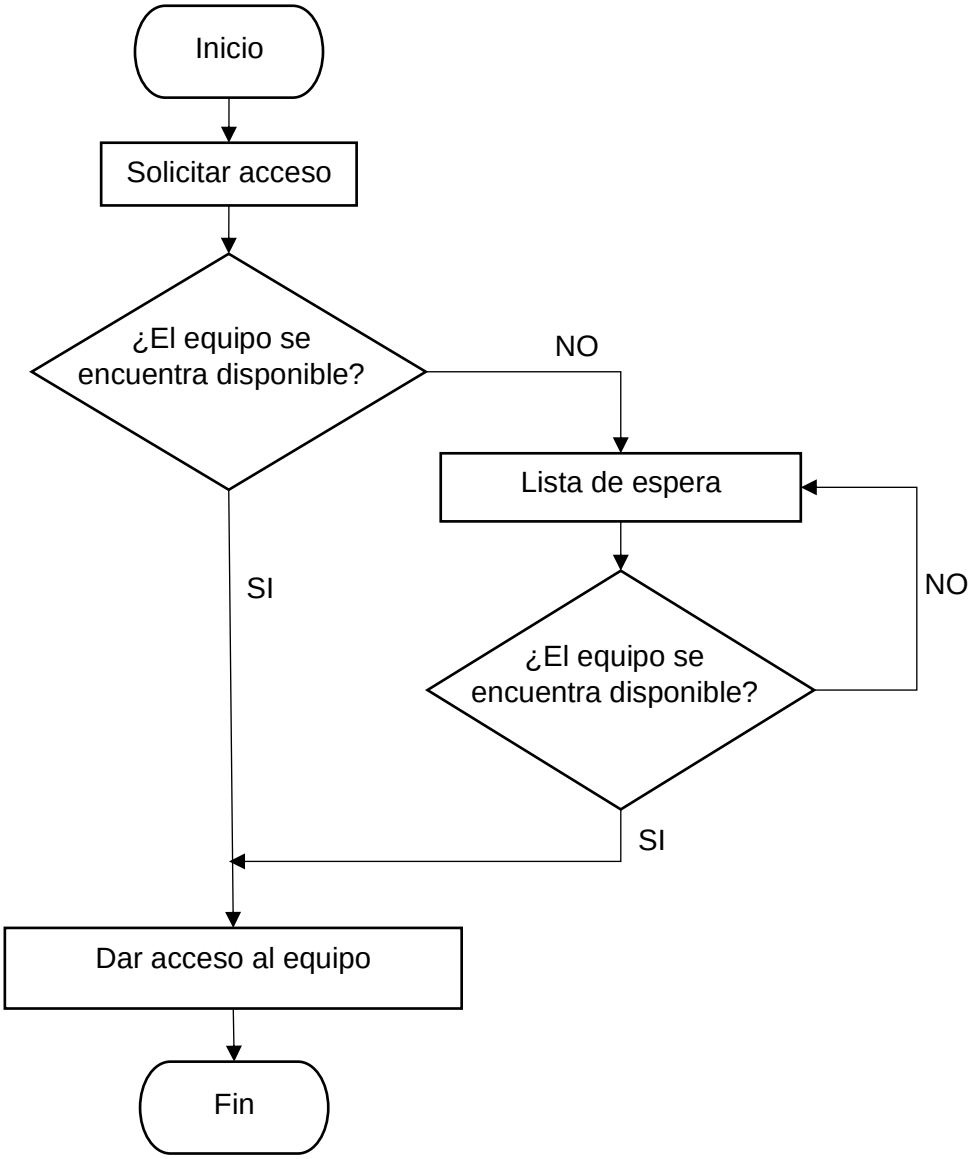


Figura 2.50. Solicitud Acceso.

En el diagrama de flujo se indica como se asignan los turnos y el funcionamiento a detalle de las listas de espera. La aplicación permite el acceso al primer usuario que solicita el acceso sin considerar la jerarquía y maneja una línea de espera representando por el siguiente diagrama.

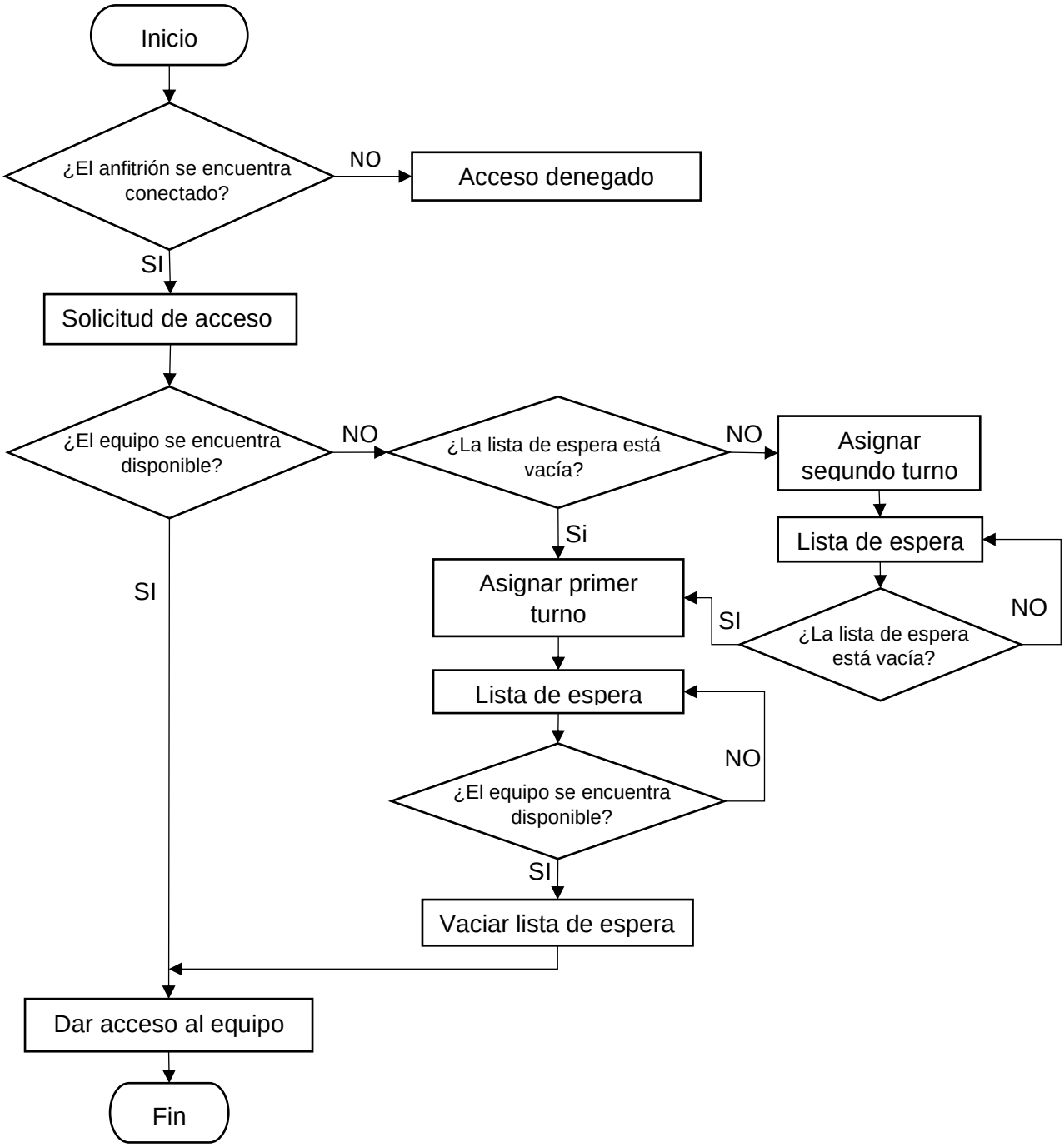


Figura 2.51. Solicitud Acceso.

Por su parte para el posicionamiento del brazo el anfitrión o PC principal siempre deberá estar activa, puesto que de lo contrario no podrá ser controlado de forma local y remota. En el siguiente diagrama de flujo se observa el proceso de posicionamiento.

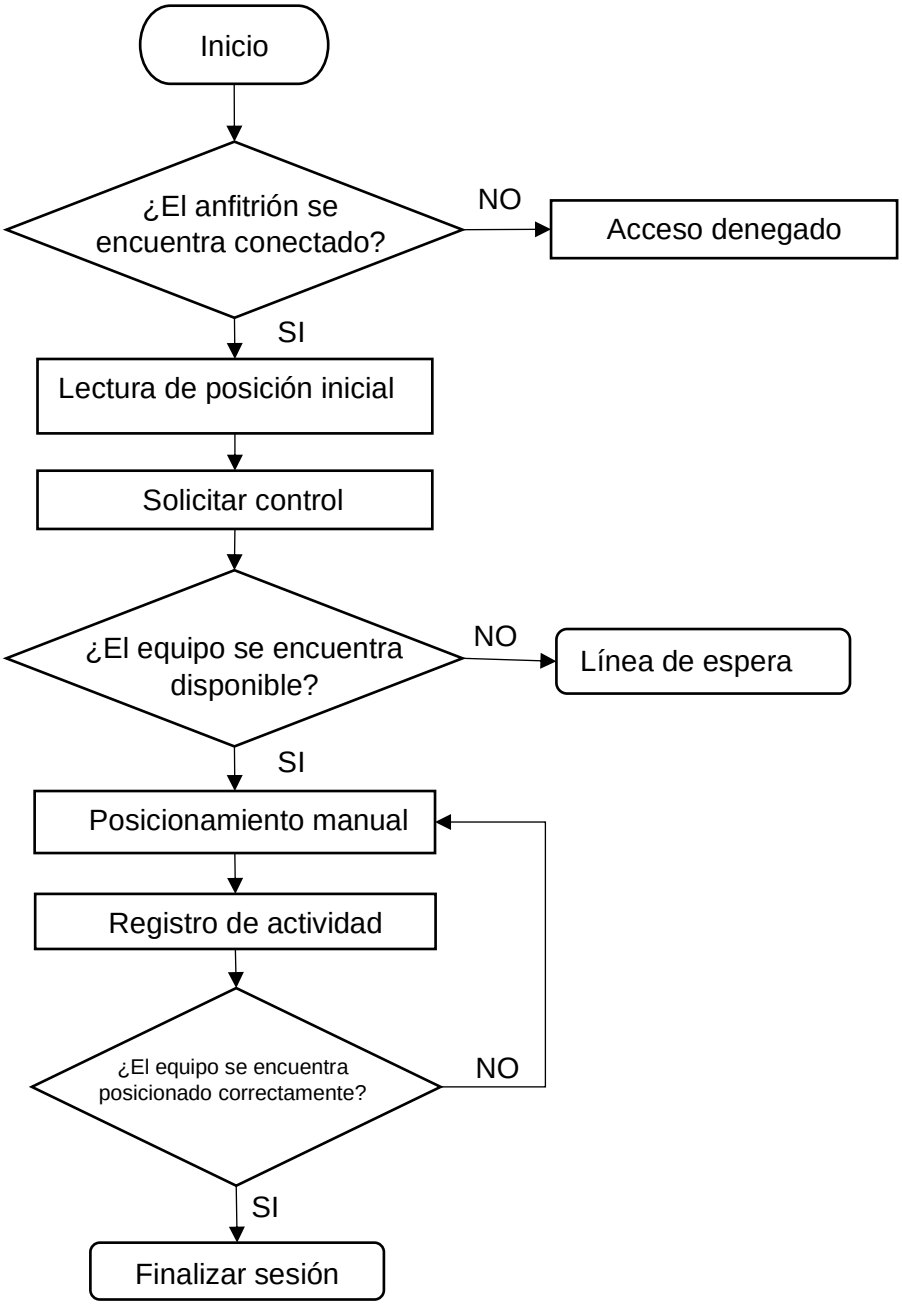


Figura 2.52. Posicionamiento del brazo robótico.

Como medida de seguridad para evitar choques o golpes del brazo robótico previo a ser utilizado, se ha establecido que únicamente pueda desconectarse cuando este en su posición de descanso, es decir debe estar en dicha posición antes del cierre de sesión.

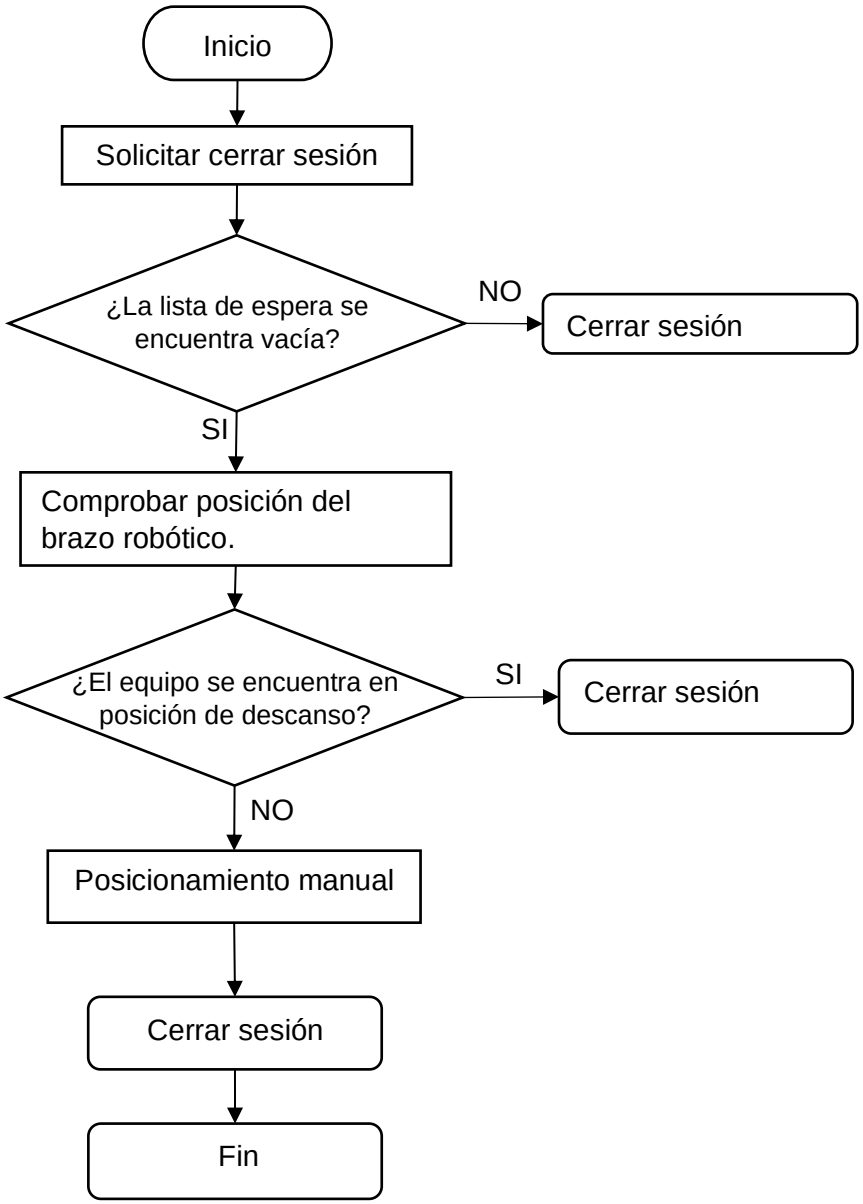


Figura 2.53. Posición de descanso previo a ser desconectado.

Por su parte, en el caso de la cinemática inversa, el usuario puede ingresar posiciones a las cuales el brazo robótico no puede llegar, o a su vez, el algoritmo puede dar como resultado ángulos fuera de los rangos permitidos, por lo que en la interfaz gráfica de usuario se indicará con una equis “X” que dicha posición digitada no puede ser posicionada.

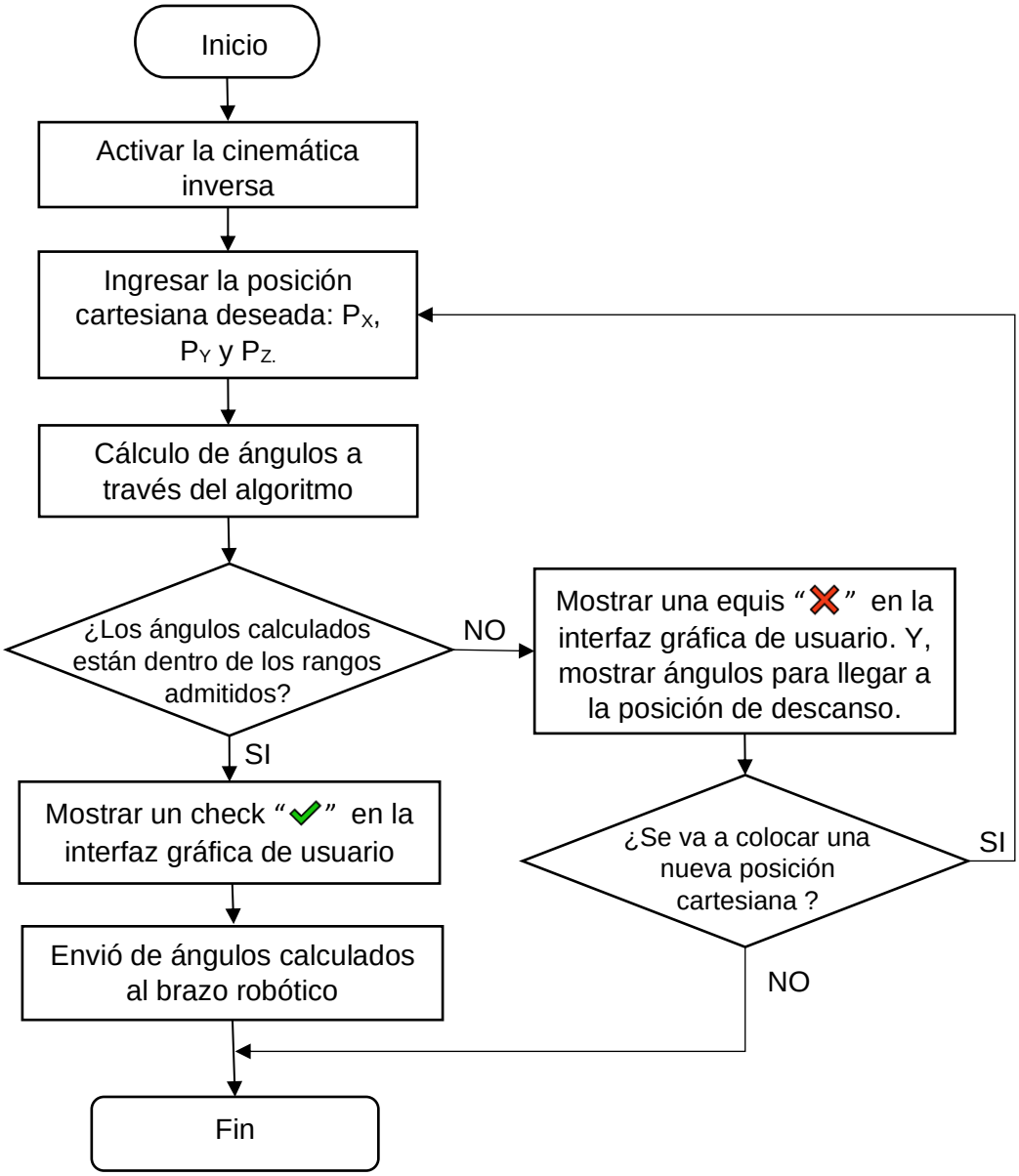


Figura 2.54. Ejecución de la Cinemática Inversa.

2.3.2. Uso de ficheros en la interfaz de comunicación

Para la implementación exitosa de la interfaz de comunicación ha sido importante el uso de ficheros. En donde el sistema de comunicación se encuentra a cargo del dispositivo denominado anfitrión, este posee archivos de texto permanentes que permiten la conexión distante y la configuración de parámetros relacionados al brazo robótico. Por ejemplo, existe un archivo de texto que tiene la función de almacenar un contador que posteriormente registra el posicionamiento del brazo en una fila de un archivo en Excel.

Un intercambio de información bidireccional y en tiempo real conlleva una constante comprobación en los datos que puede ocasionar la saturación en la red, interrumpir la fluidez del programa y perdidas de información dado que los parámetros de ejecución de la aplicación pueden cambiar.

Inicialmente en lo referente a los tiempos de respuesta entre el host - remoto, se hizo una medición del tiempo de forma directa, evaluando el intervalo entre la solicitud y la respuesta, sin necesidad de almacenar datos en archivos intermedios. Los resultados de esta demostración se reflejan en la Tabla 2.12, donde se puede observar el tiempo de respuesta en segundos obtenido durante estas pruebas. Sin embargo, debido a los tiempos de respuesta relativamente altos en la comunicación directa, se tomó la decisión de cambiar la estrategia de interacción. En lugar de ejecutar el programa de forma directa y remota, se optó por trabajar a través del uso de ficheros o archivos. La tabla resultante que ilustra esta nueva aproximación se presenta en la sección de resultados.

En base a los datos recopilados, se obtuvieron valiosos insights que contribuyeron a la eficiencia y optimización del proceso de comunicación en tiempo real, influyendo directamente en el control y desempeño del brazo robótico en esta configuración específica.

Tabla 2.12. Comunicación en tiempo real entre el Host – Remoto en segundos.

Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor
1	2,32714	6	10,25088	11	0,89303	16	0,85635
2	1,43602	7	10,82115	12	1,5255	17	0,66327
3	6,56912	8	4,57126	13	2,37236	18	0,4954
4	6,58711	9	2,8672	14	0,69551	19	0,69816
5	3,02995	10	3,55635	15	0,76919	20	0,85578

En el siguiente diagrama de flujo se muestra la aplicación de ficheros en el envío de una orden en el posicionamiento en el brazo robótico.

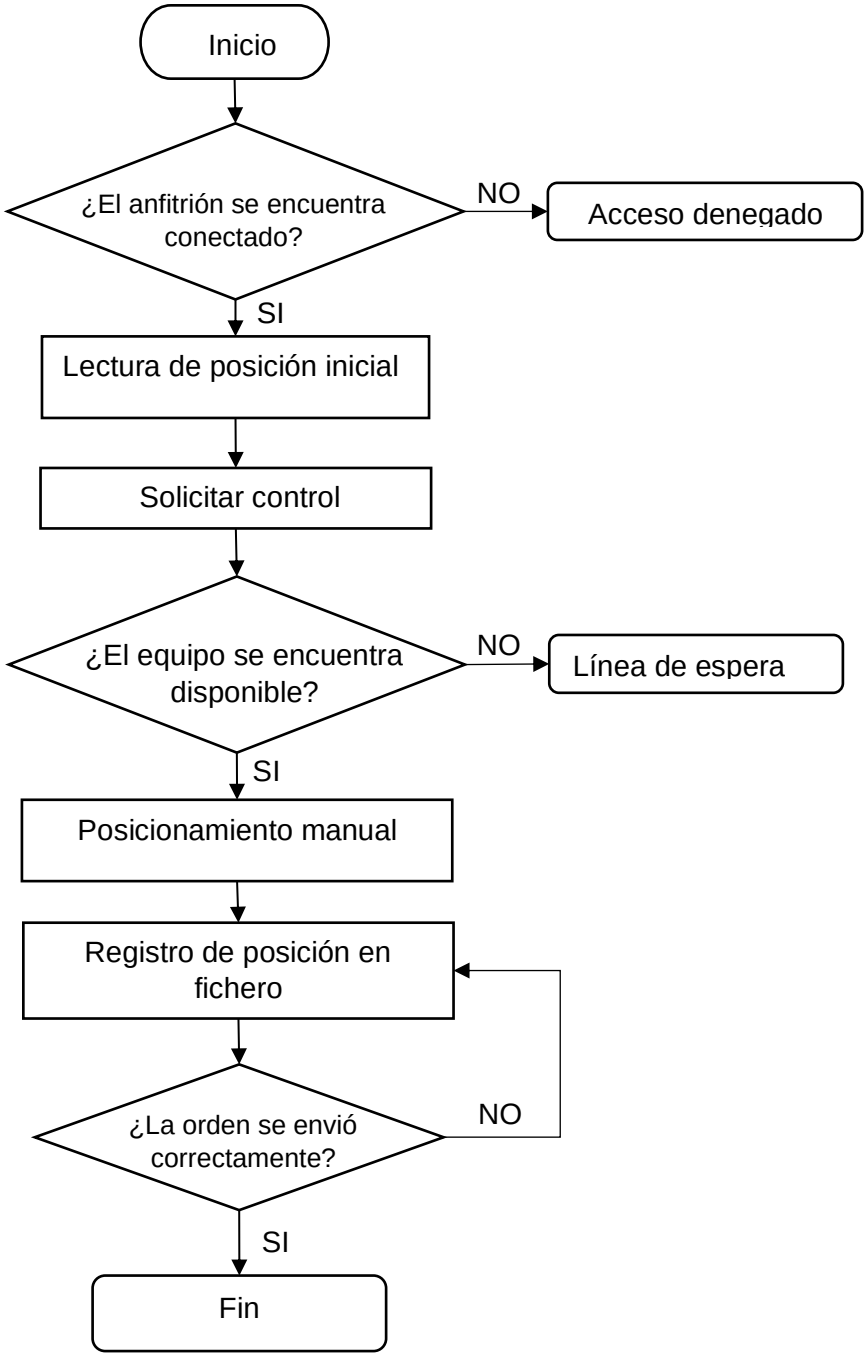


Figura 2.55. Uso de ficheros para el envío de órdenes.

De la misma forma se presenta el diagrama de flujo para la aplicación de ficheros en la recepción de una orden en el posicionamiento en el brazo robótico.

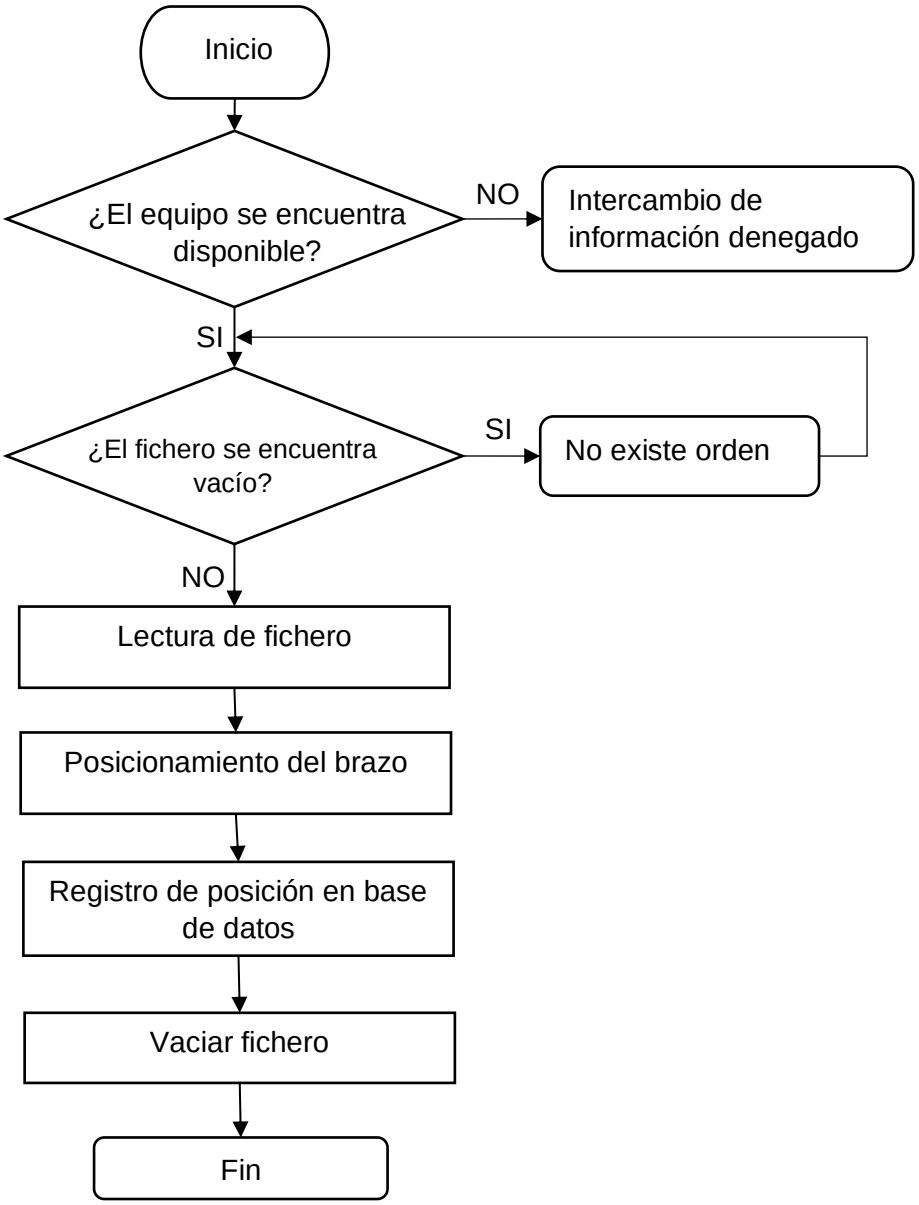


Figura 2.56. Uso de ficheros para la recepción de órdenes.

3. RESULTADOS

Para los resultados se han realizado varias pruebas tanto en los resultados obtenidos por los diferentes algoritmos, así como también de los tiempos de respuesta derivados de la puesta en marcha y de los tipos de conexiones aplicados, dichas pruebas se detallan a continuación.

3.1. Pruebas de métodos cinemáticos

3.1.1. Comparación de métodos aplicados en la cinemática directa.

Para determinar la validez de las fórmulas que se rigen a las características físicas del brazo robótico PincherX 100, se ha aplicado dos métodos, el geométrico y el de Denavit Hartenberg, esto con el fin de comparar los resultados y poder comprobar que el autómata se comporta según lo calculado por dichos métodos.

Se han realizado varias pruebas, a continuación, se muestra algunas de ellas en donde se puede apreciar los resultados de los algoritmos aplicados.

Tabla 3.1. Comparación de resultado entre métodos aplicados en la cinemática directa.

Prueba 1			
Posiciones angulares para cada articulación:			
$\theta_0 = 0°$	$\theta_1 = 90°$	$\theta_2 = 90°$	$\theta_3 = 0°$
Método Geométrico		Método Denavit Hartenberg	
Método Geométrico Posición final del efector Coordenada X: 35.0000000000000 Coordenada Y: 0 Coordenada Z: 418.600000000000		Método Denavit Hartenberg Posición final del efector Coordenada X: 35.0000000000000 Coordenada Y: 0 Coordenada Z: 418.600000000000	
Prueba 2			
Posiciones angulares para cada articulación:			
$\theta_0 = 180°$	$\theta_1 = 90°$	$\theta_2 = 90°$	$\theta_3 = 0°$
Método Geométrico		Método Denavit Hartenberg	
Método Geométrico Posición final del efector Coordenada X: -35.0000000000000 Coordenada Y: 4.28626379701574E-15 Coordenada Z: 418.600000000000		Método Denavit Hartenberg Posición final del efector Coordenada X: -35.0000000000000 Coordenada Y: 4.28626379701574E-15 Coordenada Z: 418.600000000000	
Prueba 3			
Posiciones angulares para cada articulación:			
$\theta_0 = 0°$	$\theta_1 = 90°$	$\theta_2 = 0°$	$\theta_3 = 0°$
Método Geométrico		Método Denavit Hartenberg	
Método Geométrico Posición final del efector Coordenada X: 264.150000000000 Coordenada Y: 0 Coordenada Z: 189.450000000000		Método Denavit Hartenberg Posición final del efector Coordenada X: 264.150000000000 Coordenada Y: 0 Coordenada Z: 189.450000000000	

<table>
<tr><th colspan="4">Prueba 4</th></tr>
<tr><td colspan="4">Posiciones angulares para cada articulación:</td></tr>
<tr><td>$\theta_0 = -75°$</td><td>$\theta_1 = 90°$</td><td>$\theta_2 = 45°$</td><td>$\theta_3 = 0°$</td></tr>
<tr><td colspan="2">Método Geométrico</td><td colspan="2">Método Denavit Hartenberg</td></tr>
<tr><td colspan="2">Método Geométrico
Posición final del efector
Coordenada X: 50.9960272171903
Coordenada Y: -190.319764558719
Coordenada Z: 351.483518908897</td><td colspan="2">Método Denavit Hartenberg
Posición final del efector
Coordenada X: 50.9960272171903
Coordenada Y: -190.319764558719
Coordenada Z: 351.483518908897</td></tr>
<tr><th colspan="4">Prueba 5</th></tr>
<tr><td colspan="4">Posiciones angulares para cada articulación:</td></tr>
<tr><td>$\theta_0 = -160°$</td><td>$\theta_1 = 80°$</td><td>$\theta_2 = 30°$</td><td>$\theta_3 = -15°$</td></tr>
<tr><td colspan="2">Método Geométrico</td><td colspan="2">Método Denavit Hartenberg</td></tr>
<tr><td colspan="2">Método Geométrico
Posición final del efector
Coordenada X: -257.908879097052
Coordenada Y: -93.8711551442877
Coordenada Z: 227.311267591305</td><td colspan="2">Método Denavit Hartenberg
Posición final del efector
Coordenada X: -257.908879097052
Coordenada Y: -93.8711551442877
Coordenada Z: 227.311267591305</td></tr>
<tr><th colspan="4">Prueba 6</th></tr>
<tr><td colspan="4">Posiciones angulares para cada articulación:</td></tr>
<tr><td>$\theta_0 = -110°$</td><td>$\theta_1 = 120°$</td><td>$\theta_2 = -18°$</td><td>$\theta_3 = 80°$</td></tr>
<tr><td colspan="2">Método Geométrico</td><td colspan="2">Método Denavit Hartenberg</td></tr>
<tr><td colspan="2">Método Geométrico
Posición final del efector
Coordenada X: -25.1789686076817
Coordenada Y: -69.1786476947623
Coordenada Z: 343.415034769736</td><td colspan="2">Método Denavit Hartenberg
Posición final del efector
Coordenada X: -25.1789686076817
Coordenada Y: -69.1786476947622
Coordenada Z: 343.415034769736</td></tr>
<tr><th colspan="4">Prueba 7</th></tr>
<tr><td colspan="4">Posiciones angulares para cada articulación:</td></tr>
<tr><td>$\theta_0 = 63°$</td><td>$\theta_1 = 90°$</td><td>$\theta_2 = 25°$</td><td>$\theta_3 = -52°$</td></tr>
<tr><td colspan="2">Método Geométrico</td><td colspan="2">Método Denavit Hartenberg</td></tr>
<tr><td colspan="2">Método Geométrico
Posición final del efector
Coordenada X: 109.277452418108
Coordenada Y: 214.469076130617
Coordenada Z: 173.078953132707</td><td colspan="2">Método Denavit Hartenberg
Posición final del efector
Coordenada X: 109.277452418108
Coordenada Y: 214.469076130617
Coordenada Z: 173.078953132707</td></tr>
<tr><th colspan="4">Prueba 8</th></tr>
<tr><td colspan="4">Posiciones angulares para cada articulación:</td></tr>
<tr><td>$\theta_0 = 33°$</td><td>$\theta_1 = 110°$</td><td>$\theta_2 = -81°$</td><td>$\theta_3 = 59°$</td></tr>
<tr><td colspan="2">Método Geométrico</td><td colspan="2">Método Denavit Hartenberg</td></tr>
<tr><td colspan="2">Método Geométrico
Posición final del efector
Coordenada X: 147.806893912272
Coordenada Y: 95.9869192335682
Coordenada Z: 103.420726381922</td><td colspan="2">Método Denavit Hartenberg
Posición final del efector
Coordenada X: 147.806893912272
Coordenada Y: 95.9869192335682
Coordenada Z: 103.420726381922</td></tr>
</table>

Una vez realizadas varias pruebas entre ambos métodos, se ha comprobado que las ecuaciones están planteadas correctamente y se adaptan al comportamiento del brazo robótico. Se ha realizado esta comparación puesto que el método geométrico es sencillo de comprender y plantear, pero se consiguen cuatro ecuaciones como se detalla en la sección 2.2.6.1., en donde una de ellas se obtiene a partir de la proyección de la resultante

en el plano XY. Por ellos, dichas ecuaciones obtenidas por el método geométrico no son útiles para ser aplicadas en la cinemática inversa, debido a que se requeriría de la posición en P_X, P_Y, P_Z y adicionalmente el valor de la proyección de la resultante.

Debido a la fácil comprensión del método geométrico, este fue utilizado netamente para realizar una comparación con los resultados obtenidos a través del método de Denavit Hartenberg. Puesto que, este último presenta cierta complejidad debido a sus reglas para establecer los sistemas de referencia y su algoritmo que permite obtener las matrices de transformación, sin embargo, las ecuaciones de posición que nos arroja este método, si son útiles para ser aplicadas en la cinemática inversa, debido a que estas se basan únicamente en la posición cartesiana del efector final en el espacio, P_X, P_X y P_Z.

3.1.2. Comparación de resultados entre la cinemática directa e inversa

La cinemática inversa fue obtenida a partir de las ecuaciones resultantes de la aplicación de método de Denavit Hartenberg. Mediante el uso de herramientas propias de Python y las ecuaciones antes mencionadas, se aplicó un algoritmo que calcula los valores angulares a partir de las posiciones en P_X, P_Y Y P_Z. Se ha realizado varias pruebas entre la cinemática directa e inversa, estas consisten en ingresar los valores angulares para cada servomotor y analizar los resultados obtenidos para el efector final, seguidamente dichos resultados serán ingresados en la cinemática inversa, y con los valores angulares que esta nos arroje se calculará nuevamente la cinemática directa, esto con el objetivo de comprobar que los ángulos calculados por la cinemática inversa en realidad posicionan al gripper en la posición obtenida inicialmente por la cinemática directa.

En la Tabla 3.2., se muestran los resultados obtenidos, en donde se puede observar que las respuestas entre la cinemática directa e inversa presentan una variación baja o casi nula, puesto que existe una diferencia porcentual de 0.16%, 0.19% y 0.10% en los ejes X, Y y Z, respectivamente. Considerando además que la variación total es de 0.15%, por tanto, se puede decir que tanto lo cinemática directa como la inversa están desarrolladas correctamente y dan soluciones válidas al problema en cuestión.

Además, en la Tabla 3.2 se puede observar que los ángulos iniciales no coinciden con los ángulos calculados por la cinemática inversa. Esto se debe a que existen diferentes combinaciones angulares entre los cuatro servomotores que pueden conducir al efector final a una misma posición. Es decir, variando los valores angulares de los servomotores y combinándolos adecuadamente, es posible alcanzar la misma posición del efector final.

Tabla 3.2. Porcentaje de variación entre los resultados de la Cinemática Directa e Inversa.

N°	Ángulos iniciales				Posición del Efector Final calculado por la cinemática directa a partir de los ángulos iniciales			Ángulos Calculados por la Cinemática Inversa				Posición del Efector Final a partir de los ángulos calculador por la cinemática inversa			% de Variación			
	θ_0	θ_1°	θ_2	θ_3	P_X	P_Y	P_Z	θ_0	θ_1°	θ_2	θ_3	P_X	P_Y	P_Z	P_X	P_Y	P_Z	Total
1	0	130	78.6	110.2	-170.4	0	179.1	0	75.6	-165.5	-0.8	-170.3	0	179	0.06%	0.00%	0.06%	0.04%
2	60	160	20	30	-73.2	-126.9	368.3	60	159.6	19.9	30.8	-72.9	-126.2	368.6	0.41%	0.55%	0.08%	0.35%
3	-30	100	5	30	177.5	-102.5	311.2	-30	105.2	15.1	30	177.8	-102.7	310.7	0.17%	0.19%	0.16%	0.17%
4	-20	70	50	-30	265.7	-96.7	221.4	-19.9	76.3	24.5	0	265.8	-96.2	221.2	0.04%	0.52%	0.09%	0.22%
5	-120	20	-160	90	28.6	49.6	84.3	60	162.3	-152.9	93.9	28.6	49.7	84.2	0.00%	0.20%	0.12%	0.11%
6	142.1	124.6	103.5	-16.9	133.5	-104	368.8	-37.9	97.1	43.5	-1	133.7	-104.1	368.6	0.15%	0.10%	0.05%	0.10%
7	-118.3	131.9	143.5	-39.8	117	217.3	250.8	61.7	95.4	10.7	-2.5	117.1	217.5	250.4	0.09%	0.09%	0.16%	0.11%
8	-67.6	125.5	133.5	-40.8	-79.1	191.9	311.7	112.4	34	100.9	29.5	-79.2	192.3	311.3	0.13%	0.21%	0.13%	0.15%
9	-180	64.6	123.6	-25.7	-99	0	386.9	0	152.6	17.7	29	-98.5	0	387	0.51%	0.00%	0.03%	0.18%
10	135	73	64.2	-37.6	-182.4	182.4	269.7	135	24.9	118.9	-10.3	-182.5	182.5	269.4	0.05%	0.05%	0.11%	0.07%
															0.16%	**0.19%**	**0.10%**	**0.15%**

3.2. Tiempos de registro de la base de datos

A la postre, los siguientes resultados muestran los tiempos que tarda en registrarse los datos de posición antes y después de la ejecución de movimientos del brazo robótico. Estos datos son almacenados en una base de datos en un archivo en Excel, mismo que se encuentra disponible en el computador principal o host, puesto que es a donde llegan los datos enviados de forma local o remota. Una vez obtenidos los datos de tiempo en segundos a través de pruebas experimentales, son analizados a continuación.

Tabla 3.3. Tiempo en segundos que demora en registrarse la base de datos.

Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor
1	1,642	21	1,746	41	1,869	61	1,728	81	1,842
2	1,803	22	1,861	42	1,974	62	1,825	82	1,780
3	1,681	23	1,770	43	1,917	63	1,974	83	1,895
4	1,830	24	1,916	44	1,816	64	1,801	84	1,815
5	1,760	25	1,804	45	1,950	65	1,832	85	1,751
6	1,707	26	1,721	46	1,964	66	1,901	86	1,885
7	1,919	27	1,866	47	1,829	67	1,731	87	1,850
8	1,820	28	1,956	48	1,943	68	1,843	88	1,795
9	1,713	29	1,897	49	1,939	69	1,860	89	1,873
10	1,927	30	2,001	50	1,810	70	1,746	90	1,861
11	1,838	31	1,900	51	1,999	71	1,903	91	1,759
12	1,703	32	1,883	52	1,829	72	1,850	92	1,877
13	1,837	33	1,985	53	1,712	73	1,895	93	1,854
14	1,819	34	1,903	54	1,879	74	1,886	94	1,811
15	1,692	35	1,844	55	1,854	75	1,845	95	1,968
16	1,873	36	1,994	56	1,755	76	1,776	96	1,982
17	1,850	37	1,945	57	1,872	77	1,902	97	1,860
18	1,749	38	1,805	58	1,711	78	1,854	98	1,966
19	1,856	39	1,947	59	1,863	79	1,743	99	1,995
20	1,803	40	1,896	60	1,870	80	1,862	100	1,967

Interpretación:

Es así como, de la Tabla 3.3. acerca del tiempo de registro de actividad de la base de datos, hace referencia al tiempo que demora el sistema en registrar: fecha, tipo de usuario y posición, antes y después del posicionamiento deseado.

- **Tendencias temporales:** Se ha notado que los valores de muestra presentan variaciones en el tiempo, sugiriendo que pueden existir factores que ocasionan

dicha variabilidad, como puede ser por el tipo de procesador, cantidad de datos registrados, entre otros.

- **Variabilidad en los valores de muestra:** La variabilidad en los valores de muestra indica que el brazo robótico puede estar respondiendo a diversas condiciones o influencias que afectan las mediciones, destacando la importancia de un control adaptativo.
- **Rangos de valores:** Los valores mínimos y máximos muestran una gama de variación en las mediciones, proporcionando una comprensión de la fluctuación máxima y mínima experimentada en el control de posición del brazo robótico.

Cabe mencionar que, el análisis del registro de actividad en la base de datos ha sido fundamental para comprender las variaciones y tendencias en los valores de muestra en relación con la fecha y la secuencia de las muestras. La media del tiempo que tarda en registrar los datos en el archivo en Excel es 1.84 segundos, este tiempo indica que es relativamente bueno, puesto que no involucra procesos críticos que requieran una respuesta inmediata y considerando que son 23 datos los registrados.

3.3. Tiempo de respuesta según tipo de conexión

Los tiempos de respuesta hace referencia al tiempo que demoran un paquete de datos en llegar desde un punto A hasta un punto B. Este tiempo puede variar en función a la distancia entre dispositivos, tipo de conexión, ancho de banda, etc.

Las pruebas consisten en determinar el tiempo de comunicación entre Host – Local y Host-Remoto, estas se realizan mediante la lectura y escritura bidireccional de ficheros. En donde un fichero es un archivo de texto que almacena el valor de una variable, por tanto, dichos ficheros almacenarán el instante en el emisor envía los datos, y el instante en el que el receptor los recibe. Por tanto, al sacar la diferencia entre dichos tiempos se determinará el tiempo de respuesta en segundos.

3.3.1. Tiempo de respuesta remota.

Para lo conexión remota, cada computador se conectó a su modem mediante cable Ethernet, consiguiendo que estas se comuniquen mediante internet a través de la aplicación de Hamachi. A continuación, se muestran los tiempos en los que el emisor y receptor envía y recibe los datos respectivamente, y la diferencia de dichos tiempos son los tiempos de respuesta mediante una comunicación remota.

Tabla 3.4. Tiempo en segundo de respuesta remota.

Emisor (s)	Receptor (s)	Tiempo (s)	Emisor (s)	Receptor (s)	Tiempo (s)
256.7079390	256.0577185	0.6502205	569.1317482	568.4857621	0.6459861
263.8039143	263.1517212	0.6521931	574.3470705	573.698763	0.6483075
270.0676389	269.4097178	0.6579211	579.7366292	579.085762	0.6508672
275.6573665	275.006733	0.6506335	586.0926359	585.2317624	0.8608735
282.7640274	282.1107316	0.6532958	591.5139418	590.861763	0.6521788
288.3827882	287.5167348	0.8660534	597.2661912	596.622762	0.6434292
294.9538786	294.301734	0.6521446	601.3778138	600.7277598	0.650054
300.6899340	300.0367327	0.6532013	606.7179370	606.0707645	0.6471725
307.2034693	306.5527332	0.6507361	612.0899749	611.444763	0.6452119
312.8972464	312.243733	0.6535134	617.4244392	616.782777	0.6416622
318.6547450	318.0087528	0.6459922	623.3843293	622.7357621	0.6485672
325.9847136	325.3317332	0.6529804	630.0766310	629.4297636	0.6468674
332.2325463	331.577734	0.6548123	635.3674119	634.7237833	0.6436286
339.8838327	339.0177407	0.866092	641.0013912	640.3567617	0.6446295
346.2600250	345.609756	0.650269	645.8322425	645.1947734	0.6374691
351.6413672	351.001732	0.6396352	650.9532804	650.3057654	0.647515
358.1777573	357.5257316	0.6520257	656.2562153	655.6187692	0.6374461
364.2700303	363.8287342	0.4412961	660.9393096	660.5057647	0.4335449
370.4165750	369.9777374	0.4388376	667.5662014	666.9227653	0.6434361
374.9950962	374.347734	0.6473622	672.2988517	671.4357617	0.86309
380.9447038	380.2987525	0.6459513	678.0468976	677.181765	0.8651326
386.7351540	386.0967383	0.6384157	683.3367343	682.6937604	0.6429739
392.0591388	391.4107318	0.648407	689.4696522	688.608762	0.8608902
398.5335243	397.8847313	0.648793	695.5039134	694.6447592	0.8591542
404.2050360	403.3397634	0.8652726	701.0061193	700.146784	0.8593353
411.7745295	410.9087622	0.8657673	705.3413825	704.69476	0.6466225
416.2235281	415.5737596	0.6497685	710.8272905	710.3957622	0.4315283
422.3249414	421.4647622	0.8601792	716.5359368	715.8877618	0.648175
427.7711203	426.9077616	0.8633587	720.9741495	720.329767	0.6443825
436.8261561	436.1797605	0.6463956	726.1578170	725.5187614	0.6390556
443.7056575	442.842762	0.8628955	731.6500087	731.0187593	0.6312494
449.3093052	448.655768	0.6535372	738.5929742	737.7387595	0.8542147
457.4438484	456.5827637	0.8610847	744.5227113	743.8767614	0.6459499
464.7909458	463.9267607	0.8641851	750.6341930	749.7827601	0.8514329
470.1691656	469.5187614	0.6504042	755.7508974	755.1237655	0.6271319
476.7333932	475.8667626	0.8666306	763.0019906	762.1437619	0.8582287
482.7523732	482.1057682	0.646605	768.2948067	767.6457624	0.6490443
488.0417159	487.393762	0.6479539	774.4127445	773.7717626	0.6409819
493.6017606	492.9497669	0.6519937	779.2134054	778.5737603	0.6396451
499.4606338	498.5947614	0.8658724	785.2427878	784.3807676	0.8620202

504.7963324	504.1437643	0.6525681	789.7021039	789.0677617	0.6343422
511.5529943	510.690767	0.8622273	795.4068660	794.552761	0.854105
518.2637005	517.613763	0.6499375	800.0157123	799.5937605	0.4219518
525.5822823	524.9337623	0.64852	806.3654304	805.5087628	0.8566676
530.7748933	530.1207824	0.6541109	811.4643686	811.029763	0.4346056
537.7578506	536.8977623	0.8600883	817.0582554	816.4117775	0.6464779
544.8137321	544.1617603	0.6519718	821.3270488	820.6877623	0.6392865
550.6235910	549.759765	0.863826	826.2796724	825.8557603	0.4239121
555.5736980	554.928761	0.644937	831.7533557	831.109767	0.6435887
562.3905225	561.5287685	0.861754	836.5406170	835.6797643	0.8608527

Interpretación:

Con respecto al tiempo de comunicación remota, en la Tabla 3.4. se observa el tiempo de comunicación en segundos entre un emisor y un receptor, es así como se da cabida a la siguiente interpretación:

- **Tendencias Temporales:**
  El tiempo de comunicación varía en cada muestra, lo que sugiere fluctuaciones en la latencia de la comunicación entre el emisor y el receptor. No se observa una tendencia clara a lo largo del tiempo.
- **Variabilidad en los Tiempos de Comunicación:**
  Los tiempos de comunicación presentan variabilidad, esta puede deberse a diversas condiciones de red y carga de trabajo en los sistemas de comunicación.
- **Rangos de Tiempo:**
  Los tiempos de comunicación oscilan entre valores más bajos, alrededor de 0.42 segundos, y valores más altos, cerca de 0.87 segundos. Esto indica una variabilidad significativa en la latencia de la comunicación remota.
- **Análisis de Tendencias:**
  Aunque no se observa una tendencia clara en los tiempos de comunicación, podrían realizarse análisis más detallados para identificar patrones o tendencias ocultas que puedan ayudar a mejorar la eficiencia de la comunicación remota.

3.3.2. Tiempo de respuesta local

Para la obtención de estas pruebas se conectó los computadores anfitrión y local a un mismo modem mediante cable Ethernet. A continuación, se muestran los tiempos en los que el emisor y receptor envía y recibe los datos respectivamente, y la diferencia de dichos tiempos son los tiempos de respuesta mediante una comunicación local.

Tabla 3.5. Tiempo de respuesta local.

Emisor (s)	Receptor (s)	Tiempo (s)	Emisor (s)	Receptor (s)	Tiempo (s)
68.6167997	68.5100960	0.1067037	72.5589512	72.4522958	0.1066554
68.6807602	68.5733958	0.10736435	72.6337053	72.5269957	0.1067096
68.7450738	68.6378974	0.10717649	72.7042561	72.5973979	0.1068582
68.8245946	68.7176979	0.10689669	72.7693003	72.6628959	0.1064044
68.9335678	68.8271969	0.10637086	72.8369587	72.7301962	0.1067625
69.0027906	68.8959962	0.10679439	72.9054815	72.7990961	0.10638542
69.0739035	68.9672959	0.10660761	73.0400525	72.9334960	0.10655651
69.1409311	69.0339979	0.10693316	73.0971173	72.9898963	0.10722106
69.2113517	69.1050002	0.10635152	73.1575437	73.0502962	0.1072475
69.2768117	69.1694964	0.1073153	73.2197746	73.1133957	0.10637889
69.3432708	69.2364959	0.10677487	73.2808041	73.1737979	0.10700615
69.4151862	69.3080963	0.1070899	73.3488328	73.2419958	0.10683695
69.4790114	69.3721960	0.10681541	73.4917497	73.3847979	0.10695179
69.5448709	69.4387962	0.10607469	73.5503443	73.4430980	0.10724631
69.6303045	69.5229963	0.10730816	73.6070301	73.4997961	0.10723393
69.7036148	69.5965964	0.1070184	73.6645776	73.5574958	0.10708182
69.7784686	69.6711963	0.10727238	73.7201642	73.6138963	0.10626793
69.8590927	69.7525958	0.10649686	73.7789773	73.6725979	0.10637939
69.9360009	69.8295965	0.1064044	73.8380681	73.7307960	0.10727203
70.0127343	69.9061960	0.10653832	73.8982815	73.7918957	0.10638578
70.0879880	69.9807960	0.10719195	73.9551467	73.8478961	0.10725059
70.1520486	70.0460958	0.10595282	74.0115130	73.9044957	0.10701727
70.2202506	70.1137959	0.1064547	74.0729695	73.9658960	0.1070735
70.2895741	70.1829962	0.10657789	74.1368913	74.0304963	0.10639504
70.3587930	70.2520980	0.10669499	74.2016274	74.0952989	0.10632844
70.4375366	70.3306958	0.10684076	74.2705083	74.1638961	0.10661224
70.5091517	70.4027959	0.10635584	74.3342429	74.2273960	0.10684691
70.5765693	70.4696959	0.1068734	74.3937206	74.2869982	0.10672238
70.6406468	70.5338960	0.1067508	74.4516171	74.3443964	0.10722072
70.7119844	70.6046963	0.10728815	74.5108198	74.4040963	0.10672345
70.7787432	70.6721963	0.1065469	74.5662860	74.4590959	0.1071901
70.8442612	70.7370961	0.10716508	74.6268223	74.5200964	0.10672595
70.9049238	70.7979958	0.10692802	74.6881290	74.5811958	0.10693328

70.9650458	70.8579961	0.10704966	74.7542706	74.6474958	0.10677476
71.0275610	70.9207959	0.10676515	74.8115339	74.7048959	0.10663805
71.0903827	70.9830958	0.10728684	74.8678562	74.7605957	0.10726051
71.1571415	71.0501971	0.10694442	74.9158480	74.8085964	0.10725158
71.2245661	71.1182960	0.10627017	74.9692092	74.8619962	0.10721307
71.2910878	71.1843960	0.10669179	75.0298260	74.9227958	0.10703013
71.3553482	71.2481964	0.1071518	75.0901796	74.9829959	0.10718369
71.4218028	71.3149962	0.10680659	75.1493102	75.0429964	0.1063138
71.4855464	71.3784958	0.10705064	75.2068945	75.1002962	0.10659821
71.5602961	71.4538968	0.10639935	75.2664966	75.1599957	0.10650091
71.7221519	71.6149961	0.10715573	75.3297601	75.2231981	0.10656197
71.7859911	71.6790960	0.1068951	75.3978821	75.2919959	0.10588617
71.8600374	71.7530959	0.10694145	75.4620795	75.3548965	0.10718303
71.9374639	71.8305959	0.10686806	75.5720326	75.4648959	0.1071367
72.2856226	72.1786957	0.10692692	75.6415604	75.5344959	0.10706451
72.3456706	72.2386960	0.10697455	75.7140832	75.6071959	0.10688735
72.4152849	72.3079960	0.10728891	7,578,528,264	75.6780958	0.10718682

Interpretación:

Dando continuidad al análisis, en este caso se presenta la Tabla 3.5. que, al igual que los anteriores datos presenta una comunicación en segundos entre un emisor y un receptor, pero en un sistema de comunicación local. Como tal, se da cabida a la identificación de los siguientes datos:

- **Tendencias Temporales:**
  No se observa una tendencia clara en el tiempo de comunicación a lo largo de las muestras. El tiempo de comunicación fluctúa, pero no sigue una trayectoria específica.
- **Variabilidad en los Tiempos de Comunicación:**
  Los tiempos de comunicación presentan cierta variabilidad, con valores oscilan alrededor de un rango promedio 0.106829 segundos. Esta variabilidad puede deberse a factores locales, como la carga de trabajo o tipo de procesador.
- **Rangos de Tiempo:**
  Los tiempos de comunicación oscilan entre valores más bajos, alrededor de 0.105888 segundos, y valores más altos, cerca de 0.107364 segundos. Esto indica una variabilidad moderada en la latencia de la comunicación local.

Una vez analizados los tiempos de respuesta en la comunicación local y remota, se pude notar que en la conexión local existe una latencia baja o nula, mientras que la conexión

remota presenta una latencia notoria, ver Figura 3.1. Además, al comparar los promedios de estas se puede notar claramente que la comunicación remota tarda 6 veces más que la comunicación local, esto puede deberse principalmente a la distancia entre computadores y ancho de banda utilizado para la comunicación de estas.

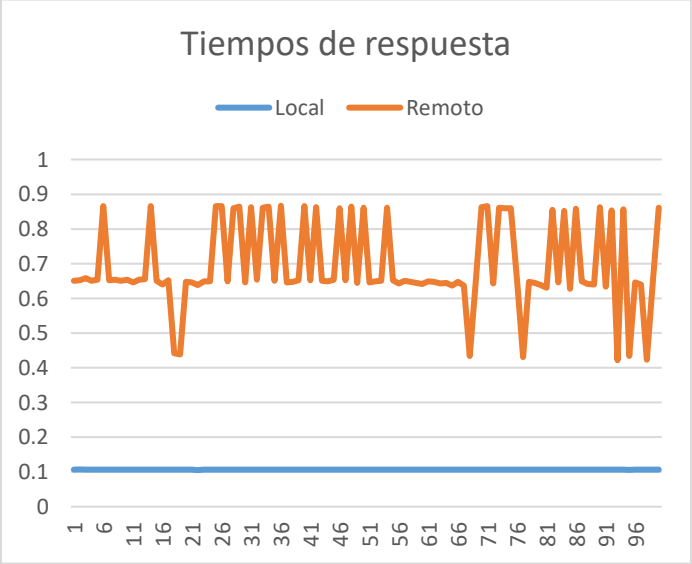


Figura 3.1. Tiempos de respuesta en las conexiones local y remota.

3.4. Tiempos de lectura y escritura de la posición actual del brazo robótico

Tabla 3.6. Tiempo en segundos de lectura de la posición actual del brazo.

Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor
1	0.012992	21	0.015987	41	0.01599	61	0.01699	81	0.015989
2	0.01499	22	0.016991	42	0.015991	62	0.01599	82	0.014991
3	0.016989	23	0.014989	43	0.015988	63	0.015993	83	0.016991
4	0.014992	24	0.015991	44	0.015989	64	0.018984	84	0.014988
5	0.016555	25	0.019988	45	0.016988	65	0.012994	85	0.01699
6	0.015989	26	0.011991	46	0.014991	66	0.020983	86	0.01499
7	0.014988	27	0.01599	47	0.01599	67	0.012993	87	0.015989
8	0.016992	28	0.01799	48	0.015989	68	0.01399	88	0.016991
9	0.014987	29	0.013989	49	0.01599	69	0.01599	89	0.01599
10	0.015989	30	0.017997	50	0.015989	70	0.01599	90	0.014988
11	0.01799	31	0.013984	51	0.015989	71	0.015988	91	0.016989
12	0.01599	32	0.017987	52	0.01599	72	0.015991	92	0.01499
13	0.01499	33	0.013991	53	0.017989	73	0.01599	93	0.016989
14	0.01699	34	0.01799	54	0.015993	74	0.016988	94	0.014991
15	0.014988	35	0.014989	55	0.013989	75	0.017989	95	0.019987

16	0.014991	36	0.01599	56	0.015987	76	0.01299	96	0.012991
17	0.015988	37	0.01599	57	0.01599	77	0.01599	97	0.014992
18	0.016989	38	0.015989	58	0.015989	78	0.017989	98	0.015987
19	0.014991	39	0.015991	59	0.01699	79	0.01399	99	0.015991
20	0.015991	40	0.01599	60	0.013991	80	0.016988	100	0.016991

Interpretación:

Por su parte, los datos que se observan en la Tabla 3.6. muestran el tiempo en segundos que se tarda en leer la posición actual del brazo robótico en diferentes muestras. De tal manera y, con el fin comprender lo que significan los enunciados datos se da cabida a la siguiente interpretación:

- **Tendencias Temporales:**
  No se observa una tendencia clara en el tiempo de lectura de la posición actual del brazo a lo largo de las muestras. Los tiempos fluctúan, pero no siguen un patrón específico.
- **Variabilidad en los Tiempos de Lectura:**
  Los tiempos de lectura presentan cierta variabilidad, con valores oscilando alrededor de un rango promedio de 0.01597132 . Esta variabilidad puede estar influenciada por diversos factores, como la carga del sistema o la precisión requerida.
- **Rangos de Tiempo:**
  Los tiempos de lectura oscilan principalmente entre valores cercanos a 0.011 segundos y 0.020 segundos. Existen excepciones, pero esta es la gama general de tiempos observados.
- **Análisis de Tendencias:**
  Los tiempos de lectura no muestran una dirección clara. Algunas muestras tienen tiempos ligeramente superiores o inferiores a otras, pero no se puede identificar una tendencia clara.
- **Picos en el Tiempo de Lectura:**
  Algunas muestras, como la muestra 66 y 95, presentan picos en el tiempo de lectura que son más altos que la mayoría de las otras muestras. Esto puede indicar posibles problemas intermitentes o fluctuaciones en el sistema.

Tabla 3.7. Tiempo en segundos de escritura para posicionar el brazo.

Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor	Muestra	Valor
1	0.0115509	21	0.0112959	41	0.0112329	61	0.0112707	81	0.011006
2	0.0110004	22	0.0111015	42	0.0112992	62	0.0112865	82	0.0112675
3	0.0113086	23	0.011375	43	0.0110603	63	0.0113393	83	0.0111241
4	0.0111003	24	0.0111804	44	0.0114406	64	0.0109996	84	0.0112029
5	0.0112487	25	0.0111441	45	0.0111272	65	0.011193	85	0.0112703
6	0.0112748	26	0.0112052	46	0.0110255	66	0.0111931	86	0.0112816
7	0.011138	27	0.0112874	47	0.0112553	67	0.0111003	87	0.0112494
8	0.011359	28	0.0111422	48	0.0111117	68	0.0113131	88	0.0110013
9	0.0110269	29	0.0111502	49	0.0112507	69	0.0111972	89	0.0112753
10	0.0112225	30	0.011269	50	0.0117356	70	0.0111985	90	0.0111007
11	0.0112014	31	0.0111452	51	0.0123333	71	0.0112042	91	0.0112104
12	0.0111391	32	0.0111045	52	0.0111753	72	0.0111035	92	0.0112036
13	0.0112542	33	0.0115517	53	0.0112825	73	0.0112948	93	0.0114076
14	0.0112081	34	0.0109487	54	0.0110384	74	0.0112523	94	0.0110931
15	0.0111009	35	0.0111984	55	0.0112797	75	0.0111507	95	0.0111026
16	0.0112442	36	0.0112792	56	0.0110996	76	0.0111757	96	0.0112002
17	0.0112105	37	0.0110926	57	0.011331	77	0.0111231	97	0.0112618
18	0.0111997	38	0.0113302	58	0.0111673	78	0.0111994	98	0.0112291
19	0.011241	39	0.0112433	59	0.0113178	79	0.0112005	99	0.0111127
20	0.011101	40	0.0110065	60	0.010988	80	0.0113991	100	0.0112516

Interpretación:

En este caso, los datos presentados en la Tabla 3.7., tratan acerca del tiempo en segundos que se tarda en escribir la posición actual del brazo robótico en diferentes muestras. Denota los siguientes hallazgos:

- **Tendencias Temporales:**
  No se observa una tendencia clara en el tiempo de escritura de la posición actual del brazo a lo largo de las muestras. Los tiempos fluctúan, pero no siguen un patrón específico.
- **Variabilidad en los Tiempos de Escritura:**
  Los tiempos de escritura presentan cierta variabilidad, con valores oscilando alrededor de un rango promedio de 0.0112198. Esta variabilidad puede estar influenciada por diversos factores, como la carga del sistema o la precisión requerida.

- **Rangos de Tiempo:**
  Los tiempos de escritura oscilan principalmente entre valores cercanos a 0.010988 segundos y 0.0123333 segundos. Existen excepciones, pero esta es la gama general de tiempos observados.
- **Análisis de Tendencias:**
  Los tiempos de escritura no muestran una dirección clara. Algunas muestras tienen tiempos ligeramente superiores o inferiores a otras, pero no se puede identificar una tendencia clara.
- **Picos en el Tiempo de Escritura:**
  Algunas muestras, como la muestra 51 y 34, presentan picos en el tiempo de escritura que son más altos que la mayoría de las otras muestras. Esto puede indicar posibles problemas intermitentes o fluctuaciones en el sistema.

Por tanto, como se observa en la Figura 3.2., se puede evidenciar que el tiempo de lectura es relativamente superior al tiempo de escritura, y a su vez es el que mayor variación presenta, esto puede deberse al tipo de hardware y software utilizado. Sin embargo, en lectura y escritura los tiempos promedios oscilan entre 0.016 y 0.011 respectivamente, en donde por la naturaleza misma del sistema no representa mayor inconveniente.

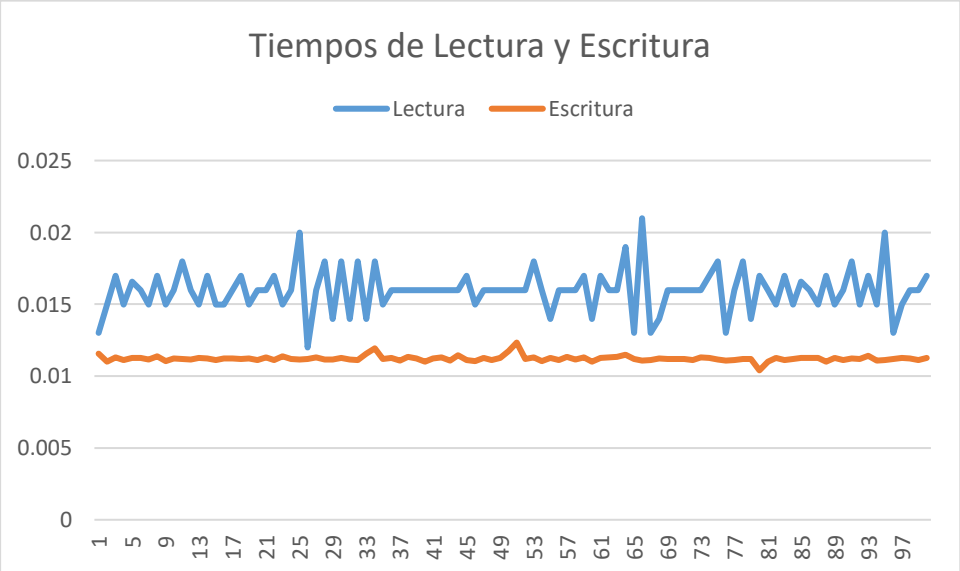


Figura 3.2. Tiempos de lectura y escritura de la posición actual del brazo.

3.5. Análisis general de resultados

En lo referente a la aplicación del método geométrico y el algoritmo de Denavit Hartenberg para determinar las ecuaciones de la cinemática directa, se ha identificado que cada uno tiene diferentes niveles de complejidad debido a la estructura matemática que aplican. Fueron aplicados los dos algoritmos a fin de realizar una comparación entre sus soluciones, con el fin de corroborar que las ecuaciones obtenidas con cada método son las que describen correctamente el comportamiento del brazo robótico. Y efectivamente, se logró llegar al mismo resultado con ambos métodos. La importancia de la aplicación de ambos métodos radica en que el primero en mención es de fácil aplicación, pero sus ecuaciones se basan en cuatro variables P_X, P_y, P_Z y la proyección de la resultante en el plano XY. Por su parte, el segundo método indicado es más extenso y complejo, pero las ecuaciones obtenidas por este permiten calcular la cinemática inversa a través de herramientas propias de Python, debido a que se fundamentan únicamente en las tres variables de posición del efector final, P_X, P_y, y P_Z.

Una vez realizado el algoritmo que represente la cinemática inversa del brazo robótico; se ha realizado varias pruebas para determinar la efectividad del código desarrollado, este consiste en calcular los ángulos en cada nodo para que el efector final adopte la posición deseada. En las cuales, se ha notado que existe una variabilidad de 0.15% con respecto a la posición calculada por medio de la cinemática directa, la variación es relativamente baja por lo que se puede decir que el algoritmo planteado es correcto.

En lo referente a los tiempos de respuesta para el registro de actividad en la base de datos, este oscila entre 1 y 2 segundos, esta variabilidad puede estar asociada a la cantidad de datos registrados, y también a las condiciones del sistema operativo utilizado, sin embargo, al no ser un sistema critico que requiere los datos de forma instantánea, los tiempos registro obtenidos son aceptables.

Por su parte, los registros de los tiempos que tarda en llegar la información desde el emisor al receptor, según el tipo de conexión utilizada, han mostrado información valiosa, puesto que la conexión local tarda en promedio 0.1068 segundos, mientras que la conexión remota presenta una media de 0.6919 segundos, por lo que, es fácil evidenciar que existe una latencia significativa al comparar dichas conexiones, debido a que la conexión remota demora 6 veces más que la conexión local. Existen varios factores que pueden generar dicho retraso, como el ancho de banda, congestión de la red de internet, ejecución excesiva

de aplicaciones, distancia de entre computadores, comunicación mediante red VPN, entre otros.

En el análisis de los tiempos de lectura y escritura de la posición del brazo robótico, se observa una variabilidad y fluctuación en los tiempos. Las fluctuaciones observadas en estos tiempos pueden estar relacionadas con la carga del sistema o la precisión requerida en la lectura y escritura de la posición.

En general, en algunos conjuntos de datos, se observan picos en ciertas muestras, indicando posibles problemas intermitentes o fluctuaciones en el sistema. En la presente investigación los valores o tiempos promedios obtenidos en las diferentes pruebas son aceptables, puesto que no se está controlando un sistema crítico.

4. DISCUCIÓN

Al examinar los datos de tiempo de comunicación entre un emisor y un receptor, se identifica una variabilidad sustancial en estos tiempos. Los valores oscilan entre 0.11 y 0.69 segundos, lo que indica fluctuaciones significativas en la latencia de la comunicación remota. Estas fluctuaciones pueden atribuirse a factores como condiciones de red y carga de trabajo en los sistemas de comunicación. Es crucial tener en cuenta esta variabilidad al momento de diseñar una interfaz de comunicación, si bien es cierto que en la presente investigación el proceso controlado no es un sistema crítico, eso no quiere decir que los tiempos de comunicación no pueda ser mejorado, para de tal forma conseguir una comunicación remota más estable y eficiente.

De tal manera, este análisis detallado de los datos de rendimiento y tiempos de comunicación proporciona una visión valiosa para el diseño e implementación de una interfaz de comunicación eficaz tanto para control local como remoto de un brazo robótico. Se destaca la necesidad de futuras investigaciones y desarrollos que aprovechen estos hallazgos para mejorar la interfaz y maximizar el rendimiento del sistema en el control de posición del brazo robótico.

Por otro lado, es importante mencionar que el presente trabajo se ha basado en el desarrollo de una interfaz de comunicación y una interfaz gráfica de usuario para el control del brazo robótico PincherX 100 mediante la aplicación de la cinemática directa e inversa, mediante el uso de Python. Entonces, el brazo robótico puede ser controlado desde un computador a través de la interfaz gráfica de usuario, sin embargo, el autómata presenta una amplia gama de funciones, entre ellas el control de movimientos mediante un control

remoto, muy similar a una palanca de juegos, esto es posible gracias a los paquetes proporcionados por el Sistema Operativo Robótico (ROS), que a su vez estos son compatibles con los autómatas de la serie X y con el lenguaje de programación Python.

Por tanto, también se puede dar continuidad a la presente investigación mediante la implementación de una palanca de mando remoto para el control de movimientos del brazo robótico PincherX 100, aplicando la cinemática directa e inversa, siendo estas últimas ya desarrolladas en este documento.

5. CONCLUSIONES

- Se ha logrado realizar el diseño e implementación de una interfaz de comunicación para el control de movimientos del brazo robótico PincherX 100 de forma local y remota, mediante el uso Python como lenguaje de programación, lo cual ha resultado beneficioso puesto que este es de fácil comprensión y sobre todo es de código abierto, por lo que el desarrollo del código no ha representado costos. Además, los servomotores DYNAMIXEL XL430-W250-T con los que se ha trabajado en este proyecto, cuentan con su propio paquete de desarrollo llamado DINAMIXEL SDK. Tanto el lenguaje de programación como el paquete han permitido establecer la comunicación directa entre el computador y el autómata, así como también han hecho posible programar los modelos cinemáticos directo e inverso para el control eficiente de los movimientos de la plataforma robótica, considerando además que han permitido el desarrollo del código de la interfaz gráfica de usuario para la transmisión y recepción de datos.
- Mediante la revisión bibliográfica exhaustiva, se concluye que existe una gran variedad de brazos robóticos, en donde sus grados de libertad y estructura misma son lo más importante a considerar para analizar y desarrollar de forma adecuada la cinemática directa e inversa. Se han revisado varios lenguajes de programación, y mediante un análisis comparativo de estos se ha determinado que Python tiene grandes ventajas en comparación a otros lenguajes. Con lo referente al control de movimientos de forma local y remota, existen varias metodologías y enfoques que posibilitan la ejecución de este requerimiento. En sí, la revisión bibliográfica ha brindado información sólida que permite aplicar estrategias de forma efectiva para desarrollar el presente proyecto.
- El objetivo de obtener los modelos cinemáticos directo e inverso del brazo robótico se ha cumplido parcialmente. Puesto que el modelo cinemático directo ha sido obtenido mediante la aplicación del método geométrico y el algoritmo de Denavit Hartenberg, consiguiendo alcanzar una comprensión profunda de las ecuaciones que describen el movimiento del brazo robótico en función de sus grados de libertad y a su vez mediante la comparación de métodos se ha comprobado que el análisis planteado es correcto puesto que las ecuaciones de ambas metodologías satisfacen al comportamiento real del brazo robótico. Sin embargo, es relevante señalar que, aunque la obtención del modelo cinemático inverso no se ha realizado de manera convencional, se ha superado este obstáculo mediante la

implementación de bibliotecas de Python especializadas, para esto se ha tomado como base las tres ecuaciones obtenidas a través del método de Denavit Hartenberg, logrando obtener soluciones validas. Al realizar un análisis comparativo entre los resultados mostrados por la cinemática directa e inversa, se ha evidenciado una variación de apenas el 0.15%. Se puede inferir entonces, que los métodos aplicados son correctos y se rigen a las características físicas de autómata.

- Se ha diseñado e implementado las interfaces de comunicación de forma satisfactoria, logrando un control efectivo del brazo robótico tanto en entornos locales como remotos. Para esto se ha utilizado un computador principal o host, el cual va a estar conectado directamente al brazo robótico y a su vez es el que recibirá las instrucciones enviadas desde el computador local o remoto. Las diferentes herramientas utilizadas, como: Python, paquete de desarrollo Dynamixel SDK, Hamachi by LogMeIn y las configuraciones para crear una red local, han sido fundamentales para establecer una correcta comunicación entre los diferentes dispositivos que intervienen en el desarrollo de la presente investigación. Se ha logrado identificar que los tiempos de respuesta de una conexión remota, tarda seis veces con respecto a los tiempos de una conexión local, cuyos valores promedios son 0.69 y 0.11 respectivamente, por tanto, se puede inferir que el brazo robótico al no ser un sistema crítico, los tiempos de respuesta obtenidos son aceptables, sin embargo, no se descarta la posibilidad de mejorar los tiempos en mención.
- La implementación del control de posición en lazo cerrado marca un logro crucial, facilitado en gran medida por la presencia y eficacia del control PID incorporado en los servomotores utilizados. La adopción del control de lazo cerrado no solo garantiza una mayor precisión en el posicionamiento del brazo robótico, sino que también establece las bases para una operación más estable. Es vital destacar que la funcionalidad del lazo cerrado, respaldada por el control PID, contribuye significativamente a la estabilidad del sistema y sienta las bases para una interacción más robusta entre el usuario y el brazo robótico, ya sea a nivel local o remoto.
- Es posible realizar el control de posición de un brazo robótico mediante el uso de softwares libres y de código abierto, logrando deslindarse del software y requerimientos del fabricante, con lo que se consiguen disminuir los costos de adquisición de este tipo de plataformas robóticas. Además, mediante el uso de diferentes aplicaciones es posible controlar los movimientos del autómata de forma

local y remota, con lo que se consigue centralizar la información para poder controlar todo un proceso desde un solo lugar.

6. RECOMENDACIONES

- Para la creación de interfaces se recomienda revisar alternativas de software que no involucren costos, y que además que sean compatibles entre sí, de esta forma se consigue prototipos funcionales y que no requieren de inversión económica.
- En lo referente a la bibliografía, se debe revisar información actualizada, puesto que la tecnología robótica está en constante evolución y cada día surgen nuevos mecanismos o métodos que ayudan a un control más eficiente de las plataformas robóticas. De igual manera sucede con las bibliotecas de Python, debido a la gran cantidad de usuarios que utilizan este lenguaje de programación y crean códigos o bibliotecas que son de gran ayuda para la resolución diferentes problemas.
- Previo a la implementación de la cinemática directa. Es importante realizar un análisis estructural de la plataforma robótica en cuestión, es decir definir la cantidad de eslabones y los grados de libertad con los que trabaja. De preferencia se recomienda utilizar las medidas reales de los eslabones, con el finde evitar desfaces angulares. En el caso del brazo robótico PincherX 100, este tiene dos eslabones ensamblados de forma fija, los cuales mantienes un ángulo de -90°, para trabajar con medidas reales y evitar desfaces en los ángulos, se aumentó un grado de libertad correspondiente a dicho ángulo, y de esta forma se pudo obtener directamente las ecuaciones que describen el comportamiento del autómata.
- Con respecto a la comunicación remota, se recomienda analizar la latencia existente y determinar las posibles causas que la provocan, esto con el fin de que se mejoren los tiempos de respuesta, ya sea aplicando protocolos de comunicación más eficientes o explorando otras alternativas.
- Se recomienda ampliar las pruebas, en donde el brazo robótico sea sometido a diferentes escenarios o situaciones, como podría ser a condiciones extremas de carga, uso de diferentes aplicaciones o configuraciones para la conexión remota, entre otras, esto con el fin de mejorar su respuesta frente a diferentes ambientes.

7. REFERENCIAS BIBLIOGRÁFICAS

[1] Ministerio de Producción, «Boletín de cifras del sector productivo. Mayo 2023.,» 2023. [En línea]. Disponible: https://www.produccion.gob.ec/wp-content/uploads/2023/05/Boletin-Cifras-Productivas-MAYO2023.pdf.

[2] M. Mead, «Los robots permiten a las empresas hacer frente a los retos económicos,» IFR International Federation of Robotics, 13 octubre 2022. [En línea]. Disponible: https://ifr.org/post/robots-enable-companies-to-meet-economic-challenges.

[3] HITBOT, «¿Cuánto cuesta un brazo robótico?,» HITBOT, 14 febrero 2023. [En línea]. Disponible: https://www.hitbotrobot.com/es/un-brazo-robotico/.

[4] G. Bottari, «Robótica en la nube: qué es, cómo funciona, ejemplos y aplicaciones,» Innovación Digital 360, 4 enereo 2022. [En línea]. Disponible: https://www.innovaciondigital360.com/cloud/cloud-computing/robotica-en-la-nube-que-es-como-funciona-ejemplos-y-aplicaciones/.

[5] Trossen Robotics, «PincherX 100 Robot Arm,» Trossen Robotics, s.f. [En línea]. Disponible: https://www.trossenrobotics.com/pincherx-100-robot-arm.aspx.

[6] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, Fundamentos de Robótica, 2da Edición, Madrid: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U., 2007, pp. 16-18-31-32-33-51-119-120-121-122-125-126-127-128.

[7] J. R. Garibay Pascual , «Robótica: Estado del arte,» Universidad de Deusto, 2006. [En línea]. Disponible: https://scholar.google.es/citations?view_op=view_citation&hl=es&user=AGRglEIAAAAJ&citation_for_view=AGRglEIAAAAJ:2osOgNQ5qMEC.

[8] Universisdad Santiago de Chile, «Elementos Constitutivos de un Robot Industrial,» Universisdad Santiago de Chile, s. f.. [En línea]. Disponible: http://www.udesantiagovirtual.cl/moodle2/mod/book/view.php?id=24908&chapterid=211.

[9] J. Perez, «Diseño y Construcción de un Brazo Robótico Manipulador Móvil,» Sexto Coloquio Interdisciplinario de Doctorado – Universidad Popular Autónoma del Estado de Puebla, 2018. [En línea]. Disponible: https://upaep.mx/micrositios/coloquios/coloquio2013/memorias/Mesa%204%20Mec%20y%20Bio/4.-Meliton%20Fernando%20Santiago.pdf.

[10] Trossen Robotics, «PincherX 100,» Trossen Robotics, s.f. [En línea]. Disponible: https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications/px100.html.

[11] Trossen Robotics, «X- Serie Arms Documentation, Especificaciones,» Trossen Robotics, s. f. [En línea]. Disponible: https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications.html.

[12] Trossen Robotics, « Robotis X-Series DYNAMIXEL Servos de robot,» Trossen Robotics, s. f. [En línea]. Disponible: https://www.trossenrobotics.com/dynamixel-xl430-w250-t.aspx.

[13] Robotis, «DYNAMIXEL XL430 - W250 - T,» Robotis, s. f. [En línea]. Disponible: https://emanual.robotis.com/docs/en/dxl/x/xl430-w250/.

[14] Robotics, «U2D2,» Robotics, s. f. [En línea]. Disponible: https://emanual.robotis.com/docs/en/parts/interface/u2d2/.

[15] F. Behrouz A., Data Communicactions and Networking, 4ta Edición, McGraw Hill, 2000, pp. 13-14.

[16] A. S. Tanenbaum y D. J. , Redes de Computadoras, 5ta edición, Naucalpan de Juárez, Estado de México: Pearson Educación de México, 2012, pp. 20-21.

[17] S. Cruzado, «Escritorio remoto: qué es, para qué sirve y qué ventajas tiene,» Infor Systems, 2021. [En línea]. Disponible: https://winforsystems.com/escritorio-remoto-que-es-para-que-sirve-y-que-ventajas-tiene/.

[18] LogMeIn, «¿Qué es Hamachi?,» Go To, 27 septiembre 2022. [En línea]. Disponible: https://support.logmeininc.com/hamachi/help/what-is-logmein-hamachi-hamachi-c-hamachi-abouthamachi.

[19] UNS, «Interfaces Gráficas de Usuario,» s. f. Universisdad Nacional del Sur (UNS). [En línea]. Disponible: https://cs.uns.edu.ar/~mlg/ipoo/downloads/Presentaciones%20y%20Apuntes/5.GUI/ApuntesIPOO_GUI.pdf.

[20] UNAM, «Lenguajes de Programación,» Universidad Nacional Autónoma de México, s. f. [En línea]. Disponible: http://fcasua.contad.unam.mx/apuntes/interiores/docs/98/4/informatica_4.pdf.

[21] Python, «El tutorial de Python,» python.org, s. f.. [En línea]. Disponible: https://docs.python.org/es/3/tutorial/.

[22] Python, «PyGame,» python.org, s. f.. [En línea]. Disponible: https://wiki.python.org/moin/PyGame.

[23] Python, «math — Funciones matemáticas,» python.org, s. f.. [En línea]. Disponible: https://docs.python.org/es/3/library/math.html.

[24] Python, «openpyxl - Trabajando con Hojas de Calculo en Python,» pythondiario.com, 2019. [En línea]. Disponible: https://pythondiario.com/2019/01/trabajando-con-hojas-de-calculo-python-openpyxl.html.

[25] Python, «time — Acceso a tiempo y conversiones,» python.org, s. f.. [En línea]. Disponible: https://docs.python.org/es/3/library/time.html.

[26] J. C. Moreno Pérez, Entornos de Desarrollo, España: Sinresis, 2018, pp. 36-37.

[27] C. D. Días Bravo, Diseño mecánico y generación de trayectoria para un robot antropomórfico de 6 gradosde libertad, Curicó, Chile, 2022, pp. 19-20-21.

[28] Robotis, «DINAMIXEL Wizard 2.0,» robotis.com, s. f.. [En línea]. Disponible: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/.

[29] Robotis, «DINAMIXEL SDK,» robotis.com, s. f.. [En línea]. Disponible: https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/.

[30] Visual Studio, «Códigode estudio visual,» visualstudio.com, s. f.. [En línea]. Disponible: https://code.visualstudio.com/docs.

[31] Trossen Robotics, «Interfaz Python-ROS,» trossenrobotics.com, s. f.. [En línea]. Disponible:https://www.trossenrobotics.com/docs/interbotix_xsarms/python_ros_interface/index.html.

[32] Scipy, «Optimización y búsqueda de raíces (scipy.optimize),» scipy.org, [En línea]. Disponible: https://docs.scipy.org/doc/scipy/reference/optimize.html.

[33] Darfix, «envolverTo2pi,» Gitlab Pages, s. f.. [En línea]. Disponible: https://xrd.gitlab-pages.esrf.fr/darfix/_generated/darfix.core.utils.wrapTo2pi.html.

[34] Tech Takes, «¿Cuáles son los tamaños más comunes de monitores y cuál es el mejor?,» hp.com, 28 abril 2021. [En línea]. Disponible: https://www.hp.com/mx-es/shop/tech-takes/cuales-son-los-tamanos-mas-comunes-de-monitores-y-cual-es-el-mejor.

[35] Microsoft, «Configurar la red de su pequeña empresa,» microsoft.com, 17 julio 2023. [En línea]. Disponible: https://learn.microsoft.com/es-es/troubleshoot/windows-client/networking/set-up-your-small-business-network.

[36] Go To, «How to Add a Hamachi Network,» logmeininc.com, [En línea]. Disponible: https://support.logmeininc.com/central/help/how-to-add-a-hamachi-network-central-t-hamachi-nw-add.

8. ANEXOS

Anexo A. Código de la Cinemática Directa – Método Geométrico

Anexo B. Código de la Cinemática Directa – Método de Denavit Hartenberg

Anexo C. Matriz de Transformación Total – Método de Denavit Hartenberg

Anexo D. Código de la Cinemática Inversa – Herramientas propias de Python

ANEXO A

```
# CINEMÁTICA DIRECTA – MÉTODO GEOMÉTRCO

import numpy as np
from sympy import symbols, cos, sin, tan, pi


#VALORES COORDENADAS ANGULARES
theta0 = np.radians(33)  # Ángulo en radianes
theta1 = np.radians(110)  # Ángulo en radianes
theta2 = np.radians(-81)   # Ángulo en radianes
theta3 = np.radians(59)   # Ángulo en radianes

thetaX = np.radians(-90)


# Longitudes de los eslabones
L0 = 89.45
L1 = 100
L2 = 35
L3 = 100
L4 = 129.15


# Obtener las coordenadas X, Y y Z del efector final
r_final=L1*cos(theta1)+L2*cos(theta1+thetaX)+L3*cos(theta1+thetaX+theta2)
        +L4*cos(theta1+theta2+theta3+thetaX)

z_final=L0+L1*sin(theta1)+L2*sin(theta1+thetaX)+L3*sin(theta1+thetaX+theta2)
        +L4*sin(theta1+theta2+theta3+thetaX)

y_final = r_final*sin(theta0)

x_final = r_final*cos(theta0)

# Imprimir las coordenadas finales
print("Método Geométrico")
print("Posición final del efector")
print(f"Coordenada X: {x_final}")
print(f"Coordenada Y: {y_final}")
print(f"Coordenada Z: {z_final}")
```

ANEXO B

```
# CINEMATICA DIRECTA – MÉTODO DE DENAVIT HARTENBERG

import numpy as np
from sympy import symbols, cos, sin, tan, pi

theta0 = np.radians(33)  # Ángulo en radianes
theta1 = np.radians(110)  # Ángulo en radianes
theta2 = np.radians(-81)   # Ángulo en radianes
theta3 = np.radians(59)   # Ángulo en radianes

# Longitudes de los eslabones
L0 = 89.45
L1 = 100
L2 = 35
L3 = 100
L4 = 129.15

theta=[theta0,theta1,-pi/2,theta2,theta3] #Ángulo de rotación alrededor del eje z
d = [L0, 0, 0, 0, 0]  # Desplazamientos a lo largo del eje z
a = [0, L1, L2, L3, L4]  # Desplazamientos a lo largo del eje x
alpha = [pi/2, 0, 0, 0, 0]  # Ángulos de rotación alrededor del eje x

# Matrices de transformación homogénea
def dh_transform(theta, d, a, alpha):
    return np.array([
        [cos(theta),-sin(theta)*cos(alpha),  sin(theta)*sin(alpha), a*cos(theta)],
        [sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha), a*sin(theta)],
        [0,           sin(alpha),            cos(alpha),            d],
        [0,           0,                     0,                     1]
    ])

# Cálculo de las matrices de transformación
T_matrices = []
for i in range(5):
    T = dh_transform(theta[i], d[i], a[i], alpha[i])
    T_matrices.append(T)

# Matriz de transformación total
T_total = np.eye(4)
for T in T_matrices:
    T_total = np.dot(T_total, T)

# Obtener las coordenadas X, Y y Z del efector final
x_final = T_total[0, 3]
y_final = T_total[1, 3]
z_final = T_total[2, 3]

# Imprimir las coordenadas finales
print("Método Denavit Hartenberg")
print("Posición final del efector")
print(f"Coordenada X: {x_final}")
print(f"Coordenada Y: {y_final}")
print(f"Coordenada Z: {z_final}")
```

ANEXO C

Matriz de transformación total:

$$\begin{bmatrix} (-\sin(\theta_1)\cdot\sin(\theta_2)\cdot\cos(\theta_0) + \cos(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_3) + (\sin(\theta_1)\cdot\cos(\theta_0)\cdot\cos(\theta_2) + \sin(\theta_2)\cdot\cos(\theta_0)\cdot\cos(\theta_1))\cdot\cos(\theta_3) & (-\sin(\theta_1)\cdot\sin(\theta_2)\cdot\cos(\theta_0) + \cos(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\cos(\theta_3) - (\sin(\theta_1)\cdot\cos(\theta_0)\cdot\cos(\theta_2) + \sin(\theta_2)\cdot\cos(\theta_0)\cdot\cos(\theta_1))\cdot\sin(\theta_3) \\ (-\sin(\theta_0)\cdot\sin(\theta_1)\cdot\sin(\theta_2) + \sin(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_3) + (\sin(\theta_0)\cdot\sin(\theta_1)\cdot\cos(\theta_2) + \sin(\theta_0)\cdot\sin(\theta_2)\cdot\cos(\theta_1))\cdot\cos(\theta_3) & (-\sin(\theta_0)\cdot\sin(\theta_1)\cdot\sin(\theta_2) + \sin(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\cos(\theta_3) - (\sin(\theta_0)\cdot\sin(\theta_1)\cdot\cos(\theta_2) + \sin(\theta_0)\cdot\sin(\theta_2)\cdot\cos(\theta_1))\cdot\sin(\theta_3) \\ (\sin(\theta_1)\cdot\sin(\theta_2) - \cos(\theta_1)\cdot\cos(\theta_2))\cdot\cos(\theta_3) + (\sin(\theta_1)\cdot\cos(\theta) + \sin(\theta_2)\cdot\cos(\theta_1))\cdot\sin(\theta_3) & -(\sin(\theta_1)\cdot\sin(\theta_2) - \cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_3) + (\sin(\theta_1)\cdot\cos(\theta_2) + \sin(\theta_2)\cdot\cos(\theta_1))\cdot\cos(\theta_3) \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \sin(\theta_0) & L_1\cdot\cos(\theta_0)\cdot\cos(\theta_1) + L_2\cdot\sin(\theta_1)\cdot\cos(\theta_0) + L_3\cdot\sin(\theta_1)\cdot\cos(\theta_0)\cdot\cos(\theta_2) + L_3\cdot\sin(\theta_2)\cdot\cos(\theta_0)\cdot\cos(\theta_1) + L_4\cdot(-\sin(\theta_1)\cdot\sin(\theta_2)\cdot\cos(\theta_0) + \cos(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_3) + L_4\cdot(\sin(\theta_1)\cdot\cos(\theta_0)\cdot\cos(\theta_2) + \sin(\theta_2)\cdot\cos(\theta_0)\cdot\cos(\theta_1))\cdot\cos(\theta_3) & P_X \\ -\cos(\theta_0) & L_1\cdot\sin(\theta_0)\cdot\cos(\theta_1) + L_2\cdot\sin(\theta_0)\cdot\sin(\theta_1) + L_3\cdot\sin(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2) + L_3\cdot\sin(\theta_0)\cdot\sin(\theta_2)\cdot\cos(\theta_1) + L_4\cdot(-\sin(\theta_0)\cdot\sin(\theta_1)\cdot\sin(\theta_2) + \sin(\theta_0)\cdot\cos(\theta_1)\cdot\cos(\theta_2))\cdot\sin(\theta_3) + L_4\cdot(\sin(\theta_0)\cdot\sin(\theta_1)\cdot\cos(\theta_2) + \sin(\theta_0)\cdot\sin(\theta_2)\cdot\cos(\theta_1))\cdot\cos(\theta_3) & P_Y \\ 0 & L_0 + L_1\cdot\sin(\theta_1) - L_2\cdot\cos(\theta_1) + L_3\cdot\sin(\theta_1)\cdot\sin(\theta_2) - L_3\cdot\cos(\theta_1)\cdot\cos(\theta_2) + L_4\cdot(\sin(\theta_1)\cdot\sin(\theta_2) - \cos(\theta_1)\cdot\cos(\theta_2))\cdot\cos(\theta_3) + L_4\cdot(\sin(\theta_1)\cdot\cos(\theta_2) + \sin(\theta_2)\cdot\cos(\theta_1))\cdot\sin(\theta_3) & P_Z \\ 0 & 1 & \end{bmatrix}$$

ANEXO D

```
# CINEMÁTICA INVERSA – HERRAMIENTAS PROPIAS DE PYTHON

import numpy as np
from scipy.optimize import fsolve

def wrap_to_2pi(angle):
    return (angle + np.pi) % (2 * np.pi) - np.pi

# PARAMETROS DEL ROBOT
L0 = 0.08945  #metros
L1 = 0.1   #metros
L2 = 0.035     #metros
L3 = 0.1   #metros
L4 = 0.12915     #metros

# ANGULOS O CONDICIONES INICIALES INICIALES
thetha0 = 0* (np.pi / 180)  # Ángulo de la base [rad]
thetha1 = 45 * (np.pi / 180)  # Ángulo del eslabon 1 [rad]
thetha2 = 90 * (np.pi / 180)  # Ángulo del eslabon 2 [rad]
thetha3 = 0 * (np.pi / 180)  # Ángulo del eslabon 3 [rad]

# CINEMÁTICA INVERSA
PX = 160.1  #milimetros
PY = -58.2   #milimetros
PZ = 179.1  #milimetros

hx, hy, hz = PX/1000, PY/1000, PZ/1000   #metros

q0 = np.array([thetha0, thetha1, thetha2, thetha3])
h = np.array([hx, hy, hz])

def brazo(q, h):
    hx, hy, hz = h

    F = np.empty(4)  # Cambiar el tamaño a 5

    F[0] = 1.0*L1*np.cos(q[0])*np.cos(q[1]) + 1.0*L2*np.sin(q[1])*np.cos(q[0])
           + 1.0*L3*np.sin(q[1])*np.cos(q[0])*np.cos(q[2])
           + 1.0*L3*np.sin(q[2])*np.cos(q[0])*np.cos(q[1])
           + L4*(-1.0*np.sin(q[1])*np.sin(q[2])*np.cos(q[0])
           + 1.0*np.cos(q[0])*np.cos(q[1])*np.cos(q[2]))*np.sin(q[3])
           + L4*(1.0*np.sin(q[1])*np.cos(q[0])*np.cos(q[2])
           + 1.0*np.sin(q[2])*np.cos(q[0])*np.cos(q[1]))*np.cos(q[3]) - hx

    F[1] = 1.0*L1*np.sin(q[0])*np.cos(q[1]) + 1.0*L2*np.sin(q[0])*np.sin(q[1])
           + 1.0*L3*np.sin(q[0])*np.sin(q[1])*np.cos(q[2])
           + 1.0*L3*np.sin(q[0])*np.sin(q[2])*np.cos(q[1])
           + L4*(-1.0*np.sin(q[0])*np.sin(q[1])*np.sin(q[2])
           + 1.0*np.sin(q[0])*np.cos(q[1])*np.cos(q[2]))*np.sin(q[3])
           + L4*(1.0*np.sin(q[0])*np.sin(q[1])*np.cos(q[2])
           + 1.0*np.sin(q[0])*np.sin(q[2])*np.cos(q[1]))*np.cos(q[3]) - hy

    F[2] = 1.0*L0 + 1.0*L1*np.sin(q[1]) - 1.0*L2*np.cos(q[1])
           + 1.0*L3*np.sin(q[1])*np.sin(q[2]) - 1.0*L3*np.cos(q[1])*np.cos(q[2])
           + L4*(1.0*np.sin(q[1])*np.sin(q[2])
           - 1.0*np.cos(q[1])*np.cos(q[2]))*np.cos(q[3])
           + L4*(1.0*np.sin(q[1])*np.cos(q[2])
           + 1.0*np.sin(q[2])*np.cos(q[1]))*np.sin(q[3]) - hz
```

```
    F[3] = 0  # Ecuación adicional para igualar el tamaño

    return F

q = fsolve(brazo, q0, args=(h,))

thetha0, thetha1, thetha2, thetha3 = q

### Aplicar la función wrap_to_2pi a cada ángulo
thetha0 = wrap_to_2pi(thetha0)
thetha1 = wrap_to_2pi(thetha1)
thetha2 = wrap_to_2pi(thetha2)
thetha3 = wrap_to_2pi(thetha3)

print("thetha0 =", thetha0* (180 / np.pi), "grados")
print("thetha1 =", thetha1* (180 / np.pi), "grados")
print("thetha2 =", thetha2* (180 / np.pi), "grados")
print("thetha3 =", thetha3* (180 / np.pi), "grados")
```