Diseño e implementación de herramientas de software para el control inalámbrico del manipulador MaxArm

Lucía Jimena de la Rosa Ruano

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de herramientas de software para el control inalámbrico del manipulador MaxArm

Trabajo de graduación presentado por Lucía Jimena de la Rosa Ruano
para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Diseño e implementación de herramientas de software para el control inalámbrico del manipulador MaxArm

Trabajo de graduación presentado por Lucía Jimena de la Rosa Ruano
para optar al grado académico de Licenciada en Ingeniería Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

Este trabajo de graduación representa la culminación de los conocimientos que he adquirido durante los 5 años de la carrera. Haber aplicado lo aprendido en un tema de robótica le dio un valor añadido a mi educación, porque me dio consciencia de las diversas áreas a las que puede llegar todos mis conocimientos. Además, me llena de satisfacción poder ver los resultados de un proyecto al que le he dedicado tiempo y esfuerzo en elaborar. Llegar a este punto de la carrera ha sido posible gracias al apoyo de las personas que estuvieron presentes durante la elaboración de este trabajo y antes de él.

En primer lugar, me gustaría agradecer a mis padres, Leticia Ruano y Estuardo de la Rosa, por su continuo apoyo y motivación para poder estudiar y cumplir mis metas. También, les agradezco por su amor incondicional en todas mis decisiones y por ser figuras ejemplares que siempre me han guiado. Además, me gustaría agradecerle a mi hermana por estar siempre presente, por ser la persona que ha crecido a mi lado y se ha alegrado primero por mis logros. Así mismo, agradezco a mi familia por su amor y presencia en todas las metas que he completado en distintas fases de mi vida.

Agradezco a los catedráticos que he tenido en el transcurso de la carrera y he podido aprender de ellos por su vocación a la enseñanza. Le agradezco a mi profesor y asesor de tesis Miguel Zea, por el tiempo, consejos y explicaciones que hicieron posible la realización a de este proyecto. Así mismo, les deseo éxito en sus metas a mis compañeros, con quienes compartí distintas experiencias, pero sobre todo recuerdos memorables.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo principal de este proyecto de graduación fue diseñar e implementar herramientas de software para el control inalámbrico del brazo robótico MaxArm dentro del ecosistema Robotat. Los objetivos específicos incluyeron la obtención del modelo cinemático bajo el estándar Denavit-Hartenberg, la creación de un modelo de simulación representativo en el software Webots y el desarrollo de una interfaz en Matlab para controlar la cinemática en tiempo real de manera inalámbrica.

La metodología utilizada para este trabajo se dividió en tres etapas. La primera consistió en el modelado cinemático del MaxArm, donde se analizaron sus relaciones geométricas y ángulos de movimiento para generar una matriz D-H que permitió simular su comportamiento. En la segunda etapa, se desarrolló un modelo del manipulador en Webots, utilizando técnicas de simulación y diseño de eslabones con sus respectivas juntas. Finalmente, en la tercera etapa, se diseñó una interfaz gráfica en Matlab para planificar trayectorias, controlar posiciones y establecer una conexión inalámbrica mediante el protocolo TCP/IP.

Los resultados obtenidos incluyeron un modelo cinemático preciso que permitió simular y verificar el movimiento del MaxArm en Matlab, el cual fue validado mediante pruebas de posiciones y trayectorias. Además, se desarrolló un modelo del manipulador en Webots que reflejó su comportamiento físico. La interfaz gráfica implementada facilitó el control en tiempo real del manipulador, estableciendo una comunicación bidireccional con el ecosistema Robotat, lo que permitió capturar posiciones y generar trayectorias basadas en datos del entorno. La integración del MaxArm al ecosistema Robotat demostró la viabilidad de implementar métodos robóticos para tareas industriales y educativas.

Abstract

The main objective of this graduation project was to design and implement software tools for the wireless control of the MaxArm robotic arm within the Robotat ecosystem. The specific objectives included obtaining the kinematic model under the Denavit-Hartenberg standard, creating a representative simulation model in the Webots software, and developing an interface in Matlab to control the kinematics in real-time wirelessly.

The methodology used for this work was divided into three stages. The first stage involved the kinematic modeling of the MaxArm, analyzing its geometric relationships and motion angles to generate a D-H matrix that allowed its behavior to be simulated. In the second stage, a manipulator model was developed in Webots using simulation techniques and the design of links with their respective joints. Finally, in the third stage, a graphical interface was designed in Matlab to plan trajectories, control positions, and establish a wireless connection using the TCP/IP protocol.

The results obtained included an accurate kinematic model that allowed the MaxArm's movement to be simulated and verified in Matlab, validated through position and trajectory tests. Additionally, a manipulator model was developed in Webots, which reflected its physical behavior. The implemented graphical interface enabled real-time control of the manipulator, establishing bidirectional communication with the Robotat ecosystem, capturing positions, and generating trajectories based on environmental data. The integration of the MaxArm into the Robotat ecosystem demonstrated the feasibility of implementing robotic methods for industrial and educational tasks.

CAPÍTULO 1

Introducción

En la Universidad del Valle de Guatemala se han realizado varios avances en la implementación de manipuladores robóticos. Entre los ejemplos de los avances se encuentra el trabajo de graduación de *Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento* [1], el cual explica el funcionamiento cinemático y software de control del manipulador serial R17 dentro de un ecosistema de robótica. Además, se ha implementado el uso de software para el manipulador myCobot en clases de robótica, este siendo un brazo robótico más actual que el R17, con funcionalidades más avanzados.

Actualmente, se esta implementado el brazo robótico MaxArm en la carrera de ingeniería biomédica. En este trabajo de graduación se presenta el desarrollo de software para poder controlar de manera inalámbrica el manipulador mencionado. Para desarrollar este trabajo se enfoca en tres puntos principales: la obtención de la cinemática del robot, la generación de modelo de simulación del robot y la creación del software para controlarlo de manera bidireccional e inalámbrica que se pueda implementar dentro del ecosistema de robótica.

Para el primer punto se utiliza la documentación geométrica del manipulador para obtener su movimiento a partir de su posición final y los ángulos de los motores que posee. En el segundo punto se crea el modelo desde cero, y con la cinemática encontrada en el punto anterior, se realizan pruebas para verificar su movimiento. Finalmente, para el tercer punto se crea una conexión mediante WiFi que conecte una interfaz con funciones que controlen su cinemática a distancia. Adicionalmente, se utilizan estas funciones para comprobar que se pueda integrar dentro del Robotat, el ecosistema de robótica, capturando distintas posiciones.

CAPÍTULO 2

Antecedentes

Los brazos robóticos actualmente se utilizan para realizar tareas en la industria de manera eficiente. Se puede considerar al manipulador MaxArm como un brazo robótico de educación para comprender el manejo y el movimiento de los mismos. En la Universidad del Valle de Guatemala, anteriormente se han realizado proyectos de manipuladores seriales en ecosistemas robóticos que permiten verificar el movimiento de este tipo de brazos. Además, se han realizado proyectos de conexión inalámbrica con para el control de los mismos. La base de estos proyectos pueden llegar a la implementación del MaxArm dentro del ecosistema robótico y su comunicación inalámbrica.

2.1. Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

En este proyecto se tuvo como objetivo principal implementar una red WiFi en conjunto con el sistema de captura movimiento OptiTrack para formar un ecosistema de experimentación de robótica, denominado Robotat, que permite que distintos robots puedan trabajar en él de manera simultánea. Se instaló el hardware, que consistió en instalar una tarjeta de expansión de ethernet; la instalación de las seis cámaras en un patrón rectangular que permitiera una mejor cobertura de área donde se creara el ecosistema. Además, se instaló un router que iba a proveer una red WiFi para conectarse a los datos del Robotat. Adicionalmente, para obtener y enviar los datos pertinentes, crearon librerías que conectarán a la red y los datos.

Durante este proyecto se observaron alcances y limitaciones que fueron mejorando, los cuales se presentan a continuación:

- Calibración: al calibrar el hardware con el software, se utilizó el proceso de wanding, el cual consiste en mover puntos reflectivos dentro del rango de las cámaras del OptiTrack para coordinar el espacio del ecosistema. Se tomaron medidas manuales y con

el OptiTrack, demostrando que el margen de error es aceptable, para asegurar que los datos del OptiTrack son precisos.

- Hardware: se tomó en cuenta que la ubicación de las cámaras debía ser alta para capturar el ecosistema, y que para que la comunicación fuera efectiva se debía implementar un router que conectará a exclusivamente la red de datos del Robotat.
- Comunicación de la red: se crearon librerías exclusivas que obtuvieron los datos del Robotat, se midió que se podían conectar 11 agentes al mismo tiempo antes de que frecuencia de decodificación fuera menor que 10Hz, y se aconseja que se tomen menos de 10.000 muestras del movimiento para asegurar la asertividad de la captura de movimiento.
- Ampliación del alcance: se logró implementar una antena que permitiera conectar un cuerpo no robótico con los datos del Robotat, además de que se agregó el filtro Kalman que disminuyó el error en el uso de sensores.

En este proyecto también se buscaron futuras implementaciones como el uso de JSON para mandar datos en tiempo real, para poder ver si esto mejora la velocidad de decodificación de datos. En este proyecto se propusó una interfaz de envío de datos en tiempo real con Python que sigue un protocolo MQTT. Para este protocolo se utilizó en el microcontrolador ESP32 como Broker, debido a su capacidad de recibir y enviar datos por WiFi y Bluetooth. Además, se buscó que este broker mandara y recibiera datos de manera que se pudieran utilizar en otros lenguajes de programación, por lo cual se decidió por un formato de cáracter[2].

2.2. Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento

El trabajo tuvo como objetivo principal diseñar un conjunto de herramientas de software que permitieran controlar el manipulador serial R17 a partir del sistema de captura del OptiTrack. Para realizar el proyecto, se dividió en tres fases: la primera consistió en la prueba de montaje y toma de datos con el OptiTrack, la segunda consistió en el modelado de cinemática y el desarrollo de de software, y la tercera consistió en la integración del manipulador al Robotat. La segunda fase es la principal del trabajo debido que existió el desarrollo y diseño de un sistema embebido con un ESP32 que pudiera conectar y mandar datos al manipulador, y al mismo tiempo, se desarrollaron las librerías que permitieran obtener los datos y enviarlos. Adicionalmente se realizaron los análisis cinemática directa e inversa para poder realizar las trayectorias.

En este trabajo se pudieron destacar los siguientes puntos que fueron importantes en cuanto el desarrollo del trabajo:

- Se utilizaron las librerías creadas en el proyecto del Robotat[2] para poder enviar los datos del OptiTrack al ESP32.

- Para poder realizar la comunicación del sistema embebido se tuvieron que modificar los voltajes de un puerto serie RS-232 a niveles de voltajes TTL para poder comunicar el ESP32 con el R17.
- Se utilizó una interfaz de Matlab, para poder realizar una simulación de la cinemática directa, la cinemática inversa y las trayectorias.
- Se utilizó la configuración de la matriz de Denavit-Hartenberg para poder realizar la simulación de cinemática directa.
- Para la realización de de trayectorias se utilizó un conjunto de puntos que se operan como cinemática inversa.
- Se creó una librería específica para poder controlar el manipulador serial con el ESP32, que leyera y enviará los comandos.

En la fase final del proyecto, se comprobó cómo funcionaba la implementación del módulo de manera experimental a partir de posiciones meta teóricas. Se comprobó que el porcentaje de error máximo de las posiciones experimentales con las posiciones meta teóricas fue del 4 %. También para comprobar las trayectorias, se utilizó el software de Tracker[3], que les permitió grabar el funcionamiento y tomar una masa puntual como objetivo para analizar su movimiento; con esto se comprobó que las trayectorias tomadas por la grabación con las planificadas hacían sentido. También, se comprobó que el manipulador se comportaba de manera planificada en cualquier modo, utilizando las implementaciones realizadas[1].

2.3. Estudio e implementación de un sistema IoT en un brazo robot y control en TIA Portal

En este trabajo se tenía como objetivo principal implementar tecnología de internet de las cosas en un brazo robótico de Universal Robots para optimizar su uso mediante datos en tiempo real, visualización de indicadores en plataforma de nube y controlar el proceso mediante una pantalla HMI en TIA Portal de Siemens. Para poder realizar el proyecto primero se tuvo que determinar hardware, el cual se seleccionó el brazo UR3, la pantalla HMI TP700 Comfort y la pasarela inteligente SIMATIC IOT2040, que permitió la interconexión entre dispositivos. Luego se definieron los entornos de programación, el URSim, para programar el brazo robótico, TIA Portal, para programas el panel HMI, y el SIMATICIoT2040, este fue el paso que conecto los datos con la nube. Adicionalmente, utilizó el VMware Workstation Player, que permitió ejecutar en una plataforma virtual en Linux.

Este robot es uno de los más pequeños de Universal Robots, tiene una carga útil de 3kg y y un radio de alcance; se programa a través de los ángulos de sus uniones. Este proyecto se adaptó para realizarse de manera virtual antes de emplearse físicamente. Para este proyecto se consiguió crear una simulación digital del proyecto del brazo robot, que se implementara comunicación en tiempo real de los datos, y adicionalmente,se realizó un estudio económico y medioambiental. La limitación principal de este proyecto fue que solamente se realizó de manera virtual y no físicamente, y todos los resultados obtenidos son a partir de la simulación. El autor considera que con este tipo de implementación se puede optimizar

la eficiencia y adaptabilidad de los procesos industriales, además que la digitalización y automatización puede reducir el consumo de recursos y mejorar la sostenibilidad[4].

CAPÍTULO 3

Justificación

En la actualidad, ha aumentado el uso de brazos robóticos para realizar distintas tareas, en áreas industriales, médicas y educativas. Además, estos pueden trabajar de manera continua y precisa, optimizando el tiempo de trabajo en la industria y medicina. Esto se debe a la alta demanda de producción de productos y la necesidad de innovar en nuevas tecnologías que permitan realizar actividades de manera rápida y segura para las personas. Guatemala es uno de los países con menos avances en esta tecnología, por lo tanto, se tiene el objetivo de fomentar conocimiento tecnológico mediante la educación y realización de proyectos. Debido a esto es importante seguir innovando en este ámbito.

En la Universidad del Valle de Guatemala, se han logrado avances robóticos que han atribuido a la educación y capacitación de las personas para que puedan manejar las nuevas tecnologías que se buscan implementar. Se ha trabajado con un ecosistema que permite hacer pruebas de robots con un sistema de captura de movimiento. Además, de que se han realizado proyectos de manipuladores seriales que funcionan de manera inalámbrica. Debido a esto, se puede considerar que se poseen los instrumentos e instalaciones necesarias para poder seguir desarrollando en la innovación de brazos robóticos. Además, las carreras tecnológicas contienen los estudios necesarios para poder desarrollar este tipo de proyectos.

Actualmente, la Universidad del Valle de Guatemala, adquirió un conjunto de manipuladores MaxArm con tecnología más reciente que la utilizada en proyectos de manipuladores anteriores, y se busca implementarlos para poder seguir innovando en la educación. Poder conocer la cinemática de su movimiento y controlarla de manera completa e inalámbrica permitará que se pueda utilizar con mayor alcance. El uso de implementaciones robóticas a distancia ayudan que se puedan realizar tareas con más comodidad y seguridad. Este proyecto se llevará a cabo con la integración de software que permitirá enviar datos desde Matlab al manipulador, además se incluirá el uso de simulaciones para garantizar que se conoce la cinemática del brazo robótico.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar e implementar herramientas de software que permitan controlar el movimiento del brazo robótico MaxArm y comunicar sus datos de manera inalámbrica dentro del ecosistema Robotat.

4.2. Objetivos específicos

- Obtener el modelo cinemático del manipulador MaxArm bajo el estándar de Denavit-Hatenberg, que permita su simulación en software.
- Crear un modelo del manipulador para su simulación dentro Webots, que sea representativo de su funcionamiento.
- Implementar una interfaz en Matlab que permita controlar la cinemática del manipulador en tiempo real y planificar trayectorias, comunicando datos de manera bidireccional de forma inalámbrica.

CAPÍTULO 5

Alcance

Este trabajo de graduación se divide en tres bloques principales. El primero se enfocó en comprender la cinemática de un brazo robótico que combina características seriales y paralelas, lo que presentó un desafío adicional para el modelado bajo la convención Denavit-Hartenberg. Para ello, fue necesario analizar sus relaciones geométricas y determinar los ángulos que configuran las posiciones del efector final. El segundo bloque se centró en la creación de un modelo de simulación representativo y visualmente similar, implementado en Webots. Cabe mencionar que este programa no incluye un modelo predefinido para este robot específico, por lo que se desarrolló desde cero basándose en investigación propia y en las especificaciones técnicas del proveedor.

El bloque más relevante fue el tercero, que abarcó el desarrollo de librerías de programación y una interfaz gráfica en Matlab para permitir la conexión inalámbrica y el control del manipulador. Esta etapa requirió modificaciones al software del proveedor para adaptarlo a los requerimientos específicos de la comunicación bidireccional mediante protocolo TCP/IP. Adicionalmente, la interfaz gráfica permitió no solo el control del robot, sino también la visualización en tiempo real de las trayectorias planificadas, lo que facilitó la validación experimental de las funcionalidades implementadas.

Es importante destacar que este trabajo consideró como limitación la complejidad adicional derivada del diseño híbrido del manipulador, lo cual dificultó la aplicación directa de métodos de modelado estándar. Además, se evaluó la factibilidad de integrar las herramientas desarrolladas con el ecosistema Robotat, logrando una comunicación bidireccional funcional que permitió realizar pruebas basadas en datos del sistema de captura de movimiento, estableciendo un punto de partida sólido para futuras investigaciones y aplicaciones.

CAPÍTULO 6

Marco teórico

Para este proyecto es importante mencionar las especificaciones del robot, principalmente las utilidades de su microcontrolador y lenguajes de programación. Así mismo, se debe profundizar en los métodos de cinemática que se utilizarán para el control del manipulador y el protocolo de comunicación para enviar los datos de está cinemática. Por el lado de la simulación, se debe describir el programa a utilizar, Webots, y la metodología para implementar el modelo dentro de este.

6.1. Especificaciones del robot

MaxArm es un brazo robótico de código abierto que funciona con el mircrocontrolador ESP32. Utiliza la tecnología de cinemática inversa para ejecutar una variedad tareas. Puede utilizar Python y Arduino como lenguaje de programación con comunicaciones a través de Bluetooth e Internet. El efector final de este brazo puede moverse en los ejes X, Y y Z. Además, puede calcular el ángulo de cada servomotor que posee para su movimiento a partir de una posición cartesiana, porque incluye un análisis de cinemática inversa y directa con los estándares de Denavit Hatenberg. Adicionalmente, se puede conectar con una variedad de sensores que le permiten realizar más tareas [5].

Elemento	Especificación
Dimensión del producto	158 x 160 x 260mm (largo x ancho x alto)
Peso	1.3 kg
Material	Metal y fibra de vidrio
Grados de libertad	4GDL
Alimentación eléctrica	12V 5A Adaptador CC
Controlador del sistema	ESP32: controlador de código abierto
Software	PC software, iOS/ Android APP
Método de comunicación	WiFi y Bluetooth
Servomotor	HTS-35H bus servo y LFD-01M micro servo
Método de control	Computadora/ Télefono inteligente/ Conexión inalámbrica/ Control con Mouse

Cuadro 1: Especificaciones del manipulador MaxArm [5].

Figura 1: Manipulador MaxArm

6.1.1. Lenguajes de programación

El manipulador MaxArm utiliza tanto Python como Arduino como entorno de programación. Python realiza a tareas de alto nivel, como el procesamiento de datos y la implementación de algoritmos complejos por las bibliotecas específicas, como las de visión artificial. Por otro lado, Arduino se emplea para el control en tiempo real y la interacción directa con los periféricos del hardware, como servomotores y sensores, aprovechando su capacidad de respuesta y su integración con la parte electrónica. Ambos lenguajes, facilitan la experimentación como el desarrollo avanzado, permitiendo a los usuarios explorar un amplio espectro de aplicaciones robóticas de control y flexibilidad del brazo robótico [5].

6.1.2. Microcontrolador ESP32

El controlador de código abierto ESP32 utilizado en el MaxArm consiste en una placa base con núcleo ESP32 y una placa de extensión multifuncional. Además de incluir las interfaces para los servomotores, también está equipado con una alarma, LED, interfaces USB y otros componentes electrónicos. Se le agregaron múltiples interfaces de expansión para que los usuarios puedan conectar directamente otros sensores y módulos de ejecución para el desarrollo secundario. Posee soporte para WiFi y Bluetooth, por lo que la placa base con núcleo ESP32 es conveniente para que los usuarios desarrollen aplicaciones de transmisión de datos inalámbricos [5].

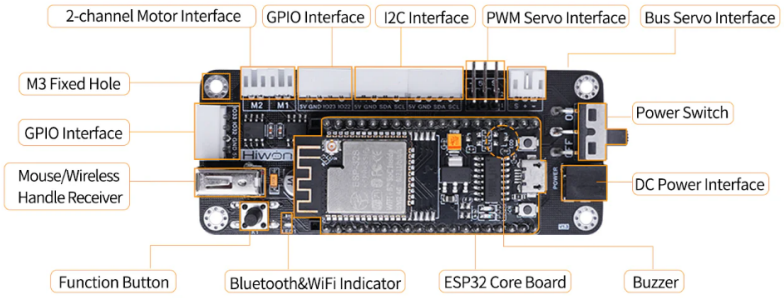

Figura 2: ESP32 de código abierto implementado en el MaxArm.

El ESP32 es un chip de con Wi-Fi y Bluetooth de 2.4 GHz diseñado con tecnología TSMC de bajo consumo de energía de 40 nm. Está diseñado para lograr el mejor rendimiento de energía y RF, mostrando robustez, versatilidad y confiabilidad en una amplia variedad de aplicaciones y escenarios de energía. Posee las siguientes características:

- Wi-Fi: Soporta desde 802.11b/g/ hasta 150 Mbps, posee modos de operación simultáneos como Estación, *SoftAP* y *Promiscuo*. Además, posee *Immediate Block ACK* para mejorar la transmisión de datos por internet
- Bluetooth: cumple con las especificaciones Bluetooth v4.2 BR/EDR y LE, posee potencia de transmisión de +9 dBm y sensibilidad de recepción de -94 dBm para Bluetooth LE.
- CPU y memoria: incluye uno o dos procesadores micro LX6 de 32 bits.
- Gestión de energía: ofrece control de potencia de alta resolución y cinco modos de energía para escenarios típicos, con un consumo en modo *Deep-sleep* de solo 10 µA [6].

El manipulador utiliza un ESP32-WROOM-32 como su módulo de control principal. Esta es una versión comúnmente utilizada del ESP32 que ofrece una combinación de potencia de procesamiento y conectividad inalámbrica para WiFi y robótica.

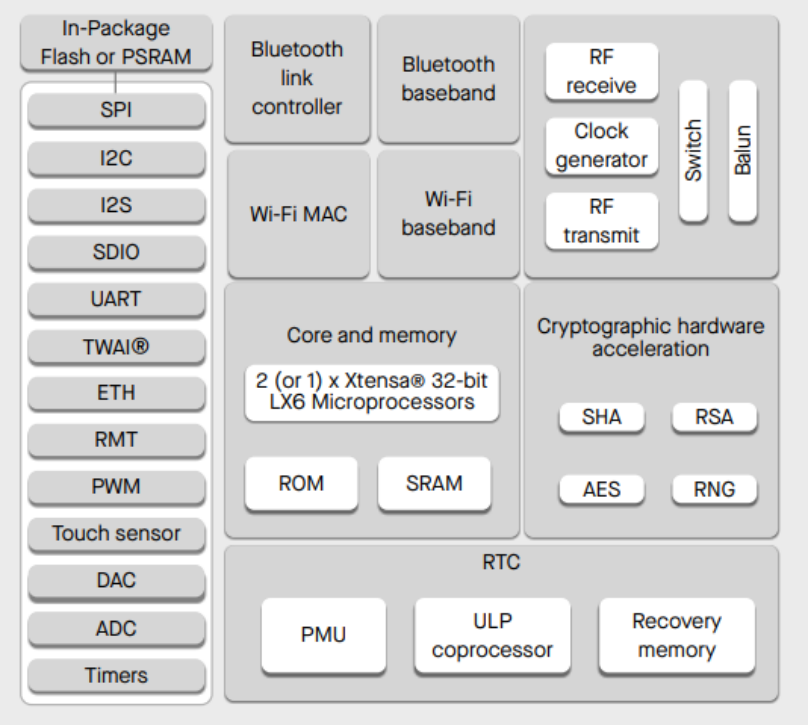

Figura 3: Bloque funcional de la base del ESP32.
[6]

6.2. Cinématica del robot

La cinemática del robot describe el movimiento del mismo con respecto a un sistema de referencia sin considerar las fuerzas que intervienen. Existen dos métodos fundamentales a resolver en la cinemática del robot. El primero es la cinemática directa, y consiste en determinar cuál es la posición y orientación del extremo final del robot con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot. El segundo método es la cinemática inversa, la cual resuelve la configuración, es decir, obtiene los ángulos de los motores que debe adoptar el robot para una posición y orientación del extremo conocidas.

Para la cinemática directa, Denavit y Hartenberg propusieron un método sistemático para representar la geometría espacial de los elementos de una cadena cinemática de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base. Por otra parte, la cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo. Esta relación viene dada por el modelo diferencial expresado mediante la matriz Jacobiana.

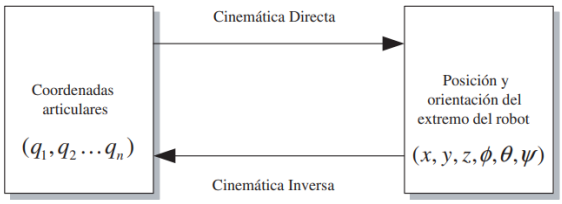

Figura 4: Relación de la cinemática directa con la inversa
[7]

6.2.1. Cinématica directa

La cinemática directa permite conocer cuál es la posición y orientación que adopta el extremo del robot cuando cada una de las variables que fijan la posición u orientación de sus articulaciones toma valores determinados. La obtención del modelo cinemático directo puede ser abordado mediante dos enfoques diferentes denominados métodos geométricos y métodos basados en cambios de sistemas de referencia. Los primeros son adecuados para casos simples, pero al no ser sistemáticos, su aplicación queda limitada a robots con pocos grados de libertad. Los métodos basados en cambio de sistemas de referencia, permiten de una manera sistemática abordar la obtención del modelo cinemático directo del robot para robots de n grados de libertad, siendo éstos los que usan las matrices de transformación homogénea y el módelo de Denavit Hartenberg.

6.2.2. Algoritmo de Denavit Hartenberg para cinémtica directa

Para describir la relación que existe entre dos elementos contiguos se suele utilizar en robótica es (la representación de Denavit-Hartenberg. Denavit y Hartenberg propusieron en 1955 un método matricial que establece la localización que debe tomar cada sistema de coordenadas ligado a cada eslabón i de una cadena articulada, para poder sistematizar la obtención de las ecuaciones cinemáticas de la cadena completa. Según la representación propuesta por D-H, es posible pasar de un eslabón al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. Estas 4 transformaciones básicas consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i–1 con el sistema del elemento i. Las transformaciones en cuestión son las siguientes:

1. Rotación alrededor del eje z un ángulo θ.
2. Traslación a lo largo del eje z, una distancia d.
3. Traslación a lo largo del eje x, una distancia a.
4. Rotación alrededor del eje x un ángulo α.

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

$$A = Rotz(\theta)T(0,0,d)T(a,0,0)Rotx(\alpha) \tag{1}$$

Para que la matriz A relacione los sistemas es necesario que los sistemas se hayan escogido de acuerdo a la definición de los 4 parámetros de Denavit Hartenberg, conforman el siguiente algoritmo para la resolución del problema cinemático directo:

1. Numerar los eslabones comenzando con 1,primer eslabón móvil de la cadena, y acabando con n ,último eslabón móvil, Se enumerará como eslabón 0 a la base fija del robot.
2. Numerar cada articulación comenzando por 1, la correspondiente al primer grado de libertad, y acabando en n.
3. Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
4. Para i de 0 a n_1 situar el eje z_1 sobre el eje de la articulación i_1.
5. Situar el origen del sistema de la base en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0.
6. Para i de 1 a n_1, situar el origen del sistema en la intersección del eje z_i con la línea normal común a $z_{\text{i–1}}$ y z_i . Si ambos ejes se cortan se situaría en el punto de corte. Si fueran paralelos se situaría en la articulación i_1.
7. Situar x_i en la línea normal común a $z_{\text{i–1}}$ y z_i .
8. Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.
9. Situar el sistema en el extremo del robot de modo que z_n coincida con la dirección de $z_{\text{n–1}}$ y x_n sea normal a $z_{\text{n–1}}$ y z_n.
10. Obtener θ_i como el ángulo que hay que girar en torno a $z_{\text{i–1}}$ para que $x_{\text{n–1}}$ y z_n queden paralelos.
11. Obtener d_i como la distancia, medida a lo largo de $z_{\text{i–1}}$, que habría que desplazar para que z_i y $x_{\text{i–1}}$ queden alineados.
12. Obtener a_i como la distancia medida a lo largo de x_i ,que coincide con $x_{\text{i–1}}$, que habría que desplazar para que su origen coincida.
13. Obtener α_i como el ángulo que habría que girar en torno a x_i, para que el eje coincida totalmente.
14. Obtener las matrices de transformación A definidas en (1).
15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot:
$$T = {}^0\mathrm{A}_1 \cdot {}^1\mathrm{A}_2 \cdot {}^{n-1}\mathrm{A}_n. \tag{2}$$
16. La matriz T define la orientación y posición del extremo referido a la base, en función de las n coordenadas articulares.

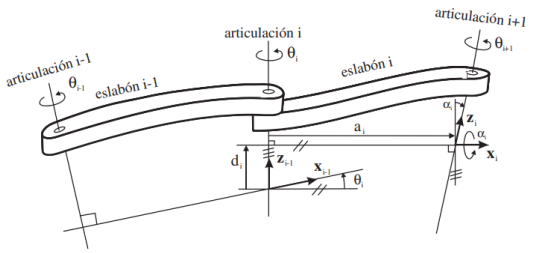

Figura 5: Parámetros de DH para eslabones
[7]

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

6.2.3. Cinemática inversa

El objetivo de la cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $q = [q_1, q_2, ..., q_i]^T$ para que su extremo se posicione y oriente según una determinada localización espacial. Es posible establecer ciertas características generales que permitan plantear y resolver el problema cinemático inverso de una manera sistemática. Los métodos geométricos permiten obtener los valores de las variables articulares, que son las que consiguen posicionar el robot. Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot. Como alternativa para resolver el mismo problema se puede recurrir a manipular directamente las ecuaciones correspondientes al problema cinemático directo, que establece la siguiente relación:

$$\begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = \left[t_{ij}\right], \tag{3}$$

donde los elementos t_{ij} son función de las coordenadas articulares $q = [q_1, q_2, ..., q_i]^T$, y se puedan despejar las n variables articulares q_i en función de las componentes de los vectores n, o, a y p. p representa al vector de dirección mientras, n, o y a representan a la matriz de orientación.

El primer paso para resolver el problema cinemático inverso es obtener la expresión (3) correspondiente al robot. Es decir, obtener la matriz T que relaciona el sistema de referencia asociado a la base con el sistema de referencia asociado a su extremo de trabajo. A partir de éstos es inmediato obtener las matrices A (1) y la matriz T (2). Obtenida la expresión de T en función de las coordenadas articulares $(q_1, q_2, ..., q_i)$, y supuesta una localización de destino para el extremo del robot definida por los vectores n, o, a y p se podría manipular directamente las ecuaciones resultantes de T a fin de despejar $q_1, q_2, ..., q_i$ en función de n, o, a y p. Los procedimientos anteriores permiten obtener los valores de las 3 primeras variables articulares del robot, aquellas que posicionan su extremo en unas coordenadas (p_x, p_y, p_z). Por el otro lado, no es suficiente con posicionar el extremo del robot en un punto del espacio, sino que es preciso también conseguir que la herramienta se oriente de una manera determinada. Para ello, los robots cuentan con otros tres grados de libertad, situados al final de la cadena cinemática y cuyos ejes se cortan en un punto.

El método de desacoplo cinemático es aplicable a aquellos robots cuyos tres últimos grados de libertad se cortan en un punto, separando los problemas de obtención del modelo cinemático inverso de posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las coordenadas del punto de corte de los 3 últimos ejes calculándose los valores de las tres primeras variables articulares que consiguen posicionar este punto.

6.2.4. Cinématica diferencial

El modelado cinemático de un robot busca las relaciones entre las variables articulares y la posición y orientación del extremo del robot. A la cinemática del robot le incumbe conocer la relación entre las velocidades de las coordenadas articulares y las de la posición y orientación del extremo, o el efecto que un movimiento diferencial de las variables articulares tiene sobre las variables en el espacio de la tarea. Esta relación queda definida por el modelo diferencial. Mediante él, el sistema de control del robot puede establecer qué velocidades debe tener a cada articulación para conseguir que el extremo desarrolle una trayectoria temporal concreta.

El modelo diferencial queda concretado en la denominada matriz Jacobiana. En general la matriz Jacobiana de un robot, relaciona el vector de velocidades articulares $(\dot{q_1}, \dot{q_2}, \dot{q_n})$ con otro vector de velocidades expresado en un espacio distinto. Primero se puede considerar la relación con las velocidades de la localización del extremo del robot, siendo ésta la posición y orientación expresada en base a sus coordenadas cartesianas y a sus ángulos de Euler. Esta relación viene dada por la denominada Jacobiana analítica del manipulador.

Una segunda opción es relacionar las velocidades articulares, con los vectores de velocidad linear y angular con que se mueve el extremo del robot, expresados en un sistema de referencia determinado, por ejemplo el del origen. La relación entre ambas velocidades se obtiene a través de la denominada matriz Jacobiana geométrica o simplemente Jacobiana del manipulador. En ambos casos, la matriz Jacobiana directa permite conocer una expresión de las velocidades del extremo del robot a partir de los valores de las velocidades de cada articulación. Por el otro lado, la matriz Jacobiana inversa permitirá conocer las velocidades articulares necesarias para obtener un vector concreto de velocidades del extremo [7].

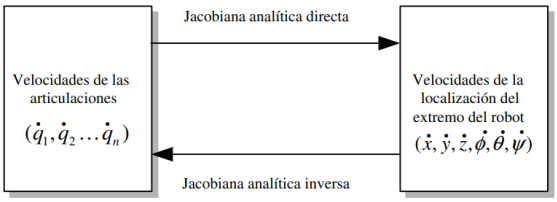

Figura 6: Jacobiana analítica directa e inversa
[7]

6.3. Comunicación de protocolo TCP

El protocolo TCP, Transmission Control Protocol, se define como un protocolo fiable y orientado a un flujo de byte. Utiliza los servicios de IP para su transporte por Internet, siendo un protocolo orientado a conexión. Esto significa que las dos aplicaciones envueltas en la comunicación, usualmente un cliente y un servidor, deben establecer previamente una comunicación antes de poder intercambiar datos. Se considera que este es un protocolo confiable por las siguientes razones:

1. Los datos a enviar son reagrupados por el protocolo en porciones denominadas segmentos. El tamaño de estos lo asigna el propio protocolo.
2. Cuando en una conexión TCP se recibe un segmento completo, el receptor envía una respuesta de confirmación o Acknowledge al emisor confirmando el número de bytes correctos recibidos.
3. Cuando se envía un segmento se inicializa un temporizadoro timer. De esta forma, si en un determinado plazo de tiempo no se recibe una confirmación o Acknowlegde de los datos enviados, estos se retransmiten.
4. TCP incorpora un checksum para comprobar la validez de los datos recibidos. Si se recibe un segmento erróneo, no se envía una confirmación. De esta forma, el emisor retransmite los datos otra vez.
5. El protocolo TCP utiliza unos números de secuencia para asegurar la recepción en orden, evitando cambios de orden o duplicidades de los bytes recibidos.
6. TCP es un protocolo que implementa un control de flujo de datos. En el envío de datos se puede ajustar la cantidad de datos enviada en cada segmento, evitando colapsar al receptor.

La cabecera del segmento TCP es bastante compleja debido a que la comunicación es más elaborada y debe proporcionar fiabilidad. Esto implica una serie de información adicional que debe mantenerse para poder conocer el estado de la comunicación en cualquier momento.

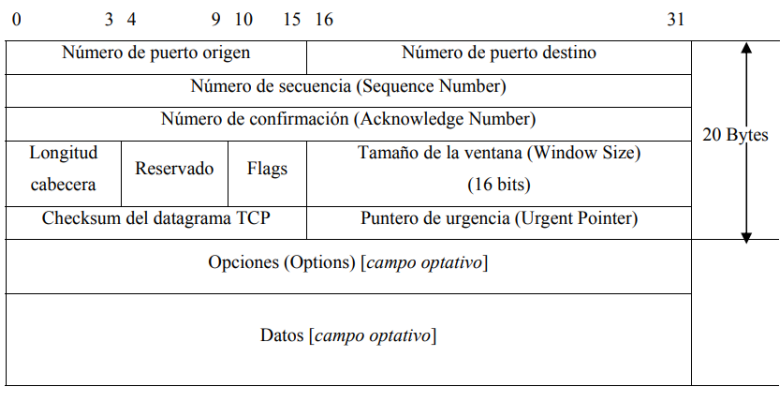

Figura 7: Cabecera de un segmento TCP

La cabecera TCP incluye las siguientes características para funcioar:

- El número de puerto origen y número de puerto destino, sirven para diferenciar una comunicación en un ordenador de las demás.
- El número de secuencia o Sequence Number, identifica el byte concreto del flujo de datos que actualmente se envía del emisor al receptor. De esta forma, TCP numera los bytes de la comunicación de una forma consecutiva a partir del número de secuencia inicial.
- El número de confirmación o Acknowledge Number, es el número de secuencia más uno. De este modo se especifica al emisor que los datos enviados hasta este número de secuencia menos uno son correctos.
- La longitud de la cabecera o header Length, especifica en palabras de 32 bits, 4 bytes, el tamaño de la cabecera del segmento TCP incluyendo las posibles opciones.
- Las banderas o flags, son las encargadas de especificar los diferentes estados de la comunicación. También, validan los valores de los distintos campos de la cabecera de control. Puede haber simultáneamente varios flags activados.
- El tamaño de la ventana o Window Size, es el número de bytes desde el número especificado en el campo de confirmación, que el receptor está dispuesto a aceptar.
- El checksum del segmento TCP tiene la función de controlar los posibles errores que se produzcan en la transmisión. Este checksum engloba la cabecera TCP y los datos. En caso de error, el segmento queda descartado y el propio protocolo es el encargado de asegurar la retransmisión de los segmentos erróneos o perdidos.
- El puntero de urgencia o Urgent Poiner, es válido sólo si el flag de URG se encuentra activado. Consiste en un valor positivo que se debe sumar al número de secuencia especificando una posición adelantada dónde podemos enviar datos urgentes.
- Las opciones u Options permiten especificar de forma opcional características extras a la comunicación.
- Los datos o Data son opcionales. Esto significa que se pueden enviar simplemente cabeceras TCP con diferentes opciones. Esta característica se utiliza al iniciar la comunicación o en el envío de confirmaciones [8].

6.4. Webots: open source robot simulator

Webots es una aplicación de escritorio de código abierto y multiplataforma que se utiliza para simular robots. Proporciona un entorno de desarrollo completo para modelar, programar y simular robots. Se caracteriza por los siguientes puntos:

- Permite diseñar fácilmente simulaciones robóticas completas utilizando la biblioteca de activos, que incluye robots, sensores, actuadores, objetos y materiales. Además, se pueden importar sus modelos CAD existentes.

- Se puede crear una amplia variedad de simulaciones que incluyen robots de dos ruedas, brazos industriales, robots con patas, robots modulares, automóviles, drones voladores, vehículos submarinos autónomos, robots con orugas, vehículos aeroespaciales, etc.
- El núcleo de Webots se basa en la combinación de una interfaz gráfica de usuario moderna, un motor de física y un motor de renderizado OpenGL 3.3, que funciona en Windows, Linux y macOS.
- El robot se puede programar en C, C++, Python, Java, MATLAB o ROS con una API simple que cubre todas las necesidades básicas de robótica [9].

6.4.1. Metodología para crear robots seriales

En Webots se pueden crear robots que sean representativos de los modelos reales y que puedan mostrar la cinemática de los mismos dentro de un ambiente de simulación:

1. Crear un nuevo proyecto con File / New / New Project Directory, se nombra el directorio del proyecto y del nuevo mundo o World.
2. Selecciona todas las opciones de la cajas, incluyendo el Add a rectangle arena.
3. A partir de este punto, se puede ver el árbol de escena o Scene Tree que contiene todos los objetos o Nodes en la simulación, se selecciona uno de los elementos del arbol principal y con click dererecho se despliaga un menú del que se selecciona Add new.
4. Se verá un nuevo menú en la pantalla del cual se selecciona "Base nodesz dentro de esta opción se selecciona la opción de Robot.
5. Cuando se tenga la opción de Robot en el Scene Tree, se despliega el menú de esta opción y se verá la opción de Children, sobre la que se debe hacer click derecho.
6. En el menú que se despliegue se selecciona Base nodes, de este punto en adelante, se trabaja con las opciones de Transform y HingeJoint. Transform se utiliza para crear los eslabones, mientras HingeJoint se utiliza para crear los ejes que harán que estos eslabones giren.
7. Primero, se trabaja con la opción de Transform para crear la base del robot, en la que luego se creará la junta giratoria. Dentro de la opción de Trasform se despliega un menú en el que se tendrá la opción de Children, a la cual se le debe dar click derecho.
8. Al tener el menú que despliega Children, se selecciona Base Nodes y dentro de está opción se selecciona Shape.
9. Dentro del Children de Transform se tendrá la opción de Shape, la cual despliega las características de geometry y appearence. Al seleccionar geometry en el menú de Base nodes se encontraran diferentes figuras en 3D para realizar la base y al seleccionar appearence se puede seleccionar el material y color de la pieza.
10. Al tener la pieza en la opción de Translation esta se puede ubicar en la posición x,y,z que se necesite.

11. Cuando se tenga la base, se repiten los pasos 5 y 6, pero se selecciona HingeJoint, al desplegar este menu se verán las opciones de device, endPoint y jointParameters.

12. Primero, haciendo doble click en jointParameters se selecciona HingeJointParameters, este paso es importante para la junta porque dentro del menú de esta opción se tiene la opción de anchor que permite anclar el eje de giro en base a la base del robot o en base a una junta previa, en está opción se debe escribir la posición x,y,z con base a la referencia que se tome. También se tiene la opción de axis en la cual se debe escribir un 1 en el eje que quiere que se gire la pieza.

13. Luego, se da click derecho a device y se selecciona la opción de RotacionalMotor para que funcione como una junta giratoria. Además, se debe expandir el menú de RotacionalMotor y en la opción de name darle un nombre único que se utilizará para la programción del controlador del robot para girar el eje.

14. Finalmente, en endPoint se da click derecho en la opción de Children, al desplegarse el menú se selecciona Base nodes y se selecciona solid.

15. Al tener solid en el menú del endPoint se despliega el menú se da click derecho en la opción de Children, y se repiten los pasos 8 a 10. Esta pieza es la que girará respecto al RotacionalMotor que se eligió previamente y representará el eslabón que se mueve.

16. A partir de este momento, se puede crear una programación de controlador agregándola a la opción de Controller dentro del menú de la opción de Robot para verificar que el punto de anclaje de giro y el eje sean los correctos.

17. Luego se pueden seguir creando más eslabones en cadena creando otra HingeJoint dentro del endPoint del eslabón previo, repitinedo los pasos el 11 al 16 hasta tener todos los eslabones de la cadena.

CAPÍTULO 7

Modelado cinemático y desarrollo de software para controlar el manipulador serial MaxArm

Con base en la documentación de cinemática proporcionada junto al manipulador serial, se obtuvieron las ecuaciones y relaciones que describen su movimiento tanto en cinemática inversa como directa. Además, se midieron los eslabones del manipulador y se compararon con los datos proporcionados por el fabricante para garantizar la precisión del modelo. También se identificó el sentido de los ejes utilizado por el proveedor para determinar las posiciones de los eslabones en el plano. Con esta información, se desarrolló una simulación del modelo en Matlab, siguiendo el estándar de Denavit-Hartenberg, y se verificó su precisión mediante la comparación de la cinemática de posiciones de la interfaz del proveedor con el modelo implementado, asegurando la validez de las relaciones obtenidas.

El brazo robótico MaxArm posee tres motores que posicionan su herramienta final y un motor adicional que rota esta misma. El proveedor de este manipulador proporcionó documentación inicial de la relaciones geométricas que enlazan los ángulos de los motores con las posiciones finales de la herramienta, con lo cual se obtuvieron las ecuaciones de cinemática del manipulador. Así mismo, se proporcionaron las medidas de los eslabones utilizadas. Esta información es importante debido a que los motores no se encuentran posicionados directamente en las juntas sino que poseen conexiones paralelas a los eslabones del manipulador para generar las posiciones. La Figura 8 muestra las relaciones de ángulos y geometrías de los eslabones que extienden y encogen el brazo.

El desarrollo de software se llevó a cabo dentro de Matlab y Arduino. En Matlab, se implementaron todas las funciones relacionadas al cálculo de la cinemática directa e inversa para controlar posiciones y trayectorias, así como la interfaz para simplificar el uso de estas librerías. El software en Arduino se enfoca en la recepción de todo estos datos de control de cinemática y al mismo tiempo, implementarlos para el movimiento del manipulador. Además, se realizó un software de conexión entre ambos lenguajes, es decir el envió de datos

desde Matlab hacia el software en Arduino.

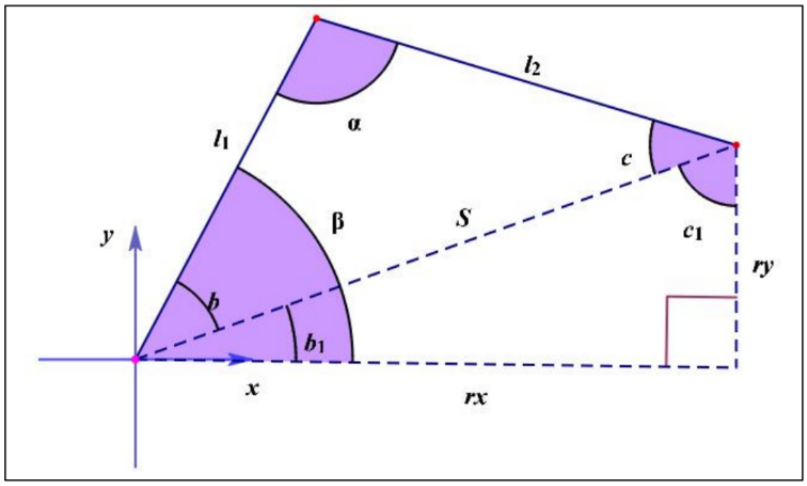

Figura 8: Relaciones geométricas del manipulador MaxArm.

7.1. Modelado y representación DH

Aunque se poseen las relaciones geométricas de los motores con las posiciones finales de la herramienta, está no proveía cuales eran las relaciones directas de los motores con los ángulos de las juntas, por lo que con pruebas de posiciones y ángulos de motores se obtuvieron estas relaciones. Inicialmente, se consideró que la posición de restablecimiento en la documentación del proveedor es cuando la posición del efector final se encuentra en $x : -1$, $y : -161$ y $z : 209$, y los ángulos $motor1 : 120^o$, $motor2 : 89.15^o$ y $motor3 : 1.57^o$. Para esta posición se consideró que las juntas estaban en su ángulo inicial o cero, es decir, β y α, en 8, son cero para esta posición. Además, se consideró el funcionamiento de los ejes de movimiento, siendo el movimiento del extensión hacia el eje y negativo y el movimiento a la izquierda hacia el eje x negativo. Con estas relaciones se obtuvieron las relaciones de los ángulos de las juntas con los ángulos de los motores.

A partir de estas relaciones se obtuvo la matriz de Denavit-Hartengberg, que representa la cinemática del brazo robótico respecto a cada una de sus juntas. Considerando que el manipulador posee 5 juntas y los ángulos de los motores que se utilizaron son $q = [q_1, q_2, q_3, q_4]$, donde q_1 es el ángulo que gira alrededor de z, q_2 y q_3 son los ángulos que provocan el movimiento en x y y, y q_4 es el ángulo de la herramienta. Además, se obtuvieron las siguientes relaciones:

$$
\begin{aligned}
q_{\theta_1} &= -q_1 - \frac{\pi}{2}, \\
q_{\theta_2} &= \frac{\pi}{2} - q_2, \\
q_{\theta_3} &= \frac{\pi}{2} - q_2 + q_3, \\
q_{\theta_4} &= q_3, \\
q_{\theta_5} &= q_4,
\end{aligned}
\tag{4}
$$

Las relaciones previas muestran el ángulo de cada junta del manipulador, a partir de los 4 ángulos de los motores que posee. Se debió realizar ese procedimiento previo por las relaciones paralelas que poseen los eslabones dependiendo de los ángulos de los motores. Con los ángulos de las juntas de los eslabones del manipualdor, se pudo obtener la matriz bajo el estándar de DH :

θ **(rad)**	**d (mm)**	**a (mm)**	α **(rad)**
q_{θ_1}	84.4	8.14	0
q_{θ_2}	0	128.4	$\frac{3\pi}{2}$
q_{θ_3}	0	138	$\frac{3\pi}{2}$
q_{θ_4}	0	16.8	$-\frac{\pi}{2}$
q_{θ_5}	0	0	π

Cuadro 2: Matriz de DH del manipulador MaxArm.

En la Figura 2, θ se refiere al angulo de giro en en el eje z, d se refiere a longitudes de eslabones en el eje z respecto al eslabón previo, a las posiciones en el eje x respecto al eslabón previo y α se refiere al angulo de compensación del eje z respecto al eje x.

Para poder comprobar que esta matriz era funcional se realizaron simulaciones gráficas con el *toolbox* de Peter Corke en Matlab. A continuación, se muestran las gráficas realizadas con el fin de compararlas con el modelo físico, visualizando sus comportamientos parecidos respecto a las relaciones geométricas empleadas.

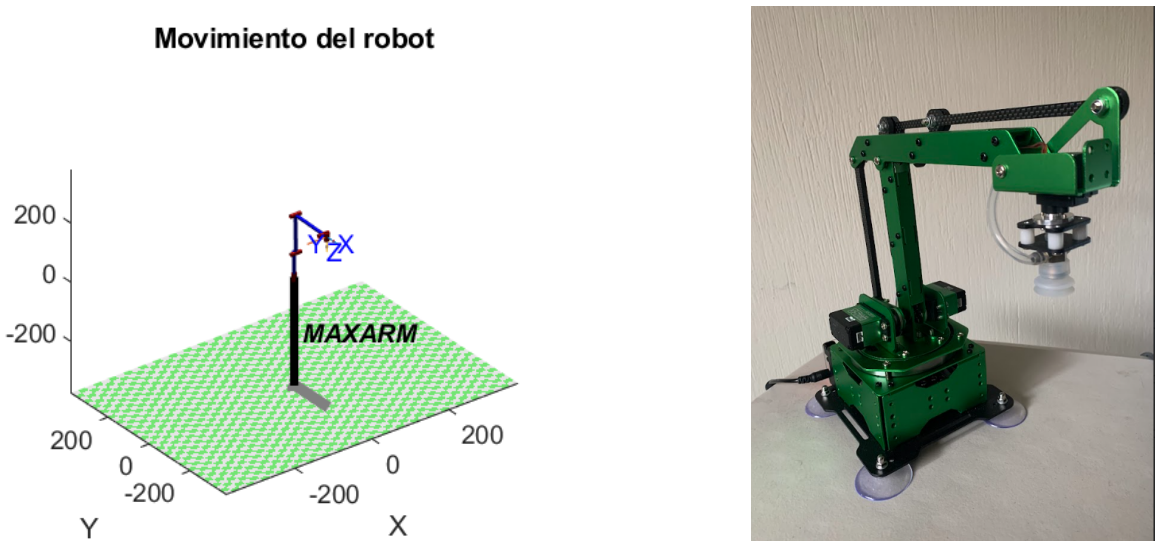

Figura 9: Prueba de posición con $q = [0, 0, 0, 0, 0]$ en las juntas

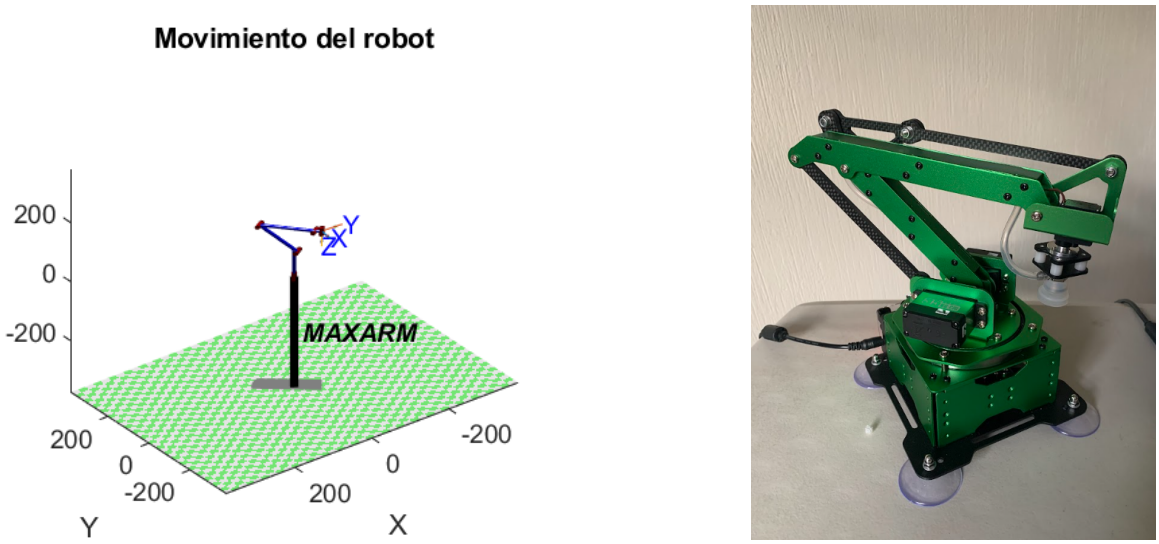

Figura 10: Prueba de posición con $q = [-60.2, 46.9, 54.0, 7.1, 45.0]$ en las juntas

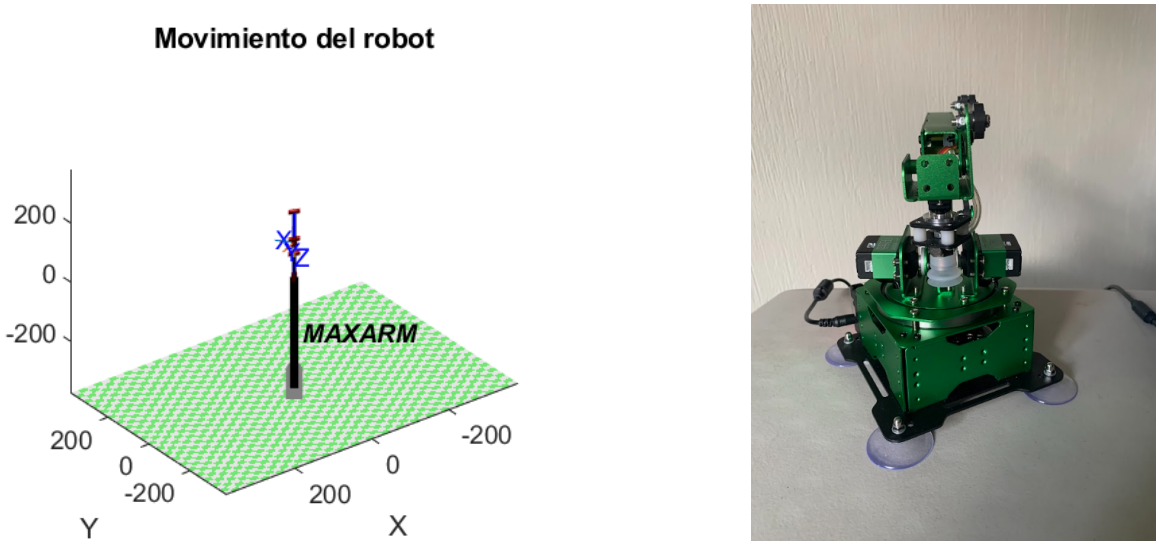

Figura 11: Prueba de posición con $q = [28.3, 53.9, 56.0, 2.1, 120.0]$ en las juntas

7.2. Implementación de cinemática directa

El manipulador MaxArm cuenta con documentación que describe la cinemática inversa que permite utilizar los tres ángulos de los motores de posición para conseguir una posición deseada. Cabe destacar que los tres motores de posición son distintos al motor de la herramienta del efector final, por lo tanto, se utilizan dos librerías de programación distintas para poder movilizar estos motores. Con las relaciones geométricas provistas se creó la función `fabricante_fkine`. Esta función posee como argumentos de entrada los tres ángulos en grados, porque el manipulador los recibe de este modo, y como argumentos de salida las coordenadas x, y y z.

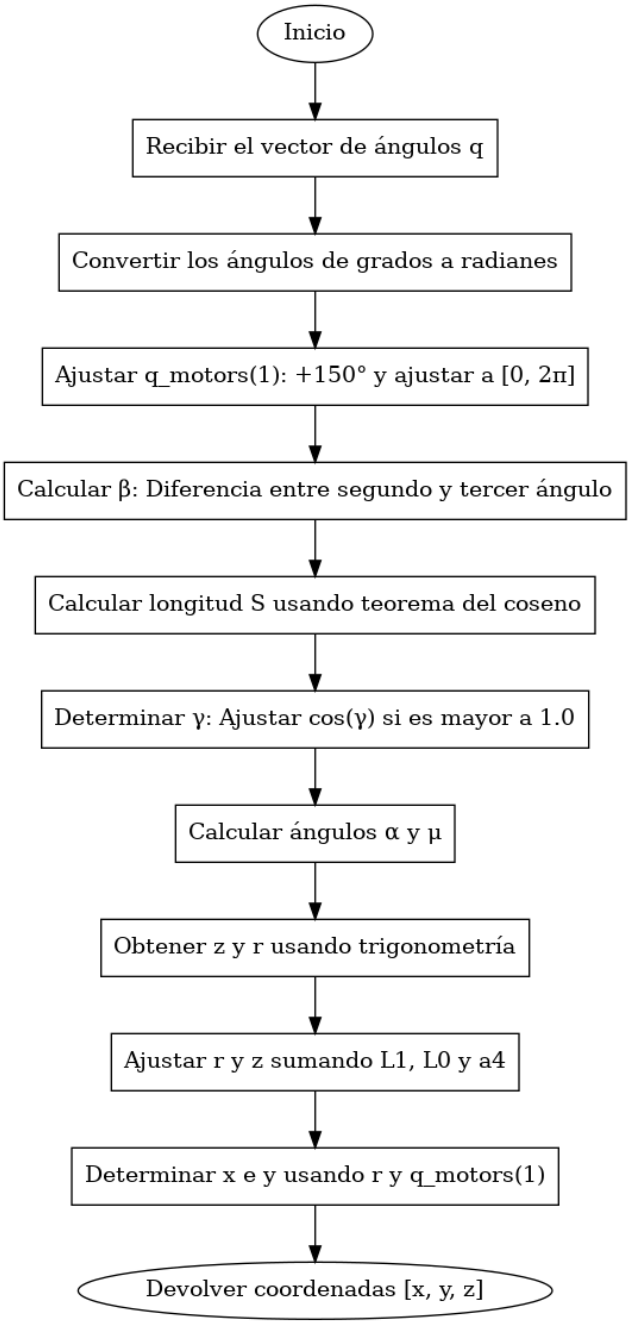

Figura 12: Función `fabricante_fkine`

A continuación se explican las relaciones geométricas para obtener las posiciones a partir de los ángulos de posición de los motores se basan en las dimensiones y ángulos representados en la Figura8. A continuación, se presenta como las relaciones definen a detalle como trabaja la función de `fabricante_fkine`, es decir, el proceso que se utilizó con las relaciones geométricas para poder conseuir los resultados necesarios .

1. Se trabajan los ángulos en grados principalmente porque el manipudor los utiliza de

esta manera en el controlador ESP32, debido a esto se realiza la conversión de q a radianes

$$q_{\mathrm{m}} = \frac{\pi}{180} q.$$

2. Se debe ajustar del ángulo del motor 1, el motor que realiza el giro alrededor del eje z y obtiene las posiciones sobre el eje x. Tambien, se considera que si el ángulo es mayor a 2π, entonces se reduce a un ángulo dentro del rango

$$q_{\mathrm{m}}(1) = q_{\mathrm{m}}(1) + \frac{150\pi}{180},$$

$$\text{si } q_{\mathrm{m}}(1) > 2\pi,\ q_{\mathrm{m}}(1) = q_{\mathrm{m}}(1) - 2\pi.$$

3. Se obtiene el ángulo β a partir de los ángulos de los motores 2 y 3, los cuales obtienen las posciones sobre el eje y y la altura sobre el eje z

$$\beta = q_{\mathrm{m}}(2) - q_{\mathrm{m}}(3).$$

4. Se calcula s, variable que representa la longitud entre la primera junta, q_{θ_1}, y la posición del efector final

$$s = \sqrt{l_1^2 + l_2^2 - 2 l_1 l_2 \cos(\beta)}.$$

5. Se calcula el coseno de b limitando el rango de movimiento de 0 a 180 grados con la condición de si el coseno de b es mayor a 1, se mantenga en 1

$$\cos(b) = \frac{s^2 + l_1^2 - l_2^2}{2 s l_1},$$

$$\text{si } \cos(b) > 1,\ \cos(b) = 1.$$

6. Obtención de b

$$b = \arccos(\cos(b)).$$

7. Se obtiene el ángulo α, el cual representa el ángulo de las segunda junta o q_{θ_2}

$$\alpha = \pi - q_{\mathrm{m}}(2) - b.$$

8 Con los datos obtenidos, se obtiene la posición z

$$z = s \sin(\alpha).$$

9. Se obtiene la distancia r, la cual representa la proyección de la distancia s sobre el plano horizontal. Se realiza el ajuste adicional de la distancia agregando el desfase a_1= 8.14 mm y a_4= 16.8 mm, del Cuadro 2:

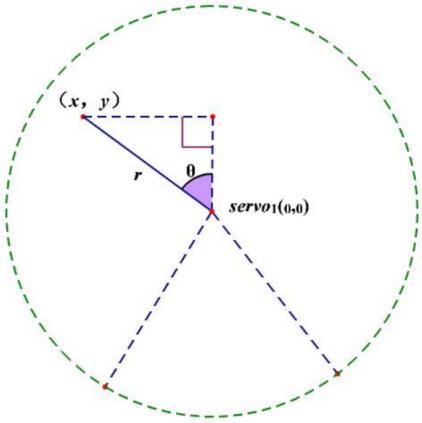

Figura 13: Relación geométrica del movimiento alrededor del eje z

$$r = \sqrt{s^2 - z^2},$$
$$r = r + a_1 + a_4.$$

10. Se ajusta la posición de z, agregándole el desfase de la altura de la base haste la junta 1, con el parámetro d_1= 84.4 mm, del Cuadro 2

$$z = z + d_1.$$

11. Obtención de las posiciones x y y

$$x = r\cos(q_{\mathrm{m}}(1)), y = r\sin(q_{\mathrm{m}}(1)).$$

.Al utilizar los valores de grados de el Cuadro 3, en la función, se obtienen los valores del Cuadro 4:

q_n	**Motor 1 (grados)**	**Motor 2 (grados)**	**Motor 3 (grados)**
q_1	120	89.15	1.57
q_2	160	43.08	7.13
q_3	160	60	40
q_4	91.69	50.82	33.39
q_5	40	159.5	10.6

Cuadro 3: Configuraciones de prueba para `fabricante_fkine`

Vector de posición	**x (mm)**	**y (mm)**	**z (mm)**
p_1	0	-161	209
p_2	44	-52	155
p_3	43	-51	107
p_4	-28	-52	108
p_5	-276	-52	104

Cuadro 4: Resultados de las configuraciones de prueba para `fabricante_fkine`

Por el otro lado, también se posee un motor que moviliza la herramienta en el efector final, por lo que se implementó otra función adicional, la cuál se nombró como `maxarm_fkine` La función mencionada toma como argumentos de entrada los tres ángulos de posicionamiento y el ángulo de la herramienta, y utilizando como base la función `fabricante_fkine`. Se obtiene como parámetro de retorno la matriz de transformación del manipulador con la posición deseada y el ángulo de rotación de la base y la heramienta final.

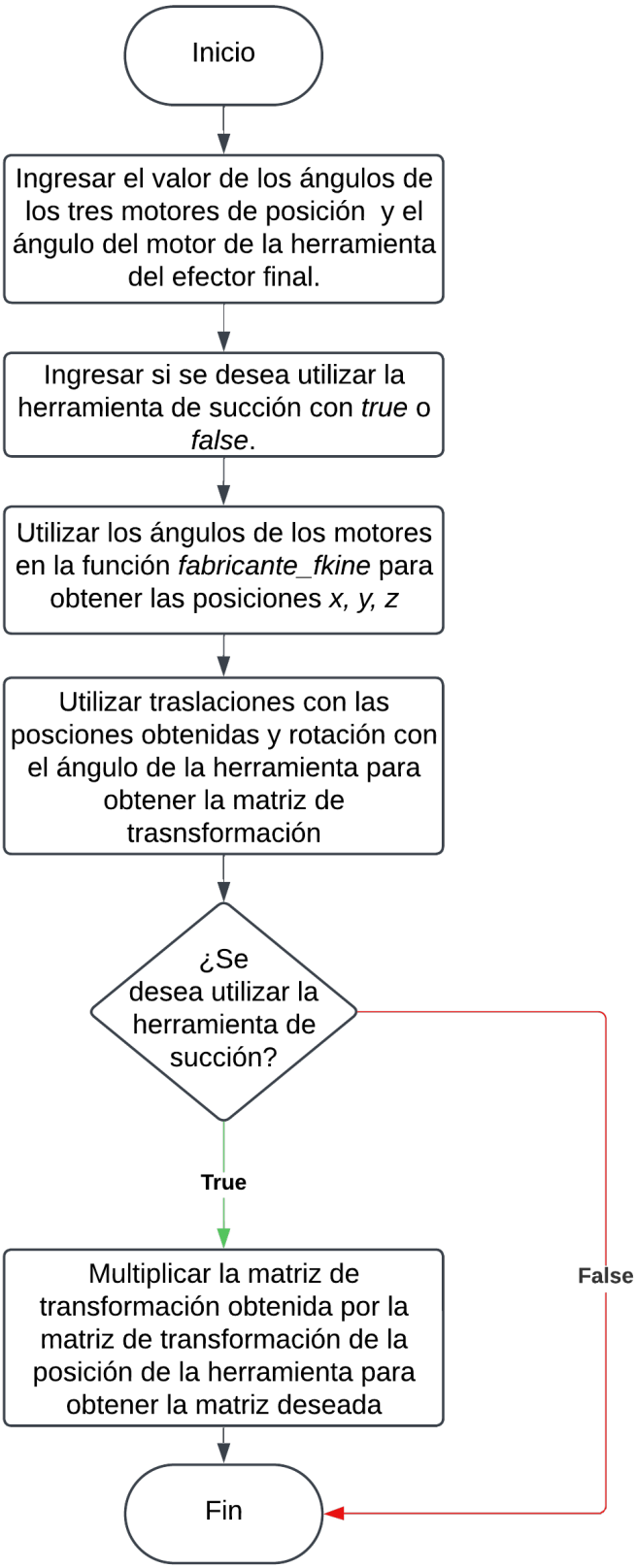

Figura 14: Función `maxarm_fkine`

A continuación, se muestran las matrices generadas a partir de ingresar los angulos de los motores de movimiento y del motor del efector final:

Matriz de transformación para $q = [120, 89.15, 1.57, 45]$ cuando no se utiliza la herramienta de succión

$$\begin{bmatrix} 0.2588 & -0.9659 & 0 & 0 \\ 0.9659 & 0.2588 & 0 & -160.9834 \\ 0 & 0 & 1.0000 & 209.0049 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{5}$$

Matriz de transformación para $q = [120, 89.15, 1.57, 45]$ cuando se utiliza la herramienta de succión

$$\begin{bmatrix} 0.2588 & -0.9659 & 0 & 0 \\ 0.9659 & 0.2588 & 0 & -160.9834 \\ 0 & 0 & 1.0000 & 249.0049 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{6}$$

Matriz de transformación para $q = [160, 43.085, 7.13, 110]$ cuando no se utiliza la herramienta de succión

$$\begin{bmatrix} 0.64 & -0.77 & 0 & 43.77 \\ 0.76 & 0.64 & 0 & -52.16 \\ 0 & 0 & 1 & 154.98 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Matriz de transformación para $q = [160, 43.085, 7.13, 110]$ cuando se utiliza la herramienta de succión

$$\begin{bmatrix} 0.64 & -0.77 & 0 & 43.77 \\ 0.76 & 0.64 & 0 & -52.16 \\ 0 & 0 & 1 & 194.98 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

7.3. Implementación de cinemática inversa

El proveedor del brazo robótico también brinda documentación para la cinemática inversa, es decir, proporciona las relaciones geométricas necesarias para poder obtener los tres ángulos de los motores de movilidad a partir de una coordenada tridimensional. Con esto se implemento la función `fabricante_ikine`, la cual toma como argumentos de entrada las coordenadas x, y y z, y a partir de relaciones geométricas, devuelve los ángulos necesarios para los motores.

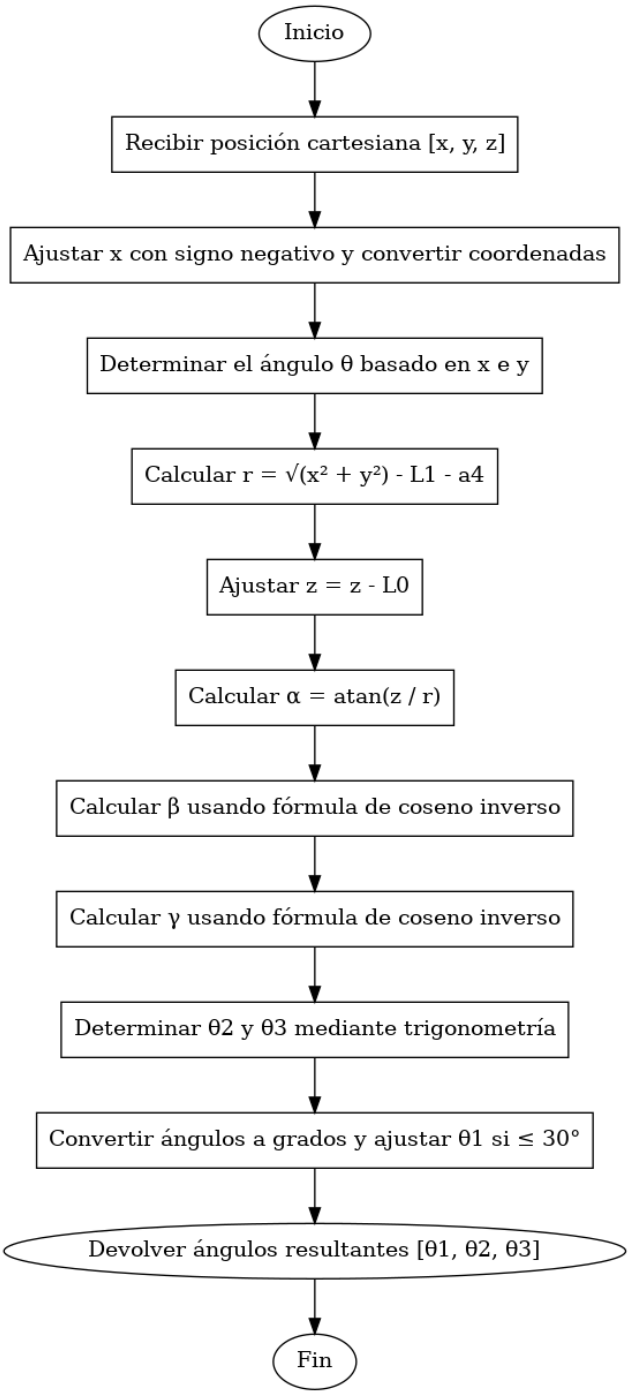

Figura 15: Función `fabricante_ikine`

A continuación, se presenta el desarrollo del algoritmo de `fabricante_ikine`:

1. Las coordenadas cartesianas de la posición son

$$x = \text{pos}(1), \quad y = \text{pos}(2), \quad z = \text{pos}(3).$$

2. Se inicializa el ángulo θ con un valor de 0.

3. A continuación, se debe asignar un valor a este ángulo en base a su posición de x y y

$$\text{Si } x = 0 : \begin{cases} y \geq 0 & \implies \theta = \frac{\pi}{2} \\ y < 0 & \implies \theta = \frac{3\pi}{2}, \end{cases}$$

$$\text{Si } y = 0 : \begin{cases} x > 0 & \implies \theta = 0 \\ x < 0 & \implies \theta = \pi. \end{cases}$$

En caso contrario

$$\text{Si } x < 0,\ \theta = \arctan\left(\frac{y}{x}\right) + \pi,$$

$$\text{Si } x > 0,\ \theta = \arctan\left(\frac{y}{x}\right) + 2\pi.$$

4. A partir de las coordenadas y dimensiones de los eslabones se obtiene la longitud de la distancia r y z

$$r = \sqrt{x^2 + y^2} - a_1 - a_4, z = z - d_1.$$

5. Con los valores de r y z, se obtiene los valores de los siguientes ángulos:

$$\alpha = \arctan\left(\frac{z}{r}\right),$$

$$\beta = \arccos\left(\frac{l_1^2 + l_2^2 - (r^2 + z^2)}{2a_2a_3}\right),$$

$$b = \arccos\left(\frac{a_2^2 + (r^2 + z^2 - a_3^2)}{2a_2\sqrt{r^2 + z^2}}\right).$$

6. Luego, se pueden encontrar los valores de los ángulos de los motores de movimiento del manipulador, obteniéndolos en grados:

$$\theta_1 = \frac{180}{\pi}\theta,$$

$$\text{Si } \theta_1 \leq 30,\ \theta_1 = \theta_1 + 360,$$

$$\theta_2 = \pi - (\alpha + \gamma),$$

$$\theta_3 = \pi - (\alpha + \beta + \gamma),$$

$$q_1 = \theta_1 - 150,$$

$$q_2 = \frac{180}{\pi}\theta_2,$$

$$q_3 = \frac{180}{\pi}\theta_3$$

Al utilizar los valores de grados del Cuadro 5, en la función, se obtienen los valores de el Cuadro 6:

Vector de posición	**x (mm)**	**y (mm)**	**z (mm)**
p_1	0	-161	209
p_2	44	-52	155
p_3	43	-51	107
p_4	-28	-52	108
p_5	-276	-52	104

Cuadro 5: Posiciones de prueba para `fabricante_ikine`

q_n	**Motor 1 (grados)**	**Motor 2 (grados)**	**Motor 3 (grados)**
q_1	121.80	88.37	3.22
q_2	161.99	42.99	9.16
q_3	162.17	68.94	47.79
q_4	90.02	52.02	34.22
q_5	41.75	144.16	23.75

Cuadro 6: Resultados de configuraciones de prueba para `fabricante_ikine`

Cabe destacar que la función antes mencionada no proporciona el ángulo de la herramienta por lo que también se implementó una función adicional llamada `maxarm_ikine`. Esta utiliza como base a la función `fabricante_ikine`, pero toma como argumento de entrada la matriz de transformación y, a partir de esta, se obtienen los cuatro ángulos necesarios para los motores del manipulador.

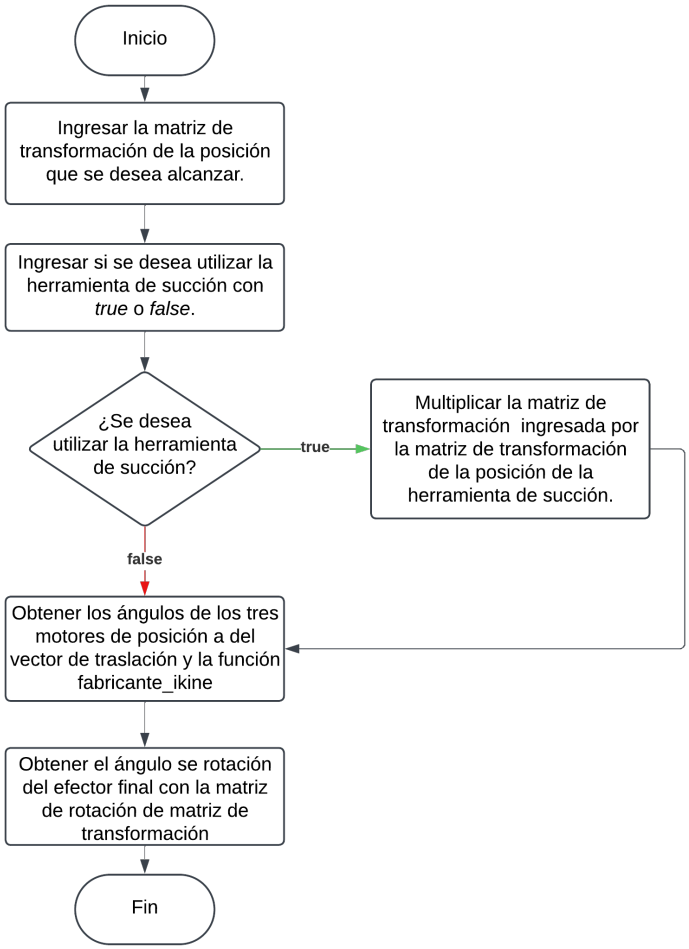

Figura 16: Función `maxarm_ikine`

Al utilizar la matriz:

$$\begin{bmatrix} 0.3706 & 0.9288 & 0 & -249.9999 \\ -0.9288 & 0.3706 & 0 & -52.0001 \\ 0 & 0 & 1.0000 & 104.0000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{9}$$

Se obtiene el siguiente vector de ángulos si se selecciona que no se utiliza la herramienta de succión:

$q = [41.75, 144.16, 23.75, 110]$.

Además, se obtiene el siguiente vector de ángulos si se selecciona que se utiliza la herramienta de succión:

$q = [41.75, 137.73, 11.18, 110]$ En estos vectores, los primeros tres ángulos de movilidad del brazo y el último ángulo es la rotación del efector final.

7.4. Implementación de ejecución de trayectorias

La implementación de trayectorias se utilizó principalmente en la aplicación para el ecosistema de róbotica y el uso de la interfaz gráfica. Para poder realizar una trayectoria principalmente se utiliza cinemática inversa, porque se utiliza un listado de posiciones a las cuales se necesita acceder y con este listada se obtienen el conjunto de ángulos necesarios, a partir de las funciones que se crearon previamente.

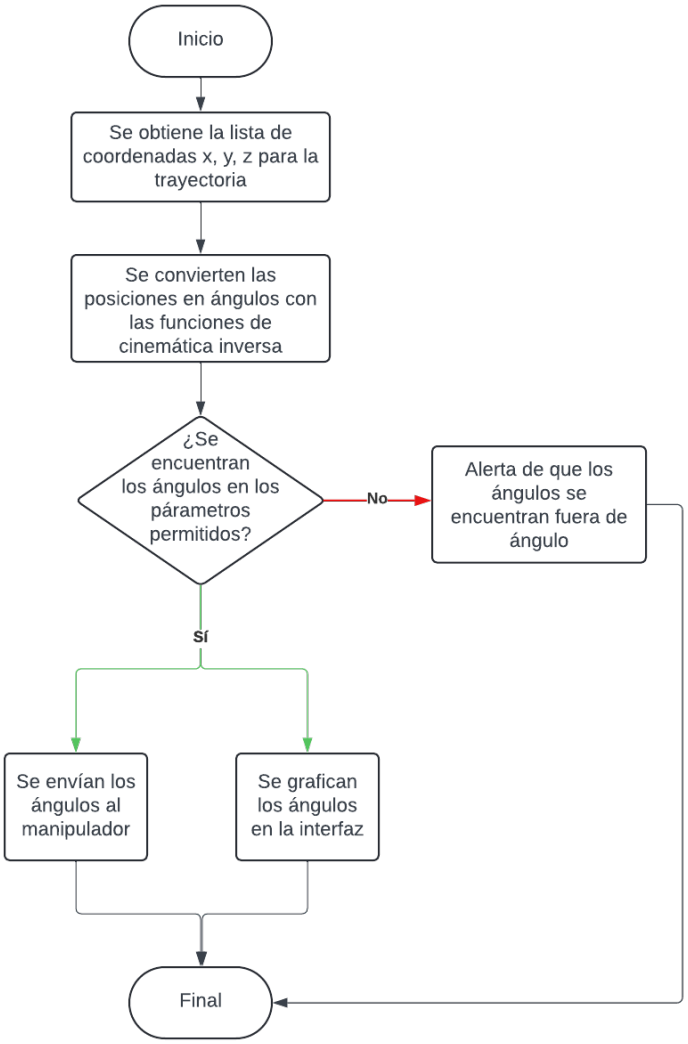

Figura 17: Diagrama de implementación de trayectoria.

Para poder verificar las trayectorias se observan las poses generadas con el toolbox de Peter Corke, y con un secuencia lineal de los puntos, antes de enviarlas al manipulador. A partir, de esto se realizó pruebas de trayectorias tanto simuladas como físicas con el siguiente conjunto de coordenadas.

Vector de posición	x (mm)	y (mm)	z (mm)
p_1	-5	-159	205
p_2	-45	-50	150
p_3	-48	-53	102
p_4	30	-52	108
p_5	250	-52	104
p_6	0	-161	209

Cuadro 7: Posiciones de prueba para una trayectoria

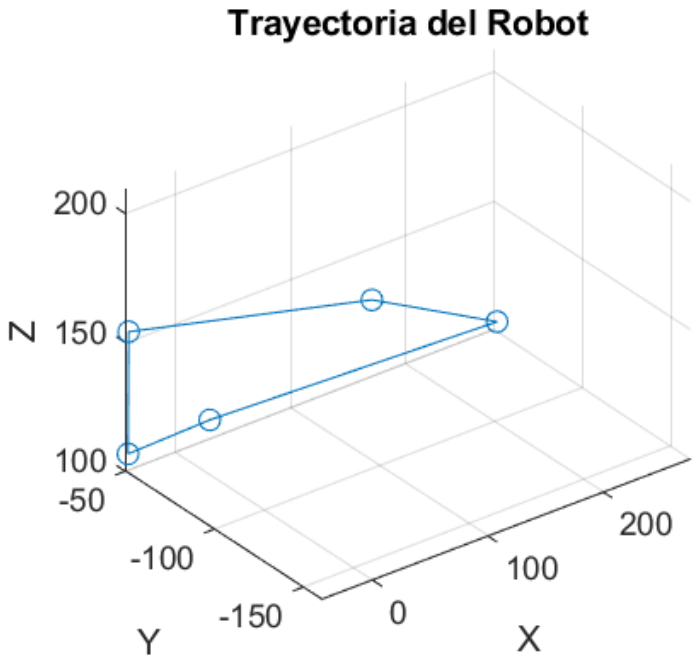

Figura 18: Resultado de una trayectoria lineal.

7.5. Simulación del manipulador en Webots

Se llevó a cabo una investigación detallada sobre los métodos para la creación de modelos de manipuladores seriales en Webots, lo que permitió comprender las herramientas y funciones disponibles en el software para este propósito. Con base en esta información, se diseñó un algoritmo que define los pasos necesarios para estructurar el modelo, incluyendo desde la definición de parámetros geométricos y cinemáticos hasta la configuración de las articulaciones y los ejes de movimiento. Posteriormente, empleando este algoritmo junto con las medidas y orientaciones obtenidas del manipulador en estudio, se desarrolló el modelo dentro del entorno de Webots, asegurando su precisión mediante la implementación de los parámetros establecidos. Finalmente, se programó un controlador basado en las ecuaciones y relaciones cinemáticas previamente derivadas, permitiendo la simulación del manipulador con movimientos coherentes y funcionales, demostrando así la validez y exactitud del modelo generado.

A pesar de que se poseía una simulación del manipulador en Matlab, también se deseaba obtener una simulación del manipulador que pudiera comportarse como el robot físico y fuera visualmente parecido. Para conseguir este simulación se utilizó la plataforma de Webots. Aunque esta plataforma incluye en sus librería varios modelos de robots, ninguno de ellos era el manipulador MaxArm. Se investigó la manera de crear un robot desde cero y implementó un flujo de pasos para seguir durante su creación.

Cabe destacar que para implementar un robot que se creó, se necesita obtener las dimensiones necesarias y al mismo tiempo ubicar el desfase correcto de cada una de las juntas del manipulador. Esto es importante porque permite que el controlador funcione de igual manera en Matlab como en Webots, lo que dará como resultado el comportamiento representativo del manipulador físico. Para demostrar que que el controlador funciona de igual manera en amabas plataformas se realizaron las siguientes pruebas que verifican su movimiento, estos valores de prueba fueron los mismos que se utilizaron para la verificación de la matriz de DH.

Adicionalmente, se realizaron las comparaciones entre el modelo creado en Webots con el modelo creado en Matlab. Esto se realizó debido a que el modelo de Matlab se comparó previamente con el módelo físico, mostrando que el modelo implementado se comportaba como el modelo real. Además, se utilizaron los mismos parámetros de desfase en juntas y funciones para ambos modelos, lo que sugiere que ambos deberían tener los mismos comportamientos, como se muestran en las figuras a continuación de esta sección.

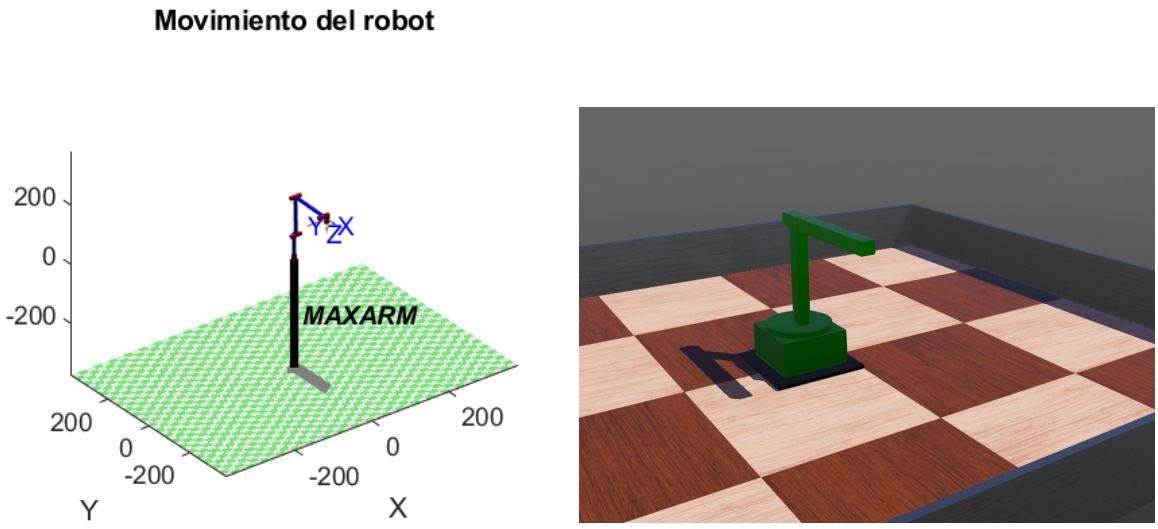

Figura 19: Prueba de posición con q = [0 0 0 0 0] en las juntas

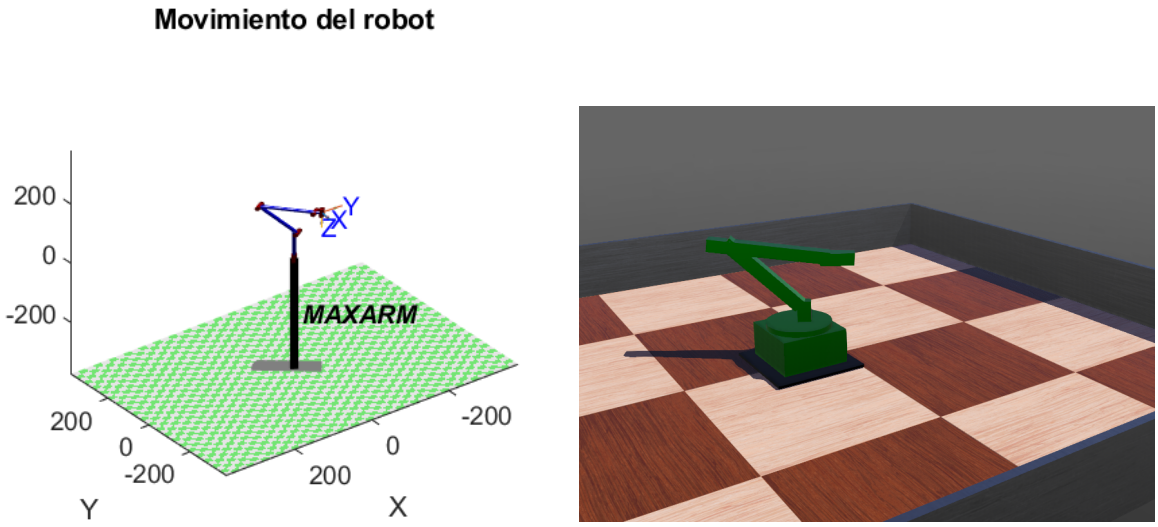

Figura 20: Prueba de posición con q = [-60.2 46.9 54.0 7.1 45.0] en las juntas

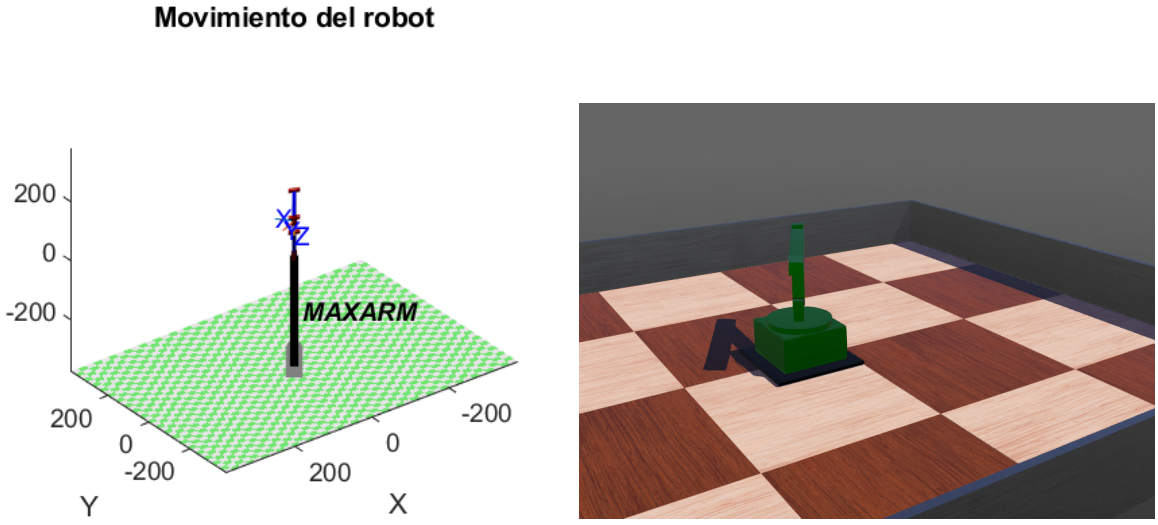

Figura 21: Prueba de posición con q = [28.3 53.9 56.0 2.1 120.0] en las juntas

CAPÍTULO 8

Integración del manipulador serial MaxArm con el ecosistema Robotat

De los objetivos principales de este trabajo está la integración del manipulador MaxArm al ecosistema de robótica, Robotat. Primero, se asignó una $IPAddress$ específica en el $router$ de Robotat al manipulador, a partir de la $MACAddress$ del dispositivo, esta dirección identifica específicamente al brazo robótico, por lo tanto, cada vez que se intente conectar se le asignará el IP: 192.168.50.140 del $router$. Esto hace efectiva la conexión y conveniente asignarle un número al robot que lo conecte a este IP.

Cabe destacar que al estar conectado el brazo robótico a la misma red que la utilizada para el ecosistema de robótica, se pueden enviar datos al robot y al mismo tiempo obtener datos del sistema de captura de movimiento. Poder obtener posiciones exactas a las que se desea llevar el efector final o la herramienta, permite la creación de trayectorias efectivas.

Además, se desarrolló una librería en Matlab diseñada para facilitar la comunicación de datos a través de una conexión Wi-Fi utilizando el protocolo TCP, optimizando la transferencia de información para el manipulador serial. Adicionalmente, se realizaron adaptaciones a la programación original del proveedor del manipulador, asegurando que este pudiera interpretar y operar eficientemente con los datos recibidos a través de la comunicación inalámbrica. Dentro del entorno de Matlab, se creó una interfaz gráfica para la planificación de trayectorias y poses, habilitando el envío de estas instrucciones al manipulador mediante la librería desarrollada. Finalmente, el manipulador fue integrado en el ecosistema de robótica Robotat, permitiendo la interacción con el sistema de captura de movimiento para tomar datos precisos de posiciones en el entorno, los cuales se emplearon para generar trayectorias y poses adaptadas a las condiciones del espacio capturado.

8.1. Librería del MaxArm

Al realizar la implementación del brazo robótico se crearon y utilizaron varias librerías poder realizar la conexión entre Matlab y Arduino, que al mismo tiempo permitieran el control del manipulador . Principalmente, las librerias del MaxArm para ESP32 se creron en Arduino, las cuales se enfocaron en la recepción de datos a través de la conexión TCP/IP y el control de la cinemática del robot de manera simultánea. Como base para la implementación se utilizaron las librerías del proveedor para poder controlar el manipulador fueron las que controlan los motores y la herramienta del efector final.

Adicionalmente, se utilizaron las librerías de Matlab para poder encontrar los ángulos y posiciones que tendrá el brazo robótico a través de su cinemática inversa y directa, así mismo se encargan de controlar el manipulador de manera inálambrica con la interfaz. También, se encargan de tomar posiciones del ecosistema de robótica y convertirlos en ángulos que permitan al manipulador alcanzarlas. A continuación, se presentan las funciones creadas en Matlab que controlan la cinemática inversa y directa del manipulador MaxArm.

Funciones Matlab	Descripción
`fabricante_fkine(q)`	Función de cinemática directa que convierte los tres ángulos de los motores de posición en coordenadas tridimensionales.
`maxarm_fkine(q, suction)`	Función de cinemática directa que convierte los tres ángulos de los motores y el ángulo de la herramienta en una matriz de transformación.
`fabricante_ikine(pos[3])`	Función de cinemática inversa que convierte las tres coordenadas en los tres ángulos de los motores
`maxarm_ikine(Td, suction)`	Función de cinemática inversa que convierta una matriz de transformación del robot en los tres ángulos de los motores de movimiento y la rotación de la herramienta en el efector final.
`maxarm_connect(number)`	Función que realiza la conexión entre el usuario y el manipulador, lo cual permite enviarle datos.
`maxarm_disconnect(robot)`	Función que desconecta el manipulador de la interfaz del usuario.
`maxarm_send(robot, q, suction)`	Función que envía el array de os valores de los ángulos y si se desea ralizar la succión a través la conexxión Wifi.
`robotat_connect()`	Realiza la conexión entre el ususario y el sistema de robótica.
`robotat_disconnect(tcp_obj)`	Deshace la conexión entre el ususrio y el sistema de robótica.
`robotat_get_pose(tcp_obj, agent_id, rotrep)`	Función que obtiene la pose de un agente dentro del sitema de robótica.

Cuadro 8: Librerías del proveedor utilizadas para el control del robot

Adicinalmente, también se utilizaron librerías dentro del controlador del manipulador que permiten realizar los movimientos deseados, y al mismo tiempo se puede recibir los datos obtenidos a través la conexión realizada con las funciones mencionadas previamente. Estas librerías fueron creadas dentro de Arduino para permitir su compatibilidad con el controlador ESP32. Las librerías provistas en conjunto con el manipulador fueron las del

siguiente cuadro.

Librería Arduino	**Descripción**
`ESPMax`	Es la librería que posee las funciones que comunican los datos de movilidad, es decir ángulos y posiciones con las funciones de los motores de posición.
`_espmax`	Esta librería se enfoca en la cinemática inversa y directa del manipulador, para producir valores que puedan controlar al MaxArm.
`LobotSerialServoControl`	Esta es la librería de la configuración de los motores de posición, permite que estos se puedan utilizar y recibir datos.
`SuctionNozzle`	Es la librería que controla la succión de la ventosa que se encuentra como herramienta final.
`Servo`	Es la librería que configura y establece los límites del motor de la herramienta del efector final.
`ESP32PWMServo`	Esta librería se enfoca en la manejo del motor de la herramienta del efector final con los datos que se le den.

Cuadro 9: Librerías del proveedor utilizadas para el control del robot

De estas librerías se utilizaron funciones específicas que permiten el control del robot. Además, se crearon nuevas funciones que permiten la conexión inalámbrica del manipulador. Es importante mencionar que para el uso de la succión se debe utilizar la correcta combinación del mandos de activación de la bomba y la válvula para que realice la tarea efectivamente. Se debe activar la bomba y cerrar la válvula para succionar, y desactivar la bomba y abrir la válvula para liberar lo que se encuentre succionado.

Función	Descripción
`connectToWiFi()`	Es la función encargada de conectar el manipulador al IP de la red deseada.
`receiveData(WiFiClient &client)`	Esta función se enfoca en recibir los datos del cliente conectado y devolver estos datos en un string que pueda utilizarse.
`deserializeData(const String &data)`	A esta función le ingresa el array obtenido, desglosa los datos en formato json y losasigna a las variables necesarias, las cuales devuelve.
`set_angles(volatile float angulos[4])`	Es la función encargada de recibir los ángulos de movilidad y enviarlos a los motores de posición.
`SetPWMServo(int id, int pul, int duration)`	Recibe el ángulo de rotación de la herramienta en formato de pulsos, la duración del movimiento y la identificación del motor que se quiere mover. Se encarga del movimiento de rotación del efector final.
`Valve_off()`	Cierra la válvula.
`Valve_on()`	Abre la válvula.
`Pump_off()`	Desactiva la bomba.
`Pump_on()`	Activa la bomba.

Cuadro 10: Funciones utilizadas para el control del robot

8.2. Interfaz gráfica

Para poder integrar todas las funciones creadas tanto en Matlab como en Arduino se implementó una interfaz que manejara todas estas librerías de manera sencilla.

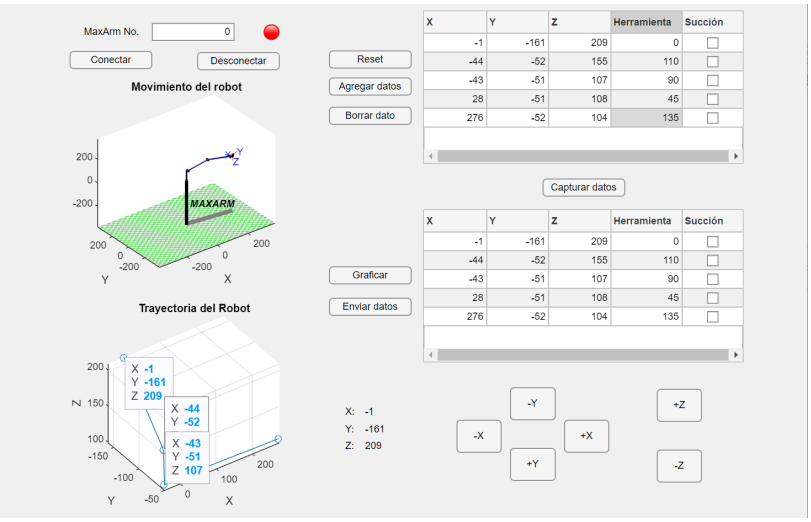

Figura 22: Interfaz para el control inalámbrico del manipulador MaxArm

8.2.1. Conexión inalámbrica

Se integró una casilla que solamente recibe valores numéricos que indica que número de robot se desea conectar. Esto se realiza a través de la función `maxarm_connect`, la cual se enfoca en conectarse al IP asignado en el router para el brazo robótico, al presionar el botón de `Conectar`. A través de este IP se pueden comunicar datos desde la interfaz de Matlab al ESP32.

Adicionalmente, se agrego un LED que indicara si la conexión pudo realizarse, este cambia a color verde si es exitosa y en el caso contrario, permanece roja y muestra una alerta. También se creó el botón de Desconectar, la cual rompe la conexión entre el manipulador y la interfaz.

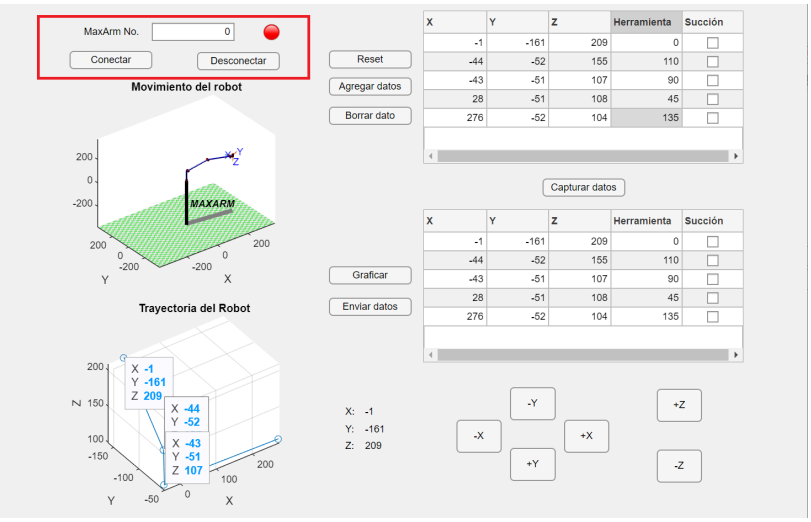

Figura 23: Panel de conexión con el manipulador.

8.2.2. Planificación e implementación de trayectorias

Para poder crear una construcción de trayectorias se agregó una tabla editable en la cual se puede ingresar los valores de las posiciones, el ángulo de la herramienta y si se desea activar la succión de la herramienta final. Luego, con el botón de `Capturar datos`, se fijan los datos que se desean utilizar en la tabla inferior que no es editable. A partir de la segunda tabla mencionada, se pueden utilizar los botones de `Graficar` y `Enviar Datos`. El botón de `Enviar Datos` manda los ángulos necesarios al ESP32 para realizar el movimiento físico del manipulador y ejecutar la trayectoria. Por el otro lado, el botón de `Graficar`, realiza una simulación visual de las poses realizadas durante la trayectoria y el movimiento lineal que se ejecuta para llegar a cada punto.

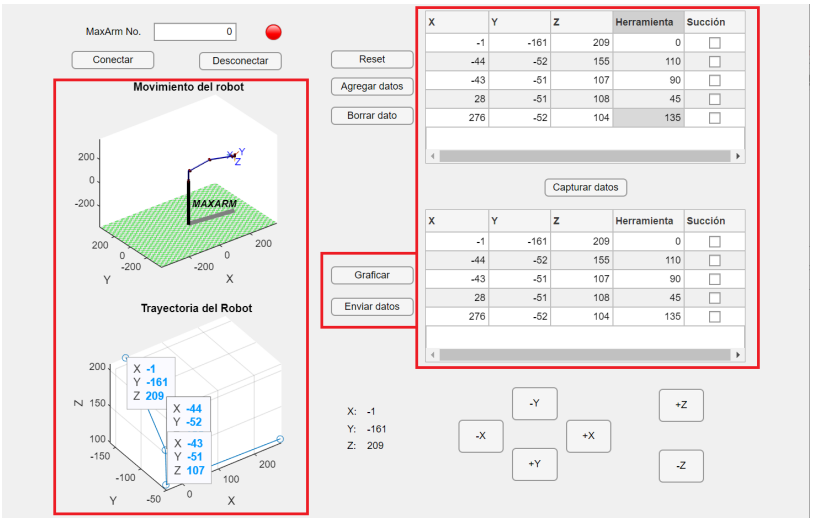

Figura 24: Opciones utilizadas para la ejecución de trayectorias.

8.2.3. Ejecución de poses

Al realizar trayectorias, previamente se necesitan saber que poses serán las que se integraran a la lista a ejecutar, por lo que se creo un panel de cambio de pose. Este consiste en botones que aumentan o disminuyen en un milímetro las posiciones en x, y y z. También, se puede visualizar en que posición se encuentra el efector final para poder tomar la pose exacta que se necesita implementar.

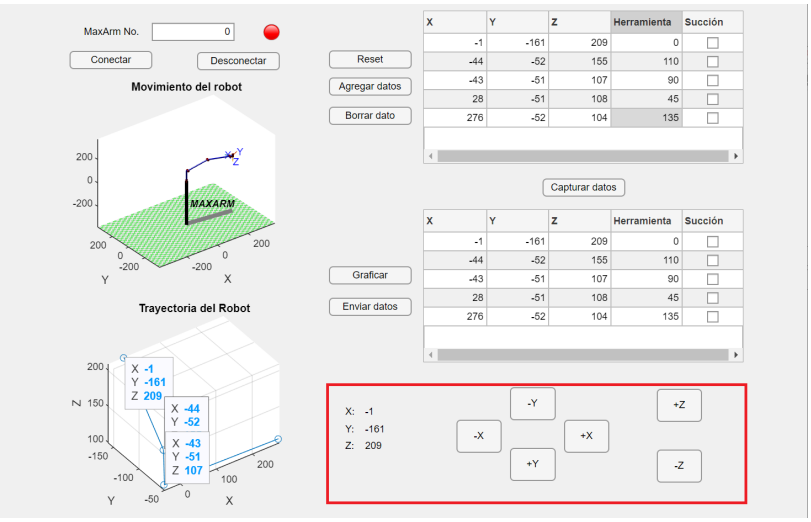

Figura 25: Panel para control de posicionamiento del manipulador.

8.2.4. Botones complementarios

Para complementar la interfaz se agregó un botón de Reset que borra todas tablas y ubicar el brazo robótico en su posición de reinicio y desactivar la herramienta. También se implementó un botón de agregar dato que solo permite ingresar una nueva fila de información, si la línea previa esta llena. Y el botón de Borrar dato, elimina la última fila de información agregada

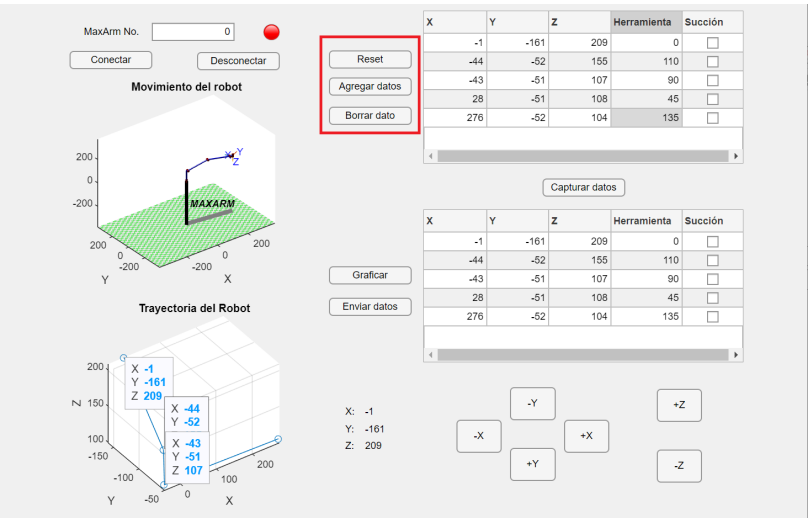

Figura 26: Botones complementarios de la interfaz.

Esta interfaz permite utilizar todas las librerías creadas al mismo tiempo. Se utiliza para realizar la conexión inalámbrica entre el manipulador y el usuario, para crear una visualización previa del modelo, la visualización de la trayectoria y al mismo tiempo crear una lista de posiciones para realizar trayectorias. Además, permite enviar esta lista de datos

directamente al robot para crear las poses planificadas en tiempo real.

8.2.5. Interacción dentro del sistema Robotat

Para poder integrar el manipulador al ecosistema de robótica se utilizaron las librerías del robotat para obtener las posiciones X, Y, Z y los ángulos de roll, pitch y yaw de los markers, de las posiciones que se requerían alcanzar y la base del manipulador. Con estos datos se construyeron las matrices de posición necesarias para complementar el movimieninto de manipulador. Adicionalmente, se utilizó la función $maxarm\_ikine$, la cual utiliza la matriz de transformación que representa la posición deseada respecto a la base, con lo cual se obtienen los ángulos para obtener esa posición. Para poder unificar los datos tomados y obtener la posición del efector final, se creó la siguiente función en Matlab:

Función de Matlab	**Descripción**
`optitrack( robotatData , int MarkerNo )`	Es la función que conecta la posición de base del brazo robótico y obtiene la matriz de transformación respecto a una pose en el ecosistema de robótica.

Cuadro 11: Función de creación de pose a partir de datos del sistema de captura de movimiento

8.3. Integración con OptiTrack

Para poder comprobar el funcionamiento de todas las funciones desarrolladas que representan la cinemática del manipulador y su conexión inalámbrica, se realizaron pruebas dentro del sistema de robótica. Estas pruebas demostraron el correcto funcionamiento del manipulador a partir de la obtención de posiciones e implementación de trayectorias con las mismas.

8.3.1. Montaje y configuración

Para comenzar con las pruebas se debió crear un marker al lado del manipulador que representará la base del mismo, el cual se debió calibrar respecto al centro del ecosistema de robótica, tomando este punto como $p = [0, 0, 0]$. Este marker fue implementado a partir de puntos refractantes a distintas distancias y alturas para obtner una configuración distinta a la de los markers existentes.

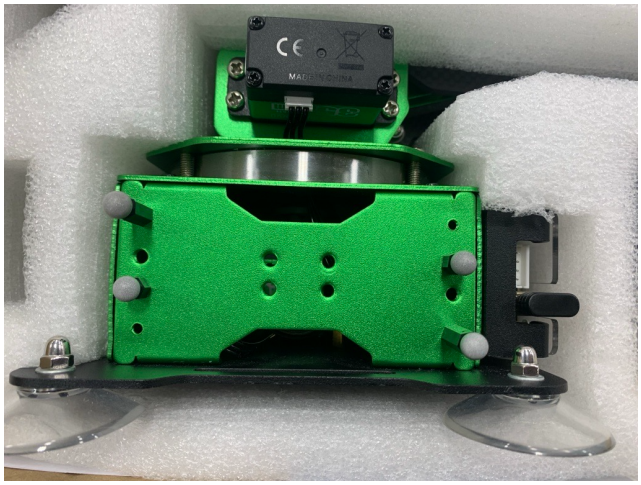

Figura 27: Marker de base del manipulador MaxArm

Adicionalmente, los cubos que fueron provistos en conjunto con el brazo robótico se utilizaron como objetivo a alcanzar con el efector final, colocando los mismos sobre el marker que representa el punto dentro del ecosistema y agregando el desplazamiento necesario, considerando la geometría del cubo.

Figura 28: Pose del manipulador respecto a la posición del cubo

Con los datos de la geometría del cubo , la posición de la base el manipulardor y los datos de las posciones de los markers dentro del ecosistema de robótica, se realizó la obtención de posiciones del efector final a través del algoritmo de cinemática inversa del manipulador. Para poder imp

8.3.2. Cinemática inversa

Para poder crear la cinématica inversa del manipulador dentro del ecosistema de robótica, se siguió un algoritmo que obtuviera los datos necesarios para crear la matriz de transformación con los datos de la posición final y rotación del manipulador para alcanzar la ubicación de un marker.

1. Primero, se colocó el manipulador en el centro del ecosistema de robótica con los ejes del manipulador orientados acorde a los ejes del ecosistema de robótica, y en esta posición se obtuvieron los datos del marker en su base con el Optitrack.

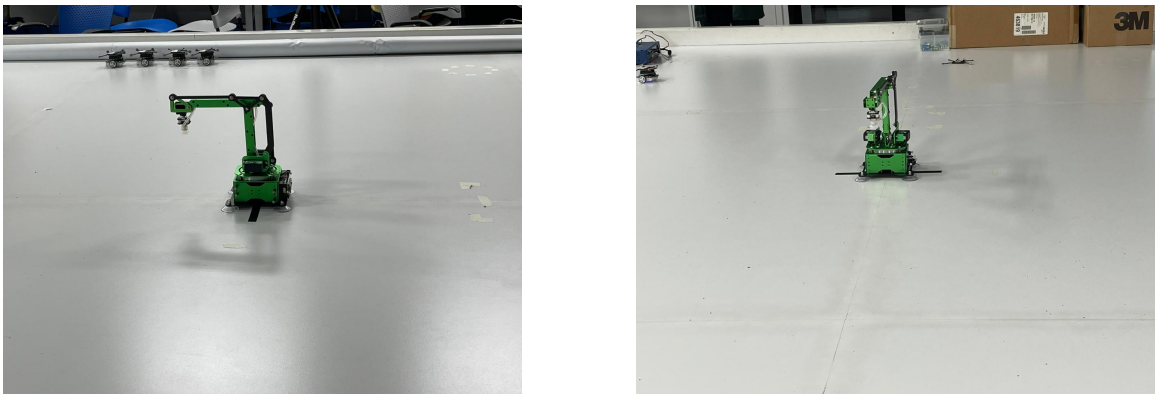

Figura 29: Posición inicial del manipulador dentro del ecosistema

2. Con los datos obtenidos se creó la matriz de transformación del manipulador respecto al origen.

$$^{o}T_{m} \tag{10}$$

3. Se asumió que la matriz de transformación obtenida en el origen debía ser la matriz identidad de tamaño 4 x 4. Esto se debe hacer porque en el origen, el marker deberia encontrarse en la posición $p = [0, 0, 0]$ y sin ningún tipo de rotación.

$$^{O}T_{B} = I \tag{11}$$

4. Con los datos obtenidos y supuestos, se buscó crear una matriz que represente la transformción que se necesita para que la matriz del robot obtenida respecto al origen sea la matriz identidad. La matriz de transformación que se obtuvó de la relación de estos datos sirvió como la calibración para ubicar la base del robot correctamente y lo que permite utilizarla en cada posición.

$$\begin{aligned} ^{m}T_{B} &= (^{o}T_{m})^{-1} \cdot ^{O}T_{B} \\ ^{m}T_{B} &= (^{o}T_{m})^{-1} \cdot I \end{aligned} \tag{12}$$

5. Para poder manipular el brazo robótico, primero se debió obtener una posición que se desea alcanzar, es decir, un punto en el ecosistema de robótica. Con la información se obtuvo una matriz de transformación con los datos de la posición del marker respecto al origen del ecosistema.

$$^{O}T_{E} \tag{13}$$

6. Luego, al ubicar el robot en cualquier posición, se obtuvieron los datos del marker del manipulador respecto al sistema de robótica. Con los cuales se obtiene la matriz de transformción que representen estos mismos.

$$^{O}T_{m} \tag{14}$$

7. Finalmente, se obtuvo la posición que debía tener el efector final respecto a la base del manipualdor para poder alcanzar la posición deseada, es decir, se obtuvo la matriz de transformación que obtuviera los datos mencionados. Para pode obtener la matriz mencionada, se crearon relaciones con las matrices obtenidas en los pasos anteriores.

$$^{B}T_{E} = (^{m}T_{B})^{-1} \cdot (^{o}T_{m})^{-1} \cdot^{O} T_{E} \tag{15}$$

A partir de esta serie de pasos se creó el flujo de pasos que se implemantaron para crear la función `optitrack`:

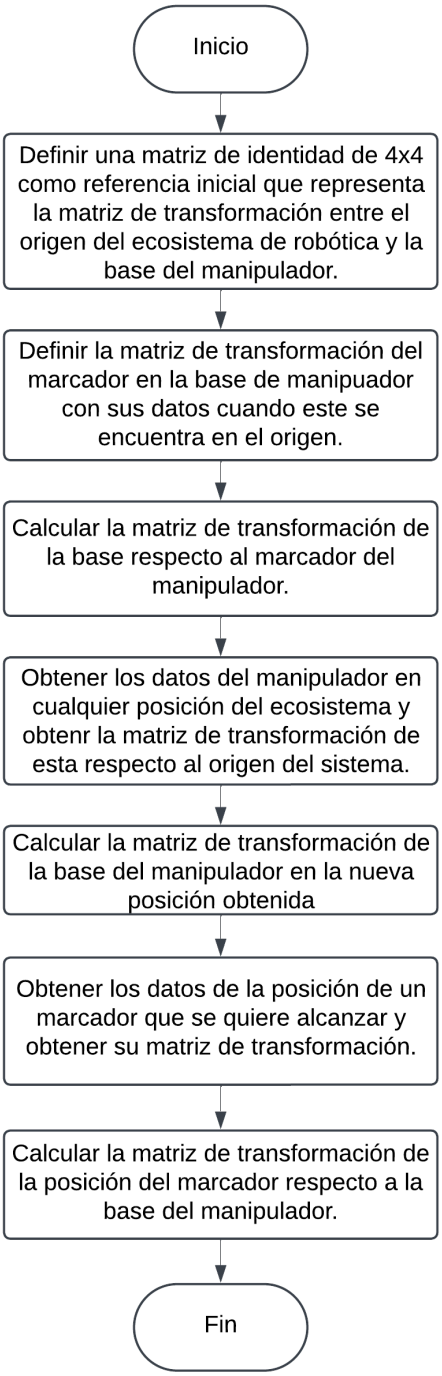

Figura 30: Funcion `optitrack`

8.3.3. Pruebas y obtención de datos

Con el objetivo de demostrar el correcto funcionamiento de las funciones dentro del ecosistema de robótica, se realizaron pruebas de obtención de datos y posicionamiento del

manipulador con tres markers distintos alrededor de él. Se utilizaron los markers 13, 20 y 11 para la toma de datos, realizando una trayectoria para que el manipulador consiguiera las posiciones de estos. Para poder conseguir las matrices de transformación que tuvieran los datos deseados para posicionar los markers sobre los markers deseados se utilizó la función `optitrack`.

El primer objetivo fue el marker 13, sobre el cual se posicionó el cubo inicialmente y se tuvo la finalidad de que el manipulador pudiera succionar el cubo desde esta posción para continuar la trayectoria. Para ubicar el manipuador a centro del objetivo, se poseía un desfase en $x : -10mm$, y se utilizó el desfase en $z : 45mn$, el cual representa la altura del cubo sobre el marcador. Esto permitió que la herramienta del manipulador sujetara el cubo inicialmente.

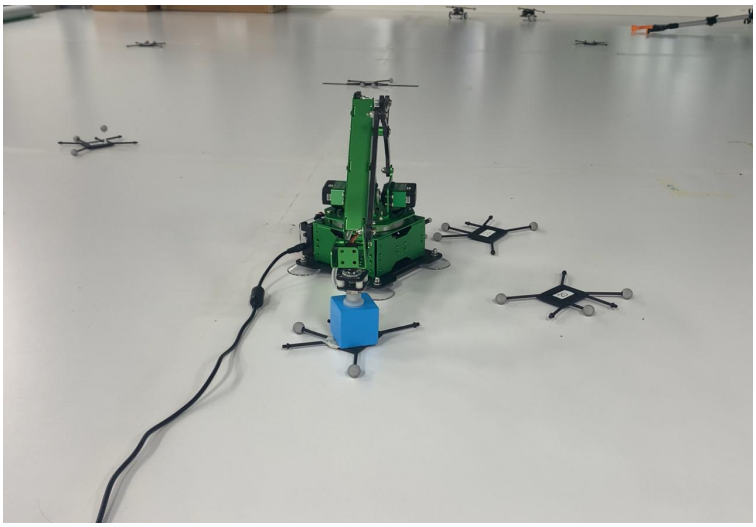

Figura 31: Pose del marcador No.13

Al utilizar los datos del marcador y los desfases necesarios con la función `optitrack`, se obtuvo la matriz de transformación del robot para obtener la pose para alcanzar al cubo en el marcador 13.

$$\begin{bmatrix} 0.7150 & -0.6991 & 0 & -156.0116 \\ 0.6991 & 0.7150 & 0 & -148.6053 \\ 0 & 0 & 1.0000 & 87.1395 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{16}$$

Con la matriz, se obtuvo una posicion de $[-156.01, -148.61, 87.14]$, las cuales representan las cordenadas x, y, z respectivamente, y el ángulo de giro de herramienta de 29.25^o.

Luego, el sigueiente objetivo fue el marker 20, el cual tuvó la finalidad de ser un punto medio en la trayectoria, por lo cual, solamente se realizó un movimiento sobre él con una altura distante al marker. Para centrarse en el matker se realizón desfase en $y : 30mm$, y se le agregó un desfase en $z : 150mm$ para obtener una altura alejada del marker.

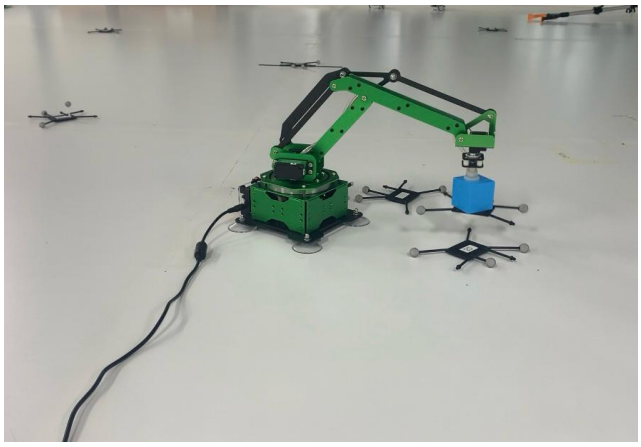

Figura 32: Pose del marcador No.20

A partir de estos datos se obtuvo la matriz de transformación que generara la pose deseada para posicionarse sobre el marker 20.

$$\begin{bmatrix} -0.7855 & 0.6188 & 0 & 24.9921 \\ -0.6188 & -0.7855 & 0 & -265.6011 \\ 0 & 0 & 1.0000 & 192.0711 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{17}$$

Obteniendo la posición $[24.99, -265.60, 192.07]$, las cuales representan las cordenadas x, y, z respectivamente, y el ángulo de giro herramienta de 87.14^o.

Finalmente, se consiguón alcanzar la posición del marker 11, el cual tenía como objetivo ser la posición meta de la trayectoria, es decir, ser el punto en el cual el manipulador colocará el cubo. Para centrar la herramienta en este marker se realizó un desfase en $x : -10mm$ y en $y : -20mm$. Adicionalmente, se le agregó el desfase en $z : 55mm$, el cual representa la altura del cubo y el marcador debajo del mismo, permitiendo colocar el objeto sobre el marker 11.

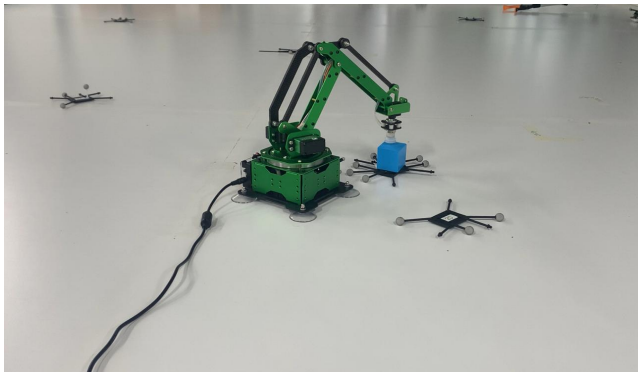

Figura 33: Pose del marcador No.11

Se obtuvo la pose final de la trayectoria con la matriz de transformación a continuación.

$$\begin{bmatrix} 0.3584 & 0.9336 & 0 & 191.4441 \\ -0.9336 & 0.3584 & 0 & -159.3937 \\ 0 & 0 & 1.0000 & 96.8100 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{18}$$

Siendo la posicion final obtenida de $[191.44, -159.39, 96.80]$, las cuales representan las cordenadas x, y, z respectivamente, y el ángulo de herramienta de 59.22^o

Este proceso de obtención de datos, construción de matrices de transformación y ejecución de poses se realizó a través de la interfaz de usuario de manera sencilla. Esto permitió el uso de las funciones desarrolladas, demostrándo su correcto funcionamiento y utilidad para generar el movimiento del robot dentro del ecosistema de robótica.

8.3.4. Ejecución de trayectorias

Para facilitar la ejecución de trayectorias se implementó una sección dentro de la interfaz que permite obtener las posiciones del marker deseado con los desfases necesarios para obtener la posición exacta. Esta seción agregada tiene tres cajas de texto en las que sue pueden escribir los siguientes datos en formato numérico: el número de marker y las desfase de las coordenadas x, y, z. Con estos datos se puede utilizar la función `optitrack` para generar las poses necesarias con los datos ingresados.

Las matrices de transformación obtenidas de la función `optitrack` se ingresan a la función de cinemática inversa `maxarm_ikine` con el obejtivo de obtener los ángulos a enviar para a través de la conexión TCP/IP que se realizaba entre la interfaz y el manipulador. Adicionalmente, los datos de posiciónes y ángulos obtenidos se podían agregar en una lista que permitía la simulación y la modificación de los valores previa a utilizarlos con el modelo físico.

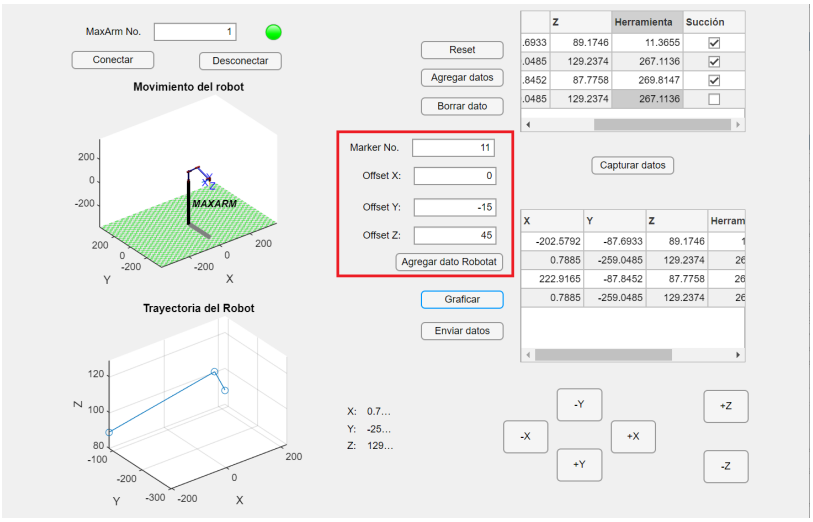

Figura 34: Integración de Optitracks en la trayectoria

Con esta sección integrada en la interfaz se pudo realizar una trayectoria obteniendo datos de los marcadores 13, 20 y 11. Tomando el marcador 13 como punto de inicio para tomar el cubo, el marcador 20 como el punto medio y el marcador 11 como el punto final donde se coloca el cubo al finalizar. Se ejecutó la trayectoria mostrada a continuación.

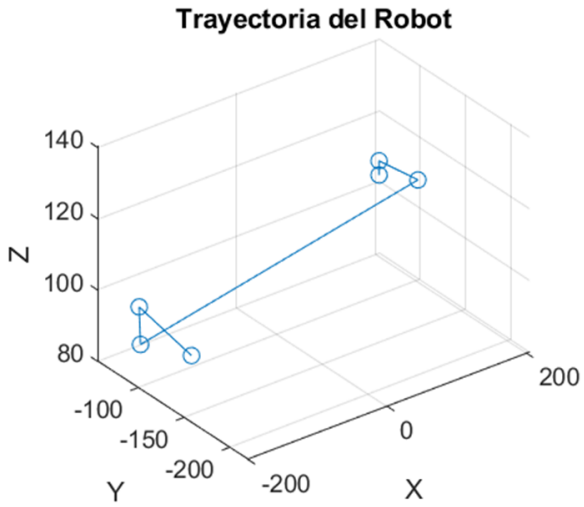

Figura 35: Trayectoria generada a partir de datos de Optitracks

Haber implementado una interfaz con todas las funciones realizadas permite comprobar que el desarrollo de las funciones y simulaciónes fue correcto. Lo que destaca de esta interfaz es que su uso es sencillo y amigable con el usuario. Adicionalmente, se le agregaron condiciones de seguridad a la interfaz que permita que el usuario siga utilizando la interfaz en caso de errores de conexión en la obtención de datos del sistema de captura de movimiento.

CAPÍTULO 9

Conclusiones

- Se logró implementar un modelo cinemático preciso utilizando el estándar Denavit-Hartenberg. Este modelo permitió calcular trayectorias planificadas con un error mínimo, validado mediante pruebas de simulación y resultados consistentes entre posiciones calculadas y ejecutadas.
- El modelo desarrollado en Webots replicó con exactitud el comportamiento físico del MaxArm, validando su funcionalidad en un entorno virtual antes de implementarlo en el control físico. Esto demostró que la simulación es una herramienta efectiva para identificar posibles fallas antes de llevar el modelo al entorno real.
- La interfaz gráfica desarrollada en Matlab facilitó la interacción con el MaxArm, permitiendo la planificación de trayectorias y el control manual en tiempo real mediante comunicación TCP/IP. Esto resultó en un control eficiente y estable, confirmando la viabilidad del sistema inalámbrico.
- El MaxArm se integró exitosamente al ecosistema Robotat, lo que permitió interactuar con datos capturados por OptiTrack y generar trayectorias dinámicas basadas en datos del entorno. Este logro demostró la capacidad del robot para adaptarse a escenarios educativos e industriales complejos.
- Este proyecto evidenció cómo la integración de herramientas de software con hardware robótico se pueden implementar en el control inalámbrico de manipuladores. Los resultados obtenidos destacan su utilidad en aplicaciones educativas e industriales, especialmente en entornos que demandan precisión y adaptabilidad.

CAPÍTULO 10

Recomendaciones

- Cuando se tiene un robot serial paralelo como lo es el manipulador MaxArm, la manera que presenta más facilidad para obtener la cinemática directa e inversa, es realizando relaciones geométricas que tomen en cuenta los ángulos de los motores respecto a la posición del efector final.
- Para poder asegurar la conexión inalámbrica efectiva es recomendable asignarle un IP address propio al dispositivo al relacionarlo con su MAC address.
- En el caso de crear un modelo robótico nuevo en Webots, asegurar que las posiciones de desfase inicial se encuentren orientadas a los lados correctos de los ejes, para asegurar que el controlador maneje el robot corrrectamente.
- Al realizar una interfaz gráfica en Matlab, implementar alertas como seguridad para que el funcionamiento de los botones no falle, así mismo, asegurarse que los datos que se ingresen sean del tipo que se necesita.
- Al momento de crear un marker para la base del manipulador, este al estar al lado del robot puede llegar a tener posiciones que hagan díficil su detección dentro del ecosistema, por lo que es importante asegurar que por lo menos tres de las cámaras puedan percibir el marker. De lo contrario, la matriz que se obtiene para llegar a una posición, será incorrecta.
- Si se desea utilizar una simulación del maipulador en la interfaz de ususrio, se recomienda la opción de crear un modelo de simulación propio y no la herramienta de Peter Corke, debido a que esta no permite visualizar las trayectorias de manera continua dentro de la interfaz, a menos que se desea generar esta visualización como una vista adicional a la que se enceuntra dentro de la interfaz.

CAPÍTULO 11

Bibliografía

[1] J. D. Pellecer Orellana, "Diseño e implementación de un paquete de herramientas de software para controlar inalámbricamente un manipulador serial R17 dentro de un ecosistema basado en captura de movimiento," Tesis, Universidad del Valle de Guatemala, 2022.

[2] C. Perafán Montoya, "Robotat: Un Ecosistema Robótico de Captura de Movimiento y comunicación inalámbrica," Tesis, Universidad del Valle de Guatemala, 2022.

[3] "Tracker: video analysis and modeling tool." (2024), dirección: `https://physlets.org/tracker/`.

[4] E. Pérez Belzuz, "Estudio e implementación de un sistema IoT en un brazo robot y control en TIA Portal," Tesis, 2020.

[5] "Hiwonder MaxArm Open Source Robot Arm Powered by ESP32 Support Python and Arduino Programming Inverse Kinematics Learning." (2022), dirección: `https://www.hiwonder.com/products/maxarm?variant=40008714092631`.

[6] "ESP32 Series: datasheet." (2024), dirección: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`.

[7] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, *Fundamentos de Robótica*. 2007.

[8] G. Martínez, "Los protocolos TCP/IP," en *SEGURIDAD EN REDES IP: Los protocolos TCP/IP*. 2010.

[9] "Webots: open source robot simulator." (1998), dirección: `https://cyberbotics.com/`.