INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Unidad Culhuacán

Ingeniería en Computación

Control basado en navegación inercial para un brazo robótico articulado

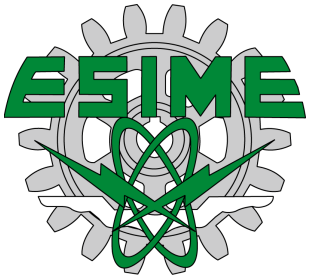
T E S I N A

que para obtener el título de

Ingeniero en Computación

presenta

Nicolas Castañeda David Emmanuel



Asesores:

M. en C. José Antonio Loaiza Brito

Ing. Enrique Cisneros Sedano

Ciudad de México, 16 de enero de 2025

Agradecimientos

Agradezco a mi asesor, José Antonio Loaiza Brito, por permitirme trabajar en un proyecto donde adquirí invaluables habilidades para aplicarlas en el desarrollo de mi carrera, y por guiar el trabajo para alcanzar el objetivo, que fue la titulación.
También agradezco a la empresa Sistemas Eléctricos de Potencia Computarizada (SEDPC), por los recursos económicos y materiales y la asesoría prestada para el desarrollo de este trabajo. En particular, quiero agradecer a los ingenieros Enrique Cisneros Sedano, Marco Emmanuel Toledano Polanco e Israel Flores Romero, quienes fueron responsables directos de mi proyecto y me sugirieron herramientas para llevarlo adelante.

Dedicatoria

A mi incansable madre Fanny, que con su trabajo, su disciplina, sus consejos y su impulso me llevaron por el camino para completar mis estudios.
A mi hermana Mia Belén, la compañera con la que compartí los sinsabores y los triunfos de la vida por todos estos años.
A mi pareja, Ana, que me acompañó hasta el cansancio y me apoyó en todas las visitas a la empresa, durante el desarrollo de este trabajo.
A mis tíos Eder y Víctor, y mi abuela Violeta, que me apoyaron desde el momento en el que decidí estudiar en el Politécnico, y que a pesar de la pandemia, nunca se negaron a hacerlo.
Gracias por guiar mi vida, y ser el motivo de mi superación.

Índice

Lista de tablas

Lista de figuras

Capítulo 1

1.1. Introducción

Los brazos robóticos articulados son sistemas mecánicos con articulaciones rotativas, diseñados para replicar funciones del brazo humano, a través de movimientos de alcance. Suelen tener una pieza en el extremo del robot llamado efector final, como se muestra en la Figura 1.1, que realiza la función del robot en el entorno (soldar, manipular objetos, etc.). Cada eje de movilidad en las articulaciones representa un grado de libertad (GDL); de este modo, si una articulación puede girar de izquierda a derecha, y también de arriba hacia abajo, se dice que tiene dos GDL.

Figura 1.1: Brazo robótico articulado de 2 GDL

Sistemas Eléctricos de Potencia Computarizada (SEPDC) es una empresa mexicana que se dedica a fabricar la serie Kaab de Controladores Lógicos Programables (PLC), computadoras especializadas para la automatización industrial (tienen inmunidad al ruido eléctrico y resistencia a la vibración y al impacto). Cada uno de ellos puede ser operado de forma remota a través de un software llamado SettDev. En la Figura 1.2 se muestra el PLC-Kaab.

Figura 1.2: PLC-Kaab fabricado por SEDPC

Los PLC´s son empleados para el control de sistemas, como lo demuestra la Figura 1.3. Por medio de una versión del software SettDev instalado en el PLC, se monitorea una planta de agua tratada, determinando los niveles de llenado del tanque.

Figura 1.3: Uso del PLCKaab para controlar un sistema de llenado de agua

1.2. Planteamiento del problema

SEDPC necesita un sistema para el control de servomotores para controlar brazos robóticos de diferentes longitudes de 2 DoF, y darle instrucciones al robot por medio del movimiento del brazo de un operador.

Configurar los grados de libertad de un brazo para llegar a una posición deseada es un problema clásico de la robótica llamado cinemática inversa. Existen métodos algebraicos, geométricos e iterativos para resolverlo, sin embargo, consumen una gran cantidad de recursos computacionales, lo que dificulta su uso en el PLC Kaab.

1.3. Objetivo general

Desarrollar e implementar un sistema de control utilizando una Raspberry Pi y sensores inerciales para controlar la posición del efector final de un brazo robótico de 2 GDL.

1.4. Objetivos específicos

- Desarrollar e implementar un programa en C++, utilizando los ángulos de inclinación en los tres ejes obtenidos por los sensores MPU6050, para determinar la posición del sensor en el extremo de la ortesis de brazo.
- Diseñar e implementar un modelo de clusterización para determinar los ángulos de la cinemática inversa.
- Implementar la comunicación entre la Raspberry Pi y el software SettDev por medio de sockets TCP en C++ y C# para guardar y reproducir las instrucciones en la simulación en 3D de un brazo robótico.
- Desarrollar e implementar una interfaz gráfica de usuario por medio del framework Qt para visualizar en una pantalla táctil de 7 pulgadas la cinemática inversa del brazo robótico a controlar.
- Implementar la comunicación entre el software SettDev y el PLC Kaab utilizando el módulo para el control de servomotores instalado en el PLC para enviar y reproducir los ángulos de inclinación en los servomotores posicionales.

1.5. Justificación

La determinación de la posición del efector final por medio de sensores inerciales permitirá que el operador pueda dar instrucciones al robot sin que sea necesaria su especialización en la determinación de la configuración de rotación del robot. Además, la clusterización permitirá almacenar las poses sin la necesidad de recalcularlas para cada posición que se solicite.

Capítulo 2

2.1. Estado del arte

Tabla 2.1: Estado del arte

Título	**Autores**	**Tipo de publicación, lugar y fecha**	**Descripción**
FIKA: A Conformal Geometric Algebra Approach to a Fast Inverse Kinematics Algorithm for an Anthropomorphic Robotic Arm [1] FIKA: Un enfoque de álgebra geométrica conforme para un eficaz algoritmo cinemático inverso para un brazo robótico antropomórfico	Oscar Carbajal-Espinosa; Leobardo Campos-Macías; Miriam Díaz-Rodríguez Instituto Tecnológico y de Estudios Superiores de Monterrey; Intel Corporation; Tecnológico Nacional de México	Artículo  Mexico 2024	Propone un método geométrico iterativo de 3 fases para resolver el problema de la cinemática inversa. Sin embargo, requiere un tiempo de procesamiento de datos inviable.
SK-PSO: A Particle Swarm Optimization Framework with SOM and K-Means for Inverse Kinematics of Manipulators [2] SK-PSO: Un marco de optimización de enjambre de partículas con SOM y K-Means para la cinemática inversa de manipuladores	Fei Liu; Changqin Gao; Lisha Liu Shenzen Polytechnic University	Artículo  China 2024	Propone un enfoque híbrido basado en optimización por enjambre de partículas (PSO), mapas autoorganizados (SOM) y K-means. Sin embargo, la clusterización por K-means requiere conocer el número exacto de clústeres de antemano.

Tabla 2.2: Estado del arte (continuación)

Título	**Autores**	**Tipo de publicación, lugar y fecha**	**Descripción**
Geometric Approach for Inverse Kinematics of the FANUC CRX Collaborative Robot [3] Enfoque geométrico para la cinemática inversa del robot colaborativo FANUC CRX	Manel Abbes; Gérard Poisson University of Orléans	Artículo  Francia 2024	Propone un método geométrico que genera las mejores configuraciones para alcanzar una posición. Sin embargo, utiliza una cantidad considerable de recursos en memoria RAM.

2.2. Marco teórico

Raspberry Pi 3 B+

- Funciona con un sistema operativo basado en la arquitectura ARMv8
- Procesador Broadcom BCM2837 de 1,4 GHz, 1 GB de RAM
- Módulo Wi-Fi de 2,4 y 5 GHz
- 40 pines Entrada/Salida de Propósito General (GPIO)
- Salidas de 3.3 y 5 V, bus I^2C

Figura 2.1: Raspberry Pi 3 B+

Qt

- Optimizada para interfaces gráficas en sistemas portátiles
- Soporte para C++
- Utiliza el lenguaje declarativo QML para programar la interfaz
- Soporta hilos para evitar el bloqueo de recursos

Figura 2.2: Logotipo de Qt Framework

Bus I2C

- Bus serial de comunicación de tipo maestro-esclavo.
- Línea serial de datos bidireccional (SDA) y línea serial de reloj (SCL).
- Espacio de direcciones, cada dirección identifica a un dispositivo conectado al bus.

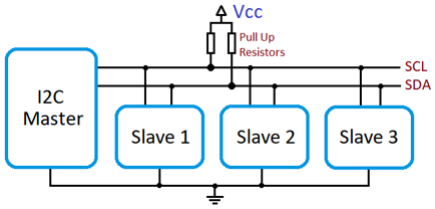


Figura 2.3: Diagrama del bus I2C

MPU-6050

- Giroscopio de vibración de Coriolis y acelerómetro de 3 ejes.
- Procesador de movimiento digital (DMP) que mide la orientación del sensor en tres ejes.
- Buffer FIFO interno.
- Comunicación por medio del bus I^2C de hasta 400 KHz.
- Pin AD0 para cambiar la dirección en el bus. Solo puede tomar las direcciones 104 y 105.

Figura 2.4: Sensor inercial MPU6050

Microservomotor posicional

- Ángulos de entrada de 0° a 180° en valores enteros
- Movimiento bidireccional

Figura 2.5: Microservomotor posicional

2.3. Propuesta de solución

La empresa requirió que las soluciones de la cinemática inversa se guardaran en un archivo, que posteriormente sea reproducido y enviado desde el software SettDev al PLC-Kaab. De este modo, la propuesta de solución se dividió en dos fases. La Figura 2.6 muestra la primera fase, la de captura de datos. En la etapa de Sensado, el sensor MPU-6050 mide la orientación del sensor, y envía los ángulos de inclinación $S(\psi, \theta, \phi)$ calculados en los 3 ejes a la Raspberry Pi. Estos valores son la entrada de la etapa de Localización, en la cual se determina la posición (x, y, z) en el espacio de la muñeca respecto al hombro (quien tiene la posición $(0, 0, 0)$), que será la que deberá alcanzar el efector final. Estas coordenadas sirven como entrada a la etapa de Procesamiento, donde, de acuerdo con la longitud de las articulaciones (L_1, L_2) especificada mediante una interfaz, se calculan los ángulos $(\alpha_1, \alpha_2, \alpha_3)$ que resuelven la cinemática inversa para alcanzar la posición (x, y, z) previamente calculada. Los ángulos de los eslabones se aplican al movimiento de un modelo de brazo robótico en 2D en la interfaz gráfica antes mencionada en la etapa de Visualización, en la que se puede simular el comportamiento que tendría el brazo robótico al aplicar dichos ángulos; todos los ángulos obtenidos se aplican al movimiento de un modelo de brazo robótico en 3D en la etapa de Referencia. Finalmente, los ángulos $(\alpha_1, \alpha_2, \alpha_3)$ son enviados al software SettDev por medio de una trama TCP y almacenados en un archivo con extensión .bin.

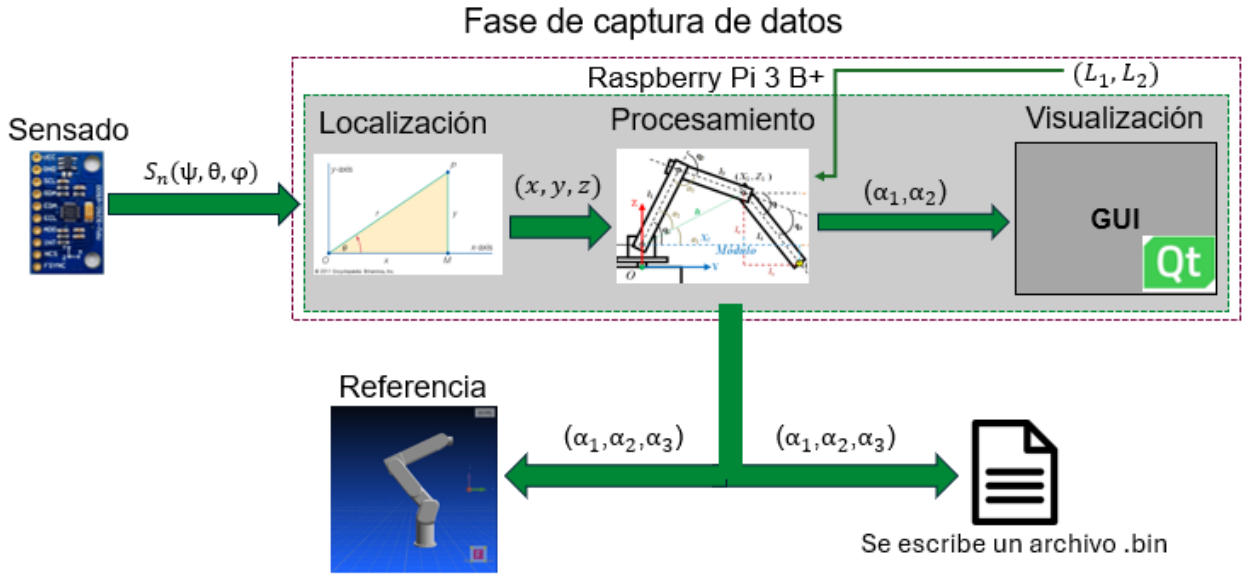


Figura 2.6: Fase de captura de datos

La Figura 2.7 muestra la segunda fase, la de ejecución. El archivo con extensión .bin previamente guardado es leído y transmitido por el software SettDev hacia el PLC-Kaab, quien interpreta los ángulos recibidos y los modula en ancho de pulso para controlar la posición de los servomotores posicionales.

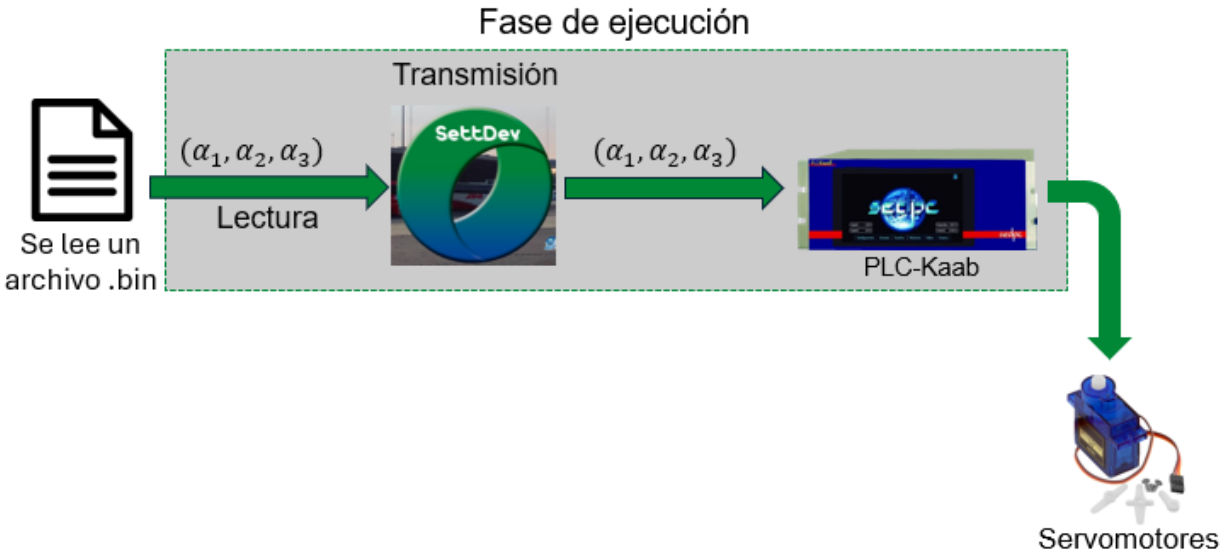


Figura 2.7: Fase de ejecución

Capítulo 3

3.1. Fase de captura de datos

3.1.1. Sensado

La empresa requirió que se utilizara una ortesis de brazo donde se montara el sistema de medición de los sensores inerciales y la Raspberry Pi. La Figura 3.1 muestra la ortesis de brazo utilizada. Para encontrar la posición del efector final, se utilizaron dos sensores MPU-6050; uno se colocó en el codo, mientras que el segundo se colocó en el extremo de la ortesis.

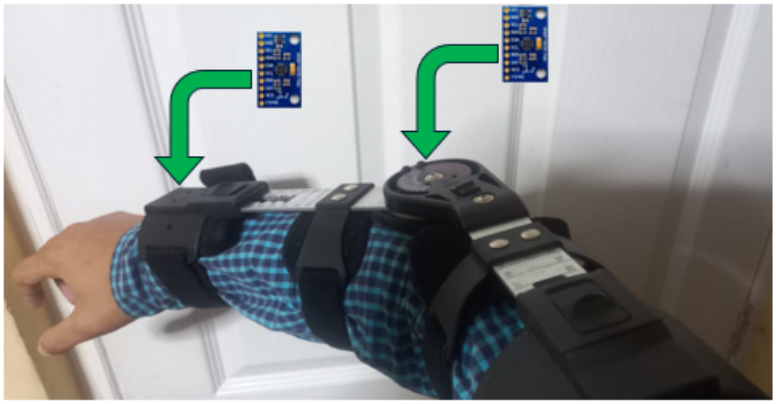


Figura 3.1: Ortesis utilizada para el proyecto

Se realizó la conexión entre los sensores y la Raspberry Pi 3 B+. La Figura 3.2 muestra el diagrama de interconexión. Los pines SDA y SCL de los sensores MPU-6050 se conectaron al bus serial I^2C.

Nótese que tanto las líneas de datos (SDA), como las líneas de reloj (SCL) de los sensores MPU-6050, se conectaron a una única línea SDA o SCL, respectivamente, hacia la Raspberry Pi. Para evitar problemas de sincronización, se eligió la misma frecuencia de reloj de 200 KHz del DMP para ambos sensores. Nótese también que se alimentó al sensor a través de la entrada AD0, en vez de VCC. Esto permite que su dirección en el bus I^2C cambie de 104 a 105. Todos los sensores se alimentaron a través de la salida de voltaje de 3,3 V de la Raspberry Pi.

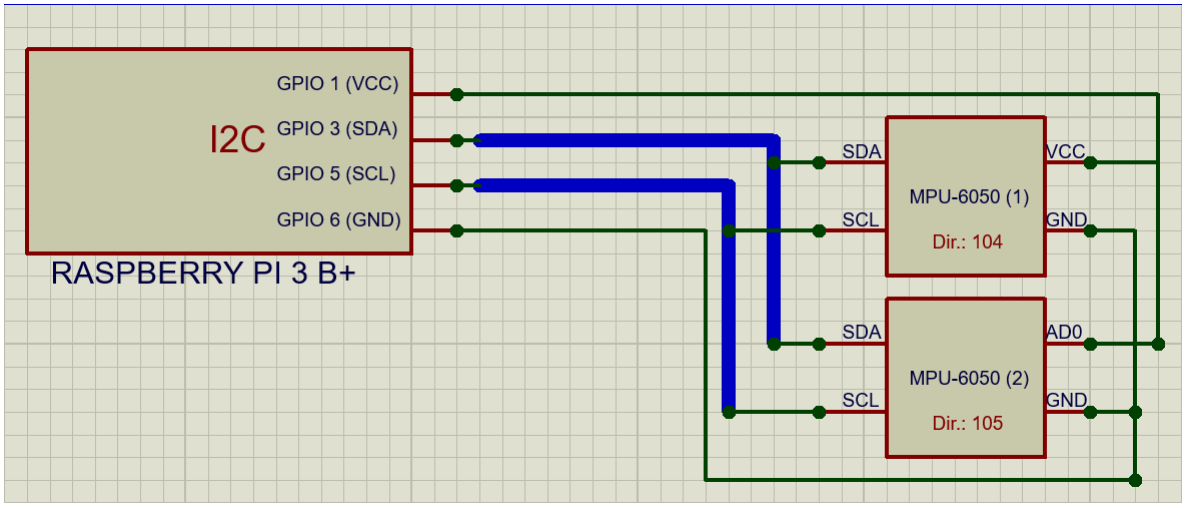


Figura 3.2: Diagrama de interconexión entre la Raspberry Pi y los sensores

Calibración

Los sensores MPU-6050 representan internamente los valores del giroscopio y acelerómetro que tienen incorporados por medio de un capacitor (Figura 3.3), en el que una de las placas, que rodea a la otra, está fija, mientras que la que se encuentra en el centro, puede desplazarse. La capacitancia entre las placas determina el voltaje de salida del sensor [4].

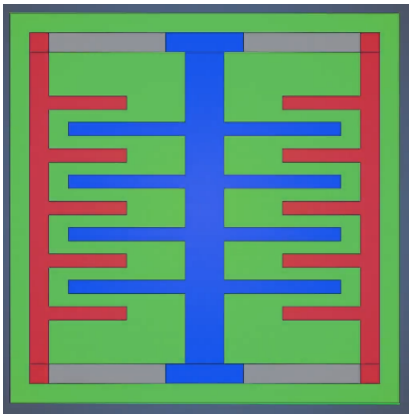


Figura 3.3: Capacitor interno por cada eje del MPU-6050. La primera placa está coloreada de color rojo, y la segunda de color azul

Los sensores inerciales fabricados con tecnología MEMS, como los MPU-6050, necesitan ser calibrados y tener un valor de referencia (offset) con el cual se corrija la orientación medida por el sensor [5]. El diagrama de la Figura 3.4 obtenido del manual [5] muestra el proceso para obtener datos de los sensores MPU-6050; los valores de referencia del acelerómetro y el giroscopio (Gyro and Accel Offset Registers) se aplican a las mediciones obtenidas por el giroscopio y el acelerómetro (Gyro and Accel MEMS), para corregir la mediciones y colocarlas en los registros del sensor (Gyro/Accel Output Registers). Luego, son procesadas por el Procesador de Movimiento Digital (DMP) y el resultado se almacena en un buffer FIFO interno.

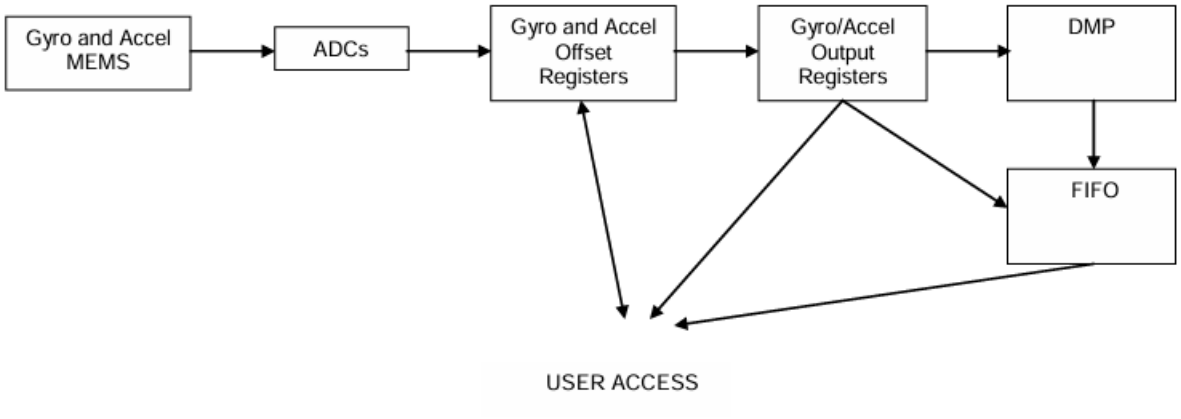


Figura 3.4: Lectura de datos de los sensores MPU

La Figura 3.5 muestra los ejes de desplazamiento y la rotación de los sensores MPU. Se colocaron los sensores MPU en la ortesis de modo que el sentido positivo del eje X quedara hacia el frente del operador (quien tiene colocada la ortesis); de acuerdo con esto, el sentido positivo de rotación en el eje Z se obtiene girando el brazo hacia la izquierda del operador; el sentido positivo de rotación en el eje Y se obtiene girando el brazo hacia abajo; y el sentido positivo de rotación en el eje X se obtiene rotando el brazo hacia la derecha.

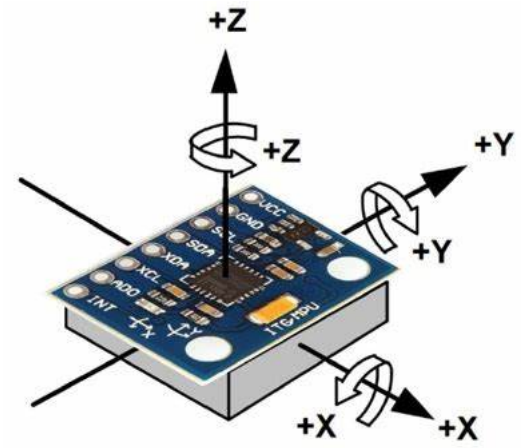


Figura 3.5: Orientación de los ejes y polaridad de rotación de los sensores MPU

Se eligieron los valores de referencia de modo que la salida del sensor en el inicio del proceso de medir la orientación fuera $X = 0, Y = 0, Z = 0$ para el giroscopio y $X = 0, Y = 0, Z = -9.8$ para el acelerómetro, debido a la constante de la aceleración de la gravedad g = -9.8 m/s^2. De acuerdo con lo anterior, al procesar los datos por el DMP, la salida deseada en el inicio del proceso de medir la orientación sería $\theta_x = 0, \theta_y = 0, \theta_z = 0$.

Si el sensor se encuentra en estado de reposo (no existe ninguna fuerza externa que lo mueva), se espera que al leer datos de él, los valores no cambien; en la práctica, estos valores pueden variar debido a interferencias como el ruido externo. De modo que se calcula un valor

medio cuyo error (la diferencia entre la medición del sensor en estado de reposo y el valor medio) sea menor que un error máximo aceptable; con base en el manual [5], se eligió un error máximo de 0.1 $^{\circ}/s$ para el giroscopio, y 0.1 m/s para el acelerómetro.

Para obtener dicho valor medio, se utilizó un control proporcional-integral (PI), en el cual se escribe el valor medido por el sensor en los registros de referencia (offset registers), y se compara dicho valor con la siguiente medición del sensor (con el último offset escrito en los registros de referencia aplicado a la nueva medición), para determinar el error; este proceso termina cuando el error obtenido se encuentra dentro del rango previamente establecido.

La Figura 3.6 muestra el proceso de calibración del sensor para cada eje, donde $X(kT)$ es la medición deseada en un tiempo T para dicho eje del giroscopio y el acelerómetro, excepto el eje Z, por lo ya antes mencionado, $e(kT)$ es la señal de error entre la medición deseada y la medición obtenida, y $Y(kT)$ es la salida de la medición en dicho eje en el mismo instante de tiempo después de aplicar el nuevo offset. El periodo de muestreo es de 2,5 μs, de acuerdo con el manual [5].

Para que la calibración sea adecuada y se obtenga una medición confiable, el sensor debe de encontrarse en el estado de reposo mencionado anteriormente.

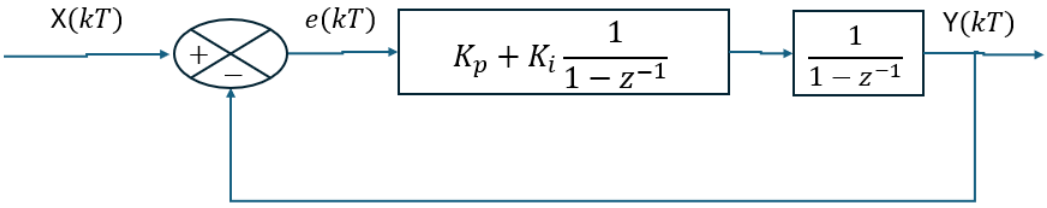


Figura 3.6: Diagrama a bloques del proceso de calibración del sensor

Posteriormente, se debe de cargar el firmware del procesador [6]. Este proceso debe de realizarse cada vez que se encienda el sensor. Aunque no se indique explícitamente, el hecho de que el firmware deba de ser cargado cada vez que se inicie el sensor sugiere que la memoria es volátil. La memoria está formada por 8 bancos, en los que se carga el firmware proporcionado por InvenSense [6].

Medición

El algoritmo utilizado por el DMP para obtener la orientación, no es de dominio público. Por otro lado, InvenSense, el desarrollador del sensor, ofrece un conjunto de bibliotecas para trabajar con el DMP [6].

El DMP representa internamente la orientación por medio de cuaterniones unitarios. Son rápidamente computables, y evitan problemas que se producen al girar más de 90°, como el bloqueo del cardán.

El cuaternión es de la forma:.

$$q = a + b\hat{i} + c\hat{j} + d\hat{k} \tag{3.1}$$

Donde a, b, c, d son las componentes del cuaternión que describe la rotación actual del sensor.

El cuaternión se convirtió a ángulos de Tailt-Bryan [7], que describen la orientación en el espacio del sensor, mediante las siguientes ecuaciones:

$$\psi = \arctan\left(\frac{2(ad + bc)}{1 - 2(b^2 + c^2)}\right) \tag{3.2}$$

$$\theta = \arcsin\left(2(bd - ca)\right) \tag{3.3}$$

$$\phi = \arctan\left(\frac{2(ab + cd)}{1 - 2(c^2 + d^2)}\right) \tag{3.4}$$

Para los casos especiales en los que $\theta = 90$ o $\theta = -90$, es decir, la situación del bloqueo del cardán

$$\phi = 0, \psi = 0 \tag{3.5}$$

Esto se realiza para cada sensor.

3.1.2. Localización

La posición $P(x, y, z)$, en centímetros, del sensor en el extremo de la ortesis, será la posición que debe alcanzar el efector final del brazo robótico. Esta se mide con respecto al hombro, que tiene las coordenadas en el origen $(0, 0, 0)$.

Si $(\psi_1, \theta_1, \phi_1)$ son los ángulos de inclinación obtenidos del sensor colocado en el codo de la ortesis, y $(\psi_2, \theta_2, \phi_2)$ son los ángulos de inclinación obtenidos del sensor colocado en el extremo de la ortesis, las siguientes ecuaciones permiten calcular las coordenadas en el espacio del sensor en el extremo de la ortesis con respecto al hombro [7]:

$$x = L_1 \cos(\psi_1) \cos(\theta_1) + L_2 \cos(\psi_2) \cos(\theta_2) \tag{3.6}$$

$$y = L_1 \operatorname{sen}(\psi_1) \cos(\theta_1) + L_2 \operatorname{sen}(\psi_2) \cos(\theta_2) \tag{3.7}$$

$$z = L_1 \operatorname{sen}(\theta_1) + L_2 \operatorname{sen}(\theta_2) \tag{3.8}$$

Donde L_1 es la distancia desde el hombro hasta el sensor colocado en el codo, y L_2 es la distancia desde el sensor colocado en el codo hasta el sensor colocado en el extremo de la ortesis. La ortesis mide 21 cm en cada articulación, de modo que $L_1 = L_2 = 42$.

El resultado es el que se muestra en la Figura 3.7. Las componentes antes calculadas permiten obtener la resultante, cuya punta es la posición $P(x, y, z)$ calculada del efector final. Este procedimiento se llama cinemática directa; a partir de los ángulos que describen la postura del robot, se determina la posición que alcanza en el espacio.

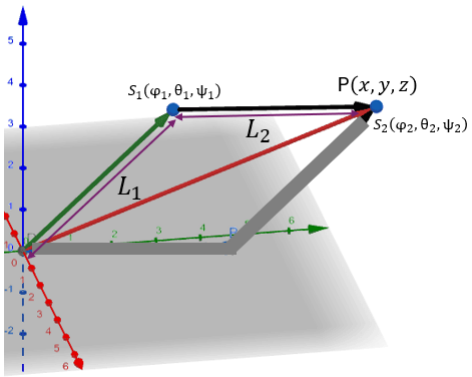


Figura 3.7: Cálculo de la resultante para obtener la posición del efector final

3.1.3. Procesamiento

Se tomaron en cuenta especificaciones sobre el brazo robótico a controlar y los servomotores posicionales a los que se envían las instrucciones.

El tipo de brazo robótico a controlar tiene un diseño antropomórfico, lo que significa que se compone de eslabones unidos por articulaciones entre sí; además es de dos grados de libertad, lo que implica que tiene exactamente dos articulaciones. Como lo que interesa es controlar la posición (y no la orientación), se restringió la cantidad de grados de libertad del brazo robótico a uno en cada articulación; esto significa que en la unión entre la base y el primer eslabón existe un grado de libertad, y el otro grado de libertad se encuentra en la unión entre el primer y segundo eslabón. Además, ambas articulaciones rotan en un plano perpendicular al eje horizontal del robot. Para alcanzar todas las posiciones posibles en el espacio, existe un grado de libertad adicional en la base que rota el brazo robótico alrededor de su eje vertical. La Figura 3.8 muestra un ejemplo de un brazo robótico que cumple con estas especificaciones.

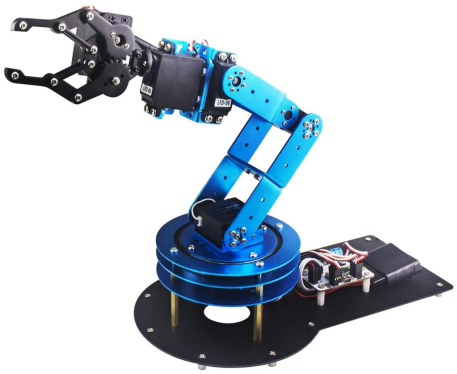


Figura 3.8: Brazo robótico de 2 DoF que cumple las restricciones impuestas

El brazo robótico tiene la base en el suelo; esto significa que se puede mover dentro y en el contorno curvo de la mitad de una esfera con centro en la base del robot y de radio igual a la longitud del robot; debido a que la ortesis utilizada para localizar la posición tiene 42 cm de largo, ésta es la máxima longitud del brazo para el sistema. Los servomotores posicionales solo pueden moverse en ángulos discretos. Esto significa que pueden alcanzar 180 posiciones. Además, muchas posiciones finales son lo suficientemente cercanas entre sí como para considerar que se aproximan a la misma posición. Para determinar la proximidad de las posiciones, se aplicó el siguiente criterio con distancia Euclidiana entre dos posiciones. Si $P_1 = (x_1, y_1, z_1)$ es un punto en el espacio tal que sus coordenadas son valores enteros y $P_2 = (x_2, y_2, z_2)$ es cualquier otro punto:

$$d(P_1, P_2) = |\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}| <= \approx 0,65cm \tag{3.9}$$

Donde la cota establecida es la máxima distancia euclidiana obtenida; esta es la distancia de un punto P_1 al centro de un cubo de 1 cm donde P_1 es un vértice del cubo; de este modo, se considera que aquellos puntos que tienen a lo más 0,65 cm de distancia euclidiana entre ellos llegan a la misma posición. Para el caso en el que una de las coordenadas del punto P_2 sea entera, su distancia euclidiana al punto P_1 más cercano será menor que o igual a 0,5 cm.

Si las posiciones alcanzables son como el punto P_1 antes descrito, esto significa que es posible agrupar ciertas poses para alcanzar una determinada región muy pequeña en el espacio.

Se consideró que los servomotores son colocados inicialmente paralelos al eje horizontal. Podemos notar que muchas posiciones pueden ser alcanzadas por un brazo robótico de una determinada longitud de articulaciones con más de una combinación de ángulos; por ejemplo, si la base tiene 0°, y ambas articulaciones tienen 90° y 0°, respectivamente, se alcanza la misma posición que si la base tiene 180°, y ambas articulaciones tienen 90° y 180°, respectivamente. En particular, si colocamos un plano paralelo al diámetro de la semiesfera que la atraviese, podemos notar las configuraciones "espejo" que alcanzan la misma posición. De modo que se restringió el rango de movilidad de las articulaciones de 0 a 90°; la base se mantuvo de 0 a 360°.

Esto nos da las siguientes restricciones para generar el conjunto de datos:

- $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{Z}$; Los ángulos de rotación de los eslabones son $0 \leq \alpha_1, \alpha_2 \leq 90$, mientras que el ángulo de la base es $0 \leq \alpha_3 \leq 360$

- Las posiciones alcanzables $P = (x, y, z)$ son tales que $x, y, z \in \mathbb{Z}$, y se asocian con un conjunto de ángulos $\alpha_1, \alpha_2, \alpha_3$ que, al ser aplicados para obtener una posición final $P_f(x_f, y_f, z_f)$, sea el punto con menor distancia euclidiana al punto P. La posición P_f se calcula con las siguientes ecuaciones, derivada de la matriz de rotacion de la especificación mencionada del brazo robótico [8]:

$$x = L_1 \cos(\alpha_1) \cdot \cos(\alpha_3) + L_2 \cos(\alpha_1 + \alpha_2) \cdot \cos(\alpha_3) \tag{3.10}$$

$$y = L_1 \cos(\alpha_1) \cdot \sin(\alpha_3) + L_2 \cos(\alpha_1 + \alpha_2) \cdot \sin(\alpha_3) \tag{3.11}$$

$$z = L_1 \sin(\alpha_1) + L_2 \sin(\alpha_1 + \alpha_2) \tag{3.12}$$

Los ángulos con las restricciones anteriores se almacenaron en archivos binarios de acuerdo con la posición especifica. De este modo, cuando se ingrese una posición $P(x, y, z)$ con longitudes L_1, L_2, primero, se determina el punto más cercano $P_i(x_i, y_i, z_i)$ a $P(x, y, z)$ tal que $x_i, y_i, z_i \in \mathbb{Z}$ de los 8 puntos que rodean a $P(x, y, z)$, si se considera esta posición dentro de un cubo (las esquinas $P_i(x_i, y_i, z_i)$ son parte de los puntos como P_1 revisados antes); posteriormente, se busca el archivo binario que se corresponde con dicha posición; si no existe, quiere decir que el brazo no puede llegar a dicha posición. De lo contrario, la combinación de ángulos $\alpha_1, \alpha_2, \alpha_3$ guardados en el archivo binario se toma como la salida de la etapa de Procesamiento.

3.1.4. Visualización

La empresa requirió que se mostrara en una interfaz gráfica un brazo en 2D controlado con los ángulos calculados de la cinemática inversa.

Para desarrollar la aplicación, se utilizó el framework Qt, en el cual se implementaron las etapas de Sensado y Localización en C++. La interfaz se implementó en QML, un lenguaje declarativo que facilitó el desarrollo de la interfaz.

El proceso de lectura de datos del sensor se ejecuta constantemente, lo que puede bloquear los procesos de la interfaz y que ésta deje de responder para el usuario. Para evitar dicho problema, se colocó dicho proceso de lectura de datos en un hilo; éste permite que un subproceso pueda ejecutarse de forma concurrente con otros subprocesos. Para que el hilo de captura de datos pueda comunicar las lecturas a la interfaz, y que éstas puedan ser visualizadas en la interfaz gráfica, se utiliza el paso de mensajes; Qt permite enviar mensajes entre hilos a través de señales. De este modo, también se evitan problemas de concurrencia si ambos hilos intentan acceder a la misma variable donde se almacenan los datos de los sensores.

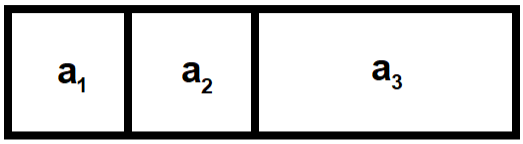
En la Figura 3.9 se observa la interfaz con el brazo en 2D. Puede observarse la posición del efector final determinada por la etapa de localización. La gráfica está graduada. El botón "GRABAR"le indica al sistema que la posición determinada por la etapa de Localización es ingresada a la etapa de Procesamiento; de este modo, se graba una instrucción.

Figura 3.9: Interfaz gráfica de usuario mostrada en la pantalla táctil de la Raspberry Pi

3.1.5. Referencia

Transmisión

Se implementó en C++ y C# la funcionalidad para transmitir los datos entre la Raspberry Pi y el software SettDev, que se ejecuta en el sistema operativo Windows; para este trabajo, se ejecutó en su versión 21.3 para Windows 10. Para la comunicación, se utilizó una trama TCP, cuya carga útil se muestra en la Figura 3.10. Se utilizan dos bytes para el ángulo de la base, y un byte para cada ángulo de los eslabones, siendo 4 bytes en total. La Raspberry posee una dirección IPv4 estática; esta dirección IP, junto con el puerto, que es el mismo en ambos extremos de la comunicación, se utiliza para verificar que el cliente (La Raspberrry Pi) es válido.



Donde:
a_1 es el ángulo de rotación del primer segmento
a_2 es el ángulo de rotación del segundo segmento
a_3 es el ángulo de rotación de la base

Figura 3.10: Campo de datos de la trama TCP entre la Raspberry Pi y SettDev

Simulación

En la Figura 3.11 se observa la interfaz de usuario de SettDev. Éste se compone por módulos; cada módulo es un componente que interactúa con una característica en particular del PLC que se controla desde el software. La interfaz que se muestra, es el módulo desarrollado por personal de SEDPC para el proyecto; éste contiene un modelo en 3D de un brazo robótico, en el que las inclinaciones de cada una de sus articulaciones son modificadas por los controles deslizantes que aparecen en la parte superior del modelo. Se desarrollaron las funcionalidades de captura de datos desde la Raspberry Pi, explicada en la subsección anterior, y transmisión al PLC; esta última se explicará en la fase de ejecución.

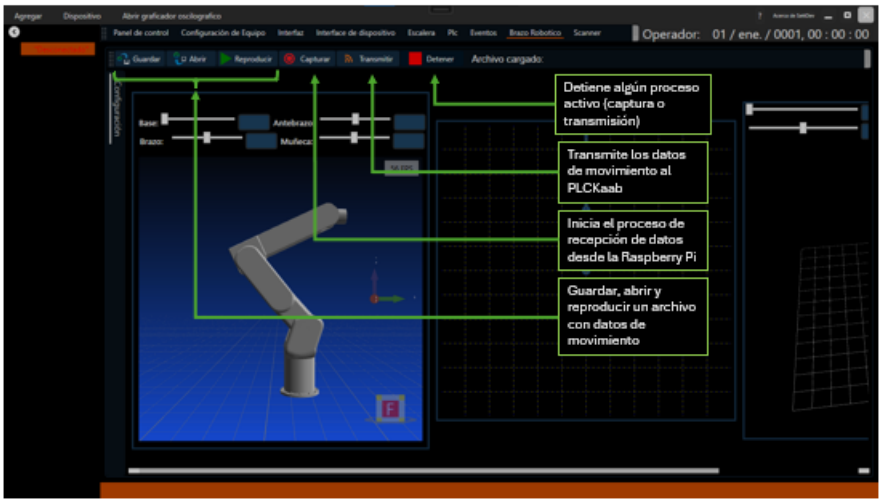


Figura 3.11: Módulo de Brazo Robótico desarrollado por SEDPC

La Figura 3.12 muestra en detalle el brazo en 3D, modelado por personal de SEDPC. Este modelo tiene 2 grados de libertad (además del de la base), y un grado de libertad adicional para el efector final (Muñeca).

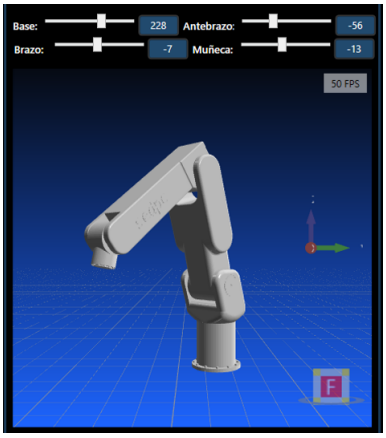


Figura 3.12: Modelo de brazo en 3D utilizado como referencia

3.1.6. Almacenamiento

Los ángulos recibidos en el software SettDev se almacenaron en un archivo binario. Este archivo contiene las instrucciones en conjuntos de 4 bytes, de acuerdo con lo mencionado en la trama utilizada para la transmisión.

3.2. Fase de ejecución

3.2.1. Lectura

Se implementó la funcionalidad en el software SettDev para leer y reproducir un archivo. La Figura 3.13 muestra el proceso de carga de un archivo. El programa permite abrir un cuadro de diálogo para seleccionar un archivo del equipo, de tipo .bin. Cuando se da clic al botón "Abrir" y se cierra el archivo, el programa verifica la extensión del archivo. Posteriormente, comienza a leer los datos del archivo.

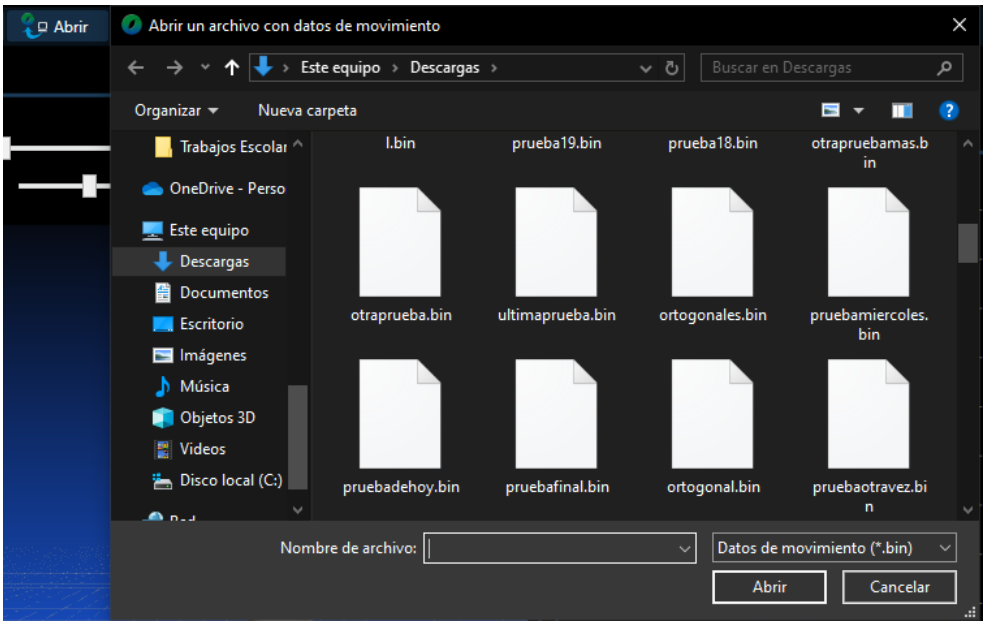


Figura 3.13: Módulo de carga de archivo implementado en SettDev

3.2.2. Transmisión

Se desarrolló e implementó la comunicación entre el software SettDev y el PLC para transmitir los datos leídos desde el archivo. El PLC-Kaab de SEDPC se controla y comunica utilizando datagramas; éstos se envían por medio del protocolo UDP, y tienen un formato de trama específico, que es el que se muestra en la Figura 3.14 obtenida de SEDPC. La trama comienza con un encabezado (Header) que siempre contiene el valor 8A, seguido de algunas indicaciones para el PLC, como el proceso Fuente del que provienen los datos, el proceso de Destino, e indicaciones internas (Internal Indications). Luego, se indica un comando (Command), un subcomando (Subcommand), la indicación de algún Error producido, y la longitud n de los datos por enviar (Largo). Los siguientes n bytes contienen la información a enviar (Data). Por último, se calcula un Checksum como medio de comprobación de errores; si se produce algún error, el PLC no reproduce el datagrama enviado. Se utilizan identificadores

con valores numéricos para indicar la Fuente o Destino (el proceso) que envía o recibe los datos, así como el comando y subcomando por ejecutar.

Header	Fuente	Destino	Internal Indications	Command	Subcommand	Error	Largo	Data	Checksum
8A	**80**	**51**	**00**	**01**	**03**	**00**	**08-00**	**12-34-56-78-9A-BC-DE-F0**	**00-00**
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes	1-243 bytes	2 bytes

Figura 3.14: Formato de trama utilizado por el PLC-Kaab

Para enviar los datos al PLC-Kaab, se tomó como referencia un módulo para el control de servomotores desarrollado por SEDPC que se encuentra instalado en el PLC-Kaab. La trama generada para enviar los ángulos de inclinación al PLC se diseñó considerando la trama de este módulo, en el que el campo de Datos contiene 1 byte para representar el número de motor a controlar, y 4 bytes en formato Little Endian que representan el ángulo enviado a dicho motor. Entre cada instrucción el sistema espera un segundo, para permitir que el brazo robótico procese las instrucciones.

Capítulo 4

4.1. Pruebas

Para medir la precisión del sistema, se aplicó la métrica de la desviación de la posición final de la norma ISO 9283:1998 [9], que calcula la distancia euclidiana entre la posición deseada (x_1, y_1, z_1) obtenida de la etapa de Localización y la posición alcanzada (x_2, y_2, z_2), de acuerdo con la ecuación:

$$d = |\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}| \tag{4.1}$$

Y se tomó el máximo valor.

Para medir la calidad del software, se aplicó la norma ISO 25000 [10]. Esta se compone por 7 criterios: Funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad. La norma no especifica un método para medir cada criterio, sino un marco de factores a tomar en cuenta para tener un desempeño general del software. Para poder realizar la comparación con el estado del arte, se tomó en cuenta el criterio de la eficiencia.

El equipo utilizado para generar las combinaciones de posiciones y combinaciones de ángulos de rotación posee las siguientes características:

- Procesador Intel Core i5 de 2,4 GHz de 4 núcleos
- 8 GB de RAM
- 256 GB de SSD
- Sistema operativo Windows 10

4.2. Resultados

El resultado de la métrica de la desviación de la posición final, tomando el máximo valor con $L = 21cm$, es de 0,65 cm. Dado que no hay un estándar sobre un umbral de precisión

para una determinada aplicación, esto último depende de las especificaciones establecidas en el momento de integrarlo en algún sistema.

La comparación de esta precisión con el estado del arte se muestra en la Tabla 4.1. Cabe resaltar que los trabajos del estado del arte se aplicaron con ángulos no enteros que no son alcanzables por servomotores posicionales, de modo que, para la aplicación específica del robot, su precisión puede verse sesgada.

Sistema implementado (cm)	**Método iterativo (cm)**	**Método por clusterización K-Means (cm)**	**Método geométrico (cm)**
0,65 cm	0,36 cm	0,45 cm	0,12 cm

Tabla 4.1: Resultados de la prueba de precisión

Los resultados de la prueba de eficiencia se muestran a continuación.

ISO	**Sistema implementado**	**Método iterativo**	**Método por clusterización K-Means**	**Método geométrico**
Tiempo promedio de generación de los datos / entrenamiento (s)	360 s	No aplica	330 s	No aplica
Complejidad temporal (O)	$O(n^2)/O(n)$	$O(n^2)$	$O(n^3)$	$O(n^2)$
Tiempo promedio del cálculo de la cinemática inversa (ms)	0,002	1,2	8	8
Complejidad espacial (O)	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n^2)$
Tamaño de los datos almacenados (MB)	2,34	No aplica	No determinado	No aplica

Tabla 4.2: Resultados de la prueba de eficiencia

4.3. Conclusiones

Finalmente, se concluye que el modelo obtenido, pese a requerir una cantidad considerable de tiempo para la generación de datos, posee un tiempo de ejecución muy reducido, lo que no compromete los recursos disponibles de la Raspberry Pi. Además, el tamaño de los archivos generados no crece exponencialmente conforme varían las longitudes de las articulaciones. Incluir

más grados de libertad para determinar la posición en el modelo aumenta exponencialmente el tiempo para la generación de datos, lo que requiere una estrategia de optimización si se quiere llevar a dicha aplicación. La precisión del modelo se establece con una cota máxima desde el principio; conforme se disminuya esta cota, se aumentará la aproximación, lo que implica que el tamaño de los clústeres aumentará; no obstante cada clúster mantendrá su tamaño original.

4.4. Trabajo a futuro

Como trabajo a futuro, se tiene lo siguiente:

- Incorporar el sistema dentro del PLC-Kaab
- Implementar un editor de instrucciones para determinar el tiempo de espera entre cada instrucción
- Configurar el sistema para brazos robóticos de 3 DoF
- Ampliar el sistema para controlar brazos robóticos de longitudes de hasta 1 m
- Agregar al sistema la determinación de la orientación del brazo robótico

Referencias

[1] M. D.-R. Oscar Carbajal-Espinosa, Leobardo Campos-Macías, "Fika: A conformal geometric algebra approach to a fast inverse kinematics algorithm for an anthropomorphic robotic arm," *Machines*, vol. 12, no. 78, pp. 1–6, 2024. [Online]. Available: https://www.mdpi.com/2075-1702/12/1/78

[2] L. L. Fei Liu, Changqin Gao, "Sk-pso: A particle swarm optimization framework with som and k-means for inverse kinematics of manipulators," *Symmetry*, vol. 16, 2024. [Online]. Available: https://www.mdpi.com/2073-8994/16/12/1667

[3] G. P. Manel Abbes, "Geometric approach for inverse kinematics of the fanuc crx collaborative robot," *Robotics*, vol. 91, 13. [Online]. Available: https://www.mdpi.com/2218-6581/13/6/91

[4] B. A. G. Seyyedeh Shirin Saberhosseini, "Design and simulation of a variable mems capacitor for tunable hmsiw resonator," *IET Circuits, Devices and Systems*, 2020. [Online]. Available: https://doi.org/10.1049/iet-cds.2019.0511

[5] Invensense, *MPU Hardware Offset Registers Application Note*, 1st ed., Invensense Corporation, San Jose, USA, 2014.

[6] ——, *Motion Driver 6.12 – User Guide*, 1st ed., Invensense Corporation, Sunnyvale, USA, 2015.

[7] MathWorks, "Spatial representation: Coordinate systems and conventions," 2025, consultado el: 20 de septiembre de 2024. [Online]. Available: https://la.mathworks.com/help/fusion/gs/spatial-representation-coordinate-systems-and-conventions.html

[8] H. R. Leyva, "Robotica 2. modelado cinemática de robots," 2025, consultado el: 20 de septiembre de 2024. [Online]. Available: https://www.utm.mx/ hugo/robot/Robot2.pdf

[9] International Organization for Standardization, "Iso 9283:1998 - manipulating industrial robots — performance criteria and related test methods," 1998, accessed: 2025-01-15. [Online]. Available: https://www.iso.org/obp/ui/iso:std:iso:9283:ed-2:v1:en

[10] ISO 25000, "Iso 25000 series - software engineering — software product quality requirements and evaluation (square)," n.d., accessed: 2025-01-15. [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards