INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Unidad Culhuacán

Ingeniería en Computación

Control inteligente basado en navegación inercial para un brazo robótico articulado

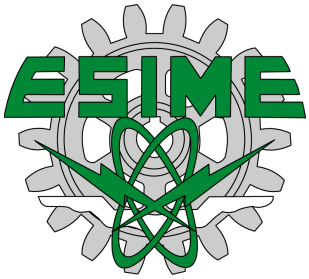
T E S I N A

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 2024

Índice

Lista de tablas

Lista de figuras

Capítulo 1

Anteproyecto

1.1. Introducción

Los brazos robóticos articulados son sistemas mecánicos con articulaciones rotativas, diseñados para replicar funciones del brazo humano, incluyendo movimientos de rotación y 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 unión en las articulaciones representa un grado de libertad (DoF).

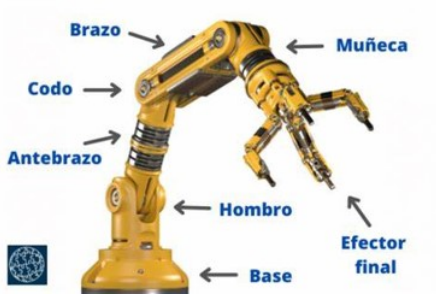


Figura 1.1: Partes de un brazo robótico articulado de 3 DoF

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 pueden ser empleados para el control de sistemas críticos, como lo demuestra la Figura 1.3. El termistor mide la temperatura en el ambiente; este valor es enviado al PLC, quien toma una decisión basado en dicho valor para controlar la apertura de una válvula por medio de un motor, dejando pasar cierta cantidad de líquido a través de la tubería.

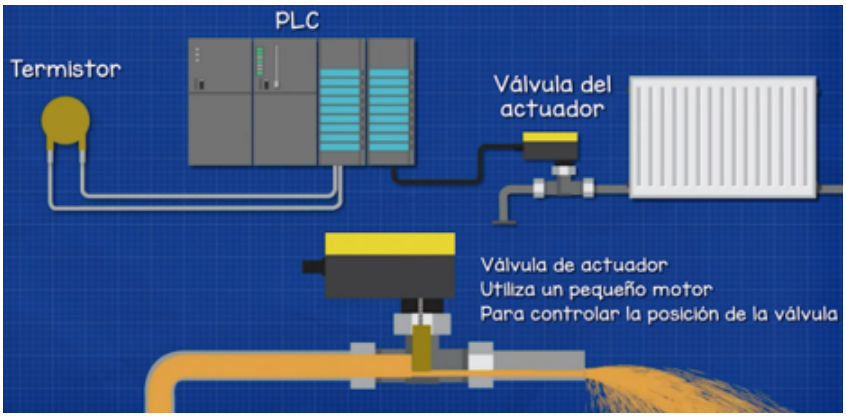


Figura 1.3: Uso de un PLC para controlar un sistema crítico

1.2. Planteamiento del problema

Los brazos robóticos se han introducido rápidamente en la industria, sin embargo, aún existen desafíos para posicionar de manera precisa el brazo robótico que dificultan su introducción en industrias donde el manejo de las materias primas es delicado, sobre todo cuando dicho posicionamiento tiene que realizarse de manera autónoma.

Ubicar las articulaciones 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, dependen de la cantidad de grados de libertad del robot, además de que consumen una gran cantidad de recursos computacionales, lo que dificulta su uso en un sistema crítico.

1.3. Objetivo general

Desarrollar e implementar un sistema crítico de control, por medio de una Raspberry Pi, sensores inerciales y una red neuronal entrenada con aprendizaje no supervisado, para controlar la posición del efector final de un brazo robótico.

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.
- Implementar la comunicación entre la Raspberry Pi y el software SettDev por medio de sockets TCP en C++ y C# para enviar los ángulos de inclinación calculados al software, para poder guardar los datos y permitir reproducirlos en la simulación en 3D de un brazo robótico.
- Desarrollar e implementar una interfaz gráfica de usuario utilizando una pantalla táctil por medio del framework Qt para visualizar la cinemática inversa del brazo robótico a controlar.
- Desarrollar e implementar en SettDev la comunicación entre el módulo del brazo robótico en 3D y el PLC por medio de sockets UDP en C# para enviar los ángulos de inclinación al controlador y reproducir los ángulos de inclinación en los servomotores posicionales.
- Implementar un sistema de inferencia neuro-difuso adaptativo (red neuronal ANFIS) utilizando C++ para resolver el problema de la cinemática inversa del brazo.
- Desarrollar e implementar un algoritmo de aprendizaje no supervisado utilizando C++ para entrenar la red neuronal.

1.5. Justificación

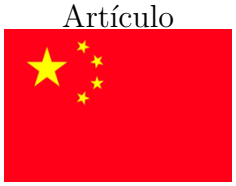
El sistema permitirá implementar un nuevo método para resolver la cinemática directa de forma trigonométrica a través de la navegación inercial, además de implementar un método que no dependa de la cantidad de grados de libertad del robot. Asimismo, será una aplicación práctica de las investigaciones previas a este trabajo sobre redes neuronales para el control de un brazo robótico.

1.6. Estado del arte

Tabla 1.1: 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 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 inaceptable en un sistema crítico.
Implementation of singularity-free inverse kinematics for humanoid robotic arm using Bayesian optimized deep neural network. Implementación de cinemática inversa sin singularidad para brazo robótico humanoide utilizando una red neuronal profunda optimizada por métodos Bayesianos.	Omur Aydogmus; Gullu Boztas Firat University	Artículo  Turquía 2024	Utiliza una red neuronal basada en aprendizaje profundo para resolver la cinemática inversa en una simulación. Sin embargo, el proyecto no se llevó a una aplicación práctica.

Tabla 1.2: Estado del arte (continuación)

Título	**Autores**	**Tipo de publicación, lugar y fecha**	**Descripción**
Inverse kinematics solution and control method of 6-degree-of-freedom manipulator based on deep reinforcement learning. Solución de cinemática inversa y método de control de un manipulador de 6 grados de libertad basado en aprendizaje de refuerzo profundo.	Chengyi Zhao; Yimin Wei; Junfeng Xiao; Yong Sun; Dongxing Zhang; Qiuquan Guo; Jun Yang University of Electronic Science and Technology of China	Artículo  China 2024	Propone un algoritmo de aprendizaje por refuerzo que calcula la distancia entre el efector final y la posición deseada. Sin embargo, se volverá ineficaz cuando se adapte a brazos de diferente longitud.

1.7. Marco teórico

Raspberry Pi 3 B+

- Funciona con un sistema operativo basado en la arquitectura ARM.
- Módulo Wi-Fi de banda dual de 2,4 y 5 GHz.
- 40 pines Entrada/Salida de Propósito General (GPIO)
- Salidas de 3.3 y 5 V, buses I^2C, SPI

Figura 1.4: Raspberry Pi 3 B+

Qt

- Optimizada para aplicaciones con interfaces gráficas en sistemas embebidos.
- Soporte para C++
- Utiliza el lenguaje declarativo QML para programar la interfaz

Figura 1.5: 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.

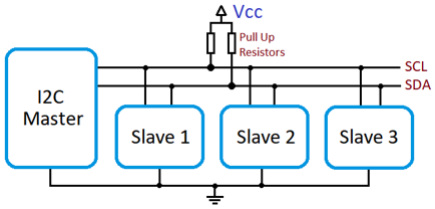


Figura 1.6: Diagrama del bus I2C

Bus SPI

- Bus serial de comunicación de tipo maestro-esclavo.
- Líneas de entrada al esclavo (MOSI) y salida al esclavo (MISO), línea de reloj (SCLK).
- Línea de selección del chip (SS).

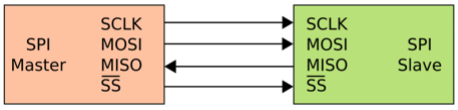


Figura 1.7: Diagrama del bus SPI

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.
- Filtro paso bajo.
- 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 1.8: Sensor inercial MPU6050

MPU-9250

- 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.
- Filtro paso bajo.
- Comunicación por medio del bus SPI de hasta 10 MHz.

Figura 1.9: Sensor inercial MPU9250

Microservomotor posicional

- Ángulo de entrada de 0° a 180°
- Utiliza señales de modulación de ancho de pulso (PWM)
- Movimiento bidireccional

Figura 1.10: Microservomotor posicional

1.8. 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 1.11 muestra la primera fase, la de captura de datos. En la etapa de Sensado, los sensores MPU-6050 y MPU-9250 miden la orientación del sensor, y cada uno envía los ángulos de inclinación $S_n(\theta_x, \theta_y, \theta_z)$ (Siendo n el número de sensor) 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 (la posición $(0, 0, 0)$), que será la que deberá alcanzar el efector final. Estas coordenadas sirven como entrada a la red neuronal ANFIS en la etapa de Procesamiento, donde, de acuerdo con la cantidad de grados de libertad y la longitud de las articulaciones, se calculan los ángulos $(\alpha_1, \alpha_2, ..., \alpha_k)$ (donde k es la cantidad de grados de libertad) que resuelven la cinemática inversa para alcanzar la posición (x, y, z) previamente calculada. Estos ángulos se aplican al movimiento de un modelo de brazo robótico en 2D en una interfaz gráfica en la etapa de Visualización, en el que se puede simular el comportamiento que tendría el brazo robótico al aplicar dichos ángulos. Asímismo, todos los ángulos calculados se envían al software SettDev, en el cual los ángulos $S_n(\theta_x, \theta_y, \theta_z)$ se aplican al movimiento de un modelo de brazo robótico en 3D en la etapa de Referencia, y los ángulos $(\alpha_1, \alpha_2, ..., \alpha_k)$ son almacenados en un archivo con extensión .bin.

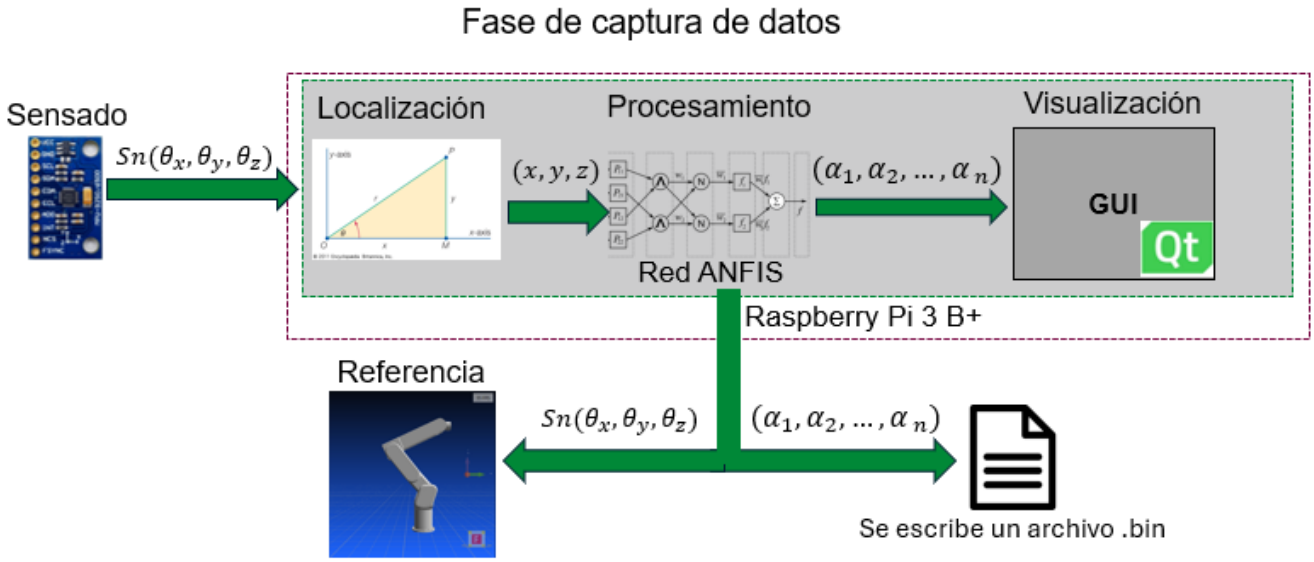


Figura 1.11: Fase de captura de datos

La Figura 1.12 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.

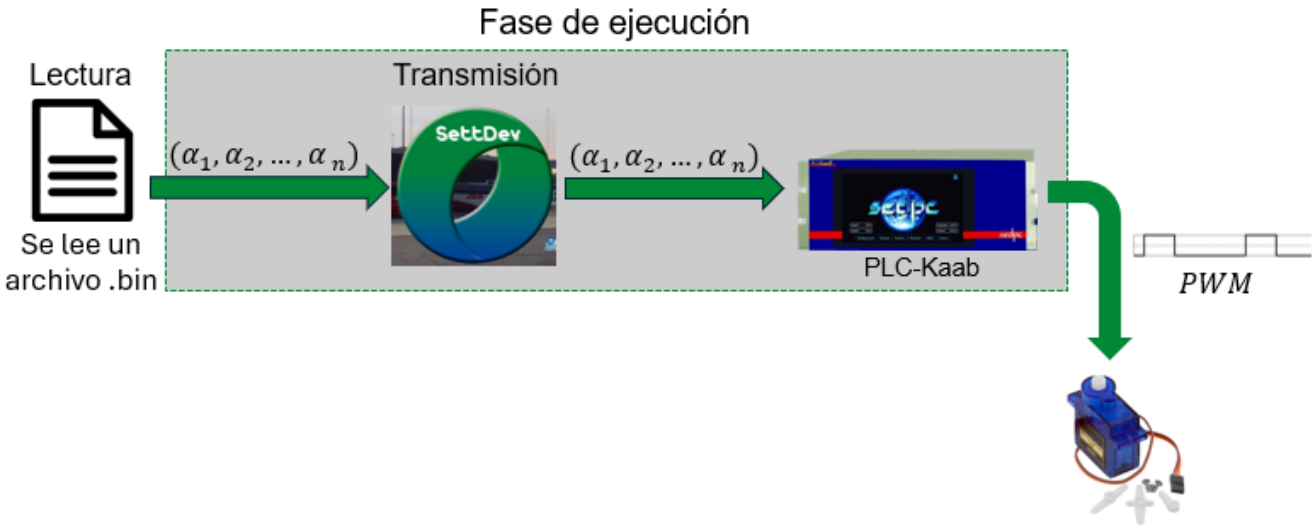


Figura 1.12: Fase de ejecución

Capítulo 2

Desarrollo del proyecto

2.1. Fase de captura de datos

2.1.1. 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 2.1 muestra la ortesis de brazo utilizada. Para encontrar la posición del efector final, se utilizaron los dos sensores MPU-6050; uno se colocó en el codo, mientras que el segundo se colocó en el extremo de la ortesis. El sensor MPU-9250 se colocó en la parte superior de la mano, para controlar el movimiento del efector final.

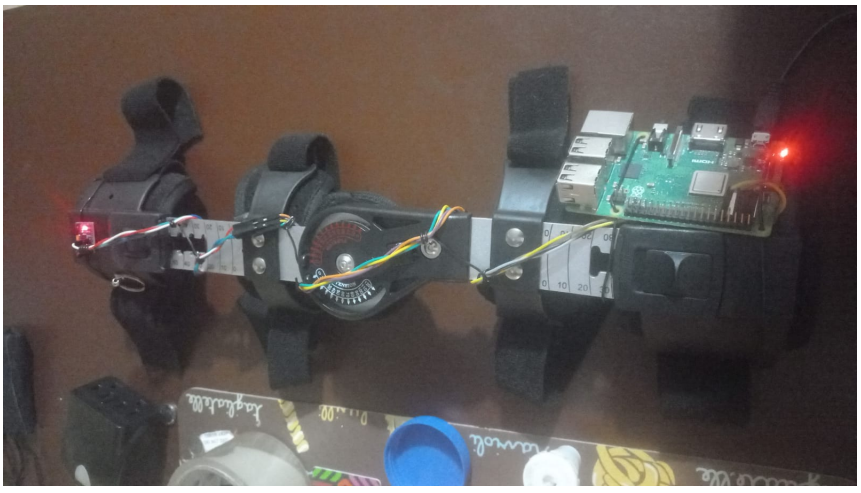


Figura 2.1: Ortesis utilizada para el proyecto

Se realizó la conexión entre los sensores y la Raspberry Pi 3 B+. La Figura 2.2 muestra el diagrama de interconexión. Los pines SDA y SCL de los sensores MPU-6050 se conectaron al bus serial I^2C, mientras que los pines MISO, MOSI y SCK del sensor MPU-9250 se conectaron al bus serial SPI.

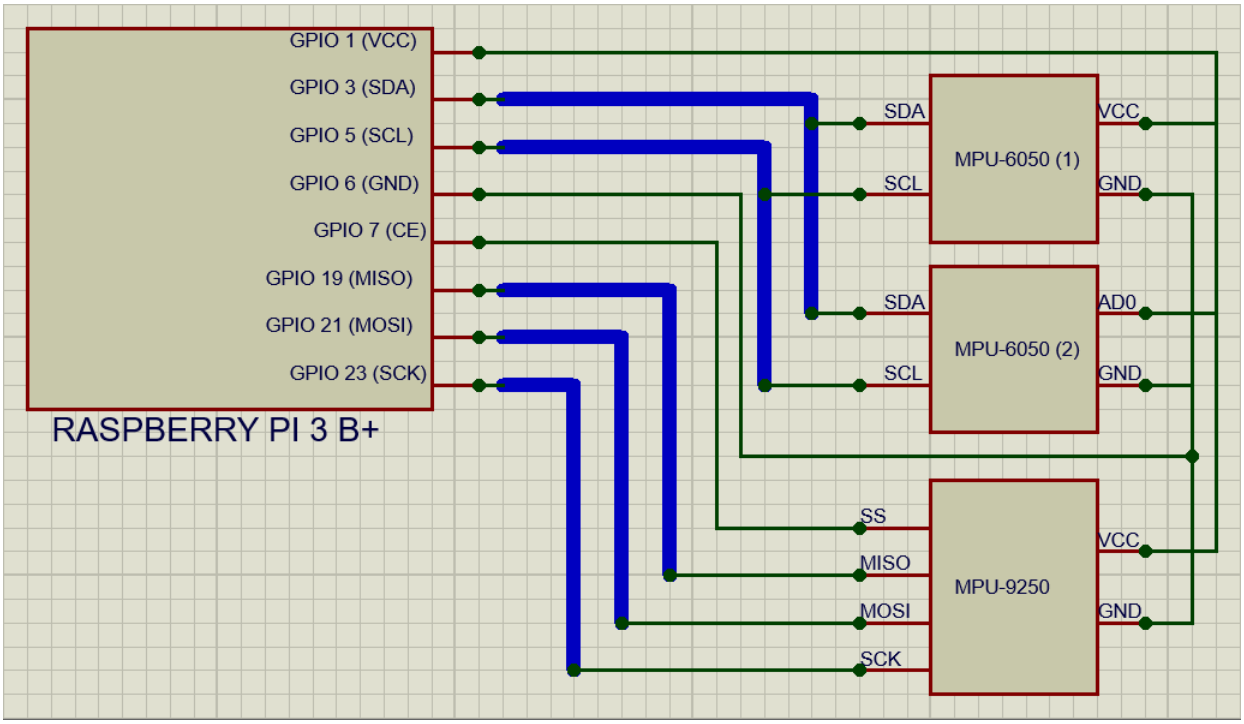


Figura 2.2: Diagrama de interconexión entre la Raspberry Pi y los sensores

La línea SS del sensor MPU-9250 se conectó a la línea Chip Enable (CE) 1 de la Raspberry Pi. 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 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.

En lo que resta del documento, cuando se haga referencia a los sensores MPU-6050 y MPU-9250 en conjunto, se utilizará el prefijo MPU; cuando se haga referencia solo a uno de ellos, se hará con su nombre completo.

Calibración

Los sensores inerciales fabricados con tecnología MEMS, como los MPU, necesitan ser calibrados y tener un valor de referencia (offset) con el cual se corrija la orientación medida por el sensor [1]. El diagrama de la Figura 2.3 obtenido del manual [1] muestra el proceso para obtener datos de los sensores MPU; 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.

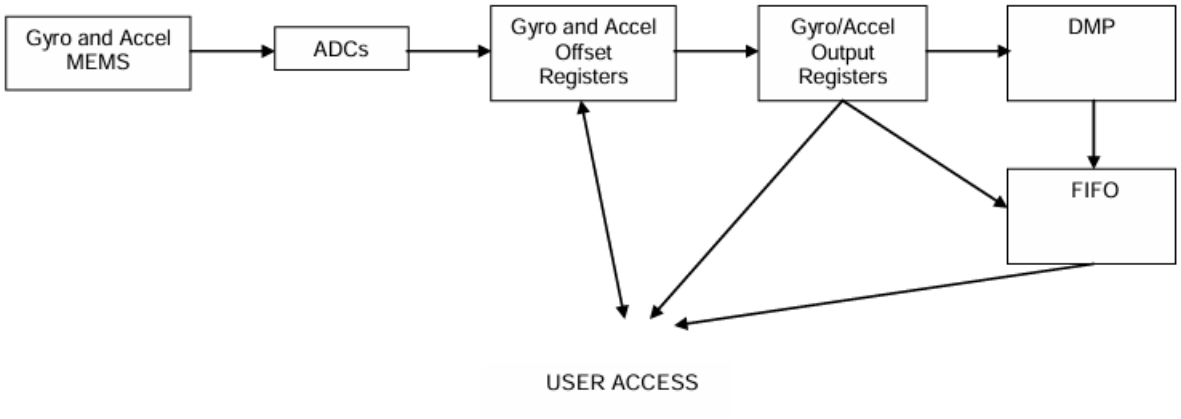


Figura 2.3: Lectura de datos de los sensores MPU

La Figura 2.4 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.

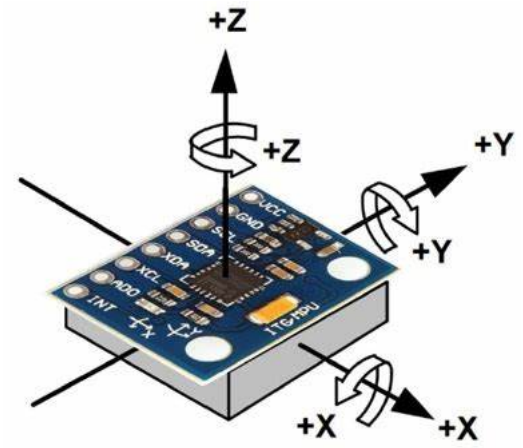


Figura 2.4: Orientación de los ejes y polaridad de rotación de los sensores MPU

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 la experiencia, 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 2.5 muestra el proceso de calibración del sensor. A cada medición se le aplicó el control PI para corregir el error. Para permitir que se establezca un valor apropiado, este proceso se repite 600 veces, un valor elegido basado en la experiencia. Después de que termina el ciclo, se compara el siguiente valor del sensor con los valores de referencia en los registros para determinar el error. Si éste es mayor que el error máximo aceptado, se repite el proceso.

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

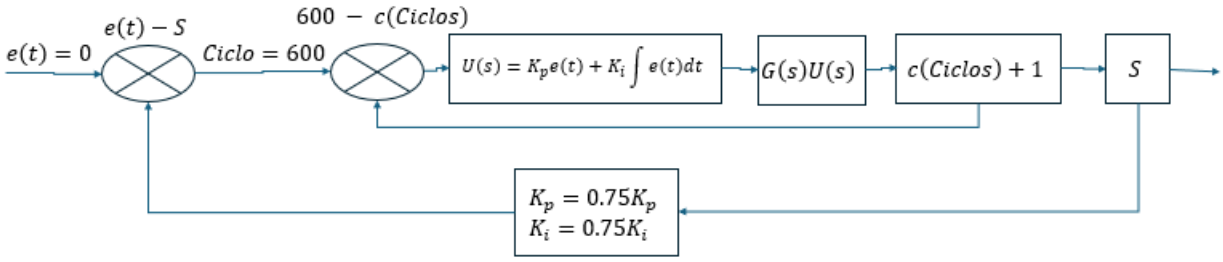


Figura 2.5: Diagrama a bloques del proceso de calibración del sensor

Posteriormente, se debe de cargar el firmware del procesador [2]. 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 que contiene el firmware del sensor es una memoria volátil. La memoria está formada por 8 bancos, en los que se carga el firmware proporcionado por InvenSense [2].

Después de esto, se habilita el DMP y el buffer FIFO escribiendo el valor 1 en el bit FIFO_EN (Bit 6) del registro 106 indicado en la Figura 2.6 para la lectura de datos.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6A	106	-	FIFO_EN	I2C_MST _EN	I2C_IF _DIS	-	FIFO _RESET	I2C_MST _RESET	SIG_COND _RESET

Figura 2.6: Registro User Control del MPU, donde se encuentra el bit (Bit6) para activar el buffer FIFO

Medición

El algoritmo utilizado por el DMP para obtener la orientación, no es de dominio público; en el Capítulo 4, se describe el posible algoritmo utilizado por el DMP. Por otro lado, InvenSense, el desarrollador del sensor, ofrece un conjunto de bibliotecas para trabajar con el DMP [2].

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:.

$$q = a + b\hat{i} + c\hat{j} + d\hat{k} \tag{2.1}$$

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

Para poder obtener la rotación del sensor, se utilizó la formula que obtiene las nuevas componentes de un vector si se le aplica una rotación descrita por un cuaternión, la cual es la siguiente:

$$v' = q \cdot v \cdot q* \tag{2.2}$$

Donde v' es el nuevo vector con la rotación aplicada, y $q*$ es el cuaternión conjugado, de la forma:

$$q* = a - b\hat{i} - c\hat{j} - d\hat{k} \tag{2.3}$$

De acuerdo con los valores de referencia definidos para el acelerómetro ($X = 0, Y = 0, Z = -9.8$), el vector de gravedad es $g = (0, 0, -1)$.

Se sustituyó en la ecuación 2.2 v por el vector de gravedad. Al resolver para v', se obtuvieron las siguientes ecuaciones para obtener cada componente:

$$v'_x = 2 \cdot (x \cdot z - w \cdot y) \tag{2.4}$$

$$v'_y = 2 \cdot (w \cdot x + y \cdot z) \tag{2.5}$$

$$v'_z = w^2 - x^2 - y^2 + z^2 \tag{2.6}$$

Para obtener los ángulos entre el vector v' y los ejes x, y, z definidos cuando se calibró el sensor (es decir, aquella orientación del sensor en la que la salida del DMP es $\theta_x = 0, \theta_y = 0, \theta_z = 0$), se utilizaron las siguientes ecuaciones:

$$\theta_x = \tan^{-1} \frac{v'_y}{v'_z} \tag{2.7}$$

$$\theta_y = \tan^{-1} \frac{v'_x}{\sqrt{v'^2_y + v'^2_z}} \tag{2.8}$$

$$\theta_z = \tan^{-1} \frac{q_x \cdot q_y + q_w \cdot q_z}{1 - 2 \cdot (q_x^2 + q_y^2)} \tag{2.9}$$

Si el sensor se encuentra boca abajo (es decir, la componente Z del vector de gravedad apunta en el sentido positivo del eje Z), es necesario invertir el sentido de la inclinación en el eje Y. Si el valor de la inclinación en el eje Y es positivo, el valor se corrige a $\pi - v'_y$; si es negativo, el valor se ajusta a $-\pi - v'_y$.

Los ángulos obtenidos θ_x, θ_y, θ_z, son la orientación del sensor.

2.1.2. Localización

La posición en (x, y, z) 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 $(\theta_{x_1}, \theta_{y_1}, \theta_{z_1})$ son los ángulos de inclinación obtenidos del sensor colocado en el codo de la ortesis, y $(\theta_{x_2}, \theta_{y_2}, \theta_{z_2})$ 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:

$$x = L_1 \cdot \cos\theta_{x_1} + L_2 \cdot \cos\theta_{x_2} \tag{2.10}$$

$$y = L_1 \cdot \cos\theta_{y_1} + L_2 \cdot \cos\theta_{y_2} \tag{2.11}$$

$$z = L_1 \cdot \cos\theta_{z_1} + L_2 \cdot \cos\theta_{z_2} \tag{2.12}$$

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.

El resultado es el que se muestra en la Figura 2.7. Las componentes antes calculadas permiten obtener la resultante, cuya punta es la posición 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.

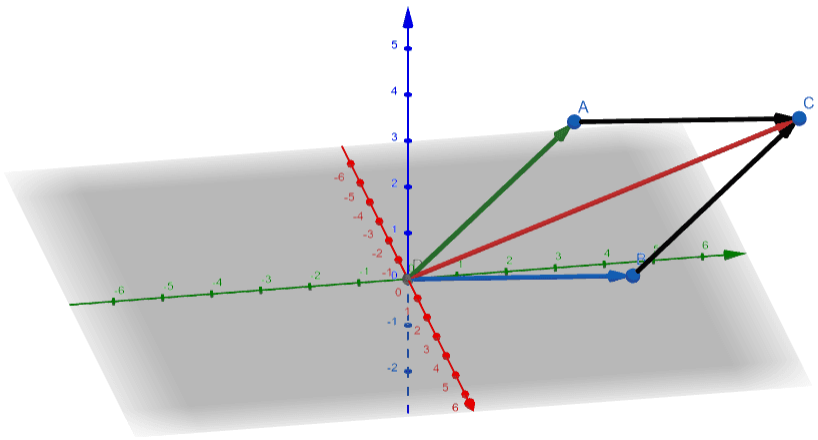


Figura 2.7: Cálculo de la resultante para obtener la posición del efector final

2.1.3. 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 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 pro-

blema, 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 2.8 se observa el brazo en 2D; puede notarse que se indica el plano que está siendo visualizado; esto puede ser configurado con el botón arriba a la izquierda. También puede observarse la posición del efector final determinada por la etapa de localización. La gráfica está graduada, de modo que se puede apreciar la precisión del modelo.

Figura 2.8: Interfaz gráfica de usuario mostrada en la pantalla táctil de la Raspberry Pi

2.1.4. Referencia

Transmisión

Se implementó en C++ y C# la funcionalidad para transmitir los datos entre la Raspberry Pi y el software SettDev. Dicho software se ejecuta en el sistema operativo Windows; para este trabajo, se ejecutó en la versión Windows 10. En la Figura 2.9 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 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 se muestran en la parte superior

del modelo. También incluye una cuadrícula donde varias líneas imitan el comportamiento del brazo robótico en 3D, en un plano. Además, cuenta con otro modelo de brazo robótico, en el que se puede ver el esquema en 3D de un brazo robótico imitar el mismo comportamiento del brazo robótico en 3D.

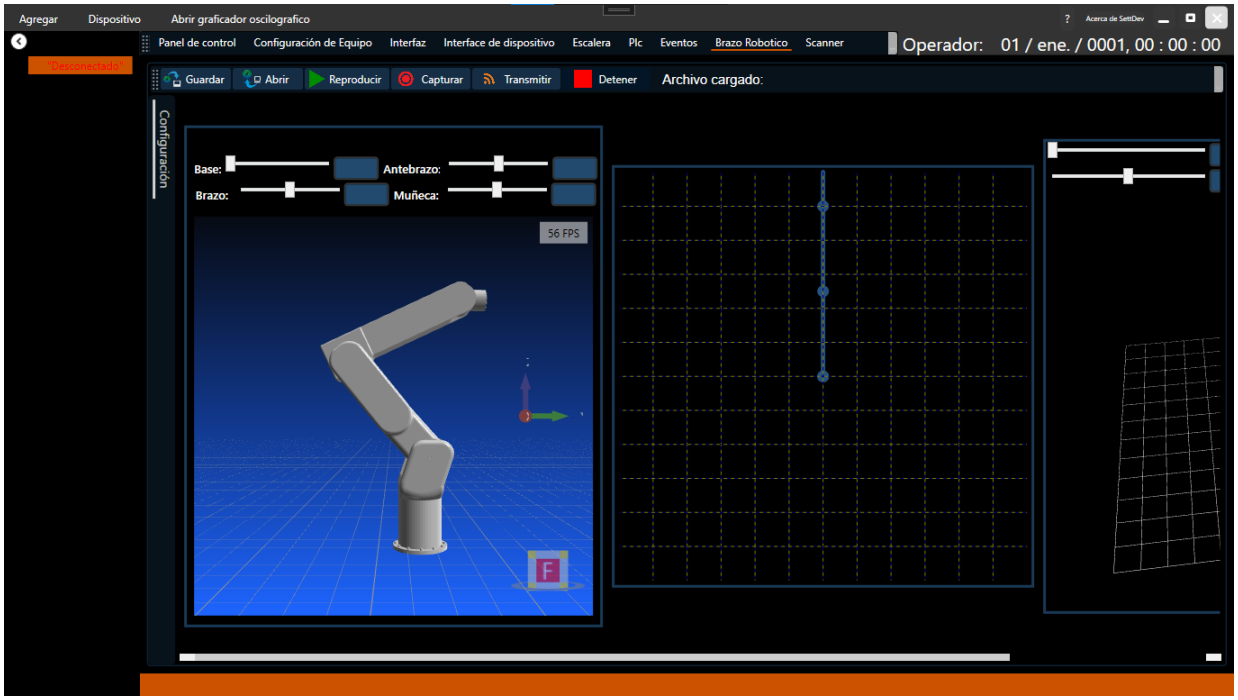


Figura 2.9: Módulo de Brazo Robótico desarrollado por SEDPC

Simulación

La Figura 2.10 muestra en detalle el brazo en 3D. Este modelo tiene 3 grados de libertad, y un grado de libertad adicional para el efector final (Muñeca). Se asignó un sensor y eje fijos para un grado de libertad específico; el eje X y eje Z del sensor colocado en el codo controlan el movimiento de la base y el antebrazo, respectivamente, mientras que el eje Z del sensor colocado en el extremo de la ortesis controla el movimiento del antebrazo. El eje Y del sensor colocado en la mano controla el movimiento de la muñeca.

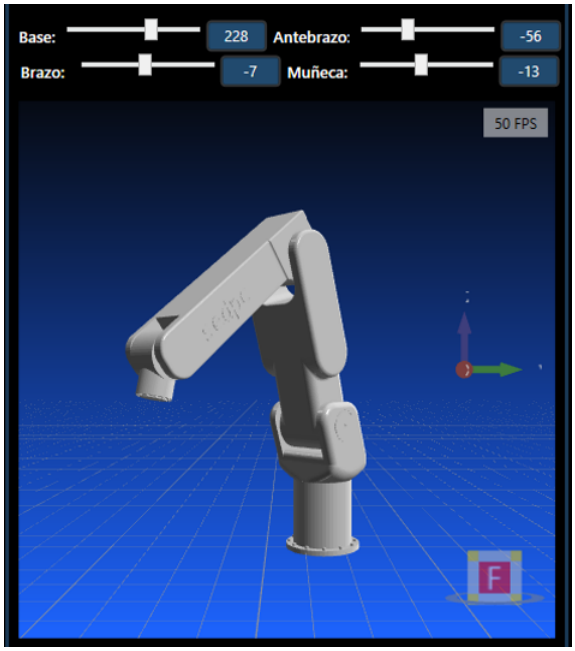


Figura 2.10: Modelo de brazo en 3D utilizado como referencia

2.1.5. Almacenamiento

Finalmente, los ángulos calculados por la red se almacenaron en un archivo. El formato para almacenar los archivos es el que muestra la Figura 2.11.

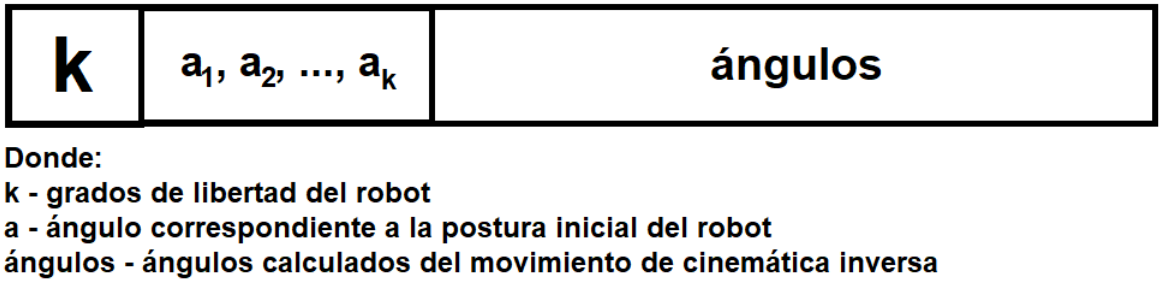


Figura 2.11: Formato del archivo que guarda los datos de movimiento

El contenido del archivo se describe como sigue:

1. El primer byte guarda la cantidad k de ángulos de cinemática inversa calculados para el robot con k grados de libertad

2. El resto del contenido del archivo son los ángulos calculados para el problema de la cinemática inversa.

De acuerdo con el mapa de registros [3], el tamaño del registro interno en los MPU que almacena el ángulo de inclinación es de 2 bytes. Esto quiere decir que, si se utilizan 6 bytes para almacenar la orientación de un sensor MPU (dos por cada eje), el archivo aumenta en 18 bytes de información por cada instante en el que se mida la inclinación del dispositivo.

2.2. Fase de ejecución

2.2.1. Lectura

Se implementó la funcionalidad en el software SettDev para leer y reproducir un archivo. La Figura 2.12 muestra el módulo 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.

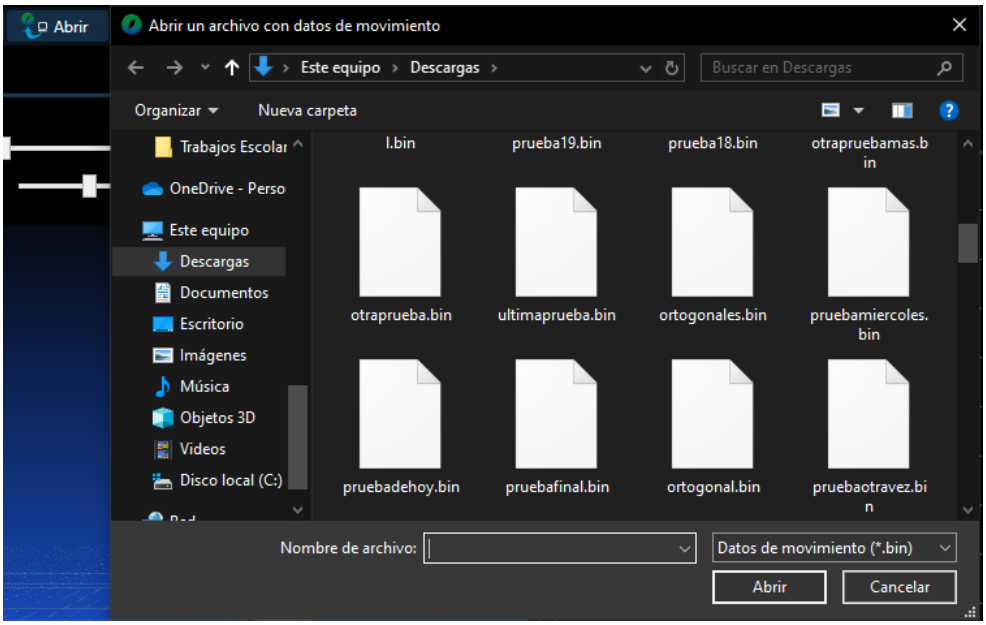


Figura 2.12: Módulo de carga de archivo implementado en SettDev

2.2.2. 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 2.13 obtenida del documento . 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 2.13: Formato de trama utilizado por el PLC-Kaab

Para elaborar la trama, se tomó como referencia la que genera un módulo desarrollado por SEDPC en SettDev para controlar servomotores. En dicha interfaz, existen controles deslizantes que indican los ángulos enviados al PLC; los valores son transmitidos al dar clic a un botón "Enviar". El contenido de la trama generado por este módulo es el que se muestra en la Figura . Los campos Fuente y Destino contienen el valor 06, que hace referencia al proceso de Interfaz Gráfica de Usuario (el software SettDev, y el software contenido en el PLC); el comando 02 indica una escritura de datos en el PLC (en este caso, el nuevo ángulo); y el subcomando 06 indica una entrada de datos digitales (los datos contenidos en la trama). Como los errores se comprueban antes de enviar la trama, el campo Error siempre contiene el valor 00; el campo Largo indica que el tamaño del campo Data es de 19 bytes. En dicho campo, los datos se indican como sigue:

1. Byte 1: Tipo de motor (el valor 1 identifica a los servomotores)
2. Bytes 3-7: Nuevo ángulo en formato Little Endian
3. Bytes 8-11: Pulso mínimo en formato Little Endian
4. Bytes 12-15: Pulso máximo en formato Little Endian
5. Bytes 16-19: Ángulo máximo en formato Little Endian

En el formato Big Endian, que es el que comúnmente se utiliza para representar un valor digital, los bytes de izquierda a derecha se ordenan desde el más significativo hasta el menos significativo. En el formato Little Endian, este orden se invierte.

2.2.3. Reproducción

Los datos se interpretaron por el PLC y se reprodujeron en los servomotores conectados al mismo. Se añadió una tarjeta de control de motores desarrollada por SEDPC que utiliza transistores MOSFET para modular el ángulo recibido por el PLC en una señal de ancho de pulso. El PLC-Kaab utilizado cuenta con 8 ranuras para conectar en cada una de ellas la línea de señal de control de cada motor a controlar. En la Figura se muestran las ranuras, en las

que se han conectado 4 líneas de señal de control de 4 servomotores. Las últimas dos ranuras son las líneas de alimentación de voltaje que alimentan a todos los servomotores conectados en paralelo.

Capítulo 3

Resultados

Referencias

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

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

[3] ——, *MPU6000 and MPU9250 Register Map and Descriptions*, 1st ed., Invensense Corporation, San Jose, USA, 2013.

Capítulo 4

Anexos