



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 S

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

Agradecimientos

Agradezco a mi asesor, José Antonio Loaiza Brito, por permitirme trabajar en un proyecto donde adquirí invaluable 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 llevar adelante el proyecto.

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

1	Anteproyecto	5
1.1	Introducción	5
1.2	Planteamiento del problema	6
1.3	Objetivo general	7
1.4	Objetivos específicos	8
1.5	Justificación	9
1.6	Estado del arte	10
1.7	Marco teórico	12
1.8	Propuesta de solución	16
2	Desarrollo del proyecto	17
2.1	Fase de captura de datos	17
2.1.1	Sensado	17
2.1.2	Localización	22
3	Resultados	23
	Referencias	24
4	Anexos	25

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).

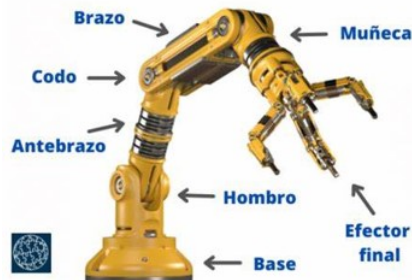


Figura 1.1: Partes de un brazo robótico articulado de 3 DoF

Sistemas Eléctricos de Potencia Computarizada (SEPMC) es una empresa mexicana que se dedica a fabricar la serie Kaab (Fig. 2) 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.

Dichos PLCs pueden ser empleados para controlar sistemas críticos, como se muestra en la Figura .



Figura 1.2: PLC-Kaab fabricado por SEDPC

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 como la textil, sobre todo cuando dicho posicionamiento tiene que realizarse de manera autónoma [2].

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 [3].

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. Justificacion

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	Oscar Carbajal-Espinosa; Leobardo Campos-Macías; Miriam Díaz-Rodríguez	Artículo  Mexico 2024	Propone un método geométrico iterativo de 3 fases para resolver el problema de la cinemática inversa.
FIKA: Un enfoque de álgebra geométrica conforme para un eficaz algoritmo cinemático inverso para un brazo robótico antropomórfico	Intel Corporation; Tecnológico Nacional de México		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.	Omur Aydogmus; Gullu Boztas	Artículo  Turquía 2024	Utiliza una red neuronal basada en aprendizaje profundo para resolver la cinemática inversa en una simulación.
Implementación de cinemática inversa sin singularidad para brazo robótico humanoide utilizando una red neuronal profunda optimizada por métodos Bayesianos.	Firat University		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.	Chengyi Zhao; Yimin Wei; Junfeng Xiao; Yong Sun; Dongxing Zhang; Qiuquan Guo; Jun Yang	Artículo  China 2024	Propone un algoritmo de aprendizaje por refuerzo que calcula la distancia entre el efector final y la posición deseada.
Solución de cinemática inversa y método de control de un manipulador de 6 grados de libertad basado en aprendizaje de refuerzo profundo.	University of Electronic Science and Technology of China		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 I²C, SPI



Figura 1.3: 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 1.4: 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.

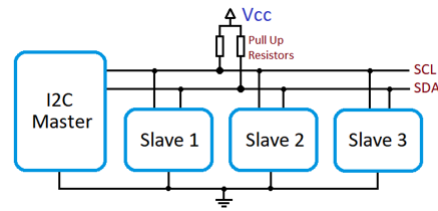


Figura 1.5: Diagrama del bus I2C

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).

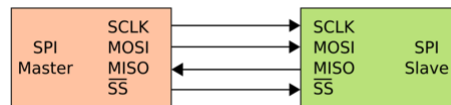


Figura 1.6: Diagrama del bus SPI

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 I²C de hasta 400 KHz.
- Pin AD0 para cambiar la dirección en el bus. Solo puede tomar las direcciones 104 y 105.



Figura 1.7: 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 1.8: Sensor inercial MPU9250

Sensor flexible capacitivo

- Mide el ángulo de torsión al que se somete el sensor en una dirección.
- El valor medido es una resistencia variable



Figura 1.9: Sensor flexible capacitivo

Convertidor analógico-digital MCP3004

- Voltaje de referencia de 2,7 V a 5,5 V



Figura 1.10: Convertidor A/D MCP3004

Microservomotor posicional

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



Figura 1.11: Microservomotor posicional

1.8. Propuesta de solución

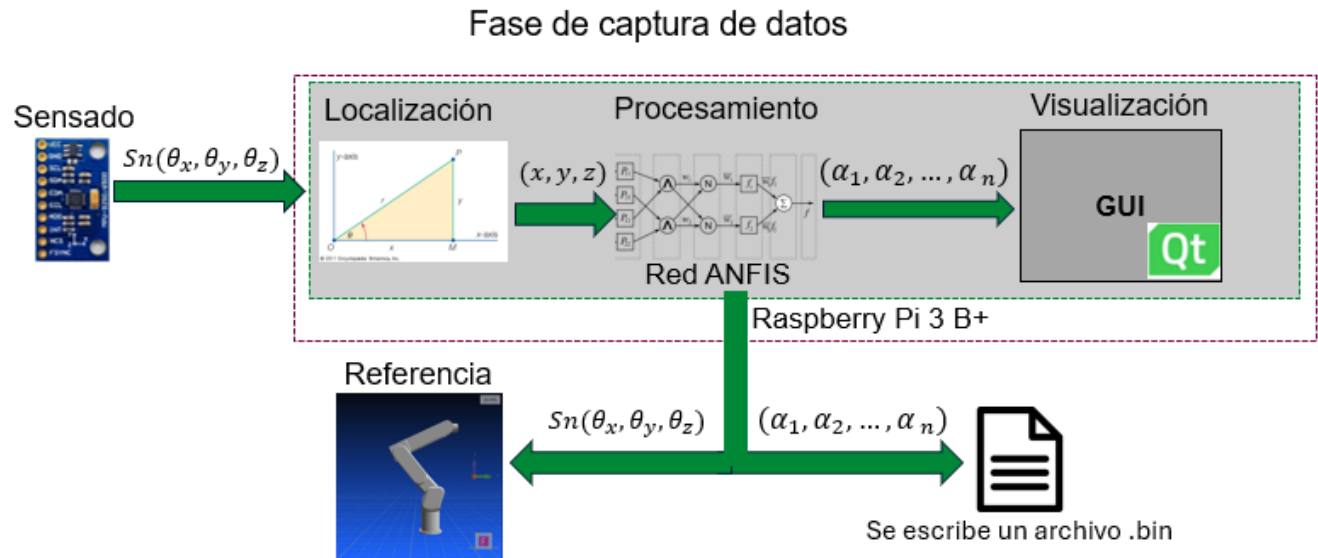


Figura 1.12: Etapa de captura de datos

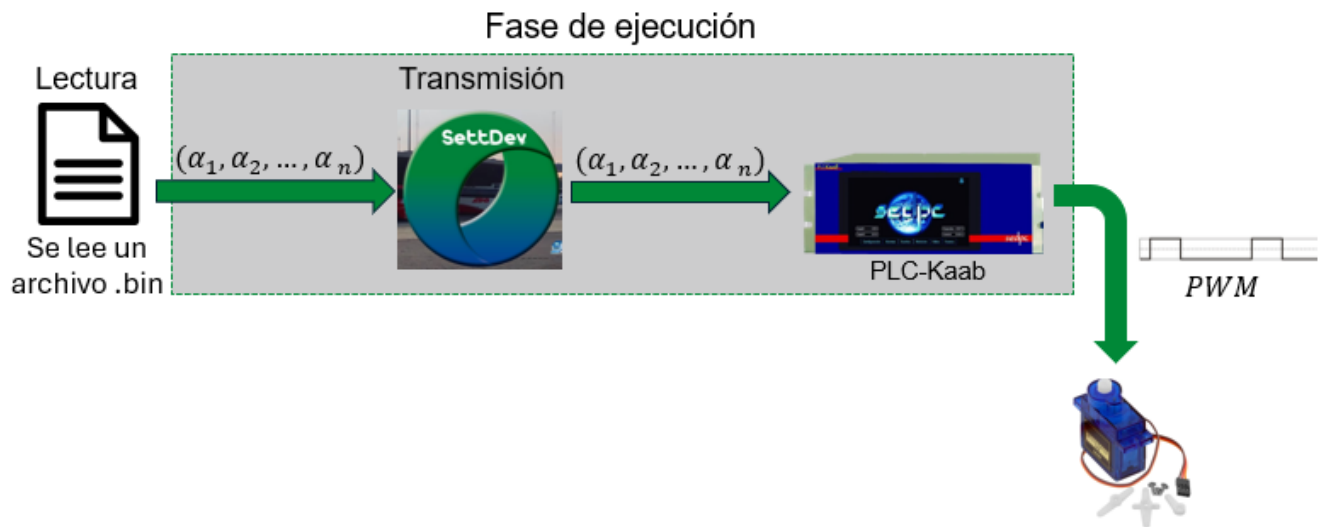


Figura 1.13: Fase de ejecución

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.



Figura 2.1: Ortesis utilizada para el proyecto

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

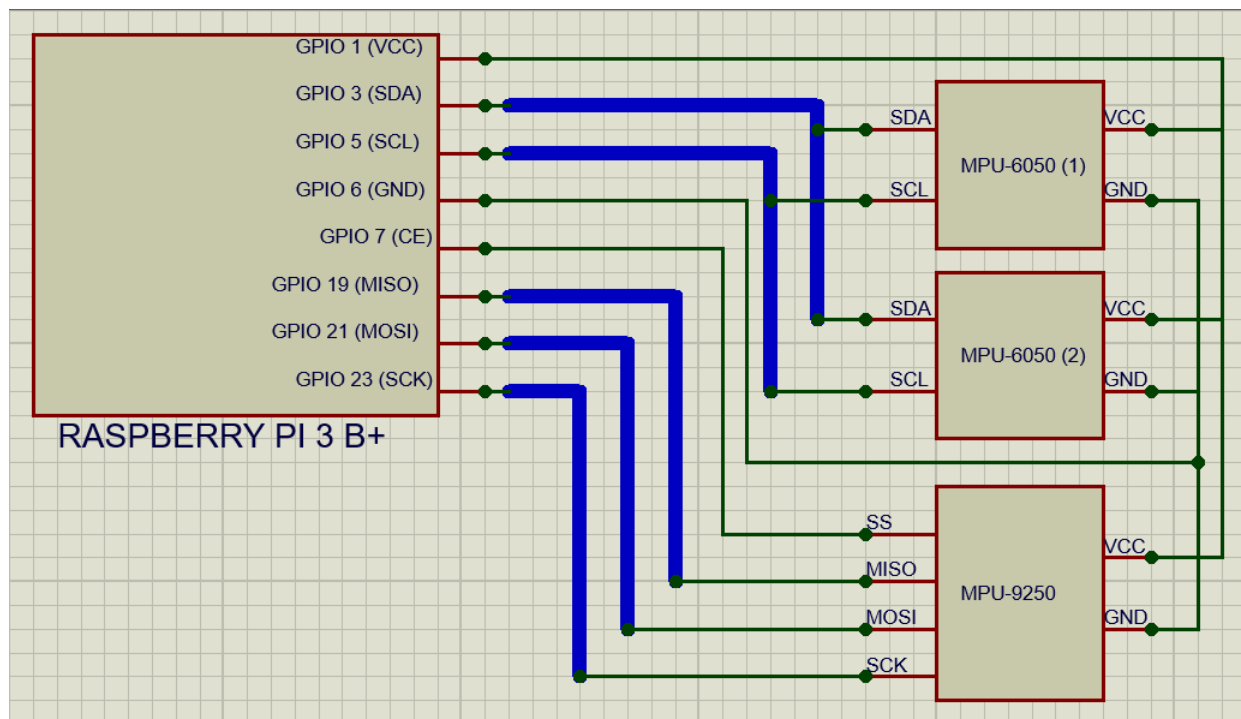


Figura 2.2: Diagrama de interconexión entre la Raspberry Pi y los sensores

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), antes de colocar dichos valores en los registros del sensor (Gyro/Accel Output Registers), o ser procesados por el Procesador de Movimiento Digital (DMP) y el buffer FIFO.

La Figura 2.4 muestra los ejes de desplazamiento y la rotación de los sensores MPU. Se

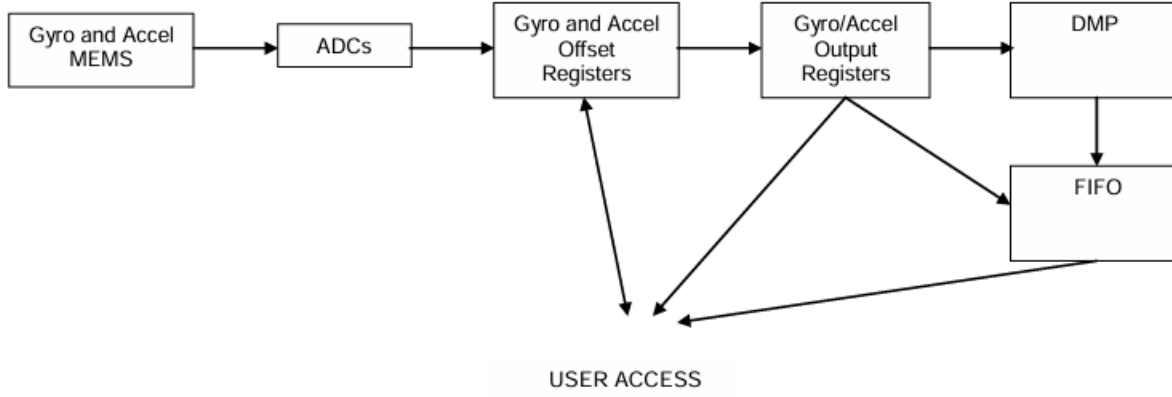


Figura 2.3: Lectura de datos de los sensores MPU

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.

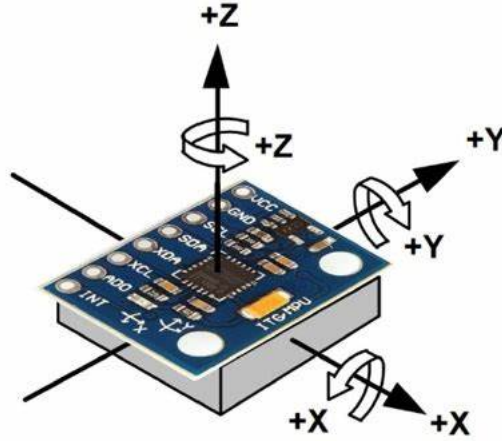


Figura 2.4: Orientación de los ejes y polaridad de rotación de los sensores MPU

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 $g = -9.8 \text{ m/s}^2$. 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.

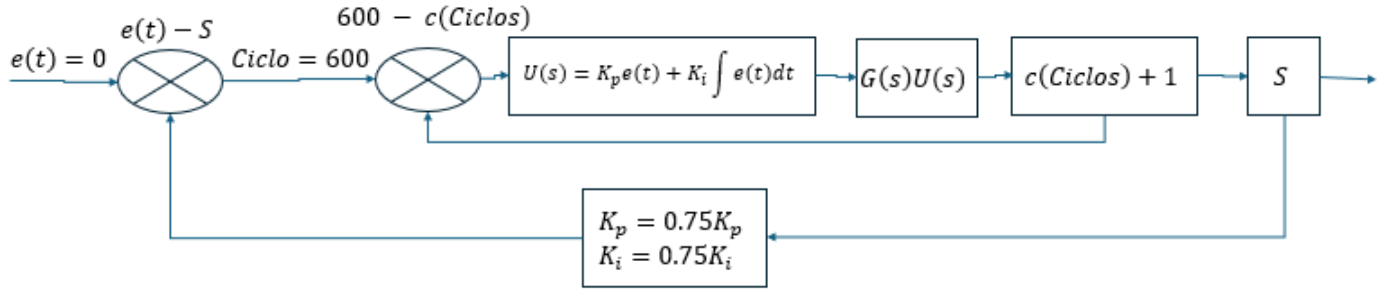


Figura 2.5: Diagrama a bloques del proceso de calibración del sensor

Se utilizó un patrón de arquitectura de segmentación de procesos (process pipeline), en el cual los datos obtenidos del DMP se colocan en un buffer interno del sensor de tipo FIFO (El primer dato que entra, es el último que sale), del cual se lee la orientación del sensor; esto permite que los datos de la orientación puedan ser procesados sin que existan errores debido a lecturas incompletas. Aunque no se indica explícitamente la estructura de datos interna del buffer, se puede representar como un buffer compartido, esto es, una cola circular. La Figura 2.6 muestra el buffer compartido. Puede notarse que se modela de tal manera que el proceso productor (el DMP) y el proceso consumidor (la lectura de datos del sensor) no accedan al mismo valor.

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].

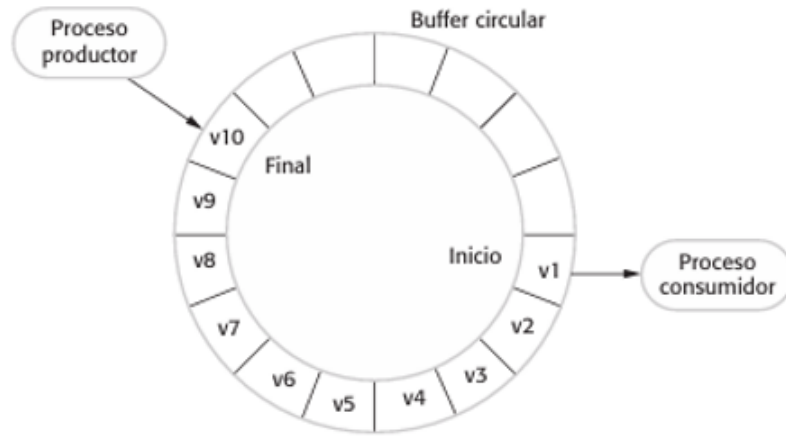


Figura 2.6: Representación del buffer interno del MPU como un buffer compartido

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 indicados en las Figuras 2.7 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 2.7: 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 utiliza cuaterniones para representar internamente la orientación; Estos son rápidamente computables y evitan problemas que se producen al girar más de 90°, como el bloqueo del cardán.

El cuaternión [3] es de la forma:.

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

La Figura muestra el proceso general para obtener la orientación del sensor. Primero, se obtiene el cuaternión medido por el DMP del buffer FIFO. Posteriormente, se obtiene el vector de gravedad; este permite calcular la aceleración de la gravedad que experimenta el sensor en los 3 ejes, para tomarlo como sesgo y corregir la medición. Por último, se convierte lo antes calculado a valores de Euler, en radianes, y finalmente, dichos valores son convertidos a su equivalente en grados, que son los que se utilizan para indicar la orientación del sensor.

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 2.8: Determinación de la orientación del sensor

El vector de gravedad se determina como lo indica la ecuación :

Para obtener la orientación de los sensores MPU en los tres ejes en radianes, se utilizan las ecuaciones :

Los valores antes obtenidos son convertidos a su equivalente en grados, como lo indican las ecuaciones :

2.1.2. Localización

Para obtener la posición (x, y, z) en el espacio, se utilizan las

Capítulo 3

Resultados

Referencias

- [1] Invensense, *MPU Hardware Offset Registers Application Note*, Invensense Corporation, San Jose, USA, 2014.
- [2] ———, *Motion Driver 6.12 – User Guide*, Invensense Corporation, Sunnyvale, USA, 2015.
- [3] F. J. Somma, “Cuaterniones y ángulos de euler para describir rotaciones en r^3 ,” Buenos Aires, Argentina, Diciembre 2018. [Online]. Available: Disponible en <https://imgbiblio.vaneduc.edu.ar/fulltext/files/TC126683.pdf>

Capítulo 4

Anexos