**Reporte: Práctica de Laboratorio 3**

| Victoria Rodríguez de León  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01656328@tec.mx](mailto:A01661890@tec.mx) | Israel Macías Santana  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01027029@tec.mx](mailto:A01661649@tec.mx) | Rodríguez Alanis Lisa Valeria  *Ingeniería en Robótica y Sistemas Digitales*  *Instituto Tecnológico y de Estudios Superiores de Monterrey*  CDMX, México  [A01656306@tec.mx](mailto:A01656306@tec.mx) |
| --- | --- | --- |

***Abstract —  
El conocimiento de diferentes protocolos de comunicación para el uso de sensores y actuadores es crucial para tomar decisiones a la hora de trabajar con sistemas embebidos. El objetivo del presente estudio es aplicar uno de los protocolos más utilizados, i2C, utilizando la tarjeta Núcleo H746zi-q con la finalidad de observar su ventajas para un caso de aplicación específico, que es la visualización de datos en una pantalla de tipo LCD. Esto involucra desde la visualización de datos básicos como texto hasta las lecturas de un sensor MPU6050. Los métodos utilizados demuestran una ejecución exitosa validada a través del hardware empleado así como en las características de los datos. A través de este estudio se ponen en práctica competencias que serán clave para futuros desafíos en el presente medio que serán de gran utilidad para el dominio de conceptos en el campo de los sistemas embebidos avanzados, que es un área de desarrollo muy predominante para los ingenieros y profesionistas de este siglo. jasjdkahsdahsjdhakjsdhajkhsdjkahsjkdhajshdkjahskdjahsjdhakjshdkjashdkjahsdjahsjdhakjshdkjahsdkjhs****Keywords* — NÚCLEO H745ZI-Q, STM32CubeIDE, I2C, LCD 16x2, MPU6050, Timers, Interrupciones, UART

I. INTRODUCCIÓN

La importancia de los protocolos de comunicación en sistemas embebidos es indiscutible, especialmente considerando su aplicación en uso de sensores y actuadores. En la presente práctica se realiza una exploración detallada de la implementación del protocolo i2C, uno de los más prevalentes en la industria, en la Núcleo H746zi-q. Al ir más allá del simple texto y abarcar la visualización de lecturas de un sensor MPU6050, se amplía el alcance de lo que generalmente se espera de una simple interfaz de usuario. Esto conlleva su nivel de dificultad, ya que se requiere hacer uso de timers por interrupción para la actualización de los datos leídos por el sensor, manejar conceptos de direcciones de memoria, entre otros. Este estudio refleja a grandes rasgos la viabilidad de I2C como un protocolo de aplicación múltiple.

II. MARCO TEÓRICO

1. **Inter-Integrated Circuit (I2C)**

I2C (Inter-Integrated Circuit), es un protocolo de comunicación serial que se utiliza para conectar múltiples dispositivos de bajo ancho de banda como sensores, microcontroladores y otros periféricos a un microprocesador. Su funcionamiento se basa en dos líneas bidireccionales para la comunicación de datos llamadas SDA (Serial Data) y SCL (Serial Clock). SDA se encarga de la transferencia de datos entre el maestro y el esclavo. SCL transporta la señal del reloj que sincroniza dicha transferencia. Cada bit de datos transferido en la línea SDA se sincroniza mediante un pulso alto o bajo de la línea SCL (Ajitgupta, 2021).

Según los protocolos 12C, la línea de datos no puede cambiar cuando la línea del reloj está alta. Sólo puede cambiar cuando la línea del reloj está baja. Las dos líneas son de drenaje abierto, por lo que se requiere una resistencia en pull-up para que las líneas estén altas ya que los dispositivos están activos en nivel bajo.  
 Los datos se transmiten en forma de paquetes de 9 bits. La secuencia de estos bits es:

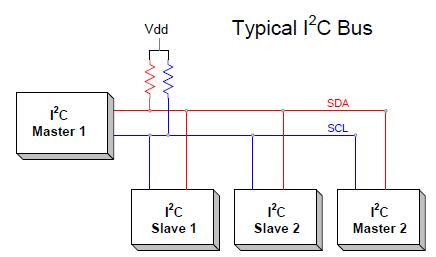
* Condición de inicio: 1 bit
* Dirección de esclavo: 8 bits
* Confirmación: 1 bit

También se tienen condiciones de inicio y de parada, conocidas como START y STOP. Esto se puede realizar manteniendo alta la línea SCL y cambiando el nivel de SDA, esto se clarifica de mejor manera a continuación:

* START: SDA HIGH ->LOW, SCL HIGH
* STOP: SDA LOW -> HIGH, SCL HIGH

Entre cada par de condiciones de inicio y parada, el bus se considera ocupado y ningún maestro puede tomar control del bus. La trama de dirección es la primera trama después de bit de inicio. La dirección del esclavo con la que el maestro desea comunicarse es enviada y cada esclavo compara su propia dirección con la enviada y envía ACK. También se tienen los bits de lectura/escritura, que permiten conocer si el maestro está enviando datos al esclavo o recibiendo datos del mismo.

Después de cada trama de datos, sigue un bit ACK/NACK. Si la trama de datos se recibe correctamente, el receptor envía el bit ACK al remitente.Respecto al formato de comunicación, los datos se transmiten en forma de paquetes de 9 bits de largo, de los cuales los primeros 8 se colocan en SDA y el último está reservado para ACK/NACK. La condición de START más el paquete de dirección más un paquete de datos más la condición de STOP forman colectivamente una transferencia de datos completa.  
La mayoría de los dispositivos I2C pueden comunicarse a 100 kHz o 400 kHz. Algunas desventajas de este protocolo es la velocidad y distancia limitada a comparación de otros protocolos de comunicación como ISP o UART; sin embargo, se puede decir que combina las mejores características de SPI y UART lo que lo vuelve un modelo muy rentable.



*Figura 1. Funcionamiento de I2C* [3]

1. **LCD 16x2**

El LCD 16x2 es un módulo de pantalla de cristal líquido que tiene 16 columnas y 2 filas para mostrar hasta 32 caracteres al mismo tiempo. Es ampliamente utilizado en proyectos de electrónica y robótica para mostrar información simple como mensajes de texto, estados de sensores o incluso sencillas interfaces de usuario; ya que tiene la capacidad de mostrar números, letras y algunos símbolos especiales. Una de las ventajas de este display es que consume mucha menos energía que las pantallas LED porque funcionan con el principio de bloquear la luz en lugar de emitirla (Söderby, 2023). Los pines comunes son:

* VSS y VDD para la alimentación.
* RS (Register Select): selecciona si los datos enviados son comandos o caracteres.
* RW (Read/Write): lectura o escritura en el módulo.
* E (Enable): habilita la escritura en los registros
* DO-D7: pines de datos
* A y K: para la retroiluminación.

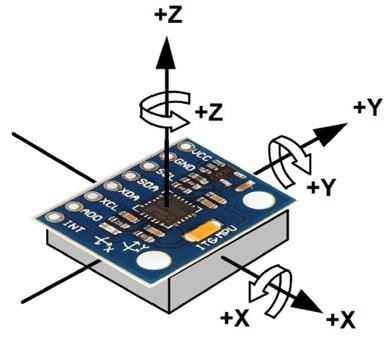


*Figura 2. LCD* [2]*.*

Gracias a su controlador, el HD44780, que está soldado a todos los pines del display, podemos conectar ahora solamente 4 cables: 2 de alimentación (voltaje y tierra) y 2 de comunicación (SCL y el SDA).

1. **Inertial Measurement Units (IMU) o MPU6050**

La Unidad de Medición Inercial, también conocida como IMU, es un dispositivo electrónico que combina un acelerómetro de 3 ejes y un giroscopio de 3 ejes para ofrecer 6 grados de libertad. Con el acelerómetro, podemos medir la aceleración lineal, es decir, el cambio de velocidad en el tiempo a lo largo de los ejes X, Y, Z. Con el giroscopio se puede medir la velocidad angular alrededor de esos mismos ejes, o en otras palabras, la tasa de cambio del ángulo. La comunicación del dispositivo es por I2C (Barak, 2021).



**Figura 3.** Sensor IMU [3]

III. METODOLOGÍA

En esta sección se explican los métodos y el procedimiento para la configuración y ejecución de cada uno de los ejercicios de este laboratorio. En estos ejercicios se habilitó el módulo de I2C, transmisión y recepción serial por medio del protocolo USART, *timers* por interrupción y la implementación de los dispositivos LCD 16x2 y MPU6050.

**Funciones para controlar LCD vía I2C**

Para la elaboración de esta práctica, se nos proporcionaron 2 archivos:  *i2c-lcd.c* y *i2c-lcd.h*. Dentro de estos archivos se encuentran las funciones necesarias para trabajar con el LCD 16x2.

Las funciones proporcionada en la biblioteca para la configuración del LCD son las siguientes:

* *lcd\_init* - inicializa el display
* *lcd\_clear* - pone el display en balnco
* *lcd\_put\_cur*- cambia el cursor de lugar
* *lcd\_send\_cmd*- envía información a un registro específico
* *lcd\_send\_data*- envía un carácter a la vez
* *lcd\_ send\_ string* - envia un string

**Funciones para enviar y recibir datos por I2C**

*HAL\_I2C\_Mem\_Read() (1)*

Esta función lee una cierta cantidad de información en modo de bloqueo desde una dirección de memoria específica. Los parámetros que recibe son los siguientes:

* **h12c**: Apuntador a la estructura I2C\_HadleTypeDef.
* **DevAddress:** Dirección del dispositivo de destino.
* **MemAddress:** Dirección de la memoria interna.
* **MemAddSize:** Tamaño de la dirección interna de memoria.
* **pData:** Apuntador al data buffer
* **Size:** Cantidad de información a mandar
* **Timeout:** Duración del timeout

*HAL\_I2C\_Mem\_Write() (2)*

Esta función escribe una cantidad de datos en modo de bloqueo en una dirección de memoria específica.

**Ejercicio I. Uso del protocolo I2C y el LCD 16x2**

Para la primera parte de esta práctica, se pide desarrollar un imprimiento de los nombres de cada integrante del equipo en una línea diferente cada 500 ms. En primer lugar, se realizó la configuración para habilitar el módulo de I2C. En la pantalla de “*Configuración de Proyecto”,* en la pestaña de *Peripherals,* se seleccionó el I2C\_4 a trabajar y se configuraron los parámetros de acuerdo a nuestras necesidades. Posteriormente, se importaron 2 archivos .c y .h a la carpeta del proyecto:

i2c-lcd.c en *LAB3>>Core>>Src. (3)*

i2c-lcd.h en *LAB3>>Core>>Inc. (4)*

Estos archivos son la biblioteca necesaria para poder controlar el LCD 16x2 vía I2C. El archivo .c fue modificado para que se adaptara a los requerimientos de nuestro programa. En el archivo .c se modificó el handler *en la sección de I2C\_HandlerTypeDef h12c4,* de acuerdo con el que se seleccionó en la configuración (IC2\_4).

En el archivo main, se inicializó el display, y posteriormente dentro del while se creó la rutina que permite que el nombre de cada integrante este apareciendo de uno en uno, recorriendo el nombre de un integrante conforme se agregue otro. Para esta rutina, en el loop del código se implementaron las funciones *lcd\_put\_cur*  y *lcd\_send\_string.* La función *lcd\_put\_cur* recibe 2 parámetros, el número de fila y el número de la columna. Esta función pone el cursor en la posición definida por columna y fila. La función *lcd\_send\_string ,* recibe como único parámetro un char string, que es el texto que se mandara y visualizará en el LCD, en la posición indicada previamente por la primera función.

El código de este ejercicio, consiste en organizar estratégicamente las únicas dos posiciones permitidas en nuestro LCD de 16x12 de cada nombre. Para esto se implementó un contador para que cada 500 ms (implementados con ayuda de un delay) aumente, y en la función de posición del cursor en número de fila es igual a cont%2 y la columna siempre es 0.

**Ejercicio 2. Uso de I2C, UART y timer por interrupciòn para el MPU6050**

Para el segundo ejercicio se configuró el *timer* 13 para generar una interrupción cada 1 segundo. Se calculó el prescaler y el *counter period*, considerando que la frecuencia del timer es de 0.005 MHz. Aplicando la fórmula (3) y (4), el valor del prescaler es de 14999 y el del *counter period* de 5000.

… (5)

… (6)

Con ayuda de la hoja de especificaciones del mapa de registros del MPU6050, se localizaron y anotaron las direcciones de los siguientes registros:

**Tabla 1: Dirección de Registros**

| Registro | Dirección |
| --- | --- |
| WHO\_AM\_I | 0x75 |
| SMPLRT\_DIV | 0x19 |
| GYRO\_CONFIG | 0x1B |
| ACCEL\_CONFIG | 0x1C |
| ACCEL\_XOUT\_H | 0x3B |
| TEMP\_OUT\_H | 0x41 |
| GYRO\_XOUT\_H | 0x43 |
| PWR\_MGMT\_1 | 0x6B |

Para la realización del ejercicio, en primer lugar se localizaron los pines dentro de la cabecera CN9 cuya función principal está dedicada a 12C (SDA y SCL). Por medio de interrupción por timer, se pide transmitir serialmente los datos recopilados del MPU6050. Se configuró el timer 13 con un prescaler de 74 y un *counter period*  de 100 para que se genere la interrupción cada segundo. Asimismo, se configuró la habilitación en CM7 para el módulo del I2C4. Por último, se definieron las direcciones de los registros de la Tabla 1. en el código, en la parte indicada del código para definiciones.

Para la recopilación de datos del MPU se implementó la función *MPU6050\_init().* Dentro de esta función se declararon 2 variables:

* *check:* almacena los valores provenientes del MPU6050.
* *data:* enviará datos al MPU6050

Además, dentro de la inicialización del MPU6050 se implementaron las funciones *(1) y (2)* para configurar las instrucciones de envío y recibo por I2C. Dentro de estas funciones se configuraron el *power management, sample rate,* el *giroscopio* y *acelerómetro.* Posteriormente, se implementaron las siguientes funciones:

* *MPU6050\_read\_gyro()*

Dentro de esta función, se crea un arreglo de 16 bits nombrado giroscopio, donde se guardan solamente los datos (eje x, eje y y eje z) recopilados del giroscopio. Primeramente, se manda a llamar la función *HAL\_I2C\_Mem\_Read(),* donde se le da la dirección de la salida del giroscopio y guarda dichos datos en el arreglo giroscopio. Finalmente los valores de cada eje son divididos entre 131 y son guardados en los arreglos *omega\_x, omega\_y y omega\_x,*  respectivamente, los datos fueron reportados en °/s .

* *MPU6050\_read\_acc()*

Dentro de esta función, se crea un arreglo de 16 bits nombrado aceleración, en este se almacenarán solamente los datos (eje x, eje y y eje z) recopilados del acelerómetro. En primer lugar, para leer el acelerómetro se debe de mandar el registro ACCEL\_XOUT\_H, el MPU6050 responderá enviando 6 bytes (2 por cada eje). Después, se manda a llamar la función *HAL\_I2C\_Mem\_Read(),* donde se le da la dirección de la salida del acelerómetro y guarda estos datos en el arreglo *acelerómetro*. Finalmente los valores de cada eje son divididos entre 16384 y son guardados en los arreglos *grav\_x, grav\_y y grav\_x,*  respectivamente. Se convirtieron los datos a gravedad y posteriormente a m/s^2.

La información recopilada de los arreglos se enviaron por serial desde la NÚCLEO hacia la computadora, cada valor del acelerómetro y del giroscopio se leerá cada segundo. La interrupción por timer envía los datos del acelerómetro y del giroscopio en las unidades anteriormente mencionadas. La transmisión serial se realizó con un BaudRate de 1152000, 8 bits de datos, 1 bit de paridad impar y 2 bits de stop.

Para la interrupciòn por timer se ocupó la función PeriodElapsedCallback(), donde después de preguntar si la interrupción del timer que recopila los datos de los 3 ejes del giroscopio

**Ejercicio 3. Lectura de la temperatura del sensor MPU6050**

Para este ejercicio, se habilitó un timer por interrupción para actualizar cada segundo la temperatura leída del sensor. La temperatura en °C recopilada, fue almacenada en un arreglo al igual que la información del acelerómetro y del giroscopio. La función implementada para este ejercicio es a siguiente:

* *MPU6050\_read\_temp()*

Dentro de esta función, se crea un arreglo de 8 bits nombrado temperatura, los datos de temperatura a través del registro TEMP\_OUT\_H, los valores recopilados son divididos entre 340 y se sumó el valor de 36.53. En primer lugar, se manda a llamar la función *HAL\_I2C\_Mem\_Read(),* donde se le da la dirección de la salida del termómetro y guarda estos datos en el arreglo. IV. RESULTADOS

**Resultados Ejercicio 1. uso del protocolo I2C y el LCD 12x6**

Los resultados obtenidos en el primer ejercicio muestran el comportamiento correcto de la proyección de nombres en el display cada 500ms. En este ejercicio se procuró que los nombres se fueran recorriendo de uno por uno sin tener que borrar la pantalla y siempre garantizando que estén dos nombres visibles. La rutina implementada siguiendo un formato simple de proyección, fue exitosa gracias al operador de modulo, que no solo realizó el trabajo pedido, si no que simplificó el código.

La rutina tuvo el siguiente comportamiento: Una vez que por ciclo se sepa qué nombre se escribe en cada renglón, después de 500 ms llega el momento de hacer un cambio de nombres y quitar el primer nombre desplegado para agregar una nuevo hasta abajo. Para lograr este resultado, se manda al lcd varios espacios en blanco, para que borre el nombre anterior y después reciba uno nuevo a reemplazar en el espacio

**Resultados Ejercicio 2. Uso de I2C, UART y timer por interrupción para el MPU6050 y Ejercicio 3. Lectura de la temperatura del sensor MPU6050**

Para el segundo ejercicio de esta actividad, se realizò un còdigo capaz de recopilar los datos del acelerómetro y giroscopio del MPU6050 por interrupciòn del timer. Los resultados fueron visibles en el IDE de Arduino, se transmitieron serialmente del núcleo a la computadora. En la terminal se despegaron los datos de dichos sensores. El formato de envío fue: nombre del sensor (temperatura, giroscopio y acelerómetro), dato *eje x* , dato *eje y* y dato *eje z*, con su respectivo número calculado. Los resultados fueron satisfactorios y se pudo entender cómo funciona el dispositivo MPU6050 y cómo acceder a las direcciones de los respectivos registros de los sensores.

V. CONCLUSIONES INDIVIDUALES

**Victoria Rodríguez de León A01656328**

Esta práctica fue una muy buena oportunidad para conocer el funcionamiento del protocolo de comunicación I2C en un medio diferente, que es la tarjeta NUCLEO H745ZI-Q. Si se comparan los métodos necesarios para hacerlo funcionar en comparación con plataformas más simples como Arduino, se puede observar cómo es necesario tener un mayor conocimiento sobre el funcionamiento en sí del protocolo para la presente práctica. Fue muy interesante de igual manera investigar cómo opera el sistema maestro esclavo, que es el que distingue a I2C y lo vuelve tan usable en la industria. En el primer ejercicio se utilizó el protocolo para ver reflejada una cadena de texto enviada en un display de tipo LCD. Posteriormente se implementó la función *MPU6050\_Init* entre otras rutinas como la que incluye la lectura del registro *WHO\_AM\_I* y la configuración del rango del giroscopio y acelerómetro para eventualmente llegar a la lectura de sus valores y reflejarlos en la misma LCD que se probó con puro texto al inicio. De la misma manera, se realizó la lectura de la temperatura.

**Lisa Valeria Rodríguez Alanís A01656306**

La práctica fue efectiva para familiarizarse con la implementación del protocolo I2C en un microcontrolador de 32 bits, específicamente en la tarjeta de desarrollo STM32 Núcleo. No solo se ganó experiencia en la lectura e interpretación de hojas de datos para configurar direcciones de memoria y pines, sino que también se reforzaron habilidades en el uso de otras herramientas integradas en la tarjeta, como el envío de datos vía USART y la manipulación de timers. Además, la práctica ofreció la oportunidad de aprender cómo incorporar archivos externos para el uso de librerías, lo que es crucial para proyectos más complejos.

**Israel Macías Santana A01027029**

Esta práctica funcionó para conocer la implementación y el uso del protocolo I2C en un microcontrolador con un núcleo de 32 bits. Su uso en la tarjeta de desarrollo STM32 Núcleo permite acostumbrarse a la lectura de la hoja de datos del sensor y de la tarjeta para configurar las direcciones de memoria del sensor a usar así como la configuración de los pines que permiten la comunicación con los dispositivos externos. Además, la práctica ayuda a reafirmar los conocimientos de activación y configuración de otras herramientas de la tarjeta como el envío de datos por USART o el uso de timers para controlar los flujos de tiempo. Incluso permite ver la forma de añadir archivos externos para leer librerías.

VI. REFERENCIAS

[1] Ajitgupta, P. (2021, February). I2C Communication Protocol. GeeksforGeeks; GeeksforGeeks. <https://www.geeksforgeeks.org/i2c-communication-protocol/>

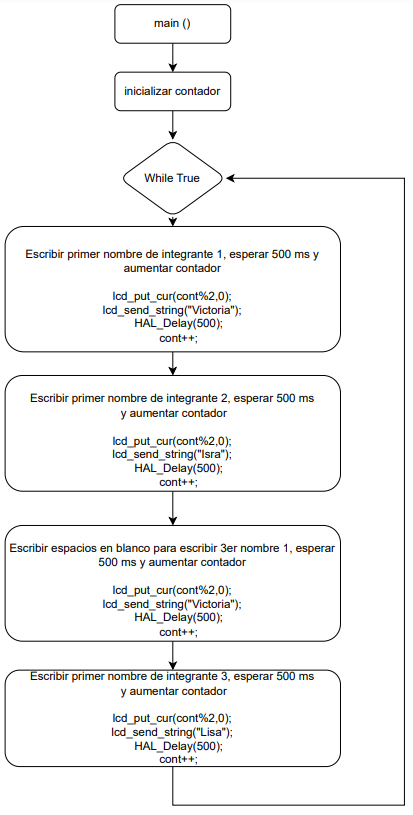
[2] Barak, O. (2021, July 31). What is IMU? - Towards Data Science. Medium; Towards Data Science. https://towardsdatascience.com/what-is-imu-9565e55b44c

[3] I2C - SparkFun Learn. (2021). Sparkfun.com. <https://learn.sparkfun.com/tutorials/i2c/all>

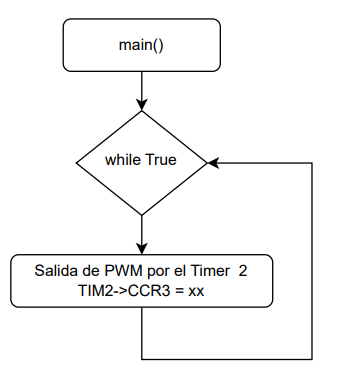
[4] Söderby, K. (2023). Liquid Crystal Displays (LCD) with Arduino | Arduino Documentation. Arduino.cc. <https://docs.arduino.cc/learn/electronics/lcd-displays>

VII. ANEXOS

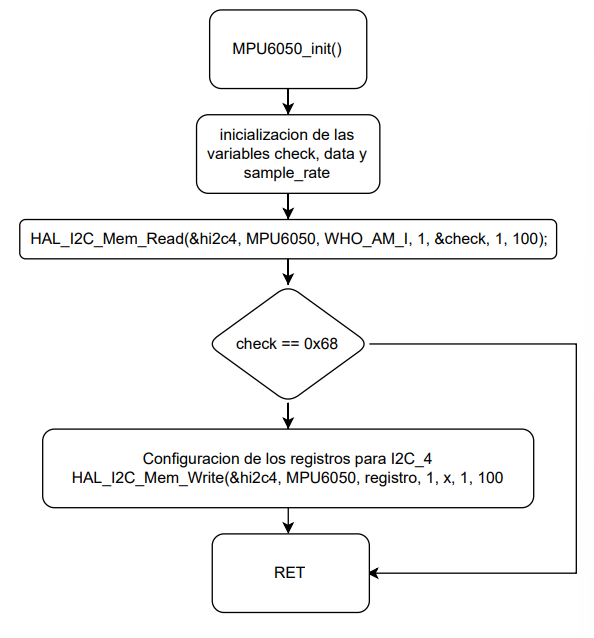
Link de YouTube (video de presentación): <https://youtu.be/RNH9XdrMZZ0>



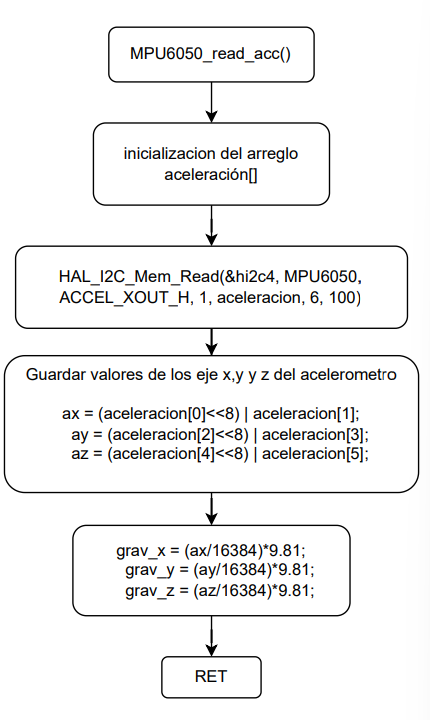
**Diagrama 1.** Ejercicio 1 main()

****

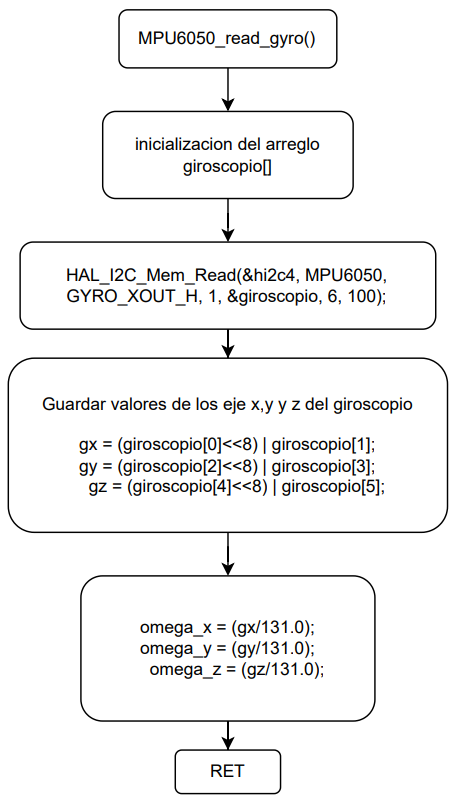
**Diagrama 2.** Ejercicio 2 main()

****

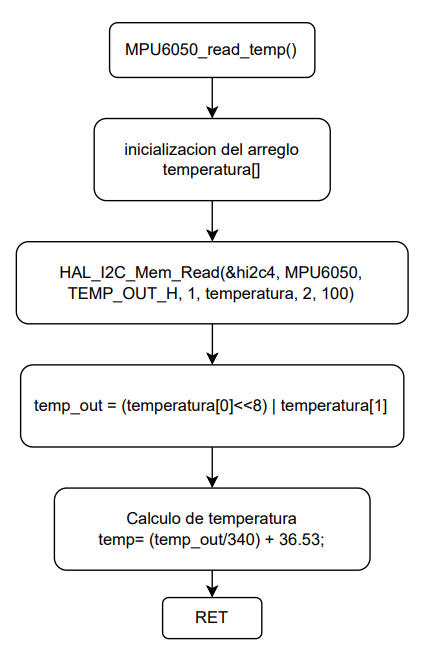
**Diagrama 3.** Ejercicio 2 MPU6050\_init()

****

**Diagrama 4.** Ejercicio 2 acelerómetro

****

**Diagrama 5.** Ejercicio 2 giroscopio

****

**Diagrama 6.** Ejercicio 3 temperatura