Trabajo Práctico Final

Sistema de Medición de Inclinación con STM32F446RE

### Materia:

### Programación de Microcontroladores

### Docente:

### Mg. Ing. Patricio Bos

### Autor:

### Ing. DURANTE MATIAS

Año: 2025

Carrera de Especialización en Sistemas Embebidos

UBA, Facultad de Ingeniería

Contenido

[**1.** **Descripción del Proyecto** 3](#_Toc211363333)

[**2.** **Comunicación I²C y UART** 4](#_Toc211363334)

[Comunicación I²C 4](#_Toc211363335)

[Comunicación UART 7](#_Toc211363336)

[**3.** **Máquina de Estados** 9](#_Toc211363337)

[**4.** **Driver DS3231** 13](#_Toc211363340)

[**5.** **DRIVER MPU6050** 14](#_Toc211363341)

[**6.** **Driver LCD 16x2** 14](#_Toc211363342)

[**7.** **Conclusiones** 15](#_Toc211363343)

[**8.** **Repositorio** 15](#_Toc211363344)

[ANEXO 15](#_Toc211363345)

[**1. Datasheet del Dispositivo** 15](#_Toc211363346)

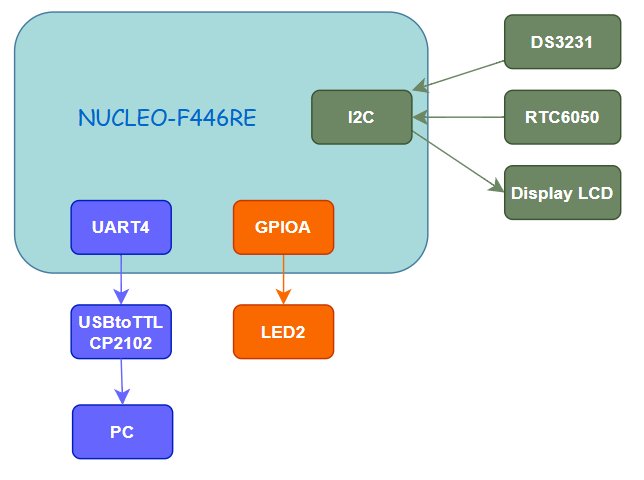
[**2. Documentación Doxygen de la API** 15](#_Toc211363347)

## **Descripción del Proyecto**

El proyecto final consiste en el desarrollo de un **firmware embebido** para la placa **NUCLEO-STM32F446RE**, diseñado para medir y visualizar la **inclinación** de un objeto en tiempo real.

El sistema integra distintos periféricos conectados al microcontrolador mediante **I²C**, permitiendo obtener lecturas de sensores, procesarlas y mostrarlas tanto en pantalla como por comunicación serial.

Para dar una referencia, el sistema está compuesto por los siguientes componentes:



* Un sensor MPU-6050, encargado de medir aceleración y velocidad angular.
* Un reloj en tiempo real DS3231, que proporciona la hora actual para el reporte del ángulo.
* Un display LCD 16x2, utilizado para mostrar en pantalla la hora y el ángulo de inclinación medido.
* La UART, que permite visualizar los mismos en una PC para su monitoreo mediante consola serial.
* Además, el LED2 de la placa actúa como indicador visual, parpadeando a una frecuencia correspondiente al ángulo medido.

En conjunto, el firmware gestiona la inicialización de todos los periféricos, la lectura periódica del sensor y el procesamiento de los datos, presentando la información del ángulo medida por el sensor DS3231 en un formato visualizable.

## **Comunicación I²C y UART**

El microcontrolador **STM32F446RE** actúa como **maestro** tanto en el bus **I²C** como en la comunicación **UART**.

A través del bus I²C se encarga de coordinar toda la comunicación con los periféricos conectados al bus: el **MPU6050** (acelerómetro/giroscopio), el **DS3231** (reloj en tiempo real) y el **LCD 16x2**, mientras que por UART transmite los datos procesados hacia la PC, que actúa como host receptor.

### Comunicación I²C

Para esta comunicación se utiliza el periférico **I2C1** de la NUCLEO, configurado en modo maestro de **7 bits** y trabajando a una frecuencia de **400 kHz.**

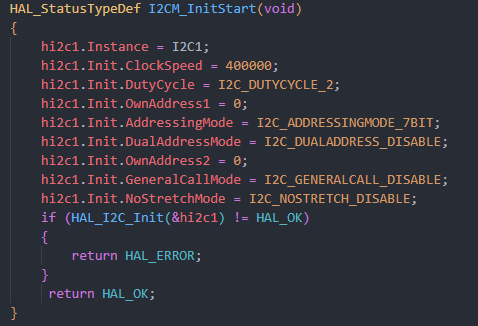
- Los pines asociados en la Nucleo-F446RE son:

* **PB8** → SCL (Clock)
* **PB9** → SDA (Data)

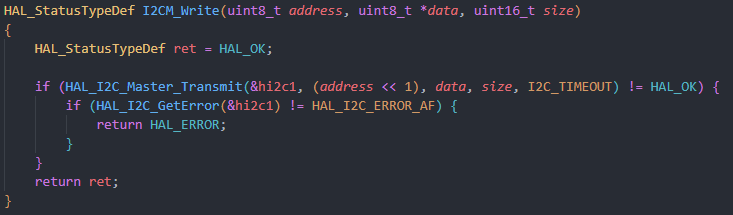
#### Implementación – Módulo **dev\_i2cm**

Toda la funcionalidad del I²C Master está encapsulada en el módulo **dev\_i2cm**, donde se implementaron las siguientes funciones:

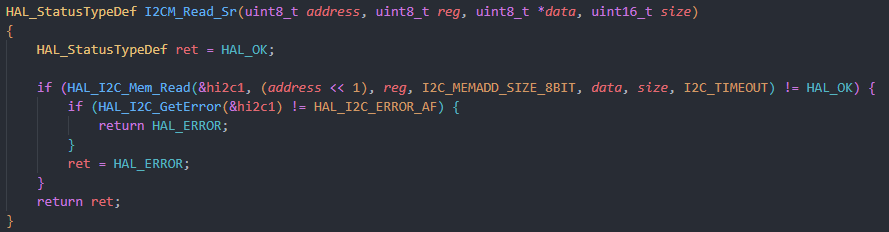
* ***I2CM\_InitStart() :*** Inicializa el periférico ***I²C1*** con los parámetros definidos ***(400 kHz, modo 7 bits, sin stretching)***.



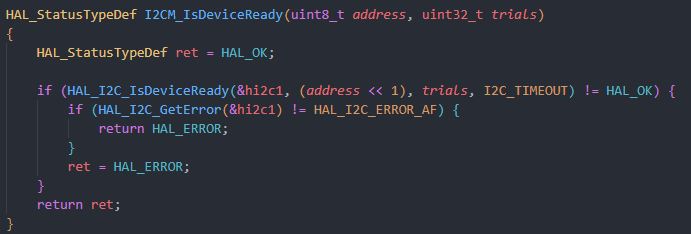
* ***I2CM\_Write(address, \*data, size):*** Permite enviar datos a un dispositivo esclavo específico.



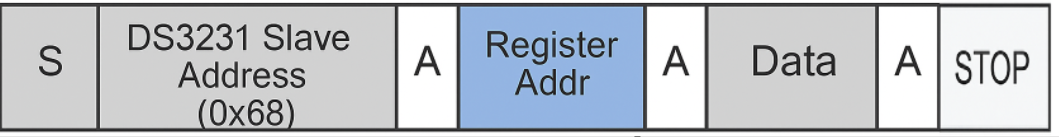
* ***I2CM\_Read\_Sr(address, reg, \*data, size):*** Implementa una lectura con ***repeated-start***, es decir, primero envía el registro interno y luego lee los datos. Esta función es la que más se utiliza, ya que los módulos mpu6050 y ds3231 requieren este tipo de lectura para acceder a múltiples bytes de manera secuencial.

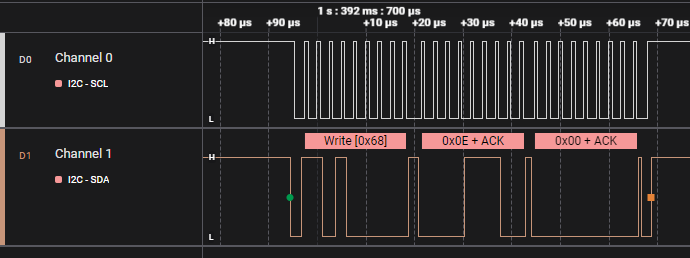


* ***I2CM\_IsDeviceReady(address, trials):*** Comprueba si el dispositivo responde a su dirección, lo que resulta útil durante la etapa de inicialización de los dispositivos.



#### EJEMPLO DE ESCRITURA DE UN REGISTRO del Ds3231



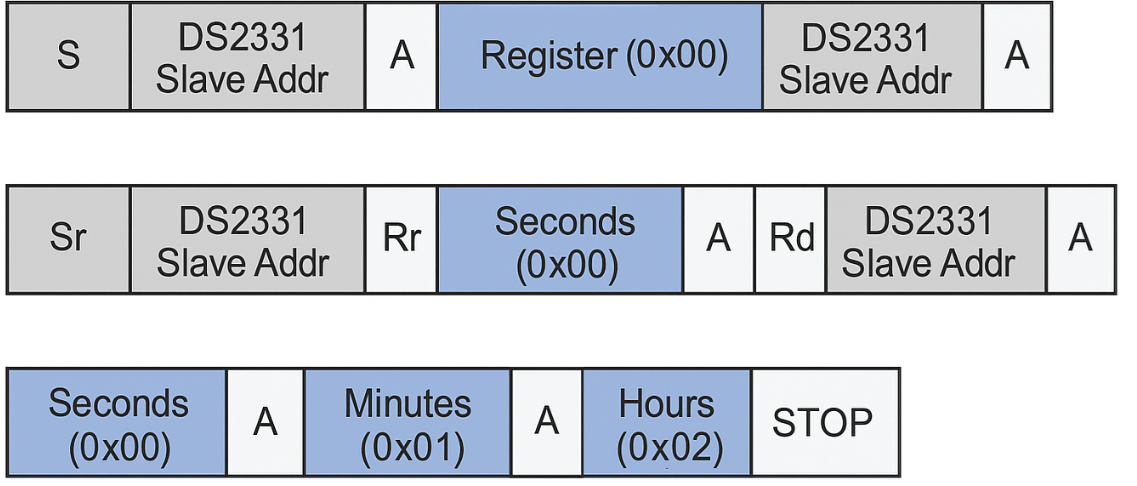


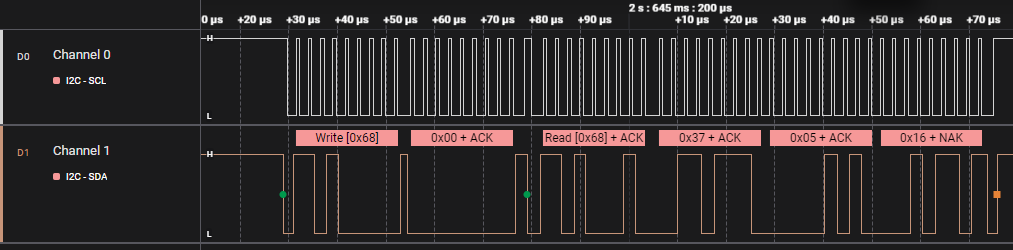
Captura del Logic2 - Ejemplo de escritura del registro de Control

#### **Descripción de cada campo**

* **S:** Condición de start generada por el Master (STM32).
* **DS3231 Slave Address:** Dirección fija → 0x68 (7 bits). + **Wr:** Bit R/W = 0 → Operación de escritura.
* **Register Addr:** Dirección registro a escribir. (Por ejemplo, 00h → para escribir los segundos).
* **Data:** Byte con los datos a escribir en el registro.
* **ACK / NACK:** El Slave envía ACK tras luego del byte escrito.
* **STOP:** Señal de finalización de la comunicación.

#### EJEMPLO DE LECTURA DE UN REGISTRO del Ds3231





Captura del Logic2 - Ejemplo de lectura de la hora

#### **Descripción de cada campo**

* **S:** Condición de start generada por el Master (STM32).
* **DS3231 Slave Address:** Dirección fija → 0x68 (7 bits).
* **Wr:** Bit R/W = 0 → Operación de escritura.
* **Register:** Dirección registro a leer (0x00 → segundos).
* **Sr:** Repeated Start.
* **Rd:** Bit R/W = 1 → Operación de lectura.
* **Seconds / Minutes / Hours:** Bytes enviados por el DS3231.
* **ACK / NACK:**
  + Master envía ACK tras cada byte leído para continuar.
  + En el último byte, el Master envía NACK para indicar fin de lectura.
* **STOP:** Señal de finalización de la comunicación.

### Comunicación UART

Para la transmisión de datos hacia el host se utiliza el periférico ***UART4*** del microcontrolador ***STM32F446RE***, configurado en modo ***transmisión y recepción (TX/RX)***, aunque en este proyecto se emplea únicamente la ***transmisión de datos***.

- Los pines asociados en la Nucleo-F446RE son:

* **PA0** → TX
* **PA1** → RX

Esta interfaz se utiliza principalmente para enviar las lecturas procesadas de los sensores **MPU6050** y **DS3231** hacia la PC, donde se pueden visualizar por medio del software ***PuTTY***.

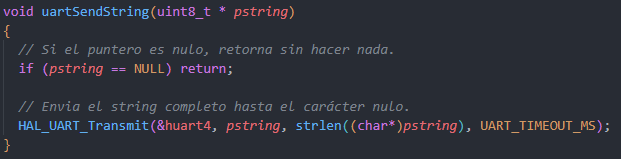
#### Implementación – Módulo **dev\_UART**

Toda la funcionalidad de la **UART** está encapsulada en el módulo **dev\_UART**, el cual se utilizó para la practica 5 de Programación de Microcontroladores, donde se implementaron las siguientes funciones:

* ***uartInit():*** Configura la UART4 e informa los parámetros de conexión al iniciar.

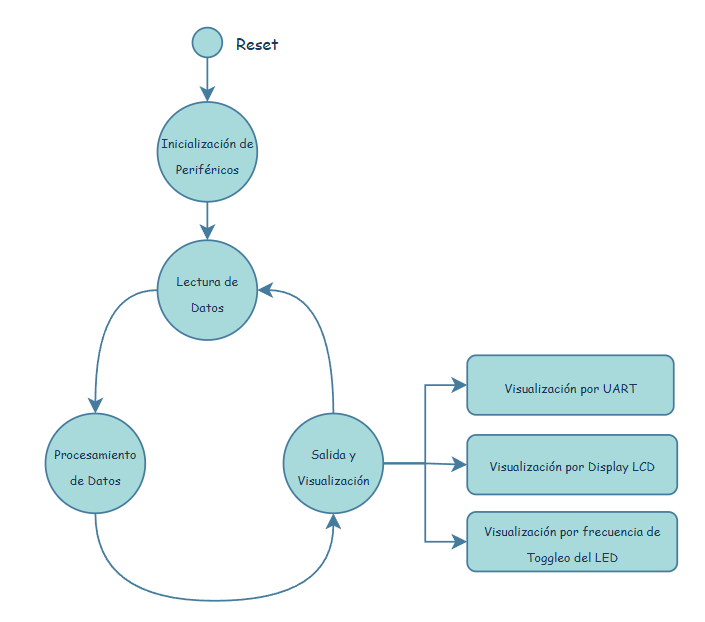
******

* ***uartSendString():*** Envía un mensaje completo (terminado en ‘\0’).



## **Máquina de Estados**

La máquina de estados que controla la funcionalidad del sistema tiene la siguiente estructura:



La función principal que la comanda es ***SM\_Iter()***. Esta función es ejecutada en un for loop infinito en el archivo **sys\_sm.c**.

Dentro de la función se realiza el flujo completo de la aplicación:

Se inicializan los periféricos en secuencia (LCD → RTC → MPU6050), y luego entra en un ciclo periódico donde lee sensores, actualiza la UI (LCD y UART) y modula el parpadeo del LED según el ángulo de inclinación medido.

### ESTADOS DE LA SM:

A continuación, se describe cada uno de los estados:

ST\_INIT  
Inicializa periféricos de forma escalonada con un número N de reintentos (máx. **BOOT\_MAX\_TRIES (3)** por dispositivo).

* **LCD:** primero, para poder mostrar mensajes de boot.
* **RTC (DS3231):** sincroniza lectura de hora.
* **MPU6050:** habilita y deja sensibilidades almacenadas en la estructura.

ST\_READ\_SENSORS  
Lee el MPU6050 y calcula el ángulo.

ST\_UPDATE\_LED  
Traduce el ángulo medido en el estado anterior en período de parpadeo del LED.

ST\_UPDATE\_UART  
Envía la hora y el ángulo medido por la UART. Vuelve al estado ST\_READ\_SENSORS.

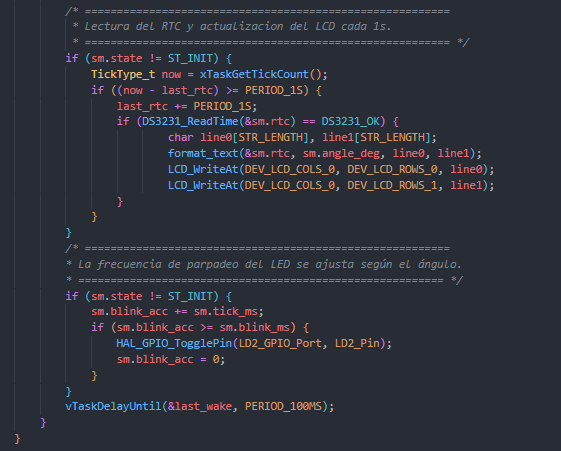
ST\_ERROR  
Muestra error en LCD y realiza un reset del MCU.

### PARPADEO DEL LED:

|  |  |  |
| --- | --- | --- |
| Inclinación | Frecuencia | Período |
| 0–10° | 1 Hz | 1000 ms |
| 10–25° | 2 Hz | 250 ms |
| 25–40° | 4 Hz | 125 ms |
| > 40° | 8 Hz | 50 **ms** |

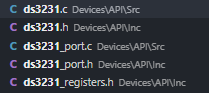






## **Driver DS3231**

El **DS3231** es un **reloj en tiempo real (RTC)** que mantiene la hora y fecha del sistema. La idea de su utilización es **poder generar un reporte con fecha y hora sobre el ángulo medido por el MPU6050**.

Para integrarlo, se desarrolló un driver capaz de inicializar el chip, leer la fecha y hora actual, y configurarla. El driver actual consta de 5 archivos:  


A continuación, se describe su funcionalidad:

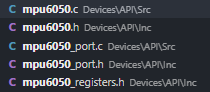
* **ds3231.c:** Se declaran las funciones de inicialización, lectura y escritura del reloj.
* **ds3231\_port.c:** Implementa las funciones de bajo nivel para leer y escribir registros a través del bus I²C.
* **ds3231.h:** Se definen estructuras y prototipos de funciones públicas del DS3231.
* **ds3231\_port.h:** Define los prototipos y macros internos utilizados por **ds3231\_port.c**
* **ds3231\_registers.h:** Contiene las direcciones de los registros y las máscaras de bits utilizadas en el driver del DS3231.

## **DRIVER MPU6050**

El **MPU6050** es un sensor que combina un acelerómetro triaxial y un giroscopio, permitiendo obtener información sobre la inclinación y el movimiento del sistema.

Para este proyecto, se desarrolló un driver modular para manejar el chip, encargado de inicializar el sensor y realizar lecturas periódicas de los ejes X, Y y Z, para posteriormente calcular el ángulo de inclinación.

El módulo está compuesto por los siguientes archivos:



* **mpu6050.c:** Contiene la inicialización del sensor, lectura de registros y conversión de valores a unidades físicas.
* **mpu6050.h:** Define las estructuras, enumeraciones y funciones públicas del módulo, así como el ***MPU6050\_Handler*** utilizado por el firmware para encapsular la información del dispositivo.
* **mpu6050\_port.c:** Al igual que ds3231\_port.c, implementa las funciones de acceso directo a los registros del dispositivo a través del bus I²C.
* **mpu6050\_port.h:** Declara los prototipos y constantes internas del puerto, utilizadas por mpu6050\_port.c.
* **mpu6050\_registers.h:** Define las direcciones de los registros internos del sensor.

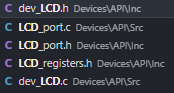
## **Driver LCD 16x2**

El **LCD 16×2** permite mostrar la hora actual y el ángulo de inclinación medido por el sensor MPU6050.

Para su control, se desarrolló un driver completamente modular que utiliza el expansor **PCF8574**, conectado mediante el bus I²C, permitiendo reducir el número de pines utilizados del microcontrolador.

El módulo implementa la inicialización completa del controlador HD44780 en modo de 4 bits, y funciones de alto nivel para imprimir texto, limpiar la pantalla, mover el cursor o centrar mensajes automáticamente.

El módulo que maneja el LCD está compuesto por los siguientes archivos:



* **dev\_LCD.c:** Contiene las funciones principales de control del display: inicialización, escritura de texto, limpieza y posicionamiento del cursor.
* **dev\_LCD.h:** Define los prototipos públicos, constantes y funciones auxiliares del módulo.
* **LCD\_port.c:** Implementa las rutinas de bajo nivel necesarias para enviar comandos y datos al LCD mediante el PCF8574 (que funciona mediante I²C).
* **LCD\_port.h:** Declara las macros internas de conexión entre el expansor PCF8574 y el LCD, a través del bus I²C.
* **LCD\_registers.h:** Contiene los comandos, flags y direcciones de memoria del controlador HD44780 utilizados internamente para el control del LCD.

## **Conclusiones**

Durante el desarrollo del proyecto se logró integrar y poner en funcionamiento los distintos módulos que conforman el sistema: el sensor **MPU6050**, el reloj **DS3231**, el display **LCD 16×2** y la comunicación por **UART**.

Cada uno de ellos fue controlado a través de drivers desarrollados específicamente, lo que permitió un desarrollo de código ordenado, claro y fácilmente portable para futuras implementaciones.

## **Repositorio**

En el siguiente enlace se encuentra mi repositorio de GitHub, donde aparecen disponibles los archivos y la documentación utilizada para el desarrollo de este proyecto.

**Enlace de acceso:**  
[Repositorio GitHub](https://github.com/matuuted/RTC_DS3231)

# ANEXO

### **1. Datasheet del Dispositivo**

**Enlace de acceso:**  
[DS3231 – Reloj de Tiempo Real (RTC)](https://www.alldatasheet.com/datasheet-pdf/pdf/112132/DALLAS/DS3231.html)

### **2. Documentación Doxygen de la API**

La documentación de la API para el driver del DS3231 se generó automáticamente con Doxygen y se encuentra disponible en el siguiente enlace:

**Enlace de acceso:**  
[Doxygen – API DS3231](https://matuuted.github.io/RTC_DS3231/)