BORDA ACADEMY

Embedded System Project

Name : Said Modalı

Mail : smodalim@gmail.com

Contents

1.General Description:

This project has been carried out as part of the Borda Academy 2025 Embedded Systems Developer Assignment. The main objective of the project is to design and implement a device at both hardware and software levels that can detect environmental data within the field of embedded systems. Accordingly, a system will be developed using I2C-based sensors capable of measuring environmental parameters such as temperature, humidity, and CO_2, and this data will be transmitted to the external world at regular intervals using Bluetooth Low Energy (BLE) technology.

The project is inspired by real-world problems and aims to achieve the following key objectives:

- Reading sensor data using I2C communication in embedded systems,
- Processing the sensor data with filtering methods (e.g., moving median filter),
- Storing the data in a circular buffer and calculating statistical summaries (standard deviation, max, min, median),
- Broadcasting this data periodically using BLE technology.

2.Used Sensors, Microcontroller, BLE, and IDE:

Here is the information about the sensors , BLE modüle and microcontroller used below.

TMP117 – Temperature Sensor

- **Type:** Digital temperature sensor
- **Interface:** I^2C
- **Accuracy:** ±0.1°C (high precision)
- **Features:**
    - Ultra-high accuracy temperature readings
    - Low power consumption

- Wide operating range: -55°C to +150°C
- On-chip EEPROM and temperature alert thresholds

- **Datasheet:**
  https://www.ti.com/lit/ds/symlink/tmp117.pdf
- **Producer:** Texas Instruments

🔹 HTU21D – Humidity Sensor

- **Type:** Digital humidity and temperature sensor
- **Interface:** I^2C
- **Humidity Range:** 0% to 100% RH (±2% RH accuracy)
- **Temperature Range:** -40°C to +125°C
- **Features:**
  - Fully calibrated and ready to use
  - Compact and low-power
  - Provides both humidity and temperature data
  - Ideal for embedded environmental sensing applications

Datasheet:
https://www.te.com/usa-en/product-CAT-HSC0004.html

🔹 SCD30 – CO_2 Sensor

- **Type:** CO_2, temperature, and humidity sensor
- **Interface:** I^2C (also supports UART)
- **CO_2 Range:** 0 – 40,000 ppm
- **Features:**
  - Accurate CO_2 concentration measurements
  - Integrated temperature and humidity compensation
  - Automatic calibration
  - Long-term stability and reliability

Datasheet:
https://sensirion.com/media/documents/6A0D6A3D/616B2954/Sensirion_CO2_Sensors_SCD30_Datasheet.pdf

---

🔹 HM-10 – BLE Module

- **Type:** Bluetooth Low Energy (BLE) 4.0 module
- **Interface:** UART
- **Operating Voltage:** 3.3V to 6V
- **Features:**
  - UART communication with easy AT commands
  - Compatible with iOS and Android
  - Supports both Master and Slave modes
  - Ideal for low-power wireless communication

Datasheet (JNHuaMao):
☞ https://www.martyncurrey.com/downloads/

STM32F411RE – Microcontroller

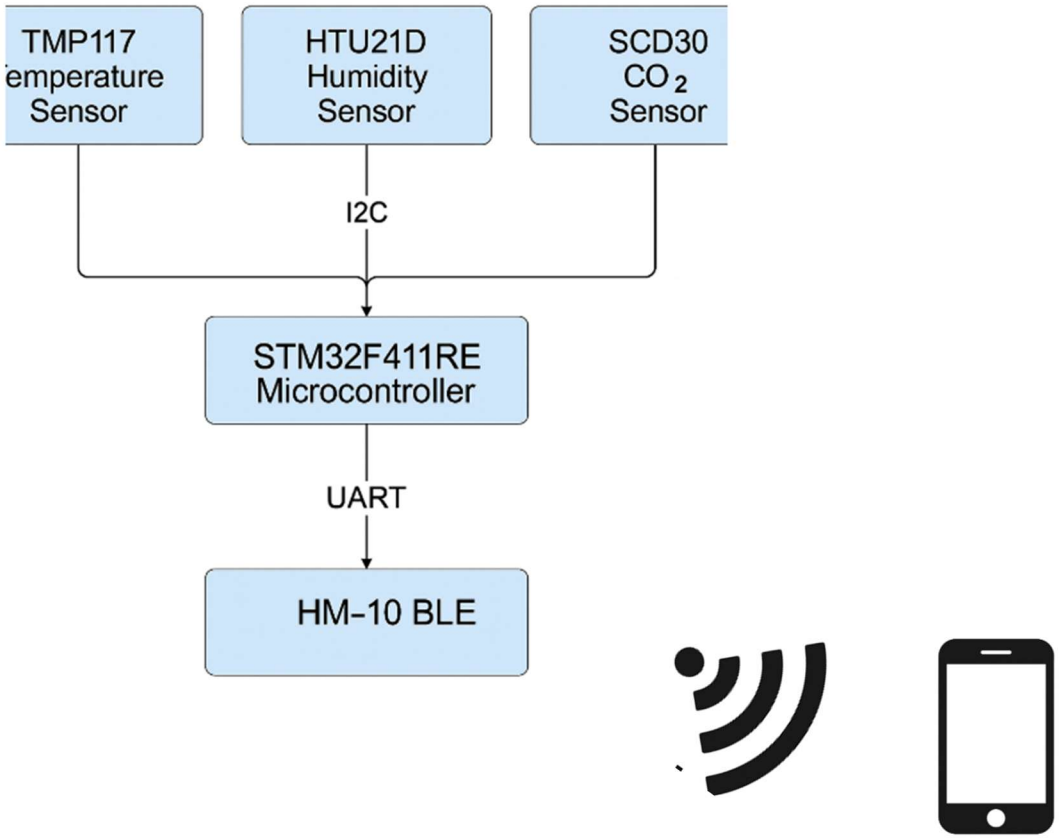
- **Core:** ARM Cortex-M4 (with FPU)
- **Clock Speed:** Up to 100 MHz
- **Flash Memory:** 512 KB
- **SRAM:** 128 KB
- **Operating Voltage:** 1.8V to 3.6V
- **Interfaces:**
  - I^2C, SPI, UART, USART
  - ADC, PWM, USB OTG
- **Package:** LQFP64
- **Features:**
  - High-performance 32-bit MCU
  - Ideal for real-time applications and signal processing
  - Supports various communication protocols
  - Low power consumption with multiple sleep modes
  - Commonly used with Nucleo development boards (e.g., Nucleo-F411RE)

IDE : Microsoft Visual Studio

3. Project Flowchart and File Design(with module)

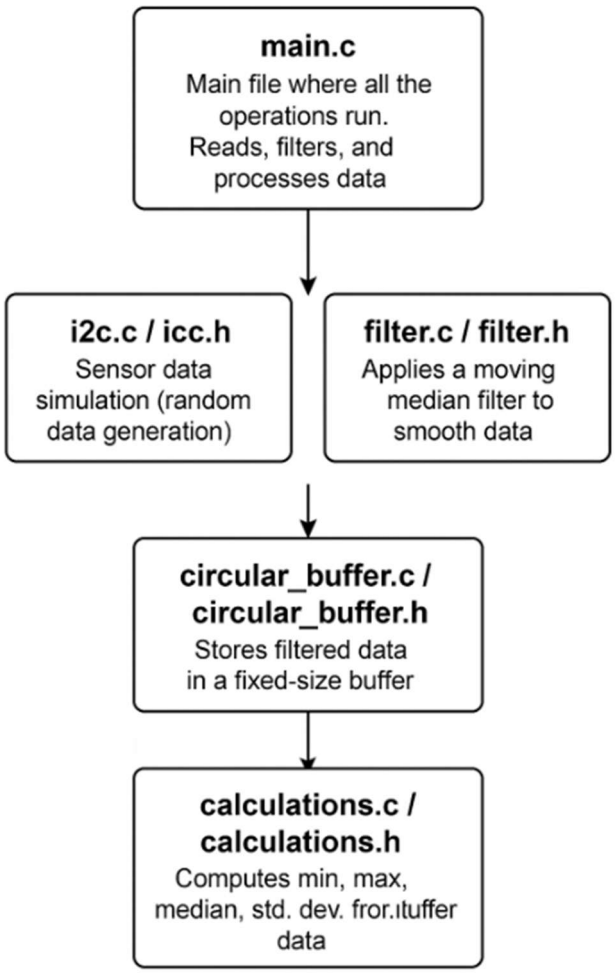
Flowchart : It is shown in Figure 3.1.

Figüre 3.1



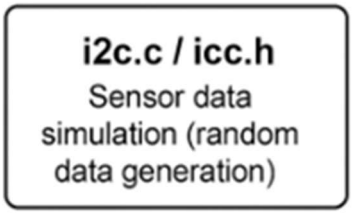
File design :

The code structure for tasks a, b, c, d, and e has been designed in a modular way to allow for easy simulation. This approach makes the system both easier to understand and more maintainable, as illustrated in Figure 3.2.



4.Tasks :

a-) and b-) :



As shown in Table 4.1, the addresses were obtained from the datasheets, and since there are no default address conflicts, they will be used as is.

I²C Address Summary

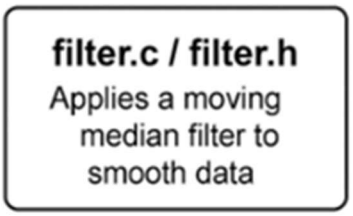
Sensor	I²C Address	Configurable?
TMP117	0x48 – 0x4B	✓ (Fixed)
HTU21D	0x40	✗ (Fixed)
SCD30	0x61	✗ (Fixed)

Table 4.1

In this section, the function filter_sensor_value is defined in the i2c.c submodule file and declared in the i2c.h file for later use in main.c. Since the sensors are not physically available, random values within a specific range are generated using the time.h and stdlib.h libraries inside the function. This function is used to simulate reading data from sensors for testing purposes.

The code has been added to the submitted file.

C-)Moving Median Filter – `filter_sensor_value()` Function:



Raw data obtained from sensors is often subject to noise, fluctuations, and occasional spikes. To make the data more stable and reliable, a **moving median filter** was implemented before storing or using the readings.

The function `filter_sensor_value()` keeps the latest N measurements (e.g., the last 5) in memory using a static array. Every time a new reading comes in, it is added to this array in a circular fashion. Then, the array is copied and sorted, and the **median** value is returned. This median value represents the central tendency of recent readings, eliminating the effect of outliers and sudden jumps.

The use of a **static buffer** inside the function allows the filtering to be continuous across calls without losing historical values.

d. Circular Buffer – `buffer_add()` and `buffer_get_all()` Functions :

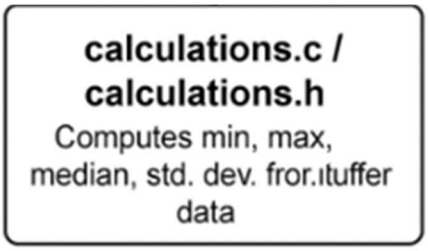
To temporarily store the filtered sensor readings, a **circular buffer** was implemented for each sensor. This data structure allows efficient use of memory by using a fixed-size array that automatically overwrites the oldest data when full.

The `buffer_add()` function adds new filtered values to the buffer. It maintains a pointer (`head`) that wraps around the array as values are added. When the buffer is full, the oldest data is overwritten by the new one.

The `buffer_get_all()` function retrieves all valid values currently stored in the buffer into an output array. This is especially useful when performing calculations such as statistical analysis on recent readings.

By using this approach, memory usage remains constant, and the system always has access to the most recent sensor values.

e. BLE Transmission of Statistical Measurements – `calculate_statistics()` Function:



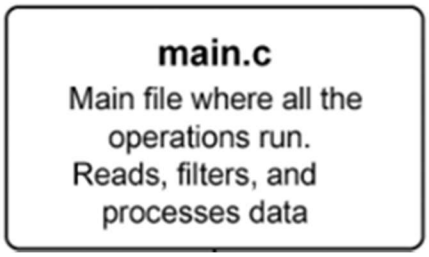
After accumulating a sufficient number of filtered values in the circular buffer, statistical analysis is performed to extract meaningful insights from the data. This is done using the `calculate_statistics()` function.

This function computes the following metrics:

- Minimum value – the lowest reading
- Maximum value – the highest reading
- Median value – the middle value after sorting
- Standard deviation – the variation of readings from the mean

These values are stored in a structure called `calculation_result_t`. The results are then formatted and printed using `printf()`, simulating what would eventually be transmitted over Bluetooth Low Energy (BLE) to a smartphone or central system.

5. Simulation:



The output is presented at 1-second intervals as shown in Table 5.1.

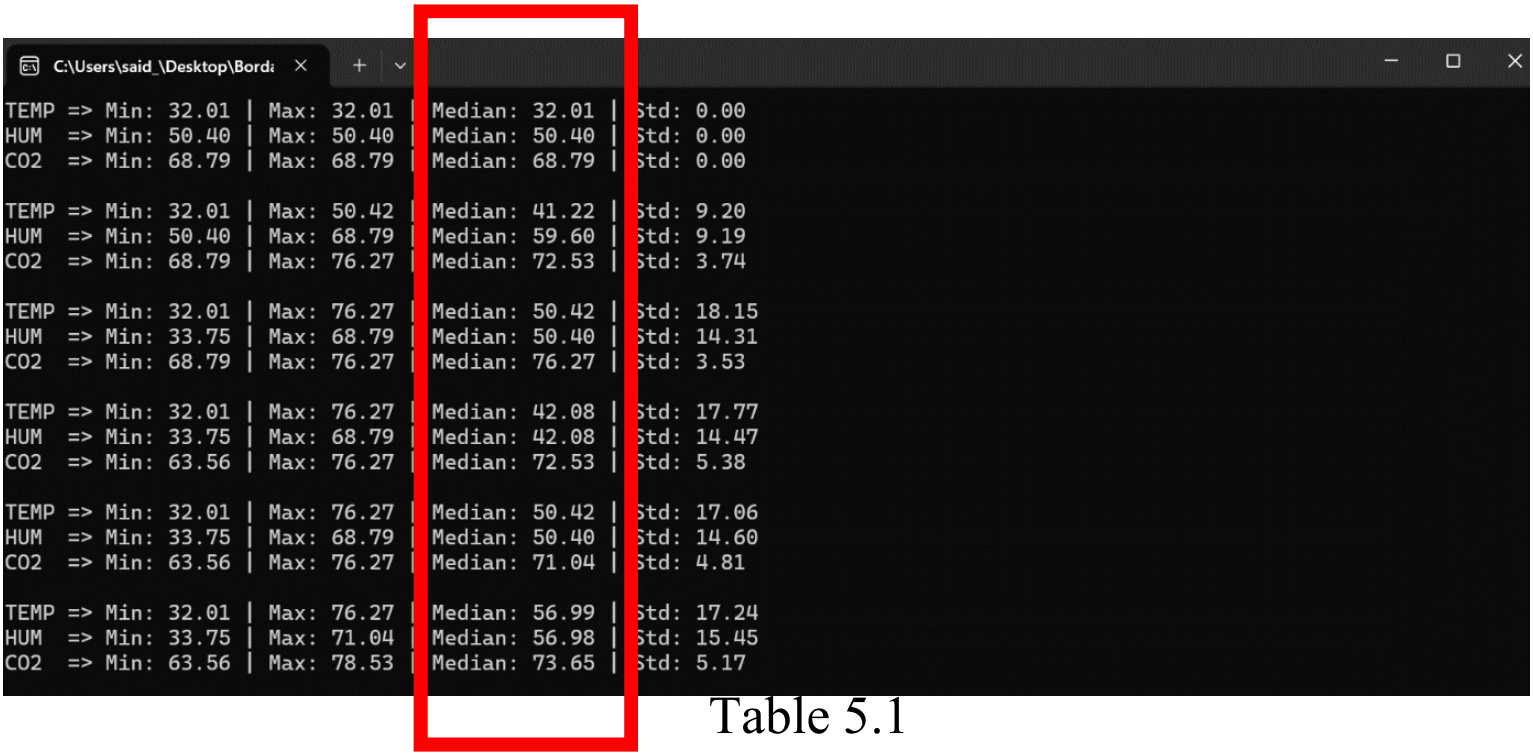


Table 5.1

As can be seen, the data is continuously changing.

6. Missing Parts and Conclusion:

Missing Parts :

The steps were implemented one by one as specified in the assignment. However, at the beginning, microcontroller initialization was not performed, and sensor data reading was done using random value generation instead of actual hardware due to the lack of real sensors. Additionally, the final task of transmitting data via BLE could not be completed because the necessary hardware was not available.

Nevertheless, all other tasks were completed according to their intended logic and structure.

Conclusion :

This assignment, despite being theory-heavy at the beginning, can clearly be considered a motivating application in terms of its real-life implementation—specifically, understanding whether a plant is in optimal conditions or not. I can say that I truly enjoyed this task, as I first worked on the design and then implemented it, which contributed greatly to my motivation.

Thanks for everything !

7. References:

[1]:https://www.youtube.com/watch?v=6IAkYpmA1DQ&t=118s

[2] :Median Filtering. In: Smith, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, 1997.

[3] : **Circular Buffer Theory.** Embedded Systems Programming, Jack Ganssle. https://www.ganssle.com