HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

PROJECT II

AIR QUALITY MONITORING STATION WEB APP

Nguyen The Thao

thao.nt192254@sis.hust.edu.vn

Supervisor:	Nguyen Thi Lan Huong
	Sign of the supervisor
Department:	Automation
School:	Electrical Engineering

Ha Noi, 01/2024

Project2-Report

January 23, 2024

Table of Contents

Chapter 1 Introduction

Chapter 2 Background Theory

The overall project architecture:

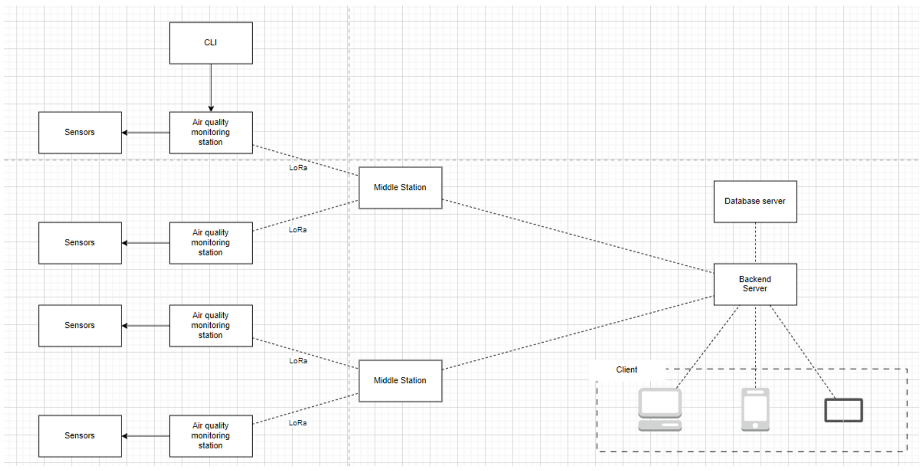


Figure 2.1. Project Architecture

In this project, I used some hardwares and some frameworks like that:

2.1 Hardware

2.1.1 STM32f103c8t6 - STM32f103c6t6

The STM32f103c8t6, also known as the blue pill, is a low-cost development board based on the ARM Cortex-M3 microcontroller from STMicroelectronics. It has a 32-bit CPU that can run up to 72 MHz, 64 KB of flash memory, 20 KB of SRAM, and various peripherals such as GPIO, ADC, DAC, SPI, I2C, UART, and USB. The blue pill board is widely used for embedded projects due to its high performance, rich features, and affordable price. In this thesis, I will use the blue pill board as the main component of an air quality monitoring station, which can measure and transmit data on the concentration of particulate matter (PM) in the ambient air.

2.1.2 ESP32

The ESP32 is another low-cost development board that has integrated Wi-Fi and Bluetooth connectivity. It can also perform as a standalone system or as a slave device to a host MCU. The ESP32 has a dual-core processor, 520 KB of RAM, and

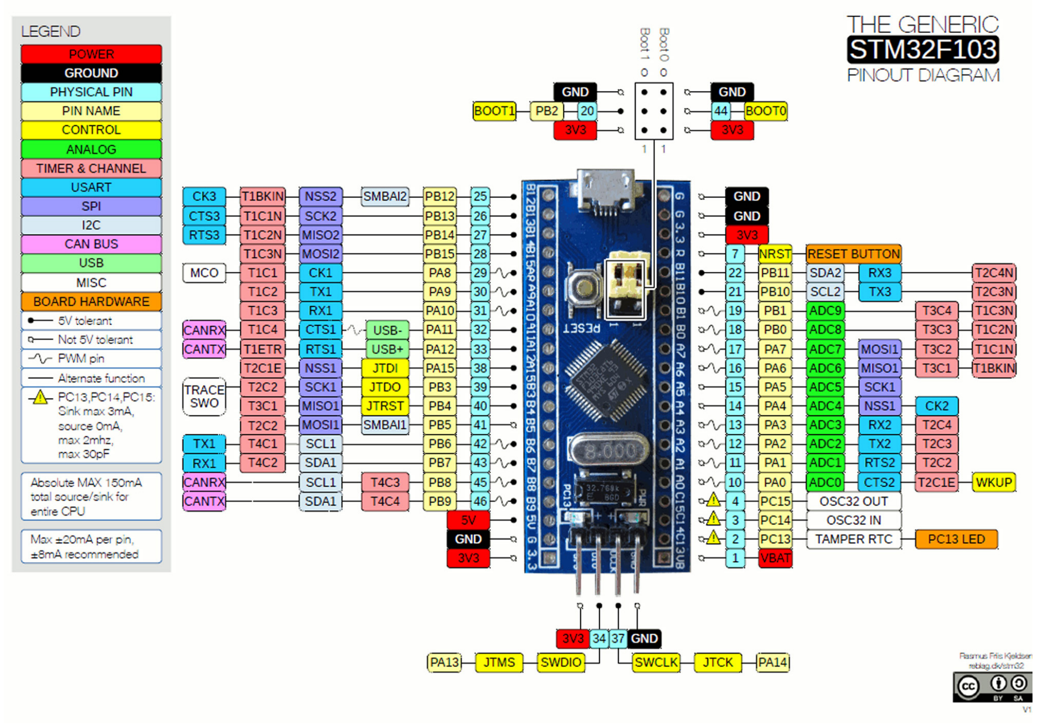


Figure 2.2. STM32f103c8t6 bluepill [?]

various peripherals such as SPI, I2C, UART, and ADC. The ESP32 can be a good choice for a middle station that can receive data from the STM32f103 via UART or SPI and transmit it to the server via Wi-Fi or Bluetooth

2.2 Sensors

! Write smt here

2.3 Software

2.3.1 ASP.NET Web API Core

! write smt here In a air quality dashboard project, a RESTful API [1] can be used to allow weather stations to push data into the server. The weather stations would use the API to send the data in a JSON format, and the server would then store the data in a database. The dashboard would then be able to access the data from the database to display it to users.

RESTful APIs are a popular choice for weather dashboard projects because they are easy to use and scalable. They also allow for the use of different data formats, which makes it easier to integrate with different weather stations.

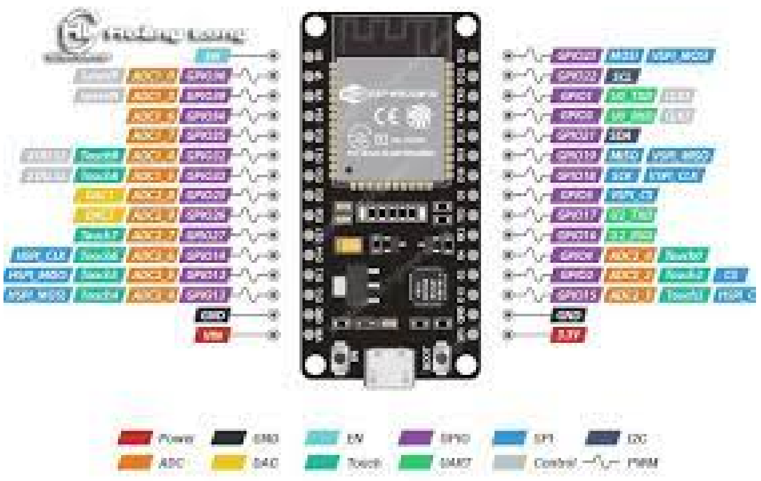


*Figure 2.3. ESP32 MCU [**?**]*

WebSockets are also used. WebSockets a computer communications protocol that enables two-way communication between a client and a server. This makes them ideal for real-time applications such as online chat, gaming, and streaming media. WebSockets are easy to use and widely supported, making them a great option for building real-time applications.

For this project, to synchronize the current measured values from the Weather Station (via the Server). By using the WebSocket API, each time the data is sent from the ESP32.

2.3.2 MySQL Database

! write smt here

2.4 ASP Maui Blazor - ASP Blazor Webassembly

! write smt here

Chapter 3 System Overview

3.1 System layers

This system contains 4 layers [3.4]: Station layer, Gateway layer, Cloud layer, Application layer.

3.1.1 Station layer

At the station layer, the main board is considered as the I2C Master, which requests data from I2C Slaves (STM32 MCU) and send them to the Gateway layer.
Each I2C Slave can stores the air parameters in an communicated bytes array which can be accessed by the Master. An I2C Slave can have one or many sensors but all these values must be stored in the communicated bytes array.
To configuring to the Master MCU, this system introduces a PC's application which can communicate and setup for this Master MCU using command line interface (CLI). Users can use that CLI to configure the slaves addresses, slaves data size or some parameters like ”station id” or publishing interval for this station.

Each time the Master MCU send data to the gateway layer, both the bytes array (which contains all information from the slaves) and the ”station id” (which is setup by the CLI) to next layer.

3.1.2 Gateway layer

This layer contains the ESP32 MCU which has Internet connection. Each time any data from the station is received, MCU extracts that data into ”station id” and byte array and create a POST request to the server to stored that record.

3.1.3 Cloud layer

The server of this project will provide the mechanism for storing and analyze the data from the station. Each station can have its own any type of sensors, users can setup into the station easily which can personalize the users' experience.

3.1.4 Application layer

This layer provide the list of user interfaces: Web application, PC's application, mobile application which users can use to and see the current result.

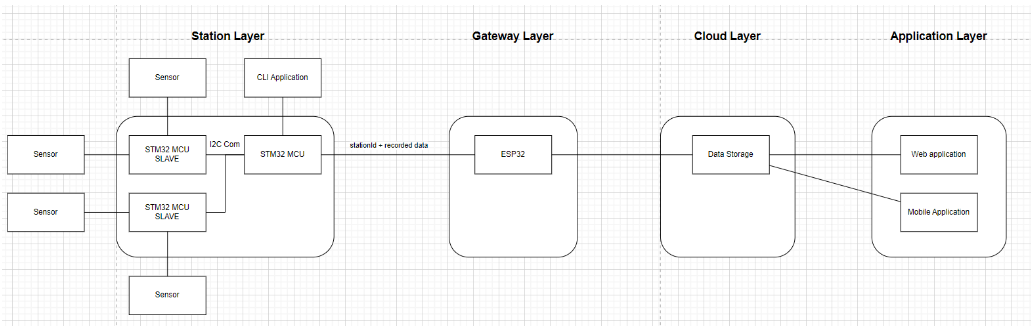


Figure 3.4. Project Layers

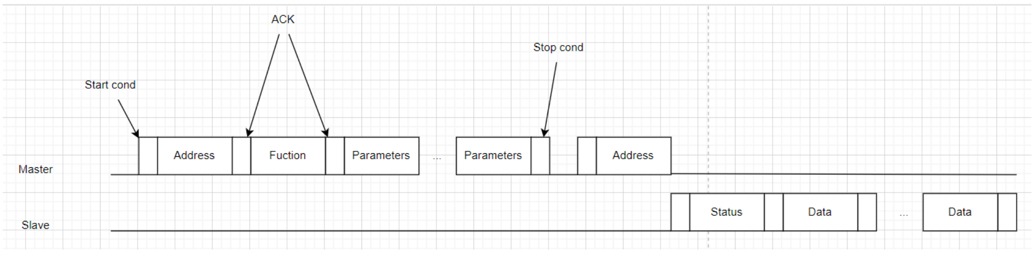
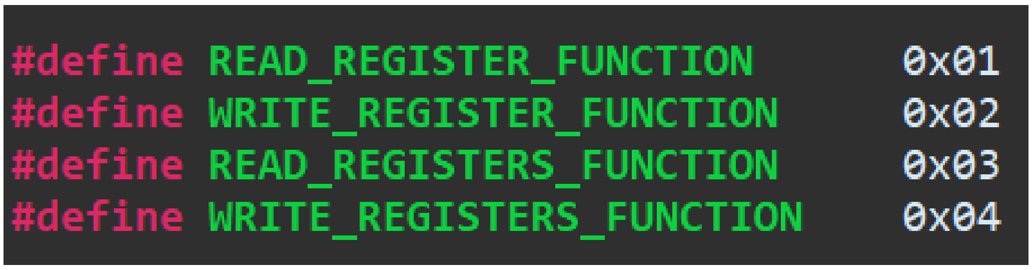


Figure 3.5. I2c protocol

3.2 I2C Protocol

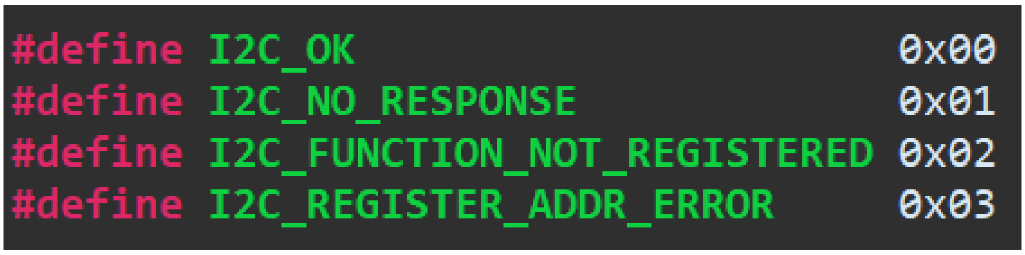
To communicate with between I2C Master MCU with the I2C Slave MCUs, this project provides the protocol [3.5].

With each function [3.6] will have the different set of the parameters. These functions provide the way which the master interacts with the certain registers of the slaves (the parameters is the address or the index of that/those register(s) in the communicated byte array of the slave). The response of the slave contains the status value [3.7]



```
#define READ_REGISTER_FUNCTION        0x01
#define WRITE_REGISTER_FUNCTION       0x02
#define READ_REGISTERS_FUNCTION       0x03
#define WRITE_REGISTERS_FUNCTION      0x04
```

Figure 3.6. List of I2c Functions



```
#define I2C_OK                         0x00
#define I2C_NO_RESPONSE                0x01
#define I2C_FUNCTION_NOT_REGISTERED 0x02
#define I2C_REGISTER_ADDR_ERROR        0x03
```

Figure 3.7. List of I2c Functions

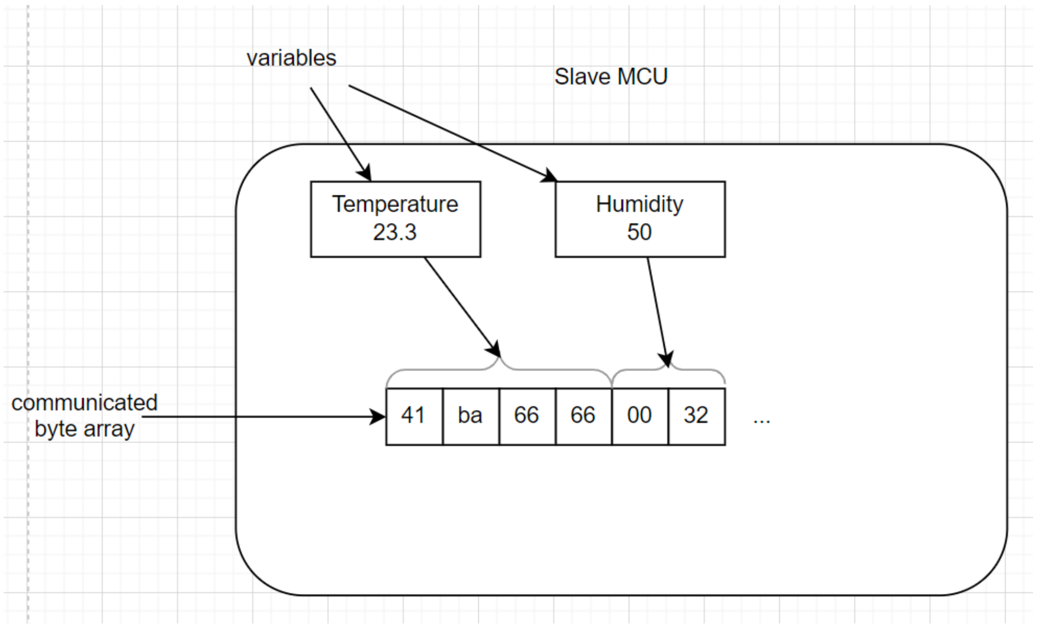


Figure 4.8. The mapping from variables to the communicated byte array

Chapter 4 System Implementation

4.1 Station layer

4.1.1 Slave MCU

The task of the Slave MCU is recording the data from the sensor(s), convert them into communicated byte array [4.8]. When the master MCU want to access any data, it will access via the start index of the MSB byte and the data size of that variable (can be configured for the master MCU via CLI which this thesis will discuss later).

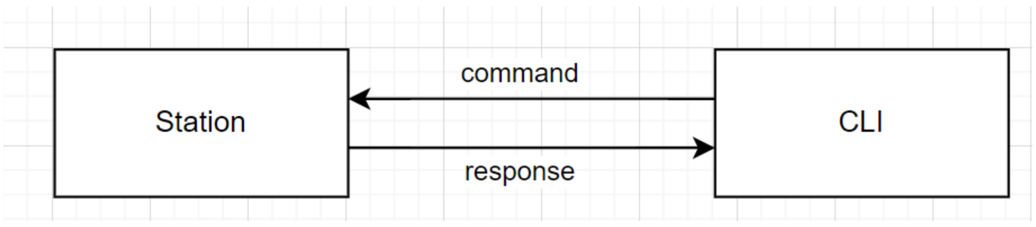


Figure 4.9. Station vs CLI application

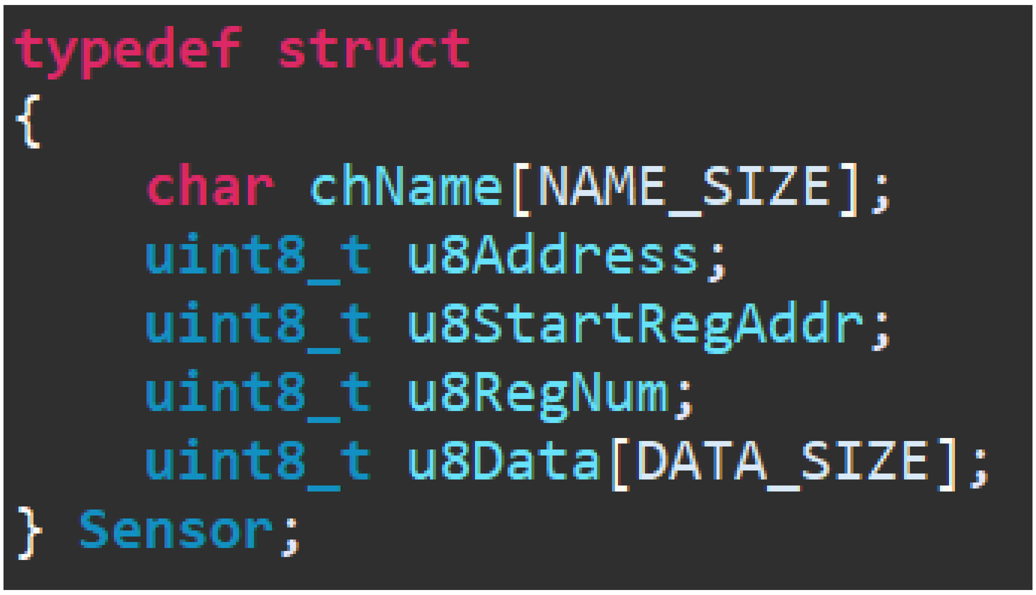
4.1.2 Master MCU

The master MCU can be configured by the user's PC via the CLI. The CLI will send any command from the PC to the MCU, the master MCU must handle these commands and respond back the CLI [4.9]. Users can using ”help” command for getting further information and manual for any command of this station.

For getting the data from slave MCUs, the Master MCU must have the mechanism for dealing with these slaves. To do that, this station defines a Sensor station struct [4.10]. The slaves' information will be saved by the master MCU by the list of the Sensor station struct which can be managed by ”sensors” command from the CLI. Beside, the list of slaves' information can be serialized and deserialized into the byte array which helps the master MCU still keeps the slaves' information after resetting.

Another task of the master MCU is publishing the sampled data to the gateway layer. In this project, the LoRa module is used for this purpose. Each time the MCU gets the data from the slave, it will start to publish these values and the station id to the gateway via LoRa module with the certain format [4.11].

About the station operation, you can follow the state diagram [4.12]



```
typedef struct
{
    char chName[NAME_SIZE];
    uint8_t u8Address;
    uint8_t u8StartRegAddr;
    uint8_t u8RegNum;
    uint8_t u8Data[DATA_SIZE];
} Sensor;
```

Figure 4.10. Struct for saving the slave MCU's information

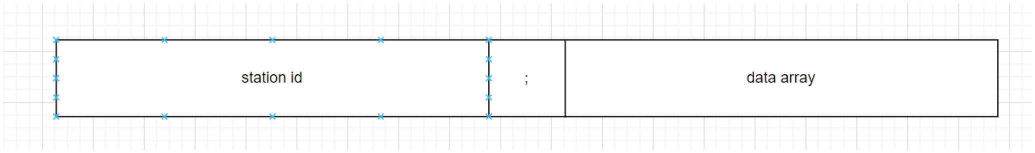


Figure 4.11. Station package which contains 2 parts: station-id and data array

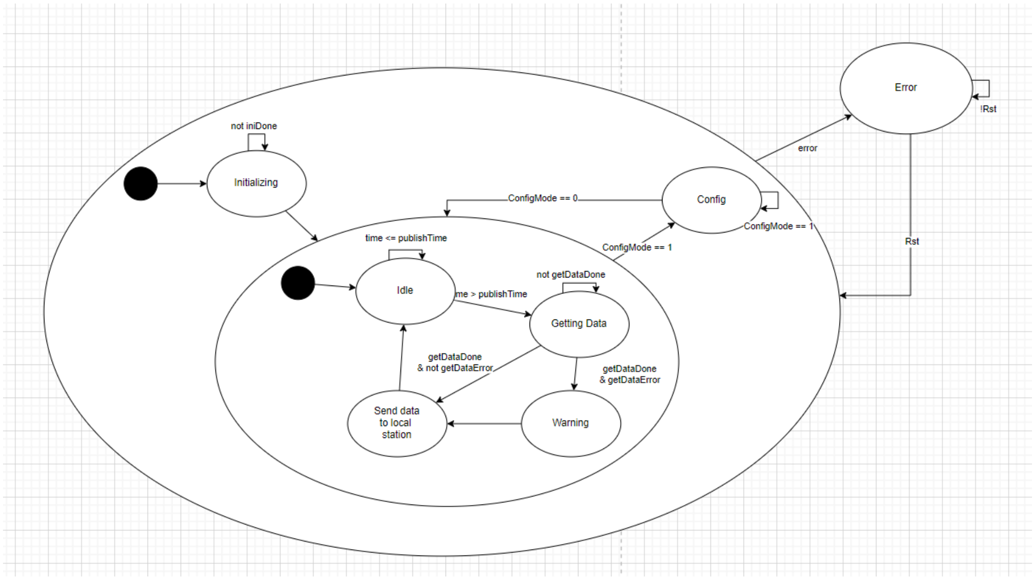


Figure 4.12. Main board diagram

Chapter 5 Result

5.1 Configuring the station id

5.2 Configuring the slave MCU

5.3 Uploaded data

The data is uploaded from the station via the Internet

5.4 API server

All tests inside the server are passed

5.5 UI

5.5.1 Login Page

The login page end-to-end tests

5.5.2 Admin Page

The admin page UI which contains the user's information and the data from the public API

5.5.3 Home Page

The home page displays the current station's weather information and the public API

5.5.4 Station Page

The Station page provides the interface with the station's setting

Chapter 6   Conclusion

!Write smt here

Reference

[1] RESTful API wikipedia