PROJECT INTRODUCTION AND FINAL APPLICATION REPORT

Project Name : Air Quality Monitoring

Group Members :

1. 181805057 – Kardelen Gel
2. 181805067 – Buse Latife Beker

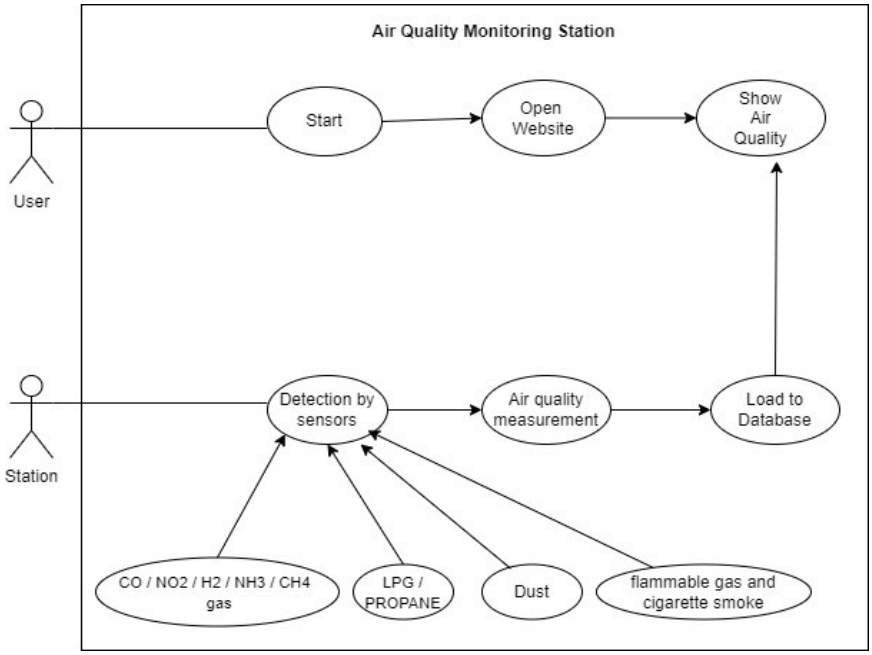
Date : 12.01.2023

1 PROJECT INTRODUCTION

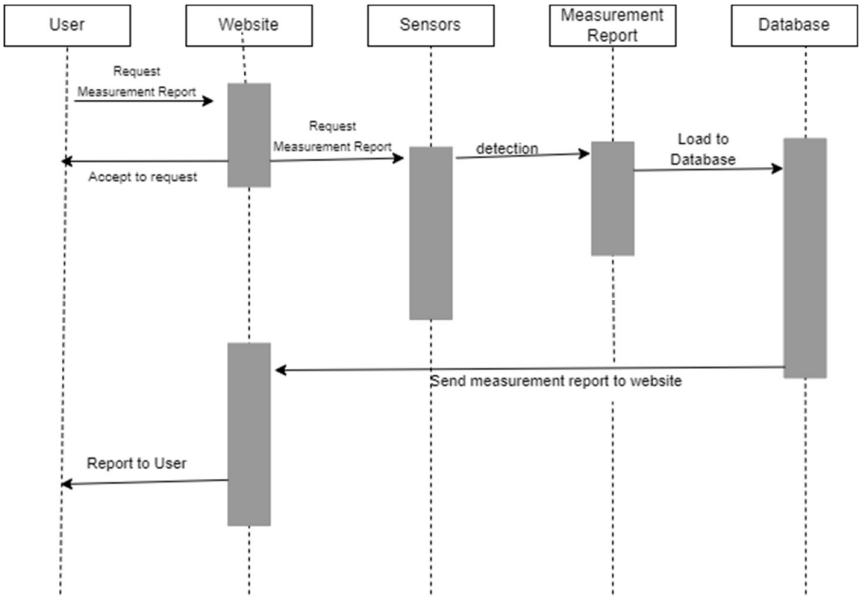
1.1 Goal of Project

The goal of our project is to realize an air quality monitoring station that can detect air pollution in the environment. For this, we used 4 different sensors with which we can detect the gas and dust in the environment. We detected the situations that could cause pollution with the help of these sensors and developed an interface where we can see the outputs of these 4 sensors together.

1.2 Use Case Diagram



1.3 Sequence Diagram

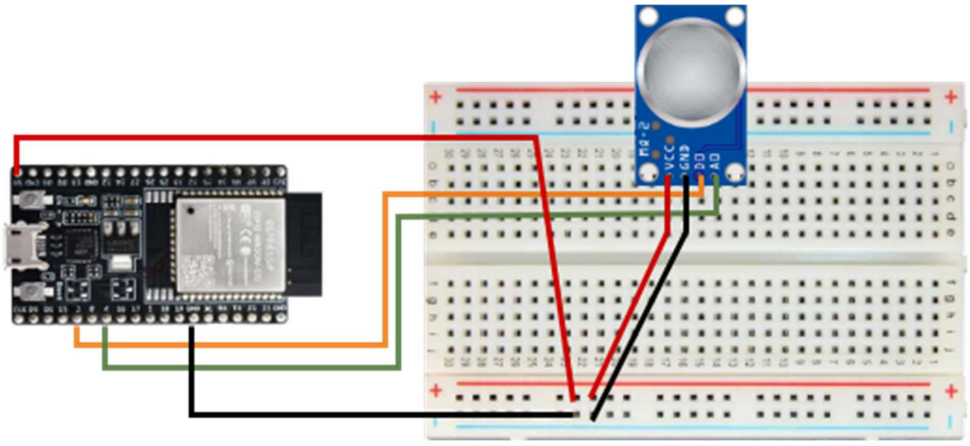


2 PROJECT HARDWARE AND CONNECTION

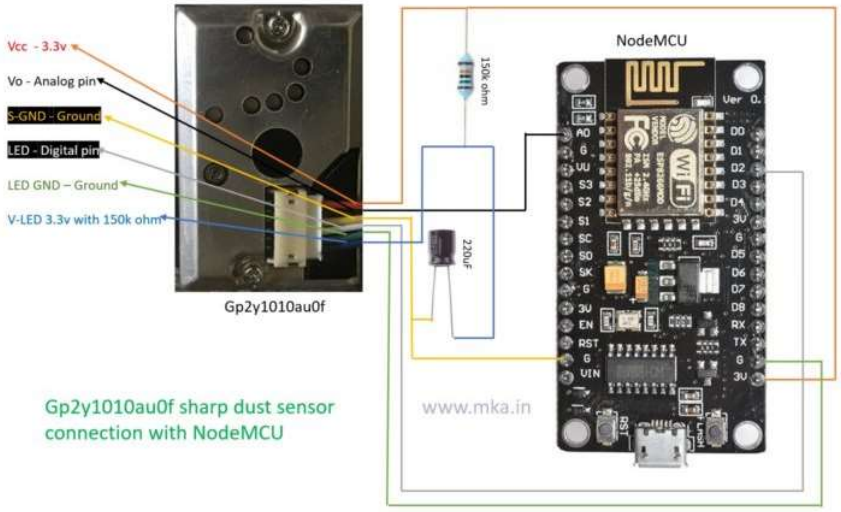
2.1 Hardware Components of Project

#	Name	Number
1	ESP32	2
2	MQ2	1
3	MQ5	1
4	SHARP GP2Y10	1
5	CJMCU-4541	1
6	BreadBoard	1
7	Male-Male, Female-Male, Female-Female Jumper Cable	30
8	Resistor	1
9	Capacitor	1

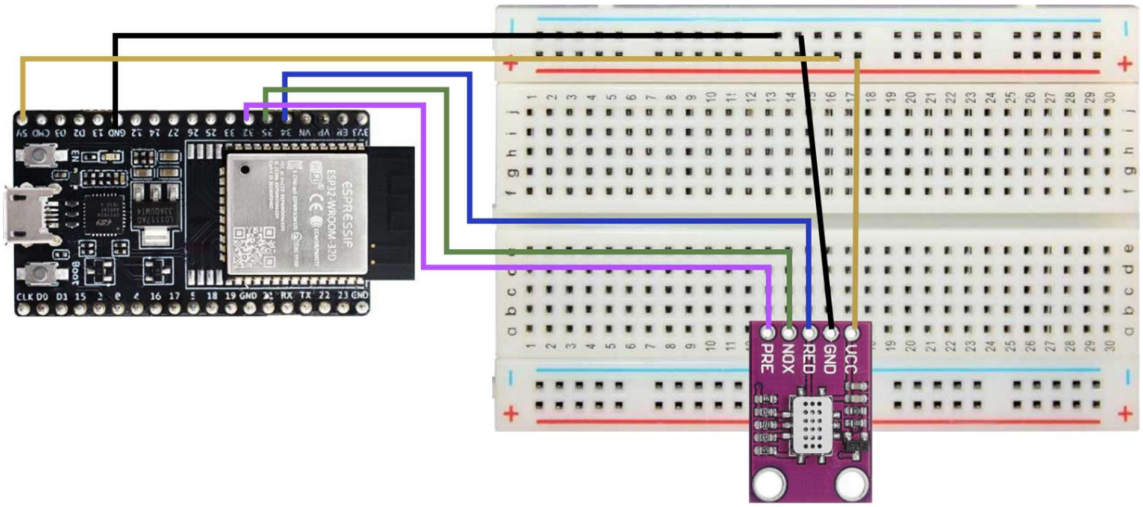
2.2 Node 1 Connection Diagram (MQ-2 & MQ-5)



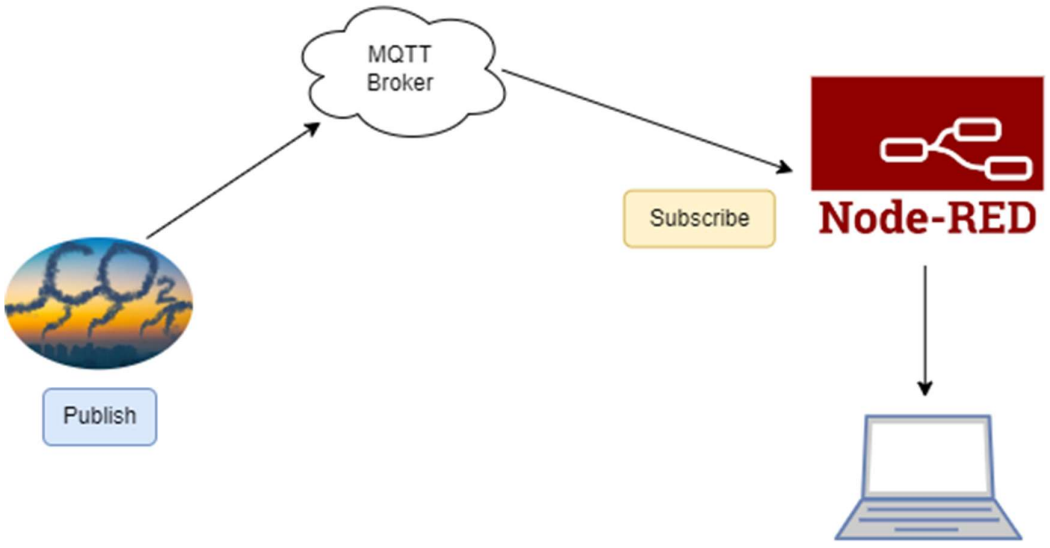
2.3 Node 2 Connection Diagram((SHARP GP2Y10))



2.4 Node 3 Connection Diagram (CJMCU-4541)



3 MQTT DIAGRAM



4 INTERNET CONNECTION

Imports all the required libraries.

```
#include <WiFi.h>
extern "C" {
  #include "freertos/FreeRTOS.h"
  #include "freertos/timers.h"
}
#include <AsyncMqttClient.h>
```

Include network credentials on the following lines.

```
#define WIFI_SSID "REPLACE_WITH_SSID"
#define WIFI_PASSWORD "REPLACE_WITH_PASSWORD"
```

We haven't added any comments to the functions defined in the next code section. Those functions come with the Async Mqtt Client library. The function's names are pretty self-explanatory.

For example, the connectToWifi() connects your ESP32 to your router:

```
void connectToWifi() {
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}
```

The connectToMqtt() connects your ESP32 to your MQTT broker:

```
void connectToMqtt() {
  Serial.println("Connecting to MQTT...");
  mqttClient.connect();
}
```

The WiFiEvent() function is responsible for handling the Wi-Fi events. For example, after a successful connection with the router and MQTT broker, it prints the ESP32 IP address. On the other hand, if the connection is lost, it starts a timer and tries to reconnect.

```
void WiFiEvent(WiFiEvent_t event) {
  Serial.printf("[WiFi-event] event: %d\n", event);
  switch(event) {
    case SYSTEM_EVENT_STA_GOT_IP:
      Serial.println("WiFi connected");
      Serial.println("IP address: ");
      Serial.println(WiFi.localIP());
      connectToMqtt();
      break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
      Serial.println("WiFi lost connection");
      xTimerStop(mqttReconnectTimer, 0);
      xTimerStart(wifiReconnectTimer, 0);
      break;
  }
}
```

The next two lines create timers that will allow both the MQTT broker and Wi-Fi connection to reconnect, in case the connection is lost.

```
mqttReconnectTimer = xTimerCreate("mqttTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToMqtt));
wifiReconnectTimer = xTimerCreate("wifiTimer", pdMS_TO_TICKS(2000), pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToWifi));
```

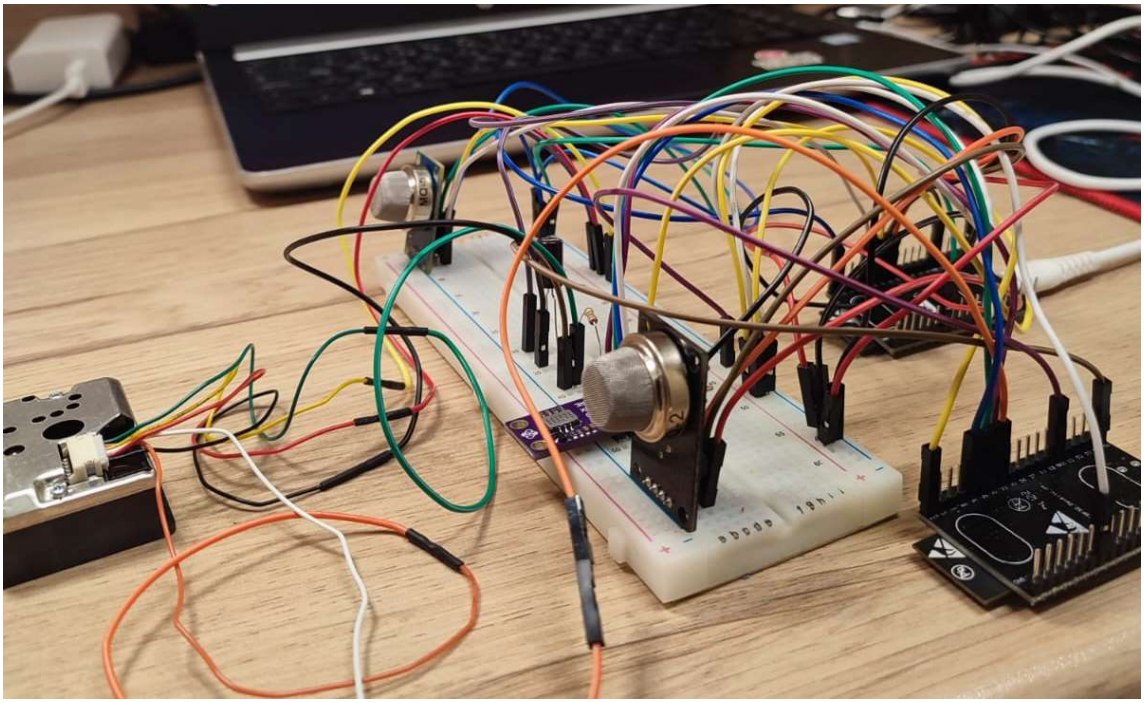
The following line assigns a callback function, so when the ESP32 connects to your Wi-Fi, it will execute the WiFiEvent() function to print the details described earlier.

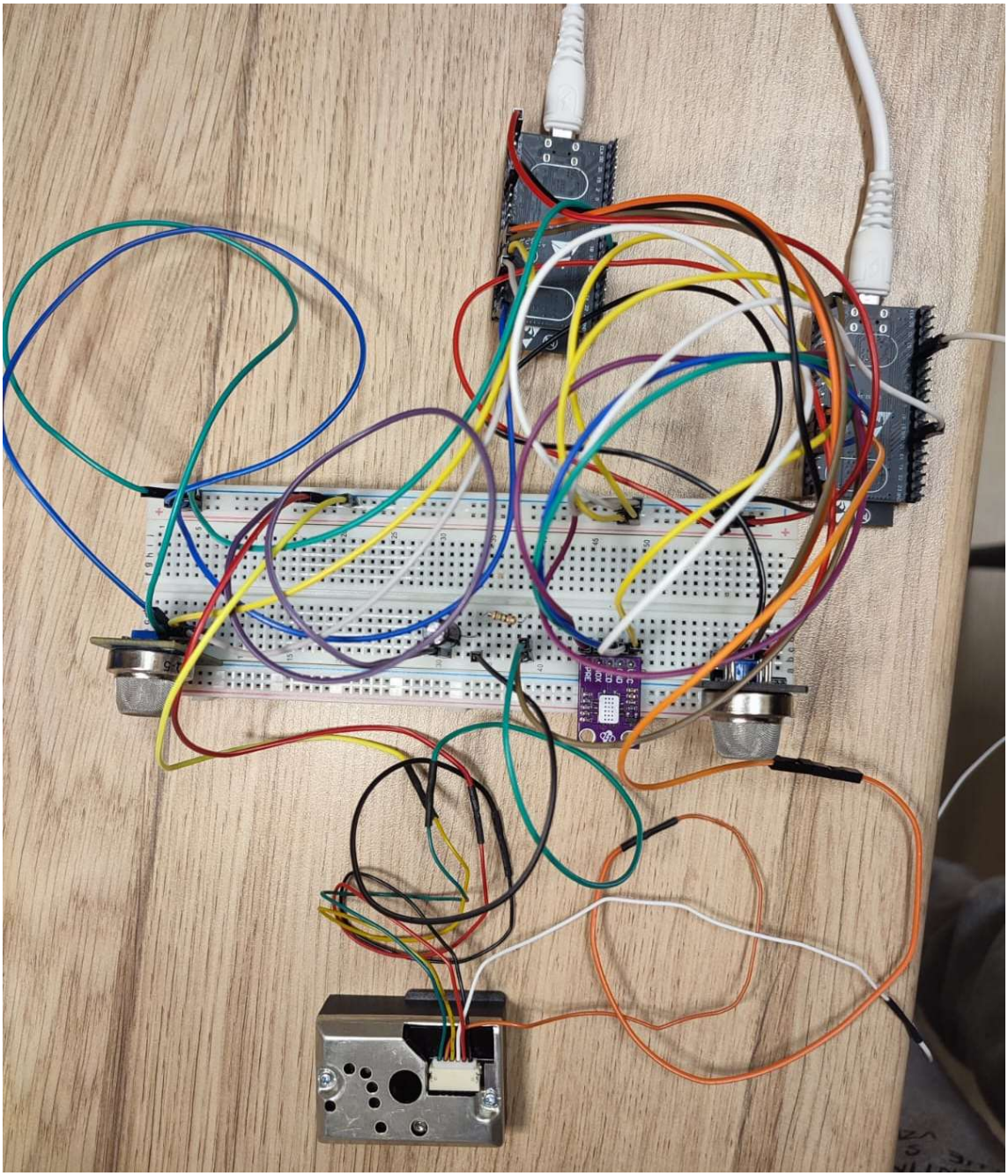
```
WiFi.onEvent(WiFiEvent);
```

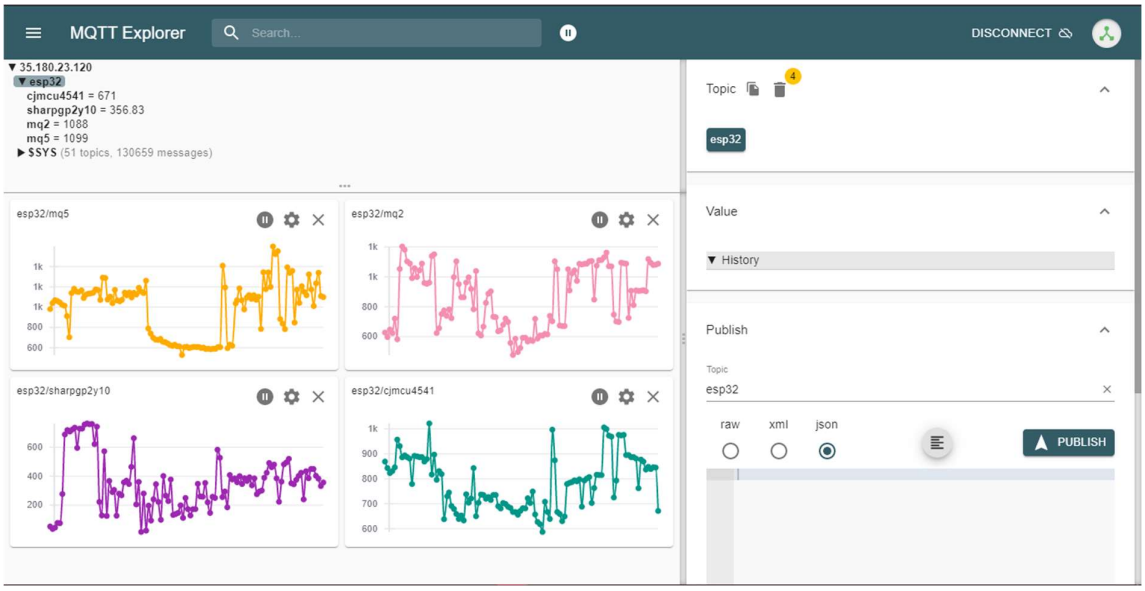
Finally, connect to Wi-Fi.

```
connectToWifi();
```

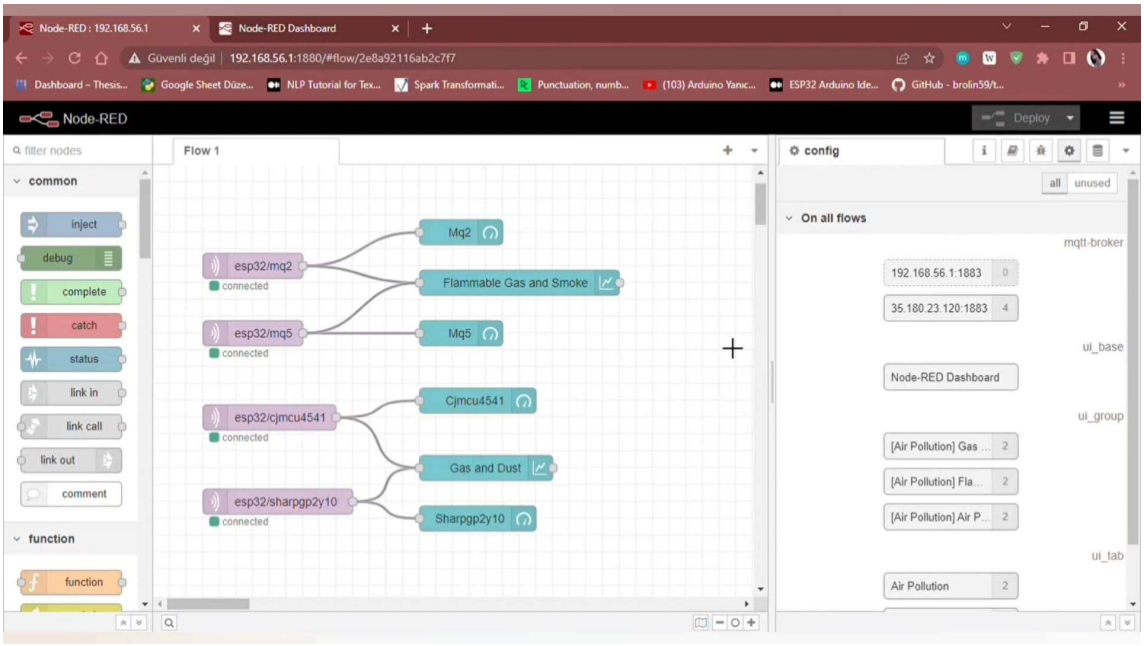
5 OPERATION OF SYSTEM



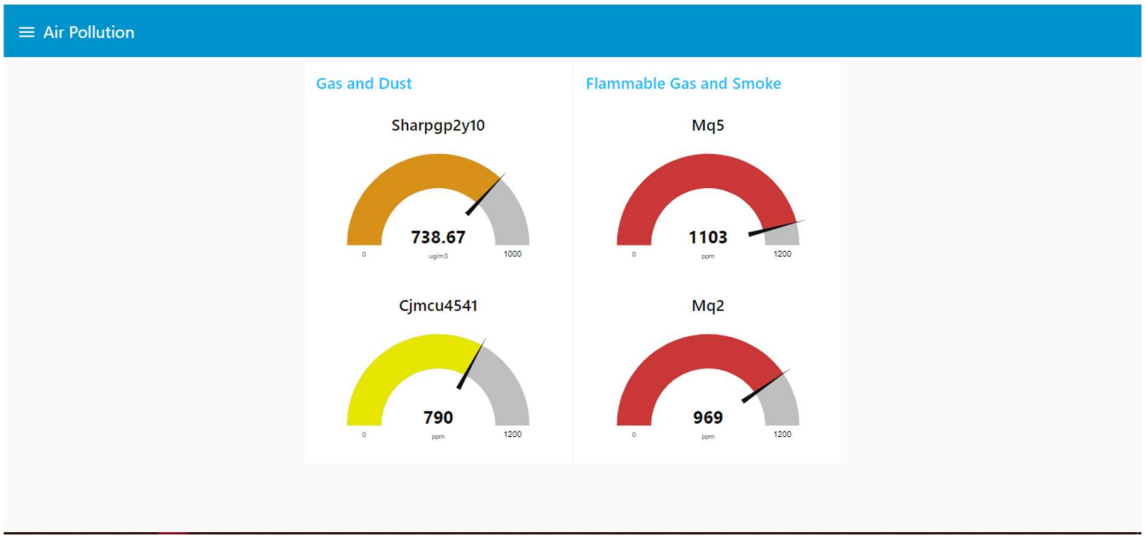




Node-RED Flow Diagram



Displaying the values detected by the sensors with specific colors and indicators



Graph of change of values in a certain time period