Air Quality Monitoring using IoT Phase-5

Project Objectives:

Air quality monitoring using IoT is a process of collecting and analyzing data on the quality of air in a specific area using Internet of Things (IoT) devices. The objective of air quality monitoring using IoT is to provide real-time data on air quality, which can be used to make informed decisions to best manage and improve the environment 1. IoT-based air pollution monitoring systems can help in preventing the negative impact of pollutants on human health by providing authorities and citizens with important information about the current level of gases and particles in different areas of the city 2. These systems work alongside traditional air quality monitoring technology to collect greater volumes of granular data, including temperature, humidity, altitude, atmospheric pressure, carbon dioxide levels, as well as pollutants like methane, carbon monoxide, and ammonium 3.

IoT Device Setup:

Setting up an IoT device for air quality monitoring involves selecting appropriate sensors, a microcontroller, and establishing a secure data transmission mechanism. Below are the key steps for setting up the IoT device:

1. **Sensor Selection:**
   - Choose sensors capable of detecting various air pollutants such as particulate matter (PM), carbon monoxide (CO), nitrogen dioxide (NO2), sulfur dioxide (SO2), ozone (O3), and volatile organic compounds (VOCs).
   - Opt for sensors with high accuracy, reliability, and sensitivity to ensure precise data collection.

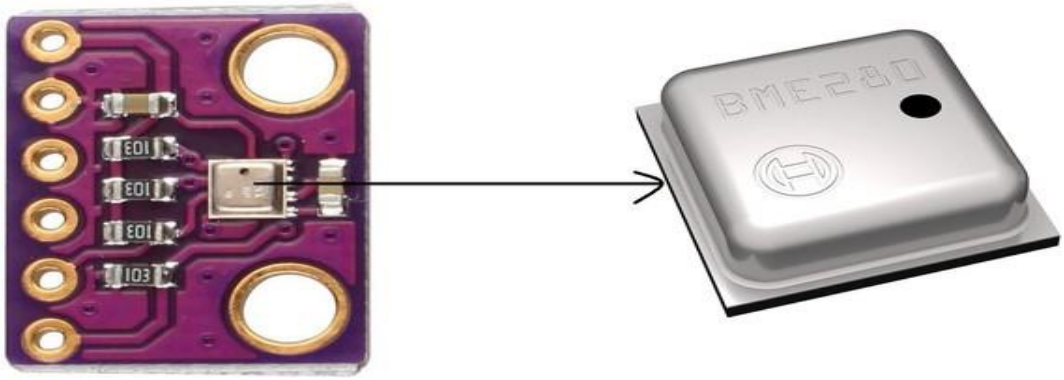
2. MQ-135 Air Quality Sensor:

The MQ-135 gas sensor senses the gases like ammonia nitrogen, oxygen, alcohols, aromatic compounds, sulfide and smoke. The MQ-3 gas sensor has a lower conductivity to clean the air as a gas sensing material. In the atmosphere, we can find polluting gases, but the conductivity of the gas sensor increases as the concentration of polluting gas increases. MQ-135 gas sensor can be implemented to detect the smoke, benzene, steam, and other harmful gases. It has the potential to detect different harmful gases. It is at a low cost and particularly suitable for Air quality monitoring applications.

3. PMS5003 PM2.5 Particulate Matter Sensor

The Plantower PMS5003 is a low-cost laser particle counter, one of a range of sensors by Plantower that also include the PMS1003, PMS3003, and PMS7003. PMS5003 is a kind of digital and universal particle concentration sensor, which can be used to obtain the number of suspended particles in the air, i.e. the concentration of particles, and output them in the form of a digital interface. This sensor can be inserted into variable instruments related to the concentration of suspended particles in the air or other environmental improvement equipment to provide correct concentration data in time.

4. **BME280 Barometric Pressure Sensor**



Bosch BME280 Humidity, Temperature & Pressure Sensor is an integrated environmental sensor which is very small-sized with low power consumption.This BME280 Atmospheric Sensor Breakout is the easy way to measure barometric pressure, humidity, and temperature readings all without taking up too much space. Basically, anything you need to know about atmospheric conditions you can find out from this tiny breakout.This module uses an environmental sensor manufactured by Bosch with temperature, barometric pressure sensor that is the next generation upgrade to the popular BMP085/BMP180/BMP183 Sensor. This sensor is great for all sorts of weather sensing and can even be used in both I2C and SPI! This precision sensor from Bosch is the best low-cost, precision sensing solution for measuring barometric pressure with ±1 hPa absolute accuracy, and temperature with ±1.0°C accuracy. Because pressure changes with altitude and the pressure measurements are so good, you can also use it as an altimeter with ±1 meter accuracy..

5. **Testing and Calibration:**

- Conduct thorough testing of the IoT device setup to validate the functionality and accuracy of the sensors, microcontroller, and data transmission mechanisms.
- Calibrate the sensors periodically to maintain data accuracy and reliability, ensuring that the measurements align with standard air quality monitoring guidelines.
- By following these steps, you can set up an efficient and reliable IoT device for air quality monitoring, capable of collecting accurate data and transmitting it securely to the designated data processing and visualization platform.

Platform Development:

Developing a platform for air quality monitoring using IoT involves several key steps. Here's a high-level overview of the project

1.Hardware Setup:

- Choose suitable IoT devices such as sensors, microcontrollers, and communication modules to measure air quality parameters like particulate matter, gases, and humidity.

2.Data Acquisition:

- Implement a system to collect data from the sensors. This could involve using communication protocols like MQTT or HTTP to transmit data to a cloud server.

3.Cloud Infrastructure:

- Set up a cloud-based infrastructure to receive, store, and process the collected data. Services like AWS IoT, Azure IoT, or Google Cloud IoT can be used for this purpose.

4.Data Processing and Analysis:

- Implement algorithms to process and analyze the collected data. This could involve data filtering, aggregation, and the use of machine learning techniques for predictive analysis.

5.User Interface:

- Develop a user-friendly interface, such as a web or mobile application, to display real-time and historical air quality data. This interface should allow users to visualize data trends and receive alerts for abnormal air quality levels.

6.Alerts and Notifications:

- Implement an alert system to notify users and stakeholders about critical changes in air quality, enabling them to take timely actions.

7. Integration and Compatibility:

- Ensure compatibility with various devices and platforms, enabling easy integration with existing systems and services.

8.Security Measures:

- Implement robust security protocols to protect sensitive data and ensure the platform's security from potential cyber threats.

9.Regulatory Compliance:

- Adhere to relevant regulations and standards in the domain of air quality monitoring to ensure that the platform meets necessary legal requirements.

10. Maintenance and Support:

- Provide regular maintenance and support for the platform to ensure its seamless operation and longevity.

By following these steps, you can create a comprehensive and effective platform for air quality monitoring using IoT.

SOFTWARE REQUIREMNTS:

1. PYTHON:

We implement a python program for to import IoT devices record some activites using the required hardware analyze the result of data daily.

2. HTML,CSS,JS:

Here HTML is used to develop a web interface app for all platform interfaces.

ADVANTAGES:

- IoT-based air quality monitoring systems offer several benefits over traditional monitoring systems.
- They provide real-time data on air quality, which can help in identifying pollution hotspots and taking immediate action to mitigate the problem.
- They are cost-effective and require fewer sensors than traditional monitoring systems.
- They use advanced sensors that are more accurate than traditional sensors, providing more precise data on air quality.
- They can collect large amounts of data, which can be analyzed to identify trends and patterns in air quality over time.
- They are easy to install and use, and can be operated remotely using a smartphone or computer.

METHOD:

To monitor air quality using IoT, you can use a microcontroller such as Arduino Uno or NodeMCU. These microcontrollers can be programmed to take input from sensors and transmit the data to the cloud 1234.

For instance, you can use an MQ135 gas sensor to detect harmful gases such as CO2, smoke, alcohol, benzene and NH3. You can then connect the sensor to an Arduino Uno along with a Wi-Fi module such as ESP8266. The data collected by the sensor can be transmitted to a web server over the internet and displayed on a webpage or mobile app in real-time 1.

Here is a list of components that you will need for this project:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module (such as ESP8266)
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor

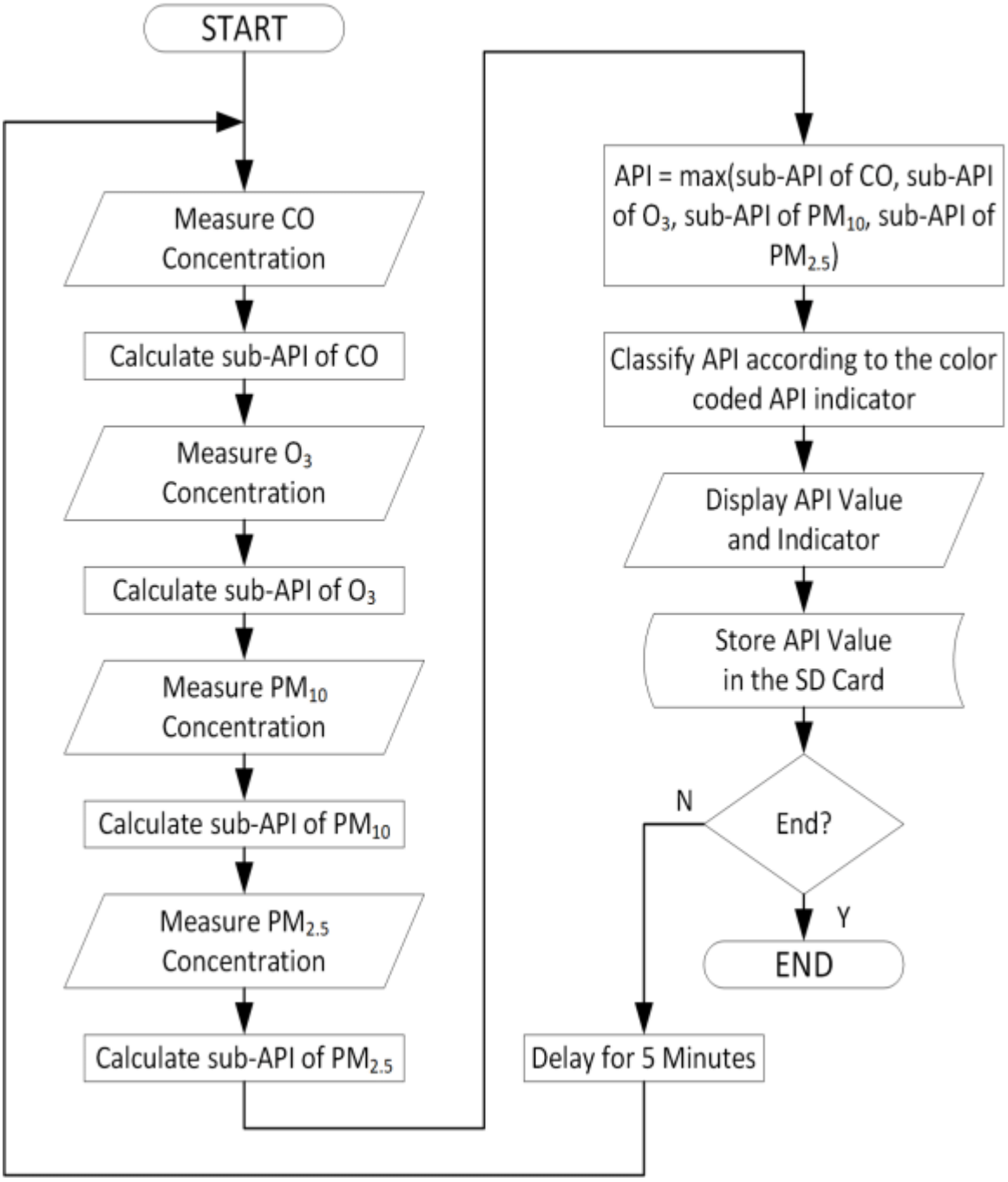
- Buzzer

You can buy all the above components from online stores. Once you have all the components, you can follow the circuit diagram and explanation provided in this tutorial 1.

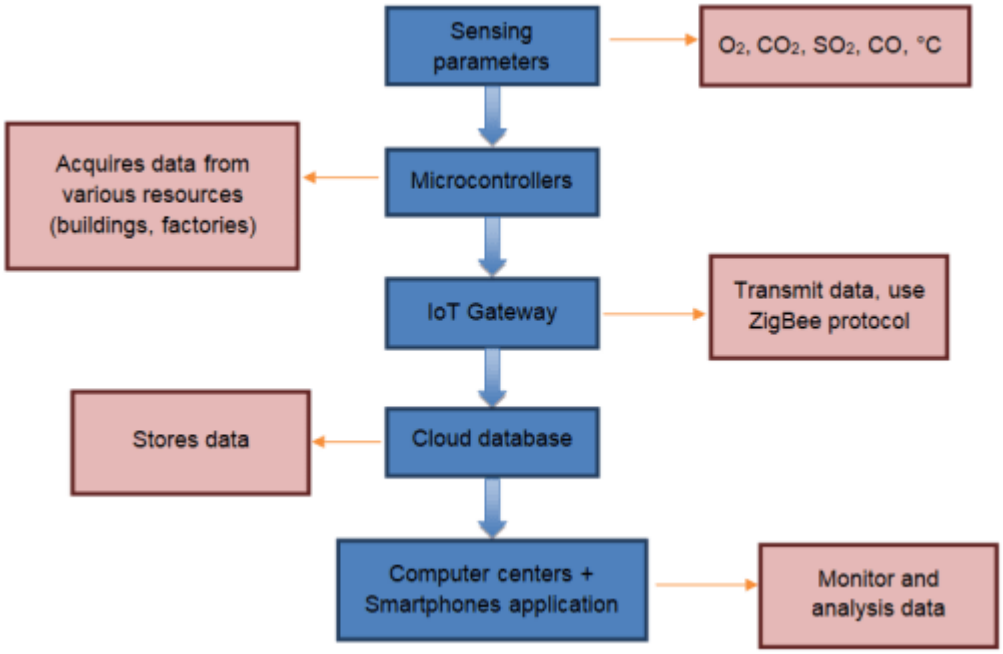
IoT-based air quality monitoring systems offer several benefits over traditional monitoring systems. They provide real-time data on air quality, which can help in identifying pollution hotspots and taking immediate action to mitigate the problem. They are cost-effective and require fewer sensors than traditional monitoring systems. They use advanced sensors that are more accurate than traditional sensors, providing more precise data on air quality. They can collect large amounts of data, which can be analyzed to identify trends and patterns in air quality over time. They are easy to install and use, and can be operated remotely using a smartphone or computer 1234**.**

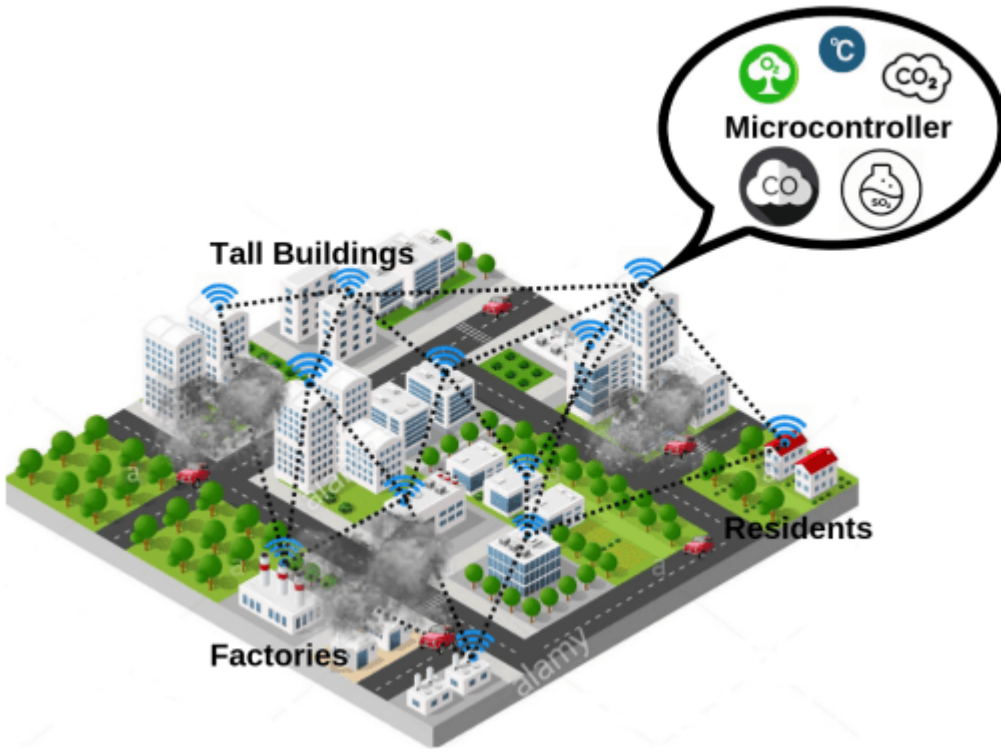
VIEW OF PROJECT:

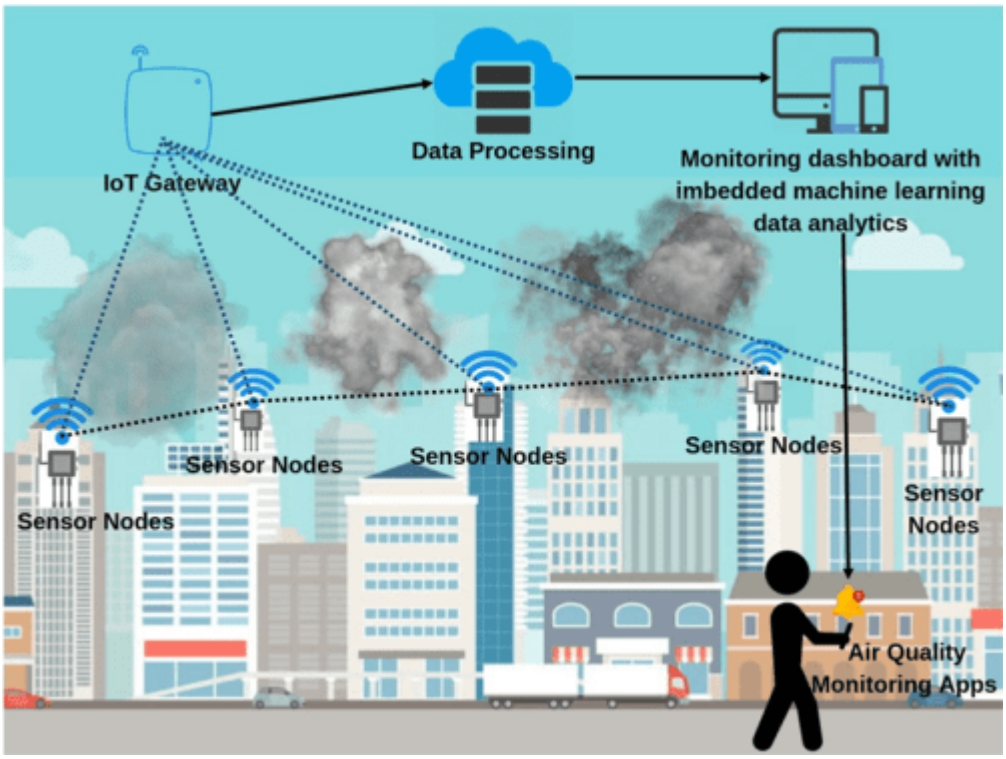
1. FLOW CHART



2. Design Overthink







CODE IMPLEMENTATION:

1. PYTHON CODE

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include "MQ135.h"
#include <Arduino.h>
#define LENG 31   //0x42 + 31 bytes equal to 32 bytes
unsigned char buf[LENG];

int PM01Value=0;          //define PM1.0 value of the air detector module
int PM2_5Value=0;         //define PM2.5 value of the air detector module
int PM10Value=0;         //define PM10 value of the air detector module
float h, t, p, pin, dp;
char temperatureFString[6];
char dpString[6];
char humidityString[6];
```

```
18.char pressureString[7];
19.char pressureInchString[6];
20.
21.Adafruit_BME280 bme; // I2C
22.String apiKey = "G2ZWC6T4HH0G45O7";
23.// replace with your routers SSID
24.const char* ssid = "IOT";
25.// replace with your routers password
26.const char* password = "AIR123";
27.
28.const char* server =
   "https://api.thingspeak.com/channels/2327392/fields/1.json?api_key=G2ZWC6T4HH0G45
   O7&results=2";
29.WiFiClient client;
30.
31.void setup()
32.{
33.  Serial.begin(9600);
34.  delay(10);
35.  Serial.println();
36.  Serial.print("Connecting to ");
37.  Serial.println(ssid);
38.  WiFi.begin(ssid, password);
39.
40.  while (WiFi.status() != WL_CONNECTED) {
41.    delay(500);
42.    Serial.print(".");
43.  }
44.  Serial.println("");
45.  Serial.println("WiFi connected");
46.
47.    // Printing the ESP IP address
48.  Serial.println(WiFi.localIP());
49.
50.  if (!bme.begin())
51.  {
52.    Serial.println("Could not find a valid BME280 sensor, check wiring!");
53.    while (1);
54.}
55.}
56.
57.void loop()
58.{
59.  if(Serial.find(0x42)){    //start to read when detect 0x42
60.    Serial.readBytes(buf,LENG);
```

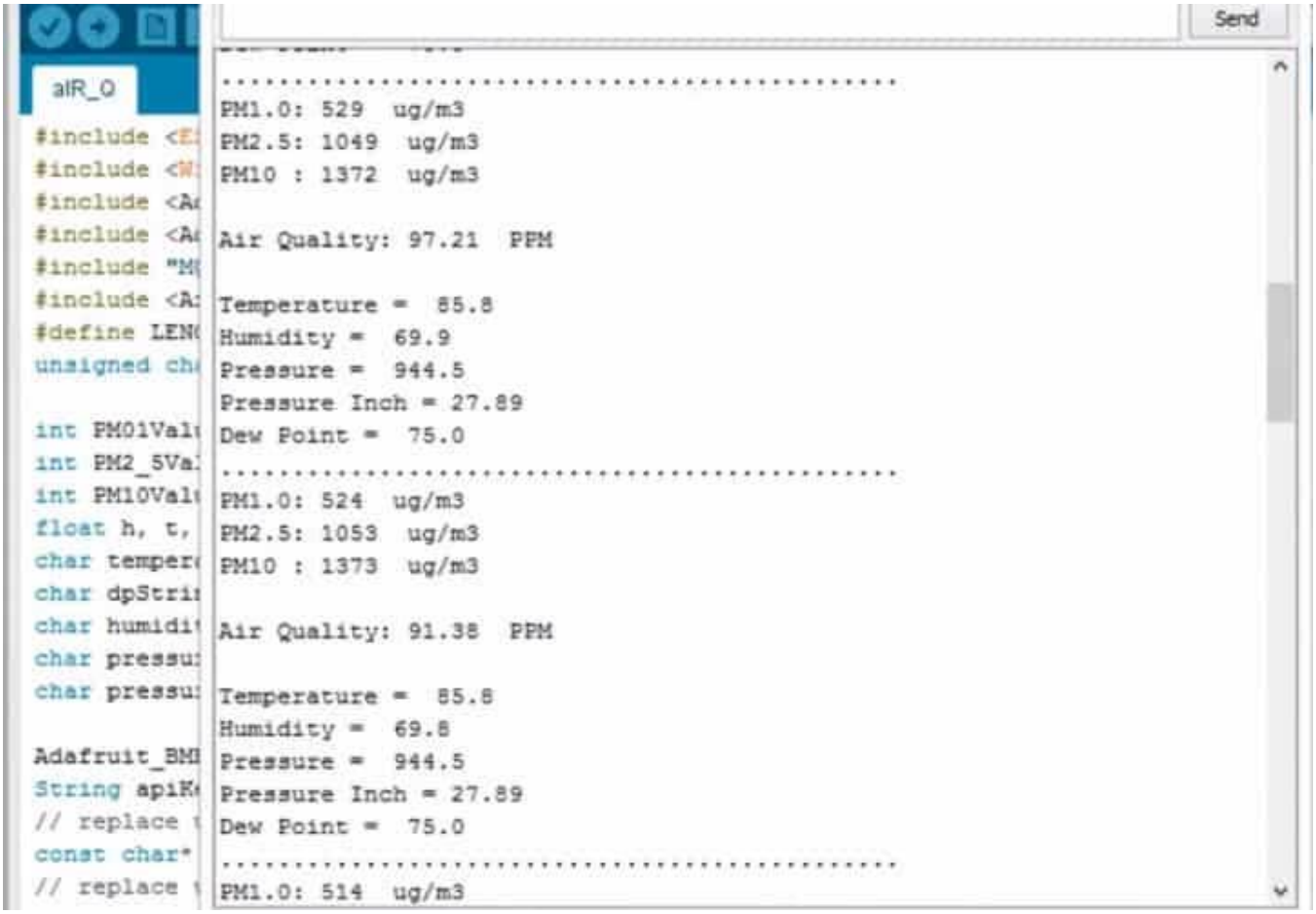
```
61.
62.    if(buf[0] == 0x4d){
63.      if(checkValue(buf,LENG)){
64.        PM01Value=transmitPM01(buf); //count PM1.0 value of the air detector
   module
65.        PM2_5Value=transmitPM2_5(buf);//count PM2.5 value of the air detector
   module
66.        PM10Value=transmitPM10(buf); //count PM10 value of the air detector
   module
67.      }
68.    }
69.  }
70.
71.  static unsigned long OledTimer=millis();
72.    if (millis() - OledTimer >=1000)
73.    {
74.      OledTimer=millis();
75.
76.      Serial.print("PM1.0: ");
77.      Serial.print(PM01Value);
78.      Serial.println("  ug/m3");
79.
80.      Serial.print("PM2.5: ");
81.      Serial.print(PM2_5Value);
82.      Serial.println("  ug/m3");
83.
84.      Serial.print("PM10 : ");
85.      Serial.print(PM10Value);
86.      Serial.println("  ug/m3");
87.      Serial.println();
88.
89.      MQ135 gasSensor = MQ135(A0);
90.      float air_quality = gasSensor.getPPM();
91.      Serial.print("Air Quality: ");
92.      Serial.print(air_quality);
93.      Serial.println("  PPM");
94.      Serial.println();
95.
96.      h = bme.readHumidity();
97.      t = bme.readTemperature();
98.      t = t*1.8+32.0;
99.      dp = t-0.36*(100.0-h);
100.
101.           p = bme.readPressure()/100.0F;
102.           pin = 0.02953*p;
```

```
            dtostrf(t, 5, 1, temperatureFString);
            dtostrf(h, 5, 1, humidityString);
            dtostrf(p, 6, 1, pressureString);
            dtostrf(pin, 5, 2, pressureInchString);
            dtostrf(dp, 5, 1, dpString);

            Serial.print("Temperature = ");
            Serial.println(temperatureFString);
            Serial.print("Humidity = ");
            Serial.println(humidityString);
            Serial.print("Pressure = ");
            Serial.println(pressureString);
            Serial.print("Pressure Inch = ");
            Serial.println(pressureInchString);
            Serial.print("Dew Point = ");
            Serial.println(dpString);

            Serial.println("..........................................");

            if (client.connect(server,80))  // "184.106.153.149" or
   https://api.thingspeak.com/channels/2327392/fields/1.json?api_key=G2ZWC6T4HH0G45O
   7&results=2
          {
              String postStr = apiKey;
              postStr +="&field1=";
              postStr += String(PM01Value);
              postStr +="&field2=";
              postStr += String(PM2_5Value);
              postStr +="&field3=";
              postStr += String(PM10Value);
              postStr +="&field4=";
              postStr += String(air_quality);
              postStr +="&field5=";
              postStr += String(temperatureFString);
              postStr +="&field6=";
              postStr += String(humidityString);
              postStr +="&field7=";
              postStr += String(pressureInchString);
              postStr += "\r\n\r\n";

              client.print("POST /update HTTP/1.1\n");
              client.print("Host:https://api.thingspeak.com/channels/2327392/fie
   lds/1.json?api_key=G2ZWC6T4HH0G45O7&results=2\n");
              client.print("Connection: close\n");
              client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
```

```
145.              client.print("Content-Type: application/x-www-form-urlencoded\n");
146.              client.print("Content-Length: ");
147.              client.print(postStr.length());
148.              client.print("\n\n");
149.              client.print(postStr);
150.            }
151.            client.stop();
152.
153.
154.            }
155.
156.          }
157.        char checkValue(unsigned char *thebuf, char leng)
158.        {
159.          char receiveflag=0;
160.          int receiveSum=0;
161.
162.          for(int i=0; i<(leng-2); i++){
163.          receiveSum=receiveSum+thebuf[i];
164.          }
165.          receiveSum=receiveSum + 0x42;
166.
167.          if(receiveSum == ((thebuf[leng-2]<<8)+thebuf[leng-1]))  //check the
   serial data
168.          {
169.            receiveSum = 0;
170.            receiveflag = 1;
171.          }
172.          return receiveflag;
173.        }
174.        int transmitPM01(unsigned char *thebuf)
175.        {
176.          int PM01Val;
177.          PM01Val=((thebuf[3]<<8) + thebuf[4]); //count PM1.0 value of the air
   detector module
178.          return PM01Val;
179.        }
180.        //transmit PM Value to PC
181.        int transmitPM2_5(unsigned char *thebuf)
182.        {
183.          int PM2_5Val;
184.          PM2_5Val=((thebuf[5]<<8) + thebuf[6]);//count PM2.5 value of the air
   detector module
185.          return PM2_5Val;
186.          }
```

```
187.      //transmit PM Value to PC
188.      int transmitPM10(unsigned char *thebuf)
189.      {
190.        int PM10Val;
191.        PM10Val=((thebuf[7]<<8) + thebuf[8]); //count PM10 value of the air
   detector module
192.        return PM10Val;
193.      }
```

SCREENSHOT:



APP DEVELOPMENT USING HTML,CSS,JS:

1. HTML:

```
<!DOCTYPE html>

<html>
<head>
    <title>Air Quality Monitor</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript"
src="https://unpkg.com/jquery@3.3.1/dist/jquery.js"></script>
    <script type="text/javascript"
src="https://unpkg.com/web3@0.20.5/dist/web3.min.js"></script>

</head>
<body>
    <h1>Air Quality Monitor</h1>
    <div id="data-container">
        <div class="data-box">
            <h2>Temperature (°C)</h2>
            <p id="temperature">Loading...</p>
        </div>
        <div class="data-box">
            <h2>Humidity (%)</h2>
            <p id="humidity">Loading...</p>
        </div>
        <div class="data-box">
            <h2>Gas Resistance (Ohms)</h2>
            <p id="gas">Loading...</p>
        </div>
    </div>

    <script src="server.js"></script>
    <script src="script.js"></script>
    <script src="https://cdn.socket.io/socket.io-1.2.0.js"></script>
</body>
</html>
```

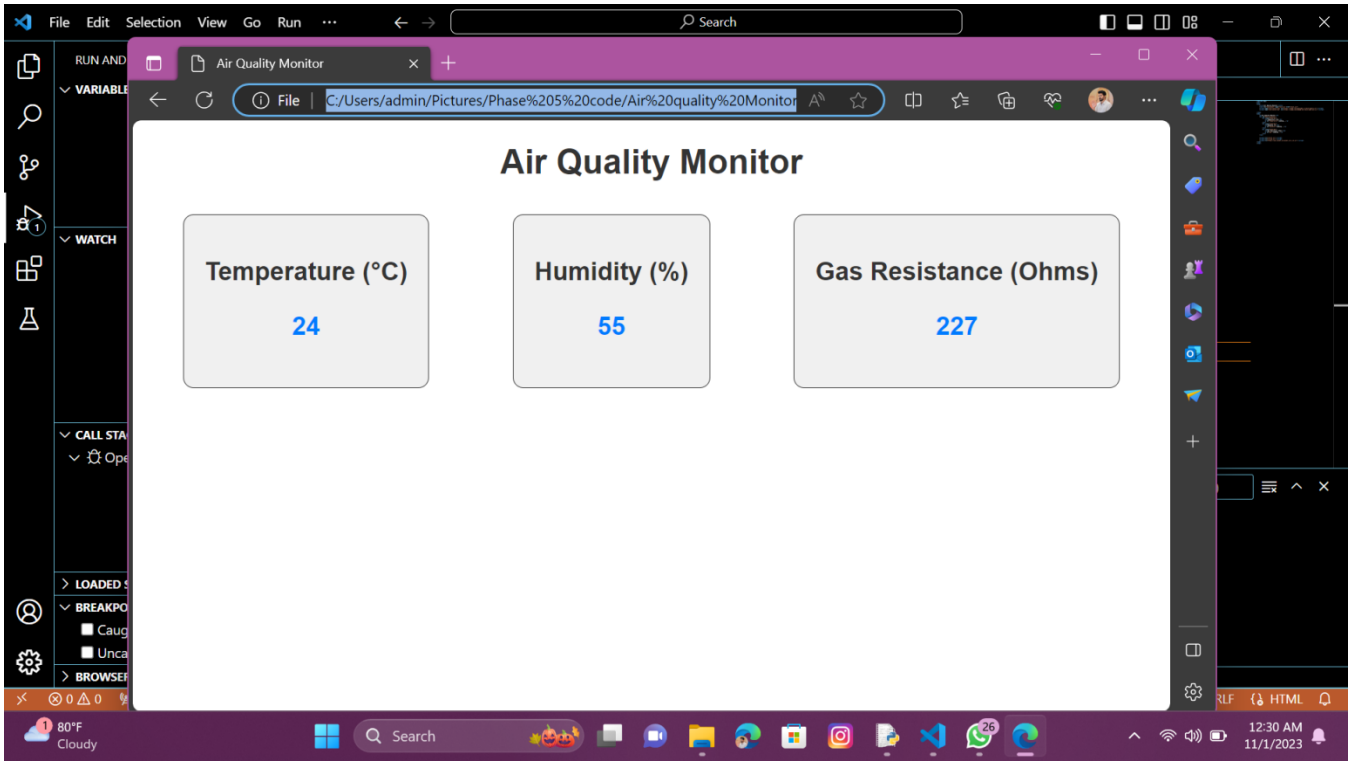
2. CSS

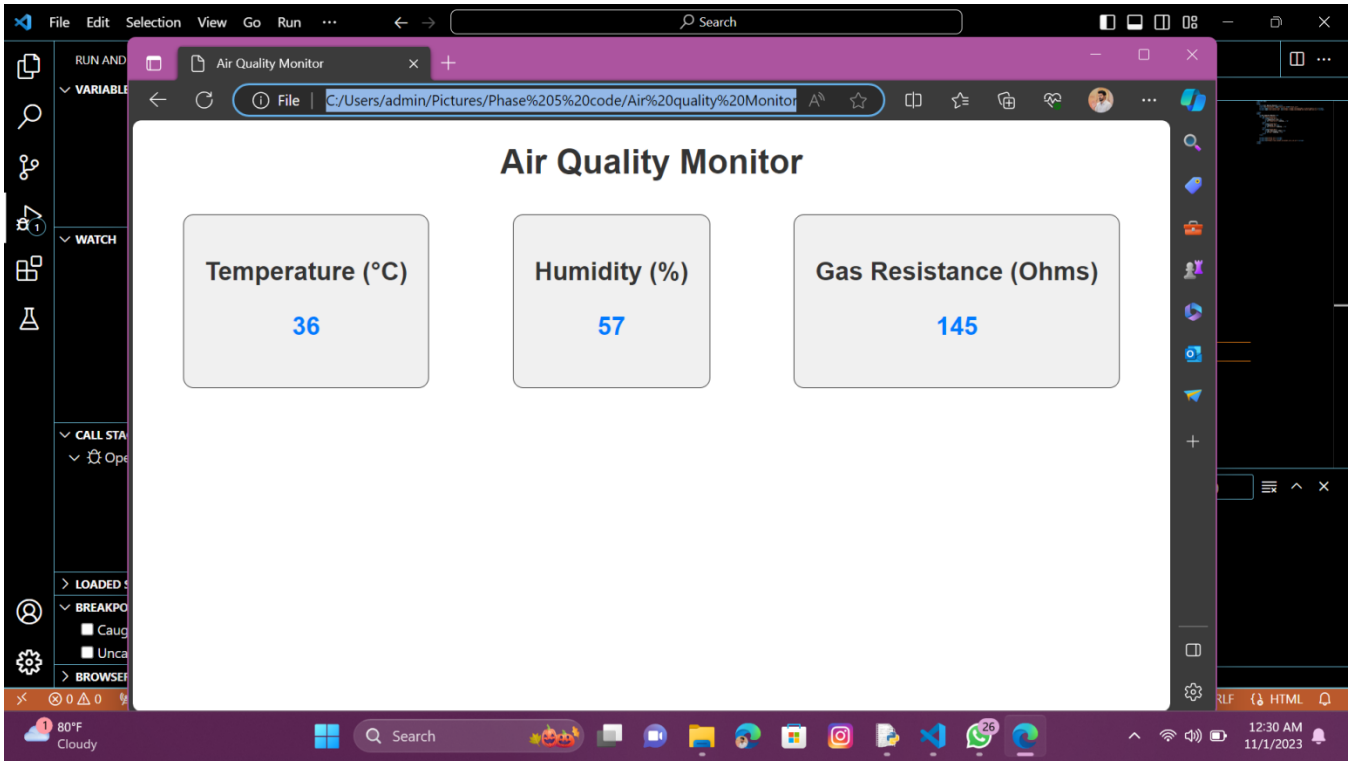
```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

h1 {
    color: #333;
}

#data-container {
    display: flex;
    justify-content: space-around;
}

.data-box {
    border: 1px solid #333;
    padding: 20px;
    border-radius: 10px;
    margin: 10px;
    background-color: #f0f0f0;
}

.data-box h2 {
    color: #333;
}

.data-box p {
    font-size: 24px;
    font-weight: bold;
    color: #007BFF;
}
```

3.JS:

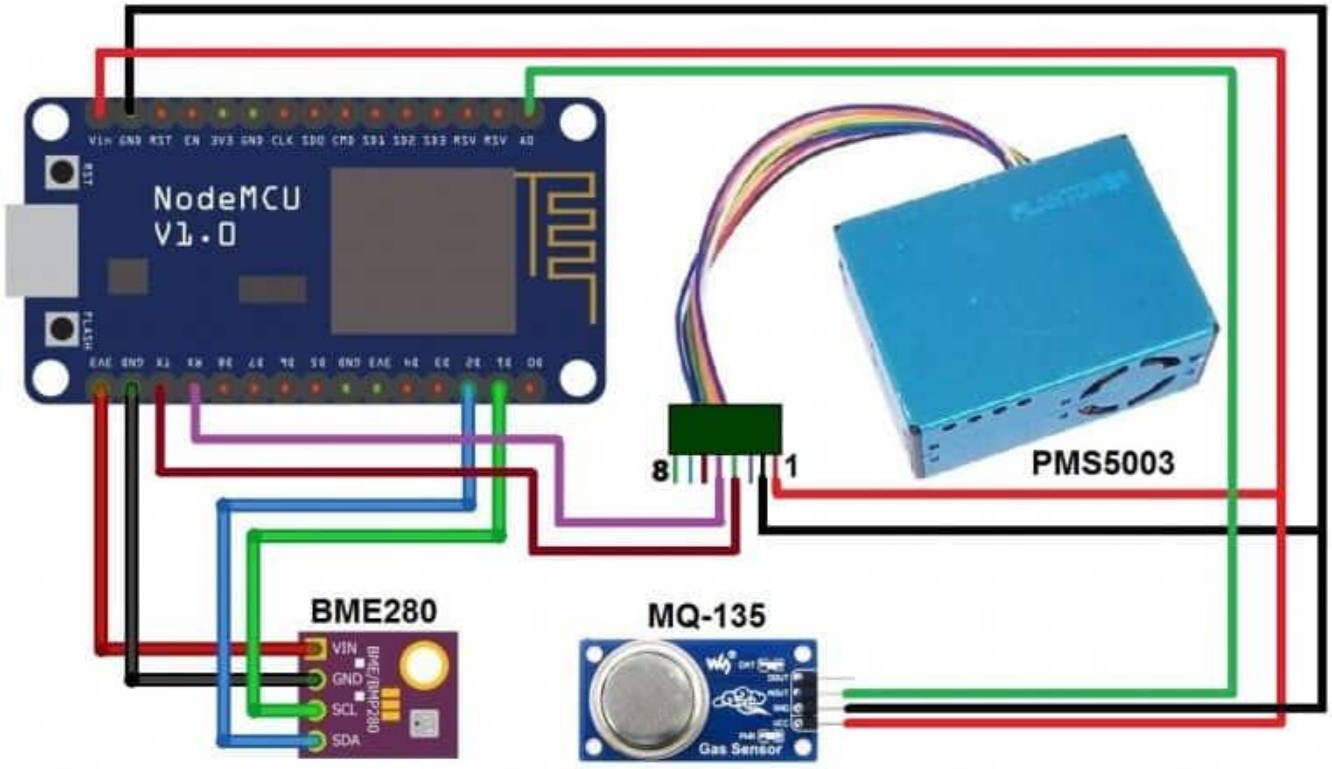
```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

h1 {
    color: #333;
}

#data-container {
    display: flex;
    justify-content: space-around;
}

.data-box {
    border: 1px solid #333;
    padding: 20px;
    border-radius: 10px;
    margin: 10px;
    background-color: #f0f0f0;
}

.data-box h2 {
    color: #333;
}

.data-box p {
    font-size: 24px;
    font-weight: bold;
    color: #007BFF;
}
```

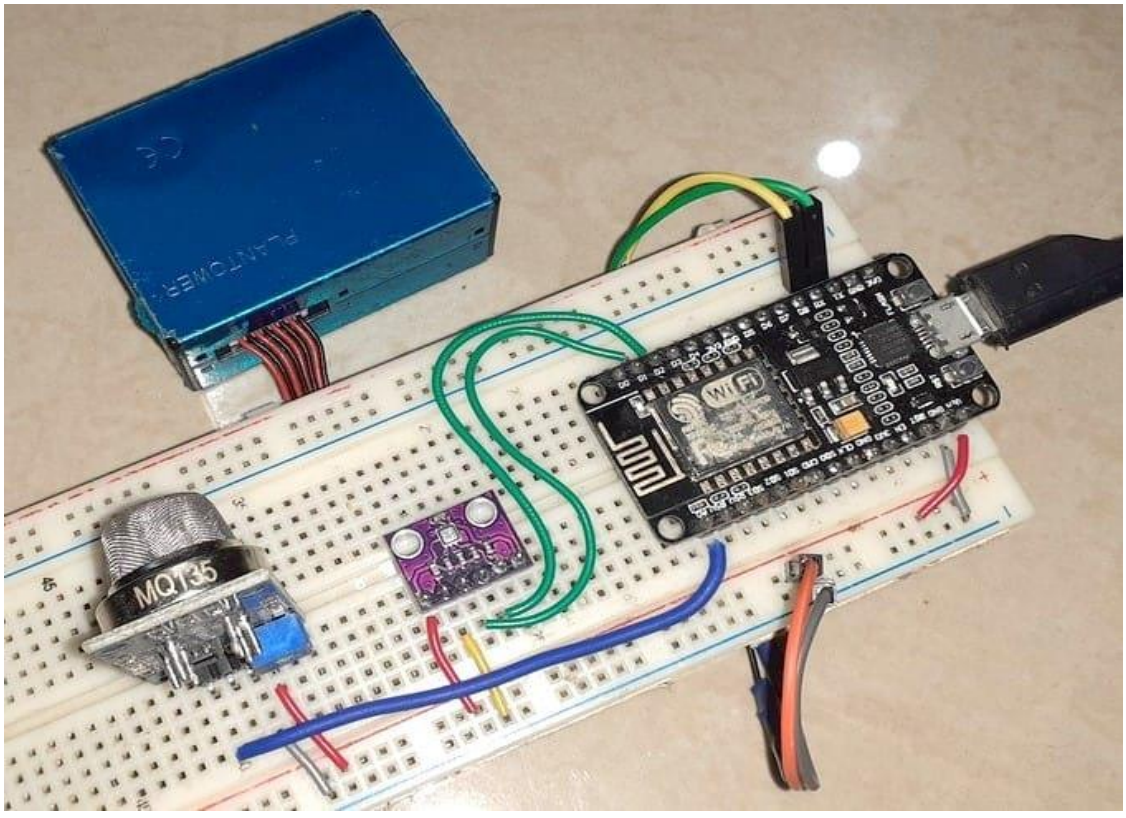
SCREENSHOT:


Air Quality Monitor
Air Quality Monitor
Temperature (°C)
24
Humidity (%)
55
Gas Resistance (Ohms)
227


Air Quality Monitor
Air Quality Monitor
Temperature (°C)
36
Humidity (%)
57
Gas Resistance (Ohms)
145

FINAL CIRCUIT DIAGRAM:





CONCLUSION:

In conclusion, air quality monitoring is a crucial component in safeguarding public health and the environment. Implementing an IoT-based air quality monitoring system offers several advantages, including real-time data collection, analysis, and prompt response to potential environmental hazards. By integrating various sensors, communication protocols, and data processing platforms, this system can provide comprehensive insights into pollutant levels, enabling informed decision-making and proactive measures to improve air quality.