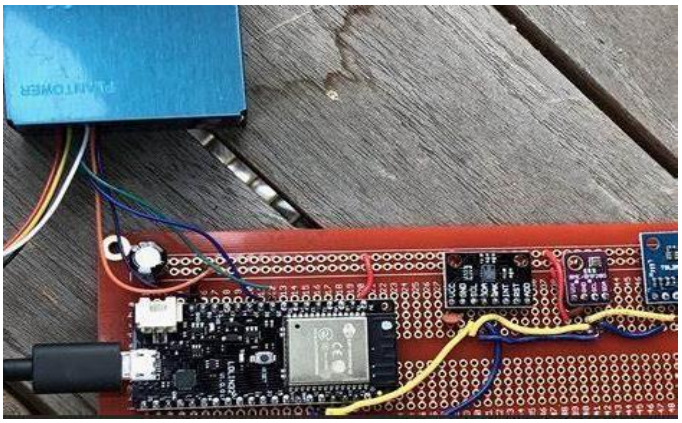
Air Quality Management

Phase:3-Development Part 1

Sensors we used for the detection of air quality:

For Particulate Matter (PM) sensors, two commonly used models are the Plantower PMS5003 and PMS7003. Both of these sensors are known for their reliability and accuracy in measuring fine particulate matter in the air. The below explains about the materials for measuring AQ.

Plantower PMS5003:



Features:

Measures PM1.0, PM2.5, and PM10 particulate concentrations.

High sensitivity and fast response time.

Suitable for indoor and outdoor air quality monitoring.

Usage:

Often used in air quality monitoring devices, weather stations, and environmental monitoring systems.

Plantower PMS7003:

Features:

Measures PM1.0, PM2.5, and PM10 particulate concentrations.

Compact and lightweight design.

UART and PWM communication options.

Usage:

Widely used in air quality monitoring applications, IoT devices, and environmental research

PM1.0, PM2.5, and PM10 refer to different size fractions of particulate matter in the air. These are commonly used terms in air quality monitoring to describe the sizes of particles being measured

PM1.0 (Particulate Matter 1.0):

Size Range: Particles with aerodynamic diameters less than or equal to 1.0 micrometers (µm).

Features:

Represents the smallest particles measured in air quality monitoring.

Includes fine and ultrafine particles, which can penetrate deeper into the respiratory system.

Usage:

Monitoring ultrafine particles that may have adverse health effects.

Used in research and studies focusing on the smallest particle sizes.

PM2.5 (Particulate Matter 2.5):

Size Range: Particles with aerodynamic diameters less than or equal to 2.5 micrometers (µm).

Features:

Represents fine particles that can be inhaled into the respiratory system.

Includes particles from various sources, such as combustion processes and natural sources.

Usage:

Widely used in air quality monitoring due to its relevance to human health.

Regulatory standards often specify limits for PM2.5 concentrations.

PM10 (Particulate Matter 10):

Size Range: Particles with aerodynamic diameters less than or equal to 10 micrometers (μm).

Features:

Represents larger particles, including coarse particles.

Includes particles from both human-made and natural sources.

Usage:

Used in air quality monitoring to assess the overall level of inhalable particles.

Regulatory standards often specify limits for PM10 concentrations.

These are the different sizes of particles present in the air

Now, next to this we are going to place these sensors in outdoor for better identification of the air quality:

Outdoor Placement:

Street-Level Monitoring:

Placing PM sensors at street level is common in urban areas to monitor traffic-related pollution. This can help assess the impact of vehicular emissions on air quality.

SOFTWARE COMPONENTS:

Air quality monitoring using IoT typically involves a combination of software components for data collection, processing, visualization, and communication. Here are the key software components used in an air quality monitoring system**:**

Sensor Drivers and Libraries:

These software components are responsible for interfacing with the air quality sensors. They provide the necessary drivers and libraries to communicate with the sensors and retrieve data.

IoT Middleware or Cloud Platform:

IoT Platforms: Platforms like AWS IoT, Google Cloud IoT, Microsoft Azure IoT, and IBM Watson IoT provide scalable, cloud-based solutions for data management, analytics, and device management.

IoT Edge Computing: Some data processing and analysis can be done at the edge (on the device) to reduce latency and minimize data transmission.

Alerting and Notification Systems:

Software that can trigger alerts and notifications based on predefined thresholds or anomaly detection. Notifications can be sent via email, SMS, or push notifications to mobile apps
.

Database Management Systems (DBMS):

Used for storing historical air quality data. Popular options include SQL databases (e.g., PostgreSQL, MySQL) or NoSQL databases (e.g., MongoDB).

Components:

1. Arduino board (e.g., Arduino Uno or Arduino Nano)
2. Air quality sensor (e.g., MQ series sensor)
3. Wi-Fi module (e.g., ESP8266 or ESP32) for IoT connectivity
4. A computer or Raspberry Pi for running the Python program
5. Blynk or a similar IoT platform for data visualization (optional)

Steps:

1. Hardware Setup:

>Connect the air quality sensor to the Arduino following the sensor's datasheet and your chosen Arduino model.
> Connect the Wi-Fi module to the Arduino, ensuring it can connect to your Wi-Fi network.

2. Arduino Code:

Write an Arduino sketch to read data from the air quality sensor and send it to the IoT platform (e.g., Blynk). Make sure to include the necessary libraries for your sensor and Wi-Fi module.
Here's a simplified example using the Adafruit CCS811 air quality sensor and an ESP8266 Wi-Fi module:

```
#include &lt;Wire.h&gt;
#include &lt;Adafruit_CCS811.h&gt;
#include &lt;ESP8266WiFi.h&gt;
#include &lt;BlynkSimpleEsp8266.h&gt;
char auth[] = &quot;Your_Blynk_Auth_Token&quot;;
char ssid[] = &quot;Your_WiFi_SSID&quot;;
char pass[] = &quot;Your_WiFi_Password&quot;;
Adafruit_CCS811 ccs;
void setup() {
Serial.begin(115200);
Blynk.begin(auth, ssid, pass);
ccs.begin();
}
void loop() {
if (!ccs.available()) {
return;
}
if (!ccs.readData()) {
```

```
return;
}
float temperature = ccs.calculateTemperature();
float humidity = ccs.calculateHumidity();
int eCO2 = ccs.geteCO2();
int TVOC = ccs.getTVOC();
Serial.print(&quot;eCO2: &quot;);
Serial.println(eCO2);
Serial.print(&quot;TVOC: &quot;);
Serial.println(TVOC);
Blynk.virtualWrite(V1, eCO2); // Send eCO2 data to virtual pin V1
Blynk.virtualWrite(V2, TVOC); // Send TVOC data to virtual pin V2
Blynk.run();
delay(1000); // Delay between readings
```

Code that integrates sensor to the environment:

```
import time

import requests

import random

# Data-sharing platform configuration (replace with your platform's details)

api_url = "https://platform.openai.com/account/api-keys"

api_key = "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"


def generate_simulated_data():

    # Simulate PM2.5 and PM10 values

    pm25 = random.uniform(5, 20)

    pm10 = random.uniform(10, 30)

    return pm25, pm10


def send_data_to_platform(pm25, pm10):

    payload = {"api_key": api_key, "field1": pm25, "field2": pm10}

    try:

        response = requests.post(api_url, data=payload)

        print("Simulated data sent successfully")

    except Exception as e:

        print(f"Error sending simulated data: {e}")


if __name__== "_main_":

    try:
```

```
    while True:
        pm25, pm10 = generate_simulated_data()
        print(f"Simulated PM2.5: {pm25} µg/m³, PM10: {pm10} µg/m³")
        send_data_to_platform(pm25, pm10)
        time.sleep(300) # Adjust this based on your data-sharing platform's limitations
except KeyboardInterrupt:
        print("Script terminated by user.")
```

Output:

Simulated PM2.5: 15.018356831988099 µg/m³, PM10: 20.09326754324212 µg/m³
Simulated data sent successfully
Simulated PM2.5: 11.843433202538392 µg/m³, PM10: 25.414319872755364 µg/m³
Simulated data sent successfully
Simulated PM2.5: 13.656690136913404 µg/m³, PM10: 22.554439980378543 µg/m³
Simulated data sent successfully
Simulated PM2.5: 15.180759281165448 µg/m³, PM10: 25.21239236030451 µg/m³
Simulated data sent successfully
Simulated PM2.5: 8.562569641229786 µg/m³, PM10: 25.159651799641956 µg/m³
Simulated data sent successfully
Simulated PM2.5: 19.934493649751055 µg/m³, PM10: 24.840927601852826 µg/m³