**Project Documentation**

**Objectives:**

The Air Quality Monitoring project aims to create a real-time air quality monitoring system using IoT devices. The objectives include measuring various air pollutants, developing a user-friendly data-sharing platform, and raising public awareness about air quality and its health impacts.

**IoT Device Setup:**

We utilized Arduino-based sensors to measure pollutants like particulate matter (PM2.5 and PM10), carbon dioxide (CO2), and volatile organic compounds (VOCs). These sensors were connected to an ESP8266 Wi-Fi module for data transmission.

**Platform Development:**

The data from IoT devices were sent to a cloud-based server. We used AWS IoT Core for secure data transmission. The backend was developed using Python and Flask, while the frontend was created using HTML, CSS, and JavaScript. The platform displayed real-time air quality data in a visually appealing manner.

**Code Implementation:**

We implemented the IoT device code in Arduino IDE using appropriate libraries for sensor data reading and Wi-Fi communication. The backend server was coded in Python using Flask, ensuring smooth data processing and storage.

**Diagrams, Schematics, and Screenshots:**

A diagram of a cloud

Description automatically generated

A diagram of a circuit board

Description automatically generated

**Raising Public Awareness:**

The real-time air quality monitoring system can raise public awareness by providing easily understandable data. The platform shows pollution levels in user-friendly graphs and charts, making it accessible to people of all backgrounds. By visualizing the air quality in their area, individuals can take necessary precautions and demand cleaner air policies.

**Submission**

**GitHub Repository:**

**https://github.com/Madhumitha2654/9536\_Ramco-Institute-Of-Technology\_Air-Quality-Monitoring**

**Replication Instructions:**

1. **Setting Up IoT Devices:**
   * Connect the sensors (PM2.5, PM10, CO2, VOCs) to the appropriate pins on the Arduino board.
   * Upload the Arduino code provided in the repository to the board.
   * Connect the ESP8266 module to the Arduino for wireless data transmission.
2. **Developing the Data-sharing Platform:**
   * Set up an AWS IoT Core account.
   * Create IoT Thing, Policy, and Certificate for secure communication.
   * Deploy the backend Python code on a server (AWS EC2 instance recommended).
   * Host the frontend files (HTML, CSS, JavaScript) on a web server (AWS S3 bucket recommended).
   * Configure CORS settings for the S3 bucket to allow communication between frontend and backend.
3. **Integration Using Python:**
   * Modify the backend code to handle incoming sensor data.
   * Implement necessary data processing and storage functions.
   * Ensure the frontend can fetch data from the backend APIs.

**Example Outputs:**

A circuit board with wires and a display

Description automatically generated