IoT-Phase 3

Topic: Air quality monitoring

Team Members: Pradeep K, Raghuram P, Hemachandhru N, Abilash R

Mentor: Sundaraj V

Configuring IoT devices to measure air quality parameters involves several steps, including selecting appropriate sensors, connecting them to a microcontroller or IoT platform, and developing the necessary code to collect and transmit the data. Below is a step-by-step guide to help you set up IoT devices to measure air quality parameters:

Step 1: Selecting Sensors

1. Air Quality Sensors:

- Gas Sensors: Choose sensors that can detect common pollutants like CO2, CO, NO2, etc. Sensors from manufacturers like Bosch, Figaro, and AMS are popular choices.

- Particulate Matter Sensors: Opt for sensors capable of measuring PM1, PM2.5, and PM10 particle concentrations.

2. Microcontroller/Platform:

- Consider using platforms like Raspberry Pi, Arduino, ESP8266, or ESP32. They provide GPIO pins to connect sensors and have Wi-Fi capabilities for data transmission.

Step 2: Connect Sensors

1. Wiring:

- Connect the sensors to the microcontroller using appropriate cables and connectors. Refer to datasheets and documentation for specific wiring instructions.

2. Power Supply:

- Ensure a stable power supply for both the microcontroller and sensors. Consider using a reliable power source, battery, or power bank.

Step 3: Writing Code

1. Set Up Development Environment:

- Install the necessary IDE and libraries for chosen microcontroller. For example, use Arduino IDE for Arduino boards, or set up Python for Raspberry Pi.

2. Write Code:

- Write code to initialize the sensors, read data, and format it for transmission. Here's an example for a hypothetical air quality sensor and an ESP8266 (using Arduino IDE):

```
cpp
#include <Wire.h>
#include <AirQualityLibrary.h> // Replace with actual library

AirQualitySensor airSensor; // Initialize air quality sensor

void setup() {
```

```
  Serial.begin(9600);
  airSensor.begin(); // Initialize air sensor
}

void loop() {
  float pollutionLevel = airSensor.getPollutionLevel(); // Get pollution level
  Serial.println("Pollution Level: " + String(pollutionLevel));
  delay(10000); // Delay for data transmission
}
```

Step 4: Data Transmission

1. Choose Data Transmission Protocol:

- Use MQTT, HTTP, or other suitable protocols to send data to data-sharing platform.

2. Configure Network Credentials:

- Set up Wi-Fi or network credentials on microcontroller to enable internet connectivity.

Step 5: Data Handling

1. Data Processing:

- If necessary, perform data processing (e.g., filtering, averaging) before transmission to ensure accurate readings.

2. Data Aggregation:

- Consider aggregating data over time intervals to reduce transmission overhead and data storage requirements.

Step 6: Set Up Data-Sharing Platform

1. Choose a Data Platform:

- Select a platform to receive and visualize the data. This could be a cloud-based service, custom server, or any platform of choice.

2. Implement Platform-Specific Code:

- Write code to handle incoming data on the platform side (e.g., MQTT subscriber, HTTP server, etc.).

3. Display and Analyze Data:

- Create dashboards or applications to display and analyze the air quality data.

Step 7: Testing and Deployment

1. Testing:

- Test the entire setup to ensure the sensors are providing accurate data and that the IoT device is transmitting it correctly.

2. Deployment:

- Deploy the IoT devices in the desired locations for air quality monitoring.

PYTHON SCRIPT

```
pip install paho-mqtt

import paho.mqtt.client as mqtt
import json
import time
import random

# Configuration for  MQTT broker
mqtt_broker_address = "mqtt.example.com"
mqtt_port = 1883
mqtt_topic = "_topic"
mqtt_username = "_username"
mqtt_password = "_password"

# Function to simulate collecting data (replace with  actual data collection logic)
def collect_data():
    # Simulate collecting data
    data = {
        "temperature": random.uniform(20, 30),
        "humidity": random.uniform(40, 60)
    }
    return data

# Callback when the client connects to the MQTT broker
```

```
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
    else:
        print(f"Connection failed with code {rc}")

# Main function to publish data to MQTT broker
def publish_data():
    client = mqtt.Client()
    client.username_pw_set(mqtt_username, mqtt_password)
    client.on_connect = on_connect

    try:
        client.connect(mqtt_broker_address, mqtt_port, 60)
    except Exception as e:
        print(f"Error connecting to MQTT broker: {str(e)}")
        return

    while True:
        data = collect_data()
        payload = json.dumps(data)

        # Publish data to the MQTT topic
        client.publish(mqtt_topic, payload)
        print(f"Published data: {payload}")
```

```
        time.sleep(5)  # Adjust the interval as needed

if __name__ == "__main__":
    publish_data()
```