AIR QUALITY MEASUREMENT

TEAM MEMBERS

963321104302 : K. RAGUL

963321104301: G. ARAVIND

963321104501: S. ALDRIN SAMUEL

Phase 2 : Development Part 1

Developing WOKWI simulation for Air Quality Monitoring

Configuring IoT devices to measure air quality parameters for simulation on platforms like Wokwi is a great way to develop and test your IoT project without the need for physical hardware. In this example, I'll guide you through configuring a virtual environment using Wokwi and simulating the data from a simulated air quality sensor.

1. Choose Simulated Hardware:

- Go to the Wokwi website (https://wokwi.com) and create an account if you haven't already.
- On the Wokwi platform, select the hardware components you want to use. You can choose a microcontroller, such as Arduino, and other components like sensors.

2. Configure the Microcontroller:

- Drag and drop an Arduino microcontroller onto the Wokwi workspace.
- You can choose an Arduino board that fits your requirements. For example, you might use an Arduino Uno for simplicity.

3. Add Sensors:

- Drag and drop simulated sensors onto the workspace. Wokwi provides a variety of sensors for simulation purposes. For air quality monitoring, you might want to use a simulated PM2.5/PM10 sensor and, optionally, a temperature and humidity sensor.
- Connect the sensors to the appropriate pins on the Arduino using virtual wires.

4. Write and Upload Arduino Sketch:

- Write an Arduino sketch to read data from the sensors and send it to a virtual IoT platform or serial monitor. For example, you can use the Wokwi IoT platform, which is a virtual platform that allows you to send and receive data.
- Here is a basic example of an Arduino sketch for simulating air quality and sending data to the Wokwi IoT platform:

```
#include <WokwiIoT.h>
#include <WokwiSerialBridge.h>

WokwiIoT iot;
WokwiSerialBridge serialBridge;

void setup() {
  iot.begin();
  iot.connect("YourDeviceName", "YourDeviceToken");
  iot.subscribeTo("YourTopic");
  Serial.begin(9600);
}

void loop() {
  // Read data from the simulated air quality sensor and the temperature/humidity sensor.
  float pm25Value = readPM25Sensor();
  float temperatureValue = readTemperatureSensor();
  float humidityValue = readHumiditySensor();

  // Create a JSON message with the sensor data
  String jsonMessage = "{\"pm25\": " + String(pm25Value) +
                       ", \"temperature\": " + String(temperatureValue) +
```

```
                ", \"humidity\": " + String(humidityValue) + "}";

  // Publish the data to the Wokwi IoT platform
  iot.publish("YourTopic", jsonMessage);

  // Send data to the serial monitor for debugging (optional)
  Serial.println(jsonMessage);

  delay(10000); // Adjust the delay as needed
}

float readPM25Sensor() {
  // Simulate PM2.5/PM10 sensor data
  return random(10, 50); // Replace with actual sensor data
}

float readTemperatureSensor() {
  // Simulate temperature sensor data
  return random(20, 30); // Replace with actual sensor data
}

float readHumiditySensor() {
  // Simulate humidity sensor data
  return random(30, 60); // Replace with actual sensor data
}
```

5. Simulate and Monitor:

- Click the "Run" button in Wokwi to simulate your project.

Developing Python Script for Collecting data from IoT devices

To develop a Python script on an IoT device to send collected data to a data-sharing platform, you would need to choose a specific data-sharing platform and the corresponding protocol for data transmission. In this example, I'll provide a generic Python script that uses the MQTT protocol to send data to an MQTT broker. You can modify this script according to your chosen data-sharing platform's requirements.

1. Install the Required Library:

Make sure you have the **paho-mqtt** library installed on your IoT device. You can install it using pip:

```
pip install paho-mqtt
```

2. Write the Python Script:

Here's a Python script that collects sample data (e.g., air quality parameters) and sends it to an MQTT broker. Modify the script to suit your specific sensors and MQTT broker settings.

```
import paho.mqtt.client as mqtt

import time

import json

import random


# MQTT broker settings

mqtt_broker = "mqtt.example.com"  # Replace with your MQTT broker address

mqtt_port = 1883  # Default MQTT port

mqtt_topic = "air_quality"  # MQTT topic where data will be published

mqtt_username = "your-username"  # Replace with your MQTT username

mqtt_password = "your-password"  # Replace with your MQTT password
```

```
# Create an MQTT client
client = mqtt.Client("AirQualityDevice")

# Set username and password (if required)
client.username_pw_set(username=mqtt_username, password=mqtt_password)

# Define an on_connect callback function
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT broker")
    else:
        print("Connection failed with error code " + str(rc))

# Define an on_publish callback function
def on_publish(client, userdata, mid):
    print("Message Published")

# Set callback functions
client.on_connect = on_connect
client.on_publish = on_publish

# Connect to the MQTT broker
client.connect(mqtt_broker, mqtt_port)

# Start the MQTT loop
```

```
client.loop_start()

while True:
    # Simulate collecting air quality data (replace with your actual sensor readings)
    air_quality_data = {
        "PM2.5": random.uniform(0, 100),
        "CO2": random.uniform(300, 2000),
        "Temperature": random.uniform(20, 30),
        "Humidity": random.uniform(30, 70)
    }

    # Convert data to JSON format
    payload = json.dumps(air_quality_data)

    # Publish the data to the MQTT broker
    client.publish(mqtt_topic, payload)

    # Print the sent data (for testing)
    print("Published data: " + payload)

    # Adjust the sleep interval (e.g., 300 seconds) based on your requirements
    time.sleep(300)

# This loop will keep running and periodically send data to the MQTT broker.
```