- ***IOT_PHASE3***

Air Quality Monitoring

Topic : Air Quality Monitoring

Name: ABHISHEK A STEPHEN

Register no: 950321104002

Building project:

Air quality monitoring using an ESP32 sensor, including a sensor diagram and Python script, is beyond the scope of a short text-based response. However, I can provide you with a step-by-step overview, a simplified wiring diagram, and a sample Python script to get you started.

1. Components Needed:

- ESP32 development board
- SDS011 air quality sensor
- Breadboard and jumper wires
- Micro USB cable for power
- A computer with the Arduino IDE or VS Code PlatformIO with the PlatformIO extension

2. Wiring Diagram:

- Connect the SDS011 sensor to the ESP32 using UART communication. The SDS011 has three pins: VCC, TX, and RX.
- Connect the VCC pin of the SDS011 to a 5V output on the ESP32.
- Connect the TX pin of the SDS011 to one of the ESP32's available RX pins (e.g., GPIO16).
- Connect the RX pin of the SDS011 to one of the ESP32's available TX pins (e.g., GPIO17).
- Connect the GND pin of the SDS011 to a ground pin on the ESP32.

3. Programming the ESP32:

- Install the Arduino IDE or PlatformIO with the ESP32 board support.
- Write a MicroPython script to read data from the SDS011 sensor and transmit it over Wi-Fi. Below is a simplified Python script:

```
Import time
From machine import UART
Import network
Import urequests as requests
```

```
# Connect to your Wi-Fi network
Ssid = “YourWiFiNetwork”
Password = “YourWiFiPassword”
Sta = network.WLAN(network.STA_IF)
Sta.active(True)
Sta.connect(ssid, password)
# Define UART pins
Uart = UART(2, baudrate=9600, tx=16, rx=17) # Modify the pins as per your ESP32 connections
# Server URL for data transmission
Server_url = http://yourserver.com/api/air_quality
While True:
While not sta.isconnected():
Pass
Data = uart.read(10) # Read 10 bytes of data
If data is not None and len(data) == 10:
Pm25 = (data[2] + (data[3] << 8)) / 10.0
Pm10 = (data[4] + (data[5] << 8)) / 10.0
Payload = {‘pm25’: pm25, ‘pm10’: pm10}
Response = requests.post(server_url, json=payload)
If response.status_code == 200:
Print(“Data sent successfully”)
Else:
Print(“Failed to send data”)
Time.sleep(60) # Adjust the interval as needed
```

---

4. **Server-Side Handling**:

On your server, create an API endpoint to receive and process the data from the ESP32. Store the data in a database or perform further actions.