AIR QUALITY MONITORING

PHASE 4

Introduction:

In this project, we are developing an air quality monitoring and reporting system using Arduino and air quality sensors. After quality weather data, the completed data is sent to the ThingSpeak platform for real-time monitoring and analysis. The system has two main components: an Arduino-based sensor node and an ESP8266-based WiFi module for data transmission.

WiFi Module (ESP8266):

To add it to the circuit, ESP8266 sends data to ThingSpeak, you need to set the WiFi certificate and install the necessary libraries. Below is the code used to connect the ESP8266 to a WiFi network and send data to ThingSpeak. Change settings with your real WiFi certificate and ThingSpeak API key.

Python Code :

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";
const char* apiKey = "YourThingSpeakAPIKey";
const char* server = "api.thingspeak.com";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  sendDataToThingSpeak();
}

void sendDataToThingSpeak() {
  HTTPClient http;
  String url = "/update?api_key=" + String(apiKey) + "&field1=" + String(airQualityValue);

  http.begin(server, 80, url);
  int httpCode = http.GET();

  if (httpCode > 0) {
    String payload = http.getString();
    Serial.println(payload);
  }

  http.end();
}

void loop() {
  // Your sensor reading logic goes here
  int airQualityValue = getAirQualityValue();
  sendDataToThingSpeak(airQualityValue);
  delay(60000);  // Send data every minute
}

int getAirQualityValue() {
  // Implement your logic to read the air quality value from the sensor
  // This will depend on your specific sensor and calibration
  return airQualityValue;
}
```

Micro Python:

```
import machine
import network
import urequests as requests
import time
import random

# WiFi credentials and ThingSpeak API key
wifi_ssid = "YourWiFiSSID"
wifi_password = "YourWiFiPassword"
thingspeak_api_key = "YourThingSpeakAPIKey"
```

```
# ThingSpeak server
server = "api.thingspeak.com"

# Initialize the WiFi connection
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

while not wifi.isconnected():
    pass

print("Connected to WiFi")

def sendDataToThingSpeak(airQualityValue):
    url = "/update?api_key=" + thingspeak_api_key + "&field1=" + str(airQualityValue)
    response = requests.get("https://" + server + url)

    if response.status_code == 200:
        print("Data sent to ThingSpeak")
    else:
        print("Failed to send data to ThingSpeak")

def getAirQualityValue():
    # Implement your logic to read air quality from the sensor here
    # This will depend on your specific sensor and calibration
    return random.randint(0, 500)  # Simulated data; replace with actual sensor data

while True:
    airQualityValue = getAirQualityValue()
    print("Air Quality (PPM):", airQualityValue)

    sendDataToThingSpeak(airQualityValue)

    time.sleep(60)  # Send data every minute
```

It contains the necessary libraries for WiFi and HTTP communication.

You set the WiFi network certificate and ThingSpeak API key.

In the setup function:
Initiates serial communication for debugging.
Connects to your WiFi network and keeps trying until it succeeds.
Calls sendDataToThingSpeak() after connecting.
sendDataToThingSpeak() Function:

Create an HTTP client and create a URL to send data to ThingSpeak.
Create an HTTP connection to ThingSpeak.
Send an HTTP GET request with accurate weather information.
Print the server's response to the serial monitor.
Close the HTTP connection.
In the Loop() function:

In the getAirQualityValue() function you must use sensor reading logic to retrieve air quality data.
Call sendDataToThingSpeak() to send data to ThingSpeak.
Periodically increase the data sending interval (every minute in this example)