IOT PHASE_3

Development Part of IOT –Air Quality Monitoring

Introduction:

Building an IoT-enabled Smart Water Fountains system involves several steps. In this example, I'll outline how you can deploy IoT sensors in public water fountains to monitor water flow and detect malfunctions, and then develop a Python script to send real-time fountain status data to a platform. Keep in mind that this is a simplified example, and a realworld deployment may require more robust solutions.

Hardware Requirements:

1. ESP32 Board:
    - The ESP32 is a low-cost, low-power microcontroller with integrated Wi-Fi and Bluetooth capabilities, making it suitable for IoT projects.
    - You can choose a specific ESP32 development board based on your preferences and availability.
2. Air Quality Sensor:
    - Select a suitable air quality sensor. Common options include the MQ series (e.g., MQ-135) or sensors like the SDS011 for particulate matter measurement.
    - Make sure the sensor provides the necessary data for your application (e.g., CO2 levels, PM2.5, PM10).
3. Power Supply:
    - ESP32 boards typically require a 5V power supply. Ensure that the power supply can provide sufficient current for both the ESP32 and the sensor.
4. Cables and Connectors:
    - Depending on the chosen sensor, you may need jumper wires and connectors to interface it with the ESP32.
5. Enclosure (Optional):

- If the monitoring system will be deployed outdoors, consider an enclosure to protect the electronics from environmental conditions.

Software Requirements:

1. Arduino IDE:
    - Download and install the Arduino IDE from the official website (https://www.arduino.cc/en/Main/Software).
2. ESP32 Board Support:
    - Add ESP32 board support to the Arduino IDE by following the instructions provided by the ESP32 Arduino Core (https://github.com/espressif/arduino-esp32).
3. Sensor Library:
    - Install the necessary libraries for your chosen air quality sensor. You can usually find these libraries in the Arduino Library Manager or on GitHub.
4. IoT Platform:
    - Choose an IoT platform to store and visualize the air quality data. Popular options include:
        - Blynk: A platform for building IoT applications with a drag-and-drop interface.
        - ThingSpeak: A platform for collecting, analyzing, and visualizing data.
        - IoT platforms by major cloud providers (e.g., AWS IoT, Google Cloud IoT, Microsoft Azure IoT).

Software Implementation:

1. Read Sensor Data:
    - Write a program in the Arduino IDE to read data from the air quality sensor using the provided library.

2. Connect to Wi-Fi:
    - Configure the ESP32 to connect to your Wi-Fi network using the built-in Wi-Fi library.
3. Send Data to IoT Platform:
    - Implement code to send the air quality data to your chosen IoT platform. This involves using the platform's API or library.
4. Optional: Data Processing and Visualization:
    - If needed, add code to process and visualize the data. This could involve smoothing algorithms, data averaging, or creating charts and graphs.
5. Power Management (Optional):
    - Implement power-saving features if you want to optimize energy consumption, especially if the monitoring system is battery-powered.
6. Error Handling:
    - Include error-handling mechanisms to ensure the robustness of your system, such as handling Wi-Fi disconnections or sensor malfunctions.

Testing and Deployment:

1. Testing:
    - Test your air quality monitoring system in a controlled environment to ensure it's working correctly.
2. Deployment:
    - Deploy the system in the target environment, considering factors like power supply and connectivity.
3. Monitoring and Maintenance:
    - Regularly monitor the system and implement any necessary maintenance, such as replacing sensors or updating software.

C program:

```
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>

// Wi-Fi credentials
char ssid[] = "your_wifi_ssid";
char pass[] = "your_wifi_password";

// Blynk credentials
char auth[] = "your_blynk_auth_token";

// Pin configuration
#define MQ_PIN A0  // Analog pin for MQ-135 sensor
#define LED_PIN 2  // Digital pin for status LED

// Blynk virtual pins
#define BLYNK_PIN_AIR_QUALITY V0

void setup() {
Serial.begin(115200);
pinMode(LED_PIN, OUTPUT);

 // Connect to Wi-Fi
 WiFi.begin(ssid, pass);
 while (WiFi.status() != WL_CONNECTED) {
 delay(1000);
 Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to WiFi");

 // Initialize Blynk
 Blynk.begin(auth, ssid, pass);

 // Other setup code
}
```

```
 void loop() {
 Blynk.run();

 // Read air quality data from sensor
 int airQuality = analogRead(MQ_PIN);

 // Map the sensor values to a range (adjust as needed)
 int mappedValue = map(airQuality, 0, 1023, 0, 100);

 // Send air quality data to Blynk
 Blynk.virtualWrite(BLYNK_PIN_AIR_QUALITY, mappedValue);

// Print to Serial Monitor for debugging
 Serial.print("Air Quality: ");
 Serial.println(mappedValue);
// Add a delay before the next reading
 delay(5000); // 5 seconds
 }

 void loop() {
Blynk.run();

// Read air quality data from sensor
int airQuality = analogRead(MQ_PIN);

// Map the sensor values to a range (adjust as needed)
int mappedValue = map(airQuality, 0, 1023, 0, 100);
// Send air quality data to Blynk
Blynk.virtualWrite(BLYNK_PIN_AIR_QUALITY, mappedValue);

// Print to Serial Monitor for debugging
Serial.print("Air Quality: ");
Serial.println(mappedValue);
// Add a delay before the next reading
delay(5000);  // 5 seconds

}
```

SUBMITTED

BY

MENTOR

S ABIKAYAL AARTHI

AP/CSE

TEAM MEMBERS

ROOHI SHIFA M(au821121104047)

ROOBIGA R(au821121104046)

SATHYA A(au821121104049)

SHARMIKA R(au821121104051)

MURUGESHWARI A(au821121104037)

AARTHI S(au821121104003)