Project Title: IoT Air Quality Monitoring System

Description: This project aims to create an IoT-based system that monitors air quality in real-time and provides users with data and alerts about the air quality in their surroundings.

Components Required:

Air Quality Sensors: You'll need sensors like PM2.5 and PM10 particulate matter sensors, carbon monoxide (CO) sensors, nitrogen dioxide (NO2) sensors, and ozone (O3) sensors.

Microcontroller: Use a microcontroller like Arduino, Raspberry Pi, or ESP8266 to collect data from the sensors and process it.

Internet Connectivity: Ensure your microcontroller can connect to the internet, either through Wi-Fi, Ethernet, or cellular connectivity.

Data Storage: Set up a database or cloud service to store the collected data.

User Interface: Create a user interface, which can be a web application or a mobile app, to display air quality data and alerts.

Steps to Build:

Connect the air quality sensors to the microcontroller and write code to read data from these sensors.

Configure the microcontroller to send this data to your chosen data storage solution, either periodically or in real-time.

Create a user interface that fetches data from the storage solution and displays it to the user. You can use charts and graphs to visualize air quality trends.

Implement alerting mechanisms. For example, send notifications to users when air quality falls below a certain threshold.

Optional Enhancements:

Geolocation: Incorporate GPS to track the location of the monitoring device, allowing users to monitor air quality at specific locations.

Historical Data: Store historical air quality data and provide users with the ability to view trends over time.

Air Purifier Integration: Integrate smart air purifiers that can be controlled based on air quality data.

Machine Learning: Use machine learning to predict air quality trends based on historical data.

Benefits:

Helps users make informed decisions about outdoor activities and health.

Raises awareness about air pollution.

Can contribute to research on air quality in different areas.

Remember to consider power management, security, and scalability while building your IoT AQM system. This project can be scaled up for community-level monitoring or used for personal air quality tracking.

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_BME680.h>

#include <WiFi.h>

#include <PubSubClient.h>

// Replace with your Wi-Fi credentials

const char\* ssid = "your\_SSID";

const char\* password = "your\_PASSWORD";

// Replace with your MQTT broker information

const char\* mqtt\_server = "your\_MQTT\_broker\_IP";

const int mqtt\_port = 1883;

const char\* mqtt\_username = "your\_MQTT\_username";

const char\* mqtt\_password = "your\_MQTT\_password";

Adafruit\_BME680 bme; // Initialize BME680 sensor

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

Serial.begin(115200);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

// Initialize BME680 sensor

if (!bme.begin(0x76)) {

Serial.println("Could not find a valid BME680 sensor, check wiring!");

while (1);

}

// Connect to MQTT broker

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

reconnect();

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

// Read sensor data

float temperature = bme.readTemperature();

float humidity = bme.readHumidity();

float pressure = bme.readPressure() / 100.0F; // Convert pressure to hPa

float gasResistance = bme.readGas();

// Publish data to MQTT topics

client.publish("aqm/temperature", String(temperature).c\_str());

client.publish("aqm/humidity", String(humidity).c\_str());

client.publish("aqm/pressure", String(pressure).c\_str());

client.publish("aqm/gas\_resistance", String(gasResistance).c\_str());

delay(10000); // Publish data every 10 seconds (adjust as needed)

}

void reconnect() {

while (!client.connected()) {

Serial.println("Connecting to MQTT...");

if (client.connect("AQM\_Client", mqtt\_username, mqtt\_password)) {

Serial.println("Connected to MQTT broker");

client.subscribe("aqm/control");

} else {

Serial.println("Failed to connect to MQTT broker. Retry in 5 seconds...");

delay(5000);

}

}

}

void callback(char\* topic, byte\* payload, unsigned int length) {

// Handle MQTT subscription messages if needed

}

Make sure to replace "your\_SSID", "your\_PASSWORD", "your\_MQTT\_broker\_IP", "your\_MQTT\_username", and "your\_MQTT\_password" with your specific Wi-Fi and MQTT broker details.