```
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <Arduino.h>

// Provide the token generation process info.
#include "addons/TokenHelper.h"
// Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// WiFi credentials
const char* ssid = "N Venkatesh";
const char* password = "5005mother";

// Firebase project credentials
#define API_KEY "AIzaSyA6b0InBLMBWSDWUWnB8adwLjdoBhcW4YM"  // Your Firebase Web API Key
#define DATABASE_URL "https://iot-and-esp-32-default-rtdb.asia-southeast1.firebasedatabase.app/"  // Your Firebase Realtime Database URL

// Define Firebase Data object
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

bool signupOK = false;

// Sensor pins
const int mq7Pin = 34;  // MQ-7 sensor pin
const int mq135Pin = 35;
const int ledPin = 16;  // MQ-135 sensor pin

// Calibration parameters for MQ-7
const float R0_MQ7 = 10.0;  // Baseline resistance in clean air in kOhms for MQ-7
const float slope_MQ7 = -0.6;  // Slope of the sensitivity curve
```

```
for MQ-7
const float intercept_MQ7 = 1.0;  // Intercept of the sensitivity
curve for MQ-7

// Calibration parameters for MQ-135
const float R0_MQ135 = 10.0;  // Baseline resistance in clean air
in kOhms for MQ-135
const float slope_MQ135 = -0.77;  // Slope of the sensitivity curve
for MQ-135
const float intercept_MQ135 = 1.2;

unsigned long sendDataPrevMillis = 0;

// Function to calculate CO subindex
float get_CO_subindex(float x) {
    if (x <= 1) {
        return x * 50 / 1;
    } else if (x > 1 && x <= 2) {
        return 50 + (x - 1) * 50 / 1;
    } else if (x > 2 && x <= 10) {
        return 100 + (x - 2) * 100 / 8;
    } else if (x > 10 && x <= 17) {
        return 200 + (x - 10) * 100 / 7;
    } else if (x > 17 && x <= 34) {
        return 300 + (x - 17) * 100 / 17;
    } else if (x > 34) {
        return 400 + (x - 34) * 100 / 17;
    } else {
        return 0;
    }
}

// Function to calculate NH3 subindex
float get_NH3_subindex(float x) {
    if (x <= 200) {
        return x * 50 / 200;
    } else if (x > 200 && x <= 400) {
        return 50 + (x - 200) * 50 / 200;
    } else if (x > 400 && x <= 800) {
        return 100 + (x - 400) * 100 / 400;
    } else if (x > 800 && x <= 1200) {
        return 200 + (x - 800) * 100 / 400;
    } else if (x > 1200 && x <= 1800) {
        return 300 + (x - 1200) * 100 / 600;
    } else if (x > 1800) {
        return 400 + (x - 1800) * 100 / 600;
    } else {
        return 0;
    }
}

// Function to calculate AQI
float calculate_aqi(float ni, float ci) {
    float aqi = 0;
    if (ni > ci) {
        aqi = ni;
    } else if (ci > ni) {
        aqi = ci;
    }
    return aqi;
}

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid,password);
    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    // Firebase config
    config.api_key = API_KEY;
    config.database_url = DATABASE_URL;
    if (Firebase.signUp(&config, &auth, "", "")){
        Serial.println("ok");
        signupOK = true;
    } else {
        Serial.printf("%s\n", config.signer.signupError.message.
c_str());
    }

    // Assign the callback function for the long running token
generation task
    config.token_status_callback = tokenStatusCallback;
      pinMode(ledPin, OUTPUT);  // see addons/TokenHelper.h

    // Begin Firebase with config and auth
    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);
}

void loop() {
    unsigned long currentMillis = millis();
    String timestamp = String(currentMillis);
    float aqi;
    if (Firebase.ready() && signupOK && (millis() -
sendDataPrevMillis > 10000 || sendDataPrevMillis == 0)) {
        sendDataPrevMillis = millis();

        float ppm_CO = readSensor(mq7Pin, R0_MQ7, slope_MQ7,
intercept_MQ7);
        float ppm_NH3 = readSensor(mq135Pin, R0_MQ135, slope_MQ135,
intercept_MQ135);

        Serial.print("CO PPM (MQ-7): ");
        Serial.println(ppm_CO);
        Serial.print("NH3 PPM (MQ-135): ");
        Serial.println(ppm_NH3);



        // Calculate subindices
        float ci = get_CO_subindex(ppm_CO);
        float ni = get_NH3_subindex(ppm_NH3);

        // Calculate AQI
        float aqi = calculate_aqi(ni, ci);
        if (aqi > 200) {
        // Blink the LED
        digitalWrite(ledPin, HIGH); // Turn the LED on
        delay(2000); // Wait for 500 milliseconds
        digitalWrite(ledPin, LOW); // Turn the LED off
        delay(2000); // Wait for 500 milliseconds
    }

        // Construct unique paths using a timestamp or count
        String path_CO = String("/pollutant_values/") + timestamp +
"/CO";
        String path_NH3 = String("/pollutant_values/") + timestamp
+ "/NH3";
        String path_AQI = String("/pollutant_values/") + timestamp
+ "/AQI";
        Serial.println(aqi);

        // Send data to Firebase
        sendToFirebase(path_CO.c_str(), ppm_CO);
        sendToFirebase(path_NH3.c_str(), ppm_NH3);
        sendToFirebase(path_AQI.c_str(), aqi);
    }
}

float readVoltage(int pin) {
    int sensorValue = analogRead(pin);
    return sensorValue * (3.3 / 4095.0); // Convert analog reading
```

```
to voltage
}

float calculateResistance(float voltage, float loadResistance) {
    return loadResistance * (3.3 / voltage - 1.0);
}

float calculatePPM(float Rs, float R0, float slope, float
intercept) {
    float ratio = Rs / R0;
    float ppm_log = (log10(ratio) - intercept) / slope;
    return pow(10, ppm_log);
}

float readSensor(int pin, float R0, float slope, float intercept) {
    float voltage = readVoltage(pin);
    float Rs = calculateResistance(voltage, R0);
    return calculatePPM(Rs, R0, slope, intercept);
}

void sendToFirebase(const char* path, float value) {
    if (Firebase.RTDB.setFloat(&fbdo, path, value)) {
        Serial.println("Data sent to Firebase successfully");
    } else {
        Serial.println("Failed to send data to Firebase");
        Serial.println(fbdo.errorReason());
    }
}
}
```