#include <SDS011.h>

#include <SPI.h>

#include <WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

#include "DHT.h"

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#include "esp32-hal-adc.h" // needed for adc pin reset

#include "soc/sens\_reg.h" // needed for adc pin reset

uint64\_t reg\_b; // Used to store Pin registers

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for SSD1306 display connected using software SPI (default case):

#define OLED\_MOSI 23

#define OLED\_CLK 18

#define OLED\_DC 4

#define OLED\_CS 5

#define OLED\_RESET 2

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT,

OLED\_MOSI, OLED\_CLK, OLED\_DC, OLED\_RESET, OLED\_CS);

const int numReadingsPM10 = 24;

const int numReadingsPM25 = 24;

const int numReadingsCO = 8;

WiFiClient client;

SDS011 my\_sds;

const char \*ssid = "Galaxy-M20"; // Enter your WiFi Name

const char \*pass = "ac312124"; // Enter your WiFi Password

#define MQTT\_SERV "io.adafruit.com"

#define MQTT\_PORT 1883

#define MQTT\_NAME "choudharyas" // Your Adafruit IO Username

#define MQTT\_PASS "988c4e045ef64c1b9bc8b5bb7ef5f2d9" // Adafruit IO AIO key

#define DHTTYPE DHT11 // DHT 11

uint8\_t DHTPin = 27;

DHT dht(DHTPin, DHTTYPE);

int error;

unsigned long interval = 3600000;

unsigned long previousMillis = 0;

int temperature, humidity, AQI;

float p10,p25;

int iMQ7 = 25;

int MQ7Raw = 0;

int MQ7ppm = 0;

double RvRo;

int ConcentrationINmgm3;

int readingsPM10[numReadingsPM10]; // the readings from the analog input

int readIndexPM10 = 0; // the index of the current reading

int totalPM10 = 0; // the running total

int averagePM10 = 0; // the average

int readingsPM25[numReadingsPM25]; // the readings from the analog input

int readIndexPM25 = 0; // the index of the current reading

int totalPM25 = 0; // the running total

int averagePM25 = 0; // the average

int readingsCO[numReadingsCO]; // the readings from the analog input

int readIndexCO = 0; // the index of the current reading

int totalCO = 0; // the running total

int averageCO = 0; // the average

Adafruit\_MQTT\_Client mqtt(&client, MQTT\_SERV, MQTT\_PORT, MQTT\_NAME, MQTT\_PASS);

Adafruit\_MQTT\_Publish AirQuality = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/AirQuality");

Adafruit\_MQTT\_Publish Temperature = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/Temperature");

Adafruit\_MQTT\_Publish Humidity = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/Humidity");

Adafruit\_MQTT\_Publish PM10 = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/PM10");

Adafruit\_MQTT\_Publish PM25 = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/PM25");

Adafruit\_MQTT\_Publish CO = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/CO");

//Adafruit\_MQTT\_Publish NH3 = Adafruit\_MQTT\_Publish(&mqtt,MQTT\_NAME "/f/NH3");

void setup()

{

my\_sds.begin(16,17);

Serial.begin(9600);

dht.begin();

display.begin(SSD1306\_SWITCHCAPVCC);

delay(10);

pinMode(DHTPin, INPUT);

pinMode(iMQ7, INPUT);

Serial.println("Connecting to ");

Serial.println(ssid);

reg\_b = READ\_PERI\_REG(SENS\_SAR\_READ\_CTRL2\_REG);

WiFi.begin(ssid, pass);

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

{

delay(550);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

for (int thisReading1 = 0; thisReading1 < numReadingsPM10; thisReading1++) {

readingsPM10[thisReading1] = 0;

}

for (int thisReading2 = 0; thisReading2 < numReadingsPM25; thisReading2++) {

readingsPM25[thisReading2] = 0;

}

for (int thisReading3 = 0; thisReading3 < numReadingsCO; thisReading3++) {

readingsCO[thisReading3] = 0;

}

display.clearDisplay();

display.display();

}

void loop()

{

unsigned long currentMillis = millis();

MQTT\_connect();

if ((unsigned long)(currentMillis - previousMillis) >= interval) {

WRITE\_PERI\_REG(SENS\_SAR\_READ\_CTRL2\_REG, reg\_b);

MQ7Raw = analogRead( iMQ7 );

Serial.print("MQ Raw: ");

Serial.println(MQ7Raw);

RvRo = MQ7Raw \* (3.3 / 4095);

MQ7ppm = 3.027\*exp(1.0698\*( RvRo ));

Serial.print("CO: ");

Serial.println(MQ7ppm);

//Serial.println();

error = my\_sds.read(&p25,&p10);

if (! error) {

Serial.println("P2.5: "+String(p25));

Serial.println("P10: "+String(p10));

}

}

temperature = dht.readTemperature();

humidity = dht.readHumidity();

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println();

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.println();

ConcentrationINmgm3 = MQ7ppm\* (28.06/24.45); //Converting PPM to mg/m3. Where 28.06 is Molecular mass of CO and 24.45 is the Molar volume

Serial.print("mg/m3: "); // for more inforation on this follow: https://www.markes.com/Resources/Frequently-asked-questions/How-do-I-convert-units.aspx

Serial.println(ConcentrationINmgm3);

totalPM10 = totalPM10 - readingsPM10[readIndexPM10];

readingsPM10[readIndexPM10] = p10;

totalPM10 = totalPM10 + readingsPM10[readIndexPM10];

readIndexPM10 = readIndexPM10 + 1;

if (readIndexPM10 >= numReadingsPM10) {

readIndexPM10 = 0;

}

averagePM10 = totalPM10 / numReadingsPM10;

Serial.print("PM10 Average: ");

Serial.println(averagePM10);

totalPM25 = totalPM25 - readingsPM25[readIndexPM25];

readingsPM25[readIndexPM25] = p25;

totalPM25 = totalPM25 + readingsPM25[readIndexPM25];

readIndexPM25 = readIndexPM25 + 1;

if (readIndexPM25 >= numReadingsPM25) {

readIndexPM25 = 0;

}

averagePM25 = totalPM25 / numReadingsPM25;

Serial.print("PM2.5 Average: ");

Serial.println(averagePM25);

totalCO = totalCO - readingsCO[readIndexCO];

readingsCO[readIndexCO] = ConcentrationINmgm3;

totalCO = totalCO + readingsCO[readIndexCO];

readIndexCO = readIndexCO + 1;

if (readIndexCO >= numReadingsCO) {

readIndexCO = 0;

}

averageCO = totalCO / numReadingsCO;

Serial.print("CO Average: ");

Serial.println(averageCO);

if (averagePM10 > averagePM25){

AQI = averagePM10;

}

else {

AQI = averagePM25;

}

if (! Temperature.publish(temperature))

{

delay(30000);

}

if (! Humidity.publish(humidity))

{

delay(30000);

}

if (! PM10.publish(averagePM10))

{

delay(30000);

}

if (! PM25.publish(averagePM25))

{

delay(30000);

}

if (! CO.publish(MQ7ppm))

{

delay(30000);

}

if (! AirQuality.publish(AQI))

{

delay(30000);

}

displayvalues();

delay(10000);

//delay(3000);

}

void displayvalues()

{

display.clearDisplay();

display.setTextSize(2);

display.setCursor(0,15);

display.println("CO: ");

display.setCursor(40,15);

display.println(averageCO);

display.setTextSize(1);

display.setCursor(68,35);

display.println("mg/m3");

display.display();

delay(2000);

display.clearDisplay();

display.setTextSize(2);

display.setCursor(0,5);

display.println("Temp:");

display.setCursor(75,5);

display.println(temperature);

display.setCursor(101,5);

display.println("C");

display.setCursor(0,28);

display.println("Humid:");

display.setCursor(75,28);

display.println(humidity);

display.setCursor(101,28);

display.println("%");

display.display();

display.clearDisplay();

delay(2000);

display.setTextSize(2);

display.setTextColor(WHITE);

display.setCursor(0,5);

display.println("PM2.5: ");

display.setCursor(75,5);

display.println(averagePM25);

display.setCursor(0,28);

display.println("PM10: ");

display.setCursor(75,28);

display.println(averagePM10);

display.setTextSize(1);

display.setCursor(90,49);

display.println("ug/m3");

display.display();

delay(2000);

}

void MQTT\_connect()

{

int8\_t ret;

// Stop if already connected.

if (mqtt.connected())

{

return;

}

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) // connect will return 0 for connected

{

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0)

{

// basically die and wait for WDT to reset me

while (1);

}

}

}