AIR QUALITY MONITORING USING IOT DEVICES

R.K.SANGEETHA ,P.S.MADHUMITHA, V.S.S.SRIANNAAMALAI

INTRODUCTION:

For the air quality monitoring project, we are going to use Adafruit_Sensor, Adafruit_BME280, and TFT_ILI9163C and ESP32. The code mentioned below is an Arduino sketch for an ESP32-based air quality and environmental monitoring system. It uses various sensors, including a Plantower PMS sensor for particulate matter (PM) measurements and an Adafruit BME280 sensor for temperature, humidity, and pressure measurements. The code also interfaces with a TFT display and uploads the sensor data to ThingSpeak(is an IoT analytics platform service) for monitoring.

LIBRARIES:

The code includes several libraries for sensor communication, display, and network communication. Some of the key libraries used are Wire, SPI, Adafruit_Sensor, Adafruit_BME280, and TFT_ILI9163C.

WIFI and THINGSPEAK:

The code connects to a Wi-Fi network with the provided SSID and password.

It sets up a ThingSpeak channel with a channel number and API key for data logging.

SENSORS:

It initializes the BME280 sensor for temperature, humidity, and pressure measurements.

The BME280 sensor's data is displayed on an ILI9163C TFT display.

The SDS011 sensor (presumed to be connected but not shown in the code) is turned on and off as needed for PM measurements.

Data Processing:

The code reads data from the BME280 sensor and normalizes PM2.5 and PM10 values using a normalization function. The data is displayed on the TFT display and printed to the Serial Monitor.

Data Upload:

The code checks the Wi-Fi status and uploads sensor data to ThingSpeak using HTTP GET requests. It uploads PM2.5, PM10, pressure, temperature, and humidity data. After successful data uploads, it clears the display screen.

Power Management:

The code provides functions to turn off and on the SDS011 sensor, presuming it supports sleep modes.

Loop Function:

In the loop() function, the code follows a sequence of turning on the SDS011 sensor, reading PM data, normalizing the data, displaying it, and uploading it to ThingSpeak.

CODE: (IMPLEMENTATION FOR ARDUINO SKETCH FOR ESP32 IN C++)

```
#include <Wire.h>

#include "processdata.h"

#include <SPI.h>

#include "iot_Sensor.h"

#include "Adafruit_BME280.h"

#include <WiFi.h>

#include <HTTPClient.h>

#include <esp_wifi.h>

#include <Adafruit_GFX.h>

#include <TFT_ILI9163C.h>


// Color definitions

#define BLACK   0x0000

#define BLUE    0x001F

#define RED     0xF800

#define GREEN   0x07E0

#define CYAN    0x07FF

#define MAGENTA 0xF81F

#define YELLOW  0xFFE0

#define WHITE   0xFFFF


#define SEALEVELPRESSURE_HPA (1013.25)

TFT_ILI9163C tft = TFT_ILI9163C(2,15,4);  //(CS,A0,RST);



char ssid[] = "....";    //  your network SSID (name)

char pass[] = "..........";   // your network password
```

```
int status = WL_IDLE_STATUS;
 WiFiClient  client;
uint64_t gap=15*1000000;
unsigned long myChannelNumber = 279012;  //replace by your channel number
String  myWriteAPIKey = "...............";// replace by your API key

const char* server = "api.thingspeak.com";
String poststr;
int httpCode;

Adafruit_BME280 bme;
float h,t,f,g;
int lc=13;

long previousMillis = 0;
const int analogInPin = A0;
int latch = 8;
int srclk = 9;
int ser = 10;
int i=0;
unsigned char displayTemp[8];
#define RXD2 16
#define TXD2 17
HTTPClient http;

void setup() {
pinMode(lc,OUTPUT);
digitalWrite(lc,HIGH);

 Serial.begin(115200);
   WiFi.begin(ssid, pass);
   while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    }
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
//  Serial.println("Serial Txd is on pin: "+String(TX));
//  Serial.println("Serial Rxd is on pin: "+String(RX));
  Pm25 = 0;
  Pm10 = 0;
bool status;
status = bme.begin(0x76); // I2C (SDA,SCL);
if (!status) {
Serial.println("Could not find a valid BME280 sensor, check wiring!");
while (1);
    }

tft.begin();
tft.setTextSize(1);
tft.setTextColor(MAGENTA,BLACK);
tft.setCursor(0,0);
tft.println("ESP32 Pollution Rpt");

}

void loop() {
  turnon();
  delay(2000);
  digitalWrite(lc,HIGH);

  ProcessSerialData();
  printValues();

// Normalization function written by Zbyszek Kiliański, Piotr Paul
// https://github.com/piotrkpaul/esp8266-sds011

 Pm25 = Pm25 / (1.0 + 0.48756 * pow((h / 100.0), 8.60068));
 Pm10 = Pm10 / (1.0 + 0.81559 * pow((h / 100.0), 5.83411));

tft.setTextColor(MAGENTA,BLACK);
tft.setCursor(0,0);
tft.println("ESP32 Pollution Rpt");

  Serial.println();
  Serial.print("Pm2.5 ");
  Serial.print(float(Pm25) / 10.0);
  Serial.println();//  Display();
  Serial.print("Pm10 ");
  Serial.print(float(Pm10) / 10.0);
  Serial.println();
  tft.setCursor(0,10);
  tft.setTextColor(WHITE,BLACK);
  tft.print("PM 2.5:");
  tft.println(float(Pm25) / 10.0);
  tft.setTextColor(GREEN,BLACK);
  tft.print("PM  10:");
  tft.println(float(Pm10) / 10.0);
  tft.setTextColor(MAGENTA,BLACK);
  tft.print("RH:");
  tft.println(h);
  tft.setTextColor(YELLOW,BLACK);
  tft.print("Altitude:");
  tft.println(f);
  tft.setTextColor(RED,BLACK);
  tft.print("Pressure:");
  tft.println(g);
  tft.setTextColor(CYAN,BLACK);
  tft.print("Temperature:");
  tft.println(t);
  tft.setCursor(0,10);


  delay(2000);
if(WiFi.status()== WL_CONNECTED){   //Check WiFi connection status
  if(Pm25>0) {upload("field2",float(Pm25)/10);
  tft.setCursor(0,60);
  tft.println("PM2.5 uploaded");
  delay(15000);}
  if(Pm10>0) {upload("field1",float(Pm10)/10);
  tft.setCursor(0,60);
  tft.println("PM10 uploaded");
  delay(15000);}
  tft.setTextColor(YELLOW,BLACK);
  digitalWrite(lc,LOW);
  turnoff();
  delay(2000);
  tft.setCursor(0,70);
  tft.println("SDS011 put to sleep");
  tft.setCursor(0,10);
  if(g>0) {upload("field4",g);
  tft.setCursor(0,60);
  tft.println("Pressure uploaded");
  delay(15000);}
  if(t>0){ upload("field3",t);
  tft.setCursor(0,60);
  tft.println("Temperature uploaded");
  delay(15000);  }


  if(h>0){ upload("field5",h);
  tft.setCursor(0,60);
  tft.println("Humidity uploaded");
  delay(15000);}
  tft.clearScreen();
//  esp_deep_sleep(60000000); //one minute off time
}
}
void turnoff() {
Serial2.flush();
for (uint8_t j = 0; j < 19; j++) Serial2.write(SLEEPCMD[j]);
}


void turnon() {
  Serial2.write(0x01);
}
void upload(String field, float datax){
  String poststr="http://api.thingspeak.com/update?api_key=0T2YL145P41LNHKA&"+field+"="; //
enter your own Write API key
poststr +=String(datax);
 http.begin(poststr);
 int  httpCode=http.GET();
      if (httpCode > 0) { //Check for the returning code
        String payload = http.getString();
         Serial.println(poststr);
      }
     else {
      Serial.println("Error on HTTP request");
    }
    datax=0.0;
}
void printValues() {
   h = bme.readHumidity();   // read humidity
   t = bme.readTemperature();  // Read temperature as Celsius
   f = bme.readAltitude(SEALEVELPRESSURE_HPA);
   g = bme.readPressure() / 100.0; //analogRead(A6);  //D34
    Serial.print("Temperature = ");
    Serial.print(t);
    Serial.println(" *C");
    Serial.print("Pressure = ");
    Serial.print(g,2);
    Serial.println(" hPa");
    Serial.print("Approx. Altitude = ");
    Serial.print(f);
    Serial.println(" m");
    Serial.print("Humidity = ");
    Serial.print(h);
    Serial.println(" %");
    Serial.println(i);
    i=i+1;
}
```