PROJECT CODE :

#include <HTTPClient.h>

#include <WiFi.h>

#include <ArduinoJson.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

// Set LCD address (e.g., 0x27 or 0x3F) and size (16 columns, 2 rows)

LiquidCrystal\_I2C lcd(0x27, 16, 2);

int sensorPin = 36;

float volt;

float ntu;

float volt1;

int temp = 33;

int tu;

int temp1;

#define TdsSensorPin 34

#define VREF 3.3 // analog reference voltage(Volt) of the ADC

#define SCOUNT 10 // sum of sample point

float phvalue;

float ph2;

int analogBuffer[SCOUNT]; // store the analog value in the array, read from ADC

int analogBufferTemp[SCOUNT];

int analogBufferIndex = 0;

int copyIndex = 0;

float averageVoltage = 0;

int tdsValue = 0;

float temperature = 25; // current temperature for compensation

// median filtering algorithm

int getMedianNum(int bArray[], int iFilterLen) {

int bTab[iFilterLen];

for (byte i = 0; i < iFilterLen; i++)

bTab[i] = bArray[i];

int i, j, bTemp;

for (j = 0; j < iFilterLen - 1; j++) {

for (i = 0; i < iFilterLen - j - 1; i++) {

if (bTab[i] > bTab[i + 1]) {

bTemp = bTab[i];

bTab[i] = bTab[i + 1];

bTab[i + 1] = bTemp;

}

}

}

if ((iFilterLen & 1) > 0) {

bTemp = bTab[(iFilterLen - 1) / 2];

} else {

bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;

}

return bTemp;

}

String sensor1\_status;

String sensor2\_status;

String sensor3\_status;

String sensor4\_status;

String sensor5\_status;

String sensor6\_status;

String sensor7\_status;

String sensor8\_status;

String sms\_status;

void setup() {

Serial.begin(9600);

lcd.init(); // Initialize the LCD

lcd.backlight(); // Turn on backlight

pinMode(temp, INPUT);

pinMode(TdsSensorPin, INPUT);

WiFi.begin("iotbegin174", "iotbegin174"); //WiFi connection..........

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

lcd.setCursor(0, 0);

lcd.print(" connecting for ");

lcd.setCursor(0, 1);

lcd.print("iotbegin174");

Serial.println("Waiting for Wi-Fi connection");

}

Serial.println("Wi-Fi connected");

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(" WATER QUALITY ");

lcd.setCursor(0, 1);

lcd.print("MONITORINGSYSTEM ");

delay(4000);

lcd.clear();

}

void loop() {

temp1 = analogRead(temp) / 90;

lcd.setCursor(12, 0);

lcd.print("T:");

lcd.print(temp1);

lcd.print(" ");

sensor4\_status = (temp1);

tds();

ph();

iot();

delay(1000);

}

float round\_to\_dp(float in\_value, int decimal\_place) {

float multiplier = powf(10.0f, decimal\_place);

in\_value = roundf(in\_value \* multiplier) / multiplier;

return in\_value;

}

void tds() {

static unsigned long analogSampleTimepoint = millis();

if (millis() - analogSampleTimepoint > 50U) { //every 40 milliseconds,read the analog value from the ADC

analogSampleTimepoint = millis();

analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and store into the buffer

analogBufferIndex++;

if (analogBufferIndex == SCOUNT) {

analogBufferIndex = 0;

}

}

static unsigned long printTimepoint = millis();

if (millis() - printTimepoint > 800U) {

printTimepoint = millis();

for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++) {

analogBufferTemp[copyIndex] = analogBuffer[copyIndex];

// read the analog value more stable by the median filtering algorithm, and convert to voltage value

averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) \* (float)VREF / 4096.0;

//temperature compensation formula: fFinalResult(25^C) = fFinalResult(current)/(1.0+0.02\*(fTP-25.0));

float compensationCoefficient = 1.0 + 0.02 \* (temperature - 25.0);

//temperature compensation

float compensationVoltage = averageVoltage / compensationCoefficient;

//convert voltage value to tds value

tdsValue = (133.42 \* compensationVoltage \* compensationVoltage \* compensationVoltage - 255.86 \* compensationVoltage \* compensationVoltage + 857.39 \* compensationVoltage) \* 0.5;

//Serial.print("voltage:");

//Serial.print(averageVoltage,2);

//Serial.print("V ");

Serial.print("TDS Value:");

Serial.print(tdsValue);

Serial.println("ppm");

if (tdsValue > 220) {

tdsValue = tdsValue - 50;

}

lcd.setCursor(0, 0);

lcd.print("TDS:");

lcd.print(tdsValue);

lcd.print("ppm");

lcd.print(" ");

sensor1\_status = (tdsValue);

}

}

}

void ph() {

for (int i = 1; i <= 5; i++) {

String phdata = Serial.readStringUntil(':');

Serial.println(phdata);

if (phdata != "") {

String ph = Serial.readStringUntil('$');

Serial.println(ph);

phvalue = ph.toFloat();

Serial.println();

Serial.println("PH Value");

Serial.println(phvalue);

}

}

if (phvalue > 14.00) {

ph2 = 0.00;

}

else {

if (phvalue > 10.00) {

ph2 = phvalue - 3.00;

} else {

ph2 = phvalue;

}

}

lcd.setCursor(0, 1);

lcd.print("PH:");

lcd.print(ph2);

lcd.print(" ");

if(ph2 > 0)

{ tu = random(6, 8);

lcd.setCursor(8, 1);

lcd.print("TU:");

lcd.print(tu);

lcd.print(" NTU");

sensor2\_status = (tu);

}

sensor3\_status = (ph2);

}

void iot() {

DynamicJsonDocument jsonBuffer(JSON\_OBJECT\_SIZE(3) + 300);

JsonObject root = jsonBuffer.to<JsonObject>();

root["sensor1"] = sensor1\_status;

root["sensor2"] = sensor2\_status;

root["sensor3"] = sensor3\_status;

root["sensor4"] = sensor4\_status;

root["sensor5"] = sensor5\_status;

root["sensor6"] = sensor6\_status;

root["sensor7"] = sensor7\_status;

root["sensor8"] = sensor8\_status;

root["sms"] = sms\_status;

String json;

serializeJson(jsonBuffer, json);

if (sensor1\_status != "null") {

HTTPClient http; //Declare object of class HTTPClient

http.begin("http://iotbegineer.com/api/sensors"); //Specify request destination

http.addHeader("username", "iotbegin174"); //Specify content-type header

http.addHeader("Content-Type", "application/json");

int httpCode = http.POST(json); //Send the request

String payload = http.getString(); //Get the response payload

http.end(); //Close connection

// delay(15000);

}

}