IOT 최종 코드

// Triger pin number 7 setup

int pin\_UL\_TRIG = 31;

// Echo pin number 8 setup

int pin\_UL\_OUT = 30;

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#define COV\_RATIO 0.2 //ug/m3

#define NO\_DUST\_VOLTAGE 400 //mv

#define SYS\_VOLTAGE 5000

#include <SPI.h>

#include <WiFi.h>

#include <dht11.h>

dht11 DHT11;

int pin\_DHT11 = 40;

//Step motor OUT1A OUT1B OUT2A OUT2B

int pin\_STEP[4] = {25, 26, 27, 28};

/\*

I/O define

\*/

const int iled = 34; //drive the led of sensor

const int vout = A0; //analog input

/\*

variable

\*/

float old\_density, density, voltage;

int adcvalue;

unsigned long microseconds, distance\_cm;

/\*

private function

\*/

int Filter(int m)

{

static int flag\_first = 0, \_buff[10], sum;

const int \_buff\_max = 10;

int i;

if(flag\_first == 0)

{

flag\_first = 1;

for(i = 0, sum = 0; i < \_buff\_max; i++)

{

\_buff[i] = m;

sum += \_buff[i];

}

return m;

}

else

{

sum -= \_buff[0];

for(i = 0; i < (\_buff\_max - 1); i++)

{

\_buff[i] = \_buff[i + 1];

}

\_buff[9] = m;

sum += \_buff[9];

i = sum / 10.0;

return i;

}

}

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

char pass[] = "12345678"; // your network password

int keyIndex = 0; // your network key Index number (needed only for WEP)

int status = WL\_IDLE\_STATUS;

WiFiServer server(80);

void setup() {

// Stepmotor pin OUTPUT setup

for(int i=0; i<4; i++)

pinMode(pin\_STEP[i], OUTPUT);

//Initialize serial and wait for port to open:

Serial.begin(9600);

// Out pin Input Setup

pinMode(pin\_UL\_OUT, INPUT);

// Trig pin Output Setup

pinMode(pin\_UL\_TRIG, OUTPUT);

digitalWrite(pin\_UL\_TRIG, 0);

pinMode(iled, OUTPUT);

digitalWrite(iled, LOW);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB port only

}

// check for the presence of the shield:

if (WiFi.status() == WL\_NO\_SHIELD) {

Serial.println("WiFi shield not present");

// don't continue:

while (true);

}

String fv = WiFi.firmwareVersion();

if (fv != "1.1.0") {

Serial.println("Please upgrade the firmware");

}

// attempt to connect to Wifi network:

while (status != WL\_CONNECTED) {

Serial.print("Attempting to connect to SSID: ");

Serial.println(ssid);

// Connect to WPA/WPA2 network. Change this line if using open or WEP network:

status = WiFi.begin(ssid, pass);

// wait 10 seconds for connection:

delay(10000);

}

server.begin();

// you're connected now, so print out the status:

printWifiStatus();

}

void loop() {

// listen for incoming clients

WiFiClient client = server.available();

if (client) {

Serial.println("new client");

// an http request ends with a blank line

boolean currentLineIsBlank = true;

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

// if you've gotten to the end of the line (received a newline

// character) and the line is blank, the http request has ended,

// so you can send a reply

if (c == '\n' && currentLineIsBlank) {

// send a standard http response header

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close"); // the connection will be closed after completion of the response

client.println("Refresh: 5"); // refresh the page automatically every 5 sec

client.println();

client.println("<!DOCTYPE HTML>");

client.println("<html>");

//Measure Temperature & Humidity by DHT11 sensor

int chk = DHT11.read(pin\_DHT11);

//Ultrasonic wave transmission to check the window is closed or not

digitalWrite(pin\_UL\_TRIG, 0); // Output pin\_ULTRASONIC\_T to LOW

delayMicroseconds(2);

// pull the Trig pin to high level for more than 10us impulse

digitalWrite(pin\_UL\_TRIG, 1); // Output pin\_ULTRASONIC\_T to HIGH

delayMicroseconds(10);

digitalWrite(pin\_UL\_TRIG, 0); // Output pin\_ULTRASONIC\_T to LOW

//Measure microdust density

digitalWrite(iled, HIGH);

delayMicroseconds(280);

adcvalue = analogRead(vout);

digitalWrite(iled, LOW);

adcvalue = Filter(adcvalue);

voltage = (SYS\_VOLTAGE / 1024.0) \* adcvalue \* 11;

if(voltage >= NO\_DUST\_VOLTAGE)

{

voltage -= NO\_DUST\_VOLTAGE;

density = voltage \* COV\_RATIO;

}

else

{

density = 0;

}

// waits for the pin to go HIGH, and returns the length of the pulse in microseconds

microseconds = pulseIn(pin\_UL\_OUT, 1, 24000);

distance\_cm = microseconds \* 17/1000; // Calculate distance from time

client.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 16 Jo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

client.println("<br />");

client.print("The current dust concentration is: ");

client.print(density);

client.print("ug/m3");

client.print("<br />");

client.print("Temperature : ");

client.print(DHT11.temperature);

client.print("[C]");

client.print("<br />");

client.print("Humidity : ");

client.print(DHT11.humidity);

client.print("[%]");

client.print("<br />");

client.print(distance\_cm);

client.print(" cm");

client.println("<br />");

client.println("</html>");

if(DHT11.humidity>=70 || density>=100){ // Humidity 70%, Microdust 100 ug/m3 -> Window closed

if(distance\_cm > 5){

turnStepmotor();

}

}

break;

}

if (c == '\n') {

// you're starting a new line

currentLineIsBlank = true;

} else if (c != '\r') {

// you've gotten a character on the current line

currentLineIsBlank = false;

}

}

}

// give the web browser time to receive the data

delay(1);

// close the connection:

client.stop();

Serial.println("client disonnected");

}

}

void turnStepmotor() {

// 1상 제어 시계방향 회전

for(int i=0; i<1024 ; i++)

{

digitalWrite(pin\_STEP[3], 1);

digitalWrite(pin\_STEP[2], 0);

digitalWrite(pin\_STEP[1], 0);

digitalWrite(pin\_STEP[0], 0);

delay(3);

digitalWrite(pin\_STEP[3], 0);

digitalWrite(pin\_STEP[2], 1);

digitalWrite(pin\_STEP[1], 0);

digitalWrite(pin\_STEP[0], 0);

delay(3);

digitalWrite(pin\_STEP[3], 0);

digitalWrite(pin\_STEP[2], 0);

digitalWrite(pin\_STEP[1], 1);

digitalWrite(pin\_STEP[0], 0);

delay(3);

digitalWrite(pin\_STEP[3], 0);

digitalWrite(pin\_STEP[2], 0);

digitalWrite(pin\_STEP[1], 0);

digitalWrite(pin\_STEP[0], 1);

delay(3);

}

}

void printWifiStatus() {

// print the SSID of the network you're attached to:

Serial.print("SSID: ");

Serial.println(WiFi.SSID());

// print your WiFi shield's IP address:

IPAddress ip = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ip);

// print the received signal strength:

long rssi = WiFi.RSSI();

Serial.print("signal strength (RSSI):");

Serial.print(rssi);

Serial.println(" dBm");

}