#include <Adafruit\_Sensor.h> // Required for all Adafruit Unified Sensor based libraries.

#include <ESP8266WiFi.h> // Arduino library with built in re-connect functionality.

#include <PubSubClient.h> // A client library for MQTT messaging.

#include "DHT.h" // Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors

#define DHTTYPE DHT11 ; //Macro Arguments. Function-like macros can take arguments, just like true functions.To  define a macro that uses arguments, you insert parameters between the pair of parentheses in the macro definition that make the macro function-like

WiFiClient smartbuilding; // WiFiClient is used to connect to the internet. Creates a client that can connect to to a specified internet IP address and port as defined in [client.connect()](https://www.arduino.cc/en/Reference/WiFiClientConnect). here smartbuilding is the name of client.

PubSubClient client(smartbuilding); // Creates a partially initialised client instance.

//Variable declaration

const char\* ssid = "NETGEAR86";

const char\* password = "greatairplane971";

const char\* mqtt\_server = "iot.iandwe.in";

int LED =D2;

int gate =D3;

int fan= D4;

int ac=D5;

float t;

const int DHTPin= D1;

long now = millis();

long lastMeasure = 0;

DHT dht(DHTPin,DHT11); //DHT is a class and dht is tha name of object inwhci we are sending pin number and DHT11

void setup()

{

Serial.begin(115200); // print the values on serial monitor where "115200" is the baud rate.

//Initialization

pinMode(LED, OUTPUT); // initialize digital pin LED as an output.

pinMode(gate, OUTPUT); // initialize digital pin gate as an output.

pinMode(fan, OUTPUT); // pin fan as output

pinMode(ac, OUTPUT); // pin ac as output

dht.begin(); //Initialize dht11

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

}

// Connecting to a WiFi network

void setup\_wifi()

{

delay(10);

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void callback(char\* topic, byte\* payload, unsigned int length) // In order to republish this payload, a copy must be made  
 // as the orignal payload buffer will be overwritten whilst  
 // constructing the PUBLISH packet

{

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("message");

String messageTemp;

String Topic = (String)topic;

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

{

Serial.print((char)payload[i]);

messageTemp +=(char)payload[i];

}

Serial.println();

if (Topic == "tiu/light")

{

Serial.print("Changing room lamp to:");

if(messageTemp == "on")

{

digitalWrite(LED, HIGH);

Serial.print("on");

}

else if(messageTemp == "off")

{

digitalWrite(LED, LOW);

Serial.print("off");

}

}

if (Topic == "tiu/gate"){

Serial.print("initializing the gate to:");

if(messageTemp == "open")

{

digitalWrite(gate,HIGH);

Serial.print("gate open");

}

else if(messageTemp == "close")

{

digitalWrite(gate,LOW);

Serial.print("gate close");

}

}

if (Topic == "tiu/fan")

{

Serial.print("changing fan to:");

if(messageTemp == "on")

{

digitalWrite(fan,HIGH);

Serial.print("fan is on");

}

else if(messageTemp == "off")

{

digitalWrite(fan,LOW);

Serial.print("fan is off");

} } }

void reconnect()

{

// Loop until we're reconnected

while (!client.connected())

{

Serial.print("Attempting MQTT connection...");

if (client.connect("smartbuilding")) // Attempt to connect

{

Serial.println("connected"); // Once connected, publish an announcement..

client.subscribe("tiu/light"); // ... and resubscribe

client.subscribe("tiu/gate");

client.subscribe("tiu/fan");

client.subscribe("tiu/ac");

}

else

{

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds"); // Wait 5 seconds before retrying

delay(5000);

} } }

void ac\_on(float t)

{

if(t>=32)

{

digitalWrite(ac,HIGH);

Serial.println("AC is on");

}

else if(t<28)

{

digitalWrite(ac,LOW);

Serial.println("AC is off");

} }

void loop() // code runs repeatedly:

{

if (!client.connected()) {

reconnect(); // calling function reconnect();

}

client.loop(); //calling callback function

now = millis();

if (now - lastMeasure > 2000)

{

lastMeasure = now;

float h = dht.readHumidity();

float t = dht.readTemperature();

if (isnan(h) || isnan(t))

{

Serial.println("Failed to read from DHT sensor!");

return;

}

float hic = dht.computeHeatIndex(t, h, false);

static char temperatureTemp[7];

dtostrf(t, 6, 2, temperatureTemp);

static char humidityTemp[7];

dtostrf(h, 6, 2, humidityTemp);

client.publish("room/temperature", temperatureTemp); //publishing to node red

client.publish("tiu/humidity", humidityTemp);

Serial.println("Humidity : ");

Serial.println(h);

Serial.println("Temperature : ");

Serial.println(t);

Serial.println(" ");

ac\_on(t); // calling function ac\_on();

}

}