#include <ESP8266WiFi.h> // library needed for WiFi setup

#include <PubSubClient.h> // library needed for publish/subsribe to MQTT server as a client

#include "DHT.h" // libary needed for DHT11 temperature sensors

#include<SPI.h> // libary needed for RFID function

#include<MFRC522.h> // libary needed for RFID sensor

#define DHTTYPE DHT11 // DHT sensor type : DHT 11

//RFID

//creating mfrc522 instance

#define RSTPIN 2

#define SSPIN 15

MFRC522 rc(SSPIN, RSTPIN);

int readsuccess;

/\* the following are the UIDs of the card which are authorised

to know the UID of your card/tag use the example code 'DumpInfo'

from the library mfrc522 it give the UID of the card as well as

other information in the card on the serial monitor of the arduino\*/

byte defcard[][4]={{0xB7,0xC1,0x59,0x0B},{0x21,0x63,0x75,0x63}}; //for multiple cards, new ID cards should be added here to be authorized

int N=2; //change this to the number of cards/tags you will use (number of above IDs)

byte readcard[4]; //stores the UID of current tag which is read

//function to get the UID of the card

int getid(){

if(!rc.PICC\_IsNewCardPresent()){

return 0;

}

if(!rc.PICC\_ReadCardSerial()){

return 0;

}

Serial.println("THE UID OF THE SCANNED CARD IS:");

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

readcard[i]=rc.uid.uidByte[i]; //storing the UID of the tag in readcard

Serial.print(readcard[i],HEX);

}

Serial.println("");

Serial.println("Now Comparing with Authorised cards");

rc.PICC\_HaltA();

return 1;

}

int uidnum=99;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Change the credentials below, so your ESP8266 connects to your router

const char\* ssid = "AndroidAP";

const char\* password = "ahmed2014";

// Change the variable to your Arduino IP address, so it connects to your MQTT broker

const char\* mqtt\_server = "192.168.43.214";

// Initializes the espClient. You should change the espClient name if you have multiple ESPs running in your home automation system

WiFiClient espClient;

PubSubClient client(espClient);

// DHT Sensor - GPIO 5 = D1 on ESP-12E NodeMCU board

const int DHTPin = 5;

const int DHTPin2 = 4;

// Lamp - LED - GPIO 4 = D2 on ESP-12E NodeMCU board

const int lamp = 16;

float difference; // tempreture diffrence

// Initialize DHT sensor.

DHT dht(DHTPin, DHTTYPE);

DHT dht2(DHTPin2, DHTTYPE);

// Timers auxiliar variables

long now = millis();

long lastMeasure = 0;

// Don't change the function below. This functions connects your ESP8266 to your router

void setup\_wifi() {

delay(10);

// We start by connecting to a WiFi network

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.print("WiFi connected - ESP IP address: ");

Serial.println(WiFi.localIP());

}

// This functions is executed when some device publishes a message to a topic that your ESP8266 is subscribed to

// Change the function below to add logic to your program, so when a device publishes a message to a topic that

// your ESP8266 is subscribed you can actually do something

void callback(String topic, byte\* message, unsigned int length) {

Serial.print("Message arrived on topic: ");

Serial.print(topic);

Serial.print(". Message: ");

String messageTemp;

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

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

messageTemp += (char)message[i];

}

Serial.println();

// Feel free to add more if statements to control more GPIOs with MQTT

// If a message is received on the topic "room/lamp", you check if the message is either on or off. Turns the lamp GPIO according to the message

if(topic=="room/lamp"){

Serial.print("Changing Room lamp to ");

if(messageTemp == "on"){

digitalWrite(lamp, HIGH);

Serial.print("On");

}

else if(messageTemp == "off"){

digitalWrite(lamp, LOW);

Serial.print("Off");

}

}

Serial.println();

}

// This functions reconnects your ESP8266 to your MQTT broker

// Change the function below if you want to subscribe to more topics with your ESP8266

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

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

// Attempt to connect

if (client.connect("ESP8266Client")) {

Serial.println("connected");

// Subscribe or resubscribe to a topic

// You can subscribe to more topics (to control more LEDs in this example) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

client.subscribe("room/lamp"); // Subscribing to Topic "room/lamp" used to switch the A/C (LED) ON/OFF by the IoT dashboard "A/C Remote Switch"

} else {

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

Serial.print(client.state());

Serial.println(" try again in 5 seconds");

// Wait 5 seconds before retrying

delay(5000);

}

}

}

// The setup function sets your ESP GPIOs to Outputs, starts the serial communication at a baud rate of 115200

// Sets your mqtt broker and sets the callback function

// The callback function is what receives messages and actually controls the LEDs

void setup() {

pinMode(lamp, OUTPUT);

dht.begin();

Serial.begin(115200); //set baud rate for serial communication between arduino and c to 115200

setup\_wifi(); // call the function to connect the wifi

client.setServer(mqtt\_server, 1883); //connect to mqtt server using ip and port

client.setCallback(callback); //run the procedure for resiving topics from mqtt server

SPI.begin();

rc.PCD\_Init(); //initialize the receiver

rc.PCD\_DumpVersionToSerial(); //show details of card reader module

}

// For this project, you don't need to change anything in the loop function. Basically it ensures that you ESP is connected to your broker

void loop() {

//keep cheking if wifi and mqtt is still connected

if (!client.connected()) {

reconnect();

}

if(!client.loop())

client.connect("ESP8266Client");

now = millis();

// Publishes new temperature and humidity every 10 seconds

if (now - lastMeasure > 10000) {

lastMeasure = now;

// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)

float h = dht.readHumidity();

float h2 = dht2.readHumidity();

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

float t2 = dht2.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

float f = dht.readTemperature(true);

float f2 = dht2.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t) || isnan(f)) {

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

return;

}

if (isnan(h2) || isnan(t2) || isnan(f2)) {

Serial.println("Failed to read from DHT2 sensor!");

return;

}

// Computes temperature values in Celsius

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

float hic2 = dht2.computeHeatIndex(t2, h2, false);

static char temperatureTemp[7];

static char temperatureTemp2[7];

dtostrf(hic, 6, 2, temperatureTemp);

dtostrf(hic2, 6, 2, temperatureTemp2);

static char humidityTemp[7];

static char humidityTemp2[7];

dtostrf(h, 6, 2, humidityTemp);

dtostrf(h2, 6, 2, humidityTemp2);

// Publishes Temperature and Humidity values

client.publish("room/temperature", temperatureTemp);

client.publish("room/temperature2", temperatureTemp2);

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

client.publish("room/humidity2", humidityTemp2);

Serial.print("Humidity: ");

Serial.print(h);

Serial.print(" %\t Temperature: ");

Serial.print(t);

Serial.print(" \*C ");

Serial.print(f);

Serial.print(" \*F\t Heat index: ");

Serial.print(hic);

Serial.println(" \*C ");

Serial.print("Humidity2: ");

Serial.print(h2);

Serial.print(" %\t Temperature2: ");

Serial.print(t2);

Serial.print(" \*C ");

Serial.print(f2);

Serial.print(" \*F\t Heat index: ");

Serial.print(hic2);

Serial.println(" \*C ");

// Serial.print(hif);

// Serial.println(" \*F");

if (t > t2 || t2 > t){ // Comparing the difference in temperature and powering A/C when difference is more than 2 degrees, also publish the LED again

difference = t - t2;

if (difference > 2) {

Serial.println("A/C ON...");

digitalWrite(lamp, HIGH);

client.publish("room/lamp3", "on");

} else if (difference < -2) {

Serial.println("A/C ON...");

digitalWrite(lamp, HIGH);

client.publish("room/lamp3", "on");

} else {

Serial.println("Balanced Temperature... A/C OFF");

digitalWrite(lamp, LOW);

client.publish("room/lamp3", "off"); //turn off light

}

}

}

delay (300);

readsuccess = getid();

if(readsuccess){

int match=0;

//this is the part where compare the current tag with pre defined tags

for(int i=0;i<N;i++){

Serial.print("Testing Against Authorised card no: ");

Serial.println(i+1);

if(!memcmp(readcard,defcard[i],4)){

match++;

uidnum=i; // shows the sequence number of the readed card based on defined IDs list after comparing it with this list at beggining.

}

}

if(match){

Serial.println("CARD AUTHORISED"); // if readed card matchs one of the defined cards, then it is an authorized card/tag

switch(uidnum){

case(0):Serial.println("Ahmed Mohamed B7C1590B");client.publish("room/rfid","Ahmed B7C1590B");break; // in case uidnum=0, then will print and publich the UID at index 0 (1st UID) on the pre-defined list "defcard"

case(1):Serial.println("Mohamed Ismail 21637563");client.publish("room/rfid","Mohamed 21637563");break; // in case uidnum=1, then will print and publich the UID at index 1 (2nd UID) on the pre-defined list "defcard"

// more cases "for example: case(2)" should be added here as well if new UIDs are added in the UID pre-defined list "defcard"

}

}

else {

Serial.println("CARD NOT Authorised"); // if readed card DO NOT match any of the defined cards, then print NOT authorized card

client.publish("room/rfid","Not authorized"); // publish card not authorized

}

}

}