LAMPIRAN

```
#include <DHT.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <analogWrite.h>

#define DHTTYPE DHT11
uint8_t pinDHT = 33;
DHT dht(pinDHT, DHTTYPE);

#define moistPin A0

int relay = 5;

//WiFi
const char* ssid = "Redmi 9";
const char* pass = "01012020";
//MQTT
const char* mqtt_server = "44.195.141.13"; //IP of the MQTT broker
const char* idTopic = "iot/mac";
const char* macTopic = "sawi/mac";
const char* humTopic = "sawi/humidity";
const char* temTopic = "sawi/temperature";
const char* moisTopic = "sawi/moisture";
const char* maTopic = "sawi/iot/mac";
const char* hTopic = "sawi/iot/humidity";
const char* tTopic = "sawi/iot/temperature";
const char* mTopic = "sawi/iot/moisture";
const char* mqtt_username = "frtris";
const char* mqtt_password = "123456";
const char* clientID = "client_sawi";
const char* wTopic = "esp/#";

//Initialise the WiFi and MQTT Client objects
WiFiClient wifiClient;

//1883 is the listener port of the Broker
PubSubClient client(wifiClient);
long lastMsg = 0;
int value = 0;
String mAc = WiFi.macAddress();
String conn;
String dis;
String konek;
String macaddress;
String hasilmac;
String manual;
String pesan;
int hasil_fuz;
int hasil;

unsigned long startMillis;
unsigned long currentMillis;
```

```
byte derajat[] = {
  B01110,
  B01010,
  B01010,
  B01110,
  B00000,
  B00000,
  B00000,
  B00000
};

//

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.print(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi Connected");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT Connection...");
    //Attempt to connect
    if (client.connect(clientID)) {
      Serial.println("connected");
      //      Once connected, publish an announcement
      if (client.publish(idTopic, String(mAc).c_str())) {
        Serial.println("MacAddress sent!");
      }
      //  Again,  client.publish  will  return  a  boolean  value
depending on whether it succeded or not.
      // If the message failed to send, we will try again, as the
connection may have broken.
      else {
        Serial.println("MacAddress failed to send. Reconnecting
to MQTT Broker and trying again");
        client.connect(clientID);
        delay(10);  //  This  delay  ensures  that  client.publish
doesn't clash with the client.connect call
        client.publish(idTopic, String(mAc).c_str());
      }
      client.subscribe(wTopic);
    } else {
      Serial.print("Failed, rc= ");
      Serial.println(client.state());
      Serial.println("Try Again in 2 seconds");
      delay(2000);
    }
  }
}

void callback(char* topic, byte* payload, unsigned int length) {
  String top = (String)topic;
  String message;
  String mess;
  String maa;
  String disc;
  int sta;
  int s;
  int st;

  Serial.print("Message arrived in topic: ");
  if (top.equals("esp/manual")) {
    manual = "";
    for (int i = 0; i < length; i++) {
      manual = (char)payload[i];
    }
    pesan = manual;
  }

  if (top.equals("esp/device")) {
    maa = "";
    for (int i = 0; i < length; i++) {
      maa = maa + (char)payload[i];
    }
    if (maa.equals(mAc)) {

      konek = maa;
    }
  }

  if (s == 1) {

    Serial.println(konek);
    if (manual.equals("1")) {
      while (manual.equals("1")) {

      }
      Serial.println("hidup");

    } if (manual.equals("0")) {
      Serial.println("mati");
    }
  }
  if (s == 0) {
    digitalWrite(relay, LOW);
  }

  if (top.equals("esp/mac")) {
    message = "";
    for (int i = 0; i < length; i++) {
      message = message + (char)payload[i];
    }
    if (message.equals(mAc)) {
      macaddress = message;
      sta = 1;
    }
  }

  if (top.equals("esp/disconnect")) {
    message = "";
    for (int i = 0; i < length; i++) {
      message = (char)payload[i];
    }
    Serial.println(message);
    dis = message;
  }

  if (top.equals("esp/connect")) {
    message = "";
    for (int i = 0; i < length; i++) {
      message = (char)payload[i];
    }
    mess = message;
  }

  if (top.equals("esp/dis")) {
    maa = "";
    for (int i = 0; i < length; i++) {
      maa = maa + (char)payload[i];
    }
    if (maa.equals(mAc)) {
      disc = maa;
      Serial.println(disc);
      sta = 0;
    } else {
      sta == 1;
    }
  }

  if (sta == 1) {
    if (macaddress.equals(mAc)) {
      Serial.println(macaddress);
      Serial.println(mess);
      conn = mess;
      if (mess.equals("")) {
        conn = 1;
      }
    }
  }
  if (sta == 0) {
    if (disc.equals(mAc)) {
      Serial.println("dis: " + dis);
      Serial.println(disc);
      conn = dis;
    }
  }

  if (top.equals("esp/hasil")) {
    message = "";
    Serial.println(top);
    for (int i = 0; i < length; i++) {
      message = message + (char)payload[i];
    }
    hasil = message.toInt();
  }

  if (top.equals("esp/hasilmac")) {
    message = "";
    Serial.println(top);
    for (int i = 0; i < length; i++) {
      message = message + (char)payload[i];
    }
    if (message.equals(mAc)) {
      st = 1;
      hasilmac = message;
    } else {
      st = 0;
    }
  }

  if (st == 1) {
    if (hasilmac.equals(mAc)) {
      Serial.println(hasilmac);
      hasil_fuz = hasil;
      Serial.println(hasil_fuz);

      for (int i = 0; hasil_fuz * 1000 > i; i++) {
        Serial.println(i);
        digitalWrite(relay, HIGH);
      }
      digitalWrite(relay, LOW);
    }
  }
}


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  pinMode(pinDHT, INPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
  dht.begin();

  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  Serial.print("ESP32 Board MAC Address:  ");
  Serial.println(WiFi.macAddress());

  digitalWrite(relay, LOW);
  delay(2000);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  int h = dht.readHumidity();
  int t = dht.readTemperature();
  float mois = 100.00 - ((analogRead(moistPin) / 4095.00) * 
100.00);

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println("C");
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.println("%");
  Serial.print("Moisture: ");
  Serial.print(mois);
  Serial.println("%");

  if (konek.equals(mAc)) {
    if (pesan.equals("1")) {
      conn = "0";
      Serial.println(pesan);
      digitalWrite(relay, HIGH);
      Serial.println("hidup");
      //      delay(1000);
    }
    if (pesan.equals("0")) {
      delay(2000);
      Serial.println(pesan);
      digitalWrite(relay, LOW);
      Serial.println("mati");
      conn = "1";
    }
  }

  //  if(message.equals(mAc)){
  if (conn.equals("1")) {
    if (t > 40 && h > 40) {
      Serial.println("Tidak publish");
    } else {
      if (client.publish(maTopic, String(mAc).c_str())) {
        Serial.println("MacAddress sent!");
      }
      // Again, client.publish will return a boolean value 
depending on whether it succeded or not.
      // If the message failed to send, we will try again, as the 
connection may have broken.
      else {
        Serial.println("MacAddress failed to send. Reconnecting 
to MQTT Broker and trying again");
        client.connect(clientID);
        delay(10); // This delay ensures that client.publish 
doesn't clash with the client.connect call
        client.publish(maTopic, String(mAc).c_str());
      }

      if (client.publish(tTopic, String(t).c_str())) {
        Serial.println("Temperature sent!");
      }
      // Again, client.publish will return a boolean value 
depending on whether it succeded or not.
```

```
      // If the message failed to send, we will try again, as the
connection may have broken.
      else {
        Serial.println("Temperature failed to send. Reconnecting
to MQTT Broker and trying again");
        client.connect(clientID);
        delay(10); // This delay ensures that client.publish
doesn't clash with the client.connect call
        client.publish(tTopic, String(t).c_str());
      }

      // PUBLISH to the MQTT Broker (topic = Humidity, defined at
the beginning)
      if (client.publish(hTopic, String(h).c_str())) {
        Serial.println("Humidity sent!");
      }
      // Again, client.publish will return a boolean value
depending on whether it succeded or not.
      // If the message failed to send, we will try again, as the
connection may have broken.
      else {
        Serial.println("Humidity failed to send. Reconnecting to
MQTT Broker and trying again");
        client.connect(clientID);
        delay(10); // This delay ensures that client.publish
doesn't clash with the client.connect call
        client.publish(hTopic, String(h).c_str());
      }

      // PUBLISH to the MQTT Broker (topic = Kelembaban Tanah,
defined at the beginning)
      if (client.publish(mTopic, String(mois).c_str())) {
        Serial.println("Moisture sent!");
      }
      // Again, client.publish will return a boolean value
depending on whether it succeded or not.
      // If the message failed to send, we will try again, as the
connection may have broken.
      else {
        Serial.println("Moisture failed to send. Reconnecting to
MQTT Broker and trying again");
        client.connect(clientID);
        delay(10); // This delay ensures that client.publish
doesn't clash with the client.connect call
        client.publish(mTopic, String(t).c_str());
      }
      delay(2000);
    }

  } if (conn.equals("0")) {
    Serial.println("Stop to Publish Data...");
  }

  long now = millis();
  if (now - lastMsg > 1000*20) {
    lastMsg = now;
    ++value;
    if (client.publish(macTopic, String(mAc).c_str())) {
      Serial.println("MacAddress sent!");
    }
    // Again, client.publish will return a boolean value depending
on whether it succeded or not.
    // If the message failed to send, we will try again, as the
connection may have broken.
    else {
      Serial.println("Temperature failed to send. Reconnecting to
MQTT Broker and trying again");
      client.connect(clientID);
      delay(10);  //  This  delay  ensures  that  client.publish
doesn't clash with the client.connect call
      client.publish(macTopic, String(mAc).c_str());
    }

    if (client.publish(temTopic, String(t).c_str())) {
      Serial.println("Temperature sent!");
    }
    // Again, client.publish will return a boolean value depending
on whether it succeded or not.
    // If the message failed to send, we will try again, as the
connection may have broken.
    else {
      Serial.println("Temperature failed to send. Reconnecting to
MQTT Broker and trying again");
      client.connect(clientID);
      delay(10);  //  This  delay  ensures  that  client.publish
doesn't clash with the client.connect call
      client.publish(temTopic, String(t).c_str());
    }

    // PUBLISH to the MQTT Broker (topic = Humidity, defined at
the beginning)
    if (client.publish(humTopic, String(h).c_str())) {
      Serial.println("Humidity sent!");
    }
    // Again, client.publish will return a boolean value depending
on whether it succeded or not.
    // If the message failed to send, we will try again, as the
connection may have broken.
    else {
      Serial.println("Humidity  failed  to  send.  Reconnecting  to
MQTT Broker and trying again");
      client.connect(clientID);
      delay(10);  //  This  delay  ensures  that  client.publish
doesn't clash with the client.connect call
      client.publish(humTopic, String(h).c_str());
    }

    //  PUBLISH  to  the  MQTT  Broker  (topic  =  Kelembaban  Tanah,
defined at the beginning)
    if (client.publish(moisTopic, String(mois).c_str())) {
      Serial.println("Moisture sent!");
    }
    // Again, client.publish will return a boolean value depending
on whether it succeded or not.
    // If the message failed to send, we will try again, as the
connection may have broken.
```

```
    else {
      Serial.println("Moisture failed to send. Reconnecting to
MQTT Broker and trying again");
      client.connect(clientID);
      delay(10); // This delay ensures that client.publish
doesn't clash with the client.connect call
      client.publish(moisTopic, String(t).c_str());
    }
  }
}
```