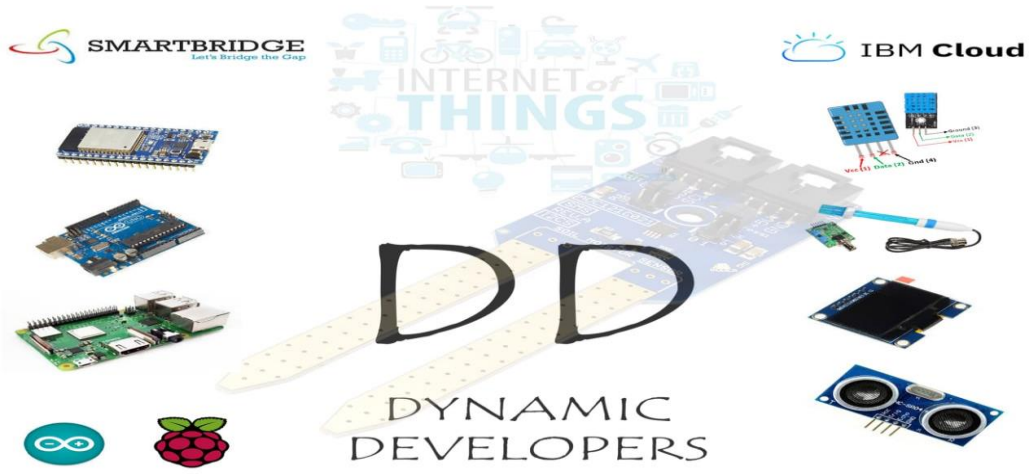
INTERNET OF THINGS (IOT)



G. ROHITH VISHAAL

RAHUL DOMALA

MD RIYAZ AHMED

ALETI YASHWANTH REDDY

UNDER THE GUIDANCE OF

DIVYA NEMURI

CONTENT

GREEN HOUSE MONITORING AND CONTROL

ABSTRACT:

The aim of this paper is to design a greenhouse monitoring system based on the Internet of things (Iot). A greenhouse is a covered area where plants grow and cultivate. It is also known as land of controlled crops and plants. There are some important parameters to be monitored inside the greenhouse are temperature, humidity, soil moisture, light dependent resistor (ldr), using DHT11, LDR, Soil moisture sensor. It will start monitoring when these sensors are connected. They are representing the technology solution to automate and improve the management of greenhouse. Internet of things (Iot) was developed for connecting a billion of devices into an internet. A huge amount of information is transferred between the electronic devices. It is a new way to interact between device and people. This shows that how the wireless communication has been for future vision in the monitoring system. Internet of things (Iot) will play a major role in day to day life in the future.

PROBLEM STATEMENT:

Using IOT optimizing greenhouse environment to grow plants under control of perfect climatic conditions for effective production.

COMPONENTS REQUIRED:

- Micro controllers
    1. Arduino Uno
    2. ESP-8266
- BASIC SHEILD
    1. Ldr (light dependent resistor)
    2. DHT11 (temperature and humidity)
- Soil moisture
- Transistor
- 9v Dc Motor
- Oled Display

MICRO-CONTROLLERS

ARDUINO UNO:

Arduino is an open source, computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world

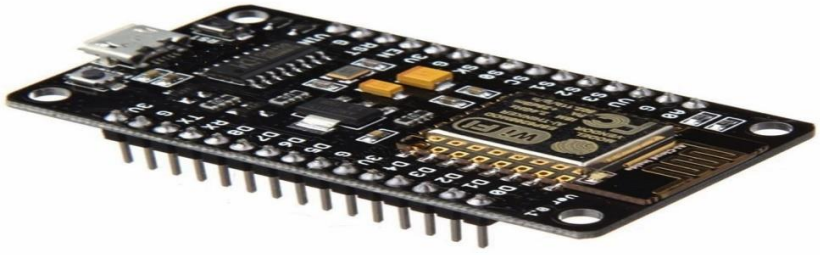
Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

- 3.3V (6) – Supply 3.3 output volt
- 5V (7) – Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

ESP8266:

ESP8266 is Wi-Fi enabled system on chip (SoC) module developed by Espressif system. It is mostly used for development of IoT (Internet of Things) embedded applications. It employs a 32-bit Ten silica microcontroller, standard digital interface, antenna switches

And power management modules into a small package.



SPECIFICATIONS:

- 2.4 GHz WI-FI(802.11 b/g/n, supporting WPA/WPA2)
- General purpose I/O(16 GPIO)
- Inter –integrated Circuit(I2C) serial Communication protocol
- Analog to Digital Conversion(10-bit ADC)
- Serial Peripheral Interface(SPI) serial communication protocol
- UART (on dedicated pins ,plus a transmit-only UART can be enabled on GPIO2)
- Pulse-width modulation (PWM)

- External QSIP lash up to 16 MB is supported(512 KB to 4MB typically included)

MEMORY:

1. 32 KB instruction RAM
2. 32 KB instruction cache RAM
3. 80 KB user-data RAM
4. 16 KB ETS system-data RAM

SENSORS

DHT11 SENSOR :

DHT11 is a brick of digital temperature and humidity sensor complex with a calibrated digital signal output. it is a single-bus operation, extremely small in size and low consumption enable it to be used in HVAC, auto motive, weather stations, dehumidifier and other applications.

WORKING PRINCIPLE OF DHT11:

Temperature:

The temperature is measured with the help of a NTC thermistor or negative temperature coefficient thermistor. These thermistors are usually made with semiconductors, ceramic and polymers. The resistance of the device is inversely proportional with temperature and follows a hyperbolic curve. Temperature using NTC often found out Steinhart Hart equation.

Humidity:

The humidity is sensed using a moisture dependent resistor. It has two electrodes and in between them there exist a moisture holding substrate which holds moisture. The

conductance and hence resistance changes with changing humidity.

SPECIFICATIONS :

PCB size	22.0mm X 20.5mm X 1.6mm
Working voltage	3.3 or 5V DC
Operating voltage	3.3 or 5V DC
Measurement range	20-95%RH；0-50℃
Resolution	8bit（temperature），8bit（humidity）
Compatible interfaces	2.54 3-pin interface and 4-pin Grove interface(1)

LDR(LIGHT DEPENDENT RESISTOR):

An LDR is a component that has a variable resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. They can be described by a variety of names from light dependent resistor, LDR, photoresistor, or even photo cell, photocell or photoconductor. It is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms.

WORKING PRINCIPLE OF LDR:

A **light dependent resistor** works on the principle of photo conductivity. Photo conductivity is an optical phenomenon.

When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These photons in the incident light should have energy greater than the band gap of the semiconductor material to make the electrons jump from the valence band to the conduction band.

Hence when light having enough energy strikes on the device, more and more electrons are excited to the conduction band which results in large number of charge carriers. The result of this process is more and more current starts flowing through the device when the circuit is closed and hence it is said that the resistance of the device has been decreased. This is the most common **working principle of LDR**.

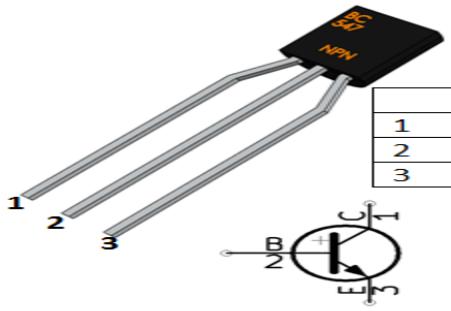
SPECIFICATIONS :

Max power dissipation	This is the maximum power the device is able to dissipate within a given temperature range. Derating may be applicable above a certain temperature.
Maximum operating voltage	Particularly as the device is semiconductor based, the maximum operating voltage must be observed. This is typically specified at 0 lux, i.e. darkness.
Peak wavelength	This photoresistor specification details the wavelength of maximum sensitivity. Curves may be provided for the overall response in some instances. The wavelength is specified in nm.
Dark resistance	This values will be given for the photoresistor. These may be specified after a given time because it takes a

	while for the resistance to fall as the charge recombine carrier photo resistors are noted for their slow response times.

TRANSISTOR:

Transistor is a semiconductor device used to amplify or switch electronic signals and electrical power. It is composed of semiconductor material usually with at least three terminals for connection to an external circuit. A voltage or current applied to one pair of the transistor's terminals controls the current through another pair of terminals.



BC-547	
1	Collector
2	Base
3	Emitter

BC547 WORKING PRINCIPLE:

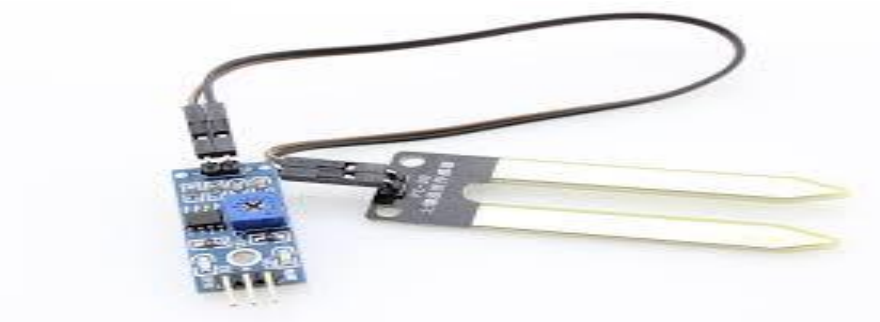
When the input voltage is applied at its terminal, some amount of current starts to flow from base to the emitter and controls the current at collector. The voltage between the base and the emitter (VBE), is negative at the emitter and positive at the base terminal for its NPN construction.

SPECIFICATIONS :

- Collector-base voltage (VCB) is the maximum voltage that can be applied across the base and collector of a transistor.
- Collector-emitter voltage (VCEO) is the maximum that can be applied across the collector and emitter of a collector.
- Collector saturation voltage is the collector-emitter voltage of a transistor that is fully conducting.
- Maximum collector dissipation (PC) is the maximum value of power dissipated in collector of a transistor.
- Collector current is the maximum value of collector current that a transistor can handle safely.
- Forward current gain is the maximum value of forward current of a transistor in CE configuration.

SOIL MOSITURE:

The Soil Moisture Sensor uses capacitance to measure the water content of soil (by measuring the dielectric permittivity of the soil, which is a function of the water content). Simply insert this rugged sensor into the soil to be tested, and the volumetric water content of the soil is reported in percent.



WORKING PRINCIPLE OF MOISTURE SENSOR:

The Soil Moisture Sensor uses capacitance to measure dielectric permittivity of the surrounding medium. In soil, dielectric permittivity is a function of the water content. The sensor creates a voltage proportional to the dielectric permittivity, and therefore the water content of the soil.

The Soil Moisture Sensor is used to measure the loss of moisture over time due to evaporation and plant, evaluate optimum soil moisture contents for various species of plants, monitor soil moisture content to control irrigation in greenhouses and enhance yield of the plant.

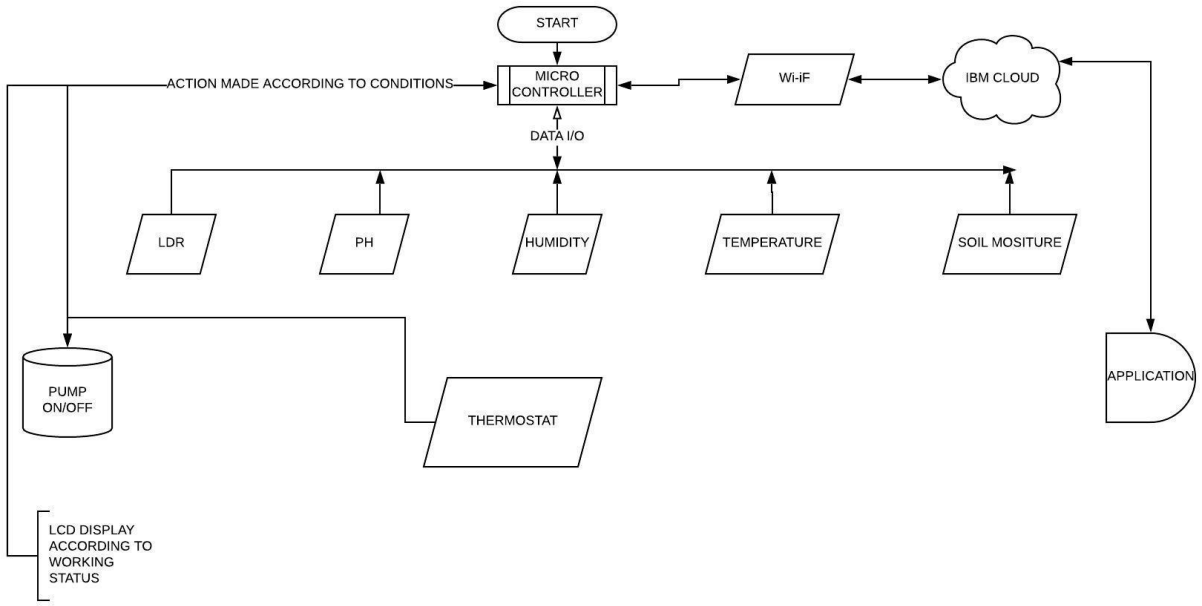
SPECIFICATIONS :

- Range: 0 to 45% volumetric water content in soil (capable of 0 to 100% VWC with alternate calibration)
- Accuracy: ±4% typical
- Typical Resolution: 0.1%
- Power: 3 mA @ 5VDC
- Operating temperature: −40°C to +60°C
- Dimensions: 8.9 cm × 1.8 cm × 0.7 cm (active sensor length 5 cm)

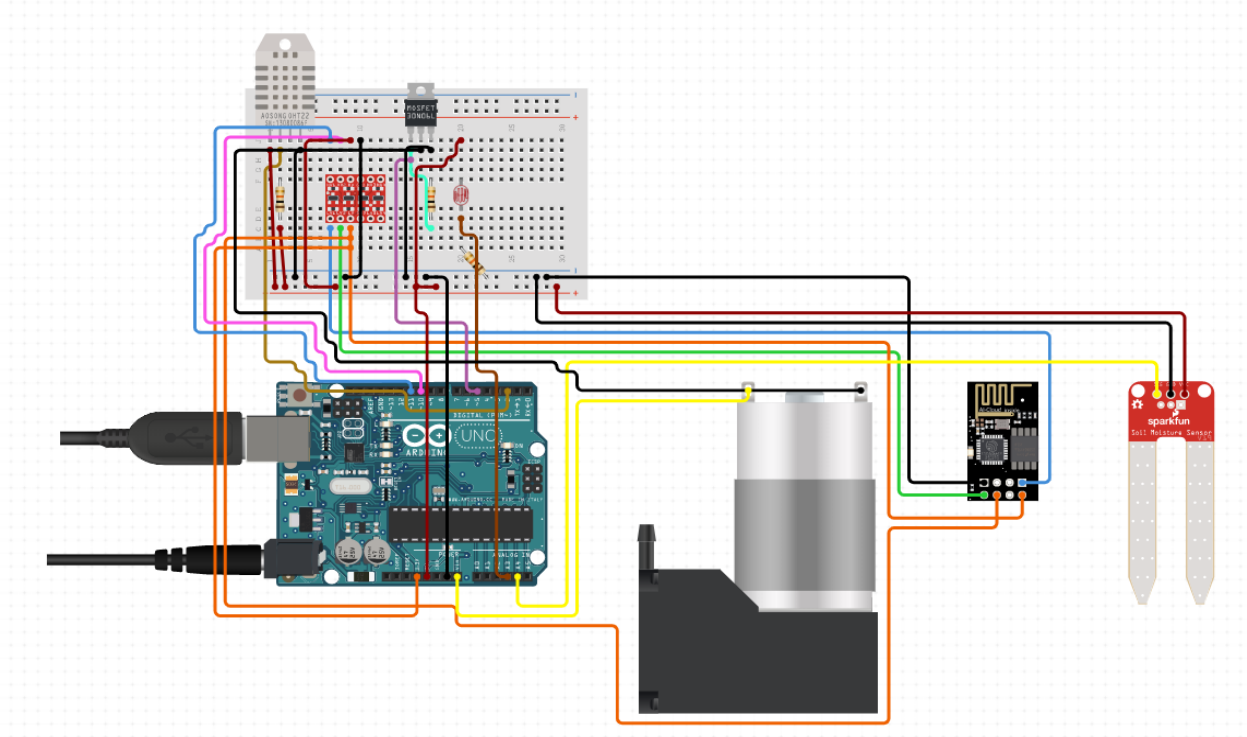
WORKING PROCESS :

- Initially we read the following parameters using respective sensors which are linked to microcontrollers.
- (Arduino Uno & NODEMCU)
    - Humidity
    - Temperature
    - Soil moisture
    - Light intensity
- Now the values related to the above parameters are uploaded to cloud where these values are displayed by means of the Web User interface and API.
- In the UI, control keys are being provided, with these keys the user can switch On and Off motor in accordance with the recorded soil moisture values.
- An automatic On & Off control of the motor is also provided for the user convenience.

FLOWCHART:



CRICUIT:



ARDUINO IDE:

Programming software:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

CODING:

```
//--------------------OLED--------------------------------//
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
#define SSD1306_LCDHEIGHT 64
#define OLED_ADDR   0x3C
Adafruit_SSD1306 display(-1);
#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
//--------------------WIFI--------------------------------//
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
//--------------------ARDUINO TRANSFER--------------------
----------------//
#include<SoftwareSerial.h>
SoftwareSerial myserial(D7,D8);
//----------------------------------------------------------//


void callback(char* topic, byte* payload, unsigned int
payloadLength);
```

```
// CHANGE TO YOUR WIFI CREDENTIALS
const char* ssid = "E=MC^2";
const char* password = "Incorrect@2019";

// CHANGE TO YOUR DEVICE CREDENTIALS AS PER IN IBM BLUMIX
#define ORG "i4lugi"
#define DEVICE_TYPE "NODE_MCU"
#define DEVICE_ID "1024"
#define TOKEN "nodemcu007" //  Authentication Token OF THE DEVICE

//----------------LDR AND MOTOR----------------------//
#define ldr A0
#define motor D6
//-----------------DHT--------------------------//
#include "DHT.h"
#define DHTPIN D3
#define DHTTYPE DHT11
DHT dht (DHTPIN, DHTTYPE);
float temperature;
```

```
int humidity;
//-------------------------------------------------------//
String data3;
String data="";
String data1;
int val;
String dataval;


//-------- Customise the above values --------
const char publishTopic[] = "iot-2/evt/Data/fmt/json";
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/cmd/data/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;



WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);
```

```
int publishInterval = 5000; // 30 seconds
long lastPublishMillis;
void publishData();

void setup() {
 pinMode(D3,OUTPUT);
 pinMode(D6,OUTPUT);
 Serial.begin(115200);
 myserial.begin(115200);
 Serial.println();
 dht.begin();
 wifiConnect();
 mqttConnect();
//-----------------------OLED-----------------------//
display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,10);
//--------------------------------------------------//
```

```
}

void loop() {
if(myserial.available()){
myserial.find('#');
data1=myserial.readStringUntil('@');
val=String(data1).toInt();
Serial.println(val);
delay(500);}
if (millis() - lastPublishMillis > publishInterval)
{
publishData(data1);
lastPublishMillis = millis();
}
if (!client.loop()) {
mqttConnect();
}
 if(dataval=="true"){
  if(val<=65){
   digitalWrite(D6,HIGH);
Serial.println(".....MOTOR is ON.....");
```

```
display.clearDisplay();
display.setCursor(0,5);
display.print("Soil moisture low\n");
display.setCursor(0,20);
display.print("Motor on");
display.display();
delay(1000);}

  if(val>65){
    digitalWrite(D6,LOW);
Serial.println(".....MOTOR is OFF.....");
display.clearDisplay();
display.setCursor(0,5);
display.print("Soil moisture normal\n");
display.setCursor(0,20);
display.print("Motor off");
display.display();
  }

}
}
```

```
void wifiConnect() {
  Serial.print("Connecting to "); Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("nWiFi connected, IP address: ");
Serial.println(WiFi.localIP());
}

void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to ");
Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
    initManagedDevice();
```

```
    Serial.println();
  }
}

void initManagedDevice() {
  if (client.subscribe(topic)) {
   // Serial.println(client.subscribe(topic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* topic, byte* payload, unsigned int
payloadLength) {
Serial.print("callback invoked for topic: ");
Serial.println(topic);
for (int i = 0; i < payloadLength; i++) {
//Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
Serial.print("data: "+ data3);
```

```
control_func();
data3 = "";
}

void control_func()
{
dataval=data3;
if(data3 == "MOTORON")
{
digitalWrite(D6,HIGH);
Serial.println(".....MOTOR is ON.....");
}
else if(data3=="MOTOROFF")
{
digitalWrite(D6,LOW);
Serial.println(".....MOTOR is OFF.....");
}
if(data3=="false"){
  digitalWrite(D6,LOW);
  display.clearDisplay();
display.setCursor(0,10);
```

```
display.print("water automation:off");
display.display();
delay(1000);
}

}



void publishData(String data1)
{
humidity = dht.readHumidity();
temperature = dht.readTemperature();
float ldr_val=analogRead(ldr);

if (isnan(humidity) || isnan(temperature)) {
Serial.println("Failed to read from DHT sensor!");
return;
}
display.clearDisplay();
display.setCursor(0,10);
```

```
display.print("humidity:"+String(humidity)+"%"+"\n");
display.display();
delay(1000);

display.clearDisplay();
display.setCursor(0,10);
display.print("temperature:"+String(temperature)+"C"+"\n");
display.display();
delay(1000);

display.clearDisplay();
display.setCursor(0,10);
display.print("LDR value:"+String(ldr_val)+"\n");
display.display();
delay(1000);

display.clearDisplay();
display.setCursor(0,10);
display.print("moisture value:"+data1+"\n");
display.display();
delay(1000);
```

```
  String payload = "{\"d\":{\"temperature\":";
  payload += temperature;
  payload += ","\"humidity\":";
  payload +=  humidity;
  payload += ","\"LDR\":";
  payload +=  ldr_val;
  payload += ","\"soilmoisture\":";
  payload +=data1;
  payload += "}}";


  Serial.print("\n");
  Serial.print("Sending payload: "); Serial.println(payload);

  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  } else {
    Serial.println("Publish FAILED");
  }
}
```