

ASSIGNMENT-4

Date	21 october 2022
Student Name	P.Gunasundari
Student Roll Number	110519106003
Maximum Marks	2 Marks

CODEING:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int
payloadLength);
#define ORG "6fmt6h"
#define DEVICE_TYPE "Gunasundari123"
#define DEVICE_ID "6003"
#define TOKEN "fzr?Kh8RVYGPkwGnW4"
String data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/Gunasundari123/fmt/json";
char subscribeTopic[]="iot-2/cmd/test/fmt/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);

#define ECHO_PIN 12
#define TRIG_PIN 13
#define led 14

void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    pinMode(led, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    wificonnect();
    mqttconnect();
}
float readDistanceCM() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
```

```

    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    int duration=random(1,200);
    //Serial.println(duration);
    //duration = pulseIn(ECHO_PIN, HIGH);
    return duration ;
    //Serial.println(duration);
}

void loop() {
    float distance = readDistanceCM();
    //Serial.println(distance);

    bool isNearby = distance < 100;
    digitalWrite(led, isNearby);

    Serial.print("Measured distance: ");
    Serial.println(distance);
    if(distance<100){
        PublishData2(distance);
    }
    else{
        PublishData1(distance);

    }
    //PublishData(distance);
    delay(1000);
    if(!client.loop()){
        mqttconnect();
    }

    //delay(200);
}
void PublishData1(float dist){
    mqttconnect();
    String payload= "{\"distance\":\"";
    payload += dist;
    payload+="}";

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

    if(client.publish(publishTopic, (char*)payload.c_str())){
        Serial.println("publish ok");
    }
    else{
        Serial.println("publish failed");
    }
}

```

```

void PublishData2(float dist){
    mqttconnect();
    String payload= "{\"ALERT\":";
    payload += dist;
    payload+="}";

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

    if(client.publish(publishTopic,(char*)payload.c_str())){
        Serial.println("publish ok");
    }
    else{
        Serial.println("publish failed");
    }
}

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

void wificonnect(){
    Serial.println();
    Serial.print("Connecting to");

    WiFi.begin("Wokwi-GUEST","",6);
    while(WiFi.status()!=WL_CONNECTED){
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WIFI CONNECTED");
    Serial.println("IP address:");
    Serial.println(WiFi.localIP());
}

void initManagedDevice(){
    if(client.subscribe(subscribeTopic)){
        Serial.println((subscribeTopic));
        Serial.println("subscribe to cmd ok");
    }
    else{
        Serial.println("subscribe to cmd failed");
    }
}

```

```

    }
}

void callback(char* subscribeTopic, byte* payload, unsigned int
payloadLength){
    Serial.print("callback invoked for topic:");
    Serial.println(subscribeTopic);
    for(int i=0; i<payloadLength; i++){
        data3 += (char)payload[i];
    }
    Serial.println("data:" + data3);
    if(data3=="lighton"){
        Serial.println(data3);
        digitalWrite(led,HIGH);
    }else{
        Serial.println(data3);
        digitalWrite(led,LOW);
    }
    data3="";
}
}

```

OUTPUT:

Less than 100cm-LED Glow:

The screenshot displays the WOKWI simulation interface. On the left, the sketch code is visible, showing the setup of an ESP32 with an HC-SR04 sensor and an LED. The code includes a callback function that triggers the LED to turn on when the topic 'lighton' is received. On the right, the simulation shows the physical components: the ESP32, the HC-SR04 sensor, and the LED. The LED is shown as glowing red, indicating it is turned on. The console output at the bottom shows the sensor reading a distance of 90.00 cm and sending a payload {'ALERT': 90.00}.

```

Measured distance: 90.00
Sending payload: {"ALERT": 90.00}
publish ok
Measured distance: 86.00
Sending payload: {"distance": 86.00}
publish ok

```

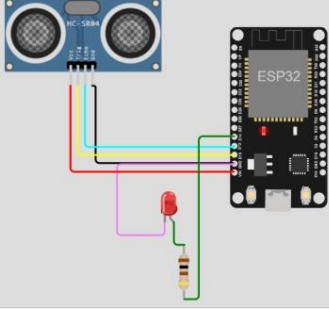
More than 100cm-LED Doesn't Glow:

WOKWI SAVE SHARE sketch.ino copy Docs SIGN UP

sketch.ino • diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "6fmc6h"
5 #define DEVICE_TYPE "Gunasundari123"
6 #define DEVICE_ID "6003"
7 #define TOKEN "#zn?Kh8RVYGPKwGnM4"
8 String data3;
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/Gunasundari123/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
16
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19
20 #define ECHO_PIN 12
21 #define TRIG_PIN 13
22 #define led 14
23
24 void setup() {
25   // put your setup code here, to run once:
26   Serial.begin(115200);
27   pinMode(led, OUTPUT);
28   pinMode(TRIG_PIN, OUTPUT);
29   pinMode(ECHO_PIN, INPUT);
30   wifiConnect();
31   mqttConnect();
32 }
33 float readDistanceCM() {
34   digitalWrite(TRIG_PIN, LOW);
35   delayMicroseconds(2);
36   digitalWrite(TRIG_PIN, HIGH);
```

Simulation



publish ok
Measured distance: 160.00
Sending payload: {"distance":160.00}
publish ok
Measured distance: 168.00
Sending payload: {"distance":168.00}
publish ok

```
Measured distance: 179.00
Sending payload: {"ALERT":179.00}
publish ok
Measured distance: 128.00
Sending payload: {"distance":128.00}
publish ok
```