

IBM assignment-4

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribtopic,byte* payload, unsigned int
payloadLength);
#define ORG "fdd82r"
#define DEVICE_TYPE "Pi"
#define DEVICE_ID "123"
#define TOKEN "12345678"
String data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/distance/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 2
#define TRIG_PIN 4
#define led 5

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(2000);
}
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"){

```

```

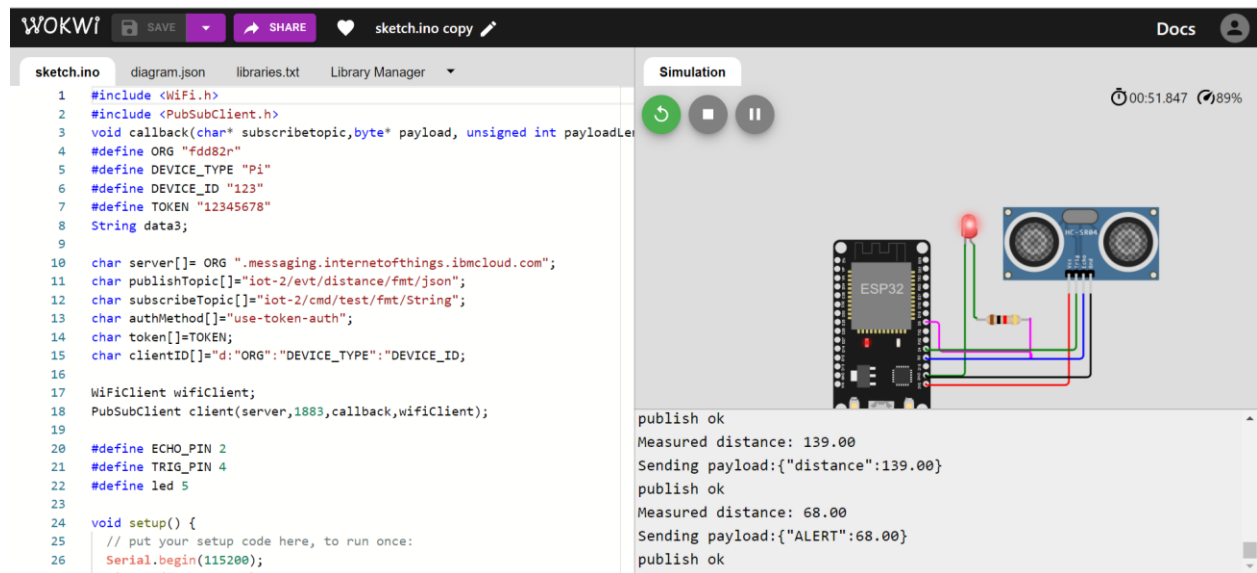
    Serial.println(data3);
    digitalWrite(led,HIGH);
  }else{
    Serial.println(data3);
    digitalWrite(led,LOW);
  }
  data3="";
}

```

WOKWI project link:

<https://wokwi.com/projects/347319793989190228>

ALERT CASE:



The screenshot shows the WOKWI IDE interface. On the left, the 'sketch.ino' file is open, displaying the following code:

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload,unsigned int payloadLen)
4 #define ORG "fdd82n"
5 #define DEVICE_TYPE "P1"
6 #define DEVICE_ID "123"
7 #define TOKEN "12345678"
8 String data3;
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/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 2
21 #define TRIG_PIN 4
22 #define led 5
23
24 void setup() {
25   // put your setup code here, to run once:
26   Serial.begin(115200);
27   pinMode(led, OUTPUT);

```

On the right, the 'Simulation' window shows a visual representation of the ESP32 board connected to an HC-SR04 ultrasonic sensor and an LED. Below the simulation, the console output shows the following sequence of events:

```

publish ok
Measured distance: 139.00
Sending payload:{"distance":139.00}
publish ok
Measured distance: 68.00
Sending payload:{"ALERT":68.00}
publish ok

```

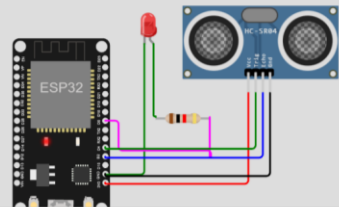
NORMAL CASE:

WOKWI SAVE SHARE sketch.ino copy Docs

sketch.ino diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload,unsigned int payloadLen)
4 #define ORG "fdd82r"
5 #define DEVICE_TYPE "Pi"
6 #define DEVICE_ID "123"
7 #define TOKEN "12345678"
8 String data3;
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/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 2
21 #define TRIG_PIN 4
22 #define led 5
23
24 void setup() {
25   // put your setup code here, to run once:
26   Serial.begin(115200);
27 }
```

Simulation 01:07.047



Measured distance: 152.00
Sending payload:{"distance":152.00}
publish ok
Measured distance: 168.00
Sending payload:{"distance":168.00}
publish ok

CLOUD STORAGE:

BrowseActionDevice TypesInterfaces

Add Device

Identity

Device Information

Recent Events

State

Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
distance	{"ALERT":95}	json	a few seconds ago
distance	{"ALERT":14}	json	a few seconds ago
distance	{"distance":193}	json	a few seconds ago
distance	{"ALERT":79}	json	a few seconds ago
distance	{"distance":186}	json	a few seconds ago

Items per page 50 | 1–1 of 1 item1 of 1 page < 1 >