

Assignment - 4

| | |
|--------------|--|
| Student Name | Sujitha R |
| Team ID | PNT2022TMID53651 |
| Project name | Project - Industry Specific Intelligent Fire Management system |

QUESTION:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to IBM cloud and display in device recent events. Upload document with wokwi share link and images of IBM cloud.

SOLUTION:

```
#include <WiFi.h>//library for wifi
```

```
#include <PubSubClient.h>//library for MQTT
```

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

```
// -----credentials of IBM Accounts-----
```

```
#define ORG "j0mda0"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "Sujitha"//Device type mentioned in ibm watson IOT Platform
```

```
#define DEVICE_ID "2301"//Device ID mentioned in ibm watson IOT Platform
```

```
#define TOKEN "Sujitha@2301" //Token
```

```
String data3;
```

```
float distance;
```

```
#define sound_speed 0.034
```

```
int trigpin=18;
```

```
int echopin=19;
```

```
int led=5;
```

```
int LED=9;
```

```
long duration;
```

```
String message;
```

//----- Customise the above values -----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name

char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send

char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING

char authMethod[] = "use-token-auth";// authentication method

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id

//-----

WiFiClient wifiClient; // creating the instance for wificlient

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server id,portand wificredential

void setup()// configureing the ESP32

```
{  
  Serial.begin(115200);  
  pinMode(trigpin,OUTPUT);  
  pinMode(echopin,INPUT);  
  pinMode(led,OUTPUT);  
  delay(10);  
  Serial.println();  
  wificonnect();  
  mqttconnect();  
}
```

void loop()// Recursive Function

```
{  
  digitalWrite(trigpin,LOW);  
  digitalWrite(trigpin,HIGH);
```

```

    delay(1000);
    digitalWrite(trigpin,LOW);
    duration=pulseIn(echopin,HIGH);
    distance=duration*sound_speed/2;

    Serial.println("distance"+String(distance)+"cm");

    if(distance<100)
    {message="Alert";
    digitalWrite(led,HIGH);}
    else
    {message="No problem";
    digitalWrite(led,LOW);}
    delay(1000);
    PublishData(distance,message);
    if (!client.loop()) {
        mqttconnect();
    }
}

/* .....retrieving to Cloud..... */

void PublishData(float d,String a) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */
    String payload = "{\"distance\":";
    payload += d;
    payload += "}";

```

```
payload += "," "{\"message\":";
payload += a;
payload += "}";
```

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

```
if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print
    publish ok in Serial monitor or else it will print publish failed
} else {
    Serial.println("Publish failed");
}
```

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

```
    initManagedDevice();
    Serial.println();
}
```

```
}
void wificonnect() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");
```

```

WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
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++) {
        //Serial.print((char)payload[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 OUTPUT:

DISTANCE GREATER THAN 100:

The screenshot shows the Wokwi IDE interface. The left pane displays the sketch code, and the right pane shows the simulation of an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The code defines variables for server, topic, device type, device ID, token, and pin configurations. It includes a callback function and a loop that publishes data to the MQTT server. The simulation window shows the sensor's output, indicating a distance of 399.94cm and 399.92cm, both greater than 100cm.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4
5 void callback(char* topic, byte* payload, unsigned int payloadlength);
6
7 // -----credentials of IBM Accounts-----
8 #define ORG "0jjs12" //IBM ORGANITION ID
9 #define DEVICE_TYPE "b1m3edevicetype" //Device type mentioned in ibm watson IOT Plat
10 #define DEVICE_ID "b1m3edevicid" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "_zLY3G?0s50?H5puUo" //Token
12
13 // #define ORG "j0mda0" //IBM ORGANITION ID
14 // #define DEVICE_TYPE "Sujitha" //Device type mentioned in ibm watson IOT Platform
15 // #define DEVICE_ID "2301" //Device ID mentioned in ibm watson IOT Platform
16 // #define TOKEN "Sujitha@2301" //Token
17 String data3;
18 float distance;
19 #define sound_speed 0.034
20 int trigpin=18;
21 int echopin=19;
22 int led=5;
23 int LED=9;
24 long duration;
25 String message;
26
27
28 //----- Customise the above values -----
29 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
30 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event per
31 char subscribTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type
32 char authMethod[] = "use-token-auth"; // authentication method
33 char token[] = TOKEN;
34 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
35
```

Simulation output:

```
Publish ok
distance399.94cm
Sending payload: {"distance":399.94},{"message":No problem}
Publish ok
distance399.92cm
Sending payload: {"distance":399.92},{"message":No problem}
Publish ok
```

DISTANCE LESSER THAN 100:

The screenshot shows the Wokwi IDE interface. The left pane displays the sketch code, and the right pane shows the simulation of an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The code defines variables for server, topic, device type, device ID, token, and pin configurations. It includes a callback function and a loop that publishes data to the MQTT server. The simulation window shows the sensor's output, indicating a distance of 73.93cm, which is less than 100cm. A dialog box for the Ultrasonic Distance Sensor is also visible, showing a distance of 74cm.

```
30 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event per
31 char subscribTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type
32 char authMethod[] = "use-token-auth"; // authentication method
33 char token[] = TOKEN;
34 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
35
36
37 //----- Customise the above values -----
38 WiFiClient wifiClient; // creating the instance for wifiClient
39 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined C
40 void setup() // configuring the ESP32
41 {
42   Serial.begin(115200);
43   pinMode(trigpin, OUTPUT);
44   pinMode(echopin, INPUT);
45   pinMode(led, OUTPUT);
46   delay(10);
47   Serial.println();
48   wifiConnect();
49   mqttConnect();
50 }
51
52 void loop() // Recursive Function
53 {
54   digitalWrite(trigpin, LOW);
55   digitalWrite(trigpin, HIGH);
56   delay(1000);
57   digitalWrite(trigpin, LOW);
58   duration = pulseIn(echopin, HIGH);
59   distance = duration * sound_speed / 2;
60
61   Serial.println("distance" + String(distance) + "cm");
62
63   if (distance < 100)
64   {
65     // Send alert message
66     String message = "Alert";
67     String payload = "{" + "distance": "73.93" + "," + "message": "Alert" + "}";
68     client.publish(publishTopic, payload);
69     delay(1000);
70   }
71 }
```

Simulation output:

```
Sending payload: {"distance":73.93},{"message":Alert}
Publish ok
distance73.93cm
Sending payload: {"distance":73.93},{"message":Alert}
Publish ok
distance73.93cm
```

DEVICE RECENT EVENTS IN IBM WATSON:

2301

Disconnected

Sujitha

Device

Nov 6, 2022 9:15 PM

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 |
|-------|--|--------|-------------------|
| Data | {\"d\":{\"distance\":308.02,\"message\":\"No problem\"}} | json | a few seconds ago |
| Data | {\"d\":{\"distance\":285.18,\"message\":\"No problem\"}} | json | a few seconds ago |
| Data | {\"d\":{\"distance\":157.66,\"message\":\"No problem\"}} | json | a few seconds ago |
| Data | {\"d\":{\"distance\":359.24,\"message\":\"No problem\"}} | json | a few seconds ago |
| Data | {\"d\":{\"distance\":138.31,\"message\":\"No problem\"}} | json | a few seconds ago |

Items per page: 50

1 of 1 page

WOWKI LINK:

<https://wokwi.com/projects/347594038348612180>