

**Assignment -4 Data**  
Publish to IOT Device

Assignment Date	27 October 2022
Student Name	MOHAMED RAZVI
Student Roll Number	110119104040
Maximum Marks	2 Marks

**Question-1:**

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less 100 cms send "alert" to ibm cloud and display in device recent events.

**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 "qxm592" //IBM ORGANITION ID
#define DEVICE_TYPE "weather_device" //Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "jwSiUN+qppnF1*xTRa" //Token
String data3;
float dist;

//----- 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

int LED = 4;
int trig = 5;
int echo = 18;
void setup()
{
  Serial.begin(115200);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
}
void loop()// Recursive Function
{

  digitalWrite(trig,LOW);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  float dur = pulseIn(echo,HIGH);
  float dist = (dur * 0.0343)/2;
  Serial.print ("Distancein cm");
  Serial.println(dist);

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

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

void PublishData(float dist) {
  mqttconnect();//function call for connecting to ibm
  /*
   creating the String in in form JSon to update the data to ibm cloud
  */
  String object;
  if (dist <100)

```

```

{
    digitalWrite(LED,HIGH);
    Serial.println("object is near");
    object = "Near";
}
else
{
    digitalWrite(LED,LOW);
    Serial.println("no object found");
    object = "No";
}

String payload = "{\"distance\": ";
payload += dist;
payload += ", \"object\": \"";
payload += object;
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=="Near")
    // {
    // Serial.println(data3);
    // digitalWrite(LED,HIGH);

    // }

    // else
    // {
    // Serial.println(data3);
    // digitalWrite(LED,LOW);

```

```
// }
data3="";

}
```

OUTPUT:

OBJECT NEAR BY DEVICE:

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribetopic, byte* payload, unsigned int payloadlength);
5
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "qxm592" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather_device" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "jwSiUWqppnF1xTRa" //Token
12 String data3;
13 float dist;
14
15 //----- Customise the above values -----
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishtopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform a
19 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND TO
20 char authmethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24 //-----
25
26 WiFiClient wificlient; // creating the instance for wificlient
27 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client
28
29
30 int LED = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()
34 {
35   Serial.begin(115200);
36   pinMode(trig, OUTPUT);
```

On the right, the 'Simulation' window shows an ESP32 microcontroller connected to an Ultrasonic Distance Sensor. The sensor's distance is displayed as 73cm. The console output shows the following messages:

```
object is near
Sending payload: {"distance":73.61,"object":"Near"}
Publish ok
Distancein cm73.61
object is near
Sending payload: {"distance":73.61,"object":"Near"}
Publish ok
```

The screenshot shows the IBM Watson IoT Platform interface. The left sidebar contains navigation icons. The main content area is titled 'Device Drilldown - weather\_today'. It features a 'Recent Events' section with a table of data points.

Event	Value	Format	Last Received
State	Data	json	a few seconds ago
Device Information	Data	json	a few seconds ago
Metadata	Data	json	a few seconds ago
Diagnostics	Data	json	a few seconds ago
Connection Logs	Data	json	a few seconds ago
Device Actions	Data	json	a few seconds ago

Below the table, there is a 'State' section with a description: 'This table shows a list of data points that are reported by this device.' and a toggle switch for 'Showing Raw Data'.

## OBJECT FAR AWAY FROM DEVICE:

The screenshot shows the Wokwi IDE interface. The left pane displays an Arduino sketch for an ESP32 connected to an ultrasonic sensor. The right pane shows a simulation of the hardware.

```

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
9 #define ORG "qxms592" //IBM ORGANIZATION ID
10 #define DEVICE_TYPE "weather_device" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "jwSiUM-qppnF1*TRa" //Token
13 String data3;
14 float dist;
15
16 //----- Customise the above values -----
17
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform a
20 char subscribeTopic[] = "iot-2/cmd/test/fmt/string"; // cmd REPRESENT command type AND TO
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25 //-----
26
27 WiFiClient wifiClient; // creating the instance for wifiClient
28 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
29
30 int LED = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()
34 {
35   Serial.begin(115200);
36   pinMode(trig, OUTPUT);
  
```

The simulation window shows an 'Editing Ultrasonic Distance Sensor' dialog with a distance of 259cm. Below the simulation, a console window displays the following output:

```

no object found
Sending payload: {"distance":261.25,"object":"No"}
Publish ok
Distance in cm 261.25
no object found
Sending payload: {"distance":261.25,"object":"No"}
Publish ok
  
```

IBM Watson IoT Platform

Device Drilldown - weather\_today

Recent Events

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

Event	Value	Format	Last Received
Data	{"distance":261.25,"object":"No"}	json	a few seconds ago
Data	{"distance":261.25,"object":"No"}	json	a few seconds ago
Data	{"distance":261.25,"object":"No"}	json	a few seconds ago
Data	{"distance":261.26,"object":"No"}	json	a few seconds ago
Data	{"distance":261.25,"object":"No"}	json	a few seconds ago

State

This table shows a list of data points that are reported by this device.

Showing Raw Data | No Interfaces Available

REFERENCE:

<https://wokwi.com/projects/347131589513183827>