

## Assignment 4

Assignment Date	19 October 2022
Student Name	Mr. Santhosh S
Student Roll Number	737819ECR163
Maximum Marks	2 Marks

### Question-1:

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

Solution:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
const int trigPin = 5;
const int echoPin = 18;

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

#define ORG "p2cfk6" //IBM ORGANITION ID
#define DEVICE_TYPE "ULTRA" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "45" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "fF3ns!LKXhrBBbe5zm" //Token

#define SOUND_SPEED 0.034

long duration;
float dist;

//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/command/fmt/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, wifiClient); //calling the predefined
client id by passing parameter like server id, port and wificredential
```

```

void setup()// configuring the ESP32
{
    Serial.begin(115200);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    dist = duration * SOUND_SPEED/2;

    // Prints the distance in the Serial Monitor
    Serial.print("Distance: ");
    Serial.print(dist);
    Serial.println(" cm");
    delay(1000);

    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 form JSON to update the data to ibm cloud
    */
    if(dist<100)
    {
        String payload = "{\"Alert! Distance is less than 100\":";
        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");
}
}
else{
    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 mqttconnect() {
    if (!client.connected()) {
        Serial.print("Reconnecting client to ");
        Serial.println(server);
        while (!client.connect(clientId, authMethod, token)) {
            Serial.print(".");
            delay(500);
        }
        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());
}

```

WOKWI LINK: <https://wokwi.com/projects/347138238103683666>

OUTPUT IN WOKWI:

WOKWI

SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 const int trigPin = 5;
4 const int echoPin = 18;
5
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "p2cfk6" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "ULTRA" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "45" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "Ff3nsLKXhr8Bbe5zm" //Token
12
13 #define SOUND_SPEED 0.034
14
15 long duration;
16 float dist;
17
18 //----- Customise the above values -----
19
20 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
21 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform a
22 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
23 char authMethod[] = "use-token-auth"; // authentication method
24 char token[] = TOKEN;
25 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
26
27 //-----
28
29 WiFiClient wificlient; // creating the instance for wificlient
30 PubSubClient client(server, 1883, wificlient); //calling the predefined client id by pass
31
32
33 void setup() // configuring the ESP32
34 {
```

Simulation

Connecting to ...  
Wifi connected  
IP address:  
10.10.0.2  
Reconnecting client to p2cfk6.messaging.internetofthings.ibmcloud.com

WOKWI

SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
17 //----- Customise the above values -----
18
19 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
20 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform a
21 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
22 char authMethod[] = "use-token-auth"; // authentication method
23 char token[] = TOKEN;
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
25
26 //-----
27
28 WiFiClient wificlient; // creating the instance for wificlient
29 PubSubClient client(server, 1883, wificlient); //calling the predefined client id by pass
30
31
32 void setup() // configuring the ESP32
33 {
34   Serial.begin(115200);
35   pinMode(trigPin, OUTPUT);
36   pinMode(echoPin, INPUT);
37   delay(10);
38   Serial.println();
39   wificlient.connect();
40   mqtt.connect();
41 }
42
43 void loop() // Recursive Function
44 {
45   digitalWrite(trigPin, LOW);
46   delayMicroseconds(2);
47   digitalWrite(trigPin, HIGH);
48   delayMicroseconds(10);
```

Simulation

Distance: 391.95 cm  
Sending payload: {"Distance":391.95}  
Publish ok  
Distance: 391.97 cm  
Sending payload: {"Distance":391.97}  
Publish ok

WOKWI

SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
17 //----- Customise the above values -----
18
19 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
20 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform a
21 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
22 char authMethod[] = "use-token-auth"; // authentication method
23 char token[] = TOKEN;
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
25
26 //-----
27
28 WiFiClient wificlient; // creating the instance for wificlient
29 PubSubClient client(server, 1883, wificlient); //calling the predefined client id by pass
30
31
32 void setup() // configuring the ESP32
33 {
34   Serial.begin(115200);
35   pinMode(trigPin, OUTPUT);
36   pinMode(echoPin, INPUT);
37   delay(10);
38   Serial.println();
39   wificlient.connect();
40   mqtt.connect();
41 }
42
43 void loop() // Recursive Function
44 {
45   digitalWrite(trigPin, LOW);
46   delayMicroseconds(2);
47   digitalWrite(trigPin, HIGH);
48   delayMicroseconds(10);
```

Simulation

Publish ok  
Distance: 64.96 cm  
Sending payload: {"Alert! Distance is less than 100":64.96}  
Publish ok  
Distance: 64.96 cm  
Sending payload: {"Alert! Distance is less than 100":64.96}  
Publish ok

## OUTPUT IN IBM CLOUD (DEVICE RECENT EVENTS):

The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows 'IBM Watson IoT Platform' and a user profile for 'santhosh.s.19ece@kongu.edu' with ID 'p2cfr6'. The main navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present, and an 'Add Device' button is in the top right.

The main content area shows a list of devices. The first device is ID 12, status 'Disconnected', type 'ABCD', class 'Device', added on 'Oct 13, 2022 11:13 AM'. The second device is ID 45, status 'Connected', type 'ULTRA', class 'Device', added on 'Oct 23, 2022 10:27 AM'. This second device is selected, and its details are shown in a modal window.

The modal window has tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a list of events. Below the tab is a note: 'The recent events listed show the live stream of data that is coming and going from this device.'

Event	Value	Format	Last Received
IoTSensor	{\"Alert! Distance is less than 100\":64.96}	json	a few seconds ago
IoTSensor	{\"Alert! Distance is less than 100\":64.96}	json	a few seconds ago
IoTSensor	{\"Distance \":391.95}	json	a minute ago
IoTSensor	{\"Distance \":391.95}	json	a minute ago
IoTSensor	{\"Distance \":391.95}	json	a minute ago