

#### Assignment -4

Assignment Date	12 october 2022
Student Name	Santhoshshivan k
Student Roll Number	513419106034
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. Upload document with wokwi share link and images of IBM cloud

##### Solution:

```
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient wifiClient;
String data3;
#define ORG "n7xtmx"
#define DEVICE_TYPE "dth"
#define DEVICE_ID "Dth_123"
#define TOKEN "Hr)eVB0veZgPG6cnwq"
#define speed 0.034
#define led 14
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();

const int trigpin=5;
const int echopin=18;
String command;
String data="";

long duration;
float dist;

void setup()
{
```

```

    Serial.begin(115200);
    pinMode(led, OUTPUT);
    pinMode(trigpin, OUTPUT);
    pinMode(echopin, INPUT);
    wifiConnect();
    mqttConnect();
}

void loop() {
    bool isNearby = dist < 100;
    digitalWrite(led, isNearby);

    publishData();
    delay(500);

    if (!client.loop()) {
        mqttConnect();
    }
}

void wifiConnect() {
    Serial.print("Connecting to "); Serial.print("Wifi");
    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.print("WiFi connected, IP address: ");
    Serial.println(WiFi.localIP());
}

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

void initManagedDevice() {
    if (client.subscribe(topic)) {
        // Serial.println(client.subscribe(topic));
        Serial.println("IBM subscribe to cmd OK");
    } else {

```

```

        Serial.println("subscribe to cmd FAILED");
    }
}
void publishData()
{
    digitalWrite(trigpin, LOW);
    digitalWrite(trigpin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigpin, LOW);
    duration=pulseIn(echopin, HIGH);
    dist=duration*speed/2;
    if(dist<100){
        String payload = "{\"Normal Distance\":\"";
        payload += dist;
        payload += "\"}";

        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);
        if (client.publish(publishTopic, (char*) payload.c_str())) {
            Serial.println("Publish OK");
        }
    }

    if(dist>101 && dist<111){
        String payload = "{\"Alert distance\":\"";
        payload += dist;
        payload += "\"}";

        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);
        if(client.publish(publishTopic, (char*) payload.c_str())) {
            Serial.println("Warning crosses 110cm -- it automatically of the loop");
            digitalWrite(led, HIGH);
        }else {
            Serial.println("Publish 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++){

```

```

    dist += (char)payload[i];
}
Serial.println("data:" + data3);
if(data3=="lighton"){
    Serial.println(data3);
    digitalWrite(led,HIGH);
}
data3="";
}
}

```

**Output:**

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch for an ESP32. The sketch includes libraries for WiFi and PubSubClient, defines constants for the server, topic, token, and LED pin, and implements a loop that publishes distance data to IBM Cloud IoT. On the right, the 'Simulation' window shows a 3D model of the ESP32 and ultrasonic sensor. Below the simulation, the serial output shows the device successfully publishing distance data to IBM Cloud IoT.

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 WiFiClient wifiClient;
4 String data3;
5 #define ORG "n7xtmx"
6 #define DEVICE_TYPE "dth"
7 #define DEVICE_ID "Dth_123"
8 #define TOKEN "Hr)eV80veZgPG6cmwq"
9 #define speed 0.034
10 #define led 14
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char topic[] = "iot-2/cmd/home/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 PubSubClient client(server, 1883, wifiClient);
18 void publishData();
19
20
21 const int trigpin=5;
22 const int echopin=18;
23 String command;
24 String data="";
25
26 long duration;
27 float dist;
28
29

```

Simulation Output:

```

Publish OK
Sending payload: {"Normal Distance":58.99}
Publish OK
Sending payload: {"Normal Distance":58.97}
Publish OK

```

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various IoT functions. The main content area displays details for a device named 'Dth\_123', which is 'Connected'. Below this, there are tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. It lists five 'Data' events, each with a JSON value containing 'Normal Distance':58.99, in 'json' format, received 'a few seconds ago'. A status indicator at the bottom right shows '1 Simulation running'.

Event	Value	Format	Last Received
Data	{"Normal Distance":58.99}	json	a few seconds ago
Data	{"Normal Distance":58.99}	json	a few seconds ago
Data	{"Normal Distance":58.99}	json	a few seconds ago
Data	{"Normal Distance":58.99}	json	a few seconds ago
Data	{"Normal Distance":58.99}	json	a few seconds ago

1 Simulation running

Wokwi link:

<https://wokwi.com/projects/347734379586388564>