

ASSIGNMENT 4

Student Name	B.Ranjini
Student Register Number	421219106303
Team ID	PNT2022TMID38891

Project Name: IoT based safety gadget for child safety monitoring and notification

QUESTION:

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 the link and image of IBM cloud

CODE:

```
#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 "epmjfz" //IBM ORGANITION ID
#define DEVICE_TYPE "IoT_Child_safety" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "ran23" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //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, port and 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();
}
```

ASSIGNMENT 4

```
}
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");
    }
}
```

ASSIGNMENT 4

```
}
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);
    // }
```

ASSIGNMENT 4

```
// }  
data3="";
```

```
}
```

Reference: <https://wokwi.com/projects/348304868712120916>

The Wokwi IDE interface displays a C++ sketch for an ESP32 microcontroller. The sketch includes the following code:

```
1 #include <WiFi.h> //library for wifi  
2 #include <PubSubClient.h> //library for MQTT  
3  
4  
5 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)  
6  
7 //-----credentials of IBM Accounts-----  
8  
9 #define ORG "epmjfz" //IBM ORGANITION ID  
10 #define DEVICE_TYPE "IoT_Child_safety" //Device type mentioned in ibm watson IOT Platform  
11 #define DEVICE_ID "ran23" //Device ID mentioned in ibm watson IOT Platform  
12 #define TOKEN "12345678" //Token  
13 String data3;  
14 float dist;  
15 //----- Customise the above values -----  
16 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name  
17 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event  
18 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command  
19 char authMethod[] = "use-token-auth"; // authentication method  
20 char token[] = TOKEN;  
21 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id  
22  
23 //-----  
24 WiFiClient wifiClient; // creating the instance for wifiClient  
25 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined  
26  
27 int LED = 4;  
28 int trig = 5;  
29 int echo = 18;  
30 void setup()  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

The simulation window shows a visual representation of the ESP32 and the HC-SR04 sensor. The sensor is connected to the ESP32 via four wires (VCC, GND, Trig, Echo). A red LED is also connected to the ESP32. The simulation output shows the following sequence of events:

```
object is near  
Sending payload: {"distance":90.77,"object":"Near"}  
Publish ok  
Distancein cm90.77  
object is near  
Sending payload: {"distance":90.77,"object":"Near"}  
Publish ok
```

The IBM Watson IoT Platform dashboard displays a list of devices. The device 'ran23' is connected and has several recent events listed. The events are as follows:

Event	Value	Format	Last Received
Data	{"distance":90.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":90.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":90.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":155.33,"object":"No"}	json	a few seconds ago
Data	{"distance":155.33,"object":"No"}	json	a few seconds ago

0 Simulations running