

Distance Detection Using Ultrasonic Sensor Assignment	6 November 2022
Date	
Student Name	Naveen Kumar T
Student Roll Number	710019106302
Maximum Marks	2 Marks
ASSIGNMENT 4	

WOKWI 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 " qz2oyu "//IBM ORGANITION ID

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

#define DEVICE_ID "1 "//Device ID mentioned in ibm watson IOT Platform

#define TOKEN " Naveen@12" //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="";
```

```
}
```

WOKWI LINK: <https://wokwi.com/projects/347574893885260372>

WOKWI CIRCUIT DIAGRAM AND WOKWI OUTPUT:

The screenshot displays the Wokwi web IDE interface. On the left, the code editor shows a C++ program for an ESP32. The code includes headers for WiFi and PubSubClient, defines MQTT server and topic details, and sets up an ultrasonic sensor (HC-SR04) on pins 5 and 18. The main loop reads the sensor's distance and publishes it to an MQTT broker. If the distance is less than 100cm, it also publishes an alert message. On the right, the simulation window shows the ESP32 and the ultrasonic sensor connected. The sensor's output is displayed as 'Distance (cm): 121.97'. The MQTT payload being sent is shown as '{\"Distance\":40.97,\"ALERT!!\":\"Distance less than 100cms\"}'.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int
4 payloadLength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "q2oyu"/IBM ORGANIZATION ID
7 #define DEVICE_TYPE "Naveenkumar"/Device type mentioned in ibm watson IOT Pla
8 #define DEVICE_ID "1"/Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "Naveen@12" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifIClient;
18 PubSubClient client(server, 1883, callback, wifIClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wifIconnect();
29   mqttconnect();
30 }
```

Simulation window output:

```
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 40.97
ALERT!!
Sending payload: {"Distance":40.97,"ALERT!!":"Distance less than 100cms"}
Publish ok
```

IBM WATSON OUTPUT:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various platform features. The main content area shows a device named 'Naveenkumar' with a status of 'Connected'. Below this, the 'Recent Events' tab is selected, displaying a table of live data events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The events listed are JSON payloads containing random numbers and distance alerts. A notification at the bottom right indicates '1 Simulation running'.

Event	Value	Format	Last Received
data	{"randomNumber":57}	json	a few seconds ago
data	{"randomNumber":98}	json	a few seconds ago
data	{"randomNumber":62}	json	a few seconds ago
Data	{"Distance":40.97,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":40.97,"ALERT!!":"Distance less than ...	json	a few seconds ago

1 Simulation running