

Distance Detection Using Ultrasonic Sensor Assignment	5 November 2022
Date	
Student Name	Mukil Arasi S
Student Roll Number	710019106030
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 "vrpc8b"//IBM ORGANITION ID

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

#define DEVICE_ID "123456"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN " Mukil@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/347571927288644178>

WOKWI CIRCUIT DIAGRAM AND WOKWI OUTPUT:

The screenshot displays the Wokwi IDE interface. On the left, the code editor shows the following C++ code:

```
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 "vrc8b"//IBM ORGANIZATION ID
7 #define DEVICE_TYPE "Ecdedevic"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "123456"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "huk1q12" //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 }
```

On the right, the simulation window shows a circuit diagram with an ESP32 microcontroller and an HC-SR04 ultrasonic sensor. The sensor's distance is displayed as 171cm. Below the diagram, the simulation output log shows the following sequence of events:

```
Publish ok
Distance (cm): 89.98
ALERT!!
Sending payload: {"Distance":89.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 170.97
Distance (cm): 170.97
```

IBM WATSON OUTPUT:

The screenshot displays the IBM Watson IoT Platform dashboard. 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 table of events with the following data:

Event	Value	Format	Last Received
data	{"randomNumber":2}	json	a few seconds ago
data	{"randomNumber":15}	json	a few seconds ago
Data	{"Distance":22.95,"ALERT!!":"Distance less than ...	json	a few seconds ago
data	{"randomNumber":62}	json	a few seconds ago
Data	{"Distance":96.99,"ALERT!!":"Distance less than ...	json	a few seconds ago

Below the table, there is a pagination control showing 'Items per page 50' and '1-1 of 1 item'. At the bottom of the dashboard, a status box indicates '1 Simulation running'. The browser's address bar shows the URL 'vrpc8b.internetofthings.ibmcloud.com/dashboard/devices/browse'. The Windows taskbar at the bottom displays the system clock as 19:48 on 06-11-2022, along with the temperature 31°C and language settings.