

# Assignment -4

Assignment Date	19 October 2022
Student Name	Sushmitha.S
Student Roll Number	731319205032

Write code and connection in Wowki for ultrasonic sensor.

Whenever distance is less than 100 cm send "Alert" to IBM cloud and display in device recent events

Wowki link: <https://wokwi.com/projects/new/esp32>

Step 1 : Add new device in IBM cloud

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Add Device

## Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
>	1234	Disconnected	abcd	Device	Oct 12, 2022 6:29 PM	
>	123456	Disconnected	aaaa	Device	Oct 12, 2022 6:43 PM	
>	5678	Connected	sulana_4	Device	Oct 24, 2022 12:08 PM	

Items per page 50 | 1-3 of 3 items

1 of 1 page

## Step 2: Complete the Circuit and run the program

The screenshot shows the WOKWI simulation environment. On the left, the sketch.ino file is open, displaying the following code:

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
5 \
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "xdhbd0" //IBM ORGANITION ID
9 #define DEVICE_TYPE "sulana_4" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "5678" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "PF32(1uMUVftCL7)h" //Token
12 String data3;
13
14 //----- Customise the above values -----
15 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
16 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform and format in
17 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND COMMAND IS T
18 char authMethod[] = "use-token-auth"; // authentication method
19 char token[] = TOKEN;
20 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
21
22 //-----
23 WiFiClient wificlient; // creating the instance for wificlient
24 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client id by passing
25 const int trigpin = 5;
26 const int echopin = 18;
27 const int ledpin = 2;
28
29
30
31 long duration ;
32 float distance;
33 #define sound_speed 0.034
34 void setup() {
35 // put your setup code here, to run once:
36
```

On the right, the simulation window shows a circuit diagram with an ESP32 microcontroller, an ultrasonic distance sensor, and an LED. The sensor is connected to the ESP32. The LED is connected to the ESP32. The serial monitor shows the following output:

```
94.94
Sending payload: {"ALERT...!! ": 94.98}
Publish ok
ALERT...!!!
94.98
Sending payload: {"ALERT...!! ": 94.96}
Publish ok
```

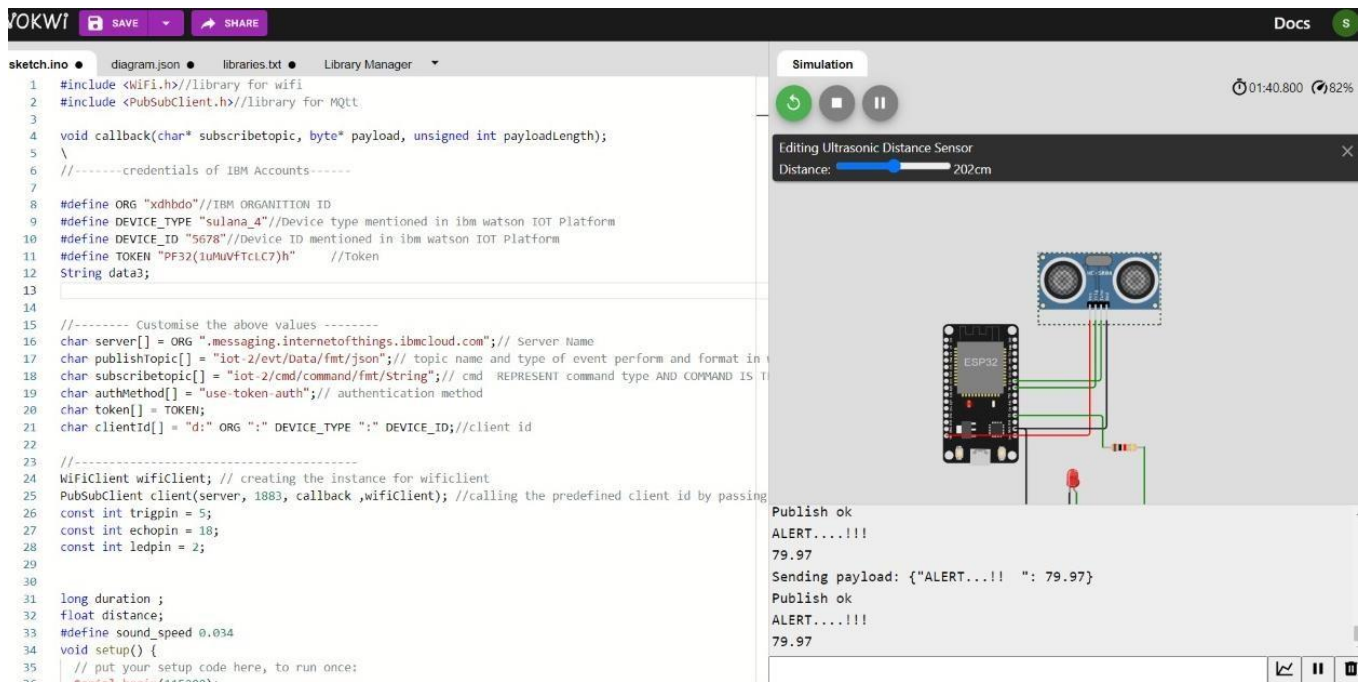
## Output in WOKWI

a) when the distance is below 100 cms

The screenshot shows the WOKWI simulation environment. On the left, the sketch.ino file is open, displaying the same code as in the previous screenshot. On the right, the simulation window shows the same circuit diagram. The serial monitor shows the following output:

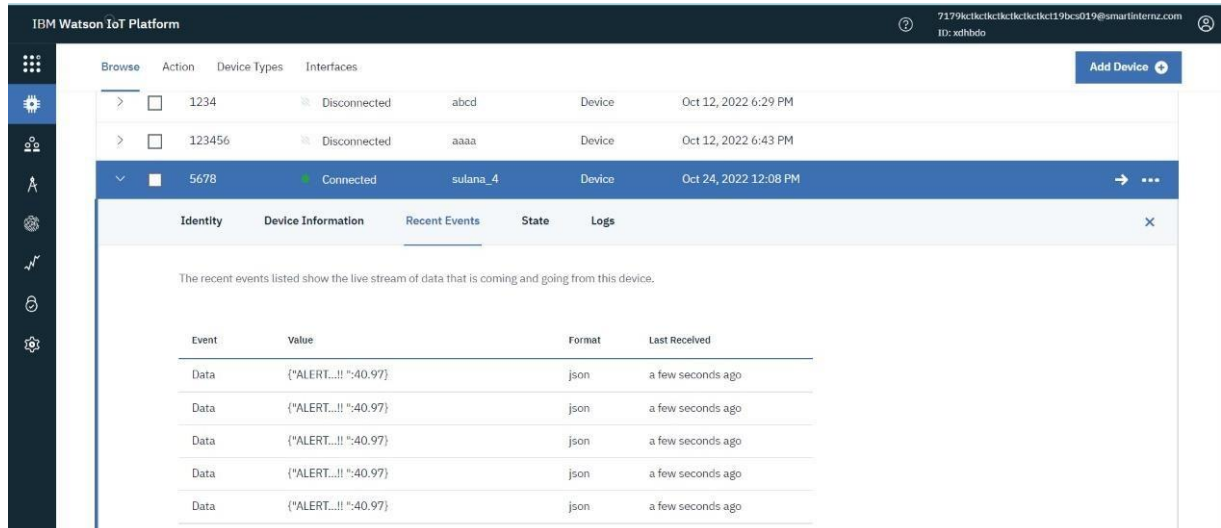
```
ALERT...!!!
79.97
Sending payload: {"ALERT...!! ": 79.97}
Publish ok
ALERT...!!!
79.97
Sending payload: {"ALERT...!! ": 79.97}
```

b) when the distance is above 100 cms ,(no alert message is displayed here for 202 cm)



Output in IBM CLOUD (Watson Platform)

Displayed in device recent events



Program:

```

#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 "xdhbdo"//IBM ORGANITION ID
#define DEVICE_TYPE "sulana_4"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "5678"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "PF32(1uMuVfTcLC7)h" //Token
String data3;
//----- 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/command/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
const int trigpin = 5;
const int echopin = 18;
const int ledpin = 2;

long duration ;
float distance;
#define sound_speed 0.034
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, OUTPUT);
  pinMode(ledpin, OUTPUT);
  wificonnect();
  mqttconnect();
}

void loop() {
  digitalWrite(trigpin, LOW);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);

  duration= pulseIn(echopin,HIGH);
  distance = duration * sound_speed /2;
  if(distance<=100){
    PublishData(distance);
    delay(1000);
    if (!client.loop()) {

```

```

    mqttconnect();
}
    digitalWrite(ledpin, HIGH);
    Serial.println("ALERT ... !!!");
    Serial.println(distance);
}
else
{
    digitalWrite(ledpin, LOW);
}
// put your main code here, to run repeatedly:
delay(10); // this speeds up the simulation
}
/*.....retrieving to Cloud..... */
void PublishData(float distance) {
    mqttconnect();//function call for connecting to ibm
    // creating the String in in form JSon to update the data to ibm cloud
    String payload = "{\"ALERT...!! \": ";
    payload += distance;
    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=="lighton")
    {
        Serial.println(data3);
    }
    else
    {
        Serial.println(data3);
    }
    data3="";
}

```