

## ASSIGNMENT – 04

Write code and connection in Wowki for ultrasonic sensor.

Whenever distance is less than 100 cms send **“Alert”** to IBM cloud and display in device recent events

Step 1 . Completed to build Circuit and run program

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing 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 "7kb3es" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "deepi_resheery" //Device type mentioned in ibm watson IOT
10 #define DEVICE_ID "56789" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "Cusp45PBuBEsQeJ_9N" //Token
12 String data3;
13
14
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 //-----
25 WiFiClient wificlient; // creating the instance for wificlient
26 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined
27 const int trigpin = 5;
28 const int echopin = 18;
```

On the right, the 'Simulation' window shows a circuit diagram. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's Trig pin is connected to the ESP32's pin 5, and its Echo pin is connected to the ESP32's pin 18. A red LED is connected to the ESP32's GND pin and a 10k resistor.

## Step 2. Output in WOWKI

### a) when distance is below 100 cms

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* subscribtopic, byte* payload, unsigned int payloadLength)
5 {
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "7kb3es" //IBM ORGANITION ID
9 #define DEVICE_TYPE "deepi_reshepeery" //Device type mentioned in ibm watson IOT
10 #define DEVICE_ID "56789" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "Cusp45PBuBEsQeJ_9N" //Token
12 String data3;
13
14
15
16 //----- Customise the above values -----
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of even
19 char subscribtopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24
25 //-----
26 WiFiClient wificlient; // creating the instance for wificlient
27 PubSubClient client(server, 1883, callback ,wificlient); //calling the predefi
28 const int trigpin = 5;
29 const int echopin = 18;
```

On the right, the simulation window shows a circuit diagram with an ESP32 microcontroller, an HC-SR04 ultrasonic sensor, and a red LED. The simulation output console displays the following log:

```
Connecting to ....
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 7kb3es.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Sending payload: {"ALERT...!! ": 48.01}
Publish ok
ALERT...!!!
48.01
```

### B) When Distance is above 100 cms

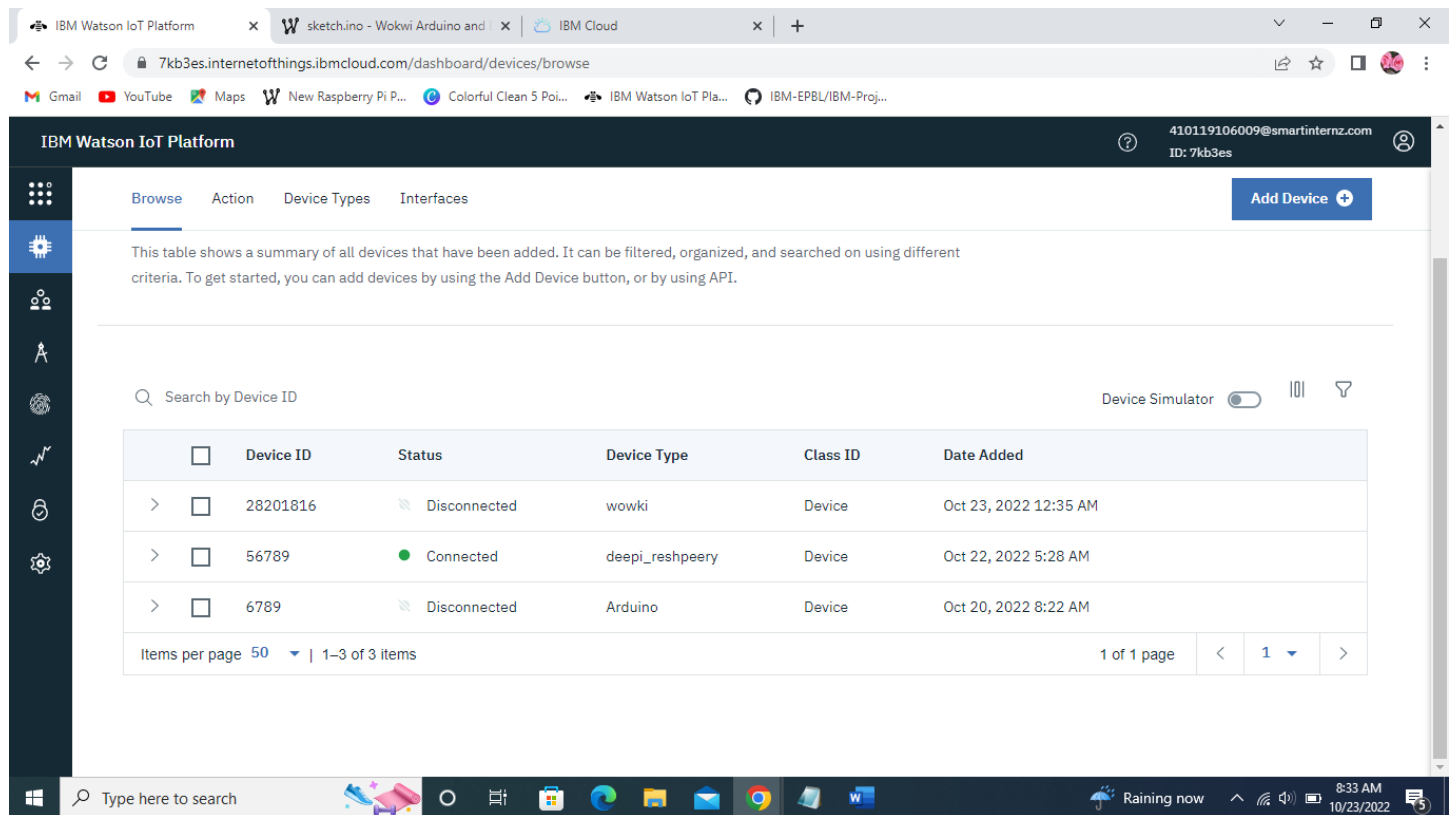
The screenshot shows the WOKWI simulation environment. On the left, the sketch.ino file is open, displaying the same code as in the previous screenshot.

On the right, the simulation window shows the same circuit diagram. The simulation output console displays the following log:

```
Publish ok
ALERT...!!!
47.97
Sending payload: {"ALERT...!! ": 47.97}
Publish ok
ALERT...!!!
47.97
```

An "Editing Ultrasonic Distance Sensor" dialog box is open, showing the distance set to 196cm.

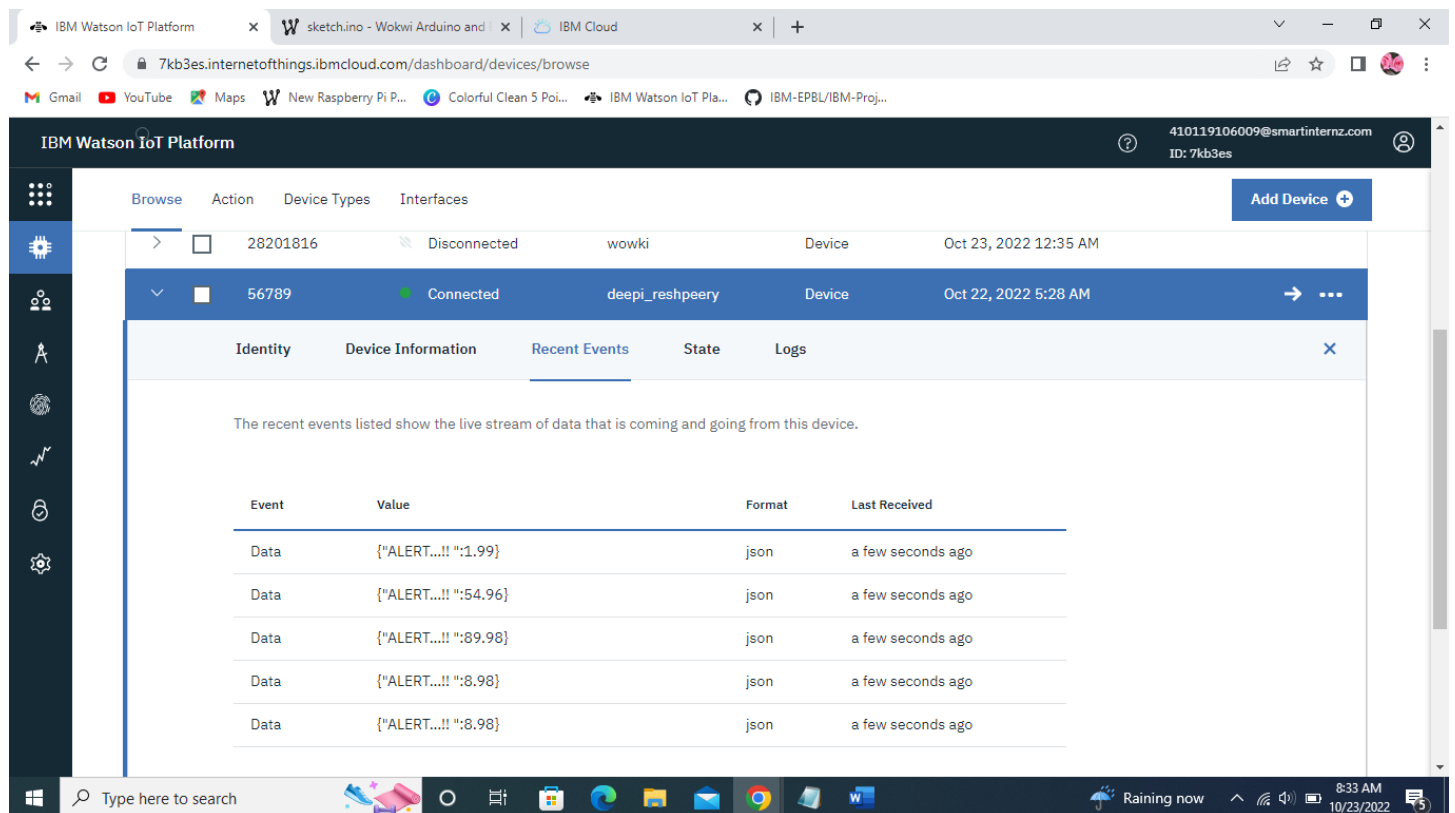
### Step 3. ADD new devices in IBM cloud



The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area displays a table of devices with columns: Device ID, Status, Device Type, Class ID, and Date Added. There are three devices listed: 28201816 (Disconnected, wowki), 56789 (Connected, deepi\_resheery), and 6789 (Disconnected, Arduino). A search bar and a 'Device Simulator' toggle are also visible.

Device ID	Status	Device Type	Class ID	Date Added
28201816	Disconnected	wowki	Device	Oct 23, 2022 12:35 AM
56789	Connected	deepi_resheery	Device	Oct 22, 2022 5:28 AM
6789	Disconnected	Arduino	Device	Oct 20, 2022 8:22 AM

### Step 4. Output in IBM CLOUD (Watson Platform)



The screenshot shows the IBM Watson IoT Platform dashboard with the 'Recent Events' tab selected for device 56789. The 'Recent Events' tab displays a table of events with columns: Event, Value, Format, and Last Received. The events are listed as follows:

Event	Value	Format	Last Received
Data	{"ALERT...!! ":1.99}	json	a few seconds ago
Data	{"ALERT...!! ":54.96}	json	a few seconds ago
Data	{"ALERT...!! ":89.98}	json	a few seconds ago
Data	{"ALERT...!! ":8.98}	json	a few seconds ago
Data	{"ALERT...!! ":8.98}	json	a few seconds ago

## 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 "7kb3es"//IBM ORGANITION ID

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

#define DEVICE_ID "56789"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "CUsp45PBuBEsQeJ_9N" //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/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

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="";  
}
```

### **Conclusion:**

Here I showed the **ALERT** and **DISTANCE** in IBM cloud when the distance is less than 100 cms.