

**V S B ENGINEERING COLLEGE, KARUR**

**Department of Electronics and Communication Engineering**

**IBM NALAIYA THIRAN**

**ASSIGNMENT 4**

**Name :** Kamalraj S

**Assignment:**

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 devicerecent events.

**Code:**

```
#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;String data3;

#define speed 0.034

#define led 15

const int trigpin=13;

const int echopin=12;

String command;

String data="";

long duration;

float dist;

//-----credentials of IBM Accounts-----

#define ORG "gw5dmy"

#define DEVICE_TYPE "baladevice"

#define DEVICE_ID "baladeviceid"

#define TOKEN "3bm0as0lp6ak2nq0jx0iw2cx"

//----- Customize the above values -----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Data/fmt/json";

char topic[] = "iot-2/cmd/command/fmt/String";

char authMethod[] = "use-token-auth";
```

```

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

PubSubClient client(server, 1883, wifiClient);

void publishData()

void setup()
{
  Serial.begin(115200);

  pinMode(led, OUTPUT);

  pinMode(trigpin, OUTPUT);

  pinMode(echopin, INPUT);

  wifiConnect();

  mqttConnect();
}

void loop()
{
  bool Nearby = dist < 100;digitalWrite(led, Nearby);

  publishData();delay(500);

  if (!client.loop())
  {
    mqttConnect();
  }
}

void wifiConnect()
{
  Serial.print("Connecting to ");

  Serial.print("Wifi");

  WiFi.begin("Wokwi-GUEST", "", 6);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
  }
}

```

```

Serial.print(".");

}

Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

}

void mqttConnect()

{

if(!client.connected())

{

Serial.print("Reconnecting MQTT client to ");

Serial.println(server);

while (!client.connect(clientId, authMethod, token))

{

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void initManagedDevice()

{

if (client.subscribe(topic))

{

// Serial.println(client.subscribe(topic));

Serial.println("IBM subscribe to cmd OK");

}

else

{

Serial.println("subscribe to cmd FAILED");

```

```

}

}

/*.....retrieving to
Cloud */

void publishData()

{
digitalWrite(trigpin,LOW);
digitalWrite(trigpin,HIGH);
delayMicroseconds(10);
digitalWrite(trigpin,LOW);
duration=pulseIn(echopin,HIGH);
dist=duration*speed/2; if(dist<100)
{
String payload = "{\"Alert Distance is \":\"";payload += dist;
payload += "}";

Serial.print("\n");
Serial.print("Sending payload:");
Serial.println(payload);
if(client.publish(publishTopic, (char*)payload.c_str()))
{
Serial.println("Publish OK");// if it successfully uploads data on thecloud then it
will print publish ok in Serial monitor or else it will printpublish failed
digitalWrite(led,HIGH);
}
}

if(dist>100)
{
String payload = "{\"Distance is \":\"";payload += dist;
payload += "}";

```

```

Serial.print("\n");

Serial.print("Sending payload: ");

Serial.println(payload);

if(client.publish(publishTopic, (char*) payload.c_str()))

{

Serial.println("Cross the alert distance");

digitalWrite(led,LOW);

}

else

{

Serial.println("Publish FAILED");

}

}

}

void callback(char* subscribeTopic, byte* payload, unsigned intpayloadLength)

{

Serial.print("callback invoked for topic:");

Serial.println(subscribeTopic);

for(int i=0; i<payloadLength; i++)

{

dist += (char)payload[i];

}

Serial.println("data:" + data3);if(data3=="lighton");

{

Serial.println(data3);digitalWrite(led,HIGH);

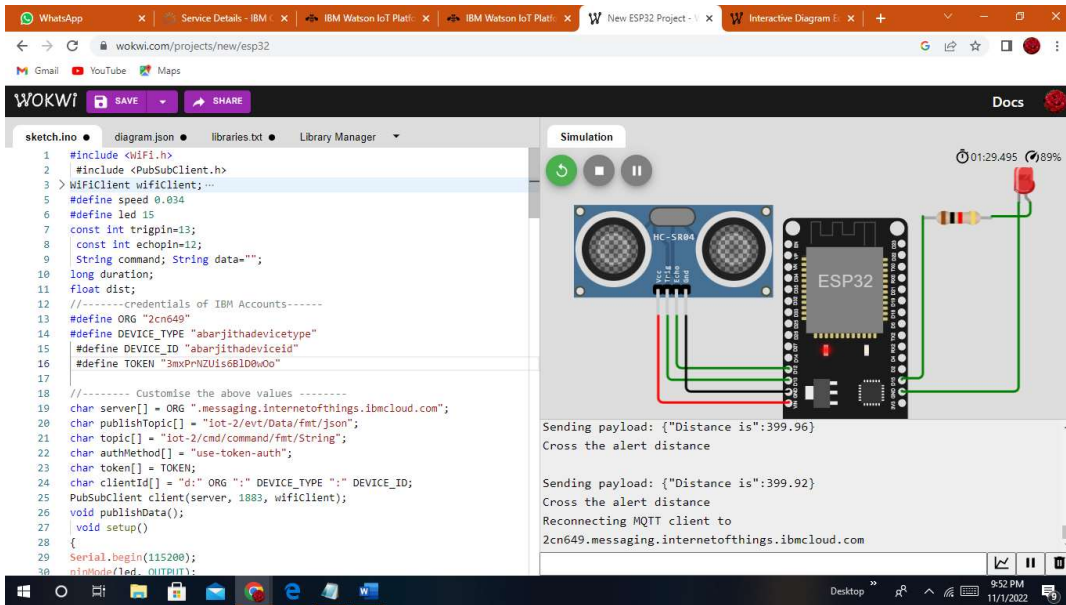
}

data3="";

}

```

## Connection:



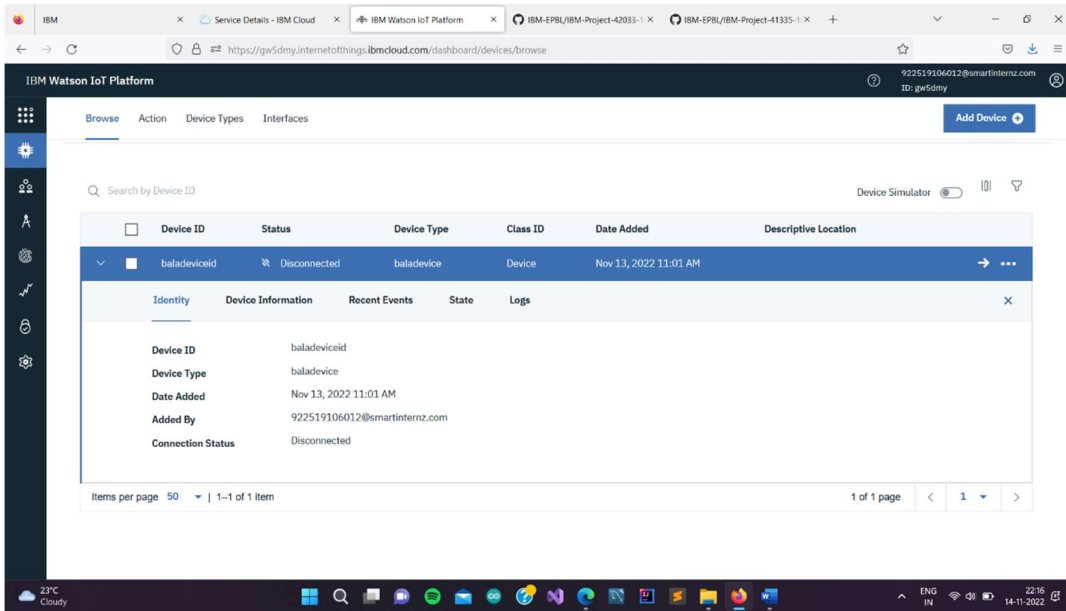
The screenshot shows the Wokwi IDE interface. On the left, the sketch.ino file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 > WiFiClient wifiClient;
4 #define speed 0.034
5 #define led 15
6 const int trigger=13;
7 const int echoPin=12;
8 String command; String data="";
9 long duration;
10 float dist;
11 //-----credentials of IBM Accounts-----
12 #define ORG "2cn649"
13 #define DEVICE_TYPE "abarrjithaddevicetype"
14 #define DEVICE_ID "abarrjithaddevicetype"
15 #define TOKEN "3mxPrnZUis681D8vOo"
16 //----- Customise the above values -----
17
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
19 char publishTopic[] = "iot-2/evt/Data/fmt/json";
20 char topic[] = "iot-2/cmd/command/fmt/String";
21 char authMethod[] = "use-token-auth";
22 char token[] = TOKEN;
23
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
25 PubSubClient client(server, 1883, wifiClient);
26 void publishData();
27 void setup()
28 {
29   Serial.begin(115200);
30   pinMode(trigger, OUTPUT);
31 }
```

The simulation window on the right shows the following output:

```
Sending payload: {"Distance is":399.96}
Cross the alert distance

Sending payload: {"Distance is":399.92}
Cross the alert distance
Reconnecting MQTT client to
2cn649.messaging.internetofthings.ibmcloud.com
```



The screenshot shows the IBM Watson IoT Platform dashboard. The 'Browse' tab is selected, and the 'Device Simulator' toggle is turned off. The table below lists the devices:

| Device ID   | Status       | Device Type | Class ID | Date Added            | Descriptive Location |
|-------------|--------------|-------------|----------|-----------------------|----------------------|
| baladviceid | Disconnected | baladvice   | Device   | Nov 13, 2022 11:01 AM |                      |

The details modal for the device 'baladviceid' is open, showing the following information:

| Identity          | Device Information            | Recent Events | State | Logs |
|-------------------|-------------------------------|---------------|-------|------|
| Device ID         | baladviceid                   |               |       |      |
| Device Type       | baladvice                     |               |       |      |
| Date Added        | Nov 13, 2022 11:01 AM         |               |       |      |
| Added By          | 922519106012@smartinternz.com |               |       |      |
| Connection Status | Disconnected                  |               |       |      |