

Assignment - 4

Assignment Date	20 October 2022
Student Name	Mr.N.Nithyananthan
Student Roll Number	621319106312
Maximum Marks	2 Marks

Question :

Write a code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send "alert" to ibm cloud and display in device recent events.

Solution :

(Link >>> <https://wokwi.com/projects/347240656089907795>)

Code in Wokwi :

(sketch.ino)

```
#include<WiFi.h>//library for wifi
#include<PubSubClient.h>//library for MQTT
```

```
voidcallback(char* subscribetopic, byte* payload, unsignedintpayloadLength);
```

```
// credentials of IBM Accounts
#defineORG "n6r19n"//IBM Organisation ID
#defineDEVICE_TYPE "NaaghuIoT"//Device type mentioned in ibm watson IOT Platform
#defineDEVICE_ID "06112002"//Device ID mentioned in ibm watson IOT Platform
#defineTOKEN "98765432"//Token
Stringdata3;
```

```
// Customise the above values
charserver[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
charpublishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
perform and format in which data to be send
charsubscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING
charauthMethod[] = "use-token-auth";// authentication method
chartoken[] = TOKEN;
charclientId[] = "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
constinttrigpin = 5;
constintechopin = 18;
```

```

const int ledpin = 2;

long duration ;
float distance;
#define sound_speed 0.034
void setup()
{
    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);
    }
    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");
    }
}

```

```

voidmqttconnect()
{
    if(!client.connected())
    {
        Serial.print("Reconnecting client to ");
        Serial.println(server);
        while(!!!client.connect(clientId, authMethod, token))
        {
            Serial.print(".");
            delay(500);
        }
        initManagedDevice();
        Serial.println();
    }
}

```

```

voidwificonnect() // 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());
}

```

```

voidinitManagedDevice()
{
    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="";
}

```

(diagram.json)

```

{
    "version": 1,
    "author": "312. Nithyananthan N",
    "editor": "wokwi",
    "parts": [
        { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 35.33, "left": -109.33,
"attrs": {} },
        {
            "type": "wokwi-hc-sr04",
            "id": "ultrasonic1",
            "top": -61.7,
            "left": 78.5,
            "attrs": { "distance": "164"}
        },
        {
            "type": "wokwi-led",
            "id": "led1",
            "top": 134.3,
            "left": 195.16,
            "attrs": { "color": "red"}
        },
        {
            "type": "wokwi-resistor",
            "id": "r1",
            "top": 214.96,
            "left": 65.17,
            "attrs": { "value": "1000"}
        }
    ]
}

```

```

    ],
    "connections": [
      [ "esp:TX0", "$serialMonitor:RX", "", [ ] ],
      [ "esp:RX0", "$serialMonitor:TX", "", [ ] ],
      [ "led1:A", "r1:2", "red", [ "v49.07", "h-96.19" ] ],
      [ "r1:1", "esp:D2", "red", [ "v1.41", "h-48", "v-59.64" ] ],
      [ "led1:C", "esp:GND.1", "black", [ "v31.74", "h-36.99", "v-23.71" ] ],
      [
        "ultrasonic1:VCC",
        "esp:VIN",
        "red",
        [ "v28.57", "h-132.45", "v-48", "h-152", "v182", "h29.84" ]
      ],
      [ "esp:D5", "ultrasonic1:TRIG", "blue", [ "h0" ] ],
      [ "esp:D18", "ultrasonic1:ECHO", "green", [ "h0" ] ],
      [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ]
    ]
  }
}

```

(libraries.txt)

Wokwi Library List
 # See <https://docs.wokwi.com/guides/libraries>

PubSubClient

Wokwi Sketch and Simulation :

The screenshot displays the Wokwi web interface for a project titled 'sketch.ino'. The left pane shows the sketch code, which includes headers for `WiFi.h` and `PubSubClient.h`, and defines constants for an IBM Watson IoT device. The code sets up a WiFi client and a PubSubClient, then defines pin numbers for a trig pin (5), an echo pin (18), and a LED pin (2). The `setup()` function is partially visible.

The right pane shows a simulation of the ESP32 board. It is connected to a breadboard with a red LED and a resistor. The simulation status bar at the top right indicates a battery level of 54% and a timer at 00:44.283. The console output on the right shows the following messages:

```

Connecting to ..
WiFi connected
IP address:
10.10.0.2
Reconnecting client to n6r19n.messaging.internetofthings.ibmcloud.com
.....

```

The bottom of the image shows the Windows taskbar with the time 12:11 AM and date 11/3/2022.

Add a new device in IBM Cloud & Output in IBM Cloud (Watson Platform) :

The screenshot shows the 'Device Drilldown - 06112002' page in the IBM Watson IoT Platform. The left sidebar contains a navigation menu with options: Device Credentials, Connection Information, Recent Events, State, Device Information, Metadata, Diagnostics, Connection Logs, and Device Actions. The main content area is titled 'Device Credentials' and includes a warning message: 'Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.' Below this, a table displays the following information:

Organization ID	n6r19n
Device Type	NaaghuIoT
Device ID	06112002
Authentication Method	use-token-auth
Authentication Token	98765432

At the bottom of the main content area, it indicates '1 Simulation running'.

The screenshot shows the 'Recent Events' page for device 06112002. The page displays a table of recent events, which are alerts triggered by distance measurements. The table has the following structure:

Event	Value	Format	Last Received
event_2	{"Alert !!!":195.2}	json	a few seconds ago
event_2	{"Alert !!!":184.1}	json	a few seconds ago
event_2	{"Alert !!!":101.3}	json	a few seconds ago
event_2	{"Alert !!!":138.4}	json	a few seconds ago
event_2	{"Alert !!!":194.3}	json	a few seconds ago

At the bottom of the page, it indicates '0 Simulations running'.

Conclusion :

Whenever the distance is less than 100 cms send an "Alert" to the IBM cloud and display in the device **Recent Events**.