

Assignment - 4

Assignment Date	20 October 2022
Student Name	Mr.M.Kishor
Student Roll Number	621319106044
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/347288598362456660>)

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 "l411"//IBM Organisation ID
#defineDEVICE_TYPE "KishorIoT"//Device type mentioned in ibm watson IOT Platform
#defineDEVICE_ID "22022002"//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");
    }
}

```

```

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="";
}

```

(diagram.json)

```

{
    "version": 1,
    "author": "044 M. kishor",
    "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 and PubSubClient, and defines constants for an IBM Watson IoT connection. The right pane shows a simulation of the hardware, featuring an ESP32 microcontroller board connected to an HC-SR04 ultrasonic sensor and a red LED. The sensor's VCC is connected to the ESP32's VIN, its GND to GND, its TRIG pin to D5, and its ECHO pin to D18. The LED's anode is connected to D2 through a resistor, and its cathode to GND.

```

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  // credentials of IBM Accounts
6
7  #define ORG "1z4n1l"//IBM Organization ID
8  #define DEVICE_TYPE "KishorIoT"//Device type mentioned in ibm watson IOT Platform
9  #define DEVICE_ID "22022002"//Device ID mentioned in ibm watson IOT Platform
10 #define TOKEN "98765432"//Token
11 String data3;
12
13 // Customise the above values
14 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
15 char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event
16 perform and format in which data to be send
17 char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type
18 AND COMMAND IS TEST OF FORMAT STRING
19 char authMethod[] = "use-token-auth";// authentication method
20 char token[] = TOKEN;
21 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;// client id
22 WiFiClient wificlient; // creating the instance for wificlient
23 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined
24 client id by passing parameter like server id, port and wificlient credential
25 const int trigPin = 5;
26 const int echoPin = 18;
27 const int ledPin = 2;
28
29 long duration;
30 float distance;
31

```

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

Distance (cm): 399.92
Distance (cm): 399.96
Distance (cm): 399.94
Distance (cm): 399.98
Distance (cm): 399.94
Distance (cm): 399.92
Distance (cm): 399.94
```

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

The screenshot displays the IBM Watson IoT Platform interface. The browser address bar shows the URL: `https://tz4nll.internetofthings.ibmcloud.com/dashboard/devices/drilldown/KishorIoT:22022002?returnTo=/devices/browse`. The user is logged in as `kishormahendran6@gmail.com` with ID `tz4nll`.

The main heading is "Device Drilldown - 22022002". A left sidebar contains navigation options: Device Credentials (selected), Connection Information, Recent Events, State, Device Information, Metadata, Diagnostics, Connection Logs, and Device Actions.

The "Device Credentials" section contains the following information:

Organization ID	tz4nll
Device Type	KishorIoT
Device ID	22022002
Authentication Method	use-token-auth
Authentication Token	98765432

Below the table, a warning message states: "Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token." A link "Find out how to add these credentials to your device" is provided.

At the bottom right, it indicates "0 Simulations running".

The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with weather (29°C Cloudy), language (ENG), and date/time (12:11, 03-11-2022).

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar and an 'Add Device' button are also present. The main content area shows a list of devices, with one device selected and its details expanded. The 'Recent Events' tab is active, displaying a table of events.

Event	Value	Format	Last Received
event_1	{"Alert !!!":16}	json	a few seconds ago
event_1	{"Alert !!!":79}	json	a few seconds ago
event_1	{"Alert !!!":21}	json	a few seconds ago
event_1	{"Alert !!!":47}	json	a few seconds ago
event_1	{"Alert !!!":35}		

1 Simulation running

Conclusion :

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