

Assignment - 4

Assignment Date	20 October 2022
Student Name	Mr.B.Anandhakrishnan
Student Roll Number	621319106302
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/347296217300992594>)

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 "l3411"//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": "302.Anandhakrishnan.B",
  "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. 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* subscribtopic, byte* payload, unsigned int payloadLength);
5 // credentials of IBM Accounts
6
7 #define ORG "l2411" //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 subscribtopic[] = "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 wifi client
23 PubSubClient client(server, 1883, callback, wifiClient); // calling the predefined
24 client id by passing parameter like server id, port and wifi credential
25 const int trigpin = 5;
26 const int echopin = 18;
27 const int ledpin = 2;
28
29 long duration;
30 float distance;

```

On the right, the 'Simulation' tab is active, showing a virtual circuit. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's VIN pin (red), and its GND pin is connected to the ESP32's GND pin (black). The sensor's TRIG pin is connected to the ESP32's D5 pin (blue), and its ECHO pin is connected to the ESP32's D18 pin (green). A red LED is connected to the ESP32's D2 pin (red) and its anode is connected to the sensor's GND pin (black). The sensor's VCC pin is also connected to a 5V pin on the ESP32. The sensor's GND pin is connected to a GND pin on the ESP32. The sensor's TRIG pin is connected to the ESP32's D5 pin. The sensor's ECHO pin is connected to the ESP32's D18 pin. A red LED is connected to the ESP32's D2 pin and its anode is connected to the sensor's GND pin. The sensor's VCC pin is connected to a 5V pin on the ESP32. The sensor's GND pin is connected to a GND pin on the ESP32. The sensor's TRIG pin is connected to the ESP32's D5 pin. The sensor's ECHO pin is connected to the ESP32's D18 pin. A red LED is connected to the ESP32's D2 pin and its anode is connected to the sensor's GND pin.

```
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: `l24nll.internetofthings.ibmcloud.com/dashboard/devices/drilldown/KishorIoT:22022002?returnTo=/devices/browse`. The user is logged in as `kishormahendran6@gmail.com` with ID `l24nll`.

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

The "Device Credentials" section contains the following information:

Organization ID	l24nll
Device Type	KishorIoT
Device ID	22022002
Authentication Method	use-token-auth
Authentication Token	98765432

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." Below this, it says "Find out how to add these credentials to your device".

At the bottom right, a status box indicates "0 Simulations running".

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

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present, and an 'Add Device' button is in the top right. The main content area shows a list of devices. One device, with ID '22022002' and status 'Disconnected', is selected. Below the device list, a modal window titled 'Recent Events' is open, showing a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. Five events are listed, all with the value '{"Alert !!!":16}', '{"Alert !!!":79}', '{"Alert !!!":21}', '{"Alert !!!":47}', and '{"Alert !!!":35}' respectively, and the format 'json'. The last received time for all events is 'a few seconds ago'. A notification at the bottom right of the modal states '1 Simulation running'.

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}	json	a few seconds ago

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**.