

## Assignment-4

Assignment date	24 <sup>th</sup> October,2022
Student Name	Vinodhini R
Roll Number	2019504055
Maximum marks	2 Marks

### Question:

Write 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. Upload document with wokwi share link and images of ibm cloud.

### Code:

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

#define ECHO_GPIO 12
#define TRIGGER_GPIO 14
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"
Ultrasonic ultrasonic(14, 12);
int distance;

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

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

#define ORG "zg14jv" //IBM ORGANIZATION ID
#define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "VinoEsp32" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "B1b9SL0l9jSbiqLNGc" //Token
String data3;
float h, t;

//----- 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/command/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

void setup()// configuring the ESP32
{
    Serial.begin(115200);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    distance = ultrasonic.read(CM);
    if(distance < 100){
        Serial.print("Distance in CM: ");
        Serial.println(distance);
        PublishData(distance);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }

    }

    delay(1000);

}

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

void PublishData(float temp) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */
    String payload = "{\"Alert Distance\":\"";
    payload += temp;
    payload += "\"}";
}

```

```

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it successfully 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 definition 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++) {
        data3 += (char)payload[i];
    }
    Serial.println("data: "+ data3);
    if(data3=="lighton")
    {
        Serial.println(data3);
    }
    else
    {
        Serial.println(data3);
    }
    data3="";
}

```

Output:

The screenshot shows the Arduino IDE interface. On the left, the 'sketch.ino' file is open, displaying the code for an ESP32-based ultrasonic sensor project. The code includes a setup function for WiFi, a callback function for MQTT messages, and logic to publish distance data. On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller board connected to an HC-SR04 ultrasonic sensor module. The output window at the bottom right displays the following text:

```

Publish ok
Distance in CM: 24
Sending payload: {"Alert Distance:":24.00}
Publish ok
Distance in CM: 24
Sending payload: {"Alert Distance:":24.00}
Publish ok

```

sketch.ino

diagram.json

Ultrasonic.h

libraries.txt

Ultrasonic.cpp

Library Manager

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #define ECHO_GPIO 12
4 #define TRIGGER_GPIO 14
5 #define MAX_DISTANCE_CM 100 // Maximum of 5 meters
6 #include "Ultrasonic.h"
7 Ultrasonic ultrasonic(14, 12);
8 int distance;
9 void callback(char* subscribtopic, byte* payload, unsigned int
10 payloadLength);
11 //-----credentials of IBM Accounts-----
12 #define ORG "zg14jv" //IBM ORGANITION ID
13 #define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IOT Platform
14 #define DEVICE_ID "VinoEsp32" //Device ID mentioned in ibm watson IOT Platform
15 #define TOKEN "B1b9SL0I9jSbiqLNgC" //Token
16 String data3;
17 float h, t;
18 //----- Customise the above values -----
19 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
20 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform and format
21 char subscribtopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND COMMAND ID
22 char authMethod[] = "use-token-auth"; // authentication method
23 char token[] = TOKEN;
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
25 //-----
26 WiFiClient wificlient; // creating the instance for wificlient
27 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client id by pas
28 void setup() // configuring the ESP32
29 {
30   Serial.begin(115200);
31   delay(10);
32   Serial.println();
33   wificlient();

```

Simulation

00:11.020 99%

Editing Ultrasonic Distance Sensor

Distance: 71cm

Publish ok

Distance in CM: 74

Sending payload: {"Alert Distance":74.00}

Publish ok

Distance in CM: 74

Sending payload: {"Alert Distance":74.00}

Publish ok

IBM Watson IoT Platform

2019504055@student.annauniv.edu ID: zg14jv

Delete

1 item selected

Cancel

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
VinoEsp32	Disconnected	ESP32	Device	Oct 30, 2022 9:44 PM	

Identity

Device Information

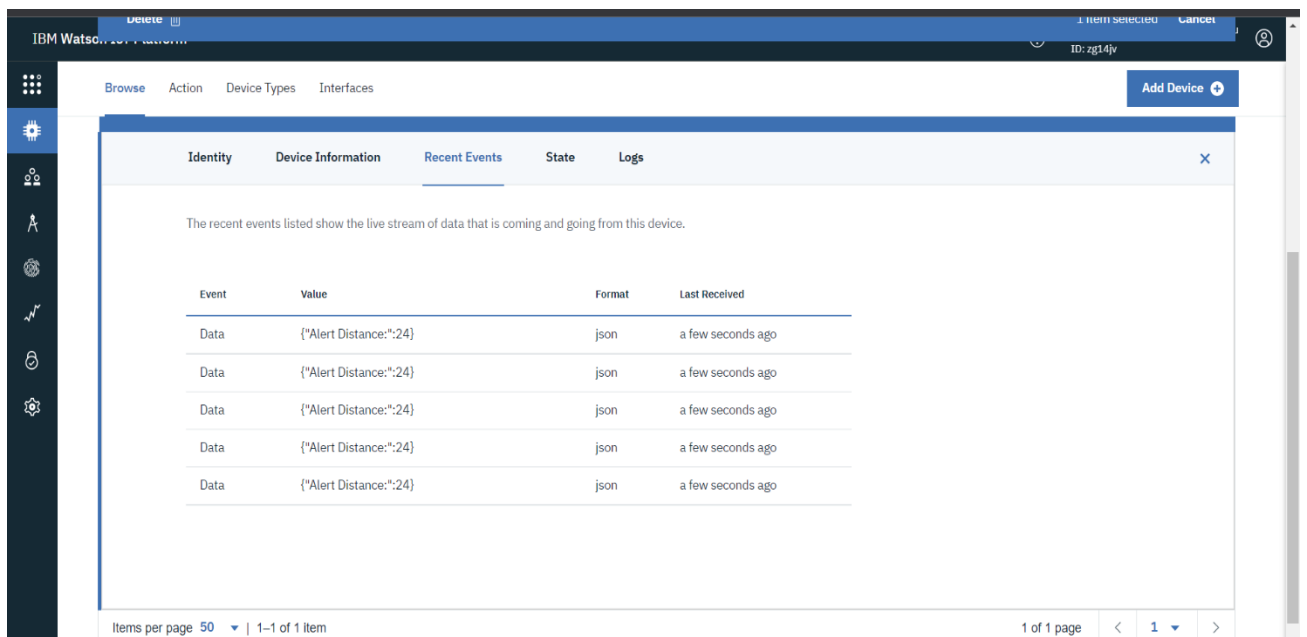
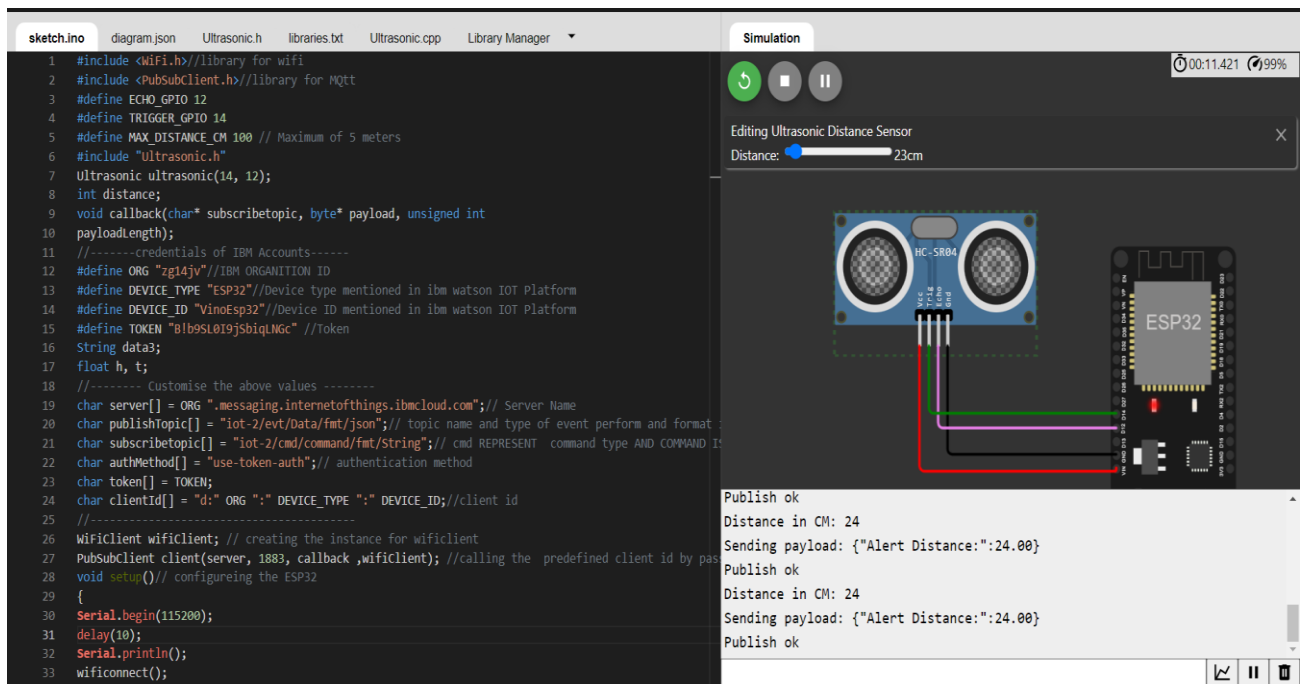
Recent Events

State

Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"Alert Distance":74}	json	a few seconds ago
Data	{"Alert Distance":74}	json	a few seconds ago
Data	{"Alert Distance":74}	json	a few seconds ago
Data	{"Alert Distance":74}	json	a few seconds ago
Data	{"Alert Distance":74}	json	a few seconds ago



## Project Link:

<https://wokwi.com/projects/346962165346861651>