

**Assignment -4**  
ESP32 Programming with IBM Cloud

Assignment Date	3 November 2022
Student Name	Siva N
Student Roll Number	717819F149
Maximum Marks	2 Marks

**Question-1:**

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.

**Solution:**

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

#define ECHO_GPIO 12
#define TRIGGER_GPIO 13
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"

Ultrasonic ultrasonic(13, 12); int
distance;

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

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

#define ORG "2melo1" //IBM ORGANITION ID
#define DEVICE_TYPE "Kruthika" //Device type mentioned in ibm Watson IOT
Platform
#define DEVICE_ID "0405" //Device ID mentioned in ibm watson IOT
Platform #define TOKEN "12345678" //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 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= "";
}

```

The screenshot displays the Wokwi IoT simulator interface. At the top, there are tabs for 'diagram.json', 'libraries.txt', 'Ultrasonic.h', 'Ultrasonic.cpp', and 'Library Manager'. The main workspace is divided into three sections:

- Code Editor (Left):** Contains C++ code for an ESP32 project using the Arduino IDE. The code includes the `Ultrasonic` library and implements a distance measurement function.
- Hardware Diagram (Right):** Shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor module. The sensor's VCC is connected to the ESP32's 5V pin, GND to GND, and the Trig and Echo pins to digital pins 4 and 5, respectively.
- Terminal (Bottom Right):** Displays the MQTT messages being sent by the device. The messages are: `Distance in CM: 28`, `Sending payload: {"Alert Distance:":28.00}`, `Publish ok`, `Distance in CM: 28`, `Sending payload: {"Alert Distance:":28.00}`, and `Publish ok`.

At the bottom of the interface, there is a table for managing devices:

Identity	Device Information	MQTT Events	State	Links
13	ESP32	MQTT Events	State	Links
14	ESP32	MQTT Events	State	Links
15	ESP32	MQTT Events	State	Links
16	ESP32	MQTT Events	State	Links
17	ESP32	MQTT Events	State	Links
18	ESP32	MQTT Events	State	Links
19	ESP32	MQTT Events	State	Links
20	ESP32	MQTT Events	State	Links
21	ESP32	MQTT Events	State	Links
22	ESP32	MQTT Events	State	Links
23	ESP32	MQTT Events	State	Links
24	ESP32	MQTT Events	State	Links
25	ESP32	MQTT Events	State	Links
26	ESP32	MQTT Events	State	Links
27	ESP32	MQTT Events	State	Links
28	ESP32	MQTT Events	State	Links
29	ESP32	MQTT Events	State	Links
30	ESP32	MQTT Events	State	Links
31	ESP32	MQTT Events	State	Links
32	ESP32	MQTT Events	State	Links
33	ESP32	MQTT Events	State	Links
34	ESP32	MQTT Events	State	Links
35	ESP32	MQTT Events	State	Links
36	ESP32	MQTT Events	State	Links
37	ESP32	MQTT Events	State	Links
38	ESP32	MQTT Events	State	Links
39	ESP32	MQTT Events	State	Links
40	ESP32	MQTT Events	State	Links
41	ESP32	MQTT Events	State	Links
42	ESP32	MQTT Events	State	Links
43	ESP32	MQTT Events	State	Links
44	ESP32	MQTT Events	State	Links
45	ESP32	MQTT Events	State	Links
46	ESP32	MQTT Events	State	Links
47	ESP32	MQTT Events	State	Links
48	ESP32	MQTT Events	State	Links
49	ESP32	MQTT Events	State	Links
50	ESP32	MQTT Events	State	Links
51	ESP32	MQTT Events	State	Links
52	ESP32	MQTT Events	State	Links
53	ESP32	MQTT Events	State	Links
54	ESP32	MQTT Events	State	Links
55	ESP32	MQTT Events	State	Links
56	ESP32	MQTT Events	State	Links
57	ESP32	MQTT Events	State	Links
58	ESP32	MQTT Events	State	Links
59	ESP32	MQTT Events	State	Links
60	ESP32	MQTT Events	State	Links
61	ESP32	MQTT Events	State	Links
62	ESP32	MQTT Events	State	Links
63	ESP32	MQTT Events	State	Links
64	ESP32	MQTT Events	State	Links
65	ESP32	MQTT Events	State	Links
66	ESP32	MQTT Events	State	Links
67	ESP32	MQTT Events	State	Links
68	ESP32	MQTT Events	State	Links
69	ESP32	MQTT Events	State	Links
70	ESP32	MQTT Events	State	Links
71	ESP32	MQTT Events	State	Links
72	ESP32	MQTT Events	State	Links
73	ESP32	MQTT Events	State	Links
74	ESP32	MQTT Events	State	Links
75	ESP32	MQTT Events	State	Links
76	ESP32	MQTT Events	State	Links
77	ESP32	MQTT Events	State	Links
78	ESP32	MQTT Events	State	Links
79	ESP32	MQTT Events	State	Links
80	ESP32	MQTT Events	State	Links
81	ESP32	MQTT Events	State	Links
82	ESP32	MQTT Events	State	Links
83	ESP32	MQTT Events	State	Links
84	ESP32	MQTT Events	State	Links
85	ESP32	MQTT Events	State	Links
86	ESP32	MQTT Events	State	Links
87	ESP32	MQTT Events	State	Links
88	ESP32	MQTT Events	State	Links
89	ESP32	MQTT Events	State	Links
90	ESP32	MQTT Events	State	Links
91	ESP32	MQTT Events	State	Links
92	ESP32	MQTT Events	State	Links
93	ESP32	MQTT Events	State	Links
94	ESP32	MQTT Events	State	Links
95	ESP32	MQTT Events	State	Links
96	ESP32	MQTT Events	State	Links
97	ESP32	MQTT Events	State	Links
98	ESP32	MQTT Events	State	Links
99	ESP32	MQTT Events	State	Links
100	ESP32	MQTT Events	State	Links