

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

Assignment Date	3 November 2022
Student Name	Darshna K
Student Roll Number	717819P210
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, which is used for developing and testing IoT applications on a virtual platform.

**Top Section: Code Editor and Simulation**

- Code Editor:** Contains a C++ program for an ESP32 microcontroller. The program uses the `Ultrasonic` library to interface with an HC-SR04 ultrasonic sensor. It defines a constant `pin = 4` and a `Ultrasonic` object `ultrasonic`. The `void setup()` function initializes the serial port at `115200` baud and the sensor at `pin`. The `void loop()` function reads the distance in centimeters using `ultrasonic.read()` and publishes it to a MQTT broker with the topic `"Alert Distance"` using `mqtt.publish()`. The program includes the `Ultrasonic` library and the `MQTT` library.
- Simulation:** A 3D model of the ESP32 microcontroller and the HC-SR04 ultrasonic sensor is shown. The sensor is connected to the ESP32 via four wires (red, blue, yellow, and black).
- Terminal Output:** The terminal window shows the following output:
 

```
Distance in CM: 28
Sending payload: {"Alert Distance":28.00}
Publish ok
Distance in CM: 28
Sending payload: {"Alert Distance":28.00}
Publish ok
```

**Bottom Section: Device Information**

The bottom section displays the 'Device Information' tab, which provides details about the simulated device. The table below shows the information for the device:

Device ID	Device Name	Device Type	Device Status	Device Location
ESP32-0000000000000000	ESP32-0000000000000000	ESP32	Online	0.00, 0.00, 0.00