

## ASSIGNMENT 4

TEAM ID	PNT2022TMID20364
---------	------------------

Question 1:

Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events. Upload document with wokwi share link and images of IBM cloud.

**Wokwi Project Link:** <https://wokwi.com/projects/348100832051331666>

CODE:

```
#include<WiFi.h>
#include<PubSubClient.h>
#define TRIGGER 2
#define ECHO 15
#define sound_speed 0.034
int distance;
void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);

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

#define ORG "0va7j8"
#define DEVICE_TYPE "ultasonic"
#define DEVICE_ID "123456"
#define TOKEN "123456789"
String data3;
//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribtopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

//-----
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);
void setup()
{
  Serial.begin(115200);
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
  delay(10);
  Serial.println();
}
```

```

    wificonnect();
    mqttconnect();
}

void loop()
{
    digitalWrite(TRIGGER, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGGER, LOW);

    int duration=pulseIn(ECHO,HIGH);
    distance=(duration*sound_speed)/2;
    Serial.print(distance);
    Serial.println(" cms.");
    if(distance<100){
        PublishData(distance);
    }
    delay(1000);
    if(!client.loop()) {
        mqttconnect();
    }
}
/*.....retrieving to Cloud.....*/

void PublishData(int d) {
    mqttconnect();
    String payload = "{\"message\":\"alert\"";
    payload += "}";
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if (client.publish(publishTopic, (char*) payload.c_str())) {
        Serial.println("Publish ok");
    } 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()
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

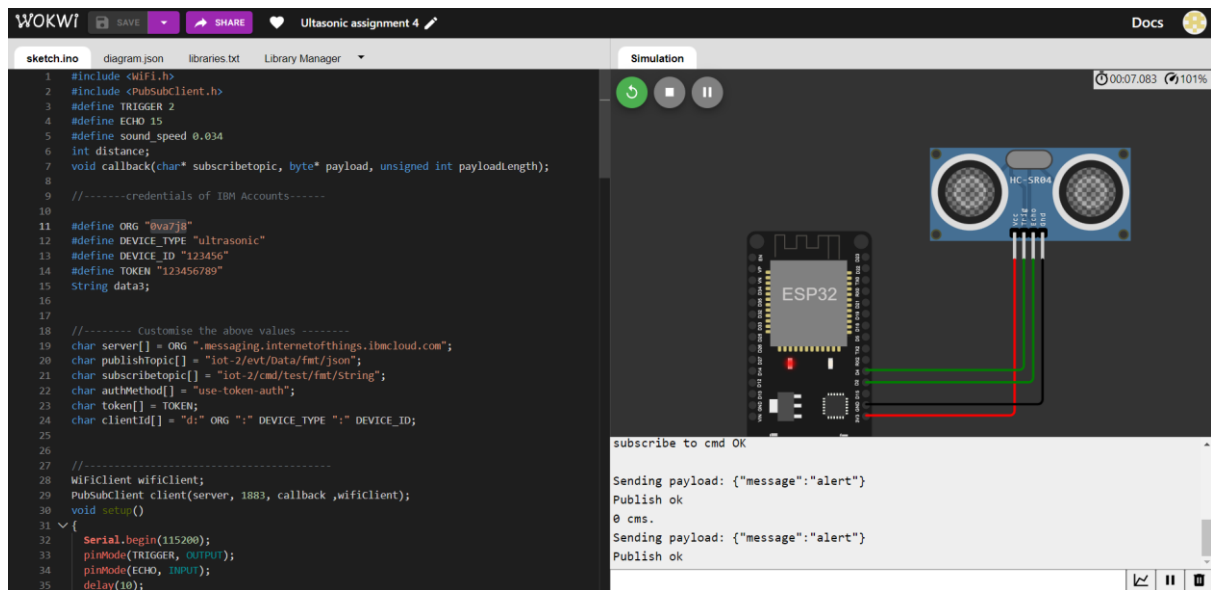
void initManagedDevice() {
    if (client.subscribe(subscribetopic)) {
        Serial.println((subscribetopic));
        Serial.println("subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}

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);

    data3="";
}

```

# Wokwi Platform Coding and Circuit Design



The screenshot displays the Wokwi platform interface. On the left, the 'sketch.ino' editor shows an Arduino sketch for an ultrasonic sensor connected to an ESP32. The code includes headers for WiFi and PubSubClient, defines pins and constants, and sets up a callback function. It configures the ESP32 as an IoT device on the IBM IoT Platform, subscribes to a command topic, and publishes an alert message. The right side features a 'Simulation' window with a visual representation of the ESP32 and HC-SR04 sensor connected by wires. Below the simulation, a console window shows the execution log, including the successful subscription to the command topic and the sending of an alert payload.

```
1 #include <wifi.h>
2 #include <PubSubClient.h>
3 #define TRIGGER 2
4 #define ECHO 15
5 #define sound_speed 0.034
6 int distance;
7 void callback(char* subscribetopic, byte* payload, unsigned int payloadlength);
8
9 //-----credentials of IBM Accounts-----
10
11 #define ORG "00a7j8"
12 #define DEVICE_TYPE "ultrasonic"
13 #define DEVICE_ID "123456"
14 #define TOKEN "123456789"
15 String data;
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
19 char publishTopic[] = "iot-2/evt/data/fmt/json";
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
21 char authMethod[] = "use-token-auth";
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
24
25 //-----
26
27 //-----
28 WiFiClient wifiClient;
29 PubSubClient client(server, 1883, callback, wifiClient);
30 void setup()
31 {
32   Serial.begin(115200);
33   pinMode(TRIGGER, OUTPUT);
34   pinMode(ECHO, INPUT);
35   delay(10);
36 }
```

subscribe to cmd OK

Sending payload: {"message":"alert"}  
Publish ok  
0 cms.  
Sending payload: {"message":"alert"}  
Publish ok

## IBM IoT Platform Device Recent Events