

# Assignment - 4

## Problem Statement:

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.

## Solution:

Wokwi Link: <https://wokwi.com/projects/347233654068478547>

## Circuit Diagram with output:

The Wokwi simulation interface displays the following code in the sketch.ino file:

```
1 #include <wifi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int
4 payloadLength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "z0kljz"//IBM ORGANIZATION ID
7 #define DEVICE_TYPE "ESP32_Controller"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "BME280_Sensor"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "YD-b2q)h0)5n7yef" //Token
10 String data;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wifiConnect();
29   mqttConnect();
30 }
31 void loop()
32 {
33   digitalWrite(trigPin, LOW);
34   delayMicroseconds(2);
35   digitalWrite(trigPin, HIGH);
```

The circuit diagram shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's Trig pin is connected to the ESP32's pin 5, and its Echo pin is connected to the ESP32's pin 18.

The simulation output shows the following messages:

```
ALERT!!
Sending payload: {"Distance":56.97,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 56.97
ALERT!!
Sending payload: {"Distance":56.97,"ALERT!!":"Distance less than 100cms"}
Publish ok
```

The IBM Watson IoT Platform dashboard shows the following details for the device BME280\_Sensor:

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
BME280_Sensor	Connected	ESP32_Controller	Device	31 Oct 2022 11:35	

The Recent Events tab shows the following events:

Event	Value	Format	Last Received
Data	{"Distance":72.96,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":72.96,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":72.96,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":56.97,"ALERT!!":"Distance less than ...	json	2 minutes ago
Data	{"Distance":56.97,"ALERT!!":"Distance less than ...	json	2 minutes ago

## Program Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

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

#define ORG "z0kljz"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_Controller"//Device type mentioned in ibm watson
IOT Platform
#define DEVICE_ID "BME280_Sensor"//Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "YD-b2q)ho)JSn7y+ef" //Token

String data3;

//-----Customise the above values-----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "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);

const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;

void setup() {
    Serial.begin(115200);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    wificonnect();
    mqttconnect();
}

void loop()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigpin on HIGH state for 10 microseconds
```

```

digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, Return the sound wave travel in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculate the distance in centimeters
distance = duration * SOUND_SPEED/2;

Serial.print("Distance (cm): ");
Serial.println(distance);
//Checking of the status
if(distance<100)
{
    Serial.println("ALERT!!");
    delay(1000);
    PublishData(distance);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}
delay(1000);
}

/*-----Retrieving to cloud-----*/
void PublishData(float dist)
{
    mqttconnect();
    /*
    Creating the string in the form of JSON to update the data to IBM
    Cloud
    */
    String payload = "{\"Distance\":\"";
    payload += dist;
    payload += "cm,\"ALERT!!\":\"\"Distance less than 100cms\"";
    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="";
}

```