

Assignment-4

Name	J. Jeffin Rohith
Roll Number	2019504030

Problem Statement:

Write code and connections in Wokwi for ultrasonic sensor. Whenever distance is less than 100 cm send "alert" to IBM cloud and display in device recent events.

Source Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
#define ORG "vkk3lh"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP-32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "2019504030"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "9876543210" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/distance/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);
#define ECHO_PIN 12
#define TRIG_PIN 13
#define led 2
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    pinMode(led, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    wificonnect();
    mqttconnect();
}
float readDistanceCM() {
    digitalWrite(TRIG_PIN, LOW); // Clear the trigger
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10
microseconds
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    int duration = pulseIn(ECHO_PIN, HIGH);
    //Serial.println(duration);
}
```

```

    //duration = pulseIn(ECHO_PIN, HIGH);
    return duration * 0.017;
    //Serial.println(duration);
}
void loop() {
    float distance = readDistanceCM();
    //Serial.println(distance);
    bool isNearby = distance < 100;
    digitalWrite(led, isNearby);
    Serial.print("Measured distance: ");
    Serial.println(distance);
    if (distance < 100) {
        PublishData2(distance);
    } else {
        PublishData1(distance);
    }
    //PublishData(distance);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
    //delay(2000);
}
void PublishData1(float dist) {
    mqttconnect();
    String payload = "{\"distance\":\"";
    payload += dist;
    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 PublishData2(float dist) {
    mqttconnect();
    String payload = "{\"ALERT\":\"";
    payload += dist;
    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 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++) {
    data3 += (char)payload[i];
  }
  Serial.println("data:" + data3);
  if (data3 == "lighton") {
    Serial.println(data3);
    digitalWrite(led, HIGH);
  } else {
    Serial.println(data3);
    digitalWrite(led, LOW);
  }
  data3 = "";
}

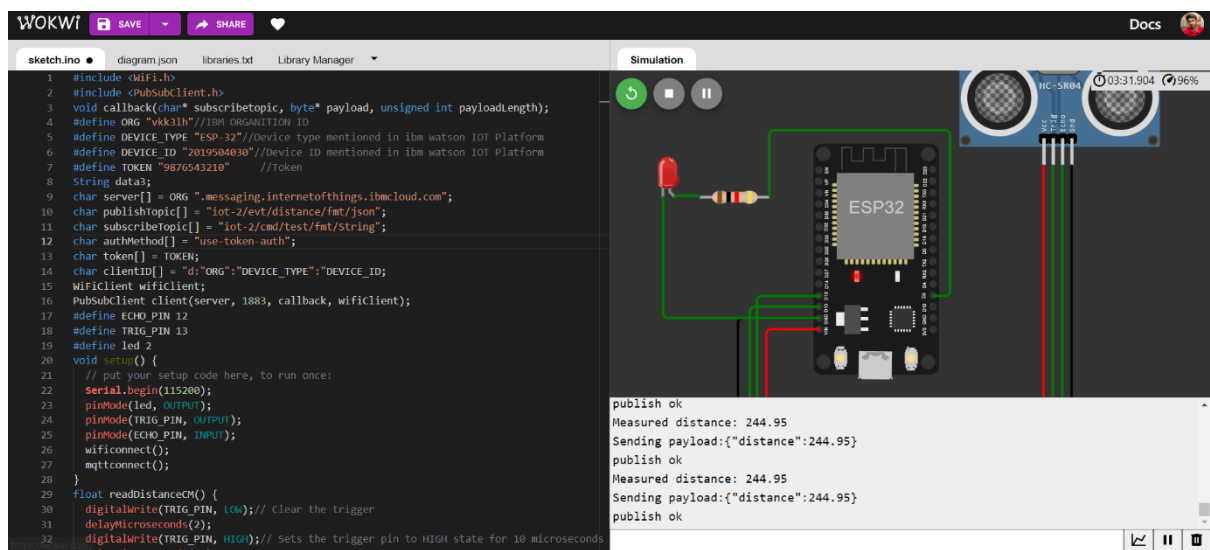
```

Wokwi Link:

<https://wokwi.com/projects/347143134975623762>

Output:

Normal Case:



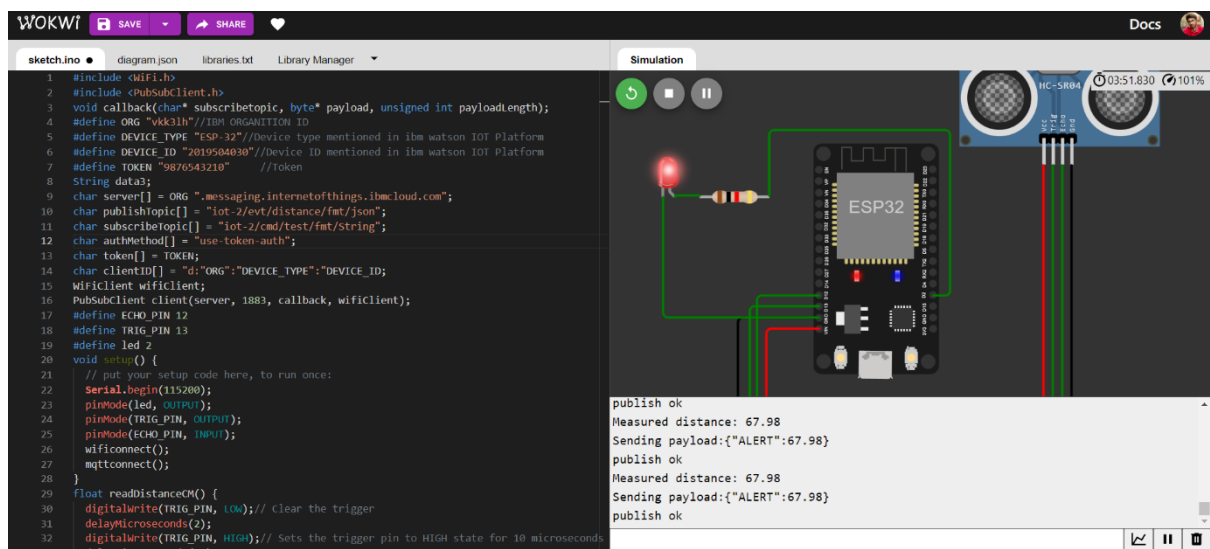
The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadlength);
4 #define ORG "vkk3lh"//IBM ORGANITION ID
5 #define DEVICE_TYPE "ESP-32"//Device type mentioned in ibm watson IOT Platform
6 #define DEVICE_ID "2019504030"//Device ID mentioned in ibm watson IOT Platform
7 #define TOKEN "9876543210" //token
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/distance/fmt/json";
11 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:"ORG":DEVICE_TYPE":DEVICE_ID;
15 WiFiClient wificlient;
16 PubSubClient client(server, 1883, callback, wificlient);
17 #define ECHO_PIN 12
18 #define TRIG_PIN 13
19 #define led 2
20 void setup() {
21 // put your setup code here, to run once:
22 Serial.begin(115200);
23 pinMode(led, OUTPUT);
24 pinMode(TRIG_PIN, OUTPUT);
25 pinMode(ECHO_PIN, INPUT);
26 wificlient.setTimeout(1000);
27 mqttconnect();
28 }
29 float readDistanceCM() {
30 digitalWrite(TRIG_PIN, LOW); // Clear the trigger
31 delayMicroseconds(2);
32 digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10 microseconds
33 delayMicroseconds(10);
34 }
```

On the right, the 'Simulation' window shows a virtual circuit with an ESP32 module, a red LED, and a HC-SR04 ultrasonic sensor. The console output shows the following sequence of events:

```
publish ok
Measured distance: 244.95
Sending payload:{"distance":244.95}
publish ok
Measured distance: 244.95
Sending payload:{"distance":244.95}
publish ok
```

Alert Case:



The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadlength);
4 #define ORG "vkk3lh"//IBM ORGANITION ID
5 #define DEVICE_TYPE "ESP-32"//Device type mentioned in ibm watson IOT Platform
6 #define DEVICE_ID "2019504030"//Device ID mentioned in ibm watson IOT Platform
7 #define TOKEN "9876543210" //token
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/distance/fmt/json";
11 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:"ORG":DEVICE_TYPE":DEVICE_ID;
15 WiFiClient wificlient;
16 PubSubClient client(server, 1883, callback, wificlient);
17 #define ECHO_PIN 12
18 #define TRIG_PIN 13
19 #define led 2
20 void setup() {
21 // put your setup code here, to run once:
22 Serial.begin(115200);
23 pinMode(led, OUTPUT);
24 pinMode(TRIG_PIN, OUTPUT);
25 pinMode(ECHO_PIN, INPUT);
26 wificlient.setTimeout(1000);
27 mqttconnect();
28 }
29 float readDistanceCM() {
30 digitalWrite(TRIG_PIN, LOW); // Clear the trigger
31 delayMicroseconds(2);
32 digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10 microseconds
33 delayMicroseconds(10);
34 }
```

On the right, the 'Simulation' window shows the same virtual circuit as the normal case. The console output shows the following sequence of events:

```
publish ok
Measured distance: 67.98
Sending payload:{"ALERT":67.98}
publish ok
Measured distance: 67.98
Sending payload:{"ALERT":67.98}
publish ok
```

IBM Cloud Storage:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A user profile for 'jwardex@gmail.com' with ID 'vkk3lh' is visible in the top right. The main content area shows a device summary for '2019504030' (ESP-32) with a 'Connected' status and a timestamp of '1 Nov 2022 21:52'. Below this, the 'Recent Events' tab is active, displaying a table of live data events.

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

Event	Value	Format	Last Received
distance	{"ALERT":67.98}	json	a few seconds ago
distance	{"ALERT":67.98}	json	a few seconds ago
distance	{"ALERT":67.98}	json	a few seconds ago
distance	{"ALERT":67.97}	json	a few seconds ago
distance	{"ALERT":67.98}	json	a few seconds ago

At the bottom, there is a pagination control showing 'Items per page: 50' and '1 of 1 page'.