

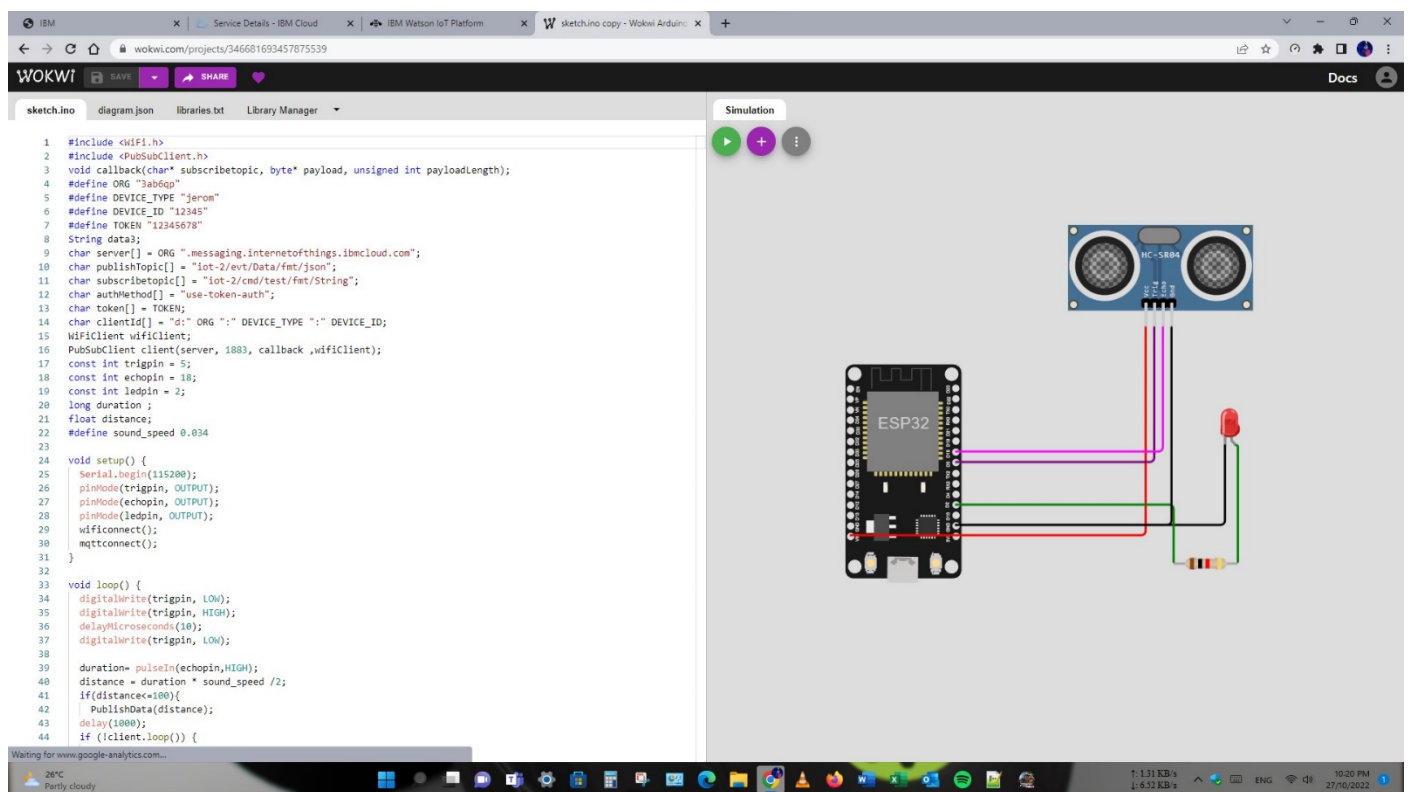
ASSIGNMENT – 04

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.

Wowki Project Link: <https://wokwi.com/projects/346681693457875539>

Step 1: Completed Setup to build Circuit



Step 2: Output in WOWKI

a) When distance is below 100 cms

The screenshot shows the WOKWI simulation environment. On the left, the code in `sketch.ino` is displayed. It includes headers for `WiFi` and `PubSubClient`, defines an MQTT broker, device ID, and token, and sets up pins for an ultrasonic sensor and an LED. The `loop` function reads the sensor distance and publishes an alert if it's below 100cm.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "3ab6qp"
5 #define DEVICE_TYPE "jerom"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/Data/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 const int trigpin = 5;
18 const int echopin = 18;
19 const int ledpin = 2;
20 long duration;
21 float distance;
22 #define sound_speed 0.034
23
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigpin, OUTPUT);
27   pinMode(echopin, OUTPUT);
28   pinMode(ledpin, OUTPUT);
29   wifiConnect();
30   mqttConnect();
31 }
32
33 void loop() {
34   digitalWrite(trigpin, LOW);
35   digitalWrite(trigpin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigpin, LOW);
38
39   duration = pulseIn(echopin, HIGH);
40   distance = duration * sound_speed / 2;
41   if (distance < 100) {
42     PublishData(distance);
43     delay(1000);
44     if (!client.loop()) {
45       mqttConnect();
46     }
47   }
48 }
```

On the right, the simulation window shows an ESP32 connected to an ultrasonic sensor and an LED. The sensor's distance is 84cm. The console output shows the device connecting to the MQTT broker and publishing an alert.

Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 3ab6qp.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Sending payload: {"ALERT....!! ": 83.95}
Publish ok
ALERT....!!
83.95
Sending payload: {"ALERT....!! ": 83.98}
Publish ok

b) When Distance is above 100 cms

The screenshot shows the WOKWI simulation environment. The code in `sketch.ino` is the same as in the previous screenshot, but the `loop` function only publishes an alert if the distance is below 100cm. In this simulation, the distance is 147cm, so no alert is published.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "3ab6qp"
5 #define DEVICE_TYPE "jerom"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/Data/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 const int trigpin = 5;
18 const int echopin = 18;
19 const int ledpin = 2;
20 long duration;
21 float distance;
22 #define sound_speed 0.034
23
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigpin, OUTPUT);
27   pinMode(echopin, OUTPUT);
28   pinMode(ledpin, OUTPUT);
29   wifiConnect();
30   mqttConnect();
31 }
32
33 void loop() {
34   digitalWrite(trigpin, LOW);
35   digitalWrite(trigpin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigpin, LOW);
38
39   duration = pulseIn(echopin, HIGH);
40   distance = duration * sound_speed / 2;
41   if (distance < 100) {
42     PublishData(distance);
43     delay(1000);
44     if (!client.loop()) {
45       mqttConnect();
46     }
47   }
48 }
```

The simulation window shows the ESP32 connected to the ultrasonic sensor and the LED. The sensor's distance is 147cm. The console output shows the device connecting to the MQTT broker and publishing the distance.

Publish ok
ALERT....!!
63.97
Sending payload: {"ALERT....!! ": 63.97}
Publish ok
ALERT....!!
63.97
Sending payload: {"ALERT....!! ": 63.97}
Publish ok
ALERT....!!
63.97
Sending payload: {"ALERT....!! ": 63.97}
Publish ok
ALERT....!!
63.97

Step 3: Output in IBM CLOUD (Watson Platform) recent events

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area displays the details for a device with ID 12345, which is in a 'Connected' state. The 'Recent Events' tab is active, showing a table of events. The table has the following data:

Event	Value	Format	Last Received
Data	["ALERT...!!":63.97]	json	a few seconds ago
Data	["ALERT...!!":63.97]	json	a few seconds ago
Data	["ALERT...!!":63.97]	json	a few seconds ago
Data	["ALERT...!!":62.95]	json	a few seconds ago

At the bottom of the dashboard, there is a status bar indicating '0 Simulations running'.

Program Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
#define ORG "3ab6qp"
#define DEVICE_TYPE "jerom"
#define DEVICE_ID "12345"
#define TOKEN "12345678"
String data3;
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;
const int ledpin = 2;
long duration ;
float distance;
#define sound_speed 0.034

void setup() {
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
```

```

pinMode(echopin, OUTPUT);
pinMode(ledpin, OUTPUT);
wificonnect();
mqttconnect();
}

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

    duration= pulseIn(echopin,HIGH);
    distance = duration * sound_speed /2;
    if(distance<=100){
        PublishData(distance);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }
        digitalWrite(ledpin, HIGH);
        Serial.println("ALERT...!!!");
        Serial.println(distance);
    }
    else
    {
        digitalWrite(ledpin, LOW);
    }
    delay(10); // this speeds up the simulation
}

void PublishData(float distance) {
    mqttconnect();

    String payload = "{\"ALERT...!! \": ";
    payload += distance;
    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);
    if(data3=="lighton")
    {
        Serial.println(data3);
    }
    else
    {
        Serial.println(data3);
    }
    data3="";
}

```