

# ASSIGNMENT 4

## Ultrasonic sensor simulation in Wokwi

### Question :

Write a code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100cms send an “Alert” to IBM cloud and display in the device recent events.

### Code:

### Sketch.ino:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "91xobn"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32PROJECT"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "ESP32"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "ESP32PROJECT" //Token
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;
#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);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;
  Serial.print("Distance (cm): ");
  Serial.println(distance);
  if(distance<100)
  {
    Serial.println("ALERT!!");
    delay(1000);
    PublishData(distance);
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
  }
  delay(1000);
}
void PublishData(float dist) {

```

```

mqttconnect();
String payload = "{\"Distance\": ";
payload += dist;
payload += ", \"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="";
}

```

diagram.json:

```

{
  "version": 1,
  "author": "001 Abika A",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 23.33, "left": -106, "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -15.04, "left": 86.5, "attrs": {} }
  ],
  "connections": [

```

```
[ "esp:TX0", "$serialMonitor:RX", "", [ ] ],  
[ "esp:RX0", "$serialMonitor:TX", "", [ ] ],  
[ "ultrasonic1:VCC", "esp:VIN", "red", [ "v168.58", "h-279.11", "v-66" ] ],  
[ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ],  
[ "ultrasonic1:TRIG", "esp:D5", "green", [ "v0" ] ],  
[ "ultrasonic1:ECHO", "esp:D18", "green", [ "v0" ] ]  
]  
}
```

#### **libraries.txt:**

```
# Wokwi Library List  
# See https://docs.wokwi.com/guides/libraries
```

PubSubClient

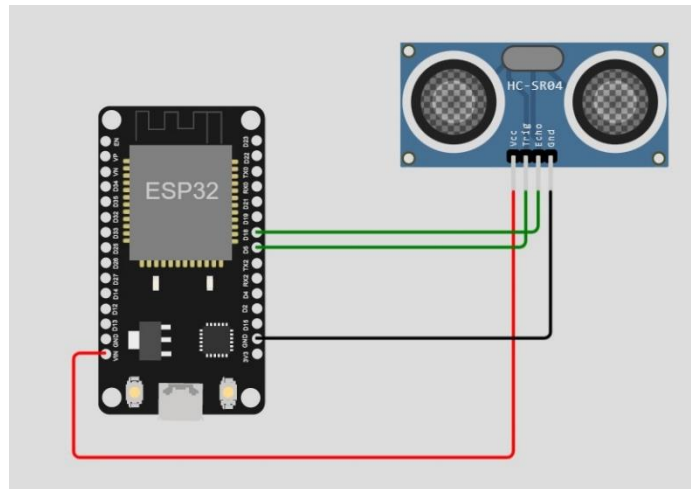
#### **Library Manager:**

Installed Libraries:  
PubSubClient

#### **Wokwi simulation link:**

<https://wokwi.com/projects/347328550626394707>

### Circuit Diagram:



### Output:

#### Wokwi output:

```
Connecting to ...  
WiFi connected  
IP address:  
10.10.0.2  
Reconnecting client to 9lxobn.messaging.internetofthings.ibmcloud.com  
iot-2/cmd/test/fmt/String  
subscribe to cmd OK  
  
Distance (cm): 399.96  
Distance (cm): 399.92  
Distance (cm): 399.94  
Distance (cm): 399.98  
Distance (cm): 399.94  
Distance (cm): 399.94  
Distance (cm): 399.94  
Distance (cm): 399.94
```

Distance (cm): 399.94  
Distance (cm): 247.96  
Distance (cm): 219.98  
Distance (cm): 171.97  
Distance (cm): 120.96  
Distance (cm): 84.95  
ALERT!!  
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 78.98  
ALERT!!  
Sending payload: {"Distance":78.98,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 78.98  
ALERT!!  
Sending payload: {"Distance":78.98,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 89.98  
ALERT!!  
Sending payload: {"Distance":89.98,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 84.97  
ALERT!!  
Sending payload: {"Distance":84.97,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 86.96  
ALERT!!  
Sending payload: {"Distance":86.96,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 84.95  
ALERT!!  
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 84.95  
ALERT!!  
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 84.95  
ALERT!!

```
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 84.95
ALERT!!
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}
Publish ok
```

## Wokwi Output Screen:

The image shows the Wokwi IDE interface with a C++ sketch for an ESP32 connected to an HC-SR04 ultrasonic sensor. The sketch configures the sensor pins and publishes distance data to a MQTT topic. The simulation window shows the sensor's current distance as 85cm.

**Sketch Code:**

```
1 #include <Wifi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic, byte* payload, unsigned int
4   payloadLength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "9lxobn"//IBM ORGANITION ID
7 #define DEVICE_TYPE "ESP32PROJECT"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "ESP32"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "ESP32PROJECT" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribetopic[] = "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   wificlient.connect();
29   mqttconnect();
30 }
31 void loop()
32 {
33   digitalWrite(trigPin, LOW);
34   delayMicroseconds(2);
35   digitalWrite(trigPin, HIGH);
```

**Simulation Window:**

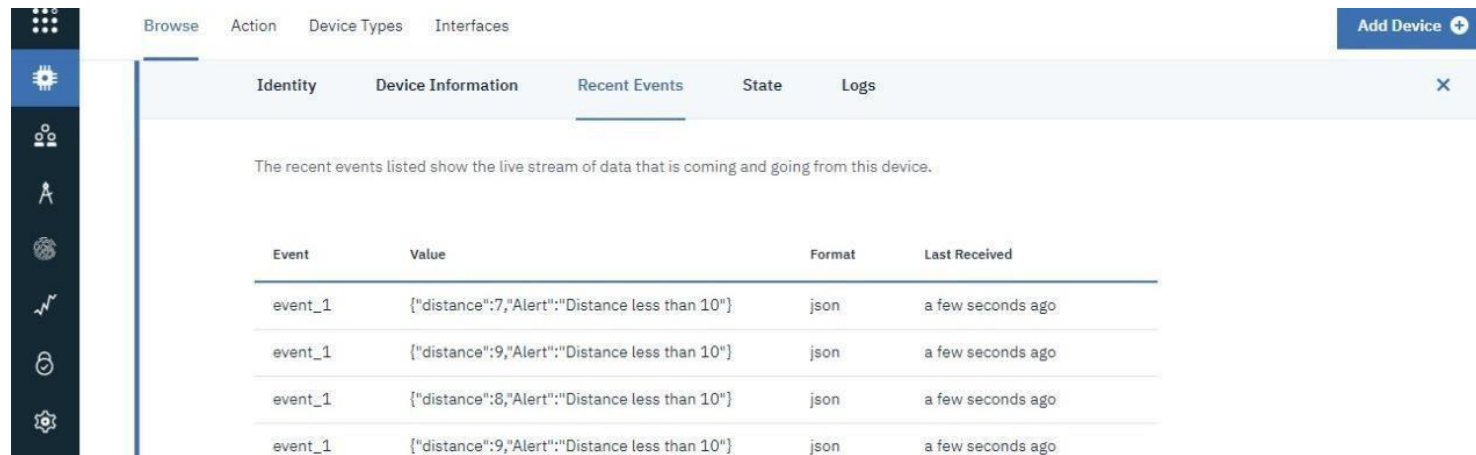
Editing Ultrasonic Distance Sensor  
Distance: 85cm

**Output Log:**

```
Publish ok
Distance (cm): 84.95
ALERT!!
Sending payload: {"Distance":84.95,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 85.00
ALERT!!
```



## IBM cloud output:



The screenshot displays the IBM Cloud IoT Platform interface. On the left is a dark sidebar with icons for navigation. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces', along with an 'Add Device +' button. The main content area is titled 'Recent Events' and contains a table of device events.

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

Event	Value	Format	Last Received
event_1	{"distance":7,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":8,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"Distance less than 10"}	json	a few seconds ago