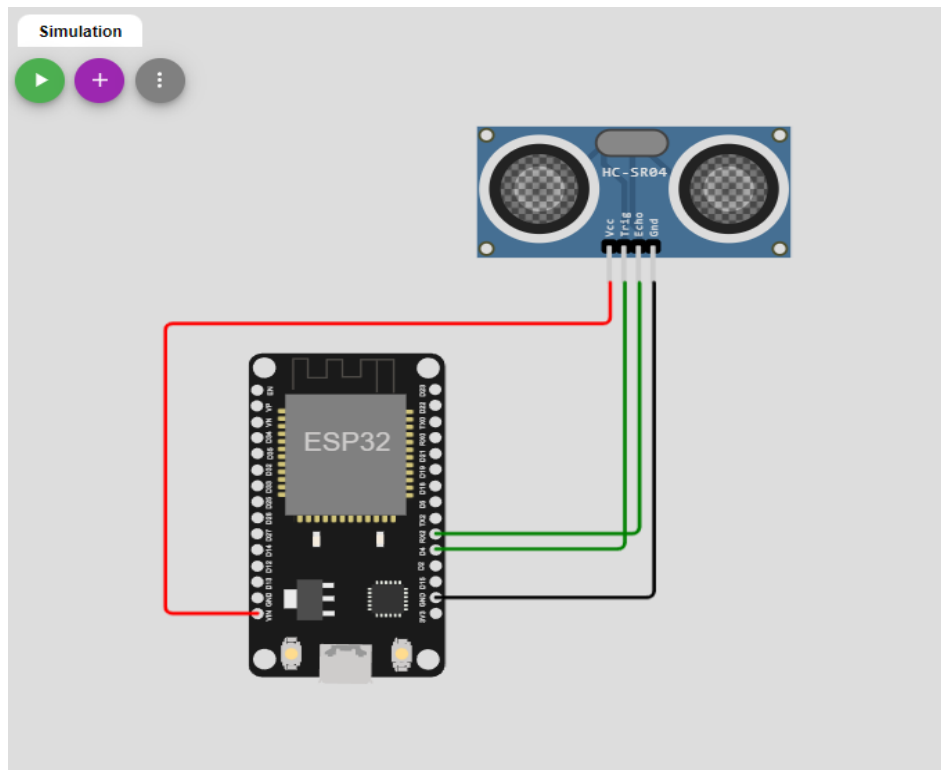


## Assignment -4

1. Write code and connections in wowki for the ultrasonic sensor. Whenever the distance is less than 100cms send an alert to ibm cloud and display in the device recent events,

**Circuit Diagram:**



**code:**

```
print("Hello, ESP32!")

#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;

String data3;

#define ORG "uyhprk"

#define DEVICE_TYPE "UltraSonicSensor"
```

```
#define DEVICE_ID "456789"

#define TOKEN "Y0*7&wous&90TR2?f8"

#define speed 0.034

#define led 14

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Jijitha AL/fmt/json";

char topic[] = "iot-2/cmd/led/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

PubSubClient client(server, 1883, wifiClient);

const int trigpin=5;

const int echopin=18;

String command;

String data="";

long duration;

float dist;

void setup()

{

    Serial.begin(115200);

    pinMode(led, OUTPUT);

    pinMode(trigpin, OUTPUT);

    pinMode(echopin, INPUT);

    wifiConnect();
```

```
mqttConnect();  
}  
  
void loop() {  
  
  bool isNearby = dist < 100;  
  
  digitalWrite(led, isNearby);  
  
  publishData();  
  
  delay(500);  
  
  if (!client.loop()) {  
  
    mqttConnect();  
  
  }  
}  
  
void wifiConnect() {  
  
  Serial.print("Connecting to "); Serial.print("Wifi");  
  
  WiFi.begin("Wokwi-GUEST", "", 6);  
  
  while (WiFi.status() != WL_CONNECTED) {  
  
    delay(500);  
  
    Serial.print(".");  
  
  }  
  
  Serial.print("WiFi connected, IP address: "); Serial.println(WiFi.localIP());  
}  
  
void mqttConnect() {  
  
  if (!client.connected()) {  
  
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);  
  
    while (!client.connect(clientId, authMethod, token)) {
```

```
Serial.print(".");  
  
delay(500);  
  
}  
  
initManagedDevice();  
  
Serial.println();  
  
}  
  
}  
  
void initManagedDevice() {  
    if (client.subscribe(topic)) {  
        // Serial.println(client.subscribe(topic));  
        Serial.println("IBM subscribe to cmd OK");  
    } else {  
        Serial.println("subscribe to cmd FAILED");  
    }  
}  
  
void publishData()  
{  
    digitalWrite(trigpin,LOW);  
    digitalWrite(trigpin,HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigpin,LOW);  
    duration=pulseIn(echopin,HIGH);  
    dist=duration*speed/2;  
    if(dist<100){
```

```
String payload = "{\"Alert Distance\":\"";
payload += dist;
payload += "\"}";
Serial.print("\n");
Serial.print("Sending payload: ");
Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
}
}

if(dist>100){
    String payload = "{\"Distance\":\"";
    payload += dist;
    payload += "\"}";
    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if(client.publish(publishTopic, (char*) payload.c_str())) {
        Serial.println("Publish OK");
    }else {
        Serial.println("Publish FAILED");
    }
}
}
```

## Wowki Stimulation Link:

<https://wokwi.com/projects/348018607895085652>

## Output:

1) When the distance is greater than 100 cms

The screenshot shows the Wokwi web-based IDE. On the left, the code for an ESP32 is displayed. It includes an `initManagedDevice` function that subscribes to a topic. The main loop checks the distance from an ultrasonic sensor. If the distance is greater than 100cm, it constructs a JSON payload: `{"Alert Distance": "147.93"}` and sends it via MQTT. The simulation window on the right shows the sensor's distance being updated to 148cm and the alert message being published.

```
59 Serial.println();
60 }
61
62 void initManagedDevice() {
63   if (client.subscribe(topic)) {
64     // Serial.println(client.subscribe(topic));
65     Serial.println("IBM subscribe to cmd OK");
66   } else {
67     Serial.println("subscribe to cmd FAILED");
68   }
69 }
70 void publishData()
71 {
72   digitalWrite(trigpin, LOW);
73   digitalWrite(trigpin, HIGH);
74   delayMicroseconds(10);
75   digitalWrite(trigpin, LOW);
76   duration=pulseIn(echopin, HIGH);
77   dist=duration*speed/2;
78   if(dist>100){
79     String payload = "{\"Alert Distance\":\"";
80     payload += dist;
81     payload += "\"}";
82     Serial.print("\n");
83     Serial.print("Sending payload: ");
84     Serial.println(payload);
85     if (client.publish(publishTopic, (char*) payload.c_str())) {
86       Serial.println("Publish OK");
87     }
88   }
89   if(dist<100){
90     String payload = "{\"Distance\":\"";
91     payload += dist;
92     payload += "\"}";
```

Simulation window output:

```
Distance: 148cm
Publish OK
Sending payload: {"Alert Distance":147.93}
Publish OK
Sending payload: {"Alert Distance":147.95}
Publish OK
```

The screenshot shows the IBM Watson IoT Platform dashboard. The page is titled 'Device Drilldown - 456789'. It displays the following information:

- Connection Information:** Recent Events, State, Device Information, Metadata, Diagnostics, Connection Logs, Device Actions.
- Recent Events:** A table showing the live stream of data coming and going from the device.
- State:** A table showing a list of data points reported by the device.

Event	Value	Format	Last Received
Jijitha AL	{"Distance":249.95}	json	a few seconds ago
Jijitha AL	{"Distance":249.97}	json	a few seconds ago
Jijitha AL	{"Distance":249.97}	json	a few seconds ago
Jijitha AL	{"Distance":249.97}	json	a few seconds ago
Jijitha AL	{"Distance":249.97}	json	a few seconds ago

State: 4 Simulations running

## 2) When the distance is less than 100 cms

The screenshot shows the Wokwi simulation environment. On the left, the Arduino IDE window displays the following code:

```
59 Serial.println();
60 }
61
62 void initManagedDevice() {
63   if (client.subscribe(topic)) {
64     // Serial.println(client.subscribe(topic));
65     Serial.println("IBM subscribe to cmd OK");
66   } else {
67     Serial.println("subscribe to cmd FAILED");
68   }
69 }
70 void publishData()
71 {
72   digitalWrite(trigpin, LOW);
73   digitalWrite(trigpin, HIGH);
74   delayMicroseconds(10);
75   digitalWrite(trigpin, LOW);
76   duration=pulseIn(echopin, HIGH);
77   dist=duration*speed/2;
78   if(dist>100){
79     String payload = "{\"Alert Distance\":\"";
80     payload += dist;
81     payload += "\"}";
82     Serial.print("\n");
83     Serial.print("Sending payload: ");
84     Serial.println(payload);
85     if (client.publish(publishTopic, (char*) payload.c_str())) {
86       Serial.println("Publish OK");
87     }
88   }
89 }
90 if(dist<100){
91   String payload = "{\"Distance\":\"";
92   payload += dist;
```

On the right, the simulation window shows an ESP32 connected to an HC-SR04 Ultrasonic Distance Sensor. The sensor's distance is set to 148cm. The console output shows the following messages:

```
Publish OK
Sending payload: {"Alert Distance":147.93}
Publish OK
Sending payload: {"Alert Distance":147.95}
Publish OK
```

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area displays a list of devices. The device '456789' is highlighted, showing its status as 'Connected' and its type as 'UltraSonicSensor'. The 'Recent Events' tab is selected, showing a list of events with the following columns: Event, Value, Format, and Last Received.

Event	Value	Format	Last Received
Jijitha AL	{"Alert Distance":147.95}	json	a few seconds ago
Jijitha AL	{"Alert Distance":147.95}	json	a few seconds ago
Jijitha AL	{"Alert Distance":147.95}	json	a few seconds ago
Jijitha AL	{"Alert Distance":147.95}	json	a few seconds ago
Jijitha AL	{"Alert Distance":147.95}	json	a few seconds ago

At the bottom of the dashboard, a status bar indicates '4 Simulations running'.

