

INDUSTRY – SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

ASSIGNMENT – 4

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.

SOURCE CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
#define ORG "486ral"
#define DEVICE_TYPE "IOT"
#define DEVICE_ID "id07"
#define TOKEN "123456789"
#define trigpin 5
#define echopin 18

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

```
WiFiClient wifiClient;
```

```
PubSubClient client(server, 1883, wifiClient);
```

```
long duration;
```

```
float dist;
```

```
void setup()
```

```
{
```

```
  Serial.begin(9900);
```

```
  pinMode(trigpin, OUTPUT);
```

```
  pinMode(echopin, INPUT);
```

```
  wifiConnect();
```

```
  mqttConnect();
```

```
}
```

```
void loop() {
```

```
  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);
    }

    Serial.println();
  }
}
```

```

void publishData()
{
    digitalWrite(trigpin,LOW);
    digitalWrite(trigpin,HIGH);
    delayMicroseconds(10);
    digitalWrite(trigpin,LOW);
    duration=pulseIn(echopin,HIGH);
    dist=duration*0.034 /2;
    if(dist<100)
    {

        String payload = "{ \"Distance\": ";
        payload += dist;
        payload += ",";
        payload += "\"Status\": ";
        payload += "\"Alert\" } ";

        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 += ",";
    payload += "\\"Status\\"";
    payload += "\\"Normal\\"}";

    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");
    }
}
}
}

```

diagram.json:

```

{
    "version": 1,
    "author": "Uri Shaked",
    "editor": "wokwi",
    "parts": [
        { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0, "attrs": { } },
        { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -109.38, "left": 180.61, "attrs": { } }
    ]
}

```

```

],
"connections": [
  [ "esp:TX0", "$serialMonitor:RX", "", [] ],
  [ "esp:RX0", "$serialMonitor:TX", "", [] ],
  [ "ultrasonic1:ECHO", "esp:D18", "green", [ "v0" ] ],
  [ "ultrasonic1:TRIG", "esp:D5", "orange", [ "v0" ] ],
  [
    "ultrasonic1:VCC",
    "esp:VIN",
    "red",
    [ "v22.14", "h-48.86", "v-27.94", "h-253.24", "v173.77" ]
  ],
  [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v250.04", "h-311.59", "v3.06" ] ]
]
}

```

OUTPUT:

The screenshot displays the Wokwi IDE interface. On the left, the code for `esp32-blink.ino` is shown. It includes headers for `MFi.h` and `PubSubClient.h`, defines constants for the organization, device type, ID, token, and pins, and sets up a Wi-Fi client and a PubSubClient. The `setup` function initializes the serial port and connects to the Wi-Fi and MQTT network.

On the right, the simulation window shows a physical representation of the ESP32 and the HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's VIN pin (red), its GND pin is connected to the ESP32's GND pin (black), and its TRIG pin is connected to the ESP32's D5 pin (orange). The ECHO pin is connected to the ESP32's D18 pin (green).

The serial output window shows the following messages:

```

Sending payload: {"Distance":399.92,"Status":"Normal"}
Publish OK

Sending payload: {"Distance":399.92,"Status":"Normal"}
Publish OK

Sending payload: {"Distance":399.92,"Status":"Normal"}

```

WOKWI LINK: <https://wokwi.com/projects/322410731508073042>

IBM CLOUD OUTPUT:

Connection Information

Recent Events

State

Device Information

Metadata

Diagnostics

Connection Logs

Device Actions

Recent Events

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

Event	Value	Format	Last Received
data	{"Distance":29.99,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":29.99,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":29.99,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":29.99,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":29.99,"Status":"Alert"}	json	a few seconds ago