

PROBLEM STATEMENT:

**IoT BASED GAS LEAKAGE MONITORING AND
ALERTING SYSTEM**

DOMAIN:

INTERNET OF THINGS

ASSIGNMENT 4:

DISTANCE DETECTION USING ULTRASONIC SENSOR

BY

NISHANTH P-623519106022

KARTHIC RAJA L V-623519106013

SANTHOSH C-623519106032

LOKESH A-623519106015

QUESTION-1:

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

WOKWI LINK:

<https://wokwi.com/projects/347922632848441938>

CODE:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "5zjrzt" //IBM ORGANITION ID
#define DEVICE_TYPE "esp32device" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "nishanth" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "7695874778" //Token
String data3;
float dist;

//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth"; // authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id

//-----
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined
client id by passing parameter like server id, port and wificredential
```

```

int LED = 4;
int trig = 5;
int echo = 18;
void setup()
{
  Serial.begin(115200);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
}
void loop()// Recursive Function
{

  digitalWrite(trig,LOW);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  float dur = pulseIn(echo,HIGH);
  float dist = (dur * 0.0343)/2;
  Serial.print ("Distancein cm");
  Serial.println(dist);

  PublishData(dist);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}

/*.....retrieving to
Cloud.....*/

void PublishData(float dist) {
  mqttconnect();//function call for connecting to ibm
  /*
    creating the String in in form JSon to update the data to ibm cloud
  */
  String object;
  if (dist <100)
  {
    digitalWrite(LED,HIGH);
    Serial.println("object is near");
    object = "Near";
  }
}

```

```

else
{
    digitalWrite(LED, LOW);
    Serial.println("no object found");
    object = "No";
}

String payload = "{\"distance\": ";
payload += dist;
payload += ", \"object\": \"";
payload += object;
payload += "\"}";

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it successfully upload data on the cloud then
    it will print publish ok in Serial monitor or else it will print publish failed
} 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() //function definition for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the
    connection
    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=="Near")
    // {
    // Serial.println(data3);
    // digitalWrite(LED,HIGH);

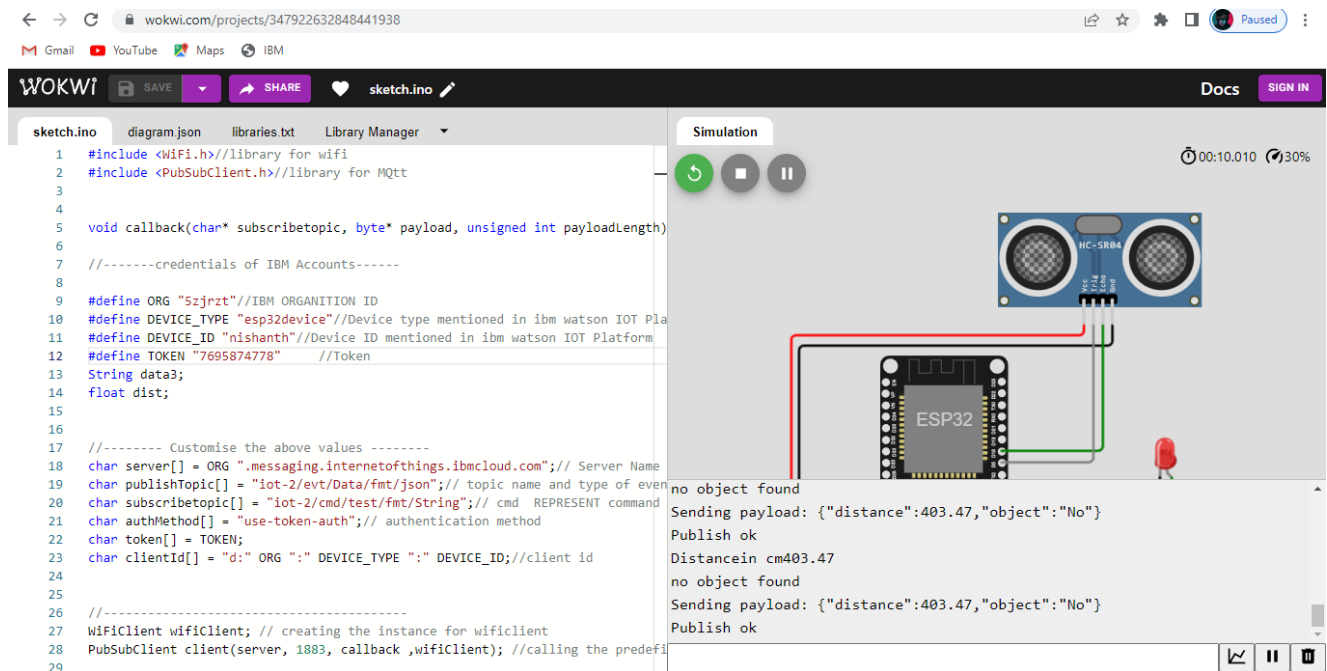
    // }

    // else
    // {
    // Serial.println(data3);
    // digitalWrite(LED,LOW);

    // }
    data3="";
}

```

OUTPUT:



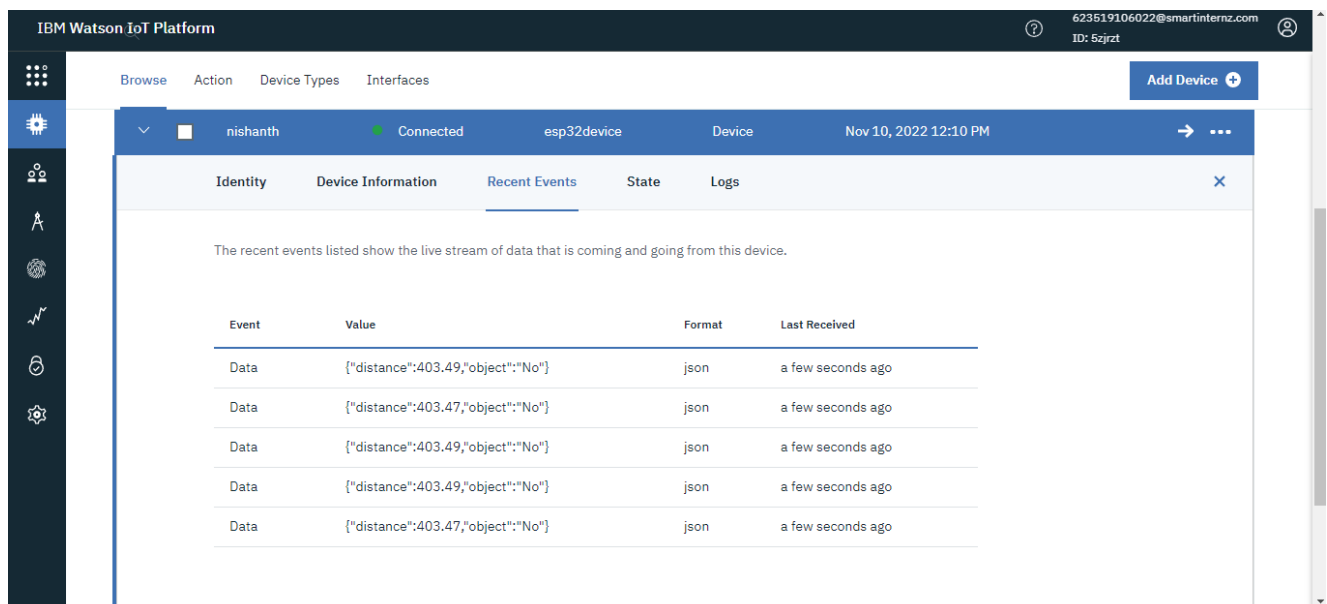
The screenshot shows the Wokwi IoT simulator interface. On the left, the 'sketch.ino' file is open, displaying C++ code for an ESP32 device connected to an IBM Watson IoT Platform. The code includes headers for WiFi and MQTT, defines credentials and device information, and sets up a callback function for MQTT messages. On the right, the 'Simulation' window shows a visual representation of the ESP32 and an HC-SR04 ultrasonic sensor. Below the simulation, the console output shows the device sending payloads to the IBM cloud when the object is far (distance 403.47 cm) and not found.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
5
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "5zjrzt" //IBM ORGANITION ID
9 #define DEVICE_TYPE "esp32device" //Device type mentioned in ibm watson IOT Pla
10 #define DEVICE_ID "nishanth" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "7695874778" //Token
12 String data3;
13 float dist;
14
15 //----- Customise the above values -----
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of even
19 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24 //-----
25 WiFiClient wifiClient; // creating the instance for wifiClient
26 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefi
27
28
```

Simulation output:

```
no object found
Sending payload: {"distance":403.47,"object":"No"}
Publish ok
Distancein cm403.47
no object found
Sending payload: {"distance":403.47,"object":"No"}
Publish ok
```

Data sent to the IBM cloud device when the object is far



The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for device management. The main content area displays a table of recent events for a device named 'nishanth' (ID: 5zjrzt). The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events show a stream of data points where the distance is 403.47 cm and the object is 'No'.

Event	Value	Format	Last Received
Data	{"distance":403.49,"object":"No"}	json	a few seconds ago
Data	{"distance":403.47,"object":"No"}	json	a few seconds ago
Data	{"distance":403.49,"object":"No"}	json	a few seconds ago
Data	{"distance":403.49,"object":"No"}	json	a few seconds ago
Data	{"distance":403.47,"object":"No"}	json	a few seconds ago