

## IBM ASSIGNMENT 4

NAME	NIVETHA.S
TOPIC	SMART SOLUTIONS FOR RAILWAYS
TEAMID	PNT2022TMID38248

Write a code and connections in wokwi for ultrasonic sensors whenever distance is less than 100cms send "alert" o ibm cloud and display in device recent events upload wokwi share link and images of ibm cloud.

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int
payloadLength);
#define ORG "swbkb4"
#define DEVICE_TYPE "iot"
#define DEVICE_ID "2022"
#define TOKEN "12348765"
String data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/distance/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);

#define ECHO_PIN 2
#define TRIG_PIN 4
#define led 5

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
```

```

    wificonnect();
    mqttconnect();
}
float readDistanceCM() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    int duration=random(1,200);
    //Serial.println(duration);
    //duration = pulseIn(ECHO_PIN, HIGH);
    return duration ;
    //Serial.println(duration);
}

void loop() {
    float distance = readDistanceCM();
    //Serial.println(distance);

    bool isNearby = distance < 100;
    digitalWrite(led, isNearby);

    Serial.print("Measured distance: ");
    Serial.println(distance);
    if(distance<100){
        PublishData2(distance);
    }else{
        PublishData1(distance);
    }
    //PublishData(distance);
    delay(1000);
    if(!client.loop()){
        mqttconnect();
    }

    //delay(2000);
}
void PublishData1(float dist){
    mqttconnect();
    String payload= "{\"distance\":";
    payload += dist;

```

```

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 PublishData2(float dist){
    mqttconnect();
    String payload= "{\\"ALERT\\":\"";
    payload += dist;
    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 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++){
        data3 += (char)payload[i];
    }
    Serial.println("data:" + data3);
    if(data3=="lighton"){
        Serial.println(data3);
        digitalWrite(led,HIGH);
    }else{
        Serial.println(data3);
        digitalWrite(led,LOW);
    }
    data3="";
}

```

**LINK :** <https://wokwi.com/projects/347477925588632147>

## NORMAL CASE:

The screenshot shows the Wokwi IDE interface. The left pane displays the sketch code for an ESP32. The right pane shows a simulation of the hardware, including an ESP32 microcontroller, a red LED, and an HC-SR04 ultrasonic sensor. The console output shows the simulation results.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "swbkb4"
5 #define DEVICE_TYPE "iot"
6 #define DEVICE_ID "2022"
7 #define TOKEN "12348765"
8 String data3;
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
16
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19
20 #define ECHO_PIN 2
21 #define TRIG_PIN 4
22 #define led 5
23
24 void setup() {
25   // put your setup code here, to run once:
26   Serial.begin(115200);
27   pinMode(led, OUTPUT);
28   pinMode(TRIG_PIN, OUTPUT);
29   pinMode(ECHO_PIN, INPUT);
30 }
```

Simulation output:

```
publish ok
Measured distance: 1.00
Sending payload:{"ALERT":1.00}
publish ok
Measured distance: 175.00
Sending payload:{"distance":175.00}
publish ok
```

## ALTER CASE:

The screenshot shows the Wokwi IDE interface. The left pane displays the sketch code for an ESP32. The right pane shows a simulation of the hardware, including an ESP32 microcontroller, a red LED, and an HC-SR04 ultrasonic sensor. The console output shows the simulation results.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "swbkb4"
5 #define DEVICE_TYPE "iot"
6 #define DEVICE_ID "2022"
7 #define TOKEN "12348765"
8 String data3;
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
16
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19
20 #define ECHO_PIN 2
21 #define TRIG_PIN 4
22 #define led 5
23
24 void setup() {
25   // put your setup code here, to run once:
26   Serial.begin(115200);
27   pinMode(led, OUTPUT);
28   pinMode(TRIG_PIN, OUTPUT);
29   pinMode(ECHO_PIN, INPUT);
30 }
```

Simulation output:

```
Measured distance: 140.00
Sending payload:{"distance":140.00}
publish ok
Measured distance: 26.00
Sending payload:{"ALERT":26.00}
publish ok
Reconnecting to swbkb4.messaging.internetofthings.ibmcloud.com
```

## IBM CLOUD:

The screenshot displays the IBM Watson IoT Platform interface in a web browser. The browser's address bar shows the URL `swbkb4.internetofthings.ibmcloud.com/dashboard/devices/browse`. The page header includes the IBM Watson IoT Platform logo and a user profile for `nivethaseharaj111@gmail.com` with ID `swbkb4`. The main navigation bar features tabs for `Browse`, `Action`, `Device Types`, and `Interfaces`, along with an `Add Device` button. The `Browse` tab is active, showing a list of device events under the `Recent Events` sub-tab. The events are displayed in a table with columns for `Event`, `Value`, `Format`, and `Last Received`. The table contains five rows of data, all for the `distance` event, with values ranging from 100 to 165. A status box at the bottom right indicates `0 Simulations running`. The Windows taskbar at the bottom shows the system clock as 13:51 on 05-11-2022.

IBM Watson IoT Platform

Service Details - IBM Cloud

swbkb4.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

nivethaseharaj111@gmail.com  
ID: swbkb4

Browse Action Device Types Interfaces

Add Device

Identity Device Information Recent Events State Logs

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

Event	Value	Format	Last Received
distance	{"distance":124}	json	a few seconds ago
distance	{"distance":100}	json	a few seconds ago
distance	{"distance":124}	json	a few seconds ago
distance	{"distance":104}	json	a few seconds ago
distance	{"distance":165}	json	a few seconds ago

0 Simulations running

Type here to search

30°C Cloudy

13:51  
05-11-2022