

## ASSIGNMENT - 4

Date	27 October 2022
Team ID	PNT2022TMID51648
Name	SMART SOLUTIONS FOR RAILWAYS - IOT
Maximum Marks	2 Marks

### QUESTION:

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.

### CODE :

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

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

#define ORG "8hcf5x"                            // IBM organisation id
#define DEVICE_TYPE "ULTASON"                  // Device type mentioned in ibm watson iot platform
#define DEVICE_ID "assignment"                 // Device ID mentioned in ibm watson iot platform
#define TOKEN "DSVsRN1CU9-eEPkcc3"           // Token
#define speed 0.034
#define led 14
String data3;
int LED = 4;

//----- customise above values -----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // server name
char publishTopic[] = "iot-2/evt/sreedhar/fmt/json";           // topic name and type of event perform and format in which data
to be send
char topic[] = "iot-2/cmd/led/fmt/String";                     // cmd Represent type and command is test format of strings
char authMethod[] = "use-token-auth";                         // authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client id

//-----

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

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(); // function call to connect to ibm
  }
}

/* -----retrieving to cloud-----*/

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("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)
  {
    digitalWrite(LED, HIGH);
    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())) // if data is uploaded to cloud successfully, prints publish ok else prints
publish failed
{
    Serial.println("Publish OK");
}

}

if(dist>100)
{
    digitalWrite(LED,HIGH);
    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
    {
        digitalWrite(LED,LOW);
        Serial.println("Publish FAILED");
    }
}

}

}

```

## OUTPUT :

Code simulation on wokwi

The screenshot displays the Wokwi web IDE interface. On the left, the code editor shows the following code:

```

1 #include <WiFi.h> // library for wifi
2 #include <PubSubClient.h> // library for MQTT
3
4
5 //----- credentials of IBM Accounts -----
6
7 #define ORG "9gbe4w" // IBM organisation id
8 #define DEVICE_TYPE "ULTASON" // Device type mentioned in ibm
9 #define DEVICE_ID "assignment" // Device ID mentioned in ibm
10 #define TOKEN "DSVsRN1CU9-eEPkc3" // Token
11 #define speed 0.034
12 #define led 14
13 String data3;
14 int LED = 4;
15
16 //----- customise above values -----
17
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
19 char publishTopic[] = "iot-2/evt/distance/fmt/json";
20 char topic[] = "iot-2/cmd/led/fmt/String";
21 char authMethod[] = "use-token-auth";
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
24
25
26 //-----
27

```

On the right, the 'Simulation' window shows a visual representation of the ESP32 and the Ultrasonic Distance Sensor. Below the simulation, the console output displays the following messages:

```

Sending payload: {"Distance":193.97}
Publish OK

Sending payload: {"Distance":193.97}
Publish OK

Reconnecting MQTT client to

```

## Data sent to IBM Cloud with distance

The screenshot displays the IBM Watson IoT Platform dashboard. The browser address bar shows the URL `9gbe4w.internetofthings.ibmcloud.com/dashboard/devices/browse`. The dashboard header includes the IBM Watson IoT Platform logo and a user profile section with the email `962319104024@students.amrita.edu.in` and ID `9gbe4w`. The main navigation bar contains tabs for `Browse`, `Action`, `Device Types`, and `Interfaces`. A sidebar on the left contains icons for various IoT functions. The main content area shows a list of devices, with the device `9gbe4w` selected. The `Recent Events` tab is active, displaying a table of recent events. The table has four columns: `Event`, `Value`, `Format`, and `Last Received`. The events listed are all `distance` events with a value of `{"distance":141.32}` and a format of `json`, all received a few seconds ago. A status message at the bottom right indicates `1 Simulation running`.

Event	Value	Format	Last Received
distance	<code>{"distance":141.32}</code>	json	a few seconds ago
distance	<code>{"distance":141.32}</code>	json	a few seconds ago
distance	<code>{"distance":141.32}</code>	json	a few seconds ago
distance	<code>{"distance":141.32}</code>	json	a few seconds ago
distance	<code>{"distance":141.32}</code>	json	a few seconds ago