

#### Assignment -4

Assignment Date	2 Nov 2022
Student Name	Kurapati.Anuhya
Student Roll Number	31211904012
Team ID	PNT2022TMID37599

#### Question-1:

Write code and connections in wokwifor 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. Upload document with wokwi share link and images of IBM cloud

#### Program:

```
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient wifiClient;
String data3;

#define ORG "v6wg8x"
#define DEVICE_TYPE "nodeMcu"
#define DEVICE_ID "NodeMCU"
#define TOKEN "123456789"
#define speed 0.034
#define led 14

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

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json"; char topic[] =
"iot-2/cmd/test/fmt/String"; char authMethod[] = "use-token-
auth"; char token[] = TOKEN; char clientId[] = "d:" ORG ":"
DEVICE_TYPE ":" DEVICE_ID; PubSubClient client(server, 1883,
callback , wifiClient); void publishData();

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 = "{ \"Normal 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>101 && dist<111){
    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("Warning crosses 110cm -- it automaticaly of the loop");
    }
    digitalWrite(led,HIGH);
}
else {
    Serial.println("Publish 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++){
        dist += (char)payload[i];
    }
    Serial.println("data:" + data3);
    if(data3=="lighton"){
        Serial.println(data3);
        digitalWrite(led,HIGH);
    }
    data3="";
}

```

## Output:

### 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	{"Normal Distance":85.99}	json	a few seconds ago
Data	{"Normal Distance":85.99}	json	a few seconds ago
Data	{"Normal Distance":85.99}	json	a few seconds ago
Data	{"Normal Distance":85.95}	json	a few seconds ago
Data	{"Alert distance":110.98}	json	a few seconds ago

```
Sending payload: {"Normal Distance":99.98}
```

```
Publish OK
```

```
Sending payload: {"Normal Distance":99.98}
```

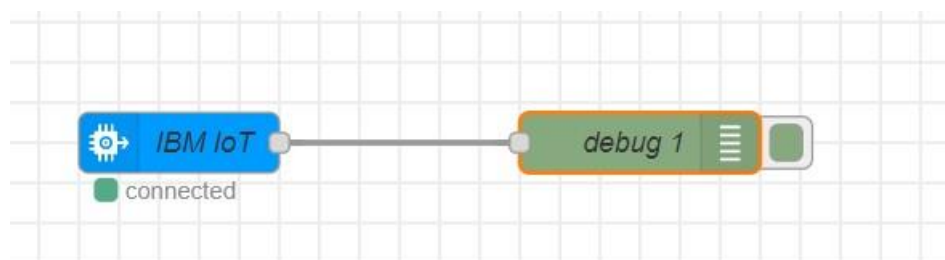
```
Publish OK
```

```
Sending payload: {"Alert distance":110.98}
```

```
Warning crosses 110cm -- it automaticaly of the loop
```

```
Sending payload: {"Normal Distance":85.95}
```

```
Publish OK
```



## Connection Information

Basic connection information about this device

Device ID	NodeMCU
Device Type	NodeMCU
Device Added	Nov 2,2022.6:47pm
Added By	<a href="mailto:312119104012@smartinternz.com">312119104012@smartinternz.com</a>
Connection Status	<b>Disconnected</b> Last connected 2,2022.6:47pm Client Address:145.40.94.93 Insecure Duration: a few seconds Data Tranferred:1.5 KB

The screenshot displays the WOKWI IoT simulator interface. On the left, a code editor shows a sketch for an ESP8266 NodeMCU. The code includes headers for WiFi and PubSubClient, defines device parameters like ORG, DEVICE\_TYPE, DEVICE\_ID, and TOKEN, and sets up an MQTT client to publish distance data. The right side features a 'Simulation' window with a visual representation of the NodeMCU and an Ultrasonic Distance Sensor. A control slider for the sensor is set to 86cm. Below the simulation, a console window shows the output of the program, including the sending of payloads for 'Alert distance' and 'Normal Distance'.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 WiFiClient wificlient;
4 String data3;
5
6 #define ORG "v6wg8x"
7 #define DEVICE_TYPE "nodeMcu"
8 #define DEVICE_ID "NodeMCU"
9 #define TOKEN "123456789"
10 #define speed 0.034
11 #define led 14
12
13 void callback(char* topic, byte* payload, unsigned int payloadLength);
14
15 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
16 char publishTopic[] = "iot-2/evt/data/fmt/json";
17 char topic[] = "iot-2/cmd/test/fmt/String";
18 char authMethod[] = "use-token-auth";
19 char token[] = TOKEN;
20 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
21 PubSubClient client(server, 1883, callback, wificlient);
22 void publishData();
23
24
25 const int trigpin=5;
26 const int echopin=18;
27 String command;
28 String data="";
29
30 long duration;
31 float dist;
32
33
34
35 void setup()
```

Simulation

Editing Ultrasonic Distance Sensor  
Distance: 86cm

Publish OK

Sending payload: {"Alert distance":110.98}  
Warning crosses 110cm -- it automatically of the loop

Sending payload: {"Normal Distance":85.95}  
Publish OK