

Assignment-4

Smart Solution For Railways

Team ID:PNT2022TMID52217

write code and connections in wowki for ultrasonic sensor. whenever distance is less than 100cm send “alert” to ibm cloud and display in device recent events. upload document with wowki share link and images of ibm cloud.

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload,unsigned intpayloadLength);#define ORG
"fdd82r"
#define DEVICE_TYPE "Pi"
#define DEVICE_ID "123"
#define TOKEN "12345678"
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;

WiFiClientwifiClient;
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("p
        ublish 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("p
        ublish 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");
    }
}

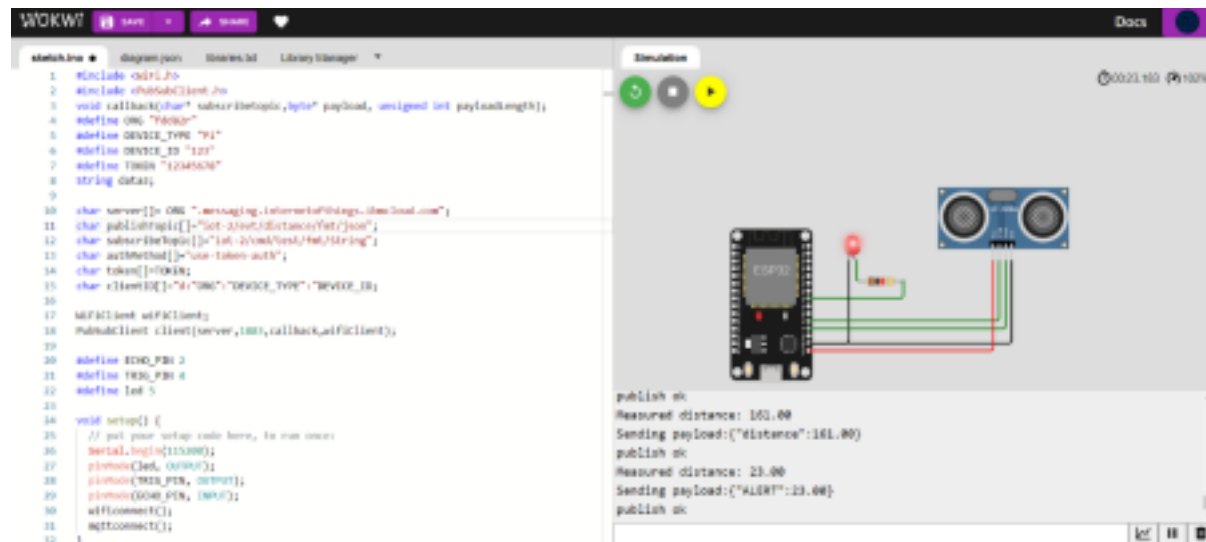
void callback(char* subscribeTopic, byte* payload, unsigned int payloadLength){
    Serial.print("callback invoked for topic:");
    Serial.println(subscribeTopic);
    for(int i=0; i<payloadLength; i++){
        data3 += (char)payload[i];
    }
    Serial.println("data:" + data3);
    if(data3=="lighton"){
        Serial.println(data3);
        digitalWrite(led,HIGH);
    }else{
        Serial.println(data3);
        digitalWrite(led,LOW);
    }
    data3="";
}

```

WOKWI project link

<https://wokwi.com/projects/346327942785139284>

Alert Case:



The screenshot displays the Wokwi IDE interface. On the left, the C++ code for the ESP8266 module is shown. The code includes the necessary libraries, defines the MQTT server and topic, and implements the callback function. The right side shows the simulation of the hardware, including the ESP8266 module, an LED, and a buzzer. The console output shows the MQTT client connecting, subscribing to the topic, and receiving a payload that triggers the alert (LED on and buzzer on).

```

1 #include <ESP8266.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribeTopic, byte* payload, unsigned int payloadLength){
4     define ON "red"
5     define OFF "blue"
6     define DEVICE_ID "1"
7     define DEVICE_ID "1"
8     define Topic "12345678"
9     String data3;
10
11     char server[] = "mqtt://mqtt://192.168.1.100:1883";
12     char publishTopic[] = "12345678/12345678";
13     char subscribeTopic[] = "12345678/12345678";
14     char authMethod[] = "mqtt-tls";
15     char token[] = "token";
16     char clientID[] = "12345678";
17     PubSubClient client(server, 1883, callback, authMethod, token, clientID);
18
19     define LED_PIN 2
20     define Buzzer_PIN 4
21     define led 5
22
23     void setup() {
24         // put your setup code here, to run once:
25         Serial.begin(115200);
26         pinMode(LED_PIN, OUTPUT);
27         pinMode(Buzzer_PIN, OUTPUT);
28         pinMode(LED_PIN, OUTPUT);
29         digitalWrite(LED_PIN, LOW);
30         client.connect();
31     }

```

publish ok
Measured distance: 100.00
Sending payload: {"distance":100.00}
publish ok
Measured distance: 20.00
Sending payload: {"ALERT":20.00}
publish ok

Normal Case:

