

| | |
|----------------------|---|
| NAME | HARIHARAN A |
| REGISTER NO | 611819106012 |
| PROJECT TITLE | SMART SOLUTION FOR RAILWAYS |
| TOPIC | CONNECTIONS IN WOKWI FOR ULTRASONIC SENSOR WHENEVER THE DISTANCE IS LESS THAN 100 CMS SEND AN ALERT TO IBM CLOUD |
| ASSIGNMENT | 04 |
| MENTOR | PRAKASAM L ASP/ECE |
| COLLEGE NAME | PSV COLLEGE OF ENGINEERING AND TECHNOLOGY |

Code :

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int
payloadLength);
#define ORG "112t39"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "54321"
#define TOKEN "123456789"
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="";
}
```

WOKWI LINK:

<https://wokwi.com/projects/346843326547231315>

WOKWI SIMULATION

The Wokwi simulation interface displays a sketch of an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The code on the left configures the ESP32 to connect to an MQTT broker and publish distance measurements. The simulation window on the right shows the circuit and a terminal output.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "nyiusu"
5 #define DEVICE_TYPE "ESP32"
6 #define DEVICE_ID "143"
7 #define TOKEN "9360857345"
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/distance/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
16 long now;
17
18 WiFiClient wifiClient;
19 PubSubClient client(server, 1883, callback, wifiClient);
20
21 #define ECHO_PIN 2
22 #define TRIG_PIN 4
23 #define led 5
24
25 void setup() {
26   // put your setup code here, to run once:
27   Serial.begin(115200);
28   pinMode(led, OUTPUT);
29   pinMode(TRIG_PIN, OUTPUT);
30   pinMode(ECHO_PIN, INPUT);
31   wifiConnect();
32   mqttConnect();
33 }
```

publish ok
Measured distance: 114.00
Sending payload:{"distance":114.00}
publish ok
Measured distance: 155.00
Sending payload:{"distance":155.00}
publish ok

CLOUD STORAGE

The IBM Watson IoT Platform dashboard displays the device drilldown for ESP32:143. The interface shows a list of recent events and a state section.

Device Drilldown - 143

Connection Information

Recent Events

| Property | Value | Type | Event | Last Received |
|----------|------------------|------|-------|-------------------|
| distance | ["ALERT":83] | json | | a few seconds ago |
| distance | ["ALERT":19] | json | | a few seconds ago |
| distance | ["distance":157] | json | | a few seconds ago |
| distance | ["ALERT":80] | json | | a few seconds ago |
| distance | ["distance":105] | json | | a few seconds ago |

State

This table shows a list of data points that are reported by this device.

Showing Raw Data | No Interfaces Available