

| | |
|---------------------|-----------------|
| Assignment Date | 25 October 2022 |
| Student Name | MOHANRAM R |
| Student Roll Number | 811519106085 |
| Maximum Marks | 2 Marks |

ASSIGNMENT-4

Problem Statement:

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.

Source Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int payloadLength);
#define ORG "p4s6t5"
#define DEVICE_TYPE "Ultrasonic_Sensor_ESP32"
#define DEVICE_ID "1923"
#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;
WiFiClient wifiClient;
PubSubClient client(server,1883,callback,wifiClient);
#define ECHO_PIN 12
#define TRIG_PIN 13
#define led 2
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); // Clear the trigger
delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10
microseconds
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
int duration=pulseIn(ECHO_PIN, HIGH);
//Serial.println(duration);
//duration = pulseIn(ECHO_PIN, HIGH);
return duration*0.017;
//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(dista
nce);
}else{ PublishData1(dis
tance);
}
//PublishData(distance);
delay(1000);
if(!client.loop()){ mqttt
connect();
}
//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.connect
ed()){ 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(subs
cribeTopic)){ 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.pr
  intln(data3);
  digitalWrite(led,HIGH);
  }else{ Serial.println(
  data3);
  digitalWrite(led,LOW);
  }
  data3="";
  }
}

```

Wokwi Link:

<https://wokwi.com/projects/346500585879503442>

Normal and Alert case:

WOKWI

SAVE

SHARE

esp32-dht22.ino by urish

esp32-dht22.ino

diagram.json

libraries.txt

Library Manager

```

1 #include <Wifi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload, unsigned int payloadLength);
4 #define ORG "p4s6t5"
5 #define DEVICE_TYPE "Ultrasonic_Sensor_ESP32"
6 #define DEVICE_ID "1923"
7 #define TOKEN "12345678"
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/distance/fmt/json";
11 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
15 WiFiClient wifiClient;
16 PubSubClient client(server,1883,callback,wifiClient);
17 #define ECHO_PIN 12
18 #define TRIG_PIN 13
19 #define led 2
20 void setup() {
21 // put your setup code here, to run once:
22 Serial.begin(115200);
23 pinMode(led, OUTPUT);
24 pinMode(TRIG_PIN, OUTPUT);
25 pinMode(ECHO_PIN, INPUT);
26 wifiClient.connect();
27 mqttConnect();
28 }
29 float readDistanceCM() {
30 digitalWrite(TRIG_PIN, LOW); // Clear the trigger
31 delayMicroseconds(2);
32 digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10 microseconds
33 delayMicroseconds(10);
34 digitalWrite(TRIG_PIN, LOW);
35 int duration=pulseIn(ECHO_PIN, HIGH);

```

Simulation

🔄

⏏

⏸

publish ok

Measured distance: 399.98

Sending payload:{"distance":399.98}

publish ok

Measured distance: 399.92

Sending payload:{"distance":399.92}

publish ok

