

Assignment Date	22 October 2022
Student Name	Sanjay Krushnan R C
Student Roll Number	2019504577
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(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/346334151941554770>

Normal and Alert case:

The screenshot displays the Wokwi web IDE interface. On the left, the code editor shows a C++ program for an ESP32. The code includes the necessary libraries, defines the MQTT server and topics, and implements a callback function that controls an LED based on received MQTT messages. The right side of the interface shows a simulation of the hardware, including an ESP32 board, an HC-SR04 ultrasonic sensor, and an LED. Below the simulation, a console log shows the output of the program, including MQTT connection status, distance measurements, and the payloads sent to the topics.

```

esp32-blink.ino
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribtopic,byte* payload, unsigned int payloadLength);
4 #define ORG "p4s0t5"
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 mqtt.connect();
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: 326.98
Sending payload:{"distance":326.98}
publish ok
Measured distance: 326.98
Sending payload:{"distance":326.98}
publish ok

WOKWI

SAVE

SHARE

Docs

esp32-blink.ino

diagram.json

libraries.txt

Library Manager

```

1 #include <wifi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, 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 wifiConnect();
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

00:41.093 101%

Editing Ultrasonic Distance Sensor

Distance: 54cm

publish ok

Measured distance: 53.99

Sending payload: {"ALERT":53.99}

publish ok

Measured distance: 53.99

Sending payload: {"ALERT":53.99}

publish ok

IBM Cloud Storage

IBM Watson IoT Platform

2019504577@smartinternz.com ID: p4s6t5

Browse Action Device Types Interfaces

Search by Device ID

Device Simulator

Add Device

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
1923	Disconnected	Ultrasonic_Sensor_ESP32	Device	Oct 23, 2022 10:24 PM	

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
distance	{"distance":326.98}	json	a few seconds ago
distance	{"distance":326.98}	json	a few seconds ago
distance	{"distance":326.98}	json	a few seconds ago
distance	{"ALERT":53.99}	json	a few seconds ago
distance	{"ALERT":53.96}	json	a few seconds ago