

Assignment Date	22 October2022
Student Name	Vignesh S
Student Roll Number	2019504606
Maximum Marks	2 Marks

ASSIGNMENT-4

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* subscribtopic,byte* payload, unsigned int payloadLength);
#define ORG "g3hxrh"
#define DEVICE_TYPE "Nodemcu"
#define DEVICE_ID "1234"
#define TOKEN "87654321"
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 13
#define TRIG_PIN 12
#define led 14
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/346665993248965204>

OUTPUT:

NORMAL CASE:

The screenshot displays the Wokwi web IDE interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch for an ESP32. The sketch includes libraries for WiFi and PubSubClient, defines an MQTT server and topics, and sets up an ultrasonic distance sensor. The main loop publishes the measured distance to an MQTT topic. On the right, the 'Simulation' tab is active, showing a virtual representation of the ESP32 board and an ultrasonic sensor. A control panel for the sensor shows a distance of 294cm. Below the simulation, a console window displays the output of the sketch, showing the sensor's readings and the MQTT publish status.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribtopic,byte* payload,unsigned int payloadL
4 #define ORG "g3hxrh"
5 #define DEVICE_TYPE "Nodemcu"
6 #define DEVICE_ID "1234"
7 #define TOKEN "87654321"
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 13
18 #define TRIG_PIN 12
19 #define led 14
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();
```

Editing Ultrasonic Distance Sensor
Distance: 294cm

publish ok
Measured distance: 293.95
Sending payload:{"distance":293.95}
publish ok
Measured distance: 293.95
Sending payload:{"distance":293.95}
publish ok

ALERT CASE:

The screenshot shows the Wokwi IDE interface. On the left, the sketch.ino file is open, displaying the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload,unsigned int payloadLength)
4 #define ORG "g3hxrh"
5 #define DEVICE_TYPE "Nodemcu"
6 #define DEVICE_ID "1234"
7 #define TOKEN "87654321"
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 13
18 #define TRIG_PIN 12
19 #define led 14
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 }
```

On the right, the Simulation window is open, showing an Ultrasonic Distance Sensor. The distance is 86cm. Below the sensor, the following text is displayed:

```
Measured distance: 85.99
Sending payload:{"ALERT":85.99}
publish ok
Measured distance: 85.95
Sending payload:{"ALERT":85.95}
publish ok
```

IBM CLOUD STORAGE:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes the IBM logo and the text "IBM Watson IoT Platform". The main content area is titled "Browse" and shows a list of recent events for a device. The events are listed in a table with the following columns: Event, Value, Format, and Last Received.

Event	Value	Format	Last Received
distance	{"ALERT":85.95}	json	a few seconds ago
distance	{"ALERT":85.99}	json	a few seconds ago
distance	{"distance":293.98}	json	3 minutes ago
distance	{"distance":293.95}	json	3 minutes ago
distance	{"distance":293.95}	json	3 minutes ago