

Assignment Date	22 October 2022
Student Name	Agilan P
Student Roll Number	2019504003
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* subscribetopic,byte* payload, unsigned int payloadLength);
#define ORG "af6p2j"
#define DEVICE_TYPE "NodeMcu"
#define DEVICE_ID "4321"
#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 12
#define TRIG_PIN 13
#define led 26
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/new/esp32>

OUTPUT:

NORMAL CASE:

The screenshot displays the Wokwi web IDE interface. On the left, the 'sketch.ino' file contains the following code:

```

1
2 #include <Wifi.h>
3 #include <PubSubClient.h>
4 void callback(char* subscribtopic,byte* payload,unsigned int payloadLength);
5 #define ORG "af6p2j"
6 #define DEVICE_TYPE "NodeMcu"
7 #define DEVICE_ID "4321"
8 #define TOKEN "87654321"
9 String data3;
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:"ORG":"DEVICE_TYPE":"DEVICE_ID";
16 WifiClient wifiClient;
17 PubSubClient client(server,1883,callback,wifiClient);
18 #define ECHO_PIN 12
19 #define TRIG_PIN 13
20 #define led 26
21 void setup() {
22 // put your setup code here, to run once:
23 Serial.begin(115200);
24 pinMode(led, OUTPUT);
25 pinMode(TRIG_PIN, OUTPUT);
26 pinMode(ECHO_PIN, INPUT);
27 wifiConnect();
28 mqttConnect();
29 }
30 float readDistanceCM() {
31 digitalWrite(TRIG_PIN, LOW); // Clear the trigger
32 delayMicroseconds(2);
33 digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10 microseconds
34 delayMicroseconds(10);
35 digitalWrite(TRIG_PIN, LOW);

```

On the right, the 'Simulation' window shows a visual representation of the hardware. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor via four wires (VCC, GND, Trig, Echo) and to an LED. The simulation status bar at the top right indicates a runtime of 01:46.956 and 96% completion.

The output console at the bottom shows the following log messages:

```

publish ok
Measured distance: 254.95
Sending payload:{"distance":254.95}
publish ok
Measured distance: 254.93
Sending payload:{"distance":254.93}
publish ok

```

ALERT CASE:

WOKWI

SAVE SHARE

Docs

sketch.ino

```

1
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4 void callback(char* topic, byte* payload, unsigned int payloadLength);
5 #define ORG "af6p2j"
6 #define DEVICE_TYPE "NodeMCU"
7 #define DEVICE_ID "4321"
8 #define TOKEN "87654321"
9 String data;
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
16 WiFiClient wifiClient;
17 PubSubClient client(server, 1883, callback, wifiClient);
18 #define ECHO_PIN 12
19 #define TRIG_PIN 13
20 #define led 26
21 void setup() {
22   // put your setup code here, to run once:
23   Serial.begin(115200);
24   pinMode(led, OUTPUT);
25   pinMode(TRIG_PIN, OUTPUT);
26   pinMode(ECHO_PIN, INPUT);
27   wifiConnect();
28   mqttConnect();
29 }
30 float readDistance() {
31   digitalWrite(TRIG_PIN, LOW); // Clear the trigger
32   delayMicroseconds(2);
33   digitalWrite(TRIG_PIN, HIGH); // Sets the trigger pin to HIGH state for 10 microseconds
34   delayMicroseconds(10);
35   digitalWrite(TRIG_PIN, LOW);
36   float distance = 0;
37   if (digitalRead(ECHO_PIN) == HIGH) {
38     long duration = pulseIn(TRIG_PIN, HIGH);
39     distance = (duration * 0.0343) / 2;
40   }
41   return distance;
42 }
43 void loop() {
44   if (client.connected() == false) {
45     wifiConnect();
46     mqttConnect();
47   }
48   if (client.publish(publishTopic, String(readDistance()).c_str()) > 0) {
49     Serial.println("publish ok");
50     Serial.println("Measured distance: " + String(readDistance()));
51   }
52   if (client.subscribe(subscribeTopic) > 0) {
53     Serial.println("subscribe ok");
54   }
55   if (client.available() > 0) {
56     String message = client.receive();
57     Serial.println("Received: " + message);
58   }
59 }

```

Simulation

00:05:18 99%

publish ok
Measured distance: 88.98
Sending payload: {"ALERT":88.98}
publish ok
Measured distance: 89.00
Sending payload: {"ALERT":89.00}
publish ok

IBM CLOUD STORAGE:

af6p2j.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

2019504003@student.unmau.edu ID: af6p2j

Browse Action Device Types Interfaces

Add Device +

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
4321	Connected	NodeMCU	Device	Oct 27, 2022 1:25 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	{"ALERT":89}	json	a few seconds ago
distance	{"ALERT":88.98}	json	a few seconds ago
distance	{"ALERT":88.96}	json	a few seconds ago
distance	{"distance":254.93}	json	3 minutes ago
distance	{"distance":254.95}	json	3 minutes ago