

#### Assignment -4

Assignment Date	29 October 2022
Student Name	Dhasarathi K
Student Roll Number	710019106010
Maximum Marks	2 Marks

#### Question :

Write Code and Connections in Wokwi for Ultrasonic Sensor. Whenever Distance is less than 100 cm 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 "7f5hee"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "dhasarathi"//Device ID mentioned in ibm watson IOT
Platform #define TOKEN "4hTj6+UJsqKC(hsDHC" //Token
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="";
}
```

## Reference :

<https://wokwi.com/projects/347599931737899604>

## Output:

### 1) Distance less than 100cm – LED Bulb Glows and ‘Alert’ Message is Displayed along with Distance

The screenshot shows the Wokwi Arduino IDE interface. On the left, the sketch code is displayed, which includes the necessary libraries, pin definitions, and logic for connecting to the IBM Watson IoT Platform and controlling an LED based on distance measurements. The right side shows a simulation of the hardware setup, including an ESP32 microcontroller, an HC-SR04 ultrasonic sensor, and an LED. The console output shows the device successfully connecting to the IoT platform and sending alert messages when the distance is less than 100cm.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "7fshee" //IBM ORGANIZATION ID
5 #define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IOT Platform
6 #define DEVICE_ID "dhasarathi" //Device ID mentioned in ibm watson IOT Platform
7 #define TOKEN "4htj64UjsqKC(hsDhC" //Token
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[] = "id:" + 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 console output:

```
publish ok
Measured distance: 72.96
Sending payload:{"ALERT":72.96}
publish ok
Measured distance: 72.96
Sending payload:{"ALERT":72.96}
publish ok
```

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes options like 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is available for finding devices by ID. The main table lists devices, including 'dhasarathi' which is connected and of type 'ESP32'. Below the table, the 'Recent Events' tab is selected, showing a list of events with columns for Event, Value, Format, and Last Received. The events show distance measurements and alert messages being sent to the platform.

Event	Value	Format	Last Received
distance	{"ALERT":32}	json	a few seconds ago
distance	{"ALERT":72.96}	json	a few seconds ago
distance	{"ALERT":72.96}	json	a few seconds ago
distance	{"ALERT":6}	json	a few seconds ago
distance	{"ALERT":72.96}	json	a few seconds ago

2 Simulations running

## 2) Distance more than 100cm – LED Bulb OFF and Distance is Displayed

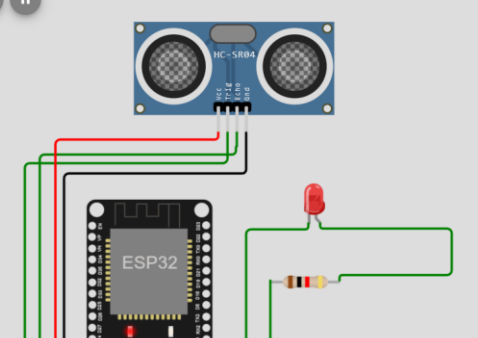
WOKWI SAVE SHARE sketch.ino Docs SIGN IN

sketch.ino diagram.json libraries.bt Library Manager

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "7f5hee" // IBM ORGANIZATION ID
5 #define DEVICE_TYPE "ESP32" // Device type mentioned in ibm watson IOT Platform
6 #define DEVICE_ID "dhasarathi" // Device ID mentioned in ibm watson IOT Platform
7 #define TOKEN "4htj6uU3sqKC(hsDHC)" // Token
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

03:03.990 93%



publish ok  
Measured distance: 134.96  
Sending payload:{"distance":134.96}  
publish ok  
Measured distance: 134.96  
Sending payload:{"distance":134.96}  
publish ok

IBM Watson IoT Platform dhasarathik2112@gmail.com ID: 7f5hee

Browse Action Device Types Interfaces Add Device

Search by Device ID Device Simulator

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
12345	Disconnected	testdevicetype	Device	Nov 5, 2022 3:15 PM	
dhasarathi	Connected	ESP32	Device	Nov 6, 2022 10:41 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":43}	json	a few seconds ago
distance	{"ALERT":36}	json	a few seconds ago
distance	{"ALERT":24}	json	a few seconds ago
distance	{"distance":134.96}	json	a few seconds ago
distance	{"distance":134.95}	json	a few seconds ago

2 Simulations running