

Assignment Date	22 October2022
Student Name	Neeraja N
Student Roll Number	2019504040
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 "jzsbm0 "
#define DEVICE_TYPE "NODEMCU"
#define DEVICE_ID "1823"
#define TOKEN "18080123"
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/347145189734744658>

OUTPUT:

NORMAL CASE:

The screenshot displays the Wokwi IDE interface for a project named 'esp32'. The left pane shows the 'sketch.ino' file with the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength) {
4   #define ORG "jssbm0"
5   #define DEVICE_TYPE "NODEMCU"
6   #define DEVICE_ID "1823"
7   #define TOKEN "18080123"
8   String data;
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    wifiConnect();
27    mqttConnect();
28  }
```

The right pane shows the 'Simulation' window. At the top, it indicates a runtime of 00:45.883 and 38% battery. Below this, there's a control bar for the 'Editing Ultrasonic Distance Sensor' with a slider set to 210cm. The simulation area shows a 3D model of the ESP32 and the HC-SR04 sensor connected by wires. The console output shows the following sequence of events:

```
publish ok
Measured distance: 209.95
Sending payload:{"distance":209.95}
publish ok
Measured distance: 209.95
Sending payload:{"distance":209.95}
publish ok
```

ALERT CASE:

The screenshot shows the Wokwi IoT simulator interface. On the left, the sketch code is displayed, which includes the necessary libraries and setup for an ESP32 to connect to an IBM Watson IoT Platform. The code defines the device type as "NODEMCU", the device ID as "1823", and the token as "18080123". It also defines the echo pin as 13 and the trig pin as 12. The setup function initializes the serial port and pins. The main loop measures the distance using the HC-SR04 ultrasonic sensor and sends an alert payload to the IoT platform.

On the right, the simulation window shows the ESP32 board connected to the HC-SR04 sensor. The sensor's distance is measured as 68.02 cm. The console output shows the following sequence of events:

```
publish ok
Measured distance: 67.98
Sending payload:{"ALERT":67.98}
publish ok
Measured distance: 68.02
Sending payload:{"ALERT":68.02}
publish ok
```

IBM CLOUD STORAGE:

The screenshot shows the IBM Watson IoT Platform dashboard. The dashboard displays the recent events for a device, which are listed in a table. The table has four columns: Event, Value, Format, and Last Received. The events are as follows:

Event	Value	Format	Last Received
distance	{"distance":209.95}	json	a few seconds ago
distance	{"distance":209.95}	json	a few seconds ago
distance	{"distance":209.95}	json	a few seconds ago
distance	{"distance":209.95}	json	a few seconds ago
distance	{"distance":209.95}	json	a few seconds ago

← → ↻ jzsbm0.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform2019504040@student.annauniv.edu ID: jzsbm0

⋮

🔧

👤

🚶

🌐

📈

🔒

⚙️

Browse

Action

Device Types

Interfaces

Recent Events

Alerts

Log

Add Device +

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":66.96}	json	a few seconds ago
distance	{"ALERT":61.97}	json	a few seconds ago
distance	{"ALERT":68.97}	json	a few seconds ago
distance	{"ALERT":61.97}	json	a few seconds ago
distance	{"ALERT":61.98}	json	a few seconds ago