

ASSIGNMENT-4

| | |
|---------------------|-----------------|
| Assignment date | 29 October 2022 |
| Student name | Jaajana Ganesh |
| Student roll number | 2019504529 |
| 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 "pvg62t"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "Jaajanaganesh"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "2022202123" //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 :

[sketch.ino - Wokwi Arduino and ESP32 Simulator](#)

Output:

The screenshot shows the Wokwi web-based simulator interface. On the left, the 'sketch.ino' file is open, displaying the following code:

```

34 digitalWrite(TRIG_PIN, LOW);
35 int duration=pulseIn(ECHO_PIN, HIGH);
36 //Serial.println(duration);
37 //duration = pulseIn(ECHO_PIN, HIGH);
38 return duration*0.017;
39 //Serial.println(duration);
40 }
41 void loop() {
42 float distance = readDistanceCM();
43 //Serial.println(distance);
44 bool isNearby = distance < 100;
45 digitalWrite(led, isNearby);
46 Serial.print("Measured distance: ");
47 Serial.println(distance);
48 if(distance<100){
49 PublishData2(distance);
50 }else{
51 PublishData1(distance);
52 }
53 //PublishData(distance);
54 delay(1000);
55 if(!client.loop()){
56 mqttconnect();
57 }
58 //delay(2000);
59 }
60 void PublishData1(float dist){
61 mqttconnect();
62 String payload= "{\"distance\":\"";
63 payload += dist;
64 payload+="}";
65 Serial.print("Sending payload:");
66 Serial.println(payload);
67 if(client.publish(publishTopic,(char*)payload_c_str())){
68 Serial.println("publish ok");

```

On the right, the 'Simulation' window shows a virtual circuit. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC is connected to the ESP32's 5V pin, and its GND is connected to the ESP32's GND pin. The sensor's TRIG pin is connected to the ESP32's D4 pin, and its ECHO pin is connected to the ESP32's D5 pin. An LED is connected to the ESP32's D8 pin (anode) and a 220Ω resistor is connected between the D8 pin and the LED's cathode. The other end of the resistor is connected to the ESP32's GND pin.

Below the simulation window, the serial output log shows the following messages:

```

publish ok
Measured distance: 81.97
Sending payload:{"ALERT":81.97}
publish ok
Reconnecting to pv62t.messaging.internetofthings.ibmcloud.com
.....
.....

```

The bottom status bar of the simulator shows the temperature as 29°C, the time as 22:41, and the date as 30-10-2022.

The screenshot displays the IBM Watson IoT Platform interface. At the top, the navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various system functions. The main content area shows a device named 'Jaajanaganes' with a status of 'Disconnected' and an ESP32 device type. The 'Recent Events' tab is active, showing a table of events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists five 'distance' events, each with a value of '{\"ALERT\":81.96}' and a timestamp of 'a few seconds ago'. The bottom status bar indicates '0 Simulations running'.

| Event | Value | Format | Last Received |
|----------|-------------------|--------|-------------------|
| distance | {\"ALERT\":81.97} | json | a few seconds ago |
| distance | {\"ALERT\":81.96} | json | a few seconds ago |
| distance | {\"ALERT\":81.96} | json | a few seconds ago |
| distance | {\"ALERT\":81.96} | json | a few seconds ago |
| distance | {\"ALERT\":81.96} | json | a few seconds ago |