

## ASSIGNMENT-4

### PYTHON PROGRAMMING

Assignment date	24-10-2022
Team Id	PNT2022TMID30880
Maximum marks	2 Marks

**Write a code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100cms send an “Alert” to IBM cloud and display in the device recent events.**

Code:

```
#include <WiFi.h>

#include<PubSubClient.h>

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "m89nt2"//IBM ORGANITION ID

#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform

#define DEVICE_ID "123"//Device ID mentioned in ibm watson IOT Platform #define

TOKEN "12345678" //Token

String data3;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Data/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);
```

```
const int trigPin = 5;

const int echoPin = 18;

#define SOUND_SPEED 0.034 long duration;

float distance;

void setup() {

  Serial.begin(115200);

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  wificonnect();

  mqttconnect();

}

void loop()

{

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);

  distance = duration * SOUND_SPEED/2;

  Serial.print("Distance (cm): ");

  Serial.println(distance);

  if(distance<100)

  {

    Serial.println("ALERT!!");

    delay(1000);

    PublishData(distance);
```

```

delay(1000);

if (!client.loop()) {
  mqttconnect();
}

}

delay(1000);

}

void PublishData(float dist) {
  mqttconnect();

  String payload = "{\"Distance\":\"";

  payload += dist;

  payload += "\",\"ALERT!!\":\"\"Distance less than 100cms\"";

  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 client 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++) {

//Serial.print((char)payload[i]);

data3 += (char)payload[i];

}

Serial.println("data: "+ data3);

data3="";

}

```

Diagram.json:

```

{

"version": 1,

"author": "sweetysharon",

"editor": "wokwi",

"parts": [

{"type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -4.67, "left": -114.67, "attrs": {}},

{"type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 15.96, "left": 89.17, "attrs": {} }

],

"connections": [

[ "esp:TX0", "$serialMonitor:RX", "", [] ],

[ "esp:RX0", "$serialMonitor:TX", "", [] ],

[

"esp:VIN",

"ultrasonic1:VCC",

"red",

[ "h-37.16", "v-178.79", "h200", "v173.33", "h100.67" ] ],

[ "esp:GND.1", "ultrasonic1:GND", "black", [ "h39.87", "v44.04", "h170" ] ],

[ "esp:D5", "ultrasonic1:TRIG", "green", [ "h54.54", "v85.07", "h130.67" ] ],

[ "esp:D18", "ultrasonic1:ECHO", "green", [ "h77.87", "v80.01", "h110" ] ]

]

}

```

```
]
}
```

## Wokwi output:

```
Connecting to ....
WiFi connected
IP address:
10.10.0.2
Reconnecting client to ytluse.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Distance (cm): 399.92
Distance (cm): 399.96
Distance (cm): 399.94
Distance (cm): 399.98
Distance (cm): 399.94
Distance (cm): 399.92
Distance (cm): 399.94
```

<https://wokwi.com/projects/348473409733132882>

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various IoT functions. The main content area shows a list of devices, with 'kaviya' selected. Below the device list, the 'Recent Events' tab is active, displaying a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events are all 'event\_1' with different random number values in JSON format, received a few seconds ago. A status bar at the bottom indicates '2 Simulations running'.

Event	Value	Format	Last Received
event_1	{"randomNumber":57}	json	a few seconds ago
event_1	{"randomNumber":86}	json	a few seconds ago
event_1	{"randomNumber":1}	json	a few seconds ago
event_1	{"randomNumber":15}	json	a few seconds ago
event_1	{"randomNumber":53}	json	a few seconds ago

## IBM cloud output:

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

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3
4 WiFiClient wificlient;
5
6 #define ORG "m89nt2"
7 #define DEVICE_TYPE "kaviya"
8 #define DEVICE_ID "123"
9 #define TOKEN "12345678"
10 #define speed 0.034
11
12
13 char server[] = ORG".messaging.internetofthings.ibmcloud.com";
14 char publishTopic[] = "iot-2/evt/event_1/fmt/json";
15 char topic[] = "iot-2/cmd/home/fmt/String";
16 char authMethod[] = "use-token-auth";
17 char token[] = TOKEN;
18 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
19 PubSubClient client(server, 1883, wificlient);
20 void publishData();
21 const int trigpin=5;
22 const int echopin=18;
23 String command;
24 String data="";
25 long duration;
26 float dist;
27 void setup()
28 {
29   Serial.begin(115200);
30   pinMode(trigpin, OUTPUT);
```

The right pane shows a simulation of the hardware, including an ESP32 module and an HC-SR04 ultrasonic sensor. Below the simulation, the console output reads:

```
Connecting to Wifi...Wifi connected, IP address: 10.10.0.2
Reconnecting MQTT client to m89nt2.messaging.internetofthings.ibmcloud.com
.....
```

