

ASSIGNMENT-4

PYTHON PROGRAMMING

Assignment date	24-10-2022
Team Id	PNT2022TMI30880
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 "4odm27"//IBM ORGANITION ID

#define DEVICE_TYPE "deepa"//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/348479377179148883>

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Service Details - IBM Cloud', 'IBM Watson IoT Platform', and 'Add Device'. The main content area shows a table of devices with columns: Device ID, Status, Device Type, Class ID, Date Added, and Descriptive Location. A device with ID 123 is selected, showing a status of 'Disconnected' and type 'deepa'. Below the table, a 'Recent Events' section displays a live stream of data. The events table has columns: Event, Value, Format, and Last Received. The events are as follows:

Event	Value	Format	Last Received
event_1	{"randomNumber":1}	json	a few seconds ago
event_1	{"randomNumber":78}	json	a few seconds ago
event_1	{"randomNumber":27}	json	a minute ago

At the bottom, a status bar indicates '1 Simulation running'.

IBM cloud output

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch that configures an ESP32 to connect to IBM Cloud IoT via MQTT. The sketch includes headers for `WiFi.h` and `PubSubClient.h`, and defines constants for the organization (`4odm27`), device type (`deepa`), device ID (`123`), token (`12345678`), and speed (`0.034`). It sets up a WiFi client and a PubSubClient, then connects to the IBM Cloud IoT endpoint (`4odm27.messaging.internetofthings.ibmcloud.com`). The `setup` function initializes the serial port and sets the pin mode for the trigger pin.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3
4 WiFiClient wifiClient;
5
6 #define ORG "4odm27"
7 #define DEVICE_TYPE "deepa"
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);
```

On the right, the 'Simulation' window shows a visual representation of the hardware. An ESP32 module is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's Trig pin is connected to the ESP32's pin 5, and its Echo pin is connected to the ESP32's pin 18. The simulation status bar indicates a connection time of 00:05.882 and a battery level of 72%.

Connecting to Wifi..Wifi connected, IP address: 10.10.0.2
Reconnecting MQTT client to 4odm27.messaging.internetofthings.ibmcloud.com
....