

Assignment -4

| | |
|---------------------|------------------|
| Assignment Date | 27 /10/2022 |
| Student Name | Karthick .S |
| Student Roll Number | 142219106043 |
| Team ID | PNT2022TMID21781 |
| Maximum Marks | 2 Marks |

Question-1:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less 100 cms send “alert” to ibm cloud and display in device recent events.

Solution:

```
#include <WiFi.h>//library for wifi
```

```
#include <PubSubClient.h>//library for MQTT
```

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

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

```
#define ORG "za7x6f"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "rj46"//Device type mentioned in ibm watson IOT Platform
```

```
#define DEVICE_ID "raj46"//Device ID mentioned in ibm watson IOT Platform
```

```
#define TOKEN "R0Q4uhcOcCD0hnom)K"
```

```
//Token String data3; float dist;
```

```
//----- Customise the above values -----char server[] = ORG
```

```
".messaging.internetofthings.ibmcloud.com";// Server Name
```

```
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and  
format in which data to be send
```

```
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command
```

```
type AND COMMAND IS TEST OF FORMAT STRING char authMethod[] = "use-  
token-auth";// authentication method
```

```

char token[] = TOKEN; char clientId[] = "d:" ORG ":"
DEVICE_TYPE ":" DEVICE_ID;//client id

//-----

WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by
passing parameter like server id,portand wificredential

int LED = 4;
int trig = 5;
int echo =
18; void
setup()
{
Serial.begin(115200);
pinMode(trig,OUTPUT
T);
pinMode(echo,INPUT
); pinMode(LED,
OUTPUT); delay(10);
wificonnect();
mqttconnect();
}
void loop()// Recursive Function
{
digitalWrite(trig,LOW);
digitalWrite(trig,HIGH);
delayMicroseconds(10);
digitalWrite(trig,LOW);
float dur =

```

```
pulseIn(echo,HIGH); float
dist = (dur * 0.0343)/2;
Serial.print ("Distancein cm");
Serial.println(dist);
```

```
PublishData(dist)
; delay(1000);
if (!client.loop())
{
mqttconnect();
}
}
/*.....retrieving to Cloud.....*/
```

```
void PublishData(float dist) {
mqttconnect();//function call for connecting to
ibm
/*    creating the String in in form JSon to update the data to
ibm cloud
*/ String
object; if
(dist <100)
{
digitalWrite(LED,HIGH);
Serial.println("object is near");
object = "Near";
}
else
{
```

```

    digitalWrite(LED,LOW);
Serial.println("no object found");
object = "No";
}

String payload =
"{\"distance\":\""; payload +=
dist; payload += ","
"\"object\":\""; payload +=
object; payload += "\"}";

Serial.print("Sending payload: ");
Serial.println(payload); if
(client.publish(publishTopic, (char*)
payload.c_str())) {

    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print
publish ok in Serial monitor or else it will print publish failed

    } 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() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");
    WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the
connection 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];

```

```
}  
data3="";  
}
```

Reference:

<https://wokwi.com/projects/347322163482591827>

The screenshot displays the Wokwi online IDE interface. On the left, the code editor shows a C++ sketch for an ESP32. The code defines a variable 'object' and uses an if-statement to check if the distance measured by the HC-SR04 sensor is less than 100. If true, it sets the LED to HIGH and prints 'object is near'. Otherwise, it sets the LED to LOW and prints 'no object found'. A JSON payload is constructed with the distance and object status, and sent via Serial.print. The right side of the interface shows a simulation of the hardware. The ESP32 is connected to the HC-SR04 sensor and a red LED. The simulation output window shows the following sequence of events:

```
object is near  
Sending payload: {"distance":59.51,"object":"Near"}  
Publish ok  
Distancein cm59.51  
object is near  
Sending payload: {"distance":59.51,"object":"Near"}  
Publish ok
```

IBM Watson IoT Platform

?

110119104034@saimec.ac.in

ID: rdegylk

← Back

Device Drilldown - weather1

Connection Information

Recent Events

State

Device Information

Metadata

Diagnostics

Connection Logs

Device Actions

Recent Events

The recent events listed show the live stream of data that is coming and going from this device.

| Event | Value | Format | Last Received |
|-------|------------------------------------|--------|-------------------|
| Data | {"distance":59.51,"object":"Near"} | json | a few seconds ago |
| Data | {"distance":59.51,"object":"Near"} | json | a few seconds ago |
| Data | {"distance":2.01,"object":"Near"} | json | a few seconds ago |
| Data | {"distance":347.01,"object":"No"} | json | a few seconds ago |
| Data | {"distance":403.49,"object":"No"} | json | a few seconds ago |

0 Simulations running