FINAL CODE

//.....Credentials For IBM .....

Organization ID

73497z

Device Type

iot_device

Device ID 1234

Authentication

Method use-

token-auth

Authentication

Token

12345678

//.....Project SourceLink on Wokwi.....

Wokwi Link - https://wokwi.com/projects/347685130732569171

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

#include <PubSubClient.h>//library for

MQtt #include "DHT.h"// Library for dht

sensor

#define DHTPIN 15 // what pin we'reconnected to

#define DHTTYPEDHT22 // define type of sensor

DHT 22 #define LED 2

DHT dht (DHTPIN,DHTTYPE);// creatingthe instance by passing pin

and typrof dht connected

void callback(char* subscribetopic, byte* payload,

unsigned int payloadLength);

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

#define ORG "88653s"//IBM ORGANITION ID

#define DEVICE_TYPE "iot_device"//Device type mentioned in

ibm watson IOT Platform

#define DEVICE_ID "1234"//Device ID mentioned in ibm

watson IOT Platform

```
#define TOKEN "12345678"

//Token String data3;

float h, t;

const float BETA = 3950; // should match the Beta Coefficient of the

thermistor

//-------- Customise the above values --------

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

ServerName

char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and

type of eventperform and format in whichdata to be send

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

REPRESENT commandtype AND COMMANDIS TEST OF

FORMAT STRING
```

```
char authMethod[] = "use-token-auth"; // authentication

methodchar 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 parameterlike server id,portand

wificredential

void setup() // configureing the ESP32
```

```
{

Serial.begin(
115200);
dht.begin();
delay(10);
Serial.println(
);
wificonnect();
mqttconnect(
);

Serial.begin(9600);
analogReadResolu
tion(10);

pinMode(18,INP
UT);
```

```
pinMode(14,OUT
PUT);
pinMode(12,OUT
PUT);

}

void loop() // Recursive Function

{
```

```
h = dht.readHumidity();

t =
dht.readTemperature
();
Serial.print("Tempera
ture:");
Serial.println(t);
Serial.print("Humidity
:"); Serial.println(h);

PublishDa
ta(t,
h);delay(1
000);

if
(!client.lo
```

```
op()) {

mqttconn

ect();


}

//.............Analog Temperature Sensor.....................
```

```
int analogValue = analogRead(18);

float celsius = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0
/ 298.15)+ 36.4;

Serial.print("Tempera
ture: ");
Serial.print(celsius);

Serial.println("
℃");
Serial.print("Alert.
.!");

if(celsius >= 35)
digitalWrite(14,
HIGH); else

digitalWrite(14,
```

```
LOW);
delay(1000);

}

/*.....retrieving to Cloud..... */

void PublishData(float temp, float humid) {
mqttconnect(); //function call for connecting
```

```
to ibm

/*

creatingtheString in in form JSon to update the data to ibm cloud

*/

String payload =

"{\"Data\":{\"temperature\":";

payload+= temp;

payload+= ","

"\"humidity\":";

payload+= humid;

payload += "}}";

Serial.print("Sending

payload: ");
```

```
Serial.println(payload);

if (client.publish(publishTopic, (char*)payload.c_str())) {

Serial.println("Publish ok"); // ifit sucessfully uploaddata on the

cloud then it will print publish ok in Serial monitoror else it will print

publish failed

Serial.println("If Temperature increased,the alarm and alert light

would indicates. ");

} 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);

}

initManagedDe

vice();
```

```
Serial.println();

}

}

void wificonnect() //function defination for wificonnect

{

Serial.println();

Serial.print("Connecti

ng to ");
```

```
WiFi.begin("Wokwi-GUEST", "", 6);//passing the wificredentials 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.localI
P());

}
```

```
void initManagedDevice() {

if (client.subscribe(subscribetopic)) {

// Serial.println((subscribetopic));

Serial.println("subscribe to

cmdOK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}
```

```
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength)
{
Serial.print("callback invokedfor topic:
");Serial.println(subscribetopic);
for (int i = 0; i <
payloadLength; i++) {
Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
Serial.println("data:
"+data3);
if(data3=="lighton")
```

```
{

Serial.println(data

3);

digitalWrite(LED,H

IGH);

}

else
```

```
{

Serial.println(data

3);

digitalWrite(LED,L

OW);

}

data3="";

}

//.....Python Script for Random Outputs of Temperature and Humidity.....

impo

rt

time

imp

ort
```

```
sys


import
ibmiotf.applicati
on import
ibmiotf.device
```

```
import random

#Provide your IBM Watson Device

Credentials organization = "bxobbs"

deviceType

="b5ibm"

deviceId=

"b5device"

authMethod =

"token" authToken

= "b55m1eibm"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" %

cmd.data['command'])
```

```
status=cmd.data['command']


if status=="l

ighton":

print("led

is on")


else :
```

```
print ("led is off")

#print(cmd)

try:

deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,

"auth-method": authMethod, "auth-token":

authToken} deviceCli =

ibmiotf.device.Client(deviceOptions)

#.....

except Exception as e:

print("Caught exception connecting device: %s" %

str(e))sys.exit()

# Connectand send a datapoint "hello" with value "world" into the

cloud as an event of type "greeting" 10 times
```

```
deviceCli.connect()

while True:

#Get Sensor Data from DHT11
```

```
temp=random.randint(0,100)

Humid=random.randint(0,100)

data = { 'temp': temp, 'Humid':

Humid } #printdata

def myOnPublishCallback():

print ("Published Temperature = %sC" % temp, "Humidity = %s %%"

% Humid, "to IBM Watson")

success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,

on_publish=myOnPublishCallback)

if not success:

print("Not connected

to IoTF") time.sleep(1)

deviceCli.commandCallback = myCommandCallback
```

```
# Disconnect the device and application from the

clouddeviceCli.disconnect()
```

Github and Project Demo link:

//.....Project SourceLink on Wokwi.....

Wokwi Link - https://wokwi.com/projects/347685130732569171

//.....Project Data in json Format... /

{

"version": 1,

"author": "DHINESH",

"editor":

"wokwi",

"parts": [

{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 10, "left": -60.67,

"attrs": {} },

```
{

"type": "wokwi-led",

"id": "led1",

"top": -109,

"left": -244.4,

"attrs": { "color": "red" }

},

{

"type": "wokwi-dht22",
```

"id": "dht1",

"top": -70.9,

"left": 157.2,

"attrs": { "temperature": "36.4", "humidity": "46.5" }

},

{

"type":"wokwi-ntc-temperature-

sensor", "id": "ntc1",

"top": -69.55,

"left": 253.55,

"rotate": 90,

"attrs": {}

},

{

"type": "wokwi-

resistor", "id": "r1",

"top": 169.5,

"left": -190.59,

"attrs": { "value": "5600" }

},

{

"type": "wokwi-buzzer",

"id": "bz1",

```
"top": -118.83,

"left": -378.64,

"attrs": { "volume": "0.1" }

}

],

"connections": [

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

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

[ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],

[ "dht1:SDA", "esp:D15", "green", [ "v0" ] ],

[ "ntc1:GND", "esp:GND.1", "black", [ "v0" ] ],
```

["ntc1:VCC", "esp:3V3", "red", ["v0"]],

["led1:C", "r1:1", "black", ["v0"]],

["r1:2", "esp:GND.2", "black", ["v0"]],

["led1:A", "esp:D14", "green", ["v-0.86", "h89.56", "v199.46"]],

["ntc1:OUT", "esp:D18", "green", ["v0"]],

["bz1:1", "esp:GND.2", "black", ["v0"]],

["bz1:2", "esp:D14", "green", ["v0"]],

["dht1:VCC", "esp:3V3", "red", ["v0"]],

["dht1:NC", "dht1:GND", "black", ["v0"]]

]

}