Sprint 4

Date	16 November 2022
Team ID	PNT2022TMID45187
Project Name	Smart Farmer-IoT Enabled smart Farming Application
Maximum Marks	4 Marks

INTRODUCTION:

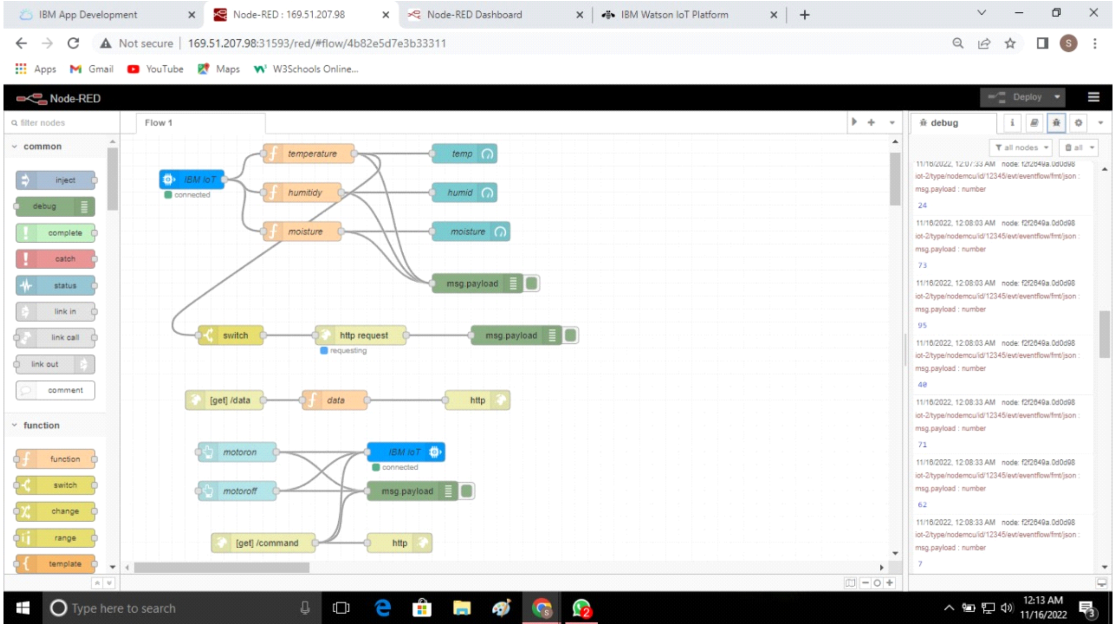
The main aim of this project is to help farmers automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity and etc and control the equipment like water motor and other devices remotely via internet without their actual presencein the field.

Sprint-4

Web UI (to make the user interact with the software) / Run a simulation using the wokwi online platform

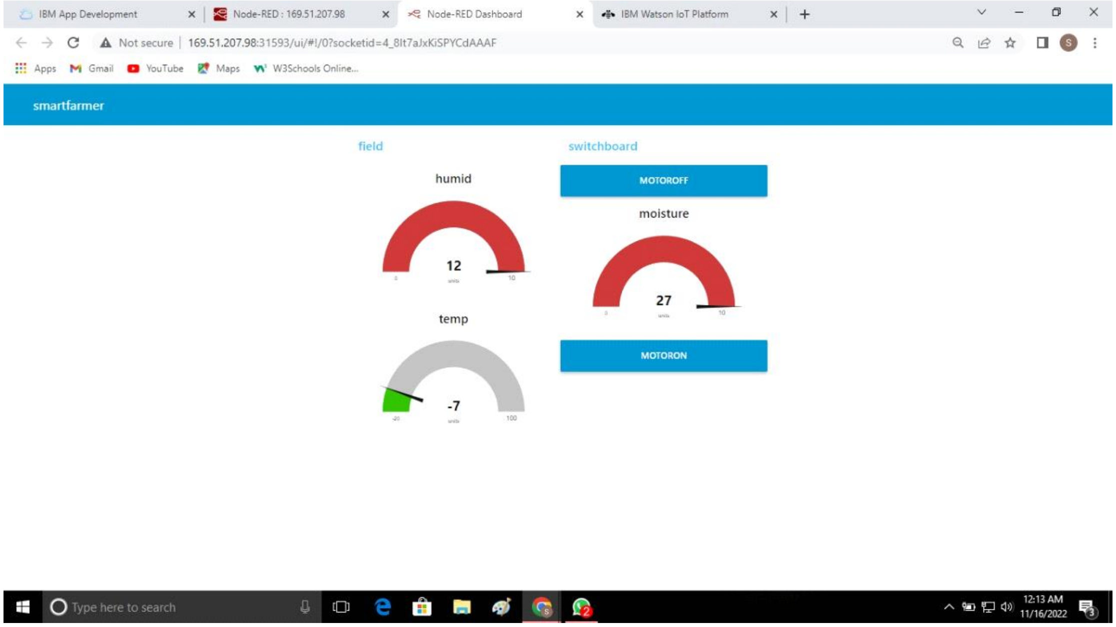
PROCESS

Using Node-Red for the Web UI process

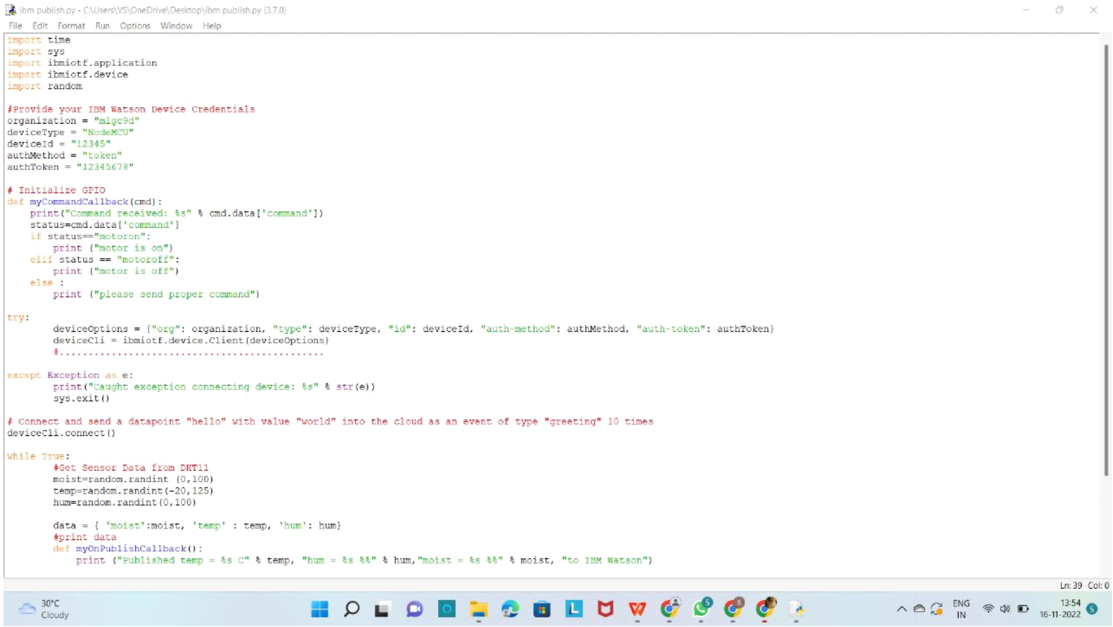


```
11/16/2022, 12:07:33 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
24
11/16/2022, 12:08:03 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
73
11/16/2022, 12:08:03 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
95
11/16/2022, 12:08:03 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
40
11/16/2022, 12:08:33 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
71
11/16/2022, 12:08:33 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
62
11/16/2022, 12:08:33 AM   node: f2f2649a.0d0d98
iot-2/type/nodemcu/id/12345/evt/eventflow/fmt/json :
msg.payload : number
7
```

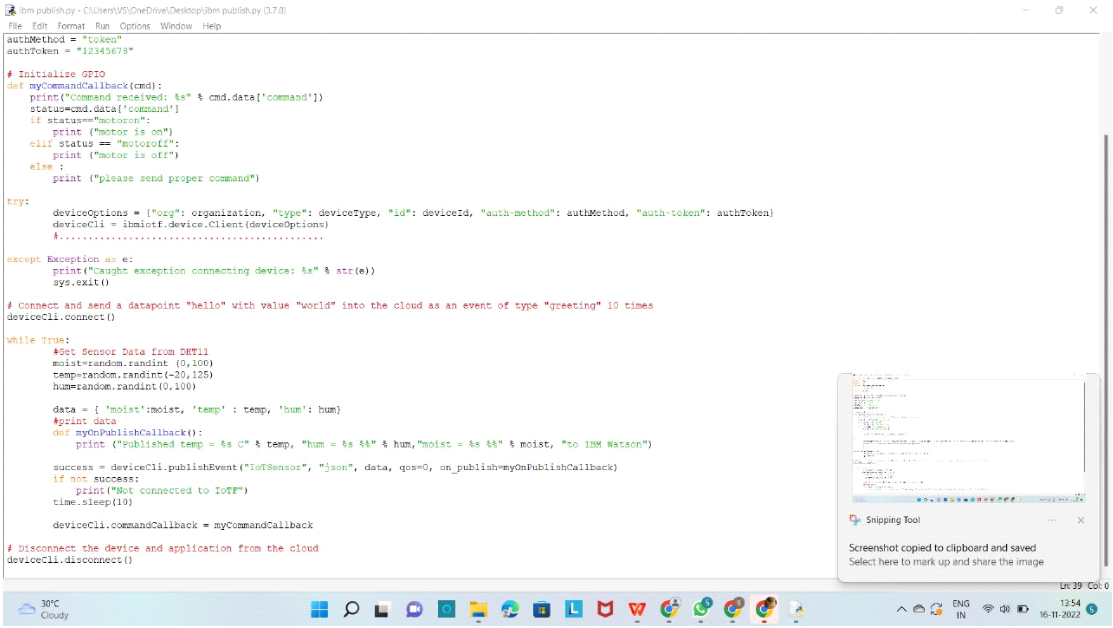
Output in Node-RED Dashboard:



PROGRAM

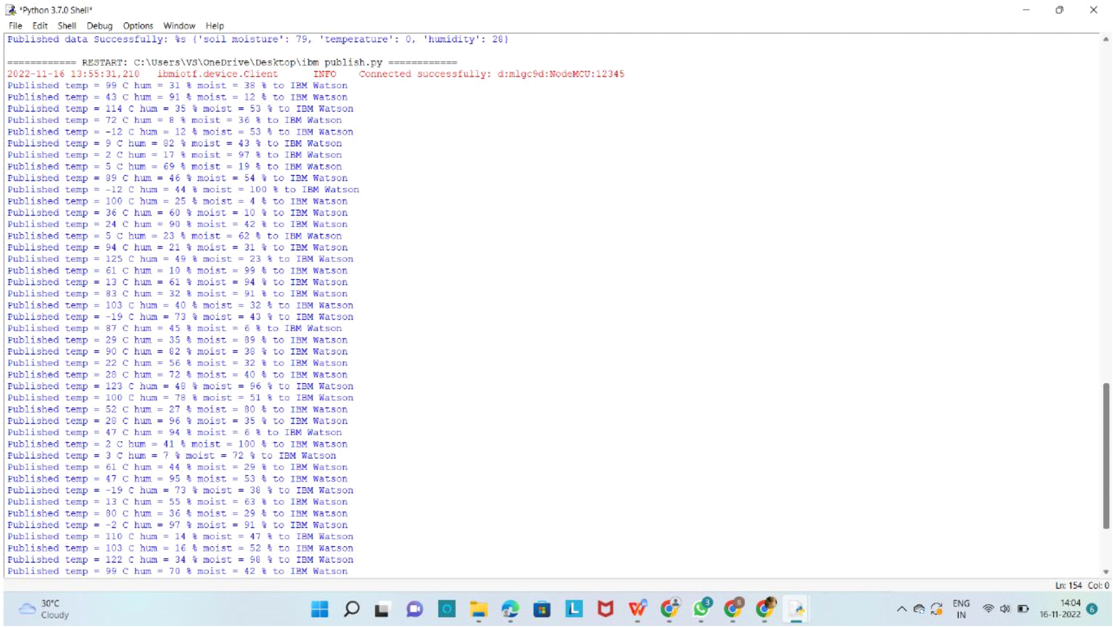

ibm publish.py - C:\Users\VS\OneDrive\Desktop\ibm publish.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 39 Col: 0
30°C
Cloudy
ENG
IN
13:54
16-11-2022

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "mlgc9d"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")

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()

# Connect and 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
        moist=random.randint (0,100)
        temp=random.randint(-20,125)
        hum=random.randint(0,100)

        data = { 'moist':moist, 'temp' : temp, 'hum': hum}
        #print data
        def myOnPublishCallback():
            print ("Published temp = %s C" % temp, "hum = %s %%" % hum,"moist = %s %%" % moist, "to IBM Watson")
```


ibm publish.py - C:\Users\VS\OneDrive\Desktop\ibm publish.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 39 Col: 0
30°C
Cloudy
ENG
IN
13:54
16-11-2022
Snipping Tool

```
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")

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()

# Connect and 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
        moist=random.randint (0,100)
        temp=random.randint(-20,125)
        hum=random.randint(0,100)

        data = { 'moist':moist, 'temp' : temp, 'hum': hum}
        #print data
        def myOnPublishCallback():
            print ("Published temp = %s C" % temp, "hum = %s %%" % hum,"moist = %s %%" % moist, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Screenshot copied to clipboard and saved
Select here to mark up and share the image



```
Published data Successfully: %s {'soil moisture': 79, 'temperature': 0, 'humidity': 28}

============ RESTART: C:\Users\VS\OneDrive\Desktop\ibm publish.py ============
2022-11-16 13:55:31,210   ibmiotf.device.Client      INFO    Connected successfully: d:mlgc9d:NodeMCU:12345
Published temp = 99 C hum = 31 % moist = 38 % to IBM Watson
Published temp = 43 C hum = 91 % moist = 12 % to IBM Watson
Published temp = 114 C hum = 35 % moist = 53 % to IBM Watson
Published temp = 72 C hum = 8 % moist = 36 % to IBM Watson
Published temp = -12 C hum = 12 % moist = 53 % to IBM Watson
Published temp = 9 C hum = 82 % moist = 43 % to IBM Watson
Published temp = 2 C hum = 17 % moist = 97 % to IBM Watson
Published temp = 5 C hum = 69 % moist = 19 % to IBM Watson
Published temp = 89 C hum = 46 % moist = 54 % to IBM Watson
Published temp = -12 C hum = 44 % moist = 100 % to IBM Watson
Published temp = 100 C hum = 25 % moist = 4 % to IBM Watson
Published temp = 36 C hum = 60 % moist = 10 % to IBM Watson
Published temp = 24 C hum = 90 % moist = 42 % to IBM Watson
Published temp = 5 C hum = 23 % moist = 62 % to IBM Watson
Published temp = 94 C hum = 21 % moist = 31 % to IBM Watson
Published temp = 125 C hum = 49 % moist = 23 % to IBM Watson
Published temp = 61 C hum = 10 % moist = 99 % to IBM Watson
Published temp = 13 C hum = 61 % moist = 94 % to IBM Watson
Published temp = 83 C hum = 32 % moist = 91 % to IBM Watson
Published temp = 103 C hum = 40 % moist = 32 % to IBM Watson
Published temp = -19 C hum = 73 % moist = 43 % to IBM Watson
Published temp = 87 C hum = 45 % moist = 6 % to IBM Watson
Published temp = 29 C hum = 35 % moist = 89 % to IBM Watson
Published temp = 90 C hum = 82 % moist = 38 % to IBM Watson
Published temp = 22 C hum = 56 % moist = 32 % to IBM Watson
Published temp = 28 C hum = 72 % moist = 40 % to IBM Watson
Published temp = 123 C hum = 48 % moist = 96 % to IBM Watson
Published temp = 100 C hum = 78 % moist = 51 % to IBM Watson
Published temp = 52 C hum = 27 % moist = 80 % to IBM Watson
Published temp = 28 C hum = 96 % moist = 35 % to IBM Watson
Published temp = 47 C hum = 94 % moist = 6 % to IBM Watson
Published temp = 2 C hum = 41 % moist = 100 % to IBM Watson
Published temp = 3 C hum = 7 % moist = 72 % to IBM Watson
Published temp = 61 C hum = 44 % moist = 29 % to IBM Watson
Published temp = 47 C hum = 95 % moist = 53 % to IBM Watson
Published temp = -19 C hum = 73 % moist = 38 % to IBM Watson
Published temp = 13 C hum = 55 % moist = 63 % to IBM Watson
Published temp = 80 C hum = 36 % moist = 29 % to IBM Watson
Published temp = -2 C hum = 97 % moist = 91 % to IBM Watson
Published temp = 110 C hum = 14 % moist = 47 % to IBM Watson
Published temp = 103 C hum = 16 % moist = 52 % to IBM Watson
Published temp = 122 C hum = 34 % moist = 98 % to IBM Watson
Published temp = 99 C hum = 70 % moist = 42 % to IBM Watson
```

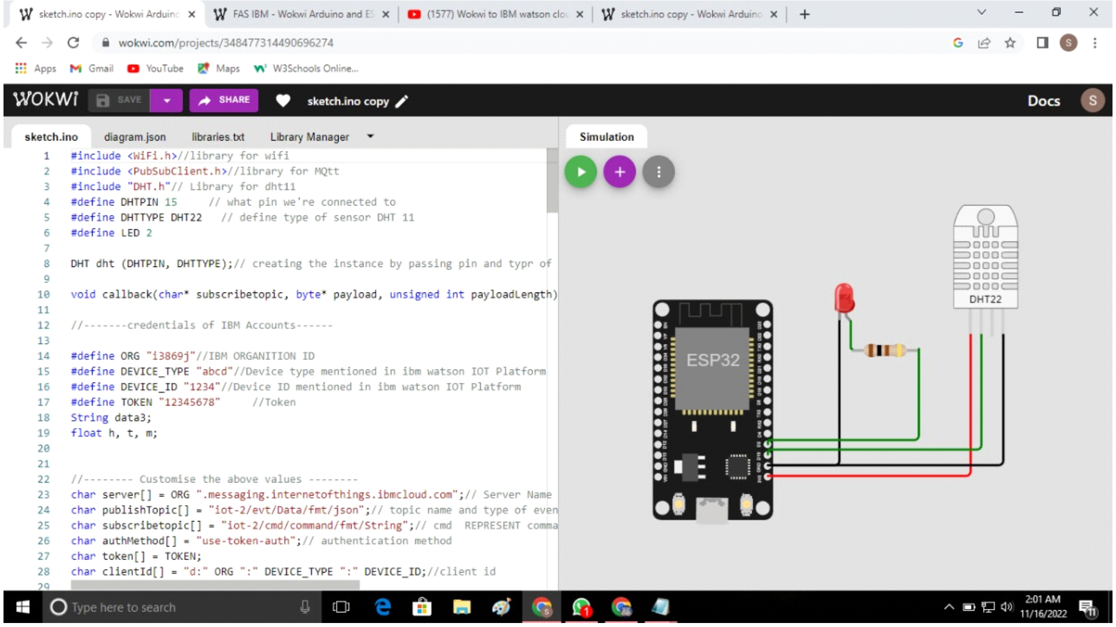
WOKWI

Create your project in the online platform of the WOKWI and execute it using the IBM Credential.

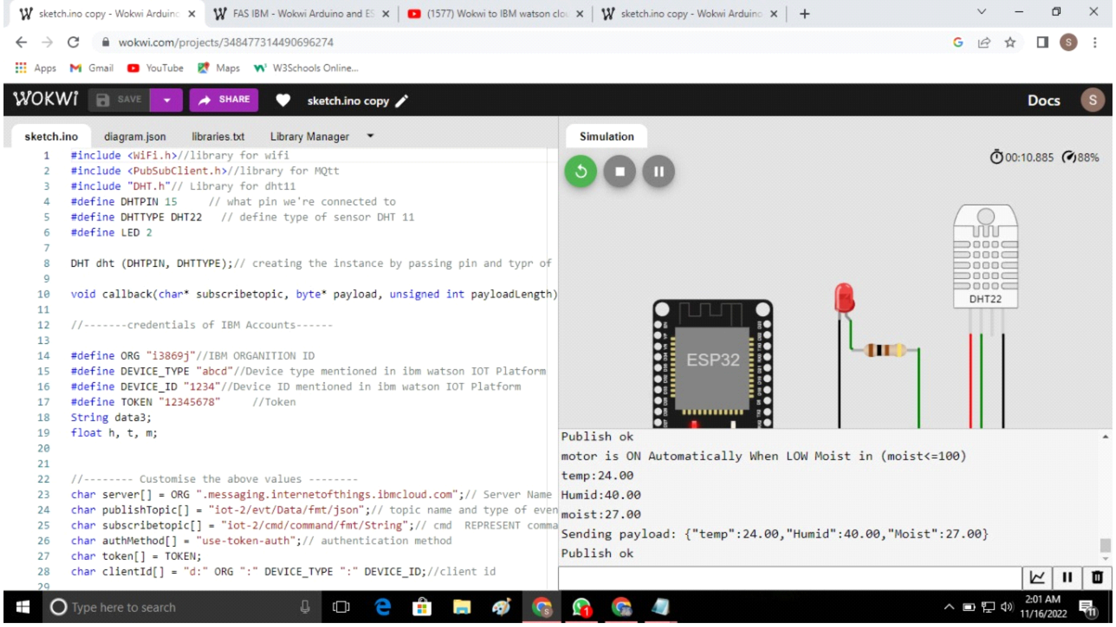
CODE

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

#include <PubSubClient.h>//library for MQtt

#include "DHT.h"// Library for dht11

#define DHTPIN 15     // what pin we're connected to

#define DHTTYPE DHT22   // define type of sensor DHT 11

#define LED 2


DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected


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


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


#define ORG "i3869j"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "abcd"//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, m;


//-------- 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/command/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


void setup()// configureing the ESP32
{
  Serial.begin(115200);
  dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
  Serial.println();
  randomSeed(analogRead(0));
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  h = dht.readHumidity();
  t = dht.readTemperature();
  m = random(100);
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
  Serial.println(h);
  Serial.print("moist:");
  Serial.println(m);

  PublishData(t, h,m);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
   {
    if(m<=100){
      Serial.print("motor is ON Automatically When LOW Moist in (moist<=100) ");
      Serial.print("\n");
    }
    else{
      Serial.println("Moist level is normal");
    }
  }
}
```



```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of

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

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

#define ORG "i3869j"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//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, m;


//-------- 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 even
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT comma
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```



```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of

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

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

#define ORG "i3869j"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//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, m;


//-------- 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 even
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT comma
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

```
Publish ok
motor is ON Automatically When LOW Moist in (moist<=100)
temp:24.00
Humid:40.00
moist:27.00
Sending payload: {"temp":24.00,"Humid":40.00,"Moist":27.00}
Publish ok
```

LINK:

https://wokwi.com/projects/348343557862457939