Sprint Delivery – 1

Team Id	PNT2022TMID16026
Project Name	Smart Farmer – IOT Enabled Smart Farming Application

1. Introduction

Agriculture is the backbone of the Indian Economy"- said Mahatma Gandhi. The main aim of our project is to help farmers with a Web App to monitor Temperature, soil moisture, humidity and to control water motor remotely via internet without going to their field.

2. Problem Statement

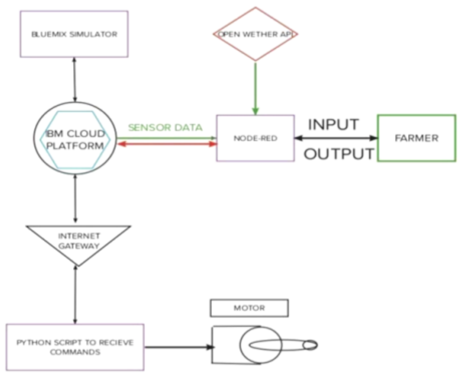
Farmers should be present in their field anytime irrespective of their health, climatic conditions even without considering their family time. They have to check the soil moisture, Temperature, Humidity before watering the crops and also ensure that the crops are well watered.

3. Proposed Solution

We aim to help the Farmers and provide easier working environment also accurate. We introduce IOT services to them which connect cloud services and internet to ensure that farmers can work remotely via internet. Also, He can monitor the field parameters and control the devices in farm.

4. Theoretical Analysis

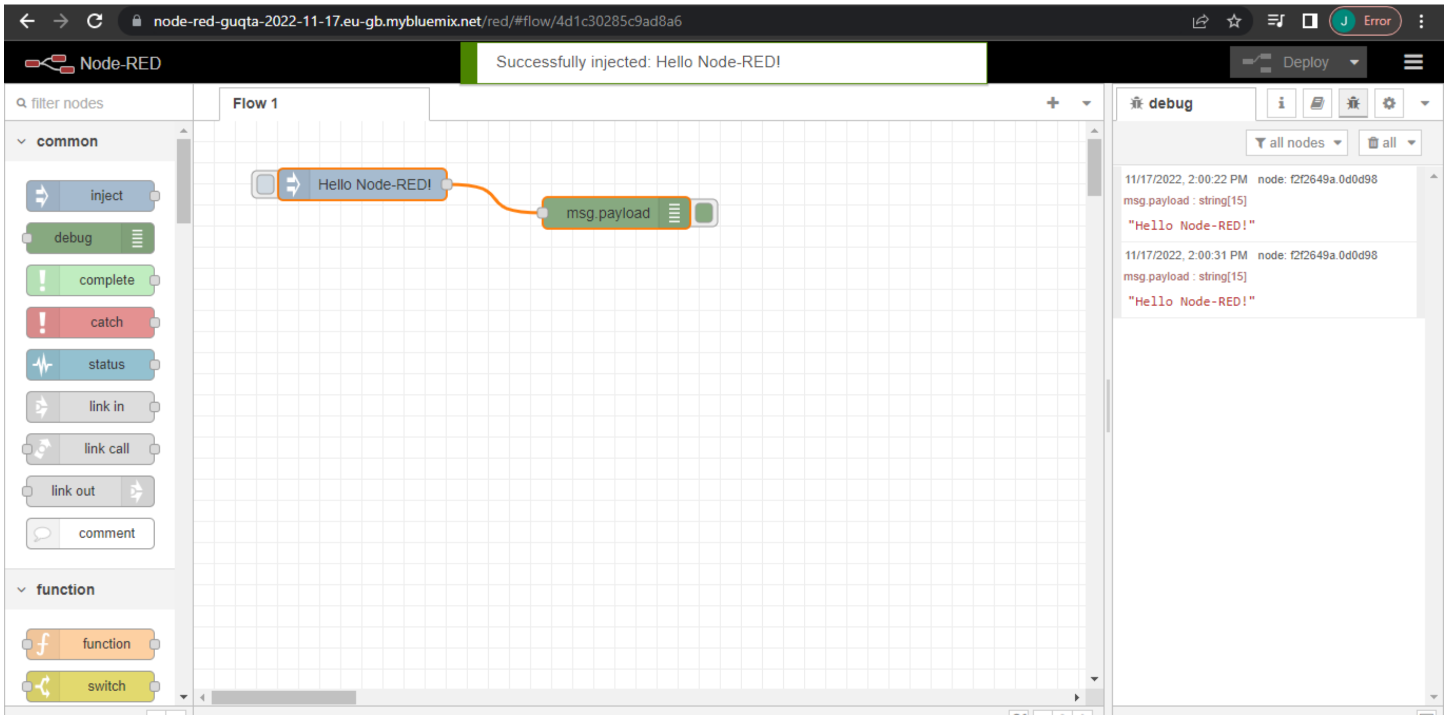
4.1 Block Diagram



4.2 Required Software Installation

4.2.1 Node-Red

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.



```
node-red-guqta-2022-11-17.eu-gb.mybluemix.net/red/#flow/4d1c30285c9ad8a6
```

Successfully injected: Hello Node-RED!

```
11/17/2022, 2:00:22 PM   node: f2f2649a.0d0d98
msg.payload : string[15]
"Hello Node-RED!"
11/17/2022, 2:00:31 PM   node: f2f2649a.0d0d98
msg.payload : string[15]
"Hello Node-RED!"
```

- First install npm/node.js
- Open cmd prompt
- Type => npm install node-red

To run the application :

- Open cmd prompt
- Type=>node-red
- Then open http://localhost:1880/ in browser

Installation of IBM IoT and Dashboard nodes for Node-Red

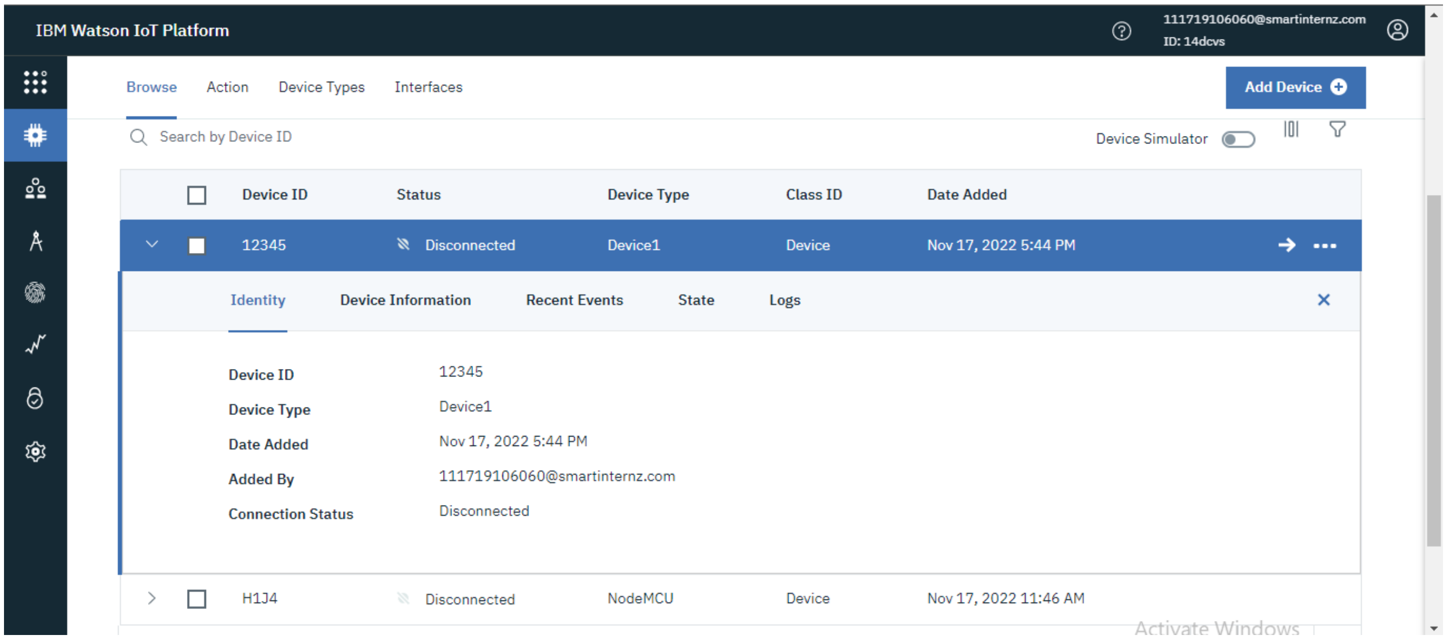
In order to connect to IBM Watson IoT platform and create the Web App UI these nodes are required 1. IBM IoT node 2. Dashboard node

4.2.2 IBM Watson IoT Platform

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.

Steps to configure:

- Create an account in IBM cloud using your email ID
- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.

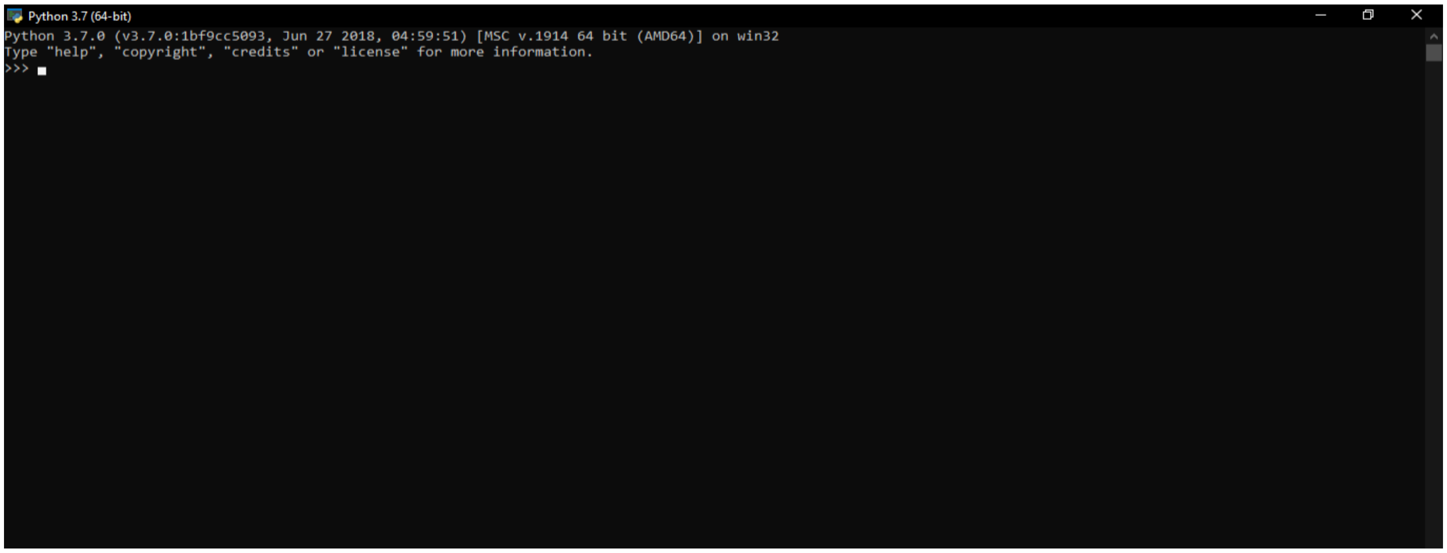


Device ID	Status	Device Type	Class ID	Date Added
12345	Disconnected	Device1	Device	Nov 17, 2022 5:44 PM
H1J4	Disconnected	NodeMCU	Device	Nov 17, 2022 11:46 AM

Device ID	12345
Device Type	Device1
Date Added	Nov 17, 2022 5:44 PM
Added By	111719106060@smartinternz.com
Connection Status	Disconnected

4.2.3 Python IDE

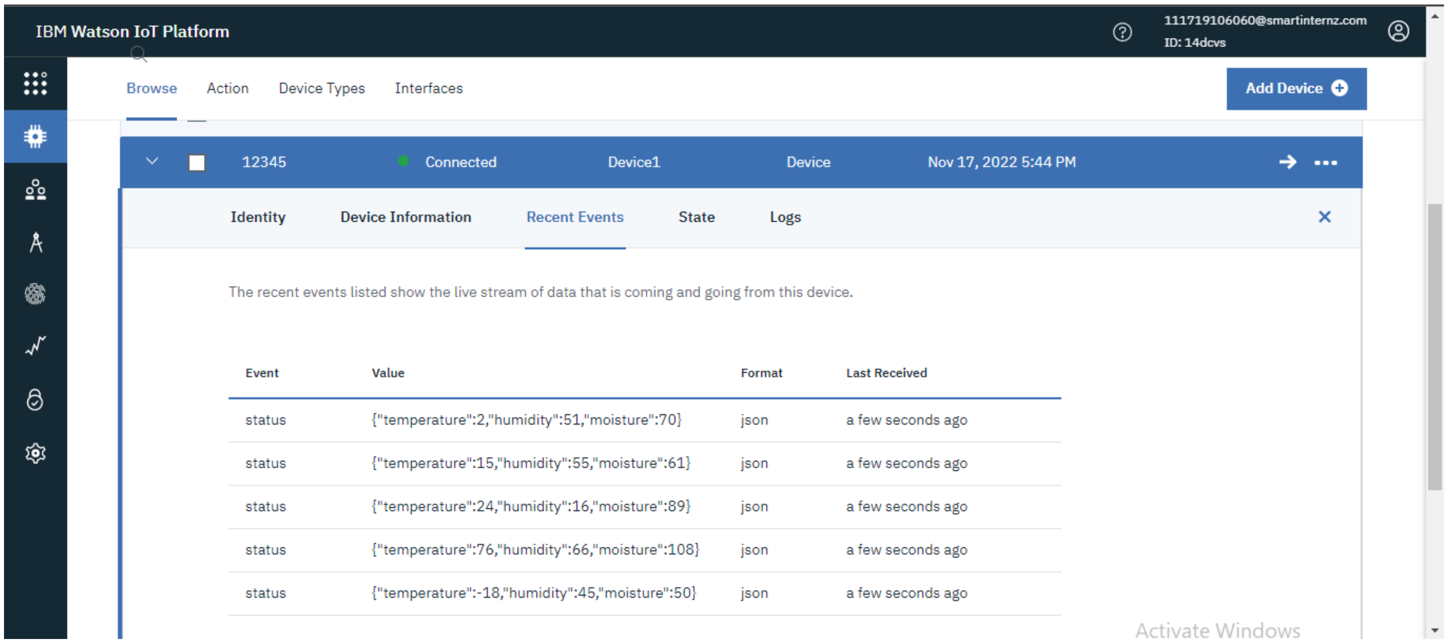
Install Python3 compiler Install any python IDE to execute python scripts


```
Python 3.7 (64-bit)
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

CODE:

```
    #IBM Watson IOT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
myConfig = {
  "identity": {
    "orgId": "14dcvs",
    "typeId": "Device1",
    "deviceId":"12345"
  },
  "auth": {
    "token": "87654321"
  }
}
```

```
def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if(m=="Motor On"):
        print("****///Motors ARE ON///****")
    else:
        print("****///Motors ARE OFF///****")


client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()


while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    Mois=random.randint(20,120)
    myData={'temperature':temp, 'humidity':hum, 'moisture':Mois}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect ()
```



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

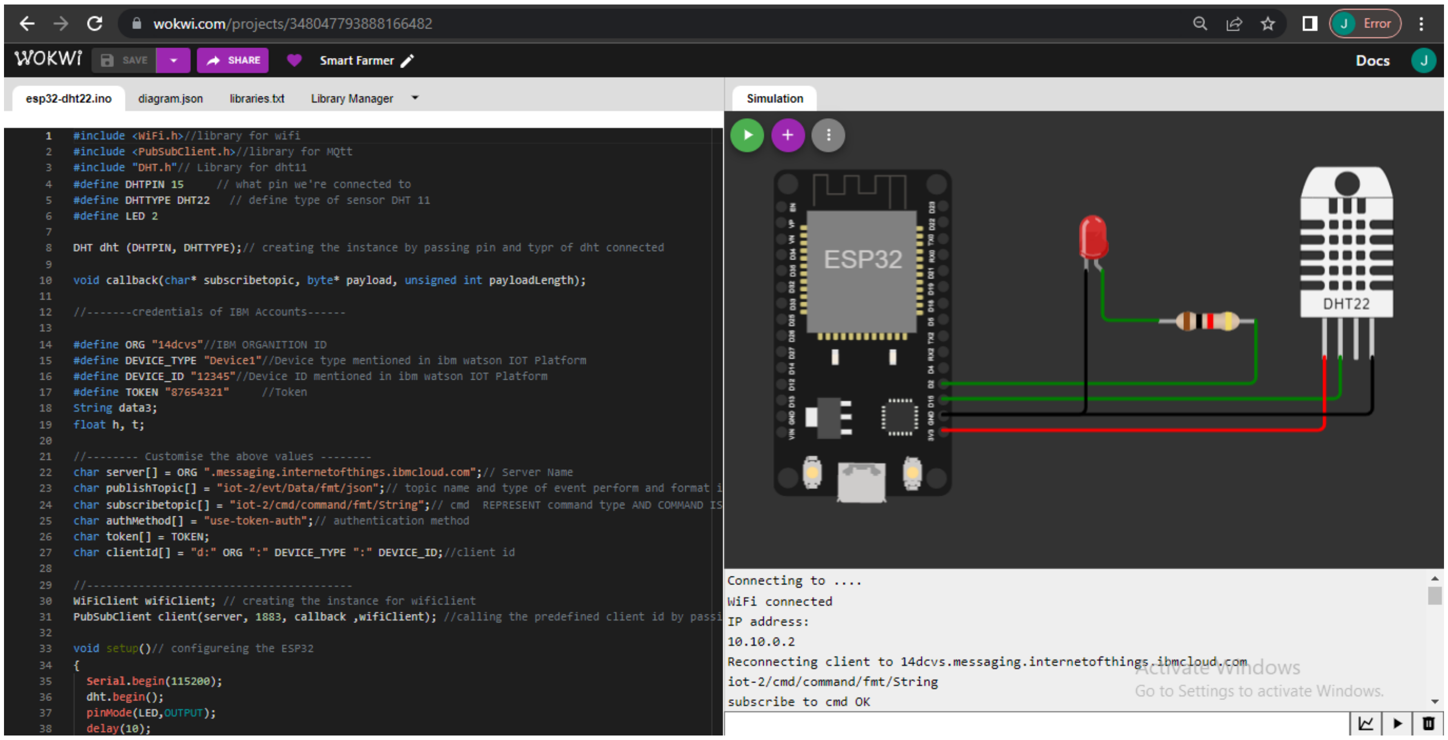
Event	Value	Format	Last Received
status	{"temperature":2,"humidity":51,"moisture":70}	json	a few seconds ago
status	{"temperature":15,"humidity":55,"moisture":61}	json	a few seconds ago
status	{"temperature":24,"humidity":16,"moisture":89}	json	a few seconds ago
status	{"temperature":76,"humidity":66,"moisture":108}	json	a few seconds ago
status	{"temperature":-18,"humidity":45,"moisture":50}	json	a few seconds ago

Arduino Code In Wokwi:

```
#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 "14dcvs"//IBM ORGANITION ID
#define DEVICE_TYPE "Device1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "87654321"     //Token
String data3;
float h, t;


//-------- 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();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

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

  PublishData(t, h);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



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

void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"temp\":";
  payload += temp;
  payload += "," "\"Humid\":";
  payload += humid;
  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];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);
  }
  else
  {
Serial.println(data3);
digitalWrite(LED,LOW);
  }
data3="";
}
```



```
#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 "14dcvs"//IBM ORGANITION ID
#define DEVICE_TYPE "Device1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "87654321"     //Token
String data3;
float h, t;

//-------- 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 i
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND COMMAND IS
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 passi

void setup()// configureing the ESP32
{
  Serial.begin(115200);
  dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
```

```
Connecting to ....
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 14dcvs.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK
```