Smart Farmer-IOT Enabled Smart Farming Application

Project Development Phase

Sprint Delivery-1

1.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 etc and control the equipment like water motor and other devices remotely via the internet without their actual presence in the field.

2.Problem Statement:

Farmers need to deal with many problems like coping with climate change, soil erosion and Biodiversity loss. Farmers are to be present at farm for its maintenance irrespective of the weather conditions.They have to ensure that the crops are well watered and the farm status is monitored by them physically.Farmers have to stay most of the time in field in order to get a good yield.In difficult times like in the presence of pandemic also they have to work hard in their fields risking their lives to provide food for the country.

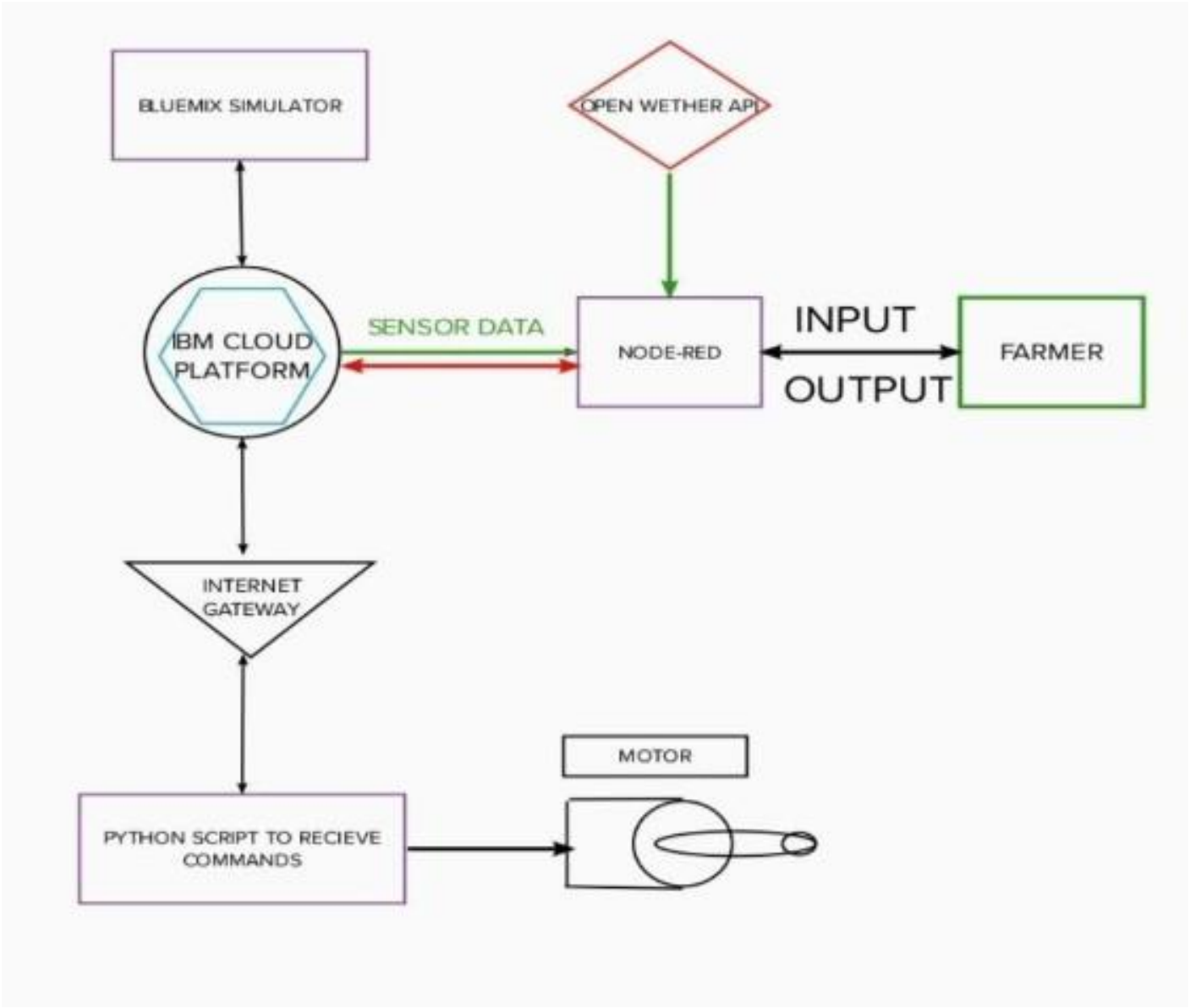
3.Proposed Solution:

To provide an efficient decision support system using wireless sensor networks which handle different activities of the farm and give useful information related to soil moisture, Temperature and Humidity content. Due to the weather condition, water level increases, Farmers get a lot of distractions which is not good for Agriculture.

2.Theoretical Analysis:

1.1 Block Diagram:

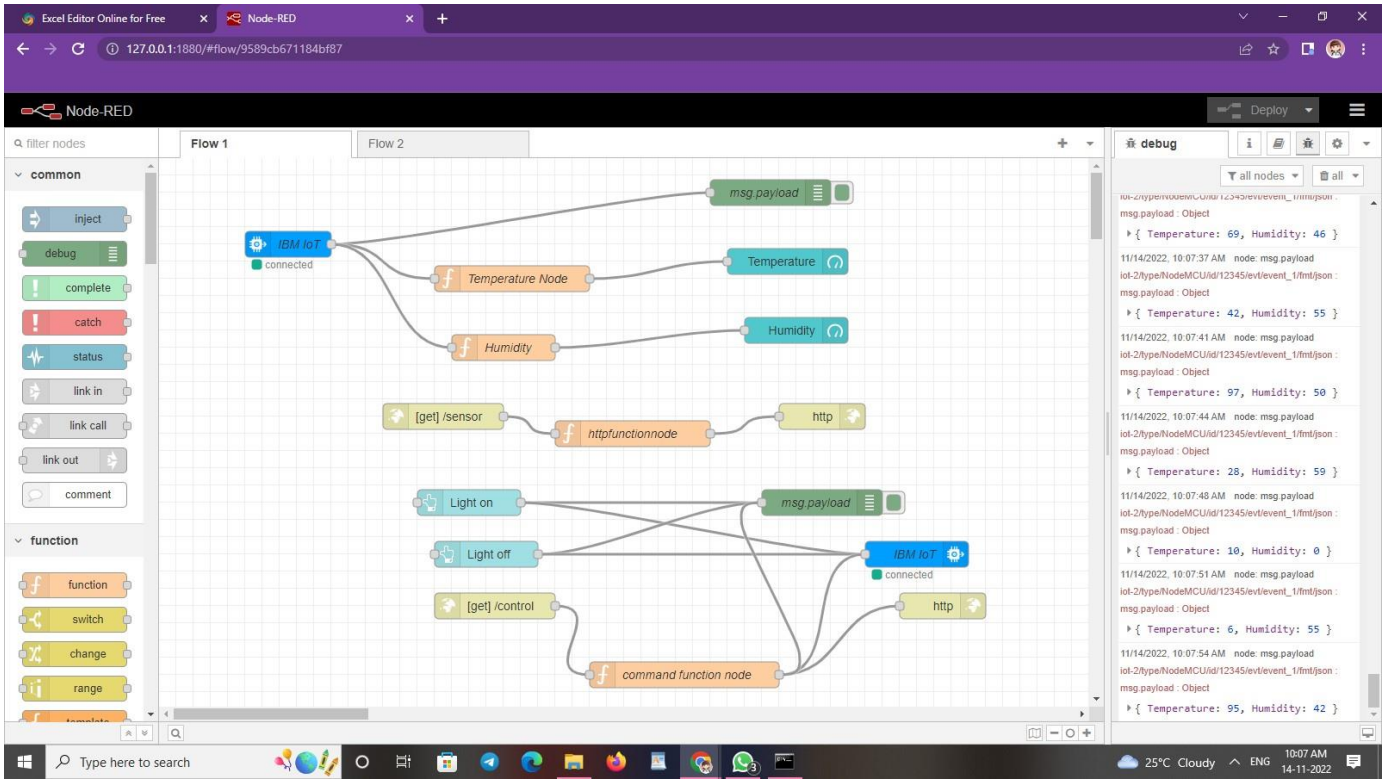
In order to implement the solution , the following approach as shown in the block diagram,is used


BLUEMIX SIMULATOR
OPEN WETHER API
IBM CLOUD PLATFORM
SENSOR DATA
NODE-RED
INPUT
OUTPUT
FARMER
INTERNET GATEWAY
MOTOR
PYTHON SCRIPT TO RECIEVE COMMANDS

1.2.Required Software Installation:

1.2.A.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.



Installation :

- 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://127.0.0.1:1880/ on your browser.

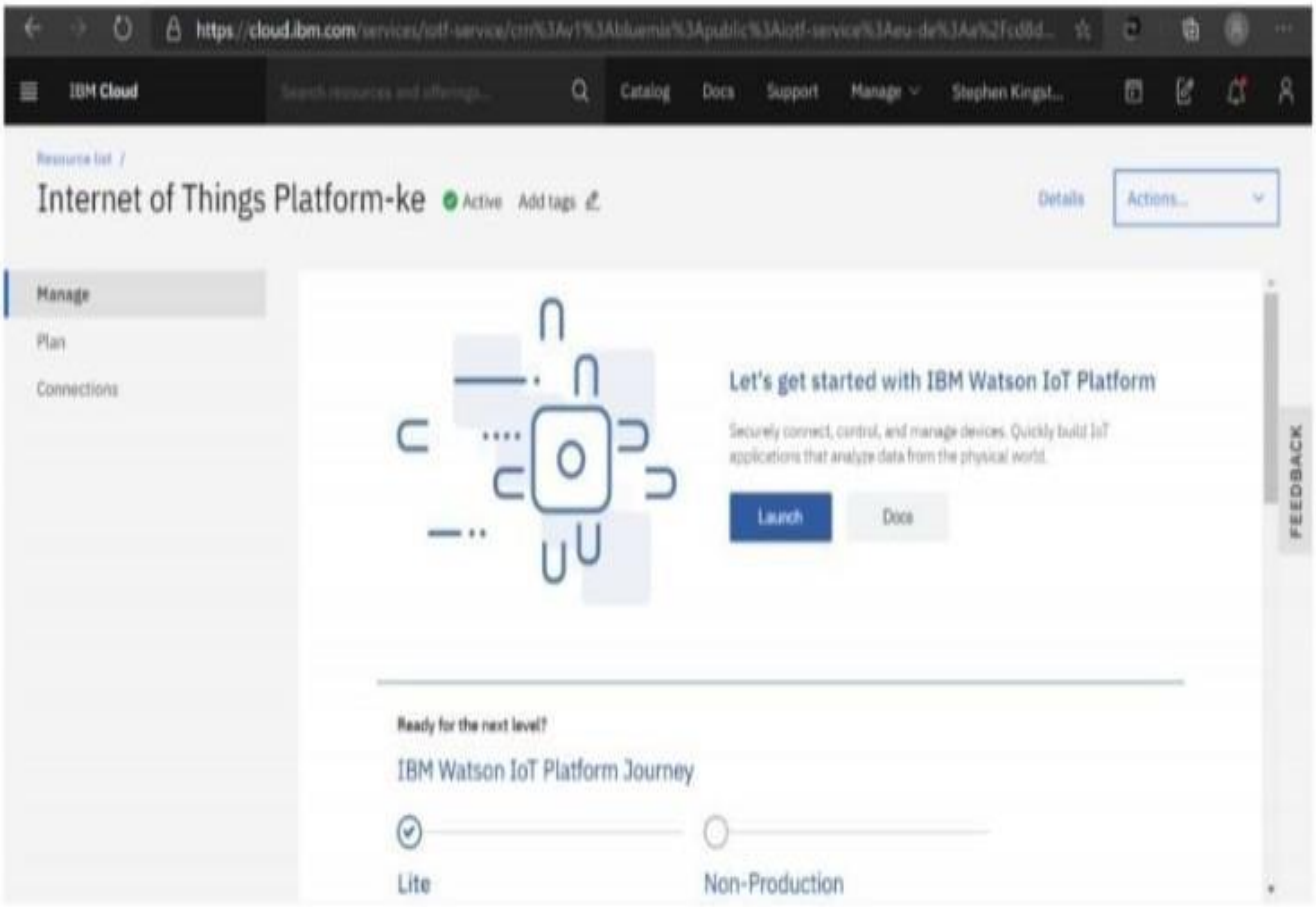
Installation of IBM IoT and Dashboard nodes for Node RED:

In order to connect to IBM Watson IoT platform and create the WEB UI these nodes are required.

1. IBM IoT node.
2. Dashboard node.

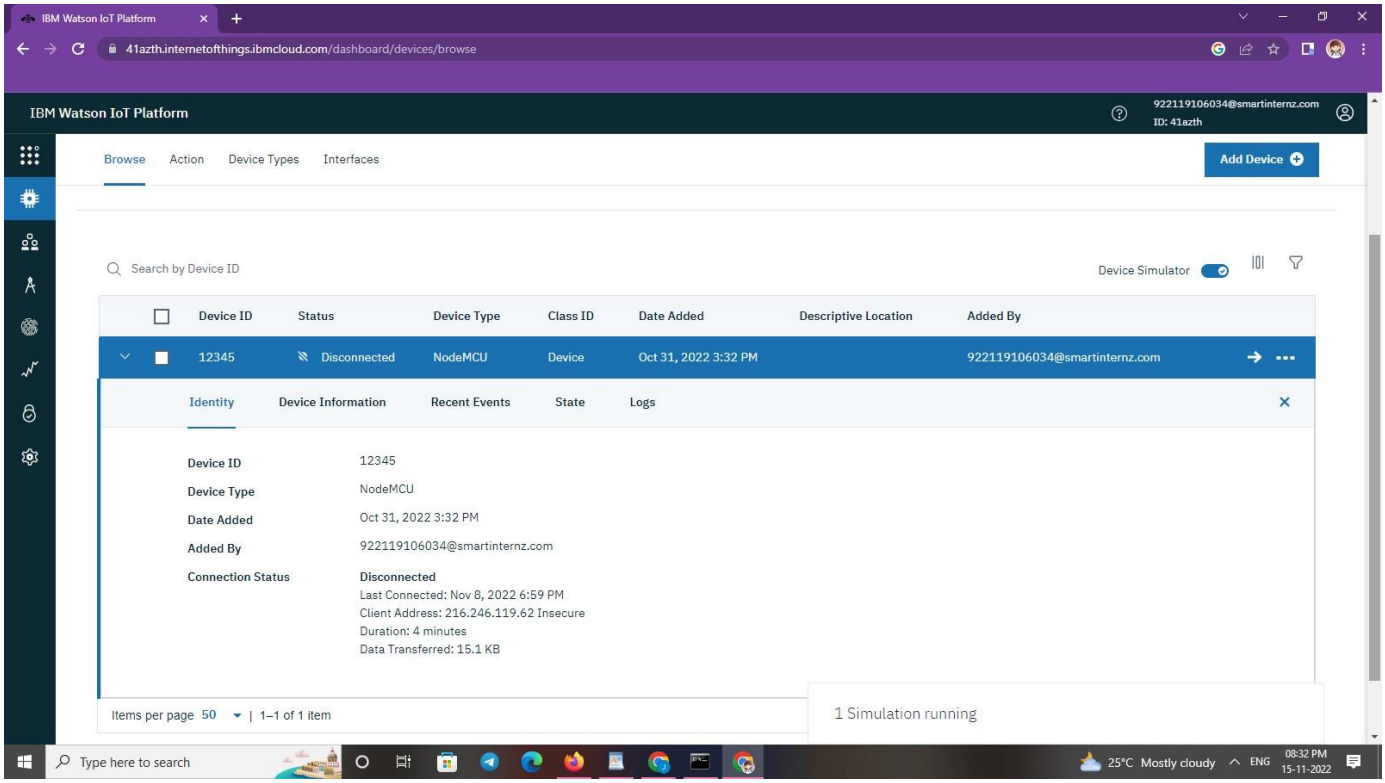
1.1.B IBM Watson IoT Platform:

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.

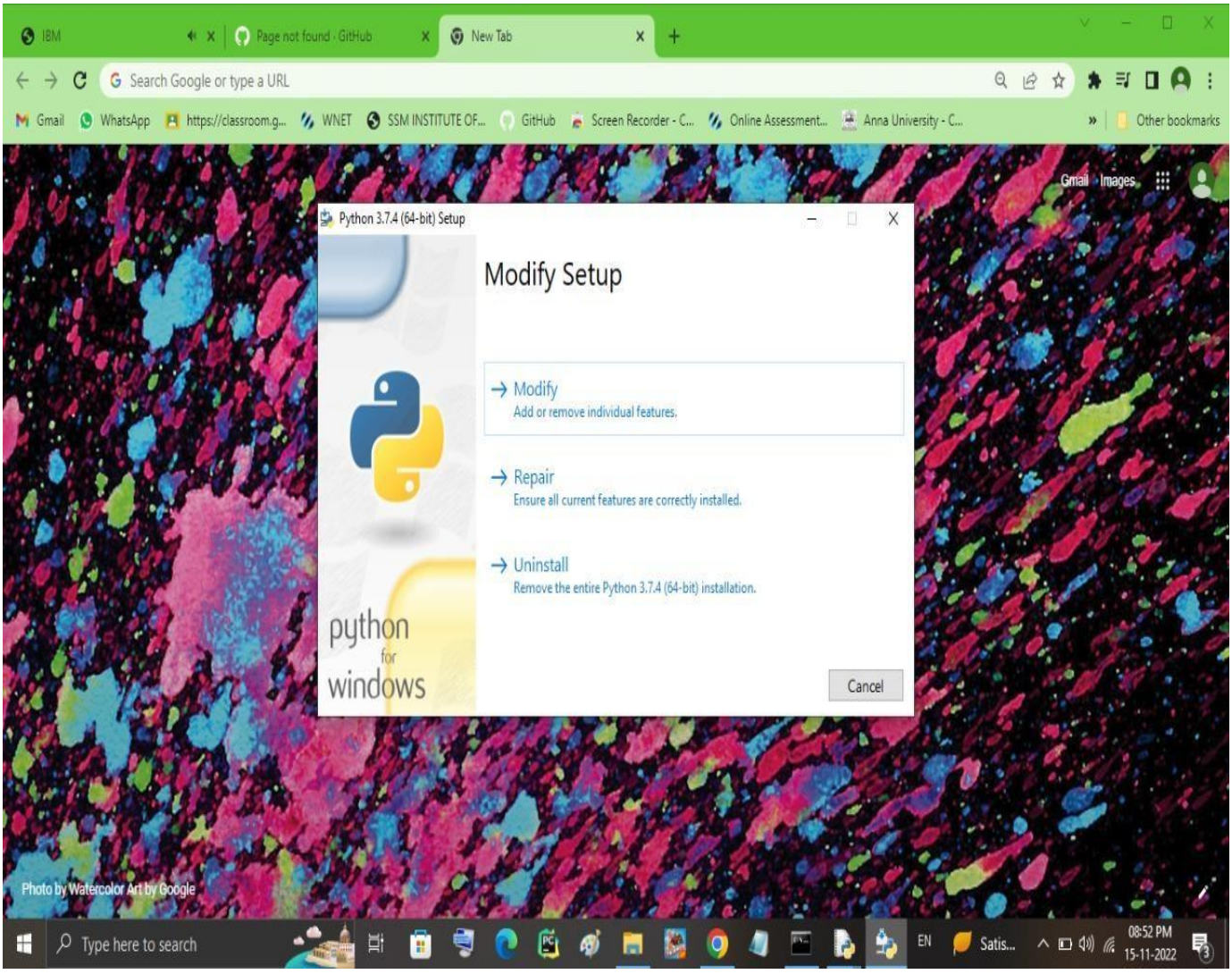


.

1.1.C Python IDE:

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used Command Pro



mpt to execute.

Code:

```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig = {
"identity": {
"orgId": "u9qhfi",
"typeId": "Devicetypel",
"deviceId": "DeviceID1"
},
"auth": {
"token": ")hSb7_ZD+evl2fRhXi"
} }
client=wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=
None)
client.connect ()
def myCommandCallback (cmd) :
```

```
    print ("Message received from IBM IoT Platform:
    %s"%   cmd.data['command'])
        m=cmd.data['command']
        if (m=="motoron"):
            print ("Motor is switched on")
            elif (m=="motoroff"):
                print ("Motor is switched OFF")
            print (" ")
while True:
        soil=random.randint (0,100)
        temp=random.randint (-20, 125)
        hum=random.randint (0, 100)
        myData={'soil moisture': soil, 'temperature':temp, 'humidity':hum}
            client.publishEvent (eventId="status",msgFormat=
            "json", data=myData, qos=0 ,onPublish=None)
            print ("Published data Successfully: %s", myData)
            time.sleep (2)
client.commandCallback = myCommandCallback
client.disconnect ()
```

Arduino code for C :

```
#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
and type of dht connected

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

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

#define ORG "u9qhfi"//IBM ORGANIZATION ID
#define DEVICE_TYPE "Devicetype1"//Device type mentioned in
ibm watson IOT Platform
#define DEVICE_ID "DeviceID1"//Device ID mentioned in ibm
```

```
watson IOT Platform
#define TOKEN ")hSb7_ZD+ev12fRhXi"      //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("temperature:");
  Serial.println(t);
  Serial.print("Humidity:");
```

```
  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 = "{\"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");// 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];
  }
```

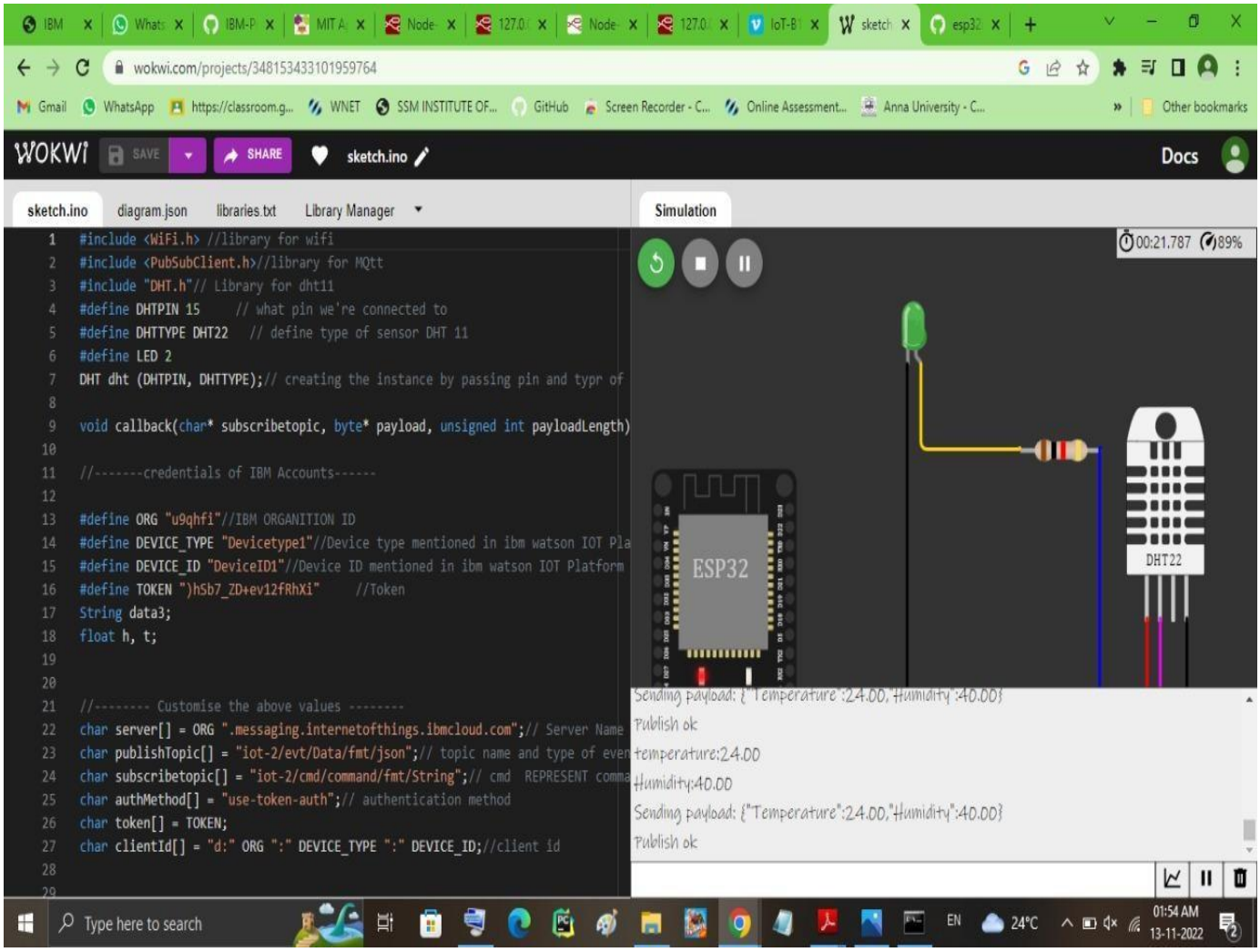
```
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);

  }

  else
  {
Serial.println(data3);
digitalWrite(LED,LOW);

  }
data3="";


}
```


wokwi.com/projects/348153433101959764
WOKWI
SAVE
SHARE
sketch.ino
Docs
sketch.ino
diagram.json
libraries.txt
Library Manager
Simulation
00:21.787
89%
ESP32
DHT22
#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 "u9qhfi"//IBM ORGANITION ID
#define DEVICE_TYPE "Devicetype1"//Device type mentioned in ibm watson IOT Pla
#define DEVICE_ID "DeviceID1"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN ")hSb7_ZD+ev12fRhXi" //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 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
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok
temperature:24.00
Humidity:40.00
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok
Type here to search
EN
24°C
01:54 AM
13-11-2022

1.2 IoT Simulator:

In our project in the place of sensors we are going to use IoT sensor simulator which give random readings to the connected Cloud.

The link to simulator:

https://41azth.internetofthings.ibmcloud.com/dashboard/devices/browse

We need to give the credentials of the created device in IBM Watson IoT Platform to connect cloud to simulator.