

Smart Farmer-IOT Enabled Smart Farming Application

Project Development Phase

Sprint Delivery -1

| | |
|-------------------|--|
| Date | 17 November 2022 |
| Team ID | PNT2022TMID33604 |
| Project Name | Smart Farmer-IOT Enabled Smart Farming Application |
| Leader Name | Selvakavin S |
| Team Members Name | Thangakavin D Yogesh Kumar S Thirumoorthi P |

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:

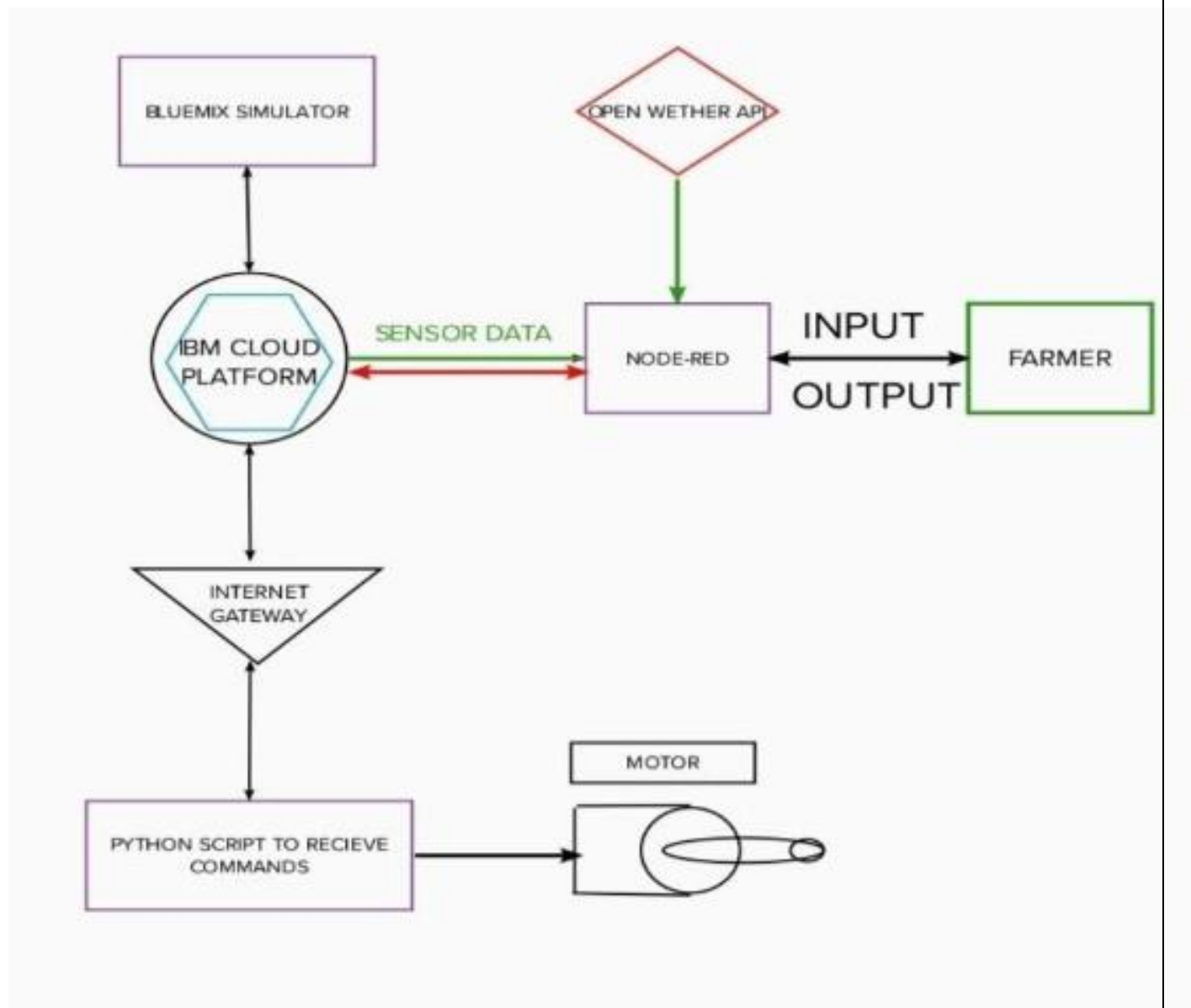
To provide an efficient decision support system using wireless sensor networks which handle different activities of the farm and give useful information related to soil moisture, Temperature and

Humidity content. Due to the weather condition, water level increases, Farmers get a lot of distractions which is not good for Agriculture.

4. Theoretical Analysis:

4.1 Block Diagram:

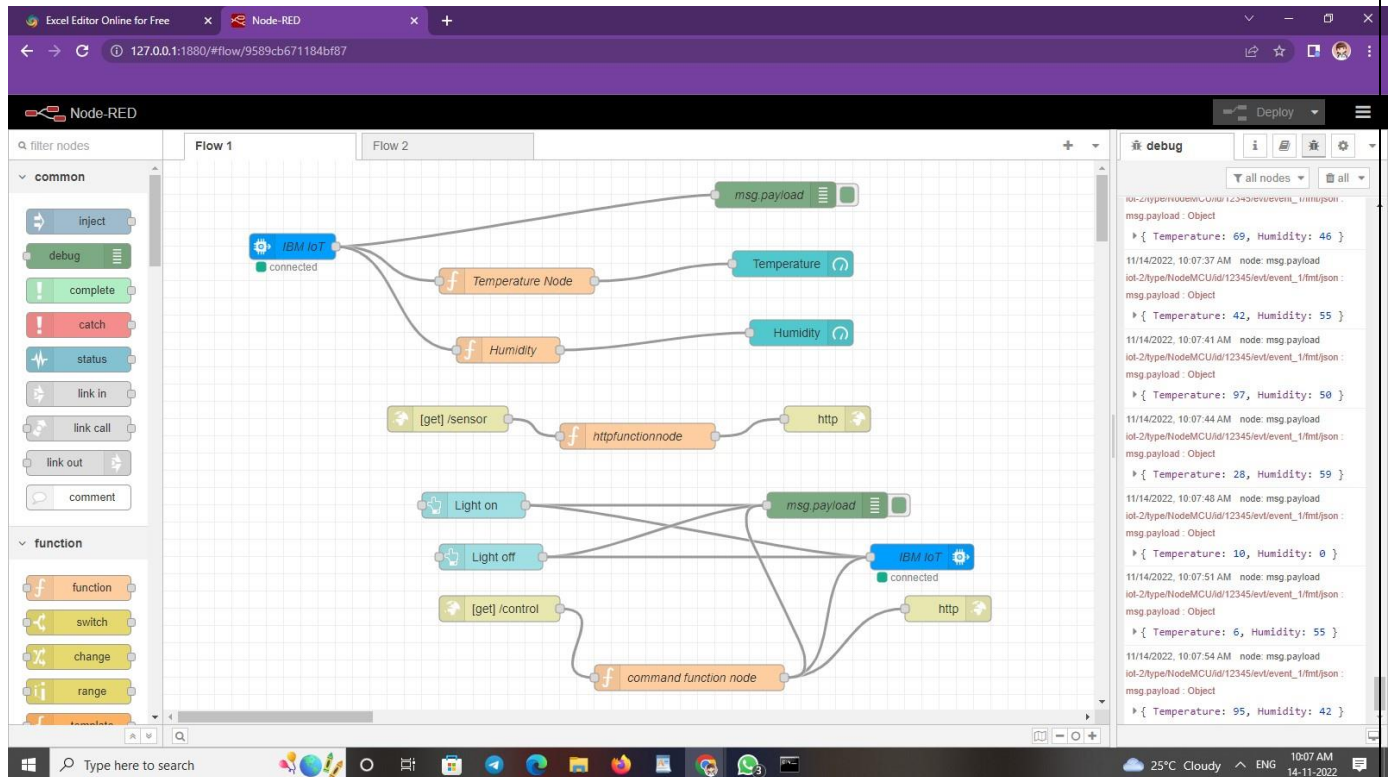
In order to implement the solution , the following approach as shown in the block diagram,is used



4.2 Required Software Installation:

4.2.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.

4.2.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.

The screenshot shows the IBM Cloud console interface for the 'Internet of Things Platform-ti' service. The browser address bar displays a long URL from cloud.ibm.com. The page header includes the IBM Cloud logo, a search bar, and navigation links for Catalog, Manage, and the user's account (Thangakavin D's Account). The main content area features a 'Manage' sidebar with links to Plan and Connections. The central section is titled 'Internet of Things Platform-ti' and includes a status indicator (Active) and an 'Add tags' link. Below this, there is a large graphic of a central square with four 'U' shaped connectors and four 'C' shaped connectors. To the right of the graphic, the text 'Let's get started with IBM Watson IoT Platform' is followed by a description: 'Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.' Below this text are 'Launch' and 'Docs' buttons. Further down, a section titled 'Ready for the next level?' introduces the 'IBM Watson IoT Platform Journey' with three stages: 'Lite', 'Non-Production', and 'Production'. Each stage has a progress indicator (a circle with a checkmark for 'Lite' and empty circles for the others) and a brief description of the service plan. The 'Lite' plan is described as a lightweight development environment. The 'Non-Production' plan is a full-featured, fully-integrated offering. The 'Production' plan is a fully managed SaaS offering for enterprise IoT data. A blue chat button is visible in the bottom right corner.

Service Details - IBM Cloud

cloud.ibm.com/services/iotf-service/cm%3Aw1%3Abluemix%3Apublic%3Aiotf-service%3Aeu-de%3Aa%2Fc983563ec9314877a8f1e6e30fe74c95%3A08b18552-b694-412d-8122-df55...

IBM Cloud

Search resources and products...

Resource list /

Internet of Things Platform-ti Active Add tags

Details Actions...

Manage

Plan

Connections

Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Launch Docs

Ready for the next level?

IBM Watson IoT Platform Journey

☒ Lite

The Lite service plan provides a lightweight development environment to get you started with the connectivity capabilities of Watson IoT Platform.

☐ Non-Production

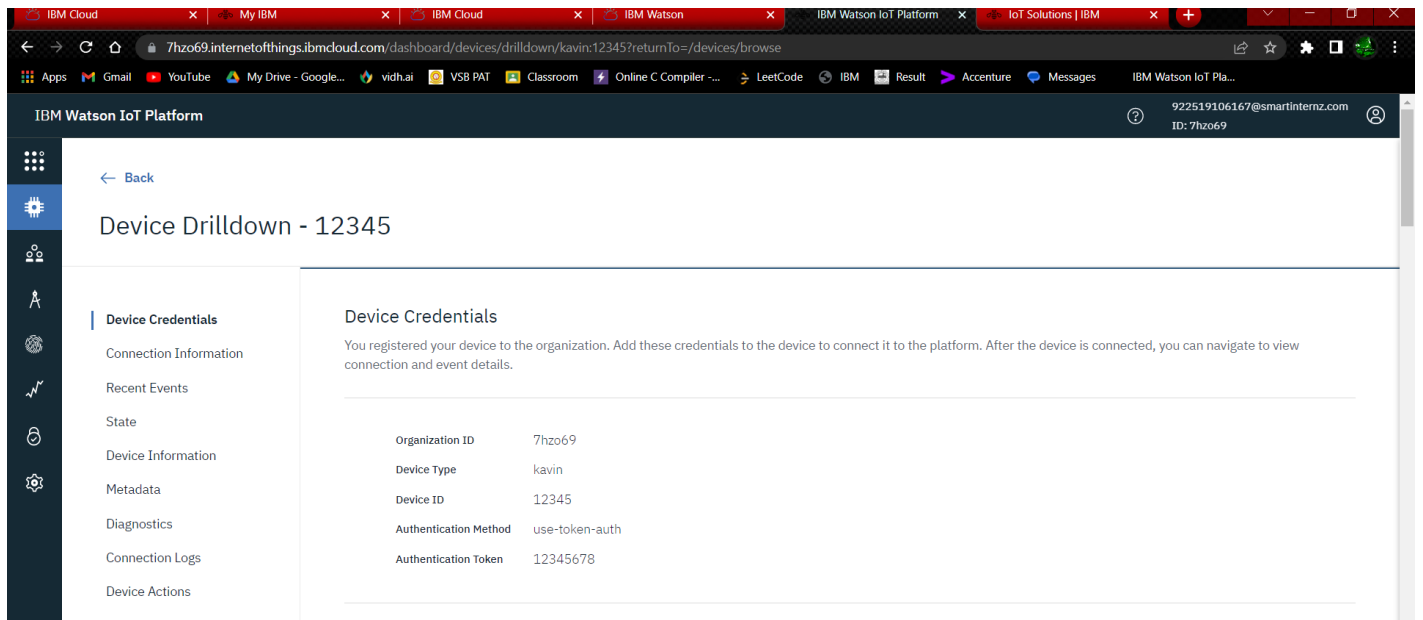
The Non-Production service plan is a full-featured, fully-integrated offering that enables you to explore Watson IoT Platform to see how the service can fit into your IoT environment.

☐ Production

The Production service is a fully managed SaaS offering that enables you to manage and analyze enterprise IoT data.

Steps to configure:

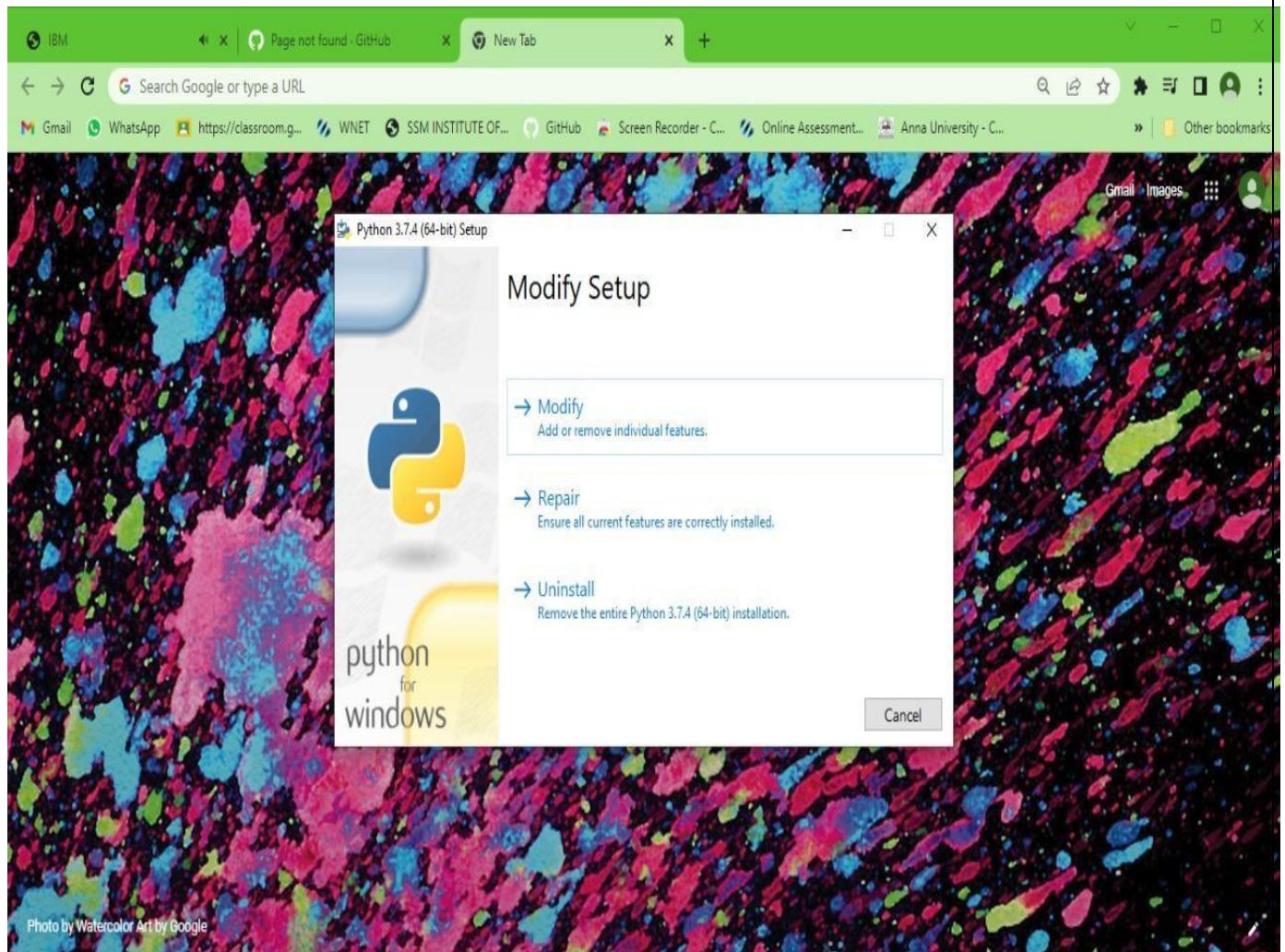
- Create an account in IBM cloud using your email ID
 - Create IBM Watson Platform in services in your IBM cloudAccount.
 - Launch the IBM Watson IoT Platform.
 - Create a new device
- .
- Give credentials like device type, device ID, Auth. Token
 - Create API key and store API key and token elsewhere.



4.2.C Python IDE:

Install Python3 compiler

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



Code:

```
import wiotp.sdk.device
import time import os
import datetime import
random myConfig = {
"identity": {
"orgId": "u9qhfi",
"typeId": "Devicetype1",
"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

SAVE SHARE sketch.ino Docs

sketch.ino diagram.json libraries.txt Library Manager

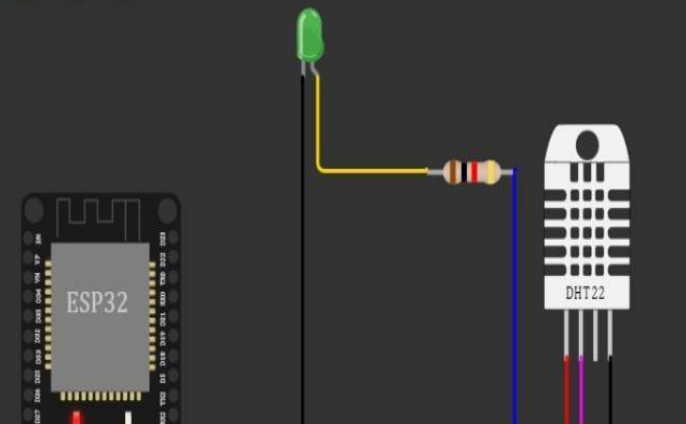
```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 2
7 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "u9qhfi" // IBM ORGANIZATION ID
14 #define DEVICE_TYPE "DeviceType1" // Device type mentioned in IBM Watson IOT Platform
15 #define DEVICE_ID "DeviceID1" // Device ID mentioned in IBM Watson IOT Platform
16 #define TOKEN "hSb7_ZD+ev12fRhXi" // Token
17 String data3;
18 float h, t;
19
20 //----- Customise the above values -----
21
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
24 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; // client id
28
29

```

Simulation

00:21.787 89%



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

4.3 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.