

# **CUSTOMER CARE REGISTRY**

## **A PROJECT REPORT**

Submitted by

<b>SULOCHANA A</b>	<b>(TEAM LEADER)</b>
<b>SWETHA S</b>	<b>(TEAM MEMBER)</b>
<b>KEERTHANA P</b>	<b>(TEAM MEMBER)</b>
<b>PAVITHRA</b>	<b>(TEAM MEMBER)</b>

*In partial fulfilment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**SALEM COLLEGE OF ENGINEERING AND TECHNOLOG**

**SALEM**



**ANNAUNIVERSITY : CHENNAI 600 02**

**JUNE 2022**

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	<b>ABSTRACT</b>	5
<b>1</b>	<b>INTRODUCTION</b>	5
	1.1 PROJECT OVERVIEW	5
<b>2</b>	<b>LITERATURE SURVEY</b>	5
	2.1 ROPOSED STATEMENT	5
	2.2 PROPOSED SOLUTION	5
<b>3</b>	<b>BLOCK DIAGRAM</b>	6
<b>4</b>	<b>CREATE DEVICE IN IBM CLOUD</b>	6
	4.1 STEP TO CREATE DEVICE IN IBM CLOUD	6
<b>5</b>	<b>IBM WATSON IOT PLATFORM</b>	9
<b>6</b>	<b>PHYTHON IDLE</b>	11
<b>7</b>	<b>IOT SIMULATOR</b>	15
<b>8</b>	<b>OPEN WEATHER API</b>	15
<b>9</b>	<b>BUILDING PROJECTS</b>	15
<b>10</b>	<b>NODE RED TO SEND COMMANDS TO IBM</b>	17

	11.1 CONFIGURATION OF NODE RED TO SEND	
	COMMANDS TO IBM	17
	11.2 ADJUSTING USER INTERFACE	19
	11.3 CREATE PROGRAM IN RED	20
<b>11</b>	<b>MOBILE APP WEB CREATION</b>	<b>22</b>
	11.1 MOBILE APP SCREENSHOT	22
	11.2 BACKEND OF MOBILE APPLICATION	23
	11.3 WEB APP UI HOME TAG	24
	11.4 COLLECTING IBM CLOUD DATA FROM NODE	
	RED	26
	11.5 COLLECTING DATA FROM OPEN WEATHER API	27
<b>12</b>	<b>OBSEVATION AND RESULT</b>	<b>32</b>
<b>13</b>	<b>ADVANTAGES</b>	
<b>14</b>	<b>CONCLUSION</b>	<b>34</b>
<b>15</b>	<b>REFERANCE</b>	<b>34</b>

## **1. INTRODUCTION**

### **1.1 Project Overview**

The main objective of this project is to provide an effective way to automate the farmer's farm land by giving them access to web application so that they can monitor the field without any personal intervention. They can monitor temperature, humidity, soil moisture etc. Based on these details he can detect the crops by controlling the motor through the app and the app gives an alert message if temperature or humidity goes beyond the threshold value

## **2 LITERATURE SURVEY**

### **2.1 PROBLEM STATEMENT:**

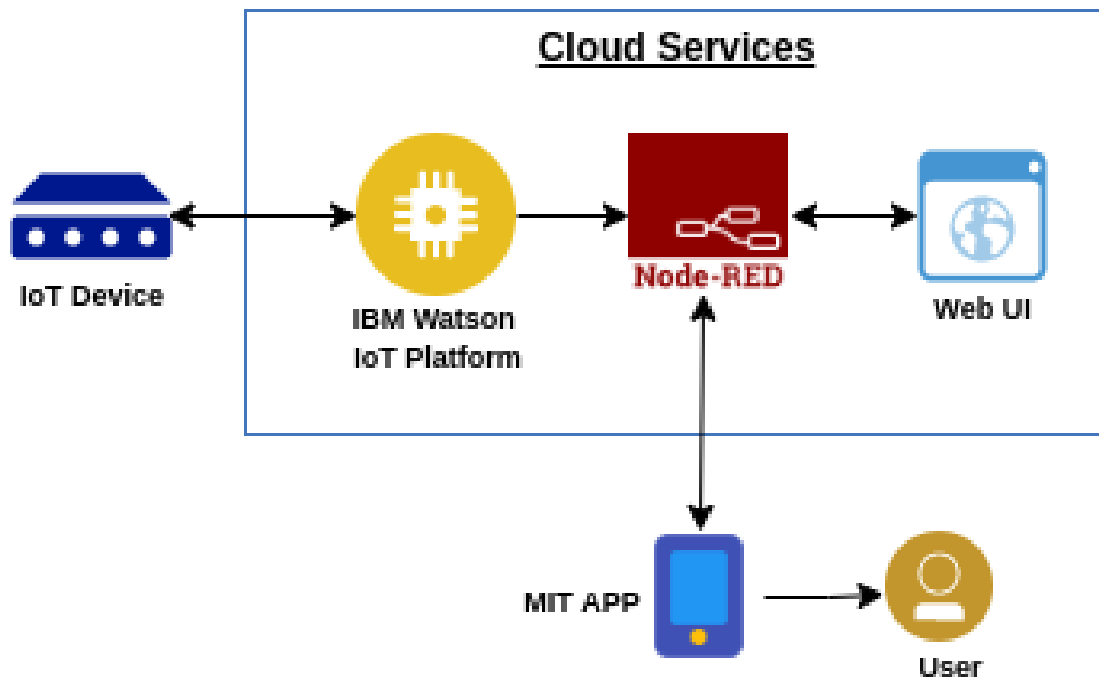
Farmers are to be present at farm for its maintenance irrespective of the weather conditions. They must ensure that the crops are well watered and the farm status is monitored by them physically. Farmer must stay most of the time in field in order to get a good yield. In difficult times like in the presence of pandemic also they must work hard in their fields risking their lives to provide food for the country.

### **2.2 PROPOSED SOLUTION:**

In order to improve the farmer's working conditions and make them easier, we introduce IoT services, in which we use cloud services and internet to enable farmer to continue his work remotely via internet. He can monitor the field parameters and control the devices in farm.

Based on these details he can water the crops by controlling the monitors through the app and the app gives an alert message if the temperature or humidity goes beyond a threshold. So that the farmer find ease to use the IoT based device in farming

### 3 BLOCK DIAGRAM:

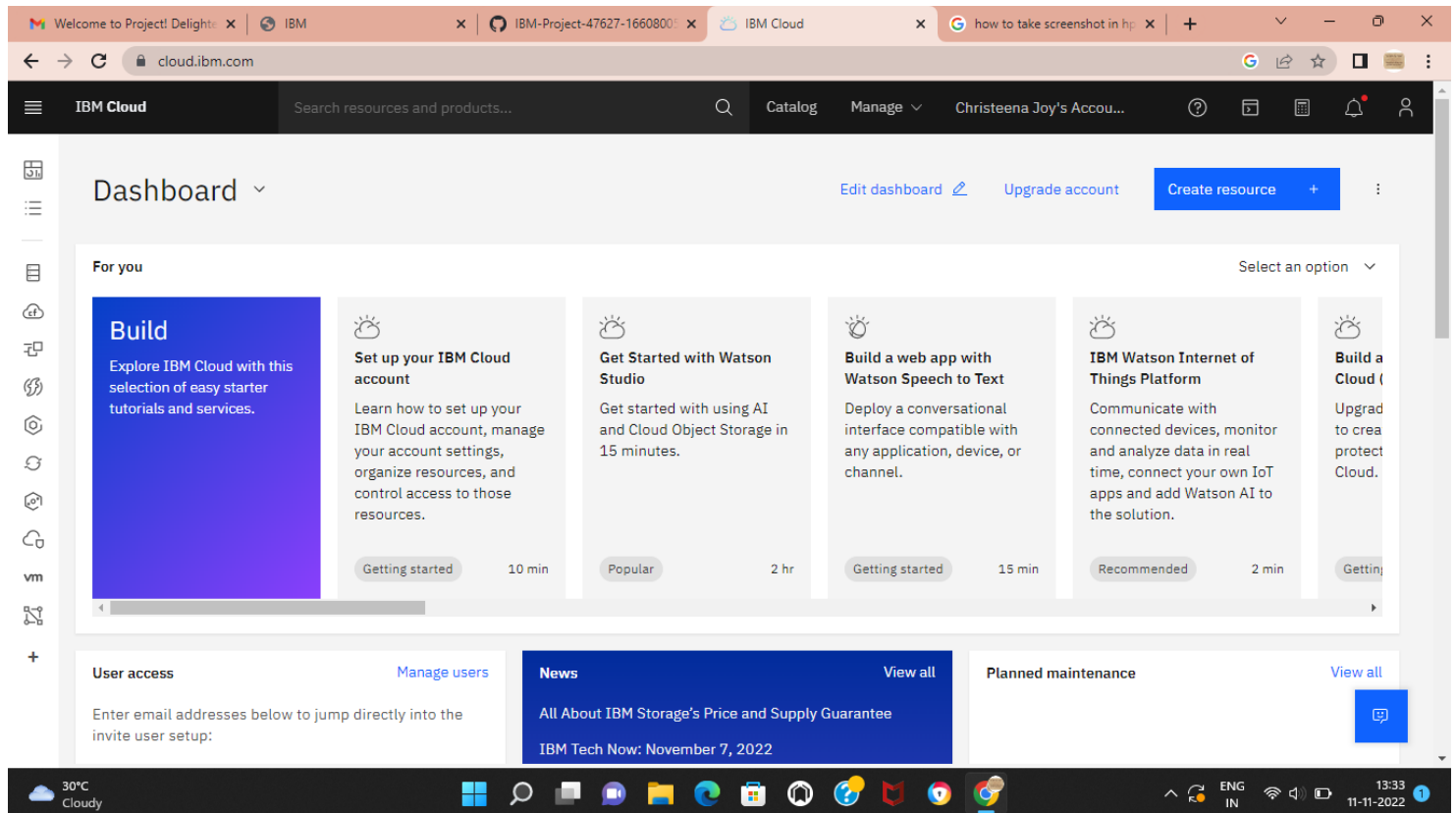


### 4 CREATED DEVICE IN IBM CLOUD

#### 4.1 STEPS TO CREATE DEVICE IN IBM CLOUD:

1. Log in to the console.
2. Navigate to the catalog, by clicking Catalog in the navigation bar. ...
3. Look for the Object Storage tile in the storage section and select it. ...

4. Give the service instance a name and choose a plan.
5. Click Create and you're automatically redirected to your new instance.



The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Node-RED: 169.51.205.205', 'IBM Watson IoT Platform', and 'sketch.ino copy - Wokwi Arduino'. The main header displays the user's email '963519104001abi@gmail.com' and ID 'hde0t6'. The dashboard has a sidebar with icons for various functions and a main content area with tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The 'Browse' tab is active, showing a table of devices. A search bar and a 'Device Simulator' toggle are also present.

Device ID	Status	Device Type	Class ID	Date Added
12345	Disconnected	abcd	Device	Nov 17, 2022 7:42 PM
98765	Connected	galaxy	Device	Nov 17, 2022 9:40 PM
abcd_1	Disconnected	abcd	Device	Nov 17, 2022 9:30 PM

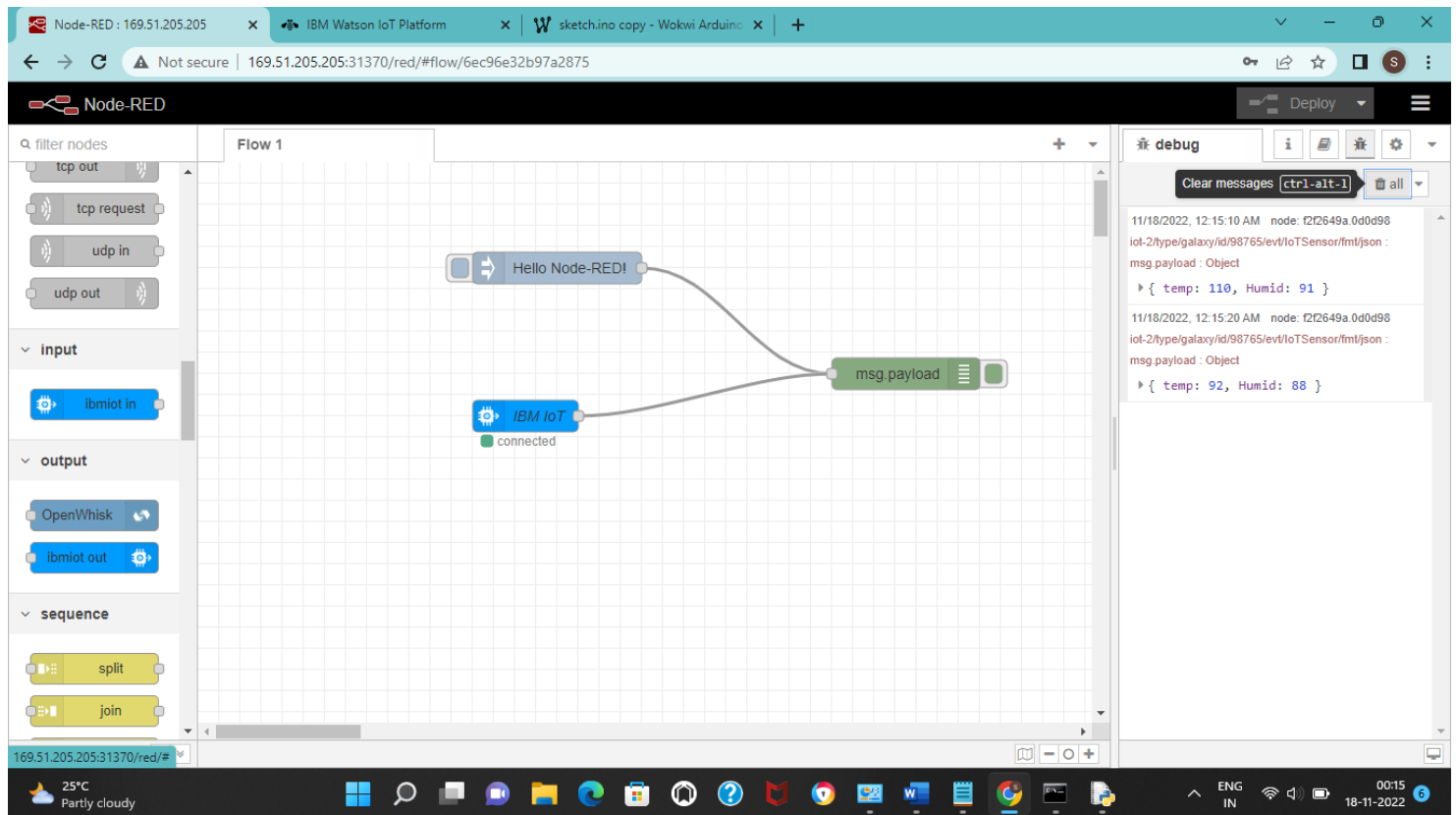
Items per page 50 | 1-3 of 3 items

1 of 1 page

## 5 NODE RED SOFTWARE:

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.



## 6 IBM WATSON IOT PLATFORM:

Watson IoT platform features Analytics and Watson APLs completely manage your IoT landscape and make better business decisions. Using a secure, Smart, and scalable platform as the hub of your IoT, get real-time analysis of user, machine and system-generate data, including speech, text video and social sentiments.

## 7 PYTHON IDLE

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.



## 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 "hde0t6"//IBM ORGANITION ID
#define DEVICE_TYPE "galaxy"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "98765"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "987654321" //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()// configuring 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 ");

```

```

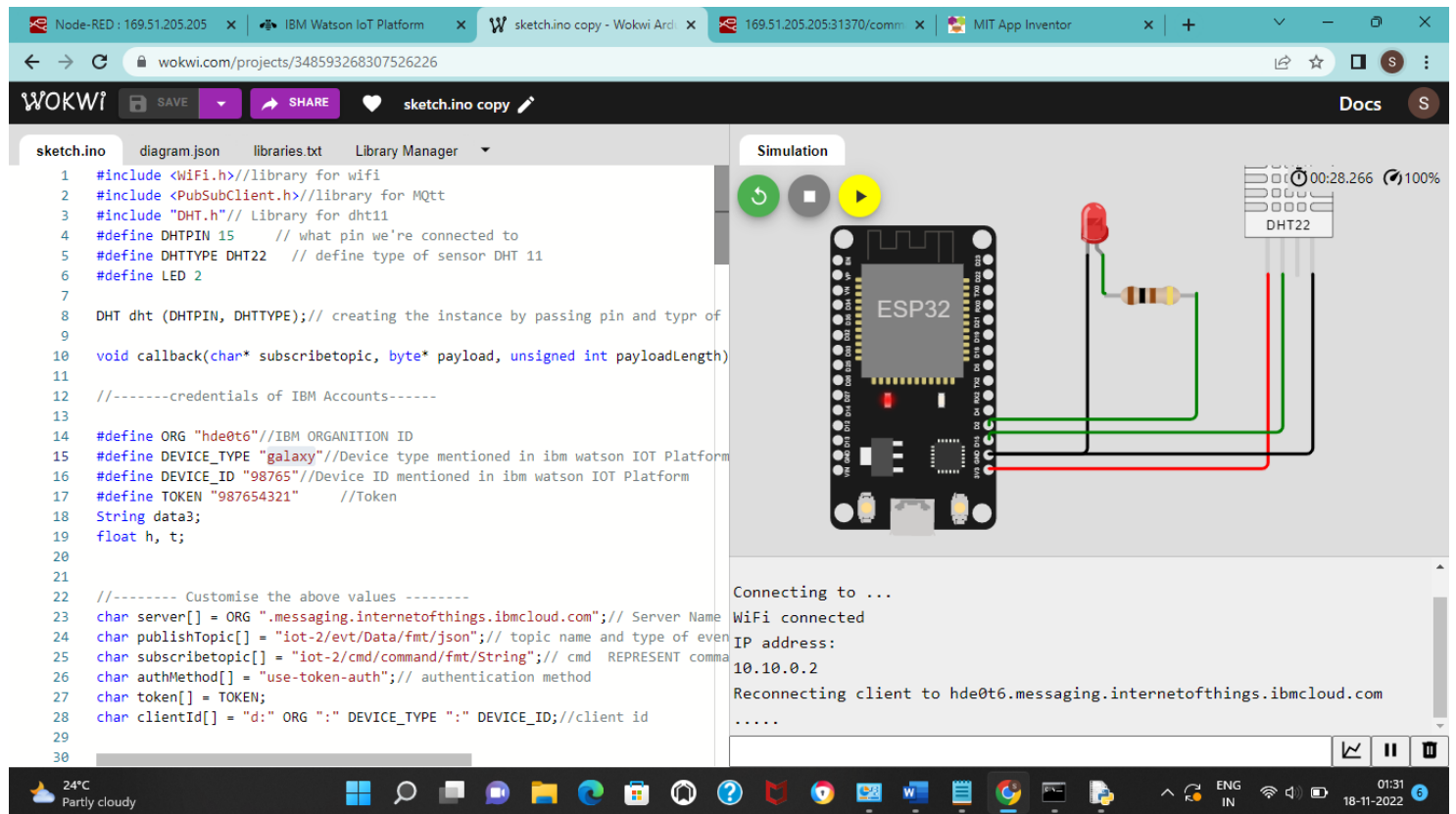
WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

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="";
}
```



## 8 IoT SIMULATOR

In our project for simulation we have used Watson iot platform for simulation. Watson IoT Platform features Analytics and Watson APIs Completely manage your IoT landscape and make better business decisions. Using a secure, smart and scalable platform as the hub of your IoT, get real-time analysis of user, machine and system-generated data, including speech, text video and social sentiment.

. The link for simulator- <https://www.ibm.com/cloud/watson-iot-platform/details>

## 9 OPEN WEATHER API

Open Weather platform is a set of elegant and widely recognizable APIs. Powered by convolutional machine learning solutions, it is capable of delivering all the weather information necessary for decision-making for any location on the globe.

Website link: <https://openweathermap.org/appid>

## 10 BUILDING PROJECTS:

Give the credentials of your device in IBM Watson IoT Platform

My credentials given to simulator are:

**OrgID:** hde0t6

**API:** a-hde0t6-nsjer2cc2z

**Device type:** galaxy

**token:** 6t2\_dpJeykNPZ-9NNW

**Device ID :** 98765

**Device Token :** 987654321

Node-RED : node-red-hdyfv-202 x +

ed/#flow/c7ddb1462b8a000c

### Edit ibmiot in node

Delete Cancel Done

#### Properties

⬆ Authentication API Key

🔍 API Key IBMIOT APIKEY

⚙ Input Type Device Event

📡 Device Type ☐ All or abcd

📶 Device Id ☐ All or 7654321

📋 Event ☒ All or +

📄 Format ☐ All or json

⊕ QoS 0

👤 Name IBM IoT

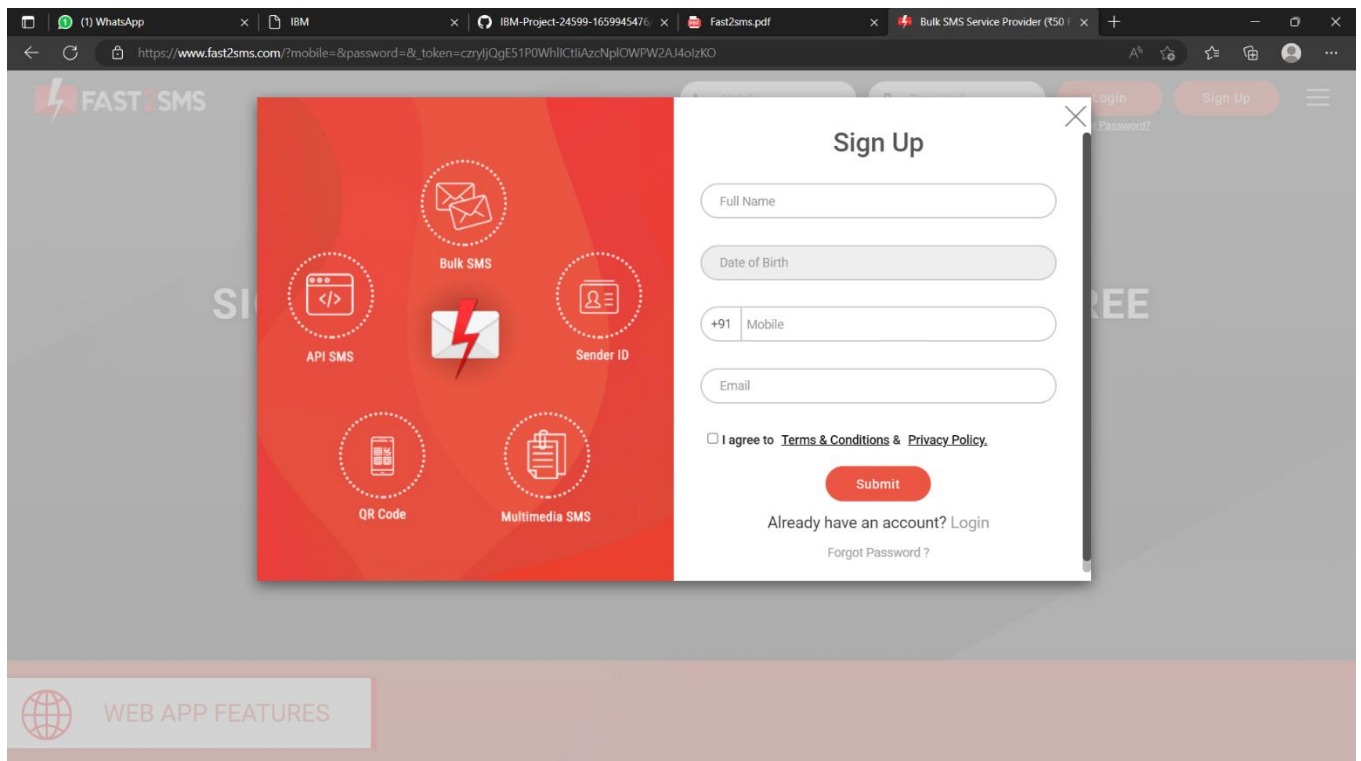
☐ Enabled

## 11 NODE RED TO SEND COMANDS TO IBM

### 11.1 CONFIGURATION OF NODE-RED TO SEND COMMANDS TO IBM:

ibmiot out node I used to send data from Node-Red to IBM Watson device. So, after adding it to the flow we need to configure it with credentials of our Watson device.

Here we add two buttons in UI





1 -> for motor on

2 -> for motor off

We used a function node to analyse the data received and assign command to each number.

The Java script code for the analyses is:

```
if(msg.payload===1)
```

```
msg.payload={"command": "ON"};
```

```
else if(msg.payload===0)
```

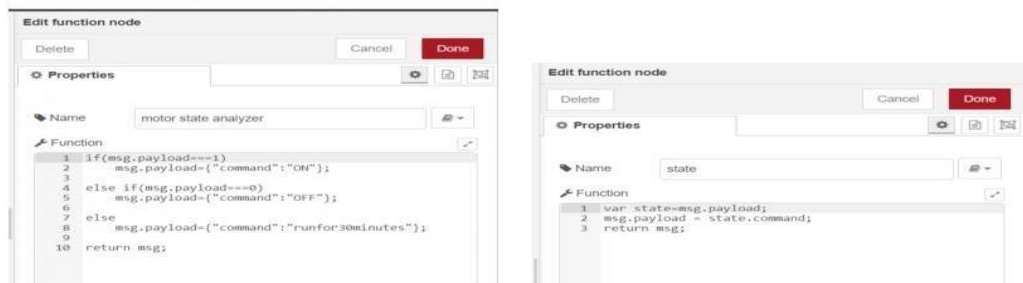
```
msg.payload={"command":
```

```
"OFF"};
```

Then we use another function node to parse the data and get the command and represent it visually with text node.

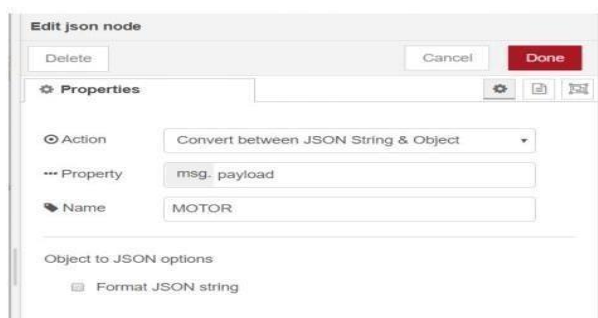
The Java script code for that function node is:

```
var state=msg.payload;  
msg.payload = state.command;  
return msg;
```

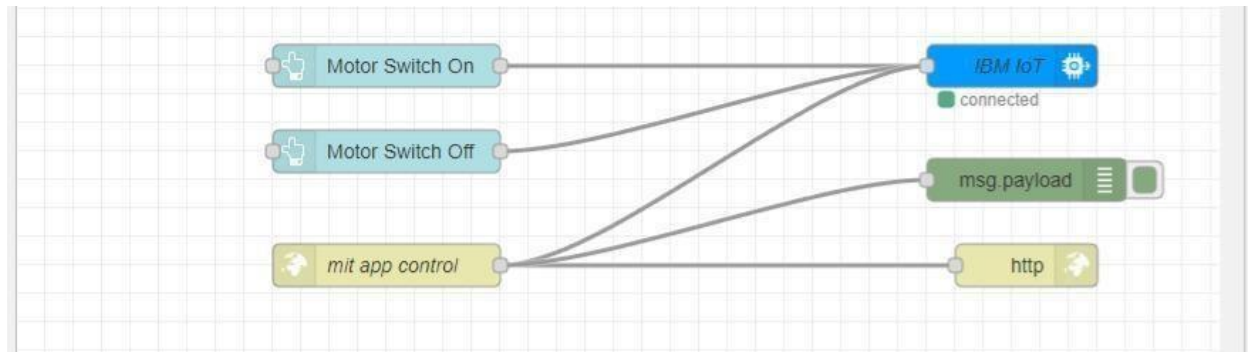


The above images show the java script codes of analyser and state function nodes.

Then we add edit Json node to the conversion between JSON string & object and finally connect it to IBM IoT Out.



Edit JSON node needs to be configured like this



This is the program flow for sending commands to IBM cloud.

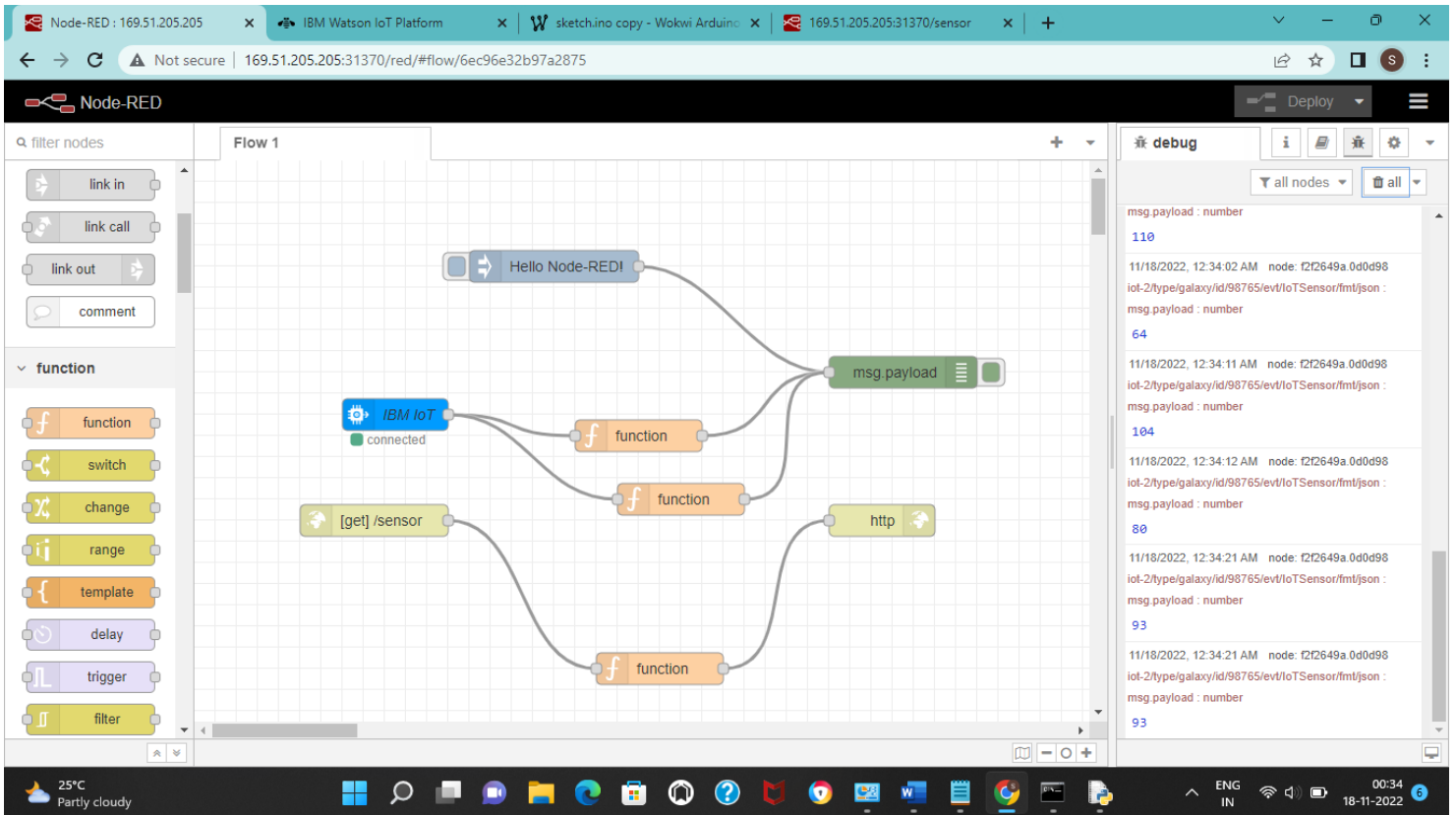
### 11.2 ADJUSTING USER INTERFACE:

In order to display the parsed JSON data a Node-Red dashboard is created

Here we are using Gauges, text and button nodes to display in the UI and helps to monitor the parameters and control the farm equipment.

Below images are the Gauge, text and button node configurations

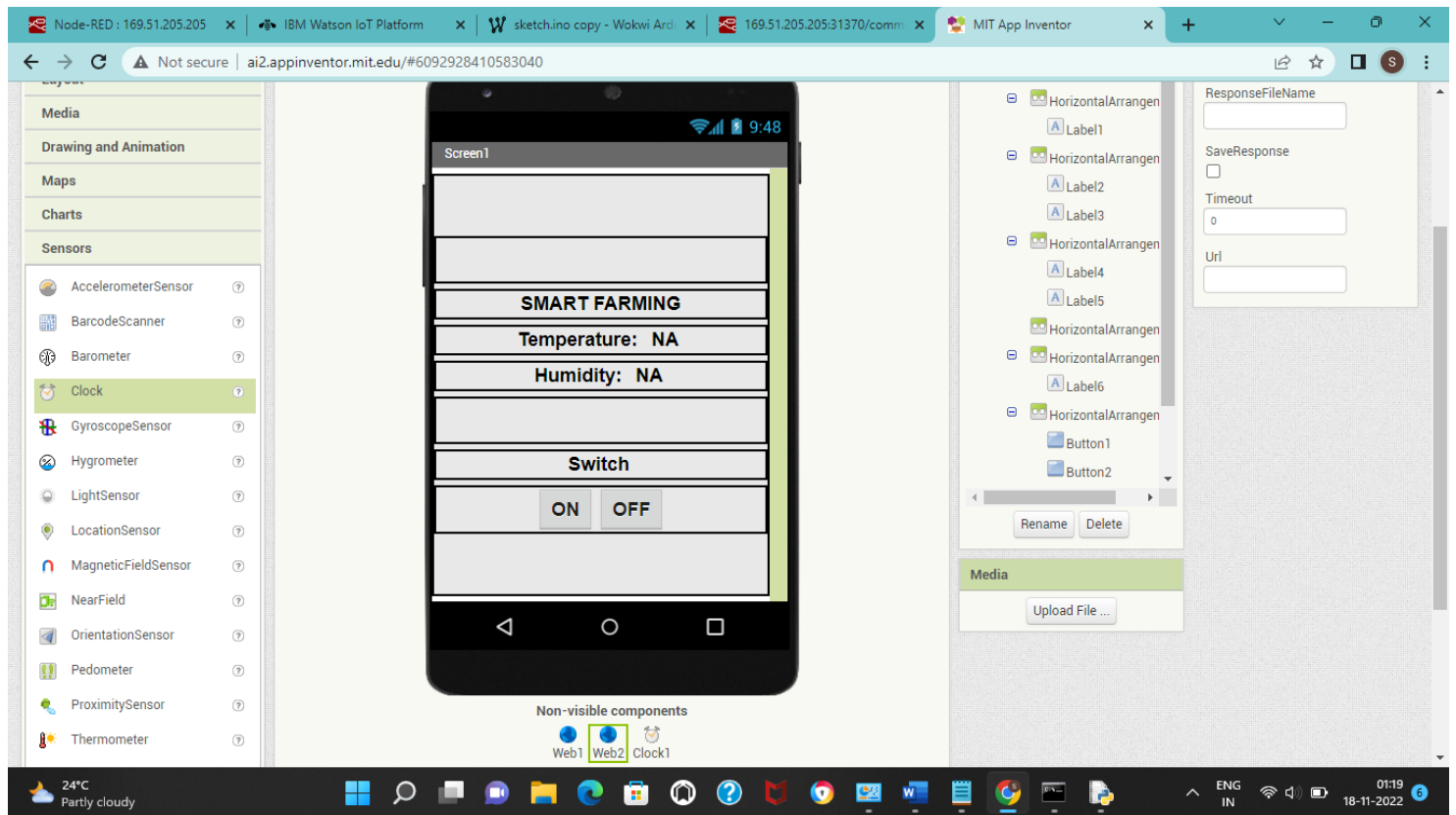
### 11.3 CREATE PROGRAM:



## 12 MOBILE APP WEB CREATION:

MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes.

### 12.1 MOBILE APP SCREENSHOT:

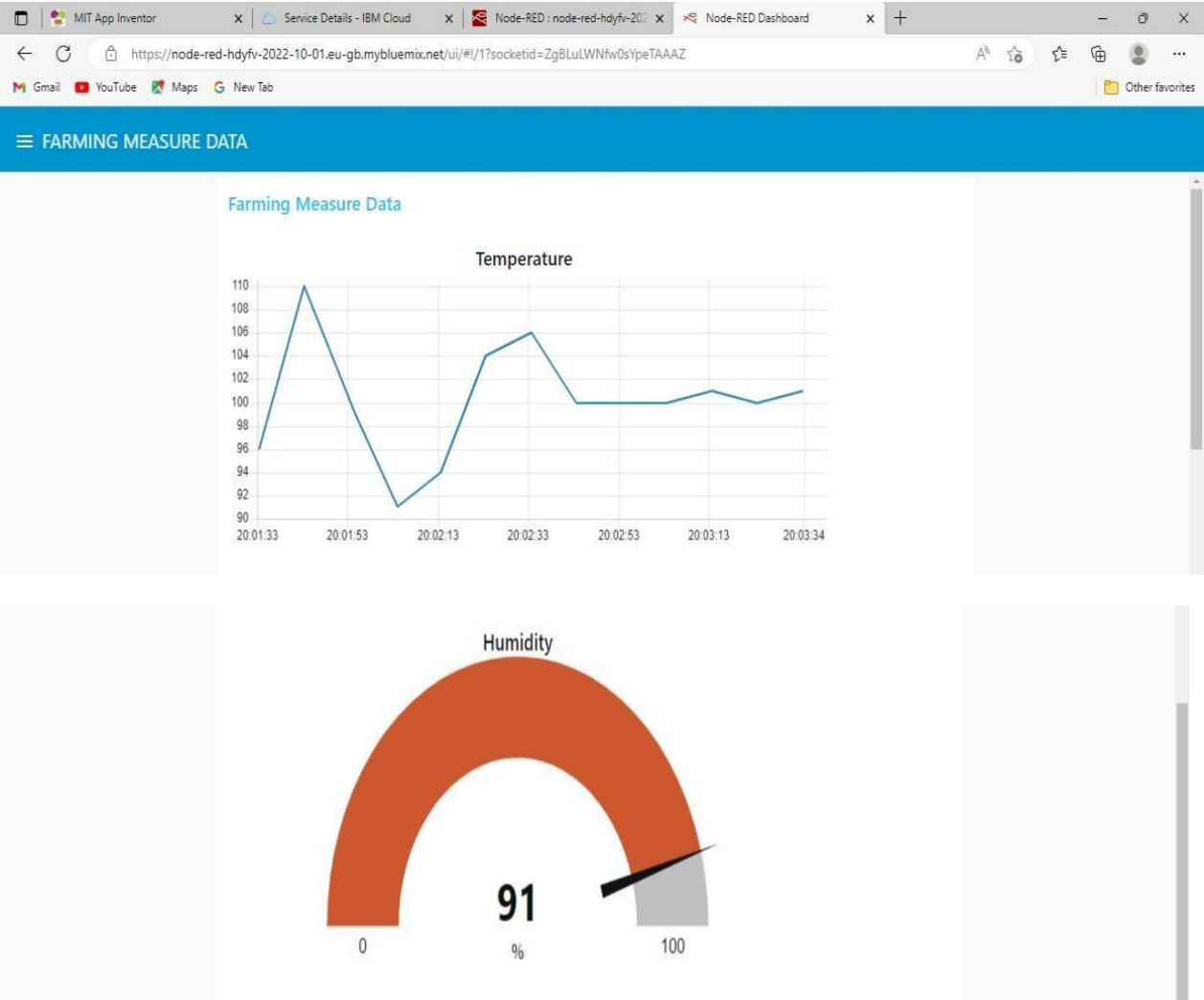


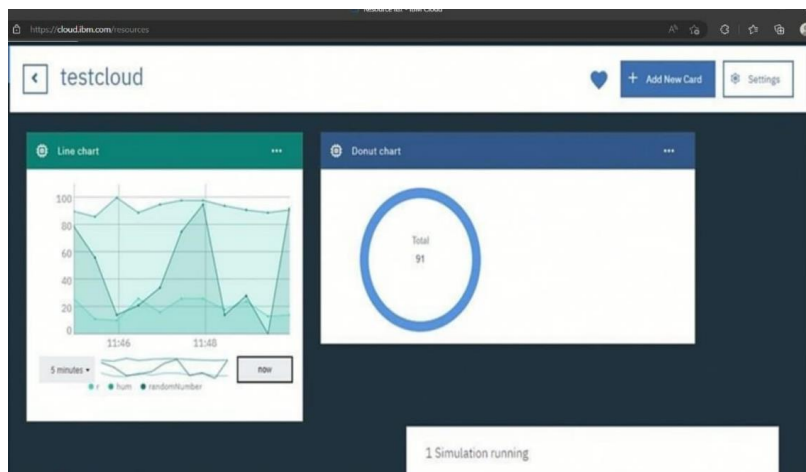
## 12.2 BACKEND OF MOBILE APPLICATION





### 12.3 WEB APP UI HOME TAB:





You can see the received data in graphs by creating cards in Boards tab You will receive the simulator data in cloud You can see the received data in Recent Events under your device .

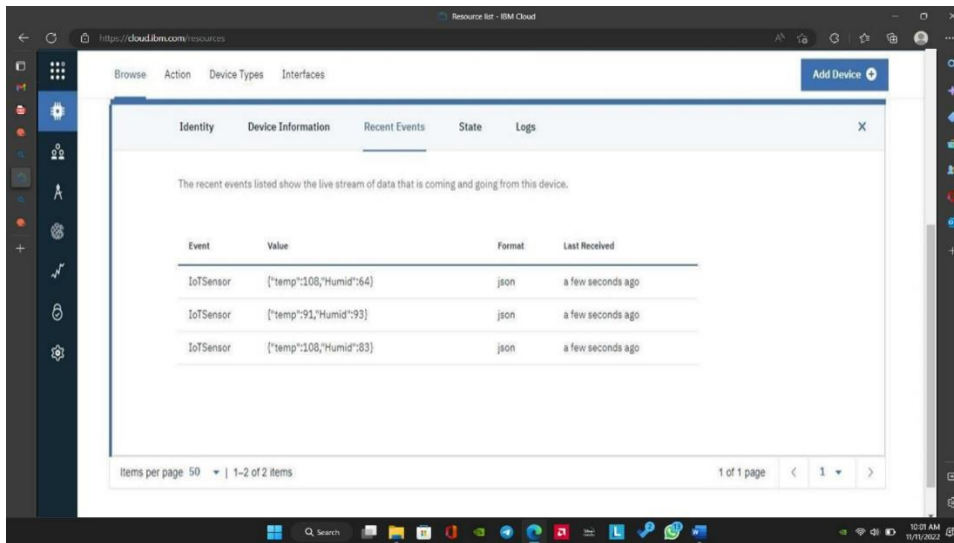
```
{
  "d": {
    "name": "wxyz",
    "temperature": 25,
    "humidity": 82,
```



▪ "Moisture ": 15

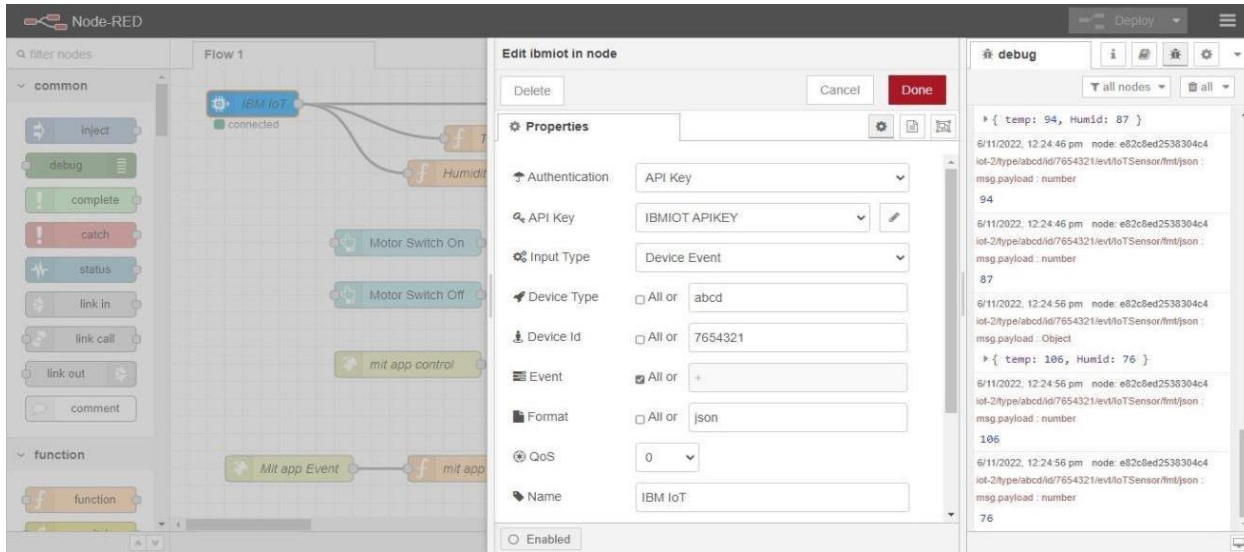
}

}



## 12.4 COLLECTING IBM CLOUD DATA BY NODE RED:

Below steps are used to collect the data from IBM cloud to NODE-Red platform



## 12.5 COLLECTING DATA FROM OPEN WEATHER API:

The node-red collects data from OpenWeather API

```
format:{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clo
uds","
description":"brokenclouds","icon":"04n"}],"base":"stations","main":{"temp
":307
59,"feels_like":305.5,"temp_min":307.59,"temp_max":307.59,"pressure":10
02,"h
umidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"de
g":170}
,"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589
933553,
"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":20
0}
```

```
var temperature = msg.payload.main.temp;
```

```
temperature = temperature-273.15;
```

```
return {payload : temperature.toFixed(2)};
```

in this code, the temperature parameter is initially in kelvin and converted to Celsius

## **12.5 Receiving commands from IBM cloud**

```
import time
```

```
import sys
```

```
import ibmiotf.application
```

```
import ibmiotf.device
```

```
import random
```

```
#Provide your IBM Watson Device Credentials
```

```
organization = "hde0t6"
```

```
deviceType = "galaxy"
```

```
deviceId = "98765"
```

```
authMethod = "token"
```

```
authToken = "987654321"
```

```
# Initialize GPIO
```

```
def myCommandCallback(cmd):
```

```

print("Command received: %s" % cmd.data['command'])

status=cmd.data['command']

if status=="lighton":

    print ("led is on")

elif status == "lightoff":

    print ("led 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
```

```
    temp=random.randint(90,110)
```

```
    Humid=random.randint(60,100)
```

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

```
    #print data
```

```
    def myOnPublishCallback():
```

```
        print ("Published Temperature = %s C" % 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(10)
```

```
deviceCli.commandCallback = myCommandCallback
```

OUT

```
deviceCli.disconnect()
```

**OUTPUT:**



```
milky.py - C:\Users\01abi\Desktop\milky.py (3.7.0)
File Edit Format Run Options Window Help

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

#Provide your IBM Watson Device Credentials
organization = "hde0t6"
deviceType = "galaxy"
deviceId = "98765"
authMethod = "token"
authToken = "987654321"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
        print ("led 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

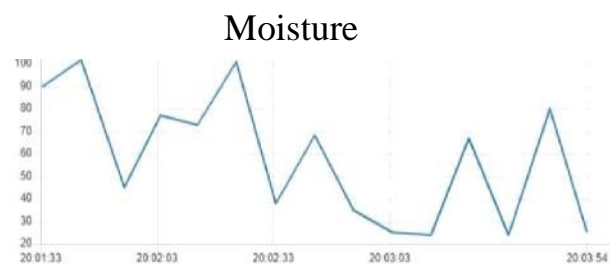
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
```

## 13 OBSERVATION AND RESULT

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (tags/v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\01abi\Desktop\milky.py =====
2022-11-18 20:19:57,570 ibmiotf.device.Client INFO Connected successfully: d:hde0t6:galaxy:98765
Published Temperature = 92 C Humidity = 96 % to IBM Watson
Published Temperature = 101 C Humidity = 62 % to IBM Watson
Published Temperature = 96 C Humidity = 66 % to IBM Watson
Published Temperature = 107 C Humidity = 68 % to IBM Watson
|
```

### Farmin5 Measure Data





Switchboard

MOTOR SWITCH ON

MOTOR SWITCH OFF



## **14 ADVANTAGES**

- Farms can be monitored and controlled remotely.
- Increase in convenience to farmers.
- Less labor cost.
- Better standards of

living.

## **15 CONCLUSION**

Thus the objective of the project to implement an IoT system in order to help farmers to control and monitor their farms has been implemented successfully.

## **16 REFERENCE**

IBM cloud reference: <https://cloud.ibm.com/>

IoT simulator : <https://watson-iot-sensor-simulator.mybluemix.net/> OpenWeather :

<https://openweathermap.org/>

## **SOURCE CODE**

```
import time
```

```
import sys
```

```
import ibmiotf.application
```

```
import ibmiotf.device
```

```
import random
```

**#Provide your IBM Watson Device Credentials**

**organization = "hde0t6"**

**deviceType = "galaxy"**

**deviceId = "98765"**

**authMethod = "token"**

**authToken = "987654321"**

**# Initialize GPIO**

**def myCommandCallback(cmd):**

**print("Command received: %s" % cmd.data['command'])**

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

**if status=="lighton":**

**print ("led is on")**

**elif status == "lightoff":**

**print ("led 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
```

```
    temp=random.randint(90,110)
```

```
    Humid=random.randint(60,100)
```

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

```
    #print data
```

```

def myOnPublishCallback():

    print ('Published Temperature = %s C' % 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 IoTTF")

        time.sleep(10)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

```

## **GITHUB LINK**

IBM-EPBL/IBM-Project-46302-1660744646

## **PROJECT DEMO LINK**

[https://www.mediafire.com/file/4z00sgw15edggel/VID-20221115-WA0023\\_%25282%2529.mp4/file](https://www.mediafire.com/file/4z00sgw15edggel/VID-20221115-WA0023_%25282%2529.mp4/file)