IOT ENABLED SMART FARMING APPLICATION.

PROJECT REPORT

TEAMID : PNT2022TMID37007

➢ Hemachandran G ➢ Murugan G ➢ Rithish Abinav ➢ SivaPriya ➢ Kaviya

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

2. Problem Statement

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. Farmer 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.

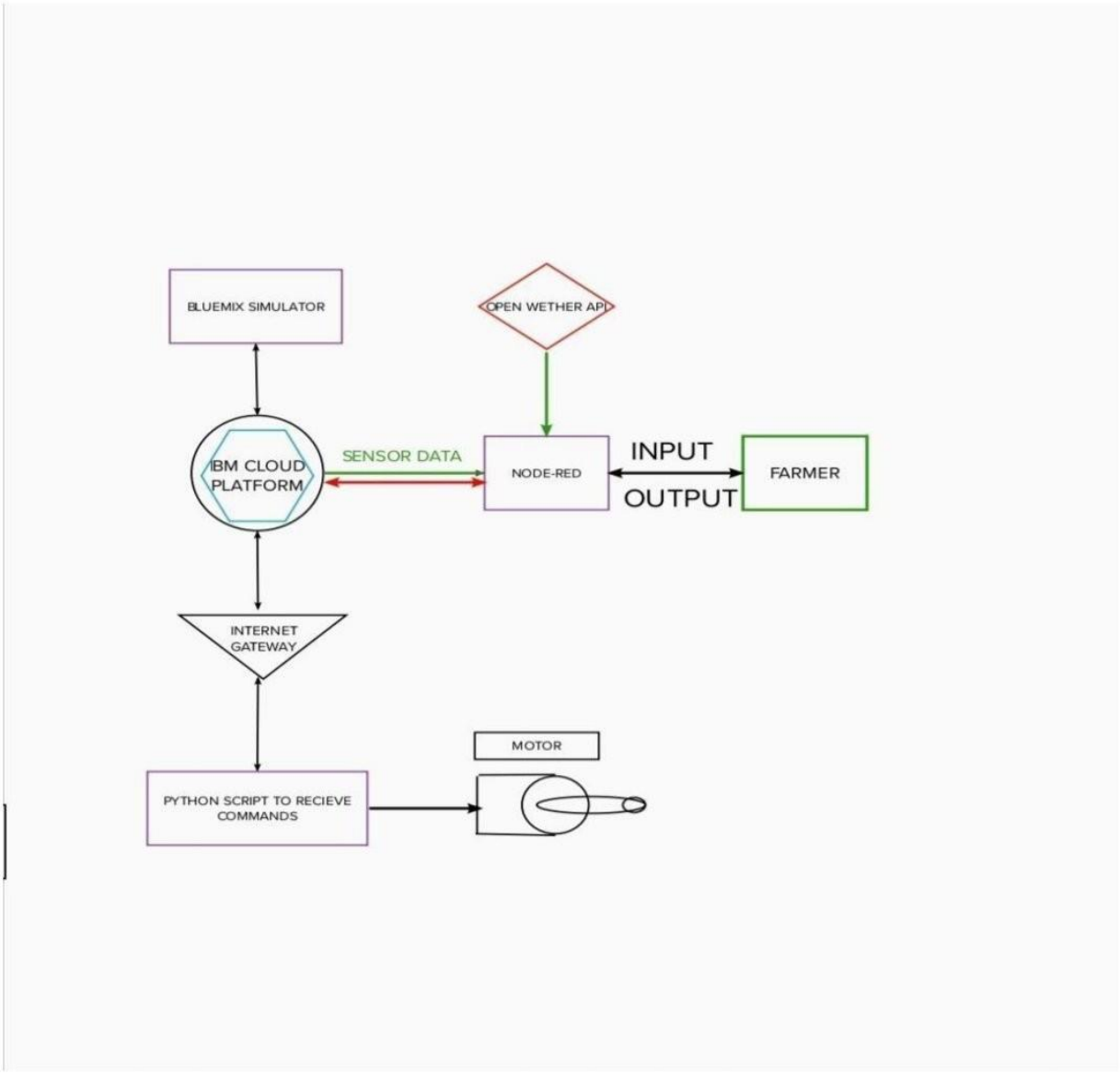
3. Proposed Solution

In order to improve the farmer's working conditions and make them easier, we introduce IoT services to him 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.

4. Theoretical Analysis

4.1 Block Diagram

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

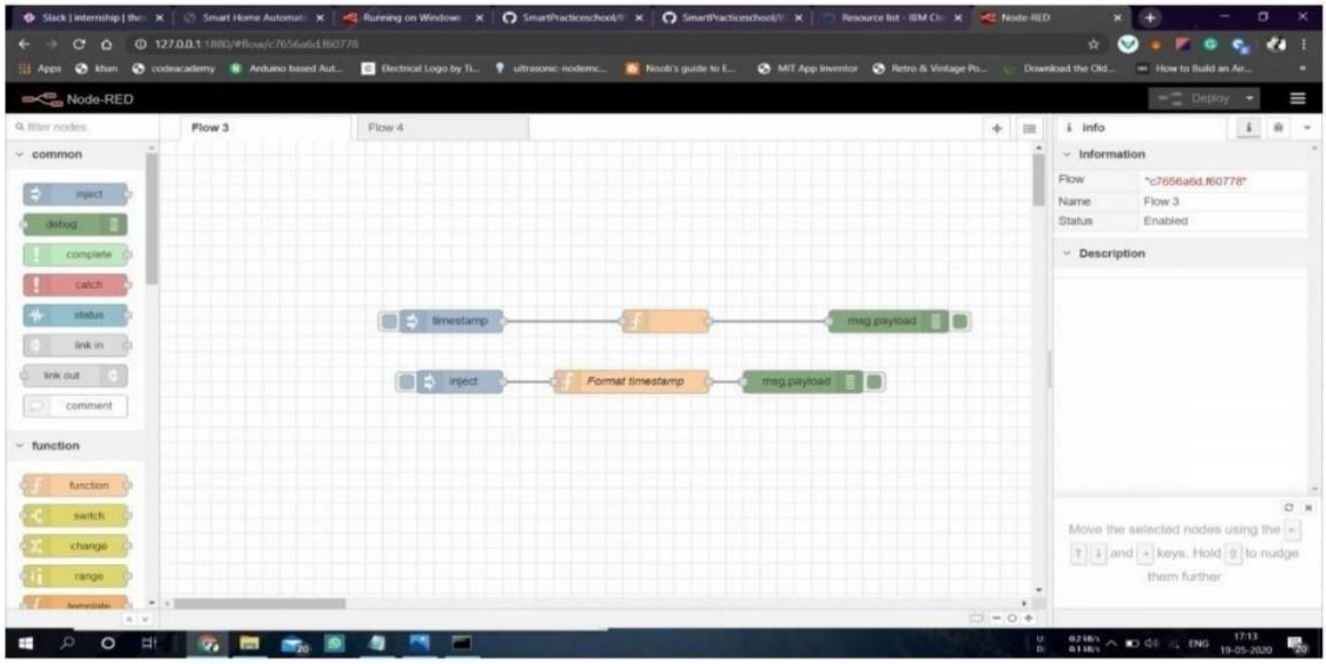


4.2 Required Software Installation

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://localhost:1880/ in browser

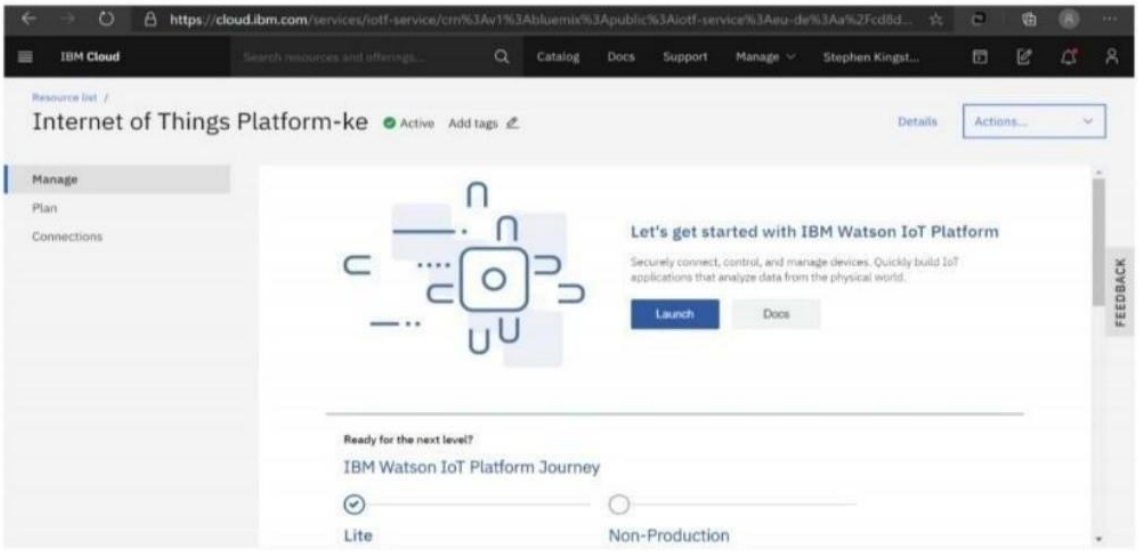
Installation of IBM IoT and Dashboard nodes for Node-Red

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

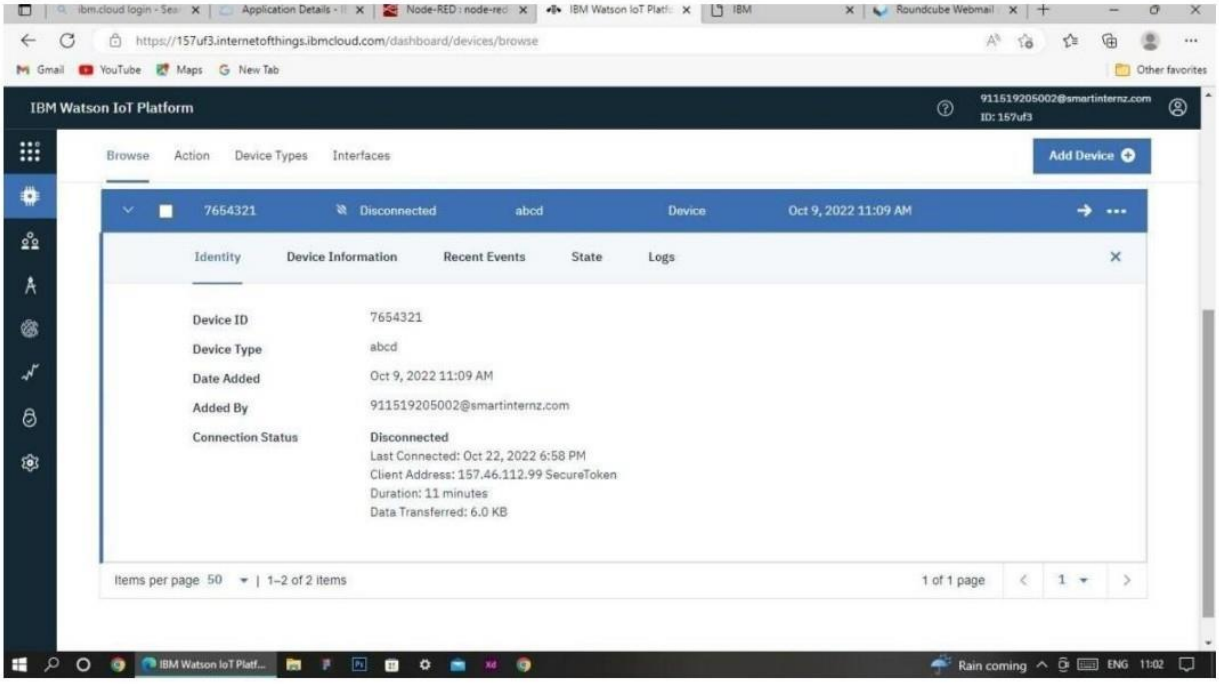
4.2.B IBM Watson IoT Platform

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.



4.2.C Python IDE

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used Spyder to execute the code.

Code: import time

import sys import

ibmiotf.application import

ibmiotf.device import

random

#Provide your IBM Watson Device Credentials

organization = "cpeq2u"

deviceType = "TestDevice"

deviceId = "082001" authMethod = "token"

authToken = "8))+idxmB_q2PM@uvP"

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


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

```
Mois=random.randint(20,120)

    data = { 'temp' : temp, 'Humid': Humid, 'Mois' :Mois}
    #print data       def
myOnPublishCallback():
print ("Published Temperature
= %s C" % temp, "Humidity = %s
%%" % Humid, "Moisture =%s
deg c" %Mois, "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()
```

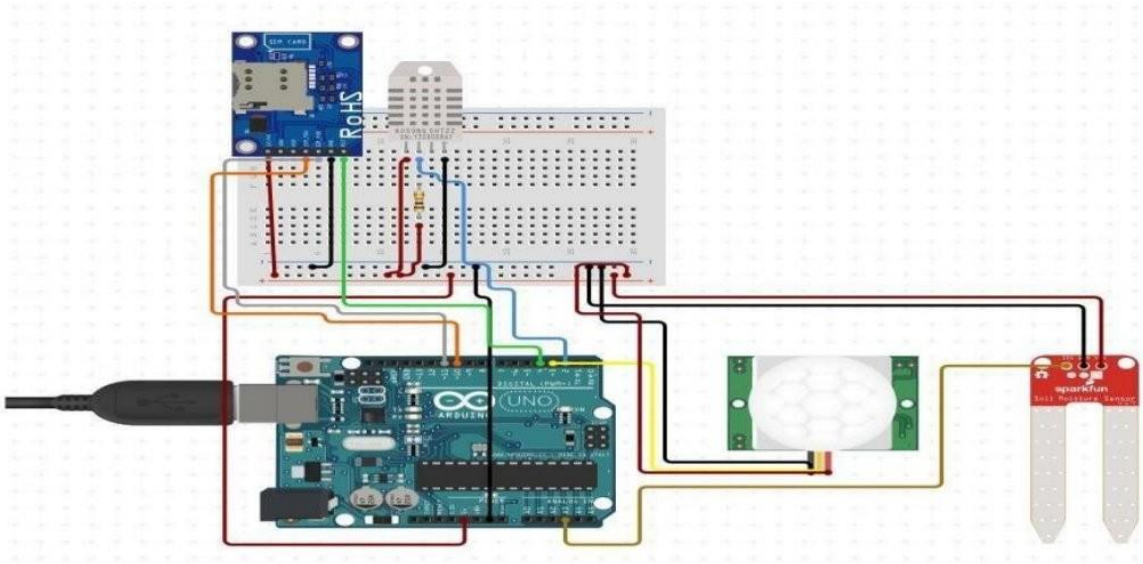
Aurdino code for C :

```
//include libraries
 #include <dht.h>
 #include <SoftwareSerial.h>
```

```
//define pins
#define dht_apin A0 // Analog Pin sensor is connected
SoftwareSerial mySerial(7,8);//serial port of gsm
const int sensor_pin = A1; // Soil moisture sensor O/P pin
int pin_out = 9;
//allocate variables
dht DHT;
int c=0;

void setup()
{
pinMode(2, INPUT); //Pin 2 as INPUT
pinMode(3, OUTPUT); //PIN 3 as OUTPUT
pinMode(9, OUTPUT);//output for pump
}
void loop()
{
 if (digitalRead(2) == HIGH)
 {
 digitalWrite(3, HIGH); // turn the LED/Buzz ON
 delay(10000); // wait for 100 msecond
 digitalWrite(3, LOW);  // turn the LED/Buzz OFF
 delay(100);
 }
  Serial.begin(9600);
    delay(1000);
   DHT.read11(dht_apin); //temprature
 float h=DHT.humidity;
 float t=DHT.temperature;
   delay(5000);
   Serial.begin(9600);
    float moisture_percentage;//moisture
 int sensor_analog;
 sensor_analog = analogRead(sensor_pin);
 moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
```

```
  float m=moisture_percentage;
  delay(1000);
  if(m<40)//pump
  {
  while(m<40)
  {
  digitalWrite(pin_out,HIGH);//open pump
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
  m=moisture_percentage;
  delay(1000);
  }
  digitalWrite(pin_out,LOW);//closepump
  }
  if(c>=0)
  {
  mySerial.begin(9600);
  delay(15000);
  Serial.begin(9600);
  delay(1000);
  Serial.print("\r");
  delay(1000);
  Serial.print("AT+CMGF=1\r");
  delay(1000);
  Serial.print("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X with 10 digit mobil
e number
  delay(1000);
  Serial.print((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moisture="+m);
  delay(1000);
  Serial.write(0x1A);
  delay(1000);
  mySerial.println("AT+CMGF=1");//Sets the GSM Module in Text Mode
  delay(1000);
```

```
  mySerial.println("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X with 10 digit
mobile number
  delay(1000);
  mySerial.println((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moisture="+m);//
message format
  mySerial.println();
  delay(100);
  Serial.write(0x1A);
  delay(1000);
  c++;

   }


}
```



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://watson-iot-sensor-simulator.mybluemix.net/

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

4.4 OpenWeather API

OpenWeatherMap is an online service that provides weather data. It provides current weather data, forecasts and historical data to more than 2 million customer.

Website link: https://openweathermap.org/guide **Steps**

to configure:

o Create account in OpenWeather o Find the name of

your city by searching o Create API key to your account

o Replace “city name” and “your api key” with your city
and API key in below red text

api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}

5,Building Project

5.1 Connecting IoT Simulator to IBM Watson IoT Platform

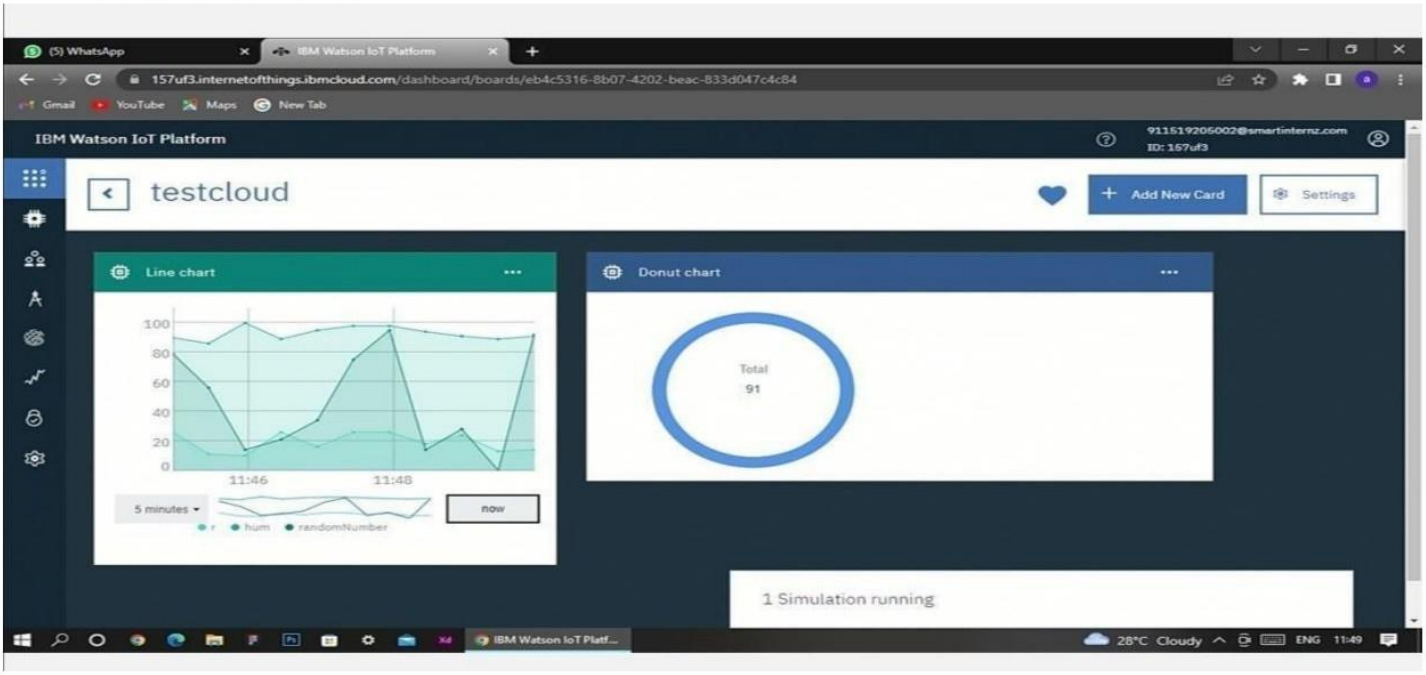
Open link provided in above section 4.3

Give the credentials of your device in IBM Watson IoT Platform

Click on connect

My credentials given to simulator are:
OrgID: **cpeq2u** api: **a-157uf3- f5rg4qxpd3** Device type: TestDevice
token:8))+idxmB_q2PM@uvP
Device ID : 802001

Device Token : **87654321**



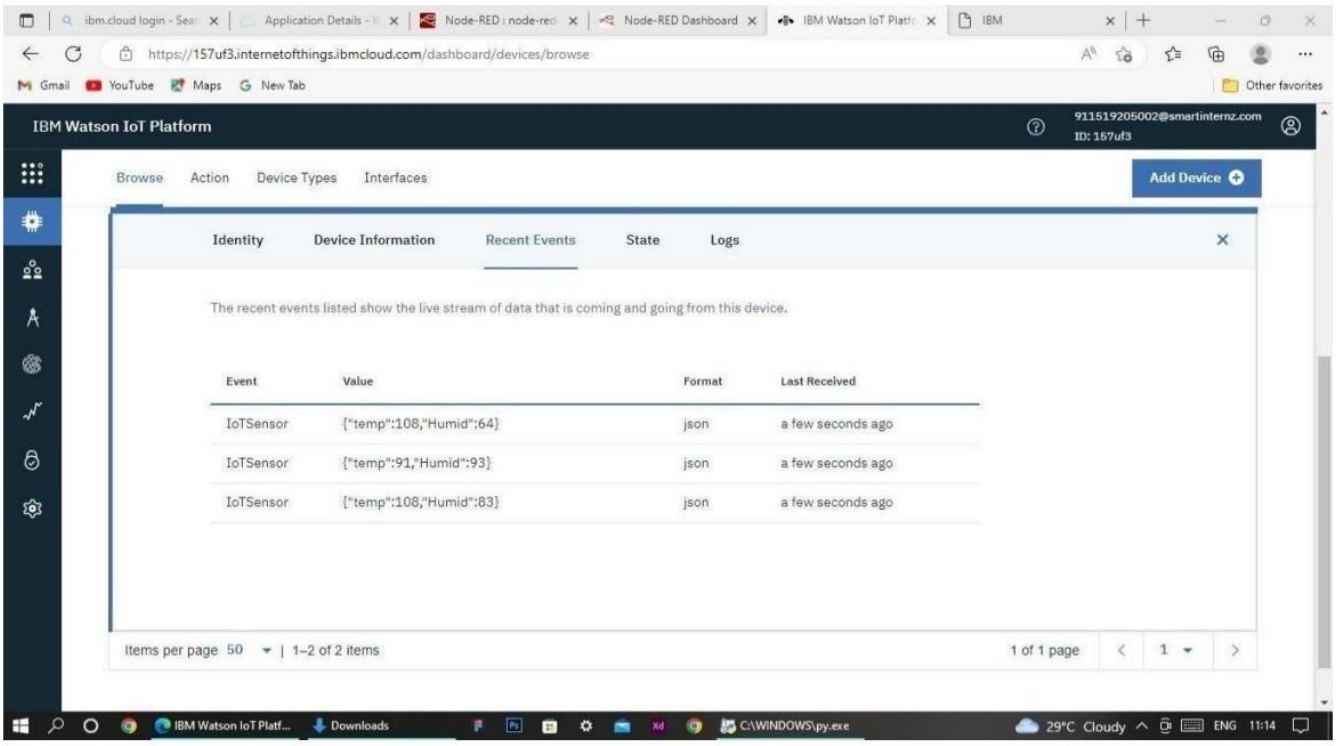
You can see the received data in graphs by creating cards in Boards tab

➢ You will receive the simulator data in cloud

➢ You can see the received data in Recent Events under your device

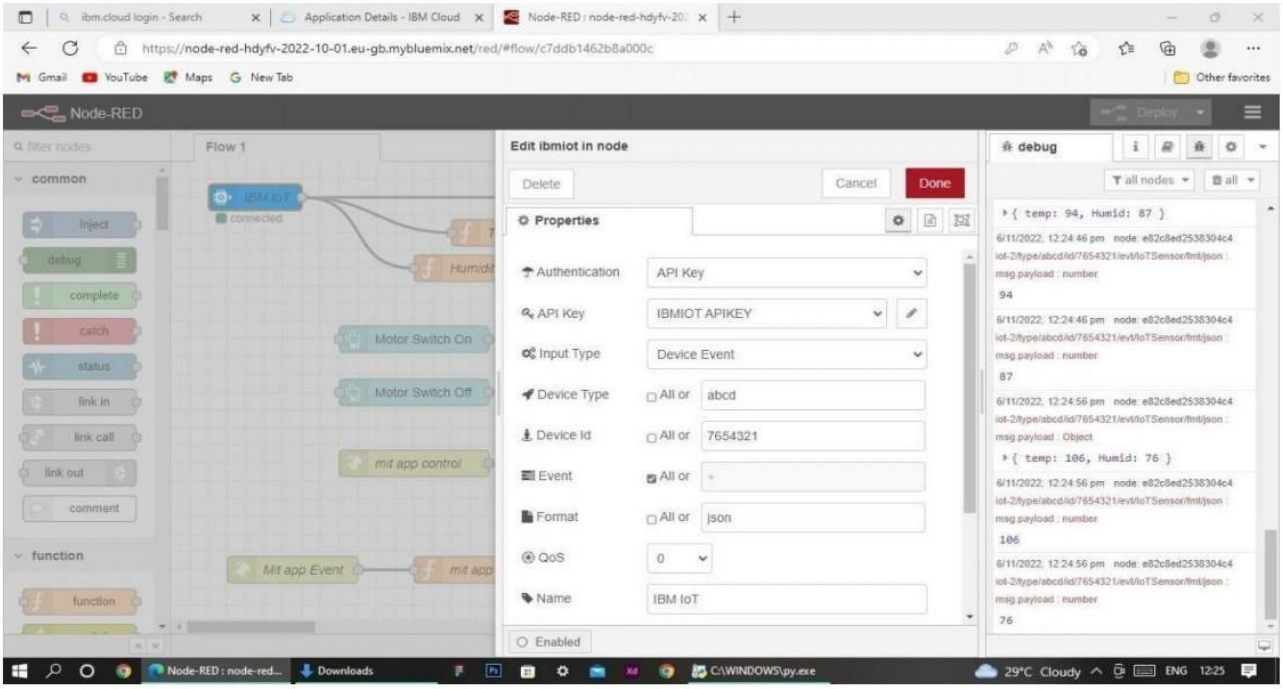
➢ Data received in this format(json)


```
{
  "d": {
▪ "name": "abcd",
▪ "temperature": 17,
▪ "humidity": 76,
▪ "Moisture ": 25
   }
}
```

5.2 Configuration of Node-Red to collect IBM cloud data

The node IBM IoT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.



Once it is connected Node-Red receives data from the device

Display the data using debug node for verification

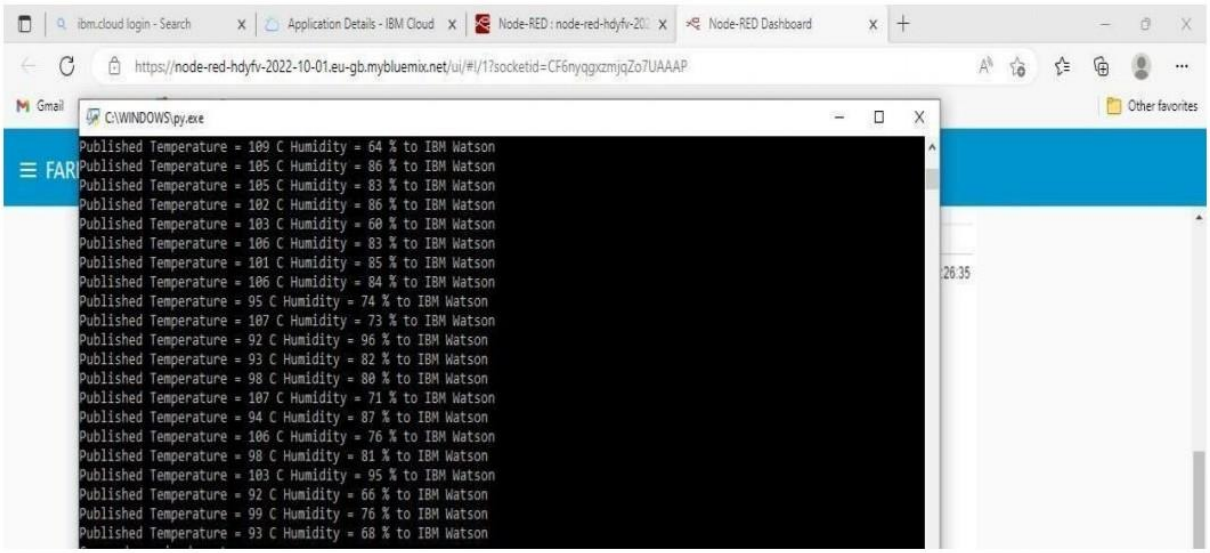
Connect function node and write the Java script code to get each reading separately.

The Java script code for the function node is:

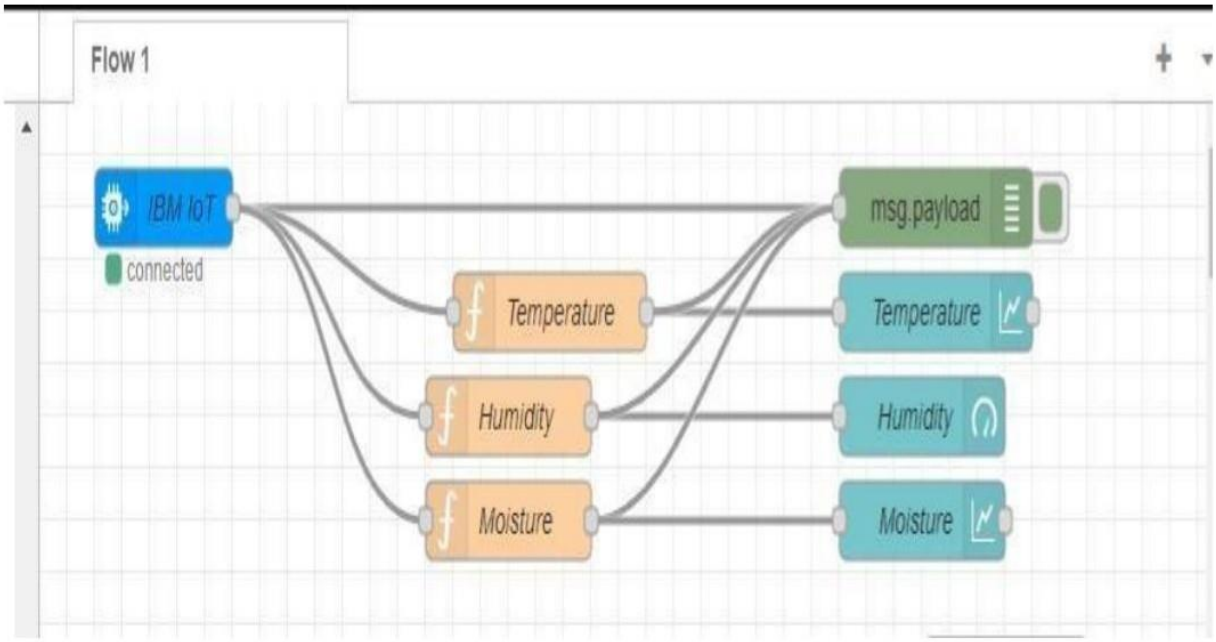
msg.payload=msg.payload.d.temperature return

msg;

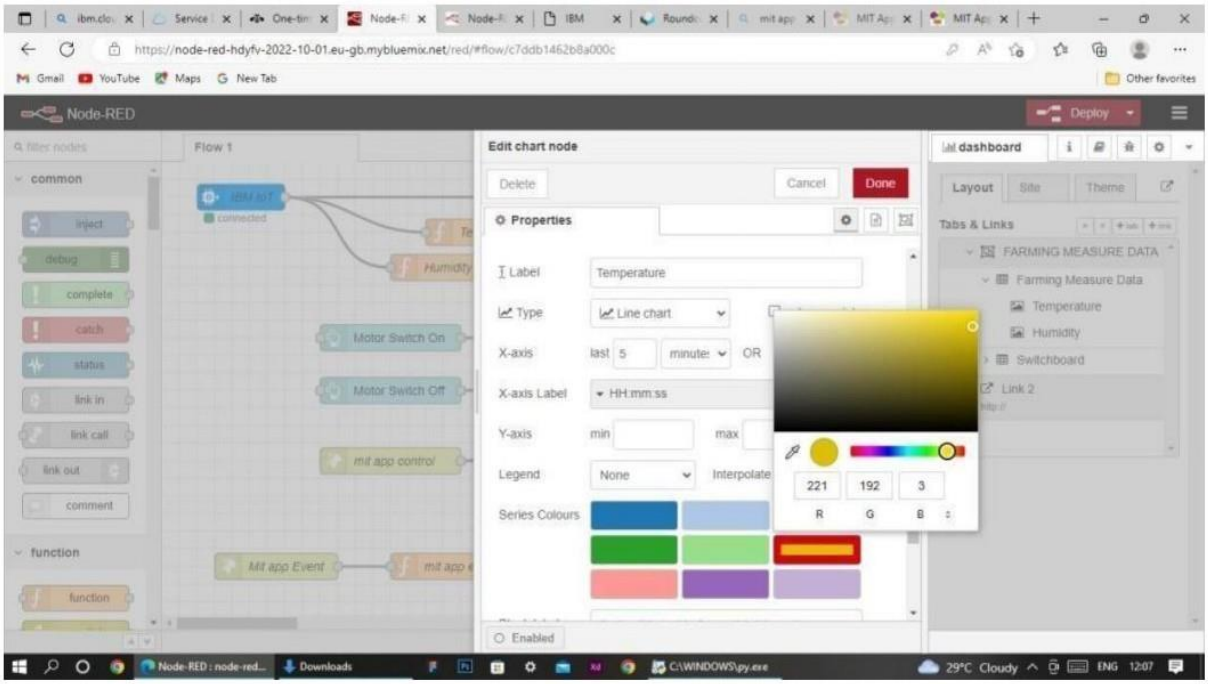
Finally connect Gauge nodes from dashboard to see the data in UI



Data received from the cloud in Node-Red console



Nodes connected in following manner to get each reading separately



This is the Java script code I written for the function node to get Temperature separately.

5.3 Configuration of Node-Red to collect data from OpenWeather

The Node-Red also receive data from the OpenWeather API by HTTP GET request. An inject trigger is added to perform HTTP request for every certain interval.

HTTP request node is configured with URL we saved before in section 4.4 The

data we receive from OpenWeather after request is in below JSON

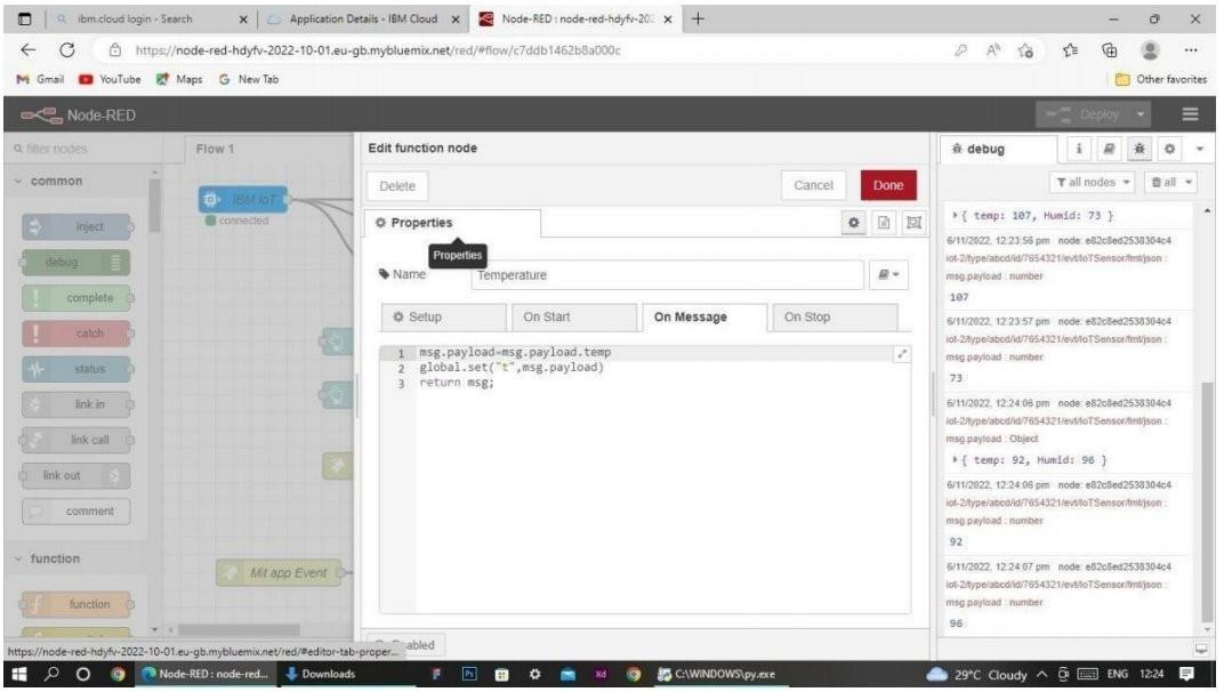
format:{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds","description":"brokenclouds","icon":"04n"}],"base":"stations","main":{"temp":30759,"feels_like":305.5,"temp_min":307.59,"temp_max":307.59,"pressure":1002,"humidity":35,"sea_level":1002,"grnd_level":1000},"wind":{"speed":6.23,"deg":170},"clouds":{"all":68},"dt":1589991979,"sys":{"country":"IN","sunrise":1589933553,"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":200}

In order to parse the JSON string we use Java script functions and get each parameters

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

temperature = temperature-273.15;

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

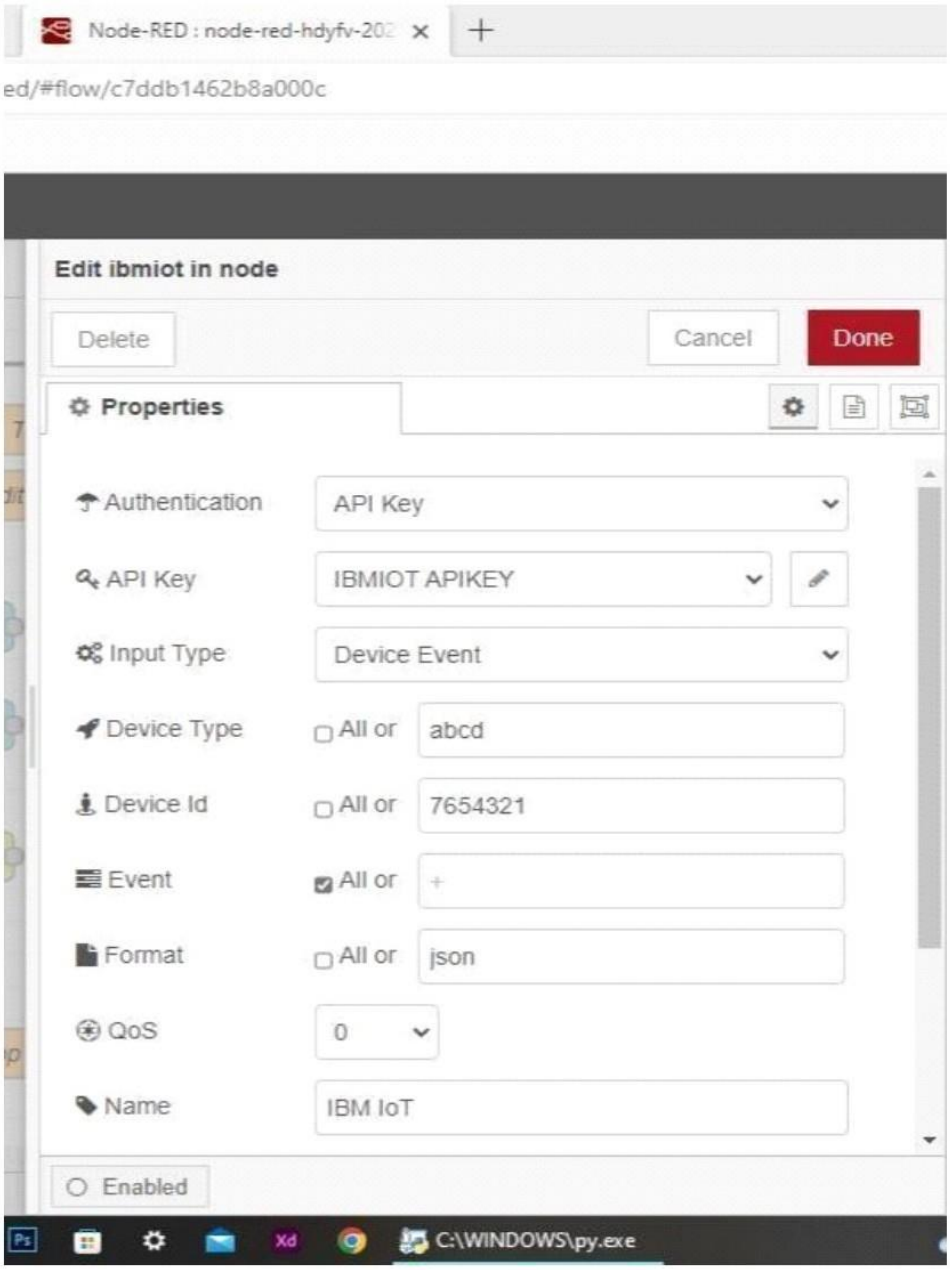
In the above Java script code we take temperature parameter into a new variable and convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI



- ## Configuration of Node-Red to send commands to IBM cloud

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 Watsondevice.



Here we add two buttons in UI

1 ->

for

mot

or

on2

->

for

mot

or

off

We used a function node to analyses the data received and assign command toeach number.

The Java script code for the

analyses is:if(msg.payload===1)

msg.payload={"command”: “ON"};

else

if(msg.payload===0)

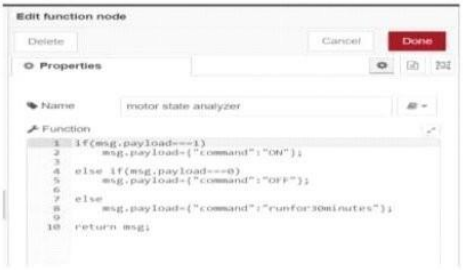
msg.payload={"comm

and”: “OFF"};

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

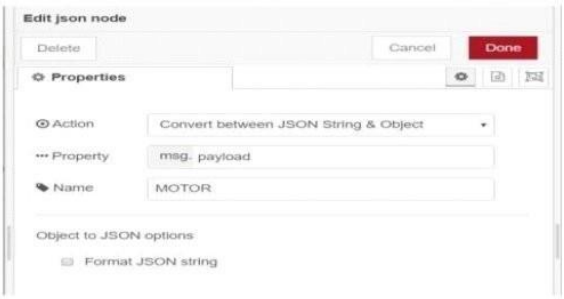
The Java script code for that function node is:

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

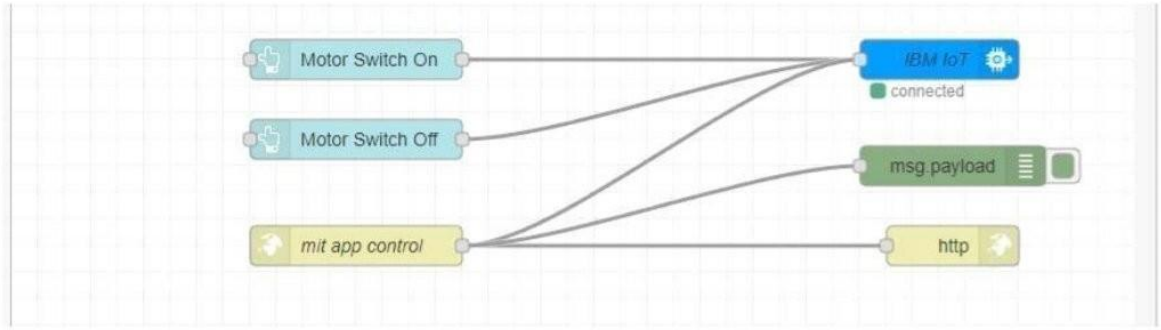


The above images show the java script codes of analyser and state function nodes.

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



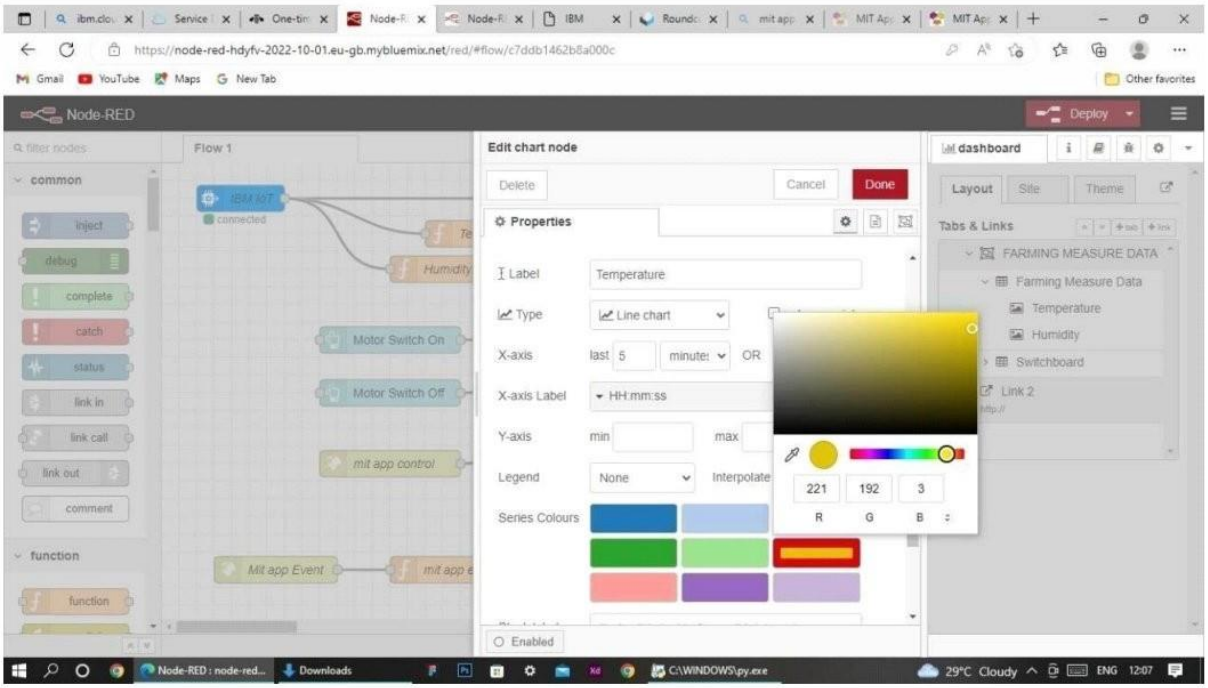
This is the program flow for sending commands to IBM cloud.

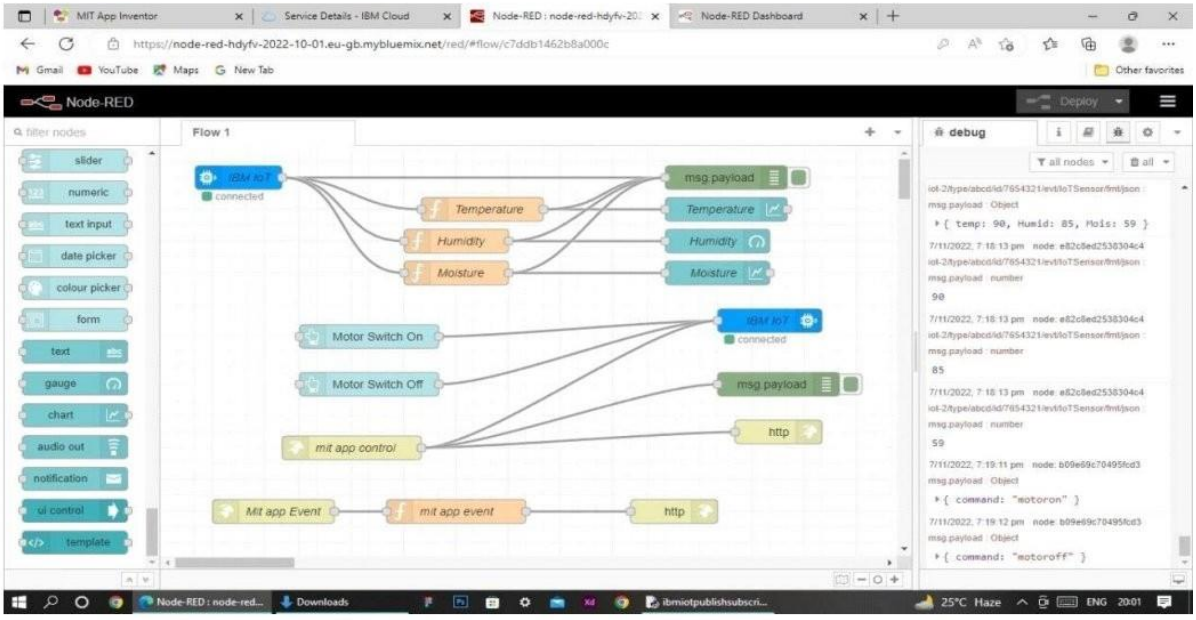
- ## Adjusting User Interface

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

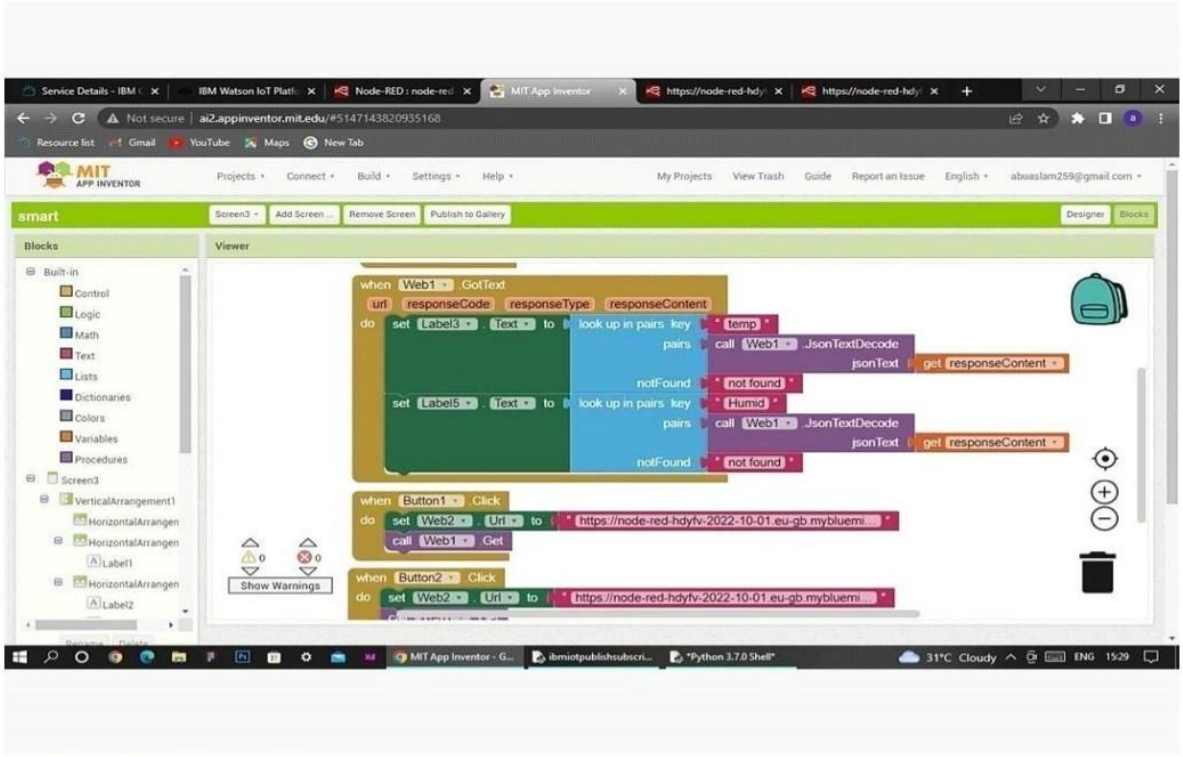
Here we are using Gauges, text and button nodes to display in the UI and helps tomonitor the parameters and control the farm equipment.

Below images are the Gauge, text and button node configurations.



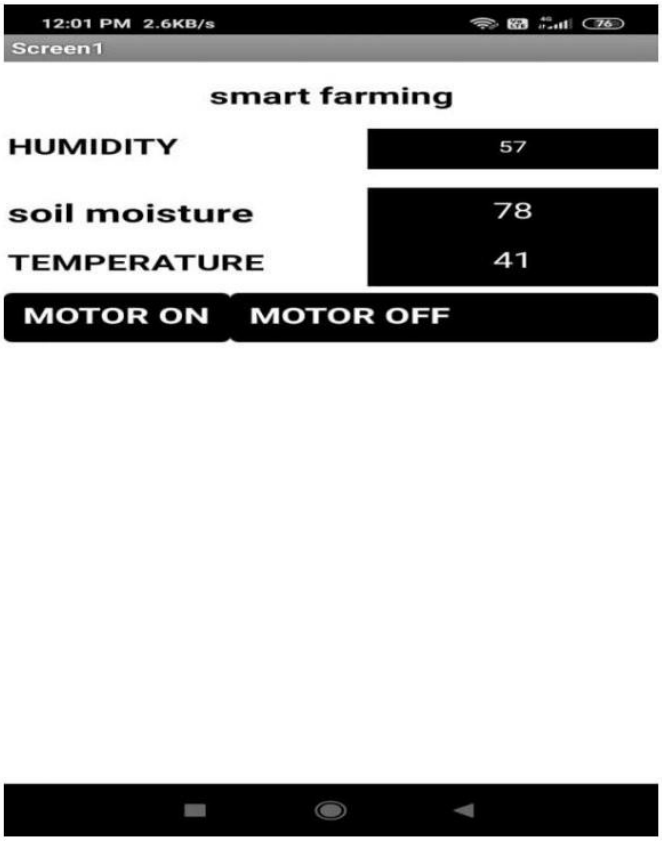


Complete Program Flow



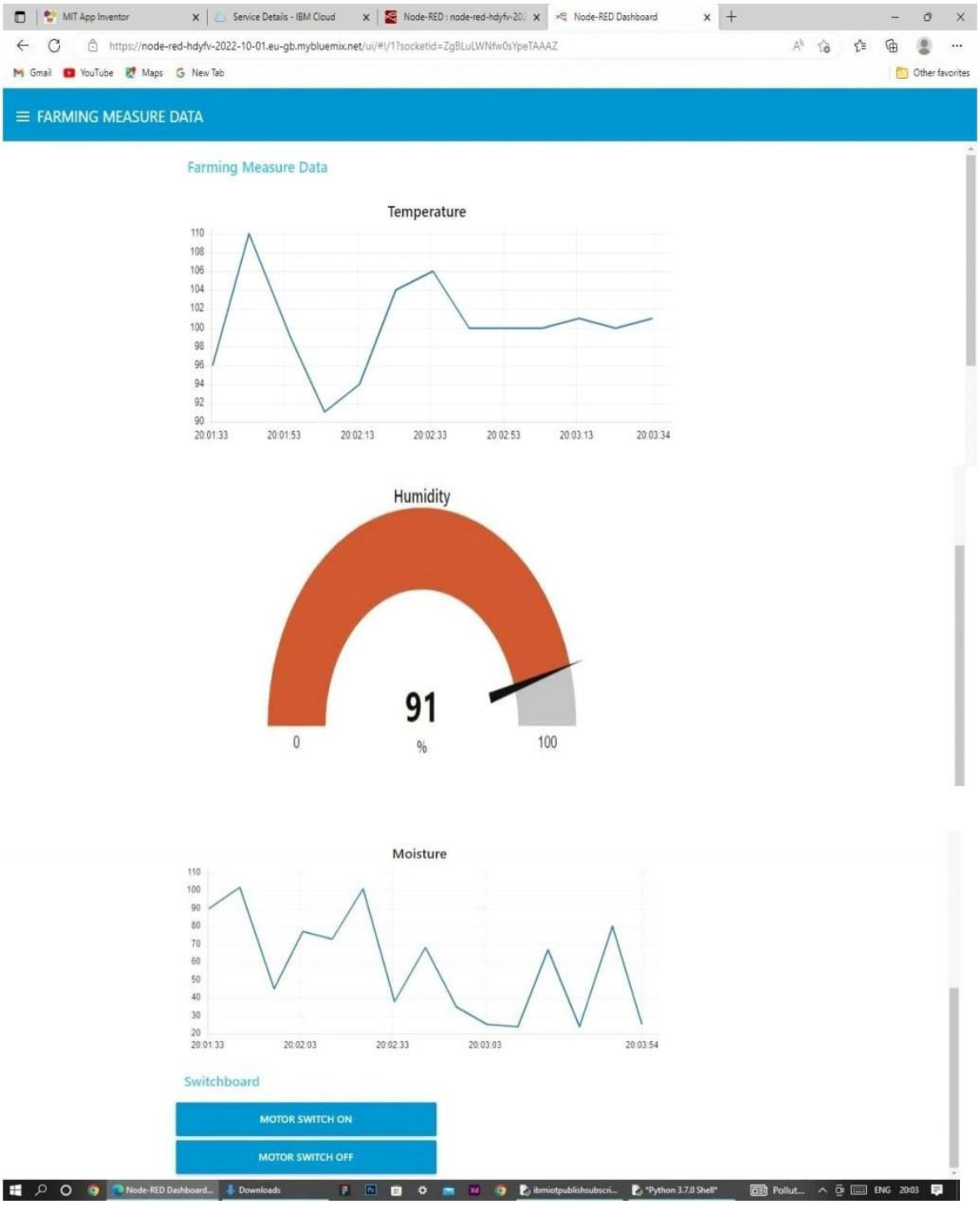
MOBILE APP WEB :

BLOCK DIAGRAM



SCREEN

Web APP UI Home Tab

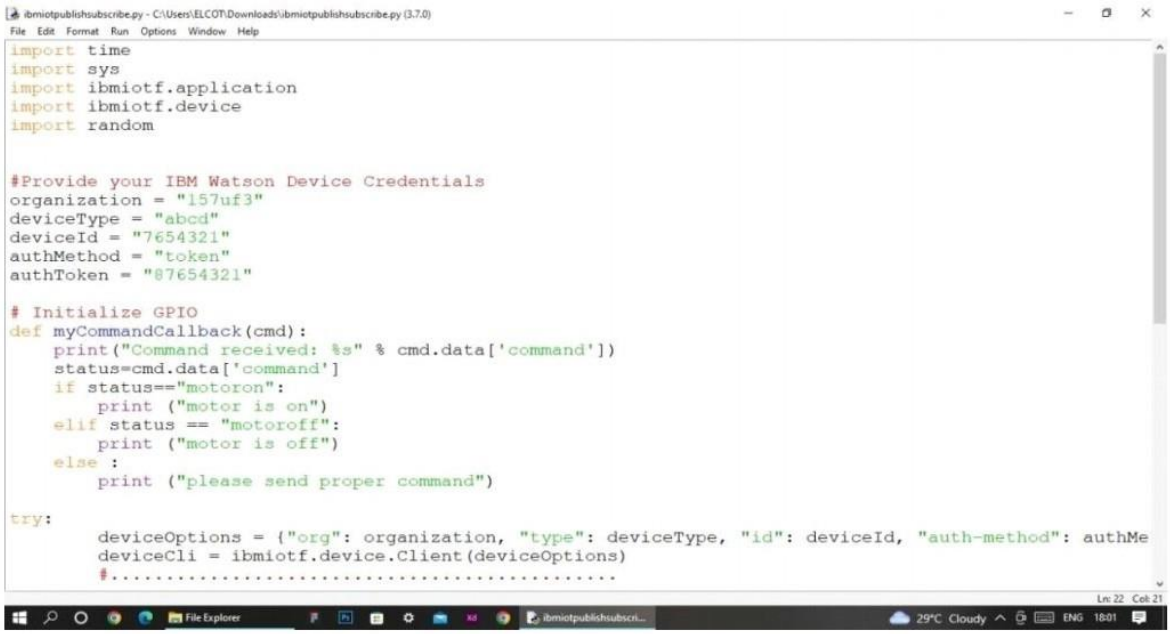

MIT App Inventor
Service Details - IBM Cloud
Node-RED : node-red-hdyfv-20
Node-RED Dashboard
https://node-red-hdyfv-2022-10-01.eu-gb.mybluemix.net/ui/#!/1?socketid=ZgBLulWNfw0sYpeTAAAZ
Gmail
YouTube
Maps
New Tab
Other favorites
FARMING MEASURE DATA
Farming Measure Data
Temperature
110
108
106
104
102
100
98
96
94
92
90
20:01:33
20:01:53
20:02:13
20:02:33
20:02:53
20:03:13
20:03:34
Humidity
91
%
0
100
Moisture
110
100
90
80
70
60
50
40
30
20
20:01:33
20:02:03
20:02:33
20:03:03
20:03:54
Switchboard
MOTOR SWITCH ON
MOTOR SWITCH OFF
Node-RED Dashboard...
Downloads
ibmiotpublishsubscri...
Python 3.7.0 Shell
Pollut...
ENG
20:03

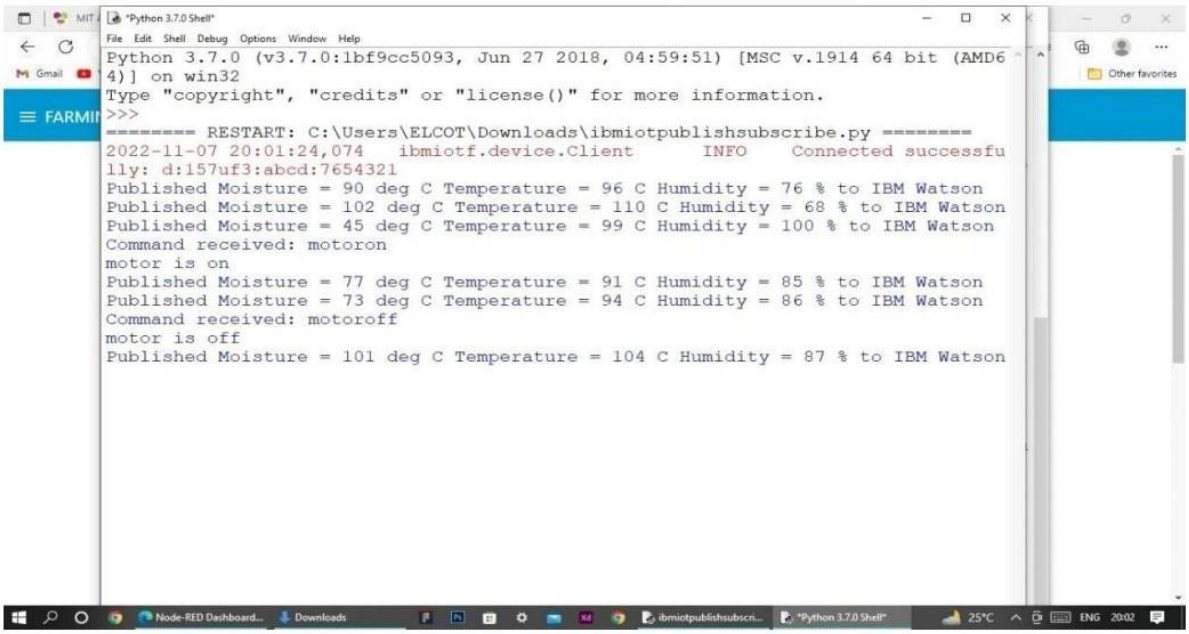
5.5 Receiving commands from IBM cloud using Python program

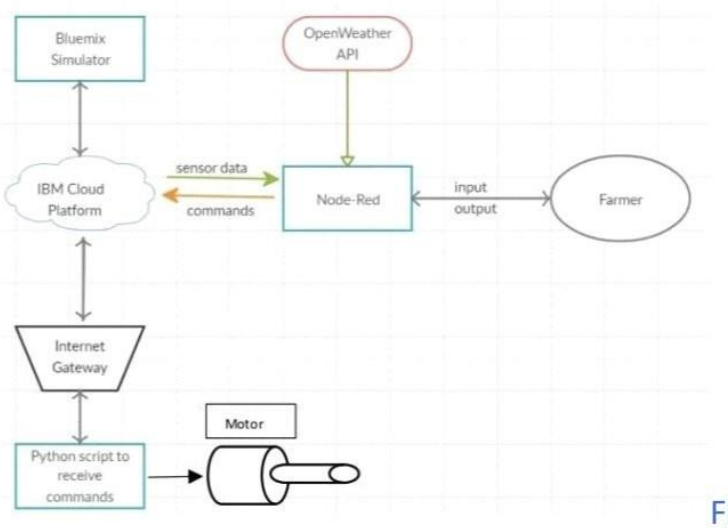
```
import time
importsys
import
ibmiotf.application
import ibmiotf.device
importrandom
```

```
#Provide your IBM Watson Device Credentials
organization = "cpeq2u"

deviceType = "TestDevice"

 deviceId = "802001" authMethod =

"token"authToken =

"8))+idxmB_q2PM@uvP"


# Initialize GPIO
def   myCommandCallback(cmd):                              print("Command received: %s" % cmd.da
      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 anevent of type "greeting" 10 times deviceCli.connect()
```

```
while True:
      #Get Sensor Data from
DHT11
temp=random.randint(90,11
0)
Humid=random.randint(60,1
00) Mois=random.
Randint(20,120)
   data = { 'temp' : temp, 'Humid':
Humid ,‘Mois’: Mois}
      #print data        defmyOnPublishCallback():
         print ("Published Temperature = %s C" % temp, "Humidity = %s
%%" %Humid, “Moisture =%s deg c” % Mois “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()
```

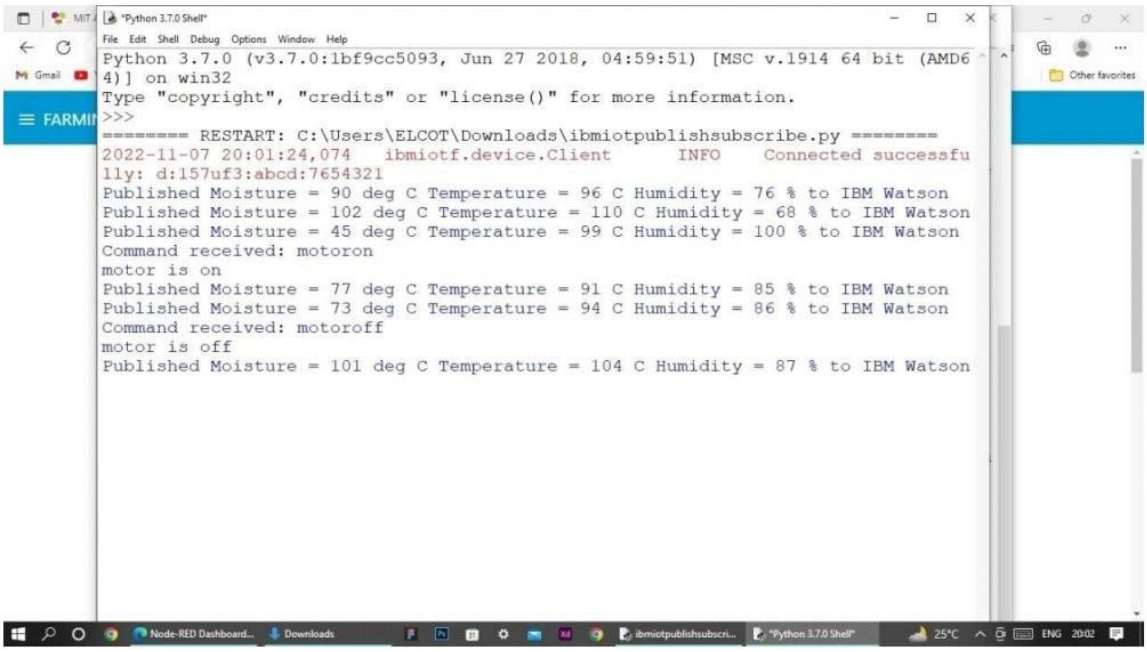

ibmiotpublishsubscribe.py - C:\Users\ELCOT\Downloads\ibmiotpublishsubscribe.py (3.7.0)
File Edit Format Run Options Window Help
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "157uf3"
deviceType = "abcd"
deviceId = "7654321"
authMethod = "token"
authToken = "87654321"
# 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": authMe
deviceCli = ibmiotf.device.Client(deviceOptions)
#.....
Ln: 22 Col: 21
File Explorer
ibmiotpublishsubscri...
29°C Cloudy
ENG
18:01

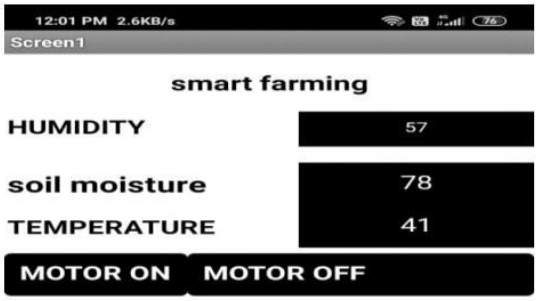

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\ELCOT\Downloads\ibmiotpublishsubscribe.py ========
2022-11-07 20:01:24,074 ibmiotf.device.Client INFO Connected successfully: d:157uf3:abcd:7654321
Published Moisture = 90 deg C Temperature = 96 C Humidity = 76 % to IBM Watson
Published Moisture = 102 deg C Temperature = 110 C Humidity = 68 % to IBM Watson
Published Moisture = 45 deg C Temperature = 99 C Humidity = 100 % to IBM Watson
Command received: motoron
motor is on
Published Moisture = 77 deg C Temperature = 91 C Humidity = 85 % to IBM Watson
Published Moisture = 73 deg C Temperature = 94 C Humidity = 86 % to IBM Watson
Command received: motoroff
motor is off
Published Moisture = 101 deg C Temperature = 104 C Humidity = 87 % to IBM Watson
Gmail
Other favorites
Node-RED Dashboard...
Downloads
ibmiotpublishsubscri...
Python 3.7.0 Shell
25°C
ENG
20:02

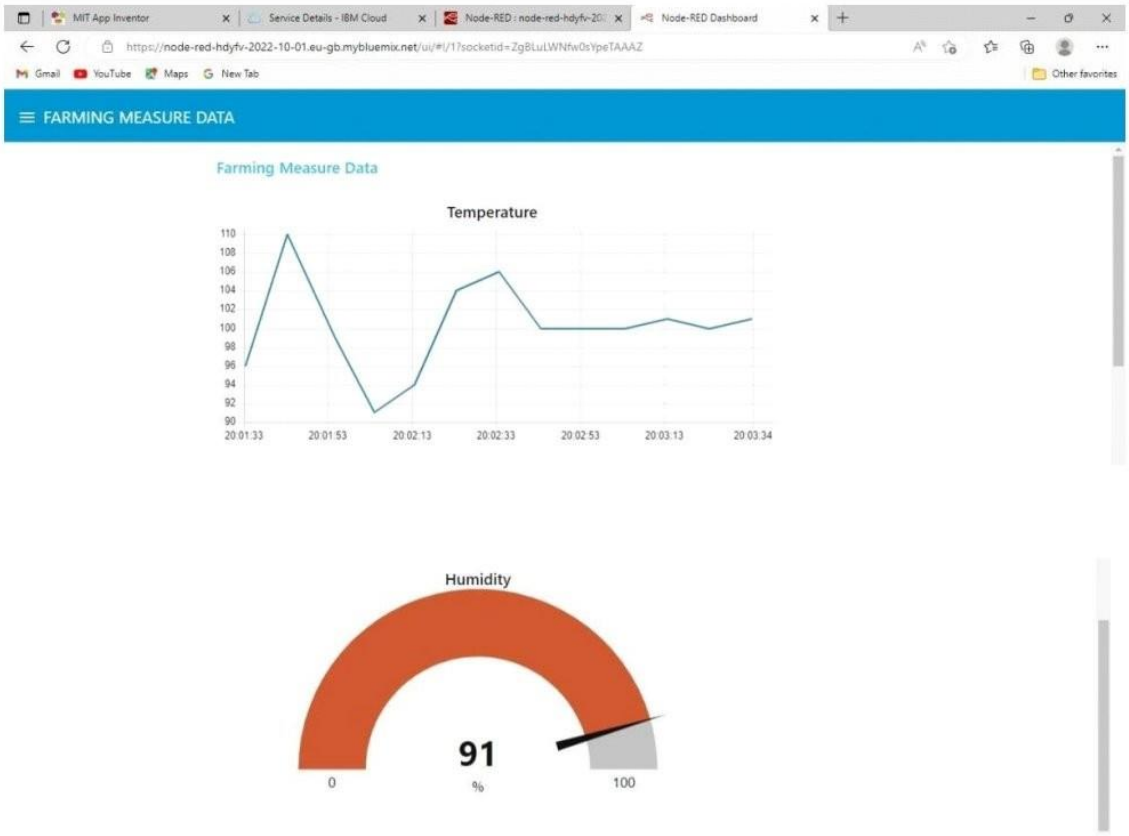
- 

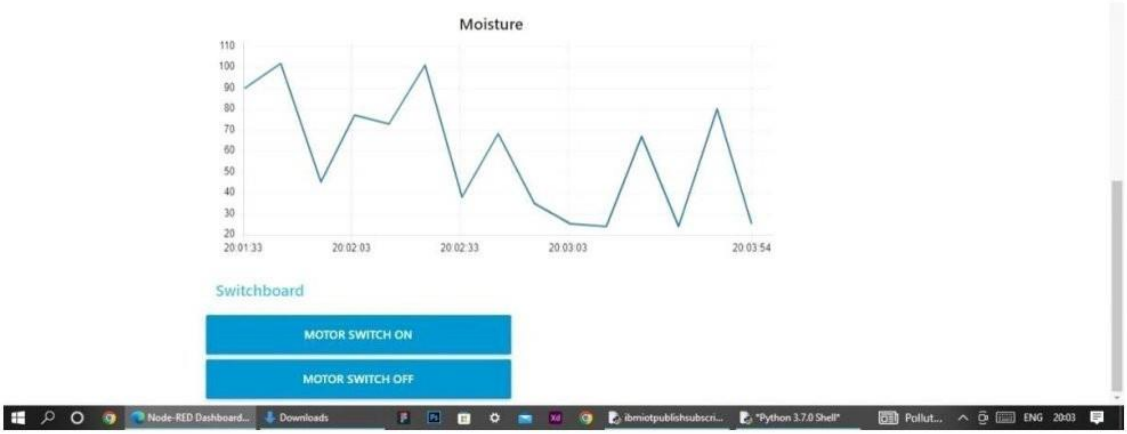
Flow Chart

- Observations & Results




12:01 PM 2.6KB/s
Screen1
smart farming
HUMIDITY
57
soil moisture
78
TEMPERATURE
41
MOTOR ON
MOTOR OFF


MIT App Inventor
Service Details - IBM Cloud
Node-RED Dashboard
https://node-red-hdyfv-2022-10-01.eu-gb.mybluemix.net/ui/#!/1?socketid=ZgBLuLWNfw0sYpeTAAAZ
Gmail
YouTube
Maps
New Tab
Other favorites
FARMING MEASURE DATA
Farming Measure Data
Temperature
110
108
106
104
102
100
98
96
94
92
90
20:01:33
20:01:53
20:02:13
20:02:33
20:02:53
20:03:13
20:03:34
Humidity
91
%
0
100



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

living.Disadvantages:

- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.
- Farmers wanted to adapt the use of Mobile

App.9.Conclusion

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

10.Bibliography

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

IoT simulator : https://watson-iot-sensor-

simulator.mybluemix.net/OpenWeather :

https://openweathermap.org/