SMART FARMING APPLICATION
USING IoT

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING
IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

Team ID: PNT2022TMID22759

ADHIKESAVAN G (732919ECR005)
ASHIFA M (732919ECR014)
HARIPRIYA P (732919ECR042)
JAGADEESH B K (732919ECR048)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY, THINDAL.

NOV-2022

VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**SMART FARMING APPLICATION"** is the bonafide record work done by **Mr ADHIKESAVAN G** (**732919ECR005**), **Ms ASHIFA M** (**732919ECR014**), **Ms HARIPRIYA P**(**732919ECR042**) and **Mr JAGADEESH B K** (**732919ECR048**) for **IBM- NALAIYATHIRAN** in **VII** semester of **B.E.,** degree course **in Electronics and Communication Engineering** branch during the academic year of 2022 - 2023.

Staff-In charge
Mr. N. KARTHIKEYAN
B.E., M.E.,

Evaluator
Dr. M. PARIMALA DEVI
B.E., M.E., M.B.A., Ph.D.,

Head of the Department
Dr. M. NISHA ANGELINE ME., Ph.D.,

ACKNOWLEDGEMENT

We express our breathless thanks to our **Prof. P. JAYACHANDAR M.E.,** the Dean, Velalar College of Engineering and Technology, Thindal, for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to Head of the Department **Dr. M. NISHA ANGELINE ME., Ph.D.,** for giving us valuable suggestion and help towards us throughout this Project.

We are highly grateful to thank our Project coordinator **Asst. Prof. N. KARTHIKEYAN B.E., M.E.,** and our Project Evaluator **Dr. M. PARIMALA DEVI B.E., M.E., Ph.D.,** Department of Electronics and Communication Engineering, Velalar College of Engineering and Technology, Thindal, for coordinating us throughout this Project.

We are very much indebted to thank all the faculty members of Department of Computer science and Engineering in our Institute, for their excellent moral support and suggestions to complete our Project work successfully.

Finally our acknowledgment does our parents, sisters and friends those who had extended their excellent support and ideas to make our Project a pledge one.

Adhikesavan G
Ashifa M
Haripriya P
Jagadeesh B K

CONTENTS

Introduction

The main aim of this project is to help farmers automate their farms by providing them witha Web

App through which they can monitor the parameters of the field like Temperature, soilmoisture, humidity, etc. and control the equipment like water motor and other devices remotelyvia internet without their actual presence in the field.

1. 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 monitoredby 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.

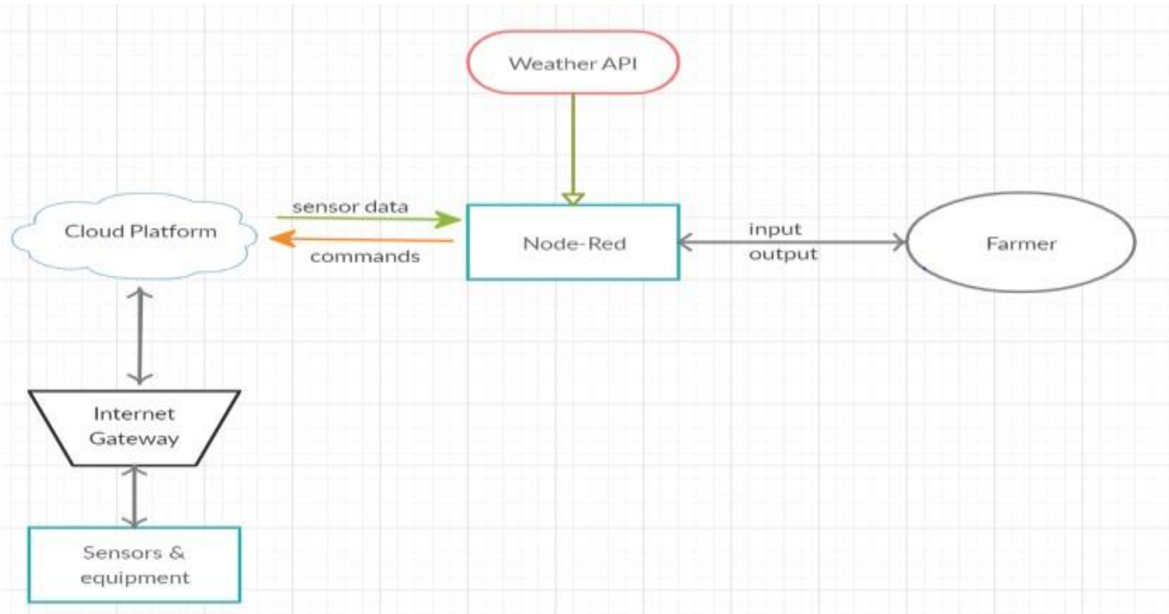
2. Proposed Solution

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.

3. Theoretical Analysis

3.1 Block Diagram

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



3.2 Required Software Installation

3.2.1 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

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

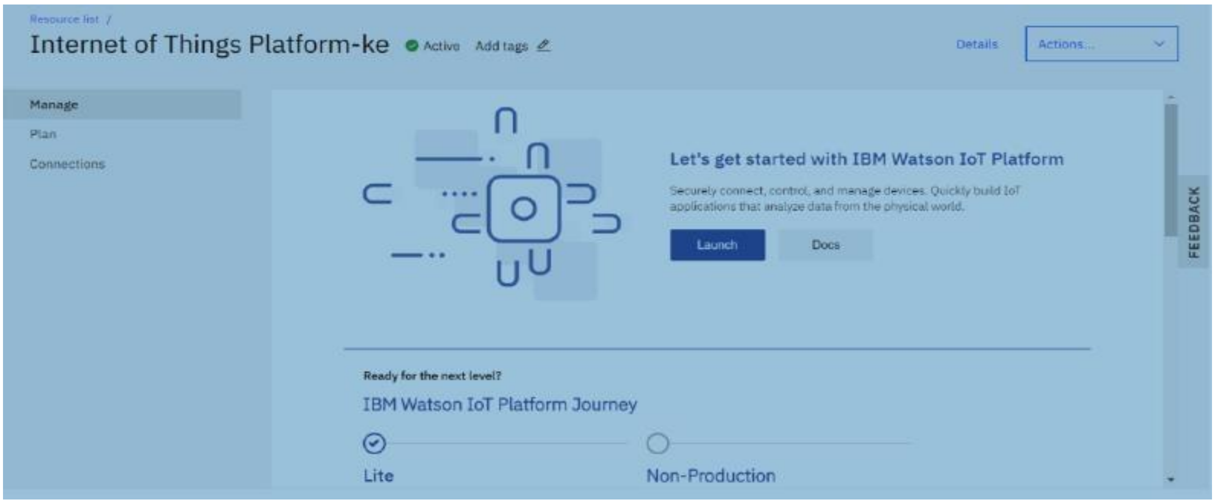
Installation of IBM IoT and Dashboard nodes for Node-Red

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

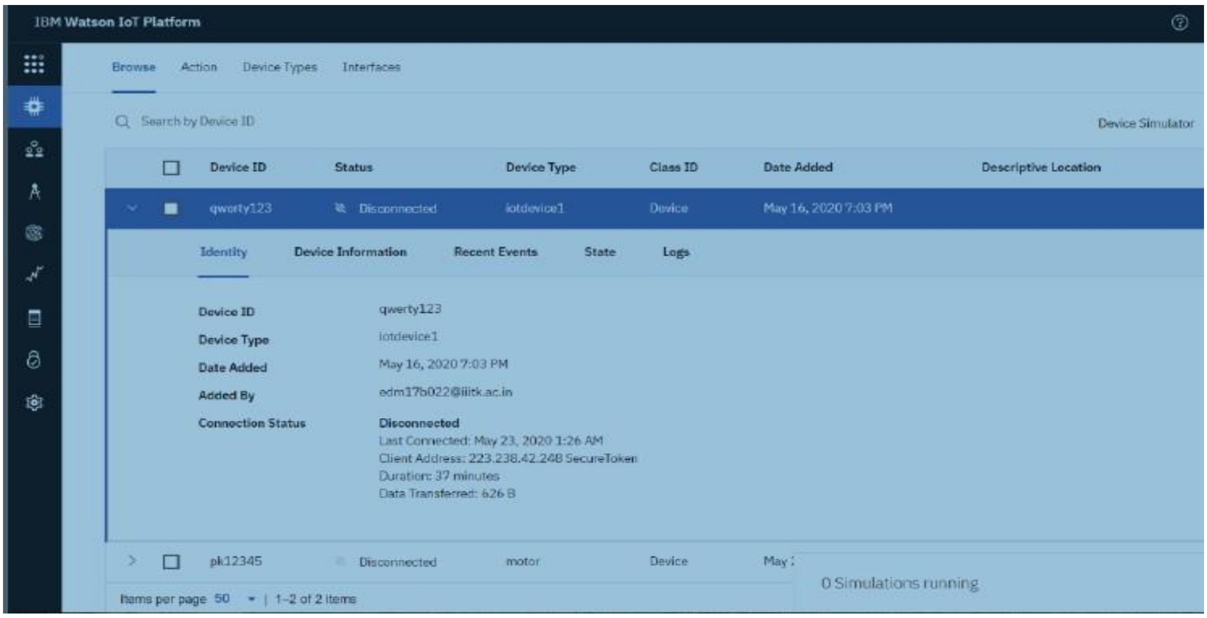
3.2.2 IBM Watson IoT Platform

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:

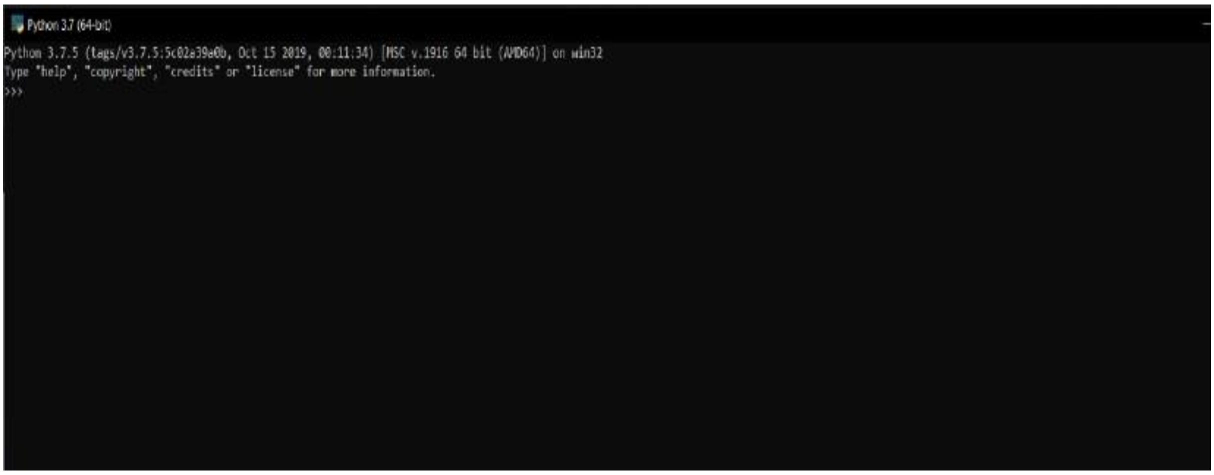
- Create an account in IBM cloud using your email ID
- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.



3.2.3 Python IDE

Install Python3 compiler

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



3.3 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.

3.4 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}

Link I used in my project:

http://api.openweathermap.org/data/2.5/weather?q=Gudur,in&appid=62354068e45f41ffa6a5b164714145fe

5. Building Project

5.1 Connecting IoT Simulator to IBM Watson IoT Platform

Open link provided in above section 4.3

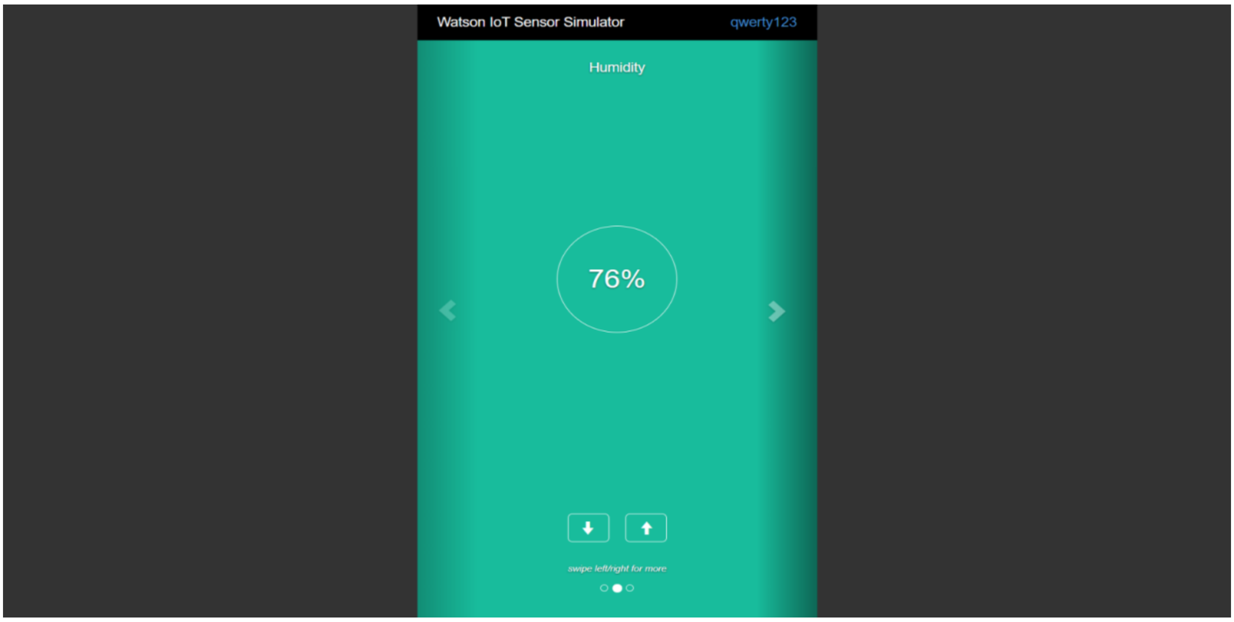
Give the credentials of your device in IBM Watson IoT Platform

Click on connect

My credentials given to simulator are:

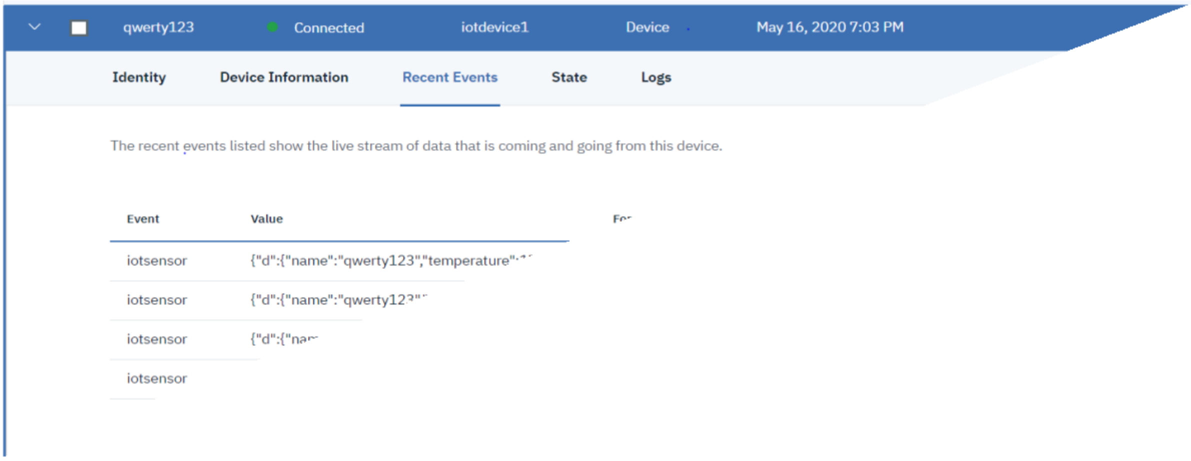
Org ID	: 9wbx5	API	: a-9wbx51qfklrf7jl
Device type	: iotdevice1	token	: JcU (4(9Z37PdL!Rmz(
Device ID	: qwerty123		

Device Token : johnyjohnyyespapa



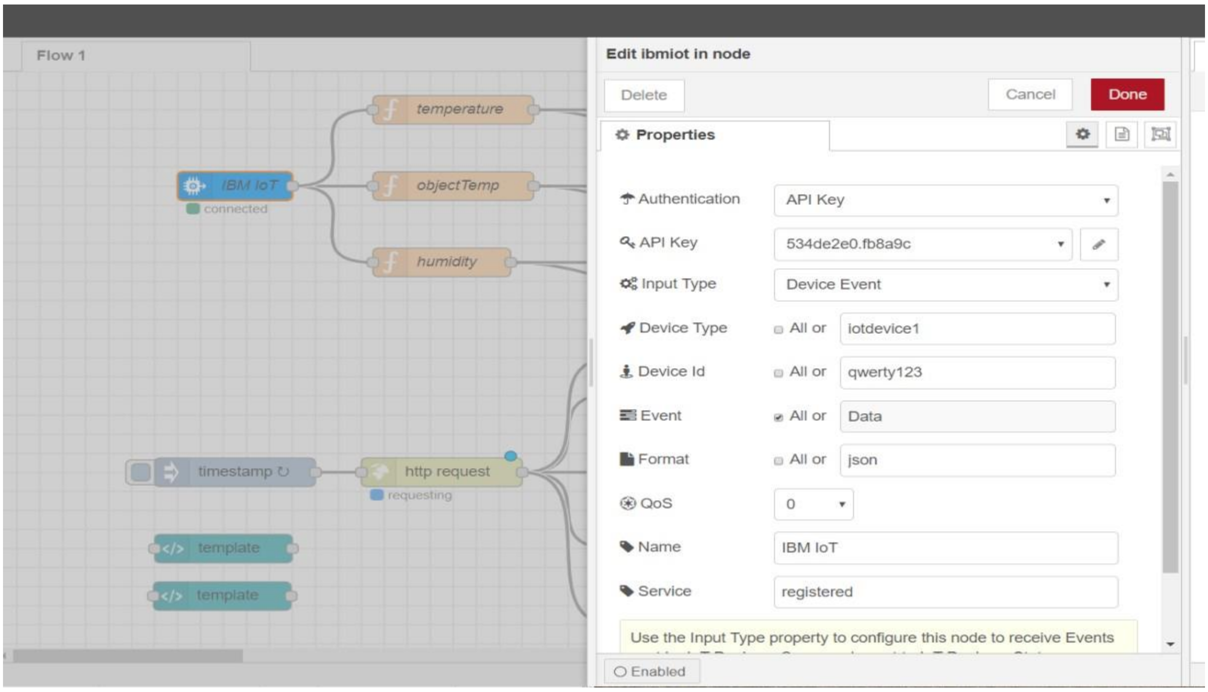
- ➢ You will receive the simulator data in cloud
- ➢ You can see the received data in Recent Events under your device
- ➢ Data received in this format (json)

- {
  - "d": {
    - "name": "qwerty123",
    - "temperature": 17,
    - "humidity": 76,
    - "objectTemp": 25
  - }
- }



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

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

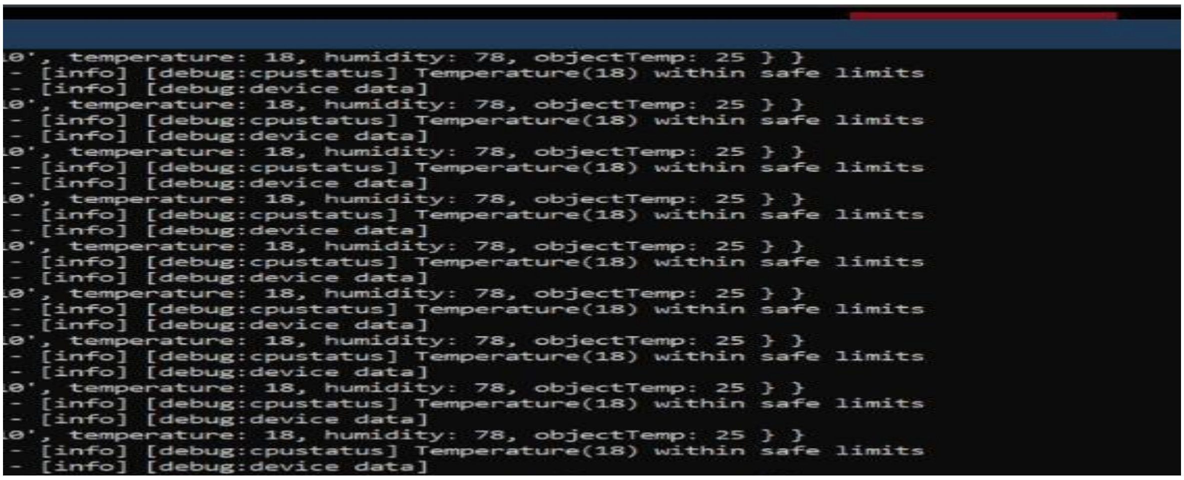
Display the data using debug node for verification

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

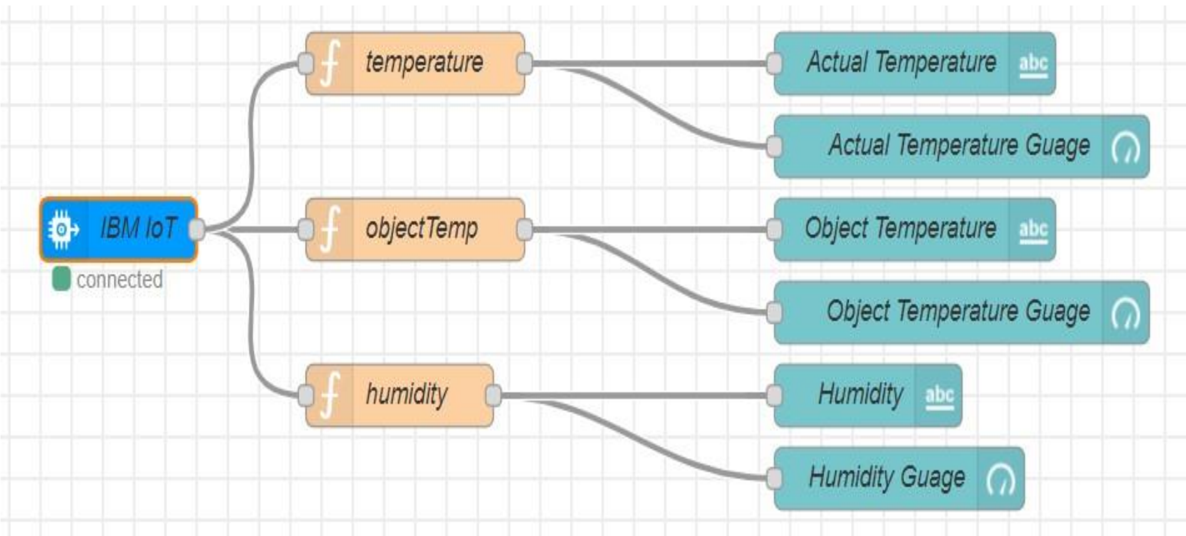
The Java script code for the function node is:

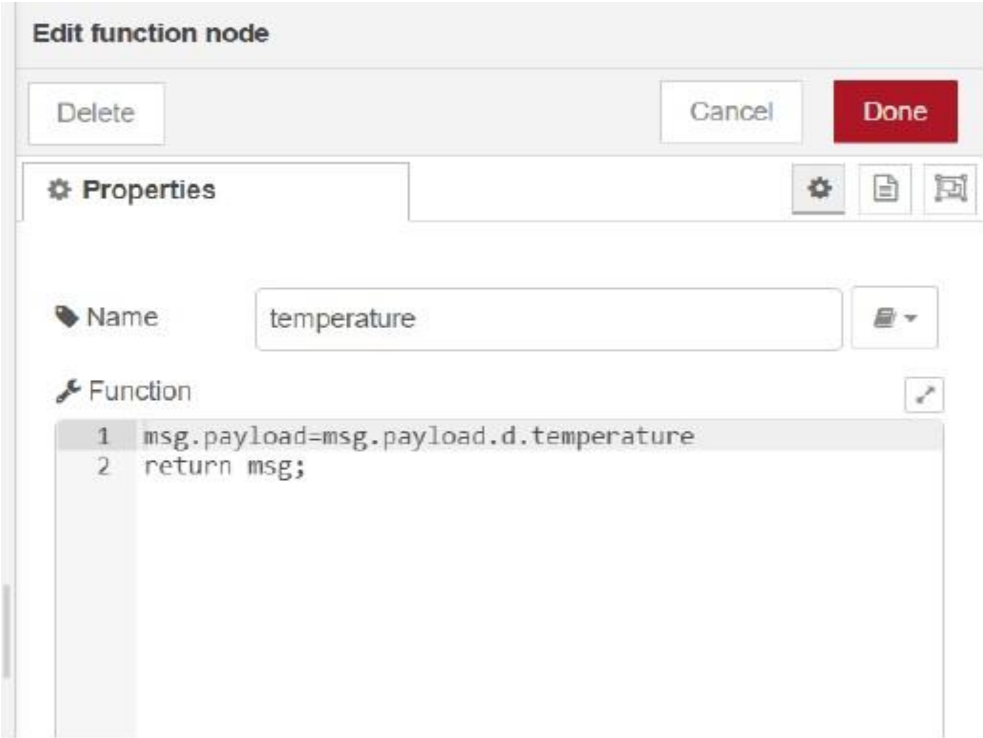
```
msg.payload=msg.payload.d.temperature
return msg;
```

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



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 Open Weather

The Node-Red also receive data from the Open Weather API by HTTP GET request. Aninject 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 Open Weather after request is in below JSON

format:

{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds","descriptio n":" broken clouds","icon":"04n"}],"base":"stations","main":{"temp":307.59,"feels_like":305.5,"temp_mi n":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":{"countr y":"IN","sunrise":158993

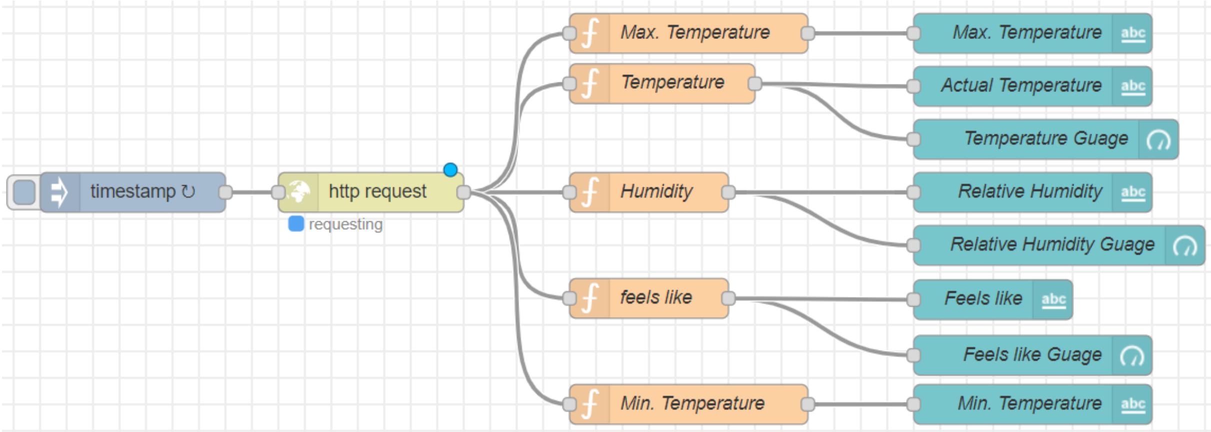
3553,"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod": 200}

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

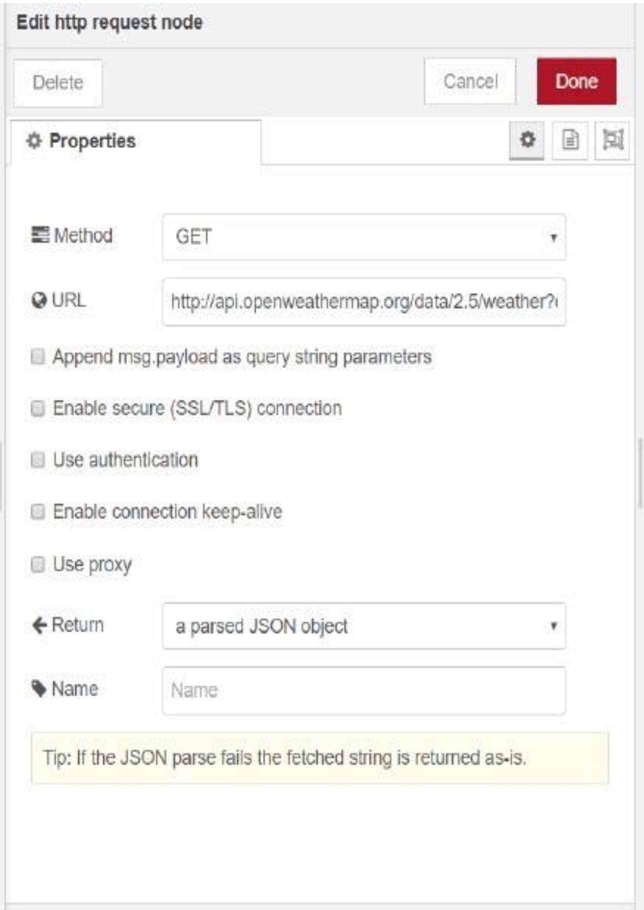
```
Var temperature = msg.payload.main.temp;

Temperature = temperature-273.15;

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

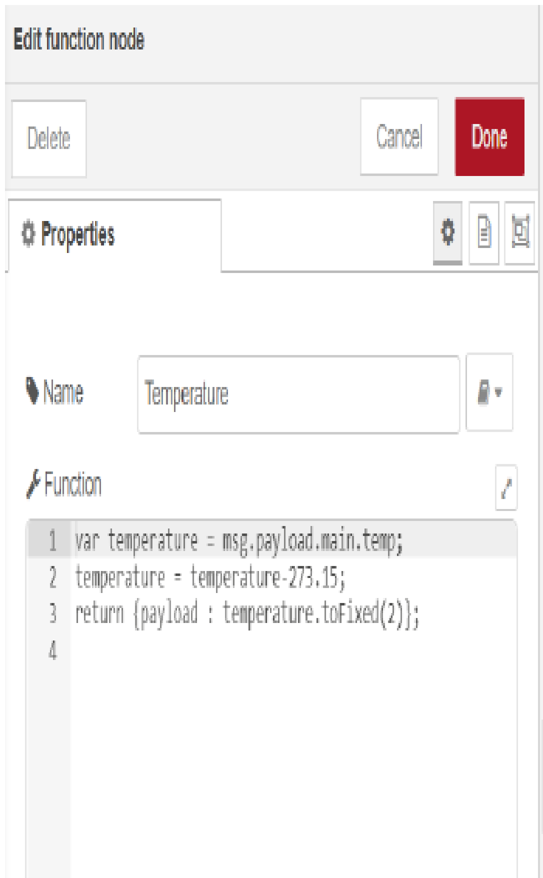
In the above Java script code we take temperature parameter into a new variable and convert it from kelvin to Celsius

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



The above image has the program flow for receiving data from Open Weather





The above two images contain http request and function node data that needs to be filled.

5.4 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 Watson device.

Here we add three button sin UI which each sends anumber0, 1and 2.

0->for motor off

1 -> for motor on

2-> for running motor continuously30 minutes

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

The Java script code for the analyser is:

```
if(msg.payload===1)
msg.payload={"command":"ON"};
else if(msg.payload===0)
msg.payload={"command":"OFF"};
else
msg.payload={"command":"runfor30minutes"};
return msg;
```

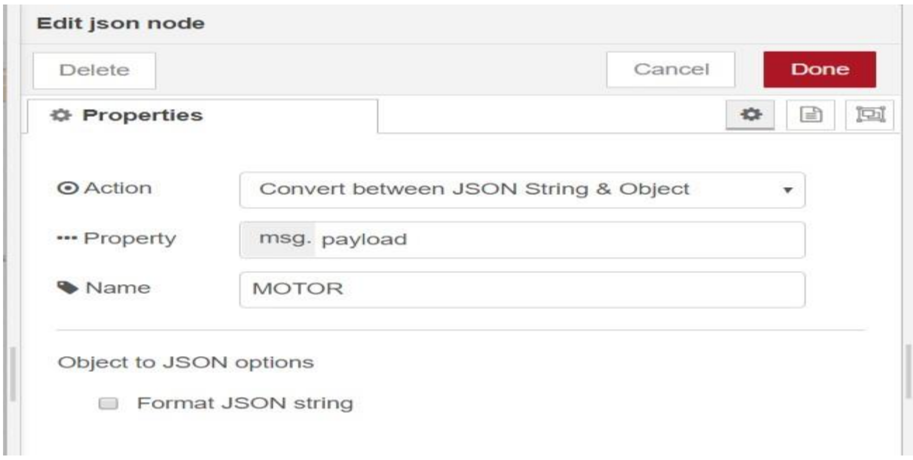
Then we use another function node to parse the data and get the command and represent it visually with text node.

The Java script code for that function node is:

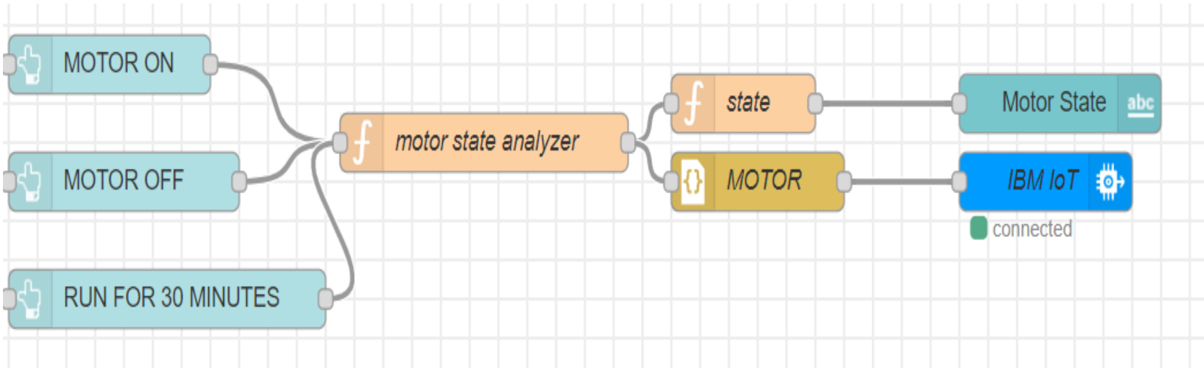
```
var state=msg.payload;
msg.payload = state.command;
return msg;
```

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



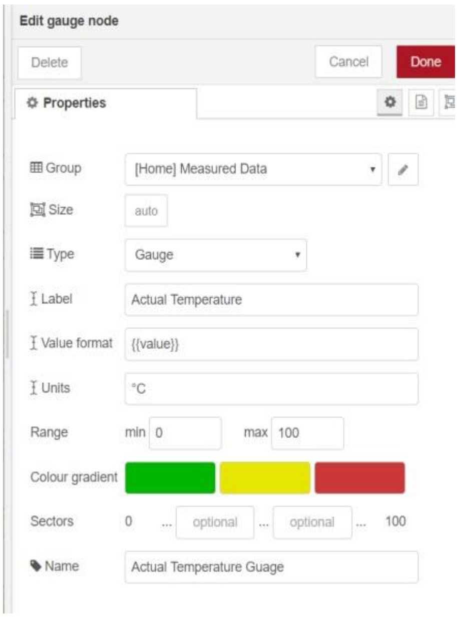
This is the program flow for sending commands to IBM cloud.

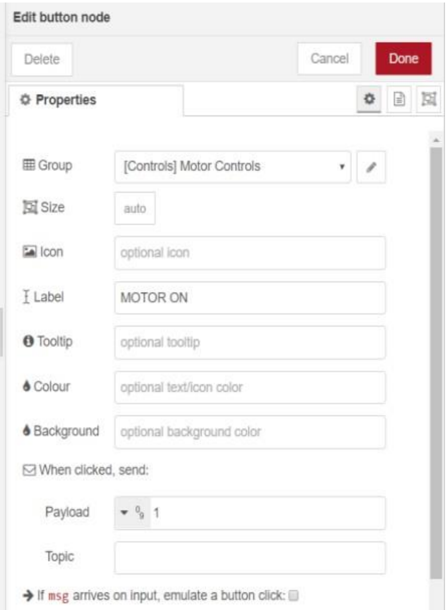
5.5 Adjusting User Interface

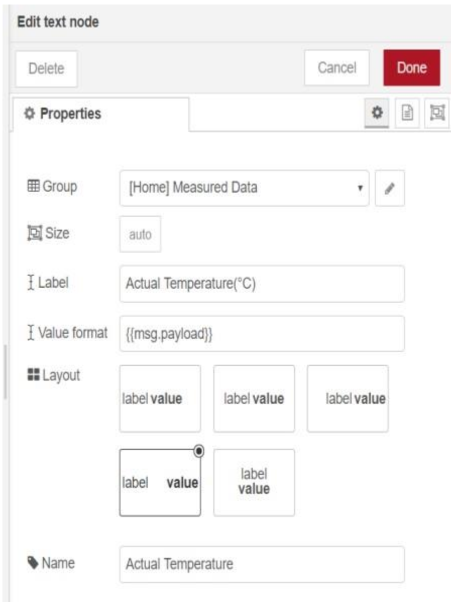
In order to display the parsed JSON data a Node-Red dashboard is created

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



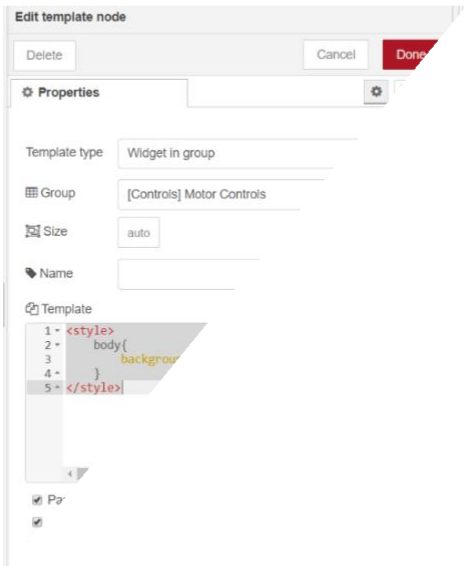




Adding Background image to the UI

To add the background image we are going to add template node and configure it with below HTML code

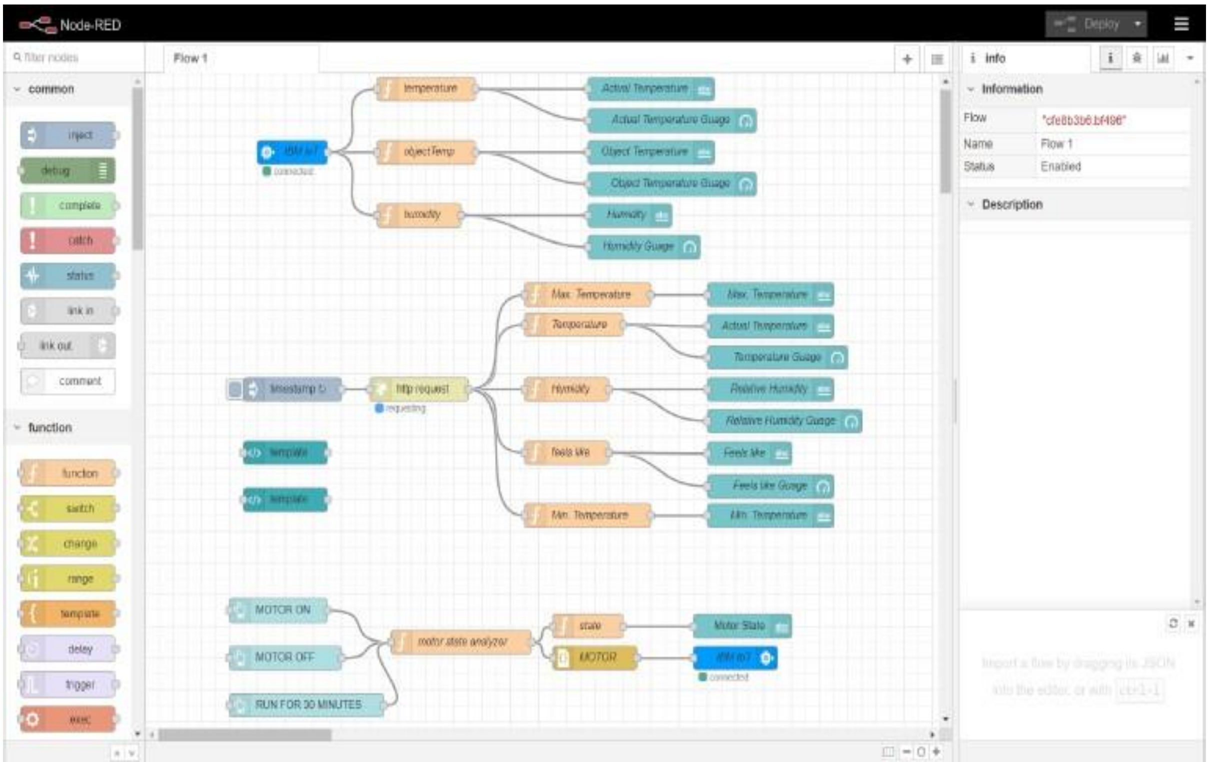
```
<style>

body {
        Background-image: url("https://images.
        unsplash.com/photo-1563514227147-6d2ff665
        a6a0?ixlib=rb-1.2.1&auto=format&fit=crop&w
        =1951&q=80");
      }

</style>
```



We add URL of image that need to be displayed

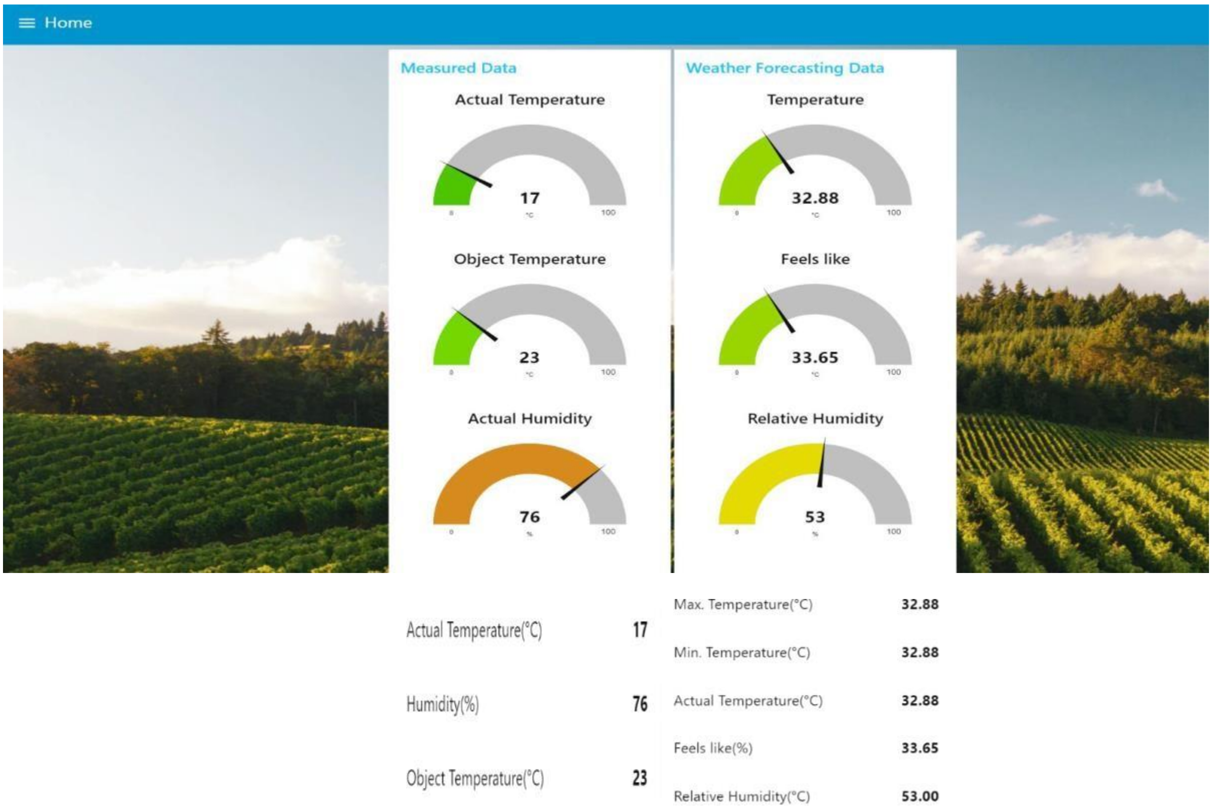
Configuration is shown in the beside image

Complete Program Flow

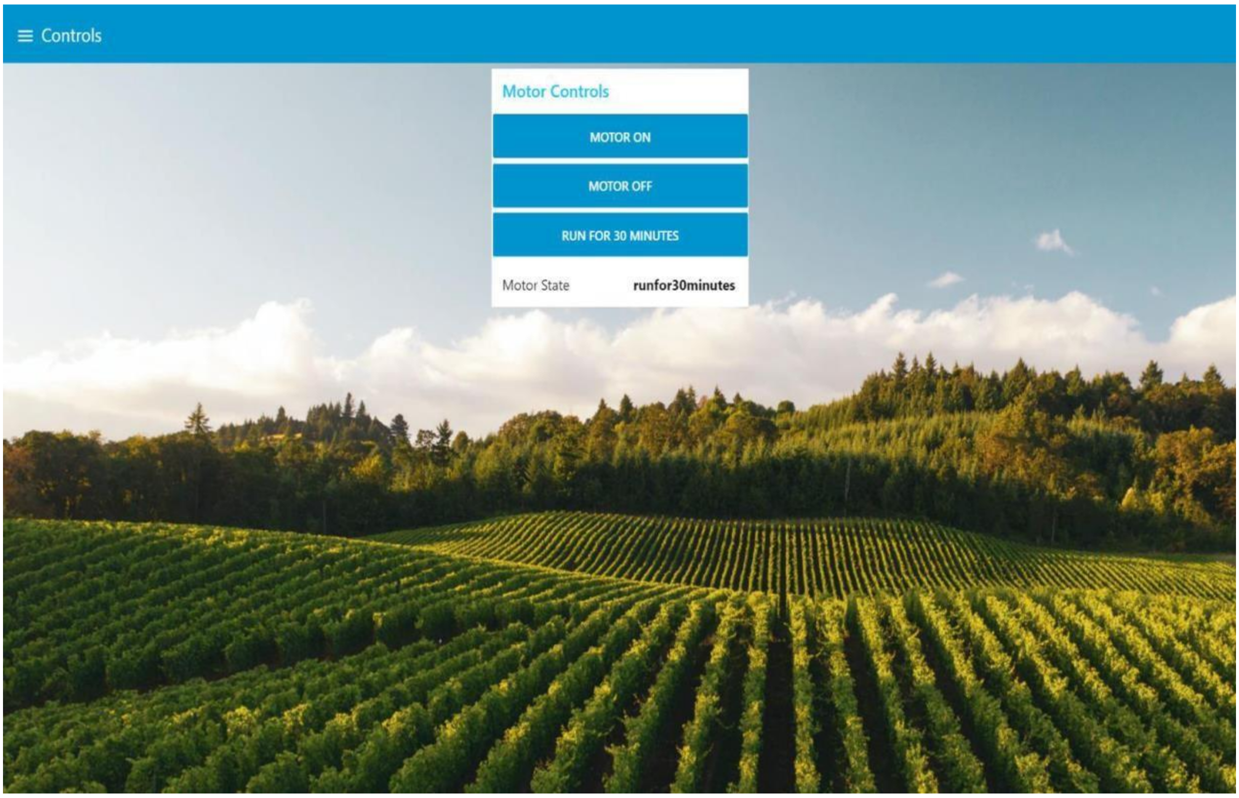


Web APP UI

Home Tab



Controls Tab

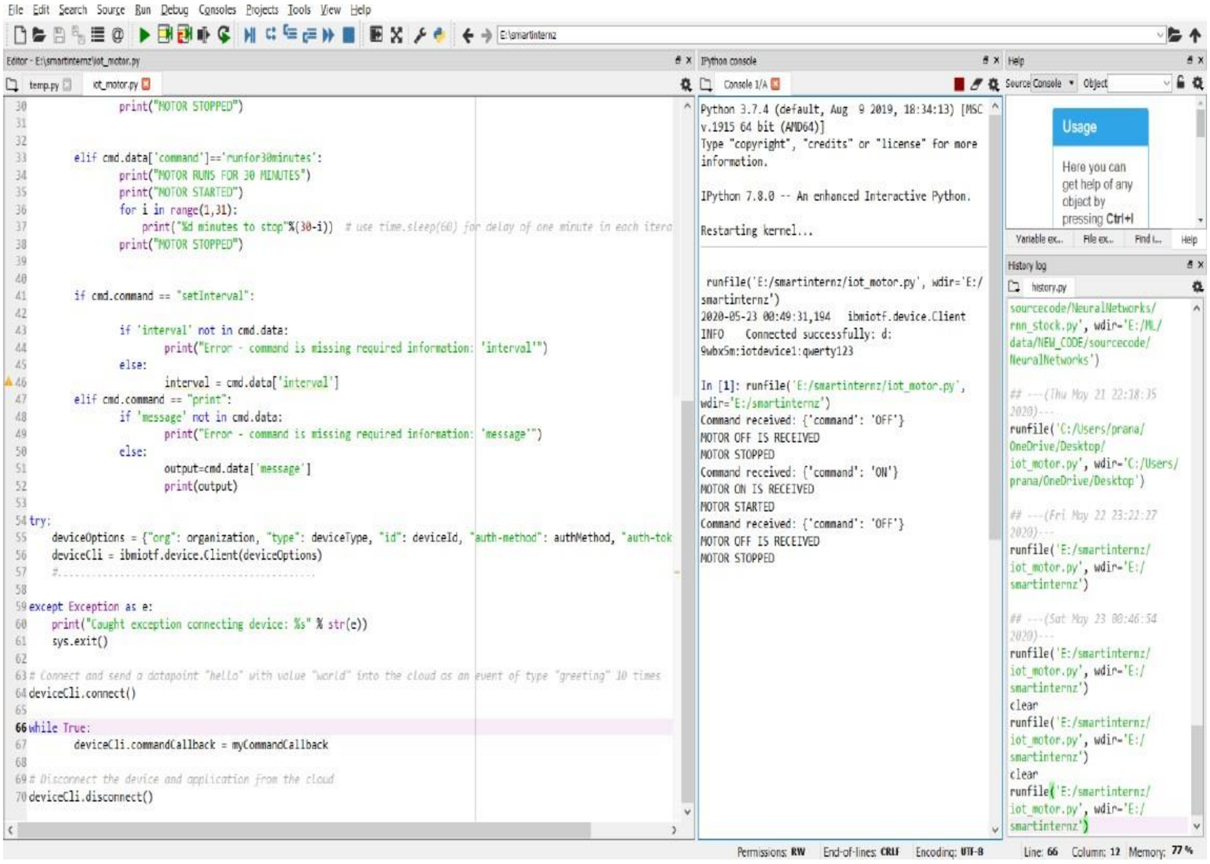


5.5Receiving commands from IBM cloud using Python program

This is the Python code to receive commands from cloud to any device like Raspberry Pi in the farm

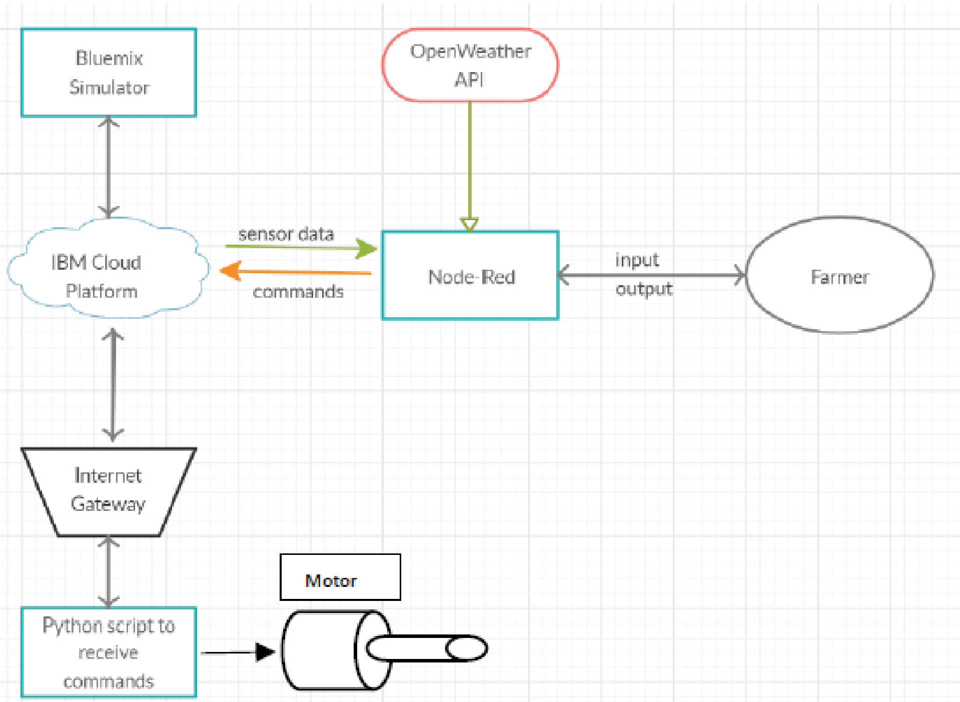
```
import time

import sys

import ibmiotf.application

import ibmiotf.device

organization = "9wbx5m"

deviceType = "iotdevice1"

deviceId = "qwerty123"

authMethod = "token"

authToken = "johnyjohnyyespapa"
```

```
    def myCommandCallback(cmd):

        print("Command received: %s" % cmd.data)

        if cmd.data['command']=='ON':

        print("MOTOR ON IS RECEIVED")

        time.sleep(1)

        print("MOTOR STARTED")
    elif cmd.data['command']=='OFF':
    print("MOTOR OFF IS RECEIVED")

    time.sleep(1)

    print("MOTOR STOPPED")

    elif cmd.data['command']=='runfor30minutes':

    print("MOTOR RUNS FOR 30 MINUTES")

    print("MOTOR STARTED")

    for i in range(1,31):

    print("%d minutes to stop"%(30-i)) # use time.sleep(60) for delay of one minute

    print("MOTOR STOPPED")
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()

deviceCli.connect()

while True:

    deviceCli.commandCallback = myCommandCallback

• deviceCli.disconnect()
```



File Edit Search Source Run Debug Consoles Projects Tools View Help
print("MOTOR STOPPED")
elif cmd.data['command']=='runfor30minutes':
print("MOTOR RUNS FOR 30 MINUTES")
print("MOTOR STARTED")
for i in range(1,31):
print("%d minutes to stop"%(30-i))
print("MOTOR STOPPED")
if cmd.command == "setInterval":
if 'interval' not in cmd.data:
print("Error - command is missing required information: 'interval'")
else:
interval = cmd.data['interval']
elif cmd.command == "print":
if 'message' not in cmd.data:
print("Error - command is missing required information: 'message'")
else:
output=cmd.data['message']
print(output)
try:
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:
deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
Python 3.7.4 (default, Aug 9 2019, 18:34:13)
Type "copyright", "credits" or "license" for more information.
IPython 7.8.0 -- An enhanced Interactive Python.
Restarting kernel...
Command received: {'command': 'OFF'}
MOTOR OFF IS RECEIVED
MOTOR STOPPED
Command received: {'command': 'ON'}
MOTOR ON IS RECEIVED
MOTOR STARTED
Command received: {'command': 'OFF'}
MOTOR OFF IS RECEIVED
MOTOR STOPPED
Usage
Here you can get help of any object by pressing Ctrl+I

6. FLOW CHART



Bluemix Simulator
OpenWeather API
IBM Cloud Platform
sensor data
commands
Node-Red
input
output
Farmer
Internet Gateway
Motor
Python script to receive commands

7. OBSERVATIONS & RESULTS

```
Connected successfully: d:9wbx5m:iotdevice1:qwerty123
Command received: {'command': 'ON'}
MOTOR ON IS RECEIVED
MOTOR STARTED
Command received: {'command': 'OFF'}
MOTOR OFF IS RECEIVED
MOTOR STOPPED
Command received: {'command': 'runfor30minutes'}
MOTOR RUNS FOR 30 MINUTES
MOTOR STARTED
29 minutes to stop
28 minutes to stop
27 minutes to stop
26 minutes to stop
25 minutes to stop
24 minutes to stop
23 minutes to stop
22 minutes to stop
21 minutes to stop
20 minutes to stop
19 minutes to stop
18 minutes to stop
17 minutes to stop
16 minutes to stop
15 minutes to stop
14 minutes to stop
13 minutes to stop
12 minutes to stop
11 minutes to stop
10 minutes to stop
9 minutes to stop
8 minutes to stop
7 minutes to stop
6 minutes to stop
5 minutes to stop
4 minutes to stop
3 minutes to stop
2 minutes to stop
1 minutes to stop
0 minutes to stop
MOTOR STOPPED
```

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Farms can be monitored and controlled remotely.
- Increase in convenience to farmers.
- Less labour 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 WebApp.

9. 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.

10. BIBLIOGRAPHY

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

Open Weather: https://openweathermap.org/