

## Project Report

# Smart Agriculture System Based on IOT

Online Internship under IOT Application

Developer

During May-June 2020

At



Submitted by :

**K MOHAMMAD HABEEB**

Project Id : SPS\_PRO\_101

Internship Id : SB52035

# I. Introduction

## I.1 Overview

## I.2 Purpose

# 2. Literature Survey

## 2.1 Existing problem

## 2.2 Proposed Solution

# 3.Theoretical analysis

## 3.1 Block Diagram

## 3.2 .Software Design

### 1) IBM Watson IOT Platform

### 2) Node Red

### 3) Python IDLE

## 3.3 Hardware Design

## 3.4 Open Weather API

# 4. Building Project

## 4.1 Connecting stimulator the IBM Watson cloud.

## 4.2 Configuration Of Node Red to get the data from IBM Cloud

## 4.3 Configuration Of Node Red to get the data from Open Weather

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

## 4.5 Building User Interface using Node Red

## 4.6 Receiving commands from IBM cloud using Python program

# 5. Flow chart

# 6. Observations & Results

# 7. Advantages & Disadvantage

# 8. Applications

# 9. Conclusion

# 10. Future Scope

# 11. Bibliography

# Introduction

## 1.1 Overview

The project is based on agriculture system, where the latest technology are used to monitor daily parameters, basically it includes live weather updates, sensors readings and also an option to control the motor. This project helps the farmers to save their precious time. The project includes the uses of different platforms like IBM cloud services , Node Red and Python.

## 1.2 Purpose

The main purpose of this smart agriculture application is to enhance the solutions of a farmer in which they are facing problems to yield a better crop. Farmer will provided with a web based application using which he can monitor the temperature, humidity and soil moisture parameters along with weather forecasting. Based on all the parameters he can water his crop by controlling the motors using the web application. Even if the farmer is not present near his crop he can water his crop by controlling the motors using the mobile application from anywhere.

# Literature Survey

## 2.1 Existing Problem

As the farmers are the face of the nation, in the agriculture system there are lots of tasks to be done as to grow a crop, for all this processes the farmers has to see all the updates of the weather forecasting, temperature, soil moisture etc. In order to get all this updates farmer has to go on field and should monitor it physically. To have control on their crops farmers has to get the time to time updates. It is difficult for the farmer to do this for each and every crop.

## 2.2 Proposed Solution

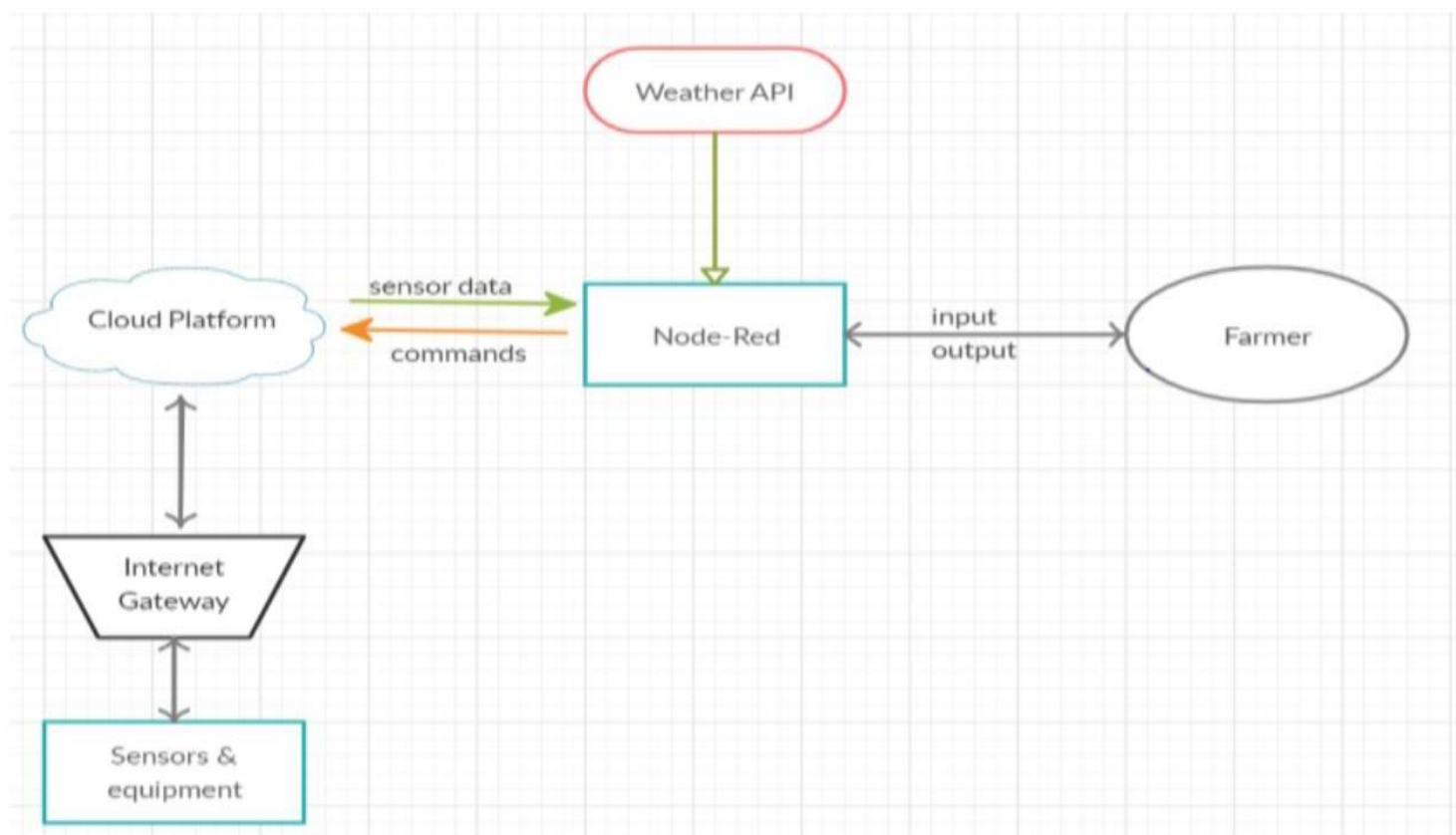
For the above listed problems only there is only one solution that is technology, as today we have latest technology called as IOT (Internet of things) the solution. As the project name suggest Smart agriculture system based on IOT, in this project farmers will get live update of the weather , wind humidity and also soil moisture. And also farmers have a flexibility to control the motor. And most interesting part is he can control or monitor it from anywhere as it is based on IBM cloud.



## Theoretical Analysis

### 3.1 Block Diagram

The block diagram of the proposed project is shown below



## 3.2 Software Design

### 1) IBM Watson IOT platform

It is basically cloud based platform where the data from the different devices and sensors can be stored.

IBM Watson IoT Platform

si05202000482@smartinternz.com  
ID: w2dens

Browse Action Device Types Interfaces

Add Device +

## Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator ☒

<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
> <input type="checkbox"/>	774411	Disconnected	RASBERRY	Device	May 28, 2020 11:07 AM		si05202000482@smartinternz.com
> <input type="checkbox"/>	112233445566	Disconnected	Motor	Device	Jun 12, 2020 1:33 PM		si05202000482@smartinternz.com

Items per page 50 | 1-2 of 2 items

1 of 1 page < 1 >

0 Simulations running

nces

### 2) Node Red

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.

The Node Red tool can be installed locally or can be used in IBM website. To install it locally the steps can be referred from the node red website.

# Node-RED

Low-code programming for event-driven applications

Latest version: v1.0.6 (npm)

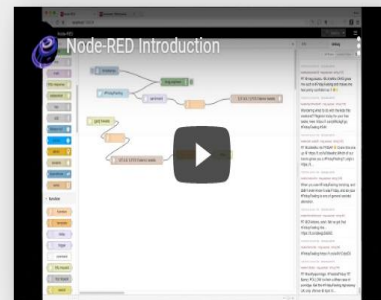
Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

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.

[Features](#)

[Get Started](#)

[Community](#)



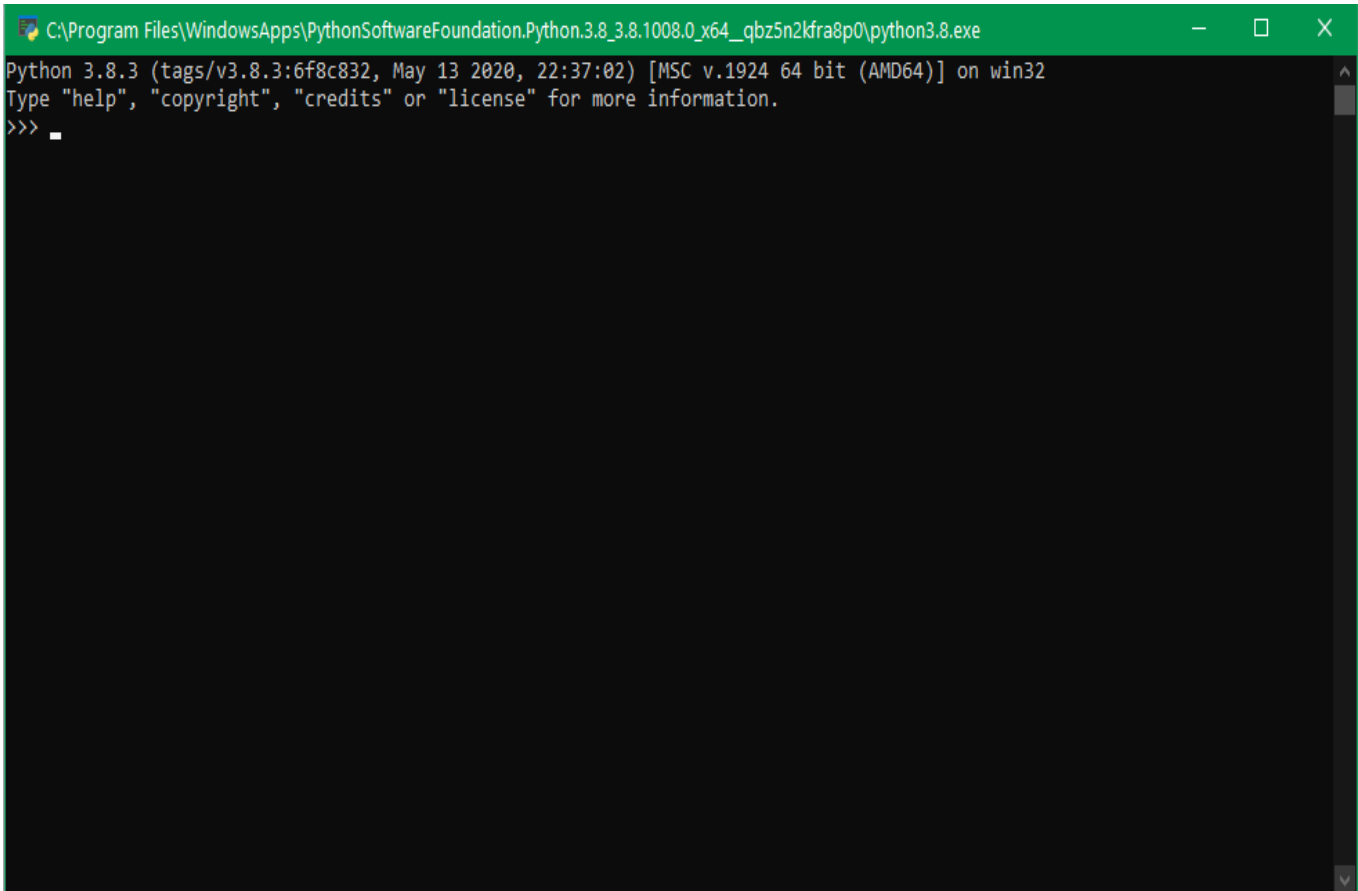
## Browser-based flow editing

Node-RED provides a browser-based flow editor that makes it

A detailed screenshot of the Node-RED web interface. The interface features a top navigation bar with the 'Node-RED' logo and a 'Deploy' button. Below the navigation bar, there are tabs for 'Flow 1', 'Flow 3', 'Smart bridge IOT', and 'Flow 2'. The main workspace is a large grid where flows are edited. On the left, a 'filter nodes' search bar is present, followed by a palette of nodes categorized into 'output' (timerswitch, OpenWhisk, ibmiot out), 'sequence' (split, join, sort, batch), 'parser', and 'storage' (cloudant in, tail). On the right, a 'dashboard' sidebar displays a hierarchical menu of widgets, including 'Student' (Management, Weather, Data, Welcome, Enter Your Name, Weather Details, Welcome To The Smart Agr, Temperature, Humidity, Pressure, Tabular), 'Sensor Details' (Motor Operator, Motor Status, text, Sensor Values, Graph), and 'Graph'. The bottom of the interface includes a status bar with icons for help, zoom, and other utility functions.

### 3) Python IDLE

This is an open resource platform, as this software is used to connect the motors practically.




### 4) Open Weather API

This is online service which gives the real values of the weather of current city, which includes parameters such as Weather description, wind speed temperature etc. It generates a API key which can be entered in Node red tool.

#### Steps to configure :


1. Create account in Open Weather
2. Find the name of your city by searching
3. Create API key to your account
4. Replace “city name” and “your api key” with your city


Weather in your city
Get Started
API
Pricing
Maps
Partners
Blog
Marketplace
Sign in
Support

# We Deliver 2 Billion Forecasts Per Day

2,000 new subscribers a day | 2,400,000 customers | 20+ weather APIs

Weather in Mysore, IN


26.0 °C

Broken clouds

Obtained at 18:33, 18 Jun 2020 Wrong data?

Local time	18:33, 18 Jun 2020
Wind	Moderate breeze 7.2 m/s West ( 260 )
Cloudiness	Broken clouds
Pressure	1011 hpa
Humidity	74 %
Sunrise	05:59
Sunset	18:49
Geo coords	[ 12.31, 76.65 ]

°C / °F

Weather and forecasts in Mysore, IN

Main

Daily

Hourly

Minutely

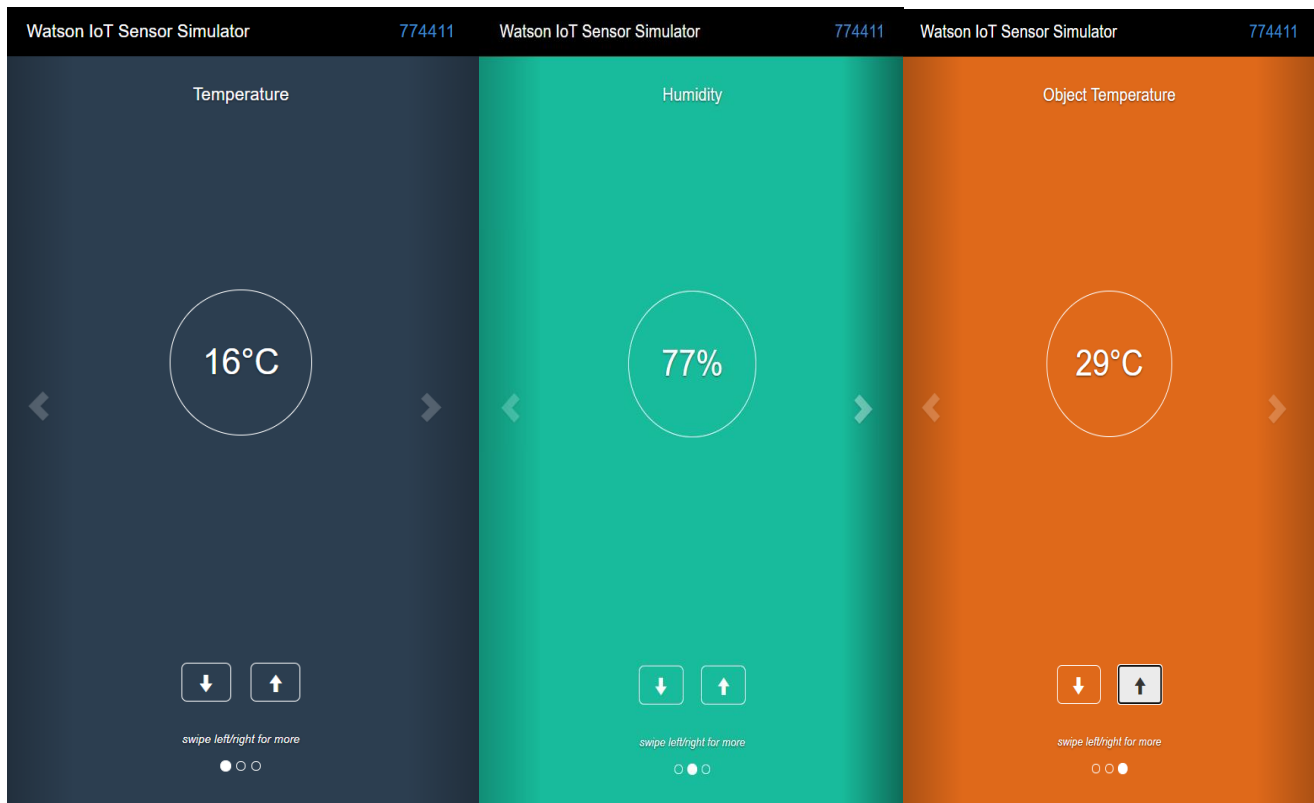


### 3.3) Hardware Design

The project requires temperature sensor, Humidity sensor and Soil moisture sensor these sensors are placed on the fields so that it can give the live updates of the land. The actual hardware sensor is show below for the project we are using stimulator.







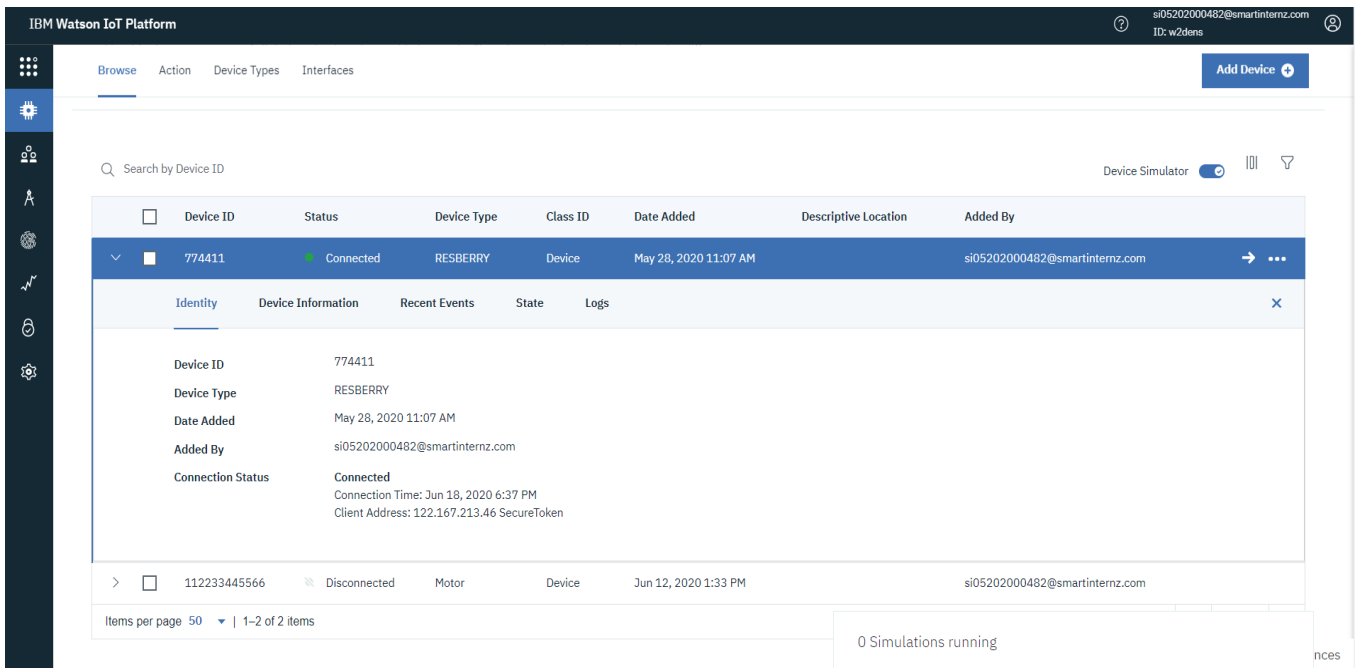
## Experimental Investigation

### 4.1) Connecting stimulator the IBM Watson cloud.

As the first task of the project is to connect the stimulator to the IBM cloud. To connect the stimulator a device need to be added ,as it generates the device credentials. That credential can be added to the IBM Watson IOT stimulator.

#### Device Credentials-1

Organization ID	: w2dens
Device Type	: RESBERRY
Device ID	: 774411
Authentication Token	: 987654321



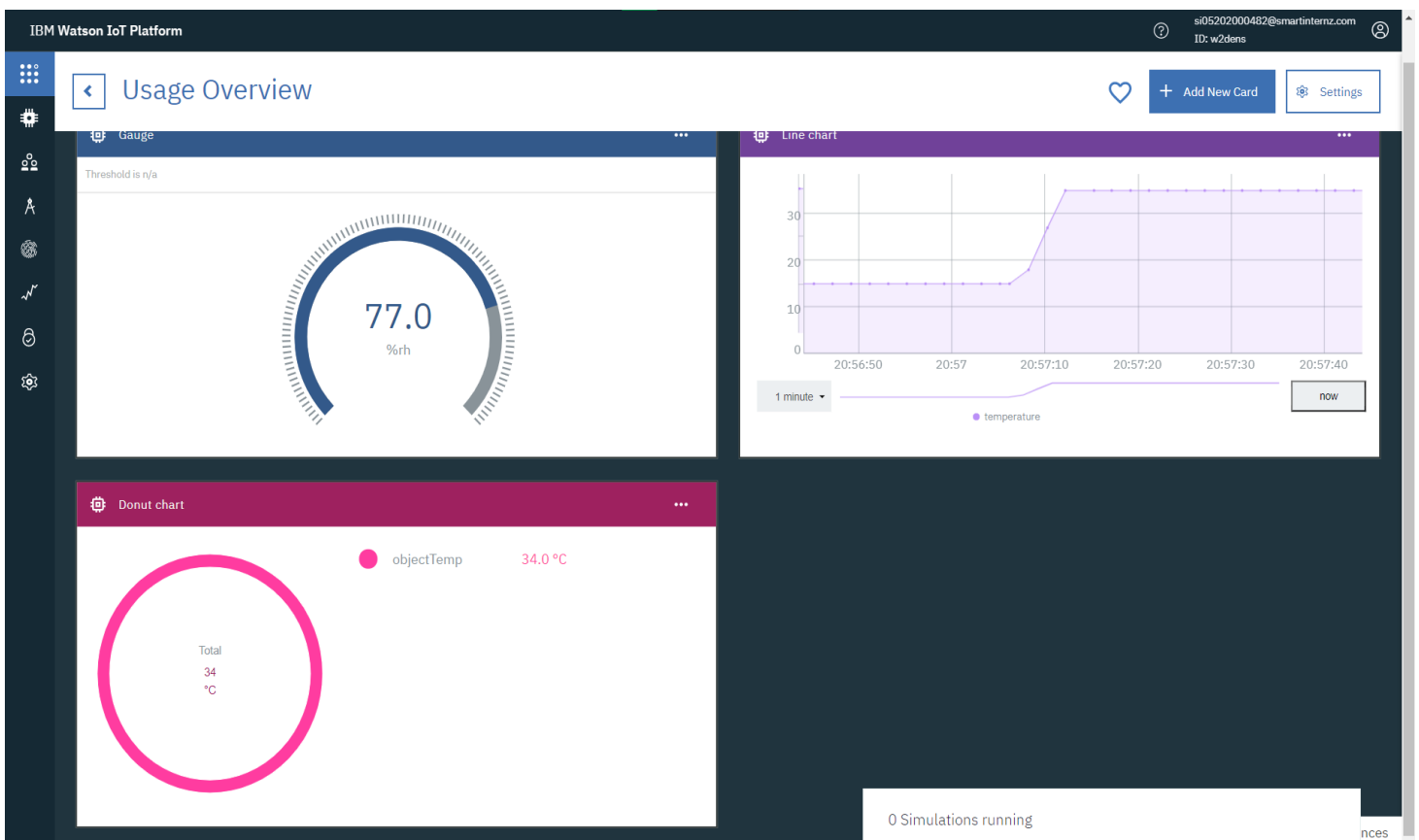
In order to obtain the API Key and Authentication token for the node red, one has to click the option called Apps management in IBM cloud website.

## API credentials

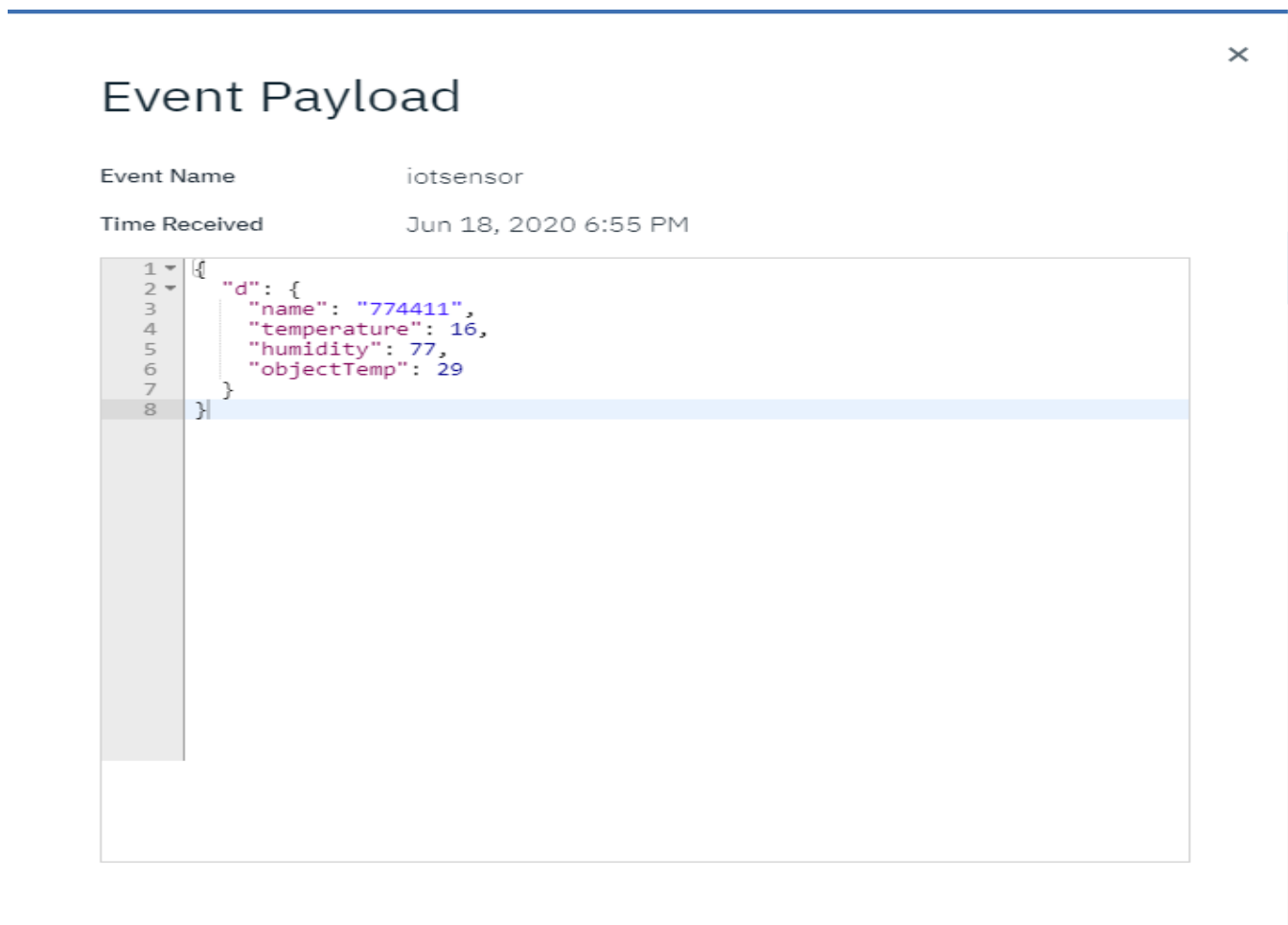
**API key** : a-w2dens-zo7kz07d03

**Authentication token** : ZpS@6Pbh(\*\_bxVZ2S6

We can also see the data in the form of line chart in IBM boards. Below is the picture of one sample line chart graph where we can see the fluctuation.



The IOT stimulator generates the values, as the live sensor in the field so that the data can be sent to IBM Watson IOT cloud. The values are in Jason format as shown below.



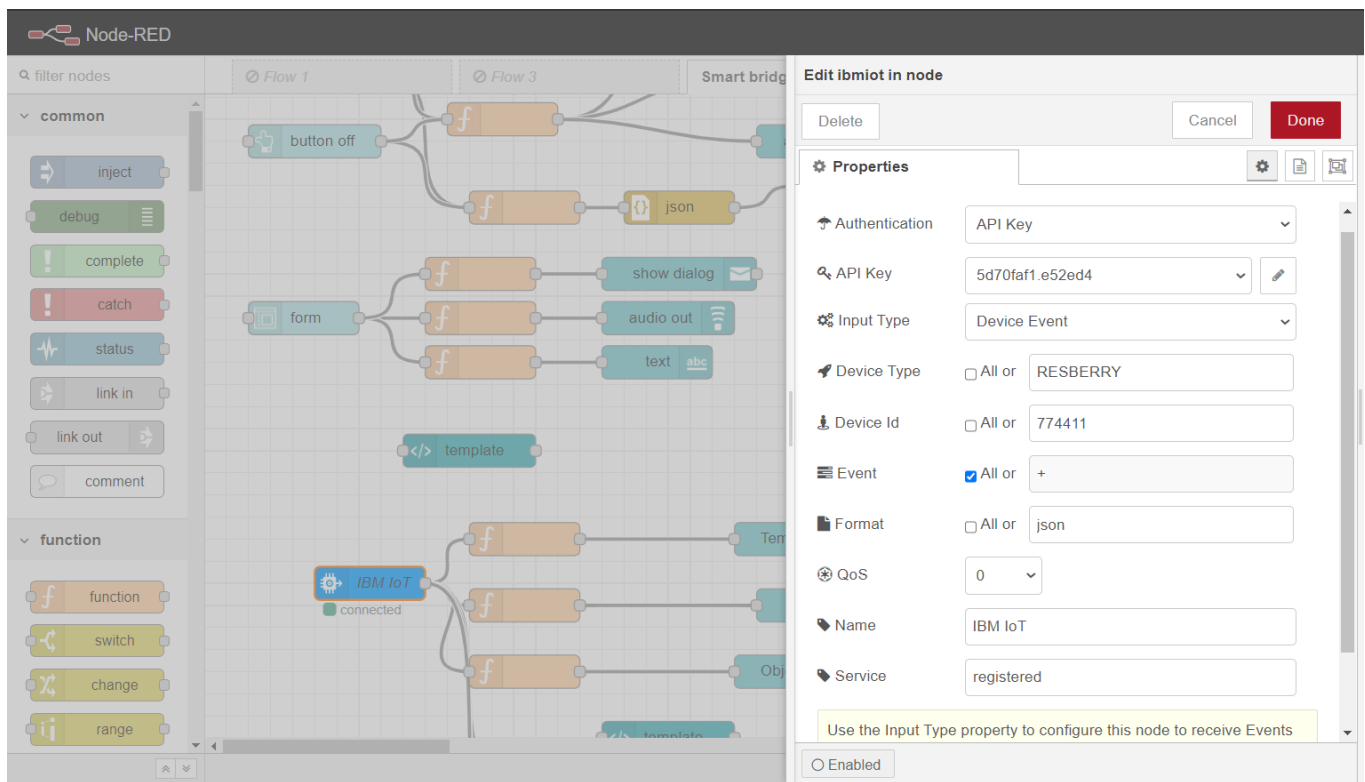
## 4.2) Configuration Of Node Red to get the data from IBM Cloud

As the node red is the main tool for this project, a certain IBM IOT library has to be installed. After this installation we can see an input node called IBM IOT. Then in order to configure the node a certain device credentials has to entered, that can be generated in IBM Watson Cloud platform.

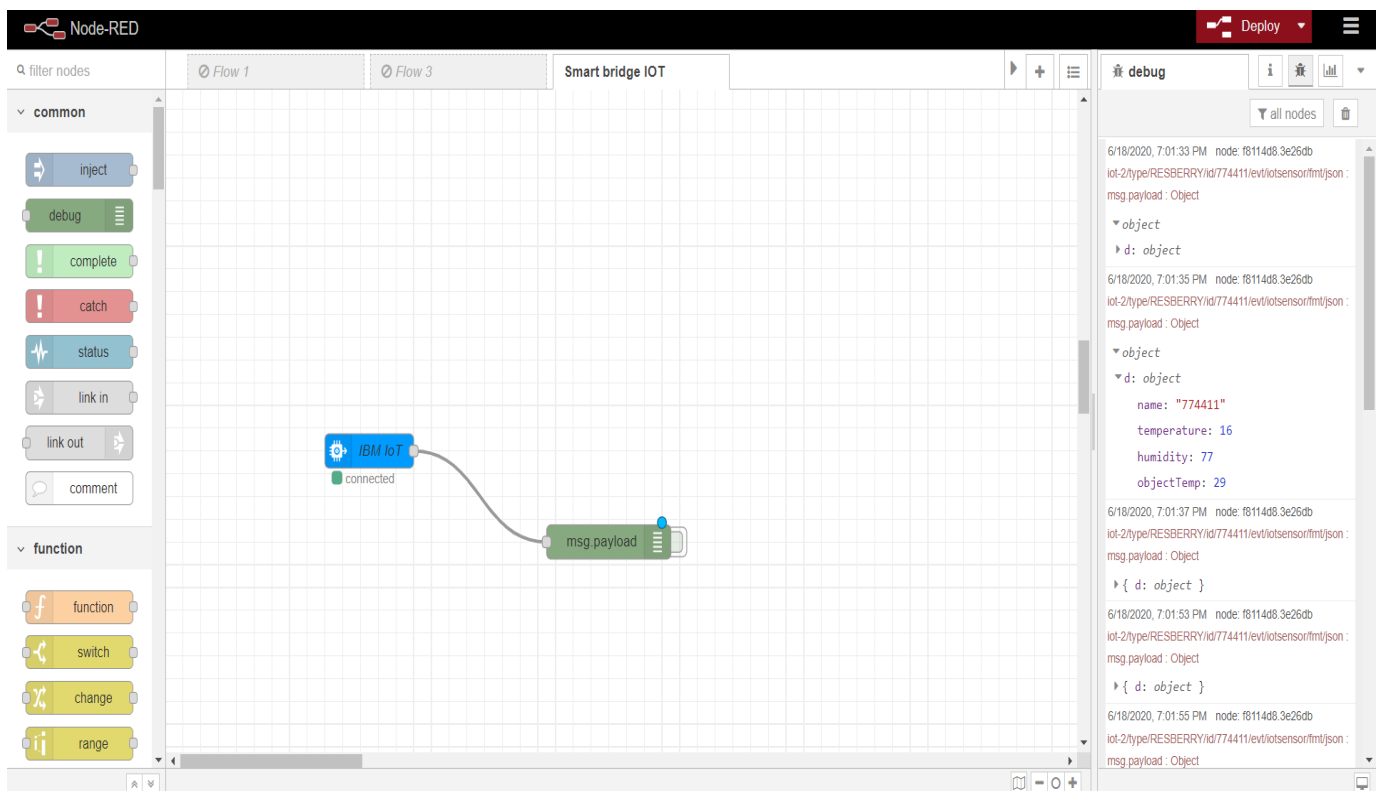
A certain program has to be written in function node, as the IBM cloud send the data in the Jason format so that a certain value of the sensors can be obtained.

**The program to obtain value for temperature is shown below.**

```
global.set('temperature',msg.payload.d.temperature)  
  
return {payload:msg.payload.d.temperature,msg}
```



Once it is connected Node-Red receives data from the device. The values can be seen in debug session in node red by using debug node and can be programmed using function node to display the data in Web UI.



### 4.3) Configuration Of Node Red to get the data from Open Weather.

The Node-Red also receive data from the Open Weather API by HTTP GET request, an inject node has to be added so that it triggers the node. In the HTTP node, the API key which is generated by open weather website.

The screenshot displays the Node-RED web interface. On the left, the 'common' and 'function' node palettes are visible. The main workspace shows a flow starting with an 'inject' node, followed by a 'timestamp' node, and then a 'weather' node. The 'weather' node is connected to a 'msg' node, which is then connected to a 'template' node. The 'Edit http request node' dialog is open, showing the following configuration:

- Method: GET
- URL: `api.openweathermap.org/data/2.5/weather?q=My:`
- Append msg payload as query string parameters: ☐
- Enable secure (SSL/TLS) connection: ☐
- Use authentication: ☐
- Enable connection keep-alive: ☐
- Use proxy: ☐
- Return: a parsed JSON object
- Name: weather

A tip at the bottom of the dialog states: "Tip: If the JSON parse fails the fetched string is returned as-is." The 'debug' console on the right shows the output of the 'weather' node, displaying a JSON object with weather data for a specific location.

The screenshot shows the OpenWeather API documentation page. The header includes the OpenWeather logo and navigation links: Weather in your city, Get Started, API, Pricing, Maps, Partners, Blog, Marketplace, Sign in, and Support. A warning box states: "Please remember that all Examples of API calls that listed on this page are just samples and do not have any connection to the real API service!"

### By city name

Description:

You can call by city name or city name, state code and country code. API responds with a list of weather parameters that match a search request.

API call:

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

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

```
api.openweathermap.org/data/2.5/weather?q={city name},{state code},{country code}&appid={your api key}
```

- JSON
- XML
- List of condition codes
- Min/max temperature in current weather API and forecast API
- Other features
  - Format
  - Units format
  - Multilingual support
  - Call back function for JavaScript code

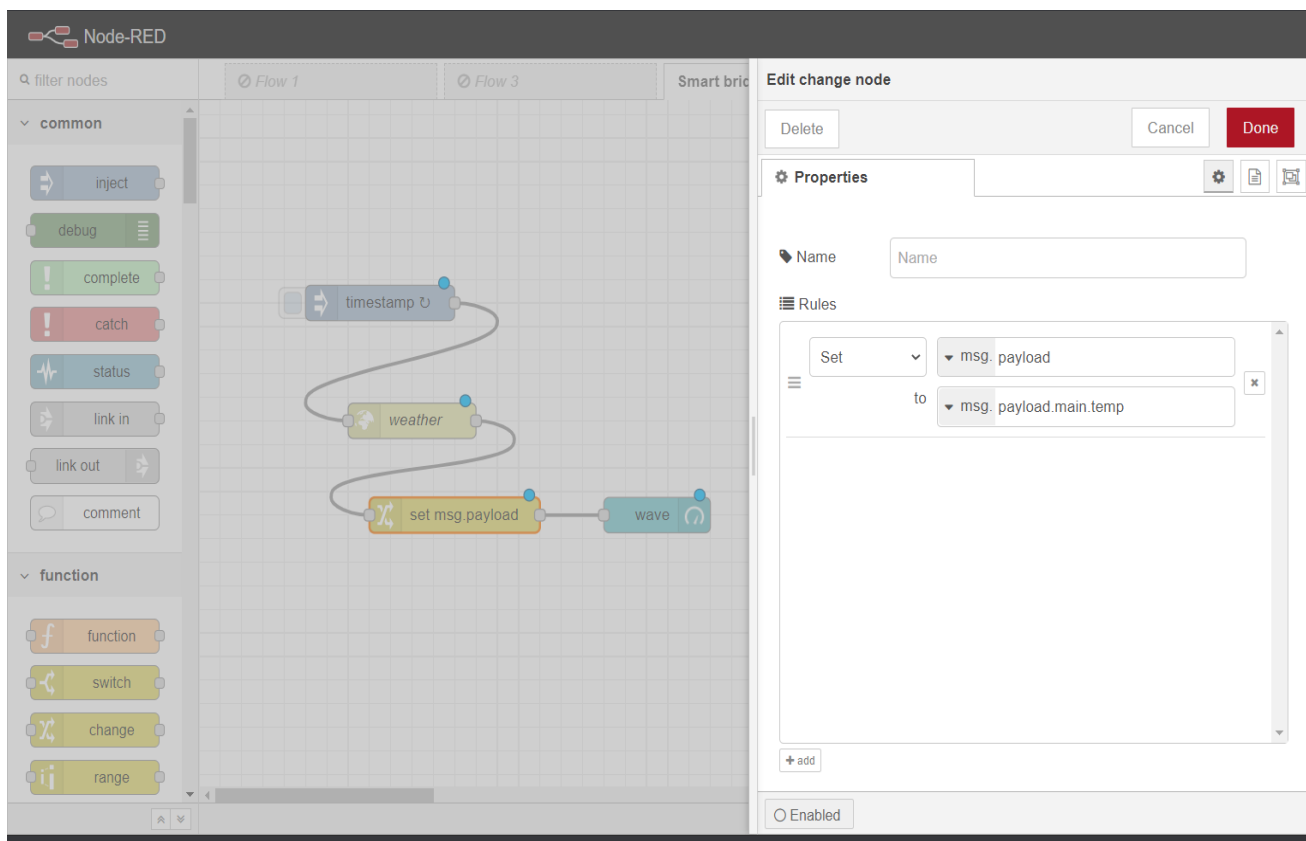
By adding the city name we can get the live weather data of current city. The values are in Jason format.

The Link is given below

[api.openweathermap.org/data/2.5/weather?q=Mysore,IN&appid=01c2b13013c958c11081bf7a6cff7122&units=metric](https://api.openweathermap.org/data/2.5/weather?q=Mysore,IN&appid=01c2b13013c958c11081bf7a6cff7122&units=metric)

```
{
  "coord": {
    "lon": 76.65,
    "lat": 12.31
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 25,
    "feels_like": 24.79,
    "temp_min": 25,
    "temp_max": 25,
    "pressure": 1012,
    "humidity": 78,
    "visibility": 6000,
    "wind": {
      "speed": 6.2,
      "deg": 270
    },
    "clouds": {
      "all": 75
    },
    "dt": 1592487596,
    "sys": {
      "type": 1,
      "id": 9212,
      "country": "IN",
      "sunrise": 1592440150,
      "sunset": 1592486399,
      "timezone": 19800,
      "id": 1262321,
      "name": "Mysore",
      "cod": 200
    }
  }
}
```

In the node red the use of Set node can be done to get the data and to send it to UI.



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

We have to use ibmiot out node to send data from Node-Red to IBM Watson device. So, after adding it to the flow we need to configure it with 2nd device credentials of our IBM Watson cloud.

## Device Credentials-2

**Organization ID** : **w2dens**

**Device Type** : **Motor**

**Device ID** : **112233445566**

**Authentication Token** : **112233445566**

The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows the user 'si05202000482@smartinternz.com' with ID 'w2dens'. The main navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. Below this, a table lists devices. The selected device is '112233445566', which is a 'Motor' type, 'Device' class, added on 'Jun 12, 2020 1:33 PM' by 'si05202000482@smartinternz.com'. The device status is 'Connected'. A detailed view of the device is shown below the table, including fields for 'Device ID', 'Device Type', 'Date Added', 'Added By', and 'Connection Status'. The 'Connection Status' is 'Connected' with a connection time of 'Jun 18, 2020 7:18 PM' and a client address of '122.167.213.46 SecureToken'. The bottom of the interface shows '0 Simulations running'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
774411	Connected	RESBERRY	Device	May 28, 2020 11:07 AM		si05202000482@smartinternz.com
112233445566	Connected	Motor	Device	Jun 12, 2020 1:33 PM		si05202000482@smartinternz.com

Identity	Device Information	Recent Events	State	Logs
Device ID	112233445566			
Device Type	Motor			
Date Added	Jun 12, 2020 1:33 PM			
Added By	si05202000482@smartinternz.com			
Connection Status	Connected Connection Time: Jun 18, 2020 7:18 PM Client Address: 122.167.213.46 SecureToken			

Output node that can be used with Watson IoT Platform to send a commands to a device or send an event on behalf of a device.

The following message properties take precedence and override the values configured in the node:

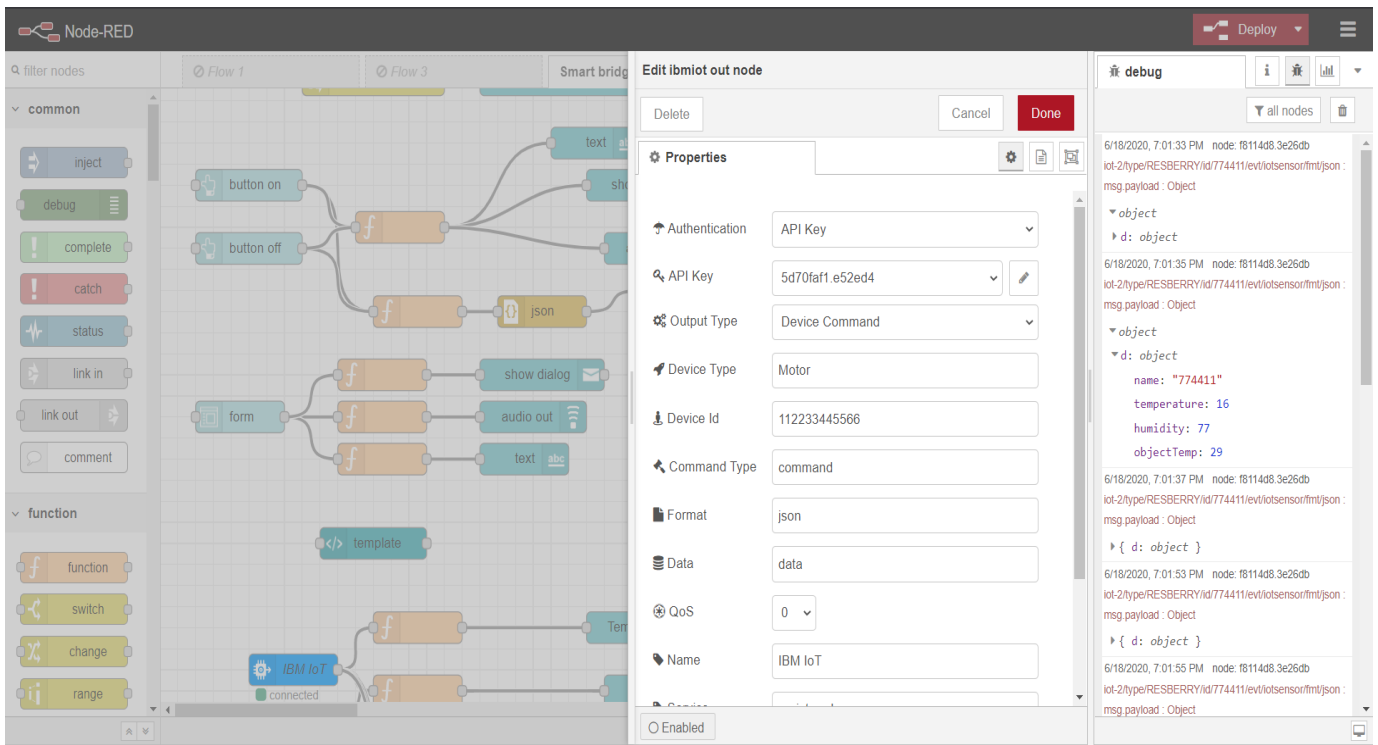
**msg.deviceId** overrides the value of "Device Id".

**msg.deviceType** overrides the value of "Device Type".

**msg.event Or CommandType** overrides the value of "Event Type" or "Command Type".

**msg.payload** overrides the value of "Data".

The configuration of the node red ibmiot out node is shown below.



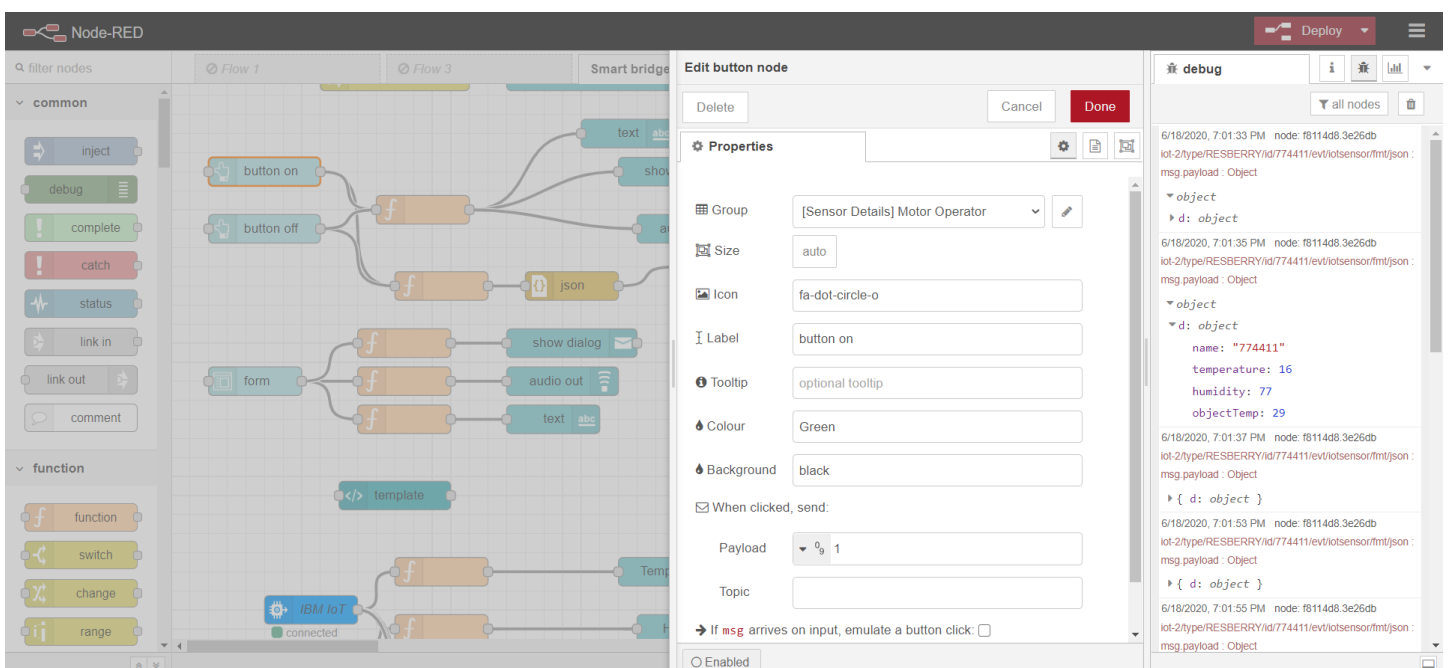
In UI dashboard when we click the motor on/off button we will receive this commands in the python code.

We are using button mode for the motor to On and Off the configuration setting for the button node is shown below.

In UI dashboard when we click the motor on/off button we will receive this commands in the python code that we will see that in lower sections.

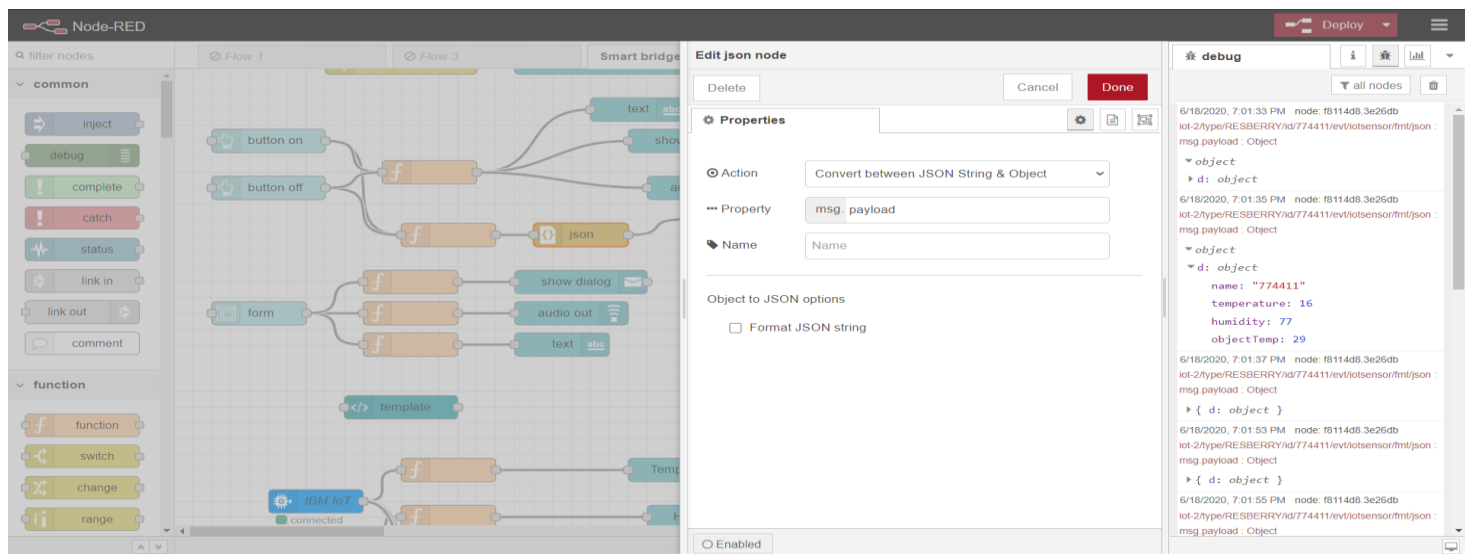
If we receive the commands thus our buttons are working successfully.

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





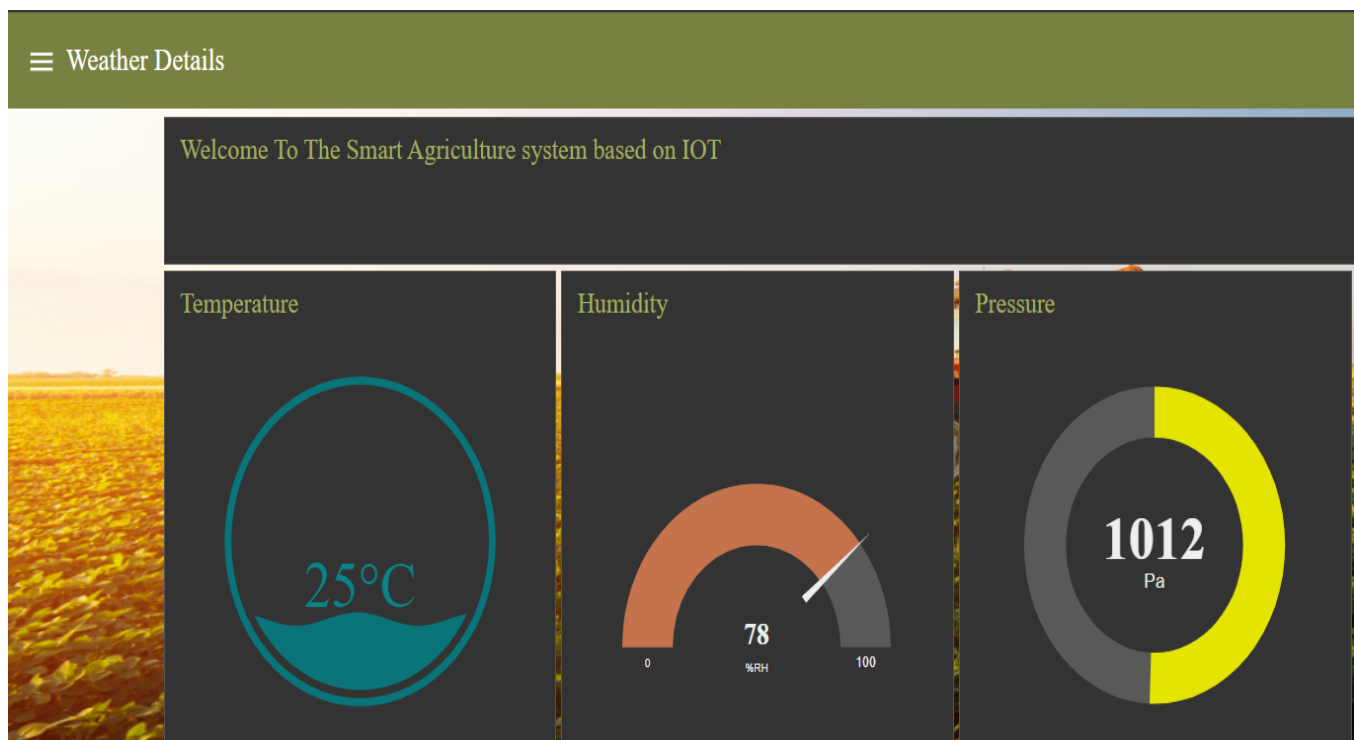
Then we are using Jason node to Convert between a JSON string and its JavaScript object to pass the command to ibmiot out node.



#### 4.5) Building User Interface using Node Red

In order to display the data of the different sensors, to monitor the weather data of the current city and also to control the motor the common platform is UI.As we have different options in the dashboard section in the pallet like Gauges, Text, Form, Notification Audio out etc.

For my UI design I have used gauge node to display the humidity, donut node for pressure ,wave node for the temperature.



And also I have included the tabular column for the weather details like the cloud description, wind speed, place and also the name of the country.

Also the sensor details can be viewed in tab, as I have included the button in motor operator tab that operates the motor, basically sends the command to the python platform and also the status of the motor can be seen in the status tab, the values of the live sensors of the land can be seen in sensors value tab.

For the comparison of the sensors data I have also added the graph.



#### 4.6) Receiving commands from IBM cloud using Python program

The second device credentials has to be added in the python code so that the motor can be controlled using the commands.

**The python code is shown below.**

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device
```

#Provide your IBM Watson Device Credentials

```
organization = "w2dens" #replace the ORG ID
deviceType = "Motor"#replace the Device type wi
deviceId = "112233445566"#replace Device ID
authMethod = "token"
authToken = "112233445566" #Replace the authtoken
```

```
def myCommandCallback(cmd): # function for Callback
```

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

```
    if cmd.data['command']=='MotorON':
```

```
        print("MOTOR ON IS RECEIVED")
```

```
    elif cmd.data['command']=='MotorOFF':
```

```
        print("MOTOR OFF IS RECEIVED")
```

```
    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:
```

```
    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

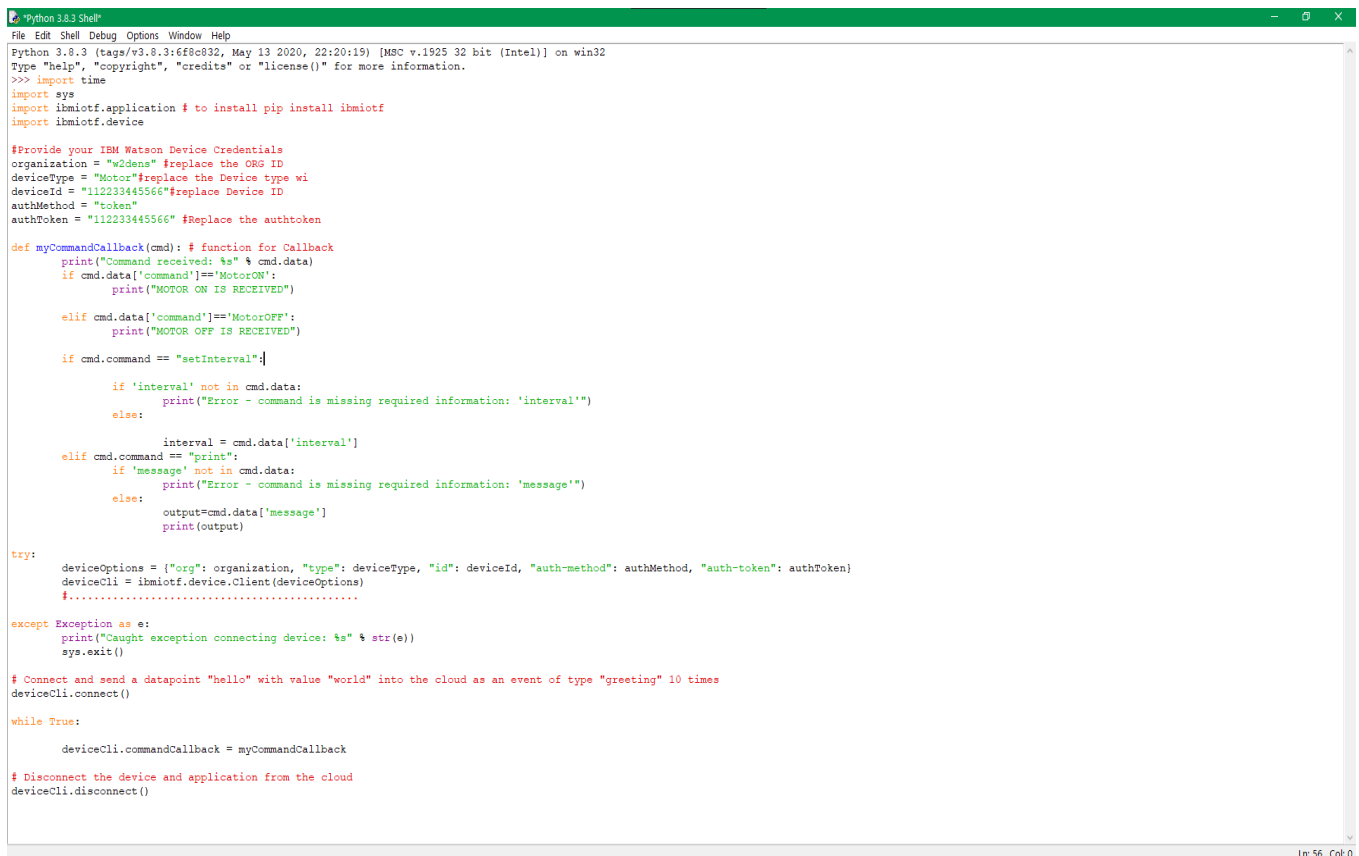
```
deviceCli.connect()
```

while True:

```
deviceCli.commandCallback = myCommandCallback
```

# Disconnect the device and application from the cloud

```
deviceCli.disconnect()
```



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "w2dens" #replace the ORG ID
deviceType = "Motor" #replace the Device type wi
deviceId = "112233445566" #replace Device ID
authMethod = "token"
authToken = "112233445566" #Replace the authToken

def myCommandCallback(cmd): # function for Callback
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='MotorON':
        print("MOTOR ON IS RECEIVED")

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

    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:
    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()

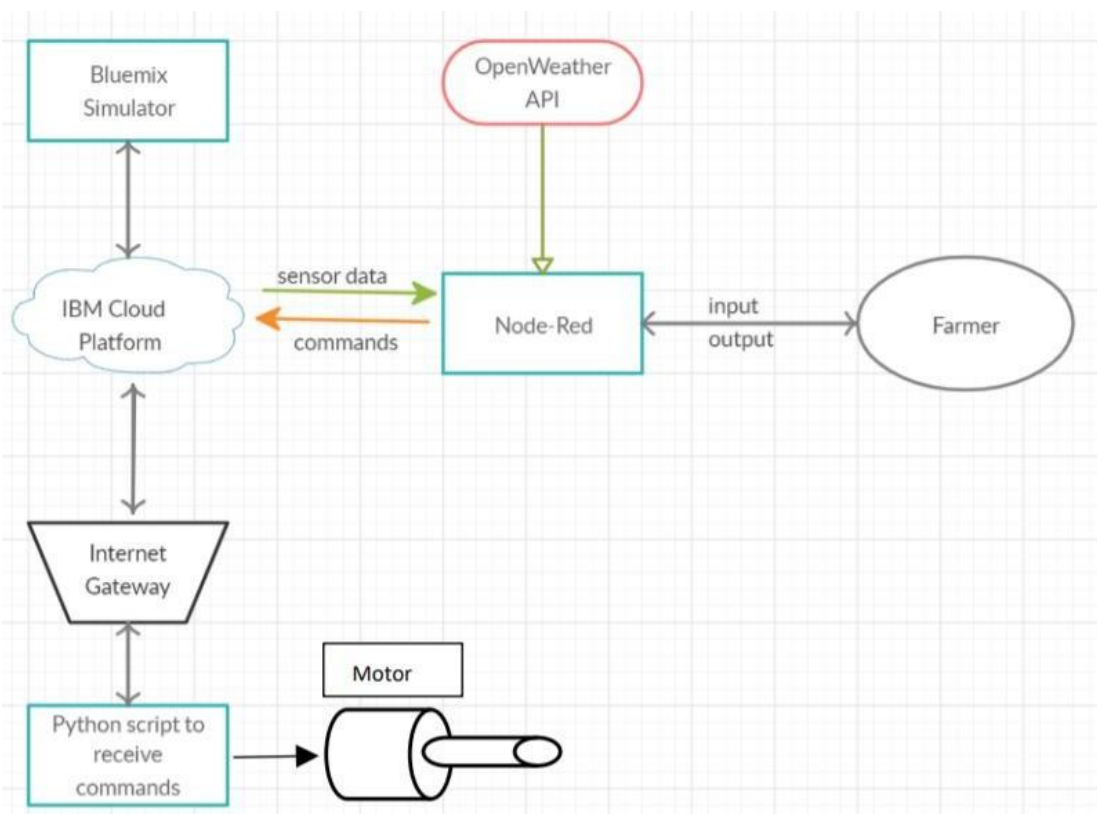
# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect()

while True:

    deviceCli.commandCallback = myCommandCallback

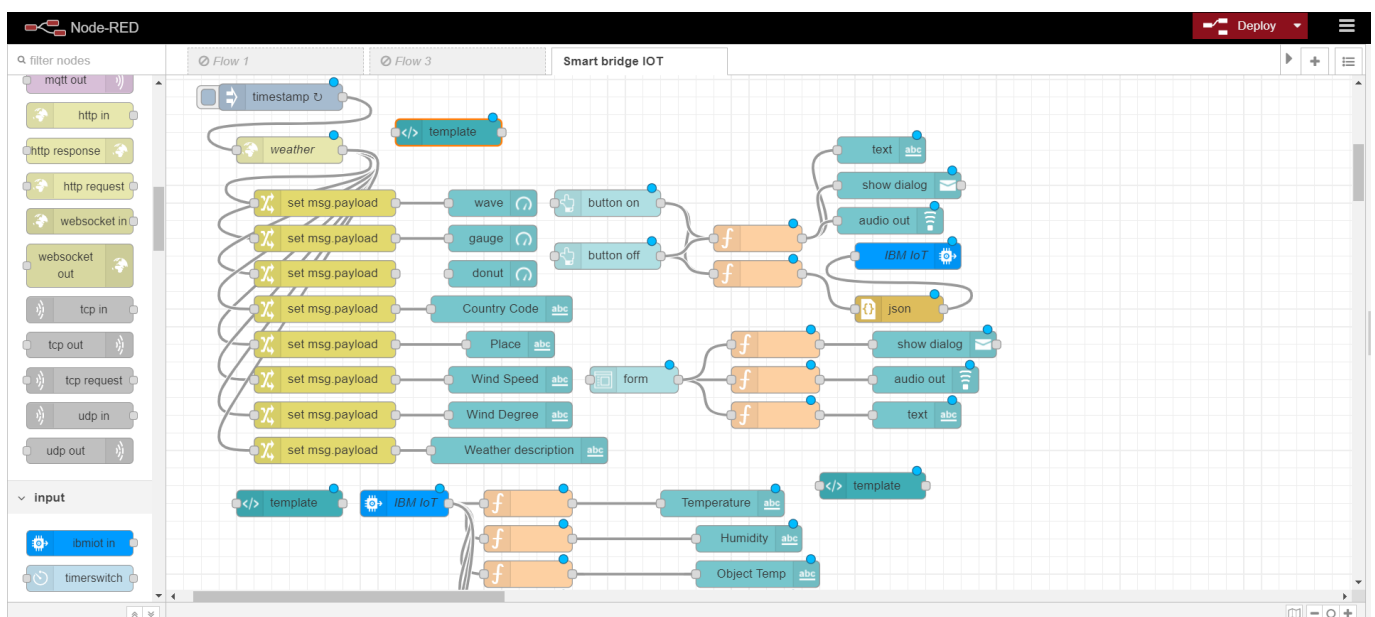
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

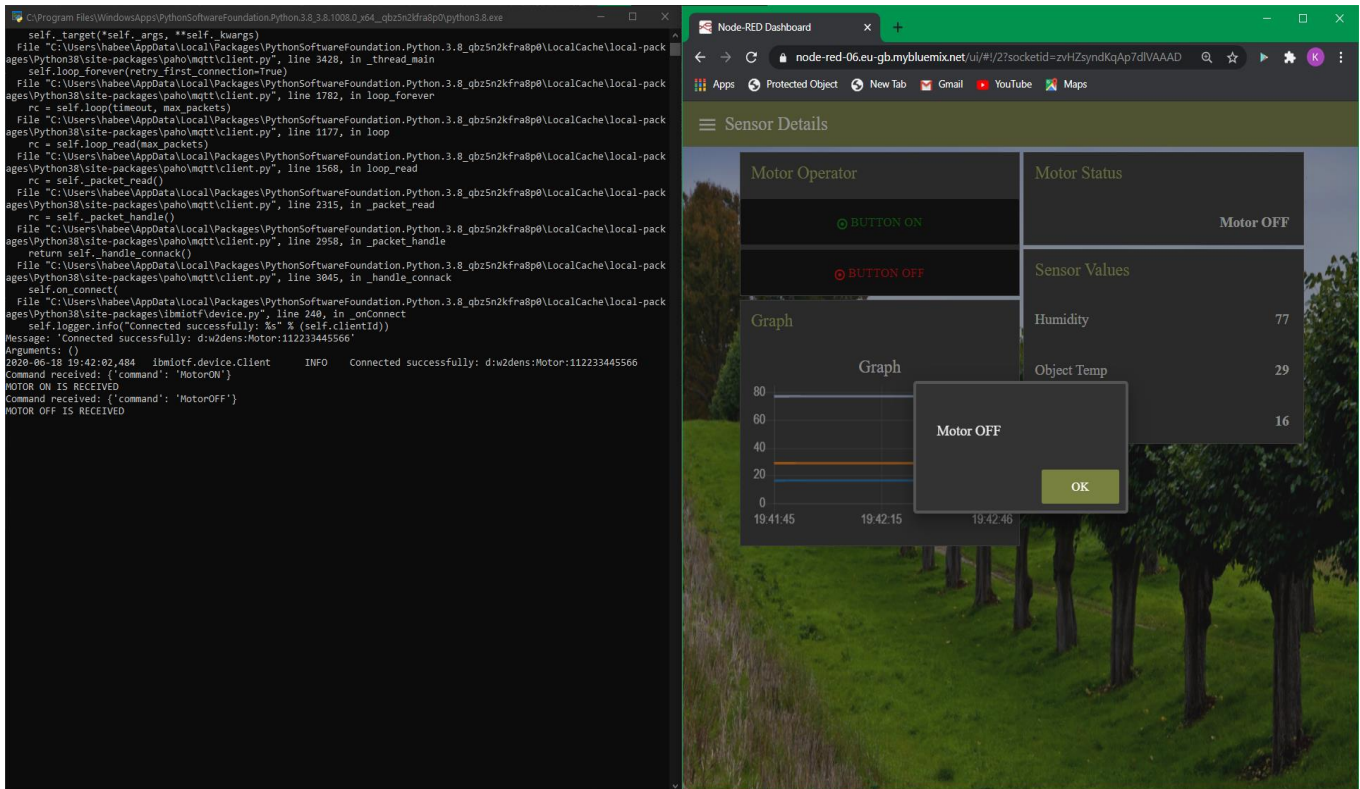
## 5 ) Flow Chart



## 6) Result

As a result for the above project , we can control the motor from the web UI, the commands are displayed in the python platform and all the live data of the sensors and the weather of the current city can be seen in UI.The below picture shows the output.





## 7) Advantages And Disadvantages

### Advantages :

1. Farms can be monitored and controlled remotely.
2. Improves livestock monitoring and management.
3. Less labour cost .
4. Better standards of living.
5. Very helpful in knowing weather conditions.
6. Increases the quality of production and water conservation.
7. Weather prediction can be done easily

### Disadvantages :

1. Lack of internet/connectivity issues.
2. Added cost of internet and internet gateway infrastructure.
3. Loss of privacy and security.
4. Less compatibility and more complexity.
5. Farmers are not used to these type of high-end technologies.

## 8) Application

1. The main applications are



2. Monitoring of the soil moisture in real time
3. Regulating the water supply and controlling usage of water
4. Remotely controllable of motor
5. Monitoring of weather conditions.

## 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.

And Node red based Web UI has been designed successful, where it receives the data from IBM cloud and sends the command to python using IBM cloud.

## 10) Future scope

Future development will be focused more on increasing sensors on this system to fetch more live data regard to pest control, food production, etc also by integrating with the GPS to enhance the Agriculture IoT technology to full fill Agriculture Precision ready product. We can use it as a home automation controller. We can remotely operate or perform the jobs. Also combining with solar panels to conserve power. So the entire system is going to be eco-friendly.

## 11 Bibliography

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

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

Open Weather: <https://openweathermap.org/>

Python code reference: <https://github.com/rachuriharish23/ibmsubscribe>