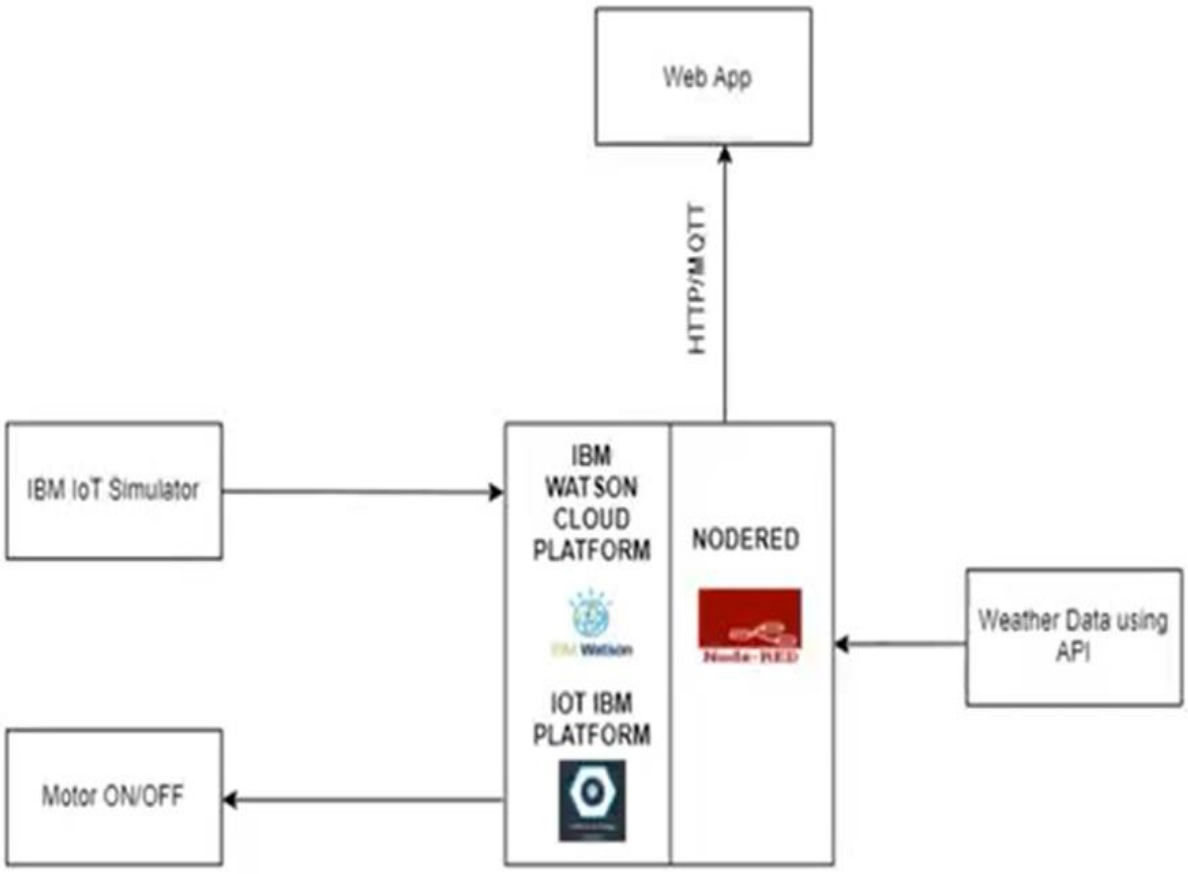
SMART AGRICULTURE SYSTEM BASED ON IOT

Aim and Scope:

Smart Agriculture System based on Iot can monitor soil moisture and climatic conditions to grow and yield a good crop. The farmer can also get the real-time weather forecasting data by using external platforms like Open Weather API. Farmer will be provided a mobile app using which he can monitor the temperature, humidity and soil moisture parameters along with weather forecasting details. Based on all the parameter, farmer can water his crop by controlling the motor using the mobile application. Thus even if the farmer is not present near his crop he can water his crop by controlling the motors using the application from anywhere.

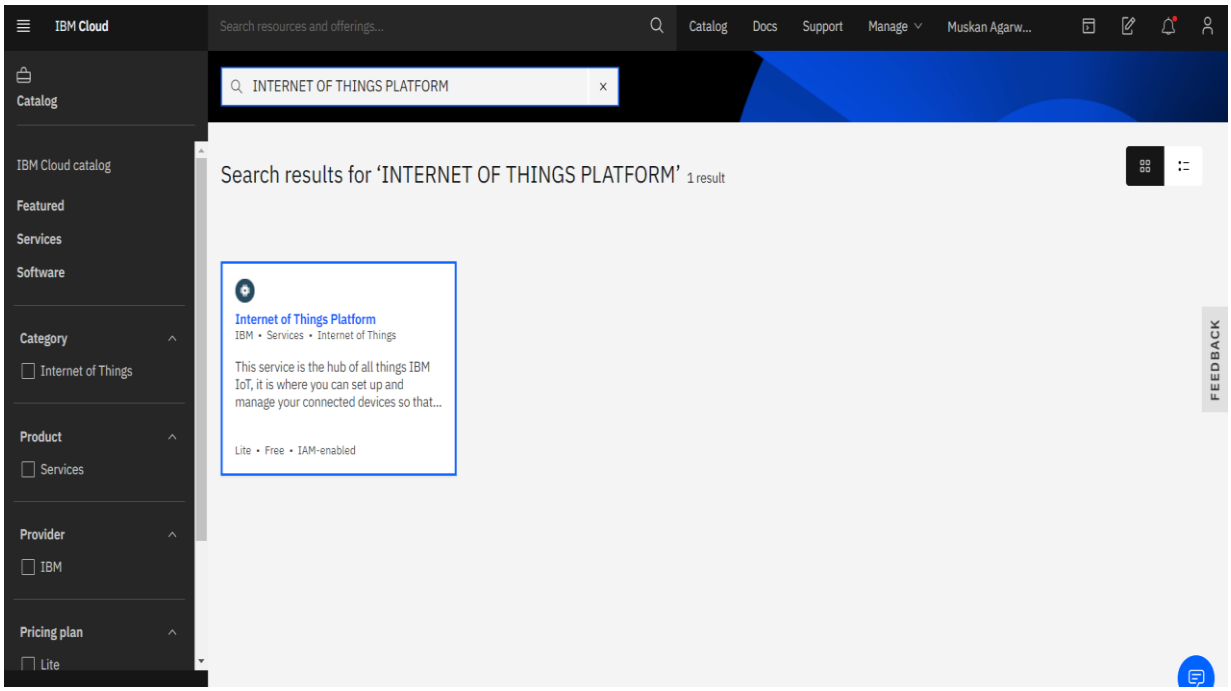
The Project Flow can be seen from the following diagram:



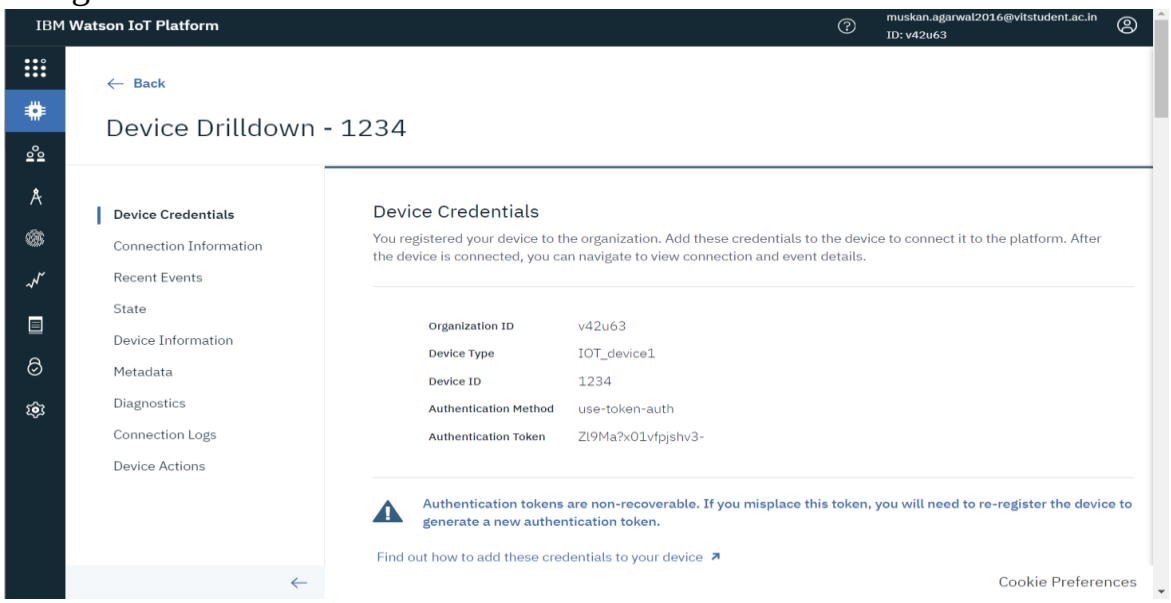
Getting Sensor Inputs into IBM Cloud:

First task to build the web app is to get the sensor data in the cloud. I am using IBM cloud for this. We need an IBM account for the same. Steps to take sensor data in the cloud:

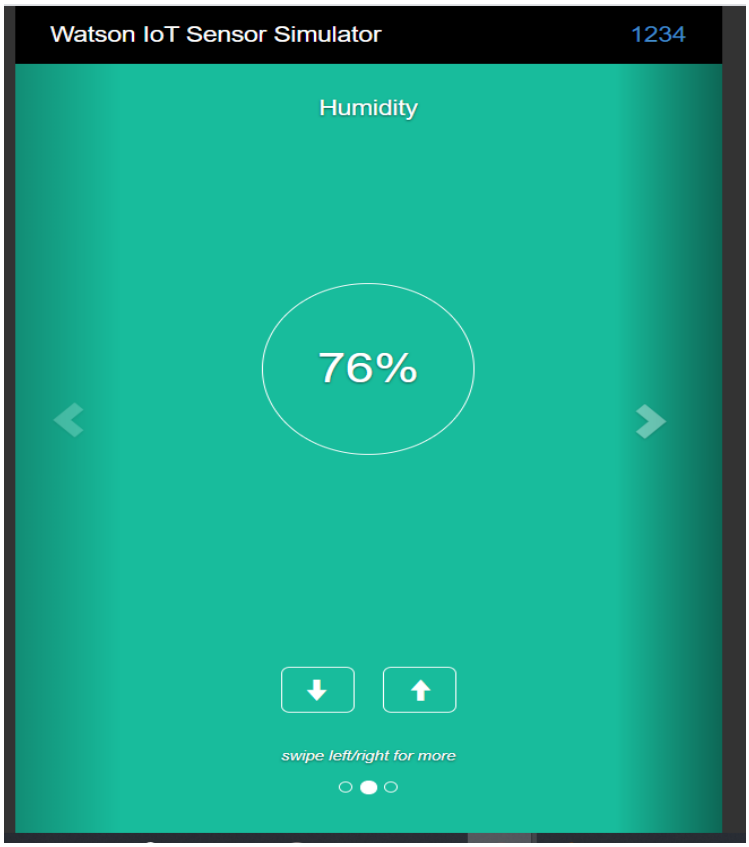
1. Sign up for IBM Academic Initiative Account using this link.
2. After this Sign up for IBM Cloud using link .
3. Go to IBM Watson IOT Platform by searching IOT platform in the catalogue in IBM Cloud



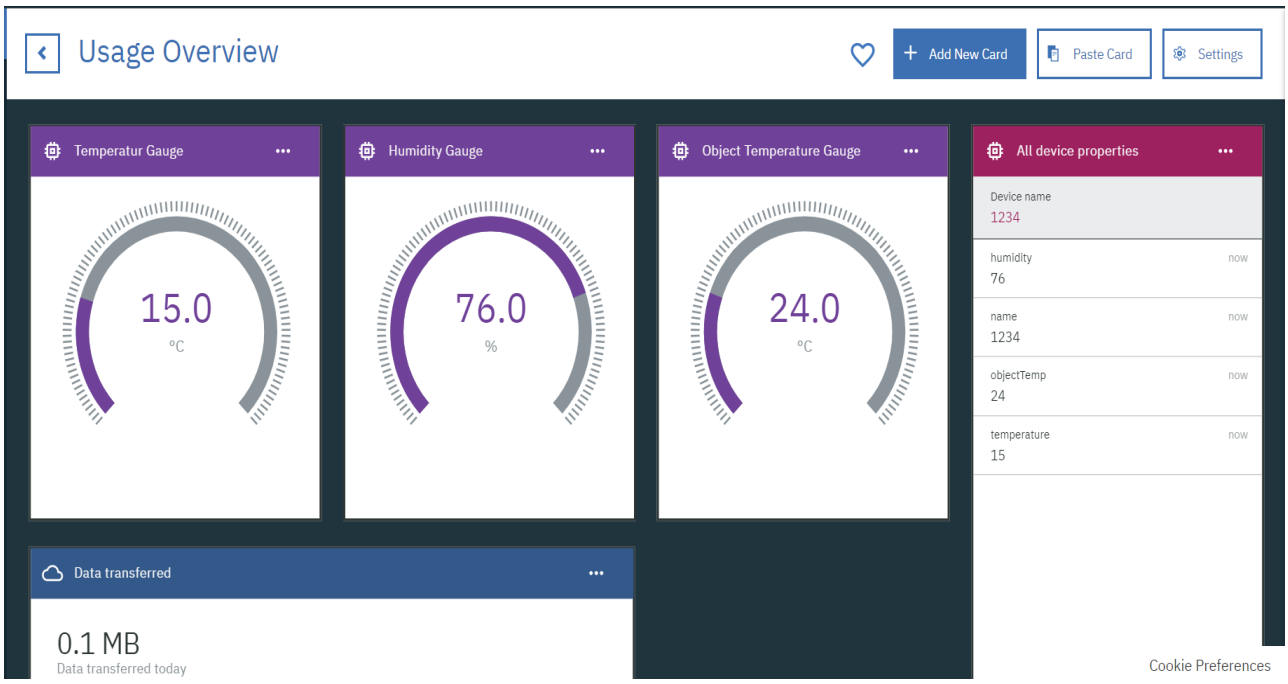
4. Go to the IOT Platform and now we will create a device here after which we will get the credentials for IOT simulator.



5. You will get Device credentials save them in a notepad so that we connect to IOT simulator. Go to [link] for IOT simulator. The following screen appears once your simulator get's connected.



6. Now in the cloud we can create cards to view the simulator data.



Node-Red

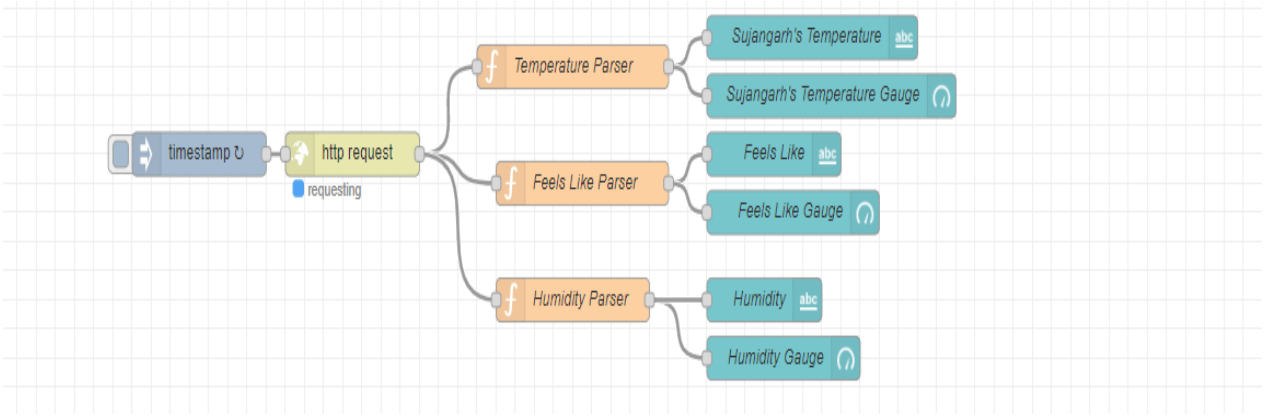
Now once we got the data in the cloud, we will use node red to get the data in a web app. To install node-red in windows follow this link. After this we would need to externally install IBM iot node in node red using the below code.

Node-red-contrib-scx-ibmiotapp

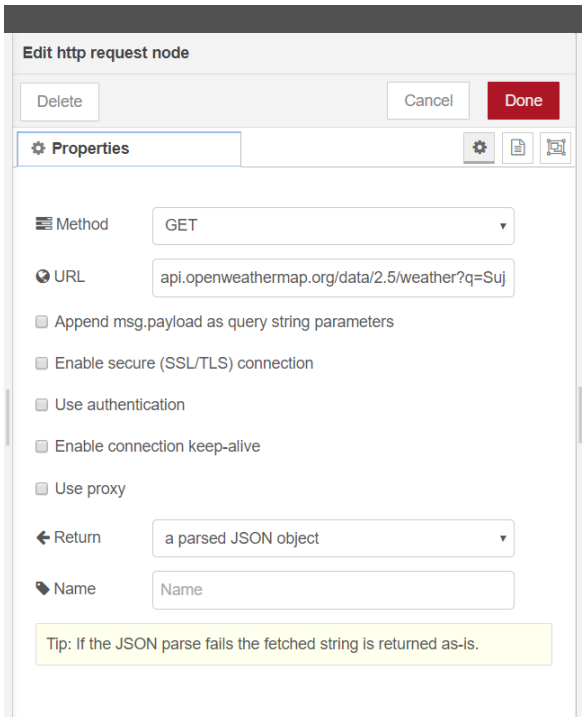
We would need 3 flows:

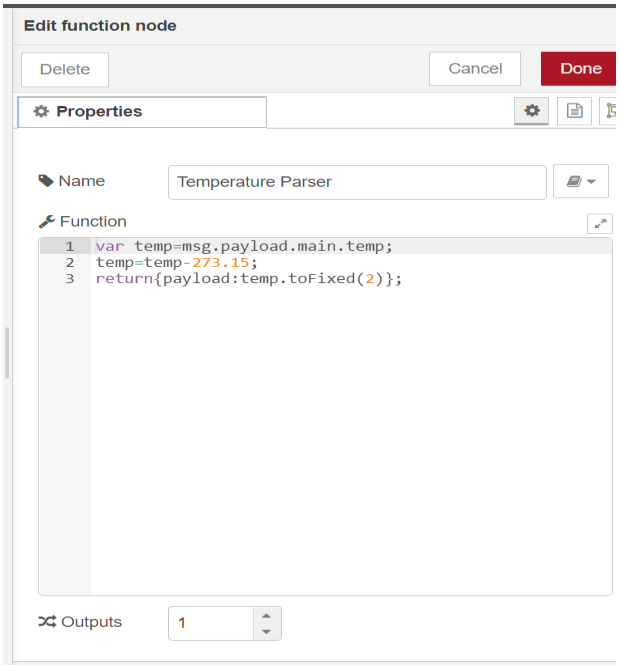
1. To take the weather data from OpenWeather API.
2. To take sensor data from the IBM cloud.
3. Finally, to transfer the motor control data to the cloud.

Flow1

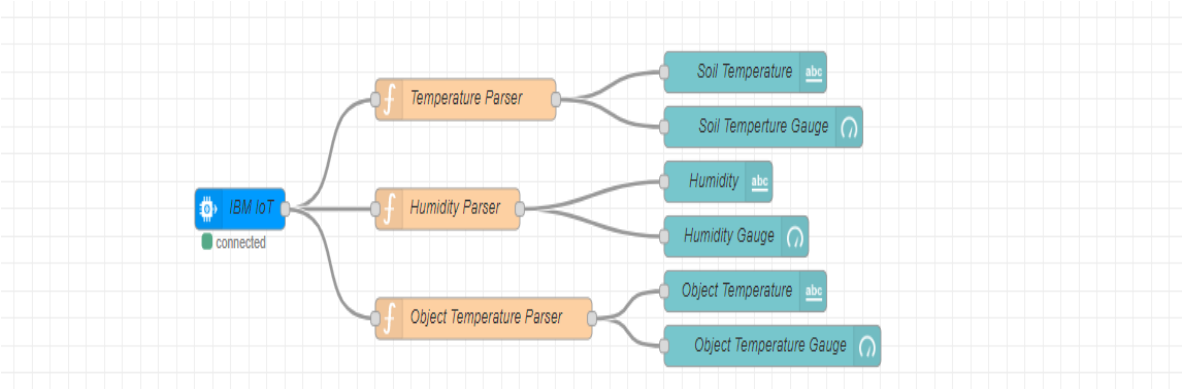


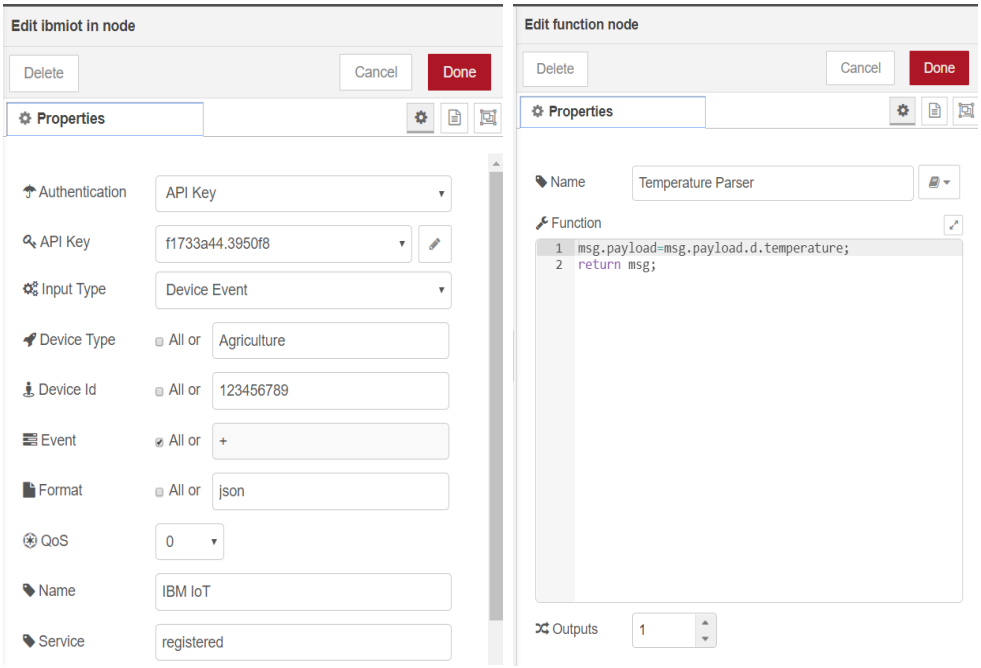
In this the settings for the function nodes and http nodes are as follows:



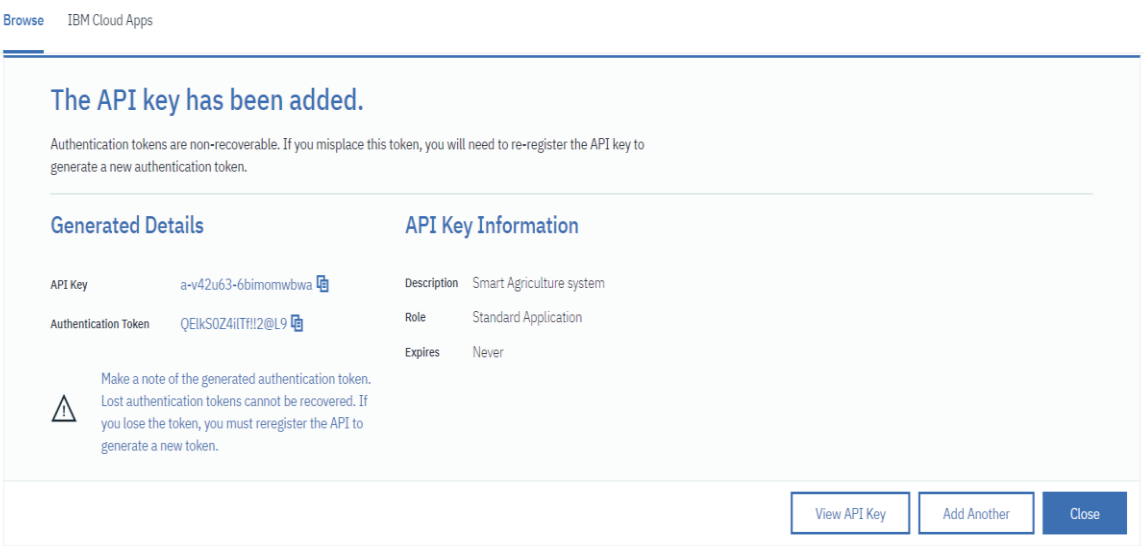


Flow-2

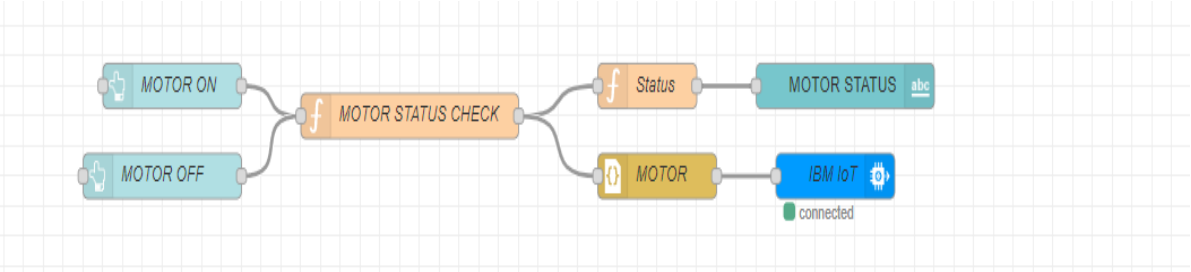




Now we need to put API in IBM IOT node which we can get form IBM cloud from Apps tab.



Flow 3



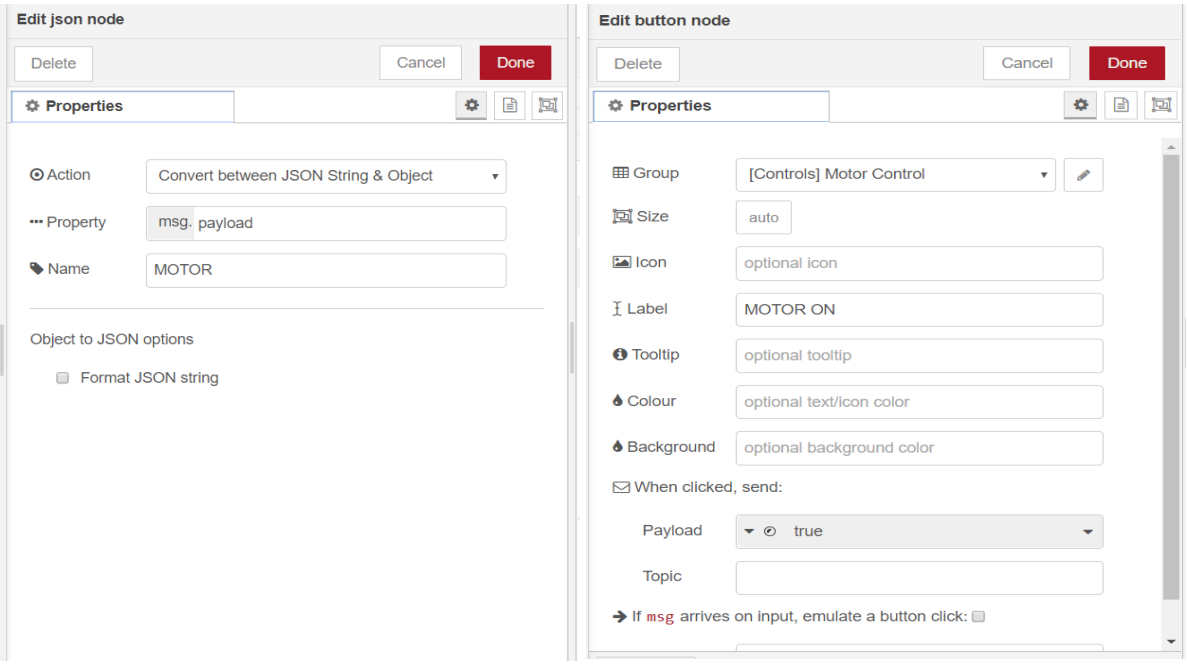
Code for “Motor Status Check”

```
if(msg.payload === true)
   msg.payload = '{"command":"ON"}'
else
   msg.payload = '{"command":"OFF"}'
return msg;
```

Code for “status”

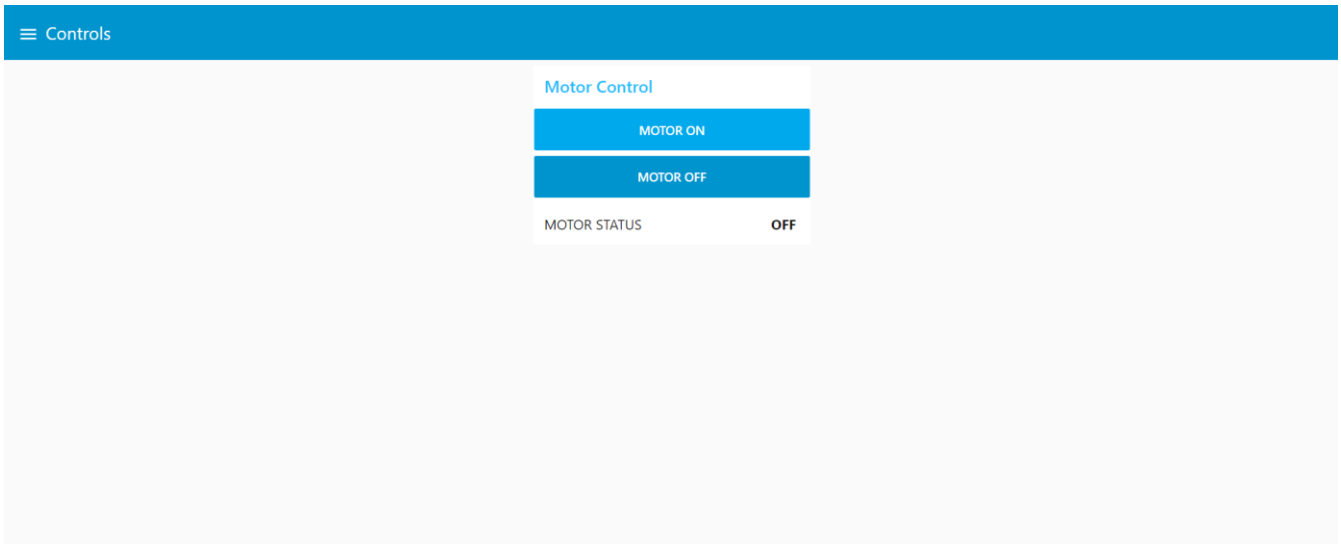
```
var data=JSON.parse(msg.payload);
msg.payload=data.command;
return msg;
```

Settings for “MOTOR” and Buttons



Web APP


Home
IBM Watson
Soil Temperture
18
0
°C
100
Humidity
27
%
Object Temperature
26
Open Weather API
Sujangarh's Temperature
37.47
Humidity
17
Feels Like Gauge
35.21


Controls
Motor Control
MOTOR ON
MOTOR OFF
MOTOR STATUS
OFF

Python program to receive commands from Watson IOT platform

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

#Provide your IBM Watson Device Credentials
organization = "v42u63" #replace the ORG ID
deviceType = "Agriculture"#replace the Device type wi
deviceId = "123456789"#replace Device ID
authMethod = "token"
authToken = "123456789" #Replace the authtoken

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

    elif cmd.data['command']=='OFF':
        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()
```

Final Output

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

    elif cmd.data['command']=='OFF':
        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)
```

```
d:v42u63:Agriculture:123456789
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'OFF'}
Motor OFF IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
Command received: {'command': 'ON'}
Motor ON IS RECEIVED
```