Smart Agriculture system based on IoT

Project aim & scope:

To design a IoT based system for farmers to receive updates on soil moisture, humidity, temperature and other farm parameters remotely and to also to enable farmers to control the farm pumps and water supply to the plants via the internet or a mobile phone app.

To achieve the above goals, IBM Cloud platform will be used to receive data from the virtual IoT device simulator. Node-RED is used to fetch data from the OpenWeather API and IBM cloud platform and display it to the farmer. The farmer can send commands throught the Node-RED app on the computer to a simulated farm which is done using a Python script on a computer.

Team:

1 Member: Stephen Kingston C.

2019608003, Madras Institute of Technology,
Anna University, Chennai.

Setup of the development environment:

- Python 3.7 and PyCharm IDE set up on PC
- IBM cloud platform setup
- GitHub profile to push code
- Node-RED setup on computer

Deliverables:

- Python script to receive commands from the web application
- GitHub repository of all code used to complete the project
- Node-RED program flow
- Project report

Node-RED

Node-RED is a browser based application that allows us to create programming workflows that can help us in creating an IoT application.
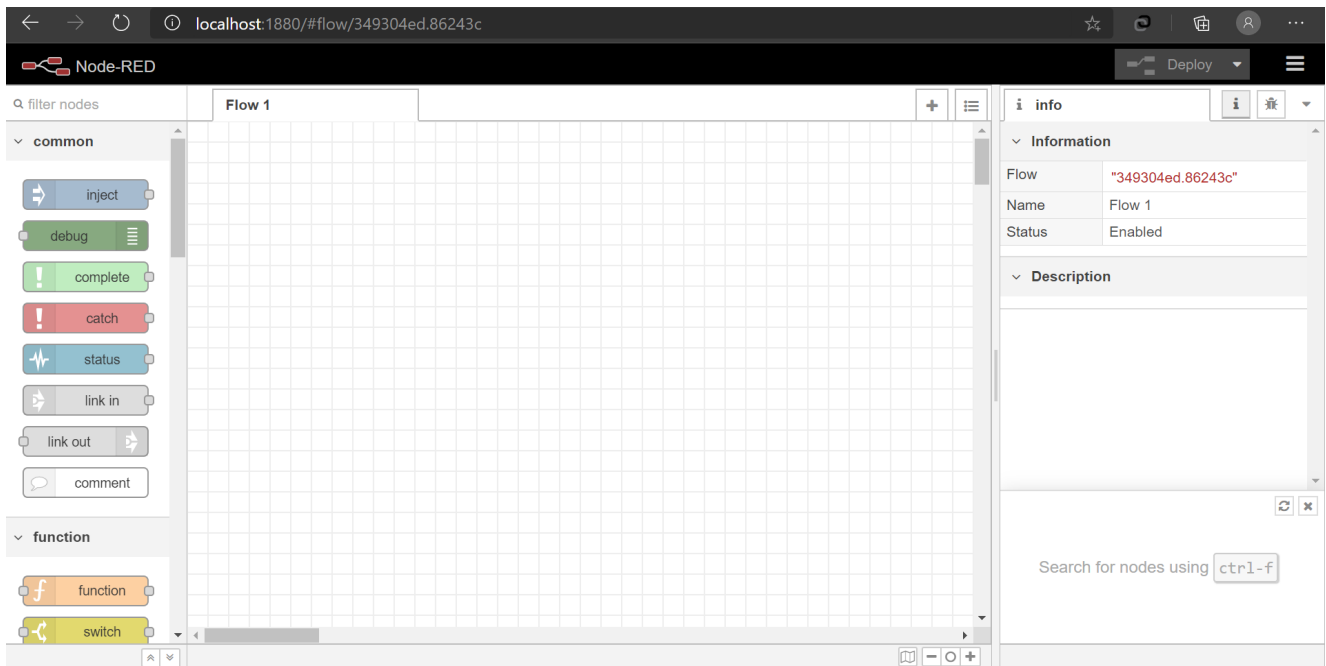
Installation of Node-RED:
Node-RED is a javascript based application and hence requires the installation of npm/Node.js. Once npm is available, node-RED can be installed via the commandline "npm install node-red"

Node-RED is started by typing 'node-red' in the command prompt. Once the Node-RED service is running, a browser is used to open http://localhost:1880/ which gives access to the Node-RED programming platform.

Installation of IBM IoT and Dashboard nodes for Node-RED:
In order to connect to IBM IoT platform and create the UI, Node-RED requires additional nodes which can be added by installing the following packages.

Installation of node-red-contrib-ibm-watson-iot via command prompt
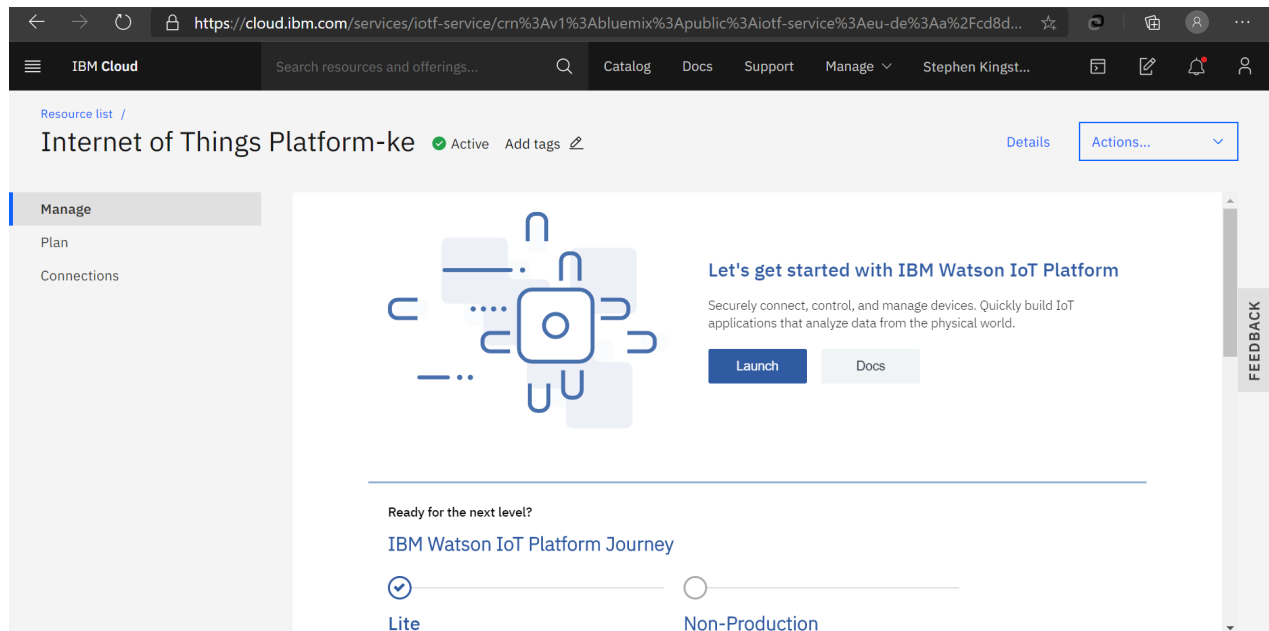   **npm install** node-red-contrib-scx-ibmiotapp

**Installation of Node-RED dashboard**

npm install node-red-dashboard

IBM Watson IoT Platform

IBM Watson IoT Platform is the cloud server that allows us to log device telemetry data from IoT devices anywhere and also allow the end user to send commands, read data to and from the IoT devices using a end user terminal.

IBM Watson allows us to connect each device by the use of unique names and authentication keys and keep us organized.
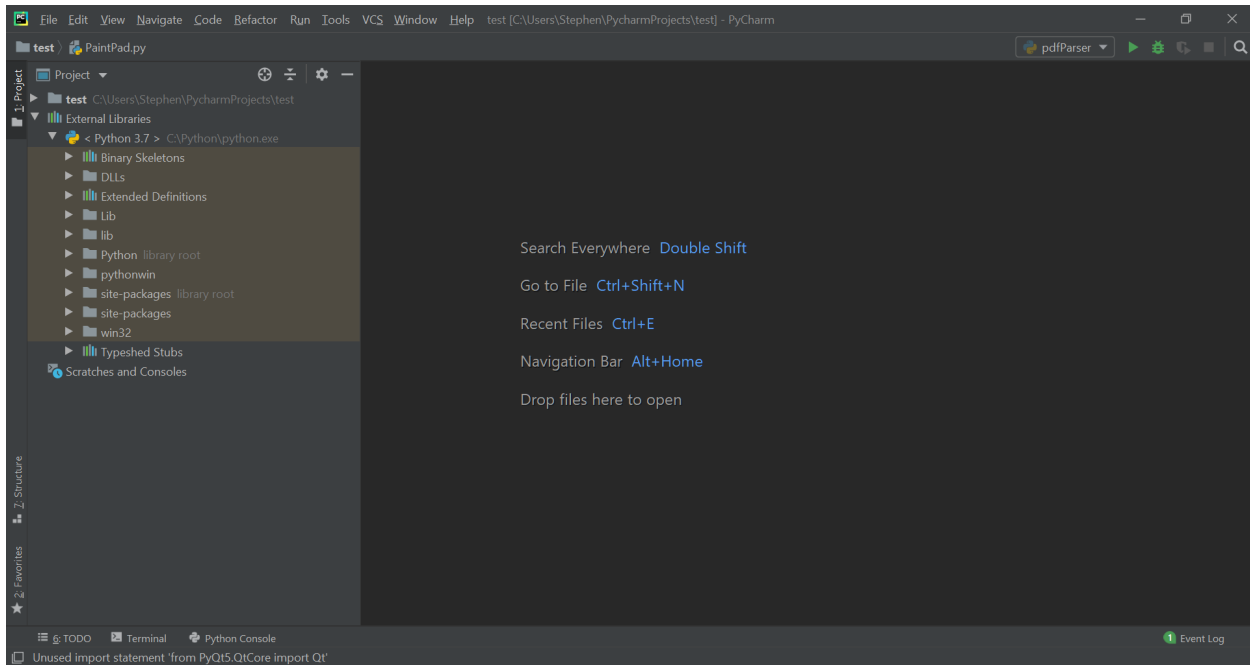
IBM Watson allows free tiers to connect devices that have low data consumption, which was made use of for this project.
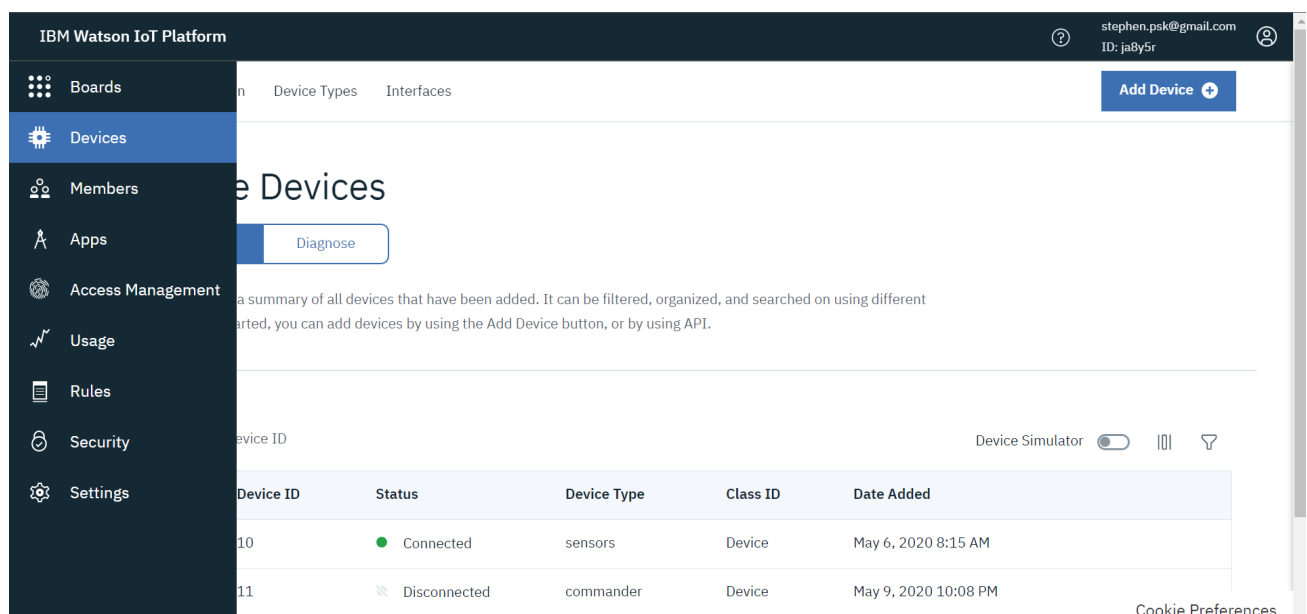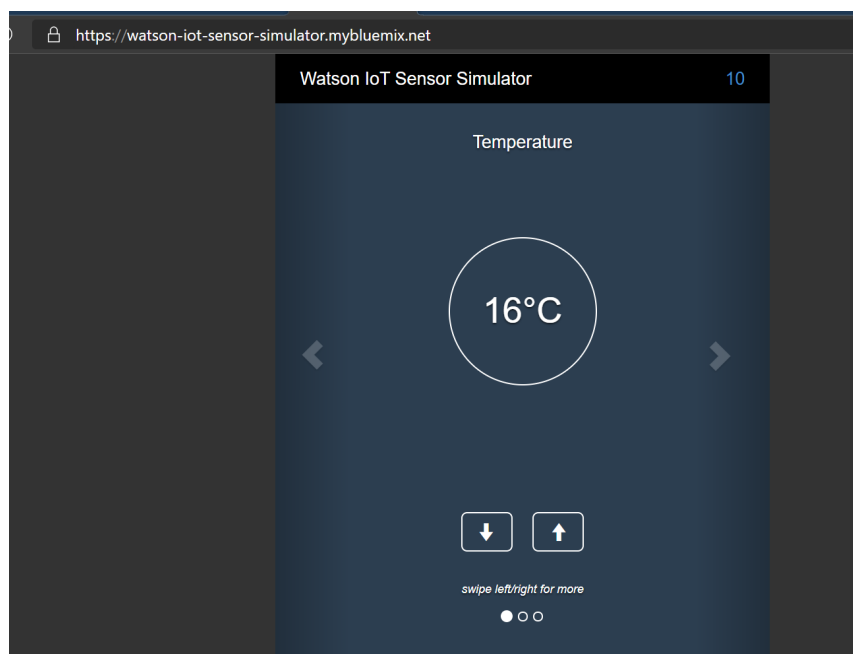
Simulators:

Python/PyCharm IDE

In the absense of an actual farm, a Python script was used to receive the commands of the farmer and control farm equipment.

IoT sensors Simulator:

In the absense of actual sensors collecting data in the farm, we make use of a device simulator that can connect to the cloud platform and send data periodically.

Open Weather API
Open weather API provides a second source of weather information that is accessed from the Node-RED using HTTP requests to OpenWeather's API.

Connecting the IoT simulator to IBM Watson Cloud Platform

IoT Simulator

A device can be added IBM Watson by giving it a Device Type and ID. It created a corresponding Authentication token which can be used to connect a real or simulated device to the cloud.
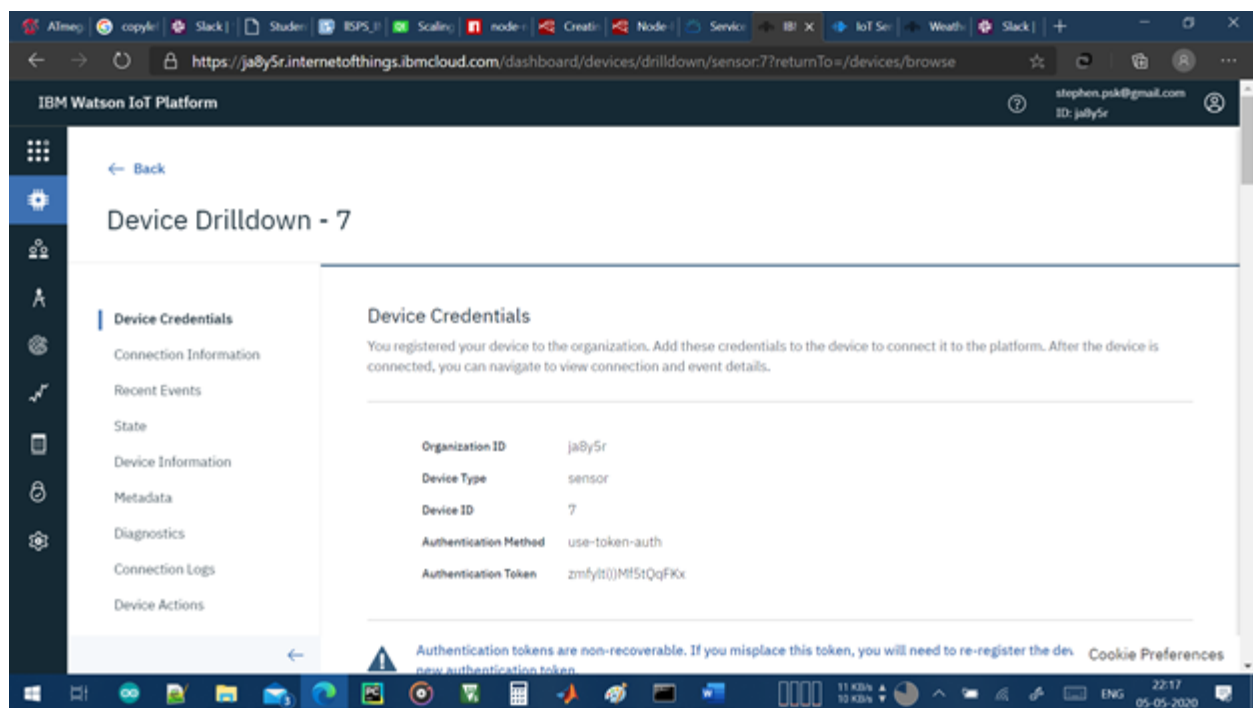
Device Credentials being displayed after successful creation of a device.
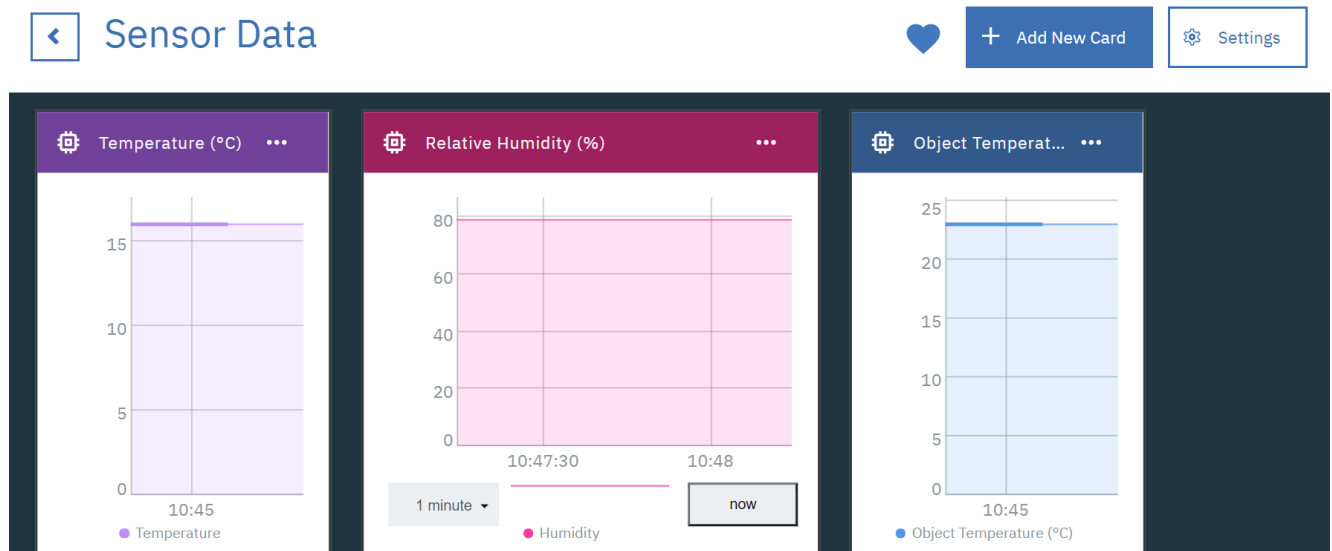
The device is connected by entering the credentials at the device simulator at
https://watson-iot-sensor-simulator.mybluemix.net/

Now the temperature, soil temperature and humidity of the simulated farm is relayed to the cloud platform at regular intervals. The data is sent in JSON format and can be viewed in the device logs on the cloud platform.

Sensor telemetry received in JSON format in IBM cloud:



Connecting Node-RED to the IBM Watson IoT platform:

The module node-red-contrib-scx-ibmiotapp is used to allow Node-RED to connect to IBM cloud IoT platform.

The node IBM IoT App In is added to the Node-RED workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch the device telemetry to Node-RED.
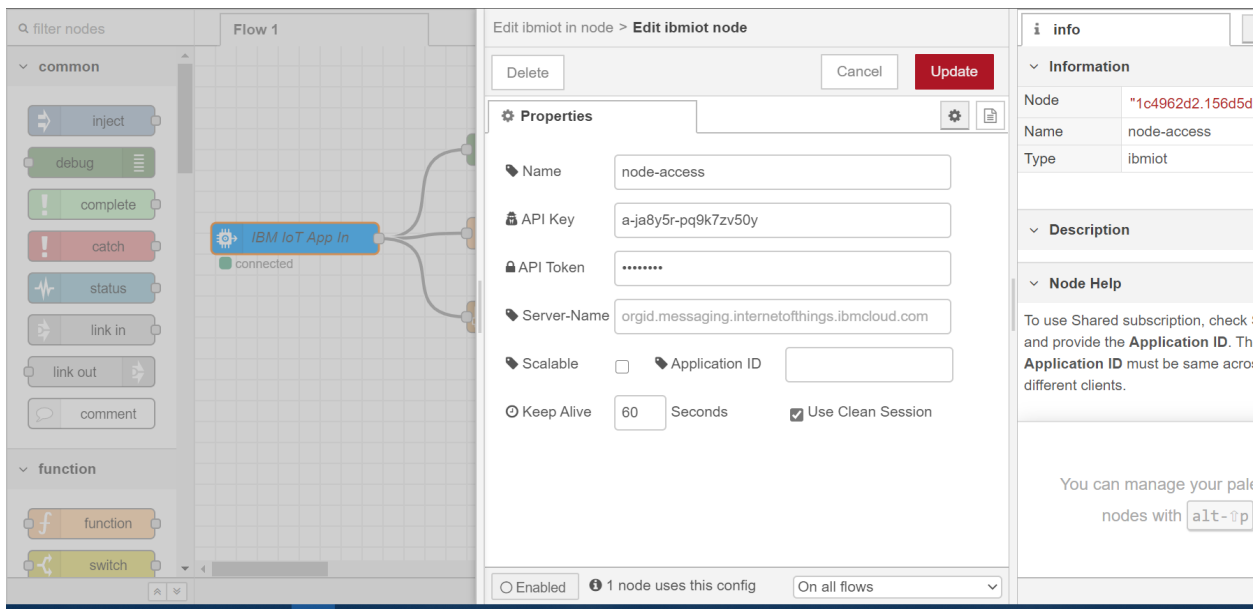
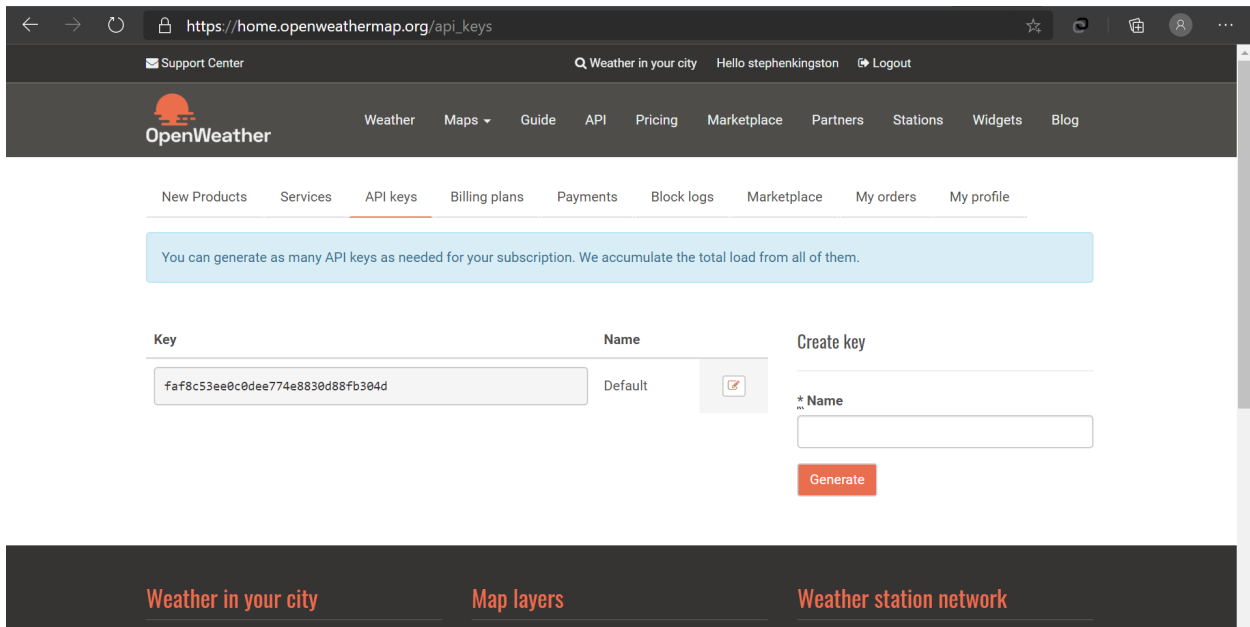**Node-RED connected to IBM Watson & receiving data from device**

Data that was sent to the IBM Cloud Platform from the IoT simulator is displayed on the console as shown in the image below.
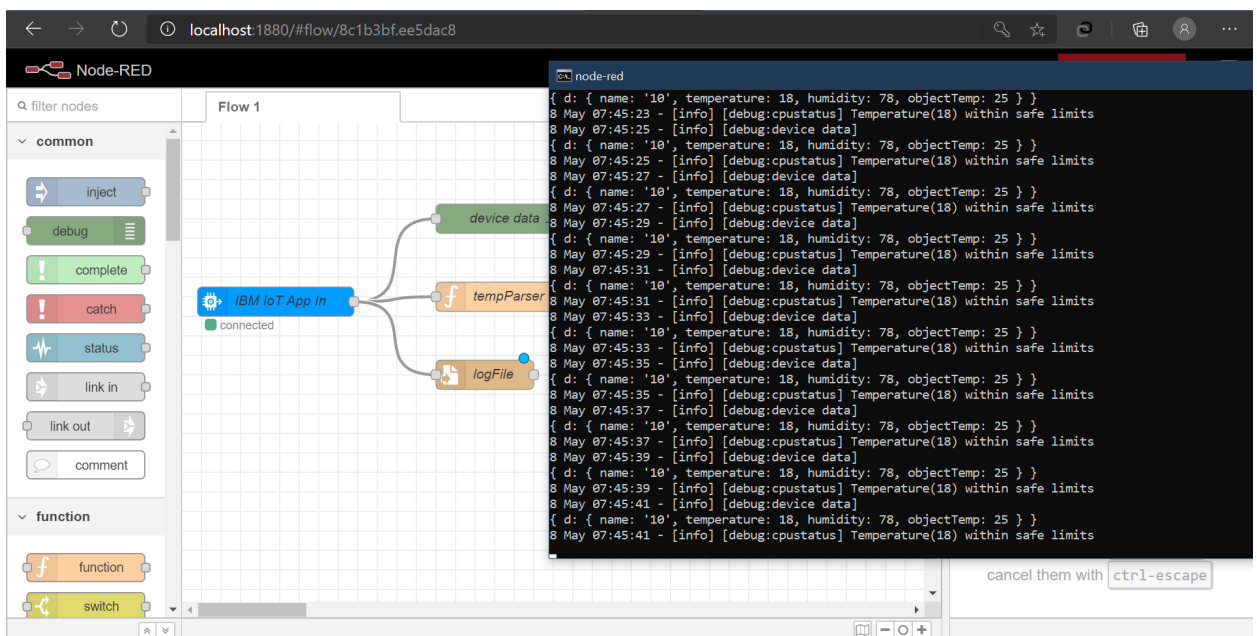
Data received from the cloud in Node-RED console

Creating OpenWeather API account:

A free OpenWeather API account was created and the API key was obtained.

Connecting Open Weather API to the Node-RED

The Node-RED can also be made to receive data from the OpenWeather API by HTTP GET requests. An inject trigger is added to perform the HTTP request every ten seconds.

The HTTP GET request placed to the server is in the following format: **api.openweathermap.org/data/2.5/weather?id={City-ID}&appid={API-KEY-obtained}**

The response is a JSON object as the following:
```
{"coord":{"lon":76.97,"lat":11},"weather":[{"id":721,"main":"Haze","description":"haze","icon":"50d"}],"base":"stations","main":{"temp":301.15,"feels_like":305.38,"temp_min":301.15,"temp_max":301.15,"pressure":1012,"humidity":78},"visibility":5000,"wind":{"speed":2.1,"deg":30},"clouds":{"all":40},"dt":1589080298,"sys":{"type":1,"id":9206,"country":"IN","sunrise":1589070646,"sunset":1589115979},"timezone":19800,"id":1273865,"name":"Coimbatore","cod":200}
```
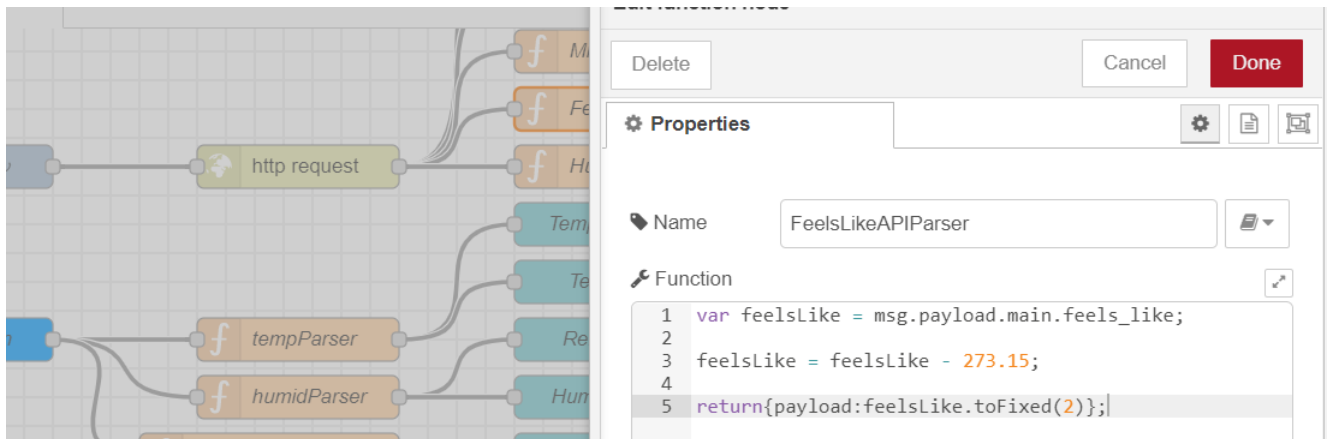
Parsing the JSON data
In order to obtain each parameter, javascript functions are used to parse the JSON string.

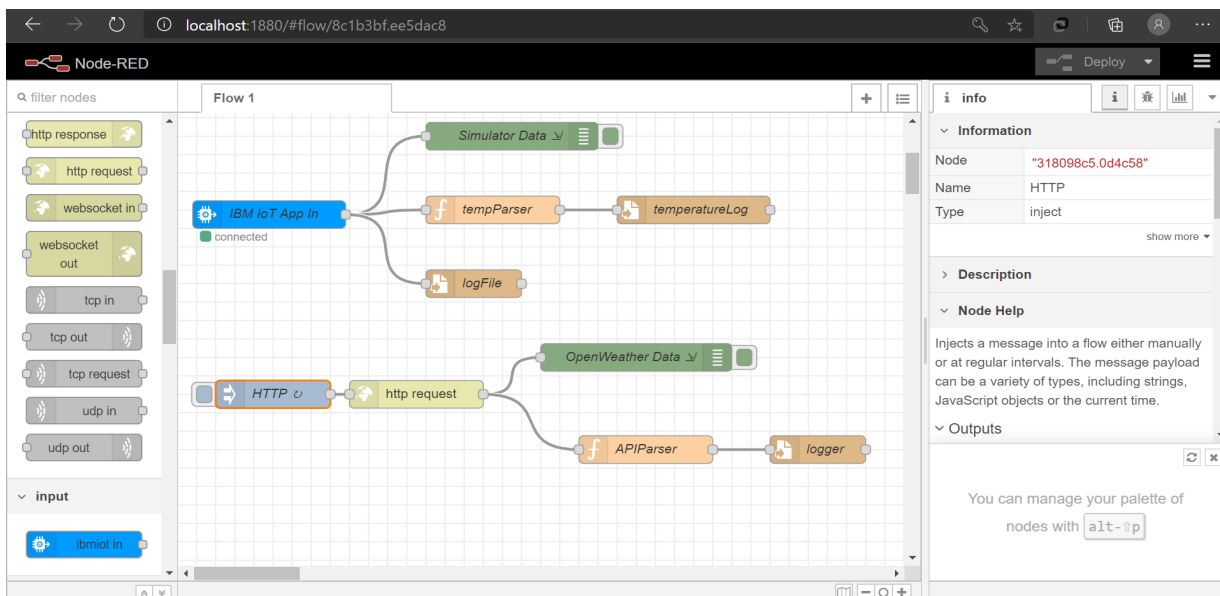**This javscript function returns the temperature in Celsius from Kelvin**

```
var temp = msg.payload.main.temp;
temp = temp - 273.15;
return{payload:temp.toFixed(2)};
```

Similiarly, other parameters can be obtained from the JSON string using the parameter path in the place 'main.temp'

Connecting both IBM Watson IoT and OpenWeather API inputs via HTTP to Node-RED

RAW JSON output displayed to the console:

```
    temp_min: 303.15,
    temp_max: 303.15,
    pressure: 1013,
    humidity: 70
  },
  visibility: 6000,
  wind: { speed: 1.5 },
  clouds: { all: 40 },
  dt: 1588911974,
  sys: {
    type: 1,
    id: 9206,
    country: 'IN',
    sunrise: 1588897878,
    sunset: 1588943155
  },
  timezone: 19800,
  id: 1273865,
  name: 'Coimbatore',
  cod: 200
}
8 May 10:04:48 - [info] Stopping flows
8 May 10:04:48 - [info] Stopped flows
8 May 10:04:48 - [info] Starting flows
8 May 10:04:48 - [info] Started flows
8 May 10:05:01 - [info] [debug:Simulator Data]
{ d: { name: '10', temperature: 18, humidity: 78, objectTemp: 25 } }
8 May 10:05:03 - [info] [debug:Simulator Data]
{ d: { name: '10', temperature: 18, humidity: 78, objectTemp: 25 } }
8 May 10:05:05 - [info] [debug:Simulator Data]
```

Node-RED UI:

In order to display the parsed JSON data, a Node-RED dashboard is created.

Creation:
Node-RED has the button, text and gauge nodes that will be used to display our parsed JSON parameters and also to enable us to control farm equipment remotely using our Node-RED web application.

Creating gauges and Text widgets on the UI

Creation of a gauge widget to display telemetry

Configuration information page for the gauge widget to display a temperature gauge.

Edit text node

		Cancel	Done
Delete			

⚙ Properties

Group   [Home] OpenWeather API

Size   auto

Label   Max. Temperature (°C)

Value format   {{msg.payload}}

Layout

label **value**	label **value**	label **value**
label **value**	label **value**	

Name   Max. Temperature

Creation of a text widget to display telemetry

Configuration information page for the gauge widget to display the temperature as text.

This process is repeated for all the parameters to be displayed, both from the IoT simulator and the OpenWeather API.

The Node-RED UI has two pages, one for the temperature, humidity and other parameters from the Watson simulatore and th OpenWeather API, and the other page for the controls.

Creation of buttons to send commands

Buttons are created by using the 'button' node from the dasboard and entering the configuration details to set what text is sent to the servers when the button is pressed. In this case, there are two farm equipments - Water pump and Borewell pump, both of which have separate ON/OFF buttons.

Button configuration:

Each of the buttons is configured to send 'true' for ON and 'false' for OFF. This output is processed by a javascript function to create a JSON string/object to be sent to the cloud IoT platform. This object is then parsed to display the status of both the equipment to the display.

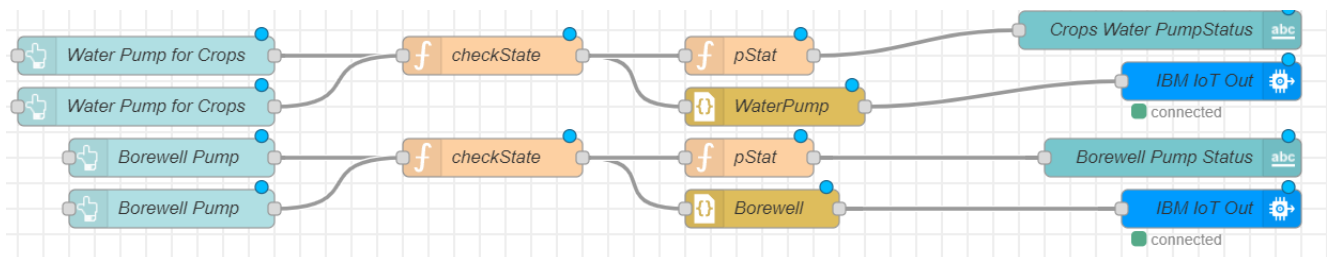**Javascript function to convert true/false to JSON string (checkState)**
if (msg.payload === true)

```
    msg.payload = '{"waterpump" : "ON"}';
else
    msg.payload = '{"waterpump" : "OFF"}';
return msg;
```
The same is repeated for the borewell buttons.

Javascript function to extract equipment status from the JSON string: (pStat)
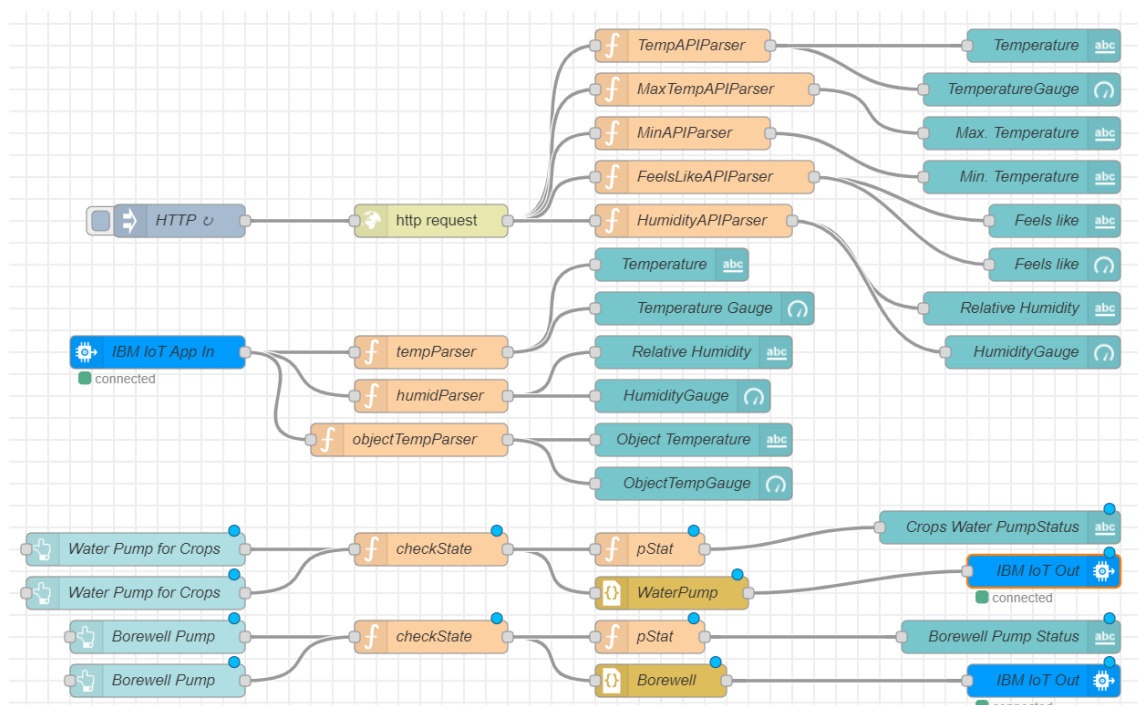
```
var parsedData = JSON.parse(msg.payload);
msg.payload = parsedData.waterpump;

return msg;
```
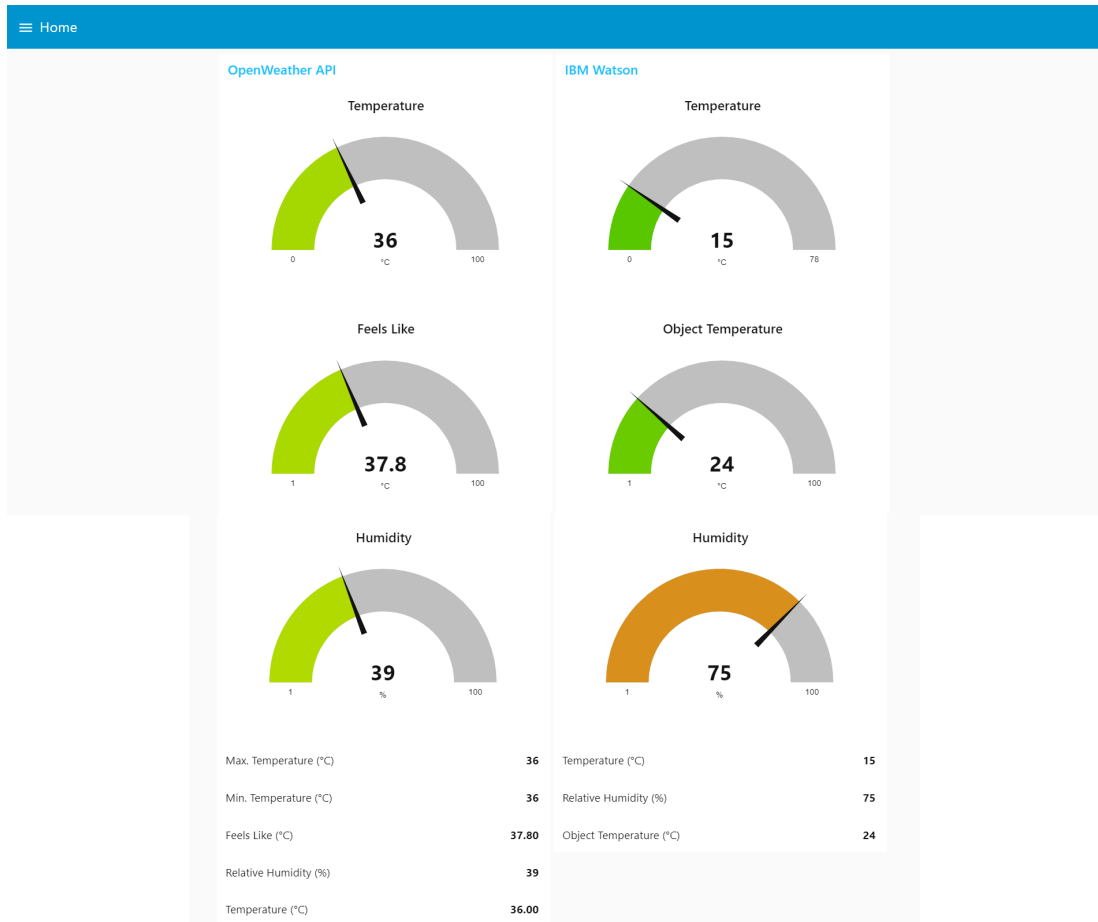
This is converted into a JSON object by the 'WaterPump'/'Borewell' functions, which is sent to the cloud.
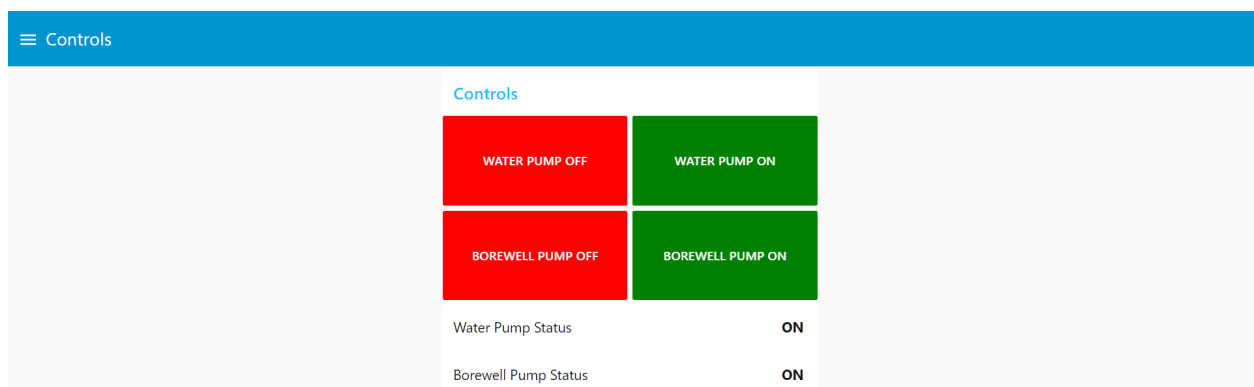
Program flow for the Node-RED application and UI

The following images show the Node-RED UI.

Controls Page on the Node-RED UI

IBM IoT Out:

The JSON Object is finally sent back to the cloud on a separate device that is created on IBM Watson.

Configuration:

Python program to receive commands from Waston IoT platform

```python
import sys
import ibmiotf.device

organization = "ja8y5r"
deviceType = "commander"
deviceId = "11"
authMethod = "token"
authToken = "g0N)@0Txn91p5ESWeL"


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

    for key in cmd.data.keys():
        if key == 'waterpump':
            if cmd.data['waterpump'] == 'ON':
                print("Water pump is turned ON")

            elif cmd.data['waterpump'] == 'OFF':
                print("Water pump is turned OFF")

        if key == 'borewell':
            if cmd.data['borewell'] == 'ON':
                print("Bore well pump is turned ON")
            elif cmd.data['borewell'] == 'OFF':
                print("Bore well pump is turned OFF")

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 = commandHandler
```

Final output

Commands are sent to the Watson IoT platform from Node-RED and Python is used to read the commands from the cloud, acting as the simulated farm microcontroller.