SMARTBRIDGE
Let's Bridge the Gap

SMART AGRICULTURE SYSTEM BASED ON IoT

1. INTRODUCTION

OVERVIEW:

The main aim of this project is to build a system which is can be used by the farmers to monitor wether conditions and soil moisture.In olden Days Farmers Used to figure the ripeness of soil and influenced suspicions to develop which to kind of yield. They didn't think about the humidity, level of water and especially climate condition which terrible a farmers increasingly The Internet of things (IOT) is remodeling the agribusiness empowering the agriculturists through the extensive range of strategies. This project involves building a smart Internet of Things based agriculture system to monitor the weather conditions and soil conditions and help the farmer to gain better yield. This will be accomplished by using the IBM Watson IoT platform and Openweather API. We use Python language to interact with the system.

The eye-catching features of this project include smart irrigation with smart control based on real time field data. Secondly temperature maintenance, humidity maintenance and other environmental parameters. And finally
the recommendation to farmer for smart agriculture.

PURPOSE:

The aim of the project is to provide the farmers with the data regarding the weather and soil conditions through a web app. This makes farming profitable and prevents the damage of the crop in a feasible manner.

2. LITERATURE SURVEY

EXISTING PROBLEM:

Indian agriculture is plagued by several problems; some of them are natural and some others are man made. Of those, being unable to predict the climate and plant requirements is the major problem. Also some of the other problems include soil erosion, lack of mechanisation, irrigation. Soil quality testing is not done effectively in India. Moreover, not being able to adapt to technological trends has become fatal to agriculture.

PROPOSED SOLUTION:

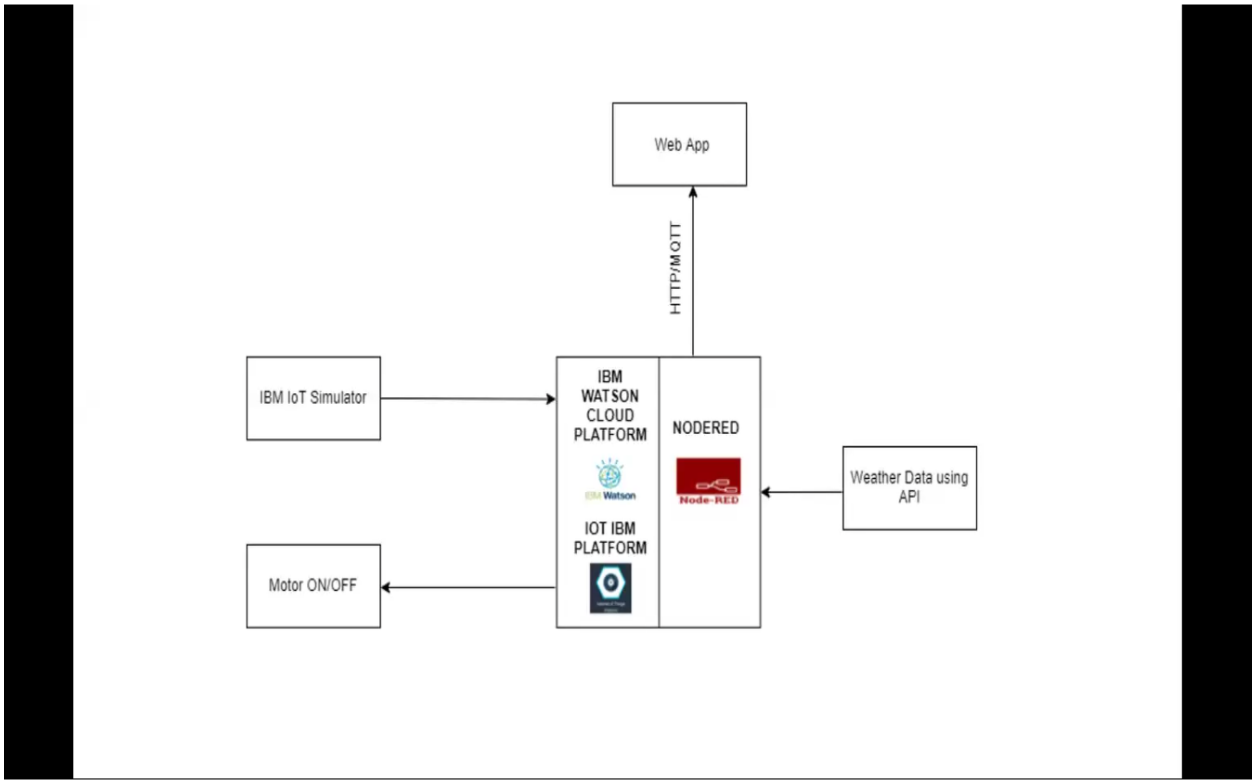
There are many new farming techniques like precision farming, GSM based farming and smart irrigation control. Here, we propose a solution using cloud and IoT to monitor the soil and weather conditions. Temperature, humidity and soil moisture sensors are used to obtain the necessary information and push them to the cloud platform. Further we create a web page which is accessed by the farmers to monitor their crop.

3. THEORETICAL ANALYSIS

PROJECT SCOPE:

We create a device in the IBM Watson IoT platform and enable simulation. The simulation is done in the watson IOT sensor simulator. The sensors take reading every minute and upload to the cloud. Node-red is used to wire together the hardware,online services and APIs. To simulate weather information , we create an account in Openweather.org and provide through the sensors. Later, these are used through a web interface to control the motor.

BLOCK DIAGRAM:

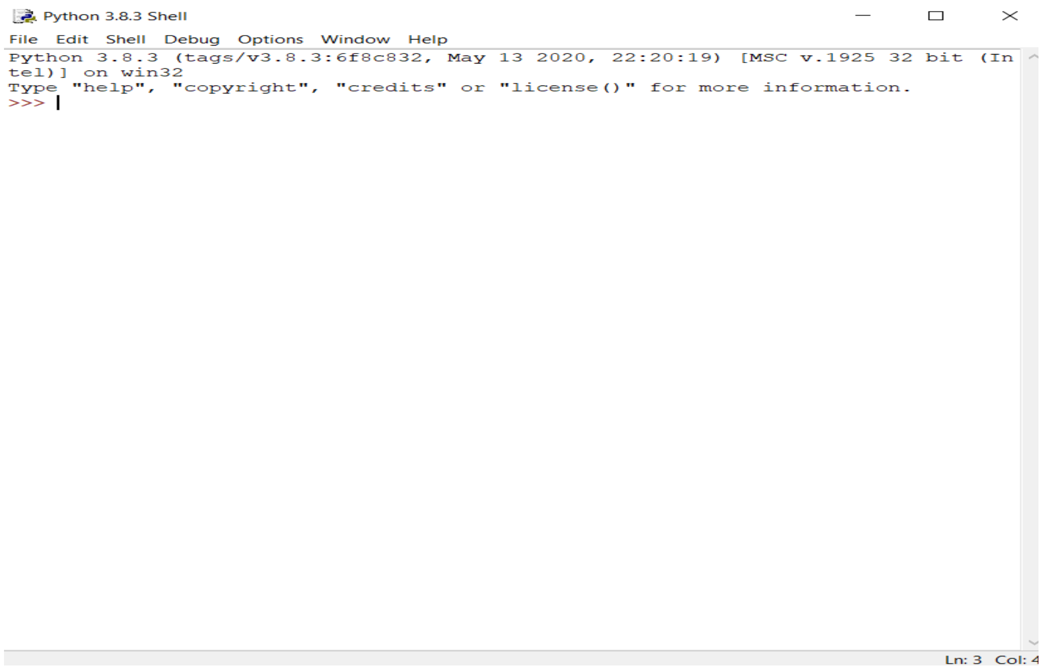


HARDWARE/SOFTWARE DESIGN:

1.Install the required tools and create the required accounts.

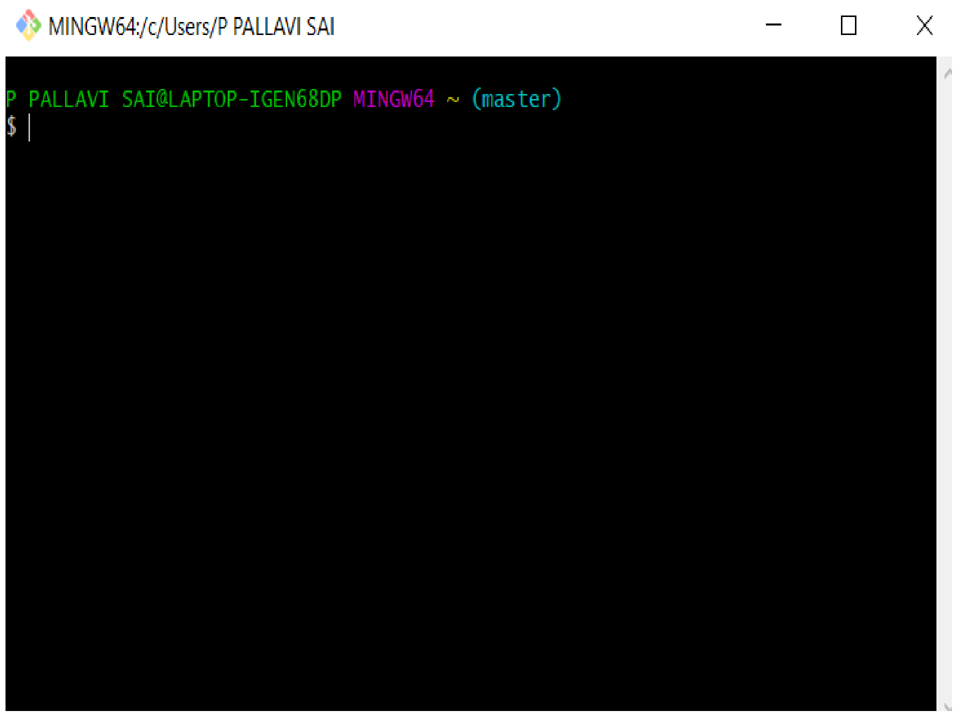
SETTING UP IDE:

We use Python IDE for the project.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
```

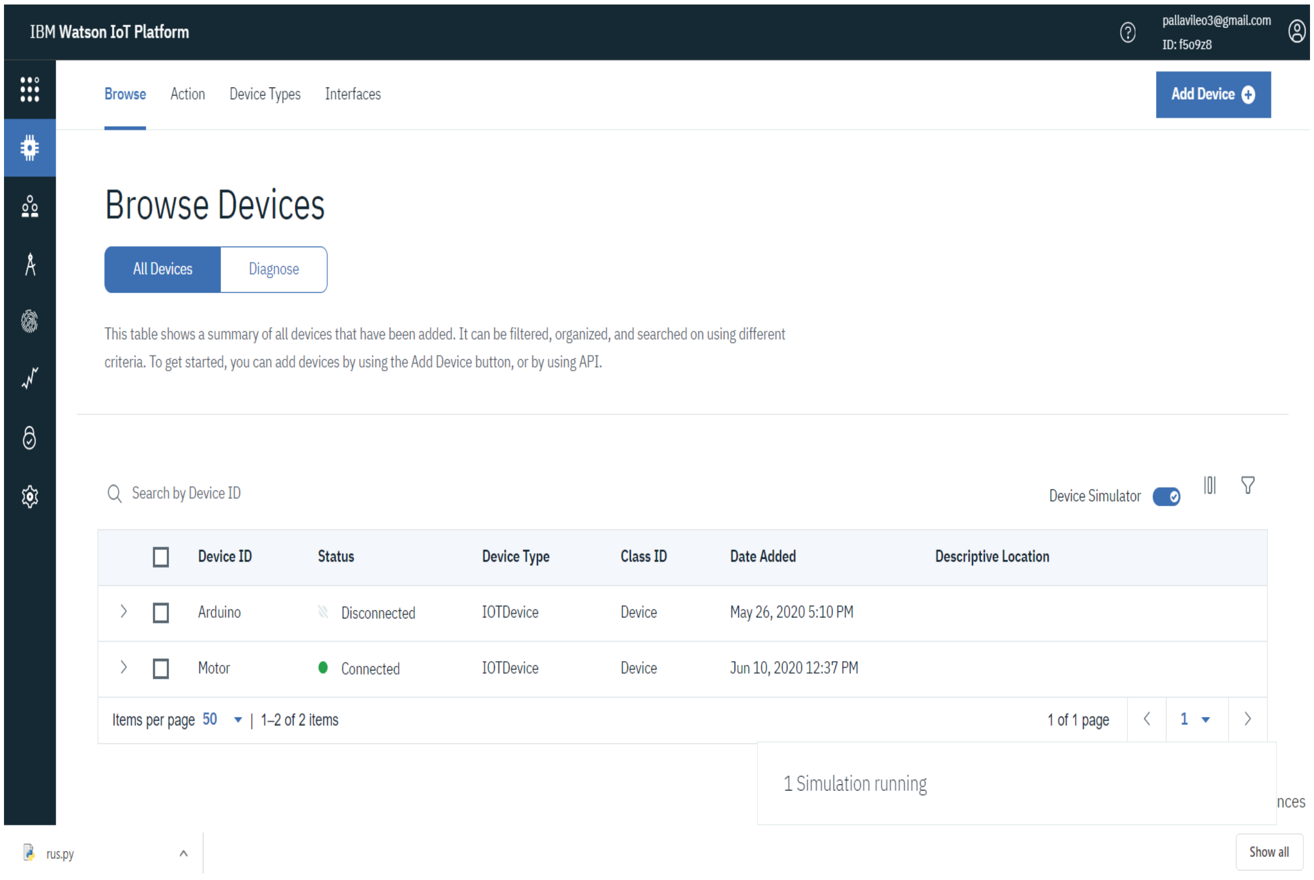
INSTALL GIT:



```
P PALLAVI SAI@LAPTOP-IGEN68DP MINGW64 ~ (master)
$ 
```

2.CREATE A DEVICE IN THE IBM WATSON IOT PLATFORM:

We need to create two devices in the platform.One acts as a processor for the sensor information and the other is an instance of a motor.Also create an API key-token pair and save in a safe place.



Browse Devices

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.

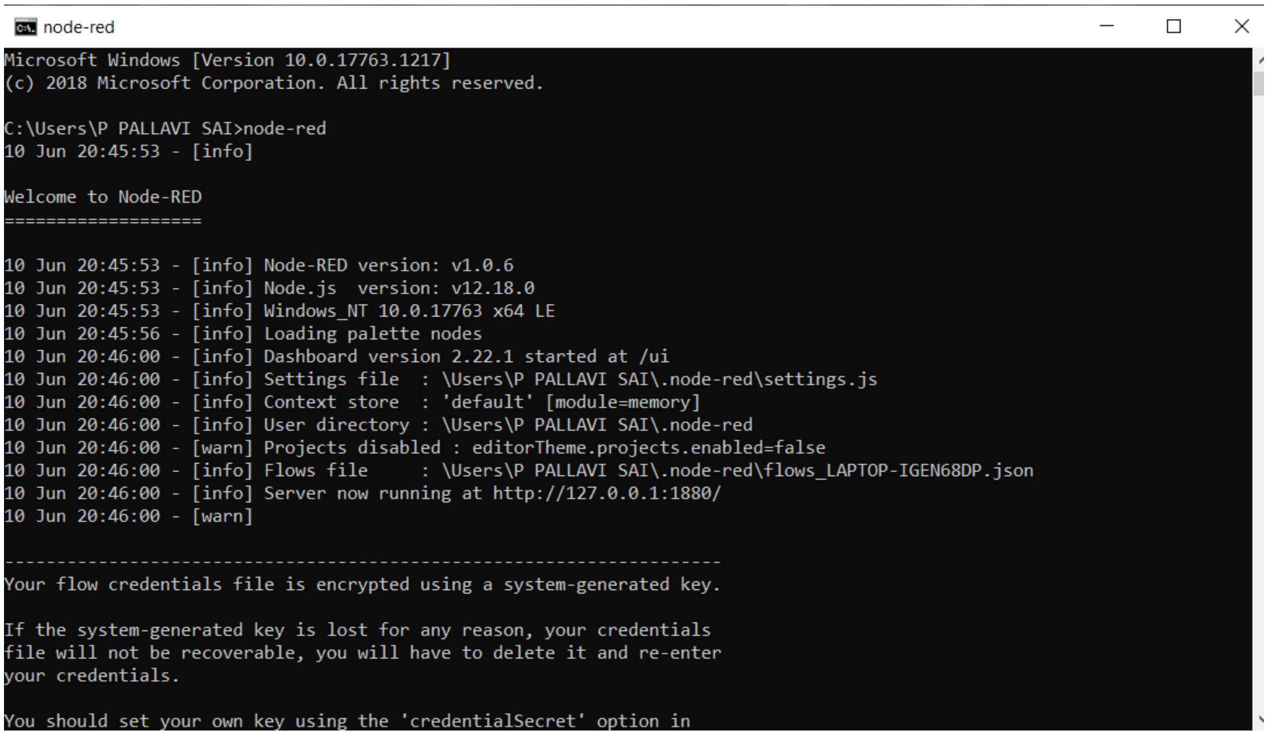
Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
Arduino	Disconnected	IOTDevice	Device	May 26, 2020 5:10 PM	
Motor	● Connected	IOTDevice	Device	Jun 10, 2020 12:37 PM	

Items per page 50 | 1–2 of 2 items   1 of 1 page   1

1 Simulation running

3.INSTALL NODE-RED LOCALLY:

Node-red is a Node.js based implementation and can be installed using chocolatey or yarn.By typing node-red in cmd promt we will get ip adress of node red loaclly and in node-red we can see different filter nodes.



```
Microsoft Windows [Version 10.0.17763.1217]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\P PALLAVI SAI>node-red
10 Jun 20:45:53 - [info]

Welcome to Node-RED
===================

10 Jun 20:45:53 - [info] Node-RED version: v1.0.6
10 Jun 20:45:53 - [info] Node.js  version: v12.18.0
10 Jun 20:45:53 - [info] Windows_NT 10.0.17763 x64 LE
10 Jun 20:45:56 - [info] Loading palette nodes
10 Jun 20:46:00 - [info] Dashboard version 2.22.1 started at /ui
10 Jun 20:46:00 - [info] Settings file  : \Users\P PALLAVI SAI\.node-red\settings.js
10 Jun 20:46:00 - [info] Context store  : 'default' [module=memory]
10 Jun 20:46:00 - [info] User directory : \Users\P PALLAVI SAI\.node-red
10 Jun 20:46:00 - [warn] Projects disabled : editorTheme.projects.enabled=false
10 Jun 20:46:00 - [info] Flows file     : \Users\P PALLAVI SAI\.node-red\flows_LAPTOP-IGEN68DP.json
10 Jun 20:46:00 - [info] Server now running at http://127.0.0.1:1880/
10 Jun 20:46:00 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
```

4.CONNECT THE IOT DEVICE TO A SIMULATOR

The iot sensor simulator requires the following information.

1.Device organization

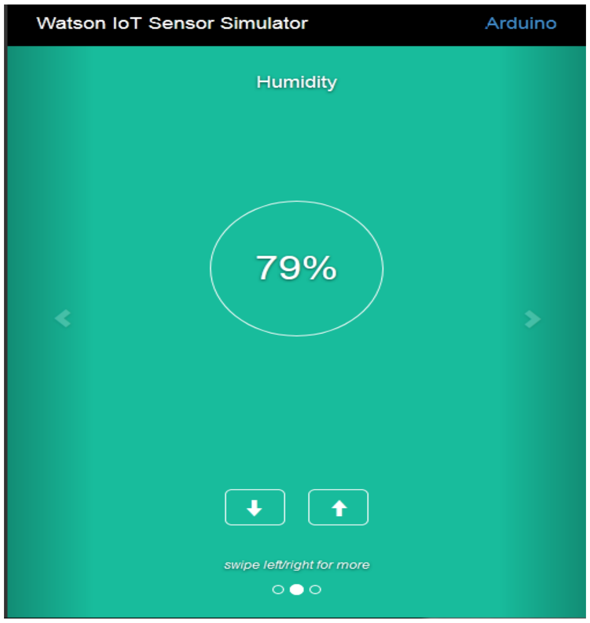
2.Device Type

3.Device Id

4.Device Token

Once given all the required instructions, the sensor is connected to the device we created.

This can be verified by the device name in blue colour on top right.



5.VISUALISING THE DATA:

Once the device is connected, we can create boards in the IBM IoT platform.Boards-->Create new board--> Add Cards
Different types of visualizations like line plot,gauge can be done in the IBM platform.

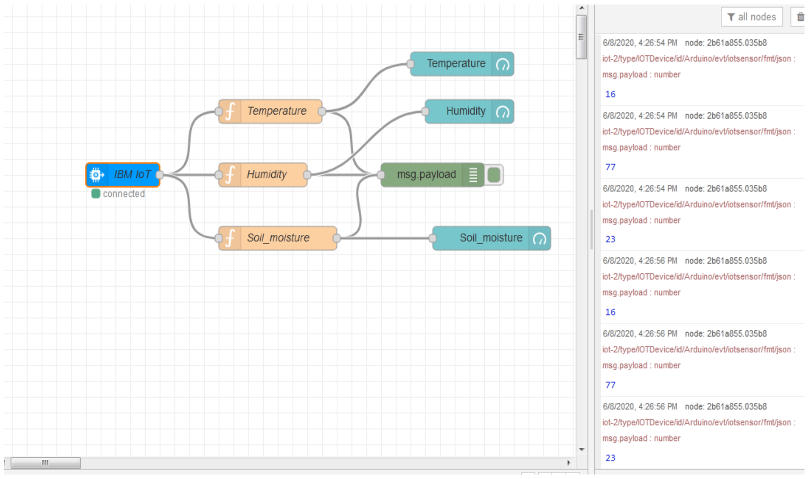
6.CONNECTING TO OPENWEATHER API:

1. Create an account in openweather.org
2. Go to API marketplace and register a API key
3. Select "By city Name' in API options
4. Copy and save the API call to be used later

7.CREATING THE UI USING NODE-RED:

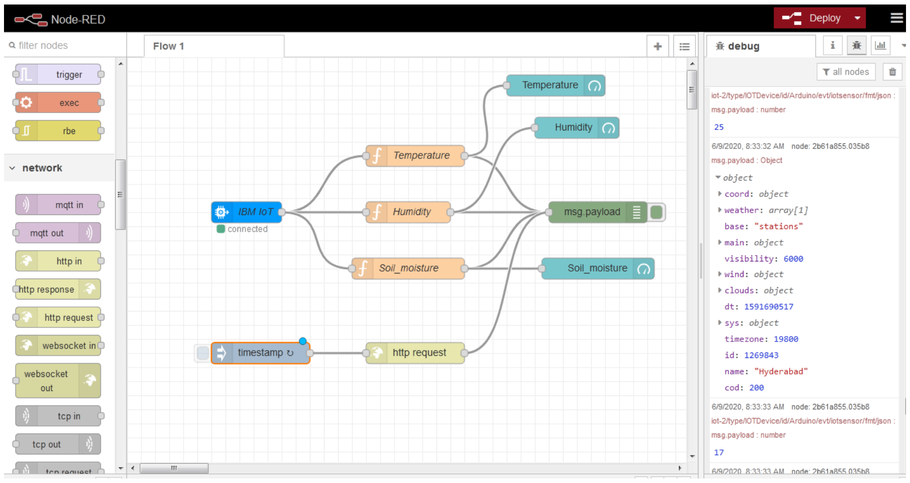
1. We need to create flows for our purposes.
2. We need to install the ibm iot node package and ibm watson package. To install a new node-set,

   manage pallette-->install-->search for required nodes
3. Firstly, to take the input data from the sensor we make the following flow .
4. This also creates a UI for the user to interact with the data and the devices.



```
6/8/2020, 4:26:54 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
16
6/8/2020, 4:26:54 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
77
6/8/2020, 4:26:54 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
23
6/8/2020, 4:26:56 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
16
6/8/2020, 4:26:56 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
77
6/8/2020, 4:26:56 PM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
23
```

5. To view the data from API,we use a http request node and inject node.



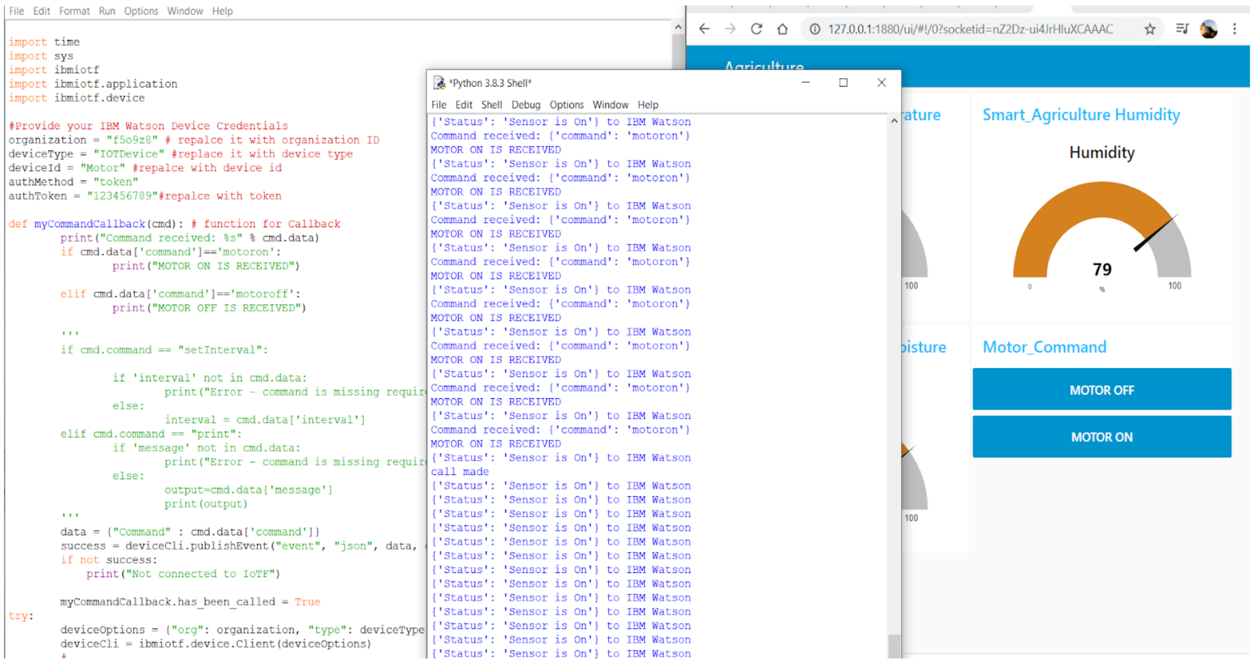
```
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
25
6/9/2020, 8:33:32 AM   node: 2b61a855.035b8
msg.payload : Object
▾object
 ▸coord: object
 ▸weather: array[1]
  base: "stations"
 ▸main: object
  visibility: 6000
 ▸wind: object
 ▸clouds: object
  dt: 1591690517
 ▸sys: object
  timezone: 19800
  id: 1269843
  name: "Hyderabad"
  cod: 200
6/9/2020, 8:33:33 AM   node: 2b61a855.035b8
iot-2/type/IOTDevice/id/Arduino/evt/iotsensor/fmt/json :
msg.payload : number
17
6/9/2020, 8:33:33 AM   node: 2b61a855.035b8
```

6. To give the instructions to the device,in our case motor we create another flow using the ibm out node and Motor ON and OFF buttons.

8.RUNNING THE PYTHON CODE AND TURNING MOTOR ON OR OFF:

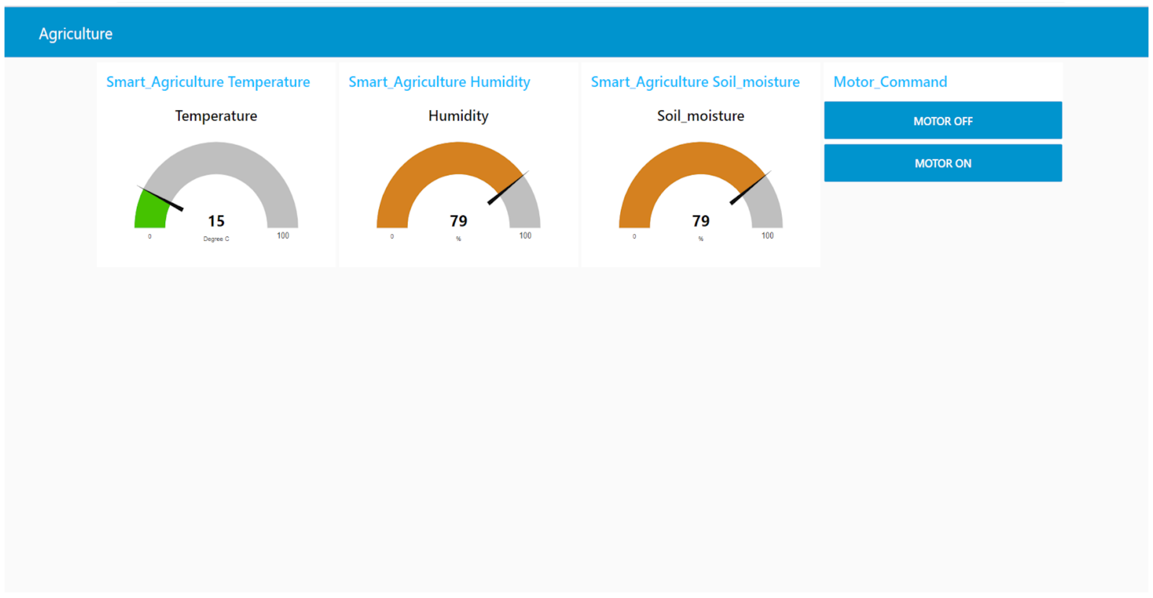
We use the python programming language to interact with the devices and the cloud. On clicking the button in the UI,the motor can be made ON or OFF.

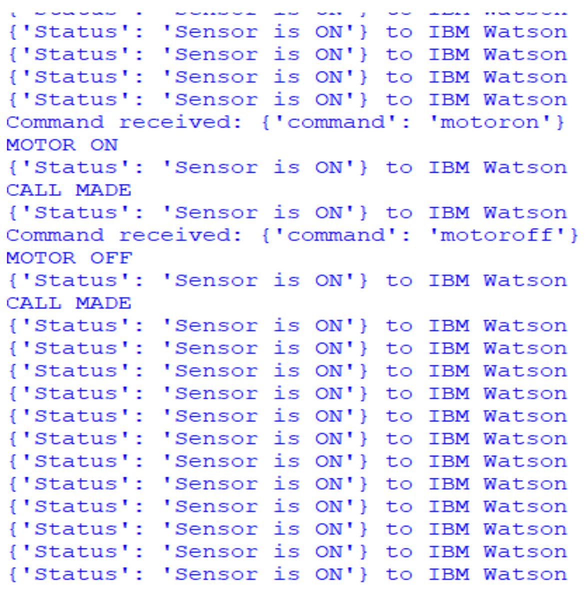
RESULTS:


File Edit Format Run Options Window Help
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
127.0.0.1:1880/ui/#!/0?socketid=nZ2Dz-ui4JrHluXCAAAC
Agriculture
Smart_Agriculture Humidity
Humidity
79
0
%
100
Motor_Command
MOTOR OFF
MOTOR ON

```
import time
import sys
import ibmiotf
import ibmiotf.application
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "f5o9z8" # repalce it with organization ID
deviceType = "IOTDevice" #replace it with device type
deviceId = "Motor" #repalce with device id
authMethod = "token"
authToken = "123456789"#repalce with token

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 requir
                else:
                        interval = cmd.data['interval']
        elif cmd.command == "print":
                if 'message' not in cmd.data:
                        print("Error - command is missing requir
                else:
                        output=cmd.data['message']
                        print(output)
        '''
        data = {"Command" : cmd.data['command']}
        success = deviceCli.publishEvent("event", "json", data, 
        if not success:
            print("Not connected to IoTF")

        myCommandCallback.has_been_called = True
try:
        deviceOptions = {"org": organization, "type": deviceType
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #
```

```
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
{'Status': 'Sensor is On'} to IBM Watson
call made
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
{'Status': 'Sensor is On'} to IBM Watson
```





```
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
Command received: {'command': 'motoron'}
MOTOR ON
{'Status': 'Sensor is ON'} to IBM Watson
CALL MADE
{'Status': 'Sensor is ON'} to IBM Watson
Command received: {'command': 'motoroff'}
MOTOR OFF
{'Status': 'Sensor is ON'} to IBM Watson
CALL MADE
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
{'Status': 'Sensor is ON'} to IBM Watson
```

FUTURE SCOPE:

The project can be further extended to enabling the usage of AI in the agriculture ecosystem. We can also integrate the system using solar pannels which replace the conventional electricity methods. We can suggest crops based on the climatic conditions of the data. Based on the water level, we can alert the farmer or automatically turn the motor off.