

Internship Project Report on
**“SMART AGRICULTURE SYSTEM BASED ON
IOT”**

at
SMARTBRIDGE

(An edTech organization with a vision to bridge the gap between academia & industry.)



MAY-JUNE 2020

Submitted by :
Name: Samiksha Mulik
Intern Id: SB51629
Project Id: SPS_PRO_101

Guided by :
Mr. Durga Prasad

CONTENTS

1 INTRODUCTION

1.1 Overview

1.1 Purpose

2 LITREATURE SURVEY

2.1 Existing Problem

2.2 Proposed Solution

3 THEORETICAL ANALYSIS

3.1 Block Diagram

3.2 Software Designing

4 EXPERIMENTAL INVESTIGATION

5 FLOW CHART

6 RESULT

7 ADVANTAGES/ DISADVANTAGES

8 APPLICATIONS

9 CONCLUSION

10 BIBLIOGRAPHY

11 APPENDIX

A.Source Code

1. INTRODUCTION

1.1 Overview

- Agricultural lands are the heart of any country for economic development. Thus, it is the primary duty of the Government to preserve and protect the fields by any means.
- Science and new technologies have evolved but nothing could replace the dependency on agricultural farm lands. New technologies have been proposed for the betterment of the farmers, so that they can get a better result with more accuracy and less effort but with some limitations.
- India holds the 2nd position in the farm output. Over 70% of the rural households depend on agriculture as their principal means of livelihood. But the pressure on farms increased due to increase in population. This leads to more consumption of non- renewable energy sources.

1.2 Purpose

- Keeping in mind the practical problems faced by the farmers, with the collaboration of SMARTBRIDGE (through RSIP- 2020), I put forward an alternative agricultural model for the betterment of the next generation.
- We are going to use various modules like soil moisture sensor, humidity sensor, temperature sensor under a single agriculture system to make it smarter.
- We get the information about various parameters that effect the crop production through these sensors and make

decisions to monitor the crop production even from distant places using cloud technology.

- In this project, we assumed the simulated data coming to the cloud from the sensors. This will benefit the farmers as they won't have to go physically to check the condition of the farm.

2. LITERATURE SURVEY

2.1 Existing problems

- Farmers have to be present at farm for their maintenance irrespective of the weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically.
- Farmers 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 us with food.
- To have a control on their work virtually by monitoring weather. For this they need minute to minute updates on weather conditions. This will save the time to go farm daily to check weather conditions.

2.2 Proposed Solution

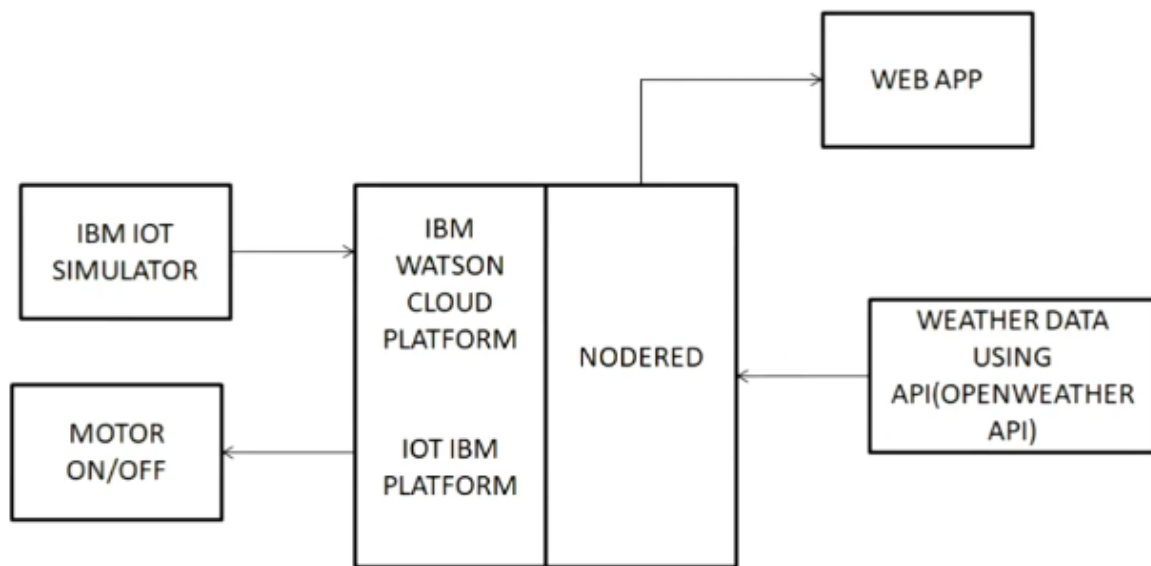
- In order to improve the farmers working conditions and make them easier, we introduce IoT services to farmer in which we use cloud services and internet to enable farmer to continue his work remotely via internet.
- He can monitor the field parameters and control the devices in farm

3. THEORETICAL ANALYSIS

3.1 Block diagram

The following approach is used for the system

WORK FLOW



3.2 Software Designing

1. Watson IOT Platform:

Two devices have been created in IBM Watson IOT Platform. One for sending the command to the User and another to receive the data from an IOT simulator (Temperature, humidity & soil moisture) and Open Weather API (recent weather information of the farm). Device is connected to the IOT Simulator to get the simulator data.

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.
- Node red is installed on the PC and required nodes are installed in the node-red to configure the device to display the received data from simulator and open weather api to user interface dashboard.

3. Web App

- A web application is created which displays the temperature, humidity, and soil moisture data which is received by the device from the IOT simulator. It also displays live weather parameters of the farm using Open Weather API.
- There are set of buttons on the web application that can be used to control the motor on the farm to turn them ON/OFF remotely.
- A python code is written to track down the commands (like turning motor ON/OFF) that are being sent by the user through web application.

4.EXPERIMENTAL INVESTIGATION

4.1 Creating IBM cloud account and setting up the device in IBM Watson IoT Platform

- IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.
- Below are the devices created as per the requirement of the project, one for input and other for output:

IBM Watson IoT Platform

si05202000429@smartinternz.com
ID: nwuprq

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 ☐

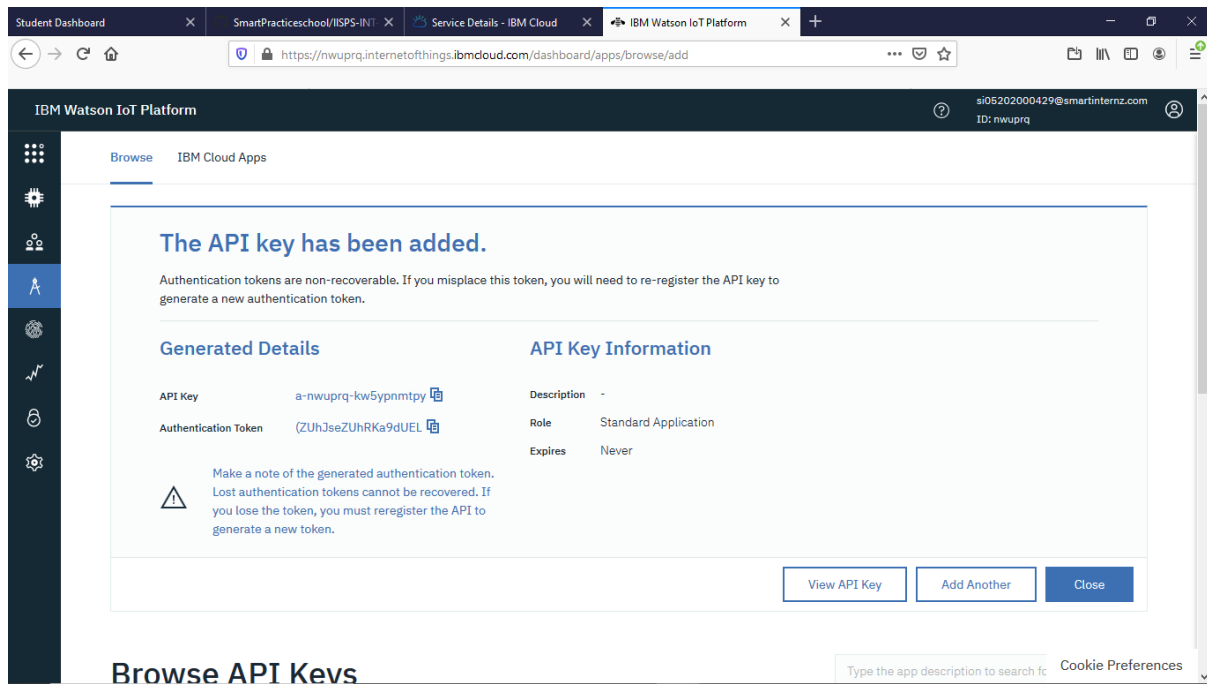
<input type="checkbox"/>	Device ID	Status	Device Type	Class ID	Date Added
> <input type="checkbox"/>	Arduino	Disconnected	IOTDevice	Device	Jun 19, 2020 9:37 AM
> <input type="checkbox"/>	Output	Disconnected	IOTDevice	Device	Jun 23, 2020 4:50 PM

Items per page 50 | 1-2 of 2 items

1 of 1 page

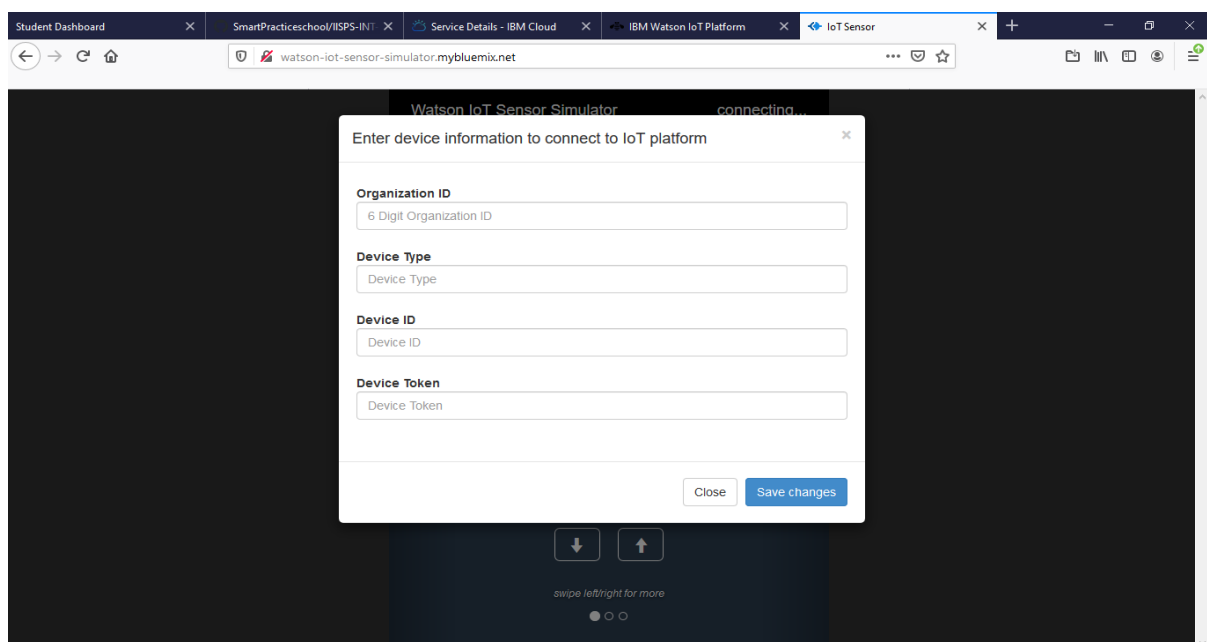
Cookie Preferences

4.2 Creating the API Key



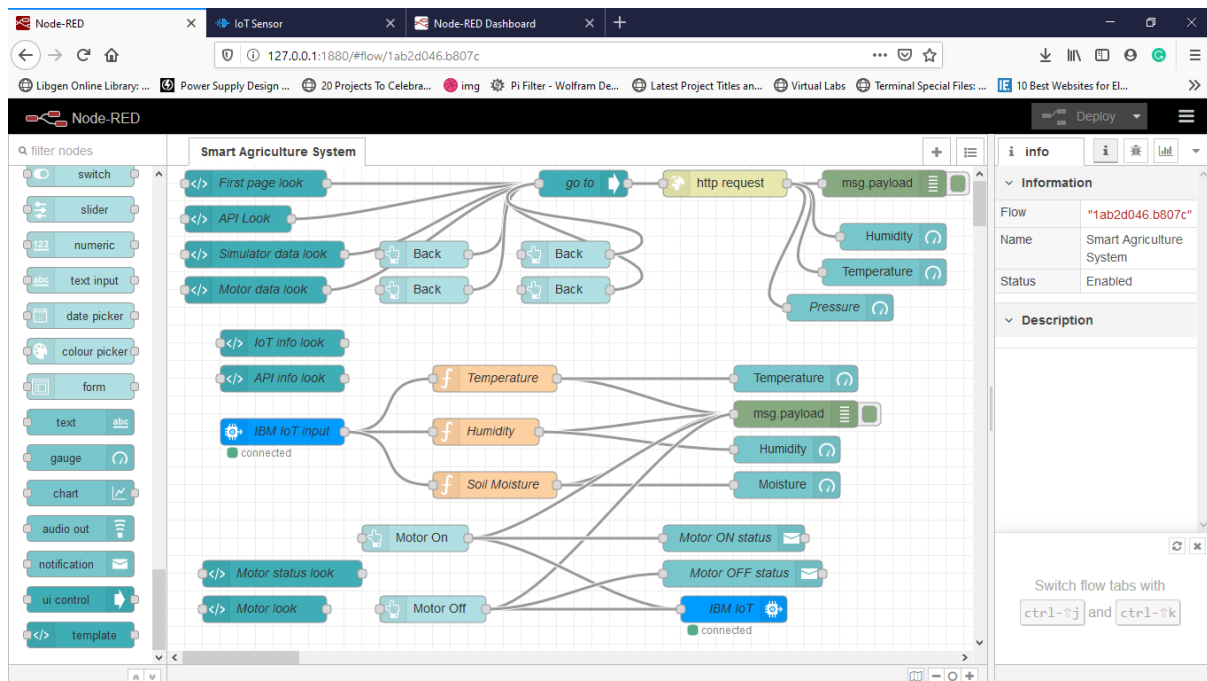
4.3 Setting IoT Sensor Simulator

Enter the details like Organization ID, Device Type, Device ID and Device token.



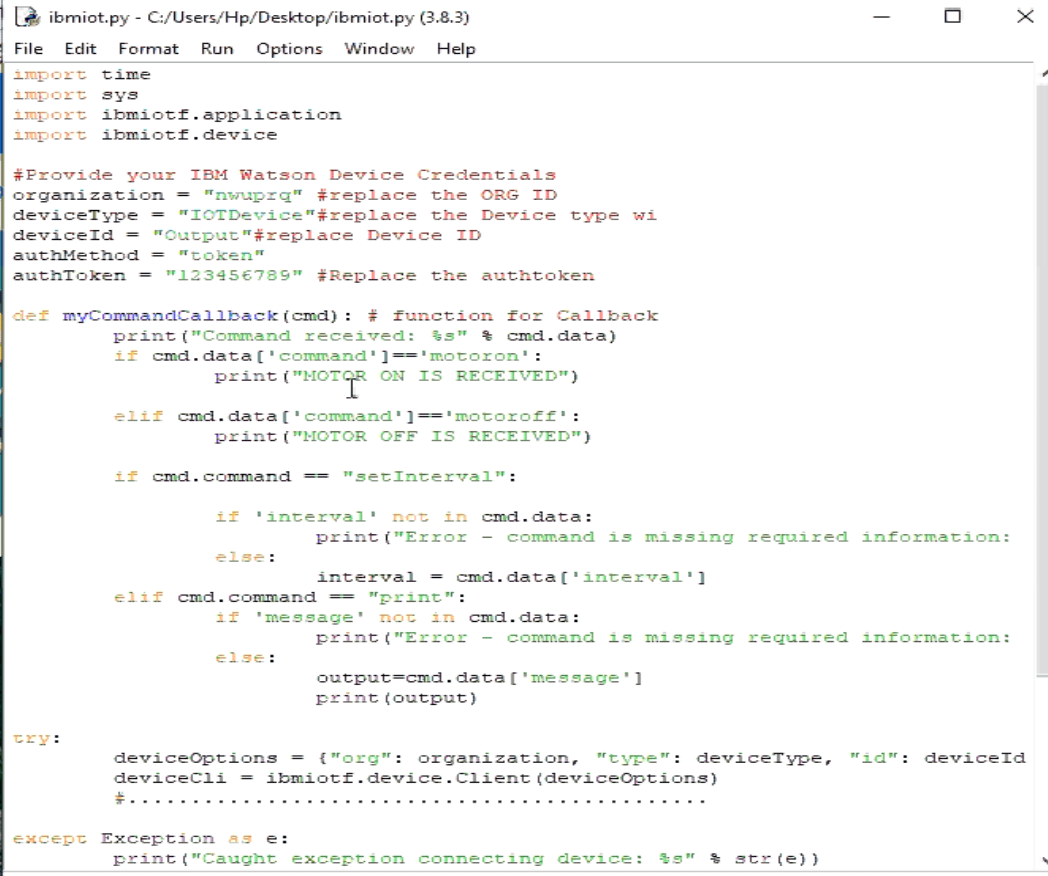
4.4 Creating the Node-Red flow for reading and to display the simulated value and the Weather API data

- The Weather API is obtained from openweather.com.
- The flow below includes the data from IBM sensor and simulator into the UI and the weather data from openweather.org and controls.



4.5 Creating Python code to read the data from the cloud.

The Python code to show whether the motor is on/off is shown below:



```
ibmiot.py - C:/Users/Hp/Desktop/ibmiot.py (3.8.3)
File Edit Format Run Options Window Help

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

#Provide your IBM Watson Device Credentials
organization = "nwuprq" #replace the ORG ID
deviceType = "IOTDevice" #replace the Device type wi
deviceId = "Output" #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']=='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:")
        else:
            interval = cmd.data['interval']

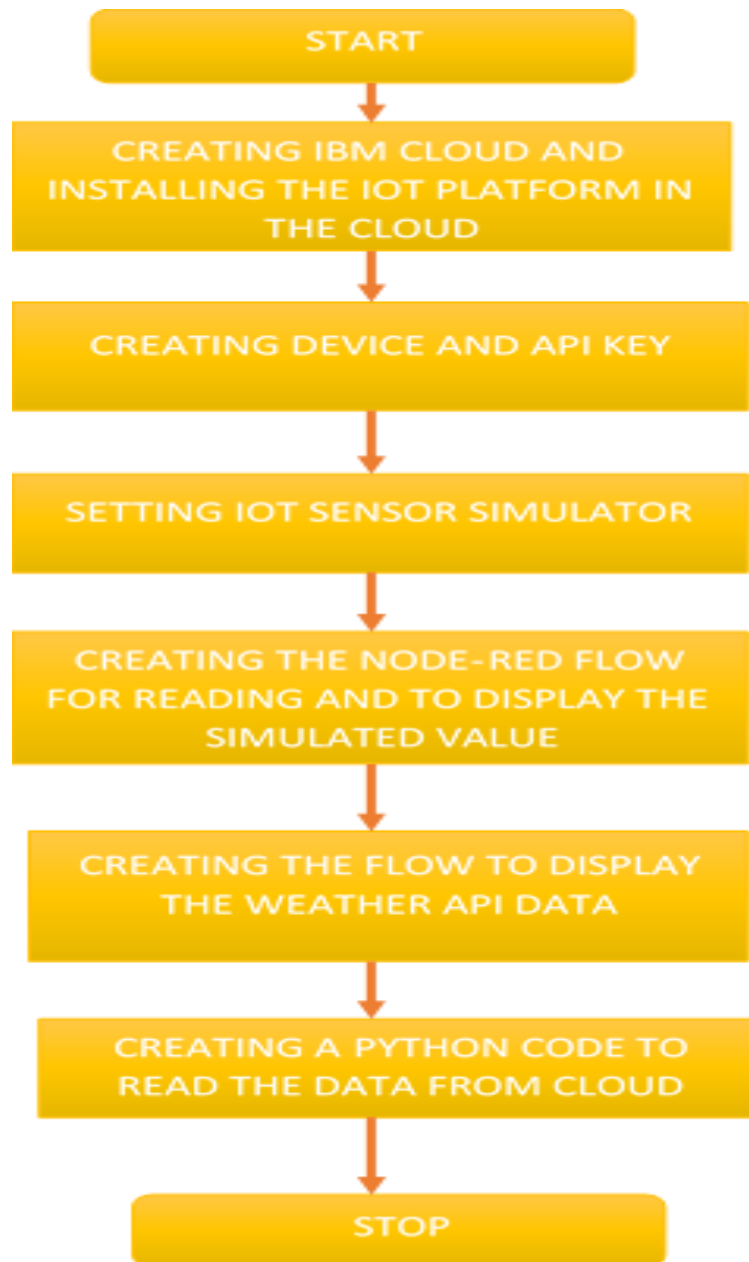
    elif cmd.command == "print":
        if 'message' not in cmd.data:
            print("Error - command is missing required information:")
        else:
            output=cmd.data['message']
            print(output)

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId}
    deviceCli = ibmiotf.device.Client(deviceOptions)
    #.....

except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
```

Ln 52 Col 0

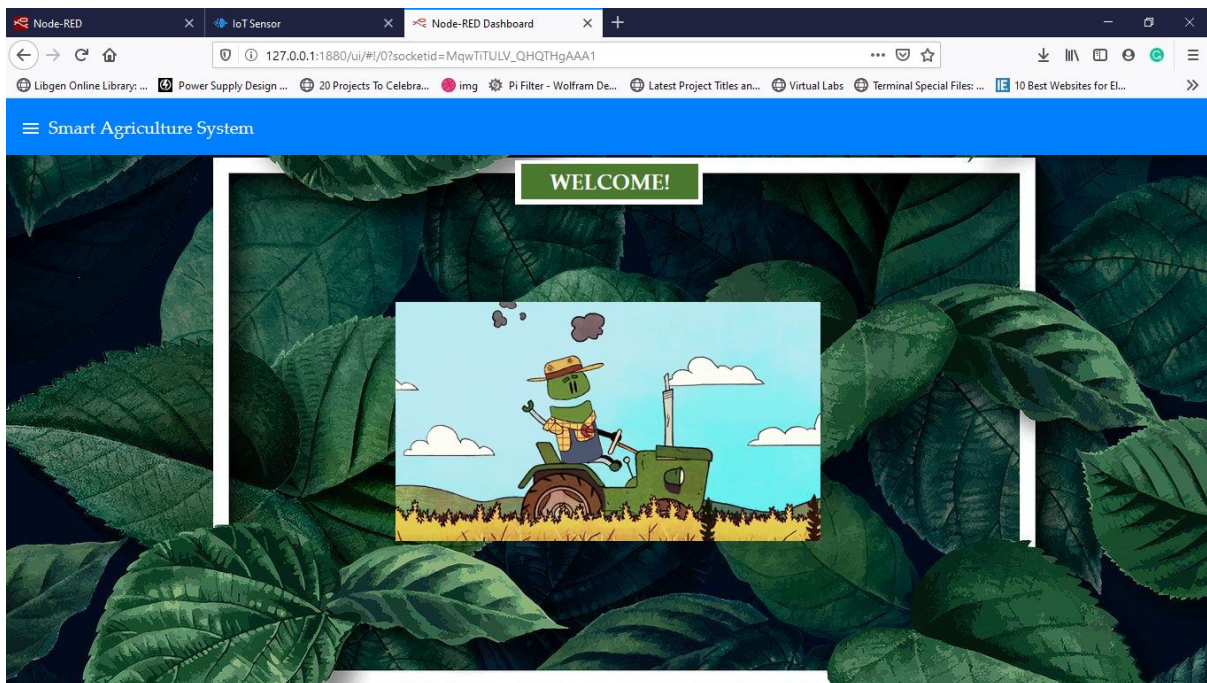
5. FLOW CHART



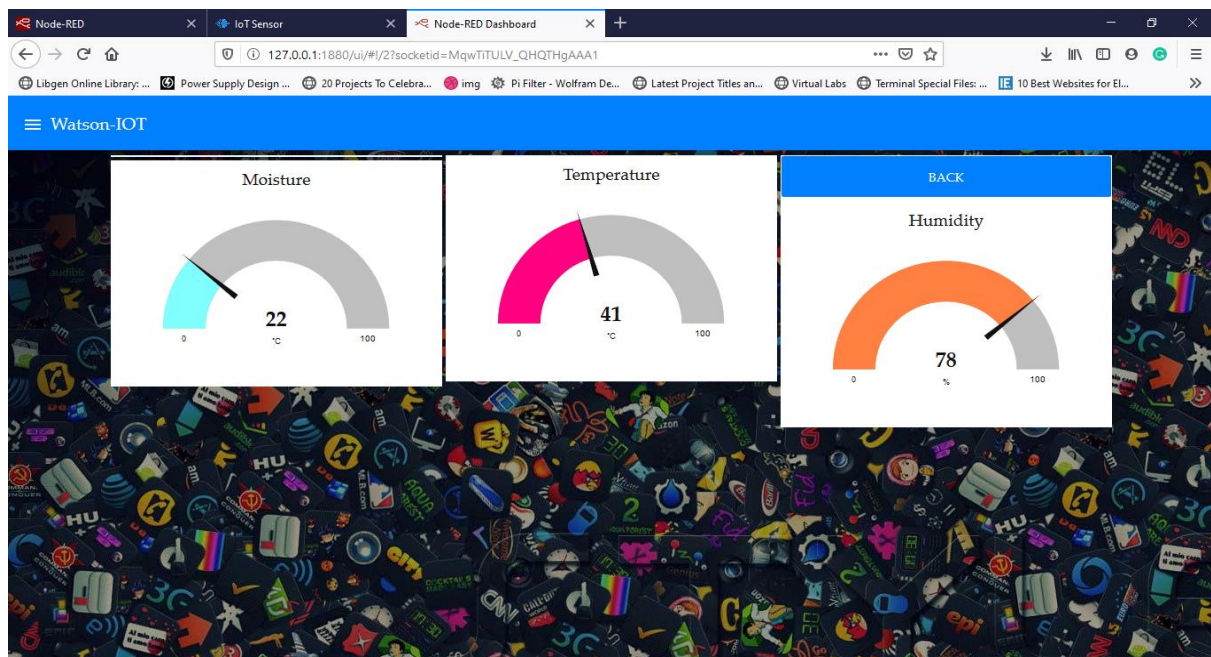
6. RESULT

The yield appeared beneath signifies the temperature, soil moisture and humidity data received from the IOT simulator sensor and open weather API. There are set of buttons on the web application that can be used to control the motor turn them ON/OFF remotely.

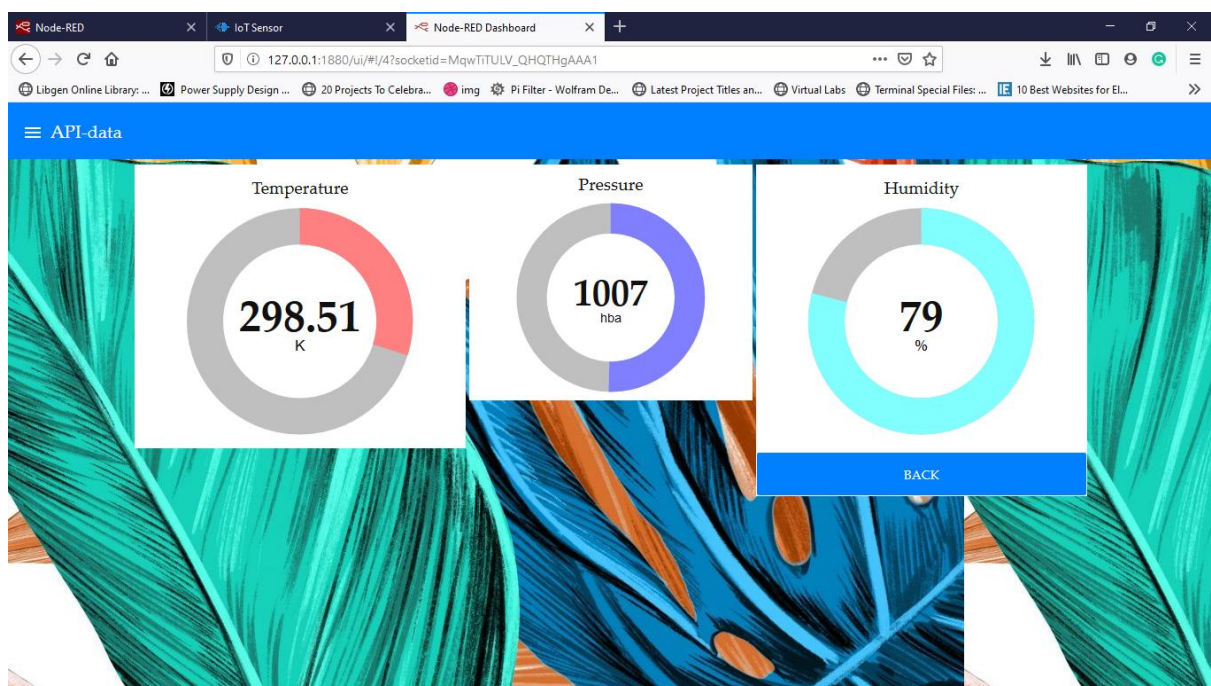
6.1 Home Page



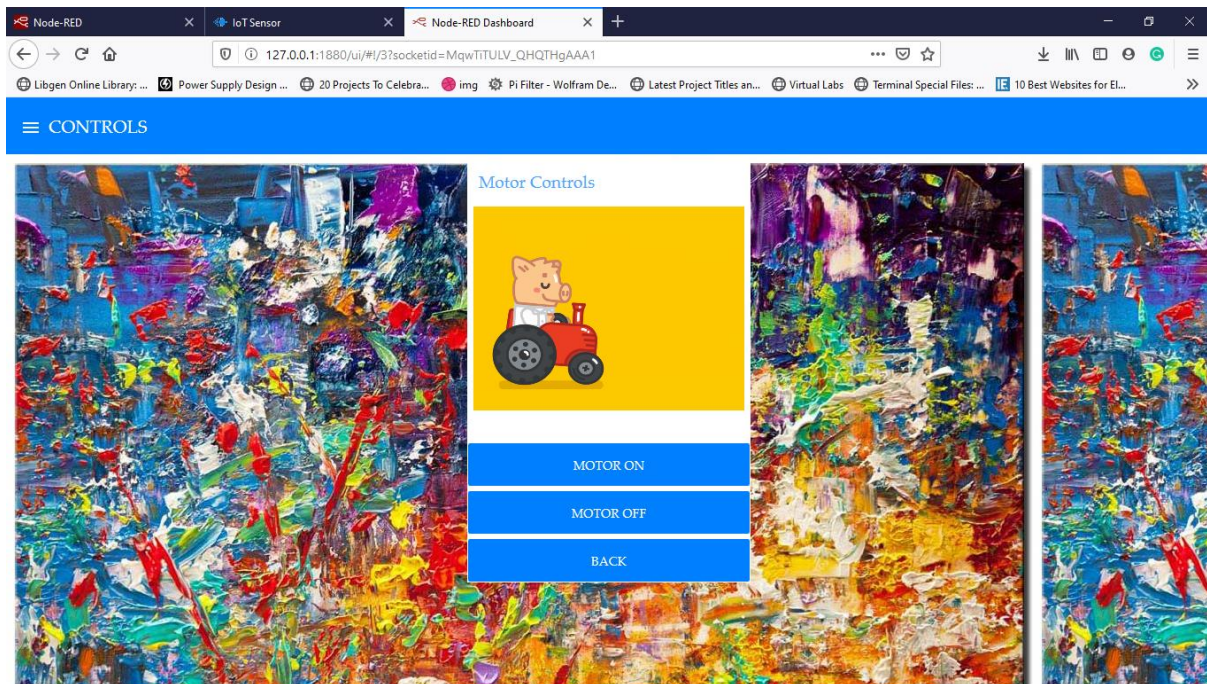
6.2 Data from IBM IOT Sensor:



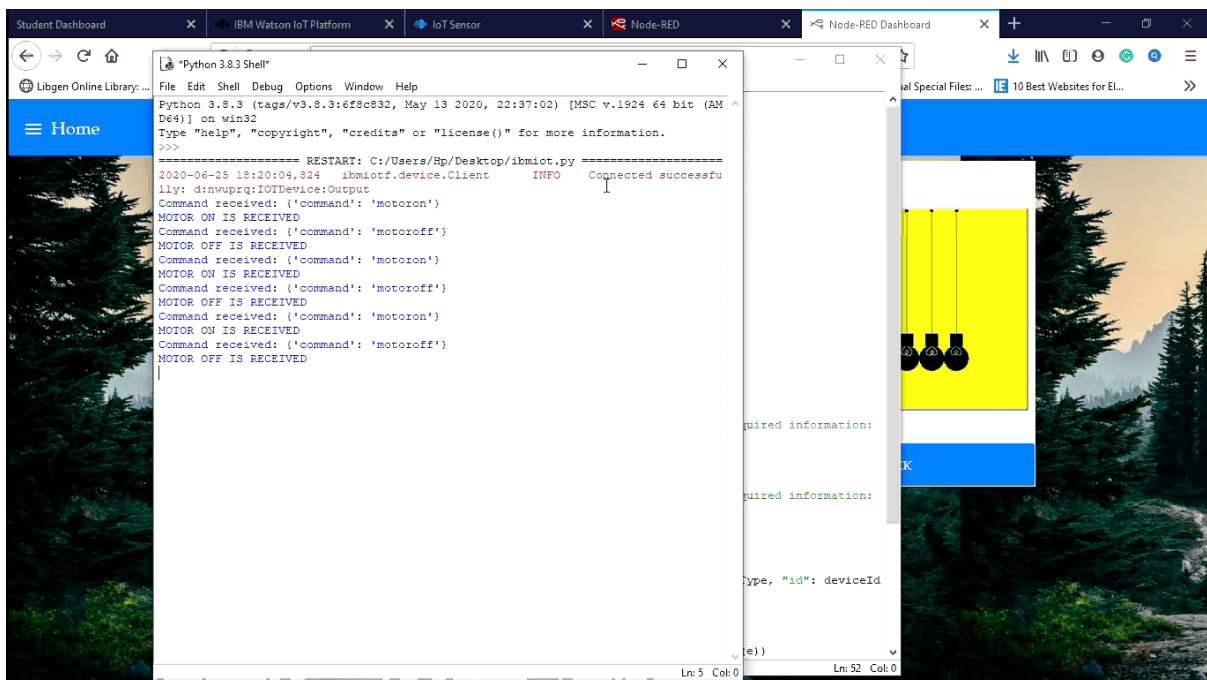
6.3 Live weather data from openweather.org



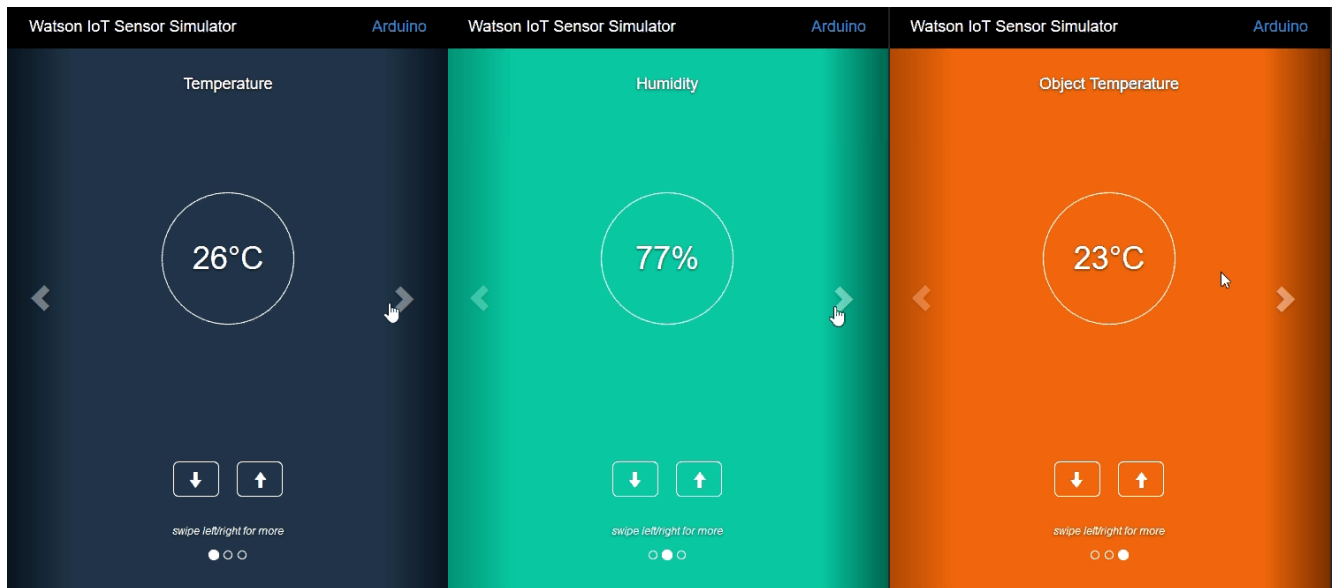
6.4 Controlling of Motor:



6.5 Output in Python



6.6 IBM IOT sensors showing different inputs



7. ADVANTAGES /DISADVANTAGES

7.1 Advantages

- Farms can be monitored and controlled remotely.
- Improves livestock monitoring and management.
- Less labor cost .
- Better standards of living.
- Helpful in knowing real-time weather conditions.
- Increases the quality of production and water conservation.

7.2 Disadvantages

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

8. APPLICATIONS

- This technique can be used in the field of home automation.
- It can also be used in the field where maintaining the process parameters are essential.
- It can also be used in controlling wheel chair for physically challenged people.
- It can be used in hospital to monitor the patient temperature, heart rate etc. During this COVID- 19 situation, it will play a huge role.
- It can also be used in material handling equipment in hospitals.

9. CONCLUSION

- IOT based SMART AGRICULTURE SYSTEM for Live Monitoring of Temperature and Soil Moisture and to control motor remotely has been proposed using Node Red and IBM Cloud Platform.
- The System has high efficiency and accuracy in fetching the live data of temperature and soil moisture.
- The IOT based smart farming System being proposed via this project will assist farmers in increasing the agriculture yield and take efficient care of food production as the system will always provide helping hand to farmers for getting accurate live feed of environmental temperature and soil moisture with more than 99% accurate results.
- Therefore, the project proposes a thought of consolidating the most recent innovation into the agrarian field to turn the customary techniques for water system to current strategies in this way making simple profitable and temperate trimming.

10. BIBLIOGRAPHY

- [https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Service%20\(1\).pdf](https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Service%20(1).pdf)
- <https://smartinternz.com/assets/docs/Sending%20Http%20request%20to%20Open%20weather%20map%20web%20site%20to%20get%20the%20weather%20forecast.pdf>
- IBM cloud reference : <https://cloud.ibm.com/>
- IoT simulator :
<https://watson-iot-sensorsimulator.mybluemix.net/>
- OpenWeather : <https://openweathermap.org>
- Python code reference:
<https://github.com/rachuriharish23/ibmsubscribe>
- IBM Watson IoT Platform :
<https://internetofthings.ibmcloud.com/>

11. APPENDIX

➤ Python Code

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

#Provide your IBM Watson Device Credentials
organization = "nwuprq" #replace the ORG ID
deviceType = "IOTDevice"#replace the Device type
deviceId = "Output"#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']=='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()
```