

# **SMART AGRICULTURE SYSTEM BASED ON IoT**

## **1. Introduction**

### **Overview:**

India is very much dependent on the 'Agriculture'. After all, India is agri-based economy. This is the project from the motivation of the farmers working in the farm lands are solely dependent on the rains and bore wells for irrigation of their land. In recent times, the farmers have been using irrigation technique through the manual control in which the farmers irrigate the land at regular intervals by turning the water-pump ON/OFF when required. This project involves building a smart IOT based agriculture system to monitor the weather conditions and soil conditions and help the farmer to gain better yield. This will be achieved by using IBM Cloud , IBM Watson IOT Platform and Open Weather API and Python Programming for interacting with the system. It provides real time data like temperature, humidity and soil temperature and help farmers to farm smartly.

### **Purpose**

The purpose of this project is to help farmers to avoid over irrigation and under irrigation of the farm. It gives information about soil, temperature, weather and humidity so the farmer can know the conditions of crops even if he is away from the field. It uses various technologies like IBM Cloud and IOT in order to make the farming affordable.

## **2. Literature Survey**

### **Existing Problem**

The Indian farming is on the hitch because of the limited technical know how of the best and efficient agricultural practices and moreover they are still dependent on conventional methods of agriculture that leads to lesser productivity of crops. Apart from these issues scarcity of resources also adds up in their problem causing hindrance or stopping framers from cultivating and hence Indian economy is also additionally getting influenced to large extent as most of the fruitful lands of the nation are being

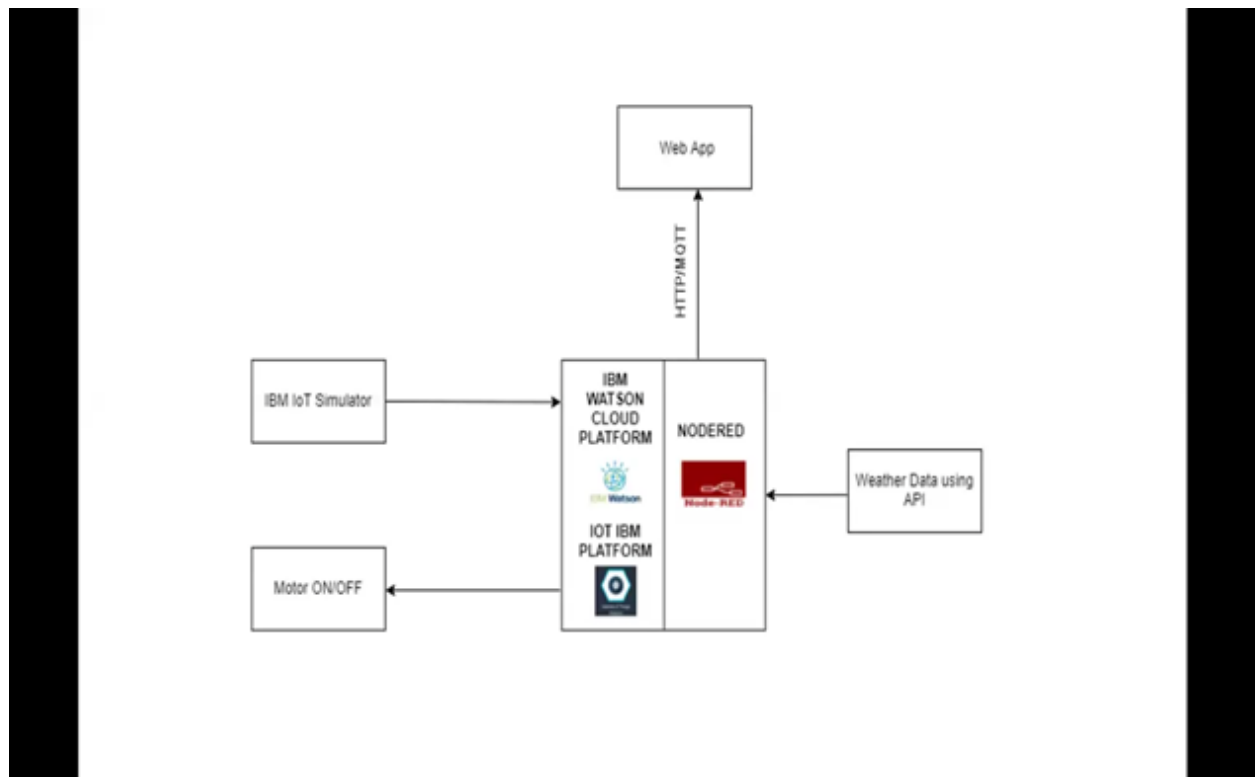
destroyed that forms the vital part of GDP.

## Proposed Solution

By using IOT and Cloud technology, we propose a solution in which we use temperature, humidity and soil sensors and send these info to cloud. We also make web app to interact with users and helps them to control the irrigation. With the sensors information, farmer controls the irrigation system. With the help to this, the productivity of crops can be maximized at minimal cost. This also reduces burden of taking up of heavy loans on farmers which they have incurred on themselves in order to sustain their livings or to get good yields of their crops.

## 3. Theoretical Analysis

### Block Diagram



## Hardware/Software Designing

- Create IBM Cloud account

IBM Cloud

Already have an IBM Cloud account? [Log in](#)

### Create an account

1. Account information
2. Verify email
3. Personal information

Email

Enter an email address.

Password

Next

Create account

© Copyright IBM Corp. 2014, 2020. All rights reserved.

## Build for free on IBM Cloud

**Develop for free, no credit card required**  
Apps, AI, analytics, and more. Build with 40+ Lite plan services at no cost to you – ever.

**Access the full catalog at your fingertips**  
Upgrade your account and unlock 190+ unique offerings, plus get a \$200 credit to use with any offering you want.

Catalog Docs

Cookie Preferences

Type here to search

11:41 09-06-2020

- Install Node-red locally on your system

First you have to install Node JS from its official website.

Installing Node-RED as a global module adds the command **node-red** to your system path. Then execute the following at the command prompt:

`npm install -g --unsafe-perm node-red.`

## Create IBM Watson IOT Device

Click on the button **"Add Device"** on right top corner to create Device Type and New Device.

The screenshot shows the 'Browse Devices' page in the IBM Watson IoT Platform. The page has a dark blue header with the 'Add Device' button in the top right corner. A sidebar on the left contains navigation icons. The main content area has a 'Browse Devices' title and two buttons: 'All Devices' (selected) and 'Diagnose'. Below this is a table of devices. The table has columns: Device ID, Status, Device Type, Class ID, Date Added, Descriptive Location, Added By, and Device Class. Two devices are listed: 'Device01' and 'SmartDevice', both with a status of 'Disconnected' and type of 'IOTsample'. At the bottom right of the table, it says '1 of 1 page'.

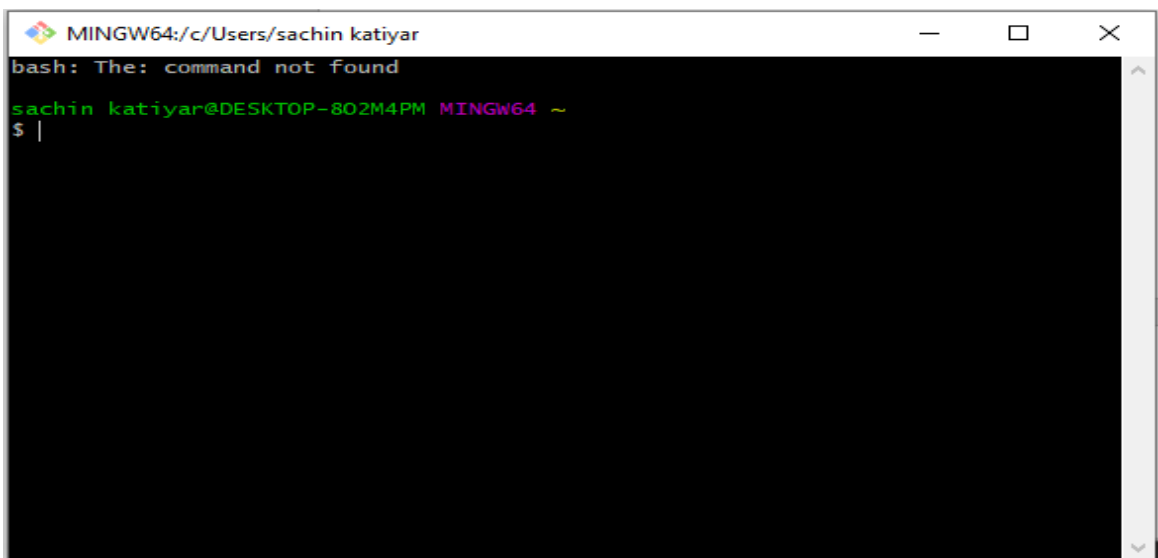
Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By	Device Class
Device01	Disconnected	IOTsample	Device	May 19, 2020 3:39 PM		17ucs135@inmit.ac.in	
SmartDevice	Disconnected	IOTsample	Device	Jun 7, 2020 4:18 PM		17ucs135@inmit.ac.in	

The screenshot shows the 'API Keys' page in the IBM Watson IoT Platform. The page has a dark blue header with the 'Generate API Key' button in the top right corner. A sidebar on the left contains navigation icons, with 'API Keys' highlighted. The main content area has a title 'API Keys' and a search bar. Below this is a table of API keys. The table has columns: Description, Role, and Expires. One API key is listed: '9-qapoukyva7' with a role of 'Standard Application'. At the bottom right of the table, it says '1 result'.

Description	Role	Expires
9-qapoukyva7	Standard Application	-

To generate API key, click on the "**Apps**" tab on left toolbar. Click on "**Generate API key**" on top right and generate key with the role as "**Standard Application**". Copy Api key and Authentication key/Token in clipboard as they are not accessible afterwards.

- Install Python 3 IDLE from [here](#) and install it with default settings.
- **Install GIT**



```
MINGW64:/c/Users/sachin katiyar
bash: The: command not found
sachin katiyar@DESKTOP-802M4PM MINGW64 ~
$ |
```

## 4. Experimental Investigations

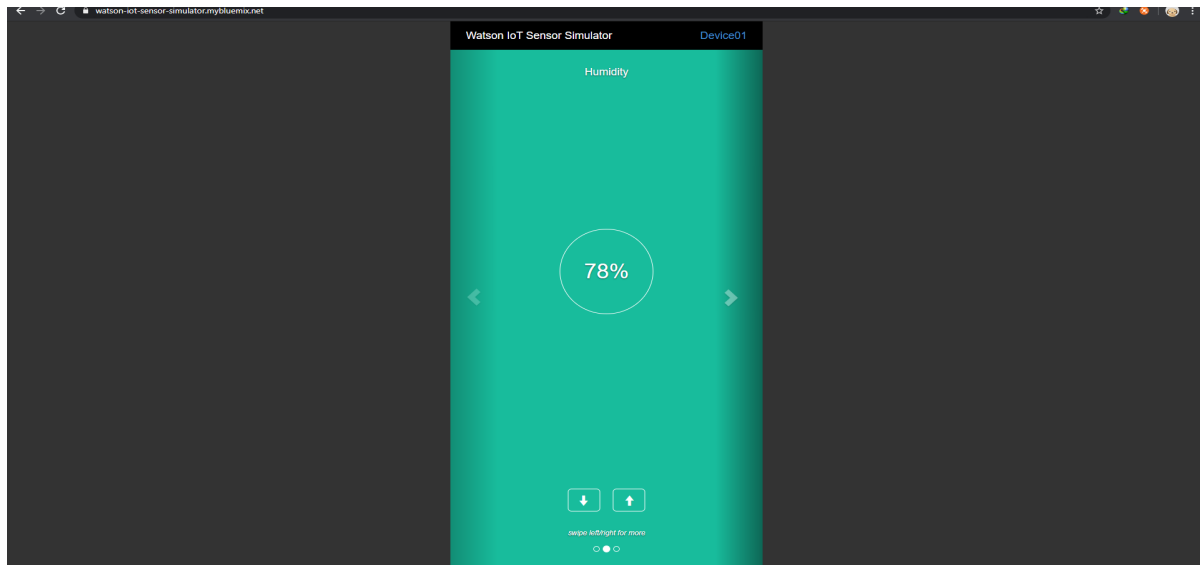
### Connect the IOT Device to Simulator

The IOT Sensor requires the following information:

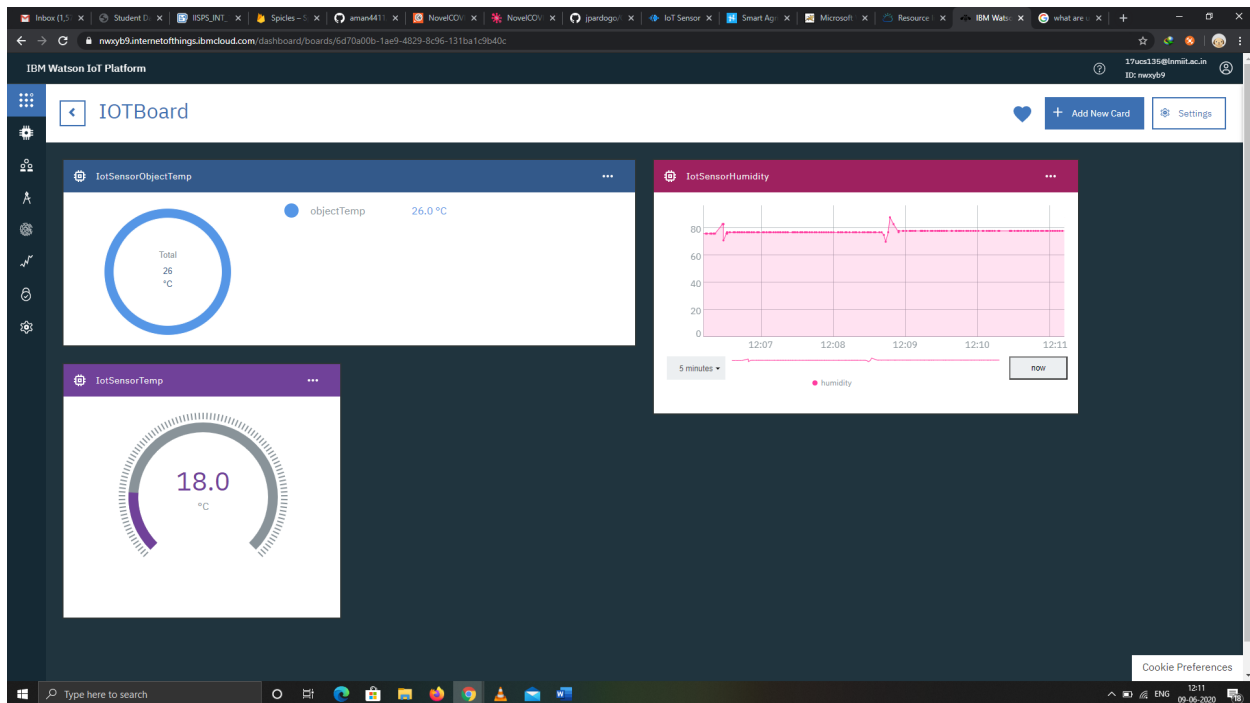
- 1.Device organization
- 2.Device Type
- 3.Device Id
- 4.Device Token

Once given all the required instructions, the sensor is connected to the device which we created on IOT Platform.

If connected, device color will be shown in blue on the top right corner.



**Boards** and **cards** in the Watson Internet of Things platform, you can build your own Custom Dashboard. You can use the boards as the landing page of interest and then make use of the cards within them, to: Create visualization charts for the real time data from your devices.



Also there are many more tabs like **Members**, **Access Management** through which you can work with your team and assign roles and access permissions to them. And can add as many as team members you want.

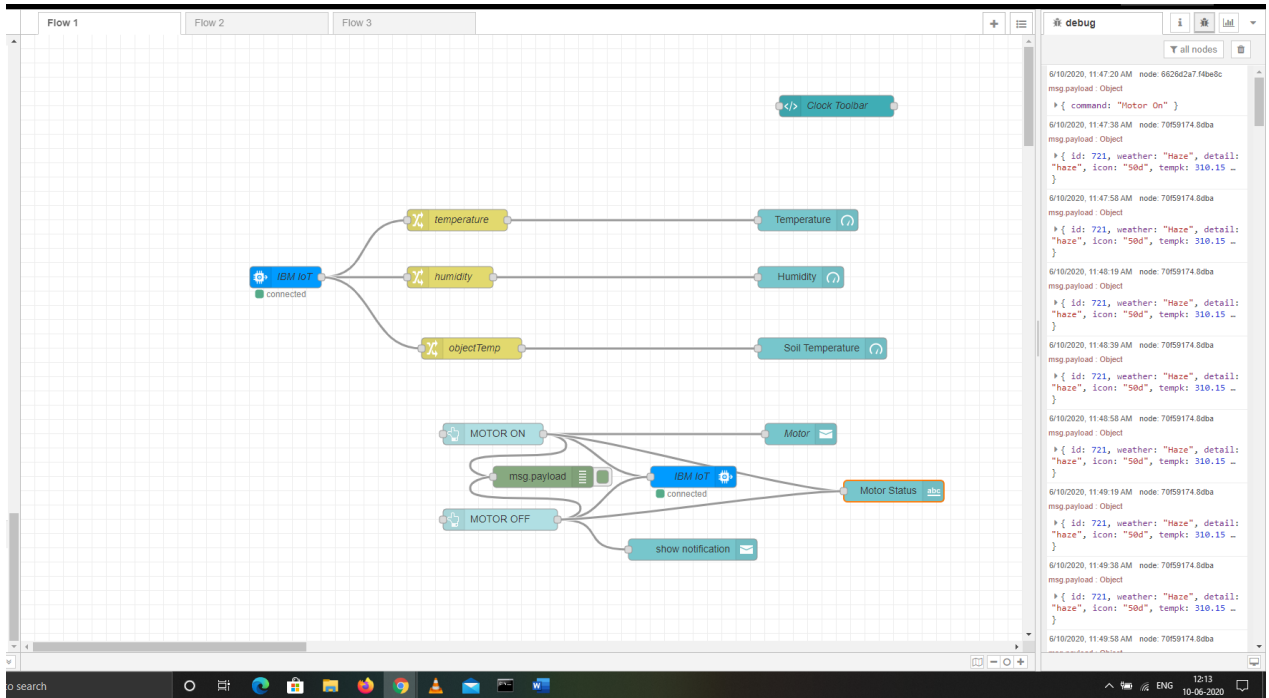
### Connecting the OpenWeather Api

- Create an account in [openweathermap.org](https://openweathermap.org)
- Go to API marketplace and register a API key
- Select "**By city Name**" in API options
- Copy and save the API call to be used later

### Create Web app using Node-red

- Create flows in node-red according to your need.
- We install IBM IOT node package. For this,  
**manage palette-->install-->search for required nodes**
- For creating UI in Node-red, we use IBM IOT input node and configure it by adding API key and authentication token, to take the data from IOT sensor. Connect IBM IOT node to debug node to check the status in **Debug** tab.
- To separate temperature and humidity, you have to add function node and pass **msg.payload.d.temperature** to separate values.

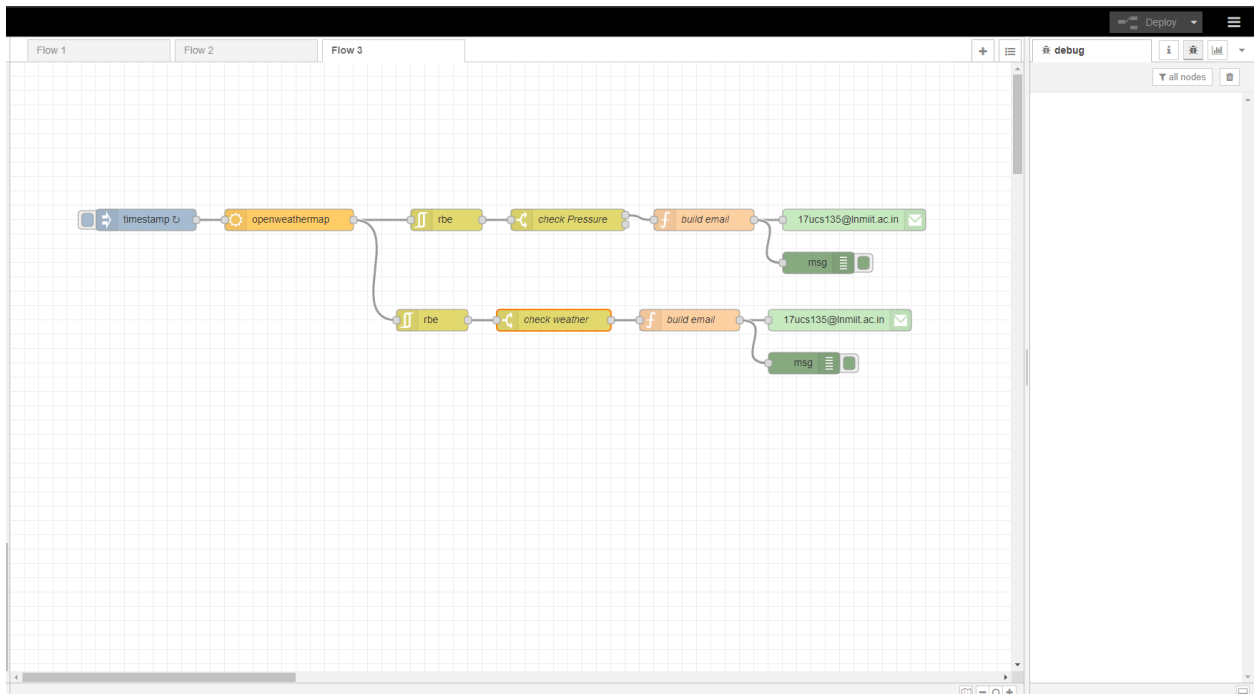
- Install the dashboard node from the manage pallet to create a UI to display temperature and humidity values in the Dashboard.
- Insert two buttons for controlling motor, i.e, MOTOR OFF and MOTOR ON, and connect it with IBM IOT output node and show status of motor.
- Also, I have added Clock Toolbar to show the time in the header section of web UI.



- In Flow 2, add http request node to view weather data in UI and configure it with OpenWeather API.
- Install OpenWeather node for retrieving weather data of next 3 hours and 6 hours forecasting so that the farmer should be aware of upcoming weather.
- Also I have added notification node to show the suitable air pressure range to the farmer.



In Flow 3, I have added the mechanism to send the email alerts to the user if weather is cloudy or rainy.

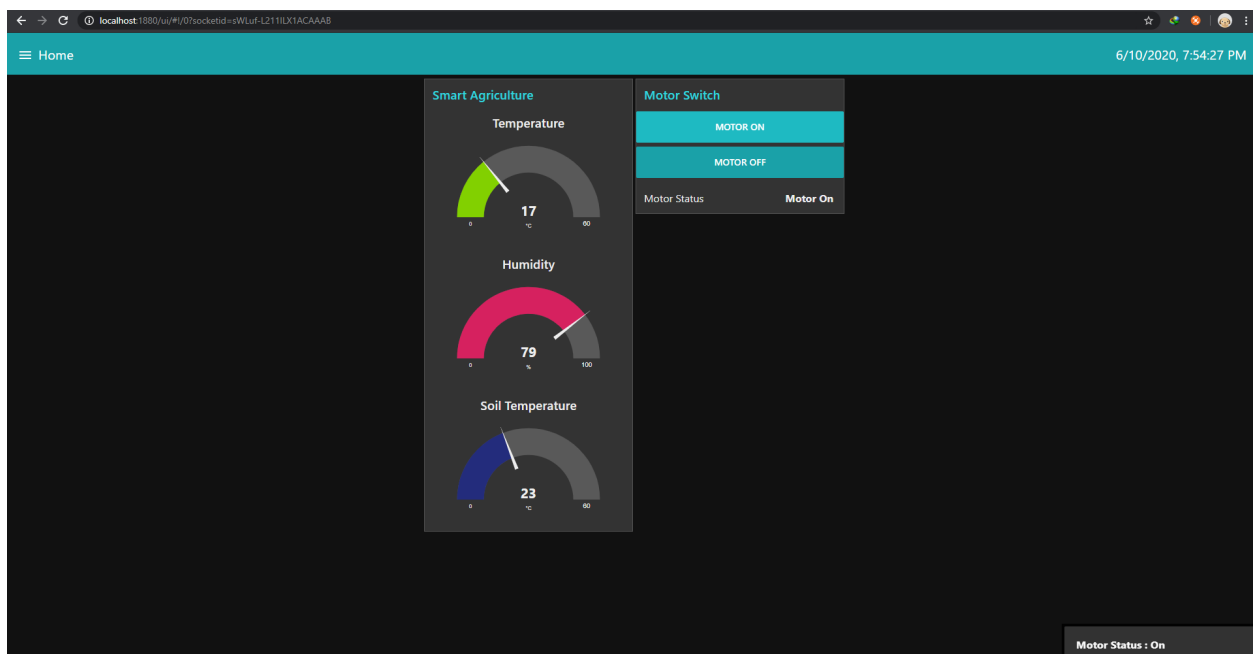


## Interacting with device through Python Code

We use Python code to interact with the cloud and IOT device.

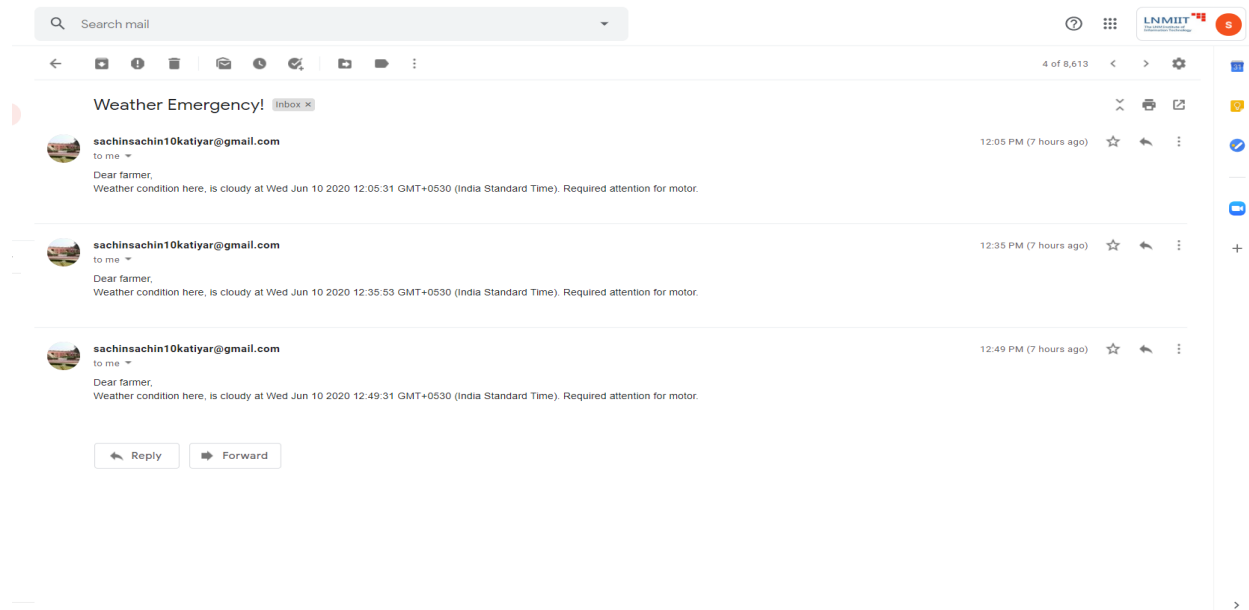
## 5. Results

UI for Web app:





Also we get alert email if weather gets cloudy or rainy.



## 6. Advantages /Disadvantages

Following are the benefits of Smart Agriculture

- It allows farmers to maximize yields using minimum resources such as water, fertilizers, seeds etc.
- Solar powered and mobile operated pumps save cost of electricity.
- Smart agriculture use drones and robots which helps in many ways. These improves data collection process and helps in wireless monitoring and control.
- It is cost effective method.
- It delivers high quality crop production.

Following are the drawbacks of Smart Agriculture:

- The smart agriculture needs availability of internet continuously. Rural part of most of the developing countries do not fulfill this requirement. Moreover internet connection is slower.
- The smart farming based equipments require farmers to understand and learn the use of technology. This is major challenge in adopting smart agriculture farming at large scale across the countries.

## **7. Applications**

With the introduction of Industrial IoT in Agriculture, far more advanced sensors are being utilized. The sensors are now connected to the cloud via cellular/satellite network. Which lets us to know the real-time data from the sensors, making decision making effective.

With the help of sensors and inter connectivity, the Internet of Things in Agriculture has not only saved the time of the farmers but has also reduced the extravagant use of resources such as Water and Electricity. It keeps various factors like humidity, temperature, soil etc. under check and gives a crystal clear real-time observation.

## **8. Conclusion**

IoT enabled agriculture has helped implement modern technological solutions to time tested knowledge. This has helped bridge the gap between production and quality and quantity yield. Data Ingested by obtaining and importing information from the multiple sensors for real time use ensures swift action and less damage to the crops. With seamless end to end intelligent operations and improved business process execution, product gets processed faster and reaches supermarkets in fastest time possible.

## **9. Future Scope**

The future scope of this project could be including IBM IOT sensor and then collecting, processing and storing the data on cloud server. This would make the predicting and analyzing processes more accurate.

Another direction in which smart farming is headed involves intensively controlled indoor growing methods. The OpenAG Initiative at MIT Media Lab uses "personal food computers" (small indoor farming environments that monitor/administrate specific growing environments) and an open source platform to collect and share data. The collected data is termed a "climate recipe" which can be downloaded to other personal food computers and used to reproduce climate variables such as carbon dioxide, air temperature, humidity, dissolved oxygen, potential hydrogen, electrical conductivity, and root-zone temperature. This allows users very precise control to document, share, or recreate a specific environment for growing and removes the element of poor weather conditions and human error. It could also potentially allow farmers to induce drought or

other abnormal conditions producing desirable traits in specific crops that wouldn't typically occur in nature.

## 10. Bibliography

[www.openweathermap.org](http://www.openweathermap.org)

[cloud.ibm.com](http://cloud.ibm.com)

[www.dzone.com](http://www.dzone.com)

[www.biz4intellia.com](http://www.biz4intellia.com)

## 11. Appendix

### Source Code

```
1
2 import time
3 import sys
4 import ibmiotf
5 import ibmiotf.application
6 import ibmiotf.device
7
8 #Provide your IBM Watson Device Credentials
9 organization = "nwxyb9" # repalce it with organization ID
10 deviceType = "IOTsample" #replace it with device type
11 deviceId = "SmartDevice" #repalce with device id
12 authMethod = "token"
13 authToken = "sachin10"#repalce with token
14
15 def myCommandCallback(cmd): # function for Callback
16     print("Command received: %s" % cmd.data)
17     if cmd.data['command']=='Motor On':
```

```

18         print("MOTOR ON IS RECEIVED")
19
20     elif cmd.data['command']=='Motor Off':
21         print("MOTOR OFF IS RECEIVED")
22
23     '''
24     if cmd.command == "setInterval":
25
26         if 'interval' not in cmd.data:
27             print("Error - command is missing
required information: 'interval'")
28         else:
29             interval = cmd.data['interval']
30     elif cmd.command == "print":
31         if 'message' not in cmd.data:
32             print("Error - command is missing
required information: 'message'")
33         else:
34             output=cmd.data['message']
35             print(output)
36     '''
37     data = {"Command" : cmd.data['command']}
38     success = deviceCli.publishEvent("event", "json",
data, qos=0, on_publish=myOnPublishCallback)
39     if not success:
40         print("Not connected to IoT")
41
42     myCommandCallback.has_been_called = True
43 try:
44     deviceOptions = {"org": organization, "type":
deviceType, "id": deviceId, "auth-method": authMethod,
"auth-token": authToken}
45     deviceCli = ibmiotf.device.Client(deviceOptions)
46     #.....
47
48 except Exception as e:

```

```

49     print("Caught exception connecting device: %s" % str(e))
50     sys.exit()
51
52 # Connect and send a datapoint "hello" with value "world"
    into the cloud as an event of type "greeting" 10 times
53 deviceCli.connect()
54
55 while True:
56
57     '''
58     T=50;
59     H=32;
60     ot=45
61
62     data = {'d':{ 'Temperature' : Status, 'Humidity':
H,'objTemp':ot }}
63     #Send Temperature & Humidity to IBM Watson
64     '''
65     myCommandCallback.has_been_called = False
66
67     Status = "Sensor is On"
68     #cmd.data['command'] = "Rest"
69     #Send Status to IBM Watson
70
71     data= {'Status' : Status}
72     #data2 = {'Command RECEIVED' : cmd.data['command']}
    }
73     #print data
74     def myOnPublishCallback():
75         print (data, "to IBM Watson")
76         #print (data2, "to IBM Watson")
77
78     success = deviceCli.publishEvent("event", "json",
data, qos=0, on_publish=myOnPublishCallback)
79     if not success:
80         print("Not connected to IoT")

```

```
82
83     deviceCli.commandCallback = myCommandCallback
84     if myCommandCallback.has_been_called == True :
85         print("call made")
86
87
88 # Disconnect the device and application from the cloud
89 #deviceCli.disconnect()
90
```