Smart Agriculture System Based on IOT

INTRODUCTION

1.1 Overview

- Farmers require to be physically present at their fields at all times to serve the requirements like watering the crops according to weather conditions.
- Using an IOT device connected to an application, many of these activities can be automated, thereby saving time, effort and money.

1.2 Purpose

- The purpose of this project is to develop an IOT app that display concerned details of a farm to the farmer and also control the working of connected devices.

LITERATURE SURVEY

2.1 Existing problem

- With increasing demand in agricultural produce across the globe, farmers find it difficult to to carry out trade in the city and serve the needs of crops at the same time.
- This is almost impossible to be done in case of a large plot of land and eventually lead to loss.
- Thus, there is a dire need to automate certain processes and keep the farmer informed of his plot even when he is away.

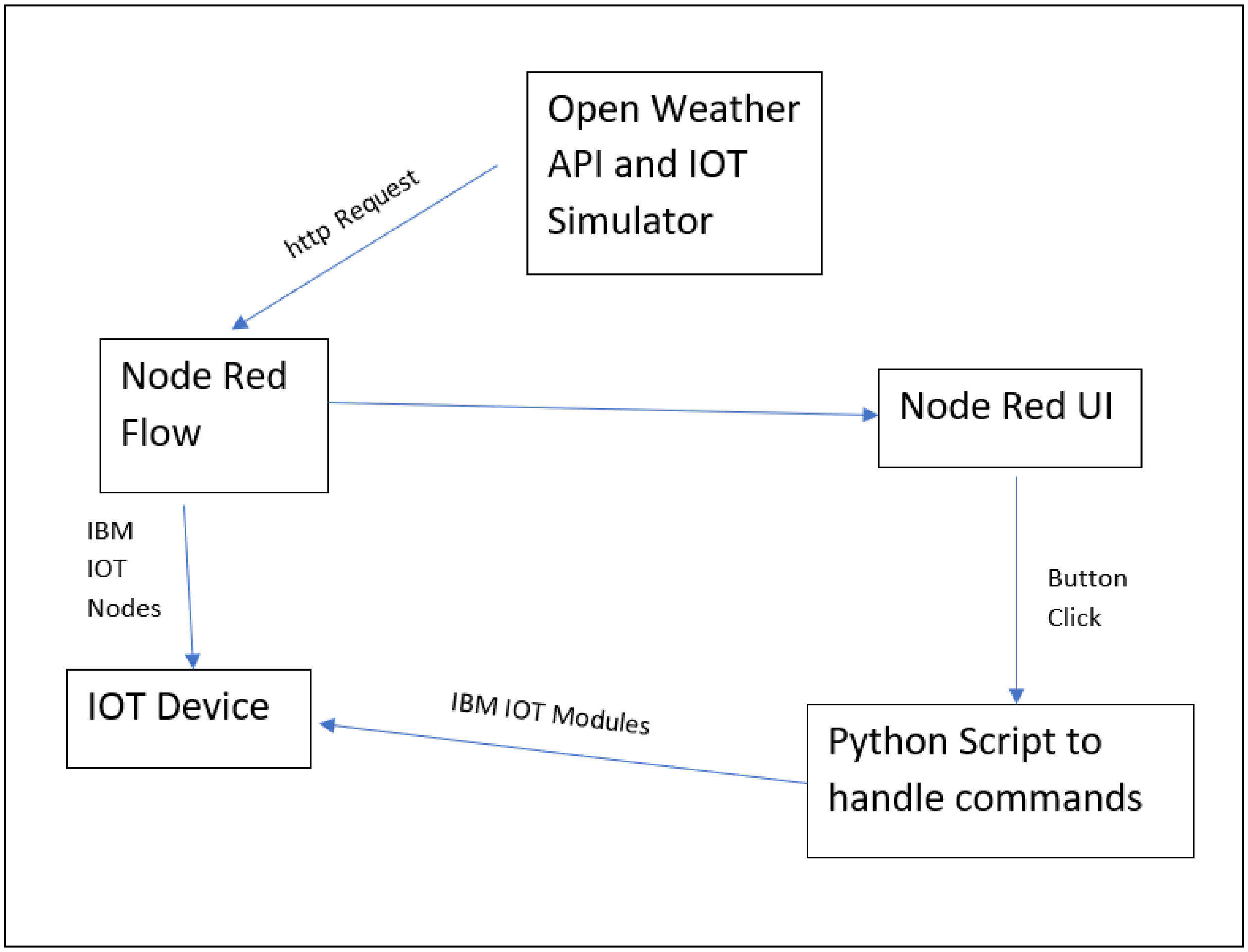
2.2 Proposed solution

- 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 realtime weather forecasting data by using external

platforms like Open Weather API.
- Farmer is 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 parameters he can water his crop by controlling the motors using the mobile application.
- Even if the farmer is not present near his crop he can water his crop by controlling the motors using the mobile application from anywhere.

THEORITICAL ANALYSIS

3.1 Block diagram

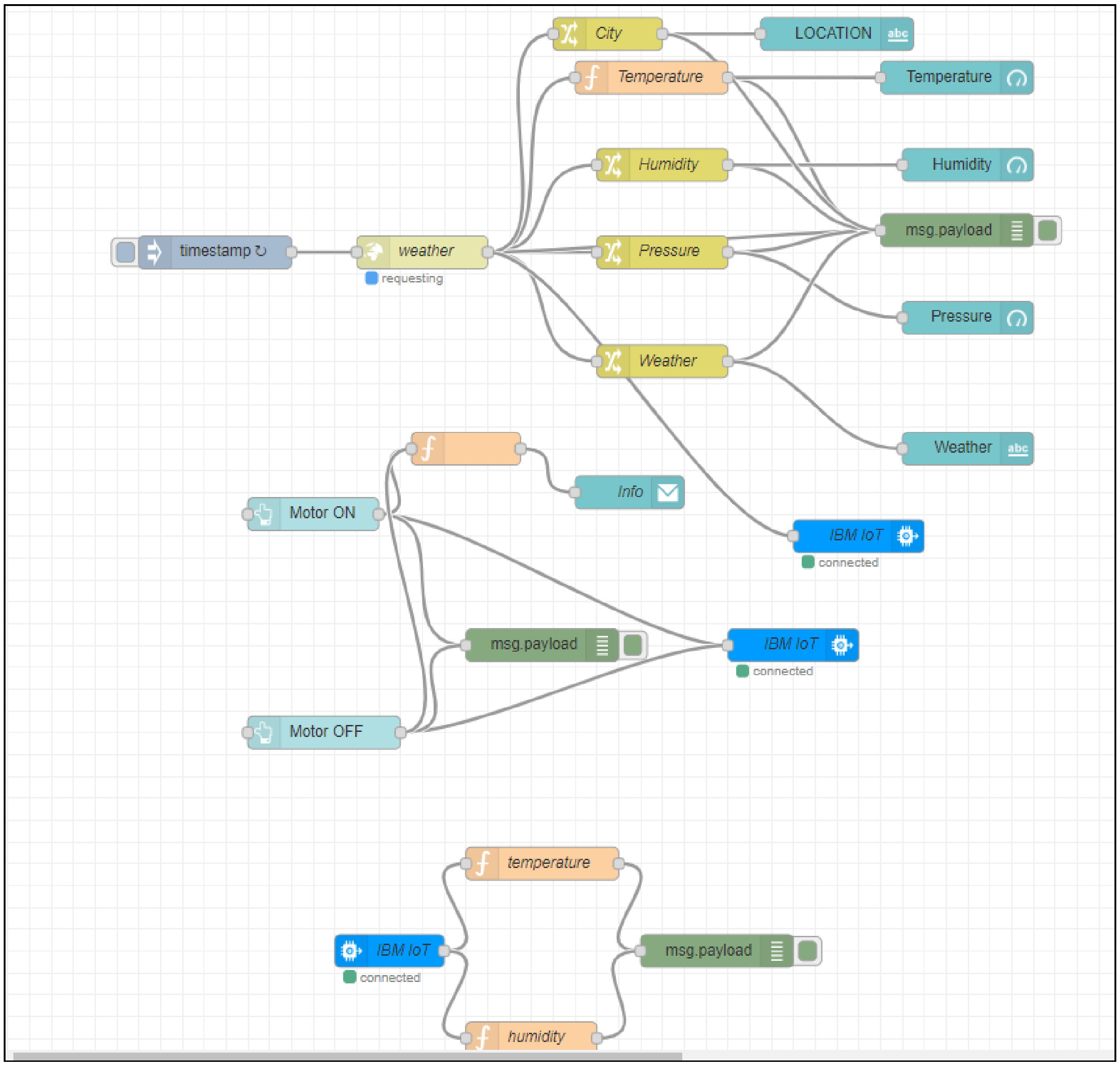


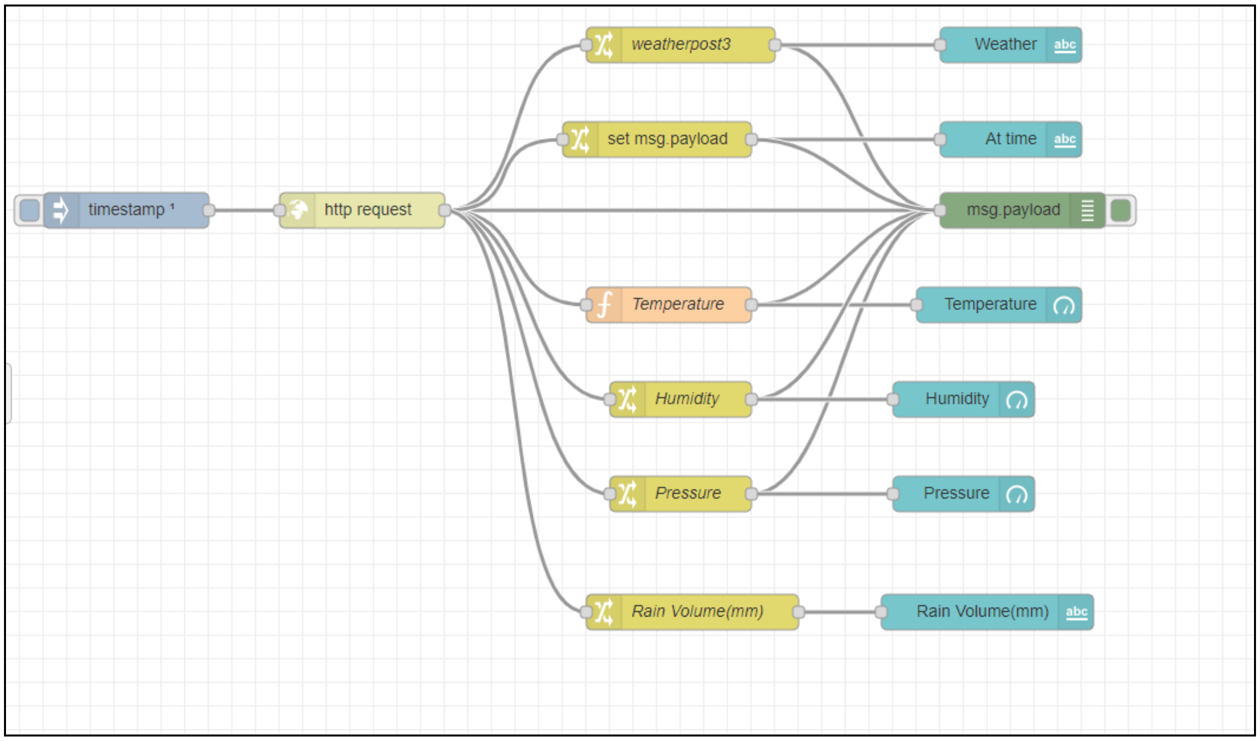
3.2 Hardware / Software designing

- IBM Watson IOT Platform is used to create a service and devices that serve as the devices which would have been physically used otherwise.

- Node Red is used to create a flow of commands which is deployed as a user interface(UI) on the Node Red Dashboard.
- Current weather data is obtained from the Open Weather API using an http request which is continuously displayed in the UI.
- The farmer can control Motor ON/OFF from the app through buttons. These buttons send commands to the IOT device using a python script.
- The Online IoT simulator is used for getting the Temperature,Humidity and Soil Moisture values.

★ *Completed Flow in Node-Red*



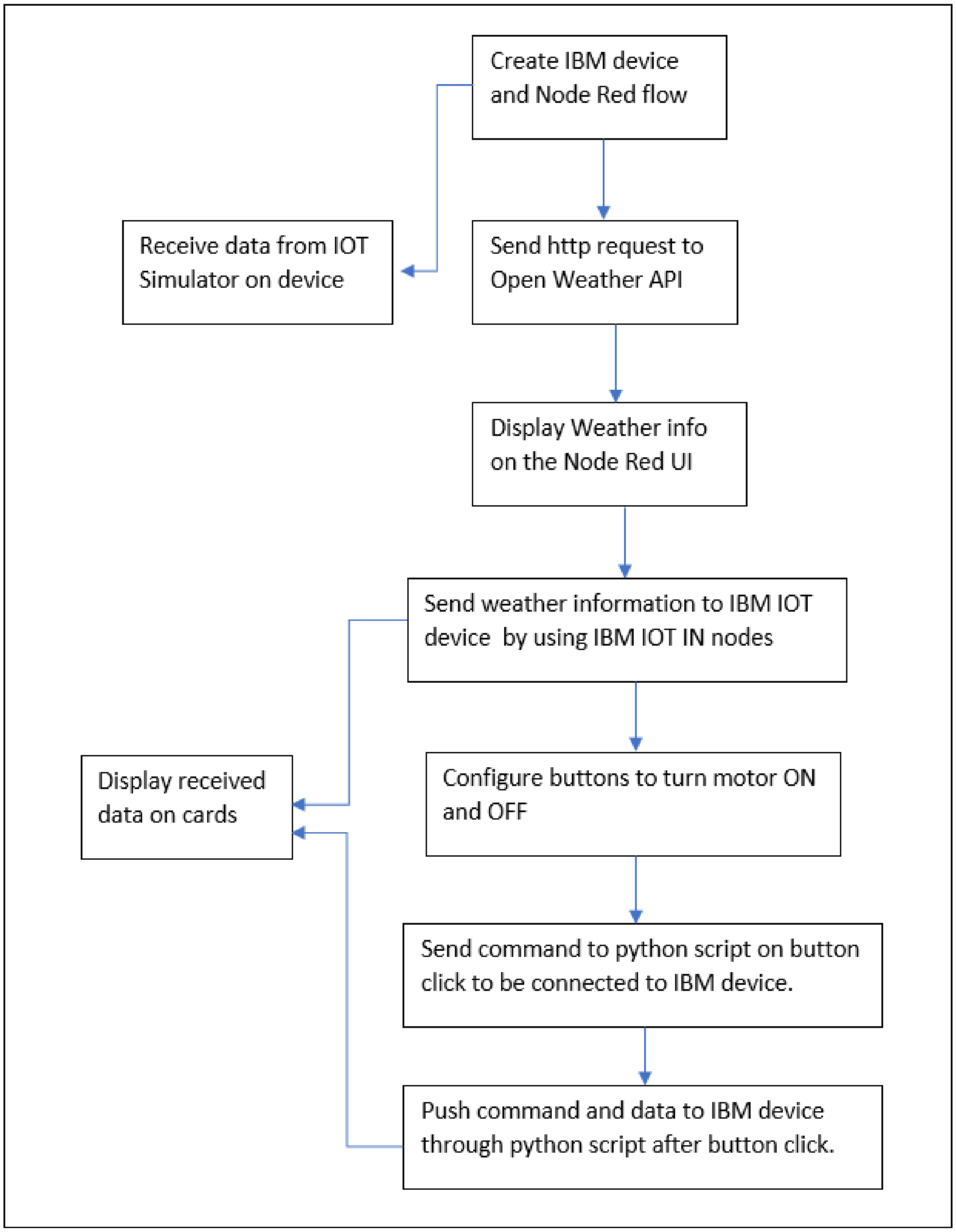


Flow for forcasting weather data.

EXPERIMENTAL INVESTIGATIONS

- Working of the app is verified using the Online IOT Simulator and the OpenWeather API.
- Data is visible on the IOT Platform Cards as well.

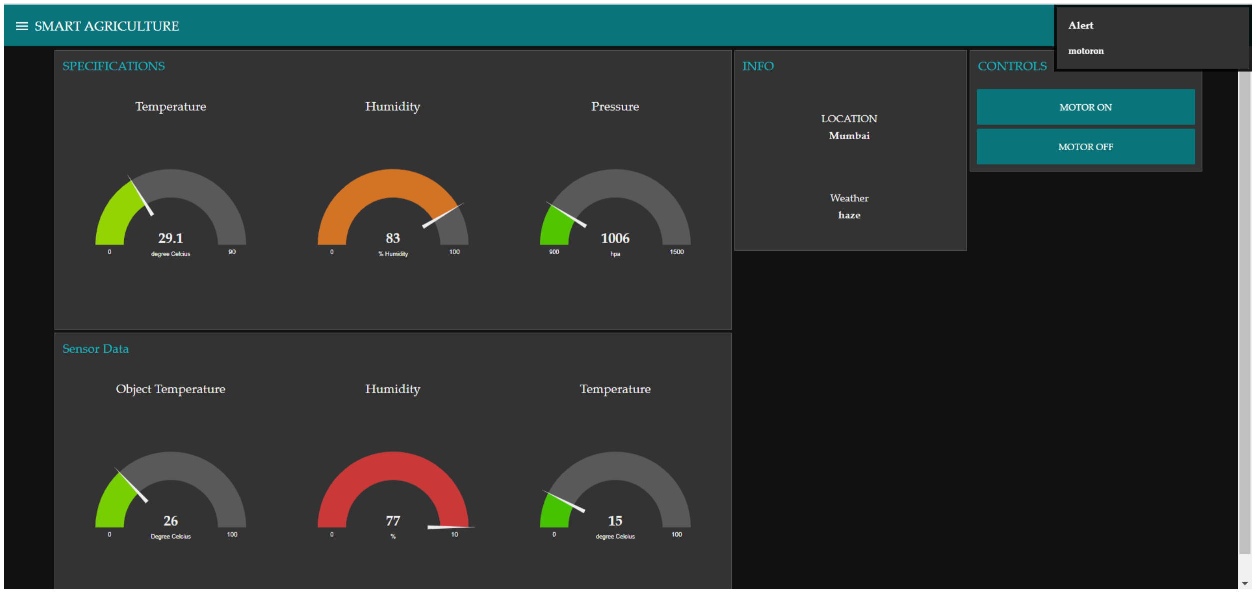
FLOWCHART

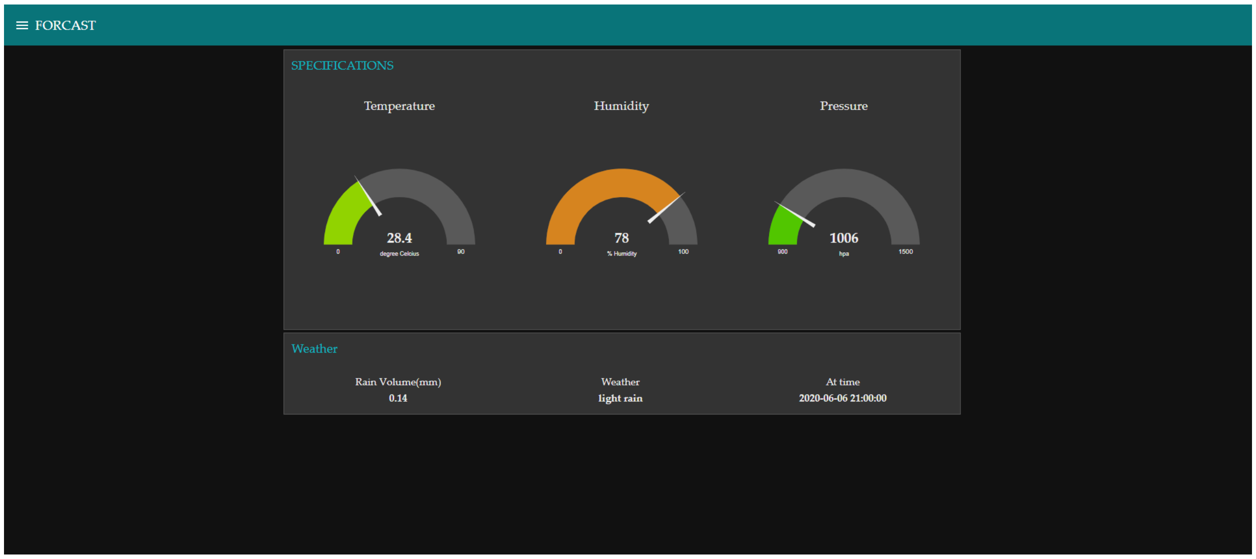


Create IBM device and Node Red flow
Receive data from IOT Simulator on device
Send http request to Open Weather API
Display Weather info on the Node Red UI
Send weather information to IBM IOT device by using IBM IOT IN nodes
Display received data on cards
Configure buttons to turn motor ON and OFF
Send command to python script on button click to be connected to IBM device.
Push command and data to IBM device through python script after button click.

RESULT

- The UI for the Smart Agriculture App is shown below.

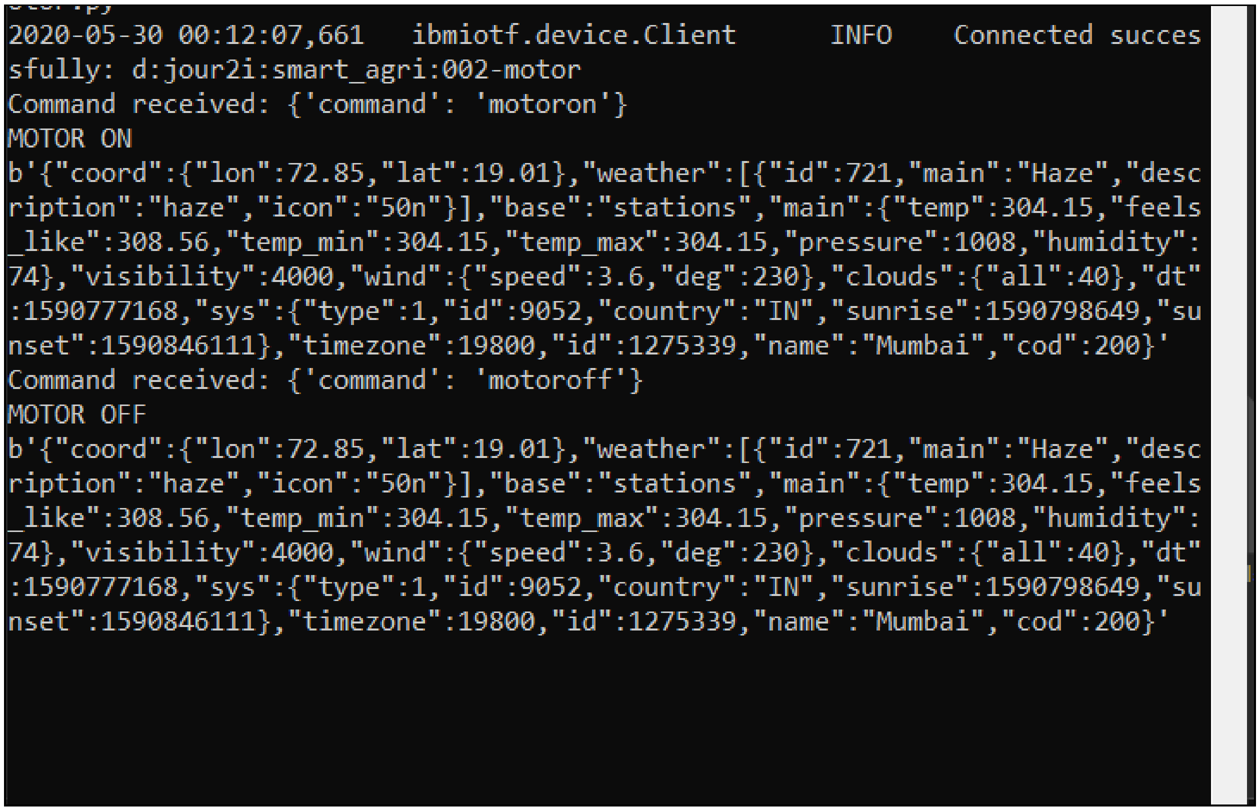




- The Temperature, humidity, pressure, location, general weather and motor ON/OFF controls are provided in the UI.
- An alert is flashed when the motor controls are triggered.
- On the 2nd tab, weather forcast for the next day is shown.

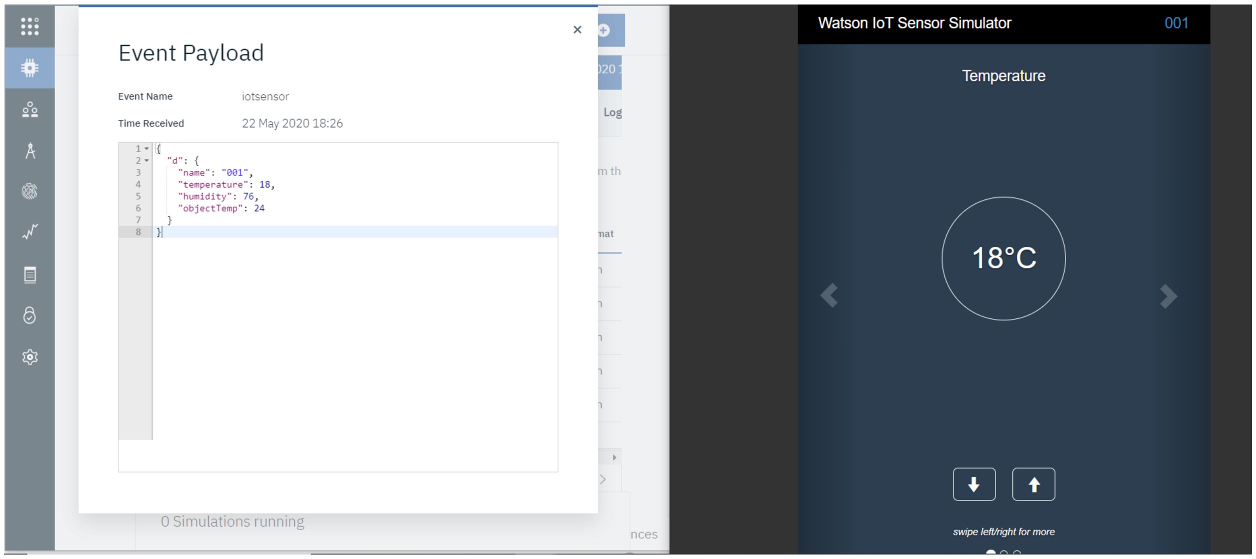
- Using the puthon script this motor control trigger is handled and suitable messages are displayed on the terminal as well as the commands are sent to the IBM IOT device.

- The command and weather status from OpenWeather API is displayed and sent to the device.



```
2020-05-30 00:12:07,661   ibmiotf.device.Client      INFO    Connected succes
sfully: d:jour2i:smart_agri:002-motor
Command received: {'command': 'motoron'}
MOTOR ON
b'{"coord":{"lon":72.85,"lat":19.01},"weather":[{"id":721,"main":"Haze","desc
ription":"haze","icon":"50n"}],"base":"stations","main":{"temp":304.15,"feels
_like":308.56,"temp_min":304.15,"temp_max":304.15,"pressure":1008,"humidity":
74},"visibility":4000,"wind":{"speed":3.6,"deg":230},"clouds":{"all":40},"dt"
:1590777168,"sys":{"type":1,"id":9052,"country":"IN","sunrise":1590798649,"su
nset":1590846111},"timezone":19800,"id":1275339,"name":"Mumbai","cod":200}'
Command received: {'command': 'motoroff'}
MOTOR OFF
b'{"coord":{"lon":72.85,"lat":19.01},"weather":[{"id":721,"main":"Haze","desc
ription":"haze","icon":"50n"}],"base":"stations","main":{"temp":304.15,"feels
_like":308.56,"temp_min":304.15,"temp_max":304.15,"pressure":1008,"humidity":
74},"visibility":4000,"wind":{"speed":3.6,"deg":230},"clouds":{"all":40},"dt"
:1590777168,"sys":{"type":1,"id":9052,"country":"IN","sunrise":1590798649,"su
nset":1590846111},"timezone":19800,"id":1275339,"name":"Mumbai","cod":200}'
```

- The received commands are displayed in the Recent Events section of the IBM device.
- Sensor data is also displayed on the device.



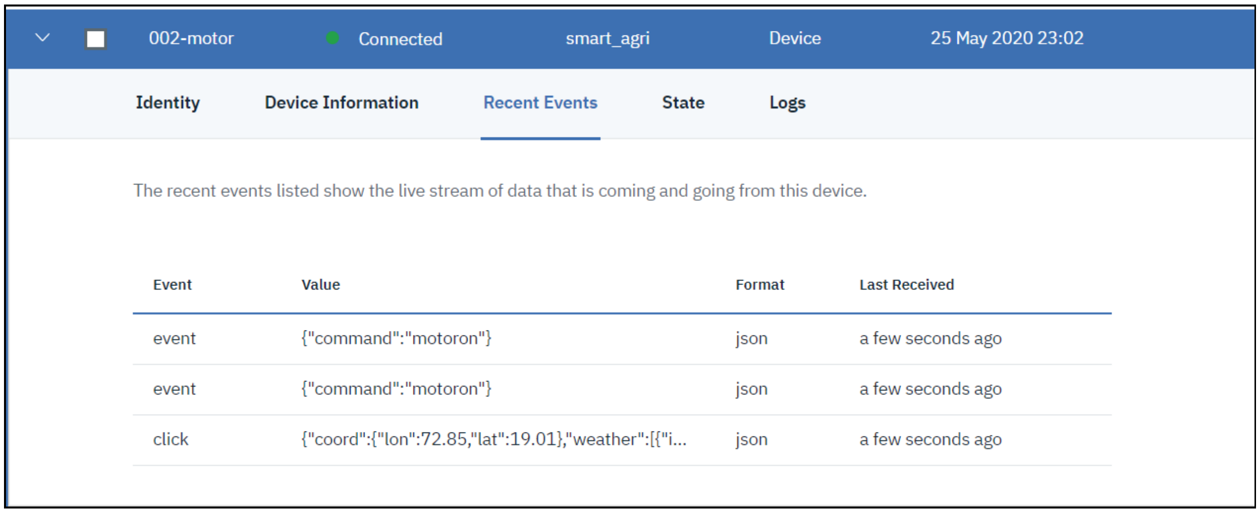
Event Payload

Event Name iotsensor

Time Received 22 May 2020 18:26

```
{
  "d": {
    "name": "001",
    "temperature": 18,
    "humidity": 76,
    "objectTemp": 24
  }
}
```

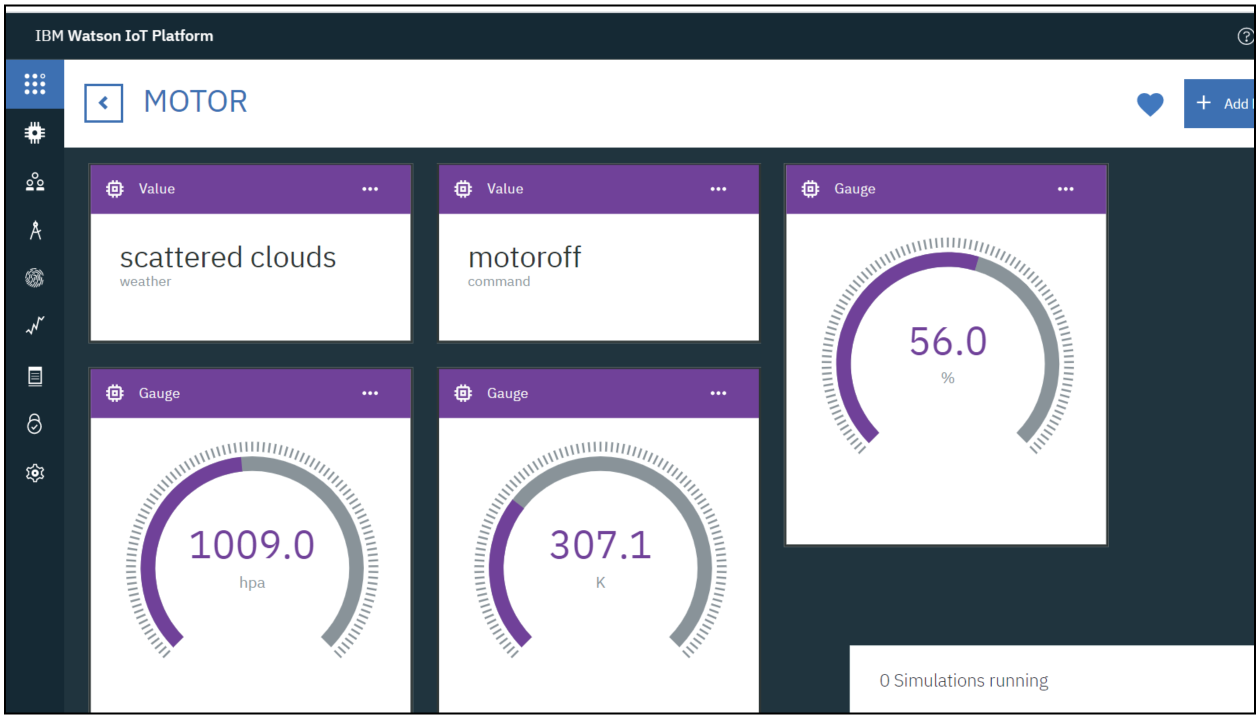
0 Simulations running

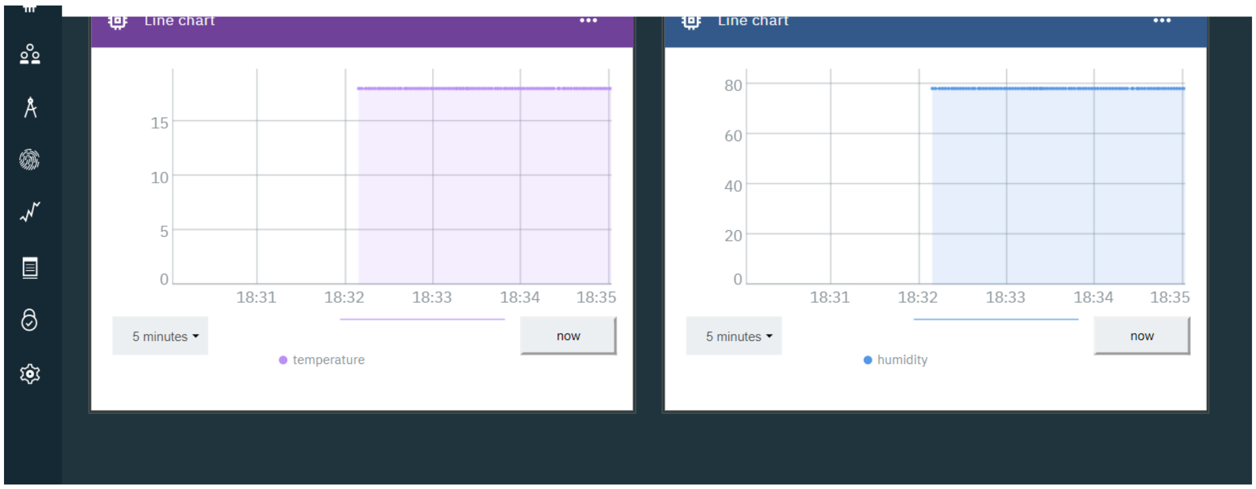


The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event	{"command":"motoron"}	json	a few seconds ago
event	{"command":"motoron"}	json	a few seconds ago
click	{"coord":{"lon":72.85,"lat":19.01},"weather":[{"i...	json	a few seconds ago

- Specific commands and data is extracted from the events received by the device and displayed in the form of cards as shown below.





- Displaying sensor data on IBM cards.

ADVANTAGES & DISADVANTAGES

Advantages:

- One time investment for farmer to install the sensors and use the app.
- Reliable real time data and controls available at the fingertips.
- Easy controls of devices that are connected.
- Savings and more profit as the farmer need not be physically present at the farm.

Disadvantages:

- Initial cost of installation of sensors might be high.
- An internet connection is needed for the app to work.
- Intelligence and judgement of humans can't be replicated accurately by an app.

APPLICATIONS

- By simply knowing the location of the farm, real time data and control can be achieved with this app.
- Since mobile phones are used by almost everyone today, this technology can be easily implemented.
- Farms of any size and at any location can be covered with the help of IOT.

CONCLUSION

- Using node red and IBM Cloud platform makes designing and deployment of IOT based applications easy.
- Required data can be requested from APIs and used in the applications.
- Smart Agriculture can be applied to all farms to automize processes and provide information and control to the farmers.

FUTURE SCOPE

- Other devices like pesticide sprays, water sprinklers and cameras can also be configured to IOT and connected to the app.
- Cameras can be used to monitor the field for intrusion or other activities, supported by security alarms if required.
- The aim will be to automize farming as much as possible and make the technology available to all farmers.

BIBILOGRAPHY

- IBM cloud Platform: https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform
- Node Red: https://nodered.org/
- IBM Watson IOT Platform: https://www.iotone.com/software/ibm-watson-iot-platform/s62
- Using IBM Watsion: https://smartinternz.com/assets/docs/Smart%20Home%20Automation%20using%20IBM%20cloud%20Services%20(1).pdf

APPENDIX

A. Source code

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

```
#Provide your IBM Watson Device Credentials
organization = "abcdef" # repalce it with organization ID
deviceType = "smart_agri" #replace it with device type
deviceId = "002" #repalce with device id
authMethod = "token"
authToken = "abcd1234"#repalce with token
str="cmd"

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='motoron':
                print
                print("MOTOR ON")
        elif cmd.data['command'] == 'motoroff':
            print("MOTOR OFF")
        str=cmd.data['command']
        data={'command':str}
        success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")



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:
        #Send weather conditions from API to IBM Watson
response=requests.get("http://api.openweathermap.org/data/2.5/weather?q=Mumbai,
IN&appid=80a19d6af38283c126f2021f4145bfea")

        #print data
        def myOnPublishCallback():
            print(response.content)

        time.sleep(1)
        deviceCli.commandCallback = myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```
