SMART AGRICULTURE SYSTEM BASED ON IOT

PROJECT REPORT

1. INTRODUCTION

1.1 OVERVIEW:

I have been assigned with the project "Smart Agriculture System Based On IOT" which as the name of the project signifies is built upon the IOT platform. An exception to the normal IOT projects(which involve hardware devices like arduino, rasberrypi etc.) ,this project has been solely developed on the IBM cloud platform in collaboartion with the smartinternz team for the RSIP 2020 event. This project dervies an inspiration from the agricultural practices that are widely practised throughout the country.

India is a vast country where the atmospheric conditions keep on varying as we move along it's longitude or latitude & with these varying conditions the agricultural practices also change. Some areas require more water & water distribution cycles as compared to others depending upon the soil moisture as well as it's retention capacity. So what we propose to do through this project is to provide the user with a real time monitoring service measuing the current weather, humidity & soil moisture which ultimately helps the farmer to be fully aware of the conditions in his field & handling these can help him boost agriculture yields.

1.2 PURPOSE

The above mentioned conditons affect the quality as well as the quantity of agricultural produce. Keeping this in mind, with this project I have in collaboration with the smartinternz platform & the IBM cloud proposed an alternative and a remote way of watering the fields once the crops have been sown.The goal of this project is to deliver a working end user product (an app) that integrates IOT & cloud to help user/farmer monitor the atmospheric conditions remotely as well as providing control for managing fresh water motors to irrigate the fields based on the data provided by the application.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

1. The problem that was identified with the current irrigation system or the agricultural system (on a large scale) was that the farmers did their agricultural activities based on approximations (whether about the weather, the humdity or the soil temp.) & without any means for finding out the accurate data for these quantities. And these are the quantities that ultimately play the most important role in determing the quality as well as the quantity of the yield , the farmers were underutilising there field's full potential

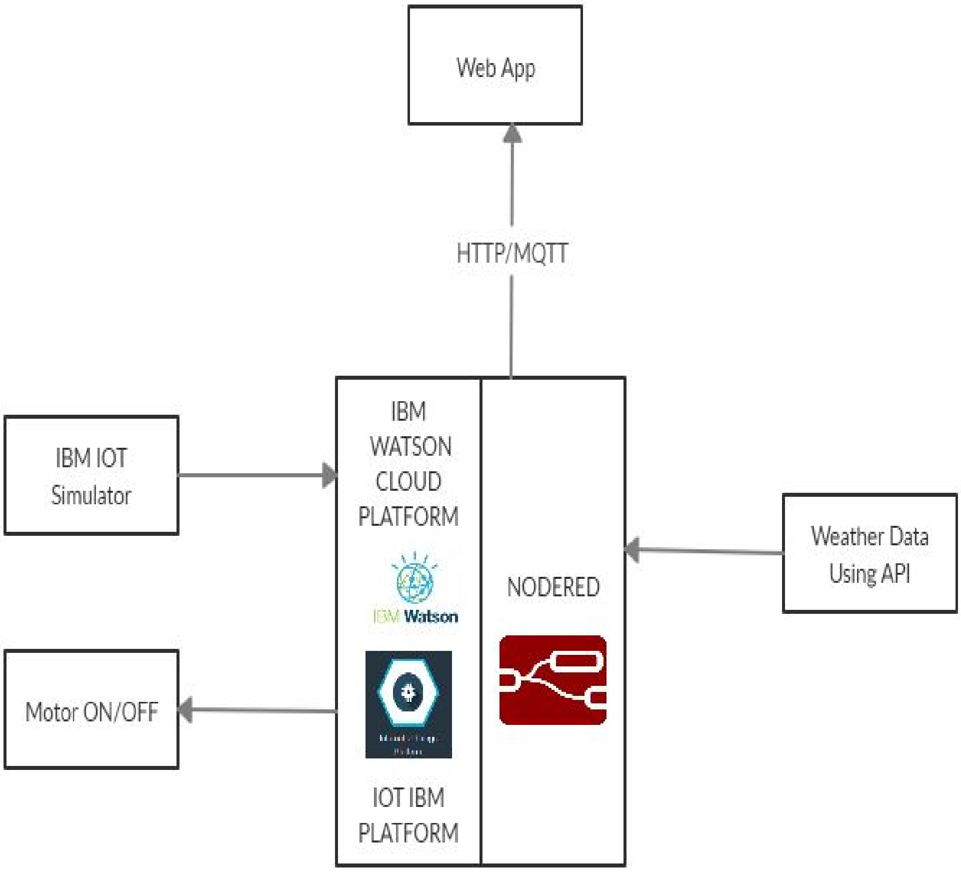
2. Another problem that was identified is that for the farmers whose houses are farther from the fields find trouble in regularly getting upto the fields only to find out that conditions on that day are not particularly suitable & they are not equipped with any system to remotely monitor the weather conditions as well as no remote control for controlling the irrigation motors remotely.

2.2 PROPOSED SOLUTION

Above some of many issues (that are prevalent in the agricultural practices of our country) were identified & used as working motive for this project. Through this project we aim to eliminate these issues by providing a mobile solution i.e an application to the farmer so that he can easily monitor the agricultural conditions from anywhere & then also provide him the control of motors remotely. This app will developed upon propreitary code(means that the app coding will not be available to general public for review).The app will be developed for all major mobile platforms(android/ios) as well as major desktop platforms like (windows ,macos, etc.). A fixed timeline would drawn out beforehand for all the beta as well stable releases.

3. THEORETICAL DESCRIPTION

3.1 BLOCK DIAGRAM

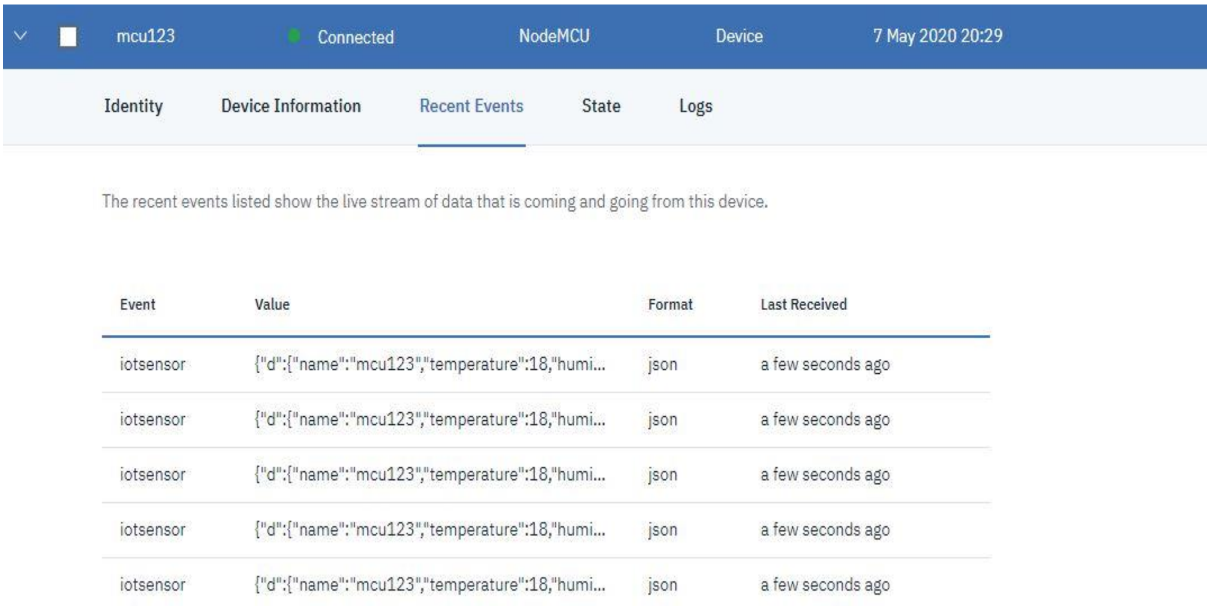


3.2 HARDWARE/SOFTWARE DESIGNING

This project does not involve any kind of hardware & is solely based upon software designing. The software for this project is designed on the IBM cloud service namely the IBM IOT Watson platform apart from some other helper services from the IBM cloud itself. The project involves creating two virtual devices one on the user's end other on the receiver end.The user end device is then connected to the IOT simulator which sends the weather condtions info. back to the device.It is further connected through the openweather api to connect the city concerned with the user.This setup ensures regular transfer of weather data at regular intervals to the device.After this node red service from IBM cloud came into use & proper connections were layed out for building as well as debugging the web app.After this the second device is configured to work with the python code for controlling the motors & the nodered app as a whole is configured to work in one single flow & respond to motor on & motor off controls.

4. EXPERIMENTAL INVESTIGATION:

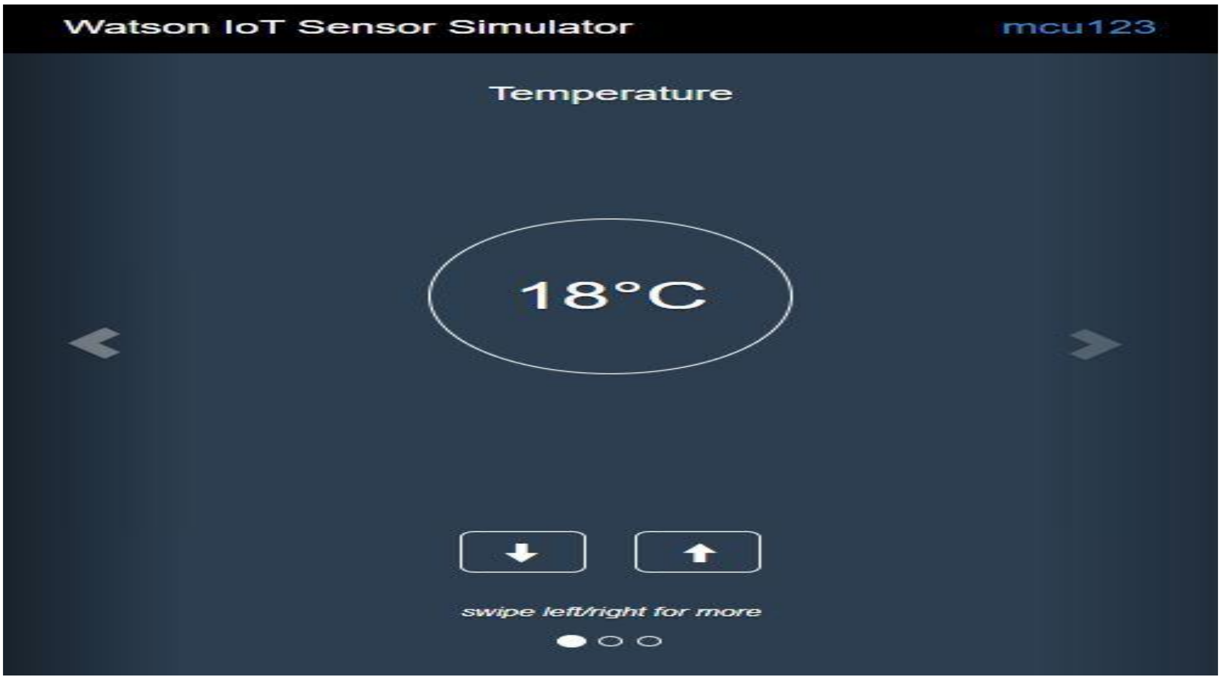
This project was developed on the IBM IOT platform that is a service provided by the IBM Cloud. Then the first device was created (which is the one going to be used by the farmer for sending commands).

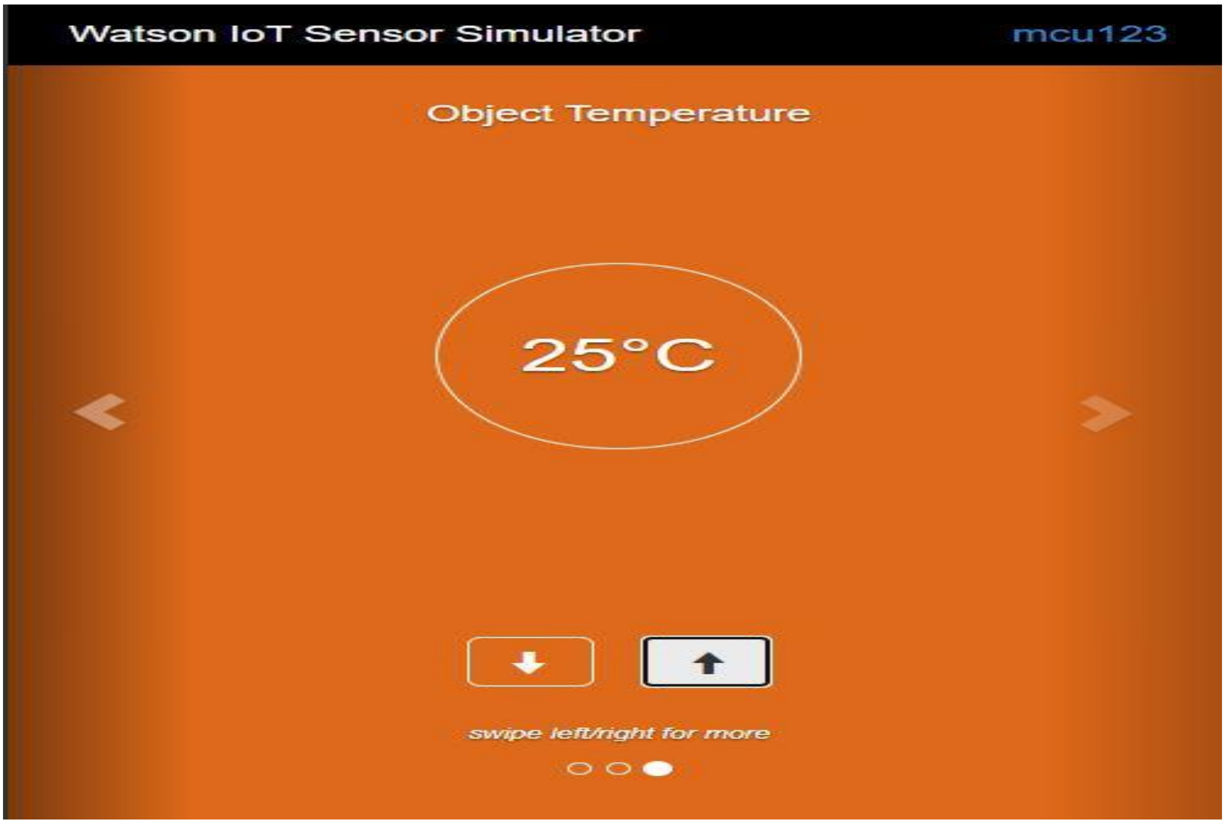


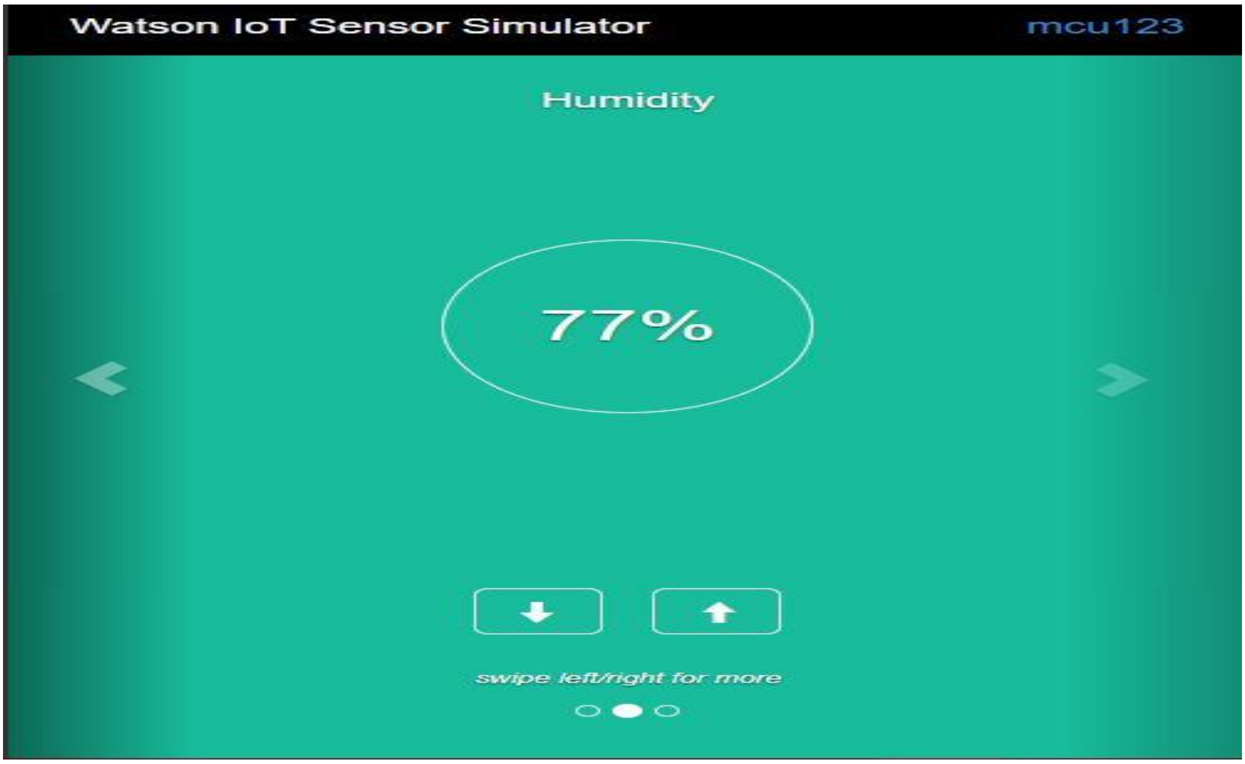
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
iotsensor	{"d":{"name":"mcu123","temperature":18,"humi...	json	a few seconds ago
iotsensor	{"d":{"name":"mcu123","temperature":18,"humi...	json	a few seconds ago
iotsensor	{"d":{"name":"mcu123","temperature":18,"humi...	json	a few seconds ago
iotsensor	{"d":{"name":"mcu123","temperature":18,"humi...	json	a few seconds ago
iotsensor	{"d":{"name":"mcu123","temperature":18,"humi...	json	a few seconds ago

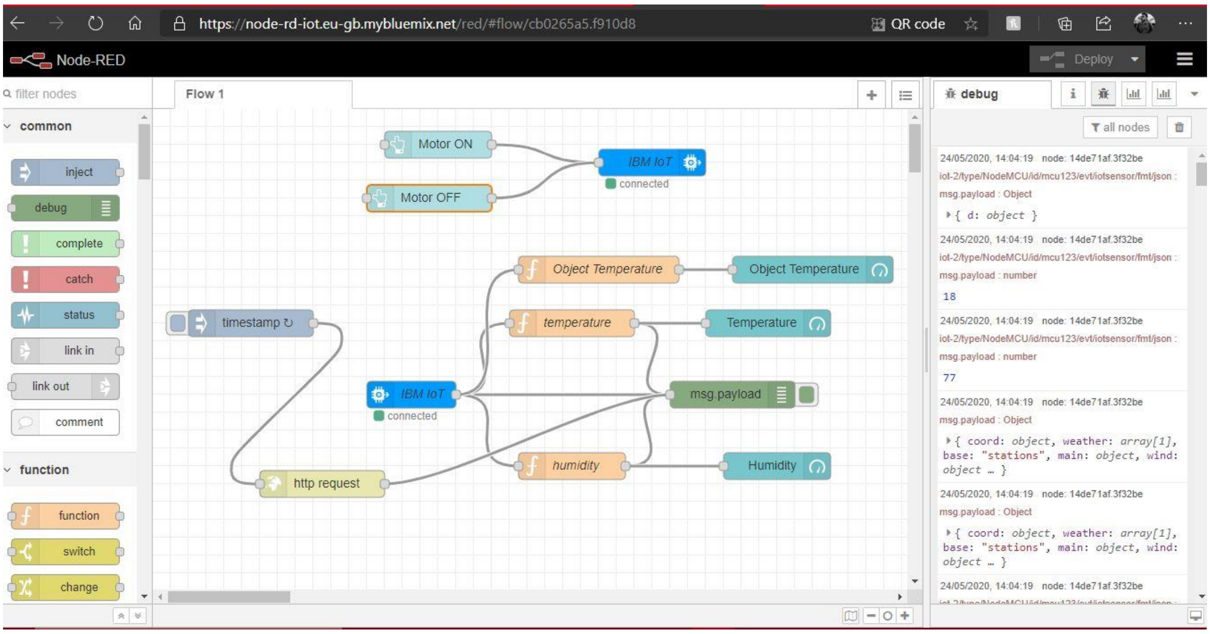
This device was then connected to the watson IOT simulator by linking the device credentials .Through this the device was connected to the cloud receiving data remotely





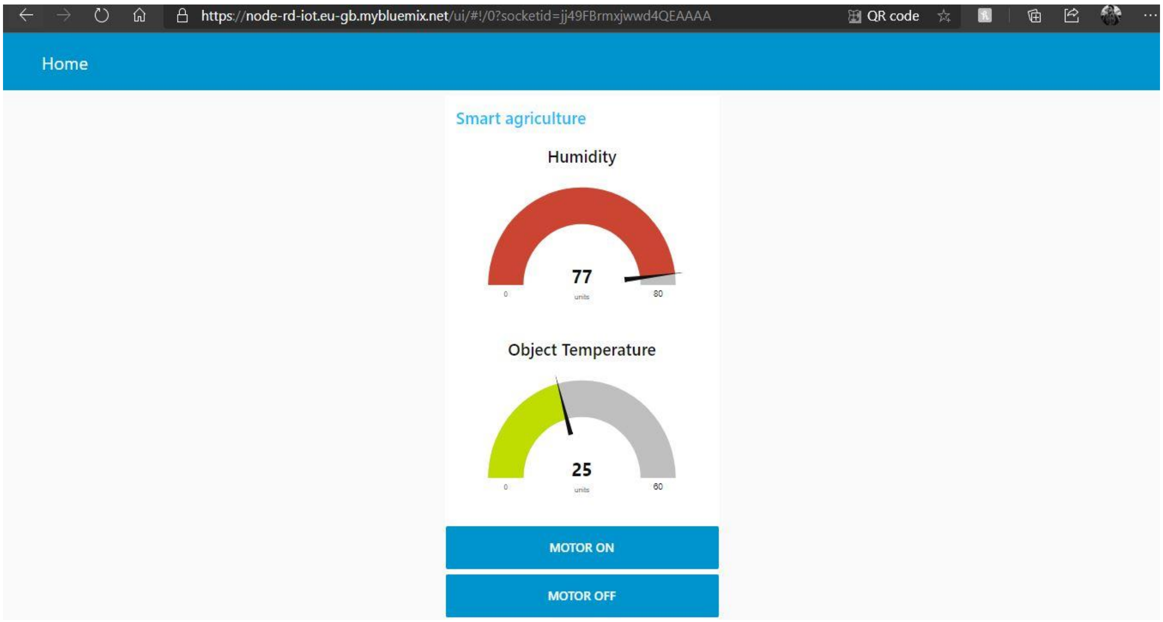


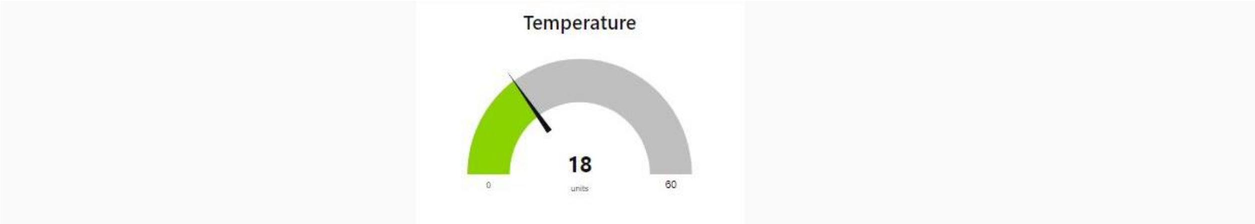
I then configured the the node red app enabling to receive data from the iot platform. It was linked with the open weather api key to get live weather details for the desire. with this the app got its first functionality of monitoring ready & working



```
24/05/2020, 14:04:19   node: 14de71af.3f32be
iot-2/type/NodeMCU/id/mcu123/evt/iotsensor/fmt/json :
msg.payload : Object
▸{ d: object }
24/05/2020, 14:04:19   node: 14de71af.3f32be
iot-2/type/NodeMCU/id/mcu123/evt/iotsensor/fmt/json :
msg.payload : number
18
24/05/2020, 14:04:19   node: 14de71af.3f32be
iot-2/type/NodeMCU/id/mcu123/evt/iotsensor/fmt/json :
msg.payload : number
77
24/05/2020, 14:04:19   node: 14de71af.3f32be
msg.payload : Object
▸{ coord: object, weather: array[1],
base: "stations", main: object, wind:
object … }
24/05/2020, 14:04:19   node: 14de71af.3f32be
msg.payload : Object
▸{ coord: object, weather: array[1],
base: "stations", main: object, wind:
object … }
24/05/2020, 14:04:19   node: 14de71af.3f32be
```

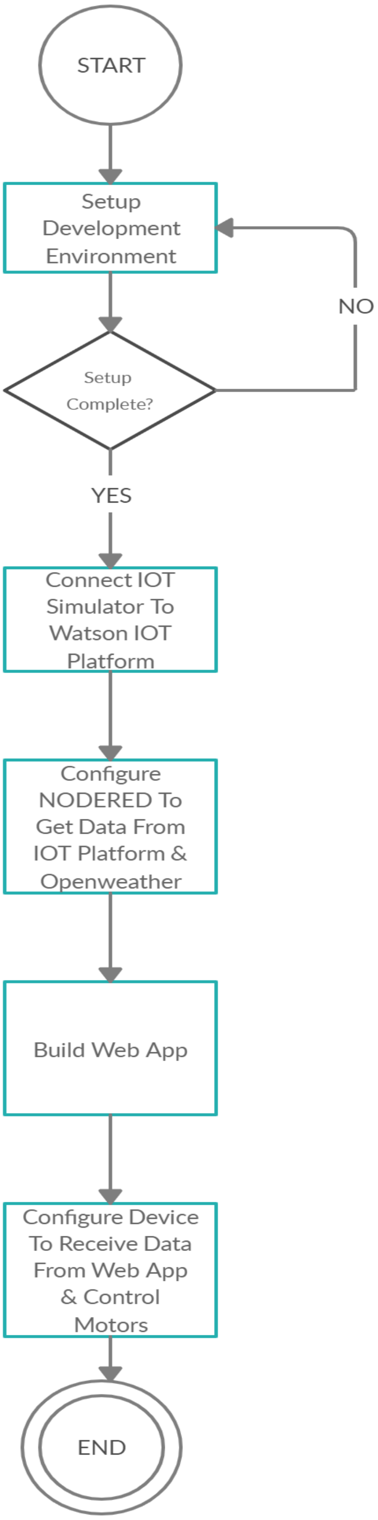
After this to add the second functionaly the nodes were configured as well as button added to control the motors remotely.The dashboard functionality was added finally finishing the app.





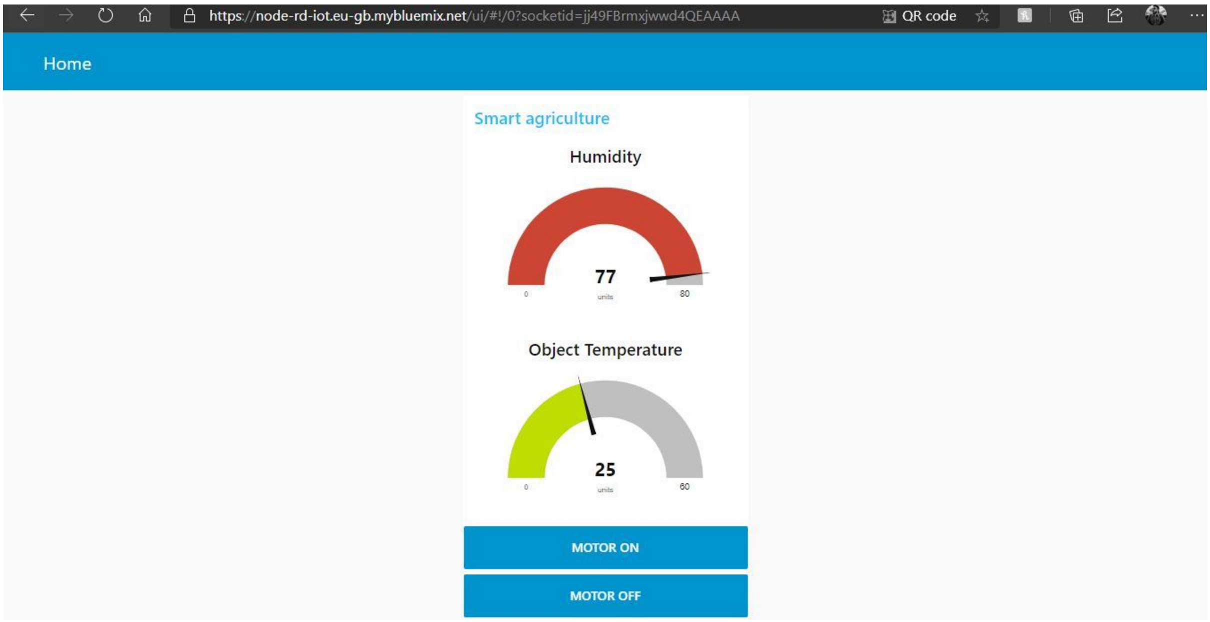
The final step was to write the python code to integrate all the mid products into single streamlined final working product.

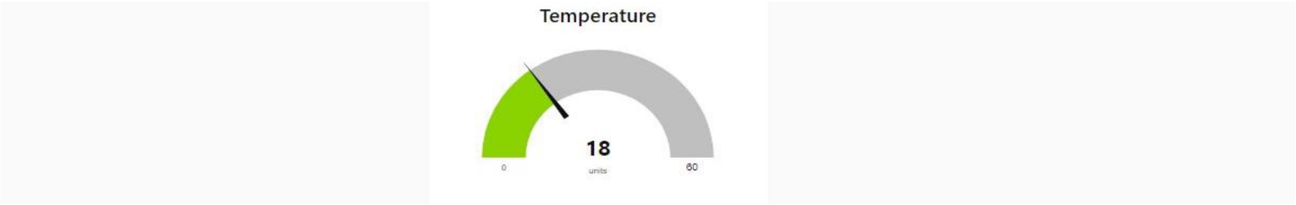
5.FLOWCHART:

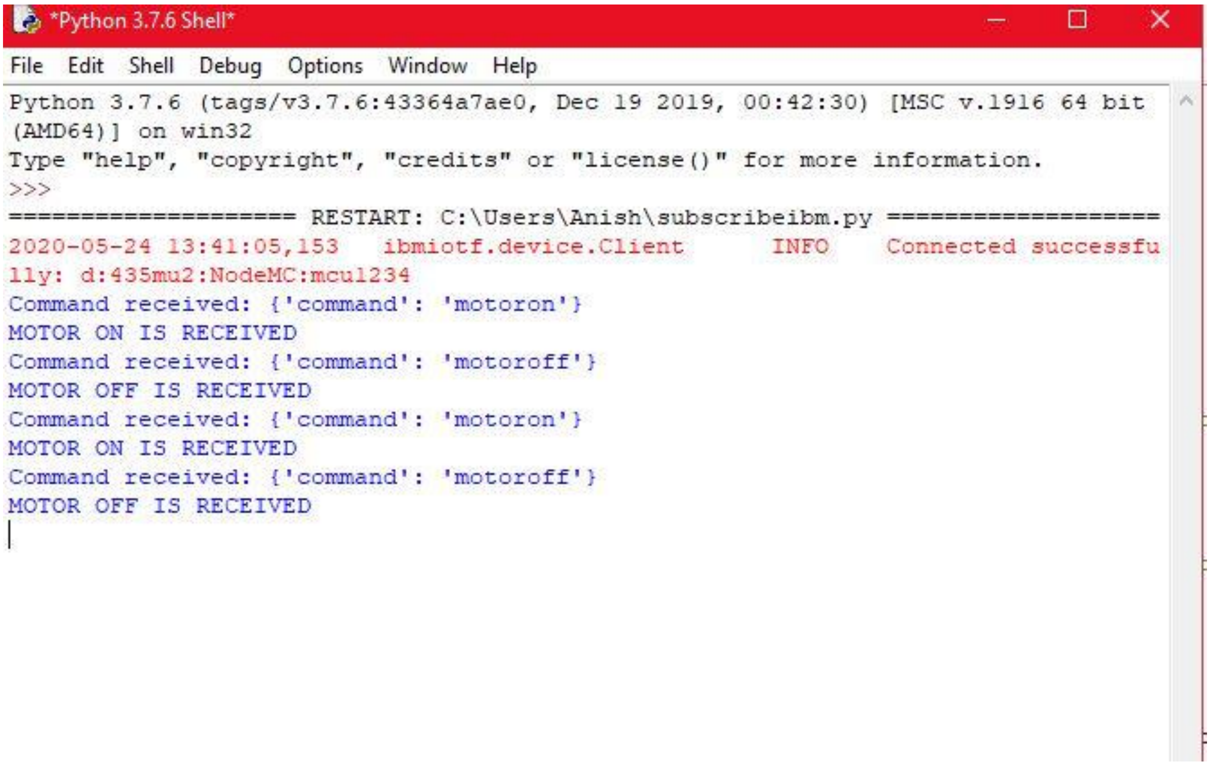

START
Setup Development Environment
Setup Complete?
NO
YES
Connect IOT Simulator To Watson IOT Platform
Configure NODERED To Get Data From IOT Platform & Openweather
Build Web App
Configure Device To Receive Data From Web App & Control Motors
END

6.RESULT:

Result of pursuing this project was to obtain a fully functiona & working end user prototype







```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: C:\Users\Anish\subscribeibm.py ===================
2020-05-24 13:41:05,153   ibmiotf.device.Client      INFO    Connected successfu
lly: d:435mu2:NodeMC:mcu1234
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
Command received: {'command': 'motoroff'}
MOTOR OFF IS RECEIVED
Command received: {'command': 'motoron'}
MOTOR ON IS RECEIVED
Command received: {'command': 'motoroff'}
MOTOR OFF IS RECEIVED
```

7.ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

1. User can monitor the weather conditions remotely as well.
2. User can easily irrigate his fields with a single tap.

DISADVANTAGES:

1. The app is not a standalone product and reuqires an external physical smart device to run it as well as various sensors as well.

2. Apart from it's own cost the app may incur other charges to the farmer which may not be affordable to him(data charges, device charges)

3. The app may not show accurate data when the atmospheric conditions keep changing rapidly (like in the monsoons)

8. Applications:

1. Smart Agriculture System based on IOT app can monitor soil moisture & climatic conditions to grow & yield a good crop

2. Farmer can get real time weather forecasting data using this app

3. Based on all the parameters he can water his crop by controlling the motors using the app

4. He can also water the fields remotely using this app

9. CONCLUSION:

The project was concluded by collecting & combining all the data & small parts of the project into the final fully working remote app for displaying the appropriate data to the user. This app is intended to be supported upon multiple platforms. Through this app the user can stay alerted of the current conditions of his fields like moisture(which was designed in the IOT platform and connected with the cloud IOT simulator which simulates & sends the live data of the crops) as well as the current climatic situation in the area(which was ensured through the integration of openweather api for the specific area that the user intends).According to the conditions identified by the user he can control the water motors &

water the fields as per his convenience. He can do this while sitting a considerable distance apart from his fields.

10. FUTURE SCOPE:

1. The app developed has a vast potential in a lot of areas relating to agriculture activities & more and more functionality can be added with subsequent versions of the app

2. Currently the app only supports monitoring conditions as well as controlling water motors but with subsequent releases the app can e turned into an allrounder managing all the possible major aspects of the farming (like automatic seeding, crop diseases detection etc.)

3. Alpha & beta versions can be developed beforehand & can be distributed to a small group of testers before releasing the stable version so as to determine the redundant functionalities present as well as the missing essential functional requirements that the users expect.

4. Apart from agriculture the app can also be employed to vast number of other industries like military & for educational purposes as well

11. BIBLIOGRAPHY

https://smartinternz,com

https://nodered.org/

https://cloud.ibm.com/

https://aaaaa.internetofthings.ibmcloud.com/dashboard/

APPENDIX

Source code:

```
import time
import sys
import ibmiotf.application  # to install pip install ibmiotf
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "435mu2" #replace the ORG ID
deviceType = "NodeMC"#replace the Device type wi
deviceId = "mcu1234"#replace Device ID
authMethod = "token"
authToken = "smartaggri123" #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()
```