1)INTRODUCTION:-

1.1) OVERVIEW:-

- In this "SMART AGRICULTURE" project, we have created a UI based on iot using nodered, IBM cloud services and ibm iot sensor.we have connected the iot sensor virtually to IBM cloud and UI.

1.2) Purpose:-

-This project focuses on agriculture, and this enables the farmers to monitor the field parameters and control the motors for irrigation from anywhere.

- This technology would ultimately improve the yield and limit the water usage to a greater exrtend, as farmers can consider the field paramters such as moisture, temp and humidty and climate parameters from open weather api before irrigation.

2.LITERATURE SURVEY:-

2.1)Existing problem:

- Farmers of today are not aware about changing climatic conditons due to global warming. This makes them irrigate crops without considering the weather forecast. They even doesnt consider their own field parameters before irrigation. This causes crop failure nd water wastage.

- This issue causes both water waste and crop failure to farmers and pushes them in high interest loans.

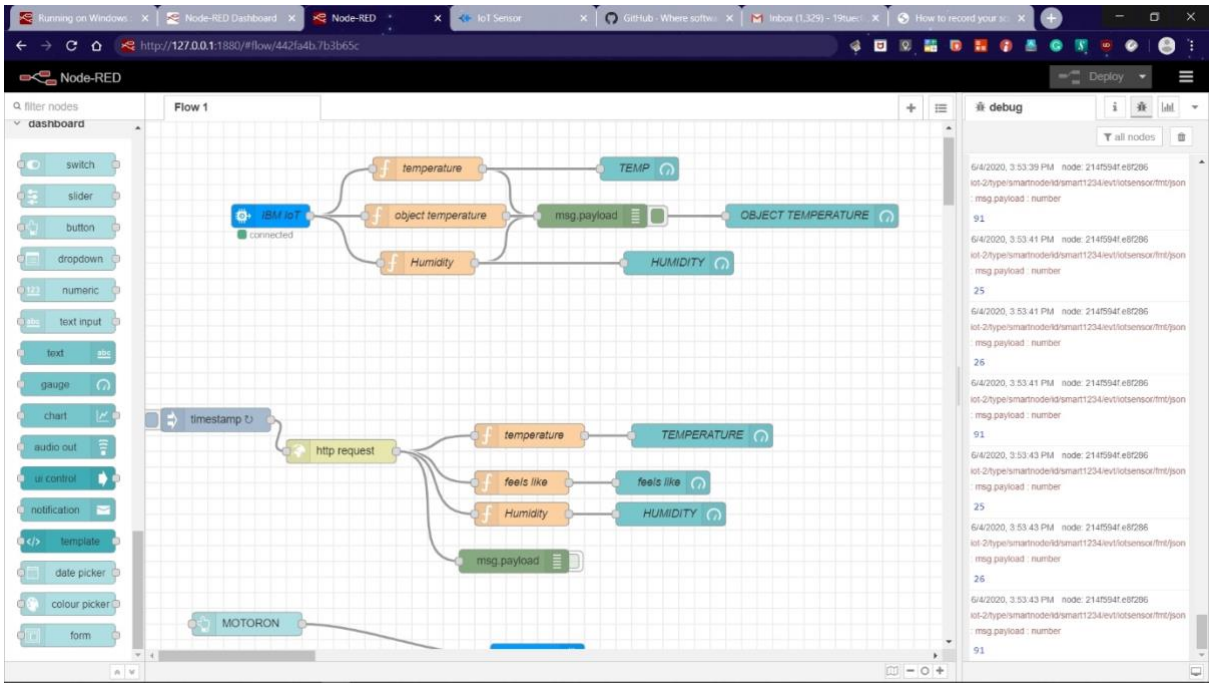
2.2)Proposed solution.

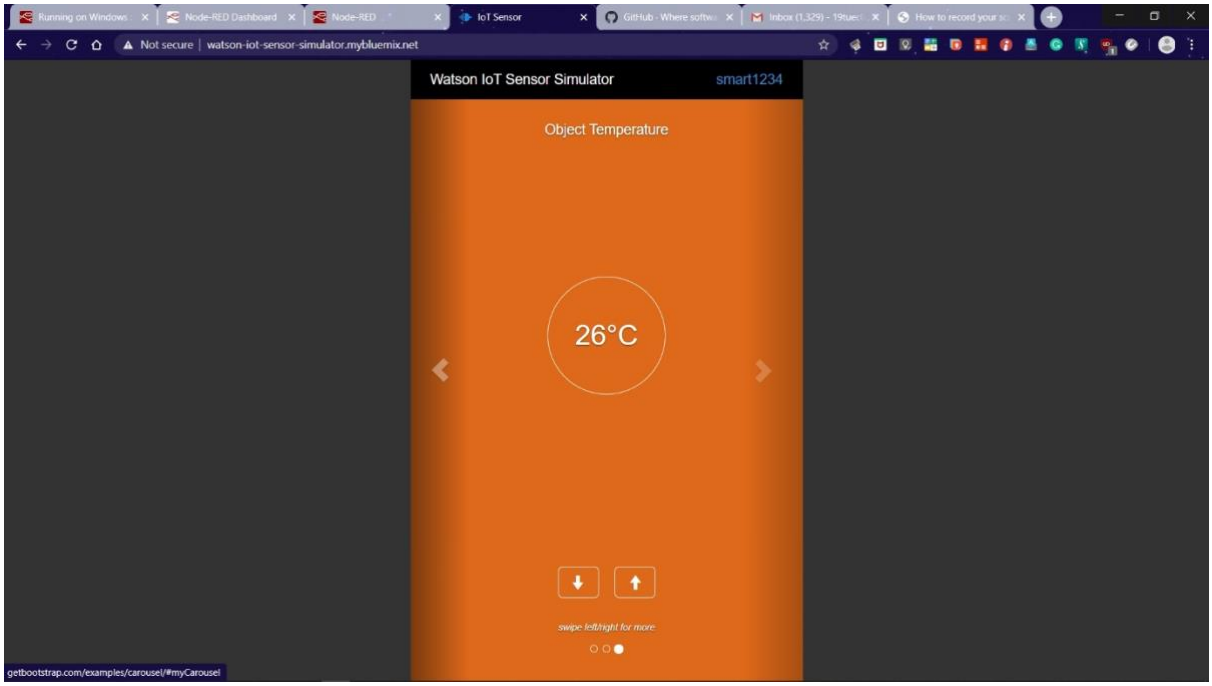
:

-Our solution is to provide a UI to farmers, through which they can monitor the field paramaters|(humidity, temperature,moisture), weather data from open weather api. This UI also provide two controll buttons, using which they can control motors for irrigation.

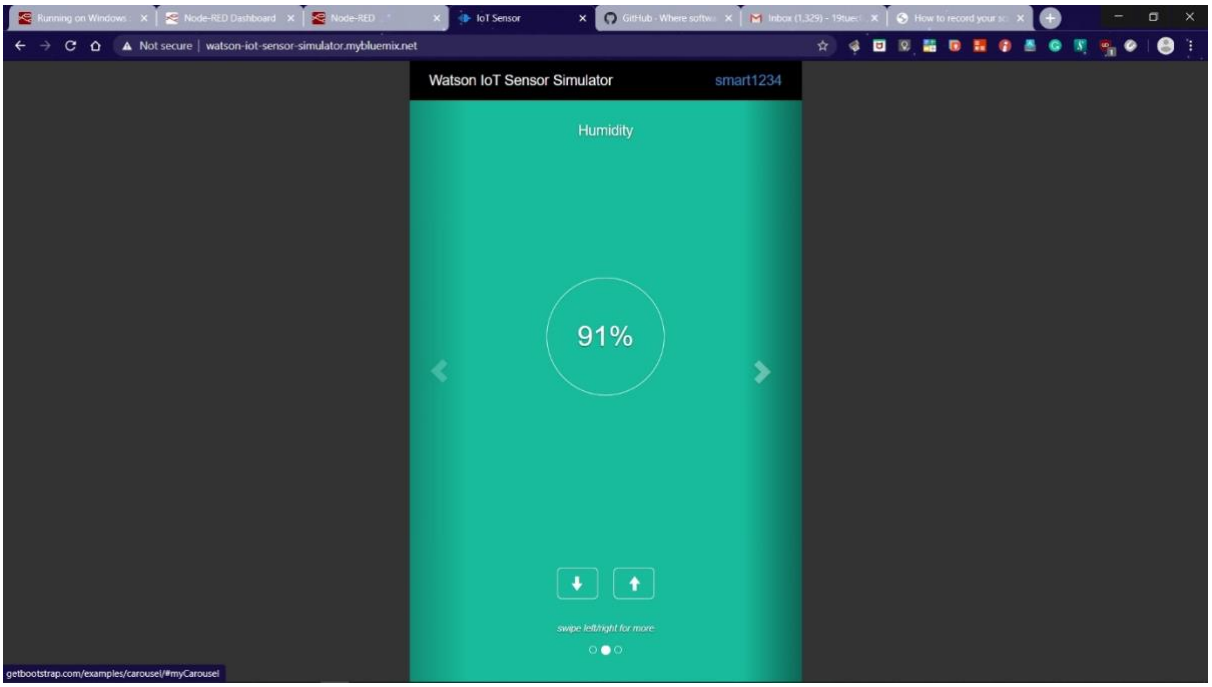
- This would reduce water wastage and increase productivity.

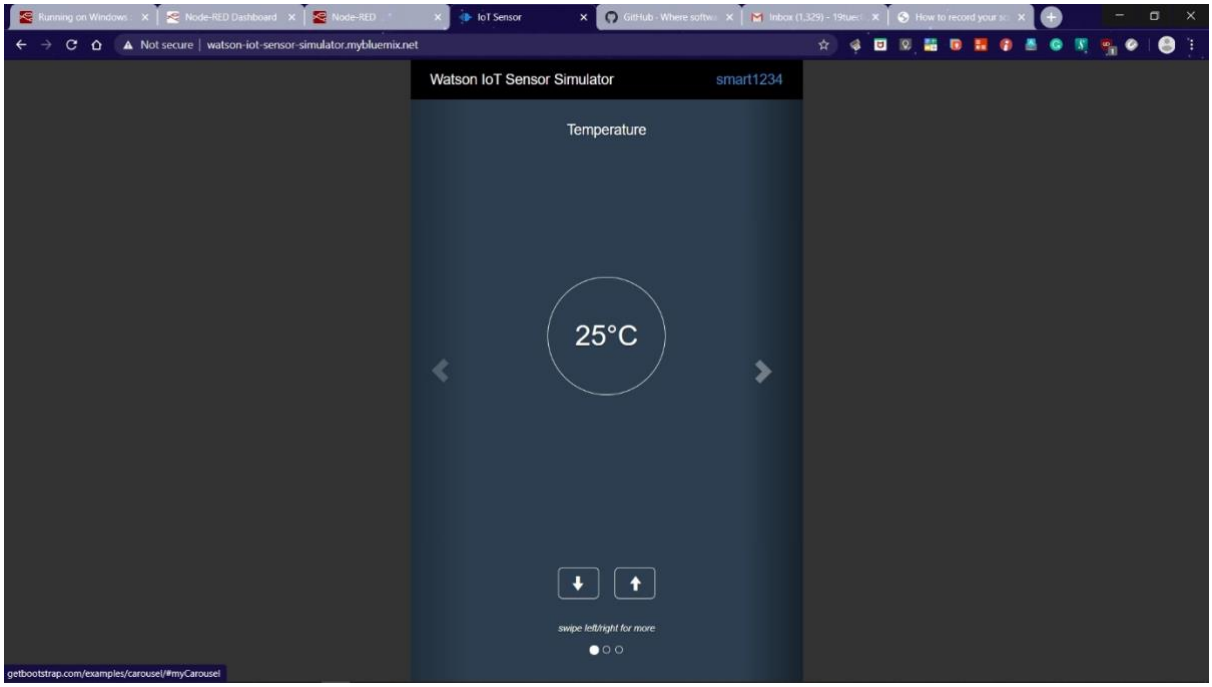
3)HARDWARE AND SOFTWARE DESIGNING:





Jj





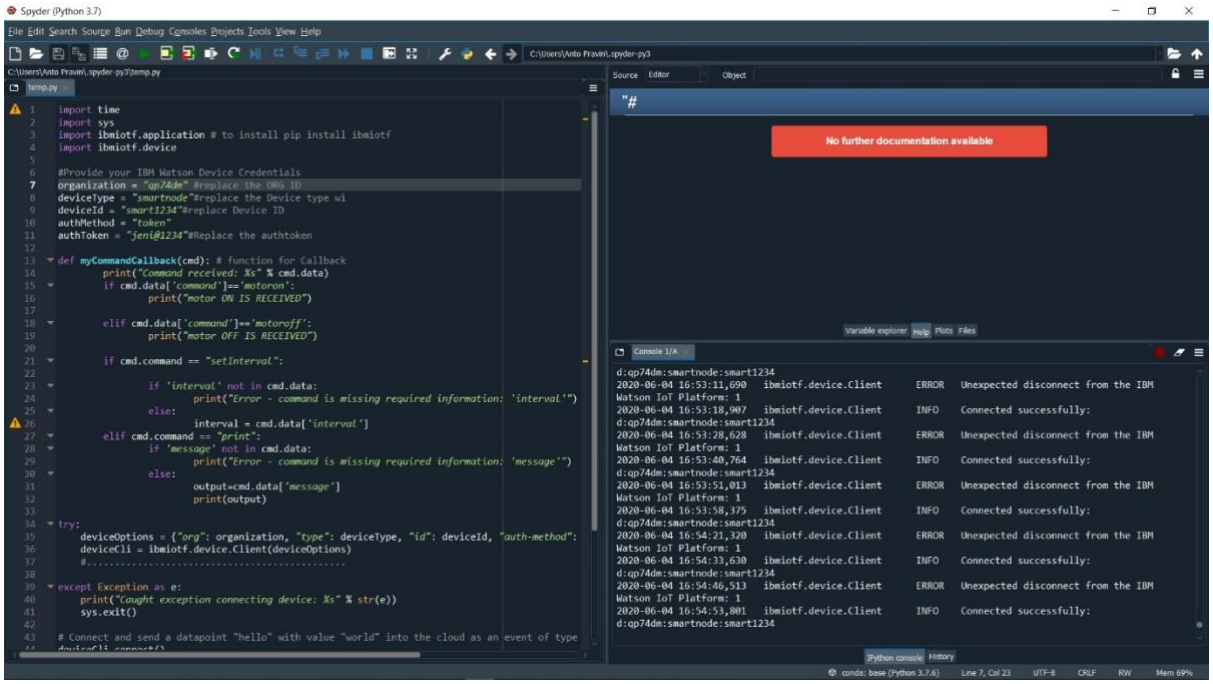
EXPERIMENTAL INVESTIGATION:

-The UI was created using NODERED, all the nodes where configured and the UI was run.

- This UI was abled to display the different gauges to show the fieldparamters.

-This UI was also abled to display the data from open weather api by gauges.

-This UI provided two control buttons fro controlling the motor,and when one of any these two buttons are clicked, this returned "motor-on command received" in the idle where the python code was ran. This act show that the UI and the code are connected.



5)RESULT:

-Thus the required UI with complete required specifications in created and Tested

6)disadvantage

This UI can't make its own wiser decision, which would be very useful for illiterate farmers.

ADVANTAGES:

-This UI saves time for farmers as they controll and monito their farm without visiting the farm .this makes them concentrate in other activities to gain money.

7)APPLICATIONS:

-In Agricultural and Horticultural fields.

8)CONCLUSION:

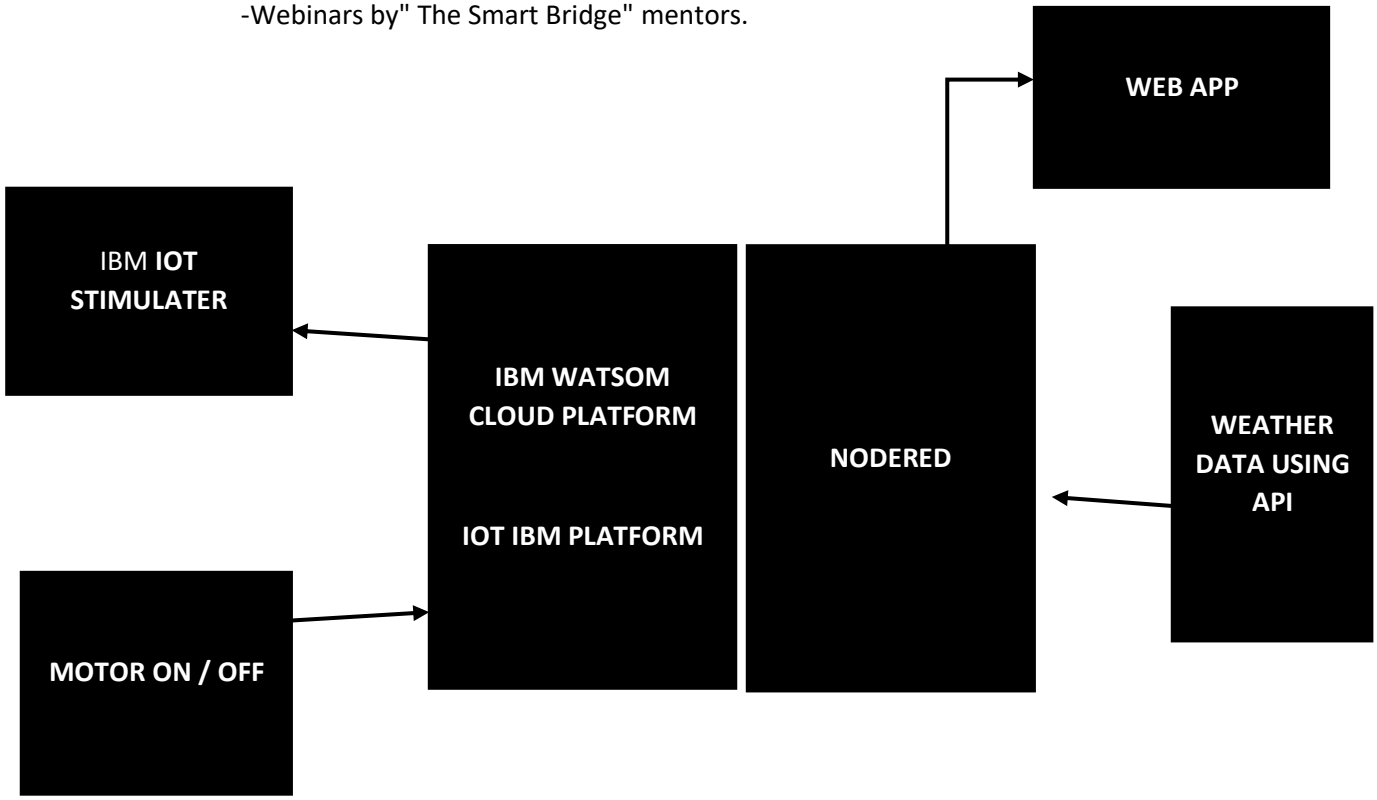
I strongly beleive that, this project would boost up the yield to feed growing population, and save water.

9)FUTURE SCOPE:

-This UI would further devoloped to provide wise decision for illitrate farmers.

10)BIBILOGRAPHY:

-Webinars by" The Smart Bridge" mentors.




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

```
#Provide your IBM Watson Device Credentials
organization = "qp74dm" #replace the ORG ID
deviceType = "smartnode"#replace the Device type wi
deviceId = "smart1234"#replace Device ID
authMethod = "token"
authToken = "jeni@1234"#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()
```