Project Report

SMARTFARMER - IOT ENABLED SMART FARMING APPLICATION

Team ID: PNT2022TMID54001

Table of Contents

1) Introduction:

1.1 Overview

This is a Smart Agriculture System project based on Internet Of Things (IoT), that can measure soil moisture, Humidity and temperature conditions for agriculture using Watson IoT services. IoT is network that connects physical objects or things embedded with electronics, software and sensors through network connectivity that collects and transfers data using cloud for communication. Data is transferred through internet without human to human or human to computer interaction.

In this project we have not used any hardware. Instead of real soil moisture, Humidity and Temperature data obtained from sensors we make use of IBM IoT Simulator which can transmit these parameters as required.

- ○ **Project Requirements** : Node-RED, IBM Cloud, IBM Watson IoT, Node.js, IBM Device, IBM IoT Simulator, Python 3.8, Open Weather API platform.
- ○ **Project Deliverables** : Application for IoT based Smart Agriculture System

1.2 Purpose

IoT based farming is grooming nowadays because it improves the entire agriculture system by monitoring the field in real-time. With the help of IoT in agriculture not only saves the time but also reduces the extravagant use of resources such as water and electricity.

Sometimes due to over or less supply of water in the agricultural field crops may not grow proper. Using IoT supply of water and growth of plants can be satisfied to a greater extent. The flow of water can be controlled from the application. Thus this approach towards Agriculture will help the farmers to get better yield at low cost and without much usage of Resources .

1.2.1 Scope of Work

- Create an IBM account.
- Create a device in IBM Cloud Account.
- Install Node-RED and download all the required nodes and configure the nodes.
- Create the open weather map account and get the api key and the weather conditions using api key in the Node-RED.
- Create a web application for user interaction for observation and control actions.

2) Literature Survey:

2.1 Existing Problem

- Agriculture is a field which forms the basis of our economy. Yet it faces a lot of problems in terms of availability of resources, Irrigation, increasing rate of Pesticides, Climatic disasters, Insects which ruin the crops and makes a huge loss this sector.
- In agriculture water is needed for the crops for their growth. If the Soil gets dry it is necessary to supply water. But sometime if the farmer doesn't visit the field it is not possible to know the condition of soil.
- Sometimes over supply of water or less supply of water affects the growth of crops.
- Sometimes if the weather/temperature changes suddenly it is necessary to take certain actions.
- Specific crops grow better in specific conditions, they may get damaged due to bad weather.

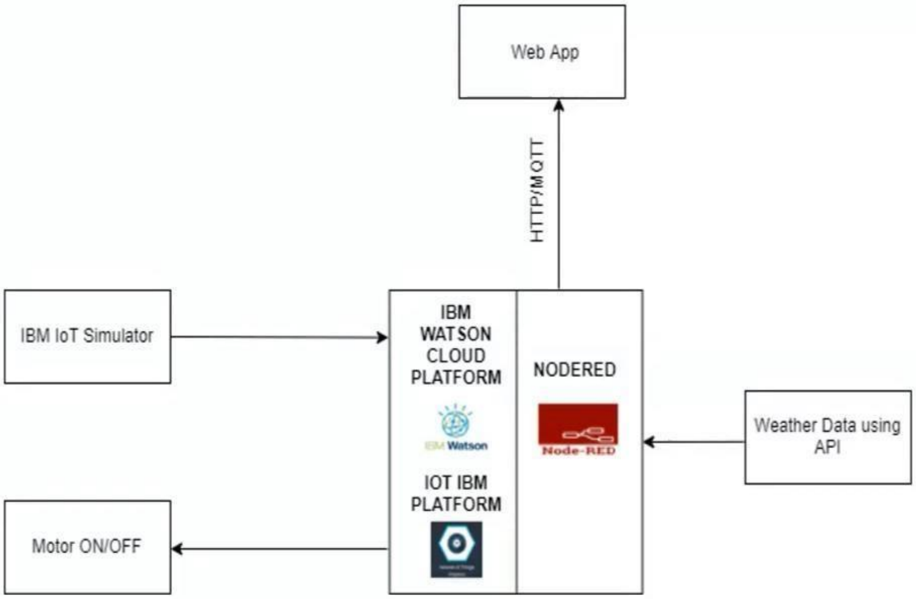
2.1 Proposed Solution

- To develop a Smart Agricultural System based on IOT which can give real time data and can help farmers in a very efficient manner.
- Soil Moisture can be checked by using the sensors that can sense the soil condition and send the data (moisture content in the soil) over the cloud services to the web application.

- The supply of water can be controlled from anywhere by controlling the motor state (ON/OFF), using web application.
- Surrounding temperature can also be sensed by the sensors and displayed on the application.
- Real time weather conditions can also be known by using different weather API's from different websites and displayed on our Application.

3) Theoretical analysis:

3.1 Block Diagram



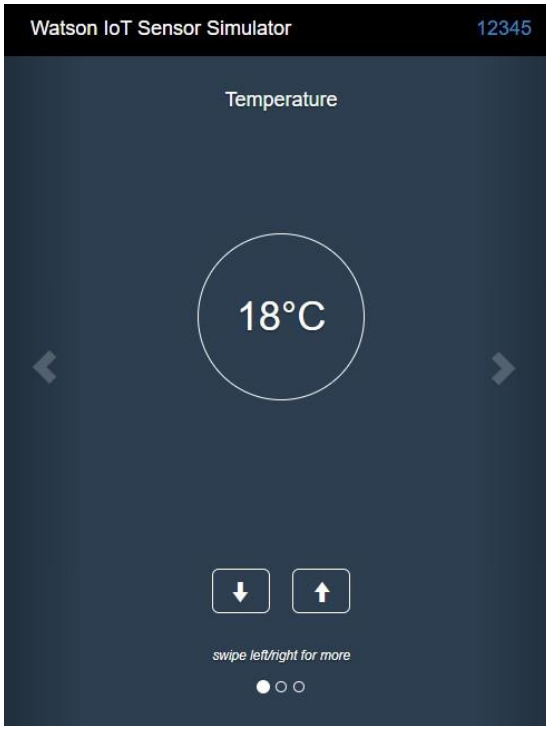
3.2 Hardware / Software Designing

1. Create a device in IBM Cloud.

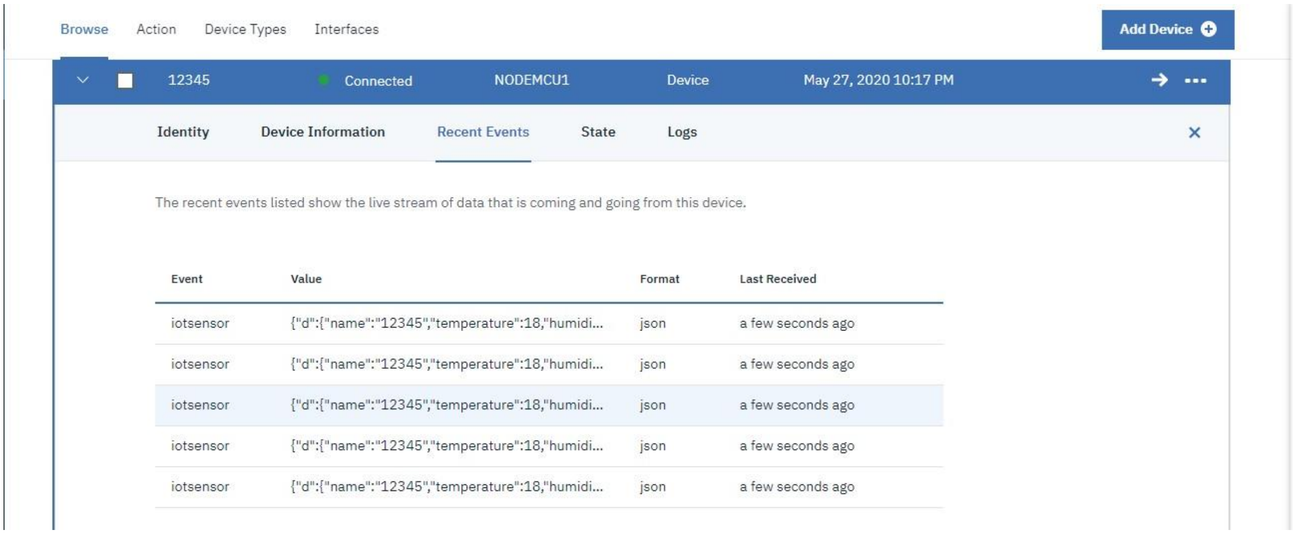
2. Connect the device to IBM Simulator to get the weather conditions.

3. Build Node-RED flow to build a web application to display the weather conditions and control the devices.
4. Get the real time weather condition data from open weather map and integrate it in the Node-RED.
5. Control the working of the web application to the devices by python coding.

4) Experimental Investigation:



Fig(a) Watson Iot Simulator

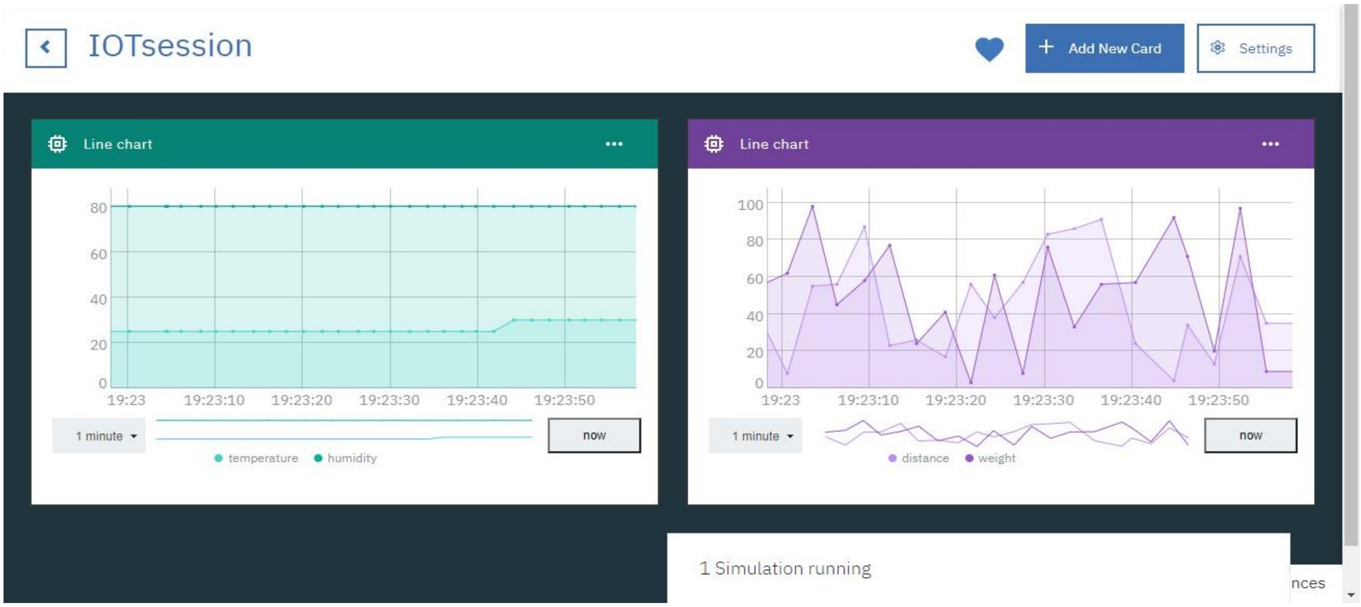


12345 Connected NODEMCU1 Device May 27, 2020 10:17 PM

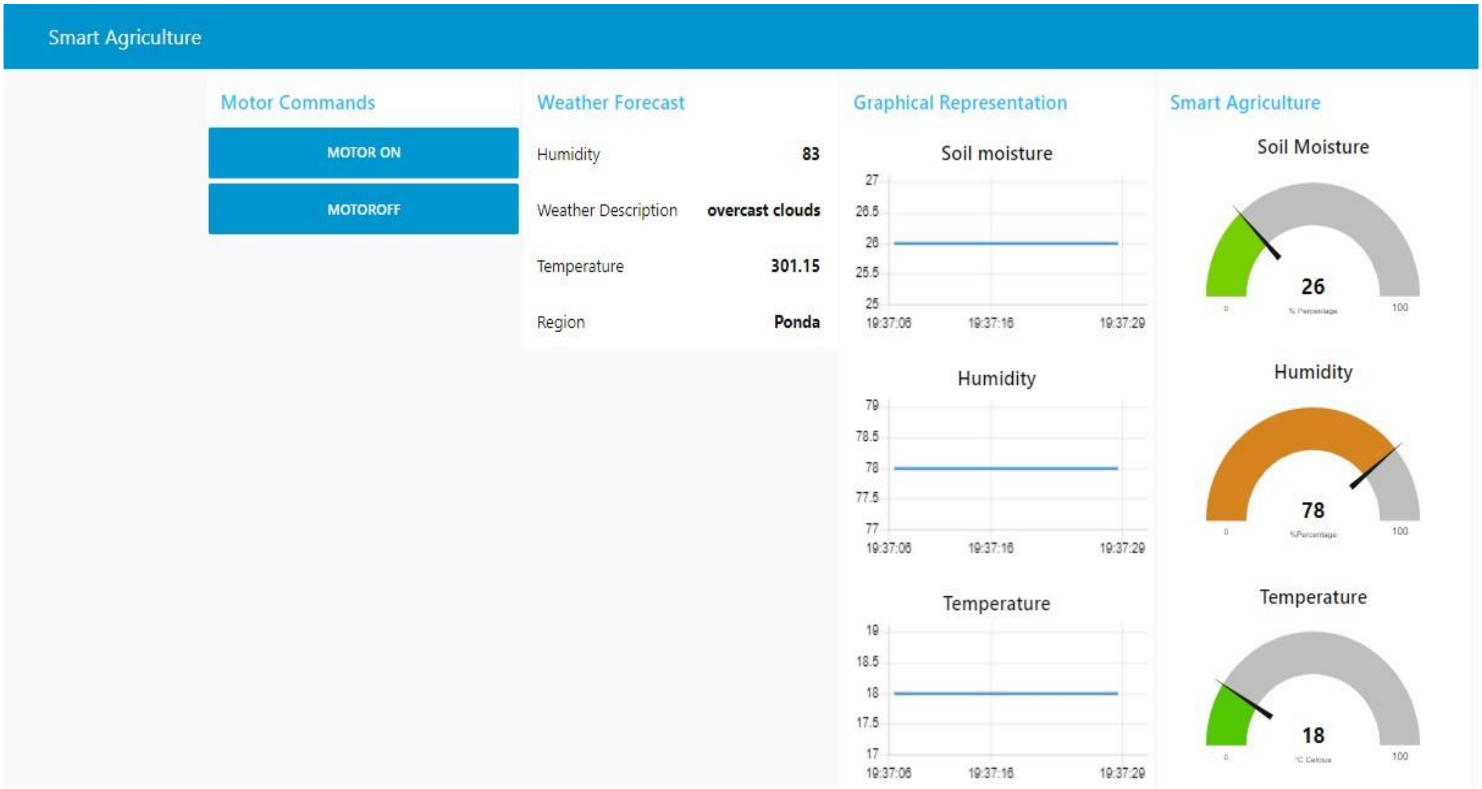
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
iotsensor	{"d":{"name":"12345","temperature":18,"humidi...	json	a few seconds ago
iotsensor	{"d":{"name":"12345","temperature":18,"humidi...	json	a few seconds ago
iotsensor	{"d":{"name":"12345","temperature":18,"humidi...	json	a few seconds ago
iotsensor	{"d":{"name":"12345","temperature":18,"humidi...	json	a few seconds ago
iotsensor	{"d":{"name":"12345","temperature":18,"humidi...	json	a few seconds ago

Fig(b)Receiving data from Iot simulator to the IBM watson IOT platform through IBM cloud

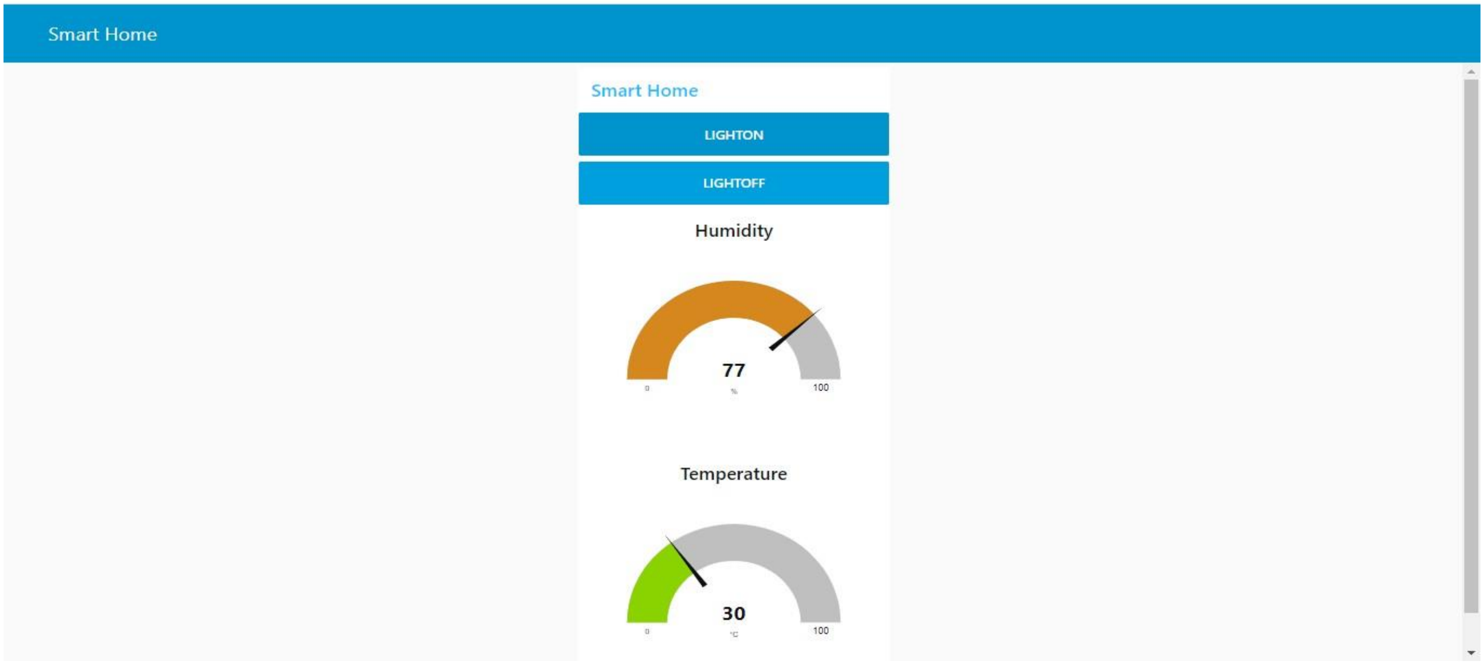


Fig(c)Representation of Received data through graphs

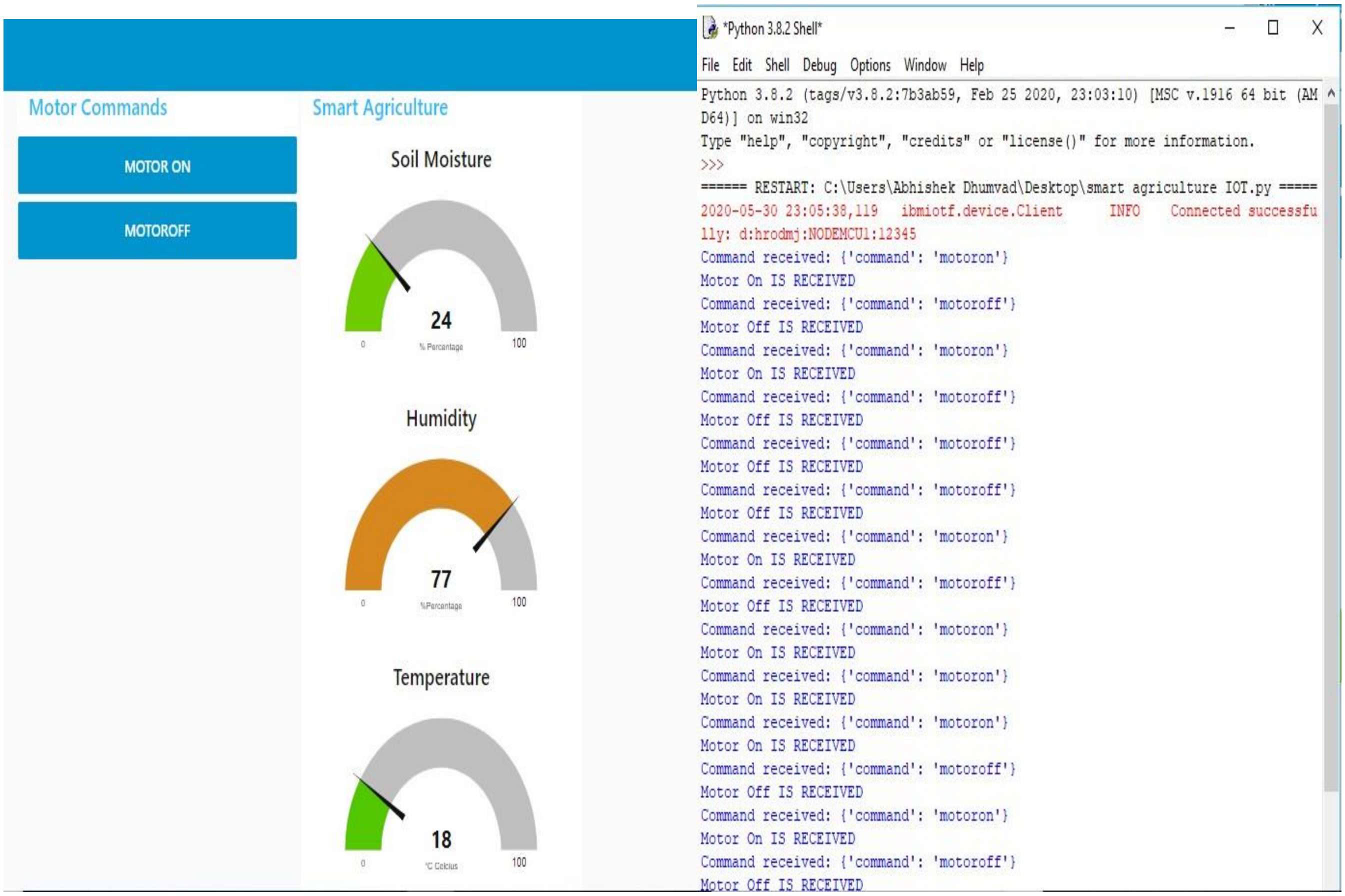


Humidity	83
Weather Description	overcast clouds
Temperature	301.15
Region	Ponda

fig(d) Web Application for Smart Agriculture System



Fig(e) An illustration of Web App - Smart Home



```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\Users\Abhishek Dhumvad\Desktop\smart agriculture IOT.py =====
2020-05-30 23:05:38,119   ibmiotf.device.Client      INFO    Connected successfu
lly: d:hrodmj:NODEMCU1:12345
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
Command received: {'command': 'motoroff'}
Motor Off 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
Command received: {'command': 'motoron'}
Motor On IS RECEIVED
Command received: {'command': 'motoron'}
Motor On IS RECEIVED
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
```

fig(f) Device Control Action

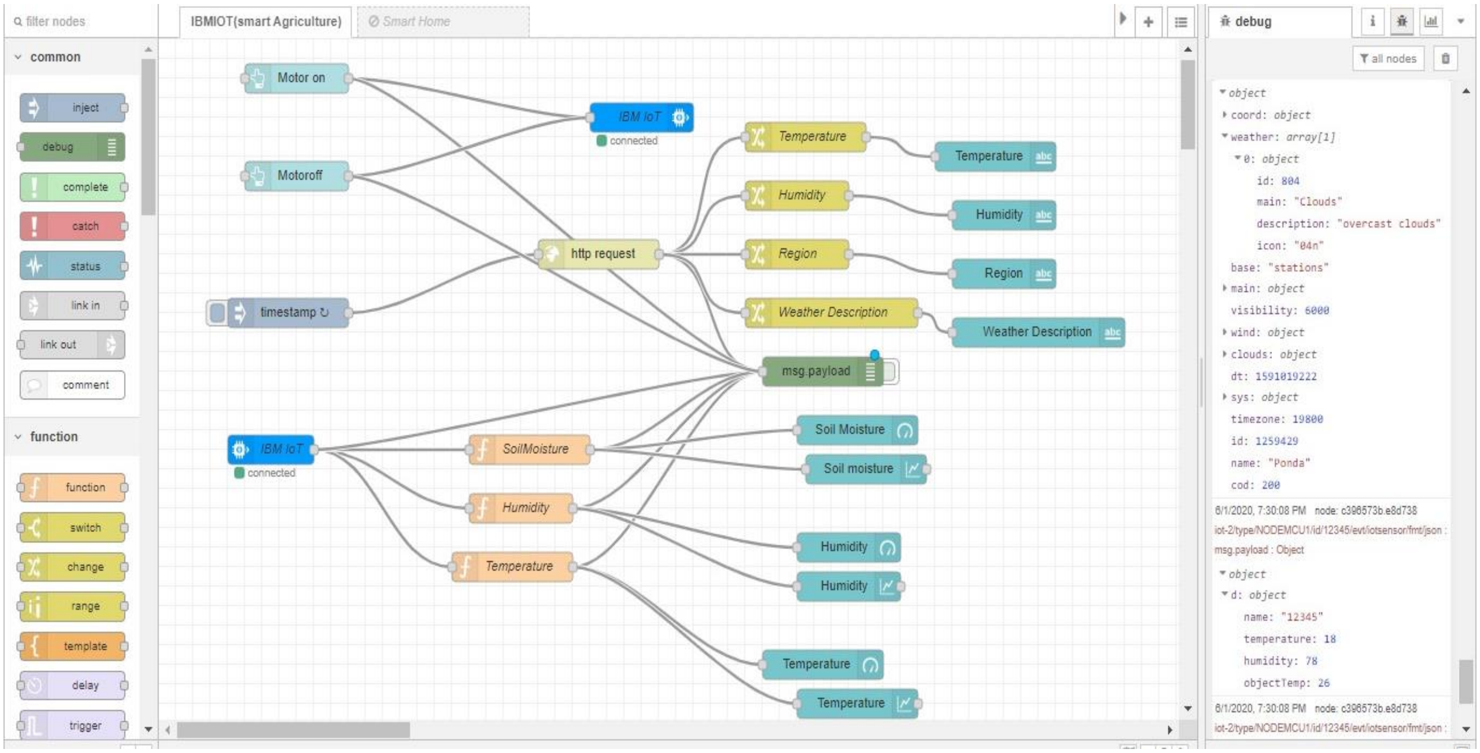
In this project we send the weather data through IoT Simulator shown in fig(a) instead of real soil and temperature conditions. Simulator passes the data through IBM Cloud to the web application. The data is displayed on the dashboard show in fig(b1 & b2). Web Application is build using Node-RED. We have created 2 tabs:

1. IoT Smart Agriculture.

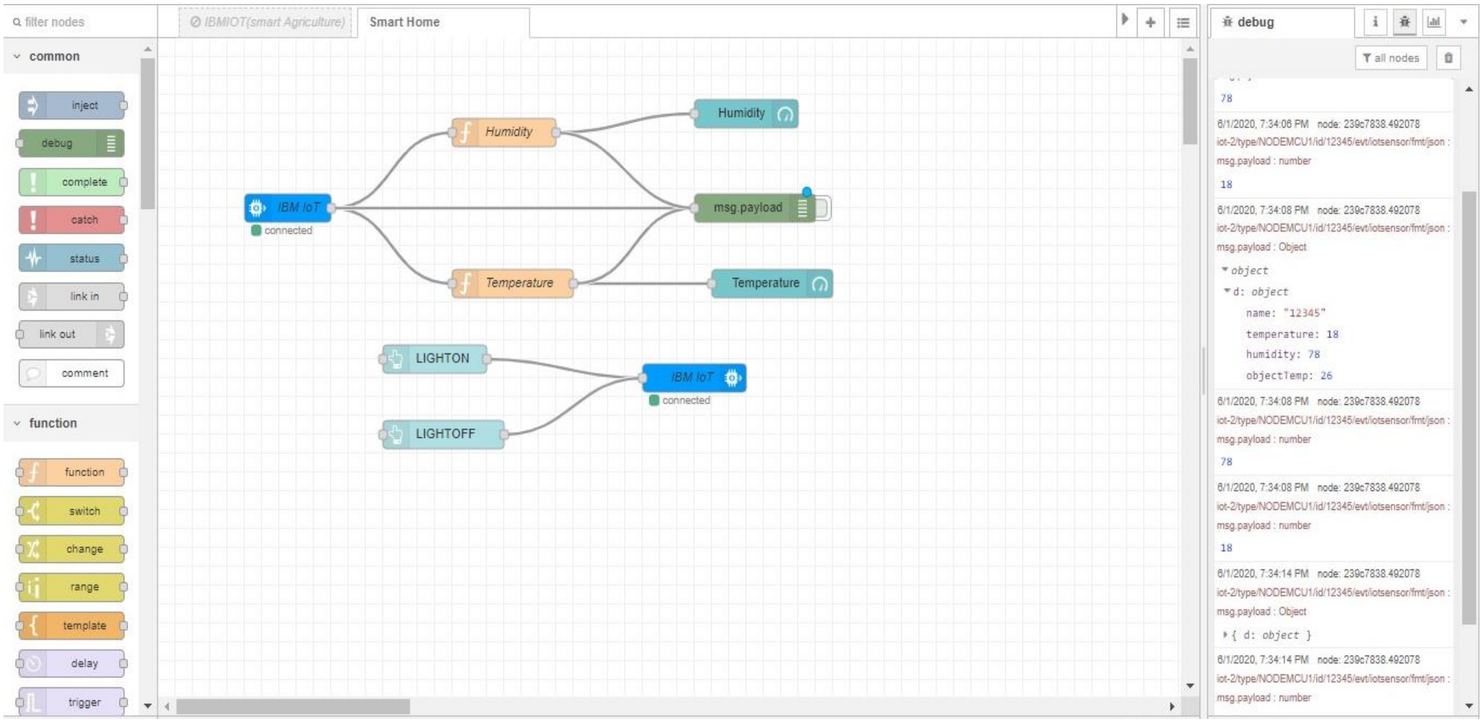
2. Graphical Representation.

Web Application is also used to control the devices further like motor, pumps, lights, or any other devices in the agricultural field. In this project the output is passed using python code and the control action is displayed in python code console window in fig(c).

5) Flowchart:



```
▾object
 ▸coord: object
 ▾weather: array[1]
   ▾0: object
      id: 804
      main: "Clouds"
      description: "overcast clouds"
      icon: "04n"
  base: "stations"
 ▸main: object
  visibility: 6000
 ▸wind: object
 ▸clouds: object
  dt: 1591019222
 ▸sys: object
  timezone: 19800
  id: 1259429
  name: "Ponda"
  cod: 200
6/1/2020, 7:30:08 PM   node: c396573b.e8d738
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : Object
▾object
 ▾d: object
    name: "12345"
    temperature: 18
    humidity: 78
    objectTemp: 26
6/1/2020, 7:30:08 PM   node: c396573b.e8d738
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
```



```
78
6/1/2020, 7:34:06 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : number
18
6/1/2020, 7:34:08 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : Object
▾object
 ▾d: object
    name: "12345"
    temperature: 18
    humidity: 78
    objectTemp: 26
6/1/2020, 7:34:08 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : number
78
6/1/2020, 7:34:08 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : number
18
6/1/2020, 7:34:14 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : Object
 ▸{ d: object }
6/1/2020, 7:34:14 PM   node: 239c7838.492078
iot-2/type/NODEMCU1/id/12345/evt/iotsensor/fmt/json :
msg.payload : number
```

Following are the nodes used in the project in the Web Application:

1. IBM IoT : IN and OUT Nodes.
2. function Nodes.
3. Gauge Nodes.
4. Chart Nodes.
5. Debug Node.
6. Button Nodes.

Following are the nodes used for the weather condition from open weather map:

1. Timestamp Node.
2. http request Node
3. Function Nodes.
4. Text Nodes.
5. Debug Node.

6) Result:

We have successfully built a web-based UI and integrated all the services using Node-RED

7) Advantages & Disadvantages:

7.1 Advantages

- All the data like climatic conditions and changes in them, soil or crop conditions everything can be easily monitored.
- Risk of crop damage can be lowered to a greater extent.
- Many difficult challenges can be avoided making the process automated and the quality of crops can be maintained.
- The process included in farming can be controlled using the web applications from anywhere, anytime.

7.1 Disadvantages

- Smart Agriculture requires internet connectivity continuously, but rural parts can not fulfill this requirement.
- Any faults in the sensors can cause great loss in the agriculture, due to wrong records and the actions of automated processes.
- IoT devices need much money to implement.

8) Applications:

- Precision Farming that is farming processes can be made more controlled and accurate.
- Live monitoring can be done of all the processes and the conditions on the agricultural field.
- All the controls can be made just on the click.
- Quality can be maintained

9) Conclusion:

An IoT Web Application is built for smart agricultural system using Watson IoT platform, Watson simulator, IBM cloud and Node-RED.

10)Future Scope:

In future due to more demand of good and more farming in less time, for betterment of the crops and reducing the usage of extravagant resources like electricity and water IoT can be implemented in most of the places.

11) Bibliography:

- IBM Cloud :https://cloud.ibm.com/docs/overview?topic=overview-whatis platform
- Watson IoT :https://www.iotone.com/software/ibm-watson-iotplatform/s62
- Open weather map: https://openweathermap.org/
- GitHUB: https://github.com/SmartPracticeschool/llSPS-INT-836-Smart-Agriculture-system-based-on-IoT.git
- Node-RED:
  https://www.youtube.com/watch?v=cicTw4SEdxk
  https://nodered.org/docs/getting-started/windows#3-run nodered

12 Appendix:

A. Source Code

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

#Provide your IBM Watson Device Credentials
organization = "hrodmj" #replace the ORG ID deviceType
= "NODEMCU1"#replace the Device type wi
deviceId = "12345"#replace Device ID authMethod
= "token"
authToken = "abhi1234" #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()
```