Smart Agricultural System Based On IoT

By: NOEL PAUL FREDY

VELLORE INSTITUTE OF TECHNOLOGY

BHOPAL

CONTENTS

1. Introduction

The main aim of this project is to help farmers automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity and etc, and control the equipment like water motor and other devices remotely via the internet without their actual presence in the field.

2. Problem Statement

Farmers are to be present at the farm for its maintenance irrespective of the weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically. Farmers have to stay most of the time in the field in order to get a good yield. In difficult times like in the presence of pandemic also they have to work hard in their areas risking their lives to provide food for the country.

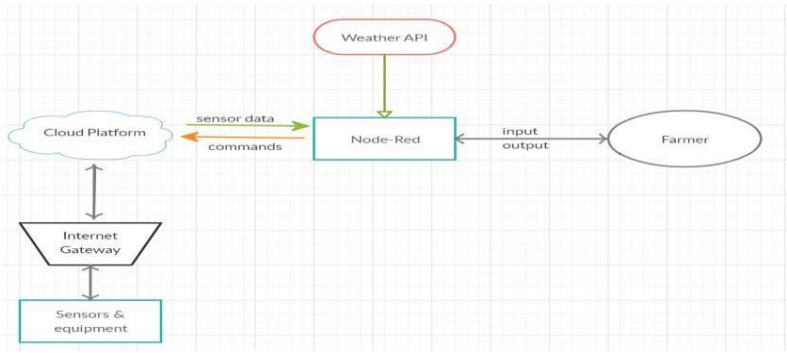
3. Proposed Solution

In order to improve the farmer's working conditions and make them more manageable, we introduce IoT services to him in which we use cloud services and the internet to enable the farmer to continue his work remotely via the internet. He can monitor the field parameters and control the devices on the farm.

4. Theoretical Analysis

4.1 Block Diagram

In order to implement the solution, the following approach as shown in the block diagram is used

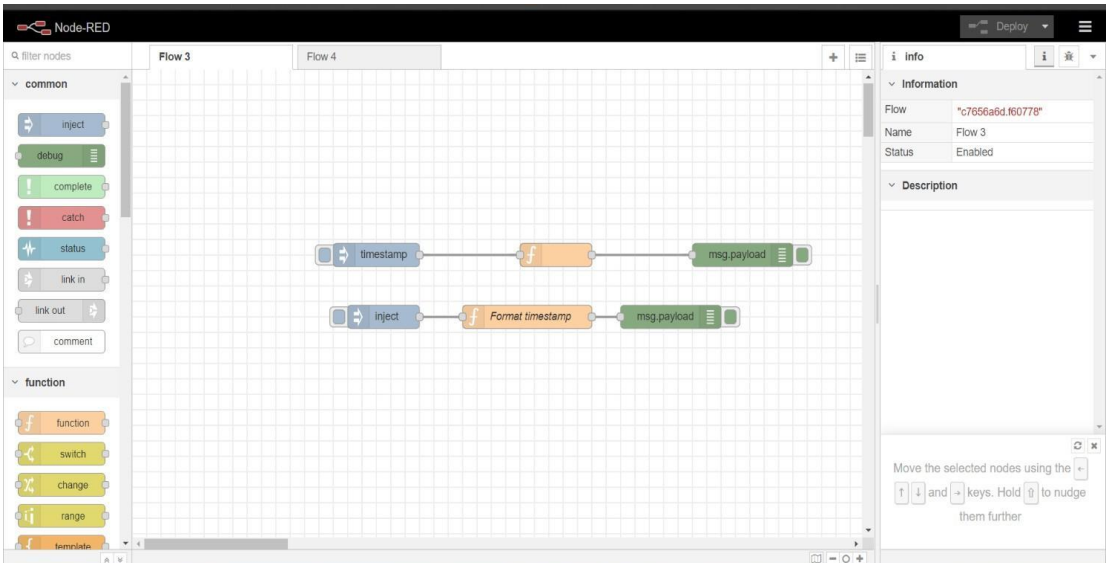


4.2 Required Software installation Node-Red

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.
Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.

Installation :

- First install npm/node.js
- Open cmd prompt
- Type => npm install node-red



To run the application:

- Open cmd prompt
- Type=> node-red
- Then open http://localhost:1880/ in browser

Installation of IBM IoT and Dashboard nodes for Node-Red

In order to connect to IBM Watson IoT platform and create the Web App UI these nodes are required

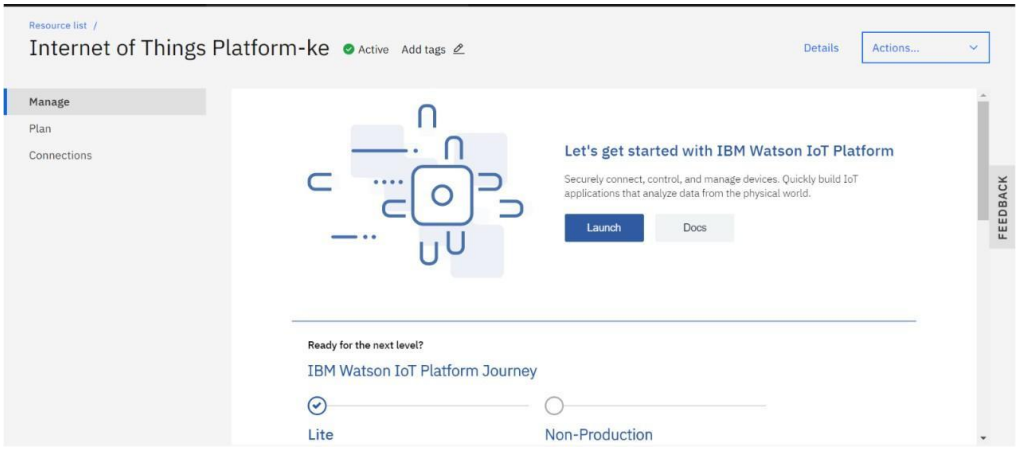
1. IBM IoT node
2. Dashboard node

4.3 IBM Watson IoT Platform

A fully managed cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization, and data storage. IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices

Steps to configure:

- Create an account in IBM cloud using your email ID



- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.

4.4 Python IDE

Install Python3 compiler

4.5 IoT Simulator

In our project in the place of sensors we are going to use IoT sensor simulator which gives random readings to the connected cloud.
The link to simulator: https://watson-iot-sensor-simulator.mybluemix.net/

We need to give the credentials of the created device in IBM Watson IoT Platform to connect cloud to a simulator.

OpenWeather API

OpenWeatherMap is an online service that provides weather data. It provides current weather data, forecasts, and historical data to more than 2 million customers.

Website link: https://openweathermap.org/guide

Steps to configure:

- Create an account in OpenWeather
- Find the name of your city by searching
- Create API key to your account
- Replace "city name" and "your API key" with your city and API key in below red text

api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}

Link I used in my project:

http://api.openweathermap.org/data/2.5/weather?q=Gudur,in&appid=62354068e45f41ffa6a5b164714145fe

5. Building Project

5.1 Connecting IoT Simulator to IBM Watson IoT Platform

Open the link provided in the above section 4.3

Give the credentials of your device in IBM Watson

IoT Platform Click on connect

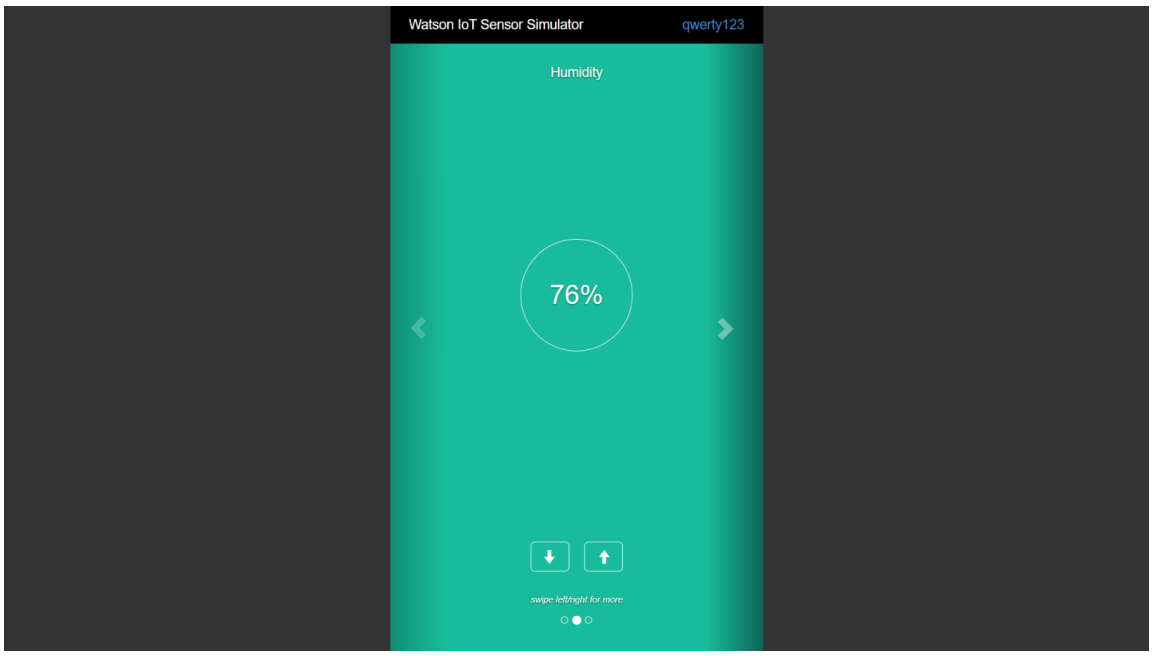
My credentials given to simulator are:

OrgID: 9wbx5 API: a-9wbx5m-1qfklrf7jl

Device type: iotdevice1 token:

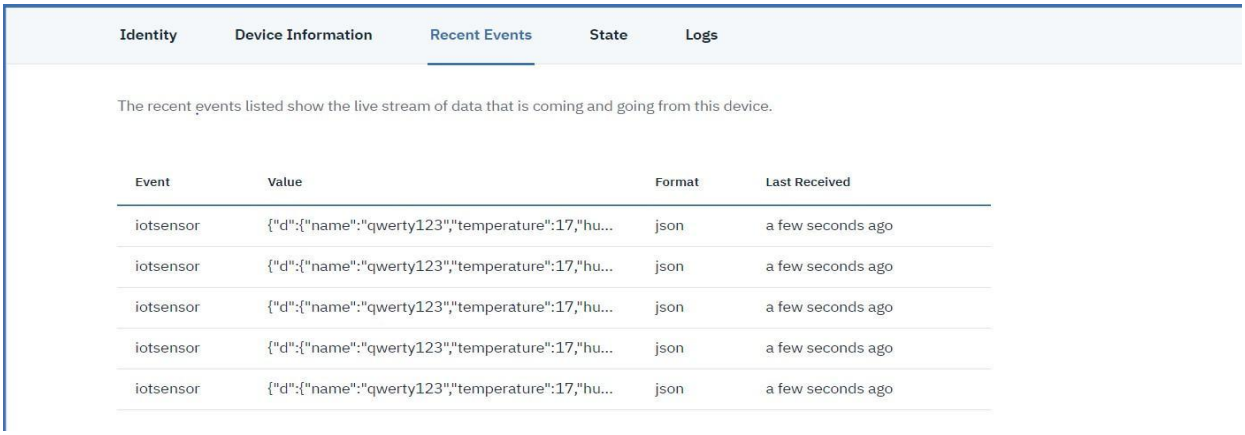
JcU(4(9Z37PdL!Rmz(Device ID : qwerty123

Device Token : apple

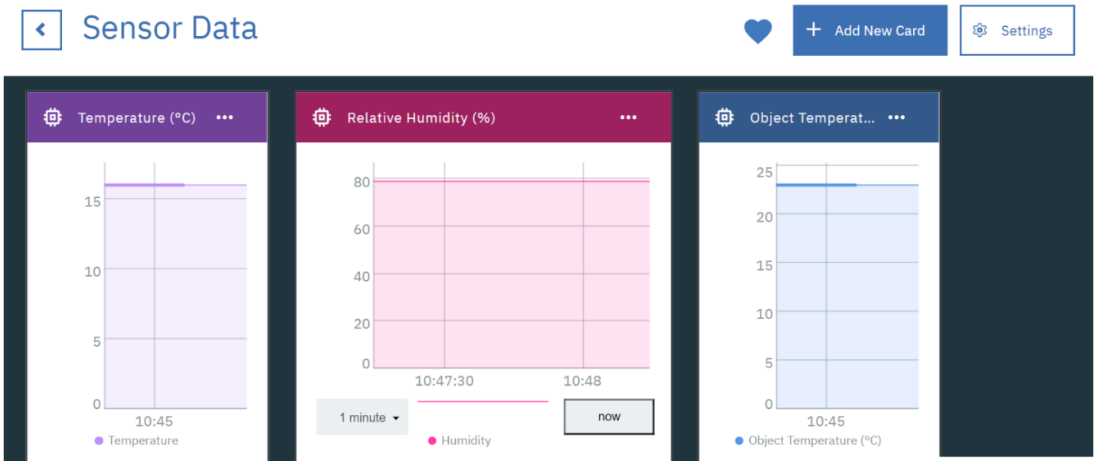


You will receive the simulator data in cloud
You can see the received data in Recent Events under your device
Data received in this format(json)

- {
  - "d": {
    - "name": "qwerty123",
    - "temperature": 17,
    - "humidity": 76,
    - "objectTemp": 25
  - }
- }

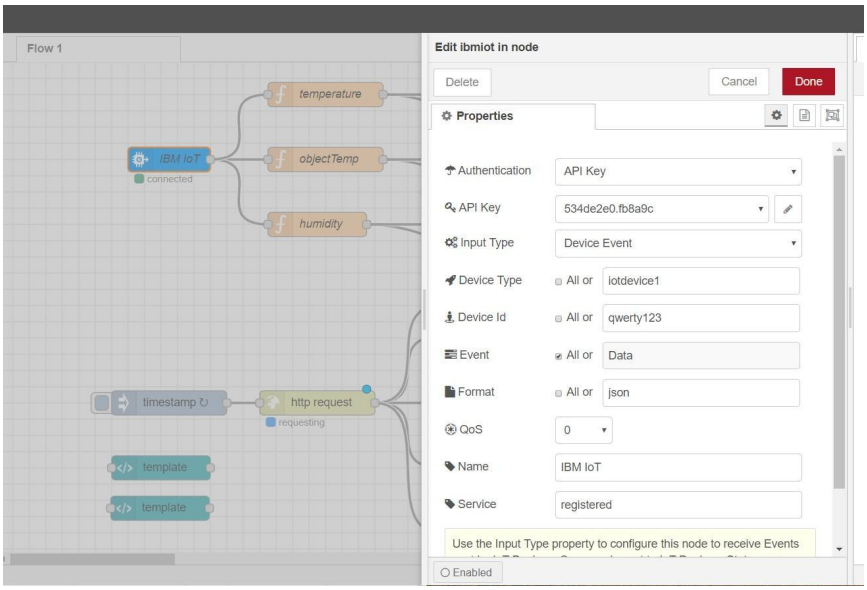


You can see the received data in graphs by creating cards in Boards



5.2 Configuration of Node-Red to collect IBM cloud data

The node IBM IoT App In is added to Node-Red workflow. Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red

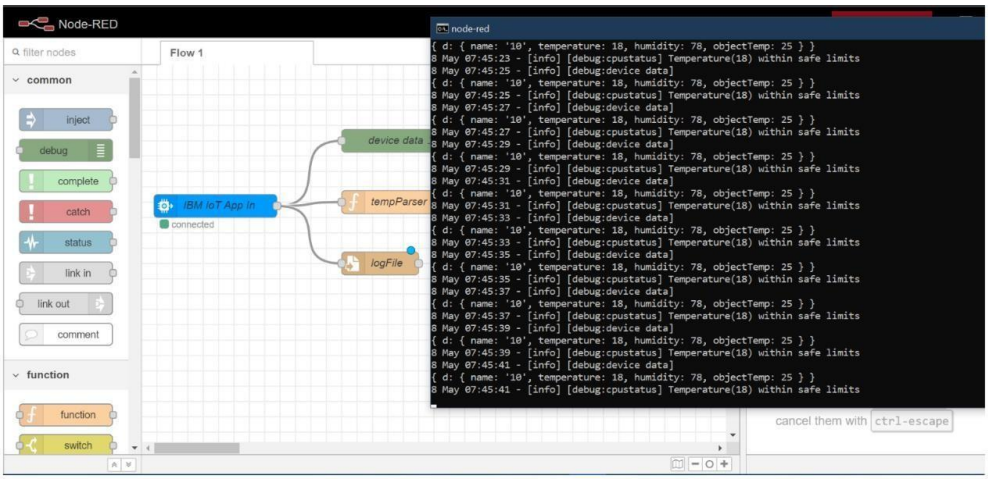


Once it is connected Node-Red receives data from the device and Displays the data using debug node for verification

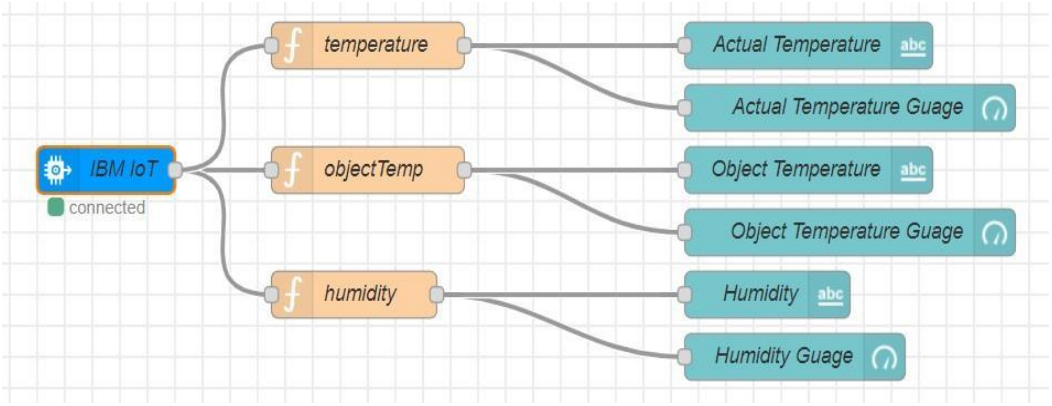
Connect the function node and write the Javascript code to get each reading separately. The Javascript code for the function node is

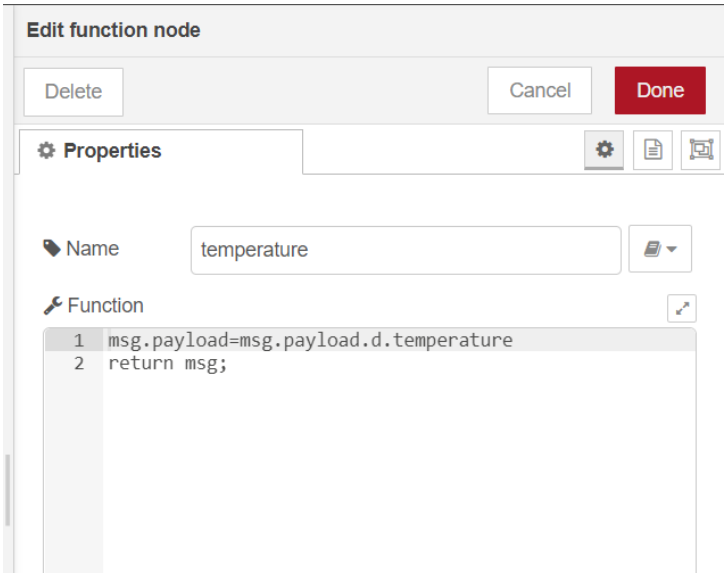
- msg.payload=msg.payload.d.tempera
  ture return msg;

Finally connect Gauge nodes from the dashboard to see the data in UI



Data received from the cloud in Node-RED console





This is the Java script code I have written for the function node to get Temperature separately

5.3 Configuration of Node-Red to collect data from OpenWeather

The Node-Red also receives data from the OpenWeather API by HTTP GET request. An inject trigger is added to perform HTTP requests for every certain interval.

HTTP request node is configured with the URL we saved before in section 4.4

The data we receive from OpenWeather after the request is in below JSON format:

{"coord":{"lon":79.85,"lat":14.13},"weather":[{"id":803,"main":"Clouds","description":"bro
ken
clouds","icon":"04n"}],"base":"stations","main":{"temp":307.59,"feels_like":305.5

,"temp_min":307.
59,"temp_max":307.59,"pressure":1002,"humidity":35,"sea_level":1002,"grnd_lev
el":1000},"wind":{
"speed":6.23,"deg":170},"clouds":{"all":68},"dt":1589991979,"sys":{"country":"I
N","sunrise":158993

3553,"sunset":1589979720},"timezone":19800,"id":1270791,"name":"Gūdūr","cod":200}

In order to parse the JSON string we use Java script functions and get

each parameters var temperature = msg.payload.main.temp;

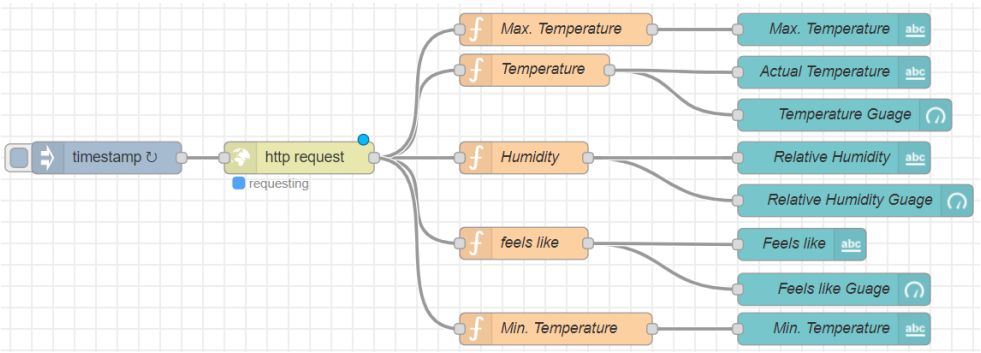
temperature = temperature-273.15;

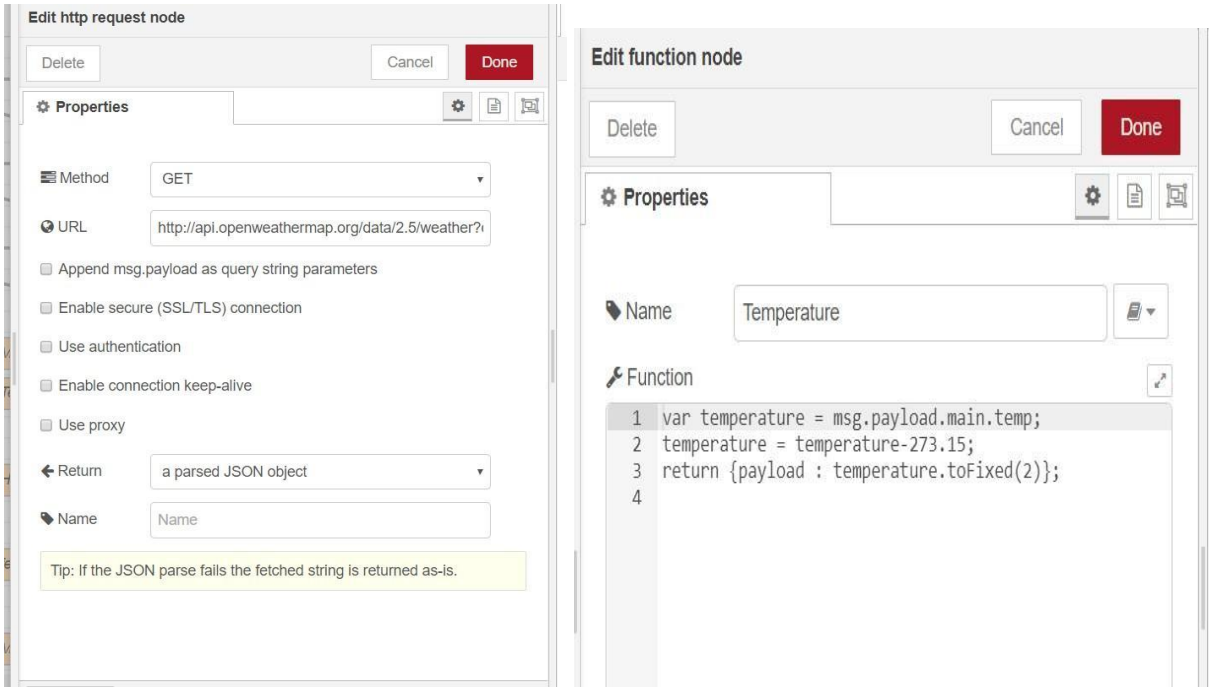
return {payload :

temperature.toFixed(2)};

In the above Java script code we take temperature parameter into a new variable and convert it from kelvin to Celsius

Then we add Gauge and text nodes to represent data visually in UI

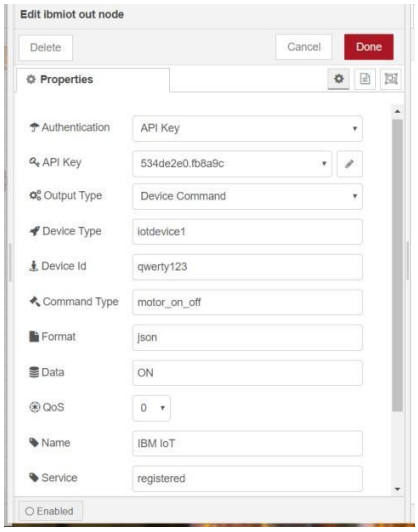




The above two images contain http request and function node data that needs to be filled.

5.4 Configuration of Node-Red to send commands to IBM cloud

ibmiot out node I used to send data from Node-Red to IBM Watson device. So, after adding it to the flow we need to configure it with the credentials of our Watson device.



Here we add three buttons in UI which each sends a number

0,1 and

2. 0 -> for motor off

1-> for motor on

2-> for running motor continuously for 30 minutes

We used a function node to analyze the data received and assign command to each number.

The Javascript code for the analyzer is:

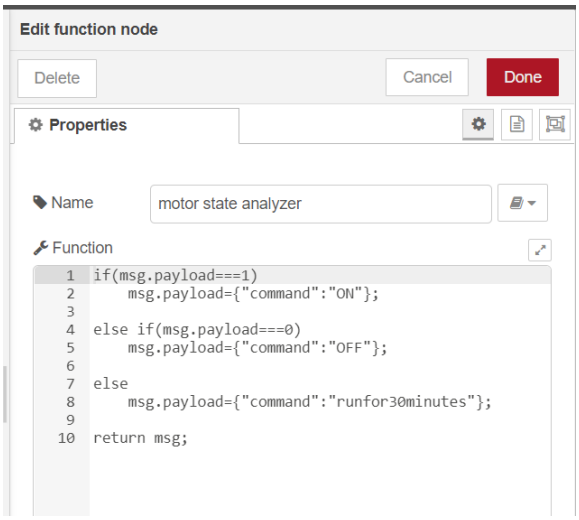
```
if(msg.payload===1)
msg.payload={"command":"ON
"}; else if(msg.payload===0)
msg.payload={"command":"OF
F"}; else
msg.payload={"command":"runfor30mi
nutes"};return msg;
```

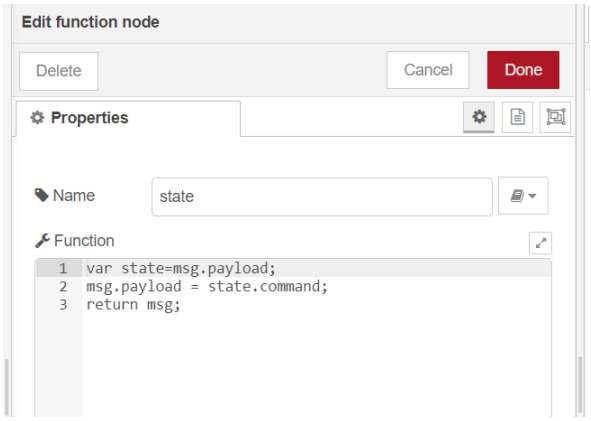
Then we use another function node to parse the data and get the command and represent it visually with text node.
The Java script code for that function

node is:

```
var state=msg.payload;

msg.payload =
state.command; return
msg;
```

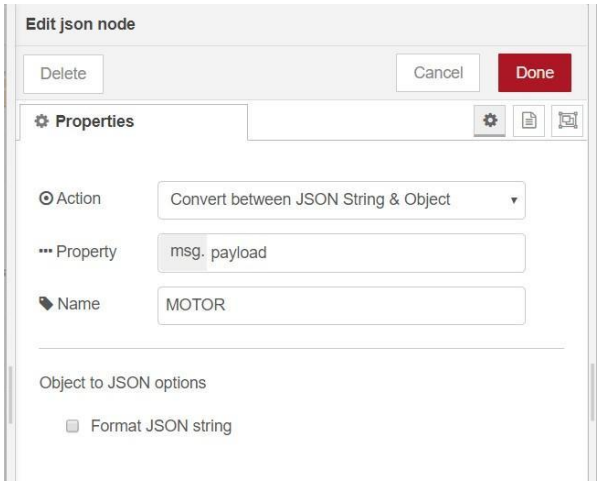
The above images show the java script codes of analyzer and state function nodes.

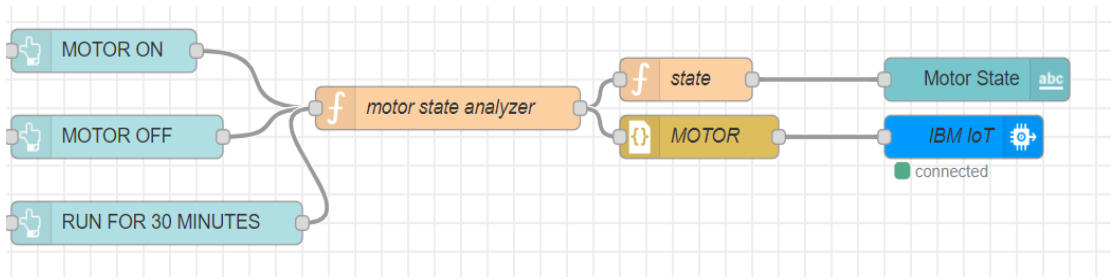




Then we add edit Json node to the conversion between JSON string & object and finally connect it to IBM IoT Out.

Edit JSON node needs to be configured like this





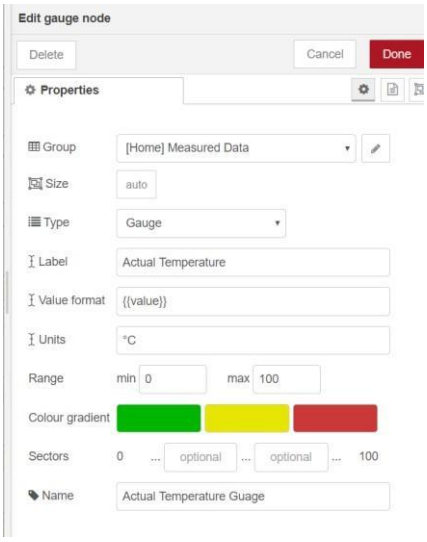
This is the program flow for sending commands to IBM cloud.

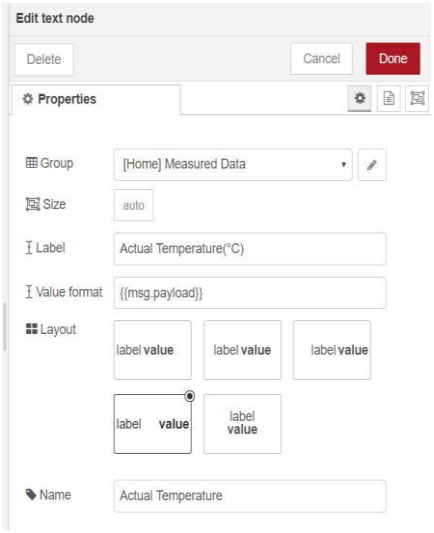
5.5 Adjusting User Interface

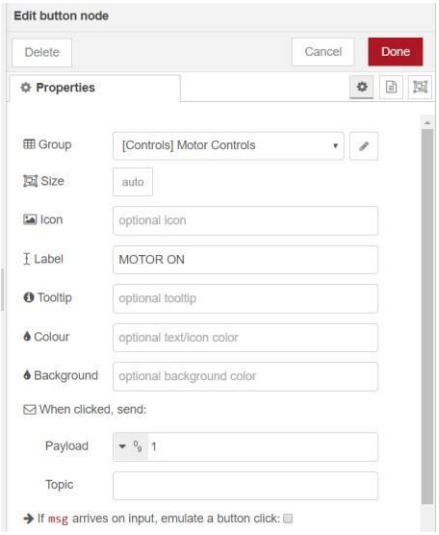
In order to display the parsed JSON data a Node-Red dashboard is created

Here we are using Gauges, text and button nodes to display in the UI and helps to monitor the parameters and control the farm equipment.

Below images are the Gauge, text and button node configurations







5.6 Adding Background image to the UI

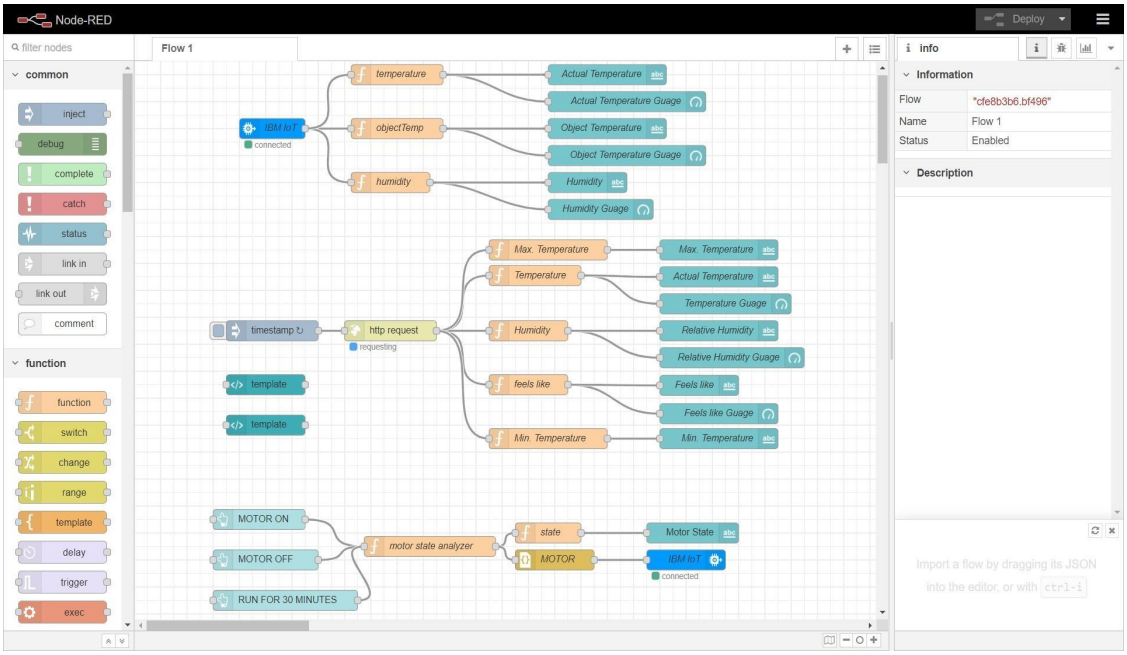
To add the background image we are going to add a template node and configure it with the below HTML code

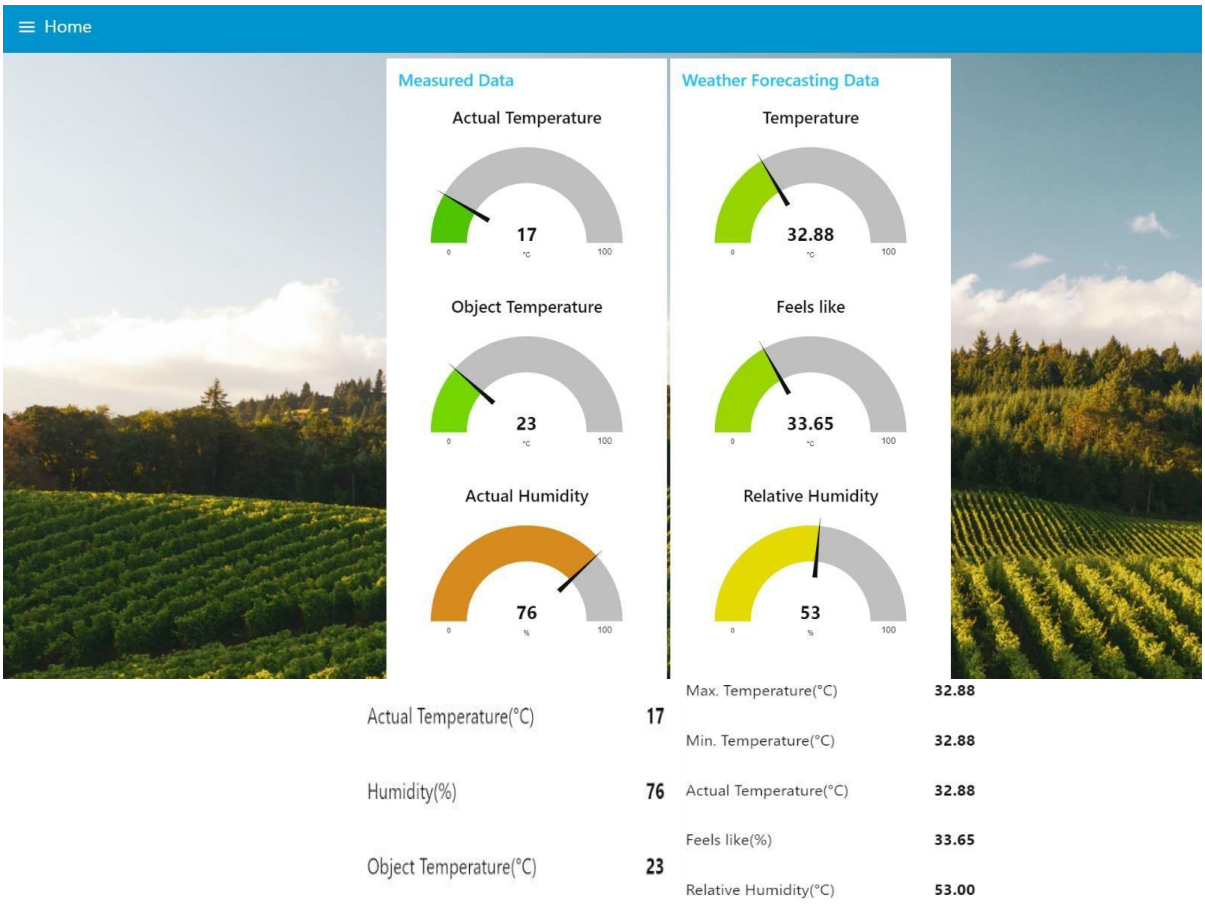
```
<style>
body{
background-image: url("https://images.
unsplash.com/photo-1563514227147-6d2ff665
a6a0?ixlib=rb-1.2.1&auto=format&fit=crop&w
=1951&q=80");
}
</style>
```

Complete Program Flow

Web APP UI

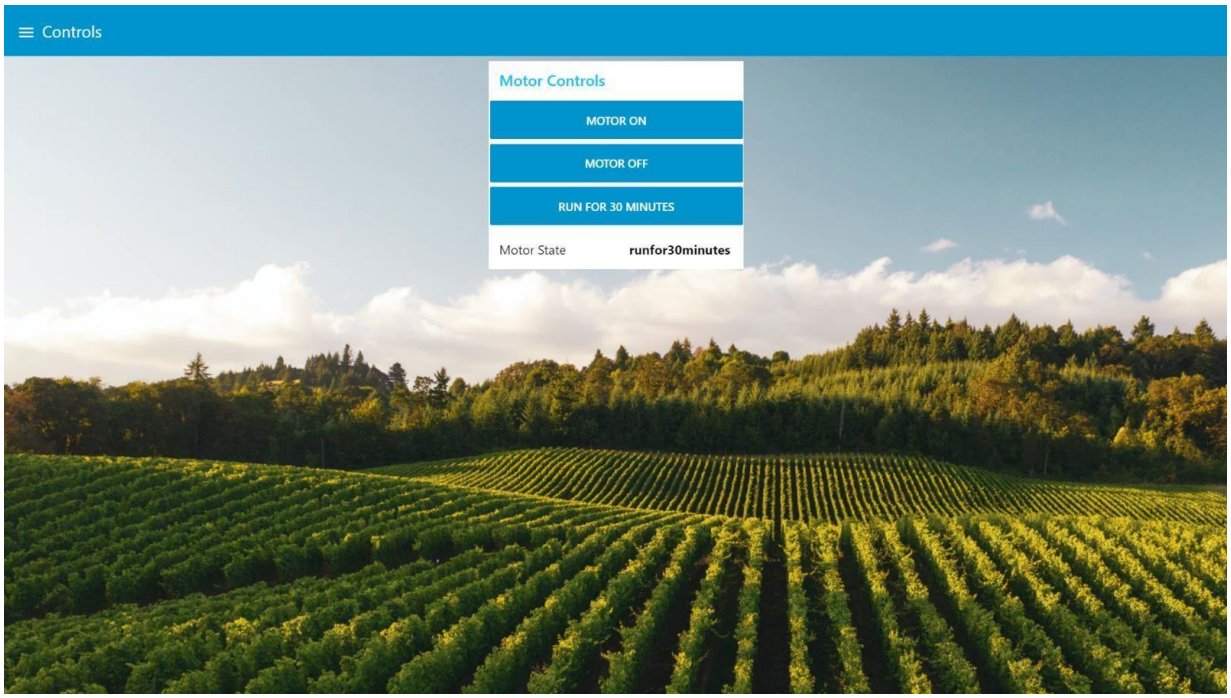
Home Tab





Controls Tab

5.6 Receiving commands from IBM cloud using Python program



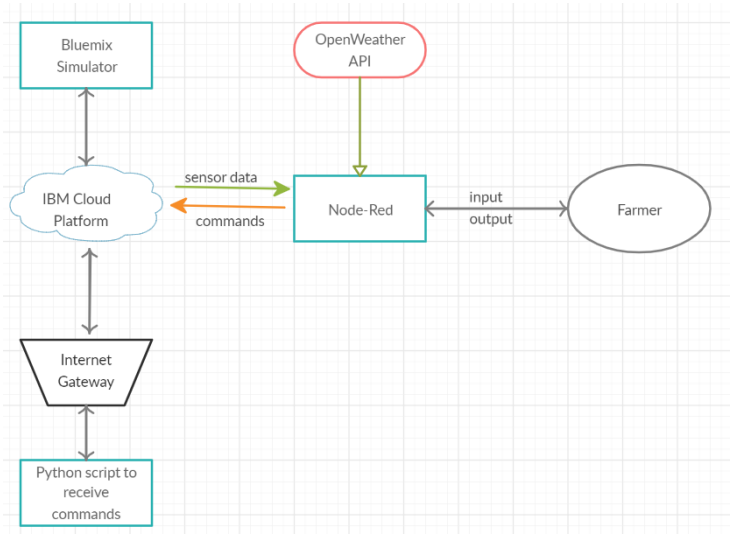
This is the Python code to receive commands from the cloud to any device like Raspberry Pi in the farm

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

#Provide your IBM Watson Device Credentials

organization = "9wbx5m"
deviceType = "iotdevice1"
deviceId = "qwerty123"
authMethod = "token"
authToken = "apple"

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='ON':
                print("MOTOR ON IS RECEIVED")
                time.sleep(1)
                print("MOTOR STARTED")

        elif cmd.data['command']=='OFF':
                print("MOTOR OFF IS RECEIVED")
                time.sleep(1)
                print("MOTOR STOPPED")
```

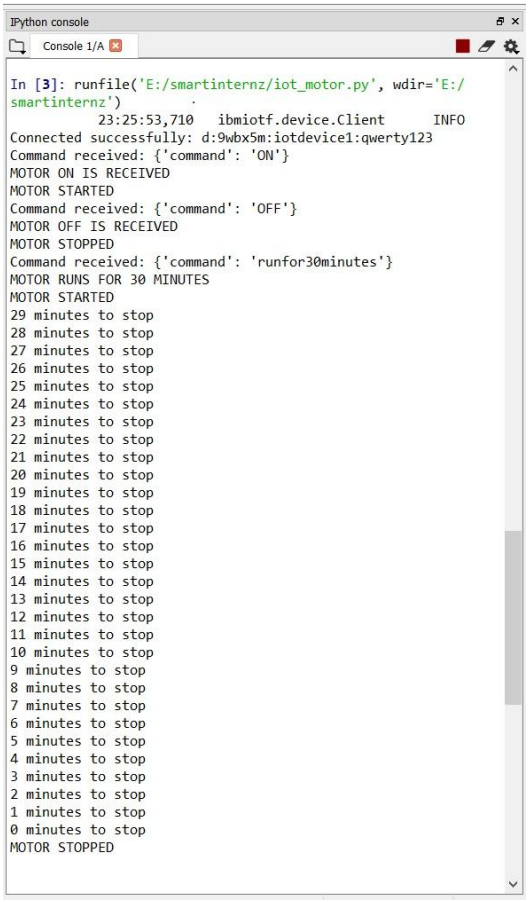
```
        elif cmd.data['command']=='runfor30minutes':
                print("MOTOR RUNS FOR 30 MINUTES")
                print("MOTOR STARTED")
                for i in range(1,31):
                   print("%d minutes to stop"%(30-i))
                print("MOTOR STOPPED")



        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()
```

```
deviceCli.connect()

while True:
        deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
```

6. Flow Chart



7. Observations & Results



8. Advantages & Disadvantages

Advantages:

Farms can be monitored and controlled remotely.
Increase in convenience to farmers.
Less labor cost.
Better standards of living.

Disadvantages:

Lack of internet/connectivity issues.
The added cost of internet and internet gateway infrastructure.
Farmers wanted to adopt the use of WebApp.

9. Conclusion

Thus the objective of the project to implement an IoT system to help farmers control and monitor their farms has been implemented successfully.

10. Bibliography

IBM cloud reference: https://cloud.ibm.com/

IoT simulator : https://watson-iot-sensor-simulator.mybluemix.net/

OpenWeather : https://openweathermap.org/