

Project Report

Smart Irrigation System using Weather Forecasting API

*Carried out as a part of Online Internship
under IoT Application Developer
During June 20201*

At



SMARTBRIDGE
Let's Bridge the Gap

DEVELOPED BY : PROJECT ID : SPS_AP_20210012541

SBID : SB20210142800

VIKAS KUMAR NIGAM

CONTENT

1. Introduction
 - 1.1 Overview
 - 1.2 Purpose
2. Literature Survey
 - 2.1 Existing problem
 - 2.2 Proposed Solution
- 3.Theoretical analysis
 - 3.1Block Diagram
 - 3.2Required Software installation
 - 3.2. a Node-Red
 - 3.2. b IBM Watson IoT Platform
 - 3.2. c Python IDE
 - 3.3 IoT simulator

- 3.4 OpenWeather API
- 4. Building Project
 - 4.1 Connecting IoT Simulator to IBM Watson IoT Platform
 - 4.2 Configuration of Node-Red to collect IBM cloud data
 - 4.3 Configuration of Node-Red to collect data from OpenWeather
 - 4.4 Configuration of Node-Red to send commands to IBM cloud
 - 4.5 Adjusting User Interface
 - 4.6 Receiving commands from IBM cloud using Python program
- 5. Flow chart
- 6. Observations & Results
- 7. Advantages & Disadvantage
- 8. Applications
- 9. Conclusion
- 10. Future Scope
- 11. Bibliography

ACKNOWLEDGMENT:

The success and outcome of this project required guidance and assistance from **Smart Bridge**. I am extremely privileged to have got this all along with the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I thank **Smart Bridge**, for providing me an opportunity to do the project work in the internship **Smart Irrigation System using Weather Forecasting API** and giving me all support and guidance which made me complete the project duly. I am extremely thankful to smart bridge for providing such a nice support and guidance.

Introduction

Overview

Agriculture plays a critical role in the entire life of a given economy. Agriculture is the key development in the rise of a sedentary human civilization. There are various issues that are hampering the development of the country. The possible solutions for the problems faced is to opt for modernized agriculture. Agriculture can be made smart by using Internet of Things (IoT) technologies. Smart Agriculture increases quality, quantity, sustainability and cost- effectiveness of crop production and also analyses the weather conditions. This paper proposes a system which is useful in monitoring the field

data as well as controlling the field operations which provides the flexibility. The paper aims at making agriculture smart using automation and IoT technologies.

Purpose

The Smart Agriculture System is a hi-tech and effective system of doing cultivation and growing food in a sustainable way. It majorly depends on IoT thus eliminating the need of physical work of farmers and growers and thus increasing the productivity in every possible manner. IoT based Smart Agriculture System improves the entire Agriculture system by monitoring the field in real-time. Several great uses for agriculture IoT in this space:

1. Sensing for soil moisture and nutrients.
2. Controlling water usage for optimal plant growth.
3. Reporting weather conditions.

With the help of sensors and interconnectivityinter connectivity, the IoT in Agriculture has not only saves the time of the farmers but also reduces the extravagant use of resources such as water and electricity. It keeps various factors like Humidity, Temperature, Soil Moisture etc. under check and gives a crystal-clear real-time observation.

Literature Survey

Existing Problem

Farmers are to be present at 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. Farmer have to stay most of the time in 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 fields risking their lives to provide food for the country.

Proposed Solution

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

"SMART AGRICULTURE SYSTEM based on IoT" is regarded as the IoT gadget focusing on Live Monitoring of Environmental data in terms of Temperature, Moisture and Humidity of atmosphere and the plant/crop. The system provides the concept of "Plug and Sense" in which farmers can directly implement smart farming by such as putting the System on the field and getting Live Data feeds on various electronic devices using Web Application. Moreover, the data generated via sensors can be easily shared and viewed by agriculture consultants anywhere remotely via Cloud Computing technology integration. The system allows manually to turn the pumping motor ON and OFF on sensing the moisture content of the soil. The Farmer can also use "Run Motor for 30 minutes " Button , where the Motor runs for 30 minutes and turns off automatically with Audio output of Motor turning ON, OFF and Run.

Theoretical Analysis

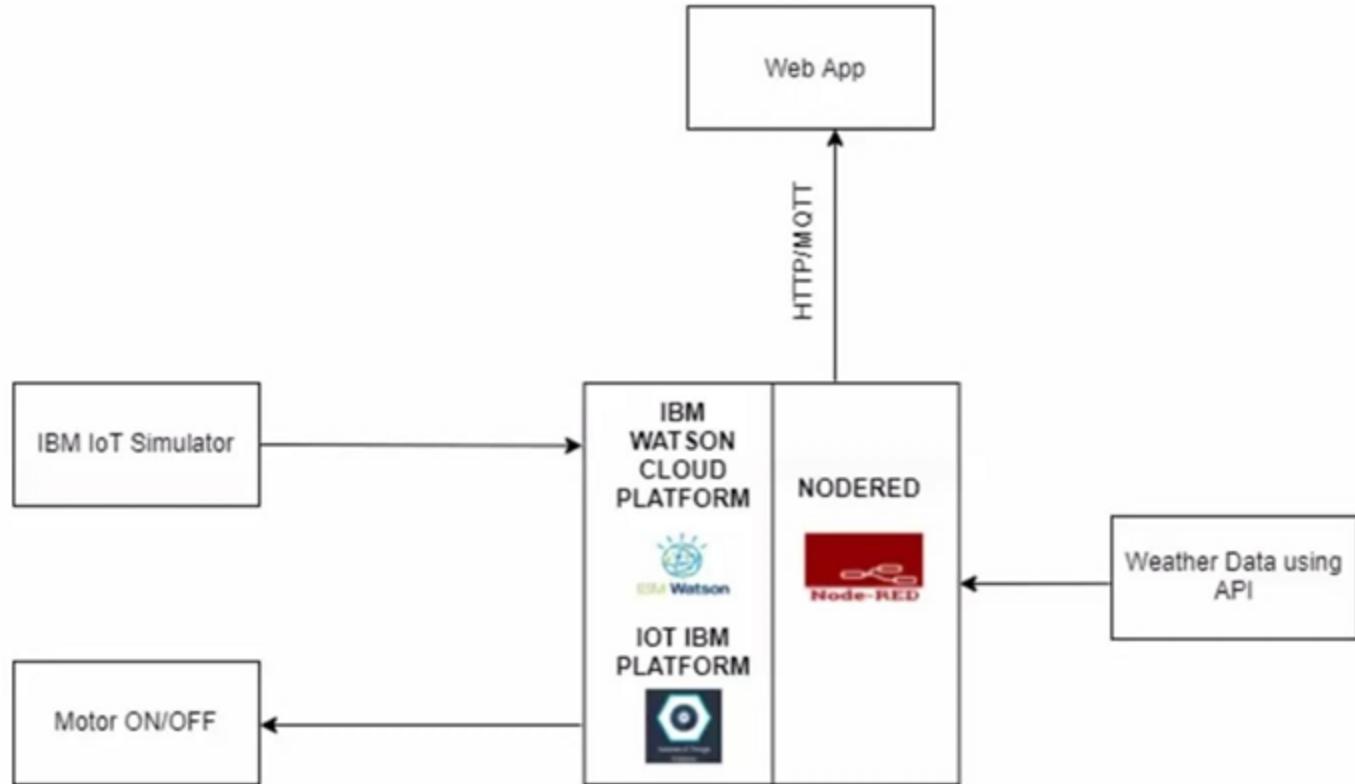
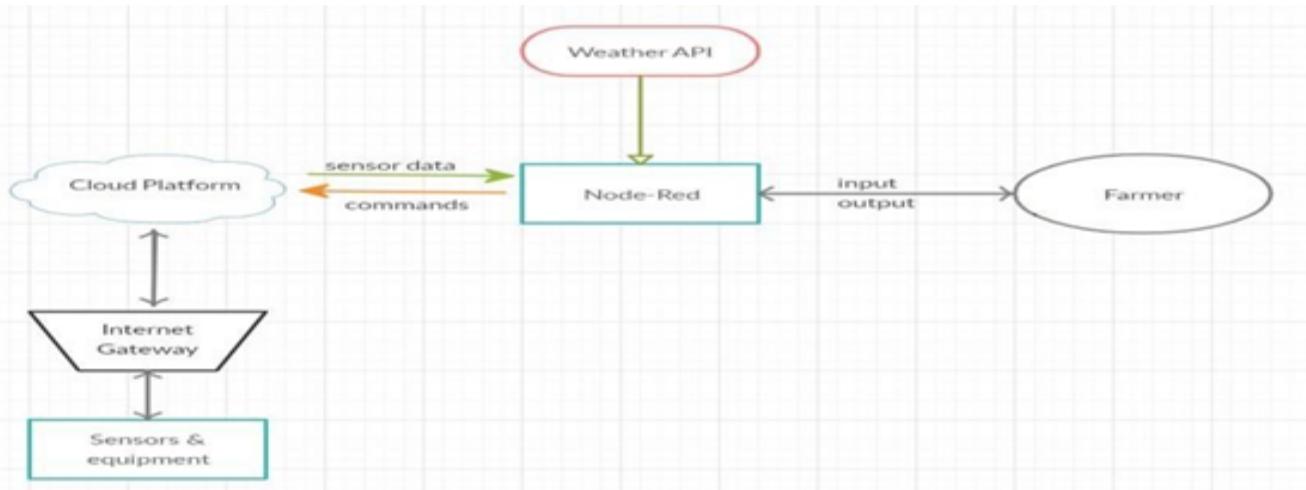
1. **IBM Cloud** is a set of cloud computing services for business offered by the information technology company IBM. Here , IBM platform acts as an interface to control the electronic devices in real time through Watson IoT platform.
2. **The Watson IOT platform** is used when the projects contain hardware such as sensors , electronic devices .

It acts as an interface between devices and IBM Cloud by retrieving data from the devices and sending it to cloud and vice-versa.

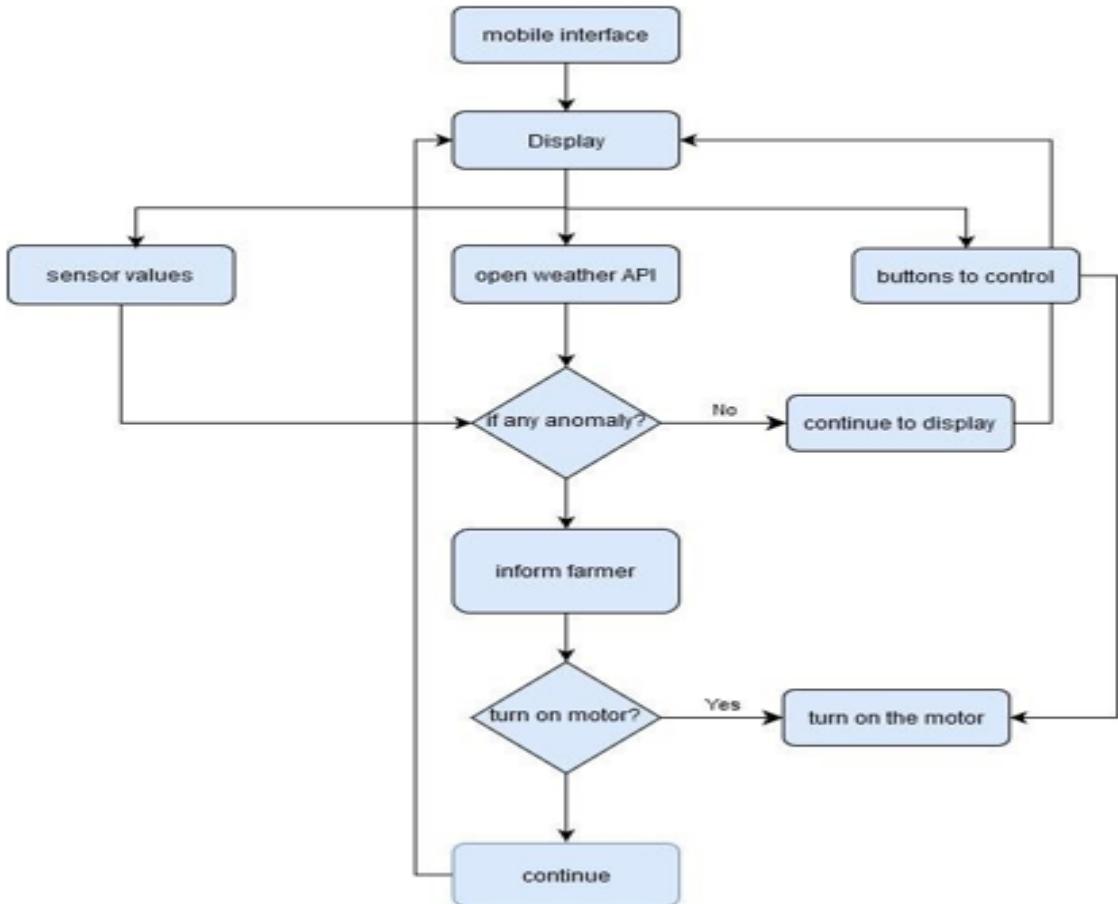
3. The simulated data is sent from **IBM IOT simulator** to IBM Watson cloud platform and then presented in **web interface using http/mqtt web protocols**.
4. To know the real time data of any agricultural farm, we use **Open weather API** (Application Program Interface), which has some predefined APIs.
5. Once the data is received , it is sent to the cloud via **Node-red platform** .
6. The data is monitored continuously (here temperature , humidity and soil temperature) from a web interface.
7. In case of watering crops **motor** can be turned on or off through the web interface controlled through internet.

BlockDiagram

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



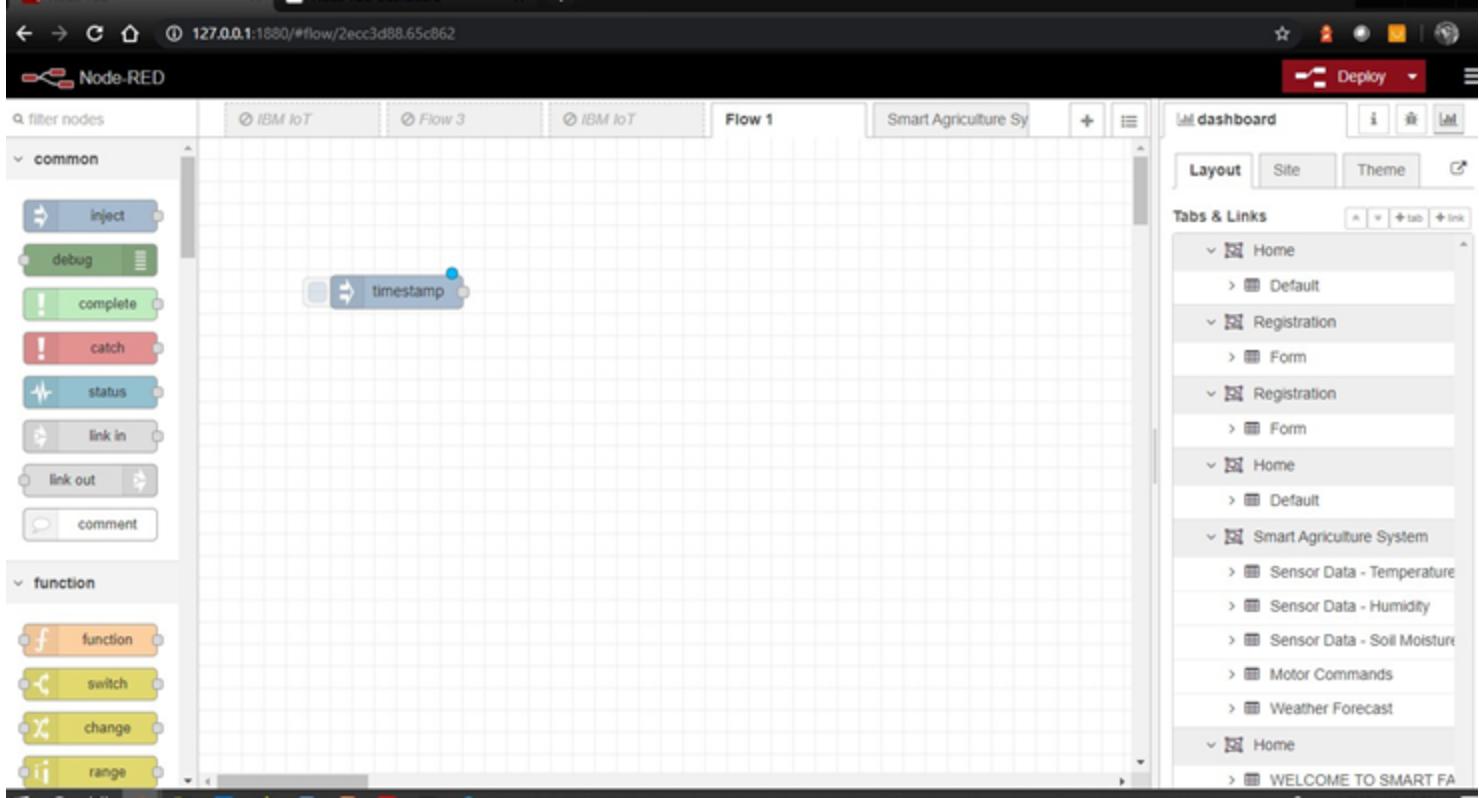
Flowchart:



Required Software Installation

3.2. a 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 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

IBM Watson IoT Platform

Resource list /

Internet of Things Platform-xr

Active

Add tags

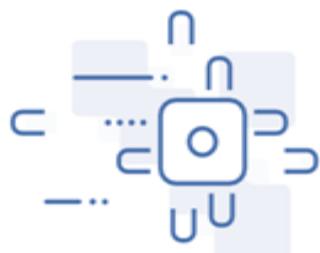
Details

Actions...

Manage

Plan

Connections



Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyse data from the physical world.

[Launch](#)[Docs](#)

Ready for the next level?

IBM Watson IoT Platform Journey



Lite



Non-Production

The Lite service plan provides a lightweightThe Non-Production service plan is a full-

This service is the hub for IBM Watson IoT and lets you communicate with and consume data from connected devices and gateways. Use the built-in web console dashboards to monitor your IoT data and analyze it in real time. Then, enhance and customize your IBM Watson IoT Platform experience by building and connecting your own apps by using messaging and REST APIs.

Offering

Internet of Things Platform

Created

7/9/2021

Created by

vikaskumar.nigam20
19@vitstudent.ac.in

Resource group

Default

Location

London

Status

Active

[Add tags](#)

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

12345 Disconnected VITDevice Jul 9, 2021 11:21 AM

Identity	Device Information	Recent Events	State	Logs
Device ID	12345			
Device Type	VITDevice			
Date Added	Jul 9, 2021 11:21 AM			
Added By	vikaskumar.nigam2019@vitstudent.ac.in			
Connection Status	Disconnected Last Connected: Jul 29, 2021 4:46 PM Client Address: 106.200.193.205 SecureToken Duration: a few seconds Data Transferred: 209 B			

Items per page 50 | 1–1 of 1 item

Python IDE

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used python 3.9.6 shell to execute the code.

```
IDLE Shell 3.9.6
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

IoT Simulator

In our project in the place of sensors we are going to use IoT sensor simulator which give 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 simulator.

Open WeatherAPI

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

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

The screenshot shows the OpenWeatherMap Weather API homepage. At the top, there's a search bar with the placeholder "Weather in your city" and navigation links for "Guide", "API", "Pricing", and "Maps". Below the header, the title "Weather API" is displayed. A note encourages users to sign up for a freemium account to access the entire list of APIs. A section titled "Current & Forecast weather data collection" is highlighted in red. Two main service descriptions are shown: "Current Weather Data" and "Hourly Forecast 4 days", each with "API doc" and "Subscribe" buttons and a list of features.

Please, [sign up](#) to use our fast and easy-to-work weather APIs for free. In our freemium account conditions, you may check the entire list of our [subscriptions](#) and enjoy using our powerful weather APIs right now.

Current & Forecast weather data collection

Current Weather Data

[API doc](#) [Subscribe](#)

- Access current weather data for any location including over 200,000 cities
- We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations

Hourly Forecast 4 days

[API doc](#) [Subscribe](#)

- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- JSON and XML formats
- Included in the Developer, Professional and Enterprise subscriptions

1. Create account in OpenWeather
2. Find the name of your city by searching
3. Create API key to your account
4. 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}&app id={your api key}

Link I used in my project:

URL : api.openweathermap.org/data/2.5/weather?q=Vellore, IN&units=metric&appid=a0bfd7e12cd43d39a44c301354137ea5

Building Project

Connecting IoT Simulator to IBM Watson IoT Platform

From the Web UI go to “ Connect to Watson IoT Simulator” TAB Give the credentials of your device in IBM Watson IoT Platform Click on connect

My credentials given to simulator are:

Organization ID : :

7jh6o2

Device Type : VITDevice

Device ID : 12345

API Key : a-7jh6o2-dzuixifgmn

Authentication Token :

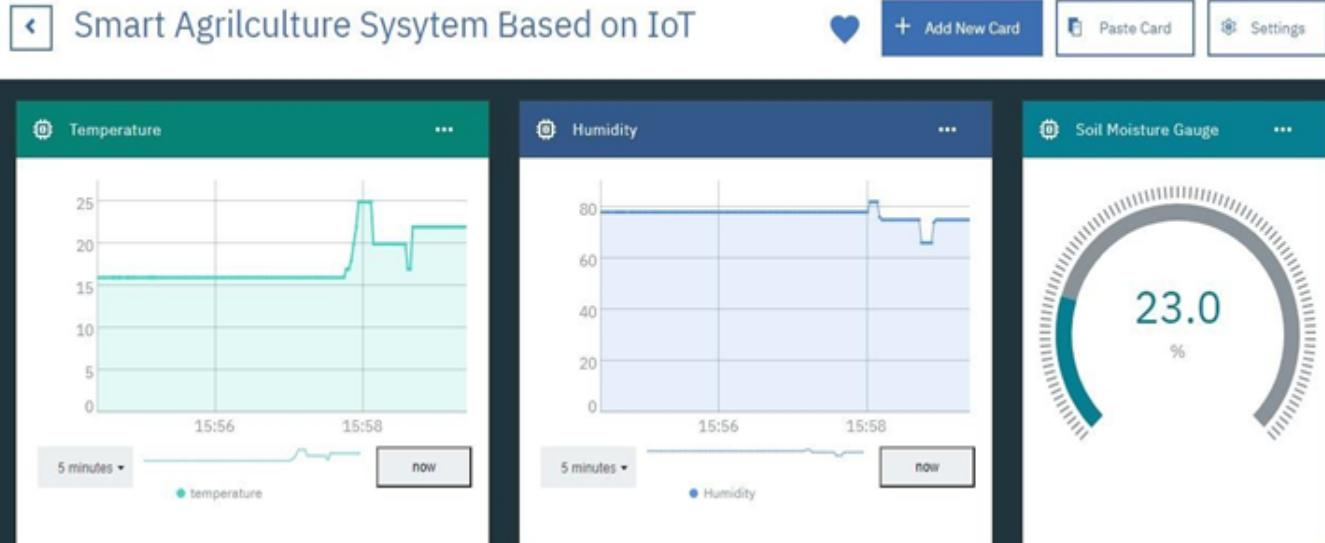
&m9FkAtga+v7sEF?f

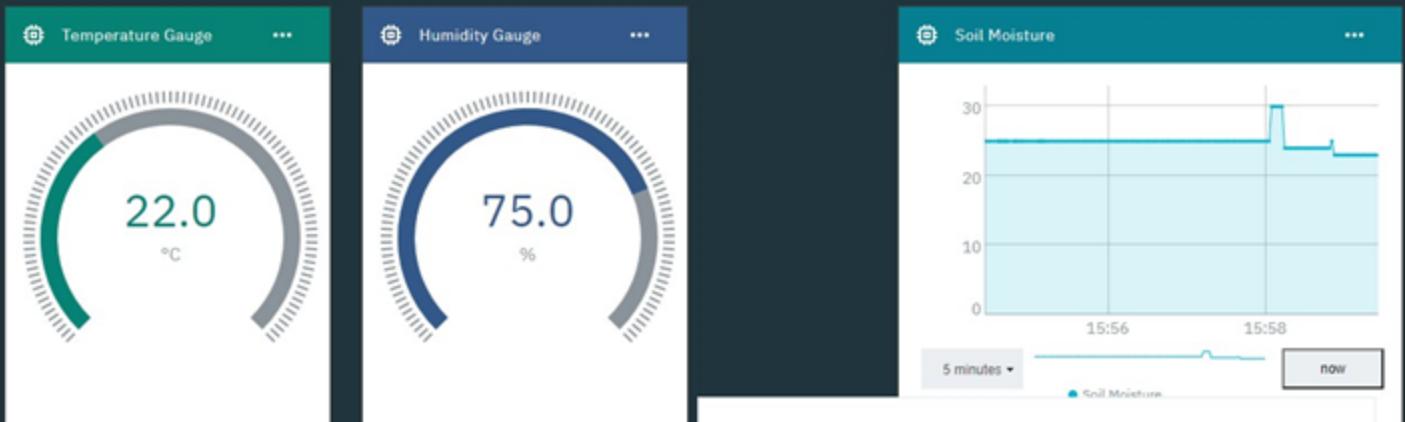
The screenshot shows the 'Recent Events' tab for a device. On the left is a sidebar with icons for Identity, Device Information, Recent Events, State, and Logs. The main area has tabs for Identity, Device Information, Recent Events (which is active), State, and Logs. A message says 'The recent events listed show the live stream of data that is coming and going from this device.' Below is a table of events:

Event	Value	Format	Last Received
iotsensor	{"d": [{"name": "555777", "temperature": 16, "humidity": 80}], "t": 15:57:55}	json	a few seconds ago
iotsensor	{"d": [{"name": "555777", "temperature": 16, "humidity": 80}], "t": 15:57:55}	json	a few seconds ago
iotsensor	{"d": [{"name": "555777", "temperature": 16, "humidity": 80}], "t": 15:57:55}	json	a few seconds ago
iotsensor	{"d": [{"name": "555777", "temperature": 16, "humidity": 80}], "t": 15:57:55}	json	a few seconds ago
iotsensor	{"d": [{"name": "555777", "temperature": 16, "humidity": 80}], "t": 15:57:55}	json	a few seconds ago

At the bottom right, it says '0 Simulations running'.

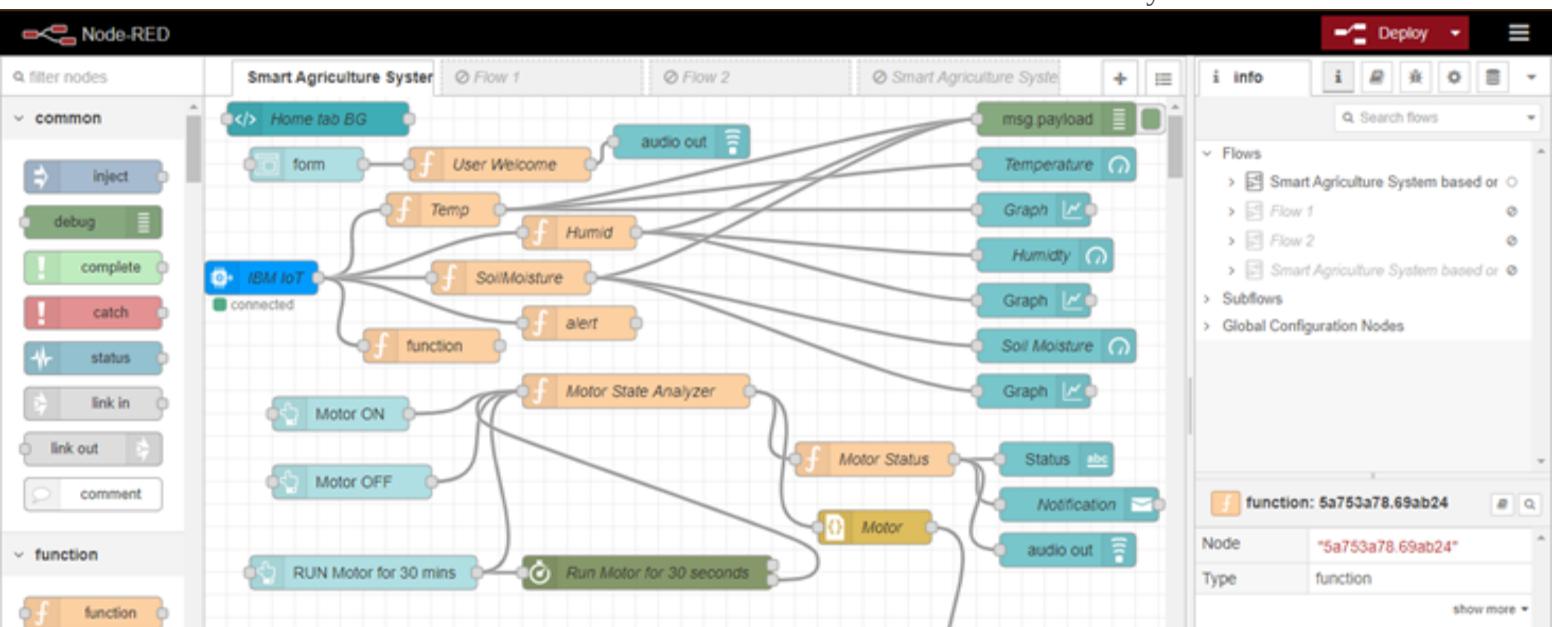
IBM Watson IoT Platform

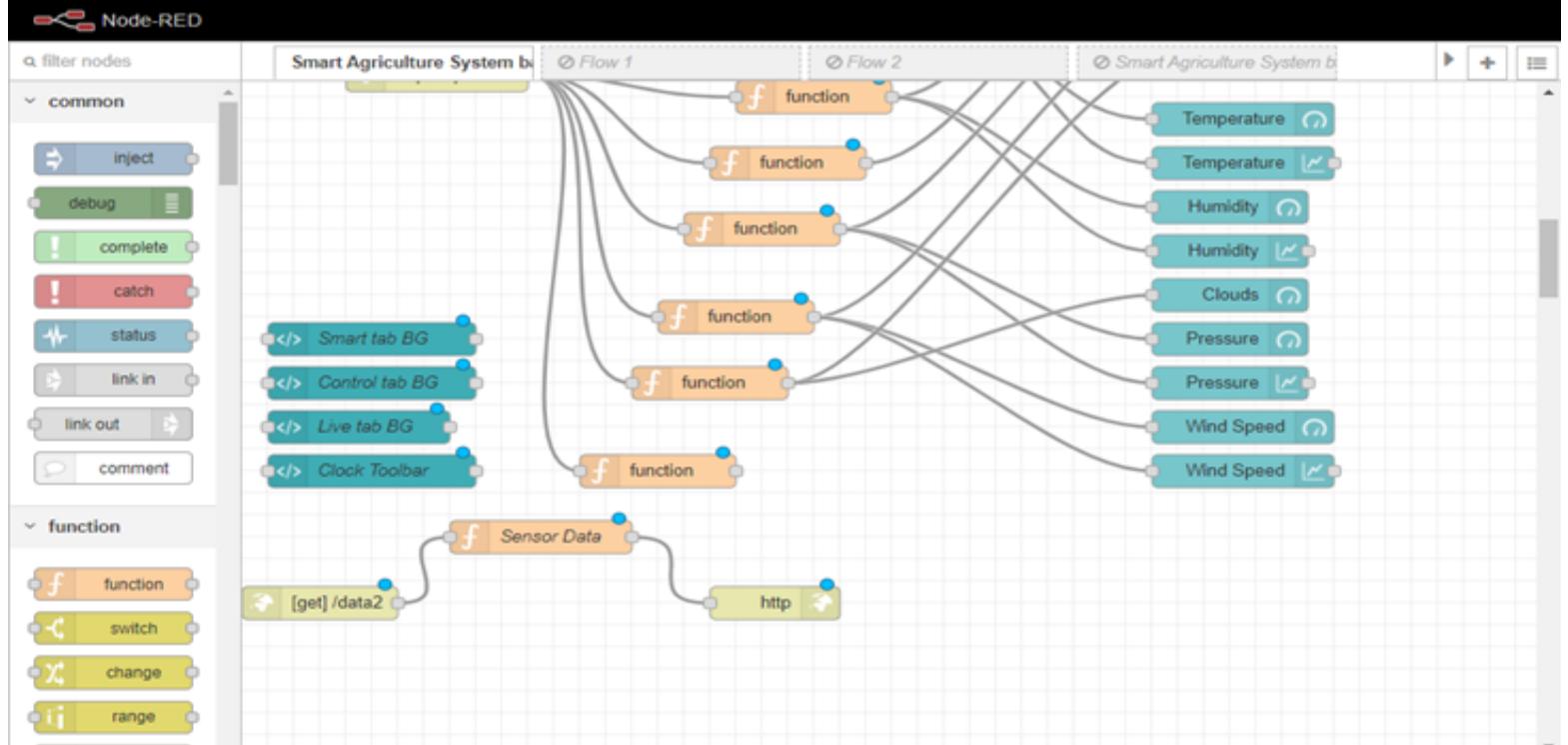
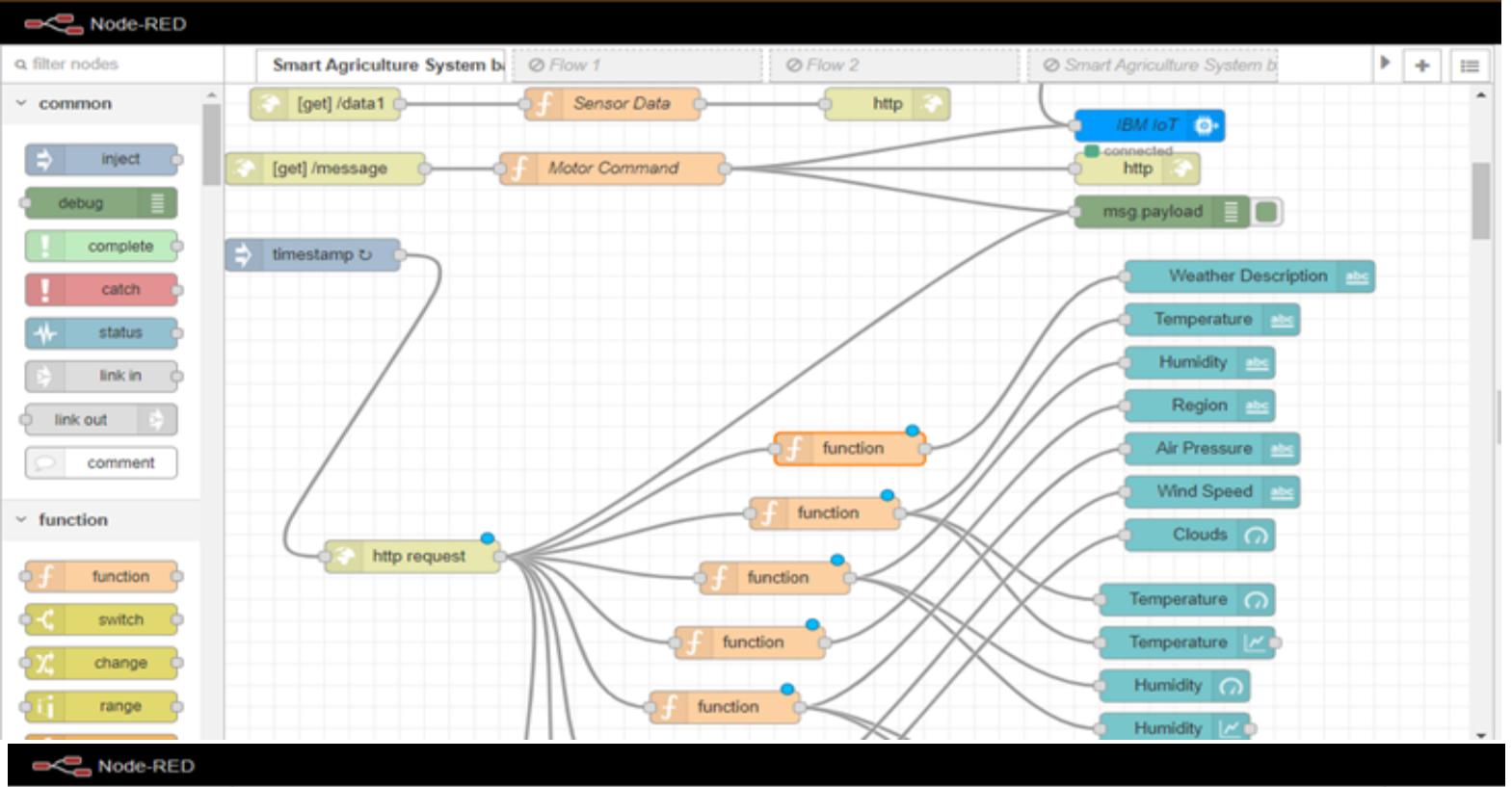




Configuration of Node-Red to collect IBM cloud data

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

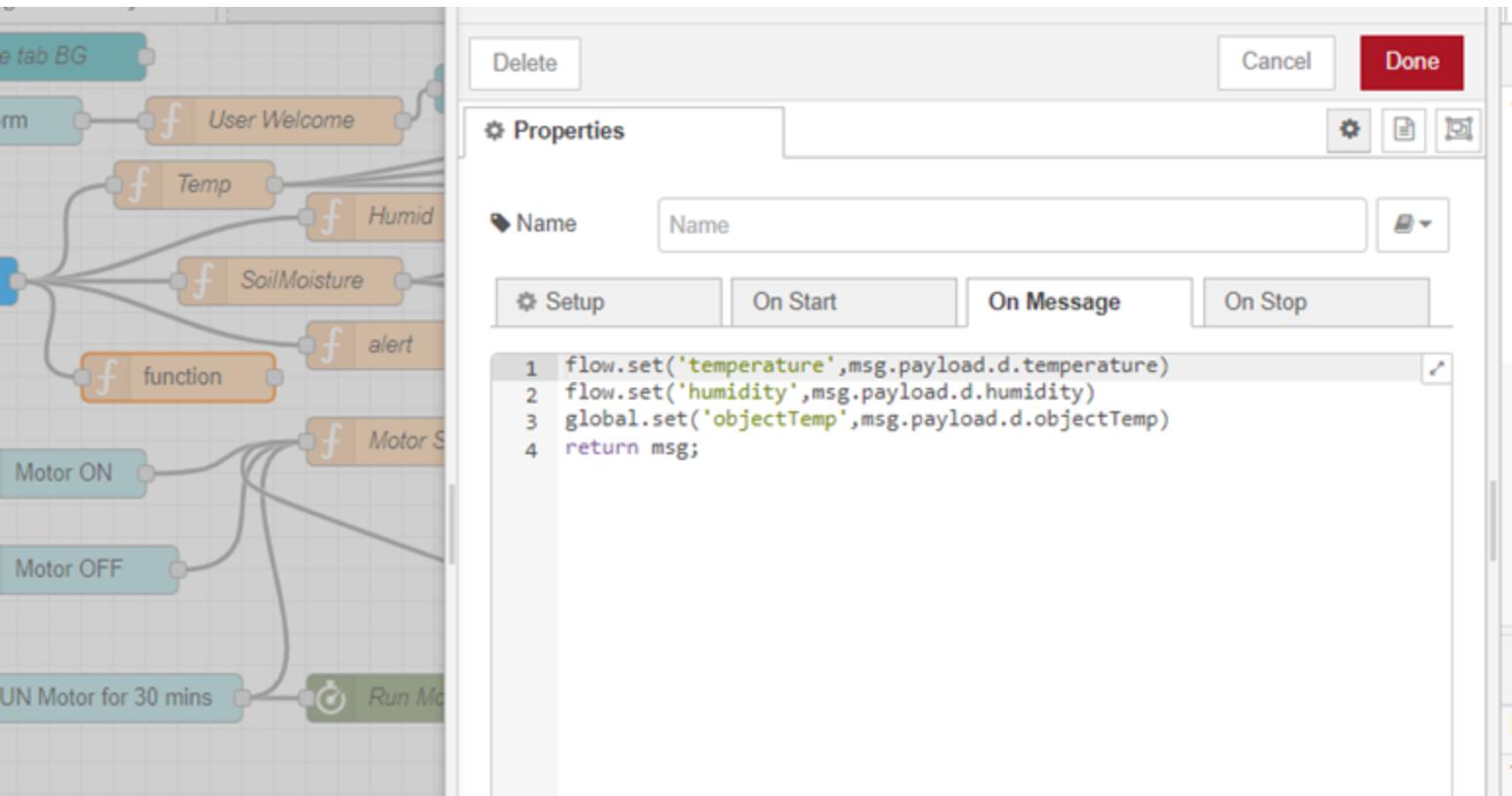
Display the data using debug node for verification

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

```
msg.payload=msg.payload.d.temperature return  
msg;
```

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

connected in following manner to get each reading separately.



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

Configuration of Node-Red to collect data from OpenWeather

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

HTTP request node is configured with URL we saved before in section 4.4 The data we receive from OpenWeather after request is in below JSON format:

```
{"coord":{"lon":76.65,"lat":12.31},"weather":[{"id":803,"main":"Clouds","description":"broken clouds","icon":"04d"}],"base":"stations","main":{"temp":27,"feels_like":26.75,"temp_min":27,"temp_max":27,"pressure":1009,"humidity":69},"visibility":8000,"wind":{"speed":6.2,"deg":270},"clouds":{"all":75}, "dt":1592387772,"sys":{"type":1,"id":9212,"country":"IN","sunrise":1592353739,"sunset":1592399985}, "timezon":19800,"id":1262321,"name":"Chennai","cod":200}
```

```
"1592387772,"sys":{"type":1,"id":9212,"country":"IN","sunrise":1592353739,"sunset":1592399985}, "timezon":19800,"id":1262321,"name":"Chennai","cod":200}
```

The above image has the program flow for receiving data from OpenWeather

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

The image shows two separate configuration dialog boxes for Node-RED nodes.

Edit http request node

- Properties:** Method: GET, URL: <http://api.openweathermap.org/data/2.5/weather?>
- Options:** Append msg.payload as query string parameters, Enable secure (SSL/TLS) connection, Use authentication, Enable connection keep-alive, Use proxy.
- Return:** a parsed JSON object
- Name:** Name
- Tip:** If the JSON parse fails the fetched string is returned as-is.

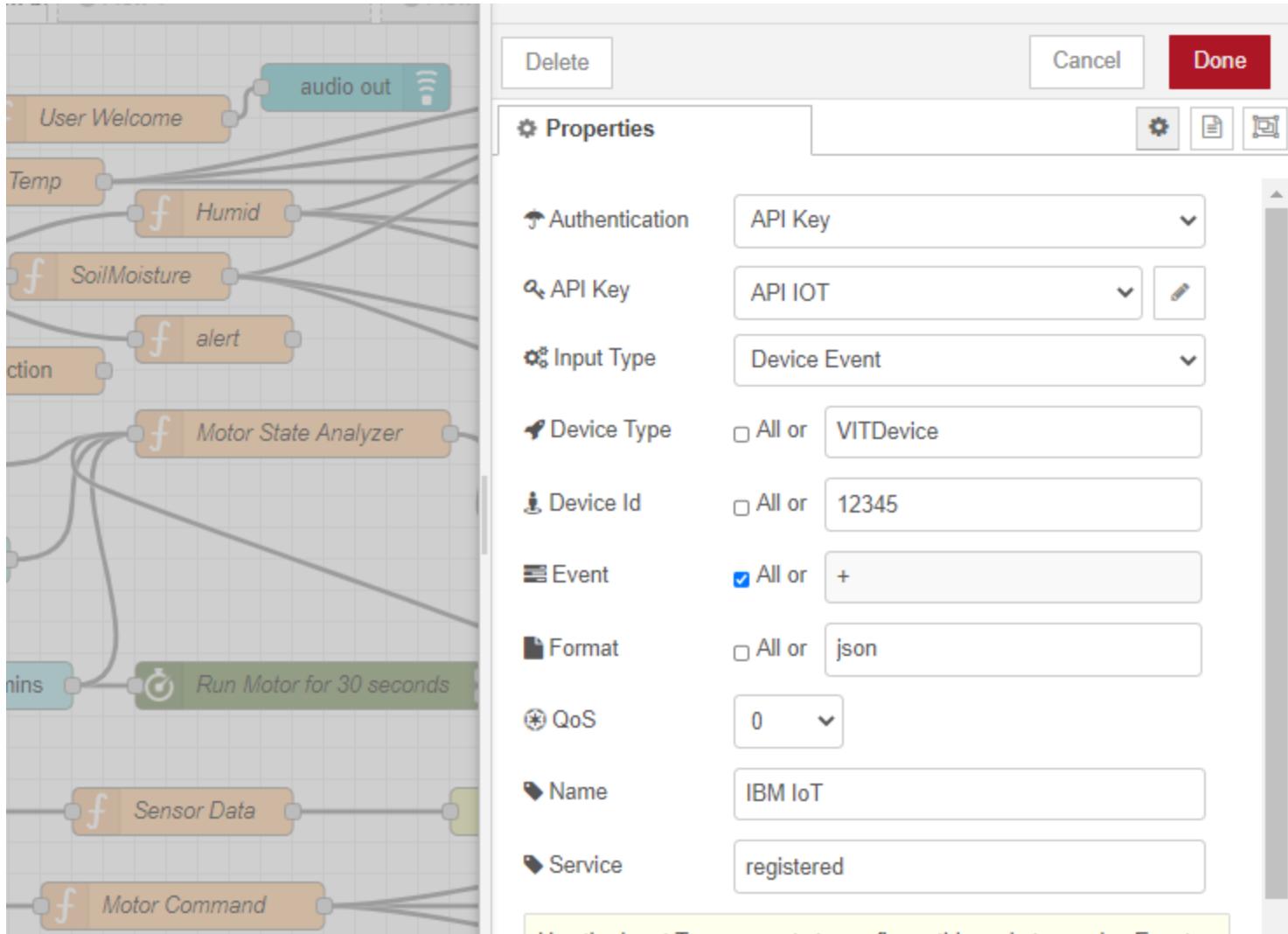
Edit change node

- Properties:** Name: Weather Description
- Rules:** Set msg.payload to msg.payload.weather.0.description

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

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 credentials of our Watson device.



Here we add three buttons in UI which each sends a number 0,1 and 2.

1. -> for motor off
2. -> for motor on
3. -> for running motor continuously 30 minutes

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

The Java script code for the analyser is:

```

if(msg.payload==1)
    msg.payload={"command":"ON"}; else
if(msg.payload==0)
    msg.payload={"command":"OFF"}; else
    msg.payload={"command":"runfor30minutes"};return
msg;

```

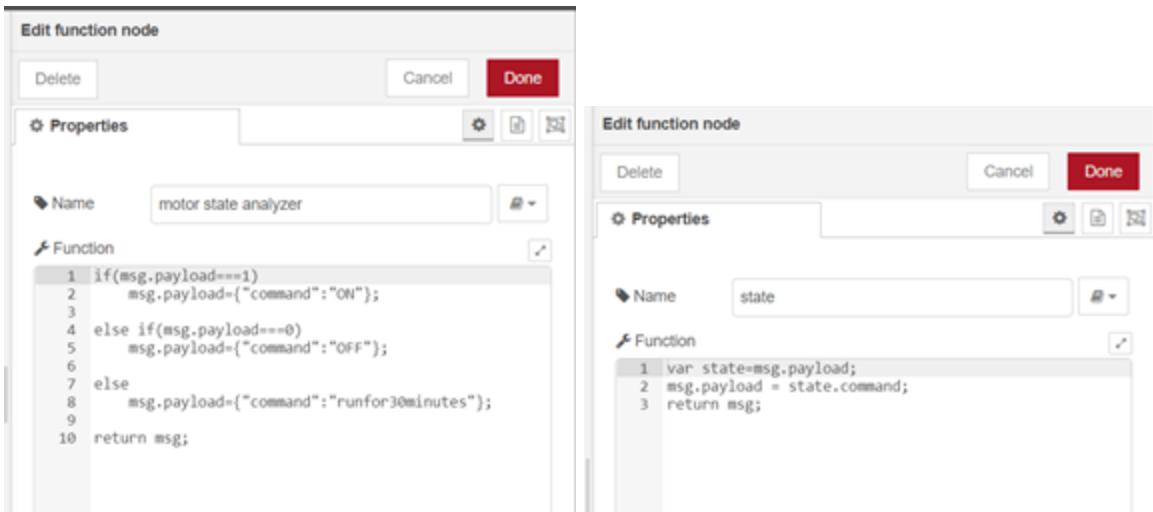
Then we use another function node to parse the data and get the command and represent it visually with text node.

The Java script code for that function node is:

```

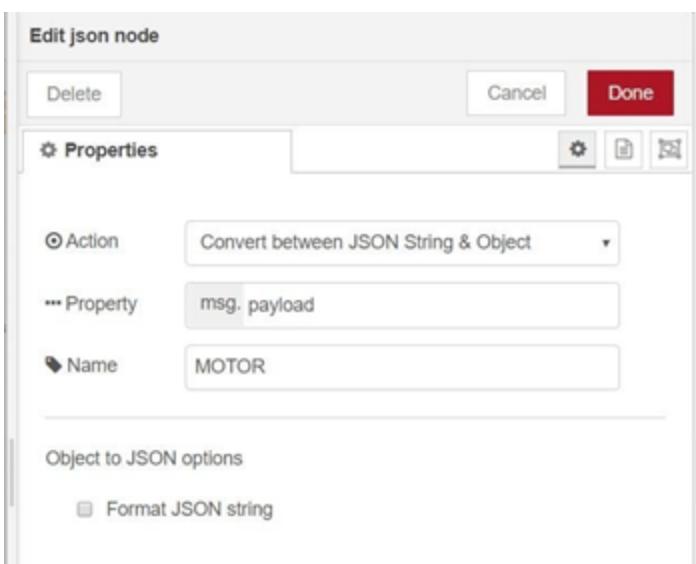
var
state=msg.payload; msg.payload =
state.command;
return msg;

```



The above images show the java script codes of analyser 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

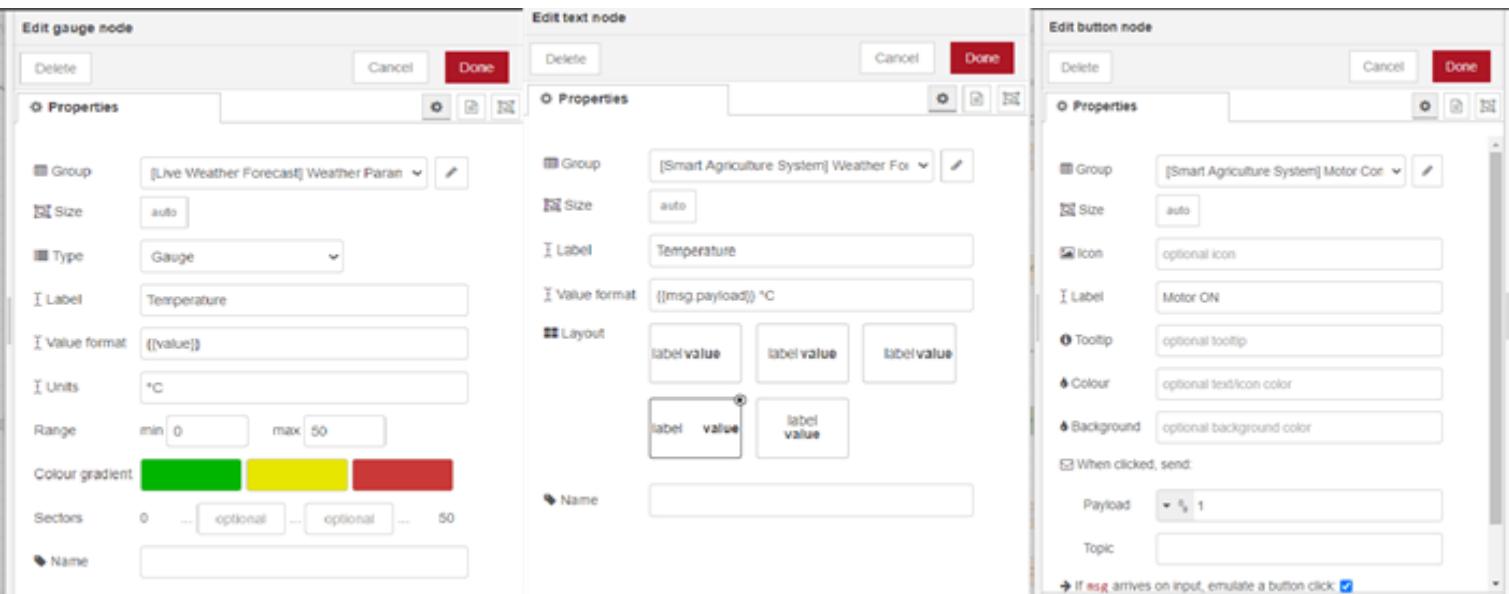
This is the program flow for sending commands to IBM cloud.

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.



Adding Background image to the UI

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

We add URL of image that need to be displayed

Edit template node

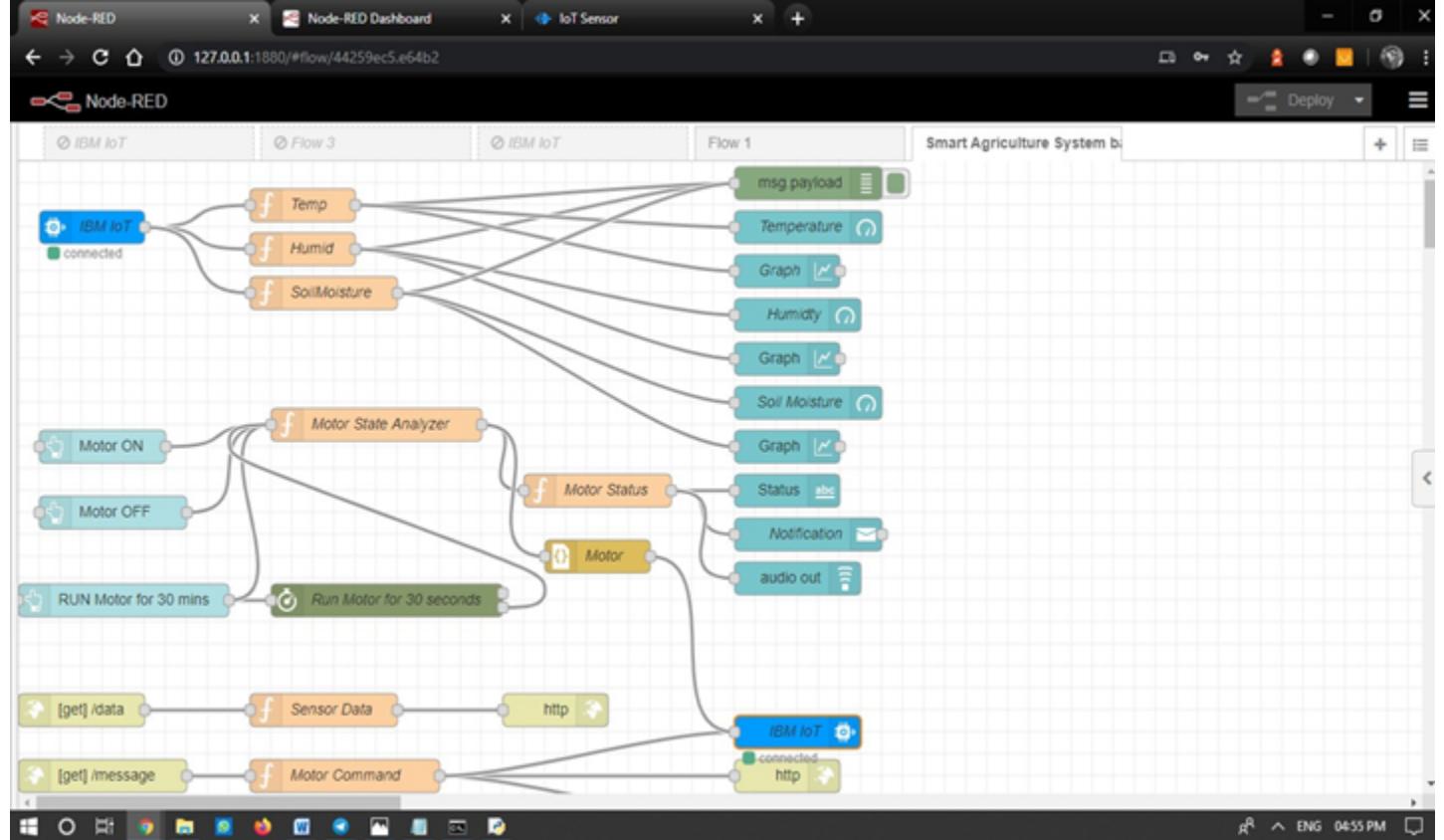
[Delete](#) [Cancel](#) [Done](#)

Properties

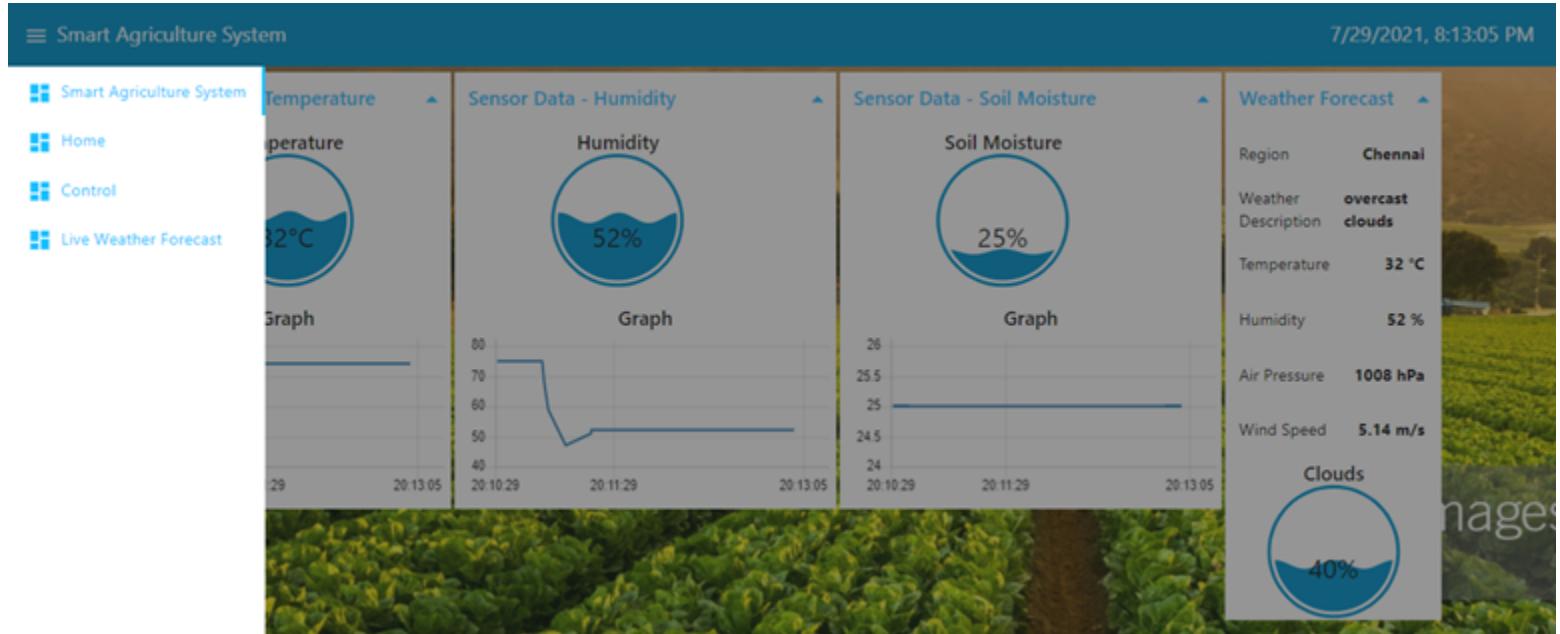
Template type	Widget in group
Group	[Home] WELCOME TO SMART FARM
Size	auto
Name	Home tab BG
Template	
<pre>1 <style> 2 body{ 3 background-image: url("https://media.gettyimages.com/photos/what-is-smart-farming-and-how-is-it-changing-agriculture-picture-id117900001?s=2048x2048"); 4 background-size: 150% 100%; 5 } 6 </style></pre>	
<input checked="" type="checkbox"/> Pass through messages from input. <input checked="" type="checkbox"/> Add output messages to stored state. <input checked="" type="checkbox"/> Reload last value on refresh.	

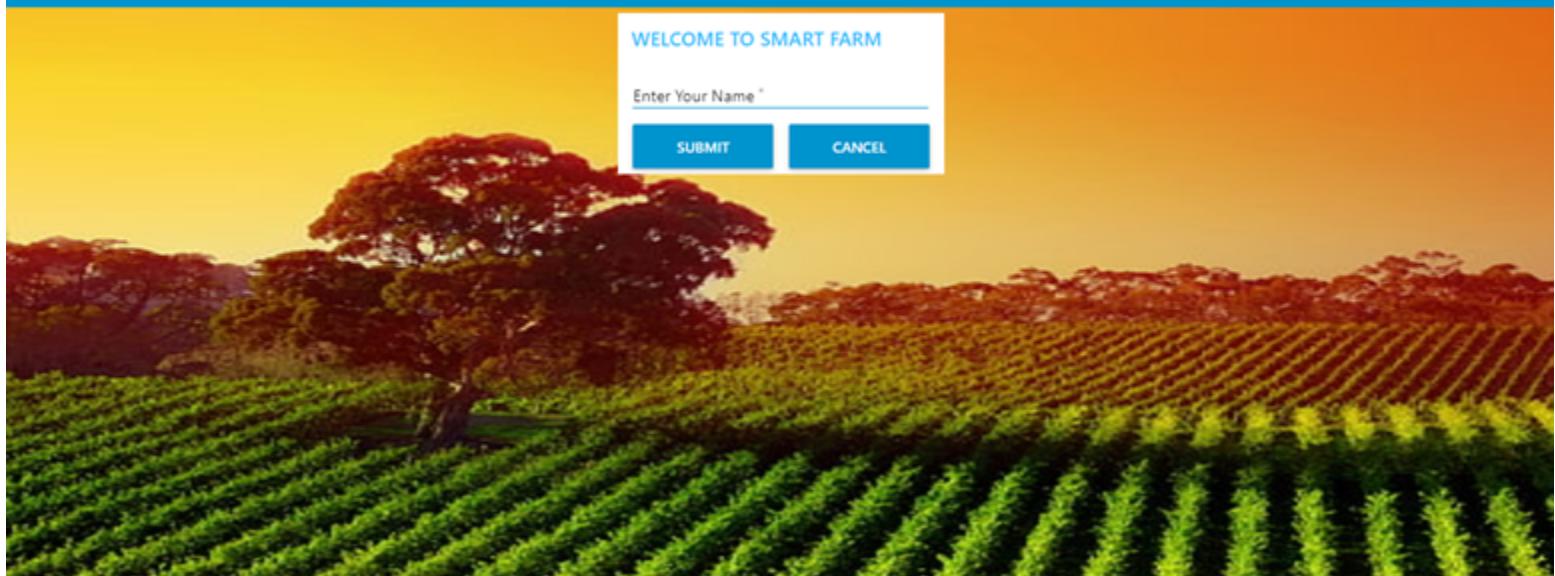
Configuration is shown in the above image

Complete Program Flow



Side View Tab





a. Receiving commands from IBM cloud using Python program

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

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

#Provide your IBM Watson Device Credentials
organization = "" #replace the
ORG ID deviceType = "MOTOR"#replace the Device type
wi

deviceId = "VITDevice"#replace Device ID
authMethod = "token"
authToken = "" #Replace the auth token
```

```
def myCommandCallback(cmd): # function for Callback print("Command
    received: %s" % cmd.data)
    if cmd.data['command']=='Motor turned ON':
        print("MOTOR ON IS RECEIVED")

    elif cmd.data['command']=='Motor turned OFF':
        print("MOTOR OFF IS RECEIVED")

    elif cmd.data['command']=='Running Motor for 30 minutes':
        print("MOTOR ON FOR 30 MINUTES 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()

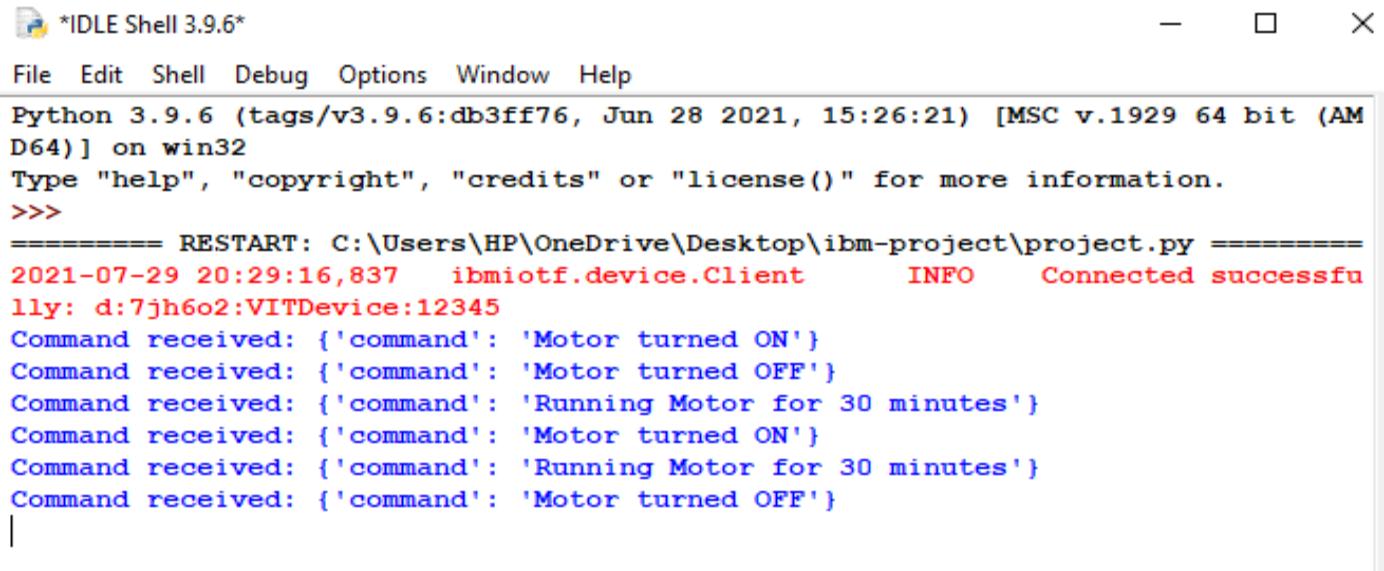
while True:

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

Device Commands are received by the local host through IBM Watson Cloud Platform

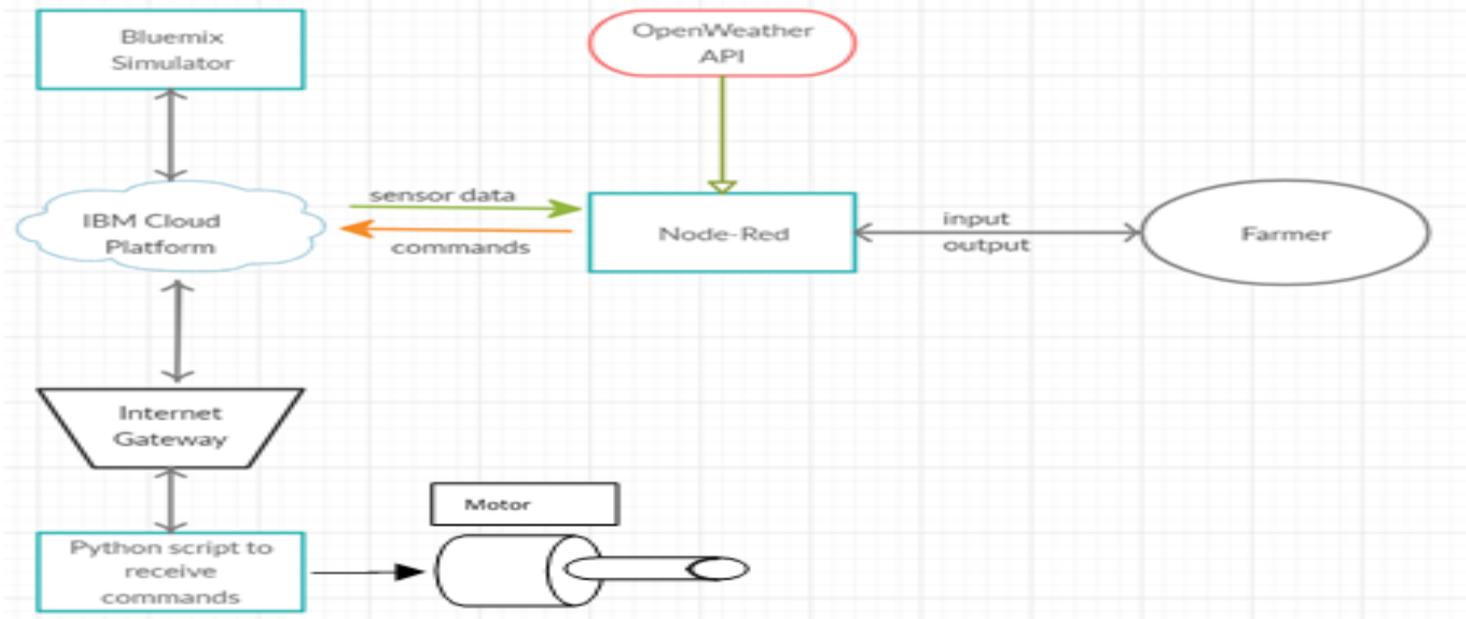


The screenshot shows the Python 3.9.6 command-line interface in IDLE. The window title is "IDLE Shell 3.9.6*". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The command-line area displays the following output:

```

Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\HP\OneDrive\Desktop\ibm-project\project.py =====
2021-07-29 20:29:16,837    ibmiotf.device.Client      INFO    Connected successfu
lly: d:7jh6o2:VITDevice:12345
Command received: {'command': 'Motor turned ON'}
Command received: {'command': 'Motor turned OFF'}
Command received: {'command': 'Running Motor for 30 minutes'}
Command received: {'command': 'Motor turned ON'}
Command received: {'command': 'Running Motor for 30 minutes'}
Command received: {'command': 'Motor turned OFF'}
|
```

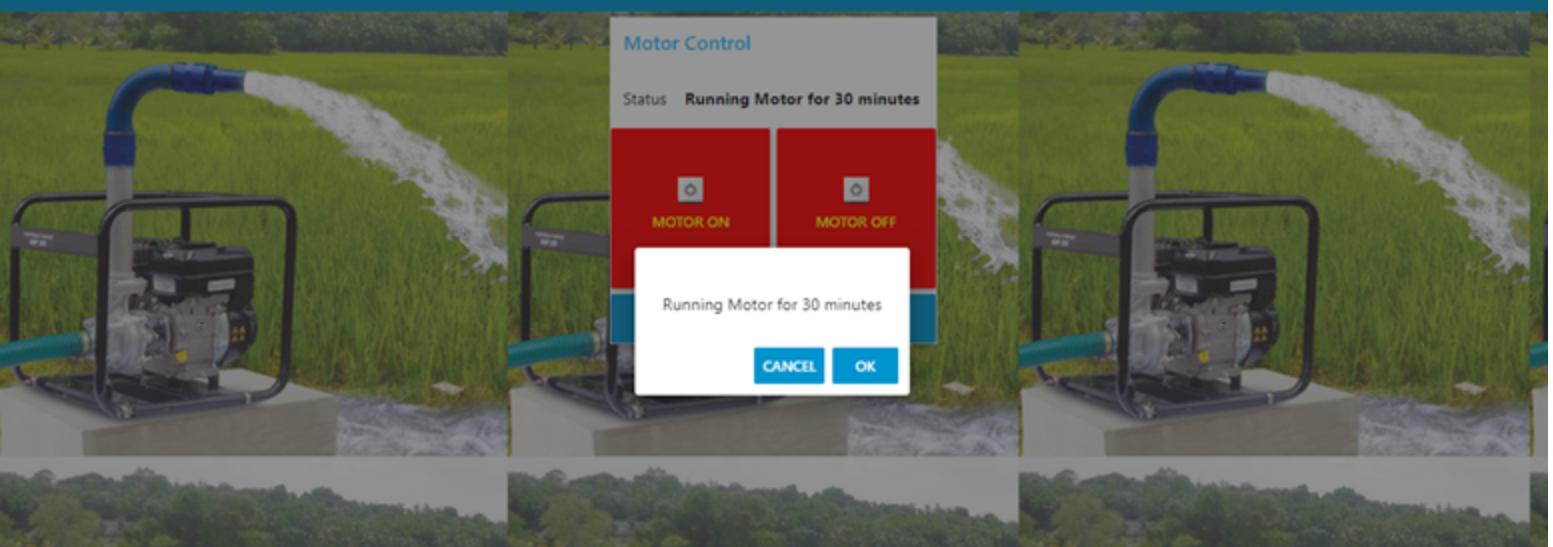
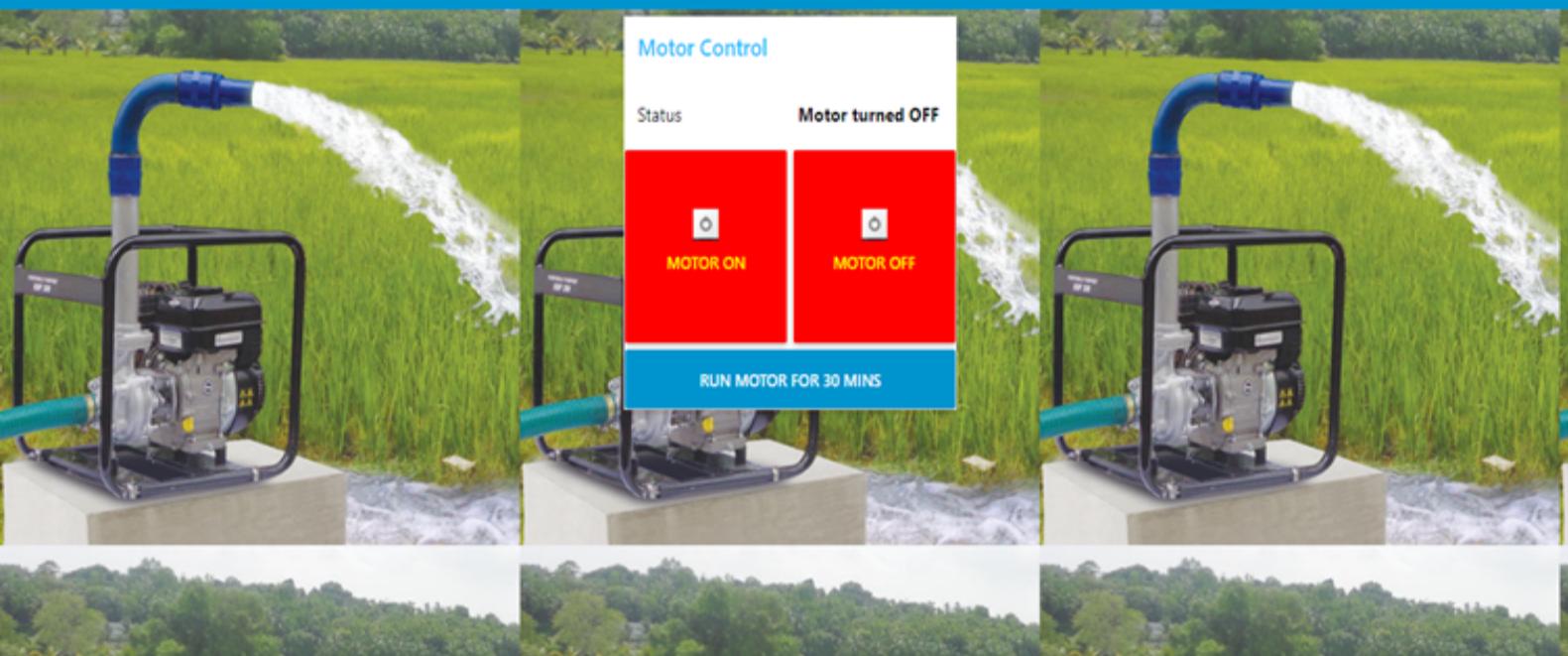
Flow Chart

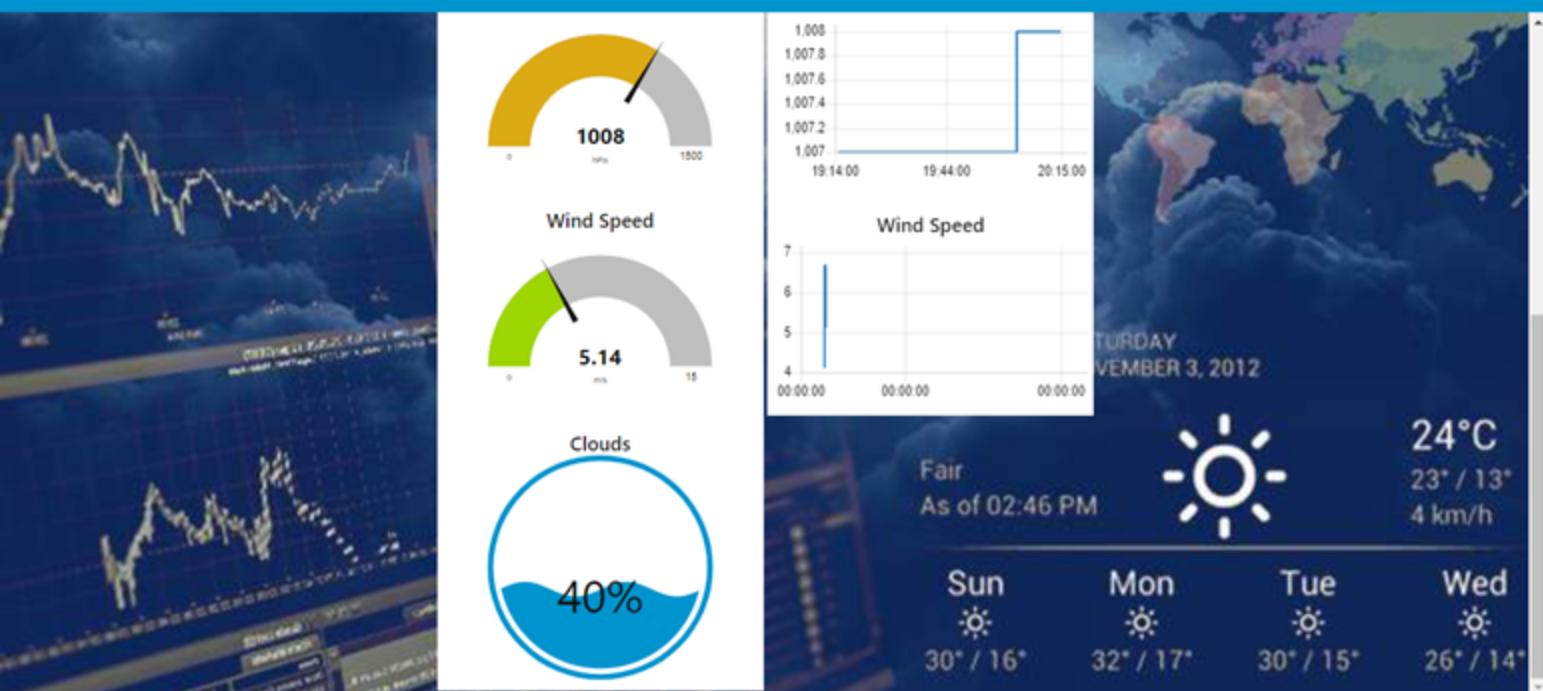
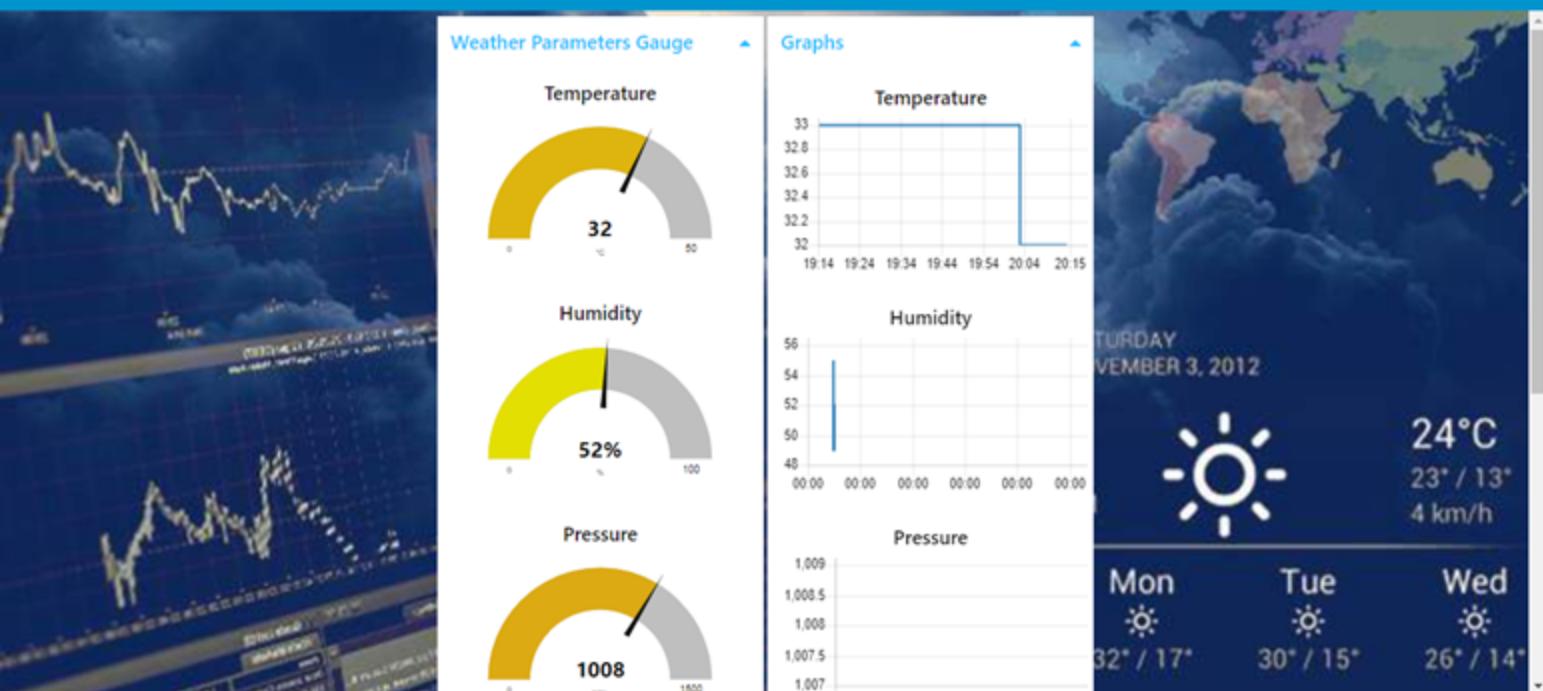


Observations & Results

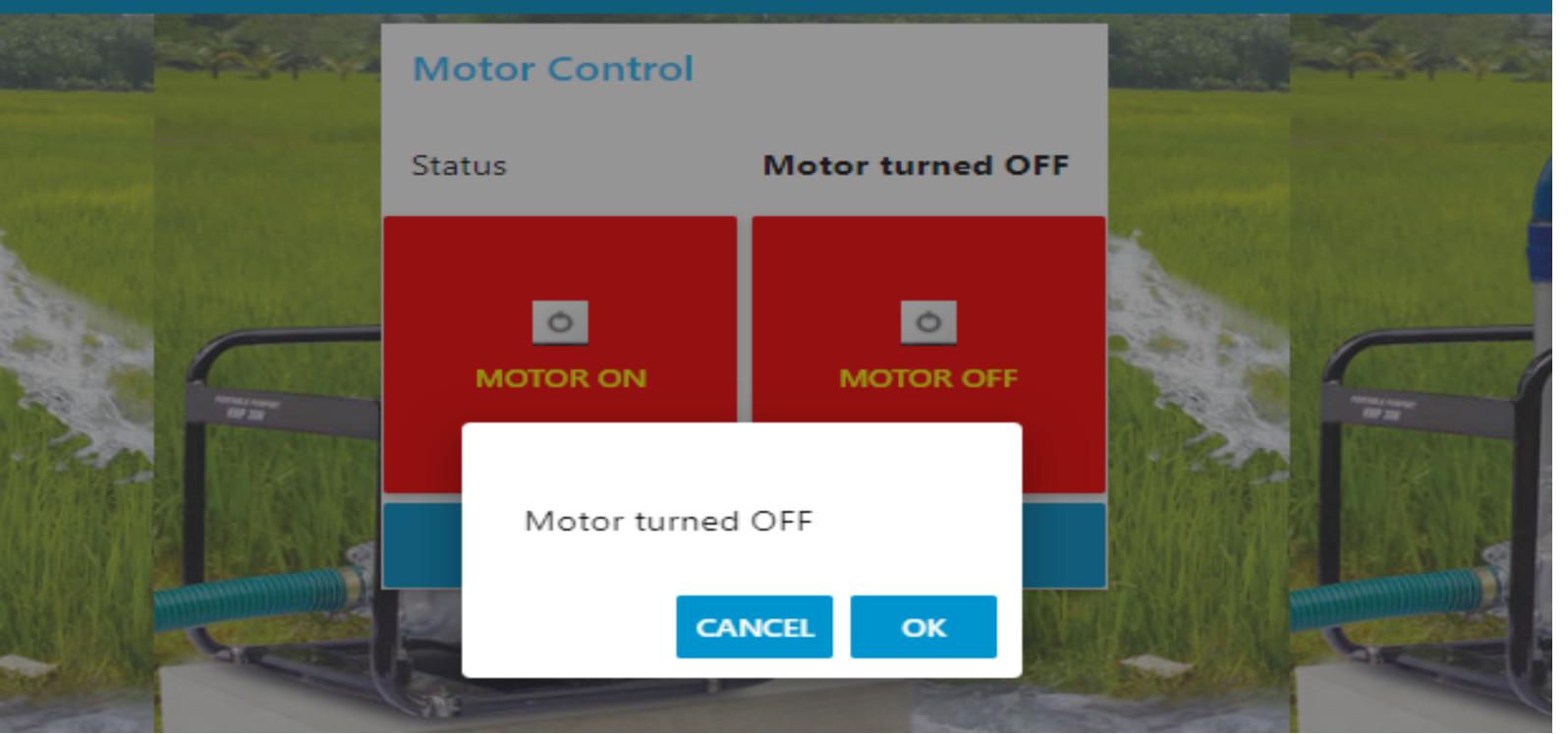
Simulator Data is received as shown in gauge and graph







Motor Commands



SMART_FARM APP

Screenshot of a mobile application development tool showing the 'Smart_Farm-App' interface.

The interface consists of three main sections:

- Palette:** On the left, a list of UI components including Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, Switch, TextBox, TimePicker, and WebViewer.
- Viewer:** In the center, a preview of the mobile phone screen titled "Screen1". The screen displays the "Smart_Farm-App" title, followed by "Sensor-Data" and "Weather forecast" sections, each containing four colored cards for Temperature, Humidity, Moisture, and Pressure. The "Region" section is shown at the bottom.
- Components:** On the right, a tree view of the application's components. The root node is "Screen1", which contains several HorizontalArrangement components, each with a Label and a TextBox. There are also sections for "Components" and "Properties".

```

when Screen1 .Initialize
do set reading .Url to "https://docs.google.com/spreadsheets/d/1iEKqWg7W..." 
call reading .Get

initialize global motor_commands to [ create empty list ]

when Web2 .GotText
url responseCode responseType responseContent
do call reading .Get

when Button2 .Click
do set Web2 .Url to [ join [ "https://docs.google.com/forms/d/14VVX0lYqWovq3IU..." ]
[ "?entry.1941799185=" ] ]
call Web2 .UriDecode
text Motoroff
call Web2 .Get

when Button1 .Click
do set Web2 .Url to [ join [ "https://docs.google.com/forms/d/14VVX0lYqWovq3IU..." ]
[ "?entry.1941799185=" ] ]
call Web2 .UriDecode
text Motoron
call Web2 .Get

when Clock2 .Timer
do set open_weather .Url to "https://node-red-reowq-2021-07-09.eu-gb.mybluemix.net"
call open_weather .Get

when Clock1 .Timer
do set Web1 .Url to "https://node-red-reowq-2021-07-09.eu-gb.mybluemix.net"
call Web1 .Get

when Web1 .GotText
url responseCode responseType responseContent
do set TextBox2 .Text to [ look up in pairs key [ "temperature" ]
pairs [ call Web1 .JsonTextDecode
jsonText get responseContent ] ]
notFound [ "not found" ]
set TextBox3 .Text to [ look up in pairs key [ "humidity" ]
pairs [ call Web1 .JsonTextDecode
jsonText get responseContent ] ]
notFound [ "not found" ]
set TextBox4 .Text to [ look up in pairs key [ "objectTemp" ]
pairs [ call Web1 .JsonTextDecode
jsonText get responseContent ] ]
notFound [ "not found" ]
set TextBox10 .Text to [ look up in pairs key [ "alert" ]
pairs [ call Web1 .JsonTextDecode
jsonText get responseContent ] ]
notFound [ "not found" ]

```



```
when reading .GetText
    url responseCode responseType responseContent
do set global motor_commands to list from csv table text get responseContent
    remove list item list get global motor_commands
        index 1
```

MOBILE APP

Smart_Farm-A

Sensor-Data

Temperature 32

Humidity 44

Moisture 25

Weather forecast

Temperature 32

Humidity 44

Pressure 1008

Region Vellore

Weather overcast clouds

Alert turn on motor

Motor ON **Motor OFF**

Advantages & Disadvantages

Advantages:

- Farms can be monitored and controlled remotely.
- Increase in convenience to farmers.
- Lowered labor and operating cost.
- Better standards of living.
- Water Conservation.

Disadvantages:

- Lack of internet/connectivity issues.
- Added cost of internet and internet gateway infrastructure.
- Farmers wanted to adapt the use of WebApp.

Applications

The common applications of the Smart Agriculture are:

- Sensor based systems for monitoring crops, soil, fields, livestock, storage

- facilities, or basically any important factor that influences the production.
- 2. Smart agriculture vehicles, drones, autonomous robots and actuators.
 - 3. Connected agriculture spaces such as smart greenhouse or hydroponics.
 - 4. Data analytics, visualization and management systems.
 - 5. Crop water management.
 - 6. Pest management and control works.
 - 7. Precision agriculture.
 - 8. Food production and safety, etc

Conclusion

IoT enabled agriculture has helped implement modern technological solutions to time tested knowledge. This has helped bridge the gap between production and quality and quantity yield. Data Ingested by obtaining and importing information from the multiple sensors for real time use or storage in a database ensures swift action and less damage to the crops. With seamless end to end intelligent operations and improved business process execution, produce gets processed faster and reaches supermarkets in fastest time possible. Thus the objective of the project to implement an IoT system in order to help farmers to control and monitor their farms has been implemented successfully.

Future scope

Future development will be focused more on increasing sensors on this system to fetch more live data regard to pest control, food production, etc also by integrating with the GPS to enhance the Agriculture IoT technology to full fill Agriculture Precision ready product. We can use it as a home automation controller. We can remotely operate or perform the jobs. Also combining with solar panels to conserve power. So the entire system is going to be eco-friendly.

Bibliography

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

Python code reference: <https://github.com/rachuriharish23/ibmsubscribe> IoT simulator : <https://watson-iot-sensor-simulator.mybluemix.net/>

OpenWeather : <https://openweathermap.org/>