

**INTERNSHIP PROJECT REPORT ON**

**INTERNET OF THINGS**

**Project Name:**

**Smart Irrigation System Based On IOT**

**Project ID:**

**SPS\_PRO\_249**

**Submitted By:**

**Bhavin Manani**

**Intern Id:**

**IlSPS\_INT\_4371**

**Table Of Content**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Content** | **Page No.** |
| 1. | **Introduction** | 3. |
|  | 1.1 Overview | 3. |
|  | 1.2 Purpose | 4. |
| 2. | **Literature Survey** | 5. |
|  | 2.1 Existing Problem | 5. |
|  | 2.2 Proposed System | 5. |
| 3. | **Theoretical Analysis** | 6. |
|  | 3.1 Block Diagram | 6. |
|  | 3.2 Hardware Designing | 7. |
|  | 3.3 Software Designing | 8. |
| 4. | **Experimental Investigations** | 9. |
| 5. | **Flowchart** | 10. |
| 6. | **Result** | 11. |
| 7. | **Advantages and Disadvantages** | 15. |
| 8. | **Applications** | 16. |
| 9. | **Conclusion** | 17. |
| 10. | **Future Scope** | 18. |
| 11. | **Bibliography** | 19. |
| 12. | **Appendix** | 20. |
|  | 12.1 Source Code | 21. |
|  | 12.2 UI Output Screenshot | 25. |

**CHAPTER 1**

**Introduction**

**1*.*1 Overview**

Agriculture is the major source of income for the largest population in India and is major contributor to Indian economy. However, technological involvement and its usability have to be grown still and cultivated for agro sector in India. Although few initiatives have also been taken by the Indian Government for providing online and mobile messaging services to farmers related to agricultural queries and agro vendor’s information to farmers. Based on the survey it is observed that agriculture contributes 27% to GDP, and Provides employment to 70% of Indian population.

IoT is changing the agriculture domain and empowering farmers to fight with the huge difficulties they face. The agriculture must overcome expanding water deficiencies, restricted availability of lands, while meeting the expanding consumption needs of a world population. New innovative IoT applications are addressing these issues and increasing the quality, quantity, sustainability and cost effectiveness of agricultural production.

Agriculture is the backbone of Indian Economy. In today’s world, as we see rapid growth in global population, agriculture becomes more important to meet the needs of the human race. However, agriculture requires irrigation and with every year we have more water consumption than rainfall, it becomes critical for growers to find ways to conserve water while still achieving the highest yield. But in the present era, the farmers have been using irrigation technique through the manual control in which they irrigate the land at the regular interval.

According to statistics, agriculture uses 85% of available freshwater resources worldwide, and this percentage will continue to be dominant in water consumption because of population growth and increased food demand. There is an urgent need to create strategies based on science and technology for sustainable use of water, including technical, agronomic, managerial and institutional improvements. Agricultural irrigation based on Internet technology is based on crop water requirement rules. By using Internet technology and sensor network technology we can control water wastage and to maximize the scientific technologies in irrigation methods. Hence it can greatly improve the utilization of water and can increase water productivity

The Internet of Things (IoT) is a technology where in a mobile device can be used to monitor the function of a device. The Internet of Things (IoT) is concerned with interconnecting communicating objects that are installed at different locations that are possibly distant from each other. Internet of Things (IoT) is a type of network technology, which senses the information from different sensors and makes anything to join the Internet to exchange information.

**1.2 Purpose**

The proposed system has been designed to overcome the unnecessary water flow into the agricultural lands. Temperature, moisture and humidity readings are continuously monitored by using temperature, moisture and humidity sensor and send these values to the assigned IP address. Android application continuously collects the data from that assigned IP address. Once the soil moisture values are exceeded the particular limit then the relay, which is connected to the Arduino microcontroller controls the motor. The android application is a simple menu driven application, with 4 options. This includes motor status, moisture, temperature and humidity values. The motor status indicates the current status of the pump.

**CHAPTER 2**

**Literature Survey**

**2.1 Existing Problem**

* The existing System does not have any support towards controlling the system for longer distance from web application and for parsing the sensor values from microcontroller to web ui.
* There is no support for obtaining the weather data from online weather stations, so that the operator can sense the change in climate and can manually turn off motor when it will be raining, hence water hereby could be saved and preserved.

**2.2 Proposed Solution**

* The proposed system will be having IBM cloud platform for sending sensor data towards any distance client with the help of internet and could control the system from anywhere with the help of ZigBee module and IBM platform.
* The system will also have the weather forecast data readings for monitoring the climate change and to get climate data like wind speed, humidity, temperature, and various information regarding weather with the help of openweather api.
* The system will display all this reading on web application so that it can be manually controlled.

**CHAPTER 3**

**Theoretical Analysis**

**3.1 Block Diagram**

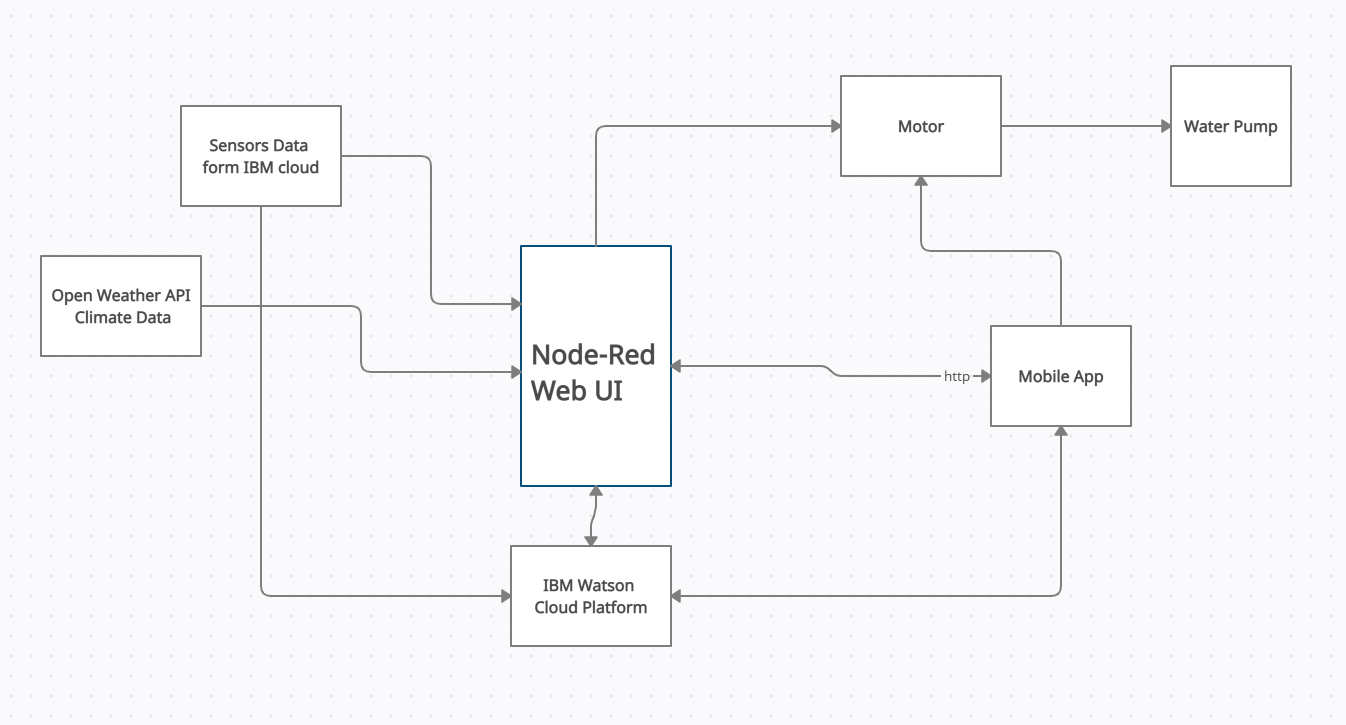
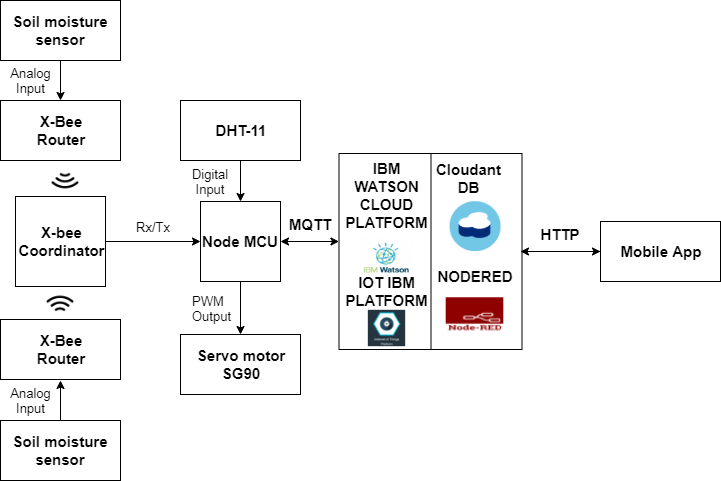


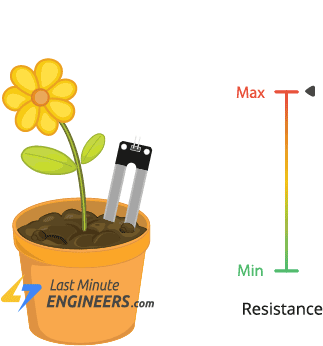
Figure 3.1 Flow Diagram

Figure 3.2 Project Diagram

**3.2** **Hardware Designing**

**Soil Moisture Sensor:**

The soil moisture sensor is one kind of sensor used to gauge the volumetric content of water within the soil. As the straight gravimetric dimension of soil moisture needs eliminating, drying, as well as sample weighting. These sensors measure the volumetric water content not directly with the help of some other rules of soil like dielectric constant, electrical resistance, otherwise interaction with neutrons, and replacement of the moisture content.

****

**Node MCU:**

NodeMCU is an open source platform, it’s hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip.The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. It can be used for a wide variety of IoT applications.

**Servo Motor:**

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

**3.3 Software Designing**

**Node-Red:**

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

**MIT App Inventor:**

MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

**IBM Cloud:**

IBM Cloud is online combined service like PaaS and IaaS, it is used at backend for sending real time values and storing the values at node-red cloudant database. For creating virtual devices IBM cloud is used.

**Fast2SMS:**

For sending text messages regarding the moisture content is too low than a specified value Fast2SMS provides a dev API service to send a quick message to user.

**OpenWeatherMap:**

For getting access to weather forecast and climate data openweathermap provides api for getting access of specific country weather forecast access.

**NOTE: AS WE DON'T HAVE SENSORS TO INTEGRATE USE RANDOM VALUES AS SENSOR DATA**

**Chapter 4**

**Experimental Investigations**

1.Start With Creating IBM Account

2.Create Cloud Foundry apps

* Node-red Web app

3.Create Cloud Foundry Services

* Node-red app services

4.Create Internet of Things platform

* For Create Virtual Devices

5.Create Node-red Flow diagram

* Create a flow diagram and connect the Virtual Devices

6.Develop the python code

* Develop python in Python IDLE
* Connect The code with Device using IoT Platform's Configuration like Device ID ,DeviceType,Organization,Authentication Method and Authentication Token.
* Send Data to Node-Red Flow Diagram

7.Create Web Application Using Node-Red Application Using Different Nodes

* Design UI to Display The Quantity Of The product in jar
* Display The gas sensor value as % of Toxic air
* Display The cylinder Weight In KG

8.Develop HTTP Request For Sending SMS alert in Fast2SMS Services.Copy that link into Node-red Application

9.Develop Mobile Application Using MIT App Inventor.

**Chapter 5**

**Flowchart**

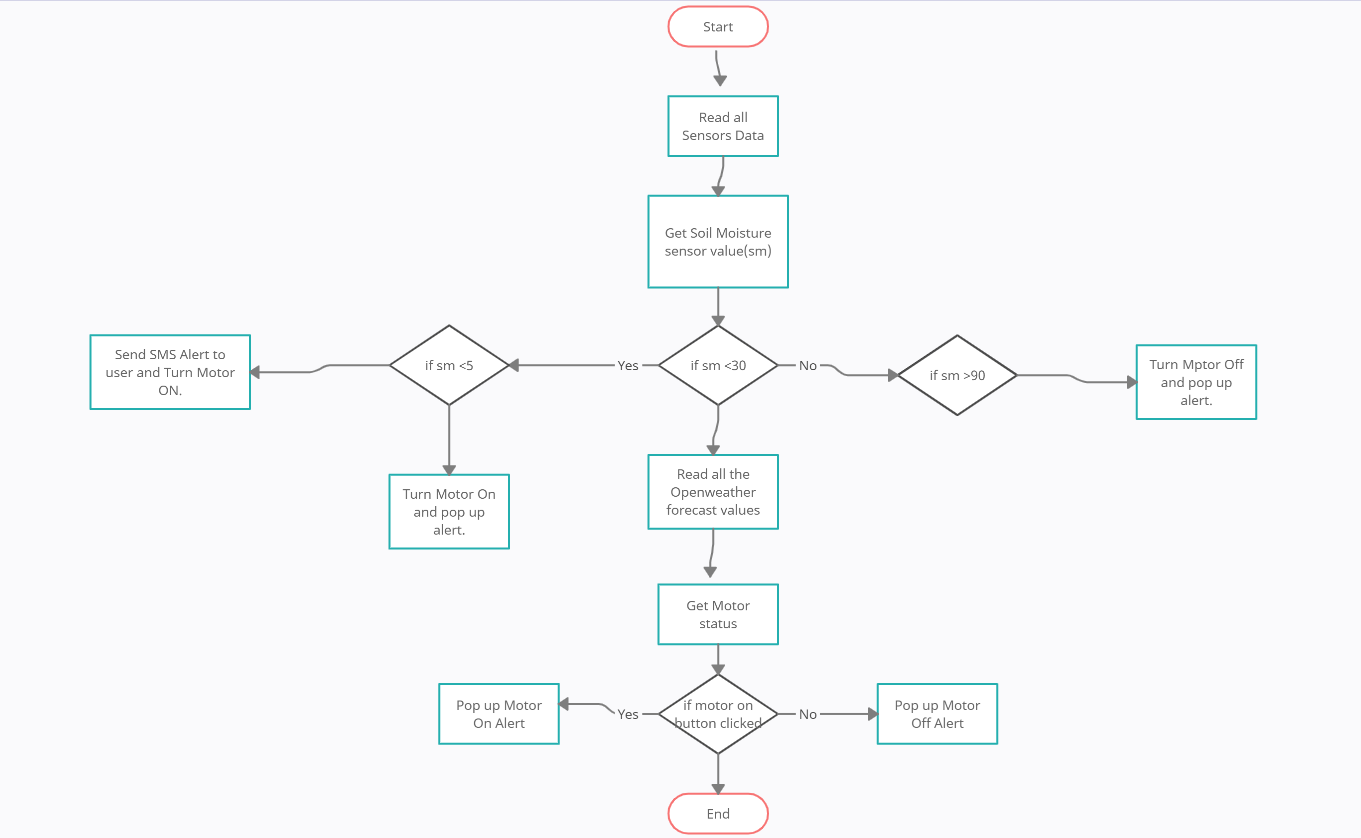
****

Figure 4.1 Flowchart

**Chapter 6**

**Result**

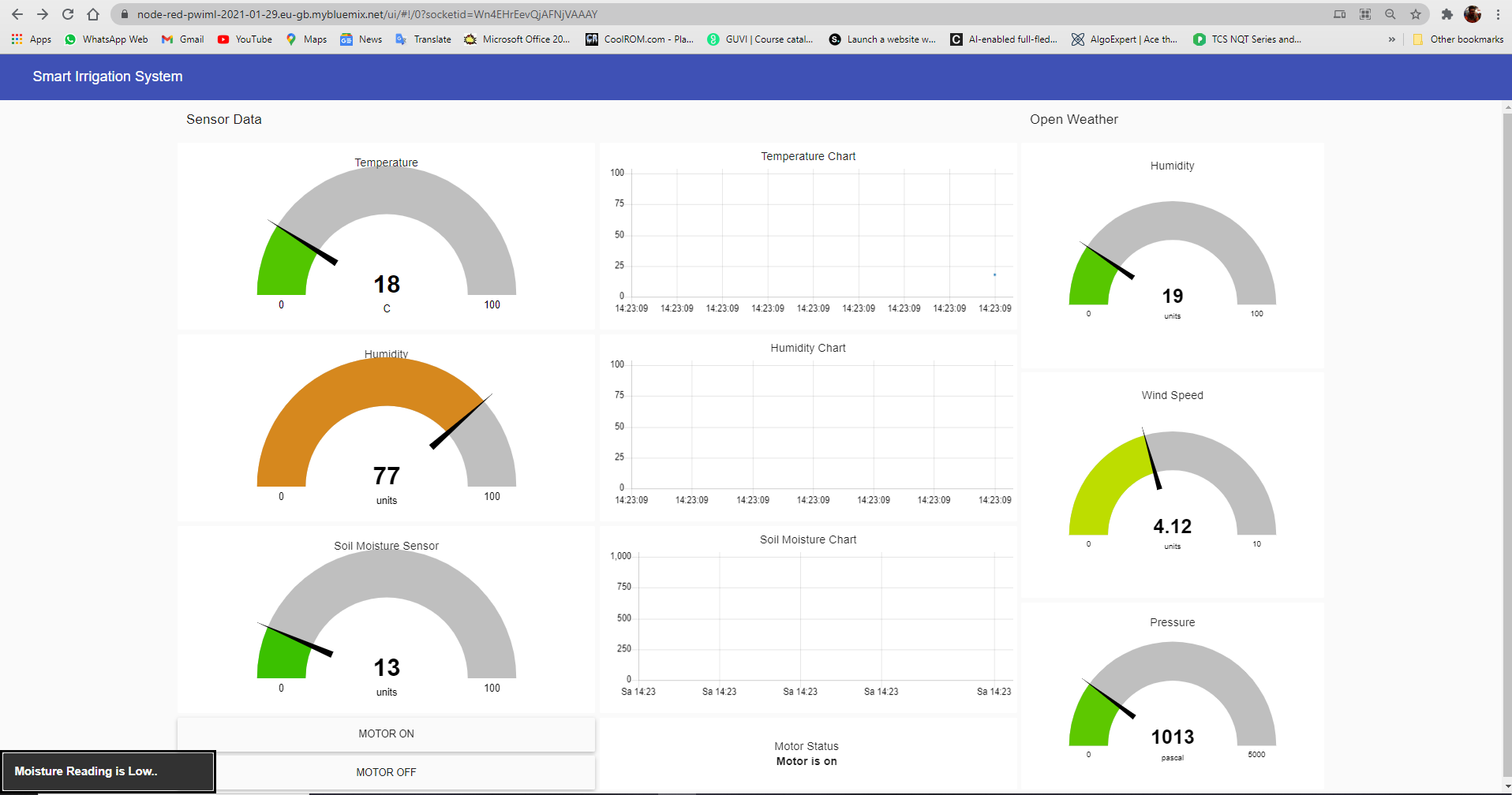
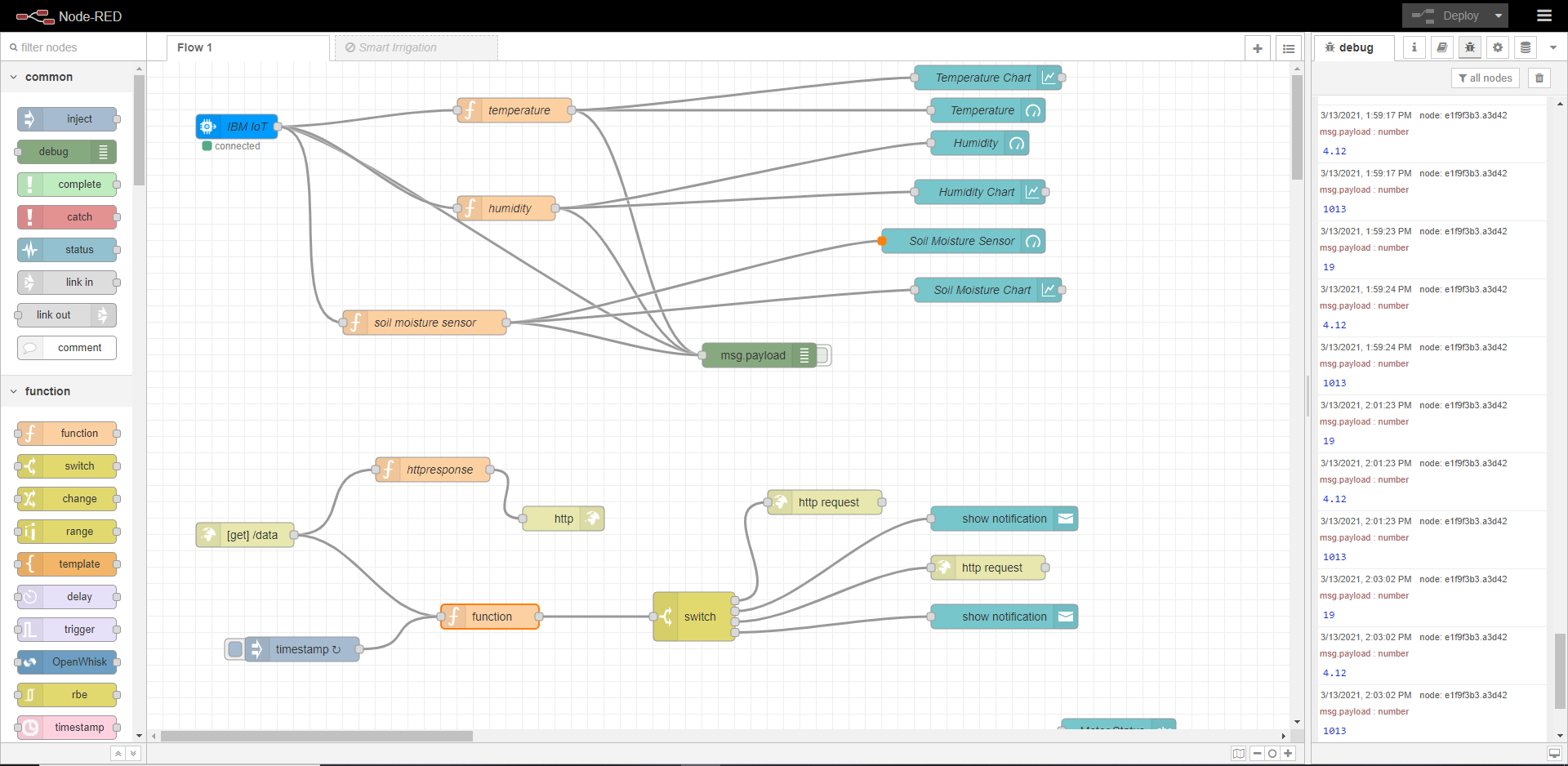


Figure 6.1 Node-red Web UI

Figure 6.2 Node-red Sensor Data Flow

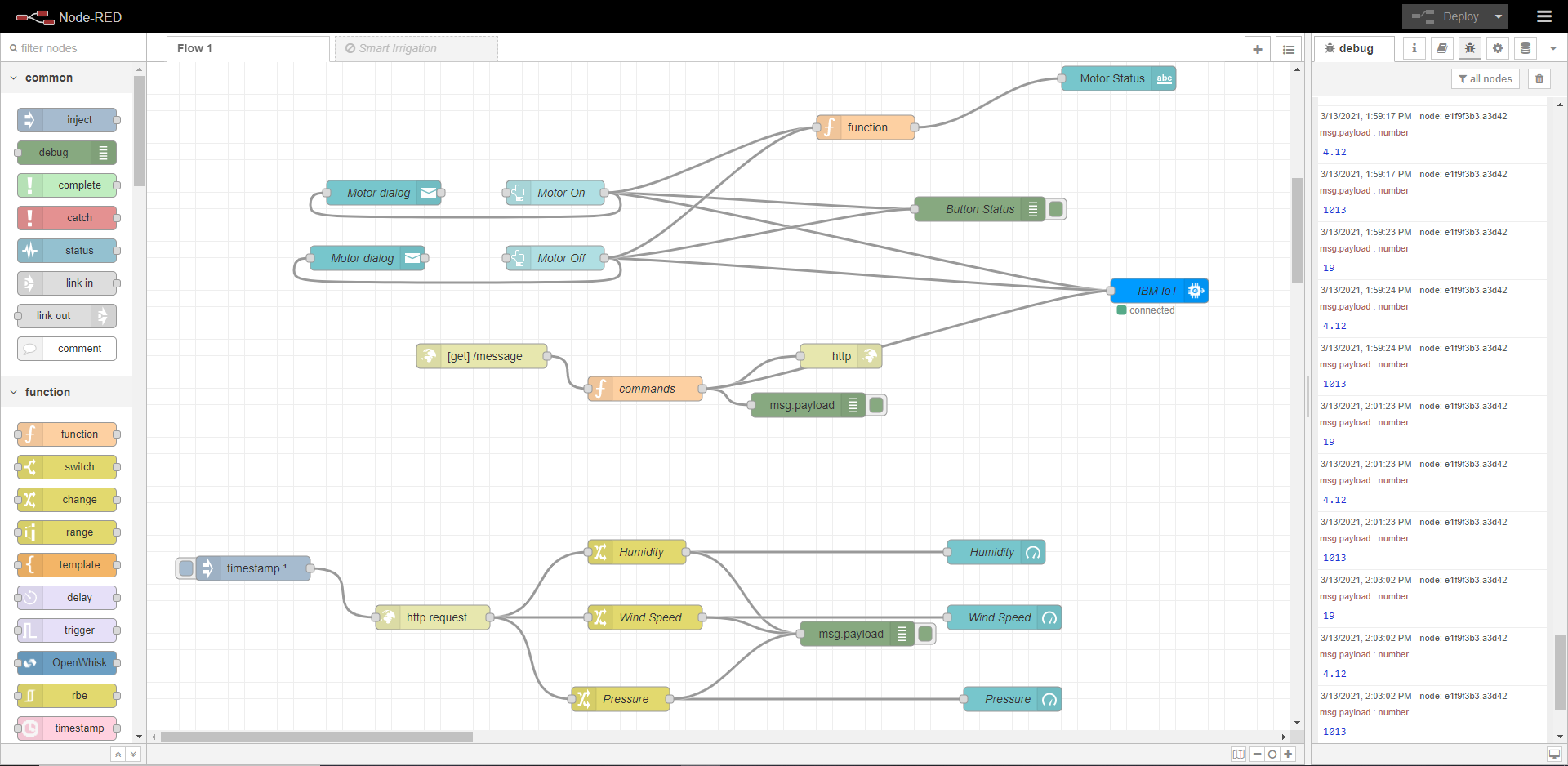


Figure 6.3 Node-red Weather Forecast

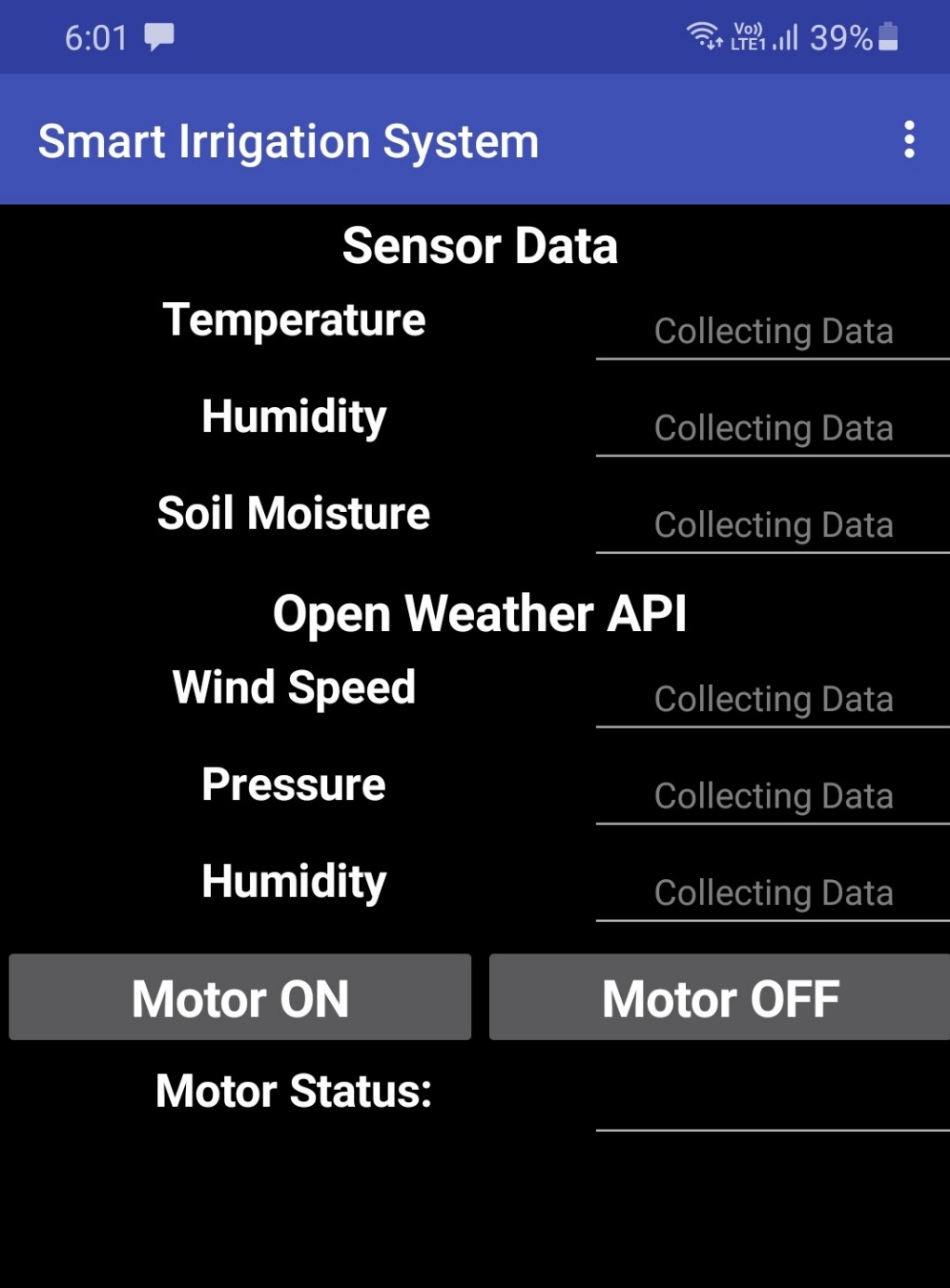


Figure 6.4 Mobile App

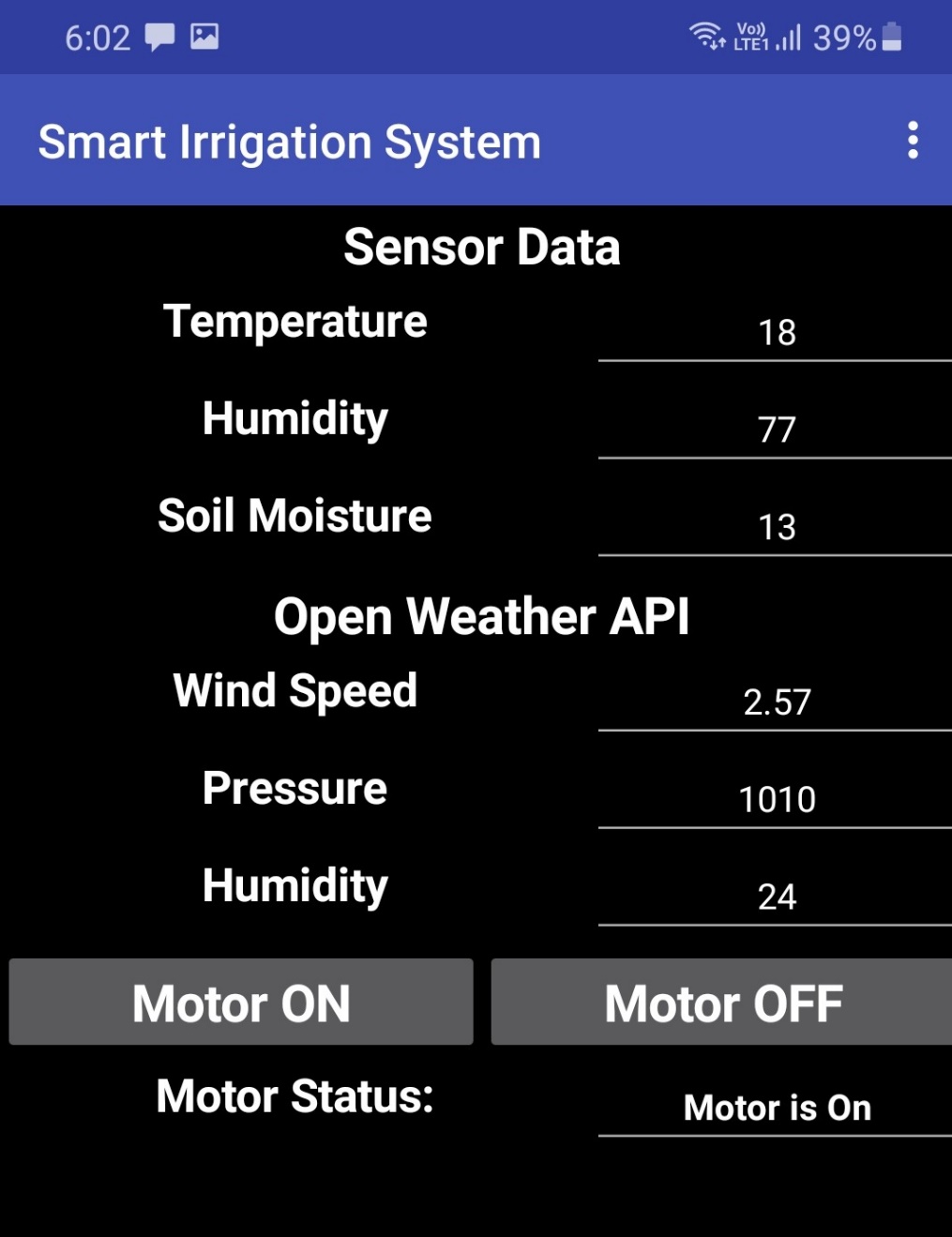


Figure 6.5 Mobile App Displaying data

**Chapter 7**

**Advantages and Disadvantages**

**Advantages:**

* The System is controlled automatically and also manually through monitoring the weather forecast, that is whether to turn motor on or off.
* The System is based on internet, thereby it can be control or monitor from anywhere and also user interface is so friendly that it can be managed by anyone.
* Most importantly system is automatic so if the soil moisture content will decrease the system will automatically turn motor on and the field will be supplied with water and if the moisture content will increase the motor will be turned off automatically.

**Disadvantages:**

* Since the real time sensor will be connected to the controller, the controller requires continuous supply of internet to transfer the data which will increase cost of internet.
* Also the whole field will be having multiple sensors so the cost will also be increases buying such amount of sensors and also modules to send data from each sensor to internet.

**Chapter 8**

**Applications**

**Applications:**

* This technique can be used by person owning a landyard.
* All the sensor values will be monitored on web application or mobile app and can be control automatically.
* This technique will reduce all the labour work.

**Chapter 9**

**Conclusion**

The various parameters like soil moisture, temperature, humidity were monitored using web application. The data from open weather map like wind speed, temperature, humidity, etc were also displayed in the web browser. The device was also able to turn motor on/off manually and also automatically. Also the system was able to send alert notification and quick sms to user.

**Chapter 10**

**Future Scope**

* In future the system will be connected with the actual sensor and will be displaying the actual values of sensor and monitor on web app.
* We can also add voice connectivity for turning motor on/off.
* In future we can add rain forecast so if there is already raining and if moisture content is low motor will not be turned on as its already raining field will get water and also water from motor will not be wasted.

**Chapter 11**

**Bibliography**

* https://lastminuteengineers.com/soil-moisture-sensor-arduino-tutorial/
* https://nodered.org/
* https://appinventor.mit.edu/explore/about-us.html
* http://watson-iot-sensor-simulator.mybluemix.net/

**Chapter 12**

**Appendix**

**12.1 Source Code**

* **Python Code:**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device Credentials

organization = "l0iga1"

deviceType = "rsip"

deviceId = "1001"

authMethod = "token"

authToken = "1234567890"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

print(type(cmd.data))

i=cmd.data['command']

if i=='motoron':

print("Motor is on")

elif i=='motoroff':

print("Motor is off")

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:

hum=30

#print(hum)

temp = 50

moi =40

#Send Temperature & Humidity to IBM Watson

data = {'d':{'temperature' : temp, 'humidity': hum,'soilmoisturesensor': moi}}

#print (data)

def myOnPublishCallback():

print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum, "Object Temperature = %s C" % moi, "to IBM Watson")

if moi <= 5:

print("Moisture is Low.....Motor Turned On !!!")

if moi >=90:

print("Moisture is High.....Motor Turned Off !!!")

success = deviceCli.publishEvent("DHT11", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(5)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device Credentials

organization = "l0iga1"

deviceType = "rsip"

deviceId = "1001"

authMethod = "token"

authToken = "1234567890"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

print(type(cmd.data))

i=cmd.data['command']

if i=='motoron':

print("Motor is on")

elif i=='motoroff':

print("Motor is off")

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:

hum=30

#print(hum)

temp = 50

moi =40

#Send Temperature & Humidity to IBM Watson

data = {'d':{'temperature' : temp, 'humidity': hum,'soilmoisturesensor': moi}}

#print (data)

def myOnPublishCallback():

print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum, "Object Temperature = %s C" % moi, "to IBM Watson")

if moi <= 5:

print("Moisture is Low.....Motor Turned On !!!")

if moi >=90:

print("Moisture is High.....Motor Turned Off !!!")

success = deviceCli.publishEvent("DHT11", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(5)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

**12.2 UI Screenshot Output**

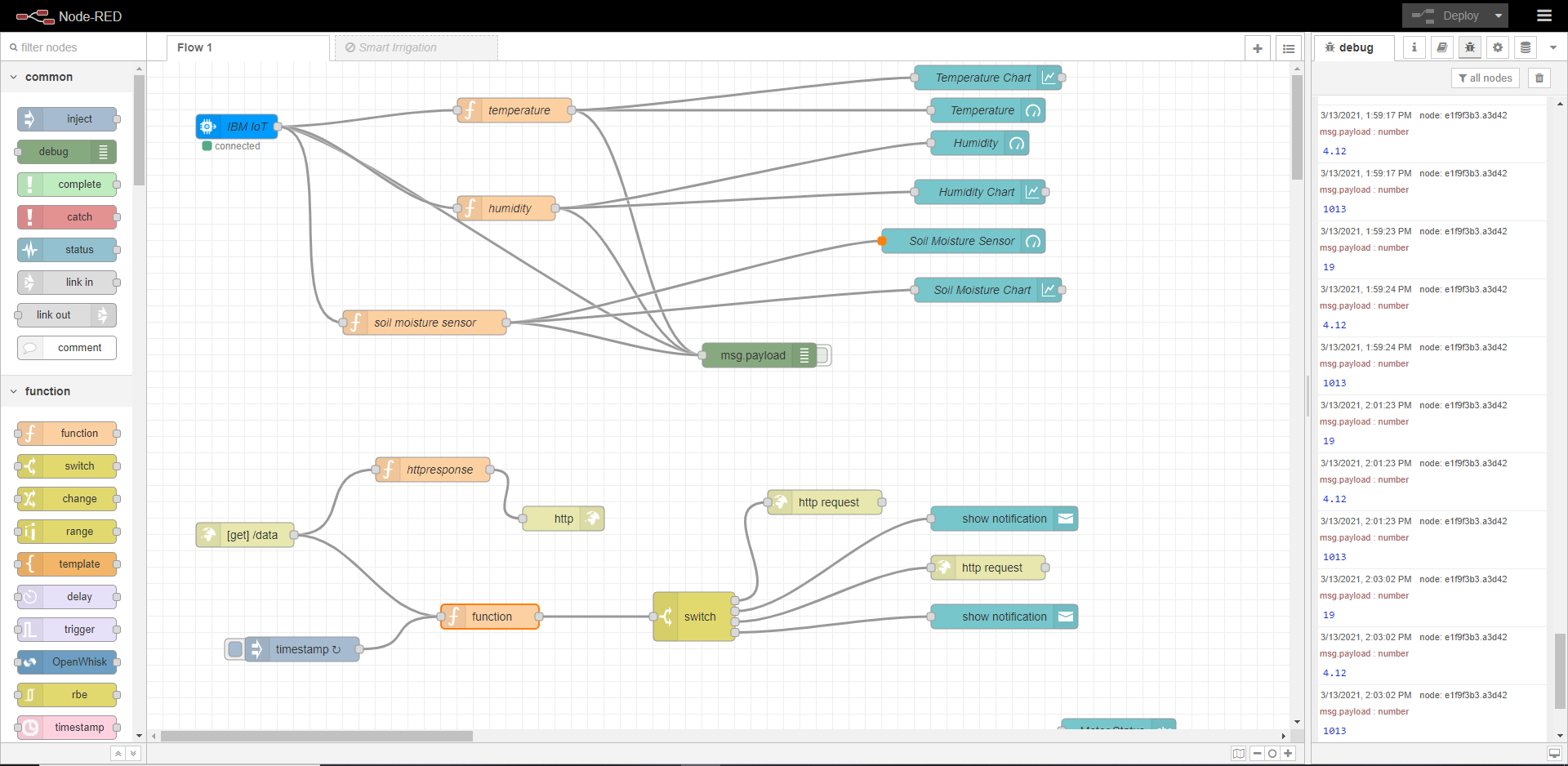
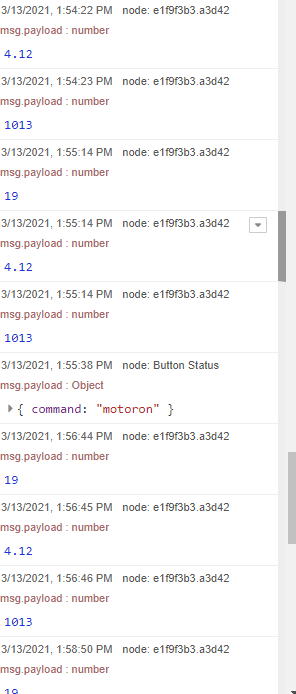
****

Figure 12.1 Node-red UI

Figure 12.2 Sensor Data

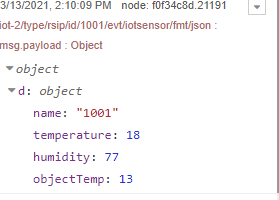


Figure 12.3 IBM Sensor Simulator

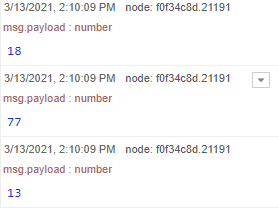


Figure 12.4 OpenWeather Climate Forecast

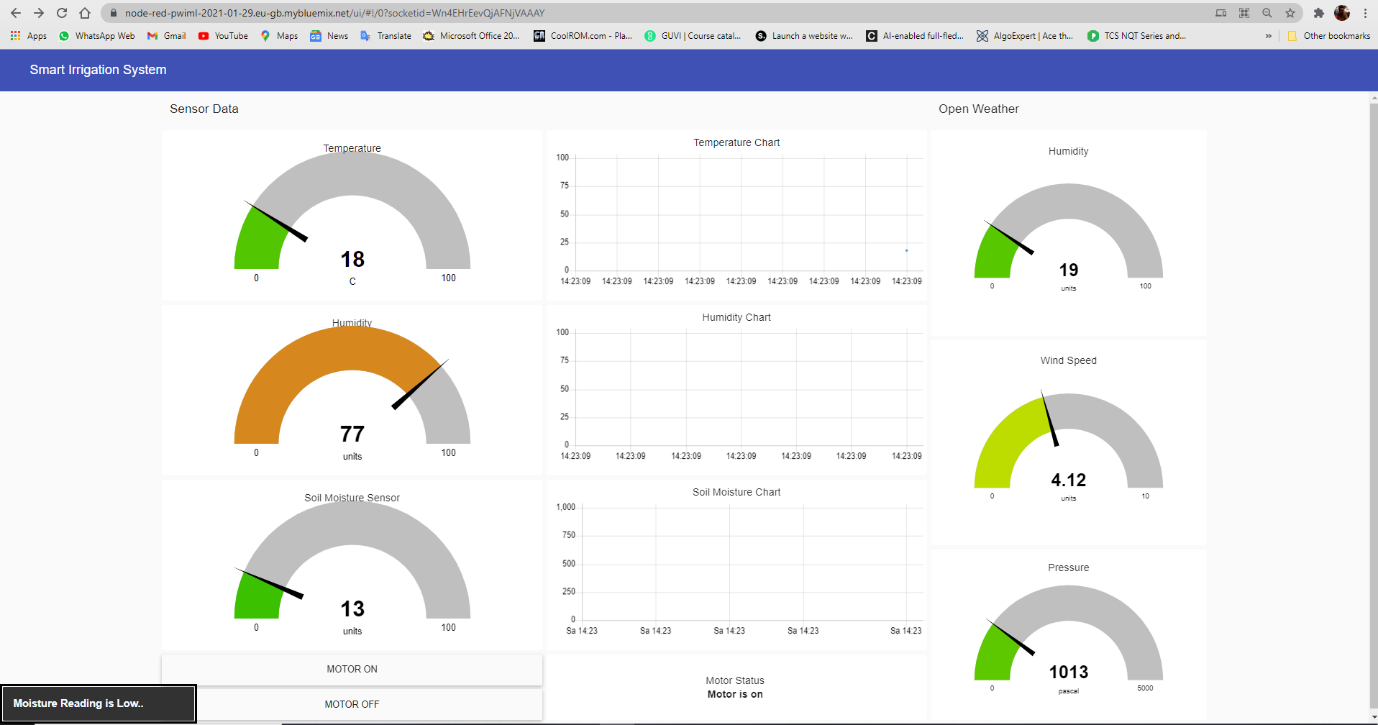


Figure 12.5 Node-red WEB UI

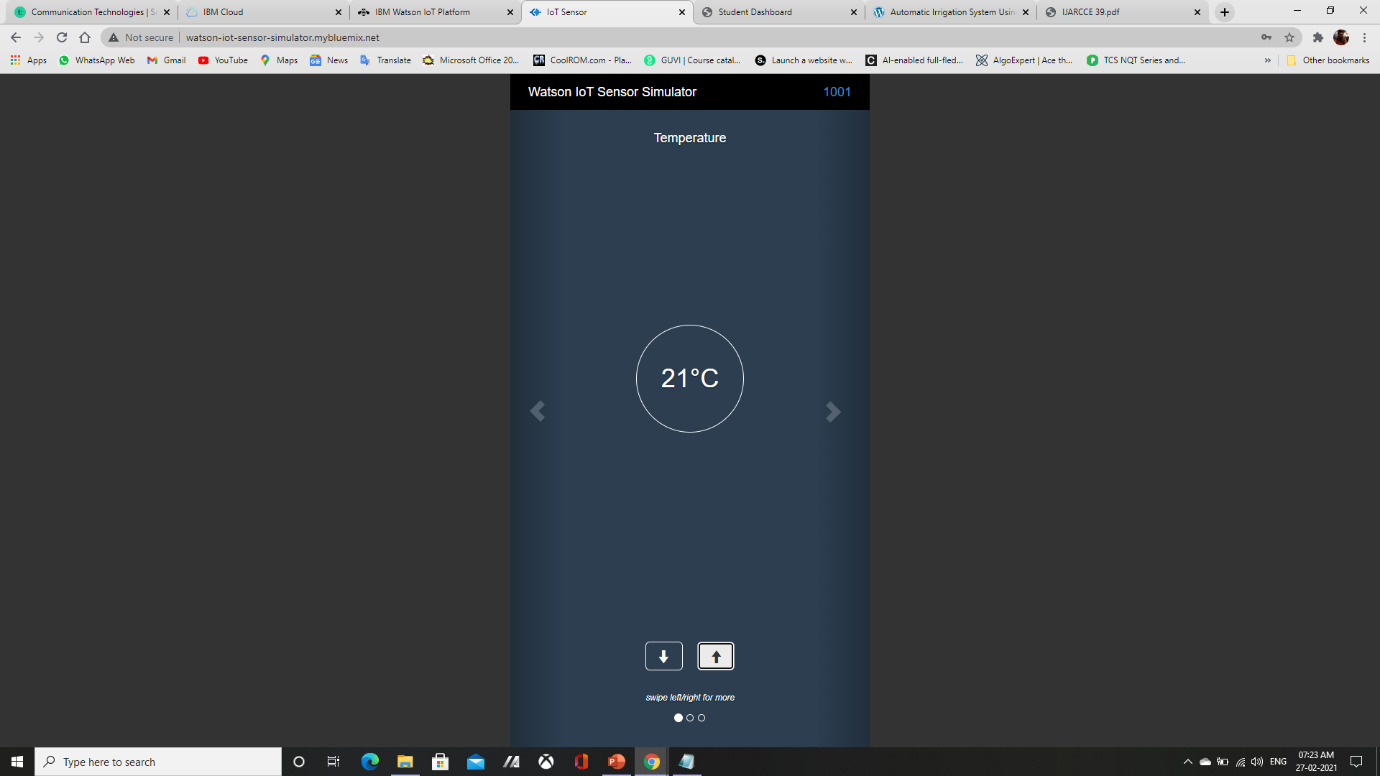


Figure 12.6 Sensor Simulator

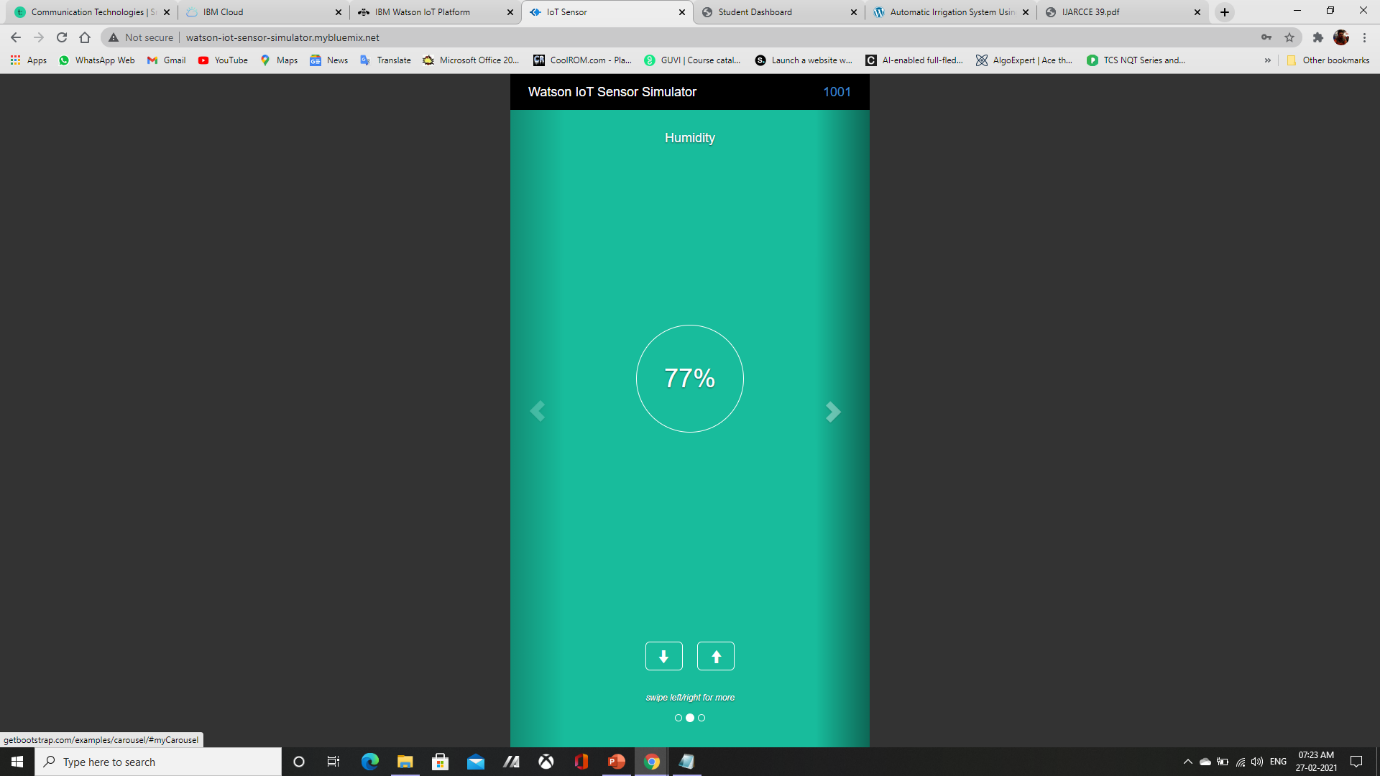


Figure 12.7 Sensor simulator(2)

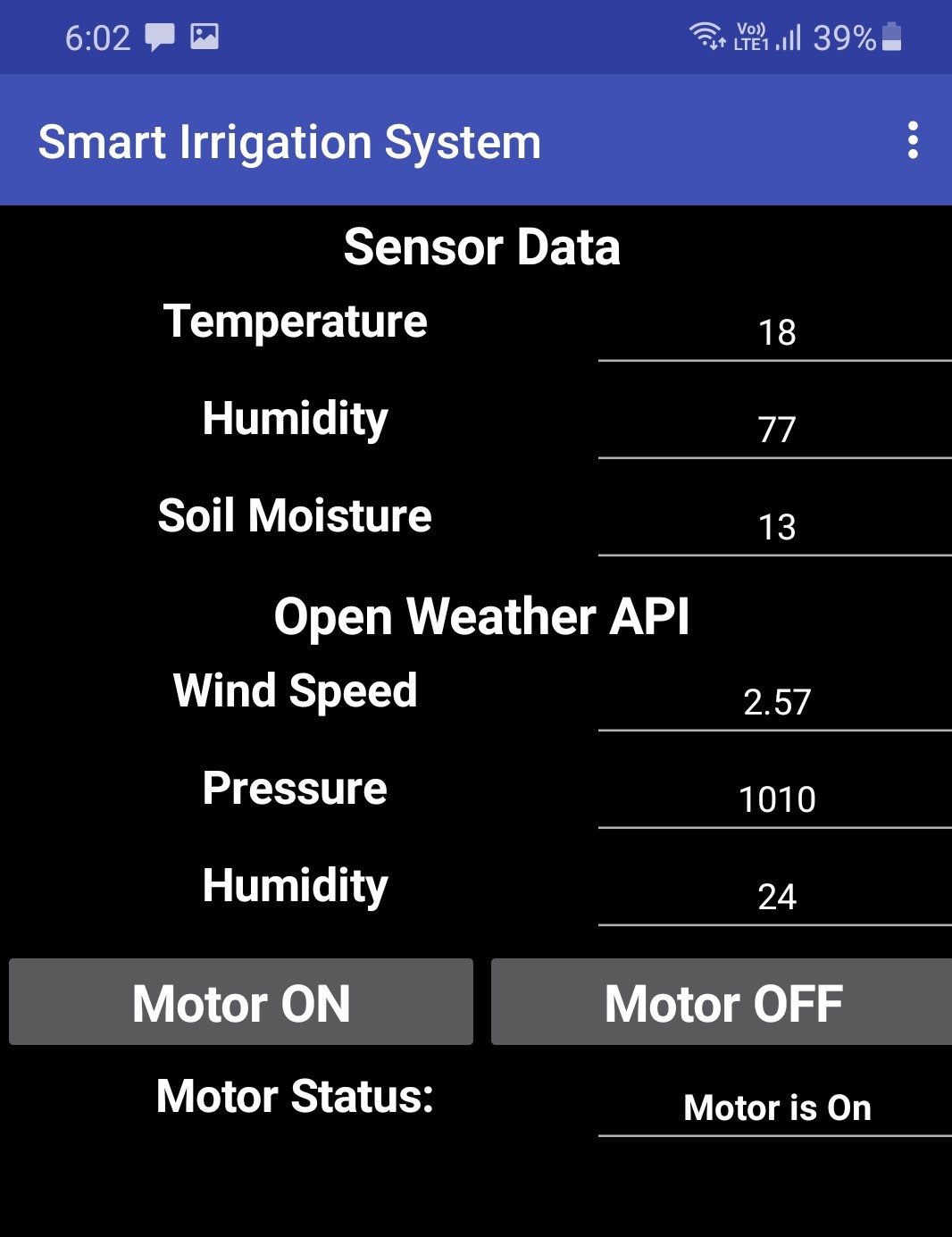


Figure 12.6 Mobile App Output