

**SMART FARMER - IOT ENABLED SMART
FARMING APPLICATION
NALAIYA THIRAN PROJECT**

Submitted by

JEEVIKA D (TEAM LEAD) -737819ECR070

MANEESHA A B -737819ECR098

KAUSHIKA S -737819ECR077

HARISH K -737819ECR055

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Team ID: PNT2022TMID04642

PROJECT REPORT

TABLE OF CONTENTS

S.No	NAME OF THE CHAPTER	PAGE NUMBER
1.	INTRODUCTION	3
2.	LITERATURE SURVEY	4
3.	IDEATION AND PROPOSED SOLUTION	7
4.	REQUIREMENT ANALYSIS	13
5.	PROJECT DESIGN	15
6.	PROJECT PLANNING AND SCHEDULING	17
7.	CODING AND SOLUTIONING	24
8.	PERFORMANCE METRICS	36
9.	ADVANTAGES AND DISADVANTAGES	37
10.	CONCLUSION	38
11.	FUTURE SCOPE	39
12.	APPENDIX	40

CHAPTER 1 – INTRODUCTION

1.1 Project Overview

Any economic development has always been based on agriculture. Agriculture needs to be connected with contemporary practises and technologies to support future growth. Technology can be employed in farming to make farmers' tasks easier to complete given its widespread adoption. Numerous personal assistant devices now make use of electronics and IoT. This can be applied to many important industries, such as agriculture, where their assistants might assist in resolving several problems. Devices can use electronics to evaluate and gather data, as well as to link physically to their operational environment. IoT can assist with data analysis and transmission to the user. When these are combined, an all-in-one tool that can perform a task is created.

1.2 Purpose

Farmers have recently had difficulty anticipating the ideal weather conditions to start farming due to the unpredictable weather and climatic fluctuations. Although it appears unexpected at first glance, crop planning may be done by using certain characteristics to forecast it. It's crucial to maintain farmland both during and after agriculture. Temperature, humidity, and soil moisture measurements can be used to carry out these. Physical sensors are used in the measurement of these properties. This technology is linked to an IoT system, which can offer farmers a simple interface to read, evaluate, and act based on the situation as it is now. A step further, the technology may automate the operation of motors and other electrical farming equipment by gaining access to them. This can aid in assuring accuracy and faster response times during unattended operation.

CHAPTER 2 - LITERATURE SURVEY

2.1 Existing problem

Various approaches and solutions have been made to assist farmers in implementing technology methods. Only a few solutions had their performance constrained by suggestions and alerts. While few used independent electronics for IoT. Below is a brief description of a few examples of earlier attempts and studies.

- i. IoT-based smart sensors farm stick for real-time temperature and moisture monitoring using solar power, the cloud, and Arduino. Things Speak, a cloud computing platform, was used for this work's data collecting. DHT 11 sensors and an Arduino board were used to create the circuit.
- ii. IoT-based smart farming is a way to monitor farming conditions in the best way possible. The ESP-32 based IoT platform and Blynk mobile application were employed in this experiment.
- iii. “Smart farming using IoT”. The automation and interface part made use of water pump and HTTP protocol for parameters monitoring using website. The aforementioned earlier works were missing one or two elements that, if present, could have improved performance. In the first project, switching to a Raspberry Pi-based controller from an Arduino-based one can assist shrink the design space while also giving the microcontroller more UI and IoT interfaces. In the second mentioned project, switching from the Blynk application to the MIT app inventor can increase the likelihood of feature extension. In order to add new features, farmers or developers won't need to purchase a commercial edition of the software. 6 In the third piece of work,

servo-based water valves are used instead of bistated logic to improve the control of water pumps by directing and controlling the flow of water.

2.2 References

The following were the source of references:

- i. https://www.researchgate.net/publication/313804002_Smart_farming_IoT_based_smart_sensors_agriculture_stick_for_live_temperature_and_moisture_monitoring_using_Arduino_cloud_computing_solar_technology.
- ii. “Smart Farming Using IOT”, CH Nishanthi; Dekonda Naveen, Chiramdasu Sai Ram , Kommineni Divya , Rachuri Ajay Kumar; ECE Dept., Teegala Krishna Reddy Engineering College, Hyderabad, India 2,3,4,5student, ECE Dept., Teegala Krishna Reddy Engineering College, Hyderabad, India.
- iii. “Smart farming: IoT based smart sensors agriculture stick forlive temperature and moisture monitoring using Arduino, cloud computing & solar technology”, Anand Nayyar Assistant Professor, Department of Computer Applications & IT KCL Institute of Management and Technology, Jalandhar, PunjabEr. Vikram Puri M.Tech(ECE) Student, G.N.D.U Regional Center, Ladewali Campus, Jalandhar
- iv. “Smart Farming using IoT, a solution for optimally monitoring farming conditions”, Jash Doshi; Tirth kumar ; Patel Santosh kumar Bharati. 8

2.3 Problem Statement

In a nutshell, the issue statement encompasses all the technological elements that a farmer may use to transform farming into smart and effective farming. On a larger scale, IoT-enabled smart farming focuses on integrating all the separately operational farming automation subsystems into a unified entity. The farmer may monitor several field characteristics, such as soil moisture, temperature, and humidity, using an IoT-based agriculture system.

With the use of mobile and online applications, the IoT concept is further expanded so that farmers can keep track of all sensor values even while they are far from their fields. One of the crucial tasks for farmers is to water the crops. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and controlling the motor pumps from the mobile application itself.

CHAPTER 3 - IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

Top 4 ideas:

1st Idea:

Name: JEEVIKA D

Idea name: For farming parameters, a user-friendly programme

Description: A complete application with sophisticated features to monitor all the field parameters and making it user friendly so that every farmer can benefit from that.

2nd Idea:

Name: MANEESHA A B

Idea name: A product and feature guide on the internet

Description: A website that describes the features of the application, also provides important advices in farming regarding optimum conditions, crop health, use of resources etc. With this, along with farmers any hobbyist who is interested in farming can start cultivating in an efficient way with this guide.

3rd Idea:

Name: KAUSHIKA S

Idea name: Raspberry pi usage for IoT applications as opposed to using another processor

Description: for real time monitoring and high processing power, usage of raspberry pi is a better choice over Arduino and other microcontrollers. And to utilize IoT related applications, raspberry pi platform is the best choice.

4th idea:

Name: HARISH K

Idea name: An app to report the parameters to the farmers

Description: Monitoring the weather and soil continuously by the usage of raspberry pi is a better choice over Arduino and other microcontrollers. And to utilize IoT related applications, raspberry pi platform is the best choice

16 ideas from the brain storming session

JEEVIKA D

Idea 1: User friendly application for farmland parameters

Idea 2: Awareness about IoT in agricultural domain through various applications

Idea 3: Computer vision for detecting crop diseases

Idea 4: Automating watering process to save water

MANEESHA A B

Idea 1: Usage of raspberry pi over other processor for IoT applications.

Idea 2: Deciding the specification of sensors to be deployed based on the range of operation per square feet of farmlands.

Idea 3: Protection case for sensors to avoid wear and tear during adverse conditions.

Idea 4: Using RF based communication along with IoT protocols to deliver data.

KAUSHIKA S

Idea 1: A website guide about the product and features

Idea 2: Provision through application to remotely control agricultural instruments

Idea 3: Data collection about various field parameters

Idea 4: Features in website to control agricultural actuators

MOHAMMED ANEESUDDIN J

Idea 1: Various protocols used in IoT.

Idea 2: Application with simple UI but efficient usage.

Idea 3: Interface between website, application and sensors.

Idea 4: Utilizing ai to improvise accuracy.

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Watering plants takes long time and requires continuous monitoring Using water and acting accordingly by predicting the weather is a great challenge for farmers

		Protection of crops from pesticides and disease is also difficult
2.	Idea / Solution description	<p>The Data collected by sensors, In terms of humidity, temperature, moisture, and dew detections help in determining the weather pattern and can be used by the farmers accordingly for farming</p> <p>By determining the acidity level of the soil, the usage of pesticides and fertilizers can be determined</p>
3.	Novelty / Uniqueness	<p>The farmer can be alerted for watering the crops and the values from the sensors can be used by the farmers for various applications</p> <p>The app will help the farmer to find the condition of crops from anywhere</p>
4.	Social Impact / Customer Satisfaction	<p>It saves a lot of time and reduces man power</p> <p>The money spent for wages can be reduced</p> <p>Accurate prediction of weather and watering</p>
5.	Scalability of the Solution	Scalability of the product is large as it targets all the farmers. Since everyone uses smart phones now a days the scalability will be high

3.4 Problem Solution fit

Define CS, fit into CC	<div>1. CUSTOMER SEGMENT(S)<div>CS</div></div> <div>Who is your customer?</div> <div>The customers for this product are farmer. The goal of the product is to help them, monitor field parameters remotely. This product can saves agriculture from extinction.</div>	<div>6. CUSTOMER CONSTRAINTS<div>CC</div></div> <div>What constraints prevent your customers from taking action or limit their choices of solutions?</div> <div>Using many sensors is difficult. An unlimited or continuous internet connection may be required for success.</div>	<div>5. AVAILABLE SOLUTIONS<div>AS</div></div> <div>Which solutions are available to the customers when they face the problem</div> <div>or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? I.e. pen and paper is an alternative to digital notetaking</div> <div>The irrigation process is automated and the data collected from the sensors and the weather data can be used by the farmers</div>	Explore AS, differentiate
	<div>2. JOBS-TO-BE-DONE / PROBLEMS<div>J&P</div></div> <div>Which jobs-to-be-done (or problems) do you address for</div> <div>The purpose of this product is to use sensors to acquire various field parameters and process them using a central processing system. The cloud is used to store and transmit data using IoT. The Weather API is used to help farmers make decisions. Farmers can make decisions through mobile applications. your customers? There could be more than one; explore different sides.</div>	<div>9. PROBLEM ROOT CAUSE<div>PR</div></div> <div>What is the real reason that this problem exists? What is the back story behind the need to do this job?</div> <div>Frequent changes and unpredictable weather and climate made it difficult for farmers to engage in agriculture. These factors play an important role in deciding whether to water your plants.</div>	<div>7. BEHAVIOUR<div>BE</div></div> <div>Use a proper drainage system to overcome the effects of excess water from heavy rain. Use of hybrid plants that are resistant to pests.</div>	
Focus on J&P, tap into	<div>3. TRIGGERS<div>TR</div></div> <div>What triggers customers to act? I.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.</div> <div>Farmers find it difficult to provide correct amount of irrigation. Inadequate water supply reduces yields and affects farmers' profit levels. Farmers have a hard time predicting the weather.</div>	<div>10. YOUR SOLUTION<div>SL</div></div> <div>If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.</div> <div>If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.</div> <div>The app collects data from various types of sensors and sends the values to our main server. It also collects weather data from the Weather API. The final decision to irrigate the crop is made by the farmer using a mobile application.</div>	<div>8. CHANNELS of BEHAVIOUR<div>CH</div></div> <div>8.1 ONLINE</div> <div>What kind of actions do customers take online? Extract online channels from #7</div> <div>8.2 OFFLINE</div> <div>What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</div> <div>ONLINE: Providing online assistance to the farmer, in providing knowledge regarding the pH and moisture level of the soil. Online assistance to be provided to the user in using the product.</div> <div>OFFLINE: Awareness camps to be organized to teach the importance and advantages of the automation and IoT in the development of agriculture.</div>	
	<div>4. EMOTIONS: BEFORE / AFTER<div>EM</div></div> <div>How do customers feel when they face a problem or a job and afterwards? I.e. lost, insecure > confident, in control - use it in your communication strategy & design.</div> <div>BEFORE: Lack of weather predicting and forecasting → Random decisions → low yield. AFTER: Data from reliable source → correct decision → high yield</div>			

CHAPTER 4 - REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Functional requirements involve the hardware and software components needed to design the system. They are:

- i) Sensors: Rain sensor, DHT11–Temperature and Humidity sensor, Soil Moisture sensor.
- ii) Actuators: Water pumps, Motors, Servos.
- iii) MCUs (Microcontroller Units): Raspberry Pi, ESP-8266.
- iv) Software Components: Web UI, Node red, IBM Watson as the cloud platform, Mobile application using MIT App inventor.

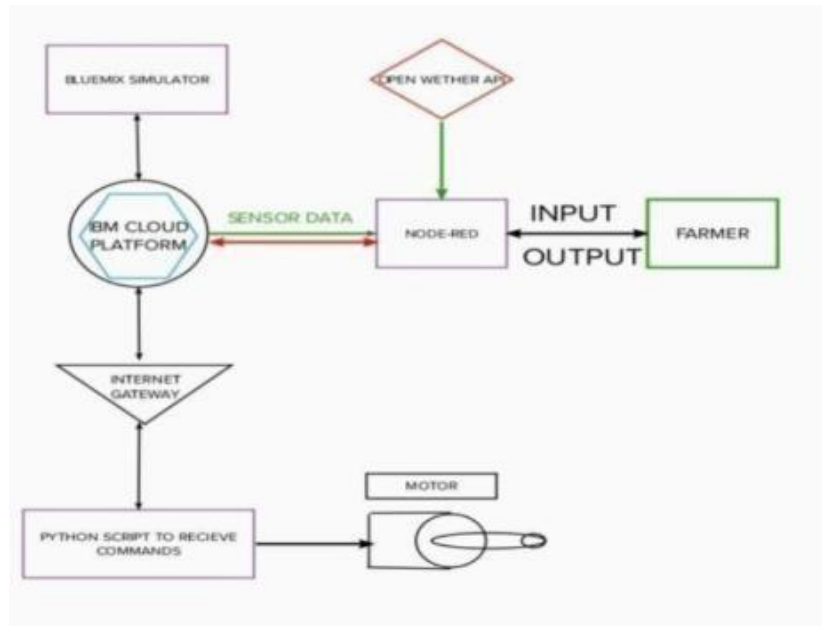
4.2 Non-functional Requirements: Non-functional requirements deal more of a customer or business model point of view. These requirements play a major role when the project is ready as a market product. Some of those non-technical requirements are:

- 1) Usability: Can be used for both large scale agricultural farms and domestic gardens for soil monitoring and watering of plants.
- 2) Security: Since the user uses his/her own cloud account to store and process sensor data, data privacy is maintained to a significant extent.
- 3) Reliability: Inclusion of real-time monitoring of sensor data and interactive mobile application makes the product more reliable.
- 4) Performance: Performance of the system is significantly high as MCUs with high processing capability such as Raspberry Pi are being used.

- 5) Availability: After successful completion of the design, the model will be available in the market, and people can purchase the product according to their requirements.
- 6) Scalability: The design can be scaled to be used for large sized farms by including sophisticated hardware components and sensors like TDS sensor, according to the requirements and physical parameters.

CHAPTER 5 - PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution and Technical Architecture

The technical architecture diagram is as follows:

20 Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local/ Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components /services

5. Indicate interface to machine learning models (if applicable)

- The different soil parameters temperature, soil moistures and humidity are sensed using different sensors and obtained value is stored in the IBM cloud.
- Here, instead of using Raspberry Pi processor unit, random values are generated for various soil parameters using Python.
- NODE-RED is used as a programming tool to write the hardware, software, and APIs. The MQTT protocol is followed for the communication.
- All the collected data are provided to the user through a mobile application that was developed using the MIT app inventor. The user could decide through an app, weather to water the crop or not 21 depending upon the sensor values. By using the app, they can remotely operate the motor switch.

CHAPTER 6 - PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation



6.2 Sprint Delivery and Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Member
Sprint -1	Registration (Farmer Mobile User)	UNS-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Jeevika D (Leader)
Sprint -1	Login	UNS-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Maneesha A B (Member 1)

Sprint-2	User Interface	UNS-3	As a user, I can registerfor the application through Facebook	3	Low	Kaushika S (Member 2)
Sprint-1	Data Visualiz ation	UNS-4	As a user, I can register for the application through GMAIL	2	Mediu m	Harish K (Member 3)
Sprint-3	Registrat ion (Farmer - Web User)	USN - 1	As a user, I can log intothe application by entering email and password	3	High	Jeevika D(Leader)
Sprint - 2	Login	USN - 2	As a registered user, I need to easily login log into my registered account via the web page in minimum time	3	High	Jeevika D (Leader)
Sprint - 4	Web UI	USN - 3	As a user, I need to have a friendly user interface to easily view and	3	Mediu m	Maneesha A B (Member 1)

			access the resources			
Sprint - 1	Registration (Chemical Manufacturer -Web user)	USN - 1	As a new user, I want to first register using my organization email and create a password for the account.	2	High	Kaushika S (Member 2)

Sprint - 4	Login	USN - 2	As a registered user, I need to easily log in using the registered account via the web page.	3	High	Harish K (Member 3)
Sprint - 3	Web UI	USN - 3	As a user, I need to have a user friendly interface to easily view and access the resources.	3	Medium	Maneesha A B (Member 1)
Sprint - 1	Registration (Chemical Manufacturer - Mobile User)	USN - 1	As a user, I want to first register using my email and create a password for the account.	1	High	Jeevika D (Leader)
Sprint - 1	Login	USN - 2	As a registered user, I need to easily log in to the application.	2	Low	Kaushika S (Member 2)

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	12	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	20	30 OCT 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	20	6 NOV 2022
Sprint-4	6	6 Days	14 Nov 2022	19 Nov 2022	20	7 NOV 2022

Velocity:

AV for sprint 1= Sprint

Duration /velocity =12/6=2

AV for sprint 2= Sprint

Duration/Velocity=6/6=1

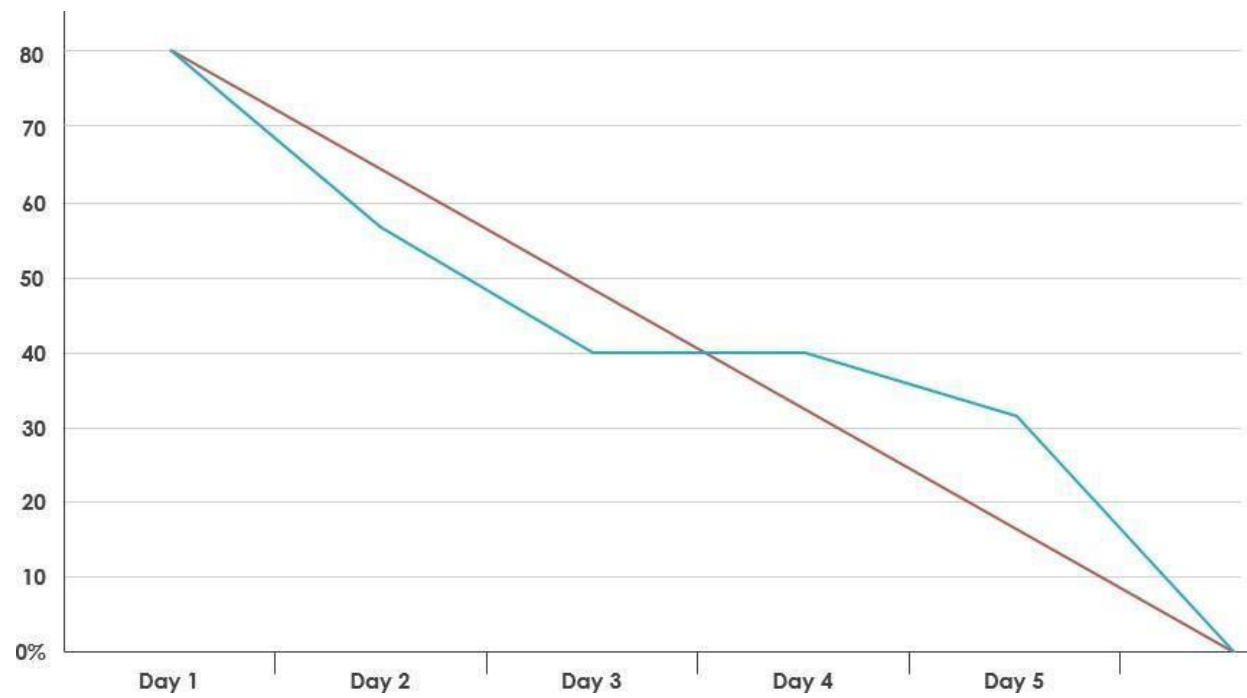
AV for Sprint 3=Sprint

Duration/Velocity=6/6=1

AV for Sprint 4=Sprint

Duration/Velocity=6/6=1

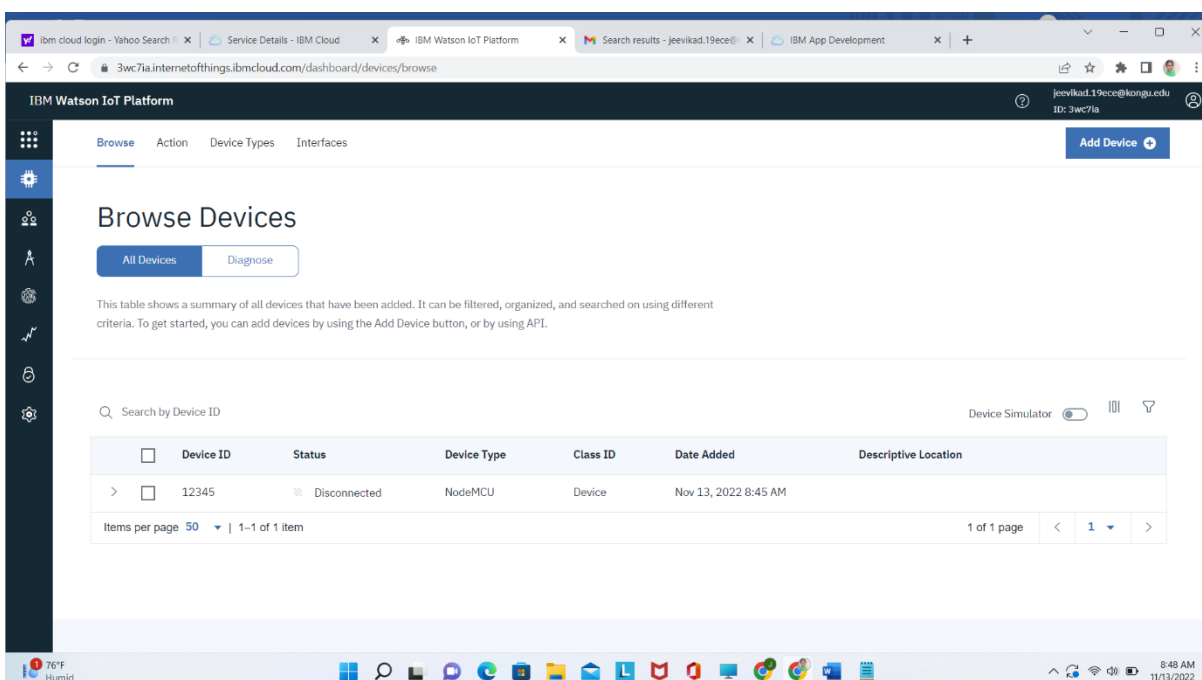
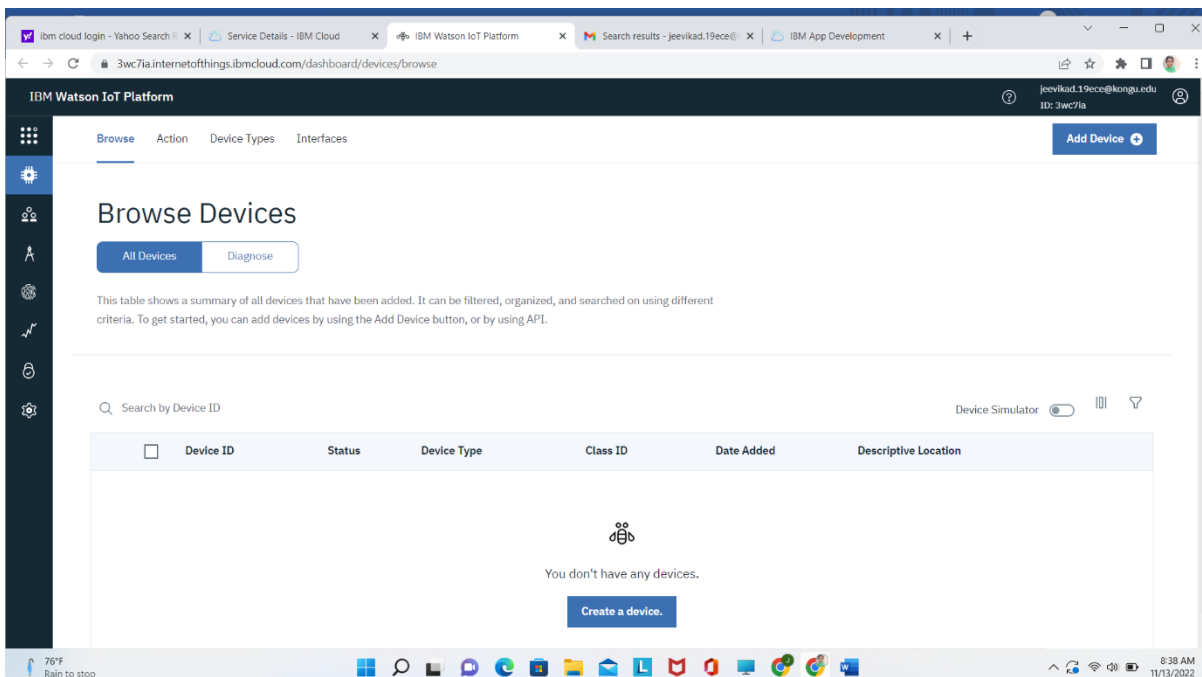
Burndown Chart:



CHAPTER 7 - CODING AND SOLUTION

CREATING IBM WATSON IoT PLATFORM AND CREATING A DEVICE ON IOT

In the IBM Watson IoT Platform, under the catalog list, under the Internet of Things platform, a device has been created. From that the device credentials such as Device ID, Device Type, Organization ID, Authentication token were obtained.



1) Connecting IOT Simulator to IBM Watson IOTPlatform

Give the credentials of your device in IBM Watson IOTPlatform

Click on connect

My credentials given to simulator are:

OrgID: 3wc7ia

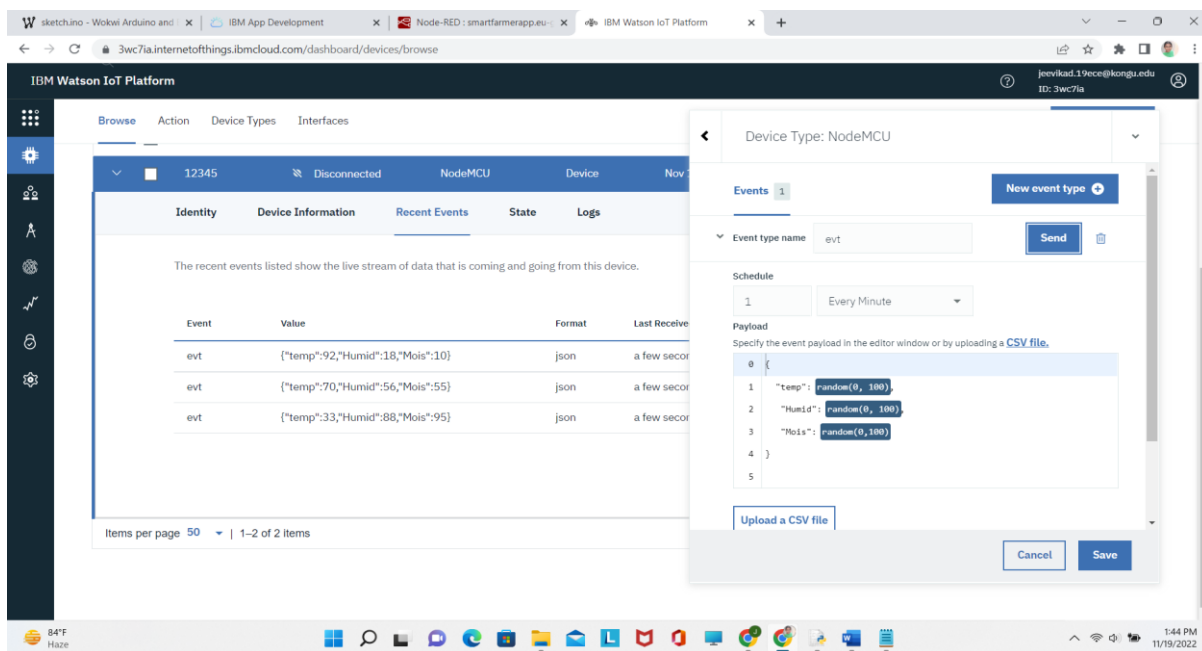
api: a-3wc7ia-gqztwdvblr

Device type:NodeMCU

Token: 8cdIkQot_LAzMFmzAE

Device ID : 12345

Device Token : 12345678



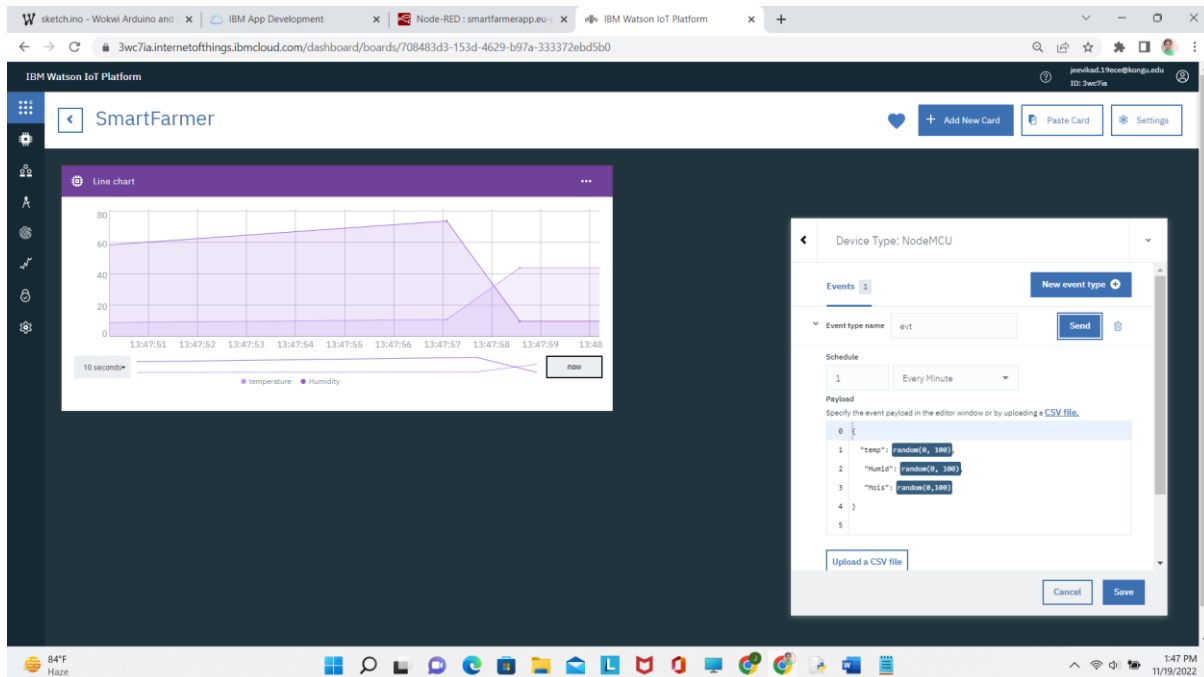
You can see the received data in graphs by creating cards in Boards tab

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": "NodeMCU", "temperature": 17, "humidity": 76, "Moisture": 25 } }



2) Configure Node-Red to IBM Cloud

The node IBM IOT App In is added to Node-Red workflow.

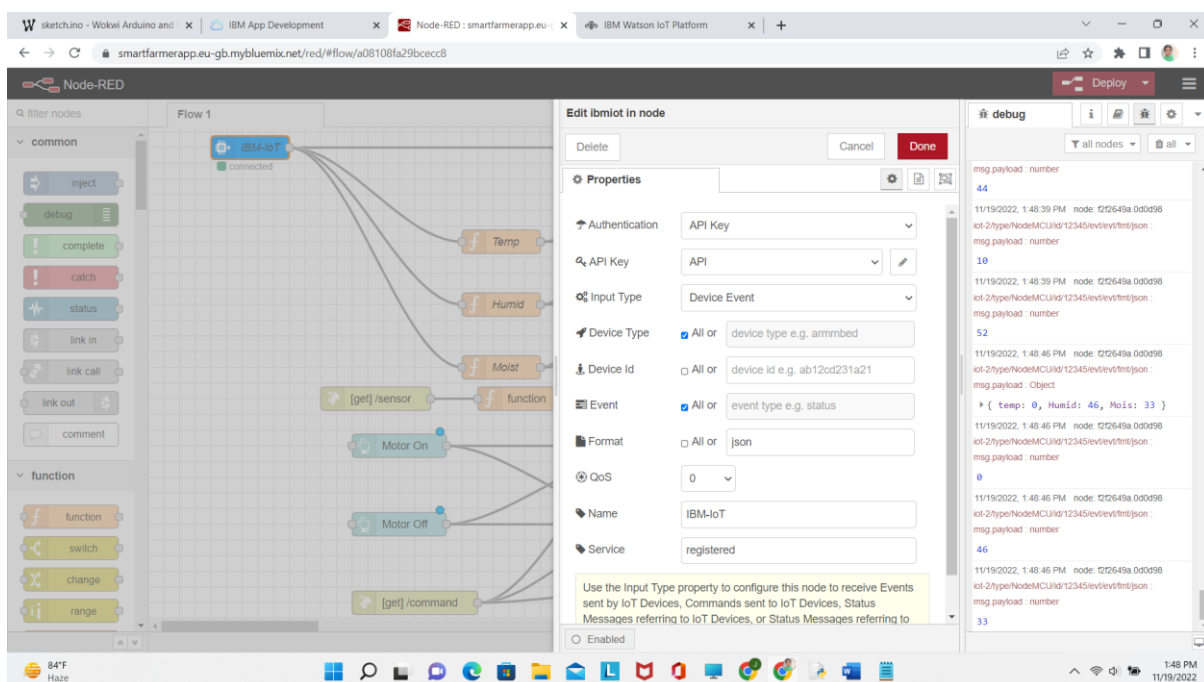
Then the appropriate device credentials obtained earlier are entered into the node to connect and fetch device telemetry to Node-Red.

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.

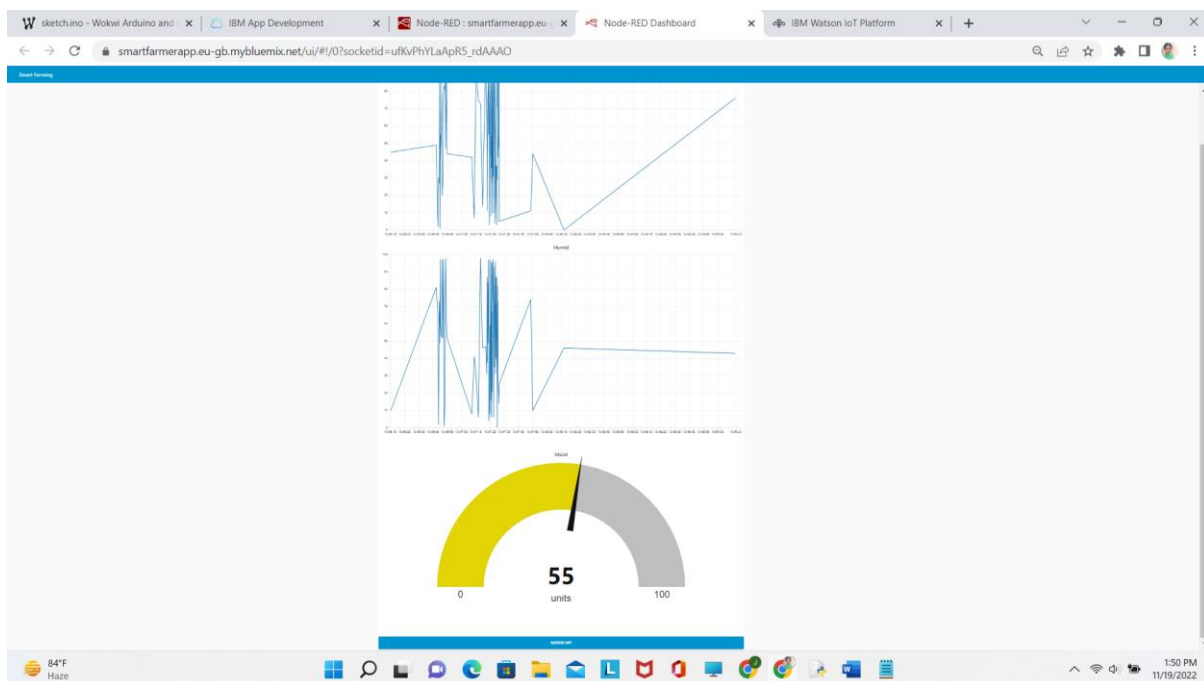
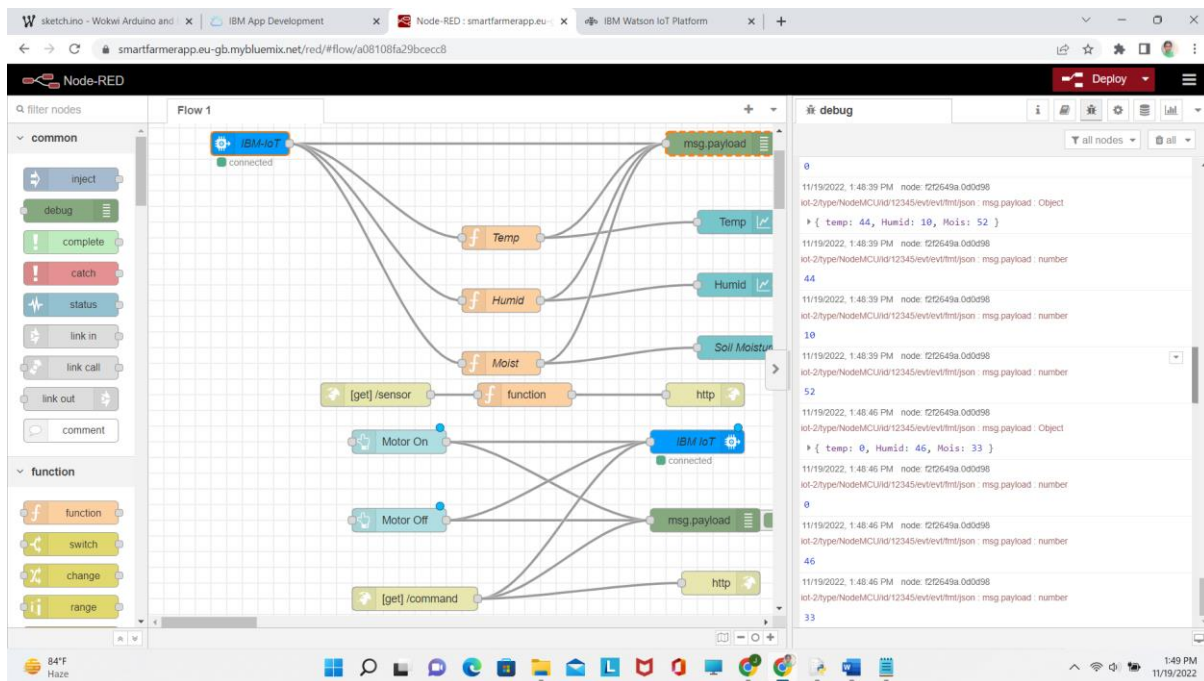
The Java script code for the function node is:

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

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



Data received from the cloud in Node-R



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

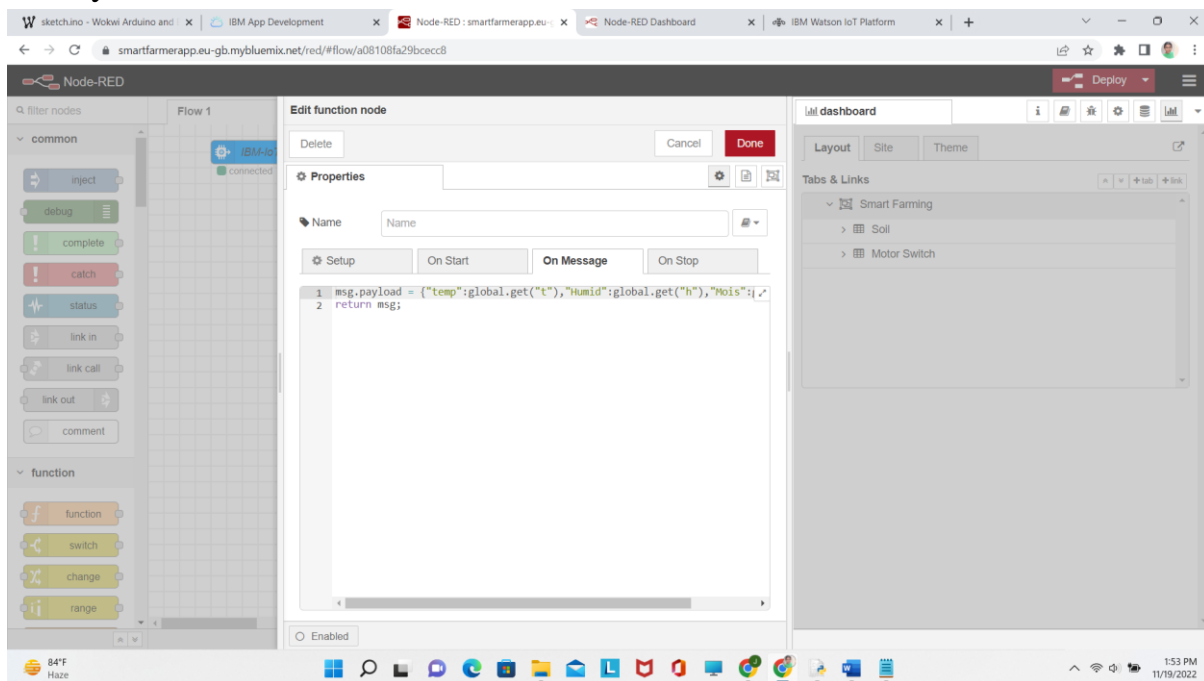
Configuration of Node-Red to collect data from Open Weather

The Node-Red also receive data from the Open Weather API by HTTP GET request.

An inject trigger is added to perform HTTP request for every certain interval.

```
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



Configuring Node-Red to receive and pass data

Ibm iot 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 two buttons in UI

- 1 -> for motor on
- 2 -> for motor off

We used a function node to analyses the data received and assign command to each number. The Java script code for the analyses is:

```
if(msg.payload=="motoron")
msg.payload={"command": "motoron"};
else if(msg.payload=="motoroff")
msg.payload={"command": "motoroff"};
```

Node-RED interface showing the configuration of an IBM IoT node. The left sidebar displays the "common" and "function" node palettes. The main workspace shows a flow with nodes: inject, debug, complete, catch, status, link in, link call, link out, comment, function, switch, change, and range. The "Edit ibmiot out node" dialog is open, showing the following properties:

- Authentication: API Key
- API Key: API
- Output Type: Device Command
- Device Type: NodeMCU
- Device Id: 12345
- Command Type: cmd
- Format: json
- Data: data
- QoS: 0
- Name: IBM IoT
- Service: registered

The "debug" console on the right shows the following log entries:

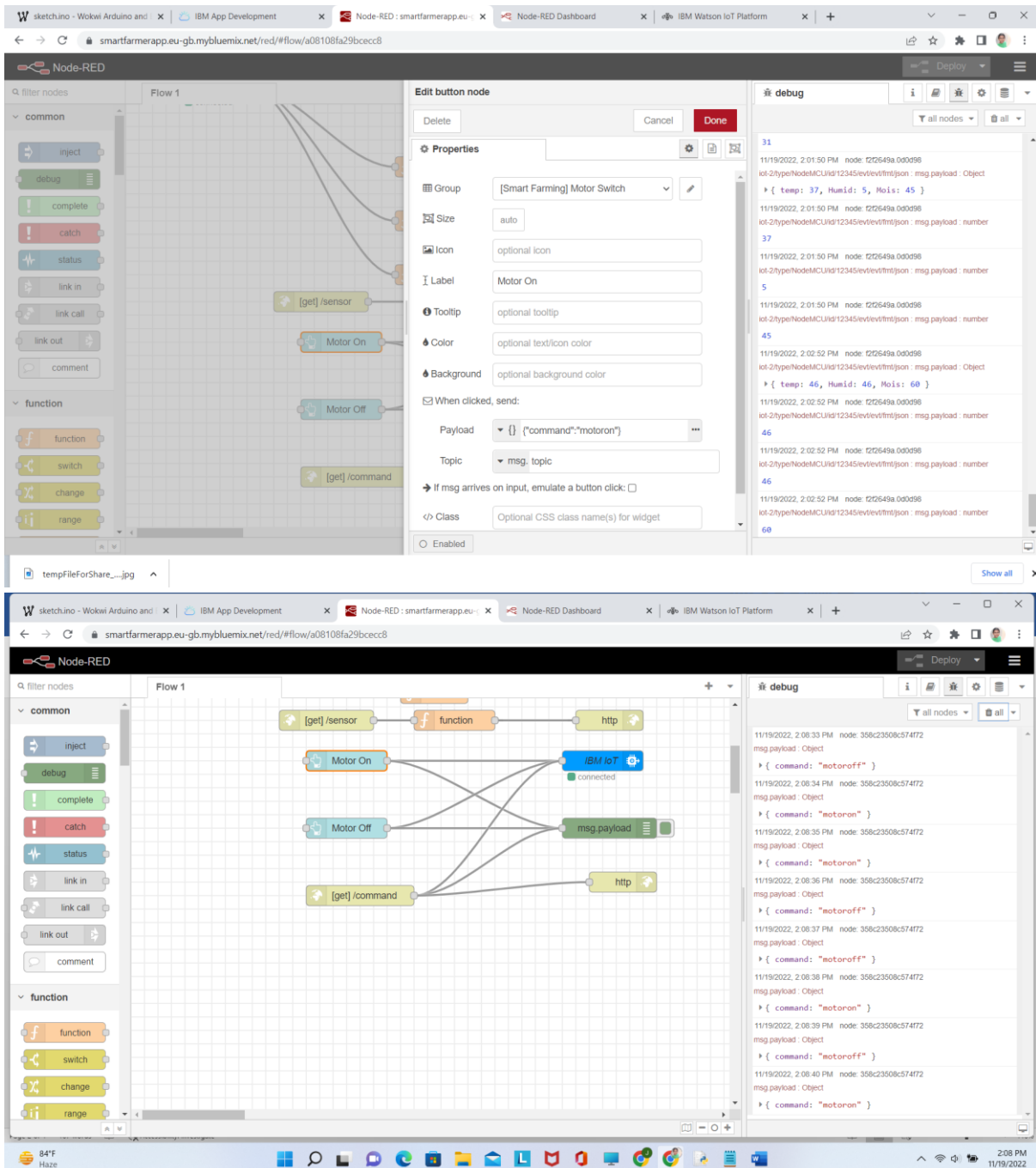
```
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: Object  
* { temp: 37, Humid: 5, Mois: 45 }  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
37  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
5  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
45  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: Object  
* { temp: 46, Humid: 46, Mois: 60 }  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
46  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
46  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
60
```

Node-RED interface showing the configuration of an HTTP node. The left sidebar displays the "common" and "function" node palettes. The main workspace shows a flow with nodes: inject, debug, complete, catch, status, link in, link call, link out, comment, function, switch, change, and range. The "Edit http in node" dialog is open, showing the following properties:

- Method: GET
- URL: /command
- Name: Name

The "debug" console on the right shows the following log entries:

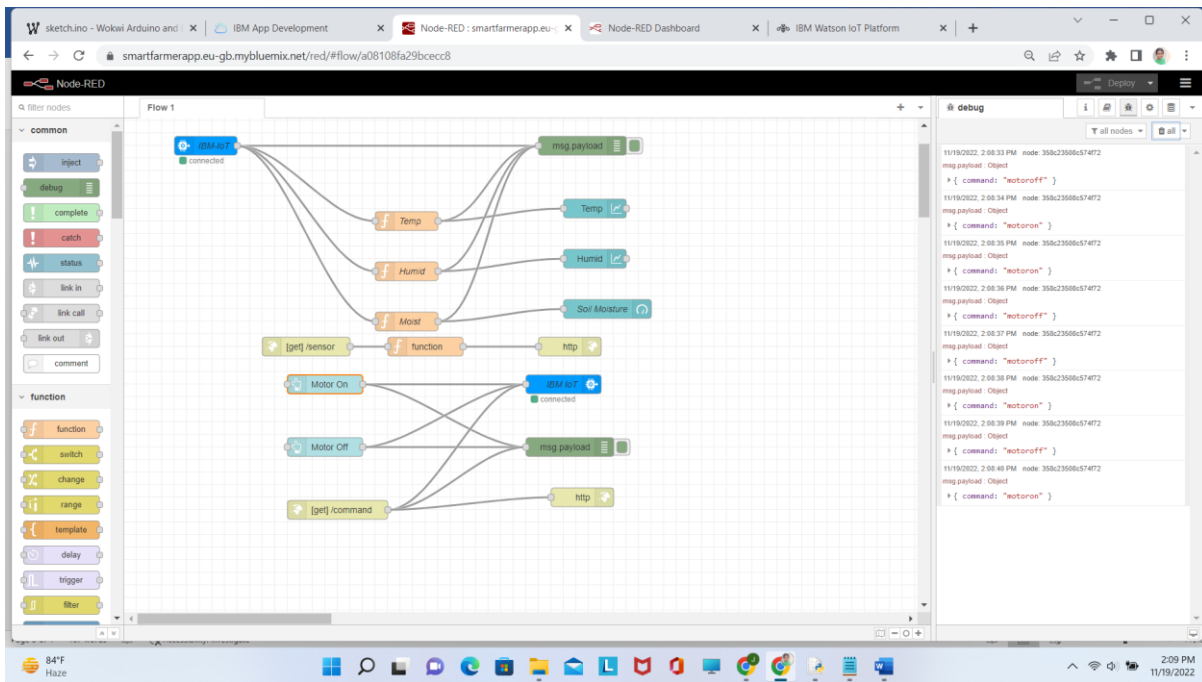
```
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: Object  
* { temp: 37, Humid: 5, Mois: 45 }  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
37  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
5  
11/19/2022, 2:01:50 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
45  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: Object  
* { temp: 46, Humid: 46, Mois: 60 }  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
46  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
46  
11/19/2022, 2:02:52 PM node: i2D649a.0d0d98  
i2D649a.0d0d98: msg payload: number  
60
```



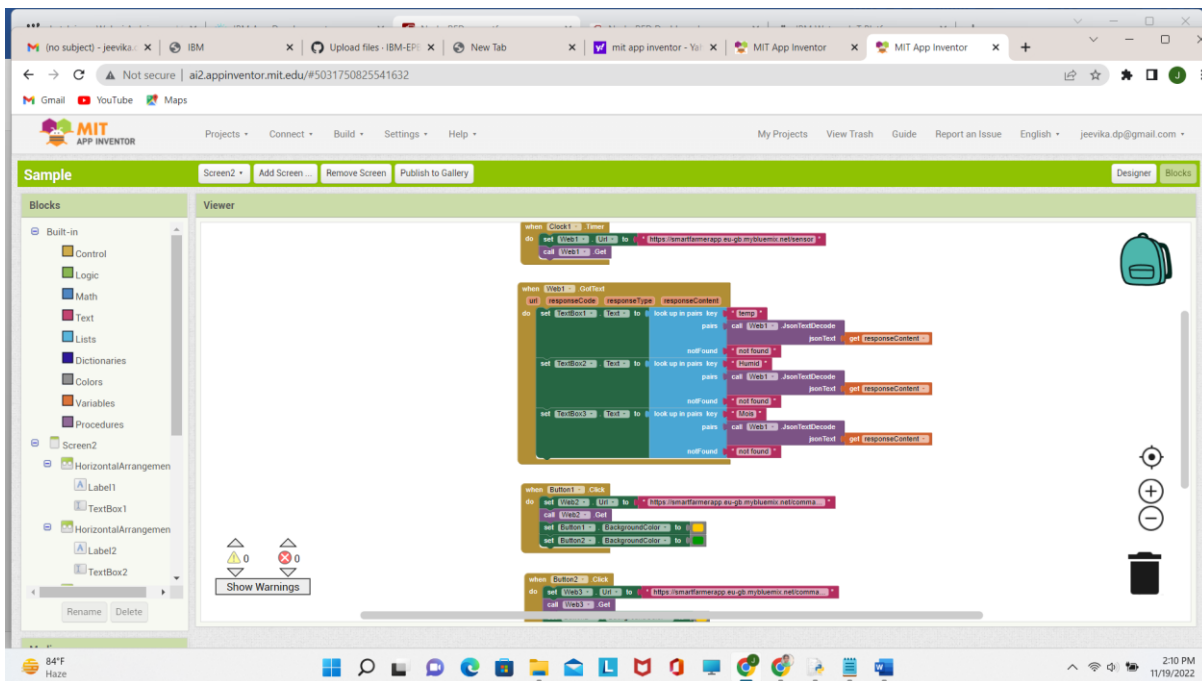
Adjusting User Interface

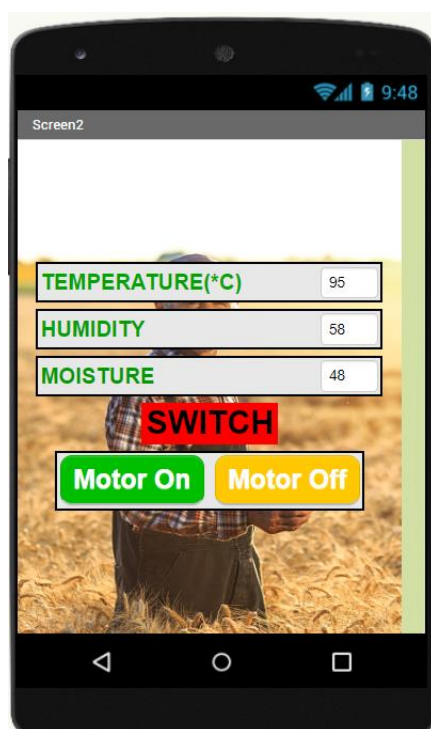
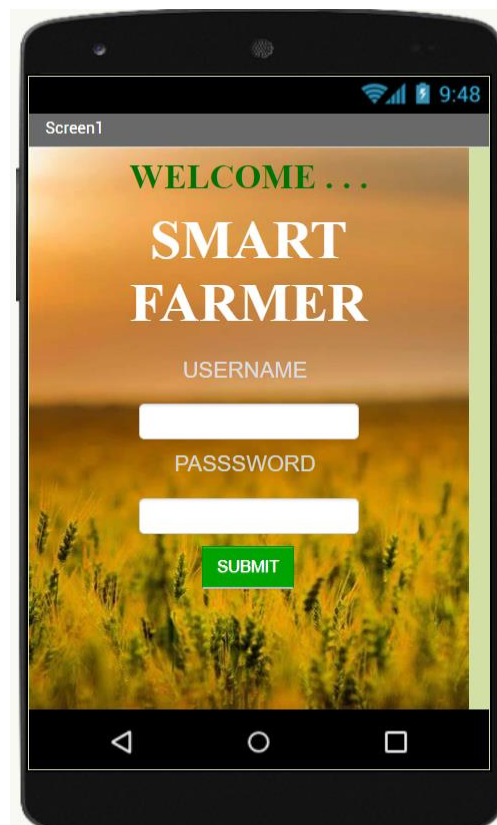
In order to display the parsed JSON data a Node-RED dashboard is created. Here we are using Gauges, text and button nodes to display in the UI and helps to monitor the parameters and control the farm equipment. Below images are the Gauge, text and button node configuration.

Program Flow



Mobile App





Program

```

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3wc7ia"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")
    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:

#Get Sensor Data from DHT11

temp=random.randint(90,110)

Humid=random.randint(60,100)

Mois=random.Randint(20,120)

data = { 'temp' : temp, 'Humid': Humid , 'Mois': Mois}

#print data

def myOnPublishCallback( ):

    print ("Published Temperature = %s C" % temp, "Humidity = %s %" % Humid,
    "Moisture =%s deg c" % Mois "to IBM Watson")

    success = deviceCli.publishEvent("IoTSensor", "json", data,
    qos=0,on_publish=myOnPublishCallback) if not success:

    print("Not connected to IoTTF")

    time.sleep(10)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

```

```

Pubsub.py - C:\Users\jeevi\OneDrive\Documents\Pubsub.py (3.7.4)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3wc7ia"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroft":
        print ("motor is off")
    else :
        print ("please send proper command")

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:
    #Get Sensor Data from DHT11
    temp=random.randint(90,110)
    Humid=random.randint(60,100)
    Mois=random.Randint(20,120)
    data = { 'temp' : temp, 'Humid': Humid , 'Mois': Mois}
    #Print data
    def myOnPublishCallback( ):
        print("Published Temperature = %s C" % temp, "Humidity = %s %" % Humid, "Moisture =%s deg c" % Mois, " to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTTF")
    time.sleep(10)
    deviceCli.commandCallback = myCommandCallback

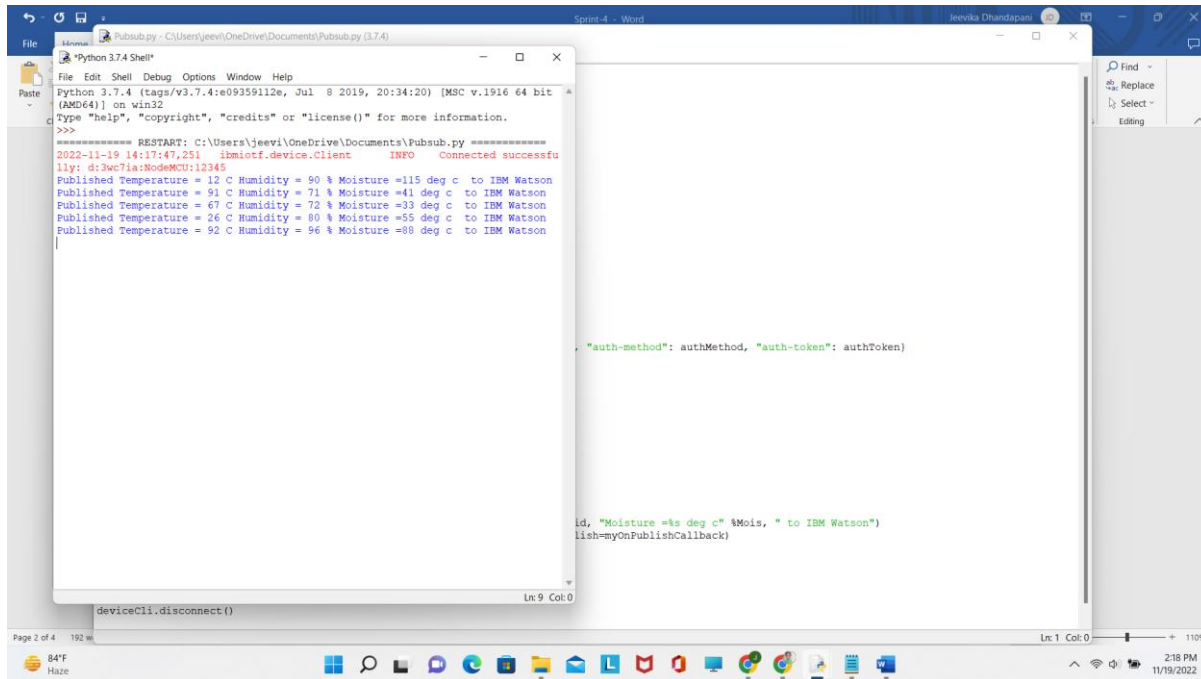
# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

84°F
Haze

Ln: 1 Col: 0
2:17 PM
11/19/2022

Result



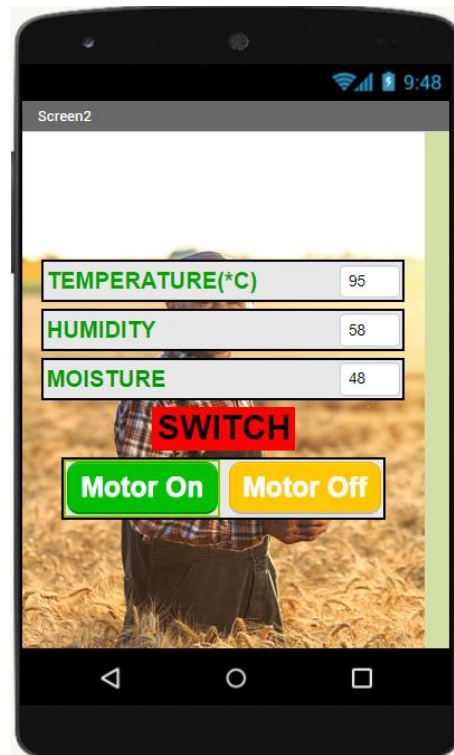
The screenshot shows a Windows desktop environment. In the foreground, a terminal window titled "Python 3.7.4 Shell" is open, displaying the execution of a Python script named "Pubsub.py". The script is running on a Windows 32-bit system. The output shows a successful connection to IBM Watson IoT Platform and the publication of sensor data (Temperature, Humidity, and Moisture) to the cloud. The data is published at regular intervals. In the background, a Microsoft Word document titled "Word" is open, showing a snippet of JSON data with "auth-method" and "auth-token" fields. The taskbar at the bottom shows the system clock as 2:18 PM on 11/19/2022, and the weather as 84°F and Hazy.

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:0935112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\jeevi\OneDrive\Documents\Pubsub.py =====
2022-11-19 14:17:47.251 - ibmiotf.device.Client INFO Connected successfully: d:3wc7ia:NodeMCU:12345
Published Temperature = 12 C Humidity = 90 % Moisture =115 deg c to IBM Watson
Published Temperature = 91 C Humidity = 71 % Moisture =41 deg c to IBM Watson
Published Temperature = 67 C Humidity = 72 % Moisture =33 deg c to IBM Watson
Published Temperature = 26 C Humidity = 80 % Moisture =55 deg c to IBM Watson
Published Temperature = 92 C Humidity = 96 % Moisture =88 deg c to IBM Watson

"auth-method": authMethod, "auth-token": authToken}

id, "Moisture =%s deg c" %Mois, " to IBM Watson")
lish=myOnPublishCallback)

deviceCli.disconnect()
Ln 9 Col 0
```



CHAPTER 8 - PERFORMANCE METRICS

S. No.	Name of the Phase	Tasks Performed	Performance Metrics
1.	Development of Problem Statement	The underlying problem analyzed and a rough idea of the solution was planned	The Problem statement was developed
2.	Ideation Phase	Extracting use and test cases	Empathy map, Ideation and Literature survey were formulated.
3.	Project Design Phase 1	Solution for the problem is formulated and architecture is designed	Problem solution fit was designed and the Proposed solution is finalized with the help of Solution architecture.
4.	Project Design Phase 2	In depth analysis of the solution is performed including requirements, tech stack, etc.	Solution Requirements, Overall Technology stack, Data flow diagrams, User stories were formulated.
5.	Project Planning Phase	Various sprints were designed as individual progressive steps.	Project Milestone and Sprint Plans were developed.

CHAPTER 9 - ADVANTAGES AND DISADVANTAGES

9.1 Advantages:

- 1) By monitoring the soil parameters of the farm, the user can have a complete analysis of the field, in terms of numbers. Using the website and the application, an interactive experience can be achieved.
- 2) As the data gets pushed to the cloud, one can access the data anywhere from this world.
- 3) Without human intervention, water pump can be controlled through the mobile application and it's flow can be customized using servo motors. o By using Raspberry Pi MCU, scalability can be increased due to its high processing power and enough availability of GPIO pins

9.2 Disadvantages:

- 1) Data transfer is through the internet. So data fetch and push might delay due to slow internet connection, depending on the location and other physical parameters.
- 2) System can only monitor a certain area of the field. In order to sense and monitor an entire field, sensors should be placed in many places, which may increase the cost.
- 3) Data accuracy may vary according to various physical parameters such as temperature, pressure, rain.
- 4) Cost of the system is high due to usage of Raspberry Pi.
- 5) Rodent and insects may cause damage to the system.

CHAPTER 10 – CONCLUSION

The project thus monitors important parameters present in the field such as temperature, humidity, soil moisture etc., and controls important actuators such as motors etc. It is helpful for farmers to remotely monitor their fields even during adverse weather conditions and help them control farming equipment remotely using cloud.

CHAPTER 11 - FUTURE SCOPE

The project can be further extended by monitoring other parameters such as nutrient contents in the soil, soil texture etc. AI techniques integrated with cloud can be integrated to monitor any pest attacks present in the plant. The application can be made interactive which provides suggestions to farmers to improve their farmlands. 43

CHAPTER 12 – APPENDIX 12.1

Source Code:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "3wc7ia"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")
    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:
        #Get Sensor Data from DHT11
        temp=random.randint(90,110)
        Humid=random.randint(60,100)
        Mois=random.Randint(20,120)
        data = { 'temp' : temp, 'Humid': Humid , 'Mois': Mois}
        #print data
        def myOnPublishCallback( ):
            print ("Published Temperature = %s C" % temp, "Humidity = %s %% " %Humid,
                "Moisture =%s deg c" % Mois "to IBM Watson")
            success = deviceCli.publishEvent("IoTSensor", "json", data,
                qos=0,on_publish=myOnPublishCallback) if not success:
                print("Not connected to IoTF")
            time.sleep(10)
        deviceCli.commandCallback = myCommandCallback
        # Disconnect the device and application from the cloud
        deviceCli.disconnect()

```

```

Pubsub.py - C:\Users\jeevi\OneDrive\Documents\Pubsub.py (3.7.4)
File Edit Format Run Options Window Help

import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "3wc7ia"
deviceType = "NodeMCU"
deviceId = "i2345"
authMethod = "token"
authToken = "i2345678"
# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="motoron":
        print ("motor is on")
    elif status == "motoroff":
        print ("motor is off")
    else :
        print ("please send proper command")
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:
    #Get Sensor Data from DHT11
    temper=random.randint(0,100)
    Humid=random.randint(60,100)
    Mois=random.randint(20,120)
    data = { 'temp': temp, 'Humid': Humid, 'Mois': Mois}
    #print data
    def myOnPublishCallback():
        print("Published Temperature = %s C" % temp, "Humidity = %s %% " %Humid, "Moisture =%s deg c" %Mois, " to IBM Watson")
        success = deviceCli.publishEvent("IotSensor", "json", data, qos=0,on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoT")
        time.sleep(10)
    deviceCli.commandcallback = myCommandCallback
    # Disconnect the device and application from the cloud
    deviceCli.disconnect()

```

Ln: 1 Col: 0
84°F Haze 2:17 PM 11/19/2022

MIT APP INVENTOR

Projects • Connect • Build • Settings • Help • My Projects View Trash Guide Report an Issue English • jeevika.dp@gmail.com •

Sample Screen2 Add Screen Remove Screens Publish to Gallery Designer Blocks

Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Dictionaries
 - Colors
 - Variables
 - Procedures
- Screen2
 - HorizontalArrangemen
 - Label1
 - TextBox1
 - HorizontalArrangemen
 - Label2
 - TextBox2

Viewer

when Clicked Timer

do

- set [Web1] URL to https://matfamerapp.firebaseio.com/
- call [Web1] Get

when Web1 Get

do

- responseCode
- if responseCode is 200
 - look up in pairs key
 - set [Web1] JSON to responseCode
 - call [Web1] JSON to responseCode
 - set [Web1] JSON to responseCode
 - call [Web1] JSON to responseCode
 - set [Web1] JSON to responseCode
 - call [Web1] JSON to responseCode
 - set [Web1] JSON to responseCode
 - call [Web1] JSON to responseCode
- if responseCode is not 200
 - set [Web1] JSON to responseCode
 - call [Web1] JSON to responseCode

when Button1 Click

do

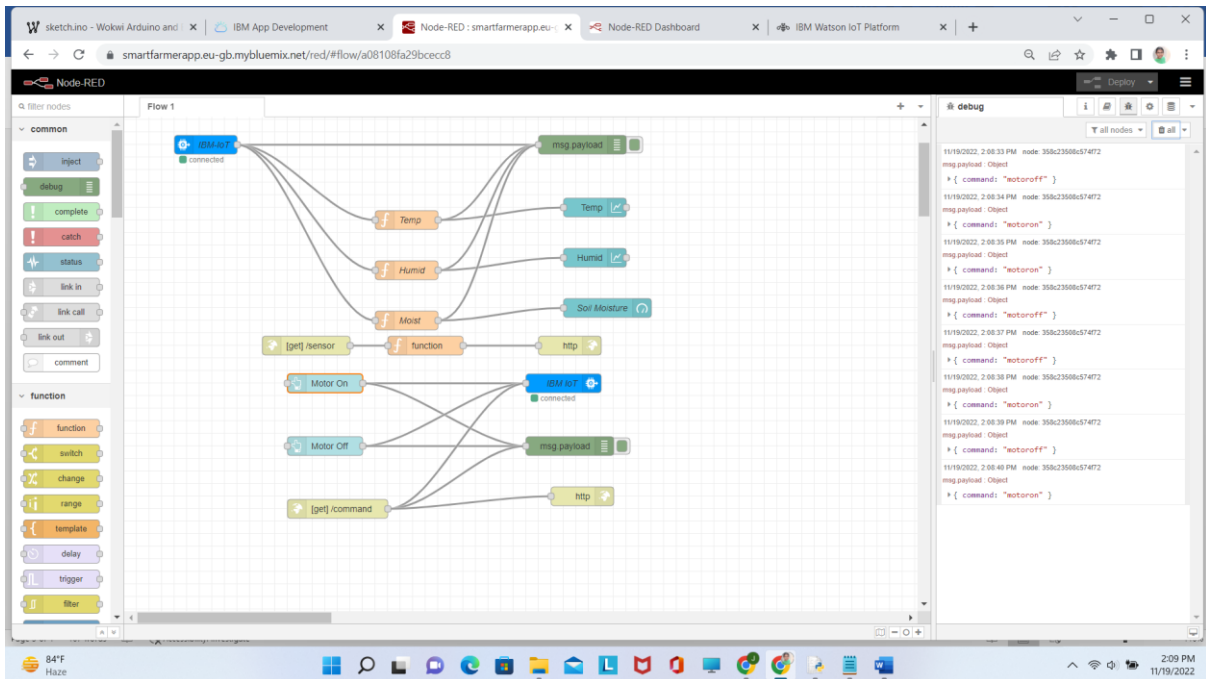
- set [Web2] URL to https://matfamerapp.firebaseio.com/
- call [Web2] Get
- set [Web2] JSON to responseCode
- call [Web2] JSON to responseCode

when Button2 Click

do

- set [Web3] URL to https://matfamerapp.firebaseio.com/
- call [Web3] Get

84°F Haze 2:10 PM 11/19/2022



Output

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\jeevi\OneDrive\Documents\Pubsub.py =====
2022-11-19 14:17:47.251 ibmiotf.device.Client INFO Connected successfully
lii: d13w71a:NodeMCU:12345
Published Temperature = 12 C Humidity = 90 % Moisture =115 deg c to IBM Watson
Published Temperature = 91 C Humidity = 71 % Moisture =41 deg c to IBM Watson
Published Temperature = 67 C Humidity = 72 % Moisture =33 deg c to IBM Watson
Published Temperature = 26 C Humidity = 80 % Moisture =55 deg c to IBM Watson
Published Temperature = 92 C Humidity = 96 % Moisture =88 deg c to IBM Watson
>>>
```

