SMART FARMER - IOT ENABLED SMART FARMING APPLICATION

Project Report

Submitted by

Team ID: PNT2022TMID19524

ABINATH M (722819106002)

GURU PRASATH I (722819106028)

JUGASRI K (722819106035)

KAILASH S (722819106036)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI ESHWAR COLLEGE OF ENGINEERING

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Kondampatti (post), Vadasithur (via), Kinathukadavu, Coimbatore – 641 202, Tamil Nadu.

TABLE OF CONTENTS

PROJECT REPORT

CHAPTER 1 - INTRODUCTION

1.1 PROJECT OVERVIEW

Agriculture has always been the backbone of any economic development. To promote further growth of agriculture, it must be integrated with modern practices and technologies. With the wide spread acceptance of technology, it can be used in farming to make farmers perform their activity with ease. Electronics and IoT has found its application in many of the personal assistant devices. This can be extended to many vital fields like agriculture where their assistants can help solve many issues faced. Electronics can help devices get physically connected with their operational environment and analyze and collect data. IoT can help analyze and transfer the data to the user. The combination of these gives rise to an all-in-one device capable of carrying out a task.

1.2 Purpose

In recent times, the erratic weather and climatic changes have caused issues for farmers in predicting the perfect conditions to initiate farming. Though on a superficial scale it seems unpredictable, it can be determined with certain parameters with which crop planning can be done. Maintenance of farm fields during and after cultivation are also important. These can be performed by measuring soil moisture, humidity and temperature.

Measurement of these parameters are performed using physical sensors. This system is in turn connected to IoT system which can provide a easy to access interface for farmers to read, analyze and take action based on the presented condition. Taking it a step ahead, the system can also gain access to motors and other electrical equipment used in farming and automate their operation. This can help with unsupervised operation ensuring accuracy and lesser response time.

CHAPTER 2 - LITERATURE SURVEY

2.1 EXISTING PROBLEM:

There has been several attempts and solution to help farmers adopt technological practices. Few solutions restricted their performance with just suggestions and alerts. While few employed IoT independent electronics. Few of the cases of previous attempts and researches are described below.

i. “IoT based smart sensors agriculture stick for live temperature and moisture monitoring using Arduino, cloud computing & solar technology”. This work was performed using Cloud computing platform (Things Speak) for data acquisition. The circuit was designed using Arduino and DHT 11 sensors.

ii. “Smart Farming using IoT, a solution for optimally monitoring farming conditions”. This work used ESP-32 based IoT platform and Blynk mobile application.

iii. “Smart farming using IoT”. The automation and interface part made use of water pump and HTTP protocol for parameters monitoring using website.

The above stated prior works lacked one or two features, which when included could have enhanced the performance. In the first work, including a Raspberry Pi based controller in place of Arduino can help reduce the design area while also providing microcontroller with additional UI and IoT interfaces. In the second stated work, going with MIT app inventor instead of Blynk application can improve the possibility of feature expansion. Farmers or developers won’t need to go for a paid version of the app to include new features. In the third work, control of water pump can be enhanced with the use of servo-based water valves to direct and control the flow of water rather than using a bi-stated logic.

2.2 References

The following were the source of references:

[1] 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.

[2] https://www.sciencedirect.com/science/article/pii/S1877050919317168

[3] *“Smart farming: IoT based smart sensors agriculture stick for live 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, Punjab Er. Vikram Puri M.Tech(ECE) Student, G.N.D.U Regional Center, Ladewali Campus, Jalandhar

[4] *“Smart Farming using IoT, a solution for optimally monitoring farming conditions”,* Jash Doshi; Tirth kumar ; Patel Santosh kumar Bharati

[5] *“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.

2.3 PROBLEM STATEMENT DEFINITION

The problem statement in a nutshell covers all the possible technical aspects that can be included by farmer to convert farming in to smart and efficient farming. IoT enabled smart farming, on a wider perspective, concentrates on connecting all the independently operating sub-systems in farming automation into a single entity. IoT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors.

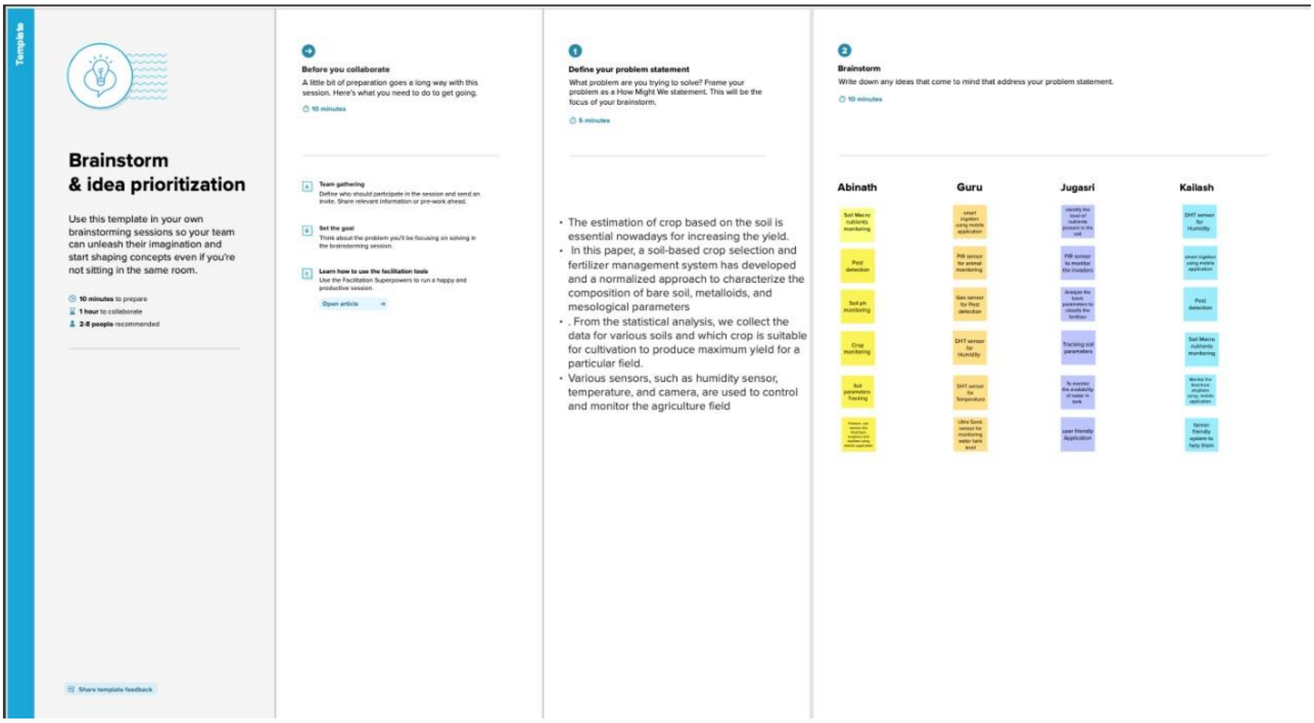
The idea of IoT is further extended with the help of mobile and web application where farmers can monitor all the sensor parameters even if the farmer is not near his field. Watering the crop is one of the important tasks for the farmers. They can make the decision whether to water the crop or postpone it by monitoring the sensor parameters and controlling the motor pumps from the mobile application itself.

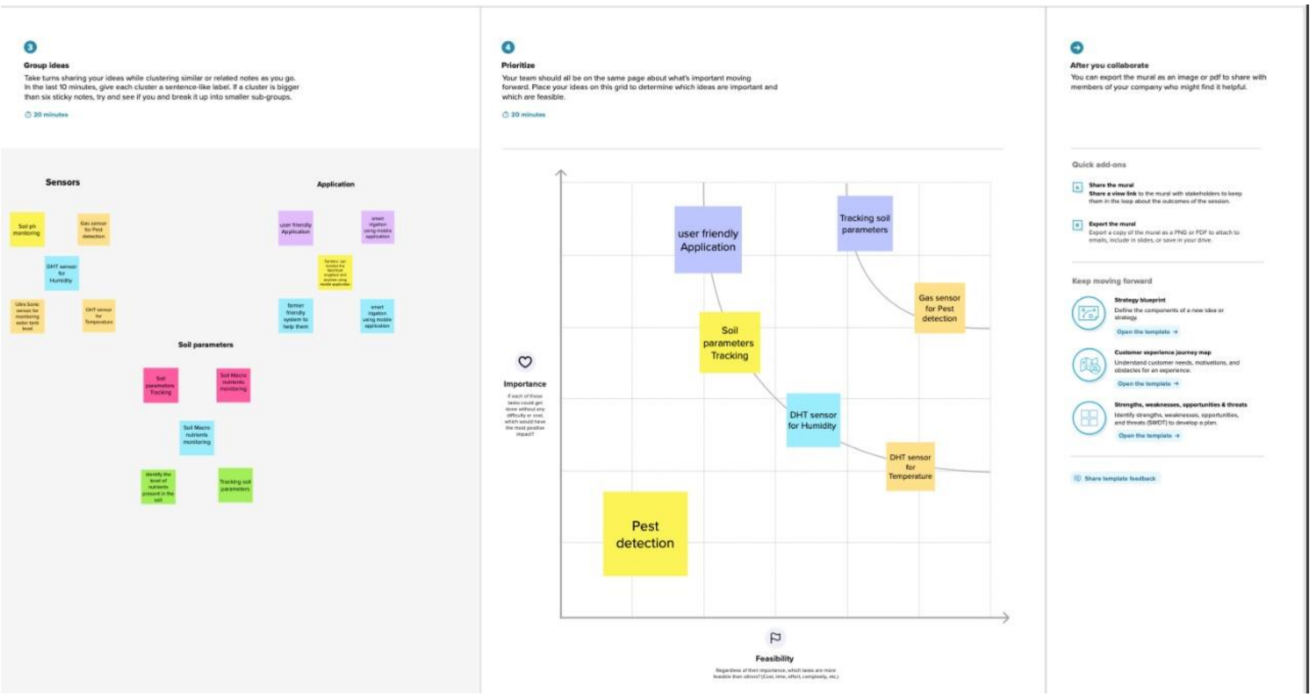
CHAPTER 3 - IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP:

An empathy map is a collaborative tool team can use to gain a deeper insight into their customers.

3.2 IDEATION AND PROPOSED SOLUTION:

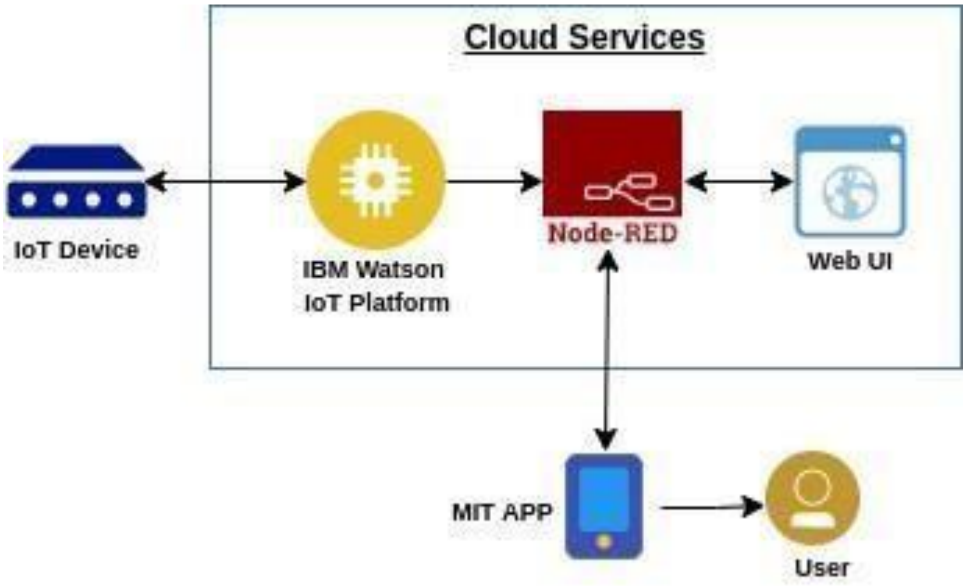




3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Soil parameters indicate the **state of soil ecosystem characteristics**, which especially reflect production, buffering, filter and other soil functions. • Nowadays inadequate of water and outdated irrigation techniques leads to affecting the production. • **Crop damage** caused by wildlife is a serious **problem , e**lectric fencing is used in many agricultural areas.
2.	Idea / Solution description	• The system checks the soil parameters and automates irrigation system and captures real time images using camera interfaced to microcontroller. • The system also consists of an android application which allows the user to give input based on which the watering will be controlled. • The system also senses the invasion of animals and sends notification to farmer.
3.	Novelty / Uniqueness	• **Soil fertile is identified by analyzing the availability of the soil nutrients level to grow suitable crop in that particular soil a**nd also it is useful to find the deficiency of soil nutrients. • The soil NPK sensor is suitable for detecting the content of nitrogen, phosphorus, and potassium in the soil, and judging the fertility of the soil.

4.	Social Impact / Customer Satisfaction	• By identifying the nutrient level in the soil and recommending the fertilizer at the earlier stage helps in improvement of production and the quality. • Which helps farmers and prevention of plant loss.
5.	Business Model (Revenue Model)	• Helps the farmers to make good production of food products and to reduce the production loss at an earlier stage. • With the proposed system crop yield, crop efficiency, agricultural product output will be increased. • A high gain can be seen in agricultural output and profit will be increased.
6.	Scalability of the Solution	• Business to business and business to customer can be implemented and it can be used for enhancing the profit in large scale.



3.4 PROBLEM SOLUTION FIT:

Problem-Solution canvas is a tool for entrepreneurs, marketers and corporate innovators, which help them, identify solutions with higher chances for a solution adoption, reduce time spent on solution testing and get a better overview of current situation.

1. **Customers segment**:

Farmers Large land owners,

Gardeners, Government

2. Customer constraints:

First of all, they should have land, they have to install the sensors on their farm which are required LoRa devices and LoRa WAN will be needed to receive the data from their farm to their mobile application

3.Available solutions:

- Crop monitoring
- Local weather monitoring
- Soil quality monitoring
- Irrigation control

4.Problems:

- Improper irrigation
- Crop rotation
- Soil erosion
- Climate change

5. ROOT/ CAUSE

- Watering the crops in the correct amount and time
- Monitoring the weather

6.BEHAVIOUR:

- Have the good internet connection
- Check the sensors regularly
- Check the notification regularly

7.Triggers:

Farmers want to make their crops healthy, control them from anywhere, and want to reduce the wages of labors. They also want to increase their yield

8.Emotions: Before: Difficulty in predicting the climate and to monitor the crops from anywhere Difficulty in watering the crops

After: Farm can be monitored easily from anywhere.

9.Solution:

- Using Local weather API, we can monitor the weather conditions.
- By using LoRa device to monitor the status of the field, and climate
- By using dth11 sensors and PIR sensor to sense the condition of the field

10.Channels of Behavior:

- ONLINE:

To get the information from the farm to the mobile application

- OFFLINE:

Checking the sensors regularly

CHAPTER 4 - REQUIREMENT ANALYSIS:

4.1 Functional Requirements:

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

- ***Sensors***: Rain sensor, DHT11–Temperature and Humidity sensor, Soil Moisture sensor.
- ***Actuators***: Water pumps, Motors, Servos.
- ***MCUs (Microcontroller Units)***: Raspberry Pi, ESP-8266.
- ***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:

• **Usability:** Can be used for both large scale agricultural farms and domestic gardens for soil monitoring and watering of plants.

• **Security:** Since the user uses his/her own cloud account to store and process sensor data, data privacy is maintained to a significant extent.

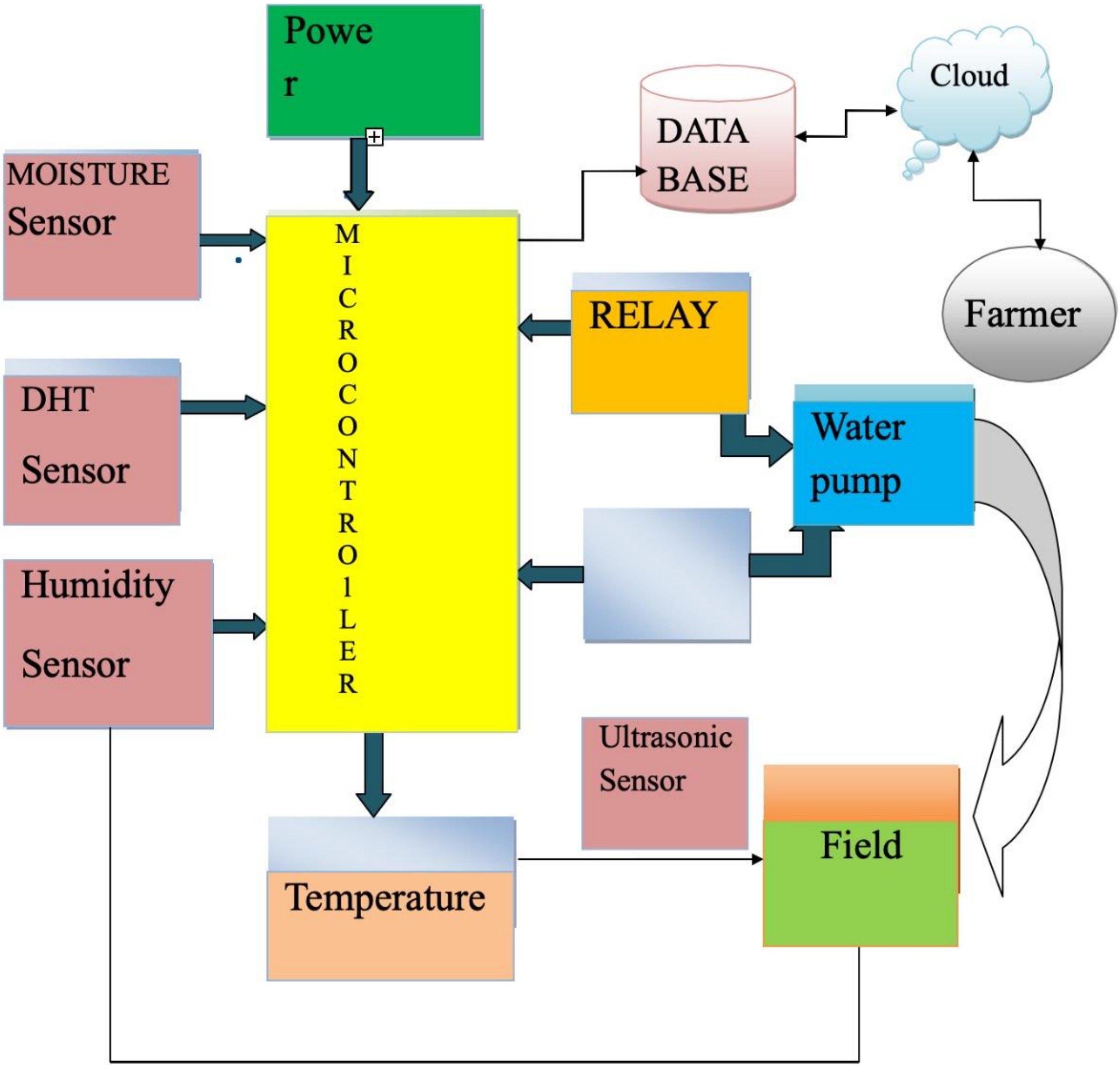
• **Reliability:** Inclusion of real-time monitoring of sensor data and interactive mobile application makes the product more reliable.11 Page 16 of 46 • Performance: Performance of the system is significantly high as MCUs with high processing capability such as Raspberry Pi are being used.

• **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.

• **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.

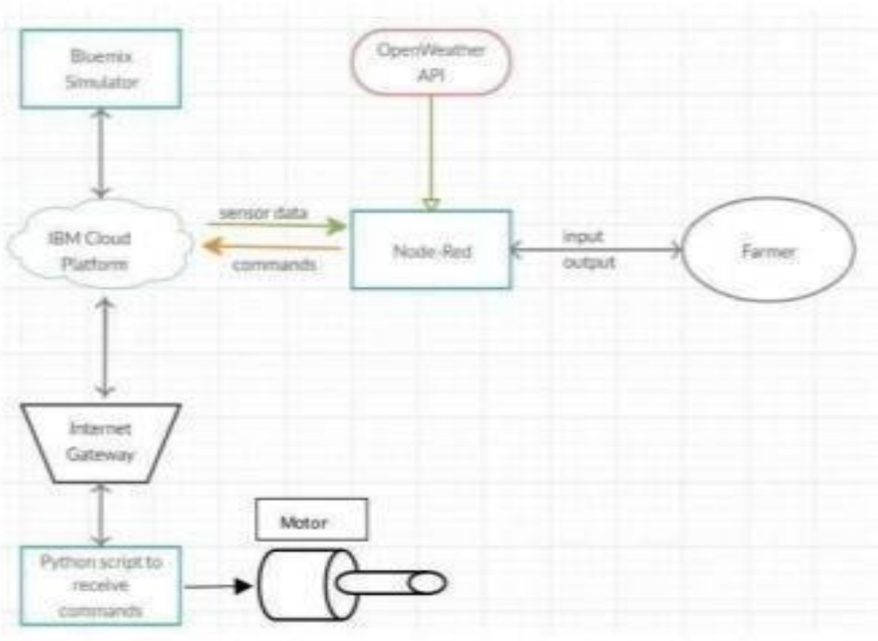
CHAPTER 5 - PROJECT DESIGN

Project design is an early phase of the project lifecycle where ideas, processes, resources, and deliverables are planned out. A project design comes before a project plan as it's a broad overview whereas a project plan includes more detailed information. A project design is the process of outlining all of a project's stages and creating a project plan. It includes a strategy of ideas, resources and processes to achieve project goals and keep within a budget and deadline. Project managers could add flowcharts, sketches, photo impressions and prototypes to help fully outline the project. Project managers present the project plan to senior stakeholders and investors to get final approval before beginning the project. In many cases, project managers create more than one pan for each project so stakeholders can choose which one they think would work best for the project.



5.2 Solution and Technical Architecture

The technical architecture diagram is as follows:



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 Page 19 of 46 depending upon the sensor values. By using the app, they can remotely operate the motor switch.

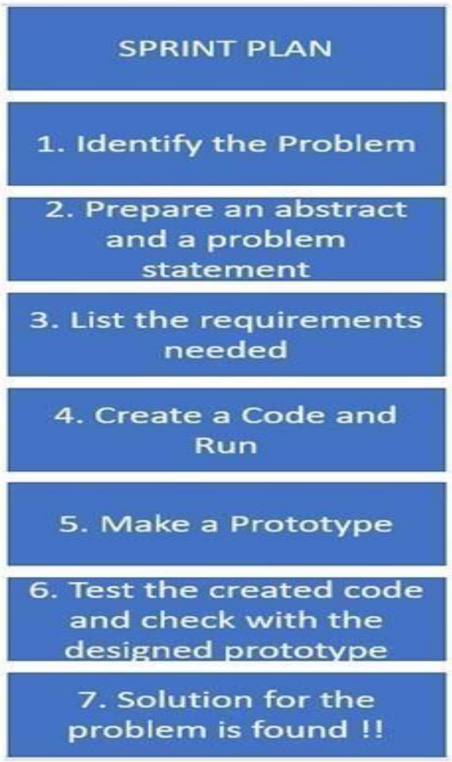
3.5 USER STORIES:

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer. The purpose of a user story is to articulate how a piece of work will deliver a particular value back to the customer.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-4	As a user, I can have access to all the sensor data on my dashboard		Low	Sprint-2
	Control	USN-5	As a user, I can control the agricultural devices connected over internet	I can control using user buttons	Low	Sprint-2
Customer (App user)	App - Control	USN-6	As a user, I can control the agricultural devices connected over internet	I can control using user buttons	Medium	Sprint-2
	App – monitor	USN-7	As a user, I can have access to all the sensor data on my app - dashboard		Medium	Sprint-2
Administrator	Setting defaults	USN-8	As a administrator, I can set default conditions to trigger an event		High	Sprint-1

CHAPTER 6 - PROJECT PLANNING AND SCHEDULING

6.1SPRINT PLANNING AND ESTIMATION



6.2 SPRINT DELIVERY AND SCHEDULE

The Sprint schedule is as follows:

Sprint	Functional Requirement (Epic)	User Story / Task	User Story Number	Story Points	Priority	Team Members
Sprint-1	Simulation creation	Connect Sensors and Arduino with python code	USN-1	2	High	Abinath,Kailash
Sprint-2	Software	Creating device in the IBM Watson IoT platform, workflow for IoT scenarios using Node-Red	USN-2	2	High	Abinath,Kailash, Guru Prasath
Sprint-3	MIT App Inventor	Develop an application for the Smart farmer project using MIT App Inventor	USN-3	2	High	Abinath,Kailash, Jugasri
Sprint-3	Dashboard	Design the Modules and test the app	USN-3	2	High	Abinath, Kailash
Sprint-4	Web UI	To make the user to interact with software.	USN-4	2	High	Jugasri, Guru Prasath

Project Tracker, Burndown chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Oct 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		15 Oct 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

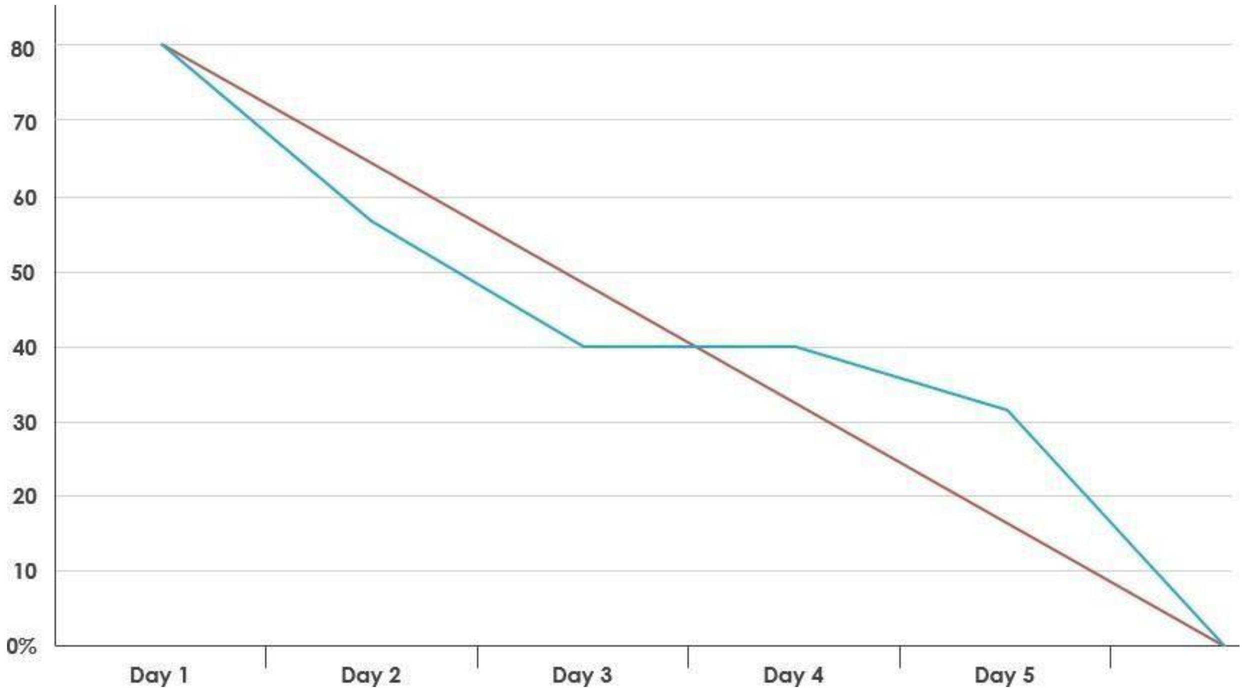
AV for sprint 1= Sprint Duration /velocity =12/6=2

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 SOLUTIONING

- **Connect Sensors and Arduino with python code**

```
#include <Servo.h>

Servo s; int Sensor = 0;

int data = 0;

int motorPin = 9;


void setup()

{

Serial.begin(9600);
pinMode(A0,INPUT);
//Temperature      pinMode(A1,INPUT);
Sensor

//Soil             pinMode(10,OUTPUT);
Moisture
Sensor

//GREEN            pinMode(11,OUTPUT);
light for LED

//BLUE light       pinMode(12,OUTPUT);
for LED

//RED light        s.attach(3);
for LED

//Servo Motor

pinMode(motorPin, OUTPUT); //DC motor

}

void loop(){
```

```
Sensor = analogRead(A1); //Reads data from Soil Moisture sensor
data = map(Sensor,0, 1023, 0, 100); //Low analog value indicates HIGH moisture level and High analog value indicates LOW moisture level

//data = map(analogValue,fromLOW,fromHIGH,toLOW,toHIGH)

Serial.print("Soil Moisture value:");

Serial.println(data);

//'data = 0' indicates wet and 'data = 100' indicates dry


double a = analogRead (A0); //Reads data from Temperature sensor

double t = (((a/1024)*5)-0.5)*100;

Serial.print("Temperature value:");

Serial.println(t);


if (t>40 & t<50)

{

digitalWrite(10,0); digitalWrite(11,1);

digitalWrite(12,0);

s.write(90);

digitalWrite(motorPin,  HIGH);

Serial.println("Water Partially Flows");

}

else if (t>50)

{

digitalWrite(10,0); digitalWrite(11,0);

digitalWrite(12,1);

s.write(180); digitalWrite(motorPin, HIGH);

Serial.println("Water Fully Flows");
```

```
}
else if (t>30 & data<30)
{

digitalWrite(10,1); digitalWrite(11,1);

digitalWrite(12,0);

s.write(90); digitalWrite(motorPin,

HIGH);

Serial.println("Water Partially Flows");

}

else if (data<50)

{

digitalWrite(10,0); digitalWrite(11,1);

digitalWrite(12,1);

s.write(90);

digitalWrite(motorPin,  HIGH);

Serial.println("Water Partially Flows");

}
else

{

digitalWrite(10,1); digitalWrite(11,0);

digitalWrite(12,0);


s.write(0);

digitalWrite(motorPin,     LOW);

Serial.println(" "); delay(1000);

}

}
```

PROGRAM

```
#include <Adafruit_LiquidCrystal.h> //Includes the library for LCD Display
#include              //Includes the library
<Wire.h>              for connections
#include              //Includes the library
<Servo.h>             for Servo Motor


Servo s; int e = 4; int t

= 5; int r = 12; int b =

11; int g = 10; int sec =

0; int Sensor = 0; int

soil = 0; int motorPin =

9;

Adafruit_LiquidCrystal lcd(0);

void setup()

{
Wire.begin();
pinMode(A0,              // Temperature
INPUT);                  Sensor
pinMode(A1,              // Soil Moisture
INPUT);                  Sensor
pinMode(t,               // Ultra sonic
OUTPUT);                 Trigger



pinMode(e, INPUT);       // Ultra sonic Echo
pinMode(b,               // GREEN light for
OUTPUT);                 LED
pinMode(g,               // BLUE light for
OUTPUT);                 LED
pinMode(r,               // RED light for
OUTPUT);                 LED
pinMode(motorPin, OUTPUT); // DC motor
```

```
s.attach(3); // Servo Motor lcd.begin(16,

2); // LCD 16x2 Display

lcd.setBacklight(0);

Serial.begin(9600);

}

float readDistanceCM()

{

digitalWrite(t, LOW);

delayMicroseconds(2);

digitalWrite(t, HIGH);

delayMicroseconds(10);

digitalWrite(t, LOW); int duration =

pulseIn(e, HIGH); return duration *

0.034 / 2;

}
void loop()
{

// Soil Moisture:

Sensor = analogRead(A1);   // Reads data from Soil Moisture sensor

soil = map(Sensor, 0, 1023, 0, 117);



// Low analog value indicates HIGH moisture level and High analog value indicates LOW
moisture level

// data = map(analogValue,fromLOW,fromHIGH,toLOW,toHIGH)

Serial.print("Soil Moisture value:");

Serial.println(soil);

//'data = 0' indicates total wetness and 'data = 100' indicates total dryness
```

```
// Temperature:

double a = analogRead(A0); // Reads data from Temperature sensor

double t = (((a / 1024) * 5) - 0.5) * 100;

Serial.print("Temperature value:"); //Temperature value in Celsius

Serial.println(t);



// Ultrasonic sensor:

float distance = readDistanceCM(); //Reads data from Ultrasonic sensor

Serial.print("Measured distance: ");

Serial.println(readDistanceCM());



// LCD Display:

lcd.setBacklight(1);  //ON the background light in LCD

lcd.clear();


// Conditions:


/*If the temperature is Greater than 20 and less than 35 and also the moisture of soil is less
than 60 then the
GREEN light will be turned ON indicating the Normal condition */

if (t >= 20 && t < 35 && soil >= 40 && soil < 50)

{

digitalWrite(b, 0);
```

```
digitalWrite(g, 1);

digitalWrite(r, 0);

s.write(90);


digitalWrite(motorPin, HIGH);

lcd.setCursor(3, 0); lcd.print("ON

MOTOR"); delay(1000);

lcd.clear();

Serial.println("Water Partially Flows");

}
```

/*If the temperature is Greater than 35 and less than 45, then the BLUE light will be turned ON indicating the
Intermediate risk condition due to slightly warm weather */

```
else if (t >= 35 && t < 45)

{

digitalWrite(b, 1);

digitalWrite(g, 0);

digitalWrite(r, 0);

s.write(90); digitalWrite(motorPin,

HIGH); lcd.setCursor(3, 0);

lcd.print("ON MOTOR");

delay(1000);

lcd.clear();
```

```
Serial.println("Water Partially Flows");

}
```

/*If the temperature is Greater than 45 or the moisture of soil is less than 30, then the RED light will be turned
ON indicating the Critical condition due to highly warm weather or the low moisture content in soil */

```
else if (t >= 45 || soil < 30)

{

digitalWrite(b, 0);

digitalWrite(g, 0);

digitalWrite(r, 1);

s.write(180); digitalWrite(motorPin,

HIGH); Serial.println("Water Fully

Flows"); lcd.setCursor(2, 0);

lcd.print("ON MOTOR!!!");

lcd.setCursor(3, 1); lcd.print("Low

Water"); delay(1000);

lcd.clear();

}
```

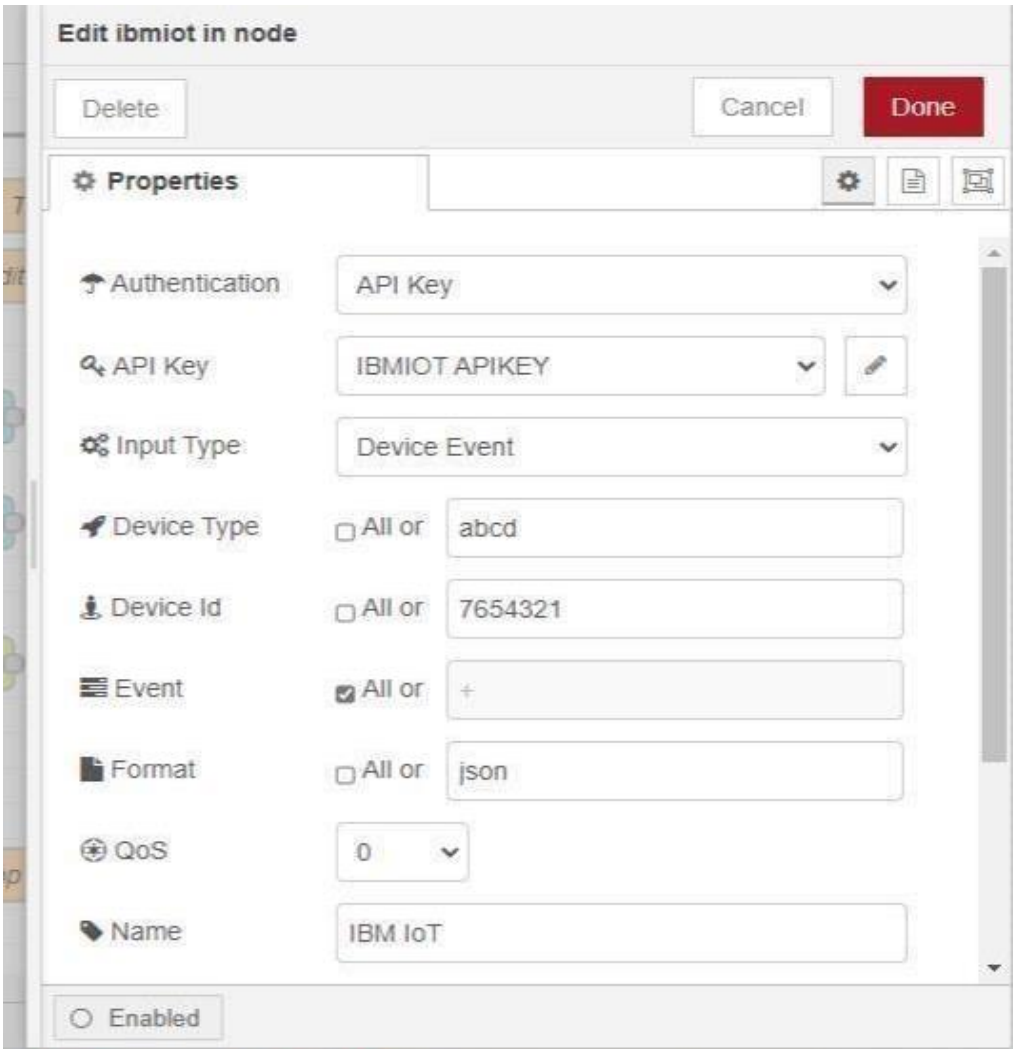
/*If the level of water is MORE in the field it will be indicated by distance sensor for less than 10cm and also the moisture of soil is greater than 80, then the

```
YELLOW light will be turned ON indicating the high water level */
else if (distance<10 && soil> 80)
{
digitalWrite(b, 0);
digitalWrite(g, 1);
digitalWrite(r, 1);
s.write(0); digitalWrite(motorPin, LOW);
Serial.println("Water Does Not Flow");
lcd.clear(); lcd.setCursor(3, 0);
lcd.print("OFF MOTOR"); delay(1000);
lcd.clear(); lcd.setCursor(1, 0);
lcd.print("DRAIN WATER!!!");
delay(1000); lcd.clear();

}

else
{
digitalWrite(b, 1);
digitalWrite(g, 1);
digitalWrite(r, 0);
s.write(0); digitalWrite(motorPin,
LOW); lcd.setCursor(3, 0);
```

```
lcd.print("OFF MOTOR");

delay(1000); lcd.clear();

Serial.println("Water Does Not Flow");

}



lcd.setCursor(0, 0);

lcd.print("Temp:"); lcd.print(t);

lcd.print("degree");

lcd.setCursor(0, 1);

lcd.print("SoilWetness:");

lcd.print(soil); lcd.print("%");



Serial.println("          ");

delay(1000); }
```

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

  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.



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

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===1)

msg.payload={"command”:

“ON"}; else

if(msg.payload===0)

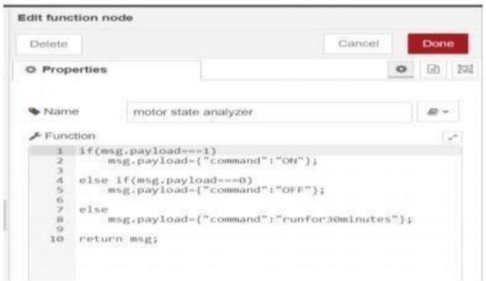
msg.payload={"command”:

“OFF"};

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

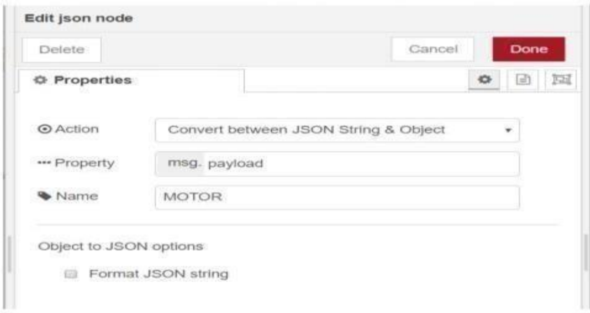
The Java script code for that function node is:

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

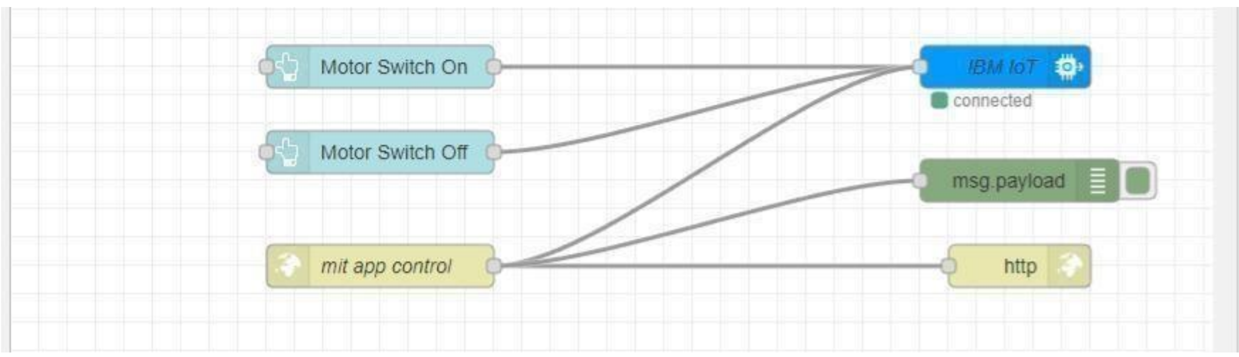


The above images show the java script codes of analyser and state function nodes.

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

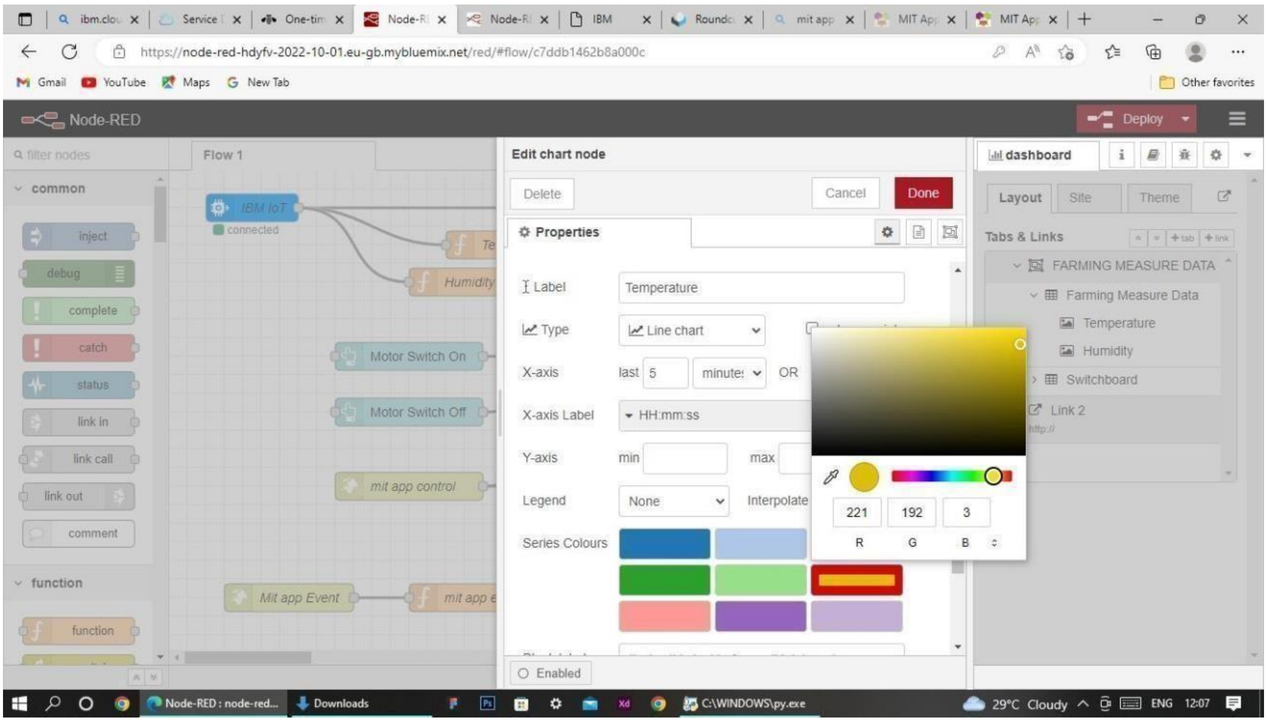


- ## Adjusting User Interface

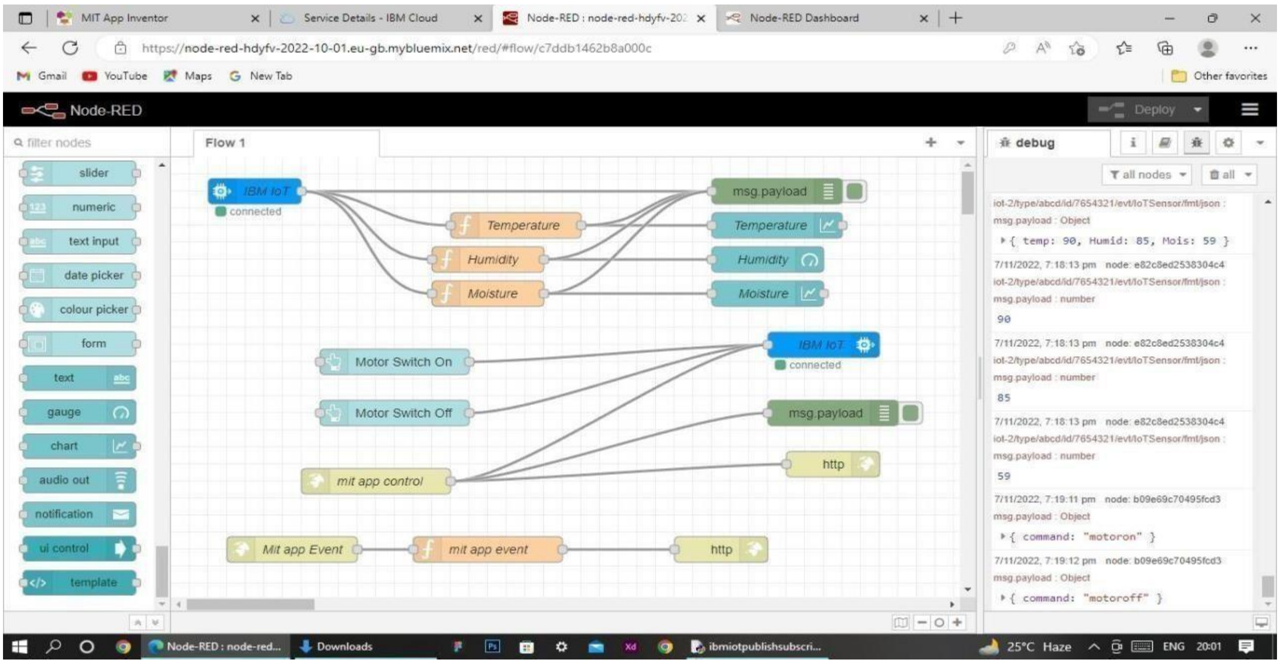
In order to display the parsed JSON data a Node-Red dashboard is created

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.

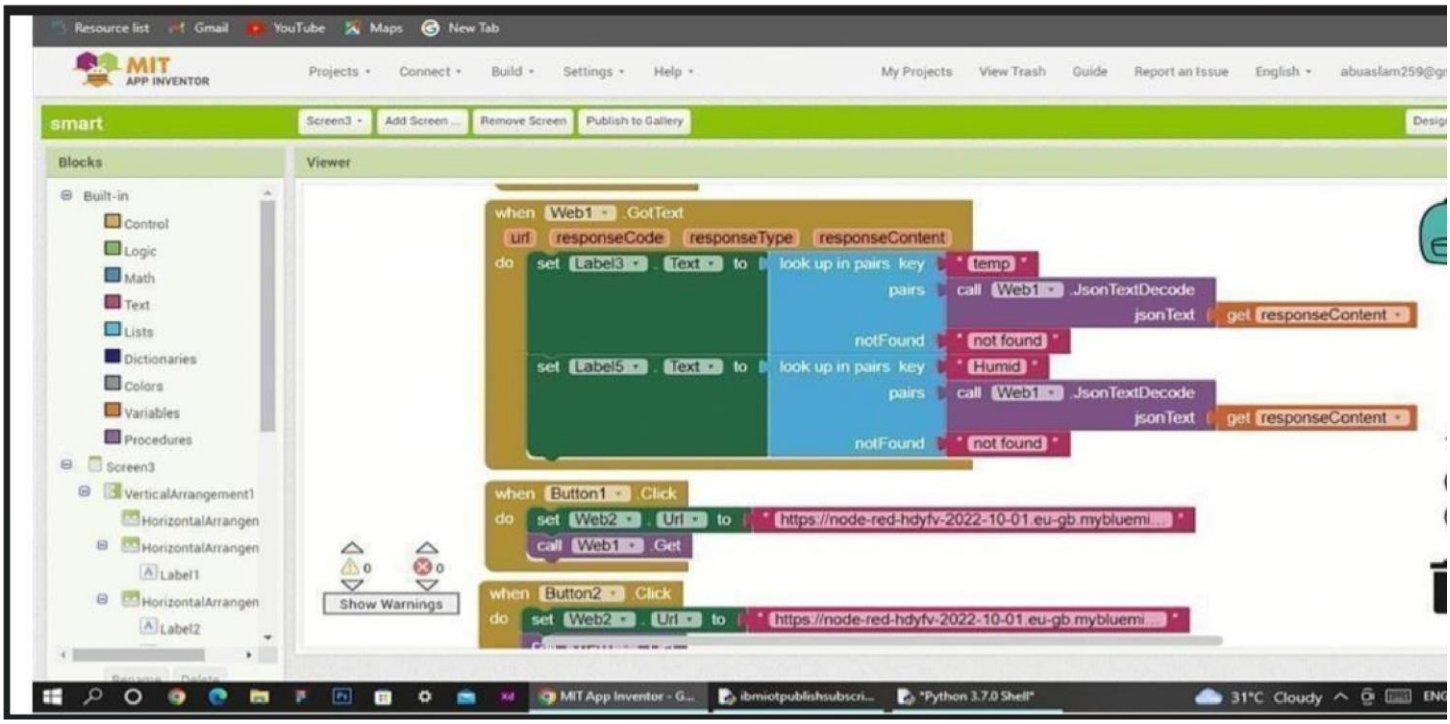
Below images are the Gauge, text and button node configurations.

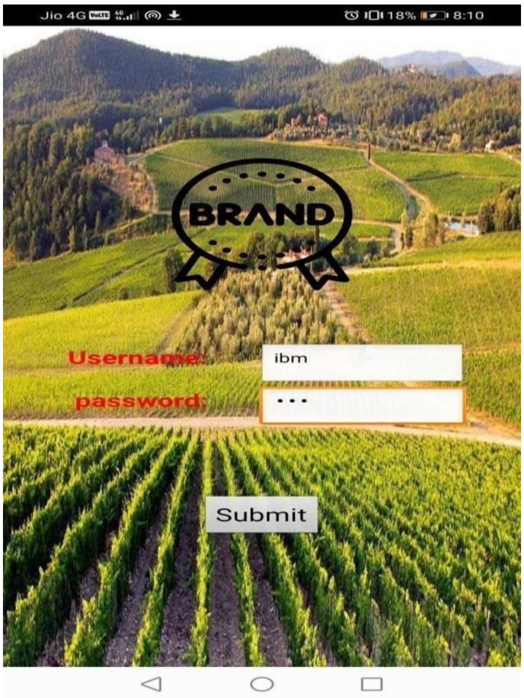


- **Complete Program Flow**



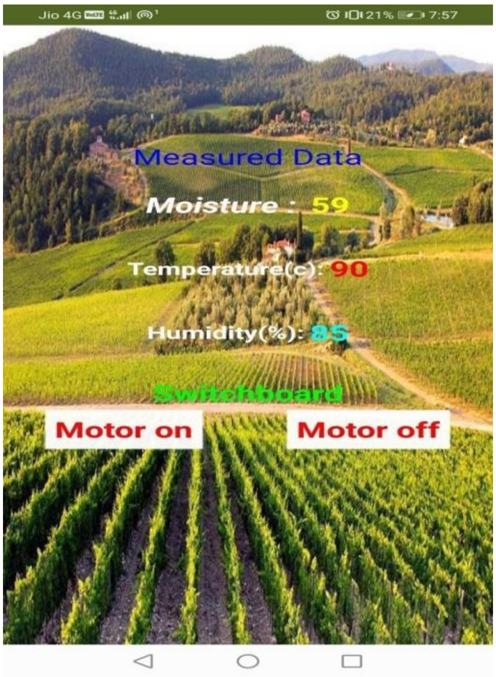
- **MOBILE APP WEB :**





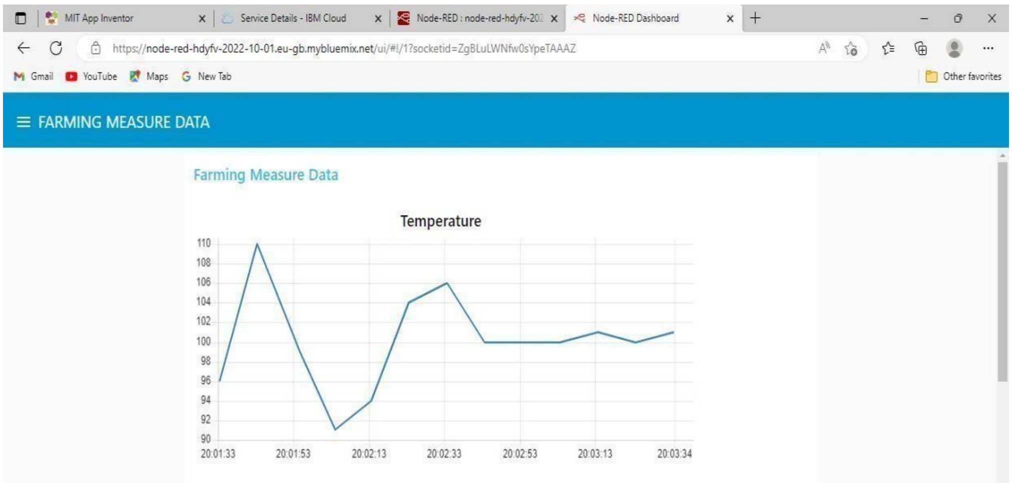
SCREEN – 1

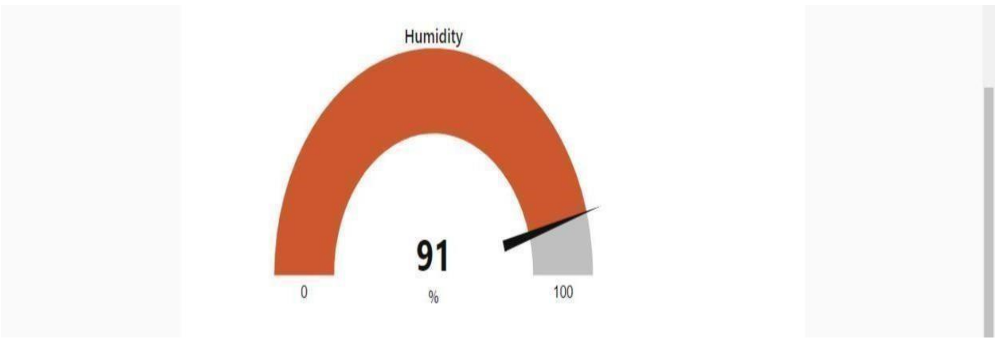
SCREEN - 2

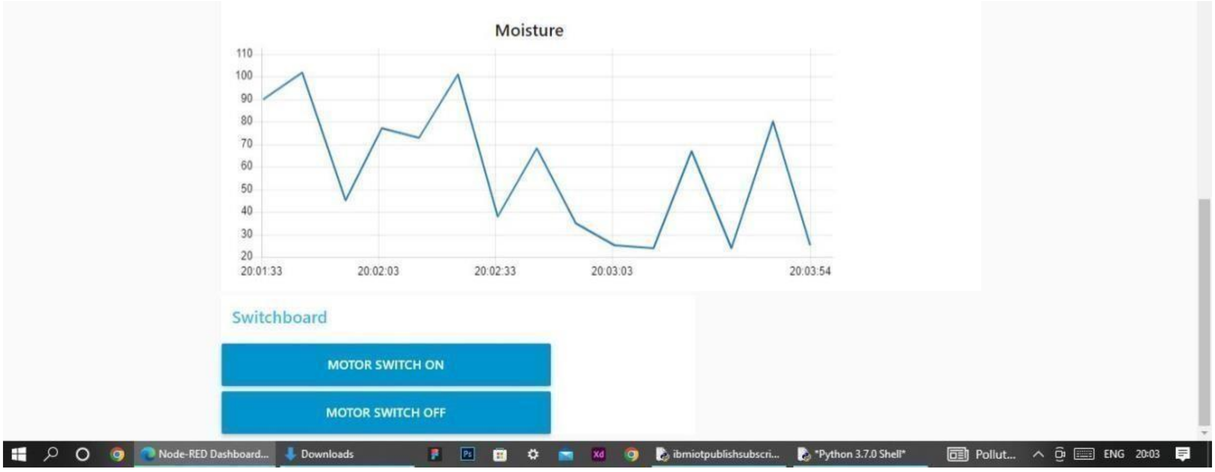


SCREEN - 3

- **Web APP UI Home Tab**







- **Receiving commands from IBM cloud using Python program**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device Credentials

organization = "157uf3" deviceType = "abcd" deviceId = "7654321" authMethod = "token" authToken = "87654321"

 # 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":

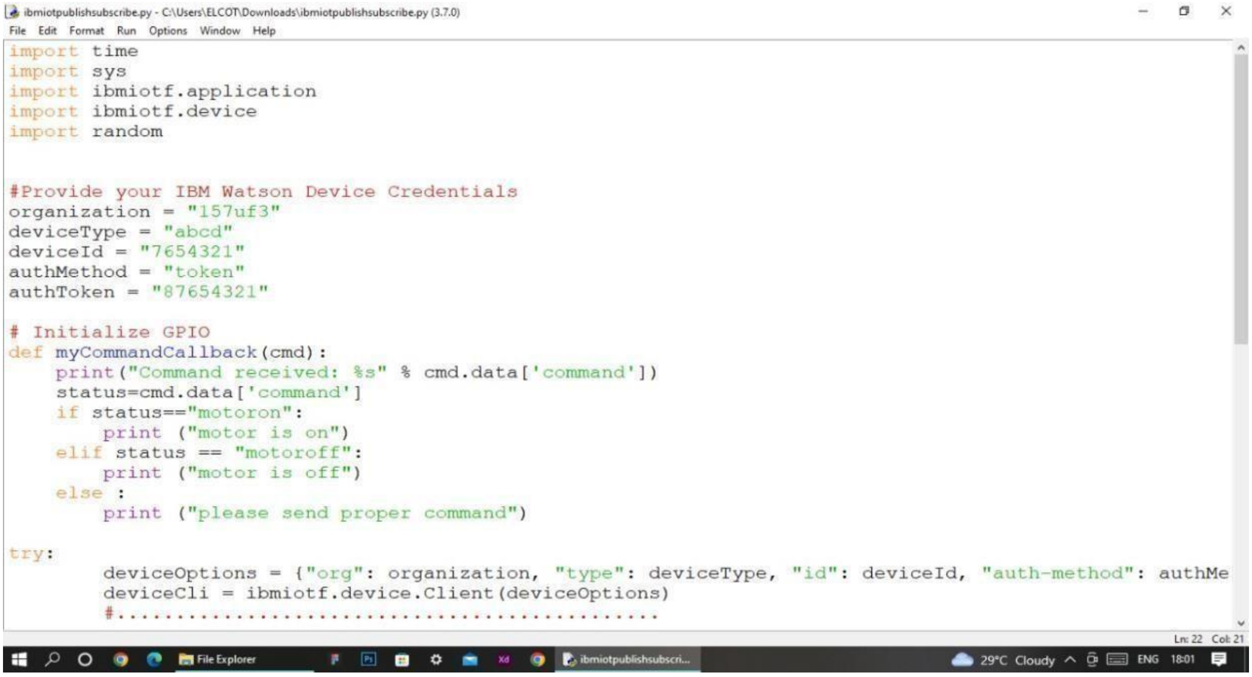
print ("motor is off")

else :

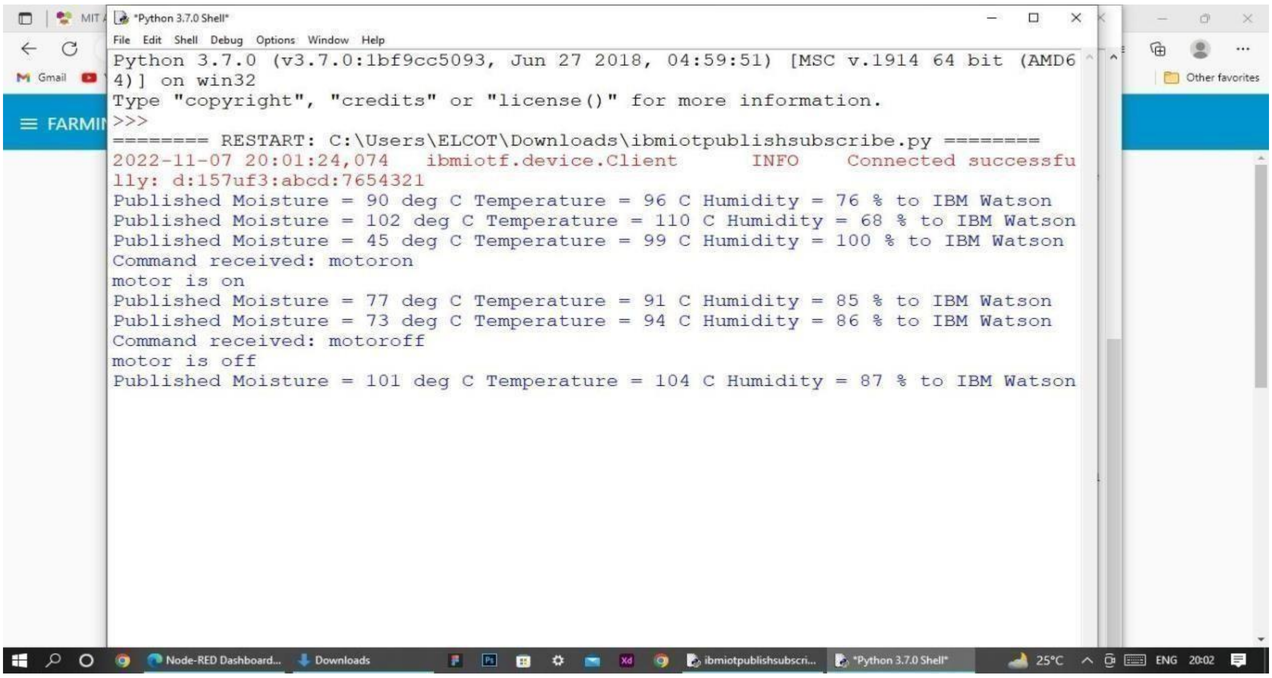
print ("please send proper command")

```
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()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of
type "greeting" 10 times 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()
```

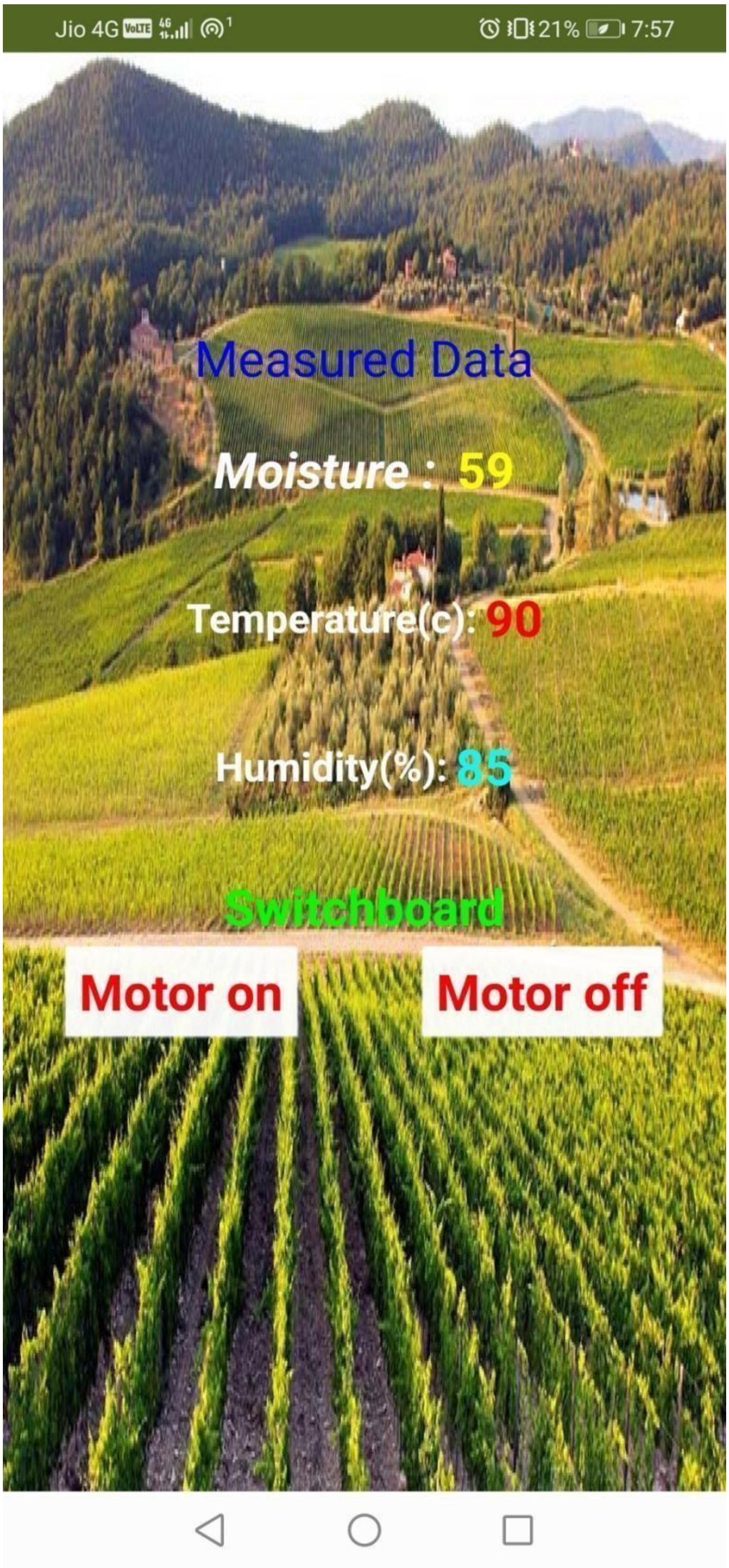
OUTPUT:



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

#Provide your IBM Watson Device Credentials
organization = "157uf3"
deviceType = "abcd"
deviceId = "7654321"
authMethod = "token"
authToken = "87654321"

# 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": authMe
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #..............................................
```



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\ELCOT\Downloads\ibmiotpublishsubscribe.py ========
2022-11-07 20:01:24,074   ibmiotf.device.Client      INFO    Connected successfu
lly: d:157uf3:abcd:7654321
Published Moisture = 90 deg C Temperature = 96 C Humidity = 76 % to IBM Watson
Published Moisture = 102 deg C Temperature = 110 C Humidity = 68 % to IBM Watson
Published Moisture = 45 deg C Temperature = 99 C Humidity = 100 % to IBM Watson
Command received: motoron
motor is on
Published Moisture = 77 deg C Temperature = 91 C Humidity = 85 % to IBM Watson
Published Moisture = 73 deg C Temperature = 94 C Humidity = 86 % to IBM Watson
Command received: motoroff
motor is off
Published Moisture = 101 deg C Temperature = 104 C Humidity = 87 % to IBM Watson
```


Jio 4G
21% 7:57
Measured Data
Moisture : 59
Temperature(c): 90
Humidity(%): 85
Switchboard
Motor on
Motor off

CHAPTER 9 - ADVANTAGES AND DISADVANTAGES

9.1 ADVANTAGES:

- 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.
- As the data gets pushed to the cloud, one can access the data anywhere from this world. ○ Without human intervention, water pump can be controlled through the mobile application and it's flow can be customized using servo motors.
- By using Raspberry Pi MCU, scalability can be increased due to its high processing power and enough availability of GPIO pins

9.2 DISADVANTAGES:

- 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.
- 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.
- Data accuracy may vary according to various physical parameters such as temperature, pressure, rain.
- Cost of the system is high due to usage of Raspberry Pi.
- 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 equipments 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.

CHAPTER 12 – APPENDIX

The Project deliverables are uploaded in Git repository and in the IBM dashboard.

GIT LINK

IBM-EPBL/IBM-Project-4438-1658732418: SmartFarmer - IoT Enabled Smart Farming Application (github.com)

DEMO LINK

https://www.youtube.com/watch?v=jsVZYjWybXs