SMART FARMER - IOT ENABLED SMART FARMING APPLICATION

NALAIYA THIRAN PROJECT

Submitted by

NAVEEN.R.S(TEAM LEAD)	***-921319106140***
NIRMAL.R	***-921319106147***
PRANESWAR.G	***-921319106155***
NAVEEN.M	***-921319106138***

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Team ID: PNT2022TMID05266

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:

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. https://www.sciencedirect.com/science/article/pii/S1877050919317168

iii. *"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 iv. *"Smart Farming using IoT, a solution for optimally monitoring farming conditions",* Jash Doshi; Tirth kumar ; Patel Santosh kumar Bharati.

v. *"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.

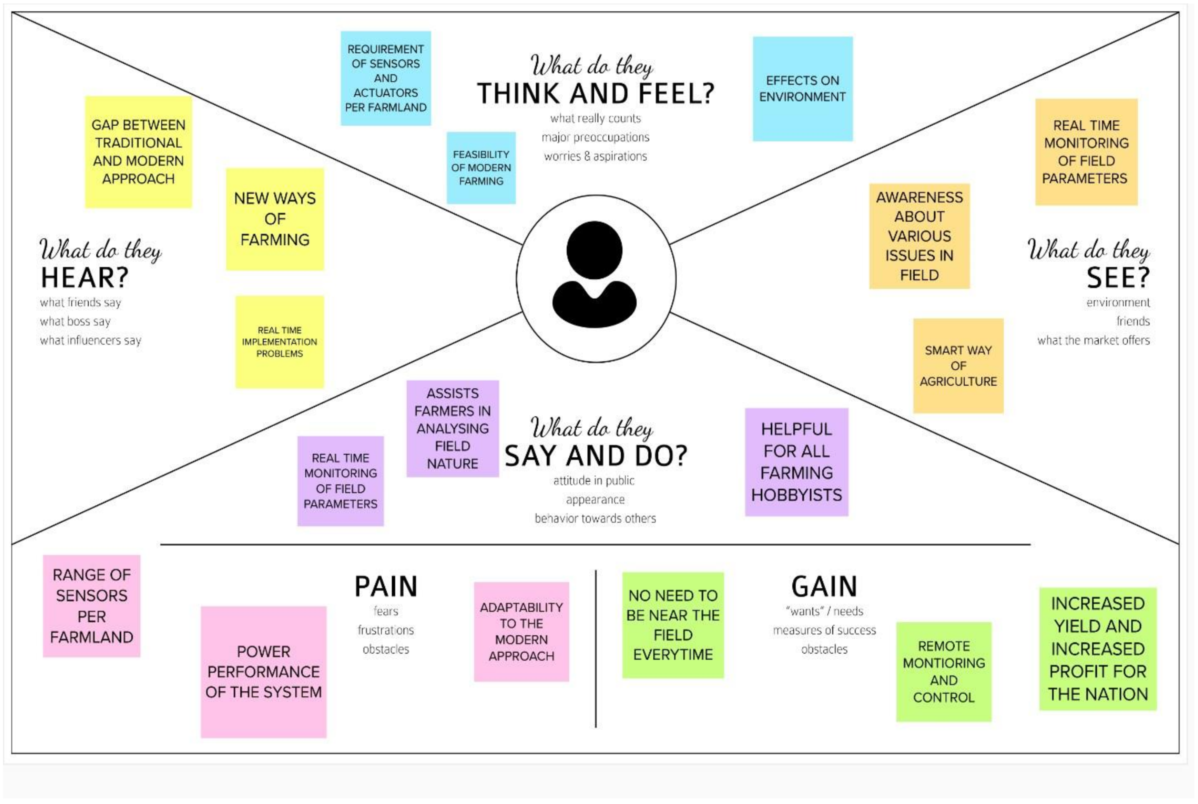
2.3 Problem Statement Definition

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 Canvas


REQUIREMENT OF SENSORS AND ACTUATORS PER FARMLAND
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
EFFECTS ON ENVIRONMENT
FEASIBILITY OF MODERN FARMING
GAP BETWEEN TRADITIONAL AND MODERN APPROACH
NEW WAYS OF FARMING
REAL TIME IMPLEMENTATION PROBLEMS
What do they HEAR?
what friends say
what boss say
what influencers say
REAL TIME MONITORING OF FIELD PARAMETERS
AWARENESS ABOUT VARIOUS ISSUES IN FIELD
SMART WAY OF AGRICULTURE
What do they SEE?
environment
friends
what the market offers
ASSISTS FARMERS IN ANALYSING FIELD NATURE
REAL TIME MONITORING OF FIELD PARAMETERS
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
HELPFUL FOR ALL FARMING HOBBYISTS
RANGE OF SENSORS PER FARMLAND
POWER PERFORMANCE OF THE SYSTEM
PAIN
fears
frustrations
obstacles
ADAPTABILITY TO THE MODERN APPROACH
NO NEED TO BE NEAR THE FIELD EVERYTIME
GAIN
"wants" / needs
measures of success
obstacles
REMOTE MONTIORING AND CONTROL
INCREASED YIELD AND INCREASED PROFIT FOR THE NATION

3.2 Ideation & Brainstorming

Top 3 ideas:

Top 1st:

Name: Naveen.R.S

Idea name: User friendly application for farmland parameters

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.

Top 2nd:

Name: Nirmal.R

Idea name: A website guide about the product and features

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.

Top 3rd:

Name: Praneswar.G

Idea name: usage of raspberry pi over other processor for IoT applications

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.

16 ideas from the brain storming session

Naveen.R.S

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

Nirmal.R

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.

Praneswar.

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

Naveen.M

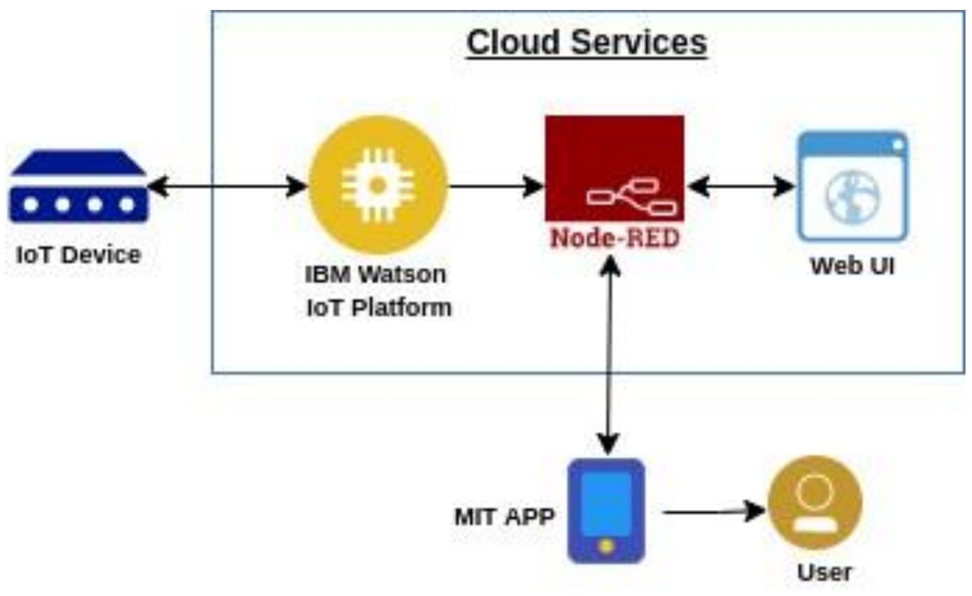
Idea 1:	Various protocols used in IoT.
Idea 2:	Application with simple UI but efficient usage.
Idea 3:	Interface between website, application and sensors.
Idea 4:	Utilizing ai to improvise accuracy.

3.3 Proposed Solution

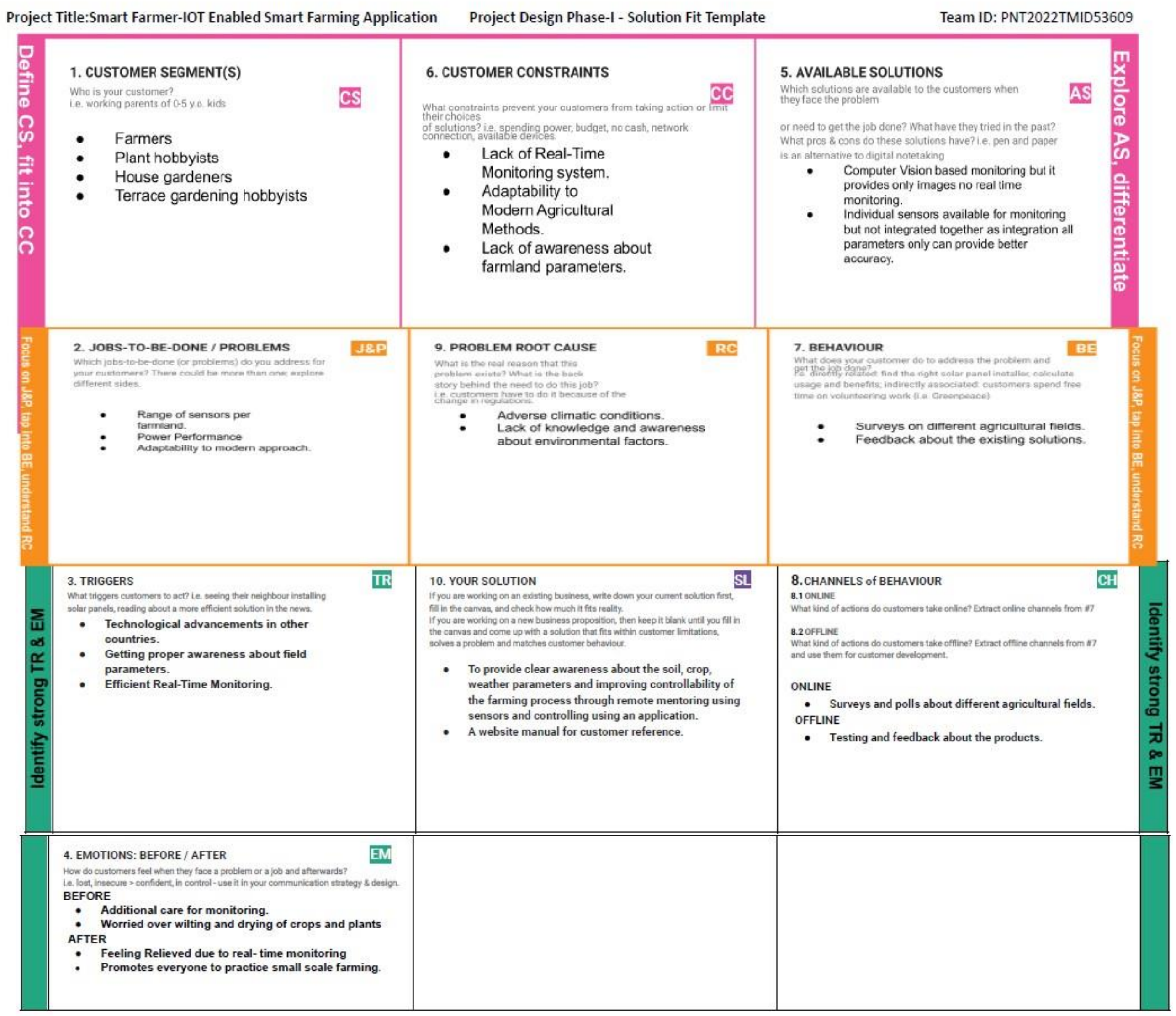
S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To overcome the lack of awareness and control of soil parameters such as temperature, moisture, humidity thereby improving quality of crops and plants, by implementing automated watering of plants.
	Idea / Solution description	To provide clear awareness about the

2.		soil, crop, weather parameters and improving controllability of the farming process through remote monitoring using sensors and controlling using an application.
3.	Novelty / Uniqueness	• Usage of Raspberry Pi for increased processing power and user-friendly programming using Python. • Usage of Servo motors to properly direct the water flow according to the needs.
4.	Social Impact / Customer Satisfaction	• Improvisation in control of farming which ultimately leads to increased number of plants and trees. • Effective reduction of greenhouse effect and global warming.
5.	Business Model (Revenue Model)	Based on the scale of application and the sophistication of the motors, the cost of the model will vary.
6.	Scalability of the Solution	□ Variables of the entire field can be monitored using a single cloud account.

 Small scale farming (gardening) applications for domestic purposes can use small scale model and large-scale farming can include more sophisticated sensors such as rain sensor, TDS sensor, etc.



3.4 Problem Solution fit



Project Title:Smart Farmer-IOT Enabled Smart Farming Application   Project Design Phase-I - Solution Fit Template   Team ID: PNT2022TMID53609

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

- Farmers
- Plant hobbyists
- House gardeners
- Terrace gardening hobbyists

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices
of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- Lack of Real-Time Monitoring system.
- Adaptability to Modern Agricultural Methods.
- Lack of awareness about farmland parameters.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem

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

- Computer Vision based monitoring but it provides only images no real time monitoring.
- Individual sensors available for monitoring but not integrated together as integration all parameters only can provide better accuracy.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

- Range of sensors per farmland.
- Power Performance
- Adaptability to modern approach.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists? What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- Adverse climatic conditions.
- Lack of knowledge and awareness about environmental factors.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- Surveys on different agricultural fields.
- Feedback about the existing solutions.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

- **Technological advancements in other countries.**
- **Getting proper awareness about field parameters.**
- **Efficient Real-Time Monitoring.**

10. YOUR SOLUTION SL

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.
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.

- **To provide clear awareness about the soil, crop, weather parameters and improving controllability of the farming process through remote mentoring using sensors and controlling using an application.**
- **A website manual for customer reference.**

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

ONLINE

- **Surveys and polls about different agricultural fields.**

OFFLINE

- **Testing and feedback about the products.**

Identify strong TR & EM

4. EMOTIONS: BEFORE / AFTER EM

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.

BEFORE

- **Additional care for monitoring.**
- **Worried over wilting and drying of crops and plants**

AFTER

- **Feeling Relieved due to real- time monitoring**
- **Promotes everyone to practice small scale farming.**

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:

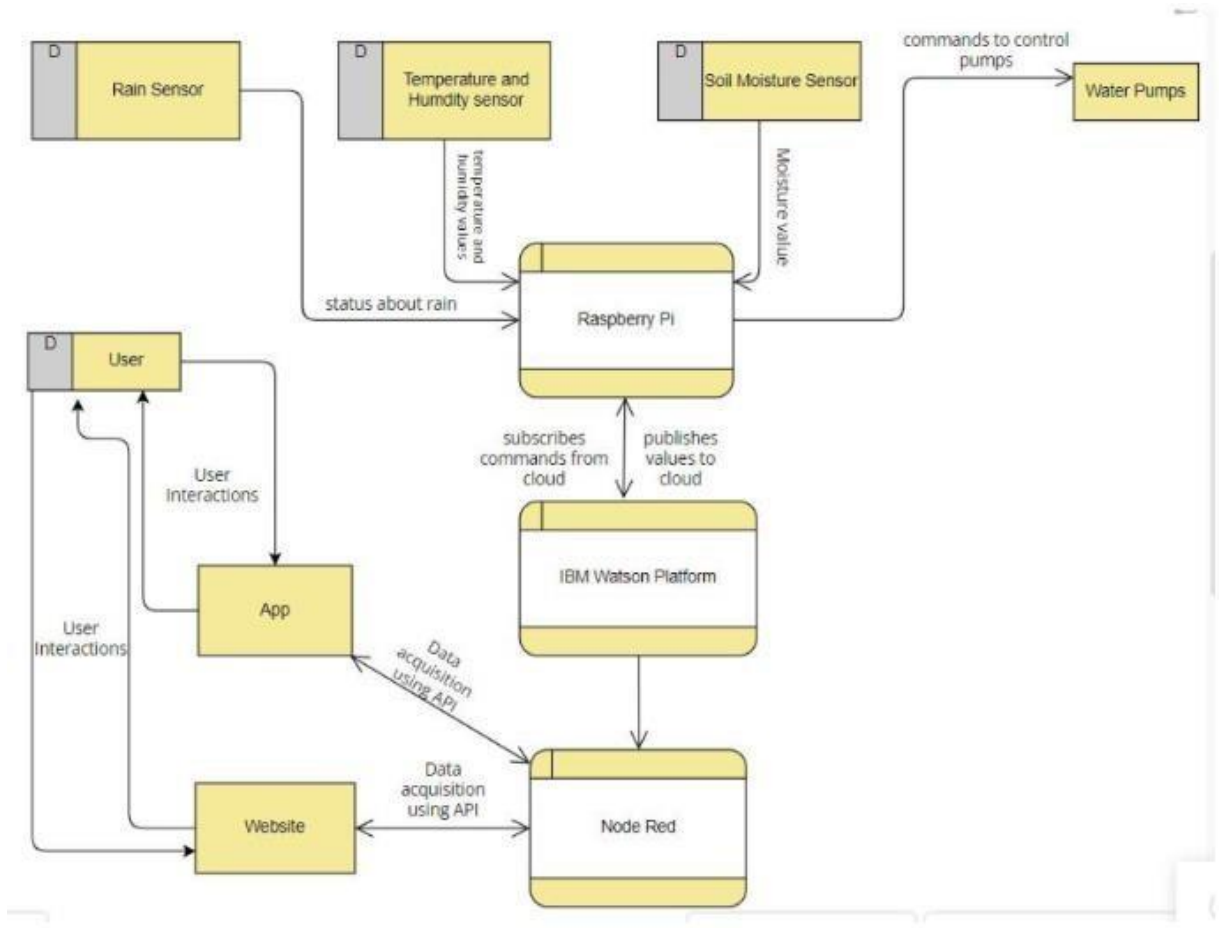
Non-functional requirements deal more of a customer or business model point of view. These requirements play a major role when the project is ready as a market product. Some of those non-technical requirements are:

- ***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.
- ***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.

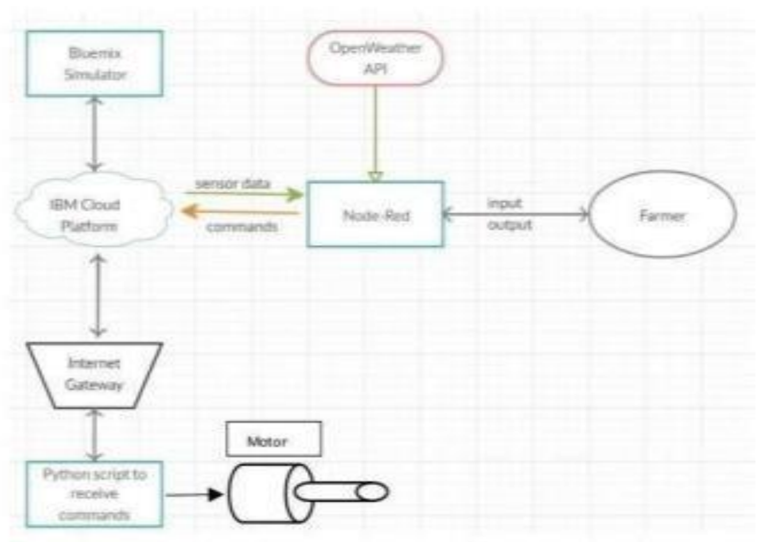
CHAPTER 5 - PROJECT DESIGN

5.1 Data Flow Diagrams


D
Rain Sensor
D
Temperature and Humdity sensor
D
Soil Moisture Sensor
commands to control pumps
Water Pumps
temperature and humidity values
Moisture value
status about rain
Raspberry Pi
D
User
User Interactions
subscribes commands from cloud
publishes values to cloud
IBM Watson Platform
App
User Interactions
Data acquisition using API
Website
Data acquisition using API
Node Red

5.2 Solution and Technical Architecture

The technical architecture diagram is as follows:

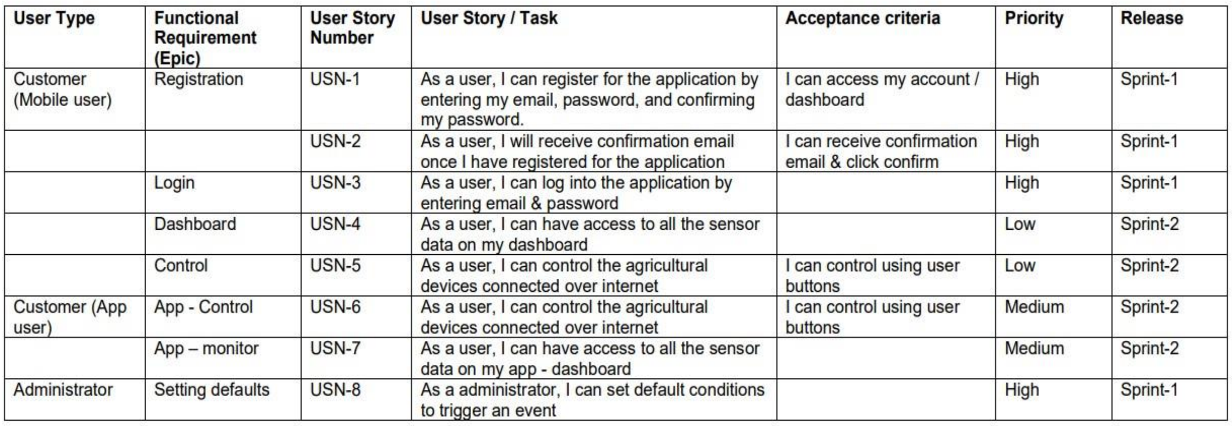


Guidelines:

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

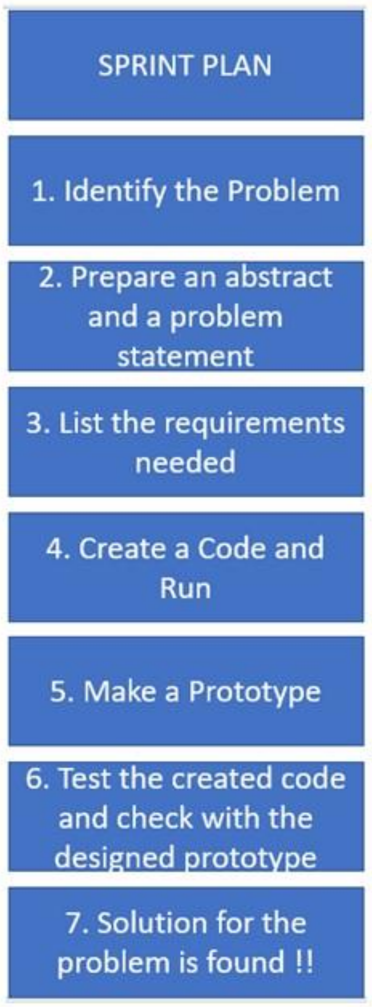
5.3 User Stories



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

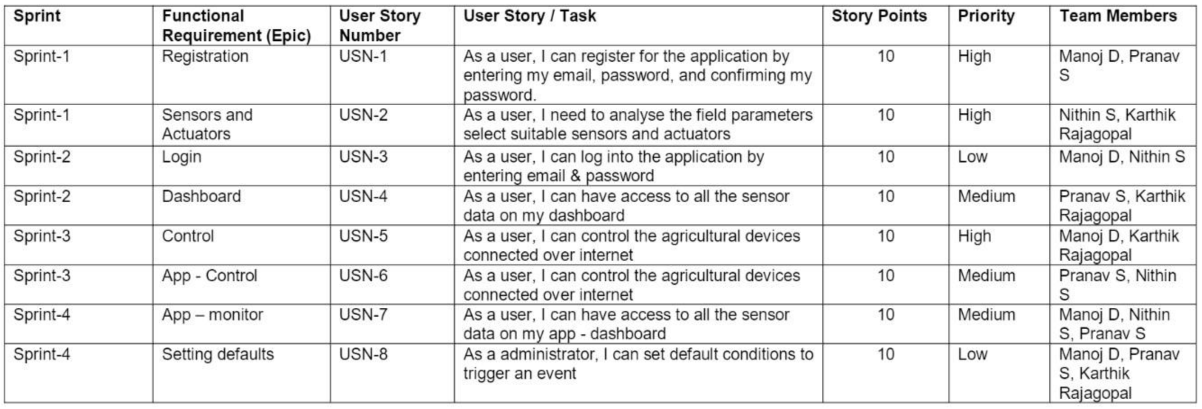
CHAPTER 6 - PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation


SPRINT PLAN
1. Identify the Problem
2. Prepare an abstract and a problem statement
3. List the requirements needed
4. Create a Code and Run
5. Make a Prototype
6. Test the created code and check with the designed prototype
7. Solution for the problem is found !!

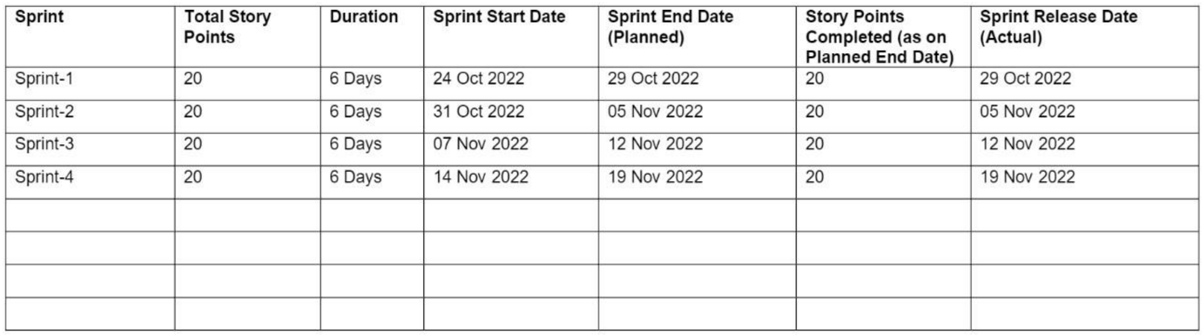
6.2 Sprint Delivery and Schedule

The Sprint schedule is as follows:



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	Manoj D, Pranav S
Sprint-1	Sensors and Actuators	USN-2	As a user, I need to analyse the field parameters select suitable sensors and actuators	10	High	Nithin S, Karthik Rajagopal
Sprint-2	Login	USN-3	As a user, I can log into the application by entering email & password	10	Low	Manoj D, Nithin S
Sprint-2	Dashboard	USN-4	As a user, I can have access to all the sensor data on my dashboard	10	Medium	Pranav S, Karthik Rajagopal
Sprint-3	Control	USN-5	As a user, I can control the agricultural devices connected over internet	10	High	Manoj D, Karthik Rajagopal
Sprint-3	App - Control	USN-6	As a user, I can control the agricultural devices connected over internet	10	Medium	Pranav S, Nithin S
Sprint-4	App – monitor	USN-7	As a user, I can have access to all the sensor data on my app - dashboard	10	Medium	Manoj D, Nithin S, Pranav S
Sprint-4	Setting defaults	USN-8	As a administrator, I can set default conditions to trigger an event	10	Low	Manoj D, Pranav S, Karthik Rajagopal

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	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

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

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

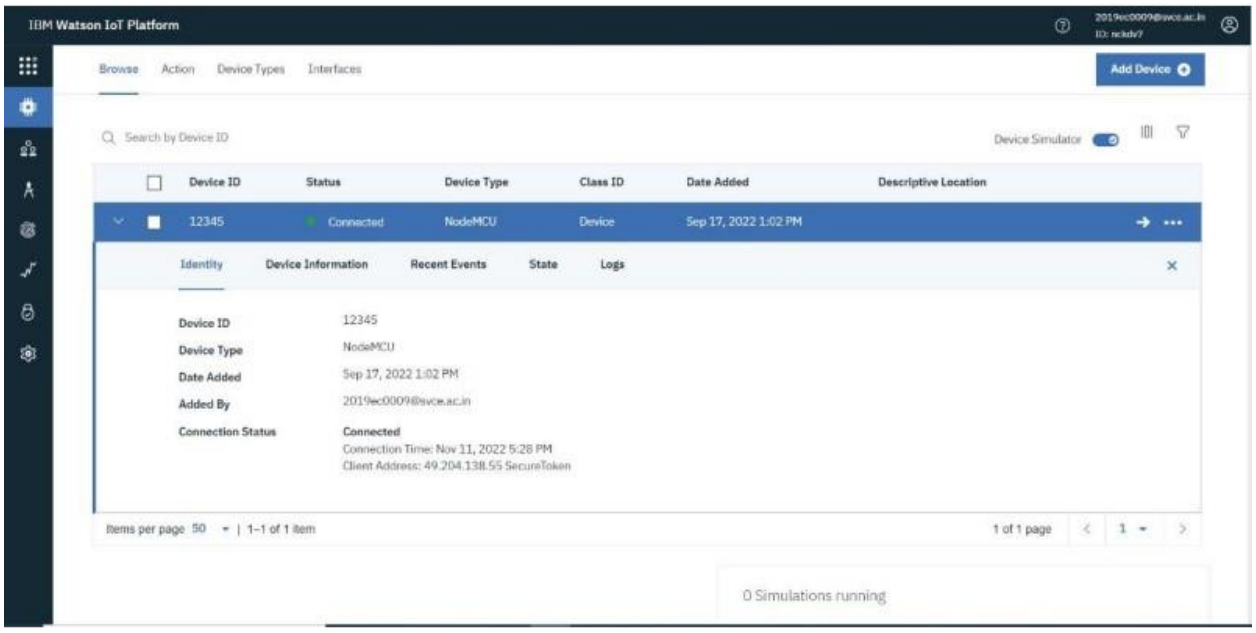
Burndown Chart:

A burndown chart is the graphical representation of work left to be done 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.

CHAPTER 7 - CODING AND SOLUTIONING

- **Configuration of the IBM Watson IOT Platform and a device:**

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.



	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
	12345	Connected	NodeMCU	Device	Sep 17, 2022 1:02 PM	

Device ID	12345
Device Type	NodeMCU
Date Added	Sep 17, 2022 1:02 PM
Added By	2019ec0009@svce.ac.in
Connection Status	Connected Connection Time: Nov 11, 2022 5:28 PM Client Address: 49.204.138.55 SecureToken

Items per page 50 | 1–1 of 1 item 1 of 1 page 1

- **Development of Python Script to publish data to IBM Watson IOT platform:**

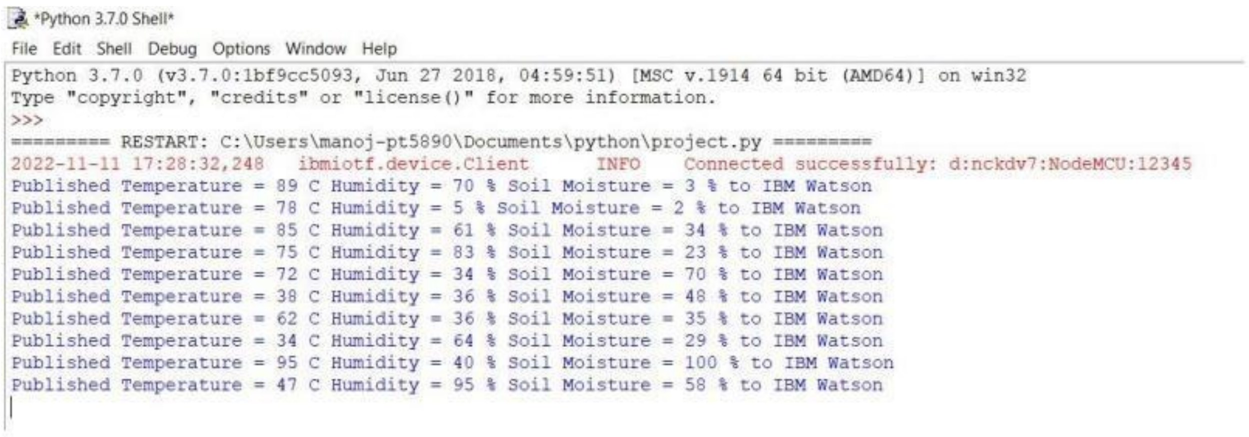
<u>Code</u>:

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

```
#Provide your IBM Watson Device Credentials
organization = "nckdv7"  deviceType =
"NodeMCU"
deviceId = "12345" authMethod = "token"  authToken = "12345678" #
Initialize GPIO  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(0,100)
pulse=random.randint(0,100)      moisture=    random.randint(0,100)
humidity=random.randint(0,100);  lat = 17  lon = 18  data = { 'temperature'
: temp, 'humidity' : humidity, 'Moisture' :
        moisture}  #print
        data


def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s
        %%" % humidity, "Soil Moisture = %s %%" % moisture,"to IBM
        Watson")
success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)  if not success:  print("Not connected
to IoTF")
```

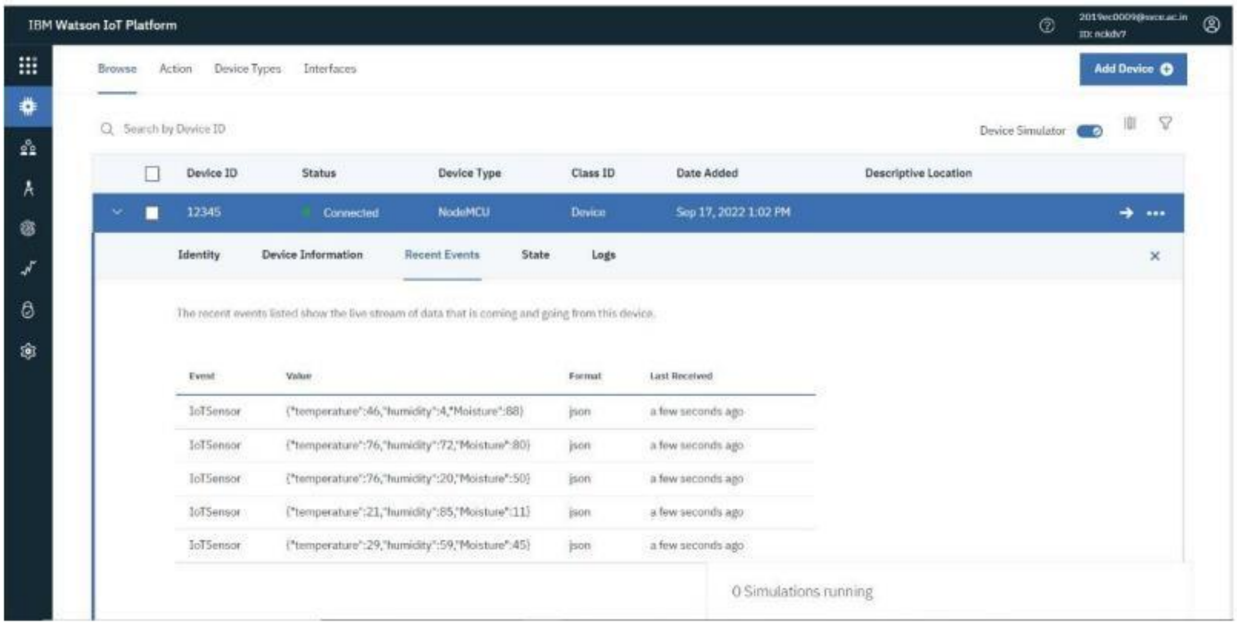
time.sleep(1) deviceCli.commandCallback = myCommandCallback # Disconnect the device and application from the cloud deviceCli.disconnect()

Python Code Output:



```
*Python 3.7.0 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========= RESTART: C:\Users\manoj-pt5890\Documents\python\project.py =========
2022-11-11 17:28:32,248   ibmiotf.device.Client      INFO    Connected successfully: d:nckdv7:NodeMCU:12345
Published Temperature = 89 C Humidity = 70 % Soil Moisture = 3 % to IBM Watson
Published Temperature = 78 C Humidity = 5 % Soil Moisture = 2 % to IBM Watson
Published Temperature = 85 C Humidity = 61 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 75 C Humidity = 83 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 72 C Humidity = 34 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 38 C Humidity = 36 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 62 C Humidity = 36 % Soil Moisture = 35 % to IBM Watson
Published Temperature = 34 C Humidity = 64 % Soil Moisture = 29 % to IBM Watson
Published Temperature = 95 C Humidity = 40 % Soil Moisture = 100 % to IBM Watson
Published Temperature = 47 C Humidity = 95 % Soil Moisture = 58 % to IBM Watson
```

IBM Cloud after publishing data:



Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
12345	Connected	NodeMCU	Device	Sep 17, 2022 1:02 PM	

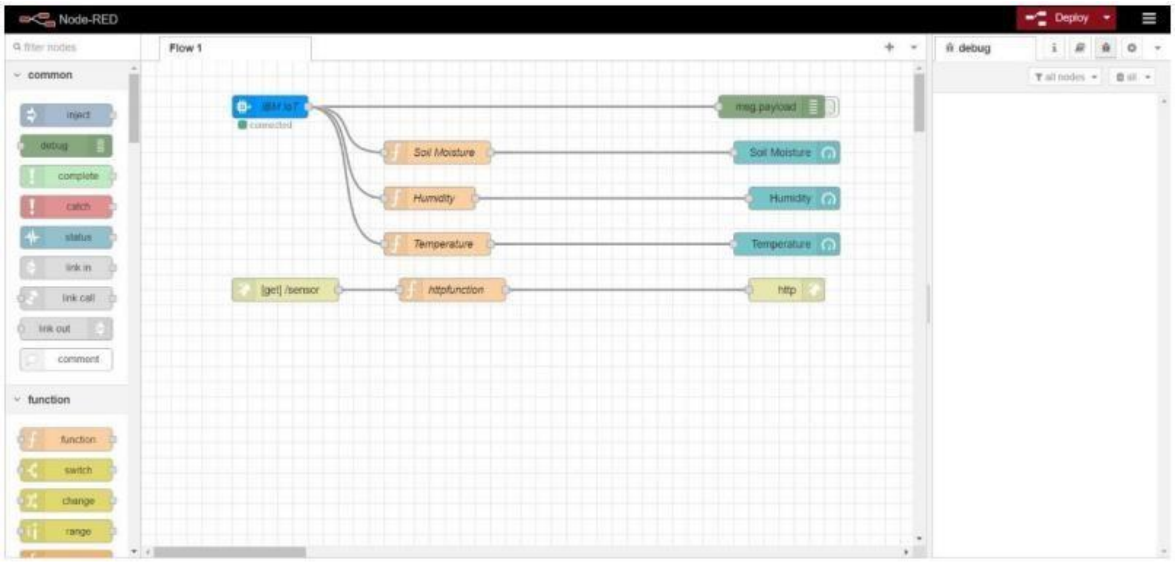
The recent events listed show the live stream of data that is coming and going from this device.

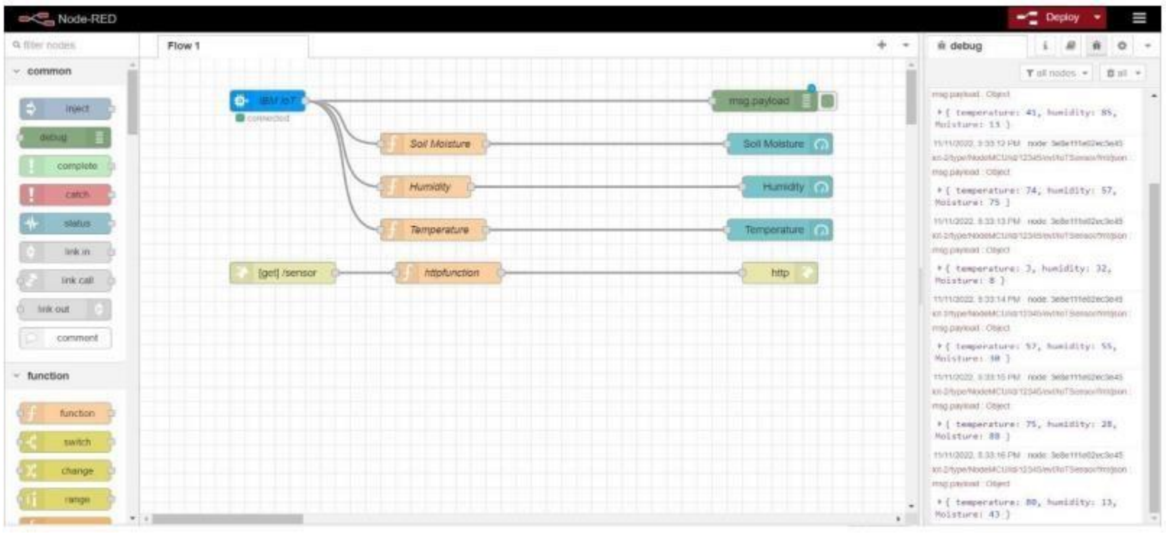
Event	Value	Format	Last Received
IoTSensor	{"temperature":46,"humidity":4,"Moisture":88}	json	a few seconds ago
IoTSensor	{"temperature":76,"humidity":72,"Moisture":80}	json	a few seconds ago
IoTSensor	{"temperature":76,"humidity":20,"Moisture":50}	json	a few seconds ago
IoTSensor	{"temperature":21,"humidity":85,"Moisture":11}	json	a few seconds ago
IoTSensor	{"temperature":29,"humidity":59,"Moisture":45}	json	a few seconds ago

0 Simulations running

- **Creation of Node Red Service for device events:**
  In the IBM Watson IOT platform, under the catalog, under the Node Red app service, an application is deployed using cloud foundry. In the cloud foundry, a group has been created and using the ci pipeline, the app url is obtained. Using the URL, the Node red is launched. The IBM Watson IOT platform is connected to Node red using the IBM IoT palette. Using

appropriate palettes, the data published in the IBM IoT platform is printed in the debug window of Node red.





```
msg.payload : Object
▸{ temperature: 41, humidity: 85, Moisture: 13 }
11/11/2022, 3:33:12 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json : msg.payload : Object
▸{ temperature: 74, humidity: 57, Moisture: 75 }
11/11/2022, 3:33:13 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json : msg.payload : Object
▸{ temperature: 3, humidity: 32, Moisture: 8 }
11/11/2022, 3:33:14 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json : msg.payload : Object
▸{ temperature: 57, humidity: 55, Moisture: 10 }
11/11/2022, 3:33:15 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json : msg.payload : Object
▸{ temperature: 75, humidity: 28, Moisture: 88 }
11/11/2022, 3:33:16 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json : msg.payload : Object
▸{ temperature: 80, humidity: 13, Moisture: 43 }
```

Code block for the function palette:

1) **Soil moisture:**

```
Soil = msg.payload.Moisture
msg.payload = "Soil Moisture :
" global.set('m',Soil)
msg.payload = Math.round(Soil)
return msg;
```

2) **Humidity:**

Humidity = msg.payload.humidity
msg.payload = "Humidity : "
global.set('h',Humidity) msg.payload
= Math.round(Humidity) return msg;

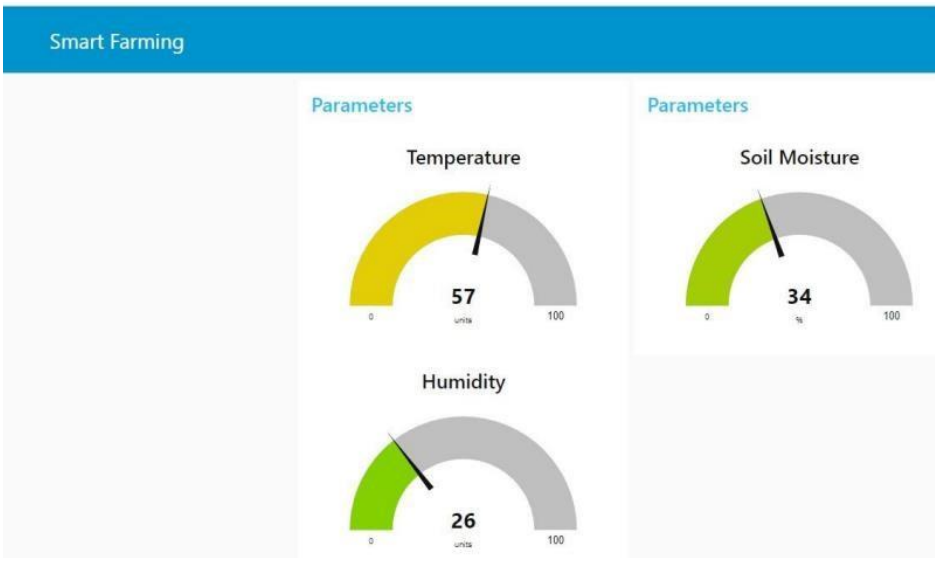
3) Temperature:

Temperature = msg.payload.temperature
msg.payload = "Temperature : "
global.set('t',Temperature) msg.payload
=Math.round(Temperature) return msg;

4) HTTP Function:

msg.payload = {"Temperature:": global.get('t'),"Humidity:":
global.get('h'),"Soil Moisture:": global.get('m')} return
msg;

- **Creation of Website dashboard:**
A website dashboard has been created using the gauge palette. It can be accessed by adding "/ui" in the main url of Node red. This dashboard displays the gauge representation of the data published in the IBM IOT platform.



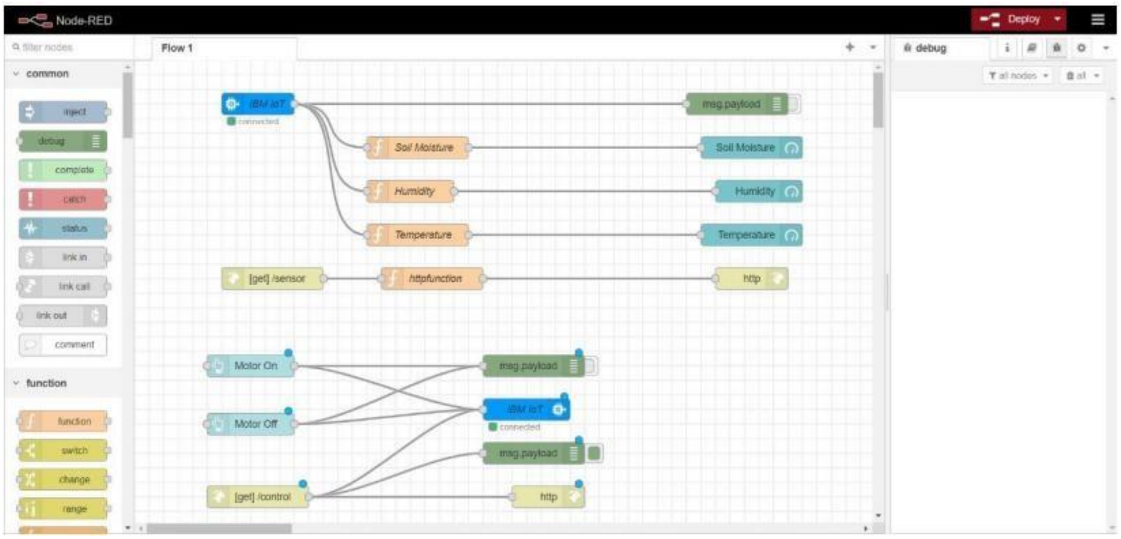
Python code used:

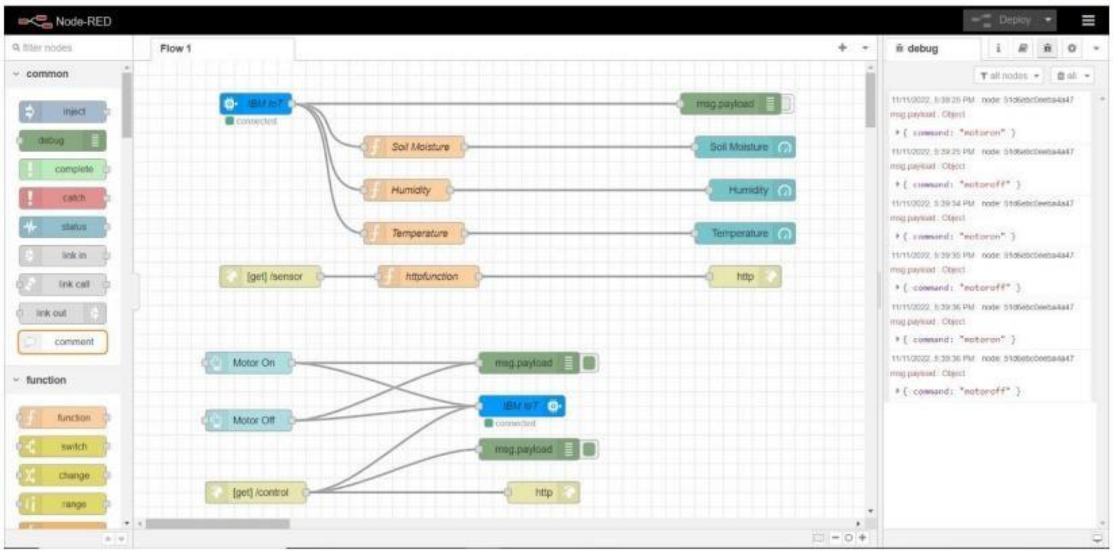
```
import time import sys
import
ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device
Credentials  organization = "nckdv7"
deviceType = "NodeMCU"  deviceId =
"12345"  authMethod = "token"  authToken =
"12345678" # Initialize GPIO  try:
        deviceOptions = {"org": 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(0,100)            pulse=random.randint(0,100)
moisture=                                   random.randint(0,100)
humidity=random.randint(0,100);   lat = 17  lon = 18  data = {
'temperature' : temp, 'humidity' :
        humidity, 'Moisture' : moisture}
```

```
          #print     data          def
      myOnPublishCallback():
          print ("Published Temperature = %s C" % temp, "Humidity =
          %s  %%"  %  humidity,  "Soil  Moisture  =  %s  %%"  %
          moisture,"to IBM Watson")
      success = deviceCli.publishEvent("IoTSensor", "json", data,
      qos=0, on_publish=myOnPublishCallback)  if not success:
      print("Not connected to IoTF")
      time.sleep(1)           deviceCli.commandCallback      =
myCommandCallback  # Disconnect the device and application
from the cloud  deviceCli.disconnect()
```

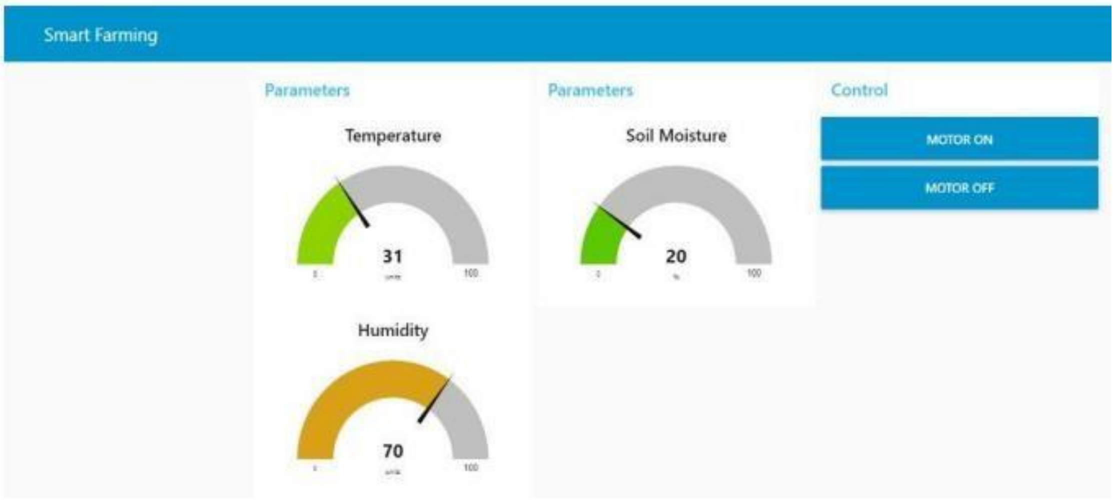
 Creation of Node red service for device commands:

In addition to the palettes used in the Sprint-2, additional palettes such as buttons have been included to control devices by giving commands and the output is printed in the debug whenever a specific command is given.





```
11/11/2022, 5:39:25 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoron" }
11/11/2022, 5:39:25 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoroff" }
11/11/2022, 5:39:34 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoron" }
11/11/2022, 5:39:35 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoroff" }
11/11/2022, 5:39:36 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoron" }
11/11/2022, 5:39:36 PM  node: 51d6ebc0eeba4a47
msg.payload : Object
▸{ command: "motoroff" }
```



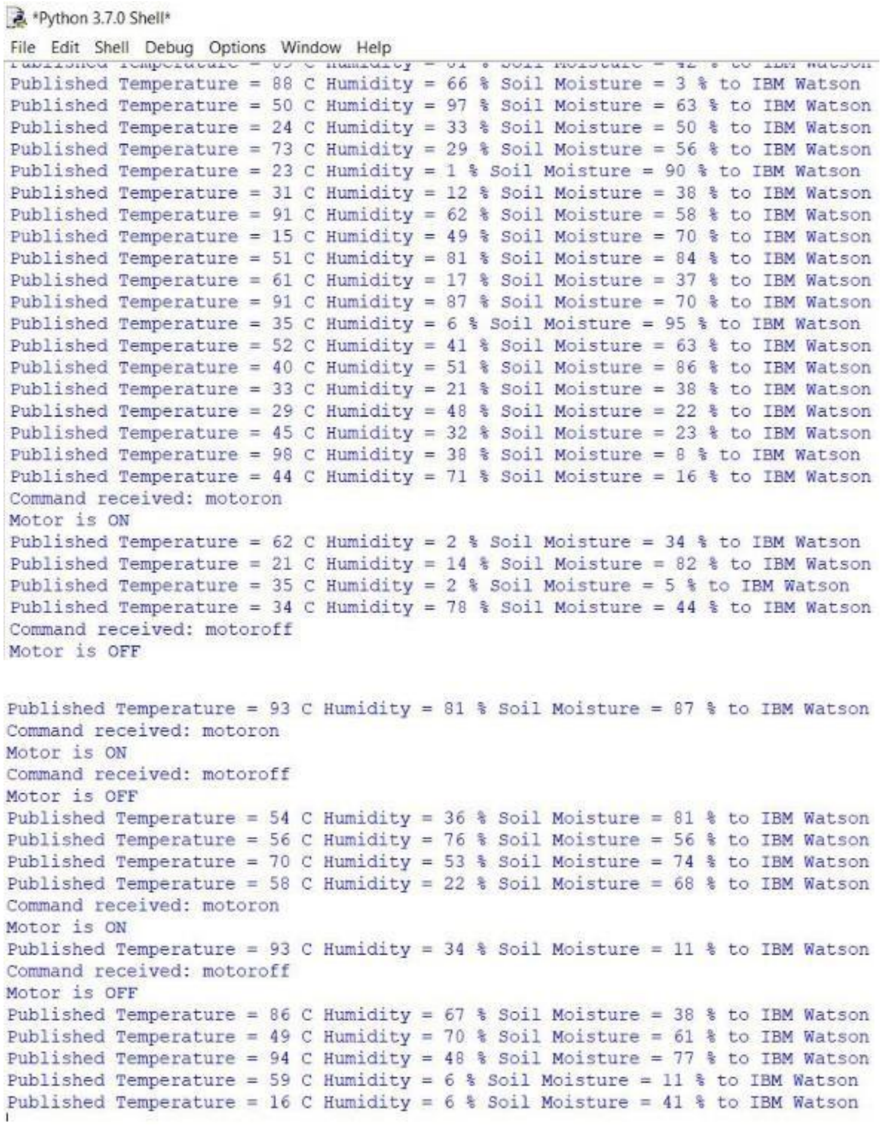
Development of Python script to subscribe command from the IBM IOT platform:

Code:

```
import time import sys
import
ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials organization =
"nckdv7" 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")  else:
                print("Motor is OFF")
         #print(cmd) try: deviceOptions = {"org": organization, "type":
deviceType, "id": deviceId, "authmethod": 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(0,100)
        pulse=random.randint(0,100) moisture= random.randint(0,100)
        humidity=random.randint(0,100); lat = 17 lon = 18 data = {
        'temperature' : temp, 'humidity' : humidity, 'Moisture' :
```

```
        moisture}
        #print data


    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s
        %%" % humidity, "Soil Moisture = %s %%" % moisture,"to IBM
        Watson")  success = deviceCli.publishEvent("IoTSensor",
    "json", data, qos=0,   on_publish=myOnPublishCallback)  if not
    success:
        print("Not connected to IoTF")
time.sleep(1)              deviceCli.commandCallback     =
    myCommandCallback  #  Disconnect  the  device  and
    application from the cloud deviceCli.disconnect()
```

<u>Output:</u>

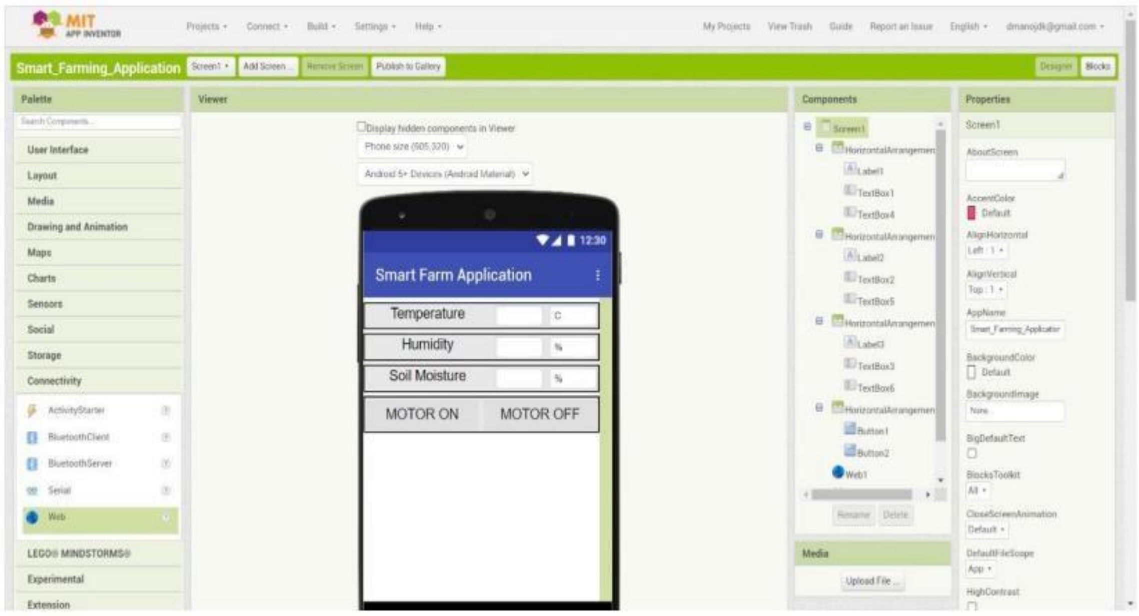


```
Published Temperature = 88 C Humidity = 66 % Soil Moisture = 3 % to IBM Watson
Published Temperature = 50 C Humidity = 97 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 24 C Humidity = 33 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 73 C Humidity = 29 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 23 C Humidity = 1 % Soil Moisture = 90 % to IBM Watson
Published Temperature = 31 C Humidity = 12 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 91 C Humidity = 62 % Soil Moisture = 58 % to IBM Watson
Published Temperature = 15 C Humidity = 49 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 51 C Humidity = 81 % Soil Moisture = 84 % to IBM Watson
Published Temperature = 61 C Humidity = 17 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 91 C Humidity = 87 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 35 C Humidity = 6 % Soil Moisture = 95 % to IBM Watson
Published Temperature = 52 C Humidity = 41 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 40 C Humidity = 51 % Soil Moisture = 86 % to IBM Watson
Published Temperature = 33 C Humidity = 21 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 29 C Humidity = 48 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 45 C Humidity = 32 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 98 C Humidity = 38 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 44 C Humidity = 71 % Soil Moisture = 16 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 62 C Humidity = 2 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 21 C Humidity = 14 % Soil Moisture = 82 % to IBM Watson
Published Temperature = 35 C Humidity = 2 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 34 C Humidity = 78 % Soil Moisture = 44 % to IBM Watson
Command received: motoroff
Motor is OFF

Published Temperature = 93 C Humidity = 81 % Soil Moisture = 87 % to IBM Watson
Command received: motoron
Motor is ON
Command received: motoroff
Motor is OFF
Published Temperature = 54 C Humidity = 36 % Soil Moisture = 81 % to IBM Watson
Published Temperature = 56 C Humidity = 76 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 70 C Humidity = 53 % Soil Moisture = 74 % to IBM Watson
Published Temperature = 58 C Humidity = 22 % Soil Moisture = 68 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 93 C Humidity = 34 % Soil Moisture = 11 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 86 C Humidity = 67 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 49 C Humidity = 70 % Soil Moisture = 61 % to IBM Watson
Published Temperature = 94 C Humidity = 48 % Soil Moisture = 77 % to IBM Watson
Published Temperature = 59 C Humidity = 6 % Soil Moisture = 11 % to IBM Watson
Published Temperature = 16 C Humidity = 6 % Soil Moisture = 41 % to IBM Watson
```

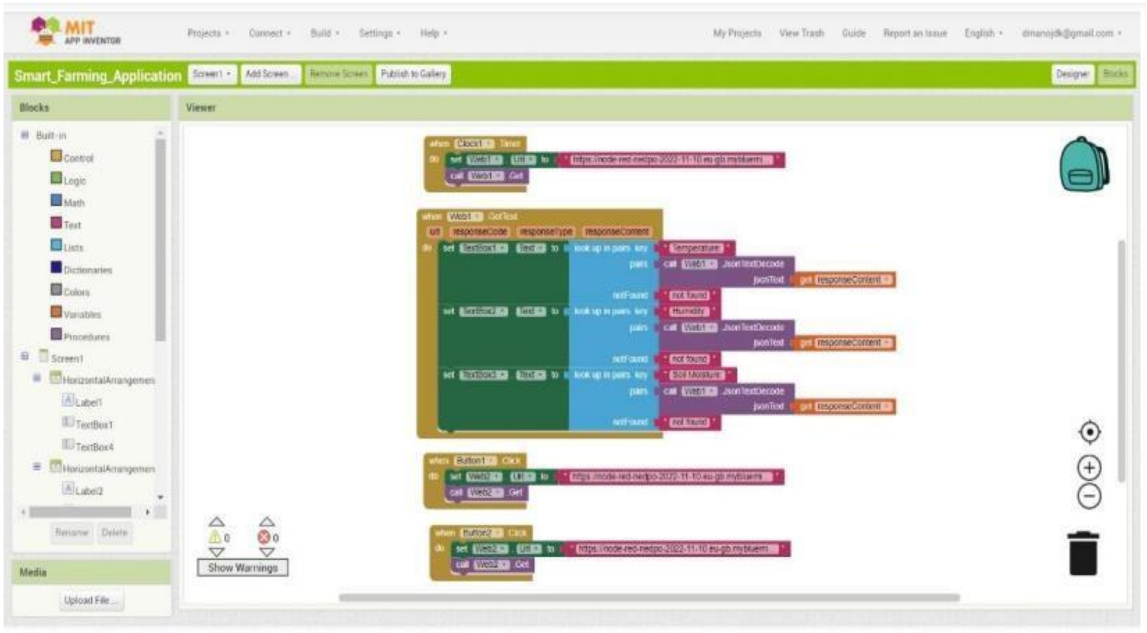
- **Development of Mobile application using MIT App Inventor:**

  In the MIT App Inventor platform, an application is created which monitors the farmland parameters such as temperature, humidity, soil moisture and controls the actuators such as motors.

<u>MIT App Front End:</u>



Backend:

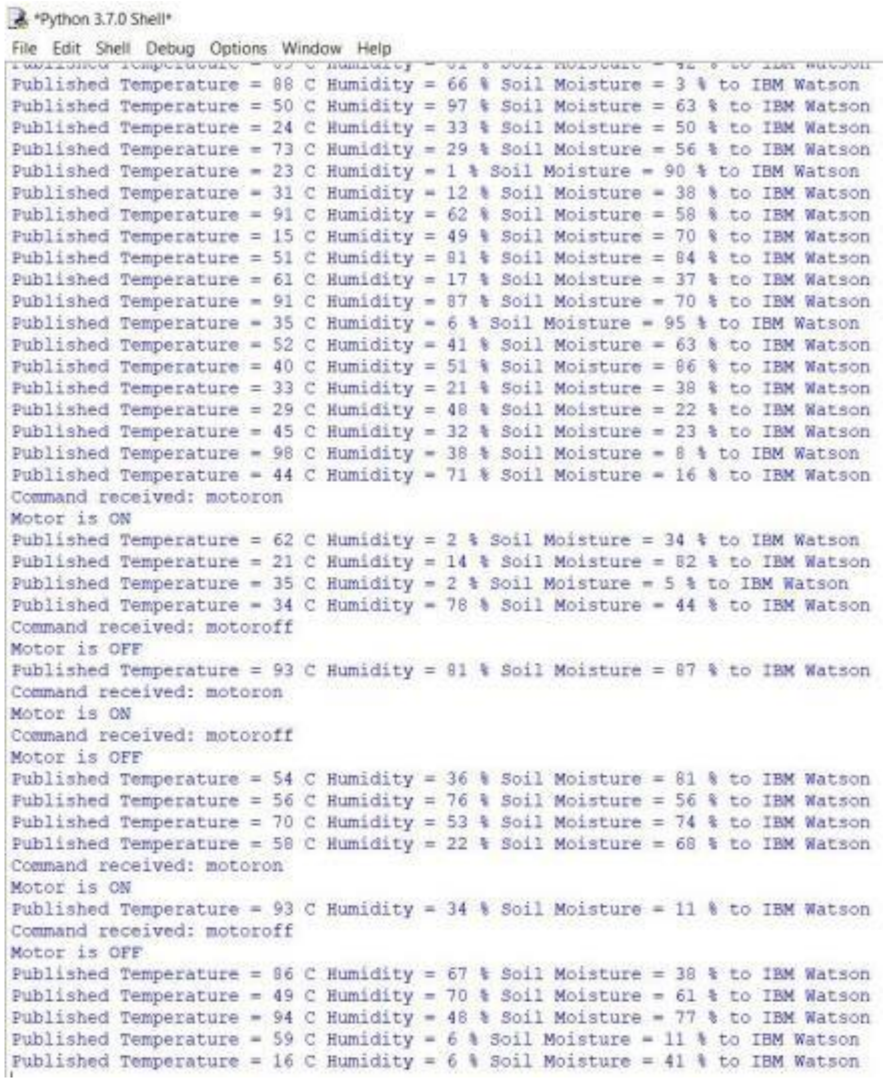


App working:

The app works based on HTTP protocol. The app uses HTTP GET method to parse the JSON data from the Node red website and displays the value in the UI. Using the HTTP POST method, the app sends command when a specific button is pressed. From where, the python code subscribes the command data from the cloud thereby notifying the command is received.

Python code:
```
import time
import sys import
ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials organization =
"nckdv7" 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") else:
                print("Motor is OFF")
                #print(cmd)

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(0,100)
        pulse=random.randint(0,100) moisture= random.randint(0,100)
        humidity=random.randint(0,100); lat = 17 lon = 18 data = {
        'temperature' : temp, 'humidity' : humidity, 'Moisture' :
          moisture}
          #print     data          def
myOnPublishCallback():
          print ("Published Temperature = %s C" % temp, "Humidity = %s
            %%" % humidity, "Soil Moisture = %s %%" % moisture,"to IBM
          Watson") success = deviceCli.publishEvent("IoTSensor",
"json", data, qos=0,  on_publish=myOnPublishCallback)  if not
success:
  print("Not    connected    to    IoTF")       time.sleep(1)
deviceCli.commandCallback  =  myCommandCallback
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

Output:



```
Published Temperature = 88 C Humidity = 66 % Soil Moisture = 3 % to IBM Watson
Published Temperature = 50 C Humidity = 97 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 24 C Humidity = 33 % Soil Moisture = 50 % to IBM Watson
Published Temperature = 73 C Humidity = 29 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 23 C Humidity = 1 % Soil Moisture = 90 % to IBM Watson
Published Temperature = 31 C Humidity = 12 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 91 C Humidity = 62 % Soil Moisture = 58 % to IBM Watson
Published Temperature = 15 C Humidity = 49 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 51 C Humidity = 81 % Soil Moisture = 84 % to IBM Watson
Published Temperature = 61 C Humidity = 17 % Soil Moisture = 37 % to IBM Watson
Published Temperature = 91 C Humidity = 87 % Soil Moisture = 70 % to IBM Watson
Published Temperature = 35 C Humidity = 6 % Soil Moisture = 95 % to IBM Watson
Published Temperature = 52 C Humidity = 41 % Soil Moisture = 63 % to IBM Watson
Published Temperature = 40 C Humidity = 51 % Soil Moisture = 86 % to IBM Watson
Published Temperature = 33 C Humidity = 21 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 29 C Humidity = 48 % Soil Moisture = 22 % to IBM Watson
Published Temperature = 45 C Humidity = 32 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 98 C Humidity = 38 % Soil Moisture = 8 % to IBM Watson
Published Temperature = 44 C Humidity = 71 % Soil Moisture = 16 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 62 C Humidity = 2 % Soil Moisture = 34 % to IBM Watson
Published Temperature = 21 C Humidity = 14 % Soil Moisture = 82 % to IBM Watson
Published Temperature = 35 C Humidity = 2 % Soil Moisture = 5 % to IBM Watson
Published Temperature = 34 C Humidity = 78 % Soil Moisture = 44 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 93 C Humidity = 81 % Soil Moisture = 87 % to IBM Watson
Command received: motoron
Motor is ON
Command received: motoroff
Motor is OFF
Published Temperature = 54 C Humidity = 36 % Soil Moisture = 81 % to IBM Watson
Published Temperature = 56 C Humidity = 76 % Soil Moisture = 56 % to IBM Watson
Published Temperature = 70 C Humidity = 53 % Soil Moisture = 74 % to IBM Watson
Published Temperature = 58 C Humidity = 22 % Soil Moisture = 68 % to IBM Watson
Command received: motoron
Motor is ON
Published Temperature = 93 C Humidity = 34 % Soil Moisture = 11 % to IBM Watson
Command received: motoroff
Motor is OFF
Published Temperature = 86 C Humidity = 67 % Soil Moisture = 38 % to IBM Watson
Published Temperature = 49 C Humidity = 70 % Soil Moisture = 61 % to IBM Watson
Published Temperature = 94 C Humidity = 48 % Soil Moisture = 77 % to IBM Watson
Published Temperature = 59 C Humidity = 6 % Soil Moisture = 11 % to IBM Watson
Published Temperature = 16 C Humidity = 6 % Soil Moisture = 41 % to IBM Watson
```

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:

- 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. o 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. o 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.

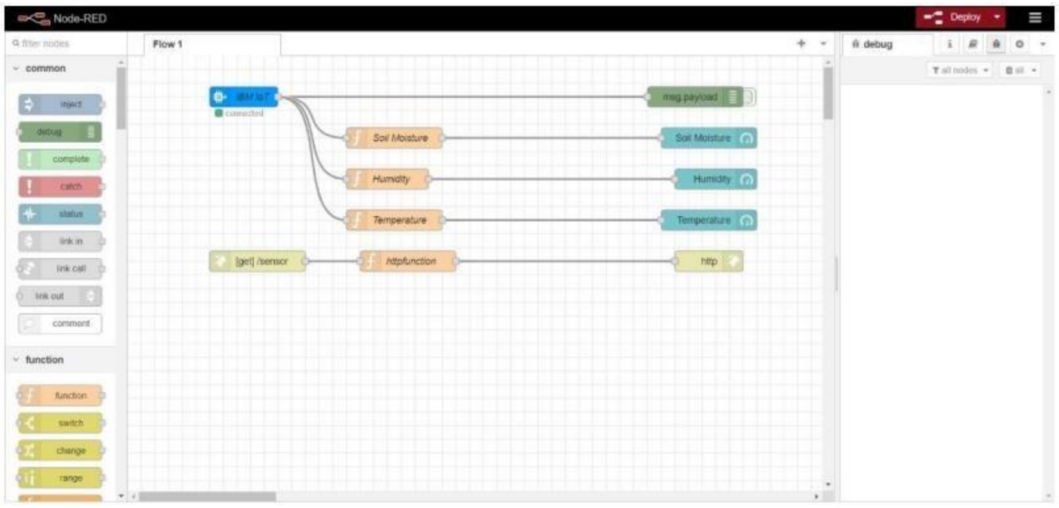
CHAPTER 12 – APPENDIX

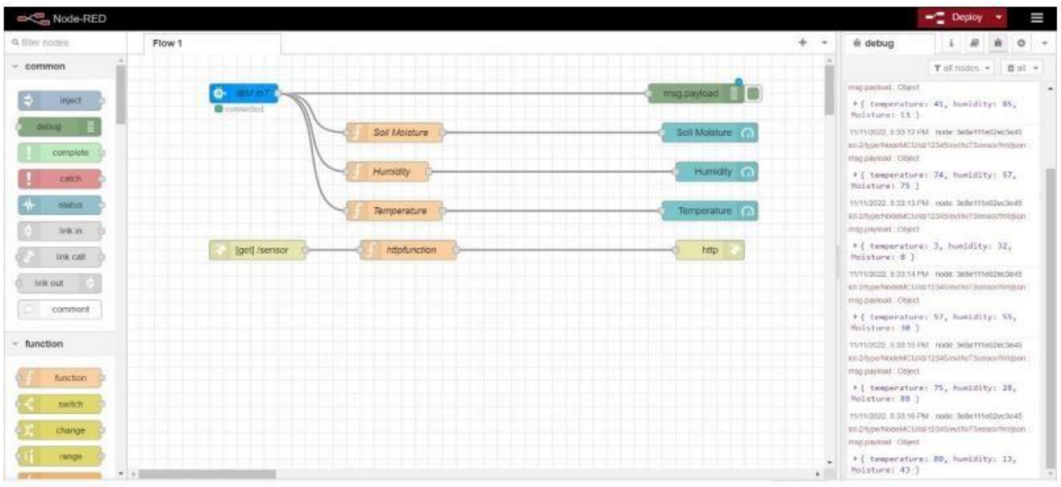
12.1 Source Code:

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

#Provide your IBM Watson Device
Credentials organization = "nckdv7"
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") else:
        print("Motor is OFF")
    #print(cmd) 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(0,100)
pulse=random.randint(0,100)
moisture= random.randint(0,100)
humidity=random.randint(0,100);
        lat   =      17
lon = 18

        data = { 'temp' : temp, 'humidity' : humidity, 'Soil Moisture' :
moisture}
        #print  data                     def
myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity
= %s %%" % humidity, "Soil Moisture = %s %%" % moisture,"to
IBM Watson")

        success  =  deviceCli.publishEvent("IoTSensor",  "json",  data,
qos=0,  on_publish=myOnPublishCallback)              if  not  success:
print("Not connected to IoTF")         time.sleep(1)

        deviceCli.commandCallback = myCommandCallback

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

Node Red Service Creation:





```
msg.payload : Object
▸{ temperature: 41, humidity: 85, Moisture: 13 }
11/11/2022, 5:33:12 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 74, humidity: 57, Moisture: 75 }
11/11/2022, 5:33:13 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 3, humidity: 32, Moisture: 8 }
11/11/2022, 5:33:14 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 57, humidity: 55, Moisture: 10 }
11/11/2022, 5:33:15 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 75, humidity: 28, Moisture: 88 }
11/11/2022, 5:33:16 PM  node: 3e8e111e02ec3e45
iot-2/type/NodeMCU/id/12345/evt/IoTSensor/fmt/json
msg.payload : Object
▸{ temperature: 80, humidity: 13, Moisture: 43 }
```

Code block for the function palette:

1) <u>Soil moisture</u>:

Soil = msg.payload.Moisture msg.payload = "Soil Moisture : " global.set('m',Soil) msg.payload = Math.round(Soil) return msg; **2)**

<u>Humidity:</u>

Humidity = msg.payload.humidity msg.payload = "Humidity : "

global.set('h',Humidity) msg.payload = Math.round(Humidity) return msg;

3) **<u>Temperature:</u>**

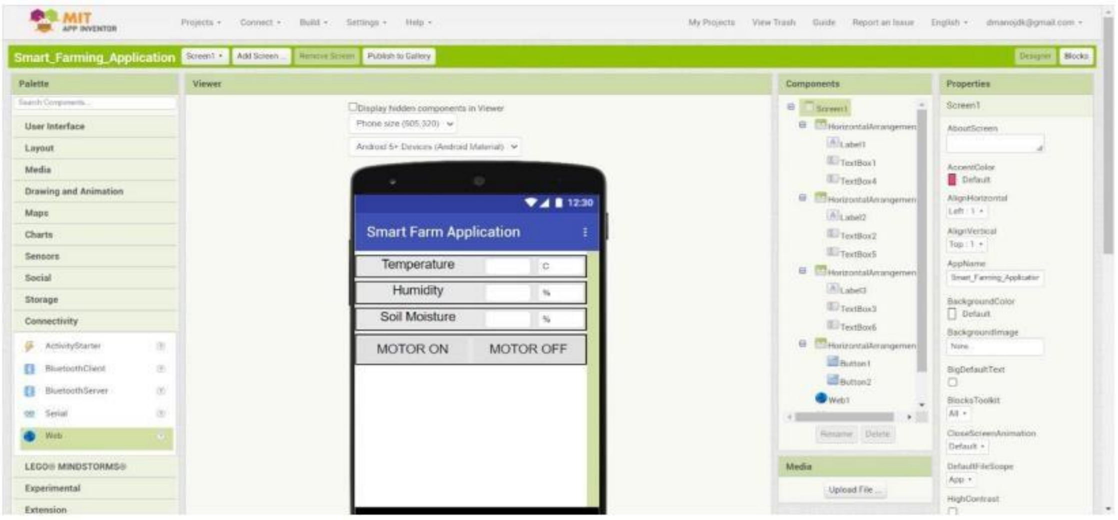
Temperature = msg.payload.temperature msg.payload = "Temperature : " global.set('t',Temperature) msg.payload =Math.round(Temperature) return msg;

4) **<u>HTTP Function:</u>**

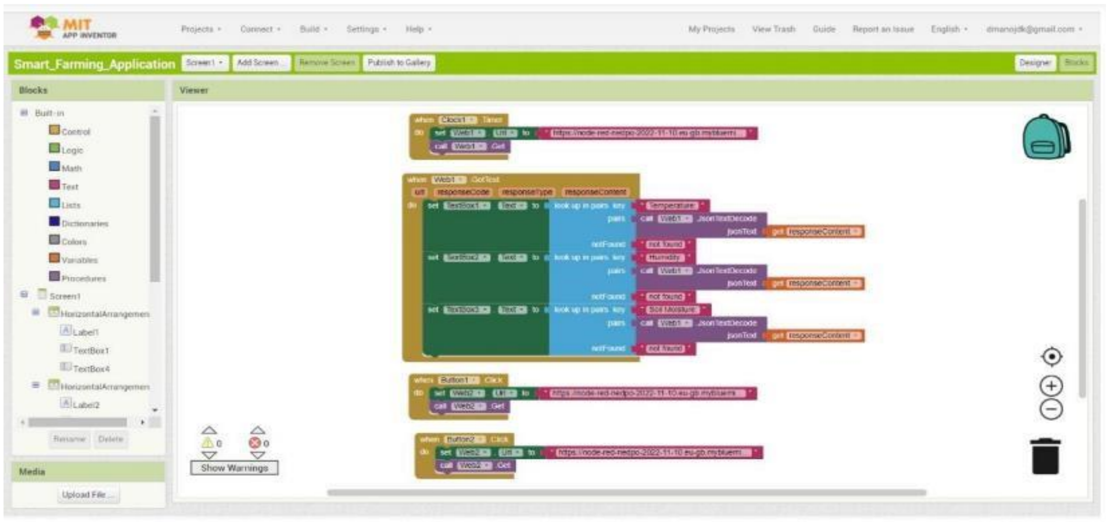
msg.payload = {"Temperature:": global.get('t'),"Humidity:": global.get('h'),"Soil Moisture:": global.get('m')}

return msg;

<u>MIT App Front End:</u>



<u>Backend</u>:



```
when Clock1 .Timer
do  set Web1 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web1 .Get

when Web1 .GotText
 url  responseCode  responseType  responseContent
do  set TextBox1 . Text to  look up in pairs key "Temperature"
                                          pairs call Web1 .JsonTextDecode
                                                      jsonText get responseContent
                                       notFound "not found"
    set TextBox2 . Text to  look up in pairs key "Humidity"
                                          pairs call Web1 .JsonTextDecode
                                                      jsonText get responseContent
                                       notFound "not found"
    set TextBox3 . Text to  look up in pairs key "Soil Moisture"
                                          pairs call Web1 .JsonTextDecode
                                                      jsonText get responseContent
                                       notFound "not found"

when Button1 .Click
do  set Web2 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web2 .Get

when Button2 .Click
do  set Web2 . Url to "https://node-red-nedpo-2022-11-10.eu-gb.mybluemi..."
    call Web2 .Get
```

12.2 GitHub and Project Demo Link:

GitHub:

https://github.com/IBM-EPBL/IBM-Project-9537-1659017418

Project Demo Link:

https://drive.google.com/drive/folders/122gQH0fA1KAaUMibmFjXdTvlKfELJKjq?usp=sharing