PROJECT REPORT

Smart Farmer - IoT Enabled Smart Farming Application

TEAM ID: PNT2022TMID28706

TEAM MEMBERS:

K.Roopini

Roshni.T

P.Swathi Mithra

Daddala Gowthami

INDEX

1. INTRODUCTION

1 PROJECT OVERVIEW:

In Agriculture, yield depends on many factors such as seeds quality, soil type, moisture, temperature, and other climatic factors. As a result, production of food-grains fluctuates year after year if any of factors make an impact. A year of abundant output of cereals is often followed by a year of acute shortage especially in India. Due to this problem, obtained total yield was not meeting to food requirements of people and as a result leaving many people to starvation.

This has been for many years due to Traditional Agriculture was followed. In the recent years Government started many initiatives like setting soil testing labs, good quality fertilizers and seeds and modern equipment like tractors etc. and most importantly Modern Agriculture had taken shape. But still many farmers do not have any information on climatic and plant conditions beforehand so that requires action can be taken care.

1.2PURPOSE:

Through many scientific research, it is found that knowing beforehand the climatic conditions by farmers with an easy UI (User Interface) so that they can monitor closely and perform required actions.

Therefore, the purpose of this project is to make a Smart Agriculture System based on Internet of Things where the dashboard can give all the agricultural conditions of crops and weather conditions, also the water pump can be toggled on/ off through the same dashboard, instead of doing it manually.

Also, the all the climatic and crop condition information is recorded for future reference and analysis.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The Traditional agriculture methods is still used by many farmers and though a small percentage of farmers converted into modern agriculture, majority of yield is not produced due to no easy-to-use system to closely monitor the crop conditions like moisture, temperature etc.

Also, another major issue is the unpredictable weather conditions and the farmers wholly depend on Television broadcast which does not give real time updates.

2.2 REFERENCE:

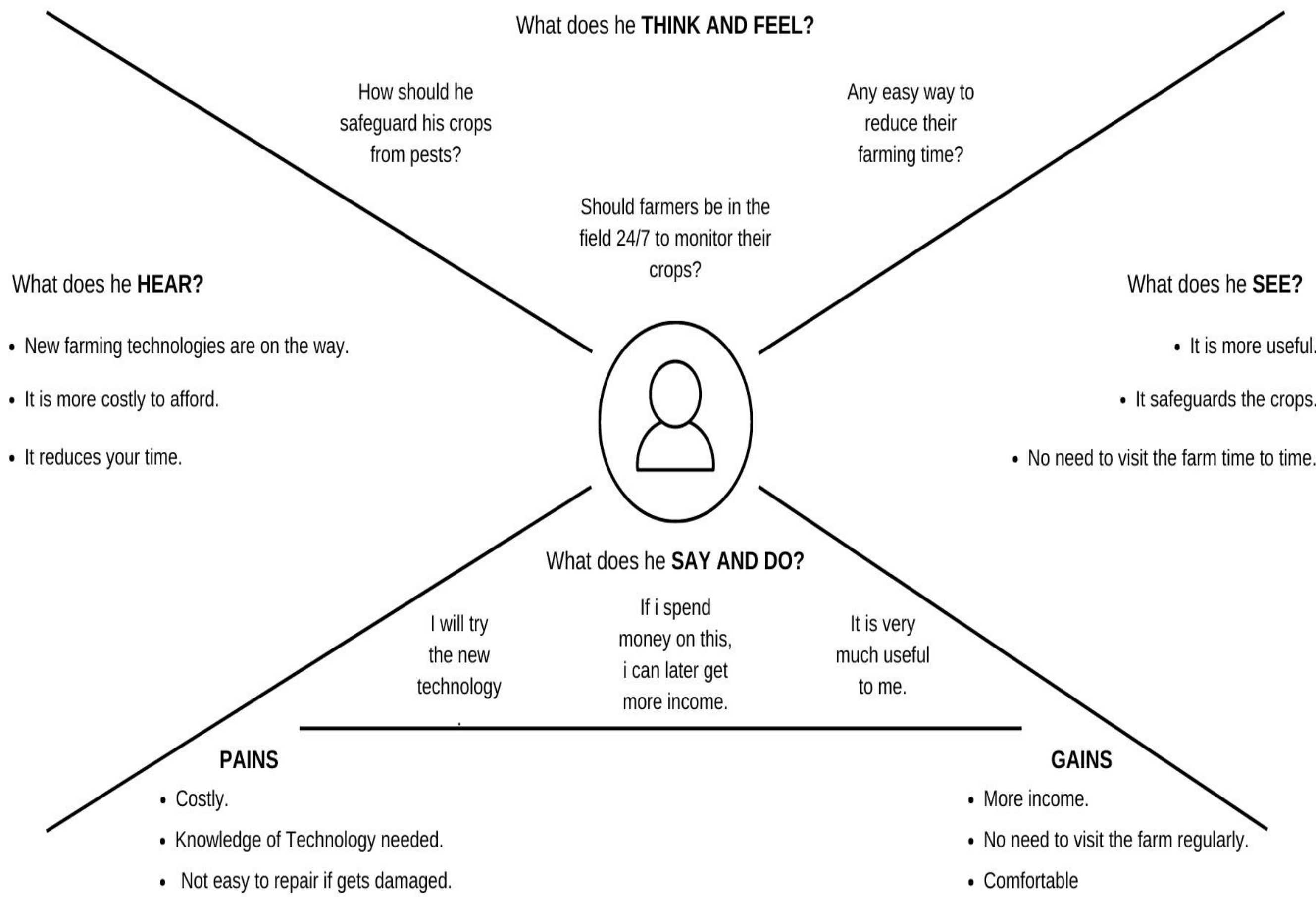
To overcome the above mentioned we proposed to build user friendly dashboard where the farmer can get all the crop conditions in real time and that too remotely and can track the climatic changes and conditions at his/her location.

To track the crop conditions, we can choose from variety of IoT devices and micro controllers which monitor the condition and shows the gathered information in the dashboard.

Also, the motor pump can be controlled remotely. For the weather changes to be shown in the dashboard we can open weather API for accurate and real time information.

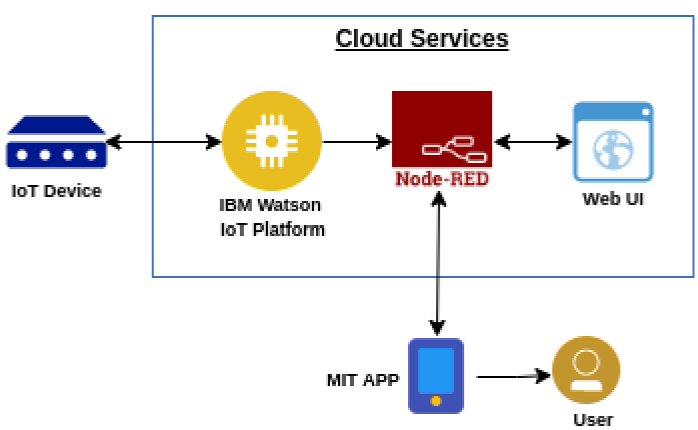
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



Brainstrom and idea Prioritization	Problem Statement	Ideas	Group Ideas
Team Members • K Roopini • Roshni T • P Swathimithra • Daddala Gowthami **Goals** Monitoring and Maintenance of crops remotely.	IOT-based agriculture system helps the farmer in monitoring different parameters of his field like soil moisture,Temperature ,humidity using some sensors. Farmers can monitor all the sensor parameters by using a web or mobile application 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.	**K Roopini** Monitoring crops using sensors. Controlling irrigation ,remotely using internet. Automated farming if farmer is not present. **Roshni T** Controlling irrigation through smartphone. Flexible ui interface. Crop health prediction. **P Swathimithra** Irrigation calander. Sensor parameter database **Daddala Gowthami** Temperature sensor Humidity sensor Soil moisture sensor	Farmer Controlling and monitoring crops remotely using smartphone. Autonomous irrigation mode. **Specification** Monitoring temperature,humidity and soil moisture using sensors. Controlling irrigation if needed autonomously.

ARCHITECTURE:



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Farmers should be in the farm field to monitor their crop field, if any emergency occurs for farmer to go outside there will be lack of irrigation in farm field which lead to damage in crops health.
2.	Idea / Solution description	IoT-based agriculture system helps the farmer to monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors by using a web or mobile application.
3.	Novelty / Uniqueness	When the farmer is not near his field, he 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.
4.	Social Impact / Customer Satisfaction	A monthly subscription is charged to farmers for prediction and suggesting the irrigation timing based on sensors parameters like temperature, humidity, soil moisture.
5.	Business Model (Revenue Model)	A monthly subscription is charged to farmers for prediction and suggesting the irrigation timing based on sensors parameters like temperature, humidity, soil moisture.
6.	Scalability of the Solution	Image recognition-based prediction of crops health AI based automated irrigation using temperature, pressure, humidity, and soil moisture sensors.

3.4 Problem Solution Fit

1. CUSTOMER SEGMENT(S) CS

- Farmers who have farm field to yield crops who is seeking to save 80% of time and who needs to monitor and control more than one field at a time are out target Customers.

6. CUSTOMER CONSTRAINTS C

- Farmers who are uneducated will suffer operating smart phones and will find difficulty in reading and understanding crop parameters and will find difficult to control irrigation.

5. AVAILABLE AS

- Farmers can monitor crop parameters and control irrigation remotely using smart phone integrated to IoT.

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

- Farmers are forced to be in farm field, if any emergency Situation occurs and farmer is not in farm field there will be lack of irrigation which leads to crop damages.
- Satisfy customer's changing taste and expectations.

9. PROBLEM ROOT CAUSE RC

- In accuracy in predicting crop parameters manually, wasting lots of time and energy in farm field.

7. BEHAVIOUR BE

- Sensors are integrated in the farm field to monitor parameters and data in processed and sent to the cloud (node red) using raspberry pi, the farmer can see parameters and control irrigation using smart phone.

3. TRIGGERS TR

- Farmer want to save his time and control irrigation more than one farm field at same time.

10. YOUR SOLUTION SL

- IoT integrated remote farming using sensors, irrigation system and raspberry pi connected to node red, where farmer can monitor and control irrigation remotely.

8.1 ONLINE CHANNELS CH

- The emerging out of convergences of IT and farming techniques. it enhances the agricultural value chain through the application of Internet

4. EMOTIONS: BEFORE / AFTER EM

- Farmer get bored by wasting time in farm field for irritating, what if farmer were able to control irrigation by watching movie in theatre or by watching TV.

8.2 OFFLINE CHANNELS CH

- Users are in offline they are only known about the previous information about the field

4. REQUIREMENT ANALYSIS

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	raspberry pi	To interface temperature, humidity, soil moisture sensors and irrigation system(motor).
FR-2	IBM cloud	To store and display sensor parameters and to control irrigation using internet.
FR-3	Node-RED	To program raspberry pi and integrate it with cloud.
FR-4	MIT app inventor	To create app which will the display sensor parameters and to control irrigation systems.
FR-5	Open Weather API	Get the data and access the resource.

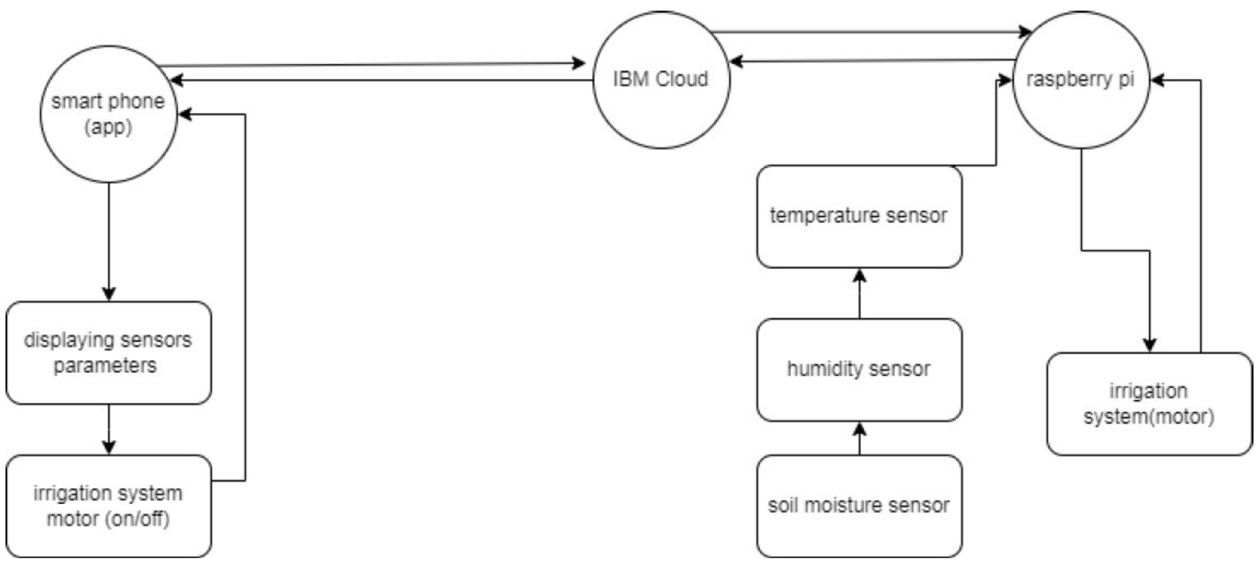
Non-Functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The temperature sensor, humidity sensor, soil moisture sensor and irrigation system(motor) is connected to raspberry pi which is connected to IBM cloud, the farmer can view temperature, humidity and soil moisture in his smart phone and can also control irrigation using his smart phone connected to internet
NFR-2	**Security**	User ID and password is provided to farmer to prevent third party access.
NFR-3	**Reliability**	It specifies how likely the system or its element would run without a failure.
NFR-4	**Performance**	Every 10 seconds to raspberry pi will update sensor parameters to cloud.
NFR-5	**Scalability**	IoT enabled smart farming system can be automated autonomously without farmers input and disease detection can be implemented using OpenCV.

5. PROJECT DESIGN

5.1 Data Flow Diagram

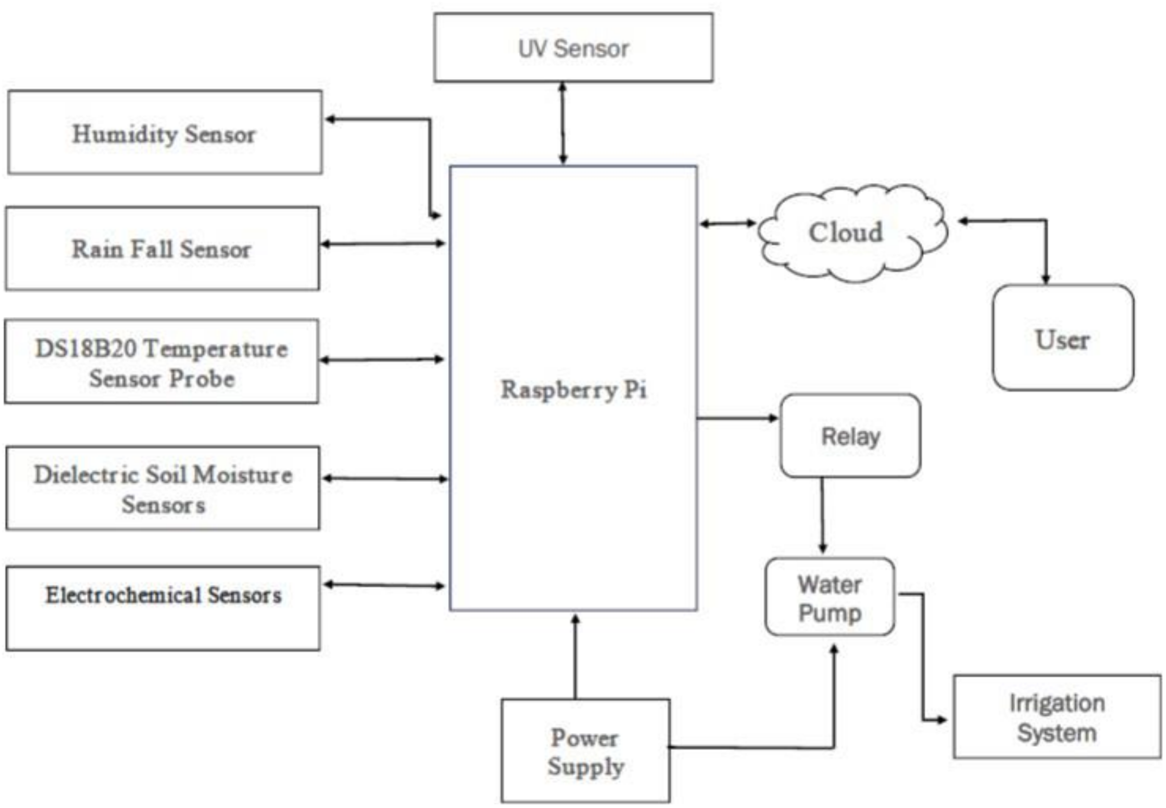
A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships



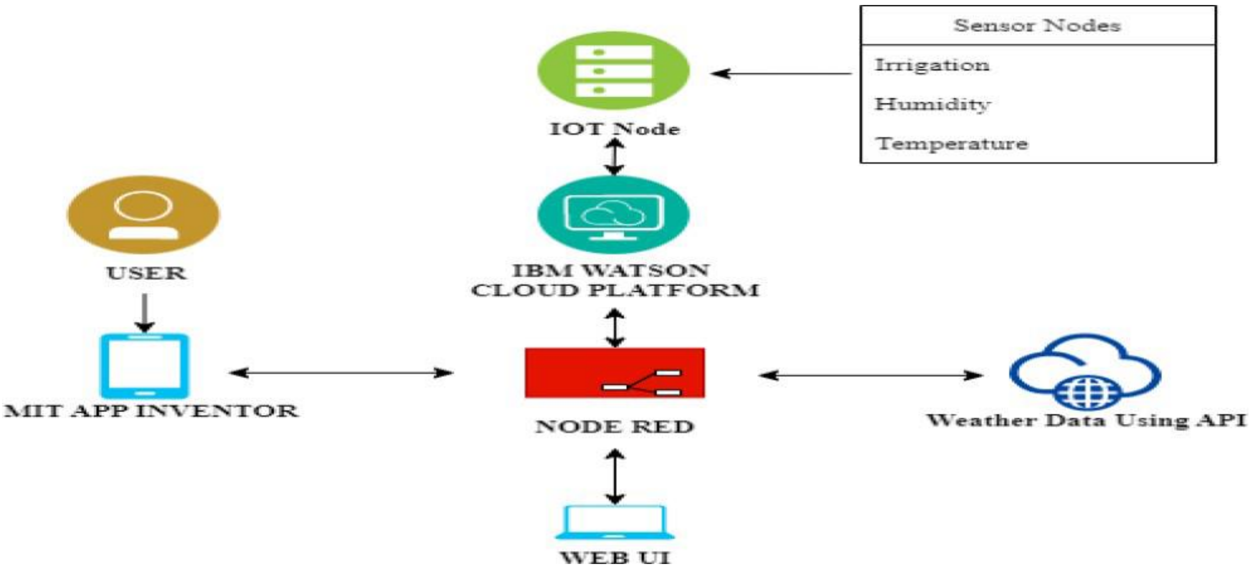
5.2 Solution & Technical Architecture Solution

IoT-based agriculture system helps the farmer to monitoring different parameters of his field like soil moisture, temperature, and humidity using some sensors by using a web or mobile application when the farmer is not near his field, he 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. 9 Technical Architecture Diagram

Solution Architecture Diagram:



Technical Architecture Diagram



5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
farmer (Mobile app)	displaying sensor parameters	USN-1	farmer can view temperature, humidity and soil moisture in his mobile connected to ibmcloud	displaying sensor parameters	High	Sprint-1
farmer (Mobile app)	controlling irrigation	USN-2	after seeing the sensor parameters farmer can turn on or off the irrigation system(motor)using mobile phone	controlling irrigation system	High	Sprint-1
raspberry pi	microcomputer setup in farm field	USN-3	temperature sensor, humidity sensor, soil moisture sensor and irrigation system is interface with raspberry pi which is connectedto IBM cloud	smart farming system is setup in farm field	high	Sprint-2
IBM cloud	Iot(data transfer)	USN-4	raspberry pi is connected to IBM cloud to monitor and control farm field remotely using internet	data exchange using internet	Medium	Sprint-1

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Interfacing Sensors and Motor Pump and IBM cloud	USN-1	Develop a python code to Interface Sensors, Motor Pump and IBM cloud.	20	High	Annamalai K
Sprint-2	Node-Red	USN-2	Develop a web Application Using a Node-Red.	20	High	Annamalai K
Sprint-3	Mobile Application	USN-3	Develop a mobile Application using MIT-App Inventor.	20	High	Janeshwar S
Sprint-4	Integration & Testing	USN-4	Integrating Python Script, Web application & Mobile App	20	Medium	Janeshwar S

6.2 Sprint Delivery Schedule

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	11 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	17 Nov 2022

7. CODING & SOLUTION

7.1 Feature

1 •We Added Weather Map Parameter like (Temperature, Pressure, Humidity) of Farmer’s Location, that is Displayed in Mobile Application & WEB UI

• Python Code

python code with cmments.py - C:\Users\B.SOMESHWARAN\Desktop\IBM\Project Development Phase\sprint -1\python code with cmments.py (3.8.10)

File Edit Format Run Options Window Help

```
#IBM Watson IOT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
import requests, json

ms=0
# Enter your API key here
api_key = "a0db30a689a774b93ffcb58ef2eddfda"
# base_url variable to store url
base_url = "http://api.openweathermap.org/data/2.5/weather?"
# Give city name
city_name = 'Chennai, IN'
# complete_url variable to store
# complete url address
complete_url = base_url + "appid=" + api_key + "&q=" + city_name

status='motor off'
myConfig = {
    "identity": {
        "orgId": "17lsro",
        "typeId": "MyDeviceType",
        "deviceId":"12345"
    },
    "auth": {
        "token": "GkatKdiUS?UVHKvnAD"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']
    if(m=="MOTOR ON"):#if motor is on
        print("MOTOR  IS ON")
        global status
        status='motor on'
        myData={'temperature':temp, 'humidity':hum,'soilmoisture':sm_percentage,'status':status,'api_temperature':api_temperature,'api_pressure':api_pressure
        client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
        print("Published data Successfully: %s", myData)
```

```
        time.sleep(2)


client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    # get method of requests module
    # return response object
    response = requests.get(complete_url)
    # json method of response object
    # convert json format data into
    # python format data
    x = response.json()
    # Now x contains list of nested dictionaries
    # Check the value of "cod" key is equal to
    # "404", means city is found otherwise,
    # city is not found
    if x["cod"] != "404":




        y = x["main"]


        api_temperature = y["temp"]#getting api temperature data


        api_pressure = y["pressure"]#getting api pressure data


        api_humidity = y["humidity"] #getting api humidity data


        z = x["weather"]


        api_weather_description = z[0]["description"]#getting api weather condition data
```

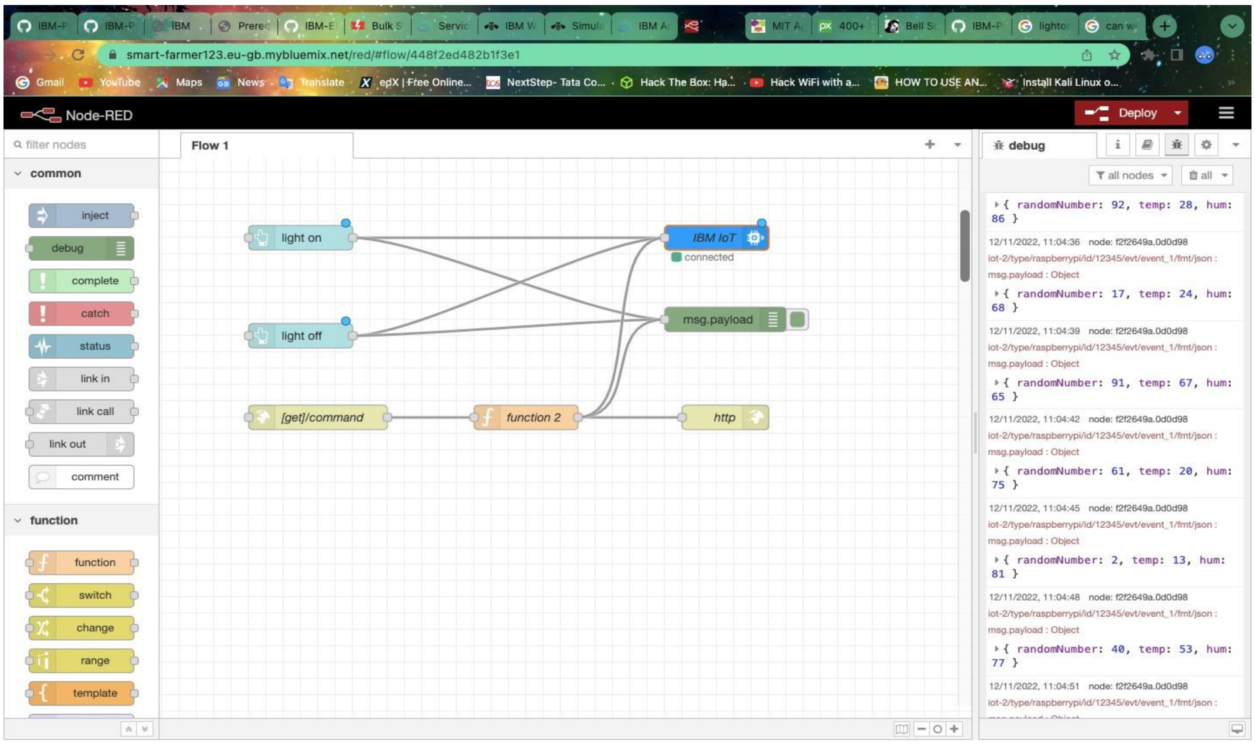
```
        api_weather_description = z[0]["description"]#getting api weather condition data

    temp=random.randint(-20,125)#geneating ranom values for temperature
    hum=random.randint(0,100)#geneating ranom values for humidity
    soilmoisture=random.randint(0,1023)#analog sensor
    sm_percentage=(soilmoisture/1023)*100
    sm_percentage=int(sm_percentage)#geneating ranom values for soilmoisture
    myData={'temperature':temp, 'humidity':hum,'soilmoisture':sm_percentage,'status':status,'api_temperature':api_temperature,'api_pressure':api_pressure,'ap
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)

time.sleep(2)
client.disconnect()
```

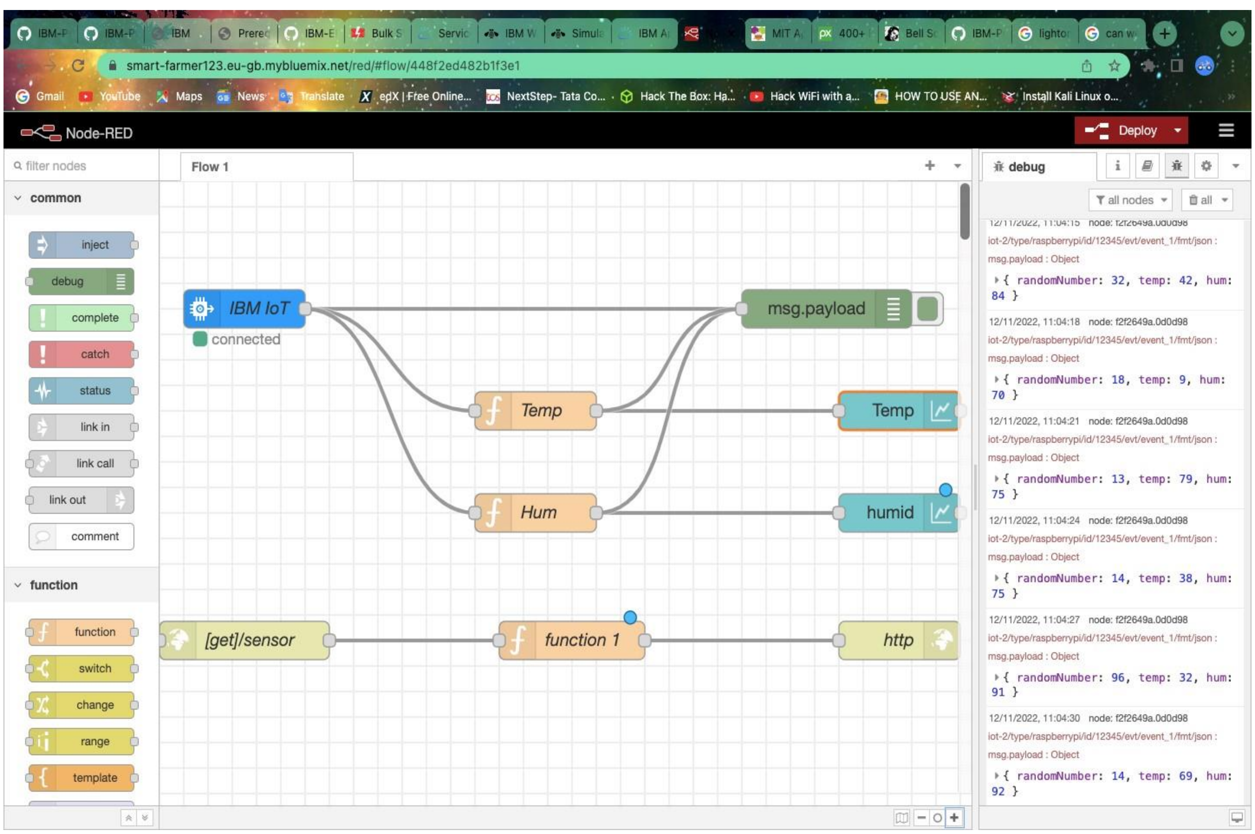
8.TESTING

8.1 Testing Output of Python Code

```
*IDLE Shell 3.8.10*
File  Edit  Shell  Debug  Options  Window  Help
Published data Successfully: %s {'temperature': 73, 'humidity': 74, 'soilmoistur
e': 53, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 96, 'humidity': 96, 'soilmoistur
e': 51, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 25, 'humidity': 30, 'soilmoistur
e': 74, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 120, 'humidity': 17, 'soilmoistu
re': 84, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
 'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 26, 'humidity': 61, 'soilmoistur
e': 29, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 74, 'humidity': 25, 'soilmoistur
e': 14, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': -11, 'humidity': 59, 'soilmoistu
re': 12, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
 'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 22, 'humidity': 78, 'soilmoistur
e': 26, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 71, 'humidity': 23, 'soilmoistur
e': 4, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012, '
api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 16, 'humidity': 38, 'soilmoistur
e': 57, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 53, 'humidity': 58, 'soilmoistur
e': 74, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 58, 'humidity': 79, 'soilmoistur
e': 84, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Published data Successfully: %s {'temperature': 25, 'humidity': 17, 'soilmoistur
e': 65, 'status': 'motor off', 'api_temperature': 299.14, 'api_pressure': 1012,
'api_humidity': 73, 'api_weather_description': 'haze'}
Ln: 5  Col: 0
```

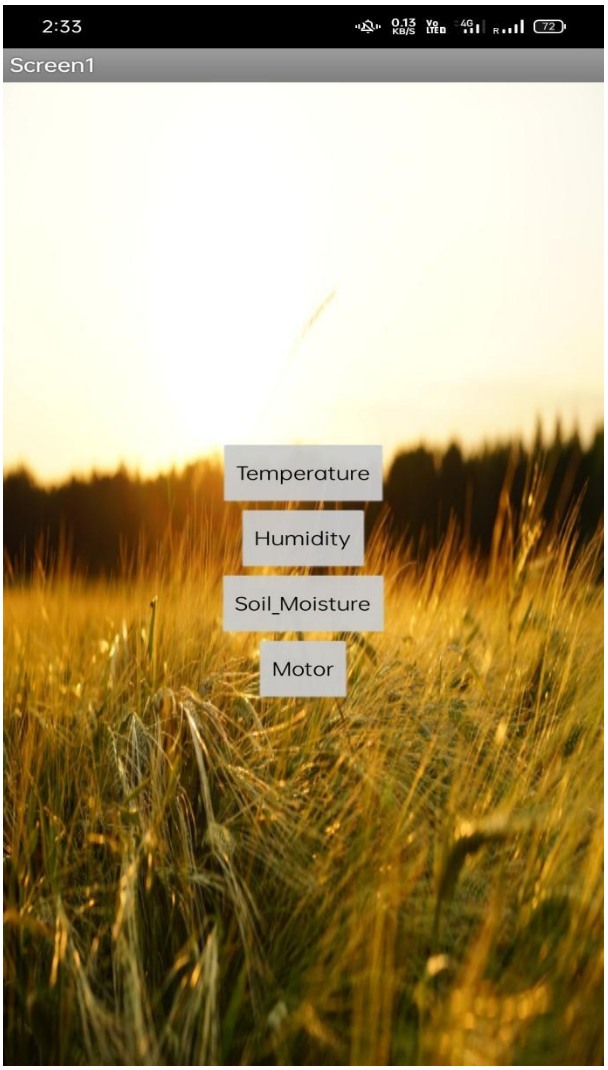
Node Red Connected and Publishes the Value

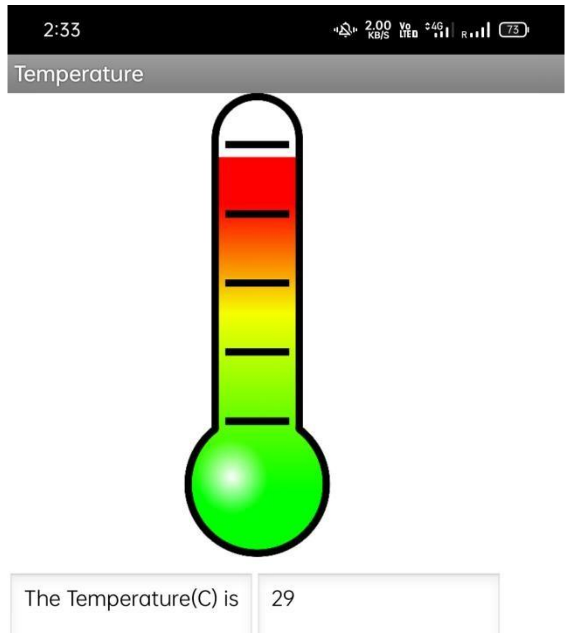


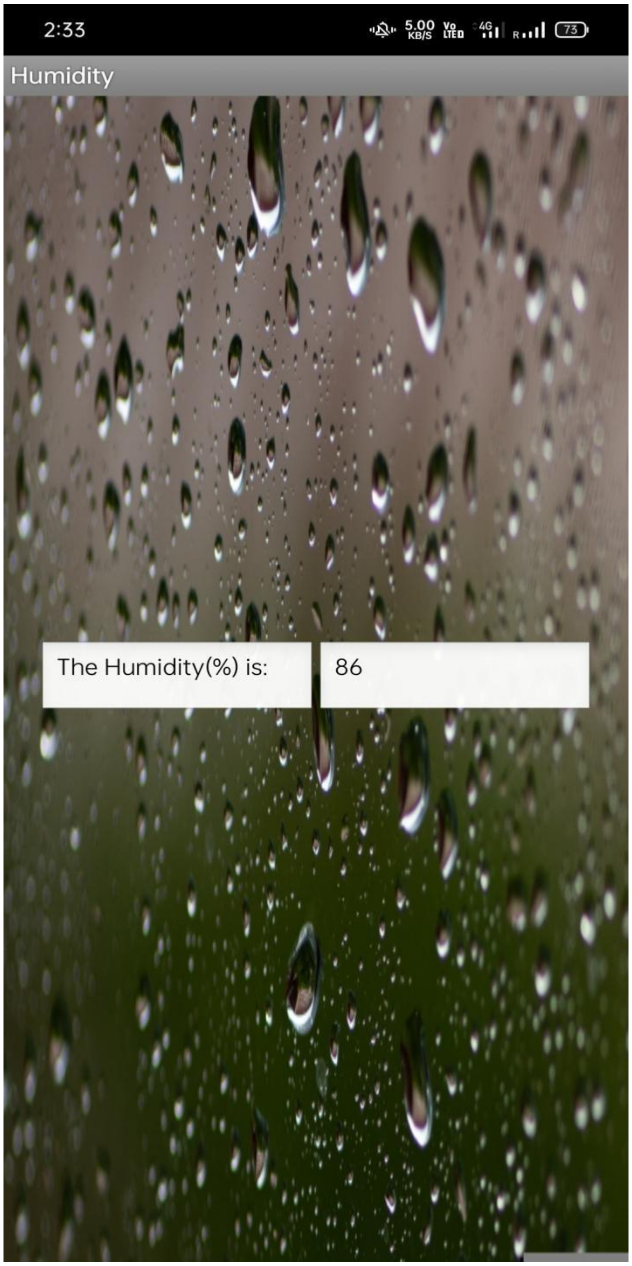


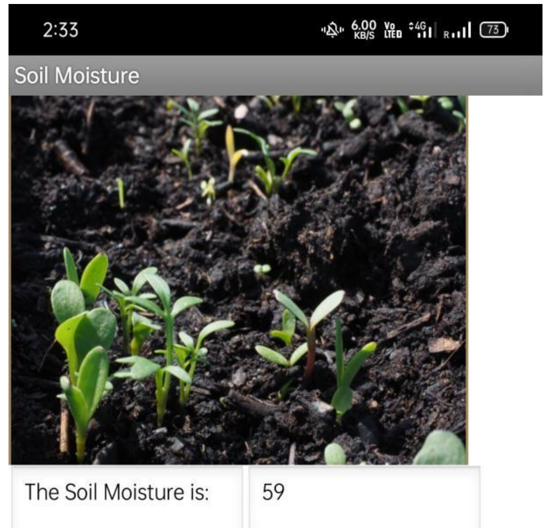
8.2 User Acceptance Testing

The Output Live Data is Show In Mobile Application





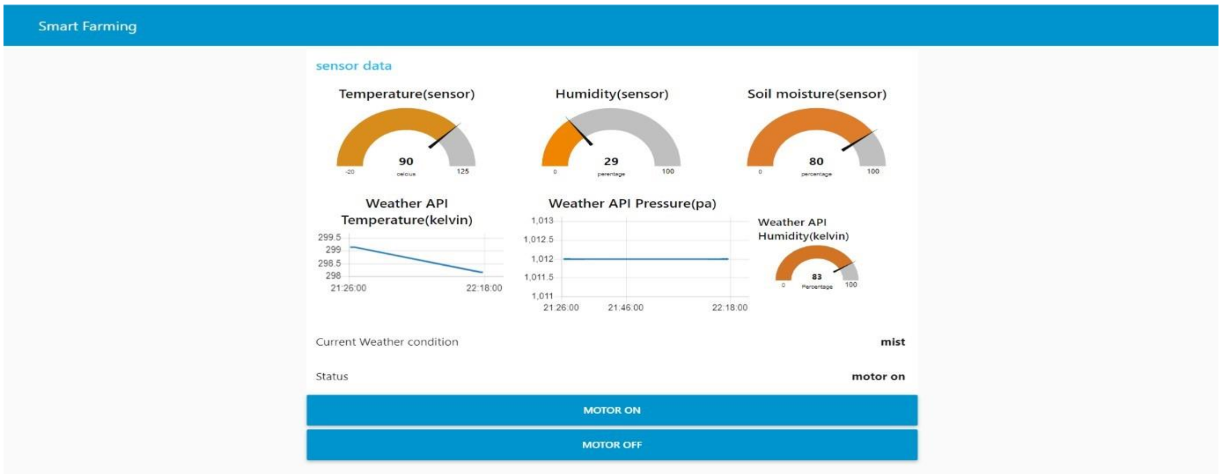

2:33
Humidity
The Humidity(%) is:
86


2:33
Soil Moisture
The Soil Moisture is:
59


2:33
Motor
Motor ON/OFF

9.RESULTS

9.1 Performance Metrics



10. ADVANTAGES AND DISADVANTAGES

Advantages:

• Farms can be monitored and controlled remotely.

• Increase in convenience to farmers.

• Less labor cost.

• Better standards of living.

Disadvantages:

• Lack of internet/connectivity issues.

• Added cost of internet and internet gateway infrastructure.

• Farmers wanted to adapt the use of Mobile App. 19 11.

11.CONCLUSION

Smart Agriculture System Based on Internet of Things can deliver the farmer all the required information like temperature, humidity, soil moisture of the crop in real time and also the weather forecast at fingertips. Also, instead of using manual based Motor control, the farmer can do this remotely anywhere as long as he's connected to network. To make this possible we have used IBM Cloud Platform, Watson IoT Platform, Open weather API and Node-red to gather and show the information on Web Application.

By using a Python Script, we were able to subscribe to IBM platform to send and receive commands to motor for controlling it. Using this Smart Agriculture System the farmer can not only monitor all the required data in real time but also can make smart decisions for better yield based on the data collected. In this way he can produce yield effectively and also earn profitably more based on accurate data received.

12. FUTURE SCOPE

Future scope of this smart agriculture system will be to add more sensors to the existing micro controller, to add increase the current functionality or to do more automated tasks like automatic watering system, adding pest control information and geo tagging the farm etc. This information can be shared on consent to Government authorities or Private companies for more suggestions of better techniques remotely. As the data stored can be used for reference and analysis which can be very helpful in future.

13. APPENDIX

Source Code

```
import time
 import sys
import ibmiotf.application
 import ibmiotf.device
 import random
#IBM
 organization = "farmer
" deviceType = "raspberrypi"
 deviceId = "12345"
 authMethod = "use-token-auth"
 authToken = "12345678"
 #Gpio
 def mycommandCallback(cmd)
: print("Command Received: %s" %cmd.data['command'])
 status = cmd.data['command']
if status=="lighton":
 print("LED is ON")
 elif status=="lightoff":
 print("LED 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

deviceCli.connect() while True: t

emp=random.randint(0,100)

hum=random.randint(0,100)data={'temp':temp,'hum':hum}

def myOnPublishCallback():

print("Published Temperature = %s C"%temp,"Humidity = %s %%" %hum, "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

deviceCli.disconnect()

GitHub GitHub Link: https://github.com/IBM-EPBL/IBM-Project-51081-1660970616

BIBLIOGRAPHY

IBM cloud reference: https://cloud.ibm.com/
Open Weather: https://openweathermap.org/