PROJECT REPORT

Date	19 November 2022
Team ID	PNT2022TMID45187
Project Name	SMARTFARMER – IoT ENABLED SMART FARMING APPLICATION

1.INTRODUCTION

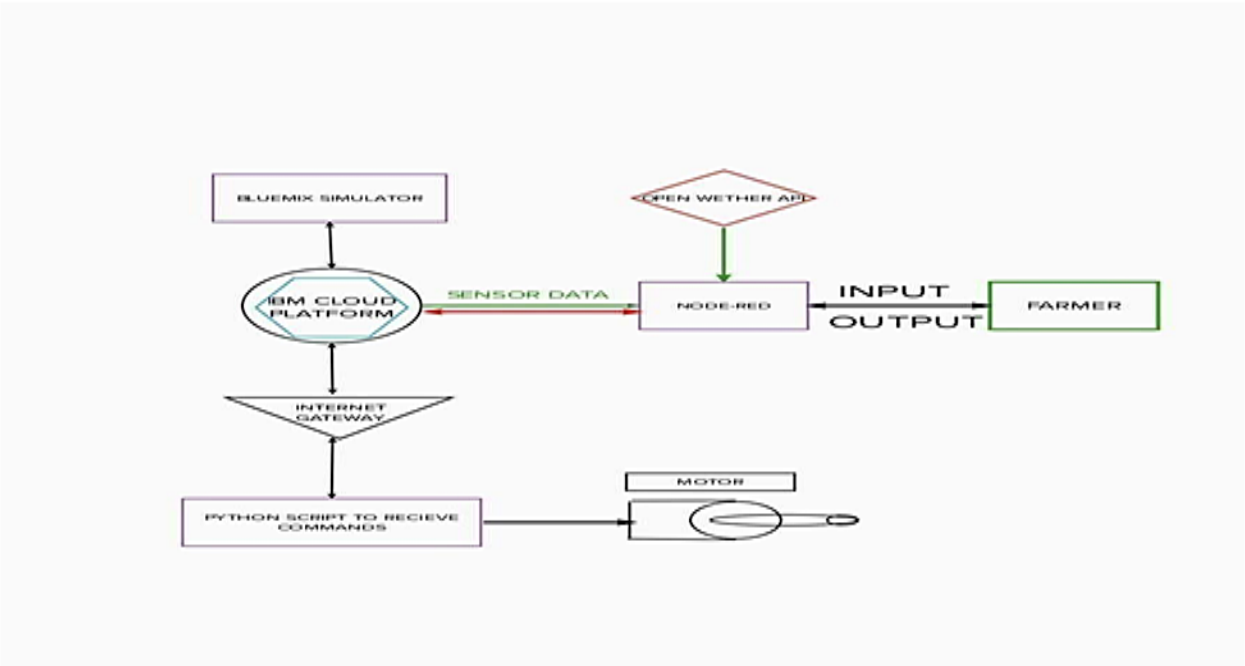
The main aim of this project is to help farmers automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity and etc and control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

1.1PROJECT OVERVIEW

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.

BLOCK DIAGRAM:

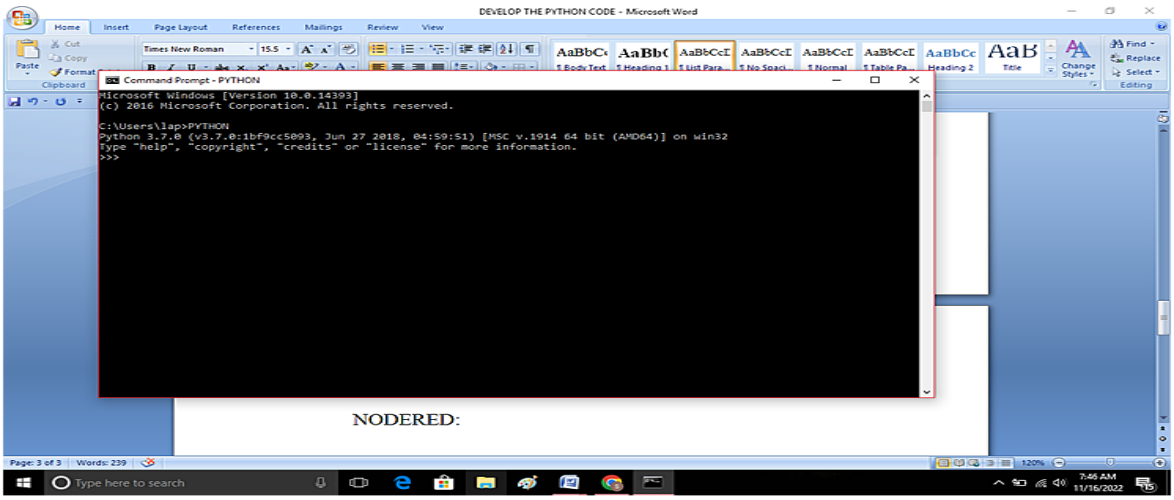
In order to implement the solution , the following approach as shown in the blockdiagram is used.



PREREQUISITES:

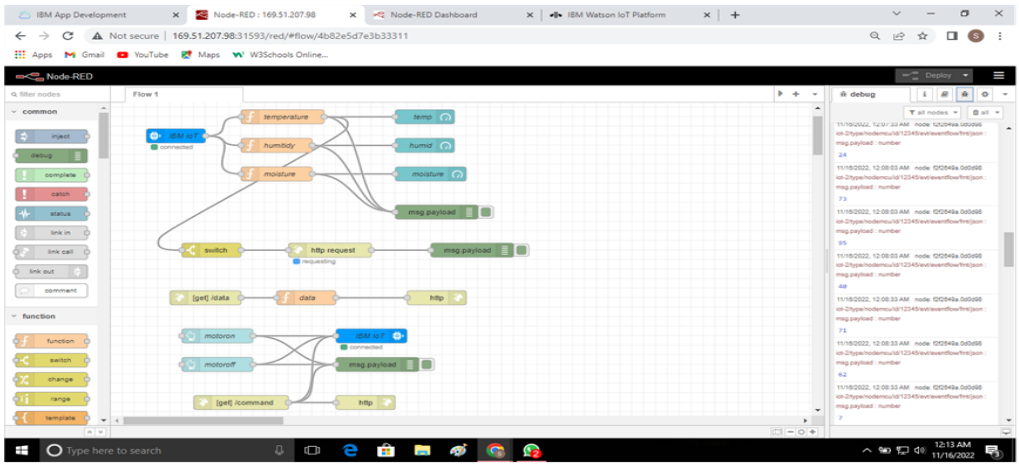
1.Python IDE

Install python 3.7.0 compiler to execute pythonscripts.



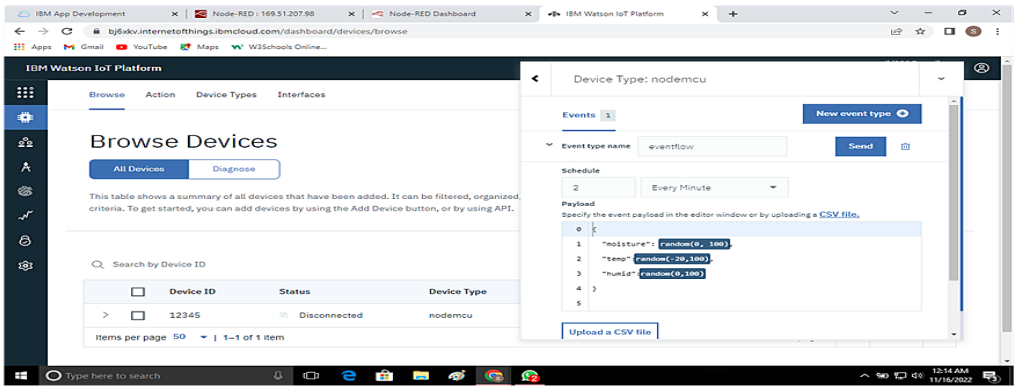
2.NODERED

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internetof Things. Node-REDprovides a web browser-based flow editor, which can be used to create JavaScript functions.

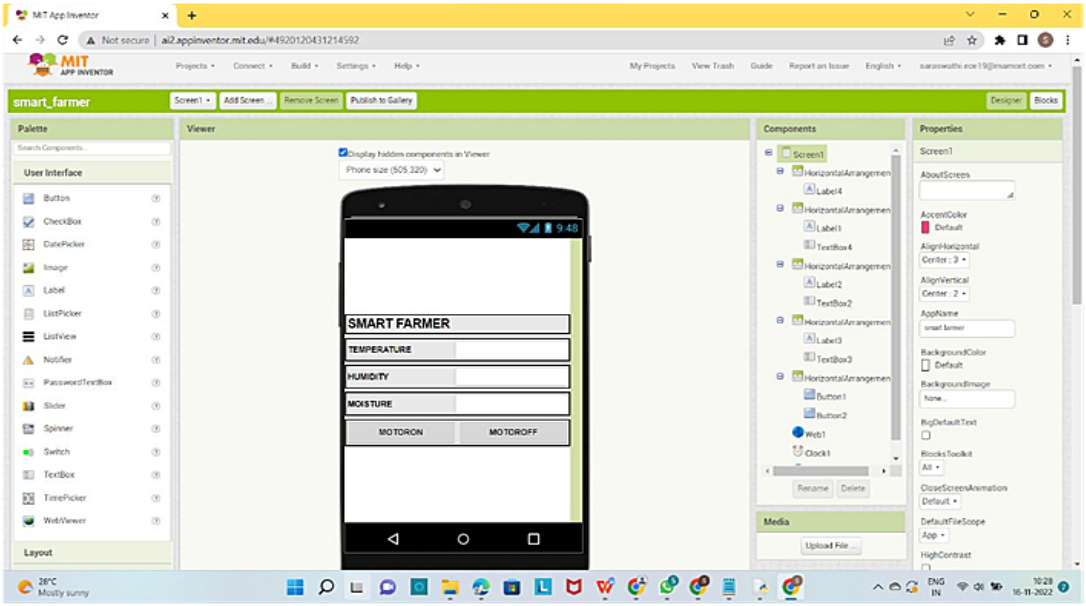


3.IBM Watson IoT Platform

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage.IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.



4. MIT APP INVENTOR:



1.2 PURPOSE

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.

2.LITERATURE SURVEY

2.1 Existing problem

The challenges of a smart agriculture system include the integration of these sensors and tying the sensor data to the analytics driving automation and response activities. When integrated, the use of data analytics can reduce the overall cost of agriculture and contribute to higher production from the same amount of area through precise control of water, fertilizer and light. Smart methods allow for farming on smaller and more distributed lands through

remote monitoring, whether indoor or outdoor.

To successfully deploy a smart agriculture system, consider setting up a communications network that can integrate a limited number of sensors across a large area of farmland. This will require third-party network provisioning or setting up a private network consisting of access points and uplinks to a private backhaul network, which channels all the data traffic to centralized monitoring software or an analytics head-end system .

• It is not a secure system.
• There is no motion detection for protection of agriculture field.
• Automation is not available.

2.2 References

[1] ISSN No:-2456-2165 Volume 4, Issue 2 Feb – 2019: "Solars' Energy: - A safe and reliable, eco-friendly and sustainable Clean Energy Option for Future India: - A Review."

[2] Universal Paper of advanced science and science and exploration technology. [2] GRD Journals- Global Research and Development Journal for Engineering | Volume 4 | Issue 3 | February (2019) ISSN: 2455-5703 “Design and Implementation of an Advanced Security System for Farm Protection from Wild Animals”.

[3] International Journal of Innovationsin Engineering and Science, Impact Factor Value 4.046 e-ISSN: 2456-3463 Vol.4, No. 5, 2019 “Solar Powered Smart Fencing System for Agriculture Protection using GSM & Wireless Camera”.

[4] International Journal of Management, Technology And Engineering ISSN

NO : 2249-7455 Volume 8, Issue VII, JULY/2018”Protecting Crops From Birds, Using Sound Technology In Agriculture” .

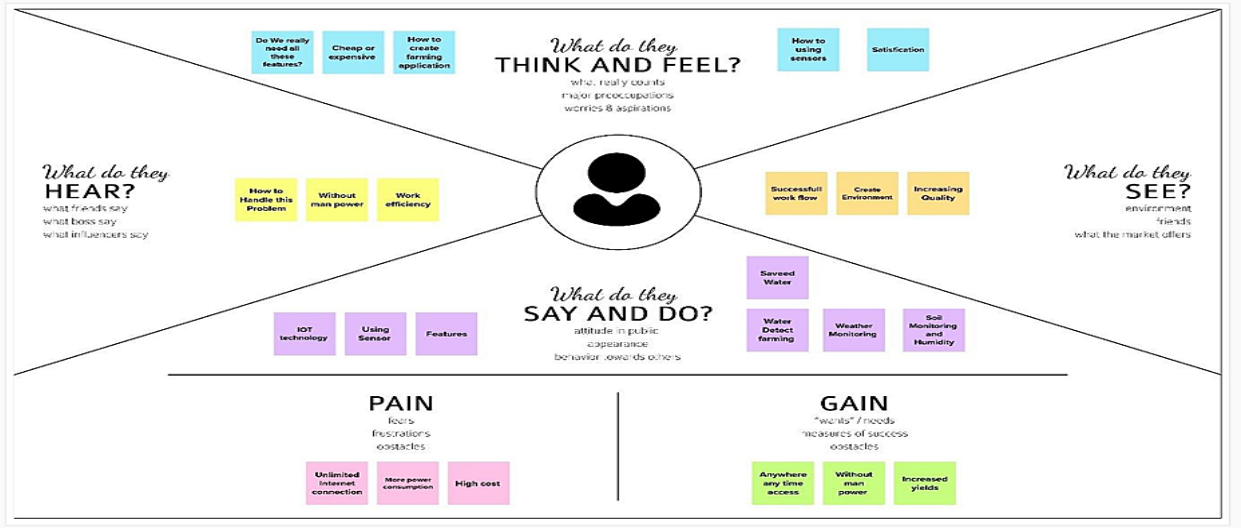
[5] American Journal of Engineering Research (AJER)2018 eISSN: 2320-0847 pISSN : 2320- 0936 Volume-7, Issue-7, pp-326-330 “Moisture Sensing Automatic Plant Watering System Using Arduino Uno”.

2.3 PROBLEM STATEMENT

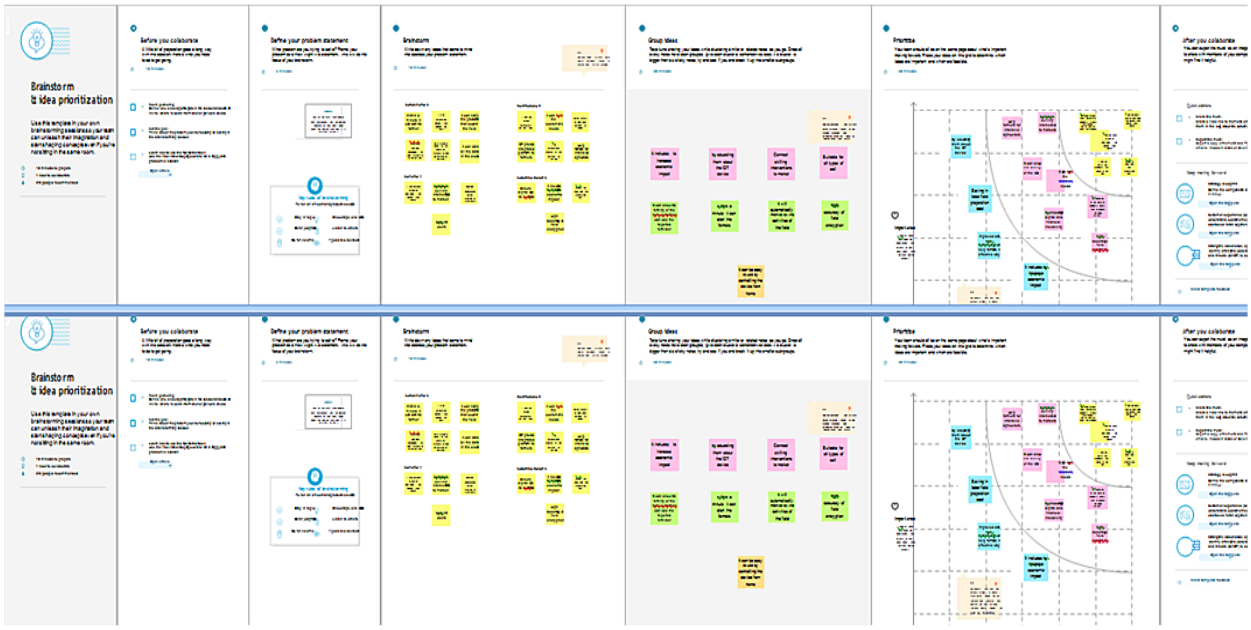
Farmers are to be present at farm for its maintenance irrespective of weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically. Farmer have to stay most of the time in field in order to get a good yield. In difficult times like in the presence of pandemic also they have to work hard in their fields risking their lives to provide food for the country.

3.IDEATION &PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation &Brainstorming

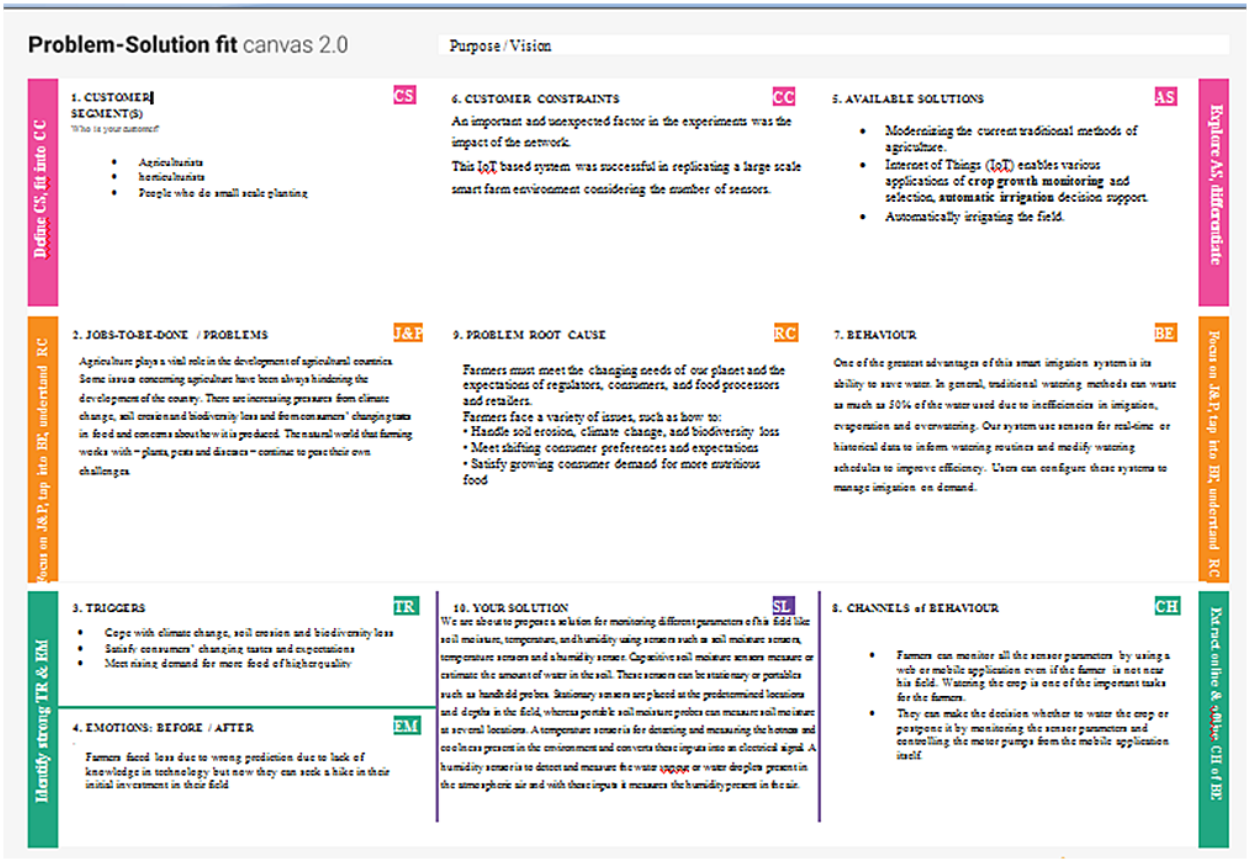


tion

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To make farming easier by choosing several constraints in agriculture and to overcome those constraints, to increase production quality and quantity using IOT.
2.	Idea / Solution description	Using smart techniques like monitoring farms climate, smart irrigation and soil analysis.
3.	Novelty / Uniqueness	Solar power smart irrigation system which helps you to monitor temperature, moisture ,humidity using smart sensors.

4.	Social Impact / Customer Satisfaction	It is better than the present modern irrigation system by using this method we can control soil erosion. There will be better production yield.
5.	Business Model (Revenue Model)	As the productivity increases customer satisfaction also increases and hence need for the application also increases, which increases the revenue of the business.
6.	Scalability of the Solution	It is definetly scalable we ca increase the constraints when the problem arises.

3.4 Problem Solution fit



4.REQUIREMENT ANALYSIS

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	IoT devices	Sensors and Wifi module.
FR-2	Software	Web UI, Node-red, IBM Watson, MIT app

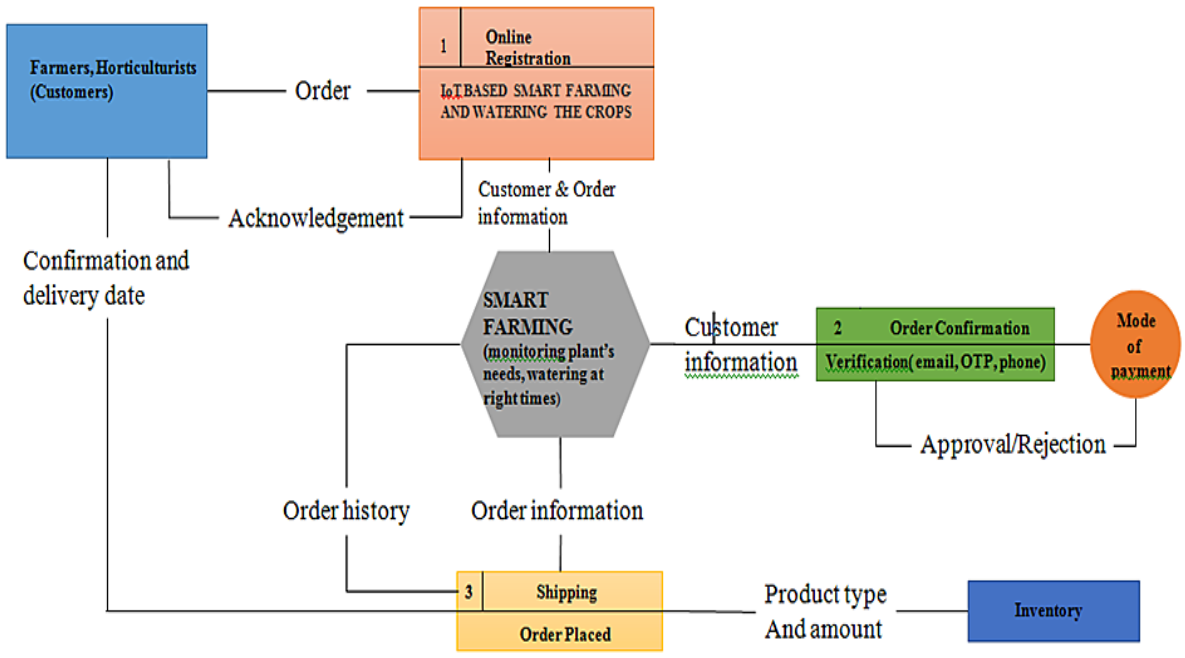
Non Functional Requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Have an easy-to-understand guidebook. Simpler to use The product is easy to use even by farmers who are illiterate.
NFR-2	**Security**	Application security requires two-step authorization. The user's needs will determine how passwords and passkeys are assigned.
NFR-3	**Reliability**	Hardware needs to be checked and maintained regularly. Periodic software updates are possible. Any system breakdown will result in an immediate alarm.
NFR-4	**Performance**	The programme needs a good user interface. It should just demand a small amount of energy. It must conserve energy and water.

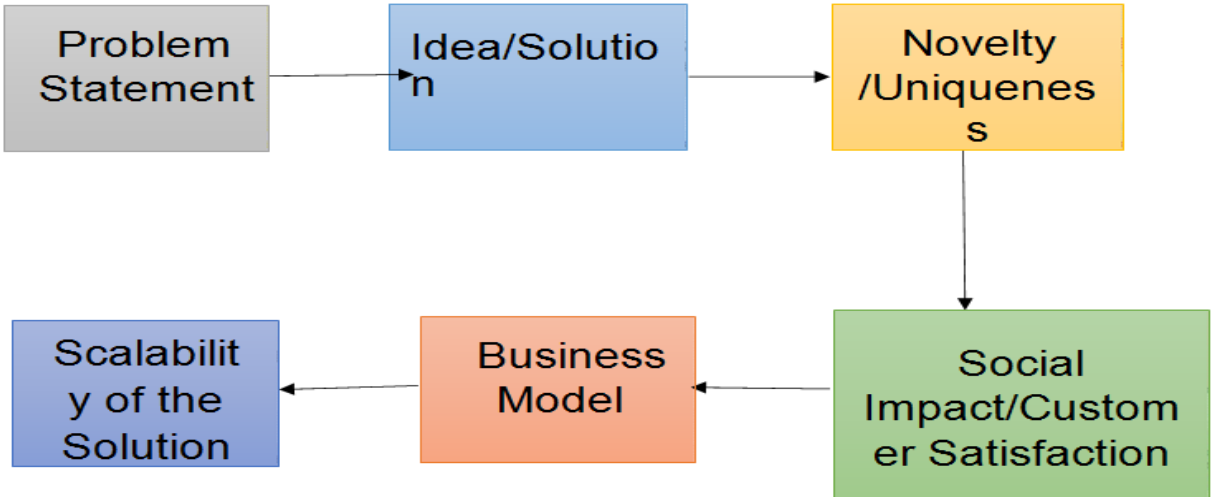
NFR-5	**Availability**	Every function will be accessible whenever the user needs it. It depends on the farmer's requirements and the level of customization the user has undertaken.
NFR-6	**Scalability**	Regardless of the size or area of a farm field, the product must cover the entire area of the ground.

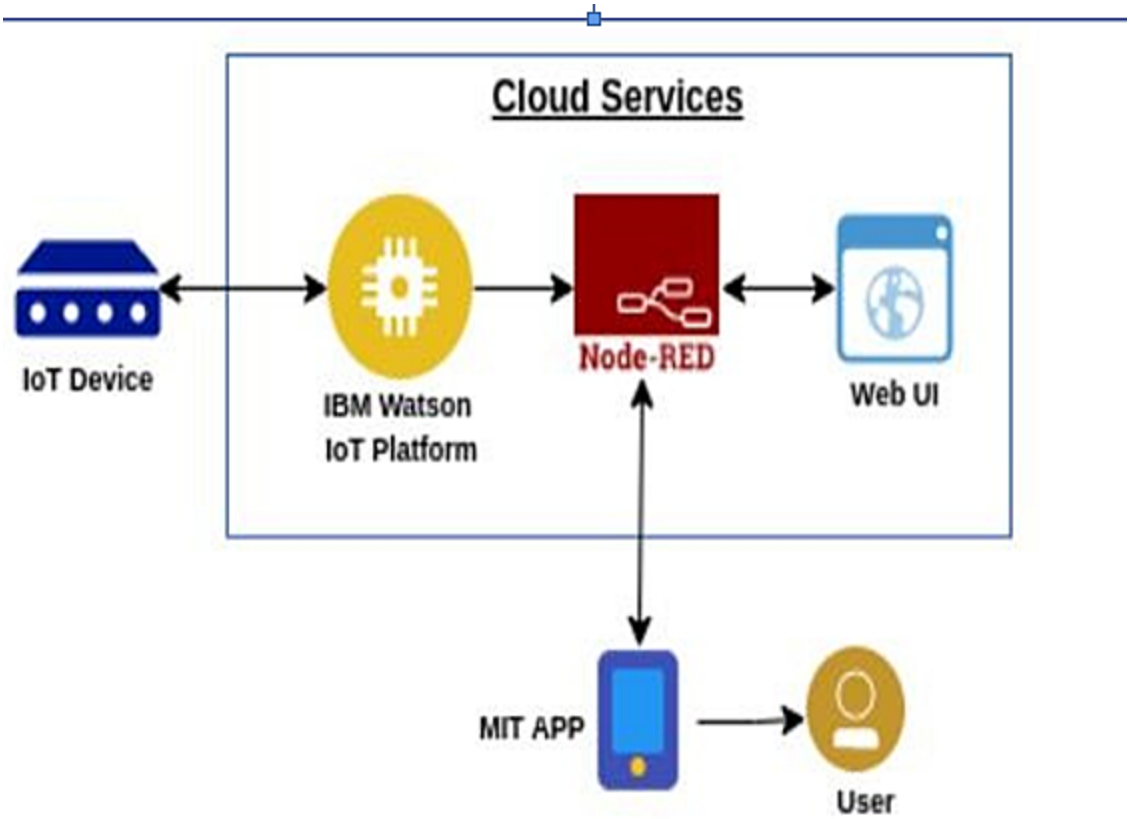
5.PROJECT DESIGN

5.1 Data Flow Diagram



5.2 Solution & Technical Architecture





5.3 USER STORIES

User Story / Task
Connect Sensors and Arduino with python code
Creating device in the IBM Watson IoT platform, workflow for IoT scenarios using Node-Red
Develop an application for the Smart farmer project using MIT App Inventor
Design the Modules and test the app
To make the user to interact with software

6.PROJECT PLANNING AND SCHEDULING

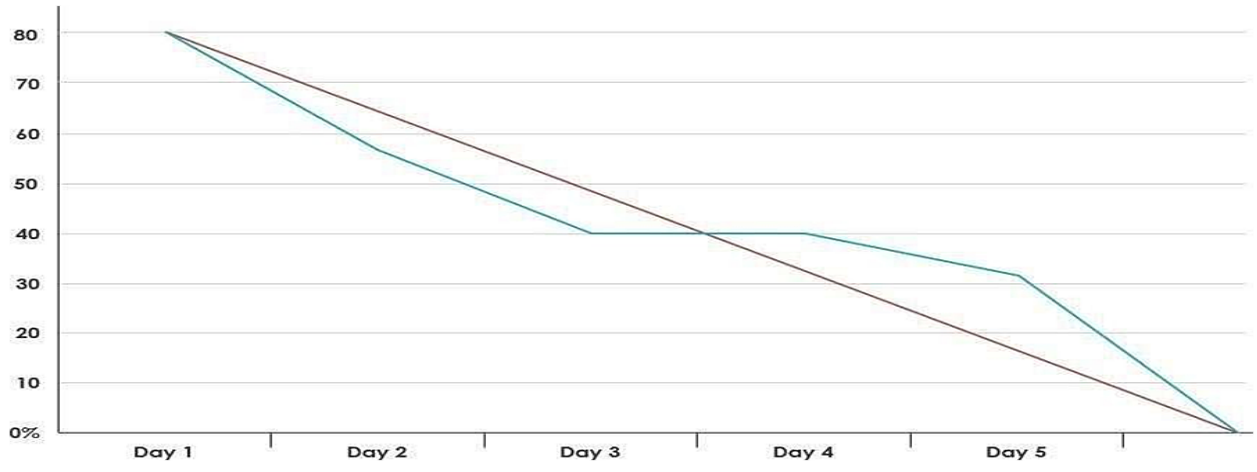
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Simulation Creation	USN-1	Connect Sensors and Arduino with python code	2	High	Saraswathi, Ramprakash, Suwathi, Sabarishkumar
Sprint-2	Software	USN-2	Creating device in the IBM Watson IoT platform, workflow for IoT scenarios using Node-Red	2	High	Saraswathi, Ramprakash, Suwathi, Sabarishkumar

Sprint-3	MIT App Inventor	USN-3	Develop an application for the Smart farmer project using MIT App Inventor	2	High	Saraswathi, Ramprakash, Suwathi, Sabarishkumar
Sprint-3	Dashboard	USN-3	Design the Modules and test the app	2	High	Saraswathi, Ramprakash, Suwathi, Sabarishkumar
Sprint-4	Web UI	USN-4	To make the user to interact with software	2	High	Saraswathi, Ramprakash, Suwathi, Sabarishkumar

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	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022



7.CODING &SOLUTIONING

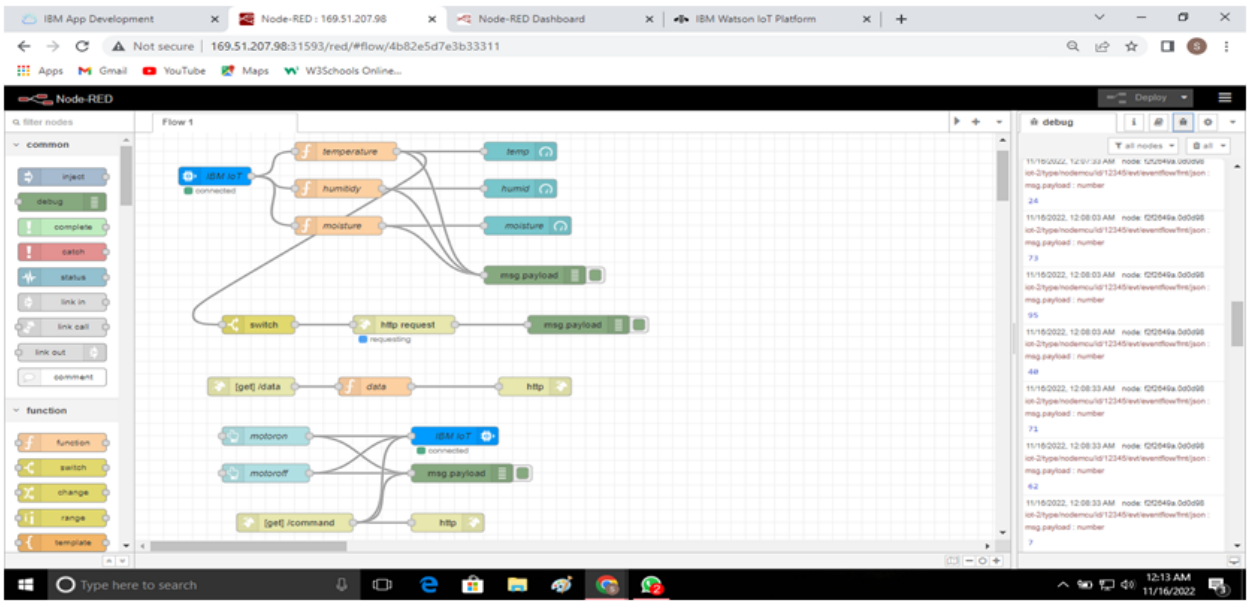
```
import wiotp.sdk.device
import time
import os
import datetime
import random
myConfig = {
  "identity": {
     "orgId": "bj6xkv",
     "typeId": "nodemcu",
     "deviceId": "12345"
  },
  "auth": {
     "token": "12345678"
  }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect ()

def myCommandCallback (cmd) :
    print ("Message received from IBM IoT Platform: %s"
%cmd.data['command'])
    m=cmd.data['command']
    if (m=="motoron"):
        print ("Motor is switched on")
    elif (m=="motoroff"):
        print ("Motor is switched OFF")
    print (" ")
```

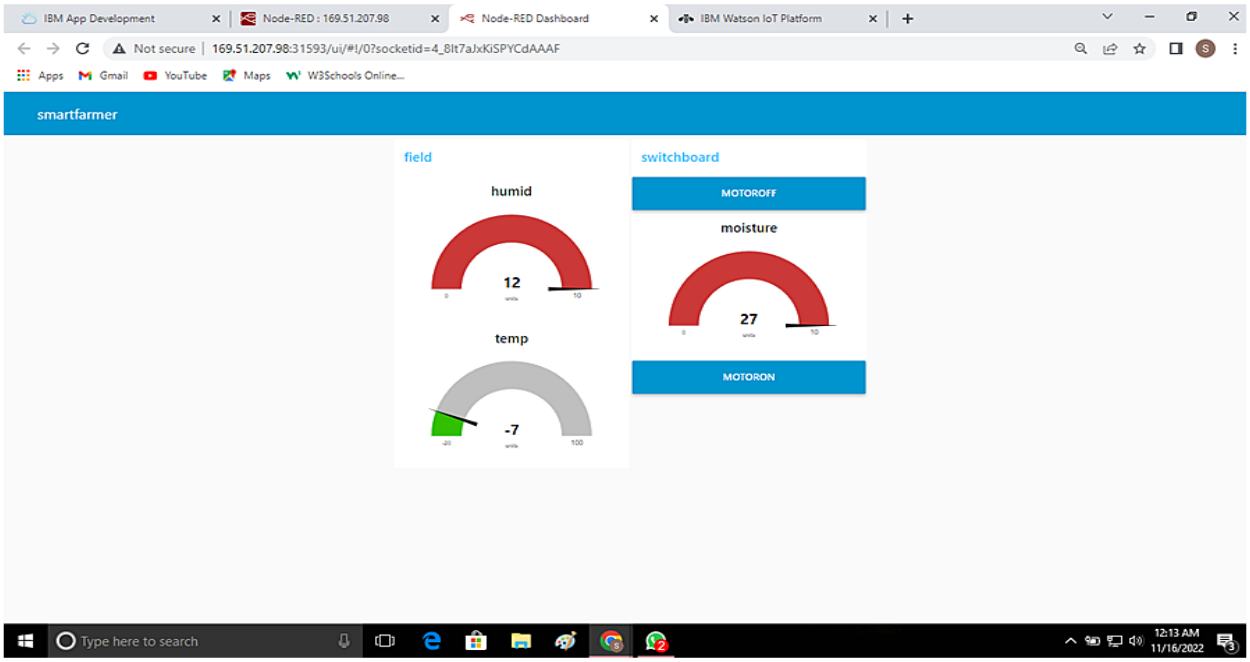
```
while True:
    soil=random.randint (0,100)
    temp=random.randint (-20, 125)
    hum=random.randint (0, 100)
    myData={'soil moisture': soil, 'temperature':temp, 'humidity':hum}
    client.publishEvent (eventId="status", msgFormat="json",
    data=myData, qos=0 , onPublish=None)
    print ("Published data Successfully: %s", myData)
    time.sleep (2)
    client.commandCallback = myCommandCallback
client.disconnect ()
```

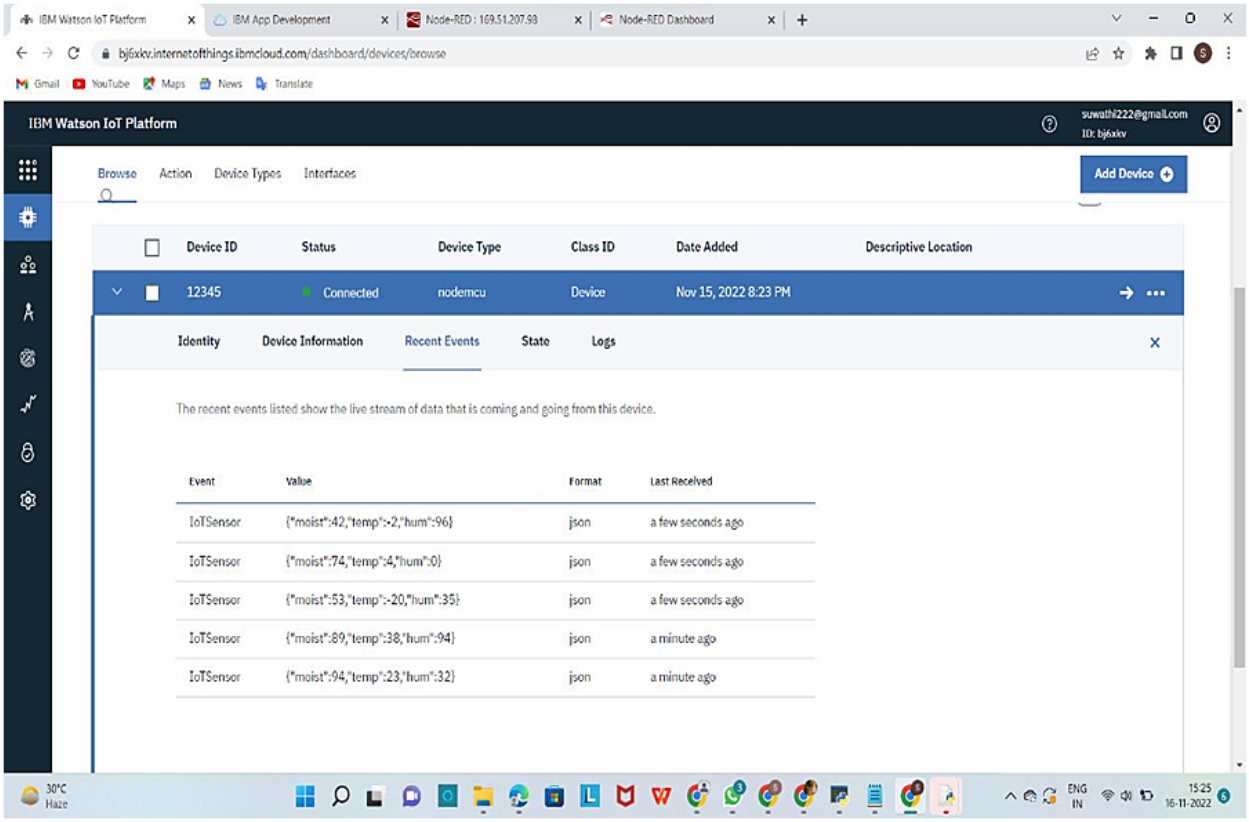
8.TESTING

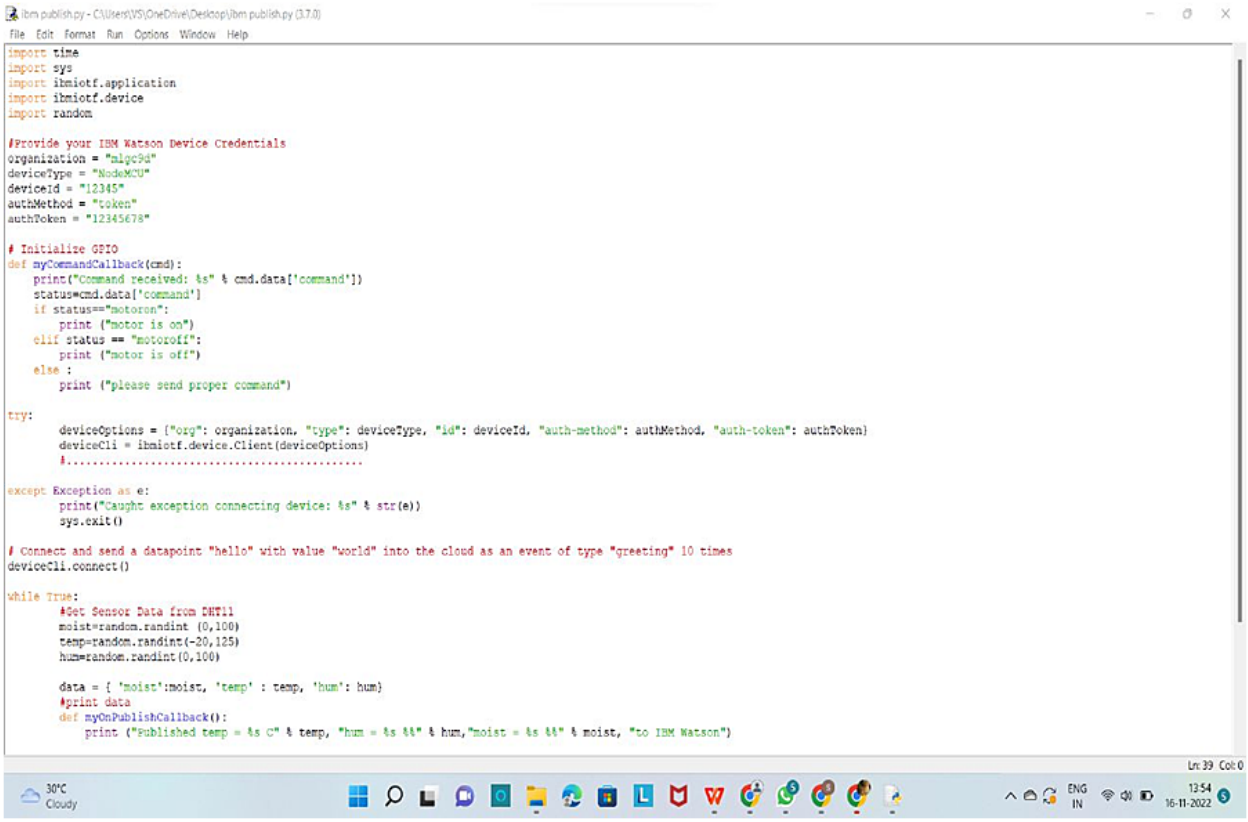
8.1 Test case

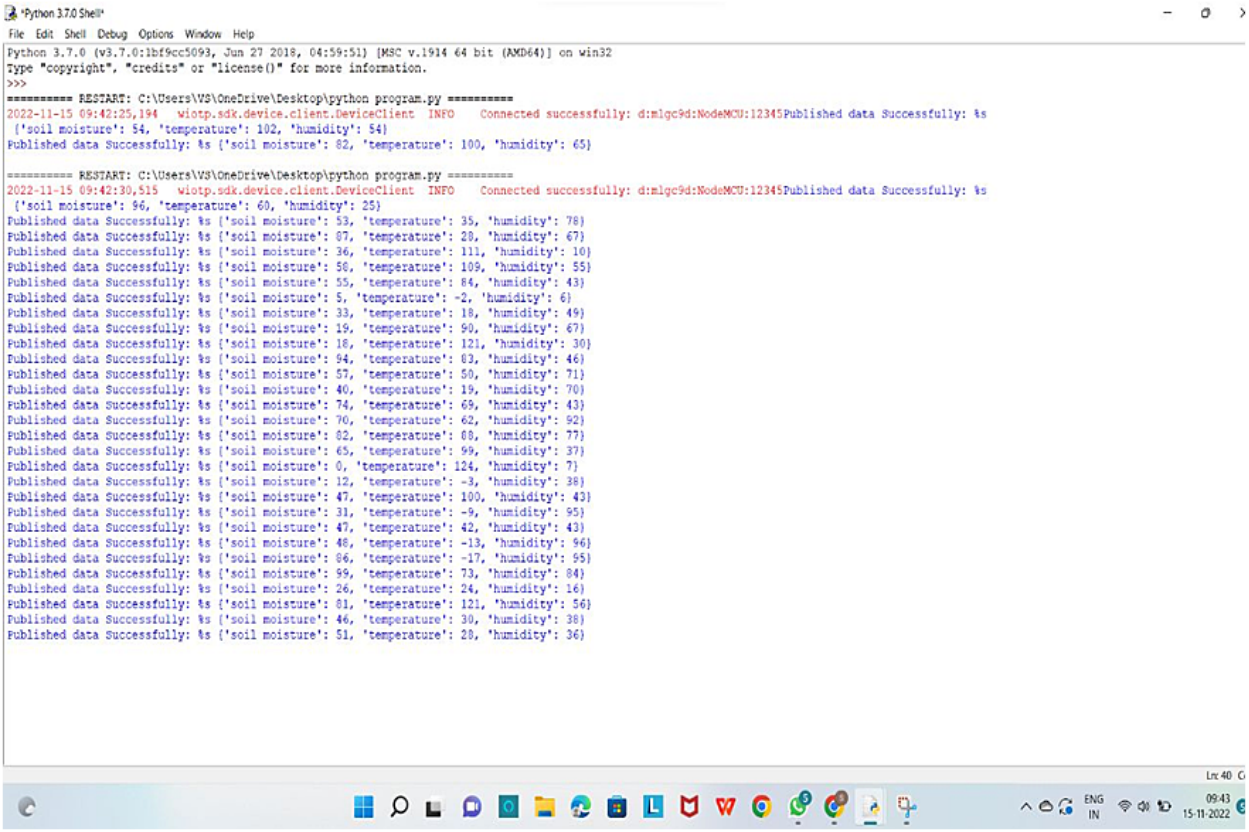
Web application using Node Red



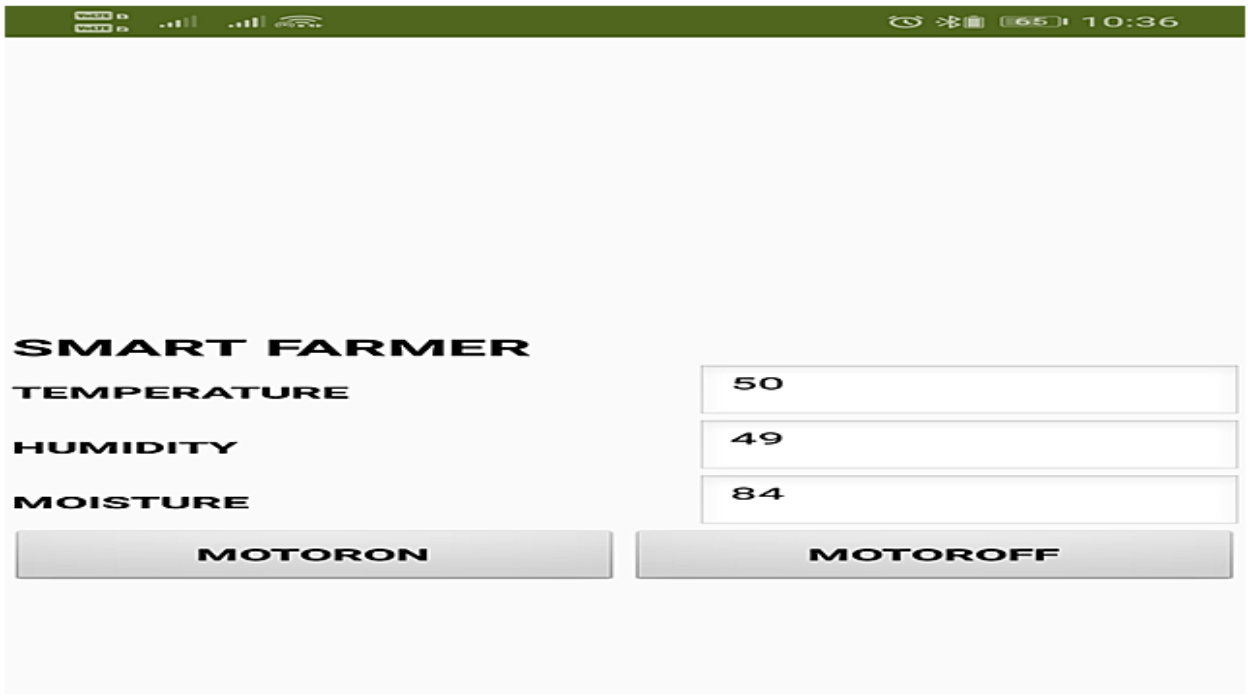

IBM App Development
Node-RED : 169.51.207.98
Node-RED Dashboard
IBM Watson IoT Platform
Not secure | 169.51.207.98:31593/ui/#!/0?socketid=4_8lt7aJxKiSPYCdAAAF
Apps
Gmail
YouTube
Maps
W3Schools Online...
smartfarmer
field
humid
12
temp
-7
switchboard
MOTOROFF
moisture
27
MOTORON
Type here to search
12:13 AM
11/16/2022


IBM Watson IoT Platform
IBM App Development
Node-RED : 169.51.207.98
Node-RED Dashboard
bj6xkv.internetofthings.ibmcloud.com/dashboard/devices/browse
Gmail
YouTube
Maps
News
Translate
IBM Watson IoT Platform
suwathi222@gmail.com
ID: bj6xkv
Browse
Action
Device Types
Interfaces
Add Device
Device ID
Status
Device Type
Class ID
Date Added
Descriptive Location
12345
Connected
nodemcu
Device
Nov 15, 2022 8:23 PM
Identity
Device Information
Recent Events
State
Logs
The recent events listed show the live stream of data that is coming and going from this device.
Event
Value
Format
Last Received
IoTSensor
{"moist":42,"temp":-2,"hum":96}
json
a few seconds ago
IoTSensor
{"moist":74,"temp":4,"hum":0}
json
a few seconds ago
IoTSensor
{"moist":53,"temp":-20,"hum":35}
json
a few seconds ago
IoTSensor
{"moist":89,"temp":38,"hum":94}
json
a minute ago
IoTSensor
{"moist":94,"temp":23,"hum":32}
json
a minute ago
30°C
Haze
ENG
IN
15:25
16-11-2022


ibm publish.py - C:\Users\VS\OneDrive\Desktop\ibm publish.py (3.7.0)
File Edit Format Run Options Window Help
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "mlgc9d"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
if status=="motoron":
print ("motor is on")
elif status == "motoroff":
print ("motor is off")
else :
print ("please send proper command")
try:
deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
deviceCli = ibmiotf.device.Client(deviceOptions)
#.....
except Exception as e:
print("Caught exception connecting device: %s" % str(e))
sys.exit()
# 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
moist=random.randint (0,100)
temp=random.randint(-20,125)
hum=random.randint(0,100)
data = { 'moist':moist, 'temp' : temp, 'hum': hum}
#print data
def myOnPublishCallback():
print ("Published temp = %s C" % temp, "hum = %s %%" % hum,"moist = %s %%" % moist, "to IBM Watson")
Ln: 39 Col: 0
30°C
Cloudy
ENG
IN
13:54
16-11-2022

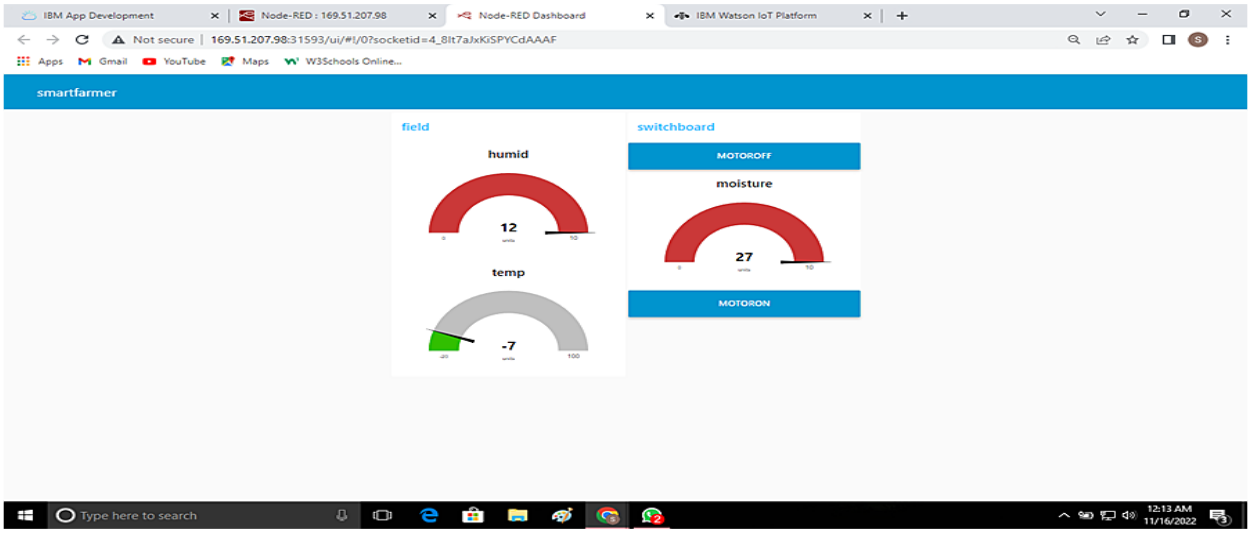

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\VS\OneDrive\Desktop\python program.py ==========
2022-11-15 09:42:25,194 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:mlgc9d:NodeMCU:12345Published data Successfully: %s
{'soil moisture': 54, 'temperature': 102, 'humidity': 54}
Published data Successfully: %s {'soil moisture': 82, 'temperature': 100, 'humidity': 65}
========== RESTART: C:\Users\VS\OneDrive\Desktop\python program.py ==========
2022-11-15 09:42:30,515 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:mlgc9d:NodeMCU:12345Published data Successfully: %s
{'soil moisture': 96, 'temperature': 60, 'humidity': 25}
Published data Successfully: %s {'soil moisture': 53, 'temperature': 35, 'humidity': 78}
Published data Successfully: %s {'soil moisture': 87, 'temperature': 28, 'humidity': 67}
Published data Successfully: %s {'soil moisture': 36, 'temperature': 111, 'humidity': 10}
Published data Successfully: %s {'soil moisture': 58, 'temperature': 109, 'humidity': 55}
Published data Successfully: %s {'soil moisture': 55, 'temperature': 84, 'humidity': 43}
Published data Successfully: %s {'soil moisture': 5, 'temperature': -2, 'humidity': 6}
Published data Successfully: %s {'soil moisture': 33, 'temperature': 18, 'humidity': 49}
Published data Successfully: %s {'soil moisture': 19, 'temperature': 90, 'humidity': 67}
Published data Successfully: %s {'soil moisture': 18, 'temperature': 121, 'humidity': 30}
Published data Successfully: %s {'soil moisture': 94, 'temperature': 83, 'humidity': 46}
Published data Successfully: %s {'soil moisture': 57, 'temperature': 50, 'humidity': 71}
Published data Successfully: %s {'soil moisture': 40, 'temperature': 19, 'humidity': 70}
Published data Successfully: %s {'soil moisture': 74, 'temperature': 69, 'humidity': 43}
Published data Successfully: %s {'soil moisture': 70, 'temperature': 62, 'humidity': 92}
Published data Successfully: %s {'soil moisture': 82, 'temperature': 68, 'humidity': 77}
Published data Successfully: %s {'soil moisture': 65, 'temperature': 99, 'humidity': 37}
Published data Successfully: %s {'soil moisture': 0, 'temperature': 124, 'humidity': 7}
Published data Successfully: %s {'soil moisture': 12, 'temperature': -3, 'humidity': 38}
Published data Successfully: %s {'soil moisture': 47, 'temperature': 100, 'humidity': 43}
Published data Successfully: %s {'soil moisture': 31, 'temperature': -9, 'humidity': 95}
Published data Successfully: %s {'soil moisture': 47, 'temperature': 42, 'humidity': 43}
Published data Successfully: %s {'soil moisture': 48, 'temperature': -13, 'humidity': 96}
Published data Successfully: %s {'soil moisture': 86, 'temperature': -17, 'humidity': 95}
Published data Successfully: %s {'soil moisture': 99, 'temperature': 73, 'humidity': 84}
Published data Successfully: %s {'soil moisture': 26, 'temperature': 24, 'humidity': 16}
Published data Successfully: %s {'soil moisture': 81, 'temperature': 121, 'humidity': 56}
Published data Successfully: %s {'soil moisture': 46, 'temperature': 30, 'humidity': 38}
Published data Successfully: %s {'soil moisture': 51, 'temperature': 28, 'humidity': 36}
Ln: 40 C
ENG
IN
09:43
15-11-2022

8.2 User Acceptance Testing



9.RESULTS

9.1 Performance metrics



10.ADVANTAGES &DISADVANTAGES

Advantages:

a. All the data like climatic conditions and changes in them, soil or crop conditions everything can be easily monitored.
b. Risk of crop damage can be lowered to a greater extent.
c. Many difficult challenges can be avoided making the process automated and the quality of crops can be maintained.

Disadvantages:

➢ The smart agriculture needs availability of internet continuously. Rural part of most of the developing countries do not fulfil this requirement. Moreover internet connection is slower.

➢ The smart farming based equipment require farmers to understand and learn the use of technology. This is major challenge in adopting smart agriculture farming at large scale across the countries.

11. CONCLUSION

Farmers can benefit greatly from an IoT-based smart agriculture system. As a result of the lack of irrigation, agriculture suffers. Climate factors such as humidity, temperature, and moisture can be adjusted dependent on the local environmental variables. This technology also detects animal invasions, which are a major cause of crop loss. This technology aids in the scheduling of irrigation based on present data from the field and records from a climate source. It helps in deciding the farmer to whether to do irrigation or not to do. Continuous internet connectivity is required for continuous monitoring of data from sensors. This also can be overcome by using GSM unit as an alternative of mobile app.

12.FUTURE SCOPE

In the current project we have implemented the project that can protect and maintain the the crop. In this project the farmer monitor and control the field remotely. In future we can add or update few more things to this project .

• We can create few more models of the same project ,so that the farmer can have information of a entire.

• We can update the this project by using solar power mechanism. So that the power supply from electric poles can be replaced with solar panels. It reduces the power line cost. It will be a one time investment. We can add solar fencing technology to this project.

• We can use GSM technology to this project so that the farmers can get the information directly to his home through SMS. This helps the farmer to get information if there is a internet issues.

• We can add camera feature so that the farmer can monitor his field in real time. This helps in avoiding thefts.

13.APPENDIX

Source Code:

```
import wiotp.sdk.device
import time
import os
import datetime
import random
```

```
myConfig = {
  "identity": {
     "orgId": "bj6xkv",
     "typeId": "nodemcu",
     "deviceId": "12345"
  },
  "auth": {
     "token": "12345678"
  }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect ()

def myCommandCallback (cmd) :
    print ("Message received from IBM IoT Platform: %s"
%cmd.data['command'])
    m=cmd.data['command']
    if (m=="motoron"):
        print ("Motor is switched on")
    elif (m=="motoroff"):
        print ("Motor is switched OFF")
    print (" ")

while True:
    soil=random.randint (0,100)
    temp=random.randint (-20, 125)
    hum=random.randint (0, 100)
    myData={'soil moisture': soil, 'temperature':temp, 'humidity':hum}
    client.publishEvent (eventId="status", msgFormat="json",
    data=myData, qos=0 , onPublish=None)
    print ("Published data Successfully: %s", myData)
    time.sleep (2)
```

```
        client.commandCallback = myCommandCallback
client.disconnect ()
```

Github link:https://github.com/IBM-EPBL/IBM-Project-43581-1660718206

Demo Video Link:

https://drive.google.com/drive/folders/11IX-zWdiIKSq1g9UqUmasFiccFj2CrGU