

SMARTFARMER- IOT ENABLED SMART FARMING

APPLICATION

-

Team ID: PNT2022TMID07143

Team Member

Dharshini S -610519106009

Jayaraman R - 610519106020

Lalprasanth N - 610519106029

Shankari - 610519106062

1. INTRODUCTION

- a. Project Overview
- b. Purpose

2. LITERATURE SURVEY

- a. Existing problem
- b. References
- c. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- a. Empathy Map Canvas
- b. Ideation & Brainstorming
- c. Proposed Solution
- d. Problem Solution fit

4. REQUIREMENT ANALYSIS

- a. Functional requirement
- b. Non-Functional requirements

5. PROJECT DESIGN

- a. Data Flow Diagrams
- b. Solution & Technical Architecture
- c. User Stories

6. PROJECT PLANNING & SCHEDULING

- a. Sprint Planning & Estimation
- b. Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

- a. Feature 1
- b. Feature 2
- c. Database Schema (if Applicable)

8. IBM cloud & TESTING

- a. Test Cases
- b. User Acceptance Testing in MIT app
- c. Fast2sms

9. RESULTS

- a. Performance Metrics

10.ADVANTAGES & DISADVANTAGES

11.CONCLUSION

12.FUTURE SCOPE

13.APPENDIX

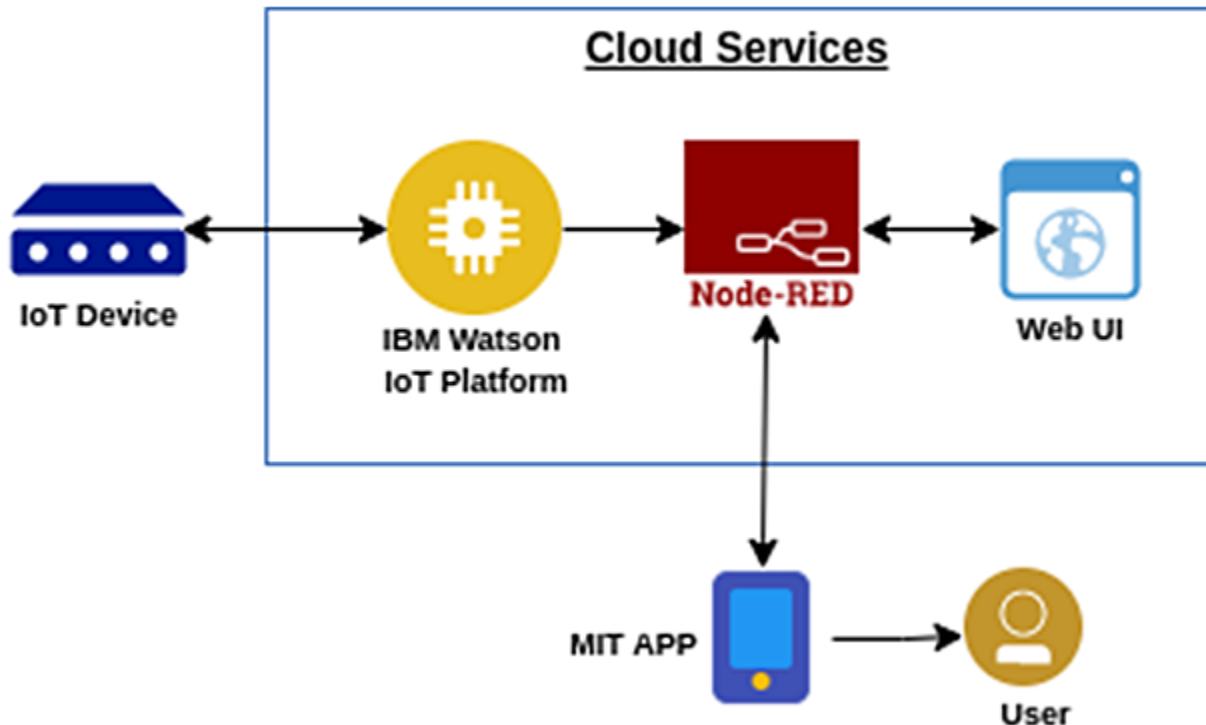
Source Code

GitHub & Project Demo Link

INTRODUCTION

1.1 Project 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.



1.2 Purpose

The smart agriculture model main aim to avoid water wastage in the irrigation process. It is low cost and efficient system Is shown below. It includes NodeMCU, Arduino Nano, sensors like soil moisture and Dht11, solenoid valves, relays.

2.LITERATURE SURVEY

2.1 Existing problem

Smart agriculture farming system is a new idea of farming in agriculture, because which uses IoT technology to monitor the crop 24/7 and sends the information to the cloud. This emerging system increases the quality and quantity of agricultural products. IoT technology provides the information about farming fields and then takes action depending on the farmer input. In this project we can design an IoT based advanced solution for monitoring the soil conditions and atmosphere for efficient crop growth is presented. The developed system is capable of monitoring temperature, humidity, soil moisture type using Arduino UNO R3 and different sensors connected to the microcontroller. Also, a notification is shown in farmer's phone using Wi-Fi about environmental condition and water levels of the crop field

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

2.2 References

LITERATURE SURVEY

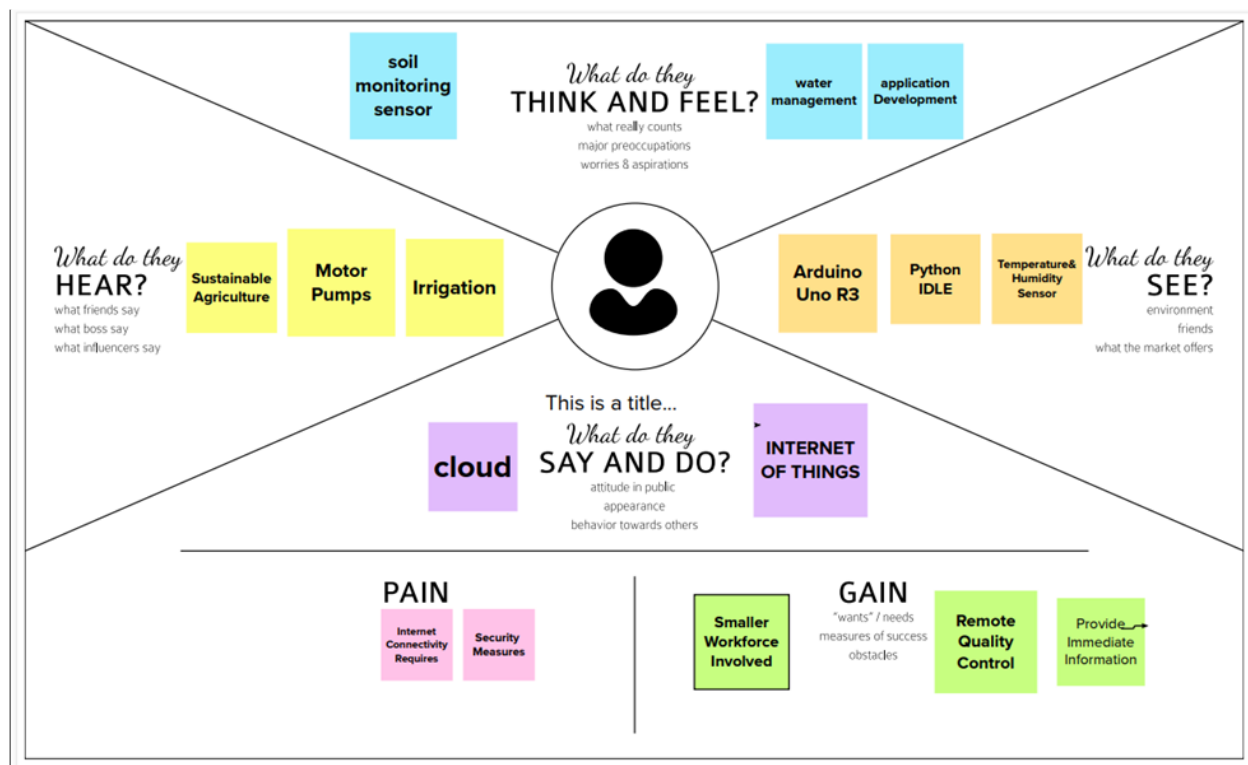
PAPER TITLE	AUTHOR NAME	OBJECTIVES
Smart Farming management system using IOT Reference HERE	Garigipati Vijay Kumar, Pallikonda Jashuva	Due to the usage of this system,adequate water is pumped and rain is also utilized efficiently using Node MCU,Sensors like Soil Moisture sensor,Humidity and Temperature Sensor and All the information are send to the farmers mobile using wifi technology.
IoT Based Intelligent Agriculture Field Monitoring System Reference HERE	Md Ashifuddin Mondal, Zeenat Rehena	Arduino board controls the high voltage farming equipments without human intervention.Also the system sends the environmental parameters values to the cloud from the field in real time through wireless communication in every certain time interval.
Survey on Smart Agriculture Using IOT Reference HERE	Shweta A M, Dr V. Nagaveni	Smart agriculture reduces wastage of water, fertilizers and increases the crop yield.using Internet of Things (IoT), Agriculture, Agriculture Precision, Raspberry Pi, Temperature Sensor, Smart Farming, Soil Moisture Sensor

2.3 Problem Statement Definition

The soil moisture sensor measures wetness content in the soil. The Arduino UNO microcontroller used to receive input from a various sensor and it can be controlled automatically. When soil moisture sensor goes low the water pump will be on and it exceeds defined levels of the water motor will turn off automatically. We can constantly monitor the growth of a crop using ultrasonic sensor. PIR sensor detects the motion or unusual movement in the agricultural land. This device his very helpful to the former to monitor and control environmental parameters at their field. The farmers did not go to their field, they can remotely monitor and control using cloud

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming



3.3 Proposed Solution

S.No	Parameters	Description
1	Problem statement (problem to be solved)	SmartFarmer-Iot Enabled Smart Farming Application
2	dea/Solution Description	This project helps the farmer in monitoring different parameters of his field like soil, moisture, Temperature and humidity using some sensors. Farmers can monitor all the sensor parameters by using a web or mobile application even if the farmers is not near his field.
3	Novelty/Uniqueness	Watering the crops 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 parameter and controlling the motor pumps from the mobile application .
4	Social impact/Customer Satisfaction	Smart Farming allows farmers to be much more precise. By Using Iot farmers boosts operational proficiency and reduces energy waste. Farmer can monitoring and get all the sensor parameters data about his field in real time through web application
5	Business Model(Revenue Model)	<ul style="list-style-type: none"> • Cost Efficiency • Fuel Efficiency
6	Scalability the solution	This project has HIGH SCALABILITY in monitoring all the sensor parameters and controlling the motor pump through mobile applications

3.4 Problem solution fit

<p>1.Customer Segments</p> <p>The customers who are going to adapt this project are:</p> <ul style="list-style-type: none"> *Large Scale Farmer *Remote Farmers 	<p>6. Customer constrains</p> <p>The customer wants a device which could be the solution in the field of watering the crops even the farmer is not presence in field and the device should fulfill all the following constraints</p> <ul style="list-style-type: none"> * Cost and Time Efficient * Resource Efficient 	<p>5. Available solutions</p> <p>The moisture-controlled irrigation system could be the best solution for this problem statement that has been provided by the farmers and also it specifically satisfies the customer constraints also.</p>
---	--	--

<p>2. Jobs to be done.</p> <p>The customer needs to automate the process of irrigation in cost efficient, energy efficient, reduced power consumption and can be achieved in a reliable</p>	<p>9.Problem route cause</p> <p>1) The problem has its route stabled at the rate of the fast-moving world since people move fast of the times and since they have their work to be stagnated similarly farmers face the inability in the process of irrigation</p> <p>2)In some times, farmers can't get or predict the sensing parameters data accurately (humidity, moisture, temperature)</p>	<p>7.Bahaviour</p> <p>The customer wants to make the revolutionary propagation in the rating of the irrigation through the reliability of amount of water availability on the land.</p>
<p>3.Triggers</p> <p>The reliability and easy accessibility of this finished projects yields the Customer's attraction and they can easily be installed in their fields</p>	<p>10. Solution.</p> <p>Our solution for this project is to initiate the reliability of the irrigation system using the sensor sensed information from the field and also make the automation is on and off of water pump</p>	<p>8. Channels of behavior</p> <p>The channels of behaviorrecombine the ration of following: *ONLINE: Using mobile application the farmers can controlling the motor pumps through online mode. *OFFLINE: Farmers can get the sensing parameters data via SMS.</p>
<p>4. Emotions</p> <p>Before – manual system is necessary to monitor the fields After – Manual system is not Necessary since in our project we'll develop a mobile or web applications so the farmers can control and monitor about his field. So, the customers feel comfortable and Happy.</p>		

4.REQUIREMENT ANALYSIS

FR.No	Functional Requirement (Epic)	Sub Requirement (Story/Sub-Task)
FR-1	User Registration	Registration through Gmail Registration through Form
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Log in to System	Check credentials check Role of Access
FR-4	Manage Modules	Manage System Admin Manage Roles of User Manage User permission
FR-5	Check Credentials Details	Temperature details Humidity details
FR-6	Log out	Exit the application

4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Usability includes easy learn ability, efficiency in use, remember ability, lack of errors in operation and subjective pleasure
NFR-2	Security	Sensitive and private data must be protected from their production until the decision-making and storage stages
NFR-3	Reliability	The shared protection achieves a better trade-off between costs and reliability.
NFR-4	Performance	The idea of implementing sensors with sensing soil and environmental or ambient parameters in farming will be more efficient for overall monitoring

5.PRODUCT DESIGN

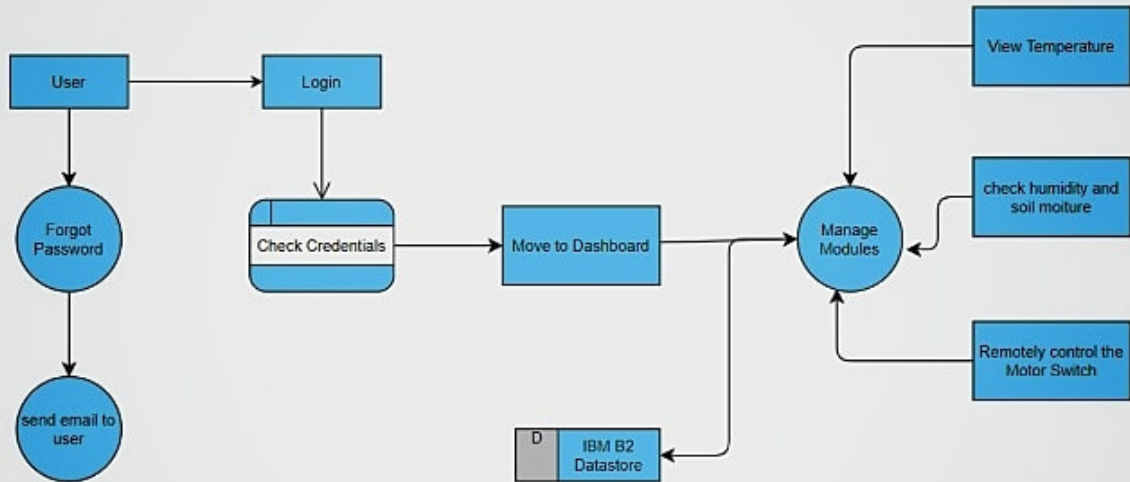
5.1Data flow diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement

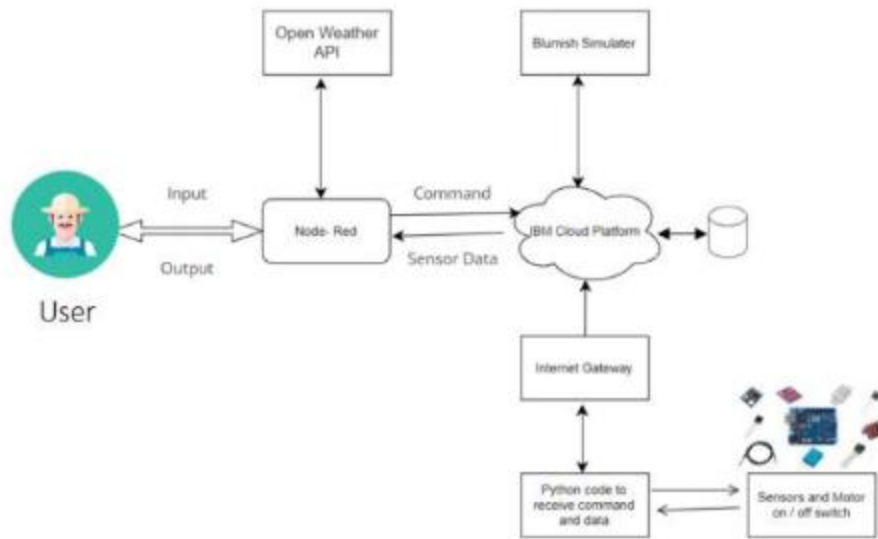
graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

- The different soil parameters temperature, soil moistures and then humidity are sensed using different sensors and obtained value is stored in the IBM cloud.
- Arduino UNO is used as a processing Unit that process the data obtained from the sensors and whether data from the weather API.
- 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 plan through an app, weather to water the crop or not depending upon the sensor values. By using the app they can remotely operate to the motor switch

Data Flow Diagrams:



5.2 Solution and Technical Architecture



- The different soil parameters temperature, soil moistures and then humidity are sensed using different sensors and obtained value is stored in IBM cloud.
- Arduino Uno is used as a processing unit that process the data obtained from the sensors and weather data from the weather API
- 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, whether to water the crop or not depending upon the sensor values, By using the app,they can remotely operate the motor switch(On/off).

6.PROJECT PLANNING AND SCHEDULING

Sprint	User Story Number	User Story/ Task	Story Points	Priority	Team Mambers
Sprint 1	USN-1	Sensors and wi-fi module with python code	2	High	Lalprasanth N Dharshini S Shankari J Jayaraman R

Sprint 2	USN-2	IBM Watson IoT Platform, workflows for IoT Scenarios using Node-Red	2	High	Lalprasanth N Dharshini S Shankari J Jayaraman R
Sprint 3	USN-3	To Develop an Mobile application using MIT	2	High	Lalprasanth N Dharshini S Shankari J Jayaraman R
Sprint 4	USN-4	To make the user to interact with Software	2	High	Lalprasanth N Dharshini S Shankari J Jayaraman R

7.CODING AND SOLUTIONING

7.1 Feature

```

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

#Provide your IBM Watson Device Credentials

orgId = "oy32g"
deviceType = "Dharshini"
deviceId = 2002
token = "sridharan11"
authMethod = "use-token-auth"

# 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": orgId, "type": deviceType, "id": deviceId, "auth-method":
authMethod, "auth-token": Token}
    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(-20,125)
    hum=random.randint(0,100)
    soil=random.randint(0,100)
    data = { 'temp' : temp, 'hum': hum , 'soil': soil}
    #print data
    def myOnPublishCallback( ):
        print (f"Published temp = {temp} C , hum = {hum} , soil = {soil} deg c to IBM Watson")
        success = deviceCli.publishEvent("IoTSensor", "json", data,
qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTTF")
        time.sleep(10)
    deviceCli.commandCallback = myCommandCallback
    # Disconnect the device and application from the cloud
    deviceCli.disconnect()

```

```
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
orgId = "cy32g"
deviceType = "Dharshini"
deviceId = 2002
token = "sri@haranil"
authMethod = "use-token-auth"
# 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": orgId, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": Token}
    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(-20,125)
    hum=random.randint(0,100)
    soil=random.randint(0,100)
    data = { 'temp': temp, 'hum': hum, 'soil': soil}
    #print data
    def myOnPublishCallback():
        print ("Published temp = {temp} C, hum = {hum} , soil = {soil} deg c to IBM Watson")
        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoT")
        time.sleep(10)
    deviceCli.commandCallback = myCommandCallback
    # Disconnect the device and application from the cloud
    deviceCli.disconnect()
```

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (tags/v3.7.0:1bb39cc593, Jun 27 2018, 04:19:51) [AMD64] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\manoj-gt5990\Documents\python\project.py =====
2022-11-18 11:57:44.438  IBMiotf.device.Client  INFO  Connected successfully: d1ackdf7b0de8c7f12345
Published Temperature = 30 C Humidity = 67 % Soil Moisture = 48 % to IBM Watson
Published Temperature = 6 C Humidity = 17 % Soil Moisture = 23 % to IBM Watson
Published Temperature = 17 C Humidity = 49 % Soil Moisture = 52 % to IBM Watson
Published Temperature = 64 C Humidity = 25 % Soil Moisture = 40 % to IBM Watson
Published Temperature = 83 C Humidity = 92 % Soil Moisture = 1 % to IBM Watson
Published Temperature = 98 C Humidity = 69 % Soil Moisture = 38 % to IBM Watson
```

8. IBM- Cloud & Testing

8.1 Testcase

IBM Cloud

cloud.ibm.com

Settings New Tab Gmail YouTube Maps SmartFarmer - IoT...

IBM Cloud Search resources and products...

Dashboard

Edit dashboard Upgrade account Create resource +

For you

Select an option

Build

Explore IBM Cloud with this selection of easy starter tutorials and services.

Build a web app with Watson Speech to Text

Deploy a conversational interface compatible with any application, device, or channel.

Get started with Watson Discovery

Get up to speed on Watson Discovery with step-by-step tutorials, deep-dive videos, and complete examples of working code.

Access to a service within resource groups

Use IAM to manage access to resource groups, and thus give other users access to your service instance.

Unlock the entire catalog

Upgrade your account to start using everything IBM Cloud has to offer.

Set up your IBM Cloud account

Learn how to set up your IBM Cloud account, manage your account settings, organize resources, and control access to those resources.

Getting started 15 min Recommended 2 hr Recommended 5 min Recommended 10 min Getting started 10 min

Type here to search

cloud.ibm.com/catalog/services/internet-of-things-platform

Settings New Tab Gmail YouTube Maps SmartFarmer - IoT...

IBM Cloud

Search resources and products...

Catalog Manage Dharshini Sridharan's A...

Internet of Things Platform

This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can access their live and historical data.

Create About

Type Service

Provider IBM

Last updated 08/15/2022

Category Internet of Things

Compliance IAM-enabled

Location Frankfurt London Dallas Washington DC

Related links Docs

Select a location

London (eu-gb)

Select a pricing plan

Displayed prices do not include tax. Monthly prices shown are for country or location: [United States](#)

Plan	Features	Pricing
Lite	Includes up to 500 registered devices, and a maximum of 200 MB of each data metric Maximum of 500 registered devices Maximum of 500 application bindings Maximum of 200 MB of each of data exchanged, data analyzed and edge data analyzed	Free

The Lite service plan for Internet of Things Platform includes up to 500 registered devices, and a maximum of 200 MB each of

Summary

Internet of Things Platform Free

Location: London

Plan: Lite

Service name: Internet of Things Platform-3q

Resource group: Default

☒ I have read and agree to the following license agreements:
[Terms](#)

Create

Add to estimate

27°C Haze 07:35 PM 07-11-2022

IBM Cloud Service Details - IBM Cloud

cloud.ibm.com/services/iotf-service/cm%3Av1%3Abluemix%3Apublic%3AIotf-service%3Aeu-gb%3Aa%2Fa08c2cbca71b4890b41df63b841b5876%3A0fcc494a-001a-4998-abf7-1fc86d59255e...

Settings New Tab Gmail YouTube Maps SmartFarmer - IoT...

IBM Cloud Search resources and products...

Resource list /

Internet of Things Platform-3q

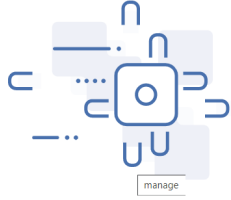
Active Add tags

Details Actions...

Manage

Plan

Connections



Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Launch Docs

Ready for the next level?

IBM Watson IoT Platform Journey

Lite

The Lite service plan provides a lightweight development environment to get you started with the connectivity capabilities of Watson IoT Platform.

Non-Production

The Non-Production service plan is a full-featured, fully-integrated offering that enables you to explore Watson IoT Platform to see how the service can fit into your IoT environment.

Production

The Production service is a fully managed SaaS offering that enables you to manage and analyze enterprise IoT data.

Free

Starts at \$500 per month

Includes IBM Service & Support

Web application using node red.

IBM App Development IBM Cloud Account IBM Watson IoT Platform Node-RED Dashboard Node-RED : node-red-cd Node-RED Dashboard

node-red-cdmi-2022-11-15-eu-gb.mybluemix.net/red/#flow/da1bafa2e25950fc

Settings New Tab Gmail YouTube Maps SmartFarmer - IoT...

Node-RED

filter nodes

Flow 1

common

inject

debug

complete

catch

status

link in

link call

link out

comment

function

function

switch

change

range

IBM IoT

connected

soil moisture

humidity

temperature

Motor on

Motor off

[get]/data

data

Edit ibmiot in node

Delete Cancel Done

Properties

Authentication API Key

API Key newap

Input Type Device Event

Device Type All or Dharshini

Device Id All or 2002

Event All or +

Format All or json

QoS 0

Name IBM IoT

Service registered

Use the Input Type property to configure this node to receive Events sent by IoT Devices. Commands sent by IoT Devices - Status

Enabled

debug

all nodes all

43

11/16/2022, 10:21:16 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

95

11/16/2022, 10:21:16 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

95

11/16/2022, 10:21:16 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

87

11/16/2022, 10:21:19 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

33

11/16/2022, 10:21:19 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

34

11/16/2022, 10:21:19 AM node: f22649a.0d0d98
iot-2/type/Dharshini/id/2002/event_11fmi/json :
msg.payload : number

2

The screenshot displays the IBM Watson IoT Platform interface. The main dashboard shows a list of devices, with one device (ID: 2002) highlighted. The device details panel on the right shows the device is disconnected and provides options to add devices or use the 'Add Device' button. The 'Device Type: Dharshini' configuration panel is open, showing the 'Events' section. The 'Event type name' is 'event_1', and the 'Schedule' is set to 'Every Minute'. The 'Payload' section shows a JSON payload with random values for 'soil', 'hum', and 'temp'.

Device ID	Status	Device Type	Class ID	Date
2002	Disconnected	Dharshini	Device	Nov 15, 2022 6:22 PM

Device Details:

- Device ID: 2002
- Device Type: Dharshini
- Date Added: Nov 15, 2022 6:22 PM
- Added By: dharshinis009.ece@dgct.ac.in
- Connection Status: Disconnected

Event Configuration:

- Event type name: event_1
- Schedule: 20 Every Minute
- Payload:


```
{
    "soil": random(0, 100),
    "hum": random(0, 100),
    "temp": random(-20, 125)
  }
```

The screenshot displays the Node-RED interface. The main workspace shows a flow with several nodes: 'IBM IoT' (connected), 'soil moisture', 'humidity', 'temperature', 'moisture', 'Humidity', 'Temperature', 'msg.payload', 'Motor on', 'Motor off', and 'IBM IoT' (connected). The flow is configured to process data from the IoT device and control the motor. The 'debug' console on the right shows the flow's execution history, including timestamps and node IDs.

Flow Configuration:

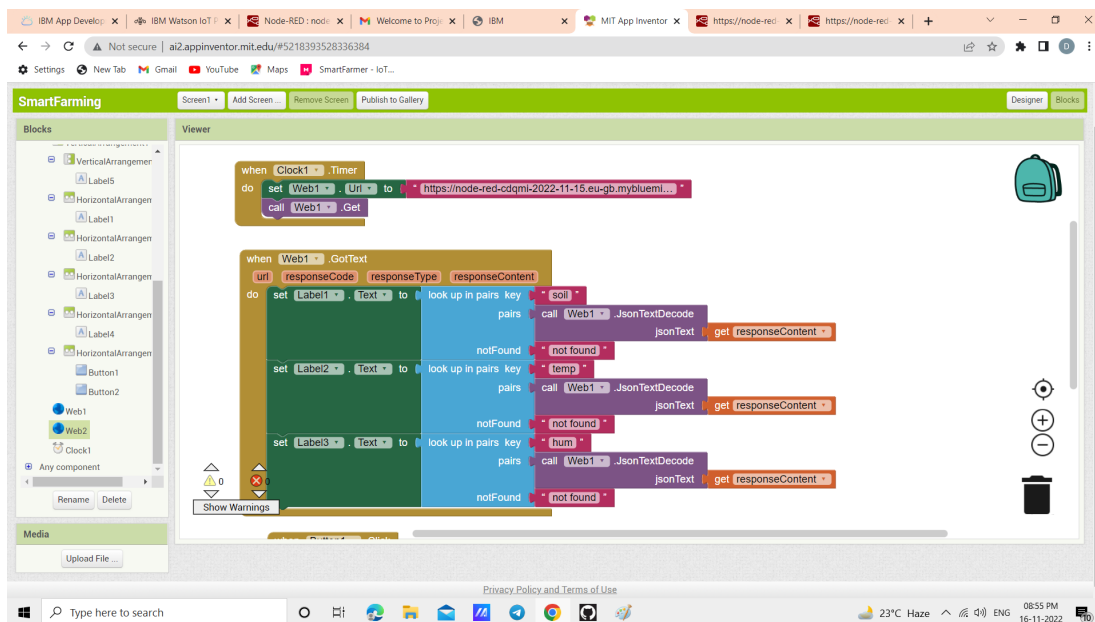
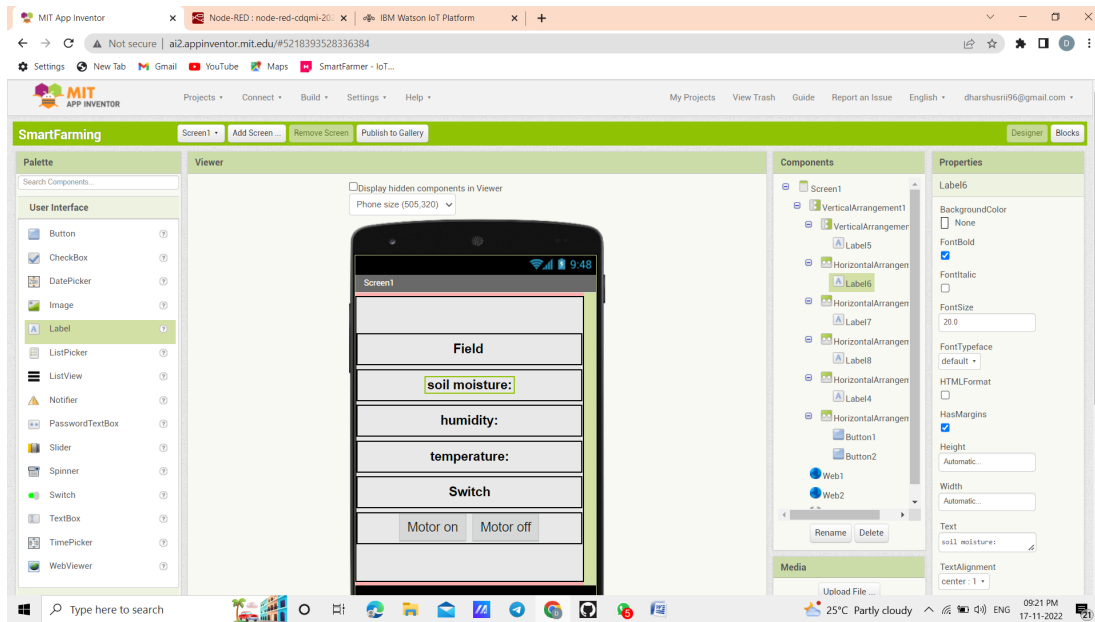
- Input: IBM IoT (connected)
- Processors: soil moisture, humidity, temperature
- Outputs: moisture, Humidity, Temperature
- Control: Motor on, Motor off
- Destination: IBM IoT (connected)

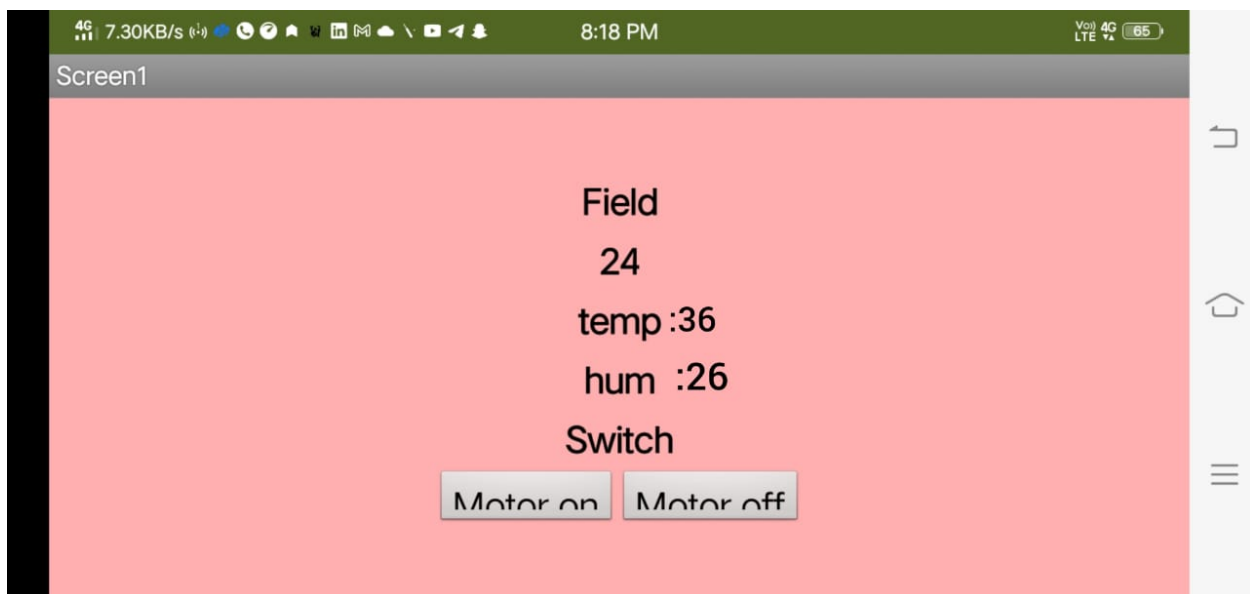
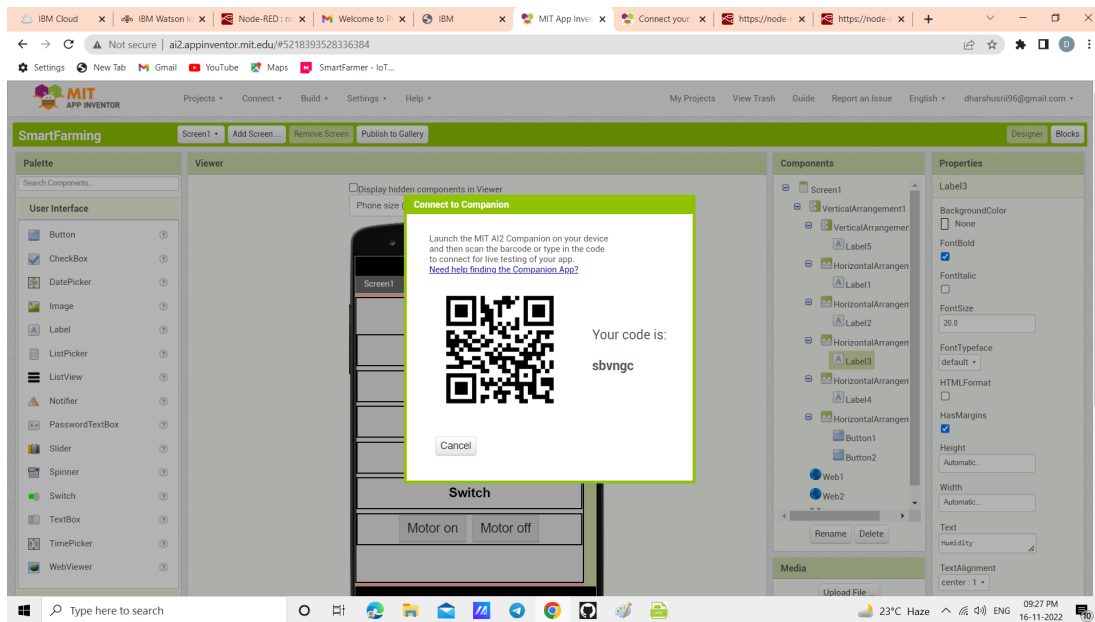
Debug Console:

```

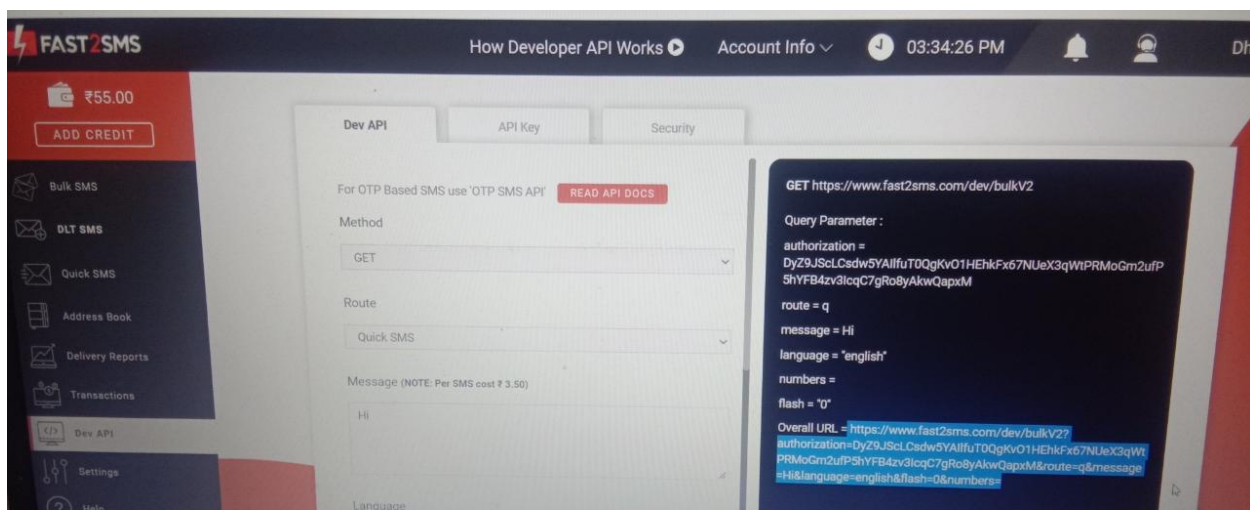
11/15/2022, 9:11:28 PM node: 22649a.0d0d98
iot-2/type/Dharshini/id/2002/ev/event_1/fmt/json :
msg.payload : undefined
undefined
11/15/2022, 9:11:28 PM node: 22649a.0d0d98
iot-2/type/Dharshini/id/2002/ev/event_1/fmt/json :
msg.payload : undefined
undefined
11/15/2022, 9:11:28 PM node: 22649a.0d0d98
iot-2/type/Dharshini/id/2002/ev/event_1/fmt/json :
msg.payload : number
23
11/15/2022, 9:11:31 PM node: 22649a.0d0d98
iot-2/type/Dharshini/id/2002/ev/event_1/fmt/json :
msg.payload : undefined
undefined
11/15/2022, 9:11:31 PM node: 22649a.0d0d98
iot-2/type/Dharshini/id/2002/ev/event_1/fmt/json :
msg.payload : number
57
  
```

8.2 User Aceptance Testing in MIT app



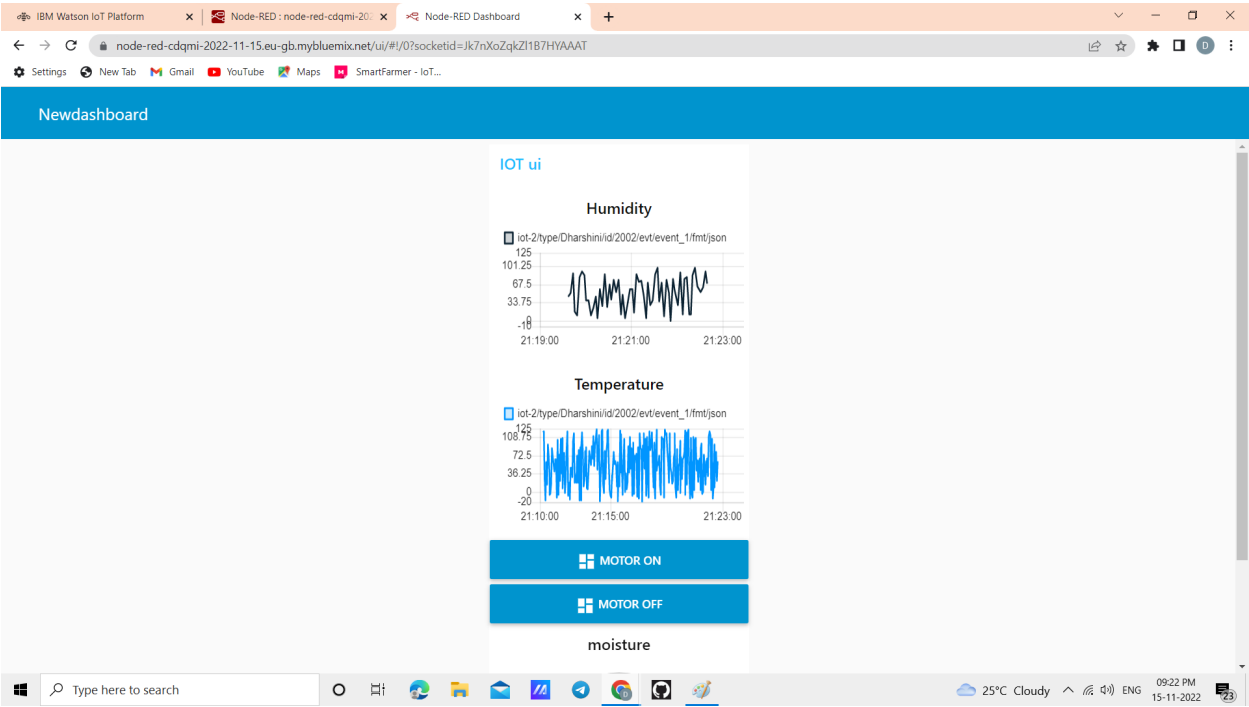


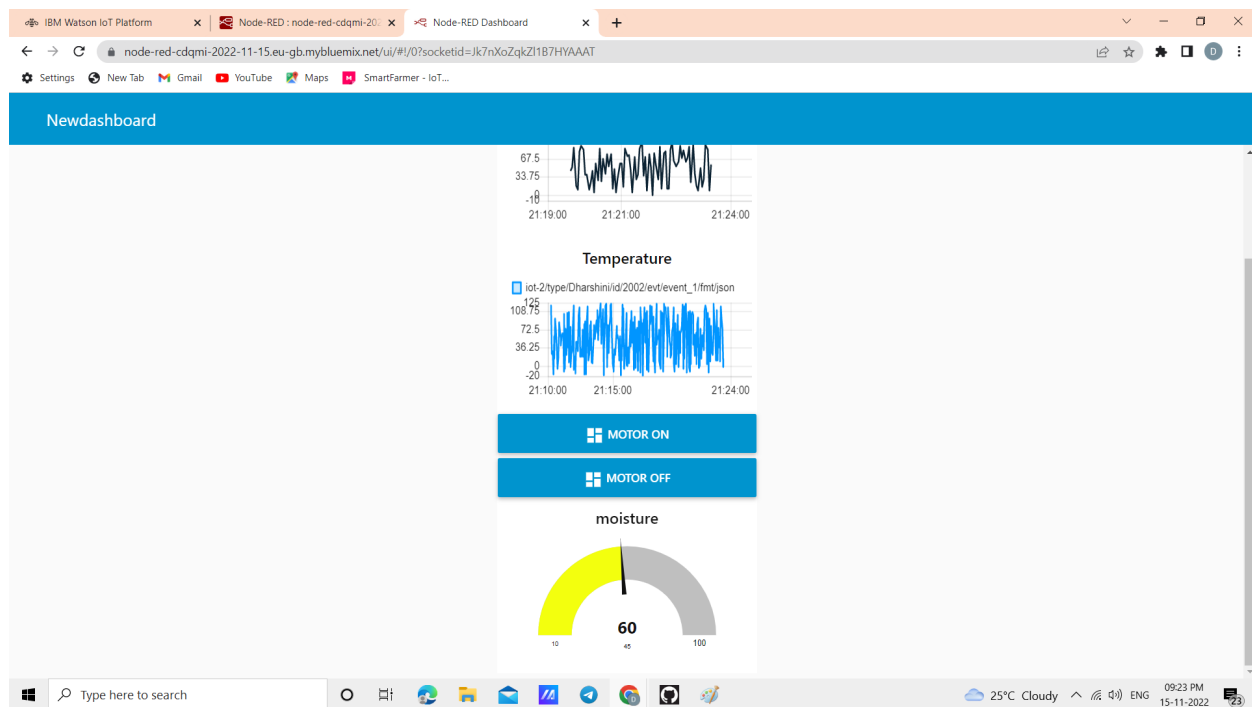
Fast2SMS:



9.Results

9.1 Performance Metrics





10. Advantages and disadvantages

Advantages:

- A remote-control system can help in working irrigation system valves dependent on schedule. Irrigating remote farm properties can be exceptionally troublesome and laborintensive. It gets hard to comprehend when the valves were started and whether the ideal measure of water was distributed.
- For situations where a quick reaction is required, manual valve actuation may not be conceivable constantly. Thus, remote observing and control of irrigation systems, generators or wind machines or some other motor-driven hardware become the next logical step.
- Various solutions are available to monitor engine statistics and starting or stopping the engine. When the client chooses to begin or stop the motor, the program transmits a sign to the unit within seconds by means of a mobile phone system.
- Submersible weight sensors or ultrasonic sensors can screen the degree of tanks, lakes, wells and different kinds of fluid stockpiling like fuel and compost. The product figures volume dependent on the tank or lake geometry after some time. It conveys alarms dependent on various conditions.

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. By GSM, SMS can be sent to farmer's phone.

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

#Provide your IBM Watson Device Credentials

orgId = "oy32g"
deviceType = "Dharshini"
deviceId = 2002
token = "sridharan11"
authMethod = "use-token-auth"

# 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": orgId, "type": deviceType, "id": deviceId, "auth-method":
authMethod, "auth-token": Token}
```

```

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(-20,125)
    hum=random.randint(0,100)
    soil=random.randint(0,100)
    data = { 'temp' : temp, 'hum': hum , 'soil': soil}
    #print data def myOnPublishCallback( ):
    print (f"Published temp = {temp} C , hum = {hum} , soil = {soil} deg c to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data,
qos=0,on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTTF")
        time.sleep(10)
    deviceCli.commandCallback = myCommandCallback
    # Disconnect the device and application from the cloud
    deviceCli.disconnect()

```

Github Link : <https://github.com/IBM-EPBL/IBM-Project-31209-1660197678.git>

Project Demo link

https://drive.google.com/file/d/1zZDA866EhZg7gj_RaLxPpj4LjCHus5dW/view?usp=drive_sdk

