IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

TEAM ID : PNT2022TMID08774

TEAM MEMBERS:

I. N.DHINESH – 727619bec042
II. S.RAJADURAI – 727619bec034
III. S.KARTHIKEYAN – 727619bec018
IV. V.SELVARAJ – 727619bec064

1. INTRODUCTION

1.1 PROJECT OVERVIEW:

Crops in farms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. This leads to huge losses for the farmers. It is not possible for farmers to barricade entire fields or stay on field 24 hours and guard it. So here we propose an automatic crop protection system from animals. The system now sound an alarm to woo the animal away from the field as well as sends SMS to the farmer so that he may about the issue and come to the spot in case the animal don't turn away by the alarm. This ensures complete safety of crop from animals thus protecting farmers loss.

1.2 PURPOSE:

An intelligent crop protection system helps the farmers in protecting the crop from the animals and birds which destroy the crops. The images of the animals can be captured using the help of camera which can be further processed with the help of python modules. The system also helps the farmer to monitor the moisture level in the field, temperature and humidity values of the field. Several sensors like DHT11, soil moisture sensor module. All the data are stored in the IBM cloud which can be later analyzed. The motor and sprinklers in the field can controlled using the mobile application. The current status of the crops can be viewed remotely.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The existing system mainly provide the surveillance functionality. Also, these system don't provide protection from wild animals, especially in such an application area. They also need to take actions based on the type of animal that tries to enter the area, as different methods are adopted to prevent different animals from entering restricted areas. The other commonly used method by farmer in order to prevent the crop vandalization by animals include building physical barriers, use of electric fences and manual surveillance andvarious such exhaustive and dangerous method.

2.2 REFERENCES:

- **IoT Based Crop Protection System against Birds and Wild Animal Attacks.**
  (P.Navaneetha, R.Ramiya Devi, S.Vennila, P.Manikandan, Dr.S.Saravanan, Muthayammal Engineering College, Namakkal, Tamilnadu – April 2020)

- **Smart crop protection system from wild animals and birds using IOT.**
  (P.B.Sumana, R.Sanjana, M.Sharanya, N.J.Harish, Global Academy of Technology, Bengaluru, Karnataka – 2021)

- **IoT Based Crop Monitoring from Animals.**
  (K.B.PavanKumar, T.Bhavitha, S.Karishma, M.Pavithra, M.PrashanthKumar, Mother Theresa Institute of Engineering and Technology, Palamaner, AP – March 2019)

- **Smart Crop Protection System from Birds and Animals**
  (V.Shashi Kiran, N.Manoj, M.HemanthKumar, M.N.Namith, VVIET, Mysore – July 2022)

- **Smart Crop Protection System from Animals**
  (Mr.Jayesh Redij, Mr.Pranav Shitap, Mr.Shikhar Singh, Mr.Durvesh Zagade, Dr.Sharada Chougule, Finolex Academy of Management and Technology, Ratnagiri – April 2022)

2.3 PROBLEM STATEMENT DEFINITION:

- In this system, cameras will be installed. With its help, we can capture the image of the animal or bird. After the identification of the species, corresponding measures will be taken to divert that animal or bird from the field.

- This system won't harm the animals or birds. This will just give alerting signal such as sound which tends them move out the farm area, also by the data stored in the cloud can be used to find the type of animals which often invading into the area.

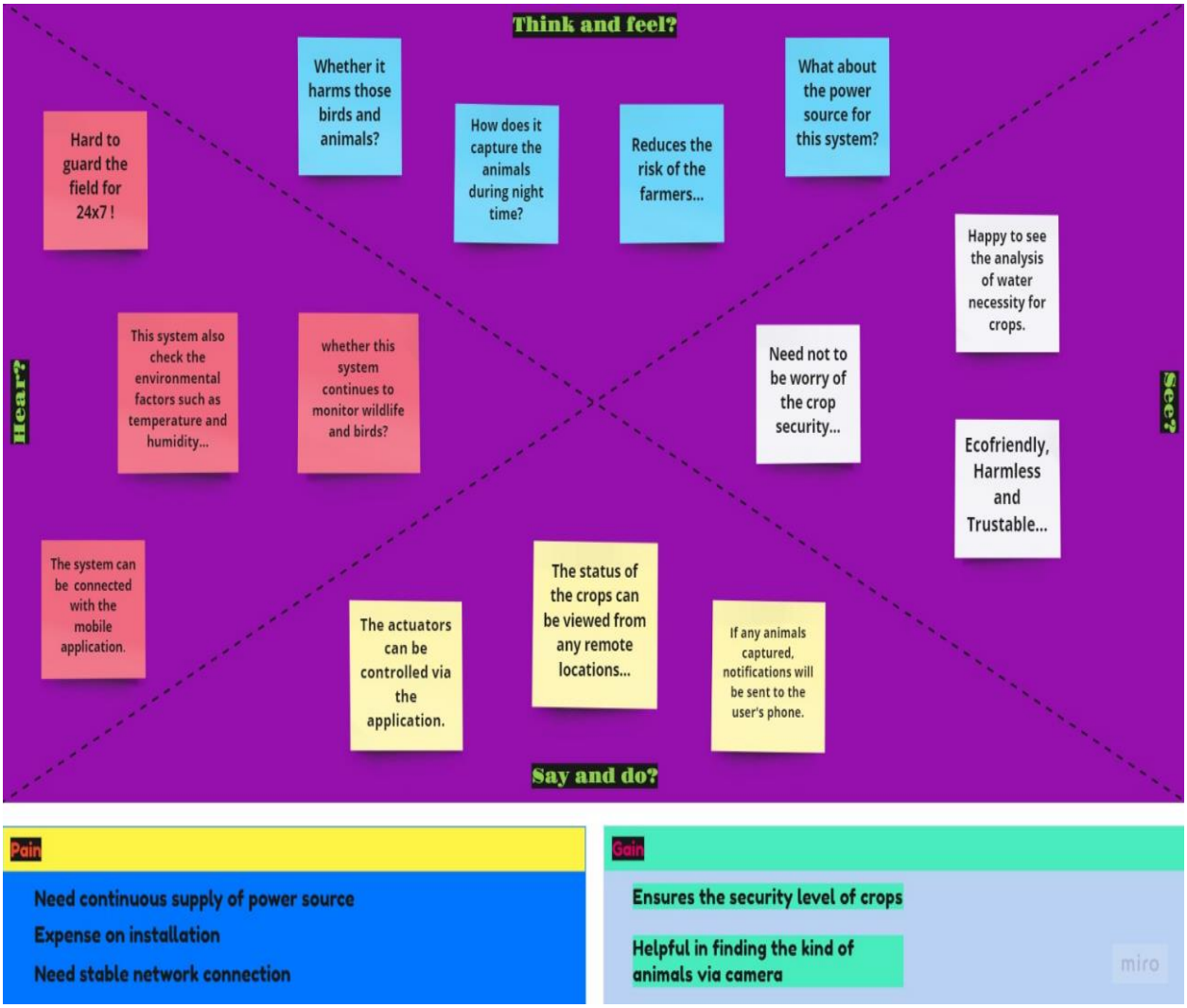
- In hilly areas, by installation of this system can be very useful to improve the security level of the crops. If there is any caught of the sign of the wild animals such as Elephant, Bison and Wild pigs, a notification will be sent to user as well as alerting message will be sent to the forest department of the particular area.

- Status of the crops and factors affecting the field environment such as temperature, humidity and soil moisture can be monitored continuously from any remote locations.

- The actuators such as sprinklers, solenoid valve to control the flow of the water into the field can be controlled with the help of the mobile application. The actuators depends on the kind of the land area such as hill, plain, valleys.

3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION AND BRAINSTROMING:

The device will detect the animals and birds using sensors.

Generates an alarm thereby to drive away the animals if captured.

TIP

Add customizable tags to st notes to make it easier to fin browse, organize, and categorize important ideas a themes within your mural.

To keep a man to guard the farm and to maintain the farm.

The water level in the field should be able to observable and controllable.

Using of camera to monitor the movement of animals.

Creating a web application to check status of field remotely.

Electric agricultural fences.

The image URL is stored in the IBM cloudant DB.

The image stored can be viewed by application.

Soil moisture, temperature and humidity also viewed with the help of application.

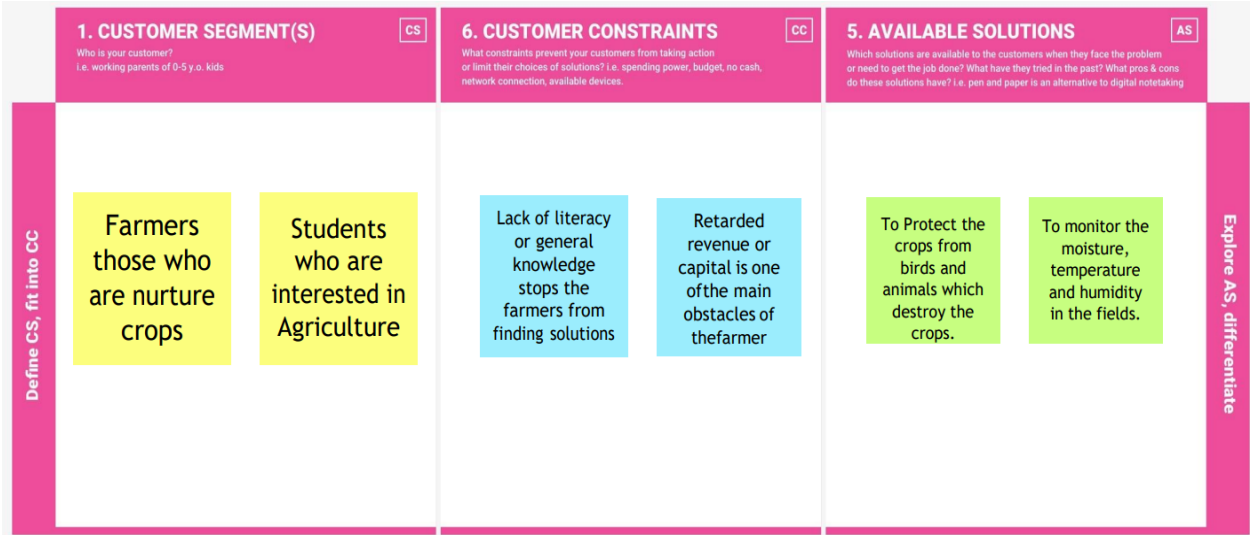
User can also control the motors through mobile application.

In hilly areas, whenever the wild animals caught, the system should be able to alert forest department.

3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To protect crop from animals and used to monitor the moisture, temperature and humidity in the field. This also used to control the motors using the mobile application.
2.	Idea / Solution description	To suggest the farmers to efficiently use the application to protect the crops and monitor the conditions in the field.
3.	Novelty / Uniqueness	Customer's ease usage of the application is the key uniqueness, as it is very easy to use.
4.	Social Impact / Customer Satisfaction	It will have a momentum in agriculture industry because of its simplicity and user-friendly nature.
5.	Business Model (Revenue Model)	It is cost efficient since farmer can get guidance by the application itself rather than consulting an agriculturist.
6.	Scalability of the Solution	It is capable of adapting itself for larger customer space and works efficiently and faster.

3.4 PROBLEM SOLUTION FIT:



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.

- It's complex to monitor and control
- Aware of harmful insects and pathogens which affects the crops
- Could not be able to know if the application correctly works or not

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.

- We can't set a person to guard, If set the guard couldn't able to cover the whole farm in the night times
- If any works other than farming is to be done by the farmer, he could able to see the status of the farm remotely

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)

- Direct related: The farmer himself tries to find a solution for the problem
- Indirect associated: The sources such as power or network should be uninterrupted

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

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

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.

- Visiting the government's agri-tech sites and gain knowledge and share with the other farmers

10. YOUR SOLUTION [SL]

What kind of solution suits Customer scenario the best?
Adjust your solution to fit Customer behaviour, use Triggers, Channels & Emotions for marketing and communication.

- Suggesting to use the proposed application to protect crops and monitor the conditons in the field.

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.

8.1 ONLINE CHANNELS [CH]

What kind of actions do customers take online?
Extract online channels from box #7 Behaviour

- Farmers can monitor the movement of the animals in the field
- Monitor and operate the conditons and motors or sprinklers through mobile.

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: The farmers were dejected and displeased because of the crop diseases
- AFTER: The farmers will be delight and cultivate the crops in good manner.

8.2 OFFLINE CHANNELS [CH]

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

- Cultivators can collect the preventive measures from the other farmers
- Students can perform a case study and the visit the farm lands also

Define CS, fit into CL

Explore AS, differentiate

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

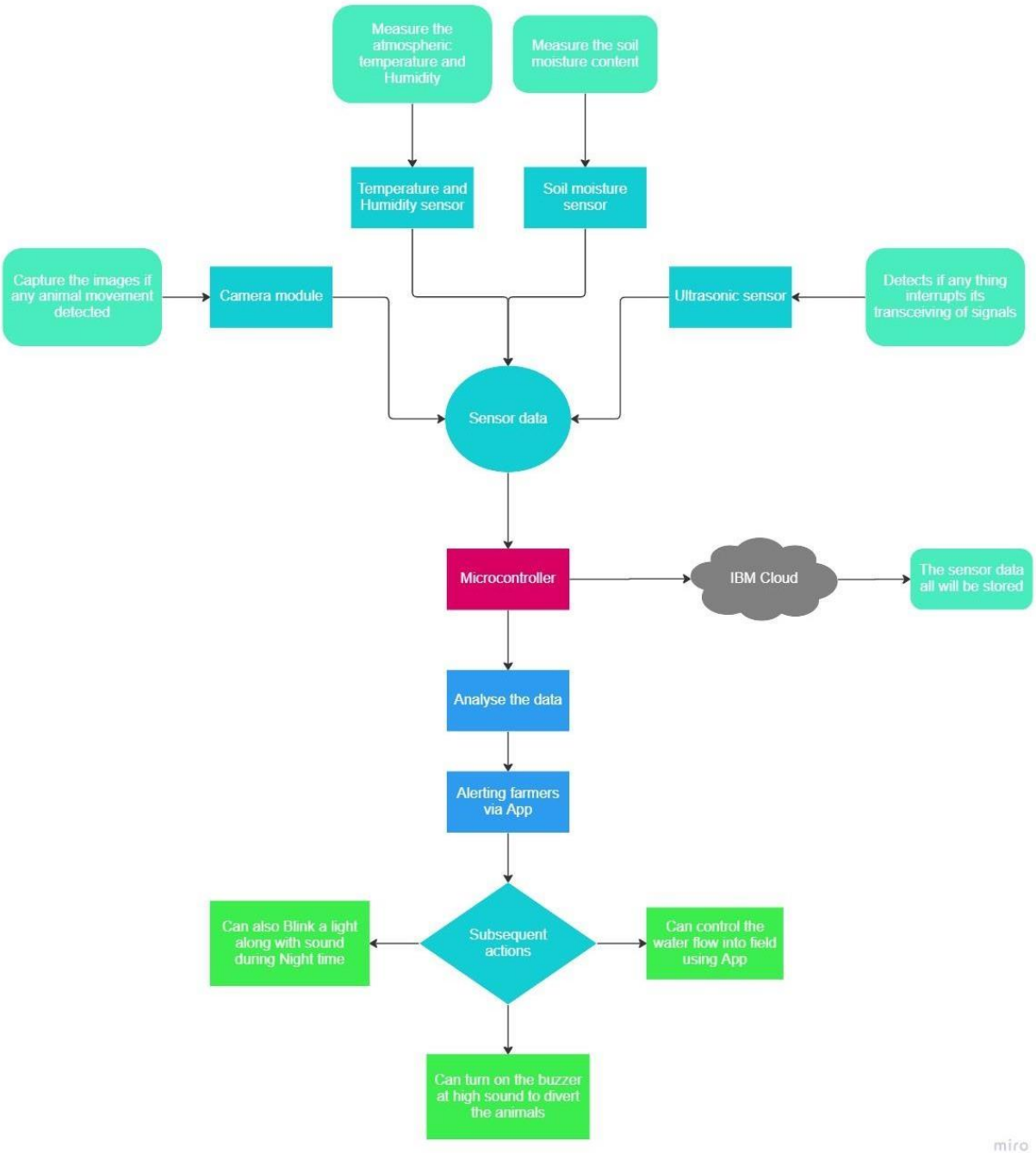
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Sensing animals approaching the crop field, the device sends the farmer an SMS and plays an alarm to scare them away.
FR-2	User Confirmation	Data such as sensor readings for temperature, humidity, and soil moisture are received by SMS.
FR-3	User Understanding	Information regarding the current state of farmed land is obtained based on sensor data values.
FR-4	User Action	Actions that must be taken by the user include crop residue destruction, deep ploughing, crop rotation, fertiliser application, strip cropping, and scheduled planting operations.

4.2 NON-FUNCTIONAL REQUIREMENT:

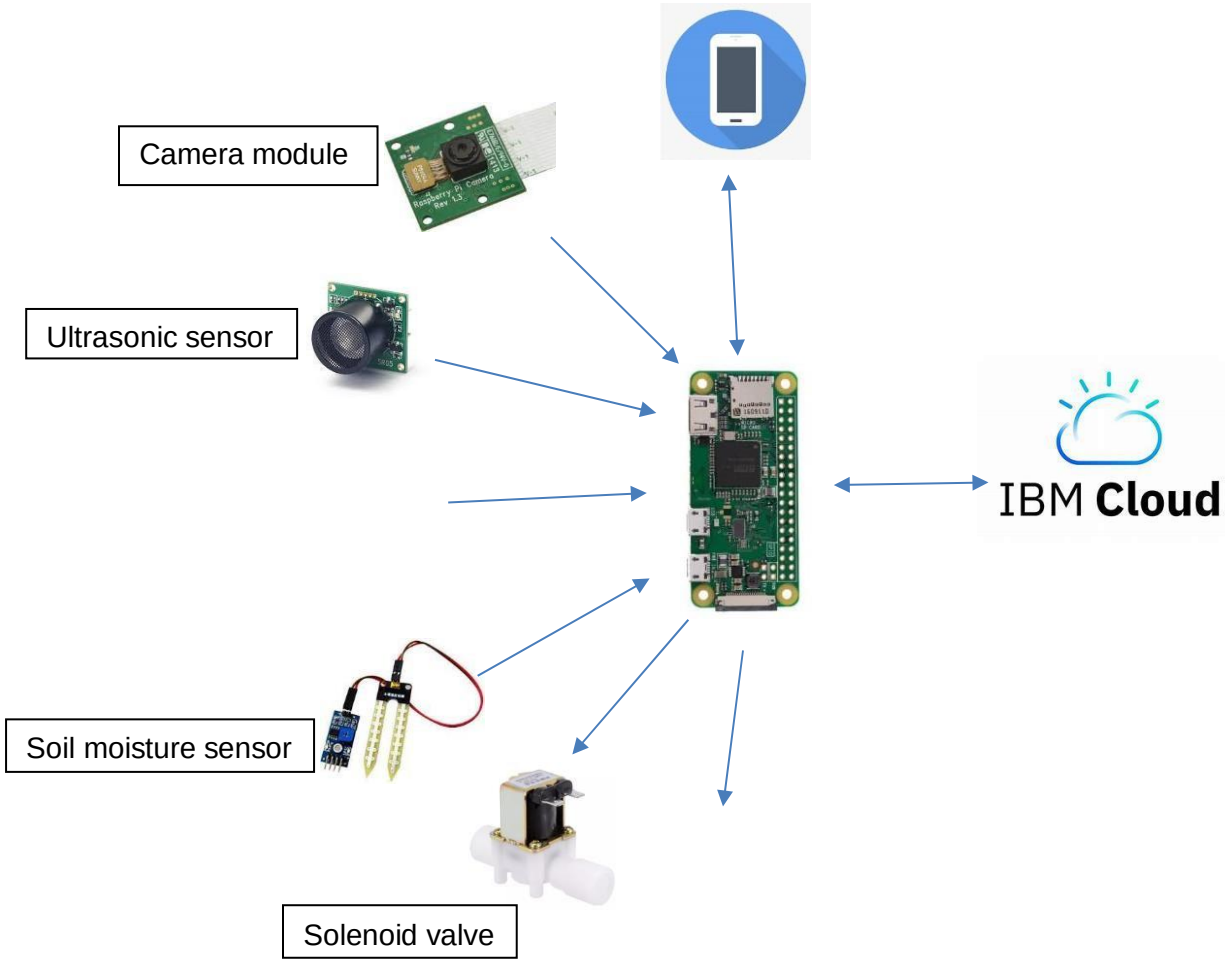
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Mobile assistance. Given the capabilities of mobile devices, users must be able to interact in the same roles and tasks on PCs and mobile devices when practicable.
NFR-2	**Security**	Authorized users of the system who share information must be able to register and communicate securely on devices with data that requires secure access.
NFR-3	**Reliability**	It has the ability to detect disturbances close to the field and doesn't issue an erroneous warning signal.
NFR-4	**Performance**	Regardless of the amount of data that is saved and the background analytics, it must offer users acceptable response speeds. Communications that are bidirectional and nearly real-time must be supported. The necessity to support industrial and device protocols at the edge is connected to this requirement.
NFR-5	**Availability**	For 24x7 operations, IoT solutions and domains require highly available systems. is not a vital production application, thus if the IoT solution goes down, neither operations nor production are Affected.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

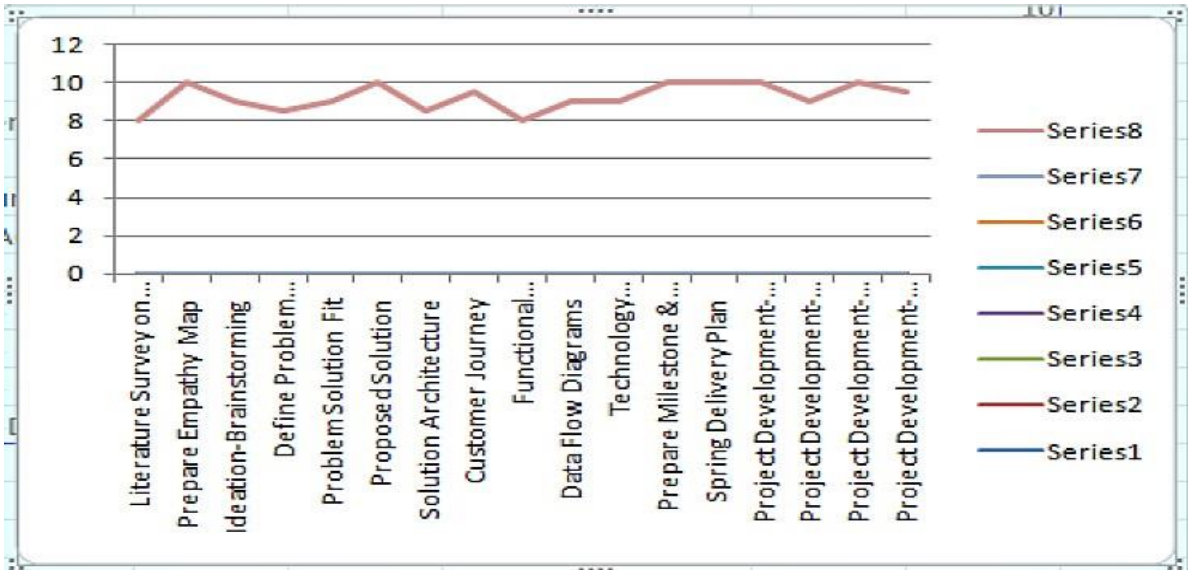


5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	User can enter into the web app	I can access my account / dashboard	High	Sprint-1
		USN-2	User can register with their corresponding mail, create account with password and Id	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	User can login to the application by entering the password and Id	I can register & access the dashboard with Facebook Login	High	Sprint-2
	Dashboard	USN-4	User can view the current statues of field such as temperature, humidity, Soil moisture.	I can view the parameters of the field's environment as a registered user.	Medium	Sprint-2
Customer (Web user)	Viewable	USN-1	Same procedure as mentioned above for signup and Login. User can view the data in the corresponding web page.	I can view the data from the sensors in the field.	Medium	Sprint -2
Customer of both users	Accessible	USN-1	The User can control the water flow into the field, also can Turn on the light, buzzer in the field with the help of the web's UI.	I can access the actuators and light, buzzer in the field remotely from anywhere.	High	Sprint- 2

6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION:



Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	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's calculate the team's average velocity (AV) per iteration unit (story points per day)

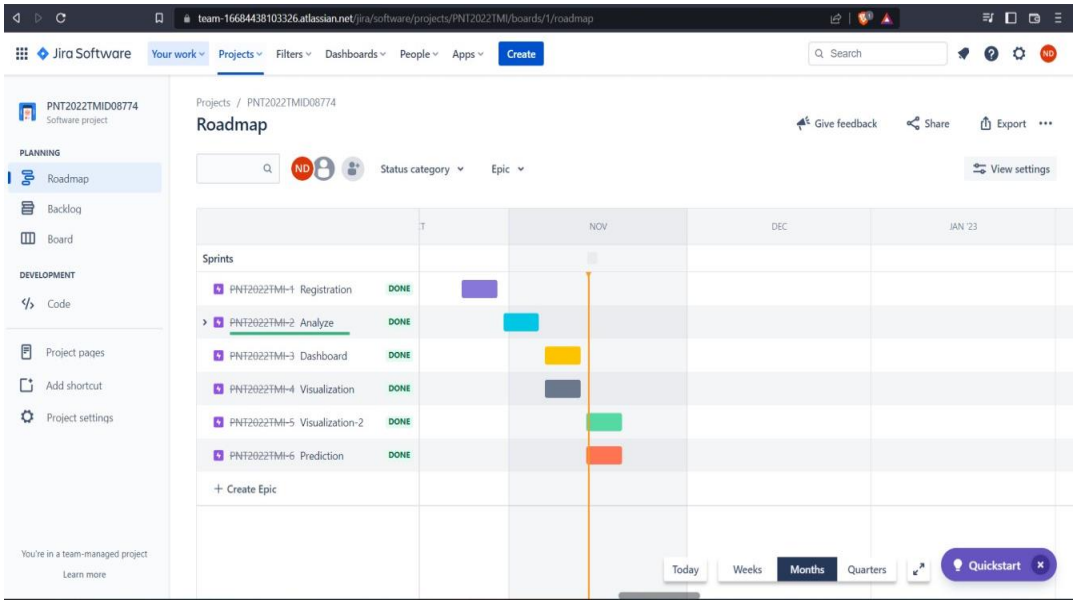
$$AV = \frac{Sprint\ Duration}{Velocity} = \frac{20}{10} = 2$$

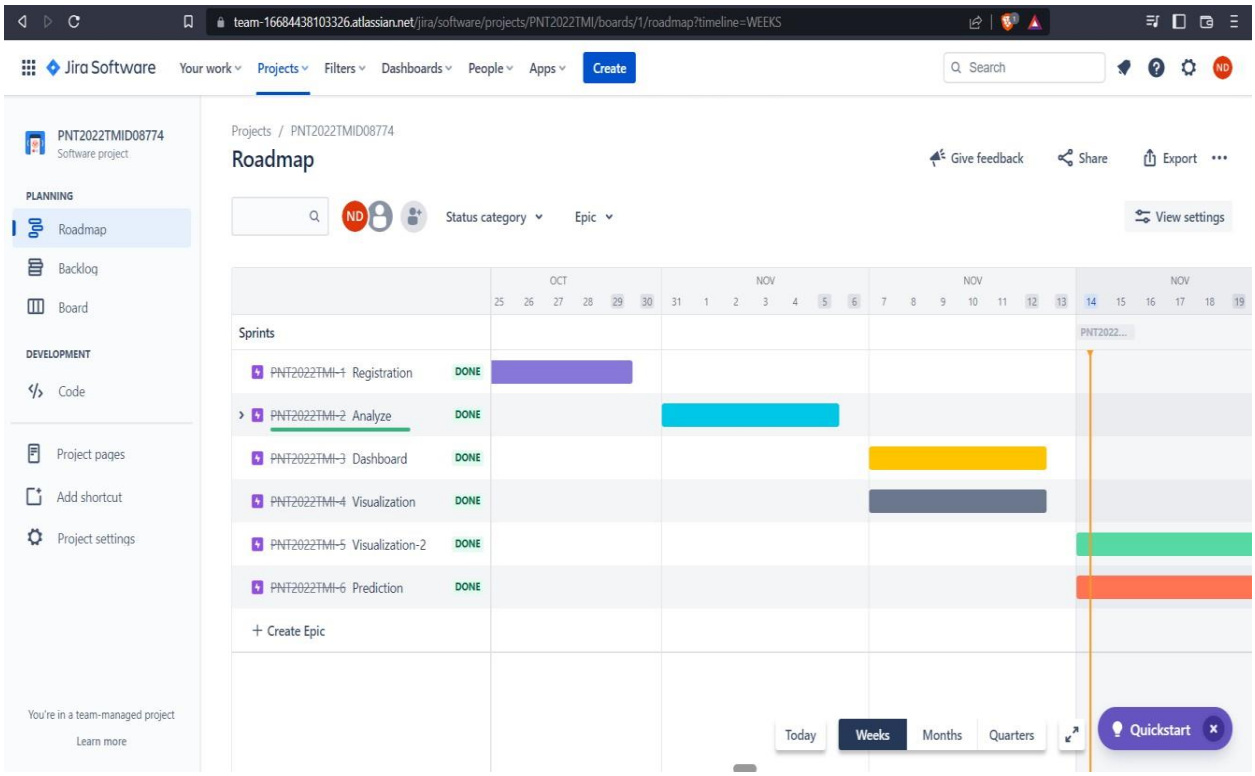
6.2 SPRINT DELIVERY SCHEDULE

Sprint	Functional Requirement (Epic)	User Number Story	User Story/Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	I can create account in IBM cloud and the data are collected.	20	High	2 Members
Sprint-2	Analyze	USN-2	All the data that are collected is cleaned and uploaded in the database or IBM cloud.	20	Medium	2 Members
Sprint-3	Dashboard	USN-3	I can use my account in my dashboard for uploading dataset.	10	Medium	2 Members
Sprint-3	Visualization	USN-4	I can prepare data for Visualization.	10	High	2 Members
Sprint-4	Visualization	USN-5	I can present data in my dashboard.	10	High	2 Members
Sprint-4	Prediction	USN-6	We can Protect the crops from the animals.	10	High	2 Members

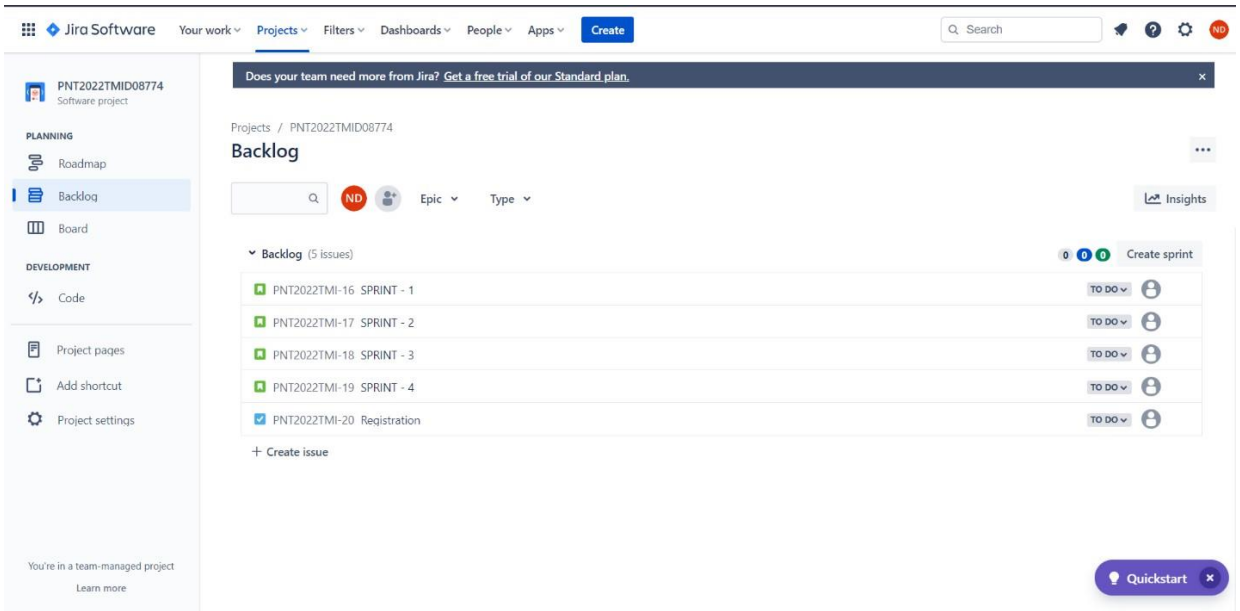
6.3 REPORTS FROM JIRA:

ROADMAP

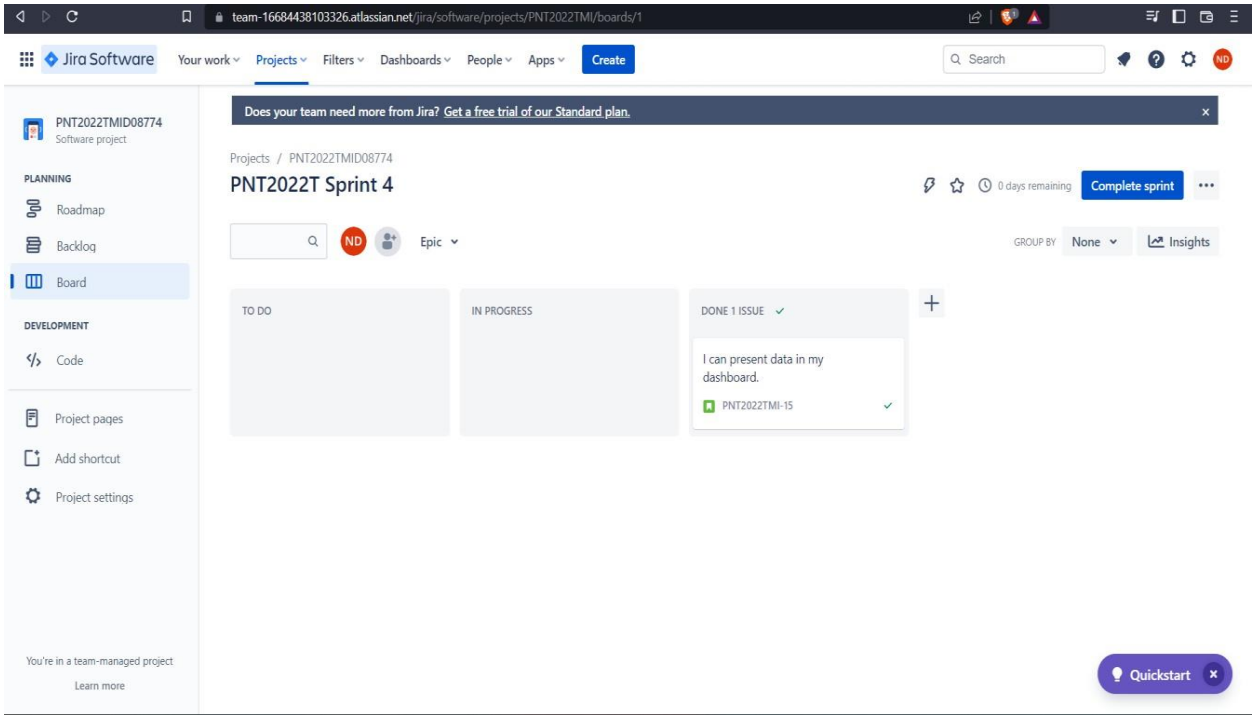



team-1668443810326.atlassian.net/jira/software/projects/PNT2022TMI/boards/1/roadmap?timeline=WEEKS
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
PNT2022TMID08774
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / PNT2022TMID08774
Roadmap
Give feedback
Share
Export
Status category
Epic
View settings
OCT
NOV
NOV
NOV
Sprints
PNT2022TMI-1 Registration DONE
PNT2022TMI-2 Analyze DONE
PNT2022TMI-3 Dashboard DONE
PNT2022TMI-4 Visualization DONE
PNT2022TMI-5 Visualization-2 DONE
PNT2022TMI-6 Prediction DONE
Create Epic
Today
Weeks
Months
Quarters
Quickstart

BACKLOG


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Does your team need more from Jira? Get a free trial of our Standard plan.
PNT2022TMID08774
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / PNT2022TMID08774
Backlog
Epic
Type
Insights
Backlog (5 issues)
Create sprint
PNT2022TMI-16 SPRINT - 1 TO DO
PNT2022TMI-17 SPRINT - 2 TO DO
PNT2022TMI-18 SPRINT - 3 TO DO
PNT2022TMI-19 SPRINT - 4 TO DO
PNT2022TMI-20 Registration TO DO
Create issue
Quickstart

BOARD


team-16684438103326.atlassian.net/jira/software/projects/PNT2022TMI/boards/1
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
PNT2022TMID08774
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / PNT2022TMID08774
PNT2022T Sprint 4
0 days remaining
Complete sprint
Epic
GROUP BY
None
Insights
TO DO
IN PROGRESS
DONE 1 ISSUE
I can present data in my dashboard.
PNT2022TMI-15
Quickstart

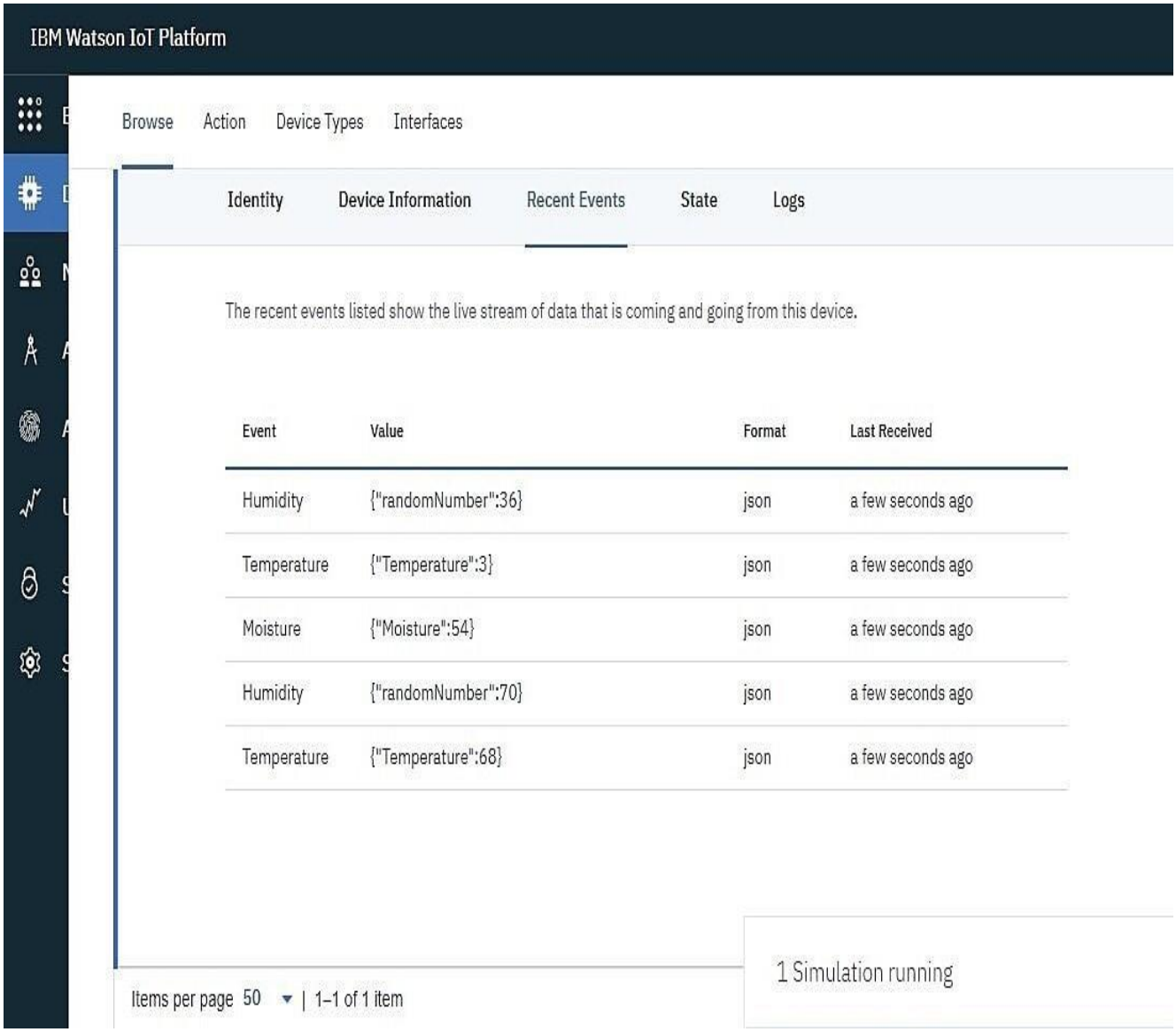
7. CODING AND SOLUTIONING

```
import random
import ibmiotf.application
import ibmiotf.device
 from time import sleep
 import sys
#IBM Watson Device Credentials.
 organization = "op701j"
 deviceType = "Lokesh"
deviceId = "Lokesh89"
 authMethod = "token"
 authToken = "1223334444"
 def myCommandCallback(cmd):
 print("Command received: %s" % cmd.data['command'])
status=cmd.data['command']
 if status=="sprinkler_on":
 print ("sprinkler is ON") else :
 print ("sprinkler 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()
#Connecting to IBM watson.
deviceCli.connect()
while True:
 #Getting values from sensors.
 temp_sensor = round( random.uniform(0,80),2)
 PH_sensor = round(random.uniform(1,14),3)
```

```
 camera = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected","Not Detected",]
camera_reading = random.choice(camera)

flame = ["Detected","Not Detected","Not Detected","Not Detected","Not Detected","Not Detected",] flame_reading
= random.choice(flame)

moist_level = round(random.uniform(0,100),2)

water_level = round(random.uniform(0,30),2)

#storing the sensor data to send in json format to cloud

. temp_data = { 'Temperature' : temp_sensor }

PH_data = { 'PH Level' : PH_sensor }

camera_data = { 'Animal attack' : camera_reading}

 flame_data = { 'Flame' : flame_reading }

 moist_data = { 'Moisture Level' : moist_level}

 water_data = { 'Water Level' : water_level}

# publishing Sensor data to IBM Watson for every 5-10 seconds.

 success = deviceCli.publishEvent("Temperature sensor", "json", temp_data, qos=0)

 sleep(1)

 if success:

 print (" .....publish ok..... ")

 print ("Published Temperature = %s C" % temp_sensor, "to IBM Watson")

success = deviceCli.publishEvent("PH sensor", "json", PH_data, qos=0)

sleep(1)

 if success:

print ("Published PH Level = %s" % PH_sensor, "to IBM Watson")

success = deviceCli.publishEvent("camera", "json", camera_data, qos=0)

 sleep(1)

 if success:

 print ("Published Animal attack %s " % camera_reading, "to IBM Watson")

 success = deviceCli.publishEvent("Flame sensor", "json", flame_data, qos=0)

 sleep(1)

 if success:

 print ("Published Flame %s " % flame_reading, "to IBM Watson")

success = deviceCli.publishEvent("Moisture sensor", "json", moist_data, qos=0)

 sleep(1)
```

```
if success:
 print ("Published Moisture Level = %s " % moist_level, "to IBM Watson")
success = deviceCli.publishEvent("Water sensor", "json", water_data, qos=0)
 sleep(1)
 if success:
print ("Published Water Level = %s cm" % water_level, "to IBM Watson")
print ("")
 #Automation to control sprinklers by present temperature an to send alert message to IBM Watson.
 if (temp_sensor > 35):
 print("sprinkler-1 is ON")
 success = deviceCli.publishEvent("Alert1", "json",{ 'alert1' : "Temperature(%s) is high, sprinkerlers are turned ON" %temp_sensor } , qos=0)
sleep(1)
if success:
print( 'Published alert1 : ', "Temperature(%s) is high, sprinkerlers are turned ON" %temp_sensor,"to IBM Watson")
 print("")
 else:
print("sprinkler-1 is OFF")
print("")
#To send alert message if farmer uses the unsafe fertilizer to crops.
if (PH_sensor > 7.5 or PH_sensor < 5.5):
success = deviceCli.publishEvent("Alert2", "json",{ 'alert2' : "Fertilizer PH level(%s) is not safe,use other fertilizer" %PH_sensor } , qos=0)
 sleep(1)
if success:
print('Published alert2 : ' , "Fertilizer PH level(%s) is not safe,use other fertilizer" %PH_sensor,"to IBM Watson")
 print("")
#To send alert message to farmer that animal attack on crops.
if (camera_reading == "Detected"):
 success = deviceCli.publishEvent("Alert3", "json", { 'alert3' : "Animal attack on crops detected" }, qos=0)
 sleep(1)
if success:
print('Published alert3 : ' , "Animal attack on crops detected","to IBM Watson","to IBM Watson") print("")
```

```
#To send alert message if flame detected on crop land and turn ON the splinkers to take immediate action.

 if (flame_reading == "Detected"):

print("sprinkler-2 is ON")

 success = deviceCli.publishEvent("Alert4", "json", { 'alert4' : "Flame is detected crops are in danger,sprinklers turned ON" }, qos=0)

 sleep(1)

if success:

print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned ON","to IBM Watson")

#To send alert message if Moisture level is LOW and to Turn ON Motor-1 for irrigation. if (moist_level < 20):

print("Motor-1 is ON")

 success = deviceCli.publishEvent("Alert5", "json", { 'alert5' : "Moisture level(%s) is low, Irrigation started" %moist_level }, qos=0)

 sleep(1) if success:

 print('Published alert5 : ' , "Moisture level(%s) is low, Irrigation started" %moist_level,"to IBM Watson" )

print("")

#To send alert message if Water level is HIGH and to Turn ON Motor-2 to take water out.

if (water_level > 20):

 print("Motor-2 is ON")

 success = deviceCli.publishEvent("Alert6", "json", { 'alert6' : "Water level(%s) is high, so motor is ON to take water out " %water_level }, qos=0)

 sleep(1)

 if success:

 print('Published alert6 : ' , "water level(%s) is high, so motor is ON to take water out " %water_level,"to IBM Watson" )

print("")

 #command recived by farmer

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect().
```


IBM Watson IoT Platform
Browse
Action
Device Types
Interfaces
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
Humidity
{"randomNumber":36}
json
a few seconds ago
Temperature
{"Temperature":3}
json
a few seconds ago
Moisture
{"Moisture":54}
json
a few seconds ago
Humidity
{"randomNumber":70}
json
a few seconds ago
Temperature
{"Temperature":68}
json
a few seconds ago
Items per page 50
1–1 of 1 item
1 Simulation running

7.1 FEATURE – 1

Output: Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor. Power supply: 5V-12V input voltage for most modules (they have a3.3V regulator), but 5V is ideal in case the regulator has different specs.

BUZZER

Specifications

- Rated Voltage : 6V DC
- Operating Voltage : 4 to 8V DC
- Rated Current*: ≤30mA
- Sound Output at 10cm* : ≥85dB
- Resonant Frequency : 2300 ±300Hz
- Tone: Continuous A buzzer is a loud noise maker.

Most modern ones are civil defense or air- raid sirens, tornado sirens, or the sirens on emergency service vehicles such as ambulances, police cars and fire trucks. There are two general types, pneumatic and electronic.

7.2 FEATURE – 2

i. Good sensitivity to Combustible gas in wide range.
ii. High sensitivity to LPG, Propane and Hydrogen.
iii. Long life and low cost.
iv. Simple drive circuit.

8. TESTING

8.1 TEST CASES:

			Team ID:	PNT2022TMID08774				
			NFT - Risk Assessment					
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Load/Volume Changes	Risk Score	Justification
1	Motor ON/OFF	Existing	Moderate	No Changes	Moderate	>10 to 30%	ORANGE	Changes occurs less
2	Sensor values	Existing	Moderate	No Changes	Moderate	>10 to 30%	ORANGE	Some changes occurs

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Approvals/SignOff	Assumptions/Dependencies/Risk
1	Python script	Python coding	https://www.python.org/psf/sponsors/#heroku	Depend on the delivered code
2	Node Red	Sensor & command values	https://nodered.org/	Sensor values
3	MIT App Inventor	Motor control/Sensors notification	https://appinventor.mit.edu/about/termsofservice	Notifications

End Of Test Report								
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Identified Defects (Detected/Closed/Open)	Recommendations	Approvals/SignOff
1	Python Code	Python coding	Met	Pass	GO	Closed	Efficient code	https://www.python.org/psf/sponsors/#heroku
2	Node Red	Sensors&command values	Met	Pass	GO	Closed	Sensing the values perfectly	https://nodered.org/
3	MIT App Inventor	tor control/Sensors notificat	Met	Pass	GO	Closed	Notifies the users at correct time	https://appinventor.mit.edu/about/termsofservice

8.2 USER ACCEPTANCE TESTING:

Defect Analysis

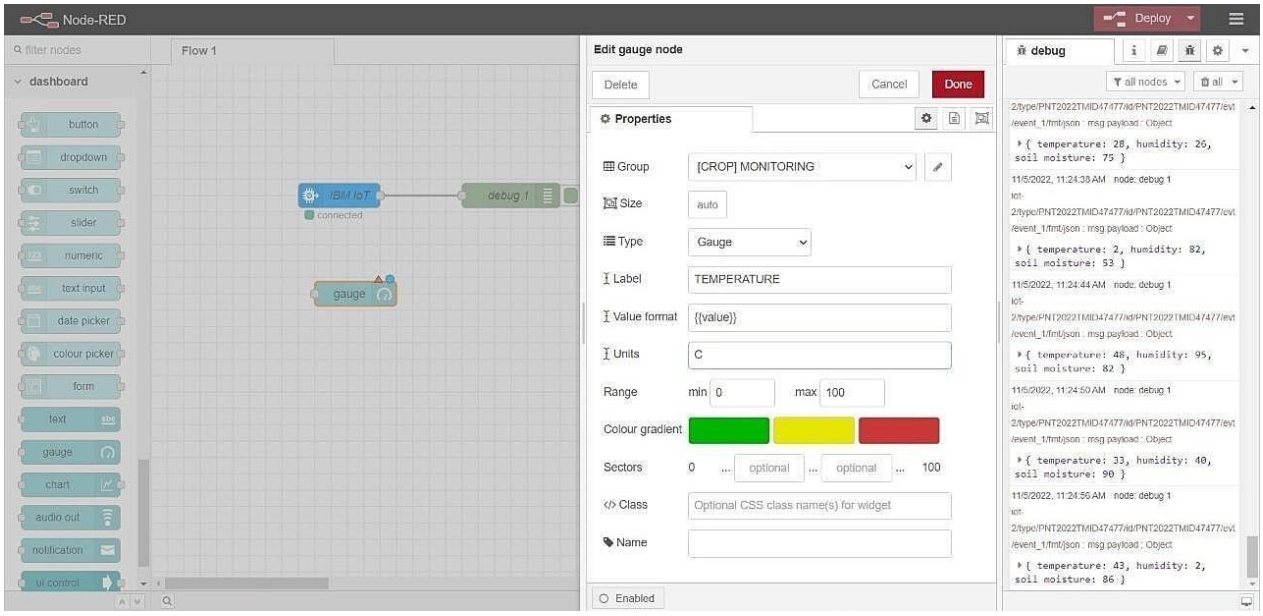
This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity2	Severity3	Severity4	Subtotal
By Design	11	4	2	2	19
Duplicate	1	1	2	0	4
External	2	3	0	1	6
Fixed	10	2	3	20	35
Not Reproduced	0	0	2	0	2
Skipped	0	0	2	1	3
Won't Fix	0	5	2	1	8
Totals	24	15	13	25	77

Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	1	4
Client Application	47	0	2	45
Security	3	0	0	3
Outsource Shipping	2	0	0	2
Exception Reporting	11	0	2	9
Final Report Output	5	0	0	5
Version Control	3	0	1	2


Node-RED
Deploy
filter nodes
Flow 1
dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
IBM IoT
connected
debug 1
gauge
Edit gauge node
Delete
Cancel
Done
Properties
Group
[CROP] MONITORING
Size
auto
Type
Gauge
Label
TEMPERATURE
Value format
{{value}}
Units
C
Range
min 0
max 100
Colour gradient
Sectors
0
optional
optional
100
Class
Optional CSS class name(s) for widget
Name
Enabled
debug
all nodes
all
{ temperature: 28, humidity: 26, soil moisture: 75 }
{ temperature: 2, humidity: 82, soil moisture: 53 }
{ temperature: 48, humidity: 95, soil moisture: 82 }
{ temperature: 33, humidity: 40, soil moisture: 90 }
{ temperature: 43, humidity: 2, soil moisture: 86 }

9. RESULTS

The problem of crop vandalization by wild animals andfire has become a major social problem in current time.

It requires urgent attention as no effective solution existstill date for this problem. Thus this project carries a greatsocial relevance as it aims to address this problem. This project will help farmers in protecting their orchards and fields and save them from significant financial losses andwill save them from the unproductive efforts that they endure for the protection their fields. This will also helpthem in achieving better crop yields thus leading to their economic wellbeing.

10.ADVANTAGES AND DISADVANTAGES

ADVANTAGE:

Controllable food supply. you might have droughts or floods, but ifyou are growing the crops and breeding them to be hardier, you have a better chance of not straving. It allows farmers to maximize yields using minimum resources such as water fertilizers.

DISADVANTAGES:

The main disadvantage is the time it can take to process the information.in order to keep feeding people as the population grows you have to radically change the environment of the planet.

11. CONCLUSION

A IoT Web Application is built for smart agricultural system using Watson IoTplatform, Watson simulator, IBM cloud and Node-RED.

12. FUTURE SCOPE

In the future, there will be very large scope, this project can be made based on Image processing in which wild animal and fire can be detectedby cameras and if it comes towards farm then system will be directly activated through wireless networks. Wild animals can also be detected by using wireless networks such as laser wireless sensors and by sensingthis laser or sensor’s security system will be activated.

13. APPENDIX

SOURCE CODE:

```
import time importsys

import ibmiotf.application

 # toinstallpip install ibmiotf importibmiotf.device

# Provide your IBM Watson Device Credentials organization =
"8gyz7t" # replace the ORG ID deviceType = "weather_monitor"
#replace the Device type deviceId = "b827ebd607b5" # replace
Device ID authMethod = "token" authToken =
"LWVpQPaVQ166HWN48f" # Replace the authtoken
def myCommandCallback(cmd):  # function for Callbackif


    cm.data['command'] == 'motoron':
print("MOTOR ON IS RECEIVED")

elif cmd.data['command'] == 'motoroff':print("MOTOR OFF IS

RECEIVED")if cmd.command == "setInterval":

else:

if 'interval' not in cmd.data:
  print("Error - command is missing requiredinformation: 'interval'")


  interval =

cmd.data['interval']elif

cmd.command == "print":
if 'message' not in cmd.data:
        print("Error - commandis missing requiredinformation:
        'message'")else:output = cmd.data['message']
        print(output)



try:
    deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId,"authmethod":authMethod,
                "auth-token": authToken}          deviceCli
```

```
= ibmiotf.device.Client(deviceOptions)#
.....

exceptException 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 oftype
   "greeting" 10 times
deviceCli.connect()

while True:
        deviceCli.commandCallback = myCommandCallback


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

SENSOR.PY

```
   import time import
   sysimport
   ibmiotf.application
   importibmiotf.devi
   ce
import random


   # Provide your IBM Watson Device Credentials organization =
   "8gyz7t" # replace the ORG ID deviceType = "weather_monitor"
   #replace the Device type deviceId = "b827ebd607b5" # replace
   Device ID authMethod = "token"authToken =
   "LWVpQPaVQ166HWN48f" # Replace the authtoken

def myCommandCallback(cmd):


        print("Command received: %s" %
   cmd.data['command'])print(cmd)


 try:
                deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
   "auth-method": authMethod, "auth-token":
   authToken}deviceCli =
   ibmiotf.device.Client(deviceOptions)
   #.....
```

```
exceptException 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 oftype
  "greeting" 10 times
deviceCli.connect()


while True:
        temp=random.randint(0,1
  00)
  pulse=random.randint(0,100)
        soil=random.randint(0,10
        0)
        data = { 'temp' : temp, 'pulse': pulse
        ,'soil':soil}#print data          def
  myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%"
  %pulse,"Soil Moisture = %s %%" % soil,"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 FLOW:

```
[
{
"id":"625574ead9839b
34",
"type":"ibmiotout",
"z":"630c8601c5ac3295",
"authentication":"apiKey",
"apiKey":"ef745d48e395ccc0",
"outputType":"cmd",
```

```
"deviceId":"b827ebd607b5",
"deviceType":"weather_monitor",
"eventCommandType":"data",
"format":"json",
"data":"data",
"qos":0,
"name":"IBM
IoT",
"service":"registe
red","x":680,
"y":220,
"wires":[]
},
{
"id":"4cff18c3274cccc4","type":"ui_button",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c
","order":2,
"width":"0",
"height"

"passthru":false,
"label":"MotorON"
,
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoron\"
}","payloadType":"str",
"topic":"motoron",
"topicType":
"s
tr","x":360,
"y":160, "wires":[["625574ead9839b34"]]},
{
"id":"659589baceb4e0b0",
"type":"ui_button",
```

"z":"630c8601c5ac3295","name":"",
"group":"716e956.00eed6c
","order":3,
"width":"0",
"height":"0",
"passthru":true,
"label":"Motor
OFF",
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoroff\"
}","payloadType":"str",
"topic":"motoroff",
"topicType":
"s
tr","x":350,

"y":220, "wires":[["625574ead9839b34"]]},

{"id":"ef745d48e395ccc0","type":"ibmio
t",
"name":"weather_monitor","keepalive":"6
0","serverName":"",
"cleansession":tru
e,"appId":"",
"shared":false},
{"id":"716e956.00eed6c",
"type":"ui_group",
"name":"Form",
"tab":"7e62365e.b7e6b
8","order":1,
"disp":true,
"width":"6",
"collapse":f
alse},
{"id":"7e62365e.b7e6b8",
"type":"ui_tab",

"name":"contorl
",
"icon":"dashboar
d","order":1,
"disabled":false,
"hidden":false}
]

[
{
"id":"b42b5519fee73ee2", "type":"ibmiotin",
"z":"03acb6ae05a0c712",
"authentication":"apiKey",
"apiKey":"ef745d48e395ccc0",

"inputType":"evt",
"logicalInterface":"",
"ruleId":"",
"deviceId":"b827ebd607b
5","applicationId":"",
"deviceType":"weather_monitor",

"eventType":"+",
"commandType":"",
"format":"json",
"name":"IBMIoT",
"service":"registered",
"allDevices":"",
"allApplications":"",
"allDeviceTypes":"",
"allLogicalInterfaces":
"","allEvents":true,
"allCommands":"",
"allFormats
":"",
"qos":0,
"x":270,
"y":180,
"wires":[["50b13e02170d73fc","d7da6c2f5302ffaf","a949797028158f3f","a71f164bc3 78bcf1"]]
},

```
{
"id":"50b13e02170d73
fc",
"type":"function",
"z":"03acb6ae05a0c7
12","name":"Soil
Moisture",
   "func":"msg.payload = msg.payload.soil;\nglobal.set('s',msg.payload);\nreturn msg;",
   "outputs":1,
"noerr":
0,
"initializ
e":"",
"finalize":"",
"libs":[],

"x":490,
"y":120,
"wires":[["a949797028158f3f","ba98e701f55f04fe"]]
},

{
"id":"d7da6c2f5302ffaf","type":"function"
,"z":"03acb6ae05a0c712",
"name":"Humidity",
   "func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn msg;",
   "outputs":1,
"noerr":
0,
"initializ
e":"",
"finalize":"",
"l
i
bs
":
[
],
"
x
":
48
```

0,
"y":260, "wires":[["a949797028158f3f","70a5b076eeb80b70"]]
},
{
"id":"a949797028158f
3f",
"type":"debug",
"z":"03acb6ae05a0c7
12
","name":"IBMo/p",
"active":true,
"tosidebar":true,
"console":false,
"tostatus":false,
"complete":"payload"
, "targetType":"msg",
"statusVal":"",
"statusType":"auto
","x":780,
"y":180,
"wires":[]
},

{
"id":"70a5b076eeb80b70",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a
4","order":6,
"width":"0",
"height":"0",
"gtype":"gage",
"title":"Humidity",
"label":"Percentage(%)",
"format":"{{value}}
","min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"],
"seg1":"","seg2":"",
"classNam

```
e
":"","x":86
0,
"y":260,
"wires":[]
},
{
"id":"a71f164bc378bcf1","type":"function",
"z":"03acb6ae05a0c712",
"name":"Temperature",
  "func":"msg.payload=msg.payload.temp;\nglobal.set('t',msg.payload);\nreturn msg;","outputs":1,
"noerr":
0,
"initializ
e":"",
"finalize":"
","li
bs
":[
],

"
x
":
49
0,
"y":360
,

"wires":[["8e8b63b110c5ec2d","a949797028158f3f"]]
},
{
"id":"8e8b63b110c5ec2d",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a
4","order":11,
"width":"0",
"height":"0",
"gtype":"gage",
```

"title":"Temperature",
"label":"DegreeCelciu
s",
"format":"{{value}}"
, "min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"],"seg1":"",
"seg2":"",
"classNam
e":"",
"x":790,
"y":360,
"wires":[]
},
{
"id":"ba98e701f55f04fe",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a
4","order":1,
"width":"0",
"height":"0",
"gtype":"gage",

"title":"Soil
Moisture",
"label":"Percentage(%
)",
"format":"{{value}}
","min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"],"seg1":"",
"seg2":"",
"classNam
e":"",
"x":790,
"y":120,
"wires":[]
},

```
{
"id":"a259673baf5f0f
98","type":"httpin",
"z":"03acb6ae05a0c7
12","name":"",
"url":"/sensor",
"method":"g
et",
"upload":fals
e,
"swaggerDo
c"
:"","x":37
0,"y":500,
"wires":[["18a8cdbf7943d27a"]]
},
{
"id":"18a8cdbf7943d27a","type":"function",
"z":"03acb6ae05a0c712",
"name":"httpfunction",
  "func":"msg.payload{\"pulse\":global.get('p'),\"temp\":global.get('t'),\"soil\":global.get( 's')};\nreturn
  msg;",
"outputs":1,
"noerr":0,

"initialize":"",
"finalize":"
","li
bs
":[
],
"
x
":
63
0,
"y":500, "wires":[["5c7996d53a445412"]]
},
{
"id":"5c7996d53a4454
12"type":"httprespon",
```

"z":"03acb6ae05a0c71
2","name":"",
"statusCode":"",
"header
s":{},
"x":870,
"y":500,
"wires":[]
},
{
"id":"ef745d48e395ccc0",
"type":"ibmiot",
"name":"weather_monitor
","keepalive":"60",
"serverName":"",
"cleansession":tru
e,"appId":"",
"shared":false},
{
"id":"f4cb8513b95c98a4","type":"ui_group",
"name":"monitor",
"tab":"1f4cb829.2fde
e8","order":2,
"disp":
true,
"widt
h
":"6",
"collapse":
false,
"classNam
e":""
},{
"id":"1f4cb829.2fdee8",
"type":"ui_tab",
"name":"Home",
"icon":"dashboar
d","order":3,
"disabled":false,
"hidden":false }

14. GitHub & Project Demo Link

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-8043-1658907089

Demo Video Link:

https://drive.google.com/file/d/1YS9mfrz2v0wob2FG8SX4XeQuooLDjwnQ/view?usp=share_link