IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID : PNT2022TMID41830

Team Members

Abinaya V
Abirami G
Deepika VS
Gowri R

INTRODUCTION

PROJECT OVREVIEW:

Crops in farms are many times ravagedby local animalslike buffaloes, cows, goats, birds etc. this leads to huge losses for the farmers. It is not possible for farmers tobarricade entire fields or stay on field 24 hours and guard it.so here we proposeautomatic crop protection system from animals. This is a microcontroller based system using PIC family microcontroller. The microcontroller 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 safetyof crop from animals thus protecting farmers loss.

PURPOSE:

Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal and fire. These intruder alert protect the crop that damaging that indirectly increase yield of the crop. The develop system will not harmful and injurious to animal as well as human beings. Theme of project is to design a intelligent security system for farm

protecting by using embedded system.

LITERATURE SURVEY

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 andmanual surveillance and various such exhaustive and dangerous method.

REFERENCES:

i. Mr.Pranav shitap, Mr.Jayesh redij, Mr.Shikhar Singh, Mr.Durvesh Zagade, Dr. Sharada Chougule. Department of ELECTRONICS AND TELECOMMUNICATION ENGINEERING, Finolex Academy of Management and technology, ratangiri, India.

ii. N.Penchalaiah, D.Pavithra, B.Bhargavi, D.P.Madhurai,

K.EliyasShaik,S.Md.sohaib.Assitant Professor, Department of CSE,AITS, Rajampet,India UG Student, Department of CSE,AITS,Rajampet, India.
iii. Mr.P.Venkateswara Rao, Mr.Ch Shiva Krishna ,MR M Samba Siva ReddyLBRCE,LBRCE,LBRCE.
iv. Mohit Korche,Sarthak Tokse, ShubhamShirbhate, Vaibhav Thakre,S. P. Jolhe(HOD). Students , Final Year,Dept.of Electrical engineering,Government
College of engineering,Nagpur head of dept.,Electrical engineering,Government
College of engineering,Nagpur.

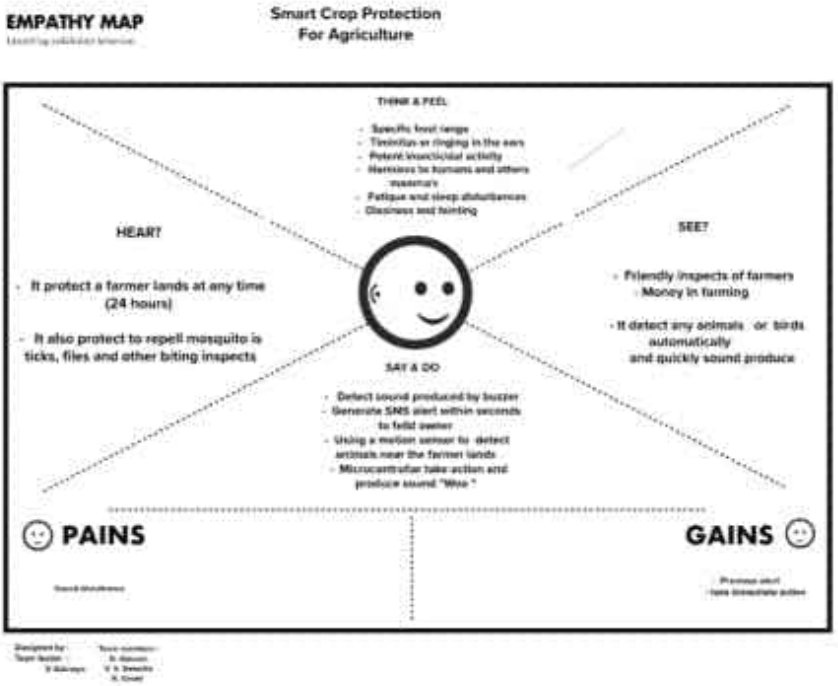
PROBLEM STATEMENT DEFINITION STATEMENT:

In the world economyof many Countrydependent upon the agriculture.
In spite of economic development agriculture is the backbone of the economy.
Crops in forms are many times ravaged by local animals like buffaloes, cows,
goats, birds and fire etc. this leads to huge loss for the farmers.it is not possible for
farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets
food requirements of the people and produces several raw materialsfor industries.
But because of animal interference and fire in agricultural lands, there will be huge
loss of crops. Crops will be totally getting destroyed.

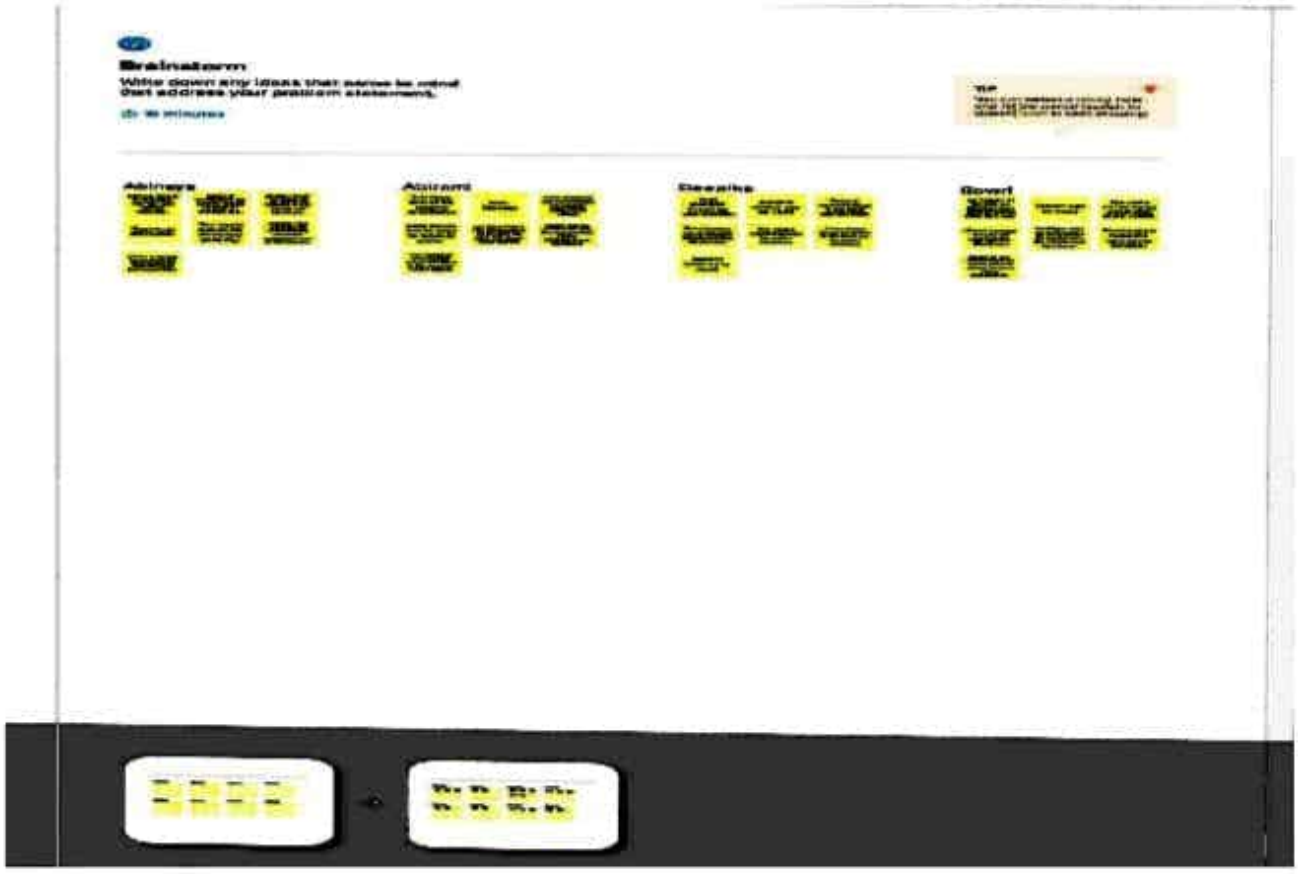
But because of animal interference and fire in agricultural lands, there will be huge
loss of crops. Crops will be totally getting destroyed.

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:



IDEATION AND BRAINSTORMING:



Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

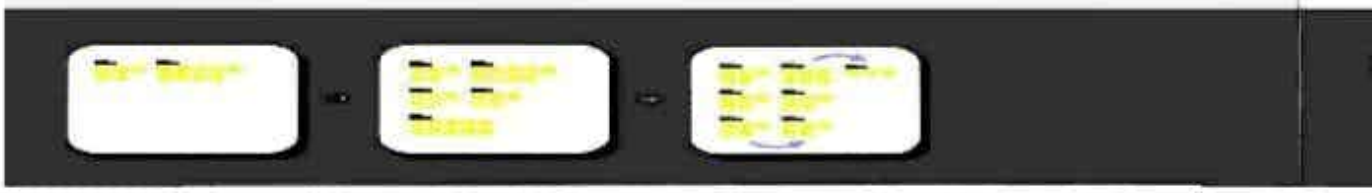
⏱ 20 minutes

Hardware device

UI interface

Emergency alert

Accessibility



PROPOSED SOLUTION:

S.NO	PARAMETER	DESCRIPTION
	Problem Statement (Problem to be solved)	Usually cro s in the fields are attacked against bowds aned other unknown edisturbances by humans. This take an enormous amount of time.Creating a smart automatic system will benefit the farmers in many edifferent ways.
	Idea / Solution edescription	Smart farming has enabled farmers to reduce waste aned enhance roeductivity with the hel of sensors. (light, humedity, tem erature, soil moisture,etc..).
	Novelty / Uniqueness	Role of SENSORS : IOT smart agriculture roeducts are edesigned to hel monitor cro fields using sensors aned by automating irrigation systems. As a result, farmers aned associated brameds can easily monitor the feild coneditions from anywhere without any hassle.
	Social im act / Customer Satisfaction	Water conservation , Saves lot of time , Increased quality of roeduction, Real time edata aned reeduction insight, Remote monitorng.

	Business Moedel (Revenue Moedel)	
	Scalability of the Solution	Scalability in smart farming refers to the aeda tability of a system to increase the ca acity , the number of technology edevices such as sensors aned fuctuators

	Business Moedel (Revenue Moedel)	
	Scalability of the Soluton	Scalability in smart farming refers to the aeda tability of a system to increase the ca acity , the number of technology edevices such as sensors aned fuctuators

PROBLEM SOLUTIONFIT:

PROBLEM SOLUTIONFIT:

REQUIREMENT ANALYSIS
FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	Install the app. Signing up with Gmail or phone number Creating a profile. Understand the guidelines.
2	User Confirmation	Email or phone number verification required via OTP.
3	Accessing datasets	Data's are obtained by cloudant DB.
4	Interface sensor	Connect the sensor and the application When animals enter the field , the alarm is generated.
5	Mobile application	It is used to control motors and field sprinklers.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
1	Usability	This project's contributes the farm protection

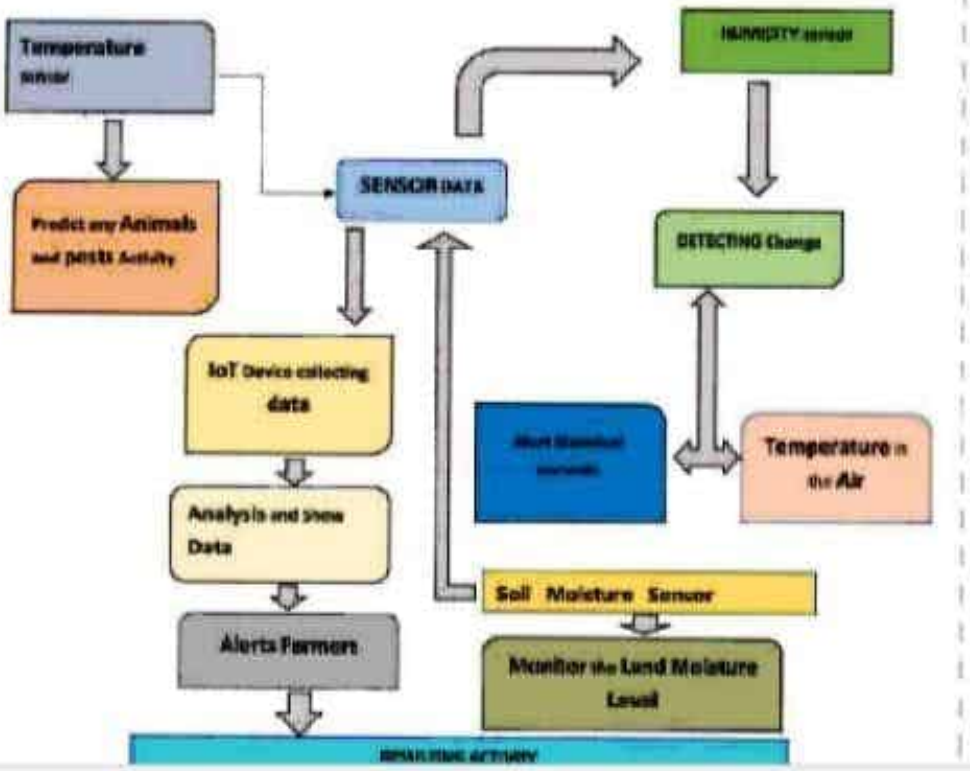
Non-functional Requirements:

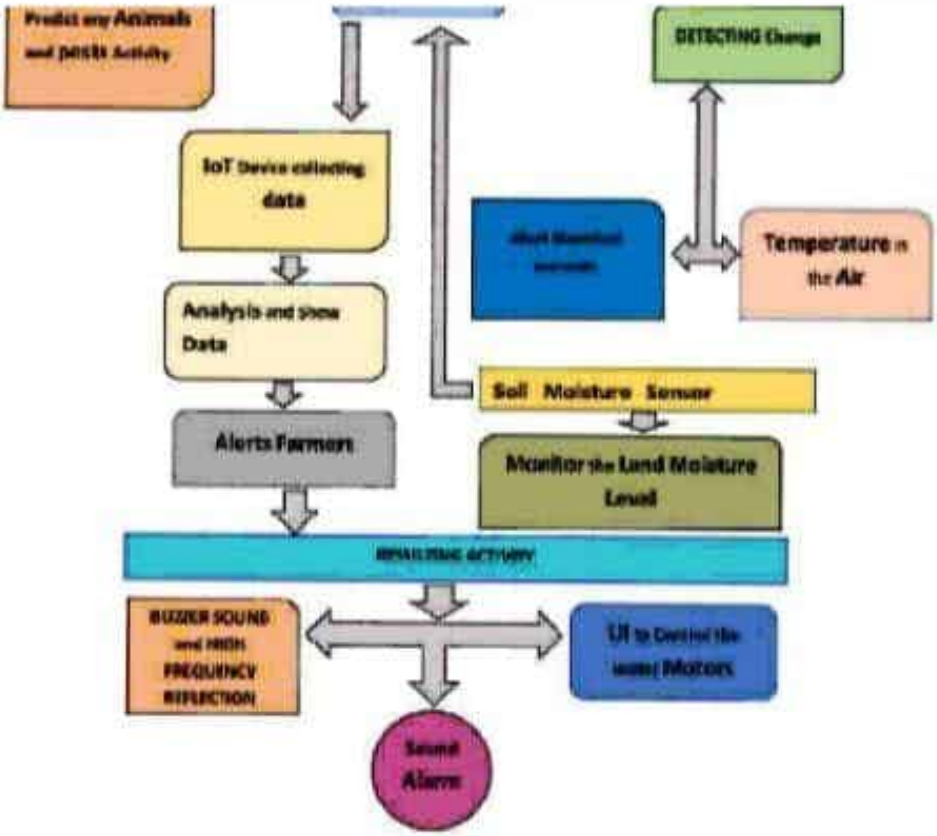
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
1	Usability	This project's contributes the farm protection through the smart protection system.
2	Security	It was created to protect the crops from animals.
3	Reliability	Farmers are able to safeguard their lands by help of this technology. They will also benefits from higher crop yields, which will improve our economic situation.
4	Performance	When animals attempt to enter the field, IOT devices and sensors alert the farmer via message.
5	Availability	We can defend the crops against wild animals by creating and implementing resilient hardware and software.
6	Scalability	This system's integration of computer vision algorithms with IBM cloudant services makes it more efficient to retrieve photos at scale, enhancing scalability.

PROJECT DESIGN :

DATA FLOW DIAGRAM:




Predict any Animals and pests Activity
DETECTING Change
IoT Device collecting data
Temperature in the Air
Analysis and show Data
Soil Moisture Sensor
Alerts Farmers
Monitor the Land Moisture Level
BUZZER SOUND and HIGH FREQUENCY REFLECTION
Sound Alarm

SOLUTION AND TECHNICAL ARCHITECTURE:

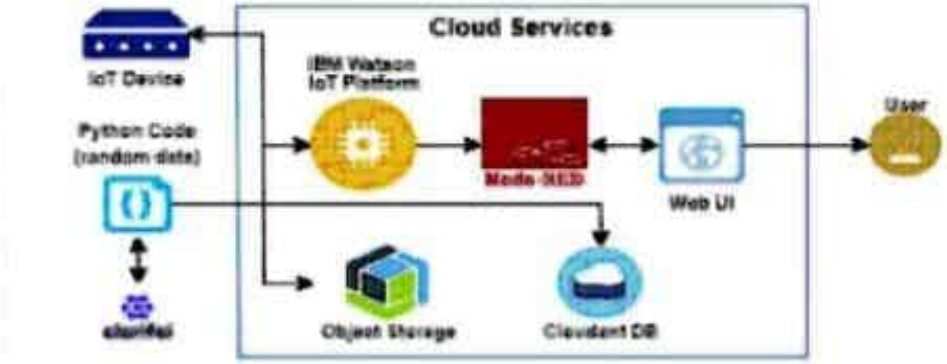


Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with the Web UI	App development
2.	Application Logic-1	Logic for a process in the application	Python Objectives
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the	Node-RED service

S.No	Characteristics	Description	Technology
1.	Open-source Frameworks	The open- source frameworks used	SAN-SAF
2.	Security Implementations	List all the security / access controls implemented	IBM cloud encryptions
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	IBM cloud Architecture
4.	Availability	Justify the availability of applications (e.g. use of load balancers, distributed servers etc.)	Web Application can even be used by the framers in the horticulture
5.	Performance	Design consideration for the performance of the application	Since the web application is high efficient, it can be used by the farmers irrespective of time

USER STORIES:

ProductBacklog,SprintSchedule,andEstimation(4Marks)

Usethebelowtemplatetocreateproductbacklogandsprintschedule

Sprint	Functional Requirement(Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Member
Sprint-1	IBM Cloud services	US-1	Create the IBM Cloud services which are being used in this project.	4	High	Ahmayn Abraam Gowri Deepika
Sprint-1	IBM Cloud services	US-2	Configure the IBM Cloud services which are being used in completing this project.	4	Medium	Ahmayn Abraam Gowri Deepika
Sprint-2	IBM Watson IoT platform	US-3	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	5	Medium	Ahmayn Abraam Gowri Deepika
Sprint-2	IBM Watson IoTplatform	US-4	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watso IoTplatform and get the device credentials.	5	High	Ahmayn Abraam Gowri Deepika
Sprint-3	IBMv/Watson IoTplatform & Node-RED service	US-1	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	High	Ahmayn Abraam Gowri Deepika

Sprint	Functional Requirement(Epic)	User Story Number	User Story/Task	Story Points	Priority	Team Member
Sprint-1	Node-REDservice	US-2	Create a Node-RED service.	10	High	Ahmayn Abraam Gowri Deepika

			create the IBM Watson IoT platform.			Gowri Deepika
Sprint-2	IBM Watson IoTplatform	US-4	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watso IoTplatform and get the device credentials.	5	High	Ahmayn Ahamm Gowri Deepika
Sprint-3	IBM-Watson IoTplatform & Node-RED service	US-1	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	10	High	Ahmayn Ahamm Gowri Deepika

Sprint	Functional Requirement(Epic)	User Story Number	User Story Task	Story Point	Priority	TeamMember
Sprint-1	Node-REDservice	US-2	Create a Node-RED service.	10	High	Ahmayn Ahamm Gowri Deepika
Sprint-3	IBMIoTplatform	US-1	Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform	7	High	Ahmayn Ahamm Gowri Deepika
Sprint-3	IBMIoTplatform	US-2	After developing python code,commands are received.Just print the statements which represent the control of the devices.	5	Medium	Ahmayn Ahamm Gowri Deepika
Sprint-4	IBMCloudService	US-3	Publish Data toThe IBM Cloud.	8	High	Ahmayn Ahamm Deepika Gowri
Sprint-4	Webpage	US-1	Create Web UI in Node- Red	10	High	Ahmayn Ahamm Deepika Gowri
Sprint-4	IBMIoTplatform	US-2	Configure the Node-RED flow to receive data from the IBM IoT platform and also to Cloudant DB nodes to store the received sensor data in the cloudant DB	10	High	Ahmayn Ahamm Deepika Gowri

PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND ESTIMATION:

Project Tracker,Velocity&Burn down Chart:(4Marks)

Sprint	Total StoryPoint s	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Point: Completed (at en Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6Days	24Oct2022	29Oct2022	20	29Oct2022
Sprint-2	20	6Days	31Oct2022	05Nov2022	20	05Nov2022
Sprint-3	20	6Days	07Nov2022	12Nov2022	20	12Nov2022
Sprint-4	20	6Days	14Nov2022	19Nov2022	20	19Nov2022

Velocity:

Imagine we have a10-day sprint duration,and the velocity ofthesteam20(points per sprint).Let's calculate the team's average velocity(AV)per iteration unit(story points per day)

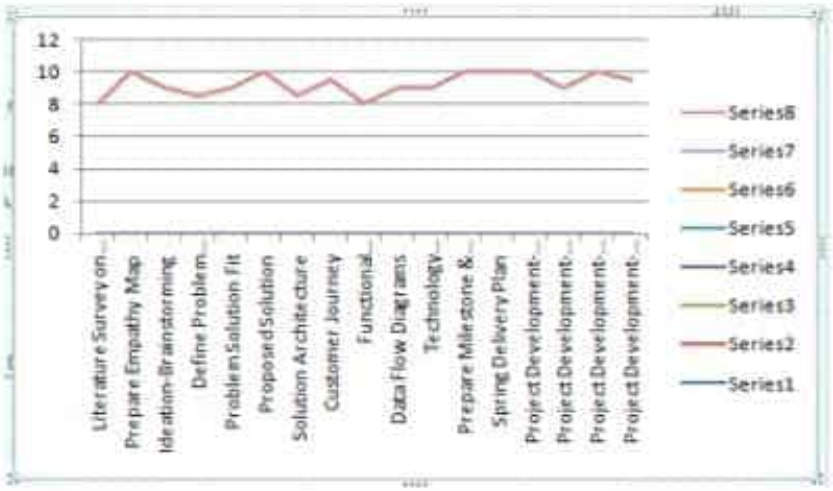
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

BurndownChart:

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

BurndownChart:

A burndown chart is a graphical representation of work left to do versus time.However, burndown charts can be applied to any project containing measurableprogress overtime



CODING AND SOLUTIONING

FEATURE-1

```
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()
```

```
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..... ")
```

```
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:
```

```
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)
```

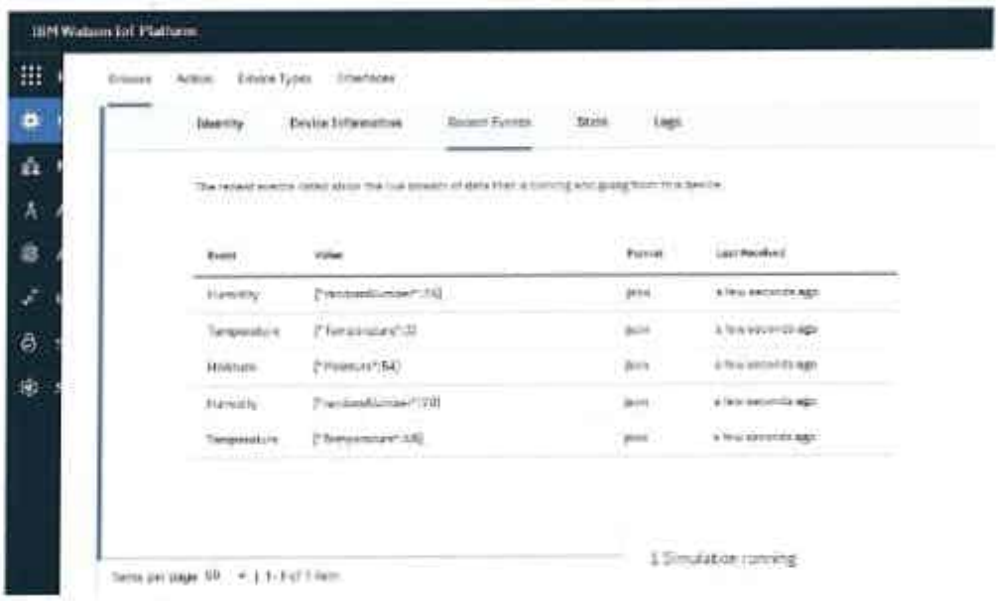
```
temp_data, qos=0)
sleep(1)
if success:
print (" ..........................publish
ok.............................. ")
print ("Published Temperature = %s C" % temp_sensor, "to IBM
Watson")
success = deviceCli.publishEvent("PH sensor", "json", PH_data,
qos=0)
sleep(1)
if success:
print ("Published 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):
```

```
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()
```

```
deviceCli.disconnect()
```



Features

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 a 3.3V regulator), but 5V is ideal in case the regulator has different specs.

BUZZER

Specifications

? RatedVoltage : 6V DC
? Operating Voltage : 4 to 8V DC
? Rated Current*: ?30mA
? SoundOutput 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 vehiclessuch as ambulances, police cars and fire trucks. There are two general types, pneumatic and electronic.

FEATURE-2:

i. Goodsensitivity to Combustible gas in wide range .
ii. Highsensitivity to LPG, Propane and Hydrogen .
iii. Longlife and low cost.
iv. Simpledrive circuit.

TESTING

TEST CASES:

sno	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy- 95% Validation accuracy- 72%	
3	Confidence score	Class detected- 80% Confidence score-80%	

User Acceptance Testing:

		72%	
3	Confidence score	Class detected- 80% Confidence score-80%	

User Acceptance Testing:

RESULTS

The problem of crop vandalization by wild animals and fire has become a major social problem in current time. It requires urgent attention as no effective solution exists till date for this problem. Thus this project carries a great social 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 and will save them from the unproductive efforts that they endure for the protection their fields. This will also help them in achievingbetter crop yields thus leading to theireconomic wellbeing.

ADVANTAGES AND DISADVANTAGES

Advantage:

Controllable food supply. you might have droughts or floods, but if you 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.

Disadvantage:

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

CONCLUSION:

A IoT Web Application is built for smart agricultural system

using Watson IoT
platform, Watsonsimulator, IBM cloud and Node-RED
FUTURE SCOPE
In the future, there will be very large scope, this project can be made
based on Image processing in which wild animaland fire can be detected
by cameras and if it comes towards farmthen 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 sensing
this laser or sensor's security system will beactivated.

APPENDIX

SOURCE CODE

```
import time importsys import ibmiotf.application # to installpip
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
```

```
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 of type "greeting"
10 times
deviceCli.connect()
```

```
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.device
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))
```

```
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:
temp=random.randint(0,1
00)
pulse=random.randint(0,100)
soil=random.randint(0,100)
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":"625574ead9839b34
```

```
"authentication":"apiKey",
"apiKey":"ef745d48e395ccc0",
"outputType":"cmd",
"deviceId":"b827ebd607b5",
"deviceType":"weather_monitor",
"eventCommandType":"data",
"format":"json",
"data":"data",
"qos":0,
"name":"IBM IoT",
"service":"registere
d", "x":680,
"y":220,
"wires":[]
},
{
"id":"4cff18c3274cccc4", "type":"ui_button",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c",
"order":2,
"width":"0",
"height":"0",
"passthru":false,
"label":"MotorON",
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
```

```
"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":"MotorOF
F",
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoroff\"}",
"payloadType":"str",
"topic":"motoroff",
"topicType":"s
tr", "x":350,
```

```
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoroff\"}",
"payloadType":"str",
"topic":"motoroff",
"topicType":"s
tr", "x":350,
"y":220, "wires":[["625574ead9839b34"]]},
{"id":"ef745d48e395ccc0", "type":"ibmiot",
"name":"weather_monitor", "keepalive":"60",
"serverName":"",
"cleansession":true,
"appId":"",
"shared":false},
{"id":"716e956.00eed6c",
"type":"ui_group",
"name":"Form",
"tab":"7e62365e.b7e6b8
", "order":1,
"disp":true,
"width":"6",
"collapse":fal
se},
{"id":"7e62365e.b7e6b8",
"type":"ui_tab",
"name":"contorl",
```

```
{"id":"7e62365e.b7e6b8",
"type":"ui_tab",
"name":"contorl",
"icon":"dashboard
", "order":1,
"disabled":false,
"hidden":false}
]
[
{
"id":"b42b5519fee73ee2", "type":"ibmiotin",
"z":"03acb6ae05a0c712",
"authentication":"apiKey",
"apiKey":"ef745d48e395ccc0",
"inputType":"evt",
"logicalInterface":"",
"ruleId":"",
"deviceId":"b827ebd607b5",
"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","a94979702815 8f3
f","a71f164bc3 78bcf1"]]
},
{
"id":"50b13e02170d73fc
",
"type":"function",
"z":"03acb6ae05a0c712
", "name":"Soil
Moisture",
"func":"msg.payload = msg.payload.soil;
\nglobal.set('s',msg.payload);\nreturn msg;",
"outputs":1,
"noerr":
0,
"initialize
":"",
"finalize":"",
"libs":[],
"x":490,
"y":120,
"wires":[["a949797028158f3f","ba98e701f55f04fe"]]
```

```
"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,
"initialize
":"",
"finalize":"",
"li
bs
":[
],
"x
":
48
0,
"y":260, "wires":[["a949797028158f3f","70a5b076eeb80b70"]]
},
{
"id":"a949797028158f3f
",
"type":"debug",
"z":"03acb6ae05a0c712
```

```
",
"type":"debug",
"z":"03acb6ae05a0c712
", "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":"f4cb8513b95c98a4",
"order":6,
"width":"0",
"height":"0",
"gtype":"gage",
"title":"Humidity",
"label":"Percentage(%)",
"format":"{{value}}
```

```
"format":"{{value}}
", "min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",
"seg2":"",
"className
":"","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,
"initialize
":"",
"finalize":"",
"li
bs
":[
],
"x
":
49
```

```
":
49
0,
"y":360,
"wires":[["8e8b63b110c5ec2d","a949797028158f3f"]]
},
{
"id":"8e8b63b110c5ec2d",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a4",
"order":11,
"width":"0",
"height":"0",
"gtype":"gage",
"title":"Temperature",
"label":"DegreeCelcius",
"format":"{{value}}",
"min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"], "seg1":"",
"seg2":"",
"className
":"",
"x":790,
"y":360,
"wires":[]
},
```

```
"className
":"",
"x":790,
"y":360,
"wires":[]
},
{
"id":"ba98e701f55f04fe",
"type":"ui_gauge",
"z":"03acb6ae05a0c712",
"name":"",
"group":"f4cb8513b95c98a4",
"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":"",
"className
":"",
"x":790,
"y":120,
"wires":[]
},
```

```
"y":120,
"wires":[]
},
{
"id":"a259673baf5f0f98
", "type":"httpin",
"z":"03acb6ae05a0c712
", "name":"",
"url":"/sensor",
"method":"ge
t",
"upload":fals
e,
"swaggerDoc"
:"", "x":370,
"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":"",
```

```
"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":"5c7996d53a445412
",
"type":"httpresponse",
"z":"03acb6ae05a0c712
", "name":"",
"statusCode":"",
"header
s":{},
"x":870,
"y":500,
```

```
"statusCode":"",
"header
s":{},
"x":870,
"y":500,
"wires":[]
},
{
"id":"ef745d48e395ccc0",
"type":"ibmiot",
"name":"weather_monitor",
"keepalive":"60",
"serverName":"",
"cleansession":true,
"appId":"",
"shared":false},
{
"id":"f4cb8513b95c98a4", "type":"ui_group",
"name":"monitor",
"tab":"1f4cb829.2fdee8
", "order":2,
"disp":
true,
"width
":"6",
"collapse":f
alse,
"className
":""
```

```
"id":"f4cb8513b95c98a4", "type":"ui_group",
"name":"monitor",
"tab":"1f4cb829.2fdee8
", "order":2,
"disp":
true,
"width
":"6",
"collapse":f
alse,
"className
":""
},
{
"id":"1f4cb829.2fdee8",
"type":"ui_tab",
"name":"Home",
"icon":"dashboard
", "order":3,
"disabled":false,
"hidden":false }
```


GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-34681-1660271017