IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID : PNT2022TMID15968

Team Members :

Santhiya C

Shanmuga Priya S

Preethi Shre A

Gayathri S

INTRODUCTION

PROJECT OVREVIEW:

The purpose of this project is to secure or protect the farm from the theft in the farm or main purpose of this project is to alert the farmer as well as fear the animals with getting harm to animals. Crops in the farms are many times devastated by the wild as well as domestic animals and low productivity of crops is one of the reasons for this. It is not possible to stay 24 hours in the farm to sentinel the crops. So to surmount this issue an automated perspicacious crop aegis system is proposed utilizing Internet of Things (IOT). The system consists of esp8266 (node MCU), soil moisture sensor, dihydrogen monoxide sensor, GPRS and GSM module, servo motor, dihydrogen monoxide pump, etc. to obtain the required output. As soon as any kineticist is detected the system will engender an alarm to be taken and the lights will glow up implemented at every corner of the farm. This will not harm any animal and the crops will stay forfended.

PURPOSE:

Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal and fire and to maintain the farm by providing the correct water level . 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 and manual surveillance and various such exhaustive and dangerous method.

REFERENCES:

i. **AUTHOR: Dr.M. Chandra and Mohan Reddy**

ii. **AUTHOR: P.Rekha and T.Saranya**.

iii. **AUTHOR: ANJANA and RAVICHANDRAN**

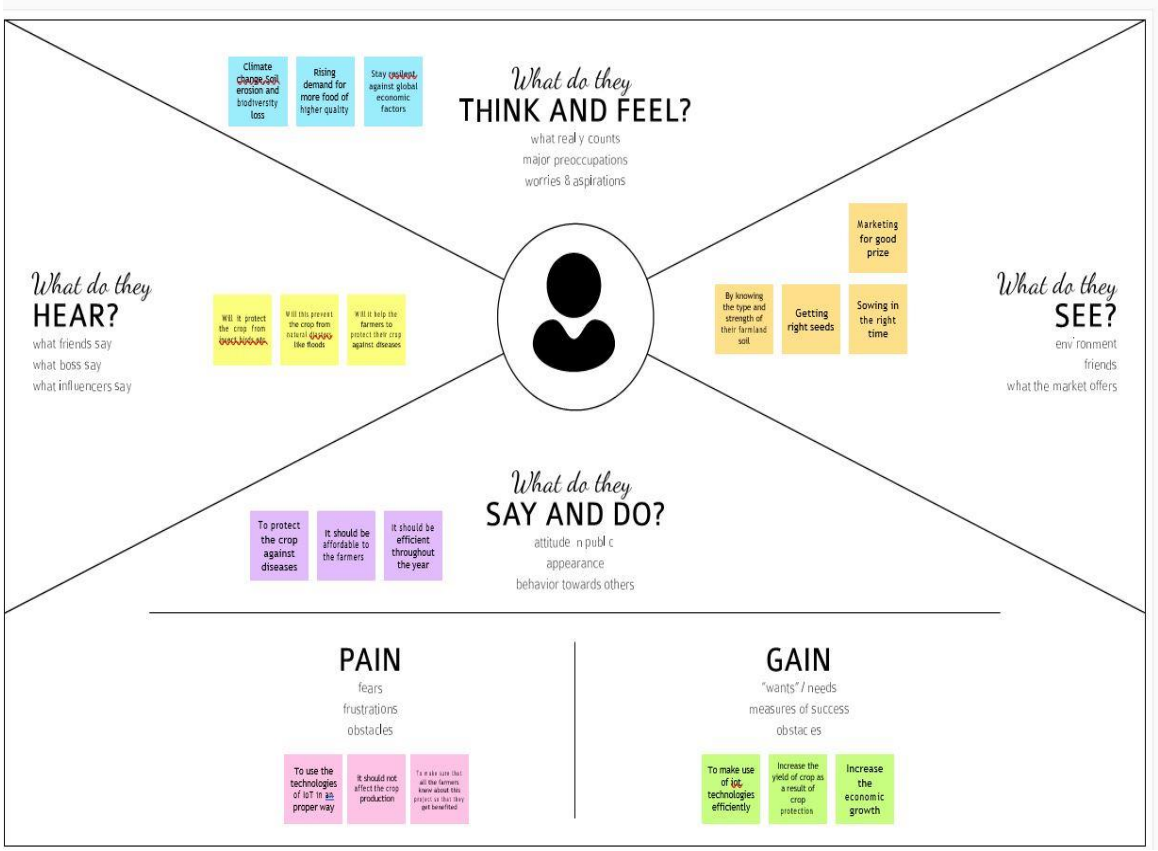
PROBLEM STATEMENT DEFINITION STATEMENT:

In the world economy of many Country dependent upon the agriculture.

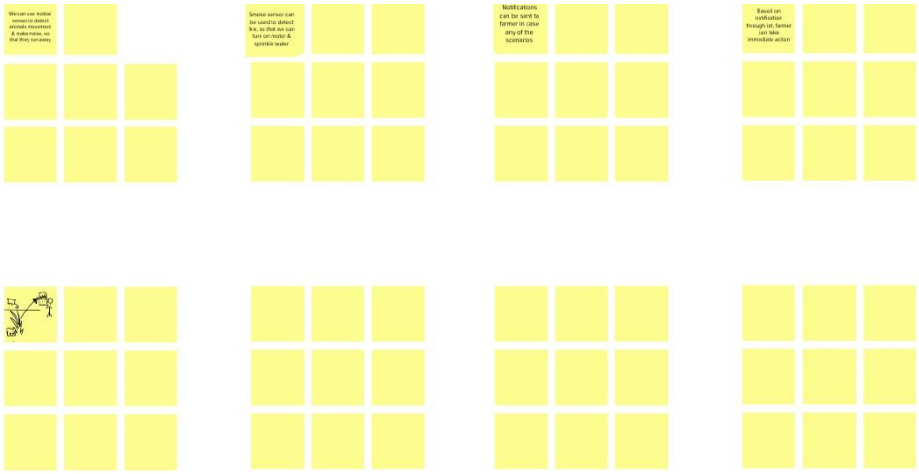
In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds and fire etc. this leads to huge loss for the farmers and maintaining directly is also difficult when farmer is not near to feild .it 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 materials for industries. 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:



motion sensor can be used to detect wild animals approaching near the field and smoke sensor can be used to detect the fire.

In such a case the sensor can signal the microcontroller to take action. The microcontroller now sounds an alarm to woo the animals away from the field as well as sends SMS to the farmer and makes call, so that farmer may know about the issue and come to the spot in case the animals don't turn away by the alarm

If there is a smoke, it can immediately turns ON the motor. This can ensure complete safety of crops from animals and from fire thus protecting the farmer's loss.

PROPOSED SOLUTION:

a.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Crops are not irrigated properly due to insufficient labour forces. • Improper maintenance of crops against various environmental factors such as temperature climate, topography and soil quality which results in crop destruction. • Lack of knowledge among farmers in usage of fertilizers and hence crops are affected due to high ammonia, urea, potassium and high PH level fertilizers. • Requires protecting crops from Wild animals attacks, birds and pests.
2.	Idea / Solution description	•Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT. • Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers. • IOT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers. • Image processing techniques with IOT is followed for crop protection against animal attacks.
3.	Novelty / Uniqueness	• Automatic crop maintenance and protection using embedded and IOT technology.
4.	Social Impact / Customer Satisfaction	• This proposed system provides many facilities which helps the farmers to maintain the crop field without much loss.
5.	Business Model (Revenue Model)	• This prototype can be developed as product with minimum cost with high performance .
6.	Scalability of the Solution	• This can be developed to a scalable product by using sensors and transmitting the data through Wireless Sensor Network and Analysing the data in cloud and operation is performed using robots

PROBLEM SOLUTIONFIT:

Project Title: IOT BASED SMART CROP PROTECTION SYSTEMFOR AGRICULTURE

Project Design Phase-I - Solution Fit

Team ID: PNT2022TMID15968

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS

- Farmers who trying to protect crops from various problems

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL

- Limited supervision.
- Limited financial constrains.
- Lack of man power.

5. AVAILABLE SOLUTIONS PLUSES & MINUSES AS

- Automation in irrigation.
- CCTV camera to monitor and supervise the crops.
- Alarm system to give alert while animals attacks the crops.

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY PR

- Crops are not irrigated properly.
- Improper maintenance of crops.
- Lack of knowledge among farmers in usage of fertilizers and hence crops are affected.
- Requires protecting crops from Wild animals attacks, birds and pests.

9. PROBLEM ROOT / CAUSE RC

- Due to insufficient labour forces.
- Due to various environmental factors such as temperature climate, topography and soil quality which results in crop destruction.
- Due to high ammonia, urea, potassium and high PH level fertilizers.
- Crops are damaged and it affects growth.

7. BEHAVIOR + ITS INTENSITY BE

- Asks suggestions from surrounding peoples and implement the recent technologies.
- Consumes more time in crop land.
- Searching for an alternative solution for an existing solution.

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR

- By seeing surrounding crop land with installing machineries.
- Hearing about innovative technologies and effective solutions.

4. EMOTIONS BEFORE / AFTER EM

- Mental frustrations due to insufficient production of crops.
- Felt smart enough to follow the available technologies with minimum cost.

10. YOUR SOLUTION SL

- Moisture sensor is interfaced with Arduino Microcontroller to measure the moisture level in soil and relay is used to turn ON and OFF the motor pump for managing the excess water level. It will be updated to authorities through IOT.
- Temperature sensor connected to microcontroller is used to monitor the temperature in the field. The optimum temperature required for crop cultivation is maintained using sprinklers.
- IOT based fertilizing methods are followed, to minimize the negative effects on growth of crops while using fertilizers .
- Image processing techniques with IOT is followed for crop protection against animal attacks.

8. CHANNELS of BEHAVIOR CH

ONLINE

- Using different platforms /social media to describe the working and uses of smart crop protection device.

OFFLINE

- Giving awareness among farmers about the application of the device.

Extract online & offline CH of BE

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT:

Following are the functional requirements of the proposed solution.

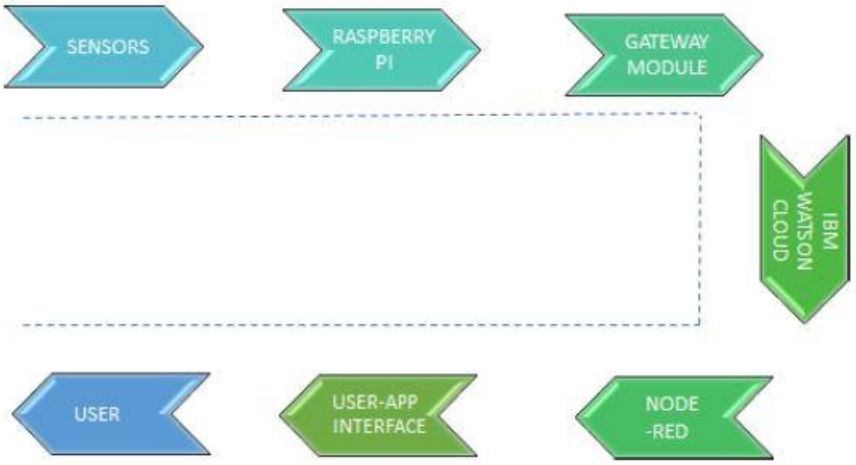
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Interfacing with hardware	Interface the sensors with the software application so as to alert the farmers in case of any harm for crops
FR-4	Database Connection	Databases are retrieved from IBM Cloud ant
FR-5	Mobile Application	Alarm and motors can be accessed from the mobile app

NON FUNCTINAL REQUIREMENT:

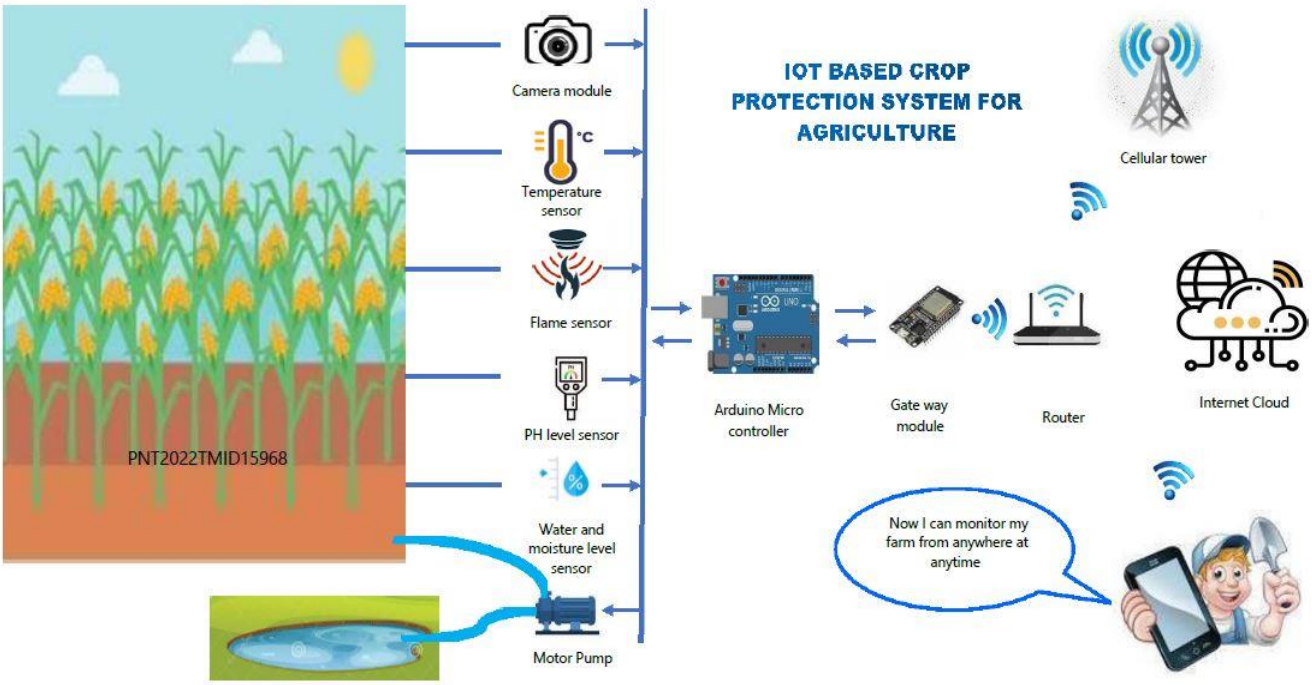
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The smart crop protection alerts the farmers in case of any obstacles and helps in protecting the crops
NFR-2	**Security**	Smart Agriculture can improve the farming practices and maintain sustainable production of crops especially by preventing the animals into the agricultural lands through IoT enabled devices.
NFR-3	**Reliability**	With a proper power supply, SD card and programming the processor should be able to run 24/7 for years. The SD card and power supply will likely wear out faster than the Pi. The possible reasons behind Raspberry Pi failure can be power breakdowns, SD card failures, and ineligible environments.
NFR-4	**Performance**	Usage of an SD card module that helps to store a specified sound to scare the animals. Crop damage due to animal attack can be sensed. Network and Design Evaluation
NFR-5	**Availability**	Agriculture for different variety of crops is based on the monsoon changes, indoor and outdoor climatic temperatures, availability of rainfall and irrigation methods.
NFR-6	**Scalability**	The product shall be made available to everyone especially in remote areas for better efficiency of

PROJECT DESIGN

DATA FLOW DIAGRAM:



SOLUTION AND TECHNICAL ARCHITECTURE:



a.

USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Farmer)	Maintaining Fields	USN-1	As a user, I can monitor the growth of crops and protect the crops against animals	I can maintain the fields with less labor	High	Sprint-1
	Analyzing Problems	USN-2	As a user, I collect the required information about the problems on agriculture fields	I can ask my field owner directly.	Low	Sprint-2
		USN-3	As a user, I can monitor the moisture level in soil and solve the problems by using Smart IOT System	I can take remedial action immediately	High	Sprint-1
Project Designers	Identifying the problem and provide solutions	USN-4	As a user, I can sense the water level and flame in the field using sensor and monitor using IOT	I can perform this actions via IoT.	Medium	Sprint-1
		USN-5	As a user, I can make services for Irrigation, pesticides, Fertilization, and Soil preparation	I can solve this problem using IOT	High	Sprint-1
			As a user, I can monitor the field against animal attacks using a camera interface module and appropriate actions can be taken	I can monitor the field continuously.	Medium	Sprint-2
Customer (Field Maintainer)	Problem solutions	USN-6	As a user, areas can be monitored from a remote place	Checking Process	Medium	Sprint-3
	Application	USN-7	As a user, I can respond to the problems in the fields immediately	Continuous monitoring and remedial actions.	Medium	Sprint-3
	Final Process	USN-8	This proposed smart IOT-based crop protection device is found to be cost-effective and efficient	I can take necessary action if required.	Medium	Sprint-4

PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND ESTIMATION:

Sprint	Functional Requirement (Epic)	Task	Story Points	Priority	Team Members
Sprint-1	Registration	As a team lead , I can enrolled for the project byentering my email, password and within that I can enter my team members name and their email.	2	High	Santhiya C
Sprint-1		As a team lead , I will receive confirmation email once , I have enrolled for the project with team id and along with team members name.	2	High	Santhiya C
Sprint-2	Login	As a team member, I can login to the IBM portal by entering email & password	1	Medium	Gayathri S
Sprint-2		As a team member, I can login to the IBM portal by entering email & password	1	Medium	Preethi shre A
Sprint-2		As a team member, I can login to the IBM portal by entering email &	1	Medium	Shanmuga Priya S
Sprint-2		As a team member, I can login to the IBM portal by entering email & password	1	Medium	Preethi shre A

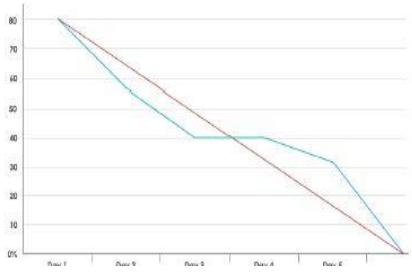
Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date(Planned)	Story Points Completed (ason Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	22 Oct 2022	27 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	30	30 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	49	06 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	50	07 Nov 2022

Velocity:

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

Burndown Chart:



CODING AND SOLUTIONING

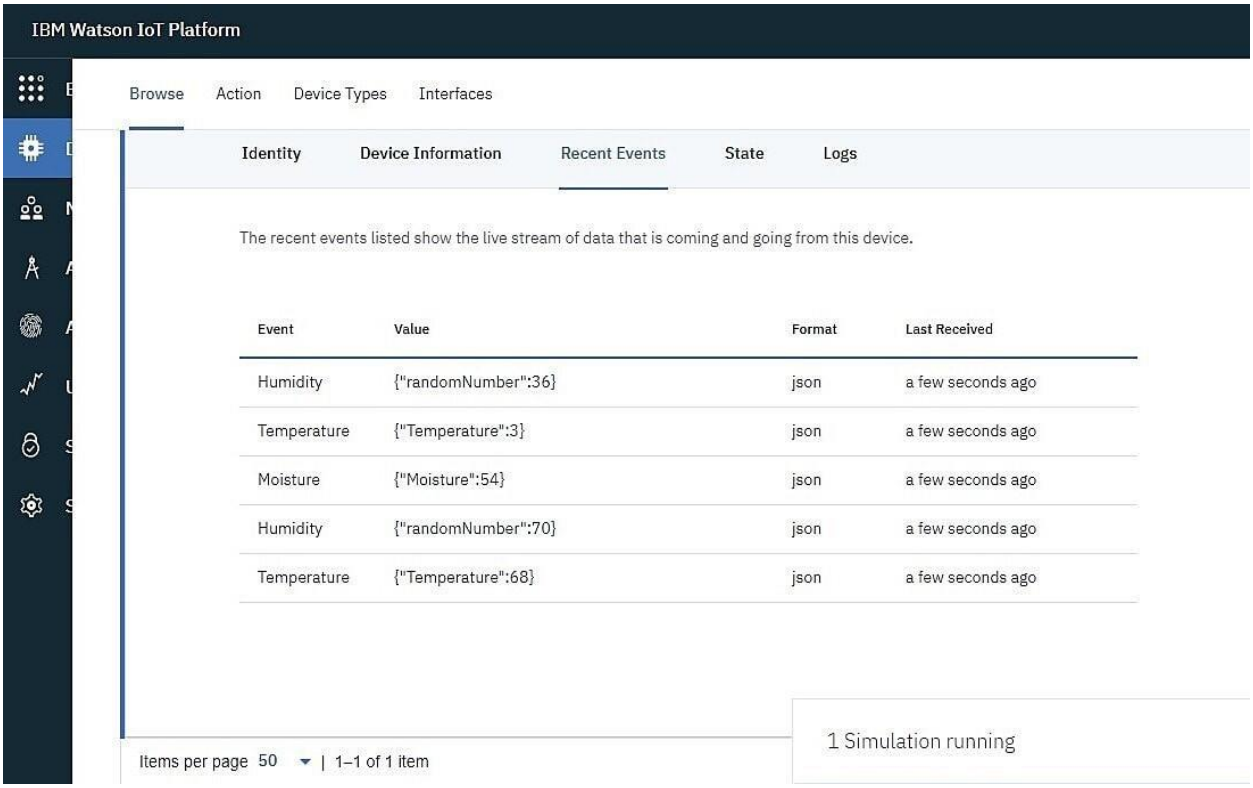
FEATURE-1

```
import random
import ibmiotf.application
import ibmiotf.device
from time import sleep
import sys
#IBM Watson Device Credentials.
organization = "24p6oo"
deviceType = "PNT2022TMID15968"
deviceId = "36982"
authMethod = "use-token-auth"
authToken = "kaviya@123"
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()
```



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 ±300HzTone: 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:

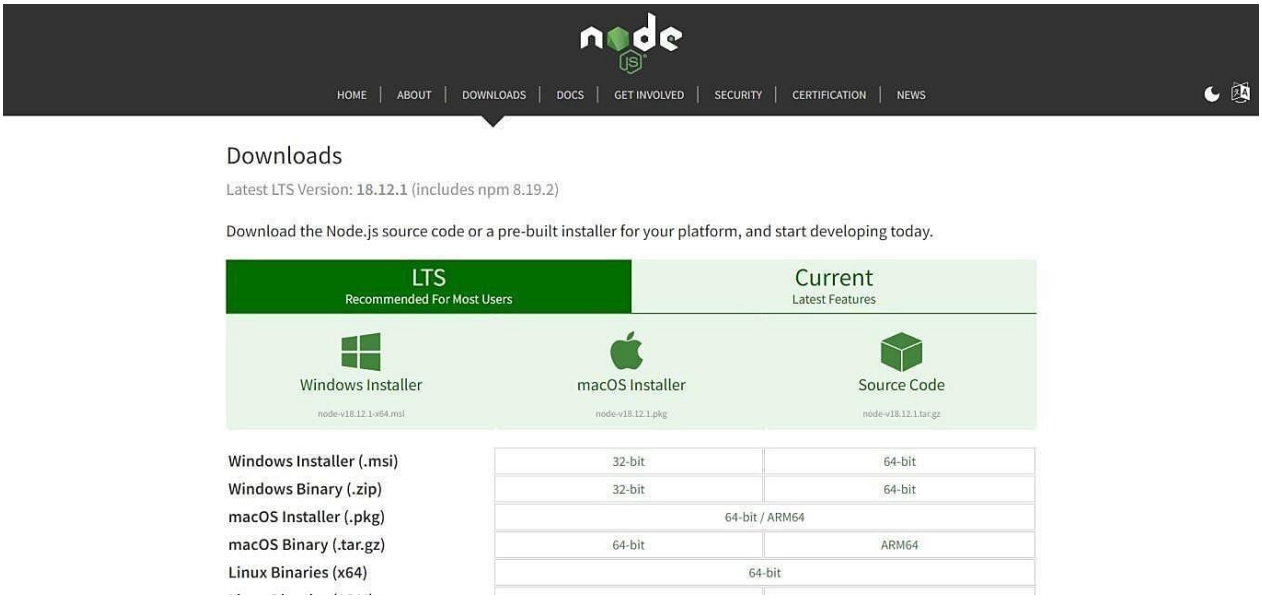
i. Goodsensitivity to Combustible gas in wide range .
ii. Highsensitivity to LPG, Propane and Hydrogen .
iii. Longlife and low cost.
iv. Simpledrive circuit.

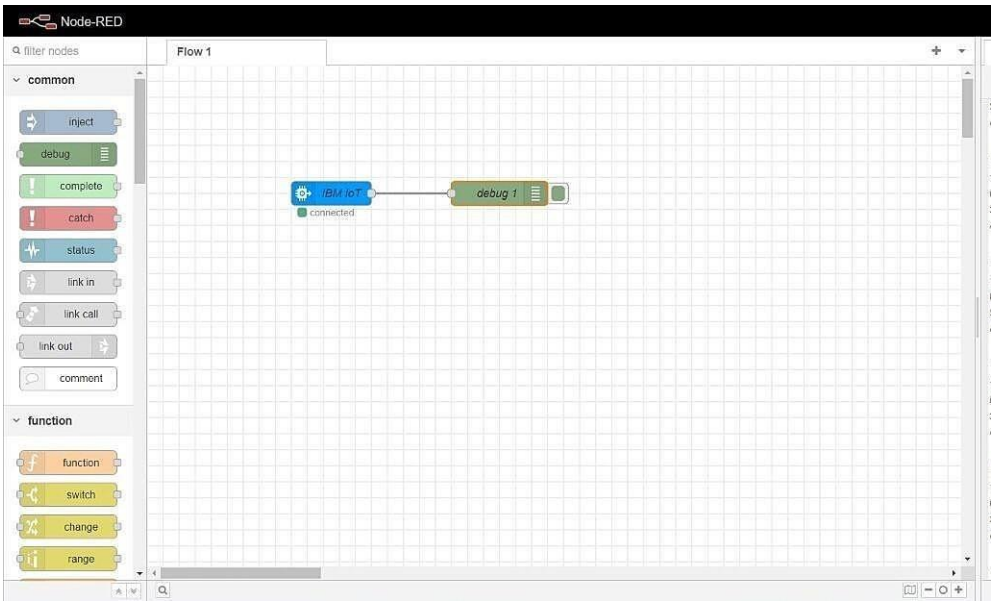
TESTING

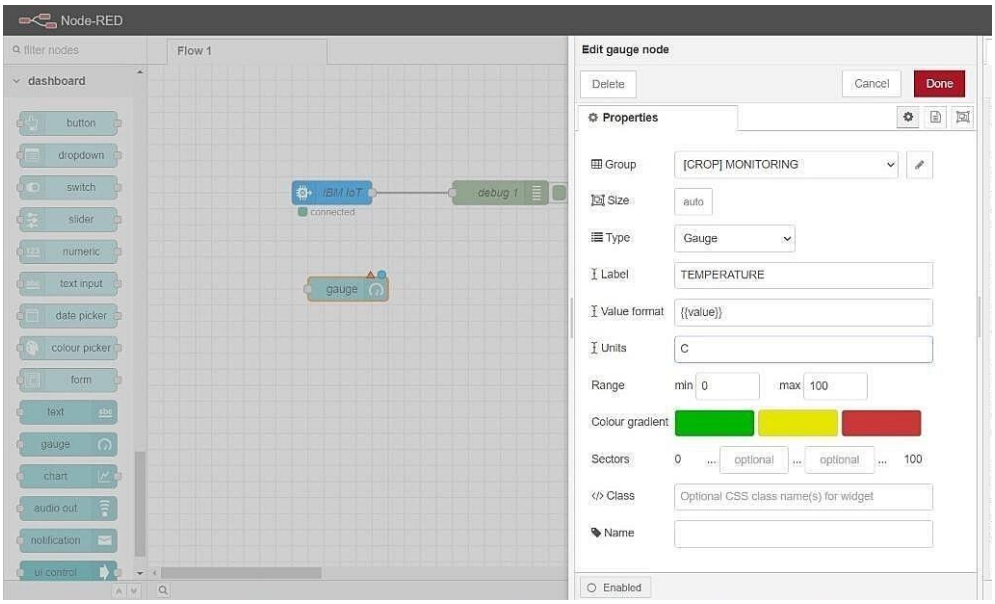
TEST CASES:

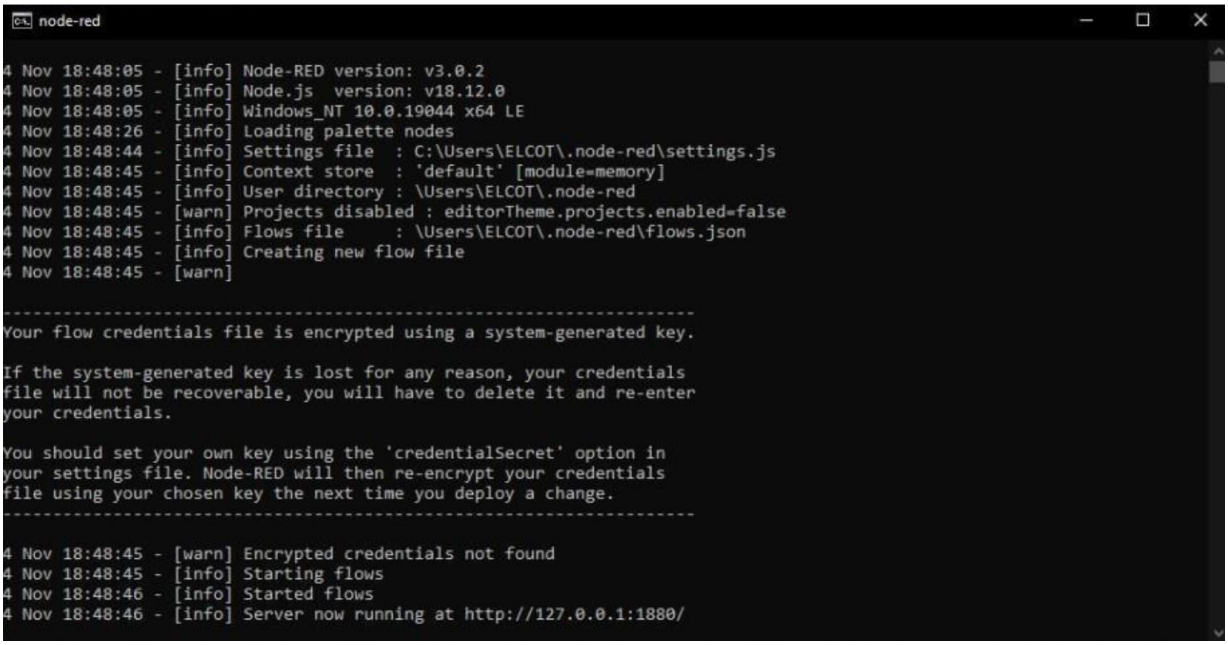
sno	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy-95% Validation accuracy-72%	
3	Confidence score	Class detected-80% Confidence score-80%	

User Acceptance Testing:






Node-RED
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


node-red
4 Nov 18:48:05 - [info] Node-RED version: v3.0.2
4 Nov 18:48:05 - [info] Node.js version: v18.12.0
4 Nov 18:48:05 - [info] Windows_NT 10.0.19044 x64 LE
4 Nov 18:48:26 - [info] Loading palette nodes
4 Nov 18:48:44 - [info] Settings file : C:\Users\ELCOT\.node-red\settings.js
4 Nov 18:48:45 - [info] Context store : 'default' [module=memory]
4 Nov 18:48:45 - [info] User directory : \Users\ELCOT\.node-red
4 Nov 18:48:45 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Nov 18:48:45 - [info] Flows file : \Users\ELCOT\.node-red\flows.json
4 Nov 18:48:45 - [info] Creating new flow file
4 Nov 18:48:45 - [warn]
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
4 Nov 18:48:45 - [warn] Encrypted credentials not found
4 Nov 18:48:45 - [info] Starting flows
4 Nov 18:48:46 - [info] Started flows
4 Nov 18:48:46 - [info] Server now running at http://127.0.0.1:1880/

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.

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 starving. 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 IoTplatform, Watson simulator, 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 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.

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 =
"PNT2022TMID15968" #replace the Device type deviceId = "36982"
# replace Device ID authMethod = "token"authToken =
"Kaviya@123" # 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":"0",

"passthru":false,
"label":"MotorO
N",
"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":"ibmiot",
"name":"weather_monitor","keepalive":"60",
"serverName":"",
"cleansession":true,
"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":"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","a949797028158f3f","a71f164bc3 78bcf1"]]
},
{
"id":"50b13e02170d73f
c",
"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":"03acb6ae05a0c71
2","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}}
","min":0,
"max":"100",
"colors":["#00b500","#e6e600","#ca3838"],
"seg1":"", "seg2":"",
"classNam
e
":"","x":86
0,
"y":260,
"wires":[]
},
{
"id":"a71f164bc378bcf1","type":"functi
on","z":"03acb6ae05a0c712",
"name":"Temperature",
  "func":"msg.payload=msg.payload.temp;\nglobal.set('t',msg.payload);\nreturn msg;","outputs":1,
"noerr":
0,
"initializ
e":"",
"finalize":"",
"l
i
bs
":
[
],
```

"
x
":
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":"",
"classNam
e":"",
"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":"",

"classNam
e":"",
"x":790,

"y":120,

"wires":[]

},

{

"id":"a259673baf5f0f9
8","type":"httpin",
"z":"03acb6ae05a0c71
2","name":"",
"url":"/sensor",

"method":"g
et",
"upload":fals
e,
"swaggerDo
c"
:"","x":37

0,"y":500,
"wires":[["18a8cdbf7943d27a"]]
},

{
"id":"18a8cdbf7943d27a","type":"functi
on","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":"httpresponse",
"z":"03acb6ae05a0c71
2","name":"",
"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_gro
up","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 }

GitHub & Project Demo Link

https://github.com/IBM-EPBL/IBM-Project-12190-1659440124

PROJECT DEMO LINK

https://drive.google.com/file/d/1DP_BajQr9s6ZTV8ch00SJ0pmS_NHAsFM/view?usp=drivesdk