IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID : PNT2022TMID02250

TEAM MEMBERS:
SAGAR Y
RITHESH S
SAMYUKTHA RAJKUMARAN
NAVEEN KUMAR B

TABLE OF CONTENTS

1.INTRODUCTION

1.1 PROJECT OVERVIEW

Agriculture is the science and art of cultivating plants. Agriculture plays a major role in the economic development of our country and this has been a primary occupation for many years. In order to increase the productivity of the crops and to minimize the expenses of agricultural practices we adopt smart agriculture techniques using IOT. It includes various features like solar power generation and advancement in modern agriculture crop protection using greenhouse/polyhouse. It makes use of GSM technology for intimating the cultivators about various environmental factors continuously via SMS.

1.2. PURPOSE

The main motivation or purpose of this project is to provide an efficient solution to pests and other animal intruders who ruin crops. This enables farmers by minimizing losses which are the repercussions of being pervious to intruders. This has an alert system which also takes actions to fend off the intruder if possible in the circumstance.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The existing systems provide the real-time alert feature. They however fail to take action based on the kind of intruder that enters the field. This turns out to be futile as by the time the farmer arrives at the scene to take action most of the damage is already done and the loss is incurred.This only serves as an elementary improvement to building high or electric fences.

2.2 REFERENCES:

1)Smart Crop protection system from living objects and fire using Adriano

2)Review on IOT in Agricultural Crop Protection and Power Generation

3)Development of IOT based Smart Security and Monitoring Devices for Agriculture

4)Shweta B. Saraf, Dhanashri H. Gawali

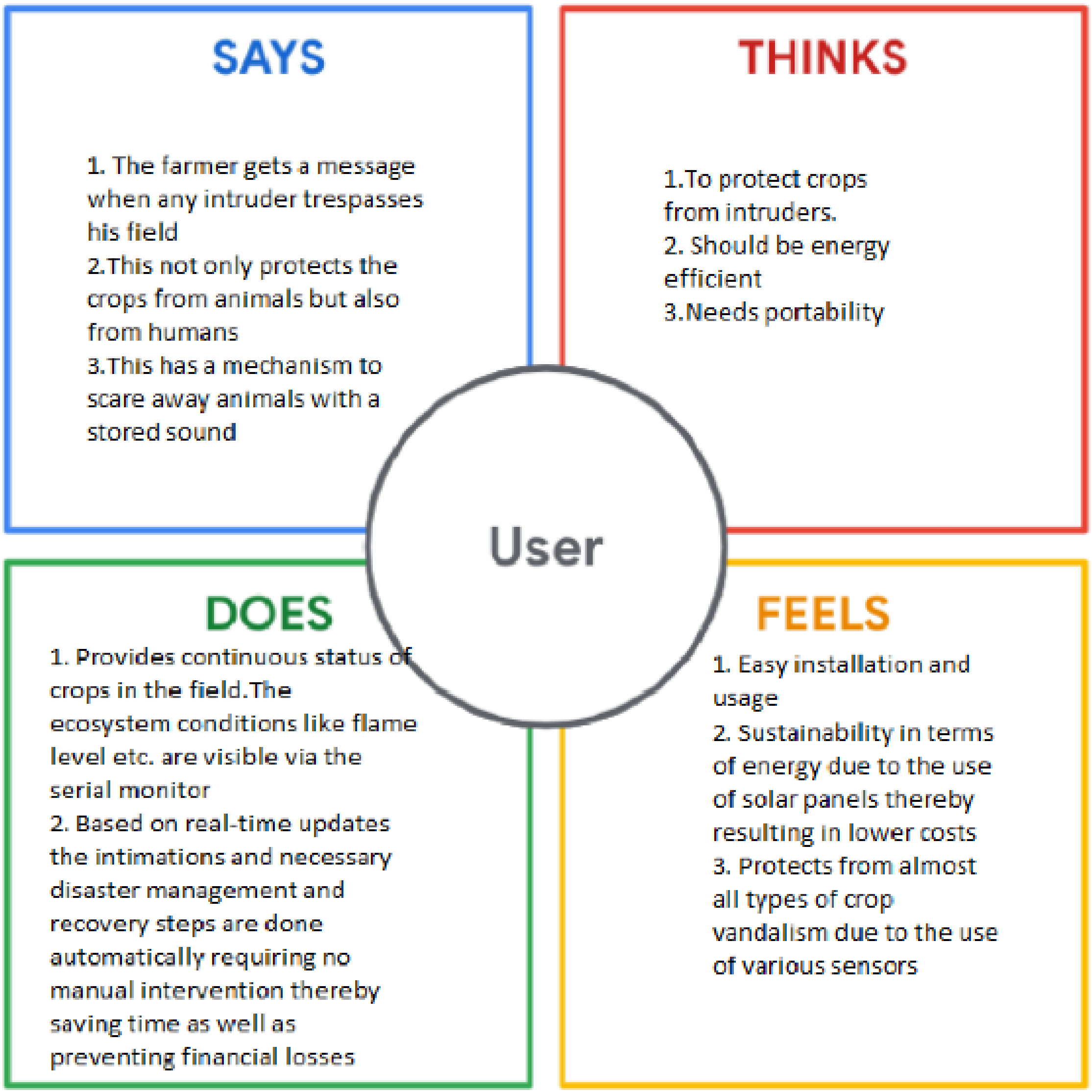
5)G. Sushanth, and S. Sujatha

2.3 PROBLEM STATEMENT DEFINITION

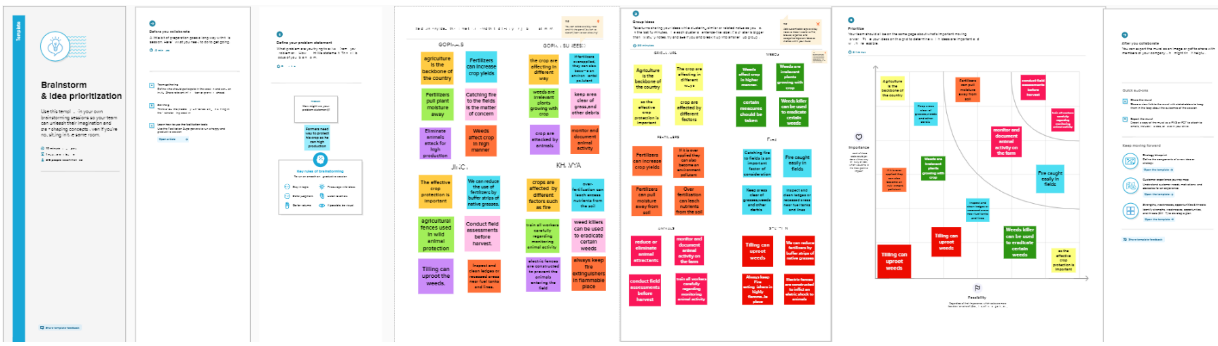
More than 95% farmers in India uses the traditional farming technique. In order to earn more profit from agriculture, modern farming technologies must be adopted such as greenhouse farming, hydroponic farming, tissue culture, vertical farming etc. Greenhouse farming is a technique of adopting modern farming technologies to provide favourable environmental conditions to the plants. This technique is used to protect the plants from adverse aerial conditions like wind, cold, excessive radiation, extreme temperature, diseases,insects and even animals.

3.IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

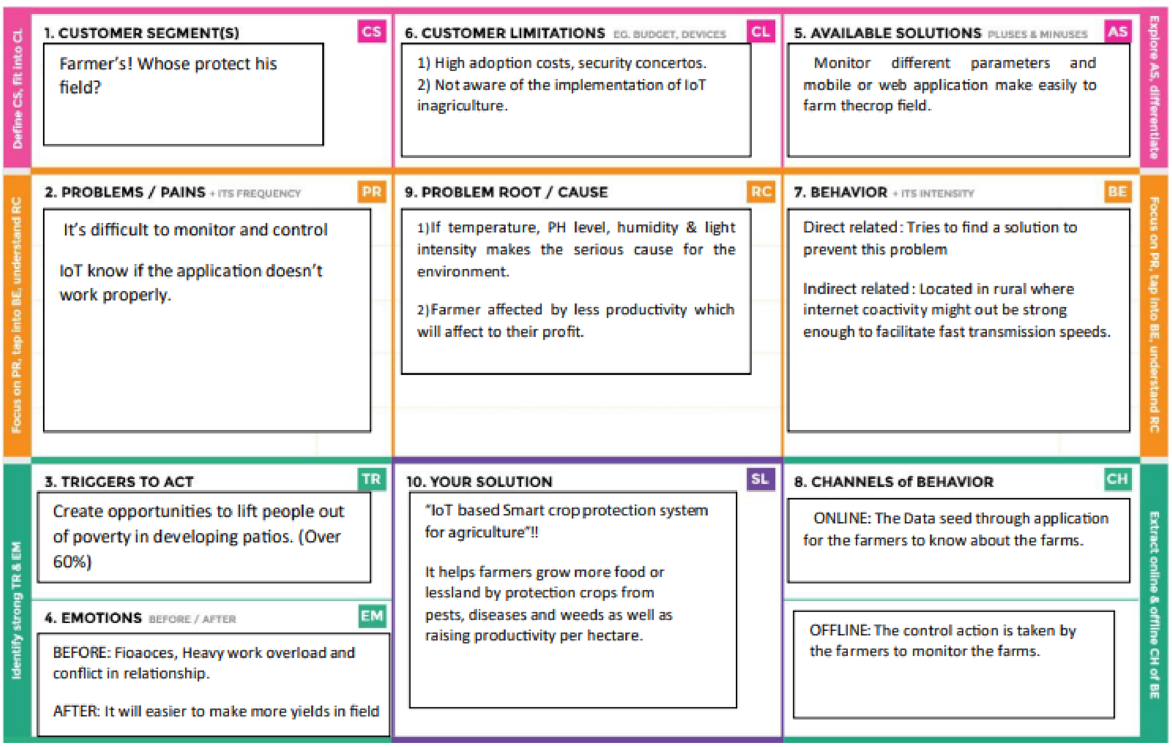


3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement. (Problem to be solved)	Crops are not irrigated properly due to insufficient labor forces. Improper maintenance of crops against various environmental factors such as temperature climate, topography and soil quantity which results in crop destruction. Requires protecting crops from wild animal 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 & 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. Image processing techniques with IOT is followed for crop protection against animal attack.
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 a product with minimum cost with high performance.
6.	Scalability of the solution	This can be developed to a scalable product by using solution sensors and transmitting the data through Wireless Sensor Network and analyzing the data in cloud and operation is performed using robots

3.4 PROBLEM SOLUTION FIT



4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sensing animals approaching the crop field, the device sends the farmer an SMS and plays an alarm to scare them away.
FR-2	User Reception	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, fertilizer application, strip cropping, and scheduled planting operations.

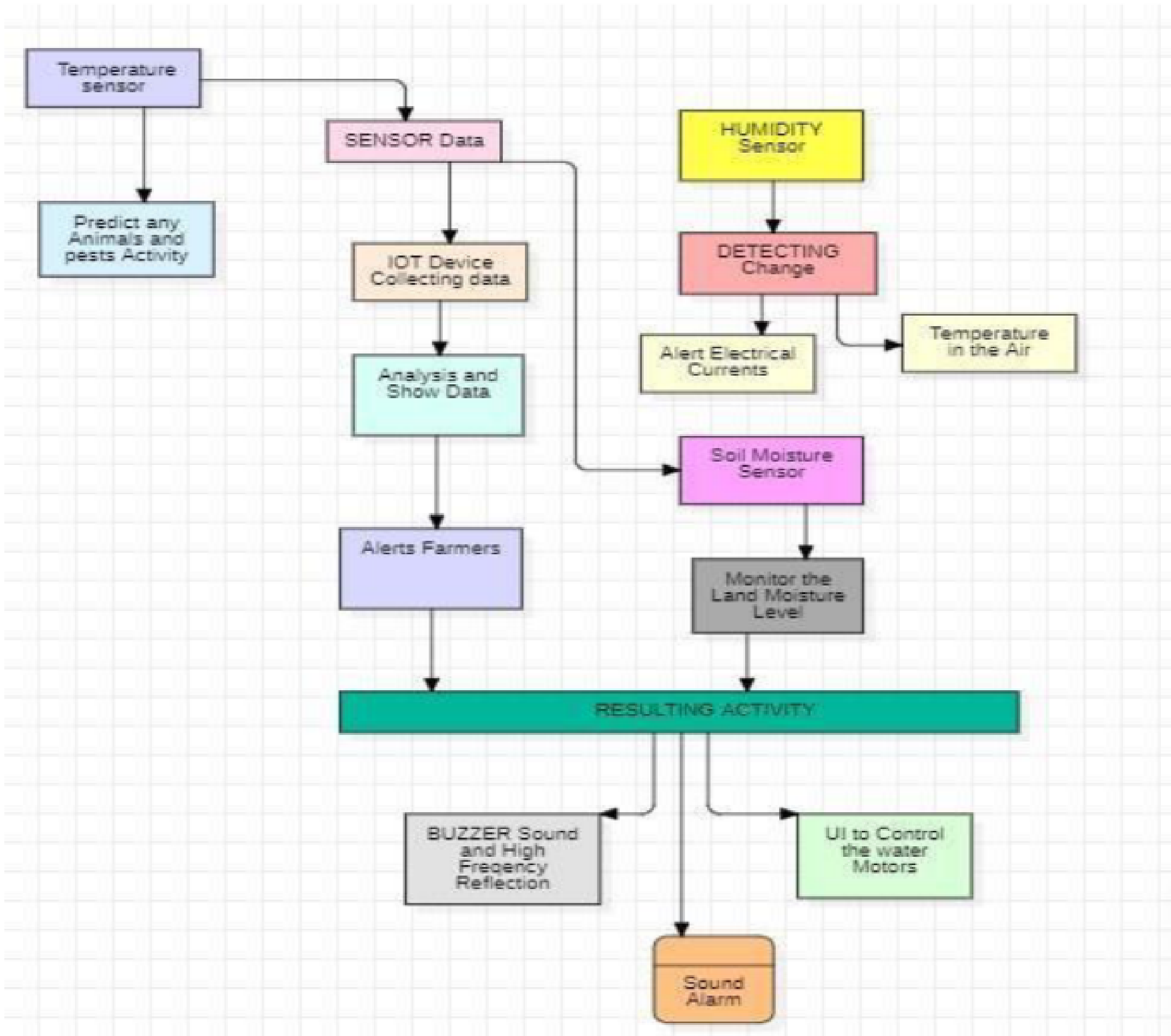
4.2 NON-FUNCTIONAL REQUIREMENTS

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

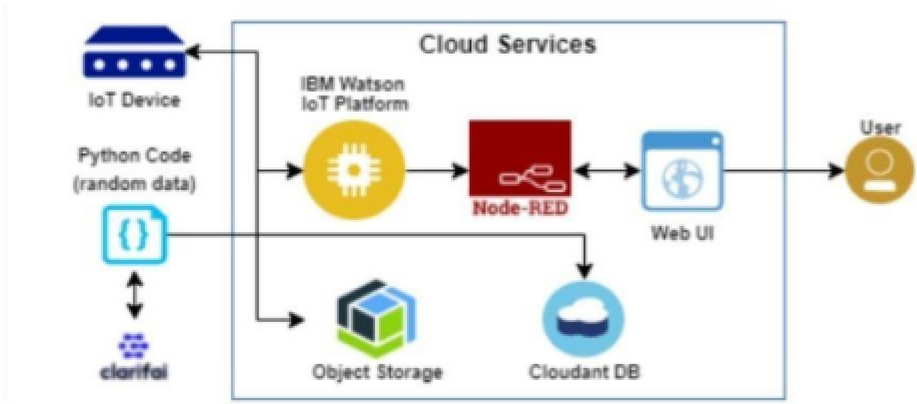
FR No	Non-Functional Requirement	Description
NFR-1	Usability	Mobile assistance. Given the capabilities of mobile devices, users must be able to interact in the same roles and tasks on PCs and mobile devices when practicable.
NFR-2	Security	Authorized users of the system who share information must be 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.
NFR-6	Scalability	System must manage increasing load and data retention requirements based on the scalability of the solution, such as additional buildings and manufacturing facilities.

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE



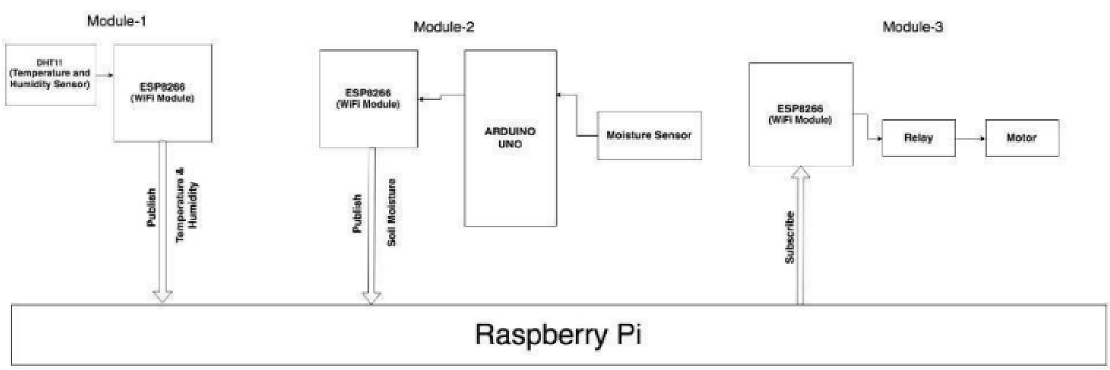


Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Mobile Application	HTML, CSS, JavaScript / AngularJS / Node Red.
2.	Application Logic-1	Logic for a process in the application	Java / Python
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 application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	IoT Model	Purpose of the IoT Model is for integrating the sensors with a user interface.	IBM IoT Platform
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

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 application	I can access my account /dashboard	High	Sprint 1
		USN-2	User can register their credentials like email id and password	I can receive confirmation email and click confirm	High	Sprint 1
	Login	USN-3	User can log into the application y entering email and password	I can login to my account	High	Sprint 1
	Dashboard	USN-4	User can View the temperature	I can view the data given by the device	High	Sprint 2
		USN-5	User can view the level of sensor monitoring value	I can view the data given by the device	High	Sprint 2
Customer (Web user)	Usage	USN-1	User can view the web page and get the information	I can view the data given by the device	High	Sprint 3
customer	Working	USN-1	User act according to the alert given by the device	I can get the data work according to it	High	Sprint 3
		USN-2	User turns ON the water motors/Buzzer/Sound Alarm when occur the disturbance on field	I can get the data work according to it		Sprint 4

Customer care Executive	Action	USN-1	User solve the problem when some faces any usage issues	I can solve the issues when someone fails to understand the procedure	High	Sprint 4
Administration	Administration	USN-1	Use store every information	I can store the gained information	High	Sprint 4

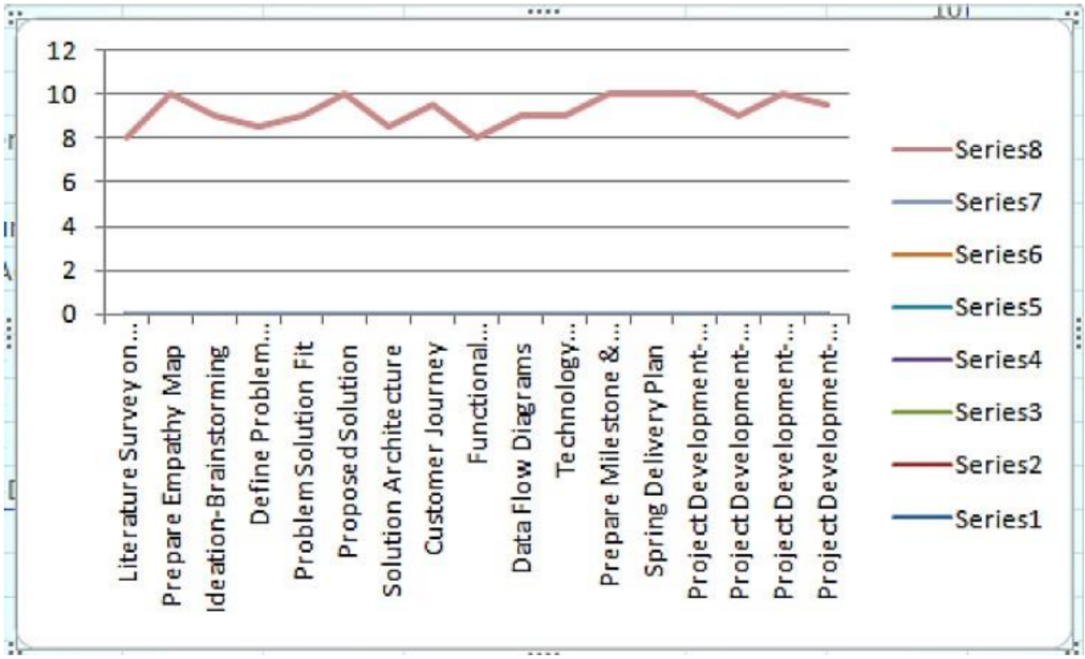
6. PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION:-

Sprint	Total Story Points	Duration	Sprint start date	Sprint End Date (planned)	Story point completed (as on planned end date)	Sprint Release Date (actual)
Sprint 1	20	8 days	26 Oct 2022	**2 Nov 2022**	**20**	
Sprint 2	20	**8 days**	**7 Nov 2022**	**14 Nov 2022**	**20**	
Sprint 3	**20**	**8 days**	**14 Nov 2022**	**21 Nov 2022**	**20**	
Sprint 4	**20**	**8 days**	**19 Nov 2022**	**26 Nov 2022**	**20**	

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 iteration unit (story points per day)

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



7. CODING AND SOLUTIONING

Feature - 1 :-

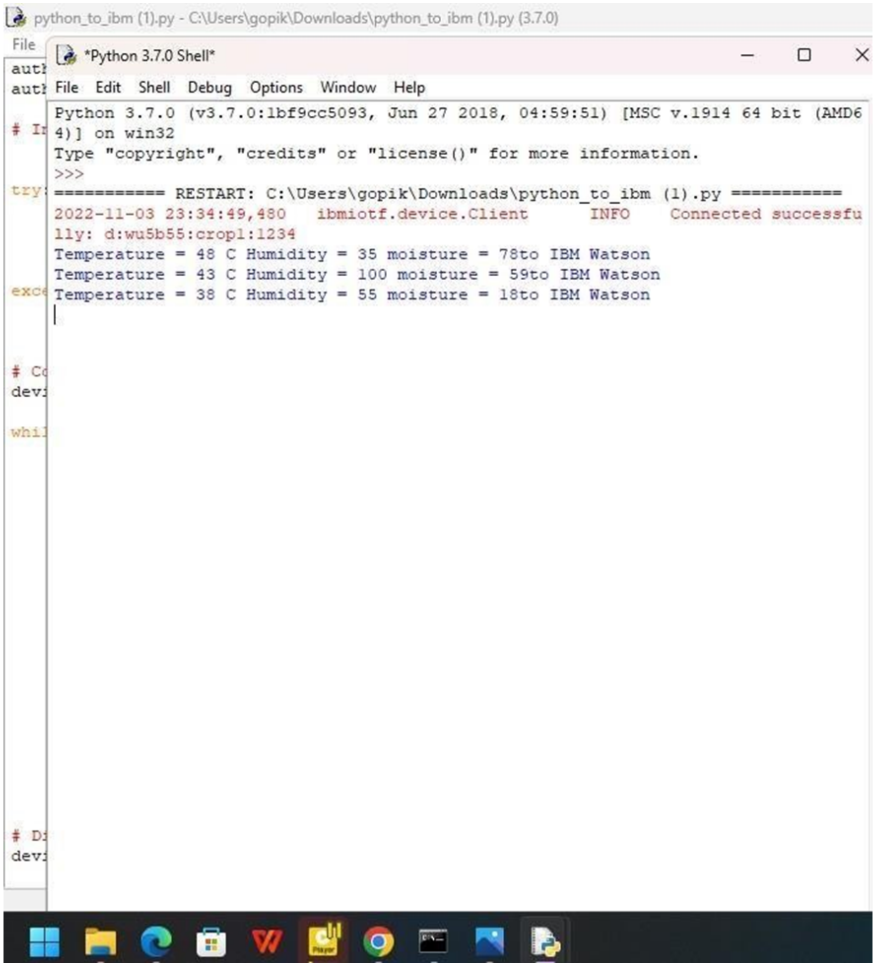
```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "041ole"
deviceType = "IoT"
deviceId = "12345"
authMethod = "token"
authToken = "12345689"
# Initialize GPIO
try:
deviceOptions = {"org": organization, "type":
deviceType, "id": deviceId, "authmethod":
authMethod, "auth-token": authToken}
deviceCli =
ibmiotf.device.Client(deviceOptions)
#.............................................
except Exception as e:
print("Caught exception connecting device: %s"
% str(e))
sys.exit()
# Connect and send a datapoint "hello" with
```

value "world" into the cloud as an event of type "greeting" 10 times

```
deviceCli.connect()
while True:
temp=random.randint(0,100)
Hum=random.randint(0,100)
moisture=random.randint(0,100)
data = { 'temperature' : temp, 'Humidity': Hum,
'Moisture':moisture }
def myOnPublishCallback():
print ("Temperature = " + str(temp)+" C
Humidity = " + str(hum)+ " moisture = " +
str(moisture) + "to IBM Watson")
success = deviceCli.publishEvent("IoTSensor",
"json", data, qos=0,
on_publish=myOnPublishCallback)
if not success:
print("Not connected to IoTF")
time.sleep(10)
deviceCli.commandCallback =
myCommandCallback
# Disconnect the device and application from the
cloud
deviceCli.disconnect()
```

Output:-

Output:-

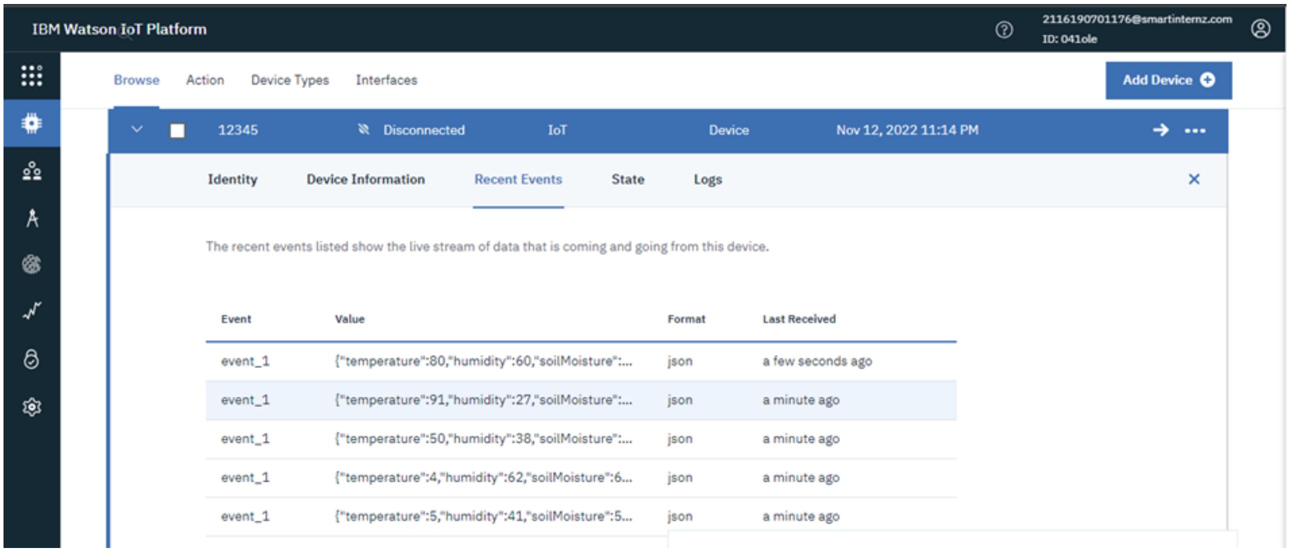


Source code is deployed on IBM Watson IoT platform to generate sensor data.

Source code

```
{
"temperature": random(0, 100),
"humidity": random(0, 100),
"moisture": random(0, 100),
" animalDetected ":random(0,2)
}
```

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.

7.2 Feature 2:-

i. Goodsensitivity to Combustible gas in wide range .

ii. Highsensitivity to LPG, Propane and Hydrogen .

iii. Longlife and low cost.

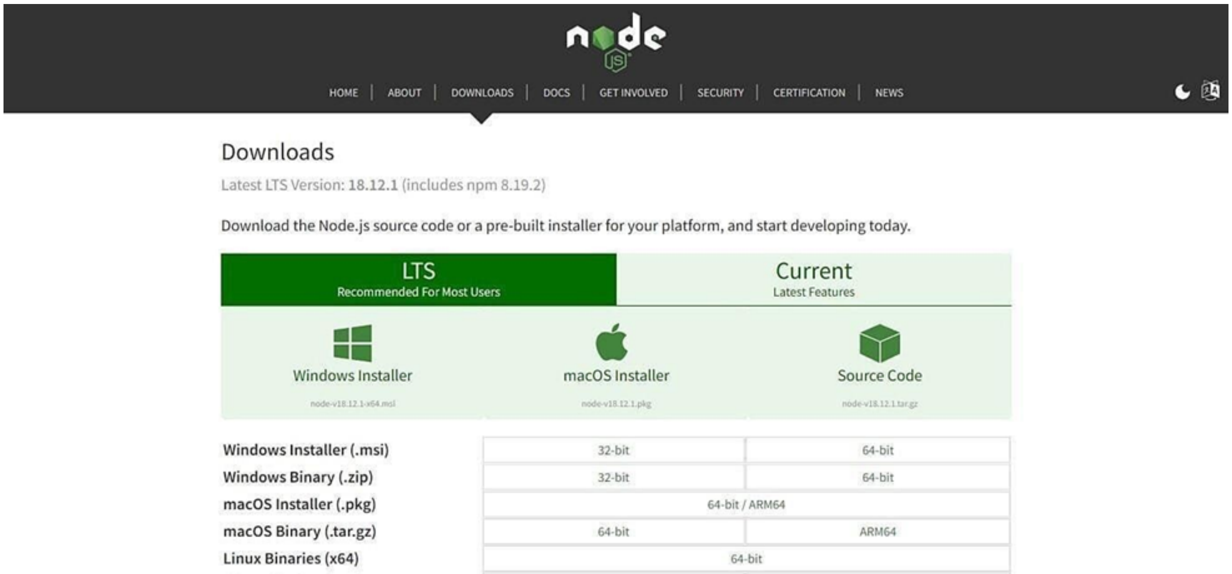
iv. Simpledrive circuit.

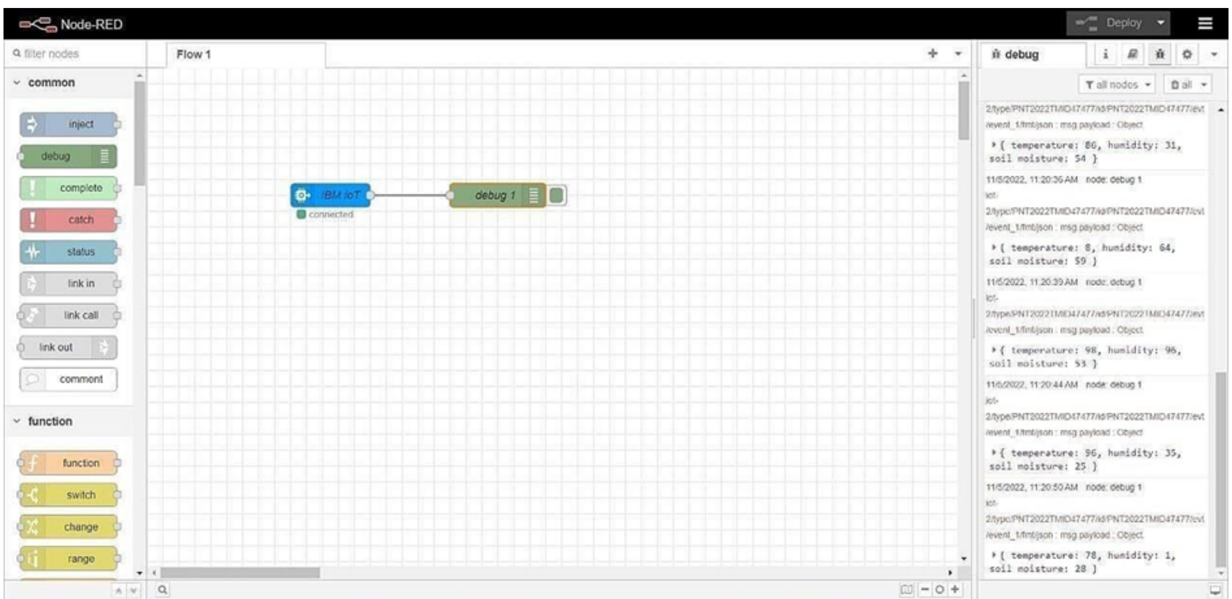
8.TESTING

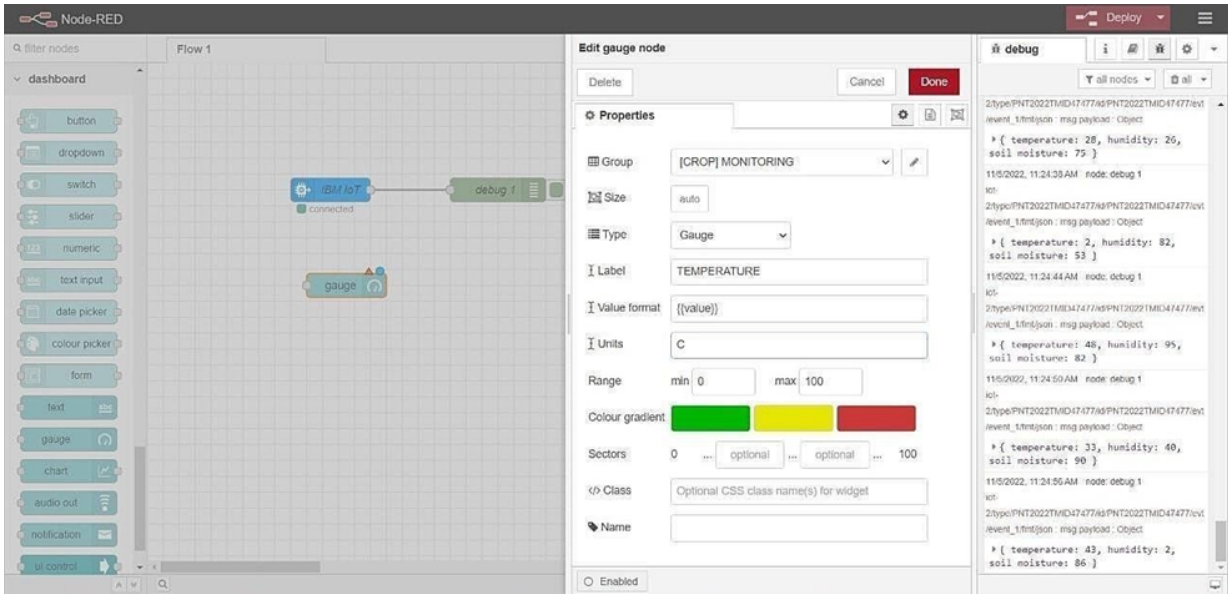
8.1 TEST CASES:-

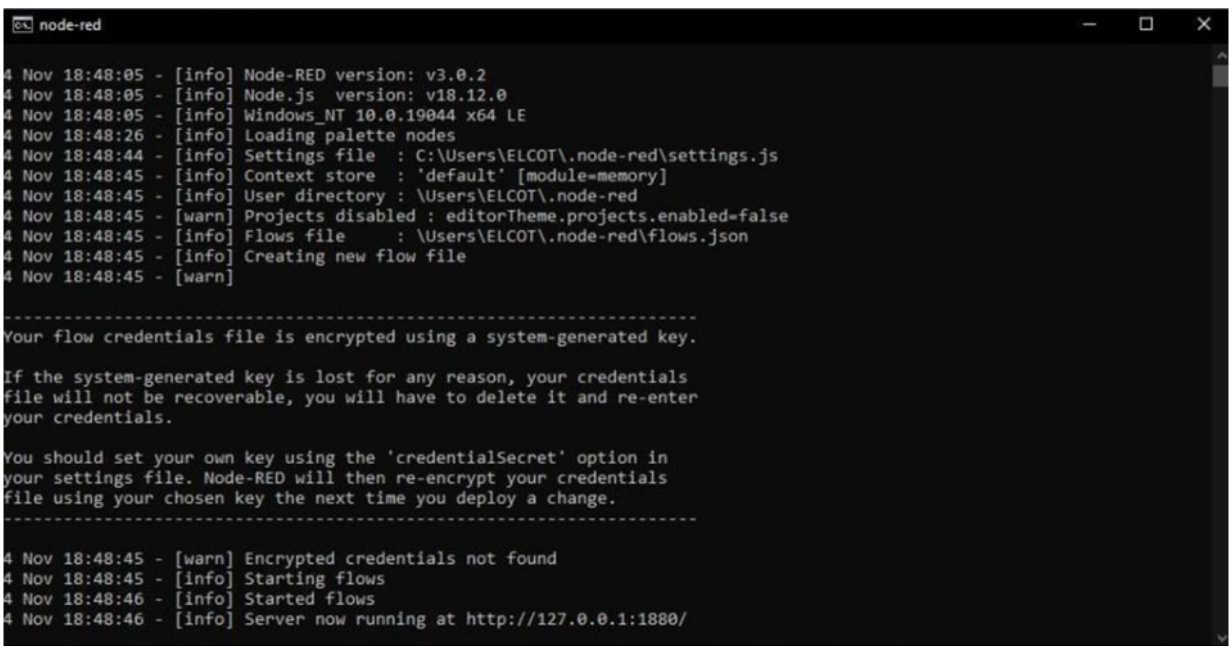
	Parameter	values	screenshot
1	model summary	-	
2	accuracy	training accuracy 95% validation accuracy 72%	
3	confidence score	class detected 80% confidence score 80%	

8.2 USER ACCEPTANCE TESTING






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 }
11/5/2022, 11:24:38 AM node: debug 1
{ temperature: 2, humidity: 82, soil moisture: 53 }
11/5/2022, 11:24:44 AM node: debug 1
{ temperature: 48, humidity: 95, soil moisture: 82 }
11/5/2022, 11:24:50 AM node: debug 1
{ temperature: 33, humidity: 40, soil moisture: 90 }
11/5/2022, 11:24:56 AM node: debug 1
{ temperature: 43, humidity: 2, soil moisture: 86 }


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/

9. RESULTS

The problem of Crop vandalisation 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 the orchards and Fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection of their fields.

10. ADVANTAGES AND DISADVANTAGES

Advantages:

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 starving. 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 IoT platform, Watsonsimulator, 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 animals and fire can be detected by cameras and if it comes towards farms then the 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 be activated.

13. APPENDIX

13.1 SOURCE CODE:-


```
[
  {
    "id": "970066790c008e12",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": "",
    "env": []
  },
  {
    "id": "b058b4f450dd6536",
    "type": "ibmiot in",
    "z": "970066790c008e12",
    "authentication": "apiKey",
    "apiKey": "1e36be401deafae4",
    "inputType": "evt",
    "logicalInterface": "",
    "ruleId": "",
    "deviceId": "",
    "applicationId": "",
    "deviceType": "+",
    "eventType": "+",
    "commandType": "",
    "format": "json",
    "name": "IBM IoT",
    "service": "registered",
    "allDevices": "",
    "allApplications": "",
    "allDeviceTypes": true,
    "allLogicalInterfaces": "",
    "allEvents": true,
    "allCommands": "",
    "allFormats": "",
    "qos": 0,
    "x": 120,
    "y": 235,
    "wires": [
      [
```

```
            "506e96584b1c1fd6",
            "0ec2290a3291a594",
            "95a0b843b5ad5116",
            "0969093df4ed6ef5"
        ]
    ],
    "l": false
},
{
    "id": "506e96584b1c1fd6",
    "type": "function",
    "z": "970066790c008e12",
    "name": "Temperature",
    "func": "msg.payload=msg.payload.temperature;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 269.5,
    "y": 214,
    "wires": [
        [
            "595552e55599f039",
            "0969093df4ed6ef5"
        ]
    ],
    "l": false
},
{
    "id": "0ec2290a3291a594",
    "type": "function",
    "z": "970066790c008e12",
    "name": "Humidity",
    "func": "msg.payload=msg.payload.humidity;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
```

```
    "x": 271.5,
    "y": 285,
    "wires": [
      [
        "0b245f809f4583f2",
        "0969093df4ed6ef5"
      ]
    ],
    "l": false
  },
  {
    "id": "95a0b843b5ad5116",
    "type": "function",
    "z": "970066790c008e12",
    "name": "Soil Moisture",
    "func": "msg.payload=msg.payload.soilMoisture;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 275.5,
    "y": 353,
    "wires": [
      [
        "05574b9dedf57636",
        "0969093df4ed6ef5"
      ]
    ],
    "l": false
  },
  {
    "id": "595552e55599f039",
    "type": "ui_gauge",
    "z": "970066790c008e12",
    "name": "",
    "group": "2cfb16950d96cc31",
    "order": 1,
    "width": 0,
    "height": 0,
```

"gtype": "gage",
"title": "Temperature",
"label": "units",
"format": "{{value}}",
"min": 0,
"max": "50",
"colors": [
"#00b500",
"#e6e600",
"#ca3838"
],
"seg1": "",
"seg2": "",
"className": "",
"x": 421.5,
"y": 210,
"wires": [],
"l": false
},
{
"id": "0b245f809f4583f2",
"type": "ui_gauge",
"z": "970066790c008e12",
"name": "",
"group": "2cfb16950d96cc31",
"order": 2,
"width": 0,
"height": 0,
"gtype": "gage",
"title": "Humidity",
"label": "units",
"format": "{{value}}",
"min": 0,
"max": "100",
"colors": [
"#00b500",
"#e6e600",
"#ca3838"
],
"seg1": "",

```
    "seg2": "",
    "className": "",
    "x": 425.5,
    "y": 271,
    "wires": [],
    "l": false
},
{
    "id": "05574b9dedf57636",
    "type": "ui_gauge",
    "z": "970066790c008e12",
    "name": "",
    "group": "2cfb16950d96cc31",
    "order": 3,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "Soil Moisture",
    "label": "units",
    "format": "{{value}}",
    "min": 0,
    "max": "100",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "className": "",
    "x": 419.5,
    "y": 351,
    "wires": [],
    "l": false
},
{
    "id": "0969093df4ed6ef5",
    "type": "debug",
    "z": "970066790c008e12",
    "name": "",
```

```
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 581.5,
    "y": 265,
    "wires": [],
    "l": false
  },
  {
    "id": "a2b01fd3bbae8067",
    "type": "ui_button",
    "z": "970066790c008e12",
    "name": "",
    "group": "2cfb16950d96cc31",
    "order": 4,
    "width": 0,
    "height": 0,
    "passthru": false,
    "label": "Motor On",
    "tooltip": "",
    "color": "",
    "bgcolor": "",
    "className": "",
    "icon": "",
    "payload": "",
    "payloadType": "str",
    "topic": "topic",
    "topicType": "msg",
    "x": 189.5,
    "y": 420,
    "wires": [
      []
    ],
    "l": false
  },
```

{
"id": "8dcfa00a88ae9f89",
"type": "ui_button",
"z": "970066790c008e12",
"name": "",
"group": "2cfb16950d96cc31",
"order": 5,
"width": 0,
"height": 0,
"passthru": false,
"label": "Motor Off",
"tooltip": "",
"color": "",
"bgcolor": "",
"className": "",
"icon": "",
"payload": "",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",
"x": 300.5,
"y": 416,
"wires": [
[]
],
"l": false
},
{
"id": "521f3ad04f6a89c6",
"type": "ui_button",
"z": "970066790c008e12",
"name": "",
"group": "2cfb16950d96cc31",
"order": 8,
"width": 0,
"height": 0,
"passthru": false,
"label": "Take a picture",
"tooltip": "",
"color": "",

"bgcolor": "",
"className": "",
"icon": "",
"payload": "",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",
"x": 185.5,
"y": 474,
"wires": [
[
"c5a91ac989c3dbc6"
]
],
"l": false
},
{
"id": "ee4ec4e98677d9f3",
"type": "ui_button",
"z": "970066790c008e12",
"name": "",
"group": "2cfb16950d96cc31",
"order": 6,
"width": 0,
"height": 0,
"passthru": false,
"label": "Alarm On",
"tooltip": "",
"color": "",
"bgcolor": "",
"className": "",
"icon": "",
"payload": "",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",
"x": 185.5,
"y": 543,
"wires": [
[]

],
"l": false
},
{
"id": "16c981887b6f3424",
"type": "ui_button",
"z": "970066790c008e12",
"name": "",
"group": "2cfb16950d96cc31",
"order": 7,
"width": 0,
"height": 0,
"passthru": false,
"label": "Alarm Off",
"tooltip": "",
"color": "",
"bgcolor": "",
"className": "",
"icon": "",
"payload": "",
"payloadType": "str",
"topic": "topic",
"topicType": "msg",
"x": 313.5,
"y": 541,
"wires": [
[]
],
"l": false
},
{
"id": "c5a91ac989c3dbc6",
"type": "http request",
"z": "970066790c008e12",
"name": "",
"method": "use",
"ret": "txt",
"paytoqs": "ignore",
"url":
"https://www.pexels.com/photo/nature-animals-pig-alp-rona-63285/",

"tls": "",
"persist": false,
"proxy": "",
"authType": "",
"senderr": false,
"x": 320.5,
"y": 472,
"wires": [
[
"5c8416e90f721e9a"
]
],
"l": false
},
{
"id": "5c8416e90f721e9a",
"type": "debug",
"z": "970066790c008e12",
"name": "",
"active": true,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "false",
"statusVal": "",
"statusType": "auto",
"x": 449.5,
"y": 458,
"wires": [],
"l": false
},
{
"id": "b4b91d9b482b8ac0",
"type": "ui_spacer",
"z": "970066790c008e12",
"name": "spacer",
"group": "2cfb16950d96cc31",
"order": 1,
"width": 1,
"height": 1

    },
    {
        "id": "1e36be401deafae4",
        "type": "ibmiot",
        "name": "IBMiot",
        "keepalive": "60",
        "serverName": "",
        "cleansession": true,
        "appId": "",
        "shared": false
    },
    {
        "id": "2cfb16950d96cc31",
        "type": "ui_group",
        "name": "Crop Protection System",
        "tab": "f020ef3daae2da71",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false,
        "className": ""
    },
    {
        "id": "f020ef3daae2da71",
        "type": "ui_tab",
        "name": "Demo",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]

13.2 GITHUB AND PROJECT DEMO LINK:-

Github Link:-
https://github.com/IBM-EPBL/IBM-Project-19320-1659696121

Project Demo Link:-