IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID : PNT2022TMID33498

Team Members

Gayathri L

Bhuvaneshwari

Kanimozhi T

Gopika S

INTRODUCTION

PROJECT OVREVIEW:

Agriculture is the backbone of the economy but because of animal interference in agricultural lands, there will be huge loss of crops. This article provides a comprehensive review of various methods adopted by farmers to protect their crops. The article also discusses use of modern technology in agriculture. Finally, this article reviews smart crop protection system using sensors, microcontroller and gsm module. 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 kinetics 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. 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 Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other objects to connect and items embedded with electronics, software, sensors, actuators, and connectivity which enables these exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure. Agriculture uses 85% of available freshwater resources worldwide, and this percentage will continue to be dominant in water consumption because of population growth and increased food demand. There is an urgent need to create strategies based on science and technology for sustainable use of water, including technical, agronomic, managerial, and institutional improvement. Hence there is need to implement modern science and technology in the agriculture sector for increasing the yield. Most of the papers signifies the use of wireless sensor networkwhich collects the data from different types of sensors and then send it to main server using wireless protocol. The paper aims at making agriculture smart using automation and IoT technologies. The highlighting features of this paper includes smart transmission of data using cloud

(Things peak). Secondly, it includes smart irrigation with smart control based on real time field data. Thirdly, smart warehouse management which includes; temperature maintenance, humidity maintenance, soil moisture maintenance and pH maintenance.

REFERENCES:

Zuraida Muhammad,Muhammad Azri Asyraf Mohd Hafez,Nor Adni Mat" Smart Agriculture Using Internet of Things with Raspberry Pi." 2020.

Divya J, Divya M,JananiV." IoT based Smart Soil Monitoring System for Agricultural Production"
2017.

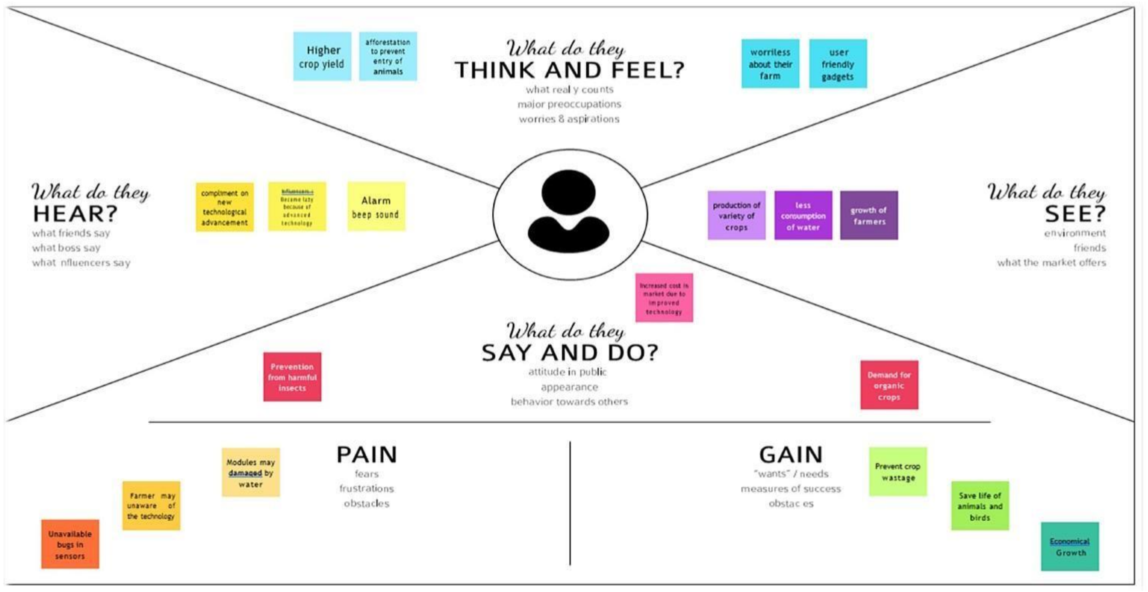
H.G.C.R.Laksiri, H.A.C.Dharmagunawardhana, J.V.Wijayakulasooriya " Design and Optimization of loT Based Smart Irrigation System in Sri Lanka"2019.

PROBLEM STATEMENT DEFINITION STATEMENT:

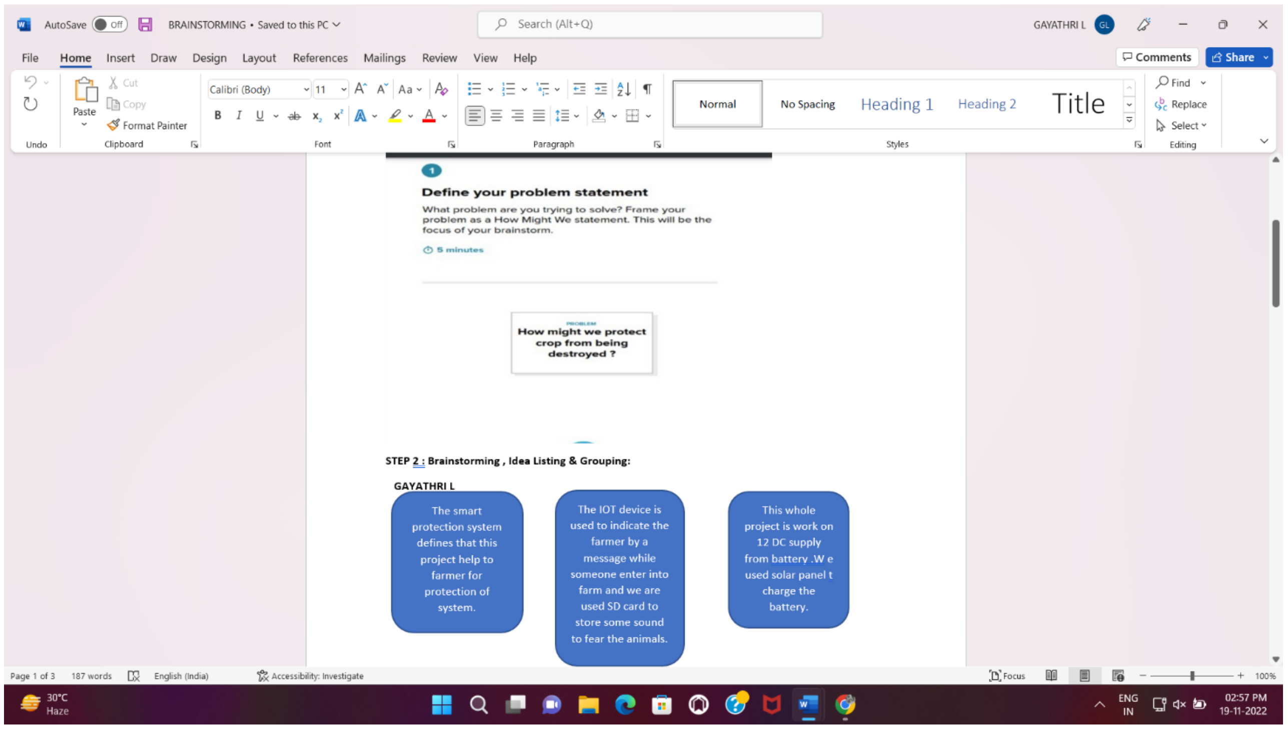
By this modernized agriculture we can provide high quality crops and also preserving the crops from the animals. This can be achievable by using the IOT sensor .Now a days ,crops damagedby insects and some other animals , so we decided to do projecton smart crop protection based on IOT. This is very important to turn towards modernized agriculture of existing methodsand using Information Technology to predictthe crops from destroying.

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:

a. 

IDEATION AND BRAINSTORMING:



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we protect crop from being destroyed ?

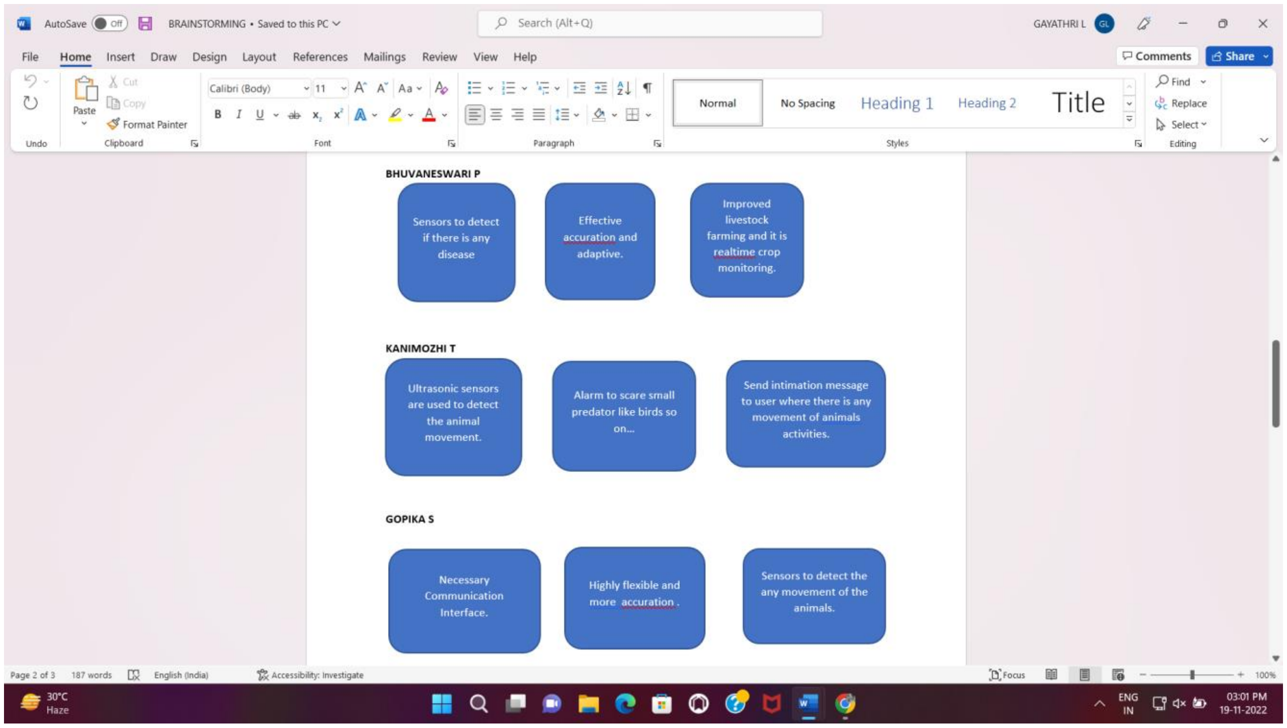
STEP 2 : Brainstorming , Idea Listing & Grouping:

GAYATHRI L

The smart protection system defines that this project help to farmer for protection of system.

The IOT device is used to indicate the farmer by a message while someone enter into farm and we are used SD card to store some sound to fear the animals.

This whole project is work on 12 DC supply from battery .W e used solar panel t charge the battery.



BHUVANESWARI P

Sensors to detect if there is any disease

Effective accuration and adaptive.

Improved livestock farming and it is realtime crop monitoring.

KANIMOZHI T

Ultrasonic sensors are used to detect the animal movement.

Alarm to scare small predator like birds so on…

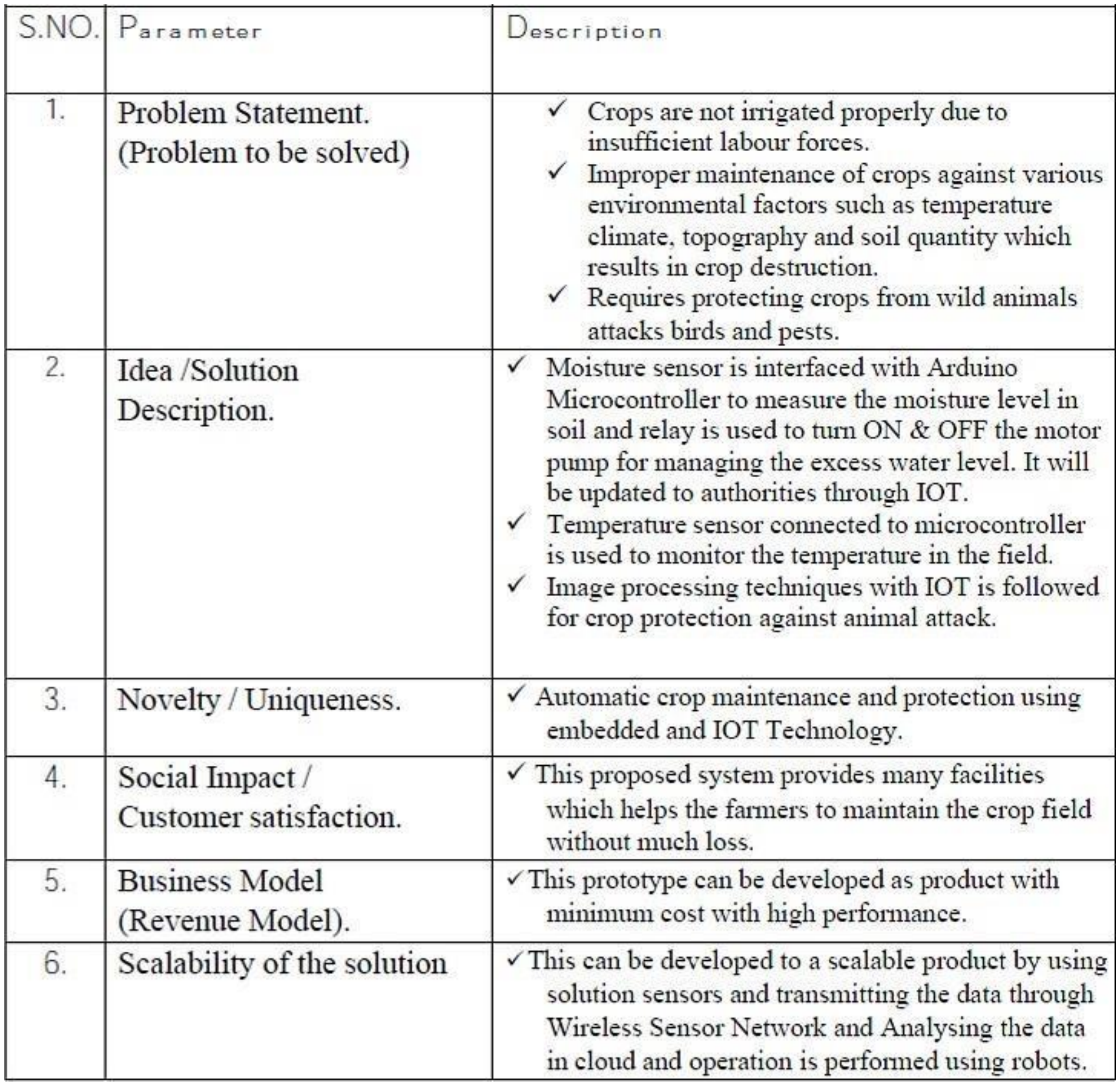
Send intimation message to user where there is any movement of animals activities.

GOPIKA S

Necessary Communication Interface.

Highly flexible and more accuration .

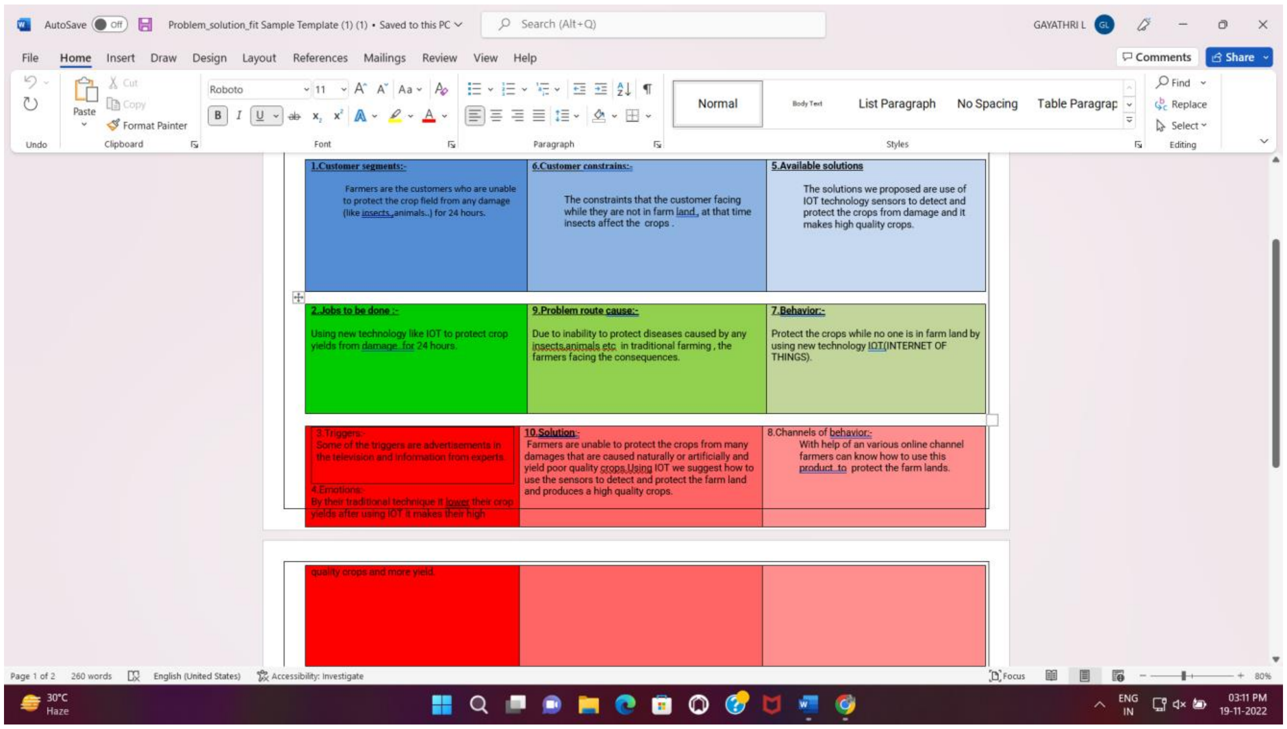
Sensors to detect the any movement of the animals.



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 quantity which results in crop destruction. ✓ 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 & 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 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 Analysing the data in cloud and operation is performed using robots.

a.

PROBLEM SOLUTIONFIT:

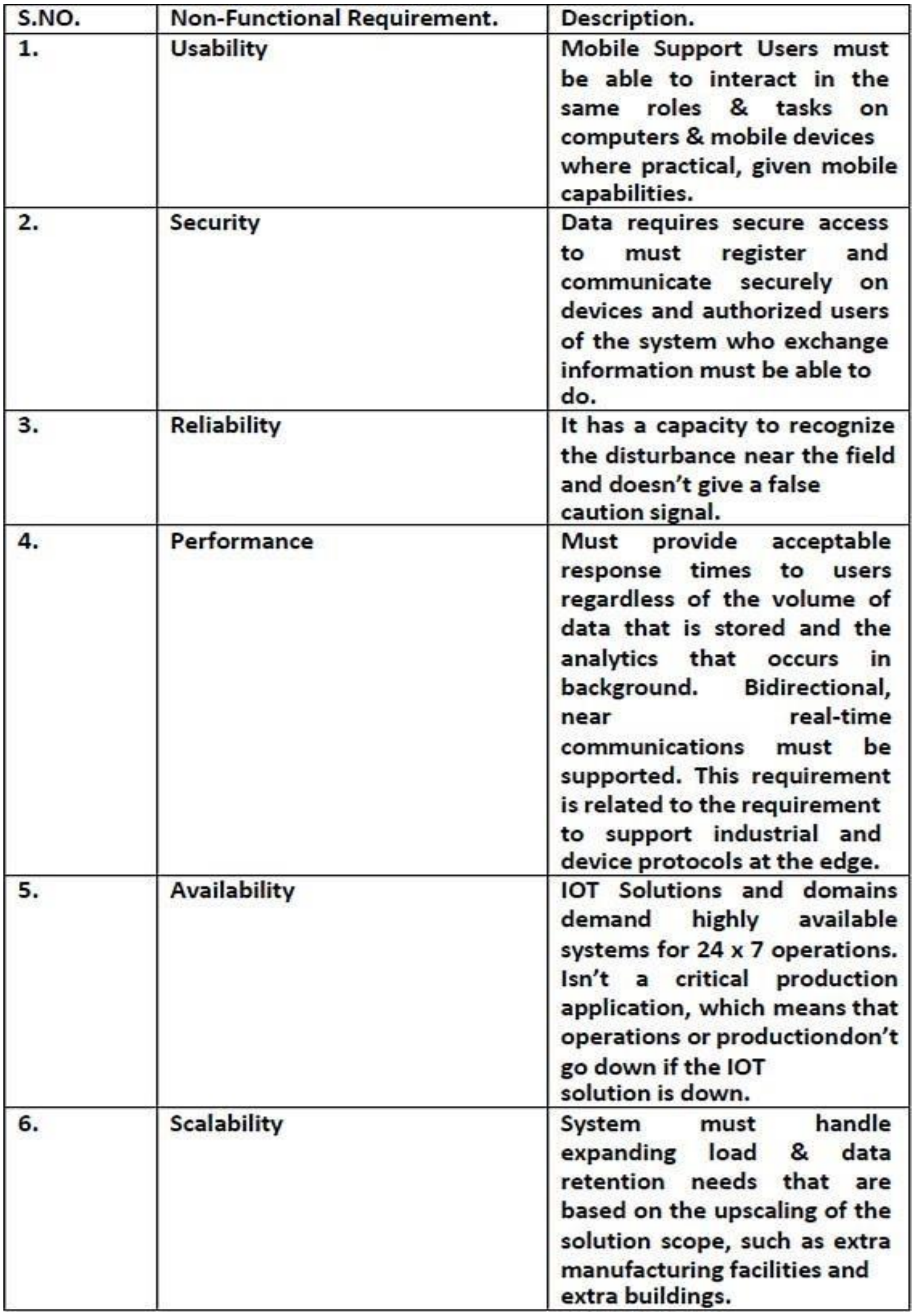


1.Customer segments:- Farmers are the customers who are unable to protect the crop field from any damage (like insects.,animals..) for 24 hours.	**6.Customer constrains:-** The constraints that the customer facing while they are not in farm land., at that time insects affect the crops .	**5.Available solutions** The solutions we proposed are use of IOT technology sensors to detect and protect the crops from damage and it makes high quality crops.
2.Jobs to be done :- Using new technology like IOT to protect crop yields from damage..for 24 hours.	**9.Problem route cause:-** Due to inability to protect diseases caused by any insects,animals etc in traditional farming , the farmers facing the consequences.	**7.Behavior:-** Protect the crops while no one is in farm land by using new technology IOT(INTERNET OF THINGS).
3.Triggers:- Some of the triggers are advertisements in the television and information from experts. 4.Emotions:- By their traditional technique it lower their crop yields after using IOT it makes their high quality crops and more yield.	**10.Solution**:- Farmers are unable to protect the crops from many damages that are caused naturally or artificially and yield poor quality crops.Using IOT we suggest how to use the sensors to detect and protect the farm land and produces a high quality crops.	8.Channels of behavior:- With help of an various online channel farmers can know how to use this product to protect the farm lands.

FUNCTIONAL REQUIREMENT:

S.NO.	Functional Requirement.	Sub Requirement.
1.	User Visibility	Sense animals nearing the crop field & sounds alarm to woo them away as well as sends SMS to farmer using cloud service.
2.	User Reception	The Data like values of Temperature, Humidity, Soil moisture Sensors are received via SMS.
3.	User Understanding	Based on the sensor data value to get the information about the present of farming land.
4.	User Action	The User needs take action like destruction of crop residues, deep plowing, crop rotation, fertilizers, strip cropping, scheduled planting operations.

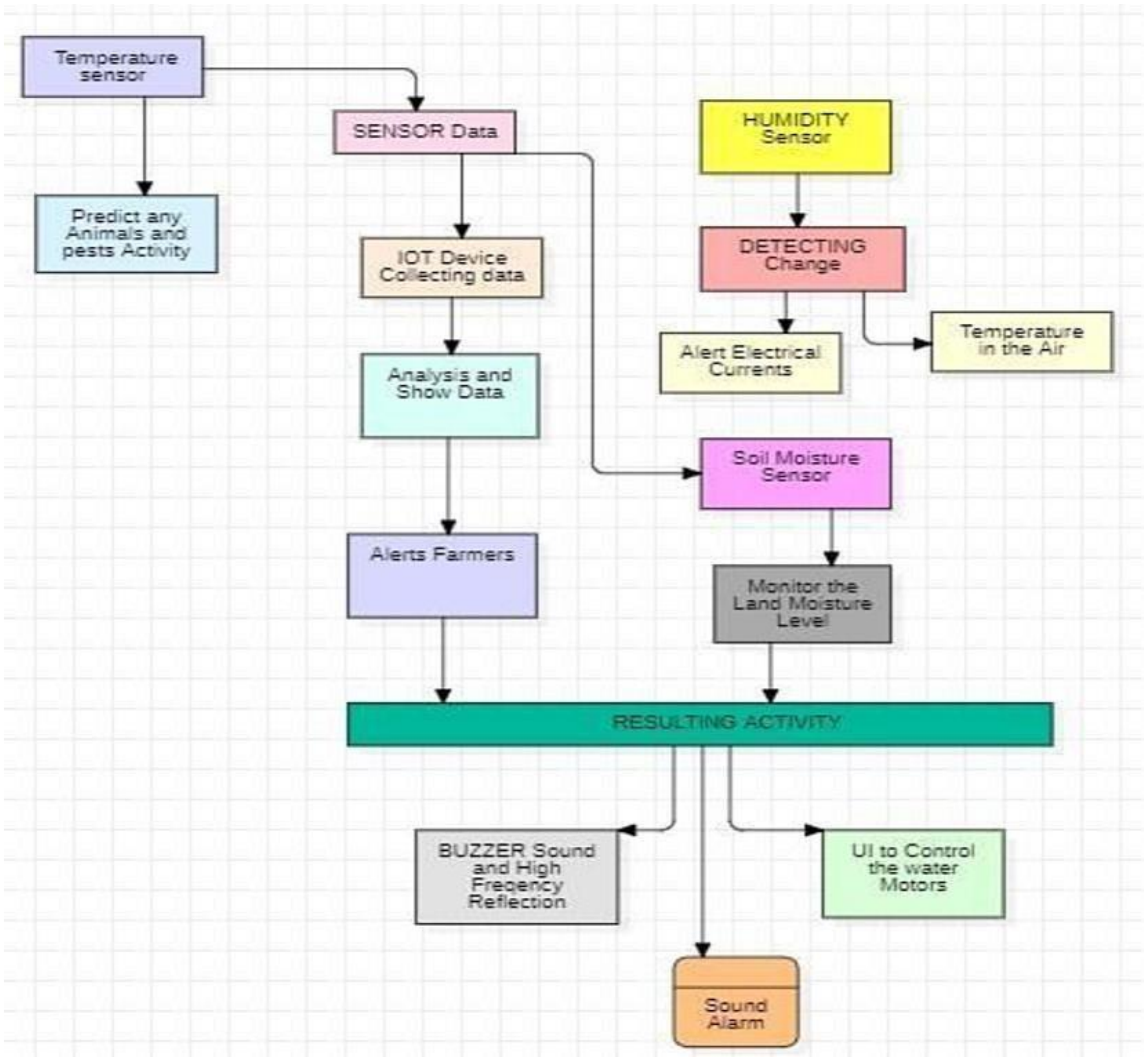
<u>NON FUNCTINAL REQUIREMENT:</u>



S.NO.	Non-Functional Requirement.	Description.
1.	Usability	Mobile Support Users must be able to interact in the same roles & tasks on computers & mobile devices where practical, given mobile capabilities.
2.	Security	Data requires secure access to must register and communicate securely on devices and authorized users of the system who exchange information must be able to do.
3.	Reliability	It has a capacity to recognize the disturbance near the field and doesn't give a false caution signal.
4.	Performance	Must provide acceptable response times to users regardless of the volume of data that is stored and the analytics that occurs in background. Bidirectional, near real-time communications must be supported. This requirement is related to the requirement to support industrial and device protocols at the edge.
5.	Availability	IOT Solutions and domains demand highly available systems for 24 x 7 operations. Isn't a critical production application, which means that operations or productiondon't go down if the IOT solution is down.
6.	Scalability	System must handle expanding load & data retention needs that are based on the upscaling of the solution scope, such as extra manufacturing facilities and extra buildings.

PROJECT DESIGN

DATA FLOW DIAGRAM:


Temperature sensor
SENSOR Data
HUMIDITY Sensor
Predict any Animals and pests Activity
IOT Device Collecting data
DETECTING Change
Alert Electrical Currents
Temperature in the Air
Analysis and Show Data
Soil Moisture Sensor
Alerts Farmers
Monitor the Land Moisture Level
RESULTING ACTIVITY
BUZZER Sound and High Freqency Reflection
UI to Control the water Motors
Sound Alarm

SOLUTION AND TECHNICAL ARCHITECTURE:

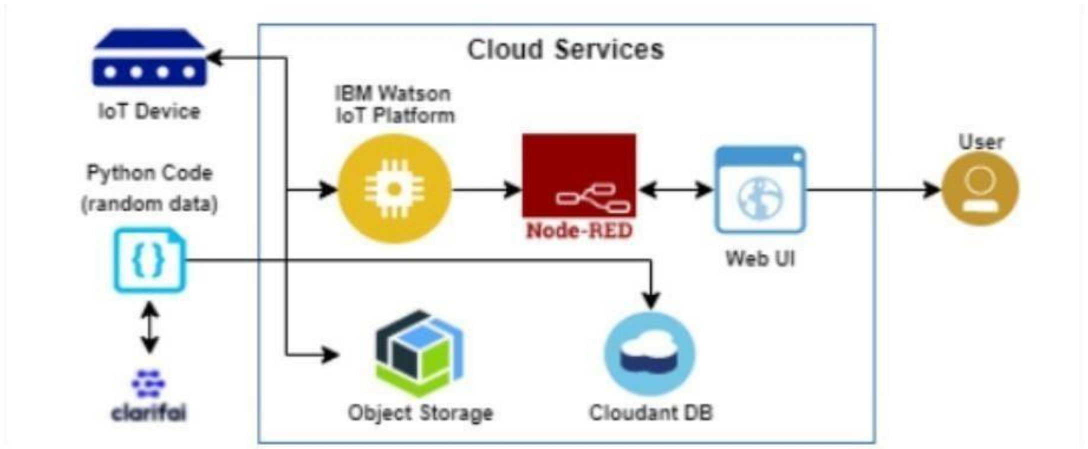


TABLE-1:

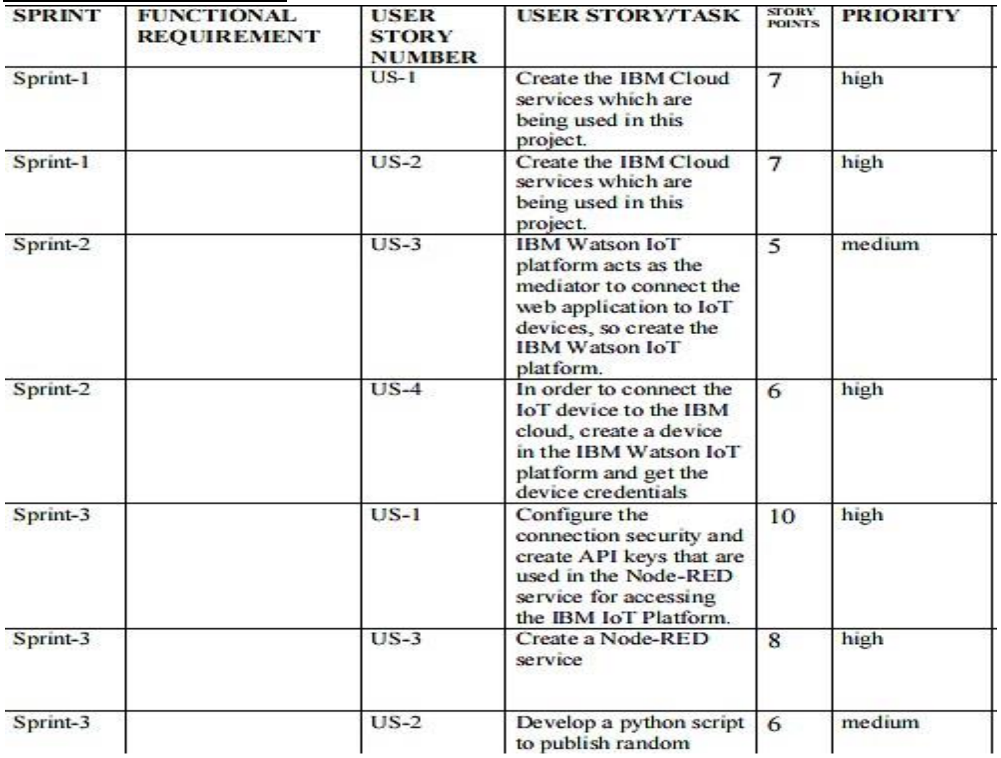
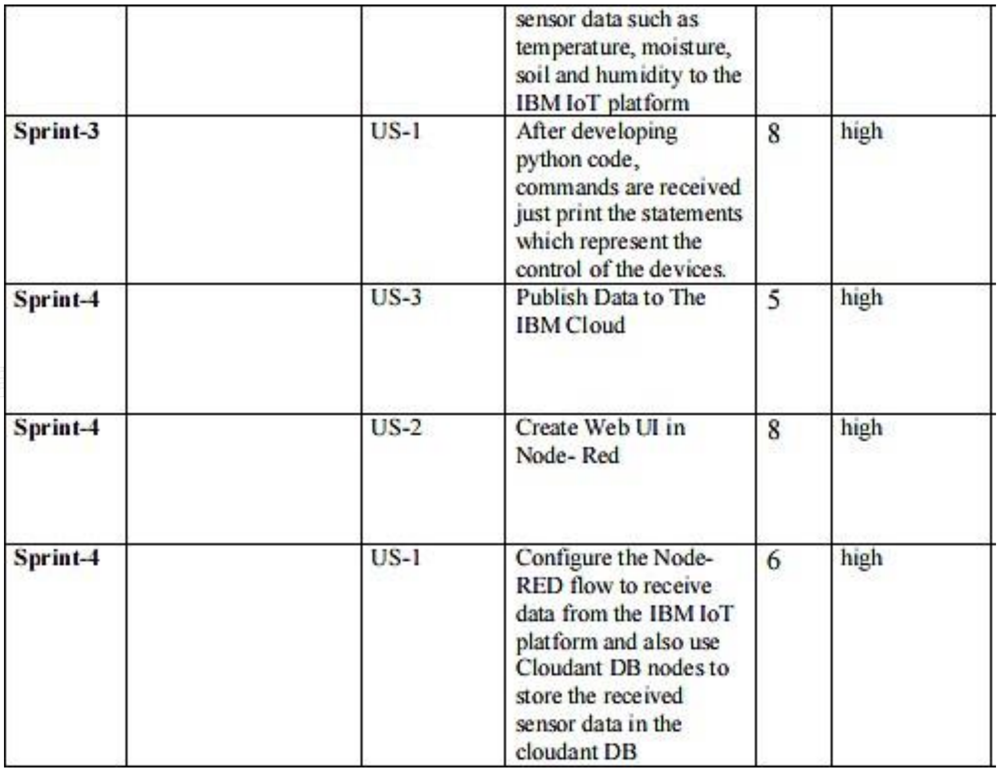
sno	components	description	Technology
1	User interface	Interacts with iot device	Html,css,angular js etc..
2	Application logic-1	Logic for a process in the application	Python
3	Application logic-2	Logic for process in the application	Clarifai
4	Application logic-3	Logic for process in the application	IBM Waston Iot platform
5	Application logic-4	logic for the process	Node red app service
6	User friendly	Easily manage the net screen appliance	Web uI

TABLE-2: **APPLICATION AND CHARACTERISTICS**

sno	Characteristics	Description	Technology
1	Open source framework	Open source framework used	Python
2	Security implementations	Authentication using encryption	Encryptions
3	Scalable architecture	The scalability of architecture consists of 3 models	Web UI Application server-python, clarifai Database server-ibm cloud services.
4	Availability	It is increased by cloudant database	IBM cloud services

a.

USER STORIES:

SPRINT	FUNCTIONAL REQUIREMENT	USER STORY NUMBER	USER STORY/TASK	STORY POINTS	PRIORITY
Sprint-1		US-1	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-1		US-2	Create the IBM Cloud services which are being used in this project.	7	high
Sprint-2		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
Sprint-2		US-4	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials	6	high
Sprint-3		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
Sprint-3		US-3	Create a Node-RED service	8	high
Sprint-3		US-2	Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform	6	medium
Sprint-3		US-1	After developing python code, commands are received just print the statements which represent the control of the devices.	8	high
Sprint-4		US-3	Publish Data to The IBM Cloud	5	high
Sprint-4		US-2	Create Web UI in Node- Red	8	high
Sprint-4		US-1	Configure the Node-RED flow to receive data from the IBM IoT platform and also use Cloudant DB nodes to store the received sensor data in the cloudant DB	6	high

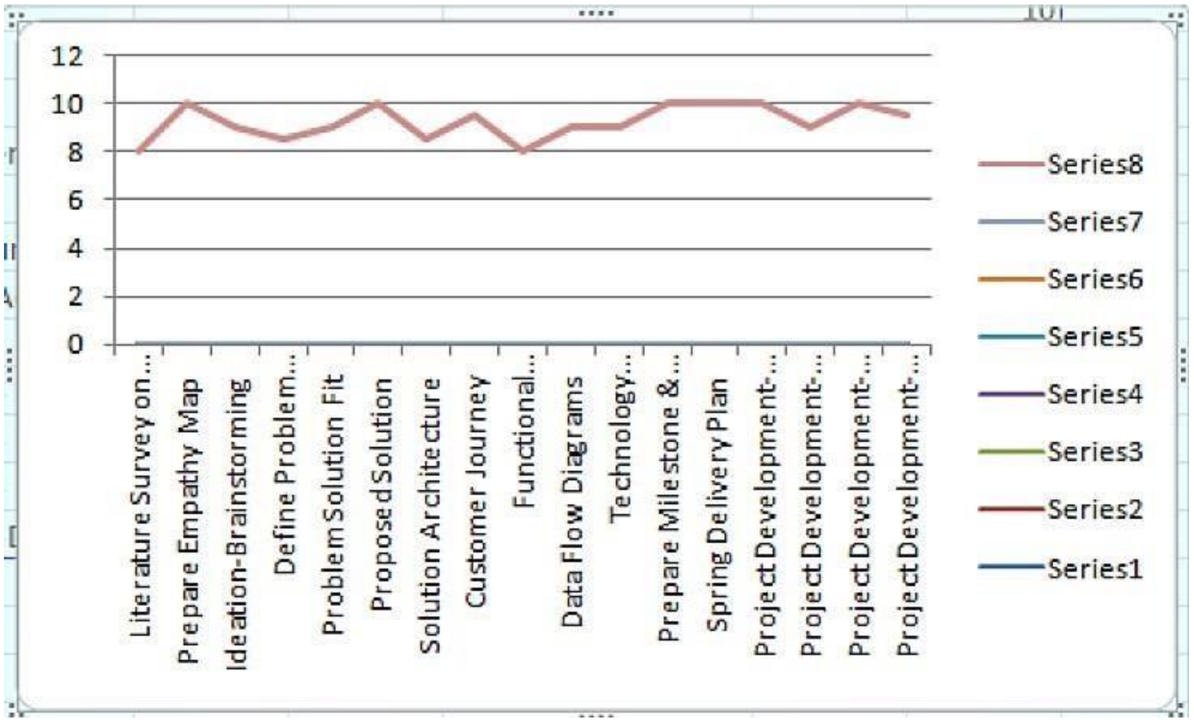
PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND ESTIMATION:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity
iteration unit (story points per day)

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

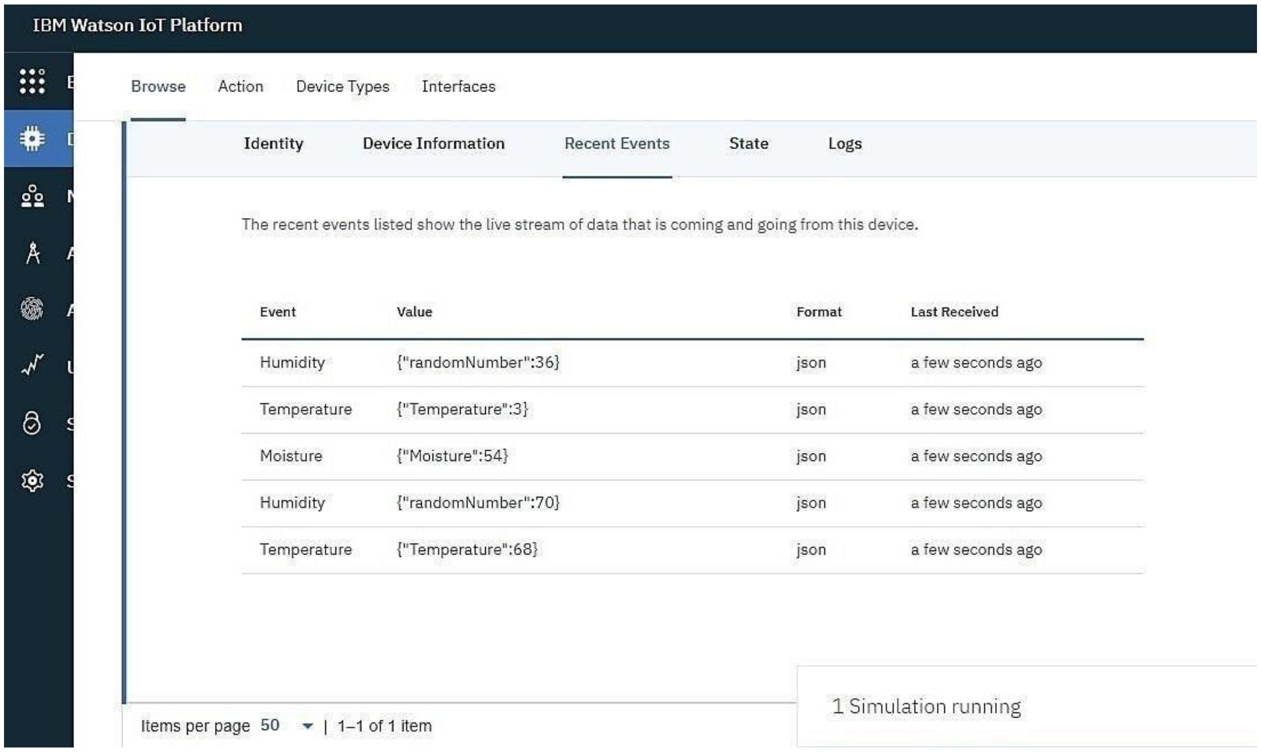


CODING AND SOLUTIONING

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()
#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()
```



The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Humidity	{"randomNumber":36}	json	a few seconds ago
Temperature	{"Temperature":3}	json	a few seconds ago
Moisture	{"Moisture":54}	json	a few seconds ago
Humidity	{"randomNumber":70}	json	a few seconds ago
Temperature	{"Temperature":68}	json	a few seconds ago

Items per page 50 | 1–1 of 1 item

1 Simulation running

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:

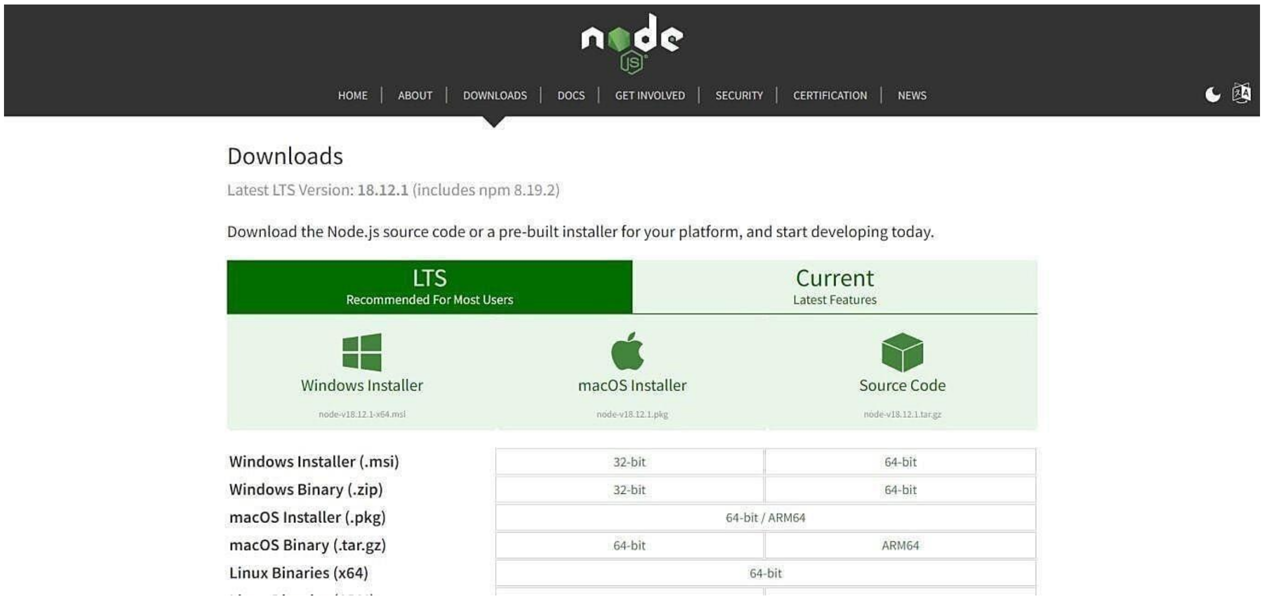
i. Goodsensitivity to Combustible gas in wide range .
ii. Highsensitivity to LPG, Propane and Hydrogen . iii. Longlife and low cost. iv. Simpledrive circuit.

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:



Downloads

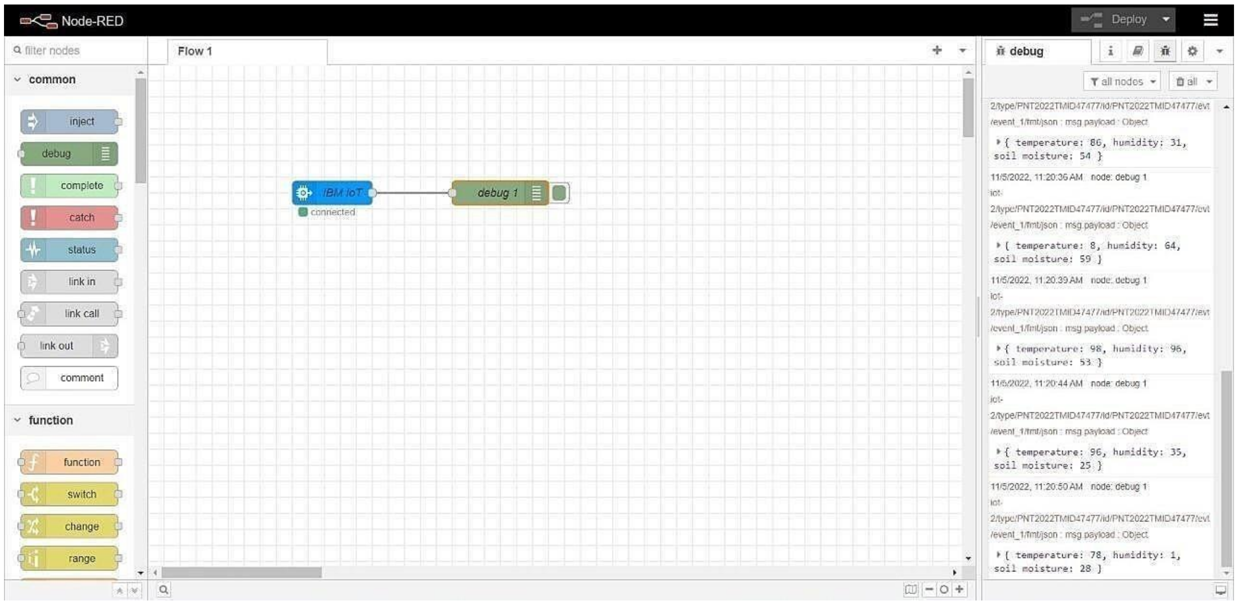
Latest LTS Version: **18.12.1** (includes npm 8.19.2)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

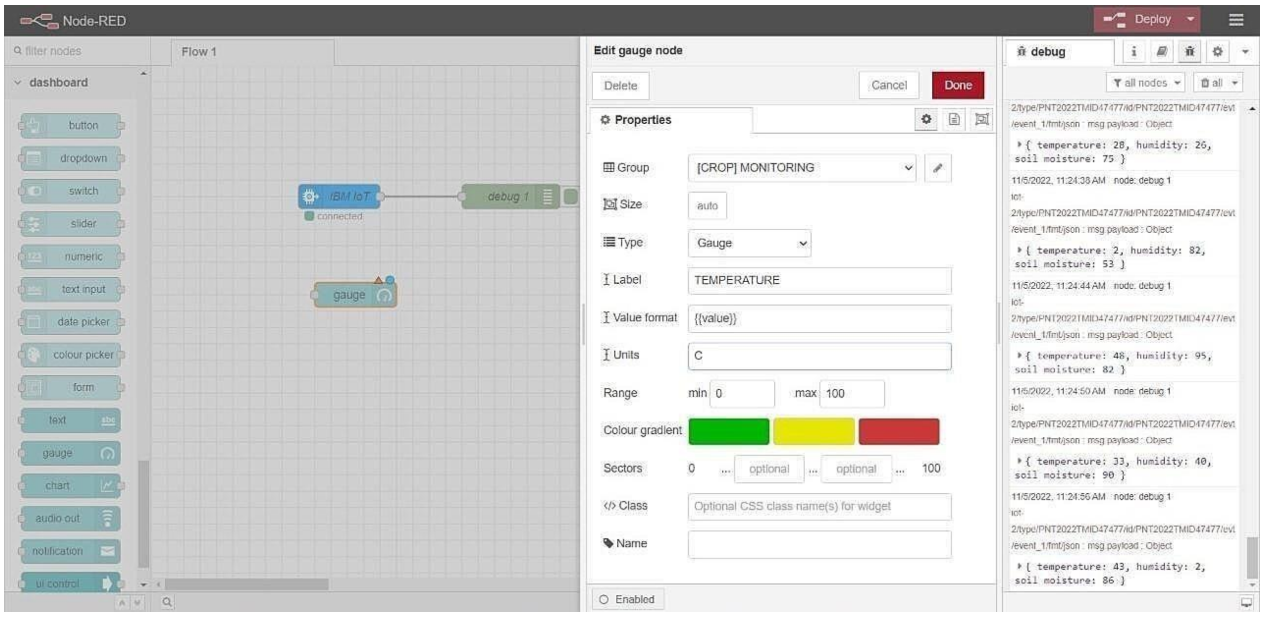
LTS Recommended For Most Users	Current Latest Features

Windows Installer	macOS Installer	Source Code
node-v18.12.1-x64.msi	node-v18.12.1.pkg	node-v18.12.1.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	



```
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 86, humidity: 31,
soil moisture: 54 }
11/5/2022, 11:20:36 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 8, humidity: 64,
soil moisture: 59 }
11/5/2022, 11:20:39 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 98, humidity: 96,
soil moisture: 53 }
11/5/2022, 11:20:44 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 96, humidity: 35,
soil moisture: 25 }
11/5/2022, 11:20:50 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 78, humidity: 1,
soil moisture: 28 }
```

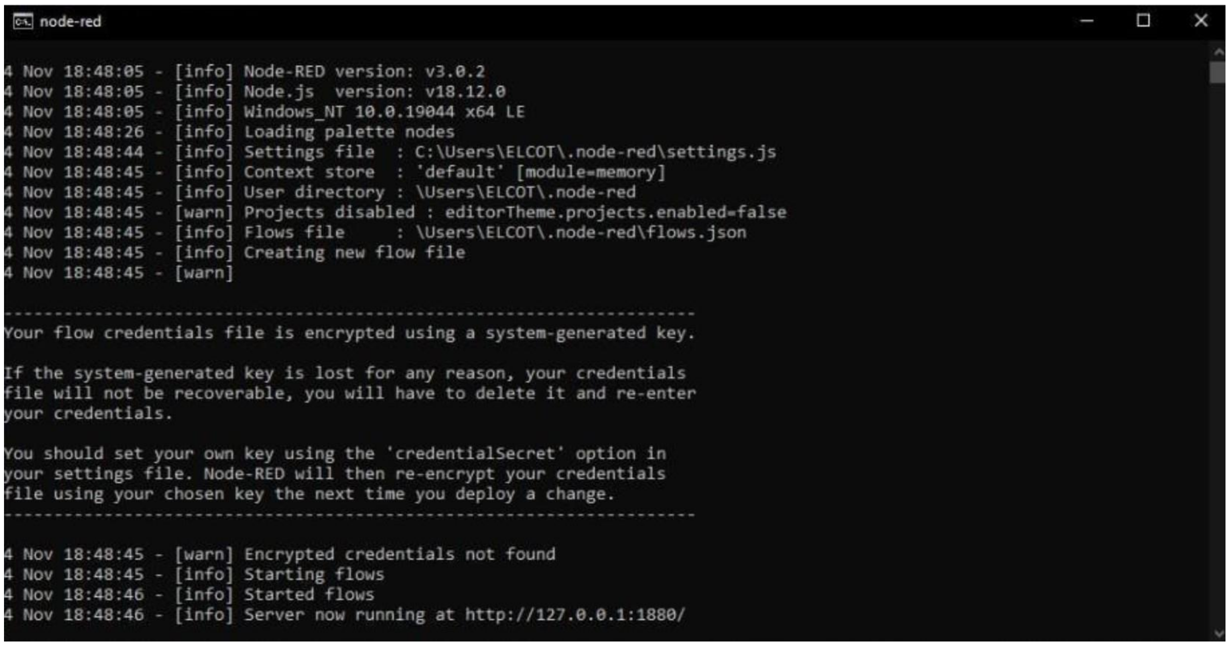


Edit gauge node

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:

debug

```
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 28, humidity: 26,
soil moisture: 75 }
11/5/2022, 11:24:38 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 2, humidity: 82,
soil moisture: 53 }
11/5/2022, 11:24:44 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 48, humidity: 95,
soil moisture: 82 }
11/5/2022, 11:24:50 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ temperature: 33, humidity: 40,
soil moisture: 90 }
11/5/2022, 11:24:56 AM  node: debug 1
iot-
2/type/PNT2022TMID47477/id/PNT2022TMID47477/evt
/event_1/fmt/json : msg.payload : Object
▸{ 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/
```

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 willhelp 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 chanceof 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 theenvironment 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 # toinstallpip
 install ibmiotf importibmiotf.device


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


 def myCommandCallback(cmd): # function for Callbackif


     cm.data['command'] == 'motoron':


 print("MOTOR ON IS RECEIVED") elif cmd.data['command'] ==

'motoroff':print("MOTOR OFF IS RECEIVED") if cmd.command ==

"setInterval":


else:

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


  interval = cmd.data['interval']

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

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

exceptException as e:
        print("Caught exception connecting device: %s" % str(e))sys.exit()


 # Connect and send a datapoint "hello" with value "world" into the cloud as an event oftype "greeting"
    10 times deviceCli.connect()


while True:
        deviceCli.commandCallback = myCommandCallback


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

SENSOR.PY

```
   import time import
   sysimport ibmiotf.application
   importibmiotf.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))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(0,100)


           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
",
"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":"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":"",
"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,

"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",
"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","a949797028158f3f","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"]]
},
{
"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
","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":"",
"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
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":[]
},
{
"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":[]
},
{

"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":"",
"li bs
":[
],
"x ":
63
0,
"y":500, "wires":[["5c7996d53a445412"]]
},
{
"id":"5c7996d53a445412
",
"type":"httpresponse",
"z":"03acb6ae05a0c712
","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_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://drive.google.com/file/d/1x4ceQg4KliZShGXHOBDjZaGMhwcLRE98/view?usp=share_link