IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID : PNT2022TMID15998

1. INTRODUCTION

1.1 PROJECT OVERVIEW

In this project a centralizing method in the area of IIoT (Industrial Internet of Things) contrived for understanding agriculture which is preceding the arrangements low-power devices. This paper yields a monitoring procedure for farm safety against animal attacks and climate change conditions. IIoT advances are frequently used in smart farming to emphasize the standard of agriculture. It contains types of sensors, controllers. On behalf of WSN, the ARM Cortex-A board which consumes 3W is the foremost essence of the procedure . Different sensors like DHT 11 Humidity & Temperature Sensor, PIR Sensor, LDR sensor, HC-SR04 Ultrasonic Sensor, and camera are mounted on the ARM Cortex-A board. The PIR goes high on noticing the movement within the scope, the camera starts to record, and the data will be reserved on-board and in the IoT cloud, instantaneously information will be generated automatically towards the recorded quantity using a SIM900A unit to notify about the interference with the information of the weather conditions attained by DHt11. If a variance happens, the announcement of the threshold rate will be sent to the cell number or to the website. The result will be generated on a catalog of the mobile of the person to take the necessary action.

1.2 PURPOSE

The Internet is essentially a network of computers that are connected. However, as the world changes, its use is expanding beyond just email and web browsing. The creation of smart homes, smart rural communities, and e-health are all products of today's internet, which also deals with embedded sensors.

The idea of IoT was introduced by care's etc. Without human-to-human or human-to-computer interaction, the Internet of Things refers to the connection or communication between two or more devices. With the use of sensors or actuators, connected devices may sense their environment. Sensing the device, gaining access to the device, processing the device's data, and offering applications and services make up the four main parts of IOT. Along with this, it also offers data security and privacy. All facets of our daily life have been impacted by automation. In order to save time and reduce human effort, more advancements are being made in practically every industry. The same is being 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.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

Solar power generation and rainwater harvesting as technology method is implemented along with crop safety. The moisture contents in the soil is sensed by using the moisture sensor and it will identify the amount of water supply required to the crop and sends data to RFID and enables the sensor to supply water which automatically turn on the water source and turn off it when need is satisfied. To monitor temperature, humidity and moisture in the soil of agricultural land done by using IOT. To improve irrigation, to monitor things, a system to identify and classify affected plants autonomous rover is used. The gathered data from sensing nodes are fed to machine learning algorithm displaying both data and warning message through a graphical user interface.

Crops in farms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. This leads to huge losses for the farmers. It is not possible for farmers to barricade entire fields or stay on field 24 hours and guard it. So here we propose automatic crop protection system from animals. Animal detection system is designed to detect the presence of animal and offer a warning. In this project we used PIR and ultrasonic sensors to detect the movement of the animal and send signal to the controller. It diverts the animal by producing sound and signal further, this signal is transmitted to GSM and which gives an alert to farmers and forest department immediately. There are remote methods that can be used to track and identify animals visually and through acoustic signals. The design system will not be dangerous to animal and human being, and it protects farm.

2.2 REFERENCES

1. J. Padhye, V. Firoiu, andD. Towsley, —A stochastic model of TCP Reno congestion avoidance and control, Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02,1999.

2. Infantian Rubala,D.Anitha.,“Agriculture Field Monitoring using Wireless Sensor Networks to Improving Crop Production” 2017 IEEE International(2017).

3. Hanshi Wang; Jingli Lu;Lizhen Liu; Wei Song;ZhaoxiaWang;“Community Alarm System Design Based On MCU And GSM” Year: 2015.

4. ArturFrankiewicz;RafałCpek.” Smart PassiveInfrared Sensor -Hardware Plat- form “Year: 2013 IECON 2013 -39th Annual Conference of the IEEE Industrial Electronics Society Pages: 7543 – 7547.

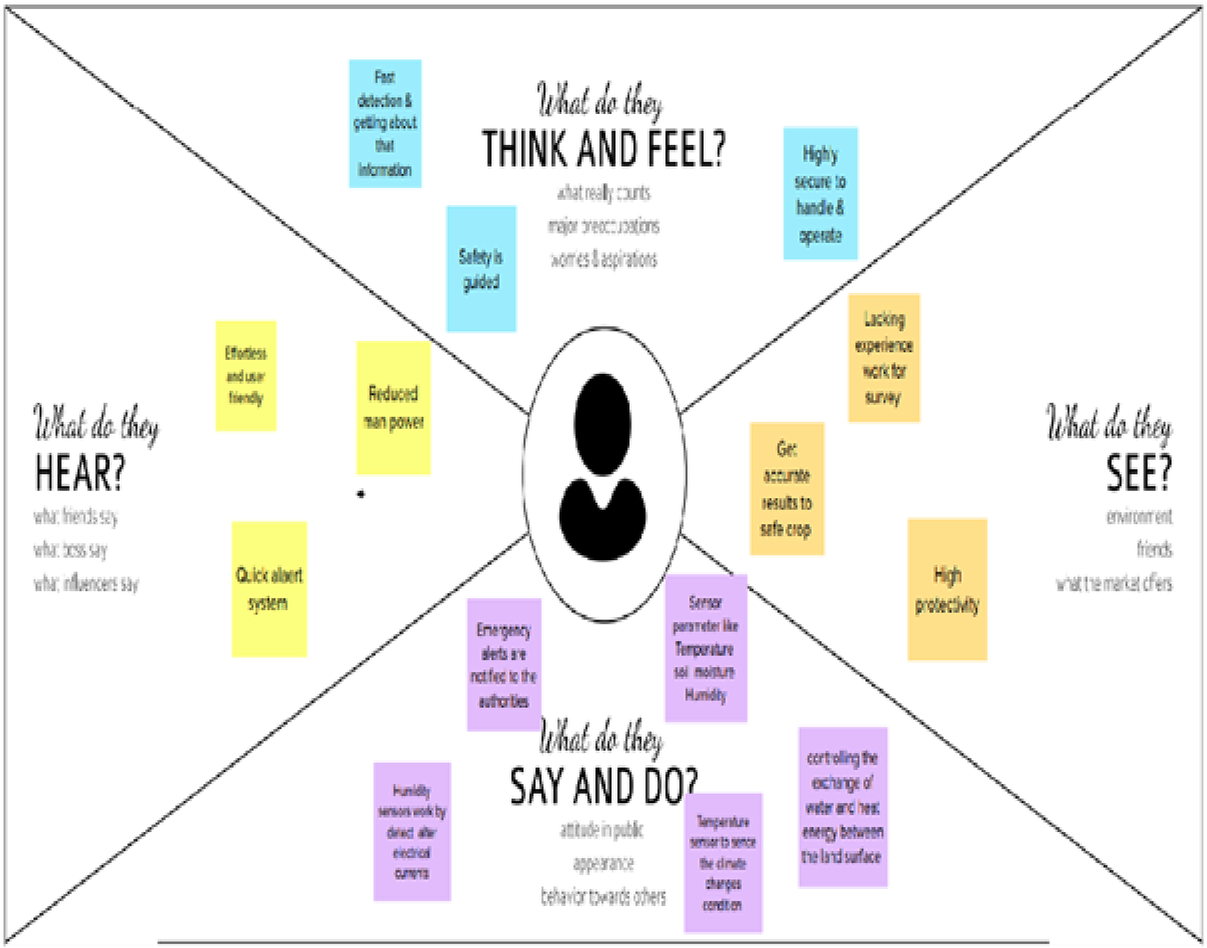
5. Nagur, Nehaparveen Binkadakatti, Pavitra Gokavi, Mouneshwari on Android and IOT Based agriculture system, “International Journal of Recent of engineering and Research 2017”.

2.3 PROBLEM STATEMENT DEFINITION

The project helps farmers get Live Data (Temperature, Humidity, Soil Moisture, and Soil Temperature) for efficient environment monitoring, allowing them to boost overall yield and product quality. It is a high-tech system for bulk crops to be produced sustainably and cleanly. It includes the application of current information and communication technologies in agriculture. The system's primary goal is to improve overall product quality and production. Such solutions can perform tasks that range from seeding and watering to harvesting and sorting. Farming is a laborintensive task that requires lots of time and effort. Usually, these tasks are repetitive and monotonous. Farmers can delegate these labor-intensive tasks to robotics and automation-based solutions. Eventually, this technology integration would result in higher productivity with minimal resource wastage. Robotic Machinery also helps in supporting farm machinery. It is useful for sowing, harvesting, and other services and helps in avoiding human errors. Farms can utilize robotic systems for pesticide spraying, harvesting, cultivating, and other such activities.

3.IDEATION AND PROPOSED SOLUTION

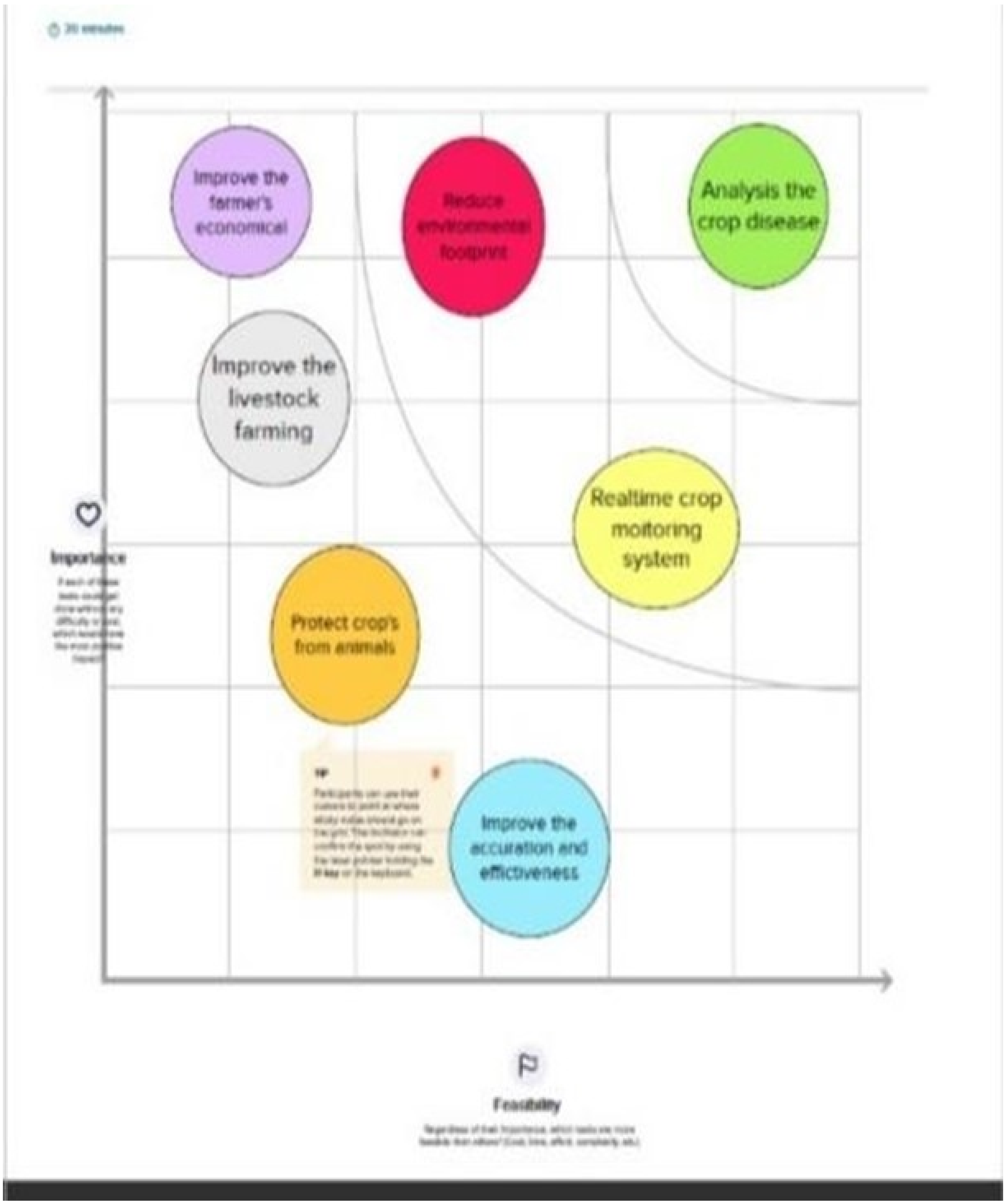
3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

GROUP IDEAS:


Improve yield and effictiveness
Remote monitoring
Solar ennergy can be harnessed to operate our prototype so that it can pave way to conventionalization
Improved livestock farming
Send intimation message to user for animal activities


Improve the farmer's economical
Reduce environmental footprint
Analysis the crop disease
Improve the livestock farming
Realtime crop moitoring system
Importance
Protect crop's from animals
Improve the accuration and effictiveness
Feasibility

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Develop an efficient system & an application that can monitor and alert the users(farmers)
2.	Idea/Solution description	1. This product helps the field in monitoring the animals other disturbance 2. In several areas, the temperature sensors will be integrated to monitor the temperature & humidity 3. If in any area feel dry orwetless is detected by admins, will be notified along with the location in the web application
3.	Novelty/Uniqueness	1. Fastest alertsto the farmers 2. The increasing demand for quality food 3. User friendly
4.	Social Impact/Customer Satisfaction	1. Easy installation and provide efficient results 2. Can workwith irrespective of fear
5.	Business Model(Revenue Model)	1. As the product usage can be understood by everyone, it is easy for them to use it properly fortheir safest organization 2. The product is advertised all over the platforms. Sinceit is economical, even helps small scale farming land from disasters.
6.	Scalability of the Solution	1. Even when the interruption is more, the product sensethe accurate location and alerts the farmers effectively

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) CS

Farmer's ! Who's not near his

field

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL

1)High adoption costs , security concerns. 2)Not aware of the implementation of IoT inagriculture.

5. AVAILABLE SOLUTIONS PLUSES & MINUSES AS

Monitor different parameters and mobile or web application make easily to farm thecrop field .

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY PR

- It's difficult to monitor and

  control
- Ain't known if the applicationdoesn't work properly.

9. PROBLEM ROOT / CAUSE RC

If temperature ,PH level ,humidity & light intensity makes the serious cause for the environment.

Farmer affected by less productivity which will affect in their profit.

7. BEHAVIOR + ITS INTENSITY BE

Direct related: Tries to find a solution to prevent this problem

Indirect related: Located in rural where internet connectivity might not be strong enough to facilitate fast transmission speeds.

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT TR

Create opportunities to lift people out of poverty in developing nations. (Over60%)

4. EMOTIONS BEFORE / AFTER EM

BEFORE: Finances, Heavy work overload andconflict in relationship.

AFTER: It will easier to make more yield in

10. YOUR SOLUTION SL

"IoT based Smart crop protection systemfor agriculture" !!

It help farmers grow more food on lessland by protection crops from pests, diseases and weeds as well as raising productivity per hectare.

8. CHANNELS of BEHAVIOR CH

ONLINE: The Data send through applicationfor the farmers to know about the farms.

OFFLINE: The control action is taken bythe farmers to monitor the farms.

Extract online & offline CH of BE

4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

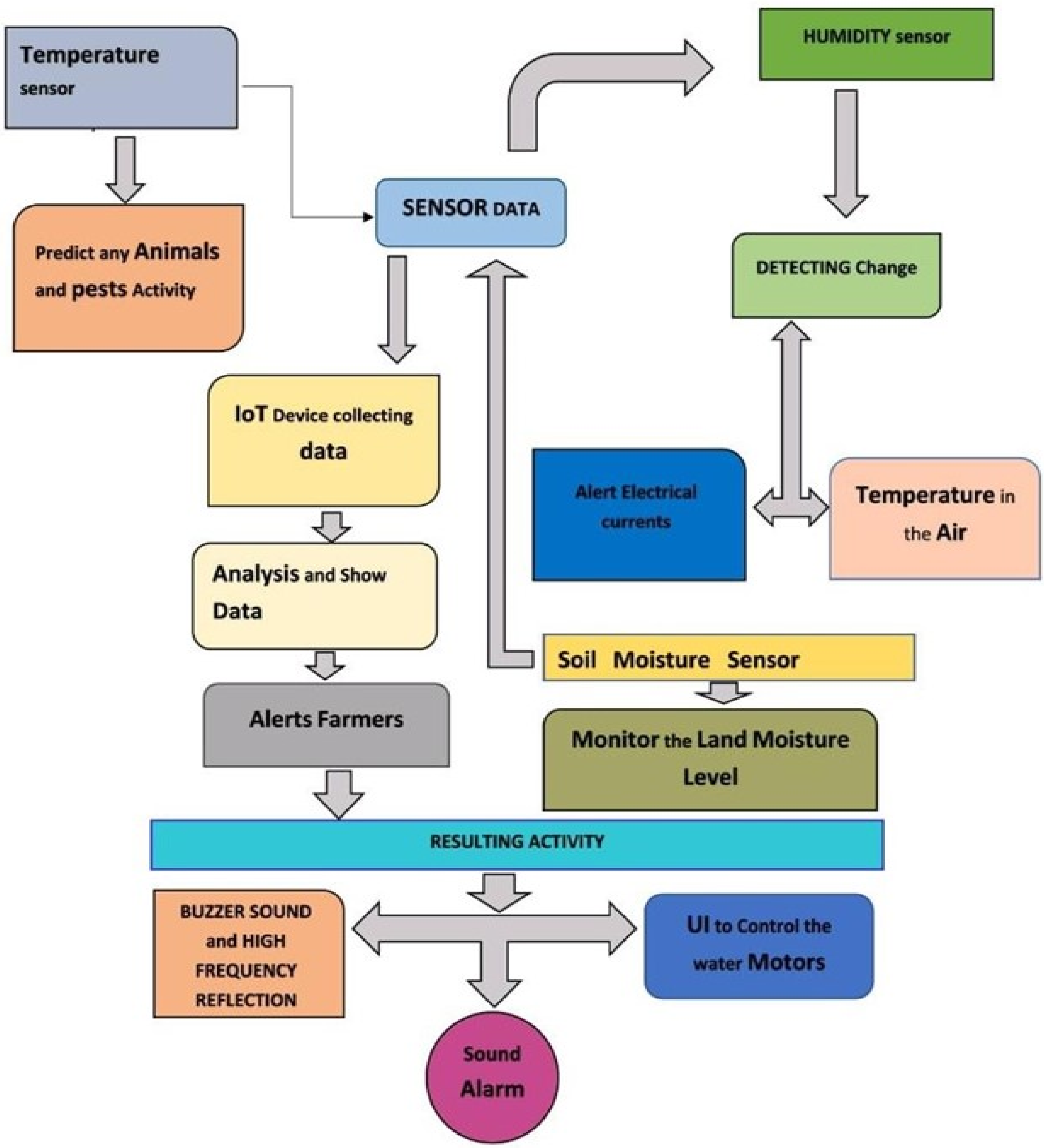
FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sense animals nearing the crop field and sounds alarm to woo them away as well as sendsSMS to farmer using cloud service.
FR-2	User Reception	The Data like values of Temperature, Humidity, Soil moisture sensorsare receivedviaSMS
FR-3	User Understanding	Based on the sensor data value to get the information aboutpresent of farmingland
FR-4	User Action	The user needs take action like destructionof crop residues, deep plowing, crop rotation, fertilizers, strip cropping, scheduled planting operations.

4.2 NON – FUNCTIONAL REQUIREMENTS

FR No	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	**Mobile support. Users must be able to interact in the same roles & tasks on computers & mobile devices wherepractical, given mobilecapabilities.**
NFR-2	**Security**	**Data requires secure access to must register and communicate securely on devices and authorized users of the system who exchange informationmust be able to do.**
NFR-3	**Reliability**	**It has a capacity to recognize the disturbance nearthe field and doesn'tgive a falsecaution signal.**
NFR-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 mustbe supported. This requirement is related to the requirement to support industrial and device protocols at the edge.**
NFR-5	**Availability**	**IoT solutions and domains demand highly availablesystems for 24x7 operations. Isn't a *critical production* application, which means that operations or production don't go down if the IoT solution is down.**
NFR-6	**Scalability**	**System must handle expanding load and data retention needs that are based on the upscaling of the solution scope, such as extra manufacturing facilities andextra buildings.**

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

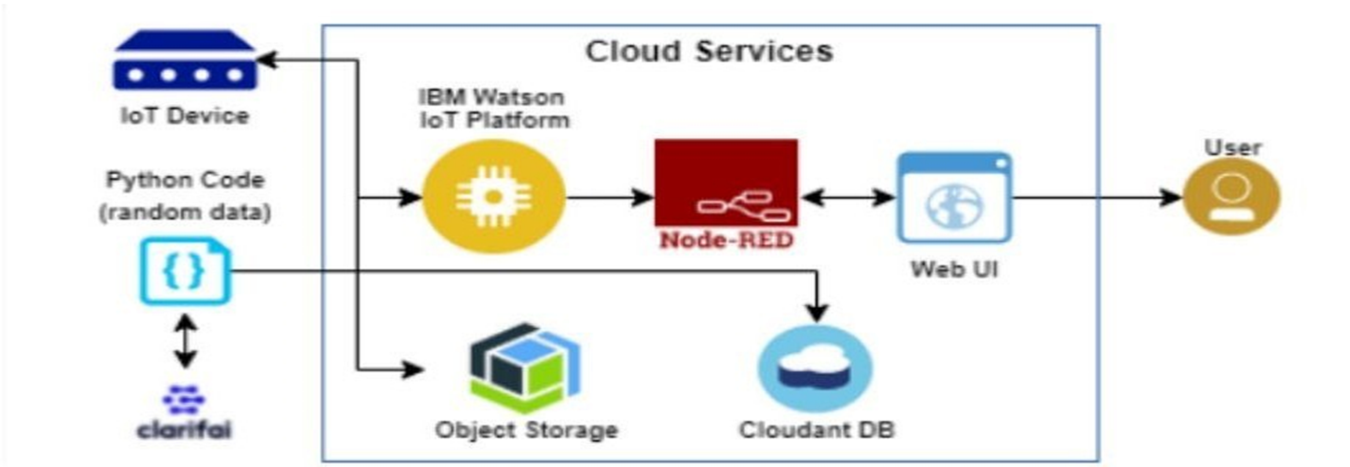


TABLE-1:

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

<u>TABLE-2:</u> 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

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

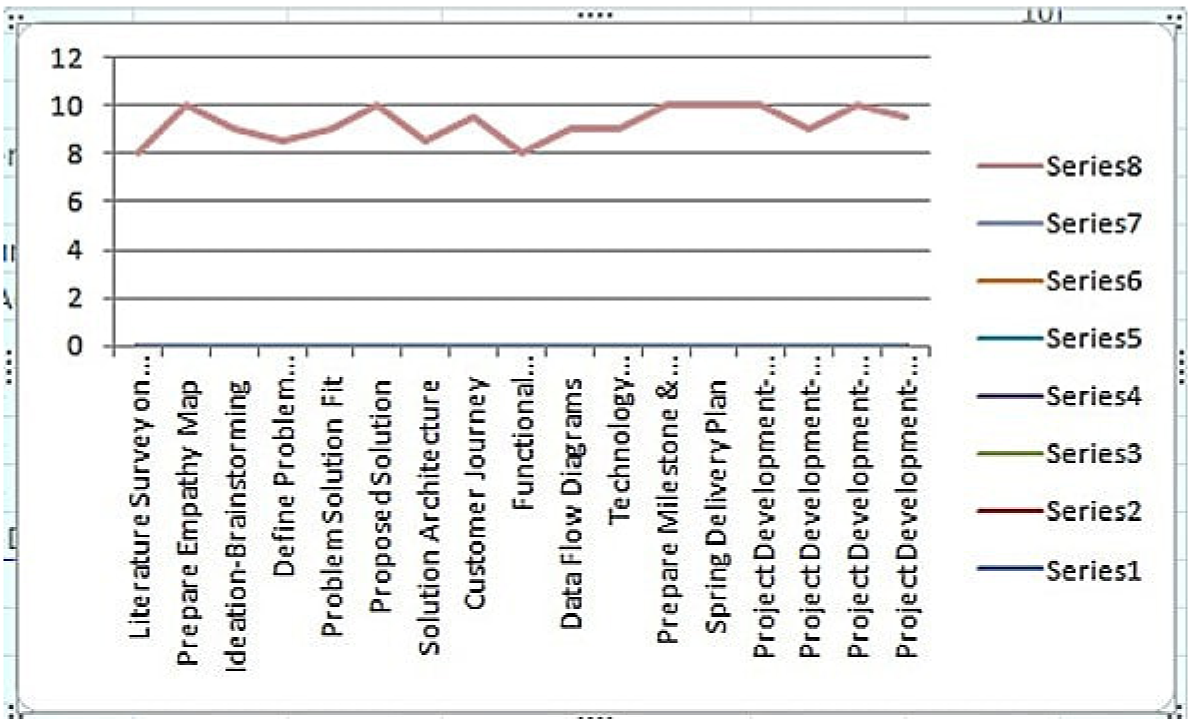
SPRINT PLANNINGAND ESTIMATION:

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)

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



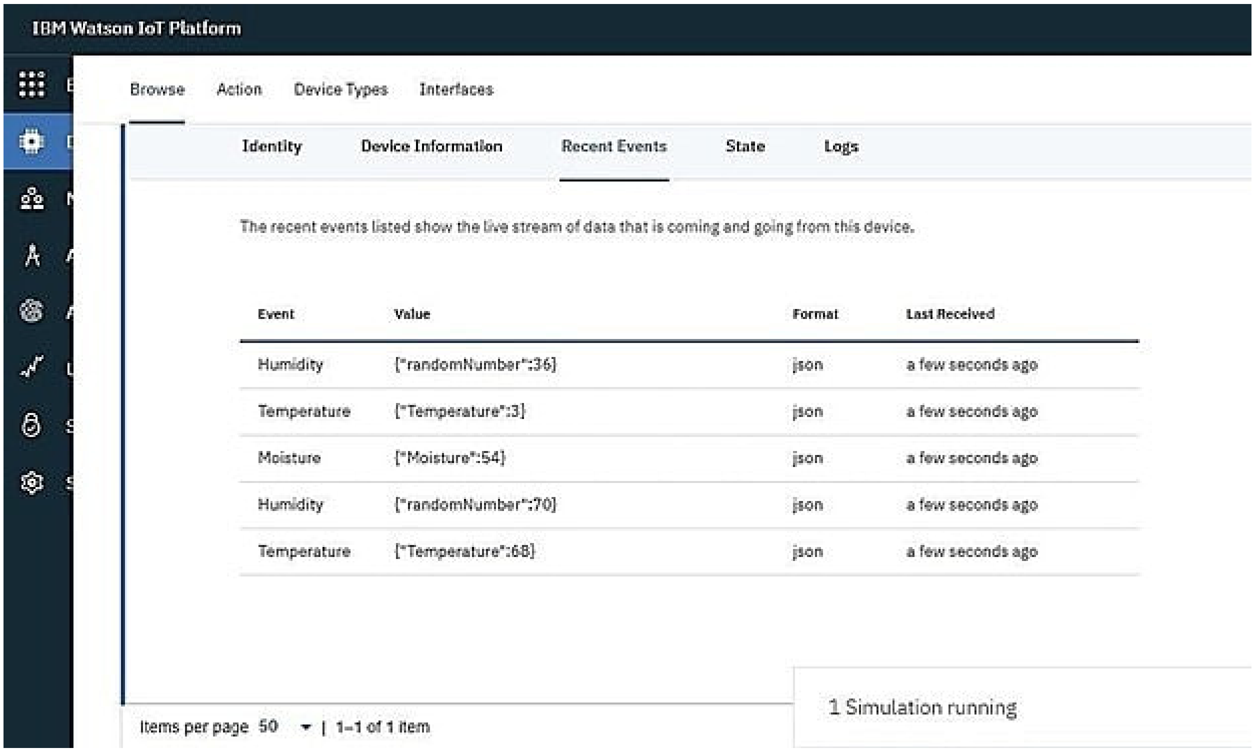
CODING AND SOLUTIONING

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 sensordata to send in json

   format to cloud. temp_data= {

   'Temperature' : temp_sensor }

   PH_data = { 'PH Level' : PH_sensor }
   camera_data = { 'Animalattack' :
   camera_reading}flame_data = {
   'Flame' : flame_reading }
```

```
moist_data = { 'Moisture Level' :
moist_level} water_data = {
'WaterLevel' : water_level}

# publishing Sensor data to IBM Watsonfor 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, "toIBM Watson")

success = deviceCli.publishEvent("PH sensor", "json",
PH_data, qos=0)sleep(1)
if success:
   print ("Published PH Level = %s" % PH_sensor, "toIBM 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, "toIBM Watson")

success = deviceCli.publishEvent("Moisture sensor", "json",
moist_data, qos=0)sleep(1)
if success:
    print ("Published MoistureLevel = %s " % moist_level, "toIBM Watson")
success = deviceCli.publishEvent("Water sensor", "json",
water_data, qos=0)sleep(1)
if success:
   print ("Published Water Level = %s cm"% water_level,
"toIBM Watson") print ("")
#Automation to control sprinklers by present temperature an to send alert messageto 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("spri
nkler-1 is
OFF")
print("")
#To send alert messageif farmer uses the

unsafefertilizer 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,useother fertilizer" %PH_sensor } ,
qos=0) sleep(1)
 if success:
   print('Published alert2 : ' , "Fertilizer PH level(%s) is not safe,useother fertilizer" %PH_sensor,"to IBM
 Watson") print("")
 #To send alert messageto farmer that animal

 attackon crops. if (camera_reading ==

 "Detected"):

   success = deviceCli.publishEvent("Alert3", "json", { 'alert3' : "Animal attackon crops
 detected" }, qos=0) sleep(1)
 if success:
   print('Published alert3: ' , "Animal attackon crops detected","to IBM Watson","to IBM
 Watson") print("")
 #To send alert messageif flame detectedon crop land and turn ON the splinkers to take immediateaction.

 if (flame_reading == "Detected"):
   print("sprinkler-2 is ON")
 success = deviceCli.publishEvent("Alert4", "json", { 'alert4' : "Flame is detected crops are in danger,sprinklers
 turnedON" }, qos=0)sleep(1)
 if success:
   print( 'Published alert4: ' , "Flame is detected cropsare in danger,sprinklers turned ON","to IBM Watson")


 #To send alert messageif Moisture levelis LOW and to Turn ON Motor-
 1for 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 messageif Water level is HIGH and to Turn ON Motor-2 to take water out.
 if (water_level > 20):
   print("Motor-2is 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 deviceand application
from the clouddeviceCli.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 3.3Vregulator),but 5V is ideal in case the regulator has different specs.

BUZZER

Specifications

1. RatedVoltage : 6V DC
2. Operating Voltage: 4 to 8V DC
3. Rated Current*:≤30mA
4. SoundOutput at 10cm* : ≥85dB
5. Resonant Frequency: 2300 ±300Hz
6. Tone: Continuous A buzzeris a loud noise maker.

Most modern ones are civil defense or air- raid sirens, tornadosirens, or the sirens on emergency servicevehiclessuch 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, Propaneand Hydrogen .

iii. Longlife and low cost.

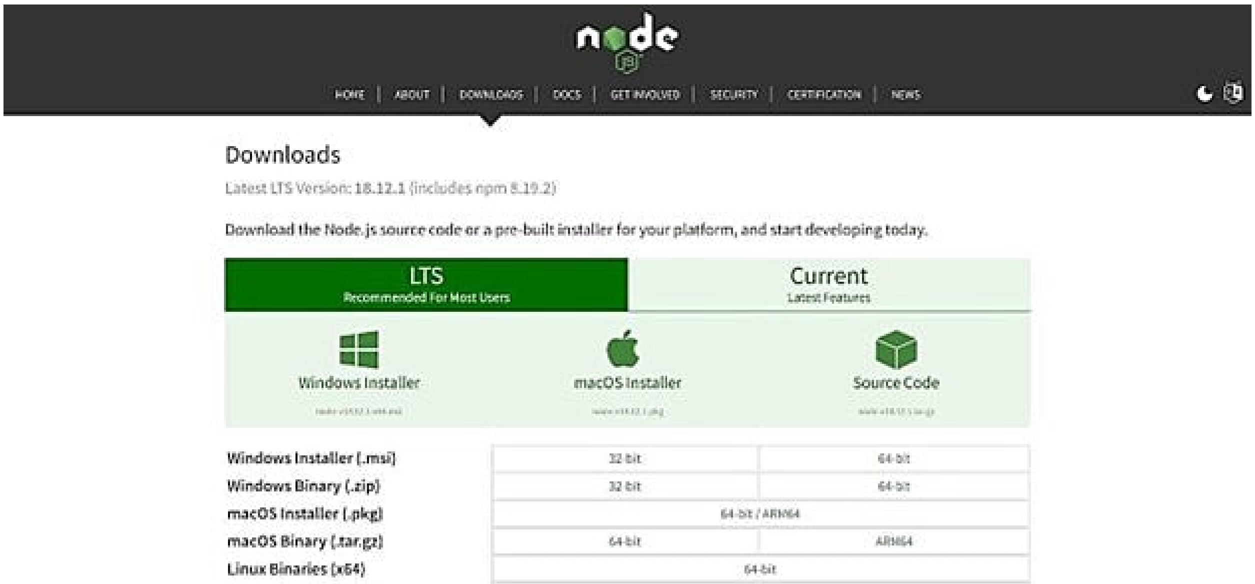
iv. Simpledrive circuit.

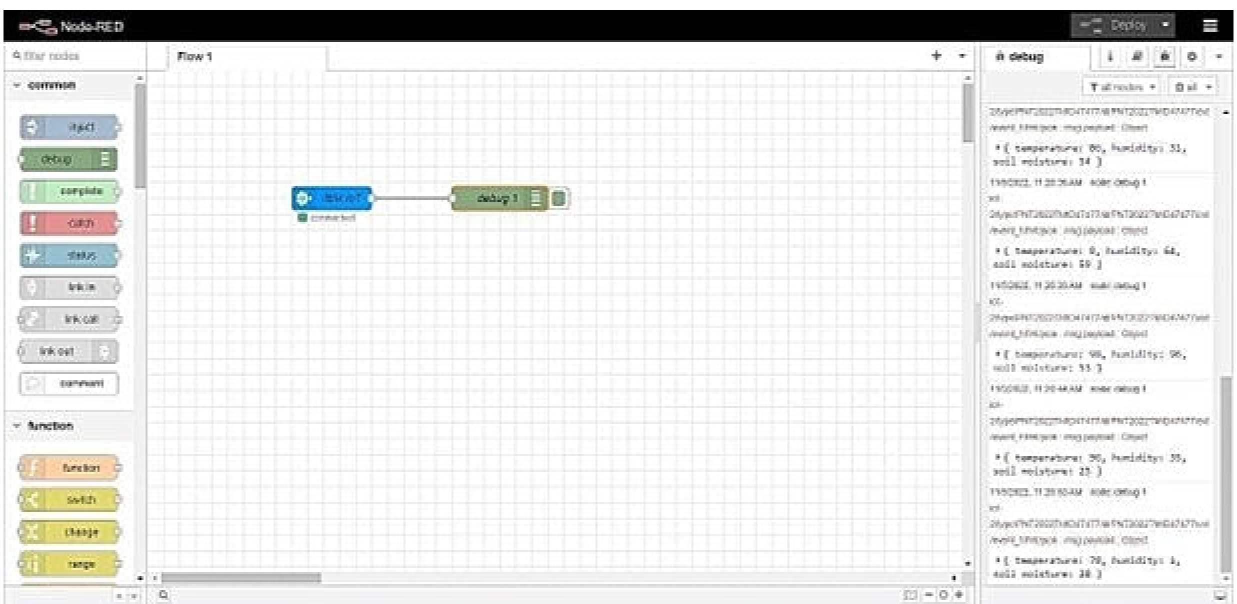
TESTING

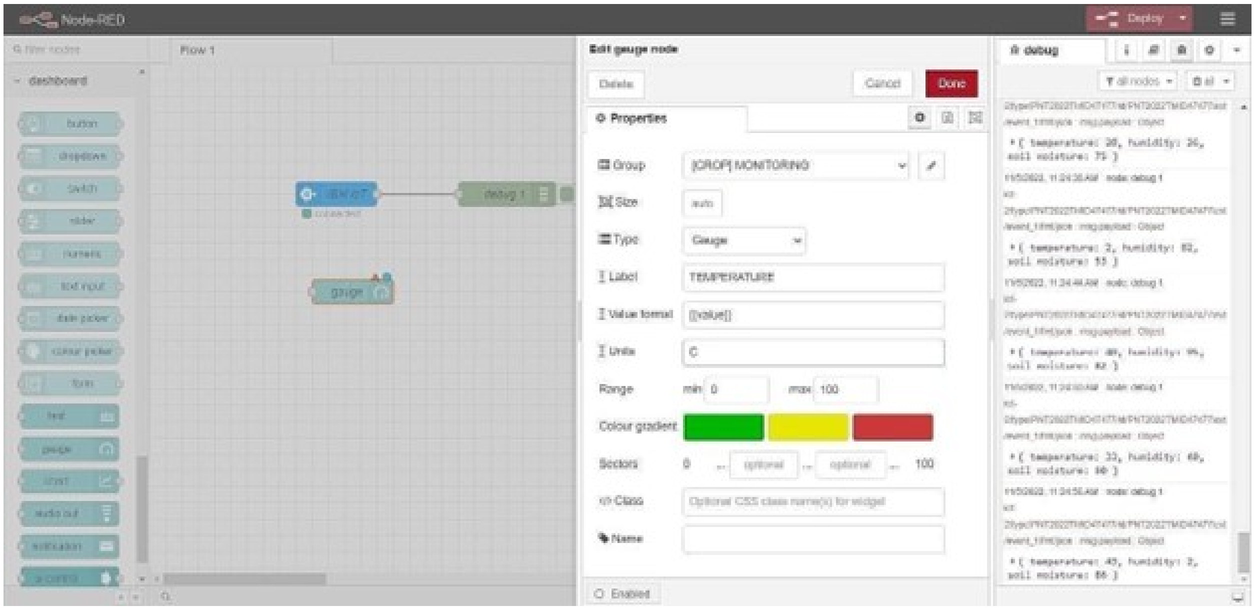
TEST CASES:

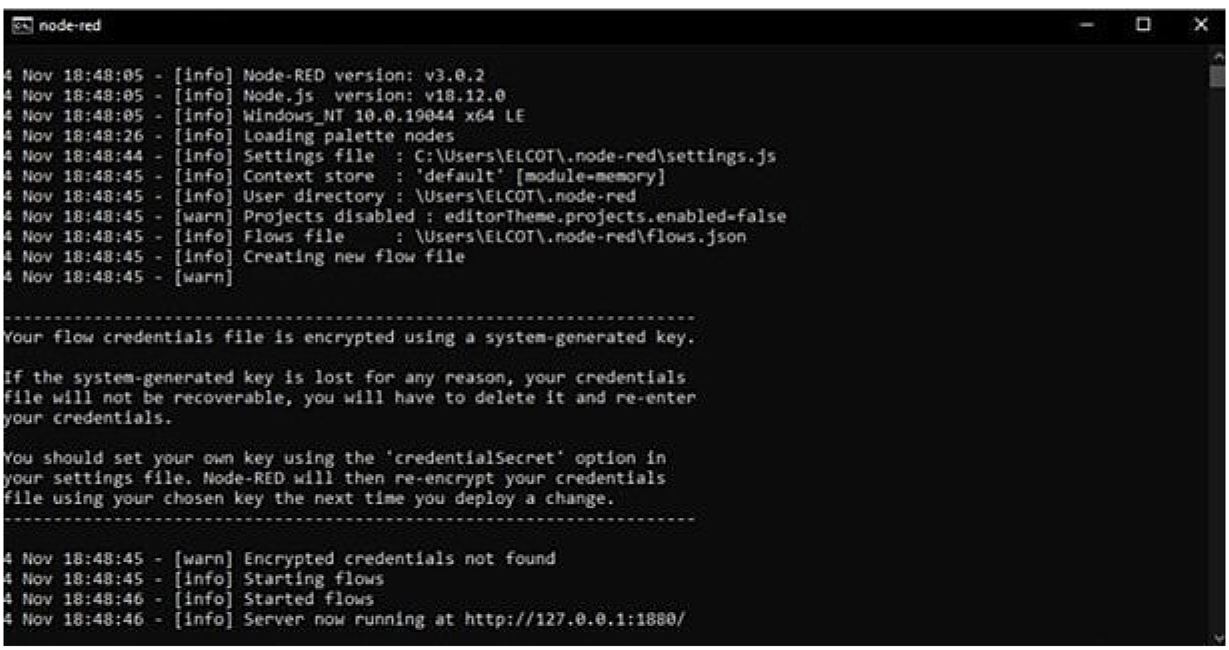
S.no	parameter	Values	Screenshot
1	Model summary	-	
2	accuracy	Training accuracy-95% Validation accuracy-72%	
3	Confidencescore	Class detected-80% Confidence score-80%	

User Acceptance Testing:






Node-RED
Deploy
Flow 1
dashboard
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
debug 1
gauge
Edit gauge node
Delete
Cancel
Done
Properties
Group
[CROP] MONITORING
Size
auto
Type
Gauge
Label
TEMPERATURE
Value format
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


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 urgentattention as no effective solutionexists till date for this problem. Thus this project carries a greatsocial 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 willsave 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 well being.

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 minimumresources 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 WatsonIoT 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 animal and fire can be detected by 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 sensing this laser or sensor'ssecurity system will be activated.

APPENDIX

SOURCE CODE

```
import time importsys import ibmiotf.application # toinstallpipinstall 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 DeviceID authMethod = "token"
authToken = "LWVpQPaVQ166HWN48f" # Replace the authtoken


def myCommandCallback(cmd):  # functionfor 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 - commandis missing requiredinformation: 'interval'")

    interval = cmd.data['interval']elifcmd.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 exceptionconnecting 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 clouddeviceCli.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 DeviceID 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}
#printdata                    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 connectedto
IoTF")time.sleep(1)
```

deviceCli.commandCallback = myCommandCallback

Disconnect the device and application from the clouddeviceCli.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":"registered","x":680, "y":220,
"wires":[]

},

{

"id":"4cff18c3274cccc4","type":"ui_button",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00eed6c","order":2,
"width":"0",
"height":"0",

"passth
ru":fals
e,
"label":"
MotorO
N",
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"m
otoron\"}",
"payloadType":"str",
"topic":"motoron",
"to
pic
Ty
pe
":
"s
tr"
,"x
":3
60,
"y":160, "wires":[["625574ead9839b34"]]},
{
"id":"659589baceb4e0b0",
"type":"ui_button",
"z":"630c8601c5ac3295",
"name":"",
"group":"716e956.00ee
d6c","order":3,
"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":"MotorOFF",
"tooltip":"",
"color":"",
"bgcolor":"",
"className":"",
"icon":"",
"payload":"{\"command\":\"motoroff\"}",
"payloadType":"str",
"topic":"motoroff",
"topicType":"str","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":false},
{"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":"f
unction",
"z":"03acb6ae05a0c712",
"name":"Humidity",
"func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn msg;",
"outputs":1,
"noerr":
0,
"in
itia
lize
":"
",
"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":"",
"classNa
me
":"","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","#e6e
600","#ca3838"],"seg1":"",
"seg2":"",

"className
":"",
"x":790,

"y":120,

"wires":[]

},

{

"id":"a259673baf5f0f98
","type":"httpin",
"z":"03acb6ae05a0c712
","name":"",
"url":"/sensor",

"method":"get",

"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 Link

https://github.com/IBM-EPBL/IBM-Project-15522-1659599834

ProjectDemo Link

https://drive.google.com/file/d/14Fa4ii_3OcULe6BDdg-tqHp648vtYqth/view?usp=share_link