

# **IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE**

**TEAM ID: PNT2022TMID21791**

**TEAM MEMBERS:**

**S.LOGAPRIYA**

**E.MADHUMITHA**

**V.MALINI**

**N.MONIKA**

- 1. INTRODUCTION**
  - 1.1 Project Overview
  - 1.2 Purpose
- 2. LITERATURE SURVEY**
  - 2.1 Existing problem
  - 2.2 References
  - 2.3 Problem Statement Definition
- 3. IDEATION & PROPOSED SOLUTION**
  - 3.1 Empathy Map Canvas
  - 3.2 Ideation & Brainstorming
  - 3.3 Proposed Solution
  - 3.4 Problem Solution fit
- 4. REQUIREMENT ANALYSIS**
  - 4.1 Functional requirement
  - 4.2 Non-Functional requirements
- 5. PROJECT DESIGN**
  - 5.1 Data Flow Diagrams
  - 5.2 Solution & Technical Architecture
  - 5.3 User Stories
- 6. PROJECT PLANNING & SCHEDULING**
  - 6.1 Sprint Planning & Estimation
  - 6.2 Sprint Delivery Schedule
  - 6.3 Reports from JIRA
- 7. CODING & SOLUTIONING**
  - 7.1 Feature 1
  - 7.2 Feature 2
  - 7.3 Database Schema (if Applicable)
- 8. TESTING**
  - 8.1 Test Cases
  - 8.2 User Acceptance Testing
- 9. RESULTS**
  - 9.1 Performance Metrics
- 10. ADVANTAGES & DISADVANTAGES**
- 11. CONCLUSION**
- 12. FUTURE SCOPE**
- 13. APPENDIX**

**Source Code**

**GitHub & Project Demo Link**

# **1. INTRODUCTION:**

## **1.1 PROJECT OVERVIEW:**

Crops in farms are frequently devastated by local animals such as buffaloes, cows, goats, birds, and so on. This results in massive losses for the farmer. Deforestation happens as a result of overpopulation, resulting in a lack of food, water, and shelter in forest areas. As a result, animal interference in residential areas is increasing, threatening human life and property, leading to human-animal conflict. However, according to nature's rules, every living creature on this planet plays an important role in the eco-system. Elephants and other animals that come into touch with humans have a negative impact in a variety of ways, including agricultural depredation, damage to grain stores, water supply, dwellings and other assets, and human injury and death.

## **1.2 PURPOSE:**

Electric Fences:

Electric fences were used to control livestock in the United States in the early 1930s, and electric fencing technology developed in both the United States and New Zealand. An early application of the electric fence for livestock control was developed in 1936–1937 by New Zealand inventor Bill Gallagher. One of the major disadvantages of having an electric fence installed is that it requires regular maintenance. Scarecrow:

Scarecrow genealogy is rooted in a rural life style. The Egyptians used the first scarecrows in recorded history to use to protect wheat fields along the Nile River from flocks of quail. Egyptian farmers installed wooden frames in their fields and covered them with nets. While traditional, motionless scarecrows do work against “pest birds” (e.g., crows and blackbirds), the effect is almost always temporary. Over time, the birds get used to stationary dummies and resume their destructive habits.

# **2. LITERATURE SURVEY:**

## **2.1 EXISTING PROBLEM:**

One of the major economic issues faced by the country is agriculture as this is the sector which is source of livelihood for about 54% of Indians till date. Still today this sector is not well developed and faces lots of problems resulting into low productivity of crops. As 43% of land in India, is used for farming but contributes only 18% of the nation's GDP. The poor condition of agriculture in the country is the point of concern for Indians. The rural farmers in India suffer from poverty and most of them are illiterate so there is lack of good extension services. The problem of wild life attack on crops i.e., crop Vandalization is becoming very common in the states of Tamil Nadu, Himachal Pradesh, Punjab, Haryana, Kerala and many other states. Wild animals like monkeys, elephants, wild pigs, deer, wild dogs, bison, nilgais, estray animals like cows and buffaloes and even birds like parakeets cause a lot of damage to crops by running

over them, eating and completely vandalizing them. This lead to poor yield of crops and significant financial loss to the owners of the farmland. This problem is so pronounced that sometimes the farmers decide to leave the areas barren due to such frequent animal attacks Another major problem faced by Indian farmer is their dependency on nature and poorly maintained irrigation system. Current agricultural practice are neither economically nor environmentally sustainable and India's yields for many agricultural commodities are low. Poorly maintained irrigation system and almost universal lack of good extension service are among the factor responsible. Poor roads to market from village, rudimentary market infrastructure, and excessive regulation are few of the other concerned points for the agriculture sector in India. The low productivity in India is a result of the following factor:

## 2.2 REFERENCES:

- [1] ArturFrankiewicz; RafałCupek.” Smart Passive Infrared Sensor - Hardware Platform “Year: 2013 IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society Pages: 7543 – 7547
- [2] Discant, A. Rogozan, C. Rusu and A. Bensrhair, “Sensors for Obstacle Detection” 2007 30th International Spring Seminar on Electronics Technology (ISSE), Cluj-Napoca, 2007, pp. 100-105. doi: 10.1109/ISSE.2007.4432828 Volume:01 Pages:859-862, DOI:10.1109/ICCSNT.2015.7490876, IEEE Conference Publications.
- [3] Mustapha, Baharuddin, AladinZayegh, and Rezaul K. Begg. “Ultrasonic And Infrared Sensors Performance in A Wireless Obstacle Detection System” Artificial Intelligence, Modelling and Simulation (AIMS), 2013 1st International Conference on. IEEE, 2013.
- [4] Padmashree S. Dhake, Sumedha S. Borde, “Embedded Surveillance System Using PIR Sensor”, International Journal of Advanced Technology in Engineering and Science, www.ijates.com Volume No.02, Issue No. 03, March 2014
- [5] Wang, Z., Wang, H., Liu, L., Song, W., & Lu, J. (2015, December). Community alarm system design based on MCU and GSM. In 2015 4th International Conference on Computer Science and Network Technology (ICCSNT) (Vol. 1, pp. 859-862). IEEE.
- [6] Shende, P. Y., Raut, S. M., Ingale, P. S., Nagose, A. K., Katakpur, P. S., & Kathane, S. S. Solar Electric Fencing for Irrigation of Animal Man Conflict
- [7] Mohammad, T. (2009). Using ultrasonic and infrared sensors for distance measurement. World Academy of Science, Engineering and Technology, 51, 293-299
- [8]Volume:01 Pages:859-862, DOI:10.1109/ICCSNT.2015.7490876 , IEEE Conference Publications.

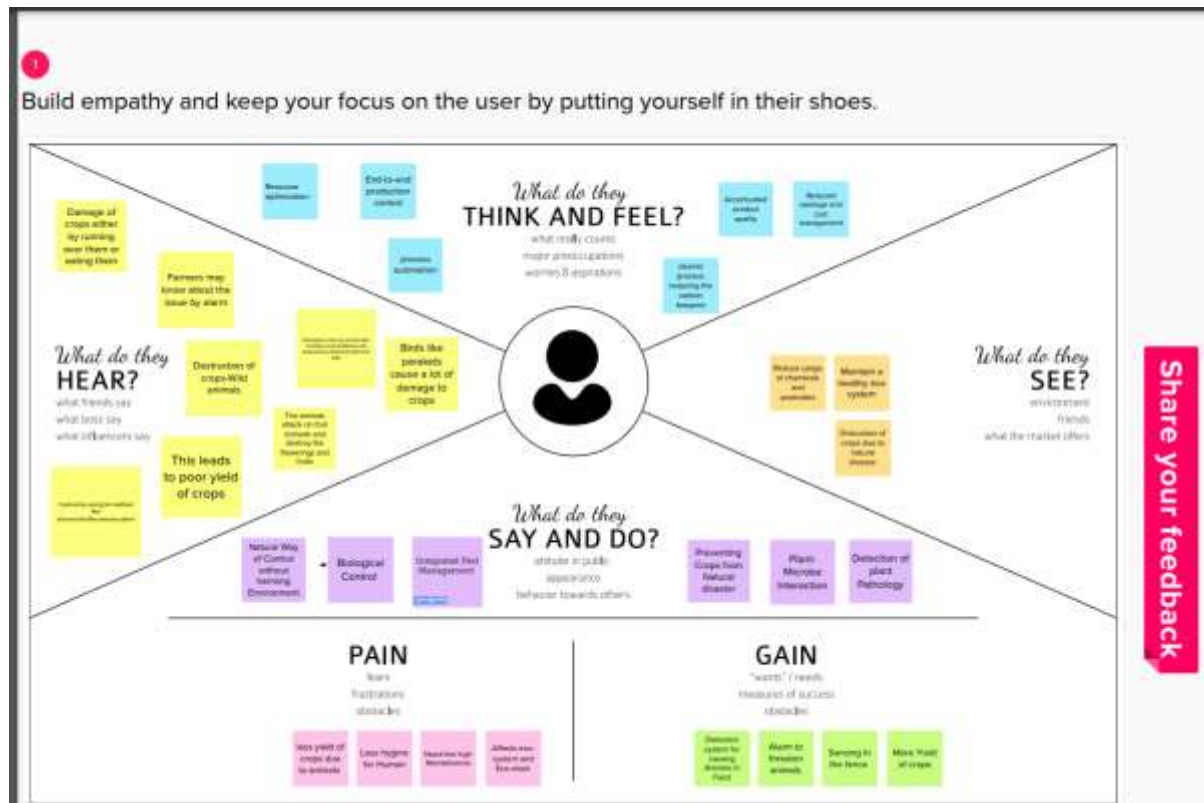
## 2.3 PROBLEM STATEMENT DEFINITION:

<b>Problem Statement (PS)</b>	<b>I am (Customer)</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
<p>PS-1 Develop a system for predicting cotton crop production and probable pest, disease, and insect attacks (before at least 15 days and more).</p> <p>(Technology Bucket: Satellite images, Cloud computing, Big Data, In-field sensor data, drone imagery,</p>	Cost effective, small size	To create an app-based forecasting system that can forecast potential pest, disease, and insect attacks on cotton crops.	More cost	Involves more requirements	Instead of using general recommendations, forecast the cotton crop yield production for farmers in the Vidarbha region using farm historical data, local terrain, weather scenarios, and numerous sensor inputs

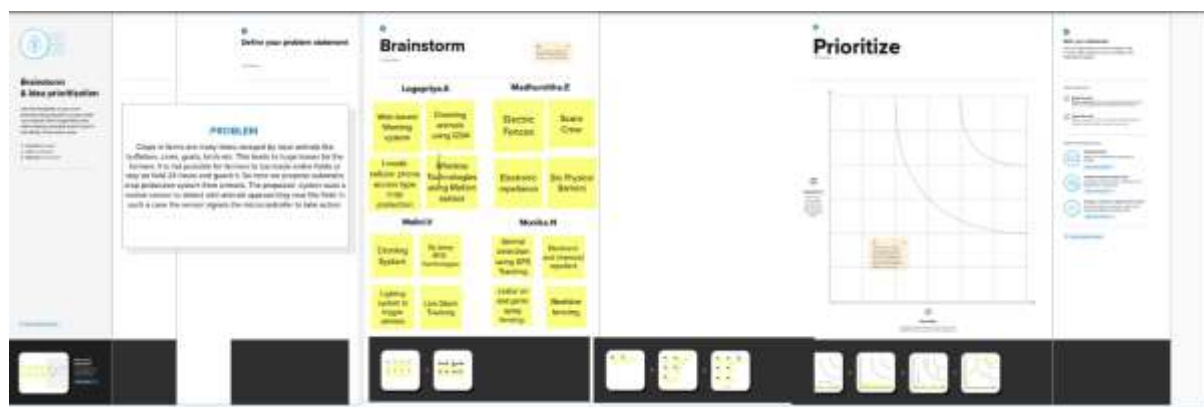
and other IoT data					
PS-2 Wild animals like elephant, wild pig, deer, wild dogs, bison may enter into the field which in turn destroy the crop and reduce the yield in farming	Eco-friendly, customer satisfaction	1. Identify and evaluate risk possessed by wild and domestic animals. 2. Monitor and document animal activity on the farm. 3. Conduct field assessment before harvest	Implementation is harder	Coverage area is larger	Crop protection needs particularly cautious approach

### 3. IDEATION & PROPOSED SOLUTION:

#### 3.1 EMPATHY MAP CANVAS:



#### 3.2 IDEATION AND BRAINSTORMING:

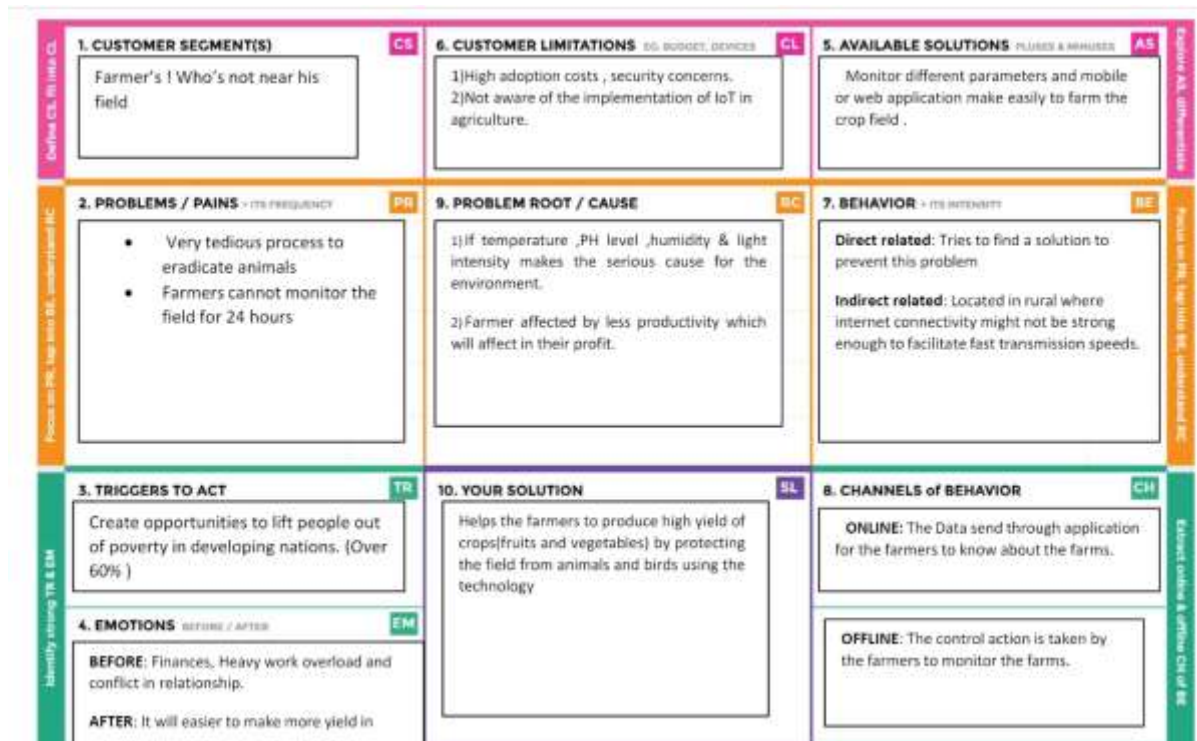


### 3.3 PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>Crop damage cost by animal and bird reduces the crop yield. Crops are mainly damaged by the local animals like buffaloes, cows, pigs, goats.</p> <p>Wild animals are the special challenge for the farmers throughout the world. Farmers with large area of agricultural land find it very tedious to irrigate their land.</p>
2.	Idea / Solution description	<p>An animal detection system to detect the presence of animals and it offers a warning and diverts the animal without any harm.</p> <p>The system will continuously check for any animal to enter the field.</p> <p>Sensors are used to detect animal movement and to give signal to the controller.</p> <p>Further the animals are being diverted by generating sound and signals.</p>
3.	Novelty / Uniqueness	<p>The signal is being transmitted to the GSM (Global System for Mobiles) and instantly give farmers warning.</p> <p>The difficulty and the availability to spot just in-case the animals do not show off by alarm</p>
4.	Social Impact / Customer Satisfaction	<p>The complete safety of crops was ensured by the system from animals thus protecting the animal farmers loss.</p>
5.	Business Model (Revenue Model)	<p>Gives raise to high yield of production.</p> <p>Farmers need not to stay on the field for 24 hours and protect it.</p>
6.	Scalability of the Solution	<p>This technology can be widen based on the area of the field that is used for protection.</p>



### 3.4 PROBLEM SOLUTION FIT:



## 4. REQUIREMENT ANALYSIS:

### 4.1 FUNCTIONAL REQUIREMENT:

Following are the functional requirements of the proposed solution.

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sensen animals nearing the crop field and sounds alarm to woo them away as well assends SMS to farmer using cloud service.
FR-2	User Reception	The Data like values of Temperature, Humidity, Soil moisture sensors are received via SMS
FR-3	User Understanding	Based on the sensor data value to get the information about present of farming land
FR-4	User Action	The user needs take action like destruction of crop residues, deep plowing, crop rotation, fertilizers, strip cropping, scheduled planting

		operations.
--	--	-------------

## 4.2 NON-FUNCTIONAL REQUIREMENTS:

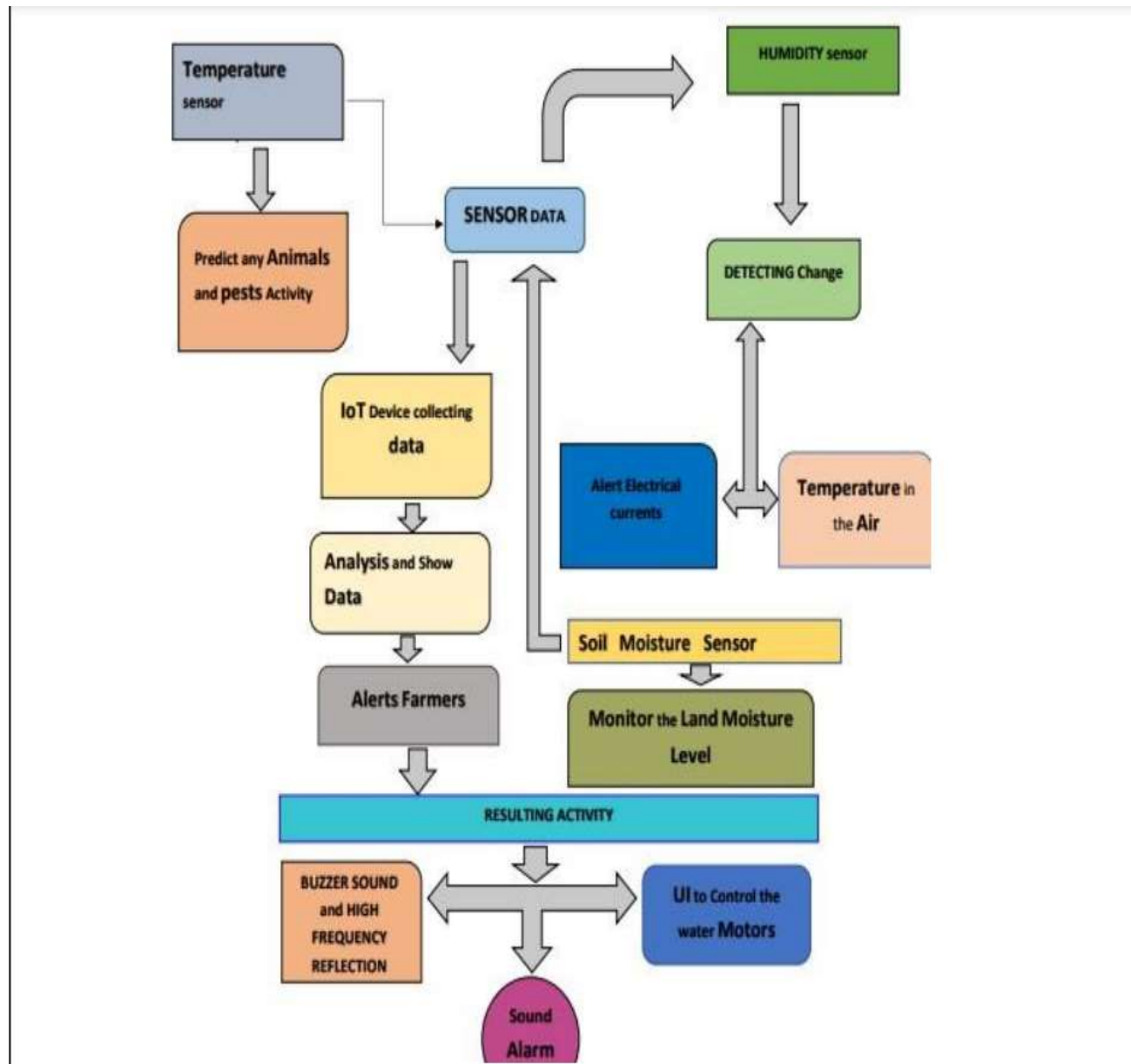
Following are the non-functional requirements of the proposed solution.

FR No	Non-Functional Requirement	Description
NFR-1	Usability	Mobile support. Users must be able to interact in the same roles & tasks on computers & mobile devices where practical, given mobile capabilities.
NFR-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.

NFR-3	Reliability	It has a capacity to recognize the disturbance near the field and doesn't give a false caution 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 must be 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 available systems for 24x7 operations. Isn't a <i>critical production</i> 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 and extra buildings.

## 5. PROJECT DESIGN:

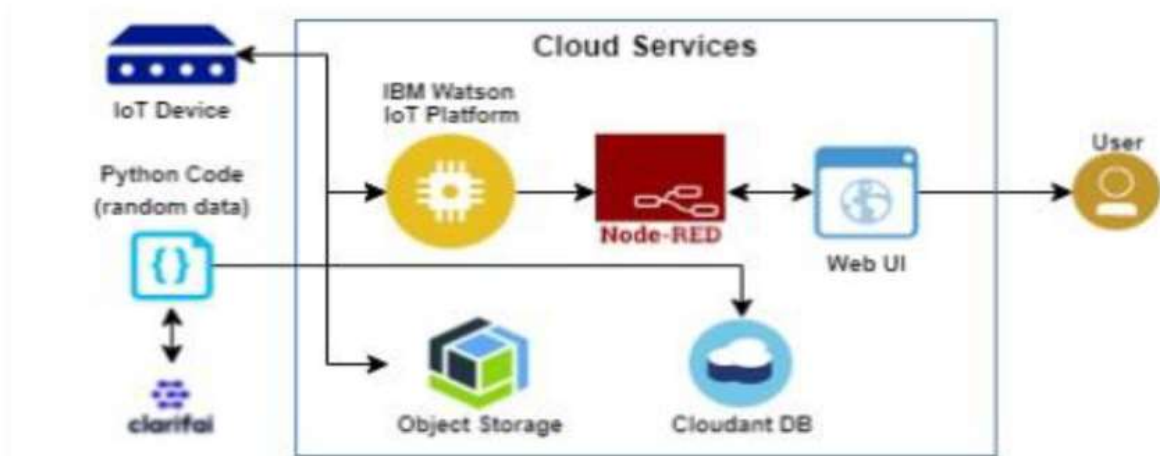
### 5.1 DATA FLOW DIAGRAM:



### 5.2 SOLUTION & TECHNICAL ARCHITECTURE:

#### TECHNICAL ARCHITECTURE:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table2.



## COMPONENTS & TECHNOLOGIES:

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

## 5.3 USER STORIES:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1		US-1	Create the IBM Cloud services which are being used in this project.	6	High	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.

Sprint-1		US-2	Configure the IBM Cloud services which are being used in completing this project.	4	Medium	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.
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	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.
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.	5	High	N.Monika V.Malini S.Logapriya, E.Madhumitha,
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	N.Monika, V.Malini. S.Logapriya, E.Madhumitha,

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		US-2	Create a Node-RED service.	10	High	V.Malini S.Logapriya
Sprint-3		US-1	Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform	7	High	E.Madhumitha S.Logapriya
Sprint-3		US-2	After developing python code, commands are received just print the statements which represent the control of the devices.	5	Medium	E.Madhumitha S. Logapriya
Sprint-4		US-3	Publish Data to The IBM Cloud	8	High	N.Monika V.Malini
Sprint-4		US-1	Create Web UI in Node- Red	10	High	S.Logapriya V.Malini
Sprint-4		US-2	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	10	High	E.Madhumitha N.Monika

## 6. PROJECT PLANNING & SCHEDULING:

### 6.1 SPRINT PLANNING & ESTIMATION:

TITLE	DESCRIPTION	DATE
Literature Survey on TheSelected Project and Information Gathering	A literature survey is a comprehensive summary of previous research on a topic. The literature review surveys scholarly articles, books, and other sources relevant to a particular area of research.	3 <sup>rd</sup> September 2022
Prepare Empathy Map	Empathy map is a collaborative tool teams can use to gain a deeper insightinto their customers.	10 <sup>th</sup> September 2022
Ideation-Brainstorming	Brainstorming is a group problem-solving method that involves the spontaneous contribution ofcreative ideas and solution.	17 <sup>th</sup> September 2022
Define Problem Statement	Problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the gap between the current sate anddesired state of a process or product.	18 <sup>th</sup> Semptember 2022
Proposed Solution	Proposed solution means the technical solution to be provided by the implementation agency in response to the requirementsand the objectives of the project.	24 <sup>th</sup> Semptember 2022
Solution Architecture	Solution architecture is the practice of designing, describing, and managing solution engineering to match it with specific business problems.	1 <sup>st</sup> October 2022

Customer Journey	A customer journey is a tool that helps markets understand the series of connected experiences that customers desire and needs-whether that be completing a desired task or traversing the end-to-end journey from prospect to customer to loyal advocate.	8 <sup>th</sup> October 2022
Functional Requirement	Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks.	15 <sup>th</sup> October 2022
Data Flow Diagrams	It is a graphical representation which is very easy to understand; it helps visualize contents. Data flow diagrams represent detailed and well-explained diagrams of system components.	15 <sup>th</sup> October 2022

Technology Architecture	Technology Architecture is a more well-defined version of solution architecture. It helps us analyze and understand various technologies that need to be implemented in the project.	15 <sup>th</sup> October 2022
Prepare Milestone & Activity	A milestone is a specific point within a project's life cycle used to measure the progress toward the ultimate goal.	22 <sup>nd</sup> October 2022

Sprint Delivery Plan	Sprint planning is an event in the scrum framework where the team determines the product backlog items they will work on during that sprint and discusses their initial plan for completing those product backlog items.	19 <sup>th</sup> November 2022
----------------------	--	--------------------------------

## 6.2 SPRINT DELIVERY SCHEDULE:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1		US-1	Create the IBM Cloud services which are being used in this project.	6	High	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.
Sprint-1		US-2	Configure the IBM Cloud services which are being used in completing this project.	4	Medium	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.
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	S.Logapriya, E.Madhumitha, N.Monika, V.Malini.
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.	5	High	N.Monika V.Malini S.Logapriya, E.Madhumitha,
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	N.Monika, V.Malini. S.Logapriya, E.Madhumitha,



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		US-2	Create a Node-RED service.	10	High	V.Malini S.Logapriya
Sprint-3		US-1	Develop a python script to publish random sensor data such as temperature, moisture, soil and humidity to the IBM IoT platform	7	High	E.Madhumitha S.Logapriya
Sprint-3		US-2	After developing python code, commands are received just print the statements which represent the control of the devices.	5	Medium	E.Madhumitha S. Logapriya
Sprint-4		US-3	Publish Data to The IBM Cloud	8	High	N.Monika V.Malini
Sprint-4		US-1	Create Web UI in Node- Red	10	High	S.Logapriya V.Malini
Sprint-4		US-2	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	10	High	E.Madhumitha N.Monika

**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

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

## 7. CODING AND SOLUTIONING:

### 7.1 FEATURE 1:

```
import random

import

ibmiotf.application

import ibmiotf.device

from time import sleep

import sys

#IBM Watson Device Credentials.

organization = "66gns9"

deviceType = "abcd"

deviceId = "123"

authMethod = "token"

authToken =

"23456789"

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)

```

```

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 and 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,
sprinklerlers areturned ON" %temp_sensor }

, qos=0)

    sleep(1

)

    if success:

        print( 'Published alert1 : ', "Temperature(%s) is high, sprinklerlers are turned ON"
%temp_sensor,"toIBM 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,useother 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 IBMWatson")

                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 indanger,sprinklers turned ON" }, qos=0)
```

```
    sleep(1)
```

if success:

```
    print( 'Published alert4 : ' , "Flame is detected crops are in danger,sprinklers turned  
ON","to IBMWatson")
```

```
#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,  
Irrigationstarted" %moist_level }, qos=0)
```

```
    sleep(1)
```

if success:

```
    print('Published alert5 : ' , "Moisture level(%s) is low, Irrigation started"  
%moist_level,"to IBMWatson" )
```

```
    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 totake 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 received at farmer end

deviceCli.commandCallback = myCommandCallback

# Disconnecting the device and application from the cloud

deviceCli.disconnect()import

randomimport

ibmiotf.application

import

ibmiotf.device from

time import sleep

import sys

#IBM Watson Device Credentials.

organization = "66gns9"

deviceType = "abcd"

deviceId = "123"

authMethod = "token"

authToken =

"23456789"

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(cm

d)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 and 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 areturned ON" %temp_sensor }

, qos=0)

    sleep(1

)

if success:

    print( 'Published alert1 : ', "Temperature(%s) is high, sprinkerlers are turned ON"
%temp_sensor,"toIBM Watson")

    print("")

)else:

    print("sprinkler-1 is

OFF")print("")

    # Sending 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,useother 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 IBMWatson")

```

```

print("")
# Sending 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("")

        # Sending 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 indanger,sprinklers turned ON" }, qos=0)

            sleep(1)

            if success:

                print( 'Published alert4 : ', "Flame is detected crops are in danger,sprinklers turned
ON", "to IBMWatson")

                #Sending 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,
Irrigationstarted" %moist_level }, qos=0)

                    sleep(1)

                    if success:

                        print('Published alert5 : ', "Moisture level(%s) is low, Irrigation started"
%moist_level, "to IBMWatson" )

                        print("")

```

#Sending 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 received by farmer

```
deviceCli.commandCallback = myCommandCallback
```

# Disconnect the device and application from the cloud

```
deviceCli.disconnect()
```

IBM Watson IoT Platform

Browse Action Device Types Interfaces

Identity Device Information Recent Events State Logs

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

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

- Rated Voltage : 6V DC
- Operating Voltage : 4 to 8V DC
- Rated Current\*:  $\leq 30\text{mA}$
- Sound Output at 10cm\* :  $\geq 85\text{dB}$
- Resonant Frequency :  $2300 \pm 300\text{Hz}$
- 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 vehicles such as ambulances, police cars and fire trucks. There are two general types, pneumatic and electronic.**

### 7.2 FEATURE 2:

- Good sensitivity to Combustible gas in wide range .
- High sensitivity to LPG, Propane and Hydrogen .
- Longlife and low cost.
- Simple drive circuit


## 8. TESTING:

### 8.1 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%	



## 8.2 USER ACCEPTANCE TESTING:






[HOME](#) | [ABOUT](#) | [DOWNLOADS](#) | [DOCS](#) | [GET INVOLVED](#) | [SECURITY](#) | [CERTIFICATION](#) | [NEWS](#)

## 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	
 <b>Windows Installer</b> <small>node-v18.12.1-win.exe</small>	 <b>macOS Installer</b> <small>node-v18.12.1.pkg</small>	 <b>Source Code</b> <small>node-v18.12.1.tar.gz</small>

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

macOS Binary (.tar.gz)

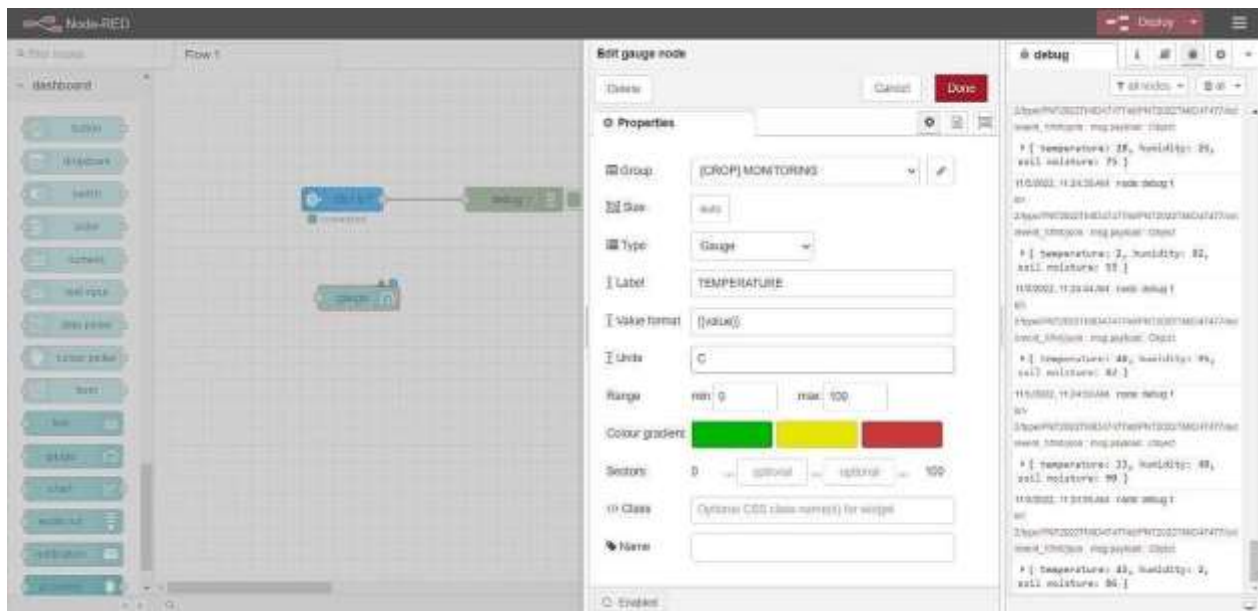
Linux Binaries (x64)

32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	

The screenshot shows the Node-RED web interface. On the left, the 'common' and 'function' palettes are visible. The main workspace contains a flow with an 'inject' node (blue) connected to a 'debug' node (green). The 'debug' node is configured to log 'msg.payload'. The right-hand pane shows the 'debug' console with a list of log messages. The messages are as follows:

```

type:INJECT,msg:msg.payload:0sec
{
  temperature: 80,
  humidity: 31,
  soil_moisture: 54
}
FRI002, 11:20:32 AM now debug 1
on
type:INJECT,msg:msg.payload:0sec
{
  temperature: 8,
  humidity: 84,
  soil_moisture: 55
}
FRI002, 11:20:33 AM now debug 1
on
type:INJECT,msg:msg.payload:0sec
{
  temperature: 80,
  humidity: 86,
  soil_moisture: 55
}
FRI002, 11:20:44 AM now debug 1
on
type:INJECT,msg:msg.payload:0sec
{
  temperature: 90,
  humidity: 33,
  soil_moisture: 55
}
FRI002, 11:20:51 AM now debug 1
on
type:INJECT,msg:msg.payload:0sec
{
  temperature: 70,
  humidity: 4,
  soil_moisture: 36
}
  
```



```

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 vandalization by wild animals and fire has become a major social problem in current time.

It requires urgent attention as no effective solution exists till date for this problem. Thus this project carries a great social relevance as it aims to address this problem. This project will help farmers in protecting their orchards and fields and save them from significant financial losses and will save them from the unproductive efforts that they endure for the protection their fields. This will also help them in achieving better crop yields thus leading to their economic wellbeing.

## **10.ADVANTAGES & DISADVANTAGES:**

### **ADVANTAGE:**

The main advantage is that it protects the crops in farm area from animals. It mainly detects the soil moisture, humidity and temperature ultrasonic sensor detects the birds entering in the fields. This system will continuously check for soil characteristics inside the field.

### **DISADVANTAGE:**

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.

## **11.CONCLUSION:**

Farmers encounter severe threats in rural parts of India LIKE damage done by birds and animals. Hence, to overcome this issue we have designed a system in which sound is played to scare the animals and birds, so that it will automatically run away. Therefore, the designed system is affordable and useful to the farmers. The designed system won't be harmful to animals and person, and it protects the farm areas.

## **12.FUTURE SCOPE:**

In the future, there will be a large scope for this system. The IR sensors and Ultrasonic sensors are used to collect the information and transmitted it through GSM. This project is further enhanced by wireless sensor network. The type of sensors like finding the moisture content of the soil, growth of the crop and nutrition content in the soil. These sensors gather informations which is useful to the farmers and able to conscious of the farm land from anyplace in the world.

### **13. APPENDIX:**

#### **SOURCE CODE:**

```
import time
import sys
import ibmiotf.application
# to install pip install ibmiotf
import ibmiotf.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 auth token

def myCommandCallback(cmd): # function for Callback if

    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 required information: 'interval'")

        interval = cmd.data['interval']
        elif cmd.command == "print":

        if 'message' not in cmd.data:

        print("Error - command is missing required information: '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)#
```

```
.....
```

```
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:
```

```
deviceCli.commandCallback = myCommandCallback
```

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

## SENSOR.PY

```
import time  
import sys  
import
```

```
ibmiotf.application
```

```
import ibmiotf.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 = "LWVpQPpVQ166HWN48f" # 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 of type "greeting" 10 times
```

```
deviceCli.connect()
```

```
while True:
```

```
temp=random.randint(0,1
```

```
00)
```

```
pulse=random.randint(0,100) soil=random.randint(0,100)
```

```

data = { 'temp': temp, 'pulse': pulse, 'soil': soil } # print data
def
myOnPublishCallback():
print ("Published Temperature = %s C" % temp, "Humidity = %s %" % pulse, "Soil Moisture = %s %" % soil, "to IBM
Watson")

success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)    if not
success: print("Not connected to

IoT")
time.sleep(1)

deviceCli.commandCallback = myCommandCallback

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

```

Node-RED FLOW :

```

[
{
  "type": "ibmiotout", "z": "630c8601c5ac3295", "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",
"passthru": false, "label": "MotorON",
"tooltip": "",
"color": "",
"bgcolor": "",
"className": "",
"icon": "", "payload": {"command": "motoron"}, "payloadType": "str",
"topic": "motoron",
"topicType": "str", "x": 360,
"y": 160, "wires": [{"625574ead9839b34"}]},
{
  "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"}]},
```



```
"name":"weather_monitor","keepalive":"60","serverName":"","  
"cleansession":true,"appld":"","  
"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}  
]
```

```
[  
{  
"type":"ibmiotin","z":"03acb6ae05a0c712",  
"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"}]
},
{
"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"}]
},
```

```
{
  "name":"Humidity",
  "func":"msg.payload = msg.payload.pulse;\nglobal.set('p',msg.payload)\nreturn msg;",
  "outputs":1,
  "noerr":
  0,
  "initialize":"",
  "finalize":"",
  "libs":[
  ],
  "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": 860,
  "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": "",
  "libs": [
  ],
}
```

"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",

"ctype": "gauge",


"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": "get", "upload": false, "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":"","
  "headers":{},
  "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":false, "className":""
},
{
```

```
"id": "1f4cb829.2fdee8",  
"type": "ui_tab",  
"name": "Home",  
"icon": "dashboard", "order": 3, "disabled": false, "hidden": false }
```

GitHub & Project Demo Link

<https://inshotapp.page.link/InShotEditor>

[IBM-EPBL/IBM-Project-13225-1659514627](#)



