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



**Team ID:** PNT2022TMID42671

**Team Members**

Kandra Bhanu Prasad Reddy

Narmadha. S

Dhilipan. M

Jothika. N

**INTRODUCTION**

**PROJECT OVREVIEW:**

An intelligent crop protection system helps the farmers in protecting the crop from the animals and birds which destroy the crop. This system also helps farmers to monitor the soil moisture levels in the field and also the temperature and humidity values near the field. The motors and sprinklers in the field can be controlled using the mobile application.

**PURPOSE:**

Our main purpose of the project is to develop intruder alert to the farm, to avoid losses due to animal. 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. To identify the field condition like temperature, humidity and soil moisture we can use the mobile application to find the condition of the field

# LITERATURE SURVEY

**EXISTING PROBLEM:**

The existing system mainly provide the surveillance functionality. Also, these systems don’t provide protection from wild animals, especially in such an application area. They also need to take actions based on the type of animal that tries to enter the area, as different methods are adopted to prevent different animals from entering restricted areas. The other commonly used method by farmer in order to prevent the crop vandalization by animals include building physical barriers, use of electric fences and manual surveillance and various such exhaustive and dangerous method.

**REFERENCES:**

1. Mr. Pranav shita , M. Jayesh redij, Mr.Shikhar Singh, Mr.Durvesh Zagade, Dr. Sharada Chougule. Department of ELECTRONICS AND TELECOMMUNICATION ENGINEERING,

Finolex Academy of Management and technology, ratangiri,

India.

1. N.Penchalaiah, D.Pavithra, B.Bhargavi, D.P.Madhurai,

K.EliyasShaik,S.Md.sohaib.Assitant Professor, Department of CSE,AITS, Rajampet,India UG Student, Department of CSE,AITS,Rajampet, India.

1. Mr.P.Venkateswara Rao, Mr.Ch Shiva Krishna ,MR M Samba Siva ReddyLBRCE,LBRCE,LBRCE.
2. Mohit Korche,Sarthak Tokse, ShubhamShirbhate, Vaibhav Thakre,S. P. Jolhe(HOD). Students , Final Year,Dept.of

Electrical engineering,Government

College of engineering,Nagpur head of dept.,Electrical engineering,Government College of engineering,Nagpur.

**PROBLEM STATEMENT DEFINITION STATEMENT:**

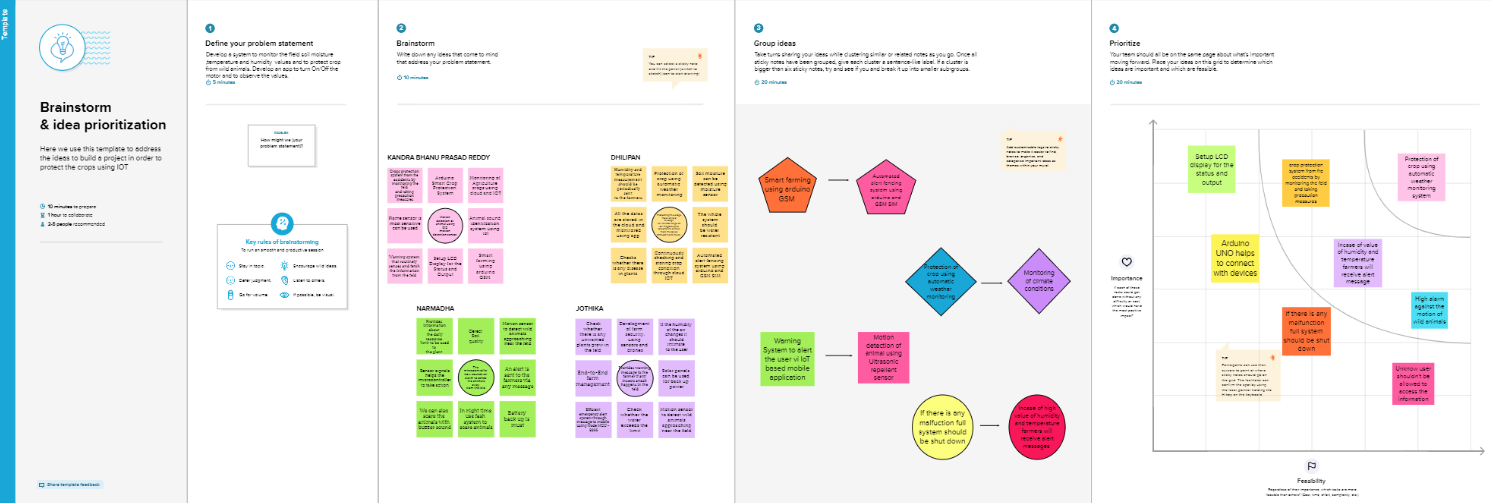
In the world economy of many Countries dependent upon the agriculture. In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. this leads to huge loss for the farmers.it is not possible for farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets food requirements of the people and produces several raw materials for industries. But because of animal interference in agricultural lands, there will be huge loss of crops. Crops will be totally getting destroyed.

**IDEATION AND PROPOSED SOLUTION**

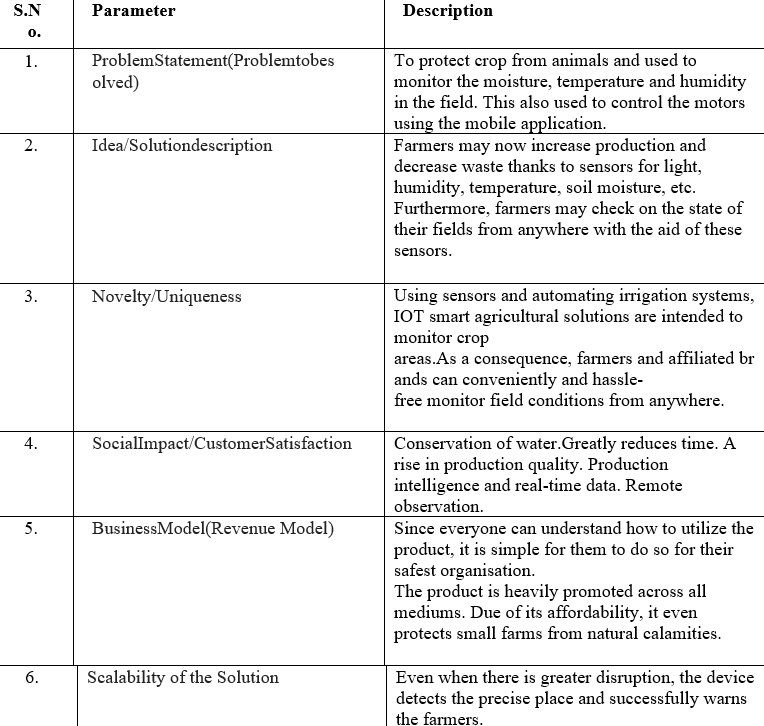
**EMPATHY MAP CANVAS:**

a. 

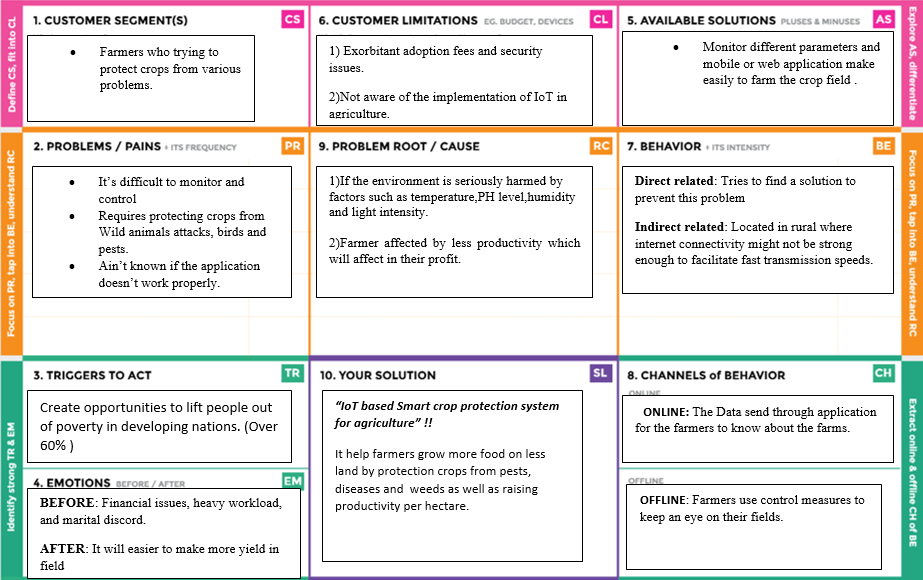
**IDEATION AND BRAINSTORMING:**



**PROPOSED SOLUTION:**

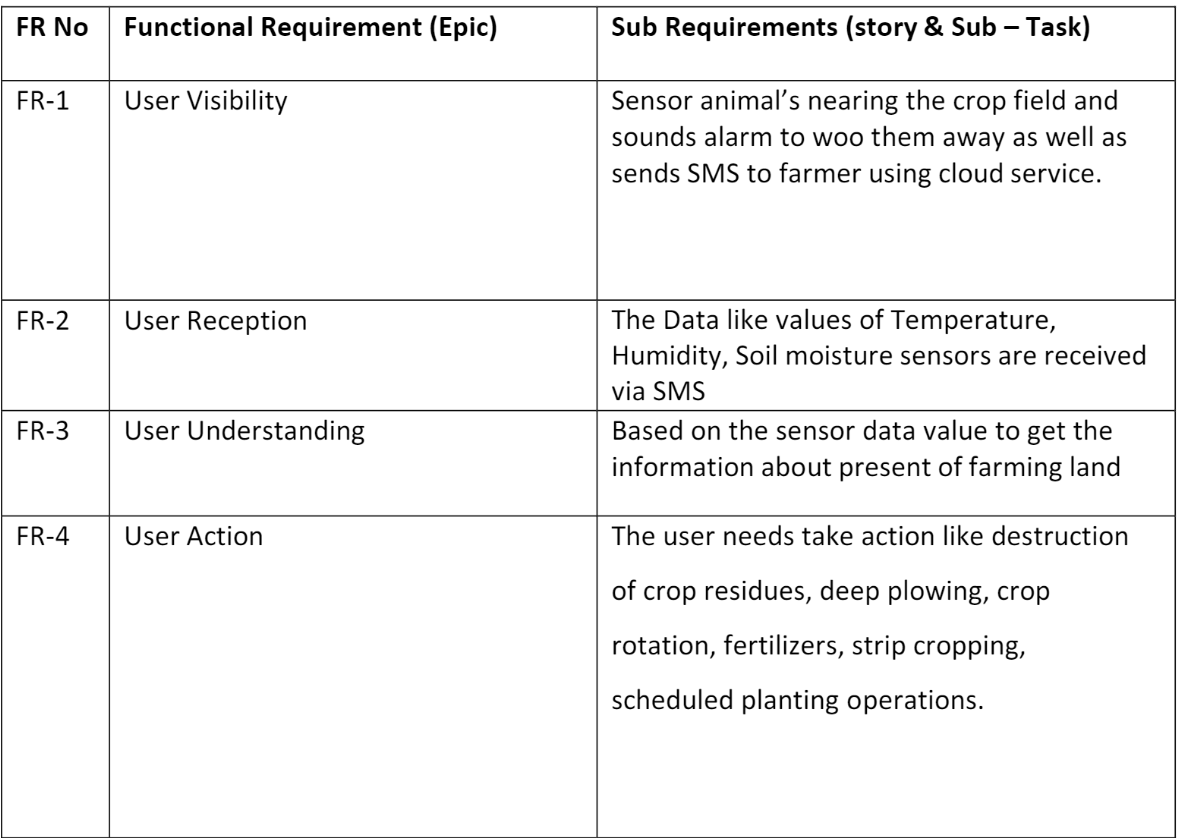


**PROBLEM SOLUTION FIT:**

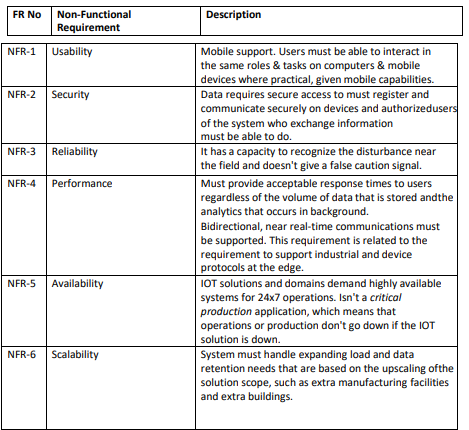


**REQUIREMENT ANALYSIS**

**FUNCTIONAL REQUIREMENT:**

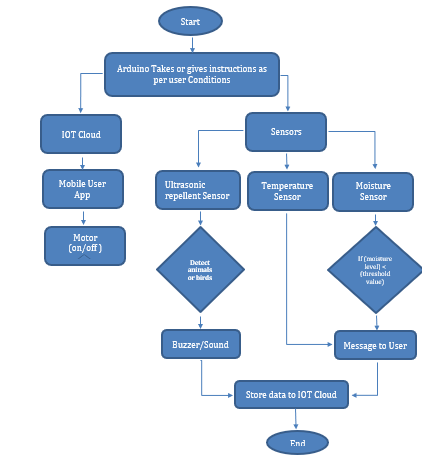


**NON FUNCTINAL REQUIREMENT:**

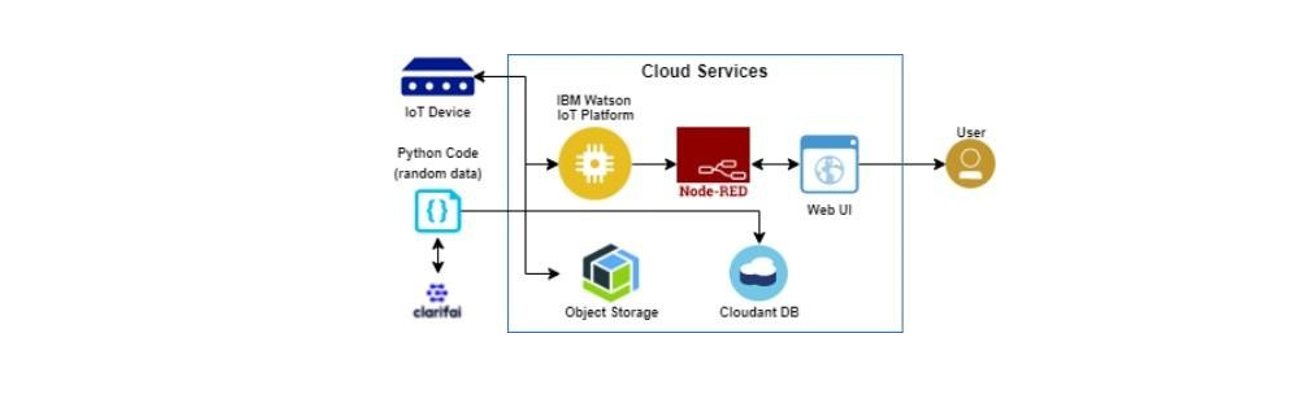


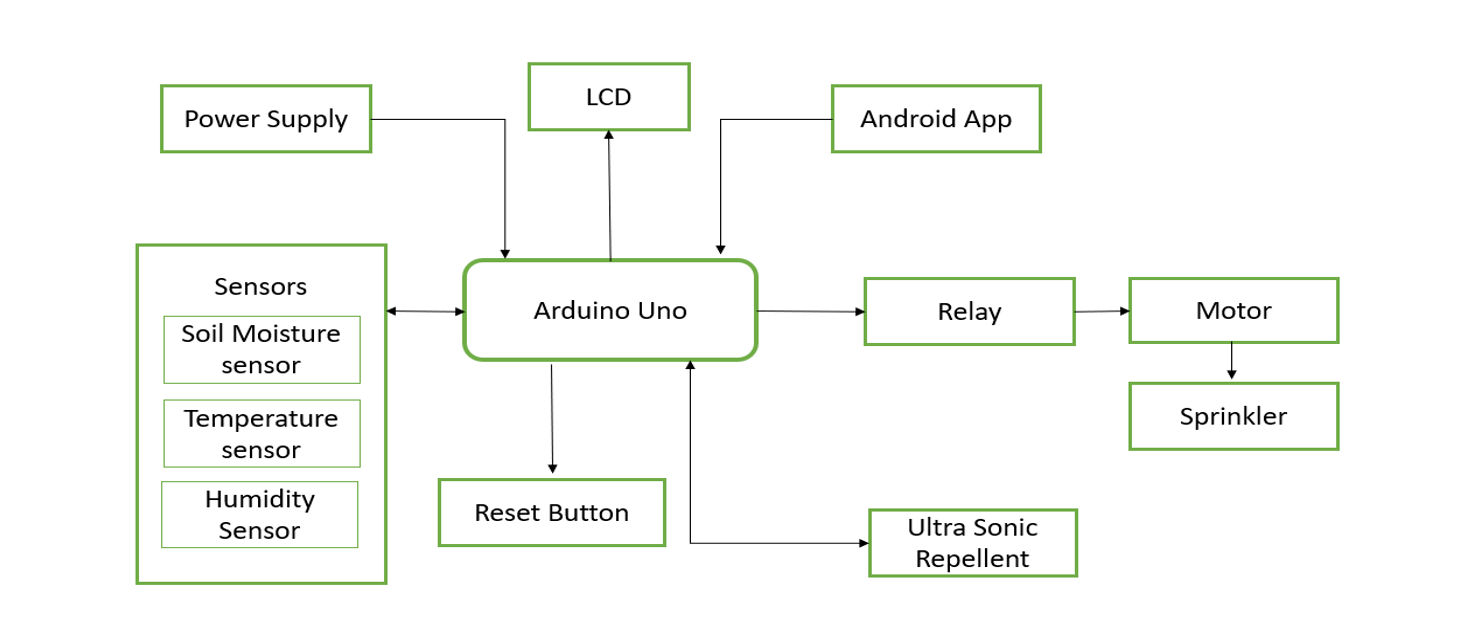
**PROJECT DESIGN**

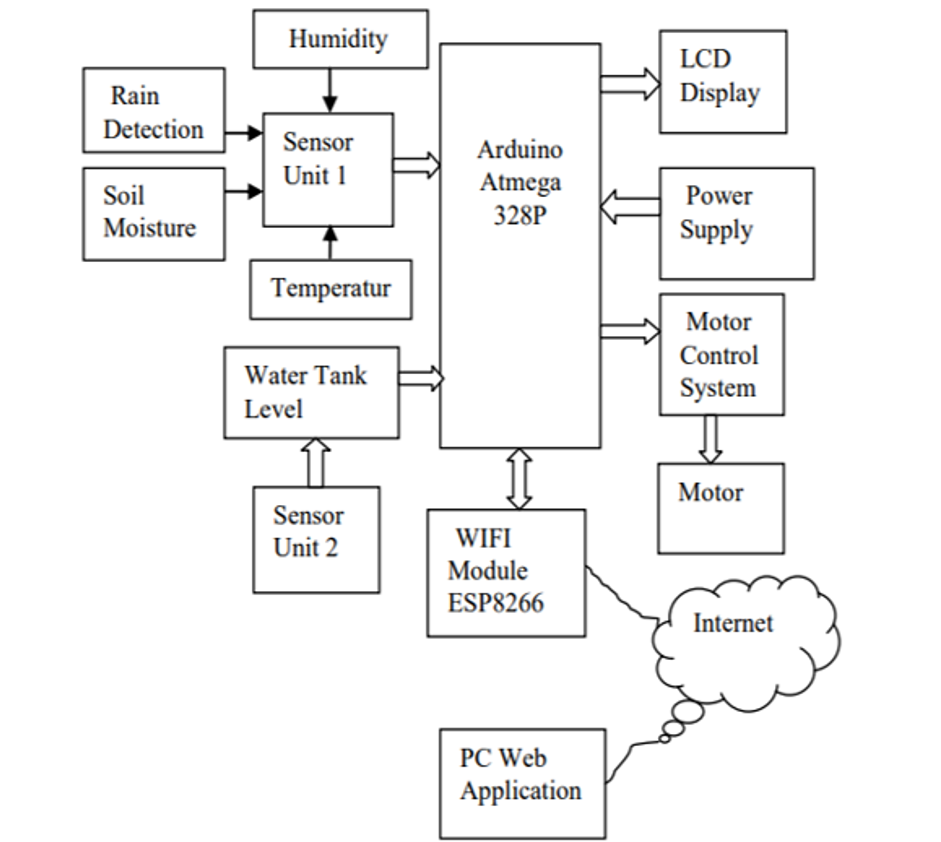
**DATA FLOW DIAGRAM:**

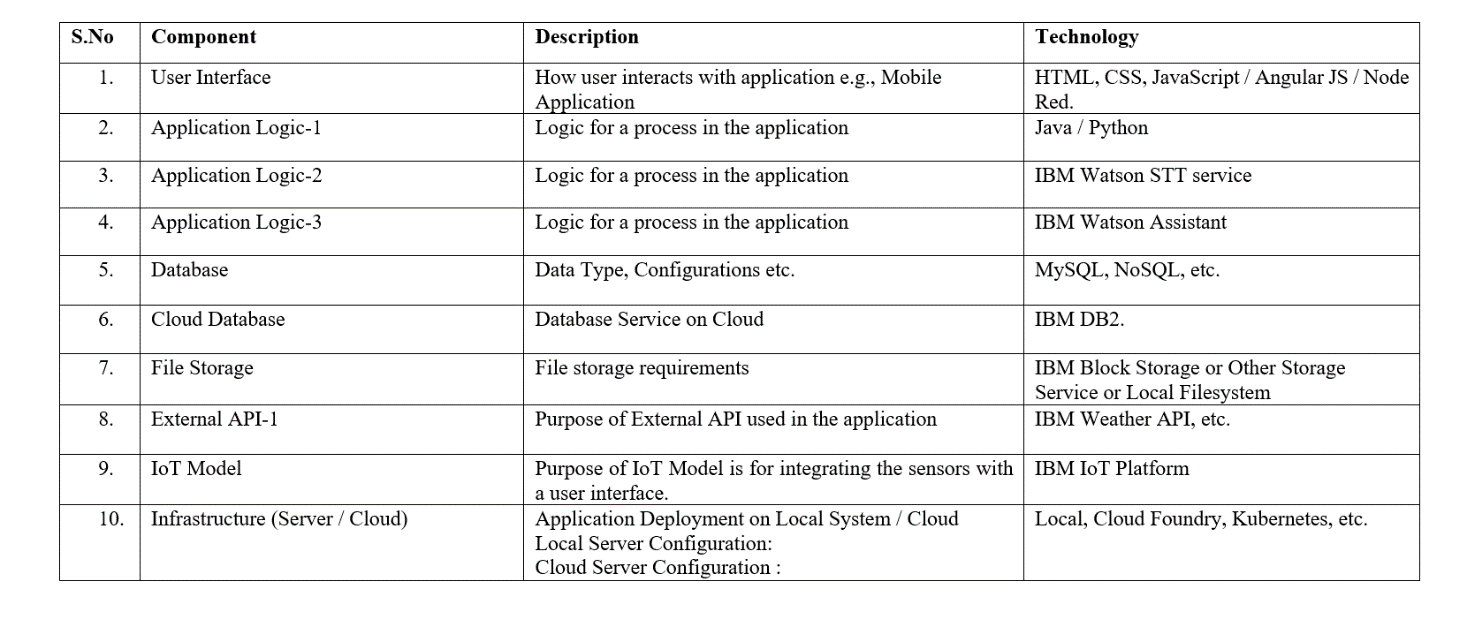


**SOLUTION AND TECHNICAL ARCHITECTURE:**



**Hardware Block Diagram**





**USER STORIES:**

**Sprint 1**

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "dswbln"

deviceType = "Crop\_protector"

deviceId = "123456"

authMethod = "token"

authToken = "1234567890"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command'])

status = cmd.data['command']

if status == "motoron":

print("motor is on")

elif status == "motoroff":

print("motor is off")

else:

print("please send proper command")

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()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times

deviceCli.connect()

while True:

# Get Sensor Data from DHT11

temperature = random.randint(70, 80)

humidity = random.randint(50, 60)

soil\_moisture = random.randint(21, 40)

data = {'temperature': temperature, 'humidity': humidity, 'soil\_moisture': soil\_moisture}

# print data

def myOnPublishCallback():

print("Published Temperature = %s C" % temperature, "Humidity = %s %%" % humidity, "Soil\_moisture = %s %%" % soil\_moisture,"to IBM Watson")

success = deviceCli.publishEvent("Bhanu cropprotector", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

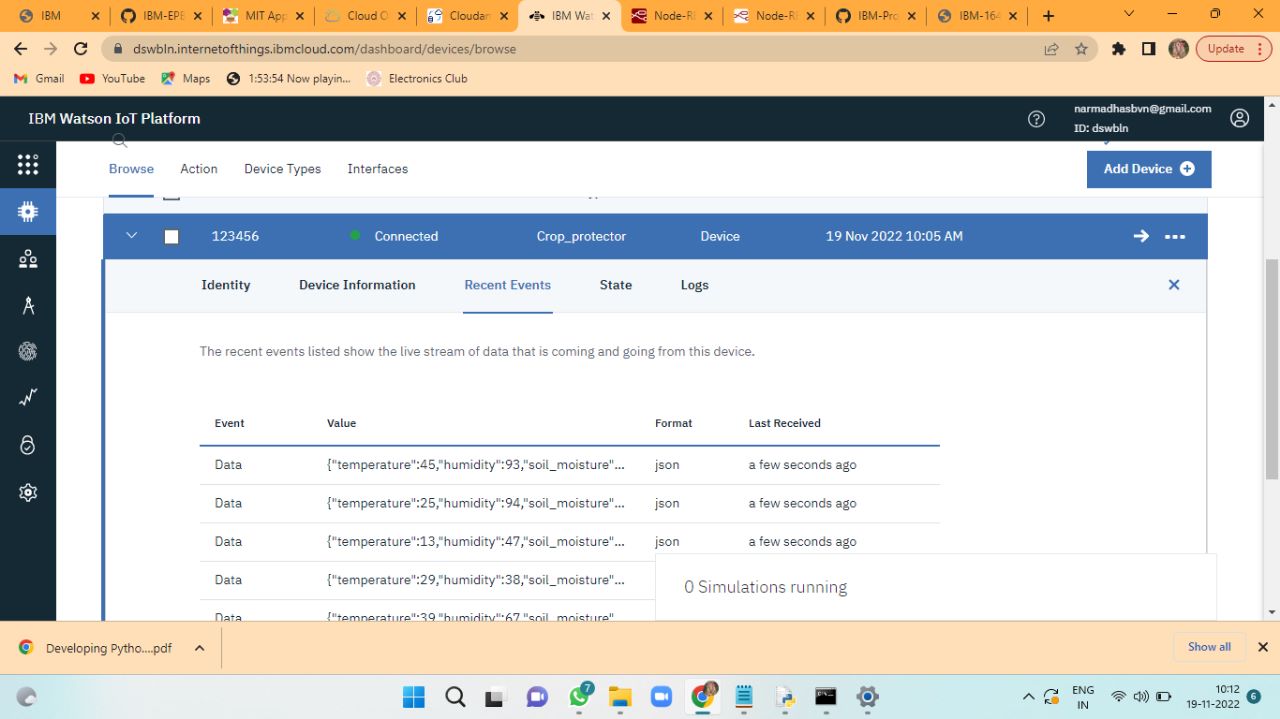
print("Not connected to IoTF")

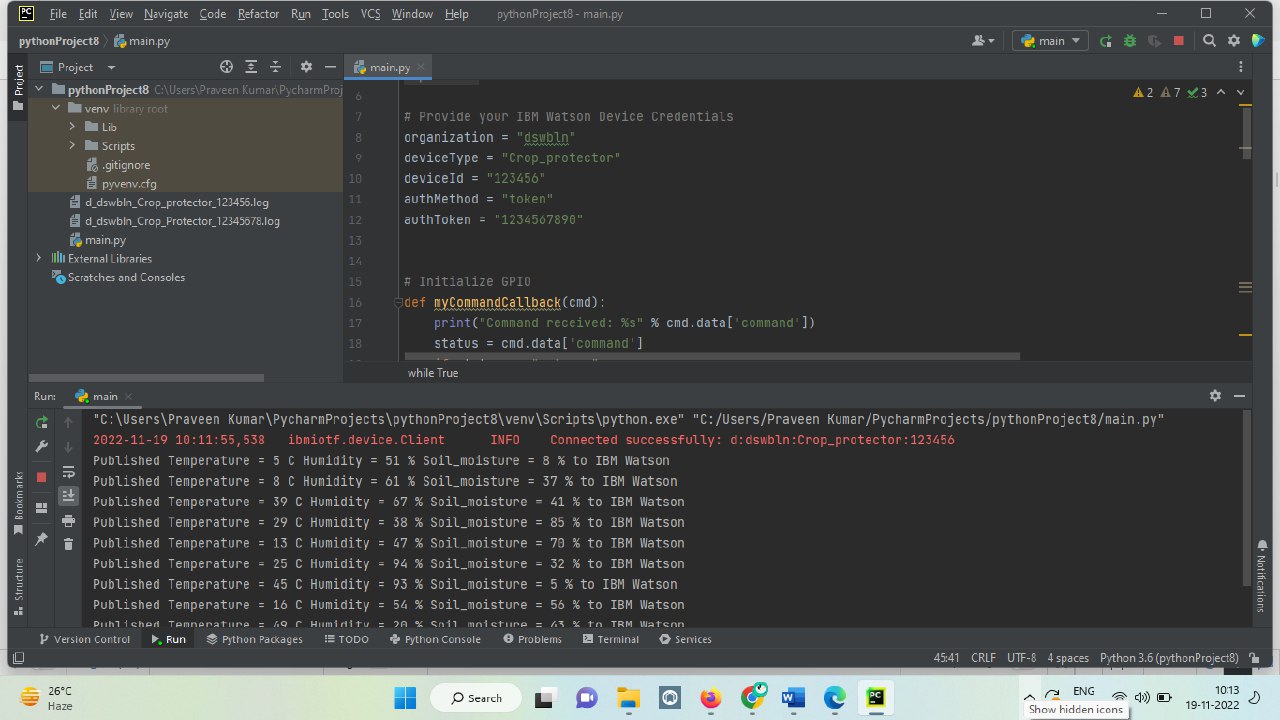
time.sleep(10)

deviceCli.commandCallback = myCommandCallback

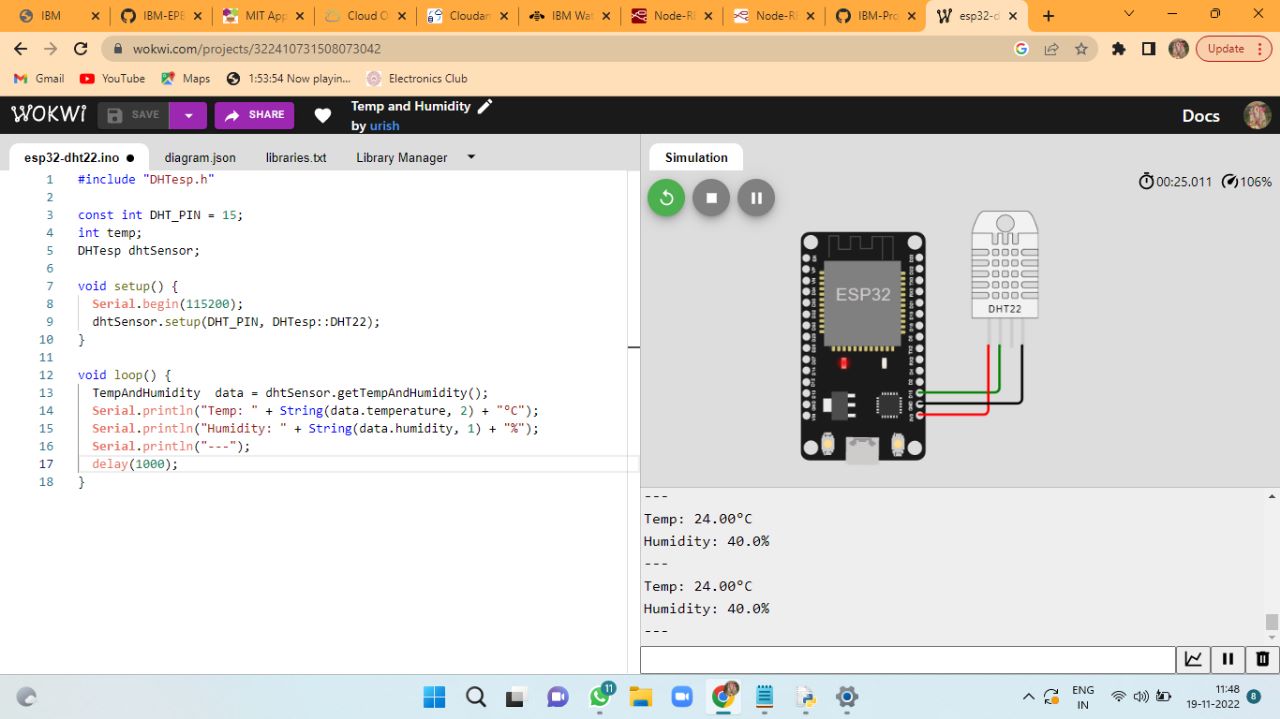
# Disconnect the device and application from the cloud

deviceCli.disconnect()

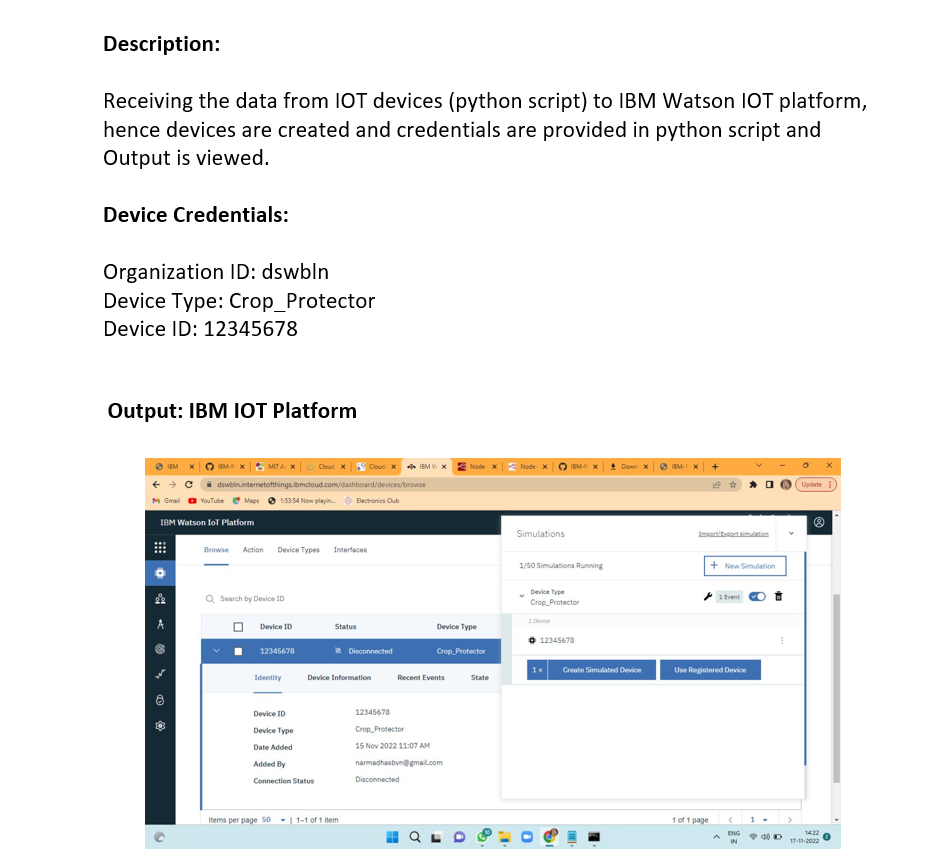


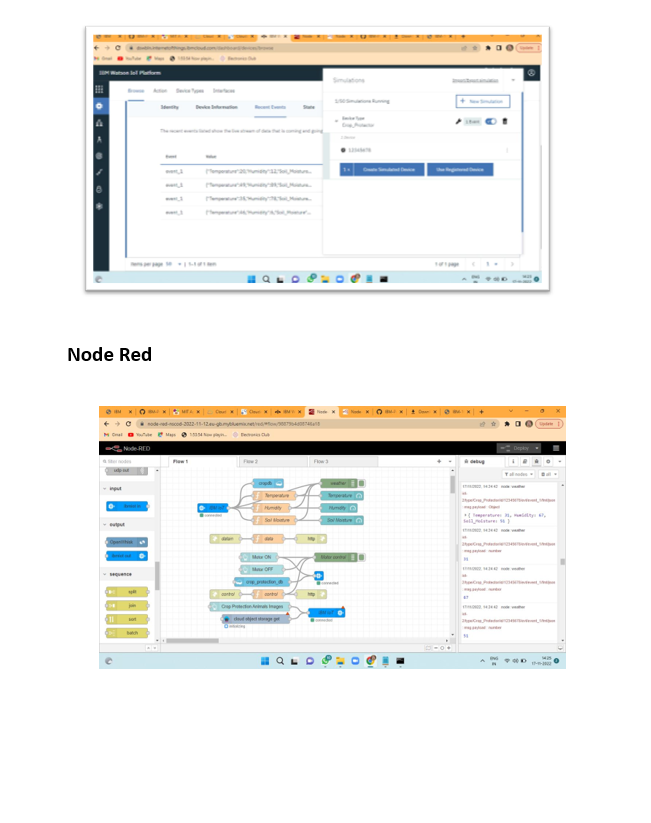
****

**CONNECTING SENSOR WITH ESP32-RASP USING C++ CODE**

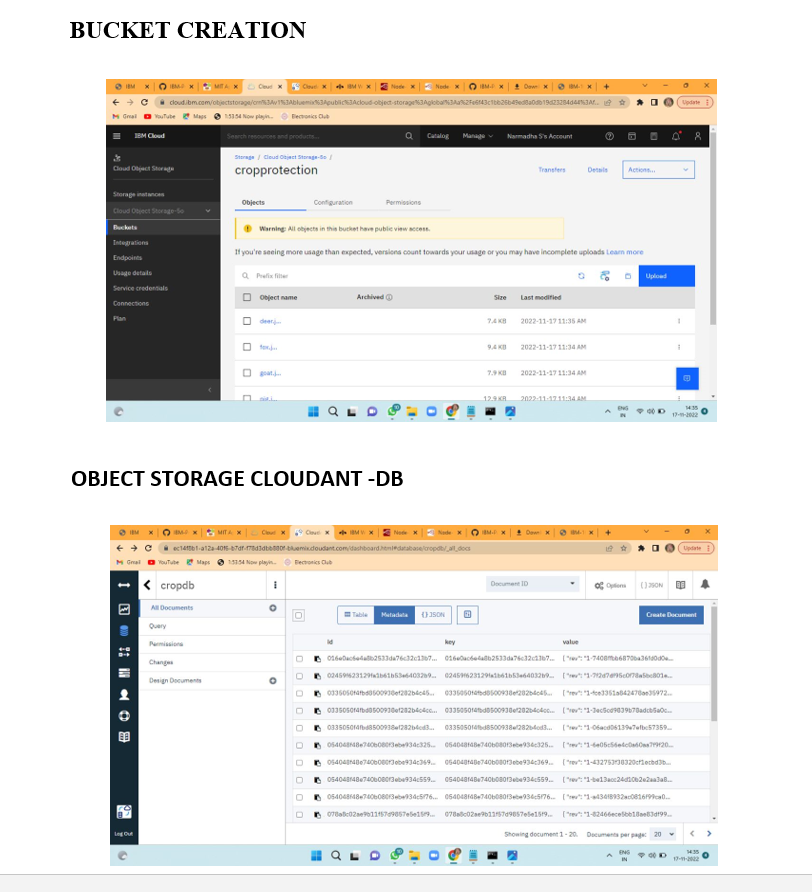
****

**Sprint 2**

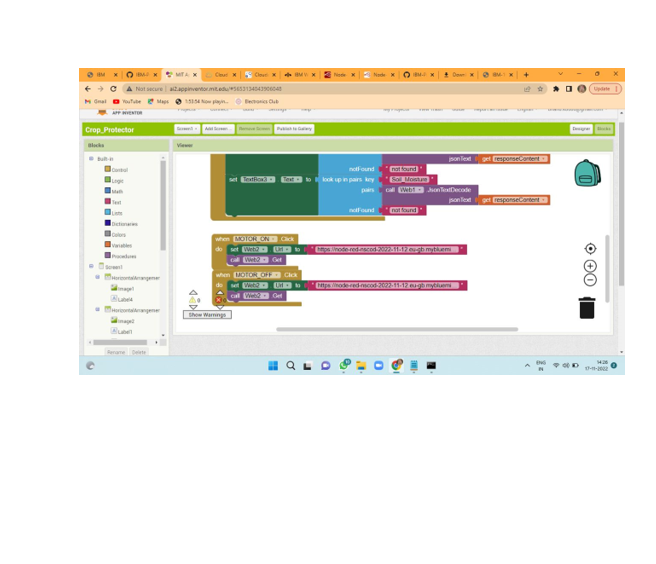
****

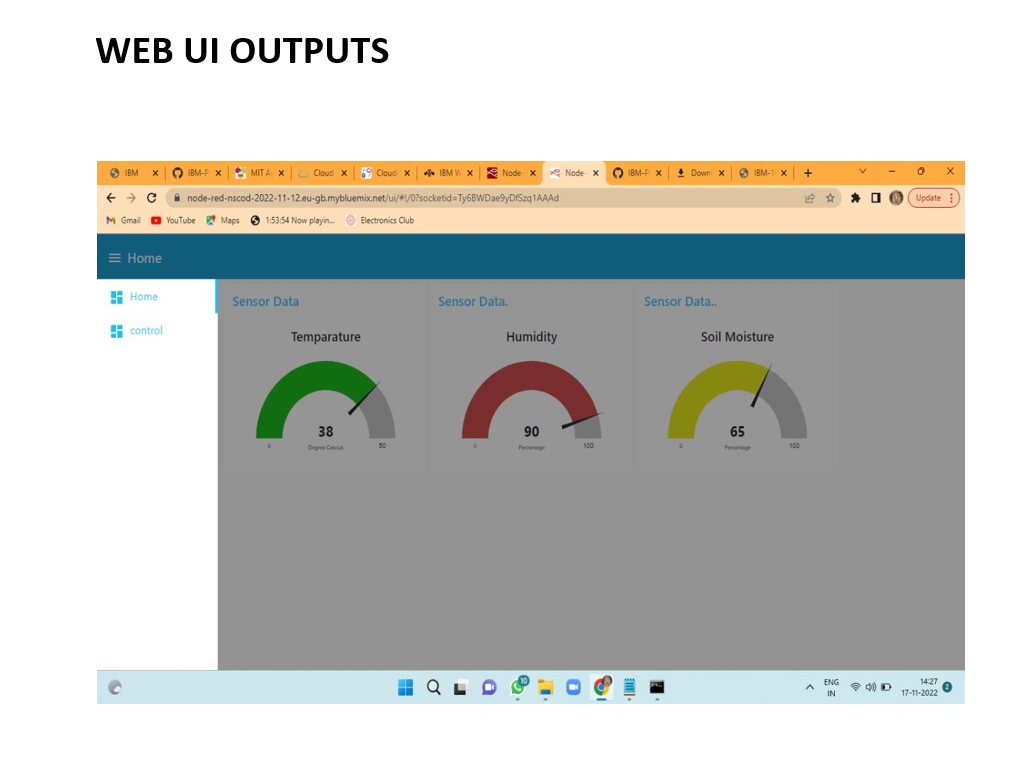


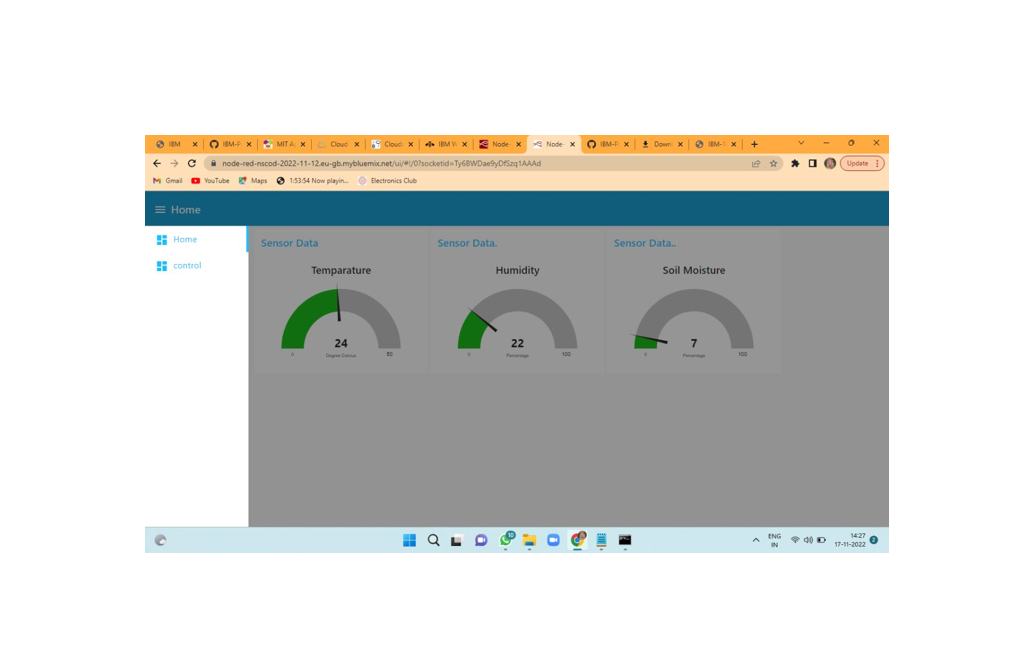
**SPRINT 3**

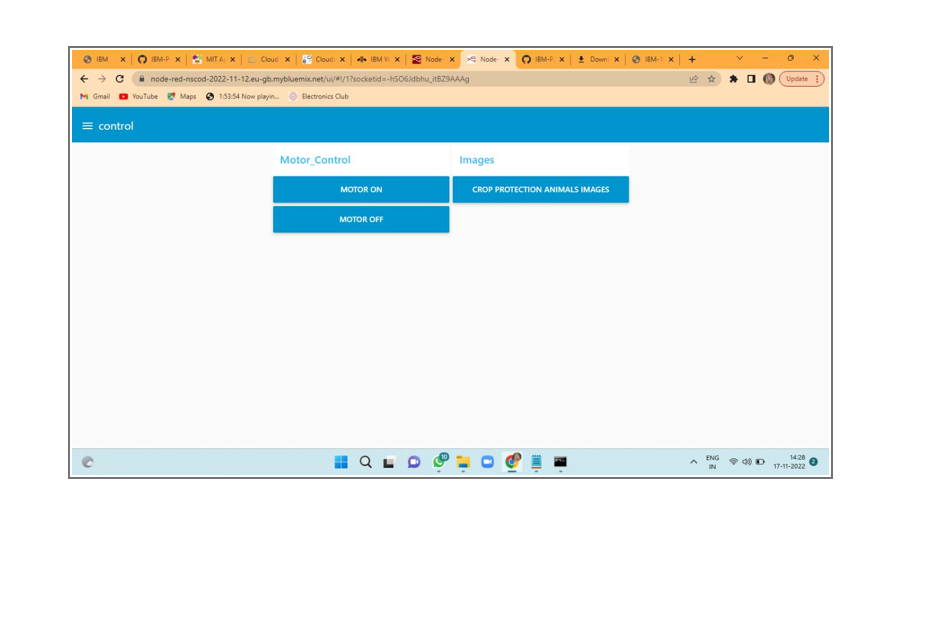
****

****



**Sprint 4**

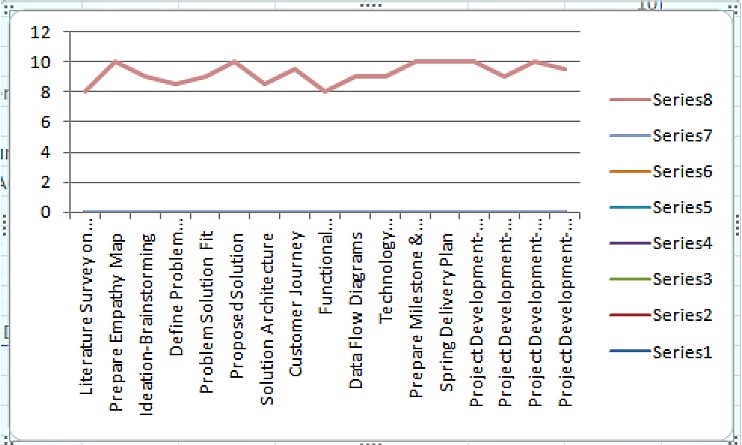




**PROJECT PLANNINGAND SCHEDULING**

**SPRINT PLANNING AND ESTIMATION:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date** | **Story Points Compeleted ( as on planned end date)** | **Sprint Release date (Actual**) |
| Sprint 1 | 20 | 6 days | 27 Oct 2022 | 1 Nov 2022 | 20 | 1 Nov 2022 |
| Sprint 2 | 20 | 4 days | 31 Oct 2022 | 3 Nov 2022 | 20 | 3 Nov 2022 |
| Sprint 3 | 20 | 5 days | 5 Nov 2022 | 9 Nov 2022 | 20 | 9 Nov 2022 |
| Sprint 4 | 20 | 4 days | 9 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |



**CODING AND SOLUTIONING**

# FEATURE-1

import cv2

import numpy as np

import wiot.sdk.device

import playsound

import random

import time

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

#CloudantDB

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result, ResultByKey

from clarifai\_grpc.channel.clarifai\_channel import ClarifaiChannel

from clarifai\_grpc.grpc.api import service\_pb2\_grpc

stub = service\_pb2\_grpc.V2Stub(clarifaiChannel.get.grpc\_channel())

from clarifai\_grpc.grpc.api import service\_pb2, resource\_pb2

from clarifai\_grpc.grpc.api.status import status\_code\_pb2

#This is how you authenticate

metadata = (('authorization', 'key 00a3821c08445ca1b9c031ff931243e8'),)

COS\_ENDPOINT = "https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints"

COS\_API\_KEY\_ID = "Zoqf\_NFV\_WLd0AvrD\_JXe7bAImD8gQzN62M5y5R6IYhC"

COS\_AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"

COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/e6f43c1bb26b49ed8a0db19d23284d44:f5a7d673-5fae-4bac-b7bf-6a55fd6b6788::"

clientdb = cloudant("apikey-v2-xnzlgzuusjqspisc90g0l4l38lgb2r0kcyfy0dtgbff", "535888a02ad96bda3decb2c0291820d5", url: "https://apikey-v2-xnzlgzuusjqspisc90g0l4l38lgb2r0kcyfy0dtgbff:535888a02ad96bda3decb2c0291820d5@ec14f8b1-a12a-40f6-b7df-f78d3dbb880f-bluemix.cloudantnosqldb.appdomain.cloud")

clientdb.connect()

#Create resource

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_RESOURCE\_CRN,

ibm\_auth\_endpoint=COS\_AUTH\_ENDPOINT,

config=Config(signature\_version="oauth"),

endpoint\_url=COS\_ENDPOINT

)

def = multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

#set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

#set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

#set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size

)

#the upload\_fileobj method will automatically execute a multi-part upload

#in 5 MB chunks size

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

command=cmd.data['command']

print(command)

elif(command=="motoron"):

print('motoron')

elif(command=="motoroff"):

print('motoroff')

myConfig = {

"identity": {

"orgId": "dswbln",

"typeId": "Crop\_Protector",

"deviceId": "12345678"

},

"auth": {

"token": "1234567890"

}

}

client = wiot.sdk.device.DeviceClient(config=myConfig, logHandlers=None)

client.connect()

database\_name = "cropdb"

my\_database = clientdb.create\_database(database\_name)

if my\_dtabase.exists():

print(f"'(database\_name)' successfully created.")

cap=cv2.VideoCapture("garden.mp4")

if(cap.isOpened()==True):

print('File opened')

else:

print('File not found')

while(cap.isOpened()):

ret, frame = cap.read()

gray = cv3.cvtColor(frame, cv2.COLOR\_BGR@GRAY)

imS= cv2.resize(frame, (960,540))

cv2.inwrite('ex.jpg',imS)

with open("ex.jpg", "rb") as f:

file\_bytes = f.read()

#This is the model ID of a publicly available General model. You may use any other public or custom model ID.

request = service\_pb2.PostModeloutputsRequest(

model\_id='82eaf1c767a74869964531e4d9de5237',

inputs=[resources\_pb2.Input(data=resources\_pb2.Data(image=resources\_pb2.Image(base64=file\_bytes))

)])

response = stub.PostModelOutputs(request, metadata=metadata)

if response.status.code != status\_code\_pb2.SUCCESS:

raise Exception("Request failed, status code: " + str(response.status.code))

detect=False

for concept in response.outputs[0].data.concepts:

#print('%12s: %.f' % (concept.name, concept.value))

if(concept.value>0.98):

#print(concept.name)

if(concept.name=="animal"):

print("Alert! Alert! animal detected")

playsound.playsound('alert.mp3')

picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")

cv2.inwrite(picname+'.jpg',frame)

multi\_part\_upload('Umamaheswari', picname+'.jpg', picname+'.jpg')

json\_document={"link":COS\_ENDPOINT+'/'+'dear'+'/'+picname+'.jpg'}

new\_document = my\_database.create\_document(json\_document)

if new\_document.exists():

print(f"Document successfully created.")

time.sleep(5)

detect=True

moist=random.randint(0,100)

humidity=random.randint(0,100)

myData={'Animal':detect,'moisture':moist,'humidity':humidity}

print(myData)

if(humidity!=None):

client.publishEvent(eventId="status",msgFormat="json", daya=myData, qos=0, onPublish=None)

print("Publish Ok..")

client.commandCallback = myCommandCallback

cv2.imshow('frame',imS)

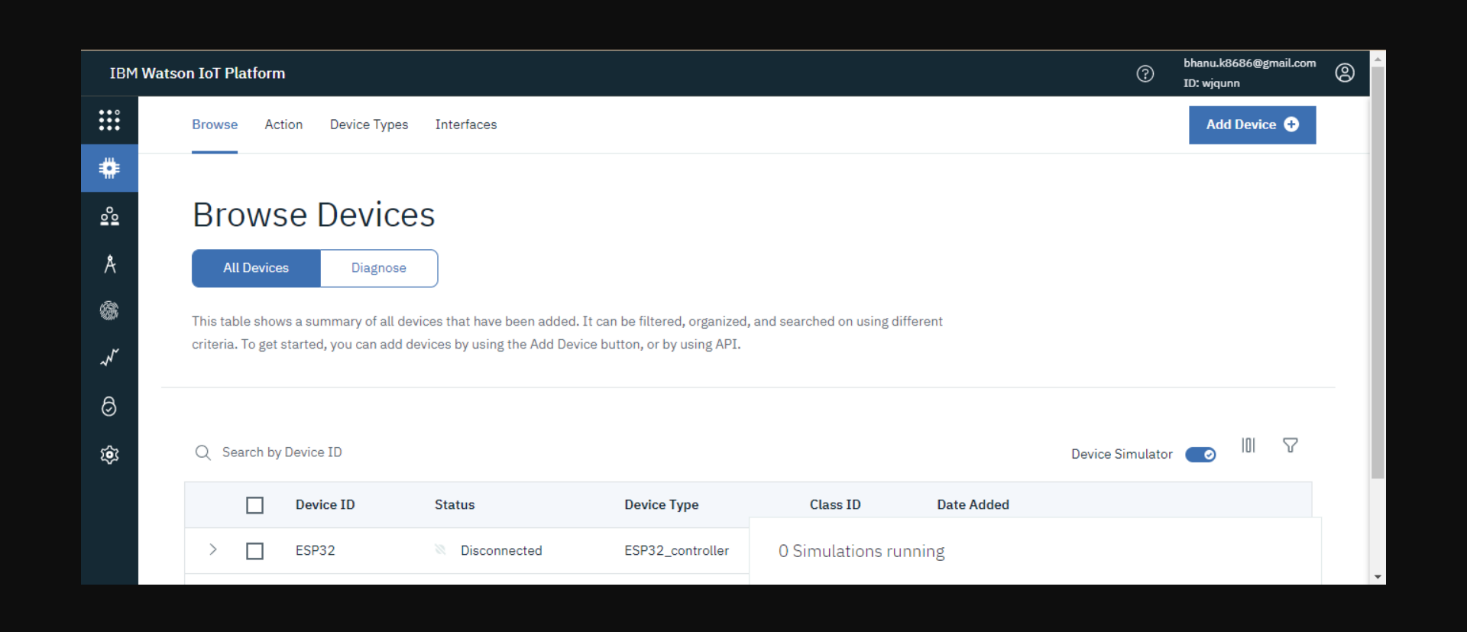
if cv2.waitKey(1) & 0xFF == ord('q'):

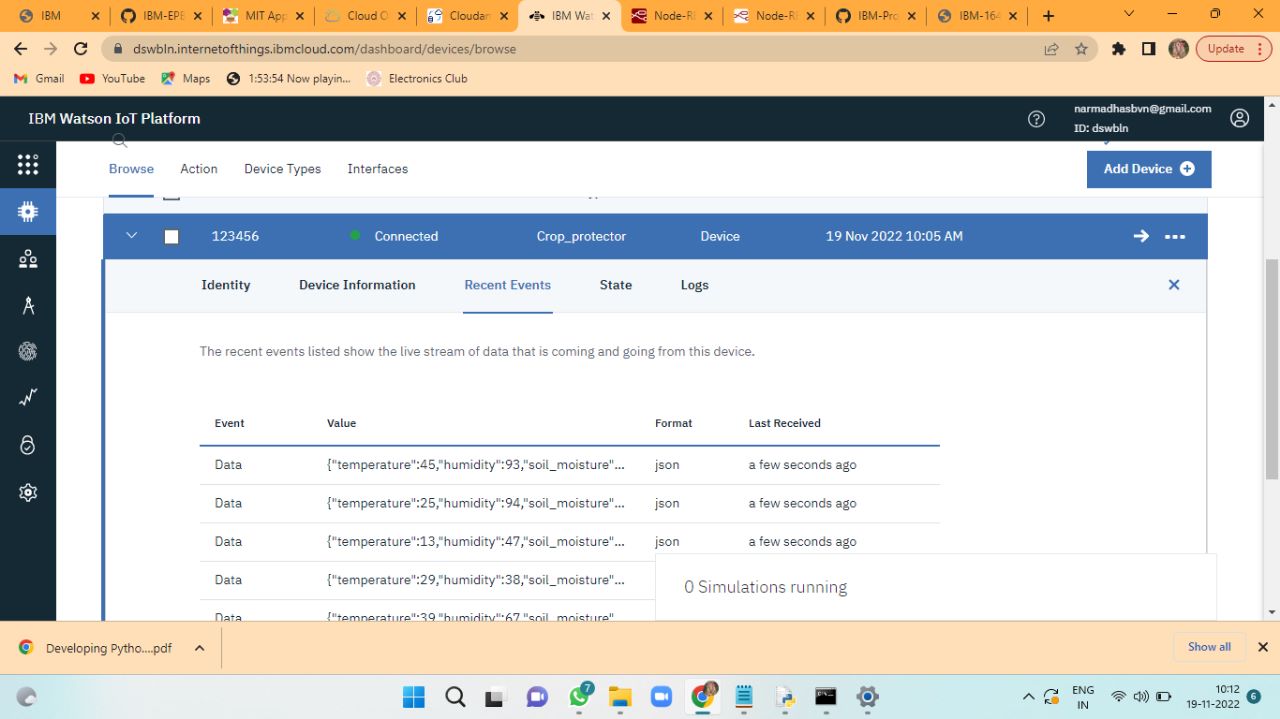
break

client.disconnect()

cap.release()

cv2.destroyAllWindows()





## Features

Output: Digital pulse high (3V) when triggered (motion on 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\*: ≤30mA
* Sound Output at 10cm\*: ≥85dB
* Resonant Frequency: 2300 ±300Hz

Most modern ones are civil defence 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, pneuma c and electronic.

**FEATURE-2:**

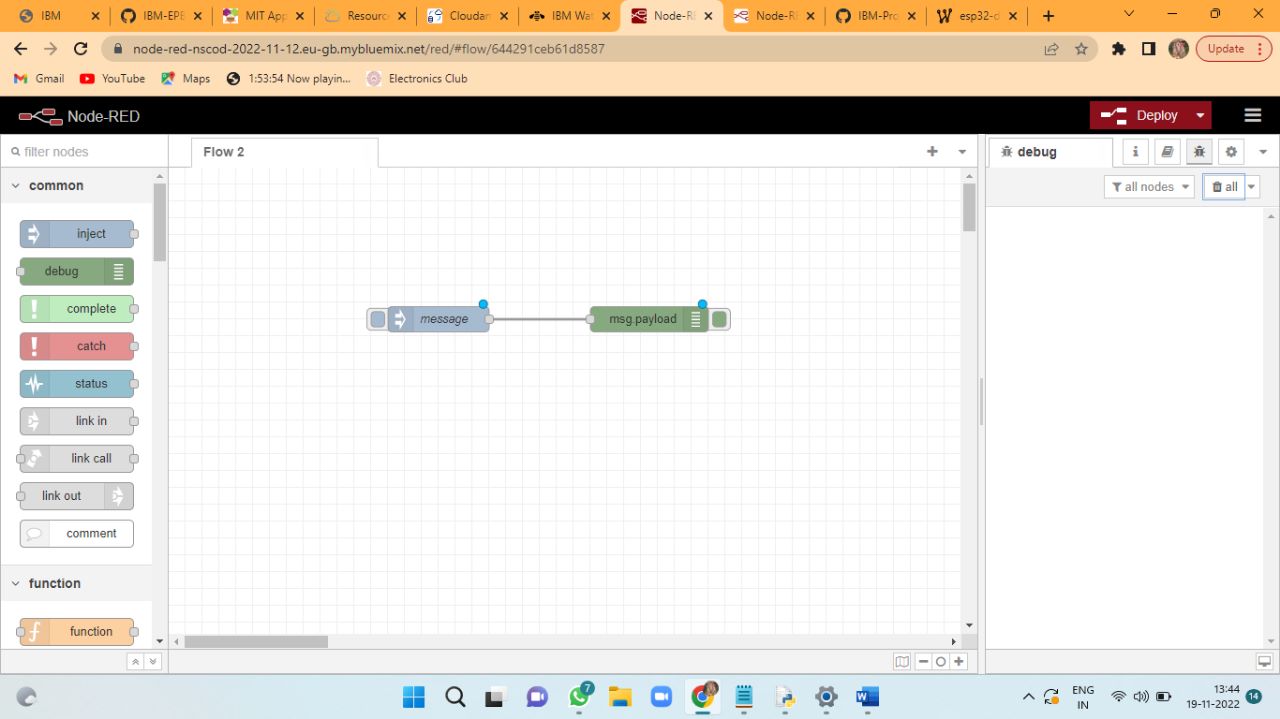
* 1. Good sensitivity to Combustible gas in wide range.
  2. High sensitivity to LPG, Propane and Hydrogen.
  3. Long life and low cost.
  4. Simple drive circuit.

# TESTING

**TEST CASES:**

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

**User Acceptance Testing:**



**RESULTS**

The problem of crop vandalization by wild animals 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.

## ADVANTAGES AND DISADVANTAGES

**Advantage:**

Intelligent data collection. Sensors installed on IoT devices are able to collect a large volume of useful information for farmers. With greater production control, IoT in agriculture facilitates cost-efficient management. A repelling and a monitoring system is provided to prevent potential damages in Agriculture, both from wild animal attacks and weather conditions. Soil moisture is detected periodically and field is watered to avoid crop damage.

**Disadvantage:**

IoT farming will require certain skill sets in particular in order to understand and operate the equipment.

**CONCLUSION:**

A IoT Web Application is built for smart agricultural system using Watson IoT platform, Watson simulator, IBM cloud and Node-RED

## FUTURE SCOPE

In the future, there will be very large scope, this project can be made based on Image processing in which wild animal 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’s security system will be activated.

**APPENDIX**

## SOURCE CODE

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "dswbln"

deviceType = "Crop\_protector"

deviceId = "123456"

authMethod = "token"

authToken = "1234567890"

# Initialize GPIO

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command'])

status = cmd.data['command']

if status == "motoron":

print("motor is on")

elif status == "motoroff":

print("motor is off")

else:

print("please send proper command")

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()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times

deviceCli.connect()

while True:

# Get Sensor Data from DHT11

temperature = random.randint(70, 80)

humidity = random.randint(50, 60)

soil\_moisture = random.randint(21, 40)

data = {'temperature': temperature, 'humidity': humidity, 'soil\_moisture': soil\_moisture}

# print data

def myOnPublishCallback():

print("Published Temperature = %s C" % temperature, "Humidity = %s %%" % humidity, "Soil\_moisture = %s %%" % soil\_moisture,"to IBM Watson")

success = deviceCli.publishEvent("venkatesh\_smartfarmer", "json", data, qos=0, on\_publish=myOnPublishCallback)

if not success:

print("Not connected to IoTF")

time.sleep(10)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

## GitHub & Project Demo Link

**https://github.com/IBM-EPBL/IBM-Project-20850-1659764981**