IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

Team ID: PNT2022TMID11540

Team Members

Thavamurukan R

Tharani S R

Vishwaa I

Shri Vengadalakshmi S R

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 andmanual surveillance and various such exhaustive and dangerous method.

REFERENCES:

i. Mr.Pranavshita, M.Jayeshredij, Mr.Shikhar Singh, Mr.DurveshZagade, Dr. Sharada Chougule. Department of ELECTRONICS AND TELECOMMUNICATION ENGINEERING, Finolex Academy of Management and technology, ratangiri, India.

ii. 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.

iii. Mr.P.Venkateswara Rao, Mr.Ch Shiva Krishna ,MR M Samba Siva ReddyLBRCE,LBRCE,LBRCE.

iv. Mohit Korche,SarthakTokse, ShubhamShirbhate, Vaibhav Thakre,S. P. Jolhe(HOD). Students , Final Year,Dept.of Electrical engineering,Government

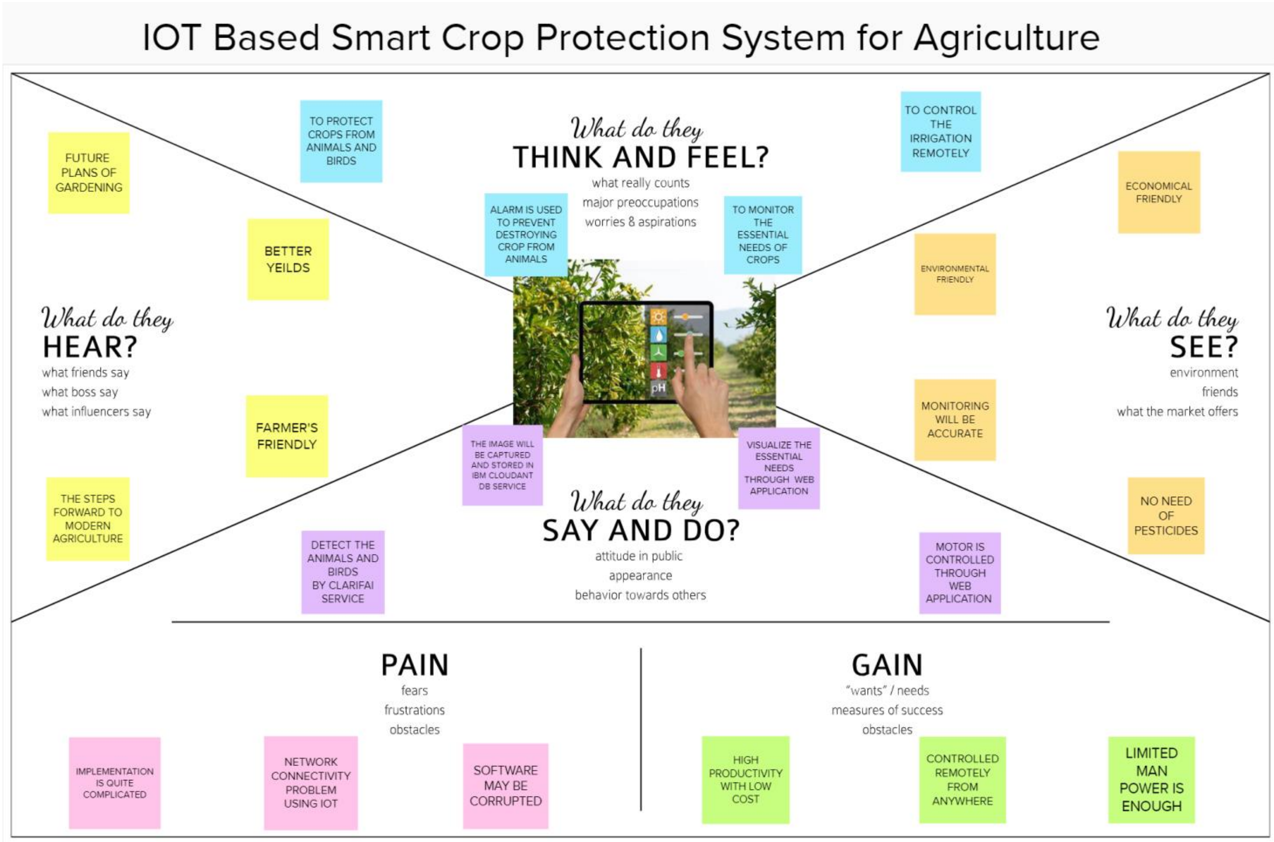
College of engineering,Nagpur head of dept.,Electricalengineering,Government College of engineering,Nagpur.

PROBLEM STATEMENT DEFINITION STATEMENT:

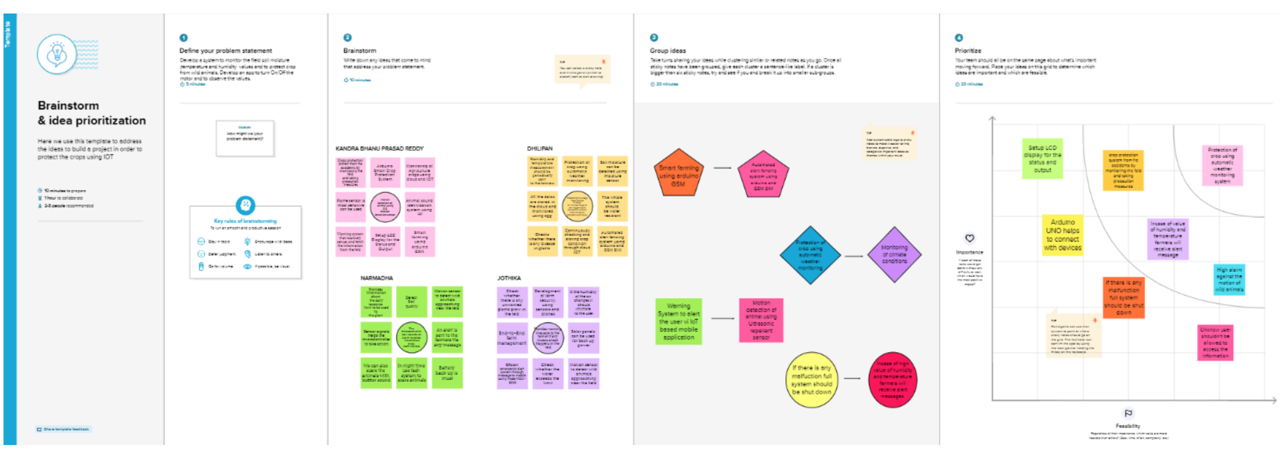
In the world economyof many Countriesdependent upon the agriculture. In spite of economic development agriculture is the backbone of the economy. Crops in forms are many times ravaged by local animals like buffaloes, cows, goats, birds etc. this leads to huge loss for the farmers.it is not possible for farmers to blockade to entire fields or stay 24 hours and guard it. Agriculture meets food requirements of the people and produces several raw materialsfor 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

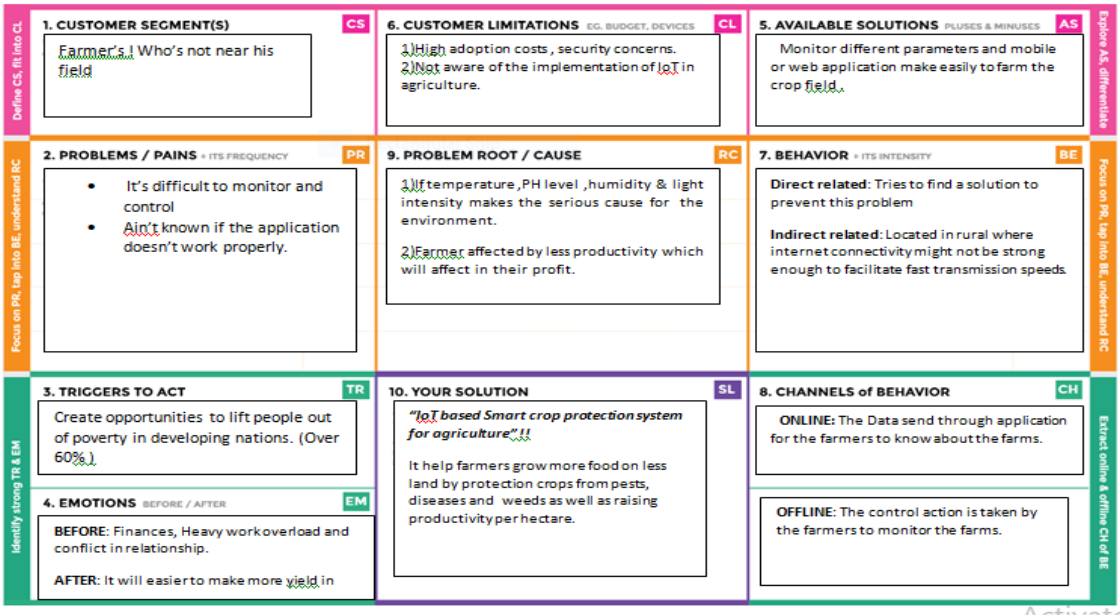
EMPATHY MAP CANVAS:



IDEATION AND BRAINSTORMING:



PROPOSED SOLUTION:



PROBLEM SOLUTIONFIT:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Usually crops in the fields are protected against birds and other unknown disturbances by humans. This take an enormous amount of time.Creating a smart automatic system will benefit the farmers in many different ways.
2.	Idea / Solution description	Smart Farming has enabled farmers to reduce waste and enhance productivity with the help of sensors (light, humidity, temperature, soil moisture,etc..) . Further with the help of these sensors, farmers can monitor the field conditions from anywhere.
3.	Novelty / Uniqueness	Role of SENSORS : IOT smart agriculture products are designed to help monitor crop fields using sensors and by automating irrigation systems. As a result, farmers and associated brands can easily monitor the field conditions from anywhere without any hassle.
4.	Social Impact / Customer Satisfaction	Water conservation . Saves lot of time . Increased quality of production. Real time data and production insight. Remote monitoring.
5.	Business Model (Revenue Model)	
6.	Scalability of the Solution	Scalability in smart farming refers to the adaptability of a system to increase the capacity , the number of technology devices such as sensors and fluctuators.

REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT:

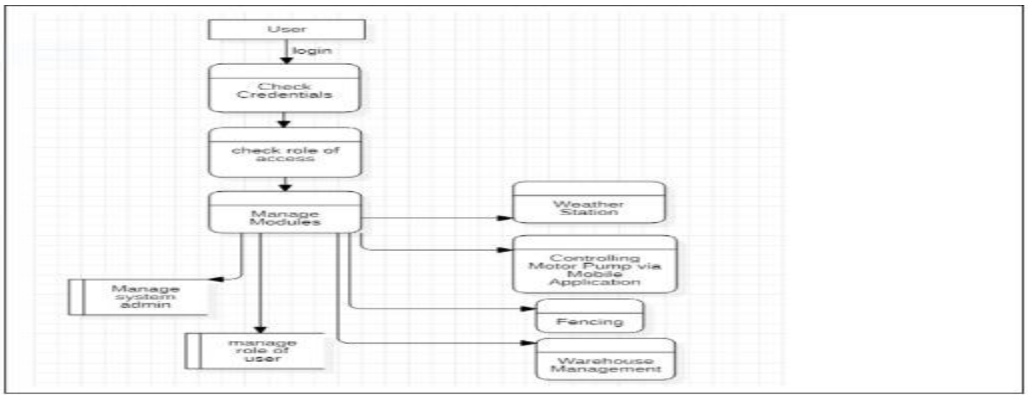
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Install the app. Signing up with Gmail or phone number Creating a profile. Understand the guidelines.
FR-2	User Confirmation	Email or phone number verification required via OTP.
FR-3	Accessing datasets	Data's are obtained by cloudant DB.
FR-4	Interface sensor	Connect the sensor and the application When animals enter the field , the alarm is generated.
FR-5	Mobile application	It is used to control motors and field sprinklers.

NON FUNCTINAL REQUIREMENT:

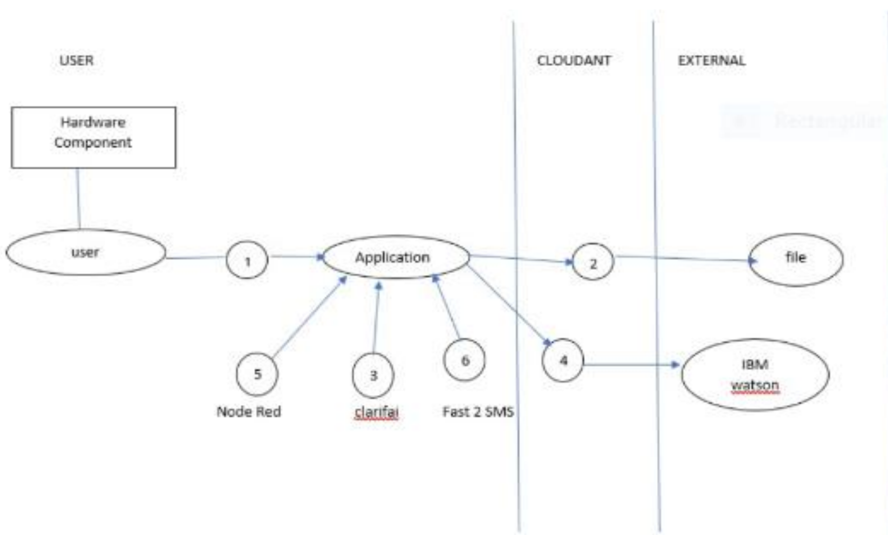
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This project's contributes the farm protection through the smart protection system.
NFR-2	**Security**	It was created to protect the crops from animals.
NFR-3	**Reliability**	Farmers are able to safeguard their lands by help of this technology. They will also benefits from higher crop yields, which will improve our economic situation.
NFR-4	**Performance**	When animals attempt to enter the field, IOT devices and sensors alert the farmer via message.
NFR-5	**Availability**	We can defend the crops against wild animals by creating and implementing resilient hardware and software.
NFR-6	**Scalability**	This system's integration of computer vision algorithms with IBM cloudant services makes it more efficient to retrieve photos at scale, enhancing scalability.

PROJECT DESIGN

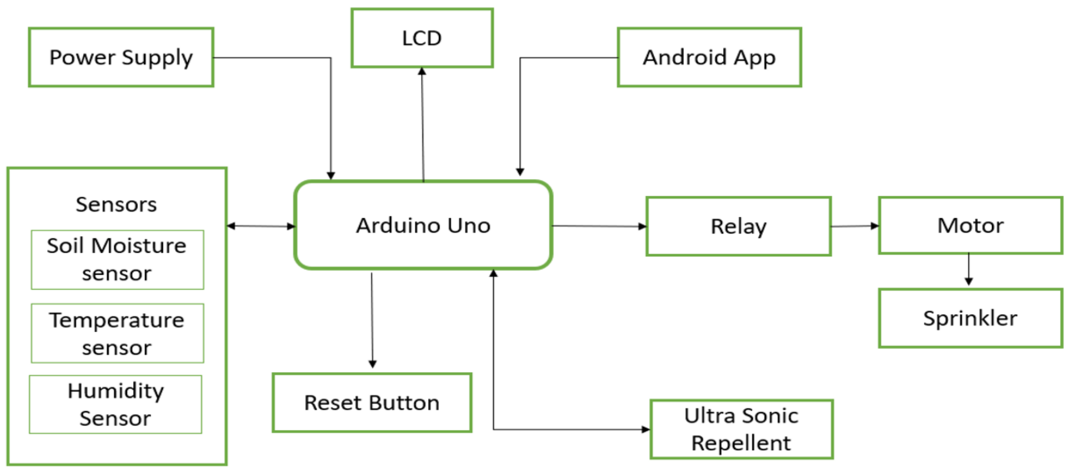
DATA FLOW DIAGRAM:

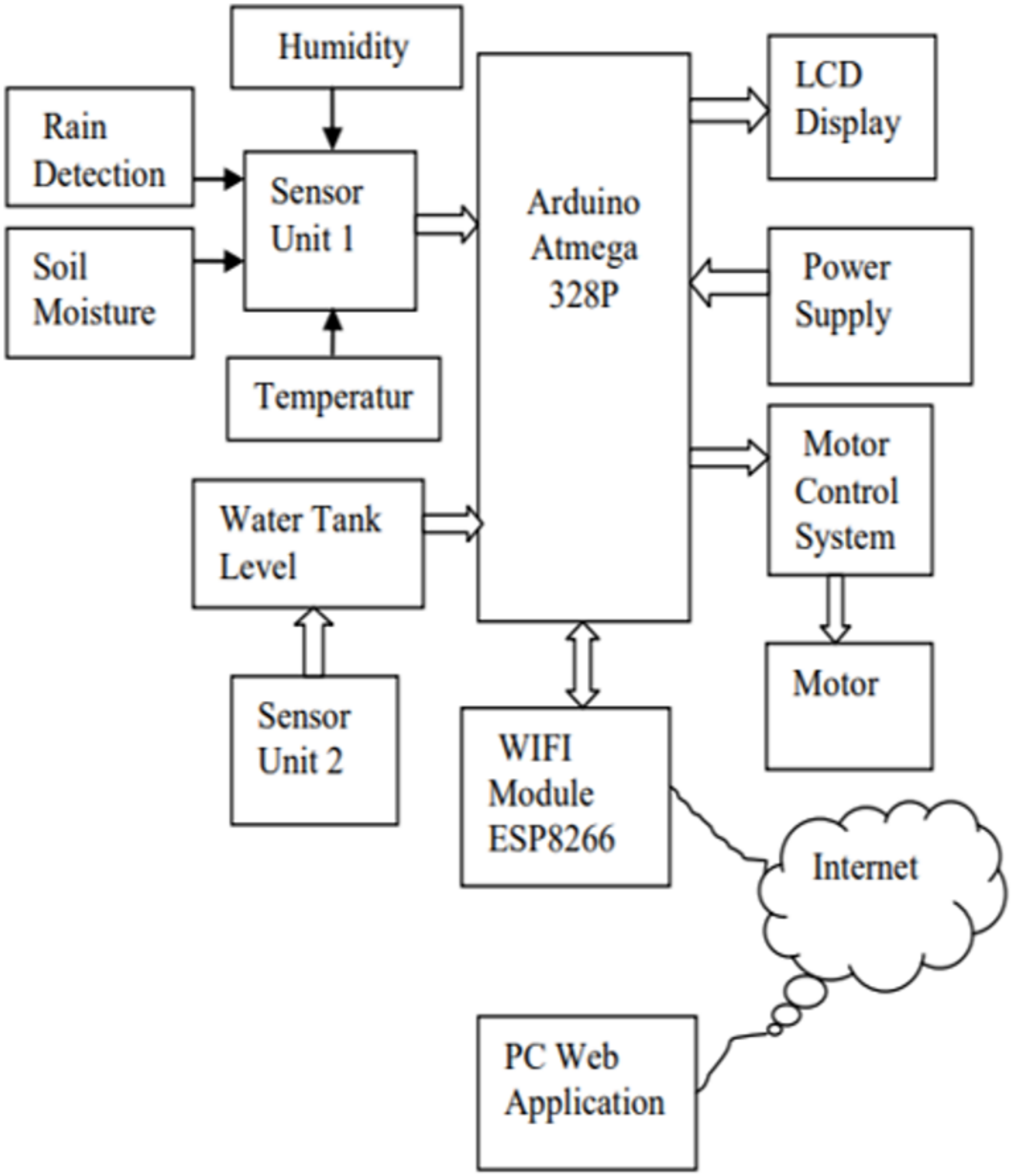


SOLUTION AND TECHNICAL ARCHITECTURE:



Hardware Block Diagram




Humidity
Rain Detection
Sensor Unit 1
Soil Moisture
Temperatur
Arduino Atmega 328P
LCD Display
Power Supply
Motor Control System
Motor
Water Tank Level
Sensor Unit 2
WIFI Module ESP8266
Internet
PC Web Application

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g., Mobile Application	HTML, CSS, JavaScript / Angular JS / Node Red.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	IoT Model	Purpose of IoT Model is for integrating the sensors with a user interface.	IBM IoT Platform
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

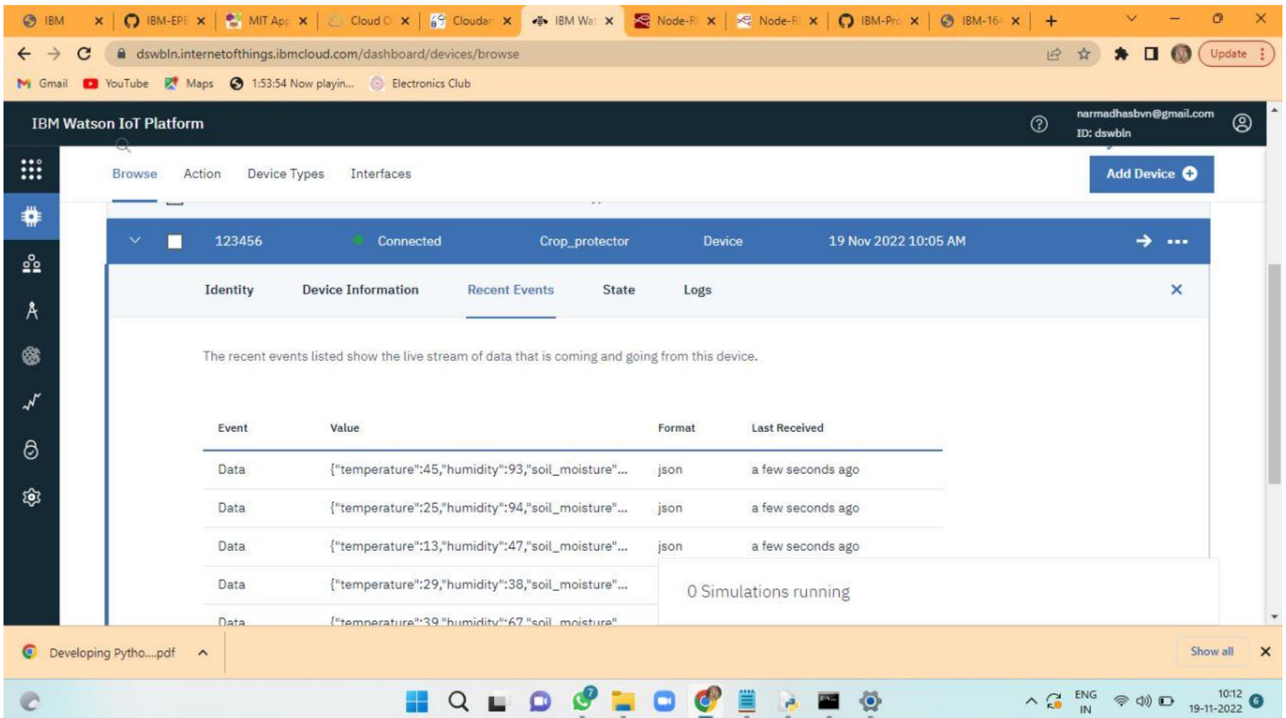
USER STORIES:

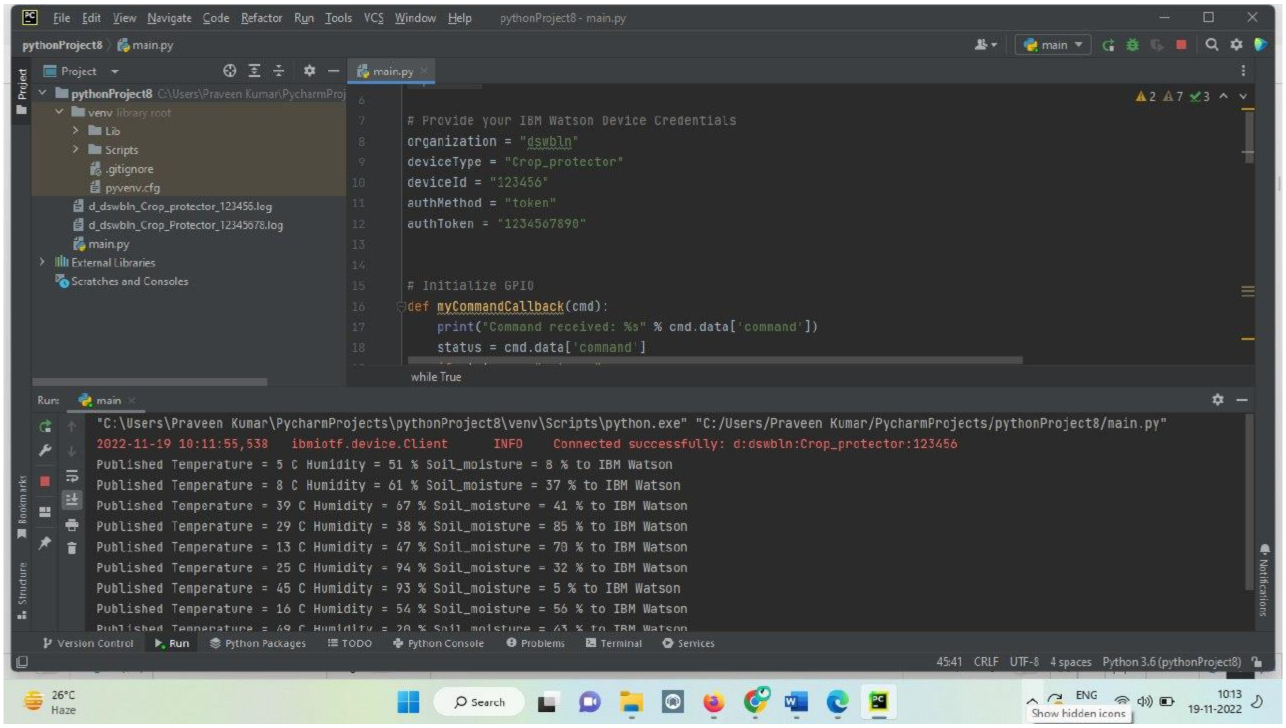
Sprint 1

```
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}
```

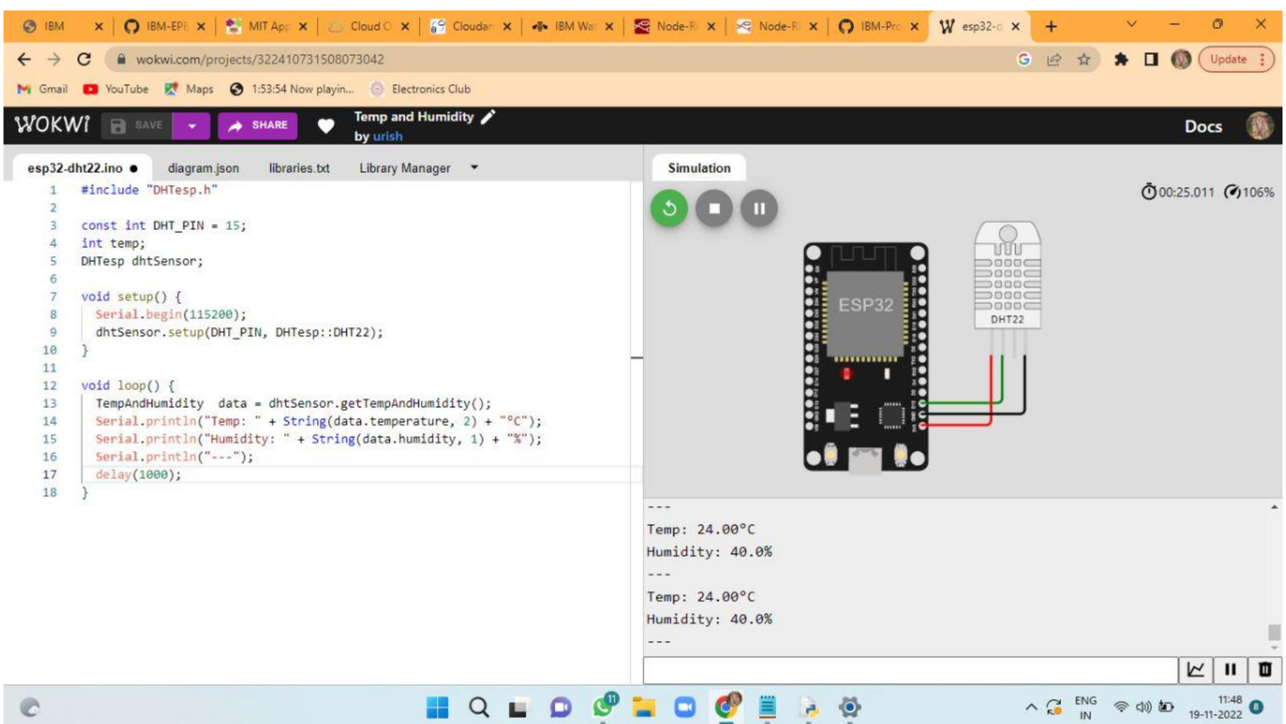
```
deviceCli = ibmiotf.device.Client(deviceOptions)
# ..............................................

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

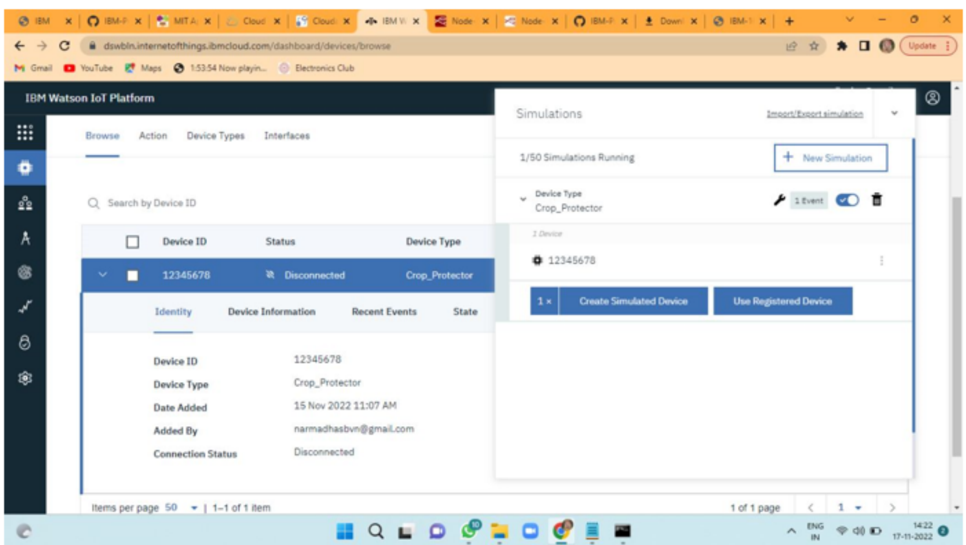
Description:

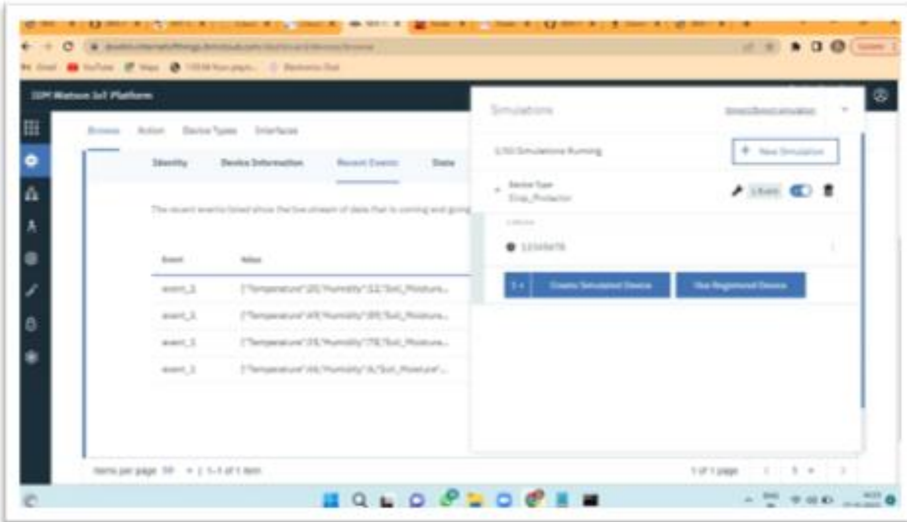
Receiving the data from IOT devices (python script) to IBM Watson IOT platform, hence devices are created and credentials are provided in python script and Output is viewed.

Device Credentials:

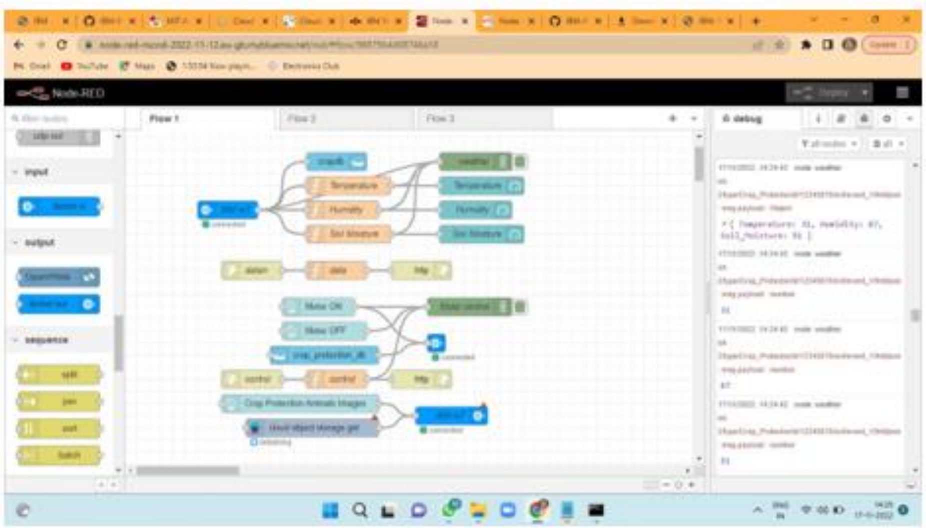
Organization ID: dswbln
Device Type: Crop_Protector
Device ID: 12345678

Output: IBM IOT Platform



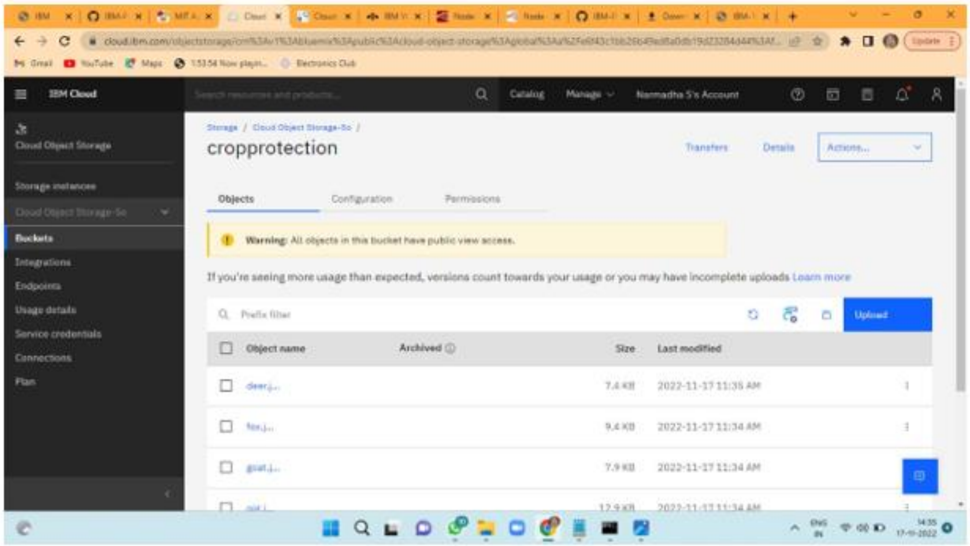


Node Red

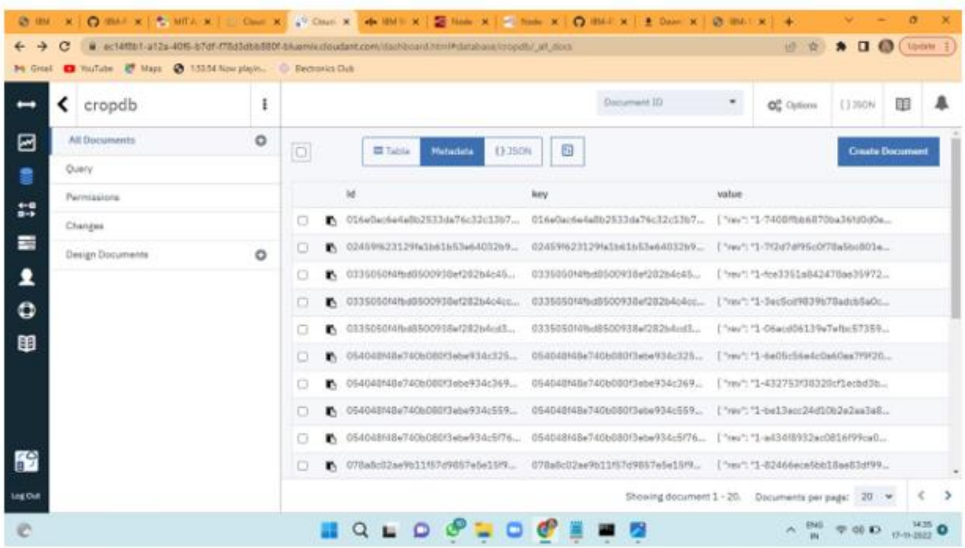


SPRINT 3

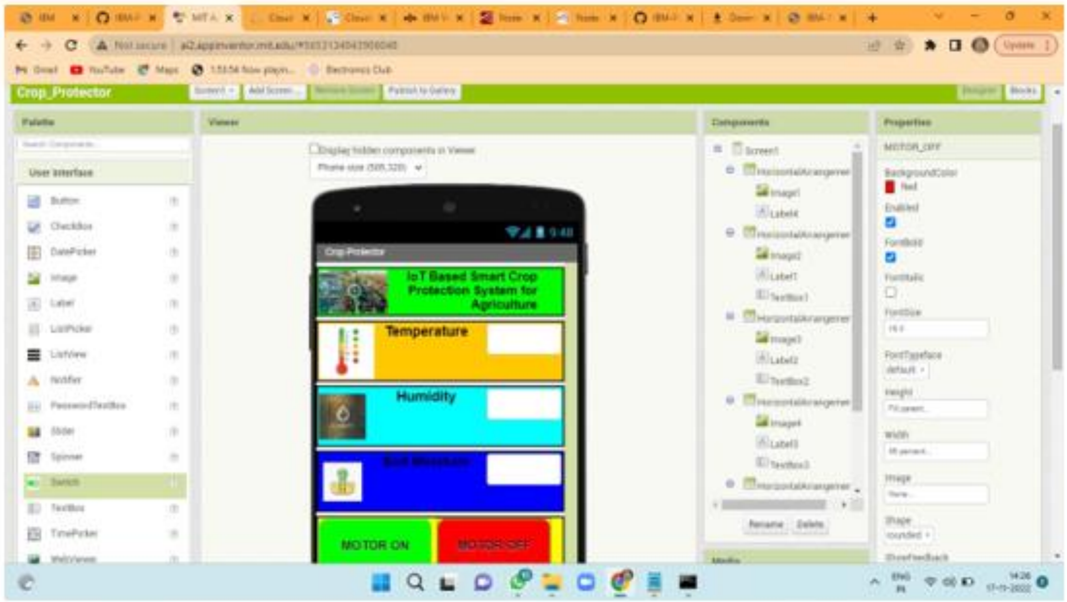
BUCKET CREATION



OBJECT STORAGE CLOUDANT -DB



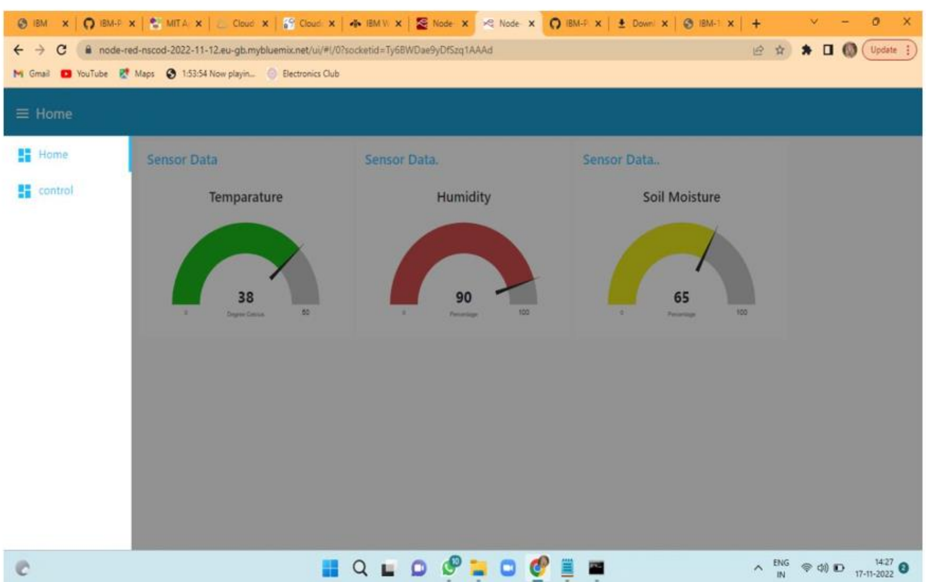
MIT DESIGN CREATION

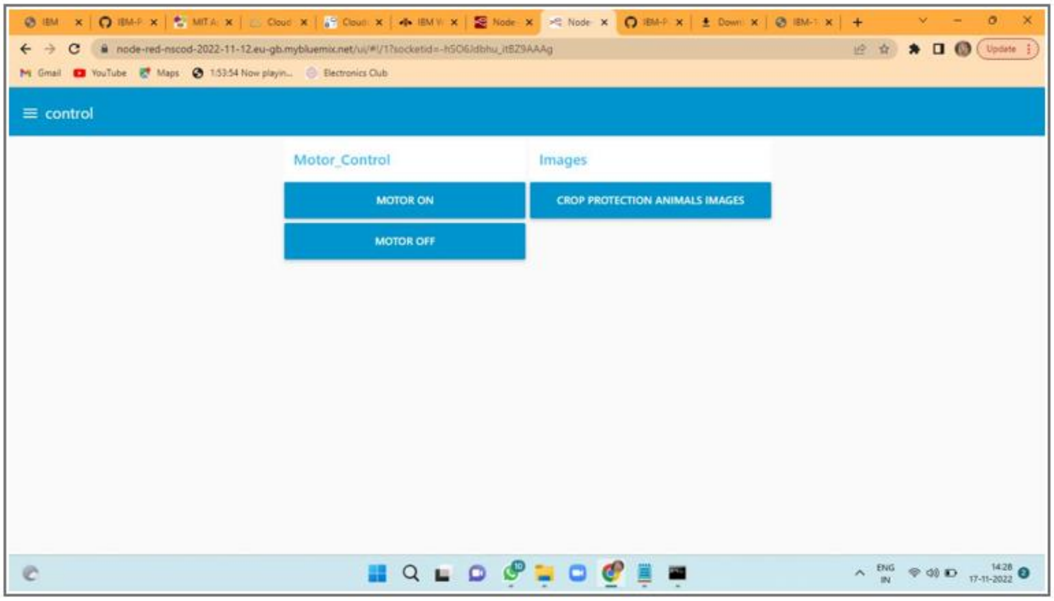


MIT BLOCK CREATION

Sprint 4

WEB UI OUTPUTS





PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND 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	24Oct 2022	29 Oct 2022	20	29 Nov 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	19ss 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
```

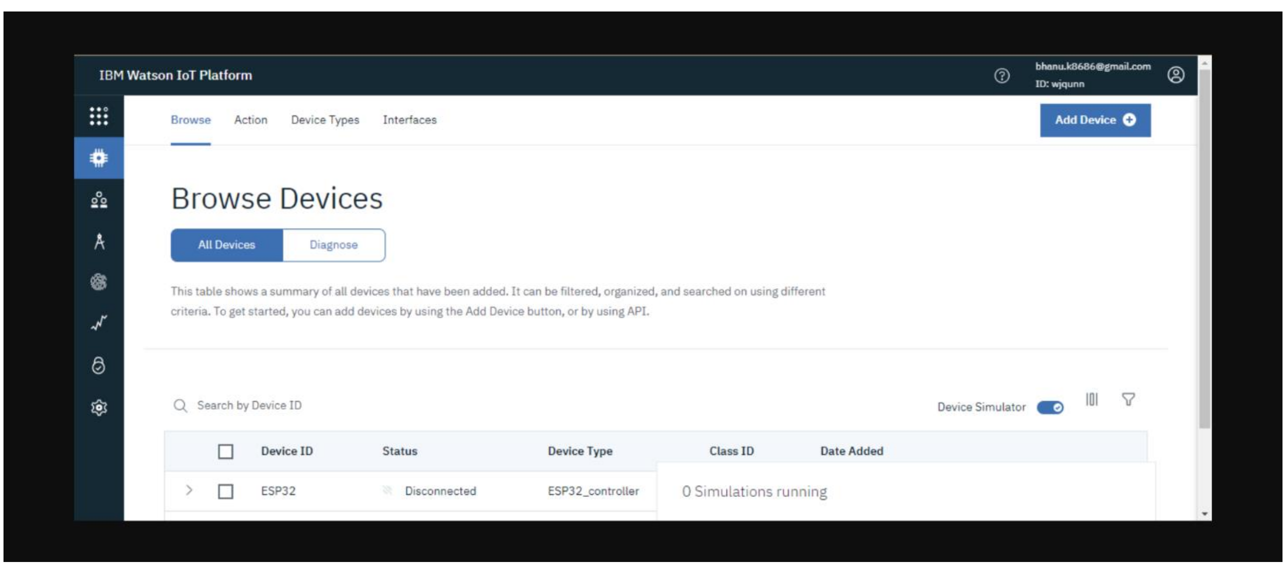
```
#CloudantDB

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
```

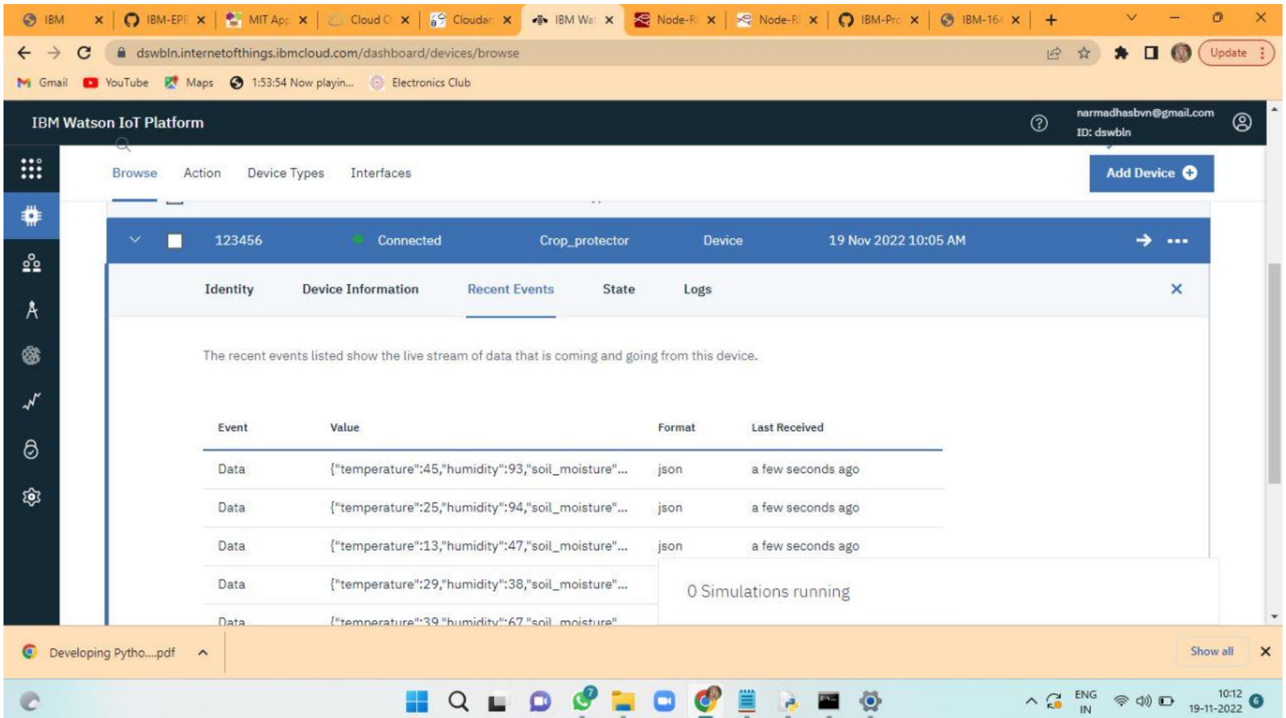
```
                    )
def = multi_part_upload(bucket_name, item_name, file_path):
   try:
print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
      #set 5 MB chunks
part_size = 1024 * 1024 * 5
      #setthreadhold to 15 MB
file_threshold = 1024 * 1024 * 15
      #set the transfer threshold and chunk size
transfer_config = ibm_boto3.s3.transfer.TransferConfig(
multipart_threshold=file_threshold,
multipart_chunksize=part_size
         )
      #theupload_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

- RatedVoltage: 6V DC
- Operating Voltage: 4 to 8V DC
- Rated Current*: ≤30mA
- SoundOutput 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 vehiclessuch as ambulances, police cars and fire trucks. There are two general types, pneuma c and electronic.

FEATURE-2:

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

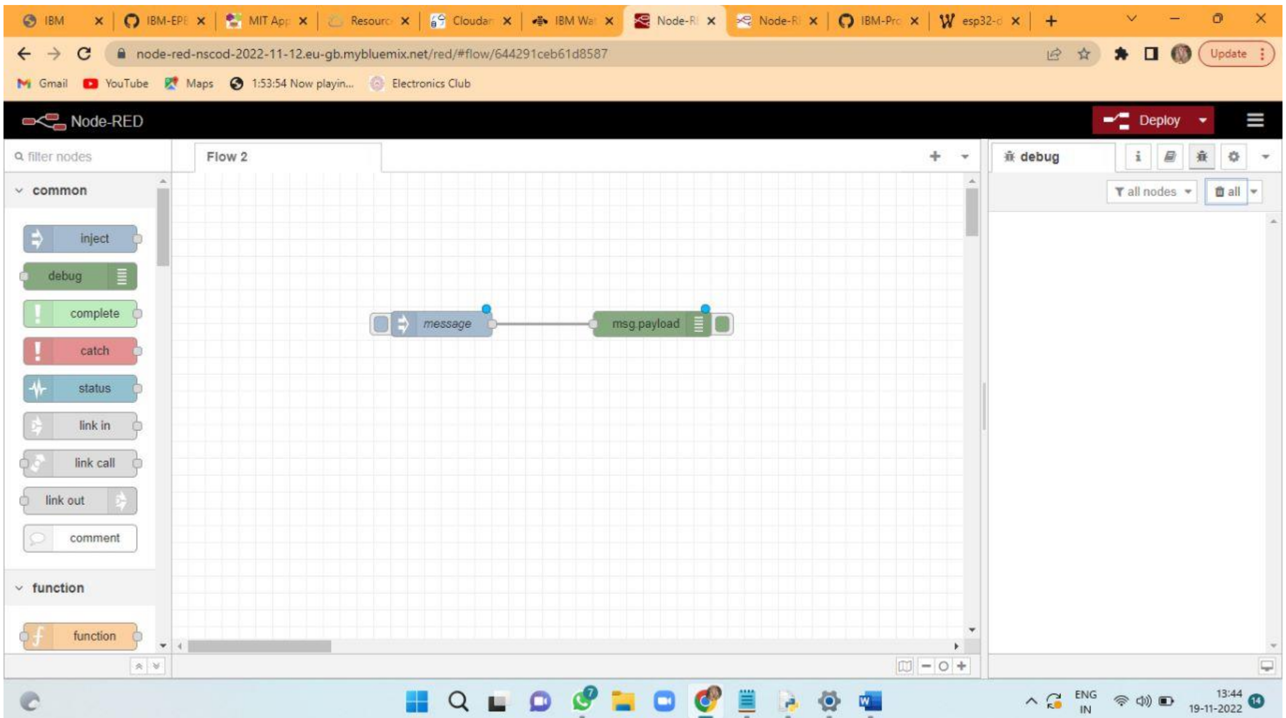
TESTING

TEST CASES:

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 achievingbetter crop yields thus leading to theireconomic 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, 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 can be detected by cameras and if it comes towards farmthen system will be directly activated through wireless networks. Wild animals can also be detected by using wireless networks such as laser wireless sensors and by sensingthis laser or sensor's security system will beactivated.

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-38676-1660384301