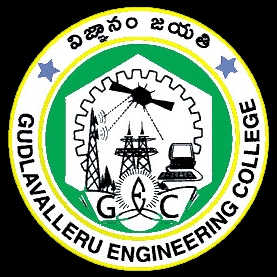
**GUDLAVEALLERU ENGINEERING COLLEGE**

**INFORMATION TECHNOLOGY**

**PROJECT NAME :** INTELLIGENT CROP PROTECTION USING IBM CLOUD

SUBNITTED BY:IT\_B15

M.DURGA MAHESH(18481A1270)

T.CHANDAN(18481A1294)

T.RAVI KIRAN(18481A1295)

V.BHARGAVI(18481A12A6)

**INDEX**

**TITLE**

**1. INTRODUCTION**

1.1 OVERVIEW

1.2 PURPOSE

**2. LITERATURE SURVEY**

2.1 EXISTING PROBLEM

2.2 PROPOSED SOLUTION

**3. THEORETICAL ANALYSIS**

3.1 BOCK DIAGRAM

3.2 SOFTWARE DESIGNING

**4. EXPERIMENTAL INVESTIGATIONS**

**5. FLOWCHART**

**6. RESULT**

**7. ADVANTAGES & DISADVANTAGES**

**8. APPLICATIONS**

**9. CONCLUSION**

**10. FUTURE SCOPE**

**11. APPENDIX**

11.1 SOURCE CODE

11.2UI OUTPUT SCREENSHOT

**1.INTRODUCTION**

**1.1 Overview**

Various methods aim only at surveillance which is mainly for human intruders, but we tend to forget that the main enemies of such farmers are the animals which destroy the crop. The problems of wild life attack on crops i.e., crop Vandalization is becoming very common in the states of Tamil Nadu, Himachal Pradesh,Punjab,Harayana,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 vandalization them. This problem is so pronounced that sometimes the farmers decide to leave the areas barrendue to such frequent animals attack. This system helps us to keep away such wild animals form the farmlands and it is also preventing the loss of crops**.**

* 1. **PURPOSE**

The purpose of this project is to help the farmers in protecting the crop from the animals which destroys the crop, and also helps farmers to monitor the soil moisture levels in the field and humidity values near the field.

**2. LITERATURE REVIEW**

**2.1 Existing Problem**

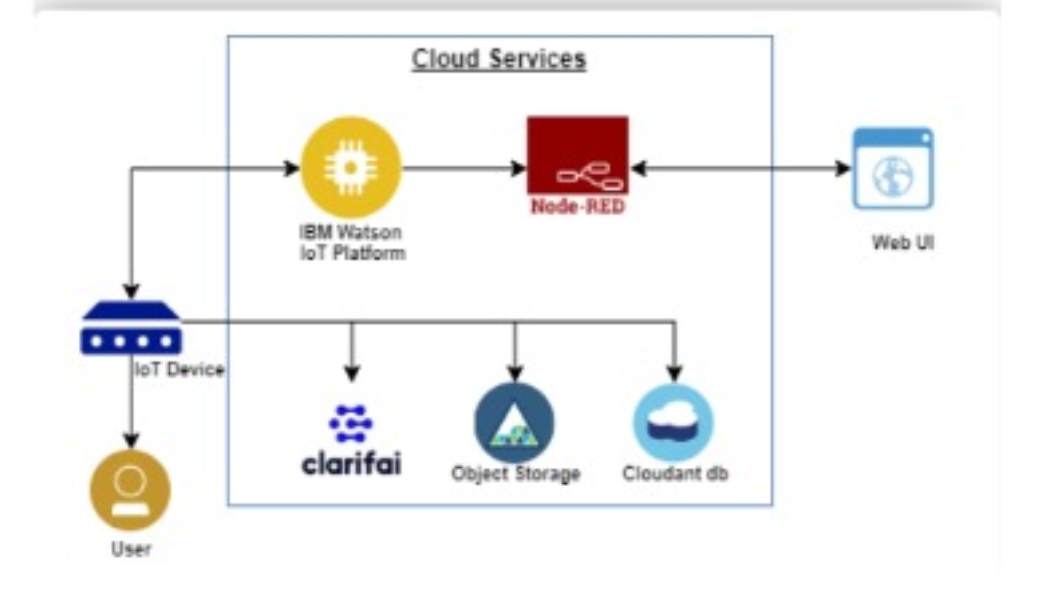
The existing system mainly provide the surveillance functionality, these systems do not provide protection from wild animals, especially in such an application area they also need to take actions based on the type of animals that tries to enter the area, as different methods are adopted to prevent different animals from entering such restricted areas. Also the farmers resort to the other methods by erecting human puppets and effigies in their farms, which is ineffective in warding off the wild animals, though is useful to some extend to ward off birds. The other commonly used methods by the farmers in order to prevent the crop vandalization by animals include physical barriers, use of electric fences

and manual surveillance and various such exhaustive and dangerous methods.

**2.2 Proposed Solution**

* By Designing a Security System for Farm Protection the farmers can do their work easily.
* By using Security System which can detect the animals we can prohibit the entry of animals into the farm.
* By using clarifai service we can detect the animal images.
* We should design a system that sounds when animals tries to enter into the farm.

**3.THEORETICAL ANALYSIS**

**3.1 BLOCK DIAGRAM**

**3.2 SOFTWARE DESIGNING**

* Code for Text to Speech
* Code for IBM Watson Assistant

**Tools Used**

* NODE-RED
* IBM WATSON
* IBM CLOUD OBJECT STORAGE
* CLOUDANT DB
* PYTHON IDLE

**4.EXPERIMENTAL INVESTIGATIONS**

Initially we start our project by creating node connections in NODE-RED. In order to operate the motor and light we use four different buttons. In order to connect it to ibm iot we connect four buttons to ibm iot out node.To get the image of animal from database we used a http request node,To view the image of that particular animal we used template node. To get the humidity and moisture values . Humidity function node is connected to the guage node. The another http request node is used to get the messege.

The next step is to write the python code to integrate it with node-red through ibm Watson.For integration we have to use ibm Watson credentials in python code.And we have to integrate ibm Watson code with the text to speech code in order to get the voice out command.

**5.FLOWCHART**

**Display humidity and moisture values**

**Play alert sound &print animal detected . show animal image on nodeui and python o/p**

**Check whether animal is present or not**

No

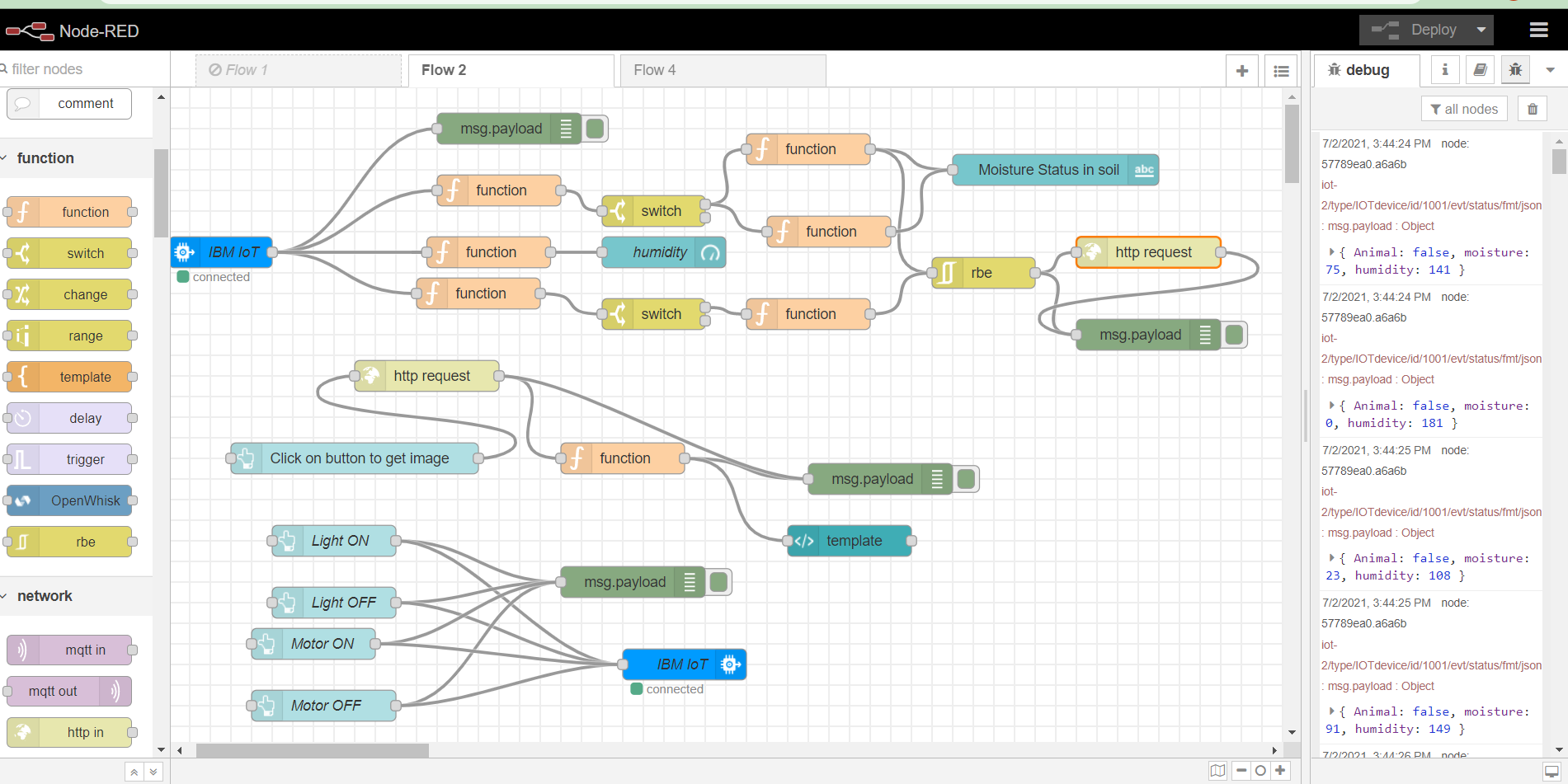
Yes

**Disply garden frame in python output**

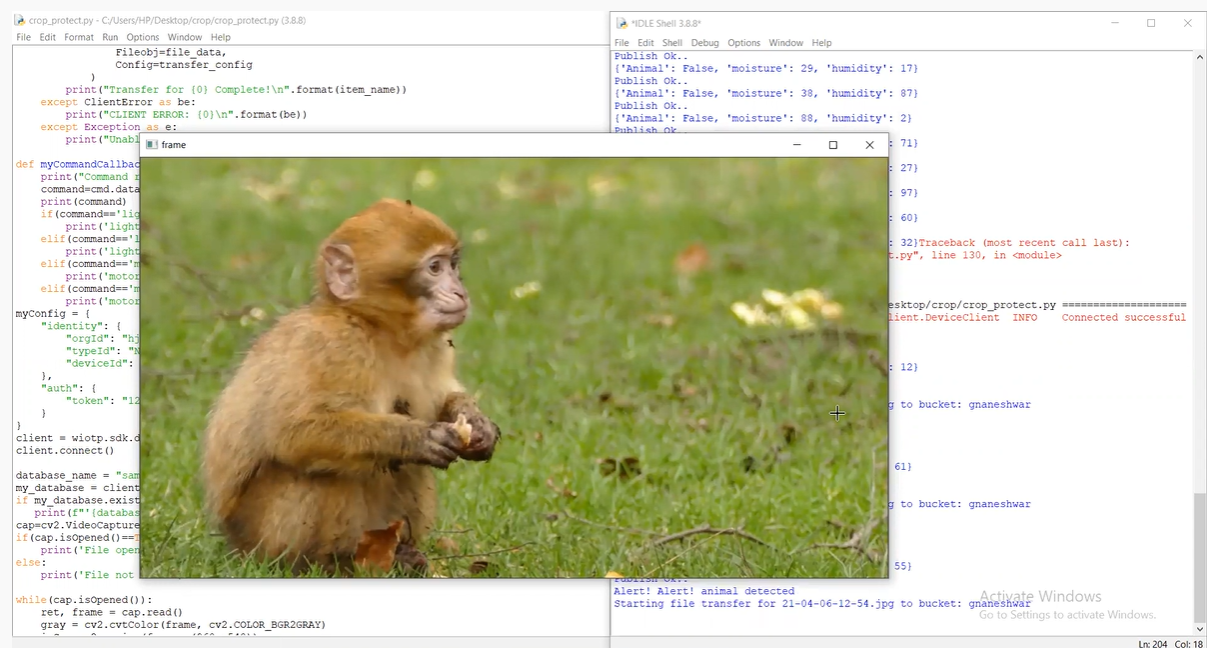
**6.RESULT**

The Result for smart notice board can be obtained by connecting NODE-RED with python idle through IBM Watson. The Web app of node-red is used to give commands to python shell. The python shell take the commands from node-red and display them and corresponding voice out command are also obtained. The commands that displayed on python shell are also stored in cloudant database

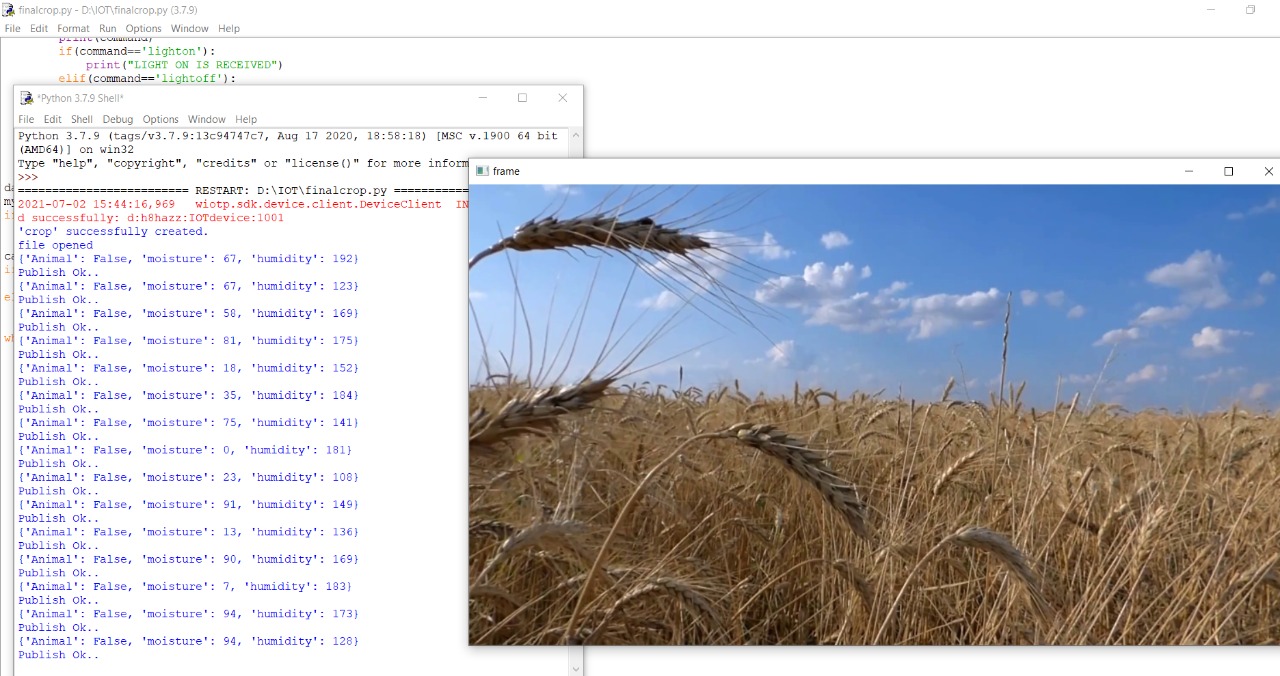
**NODE -RED**



**Python Shell Output**

If the animal present in the crop, the output will be:

If there there is no animal in the field the output will be:



**7.Advantages and Dis-advantages**

**Advantages**

* In Smart Agriculture we use security system for farm protection
* It helps to prohibit the entry of animals into the farm
* We can detect the animal images by clarify service.
* The system will plays an alert sound, so we can prohibit the animal entry to the field.
* We can monitor the moisture values in the filed and humidity values near the field.

**Dis-advantages**

* When we use hardware components to integrate

The crop protection, during power failures it does not work.

* Clarifai service cannot detect the any images like birds rather than animals.

**8.APPLICATIONS**

* Used to detect animals when they enter into the farm
* Design a Security system for farm protection
* Used to monitor the humidity and moisture levels in the field

**9.Conclusion**

By the smart agriculture we can save the farm from animals by detecting them easily by IBM cloud. By this farmers can protect their farm and prohibit the entry of animals into the farm.We can monitor the humidity values and also the moisture levels in the field.

**10.Future Scope**

In the future. there will be very large scope, this can b l e made based on wireless networks. Wireless network and different types of networks are used to collect the information of crop conditions and environmental change and these in-formation is transmitted through network to the farmers that initiates corrective sections.

Farmers are connected and aware of the conditions of the agriculture field at anytime and anywhere in the world.

**11.APPENDIX**

**11.1 SOURCE CODE**

import cv2

import numpy as np

import wiotp.sdk.device

import playsound

import random

import datetime

import ibm\_boto3

import time

from ibm\_botocore.client import Config, ClientError

#cloudant db

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, resources\_pb2

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

myConfig={

"identity":{

"orgId":"h8hazz",

"typeId":"IOTdevice",

"deviceId":"1001"

},

"auth":{

"token":"1234567890"

}

}

clientx=wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)

clientx.connect()

# This is how you authenticate.

metadata = (('authorization', 'Key 83727a951ba743a7a9cdb5ef43b20ab1'),)

COS\_ENDPOINT= "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS\_API\_KEY\_ID="SMJxiYSCZMfvBCZGmziBjNWBwIUMKlkLWiq7kPq5pgtk"

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

COS\_RESOURCE\_CRN="crn:v1:bluemix:public:cloud-object-storage:global:a/ec51d848d56a4315a195eac64a73a058:dd8894be-a232-43f4-b2c1-a9edefc8d3ac::",

clientdb=Cloudant("apikey-v2-23p0i2j8e7g6gkps50q3po7a58yngzigkfheraedl10u","e15afd6e9f04035a381a83b620a6f287",

url="https://apikey-v2-23p0i2j8e7g6gkps50q3po7a58yngzigkfheraedl10u:e15afd6e9f04035a381a83b620a6f287@7bb92285-7884-4652-9c40-ddabe949b4fc-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 for all files over 15 MB

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)

if(command=='lighton'):

print("LIGHT ON IS RECEIVED")

elif(command=='lightoff'):

print("LIGHT OFF IS RECEIVED")

elif(command=='motoron'):

print("MOTOR ON IS RECEIVED")

elif(command=='motoroff'):

print("MOTOR OFF IS RECEIVED")

database\_name="crop"

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

if my\_databse.exists():

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

cap=cv2.VideoCapture('animal.mp4')

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

print("file opened")

else:

print("file not found")

while(cap.isOpened()):

ret,frame=cap.read()

gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

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

detect=False

cv2.imwrite("ex.jpg",imS)

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

file\_bytes=f.read()

#model ID of a publicity available general model

request = service\_pb2.PostModelOutputsRequest(

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

model\_id='aaa03c23b3724a16a56b629203edc62c',

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))

#print(response)

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

#print(concept)

if(concept.value>0.9):

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

print("Alert! Alert! Animal Detected")

playsound.playsound('sound.mp3')

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

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

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

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

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

if new\_document.exists():

print("Document successfully created")

time.sleep(5)

detect=True

moist=random.randint(0,100)

humidity=random.randint(100,200)

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

print(myData)

if(humidity!=None):

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

print("Publish Ok..")

clientx.commandCallback=myCommandCallback

cv2.imshow('frame',imS)

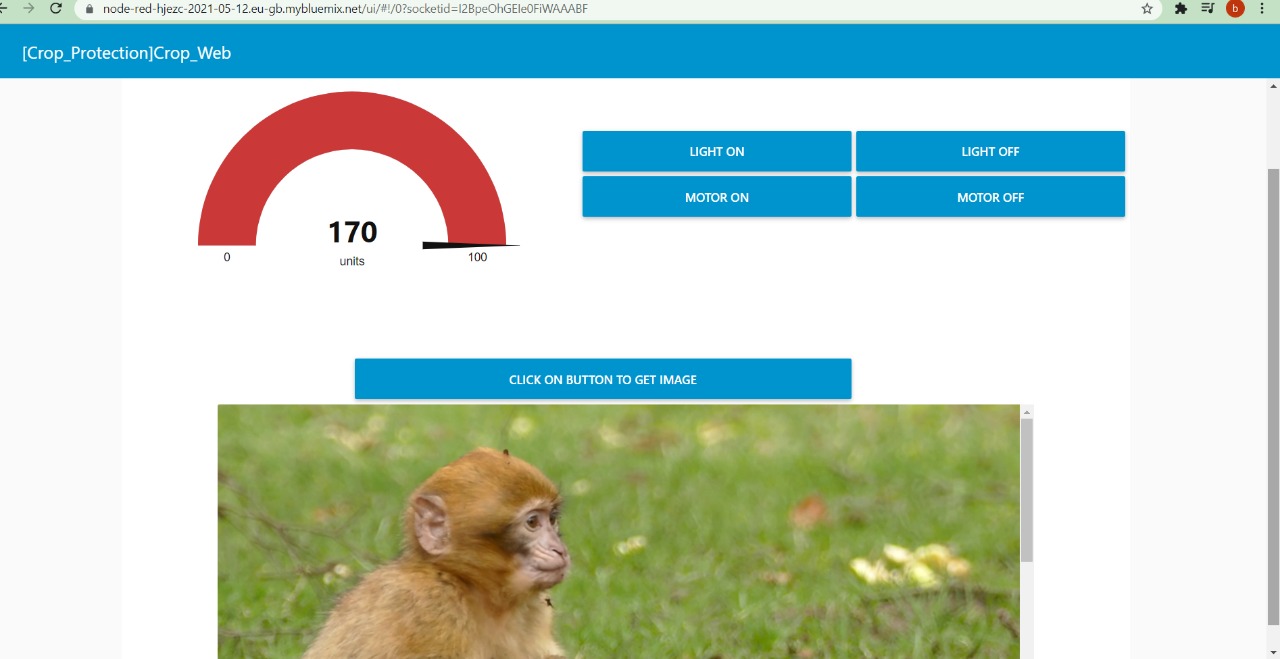
if cv2.waitKey(1) & 0xFF==ord('q'):

break

clientx.disconnect()

cap.release()

cv2.destroyAllWindows()

**11.2 NODE UI OUTPUT SCREENSHOT**