**INTRODUCTION**

***Overview:***

Surveillance plays a major role in many fields be it at home, hospitals, schools, public places, farmlands etc. It helps us to monitor a certain area and prevent theft and also provides proof of evidence. In the case of farmlands or agricultural lands surveillance is very important to prevent unauthorized people from gaining access to the area as well as to protect the area from animals. 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 crops. This leads 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. This system helps us to keep away such wild animals from the farmlands as well as provides surveillance functionality.

***Purpose:***

Our main purpose of project is to develop intruder alert to the farm, to avoid losses due to animals and fire. These intruder alert protect the crop from damaging that indirectly increase yield of the crop.

**LITERATURE SURVEY:**

***Existing problem:***

Wild animals are a special challenge for farmers throughout the world. Animals such as deer, wild boars, rabbits, moles, elephants, monkeys, and many others may cause serious damage to crops. They can damage the plants by feeding on plant parts or simply by running over the field and trampling over the crops. Therefore, wild animals may easily cause significant yield losses and provoke additional financial problems. Another aspect to consider is that wild animal crop protection requires a particularly cautious approach. In other words, while utilizing his crop production, every farmer should be aware and take into consideration the fact that animals are living beings and need to be protected from any potential suffering.

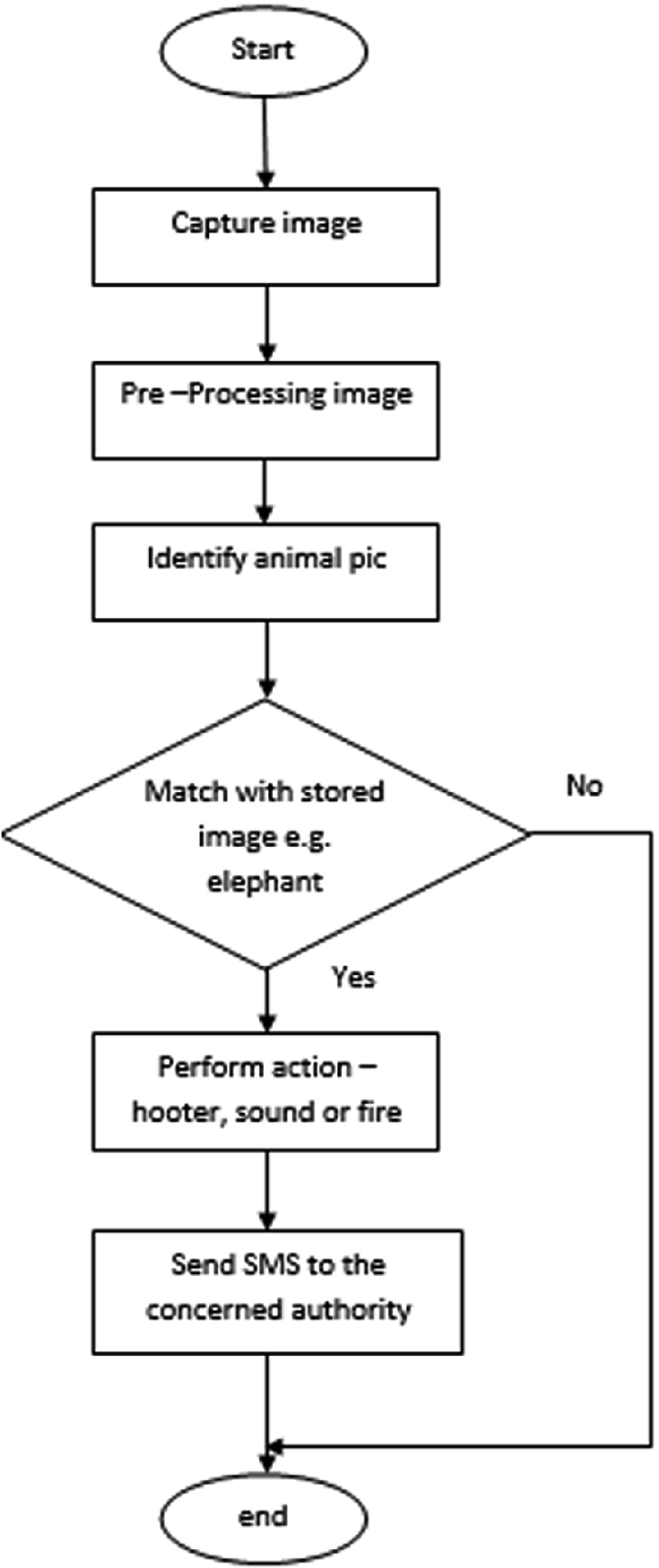
***Proposed solution:***

The development of Internet of Things application for crop protection to prevent animal intrusions in the crop field. A repelling and a monitoring system is provided to prevent potential damages in Agriculture, both from wild animal attacks and weather conditions.

It aims at using a scientific way of irrigation system which is based on moisture content of the soil.

**THEORITICAL ANALYSIS:**

***BLOCK DIAGRAM:***



**Hardware/software designing:**

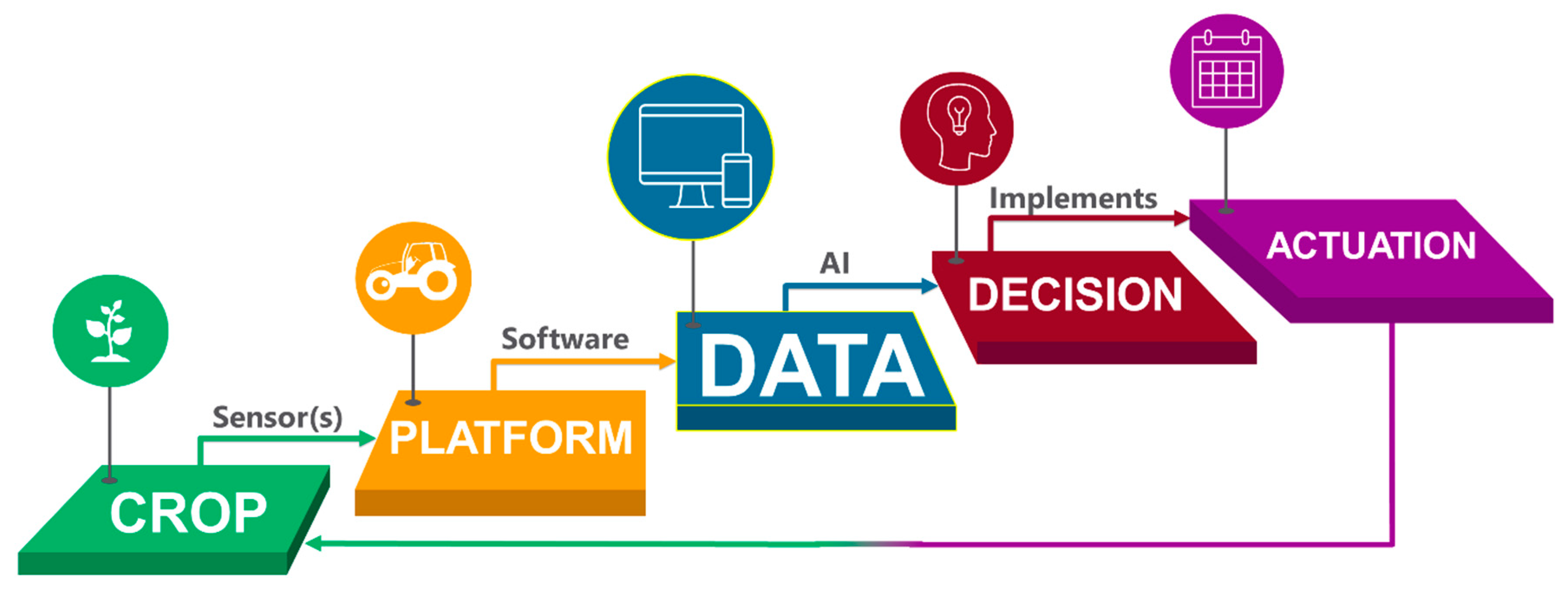
The Software designing involves genera We used IBM Cloud Services to create

Cloud storage. In cloud storage we create a bucket. We use these cloud storage

credentials in Python program then we make use of the Node-Red platform to

display the image.With the help of MIT APP Inverter we designed the app &amp;

integrated with the Node-Red to observe the object.



**EXPERIMENTAL INVESTIGATION:**

To complete our project we have collected required information from internet and other research papers.

After getting the complete knowledge we work according to our

roles in the project. At first we create the IBM Cloud account then we created the

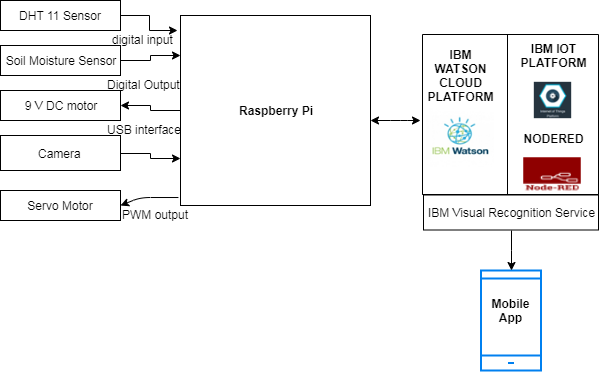
Cloud storage service after we wrote a python code in IDLE to connect Cloud

storage. Next we created the Node-Red Services. This service helps us to show

virtual flow graphs. From Node-Red we send image to the MIT APP. From app we

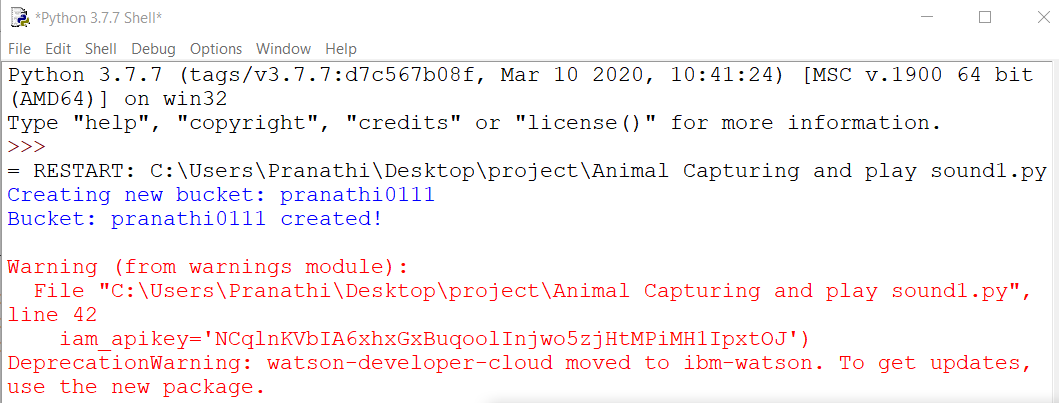
can view the details of the object.

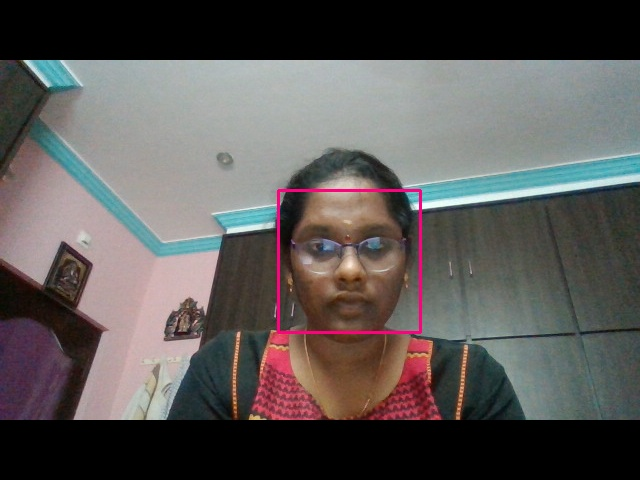
**FLOW CHART:**



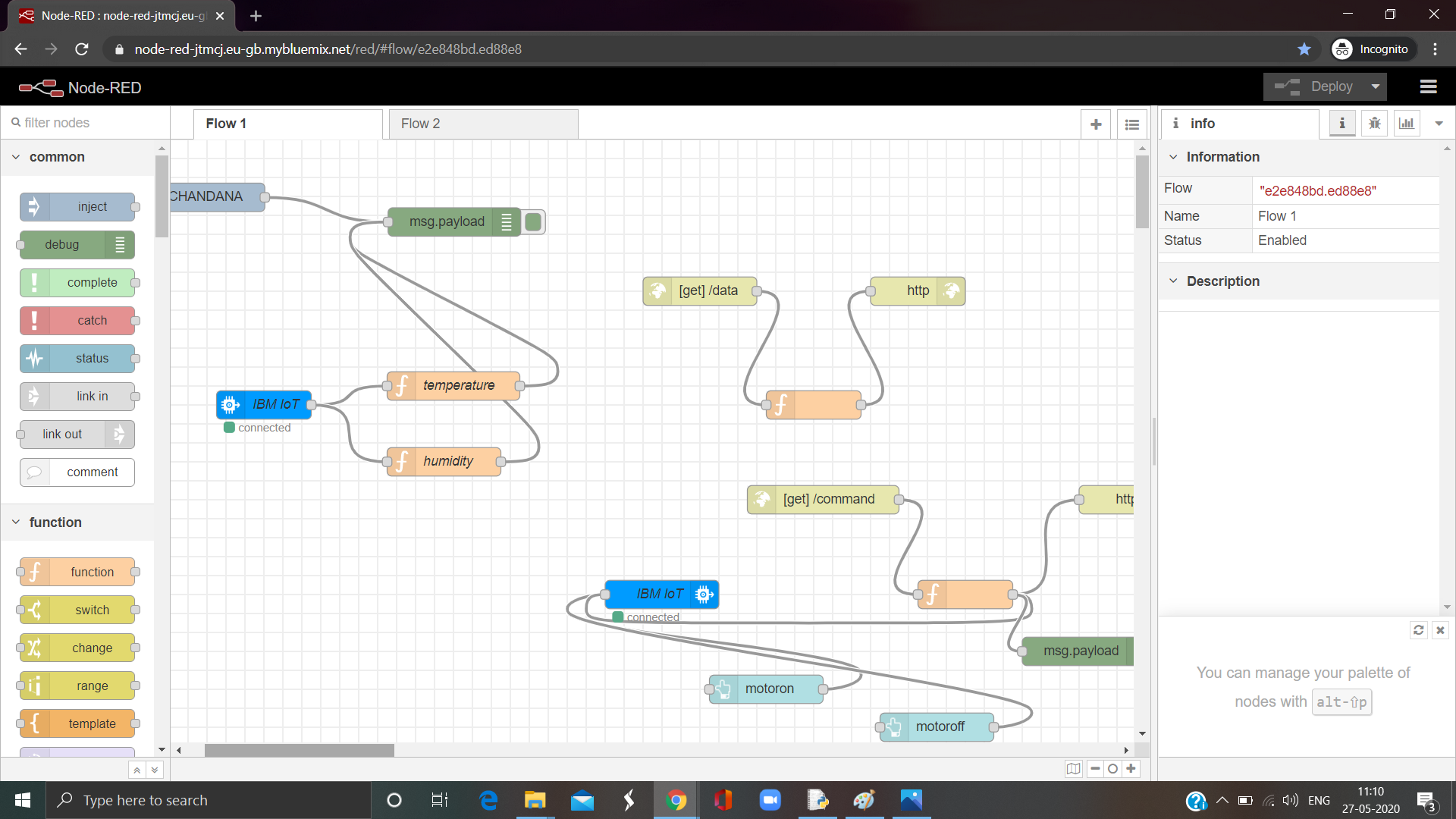
**RESULT:**

***PYTHON CODE:***

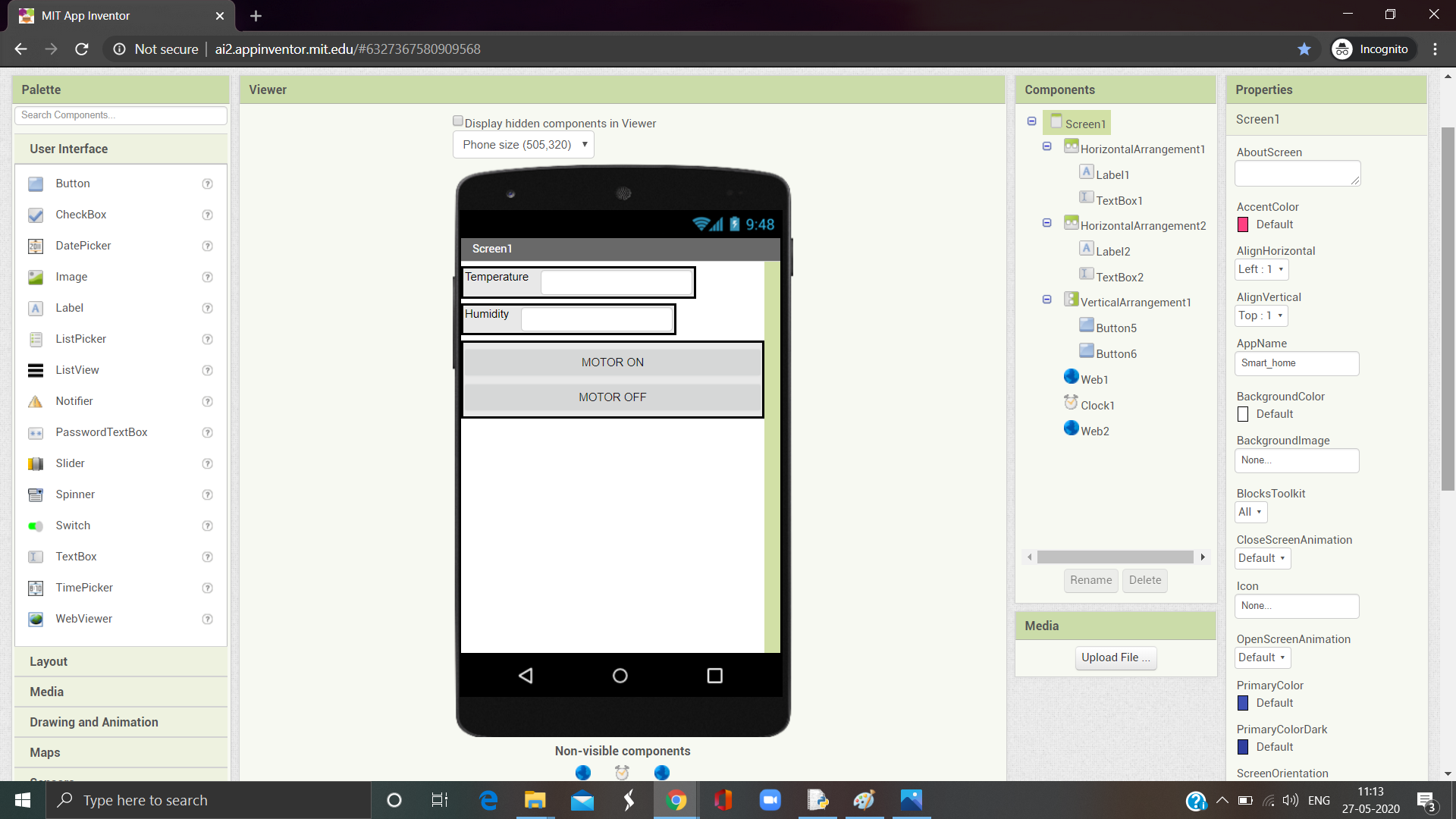


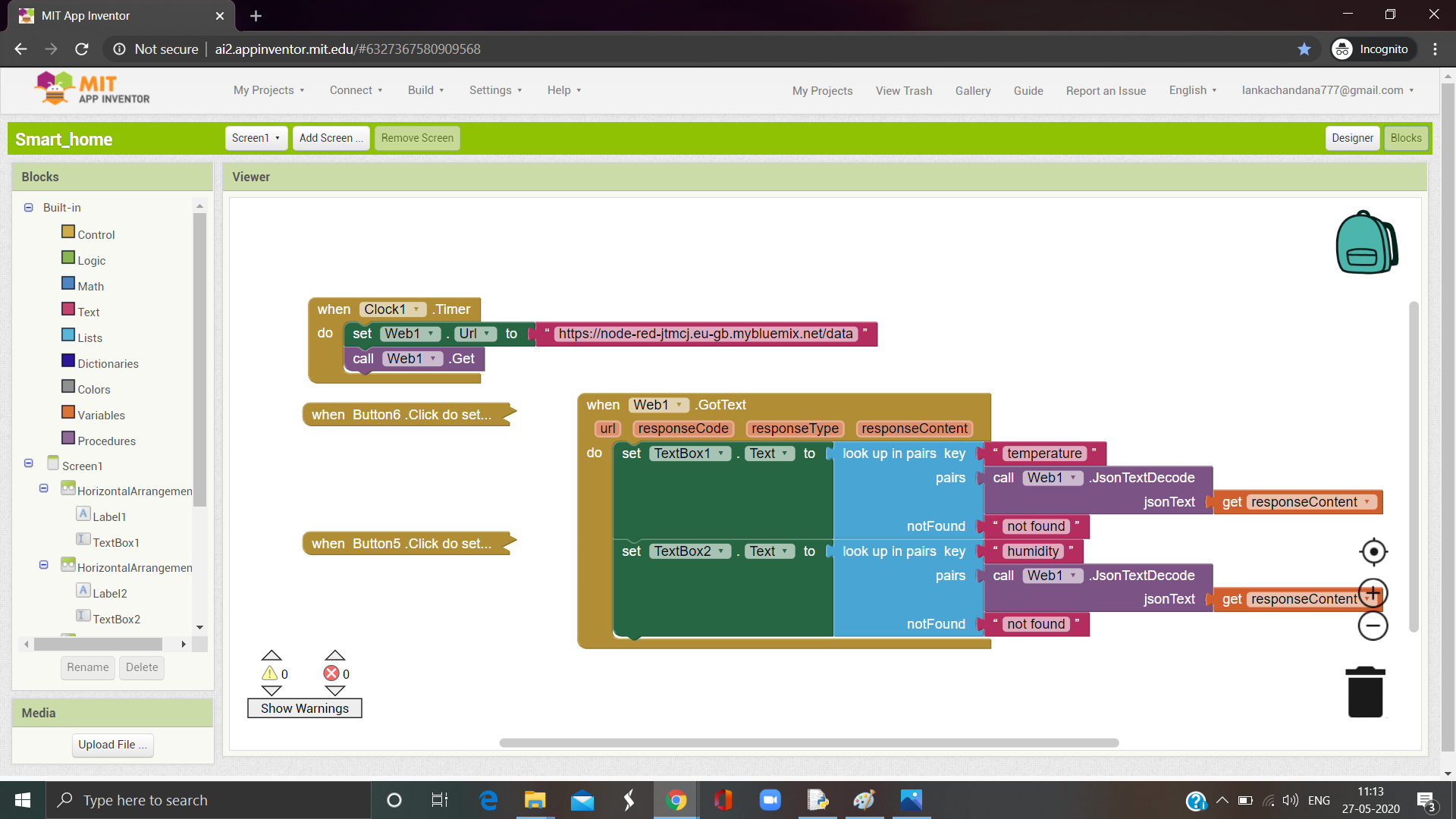


***NODE-RED:***



***MIT-APP INVENTOR:***







**ADVANTAGES AND DISADVANTAGES:**

**Advantages:**

1) Increase ease of control in rural areas

2) Keep track of who comes and goes

3) Protect against animals

4) create safe Environment

5) Reduce Accidents and Deaths

6) Easy Monitoring

**DISADVANTAGES:**

1) Animal detection systems can be hacked and crashes may occur which results to poor results.

**APPLICATIONS:**

1)Farm land at rural areas

2)Food crops

**CONCLUSION:**

By using smart crop protection system we can avoid the damage to the crops from the wild animals.This technique of using camera we can track every movement of aminals and can continuously monitor the crops.The moisture levels are perfectly maintained.Therefore we can prevent crops from animals more effictively and can observe healthy growth in crops.

**FUTURE SCOPE:**

Smart farming is a concept quickly catching on in the agricultural business. IoT sensors capable of providing farmers with information about crop yields, rainfall, pest infestation, and soil nutrition are invaluable to production and offer precise data which can be used to improve farming techniques over time.

**BIBILOGRAPHY:**

<https://cloud.ibm.com/>

<https://node-red-jtmcj.eu-gb.mybluemix.net/red/#flow/e2e848bd.ed88e8>

<http://ai2.appinventor.mit.edu/#6327367580909568>

<https://cloud.ibm.com/objectstorage/crn%3Av1%3Abluemix%3Apublic%3Acloud-object-storage%3Aglobal%3Aa%2F1c21626fb1ca47d888d3693d1bb661a7%3A2e1cfbd8-2e7d-47c1-8c02-b84d2d629cb5%3A%3A?paneId=manage>

**APPINDIX:**

***SOURCE CODE:***

*import ibm\_boto3*

*from ibm\_botocore.client import Config, ClientError*

*# Constants for IBM COS values*

*COS\_ENDPOINT = "https://s3.jp-tok.objectstorage.softlayer.net" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints*

*COS\_API\_KEY\_ID = "NslIgIb4n0nFaCrMsfHNHD2OPiFuL0VwCYBZDO0VBe\_I" # eg "W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"*

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

*COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/1c21626fb1ca47d888d3693d1bb661a7:2e1cfbd8-2e7d-47c1-8c02-b84d2d629cb5::" # eg "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"*

*# 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 create\_bucket(bucket\_name):*

*print("Creating new bucket: {0}".format(bucket\_name))*

*try:*

*cos.Bucket(bucket\_name).create(*

*CreateBucketConfiguration={*

*"LocationConstraint":"jp-tok-standard"*

*}*

*)*

*print("Bucket: {0} created!".format(bucket\_name))*

*except ClientError as be:*

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

*except Exception as e:*

*print("Unable to create bucket: {0}".format(e))*

*create\_bucket("pranathi0111")*

*import cv2*

*import numpy as np*

*import datetime*

*import json*

*from watson\_developer\_cloud import VisualRecognitionV3*

*visual\_recognition = VisualRecognitionV3(*

*'2018-03-19',*

*iam\_apikey='NCqlnKVbIA6xhxGxBuqoolInjwo5zjHtMPiMH1IpxtOJ')*

*face\_classifier=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")*

*eye\_classifier=cv2.CascadeClassifier("haarcascade\_eye.xml")*

*#It will read the first frame/image of the video*

*video=cv2.VideoCapture(0)*

*while True:*

*#capture the first frame*

*check,frame=video.read()*

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

*#detect the faces from the video using detectMultiScale function*

*faces=face\_classifier.detectMultiScale(gray,1.3,5)*

*eyes=eye\_classifier.detectMultiScale(gray,1.3,5)*

*#drawing rectangle boundries for the detected face*

*for(x,y,w,h) in faces:*

*cv2.rectangle(frame, (x,y), (x+w,y+h), (127,0,255), 2)*

*cv2.imshow('Face detection', frame)*

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

*cv2.imwrite(picname+".jpg",frame)*

*#drawing rectangle boundries for the detected eyes*

*for(ex,ey,ew,eh) in eyes:*

*cv2.rectangle(frame, (ex,ey), (ex+ew,ey+eh), (127,0,255), 2)*

*cv2.imshow('Face detection', frame)*

*cv2.imwrite("face.jpg",frame)*

*#waitKey(1)- for every 1 millisecond new frame will be captured*

*Key=cv2.waitKey(1)*

*if Key==ord('q'):*

*#release the camera*

*video.release()*

*#destroy all windows*

*cv2.destroyAllWindows()*

*break*

*with open('./face.jpg', 'rb') as images\_file:*

*classes = visual\_recognition.classify(*

*images\_file,*

*threshold='0.6',*

*classifier\_ids='pranathi\_854492345').get\_result()*

*a=json.dumps(classes, indent=2)*

*b=json.loads(a)*

*print(b)*

*c=b['images']*

*for i in c:*

*for j in i['classifiers']:*

*k=j['classes']*

*for l in k:*

*print(l['class'])*

*x=l['class']*

*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))*

*multi\_part\_upload("pranathi0111", "animal1.jpg", picname+".jpg")*

*from ibm\_watson import TextToSpeechV1*

*from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator*

*from playsound import playsound*

*authenticator = IAMAuthenticator('BK5EvafK89DsEyIONVsgWy2G13\_lIyX8B6A11RDU6HUG')*

*text\_to\_speech = TextToSpeechV1(*

*authenticator=authenticator*

*)*

*text\_to\_speech.set\_service\_url('https://api.eu-gb.text-to-speech.watson.cloud.ibm.com/instances/84ac80fd-3de6-4403-84d7-7bc58ea7a953')*

*with open('pranathi.mp3', 'wb') as audio\_file:*

*audio\_file.write(*

*text\_to\_speech.synthesize(*

*f'the item is {x}',*

*voice='en-US\_AllisonVoice',*

*accept='audio/mp3'*

*).get\_result().content)*

*playsound('pranathi.mp3')*

*SOIL MOISTURE:*

*import time*

*import sys*

*import ibmiotf.application*

*import ibmiotf.device*

*import random*

*#Provide your IBM Watson Device Credentials*

*organization = "6vmi9b"*

*deviceType = "raspberrypi"*

*deviceId = "000777"*

*authMethod = "token"*

*authToken = "111222333"*

*# Initialize GPIO*

*def myCommandCallback(cmd):*

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

*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:*

*hum=20*

*#print(hum)*

*temp = 32*

*#Send Temperature & Humidity to IBM Watson*

*data = { 'Temperature' : temp, 'Humidity': hum }*

*#print (data)*

*def myOnPublishCallback():*

*print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum, "to IBM Watson")*

*success = deviceCli.publishEvent("DHT11", "json", data, qos=0, on\_publish=myOnPublishCallback)*

*if not success:*

*print("Not connected to IoTF")*

*time.sleep(2)*

*deviceCli.commandCallback = myCommandCallback*

*# Disconnect the device and application from the cloud*

*deviceCli.disconnect()*