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:

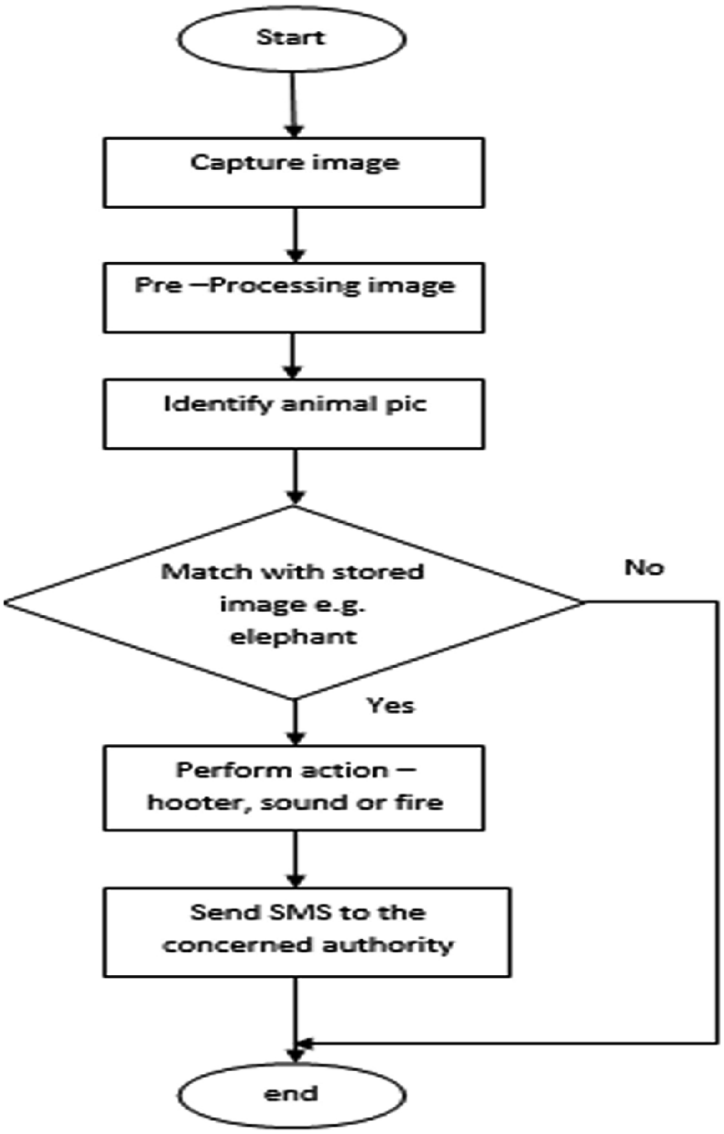
Wild animals are a special challenge for farmers throughout the world. Animals such as deer, wild boars, rabbits, moles, elephants, monkeys, and many others may cause serious damage to crops. They can damage the plants by feeding on plant parts or simply by running over the field and trampling over the 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.

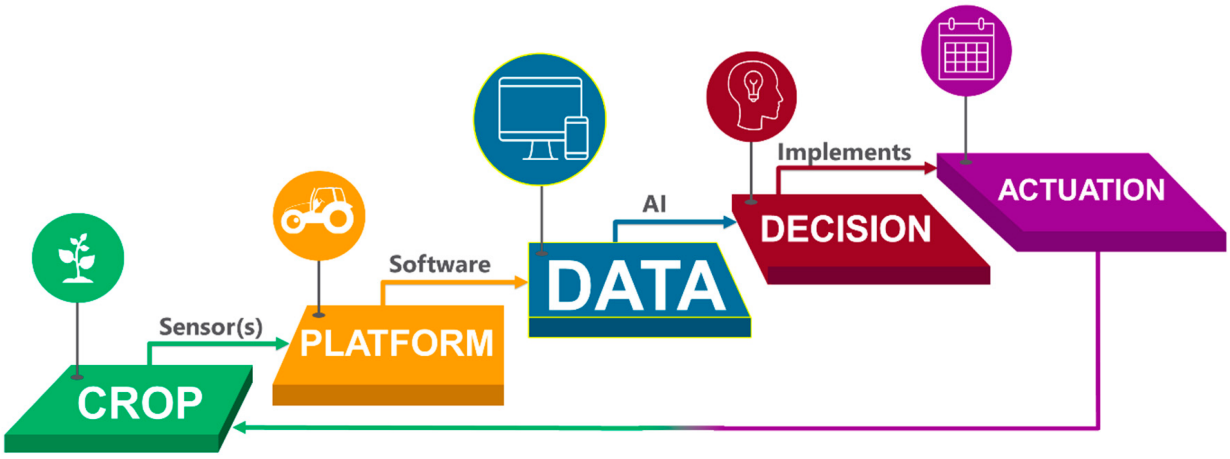
THEORITICAL ANALYSIS:

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 & integrated with the Node-Red to observe the object.



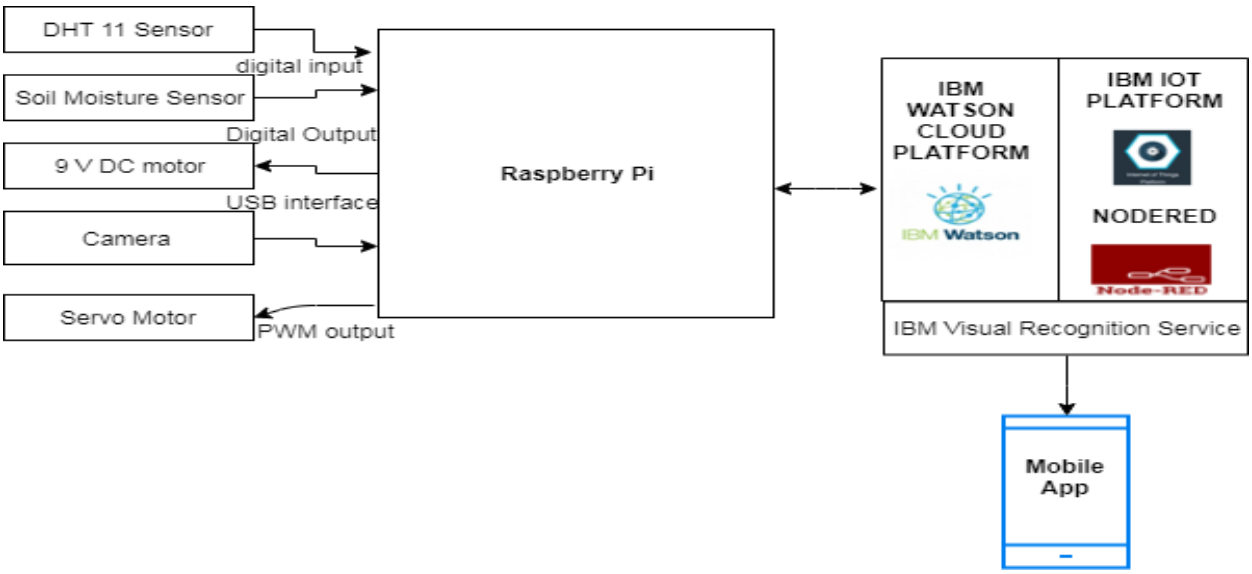
EXPERIMENTAL INVESTIGATION:

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.

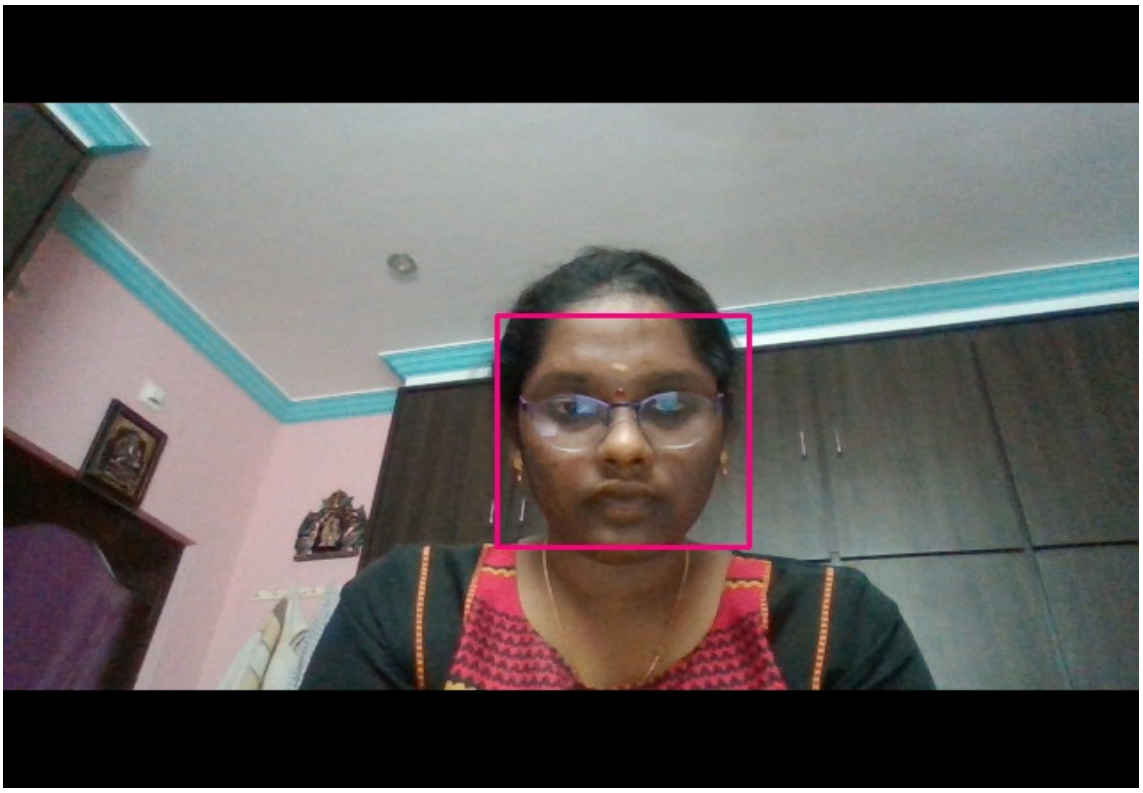
FLOW CHART:



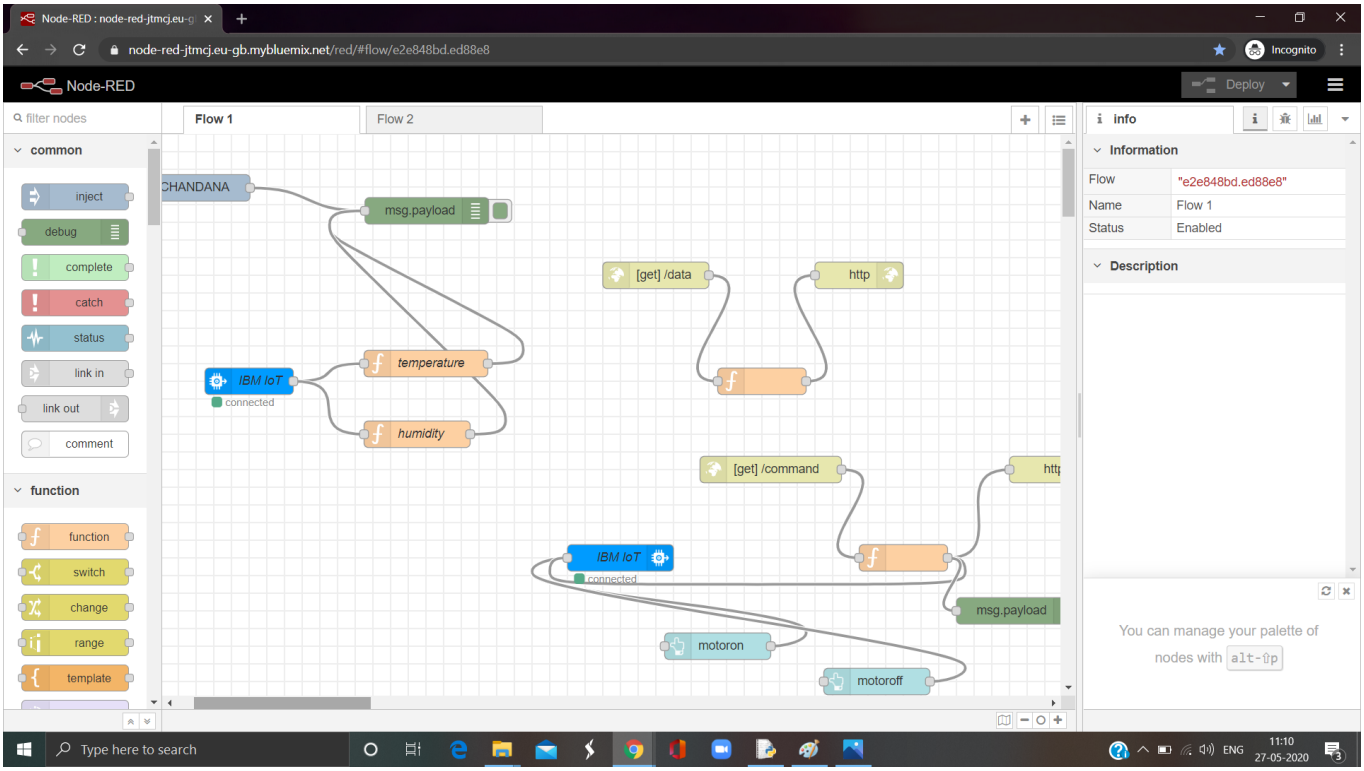
RESULT:

PYTHON CODE:

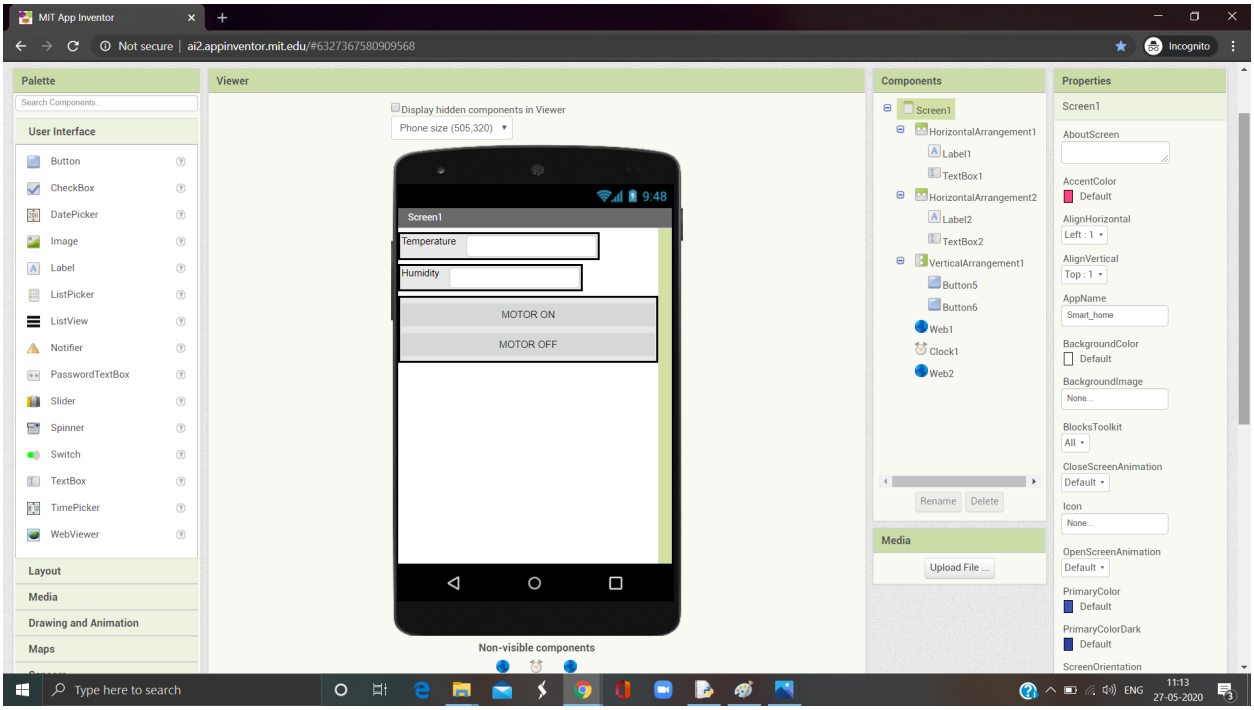
```
*Python 3.7.7 Shell*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Pranathi\Desktop\project\Animal Capturing and play sound1.py
Creating new bucket: pranathi0111
Bucket: pranathi0111 created!

Warning (from warnings module):
  File "C:\Users\Pranathi\Desktop\project\Animal Capturing and play sound1.py",
line 42
    iam_apikey='NCqlnKVbIA6xhxGxBuqoolInjwo5zjHtMPiMH1IpxtOJ')
DeprecationWarning: watson-developer-cloud moved to ibm-watson. To get updates,
use the new package.
```

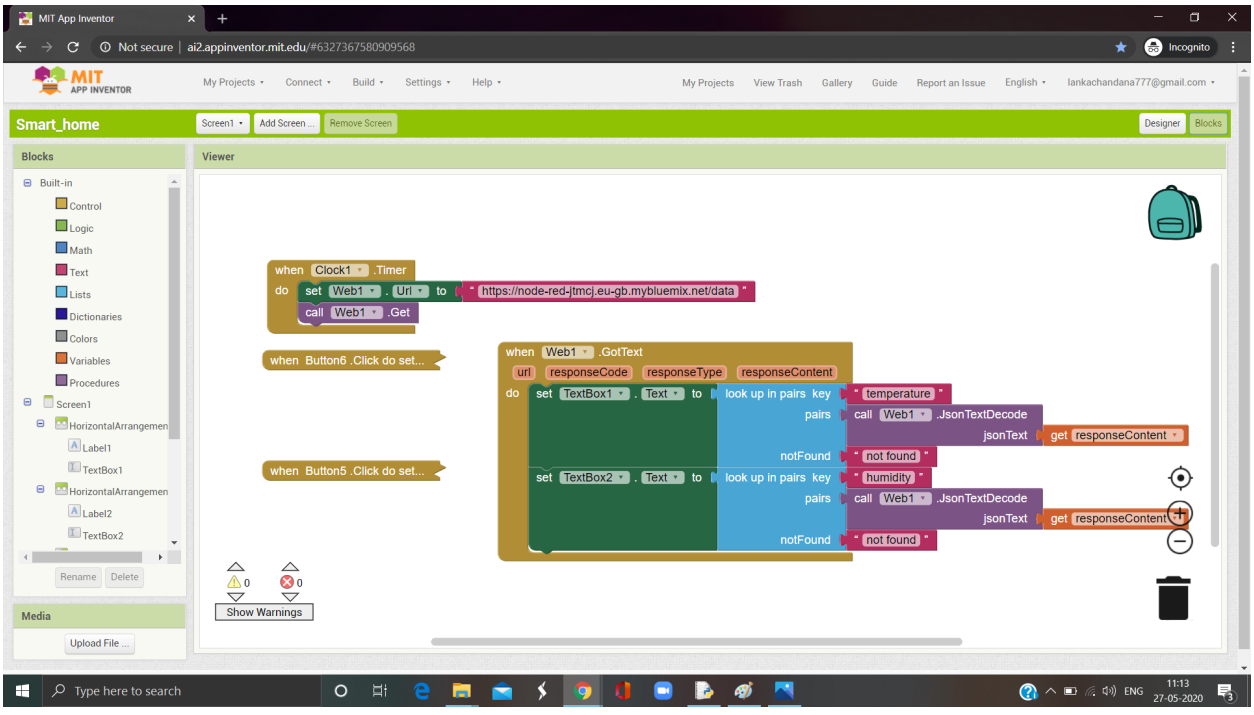


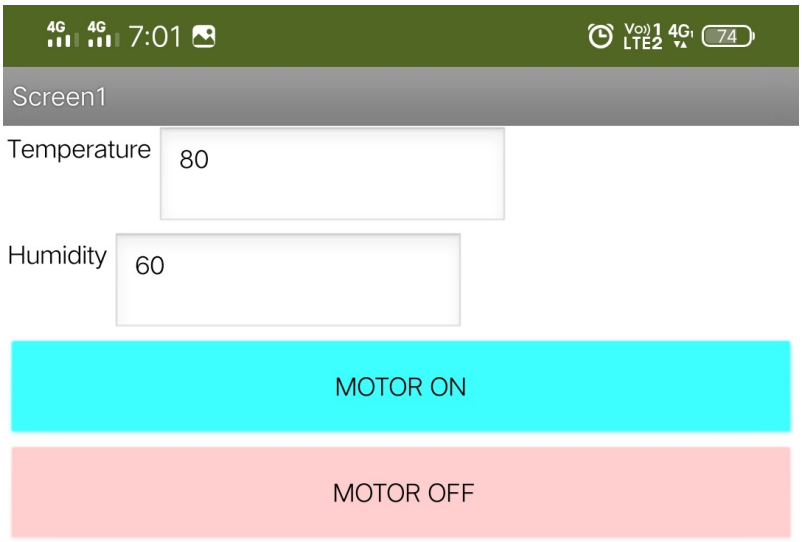
NODE-RED:


Node-RED : node-red-jtmcj.eu-g
node-red-jtmcj.eu-gb.mybluemix.net/red/#flow/e2e848bd.ed88e8
Incognito
Node-RED
Deploy
filter nodes
Flow 1
Flow 2
common
inject
debug
complete
catch
status
link in
link out
comment
function
function
switch
change
range
template
CHANDANA
msg.payload
[get] /data
http
temperature
IBM IoT
connected
humidity
[get] /command
IBM IoT
connected
motoron
motoroff
msg.payload
info
Information
Flow
"e2e848bd.ed88e8"
Name
Flow 1
Status
Enabled
Description
You can manage your palette of nodes with alt-⇧p
Type here to search
ENG
11:10
27-05-2020

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_IlyX8B6A11RDU6HUG')
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()
```