

FINAL DELIVERABLES

PROJECT REPORT

Date	18 November 2022
Team ID	PNT2022TMID30932
Project Name	IoT Based Smart Crop Protection System for Agriculture.

Team Leader

Sujithraa.S(620119106098)

Team Members

Sakthi.R(620119106079)

Vaishnavi.K.S (620119106100)

Vannamathi.S(620119106101)

Bachelor of Engineering
In
Electronics and Communication Engineering

AVS Engineering College, Salem.

Project Report Index

1. INTRODUCTION

1. Project Overview
2. Purpose

2. LITERATURE SURVEY

1. Existing problem
2. References
3. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Proposed Solution
4. Problem Solution fit

4. REQUIREMENT ANALYSIS

1. Functional requirement
2. Non-Functional requirements

5. PROJECT DESIGN

1. Data Flow Diagrams
2. Solution & Technical Architecture
3. User Stories

6. PROJECT PLANNING & SCHEDULING

1. Sprint Planning & Estimation
2. Sprint Delivery Schedule
3. Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

1. Feature 1
2. Feature 2
3. Database Schema (if Applicable)

8. TESTING

9. ADVANTAGES & DISADVANTAGES

10. CONCLUSION

11. FUTURE SCOPE

12. APPENDIX

1. Source Code
2. GitHub & Project Demo Link

IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

I. Introduction:

1. Project Objectives:

- The device will detect the animals and birds using the Clarifai service.
- If any animal or bird is detected the image will be captured and stored in the IBM Cloud object storage.
- It also generates an alarm and avoid animals from destroying the crop.
- The image URL will be stored in the IBM Cloudant DB service.
- The device will also monitor the soil moisture levels, temperature, and humidity values and send them to the IBM IoT Platform.
- The image will be retrieved from Object storage and displayed in the web application.
- A web application is developed to visualize the soil moisture, temperature, and humidity values.
- Users can also control the motors through web applications.

2. Purpose:

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.

II. Literature Survey:

1. Existing problem:

Crops in the farms are many times devastated by the wild as well as domestic animals and low productivity of crops is one of the reasons for this. It is not possible to stay 24 hours in the farm to guard the crops.

2. References:

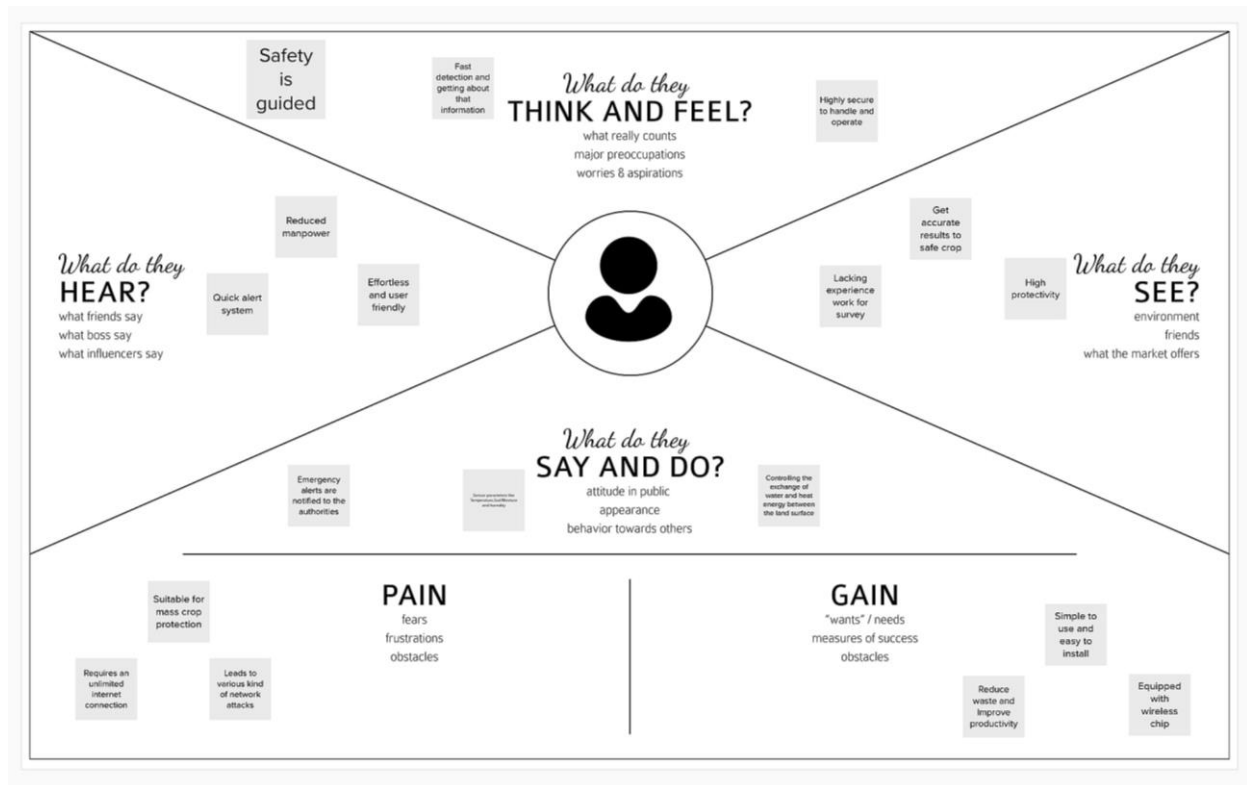
- **Title:** IOT IN AGRICULTURE CROP PROTECTION AND POWER GENERATION (2020)
Author: Anjana M, Charan Kumar A, Monisha R, Sahana R H
- **Title:** IOT BASED CROP PROTECTION SYSTEM AGAINST AND WILD ANIMAL ATTACKS (2020)
Author: Navaneetha P, RamiyaDevi R, Vennila S, Manikandan P, Dr. Saravanan S
- **Title:** SMART CROP PROTECTION SYSTEM (2021)
Author: Krunal Mahajan, Riya Parate, Ekta Zade, Shubham Khante, Shishir Bagal

3. Problem statement:

Problem Statement (PS)	I am (Customer)	I am trying to	But	Because	Which makes me feel
PS-1	Farmer	Monitoring the growing condition	It involves risk on related equipment and understand the use of technology	Requires more knowledge and skills	Irritated
PS-2	Farmer	Smart and precision irrigation	Climates changes to increased maintenance of channels	Purchasing and installing costs high	Suitable for mass crop protection

III. IDEATION & PROPOSED SOLUTION

1. Empathy Map:



2. Ideation & Brainstorming:



3. Proposed Solution:

SI No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Develop an efficient system & an application that can monitor and alert the users(farmers)
2.	Idea/Solution description	<ol style="list-style-type: none">1. This product helps the field in monitoring the animal other disturbance2. In several areas, the temperature sensors will be integrated to monitor the temperature & humidity3. If in any area feel dry or wet less is detected by admins, will be notified along with the location in the web application
3.	Novelty/Uniqueness	<ol style="list-style-type: none">1. Fastest alerts to the farmers2. The increasing demand for quality food , User friendly
4.	Social Impact/Customer Satisfaction	<ol style="list-style-type: none">1. Easy installation and provide efficient results2. Can work with irrespective of fear
5.	Business Model (Revenue Model)	<ol style="list-style-type: none">1. As the product usage can be understood by everyone, it is easy for them to use it properly for their safest organization2. The product is advertised all over the platforms. Since it is economical, even helps small scale farming land from disasters.
6.	Scalability of the Solution	🔗 Even when the interruption is more, the product sense the accurate location and alerts the farmers effectively

4. Problem Solution fit:

Define CS, fit into CL	1. CUSTOMER SEGMENT(S) CS <ul style="list-style-type: none"> Farmers, who's not near his field. Crop importers 	6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> <ul style="list-style-type: none"> High adoption costs, security concerns. Prevent the unnecessary use of this device. Use it according to the climate change 	5. AVAILABLE SOLUTIONS AS <small>PLUSES & MINUSES</small> <ul style="list-style-type: none"> Monitor different parameters and mobile or web application make easily to farm the crop field. Certain cultural practices can prevent or reduce insect crop damage. 	Explore AS, differentiate
	2. PROBLEMS / PAINS PR <small>ITS FREQUENCY</small> <ul style="list-style-type: none"> It's difficult to monitor and control Ain't known if the application doesn't work properly. 	9. PROBLEM ROOT / CAUSE RC <ul style="list-style-type: none"> If temperature, PH level ,humidity & light intensity makes the serious cause for the environment. Farmer affected by less productivity which will affect in their profit. 	7. BEHAVIOR BE <small>ITS INTENSITY</small> <p>Direct related: Tries to find a solution to prevent this problem</p> <p>Indirect related: Located in rural where internet connectivity might not be strong enough to facilitate fast transmission speeds.</p>	
Identify strong TR & EM	3. TRIGGERS TO ACT TR <p>Create opportunities to lift people out of poverty in developing nations. (Over 60%)</p>	10. YOUR SOLUTION SL <p><i>"IoT based Smart crop protection system for agriculture"!!</i></p> <p>It help farmers grow more food on less land by protection crops from pests, diseases and weeds as well as raising productivity per hectare.</p>	8. CHANNELS of BEHAVIOR CH <p>ONLINE</p> <p>ONLINE: The Data send through application for the farmers to know about the farms.</p>	Extract online & offline CH of BE
	4. EMOTIONS EM <small>BEFORE / AFTER</small> <p>BEFORE: Finances, Heavy work overload and conflict in relationship.</p> <p>AFTER: It will easier to make more yield in</p>		<p>OFFLINE</p> <p>OFFLINE: The control action is taken by the farmers to monitor the farms.</p>	

IV. REQUIREMENT ANALYSIS:

1. Functional requirement:

FR. No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Sensor animal's nearing the crop field and sounds alarm to woo them away as well as sends SMS to farmer using cloud service.
FR-2	User Reception	The Data like values of Temperature, Humidity, Soil moisture sensors are received via SMS
FR-3	User Understanding	Based on the sensor data value to get the information about present of farming land
FR-4	User Action	The user needs take action like destruction of crop residues, deep ploughing, crop rotation, fertilizers, strip cropping, scheduled planting operations.

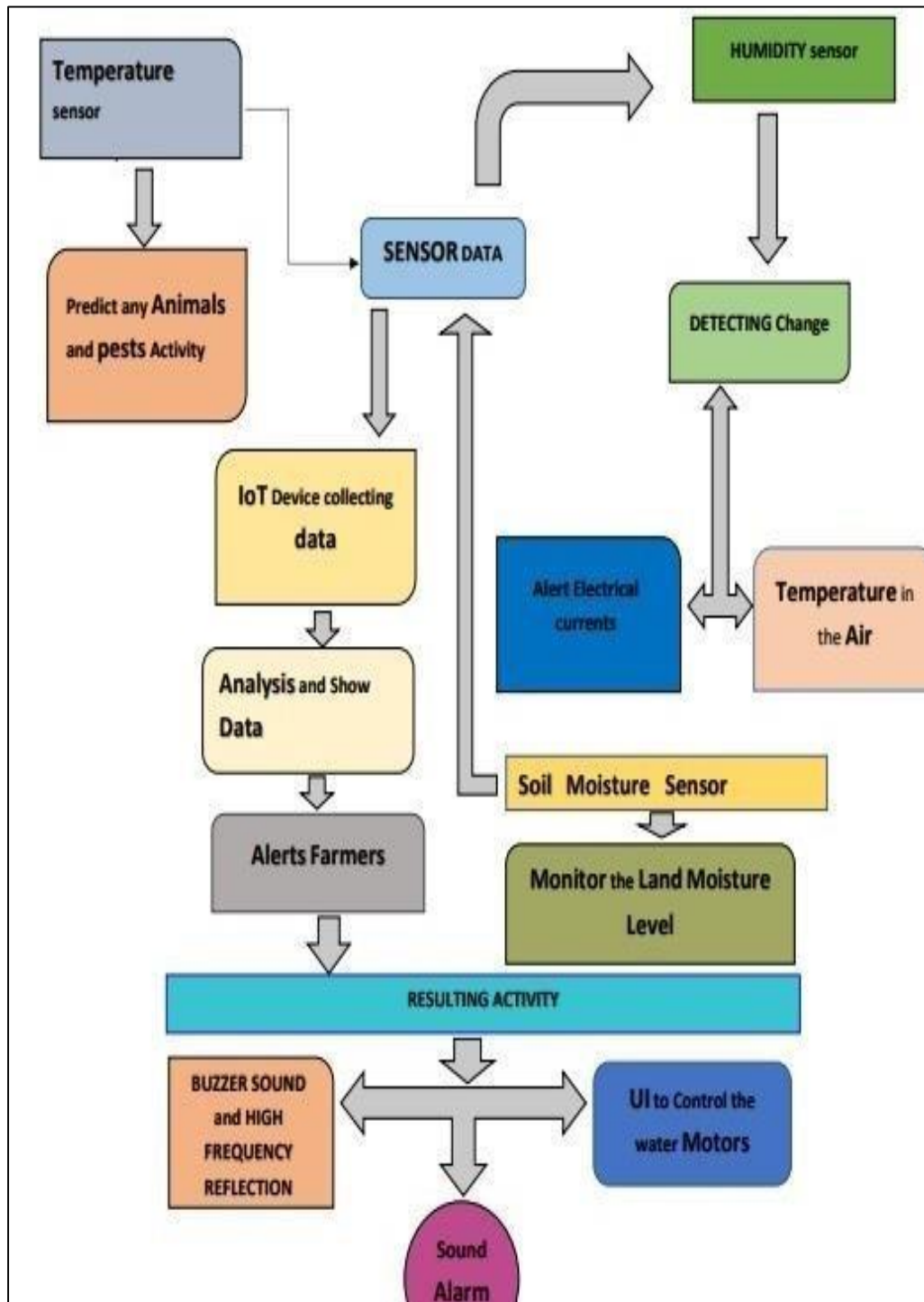
2. Non-Functional requirements:

FR No	Non-Functional Requirement	Description
NFR-1	Usability	Mobile support. Users must be able to interact in the same roles & tasks on computers & mobile devices where practical, given mobile capabilities.
NFR-2	Security	Data requires secure access to must register
NFR-3	Reliability	It has a capacity to recognize the disturbance near the field and doesn't give a false caution signal.

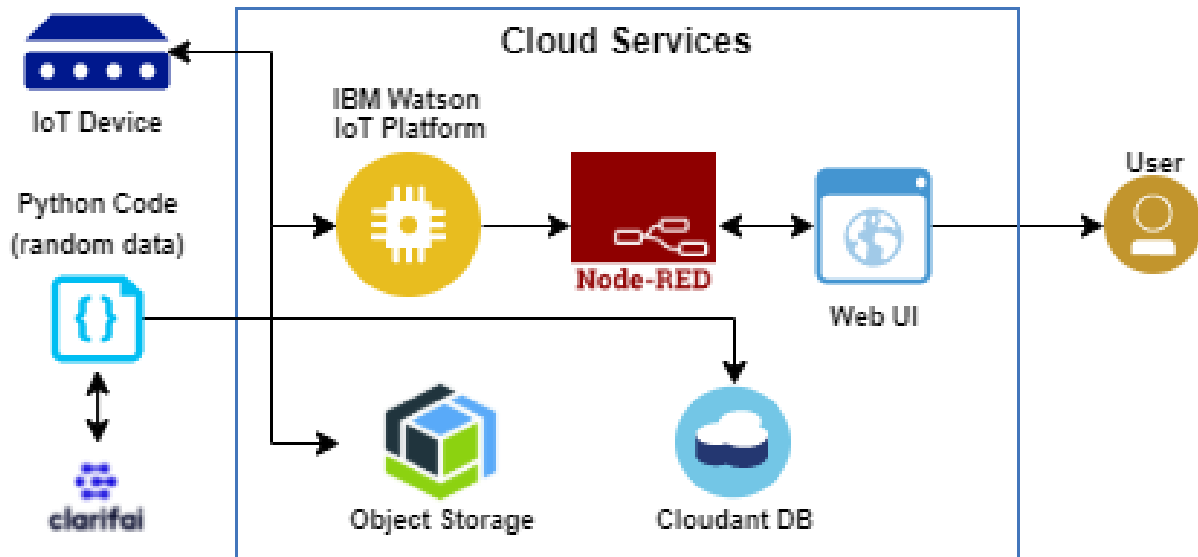
NFR-4	Performance	<p>Must provide acceptable response times to users regardless of the volume of data that is stored and the analytics that occurs in background.</p> <p>Bidirectional, near real-time communications must be supported. This requirement is related to the requirement to support industrial and device protocols at the edge.</p>
NFR-5	Availability	<p>IOT solutions and domains demand highly available systems for 24x7 operations. Isn't a <i>critical production</i> application, which means that operations or production don't go down if the IOT solution is down.</p>
NFR-6	Scalability	<p>System must handle expanding load and data retention needs that are based on the upscaling of the solution scope, such as extra manufacturing facilities and extra buildings.</p>

V. PROJECT DESIGN:

1. Data Flow Diagrams:



2. Solution & Technical Architecture:



► Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with the Web UI	App development
2.	Application Logic-1	Logic for a process in the application	Python Objectives
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	Node-RED service
5.	Database	Data Type	Database Cloud DB
6.	Cloud Database	Database Service on Cloud	Cloud Object store service
7.	File Storage	File storage requirements	IBM Block Storage

8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration:	Cloud Foundry
----	------------------------------------	---	---------------

► **Application characteristics:**

SI No	Characteristics	Description	Technology
1.	Open-source Frameworks	The open - source frameworks used	SAN-SAF
2.	Security Implementations	List all the security / access controls implemented	IBM cloud encryptions
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	IBM cloud Architecture
4.	Availability	Justify the availability of applications (e.g., use of load balancers, distributed servers etc.)	Web Application can even be used by the framers in the horticulture
5.	Performance	Design consideration for the performance of the application	Since the web application is high efficient, it can be used by the farmers irrespective of time.

3. User Stories:

User Type	Functional requirement (Epic)	User Story number	User Story/Task	Acceptance criteria	Priority	Release
Mobile users	Registration	USN-1	User can enter into the web application	I can access my account /dashboard	High	Sprint 1
		USN-2	User can register their credentials like email id and password	I can receive confirmation email & click confirm	High	Sprint 1
	Login	USN-3	User can log into the application by entering email & password	I can login to my account	High	Sprint 1
	Dashboard	USN-4	User can view the temperature	I can view the data given by the device	High	Sprint 2
		USN-5	User can view the level of sensor monitoring value	I can view the data given by the device	High	Sprint 2
Web users	Usage	USN-1	User can view the web page and get the information	I can view the data given by the device	High	Sprint 3
Customer	Working	USN-1	User act according to the alert given by the device	I can get the data work according to it.	High	Sprint 3
		USN-2	User turns ON the water motors/Buzzer/Sound Alarm when occur the disturbance on field.	I can get the data work according to it.		Sprint 4

Customer care Executive	Action	USN-1	User solve the problem when some faces any usage issues	I can solve the issues when someone fails to understanding the procedure	High	Sprint 4
Administration	Administration	USN-1	User store every information	I can store the gained information	High	Sprint 4

VI. PROJECT PLANNING & SCHEDULING:

1. Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	S Sujithraa
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	High	S Sujithraa
Sprint-2	Cloud Service	USN-3	As a user, I can register for the application through Facebook or any social media	1	Low	KS Vaishnavi
Sprint-4		USN-4	As a user, I can register for the application through Gmail / web service	2	Medium	S.Vannamathi.
Sprint-3	Login	USN-5	As a user, I can log into the application by entering email & password	4	High	R Sakthi
Sprint-2	Pre processing	USN-6	As a farmer, the user must be able to find the system easy to access so the Prep-processes and other task must be perfect	3	High	KS Vaishnavi
Sprint-1	Collecting Dataset	USN-7	To collect various sources of animal threats and keep developing a dataset using Clarifai.	3	Medium	S Sujithraa
Sprint-4	Integrating	USN-8	To integrate the available dataset and keep improving the	2	Medium	S.Vannamathi.

			accuracy of finding animals			
Sprint-3		USN-9	To find and use appropriate compiler to run and test the data so that we can implement our program	1	Low	R Sakthi
Sprint-2		USN-10	Request AVS Engineering College to deploy the project in our campus and test	1	Low	KS Vaishnavi
Sprint-1	Training	USN-11	As programmer, we need to train our data perfectly so that the program runs smoothly	3	High	S Sujithraa
Sprint-3		USN-12	Train the data using out available service and IBM dataset from server and improve that	2	Medium	R Sakthi
Sprint-4	Coding	USN-13	To modify the code according to our program and improve the efficiency of that code	4	High	S.Vannamathi.
Sprint-2		USN-13	To improve performance	1	Low	KS Vaishnavi
Sprint-2	Record	USN-5	To record the data and plot the graph to show the characteristics officially	4	Medium	KS Vaishnavi
Sprint-1	Planning	USN-4	Plan the programming language and feasibility	3	High	S Sujithraa
Sprint-4		USN-14	Demonstrate the working and improve accuracy overall	2	Low	KS Vaishnavi

2. Sprint Delivery Schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	20Oct 2022	24 Oct 2022	20	21 Oct 2022
Sprint-2	20	6 Days	25 Oct 2022	29 Oct 2022	20	27 Oct 2022
Sprint-3	20	6 Days	31 Oct 2022	4 Nov 2022	20	2 Nov 2022
Sprint-4	20	6 Days	5 Nov 2022	11 Nov 2022	20	8 Nov 2022

VII. CODING & SOLUTIONING:

1. Feature 1:

IoT based smart crop protection system was implemented using traditional farming concepts and it has a user interfacing system to monitor the temperature humidity and moisture level of the soil. It enables smart farming through that the farmer can access the environmental parameters. The Random module used to generate the values for moisture, temperature and humidity. These values are further sent to the Watson platform.

2. Feature 2:

Further the smart crop protection system was enhanced by creating the user interface. Node red web user interface and MIT app inventor were used to create the user interface. The data from the python script were stored in Watson and the animal detected information were uploaded in the object storage. The opencv2 module is used to capture the animal picture in the field and alter message will be sent to the farmer through the web user interface and mobile application.

3. Database Schema:

► IBM Watson IoT platform:

Random temperature, humidity, and moisture values are generated using the python code and the values are sent to the IBM cloud. IBM cloud sends those values to the node red and shown in the node red dashboard

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. Below the navigation bar, there is a table listing devices. The first device, '6020', is highlighted, and its details are shown in a modal window. The modal window has tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a table of events with columns: Event, Value, Format, and Last Received. The events are status updates from an animal, including moisture and temperature readings. At the bottom of the dashboard, there is a status bar indicating '0 Simulations running'.

Event	Value	Format	Last Received
status	{\"Animal\":\"false\",\"moisture\":\"2\",\"hum\":\"120\",\"temp\"...	json	a few seconds ago
status	{\"Animal\":\"false\",\"moisture\":\"20\",\"hum\":\"75\",\"temp\"...	json	a few seconds ago
status	{\"Animal\":\"false\",\"moisture\":\"52\",\"hum\":\"81\",\"temp\"...	json	a few seconds ago
status	{\"Animal\":\"false\",\"moisture\":\"100\",\"hum\":\"114\",\"temp\"...	json	a few seconds ago
status	{\"Animal\":\"false\",\"moisture\":\"47\",\"hum\":\"122\",\"temp\"...	json	a few seconds ago

► Cloud object storage :

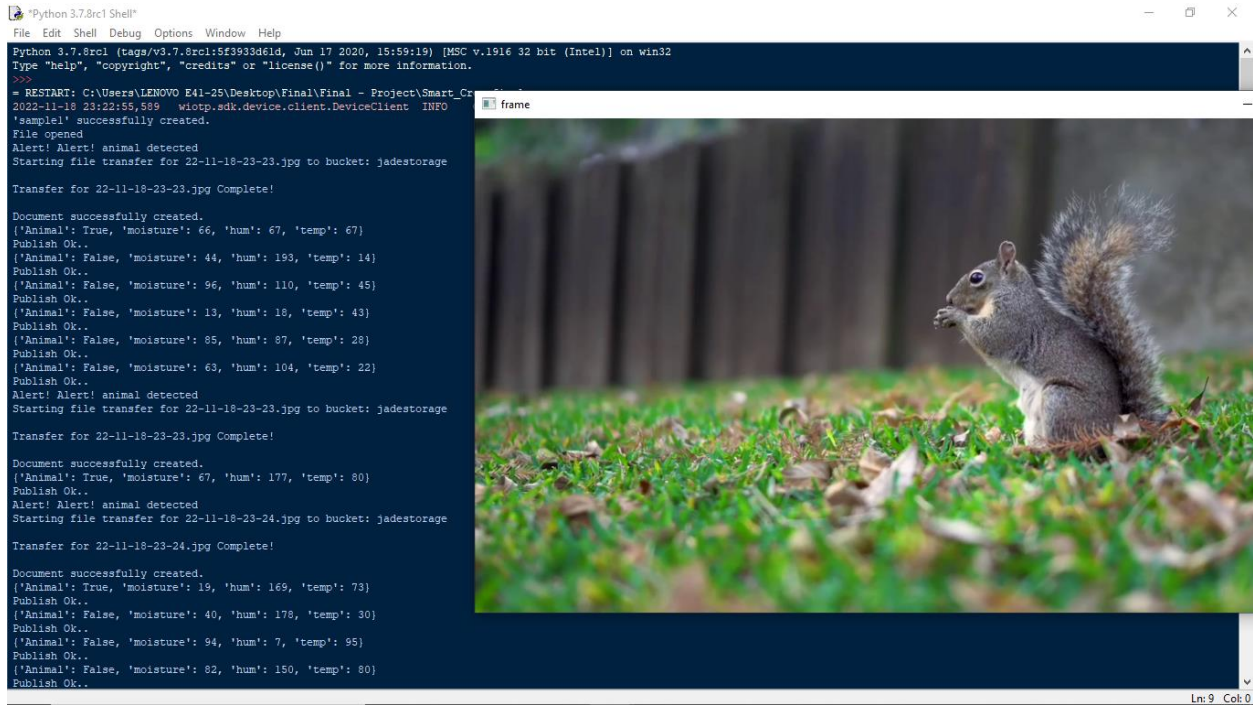
This is the cloud storage area where we can store the images of the detected animal.

The screenshot shows the IBM Cloud Object Storage console. The left sidebar contains navigation options: 'Cloud Object Storage', 'Storage instances', 'Buckets', 'Integrations', 'Endpoints', 'Usage details', 'Service credentials', 'Connections', and 'Plan'. The main area displays the 'jadestorage' bucket. A warning message states: 'Warning: All objects in this bucket have public view access.' Below the warning, there is a table of objects with columns: Object name, Archived, Size, and Last modified. The objects are listed with their names, sizes, and modification times. An 'Upload' button is visible in the top right corner of the object list.

Object name	Archived	Size	Last modified
22-11-18-14-...		52.4 KB	2022-11-18 2:35 PM
22-11-18-14-...		52.4 KB	2022-11-18 2:36 PM
22-11-18-14-...		51.6 KB	2022-11-18 2:37 PM
22-11-18-14-...		52.2 KB	2022-11-18 2:38 PM
22-11-18-14-...		52.5 KB	2022-11-18 2:39 PM
22-11-18-14-...		211.1 KB	2022-11-18 2:51 PM
22-11-18-14-...		210.3 KB	2022-11-18 2:52 PM
22-11-18-14-...		210.9 KB	2022-11-18 2:53 PM

VIII. TESTING

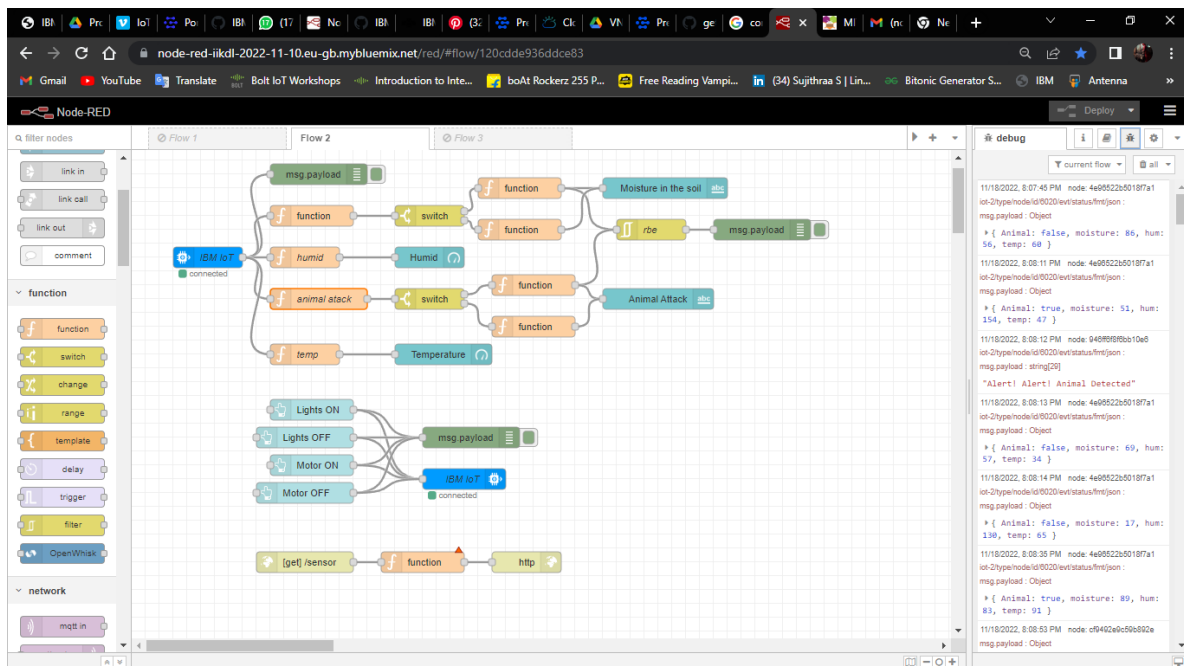
► Python code testing:



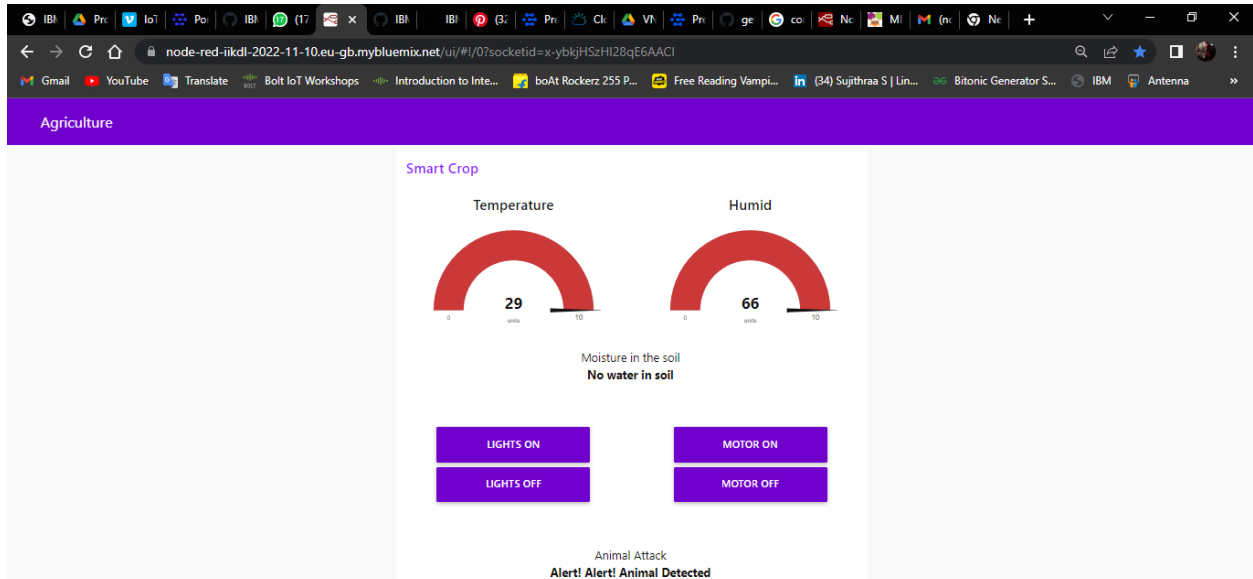
```
Python 3.7.8rc1 (tags/v3.7.8rc1:5f3933d61d, Jun 17 2020, 15:59:19) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
* RESTART: C:\Users\LENOVO E41-25\Desktop\Final\Final - Project\Smart_C...
2022-11-18 23:22:55,589 wiotp.sdk.device.client.DeviceClient INFO
'sample1' successfully created.
File opened
Alert! Alert! animal detected
Starting file transfer for 22-11-18-23-23.jpg to bucket: jadestorage
Transfer for 22-11-18-23-23.jpg Complete!
Document successfully created.
{'Animal': True, 'moisture': 66, 'hum': 67, 'temp': 67}
Publish Ok..
{'Animal': False, 'moisture': 44, 'hum': 193, 'temp': 14}
Publish Ok..
{'Animal': False, 'moisture': 96, 'hum': 110, 'temp': 45}
Publish Ok..
{'Animal': False, 'moisture': 13, 'hum': 18, 'temp': 43}
Publish Ok..
{'Animal': False, 'moisture': 85, 'hum': 87, 'temp': 28}
Publish Ok..
{'Animal': False, 'moisture': 63, 'hum': 104, 'temp': 22}
Publish Ok..
Alert! Alert! animal detected
Starting file transfer for 22-11-18-23-23.jpg to bucket: jadestorage
Transfer for 22-11-18-23-23.jpg Complete!
Document successfully created.
{'Animal': True, 'moisture': 67, 'hum': 177, 'temp': 80}
Publish Ok..
Alert! Alert! animal detected
Starting file transfer for 22-11-18-23-24.jpg to bucket: jadestorage
Transfer for 22-11-18-23-24.jpg Complete!
Document successfully created.
{'Animal': True, 'moisture': 19, 'hum': 169, 'temp': 73}
Publish Ok..
{'Animal': False, 'moisture': 40, 'hum': 178, 'temp': 30}
Publish Ok..
{'Animal': False, 'moisture': 94, 'hum': 7, 'temp': 95}
Publish Ok..
{'Animal': False, 'moisture': 82, 'hum': 150, 'temp': 80}
Publish Ok..
```

► Node-red testing:

Connection and output:

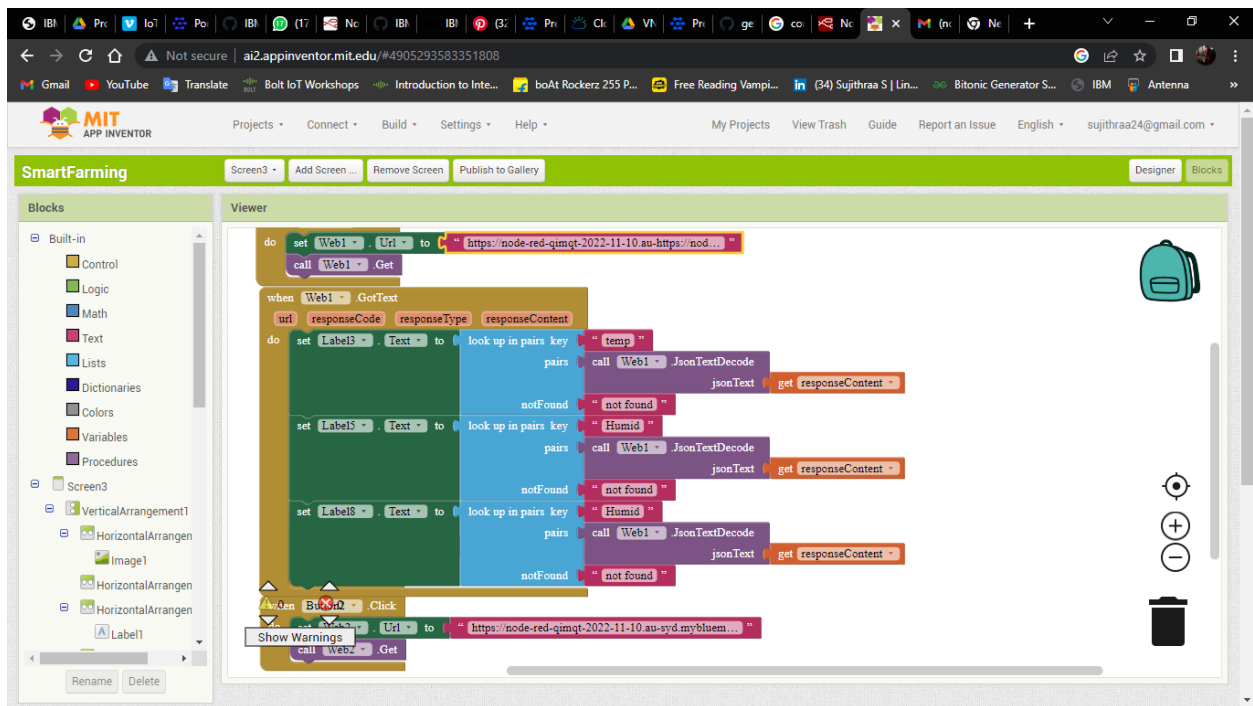


Web application testing:



➤ **Mobile Application testing:**

Mobile application creation :

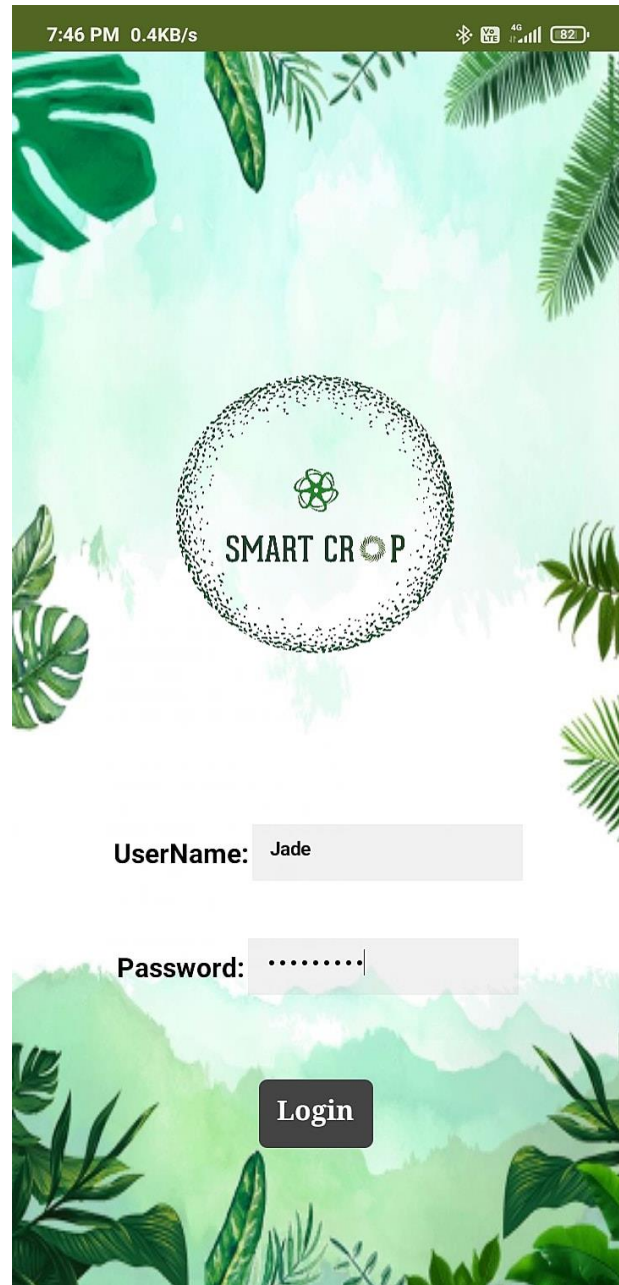


Mobile Application:

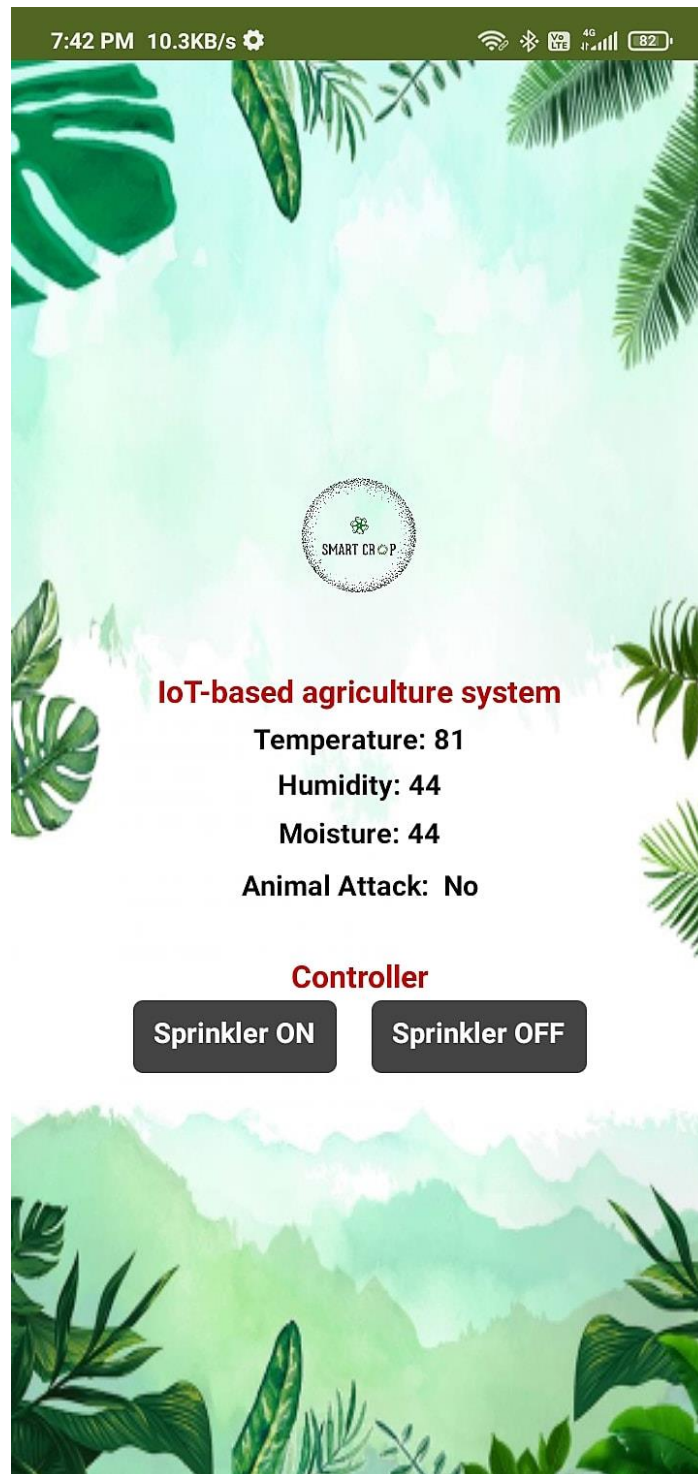
App interface:



Login page:



Dashboard Page:



IX. ADVANTAGES & DISADVANTAGES:

1. ADVANTAGES:

- Farmers can monitor the health of farm animals closely, even if they are physically distant.
- Smart farming systems reduce waste, improve productivity and enable management of a greater number of resources through remote sensing.
- High reliance.
- Enhanced Security.

2. DISADVANTAGES:

- Farms are located in remote areas and are far from access to the internet.
- A farmer needs to have access to crop data reliably at any time from any location, so connection issues would cause an advanced monitoring system to be useless.
- High Cost
- Equipment needed to implement IoT in agriculture is expensive.

X. CONCLUSION

As a result of this system, we can detect the changes in the field easily and intimate the farmers about it and also, we can take precautions and do remedies accordingly. Here we use very low power consuming highly efficient components that give us accurate results and also, they perform at low data rate conditions without any lag and help in finding the remedies. This crop protection system helps in detection of all kinds of external dangers and it saves time and money to the farmers before any loss that may occur. With the help of this system the farmers can be in a peaceful environment at ease without any pressure.

XI. FUTURE SCOPE:

Study and analysis of the developed Crop protection systems for its cost effectiveness with the development of Arduino based variable frequency Ultrasonic bird deterrent circuit. outline of the crop damage caused by a particular Wild animal if the behavioural features of the with the reduced cost in the smart phones.

XII. APPENDIX:

1. Source code:

```
import cv2

import numpy as np

import wiotp.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, resources_pb2

from clarifai_grpc.grpc.api.status import status_code_pb2


#This is how you authenticate

metadata = (('authorization', 'key 83ddcfb774c54cfd81d7a67ba69a0678'),)
```

```
COS_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"
```

```
COS_API_KEY_ID = "kn05el2QeCyawCFMRytUXLFirKVxw8v5HAIRvDKsIHmu"
```

```
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
```

```
COS_RESOURCE_CRN="crn:v1:bluemix:public:cloudantnosqldb:eu-  
gb:a/98d92dfd0ccf4f32a116d3d0fe24e15c:02d1fcad-1310-4403-93a6-  
a0eabc4c768b::"
```

```
Clientdb=Cloudant("apikey-v2-  
d8mn8ful7bxv3pw2cq0o1p1d8z3icznh8qu8y2xsv5",  
"400eef0a90d31fd7fa41c9dd0a2baa4b", url="https://cbf0b64e-c2d3-4404-be21-  
36565dc150b9-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 threshold 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)

```

```

    if(command=="lighton"):
        print('lighton')
    elif(command=="lightoff"):
        print('lightoff')
    elif(command=="motoron"):
        print('motoron')
    elif(command=="motoroff"):
        print('motoroff')
myConfig = {
    "identity": {
        "orgId": "tw9ckq",
        "typeId": "node",
        "deviceId": "6020"
    },
    "auth": {
        "token": "27102001"
    }
}
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

database_name = "sample1"
my_database = clientdb.create_database(database_name)
if my_database.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 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    imS= cv2.resize(frame, (960,540))
    cv2.imwrite('ex.jpg',imS)
    with open("ex.jpg", "rb") as f:
        file_bytes = f.read()
        detect=False
        t=random.randint(-1,1)
        if(t==0):
            detect=True
            print("Alert! Alert! animal detected")
            #playsound.playsound('alert.mp3')
            picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")
            cv2.imwrite(picname+'.jpg',frame)
            multi_part_upload('jadestorage',      picname+'.jpg',      picname+'.jpg')
            json_document=
            {"link":COS_ENDPOINT+'/'+'jadestorage'+'/'+'picname+'.jpg'}

```

```
new_document = my_database.create_document(json_document)
if new_document.exists():
    print(f"Document successfully created.")
    time.sleep(5)

moist=random.randint(0,100)
humidity=random.randint(0,200)
temperature=random.randint(0,100)
myData={'Animal':detect,'moisture':moist,'hum':humidity,'temp':temperature}
print(myData)
if(humidity!=None):
    client.publishEvent(eventId="status",msgFormat="json",      data=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()
```


► **GITHUB LINK:**

<https://github.com/IBM-EPBL/IBM-Project-28202-1660108638>

► **PROJECT DEMO:**

https://drive.google.com/file/d/1zA712kAELFAFpPNWGCeNDU6_joQvA/UTS/view?usp=share_link