**Smart Guest Identifier with remote access management**

M.Ramachandhra sai(18481A04D0), M.Rathna Rahul(18481A04D1), M.SuragaPallavi(18481A04D2), M.Aruna(18481A04D3), Md.Sharmila(18481A04D4).

Department of Electronics and Communication Engineering, Gudlavalleru engineering college.

**Contents**

1.Introduction

1.1. Overview

1.2. Purpose

2.Literature survey

2.1. Existing problem

2.2. Proposed solution

3.Theoretical analysis

3.1. Block diagram

3.2. Hardware/Software designing

4.Experimental Investigations

5.Flowchart

6.Result

7.Advantages & Disadvantages

8.Applications

9.Conclusion

10.Future Scope

11.Bibliography

12. Appendix

12.1. Source Code

12.2. UI Output Screenshot

**1.Introduction**

**1.1 Overview :**

Electronics visual display enabled by touchscreen technologies evolves as one of the universal multimedia output methods and a popular input intermediate with touch interaction

Since remote screen sharing systems are also increasingly prevalent, we propose a cross platform middle ware infrastructure which supports monitoring. We can use the most convenient screen to manage the most needed but unreachable devices within short period of time.

Smart guest identifier with remote access management is a device which is integrated at the entrance. There will be video streaming and whenever any person comes near the entrance, it will detect the person using IBM Watson visual recognition services and then it captures a picture and then that picture will be sent to the mobile application of the owner through the cloud platform. So that the person can access the door through the mobile application. He can send the alert messages to concerned security personnel in case of critical situation.

**1.2 Purpose :**

Security and Automation are the dual aspects of this project. The currently built prototype of the system sends pictures to the owner if any person is detected at the entrance. The provision for sending alert messages to concerned security is also built into the system.

Other hand if the owner identifies that the person entering his house/office is not an intruder but an unexpected guest of his then instead of triggering the security alarm, the owner can open the door. The same can be done when the user himself enters the room.

Thus using the same set of sensors the dual problems of security, applications can be solved on a complementary basis.

**2.Literature survey**

**2.1 Existing problem:**

Sometimes we will be in our office or we may go out for any reason. Then our relatives or friends or any other loved ones may visit our home. We may miss them or we won’t be able to receive them, as we don’t have any idea that someone has come to meet us.

It is very often to miss our key or forget our key in office which is too far. So, in order to open our door we have to go back to get it. that kills our time, energy as well.

We may also have another problem, that robbers may enter or any other unusual person attack our house we may not know what is going on there or we may visit after robbery .and none are able to provide security.

**2.2 Proposed solution:**

The device should be integrated at the entrance. There will be video streaming and whenever any person comes near the entrance, that should detect the person using IBM Watson visual recognition services. And then captures a picture and that picture is sent to the mobile application of the owner through the cloud platform

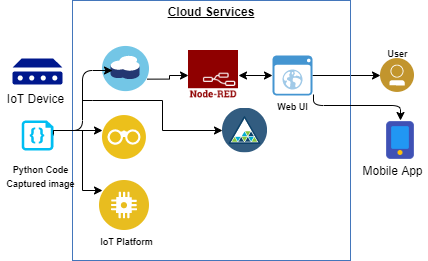
In the mobile application, the owner could see the person who is in front of their door even though he is not at home and they can also access the door. So that they can open the door through the mobile application by pressing a button.

If the person detected is recognized as a authorized person the door should be opened automatically

There must be one emergency button if there are any emergencies the person can press the button to send alerts to the concerned persons.

**3.Theoretical analysis**

**3.1 Block diagram:**

**

**3.2 Hardware/software** **designing:**

We follow a step-by-step procedure to set up all the interfaces required for our project and develop the code in python to send random sensor data to the cloud. The following software is required:

* Python Idle (with specified packages installed)
* IBM cloud
* Node Red service
* Mit app inventor

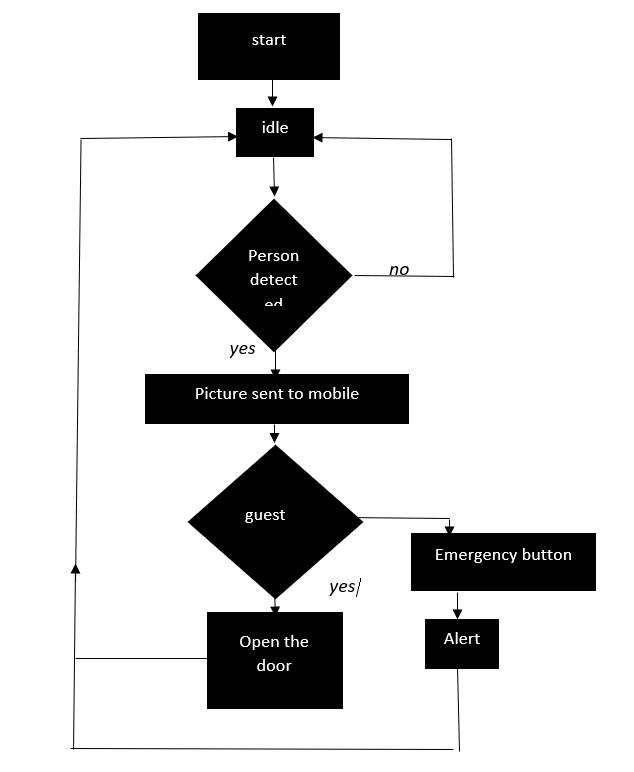
**4. Experimental investigations:**

To successfully recognise the guest who visited the home the installed camera capture the image of guest using python code snippet. The captured image is sent to the ibm cloud object by using ibm cloud credentials. If the guest is authorized person the door opens automatically.

By using Node red service, a node red flow is created to retrive the image from ibm cloud object storage and an UI is created to display the image and to control the access.

An android app is created to send image to the owner for authorization. If the owner gives access to the guest ,the door will open.

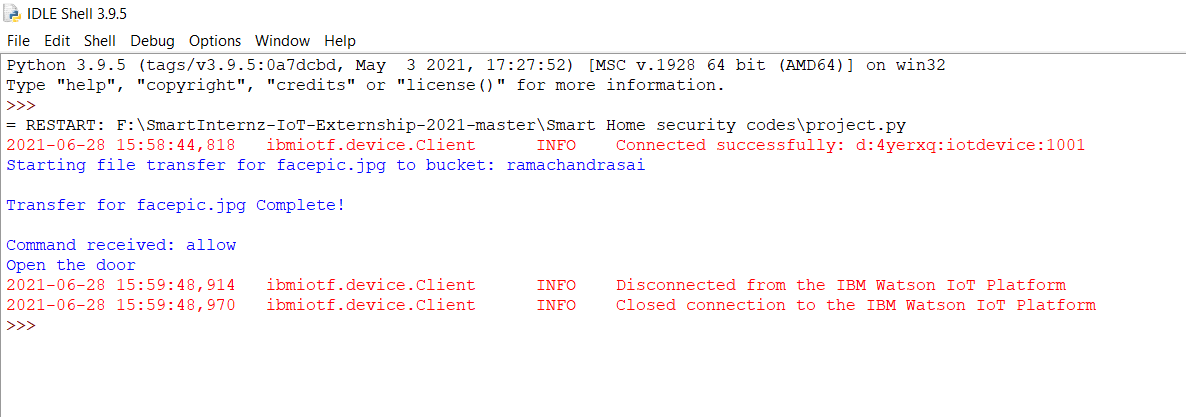
**5 Flow chart:**

**

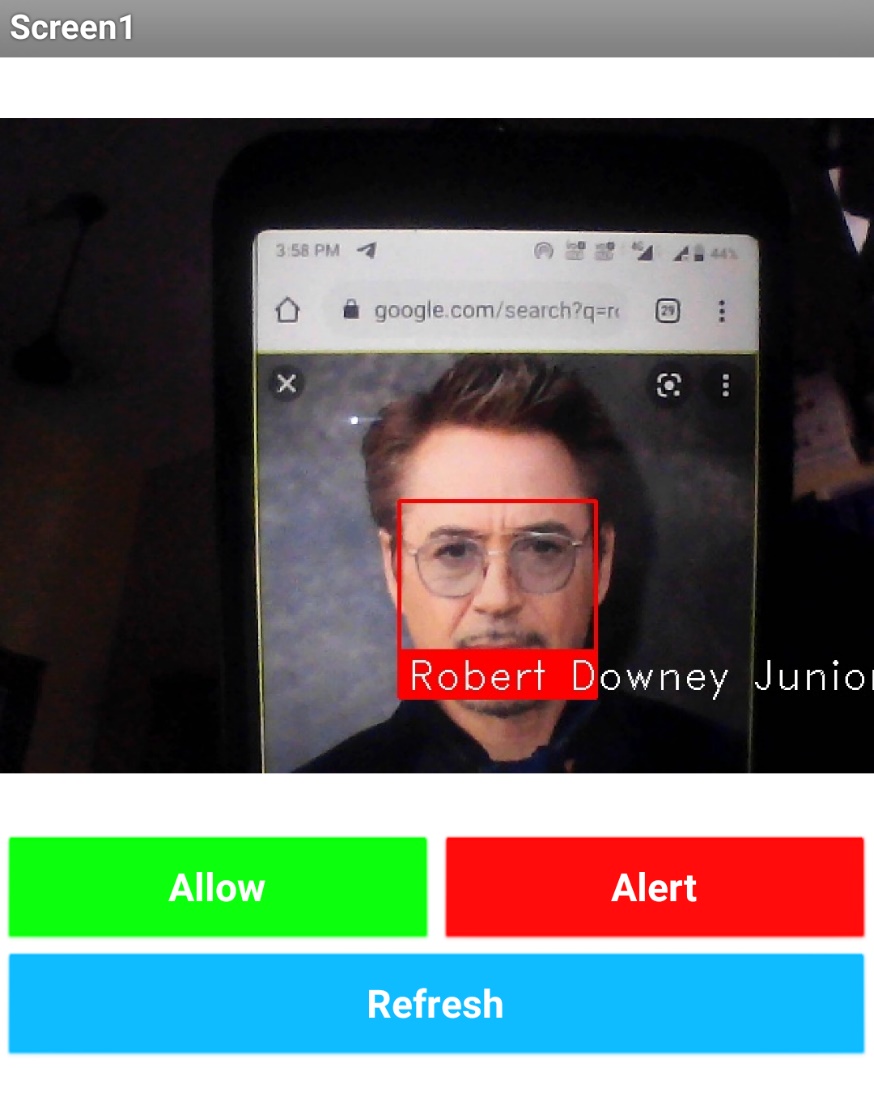
**6. Result:**

When ever a person is detected image is captured using python code and sent to cloud object storage. This image is retrived by using node red and displays in ui. Using android app we can control the access to the person. For authorised person door opens automatically with voice output of welcome message . If person is not authorised person owner can send alert message to respective persons.

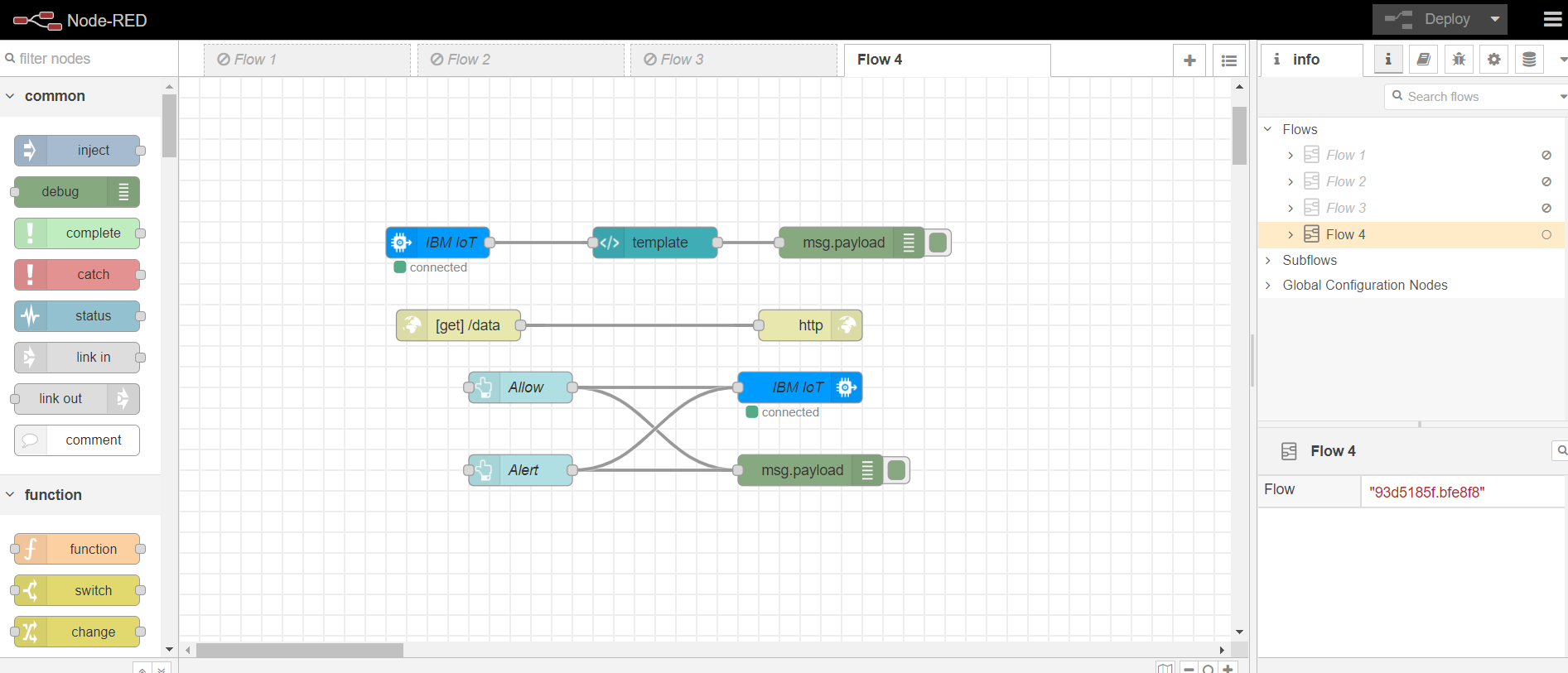
**Python code output**

**

**Android App**

**

**Node red**

**

**7. Advantages and Disadvantages**

**Advantages:**

* Security will be enhanced: through the smart guest identifier the control of persons entering their house/office will be under the owner. so that no other person can enter without their assent. Provides an additional layer of security beyond keys, fobs and cards.
* Automated identification: it allows the identification process to be automated, thus it saves time and increase accuracy.
* Cost efficient: smart guest identifier offers both security as well as automation at low cost.
* Easy to integrate: smart guest identifier is easy to integrate and most solutions are compatible with the majority of security software.
* Easy to access: The process of accessing is not much complicated. It is easy to understand and implement.
* Identity protection: Guarantee that only authorized individuals gain automatic access to secure locations, while requiring far less personal information than other forms of identification.
* Seamless convenience: Face recognition is as fast as it is accurate, instantly validating identity in order to protect access to sensitive areas.

**Disadvantages:**

Smart guest identifier provides several large benefits to users, it also holds various disadvantages, including:

* Detection is vulnerable: while face detection provides more accurate results than manual identification processes, it can also be more easily thrown off by changes in appearance or camera angles.
* The storage of sensitive personal data and the challenges that come with it is database containing facial scans may breach.
* The technology isn’t as effective at identifying of color and women as it is white males.

**8.Applications:**

* Office: Smart guest identifier can be used in offices to allow only their staff as authorized persons.
* Airports: In airports this application can be used to allow the persons who booked their tickets.
* Banks: In banks smart guest identifier can be used in highly secured lockers to enhance the security.
* Colleges: In colleges it can be used to allow the students and staff in to the college
* Schools: In schools it can be used to allow the students and staff in to the corresponding school.
* Hotels: in hotels this can be used in guest room so that it can allow only the room owners.
* Restaurants: In restaurants smart guest identifier can be used to allow the prebooked customers as authorized persons. And staff can also be authorized persons
* House: In house the owners can be considered as authorized persons and he can allow his guests.
* Apartments/hostels: in apartments only the persons who belongs to the apartment can enter
* Hospitals: In hospitals this can be used for highly secured rooms for vips
* Gatherings: smart guest identifier can be used in gatherings to allow only specific persons in family gatherings, meetings etc..,

**9.Conclusion:**

Face recognition technology has come a long way in the last twenty years. Today machines are able to automatically verify identity information for security tasks and for access control tasks. Smart guest identifier with remote access management can be implemented manually as well as automatically. These applications usually wok in controlled environments and recognition algorithms can the advantage of the environmental constrains to obtain high recognition accuracy. This smart guest identification technology work reliable, in widely varying conditions using information from single or multiple modalities.

**10.Future Scope:**

The proposed system does not make use intricate routing system architecture. Rather it uses simple algorithms in order to vanish the existing problems. The project can be further extended, to prevent the disadvantages and issues in present algorithm. This application can be much cheaper and beneficial as compared to the manual system. In future this project can be integrated with other systems and can be used in higher applications

**11.Bibliography:**

<https://partheniumprojects.com/7763-2/>

<https://www.researchgate.net/publication/312559421_IoT_based_smart_security_and_home_automation_system>

**12. Appendix:**

**12.1. Source Code:**

import face\_recognition

import cv2

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import sys

import time

#import pyttsx3

from gtts import gTTS

from playsound import playsound

import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

import json

#Provide your IBM Watson Device Credentials

organization = "4yerxq"

deviceType = "iotdevice"

deviceId = "1001"

authMethod = "token"

authToken = "1234567890"

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data['command'])

if cmd.data['command']=='allow':

print("Open the door")

elif cmd.data['command']=='alert':

print("Alert IS RECEIVED")

if cmd.command == "setInterval":

if 'interval' not in cmd.data:

print("Error - command is missing required information: 'interval'")

else:

interval = cmd.data['interval']

elif cmd.command == "print":

if 'message' not in cmd.data:

print("Error - command is missing required information: 'message'")

else:

print(cmd.data['message'])

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()

deviceCli.connect()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times

# This is a demo of running face recognition on live video from your webcam. It's a little more complicated than the

# other example, but it includes some basic performance tweaks to make things run a lot faster:

# 1. Process each video frame at 1/4 resolution (though still display it at full resolution)

# 2. Only detect faces in every other frame of video.

# PLEASE NOTE: This example requires OpenCV (the `cv2` library) to be installed only to read from your webcam.

# OpenCV is \*not\* required to use the face\_recognition library. It's only required if you want to run this

# specific demo. If you have trouble installing it, try any of the other demos that don't require it instead.

# Get a reference to webcam #0 (the default one)

video\_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

obama\_image = face\_recognition.load\_image\_file(r"G:/Robert.jpg")

obama\_face\_encoding = face\_recognition.face\_encodings(obama\_image)[0]

# Load a second sample picture and learn how to recognize it.

biden\_image = face\_recognition.load\_image\_file(r"G:/PierceBrosnan.jpg")

biden\_face\_encoding = face\_recognition.face\_encodings(biden\_image)[0]

name=""

biden\_image1 = face\_recognition.load\_image\_file("G:/18481A04d0.jpg")

biden\_face\_encoding1 = face\_recognition.face\_encodings(biden\_image1)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [

obama\_face\_encoding,

biden\_face\_encoding,

biden\_face\_encoding1

]

known\_face\_names = [

"Robert Downey Junior",

"Pierce Brosnan",

"Sai"

]

# Initialize some variables

face\_locations = []

face\_encodings = []

face\_names = []

process\_this\_frame = True

while True:

# Grab a single frame of video

ret, frame = video\_capture.read()

# Resize frame of video to 1/4 size for faster face recognition processing

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

# Convert the image from BGR color (which OpenCV uses) to RGB color (which face\_recognition uses)

rgb\_small\_frame = small\_frame[:, :, ::-1]

# Only process every other frame of video to save time

if process\_this\_frame:

# Find all the faces and face encodings in the current frame of video

face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

face\_names = []

for face\_encoding in face\_encodings:

# See if the face is a match for the known face(s)

matches = face\_recognition.compare\_faces(known\_face\_encodings, face\_encoding)

name = "Unknown"

#time.sleep(5000)

#cv2.imwrite('facepic.jpg',frame)

# If a match was found in known\_face\_encodings, just use the first one.

if True in matches:

first\_match\_index = matches.index(True)

name = known\_face\_names[first\_match\_index]

# Import the required module for text

# initialize Text-to-speech engine

face\_names.append(name)

process\_this\_frame = not process\_this\_frame

# Display the results

for (top, right, bottom, left), name in zip(face\_locations, face\_names):

# Scale back up face locations since the frame we detected in was scaled to 1/4 size

top \*= 4

right \*= 4

bottom \*= 4

left \*= 4

# Draw a box around the face

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# Display the resulting image

cv2.imshow('Video', frame)

cv2.imwrite('facepic.jpg',frame)

# play the speech

# Hit 'q' on the keyboard to quit!

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

break

# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()

# -\*- coding: utf-8 -\*-

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

#s3.us-south.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints

COS\_API\_KEY\_ID = "guft-62iXJ2do-8HAabO6u19JARQa5kMYvG2xtGFC4CU" # eg "W00YiRnLW4a3fTjMB-oiB-2ySfTrFBIQQWanc--P3byk"

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

COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/79bdb6666e7345609b949a5a9b23cfa1:27a4b271-6711-45f5-b48e-64129234732f::"

# 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 \* 10

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 25

# 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))

if name!="Unknown":

text = "Welcome"+name

language = 'en'

obj = gTTS(text=text, lang=language, slow=False)

obj.save("text.mp3")

playsound("text.mp3")

multi\_part\_upload("ramachandrasai","facepic.jpg","F:/SmartInternz-IoT-Externship-2021-master/Smart Home security codes/facepic.jpg")

# waiting for command from owner

i=0

while (i<20):

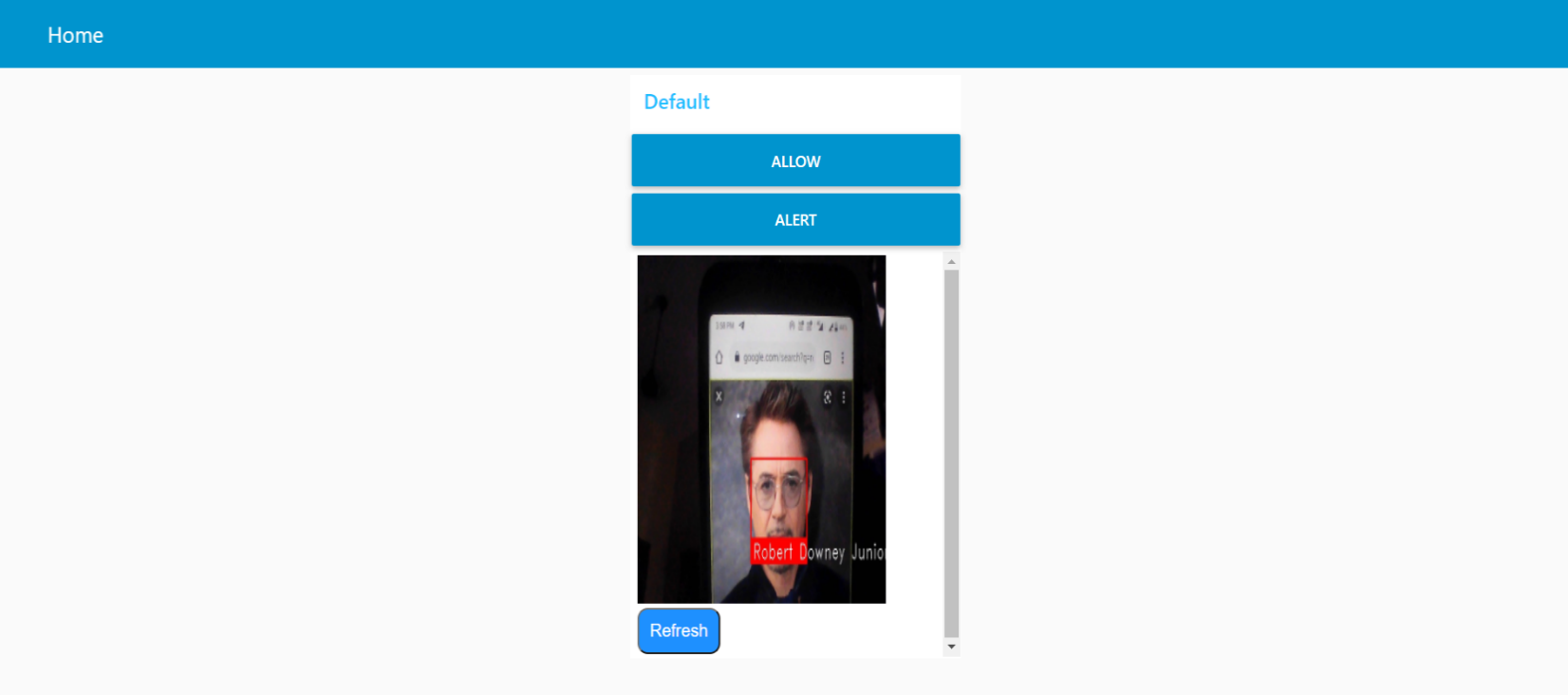
deviceCli.commandCallback = myCommandCallback

i+=1

time.sleep(2)

deviceCli.disconnect()

**12.2. UI Output Screenshot:**

**