# Security Camera System - Documentation

## Section 1 – Blog Link:

<https://www.notion.so/Embedded-Project-085f3d8bb71243fd8b51e931bfd857bc>

## Section 2 – Advanced Features:

In this project, three advanced features were chosen to make this project function as a security camera.

### Open-CV Computer Vision:

Open-CV is a python library that provides real-time computer vision and machine learning tools related to image capturing or video capturing. This module was used in this project to provide video recording and facial detection capabilities.

### Photoresistor:

This is a component that decreases its resistance depending on the light that is touching its surface. The photoresistor was used so the light level of the room which the camera is in would be displayed to whoever uses it.

### Flask Server:

The Flask Server is a micro web framework python library that offers tools for developing web applications. This library was used in this project to make a website in which it would display relevant information about the camera and its surroundings.

## Section 3 – Fritzing Diagram:

A circuit board with wires

Description automatically generated with low confidence

## Section 4 – Hardware/Software Challenges:

### Challenge:

A picture containing text, line, screenshot, receipt

Description automatically generatedWhen developing the email alert system of the project, whenever it tried to send an email with a video attachment, the attachment ends up not being properly sent. An email I had sent to myself through python should have an attachment called “test.mp4”, however, the attachment with the email is called “noname” instead.

### Solution:

A screen shot of a computer

Description automatically generated with low confidenceThis error was caused by a typo in the code that finalises the email attachment before trying to send it through my Gmail account.

In the attachment.package.add\_header line of code, “Content-Disposition” was written as “Content=Dispostion” instead, which was causing the filename to not get picked up. This is what was resulting the email to be sent with a corrupted attachment.

### Challenge:

The code using the opencv2 module was working without any errors, but the way I had implemented it meant that it would always override the same video if it had to record again. This was an issue as these videos had to be saved as different copies.

### Solution:

As a solution, I had created a function that contains a loop. This loop would keep looking for any instances of the file names like “video”, ”video1”, “video2” etc. Once it sees that there’s an instance that isn’t taken, it gives the recording the appropriate filename so it wouldn’t get overwritten.

### Challenge:

The buzzer is not running the code that has been set for it to execute. Everything when it comes to the wiring and the code should be fine, but the code still seems to not execute.

### Solution:

The pin variable in the code was not matching with the pin that the buzzer was connected to physically.

A picture containing text, font, screenshot, white

Description automatically generated

The buzzer variable had the number “27” instead of “26”, which is the reason to why the code was not executing when everything was fine.

### Challenge:

Values from the photoresistor wouldn’t change, no matter if you cover the top of it or flash a bright light over it. The wiring was checked, and it should be fine. The code was also checked, and it should have been fine as well.

A close-up of a circuit board

Description automatically generated with medium confidence

### Solution

To solve this problem regarding the photoresistor, a 1 uF capacitor had to be installed into the system to get the varying results that the photoresistor was meant to display for the project.

## Section 5 – Code:

### main.py code:

import cv2

import os

import datetime

import subprocess

from time import sleep

import RPi.GPIO as GPIO

from flask import Flask, redirect, url\_for, render\_template

from gpiozero import LightSensor

from send\_email import attachVideo

# Variables

led = 21 # Pin 40

switch = 20 # Pin 38

buzzer = 26 # Pin 37

resistor = 4 # Pin 7

face\_cascade = cv2.CascadeClassifier("/home/dale4e/Python/Home Assignment/haarcascade\_frontalface\_default.xml")

eye\_cascade = cv2.CascadeClassifier("/home/dale4e/Python/Home Assignment/haarcascade\_eye\_tree\_eyeglasses.xml")

defaultFile = "/home/dale4e/Python/Home Assignment/Videos/Video"

saveFile = defaultFile

cap = cv2.VideoCapture(0)

cap.set(cv2.CAP\_PROP\_BUFFERSIZE, 1)

cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH,960)

cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT,720)

w = cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

h = cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

fourcc = cv2.VideoWriter\_fourcc(\*'H264')

app = Flask(\_\_name\_\_)

@app.route("/")

# Functions

def home():

dt = getRecentVideoDate()

lt = getLightLevel()

vd = getVideoLength()

return render\_template("index.html", date=dt, light=lt, length=vd)

def setupGPIO():

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

GPIO.setup(led, GPIO.OUT)

GPIO.setup(buzzer, GPIO.OUT)

GPIO.setup(switch, GPIO.IN, pull\_up\_down = GPIO.PUD\_UP)

def setupVideo():

global out

global saveFile

index = 1

if(os.path.exists(defaultFile + ".mp4") == True):

while (os.path.exists(saveFile + ".mp4" ) == True):

saveFile = defaultFile + str(index)

index += 1

out = cv2.VideoWriter(saveFile + '.mp4',fourcc, 8.0, (int(w),int(h)))

def detectFace():

ret, frame = cap.read()

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

faces = face\_cascade.detectMultiScale(frame\_gray, 1.25, 4)

eyes = eye\_cascade.detectMultiScale(frame\_gray)

GPIO.output(led,0)

GPIO.output(buzzer,0)

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

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

rec\_gray = frame\_gray[y:y+h, x:x+w]

rec\_color = frame[y:y+h, x:x+w]

GPIO.output(buzzer,1)

GPIO.output(led,1)

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

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

GPIO.output(buzzer,1)

GPIO.output(led,1)

if ret == False:

print("Can't see.")

else:

out.write(frame)

cv2.imshow('Video Capture', frame)

key = cv2.waitKey(1)

def getLightLevel():

ldr = LightSensor(resistor)

lighting = ""

if(ldr.value\*100 >= 80):

lighting = "Very bright light"

if(ldr.value\*100 >= 70 and ldr.value\*100 < 80):

lighting = "Bright light"

elif(ldr.value\*100 >= 40 and ldr.value\*100 < 70):

lighting = "Medium light"

elif(ldr.value\*100 >= 10 and ldr.value\*100 < 40):

lighting = "Dim light"

elif(ldr.value\*100 < 10):

lighting = "No light"

return lighting

def getRecentVideoDate():

global saveFile

recentVideo = os.path.getctime(saveFile + ".mp4")

dateAndtime = datetime.datetime.fromtimestamp(recentVideo).date()

return dateAndtime

def getVideoLength():

global saveFile

video = cv2.VideoCapture(saveFile + ".mp4")

frames = video.get(cv2.CAP\_PROP\_FRAME\_COUNT)

fps = video.get(cv2.CAP\_PROP\_FPS)

seconds = round(frames/fps)

video\_time = datetime.timedelta(seconds=seconds)

return video\_time

def closeVideo():

print("Ending recording...")

cap.release()

out.release()

cv2.destroyAllWindows()

def closeGPIO():

print("Exiting GPIO ports..")

GPIO.output(led,0)

GPIO.output(buzzer,0)

GPIO.cleanup()

# Main Code

try:

setupGPIO()

setupVideo()

print("Press button to stop recording,")

while True:

if GPIO.input(switch) == True:

detectFace()

else:

closeVideo()

closeGPIO()

attachVideo(saveFile + '.mp4')

break

if \_\_name\_\_ == "\_\_main\_\_":

print("Loading website...")

app.run(host='192.168.0.10', port=5000)

except KeyboardInterrupt:

closeVideo()

closeGPIO()

if \_\_name\_\_ == "\_\_main\_\_":

print("Loading website...")

app.run(host='192.168.0.10', port=5000)

send\_email.py code:

import smtplib

import ssl

from os.path import basename

from email.mime.base import MIMEBase

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from email.mime.application import MIMEApplication

from email import encoders

host = "smtp.gmail.com"

port = "465"

login = "dalefenech2006@gmail.com"

password = "qbjx ivbb cyai upql"

context = ssl.create\_default\_context()

to = "dalefenech2006@gmail.com"

subject = "Recorded Video"

body = "Here is the most recent recorded video."

msg = MIMEMultipart()

msg['From'] = login

msg['To'] = to

msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

def attachVideo(filename):

attachment = open(filename,"rb")

attachment\_package = MIMEBase('application', 'octet-stream')

attachment\_package.set\_payload((attachment).read())

encoders.encode\_base64(attachment\_package)

attachment\_package.add\_header("Content-Disposition","attachment; filename=" + filename)

msg.attach(attachment\_package)

email = msg.as\_string()

with smtplib.SMTP\_SSL(host,port,context=context) as server:

server.login(login, password)

server.sendmail(login,to,email)