```python
import threading
import time
import string
import secrets
import digitalio
import board
import adafruit_matrixkeypad
from gpiozero import RGBLED, AngularServo
from guizero import App, Slider
from colorzero import Color
import adafruit_character_lcd.character_lcd_i2c as character_lcd
import busio
from adafruit_ht16k33 import segments
from imutils.video import VideoStream
from imutils import resize
from datetime import datetime, timedelta
import json
import requests
import cv2
import numpy as np



# ************************************************** User-Defined Functions ********************************************************************************************************************
def pin_check(pin,guess,sec,num,n):

    #Converts the digits entered list into a number
    guess = guess[0]*1000 + guess[1]*100 + guess[2]*10 + guess[3]

    #If the user entered the correct passcode
    if pin == guess:
        lcd.clear()
        rgb_led.color = Color(0, 255, 0 )

        #If the user entered the second password correctly
        if sec == True:
            lcd.message = 'Come in!'
            rgb_led.color = Color(0, 0, 255)
            servo_door.angle = 90

        #If the user entered the first password correctly
        else:
            sec = True
```

```python
            time.sleep(1.5)
            lcd.message ='Enter 2nd passcode: '
            n = 0
    #If the user entered the password incorrectly
    if pin != guess:
        lcd.clear()
        lcd.message = "Password Incorrect: Try Again"
        time.sleep(1.5)
        lcd.clear()
        lcd.message = 'Enter passcode: '

        #incorrect password counter
        n=n+1

        #If the user has entered the wrong password three consecutive times
        if n==3:
            lcd.clear()
            lcd.message = "Calling the cops!"
            time.sleep(1.5)

            #Displays a 5 second countdown on the 7-segment display
            for i in num:
                display.fill(0)
                display.print(str(i))
                time.sleep(1)
    return n, sec

def listToString(s):

    # initialize an empty string
    str1 = ""

    # traverse in the string
    for ele in s:
        str1 += ele

    # return string
    return str1



# **********************************************************************Initial Setup*********************************************************************************************************************************************

```

```
87  # Initialising I2C bus
88  i2c = board.I2C()  # uses board.SCL and board.SDA
89
90  # Creating the LED segment class. This creates a 7 segment 4 character display:
91  display = segments.Seg7x4(i2c)
92  display.fill(0)
93
94
95  # LCD Display setup
96  lcd_columns = 16
97  lcd_rows = 4
98  lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)
99
100 lcd.backlight = True
101
102
103 # Setting up the membrane 3x4 matrix keypad on Raspberry Pi -
    https://www.adafruit.com/product/419
104 cols = [digitalio.DigitalInOut(x) for x in (board.D26, board.D20, board.D21)]
105 rows = [digitalio.DigitalInOut(x) for x in (board.D5, board.D6, board.D13, board.D19)]
106
107 keys = ((1, 2, 3),
108         (4, 5, 6),
109         (7, 8, 9),
110         ("*", 0, "#"))
111
112 keypad = adafruit_matrixkeypad.Matrix_Keypad(rows, cols, keys)
113
114 # Setting up the RGB LED and initial color values
115 rgb_led = RGBLED(18, 23, 24)
116
117 red = 0
118 green = 0
119 blue=0
120
121 rgb_led.color = Color(255, 0, 0)
122
123
124 # Setting up the zero
125 servo_door = AngularServo(25, min_angle=-90, max_angle=90)
126 servo_door.angle = -90
127
128 # Setting up global variables
129 code=None
130 password_counter = 0
```

```
pin_gen= 5678
second_pin=False
numbers = [5,4,3,2,1]

pin_guess=[]
pin_base = input("Please input a 4-digit passcode:\t")
pin_base = int(pin_base)

lcd.message = 'System Secure!'
time.sleep(3)
lcd.clear()
lcd.message = 'Enter passcode:'


# ************************************************************Setting up the motion/face detection with uploading to google drive************************************************************************************
*

## Followed tutorial from https://www.youtube.com/watch?v=JwGzHitUVcU
## To generate Tokens: https://developers.google.com/oauthplayground/?code=4/0AVHEtk73eWOZvkVXY1H_fDGe-vpiEHhqh1vZSYUH7ZDQTaaK7nrvGbyTlOErsGYmQbxPnA&scope=https://www.googleapis.com/auth/drive
diff_threshold = 1000000
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
vs = VideoStream(src=0).start()

# Header dictionary with a recently generated token
header = {"authorization": "Bearer ya29.a0Ael9sCPc9xZgTMQpQ6d7indu3ywp8PD6pGAsaxJltC_QBBv0h8zzfoOOxYpPQO5Jkgnj20MV_Hx7krve_XBeZR0FDC2yw8WPltcUz6wLzUM9ff_IeNp2PoMDNApx_wFcp0BdkA2sXt2-wDvhPtYTrJM_aL3AaCgYKAWsSARASFQF4udJhs558nPcakgz9xOw95cA71Q0163"}

#parameter dictionary with the image file and url location of the folder on the Google Drive
param = {"name": "face.jpg",
        "parents": ['1AR3SLYY2wgiHyrXQBzQYqjTzHXERiog_']}

# Converts param dictionary to a json string in order to upload the photo to the Google drive.
files = {'data': ('metadata', json.dumps(param), 'application/json;charset=UTF-8'),
         'file':('face.jpg', open('face.jpg', 'rb'), 'image/jpeg')}

face_detected = False
```

```
start_face = None

while True:

    # Starts the camera, saves two images, converts to grayscale, blurs the images, and finds the absolute difference between them
    old_image = vs.read()
    old_image = cv2.cvtColor(old_image, cv2.COLOR_BGR2GRAY)
    old_image = cv2.blur(old_image, (20, 20))

    new_image = vs.read()
    new_image = cv2.cvtColor(new_image, cv2.COLOR_BGR2GRAY)
    new_image = cv2.blur(new_image, (20, 20))

    diff = cv2.absdiff(old_image, new_image)
    diff_score = np.sum(diff)

    #If the sum of the difference between the two images is greater than the difference threshold, then movement was detected
    if diff_score > diff_threshold:
        print("Movement detected")

    #Tries to determine if a face was able to be seen from the image
    faces = face_cascade.detectMultiScale(new_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    if len(faces) > 0:

        #If a face has been detected and and the face detection counter is set to None (has not started)
        if (not face_detected) and (start_face is None):

            #Begin the face detection timer
            start_face = datetime.now()
            print("Face detected")

            # Take picture and set face_detected flag to True
            cv2.imwrite("face.jpg", new_image)

            #Upload the picture to google drive using the header and files dictionaries that contain the json string and token
            face_detected = True
            r = requests.post(

                "https://www.googleapis.com/upload/drive/v3/files?uploadType=multipart",
```

```
                headers = header,
                files = files)

            if r.status_code == 200:
                print("File uploaded succesfully!")
            else:
                print("Failed to upload files :(")
    else:

        #If a face has not been detected
        if face_detected:
            face_detected = False

    #If the face detection timer is not none (has already begun) and it has been more than 1 minute since it has begun
    if (start_face is not None) and (datetime.now() > start_face + timedelta(minutes=1)):
        print("reset")

        #Reset the time
        start_face = None

    old_image = new_image

    #If the keypad has been pressed
    if keypad.pressed_keys:

        #appends to the pressed keys list
        pin_guess.append((keypad.pressed_keys)[0])

        #Clears the LCD screen and displays the keys that hav been entered
        lcd.clear()
        lcd.message = 'Code: {}'.format(pin_guess)
        print(pin_guess)
        time.sleep(0.2)

        #If 4 digits have been entered and the user is on the first passcode
        if (len(pin_guess)>3) & (second_pin == False):
            password_counter,second_pin = pin_check(pin_base,pin_guess,second_pin,numbers, password_counter)
            pin_guess=[]

        #If 4 digits have been entered and the user is on the second password
        if (len(pin_guess)>3) & (second_pin == True):
            password_counter,second_pin = pin_check(pin_gen,pin_guess,second_pin,
```

```
            numbers, password_counter)
245         pin_guess=[]
246
247
248
249
250
```