```python
'''        __                          __
          /'{>                        /'{>
      ____) (____                 ____) (____
    //'--;   ;--'\\             //'--;   ;--'\\
   ///////\_/\\\\\\\           ///////\_/\\\\\\\
          m m                         m m

****************************************************************
*                                                              *
*  EEET-418                                                    *
*  27 April 2022                                               *
*  Senior Project: Smart package Delivery Box                  *
*                                                              *
*  The purpose of this program is to act as the main pogram    *
*  for the SPDB project.                                       *
*                                                              *
*  Created by: Brandon Empie                                   *
*                                                              *
****************************************************************
'''
import threading #importing the threading module
import time #importing time module
from time import sleep #importing the sleep function from time module
import datetime
from datetime import datetime #importing the class datetime from the module
import serial
from picamera import PiCamera, Color
import boto3 #AWS SDK
from twilio.rest import Client #importing Client function from twilio.rest module
from BlinkM import * #importing BlinkM.py
from keypad import * #importing keypad.py
import vlc
import os
import sys
import traceback
import PySimpleGUI as sg #importing pysimpleGUI as an object
import PIL #python image library used to convert image from jpeg to PNG
from PIL import Image
import RPi.GPIO as GPIO


class glb(): #class used for global variables, avoids having to declare in each function
        tracking = ""
        courier = ""
        lock = 27
        sensorPin = 17
        currentHour = ""
        darkOut = 0
        ser = serial.Serial('/dev/ttyACM0', baudrate=9600, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE)
        camera = PiCamera()
        picTaken = 0
        boxOpen = 0
        boxClosed = 0
        boxFull = 0
        boxReady = 0
        s3NewName = ""
        scanner = 0
        keypadAttempt = 0
        verified = 0
        addrList = ["BRANDON EMPIE", "16405 110TH AVE", "RODNEY MI 49342"]
        bucket = 'spdb1'
        s3 = boto3.client('s3') #creating object s3 using boto3 (AWS SDK)
        account_sid = 'AC40cf178e455696603ce4c623eff527bb' #this is bad practice and should be stored as an env. variable
        auth_token = '24e1851dae35ff6faa41706ea8a1457b'    #due to being a prototype however it was done this way to save time
        twilioClient = Client(account_sid, auth_token) #using my account ID and auth. token to create constructor
def main():
        GPIO.setwarnings(False) #initilizing GPIO inputs and ouputs for the lock and sensor
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(glb.lock, GPIO.OUT)
        GPIO.setup(glb.sensorPin, GPIO.IN)
        GPIO.output(glb.lock, 0) #turning the lock output pin off at startup
        rek = boto3.client('rekognition') #creating rekognition object using the client
        glb.camera.resolution = (2592, 1944) #initializing camera resolution
        glb.camera.annotate_background = Color('black') #creating camera overlay
        glb.camera.annotate_foreground = Color('white')#creating camera overlay
        glb.camera.annotate_text_size = 48#creating camera overlay
        glb.camera.annotate_text = datetime.now().strftime('%Y-%m-%d %H:%M:%S')#creating camera overlay
        barcodeThread = threading.Thread(target=barcode, name='barcodethread') #creating barcode thread
        keypadThread = threading.Thread(target=keypd, name='keypadthread') #creating keypad thread
        displayThread = threading.Thread(target=analogDisplay, name='displaythread') #creating analogDisplay
        thread
        photoCellThread = threading.Thread(target=photoSensor, name='photoCellthread') #creating photoresistor
        thread
        glb.ser.close() #initializing serial port closed
        barcodeThread.start() #starting barcode thread
        keypadThread.start() #starting keypad thread
        displayThread.start() #starting display thread
        photoCellThread.start() #starting photo resistor thread
        StartSequence() #calling BlinkM start sequence so user knows box is ready for data
        while(True): #Starts main process
                if glb.scanner == 1: #if the barcode scanner flag has been raised
                        picture() #take the picture
                        glb.picTaken = 1 #raise flag to notify other threads a picture has been taken
                        glb.s3NewName = datetime.now().strftime('%Y-%m-%d %H:%M:%S') + ".jpeg" #sets picture
                        filename
                        glb.s3.upload_file('lastPicture.jpeg', glb.bucket, glb.s3NewName,
                        ExtraArgs={'ContentType': 'image/jpeg'}) #uploads picture to AWS S3
                        if glb.courier == "Not recognized": #if the courier was not identified
                                SolidColor(red) #calling BlinkM
                                glb.boxClosed = 1 #let display thread know
                                sleep(5) #wait 5 seconds also freeing up resources
                                ContactOwner(glb.verified) #contact the homeonwer via Twilio with verification
                                status
                                SolidColor(off) #calling BlinkM
                                glb.boxReady = 1 #notify display thread to go back to the home screen
                        else: #otherwise if the courier was recognized
                                SolidColor(cyan) #calling BlinkM
                                response =
                                rek.detect_text(Image={'S3Object':{'Bucket':glb.bucket,'Name':glb.s3NewName}})#us
                                ing rek client to analyze object in bucket
                                textDetections = response['TextDetections'] #storing the text results
                                SolidColor(orange) #calling BlinkM
                                extraction = open("lastExtraction.txt", "w")
                                for text in textDetections: #for each item in textDetections
                                        if text['Type'] == "LINE": #if the item is listed as a line (opposed to
                                        a word)
                                                extraction.write(text['DetectedText'] + "\n") #write it to text
                                                file adding new line characer to it
                                extraction.close() #close the text file now that we are done writing to it
                                verifyAddr() #calling verifyAddr() function
                                if glb.verified == 0:
                                        SolidColor(red) #calling BlinkM
                                        glb.boxClosed = 1
                                        ContactOwner(glb.verified)
                                        sleep(5)
                                        SolidColor(off) #calling BlinkM
                                        glb.boxReady = 1
                                elif glb.verified == 1:
                                        SolidColor(green) #calling BlinkM
                                        unlockSPDB()
                                        glb.boxOpen = 1
                                        sleep(5)
                                        SolidColor(off) #calling BlinkM
                                        glb.boxReady = 1
                                        ContactOwner(glb.verified)
                                        glb.verified = 0
                        if glb.darkOut == 1:
                                glb.darkOut = 0
                        glb.scanner = 0 #clearing flag for next scan
                if glb.keypadAttempt == 1: #if keypad flag has been raised
                        if glb.darkOut == 1: #if beacon is active shut it off
                            StopScript() #stop the script
                            SolidColor(off) #turn the BlinkM off
                        if glo.pinSequence == glo.masterPin: #if masterpin has been entered
                                SolidColor(green)
                                unlockSPDB()
                                glo.pinSequence = ""
                                glb.boxOpen = 1
                                sleep(5)
                                SolidColor(off)
                                glb.boxReady = 1
                        elif glo.pinSequence == glo.markedFull: #if the # A pin has been entered
                                FullHelper()
                                glb.boxFull = 1
                                glb.scanner = 1
                                glb.twilioClient.messages.create(
                    from_='+12316608834',
                    body = 'Your SPDB has been marked full on ' + datetime.now().strftime('%Y-%m-%d %H:%M:%S')
                    + '. Please empty to recieve new deliverys.',
                    to ='+16167995626')
                                glo.pinSequence = ""
                                while glo.pinSequence != glo.masterPin:
                                        readC1()
                                        readC2()
                                        readC3()
                                        readC4()
                                        if len(glo.pinSequence) == 8 and glo.pinSequence != glo.masterPin:
                                                StopScript()
                                                SolidColor(cyan)
                                                sleep(3)
                                                FullHelper()
                                                glo.pinSequence = ""
                                        elif glo.pinSequence.__contains__("C"):
                                                glo.pinSequence = ""
                                glb.scanner = 0
                                StopScript()
                                SolidColor(green)
                                unlockSPDB()
                                glb.boxOpen = 1
                                sleep(5)
                                SolidColor(off)
                                sleep(10)
                                glb.boxReady = 1
                        else:
                                SolidColor(red)
                                glb.boxClosed = 1
                                sleep(5)
                                glb.boxReady = 1
                                SolidColor(off)
                        glo.pinSequence = ""
                        glb.keypadAttempt = 0
                        if glb.darkOut == 1:
                                glb.darkOut = 0

        return
def unlockSPDB():
        GPIO.output(glb.lock, 1) #open lock - giving 3.3v output
        sleep(.2) #waiting 200ms
        GPIO.output(glb.lock, 0) #output to 0v - closing lock

def barcode():
        codein = "" #initilizing codein
        while(True): #thread will run forever
            codein = serialScan() #calling serialScan function to get barcode data
            #if data is already being processed, dump new, and block during non delivery hours
            if glb.scanner == 0 and glb.currentHour > "08" and glb.currentHour < "20":
                    if glb.darkOut == 1: #if beacon is active shut it off
                            StopScript() #stop the script
                            SolidColor(off) #turn the BlinkM off
                    HSBFade(pink) #calling BlinkM
                    prefix = codein[0] #taking first char. of the scanned code/keyboard input
                    if prefix == "]":# otherwise if barcode scanner prefix is detected then process data
                            glb.tracking = codein[1:] #splicing off the prefix
                            if len(glb.tracking) == 35:
                                    if glb.tracking[0:3] == "420": #checking for courier specific designation
                                            glb.tracking = glb.tracking[13:] #splicing off everything but the tracking number
                                            glb.courier = "USPS First-Class" #saving result for main thread processing
                                            glb.scanner = 1 #raising flag for main to process data
                            elif len(glb.tracking) == 34: #following code used to identify tracking number and courier
                                    if glb.tracking[0:2] == "96" : #checking for courier specific designation
                                            glb.tracking = glb.tracking[22:] #splicing off everything but the tracking number
                                            glb.courier = "FedEx Ground" #saving result for main thread processing
                                            glb.scanner = 1 #raising flag for main to process data
                            elif len(glb.tracking) == 18:
                                    if glb.tracking[0:2] == "1Z": #checking for courier specific designation
                                            glb.courier = "UPS Ground" #saving result for main thread processing
                                            glb.scanner = 1 #raising flag for main to process data
                            else:
                                    glb.courier = "Not recognized" #saving result for main thread processing
                                    glb.scanner = 1 #raising flag for main to process data
def picture():
    glb.camera.start_preview(fullscreen=False,window=(400,500,10,10)) #allows camera to start and focus for 2 seconds
    sleep(2)
    captureTime = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    glb.s3NewName = captureTime + ".jpeg" #setting picture filename for S3
    glb.camera.annotate_text = captureTime #setting timestamp on picture
    glb.camera.capture('lastPicture.jpeg') #saves temp picture
    glb.camera.stop_preview() #ends preview
    return

```

```
def verifyAddr():
        X = Y = Z = 0 #initializing component variables
        extraction = open("lastExtraction.txt", "r") #open extraction data for reading
        for line in extraction.readlines(): #for each line in the txt file
            line = line.upper() #remove case sensitive
            line = line.strip("\n") #remove new line character
            if line.find(",") > 0: #if there is a comma in address remove it
                line = line.replace(",","")
            if glb.addrList[0] in line: #test for the name
                    X = 1
            elif glb.addrList[1] in line: #test for street address
                    Y = 1
            elif glb.addrList[2] in line: #test for county,state, zip code
                    Z = 1
        if X == Y == Z == 1: #if all components of address are a match
                glb.verified = 1 #raise flag for main
        extraction.close() #close txt file now that were done with it
        return  #return to main
def ContactOwner(status):
        dateTaken = glb.s3NewName.strip(".jpeg") #stripping off the file name to get just the timestamp
        # generating a url for the picture that will only last for 5 seconds, key to maintaining security
        url = glb.s3.generate_presigned_url('get_object', Params={'Bucket': glb.bucket, 'Key': glb.s3NewName}, ExpiresIn=5)
        if status == 1: #if address was verified
                glb.twilioClient.messages.create(
                    from_='+12316608834',
                    body = 'Package delivered via ' + glb.courier + ' with tracking #: ' + glb.tracking + ' on ' + dateTaken,
                                    media_url=[url],
                    to ='+16167995626')
        elif status == 0: #if address was not verified
                glb.twilioClient.messages.create(
                    from_='+12316608834',
                    body = 'Package delivery attempt ' + glb.courier + ' on ' + dateTaken,
                                    media_url=[url],
                    to ='+16167995626')
def keypd():
        while True: #runs thread forever
                while glb.keypadAttempt == 0: #while a keypad attempt is not being processed
                        try:
                                readC1() #check column 1, one row at a time
                                readC2() #check column 2, one row at a time
                                readC3() #check column 3, one row at a time
                                readC4() #check column 4, one row at a time
                                if glo.pinSequence.__contains__("C"): #clear pinSequence if a C has been entered
                                        glo.pinSequence = ""
                                elif len(glo.pinSequence) == 8: #if 4 digits have been entered
                                        glb.keypadAttempt = 1 #notify main a pin has been entered
                                        sleep(5) #wait 10 seconds before the next pin can be entered
                                elif glo.pinSequence == glo.markedFull: #if the box has been marked full
                                        glb.keypadAttempt = 1 #notify main
                        except:
                                GPIO.cleanup() #cleans up ports if theres an error
def serialScan():
    data = "" #initializes local variable as empty string
    item = "" #intializes local variable as empty string
    glb.ser.open() #open serial port
    while item != '\r': #while a cariage return is not read
            item = glb.ser.read().decode("utf-8")#read serial port one byte at a time and decode it
            data += item #store the decoded byte and concatonate the string
    data = data.strip("\r") #remove it from the string
    glb.ser.close() #close serial port now that were done with it
    return(data) #return the scanned barcode back to 'barcode thread'
def analogDisplay():
    sg.theme('DarkAmber')   # sets the theme in window
    layout = [  [sg.Text('Ready to scan barcode.', key='-LINE1-')],
            [sg.Text('', key='-LINE2-')],
            [sg.Text('Hold lable still while', key='-LINE3-')],
            [sg.Text('light is pink...', key='-LINE4-')],
            [sg.Text('Enter: #A if SPDB is full', key='-LINE5-', text_color='cyan')],
            [sg.Text('', key='-PIN-', text_color='red')] ]

# Create the Window
    window = sg.Window('Window Title', layout, no_titlebar=True ,size=(720,480), font=("Helvetica",50),
                    finalize=True)
    window.bind("<Button-1>", 'Window Click')
    window.set_cursor("none") #hides the mouse arrow on the display
    # creating vlc media player object
    media_player = vlc.MediaPlayer()
    # toggling full screen
    media_player.toggle_fullscreen()
    # media object
    shrek2 = vlc.Media("Shrek2.mp4")
    nopass = vlc.Media("notpass.mp4")

    scrolledText = 'Enter: #A if SPDB is full or C to clear pin ' #43 characters
    first = 0
    last = 25
    endofline = 0
    tempString = ""
    newString = ""
    scroll = 1
    while True:
        event, values = window.read(timeout=10) #critical that the timeout be 10 seconds
        if event == sg.WIN_CLOSED or event == 'Window Click': # if user closes window or clicks cancel
            break
        if glo.pinSequence != "":
            window['-PIN-'].update('Pin: ' + glo.pinSequence)
        else:
            window['-PIN-'].update('')
        if glb.picTaken == 1: #if picture taken
            scroll = 0 #stop scrolling line
            im = Image.open('/home/pi/lastPicture.jpeg') #open the picture that was taken
            im.resize((720, 480),PIL.Image.BICUBIC).save('/home/pi/lastPicture.png') #resize it and convert to PNG for the display
            layout1 = [[sg.Image('/home/pi/lastPicture.png')]]
            pic = sg.Window('Window Title', layout1, no_titlebar=True ,size=(720,480), font=("Helvetica",50),
                   finalize=True)
            sleep(5)
            window['-PIN-'].update('')
            pic.close()
            glb.picTaken = 0 #clear for next picture
        elif glb.boxReady == 1:
                window['-LINE1-'].update('Ready to scan barcode.')
                window['-LINE2-'].update('')
                window['-LINE3-'].update('Hold lable still while')
                window['-LINE4-'].update('light is pink...')
                window['-LINE5-'].update('    Enter: # A if full')
                window['-PIN-'].update('')
                window.refresh()
                scroll = 1
                glb.boxReady = 0
        elif glb.boxFull == 1:
                scroll = 0
                window['-LINE1-'].update('         SPDB is full')
                window['-LINE2-'].update('')
                window['-LINE3-'].update('Please place deliverys')
                window['-LINE4-'].update('         in garage')
                window['-LINE5-'].update('')
                window['-PIN-'].update('')
                window.refresh()
                glb.boxFull = 0
        elif glb.boxOpen == 1: #if box is open goes here
                scroll = 0
                media_player.set_media(shrek2)
                media_player.play()
                sleep(3)
                media_player.stop()
                window['-PIN-'].update('')
                window['-LINE1-'].update('Remember to close lid')
                window['-LINE2-'].update('')
                window['-LINE3-'].update('     Have a nice day!')
                window['-LINE4-'].update('')
                window['-LINE5-'].update('')
                window.refresh()
                sleep(10)
                glb.boxOpen = 0
        elif glb.boxClosed == 1: #box is closed
                scroll = 0
                media_player.set_media(nopass)
                media_player.play()
                sleep(5)
                window['-PIN-'].update('')
                window['-LINE1-'].update('')
                window['-LINE2-'].update('')
                window['-LINE3-'].update('     Please try again  ')
                window['-LINE4-'].update('')
                window['-LINE5-'].update('')
                window.refresh()
                media_player.stop()
                sleep(5)
                glb.boxClosed = 0
        if scroll == 1:
                sleep(.2)
                if last >= 44:
                        first += 1
                        newString = newString + scrolledText[endofline]
                        tempstring = scrolledText[first:last] + newString
                        window['-LINE5-'].update(tempstring)
                        last +=1
                        endofline += 1
                        if first == 43:
                            first = 0
                            last = 25
                            endofline = 0
                            newString = ""
                else:
                        window['-LINE5-'].update(scrolledText[first:last])
                        first += 1
                        last += 1
                window.refresh()
    window.close()

def photoSensor():
        while True:
                sleep(15) #checking every 15 seconds
                glb.currentHour = time.strftime("%H")
                if glb.currentHour > "08" and glb.currentHour < "20": #if during delivery hours
                        if(GPIO.input(glb.sensorPin) and glb.darkOut == 0): #if input received its dark
                                LocationHelper()
                                glb.darkOut = 1
                        elif  glb.darkOut == 1 and  not (GPIO.input(glb.sensorPin)): #if its not dark anymore but was
                                        StopScript() #stop the beacon
                                        SolidColor(off)
                                        glb.darkOut = 0 #clear variable

try:
        main()
except:
        with open("exceptions.log", "w") as logfile: #logging any errors so they can be viewed in a text file
                traceback.print_exc(file=logfile)
        raise
```