```
import cv2

import dlib

import numpy as np

from scipy.spatial import distance

from imutils import face_utils

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.image import MIMEImage

import os

import serial

import time

from datetime import datetime

#import playsound

#import threading



# Setup Arduino serial connection

try:

   # Change COM port and baud rate as needed

   arduino = serial.Serial('COM5', 9600, timeout=1)

   print("Connected to Arduino")

   # Allow time for serial connection to establish

   time.sleep(2)

   ARDUINO_CONNECTED = True

except Exception as e:

   print(f"Failed to connect to Arduino: {e}")

   ARDUINO_CONNECTED = False
```

```
# Load face detector and landmark predictor

face_detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

landmark_predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


# Constants

EYE_AR_THRESH = 0.25  # Eye Aspect Ratio threshold

EYE_AR_CONSEC_FRAMES = 20  # Number of frames eyes must be closed

YAWN_THRESH = 30  # Yawn threshold based on mouth aspect ratio


# Email Configuration

SMTP_SERVER = "smtp.gmail.com"  # Change based on your email provider

SMTP_PORT = 587

SENDER_EMAIL = "shaikhaseena1603@gmail.com"  # Replace with your email

SENDER_PASSWORD = "pbzt hxzp ekwd jizj"  # Replace with your app password

RECEIVER_EMAILS = ["skhaseenagoldenrose@gmail.com"]



# Initialize variables

COUNTER = 0

YAWN_COUNTER = 0

ALARM_ON = False

LAST_EMAIL_TIME = None

EMAIL_COOLDOWN = 60  # Seconds between emails to prevent spam

ARDUINO_COOLDOWN = 5  # Seconds between Arduino alerts


# Function to play an alert sound

'''def sound_alarm():
    global ALARM_ON
    if not ALARM_ON:
        ALARM_ON = True
        playsound.playsound("alarm_clock.mp3")
        ALARM_ON = False'''

# Function to send commands to Arduino
def send_arduino_alert(alert_type='D'):
    if not ARDUINO_CONNECTED:
        return

    try:
        # Send 'D' for drowsiness detection
        arduino.write(alert_type.encode())
        print(f"Alert signal '{alert_type}' sent to Arduino")
    except Exception as e:
        print(f"Failed to send alert to Arduino: {e}")



# Function to send email with image
def send_alert_email(image, alert_type="Drowsiness"):
    global LAST_EMAIL_TIME

    # Check if we've sent an email recently
    current_time = datetime.now()
    if LAST_EMAIL_TIME is not None and (current_time - LAST_EMAIL_TIME).total_seconds()
< EMAIL_COOLDOWN:
        return
    LAST_EMAIL_TIME = current_time

    # Create timestamp for the image filename
    timestamp = current_time.strftime("%Y%m%d_%H%M%S")
    image_filename = f"{alert_type.lower()}alert{timestamp}.jpg"

    # Save the image temporarily
    cv2.imwrite(image_filename, image)

    try:
        # Create email
        msg = MIMEMultipart()
        msg['Subject'] = f"{alert_type.upper()} ALERT - {timestamp}"
        msg['From'] = SENDER_EMAIL

        # Email body
        body = MIMEText(
            f"Driver {alert_type.lower()} detected! Please check the attached image and take
appropriate action.")
        msg.attach(body)

        # Attach image
        with open(image_filename, 'rb') as img_file:
            img_data = img_file.read()
            image = MIMEImage(img_data)
            image.add_header('Content-Disposition', 'attachment', filename=image_filename)
            msg.attach(image)
        # Connect to SMTP server and send email
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(SENDER_EMAIL, SENDER_PASSWORD)

            for receiver_email in RECEIVER_EMAILS:
                msg['To'] = receiver_email
                server.send_message(msg)

        print(f"{alert_type} alert email sent to {len(RECEIVER_EMAILS)} recipients with image")

    except Exception as e:
        print(f"Failed to send email: {e}")

    finally:
        # Remove temporary image file
        if os.path.exists(image_filename):
            os.remove(image_filename)


# Calculate Eye Aspect Ratio (EAR)
def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
```

```
# Calculate Mouth Aspect Ratio (MAR) for yawning
def mouth_aspect_ratio(mouth):
    A = distance.euclidean(mouth[2], mouth[10])
    B = distance.euclidean(mouth[4], mouth[8])
    C = distance.euclidean(mouth[0], mouth[6])
    mar = (A + B) / (2.0 * C)
    return mar



# Start Video Capture
cap = cv2.VideoCapture(0)

# Initialize last alert time for Arduino
last_arduino_alert = None

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

    # If no faces detected, display message
    if len(faces) == 0:
        cv2.putText(frame, "No Face Detected", (50, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)

    for (x, y, w, h) in faces:
        face_rect = dlib.rectangle(x, y, x + w, y + h)
        shape = landmark_predictor(gray, face_rect)
        shape = face_utils.shape_to_np(shape)

        # Extract eye and mouth landmarks
        leftEye = shape[36:42]
        rightEye = shape[42:48]
        mouth = shape[48:68]

        # Compute EAR and MAR
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        mar = mouth_aspect_ratio(mouth)

        # Draw landmarks
        for (x, y) in np.concatenate((leftEye, rightEye, mouth), axis=0):
            cv2.circle(frame, (x, y), 2, (0, 255, 0), -1)

        # Display EAR and MAR values
        cv2.putText(frame, f"EAR: {ear:.2f}", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
        cv2.putText(frame, f"MAR: {mar:.2f}", (10, 60),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
        # Drowsiness Detection
        if ear < EYE_AR_THRESH:
            COUNTER += 1
            cv2.putText(frame, f"Eyes Closed: {COUNTER}", (10, 90),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                print("DROWSINESS ALERT!")
                cv2.putText(frame, "DROWSINESS ALERT!", (50, 150),
                            cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 3)
                '''if not ALARM_ON:
                    threading.Thread(target=sound_alarm, daemon=True).start()'''

                # Send alert to Arduino (with cooldown)
                current_time = time.time()
                if last_arduino_alert is None or (current_time - last_arduino_alert > 
ARDUINO_COOLDOWN):
                    send_arduino_alert('D')  # 'D' for drowsiness
                    last_arduino_alert = current_time
                # Capture image and send email alert
                send_alert_email(frame, "Drowsiness")
        else:
            COUNTER = 0
            cv2.putText(frame, "Eyes Open", (10, 90),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

        # Yawning Detection
        if mar > YAWN_THRESH:
            YAWN_COUNTER += 1
            cv2.putText(frame, "YAWNING DETECTED!", (50, 200),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (255, 0, 0), 3)

            # If yawning persists for multiple frames, send alert
            if YAWN_COUNTER >= 10:  # Adjust this threshold as needed
                # Different alert code for yawning (optional)
                current_time = time.time()
                if last_arduino_alert is None or (current_time - last_arduino_alert > 
ARDUINO_COOLDOWN):
                    send_arduino_alert('Y')  # 'Y' for yawning (you can implement this in Arduino 
code)
                    last_arduino_alert = current_time
                send_alert_email(frame, "Yawning")
                YAWN_COUNTER = 0
        else:
            YAWN_COUNTER = max(0, YAWN_COUNTER - 1)  # Decrease counter gradually

    # Add system status information
    cv2.putText(frame, "Status: Monitoring", (frame.shape[1] - 250, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)

    cv2.imshow("Drowsiness Detector", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# Clean up
cap.release()
cv2.destroyAllWindows()
```

```
if ARDUINO_CONNECTED:
    arduino.close()
```