7. CODING AND ITS IMPLEMENTATION

7.1 Source code:

```
from imutils import face_utils
from utils import *
import numpy as np
import pyautogui as pag
import imutils
import dlib

import cv2
MOUTH_AR_THRESH = 0.6
MOUTH_AR_CONSECUTIVE_FRAMES = 15
EYE_AR_THRESH = 0.19
EYE_AR_CONSECUTIVE_FRAMES = 15
WINK_AR_DIFF_THRESH = 0.04
WINK_AR_CLOSE_THRESH = 0.19
WINK_CONSECUTIVE_FRAMES = 10
MOUTH_COUNTER = 0
EYE_COUNTER = 0
WINK_COUNTER = 0
INPUT_MODE = False
EYE_CLICK = False
LEFT_WINK = False
RIGHT_WINK = False
SCROLL_MODE = False
ANCHOR_POINT = (0, 0)
WHITE_COLOR = (255, 255, 255)
YELLOW_COLOR = (0, 255, 255)
```

```
RED_COLOR = (0, 0, 255)
GREEN_COLOR = (0, 255, 0)
BLUE_COLOR = (255, 0, 0)
BLACK_COLOR = (0, 0, 0)
shape_predictor = "shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor)
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]



vid = cv2.VideoCapture(0)
resolution_w = 1366#PIXEL RESOLUTION OF WIDTH
resolution_h = 768#PIXEL RESOLUTION OF HEIGHT
cam_w = 640#CAMERA WIDTH
cam_h = 480#CAMERA HEIGHT
unit_w = resolution_w / cam_w
unit_h = resolution_h / cam_h

while True:
    _, frame = vid.read()#TO READ VIDEO
    frame = cv2.flip(frame, 1)#FRAME BY FRAME
    frame = imutils.resize(frame, width=cam_w, height=cam_h)
    #HERE WE ARE ADJUSTING OUR CAM TO READ FRAMES EXACTLY
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#CHANGIGN TO GRAY SCALE
    rects = detector(gray, 0)
  if len(rects) > 0:#HERE WRITING IF ELSE TO READ GRAY SCALE IMAGE SOME TIMES
IT WILL NOT CONVERT TO GRAY SCALE
        rect = rects[0]#GRAY SCALE IMAGE HAS VALUES IN NEGATIVE HENCE IF GRAY
SCALE IMAGE READING THROUGH ELSE BLOCK
```

```
else:
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        continue
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
    mouth = shape[mStart:mEnd]
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    nose = shape[nStart:nEnd]

    # Because I flipped the frame, left is right, right is left.
    temp = leftEye
    leftEye = rightEye
    rightEye = temp

    # Average the mouth aspect ratio together for both eyes
    mar = mouth_aspect_ratio(mouth)
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    diff_ear = np.abs(leftEAR - rightEAR)#CONVETING TO NUMERIC AND ABSOLUTE
VALUES
    nose_point = (nose[3, 0], nose[3, 1])
    mouthHull = cv2.convexHull(mouth)#drawing covexhull for mouth
    leftEyeHull = cv2.convexHull(leftEye)#drawing covexhull for left eye
    rightEyeHull = cv2.convexHull(rightEye)#drawing covexhull for right eye
```

```
cv2.drawContours(frame, [mouthHull], -1, YELLOW_COLOR, 1)#here yellow color contour is drawn across mouth
   cv2.drawContours(frame, [leftEyeHull], -1, YELLOW_COLOR, 1)#here yellow color contour is drawn across left eye
   cv2.drawContours(frame, [rightEyeHull], -1, YELLOW_COLOR, 1)#here yellow color contour is drawn across right eye

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

   if diff_ear > WINK_AR_DIFF_THRESH:

      if leftEAR < rightEAR:
         if leftEAR < EYE_AR_THRESH:
            WINK_COUNTER += 1

            if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
               pag.click(button='left')# this lines of code represent left eye wink and to move cursor

               WINK_COUNTER = 0

      elif leftEAR > rightEAR:
         if rightEAR < EYE_AR_THRESH:
            WINK_COUNTER += 1

            if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
               pag.click(button='right')# this lines of code represent right eye wink and to move cursor

               WINK_COUNTER = 0
      else:
         WINK_COUNTER = 0
   else:
      if ear <= EYE_AR_THRESH:
         EYE_COUNTER += 1
```

```
            if EYE_COUNTER > EYE_AR_CONSECUTIVE_FRAMES:
                SCROLL_MODE = not SCROLL_MODE
                # INPUT_MODE = not INPUT_MODE
                EYE_COUNTER = 0

                # nose point to draw a bounding box around it

        else:
            EYE_COUNTER = 0
            WINK_COUNTER = 0

    if mar > MOUTH_AR_THRESH:# this lines of code represent mouth, to move cursor
        MOUTH_COUNTER += 1

        if MOUTH_COUNTER >= MOUTH_AR_CONSECUTIVE_FRAMES:
            # if the alarm is not on, turn it on
            INPUT_MODE = not INPUT_MODE
            # SCROLL_MODE = not SCROLL_MODE
            MOUTH_COUNTER = 0
            ANCHOR_POINT = nose_point

    else:
        MOUTH_COUNTER = 0

    if INPUT_MODE:#here drawing the bounding box for nose
        cv2.putText(frame, "READING INPUT!", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
RED_COLOR, 2)
        x, y = ANCHOR_POINT
        nx, ny = nose_point
        w, h = 60, 35
        multiple = 1
```

```
    cv2.rectangle(frame, (x - w, y - h), (x + w, y + h), GREEN_COLOR, 2)#creating rectangle bounding box to move cursor point
    cv2.line(frame, ANCHOR_POINT, nose_point, BLUE_COLOR, 2)#line inside bounding box to attach the nose

    dir = direction(nose_point, ANCHOR_POINT, w, h)#directional attachment of nose point
    cv2.putText(frame, dir.upper(), (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
    drag = 18
    if dir == 'right':#when nose moves right with bounding box and line
        pag.moveRel(drag, 0)
    elif dir == 'left':#when nose moves left with bounding box and line
        pag.moveRel(-drag, 0)
    elif dir == 'up':#when nose moves up with bounding box and line
        if SCROLL_MODE:
            pag.scroll(40)
        else:
            pag.moveRel(0, -drag)
    elif dir == 'down':#when nose moves down with bounding box and line
        if SCROLL_MODE:
            pag.scroll(-40)
        else:
            pag.moveRel(0, drag)
```

```
    if SCROLL_MODE:
        cv2.putText(frame, 'SCROLL MODE IS ON!', (10, 60), cv2.FONT_HERSHEY_SIMPLEX,
0.7, RED_COLOR, 2)

    # cv2.putText(frame, "MAR: {:.2f}".format(mar), (500, 30),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Right EAR: {:.2f}".format(rightEAR), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Left EAR: {:.2f}".format(leftEAR), (460, 130),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Diff EAR: {:.2f}".format(np.abs(leftEAR - rightEAR)), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # Show the frame
    cv2.imshow("Frame", frame)#to see the face through cam and wait key is q
    key = cv2.waitKey(1) & 0xFF

    # If the `Esc` key was pressed, break from the loop
    if key == 27:
        break

# Do a bit of cleanup
cv2.destroyAllWindows()
vid.release()
```