

Додаток А

Код модулю відстеження руки

```
import cv2
import mediapipe as mp
import time
import math
import numpy as np

class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5,
trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode,
self.maxHands,
self.detectionCon,
self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        # print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
```

```

        if draw:
            self.mpDraw.draw_landmarks(img, handLms,

self.mpHands.HAND_CONNECTIONS)

        return img

def findPosition(self, img, handNo=0, draw=True):
    xList = []
    yList = []
    bbox = []
    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            # print(id, lm)
            h, w, c = img.shape
            cx, cy = int(lm.x * w), int(lm.y * h)
            xList.append(cx)
            yList.append(cy)
            # print(id, cx, cy)
            self.lmList.append([id, cx, cy])
            if draw:
                cv2.circle(img, (cx, cy), 5, (255, 0, 255),
cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

        if draw:
            cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax
+ 20, ymax + 20),

```

```

                                (0, 255, 0), 2)

    return self.lmList, bbox

def fingersUp(self):
    fingers = []
    # Thumb
    if self.lmList[self.tipIds[0]][1] >
self.lmList[self.tipIds[0] - 1][1]:
        fingers.append(1)
    else:
        fingers.append(0)

    # Fingers
    for id in range(1, 5):
        if self.lmList[self.tipIds[id]][2] <
self.lmList[self.tipIds[id] - 2][2]:
            fingers.append(1)
        else:
            fingers.append(0)

    # totalFingers = fingers.count(1)

    return fingers

def findDistance(self, p1, p2, img, draw=True, r=15, t=3):
    x1, y1 = self.lmList[p1][1:]
    x2, y2 = self.lmList[p2][1:]
    cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

    if draw:
        cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
        cv2.circle(img, (x1, y1), r, (255, 0, 255),

```

```

cv2.FILLED)
        cv2.circle(img, (x2, y2), r, (255, 0, 255),
cv2.FILLED)
        cv2.circle(img, (cx, cy), r, (0, 0, 255),
cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]

```

```

def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70),
cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)

```

```
if __name__ == "__main__":  
    main()
```

Додаток Б

Код програми розпізнавання та підрахунку пальців долоні

```
import cv2

import time

import os

import HandTrackingModule as htm

wCam, hCam = 640, 480

cap = cv2.VideoCapture(0)

cap.set(3, wCam)

cap.set(4, hCam)

folderPath = "FingerNumber"

myList = os.listdir(folderPath)

print(myList)

overlayList = []

for imPath in myList:

    image = cv2.imread(f'{folderPath}/{imPath}')

    # print(f'{folderPath}/{imPath}')

    overlayList.append(image)

print(len(overlayList))

pTime = 0

detector = htm.handDetector(detectionCon=0.75)

tipIds = [4, 8, 12, 16, 20]
```

```
while True:

    success, img = cap.read()

    img = detector.findHands(img)

    lmList = detector.findPosition(img, draw=False)

    # print(lmList)

    if len(lmList) != 0:

        fingers = []

        # Thumb

        if lmList[tipIds[0]][1] > lmList[tipIds[0] - 1][1]:

            fingers.append(1)

        else:

            fingers.append(0)

        for id in range(1, 5):

            if lmList[tipIds[id]][2] < lmList[tipIds[id] - 2][2]:

                fingers.append(1)

            else:

                fingers.append(0)

        # print(fingers)

        totalFingers = fingers.count(1)

        print(totalFingers)

        h, w, c = overlayList[totalFingers - 1].shape
```

```
img[0:h, 0:w] = overlayList[totalFingers - 1]

cv2.rectangle(img, (20, 225), (170, 425), (0, 255, 0),
cv2.FILLED)

cv2.putText(img, str(totalFingers), (45, 375),
cv2.FONT_HERSHEY_PLAIN,

            10, (255, 0, 0), 25)

cTime = time.time()

fps = 1 / (cTime - pTime)

pTime = cTime

cv2.putText(img, f'FPS: {int(fps)}', (400, 70),
cv2.FONT_HERSHEY_PLAIN,

            3, (255, 0, 0), 3)

cv2.imshow("Image", img)

cv2.waitKey(1)
```

Додаток В

Код програми управління налаштуваннями гучності

```
import cv2

import time

import numpy as np

import HandTrackingModule as htm

import math

from ctypes import cast, POINTER

from comtypes import CLSCTX_ALL

from pyaw.pyaw import AudioUtilities, IAudioEndpointVolume

wCam, hCam = 640, 480

cap = cv2.VideoCapture(0)

cap.set(3, wCam)

cap.set(4, hCam)

pTime = 0

detector = htm.handDetector(detectionCon=0.7)

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(

    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
```

```
volume = cast(interface, POINTER(IAudioEndpointVolume))
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = 0
volBar = 400
volPer = 0
while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList = detector.findPosition(img, draw=False)
    if len(lmList) != 0:
        x1, y1 = lmList[4][1], lmList[4][2]
        x2, y2 = lmList[8][1], lmList[8][2]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
        cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
        cv2.circle(img, (x2, y2), 15, (255, 0, 255), cv2.FILLED)
        cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 3)
        cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)
```

```
vol = np.interp(length, [50, 300], [minVol, maxVol])
volBar = np.interp(length, [50, 300], [400, 150])
volPer = np.interp(length, [50, 300], [0, 100])
print(int(length), vol)
volume.SetMasterVolumeLevel(vol, None)
if length < 50:
    cv2.circle(img, (cx, cy), 15, (0, 255, 0), cv2.FILLED)
cv2.rectangle(img, (50, 150), (85, 400), (255, 0, 0), 3)
    cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0),
cv2.FILLED)
    cv2.putText(img, f'{int(volPer)} %', (40, 450),
cv2.FONT_HERSHEY_COMPLEX,
    1, (255, 0, 0), 3)
cTime = time.time()
fps = 1 / (cTime - pTime)
pTime = cTime
    cv2.putText(img, f'FPS: {int(fps)}', (40, 50),
cv2.FONT_HERSHEY_COMPLEX,
    1, (255, 0, 0), 3)
cv2.imshow("Img", img)
cv2.waitKey(1)
```

Додаток Г

Код програми для управління курсором

```
import cv2
import numpy as np
import HandTrackingModule as htm
import time
import autopy

wCam, hCam = 640, 480
frameR = 100 # Frame Reduction
smoothing = 7

pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0

cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.handDetector(maxHands=1)
wScr, hScr = 2560, 1440
# print(wScr, hScr)

while True:

    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)

    if len(lmList) != 0:
```

```

x1, y1 = lmList[8][1:]
x2, y2 = lmList[12][1:]

fingers = detector.fingersUp()
# print(fingers)
cv2.rectangle(img, (frameR, frameR), (wCam - frameR,
hCam - frameR),
              (255, 0, 255), 2)

if fingers[1] == 1 and fingers[2] == 0:

    x3 = np.interp(x1, (frameR, wCam - frameR), (0,
wScr))
    y3 = np.interp(y1, (frameR, hCam - frameR), (0,
hScr))

    clocX = plocX + (x3 - plocX) / smoothening
    clocY = plocY + (y3 - plocY) / smoothening

    autopy.mouse.move(wScr - clocX, clocY)
    cv2.circle(img, (x1, y1), 15, (255, 0, 255),
cv2.FILLED)
    plocX, plocY = clocX, clocY

if fingers[1] == 1 and fingers[2] == 1:

    length, img, lineInfo = detector.findDistance(8, 12,
img)
    print(length)

```

```
    if length < 40:
        cv2.circle(img, (lineInfo[4], lineInfo[5]),
                    15, (0, 255, 0), cv2.FILLED)
        autopy.mouse.click()

    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
    cv2.putText(img, str(int(fps)), (20, 50),
cv2.FONT_HERSHEY_PLAIN, 3,
                (255, 0, 0), 3)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```