BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM
KHOA ĐIỆN – ĐIỆN TỬ

HCMUTE

BÁO CÁO CUỐI KỲ

ỨNG DỤNG HANDTRACKING TRONG XỬ LÝ ẢNH ĐỂ ĐIỀU KHIỂN THIẾT KHÔNG CHẠM

MÔN HỌC: XỬ LÝ ẢNH
MÃ MÔN HỌC : IMPR432466_04UTExMC
NHÓM THỰC HIỆN: 01
HỌC KỲ II – NĂM HỌC 2021-2022
GIẢNG VIÊN HƯỚNG DẪN: TS.NGUYỄN VĂN THÁI

TP.HỒ CHÍ MINH, Tháng 6 năm 2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN XỬ LÝ ẢNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập-Tự do- Hạnh phúc

TPHCM , Tháng 6 năm 2022

DANH SÁCH NHÓM VIẾT BÁO CÁO CUỐI KỲ
MÔN XỬ LÝ ẢNH
HỌC KỲ II NĂM HỌC 2020-2021

1. Mã lớp môn học: *IMPR432466_04UTExMC*
2. Giảng viên hướng dẫn: TS. Nguyễn Văn Thái
3. Tên đề tài: *Ứng dụng HandTracking trong xử lý ảnh để điều khiển thiết bị không chạm.*
4.Nhóm thực hiện: 01
5. Danh sách nhóm viết báo cáo cuối kỳ:

STT	HỌ VÀ TÊN	MSSV	TỶ LỆ HOÀN THÀNH	SỐ ĐIỆN THOẠI
1	Âu Đoàn Trung	20151201	100%	0907.664.912
2	Trương Khương Duy	20151451	100%	0911.312.237
3	Phạm Phi Long	20151507	100%	0869.181.365

Ghi chú:

Tỷ lệ % = 100%

Nhận xét của Giảng Viên:

……
……
……
……
……
……
……
……
……
……
……

……
……
……
……
……
……
……

Ngày…Tháng…Năm 2022

Giảng viên chấm điểm

LỜI CẢM ƠN

“Để hoàn thành báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến:

Ban giám hiệu trường Đại Học Sư phạm Kỹ Thuật Thành phố Hồ Chí Minh vì đã tạo điều kiện về cơ sở vật chất với hệ thống thư viện hiện đại, đa dạng các loại sách, tài liệu thuận lợi cho việc tìm kiếm, nghiên cứu thông tin.

Xin cảm ơn giảng viên bộ môn – TS. Nguyễn Văn Thái đã giảng dạy tận tình, chi tiết để chúng em có đủ kiến thức và vận dụng chúng vào bài báo cáo này.

Do chưa có nhiều kinh nghiệm làm để tài cũng như những hạn chế về kiến thức, trong bài báo cáo chắc chắn sẽ không tránh khỏi những thiếu sót. Rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ phía Thầy để bài báo cáo được hoàn thiện hơn.

Lời cuối cùng,chúng em xin kính chúc Thầy nhiều sức khỏe, thành công và hạnh phúc.”

MỤC LỤC

PHẦN MỞ ĐẦU

1. Đặt vấn đề:

Việt Nam vừa phải đi qua một cơn địa chấn với những mất mát chưa từng có mang tên " Covid-19". Đại dịch Covid-19 đã tạo ra 4 làn sóng lớn tấn công sâu rộng trên toàn quốc với gần 1,7 triệu người nhiễm SARS-CoV-2 và hơn 31 nghìn người tử vong. Chứng kiến sự tàn khốc của đại dịch, sự quá tải của hệ thống y tế khi số ca nhiễm Covid-19 tăng nhanh khủng khiếp, bệnh nhân từ nhẹ chuyển nặng và tử vong chỉ trong thời gian ngắn, chắc hẳn chúng ta ai cũng phải thốt lên rằng: "Những gì chứng kiến có lẽ đã đủ đau thương cho cả đời người". Nhìn lại 2 năm qua, nước ta trải qua nhiều tổn thương, mất mát, có lẽ sẽ cần nhiều và thật nhiều thời gian để chúng ra bù đắp và xoa dịu nỗi đau này.

Hiện nay đại dịch covid đã được kiểm soát tuy nhiên chúng ta vẫn cần phải đề phòng đại dịch Covid-19 sẽ quay trở lại một lần nữa. Một trong những nguyên nhân phổ biến dẫn đến tình trạng lây nhiễm SARS-CoV2 một cách kinh khủng là do sự tiếp xúc giữa người -người, người - vật.

Để khắc phục vấn đề này, nhóm chúng em đã nghiên cứu và ứng dụng công nghệ điều khiển không chạm – một trong những công nghệ phổ biến của thời đại công nghệ 4.0 hiện nay, phần mềm này sẽ phần nào khắc phục được sự tiếp xúc trực tiếp khi dịch bùng nổ, ngoài ra còn ứng dụng để điều khiển các thiết bị trong công nghiệp.

2. Mục tiêu đề tài:

Đề tài tìm hiểu những ứng dụng của xử lý ảnh trong thực tế, áp dụng những gì đã học vào thực tiễn. Mục tiêu nghiên cứu và tìm hiểu của nhóm nhằm hướng tới là:

+ Tìm hiểu những kiến thức và ứng dụng của xử lý ảnh trong cuộc sống.
+ Tìm hiểu về công nghệ điều khiển không chạm (cụ thể là Handtracking)
+ Xây dựng thuật toán và đưa vào ứng dụng phần mềm điều khiển không chạm (HandTracking) kết hợp với Arduino.

3. Nội dung nghiên cứu:

+ Tìm hiểu những kiến thức cơ bản và nâng cao về xử lý ảnh
+ Tìm hiểu về ngôn ngữ lập trình Python ..
+ Tìm hiểu về HandTracking .
+ Tìm hiểu về các thuật toán nhận diện .
+ Tìm hiểu về sự giao tiếp giữa Python và Arduino.
+ Tìm hiểu về thuật toán điều khiển thiết bị.
+ Viết chương trình .
+ Đánh giá kết quả thực hiện .

4. Giới hạn

Đặc tính của hệ thống xử lý ảnh thường bị ảnh hương bởi nhiều yếu tố khác nhau. Trong điều kiện thực tế cho phép, nhóm thực hiện đề tại trong một số điều kiện giới hạn sau:

+ Điều kiện thu nhận hình ảnh vào ban ngày và ban đêm khác nhau, cho nên nhóm chỉ tìm hiểu trong điều kiện ánh sáng ổn định.
+ Đối với nhận dạng, thì khoảng cách nhận dạng từ camera đến đối tượng không quá 0.5 mét, trên khoảng cách này thì việc nhận dạng có thể không được chính xác.
+ Do hạn chế về việc học online, nhóm chỉ tập trung vào xử lý các thuật toán xử lý ảnh đơn giản.
+ Kết hợp Arduino và thuật toán Python ở mức .

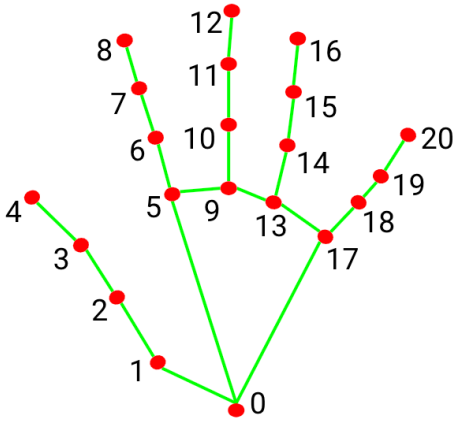
5. Bố cục đề tài:

Đề tài gồm có 4 chương.

+ Chương 1: Giới thiệu Module HandTracking.
+ Chương 2: Giao tiếp Python và Arduino.
+ Chương 3: Phần code và giải thích hoạt động chương trình.
+ Chương 4: Nhận xét và đánh giá kết quả.

CHƯƠNG 1: GIỚI THIỆU MODULE HANDTRACKING

Mediapipe cung cấp các model có độ chính xác rất tốt và độ trễ rất ít trên cả máy tính để bàn và thiết bị di động. Nó cung cấp mô hình 3D Hand Landmark sử dụng kỹ thuật máy học để dự đoán 21 điểm từ một khung hình duy nhất và có thể hoạt động trên máy tính để bàn, thiết bị di động hoặc trình duyệt, v.v. Dưới đây là chi tiết về tất cả các điểm mà mediapipe cung cấp cho một tay.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Nó có độ trễ rất thấp ngay cả trên các thiết bị cpu và có thể cung cấp kết quả rất tốt. Chúng tôi sẽ sử dụng python để xử lý hình ảnh hoặc khung hình từ video hoặc webcam.

Gói Mediapipe có thể được cài đặt bằng lệnh này.

```
2   import mediapipe as mp
```

Nó lấy hình ảnh đầu vào dưới dạng mảng numpy và cung cấp các model khác nhau dựa trên độ phức tạp về độ chính xác và tốc độ. Giống như các dòng máy khác, chúng ta có thể sử dụng chế độ hình ảnh tĩnh hoặc khi sử dụng video, chúng ta có thể đặt nó thành false để theo dõi các điểm mốc trong các khung hình khác cho phù hợp. Đây là danh sách các tùy chọn cấu hình mà chúng ta có thể sử dụng với nó.

- **STATIC_IMAGE_MODE** : Nếu đầu vào là một hình ảnh, chúng tôi đặt nó thành true, nếu không thì đặt false để theo dõi khung hình.
- **MAX_NUM_HANDS** : Số tay tối đa trong khung, mặc định là 2.
- **MODEL_COMPLEXITY** : Hai model 0 hoặc 1 trong đó 1 cung cấp kết quả tốt hơn 0.
- **MIN_DETECTION_CONFIDENCE** : Độ tin cậy khi phát hiện.
- **MIN_TRACKING_CONFIDENCE** : Nếu theo dõi khung, thì theo dõi độ tin cậy.

Đầu ra cho hình ảnh đầu vào có thể chứa các giá trị này

- **MULTI_HAND_LANDMARKS** : Các landmark phát hiện hoặc được theo dõi dưới dạng danh sách.
- **MULTI_HAND_WORLD_LANDMARKS** : Tọa độ 3D trong thế giới thực.
- **MULTI_HANDEDNESS** : Phát hiện tay trái hoặc tay phải cùng với điểm số.

Độ thuận tay được xác định giả sử hình ảnh đầu vào được phản chiếu, tức là được chụp bằng máy ảnh mặt trước / selfie với hình ảnh được lật theo chiều ngang. Nếu không phải như vậy, chúng ta cần lật ảnh trước khi nhập vào mô hình.

Bây giờ chúng ta có thể nhập các gói yêu cầu và tạo một đối tượng suy luận mô hình. Nó tìm nạp mô hình từ internet và tự động tải vào bộ nhớ khi tải.

```
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands

# For static images:
mp_model = mp_hands.Hands(
    static_image_mode=True, # only static images
    max_num_hands=2, # max 2 hands detection
    min_detection_confidence=0.5) # detection confidence

# we are not using tracking confidence as static_image_mode is true.
```

Bây giờ chúng tôi xử lý tất cả các kết quả từ mô hình và trực quan hóa và vẽ trên hình ảnh.

```
# Get handedness
print(results.multi_handedness)
```

```
[
    classification {
        index: 1
        score: 0.9940045475959778
        label: "Right"
    }
]
```

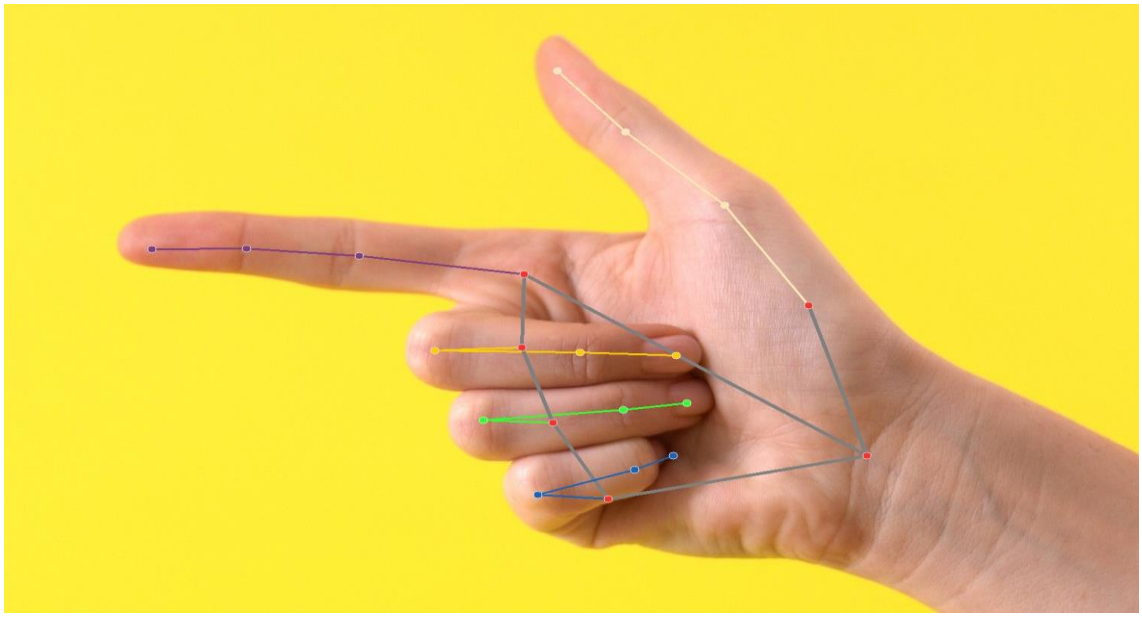
Bây giờ chúng tôi có thể lấy các điểm mốc từ danh sách các điểm mốc và chúng tôi in một số điểm mốc và vẽ trên hình ảnh bằng cách sử dụng các tiện ích vẽ trung gian. Để nhận danh sách tất cả các điểm mốc bàn tay có thể có, chúng ta có thể sử dụng mp_hands.HandLandmark cung cấp danh sách các nhãn điểm. Các điểm đầu ra Mediapipe ở đầu ra chuẩn hóa và chúng ta phải nhân tất cả các giá trị x với chiều rộng hình ảnh và tất cả các giá trị y với chiều cao hình ảnh.

```
image_height, image_width, c = image.shape # get image shape
# iterate on all detected hand landmarks
for hand_landmarks in results.multi_hand_landmarks:
      # we can get points using mp_hands
      print(f'Ring finger tip coordinates: (',
          f'{hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_TIP].x * image_width}, '
          f'{hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_TIP].y * image_height})'
      )
```

```
Ring finger tip coordinates: ( 1339.2894973754883, 1302.6893091201782)
```

Bây giờ chúng ta có thể vẽ trên hình ảnh bằng cách sử dụng các utils bản vẽ của mediapipe.

```
for hand_landmarks in results.multi_hand_landmarks:
    mp_drawing.draw_landmarks(
        image, # image to draw
        hand_landmarks, # model output
        mp_hands.HAND_CONNECTIONS, # hand connections
        mp_drawing_styles.get_default_hand_landmarks_style(),
        mp_drawing_styles.get_default_hand_connections_style())
```



CHƯƠNG 2 : GIAO TIẾP PYTHON VỚI ARDUINO

Arduino là một nền tảng mã nguồn mở bao gồm phần cứng và phần mềm cho phép phát triển nhanh chóng các dự án điện tử tương tác. Sự xuất hiện của Arduino đã thu hút sự chú ý của các chuyên gia từ nhiều ngành khác nhau, góp phần vào sự khởi đầu của Phong trào Maker .

Arduino sử dụng ngôn ngữ lập trình của riêng nó, tương tự như C ++ . Tuy nhiên, có thể sử dụng Arduino với Python hoặc một ngôn ngữ lập trình cấp cao khác. Trên thực tế, các nền tảng như Arduino hoạt động tốt với Python, đặc biệt là đối với các ứng dụng yêu cầu tích hợp với cảm biến và các thiết bị vật lý khác.

Chúng ta sử dụng thư viện cvzone để truyền dữ liệu từ Python qua sang Arduino.

```
9   from cvzone.SerialModule import SerialObject
```

Và trong Arduino IDE

```
serialData.begin();
```

Kết nối với COM3 của máy tính.

```
20    arduino = SerialObject("COM3")
```

Chúng ta cần truyền 3 giá trị về cho arduino để điều khiển loa, led đỏ , led xanh và cần 3 chữ số để biểu diễn cho mỗi giá trị.

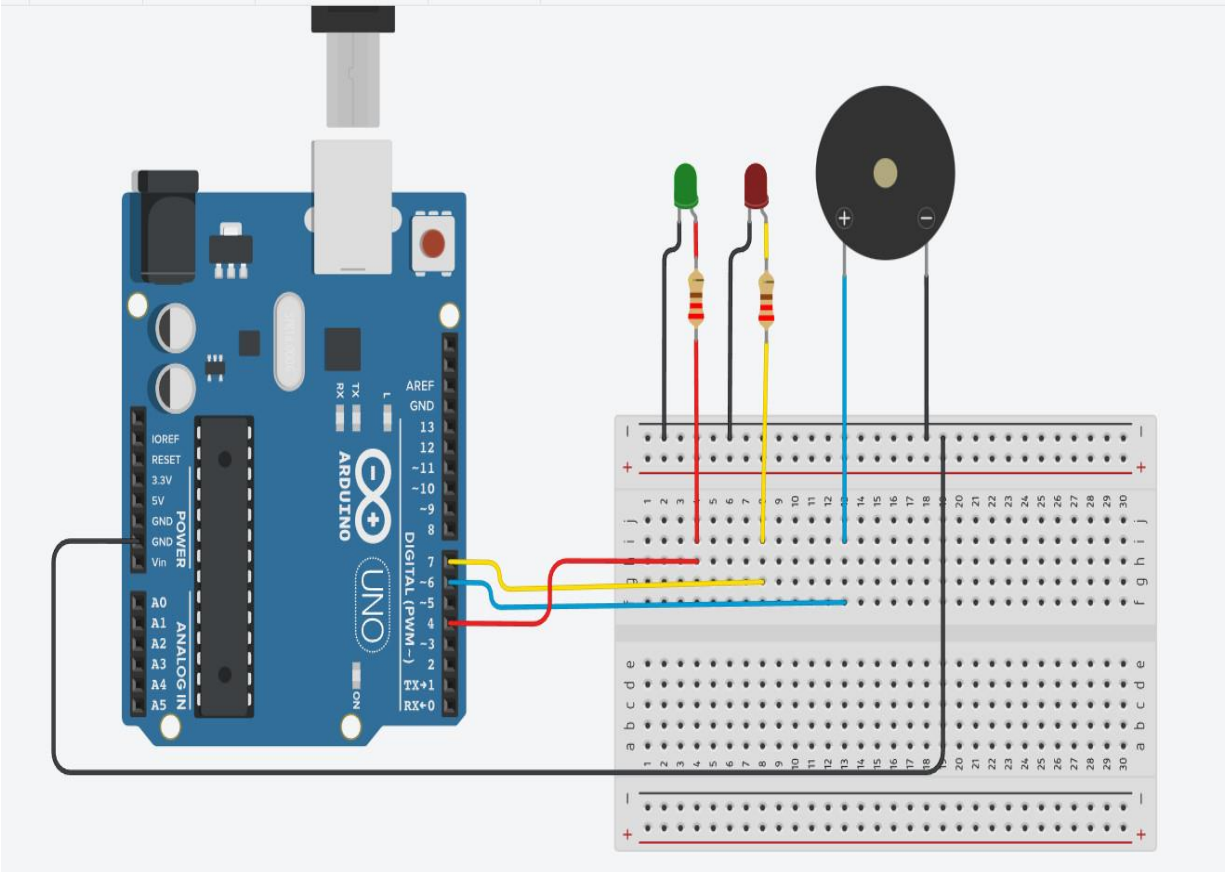
```
#include <cvzone.h>
SerialData serialData(3,3);
int valsRec[3];
```

Python sẽ gửi về chuỗi $abcdefghj (9 chữ số phía sau) và arduino sẽ tách thành 3 giá trị(abc,def,ghj) mỗi giá trị gồm 3 chữ số. Và chúng ta dùng các giá trị này để điều khiển loa và các led.

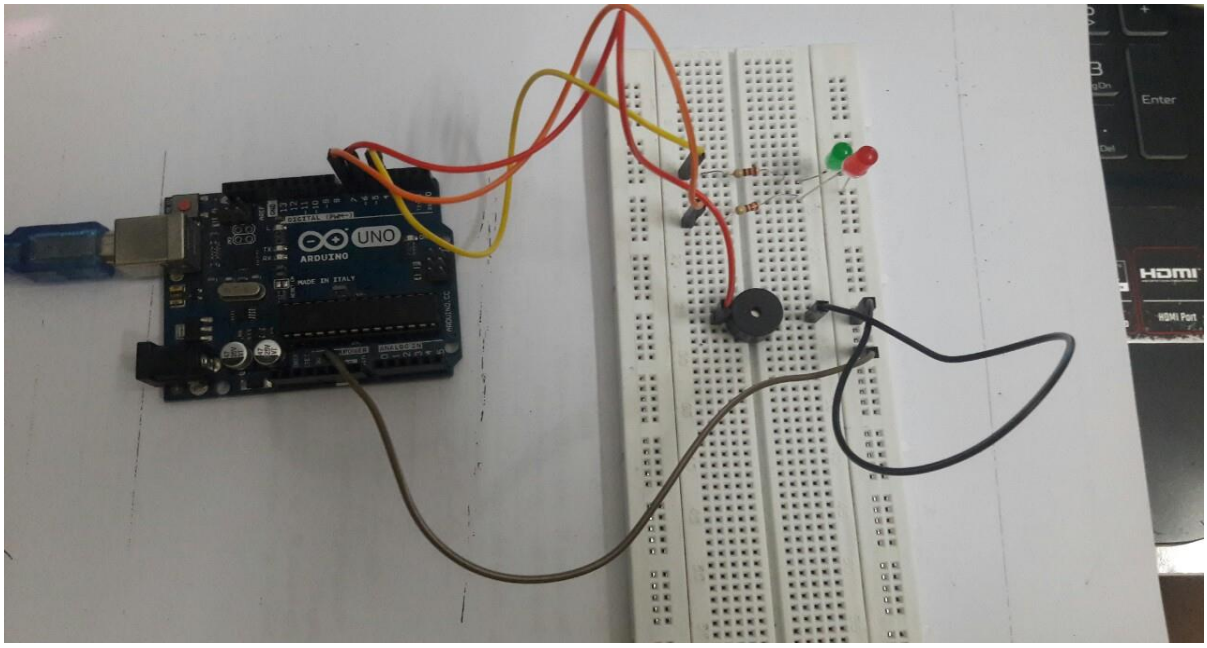
```
analogWrite(coi, valsRec[0]);
digitalWrite(ledred, valsRec[1]);
digitalWrite(ledgreen2, valsRec[2]);
```

Nếu như có biến thay đổi giá trị thì ta nhân cho số thập phân tương ứng.

```
68          arduino.sendData([volcoi*1000000+1])
```



Mô phỏng trên Tinkercad



KẾT NỐI THỰC TẾ

CHƯƠNG 3: PHẦN CODE VÀ GIẢI THÍCH HOẠT ĐỘNG CHƯƠNG TRÌNH

3.1 Phần CODE chương trình.

3.1.2 Module_HandTracking.py

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

class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        #Số tay tối đa trong khung, mặc định là 2
        self.maxHands = maxHands
        #Độ tin cậy khi phát hiện
        self.detectionCon = detectionCon
        #Nếu theo dõi khung, thì theo dõi tin cậy
        self.trackCon = trackCon
        #Các hàm sử dụng khi dùng Handtracking
        self.mpHands = mp.solutions.hands
        #Hàm trả về tọa độ các landmark.
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        #List các đầu ngón tay
        self.tipIds = [4, 8, 12, 16, 20]

    #Hàm phát hiện bàn tay.
    def findHands(self, img, draw=True):
        #Xử lý Handstracking làm việc với ảnh RGB nên chúng ta chuyển BGR sang RGB
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        #Nếu mà nhận diện được bàn tay
        if self.results.multi_hand_landmarks:
            #Quét lần lượt 21 điểm ảnh
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    #Nối các Landmark lại với nhau
                    self.mpDraw.draw_landmarks(img,
handLms,self.mpHands.HAND_CONNECTIONS)
        return img

```

```
    #Hàm tìm vị trí bàn tay
    def findPosition(self, img, handNo=0, draw=True):
        #Các mảng x,y chứa tọa độ của các Landmark
        xList = []
        yList = []
        #Mảng bbox chứa tọa độ của đường biên hình chữ nhật.
        bbox = []
        #Mảng chứa tọa độ của các Landmark
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            #Xét lần lượt các điểm landmark
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                #Nhân cho kích thước của các cạnh camera để hiện tọa độ theo tỉ lệ
                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:
                    #Vẽ các vòng tròn màu vào các điểm có tọa độ cx cy các landmark
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
            #Tìm 4 tọa độ x y của 4 đỉnh hình chữ nhật biên
            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            #Add vào list bbox
            bbox = xmin, ymin, xmax, ymax
            #Vẽ hình chữ nhật biên nhưng rộng ra thêm 20 đơn vị
            if draw:
                cv2.rectangle(img, (bbox[0] - 20, bbox[1] - 20),
            (bbox[2] + 20, bbox[3] + 20), (0, 255, 0), 2)
        #Trả về list các tọa độ landmark và list đường biên
        return self.lmList,bbox
    #Hàm kiểm tra ngón tay up down
    def fingersUp(self):
        fingers = []
        # Ngón cái - Thì xét tọa độ x của đầu ngón cái có lớn hơn điểm số 3 không.
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
        # 4 ngón còn lại - Tọa độ y của các đầu ngón tay nhỏ hơn 2 đốt dưới ko.
        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)
        return fingers
    #Hàm tìm khoảng cách
    def findDistance(self, p1, p2, img, draw=True):
        #Lấy tọa độ x,y của các điểm đầu ngón tay
        x1, y1 = self.lmList[p1][1], self.lmList[p1][2]
        x2, y2 = self.lmList[p2][1], self.lmList[p2][2]
        #Tìm tọa độ trung điểm và làm tròn.
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            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)
        #Hàm tính khoảng cách giữa 2 điểm trong math.
        length = math.hypot(x2 - x1, y2 - y1)
        return length, img, [x1, y1, x2, y2, cx, cy]

```

```python
def main():
    pTime = 0
    cTime =0
    #Sử dụng camera của thiết bị laptop
    cap = cv2.VideoCapture(0)
    #Gọi class handetector
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])
        #Tính độ phân giải
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        #Hiển thị độ phân giải.
        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()
```

3.1.2 Final_Prject.py

```
import cv2
import time
import numpy as np
import Hand_TrackingModule as htm
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
from cvzone.SerialModule import SerialObject
from time import sleep
from tkinter import *
from PIL import ImageTk,Image

def Touchless_device_control():
    wCam, hCam = 640, 480

    cap = cv2.VideoCapture(0)
    #Set kích thước cam
    cap.set(3, wCam)
    cap.set(4, hCam)
    pTime = 0
    arduino = SerialObject("COM3")
    detector = htm.handDetector(detectionCon=0.7, maxHands=1)

    volBar = 400
    volPer = 0
    volcoi=0
    area = 0
    colorVol = (255, 0, 0)

    while True:
        success, img = cap.read()

        # Find Hand
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img, draw=True)
        if len(lmList) != 0:

            # Filter based on size
            area = (bbox[2] - bbox[0]) * (bbox[3] - bbox[1]) // 100
            if 250 < area < 1000:
                # Tìm khoảng cách giữa ngón trỏ và ngón cái.
                length, img, lineInfo = detector.findDistance(4, 8, img)

                # Chuyển đổi theo tỷ lệ: Thanh âm lượng, phần trăm âm lượng và âm lượng.
                volBar = np.interp(length, [50, 200], [400, 150])
                volPer = np.interp(length, [50, 200], [0, 100])
                volcoi = int(np.interp(length, [50, 200], [0, 255]))

                # Giảm độ phân giải để làm cho nó mượt mà hơn.
                smoothness = 10
                volPer = smoothness * round(volPer / smoothness)

                # Kiểm tra ngón tay thẳng
                fingers = detector.fingersUp()

                # Nếu ngón út down , nếu ngón áp út down và các TH còn lại.
                if not fingers[4]:
```

```
                    #Còi reo mức cực đại và đèn đỏ sáng cộng tín hiệu SOS!!!
                    cv2.putText(img, "SOS!!!", (50, 100),
                    cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 255), 3)
                    cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0),
cv2.FILLED)
                    colorVol = (0, 255, 0)
                    arduino.sendData([255001000])
                elif not fingers[3]:
                    cv2.putText(img, "NEED", (50, 100),
                    cv2.FONT_HERSHEY_PLAIN, 3, (0, 255, 0), 3)
                    cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0),
cv2.FILLED)
                    colorVol = (0, 255, 0)
                    arduino.sendData([volcoi*1000000+1])
                else:
                    #Tắt hết còi và đèn.
                    colorVol = (255, 0, 0)
                    arduino.sendData([0,0,0])


        # Drawings
        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)

        # Tính độ phân giải.
        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("Nhan dien ho tro benh nhan can giup do", img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
                break
#Chương trình giao diện tương tác
window = Tk()
#Set up kích thước cửa số giao diện
window.geometry("1000x562")
#Đặt tên cho cửa sổ giao diện
window.title("Touchless device control with image processing")
#Lấy hình ảnh đưa lên giao diện
anh=Image.open("Anhgiaodien.jpg")
#Resize lại kích thước để phù hợp với giao diện
resize_Image=anh.resize((1000,562))
#Đưa ảnh lên giao diện Tkinter
image_bia= ImageTk.PhotoImage(resize_Image)
#Đặt tên cho giao diện ảnh
Img_1=Label(image=image_bia)
#Đặt vị trí cho giao diện Tkinter
Img_1.grid(column=0,row=0)
#Tạo nút nhấn trên giao diện Tkinter
button=Button(window,text="Start",font=("Times New
Roman",20),bg='silver',fg='yellow',command= Touchless_device_control)
#Đặt vị trí cho nút nhấn
button.place(relx=0.45,rely=0.53)
#Vòng lặp vô tận và dừng lại khi nhấn close
window.mainloop()
```

3.2 Giải thích chương trình.

Chương trình Module Handtracking được viết riêng để có thể sử dụng tiếp tục cho các dự án khác liên quan đến Handtracking bằng cú pháp:

```
import Hand_TrackingModule as htm
```

3.2.1 Cài đặt các thư viện cần thiết.

Các thư viện ở chương trình Module HandTracking.

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

Các thư viện ở chương trình chính.

```
import cv2
import time
import numpy as np
import Hand_TrackingModule as htm
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
from cvzone.SerialModule import SerialObject
from time import sleep
```

*Thư viện cv2:

OpenCV là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực.

Image/video I/O, xử lý, hiển thị (core, imgproc, highgui) Phát hiện các vật thể (objdetect, features2d, nonfree) Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab) Computational photography (photo, video, superres) Machine learning & clustering (ml, flann) CUDA acceleration (gpu).

*Thư viện numpy

Numpy (Numeric Python): là một thư viện toán học rất phổ biến và mạnh mẽ của Python. NumPy được trang bị các hàm số đã được tối ưu, cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng Python đơn thuần.

*Thư viện Mediapipe (đã trình bày ở chương 1).

*Cvzone :

Đây là gói Thị giác máy tính giúp dễ dàng chạy các chức năng Xử lý hình ảnh và AI. Về cốt lõi, nó sử dụng các thư viện OpenCV và Mediapipe.

3.2.2 Các hàm trong Module Handtracking

Đây là một số khai báo mặc định thông thường khi sử dụng thư viện Mediapipe để theo dõi cử chỉ ngón tay.

```
class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        #Số tay tối đa trong khung, mặc định là 2
        self.maxHands = maxHands
        #Độ tin cậy khi phát hiện
        self.detectionCon = detectionCon
        #Nếu theo dõi khung, thì theo dõi tin cậy
        self.trackCon = trackCon
        #Các hàm sử dụng khi dùng Handtracking
        self.mpHands = mp.solutions.hands
        #Hàm trả về tọa độ các landmark.
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        #List các đầu ngón tay
```

Tạo 1 mảng chứa các chỉ số của các LandMark đầu ngón tay.

```
        #List các đầu ngón tay
        self.tipIds = [4, 8, 12, 16, 20]
```

Hàm này trả về là một ảnh chứa các Landmark đã được phát hiện và vẽ các kết nối các Landmark với nhau. Quá trình xử lý ở đây dùng cho ảnh RGB nên chúng ta chuyển ảnh BGR sang RGB.

```
    #Hàm phát hiện bàn tay.
    def findHands(self, img, draw=True):
        #Xử lý Handstracking làm việc với ảnh RGB nên chúng ta chuyển BGR sang RGB
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        #Nếu mà nhận diện được bàn tay
        if self.results.multi_hand_landmarks:
            #Quét lần lượt 21 điểm ảnh
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    #Nối các Landmark lại với nhau
                    self.mpDraw.draw_landmarks(img, handLms,self.mpHands.HAND_CONNECTIONS)
        return img
```

Hàm findPosition trả về một list chứa 21 tọa độ (x,y) của các Landmark và một list chứa tọa độ của các đường biên. Hàm này quét lần lượt 21 Landmark và ghi nhận tọa độ của chúng và Lmlist. Ở đây chúng ta phải nhân kích thước của camera cho các tọa độ (0-1) để có được tọa độ theo tỉ lệ camera. Sau đó để tìm biên thì ta lấy các giá trị xmin, xmax ,ymin, ymax của các tọa độ trong Lmlist. Để vẽ hình chữ nhật biên dễ nhìn hơn thì ta tăng thêm 20 đơn vị cho chúng.

```python
    #Hàm tìm vị trí bàn tay
    def findPosition(self, img, handNo=0, draw=True):
        #Các mảng x,y chứa tọa độ của các Landmark
        xList = []
        yList = []
        #Mảng bbox chứa tọa độ của đường biên hình chữ nhật.
        bbox = []
        #Mảng chứa tọa độ của các Landmark
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            #Xét lần lượt các điểm landmark
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                #Nhân cho kích thước của các cạnh camera để hiện tọa độ theo tỉ lệ
                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:
                    #Vẽ các vòng tròn màu vào các điểm có tọa độ cx cy các landmark
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)
```

Hàm kiểm tra ngón tay Up-Down.Hàm này trả về một mảng chứa 5 giá trị 0 hoặc 1 tượng trưng cho Up và Down.Trường hợp đặc biệt là ngón cái chúng ta không xét theo trục y mà chúng ta xét theo trục x. Đối với bàn tay phải thì khi tọa độ x của đầu ngón cái mà lớn hơn đốt thì ngón cái được xem như khép lại. Bốn ngón còn lại chúng ta xét theo trục y. Kiểm tra tọa độ y của các đầu ngón tay có phía dưới đốt ngón tay không.

```python
    #Hàm kiểm tra ngón tay up down
    def fingersUp(self):
        fingers = []
        # Ngón cái - Thì xét tọa độ x của đầu ngón cái có lớn hơn điểm số 3 không.
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
        # 4 ngón còn lại - Tọa độ y của các đầu ngón tay nhỏ hơn 2 đốt dưới ko.
        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)
        return fingers
```

Hàm tìm khoảng cách trả về khoảng cách giữa hai điểm có tọa độ p1,p2,ảnh đã vẽ đường nối hai landmark và tọa độ trung điểm.

```python
    #Hàm tìm khoảng cách
    def findDistance(self, p1, p2, img, draw=True):
        #Lấy tọa độ x,y của các điểm đầu ngón tay
        x1, y1 = self.lmList[p1][1], self.lmList[p1][2]
        x2, y2 = self.lmList[p2][1], self.lmList[p2][2]
        #Tìm tọa độ trung điểm và làm tròn.
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            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)
        #Hàm tính khoảng cách giữa 2 điểm trong math.
        length = math.hypot(x2 - x1, y2 - y1)
        return length, img, [x1, y1, x2, y2, cx, cy]
```

```
        #Tính độ phân giải
        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime
        #Hiển thị độ phân giải.
        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
        (255, 0, 255), 3)
```

Thuật toán tính và hiển thị tốc độ khung hình (Frame rate).

3.2.3 Các thuật toán trong Final_Project.py

Đầu tiên chúng ta set kích thước camera và truyền số 0 vào VideoCapture để sử dụng camera của thiết bị.

```
wCam, hCam = 640, 480
################################

cap = cv2.VideoCapture(0)
#Set kích thước cam
cap.set(3, wCam)
cap.set(4, hCam)
```

Tính diện tích đường bao và đặt điều kiện giới hạn để tránh nhiễu. Tiếp theo chúng ta dùng hàm np.interp để chuyển đổi tỷ lệ giữa khoảng cách thành âm lượng và tỷ lệ %.

```
        area = (bbox[2] - bbox[0]) * (bbox[3] - bbox[1]) // 100
        if 250 < area < 1000:
            # Tìm khoảng cách giữa ngón trỏ và ngón cái.
            length, img, lineInfo = detector.findDistance(4, 8, img)

            # Chuyển đổi theo tỷ lệ: Thanh âm lượng, phần trăm âm lượng và âm lượng.
            volBar = np.interp(length, [50, 200], [400, 150])
            volPer = np.interp(length, [50, 200], [0, 100])
            volcoi = int(np.interp(length, [50, 200], [0, 255]))
```

Giảm độ phân giải để điều chỉnh âm lượng loa được mượt hơn.

```
            # Giảm độ phân giải để làm cho nó mượt mà hơn.
            smoothness = 10
            volPer = smoothness * round(volPer / smoothness)
```

Chúng ta kiểm tra trạng thái của ngón út (chỉ số 4). Nếu down thì gửi cho arduino tín hiệu để cài đặt âm lượng loa tối đa(255) và đèn báo đỏ sáng(001) và đèn xanh tắt(000).

```
        # Nếu ngón út down , nếu ngón áp út down và các TH còn lại.
        if not fingers[4]:
            #Còi reo mức cực đại và đèn đỏ sáng công tín hiệu SOS!!!
            cv2.putText(img, "SOS!!!", (50, 100),
            cv2.FONT_HERSHEY_PLAIN, 3, (0, 0, 255), 3)
            cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)
            colorVol = (0, 255, 0)
            arduino.sendData([255001000])
```

Dùng elif để kiểm tra trường hợp tiếp theo là trạng thái của ngón áp út (chỉ số 3). Nếu down thì gửi cho arduino tín hiệu để bật đèn xanh(001), tắt đèn đỏ (000) và âm lượng lúc này phụ thuộc và khoảng giữa 2 ngón trỏ và cái.

Và nếu không thuộc trường hợp nào trên thì tắt còi và các đèn.

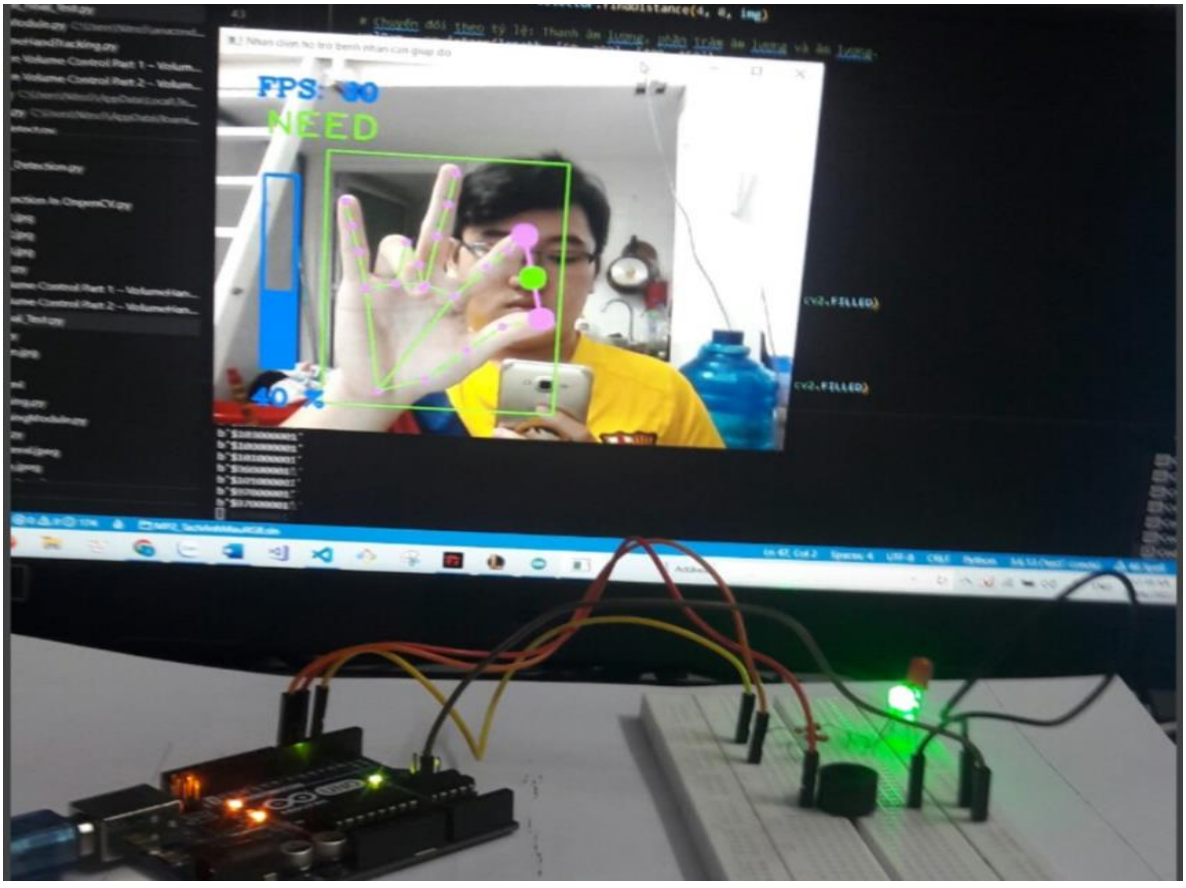
CHƯƠNG 4: NHẬN XÉT VÀ ĐÁNH GIÁ KẾT QUẢ

4.1 Kết quả.

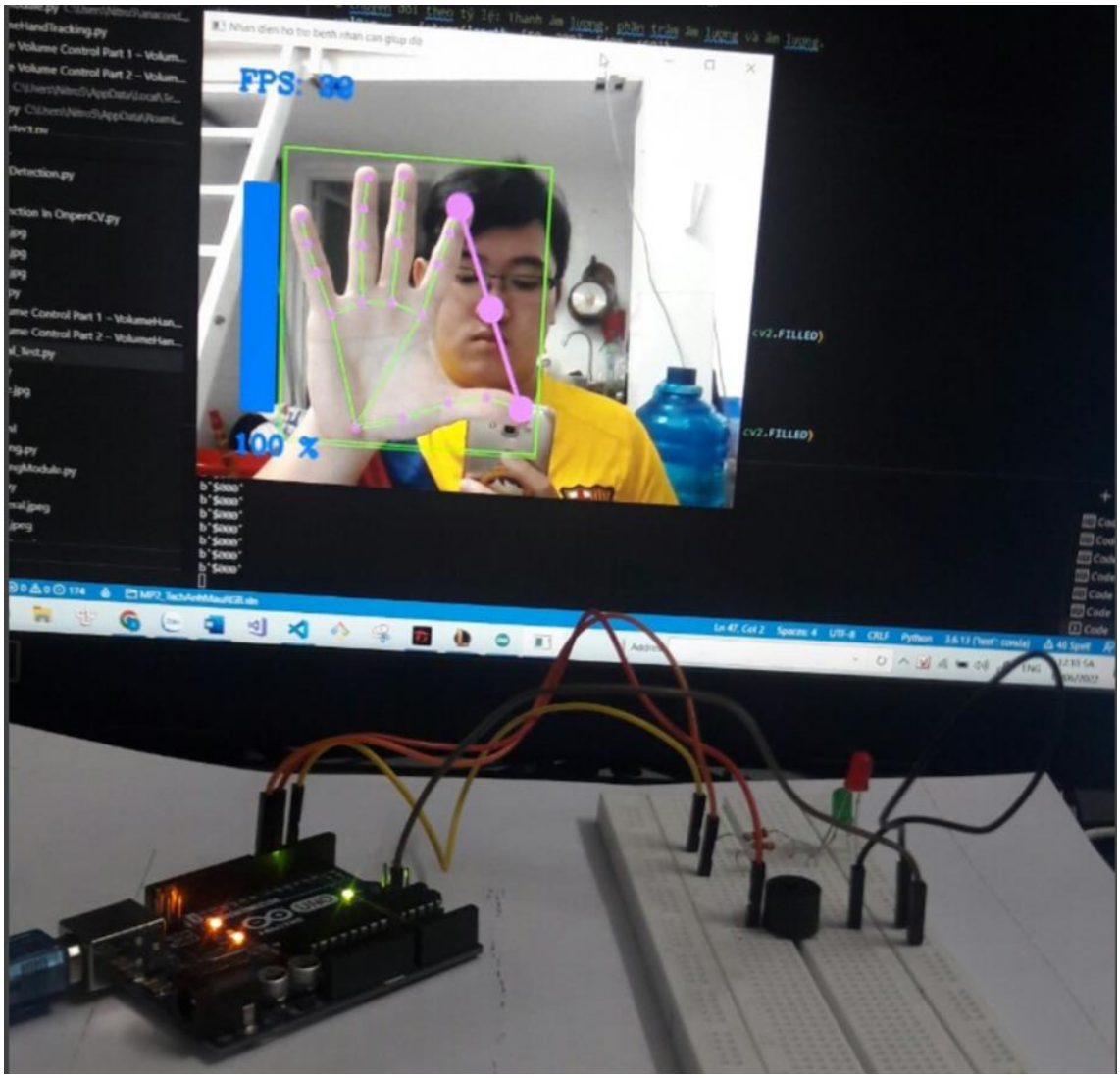
Hình ảnh giao diện chương trình

Hình ảnh khi bỏ xuống ngón út và âm lượng mức tối đa 100%

Hình ảnh khi bỏ ngón áp út xuống và set âm lượng mức tối đa 100%



Hình ảnh khi bỏ ngón áp út xuống và set âm lượng mức 40%



Hình ảnh tắt hết các thiết bị

4.2 Nhận xét.

4.2.1 Ưu điểm.

+ Tốc độ xử lý nhanh và chống nhiễu có thể giúp người điều khiển nhanh chóng và chính xác.

+ Giúp người sử dụng không cần phải chạm tay đặc biệt ở những nơi công cộng.

+ Giá thành hợp lý và ít hư hỏng.

+ Dễ dàng thay đổi linh kiện cần điều khiển.

4.2.2 Nhược điểm.

+ Chưa tối ưu nếu điều khiển những hệ thống hay linh kiện phức tạp.

+ Cần sử dụng Camera có độ phân giải cao để thu được kết quả tốt nhất.

+ Giao diện chưa bắt mắt và thân thiện với người dùng.

4.3 Hướng phát triển.

+ Có thể thay Arduino thành module điều khiển không dây khác để có thể điều khiển từ xa.

+ Có thể mở rộng nhiều chức năng điều khiển khác để sử dụng được ở nhiều nơi.

+ Có thể nghiên cứu sử dụng những giải thuật khác để chương trình chạy tối ưu nhất.

PHẦN III: KẾT LUẬN

Hệ thống nhận diện và điều khiển sử dụng công nghệ không chạm sẽ là một công cụ hữu ích đã và đang được các nước và các công ty công nghệ hàng đầu trên thế giới quan tâm. Hiện nay, hầu hết sự tương tác giữa con người với công nghệ thực tế ảo đều thông qua các thiết bị hỗ trợ như găng tay cảm biến,…. nên đây hứa hẹn sẽ là một bước đột phá mới thay thế những công nghệ cũ. Hệ thống có thể ứng dụng đa năng được cho nhiều lĩnh vực từ y tế đến công nghiệp hay thậm chí là sinh hoạt hằng ngày (đây là điều quan trọng mà một nhóm hay một tổ chức nghiên cứu nào cũng hướng đến). Trong bài báo cáo này nhóm chúng em chỉ đưa ra một cách tổng quan, dễ hiểu về đề tài, trong tương lai nhóm em muốn phát triển đề tài xa hơn, hoàn thiện hơn, ứng dụng đa lĩnh vực để có thể giúp ích được cho con người và xã hội.