HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA KỸ THUẬT ĐIỆN TỬ I

BÀI TẬP LỚN
MÔN HỌC: ĐỒ ÁN THIẾT KẾ HỆ THỐNG NHÚNG

ĐỀ TÀI: "Hệ thống tự động mở cửa, bật đèn bằng nhận diện khuôn mặt có điều khiển qua web"

Giảng viên HD:	Nguyễn Ngọc Minh
Nhóm:	01
Nhóm báo cáo:	07
Thành viên:	Phan Thị Thanh Thúy - B19DCDT240
	Lương Anh Đức - B19DCDT059
	Tô Đức Thắng - B19DCDT234
	Nguyễn Xuân Dương - B19DCDT040

Hà Nội, 5/2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin được gửi lời cảm ơn đến Ban lãnh đạo Học viện Công nghệ Bưu chính Viễn thông đã tạo môi trường rèn luyện tốt để chúng em có thể học tập và tiếp thu được những kiến thức quý báu trong những năm qua.

Chúng em xin được gửi lời cảm ơn đến thầy Nguyễn Ngọc Minh là người đã trực tiếp hướng dẫn chúng em trong quá trình thực hiện đồ án này. Chúng em xin chân thành cảm ơn thầy đã tận tình hướng dẫn, tạo điều kiện giúp đỡ, cho chúng em nhiều lời khuyên bổ ích trong suốt quá trình làm đồ án lần này.

Trong quá trình nghiên cứu và thực hiện báo cáo chúng em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy và bạn bè. Mặc dù đã cố gắng hết sức song không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và chỉ bảo tận tình của thầy và các bạn để chúng em có thể hoàn thành tốt hơn.

Cuối cùng chúng em xin kính chúc thầy và các bạn dồi dào sức khỏe, thành công trong sự nghiệp.

MỤC LỤC

Trang

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

HTML	HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
CSS	Cascading Style Sheet	Biểu định kiểu xếp tầng
MQTT	Message Queuing Telemetry Transport	Truyền tải từ xa xếp hàng tin nhắn
GPIO	General Purpose Input/Output	Đầu vào / Đầu ra mục đích chung
IoT	Internet of Things	Internet vạn vật
OpenCV	Open Source Computer Vision	Thị giác máy tính code nguồn mở
IDE	Integrated Development Environment	Môi trường phát triển tích hợp

DANH MỤC HÌNH VẼ

MỞ ĐẦU

Hiện nay, cùng với sự phát triển của xã hội, vấn đề an ninh bảo mật đang được yêu cầu khắt khe tại mọi quốc gia trên thế giới. Các hệ thống xác định, nhận dạng con người được ra đời với độ tin cậy cao. Một trong những bài toán nhận dạng con người được quan tâm nhất hiện nay đó là nhận dạng qua khuôn mặt. Vì nhận dạng qua khuôn mặt là cách mà con người sử dụng để phân biệt nhau. Bên cạnh đó, ngày nay việc thu thập, xử lý thông tin qua ảnh để nhận biết đối tượng đang được quan tâm và ứng dụng rộng rãi. Với phương pháp nay, chúng ta có thể thu nhận được nhiều thông tin từ đối tượng mà lại không cần tác động nhiều đến đối tượng nghiên cứu. Với sự phát triển của khoa học máy tính, bài toán nhận dạng mặt người từ ảnh số đang có được môi trường phát triển hết sức thuận lợi.

Dựa trên các lý thuyết về tách mặt người và nhận dạng một người, nhóm 07 chúng em đã tìm hiểu và xây dựng đồ án **"Hệ thống tự động mở cửa, bật đèn bằng nhận diện khuôn mặt có điều khiển qua web"** là một hệ thống thu thập và nhận dạng khuôn mặt một người từ ảnh số để áp dụng vào việc mở cửa tự động. Qua quá trình thực hiện, chúng em đã đạt được một số kết quả nhất định.

CHƯƠNG I: KHÁI QUÁT VỀ ĐỀ TÀI

Dựa vào ý tưởng của nhóm, chúng em thực hiện chương khái quát đề tài để phân tích bài toán, các yêu tố yêu cầu khi nhận diện khuôn mặt và các công cụ sử dụng trong bài toán cũng như hướng giải quyết mà chúng em đề ra.

1.1. Phân tích bài toán

Bài toán Nhận Diện Khuôn Mặt(Face Recognition) bao gồm các bài toán khác nhau như: Phát hiện khuôn mặt(Face detection), đánh dấu(Facial landmarking), trích chọn(rút) đặc trưng(Feature extration), gán nhãn, phân lớp(classification).

Như đã trình bày, hiện nay nhiều phương pháp đã được đưa ra phục vụ cho nhận dạng khuôn mặt và đã đạt được độ tin cậy nhất định. Nhưng có nhiều yếu tố ảnh hưởng tới việc nhận dạng của các phương pháp.

Các yếu tố ảnh hưởng tới kết quả nhận dạng (ảnh hưởng của điều kiện biên):

- *Ánh sáng:* Ảnh số biểu diễn cường độ sáng của đối tượng, do đó khi ánh sáng thay đổi, thông tin về đối tượng sẽ bị ảnh hưởng.
- *Cự ly của đối tượng so với camera:* Khoảng cách đối tượng so với camera sẽ xác định số pixel ảnh quy định nên khuôn mặt.
- *Cảm xúc, biểu cảm trên khuôn mặt:* Các nét biểu cảm cảm xúc trên khuôn mặt gây ra nhiễu, việc loại nhiễu này vẫn chưa có phương pháp hiệu quả.
- *Tư thế đứng của đối tượng (nghiêng, xoay...):* Tư thế của đối tượng sẽ xác định thông tin của đối tượng đó. Việc tư thế thay đổi quá lớn sẽ làm thay phần lớn thông tin về đối tượng, dẫn đến kết - quả nhận dạng sai.
- *Trang phục của đối tượng*: Kết quả nhận dạng có thể bị ảnh hưởng lớn nếu như đối tượng có các trang phục khác biệt so với mẫu như đeo kính, đội mũ...

Để giải quyết được bài toán này, đồ án đưa ra các biện pháp để giảm thiểu các khả năng gây sai số đã nêu trên như yêu cầu ánh sáng ổn định, tư thế của đối tượng là tương đối thẳng, cự ly từ đối tượng đến camera được quy định cụ thể....

Yêu cầu của ảnh đầu vào:

Ảnh được lấy từ Camera kỹ thuật số, có chứa hoặc không chứa đối tượng, được truyền về PC để lưu trữ, xử lý và phân tích.

Yêu cầu đối với ảnh:

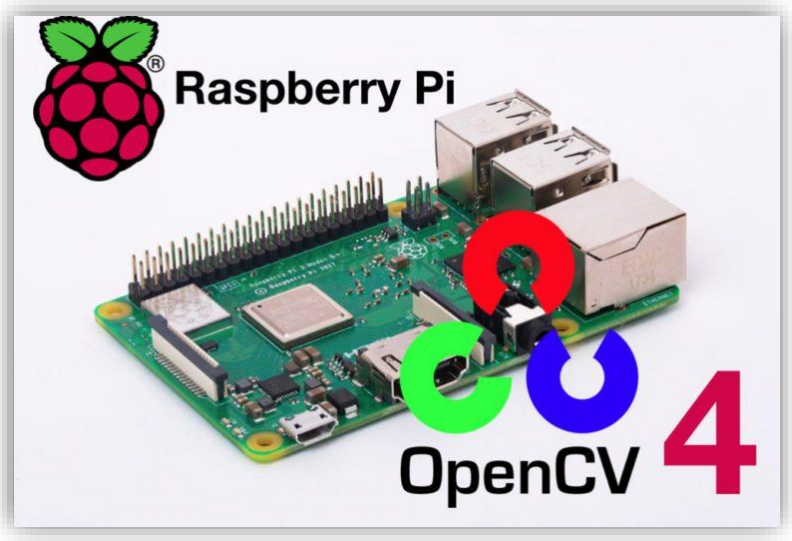
- Nền không đổi, ánh sáng tương đối ổn định, camera được đặt cố định.
- Người cần nhận dạng đứng cách camera khoảng 1m, tư thế thẳng và ngay ngắn.
- Người cần nhận dạng không nên để tóc phủ mắt, đội mũ che mặt.

Yêu cầu đầu ra:

- Nếu ảnh không chứa đối tượng, hoặc là người chưa có trong cơ sở dữ liệu (tương ứng với hệ số tương thích nhỏ) thì kết quả trả lời là không nhận biết được.
- Nếu ảnh chứa đối tượng và người đó có trong cơ sở dữ liệu, hệ thống đưa ra màn hình ảnh và thông tin về người đó.

1.2. Các công cụ được sử dụng

Để giải quyết bài toán nhận dạng khuôn mặt trong đồ án này chúng em sử dụng thư viện OpenCV và ngôn ngữ Python và thực hiện trên Raspberry Pi.



Hình 1.1: OpenCV trên Raspberry Pi để nhận diện khuôn mặt

1.3. Phương án giải quyết

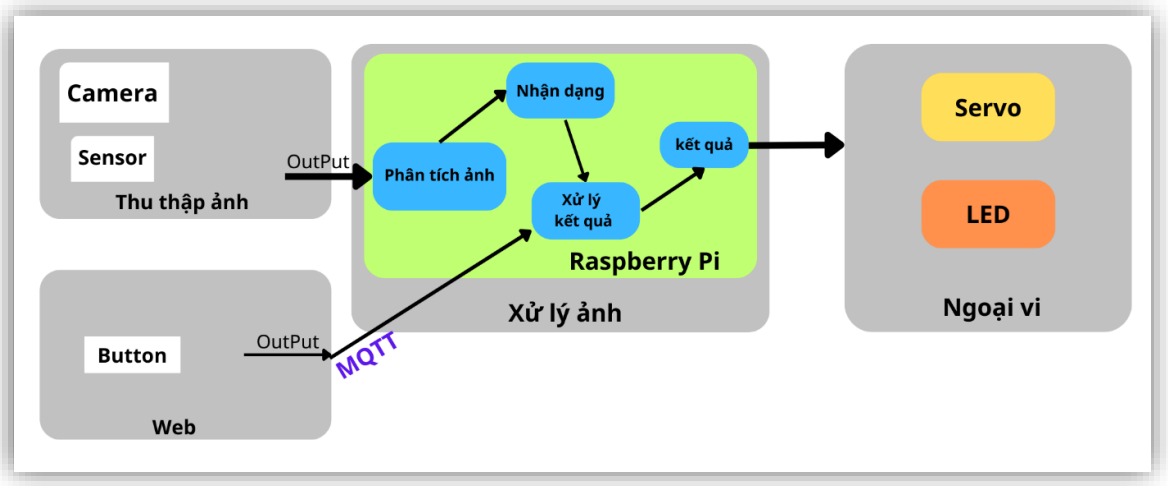
- Đầu tiên, nhìn vào hình và tìm tất cả các khuôn mặt có trong màn hình mà Camera thu được.
- Thứ 2, tập trung vào một khuôn mặt của một người và nhận diện ngay cả khi khuôn mặt quay đi hướng khác, hoặc trong môi trường thiếu ánh sáng.
- Thứ 3 là chọn ra những đặc điểm đặc trưng của khuôn mặt sử dụng để phân biệt với khuôn mặt của người khác.
- Cuối cùng, đối chiếu những đặc điểm đặc trưng đó với những người đã biết và xác định được tên người đó: nếu đúng người thì mở cửa, không đúng người thì đóng cửa.

CHƯƠNG II: PHÂN TÍCH HỆ THỐNG VÀ CƠ SỞ LÝ THUYẾT

Sau khi có những ý tưởng và phân tích bài toán, chúng em tiếp tục thực hiện phân tích hệ thống của đề tài và có những cơ sở lý thuyết về bài toán nhóm em thực hiện.

2.1. Phân tích hệ thống

2.1.1. Sơ đồ khối của hệ thống



Hình 2.1: Sơ đồ các khối chính của hệ thống

Camera thực hiện thu thập hình ảnh và đưa đến bước phân tích và nhận dạng. Raspberry Pi sẽ thực hiện xử lý hình ảnh, nếu khuôn mặt nhận được không có trong cơ sở dữ liệu thì sẽ gửi thông tin đến khối ngoại vi và cửa không mở, đèn sẽ không bật. Nếu đúng khuôn mặt có trong cơ sở dữ liệu thì gửi thông tin tới khối ngoại vi, khối ngoại vi nhận thông tin và mở cửa, bật đèn; sau khi mở cửa 5 giây thì tự động đóng cửa. Ở khối web có 2 nút nhấn, thông qua giao thức MQTT broker để nhận dữu liệu đóng/ mở cửa hay tắt mở đèn rồi truyền thông tin tới khu vực xử lý ảnh và gửi tới khối ngoại vi.

2.1.2. Chi tiết các khối

a. Khối thu thập ảnh

Trong khối thu thập ảnh, nhóm chúng em sử dụng app "IP Webcam" để chụp ảnh chủ thể cần nhận dạng thay thế cho modul camera để tiết kiệm chi phí.

b. Khối Web

Ở khối này, chúng em đã tạo một giao diện web sử dụng HTML và CSS. Khối này sẽ hiển thị lên các button để có thể điều khiển On/Off trực tiếp các thiết bị ngoại vi.

c. Khối xử lý ảnh

Khối xử lý ảnh, chúng em đã sử dụng một "Raspberry Pi 4b" với mục đích là đầu não trung tâm, xử lý các dữ liệu input được output ra từ các **khối thu thập ảnh** và **khối web.**

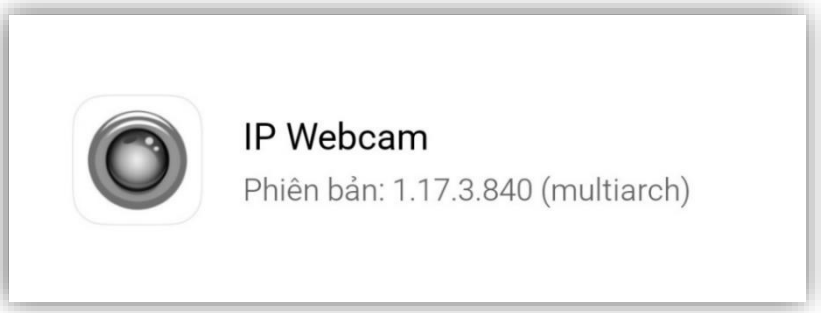
d. Khối ngoại vi

Ở khối này, bao gồm động cơ servo SG90 dùng để đóng/mở cửa, và đèn led.

2.2. Cơ sở lý thuyết

2.2.1. IP Webcam

Trên điện thoại sử dụng hệ điều hành Androi, chúng ta sẽ vào “CH Play” để tải ứng dụng “*IP Webcam*”. Tiếp theo, sau khi cài đặt xong, ta sẽ mở ứng dụng lên và chọn vào "*Start server*". Sau đó, camera sẽ được khởi động. Lúc này, ta để ý trên màn hình có dòng "IPv4", chúng ta sẽ để ý thông số này để thực hiện cho các bước tiếp theo.



Hình 2.2: Ứng dụng IP Webcam

2.2.2. Web

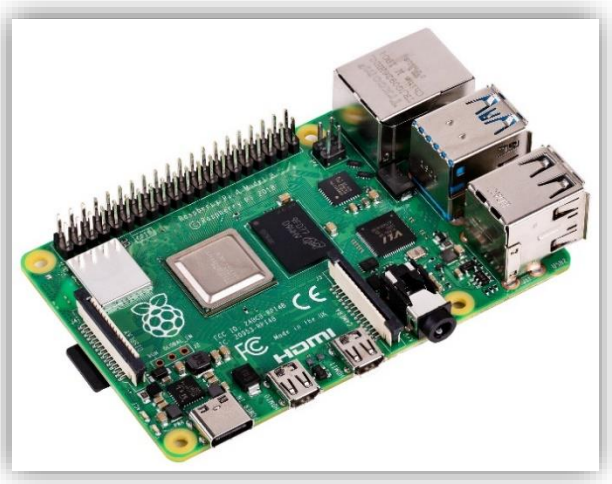
HTML (HyperText Markup Language) là ngôn ngữ đánh dấu được sử dụng để tạo nên cấu trúc và nội dung của một trang web. CSS (Cascading Style Sheets) là ngôn ngữ được sử dụng để định dạng và trang trí trang web, bao gồm màu sắc, kiểu chữ, bố cục và các hiệu ứng khác. Khi kết hợp với nhau, HTML và CSS cho phép người dùng tạo ra các trang web đẹp mắt và hấp dẫn.

Hình 2.3: Ngôn ngữ HTML

2.2.3. Raspberry Pi 4B

Raspberry Pi là một máy tính nhỏ gọn, kích thước hai cạnh như bằng khoảng một cái thẻ ATM và chạy hệ điều hành Linux. Raspberry Pi được phát triển bởi Raspberry Pi Foundation - một tổ chức phi lợi nhuận 1. Người dùng có thể sử dụng Raspberry Pi như một máy vi tính bởi người ta đã tích hợp mọi thứ cần thiết trong đó.

Raspberry Pi 4 Model B là một máy tính nhỏ gọn thuộc thế hệ mới của Raspberry Pi, hỗ trợ nhiều RAM hơn và có hiệu suất CPU, GPU và I/O được cải thiện đáng kể.



Hình 2.4: Raspberry Pi 4B

Thông số kỹ thuật của Raspberry Pi 4 Model B:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Có 3 lựa chọn RAM: 2GB, 4GB hoặc 8GB LPDDR4-2400 SDRAM
- Wifi chuẩn 2.4 GHz và 5.0 GHz IEEE 802.11ac. Bluetooth 5.0, BLE
- Cổng mạng Gigabit Ethernet
- 2 cổng USB 3.0 và 2 cổng USB 2.0
- Chuẩn 40 chân GPIO, tương thích với các phiên bản trước
- Hỗ trợ 2 cổng ra màn hình chuẩn Micro HDMI với độ phân giải lên tới 4K
- Cổng MIPI DSI
- Cổng MIPI CSI
- Cổng AV 4 chân
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Khe cắm Micro-SD cho hệ điều hành và lưu trữ
- Nguồn điện DC 5V – 3A DC chuẩn USB-C
- 5V DC via GPIO header (minimum 3A*)
- Hỗ trợ Power over Ethernet (PoE) (yêu cầu có PoE HAT)

2.2.4. Động cơ servo SG90 (Góc Quay 180)

Động cơ Servo SG90 là Servo phổ biến dùng trong các mô hình điều khiển nhỏ và đơn giản như cánh tay robot. Động cơ servo SG90 180 độ có tốc độ phản ứng nhanh, các bánh răng được làm bằng nhựa nên cần lưu ý khi nâng tải nặng vì có thể làm hư bánh răng, động cơ RC Servo 9G có tích hợp sẵn Driver điều khiển động cơ bên trong nên có thể dễ dàng điều khiển góc quay bằng phương pháp điều độ rộng xung PWM.

Hình 2.5: Động cơ servo SG90

Các thông số kỹ thuật của động cơ servo:

- Điện áp hoạt động: 4.8-5VDC
- Tốc độ: 0.12 sec/ 60 deg (4.8VDC)
- Lực kéo: 1.6 Kg.cm
- Kích thước: 21x12x22mm
- Trọng lượng: 9g.

Phương pháp điều khiển PWM:

- Độ rộng xung 0.5ms ~ 2.5ms tương ứng 0-180 độ
- Tần số 50Hz, chu kỳ 20ms

Sơ đồ dây:

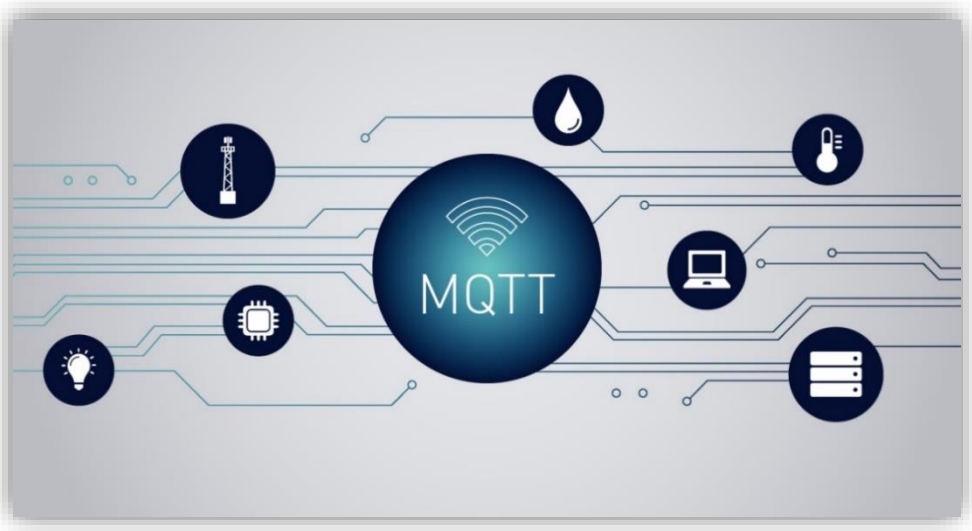
- Đỏ: Dương nguồn
- Nâu: Âm nguồn
- Cam: Tín hiệu

2.2.5. Đèn led

Hình 2.6: Bóng led

- Điện áp 3V - 3.2V
- Cường độ sáng: 12000 - 14000 mcd

2.2.6. Giao thức MQTT



Hình 2.7: Giao thức MQTT

MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe (cung cấp / thuê bao), được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Nó dựa trên một Broker (tạm dịch là "Máy chủ môi giới") "nhẹ" (khá ít xử lý) và được thiết kế có tính mở (tức là không đặc trưng cho ứng dụng cụ thể nào), đơn giản và dễ cài đặt.

2.2.6. Socket IO

Hình 2.8: Socket IO

Khi truy cập vào 1 trang web hoặc ứng dụng bất kỳ thì việc giao tiếp giữa máy chủ (Server) và máy khách (Client) là việc rất quan trọng. Để máy chủ và máy khách có thể nhận biết được sự thay đổi của đối phương thì cần sử dụng những cách thức như AJAX, long-polling, short-polling, & HTML5 server-sent events. Việc sử dụng cách giao tiếp bằng những công cụ kể trên tồn tại nhiều nhược điểm trong đó có thể kể đến là kết quả trả về chậm và tốn rất nhiều tài nguyên.

Để khắc phục những nhược điểm này, công cụ socket.io ra đời để giúp cho việc giao tiếp giữa Server và Client diễn ra tức khắc và chiếm ít tài nguyên nhất.

Socket IO là 1 module trong Node.js được nhà sáng chế tạo ra và phát triển từ năm 2010. Mục đích lớn nhất của Socket io là để tạo môi trường giao tiếp thuận lợi trên Internet giúp trả về các giá trị thực ngay tại thời điểm giao tiếp giữa các bên với nhau (thường là giữa server và client).

Việc giao tiếp 2 chiều giữa máy khách và máy chủ được thực hiện bởi socket io khi và chỉ khi máy khách có module này trong trình duyệt và máy chủ cũng đã tích hợp sẵn gói socket io. Các ứng dụng sử dụng socket io thường đòi hỏi tốc độ phản hồi ngay lập tức. Một số ví dụ điển hình như xổ số, trực tiếp bóng đá, chat…

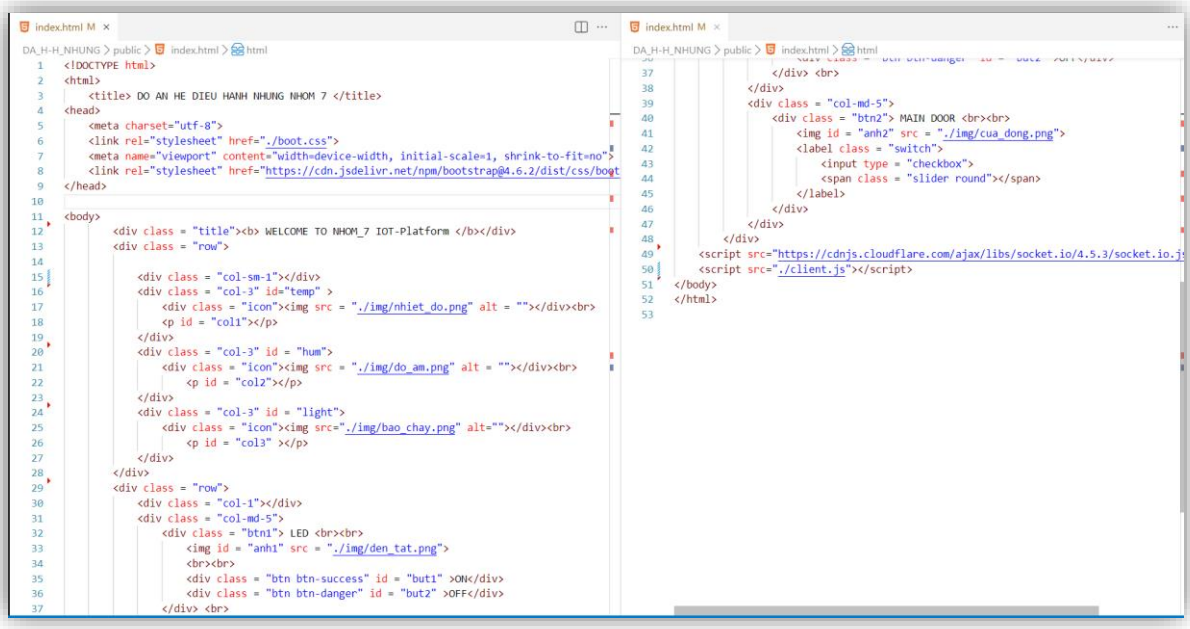
Socket IO không phải là 1 ngôn ngữ, vì vậy nó được sử dụng kết hợp với những ngôn ngữ khác như PHP, asp.net, NodeJS.

CHƯƠNG III: TRIỂN KHAI HỆ THỐNG VÀ KẾT QUẢ

Dựa trên các cơ sở lý thuyết và phương hướng giải quyết bài toán mà chúng em đưa ra, ở chương III này, chúng em thực hiện các bước theo đã đề ra.

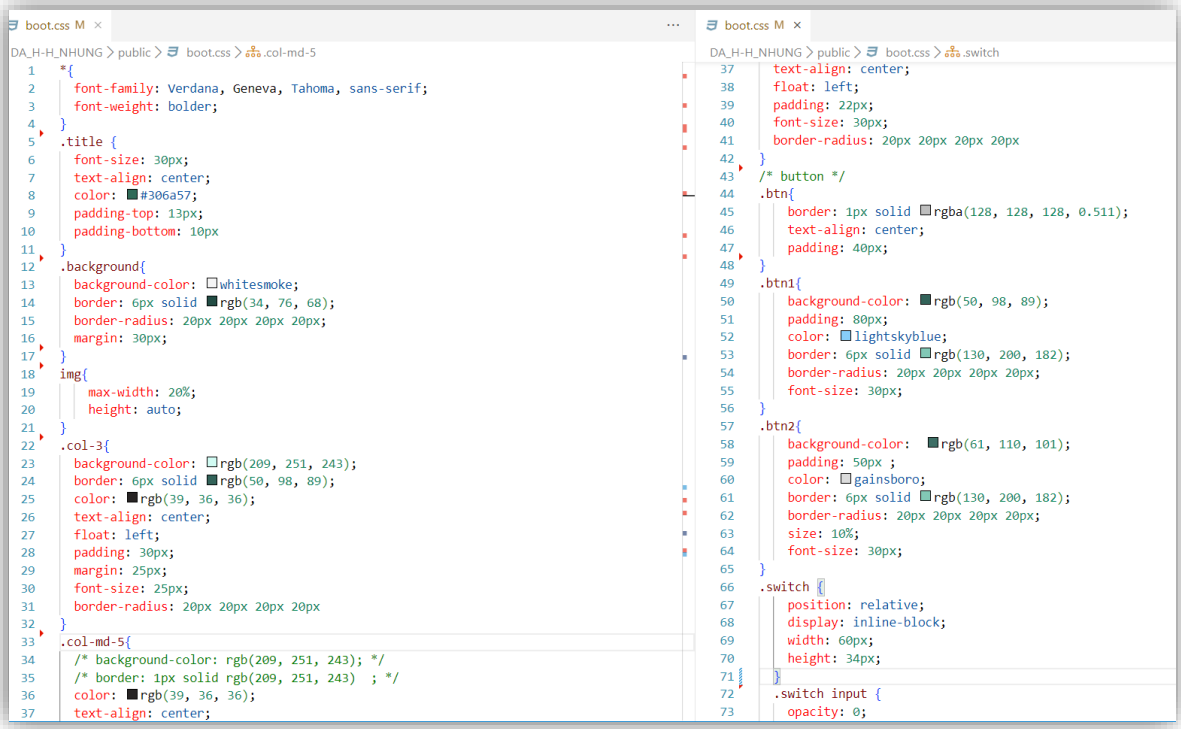
3.1. Giao diện web

Tạo giao diện web sử dụng HTML bằng file ***index.html***:



Hình 3.1: Code HTML

Dùng CSS để trang trí, tạo kiểu… trong ***boot.css***:



Hình 3.2: Code CSS

3.2. Server kết nối Raspberry Pi 4B và web

Dùng JS để tạo tương tác với web với file ***client.js***:

client.js M

```
let lightOnNoti='Bạn có muốn bật đèn không?'
let lightOffNoti='Bạn có muốn tắt đèn không?'
let DoorOpenNoti='Bạn có muốn mở cửa không?'
let DoorCloseNoti='Bạn có muốn đóng cửa không?'
document.querySelector('#but1').addEventListener('click',()=> {
    if (confirm(lightOnNoti)){
        socket.emit('led', 'on')
        // document.getElementById("but1").style.background="blue"
        // document.getElementById("but2").style.background="gray"
        document.querySelector('#anh1').src='./img/den_bat.png'
        // document.getElementById('nutall').style.background= 'red'
    }
})
document.querySelector('#but2').addEventListener('click',()=> {
    if (confirm(lightOffNoti)){
        socket.emit('led', 'off')
        // document.getElementById("but2").style.background="blue"
        // document.getElementById("but1").style.background="gray"
        document.querySelector('#anh1').src='./img/den_tat.png'
    }
})
document.querySelector('input').onclick = function(e){
    if (this.checked){
        if (confirm(DoorOpenNoti)){
            socket.emit('door', 'on')
            document.querySelector('#anh2').src='./img/cua_mo.png'
            document.querySelector('input').checked= true
        }
        else{
            document.querySelector('input').checked= false
            document.querySelector('#anh2').src='./img/cua_dong.png'
        }
    }
    else{
        if (confirm(DoorCloseNoti)){
            socket.emit('door', 'off')
            document.querySelector('#anh2').src='./img/cua_dong.png'
            document.querySelector('input').checked= false
        }
        else{
            document.querySelector('input').checked= true
            document.querySelector('#anh2').src='./img/cua_mo.png'
        }
    }
}
//------Socket IO ----
const socket = io();
socket.on('updateSensor', msg => {     //lang nghe du lieu tu mqtt
    console.log(msg);
    handlingData(msg);
});
socket.on('led', msg => {
    if (msg === 'on') {
        document.querySelector('#anh1').src = './img/den_bat.png';
    }
    if (msg === 'off') {
        document.querySelector('#anh1').src = './img/den_tat.png';
    }
    console.log(`led ${msg}`);
});
socket.on('door', msg => {
    if (msg === 'on') {
        document.querySelector('#anh2').src = './img/cua_mo.png';
    }
    if (msg === 'off') {
        document.querySelector('#anh2').src = './img/cua_dong.png';
    }
    console.log(`door ${msg}`);
});
```

Hình 3.3: File client.js tạo tương tác web

Kết nối server bằng JS dùng file ***server.js***:

server.js M

```
const mqtt = require('mqtt')
const express = require('express')
const app = express()
const http = require('http')
const server = http.createServer(app)
const { Server } = require('socket.io')
const io = new Server(server)
app.use(express.static('public'));

const options = {
    port: 8884,
    host: 'broker.mqttdashboard.com',
    clientId: 'clientId-gAe5UTm3Ou',
    username: 'duc',
    password: '123456',
}
const client = mqtt.connect(options);
client.on('connect', () => {
    console.log('MQTT connected!!');
});
const sensors = 'sensorData'
const led = 'LED'
const door = 'DOOR'
client.subscribe(sensors, () => {
    client.on('message', (topic, message, packet) => {
        console.log(message.toString());
        io.sockets.emit('updateSensor', message.toString().split(' '))
    });
});
```

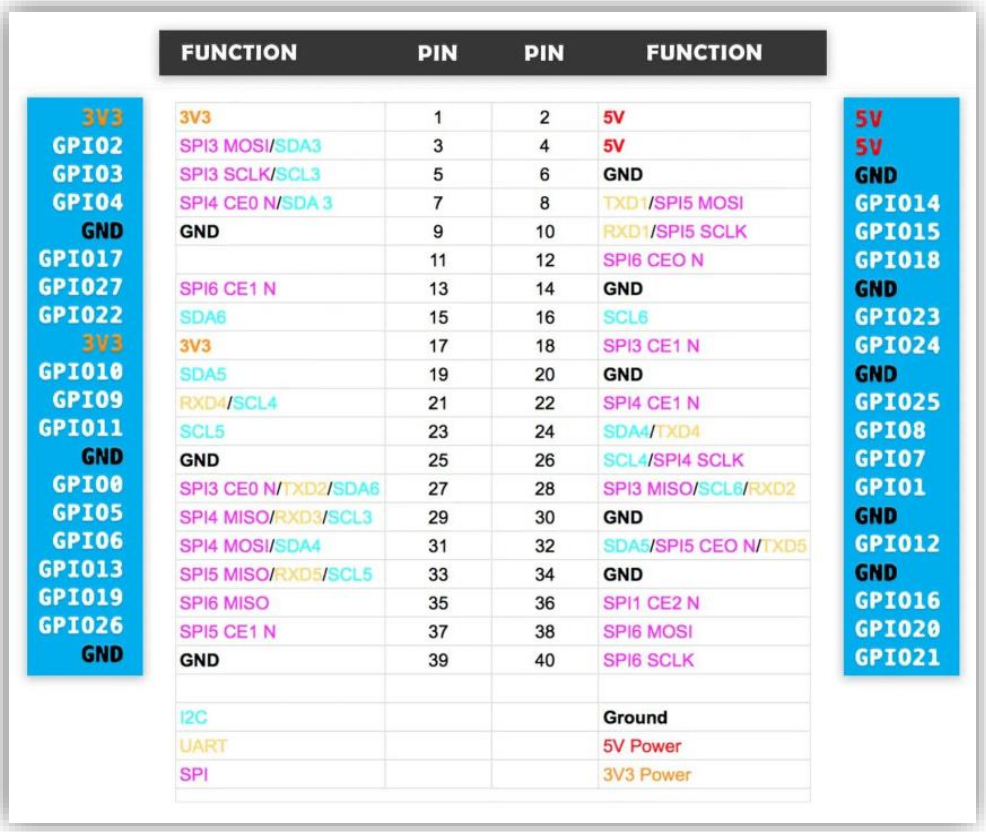
server.js M

```
io.on('connection', socket => {
    console.log(`user ${socket.id} connected`)
});
io.on('connection', socket => {
    socket.on('led', msg => {
        io.sockets.emit('led', msg);
        msg === 'on' && client.publish(led, msg)
        msg === 'off' && client.publish(led, msg)
    });
    socket.on('door', msg => {
        io.sockets.emit('door', msg);
        msg === 'on' && client.publish(cua, msg)
        msg === 'off' && client.publish(cua, msg)
    })
})

server.listen(0007, () => {
    console.log('listening on *:0007')
});
```

Hình 3.4: Code kết nối server

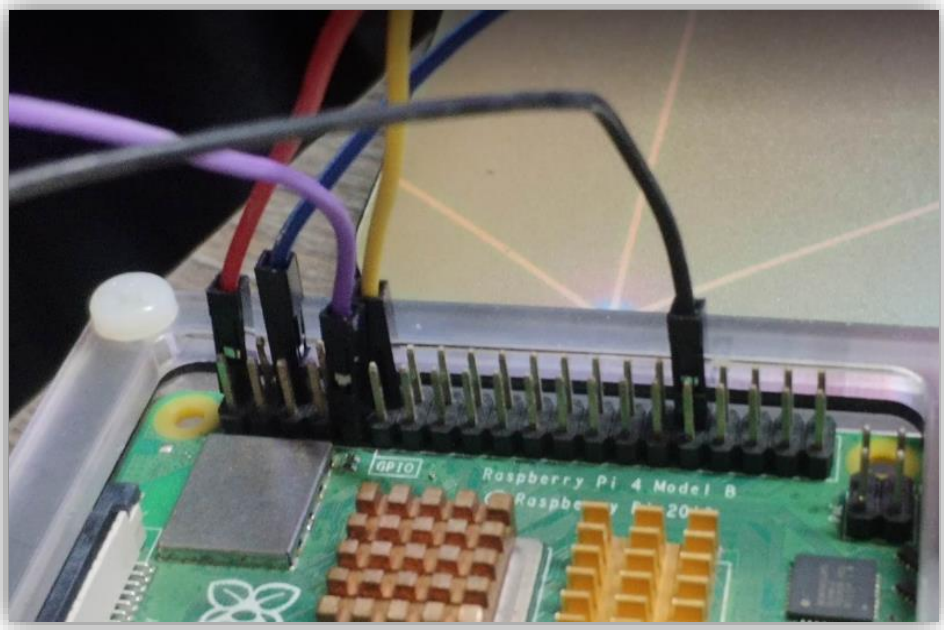
3.3. Kết nối thiết bị ngoại vi với Raspberry Pi 4B

3.3.1. Sơ đồ các chân kết nối của Pi



	FUNCTION	PIN	PIN	FUNCTION	
3V3	3V3	1	2	5V	5V
GPIO2	SPI3 MOSI/SDA3	3	4	5V	5V
GPIO3	SPI3 SCLK/SCL3	5	6	GND	GND
GPIO4	SPI4 CE0 N/SDA 3	7	8	TXD1/SPI5 MOSI	GPIO14
GND	GND	9	10	RXD1/SPI5 SCLK	GPIO15
GPIO17		11	12	SPI6 CEO N	GPIO18
GPIO27	SPI6 CE1 N	13	14	GND	GND
GPIO22	SDA6	15	16	SCL6	GPIO23
3V3	3V3	17	18	SPI3 CE1 N	GPIO24
GPIO10	SDA5	19	20	GND	GND
GPIO9	RXD4/SCL4	21	22	SPI4 CE1 N	GPIO25
GPIO11	SCL5	23	24	SDA4/TXD4	GPIO8
GND	GND	25	26	SCL4/SPI4 SCLK	GPIO7
GPIO0	SPI3 CE0 N/TXD2/SDA6	27	28	SPI3 MISO/SCL6/RXD2	GPIO1
GPIO5	SPI4 MISO/RXD3/SCL3	29	30	GND	GND
GPIO6	SPI4 MOSI/SDA4	31	32	SDA5/SPI5 CEO N/TXD5	GPIO12
GPIO13	SPI5 MISO/RXD5/SCL5	33	34	GND	GND
GPIO19	SPI6 MISO	35	36	SPI1 CE2 N	GPIO16
GPIO26	SPI5 CE1 N	37	38	SPI6 MOSI	GPIO20
GND	GND	39	40	SPI6 SCLK	GPIO21
	I2C			Ground	
	UART			5V Power	
	SPI			3V3 Power	

Hình 3.5: Sơ đồ các chân kết nối của Pi



Hình 3.6: Hình ảnh thực tế sau khi nối ngoại vi với Rasp Pi

3.3.2. Kết nối động cơ servo với Pi

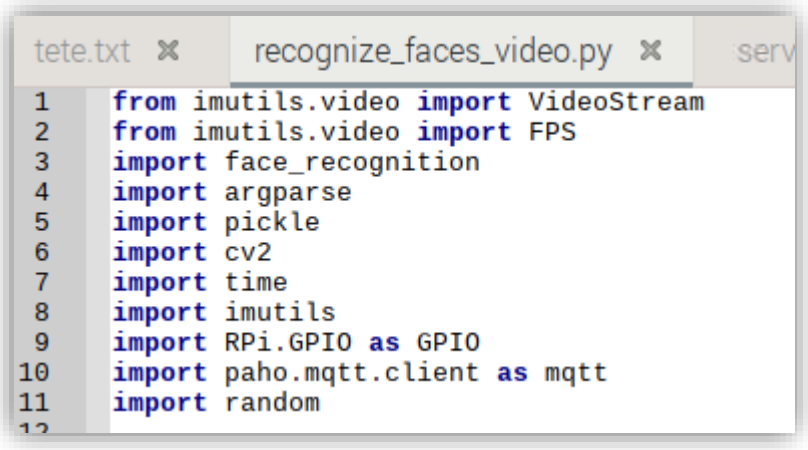
- Chân PWM của servo sẽ cắm vào Pin12 (GPIO18 của Rasp Pi).
- Chân VCC của servo sẽ cắm vào Pin2 (5V của Rasp Pi).
- Chân GND của servo sẽ cắm vào Pin6 (GND của Rasp Pi).

3.3.3. Kết nối bóng Led với Rasp Pi

- Cực dương của đèn sẽ mắc nối tiếp với một con trở kháng và sau đó được cắm vào chân Pin32 (GPIO12 của Rasp Pi).
- Ta sẽ cấp thêm nguồn 3V3 vào cực dương của đèn.
- Cực âm của đèn sẽ cắm vào chân Pin9 (GND của Rasp Pi).

3.4. Lập trình trên Raspberry Pi

3.4.1. Các IDE sử dụng trên Rasp Pi



Hình 3.7: Các thư viện sử dụng trong code lập trình Rasp Pi

IDE (Integrated Development Environment) là môi trường tích hợp dùng để viết code để phát triển ứng dụng. Ngoài ra IDE tích hợp các tool hỗ trợ khác như trình biên dịch (Compiler), trình thông dịch (Interpreter), kiểm tra lỗi (Debugger), định dạng hoặc highlight code, tổ chức thư mục code, tìm kiếm code…Các môi trường IDE thường bao gồm:

- Một trình soạn thảo code nguồn (source code editor): dùng để viết code.
- Trình biên dịch (compiler) và/hoặc trình thông dịch (interpreter).
- Công cụ xây dựng tự động: khi sử dụng sẽ biên dịch (hoặc thông dịch) code nguồn, thực hiện liên kết (linking), và có thể chạy chương trình một cách tự động.
- Trình gỡ lỗi (debugger): hỗ trợ dò tìm lỗi.
- Ngoài ra, còn có thể bao gồm hệ thống quản lý phiên bản và các công cụ nhằm đơn giản hóa công việc xây dựng giao diện người dùng đồ họa (GUI).
- Nhiều môi trường phát triển hợp nhất hiện đại còn tích hợp trình duyệt lớp (class browser), trình quản lý đối tượng (object inspector), lược đồ phân cấp lớp (class hierarchy diagram) … để sử dụng trong việc phát triển phần mềm theo hướng đối tượng.

Các IDE được sử dụng trong bài toán:

- *RPi.GPIO*: Là một thư viện Python cho phép điều khiển các chân GPIO (General Purpose Input/Output) trên Raspberry Pi. Người dùng có thể sử dụng thư viện này để giao tiếp với các thiết bị ngoại vi và các mô-đun được kết nối với Raspberry Pi thông qua các chân GPIO.
- *paho-mqtt*: Là một thư viện Python cho phép sử dụng giao thức MQTT (Message Queuing Telemetry Transport) để gửi và nhận dữ liệu giữa các

thiết bị thông minh. MQTT là một giao thức nhắn tin nhẹ được sử dụng rộng rãi trong các ứng dụng IoT (Internet of Things).

- *random*: Là một thư viện Python cung cấp các hàm để tạo ra các số ngẫu nhiên.
- *imutils.video*: Là một module của thư viện imutils cung cấp các công cụ để làm việc với video trong Python.
- *face_recognition*: Là một thư viện Python cho phép nhận diện khuôn mặt trong hình ảnh và video. Thư viện này sử dụng các thuật toán học máy để phát hiện và nhận diện khuôn mặt một cách chính xác.
- *argparse*: Là một thư viện Python cung cấp các công cụ để phân tích các đối số dòng lệnh.
- *pickle*: Là một thư viện Python cho phép lưu trữ và tải lại các đối tượng Python.
- *cv2*: Là module của thư viện OpenCV (Open Source Computer Vision) cho Python, cung cấp các công cụ để xử lý hình ảnh và video.
- *time*: Là một thư viện Python cung cấp các hàm để làm việc với thời gian.
- *imutils*: Là một thư viện Python cung cấp các công cụ tiện ích để làm việc với hình ảnh và video trong Python.
- *keyboard*: Là một thư viện Python cho phép kiểm soát và giám sát bàn phím.

3.4.2. Code thực hiện

Code bắt đầu bằng cách định nghĩa hai biến *servo_pin* và *led_pin* đại diện cho các chân GPIO sẽ được sử dụng để điều khiển động cơ servo và LED tương ứng. Sau đó là thiết lập các chân GPIO bằng cách gọi *GPIO.setmode(GPIO.BCM)* để đặt chế độ đánh số chân thành BCM (kênh SOC Broadcom) và sau đó gọi *GPIO.setup(servo_pin, GPIO.OUT)* và *GPIO.setup(led_pin, GPIO.OUT)* để đặt chân servo và LED làm chân đầu ra.

Sau khi thiết lập các chân GPIO, tạo một đối tượng PWM cho chân servo bằng cách gọi *GPIO.PWM(servo_pin, 50)* với tần số 50Hz. Đối tượng PWM sau đó được khởi động với chu kỳ xung nhịp là 0 bằng cách gọi *pwm.start(0)*.

```
servo_pin = 18  #Chân GPIO điều khiển servo
led_pin = 12  #Chân GPIO để điều khiển đèn LED

GPIO.setmode(GPIO.BCM)
GPIO.setup(servo_pin, GPIO.OUT)
GPIO.setup(led_pin, GPIO.OUT)

pwm = GPIO.PWM(servo_pin, 50)  #Tạo tín hiệu PWM với tần số 50Hz(chu kỳ 20ms)
pwm.start(0)  #Bắt đầu PWM với duty cycle 0(motor dừng)
```

Tiếp đó là khai báo các biến kết nối với MQTT. Biến *broker* được đặt thành địa chỉ của MQTT, biến *port = 1883* là số cổng để sử dụng để kết nối với MQTT và biến *client_id* là ID người dùng.

Tiếp theo, định nghĩa một hàm *connect_mqtt()* được sử dụng để kết nối với trình môi giới MQTT. Hàm này tạo một đối tượng người dùng MQTT bằng cách gọi *mqtt.Client(client_id)* và thiết lập hàm gọi lại *on_connect* của nó thành một hàm in ra thông báo cho biết kết nối có thành công hay không. Hàm *client.connect(broker, port)* để kết nối với trình môi giới.

```
broker = 'broker.emqx.io'
port = 1883
client_id = f'python-mqtt-{random.randint(0, 1000)}'

def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)
    client = mqtt.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

Sau khi định nghĩa hàm *connect_mqtt()*, ta định nghĩa một hàm khác *subscribe(client)* được sử dụng để theo dõi các chủ đề trên trình môi giới MQTT. Hàm này nhận một đối tượng người dùng MQTT làm tham số và thiết lập hàm gọi lại *on_message* thành một hàm xử lý các message nhận được. Hàm *client.subscribe("light")* và *client.subscribe("servo")* để theo dõi các topic "light" và "servo".

```
def subscribe(client):
    def on_message(client, userdata, msg):
        print(msg.topic + " " + str(msg.payload))
        if msg.topic == "light":
            if msg.payload == b'on':
                GPIO.output(led_pin, GPIO.HIGH)
            elif msg.payload == b'off':
                GPIO.output(led_pin, GPIO.LOW)
        if msg.topic == "servo":
            position = int(msg.payload)
            if position >= 0 and position <= 180:
                set_angle(position)
    client.subscribe("light")
    client.subscribe("servo")
    client.on_message = on_message
```

Hàm *on_message* kiểm tra xem message nhận được có phải là topic "light" hay không và liệu nó là "on" hay "off". Nếu là "on" thì bật LED bằng cách gọi *GPIO.output(led_pin, GPIO.HIGH)*. Nếu là "off" thì tắt LED bằng cách gọi *GPIO.output(led_pin, GPIO.LOW)*. Nếu message nhận được là topic "servo" thì nó sẽ message của nó thành số nguyên và gọi hàm *set_angle(position)* với giá trị đó làm tham số.

```
def set_angle(angle):
    duty_cycle = angle / 18.0 + 2.5  # Chuyển đổi góc sang chu kỳ duty cycle
    pwm.ChangeDutyCycle(duty_cycle)
    time.sleep(1)  # Chờ 1 giây để servo đạt được vị trí mới

def run():
    client = connect_mqtt()
    client.loop_start()
    subscribe(client)
```

Hàm *set_angle(angle)* nhận một góc làm tham số và thiết lập góc của động cơ servo tương ứng. Hàm này tính toán giá trị chu kỳ xung nhịp dựa trên góc đã cho bằng công thức *duty_cycle = angle / 18.0 + 2.5*. Sau đó sử dụng *pwm.ChangeDutyCycle(duty_cycle)* để thiết lập chu kỳ xung nhịp của tín hiệu PWM để điều khiển động cơ servo.

Sau khi định nghĩa tất cả các hàm này, ta định nghĩa một khối chính gọi các hàm này theo thứ tự. Bắt đầu bằng cách gọi *connect_mqtt()* để kết nối với trình môi giới MQTT và sau đó gọi *client.loop_start()* để bắt đầu xử lý lưu lượng mạng trong

```
ap = argparse.ArgumentParser()
# Đường dẫn đến file encodings đã lưu
ap.add_argument("-e", "--encodings", required=True, help="path to the serialized db of facial encodings")
# Lưu video từ webcam
ap.add_argument("-o", "--output", type=str, help="path to the output video")
ap.add_argument("-y", "--display", type=int, default=1, help="whether or not to display output frame to screen")
ap.add_argument("-d", "--detection_method", type=str, default="hog", help="face detection model to use: cnn or hog")
args = vars(ap.parse_args())

if __name__ == '__main__':
    run()
```

một luồng riêng biệt. Sau đó là gọi *subscribe(client)* để theo dõi các chủ đề MQTT.

Sử dụng mô-đun *argparse* để phân tích các đối số dòng lệnh. Tạo một đối tượng *ArgumentParser* và thêm một số đối số vào nó bằng phương thức *add_argument()*. Các đối số được thêm vào là:

- "-e", "—encodings": Đối số bắt buộc chỉ định đường dẫn đến cơ sở dữ liệu được tuần tự hóa của mã hóa khuôn mặt.
- "-o", "—output": Đối số tùy chọn chỉ định đường dẫn đến tệp video đầu ra.
- "-y", "—display": Đối số tùy chọn chỉ định liệu có hiển thị khung đầu ra ra màn hình hay không. Giá trị mặc định là 1 (Đúng).
- "-d", "--detection_method": Đối số tùy chọn chỉ định mô hình phát hiện khuôn mặt để sử dụng. Có thể là "cnn" hoặc "hog" và có giá trị mặc định là "hog".

Sau khi định nghĩa các đối số, gọi *ap.parse_args()* để phân tích các đối số dòng lệnh và sau đó gọi *vars()* trên kết quả để chuyển nó thành một từ điển. Từ điển kết quả được lưu trữ trong biến args.

```
# Load các khuôn mặt đã được mã hóa
print("[INFO] loading encodings...")
with open(args["encodings"], "rb") as f:
    data = pickle.load(f)

# Khởi tạo video stream và pointer to the output video file
print("[INFO] starting video stream...")
vs = VideoStream(src='http://172.20.10.2:8080/video').start()
time.sleep(2.0)

# Khởi động bộ đếm FPS
fps = FPS().start()

if args["output"] is not None:
    fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    writer = cv2.VideoWriter(args["output"], fourcc, 20, (640, 480))
```

Đoạn code cũng bao gồm một khối chính kiểm tra xem tập lệnh có được chạy trực tiếp hay không (thay vì được nhập như một mô-đun) bằng cách kiểm tra xem *__name__* có bằng *'__main__'* hay không. Nếu có thì gọi một hàm có tên là *run()*.

Load một cơ sở dữ liệu được tuần tự hóa của mã hóa khuôn mặt bằng cách sử dụng mô-đun *pickle* và mở tệp được chỉ định bởi đối số dòng lệnh *encodings* ở chế độ nhị phân và gọi *pickle.load()* để tải dữ liệu từ tệp. Dữ liệu được tải được lưu trữ trong biến *data*.

Sau đó mở một luồng video bằng cách tạo một đối tượng *VideoStream* và sử dụng nguồn video của Cameraa IP "http://172.20.10.2:8080/video". Tiếp đó sử dụng *start()* trên đối tượng *VideoStream* để bắt đầu chụp khung hình video. Mã cũng bao gồm một lời gọi tới *time.sleep(2.0)* để chờ trong 2 giây trước khi tiếp tục.

Sau khi bắt đầu luồng video, mã bắt đầu một bộ đếm khung hình/giây (FPS) bằng cách tạo một đối tượng FPS và gọi phương thức start() của nó. Ta cũng kiểm tra xem có chỉ định tệp video đầu ra hay không bằng cách kiểm tra xem args["output"] có phải là None hay không. Nếu không phải là None, nó tạo một đối tượng *VideoWriter* bằng cách gọi *cv2.VideoWriter()* với đường dẫn tệp đầu ra được chỉ định, mã FourCC cho codec MJPG, tỷ lệ khung hình 20 khung hình/giây và kích thước khung hình 640x480 pixel.

```
# Lặp lại các khung hình từ luồng tệp video
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Phát hiện khuôn mặt trong thang độ xám
    rects = face_recognition.face_locations(gray, model=args["detection_method"])
    # OpenCV trả về tọa độ hộp giới hạn theo thứ tự (trên, phải, dưới, trái)
    boxes = [(top, right, bottom, left) for (top, right, bottom, left) in rects]
    # tính toán các nhúng khuôn mặt cho mỗi hộp giới hạn khuôn mặt
    encodings = face_recognition.face_encodings(rgb, boxes)
    names = []
```

Bắt đầu vòng lặp các khung hình từ luồng video và thay đổi kích thước chúng thành chiều rộng 500 pixel bằng hàm *imutils.resize* và chuyển đổi khung hình sang thang độ xám và RGB. Sau đó sử dụng phương pháp phát hiện khuôn mặt được chỉ định để tìm các vị trí khuôn mặt trong khung hình thang độ xám và tính toán các nhúng khuôn mặt cho mỗi hộp giới hạn khuôn mặt.

```
# Loop over the facial embeddings
for encoding in encodings:
    # Match từng khuôn mặt trong hình ảnh đầu vào với các mã hóa đã biết
    matches = face_recognition.compare_faces(data["encodings"], encoding)
    name = "Unknown"  # Tên mặc định là Unknown

    if True in matches:
        # Tìm indexs của tất cả các khuôn mặt đã mã hóa
        # Khởi tạo một từ điển enumerate để đếm tổng số lần mỗi khuôn mặt đã được mã hóa
        matched_idxs = [i for (i, b) in enumerate(matches) if b]
        counts = {}

        # Lặp lại các indexs phù hợp và duy trì số lượng cho từng khuôn mặt được nhận diện
        for idx in matched_idxs:
            name = data["names"][idx]
            counts[name] = counts.get(name, 0) + 1

        # Xác định khuôn mặt được nhận dạng thường xuyên nhất
        name = max(counts, key=counts.get)

    # Update danh sách tên
    names.append(name)

#  Loop các khuôn mặt được nhận diện
for ((top, right, bottom, left), name) in zip(boxes, names):
    # Vẽ tên khuôn mặt dự đoán
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
```

Tiếp tục vòng lặp khác sử dụng phương thức *compare-faces* của thư viện face_recognition để so sánh chúng với các khuôn mặt đã biết. Nếu có bất kỳ khuôn mặt nào được khớp thì sẽ tự động tìm các chỉ số của tất cả các khuôn mặt được khớp và khởi tạo một từ điển *enumerate* để đếm tổng số lần mỗi khuôn mặt được khớp. Sau đó là xác định khuôn mặt được nhận dạng thường xuyên nhất và cập nhật lại danh sách tên.

Tạo thêm một vòng lặp các khuôn mặt được nhận diện và sử dụng thư viện *OpenCV* để vẽ tên khuôn mặt được nhận diện lên khung hình. Đầu tiên, vẽ một hộp giới hạn xung quanh khuôn mặt bằng cách sử dụng phương thức *rectangle* của

OpenCV và tính toán vị trí y để đặt tên và sử dụng phương thức *putText* của *OpenCV* để vẽ tên lên khung hình.

```
#  Loop các khuôn mặt được nhận diện
for ((top, right, bottom, left), name) in zip(boxes, names):
    # Vẽ tên khuôn mặt dự đoán
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

# Khởi tạo trình video nếu đường dẫn tệp video đầu ra đã được cung cấp và trình ghi video chưa được khởi tạo
if args["output"] is not None and writer is None:
    writer = cv2.VideoWriter(args["output"], fourcc, 20, (frame.shape[1], frame.shape[0]), True)
    # Nếu trình ghi video không phải là None, ghi khung vào tệp video đầu ra
    writer.write(frame)

# Kiểm tra xem khung có được hiển thị trên màn hình không
if args["display"] > 0:
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # Nhấn 'q' để thoát
    if key == ord("q"):
        break

# Cập nhật bộ đếm FPS
fps.update()
```

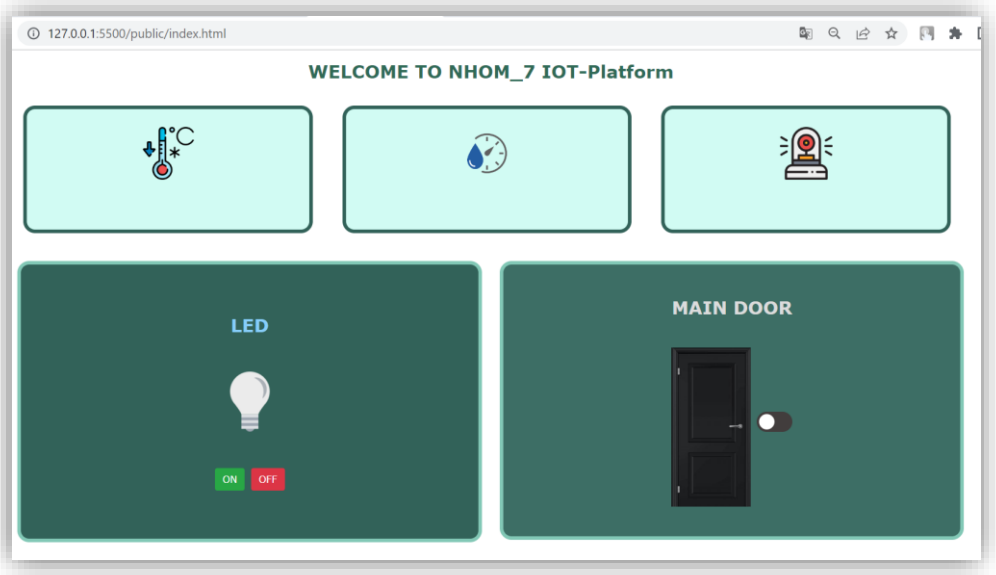
Nếu đường dẫn tệp video đầu ra đã được cung cấp và trình ghi video chưa được khởi tạo, đoạn mã sẽ khởi tạo trình ghi video bằng cách sử dụng phương thức VideoWriter của OpenCV. Nếu trình ghi video không phải là None, đoạn mã sẽ hiển thị khung vào tệp video output. Nếu khung được hiển thị trên màn hình thì sử dụng phương thức *imshow* của *OpenCV* để hiển thị khung và kiểm tra xem người dùng có nhấn phím 'q' để thoát không. Nếu có, vòng lặp sẽ kết thúc.

```
# Dừng hẹn giờ và hiển thị thông tin FPS
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

cv2.destroyAllWindows()

vs.stop()
if writer is not None:
    writer.release()
GPIO.cleanup()
```

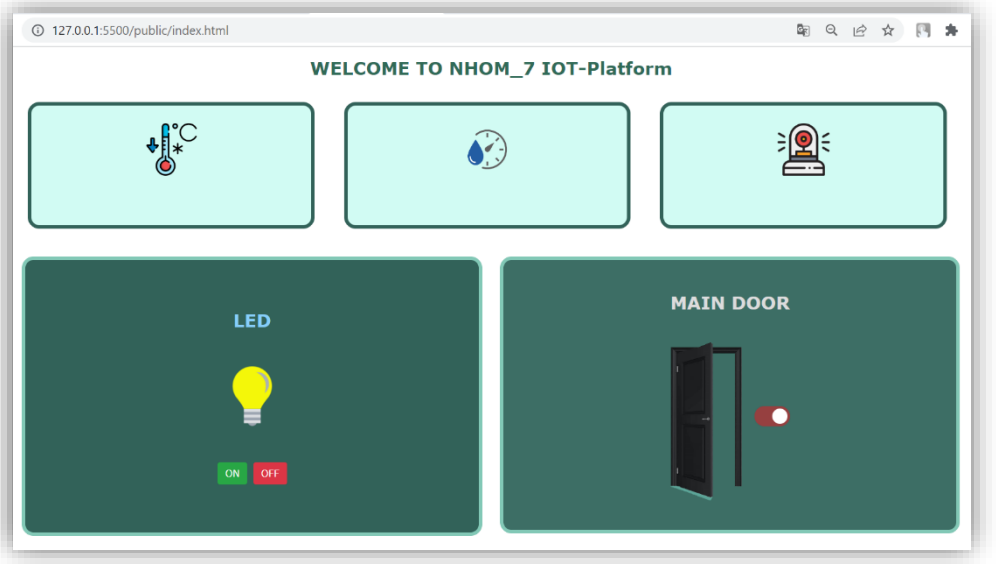
Sau khi người dùng nhấn’q’thì chương trình sẽ dừng bộ đếm FPS và hiển thị thời gian đã trôi qua và số khung hình trên giây xấp xỉ. Sau đó, đóng tất cả các cửa sổ OpenCV và dừng luồng video. Nếu trình ghi video không phải là None, đoạn mã sẽ giải phóng tài nguyên bộ nhớ.

3.5. Kết quả

3.5.1. Giao diện web



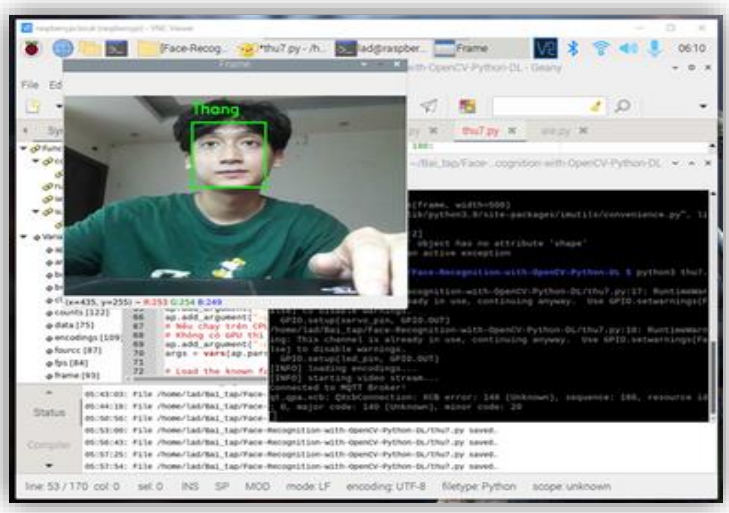
Hình 3.8: Giao diện web lúc cửa đóng, đèn tắt



Hình 3.9: Giao diện web lúc cửa mở, đèn bật

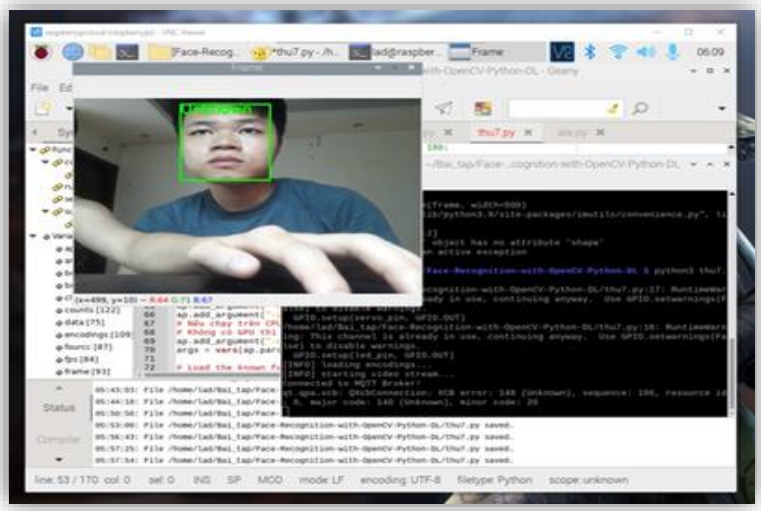
3.5.2. Nhận diện khuôn mặt mở cửa, bật đèn

Sau khi thu thập, training khuôn mặt và lưu vào cơ sở dữ liệu, chúng em đã thực hiện cho Camera IP nhận hình ảnh và xử lý thông tin, những khuôn mặt xuất hiện trong camera sẽ được gửi về khối ngoại vi. Nếu là khuôn mặt đã có trong cơ sở dữ liệu thì nó sẽ hiển thị tên lên màn hình và mở cửa, bật đèn. Nếu khuôn mặt nhận được không có trong cơ sở dữ liệu thì trên màn hình sẽ hiện Unknow là không mở cửa và cũng không bật đèn.

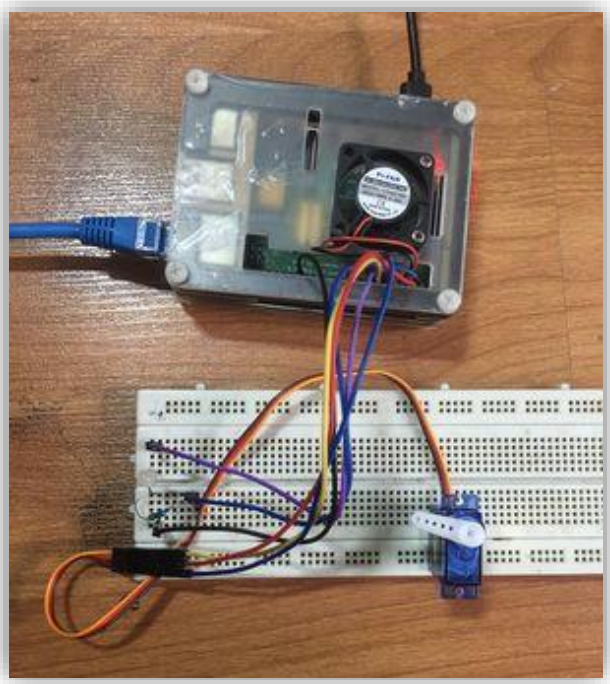


Hình 3.10: Hiển thị tên khi nhận đúng khuôn mặt

Hình 3.11: Đúng khuôn mặt thì cửa sẽ tự động mở và bật đèn



Hình 3.12: Hiển thị Unknow khi nhận không đúng khuôn mặt



Hình 3.13: Cửa đóng và không bật đèn khi không nhận đúng khuôn mặt

Tuy nhiên, do số lượng hình ảnh mã hóa chỉ từ 5-10 ảnh nên độ chính xác khi nhận diện khuôn mặt không được chính xác hoàn toàn và nhận diện chính xác khi khuôn mặt xuất hiện trong khoảng 1m. Khi xuất hiện nhiều khuôn mặt trong khung hình cùng một lúc thì khuôn mặt xuất hiện ở xa màn hình có thể bị nhận diện không đúng.

Để giải quyết vấn đề ấy thì chúng em đưa ra giải pháp là chúng ta nên mã hóa khuôn mặt qua 50-100 ảnh để được độ chính xác cao hơn.

KẾT LUẬN

Trong đồ án này, chúng em đã xây dựng một hệ thống tự động mở cửa và bật đèn bằng nhận diện khuôn mặt có điều khiển qua web. Chúng em đã sử dụng các công cụ như Raspberry Pi 4B, IP Webcam và thư viện OpenCV để thu thập và xử lý hình ảnh khuôn mặt. Kết quả cho thấy hệ thống hoạt động ổn định.

Trong quá trình thực hiện đồ án, chúng em đã gặp phải một số khó khăn và thách thức như việc đảm bảo chất lượng hình ảnh và độ chính xác của thuật toán nhận diện khuôn mặt. Tuy nhiên, chúng em đã tìm ra các giải pháp để giải quyết những vấn đề này và đạt được kết quả tốt.

Hệ thống tự động mở cửa và bật đèn bằng nhận diện khuôn mặt có điều khiển qua web là một giải pháp hiện đại và tiện lợi cho việc quản lý an ninh và tiết kiệm năng lượng. Sử dụng công nghệ nhận diện khuôn mặt để tự động mở cửa và bật đèn giúp tăng cường an ninh và tiện ích cho người dùng. Việc điều khiển qua web cho phép người dùng dễ dàng quản lý hệ thống từ xa. Tuy nhiên, để đảm bảo hoạt động ổn định và an toàn của hệ thống, cần phải đảm bảo các yếu tố như chất lượng hình ảnh, độ chính xác của thuật toán nhận diện khuôn mặt và tính bảo mật của hệ thống điều khiển qua web.

Trong tương lai, hệ thống có thể được phát triển thêm bằng cách tích hợp thêm các tính năng mới như nhận diện giọng nói, nhận diện vân tay hoặc nhận diện mống mắt để tăng cường an ninh và tiện ích cho người dùng. Ngoài ra, hệ thống cũng có thể được kết nối với các thiết bị cảm biến như cảm biến khí gas, cảm biến nhiệt độ, độ ẩm…hay các thiết bị thông minh khác trong nhà để tạo thành một hệ thống nhà thông minh hoàn chỉnh. Việc phát triển các ứng dụng di động cho phép người dùng dễ dàng quản lý và điều khiển hệ thống từ xa cũng là một hướng phát triển tiềm năng.

DANH MỤC TÀI LIỆU THAM KHẢO

Danh mục các website tham khảo

[1] *https://openai.com/product/chatgpt*

[2] *https://dientu360.com/dong-co-servo-sg90*

[3] *https://raspberrypi.vn/san-pham/raspberry-pi-4-model-b-2019*

[4] *https://blog.mecsu.vn/raspberry-pi-4*

[5] *https://bizflycloud.vn/tin-tuc/socketio*

[6] *https://viblo.asia/p/mqtt*