

ĐẠI HỌC CÔNG NGHỆ  
ĐẠI HỌC QUỐC GIA HÀ NỘI



## BÁO CÁO CUỐI KỲ

**Đề tài : Hệ thống mở khoá nhận diện khuôn mặt  
sử dụng raspberry Pi**

***Môn học : Lập trình nâng cao trong ứng dụng đo lường điều khiển***

***Giảng viên : TS. Đỗ Trần Thắng  
CN. Phạm Mạnh Tuấn***

### NHÓM 12

***Sinh viên :***

1	Nguyễn Văn Tùng	K64M – CLC2
2	Bùi Đức Duy	K64M – CLC2
3	Lê Mạnh Dũng	K64M – CLC2
4	Bùi Thị Dương Hải	K64M – CLC2

*Hà Nội, ngày 15 tháng 5 năm 2022*

## LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến trường Đại học Công Nghệ - ĐHQGHN đã đưa môn học Lập trình nâng cao trong ứng dụng đo lường điều khiển vào chương trình giảng dạy. Đặc biệt, chúng em xin cảm ơn sâu sắc thầy Đỗ Trần Thắng và Thầy Phạm Mạnh Tuấn đã dạy dỗ, truyền đạt những kiến thức quý báu cho chúng em trong suốt kỳ học vừa qua. Trong suốt thời gian qua, chúng em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập, làm việc nhóm hiệu quả, nghiêm túc. Đây chắc chắn sẽ là những kiến thức quý báu và là hành trang để chúng em có thể vững bước sau này.

Do sự tiếp nhận kiến thức của mỗi chúng em luôn tồn tại những hạn chế nhất định, nên trong bài tập lớn chắc không tránh khỏi những thiếu sót. Vì vậy, chúng em mong nhận được những đóng góp ý kiến đến từ các thầy để bài tập lớn của chúng em đạt được kết quả tốt nhất.

Chúng em xin kính chúc các thầy sức khỏe, hạnh phúc, thành công trên con đường sự nghiệp của mình.

# MỤC LỤC

DANH MỤC HÌNH ẢNH.....	4
DANH MỤC BẢNG BIỂU .....	5
I. ĐẶT VẤN ĐỀ.....	6
1. MỤC ĐÍCH CHỌN ĐỀ TÀI.....	6
II. GIẢI QUYẾT VẤN ĐỀ .....	6
1. CÁCH TIẾP CẬN .....	6
2. TỔNG QUAN VỀ XỬ LÝ ẢNH .....	7
3. SƠ LƯỢC HỆ THỐNG .....	7
3.1. <i>Lựa chọn thiết bị</i> .....	7
3.2. <i>Phần mềm</i> .....	14
4. MÔ PHỎNG HỆ THỐNG .....	17
4.1. <i>Mô phỏng trên proteus</i> .....	17
4.2. <i>Code</i> .....	20
5. THỰC TẾ HỆ THỐNG .....	28
5.1. <i>Phân cứng</i> .....	28
5.2. <i>Một số kết quả</i> .....	29
III. KẾT LUẬN .....	30
1. KẾT LUẬN.....	30
2. HẠN CHẾ .....	31
3. HƯỚNG PHÁT TRIỂN.....	31
IV. MỘT SỐ TÀI LIỆU THAM KHẢO.....	31

## DANH MỤC HÌNH ẢNH

Ảnh 1: Raspberry Pi 4 .....	8
Ảnh 2 : Webcam.....	9
Ảnh 3 : Động cơ servo SG90.....	10
Ảnh 4 : Màn hình LCD 20x04 với module I2C .....	11
Ảnh 5 : Cảm biến PIR .....	12
Ảnh 6 : Buzzer.....	13
Ảnh 7 : Thuật toán face recognition của thư viện OpenCV .....	15
Ảnh 8 : Hệ thống trên mô phỏng Proteus .....	20
Ảnh 9 : Sơ đồ miêu tả hoạt động của hệ thống .....	21
Ảnh 10 : Mô hình cấu trúc tập dataset.....	22
Ảnh 11 : Mô phỏng LCD hiển thị khi gương mặt được tin cậy .....	27
Ảnh 12 : Mô phỏng LCD hiển thị khi gương mặt không được tin cậy .....	27
Ảnh 13 : Mail cảnh báo được gửi về cho chủ nhà.....	28
Ảnh 14 : Hình ảnh mạch mô phỏng hệ thống.....	29
Ảnh 15 : LCD hiển thị khi hệ thống đang set up.....	29
Ảnh 16 : LCD hiển thị khi hệ thống set up xong .....	30
Ảnh 17 : LCD hiển thị khi gửi mail .....	30

## **DANH MỤC BẢNG BIỂU**

Bảng 1 : Một số câu lệnh trong lớp face_recognition được sử dụng trong hệ thống ...	17
Bảng 2 : Các linh kiện trong file mô phỏng trên Proteus .....	19

## **I. ĐẶT VẤN ĐỀ**

### **1. Mục đích chọn đề tài**

Trong những năm gần đây, trên thế giới nghiên cứu ứng dụng công nghệ xử lý và nhận dạng ảnh đang là hướng nghiên cứu tập trung của rất nhiều nhà khoa học trong các lĩnh vực. Từ những năm 1970 khi mà năng lực tính toán của máy tính ngày càng trở nên mạnh mẽ hơn, các máy tính lúc này có thể xử lý được những tập dữ liệu lớn như các hình ảnh, các đoạn video thì một khái niệm nữa về xử lý ảnh ra đời đó là: Thị giác máy – Computer vision. Có thể nói xử lý ảnh số và thị giác máy đã được phát triển và trở thành một lĩnh vực khoa học. Xử lý ảnh số không chỉ nâng cao chất lượng của ảnh mà còn phân tích và lý giải tìm ra giải thuật để ứng dụng vào thực tiễn. Thị giác máy bao gồm lý thuyết và các kỹ thuật liên quan nhằm mục đích tạo ra một hệ thống nhân tạo có thể tiếp nhận thông tin từ các hình ảnh thu được hoặc các tập dữ liệu đa chiều. Việc kết hợp giữa thị giác máy với các kỹ thuật khác như công nghệ thông tin, truyền thông, điều khiển, điều khiển tự động, cơ khí, ... cho chúng ta rất nhiều ứng dụng trong đời sống hàng ngày cũng như trong khoa học, an ninh, y học, quân sự, .. Ngày nay, ứng dụng của thị giác máy đã trở nên rất rộng lớn và đa dạng, len lỏi vào mọi lĩnh vực từ quân sự, khoa học, vũ trụ, cho đến y học, sản xuất và tự động hoá toà nhà. Nhận thức được vấn đề đó, chúng em đã lựa chọn đề tài “*Mở khoá cửa bằng hệ thống nhận diện khuôn mặt*” với hi vọng có thể vận dụng những kiến thức đã được học tập tại môi trường đại học để giải quyết vấn đề đã đặt ra.

## **II. GIẢI QUYẾT VẤN ĐỀ**

### **1. Cách tiếp cận**

Dựa vào các kiến thức đã được học và tích lũy tại trường lớp. Chúng em đã được học các kiến thức thiết kế, kiến thức mô phỏng, kiến thức về lập trình Python, các lý thuyết chuyên ngành về điện tử, đo lường, điều khiển, ... Đó là các công cụ cho chúng em tìm tòi và phát triển các hướng giải quyết các vấn đề đã đặt ra.

Tiếp cận qua Internet, các thông tin từ các nguồn khác nhau : qua các bài báo khoa học, các thông tin tìm hiểu được qua các trang công nghệ và các kênh thông tin thời sự,

chúng em đã tích lũy được một lượng kiến thức nhất định và hướng phát triển về xử lý ảnh.

## **2. Tổng quan về xử lý ảnh**

Xử lý ảnh bao gồm lý thuyết về các kỹ thuật liên quan nhằm mục đích tạo ra một hệ thống nhân tạo có thể tiếp nhận thông tin từ các hình ảnh thu được hoặc các tập dữ liệu đa chiều. Đối với mỗi người chúng ta, quá trình “học” thông qua thế giới bên ngoài là một điều dễ dàng. Quá trình nhận thức đó được “học” thông qua quá trình sống của mỗi người. Tuy nhiên với các vật vô tri vô giác như máy tính, robot, ... thì điều đó quả thực là một bước tiến rất gian nan. Các thiết bị ngày nay không chỉ nhận thông tin ở dạng tín hiệu đơn lẻ mà còn có thể có cái “nhìn” thật với thế giới bên ngoài. Cái “nhìn” này qua quá trình phân tích, kết hợp với các mô hình như máy móc, mạng nơtron, .. sẽ giúp cho thiết bị tiến dần tới một hệ thống nhân tạo có khả năng ra quyết định linh hoạt và đúng đắn hơn nhiều. Một số lĩnh vực ứng dụng công nghệ xử lý ảnh mang tính đột phá như sau

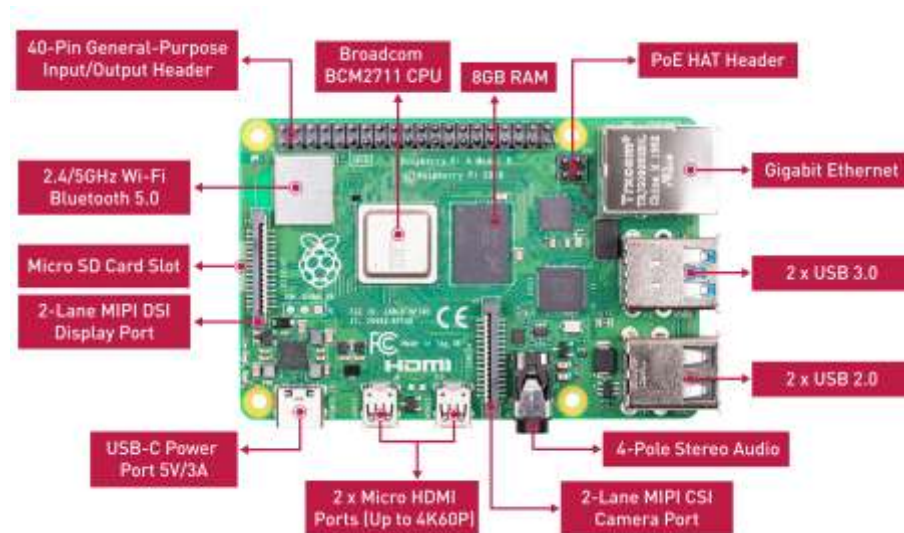
## **3. Sơ lược hệ thống**

### **3.1. Lựa chọn thiết bị**

#### **3.1.1. Raspberry pi 4**

Raspberry Pi là từ để chỉ các máy tính bo mạch đơn (hay còn gọi là máy tính nhúng) kích thước chỉ bằng một thẻ tín dụng, được phát triển tại Anh bởi Raspberry Pi Foundation với mục đích ban đầu là thúc đẩy việc giảng dạy về khoa học máy tính cơ bản trong các trường học và các nước đang phát triển.

Raspberry Pi 4 Model B là phiên bản mới nhất của máy tính Raspberry Pi. Raspberry Pi 4 Model B – 4GB có thể sử dụng như một chiếc PC với các tác vụ lướt web, viết chương trình Python, hoặc đa nhiệm



*Ảnh 1: Raspberry Pi 4*

***Thông số kỹ thuật :***

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB hoặc 8GB LPDDR4-3200 SDRAM (tùy thuộc vào kiểu máy)
- 2,4 GHz và 5,0 GHz IEEE 802.11ac wireless
- Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 cổng USB 3.0, 2 cổng USB 2.0
- Đầu cắm GPIO 40 chân tiêu chuẩn Raspberry Pi
- 2 × cổng micro-HDMI (hỗ trợ lên đến 4kp60)
- Cổng hiển thị MIPI DSI 2 làn
- Cổng camera MIPI CSI 2 làn
- Cổng video tổng hợp và âm thanh stereo 4 cực
- H.265 (giải mã 4kp60), H264 (giải mã 1080p60, mã hóa 1080p30)
- Đồ họa OpenGL ES 3.0
- Khe cắm thẻ nhớ Micro-SD để tải hệ điều hành và lưu trữ dữ liệu
- 5V DC qua đầu nối USB-C (tối thiểu 3A \*)
- 5V DC qua đầu cắm GPIO (tối thiểu 3A \*)



- Bật nguồn qua Ethernet (PoE) (yêu cầu PoE HAT riêng biệt)
- Nhiệt độ hoạt động: 0-50 độ C

### 3.1.2. Webcam



*Ảnh 2 : Webcam*

Webcam là từ viết tắt của Website Camera, đây là loại thiết bị ghi hình kỹ thuật số được kết nối với máy tính để truyền trực tiếp hình ảnh mà nó ghi được đến một máy tính khác hoặc truyền lên một website nào đó thông qua mạng internet.

#### ***Thông số kỹ thuật :***

- Ống kính tiêu cự cố định
- Độ phân giải tối đa : 720p/30fps
- Camera mega pixel : 1.2
- Kết nối USB type A

### 3.1.3 Động cơ servo SG90

Động cơ servo SG90 có kích thước nhỏ, là loại được sử dụng nhiều nhất để làm các mô hình nhỏ hoặc các cơ cấu kéo không cần đến lực nặng.

Động cơ servo SG90 180 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.



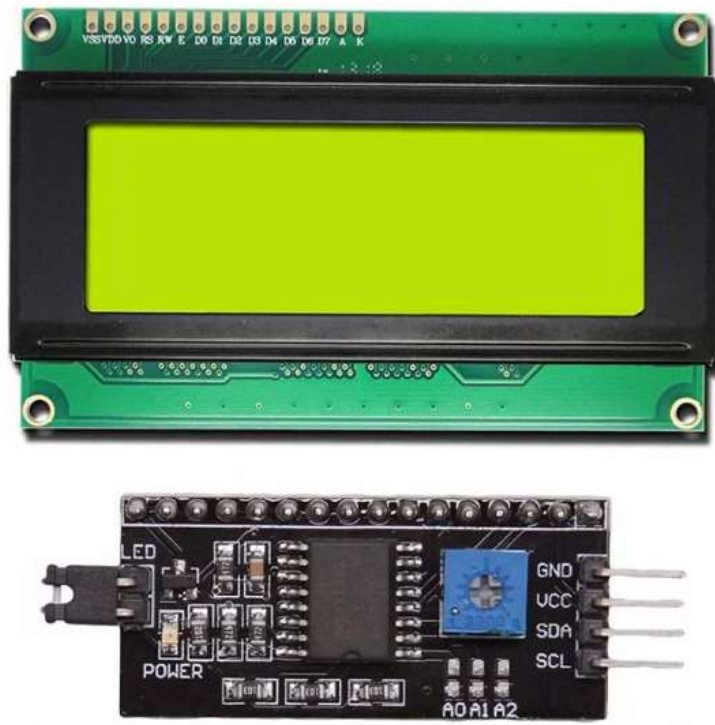
*Ảnh 3 : Động cơ servo SG90*

***Thông số kỹ thuật :***

- *Khối lượng : 9g*
- *Kích thước : 22.2 x 11.8 x 32 mm*
- *Momen xoắn : 1.8kg/cm*
- *Tốc độ hoạt động : 60°/0.1s*
- *Điện áp hoạt động : 4.8V (~5V)*
- *Nhiệt độ hoạt động : 0°C – 55°C*
- *Kết nối dây màu đỏ với 5V, dây màu nâu với mass, dây màu cam với chân phát xung của vi điều khiển. Ở chân xung cấp một xung từ 1ms-2ms theo để điều khiển góc quay theo ý muốn.*

***3.1.4 Màn hình LCD 16x2 giao tiếp I2C***

Màn hình text LCD 2004 kèm module I2C sử dụng driver HD44780, có khả năng hiển thị 4 dòng với mỗi dòng 20 ký tự, màn hình có độ bền cao, rất phổ biến. Màn hình LCD được hàn sẵn module giao tiếp I2C giúp tiết kiệm chân cho vi điều khiển và giúp việc giao tiếp được dễ dàng và nhanh chóng hơn rất nhiều.



*Ảnh 4 : Màn hình LCD 20x04 với module I2C*

***Thông số kỹ thuật của module chuyển đổi i2c :***

- *Kích thước 41.5(L) x 19(W) x 15.3 mm(H)*
- *Trọng lượng ; 5g*
- *Điện áp hoạt động : 2.5-6V DC.*
- *Giao tiếp : I2C*
- *Jump Chốt : Cung cấp đèn cho LCD hoặc ngắt.*
- *Biến trở xoay độ tương phản cho LCD*

***Thông số kỹ thuật màn hình LCD\_2004 :***

- *Điện áp hoạt động : 5V*
- *Kích thước : 98 x 60 x 13.5 mm*
- *Driver : HD44780*
- *Số ô hiển thị : 80 ô nhớ hiển thị*
- *Chân : 16*

- *Nền : Xanh lá, xanh dương*
- *Khoảng cách giữa hai chân kết nối là 0.1 inch*

### 3.1.5 Cảm biến PIR

Cảm biến thân nhiệt chuyển động PIR (Passive infrared sensor) HC-SR501 được sử dụng để phát hiện chuyển động của các vật thể phát ra bức xạ hồng ngoại (con người, con vật, các vật phát nhiệt, ...), cảm biến có thể chỉnh được độ nhạy để giới hạn khoảng cách bắt xa gần cũng như cường độ bức xạ của vật thể mong muốn, ngoài ra cảm biến còn có thể điều chỉnh thời gian kích trễ (giữ tín hiệu bao lâu sau khi kích hoạt) qua biến trở tích hợp sẵn.



*Ảnh 5 : Cảm biến PIR*

#### **Thông số kỹ thuật :**

- *Phạm vi phát hiện : góc 360° hình nón, độ xa tối đa 6m*
- *Nhiệt độ hoạt động : 32 – 122°F (~0 – 50°C)*
- *Điện áp hoạt động : DC 3.8V – 5V*
- *Mức tiêu thụ dòng :  $\leq 50\mu A$*
- *Thời gian báo : 30s có thể tùy chỉnh bằng biến trở.*
- *Độ nhạy có thể điều chỉnh bằng biến trở.*
- *Kích thước : 32.2 x 24.3 x 25.4 mm*

### 3.1.6 Buzzer

Buzzer là một thiết bị tạo ra tiếng còi hoặc tiếng bíp. Có nhiều loại nhưng cơ bản nhất là buzzer áp điện, là một miếng phẳng của vật liệu áp điện với hai điện cực. Loại buzzer này đòi hỏi phải có các bộ dao động (hoặc vi điều khiển) để điều khiển nó. Nếu bạn sử dụng điện áp một chiều, nó chỉ kêu lách cách. Chúng được sử dụng ở những vị trí cần phát ra âm thanh nhưng không quan tâm đến việc tái tạo âm thanh trung thực, như lò vi sóng, báo cháy và đồ chơi điện tử. Chúng rẻ và kêu to mà không cần sử dụng nhiều năng lượng.



Ảnh 6 : Buzzer

#### **Thông số kỹ thuật :**

- Nguồn :  $3.5V - 5.5V$
- Dòng điện tiêu thụ :  $<25mA$
- Tần số cộng hưởng :  $2300Hz \pm 500Hz$
- Biên độ âm thanh :  $>80 dB$
- Nhiệt độ hoạt động :  $-20^{\circ}C - 70^{\circ}C$
- Kích thước :  $12 \times 9.7 mm$

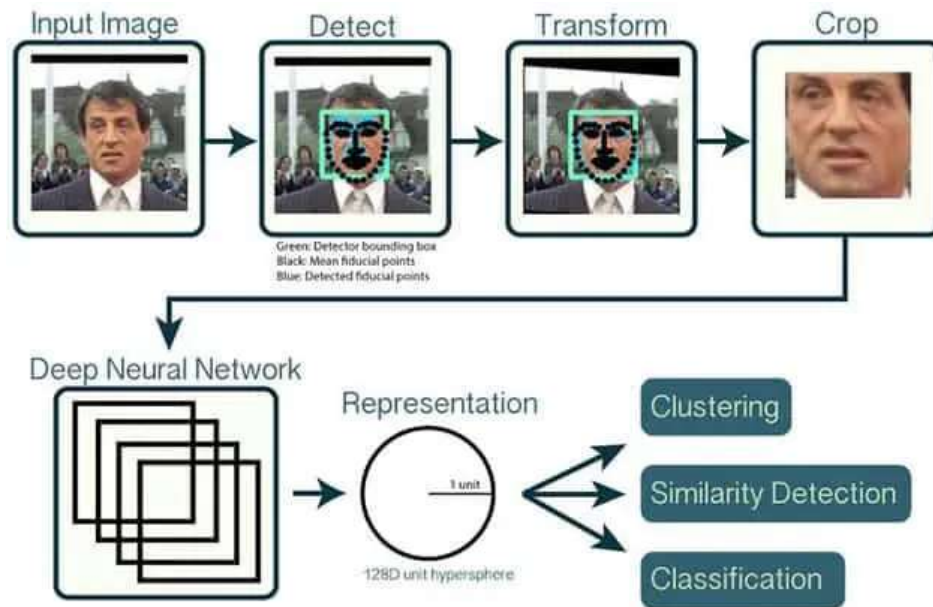
### 3.1.7 Một số thiết bị khác



### 3.2. Phần mềm

Thuật toán được chúng em sử dụng hiện tại là face\_recognition class trong thư viện Open CV . Đây là một thư viện thị giác máy tính phổ biến do Intel bắt đầu vào năm 1999 .

Cách thức hoạt động của thuật toán như sau :



Ảnh 7 : Thuật toán face recognition của thư viện OpenCV

Ban đầu, hình ảnh đầu vào được áp dụng thuật toán nhận diện gương mặt để phát hiện vị trí. Thuật toán này chỉ xác định vị trí của gương mặt chứ không có tính xác thực thông tin của gương mặt. Sau đó, là quá trình cấu trúc lại gương mặt. Ở quá trình này bức ảnh được xử lý bằng các phép xoay, tịnh tiến để gương mặt về dạng chính tắc nhất. Cuối cùng ảnh được đưa vào một mạng nơ-ron học sâu để nhận diện.

Để mạng nơ-ron này có thể phân biệt được các gương mặt với nhau, người ta sử dụng dữ liệu đầu vào để training cho mạng này gồm ba ảnh :

1. *The anchor* : Đây là ảnh chỉ có mặt của người A
2. *The positive image* : Đây là bức ảnh có nhiều người , trong đó có mặt của người A
3. *The negative image* : Đây là bức ảnh nhiều người nhưng không có mặt người A

Qua quá trình huấn luyện thì mạng nơ-ron sẽ thay đổi trọng số của mạng (thông qua hàm bộ ba) sao cho :

- Ảnh nhúng của the anchor và positive image phải gần nhau hơn
- Ảnh nhúng của the anchor và negative image phải xa nhau hơn

**Một số câu lệnh từ thư viện được sử dụng trong hệ thống :**

<code>face_recognition.api.batch_face_locations(image_s, number_of_times_to_upsample=1, batch_size=128)</code>	<p><u>Thông số :</u></p> <ul style="list-style-type: none"> <li>▪ <b>images</b> - Một danh sách các hình ảnh (mỗi hình ảnh là một mảng nhỏ)</li> <li>▪ <b>number_of_times_to_upsample</b> - Số lần lấy mẫu hình ảnh tìm kiếm khuôn mặt. Số cao hơn tìm thấy khuôn mặt nhỏ hơn.</li> <li>▪ <b>batch_size</b> - Số lượng hình ảnh cần bao gồm trong mỗi lô xử lý GPU.</li> </ul> <p><u>Trả về :</u> Một danh sách gồm nhiều vị trí khuôn mặt được tìm thấy theo thứ tự css (trên, phải, dưới, trái)</p>
<code>face_recognition.api.compare_faces(known_face_encodings, face_encoding_to_check, tolerance=0.6)</code>	<p>So sánh danh sách các mã hóa khuôn mặt với một mã hóa ứng viên để xem chúng có khớp hay không.</p> <p><u>Thông số:</u></p> <ul style="list-style-type: none"> <li>▪ <b>known_face_encodings</b> - Danh sách các mã hóa khuôn mặt đã biết</li> <li>▪ <b>face_encoding_to_check</b> - Một mã hóa khuôn mặt duy nhất để so sánh với danh sách</li> <li>▪ <b>tolerance</b> - Khoảng cách giữa các mặt là bao nhiêu để coi là trùng khớp. Thấp hơn là nghiêm ngặt hơn. 0,6 là hiệu suất tốt nhất điển hình.</li> </ul> <p><u>Trả về :</u> Một danh sách các giá trị True / False cho biết known_face_encodings nào khớp với mã hóa khuôn mặt để kiểm tra</p>
<code>face_recognition.api.face_distance(face_encoding_s, face_to_compare)</code>	<p>Đưa ra danh sách các mã hóa khuôn mặt, hãy so sánh chúng với một mã hóa khuôn mặt đã biết và nhận khoảng cách euclidean cho mỗi mặt so sánh. Khoảng cách cho bạn biết các khuôn mặt giống nhau như thế nào.</p> <p><u>Thông số:</u></p> <ul style="list-style-type: none"> <li>▪ <b>face_encodings</b> - Danh sách các mã hóa khuôn mặt để so sánh</li> <li>▪ <b>face_to_compare</b> - Mã hóa khuôn mặt để so sánh với</li> </ul> <p><u>Trả về:</u> Một ndarray numpy với khoảng cách cho mỗi mặt theo thứ tự như mảng 'các mặt'</p>
<code>face_recognition.api.face_encodings(face_image, known_face_locations=None, num_jitters=1, model='small')</code>	<p>Cho một hình ảnh, trả về mã hóa khuôn mặt 128 chiều cho mỗi khuôn mặt trong hình ảnh.</p> <p><u>Thông số:</u></p> <ul style="list-style-type: none"> <li>▪ <b>face_image</b> - Hình ảnh có một hoặc nhiều khuôn mặt</li> <li>▪ <b>known_face_locations</b> - Tùy chọn - các hộp giới hạn của mỗi mặt nếu bạn đã biết chúng.</li> </ul>




	<ul style="list-style-type: none"> <li>▪ <b>num_jitters</b> - Số lần lấy mẫu lại khuôn mặt khi tính toán mã hóa. Cao hơn thì chính xác hơn, nhưng chậm hơn (tức là 100 thì chậm hơn 100 lần)</li> <li>▪ <b>model</b> - Tùy chọn - mô hình nào để sử dụng. “Lớn” hoặc “nhỏ” (mặc định) chỉ trả về 5 điểm nhưng nhanh hơn.</li> </ul> <p><u>Trả về</u>: Danh sách các mã hóa khuôn mặt 128 chiều (một mã hóa cho mỗi khuôn mặt trong hình ảnh)</p>
<pre>face_recognition.api.face_locations(img, number_of_times_to_upsample=1, model='hog')</pre>	<p>Trả về một mảng các hộp giới hạn khuôn mặt người trong một hình ảnh</p> <p><u>Thông số</u>:</p> <ul style="list-style-type: none"> <li>▪ <b>img</b> - Một hình ảnh (dưới dạng một mảng numpy)</li> <li>▪ <b>number_of_times_to_upsample</b> - Số lần lấy mẫu hình ảnh tìm kiếm khuôn mặt. Số cao hơn tìm thấy khuôn mặt nhỏ hơn.</li> <li>▪ <b>model</b> - Sử dụng mô hình nhận diện khuôn mặt nào. "Hog" kém chính xác hơn nhưng nhanh hơn trên CPU. "Cnn" là một mô hình học sâu chính xác hơn được tăng tốc GPU / CUDA (nếu có). Giá trị mặc định là "hog".</li> </ul> <p><u>Trả về</u>: Một danh sách gồm nhiều vị trí khuôn mặt được tìm thấy theo thứ tự css (trên, phải, dưới, trái)</p>

Bảng 1 : Một số câu lệnh trong lớp face\_recognition được sử dụng trong hệ thống

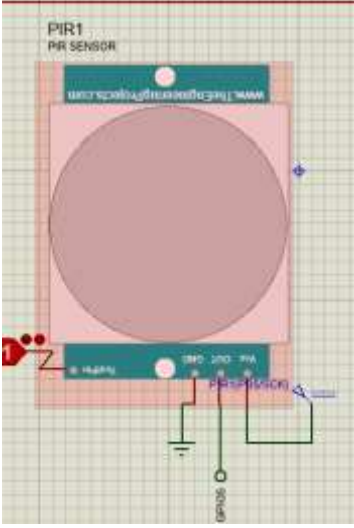
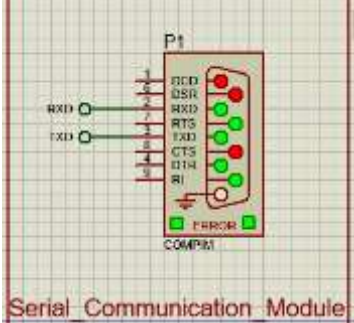
## 4. Mô phỏng hệ thống

### 4.1. Mô phỏng trên proteus

#### 4.1.1. Lấy linh kiện

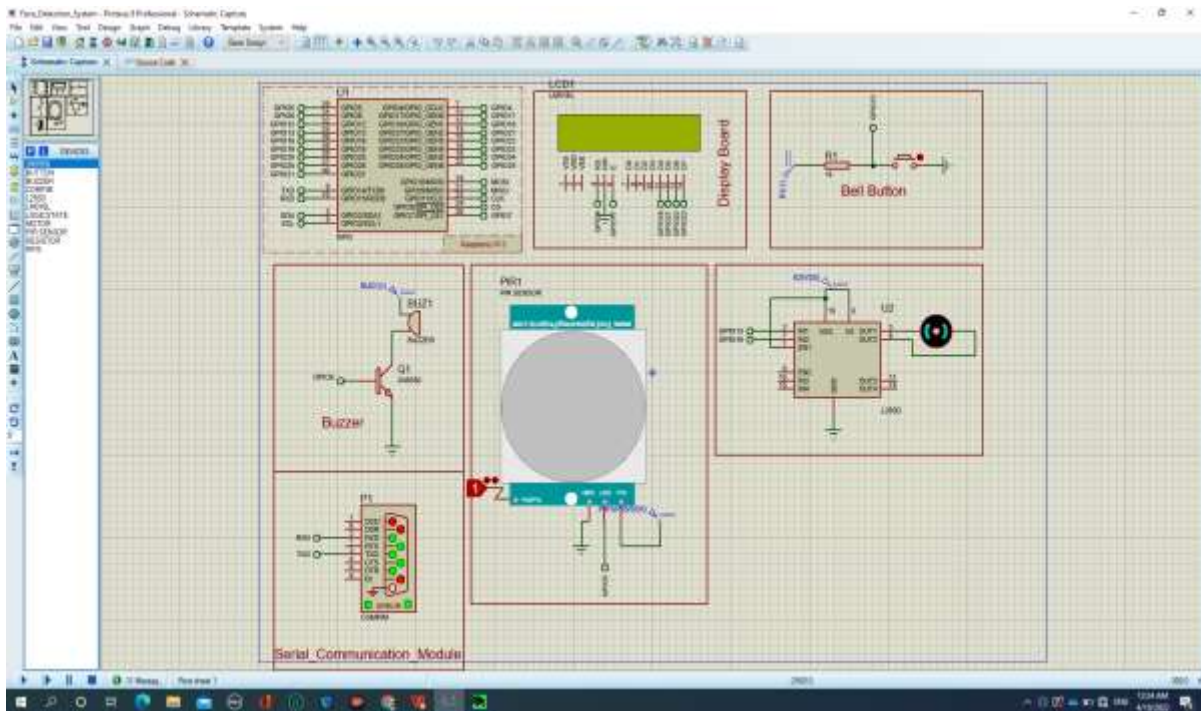
LINH KIỆN	TÊN LINH KIỆN	CHỨC NĂNG
	Raspberry pi 3	Điều khiển hệ thống



	<p>Cảm biến PIR</p>	<p>Nhận dạng chuyển động (chống trộm)</p>
	<p>Cổng giao tiếp Serial</p>	<p>Tạo giao tiếp serial</p>

Bảng 2 : Các linh kiện trong file mô phỏng trên Proteus

#### 4.1.2. Mô phỏng trên proteus



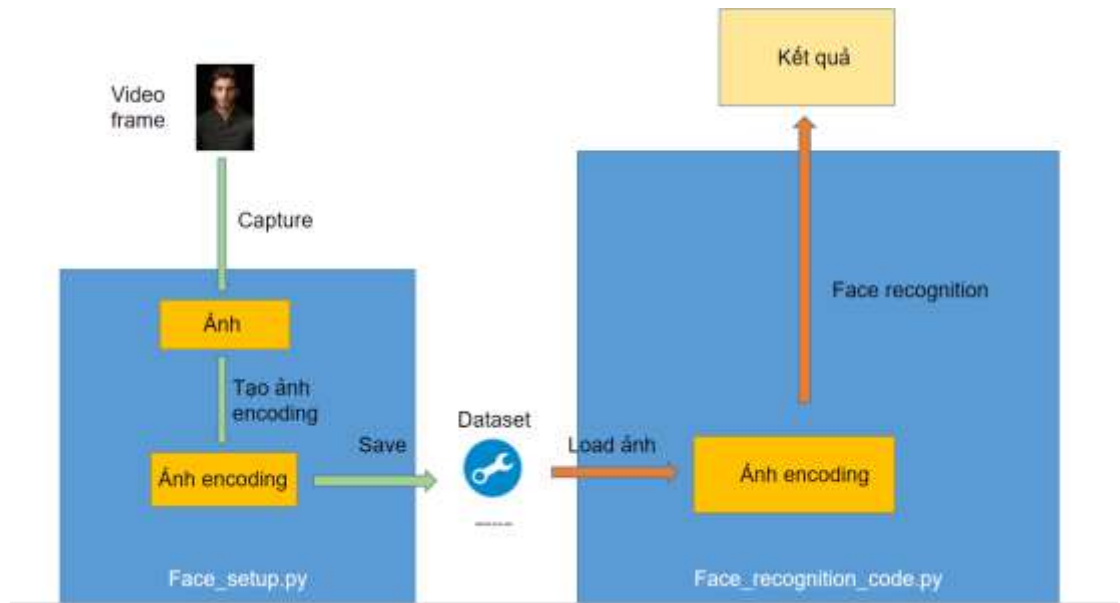
Ảnh 8 : Hệ thống trên mô phỏng Proteus

## 4.2. Code

### 4.2.1. Sơ đồ hoạt động của phần mềm hệ thống

Đây là phần mềm tạo công ảo để phân cứng trên proteus và phần mềm trên IDE kết nối với nhau.

Mô hình hoạt động của hệ thống phần mềm :



Ảnh 9 : Sơ đồ miêu tả hoạt động của hệ thống

#### 4.2.2. Phần mềm

- **Chương trình set up gương mặt (Face\_setup)**

**Ý tưởng :**

*Bước 1. Yêu dùng người dùng nhập tên (ID)*

*Bước 2. Chụp ảnh gương mặt sau đó chuyển nó về dạng encoding (dạng máy tính xử lý)*

*Bước 3. Lưu ID và ảnh encoding người dùng theo dạng thức Dictionary*

*Bước 4. Lưu trữ dictionary đó trong file .dat*

**Code của chương trình :**

```

import face_recognition
import cv2
import numpy as np
import os
import serial
import smtplib
import imghdr
import pickle

face_id = input('\n enter user id end press <return> ==> ')
print("\n [INFO] Initializing face capture. Look the camera and wait ...")
cam = cv2.VideoCapture(0)

while True:

```

```

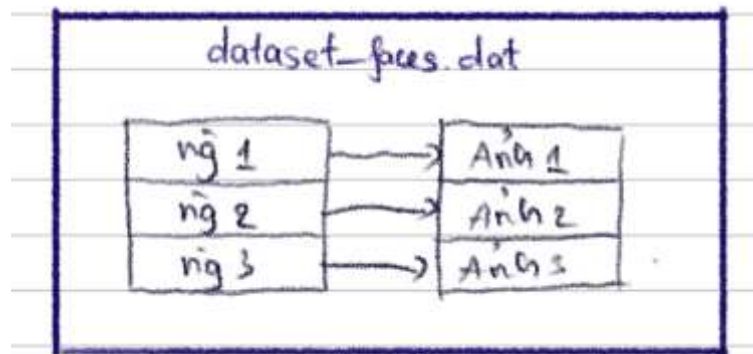
ret, image = cam.read()
cv2.imshow('Imagetest', image)
k = cv2.waitKey(1)
if k != -1:
    break
cv2.imwrite("/home/pi/Desktop/Smart_Door_Lock/Dataset/ "+ face_id
+".jpg", image)
with open('dataset_faces.dat', 'rb') as file:
    all_face_encodings = pickle.load(file)

all_face_encodings[face_id] = face_recognition.face_encodings(image)[0]
with open('dataset_faces.dat', 'wb') as f:
    pickle.dump(all_face_encodings, f)
    print("Done !!!!")
cam.release()
cv2.destroyAllWindows()

```

### **Lý do tạo chương trình :**

Chương trình set up gương mặt giúp hệ thống hoạt động tốt và trực quan hơn . Ngoài việc giúp khởi động hệ thống nhanh hơn vì dataset lúc này ở dạng máy tính có thể đọc ngay được thay vì ảnh thô ( máy sẽ mất quá trình chuyển hoá ) . Hơn nữa việc có chương trình set up gương mặt sẽ giúp người dùng dễ dàng nhập liệu gương mặt hơn .



*Ảnh 10 : Mô hình cấu trúc tập dataset*

- **Chương trình nhận diện và mở khoá cửa (face\_recognition\_system)**

### **Chương trình đơn giản hoá :**

#### **Đầu vào:**

- 1: Setup khuôn mặt
- 2: Setup email
- 3: Nhấn chuông (name)

#### **Đầu ra : Mở cửa / Không mở cửa ( servo xoay góc 90 hay 0 độ)**

- 1: Set up các chân linh kiện và camera

```

2:   Load known_face_encodings , known_face_names
3:   Hệ thống thông báo đã set up xong
4:   While True :
5:       if process_this_frame:
6:           LCD hiển thị “Press Bell”
7:           Hệ thống capture một frame trong video của Webcam
8:           Hệ thống resize frame và chuyển frame từ dạng BGR sang RGB
9:           Ảnh resize → face_locations → face_encodings
10:          for face_encoding in face_encodings do:
11:              Phân biệt gương mặt đang hiển thị (face_encodings) với những
                  gương mặt trong known_face_encodings
12:              if (name == “Unknown”) then
13:                  Cửa không mở
14:                  LCD hiển thị “ I don’t know U” , “Door Close”
15:                  if (inside_home == 1) then
16:                      LCD hiển thị “someone inside home”
17:                      Không gửi hình ảnh vào email cho chủ nhà
18:                  if (inside_home == 0) then
19:                      LCD hiển thị “nobody inside home”
20:                      Gửi hình ảnh vào email cho chủ nhà
21:              else
22:                  Mở cửa
23:                  Servo hoạt động
24:                  LCD hiển thị “Welcome” + name , “Door Open”

```

### **Chi tiết :**

Chương trình được thiết kế làm 2 phần :

- Set up hệ thống và load tập dữ liệu :

```

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

lcd = lcd_driver.lcd(0,1)
lcd.backlight(1)
lcd lcd_display_string("Initilization system ", 1,1)

#Set Up chan cho cac linh kien
buzzer_pin =36
LED_PIN = 29
switch_pin =33
PIR_PIN = 18
servo_pin = 11

# Set chan out va in

```

```

GPIO.setup(servo_pin, GPIO.OUT)
servo1 = GPIO.PWM(11,50)
GPIO.setup(LED_PIN,GPIO.OUT)
GPIO.setup(buzzer_pin,GPIO.OUT)
GPIO.setup(switch_pin,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(PIR_PIN,GPIO.IN)

#Load dataset :
with open('dataset_faces.dat', 'rb') as file:
    all_face_encodings = pickle.load(file)

known_face_encodings = list(all_face_encodings.values())
known_face_names = list(all_face_encodings.keys())

#Set Up camera
video_capture = cv2.VideoCapture(0)
#Khai bao trang thai cua PIR va trang thai mo cua
inside_home = 0
door_open_status =0
#Khai bao cac bien de detect face
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

#Hien thi chu tren man hinh
lcd lcd_clear()
lcd lcd_display_string("Welcome to", 1,1)
lcd lcd_display_string("Face recognition", 2, 1)
lcd lcd_display_string("System", 3, 1)
time.sleep(1.5)

GPIO.output(LED_PIN,GPIO.LOW)
GPIO.output(buzzer_pin,GPIO.LOW)
servo1.start(0)

```

#### o Nhận diện gương mặt và mở khoá :

```

while True:
    #Lay tin hieu tu switch
    servo1.ChangeDutyCycle(2)
    begin = 0
    door_open_status = 0
    lcd lcd_clear()
    lcd lcd_display_string("Press bell", 1,1)
    time.sleep(1)
    # Grab a single frame of video
    ret, frame = video_capture.read()
    # Resize frame of video to 1/4 size for faster face
    recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

```



```

    # Convert the image from BGR color (which OpenCV uses) to RGB
    color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, :-1]
    if GPIO.input(switch_pin) == GPIO.HIGH:
        begin = 1
    if begin == 1 :
        GPIO.output(LED_PIN,GPIO.HIGH)  #LED ON
        GPIO.output(buzzer_pin,GPIO.HIGH)  #LED ON
        time.sleep(1)
        GPIO.output(LED_PIN,GPIO.LOW)  #LED ON
        GPIO.output(buzzer_pin,GPIO.LOW)  #LED ON

        # Only process every other frame of video to save time
        if process_this_frame:
            # Find all the faces and face encodings in the
            current frame of video
            face_locations =
            face_recognition.face_locations(rgb_small_frame)
            face_encodings =
            face_recognition.face_encodings(rgb_small_frame, face_locations)

            face_names = []
            for face_encoding in face_encodings:
                # See if the face is a match for the known
                face(s)

                matches =
                face_recognition.compare_faces(known_face_encodings,
                face_encoding)

                name = "Unknown"

                # Or instead, use the known face with the
                smallest distance to the new face
                face_distances =
                face_recognition.face_distance(known_face_encodings,
                face_encoding)

                best_match_index = np.argmin(face_distances)
                if matches[best_match_index]:
                    if min(face_distances) <= 0.35 :
                        name =
                        known_face_names[best_match_index]
                        face_names.append(name)
                if(name == "Unknown"):
                    lcd.lcd_clear()
                    lcd.lcd_display_string("I dont know U" +
                    name, 1,1)

                    lcd.lcd_display_string(name,2,1)
                    lcd.lcd_display_string("Door Close" ,3,1)
                    time.sleep(0.2)
                    if GPIO.input(PIR_PIN) == GPIO.HIGH:
                        #someone inside home
                        print("someone inside home")
                        lcd.lcd_clear()
                        lcd.lcd_display_string("someone
inside home", 1,1)

```

```

        time.sleep(1)
        inside_home = 1
    else :
        #noone inside home
        print("nobody inside home")
        lcd lcd_clear()
        lcd lcd_display_string("nobody inside
home", 1,1)

        time.sleep(1)
        inside_home = 0
        #if someone inside home then do not send
image on email

        if(inside_home == 0):
            print("sending image on mail")
            lcd lcd_clear()
            lcd lcd_display_string("Dang gui
mail", 1,1)

            lcd lcd_display_string("canh bao",
2,1)

            #return_value, image =
video_capture.read()

            cv2.imwrite('opencv.png', frame)

sendemail.SendEmail('ProPythonLQS@gmail.com','LQS123456','tungbod@
gmail.com','opencv.png')
        else:
            lcd lcd_clear()
            lcd lcd_display_string("Welcome " + name,
1,1)

            lcd lcd_display_string("Door Open ", 2,1)
            time.sleep(0.2)
            if(door_open_status == 0):
                door_open_status = 1
                serv01.ChangeDutyCycle(7)
                time.sleep(5)

        process_this_frame = not process_this_frame

```

- **Chương trình gửi mail cảnh báo :**

Chương trình sẽ gửi mail cảnh báo về cho chủ nhà khi cảm biến PIR phát hiện nhà không có người bên trong và gương mặt nhận diện không được tin cậy :

```

from email.message import EmailMessage
import imghdr
import smtplib
def SendEmail (sender,pass_sender,reciever,image) :
    Sender_Email = sender
    Reciever_Email = reciever
    Password = pass_sender
    newMessage = EmailMessage()
    newMessage['Subject'] = "CANH BAO !!!"
    newMessage['From'] = Sender_Email
    newMessage['To'] = Reciever_Email

```

```

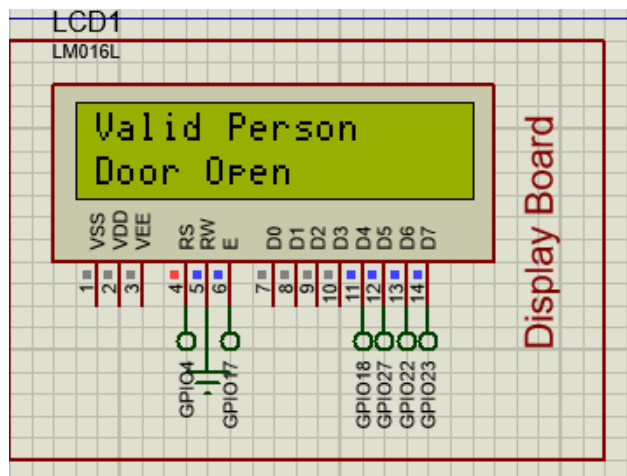
newMessage.set_content('CANH BAO AN NINH')
with open(image, 'rb') as f:
    image_data = f.read()
    image_type = imghdr.what(f.name)
    image_name = f.name
newMessage.add_attachment(image_data, maintype='image',
    subtype=image_type, filename=image_name)
with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
    smtp.login(Sender_Email, Password)
    smtp.send_message(newMessage)

```

- Ngoài ra còn có các driver hỗ trợ cảm biến PIR và LCD2004

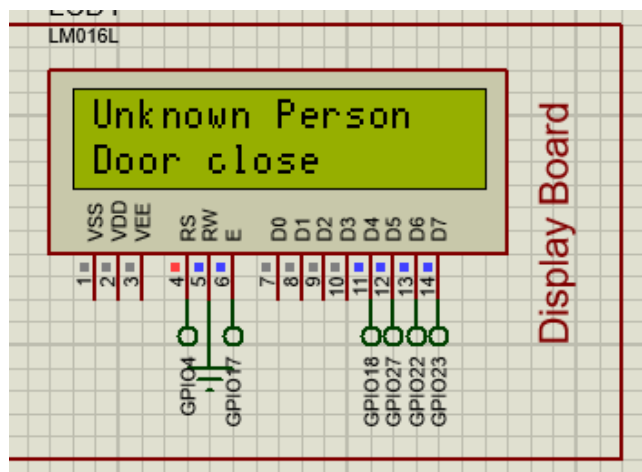
#### 4.3. Kết quả mô phỏng

Nếu đúng thì hệ thống thông báo lên LCD mở cửa cho người dùng

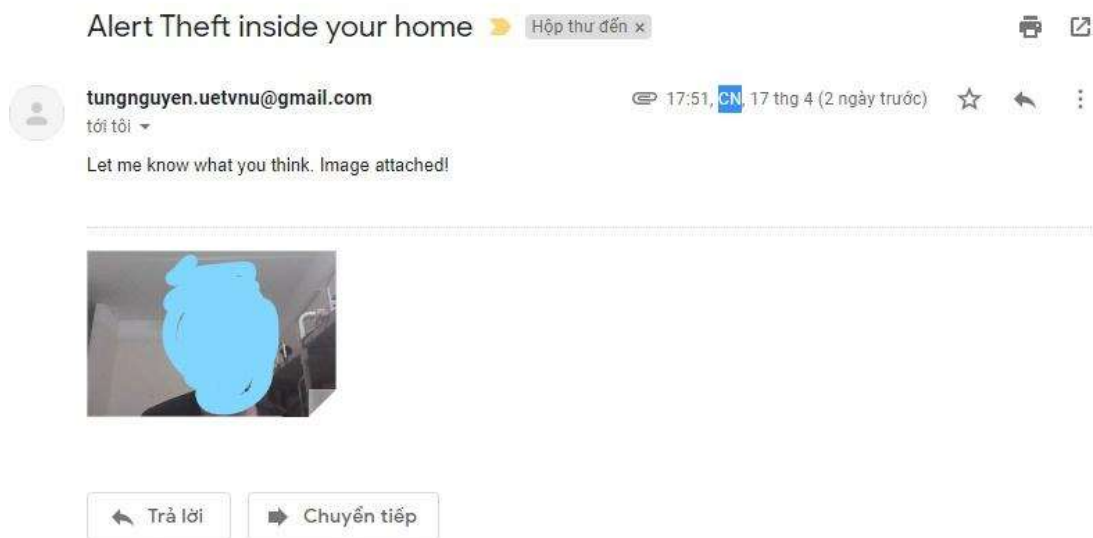


Ảnh 11 : Mô phỏng LCD hiển thị khi gương mặt được tin cậy

Nếu không đúng thì hệ thống thông báo không mở cửa và gửi mail cho chủ nhà nếu email cảm biến PIR đang ở mức 1 còn mức 0 thì không gửi mail.



Ảnh 12 : Mô phỏng LCD hiển thị khi gương mặt không được tin cậy

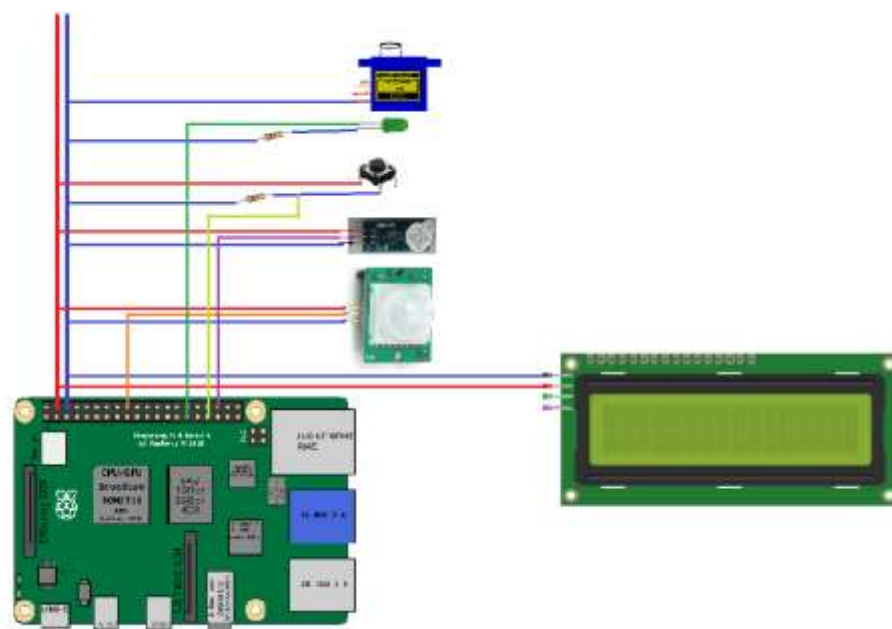


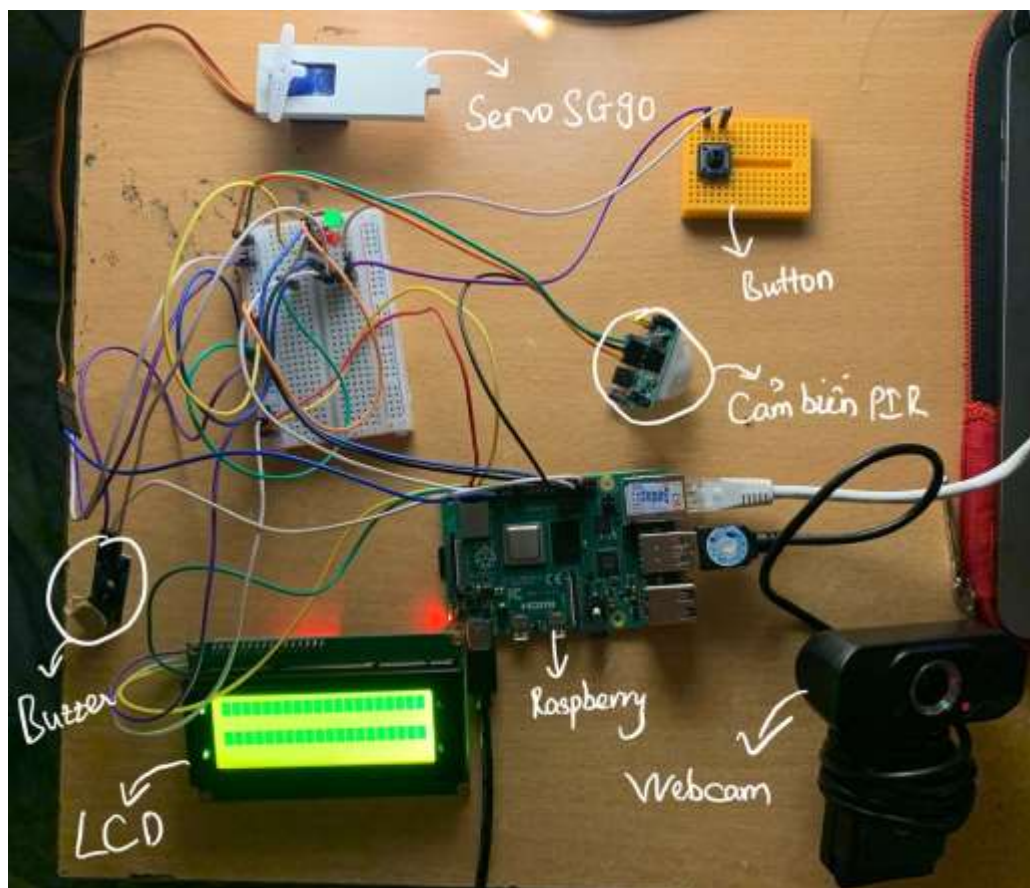
*Ảnh 13 : Mail cảnh báo được gửi về cho chủ nhà*

## 5. Thực tế hệ thống

### 5.1. Phần cứng

Sơ đồ nối dây :





Ảnh 14 : Hình ảnh mạch mô phỏng hệ thống

## 5.2. Một số kết quả



Ảnh 15 : LCD hiển thị khi hệ thống đang set up



*Ảnh 16 : LCD hiển thị khi hệ thống set up xong*



*Ảnh 17 : LCD hiển thị khi gửi mail*

### **III. KẾT LUẬN**

#### **1. Kết luận**

Qua quá trình nghiên cứu, thiết kế, xây dựng mô hình đề tài nghiên cứu của chúng rm đã thực hiện được một số công việc sau :

- Đã hoàn thành mô phỏng và mô hình thực tế

- Đã xây dựng thành công thuật toán nhận diện khuôn mặt.
- Đã có phương hướng để khắc phục những khuyết điểm và tăng độ chính xác như mong muốn.

## **2. Hạn chế**

- Hệ thống chưa đạt độ chính xác như mong muốn (*VD : chưa nhận diện được người đeo kính, đeo khẩu trang*)

## **3. Hướng phát triển**

- Hoàn thiện phần cứng
- Xây dựng GUI để set up khuôn mặt
- Thêm nhận dạng đeo kính

## **IV. MỘT SỐ TÀI LIỆU THAM KHẢO**

1. <https://pyimagesearch.com/2018/09/24/opencv-face-recognition/>
2. <https://face-recognition.readthedocs.io/en/latest/index.html>
3. [https://vi.wikipedia.org/wiki/Raspberry\\_Pi](https://vi.wikipedia.org/wiki/Raspberry_Pi)