

BAN CƠ YẾU CHÍNH PHỦ  
HỌC VIỆN KỸ THUẬT MẬT MÃ



CAO THỊ THÙY LINH

**ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI:**

**XÂY DỰNG HỆ THỐNG CHỐNG TRỘM THÔNG MINH  
SỬ DỤNG RASPBERRY PI VÀ THUẬT TOÁN HỌC MÁY**

**Ngành: Công nghệ thông tin**

**Mã số: 7480201**

**Hà Nội, 2022**

BAN CƠ YẾU CHÍNH PHỦ  
HỌC VIỆN KỸ THUẬT MẬT MÃ



**ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI:**

**XÂY DỰNG HỆ THỐNG CHỐNG TRỘM THÔNG MINH  
SỬ DỤNG RASPBERRY PI VÀ THUẬT TOÁN HỌC MÁY**

**Ngành: Công nghệ thông tin**

**Mã số: 7480201**

**Họ và tên sinh viên: Cao Thị Thùy Linh**

**Khóa: CT2**

**Người hướng dẫn:**

**ThS. Lê Thị Hồng Vân**

**Học viện Kỹ thuật mật mã**

**Hà Nội, 2022**

## LỜI CẢM ƠN

Trong quá trình thực hiện đồ án tốt nghiệp, em đã nhận được sự giúp đỡ, hướng dẫn, hỗ trợ và động viên từ gia đình, quý thầy cô cùng các bạn. Nhờ đó mà em đã hoàn thành được đồ án như mong muốn. Trong những dòng đầu tiên của đồ án tốt nghiệp này, em muốn gửi lời cảm ơn và biết ơn chân thành nhất của mình tới tất cả những người đã hỗ trợ, giúp đỡ em về kiến thức và tinh thần trong quá trình thực hiện đồ án.

Đầu tiên, em xin gửi lời cảm ơn chân thành tới các thầy cô trong Học viện Kỹ thuật mật mã nói chung và khoa Công nghệ thông tin nói riêng đã tận tình giảng dạy, truyền đạt cho tôi những kiến thức và kinh nghiệm quý báu trong suốt thời gian là sinh viên học viện; cũng như sự quan tâm và tạo mọi điều kiện thuận lợi cho em trong quá trình thực hiện đồ án tốt nghiệp này.

Và để hoàn thành đồ án tốt nghiệp này, em xin được gửi lời cảm ơn chân thành và sự biết ơn sâu sắc tới giảng viên hướng dẫn ThS. Lê Thị Hồng Vân đã tận tình giúp đỡ, trực tiếp chỉ dạy, hướng dẫn em trong suốt quá trình làm đồ án.

Sau cùng xin được gửi lời cảm ơn chân thành tới gia đình, anh chị, bạn bè đã động viên, đóng góp ý kiến và giúp đỡ em trong quá trình học tập, nghiên cứu và hoàn thành đồ án.

Em xin chân thành cảm ơn!

**Sinh viên thực hiện**

Cao Thị Thùy Linh

## MỤC LỤC

MỞ ĐẦU .....	1
1. Tính cấp thiết của đề tài .....	1
2. Mục tiêu nghiên cứu của đề tài .....	1
3. Bài toán đặt ra cho đồ án.....	2
4. Đối tượng và phạm vi nghiên cứu.....	3
4.1. Đối tượng .....	3
4.2. Phạm vi.....	3
5. Các nhiệm vụ chính cần thực hiện .....	3
6. Kết quả dự kiến .....	3
6.1. Lý thuyết .....	3
6.2. Thực nghiệm .....	4
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT .....	5
1.1. Máy tính nhúng Raspberry Pi và các phần cứng khác.....	5
1.1.1. Máy tính nhúng Raspberry Pi .....	5
1.1.2. Các thiết bị phần cứng khác .....	9
1.2. Tổng quan về giao thức MQTT .....	11
1.2.1. Giới thiệu chung về giao thức MQTT.....	11
1.2.2. Mô hình Publish/Subscriber trong giao thức MQTT.....	13
1.2.3. Cơ chế hoạt động của giao thức MQTT.....	14
1.2.4. Các khái niệm trong MQTT.....	15
1.3. Tổng quan về SSL/TLS.....	16
1.3.1. Khái niệm .....	16
1.3.2. Hoạt động của SSL/TLS .....	17
1.3.3. Các giao dịch SSL/TLS.....	18
1.3.4. Bộ mã hóa và SSL/TLS.....	18
1.4. Mã hóa AES .....	19
1.4.1. Khái niệm .....	20
1.4.2. Hoạt động của AES .....	20
1.4.3. Thuật toán AES .....	21
1.4.4. Bảo mật của mã hóa AES.....	24
1.4.5. Mã hóa AES 256 CBC .....	25

1.5.	Tổng quan về Flutter, ngôn ngữ lập trình Dart .....	26
1.5.1.	Ngôn ngữ lập trình Dart .....	26
1.5.2.	Framework Flutter .....	27
<b>CHƯƠNG 2. THUẬT TOÁN HỌC MÁY VÀ NHẬN DẠNG KHUÔN MẶT</b>		<b>30</b>
2.1.	Thuật toán học máy .....	30
2.1.1.	Tổng quan .....	30
2.1.2.	Các ứng dụng của học máy .....	31
2.2.	Thuật toán nhận dạng khuôn mặt .....	32
2.2.1.	Bài toán về nhận dạng khuôn mặt .....	32
2.2.2.	Hệ thống nhận dạng khuôn mặt .....	36
<b>CHƯƠNG 3. TRIỂN KHAI, THỰC NGHIỆM VÀ ĐÁNH GIÁ</b>		<b>44</b>
3.1.	Triển khai, thực nghiệm .....	44
3.1.1.	Sơ đồ hệ thống .....	44
3.1.2.	Thi công phần cứng .....	48
3.1.3.	Thiết kế phần mềm cho các chương trình .....	49
3.2.	Thực nghiệm .....	95
3.3.	Đánh giá .....	96
<b>KẾT LUẬN</b> .....		<b>98</b>
1.	Kết luận .....	98
2.	Mã nguồn sản phẩm .....	99
<b>TÀI LIỆU THAM KHẢO</b> .....		<b>99</b>

## DANH MỤC HÌNH VẼ

Hình 0- 1. Sơ đồ giải thuật của hệ thống.....	2
Hình 1- 1 Các kết nối của raspberry pi 3-Model B .....	6
Hình 1- 2. Giao diện hệ điều hành raspbian.....	7
Hình 1- 3. Webcam See3cam.....	10
Hình 1- 4. Mô hình Publish/Subscriber trong giao thức mqtt.....	13
Hình 1- 5. Cơ chế hoạt động của giao thức mqtt .....	15
Hình 1- 6. Trình tự các giao dịch của ssl/tls .....	18
Hình 1- 7. Minh họa bộ mật mã hiện đại được sử dụng trong SSL/TLS.....	19
Hình 1- 8. Thuật toán mã hóa AES .....	21
Hình 1- 9. Bước SubBytes trong mã hóa AES.....	22
Hình 1- 10. Bảng trạng thái của phép biến đổi SubBytes .....	22
Hình 1- 11. Bước ShiftRows trong mã hóa AES .....	23
Hình 1- 12. Bước MixColumns trong mã hóa AES.....	23
Hình 1- 13. Biểu diễn của 1 đa thức $f(x)$ trong GF (28).....	24
Hình 1- 14. Bước AddRoundKey trong mã hóa AES.....	24
Hình 1- 15. Chế độ mã hóa/ giải mã CBC .....	25
Hình 1- 16. Đặc điểm Fast Development của Flutter .....	28
Hình 1- 17. Đặc điểm Expressive and Flexible UI của Flutter .....	29
Hình 2- 1. Các nhóm giải thuật học máy .....	31
Hình 2- 2. Các ứng dụng của học máy.....	32
Hình 2- 3. Các thông tin có trong ảnh người .....	33
Hình 2- 4. Các thách thức của nhận dạng khuôn mặt .....	34
Hình 2- 5. Một số ứng dụng của nhận dạng khuôn mặt.....	35
Hình 2- 6. Các bước phát hiện phân tầng Cascade classifier.....	37
Hình 2- 7. Ví dụ minh họa sau khi dùng Haarcascade để nhận diện khuôn mặt ....	39
Hình 2- 8. Trích xuất các tính năng cục bộ từ hình ảnh.....	40
Hình 2- 9. Ví dụ về phép toán LBPH.....	41
Hình 2- 10. Trích xuất biểu đồ từ hình ảnh.....	42
Hình 3- 1. Sơ đồ tổng quan của hệ thống.....	44
Hình 3- 2. Sơ đồ phân cứng của hệ thống.....	45
Hình 3- 5. Sơ đồ luồng dữ liệu tin nhắn giữa MQTT Broker và các thiết bị khác .	46
Hình 3- 6. Sơ đồ use case chức năng cho gia chủ.....	47
Hình 3- 7. Phân cứng thi công.....	48
Hình 3- 8. Mô hình sau khi lắp ráp .....	48
Hình 3- 9. Phát hiện khuôn mặt trong khung hình.....	52

Hình 3- 10. Huấn luyện dữ liệu khuôn mặt của những người được nhận dạng.....	55
Hình 3- 11. Trang đăng nhập của phần mềm ứng dụng.....	61
Hình 3- 12. Trang điều hướng chức năng của phần mềm ứng dụng .....	62
Hình 3- 13. Trang hiển thị danh sách tên các khuôn mặt đã đăng ký.....	62
Hình 3- 14. Cửa sổ thay đổi tên khuôn mặt đã đăng ký.....	64
Hình 3- 15. Trang thêm khuôn mặt mới của phần mềm ứng dụng.....	69
Hình 3- 16. Trang thu thập dữ liệu trong phần mềm ứng dụng .....	70
Hình 3- 17. Trang nhận dạng khuôn mặt trong phần mềm ứng dụng.....	72
Hình 3- 18. Kết quả lưu hình ảnh về mobile.....	91
Hình 3- 19. Hiện cảnh báo trên mobile khi có người lạ.....	94
Hình 3- 20. Giao diện ứng dụng trên mobile .....	94

## DANH MỤC BẢNG

Bảng 1- 1. Bảng thông số của Wemos .....	10
--	----

## DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Từ đầy đủ
AES	Advanced Encryption Standard
AI	Artificial Intelligence
CA	Certificate Authority
CBC	Cipher Block Chaining
CPU	Central Processing Unit
DHE	Diffie-Hellman
DIY	Do It Yourself
ECDHE	Elliptic Curve Diffie-Hellman
FR	Face Recognition
FPS	Frames Per Second
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
HDMI	High-Definition Multimedia Interface
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet service provider
LBPH	Local binary patterns histograms
LCD	Liquid-Crystal Display
Mô hình OSI	Open Systems Interconnection Reference Model
ML	Machine Learning
MQTT	Message Queueing Telemetry Transport
OASIS	Organization For The Advancement Of Structured Information Standards
PWM	pulse width modulation
QoS	Quality of service
RAM	Random Access Memory



Rpi	Raspberry Pi
SDK	Software development kit
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
Thẻ SD	Secure Digital card
TLS	Transport Layer Security
UI	User Interface
USB	Universal Serial Bus
VTV	Vietnam Television- Đài truyền hình Việt Nam

## MỞ ĐẦU

### 1. Tính cấp thiết của đề tài

Trong tình hình dịch bệnh hiện nay, đặc biệt là dịp cuối năm thì trộm cắp tài sản đang là loại tội phạm phổ biến nhất. Kẻ trộm cắp hiện hữu ở khắp nơi từ thành phố về nông thôn như trộm cắp tài sản ở các khu chung cư, khu trọ, khu nhà dân, trường học. Một số thống kê điển hình sau trên báo điện tử VTV như:

Thủ đô Hà Nội: Theo thống kê từ năm 2020 đến nay, đã có 149 vụ xảy ra tại các khu chung cư, khu đô thị mới tại Hà Nội, chiếm 2,5% tổng số vụ phạm pháp hình sự, từ cướp tài sản, cờ bạc, hoạt động tín dụng đen nhưng chiếm phần lớn vẫn là biến chung cư trở thành nơi ăn chơi thác loạn, tụ tập bay lắc v.v.[1]

Thành phố Hồ Chí Minh: Theo thống kê của Công an TP. HCM, trong tháng 10/2021, thành phố xảy ra 184 vụ phạm pháp hình sự. So với cùng kỳ giảm hơn 44% nhưng so với thời gian liền kề (tháng 8 và 9) thì tăng 60-80%. Cơ cấu tội phạm vẫn chủ yếu là xâm phạm sở hữu, trộm cắp tài sản, v.v. [2]

Trong các vụ trên thì các đối tượng trộm cắp có những thủ đoạn hết sức tinh vi như lợi dụng thăm người thân để đột nhập nhà khác, lợi dụng người dân ngủ say lúc tối khuya về sáng để lẻn vào trộm, v.v. Dù ở mức trộm cắp nào đi chăng nữa cũng để lại hậu quả ảnh hưởng trực tiếp cho gia chủ. Hầu như các vụ trộm cắp này đều hướng tới các khu chung cư cũ, an ninh lỏng lẻo, nhà dân chưa có camera để thực hiện các hành vi này. Hiện nay khóa đang là hình thức được sử dụng thông dụng nhất, tuy nhiên các đối tượng trộm cắp này hoàn toàn có thể bẻ, phá được khóa, thậm chí là khóa điện tử. Vì vậy, việc lắp đặt một hệ thống chống trộm thông minh là rất cần thiết.

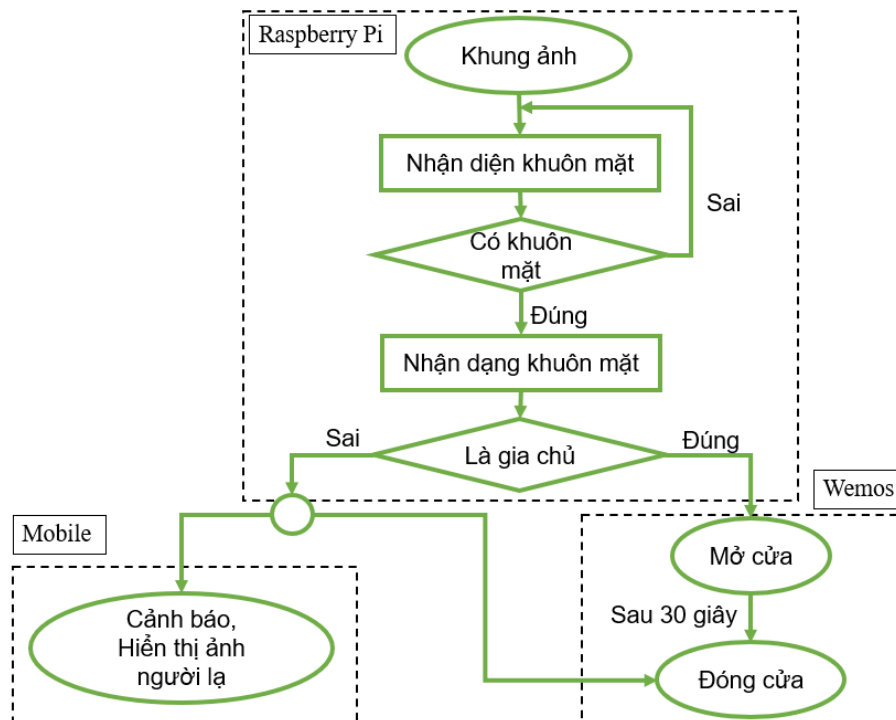
Thêm vào đó, cùng với xu hướng phát triển mạnh mẽ của công nghệ 4.0, ngành công nghệ nói chung và ngành nhúng nói riêng đã và đang phát triển từng ngày, xây dựng thị trường đầy tiềm năng tại Việt Nam. Những sản phẩm mà công nghệ nhúng đem lại với con người đã đóng góp một phần quan trọng, không thể thiếu trong cuộc sống này. Đặc biệt là những sản phẩm có áp dụng trí tuệ nhân tạo AI sử dụng thuật toán học máy

Chính vì cần có biện pháp hiệu quả hơn trong các vấn đề an ninh, phòng chống rủi ro có thể xảy ra nên em chọn đề tài “Xây dựng hệ thống chống trộm thông minh sử dụng Raspberry Pi và thuật toán học máy” để làm đồ án tốt nghiệp

### 2. Mục tiêu nghiên cứu của đề tài

Xây dựng hệ thống chống trộm thông minh sử dụng Raspberry và thuật toán học máy, áp dụng hệ thống vào thực tiễn

### 3. Bài toán đặt ra cho đề án



Hình 0- 1. Sơ đồ giải thuật của hệ thống

Khi gia chủ hoặc người lạ muốn vào nhà thì webcam lấy các khung ảnh để Raspberry nhận diện rồi nhận dạng khuôn mặt.

- Nếu là gia chủ thì Raspberry Pi sẽ gửi tin nhắn được mã hóa theo AES 256 CBC thông qua MQTT Broker đến Wemos giải mã và cửa mở, sau 30 giây cửa sẽ tự động đóng lại.
- Nếu là người lạ thì cửa vẫn đóng và raspberry có chức năng cắt frame ảnh có người lạ, mã hóa ảnh theo AES 256 CBC và gửi đến thiết bị di động của gia chủ thông qua MQTT Broker, thiết bị di động sẽ giải mã ảnh và hiển thị trên màn hình, đồng thời đưa ra cảnh báo

Để tăng sự bảo mật thì sử dụng bảo mật SSL/TLS để xác thực các thiết bị liên quan với MQTT Broker thông qua các chứng chỉ được tạo ra

Trên Raspberry: Có thêm các chức năng thêm, sửa, xóa các khuôn mặt

Trên Mobile: Có thêm chức năng điều khiển đóng mở cửa bằng nút và bằng giọng nói

Trên phần cứng: Có thêm chức năng đóng mở cửa bằng nút bấm

## **4. Đối tượng và phạm vi nghiên cứu**

### **4.1. Đối tượng**

Hệ thống chống trộm dựa trên nhận diện khuôn mặt bằng thuật toán học máy

### **4.2. Phạm vi**

- Máy tính Raspberry, thiết bị phần cứng khác
- Thuật toán nhận dạng khuôn mặt
- Giao thức MQTT
- Bảo mật SSL/TLS
- Mã hóa AES256CBC
- Thư viện hỗ trợ:
  - Raspberry Pi: Opencv, tkinter, aes, base64, paho.mqtt.client
  - Wemos: WIFIClientSecure, PubSubClient, Servo, base64
  - Mobile: mqtt\_client, encrypt, path\_provider, các thư viện hệ thống
- Ngôn ngữ: Python, C++, dart
- Công cụ hỗ trợ: Arduino IDE, Android Studio

## **5. Các nhiệm vụ chính cần thực hiện**

Nội dung nghiên cứu được tập trung vào những nội dung chính như

- Tổng quan về Raspberry, phần cứng
- Tổng quan về thuật toán học máy, nhận dạng khuôn mặt
- Tổng quan về giao thức MQTT
- Bảo mật SSL/TLS
- Tổng quan về mã hóa AES256CBC
- Tổng quan về Flutter, ngôn ngữ lập trình Dart
- Triển khai, thử nghiệm, đánh giá

## **6. Kết quả dự kiến**

### **6.1. Lý thuyết**

- Hiểu được cách hoạt động của hệ thống nhúng, cụ thể là hệ thống chống trộm thông minh
- Nguyên lý hoạt động của thuật toán nhận dạng khuôn mặt, áp dụng thuật toán vào hệ thống
- Tìm hiểu cách giao tiếp của giao thức MQTT, bảo mật SSL/TLS
- Tìm hiểu và mã hóa dữ liệu bằng mã hóa AES 256 CBC

## **6.2. Thực nghiệm**

Xây dựng và thử nghiệm hệ thống chống trộm thông minh sử dụng Raspberry Pi và thuật toán nhận dạng khuôn mặt có chức năng như sau:

- Hệ thống chống trộm có chức năng phát hiện người lạ, cảnh báo và truyền hình ảnh đến cho chủ nhà
- Sử dụng thuật toán học máy để phát hiện khuôn mặt lạ

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

Trước khi đi vào thực hiện đồ án này, thì đầu tiên đưa ra các cơ sở lý thuyết liên quan đến đồ án, cụ thể là máy tính nhúng Raspberry Pi, các thiết bị phần cứng cần thiết, giao thức MQTT để gửi các tín hiệu đến các thiết bị, bảo mật SSL/TLS để xác thực các thiết bị với MQTT broker, mã hóa AES để tăng thêm tính an toàn cho dữ liệu hệ thống

### 1.1. Máy tính nhúng Raspberry Pi và các phần cứng khác

#### 1.1.1. Máy tính nhúng Raspberry Pi

Máy tính nhúng Raspberry Pi là máy tính nhúng rất gần gũi với các sinh viên khoa công nghệ thông tin ngành nhúng, nó nhỏ gọn, thân thiện và có giá thành hợp lý, vì vậy trong đồ án này sử dụng Raspberry Pi để xử lý các khung ảnh để nhận biết được ai là gia chủ, ai là người lạ để đưa ra cảnh báo với gia chủ. Trong bài báo cáo này sẽ tập chung giới thiệu về Raspberry pi 3B+. Ở đây sẽ đặt ra câu hỏi là: "Tại sao không sử dụng raspberry 4?". Câu trả lời được đưa ra: "vì máy tính nhúng có chức năng như đã nêu trên, điều quan tâm là tốc độ xử lý ảnh, tốc độ nhanh hay chậm là do tốc độ CPU. CPU của Raspberry Pi 4 là ARM Cortex A72 1.5GHz. Trên lý thuyết, con chip này có tốc độ xử lý nhanh hơn Raspberry Pi 3B+ một chút [12] mà trong khi giá thành của Raspberry Pi 4 cao hơn nhiều so với Raspberry Pi 3B+ "

##### 1.1.1.1. Khái niệm

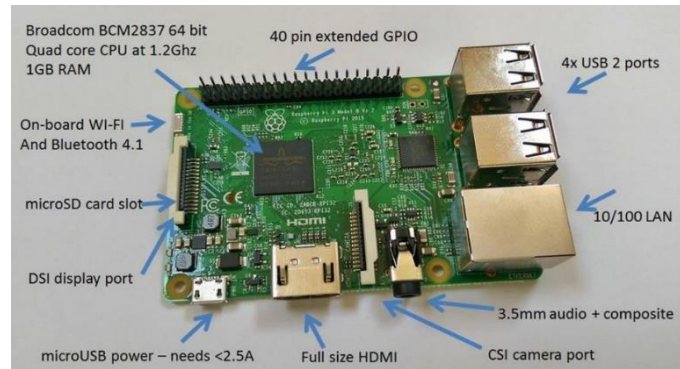
Raspberry Pi (Rpi) là máy tính kích cỡ nhỏ và chạy hệ điều hành Linux. Được phát triển bởi Raspberry Pi Foundation. Raspberry Pi sản xuất bởi 3 OEM: Sony, Qasida, Egoman. Và được phân phối chính bởi Element14, RS Components và Egoman.

Nhiệm vụ ban đầu của dự án Rpi là tạo ra máy tính rẻ tiền có khả năng lập trình cho những sinh viên, nhưng Pi đã được sự quan tâm từ nhiều đối tượng khác nhau. Đặc tính của Rpi xây dựng xoay quanh bộ xử lý SoC Broadcom BCM2835( là chip xử lý mobile mạnh mẽ có kích thước nhỏ hay được dùng trong điện thoại di động ) bao gồm CPU, GPU, bộ xử lý âm thanh/video, và các tính năng khác.v.v. tất cả được tích hợp bên trong chip có điện năng thấp này.

Rpi không thay thế hoàn toàn hệ thống để bàn hoặc máy xách tay. Không thể chạy Windows trên đó vì BCM2835 dựa trên cấu trúc ARM nên không hỗ trợ mã x86/x64, nhưng vẫn có thể chạy bằng Linux với các tiện ích như lướt web, môi

trường Desktop và các nhiệm vụ khác. Rpi là một thiết bị đa năng đáng ngạc nhiên với nhiều phần cứng có giá thành rẻ nhưng rất hoàn hảo cho những hệ thống điện tử, những dự án DIY, thiết lập hệ thống tính toán rẻ tiền cho những bài học trải nghiệm lập trình.

#### 1.1.1.2. Cấu trúc phần cứng của Rpi



Hình 1- 1 Các kết nối của raspberry pi 3-Model B

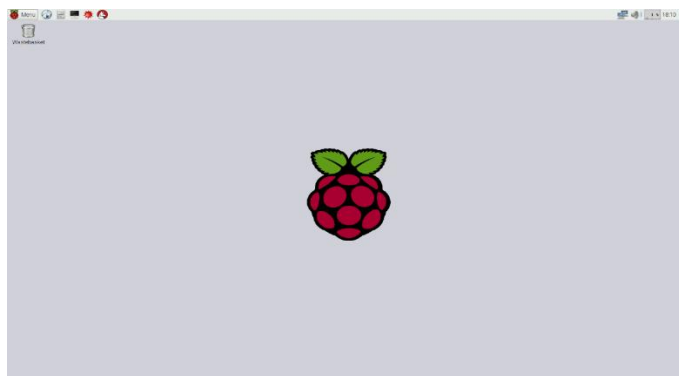
Rpi 3 Model B (Hình 1-1) đi kèm với bộ xử lý lõi tứ 64 bit, trên một board mạch với các tính năng WiFi và Bluetooth và USB.

- Nó có tốc độ xử lý từ 700 MHz đến 1,4 GHz trong đó bộ nhớ RAM dao động từ 256 đến 1GB.
- CPU của thiết bị này được coi là bộ não của thiết bị chịu trách nhiệm thực thi các câu lệnh dựa trên hoạt động toán học và logic.
- GPU (bộ xử lý đồ họa) là một chip tiên tiến khác được tích hợp trong board mạch có chức năng tính toán hình ảnh. Board mạch được trang bị cáp lõi video Broadcam chủ yếu được sử dụng để chơi các trò chơi video thông qua thiết bị.
- Rpi 3 đi kèm với các chân GPIO (General Purpose Input Output) rất cần thiết để duy trì kết nối với các thiết bị điện tử khác. Các chân đầu ra đầu vào này nhận lệnh và hoạt động dựa trên chương trình của thiết bị.
- Cổng Ethernet được tích hợp trên thiết bị này để thiết lập một đường giao tiếp với các thiết bị khác. Ta có thể kết nối cổng Ethernet với bộ định tuyến để duy trì kết nối cho internet.
- Board có bốn cổng USB được sử dụng có thể sử dụng để kết nối với bàn phím, chuột hoặc có thể kết nối USB 3G để truy cập internet và thẻ SD được thêm vào để lưu trữ hệ điều hành.

- Đầu nối nguồn điện là một phần cơ bản của board mạch được sử dụng để cung cấp nguồn 5V cho bo mạch. Ta có thể sử dụng bất kỳ nguồn nào để thiết lập nguồn cho board mạch, tuy nhiên, ta ưu tiên kết nối cấp nguồn qua cổng USB của máy tính xách tay để cung cấp 5V.
- Pi 3 hỗ trợ hai tùy chọn kết nối bao gồm HDMI và RCA Video. Cổng HDMI được sử dụng để kết nối LCD hoặc TiVi, có thể hỗ trợ cấp phiên bản 1.3 và 1.4
- Cổng USB được tích hợp trên board mạch được sử dụng để khởi động thiết bị. Vì RPi chạy hệ điều hành Linux, nên chỉ cần cắm bàn phím và chuột vào là có thể sử dụng mà không cần cài thêm thiết bị.

#### 1.1.1.3. Hệ điều hành Raspbian cho Rpi

Raspbian là một hệ điều hành máy tính dựa trên Debian dành cho Raspberry Pi. Nó được chính thức cung cấp bởi Raspberry Pi Foundation, là hệ điều hành chính cho các board nhúng Raspberry Pi. Raspbian được tạo ra bởi Mike Thompson và Peter Green như là một dự án độc lập. Việc xây dựng ban đầu đã được hoàn thành vào tháng 6 năm 2012. Hệ điều hành vẫn đang được phát triển tích cực. Raspbian được tối ưu hóa cao đối với các CPU ARM hiệu suất thấp của dòng Raspberry Pi. Raspbian sử dụng PIXEL, Pi Improved Xwindows Environment, Lightweight như môi trường máy tính để bàn. Raspbian là một hệ điều hành máy tính dựa trên Debian dành cho Raspberry Pi (Hình 1-2)



Hình 1- 2. Giao diện hệ điều hành raspbian

Mặc dù Raspbian chủ yếu là những nỗ lực của Mike Thompson và Peter Green, nhưng nó cũng được hưởng lợi từ sự ủng hộ nhiệt tình của các thành viên cộng đồng Raspberry Pi, những người muốn đạt được hiệu quả tối đa từ thiết bị của họ



#### *1.1.1.4. Thông số kỹ thuật chính*

- CPU 1,4 GHz 64 bit, Bộ xử lý lõi tứ Broadcom BCM2387 ARM Cortex-A53, nhanh hơn 10 lần so với Raspberry Pi 1.
- RAM 1GB (LPDDR2 SDRAM) cho phép ta chạy các ứng dụng nâng cao
- 802.11 b/g/n Wireless LAN
- On-board Bluetooth 4.1
- 4 cổng USB 2.0
- Ethernet 300Mbit/s
- 40 chân GPIO
- HDMI hỗ trợ phiên bản 1.3/1.4 và Composite RCA (PAL and NTSC)
- 10/100 BaseT Ethernet socket
- Camera interface (CSI), để kết nối với camera
- Display interface (DSI): được sử dụng để kết nối Raspberry Pi với màn hình cảm ứng
- Khe cắm thẻ microSD: để lưu trữ dữ liệu
- Micro USB power source
- VideoCore IV multimedia/3D graphics core @ 400MHz/300MHz

#### *1.1.1.5. So sánh Rpi với máy tính*

Rpi 3 giống hệt với máy tính thông thường về mặt thực hiện nhiều chức năng có giá trị. Nhưng nó có ưu điểm hơn so với máy tính để bàn khi nói đến chi phí và tiếp cận những nơi khó tiếp cận. Chi tiêu số tiền lớn để mua một máy tính cao cấp, khi ta đang trong giai đoạn bắt đầu học các kỹ năng máy tính, sẽ không phải là lựa chọn tốt. Tốt hơn là đầu tư vào một thiết bị nhỏ như Rpi 3, để ta có thể có cái nhìn tổng quát về những gì mà một máy tính thực sự có thể làm.

Thiết bị này có thể được sửa đổi thành bất kỳ mô-đun nào như robot hoặc máy tính xách tay, dựa trên nhu cầu và yêu cầu cụ thể.

Nếu ta bị ám ảnh bởi các trò chơi, thì thiết bị này có thể trở thành một máy chơi game rất hữu ích, vì nó có thể chạy các trò chơi đòi hỏi cấu hình thấp, cho phép ta thoát khỏi việc chi tiêu một số tiền lớn để mua máy tính xách tay.

Không còn nghi ngờ gì nữa, Raspberry Pi 3 có rất nhiều chức năng giống hệt PC thông thường, nhưng nó không nhanh bằng máy tính để bàn khi tải lên hoặc xử lý các trang web nặng.

Các game thủ có thể thấy thiết bị này gây nghiện, bởi vì nó có thể có khả năng hỗ trợ một số trò chơi cổ điển, tuy nhiên, chơi các trò chơi nâng cao hơn sẽ hơi khó hơn so với chơi trên máy tính thông thường.

#### *1.1.1.6. Lý do nên dùng Rpi*

- Giá rẻ: giá chỉ từ 5 USD cho một cái Pi cơ bản (phiên bản rút gọn Rpi Zero). Bản Raspberry Pi 3 là mạnh nhất có đầy đủ chức năng.
- Đơn giản, tiết kiệm không gian: dùng cho các công việc văn phòng đơn giản, gõ Word, Excel, PowerPoint, lướt web.
- Tự học lập trình bằng các app đơn giản của Raspberry Pi
- Tiêu thụ điện ít
- Có tính di động cao: có thể bỏ vào túi mang đi, thích hợp làm máy nghe nhạc di động, máy đọc ebook, máy dò pass Wi-Fi, máy chơi game cầm tay.

#### **1.1.2. Các thiết bị phần cứng khác**

Webcam là thiết bị không thể thiếu, vì nó có chức năng thu thập khung ảnh. Wemos và Servo G90 là phần cứng đảm nhiệm chức năng đóng mở cửa khi nhận được tín hiệu từ Raspberry gửi đến.

##### *1.1.2.1. Webcam See3cam*

See3CAM\_CU30(Hình 1- 3) là bảng camera USB Low Light 3,4 MP tuân thủ UVC dựa trên cảm biến AR0330 từ onsemi®. Camera low light board này tương thích ngược với USB 2.0 và hỗ trợ các định dạng MJPEG nén ở tốc độ khung hình bằng USB 3.1 Gen 1. Cảm biến này cho phép hiệu suất ánh sáng yếu vượt trội. Nó có chip xử lý tín hiệu hình ảnh (ISP) chuyên dụng, hiệu suất cao thực hiện tất cả các chức năng Tự động (Cân bằng trắng tự động, điều khiển phơi sáng tự động) ngoài

đường ống xử lý tín hiệu hình ảnh hoàn chỉnh cung cấp hình ảnh, video và nén MJPEG tốt nhất. See3CAM\_CU30(Hình 1- 3) là bảng camera USB Low Light 3,4 MP tuân thủ UVC dựa trên cảm biến AR0330 từ onsemi®. Camera low light board này tương thích ngược với USB 2.0 và hỗ trợ các định dạng MJPEG nén ở tốc độ khung hình bằng USB 3.1 Gen 1. Cảm biến này cho phép hiệu suất ánh sáng yếu vượt trội. Nó có chip xử lý tín hiệu hình ảnh (ISP) chuyên dụng, hiệu suất cao thực hiện tất cả các chức năng Tự động (Cân bằng trắng tự động, điều khiển phơi sáng tự động) ngoài đường ống xử lý tín hiệu hình ảnh hoàn chỉnh cung cấp hình ảnh, video và nén MJPEG tốt nhất.



Hình 1- 3. Webcam See3cam

#### 1.1.2.2. Wemos

WEMOS D1 R2 (Dùng thay thế cho ESP8266) là kit phát triển phiên bản mới nhất từ WeMos, kit được thiết kế với hình dáng tương tự Arduino Uno nhưng trung tâm lại là module wifi Soc ESP8266EX được xây dựng lại firmware để có thể chạy với chương trình Arduino. Kit thích hợp và dễ dàng thực hiện các ứng dụng thu thập dữ liệu và điều khiển qua Wifi. Dưới đây là bảng thông số của Wemos

Bảng 1- 1. Bảng thông số của Wemos

<b>Vi điều khiển</b>	<b>ESP8266EX</b>
Điện áp hoạt động	3V3
I/O Digital Pin	11
Analog Pin	1 (Max input=3V2)
Xung clock	80MHz/160MHz
Flash	4Mb
Khối lượng	25g
Kích thước	68.6mmX53.4mm

### 1.1.2.3. Servo G90

Động cơ RC Servo 9G là động phổ biến dùng trong các mô hình điều khiển nhỏ và đơn giản như cánh tay robot. Động cơ có tốc độ phản ứng nhanh, được tích hợp sẵn Driver điều khiển động cơ, dễ dàng điều khiển góc quay bằng phương pháp điều độ rộng xung PWM.

Đặc điểm:

- Kích thước: 23mmX12.2mmX29mm
- Trọng lượng: 9 grams
- Điện áp hoạt động: 4.2-6V
- Nhiệt độ: 0 °C --55 °C
- Tốc độ: 0.3 giây / 60 độ

## 1.2. Tổng quan về giao thức MQTT

Hệ thống đặt ra có giao tiếp giữa nhiều thiết bị với nhau nên cần có 1 giao thức truyền thông để làm việc này. Trong đồ án này giới thiệu về giao thức MQTT. Câu hỏi đặt ra:” Có rất nhiều giao thức truyền thông dựa trên TCP/IP như HTTP, Socket, v.v.”. Câu trả lời:” MQTT có lợi thế hơn so với các giao thức kia về tính bảo mật truyền dữ liệu. Theo mặc định, nó sử dụng SSL/TLS như một đường truyền tin nhắn khi mã hóa trọng tải (payload) [13]. Và MQTT dễ sử dụng hơn vì nó hoạt động theo cách tiếp cận đơn giản là lệnh và kết quả. Khi sử dụng lệnh thoại dựa trên MQTT để điều khiển thiết bị gia đình, bạn không cần quan tâm đến việc thiết lập kết nối [13].”

### 1.2.1. Giới thiệu chung về giao thức MQTT

MQTT (Message Queueing Telemetry Transport) là một giao thức mạng kích thước nhỏ (lightweight), hoạt động theo cơ chế publish – subscribe (tạm dịch: xuất bản – đăng ký) theo tiêu chuẩn ISO (ISO/IEC 20922) và OASIS mở để truyền tin nhắn giữa các thiết bị.

Giao thức này hoạt động trên nền tảng TCP/IP. MQTT được thiết kế cho các kết nối cho việc truyền tải dữ liệu cho các thiết bị ở xa, các thiết bị hay vì điều khiển nhỏ có tài nguyên hạn chế hoặc trong các ứng dụng có băng thông mạng bị hạn chế.

MQTT là lựa chọn lý tưởng trong các môi trường như:

- Những nơi mà giá mạng viễn thông đắt đỏ hoặc băng thông thấp hay thiếu tin cậy.
- Khi chạy trên thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ.
- Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M (Machine to Machine).

MQTT cũng là giao thức sử dụng trong Facebook và Amazon IoT.

### **Tính năng, đặc điểm nổi bật**

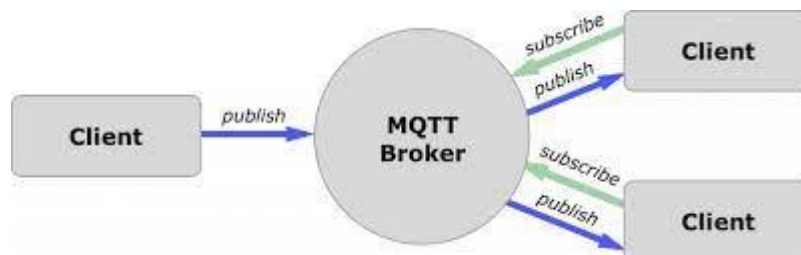
- Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.
- Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.
- Sử dụng TCP/IP là giao thức nền.
- Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of service)
  - QoS 0: Broker/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP.
  - QoS 1: Broker/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
  - QoS 2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua 4 bước bắt tay.
- Phân bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.

### **Ưu điểm của giao thức MQTT**

- Truyền thông tin hiệu quả hơn.
- Tăng khả năng mở rộng.
- Giảm đáng kể tiêu thụ băng thông mạng.
- Rất phù hợp cho điều khiển và do thám.
- Tối đa hóa băng thông có sẵn.
- Chi phí thấp.
- Rất an toàn, bảo mật.
- Được sử dụng trong các ngành công nghiệp dầu khí, các công ty lớn như Amazon, Facebook, ....
- Tiết kiệm thời gian phát triển.

- Giao thức publish/subscribe thu thập nhiều dữ liệu hơn và tốn ít băng thông hơn so với giao thức cũ như HTTP.

### 1.2.2. Mô hình Publish/Subscriber trong giao thức MQTT



Hình 1- 4. Mô hình Publish/Subscriber trong giao thức mqtt

Mô hình Publish/Subscriber trong giao thức MQTT bao gồm các thành phần: MQTT Broker, MQTT Client(Hình 1-4). Các thành phần này lần lượt được trình bày ở các mục ngay sau đây

#### 1.2.2.1. MQTT Broker

MQTT Broker hay máy chủ môi giới được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber).

Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Nhiệm vụ phụ của Broker là nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, v.v.

MQTT Broker được cung cấp dưới dạng mã nguồn mở hoặc các phiên bản thương mại giúp người dùng có thể tự cài đặt và tạo broker riêng. Ngoài ra cũng có thể sử dụng Broker trên điện toán đám mây với các nền tảng IOT như Hive Broker, Amazon, v.v.

#### 1.2.2.2. MQTT Client

Là các thiết bị/ứng dụng Client kết nối đến Broker để thực hiện truyền nhận dữ liệu. Client thì được chia thành hai nhóm là Publisher và Subscriber. Một Client có thể có một trong hai nhiệm vụ hoặc cả hai.

Publisher là thiết bị gửi bản tin lên broker

Subscriber là người nhận bản tin mỗi khi có bản tin mới gửi lên Broker.

### 1.2.2.3. Tính chất của mô hình pub/sub

#### **Tính chất:**

- Space decoupling (Không gian tách biệt)
- Time decoupling (Thời gian tách biệt)
- Synchronization decoupling (Sự đồng bộ riêng rẽ)

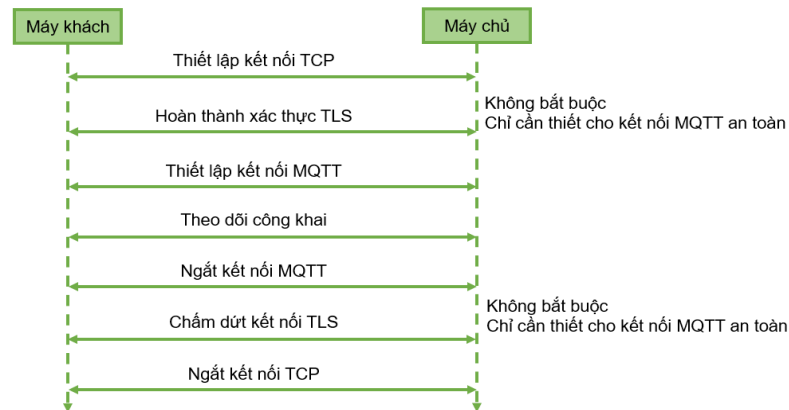
#### **Đặc điểm riêng:**

- MQTT sử dụng cơ chế lọc thông điệp dựa vào tiêu đề (subject-based)
- MQTT có một tầng gọi là chất lượng dịch vụ (Quality of Services – QoS). Nó giúp cho dễ dàng nhận biết được là message có được truyền thành công hay không.

### 1.2.3. Cơ chế hoạt động của giao thức MQTT

Một phiên MQTT được chia thành bốn giai đoạn: kết nối, xác thực, giao tiếp và kết thúc (Hình 1- 5)

1. máy khách bắt đầu bằng cách tạo kết nối Transmission Control Protocol/Internet Protocol (TCP/IP) tới máy chủ bằng cách sử dụng cổng tiêu chuẩn hoặc cổng tùy chỉnh được xác định bởi các nhà phát triển broker.
2. Các cổng tiêu chuẩn là 1883 cho giao tiếp không mã hóa và 8883 cho giao tiếp được mã hóa – sử dụng Lớp cổng bảo mật (SSL) / Bảo mật lớp truyền tải (TLS). Trong quá trình giao tiếp SSL/TLS, máy khách cần kiểm chứng và xác thực máy chủ.
3. Sau đó, máy khách sẽ gửi bản tin lên máy chủ nếu là Publisher hoặc nhận bản tin từ máy chủ về nếu là Subscriber. Quá trình kết nối này sẽ được giữ đến khi Kết thúc kết nối.
4. Sau khi kết thúc để có thể truyền nhận MQTT, lại tiếp tục quay lại các bước trên.



Hình 1- 5. Cơ chế hoạt động của giao thức mqtt

#### 1.2.4. Các khái niệm trong MQTT

**Message:** Trong giao thức MQTT, message còn được gọi là “message payload”, có định dạng mặc định là plain-text (chữ viết người đọc được), tuy nhiên người sử dụng có thể cấu hình thành các định dạng khác.

#### Topic

- Topic có thể coi như một “đường truyền” logic giữa 2 điểm là publisher và subscriber. Về cơ bản, khi message được publish vào một topic thì tất cả những subscriber của topic đó sẽ nhận được message này.
- Giao thức MQTT cho phép khai báo các topic kiểu phân cấp.

**QoS:** Ở đây có 3 tùy chọn QoS (Qualities of service) khi “publish” và “subscribe”:

- QoS0: Broker/client sẽ gửi dữ liệu đúng 1 lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP, giống kiểu đem con bỏ chợ.
- QoS1: Broker/client sẽ gửi dữ liệu với ít nhất 1 lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
- QoS2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng 1 lần, quá trình này phải trải qua 4 bước bắt tay.

Một gói tin có thể được gửi ở bất kỳ QoS nào, và các client cũng có thể subscribe với bất kỳ yêu cầu QoS nào. Có nghĩa là client sẽ lựa chọn QoS tối đa mà nó có để nhận tin. Ví dụ, nếu 1 gói dữ liệu được publish với QoS2, và client subscribe với QoS0, thì gói dữ liệu được nhận về client này sẽ được broker gửi với QoS0, và 1 client khác đăng ký cùng kênh này với QoS 2, thì nó sẽ được Broker gửi dữ liệu với QoS2.



Một ví dụ khác, nếu 1 client subscribe với QoS2 và gói dữ liệu gửi vào kênh đó publish với QoS0 thì client đó sẽ được Broker gửi dữ liệu với QoS0. QoS càng cao thì càng đáng tin cậy, đồng thời độ trễ và băng thông đòi hỏi cũng cao hơn.

### 1.3. Tổng quan về SSL/TLS

Vì dùng giao thức MQTT làm giao thức truyền thông, giao thức này có tính bảo mật nhờ có bảo mật SSL/TLS nhằm chức năng xác thực các thiết bị với Broker MQTT trong hệ thống

#### 1.3.1. Khái niệm

**TLS** (tiếng Anh: *Transport Layer Security*: "Bảo mật tầng giao vận") trước đây là **SSL** (*Secure Sockets Layer*: "Tầng socket bảo mật") là giao thức mật mã được thiết kế để cung cấp truyền thông an toàn qua một mạng máy tính. Một số phiên bản của các giao thức này được sử dụng rộng rãi trong các ứng dụng như trình duyệt Web, thư điện tử, tin nhắn nhanh, và VoIP.

Các giao thức này mật mã hóa khóa bất đối xứng bằng các chứng thực X.509 để xác thực bên kia và để trao đổi một khóa đối xứng. Sau đó, khóa phiên được dùng để mã hóa các dữ liệu được truyền qua lại hai bên. Phương pháp này cho phép bảo mật dữ liệu hoặc thông điệp và xác thực tính toàn vẹn của các thông điệp qua các mã xác thực thông điệp (*message authentication code*). Do sử dụng các chứng thực X.509, giao thức này cần các nhà cung cấp chứng thực số và hạ tầng khóa công khai để xác nhận mối quan hệ giữa một chứng thực và chủ của nó, cũng như để tạo, ký, và quản lý sự hiện lực của các chứng thực. Tuy quá trình này có thể tốt hơn việc xác nhận các danh tính qua một mạng lưới tín nhiệm, những vụ tai tiếng do thám bí mật người dân 2013 đã báo động công cộng rằng các nhà cung cấp chứng thực là một điểm yếu về bảo mật vì cho phép các tấn công xen giữa (*man-in-the-middle attack*).

Trong khung nhìn mô hình TCP/IP, TLS và SSL đều mã hóa dữ liệu của các kết nối mạng trên một tầng phụ thấp của tầng ứng dụng. Theo hệ thống tầng cấp của mô hình OSI, TLS/SSL được khởi chạy ở tầng năm (tầng phiên) rồi hoạt động trên tầng sáu (tầng trình diễn): trước tiên tầng phiên bắt tay dùng mật mã bất đối xứng để đặt cấu hình mật mã và khóa chia sẻ dành cho phiên đó; sau đó, tầng trình diễn mã hóa phần còn lại của thông điệp dùng mật mã đối xứng và khóa của phiên đó. Trong cả hai mô hình, TLS và SSL phục vụ tầng giao vận bên dưới, các đoạn trong tầng này chứa dữ liệu mật mã hóa.

### 1.3.2. Hoạt động của SSL/TLS

Khi máy khách và máy chủ giao tiếp, SSL đảm bảo rằng kết nối là riêng tư và an toàn bằng cách cung cấp xác thực, mã hóa và kiểm tra tính toàn vẹn. Xác thực xác nhận rằng máy chủ, và tùy chọn là máy khách, là người mà họ nói. Mã hóa bằng cách sử dụng khóa phiên duy nhất được thương lượng an toàn giữa máy khách và máy chủ tạo ra một “luồng” an toàn giữa cả hai để ngăn chặn bất kỳ hệ thống trái phép nào đọc dữ liệu. Kiểm tra tính toàn vẹn đảm bảo rằng bất kỳ hệ thống trái phép nào cũng không thể sửa đổi luồng được mã hóa mà không bị phát hiện.

Các thiết bị hỗ trợ SSL - chẳng hạn như máy khách sử dụng trình duyệt web như Mozilla TM, Safari TM hoặc ChromeTM - và máy chủ hỗ trợ SSL (chẳng hạn như Apache hoặc Microsoft IISTM) xác nhận danh tính của nhau bằng chứng chỉ kỹ thuật số. Chứng chỉ kỹ thuật số được cấp bởi các bên thứ ba đáng tin cậy được gọi là Tổ chức phát hành chứng chỉ (CA) và cung cấp thông tin về danh tính được xác nhận quyền sở hữu của một cá nhân cũng như khóa công khai của họ. Khóa công khai là một thành phần của hệ thống mật mã khóa công khai. Người gửi tin nhắn sử dụng khóa công khai để mã hóa dữ liệu. Người nhận tin nhắn chỉ có thể giải mã dữ liệu bằng khóa riêng tương ứng. Khóa công khai được mọi người biết đến; khóa riêng là bí mật và chỉ chủ sở hữu chứng chỉ mới biết. Bằng cách xác thực chữ ký số CA trên chứng chỉ, cả hai bên có thể đảm bảo rằng kẻ giả mạo đã không chặn việc truyền và cung cấp khóa công khai sai mà họ có khóa riêng chính xác. SSL sử dụng cả mã hóa khóa công khai và khóa đối xứng. Mã hóa khóa đối xứng nhanh hơn nhiều so với mã hóa khóa công khai, nhưng mã hóa khóa công khai cung cấp các kỹ thuật xác thực tốt hơn. Vì vậy, SSL sử dụng mật mã khóa công khai để xác thực và trao đổi các khóa đối xứng được sử dụng sau này để mã hóa dữ liệu hàng loạt.

Đường hầm bảo mật mà SSL tạo ra là một kết nối được mã hóa đảm bảo rằng tất cả thông tin được gửi giữa máy khách hỗ trợ SSL và máy chủ hỗ trợ SSL vẫn ở chế độ riêng tư. SSL cũng cung cấp một cơ chế để phát hiện nếu ai đó đã thay đổi dữ liệu trong quá trình truyền tải. Điều này được thực hiện với sự trợ giúp của kiểm tra tính toàn vẹn của thư. Các kiểm tra tính toàn vẹn của thư này đảm bảo rằng kết nối là đáng tin cậy. Nếu, tại bất kỳ thời điểm nào trong quá trình truyền, SSL phát hiện ra rằng một kết nối không an toàn, nó sẽ ngắt kết nối và máy khách và máy chủ thiết lập một kết nối an toàn mới.

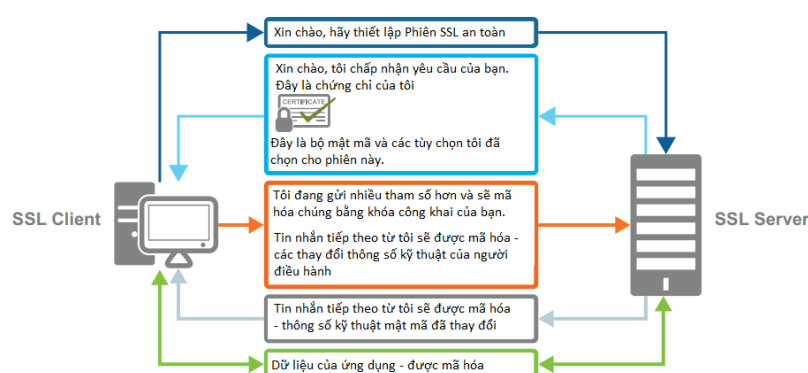
### 1.3.3. Các giao dịch SSL/TLS

Giao dịch SSL có hai giai đoạn: Bắt tay SSL (trao đổi khóa) và truyền dữ liệu SSL. Các giai đoạn này làm việc cùng nhau để bảo mật một giao dịch SSL.

Mặc dù quá trình xác thực và mã hóa có vẻ khá liên quan, nhưng nó diễn ra trong vòng chưa đầy một giây. Nói chung, người dùng thậm chí không biết nó đang diễn ra. Tuy nhiên, người dùng có thể biết khi nào đường hầm bảo mật đã được thiết lập vì hầu hết các trình duyệt web hỗ trợ SSL đều hiển thị một ổ khóa đóng nhỏ ở cuối (hoặc trên cùng) của màn hình khi kết nối được bảo mật.

Người dùng cũng có thể xác định các trang web an toàn bằng cách xem địa chỉ trang web; địa chỉ của trang web an toàn bắt đầu bằng HTTPS thay vì HTTP thông thường.

Dưới đây là minh họa trình tự của các giao dịch SSL (Hình 1- 6)



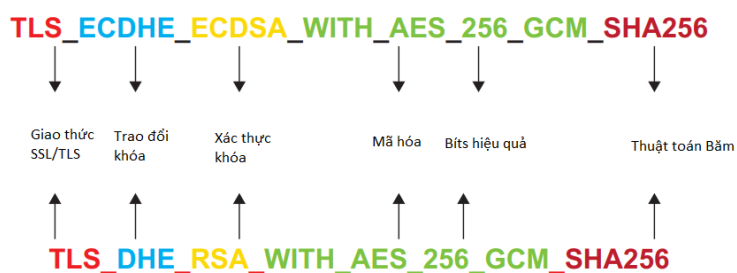
Hình 1- 6. Trình tự các giao dịch của ssl/tls

### 1.3.4. Bộ mã hóa và SSL/TLS

SSL hỗ trợ nhiều thuật toán mật mã hoặc mật mã khác nhau mà nó sử dụng để xác thực, truyền chứng chỉ và thiết lập khóa phiên. Các thiết bị hỗ trợ SSL có thể được định cấu hình để hỗ trợ các bộ mật mã khác nhau, được gọi là bộ mật mã. Nếu máy khách hỗ trợ SSL và máy chủ hỗ trợ SSL hỗ trợ nhiều bộ mật mã, máy khách và máy chủ sẽ thương lượng bộ mật mã nào họ sử dụng để cung cấp bảo mật mạnh nhất có thể được cả hai bên hỗ trợ. Một bộ mật mã chỉ định và kiểm soát các thuật toán mật mã khác nhau được sử dụng trong quá trình bắt tay SSL và các giai đoạn truyền dữ liệu. Cụ thể, một bộ mật mã cung cấp những điều sau:

- Thuật toán trao đổi khóa (**Key exchange algorithm**): Thuật toán khóa bất đối xứng được sử dụng để trao đổi khóa đối xứng. RSA, Diffie-Hellman (DHE) và Elliptic Curve Diffie-Hellman (ECDHE) là những ví dụ phổ biến.
- Thuật toán khóa công khai (**Public key algorithm**): Thuật toán khóa bất đối xứng được sử dụng để xác thực. Điều này quyết định loại chứng chỉ được sử dụng. RSA và DSA là những ví dụ phổ biến.
- Thuật toán mã hóa hàng loạt (**Bulk encryption algorithm**): Thuật toán đối xứng được sử dụng để mã hóa dữ liệu. RC4, AES, DES, 3DES và Camellia là những ví dụ phổ biến.
- Thuật toán thông báo thông điệp (**Message digest algorithm**): Thuật toán được sử dụng để thực hiện kiểm tra và xác thực tính toàn vẹn của thông điệp. SHA và SHA256 là những ví dụ phổ biến.

Các bộ mật mã được minh họa dưới đây (Hình 1- 7) cung cấp 2 ví dụ về các bộ mật mã hiện đại được sử dụng trong SSL / TLS.



Hình 1- 7. Minh họa bộ mật mã hiện đại được sử dụng trong SSL/TLS

#### 1.4. Mã hóa AES

Tuy đã có bảo mật SSL/TLS, nhưng vẫn muốn hệ thống giao tiếp an toàn hơn nên trong đề án có thêm phần mã hóa dữ liệu AES. Lý do chọn AES là vì AES có kích thước lớn nhất là 256 bits ( $2^{256}$  kết hợp duy nhất [20]), nó mạnh đến mức chống lại các cuộc tấn công từ siêu máy tính [20]. Vì vậy hệ thống dùng mã hóa AES 256 và ở chế độ mã hóa chuỗi (CBC). Lý do chọn chế độ CBC là vì [19]:

- Khả năng bảo mật cao hơn ECB. Ciphertext của một khối dữ liệu plaintext có thể khác nhau cho mỗi lần mã hóa vì nó phụ thuộc vào IV hoặc giá trị mã hóa (ciphertext) của khối dữ liệu liền trước.

- Quá trình giải mã (mã hóa nghịch) vẫn có thể thực hiện song song nhiều khối dữ liệu.

#### **1.4.1. Khái niệm**

AES là viết tắt của Advanced Encryption Standard, chuẩn mã hóa dữ liệu rất phổ biến, dùng cho nhiều mục đích và được cả chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật. AES là kiểu mã hóa đối xứng dạng khối, nghĩa là mỗi khối văn bản có một kích thước nhất định (128 bit) được mã hóa, khác với mã hóa dạng chuỗi khi từng kí tự được mã hóa. Đối xứng nghĩa là khóa để mã hóa và giải mã đều là một.

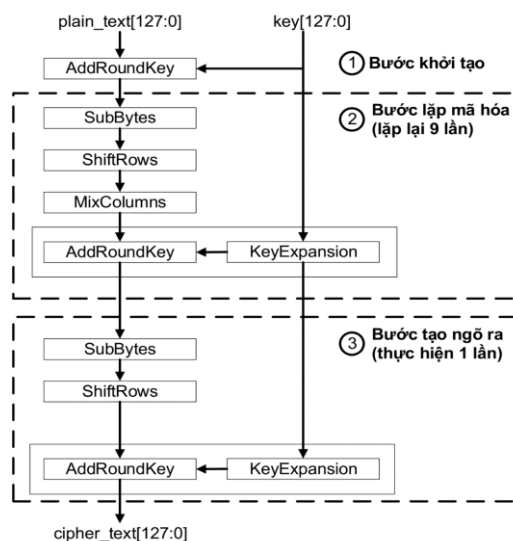
#### **1.4.2. Hoạt động của AES**

AES là kiểu mã hóa khối, mỗi khối kích thước 128 bit. Khóa đối xứng với 3 kích thước là 128, 192 và 256 bit, trong đó 2 kích thước sau được chính phủ Mỹ dùng cho các tài liệu mật cấp cao, được gọi là “Top Secret”.

AES dùng thuật toán mã hóa khối mạng thay thế hoán đổi (SPN - Substitution Permutation Network). Dữ liệu được chuyển thành dạng an toàn trong vài bước, bắt đầu là khối plain text kích thước chuẩn, sau đó chèn vào hàng và sau đó là mã hóa. Mỗi lần đều có các bước thay thế, chuyển đổi, hòa trộn.

Cũng như 3DES có 3 bước mã hóa, AES cũng có nhiều bước nhưng được thực hiện nhiều hơn, phụ thuộc vào độ dài khóa, với khóa 128-bit là 10 lần, khóa 192-bit là 12 lần và khóa 256 bit là 14 lần

### 1.4.3. Thuật toán AES



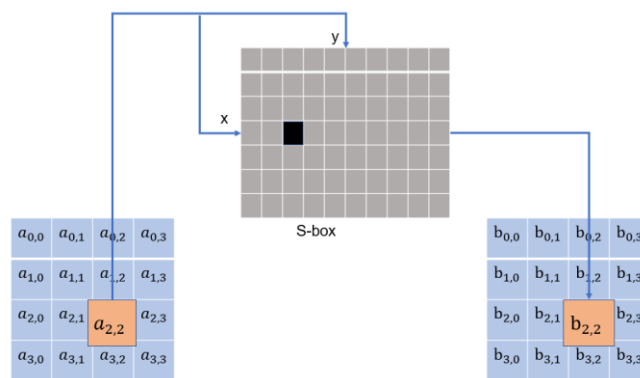
Hình 1- 8. Thuật toán mã hóa AES

Thuật toán AES (Hình 1- 8) được mô tả khái quát gồm 3 bước:

- 1 vòng khởi tạo AddRoundKey.
- $Nr-1$  vòng lặp gồm 4 phép biến đổi: SubBytes, ShiftRows, MixColumns, AddRoundKey.
- 1 vòng cuối gồm các phép biến đổi giống vòng lặp và không có MixColumns.

Phương pháp mã hóa AES đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8-bit (dùng trong smartcard) cũng như trên các vi xử lý 32 bit, chỉ dùng phép XOR và phép Shift bit. Đây chính là yếu tố cơ bản để phương pháp này được chọn làm chuẩn mã hóa của Hoa Kỳ.

- Bước SubBytes



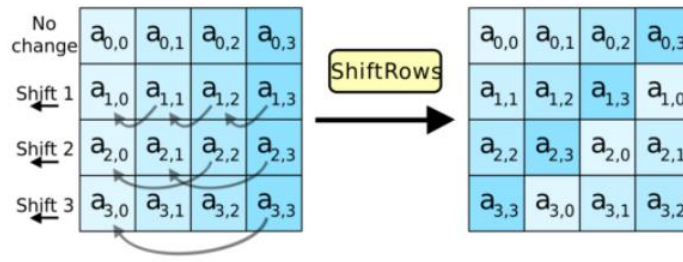
Hình 1- 9. Bước SubBytes trong mã hóa AES

Phép biến đổi SubBytes (Hình 1- 9): Là phép thay thế byte phi tuyến tính được thực hiện bằng cách tra cứu bảng thay thế (S-box thuận hoặc nghịch) với tham số đầu vào là các byte trong bảng trạng thái (Hình 1- 10).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Hình 1- 10. Bảng trạng thái của phép biến đổi SubBytes

- Bước ShiftRows

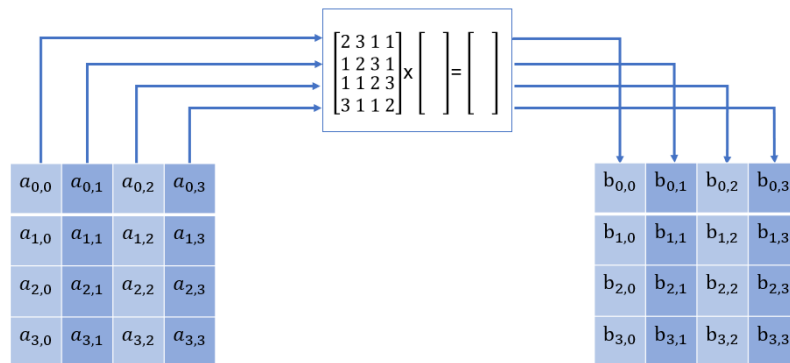


Hình 1- 11. Bước ShiftRows trong mã hóa AES

Các bước trong ShiftRows (Hình 1- 11) thực hiện hoán vị các byte trong ma trận state theo cách thức:

- Dòng thứ nhất giữ nguyên
- Dòng thứ 2 dịch vòng trái 1 byte
- Dòng thứ 3 dịch vòng trái 2 byte
- Dòng thứ 4 dịch vòng trái 3 byte

• Bước MixColumns

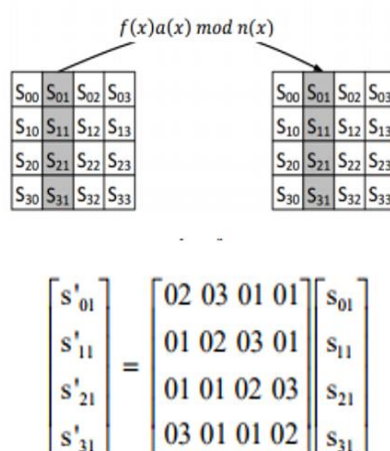


Hình 1- 12. Bước MixColumns trong mã hóa AES

Phép biến đổi MixColumns (Hình 1- 12) thực hiện biến đổi độc lập từng cột trong ma trận state bằng phép nhân đa thức. Mỗi cột của trạng thái được coi là biểu diễn của một đa thức  $f(x)$  trong  $GF(2^8)$  như vậy phép biến đổi MixColumns chính là phép nhân theo modulo với  $x^4+1$  với một đa thức cố định:  $3x^3+x^2+x+2$ . Hình 1- 13 dưới đây sẽ minh họa điều này.



$$c(x) = 3x^3 + x^2 + x + 2 \text{ (modulo } x^4 + 1)$$



Hình 1- 13. Biểu diễn của 1 đa thức  $f(x)$  trong  $GF(2^8)$

- Bước AddRoundKey

Trong bước AddRoundKey(Hình 1-14), 128 bit của ma trận trạng thái được XOR với 128 bit của khóa con của từng vòng.



Hình 1- 14. Bước AddRoundKey trong mã hóa AES

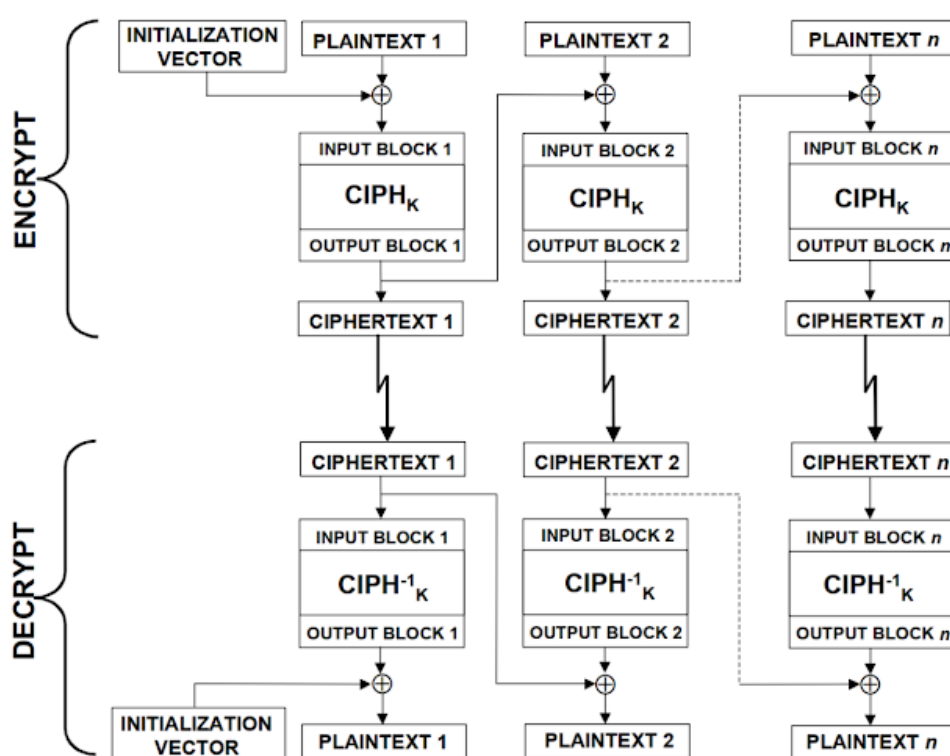
#### 1.4.4. Bảo mật của mã hóa AES

AES tuyệt đối an toàn khi nó được triển khai đúng cách. Tuy nhiên, các khóa mã hóa AES phải được bảo mật. Các hệ thống mã hóa bảo mật nhất cũng dễ dàng bị tấn công khi tin tặc có quyền truy cập vào khóa AES. Việc sử dụng mật khẩu mạnh, bảo mật mật khẩu, xác thực đa yếu tố (MFA), tường lửa và phần mềm chống vi-rút là rất quan trọng.

### 1.4.5. Mã hóa AES 256 CBC

**Mã hóa AES 256** có độ dài khóa 256 bits, hỗ trợ kích thước lớn nhất và thực tế không thể phá vỡ bằng cách tấn công brute-force (tấn công dò mật khẩu: kẻ tấn công sử dụng một công cụ mạnh mẽ, có khả năng thử nhiều username và password cùng lúc (từ dễ đến khó) cho tới khi đăng nhập thành công) dựa trên các công nghệ hiện tại. Điều này biến AES 256 trở thành tiêu chuẩn mã hóa mạnh nhất hiện nay

**CBC [19] (Cipher Block Chaining\_ chế độ xích liên kết khối mã)** là chế độ mã hóa chuỗi, kết quả mã hóa của khối dữ liệu trước (ciphertext) sẽ được tổng hợp với khối dữ liệu kế tiếp (plaintext) trước khi thực thi mã hóa. Hình 1- 15 thể hiện chi tiết chế độ này.



Hình 1- 15. Chế độ mã hóa/ giải mã CBC

Quá trình mã hóa CBC: biểu thức định nghĩa

$$C1 = CIPH_K(P_1 + IV)$$

$$C_j = CIPH_K(P_j + C_{j-1}) \text{ với } j = 2, 3, \dots, n$$

Lần mã hóa đầu tiên:

- Plaintext XOR với vector khởi tạo IV

- Kết quả bước trên là đầu vào cho việc thực thi thuật toán mã hóa với khóa mã K

Lần mã hóa sau lần đầu tiên:

- Plaintext XOR với ciphertext của lần mã hóa trước đó
- Kết quả bước trên là đầu vào cho việc thực thi thuật toán mã hóa với khóa K

Quá trình giải mã CBC : Biểu thức định nghĩa

$$P_1 = \text{CIPHINV}_k(C_1) + IV$$

$$P_j = \text{CIPH}_k(C_j) + C_{j-1} \text{ với } j = 2, 3, \dots, n$$

Lần giải mã đầu tiên:

- Ciphertext được thực thi quá trình giải mã với khóa mã K
- Kết quả bước trên được XOR với vector khởi tạo IV để tạo ra plaintext

Lần giải mã sau lần đầu tiên:

- Ciphertext được thực thi quá trình giải mã với khóa mã K
- Kết quả bước trên được XOR với ciphertext sử dụng trong lần giải mã trước để tạo ra plaintext

## 1.5. Tổng quan về Flutter, ngôn ngữ lập trình Dart

Phần này liên quan đến ứng dụng trên mobile, ứng dụng này có các chức năng như điều khiển đóng, mở cửa, hiển thị khuôn mặt lạ và đưa ra cảnh báo cho gia chủ. Câu hỏi đặt ra: "Có thể lập trình ứng dụng bằng android hoặc react native mà lại chọn Flutter". Câu trả lời ở trong nội dung phía dưới

### 1.5.1. Ngôn ngữ lập trình Dart

Dart là ngôn ngữ lập trình đa mục đích ban đầu được phát triển bởi Google và sau đó được Ecma (ECMA-408) phê chuẩn làm tiêu chuẩn. Nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động. Dart là một ngôn ngữ hướng đối tượng, được xác định theo lớp, với cơ chế garbage-collected, sử dụng cú pháp kiểu C để dịch mã tùy ý sang JavaScript. Nó hỗ trợ interface, mixin, abstract, generic, static typing và sound type. Dart là ngôn ngữ mã nguồn mở và miễn phí, được phát triển trên GitHub. Hiện nay Dart đã phát hành phiên bản 2.7.

## Tại sao chọn Dart?

Các nhà phát triển tại Google và các nơi khác sử dụng Dart để tạo các ứng dụng chất lượng cao, quan trọng cho iOS, Android và web. Với các tính năng nhắm đến sự phát triển phía khách hàng, Dart rất phù hợp cho cả ứng dụng di động và web. Dart giúp bạn tạo ra những trải nghiệm đẹp, chất lượng cao trên tất cả các màn hình, với:

- Một ngôn ngữ được tối ưu hóa cho client
- Framework mạnh mẽ
- Công cụ linh hoạt

### 1.5.2. Framework Flutter

Flutter là SDK của Google, giúp tạo các ứng dụng di động cho iOS và Android bằng cách sử dụng một cơ sở mã (gần như vậy). Nó là một người mới trong phát triển ứng dụng di động đa nền tảng và không giống như các frameworks khác ví dụ React Native, nó không sử dụng JavaScript làm Ngôn ngữ lập trình.

Flutter bao gồm Reactive framework và công nghệ hiển thị 2D (2D rendering engine) và các công cụ phát triển (development tool). Các thành phần này làm việc cùng nhau giúp thiết kế, xây dựng, test, debug ứng dụng.

#### 1.5.2.1. Tại sao lại là Flutter

Flutter của Google là một trong các phương pháp thay thế để phát triển ứng dụng Android, một framework dựa trên ngôn ngữ lập trình Dart.

Các ứng dụng được xây dựng với Flutter hầu như không thể phân biệt với những ứng dụng được xây dựng bằng cách sử dụng Android SDK, cả về giao diện và hiệu suất. Hơn nữa, với những tính chỉnh nhỏ, chúng có thể chạy trên thiết bị iOS.

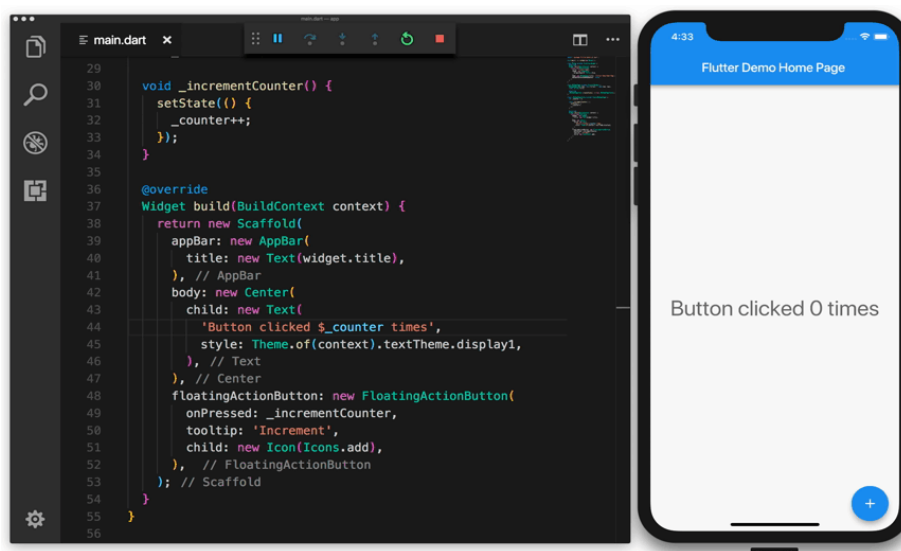
Chạy ở 60 fps, giao diện người dùng được tạo ra với Flutter thực thi tốt hơn nhiều so với những ứng dụng được tạo ra với các framework phát triển đa nền tảng khác chẳng hạn như React Native và Ionic. Một số lý do khiến bạn có thể hứng thú với Flutter:

- Flutter sử dụng Dart, một ngôn ngữ nhanh, hướng đối tượng với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.

- Flutter có các thành phần UI của riêng nó, cùng với một cơ chế để kết xuất chúng trên nền tảng Android và iOS. Hầu hết các thành phần giao diện người dùng, đều sẵn dùng, phù hợp với các nguyên tắc của Material Design.
- Các ứng dụng Flutter có thể được phát triển bằng cách sử dụng IntelliJ IDEA, một IDE rất giống với Android Studio.

#### 1.5.2.2. Đặc điểm nổi bật

**Fast Development:** (Hình 1- 16) Tính năng Hot Reload hoạt động trong milliseconds để hiển thị giao diện tới bạn. Sử dụng tập hợp các widget có thể customizable để xây dựng giao diện trong vài phút. Ngoài ra Hot Reload còn giúp bạn thêm các tính năng, fix bug tiết kiệm thời gian hơn mà không cần phải thông qua máy ảo, máy android hoặc IOS



Hình 1- 16. Đặc điểm Fast Development của Flutter

**Expressive and Flexible UI:** (Hình 1- 17) Có rất nhiều các thành phần để xây dựng giao diện của Flutter vô cùng đẹp mắt theo phong cách Material Design và Cupertino, hỗ trợ nhiều các APIs chuyển động, smooth scrolling v.v.



*Hình 1- 17. Đặc điểm Expressive and Flexible UI của Flutter*

**Native Performance**: Các widget của flutter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

## CHƯƠNG 2. THUẬT TOÁN HỌC MÁY VÀ NHẬN DẠNG KHUÔN MẶT

Sau chương 1 là Cơ sở lý thuyết thì ở chương này sẽ nêu ra tổng quan về thuật toán học máy và nhận dạng khuôn mặt làm cơ sở cho việc ứng dụng kỹ thuật này vào trong hệ thống chống trộm. Vì công nghệ phát triển, nhất là AI, và trong đề tài này thì bài toán được đưa ra là nhận dạng khuôn mặt, nên thuật toán học máy rất cần thiết. Lý do chọn ứng dụng của học máy là nhận dạng khuôn mặt trong đề án này là vì “Hệ thống này đặt ra bài toán là ai muốn vào được nhà thì cần phải nhận dạng khuôn mặt để phân biệt gia chủ hay người lạ.”. Nếu có thêm nhiều thời gian hơn thì hướng phát triển sẽ là nhận dạng cử chỉ con người để nhận biết người lạ, hướng này thì cần có bộ CPU có tốc độ xử lý cao hơn.

### 2.1. Thuật toán học máy

#### 2.1.1. Tổng quan

Học máy hay machine learning (ML) là một nhánh của trí tuệ nhân tạo (AI), nó là một lĩnh vực nghiên cứu cho phép máy tính có khả năng cải thiện chính bản thân chúng dựa trên dữ liệu mẫu (training data) hoặc dựa vào kinh nghiệm (những gì đã được học). Machine learning có thể tự dự đoán hoặc đưa ra quyết định mà không cần được lập trình cụ thể.

Phân loại: Có hai loại phương pháp học máy chính

- Phương pháp quy nạp: Máy học/phân biệt các khái niệm dựa trên dữ liệu đã thu thập được trước đó. Phương pháp này cho phép tận dụng được nguồn dữ liệu rất nhiều và sẵn có.
- Phương pháp suy diễn: Máy học/phân biệt các khái niệm dựa vào các luật. Phương pháp này cho phép tận dụng được các kiến thức chuyên ngành để hỗ trợ máy tính.

Hiện nay, các thuật toán đều cố gắng tận dụng được ưu điểm của hai phương pháp này.

Các ngành khoa học liên quan:

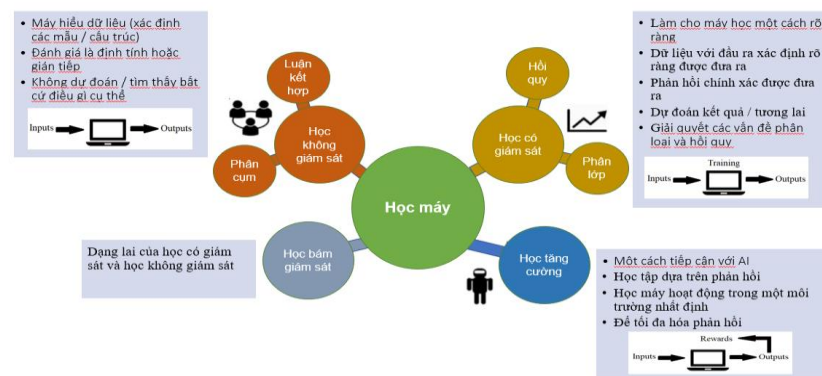
- Lý thuyết thống kê: các kết quả trong xác suất thống kê là tiền đề cho rất nhiều phương pháp học máy. Đặc biệt, lý thuyết thống kê cho phép ước lượng sai số của các phương pháp học máy.
- Các phương pháp tính: các thuật toán học máy thường sử dụng các tính toán số thực/số nguyên trên dữ liệu rất lớn. Trong đó, các bài toán như: tối ưu

có/không ràng buộc, giải phương trình tuyến tính v.v. được sử dụng rất phổ biến.

- Khoa học máy tính: là cơ sở để thiết kế các thuật toán, đồng thời đánh giá thời gian chạy, bộ nhớ của các thuật toán học máy.

Các nhóm giải thuật học máy (Hình 2-1):

- Học có giám sát: Máy tính được xem một số mẫu gồm đầu vào (input) và đầu ra (output) tương ứng trước. Sau khi học xong các mẫu này, máy tính quan sát một đầu vào mới và cho ra kết quả.
- Học không giám sát: Máy tính chỉ được xem các mẫu không có đầu ra, sau đó máy tính phải tự tìm cách phân loại các mẫu này và các mẫu mới.
- Học bán giám sát: Một dạng lai giữa hai nhóm giải thuật trên.
- Học tăng cường: Máy tính đưa ra quyết định hành động (action) và nhận kết quả phản hồi (response/reward) từ môi trường (environment). Sau đó máy tính tìm cách chỉnh sửa cách ra quyết định hành động của mình.

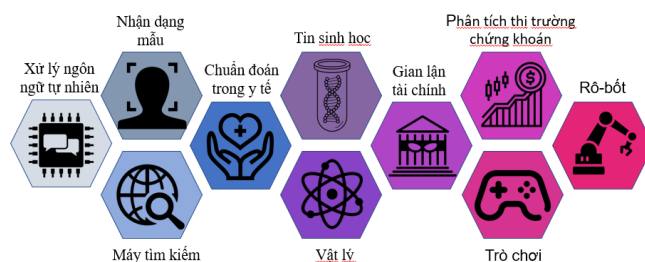


Hình 2- 1. Các nhóm giải thuật học máy

### 2.1.2. Các ứng dụng của học máy

Ứng dụng: Học máy có ứng dụng rộng khắp trong các ngành khoa học/sản xuất, đặc biệt những ngành cần phân tích khối lượng dữ liệu khổng lồ. Một số ứng dụng thường thấy (Hình 2-2)





*Hình 2- 2. Các ứng dụng của học máy*

- Xử lý ngôn ngữ tự nhiên (Natural Language Processing): xử lý văn bản, giao tiếp người – máy, ...
- Nhận dạng (Pattern Recognition): nhận dạng tiếng nói, chữ viết tay, vân tay, thị giác máy (Computer Vision) ...
- Tìm kiếm (Search Engine)
- Chẩn đoán trong y tế: phân tích ảnh X-quang, các hệ chuyên gia chẩn đoán tự động.
- Tin sinh học: phân loại chuỗi gen, quá trình hình thành gen/protein
- Vật lý: phân tích ảnh thiên văn, tác động giữa các hạt ...
- Phát hiện gian lận tài chính (financial fraud): gian lận thẻ tín dụng
- Phân tích thị trường chứng khoán (stock market analysis)
- Chơi trò chơi: tự động chơi cờ, hành động của các nhân vật ảo
- Rô-bốt: là tổng hợp của rất nhiều ngành khoa học, trong đó học máy tạo nên hệ thần kinh/bộ não của người máy

## **2.2. Thuật toán nhận dạng khuôn mặt**

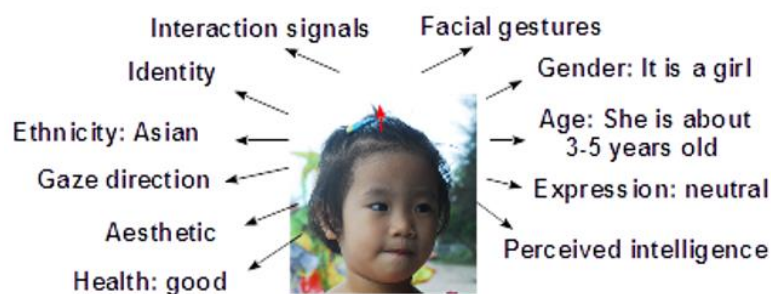
### **2.2.1. Bài toán về nhận dạng khuôn mặt**

#### *2.2.1.1. Giới thiệu*

Nhận dạng khuôn mặt (FR\_ Face Recognition) là được phát triển từ đầu những năm 90 của thập kỷ trước. Cho tới hiện nay, đây vẫn là một chủ đề nghiên cứu mở nhận được sự quan tâm của nhiều nhà nghiên cứu từ nhiều lĩnh vực nghiên cứu khác nhau như nhận dạng mẫu (Pattern Recognition), học máy (Machine Learning), thống

kê (Statistics), sinh trắc học (Biometrics). Điều này là do có rất nhiều ứng dụng thực tế cần tới một hệ thống nhận dạng mặt, từ các hệ thống quản lý đăng nhập đơn giản cho tới các ứng dụng giám sát tại các địa điểm công cộng (public areas surveillance) hoặc quản lý dân số (population management) và pháp lý (forensics). Bên cạnh đó, so với các hệ thống nhận dạng dựa trên các đặc điểm sinh trắc học khác của con người, như nhận dạng mống mắt và vân tay (fingerprint and iris recognition), dáng đi (gait recognition), nhận dạng mặt có nhiều ưu điểm:

- Một hệ thống nhận dạng mặt không đòi hỏi có sự tương tác trực tiếp giữa đối tượng được nhận dạng và hệ thống.
- Việc thu nhận dữ liệu (ảnh mặt) cho quá trình nhận dạng một con người dễ thực hiện hơn so với thu nhận các đặc điểm sinh trắc học khác (như thu nhận dấu vân tay và mống mắt).
- Dữ liệu về khuôn mặt phổ biến hơn so với các đặc trưng khác do sự bùng nổ các mạng xã hội (Facebook, Twitter ...), các dịch vụ chia sẻ dữ liệu đa phương tiện (Youtube, Vimeo ...) và sự phát triển mạnh mẽ của các thiết bị thu nhận hình ảnh.
- Từ ảnh khuôn mặt của một người ta có thể khai thác nhiều thông tin liên quan (Hình 2- 3) chứ không chỉ là danh tính, chẳng hạn như giới tính (gender), màu da (skin color), hướng nhìn (gaze direction), chủng tộc, hành vi, sức khỏe, độ tuổi, cảm xúc và mức độ thông minh ...

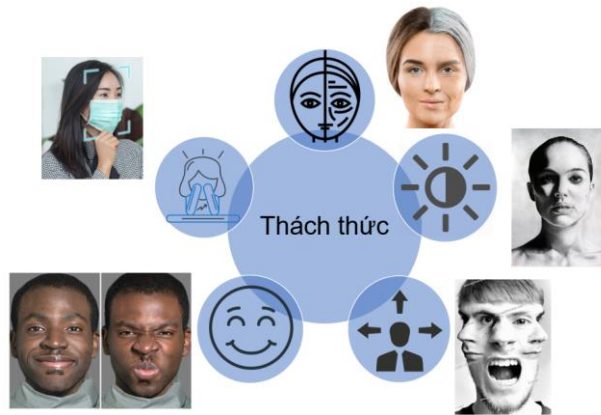


Hình 2- 3. Các thông tin có trong ảnh người

#### 2.2.1.2. Thách thức trong nhận dạng khuôn mặt

Tuy nhiên việc xây dựng một hệ thống nhận dạng mặt hoàn toàn tự động với khả năng nhận dạng chính xác cao thực sự là một thách thức đối với các nhà nghiên cứu. Điều này là do các yếu tố (chủ quan và khách quan) ảnh hưởng tới quá trình thu nhận ảnh và tạo ra các bức ảnh có độ khác biệt rất lớn của cùng một khuôn mặt.

Có thể liệt kê ra đây các yếu tố ảnh hưởng tới độ chính xác của một hệ thống nhận dạng mặt (Hình 2-4) :



*Hình 2- 4. Các thách thức của nhận dạng khuôn mặt*

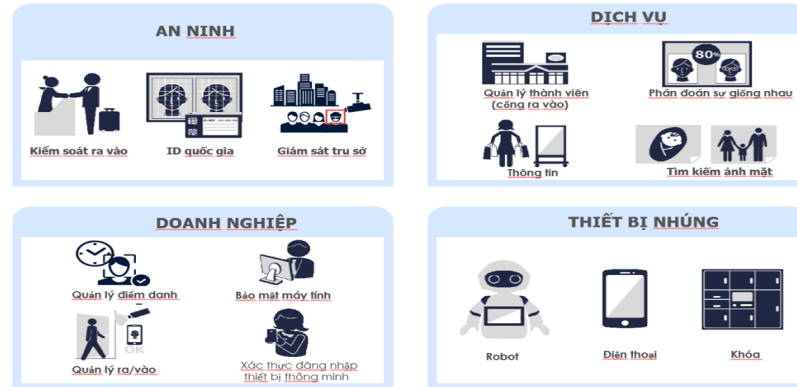
- Ánh sáng (light conditions). Các bức ảnh mặt thu nhận ở các điều kiện sáng khác nhau sẽ rất khác nhau và làm giảm sự chính xác trong quá trình nhận dạng.
- Thay đổi về tuổi (aging changes). Khuôn mặt người có các thay đổi lớn khi tuổi thay đổi và khó nhận dạng hơn ngay cả đối với hệ thống thị giác của con người.
- Các vấn đề về hướng (pose variations). Việc nhận dạng với các ảnh có góc chụp thẳng (frontal) có kết quả tốt hơn rất nhiều so với các ảnh được chụp ở góc nghiêng lớn hơn 45 độ. Giải pháp thường thấy đối với các ảnh có hướng chụp lớn là sử dụng các thuật toán nội suy để cố gắng bù đắp phần khuôn mặt bị che khuất.
- Cảm xúc (facial expression variations). Ở các trạng thái cảm xúc khác nhau, các đặc điểm quan trọng cho nhận dạng mặt (như mắt, mũi, miệng) có thể bị biến dạng (deformed) và dẫn tới các kết quả nhận dạng sai.
- Che khuất (occlusions). Các ảnh mặt có thể bị che khuất bởi các yếu tố khách quan như vật chắn ở trước mặt hoặc chủ quan như các phụ kiện trên khuôn mặt (khăn, kính mắt) và làm cho quá trình nhận dạng bị sai.

Các hệ thống nhận dạng mặt được chia thành hai loại: xác định danh tính (face identification) và xác thực (face verification). Bài toán xác định danh tính là bài toán dạng 1-N trong đó hệ thống sẽ đưa ra kết quả là danh tính của ảnh được nhận dạng dựa trên sự tương đồng của ảnh input với một danh sách N ảnh đã biết danh tính

chính xác. Trong khi đó, ở bài toán xác thực danh tính, hệ thống sẽ đưa ra câu trả lời đúng hoặc sai dựa vào việc xác định xem 2 bức ảnh có thuộc về cùng một người hay không. Trong phạm vi của đề tài này chỉ cần tập trung vào bài toán xác định danh tính.

### 2.2.1.3. Ứng dụng của nhận dạng khuôn mặt

Dưới đây là một số ứng dụng của nhận dạng khuôn mặt[14]



Hình 2- 5. Một số ứng dụng của nhận dạng khuôn mặt

- **Ứng dụng trong giám sát an ninh:** Các giải pháp kiểm soát an ninh , nhận diện khách lạ, khách VIP và đối tượng blacklist xuất hiện trong khu vực giám sát.v.v
- **Ứng dụng trong các ngành bán lẻ, dịch vụ:** theo dõi lượng khách vào ra , nhận diện khách hàng thân thiết, khách VIP và đối tượng xấu.v.v
- **Ứng dụng trong doanh nghiệp, công sở:** chấm công khuôn mặt , phát hiện hành vi (cầm dao, đeo mặt nạ, đeo khẩu trang, đeo kính đen, để râu ở những nơi quan trọng, bãi xe không dừng (đọc biển số tốc độ cao)
- **Ứng dụng trong chính phủ:** giám sát giao thông thông minh , phát hiện các hành vi vi phạm giao thông. Các giải pháp OCR giấy tờ tùy thân, ekyc cho chính phủ điện tử .v.v
- **Ứng dụng trong trường học:** điểm danh khuôn mặt, đăng ký, kiểm soát an ninh các khu vực cần theo dõi.v.v là các giải pháp nhận diện khuôn mặt cho trường học.
- **Ứng dụng trong các thiết bị IOT:** thiết bị kiểm soát ra vào bằng khuôn mặt, thiết bị đọc giấy tờ tùy thân .v.v

### 2.2.2. Hệ thống nhận dạng khuôn mặt

Một hệ thống nhận dạng khuôn mặt gồm các bước là:

- Thu thập dữ liệu: muốn thu thập được thì trước tiên phải phát hiện khuôn mặt có trong khung hình. Ở đồ án này dùng thuật toán HAAR CASCADES để nhận diện khuôn mặt. Lý do dùng nó là vì nó có độ chính xác cao đến 96,24%[15], nó phù hợp với thiết bị hạn chế tài nguyên[16] như Raspberry Pi
- Tiếp sau đó là huấn luyện dữ liệu đã thu thập được và nhận dạng khuôn mặt: Ở đây dùng thuật toán nhận dạng khuôn mặt bằng OpenCV thông qua LBPH. Lý do dùng thuật toán này là vì nó không bị ảnh hưởng bởi ánh sáng[17], là một trong những thuật toán nhận dạng khuôn mặt nhanh nhất[18]

#### 2.2.2.1. Khái niệm cơ bản về thuật toán HAAR CASCADES(phân tầng HAAR)

HAAR CASCADES là một cách tiếp cận máy học trong đó một chức năng tầng được đào tạo từ rất nhiều hình ảnh tích cực và tiêu cực. Hình ảnh tích cực là những hình ảnh bao gồm các khuôn mặt và hình ảnh tiêu cực là không có khuôn mặt. Trong nhận diện khuôn mặt, các đặc điểm hình ảnh được coi là thông tin số được trích xuất từ các hình ảnh có thể phân biệt hình ảnh này với hình ảnh khác.

Áp dụng mọi tính năng của thuật toán trên tất cả các hình ảnh đào tạo. Mọi hình ảnh đều có trọng lượng như nhau khi bắt đầu. Nó xác định ngưỡng tốt nhất sẽ phân loại các mặt thành tích cực và tiêu cực. Có thể có sai sót và phân loại sai. Chọn các tính năng có tỷ lệ lỗi tối thiểu, có nghĩa là đây là các tính năng phân loại tốt nhất hình ảnh có khuôn mặt và không có khuôn mặt.

Tất cả các kích thước và vị trí có thể có của mỗi nhân được sử dụng để tính toán nhiều tính năng.

#### 2.2.2.2. Phát hiện HAAR CASCADES trong OpenCV

OpenCV cung cấp trình huấn luyện cũng như công cụ dò tìm. Có thể đào tạo trình phân loại cho bất kỳ đối tượng nào như ô tô, máy bay và tòa nhà bằng cách sử dụng OpenCV. Có hai trạng thái chính của bộ phân loại hình ảnh theo tầng, đầu tiên là trạng thái huấn luyện và trạng thái còn lại là phát hiện.

OpenCV cung cấp hai ứng dụng để đào tạo trình phân loại tầng là `opencv_haartraining` và `opencv_traincascade`. Hai ứng dụng này lưu trữ bộ phân loại ở định dạng tệp khác nhau.

Để huấn luyện, cần một bộ mẫu. Có hai loại mẫu:

- Negative sample (Mẫu âm bản): Nó liên quan đến hình ảnh không phải vật thể.
- Positive samples (Mẫu dương tính): Là hình ảnh có liên quan với các đối tượng phát hiện.

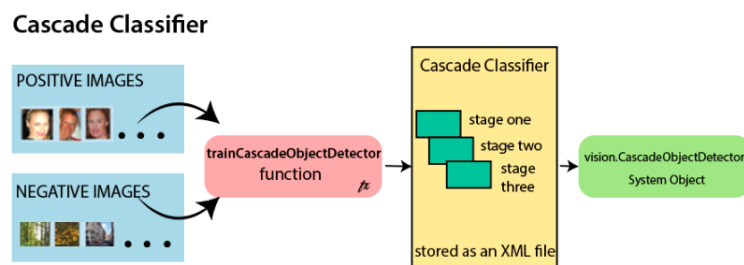
Một tập hợp các mẫu Negative sample phải được chuẩn bị theo cách thủ công, trong khi tập hợp các mẫu Positive samples được tạo bằng tiện ích `opencv_createsamples`.

### Mẫu Negative sample

Mẫu Negative sample được lấy từ hình ảnh tùy ý. Các mẫu phủ định được thêm vào một tệp văn bản. Mỗi dòng của tệp chứa một tên tệp hình ảnh (liên quan đến thư mục của tệp mô tả) của mẫu phủ định. Tệp này phải được tạo thủ công. Hình ảnh xác định có thể có kích thước khác nhau.

### Mẫu Positive samples

Các mẫu Positive samples được tạo bởi `opencv_createsamples`. Những mẫu này có thể được tạo từ một hình ảnh duy nhất với một đối tượng hoặc từ một bộ sưu tập trước đó. Điều quan trọng cần nhớ là yêu cầu một tập dữ liệu lớn về các mẫu dương tính trước khi bạn cung cấp nó cho tiện ích được đề cập bởi vì nó chỉ áp dụng phép chuyển đổi phối cảnh. Dưới đây là các bước phát hiện phân tầng Cascade Classifier



Hình 2- 6. Các bước phát hiện phân tầng Cascade classifier

Ở đây sẽ thảo luận về việc phát hiện. OpenCV đã chứa nhiều bộ phân loại được đào tạo trước khác nhau cho khuôn mặt, mắt, nụ cười, v.v. Các tệp XML đó được lưu trữ trong thư mục `opencv/data/haarcascades`

Haarcascade hoạt động trong 4 giai đoạn:

- Lựa chọn tính năng Haar: Một tính năng giống như Haar bao gồm các vùng tối và vùng sáng. Nó tạo ra một giá trị duy nhất bằng cách lấy hiệu số của tổng cường độ của các vùng tối và tổng các cường độ của các vùng sáng. Nó được thực hiện để trích xuất các yếu tố hữu ích cần thiết cho việc xác định một đối tượng Pipeline tách rời giữa feature extractors và classifier.
- Tạo hình ảnh tích hợp: Một pixel nhất định trong hình ảnh tích hợp là tổng của tất cả các pixel bên trái và tất cả các pixel phía trên nó. Vì quá trình trích xuất các tính năng giống Haar liên quan đến việc tính toán sự khác biệt của các vùng hình chữ nhật tối và sáng, sự ra đời của Hình ảnh Tích hợp làm giảm đáng kể thời gian cần thiết để hoàn thành nhiệm vụ này.
- AdaBoost Training: Thuật toán này chọn các tính năng tốt nhất từ tất cả các tính năng. Nó kết hợp nhiều “bộ phân loại yếu” (tính năng tốt nhất) thành một “bộ phân loại mạnh”. “Bộ phân loại mạnh” được tạo về cơ bản là sự kết hợp tuyến tính của tất cả các “bộ phân loại yếu”.
- Bộ phân loại theo tầng: Đây là một phương pháp để kết hợp các bộ phân loại ngày càng phức tạp hơn như AdaBoost trong một tầng cho phép nhanh chóng loại bỏ đầu vào âm (không phải mặt) trong khi chi tiêu nhiều tính toán hơn trên các vùng giống như mặt có triển vọng hoặc tích cực. Nó làm giảm đáng kể thời gian tính toán và làm cho quá trình hiệu quả hơn.

Dưới đây là code và kết quả từ code

```
import cv2
img = cv2.imread('image.jpg')
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
haar_cascade = cv2.CascadeClassifier('opencv/data/haarcascades/haarcascade_frontalface_default.xml')
faces_rest = haar_cascade.detectMultiScale(gray_img, 1.1, 9)
for (x,y,w,h) in faces_rest:
    cv2.rectangle(img, (x,y), (x+w, y+h),(0, 255, 0), 2)
cv2.imshow("Image", img)
cv2.waitKey(0)
```





Hình 2- 7. Ví dụ minh họa sau khi dùng Haarcascade để nhận diện khuôn mặt

### 2.2.2.3. Nhận dạng khuôn mặt bằng OpenCV

Nhận dạng khuôn mặt là một công việc đơn giản đối với con người. Nhận dạng khuôn mặt thành công có xu hướng nhận dạng hiệu quả các đặc điểm bên trong (mắt, mũi, miệng) hoặc các đặc điểm bên ngoài (đầu, mặt, chân tóc). Ở đây câu hỏi đặt ra là bộ não con người mã hóa nó như thế nào?

David Hubel và Torsten Wiesel chỉ ra rằng não của con người có các tế bào thần kinh chuyên biệt phản ứng với đặc điểm cục bộ độc đáo của cảnh, chẳng hạn như đường thẳng, góc cạnh hoặc chuyển động. Bộ não của kết hợp các nguồn thông tin khác nhau thành các mẫu hữu ích; không coi hình ảnh là những thứ tán xạ. Nếu định nghĩa nhận dạng khuôn mặt bằng từ đơn giản, “Nhận dạng khuôn mặt tự động là tất cả nhằm lấy ra những đặc điểm có ý nghĩa đó từ một hình ảnh và đưa chúng vào một biểu diễn hữu ích sau đó thực hiện một số phân loại trên chúng”.

Ý tưởng cơ bản của nhận dạng khuôn mặt dựa trên các đặc điểm hình học của khuôn mặt. Đây là cách tiếp cận khả thi và trực quan nhất để nhận dạng khuôn mặt. Hệ thống nhận dạng khuôn mặt tự động đầu tiên được mô tả ở vị trí của mắt, tai, mũi. Các điểm định vị này được gọi là vector đặc trưng (khoảng cách giữa các điểm).

Việc nhận dạng khuôn mặt đạt được bằng cách tính toán khoảng cách Euclidean giữa các vector đặc trưng của một đầu dò và hình ảnh tham chiếu. Phương pháp này có hiệu quả trong việc thay đổi độ chiếu sáng theo bản chất của nó, nhưng nó có một nhược điểm đáng kể. Việc đăng ký chính xác của nhà sản xuất là rất khó.

Hệ thống nhận dạng khuôn mặt có thể hoạt động về cơ bản ở hai chế độ:



- Xác thực hoặc Xác minh hình ảnh khuôn mặt: Nó so sánh hình ảnh khuôn mặt đầu vào với hình ảnh khuôn mặt liên quan đến người dùng, được yêu cầu xác thực. Đó là một so sánh  $1 \times 1$ .
- Nhận dạng khuôn mặt: Về cơ bản, nó so sánh các hình ảnh khuôn mặt đầu vào từ một tập dữ liệu để tìm ra người dùng phù hợp với khuôn mặt đầu vào đó. Đó là một so sánh  $1 \times N$ .

Có nhiều loại thuật toán nhận dạng khuôn mặt, ví dụ:

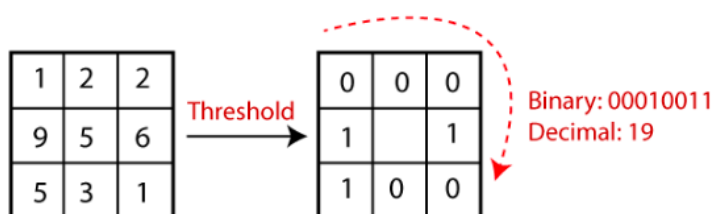
- Eigenfaces (1991)
- Biểu đồ mẫu nhị phân cục bộ (LBPH) (1996)
- Fisherfaces (1997)
- Chuyển đổi tính năng bất biến quy mô (SIFT) (1999)
- Tăng tốc các tính năng mạnh mẽ (SURF) (2006)

Mỗi thuật toán tuân theo các cách tiếp cận khác nhau để trích xuất thông tin hình ảnh và thực hiện đối sánh với hình ảnh đầu vào. Ở đây sẽ thảo luận về thuật toán Biểu đồ mô hình nhị phân cục bộ (LBPH) là một trong những thuật toán phổ biến và lâu đời nhất.

#### 2.2.2.4. Giới thiệu LBPH

Thuật toán Biểu đồ mô hình nhị phân cục bộ là một cách tiếp cận đơn giản gắn nhãn các pixel của hình ảnh ngưỡng vùng lân cận của mỗi pixel. Nói cách khác, LBPH tóm tắt cấu trúc cục bộ trong một hình ảnh bằng cách so sánh từng pixel với các pixel lân cận của nó và kết quả được chuyển đổi thành một số nhị phân. Nó được định nghĩa lần đầu tiên vào năm 1994 (LBPH) và kể từ thời điểm đó, nó đã được coi là một thuật toán mạnh mẽ để phân loại kết cấu.

Thuật toán này thường tập trung vào việc trích xuất các tính năng cục bộ từ hình ảnh. Ý tưởng cơ bản là không xem toàn bộ hình ảnh như một vector chiều cao; nó chỉ tập trung vào các tính năng cục bộ của một đối tượng.



Hình 2- 8. Trích xuất các tính năng cục bộ từ hình ảnh

Trong hình 2- 8, lấy một pixel làm trung tâm và ngưỡng đối tượng của nó. Nếu cường độ của pixel trung tâm lớn hơn bằng với hàng xóm của nó, thì biểu thị nó bằng 1 và nếu không thì biểu thị nó bằng 0.

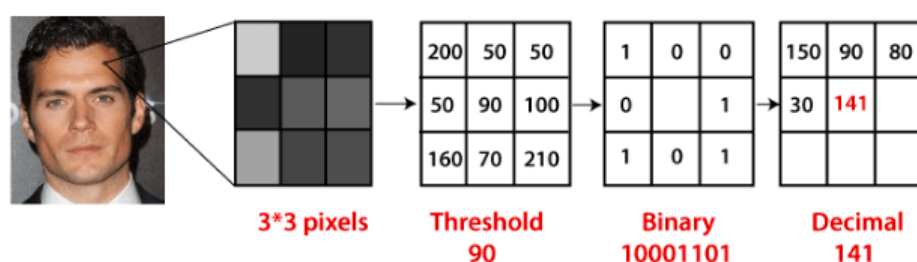
Các bước của thuật toán:

**Bước 1: Chọn các tham số: LBPH chấp nhận bốn tham số:**

- **Radius:** Nó đại diện cho bán kính xung quanh pixel trung tâm. Nó thường được đặt thành 1. Nó được sử dụng để xây dựng mẫu nhị phân cục bộ tròn.
- **Neighbors:** Số lượng điểm mẫu để xây dựng mẫu nhị phân hình tròn.
- **Grid X:** Số ô theo hướng ngang. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thước của vector đặc trưng thu được càng cao.
- **Grid Y:** Số ô theo hướng dọc. Càng biểu thị nhiều ô và lưới mịn hơn, thì kích thước của vector đặc trưng thu được càng cao.

**Bước 2: Huấn luyện thuật toán:** Bước đầu tiên là huấn luyện thuật toán. Nó yêu cầu một tập dữ liệu với hình ảnh khuôn mặt của người cần nhận dạng. Một ID duy nhất (có thể là số hoặc tên của người) phải được cung cấp cùng với mỗi hình ảnh. Sau đó, thuật toán sử dụng thông tin này để nhận dạng hình ảnh đầu vào và cung cấp cho bạn kết quả đầu ra. Hình ảnh của một người cụ thể phải có cùng một ID. Hãy hiểu tính toán LBPH trong bước tiếp theo.

- **Sử dụng phép toán LBPH:** Trong bước này, phép tính LBPH được sử dụng để tạo ra một hình ảnh trung gian mô tả hình ảnh gốc một cách cụ thể thông qua việc làm nổi bật đặc điểm khuôn mặt. Các tham số radius và neighbors được sử dụng trong khái niệm của số trượt.



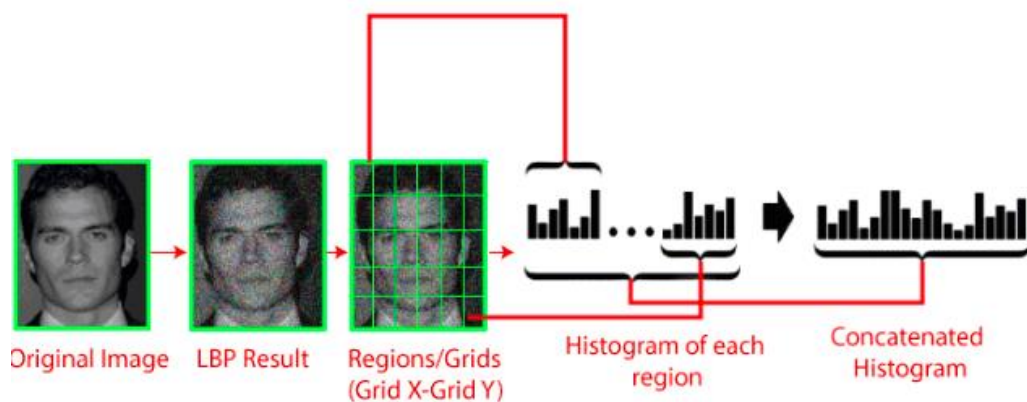
Hình 2- 9. Ví dụ về phép toán LBPH

Để hiểu một cách cụ thể hơn về hình 2- 9, chia nó thành nhiều bước nhỏ:

- Giả sử hình ảnh khuôn mặt đầu vào là thang độ xám.
- Có thể lấy một phần của hình ảnh này dưới dạng cửa sổ 3×3 pixel.

- Có thể sử dụng ma trận  $3 \times 3$  chứa cường độ của mỗi pixel (0-255).
- Sau đó, cần lấy giá trị trung tâm của ma trận sẽ được sử dụng làm ngưỡng.
- Giá trị này sẽ được sử dụng để xác định các giá trị mới từ 8 hàng xóm.
- Đối với mỗi lân cận của giá trị trung tâm (ngưỡng), cần đặt một giá trị nhị phân mới. Giá trị 1 được đặt cho giá trị bằng hoặc cao hơn ngưỡng và 0 cho giá trị thấp hơn ngưỡng.
- Bây giờ ma trận sẽ chỉ bao gồm các giá trị nhị phân (bỏ qua giá trị trung tâm). Cần quan tâm đến từng giá trị nhị phân từ mỗi vị trí từ dòng ma trận từng dòng thành các giá trị nhị phân mới (10001101). Có những cách tiếp cận khác để nối các giá trị nhị phân (theo chiều kim đồng hồ), nhưng kết quả cuối cùng sẽ giống nhau.
- Chuyển đổi giá trị nhị phân này thành giá trị thập phân và đặt nó thành giá trị trung tâm của ma trận, là một pixel từ ảnh gốc.
- Sau khi hoàn thành thủ tục LBPH thì sẽ nhận được hình ảnh mới, thể hiện các đặc điểm tốt hơn của hình ảnh gốc

**Bước 3: Trích xuất biểu đồ từ hình ảnh:** Hình ảnh được tạo ở bước cuối cùng, có thể sử dụng các tham số Grid X và Grid Y để chia hình ảnh thành nhiều lưới, hãy xem xét hình ảnh 2-10:



Hình 2- 10. Trích xuất biểu đồ từ hình ảnh

- Có một hình ảnh ở thang độ xám; mỗi biểu đồ (từ mỗi lưới) sẽ chỉ chứa 256 vị trí biểu thị sự xuất hiện của mỗi cường độ pixel.
- Cần phải tạo một biểu đồ mới lớn hơn bằng cách nối từng biểu đồ.

**Bước 4: Thực hiện nhận dạng khuôn mặt:** Bây giờ, thuật toán đã được đào tạo bài bản. Biểu đồ trích xuất được sử dụng để đại diện cho từng hình ảnh từ tập dữ liệu đào tạo. Đối với hình ảnh mới, sẽ cần thực hiện lại các bước và tạo biểu đồ mới. Để

tìm hình ảnh phù hợp với hình ảnh đã cho, chỉ cần so khớp hai biểu đồ và trả về hình ảnh có biểu đồ gần nhất.

- Có nhiều cách tiếp cận khác nhau để so sánh các biểu đồ (tính toán khoảng cách giữa hai biểu đồ), ví dụ: Khoảng cách Euclide, chi-square, giá trị tuyệt đối, v.v. có thể sử dụng khoảng cách Euclid dựa trên công thức sau:

$$D = \sqrt{\sum_{i=1}^n (\text{hist } 1_i - \text{hist } 2_i)^2}$$

- Thuật toán sẽ trả về ID dưới dạng đầu ra từ hình ảnh có biểu đồ gần nhất. Thuật toán cũng sẽ trả về khoảng cách tính toán có thể được gọi là đo độ tin cậy. Nếu độ tin cậy thấp hơn giá trị ngưỡng, điều đó có nghĩa là thuật toán đã nhận dạng khuôn mặt thành công.

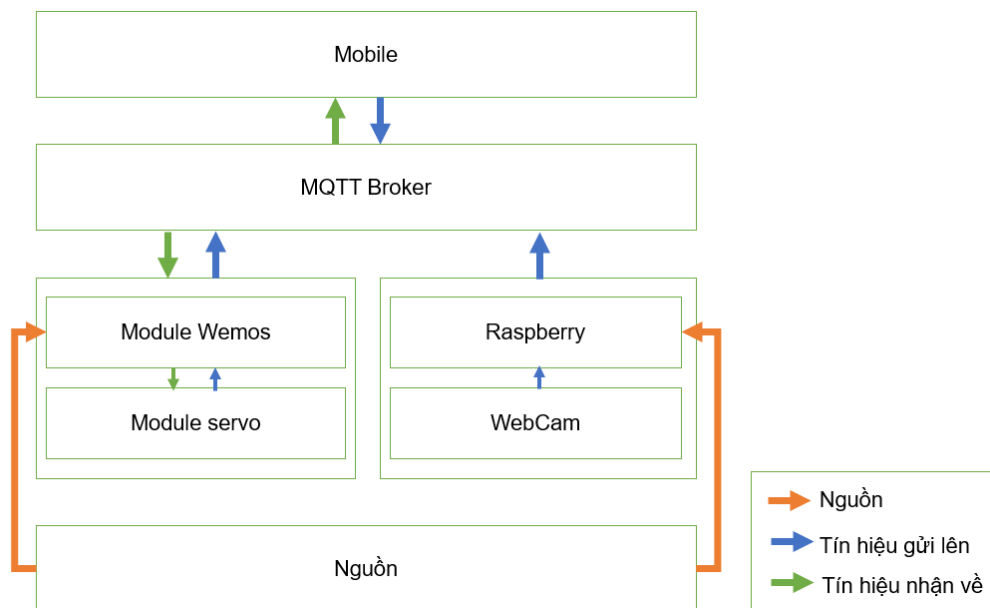
## CHƯƠNG 3. TRIỂN KHAI, THỰC NGHIỆM VÀ ĐÁNH GIÁ

Sau khi đưa ra các lý thuyết, thuật toán học máy thì đến bước triển khai, thực nghiệm hệ thống với những lý thuyết đưa ra ở các chương trên.

### 3.1. Triển khai, thực nghiệm

#### 3.1.1. Sơ đồ hệ thống

##### 3.1.1.1. Sơ đồ tổng quan



Hình 3- 1. Sơ đồ tổng quan của hệ thống

#### Tín hiệu gửi lên, nhận về của hệ thống

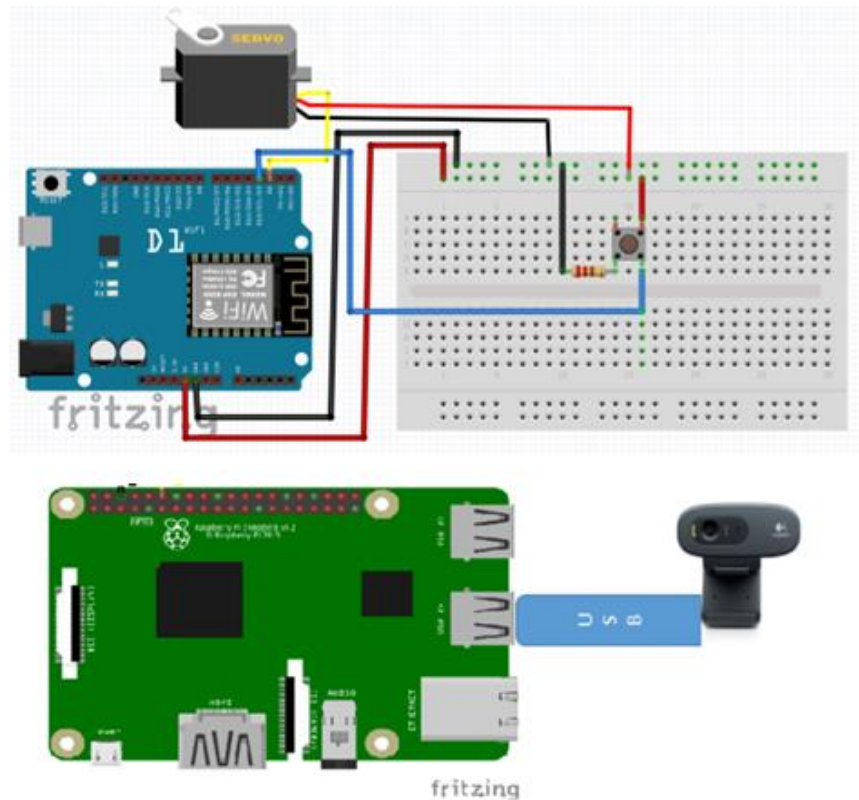
- Tín hiệu gửi lên: các tín hiệu mà các thiết bị, phần cứng phía người dùng qua ứng dụng gửi lên
- Tín hiệu nhận về: các tín hiệu mà các thiết bị, phần cứng nhận về để thực hiện các nhiệm vụ cụ thể

#### Chi tiết sơ đồ tổng quan:

- MQTT Broker: Thu nhận tín hiệu từ các thiết bị gửi lên
- Mobile: nhận tín hiệu để đưa ra cảnh báo với người dùng, đồng thời hiển thị ảnh người lạ
- Nguồn: cung cấp nguồn cho các bộ xử lý trung tâm và các module
- Module Wemos: tích hợp wifi để xử lý nhận tín hiệu từ MQTT Broker

- Module servo: nhận lệnh đóng/ mở cửa từ module Wemos
- Raspberry Pi: xử lý nhận dạng khuôn mặt
- Webcam: cung cấp các khung ảnh cho Rpi để nhận dạng khuôn mặt

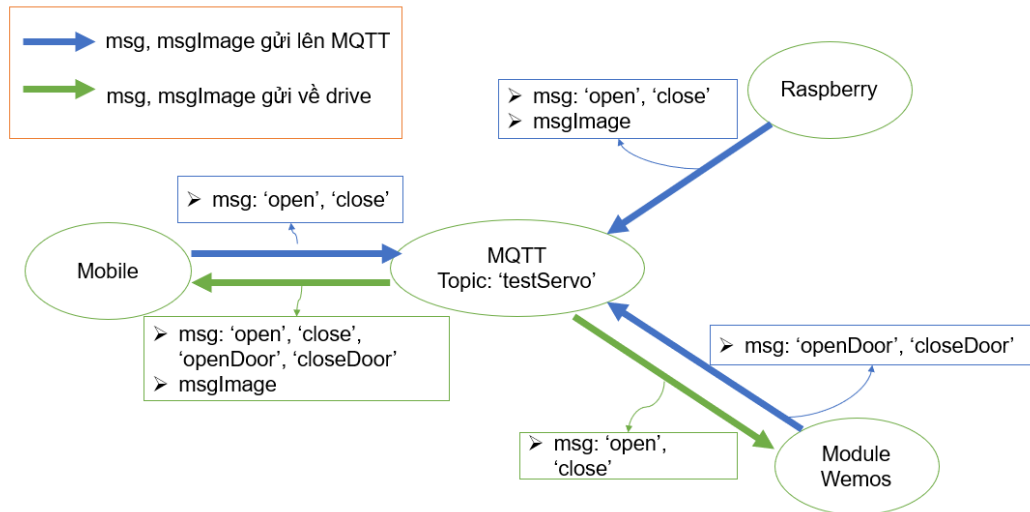
#### 3.1.1.2. Sơ đồ phần cứng



Hình 3- 2. Sơ đồ phần cứng của hệ thống

Sơ đồ phần cứng gồm Rpi với Wemos kết nối độc lập với nhau. Rpi kết nối với Webcam See3cam để thu thập các khung ảnh để gửi về Rpi xử lý phần nhận dạng khuôn mặt. Còn Womos kết nối với Servo G90 để thực hiện nhiệm vụ đóng mở cửa sau khi nhận được dữ liệu từ Rpi gửi lên MQTT Broker

### 3.1.1.3. Sơ đồ luồng dữ liệu



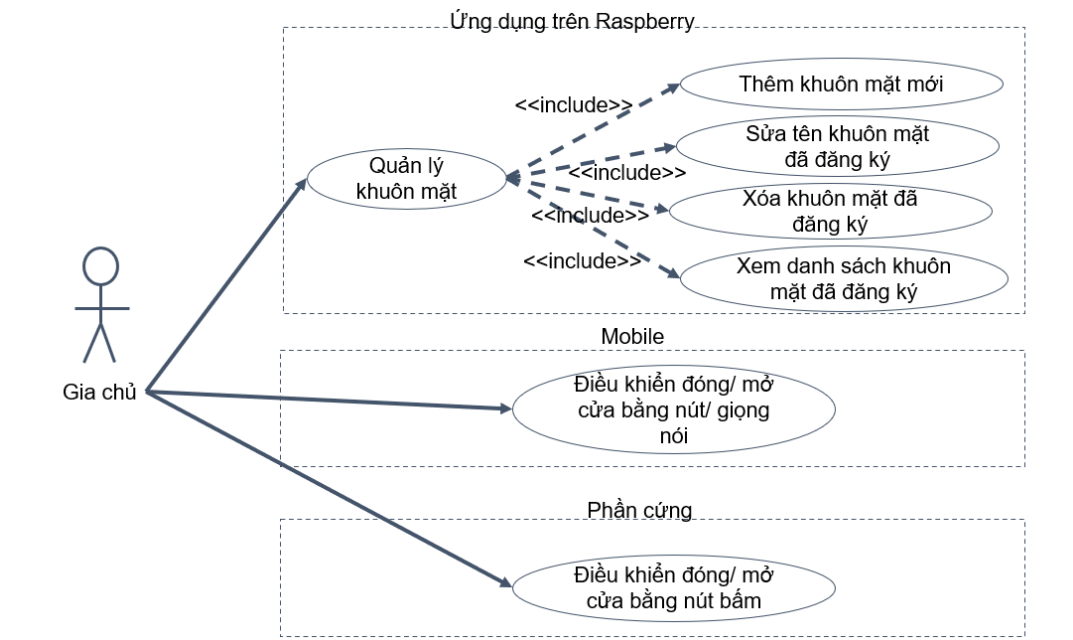
Hình 3- 3. Sơ đồ luồng dữ liệu tin nhắn giữa MQTT Broker và các thiết bị khác

Khi Rpi phát hiện ra có người và người đó là gia chủ thì Rpi sẽ gửi tin nhắn “open”, còn người lạ thì sẽ gửi tin nhắn “close” và hình ảnh người lạ “msgImage” (tất cả tin nhắn để được mã hóa AES 256 CBC) có topic là “testServo” trong MQTT Broker.

Mobile, module Wemos đã đăng ký topic sẽ nhận được tin nhắn và giải mã nó, rồi xử lý nó.

- Đối với module Wemos thì xuất tín hiệu ra các chân để điều khiển servo. Đồng thời gửi lên các tin nhắn “openDoor”, “closeDoor” khi trạng thái servo là đóng hoặc mở.
- Đối với Mobile thì sẽ hiển thị các tin nhắn về hình ảnh người lạ “msgImage” để hiển thị lên và đưa ra cảnh báo. Đồng thời, Mobile có thêm chức năng điều khiển các thiết bị nên sẽ gửi lên các tin nhắn “open”, “close”

#### 3.1.1.4. Sơ đồ use case chức năng cho gia chủ



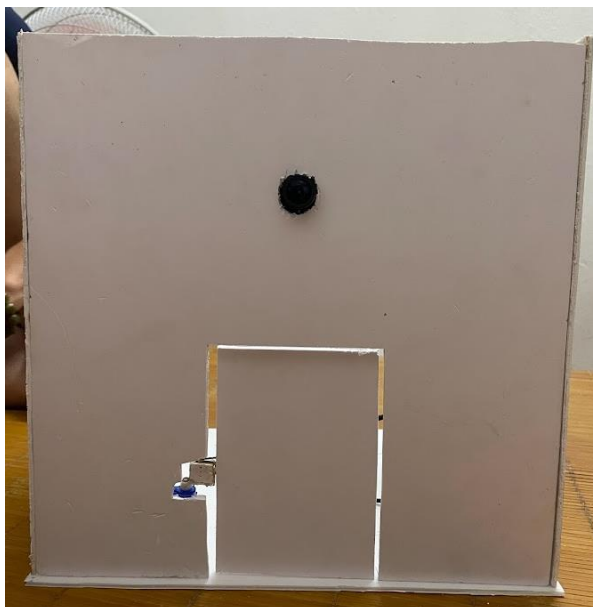
Hình 3- 4. Sơ đồ use case chức năng cho gia chủ



### 3.1.2. Thi công phần cứng



*Hình 3- 5. Phần cứng thi công*



*Hình 3- 6. Mô hình sau khi lắp ráp*

### 3.1.3. Thiết kế phần mềm cho các chương trình

Hệ thống cụ thể có các chương trình sau

#### Bảo mật SSL/TLS

- Chứng chỉ được thực nghiệm trên Windows 10

#### Chương trình nhận dạng khuôn mặt

- Chương trình được thực thi trên hệ điều hành Raspbian của Raspberry Pi 3 Model B+
- Sử dụng ngôn ngữ lập trình: Python
- Thư viện hỗ trợ: OpenCV, paho.mqtt.client, ssl, base64, aes, tkinter

#### Chương trình thực thi đóng/mở cửa

- Chương trình được thực thi trên Wemos
- Sử dụng ngôn ngữ lập trình: C++
- Thư viện hỗ trợ: WiFiClientSecure, PubSubClient, Servo, base64

#### Chương trình đưa ra cảnh báo và hiển thị ảnh người lạ

- Chương trình được thực thi trên Mobile
- Sử dụng ngôn ngữ lập trình: Dart
- Thư viện hỗ trợ: mqtt\_client, encrypt, path\_provider, các thư viện hệ thống

#### 3.1.3.1. Thiết lập chứng chỉ SSL/TLS

Hiện thì MQTT Broker đang được cài đặt trên Windown. Còn MQTT Client là các thiết bị trong hệ thống gồm: Rpi, Wemos, Mobile

#### Tạo chứng chỉ bằng OpenSSL

Đầu tiên vào command Prompt chế độ “**run as Administrator**”

Tiếp theo vào thư mục OpenSSL

Cd C:\OpenSSL\bin

Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd c:\OpenSSL\bin

c:\OpenSSL\bin>
```

### **Tạo cặp CA key**

```
openssl genrsa -des3 -out mqtt_ca.key 2048
openssl req -new -x509 -days 3650 -key mqtt_ca.key -out mqtt_ca.crt
```

```
C:\OpenSSL\bin>openssl req -new -x509 -days 3650 -key mqtt_ca.key -out mqtt_ca.
crt
Enter pass phrase for mqtt_ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:VN
State or Province Name (full name) [Some-State]:HANOI
Locality Name (eg, city) []:THANHTRI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MqttCa
Organizational Unit Name (eg, section) []:mqtt-servo
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:linhct020328@gmail.com
```

### **Tạo cặp Client key**

```
openssl genrsa -out mqtt_client.key 2048
openssl req -new -out mqtt_client.csr -key mqtt_client.key
```

```
C:\OpenSSL\bin>openssl req -new -out mqtt_client.csr -key mqtt_client.key
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:VN
State or Province Name (full name) [Some-State]:HANOI
Locality Name (eg, city) []:THANHTRI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MqttClient
Organizational Unit Name (eg, section) []:mqtt-servo
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:linhct020328@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234567890
An optional company name []:1234567890

C:\OpenSSL\bin>
```

### **Xác minh và ký yêu cầu chứng chỉ**

```
openssl x509 -req -in mqtt_client.csr -CA mqtt_ca.crt -CAkey mqtt_ca.key -
CAcreateserial -out mqtt_client.crt -days 3650
```

### **Kiểm tra SSL được tạo có chính xác hay không**

```
openssl verify -CAfile mqtt_ca.crt mqtt_client.crt
```

### **Copy các file .crt, .key vào thư mục certs**

Local Disk (C:) > mosquitto > certs

Name	Date modified	Type	Size
mqtt_ca.crt	2/26/2022 6:38 PM	Security Certificate	2 KB
mqtt_ca.key	2/26/2022 6:35 PM	KEY File	2 KB
mqtt_ca.srl	2/26/2022 6:41 PM	SRL File	1 KB
mqtt_client.crt	2/26/2022 6:41 PM	Security Certificate	2 KB
mqtt_client.csr	2/26/2022 6:40 PM	CSR File	2 KB
mqtt_client.key	2/26/2022 6:38 PM	KEY File	2 KB

### **Vào lại mosquitto, chỉnh sửa file mosquitto.conf**

```
listener 1883
protocol mqtt
allow_anonymous true
```

```
require_certificate false
```

```
cafile C:\mosquitto\certs\mqtt_ca.crt  
certfile C:\mosquitto\certs\mqtt_client.crt  
keyfile C:\mosquitto\certs\mqtt_client.key  
tls_version tlsv1.2
```

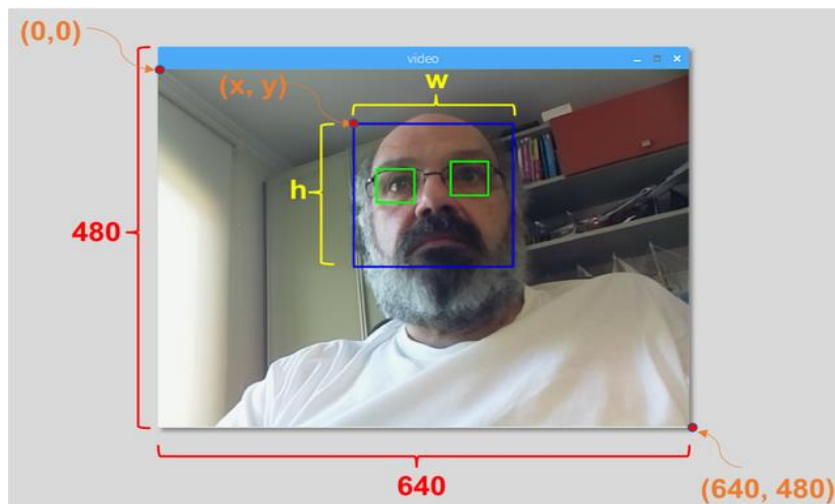
### **Chạy lại mosquitto**

```
net stop mosquitto  
net start mosquitto  
mosquitto -c mosquitto.conf -v
```

#### *3.1.3.2. Chương trình nhận dạng khuôn mặt*

##### **a. Nhận diện khuôn mặt(Face detection)**

Nhiệm vụ cơ bản nhất của Nhận dạng khuôn mặt tất nhiên là "Phát hiện ra khuôn mặt trong khung hình"



*Hình 3- 7. Phát hiện khuôn mặt trong khung hình*

Cách phổ biến nhất để phát hiện khuôn mặt (hoặc bất kỳ đối tượng nào), là sử dụng “Haar Cascade classifier”

Phát hiện đối tượng sử dụng Haar Cascade classifier là một phương pháp phát hiện đối tượng hiệu quả được đề xuất bởi Paul Viola và Michael Jones trong bài báo

của họ, " Rapid Object Detection using a Boosted Cascade of Simple Features " vào năm 2001. Đây là một cách tiếp cận dựa trên học máy, nơi một chức năng cascade được đào tạo từ rất nhiều hình ảnh tích cực và tiêu cực. Sau đó, nó được sử dụng để phát hiện các đối tượng trong các hình ảnh khác.

Ban đầu, thuật toán cần rất nhiều hình ảnh tích cực (hình ảnh khuôn mặt) và hình ảnh tiêu cực (hình ảnh không có khuôn mặt) để đào tạo phân loại. Sau đó, cần trích xuất các tính năng từ nó. Tin tốt là OpenCV đi kèm với một huấn luyện viên cũng như một máy dò.

Thư viện cần dùng

```
import numpy as np
import cv2
```

Dòng lệnh “classifier”(trong thư mục “Cascades”)

```
faceCascade =
cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml')
```

Sau đó, đặt máy ảnh trong vòng lặp, đầu vào ở chế độ màu xám. Gọi chức năng “classifier”, truyền 1 số thông số quan trọng như hệ số tỷ lệ, số lượng hàng xóm và kích thước tối thiểu của khuôn mặt được phát hiện.

```
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.2,
    minNeighbors=5,
    minSize=(20, 20)
)
```

- Gray là hình ảnh màu xám đầu vào.
- scaleFactor là tham số chỉ định kích thước hình ảnh được giảm bao nhiêu ở mỗi thang đo hình ảnh. Nó được sử dụng để tạo ra kim tự tháp quy mô.
- minNeighbors là một tham số chỉ định có bao nhiêu hàng xóm mỗi hình chữ nhật ứng cử viên nên có, để giữ lại nó. Một con số cao hơn cho kết quả dương tính giả thấp hơn.
- minSize là kích thước hình chữ nhật tối thiểu được coi là khuôn mặt.

Chức năng này sẽ phát hiện khuôn mặt trên hình ảnh. Tiếp theo, cần phải "đánh dấu" các khuôn mặt trong hình ảnh, sử dụng, ví dụ, một hình chữ nhật màu xanh. Điều này được thực hiện với phần này của mã:

```
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
```

Nếu khuôn mặt được tìm thấy, nó trả về vị trí của khuôn mặt được phát hiện dưới dạng hình chữ nhật với góc trái lên (x,y) và có "w" là Chiều rộng và "h" là Chiều cao ==> (x,y,w,h). Mời các bạn xem hình ảnh trên.

Một khi có được những vị trí này, có thể tạo một "ROI" (hình chữ nhật vẽ) cho khuôn mặt và trình bày kết quả với chức năng imshow().

```
import numpy as np
import cv2

faceCascade =
cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height

while True:
    ret, img = cap.read()
    img = cv2.flip(img, -1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.2,
        minNeighbors=5,
        minSize=(20, 20)
    )

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
```

```

cv2.imshow('video',img)

k = cv2.waitKey(30) & 0xff
if k == 27: # press 'ESC' to quit
    break

cap.release()
cv2.destroyAllWindows()

```

### b. Thu thập dữ liệu

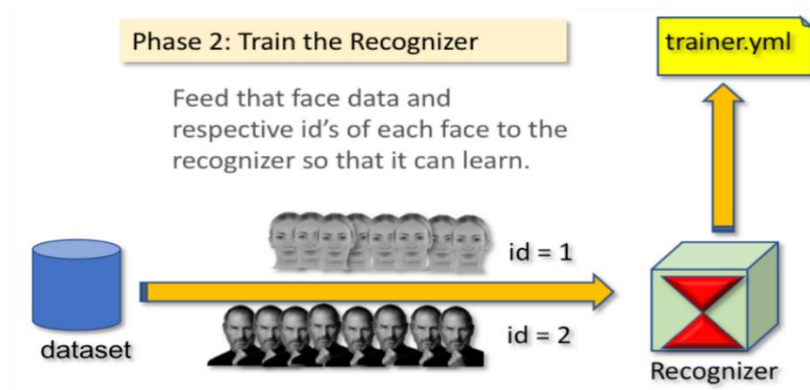
Phần này cũng giống nhận diện khuôn mặt, nhưng cho thêm lệnh đầu vào để chụp id người dùng

```
face_id = input("\n enter user id end press ==> ")
```

Và với mỗi khung hình được chụp sẽ cần lưu ảnh vào “dataset”

```
cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg",
gray[y:y+h,x:x+w])
```

### c. Huấn luyện



Hình 3- 8. Huấn luyện dữ liệu khuôn mặt của những người được nhận dạng

Trong giai đoạn thứ hai này, cần phải lấy tất cả dữ liệu người dùng từ dataset của và "trainer" OpenCV Recognizer. Điều này được thực hiện trực tiếp bởi một chức năng OpenCV cụ thể. Kết quả sẽ là một tệp .yml sẽ được lưu trên thư mục "trainer/".



Sử dụng như một bộ nhận dạng, Bộ nhận dạng khuôn mặt LBPH (LOCAL BINARY PATTERNS HISTOGRAMS), bao gồm trong gói OpenCV

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Chức năng “getImagesAndLabels(path)”, sẽ chụp tất cả các ảnh trong thư mục “dataset/”, trả về 2 mảng: “Ids” và “face”. Với những mảng đó làm đầu vào, ta có “train our recognizer”

```
recognizer.train(faces, ids)
```

Do đó 1 tệp tên “trainer.yml” sẽ được lưu trong thư mục trainer

#### **d. Nhận dạng khuôn mặt**

Ở đây, sẽ cần chụp một khuôn mặt tươi mới trên máy ảnh và nếu người này có khuôn mặt của mình được chụp và đào tạo trước đó. Nhận dạng khuôn mặt sẽ đưa ra một "dự đoán" trả lại id và chỉ số của nó, cho thấy người nhận dạng tự tin như thế nào với trận đấu này.

Names bao gồm 1 mảng, hiển thị “name” thay vì Id được đánh số

```
names = ['None', 'Hien', 'Linh', 'Tuan']
```

Khi danh sách “name” này được cho vào file “namelist.txt”. Khi đó sẽ đọc file này và lấy các “name” để được mảng Names như trên

```
names = []
f = open("listName/nameslist.txt", "r")
x = f.read()
z = x.rstrip().split("\n")
for i in z:
    smallList = i.rstrip()
    names.append(smallList)
```

Tiếp theo, sẽ phát hiện ra một khuôn mặt, giống như đã làm trước đây với haasCascade classifier. Có một khuôn mặt được phát hiện, có thể gọi chức năng quan trọng nhất trong mã trên:

```
id, confidence = recognizer.predict(gray portion of the face)
```

recognizer.predict sẽ lấy như một tham số một phần chụp được khuôn mặt để được phân tích và sẽ trả lại chủ sở hữu, cho biết id của nó với mức độ tự tin

#### e. Kết nối MQTT

Thư viện

```
import paho.mqtt.client as paho
```

Các thông tin cần thiết để kết nối MQTT

```
clientId = "mqtt-servo"  
topic = "testServo"  
mqtt_broker = "192.168.0.103"  
mqtt_port = 1883  
mqttUser = "linh99"  
mqttPassword = "1234567890"
```

Để kết nối với MQTT

```
client = paho.Client()  
client.connect(mqtt_broker, mqtt_port)  
client.username_pw_set(username=mqttUser, password=mqttPassword)
```

#### f. Bảo mật SSL/TLS

Thư viện

```
import ssl
```

Các path gọi đến các file ca, key

```
pathCa = './certs_localhost/mqtt_ca.crt'  
pathClient = './certs_localhost/mqtt_client.crt'  
pathClientKey = './certs_localhost/mqtt_client.key'
```

Cấu hình các tùy chọn mã hóa và xác thực mạng. Bật hỗ trợ SSL/TLS

```
client.tls_set(pathCa, pathClient, pathClientKey,  
tls_version=ssl.PROTOCOL_TLSv1_2)
```

```
client.tls_insecure_set(True)
```

### g. Mã hóa AES

Thư viện của file mã hóa dùng base64 và Crypto.Cipher

```
import base64
from Crypto.Cipher import AES
```

Hàm mã hóa

```
def encrypt_aes_256(clear_text, key, iv):
    key_byte = key.encode('utf-8')
    key_byte = key_byte.ljust(32, "\0".encode('utf-8'))
    if len(key_byte) > 32:
        key_byte = key_byte[:32]
    iv_byte = iv.encode('utf-8')
    iv_byte = iv_byte.ljust(16, "\0".encode('utf-8'))
    if len(iv_byte) > 16:
        key_byte = key_byte[:16]

    # PKCS#5
    pad_len = 16 - len(clear_text) % 16
    padding = chr(pad_len) * pad_len
    clear_text += padding

    cryptor = AES.new(key_byte, AES.MODE_CBC, iv_byte)
    data = cryptor.encrypt(clear_text)

    return base64.b64encode(data).decode('utf-8')
```

Hàm giải mã

```
def decrypt_aes_256(data, key, iv):
    data_byte = base64.b64decode(data.encode('utf-8'))
    key_byte = key.encode('utf-8')
    key_byte = key_byte.ljust(32, "\0".encode('utf-8'))
    if len(key_byte) > 32:
        key_byte = key_byte[:32]
    iv_byte = iv.encode('utf-8')
```

```

iv_byte = iv_byte.ljust(16, "\0".encode('utf-8'))
if len(iv_byte) > 16:
    key_byte = key_byte[:16]

cryptor = AES.new(key_byte, AES.MODE_CBC, iv_byte)
c_text = cryptor.decrypt(data_byte)

# PKCS#5
pad_len = ord(c_text.decode('utf-8')[-1])
clear_text = c_text.decode('utf-8')[:-pad_len]

return clear_text

```

Khai báo thư viện bên nhận dạng khuôn mặt

```

from Aes256CBC import *

```

Hàm chuyển đổi message sang mã hóa aes

```

def convertMsgToAes(msg, key, iv):
    encoded = encrypt_aes_256(msg, key, iv)
    return encoded

```

Khóa key có độ dài 32 bytes và IV có độ dài 16 bytes

```

key = 'qwertyuiopasdfghjklzxcvbnm123456'
iv = "caothithuylinh99"

```

Khi đó hàm đóng mở cửa sẽ gọi lại hàm convertMsgToAes để mã hóa dữ liệu đóng mở cửa

```

CMD_OPEN = 'open'
CMD_CLOSE = 'close'

def door_lock(key, iv):
    lock = convertMsgToAes(CMD_CLOSE, key, iv)
    client.publish(topic, lock)

def door_unlock(key, iv):

```

```
unlock = convertMsgToAes(CMD_OPEN, key, iv)
client.publish(topic, unlock)
```

#### **h. Gửi hình ảnh người lạ lên MQTT**

Mã hóa image bằng AES256CBC của mục trên, khi phát hiện ra người lạ, raspberry sẽ gửi các frame ảnh của người lạ lên hệ thống, đồng thời cửa sẽ được đóng lại.

Đầu tiên chuyển đổi frame ảnh thành dữ liệu trực tuyến và lưu trữ trong bộ nhớ đệm của bộ nhớ, tiếp đó chuyển dữ liệu trực tuyến thành kiểu byte. Lúc này mã hóa thì chưa chạy được, nên chuyển đổi tiếp thành kiểu hex() rồi mới mã hóa và gửi đi

```
_, img_encode = cv2.imencode('.jpg', frame)
imgByte = img_encode.tobytes()
img = imgByte.hex()
imgSend = convertMsgToAes(img, key, iv)
client.publish(topic, imgSend)
```

#### **i. Giao diện GUI**

Thư viện thiết kế giao diện

```
import os, glob

import tkinter as tk
from tkinter import font as tkfont
from tkinter import messagebox, PhotoImage
import tkinter.simpledialog as tsd
```

Thư viện chính

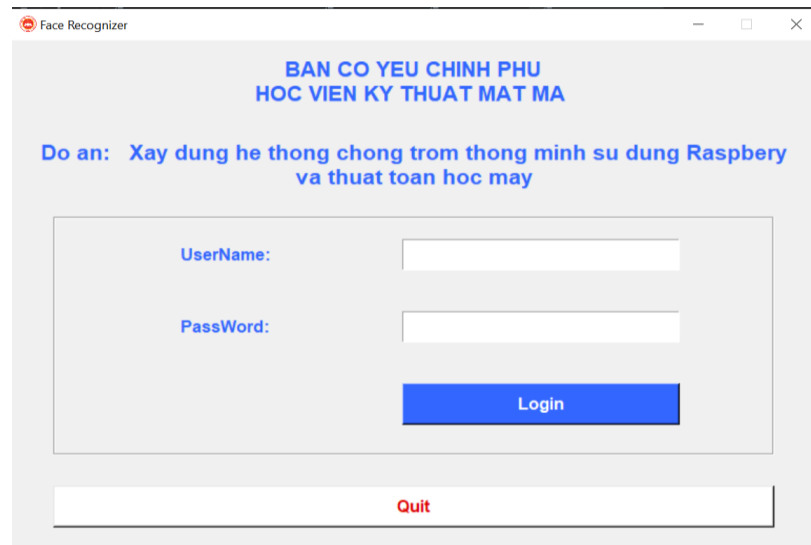
```
from mainMqttPubAes256CBC import *
from getDataset import *
from trainer import *
```

Một số biến cần thiết

```
names = set()
ids = set()
id = 0
re_id = 0
```

```
count = 0
```

**LoginPage:**



Hình 3- 9. Trang đăng nhập của phần mềm ứng dụng

- Hàm login()

```
def login(self):
    uname = self.e1.get()
    passwd = self.e2.get()
    if (uname == "" and passwd == ""):
        messagebox.showinfo("", "Blank Not allowed")

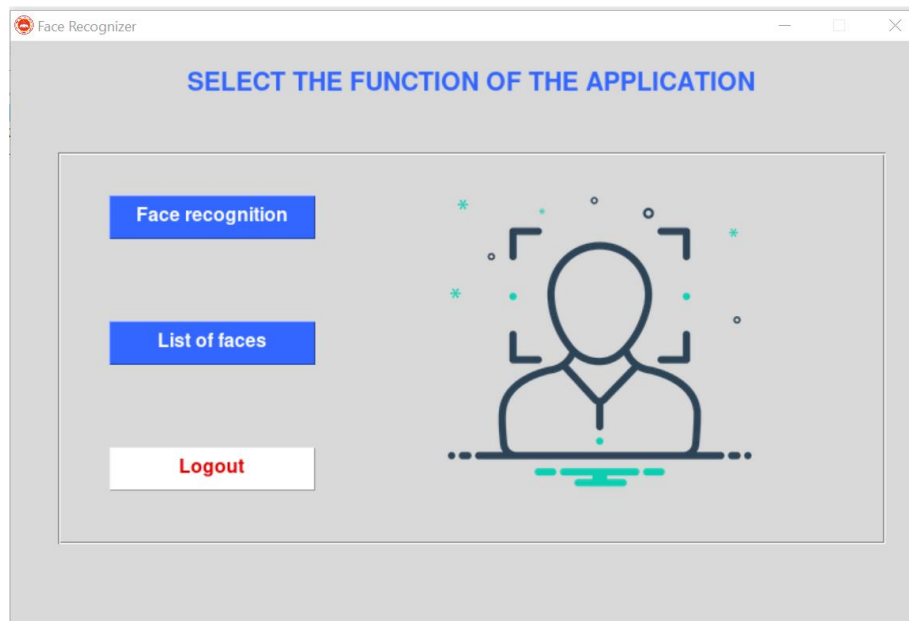
    elif (uname == "admin" and passwd == "admin"):
        self.controller.show_frame("StartPage")

    else:
        messagebox.showinfo("", "Incorrent Username and Password")
```

- Hàm on\_closing() để đóng ứng dụng

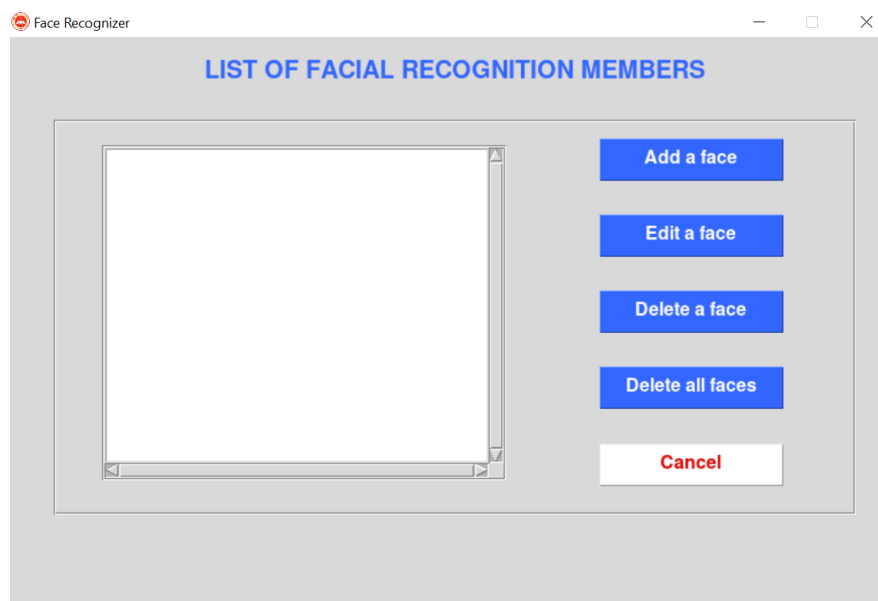
```
def on_closing(self):
    if messagebox.askokcancel("Quit", "Are you sure?"):
        self.controller.destroy()
```

**StartPage:** Trang điều hướng đến các chức năng



Hình 3- 10.Trang điều hướng chức năng của phần mềm ứng dụng

**PageTwo**: page này xuất hiện khi nhấp vào “List a faces” trong startPage



Hình 3- 11. Trang hiển thị danh sách tên các khuôn mặt đã đăng ký

Các chức năng trong page này sau khi được sử dụng sẽ refresh\_names() để cập nhập lại name, id

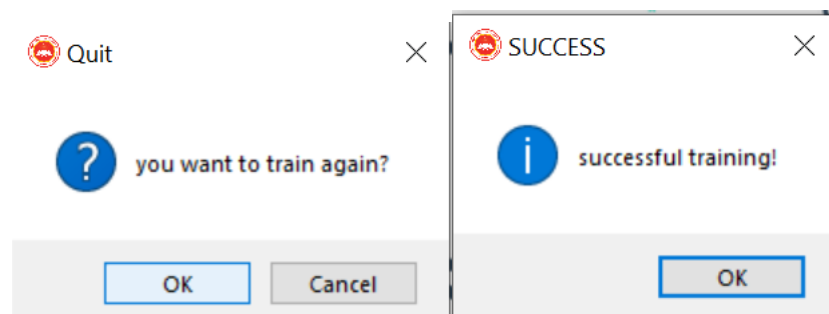
Hàm refresh\_names():

```

def refresh_names(self):#update listbox
    global names, ids
    self.listNames.delete(0, tk.END)
    list_name = []
    f = open("nameslist.txt", "r")
    x = f.read()
    z = x.rstrip().split("\n")
    for i in z:
        smallList = i.rstrip()
        list_name.append(smallList)
    lenght = len(z)
    for l in range(lenght):
        ids.add(l)
    for name in list_name:
        if name == 'None':
            continue
        self.listNames.insert(tk.END, name)
    f.close()

```

Sau khi refresh\_names xong các chức năng sẽ yêu cầu huấn luyện lại bộ dữ liệu dataset. Các dòng lệnh sau được cho vào cuối các hàm chức năng



```

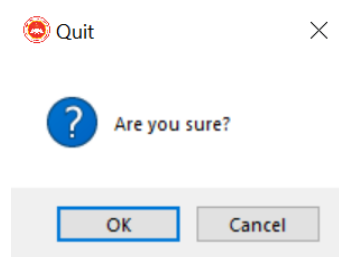
try:
    if messagebox.askyesnocancel("Quit", "you want to train again?"):
        startTrain()
        messagebox.showinfo("SUCCESS", "successful training!")
        self.controller.show_frame("PageTwo")
except:
    pass

```

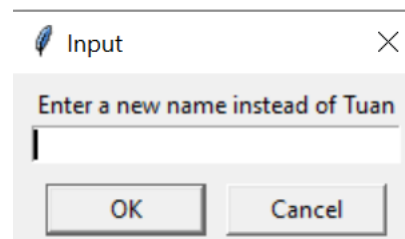


Chức năng “Add a face”: Khi nhấp vào “Add a face”, ứng dụng sẽ dẫn đến pageOne để thêm khuôn mặt mới

Chức năng “Edit a face”: Để sử dụng “Edit a face”, cần chọn tên khuôn mặt cần chỉnh sửa rồi nhấp “Edit a User”, khi đó xuất hiện cửa sổ



Ta nhấp “OK”, khi đó xuất hiện thêm cửa sổ để nhập tên khuôn mặt mới thay cho tên vừa chọn. Tên mới phải đủ điều kiện là không được để trống, trùng với tên đã có.



Hình 3- 12. Cửa sổ thay đổi tên khuôn mặt đã đăng ký

Hàm edit\_name() được sử dụng để xử lý chức năng này

```
def edit_name(self):
    global names, ids
    if messagebox.askokcancel("Quit", "Are you sure?"):
        for line in self.listNames.curselection():
            name_old = self.listNames.get(line)
            name_new = tsd.askstring("Input", "Enter a new name instead of " +
name_old)
            if name_new == None:
                messagebox.showerror("Error", "Name cannot be 'None'")
                return
            elif name_new in names:
                messagebox.showerror("Error", "User already exists!")
                return
            elif name_new == name_old:
                messagebox.showerror("Error", "User already exists!")
```

```

        return
    elif len(name_new) == 0:
        messagebox.showerror("Error", "Name cannot be empty!")
        return
    a_file = open("listNames/nameslist.txt", "r")
    list_of_lines = a_file.readlines()
    if line < self.listNames.size() - 1:
        list_of_lines[line+1] = name_new + "\n"

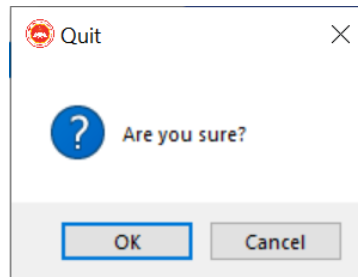
        a_file = open("listNames/nameslist.txt", "w")
        a_file.writelines(list_of_lines)
        a_file.close()
    else:
        list_of_lines[line + 1] = name_new

        a_file = open("listNames/nameslist.txt", "w")
        a_file.writelines(list_of_lines)
        a_file.close()

    self.listNames.delete(line)
    names.remove(name_old)
    names.add(name_new)
    self.listNames.insert(line, name_new)
    self.controller.active_name = name_new
    self.controller.frames["PageTwo"].refresh_names()
    try:
        if messagebox.askokcancel("Quit", "you want to train again?"):
            startTrain()
            messagebox.showinfo("SUCCESS", "successful training!")
            self.controller.show_frame("PageTwo")
    except:
        pass

```

Chức năng “Delete a face”: Để xóa 1 khuôn mặt đã đăng ký trước đó, đầu tiên chọn vào tên khuôn mặt đã có, rồi nhấp vào “Delete a face”. Cửa sổ Quit được hiện



Khấp “OK” thì tên khuôn mặt và dữ liệu hình ảnh về khuôn mặt đó sẽ bị xóa khỏi hệ thống, và id của khuôn mặt sẽ thay đổi phù hợp

Hàm `delete_name()` xử lý chức năng này

```
def delete_name(self):
    if messagebox.askokcancel("Quit", "Are you sure?"):
        global names, ids
        for line in self.listNames.curselection():

            name = self.listNames.get(line)
            p = './datasets/User.' + str(line+1) + '.' + "*" + '.jpg'
            fileDataset = glob.glob(p, recursive=True)
            for f in fileDataset:
                try:
                    os.remove(f)
                except OSError as e:
                    print("Error: %s : %s" % (f, e.strerror))
            if line < self.listNames.size() - 1:
                a_file = open("listNames/nameslist.txt", "r")
                list_of_lines = a_file.readlines()
                list_of_lines[line + 1] = ""

                a_file = open("listNames/nameslist.txt", "w")
                a_file.writelines(list_of_lines)
                a_file.close()
            else:
                a_file = open("listNames/nameslist.txt", "r")
                readFile = a_file.read()
                a_file.close()
                m = readFile.split("\n")
                lines = "\n".join(m[:-1])
                a_file = open("listNames/nameslist.txt", "w+")
```

```

        for i in range(len(lines)):
            a_file.write(lines[i])
        a_file.close()

    names.remove(name)
    ids.remove(line + 1)
    self.listNames.delete(line)

    listIds = sorted(ids)#chuyen sang lisst
    #chay tat ca cac file: neu count 2 file canh nhau la cach nhau thi
rename
    global count
    for i in listIds:
        if i > line + 1:
            path_old = './datasets/User.' + str(i) + '.' + "*" + '.jpg'
            for file_name_old in glob.glob(path_old):
                p_old = file_name_old
                p_new = './datasets/User.' + str(int(i) - 1) + '.' + str(count) +
'.jpg'

                count += 1
                os.rename(p_old, p_new)
                if count == 200:
                    count = 0

    self.controller.frames["PageTwo"].refresh_names()

    try:
        if messagebox.askokcancel("Quit", "you want to train again?"):
            startTrain()
            messagebox.showinfo("SUCCESS", "successful training!")
            self.controller.show_frame("PageTwo")
    except:
        pass

```

Chức năng “Delete all faces”: Khi nhấp vào chức năng này, tất cả các khuôn mặt trong hệ thống sẽ bị xóa hết, dữ liệu cũng bị xóa hết nếu cửa sổ hiện ra mà chọn “OK”

Hàm delete\_all\_name() sẽ thực hiện chức năng này

```

def delete_name(self):
    if messagebox.askokcancel("Quit", "Are you sure?"):
        global names, ids
        for line in self.listNames.curselection():

            name = self.listNames.get(line)
            p = './dataset/User.' + str(line+1) + '.' + "*" + '.jpg'
            fileDataset = glob.glob(p, recursive=True)
            for f in fileDataset:
                try:
                    os.remove(f)
                except OSError as e:
                    print("Error: %s : %s" % (f, e.strerror))
            a_file = open("nameslist.txt", "r")
            list_of_lines = a_file.readlines()
            list_of_lines[line+1] = ""

            a_file = open("nameslist.txt", "w")
            a_file.writelines(list_of_lines)
            a_file.close()
            names.remove(name)
            ids.remove(line + 1)
            self.listNames.delete(line)

            listIds = sorted(ids)#chuyen sang lisst
            #chay tat ca cac file: neu count 2 file canh nhau là cach nhau thì
rename
            global re_id
            for i in listIds:
                leng = len(listIds[0:re_id])
                re_id = re_id + 1
                path_old = './dataset/User.' + str(i) + '.' + "*" + '.jpg'
                for file_name_old in glob.glob(path_old):
                    p_old = file_name_old

                    if i != str(leng):
                        global count
                        p_new = './dataset/User.' + str(int(i) - 1) + '.' + str(count) +
'.jpg'
                        count += 1

```

```

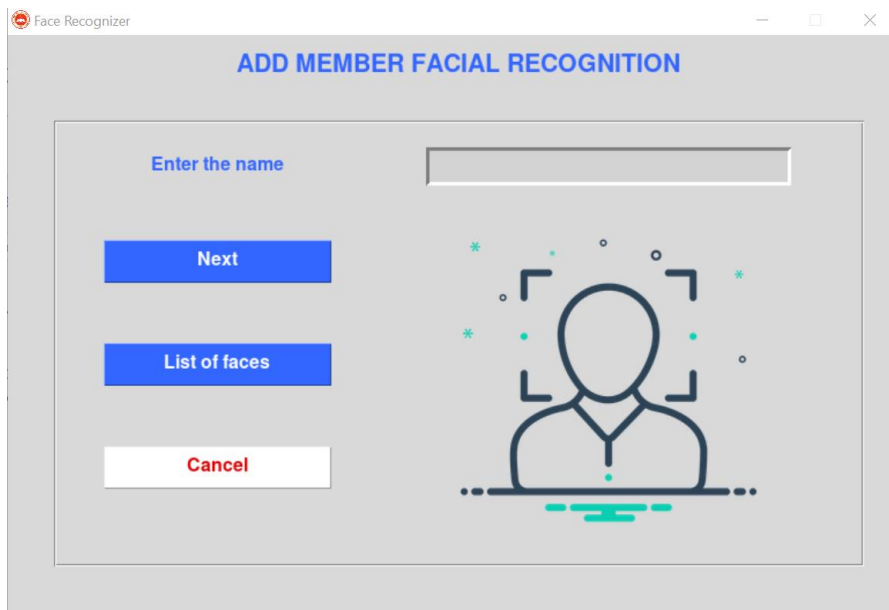
os.rename(p_old, p_new)
if count == 99:
    return count == 0

self.controller.frames["PageTwo"].refresh_names()

try:
    if messagebox.askyesnocancel("Quit", "you want to train again?"):
        startTrain()
        messagebox.showinfo("SUCCESS", "successful training!")
        self.controller.show_frame("PageTwo")
except:
    pass

```

**PageOne**: page thêm khuôn mặt mới sau khi nhấp “Add a face”



*Hình 3- 13. Trang thêm khuôn mặt mới của phần mềm ứng dụng*

Khi nhập tên mới, và nhấp “Next” thì ứng dụng sẽ gọi hàm `start_training()`. Hàm này kiểm tra xem tên đó đã có trước đó hay chưa trong file “`namelist.txt`”, nếu có thì ứng dụng báo lỗi và sẽ phải nhập lại tên mới chưa có sẵn và lưu vào file “`namelist.txt`” rồi dẫn đến page để thu thập ảnh để lưu vào dataset

- Hàm `start_training()`

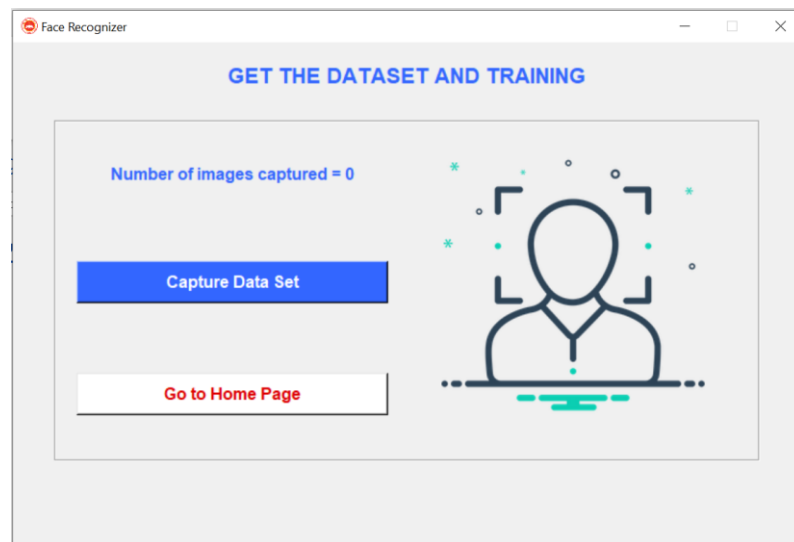
```
def start_training(self):
```

```

global names, ids, id
if self.user_name.get() == "None":
    messagebox.showerror("Error", "Name cannot be 'None'")
    return
elif self.user_name.get() in names:
    messagebox.showerror("Error", "User already exists!")
    return
elif len(self.user_name.get()) == 0:
    messagebox.showerror("Error", "Name cannot be empty!")
    return
name = self.user_name.get()
names.add(name)
self.controller.active_name = name
id = id + 1
ids.add(id)
f = open("nameslist.txt", "a+")
f.write("\n" + self.controller.active_name)
f.close()
self.controller.frames["PageTwo"].refresh_names()
self.controller.show_frame("PageThree")

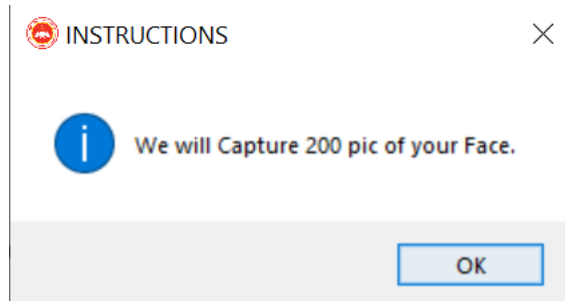
```

**PageThree:** page sau khi thêm khuôn mặt mới và ấn “Next”



Hình 3- 14. Trang thu thập dữ liệu trong phần mềm ứng dụng

- Khi nhấp vào “Capture Data Set”, ứng dụng hiển thị thông báo và gọi đến hàm `capimg()`, trong hàm này sẽ đếm số frame ảnh và trở đến hàm `start_dataset()` trong file `getDataset` để thu thập dữ liệu



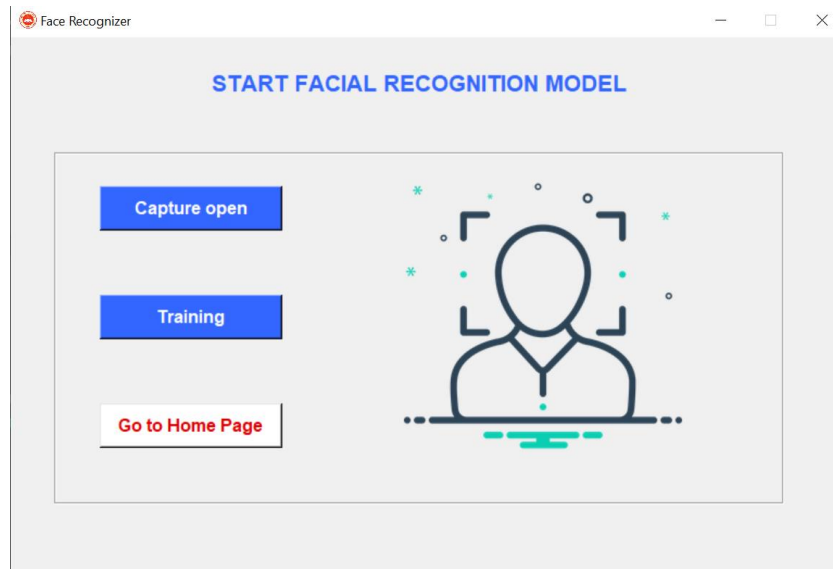
```
def capimg(self):
    global id
    self.numimlabel.config(text=str("Number of images captured = 0"))
    messagebox.showinfo("INSTRUCTIONS", "We will Capture 100 pic of
your Face.")
    x = start_dataset(id, self.controller.active_name)
    self.controller.num_of_images = x
    self.numimlabel.config(text=str("Number of images captured =
"+str(x)))
    try:
        if messagebox.askyesnocancel("Quit", "you want to train again?"):
            startTrain()
            messagebox.showinfo("SUCCESS", "successful training!")
            self.controller.show_frame("PageFour")
    except:
        pass
```

- Khi nhấp vào “training”, sẽ gọi đến hàm `trainmodel()`, trong hàm này sẽ trở đến `startTrain()` trong file `trainer` để huấn luyện dataset

```
def trainmodel(self):
    if self.controller.num_of_images < 200:
        messagebox.showerror("ERROR", "No enough Data, Capture at least
100 images!")
        return
    startTrain()
    messagebox.showinfo("SUCCESS", "successful training!")
    self.controller.show_frame("PageFour")
```



**PageFour**: sau khi train xong thì bắt đầu nhận dạng khuôn mặt, page này cũng xuất hiện khi nhấp “Face Recognition” của startPage



Hình 3- 15. Trang nhận dạng khuôn mặt trong phần mềm ứng dụng

- Khi chọn “Face Recognition”, ứng dụng sẽ gọi đến hàm openwebcam(), hàm này trỏ đến hàm startFaceRecongition() trong file mainMqttPubAes256CBC để bắt đầu nhận dạng khuôn mặt
- Khi chọn “Train”, ứng dụng gọi đến hàm training và huấn luyện khuôn mặt

### 3.1.3.3. Chương trình thực thi đóng/mở cửa

#### a. Kết nối MQTT

Thư viện

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Servo.h>
```

Khai báo các biến cho servo, buttons, vị trí pos

```
Servo myservo;
int pos = 0;
int buttonState = 0;
int directionState = 0;
```

```
#define servo D2  
#define buttonPin D3
```

Chỉnh sửa mã để phù hợp với cài đặt WiFi và MQTT

```
const char* ssid = "Meo Meo";  
const char* password = "mangcut9987";
```

Thiết lập địa chỉ máy chủ MQTT, port kết nối, username, password, topic, clientID

```
const char* mqtt_server = "192.168.0.103";  
const int mqtt_port = 1883;  
const char *topic = "testServo";  
String clientId = "mqtt-servo";  
  
const char* mqttUser = "linh99";  
const char* mqttPassword = "1234567890";
```

Sau đó, sẽ khai báo một đối tượng của lớp WiFiClient, cho phép thiết lập kết nối với một IP và cổng cụ thể. Tuyên bố một đối tượng của lớp PubSubClient, nhận được như là đầu vào của người xây dựng WiFiClient được xác định trước đó.

```
WiFiClient espClient;  
PubSubClient client(espClient);
```

Hàm kết nối wifi

```
void setup_wifi() {  
  delay(10);  
  Serial.println();  
  Serial.print("Connecting to ");  
  Serial.println(ssid);  
  
  WiFi.mode(WIFI_STA);  
  WiFi.begin(ssid, password);  
  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }
```

```

}

randomSeed(micros());

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

```

Tiếp theo, chỉ định địa chỉ và cổng của máy chủ MQTT. Gọi `setServer` trên đối tượng `PubSubClient`

```
client.setServer(mqtt_server, mqtt_port);
```

Sau đó dùng `setCallback` trên cùng 1 đối tượng để chỉ định chức năng xử lý được thực hiện khi nhận được tin nhắn MQTT

```
client.setCallback(callback);
```

Để thực hiện kết nối thực tế, cần gọi phương thức `connect`, nhận được dưới dạng nhậm mã định danh duy nhất của client, gọi là “ESP8266Client”, và username, password đã xác định. Nếu trường hợp thất bại, gọi phương thức “state”

Để hoàn tất chức năng thiết lập, chỉ cần publish một thông báo nội dung cho topic về máy chủ. Ví dụ: `client.publish(topic, “hello”);` Sau đó, sẽ đăng ký với topic: `client.subscribe(topic);`

```

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqttUser, mqttPassword)) {
      Serial.println("connected");
      client.subscribe(topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
    }
  }
}

```

```

    delay(5000);
  }
}

```

Hàm callback sẽ xử lý các tin nhắn đến cho topic đã đăng ký. Các đối số của hàm này là topic, payload(tính theo byte) và length

Trong chức năng này, trước tiên sẽ in topic và cổng nối tiếp, sau đó in từng byte của tin nhắn nhận được

```

void callback(char* topic, byte* payload, unsigned int length) {
  char messages[100];
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("]");

  for (int i = 0; i < length; i++)
  {
    char receivedChar = (char)payload[i];

    messages[i] = receivedChar;
  }
}

```

Hàm lặp chính

```

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  thuCong();
}

```

## b. Bảo mật SSL/TLS

Thêm thư viện

```
#include <WiFiClientSecure.h>
```

Mở và đọc file ca, key dùng SPIFFS

```
File ca_cert_str = SPIFFS.open("./certs/mqtt_ca.crt", "r");  
File client_cert_str = SPIFFS.open("./certs/mqtt_client.crt", "r");  
File client_key_str = SPIFFS.open("./certs/mqtt_client.key", "r");
```

Khi đó thay đổi tượng của WiFiClient thành BearSSL::WiFiClientSecure

Thư viện BearSSL được sử dụng để thực hiện tất cả các hoạt động mã hóa và TLS.

```
BearSSL::WiFiClientSecure espClient;  
PubSubClient client(espClient);
```

Chúng chỉ X509 được sử dụng để xác định các client kết nối TLS. Bất kỳ cuộc gọi nào lấy chứng chỉ X509 cũng có thể lấy danh sách chứng chỉ X509, vì vậy không có lớp X509 đặc biệt, chỉ cần BearSSL::X509List(chỉ chứa 1 chứng chỉ duy nhất). Trong bài này lấy 3 chứng chỉ và key cần thiết sau

```
BearSSL::X509List ca_cert(ca_cert_str);  
BearSSL::X509List client_cert(client_cert_str);  
BearSSL::PrivateKey client_key(client_key_str);
```

Thêm các xác thực chứng chỉ trong hàm setup()

- setTrustAnchors: Tải chứng chỉ CA vào nơi lưu trữ tin cậy, chấp nhận chứng chỉ từ xa được ký bởi bất kỳ chứng chỉ nào trong số này
- allowSelfSignedCerts và setInsecure: Không có nó, thư viện WiFiClientSecure sẽ liên tục thất bại trong việc kết nối mà không xuất hiện lỗi hữu ích
- setClientRSACert: đặt chứng chỉ client để gửi đến máy chủ TLS yêu cầu nên nó được gọi trước khi connect để thêm chứng chỉ cho client trong trường hợp máy chủ yêu cầu(bao gồm chứng chỉ và key)

```
espClient.setTrustAnchors(&ca_cert);  
espClient.allowSelfSignedCerts();  
espClient.setClientRSACert(&client_cert, &client_key);  
espClient.setInsecure();
```

### c. Mã hóa AES

Thư viện

```
#include "Base64.h"
```

Khóa key có độ dài 32 bytes và IV có độ dài 16 bytes

```
byte cipher_key[] = "qwertyuiopasdfghjklzxcvbnm123456";  
byte cipher_iv[] = "caothithuylinh99";
```

Hàm mã hóa

```
String encrypt(String plain_data){  
    int i;  
    // PKCS#7 Padding (Encryption), Block Size : 16  
    int len = plain_data.length();  
    int n_blocks = len / 16 + 1;  
    uint8_t n_padding = n_blocks * 16 - len;  
    uint8_t data[n_blocks*16];  
    memcpy(data, plain_data.c_str(), len);  
    for(i = len; i < n_blocks * 16; i++){  
        data[i] = n_padding;  
    }  
  
    uint8_t key[32], iv[16];  
    memcpy(key, cipher_key, 32);  
    memcpy(iv, cipher_iv, 16);  
  
    // encryption context  
    br_aes_big_cbcenc_keys encCtx;  
  
    // reset the encryption context and encrypt the data  
    br_aes_big_cbcenc_init(&encCtx, key, 32);  
    br_aes_big_cbcenc_run( &encCtx, iv, data, n_blocks*16 );  
  
    // Base64 encode  
    len = n_blocks*16;  
    char encoded_data[ base64_enc_len(len) ];  
    base64_encode(encoded_data, (char *)data, len);
```

```
return String(encoded_data);  
}
```

#### Hàm giải mã

```
String decrypt(String encoded_data_str){  
    int input_len = encoded_data_str.length();  
    char *encoded_data = const_cast<char*>(encoded_data_str.c_str());  
    int len = base64_dec_len(encoded_data, input_len);  
    uint8_t data[ len ];  
    base64_decode((char *)data, encoded_data, input_len);  
  
    uint8_t key[32], iv[16];  
    memcpy(key, cipher_key, 32);  
    memcpy(iv, cipher_iv, 16);  
  
    int n_blocks = len / 16;  
  
    br_aes_big_cbcdec_keys decCtx;  
  
    br_aes_big_cbcdec_init(&decCtx, key, 32);  
    br_aes_big_cbcdec_run( &decCtx, iv, data, n_blocks*16 );  
  
    // PKCS#7 Padding (Decryption)  
    uint8_t n_padding = data[n_blocks*16-1];  
    len = n_blocks*16 - n_padding;  
    char plain_data[len + 1];  
    memcpy(plain_data, data, len);  
    plain_data[len] = '\0';  
  
    return String(plain_data);  
}
```

#### d. Gửi, nhận dữ liệu lên MQTT

Khai báo 1 số biến sau

```
const char *CMD_OPEN = "open";  
const char *CMD_CLOSE = "close";
```

Trong hàm callback, sau khi nhận được dữ liệu thì cần giải mã dữ liệu rồi mới so sánh dữ liệu với các biến open, close để gọi đến các hàm đóng, mở cửa. Sau khi tin nhắn open thì cửa sẽ mở cửa và để thời gian là 30 giây, sau đó sẽ tự động đóng cửa

```
void callback(char* topic, byte* payload, unsigned int length) {
    char messages[100];
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("]");

    for (int i = 0; i < length; i++)
    {
        char receivedChar = (char)payload[i];

        messages[i] = receivedChar;
    }

    String decdata = decrypt(messages);
    Serial.println(decdata);

    Serial.println();
    if (strcmp((char*)decdata.c_str(),CMD_OPEN) == 0 && directionState == 0) {
        directionState = 1;
        openDoor();
        delay(30000);//30giay
        directionState = 0;
        closeDoor();

    }
    else if (strcmp((char*)decdata.c_str(),CMD_CLOSE) == 0 && directionState == 1) {
        directionState = 0;
        closeDoor();
    }
}
```

Và các hàm đóng, mở cửa thì sẽ cần mã hóa dữ liệu trạng thái để gửi lên MQTT



```

void closeDoor(){
    String msg = "closeDoor";
    for (pos = 0; pos <= 135; pos += 1) {
        myservo.write(pos);
        delay(15);
    }

    String encdata = encrypt(msg);

    client.publish(topic, (char*)encdata.c_str());
    client.subscribe(topic);
}

void openDoor(){
    String msg = "openDoor";
    for (pos = 135; pos >= 0; pos -= 1) {
        myservo.write(pos);
        delay(15);
    }
    String encdata = encrypt(msg);

    client.publish(topic, (char*)encdata.c_str());
    client.subscribe(topic);
}

```

Thêm chức năng đóng mở cửa bằng button

```

void thuCong(){
    buttonState = digitalRead(buttonPin);
    if (directionState == 0){
        if (buttonState == HIGH) {
            directionState = 1;
            openDoor();
        }

    } else if (directionState == 1) {
        if (buttonState == HIGH) {
            directionState = 0;
            closeDoor();
        }
    }
}

```

```
}  
}  
}
```

#### 3.1.3.4. Chương trình đưa ra cảnh báo và hiển thị ảnh người lạ

##### a. Kết nối MQTT

Trong file *pubspec.yaml*: Cài các package cần thiết

```
mqtt_client: ^6.2.0
```

Khai báo thư viện

```
import 'package:mqtt_client/mqtt_client.dart';  
import 'package:mqtt_client/mqtt_server_client.dart';
```

Sau khi tạo giao diện nhập các thông tin cần thiết của MQTT broker, các thông tin này được lấy để config và connect

Hoạt động đồng bộ *Future \_configureAndConnect() async* để lấy thông tin cần thiết của MQTT để bắt đầu bước kết nối

```
import 'package:smarthome/provider/mqtt/mqtt_service.dart';  
  
MQTTService manager;  
  
bool _isLoading = true;  
bool _isError = false;  
  
Future _configureAndConnect() async {  
  try {  
    manager = MQTTService(  
      host: widget.device.mqttBroker,  
      port: widget.device.port,  
      topic: widget.device.topic,  
      username: widget.device.userName,  
      password: widget.device.password,  
      clientId: widget.device.clientID);
```

```

manager.initMQTT();
manager.connectMQTT();
setState() {
    _isLoading = false;
    _isError = false;
});
} catch (e) {
    print(e);
    setState() {
        _isLoading = false;
        _isError = true;
    });
}

```

Dòng *manager.initMQTT()*; trở đến hoạt động đồng bộ *initMQTT()*

```

final int _keepAlivePeriod = 20;
final bool _autoReconnect = true;

MqttServerClient _client;

MqttServerClient get client => _client;

final String clientId;

final String host;
final int port;
final String topic;
final String username;
final String password;

MQTTService(
    {this.clientId,
    this.host,
    this.topic,
    this.username,
    this.password,
    this.port});

```

```

Future initMQTT() async {
  print('host: $host --- clientId: $clientId -- port: $port');
  _client = MqttServerClient.withPort(host, clientId, port);
  _client.keepAlivePeriod = _keepAlivePeriod;
  _client.onDisconnected = _onDisconnected;
  _client.secure = true;
  _client.logging(on: true);

  _client.onConnected = _onConnected;
  _client.onSubscribed = _onSubscribed;
  _client.onUnsubscribed = _onUnsubscribed;
  _client.pongCallback = _pong;
  final MqttConnectMessage connMess = MqttConnectMessage()
    .authenticateAs(username, password)
    .withWillTopic('will topic') // If you set this you must set a will message
    .withWillMessage('Test message')
    .startClean() // Non persistent session for testing
    .withWillQos(MqttQos.atMostOnce);
  print('EXAMPLE::Mosquitto client connecting....');
  _client.connectionMessage = connMess;
}

```

Dòng *manager.connectMQTT()*; trỏ đến connectMQTT()

```

connectMQTT() async {
  assert(_client != null);
  try {
    await _client.connect();
  } on Exception catch (e) {
    _client.disconnect();
    throw Exception('Client MQTT exception - $e');
  }
}

```

#### b. Bảo mật SSL/TLS

Trong file *pubspec.yaml*: Đầu tiên, sẽ cần đảm bảo rằng các tệp cần thiết được xác định

```
flutter:
```

```
assets:  
  - assets/certs_localhost/
```

Thư viện cần thiết

```
import 'dart:async';  
import 'dart:io';
```

Load các file ca, key. Khi các tệp được đảm bảo, điều này sẽ đảm bảo rằng các tệp được đóng gói với ứng dụng để có thể tải chúng bằng cách sử dụng rootBundle

```
String clientAuth = await  
rootBundle.loadString("assets/certs_localhost/mqtt_ca.crt");  
String trustedCer = await  
rootBundle.loadString("assets/certs_localhost/mqtt_client.crt");  
String privateKey = await  
rootBundle.loadString("assets/certs_localhost/mqtt_client.key");
```

Tiếp theo, và chuyển dữ liệu đến SecurityContext như byte

```
_client.secure = true;  
_client.logging(on: true);  
final context = SecurityContext.defaultContext;  
context.setTrustedCertificatesBytes(clientAuth.codeUnits);  
context.useCertificateChainBytes(trustedCer.codeUnits);  
context.usePrivateKeyBytes(privateKey.codeUnits);
```

### c. Mã hóa AES

Trong file pubspec.yaml: cần cài package cần thiết cho mã hóa aes

```
dependencies:  
  encrypt: ^4.1.0  
  typed_data: ^1.1.6
```

Thư viện

```
import 'dart:convert';
```

```
import 'package:encrypt/encrypt.dart';
import 'package:flustars/flustars.dart';
```

Khóa key có độ dài 32 bytes và IV có độ dài 16 bytes

```
var _KEY = "qwertyuiopasdfghjklzxcvbnm123456";
var _IV = "caothithuylinh99";
```

Hàm mã hóa

```
static aesEncrypt(plainText) {
  try {
    final key = Key.fromUtf8(_KEY);
    final iv = IV.fromUtf8(_IV);
    final encrypter = Encrypter(AES(key, mode: AESMode.cbc));
    final encrypted = encrypter.encrypt(plainText, iv: iv);
    return encrypted.base64;
  } catch (err) {
    print("aes encode error:$err");
    return plainText;
  }
}
```

Hàm giải mã

```
static dynamic aesDecrypt(encrypted) {
  try {
    final key = Key.fromUtf8(_KEY);
    final iv = IV.fromUtf8(_IV);
    final encrypter = Encrypter(AES(key, mode: AESMode.cbc));
    final decrypted = encrypter.decrypt64(encrypted, iv: iv);
    return decrypted;
  } catch (err) {
    print("aes decode error:$err");
    return encrypted;
  }
}
```

Khai báo thư viện trong hàm xử lý

```
import 'package:test_mqtt_ssl_tls/crypt/crypt.dart';
```

Gọi đến hàm mã hóa và giải mã thì chỉ cần gọi

```
var msgEncode = crypt.aesEncrypt(message);  
var msgDecode = crypt.aesDecrypt(text);
```

#### **d. Chức năng đóng/ mở cửa**

##### **Bảng nút bấm**

Hàm gửi tin nhắn sendMessage, trong hàm này có mã hóa dữ liệu AES 256 cbc và gửi dữ liệu lên MQTT qua *\_client.publishMessage*

```
sendMessage(String message) {  
  try {  
    final builder = MqttClientPayloadBuilder();  
    var msgEncode = crypt.aesEncrypt(message); //encoder  
    builder.addString(msgEncode);  
    _client.publishMessage(topic, MqttQos.exactlyOnce, builder.payload);  
  } catch (e) {  
    throw Exception(e.toString());  
  }  
}
```

Bên tệp giao diện có nút button cần có hàm \_remote để trỏ đến hàm sendMessage

```
void _remote(String command) {  
  widget.mqttService.sendMessage(command);  
}
```

Thêm cử chỉ onTap để chạm vào màn hình, onTap gọi lại hàm \_remote để đưa giá trị tương ứng với chức năng của từng nút

```
@override  
Widget build(BuildContext context) {  
  return Center(  
    child: Row(  

```

```

mainAxisAlignment: MainAxisAlignment.spaceAround,
children: [
  GestureDetector(
    behavior: HitTestBehavior.translucent,
    onTap: () => _remote('open'),
    child: CircleAvatar(
      backgroundColor: _isOn ? primary : Colors.grey,
      radius: 40,
      child: Icon(
        Icons.lock_open,
        size: 32,
        color: Colors.white,
      ),
    ),
  ),
  GestureDetector(
    behavior: HitTestBehavior.translucent,
    onTap: () => _remote('close'),
    child: CircleAvatar(
      backgroundColor: _isOn ? Colors.red : Colors.grey,
      radius: 40,
      child: Icon(
        Icons.lock,
        size: 32,
        color: Colors.white,
      ),
    ),
  ),
],
);
}

```

### **Bảng giọng nói :**

Khai báo các biến

```

static const String wakeup = "ok sunday";
static const String open= "open";
static const String close= "close";

```



```
static const String cancel= "cancel";
```

Bên tệp giao diện có giọng nói cần có hàm `_remoteDevice` để trỏ đến hàm `sendMessage`

```
void _remoteDevice (String command) {  
    widget.mqttService.sendMessage(command);  
}
```

Phần xử lý của điều khiển giọng nói

```
String text = "Nói 'Ok Sunday'";  
String keyFlare = FlareConstants.keyStand;  
bool isEnabled = false;  
  
@override  
void initState() {  
    // TODO: implement initState  
    super.initState();  
    VoiceControllerProvider.wakeupStreamChannel  
        .receiveBroadcastStream()  
        .listen((data) {  
            print(data);  
            if (data.toString().contains("WAKEUP")) {  
                setState() {  
                    text = "Đang nghe...";  
                    keyFlare = FlareConstants.keyThink;  
                };  
            } else if (data.toString().contains("LISTENING")) {  
                setState() {  
                    text = "Nói '${VoskConstants.wakeup}'";  
                    keyFlare = FlareConstants.keyStand;  
                };  
            } else {  
                final action = data.toString();  
                setState() {  
                    // Xét điều khiển  
                    if (action.contains(VoskConstants.open)) {  
                        _remoteDevice(ControlRemoteConstants.open);  
                    }  
                };  
            }  
        });  
}
```

```

        text = VoskConstants.open;
    } else if (action.contains(VoskConstants.close)) {
        _remoteDevice(ControlRemoteConstants.close);
        text = VoskConstants.close;
    } else {
        //Hủy bỏ hành động
        if (action.contains(VoskConstants.cancel)) {
            text = "Hủy hành động";
        } else {
            text = "Tôi không hiểu";
        }
    }
    keyFlare = FlareConstants.keyOkay;
});
}
}).onError((err) {
    setState(() {
        text = "Nhận diện giọng nói đã tắt";
    });
});
});

```

#### e. **Hiển thị hình ảnh người lạ và đưa ra cảnh báo**

##### **Hiển thị hình ảnh người lạ và lưu về điện thoại**

Theo phần “Gửi hình ảnh người lạ lên MQTT” ở trên, thì ở phần mobile thì làm ngược lại để lấy được hình ảnh

Đầu tiên giải mã dữ liệu hình ảnh(dữ liệu hình ảnh là dữ liệu có độ dài lớn hơn 24 ký tự- lý do là các dữ liệu điều khiển đóng/ mở cửa sau khi mã hóa aes256 cbc trả về 24 ký tự), khi đó trả về dữ liệu dạng hex, từ dữ liệu dạng hex, cần chuyển đổi về dạng byte. Muốn làm được như vậy cần có hàm chuyển đổi trong file hex2byte.dart

```

import 'package:convert/convert.dart';
import 'dart:convert';
import 'dart:typed_data';

class ByteUtils {
    static List<String> hexArray = '0123456789ABCDEF'.split("");

    static Uint8List hexToBytes(String hexStr) {

```

```

final bytes = hex.decode(strip0x(hexStr));
if (bytes is Uint8List) return bytes;

return Uint8List.fromList(bytes);
}

static String strip0x(String hex) {
  if (hex.startsWith('0x')) return hex.substring(2);
  return hex;
}
}

```

Trong file *pubspec.yaml*: cần cài package cần thiết

```

dependencies:
  path_provider:

```

Hàm lấy vị trí kho dữ liệu được hàm xử lý ở trong file *utils.dart*. Ở đây, các tệp hình ảnh được lưu vào `'/storage/emulated/0/Download'`

Khai báo:

```

import 'package:flutter/material.dart';
import 'package:smarthome/provider/helpers/crypt.dart';
import 'package:smarthome/provider/helpers/hex2byte.dart';

```

```

static Future<String> getDownloadPath() async {
  Directory directory;
  try {
    if (Platform.isIOS) {
      directory = await getApplicationDocumentsDirectory();
    } else {
      directory = Directory('/storage/emulated/0/Download');
      // Put file in global download folder, if for an unknown reason it didn't
      exist, we fallback
      // ignore: avoid_slow_async_io
      if (!await directory.exists())
        directory = await getExternalStorageDirectory();
    }
  }
}

```

```

    } catch (err, stack) {
        print("Cannot get download folder path");
    }
    return directory?.path;
}

```

Hàm lưu ảnh, các tệp ảnh được lưu dưới tên gồm thời gian xuất hiện người lạ

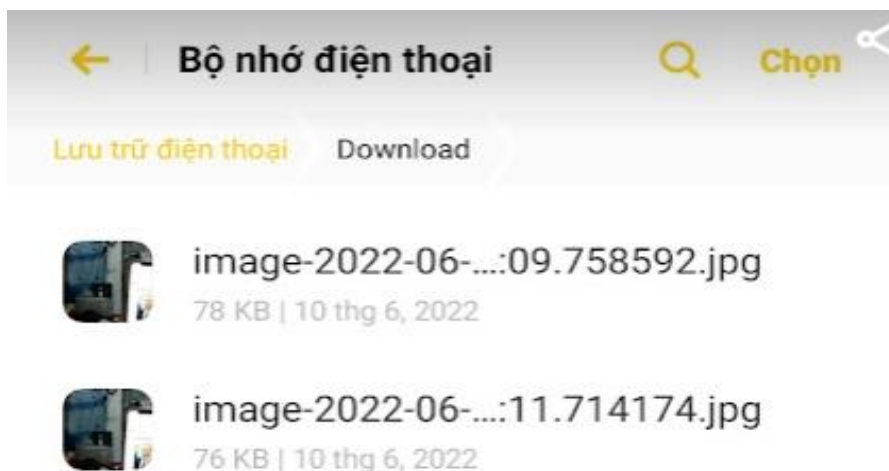
```

static Future<Image> saveImageToStorage(String aesString) async {
    final now = DateTime.now();
    final imgHex = crypt.aesDecrypt(aesString);
    final imgByte = ByteUtils.hexToBytes(imgHex);

    final downloadPath = await getDownloadPath();
    File pathFile = File('$downloadPath/image-${now.toIso8601String()}.jpg');
    final ImgSave = await pathFile.writeAsBytes(imgByte);
    return Image.file(ImgSave);
}

```

Kết quả hình ảnh lưu thành công



*Hình 3- 16. Kết quả lưu hình ảnh về mobile*

Phần hiển thị hình ảnh trên mobile

```

widget.mqttService.client.updates
    ?.listen((List<MqttReceivedMessage<MqttMessage>> c) {
        final MqttPublishMessage recMess = c[0].payload as
        MqttPublishMessage;
    }
)

```

```

final topic = c[0].topic;
final payload =
MqttPublishPayload.bytesToStringAsString(recMess.payload.message);

if (topic == widget.device.topic && payload.length > 24) {
  final img = Utils. saveImageToStorage(payload);
  setState(() {
    _image = img;
  });
}

```

### **Đưa ra cảnh báo**

Trong file *pubspec.yaml*: cần cài package cần thiết

```

dependencies:
  flutter_local_notifications: ^1.4.4+5 # config iOS

```

Khai báo thư viện

```

import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'receive_notification_entity.dart';

```

Đầu tiên thêm hàm nhận đối tượng tin nhắn *ReceiveNotificationEntity*

```

class ReceiveNotificationEntity {
  int id;
  final String title;
  final String body;
  final String payload;

  ReceiveNotificationEntity(
    {int id, @required this.title, @required this.body, this.payload}) {
    final now = DateTime.now();
    if (id == null) this.id = now.second + now.millisecond;
  }
}

```

Hàm *initLocalNotification* bắt đầu đưa ra thông báo cục bộ trên thiết bị mobile

```
Future<void> initLocalNotification(  
  { @required  
    Future<dynamic> onDidReceiveLocalNotification(  
      int id, String title, String body, String payload),  
    @required  
    Future<dynamic> selectNotification(String payload),  
    @required  
    Future<dynamic> onLaunchAppByNotification(String payload) });
```

Hàm *showNotification* để hiển thị thông báo lên màn hình

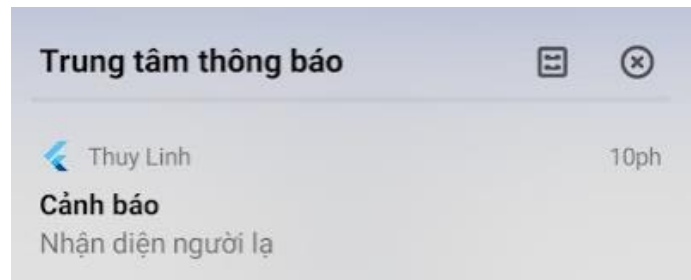
```
@override  
Future<void> showNotification(ReceiveNotificationEntity entity,  
  NotificationDetails notificationDetail) async {  
  await flutterLocalNotificationsPlugin.show(  
    entity.id,  
    entity.title,  
    entity.body,  
    notificationDetail,  
    payload: entity.payload,  
  );  
}
```

Bên phần giao diện gọi lại hàm *showNotification* trên

```
void _showNotify() {  
  const NotificationDetails platformChannelSpecifics = NotificationDetails(  
    AndroidNotificationDetails('channel-id', 'channel name', 'channel',  
      importance: Importance.Max,  
      priority: Priority.Max,  
      ticker: 'ticker'),  
    IOSNotificationDetails());  
  
  locator<LocalNotifyHelper>().showNotification(  
    ReceiveNotificationEntity(title: 'Cảnh báo', body: 'Nhận diện người lạ'),  
    platformChannelSpecifics);  
}
```

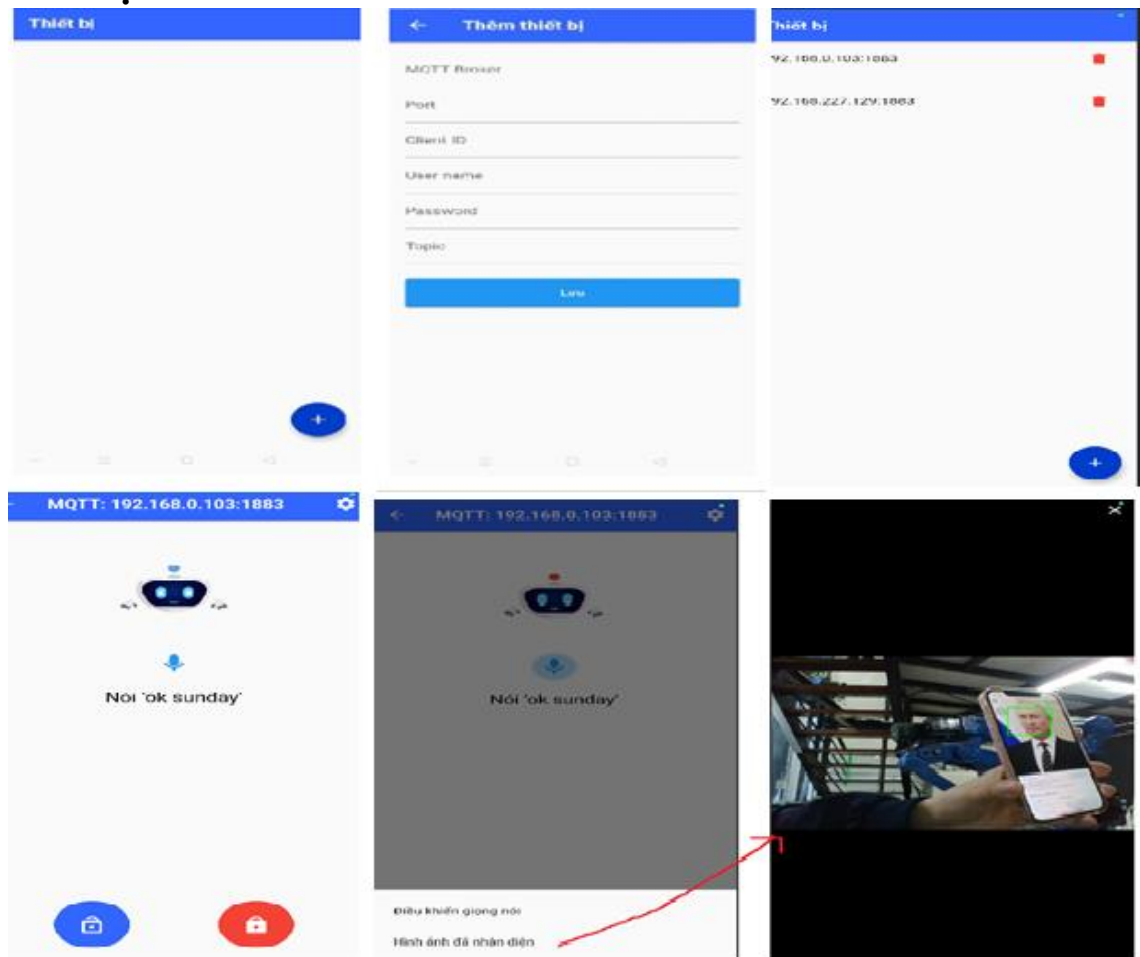
Và sau đó chỉ cần gọi `_showNotify()`;

Kết quả khi có người lạ thì điện thoại đưa ra cảnh báo



Hình 3- 17. Hiện cảnh báo trên mobile khi có người lạ

#### f. Giao diện trên mobile

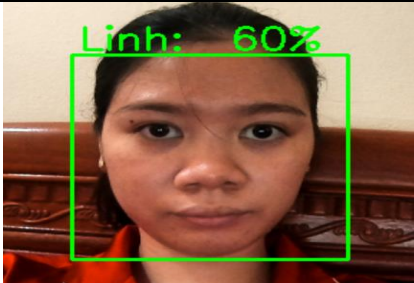


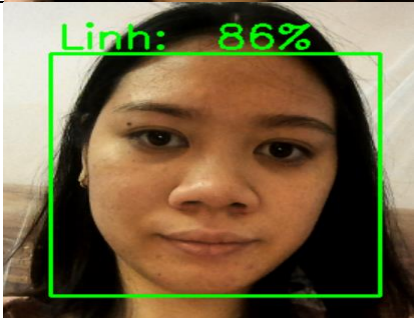


Hình 3- 18. Giao diện ứng dụng trên mobile

### 3.2. Thực nghiệm

Cấp nguồn cho thiết bị và tiến hành nhận dạng khuôn mặt qua Raspberry, điều khiển thiết bị qua Mobile. Kiểm tra tính chính xác và ổn định của hệ thống

Thực nghiệm độ chính xác với số lượng ảnh đầu vào, *kèm theo ảnh kết quả*

Số lượng ảnh đầu vào	Độ chính xác(%)	Hình ảnh minh chứng
30	Khoảng 60	 Linh: 60%
50	Khoảng 70	 Linh: 72%
100	Khoảng 80	 Linh: 80%
200	Khoảng 85	 Linh: 86%



### 3.3. Đánh giá

**Đánh giá số lượng ảnh đầu vào:** ảnh đầu vào là 200/người có độ chính xác khá cao

**Đánh giá về nhận diện khuôn mặt bằng HAAR CASCADES:**

- Phát hiện khuôn mặt nhanh
- Nhận diện khuôn mặt với độ chính xác là 96,24% [15]
- Hữu ích khi làm việc trong các thiết bị hạn chế tài nguyên[16]

**Đánh giá về nhận dạng khuôn mặt bằng OpenCV thông qua LBPH:**

- Nhận dạng khuôn mặt thông qua LBPH không bị ảnh hưởng bởi ánh sáng[17]
- Là một trong những thuật toán nhận dạng khuôn mặt dễ dàng nhất[18]
- Có thể đạt được kết quả tuyệt vời[18]

**Đánh giá hệ thống**

Công việc	Số lần thao tác	Số lần thành công	Thời gian đáp ứng	Đánh giá
Nhận dạng khuôn mặt	50	47	0,33 giây	94%
Trạng thái cửa khi có nhận dạng khuôn mặt	50	47	0,1 giây	94%
Trạng thái cửa khi điều khiển bằng nút	10	10	0,1 giây	100%
Điều khiển cửa qua Mobile bằng nút	10	10	0,1 giây	100%
Điều khiển cửa qua Mobile bằng giọng nói	10	9	0,1 giây	90%
Hiển thị thông báo trên Mobile	10	10	0,1 giây	100%
Hiển thị hình ảnh người lạ trên Mobile	10	10	0,1 giây	100%
<b>Đánh giá chung</b>				<b>96%</b>

Hệ thống hoạt động ổn định, tốc độ xử lý ảnh của phần mềm nhận dạng khuôn mặt vẫn còn thấp, chỉ được tầm 3 FPS, tức là 1 giây xử lý được 3 ảnh nên mỗi khung ảnh sau khi nhận dạng sẽ bị trễ với thời gian thực tầm 0,33 giây. Điều này do tốc độ xử lý CPU của Rpi 3 còn yếu. Tiếp theo là dữ liệu truyền đi và nhận về giữa MQTT Broker với các thiết bị còn lại còn bị hơi trễ với thời gian thực 0,1 giây do dữ liệu cần được mã hóa trước khi gửi và giải mã sau khi nhận về, đồng thời bị ảnh hưởng bởi tốc độ đường truyền mạng.

## KẾT LUẬN

### 1. Kết luận

Qua việc thực hiện đồ án “Xây dựng hệ thống chống trộm thông minh sử dụng Raspberry Pi và thuật toán học máy”, tôi đã đạt được một số kết quả nhất định:

- Hệ thống IoT hoàn chỉnh thu thập dữ liệu từ phần mềm nhận dạng khuôn mặt, truyền tải dữ liệu lên Broker, gia chủ có thể điều khiển đóng mở cửa từ xa, nhận cảnh báo và xem được hình ảnh người lạ qua thiết bị mobile
- Tìm hiểu về thuật toán học máy, cụ thể là thuật toán FR
- Tìm hiểu về mô hình, thành phần, cách thức hoạt động của giao thức MQTT
- Tìm hiểu về hoạt động, xác thực các thiết bị với Broker của SSL/TLS
- Tìm hiểu về thuật toán mã hóa AES, cụ thể là mã hóa AES 256 CBC để tăng tính an toàn cho hệ thống
- Xây dựng giao diện để sử dụng phần mềm trên Rpi
- Xây dựng phần mềm trên mobile bằng Flutter để sử dụng, dữ liệu giao tiếp với MQTT Broker
- Hệ thống hoạt động ổn định với tốc độ tầm 3fps

Ngoài việc học hỏi các kiến thức mới, thì sản phẩm đồ án này là một sản phẩm áp dụng thực tế dựa trên thực trạng hiện tại của xã hội

Hạn chế hiện tại của hệ thống là chưa có nhiều chức năng phức tạp, ví dụ như hiển thị trạng thái cửa trên Mobile. Về khả năng phát triển vì là hệ thống IoT hoàn chỉnh nên thiết bị này có thể tích hợp vào các hệ thống lớn hơn như “Nhà thông minh” hoặc tiếp tục phát triển các chức năng mới, logic hoạt động mới. Một hạn chế nữa là khi người lạ bật kín người thì hệ thống không thể nhận dạng được mà phải thông qua thuật toán nhận dạng cử chỉ con người để xem đó là gia chủ hay người lạ, mà muốn làm như vậy thì cần thêm thời gian phát triển

Để khắc phục các hạn chế trên thì hướng phát triển của tôi trong thời gian tới là:

- Phát triển hiển thị trạng thái cửa trên Mobile
- Phát triển hệ thống từ nhận dạng khuôn mặt đến nhận dạng cử chỉ con người

Đồ án “Xây dựng hệ thống chống trộm thông minh sử dụng Raspberry Pi và thuật toán học máy” là kết quả, là kiến thức, là kinh nghiệm của tôi trong suốt 5 năm theo học Học viện kỹ thuật Mật mã. Đề tài là thành quả mà tôi đã cố gắng thực hiện

Để hoàn thành được đồ án này, đầu tiên phải kể đến ThS Lê Thị Hồng Vân, người đã gắn bó với tôi ngay từ ý tưởng, đoạn code đầu tiên của đề tài. Mặc dù bị

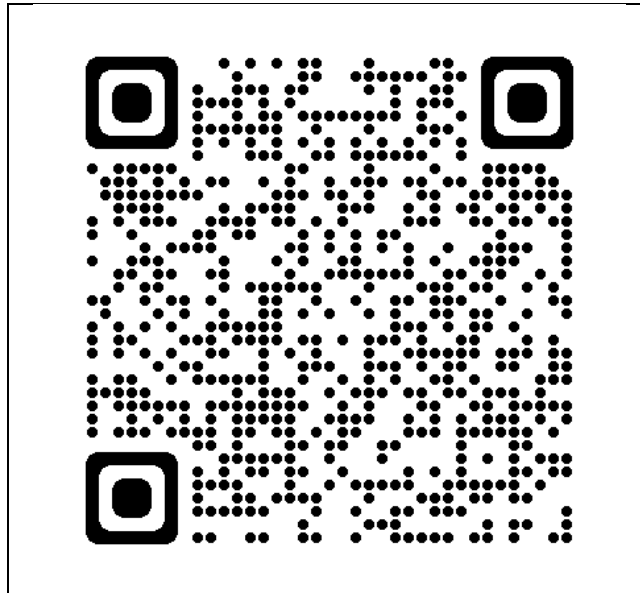
ảnh hưởng bởi dịch Covid nhưng thầy đã tận tình giúp đỡ tôi hoàn thành được đề tài này. Ngoài ra, tôi xin gửi lời cảm ơn đến lãnh đạo học viện, các quý thầy cô đã tạo điều kiện tốt nhất cho chúng tôi trong suốt quá trình học tập.

Cuối cùng, không thể thiếu đó chính là gia đình, những anh chị đồng nghiệp, bạn bè đã luôn ở bên giúp đỡ cho tôi có được môi trường thuận lợi để thực hiện đề tài cũng như công việc sau này.

Tôi xin chân thành cảm ơn!

## **2. Mã nguồn sản phẩm**

- Nhập đường link: <https://github.com/linhct020328/DoAnTotNghiep>
- Hoặc quét mã QR sau:



## TÀI LIỆU THAM KHẢO

- [1] Minh Hiếu, Thanh Tùng (ANTV), Gia tăng tội phạm trộm cắp, tệ nạn tại các khu chung cư, 21/12/2021, “<https://vtv.vn/phap-luat/gia-tang-toi-pham-trom-cap-te-nan-tai-cac-khu-chung-cu-20211221020842968.htm>”
- [2] Quốc Trung, Thiên Quý, TP. HCM: Trộm cướp gia tăng sau giãn cách, 12/11/2021,” <https://vtv.vn/xa-hoi/tp-hcm-trom-cuop-gia-tang-sau-gian-cach-20211112151947217.htm>”
- [3] Hệ thống nhúng là gì? Khái niệm, đặc điểm và ứng dụng, “<https://bkaii.com.vn/tin-tuc/508-he-thong-nhung-la-gi-khai-niem-dac-diem-va-ung%20dung>”
- [4] thuychi21, Tổng quan về học máy, 11/12/2015 ,” <http://luanvan.net.vn/luan-van/tong-quan-ve-hoc-may-71851/>”
- [5] RASPBERRYPI VIỆT NAM, Raspberry Pi là gì? Giới thiệu về Raspberry Pi. 08/01/2014,” <https://raspberrypi.vn/raspberry-pi-la-gi-gioi-thieu-ve-raspberry-pi-261.pi>”
- [6] SuperDataScience Team, Face recognition using OpenCV and Python: A beginner's guide, Thursday Aug 03, 2017, “<https://www.superdatascience.com/blogs/opencv-face-recognition>”
- [7] steve, Creating and Using Client Certificates with MQTT and Mosquitto, May 23, 2022, “<http://www.steves-internet-guide.com/creating-and-using-client-certificates-with-mqtt-and-mosquitto/>”
- [8] TheWalrus, Using Esp8266 As An IoT Endpoint With Encrypted MQTT Transport, 2019-03-28 18:10:19, “<https://www.gushiciku.cn/pl/2ypT>”
- [9] Face Recognition và face Detection trong OpenCV, “<https://websitehcm.com/face-recognition-va-face-detection-trong-opencv/#:~:text=Face%20Recognition%3A%20Thu%E1%BA%ADt%20to%C3%A1n%20nh%E1%BA%ADn,v%C3%A0%20nh%E1%BA%ADn%20d%E1%BA%A1ng%20khu%C3%B4n%20m%E1%BA%B7t.>”
- [10] VU NGOC TUAN, Giới thiệu về Flutter, thg 3 20, 2018 5:17 CH, “<https://viblo.asia/p/gioi-thieu-ve-flutter-bWrZnNxrZxw>”
- [11] Flutter là gì? Những định nghĩa và tính năng liên quan?, 01/08/2021 - 10:18, “<https://teky.edu.vn/blog/flutter-la-gi/>”

- [12] K.V, Đánh giá Raspberry Pi 4 Model B, 05/07/2019, ["https://fptshop.com.vn/tin-tuc/danh-gia/danh-gia-raspberry-pi-4-model-b-90893#:~:text=CPU%20c%E1%BB%A7a%20Raspberry%20Pi%204,m%E1%BB%99t%20c%E1%BB%97%20PC%20entry%2Dlevel."](https://fptshop.com.vn/tin-tuc/danh-gia/danh-gia-raspberry-pi-4-model-b-90893#:~:text=CPU%20c%E1%BB%A7a%20Raspberry%20Pi%204,m%E1%BB%99t%20c%E1%BB%97%20PC%20entry%2Dlevel.)
- [13] Mỹ Hạnh, MQTT và HTTP: Giao thức nào tốt hơn trong thời đại IoT?, Thứ Ba, 01/10/2019 16:52, ["https://quantrimang.com/so-sanh-mqtt-va-http-166657"](https://quantrimang.com/so-sanh-mqtt-va-http-166657)
- [14] atung, Những điều cần biết về nhận dạng khuôn mặt, Tháng Tư 15, 2020, ["https://beetinnovators.com/nhung-dieu-can-biet-ve-giai-phap-nhan-dien-khuon-mat/"](https://beetinnovators.com/nhung-dieu-can-biet-ve-giai-phap-nhan-dien-khuon-mat/)
- [15]Anirudha B Shetty, Bhoomika, Deeksha, Jeevan Rebeiro, Facial Recognition using Haar Cascade and LBP Classifiers, August 2021, ["https://www.sciencedirect.com/science/article/pii/S2666285X21000728#:~:text=The%20current%20work%20is%20based,best%20algorithm%20for%20their%20work."](https://www.sciencedirect.com/science/article/pii/S2666285X21000728#:~:text=The%20current%20work%20is%20based,best%20algorithm%20for%20their%20work.)
- [16] Adrian Rosebrock, Adrian Rosebrock, April 12, 2021, ["https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/"](https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/)
- [17] SuperDataScience Team, Face recognition using OpenCV and Python: A beginner's guide, Thursday Aug 03, 2017, ["https://www.superdatascience.com/blogs/opencv-face-recognition/"](https://www.superdatascience.com/blogs/opencv-face-recognition/)
- [18] Towards Data Science, Kelvin Salton do Prado, Face Recognition: Understanding LBPH Algorithm, Nov 10, 2017, ["https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b?gi=6189d6a68823"](https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b?gi=6189d6a68823)
- [19]Nguyễn Quân, [AES] Bài 6- Các chế độ mã hóa và giải mã, 2020.09.15, ["https://nguyenquanicd.blogspot.com/2019/10/aes-bai-6-cac-che-o-ma-hoa-va-giai-ma.html"](https://nguyenquanicd.blogspot.com/2019/10/aes-bai-6-cac-che-o-ma-hoa-va-giai-ma.html)
- [20] ClickSSL, 256 Bit Encryption – Is AES-256 Bit Encryption Safe in Modern Times? March 23, 2022, ["https://www.clickssl.net/blog/256-bit-encryption"](https://www.clickssl.net/blog/256-bit-encryption)