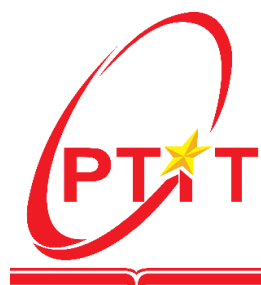


**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
**KHOA CÔNG NGHỆ THÔNG TIN 1**

---



**IOT VÀ ỨNG DỤNG**

**ĐỀ TÀI: Hệ thống khóa cửa thông minh đa phương  
thức sử dụng thẻ từ và xác thực khuôn mặt**

**Lớp: D22CMPM02**  
**Nhóm: 03**  
**Giảng viên: Kim Ngọc Bách**

**Danh sách thành viên**

<b>Mã sinh viên</b>	<b>Họ và tên</b>
<b>B22DCCN064</b>	<b>Nguyễn Đức Bảo</b>
<b>B22DCCN604</b>	<b>Đỗ Tuấn Nghĩa</b>
<b>B22DCCN244</b>	<b>Phạm Văn Đức</b>
<b>B22DCCN088</b>	<b>Lý Chí Công</b>

**HÀ NỘI – 2025**

# MỤC LỤC

1	Giới thiệu chung.....	6
1.1	Lý do chọn đề tài .....	6
1.2	Tổng quan về dự án .....	6
1.3	Mục đích của dự án.....	7
1.4	Đối tượng và phạm vi nghiên cứu .....	7
1.5	Phương pháp nghiên cứu .....	7
1.6	Công nghệ và thiết bị cần thiết cho dự án .....	8
2	Nền tảng lý thuyết .....	9
2.1	Vi điều khiển ESP32.....	9
2.1.1	Giới thiệu chung.....	9
2.1.2	Kiến trúc phần cứng và các chân giao tiếp.....	10
2.1.3	Chức năng của ESP32 trong hệ thống.....	11
2.2	Module ESP32-CAM và camera OV2640 .....	13
2.2.1	Tổng quan về ESP32-CAM.....	13
2.2.2	Cảm biến ảnh OV2640 .....	14
2.2.3	Quy trình chụp và truyền ảnh.....	14
2.3	Đầu đọc thẻ RFID RC522.....	16
2.3.1	Nguyên lý RFID .....	16
2.3.2	Cấu tạo và chức năng module RC522 .....	17
2.3.3	Giao tiếp giữa RC522 và ESP32 .....	17
2.4	Cơ cấu chấp hành: khóa điện từ, servo và relay .....	18
2.4.1	Khóa điện từ (solenoid lock).....	18
2.4.2	Servo motor trong cơ cấu khóa .....	19
2.4.3	Module relay.....	19
2.5	Hệ thống nhúng và tổ chức phần mềm trên ESP32 .....	20
2.5.1	Khái niệm hệ thống nhúng .....	20
2.5.2	FreeRTOS trên ESP32.....	21
2.5.3	Tổ chức phần mềm trên ESP32 trong đề tài.....	21
2.6	Mạng không dây và các giao thức truyền thông sử dụng .....	22

2.6.1	Wi-Fi 802.11 .....	22
2.6.2	Giao thức MQTT .....	23
2.6.3	Giao thức WebSocket .....	23
2.6.4	HTTP và REST (ở mức khái niệm).....	23
2.7	Trí tuệ nhân tạo và thị giác máy tính trong nhận diện khuôn mặt.....	24
2.7.1	Giới thiệu chung .....	24
2.7.2	Mạng nơ-ron tích chập (Convolutional Neural Network – CNN) .....	25
2.7.3	Phát hiện khuôn mặt bằng MTCNN (Multi-task Cascaded Convolutional Networks) .....	25
2.7.4	Embedding khuôn mặt và so sánh khoảng cách.....	26
2.8	Bảo mật và an toàn thông tin trong hệ thống khóa cửa thông minh.....	27
2.8.1	Các nguy cơ bảo mật tiềm ẩn .....	27
2.8.2	Các nguyên tắc bảo mật cơ bản.....	27
2.8.3	Bảo vệ dữ liệu sinh trắc học .....	28
3	PHÂN TÍCH YÊU CẦU CHỨC NĂNG .....	29
3.1	Chức năng xác thực người dùng.....	29
3.2	Chức năng mở khóa cửa .....	29
3.3	Chức năng xem thông tin truy cập theo thời gian thực .....	30
3.4	Chức năng cảnh báo an ninh.....	30
3.5	Chức năng quản lý người dùng.....	30
3.6	Chức năng điều khiển từ xa .....	31
3.7	Chức năng xem và xuất dữ liệu .....	31
4	THIẾT KẾ HỆ THỐNG.....	32
4.1	Mô hình tổng quan.....	32
4.2	Đặc tả mô hình miền.....	32
4.2.1	Lớp Thiết bị & Nhận thức (Device/Perception Layer) .....	33
4.2.2	Lớp Mạng (Network Layer) .....	33
4.2.3	Lớp Ứng dụng (Application Layer) .....	34
4.3	Đặc tả các thành phần chức năng và hoạt động.....	35
4.3.1	Thành phần chức năng .....	35
4.3.2	Thành phần hoạt động .....	35

4.4	Đặc tả luồng hoạt động .....	35
4.4.1	Thiết bị cảm biến và vi xử lý.....	35
4.4.2	Website quản lý .....	43
4.5	Tích hợp thiết bị.....	54
4.6	Phát triển ứng dụng.....	55
4.6.1	Phát triển Backend.....	55
4.6.2	Phát triển Frontend .....	58
4.6.3	Phát triển Firmware ESP32 .....	58
5	Kết Luận.....	60
5.1	Kết quả đạt được.....	60
5.1.1	Về mặt lý thuyết .....	60
5.1.2	Về mặt thực tiễn .....	60
5.2	Hạn chế .....	60
5.3	Hướng phát triển tương lai.....	61
5.3.1	Về hệ thống .....	61
5.3.2	Về bảo mật.....	61

## DANH SÁCH HÌNH VẼ

Hình 1 Bo mạch ESP32 DevKit V1 .....	9
Hình 2 ESP32-CAM.....	13
Hình 3 Modul đọc thẻ từ RC522 .....	16
Hình 4 Biểu đồ usecase mô hình tổng quan hệ thống .....	32
Hình 5 Hình vẽ mô hình miền .....	32
Hình 6 Biểu đồ luồng cảnh báo an ninh .....	40
Hình 7 Biểu đồ luồng mở cửa .....	40
Hình 8 Biểu đồ luồng xác thực khuôn mặt.....	42
Hình 9 Biểu đồ luồng xác thực RFID.....	43
Hình 10 Biểu đồ luồng mở khóa trên web.....	50
Hình 11 Biểu đồ luồng quản lý người dùng trên web .....	51
Hình 12 Biểu đồ luồng chức năng chuyển chế độ.....	52
Hình 13 Biểu đồ luồng chức năng thêm thiết bị.....	54
Hình 14 Sơ đồ mạch lắp đặt hệ thống .....	54

## DANH SÁCH BẢNG BIỂU

Bảng 1 Bảng thành phần lớp Thiết bị và Nhận thức .....	33
Bảng 2 Bảng thành phần lớp Mạng .....	34
Bảng 3 Bảng thành phần lớp Ứng dụng .....	34
Bảng 4 Bảng mô tả chức năng.....	35
Bảng 5 Bảng mô tả thành phần hoạt động.....	35
Bảng 6 Bảng các thư viện sử dụng trên ESP32/ ESP32-CAM .....	36
Bảng 7 Bảng mô tả chức năng Hàm khởi tạo.....	37
Bảng 8 Bảng mô tả chức năng Hàm vòng lặp chính .....	37
Bảng 9 Bảng mô tả chức năng Hàm xử lý phần cứng.....	38
Bảng 10 Bảng mô tả chức năng Hàm giao tiếp mạng (MQTT/Server) .....	38
Bảng 11 Bảng mô tả chức năng Hàm hệ thống - điều phối thiết bị.....	39
Bảng 12 Bảng các thư viện dùng ở Server .....	44
Bảng 13 Bảng các nhóm API sử dụng.....	47
Bảng 14 Bảng các thư viện sử dụng ở AI.....	48
Bảng 15 Bảng mô tả các hàm sử dụng nhận diện khuôn mặt .....	49
Bảng 16 Bảng các thư viện sử dụng ở Client.....	50

# 1 Giới thiệu chung

## 1.1 Lý do chọn đề tài

Trong bối cảnh nhu cầu an ninh ngày càng tăng cao và việc bảo vệ tài sản, nhà ở, phòng làm việc, văn phòng trở thành ưu tiên hàng đầu, các phương thức mở cửa truyền thống như chìa khóa cơ học hay khóa số đang bộc lộ nhiều hạn chế. Chìa khóa có thể bị sao chép hoặc bị thất lạc, mật khẩu dễ bị lộ nếu người dùng nhập nhiều lần trước người khác. Những rủi ro này khiến mức độ an toàn của các công trình không còn được đảm bảo như mong muốn. Vì vậy, việc nghiên cứu và phát triển các giải pháp mở cửa thông minh, linh hoạt và đáng tin cậy trở thành yêu cầu thiết yếu trong bối cảnh hiện nay.

Cùng lúc đó, sự phát triển nhanh chóng của công nghệ Internet of Things và trí tuệ nhân tạo, đặc biệt là công nghệ nhận diện khuôn mặt, đã mở ra nhiều cơ hội ứng dụng trong đời sống và các hệ thống an ninh. Các mô hình xác thực đa dạng phương thức, cho phép người dùng tùy chọn giữa nhiều hình thức khác nhau, đang được xem là xu hướng mới nhờ tính tiện lợi, dễ sử dụng và khả năng nâng cao mức độ bảo mật. Việc kết hợp giữa mở cửa bằng thẻ RFID và nhận diện khuôn mặt giúp người dùng linh hoạt hơn trong quá trình sử dụng, đồng thời vẫn đảm bảo độ chính xác và an toàn cao.

Xuất phát từ những yêu cầu thực tiễn đó, đề tài **Hệ thống mở cửa thông minh đa phương thức** được lựa chọn nhằm xây dựng một mô hình có khả năng hoạt động tự động, giám sát và kiểm soát người ra vào một cách nhanh chóng và chính xác. Hệ thống vừa mang lại sự tiện lợi cho người dùng, vừa nâng cao hiệu quả an ninh tại khu vực áp dụng.

## 1.2 Tổng quan về dự án

Dự án xây dựng hệ thống mở cửa thông minh đa phương thức là một ứng dụng của công nghệ Internet of Things kết hợp với xử lý ảnh và AI, cho phép người dùng thực hiện việc kiểm soát ra vào một cách tự động, linh hoạt và an toàn hơn so với các phương pháp truyền thống. Hệ thống cho phép sử dụng hai phương thức xác thực: mở cửa bằng thẻ RFID hoặc mở cửa bằng nhận diện khuôn mặt hoặc kết hợp cả hai, tùy theo nhu cầu và điều kiện của người dùng tại thời điểm sử dụng.

Hệ thống bao gồm cơ cấu phần cứng được đặt tại khu vực cửa ra vào, trong đó có vi điều khiển, camera nhận diện khuôn mặt, module RFID và bộ điều khiển khóa. Dữ liệu trạng thái truy cập có thể được ghi nhận và truyền về hệ thống quản lý thông qua mạng Internet, cho phép người dùng theo dõi lịch sử ra vào, trạng thái cửa và nhật ký hoạt động từ xa qua giao diện ứng dụng hoặc dashboard trực quan.

### 1.3 Mục đích của dự án

Hệ thống mở cửa thông minh có khả năng hoạt động tự động, an toàn và linh hoạt hơn so với các phương thức mở khóa truyền thống. Thông qua việc tích hợp hai hình thức xác thực là nhận diện khuôn mặt và đọc thẻ RFID, hệ thống phù hợp hơn với nhiều nhu cầu khác nhau, từ gia đình đến văn phòng hay các khu vực yêu cầu kiểm soát ra vào nghiêm ngặt.

Hệ thống có thể tự động quét khuôn mặt hoặc nhận diện ID của thẻ RFID, giúp quá trình mở cửa nhanh chóng và chính xác, hạn chế tối đa các sai sót. Ngoài khả năng xác thực, hệ thống cũng ghi lại đầy đủ lịch sử mở khóa, cho phép người dùng theo dõi thời điểm mở cửa, danh tính người dùng và trạng thái hiện tại của cửa (đang đóng hay mở).

Hệ thống được trang bị cơ chế cảnh báo tức thì khi phát hiện số lần xác thực thất bại vượt ngưỡng cho phép. Cảnh báo sẽ được gửi trực tiếp lên website, giúp người dùng kịp thời xử lý các tình huống truy cập trái phép.

Hệ thống hướng đến khả năng hoạt động ổn định liên tục 24/7, đảm bảo luôn sẵn sàng phục vụ trong mọi tình huống.

Ngoài ra, hệ thống còn cho phép dễ dàng thêm hoặc xóa khuôn mặt và thẻ RFID từ giao diện quản trị. Điều này giúp việc quản lý trở nên đơn giản, thuận tiện và phù hợp trong các môi trường có nhiều người sử dụng.

Hệ thống giúp tiết kiệm thời gian mở khóa, tăng mức độ bảo mật, giảm thiểu rủi ro xâm nhập trái phép và mang lại trải nghiệm tiện lợi cho người dùng.

### 1.4 Đối tượng và phạm vi nghiên cứu

**Đối tượng nghiên cứu:** Các thiết bị và công nghệ cho hệ thống mở cửa thông minh, bao gồm camera nhận diện khuôn mặt, module RFID (RC522), vi điều khiển ESP32, relay điều khiển khóa cửa, các cảm biến từ trường hỗ trợ trạng thái cửa, các thuật toán xử lý ảnh và nhận diện khuôn mặt.

**Phạm vi nghiên cứu:** Đề tài tập trung vào việc thiết kế, xây dựng và triển khai một hệ thống mở cửa thông minh hoạt động trong nhà ở. Các chức năng chính của hệ thống: thu thập dữ liệu khuôn mặt hoặc thẻ RFID, xác thực người dùng, điều khiển khóa cửa, cảnh báo khi truy cập thất bại và ghi nhận đầy đủ lịch sử mở khóa. Dữ liệu được truyền tải qua mạng Wifi và hiển thị trên giao diện quản lý trực quan, cho phép thêm/xóa người dùng và theo dõi trạng thái cửa theo thời gian thực.

### 1.5 Phương pháp nghiên cứu

**Phương pháp nghiên cứu lý thuyết:** Tìm hiểu và nghiên cứu các kiến thức liên quan đến công nghệ nhận diện khuôn mặt, xử lý ảnh, thuật toán so khớp dữ liệu khuôn mặt, cơ chế hoạt động của module RFID (RC522), vi điều khiển ESP32,

relay điều khiển khóa cửa và các giao thức truyền thông không dây sử dụng trong mô hình IoT.

**Phương pháp thực nghiệm:** Tiến hành thiết kế, lắp đặt và triển khai thực tế hệ thống dựa trên các thiết bị phần cứng để đánh giá độ chính xác của quá trình nhận diện khuôn mặt, tốc độ đọc thẻ, khả năng xử lý tín hiệu, sự ổn định khi truyền dữ liệu qua Wifi và độ tin cậy của cơ chế mở khóa. Thực nghiệm cũng bao gồm kiểm tra chức năng cảnh báo khi xác thực thất bại và đánh giá tính khả dụng của giao diện quản lý.

**Phương pháp phân tích:** Sử dụng các công cụ phân tích dữ liệu để đánh giá kết quả thu được từ quá trình thử nghiệm, bao gồm thời gian xác thực, tỷ lệ nhận diện đúng, tỷ lệ lỗi khi quét thẻ, độ ổn định của kết nối và mức độ phản hồi của hệ thống cửa. Kết quả thực nghiệm được so sánh với các mô hình lý thuyết và tiêu chuẩn kỹ thuật đã nghiên cứu nhằm xác định mức độ hiệu quả của hệ thống. Đồng thời, tiến hành phân tích nguyên nhân gây lỗi, đưa ra các điều chỉnh và tối ưu cần thiết để hoàn thiện hệ thống ở mức độ tốt nhất.

## 1.6 Công nghệ và thiết bị cần thiết cho dự án

Phần cứng:

- Module ESP32-WROOM-32 NodeMCU-32S 30 pin
- Module relay 1 kênh 5VDC tùy chọn kích hoạt mức cao hoặc thấp H/L
- Module cảm biến từ trường
- Còi chirp 3-24V 3015A Buzzer âm thanh bip dài
- Module RFID RC522 NFC 13.56Mhz
- Module ESP32-CAM Wifi Bluetooth OV2640 Ai-Thinker
- Khóa chốt điện từ LY-03 12V 0.4A loại tốt, tiết kiệm năng lượng
- Chốt giữ khóa điện từ LY-01/ LY-03 bằng inox
- Dây cắm (Jumper wires)
- Pin 12V và đế pin

Phần mềm:

- Arduino IDE: Lập trình và nạp dữ liệu vào board
- Frizing: Thiết kế mạch
- Visual Code Studio: Thiết kế trang web theo dõi thông tin, sử dụng React + Node.js

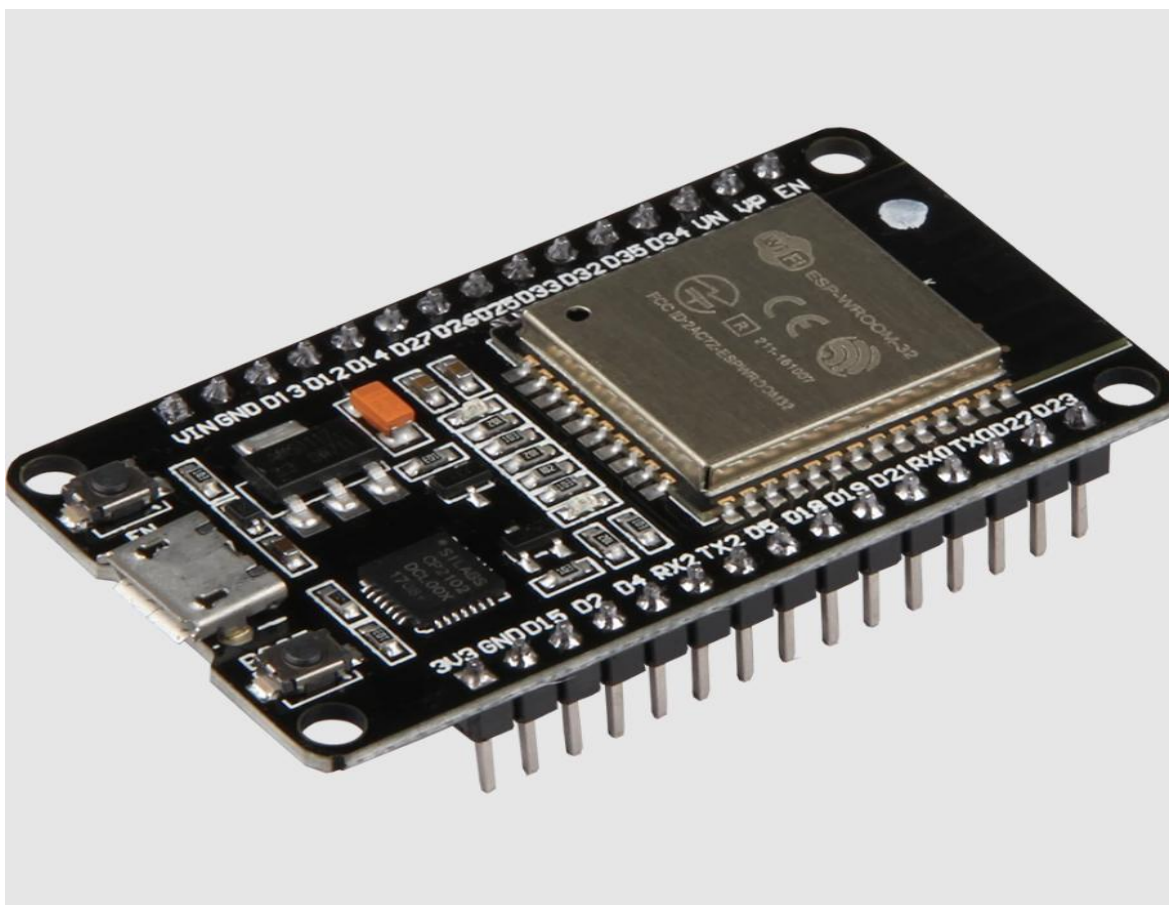
## 2 Nền tảng lý thuyết

### 2.1 Vi điều khiển ESP32

#### 2.1.1 Giới thiệu chung

ESP32 là một vi điều khiển 32 bit do hãng Espressif phát triển, được thiết kế riêng cho những ứng dụng cần vừa xử lý logic, vừa có kết nối không dây. Thay vì phải dùng một vi điều khiển và thêm một module Wi-Fi rời, ESP32 tích hợp tất cả vào trong một con chip duy nhất: bên trong có CPU, bộ nhớ, khối Wi-Fi 2,4 GHz, Bluetooth và nhiều ngoại vi như SPI, I2C, UART, ADC, PWM...

Về lõi xử lý, đa số phiên bản ESP32 sử dụng hai lõi Xtensa LX6, chạy ở tần số lên tới 240 MHz. Một lõi thường được dùng để xử lý các tác vụ mạng (Wi-Fi, TCP/IP), lõi còn lại dành cho chương trình ứng dụng. Nhờ vậy, ngay cả khi thiết bị đang truyền dữ liệu với máy chủ, các tác vụ thời gian thực như đọc thẻ RFID hay điều khiển khóa vẫn hoạt động mượt.



Hình 1 Bo mạch ESP32 DevKit V1

Trong các ứng dụng như khóa cửa thông minh, ESP32 mang lại nhiều lợi ích:

- Không cần thêm module Wi-Fi rời, giảm số lượng linh kiện và kích thước mạch.

- Tài nguyên CPU và RAM đủ để xử lý nhiều việc cùng lúc: đọc thẻ, điều khiển khóa, gửi nhận dữ liệu mạng.
- Hệ sinh thái thư viện phong phú, cộng đồng lớn, dễ tìm ví dụ và tài liệu để phát triển.

Nhờ những đặc tính đó, ESP32 được chọn làm bộ điều khiển trung tâm cho hệ thống khóa cửa trong đề tài.

### 2.1.2 Kiến trúc phần cứng và các chân giao tiếp

Một board phát triển ESP32 thông dụng (ví dụ ESP32 DevKit) ngoài con chip ESP32 còn có thêm các khối mạch phụ trợ để người dùng có thể sử dụng dễ dàng.

- **Khối cấp nguồn:**  
Board thường có chân 5V (hoặc VIN) để cấp nguồn từ cổng USB hoặc adapter, bên trong có mạch ổn áp chuyển 5V xuống 3,3V – là mức điện áp hoạt động của chip ESP32 và nhiều ngoại vi khác. Nhờ có mạch này, người dùng chỉ cần cấp 5V là đủ, không phải tự thiết kế nguồn 3,3V riêng.
- **Bộ chuyển đổi USB – UART:**  
Trên board có một chip chuyển đổi giao tiếp USB sang UART (thường là CP2102, CH340 hoặc tương đương). Khi nối cáp USB với máy tính, bộ chuyển đổi này cho phép:
  - Nạp chương trình (firmware) vào ESP32 qua UART.
  - In thông tin log, debug từ chương trình ra màn hình Serial trên máy tính.
- **Chip ESP32:**  
Đây là khối “trung tâm thần kinh” của board. Bên trong bao gồm:
  - CPU lõi kép.
  - Bộ nhớ RAM.
  - Bộ nhớ ROM chứa Bootloader và một số thư viện sẵn.
  - Khối Wi-Fi, Bluetooth.
  - Các khối ngoại vi như ADC, DAC, PWM, SPI, I2C, UART...
- **Các hàng chân GPIO:**  
Xung quanh board là các chân GPIO (General Purpose Input/Output). Mỗi chân có thể được cấu hình thành:
  - Ngõ vào số để đọc trạng thái công tắc, nút bấm.
  - Ngõ ra số để điều khiển LED, relay, servo.

- Chân giao tiếp chuẩn SPI, I2C, UART khi cần nói chuyện với module RC522, cảm biến khác...
- Một số chân đặc biệt hỗ trợ ADC để đọc giá trị analog.

Trong đề tài, các nhóm chân được sử dụng một cách có chủ đích:

- Nhóm chân SPI được nối với module RC522, giúp ESP32 gửi lệnh và nhận UID thẻ.
- Một hoặc vài chân GPIO output được nối tới module relay (điều khiển khóa điện từ) hoặc tạo xung PWM cho servo.
- Một số chân khác có thể nối LED chỉ thị (ví dụ LED báo đang kết nối Wi-Fi, LED báo cửa đang mở), hoặc nút reset/mở khẩn cấp để người quản trị thao tác trực tiếp.

Việc thiết kế và phân bổ chân hợp lý giúp mạch gọn, dễ đi dây và thuận tiện nếu sau này muốn mở rộng thêm cảm biến, nút bấm,...

### 2.1.3 Chức năng của ESP32 trong hệ thống

Trong hệ thống khóa cửa thông minh, ESP32 không chỉ là vi điều khiển đơn thuần mà đóng vai trò như một **nút điều khiển thông minh** đặt tại cửa. Các nhiệm vụ chính có thể mô tả chi tiết như sau:

#### 1. **Giao tiếp với đầu đọc thẻ RC522**

Ngay sau khi khởi động, ESP32 khởi tạo giao tiếp SPI và đặt RC522 vào chế độ chờ thẻ. Sau đó, trong vòng lặp, ESP32 liên tục yêu cầu RC522 kiểm tra xem có thẻ nằm trong vùng anten hay không.

- Nếu chưa có thẻ, ESP32 chỉ chờ trong thời gian ngắn rồi kiểm tra lại, đảm bảo không bỏ lỡ thẻ quét nhanh.
- Khi phát hiện thẻ, ESP32 yêu cầu RC522 thực hiện các bước chống xung đột (anti-collision) và đọc **UID** của thẻ. UID này được lưu lại và chuyển sang bước xử lý tiếp theo.

#### 2. **Giao tiếp mạng với hệ thống xử lý**

Trước đó, ESP32 đã kết nối vào Wi-Fi và thiết lập kết nối với máy chủ (qua MQTT hoặc WebSocket).

Khi có UID mới, ESP32 **đóng gói thông tin** gồm UID, thời gian, ID thiết bị... rồi gửi lên máy chủ. Máy chủ sẽ kiểm tra UID này trong cơ sở dữ liệu để quyết định có cho phép mở cửa hay không. Kết quả (cho phép/không cho phép, yêu cầu chụp thêm ảnh khuôn mặt...) được gửi ngược về ESP32 theo cùng kênh liên lạc.

### 3. Điều khiển cơ cấu khóa

Sau khi nhận lệnh từ máy chủ, ESP32 quyết định:

- Nếu không cho phép mở, chỉ cần ghi log và có thể chớp LED hoặc phát còi báo lỗi (nếu có).
- Nếu được phép mở, ESP32 kích chân điều khiển relay hoặc tạo xung PWM điều khiển servo để mở cửa trong một khoảng thời gian nhất định (ví dụ 3–5 giây). Sau khoảng thời gian đó, ESP32 tự động đưa tín hiệu về trạng thái ban đầu để khóa cửa lại, tránh trường hợp người dùng quên đóng.

### 4. Giám sát trạng thái hoạt động

ESP32 cũng có thể theo dõi tình trạng mạng, tình trạng nguồn, số lần quét thẻ sai liên tiếp, ... Nếu phát hiện các bất thường (mất Wi-Fi quá lâu, thẻ bị quét sai nhiều lần), thiết bị có thể gửi báo cáo về máy chủ để hệ thống sinh cảnh báo hoặc chuyển sang chế độ an ninh cao hơn.

### 5. Tổ chức phần mềm dạng nhiều tác vụ (task)

Nhờ FreeRTOS tích hợp sẵn, chương trình trên ESP32 có thể được chia thành nhiều task độc lập:

- Task đọc RFID.
- Task giao tiếp mạng.
- Task điều khiển khóa.
- Task giám sát.  
Cách tổ chức này giúp thiết bị luôn phản hồi nhanh, không bị “đơ” khi một phần nào đó (ví dụ gửi ảnh, mất mạng) mất nhiều thời gian.

Tóm lại, ESP32 là trung tâm điều phối tất cả hoạt động ở phía thiết bị tại cửa: từ đọc thẻ, gửi dữ liệu, nhận quyết định, điều khiển khóa đến giám sát và báo lỗi.

## 2.2 Module ESP32-CAM và camera OV2640

### 2.2.1 Tổng quan về ESP32-CAM



Hình 2 ESP32-CAM

ESP32-CAM là một module phát triển tích hợp chip ESP32 với một camera nhỏ và khe cắm thẻ nhớ microSD. Thay vì chỉ xử lý dữ liệu số như nhiều board vi điều khiển khác, ESP32-CAM được thiết kế để làm việc với hình ảnh, cụ thể là chụp ảnh và truyền ảnh qua mạng.

Về mặt cấu tạo, trên một module ESP32-CAM có:

- Chip ESP32, đảm nhiệm toàn bộ phần xử lý và kết nối Wi-Fi.
- Cảm biến ảnh OV2640 được gắn qua socket, có thể thay thế nếu hỏng.
- Khe cắm microSD cho phép lưu ảnh hoặc log cục bộ.
- Một số linh kiện phụ trợ như mạch ổn áp, mạch reset, thạch anh,...

Khác với ESP32 DevKit, ESP32-CAM không có sẵn cổng USB nạp code. Khi nạp chương trình, người dùng thường phải sử dụng một mạch chuyển USB–UART bên ngoài, nối tới các chân TX/RX của module và đưa chân IO0 xuống mức thấp để vào chế độ bootloader.

Trong hệ thống khóa cửa, ESP32-CAM được sử dụng như một “mắt quan sát”: chụp khuôn mặt người đang đứng trước cửa và gửi cho hệ thống xử lý để thực hiện nhận diện.

### 2.2.2 Cảm biến ảnh OV2640

Trái tim của phần camera là cảm biến OV2640 – một cảm biến CMOS độ phân giải khoảng 2 megapixel. Cảm biến này có các đặc điểm:

- Hỗ trợ nhiều độ phân giải khác nhau: từ rất thấp (QQVGA 160×120) tới trung bình (VGA 640×480) hoặc cao hơn.
- Hỗ trợ tự động chỉnh một số tham số như độ sáng, cân bằng trắng, tương phản... ở mức cơ bản, giúp ảnh chụp trong nhiều điều kiện ánh sáng khác nhau vẫn có thể sử dụng được.
- Giao tiếp với ESP32 thông qua các đường dữ liệu song song và một số chân điều khiển đồng bộ.

Trong đề tài, việc lựa chọn độ phân giải ảnh là một bài toán cân bằng:

- Nếu chọn độ phân giải cao, khuôn mặt trong ảnh sẽ rõ nét, thuận lợi cho mô hình nhận diện, nhưng kích thước file ảnh lớn, thời gian truyền qua Wi-Fi lâu.
- Nếu chọn độ phân giải thấp, ảnh gửi lên sẽ nhanh hơn nhưng chi tiết trên mặt bị mất, có thể làm giảm độ chính xác của mô hình.

Nhóm có thể chọn mức độ phân giải trung bình (ví dụ VGA) kết hợp với mức nén JPEG phù hợp để đạt sự cân bằng giữa chất lượng ảnh và tốc độ truyền.

### 2.2.3 Quy trình chụp và truyền ảnh

Quy trình ESP32-CAM làm việc trong hệ thống có thể mô tả chi tiết theo các bước:

#### 1. Khởi tạo camera

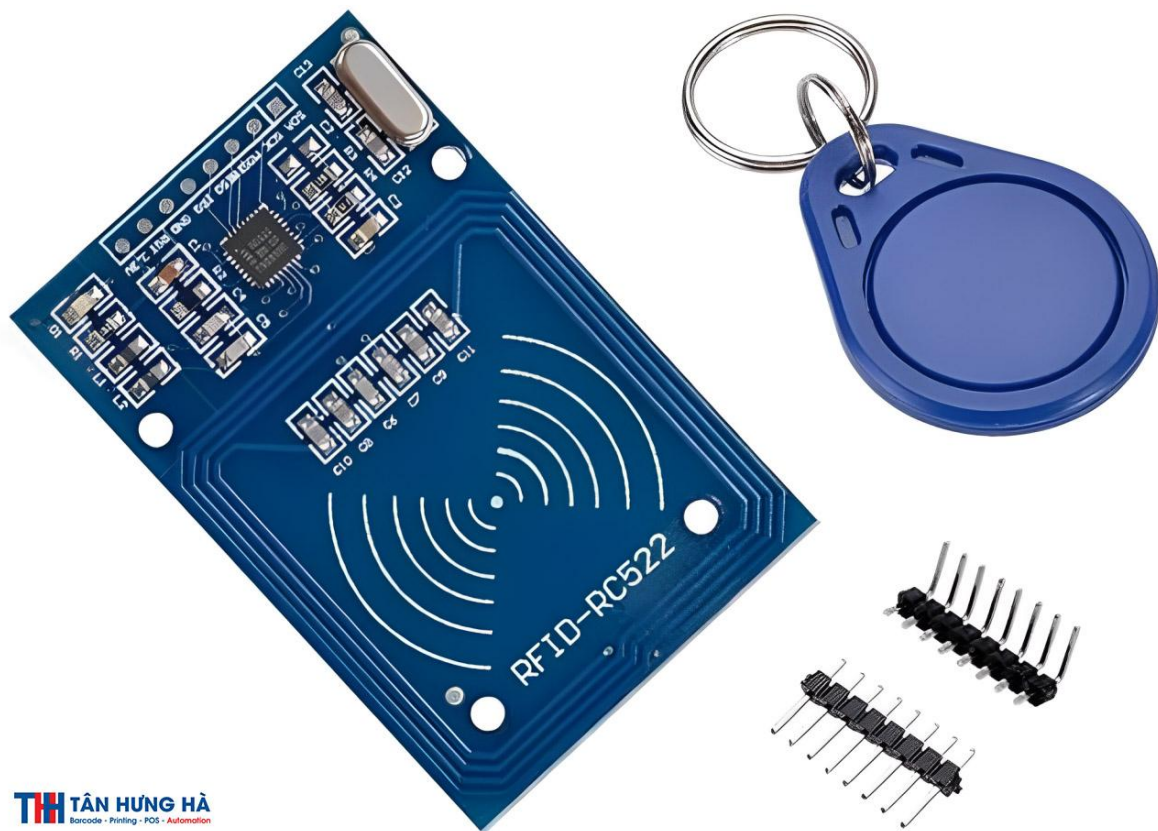
Sau khi module khởi động, chương trình tiến hành khởi tạo cảm biến OV2640. Quá trình này bao gồm việc:

- Thiết lập độ phân giải mong muốn.
- Chọn định dạng ảnh (thường là JPEG).
- Cài đặt một số tham số mặc định như độ sáng, độ tương phản...

2. Khi khởi tạo thành công, camera đã sẵn sàng để chụp.
3. **Nhận yêu cầu chụp ảnh**  
ESP32-CAM có thể tự quyết định thời điểm chụp (ví dụ chụp liên tục mỗi vài giây) hoặc nhận yêu cầu từ hệ thống. Trong đề tài, thường là khi người dùng vừa quét thẻ – ESP32 hoặc máy chủ gửi một lệnh “chụp ảnh” đến ESP32-CAM để ghi lại khuôn mặt tại thời điểm đó.
4. **Chụp và lấy dữ liệu ảnh thô**  
Khi nhận lệnh, cảm biến OV2640 thu một khung hình và đẩy dữ liệu ảnh thô (raw) về bộ nhớ của ESP32-CAM. Dữ liệu thô này chưa được nén, dung lượng còn rất lớn.
5. **Nén ảnh sang JPEG**  
Để có thể truyền qua Wi-Fi một cách nhanh chóng, dữ liệu ảnh thô được xử lý thông qua bộ mã hóa JPEG tích hợp. Sau bước này, ta thu được một mảng byte ảnh JPEG có kích thước nhỏ hơn rất nhiều.
6. **Đóng gói và gửi lên máy chủ**  
ESP32-CAM có thể truyền ảnh:
  - Dưới dạng nhị phân qua WebSocket.
  - Hoặc chuyển sang chuỗi base64 và gửi qua MQTT/HTTP.
7. Kèm theo ảnh có thể có thông tin như ID thiết bị, thời gian, ID sự kiện để máy chủ liên kết ảnh với đúng lần quét thẻ tương ứng.
8. **Xử lý trên máy chủ**  
Máy chủ nhận ảnh, giải mã JPEG thành ma trận pixel. Từ đó, các thuật toán thị giác máy tính và AI sẽ tiếp tục phát hiện và nhận diện khuôn mặt người dùng.

Trong toàn bộ quá trình, ESP32-CAM chỉ chịu trách nhiệm từ khâu chụp tới khâu nén và gửi ảnh. Những bước nặng như nhận diện, so khớp embedding được thực hiện ở hệ thống có sức mạnh tính toán lớn hơn.

## 2.3 Đầu đọc thẻ RFID RC522



**TH** TÂN HƯNG HÀ  
Barcode • Printing • POS • Automation

Hình 3 Modul đọc thẻ từ RC522

### 2.3.1 Nguyên lý RFID

RFID (Radio Frequency Identification) là công nghệ cho phép nhận dạng đối tượng dựa trên sóng vô tuyến. Về bản chất, hệ thống RFID gồm hai thành phần:

- Thẻ (tag), thường là thẻ thụ động không có nguồn điện riêng. Bên trong thẻ chứa:
  - Một anten dạng cuộn dây.
  - Một con chip nhỏ lưu trữ thông tin (như UID, một số block dữ liệu).
- Đầu đọc (reader), phát ra sóng radio ở tần số nhất định (ví dụ 13,56 MHz) và lắng nghe tín hiệu phản hồi từ thẻ.

Khi thẻ tiến lại gần đầu đọc, anten của thẻ nằm trong vùng trường điện từ của reader. Năng lượng từ trường này được “hút” vào cuộn dây, tạo ra dòng điện nhỏ cung cấp cho chip trong thẻ hoạt động. Chip sau đó mã hóa thông tin (UID, dữ liệu...) và điều chế lại tín hiệu, gửi ngược về đầu đọc bằng kỹ thuật gọi là backscatter.

Ưu điểm của RFID là:

- Không cần tiếp xúc vật lý, chỉ cần đưa thẻ vào vùng đọc.

- Có thể đọc khá nhanh, phù hợp cho việc kiểm soát ra vào.
- Không bị ảnh hưởng nhiều bởi bụi bẩn, trầy xước như mã vạch.

Trong đề tài, RFID được sử dụng để xác định thẻ của người dùng trước khi hoặc song song với bước nhận diện khuôn mặt.

### 2.3.2 Cấu tạo và chức năng module RC522

RC522 là một module đầu đọc RFID phổ biến cho các ứng dụng nhỏ. Trên module này có:

- Con chip MFRC522, chịu trách nhiệm:
  - Tạo sóng radio ở tần số 13,56 MHz.
  - Thực hiện các thủ tục giao tiếp với thẻ chuẩn MIFARE.
  - Mã hóa/giải mã dữ liệu, xử lý khung truyền.
- Anten in trực tiếp trên mạch in (PCB), đóng vai trò phát và thu sóng.
- Các chân giao tiếp với vi điều khiển (VCC, GND, RST, NSS, MOSI, MISO, SCK...).

Khi được cấp nguồn và khởi tạo đúng cách, RC522 sẽ duy trì một trường điện từ quanh anten. Bất cứ thẻ MIFARE nào đi vào vùng này sẽ có khả năng được kích hoạt và trao đổi dữ liệu với RC522.

Trong hệ thống khóa cửa, RC522 chính là “đầu đọc thẻ” đặt gần tay người dùng. Chỉ cần đưa thẻ tới gần anten, hệ thống có thể đọc được mã UID để xác định người dùng đó là ai.

### 2.3.3 Giao tiếp giữa RC522 và ESP32

Để RC522 và ESP32 “nói chuyện” được với nhau, hai bên sử dụng chuẩn SPI – một giao thức truyền dữ liệu nối tiếp đồng bộ. Trong giao tiếp SPI, ESP32 đóng vai trò master, còn RC522 là slave.

Các chân cơ bản của SPI gồm:

- MOSI (Master Out Slave In): dữ liệu từ ESP32 đi vào RC522.
- MISO (Master In Slave Out): dữ liệu từ RC522 gửi ngược về ESP32.
- SCK (Serial Clock): xung đồng hồ do ESP32 tạo ra để đồng bộ truyền nhận.
- SS/CS (Slave Select): chân chọn thiết bị; khi chân này của RC522 được kéo xuống mức thấp, ESP32 “nói chuyện” với RC522, nếu không thì RC522 sẽ bỏ qua dữ liệu trên đường MOSI/SCK.

Quy trình đọc UID thẻ từ là:

1. ESP32 khởi tạo SPI và cấu hình các thanh ghi bên trong RC522 để bật anten, chọn chế độ làm việc với thẻ MIFARE.
2. Trong vòng lặp, ESP32 gửi lệnh “tìm thẻ” (request) tới RC522. Nếu không có thẻ trong vùng anten, RC522 trả về trạng thái không tìm thấy.
3. Khi có thẻ xuất hiện, RC522 sẽ phát hiện và báo cho ESP32; tiếp theo, hai bên thực hiện thủ tục anti-collision để chọn ra một thẻ nếu có nhiều thẻ cùng lúc.
4. Sau đó, ESP32 gửi lệnh yêu cầu đọc UID, RC522 sẽ trả về chuỗi số UID tương ứng với thẻ đang chọn.
5. UID này được ESP32 lưu lại, đóng gói và gửi lên hệ thống xử lý trung tâm để kiểm tra quyền truy cập.

Nhờ RC522, hệ thống có thể phân biệt từng thẻ một cách chính xác, tạo nên phương thức xác thực nhanh và quen thuộc đối với người dùng.

## **2.4 Cơ cấu chấp hành: khóa điện từ, servo và relay**

### **2.4.1 Khóa điện từ (solenoid lock)**

Khóa điện từ là loại khóa trong đó việc mở hay khóa cửa được thực hiện nhờ lực hút của nam châm điện, chứ không phải hoàn toàn bằng cơ khí như ổ khóa truyền thống. Bên trong khóa có:

- Một cuộn dây quấn quanh lõi sắt.
- Một chốt cơ khí gắn với lõi hoặc chịu ảnh hưởng trực tiếp từ lực từ.
- Một lò xo để đưa chốt về trạng thái ban đầu khi mất điện.

Khi cấp điện cho cuộn dây, dòng điện chạy qua tạo ra từ trường; từ trường này hút lõi sắt và kéo chốt khóa. Tùy theo thiết kế, khi cuộn dây có điện, chốt có thể:

- Được kéo vào trong, tạo trạng thái “mở” (cửa mở ra được).
- Hoặc được giữ ở vị trí “khóa”.

Ngược lại, khi ngắt điện, từ trường biến mất, lò xo sẽ đẩy lõi và chốt về trạng thái ban đầu.

Trong các hệ thống kiểm soát truy cập, thường tồn tại hai kiểu:

- Fail-safe: khi mất điện, cửa tự mở (ưu tiên an toàn tính mạng, ví dụ trong cháy nổ).

- Fail-secure: khi mất điện, cửa vẫn khóa (ưu tiên an ninh, chống đột nhập).

Trong đề tài, khóa điện từ nhận nguồn 12V hoặc 24V riêng. ESP32 không thể cấp trực tiếp dòng cho khóa, mà chỉ điều khiển relay để đóng/ngắt nguồn cấp cho solenoid.

#### 2.4.2 Servo motor trong cơ cấu khóa

Servo là một loại động cơ đặc biệt, cho phép điều khiển **góc quay cụ thể** thay vì quay liên tục như động cơ DC. Bên trong một servo nhỏ thường có:

- Động cơ DC.
- Hộp số giảm tốc để tăng mô-men.
- Mạch điều khiển.
- Cảm biến vị trí (thường là biến trở gắn với trục).

Servo nhận lệnh điều khiển thông qua tín hiệu PWM:

Trong một chu kỳ khoảng 20 ms, độ rộng xung (thời gian ở mức cao) sẽ tương ứng với một góc quay. Ví dụ, xung rộng 1 ms tương ứng  $0^\circ$ , xung rộng 1,5 ms tương ứng  $90^\circ$ , xung rộng 2 ms tương ứng  $180^\circ$  (con số cụ thể tùy loại servo).

Trong hệ thống khóa cửa, servo có thể được dùng để xoay chốt khóa cơ khí của một ổ khóa sẵn có. Khi cần mở cửa, ESP32 phát xung PWM tương ứng với góc quay mở; khi đóng, ESP32 phát xung đưa servo về góc ban đầu. Servo phù hợp khi muốn nâng cấp một ổ khóa cơ mà không thay toàn bộ sang khóa điện từ.

Tuy nhiên, servo cần nguồn 5V đủ dòng, và nếu nguồn không đủ khỏe, servo có thể quay yếu, rung hoặc reset.

#### 2.4.3 Module relay

Relay là một loại công tắc điện được điều khiển bằng điện. Một module relay hoàn chỉnh thường gồm:

- Cuộn dây và tiếp điểm relay.
- Một transistor hoặc mạch driver để nâng dòng từ vi điều khiển.
- Diode bảo vệ để chống điện áp ngược khi tắt cuộn dây.
- Các chân kết nối tín hiệu điều khiển (IN), nguồn (VCC) và mass (GND).

ESP32 chỉ có thể xuất ra điện áp 3,3V với dòng rất nhỏ (vài mA), không đủ để cấp trực tiếp cho khóa điện từ hay servo công suất lớn. Vì vậy, vi điều khiển sẽ:

- Xuất tín hiệu mức cao hoặc thấp tại chân GPIO.

- Tín hiệu này điều khiển transistor trên module relay, làm cuộn dây relay có hoặc không có điện.
- Khi cuộn dây relay có điện, tiếp điểm đóng lại, kết nối nguồn 12V/24V với khóa điện từ.
- Khi cuộn dây relay mất điện, tiếp điểm mở ra, khóa điện từ mất nguồn và trở về trạng thái ban đầu.

Trong hệ thống khóa cửa, quy trình điều khiển điển hình là:

1. ESP32 nhận lệnh mở cửa (sau khi xác thực thành công).
2. Chân điều khiển relay được đưa lên mức kích hoạt, relay đóng tiếp điểm, cấp điện cho khóa điện từ.
3. Sau khoảng thời gian đã cấu hình (ví dụ 3 giây), ESP32 đưa chân điều khiển về lại trạng thái ban đầu, relay ngắt nguồn, khóa đóng.

Relay đóng vai trò “cầu nối” giữa thế giới logic mức thấp (3,3V) của ESP32 và thế giới tải công suất cao (12V/24V) của khóa điện từ.

## 2.5 Hệ thống nhúng và tổ chức phần mềm trên ESP32

### 2.5.1 Khái niệm hệ thống nhúng

Hệ thống nhúng là hệ thống trong đó một máy tính được “nhúng” vào bên trong một thiết bị vật lý để thực hiện các chức năng chuyên dụng. Khác với máy tính cá nhân, người dùng thường không thấy trực tiếp “máy tính” trong đó, mà chỉ thấy thiết bị hoạt động theo chức năng của nó: máy giặt tự chọn chế độ giặt, lò vi sóng điều khiển thời gian, ABS trên ô tô kiểm soát phanh, v.v.

Đặc điểm của hệ thống nhúng:

- Tài nguyên phần cứng hạn chế: CPU, RAM, bộ nhớ, năng lượng ít hơn máy tính để bàn.
- Chạy một tập chương trình cố định, ít khi cài phần mềm từ bên ngoài.
- Thường phải hoạt động liên tục, độ tin cậy cao; hỏng hóc có thể gây hậu quả lớn (khóa cửa không mở được, phanh ABS không hoạt động...).

Trong đề tài, cụm ESP32 + ESP32-CAM + RC522 + khóa điện từ/servo chính là một hệ thống nhúng nhỏ nằm trong cánh cửa, làm nhiệm vụ kiểm soát ra vào theo đúng logic đã lập trình.

### 2.5.2 FreeRTOS trên ESP32

FreeRTOS là một hệ điều hành thời gian thực (Real-Time Operating System) nhẹ, được tích hợp sẵn bên trong ESP32. FreeRTOS cung cấp các cơ chế:

- Tạo và quản lý task (tương tự “tiến trình nhỏ”): mỗi task là một hàm chạy vòng lặp riêng, có thể ngủ và được đánh thức theo lịch.
- Lập lịch dựa trên độ ưu tiên: task quan trọng hơn có thể được xử lý trước để đảm bảo thời gian đáp ứng.
- Hàng đợi (queue), semaphore, mutex để các task trao đổi dữ liệu, tránh tranh chấp khi cùng truy cập một tài nguyên chung (ví dụ UART, SPI, biến toàn cục...).

Lợi ích khi dùng FreeRTOS trên ESP32:

- Có thể tách riêng từng chức năng thành một task độc lập: đọc RFID, giao tiếp mạng, điều khiển khóa, giám sát,... giúp chương trình dễ hiểu, dễ bảo trì.
- Tránh việc viết một vòng **loop()** duy nhất quá dài, chứa nhiều **delay()**, dễ làm hệ thống phản hồi chậm hoặc bị treo.
- Dễ thêm, bớt chức năng mà không ảnh hưởng nhiều tới cấu trúc tổng thể.

### 2.5.3 Tổ chức phần mềm trên ESP32 trong đề tài

Một cách tổ chức phần mềm hợp lý cho đề tài có thể như sau:

- Task đọc thẻ RFID:  
Task này phụ trách toàn bộ quá trình giao tiếp với RC522. Nó liên tục dò xem có thẻ mới xuất hiện hay không. Khi đọc được một UID mới, task sẽ đẩy UID đó vào một queue gửi tới task giao tiếp mạng. Nhờ dùng queue, ta tránh việc hai task cùng thao tác với cùng một biến toàn cục, giảm nguy cơ lỗi.
- Task giao tiếp mạng:  
Task này thiết lập và duy trì kết nối Wi-Fi, đồng thời gửi nhận dữ liệu với máy chủ qua MQTT/WebSocket.
  - Khi có UID mới trong queue, task đọc ra, đóng gói thành gói tin và gửi lên máy chủ.
  - Khi máy chủ trả về kết quả (cho phép/không cho phép, yêu cầu chụp ảnh...), task sẽ nhận và gửi tín hiệu sang task điều khiển khóa.

- Task điều khiển khóa:  
Task này ít khi bận, nhưng rất quan trọng vì liên quan trực tiếp đến việc mở/đóng cửa.
  - Khi nhận yêu cầu mở cửa, task kích chân điều khiển relay hoặc servo, đồng thời bắt đầu đếm thời gian.
  - Sau khi hết thời gian cho phép, task tự động hết thời gian mở và trả khóa về trạng thái bình thường.
  - Task này có thể được đặt độ ưu tiên cao, để đảm bảo lệnh mở cửa được thực hiện kịp thời.
- Task giám sát (monitoring):  
Task này có nhiệm vụ “trông coi” toàn hệ thống:
  - Kiểm tra trạng thái Wi-Fi, nếu mất kết nối thì thử kết nối lại.
  - Theo dõi số lần thẻ quẹt sai liên tiếp.
  - Ghi log cục bộ nếu cần (ví dụ lên thẻ SD).

Cách chia task rõ ràng giúp việc phát triển, sửa lỗi và mở rộng hệ thống sau này thuận lợi hơn rất nhiều.

## 2.6 Mạng không dây và các giao thức truyền thông sử dụng

### 2.6.1 Wi-Fi 802.11

Wi-Fi là công nghệ mạng không dây sử dụng sóng vô tuyến ở các băng tần như 2,4 GHz hoặc 5 GHz để truyền dữ liệu. Trong đề tài, Wi-Fi đóng vai trò là đường kết nối giữa các thiết bị (ESP32, ESP32-CAM) và máy chủ.

Một số đặc điểm liên quan:

- Băng tần 2,4 GHz được sử dụng rộng rãi, tương thích với hầu hết router và đặc biệt là được ESP32 hỗ trợ.
- Tốc độ truyền của các chuẩn 802.11b/g/n dư sức phục vụ việc truyền gói tin nhỏ (UID thẻ) và ảnh JPEG có kích thước trung bình.
- Khoảng cách và chất lượng tín hiệu phụ thuộc vào môi trường (tường, kim loại, nhiều), nên vị trí đặt router và thiết bị cần được cân nhắc để đảm bảo độ ổn định kết nối.

Trong hệ thống khóa cửa, việc sử dụng Wi-Fi thay vì dây Ethernet giúp giảm số dây phải đi, dễ lắp đặt hơn, đặc biệt là khi muốn đặt khóa ở những vị trí khó kéo cáp.

### 2.6.2 Giao thức MQTT

MQTT là giao thức truyền thông nhẹ, được thiết kế cho các ứng dụng có băng thông hạn chế và thiết bị tài nguyên thấp. Thay vì mô hình client–server như HTTP, MQTT sử dụng mô hình publish/subscribe:

- Tất cả các thiết bị kết nối tới một broker trung tâm.
- Thiết bị có thể publish (đăng) thông điệp lên một “topic”.
- Thiết bị khác subscribe (đăng ký) topic đó sẽ nhận được thông điệp.

Trong đề tài, có thể hình dung:

- Thiết bị khóa cửa publish sự kiện “thẻ được quét” lên một topic nào đó .
- Máy chủ subscribe topic đó nên sẽ nhận được mọi sự kiện từ cửa số 01.
- Ngược lại, khi muốn mở cửa, máy chủ publish thông điệp “unlock” lên topic và thiết bị tại cửa subscribe topic này để nhận lệnh.

Ưu điểm của MQTT:

- Gói tin nhỏ, ít overhead, thích hợp cho môi trường mạng không dây.
- Dễ mở rộng nếu sau này có nhiều cửa, nhiều thiết bị cùng tham gia.

### 2.6.3 Giao thức WebSocket

WebSocket là giao thức cho phép thiết lập một kết nối TCP lâu dài giữa client và server. Sau khi kết nối được thiết lập, cả hai bên có thể chủ động gửi dữ liệu bất cứ lúc nào, không cần theo mô hình request–response như HTTP.

Điều này rất hữu ích với hệ thống khóa cửa:

- Thiết bị có thể đẩy ngay lập tức sự kiện mới (thẻ được quét, cửa đang mở...) lên máy chủ.
- Máy chủ có thể trả lời gần như ngay lập tức với lệnh mở/không mở, hoặc yêu cầu chụp ảnh.
- Độ trễ thấp giúp trải nghiệm người dùng mượt, không phải chờ lâu trước cửa.

WebSocket có thể dùng trực tiếp giữa ESP32 và server, hoặc giữa server và giao diện giám sát (ví dụ web hiển thị log thời gian thực).

### 2.6.4 HTTP và REST (ở mức khái niệm)

HTTP là giao thức truyền dữ liệu dạng văn bản phổ biến nhất trên Internet. REST là phong cách thiết kế dịch vụ web dựa trên HTTP, trong đó mỗi tài nguyên

được đại diện bằng một URL, và các thao tác lên tài nguyên được ánh xạ với các phương thức HTTP (GET, POST, PUT, DELETE...).

Trong hệ thống khóa cửa, HTTP/REST chủ yếu được dùng để:

- Xây dựng các API phục vụ cấu hình hệ thống (thêm/xóa người dùng, thẻ, ngưỡng an toàn...).
- Cung cấp dữ liệu lịch sử cho các phần mềm quản lý (lấy danh sách các lần ra/vào trong ngày, thống kê...).

Ở phần nền tảng lý thuyết, ta chỉ dừng ở việc hiểu HTTP/REST là cách chuẩn để cung cấp dịch vụ cho các ứng dụng khác, còn chi tiết triển khai web không nằm trong phạm vi chương này.

## **2.7 Trí tuệ nhân tạo và thị giác máy tính trong nhận diện khuôn mặt**

### **2.7.1 Giới thiệu chung**

Trong hệ thống khóa cửa thông minh, trí tuệ nhân tạo và thị giác máy tính được sử dụng để tự động nhận diện người dùng dựa trên khuôn mặt. Thay vì chỉ dùng thẻ RFID hoặc mật khẩu, hệ thống kết hợp thêm nhận diện khuôn mặt để tăng mức độ an toàn và tiện lợi cho người sử dụng.

Các nhiệm vụ chính của phần nhận diện khuôn mặt:

- Nhận ảnh đầu vào do ESP32-CAM gửi lên (ảnh có thể có một hoặc nhiều khuôn mặt).
- Phát hiện và xác định vị trí các khuôn mặt trong ảnh.
- Cắt và căn chỉnh từng khuôn mặt về kích thước, tư thế chuẩn.
- Trích xuất đặc trưng và so sánh với dữ liệu đã đăng ký để xác định danh tính hoặc kết luận là người lạ.

So với mật khẩu hay thẻ từ, cơ chế này:

- Khó bị “mượn” hoặc bị đánh cắp hơn, do gắn liền với đặc điểm sinh trắc học của người dùng.
- Không yêu cầu người dùng mang theo thêm vật dụng, chỉ cần đứng trước camera để được xác thực.

Tuy nhiên, dữ liệu khuôn mặt là dữ liệu nhạy cảm, cần được xử lý, truyền tải và lưu trữ an toàn để bảo vệ quyền riêng tư cho người sử dụng.

### 2.7.2 Mạng nơ-ron tích chập (Convolutional Neural Network – CNN)

CNN là kiến trúc mạng nơ-ron được thiết kế chuyên cho xử lý ảnh. Mạng khai thác cấu trúc không gian của ảnh để học ra các đặc trưng quan trọng phục vụ phân loại hoặc nhận dạng.

#### **Ý tưởng chính của CNN:**

- Ảnh được biểu diễn dưới dạng ma trận điểm ảnh (pixel).
- Các lớp tích chập sử dụng các kernel nhỏ trượt trên ảnh, tạo ra các bản đồ đặc trưng biểu diễn cạnh, góc, họa tiết, vùng sáng – tối.
- Các lớp pooling (như max pooling) giảm kích thước không gian của đặc trưng, giúp giảm số tham số, giảm overfitting và tăng tốc độ tính toán.
- Ở cuối mạng, một hoặc vài lớp fully connected tổng hợp các đặc trưng và đưa ra dự đoán hoặc vector đặc trưng cuối.

Trong bài toán nhận diện khuôn mặt của đề tài, CNN được sử dụng để:

- Là lõi của mô hình phát hiện khuôn mặt (MTCNN), giúp tìm chính xác vị trí khuôn mặt trên ảnh gốc.
- Là mô hình trích xuất đặc trưng (embedding), biến mỗi khuôn mặt đã cắt thành một vector số có số chiều cố định, dùng cho bước so sánh và nhận diện.

### 2.7.3 Phát hiện khuôn mặt bằng MTCNN (Multi-task Cascaded Convolutional Networks)

Phát hiện khuôn mặt (face detection) là bước tiền xử lý giúp hệ thống xác định vùng chứa khuôn mặt trên ảnh toàn cảnh. Nếu phát hiện sai hoặc cắt lệch, chất lượng nhận diện phía sau sẽ giảm đáng kể. Trong đề tài, nhóm sử dụng MTCNN để thực hiện bước này.

#### **Đặc điểm của MTCNN:**

- Là kiến trúc gồm ba mạng CNN nhỏ xếp tầng: P-Net, R-Net, O-Net.
- Mỗi mạng vừa dự đoán khung bao khuôn mặt, vừa dự đoán các điểm mốc (mắt, mũi, miệng), nên được gọi là “multi-task”.
- Làm việc trên ảnh ở nhiều tỉ lệ, có khả năng phát hiện cả các khuôn mặt nhỏ hoặc hơi nghiêng.

Quy trình phát hiện khuôn mặt bằng MTCNN trong hệ thống:

- Ảnh JPEG từ ESP32-CAM được giải mã và đưa vào P-Net để quét trên nhiều tỉ lệ, sinh ra các vùng ứng viên có khả năng chứa khuôn mặt.

- Các vùng ứng viên được lọc sơ bộ, sau đó cắt ra và đưa vào R-Net để loại bỏ thêm các vùng nhiễu, đồng thời tinh chỉnh vị trí, kích thước khung bao.
- Những vùng vượt qua R-Net được đưa vào O-Net để đánh giá cuối cùng, thu được danh sách khung bao khuôn mặt và các điểm mốc chính xác hơn.
- Dựa trên khung bao và điểm mốc do O-Net trả về, hệ thống cắt khuôn mặt ra khỏi ảnh gốc, xoay và căn chỉnh sao cho khuôn mặt ở tư thế chuẩn, phục vụ bước trích xuất đặc trưng.

Nhờ sử dụng MTCNN, hệ thống có khả năng phát hiện khuôn mặt ổn định hơn trong điều kiện thực tế: người dùng có thể đứng hơi lệch, ánh sáng không đồng đều, khoảng cách tới camera thay đổi,...

#### 2.7.4 Embedding khuôn mặt và so sánh khoảng cách

Sau khi khuôn mặt đã được phát hiện và căn chỉnh, hệ thống cần xác định xem khuôn mặt đó thuộc về người nào. Thay vì so sánh trực tiếp từng điểm ảnh, hệ thống sử dụng khái niệm embedding khuôn mặt để biểu diễn khuôn mặt dưới dạng vector số.

Embedding khuôn mặt là:

- Một vector đặc trưng có số chiều cố định (ví dụ 128 hoặc 512 chiều).
- Do một mạng CNN chuyên cho nhận diện khuôn mặt sinh ra.
- Mang thông tin về hình dạng và đặc điểm riêng của khuôn mặt, nhưng đã được chuẩn hóa dưới dạng số.

Quy trình trích xuất và sử dụng embedding:

- Ảnh khuôn mặt sau khi cắt và căn chỉnh được resize về kích thước chuẩn, chuẩn hóa giá trị pixel.
- Ảnh chuẩn hóa được đưa qua mạng CNN đã huấn luyện (ví dụ các họ mô hình FaceNet, ArcFace...), đầu ra là vector embedding biểu diễn cho khuôn mặt đó.
- Khi đăng ký người dùng mới, hệ thống chụp một số ảnh mặt, trích xuất embedding và lưu vào cơ sở dữ liệu gắn với ID người dùng.
- Khi cần xác thực, ảnh mới cũng được đưa qua pipeline trên để thu được embedding mới.
- Hệ thống tính khoảng cách giữa embedding mới và các embedding đã lưu (thường dùng khoảng cách Euclid hoặc cosine).

- Nếu khoảng cách nhỏ nhất nhỏ hơn một ngưỡng đặt trước, kết luận khuôn mặt thuộc về người dùng tương ứng; nếu không, kết luận là người lạ.

Việc chỉ lưu embedding thay vì lưu toàn bộ ảnh khuôn mặt giúp giảm dung lượng lưu trữ và tăng mức độ bảo mật, vì rất khó tái tạo lại ảnh gốc từ embedding, qua đó hạn chế rủi ro lộ trực tiếp hình ảnh khuôn mặt của người dùng.

## 2.8 Bảo mật và an toàn thông tin trong hệ thống khóa cửa thông minh

### 2.8.1 Các nguy cơ bảo mật tiềm ẩn

Hệ thống khóa cửa thông minh xử lý nhiều loại thông tin nhạy cảm:

- UID của thẻ ra vào.
- Dữ liệu khuôn mặt và embedding.
- Lịch sử ra/vào (ai, mở cửa lúc mấy giờ, tại cửa nào).

Nếu không được bảo vệ tốt, hệ thống có thể bị tấn công theo nhiều cách:

- Tấn công kênh truyền: kẻ xấu nghe lén gói tin để lấy UID thẻ hoặc chen thông điệp giả vào dòng dữ liệu nhằm mở cửa trái phép.
- Tấn công thiết bị: truy cập trái phép vào ESP32, ESP32-CAM, cắm trực tiếp vào dây điều khiển relay để kích khóa, hoặc nạp firmware độc hại.
- Tấn công dữ liệu: đánh cắp cơ sở dữ liệu lưu embedding khuôn mặt, danh sách thẻ và lịch sử ra/vào; chỉnh sửa hoặc xóa log nhằm xóa dấu vết hoạt động.

Vì vậy, ngay từ phần nền tảng lý thuyết, đề tài cần xác định rõ các nguyên tắc bảo mật cơ bản sẽ được áp dụng.

### 2.8.2 Các nguyên tắc bảo mật cơ bản

Một số nguyên tắc chung cần được tuân thủ:

- **Bảo mật đường truyền:**  
Nên ưu tiên sử dụng các kết nối có mã hóa (ví dụ dùng TLS đối với HTTP/WebSocket) để kẻ tấn công dù nghe lén được dữ liệu cũng khó giải mã. Không nên truyền các thông tin quan trọng như mật khẩu, token ở dạng rõ.
- **Xác thực thiết bị:**  
Mỗi thiết bị khóa cửa nên có một định danh và khóa bí mật riêng. Khi thiết bị kết nối tới máy chủ, hai bên cần xác thực lẫn nhau, tránh trường hợp kẻ xấu giả làm thiết bị thật để gửi yêu cầu giả.
- **Phân quyền người dùng:**  
Hệ thống cần phân biệt rõ:

- Người quản trị (có quyền thêm/xóa tài khoản, thẻ, dữ liệu khuôn mặt).
- Người dùng thường (chỉ có quyền mở cửa trong giới hạn, xem lịch sử của chính mình).

- **Ghi log và giám sát:**

Những hành vi nhạy cảm như đăng nhập thất bại, quẹt thẻ sai nhiều lần, mở cửa ngoài giờ, v.v. cần được ghi lại và có thể sinh cảnh báo.

Áp dụng các nguyên tắc này sẽ giúp giảm đáng kể nguy cơ bị tấn công, đồng thời hỗ trợ việc truy vết khi có sự cố.

### 2.8.3 Bảo vệ dữ liệu sinh trắc học

Dữ liệu khuôn mặt là dữ liệu sinh trắc học, khác với mật khẩu ở chỗ:

- Không thể dễ dàng “đổi” mặt như đổi mật khẩu nếu lỡ bị lộ.
- Có thể ảnh hưởng tới quyền riêng tư lâu dài của người dùng.

Do đó, việc bảo vệ dữ liệu khuôn mặt cần lưu ý:

- **Hạn chế lưu ảnh gốc:**

Chỉ lưu khi thật cần thiết (ví dụ dùng cho mục đích giám sát, điều tra) và phải có chính sách xóa sau một khoảng thời gian. Việc lưu embedding đặc trưng đủ cho bài toán nhận diện mà không cần giữ toàn bộ ảnh.

- **Kiểm soát truy cập:**

Chỉ những tài khoản có quyền cao mới được xem hoặc thao tác với dữ liệu khuôn mặt; phía người dùng bình thường chỉ thấy được thông tin cần thiết.

- **Quyền được quên:**

Khi một người dùng không còn sử dụng hệ thống nữa, cần có cơ chế xóa dữ liệu liên quan đến họ (thẻ, embedding, lịch sử), nếu điều đó phù hợp với yêu cầu của bài toán và quy định pháp luật.

Việc xây dựng hệ thống trên nền tảng lý thuyết bảo mật và quyền riêng tư rõ ràng không chỉ giúp đề tài “đẹp” hơn về mặt học thuật, mà còn quan trọng nếu trong tương lai mô hình này được triển khai thực tế.

### 3 PHÂN TÍCH YÊU CẦU CHỨC NĂNG

#### 3.1 Chức năng xác thực người dùng

Hệ thống khóa cửa thông minh hỗ trợ hai cơ chế xác thực chính, hoạt động độc lập hoặc kết hợp:

1. Xác thực bằng thẻ từ RFID
  - Module RC522 đọc UID của thẻ.
  - Mỗi thẻ được gán tương ứng với một người dùng trong cơ sở dữ liệu.
  - UID được gửi lên server thông qua kết nối WebSocket.
  - Server kiểm tra hợp lệ → trả về trạng thái truy cập.
2. Xác thực bằng khuôn mặt
  - Camera ESP32-CAM chụp ảnh khi phát hiện chuyển động hoặc có người đứng trước cửa.
  - Ảnh được xử lý trực tiếp trên ESP32-CAM (so sánh embedding) hoặc gửi lên server tùy cấu hình.
  - Server kiểm tra xem khuôn mặt có trùng khớp người dùng hay không.
3. Chế độ xác thực linh hoạt
  - Chế độ tiện lợi (OR): chỉ cần thẻ *hoặc* khuôn mặt hợp lệ.
  - Chế độ an ninh cao (AND): cần xác thực *cả thẻ và khuôn mặt*.
4. Kiểm tra điều kiện truy cập
  - Kiểm tra trạng thái thiết bị (ESP32 online/offline).
  - Kiểm tra người dùng có bị khóa quyền truy cập hay không.
  - Kiểm tra giới hạn số lần nhập sai.

#### 3.2 Chức năng mở khóa cửa

Sau khi hệ thống xác thực thành công, server gửi lệnh điều khiển đến ESP32 qua WebSocket:

- Lệnh OPEN: kích hoạt Relay tạo xung điện để mở khóa.
- Thời gian kích hoạt: từ 1–3 giây theo cấu hình.

- ESP32 phản hồi trạng thái lại server: *DoorOpened* hoặc *DoorError*.

Relay/Servo đóng vai trò như một bộ chấp hành (Actuator).

### 3.3 Chức năng xem thông tin truy cập theo thời gian thực

Người dùng có thể xem:

- Trạng thái cửa (đang khóa / đang mở)
- Trạng thái thiết bị (online / offline)
- Phương thức đang được kích hoạt (RFID, FaceID hoặc cả hai)
- Log truy cập theo thời gian thực (ai – lúc nào – vì lý do gì)

Dữ liệu được cập nhật qua WebSocket và hiển thị trên dashboard với:

- Bảng danh sách log
- Biểu đồ thống kê
- Thông báo realtime

### 3.4 Chức năng cảnh báo an ninh

Hệ thống hỗ trợ nhận diện bất thường:

1. Sai thẻ quá số lần giới hạn ( $\geq 5$  lần)
2. Sai khuôn mặt quá nhiều lần liên tiếp
3. Cố gắng mở cửa khi thiết bị không ghi nhận thẻ/khuôn mặt
4. Thao tác bất thường từ WebUI (mở cửa không đúng phân quyền)

Khi xảy ra sự kiện cảnh báo:

- Server kích hoạt sự kiện “ALARM EVENT” qua WebSocket.
- Ghi log đặc biệt trong CSDL.
- Tùy cấu hình, gửi tin nhắn đến Gmail.

### 3.5 Chức năng quản lý người dùng

Quản trị viên có thể:

- Thêm người dùng mới
- Xóa người dùng
- Gắn thẻ RFID
- Tải ảnh khuôn mặt cho việc huấn luyện

Hệ thống đảm bảo khả năng quản lý linh hoạt, hỗ trợ nhiều người dùng, phù hợp mô hình văn phòng – chung cư.

### **3.6 Chức năng điều khiển từ xa**

Hệ thống khóa cửa hỗ trợ:

- Mở cửa từ xa
- Tắt chế độ xác thực từ xa
- Chỉnh lại chế độ AND/OR
- Nhận thông báo khi cửa mở từ xa

Các thao tác đều được ghi lại vào log để đảm bảo tính truy vết (traceability).

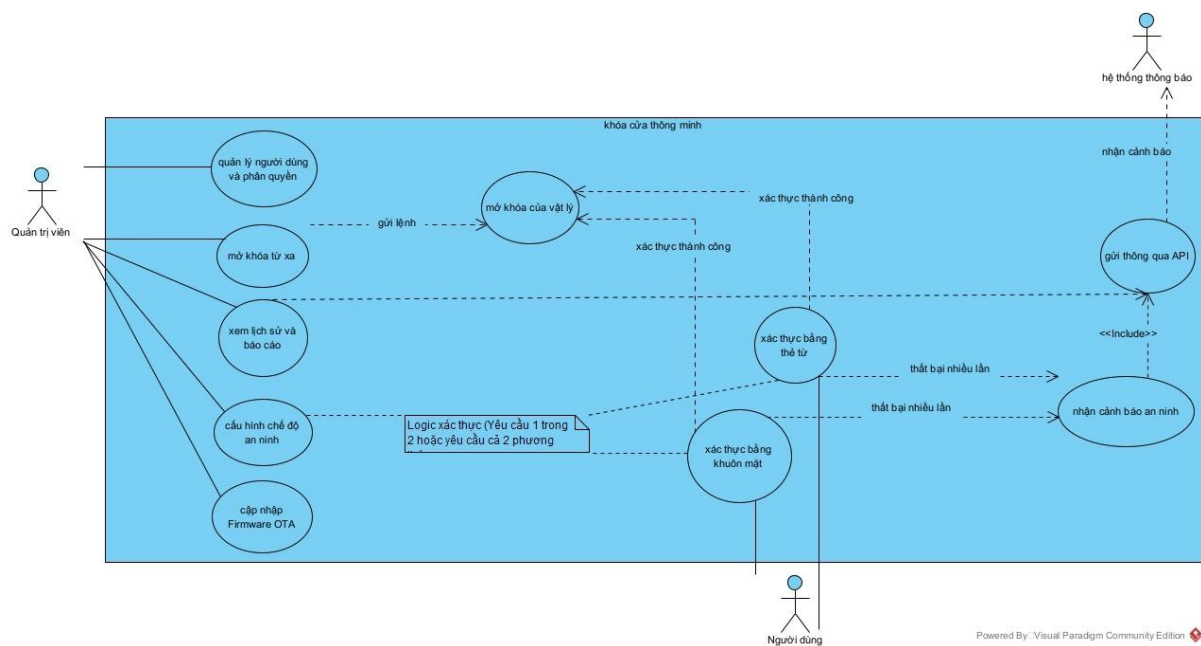
### **3.7 Chức năng xem và xuất dữ liệu**

Hệ thống cho phép:

- Lọc dữ liệu log theo user/ngày/phương thức
- Xuất dữ liệu thành file CSV/Excel
- Xem biểu đồ thống kê theo tuần/tháng

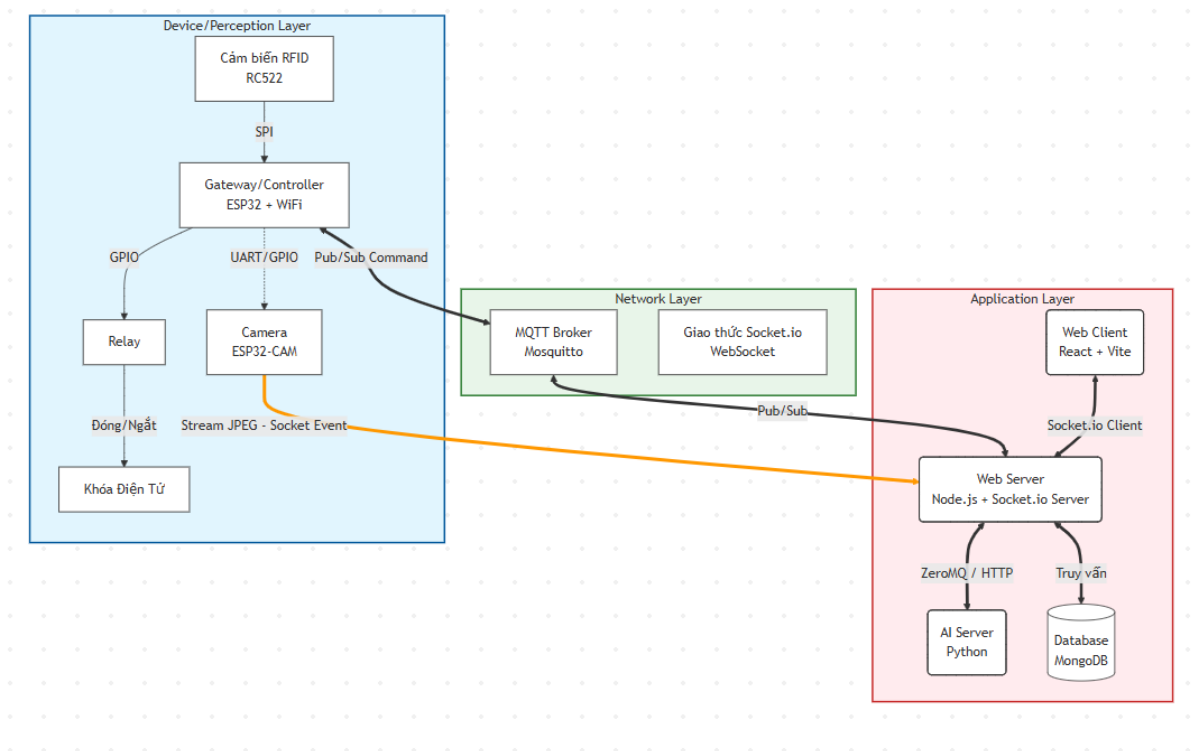
# 4 THIẾT KẾ HỆ THỐNG

## 4.1 Mô hình tổng quan



Hình 4 Biểu đồ usecase mô hình tổng quan hệ thống

## 4.2 Đặc tả mô hình miền



Hình 5 Hình vẽ mô hình miền

#### 4.2.1 Lớp Thiết bị & Nhận thức (Device/Perception Layer)

Thành phần	Vai trò Chính	Giao thức/Kết nối	Mô tả hoạt động
<b>ESP32 (Gateway)</b>	<b>Bộ điều khiển trung tâm</b>	WiFi, SPI, GPIO	Là "bộ não" tại cửa. Nhận tín hiệu từ RFID, kết nối WiFi để gửi lệnh lên Server qua MQTT, và điều khiển Relay.
<b>Cảm biến RFID (RC522)</b>	<b>Thu thập dữ liệu</b>	SPI	Đọc mã thẻ từ và gửi tín hiệu về cho ESP32 chính xử lý.
<b>ESP32-CAM</b>	<b>Thu hình ảnh</b>	Socket.io (WiFi)	Chuyên trách việc quay video. Nó <b>không</b> đi qua ESP32 chính để gửi ảnh mà có đường truyền riêng (màu cam) để stream ảnh trực tiếp.
<b>Relay</b>	<b>Công tắc điện tử</b>	GPIO	Nhận tín hiệu điện từ ESP32 để đóng hoặc ngắt mạch điện.
<b>Khóa Điện Tử</b>	<b>Cơ cấu chấp hành</b>	Dây điện	Mở chốt cửa khi Relay đóng mạch (có điện) và khóa lại khi ngắt mạch.

Bảng 1 Bảng thành phần lớp Thiết bị và Nhận thức

#### 4.2.2 Lớp Mạng (Network Layer)

Giao thức	Loại dữ liệu truyền tải	Đặc điểm	Đường dây trong ảnh
-----------	-------------------------	----------	---------------------

<b>MQTT (Mosquitto)</b>	<b>Dữ liệu điều khiển</b>	Nhẹ, ổn định, băng thông thấp	<b>Đường màu đen.</b> Dùng để gửi ID thẻ, lệnh OPEN, trạng thái cửa (Door Status), và cấu hình AND/OR.
<b>Socket.io</b>	<b>Dữ liệu hình ảnh (Stream)</b>	Nhanh, thời gian thực, băng thông cao	<b>Đường màu cam.</b> Dùng riêng để truyền luồng ảnh JPEG liên tục từ Camera lên Server mà không gây nghẽn mạng điều khiển.

Bảng 2 Bảng thành phần lớp Mạng

#### 4.2.3 Lớp Ứng dụng (Application Layer)

Thành phần	Vai trò Chính	Tương tác với	Mô tả hoạt động
<b>Web Server (Node.js)</b>	<b>Trung tâm xử lý (Hub)</b>	MQTT, Client, AI, DB	Nhận dữ liệu từ mọi nguồn, đưa ra quyết định mở cửa, và điều phối luồng dữ liệu đi các nơi khác.
<b>Web Client (React)</b>	<b>Giao diện người dùng</b>	Server (Socket.io)	Hiển thị video trực tiếp (Live view), nhận thông báo khi có người mở cửa, và cho phép Admin cấu hình hệ thống.
<b>AI Server (Python)</b>	<b>Xử lý trí tuệ nhân tạo</b>	Web Server	Nhận hình ảnh từ Node.js, chạy thuật toán nhận diện khuôn mặt và trả về kết quả (Face ID).
<b>Database (MongoDB)</b>	<b>Lưu trữ dữ liệu</b>	Web Server	Lưu thông tin người dùng (User), thiết bị (Device), và lịch sử ra vào (AccessLog).

Bảng 3 Bảng thành phần lớp Ứng dụng

### 4.3 Đặc tả các thành phần chức năng và hoạt động

#### 4.3.1 Thành phần chức năng

Bảng mô tả chức năng:

Thành phần	Chức năng
ESP32	Bộ xử lý trung tâm, quản lý RFID + Relay
ESP32-CAM	Nhận diện khuôn mặt
RC522	Đọc UID thẻ
Relay/Servo	Điều khiển cơ cấu khóa
WebServer	Xử lý xác thực, phân quyền
Database	Lưu dữ liệu hệ thống
WebUI	Dashboard điều khiển và giám sát

Bảng 4 Bảng mô tả chức năng

#### 4.3.2 Thành phần hoạt động

Hoạt động	Mô tả
RFID Handler	Đọc và gửi UID
Face Handler	Chụp ảnh và so sánh khuôn mặt
ControlHandler	Nhận lệnh mở khóa từ server
LogManager	Ghi log truy cập và cảnh báo
WebSocket Channel	Truyền dữ liệu realtime

Bảng 5 Bảng mô tả thành phần hoạt động

### 4.4 Đặc tả luồng hoạt động

#### 4.4.1 Thiết bị cảm biến và vi xử lý

a. Thư viện sử dụng

Thư viện sử dụng trên ESP32 / ESP32-CAM

Thư viện	Chức năng
----------	-----------

<b>WiFi.h</b>	Kết nối ESP32 vào WiFi để giao tiếp Internet.
<b>PubSubClient.h</b>	Thư viện MQTT để ESP32 kết nối tới Mosquitto broker, publish và subscribe các topic.
<b>SPI.h</b>	Giao tiếp SPI dùng cho module RC522.
<b>MFRC522.h</b>	Điều khiển module RFID RC522: đọc UID thẻ, xác thực thẻ, kiểm tra trạng thái thẻ.
<b>esp_camera.h</b>	Thư viện điều khiển camera OV2640 trên ESP32-CAM (chụp ảnh, xử lý buffer, mã hóa JPEG).
<b>ArduinoJson.h</b>	Tạo và phân tích JSON để gửi dữ liệu lên server qua MQTT.
<b>driver/gpio.h</b> (tích hợp trong ESP-IDF / Arduino core)	Điều khiển GPIO, dùng để <b>kích relay</b> mở khóa.
<b>HTTPClient.h</b> (nếu dùng API gửi ảnh lên server Python)	Gửi HTTP POST chứa ảnh JPEG từ ESP32-CAM lên server Python.

Bảng 6 Bảng các thư viện sử dụng trên ESP32/ESP32-CAM

## b. Các hàm chức năng

### Hàm Khởi Tạo (Setup)

Tên Hàm	Mô Tả Chức Năng
void setup()	Khởi tạo tất cả các thành phần phần cứng và kết nối mạng một lần khi ESP32 khởi động.
void initHardware()	Thiết lập các chân GPIO cho Relay, khởi tạo module RC522.

<b>void connectWiFi()</b>	Thiết lập kết nối WiFi (sử dụng SSID và Password đã lưu).
<b>void connectMQTT()</b>	Thiết lập kết nối với broker MQTT (Mosquito) và đăng ký các <b>Topics</b> cần lắng nghe (ví dụ: lệnh cấu hình, kết quả xác thực từ server).

Bảng 7 Bảng mô tả chức năng Hàm khởi tạo

## Hàm Vòng Lặp Chính (Loop)

Tên Hàm	Mô Tả Chức Năng
<b>void loop()</b>	Vòng lặp chính chạy liên tục. Nhiệm vụ chính là duy trì kết nối, kiểm tra dữ liệu đầu vào từ cảm biến và xử lý tác vụ.
<b>void checkSensors()</b>	Kiểm tra xem có thẻ RFID hoặc khuôn mặt mới đang được trình diện không.
<b>void maintainConnection()</b>	Kiểm tra và tự động kết nối lại WiFi/MQTT nếu bị mất.

Bảng 8 Bảng mô tả chức năng Hàm vòng lặp chính

## Hàm Xử Lý Phần Cứng

Tên Hàm	Mô Tả Chức Năng
<b>String readRFID()</b>	Đọc thẻ từ RC522 và trả về <b>ID thẻ</b> (String). Nếu không có thẻ, trả về chuỗi rỗng.
<b>void startVideoStream()</b>	* <b>MQTT/WebSocket:</b> Gửi từng frame ảnh (đã nén \$JPEG\$) qua MQTT hoặc WebSocket (Push) đến Server Python. <b>(Phương pháp phù hợp hơn cho giao tiếp Real-time).</b>

<b>void controlLock(bool state)</b>	Điều khiển Relay. state = true để <b>mở khóa</b> (kích hoạt Relay), state = false để <b>đóng khóa</b> (tắt Relay).
<b>void triggerOpen(int duration)</b>	Thực hiện hành động mở khóa bằng cách gọi controlLock(true), đợi trong duration (ví dụ 5 giây), sau đó gọi controlLock(false) để khóa lại.

Bảng 9 Bảng mô tả chức năng Hàm xử lý phần cứng

## Hàm Giao Tiếp Mạng (MQTT/Server)

Tên Hàm	Mô Tả Chức Năng
<b>void publishData(String topic, String payload)</b>	Gửi dữ liệu (ví dụ: ID RFID/Face ID) lên Server qua <b>MQTT</b> để yêu cầu xác thực.
<b>void mqttCallback(char* topic, byte* payload, unsigned int length)</b>	Hàm xử lý được gọi khi ESP32 nhận được tin nhắn từ Server qua các <b>Topics</b> đã đăng ký. Đây là nơi nhận kết quả xác thực và lệnh cấu hình.
<b>void processAuthResult(String message)</b>	Phân tích tin nhắn từ Server. Nếu kết quả là <b>"ACCESS_GRANTED"</b> , gọi hàm triggerOpen(). Nếu là <b>**"ACCESS_DENIED"</b> , thông báo lỗi.

Bảng 10 Bảng mô tả chức năng Hàm giao tiếp mạng (MQTT/Server)

## Nhóm hàm hệ thống – điều phối thiết bị

Tên Hàm	Mô Tả Chức Năng
---------	-----------------

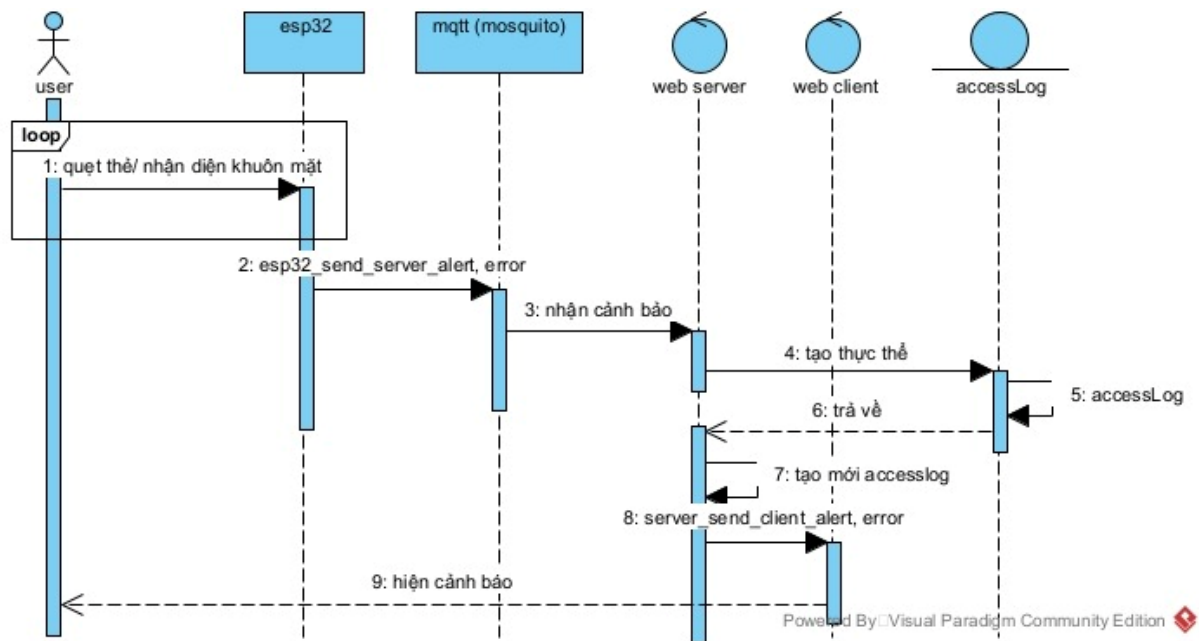
<b>void processAuthentication()</b>	<b>Hàm logic chính:</b> <ol style="list-style-type: none"> <li>1. Đọc chế độ hiện tại của thiết bị (lấy từ MQTT/Server lúc khởi tạo hoặc khi có lệnh cấu hình mới).</li> <li>2. Nếu là <b>OR</b>: Gửi yêu cầu xác thực ngay khi có bất kỳ ID nào.</li> <li>3. Nếu là <b>AND</b>: Lưu trữ tạm thời ID đầu tiên, và chỉ gửi yêu cầu khi có ID thứ hai.</li> </ol>
-----------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Bảng 11 Bảng mô tả chức năng Hàm hệ thống - điều phối thiết bị

### c. Luồng hoạt động

#### 1. Cảnh báo an ninh

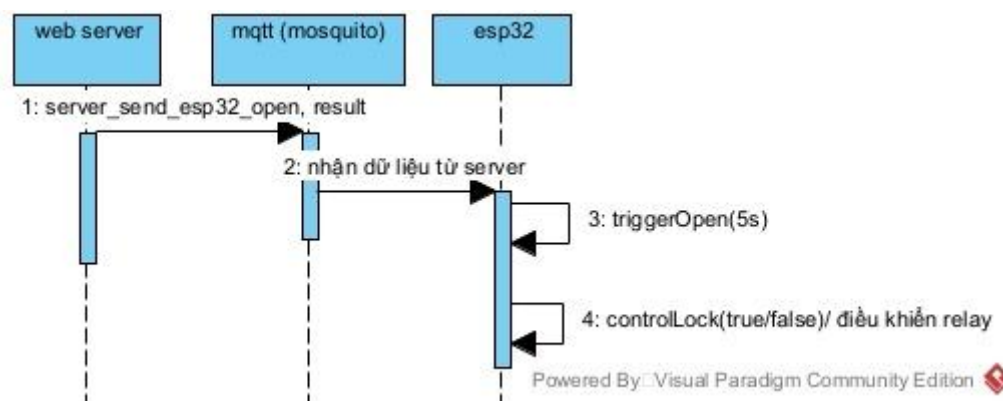
1. **user:** thực hiện hành động xác thực (quét thẻ RFID hoặc đưa mặt vào camera) nếu mà lớn hơn 5 lần.
2. **Gửi tín hiệu báo động:** ESP32 ngay lập tức gửi một tin nhắn báo động lỗi (esp32\_send\_server\_alert\_error) lên mạng qua **MQTT (Mosquito)**.
3. **Máy chủ nhận: Web Server (\$Backend\$)** đang theo dõi các tin nhắn MQTT và nhận được cảnh báo này.
4. Tạo lớp thực thể **accessLog**
5. **Ghi Log:** Web Server lập tức tạo một bản ghi mới trong cơ sở dữ liệu **accessLog** (Nhật ký Truy cập) để lưu lại sự kiện lỗi này.
6. Trả về dữ liệu accesslog
7. Tạo mới accessLog và lưu vào db
8. Web server gửi thông báo lỗi tới web client bằng socketIo (server\_send\_client\_alert\_error)
9. Hiện cảnh báo tới người dùng



Hình 6 Biểu đồ luồng cảnh báo an ninh

## 2. Mở cửa

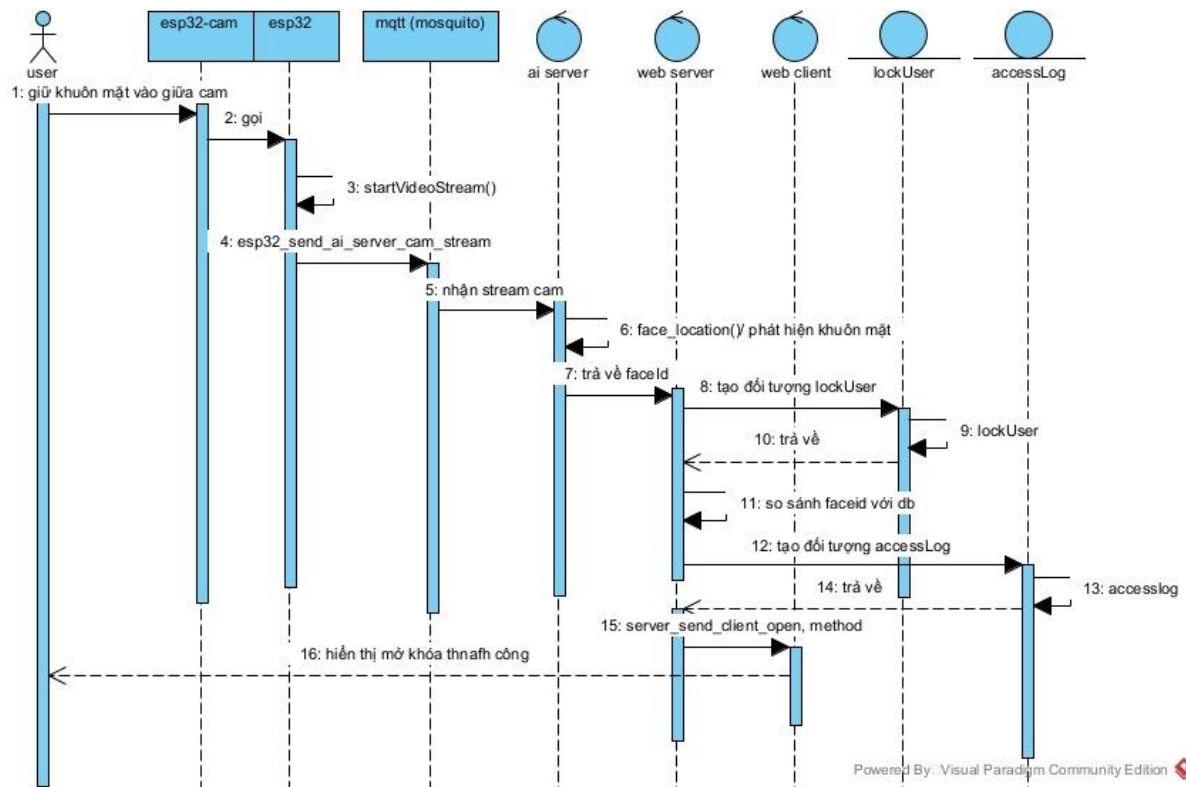
1. **Server ra lệnh: Web Server** (\$Node.js\$ Backend), sau khi xác thực thành công, gửi lệnh mở khóa đến **MQTT Broker**.
2. **ESP32 nhận lệnh:** ESP32 đang lắng nghe MQTT, nhận được lệnh mở khóa từ Server.
3. **ESP32 kích hoạt mở khóa:** ESP32 gọi hàm **triggerOpen()** và đặt thời gian mở là 5 giây.
4. Hàm **triggerOpen()** gọi hàm **controlLock(true)** để kích hoạt **Relay** (mở khóa), sau đó tự động gọi **controlLock(false)** sau 5 giây (đóng khóa).



Hình 7 Biểu đồ luồng mở cửa

## 3. Xác nhận khuôn mặt

1. Người dùng (\$User\$) đưa khuôn mặt vào phạm vi camera.
2. ESP32-CAM báo cho ESP32 (Vi điều khiển chính) biết có người.
3. **ESP32** ra lệnh cho ESP32-CAM bắt đầu chế độ truyền video/ảnh \$stream\$.
4. ESP32 bắt đầu gửi các frame ảnh stream lên Server thông qua **MQTT Broker (Mosquito)**.
5. **AI Server** (Server Python) lắng nghe Topic MQTT và nhận các frame ảnh stream.
6. **AI Server** xử lý frame ảnh nhận được và **Phát hiện vị trí** của khuôn mặt trong ảnh.
7. AI Server **Trích xuất đặc trưng** của khuôn mặt (Encoding) để tạo ra một **Face ID** duy nhất.
8. AI Server (hoặc gửi kết quả đến Web Server) tạo một đối tượng xác thực, tìm kiếm thông tin Lock User tương ứng với Face ID vừa trích xuất.
9. AI Server (hoặc gửi kết quả đến Web Server) tạo một đối tượng xác thực, tìm kiếm thông tin Lock User tương ứng với Face ID vừa trích xuất.
10. Database trả về thông tin LockUser
11. **Web Server** thực hiện kiểm tra quyền cuối cùng (so sánh Face ID và kiểm tra AccessRule dựa trên chế độ AND/OR của thiết bị).
12. Sau khi xác thực thành công, Web Server tạo một bản ghi mới trong **Nhật ký Truy cập (accessLog)**.
13. Database ghi nhận bản ghi accessLog.
14. Database trả về xác nhận đã ghi log.
15. **Web Server** gửi thông báo **mở khóa thành công** đến giao diện quản trị (Web Client) và gửi lệnh mở cửa đến ESP32 (tương tự như luồng mở/đóng cửa).
16. Khóa cửa (ESP32) nhận lệnh mở cửa và thông báo thành công cho người dùng (User)



Hình 8 Biểu đồ luồng xác thực khuôn mặt

#### 4. Xác nhận RFID

**1: Quét thẻ:** Người dùng (\$User\$) đưa thẻ từ vào module **RC522**.

**2: Gọi:** RC522 thông báo cho **ESP32** (Vi điều khiển chính) biết đã có thẻ mới.

**3: readRfid():** ESP32 gọi hàm **readRfid()** để đọc và trích xuất **ID của thẻ từ**.

**4: esp32\_send\_server\_rfid:** ESP32 gửi ID thẻ đã đọc được lên Server thông qua **MQTT Broker (Mosquito)**.

**5: Nhận thông tin rfid:** Web Server (Backend) lắng nghe Topic MQTT và nhận được ID thẻ.

**6: Tạo đối tượng rfid:** Web Server bắt đầu tạo một đối tượng yêu cầu xác thực RFID.

**7: RFID & 8: trả về:** Web Server truy vấn Database (rfid) để tìm thông tin người dùng (LockUser) liên kết với ID thẻ này.

**9: Kiểm tra trong csdl:** Web Server thực hiện logic kiểm tra quyền cuối cùng:

- Kiểm tra xem LockUser này có quyền truy cập vào thiết bị đang dùng hay không (AccessRule).
- Kiểm tra xem chế độ thiết bị (Device.mode - AND/OR) có thỏa mãn không (Ví dụ: Nếu chế độ **AND** đang bật, Server sẽ đợi thêm Face ID trước khi đưa ra quyết định cuối cùng).

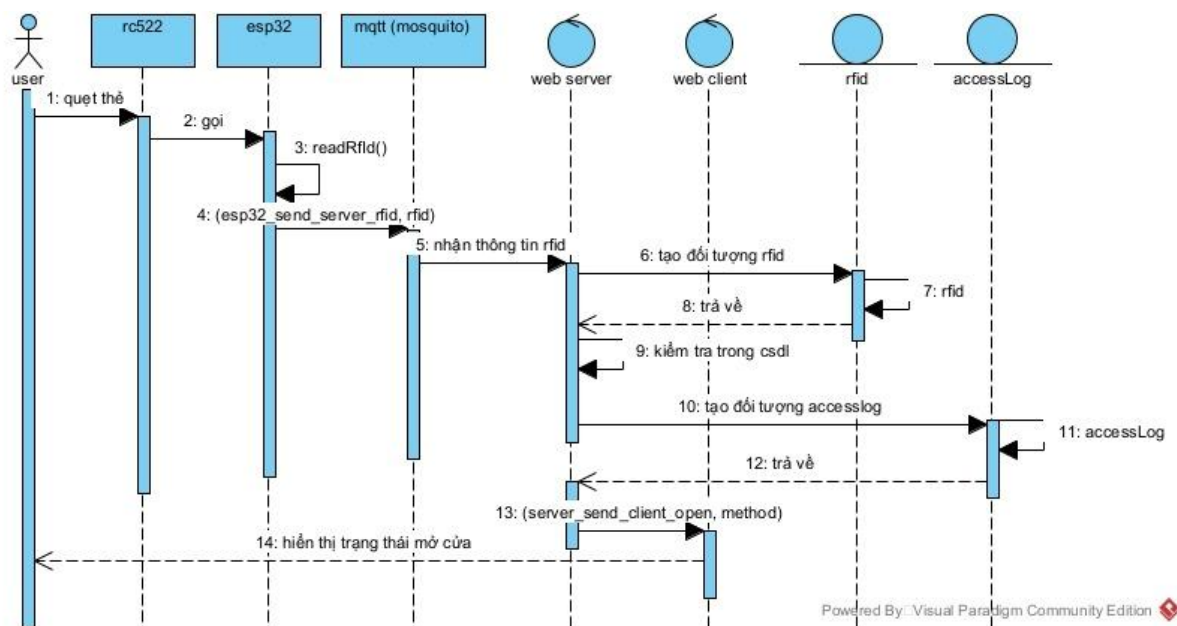
**10: Tạo đối tượng accessLog:** Sau khi xác thực thành công, Web Server tạo một bản ghi mới trong **Nhật ký Truy cập (accessLog)**.

**11: accessLog & 12: trả về:** Database ghi log và trả về xác nhận.

**13: server\_send\_client\_open, method:** Web Server:

- Gửi lệnh **mở cửa** qua **MQTT** đến ESP32.
- Gửi thông báo **mở khóa thành công** đến giao diện quản trị (Web Client) qua WebSocket.

**14: iễn thị trạng thái mở cửa:** ESP32 nhận lệnh mở cửa và điều khiển **Relay** để mở khóa, đồng thời thông báo trạng thái thành công cho người dùng (User).



Hình 9 Biểu đồ luồng xác thực RFID

#### 4.4.2 Website quản lý

##### a. Server

##### 1. Backend

Thư viện sử dụng ở Server

Thư viện	Chức năng
<b>mqtt</b>	Kết nối Node.js với MQTT broker (Mosquitto). Nhận dữ liệu từ ESP32 và gửi lệnh điều khiển.
<b>express</b>	Tạo REST API để quản lý người dùng, thiết bị, lịch sử mở khóa, cấu hình AND/OR.
<b>mongoose</b>	Kết nối và thao tác với cơ sở dữ liệu MongoDB.
<b>cors</b>	Cho phép frontend React truy cập API Node.js.
<b>dotenv</b>	Quản lý biến môi trường như MONGO_URL, MQTT_BROKER_URL.
<b>bcryptjs</b>	Mã hóa mật khẩu khi đăng ký user.
<b>jsonwebtoken (JWT)</b>	Tạo token đăng nhập và xác thực API.
<b>axios</b>	Node.js gửi request sang server Python để xử lý ảnh (nếu Python chạy dạng API).

*Bảng 12 Bảng các thư viện dùng ở Server*

Các nhóm bảng api sử dụng

API Endpoint	Phương thức	Chức năng
<b>NHÓM USER</b>		
/api/auth/register	POST	Đăng ký tài khoản mới, mã hóa mật khẩu (bcryptjs)

/api/auth/login	POST	Đăng nhập, trả về <b>JWT token</b>
<b>NHÓM LOCKUSER (Quản lý Khóa)</b>		
/api/lockuser	POST	Tạo khóa mới (Gán cho người dùng)
/api/lockuser	GET	Lấy danh sách khóa
/api/lockuser/:id	GET	Lấy thông tin chi tiết 1 khóa
/api/lockuser/:id	PUT	Cập nhật mô tả, vị trí khóa
/api/lockuser/:id	DELETE	Xóa khóa
<b>NHÓM RFID</b>		
/api/rfid	POST	Thêm thẻ <b>RFID</b> vào khóa
/api/rfid/:lock_id	GET	Lấy danh sách <b>RFID</b> của khóa
/api/rfid/:id	DELETE	Xóa <b>RFID</b>
<b>NHÓM FACE (Python)</b>		
/api/face/register	POST	Gửi ảnh sang Python để đăng ký <b>FaceID</b>
/api/face/verify	POST	Gửi ảnh sang Python để xác thực <b>FaceID</b>

/api/face/list/:lock_id	GET	Lấy danh sách <b>FaceID</b> của khóa
/api/face/delete/:id	DELETE	Xóa <b>FaceID</b>
<b>NHÓM DEVICE (ESP32 – ESP32CAM)</b>		
/api/device	POST	Thêm thiết bị
/api/device	GET	Lấy danh sách thiết bị
/api/device/:id	PUT	Cập nhật mode ( <b>AND/OR</b> ), trạng thái
/api/device/:id	DELETE	Xóa thiết bị
<b>NHÓM ACCESS RULE &amp; LỊCH SỬ</b>		
/api/access	GET	Lấy lịch sử mở khóa
/api/access	POST	Ghi nhận 1 lần mở khóa ( <b>RFID/Face/AND/OR</b> )
/api/access/rule	POST	Tạo rule mở khóa
/api/access/rule/:id	PUT	Sửa rule
/api/access/rule/:id	DELETE	Xóa rule
<b>NHÓM MQTT (Điều khiển)</b>		

/api/mqtt/unlock/:deviceId	POST	Gửi lệnh <b>mở khóa</b> đến <b>ESP32</b>
/api/mqtt/config/:deviceId	POST	Gửi <b>cấu hình AND/OR</b> đến <b>ESP32</b>
/api/mqtt/status	GET	Lấy <b>trạng thái</b> từ <b>ESP32</b> (qua subscribe topic)

Bảng 13 Bảng các nhóm API sử dụng

## 2. AI

Server Python

Thư viện	Chức năng
<b>opencv-python (cv2)</b>	Xử lý hình ảnh: đọc ảnh, resize, grayscale, phát hiện khuôn mặt, chuyển đổi định dạng.
<b>face_recognition</b>	Nhận diện khuôn mặt (so sánh mã hoá khuôn mặt – face encoding). Dựa trên dlib.
<b>numpy</b>	Xử lý ma trận ảnh, tính toán dữ liệu biểu diễn khuôn mặt.
<b>dlib</b>	Tạo face embeddings, detection, landmark (bắt buộc khi dùng face_recognition).
<b>FastAPI</b> hoặc <b>Flask</b>	Tạo HTTP API để nhận ảnh từ ESP32-CAM hoặc Node.js.
<b>uvicorn</b>	Web server chạy FastAPI/Flask.
<b>Pillow (PIL)</b>	Chuyển đổi ảnh JPEG từ ESP32 sang định dạng xử lý (RGB).

<b>pydantic</b>	Validate dữ liệu gửi lên server.
<b>requests</b>	Python gửi kết quả (match / không match) ngược về Node.js.
<b>paho-mqtt</b> (tùy chọn)	Nếu Python muốn gửi kết quả trực tiếp qua MQTT thay vì qua Node.js.

Bảng 14 Bảng các thư viện sử dụng ở AI

Các hàm sử dụng nhận diện khuôn mặt

Tên Hàm (Method)	Chức Năng Chính	Mô Tả
<b>load_image_file()</b>	<b>Tải Ảnh Đầu vào</b>	Tải tệp ảnh từ ổ đĩa và chuyển đổi thành mảng số học (NumPy array) để các thư viện có thể xử lý.
<b>cv2.cvtColor()</b>	<b>Tiền xử lý Màu sắc</b>	Chuyển đổi không gian màu của ảnh (ví dụ: từ <b>BGR</b> sang <b>RGB</b> ), đảm bảo ảnh có định dạng chính xác cho các thuật toán nhận diện.
<b>face_locations()</b>	<b>Phát hiện Khuôn mặt (Detection)</b>	Quét ảnh để tìm ra vị trí của tất cả khuôn mặt và trả về tọa độ chính xác của chúng.
<b>face_encodings()</b>	<b>Mã hóa Khuôn mặt (Encoding/Embedding)</b>	Trích xuất các đặc trưng độc nhất của khuôn mặt thành một <b>vector 128 chiều</b> (Face Embedding) để phục vụ cho việc so sánh.

<b>np.array()</b>	<b>Quản lý Dữ liệu Embedding</b>	Cấu trúc hóa và thao tác hiệu quả với các vector mã hóa khuôn mặt (Face Embeddings) và dữ liệu hình ảnh.
<b>compare_faces()</b>	<b>So sánh/Đối chiếu (Matching)</b>	So sánh vector mã hóa khuôn mặt mới với các vector đã lưu trữ (khuôn mặt đã biết), trả về kết quả <b>Đúng/Sai (True/False)</b> về sự trùng khớp.
<b>face_distance()</b>	<b>Đo độ Tương đồng</b>	Tính toán khoảng cách giữa các vector mã hóa. Khoảng cách càng nhỏ, khuôn mặt càng giống nhau và được dùng để xác định ngưỡng trùng khớp.

Bảng 15 Bảng mô tả các hàm sử dụng nhận diện khuôn mặt

## b. Client

Thư viện sử dụng ở Client

<b>Thư viện</b>	<b>Chức năng</b>
<b>react</b>	Framework xây dựng giao diện web.
<b>mqtt (mqtt.js)</b>	React kết nối trực tiếp đến Mosquitto qua MQTT over WebSocket (ws://).
<b>axios</b>	Gửi request đến backend Node.js (REST API).
<b>react-router-dom</b>	Quản lý các trang trong ứng dụng (login, dashboard, logs).
<b>redux / recoil (tùy chọn)</b>	Quản lý trạng thái người dùng, token, cấu hình hệ thống.

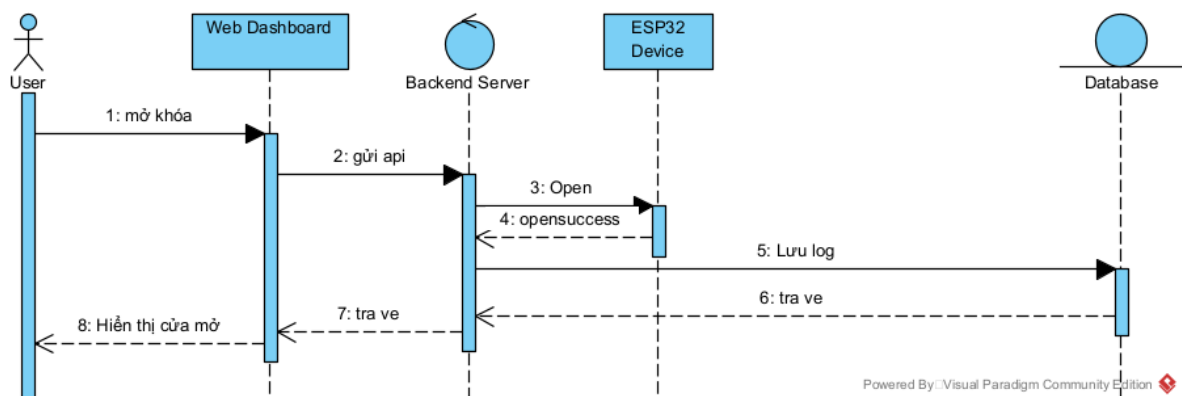
<b>tailwindcss / bootstrap</b>	Tạo giao diện nhanh và đẹp.
<b>chart.js / recharts</b>	Hiển thị biểu đồ: số lần mở khóa, log truy cập, phân tích dữ liệu.

Bảng 16 Bảng các thư viện sử dụng ở Client

### c. Luồng hoạt động

#### 1. Mở khóa trên web

1. Admin nhấn "Mở khóa".
2. Web gửi API tới Server.
3. Server kiểm tra quyền -> Gửi lệnh OPEN qua WebSocket tới ESP32.
4. ESP32 mở Relay -> Gửi lại xác nhận OPEN\_SUCCESS.
5. Server ghi Log vào DB -> Gửi thông báo lại cho Web Dashboard.

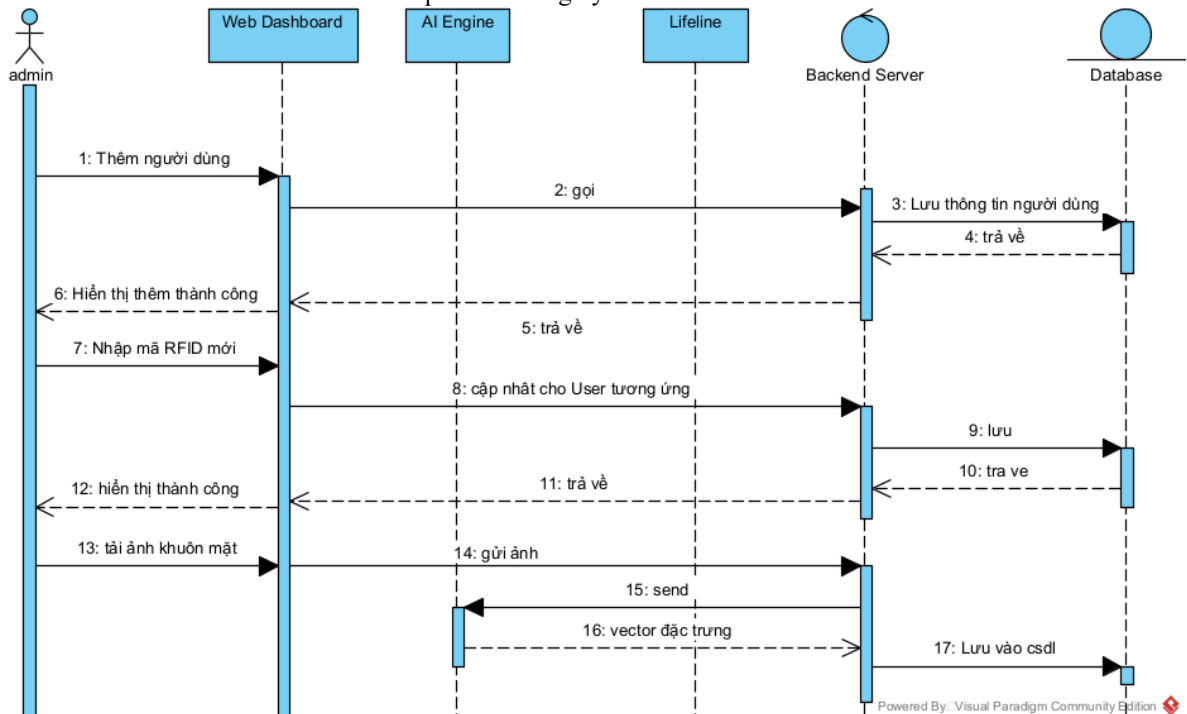


Hình 10 Biểu đồ luồng mở khóa trên web

#### 2. Quản lý người dùng

1. Admin thực hiện thao tác nhập thông tin và chọn chức năng "Thêm người dùng" trên giao diện Web Dashboard.
2. Web Dashboard gọi API gửi dữ liệu người dùng vừa nhập đến Backend Server.
3. Backend Server xử lý và gửi lệnh "Lưu thông tin người dùng" vào Database.
4. Database thực hiện lưu trữ và "trả về" kết quả (kèm ID người dùng mới) cho Backend Server.
5. Backend Server gửi phản hồi "trả về" trạng thái thành công cho Web Dashboard.

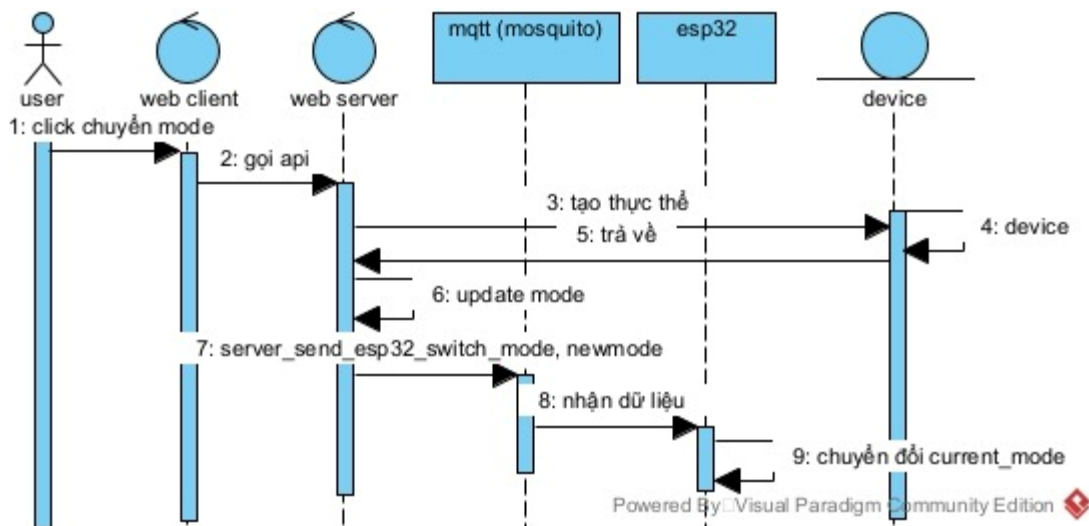
6. Web Dashboard hiển thị thông báo "Hiện thị thêm thành công" để Admin biết.
7. Admin tiếp tục thực hiện bước "Nhập mã RFID mới" cho người dùng vừa tạo trên giao diện.
8. Web Dashboard gửi yêu cầu "cập nhật cho User tương ứng" đến Backend Server.
9. Backend Server gửi dữ liệu RFID xuống Database để "lưu".
10. Database xác nhận lưu xong và "tra ve" kết quả cho Backend Server.
11. Backend Server tiếp tục "tra về" phản hồi thành công cho Web Dashboard.
12. Web Dashboard hiển thị thông báo "hiển thị thành công" việc gán thẻ cho Admin.
13. Admin thực hiện bước cuối cùng là "tải ảnh khuôn mặt" của người dùng lên để hệ thống học.
14. Web Dashboard thực hiện "gửi ảnh" này về phía Backend Server.
15. Backend Server chuyển tiếp ("send") hình ảnh sang module AI Engine để xử lý nhận diện.
16. AI Engine phân tích ảnh và trả về "vector đặc trưng" (mã hóa khuôn mặt) cho Backend Server.
17. Backend Server nhận vector này và gửi lệnh "Lưu vào csdl" (Database) để hoàn tất quá trình đăng ký.



Hình 11 Biểu đồ luồng quản lý người dùng trên web

### 3. Chuyển chế độ

1. **Click chuyển mode:** Người dùng (User) tương tác với giao diện quản lý trên **Web Client** (React.js) và nhấn nút để chuyển đổi chế độ xác thực (ví dụ: từ **OR** sang **AND**).
2. **Gọi api:** **Web Client** gửi yêu cầu thay đổi chế độ đến **Web Server** (Node.js Backend) thông qua API (hoặc WebSocket).
3. **Tạo thực thể**
4. **Device:** **Web Server** bắt đầu truy vấn để tìm và cập nhật thông tin thiết bị trong Database (device - thực chất là bảng Device trong MongoDB).
5. **Trả về:** Database trả về xác nhận rằng việc tìm kiếm/truy vấn đã hoàn tất.
6. **Update mode:** **Web Server** thực hiện lệnh cập nhật, thay đổi trường mode của thiết bị trong \$MongoDB\$ thành chế độ mới (newmode).
7. **server\_send\_esp32\_switch\_mode, newmode:** Sau khi cập nhật MongoDB thành công, **Web Server** gửi lệnh cấu hình đến **MQTT Broker (Mosquito)**. Lệnh này bao gồm chế độ mới (newmode).
8. **Nhận dữ liệu:** **ESP32** đang lắng nghe Topic MQTT và nhận được tin nhắn lệnh chuyển đổi chế độ từ Server.
9. **Chuyển đổi current\_mode:** **ESP32** phân tích tin nhắn và cập nhật biến trạng thái nội bộ của mình (current\_mode) thành chế độ mới (newmode).

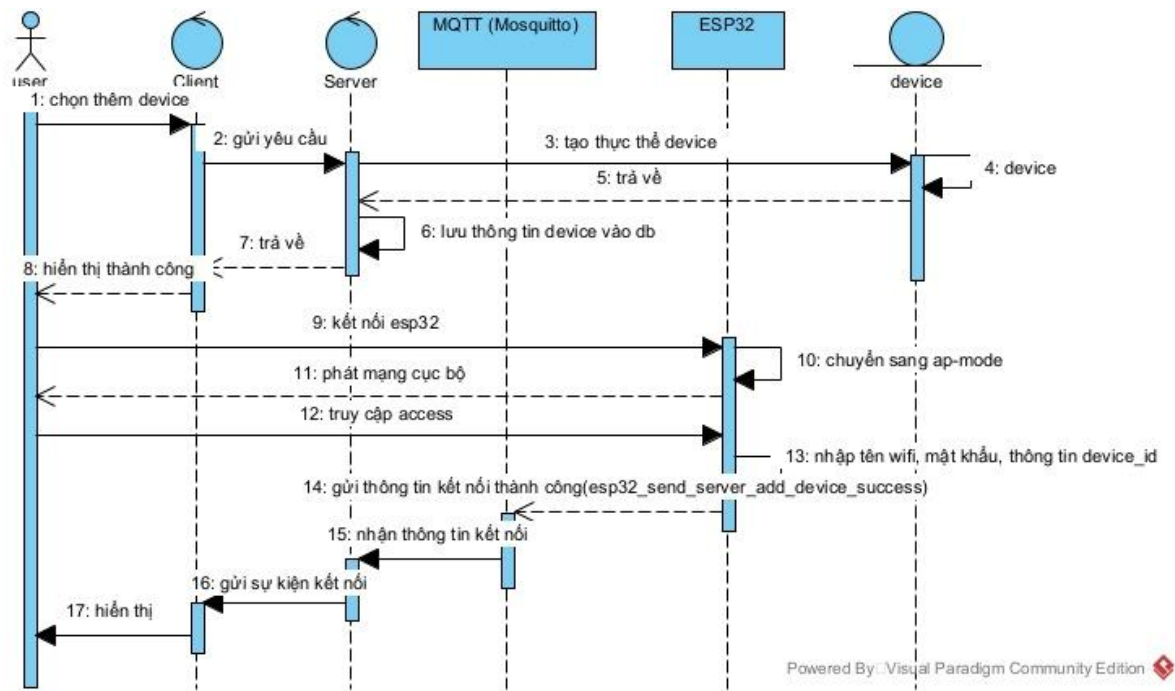


Hình 12 Biểu đồ luồng chức năng chuyển chế độ

#### 4. Thêm thiết bị

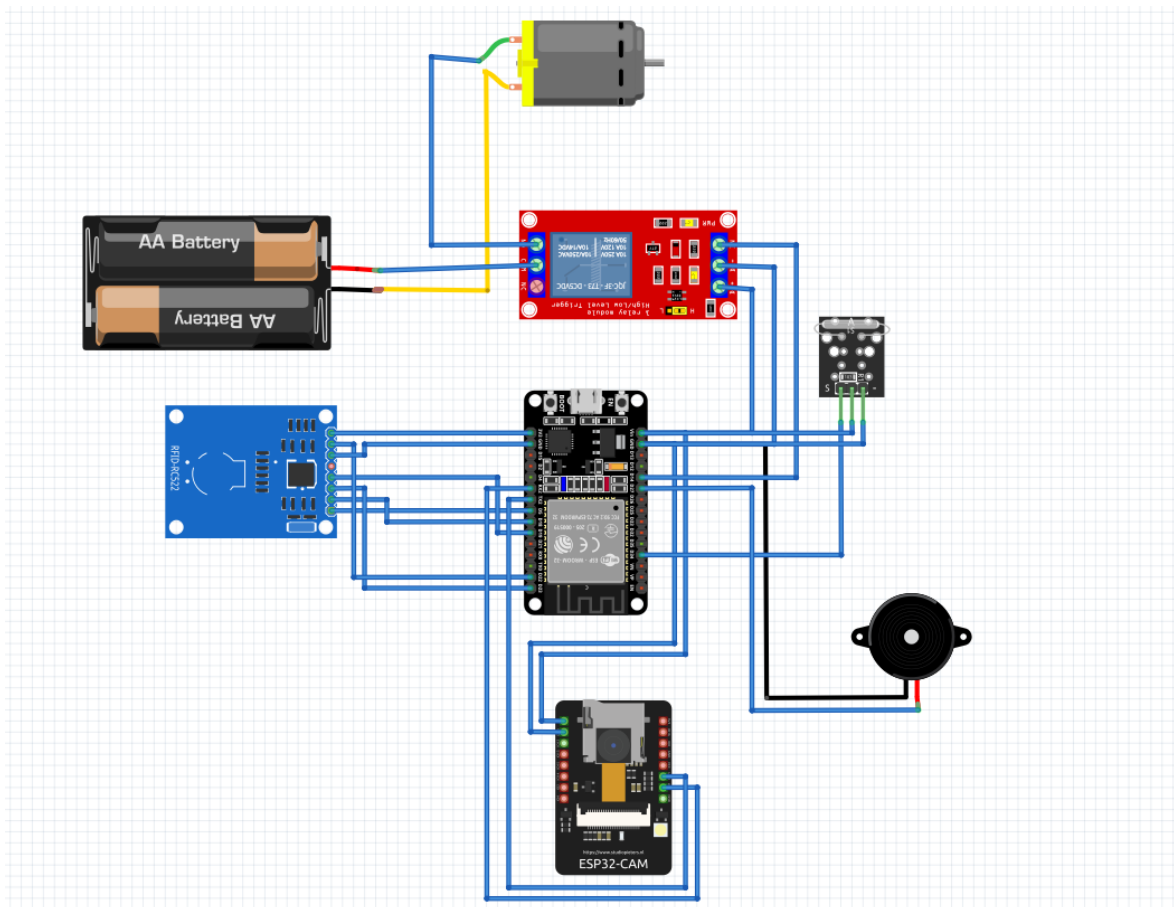
1. **Chọn thêm device:** Người dùng (User) sử dụng giao diện web **Client** để chọn chức năng thêm thiết bị mới.

2. **Gửi yêu cầu:** **Client** gửi yêu cầu thêm thiết bị đến **Server** (Backend).
3. **Tạo thực thể device**
4. **Device: Server** tạo một bản ghi mới trong database **Device** (MongoDB) để lưu trữ thông tin nhận dạng ban đầu của thiết bị (ví dụ: \$deviceId\$ duy nhất).
5. **Trả về**
6. **Lưu thông tin device vào db:** Database xác nhận và Server hoàn tất lưu thông tin thiết bị mới vào cơ sở dữ liệu.
7. **Trả về**
8. **Hiển thị thành công:** Server thông báo cho **Client** rằng việc đăng ký đã thành công, và giao diện hiển thị thông báo.
9. **Kết nối esp32:** Server/Client hướng dẫn người dùng thực hiện các bước để chuẩn bị cấu hình cho **ESP32** (ví dụ: nhấn nút trên ESP32 để kích hoạt chế độ cấu hình).
10. **Chuyển sang ap-mode:** **ESP32** chuyển sang chế độ **Access Point (AP Mode)**, tạo ra một mạng Wi-Fi cục bộ riêng.
11. **Phát mạng cục bộ:** Mạng Wi-Fi cục bộ của ESP32 hiện lên để người dùng kết nối.
12. **Truy cập access:** Người dùng sử dụng điện thoại/máy tính kết nối vào mạng Wi-Fi cục bộ của ESP32.
13. **Nhập tên wifi, mật khẩu, thông tin device\_id:** Người dùng truy cập trang cấu hình do ESP32 cung cấp, nhập:  
  
**Tên và mật khẩu Wi-Fi** (Credentials) của mạng nhà.  
  
**Device ID** (đã được tạo ở bước 3, dùng để nhận dạng).
14. **Gửi thông tin kết nối thành công**  
**(esp32\_send\_server\_add\_device\_success):** Sau khi nhận thông tin, **ESP32** lưu Credential, thoát khỏi AP Mode, kết nối vào mạng Wifi gia đình, và gửi thông báo kết nối thành công (bao gồm Device ID) qua **MQTT** đến **Server**.
15. **Nhận thông tin kết nối:** Server nhận thông báo kết nối thành công từ \$ESP32\$ qua MQTT.  
  
**Server** cập nhật trạng thái kết nối (status) của thiết bị trong database Device.
16. **Gửi sự kiện kết nối & 17: hiển thị:** Server thông báo real-time đến **Client** (qua WebSocket) rằng thiết bị ESP32 đã kết nối và sẵn sàng hoạt động. **Client** hiển thị trạng thái đã kết nối cho người dùng.



Hình 13 Biểu đồ luồng chức năng thêm thiết bị

## 4.5 Tích hợp thiết bị



Hình 14 Sơ đồ mạch lắp đặt hệ thống

- RC522 kết nối ESP32 qua SPI

- ESP32-CAM kết nối với ESP32 qua UART hoặc WiFi nội bộ
- Relay điều khiển chân GPIO
- Thiết bị kết nối router qua WiFi 2.4GHz

## 4.6 Phát triển ứng dụng

### 4.6.1 Phát triển Backend

#### Tổng quan

- Mục tiêu: Backend cung cấp API REST và WebSocket để quản lý người dùng, thiết bị, tag RFID và dữ liệu nhận diện khuôn mặt; lưu log truy cập và xử lý cảnh báo; cung cấp middleware phân quyền cho từng API.
- Tính năng chính:
  - Đăng ký/đăng nhập (authentication) và phân quyền.
  - Quản lý người dùng, thiết bị, thẻ (RFID) và khuôn mặt (face\_id).
  - WebSocket cho real-time communication với thiết bị (push event, nhận trạng thái).
  - Lưu trữ access logs và cơ chế cảnh báo khi có bất thường.
  - Middleware kiểm soát truy cập theo vai trò.

#### Kiến trúc & stack

- Kiến trúc: Server Node.js + Express (REST API) kết hợp Socket.IO cho real-time. Kiến trúc phân lớp: routes -> controllers -> models -> DB; middleware ở giữa để auth/validation; socket layer để tương tác realtime với thiết bị.
- Stack công nghệ:
  - Node.js, Express
  - TypeScript (dự án dùng .ts)
  - Socket.IO cho WebSocket
  - Database (config ở config/database.ts) — (giả sử sử dụng MySQL/Postgres/Mongo tùy config; thay tên chính xác nếu readme/tsconfig cho biết)
  - JWT cho authentication
  - Các package quản lý logs (có thể là winston hoặc console logging hiện tại)

- Luồng hoạt động chính:
  - Client/Frontend gọi API REST để quản lý người dùng, gán tag/khuôn mặt.
  - Thiết bị kết nối qua Socket.IO để nhận lệnh và gửi sự kiện (ví dụ: quét thẻ).
  - Khi có sự kiện truy cập, backend xác thực, ghi access\_log và (nếu cần) phát cảnh báo đa kênh.

WebSocket ([Socket.IO](#))

File chính: socket.io/socket.ts và socket.io/device.socket.ts.

- Namespaces / rooms:
  - Có thể có namespace /devices cho thiết bị và /clients cho frontend dashboards.
  - Devices join a room named by device id or group.
- Event examples:
  - Từ thiết bị -> server:
    - device:hello — device đăng ký kết nối (payload: { deviceId, token })
    - device:event — gửi sự kiện truy cập { deviceId, method, tagValue, timestamp }
  - Server -> device:
    - server:open — lệnh mở khóa { reason, duration }
    - server:update-config — cập nhật cấu hình access rule
  - Server -> clients:
    - log:new — phát log mới realtime { accessLog }
    - alert — cảnh báo { level, message, details }
- Auth cho socket: truyền token khi connect (query param hoặc auth handshake); validate bằng cùng secret JWT.
- Luồng:
  - Thiết bị connect -> gửi device:hello -> server validate và join room -> server xác thực khi event vào -> ghi log -> emit log:new tới dashboard.

## Logging & xử lý cảnh báo

- Lưu log:
  - Ghi mọi truy cập vào Access\_Log với đủ metadata (userId/tagValue/deviceId/result,timestamp).
  - Ghi thêm logs hệ thống (errors) => dùng winston/pino hoặc console cụ thể.
- Xử lý cảnh báo:
  - Rule-based detection: ví dụ 5 lần denied liên tiếp trong 1 phút => tạo cảnh báo.
  - Event-based: truy cập từ thiết bị lạ, tag không hợp lệ, face match fail nhiều lần.
  - Notification channels: realtime (Socket.IO alert event), email/SMS (tích hợp svc), console/DB.
- Tip: Đặt alert severity (info/warn/critical) và lưu alert record để truy cứu.

## Gắn thẻ (RFID) & nhận diện khuôn mặt (face)

- RFID flow:
  - Thiết bị đọc tag -> gửi tagValue lên server (socket or REST).
  - Server tìm rf\_id matching -> nếu có userId và rule cho phép thì grant access -> emit command mở khóa -> ghi access\_log.
  - API quản lý tag: thêm tag mới, gán cho user (POST /rf\_ids), hủy gán (PUT /rf\_ids/:id).
- Face enrollment flow:
  - Frontend thu ảnh/feature vector (client-side SDK hoặc server-side model).
  - POST /face\_id với userId và template (base64 hoặc ref).
  - Server lưu template an toàn (nên mã hóa hoặc lưu ở storage với truy cập giới hạn).
  - Khi nhận request face auth: server chạy face-matching (local lib hoặc call external svc) => nếu match user thì grant.
- Bảo mật dữ liệu sinh trắc: lưu template ở nơi an toàn, sao lưu mã hóa, tuân thủ privacy.

#### 4.6.2 Phát triển Frontend

Giao diện người dùng được xây dựng bằng ReactJS, tập trung vào trải nghiệm Real-time và trực quan hóa dữ liệu như hình ảnh Dashboard đã thiết kế.

##### a. Cấu trúc Component

- Sidebar Navigation: Điều hướng giữa Home, History, Users, Devices.
- Live Monitor (Camera Feed):
  - Hiển thị luồng video MJPEG hoặc cập nhật ảnh liên tục qua WebSocket từ ESP32-CAM.
  - Trạng thái kết nối: Hiển thị icon "Online" màu xanh (như trong ảnh) khi WebSocket heartbeat nhận tín hiệu tốt.
- Control Panel (Door Controls):
  - Nút "Open Door" và "Close Door" gọi API về Backend.
  - Hiển thị trạng thái cửa hiện tại (Locked/Unlocked) đồng bộ với cảm biến từ trường (nếu có) hoặc trạng thái Relay.
- Dashboard Widgets:
  - *Last Access*: Hiển thị Avatar + Tên + Thời gian của người vừa truy cập (lấy từ sự kiện WebSocket `new_log`).
  - *Quick Stats*: Biểu đồ tròn/cột thống kê tỷ lệ thành công/thất bại trong ngày.

b. Tích hợp Real-time (WebSocket Client) Sử dụng `useEffect` hook để duy trì kết nối WebSocket:

#### 4.6.3 Phát triển Firmware ESP32

Firmware được phát triển trên nền tảng PlatformIO (VS Code) sử dụng FreeRTOS để đảm bảo đa nhiệm thời gian thực.

1. Phân chia tác vụ (FreeRTOS Tasks) Để đảm bảo RFID đọc nhanh ( $<0.5s$ ) và Camera không bị lag, hệ thống chia thành các Task song song:

- Task 1: Network & WebSocket Handle (Core 0)
  - Duy trì kết nối WiFi và WebSocket.
  - Tự động Reconnect khi mất mạng.
  - Lắng nghe lệnh `{"cmd": "OPEN"}` từ Server để kích hoạt Relay.
- Task 2: Sensor Logic (Core 1 - App Core)

- Vòng lặp RFID: Quét liên tục (mfrc522.PICC\_IsNewCardPresent()). Khi có thẻ -> Gửi UID vào Queue xử lý.
    - Logic Camera: Khi có thẻ hoặc cảm biến chuyển động -> Chụp ảnh (esp\_camera\_fb\_get()) -> Nén JPEG -> Gửi Buffer qua WebSocket.
  - Task 3: Actuator Control (Core 1)
    - Điều khiển Relay/Servo mở chốt.
    - Logic tự động khóa lại (Auto-lock) sau 5 giây.
2. Cơ chế OTA (Over-The-Air Update) Tích hợp thư viện ArduinoOTA để đáp ứng yêu cầu bảo trì từ xa.
- Cấu hình một cổng riêng hoặc endpoint HTTP để upload file firmware .bin.
  - Kiểm tra checksum MD5 trước khi flash để tránh firmware lỗi.
3. Sơ đồ xử lý tại biên (Edge Processing) Để tối ưu tốc độ:
- ESP32 sẽ không chạy mô hình AI nặng.
  - Nó chỉ đóng gói dữ liệu: UID + Image Buffer + DeviceID + Timestamp.
  - Toàn bộ quyết định "Mở/Đóng" phụ thuộc vào phản hồi từ Server (trừ trường hợp mất mạng, có thể dùng cache UID whitelist cục bộ làm dự phòng).

## 5 Kết Luận

### 5.1 Kết quả đạt được

#### 5.1.1 Về mặt lý thuyết

Qua việc tìm hiểu và nghiên cứu các công nghệ cốt lõi như vi điều khiển ESP32, cảm biến RFID (RC522), camera ESP32-CAM, và giao thức truyền thông MQTT, nhóm đã xây dựng được nền tảng lý thuyết vững chắc cho một hệ thống kiểm soát truy cập đa yếu tố. Việc tích hợp Node.js Backend, React.js Frontend và MongoDB Database đã giúp nhóm hiểu rõ cơ chế xây dựng một ứng dụng IoT real-time dựa trên kiến trúc Web-based.

#### 5.1.2 Về mặt thực tiễn

Hệ thống đã triển khai thành công hai phương thức xác thực chính là RFID và Nhận diện Khuôn mặt, hoạt động hiệu quả trong việc cấp quyền truy cập. Đặc biệt, chức năng cho phép người dùng cấu hình chế độ AND/OR đã được thực hiện, mang lại sự linh hoạt cao trong việc thiết lập mức độ bảo mật. Hệ thống đã tự động hóa hoàn toàn quá trình xác thực và cấp quyền, cung cấp một giao diện quản lý real-time thông qua WebSocket (socket.io), giúp người dùng dễ dàng theo dõi nhật ký truy cập và các cảnh báo an ninh.

### 5.2 Hạn chế

Hệ thống vẫn còn một số hạn chế cần được cải thiện:

- Khả năng truyền dữ liệu giữa ESP32 và Server hoàn toàn phụ thuộc vào mạng Wi-Fi cục bộ. Nếu mất kết nối Wi-Fi, hệ thống sẽ không thể nhận được kết quả xác thực từ Server và không thể mở cửa.
- Hiệu suất nhận diện khuôn mặt có thể bị ảnh hưởng bởi tốc độ truyền video stream qua MQTT và khả năng xử lý của AI Server (Python), đặc biệt trong điều kiện ánh sáng yếu hoặc khi có nhiều người cùng lúc.
- Hiện tại, việc quản lý và tạo LockUser mới (đăng ký thẻ RFID hoặc thêm mẫu khuôn mặt) vẫn cần một quy trình thủ công trên hệ thống quản trị, chưa hoàn toàn tự động hóa.
- Hệ thống chưa tối ưu hóa chức năng lưu trữ và đồng bộ hóa dữ liệu xác thực trên ESP32 (local cache) để hỗ trợ tính năng mở khóa ngoại tuyến.

## 5.3 Hướng phát triển tương lai

### 5.3.1 Về hệ thống

- Hỗ trợ Mở khóa Ngoại tuyến: Phát triển tính năng lưu trữ tạm thời (cache) dữ liệu RFID và Face ID đã mã hóa trên ESP32, cho phép khóa cửa hoạt động độc lập và cấp quyền truy cập cơ bản ngay cả khi mất kết nối mạng.
- Mở rộng Giao thức Truyền thông: Nghiên cứu và áp dụng các giải pháp như HTTP/HTTPS cho video stream thay vì MQTT cho các gói dữ liệu lớn, hoặc tích hợp BLE (Bluetooth Low Energy) cho cấu hình ban đầu hoặc làm phương thức mở khóa thứ ba.
- Tích hợp App di động: Phát triển ứng dụng di động để thay thế Web\ Client, cho phép người dùng nhận cảnh báo tức thì qua thông báo đẩy (Push\ Notification) và quản lý quyền truy cập tiện lợi hơn.
- Tối ưu hóa AI\ Server: Áp dụng các thuật toán nén ảnh tiên tiến và mô hình \$AI\$ nhẹ hơn để giảm độ trễ (latency) trong quá trình nhận diện khuôn mặt.

### 5.3.2 Về bảo mật

- Mã hóa Giao tiếp: Đảm bảo tất cả dữ liệu truyền qua MQTT và WebSocket đều được mã hóa bằng TLS/SSL để bảo vệ dữ liệu cảm biến và lệnh điều khiển khỏi bị nghe lén hoặc giả mạo.
- Phân quyền RBAC: Triển khai chặt chẽ chính sách phân quyền theo vai trò (Role-Based\ Access\ Control) trên Backend Node.js và Frontend React.js, giới hạn người dùng chỉ có thể thực hiện những tác vụ phù hợp với vai trò của họ.
- Giới hạn Tần suất Truy cập: Thiết lập giới hạn số lần xác thực thất bại liên tiếp trong một khoảng thời gian (\$rate\ limiting\$) và khóa tạm thời thiết bị để ngăn chặn các cuộc tấn công dò mật khẩu hoặc brute-force.