HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BỘ MÔN IOT VÀ ỨNG DỤNG

BÁO CÁO GIỮA KÌ

MÔN HỌC: IOT VÀ ỨNG DỤNG

Nhóm lớp học: Nhóm 02

Nhóm bài tập lớn: Nhóm 06

Danh sách sinh viên

Họ và tên	Mã sinh viên
Đỗ Tuấn Nghĩa	B22DCCN604
Nguyễn Đức Bảo	B22DCCN064
Lý Chí Công	B22DCCN088
Phạm Văn Đức	B22DCCN244
Giảng viên giảng dạy:	Kim Ngọc Bách

HÀ NỘI – 2025

MỤC LỤC

Hệ thống mở cửa thông minh đa phương thức (Thẻ từ + Khuôn mặt)

I. Xác định mục tiêu và phạm vi

1. Mục tiêu hệ thống
    - Vấn đề thực tế
        + Hiện nay, nhiều nhà ở, chung cư, công ty, trường học, … vẫn sử dụng cách thức ra vào là cửa truyền thống. Nhưng bất lợi của cửa này vẫn mang lại nhiều rủi ro cho người sử dụng. Sử dụng chìa khóa cơ học thì dễ bị sao chép, mất mát hoặc quên, và có thể phá một cách dễ dàng. Do đó, việc sử dụng cửa truyền thống không mang lại an toàn bảo mật cho người sử dụng.
    - Mục tiêu IoT
        + Xác thực bằng khuôn mặt: Quét dữ liệu khuôn mặt
        + Xác thực bằng thẻ từ (RFID): Đọc ID của thẻ từ
        + Cho phép người dùng tùy chỉnh chế độ bảo mật: Sử dụng 1 trong 2 phương thức xác thực hoặc kết hợp cả 2 phương thức
        + Cảnh báo thông minh: Gửi thông báo về website nếu xác thực gặp lỗi hoặc không thành công quá 5 lần
    - Kỳ vọng:
        + Tiết kiệm thời gian mở khóa cửa
        + Khả năng bảo mật tốt hơn
        + Hoạt động ổn định 24/7
        + Cảnh báo tức thì khi truy cập trái phép
        + Dễ dàng thêm/xóa người dùng và thẻ

2. Phạm vi triển khai
    - Số lượng thiết bị
        + 1 bộ điều khiển chính ESP32
        + 1 camera nhận diện khuôn mặt ESP32-CAM OV2640
        + 1 module đọc thẻ từ RC522
        + 1 cơ cấu mở khóa Relay/Servo
    - Môi trường hoạt động
        + Trong nhà (cửa chính phòng, văn phòng, nhà ở)
        + Cần có kết nối mạng ổn định

3. Tiêu chí thành công (KPIs)

- Độ chính xác
  + Tỷ lệ đọc đúng thẻ đã đăng ký: >95%
  + Tỷ lệ nhận diện đúng chủ nhân: >90%
  + Tỷ lệ từ chối người dùng hợp lệ: <3%
- Độ trễ
  + Quét thẻ từ RFID: <0.5s
  + Nhận diện khuôn mặt: <2s
  + Gửi thông báo qua WebSocket: < 1s
  + Tổng thời gian mở khóa: < 5s
- Độ tin cậy
  + Hoạt động liên tục trừ khi bảo trì: >99%
- Chi phí
  + Tổng chi phí đầu tư ban đầu cho 1 cửa thông minh <1,500,000 VNĐ
- Khả năng mở rộng
  + Hệ thống có thể hỗ trợ quản lý cửa thông minh với số lượng lớn (ví dụ cho tòa nhà chung cư)

4. Kết quả mong đợi

- Giảm rủi ro mất chìa khóa
- Tăng mức bảo mật lên 2-3 lần: so với khóa thường nhờ kết hợp RFID + nhận diện khuôn mặt
- Chống xâm nhập thông minh: tự động khóa thẻ sau 5 lần quét sai + gửi cảnh báo tức thì

II. Thu thập yêu cầu từ các bên liên quan

1. Bối cảnh vấn đề

- Người dùng có thể quên và làm mất chìa khóa
- Sao chép chìa khóa cũng có thể tiềm ẩn rủi ro an ninh
- Không có cách nào để theo dõi lịch sử ra vào hay cấp quyền truy cập từ xa

2. Yêu cầu từ các bên liên quan

- Người dùng cuối
  + Hiển thị

        - muốn thấy trạng thái hiện tại của khóa (Đang khóa/ đã mở) và chế độ an ninh đang được áp dụng (Chế độ tiện lợi. chế độ an ninh cao)
        - xem lịch sử ra vào chi tiết
    + Cảnh báo
        - nhận thông báo đặc biệt khi có hành vi truy cập không hợp lệ lặp lại nhiều lần (vd: quẹt thẻ sai 5 lần liên tiếp).
    + Điều khiển
        - có quyền thay đổi chế độ an ninh của cửa khóa
        - có phương án dự phòng mở khóa trong trường hợp hệ thống điện tử gặp sự cố (mở cửa trên website)
        - thêm sửa xóa thẻ từ, khuôn mặt.
- Quản lý (Chủ nhà với vai trò quản trị, Quản lý văn phòng, quản lý chung cư)
    + Hiệu quả vận hành: Muốn biết các chỉ số về an ninh và mức độ ổn định của hệ thống.
        - Thống kê an ninh: Tỷ lệ truy cập thành công/thất bại, số lần cảnh báo truy cập trái phép.
        - % Thời gian hoạt động (Uptime): Tỷ lệ phần trăm thời gian khóa kết nối và hoạt động bình thường.
        - % Thời gian ngoại tuyến (Downtime): Ghi nhận số lần và tổng thời gian khóa bị mất kết nối với máy chủ.
    + Báo cáo & phân tích: Hệ thống cần xuất báo cáo tuần/tháng, giúp đánh giá và kiểm toán an ninh.
    + Khả năng mở rộng: Có thể tích hợp thêm nhiều thiết bị khóa cửa mới trong tương lai (cho cửa sau, cửa phòng họp...) mà không phải thay đổi toàn bộ hạ tầng phần mềm.
    + ROI (Return on Investment): Kỳ vọng hệ thống giúp giảm chi phí và rủi ro liên quan đến việc quản lý khóa cơ.
        - Cụ thể: Giảm 100% chi phí làm lại chìa khóa hoặc thay ổ khóa khi nhân viên nghỉ việc/làm mất thẻ từ. Tăng cường an ninh, giảm thiểu rủi ro mất cắp.
- kỹ thuật/ It (Phòng CNTT & Bảo trì)
    + Tích hợp hệ thống sẵn có: Dữ liệu từ khóa cửa cần có khả năng kết nối với các dịch vụ bên ngoài để tăng tính tiện ích.

        - Ví dụ: Tích hợp với dịch vụ thông báo như Telegram Bot hoặc Email để gửi cảnh báo truy cập trái phép hoặc thông báo mở cửa theo thời gian thực.
    + Giao thức truyền thông: Do yêu cầu phản hồi tức thì và tương tác hai chiều, lựa chọn giao thức phải phù hợp.
        - Sử dụng WebSocket qua TLS (tức WSS - WebSocket Secure) để đảm bảo kết nối real-time, ổn định và mã hóa giữa ESP32 và server.
    + Bảo mật: Dữ liệu truyền phải được mã hóa và người dùng cần được phân quyền rõ ràng trên hệ thống web.
        - Admin (Quản trị): Có toàn quyền thêm/xóa người dùng, thay đổi chế độ an ninh, xem toàn bộ lịch sử và mở khóa từ xa.
        - User (Người dùng thường): Chỉ có quyền mở khóa và xem lịch sử ra vào của chính mình.
    + Khả năng bảo trì: Hệ thống cần hỗ trợ cập nhật firmware cho thiết bị ESP32 từ xa (OTA - Over-the-Air).
        - Điều này cho phép vá lỗi bảo mật hoặc nâng cấp tính năng mới cho khóa mà không cần phải tháo gỡ thiết bị.

3. phương pháp thu nhập yêu cầu

- Phỏng vấn
    + Trao đổi trực tiếp với người dùng tiềm năng để hiểu rõ những bất tiện của khóa truyền thống và kỳ vọng của họ đối với một hệ thống khóa thông minh.(Thành viên gia đình, bạn bè (đóng vai người dùng cuối).)
- Khảo sát
    + Gửi bảng hỏi cho nhóm đối tượng lớn hơn để thu thập dữ liệu định lượng về mức độ quan trọng của từng tính năng.
- Quan sát thực tế
    + Kỹ sư đến trực tiếp vị trí dự định lắp đặt để kiểm tra các yếu tố môi trường có thể ảnh hưởng đến hoạt động của thiết bị.(Vị trí lắp đặt thực tế (cửa ra vào phòng, cửa nhà).)
        - Kiểm tra chất lượng sóng WiFi tại vị trí cửa để đảm bảo kết nối ổn định.
        - Đánh giá điều kiện ánh sáng tại vị trí lắp camera vào các thời điểm khác nhau trong ngày (sáng, trưa, tối) để lường trước ảnh hưởng tới khả năng nhận diện khuôn mặt.

- Xác định vị trí lắp đặt tối ưu cho camera và đầu đọc thẻ từ để đảm bảo người dùng ở các chiều cao khác nhau đều có thể sử dụng thuận tiện.

III. Xác định yêu cầu chức năng

1. Các chức năng cần có của hệ thống

a. Thu thập dữ liệu cảm biến
   - RC522: Đọc mã UID của thẻ RFID khi người dùng quẹt thẻ
   - ESP32-CAM: Chụp ảnh khuôn mặt khi có người dùng đứng trước cửa, sau đó xử lý và xác định người dùng
   - ESP32: Tổng hợp dữ liệu từ các module (RFID + Camera) và gửi kèm theo timestamp + device ID để đảm bảo tính truy vết

b. Truyền dữ liệu
   - Thiết bị gửi: ESP32 (vi điều khiển chính).
   - Thiết bị nhận (Gateway/Cloud): Node.js Backend (đóng vai trò là WebSocket Server).
   - Giao thức: WebSocket. Giao thức này được sử dụng để duy trì kết nối real-time hai chiều giữa ESP32 và server, rất phù hợp để gửi các thông báo sự kiện ngay lập tức (như "Thẻ X vừa được quẹt", "Nhận diện khuôn mặt thành công/thất bại").
   - Yêu cầu ổn định: Do sử dụng WebSocket qua WiFi, cần đảm bảo kết nối mạng ổn định. Nếu mất kết nối, ESP32 phải có cơ chế tự động kết nối lại (reconnect logic) để đảm bảo mọi sự kiện xác thực đều được ghi nhận và lệnh điều khiển từ xa được tiếp nhận kịp thời.

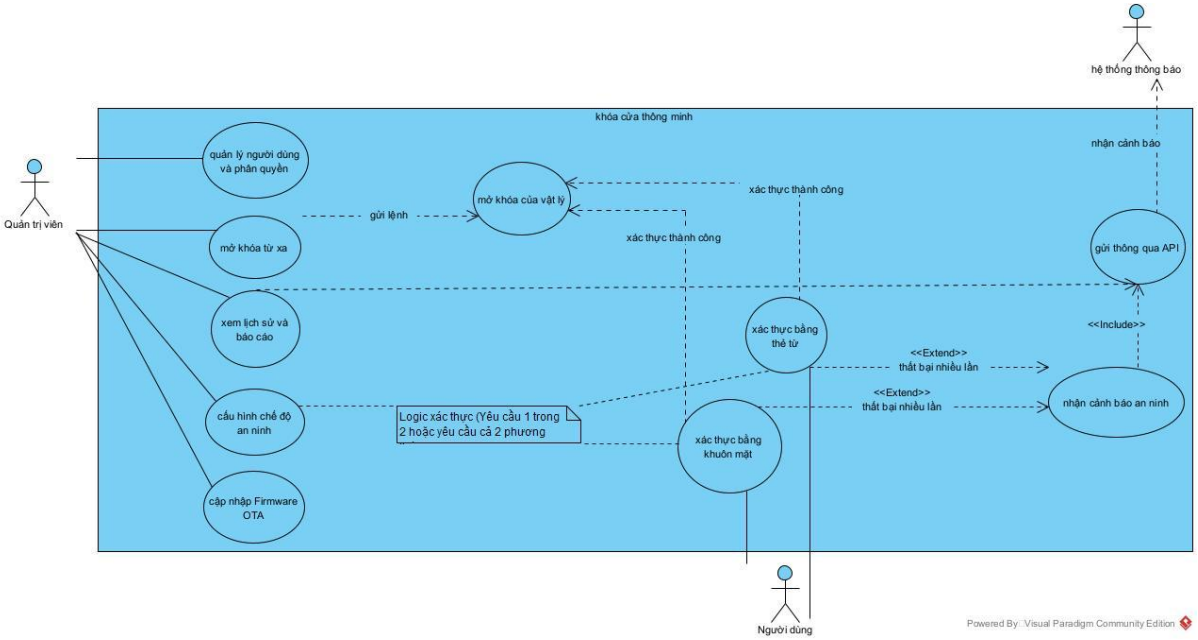
c. Lưu trữ và phân tích dữ liệu
   - Lưu trữ dữ liệu
     + Dữ liệu được lưu trữ trong MongoDB, bao gồm:
       - Bảng User: thông tin người quản lý hệ thống
       - Bảng Lock: lưu thông tin của người dùng phương thức
       - Bảng Log: ghi lại lịch sử mở khóa (thời gian, kết quả, thiết bị, phương phức)

        - Bảng TypeLock: lưu thông tin của các phương thức
    - Phân tích dữ liệu
        + Thống kê số lần mở khóa mỗi ngày, mỗi người hoặc theo thiết bị
        + Phát hiện bất thường (ví dụ: mở khóa thất bại quá 5 lần liên tiếp -> cảnh báo)

d. Hiển thị dữ liệu qua ứng dụng
    - Giao diện người dùng: Frontend (React Js) kết nối với WebSocket Client
    - Chức năng hiển thị:
        + Hiển thị trạng thái real-time của cửa (đang mở/ đóng)
        + Bảng log hoạt động: thời gian - người mở - phương thức
        + Cảnh báo: hiển thị thông báo khi có lỗi nhận diện hoặc truy cập trái phép
        + Biểu đồ thống kê: số lần mở cửa theo ngày/ người dùng
        + Cho phép chọn chế độ mở khóa (AND/OR) trực tiếp trên giao diện
        + Thêm/ sửa/ xóa dữ liệu cho các phương thức thẻ từ và khuôn mặt

e. Điều khiển/ ra lệnh
    - Từ ứng dụng web, người dùng hệ thống có thể:
        + Mở khóa cửa từ xa
        + Đổi chế độ mở khóa (1 trong 2 phương thức hoặc cả 2 phương thức)
        + Cập nhật dữ liệu cho các phương thức

2. Đặc tả luồng công việc (usecase)



- Bắt đầu: Người dùng tiếp cận cửa và thực hiện một trong hai hành động:
    + Quét thẻ từ (Use Case U1: Xác thực bằng Thẻ từ).
    + Hoặc đưa khuôn mặt vào camera (Use Case U2: Xác thực bằng Khuôn mặt).
- Xử lý logic: Hệ thống nhận diện yêu cầu và kiểm tra dựa trên quy tắc đã được Quản trị viên thiết lập trước đó thông qua Use Case U5: Cấu hình chế độ an ninh.
    + Chế độ 1 (Tiện lợi): Hệ thống chỉ cần một trong hai phương thức (thẻ từ hoặc khuôn mặt) hợp lệ.
    + Chế độ 2 (An ninh cao): Hệ thống yêu cầu cả hai phương thức phải được xác thực thành công.
- Kết quả: Nếu xác thực thành công theo đúng chế độ đã cấu hình, hệ thống sẽ thực thi Use Case "Mở khóa cửa vật lý", và Người dùng có thể vào trong.
- Kích hoạt: Luồng này bắt đầu giống như Luồng 1, nhưng Người dùng (hoặc kẻ gian) xác thực thất bại nhiều lần liên tiếp bằng thẻ từ (U1) hoặc khuôn mặt (U2).
- Mở rộng (Extend): Điều kiện "thất bại nhiều lần" sẽ kích hoạt Use Case U8: Nhận cảnh báo an ninh. Đây là một hành động mở rộng, chỉ xảy ra khi có điều kiện bất thường.
- Bao gồm (Include): Ngay khi hệ thống ghi nhận một cảnh báo (U8), nó bắt buộc phải thực thi Use Case U9: Gửi thông báo qua API.

- Kết quả: Thông tin cảnh báo được gửi qua API đến Hệ thống thông báo bên ngoài (ví dụ: bot Telegram hoặc Email), giúp Quản trị viên biết ngay lập tức có sự cố an ninh.
- Quản lý người dùng (U3): Thêm, xóa, hoặc chỉnh sửa quyền truy cập của Người dùng (cấp thẻ từ, đăng ký khuôn mặt).
- Giám sát (U4): Xem lại toàn bộ lịch sử ra vào và các báo cáo an ninh.
- Cấu hình (U5): Đây là một use case quan trọng, cho phép Quản trị viên thay đổi quy tắc nghiệp vụ của hệ thống (chuyển đổi giữa chế độ 1-trong-2 và cả-2).
- Điều khiển từ xa (U6): Mở cửa từ xa thông qua một nút bấm trên web mà không cần xác thực vật lý. Hành động này cũng dẫn đến việc thực thi "Mở khóa cửa vật lý".
- Bảo trì (U7): Cập nhật phiên bản phần mềm mới cho thiết bị khóa cửa từ xa (OTA).

IV. Xác định yêu cầu phi chức năng

1. Hiệu năng

- Độ trễ đầu-cuối:
    + Quá trình xác thực và mở cửa hoàn tất trong thời gian ≤ 5 giây.
    + Cụ thể: RFID đọc thẻ ≤ 0,5 giây, nhận diện khuôn mặt ≤ 2 giây, gửi phản hồi WebSocket ≤ 1 giây.
- Tần suất xử lý:
    + Hệ thống có thể xử lý đồng thời ≥ 10 yêu cầu xác thực/giây khi triển khai ở nhiều cửa trong cùng mạng.
- Tối ưu truyền thông:
    + Dữ liệu xác thực được gửi theo sự kiện (event-based) thay vì liên tục, giúp giảm tải mạng và tăng hiệu quả.
- Khả năng chịu tải:
    + ESP32 được thiết kế để phục vụ ≥ 1000 sự kiện mở cửa/ngày mà không cần khởi động lại.
    + Server Node.js hỗ trợ kết nối song song ≥ 100 thiết bị WebSocket.

2. Bảo mật

- Mã hóa:

    + Toàn bộ dữ liệu truyền giữa thiết bị và server sử dụng giao thức WSS (WebSocket Secure) qua TLS 1.3.
    + Thông tin người dùng (UID, khuôn mặt) được băm SHA-256 và mã hóa AES-128 trước khi lưu.
- Xác thực và phân quyền:
    + Admin có toàn quyền thêm/xóa người dùng, mở khóa từ xa, thay đổi chế độ bảo mật.
    + User chỉ có quyền mở cửa và xem lịch sử ra vào của mình.
- Cảnh báo và phát hiện bất thường:
    + Hệ thống khóa tài khoản hoặc thẻ sau 5 lần xác thực sai liên tiếp.
    + Tự động gửi cảnh báo qua Telegram hoặc Email đến quản trị viên.
- Cập nhật an toàn:
    + Thiết bị hỗ trợ cập nhật OTA (Over-The-Air), có xác minh checksum để tránh firmware giả mạo.

3. Độ tin cậy

- Độ sẵn sàng (Availability): Hệ thống hoạt động ổn định 24/7 với uptime ≥ 99%.
- Phục hồi kết nối: Khi mất WiFi hoặc nguồn, ESP32 tự động reconnect và đồng bộ dữ liệu chưa gửi khi mạng khôi phục.
- An toàn dữ liệu: MongoDB lưu log truy cập, thực hiện sao lưu định kỳ (daily backup) để tránh mất dữ liệu.
- Dự phòng hoạt động: Có phương án mở khóa dự phòng từ web khi camera hoặc đầu đọc RFID gặp sự cố.

4. Khả năng mở rộng

- Mở rộng phần cứng: Cho phép kết nối thêm nhiều bộ khóa thông minh (RFID + Camera) mà không cần thay đổi hạ tầng.
- Mở rộng người dùng: Hệ thống hỗ trợ ≥ 1.000 tài khoản người dùng và 10.000 bản ghi log mà không ảnh hưởng hiệu năng.
- Mở rộng tích hợp: Có thể kết nối với các dịch vụ khác như Telegram Bot, Email, Mobile App qua API.
- Kiến trúc linh hoạt: Backend Node.js có thể triển khai dạng microservice hoặc container hóa khi mở rộng cho tòa nhà lớn.

5. Chi phí và năng lượng

- Chi phí đầu tư:
    + Tổng chi phí thiết bị cho một bộ khóa thông minh ≤ 1.500.000 VNĐ.
    + Gồm: ESP32, ESP32-CAM OV2640, RC522, Relay/Servo, nguồn, linh kiện phụ.
- Hiệu quả năng lượng:
    + Công suất tiêu thụ trung bình ≤ 5W/h.
    + Thiết bị vào deep sleep khi không phát hiện chuyển động để tiết kiệm điện.
- Tối ưu vận hành:
    + Giảm 100% chi phí làm lại chìa khóa hoặc thay ổ khóa khi mất thẻ.
    + Giảm công bảo trì nhờ OTA và hệ thống cảnh báo sớm.

V. Rủi ro đối sách

1. Mất kết nối mạng WiFi

- Rủi ro: Thiết bị ESP32 hoặc ESP32-CAM mất kết nối do sóng yếu hoặc router lỗi.
- Đối sách:
    + Cài đặt cơ chế tự động reconnect và lưu tạm dữ liệu khi mất mạng.
    + Cho phép mở khóa dự phòng qua web hoặc thủ công trong thời gian offline.

2. Sai nhận diện khuôn mặt

- Rủi ro: Ánh sáng yếu, người dùng đeo khẩu trang hoặc đứng sai vị trí khiến hệ thống nhận sai.
- Đối sách:
    + Áp dụng thuật toán cân bằng sáng, phát hiện khuôn mặt đa góc độ.
    + Hỗ trợ xác thực kép (RFID + khuôn mặt) trong chế độ bảo mật cao.

3. Mất điện đột ngột

- Rủi ro: Thiết bị ngừng hoạt động khi cúp điện, không thể mở khóa.
- Đối sách:

+ Trang bị nguồn dự phòng (UPS mini) cho bộ điều khiển, duy trì hoạt động tối thiểu 15 phút.
+ Có phương án mở khóa cơ học trong trường hợp khẩn cấp.

4. Thẻ RFID bị sao chép hoặc đánh cắp

- Rủi ro: Kẻ xấu sử dụng thiết bị đọc lén để sao chép mã UID của thẻ.
- Đối sách:
  + Mã hóa UID và xác thực với server thay vì xử lý cục bộ.
  + Cho phép vô hiệu hóa thẻ từ xa khi bị mất hoặc nghi ngờ bị sao chép.

5. Rò rỉ dữ liệu người dùng

- Rủi ro: Tấn công mạng hoặc truy cập trái phép vào cơ sở dữ liệu.
  Đối sách:
  + Sử dụng WSS (WebSocket Secure) với TLS 1.3 để mã hóa dữ liệu. Băm dữ liệu khuôn mặt bằng SHA-256 trước khi lưu trữ.
  + Chia quyền truy cập rõ ràng (Admin / User) và bật xác thực hai lớp (2FA) cho quản trị viên.

6. Hỏng camera hoặc module đọc thẻ

- Rủi ro: Thiết bị hoạt động lâu ngày, bụi bẩn, hoặc lỗi linh kiện.
- Đối sách:
  + Thiết lập kiểm tra định kỳ; cảnh báo nếu thiết bị không phản hồi.
  + Sử dụng vỏ chống bụi, chống ẩm (chuẩn IP54) để bảo vệ phần cứng.

VI. Phân tích ràng buộc kỹ thuật và môi trường

1. Môi trường hoạt động

- Nhiệt độ và độ ẩm: Các thiết bị được đặt ở trong nhà thì tương đối ổn định, tuy nhiên vẫn chịu ảnh hưởng bởi nhiệt độ, độ ẩm và bụi bẩn khi mở rộng ra các thiết bị gắn ở cửa ngoài trời hoặc hành lang; phải chịu được biên độ nhiệt từ 0 đến 50 độ C, tương thích với điều kiện khí hậu ở Việt Nam; có khả năng chống bụi/ ẩm tối

thiểu -> Cần chọn thiết bị chịu được nhiệt và vỏ bọc chống bụi, chống ẩm

- Nhiễu sóng: Khu vực nhiều thiết bị Wifi, tín hiệu 2.4GHz, … có thể gây nhiễu -> Cần chọn kênh Wifi cố định hoặc tăng công suất ăng ten ESP32
- Nguồn cấp: Cửa thường được cấp điện trực tiếp, tuy nhiên nên có sử dụng nguồn dự phòng (pin lithium nhỏ) để duy trì kết nối với Wifi hoặc lưu trạng thái trong trường hợp mất điện

2. Ràng buộc pháp lý

- ESP32 sử dụng WiFi 2.4GHz – nằm trong dải tần số ISM (miễn cấp phép) nên tuân thủ tiêu chuẩn Việt Nam. Tuy nhiên, nếu mở rộng ra môi trường công cộng (ví dụ: tòa nhà nhiều căn hộ), cần đảm bảo không gây nhiễu lẫn nhau giữa các thiết bị.
- Bảo mật và quyền riêng tư:
  Hệ thống có nhận diện khuôn mặt và lưu dữ liệu người dùng, nên phải tuân thủ nguyên tắc bảo vệ dữ liệu cá nhân:
  + Dữ liệu khuôn mặt phải mã hóa khi truyền và lưu (HTTPS/WebSocket Secure, mã hóa ảnh hoặc embedding).
  + Cần có chính sách đồng ý của người dùng khi thu thập dữ liệu nhận diện.
  + Hạn chế truy cập: chỉ quản trị viên mới có thể xem và quản lý dữ liệu.

3. Tài nguyên thiết bị

- Bộ nhớ và vi điều khiển:
  + ESP32 và ESP32-CAM có RAM và Flash hạn chế (~520 KB RAM, ~4 MB Flash), CPU chỉ 240 MHz.
  + Không thể lưu hoặc xử lý nhiều ảnh khuôn mặt trực tiếp trên thiết bị.
    → Giải pháp: chuyển việc nhận diện khuôn mặt sang server Node.js hoặc cloud, còn ESP32-CAM chỉ chụp ảnh và gửi đi.
- Tài nguyên mạng:
  + Băng thông WiFi 2.4GHz giới hạn, nếu có nhiều camera cùng gửi ảnh sẽ gây nghẽn → cần giảm độ phân giải ảnh hoặc dùng cơ chế gửi theo sự kiện (event-based)
- Nguồn điện:

+ Khi mở khóa, servo hoặc motor điện tiêu thụ dòng lớn → nguồn phải đủ ổn định, tránh sụt áp khiến ESP32 reset.

VII. Các lý thuyết và công nghệ áp dụng

1. Kiến trúc IOT

Kiến trúc 3 tầng:

- Tầng Perception (Cảm nhận): Gồm các thiết bị phần cứng thu thập dữ liệu từ môi trường.
  + Cảm biến: RFID Reader RC522 đọc mã định danh (UID) từ thẻ từ, Camera OV2640 trên ESP32-CAM thu thập hình ảnh người dùng.
  + Cơ cấu chấp hành (Actuator): Solenoid Lock hoặc Servo Motor để điều khiển chốt khóa cửa.
- Tầng Network (Mạng): Chịu trách nhiệm truyền dữ liệu an toàn và hiệu quả.
  + Giao thức kết nối: WiFi 802.11 b/g/n để kết nối ESP32 với mạng cục bộ và Internet.
  + Giao thức ứng dụng: WebSocket (WSS) thiết lập một kênh giao tiếp hai chiều, thời gian thực và được mã hóa giữa ESP32 và máy chủ Node.js. HTTP/HTTPS được sử dụng cho giao tiếp giữa frontend (React) và backend (Node.js).
- Tầng Application (Ứng dụng): Nơi dữ liệu được xử lý, lưu trữ và cung cấp giao diện cho người dùng.
  + Backend: Máy chủ Node.js xử lý logic xác thực, nhận diện khuôn mặt, và quản lý cơ sở dữ liệu.
  + Frontend: Giao diện web React.js cho phép quản trị viên quản lý người dùng, xem lịch sử và cấu hình hệ thống.
  + Database: MongoDB lưu trữ thông tin người dùng (UID thẻ, vector đặc trưng khuôn mặt), lịch sử ra vào và cấu hình hệ thống.

Edge Computing & Server Processing:

- Xử lý tại biên (Edge Processing): Các tác vụ cơ bản được xử lý trực tiếp trên ESP32 để phản hồi nhanh: đọc UID từ thẻ RFID, chụp ảnh, điều khiển trực tiếp chốt khóa sau khi nhận lệnh.

- Xử lý tại máy chủ (Server Processing): Các tác vụ tính toán nặng và yêu cầu logic phức tạp được xử lý trên máy chủ Node.js:
    + So sánh UID thẻ từ với cơ sở dữ liệu.
    + Chạy mô hình AI nhận diện khuôn mặt trên hình ảnh nhận được.
    + Áp dụng logic chế độ an ninh (1-trong-2 hoặc cả-2) để ra quyết định cuối cùng.

2. Trí tuệ nhân tạo và Thị giác máy tính

- Nhận diện khuôn mặt (Face Recognition):
    + Nguyên lý: Sử dụng mô hình CNN (Convolutional Neural Network) để chuyển đổi một hình ảnh khuôn mặt thành một vector đặc trưng (embedding) – một mảng số duy nhất đại diện cho khuôn mặt đó.
    + Mô hình đề xuất: Sử dụng các kiến trúc như FaceNet, ArcFace, hoặc MTCNN (để phát hiện khuôn mặt) kết hợp với một mô hình embedding.
    + Luồng xử lý:
        1. Phát hiện khuôn mặt trong ảnh (Face Detection).
        2. Trích xuất vector đặc trưng từ khuôn mặt đã phát hiện (Feature Extraction).
        3. So sánh vector này với các vector đã được lưu trong MongoDB bằng cách tính toán khoảng cách (ví dụ: khoảng cách Euclidean), nếu khoảng cách đủ nhỏ thì xác nhận danh tính.
- Computer Vision với OpenCV (trên server):
    + Tiền xử lý ảnh: Thư viện OpenCV trên máy chủ Node.js (ví dụ: opencv4nodejs) được dùng để giải mã hình ảnh JPEG nhận từ ESP32-CAM, chuyển đổi không gian màu, và cắt/resize ảnh khuôn mặt trước khi đưa vào mô hình AI.
- Image Processing trên ESP32-CAM:
    + JPEG Encoding: Nén ảnh thô từ cảm biến OV2640 sang định dạng JPEG để giảm đáng kể dung lượng dữ liệu cần truyền qua WiFi, giúp tăng tốc độ phản hồi.

3. Hệ thống nhúng và Vi điều khiển

- Vi điều khiển ESP32:
  + Dual-core Xtensa LX6 @ 240MHz: Một nhân chuyên xử lý các tác vụ mạng (WiFi/WebSocket stack), nhân còn lại dành riêng cho ứng dụng chính (đọc cảm biến, điều khiển khóa), đảm bảo hiệu năng ổn định.
  + FreeRTOS: Hệ điều hành thời gian thực tích hợp sẵn, cho phép quản lý các tác vụ đồng thời một cách hiệu quả.
- RTOS Task Management:
  + Đa nhiệm (Multi-tasking): Các tác vụ như rfid_task (liên tục kiểm tra thẻ), camera_task (chụp ảnh khi có yêu cầu), websocket_task (duy trì kết nối và trao đổi dữ liệu), và lock_control_task (điều khiển chốt khóa) chạy song song.
  + Giao tiếp giữa các task (Inter-task communication): Sử dụng Queue và Semaphore của FreeRTOS để gửi dữ liệu và tín hiệu một cách an toàn giữa các task.

4. Giao thức truyền thông

- WiFi 802.11 b/g/n (2.4GHz):
  + TCP/IP Stack: Đảm bảo dữ liệu (UID thẻ, ảnh) được gửi từ ESP32 đến máy chủ một cách toàn vẹn và đúng thứ tự.
- WebSocket:
  + Kết nối hai chiều, thời gian thực: Cho phép máy chủ gửi lệnh ("mở khóa", "yêu cầu ảnh") xuống ESP32 ngay lập tức mà không cần thiết bị phải liên tục hỏi (polling). Đồng thời, ESP32 có thể gửi trạng thái và dữ liệu lên server ngay khi có sự kiện.
  + Hiệu quả: Duy trì một kết nối TCP duy nhất, giảm độ trễ và overhead so với việc tạo nhiều kết nối HTTP.
- JSON (JavaScript Object Notation):
  + Định dạng dữ liệu: Là định dạng chuẩn để đóng gói dữ liệu trao đổi qua WebSocket và API giữa frontend và backend.
  + Ví dụ: {"event": "authRequest", "uid": "AB:CD:EF:12", "imageData": "base64_encoded_string..."} hoặc {"command": "unlock"}.

5. Cảm biến và Điều khiển

- Cảm biến RFID RC522:

    + Nguyên lý: Hoạt động dựa trên cảm ứng điện từ ở tần số 13.56 MHz. Đầu đọc tạo ra một trường điện từ, thẻ từ đi vào vùng này sẽ được cấp năng lượng và gửi lại mã UID của nó.
    + Giao tiếp: Sử dụng giao thức SPI (Serial Peripheral Interface) để giao tiếp với vi điều khiển ESP32, cho tốc độ đọc dữ liệu nhanh và ổn định.
- Cơ cấu chấp hành: Solenoid Lock hoặc Servo Motor:
    + Nguyên lý (Solenoid): Sử dụng một cuộn cảm để tạo ra từ trường, hút hoặc đẩy một lõi sắt để đóng/mở chốt khóa. Điều khiển đơn giản bằng cách cấp hoặc ngắt nguồn điện qua một relay hoặc transistor.
    + Nguyên lý (Servo): Cho phép điều khiển góc quay chính xác (ví dụ: quay 90 độ để mở chốt). Điều khiển bằng tín hiệu PWM (Pulse Width Modulation).

6. Thuật toán và Xử lý

- Thuật toán Logic xác thực (trên server):
    + State Machine: Chờ -> Nhận yêu cầu -> Xác thực -> Gửi lệnh -> Chờ.
    + Rule-based logic:
        - Bước 1: Nhận dữ liệu (UID thẻ và/hoặc ảnh) từ ESP32.
        - Bước 2: Truy vấn cơ sở dữ liệu MongoDB để kiểm tra tính hợp lệ của UID và/hoặc khuôn mặt.
        - Bước 3: Đọc security_mode hiện tại từ MongoDB.
        - Bước 4 (Logic cốt lõi):
            + Nếu mode == "OR": Chỉ cần (UID hợp lệ HOẶC khuôn mặt hợp lệ) -> Gửi lệnh "mở khóa".
            + Nếu mode == "AND": Yêu cầu (UID hợp lệ VÀ khuôn mặt hợp lệ) -> Gửi lệnh "mở khóa".
            + Nếu không thỏa mãn -> Gửi lệnh "từ chối".

7. Cập nhật Firmware OTA

- OTA (Over-The-Air) Update:
    + Cơ chế: Flash của ESP32 được phân chia thành hai vùng ứng dụng. Khi cập nhật, firmware mới sẽ được ghi vào vùng

không hoạt động. Sau khi xác thực thành công, ESP32 sẽ khởi động lại từ vùng mới này.
    + Lợi ích: Cho phép nâng cấp tính năng hoặc vá lỗi bảo mật cho thiết bị từ xa mà không cần can thiệp vật lý.

8. Bảo mật

- WPA2/WPA3 Encryption: Mã hóa kết nối WiFi giữa ESP32 và router để chống nghe lén trên mạng cục bộ.
- WSS (WebSocket Secure) và HTTPS: Sử dụng TLS/SSL để mã hóa toàn bộ dữ liệu truyền giữa ESP32 và server, cũng như giữa trình duyệt người dùng và server, ngăn chặn tấn công Man-in-the-Middle.
- Bảo mật ứng dụng:
    + Xác thực & Phân quyền: Sử dụng token (ví dụ: JWT - JSON Web Tokens) để bảo vệ các API của backend, đảm bảo chỉ quản trị viên mới có quyền thực hiện các hành động nhạy cảm.
    + Mật khẩu: Mật khẩu của người dùng quản trị phải được băm (hash) bằng các thuật toán mạnh như bcrypt trước khi lưu vào MongoDB.

9. Công nghệ phát triển ứng dụng

- Backend (Node.js):
    + Framework: Express.js để xây dựng các API RESTful và quản lý request.
    + Database: Mongoose là một ODM (Object Data Modeling) để tương tác với MongoDB một cách có cấu trúc.
    + Real-time: Thư viện ws để tạo và quản lý WebSocket server.
- Frontend (React.js):
    + UI Library: Xây dựng giao diện người dùng dựa trên các Component có thể tái sử dụng.
    + State Management: Sử dụng các Hooks như useState, useEffect để quản lý trạng thái của ứng dụng.
    + API Communication: Dùng Axios hoặc fetch API của trình duyệt để giao tiếp với backend.
- Môi trường lập trình cho ESP32:
    + PlatformIO: Một IDE và công cụ build tự động hóa, giúp quản lý thư viện và môi trường phát triển cho hệ thống

nhúng một cách chuyên nghiệp, hỗ trợ framework Arduino hoặc ESP-IDF.