ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KĨ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN

HỆ THỐNG KHÓA CỬA THÔNG MINH

Giảng viên hướng dẫn: **ThS. Nguyễn Duy Xuân Bách**

Sinh viên thực hiện:

Trương Thiên Quý	**23521321**
Lê Huỳnh Thanh Phương	**23521242**
Ngô Đỗ Anh Quân	**23521257**
Lê Nhật Quang	**23521283**

TP.Hồ Chí Minh, tháng 12 năm 2025

Mục lục

Danh mục hình ảnh

Danh mục bảng biểu

Thông tin nhóm thực hiện

- Nhóm: 21
- Lớp: CE224.Q15

STT	Họ và tên	MSSV	Nhiệm vụ
1	Trương Thiên Quý	23521321	Hiện thực phần cứng, viết chương trình, làm web, kiểm thử, quay demo, làm mô hình
2	Lê Huỳnh Thanh Phương	23521242	Hiện thực phần cứng, viết chương trình, quay demo, kiểm thử, làm mô hình
3	Ngô Đỗ Anh Quân	23521257	Hiện thực phần cứng ,viết báo cáo, làm slide, kiểm thử
4	Lê Nhật Quang	23521283	Hiện thực phần cứng, viết báo cáo, viết chương trình, kiểm thử

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong bối cảnh xã hội ngày càng phát triển, nhu cầu đảm bảo an ninh và an toàn cho nhà ở, văn phòng và các khu vực riêng tư ngày càng được quan tâm. Các loại khóa cơ truyền thống vẫn tồn tại nhiều hạn chế như dễ bị sao chép chìa khóa, khó quản lý người ra vào và không có khả năng lưu trữ lịch sử truy cập.

Cùng với sự phát triển mạnh mẽ của lĩnh vực điện tử – vi điều khiển – IoT, các hệ thống khóa cửa thông minh ra đời nhằm khắc phục những hạn chế trên, cho phép mở cửa bằng nhiều phương thức khác nhau như mật khẩu, thẻ từ, hoặc thông qua Internet, đồng thời có thể quản lý và giám sát lịch sử ra vào.

Xuất phát từ nhu cầu thực tế đó, nhóm đã lựa chọn đề tài "Hệ thống khóa cửa thông minh" nhằm nghiên cứu, thiết kế và hiện thực một mô hình khóa cửa có tính ứng dụng cao, dễ triển khai và phù hợp với điều kiện học tập cũng như khả năng kỹ thuật của sinh viên.

1.2. Mục tiêu của đề tài

Mục tiêu của đồ án là thiết kế và xây dựng thành công một hệ thống khóa cửa thông minh có các chức năng cơ bản và mở rộng, cụ thể:

- Cho phép mở cửa bằng mật khẩu thông qua bàn phím ma trận.
- Cho phép mở cửa bằng thẻ RFID với khả năng quản lý danh sách thẻ hợp lệ.
- Xây dựng chế độ quản trị (Admin Mode) để thêm, xóa thẻ và thay đổi mật khẩu.
- Hiển thị trạng thái hệ thống thông qua màn hình LCD 1602 giao tiếp I2C.
- Điều khiển động cơ servo/relay để mô phỏng cơ cấu đóng mở cửa.
- Kết nối WiFi thông qua ESP8266 để:
    - Mở cửa từ xa qua giao diện web.
    - Ghi nhận và lưu trữ lịch sử ra vào (thời gian, phương thức, UID/mật khẩu).
- Lưu trữ dữ liệu quan trọng (mật khẩu, thẻ RFID) bằng EEPROM nhằm đảm bảo không mất dữ liệu khi mất nguồn.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

- Vi điều khiển và các module ngoại vi phục vụ cho hệ thống khóa cửa thông minh.
- Giao tiếp RFID, keypad, LCD, servo, relay và ESP8266.
- Giao tiếp dữ liệu giữa hệ thống nhúng và giao diện web.

1.3.2. Phạm vi nghiên cứu

- Đề tài tập trung vào mô hình khóa cửa thông minh ở mức sinh viên, phục vụ mục đích học tập và nghiên cứu.
- Hệ thống hoạt động trong phạm vi mạng nội bộ thông qua WiFi.
- Web được xây dựng ở mức cơ bản bằng HTML, phục vụ chức năng mở cửa và xem lịch sử truy cập.

- Chưa đi sâu vào các kỹ thuật bảo mật nâng cao như mã hóa SSL, xác thực đa lớp hay lưu trữ cloud.

1.4. Phương pháp thực hiện

Để hoàn thành đồ án, nhóm đã áp dụng các phương pháp sau:

- Nghiên cứu tài liệu liên quan đến khóa cửa thông minh, RFID, ESP8266 và giao tiếp I2C.
- Thiết kế phần cứng bằng cách lựa chọn và kết nối các module phù hợp.
- Thiết kế phần mềm theo dạng chương trình điều khiển, chia thành các khối chức năng rõ ràng.
- Lập trình và kiểm thử từng chức năng riêng lẻ trước khi tích hợp toàn hệ thống.
- Đánh giá kết quả thực nghiệm thông qua việc vận hành thực tế mô hình.

1.5. Ý nghĩa khoa học và thực tiễn của đề tài

1.5.1. Ý nghĩa khoa học

- Giúp sinh viên củng cố kiến thức về vi điều khiển, lập trình nhúng và giao tiếp ngoại vi.
- Làm quen với việc thiết kế một hệ thống hoàn chỉnh kết hợp giữa phần cứng – phần mềm – web.
- Rèn luyện tư duy phân tích, thiết kế và triển khai hệ thống nhúng.

1.5.2. Ý nghĩa thực tiễn

- Mô hình có thể áp dụng cho các hệ thống khóa cửa đơn giản trong thực tế.
- Có khả năng mở rộng thêm nhiều tính năng thông minh hơn trong tương lai.
- Phù hợp để làm nền tảng cho các nghiên cứu hoặc đồ án nâng cao.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

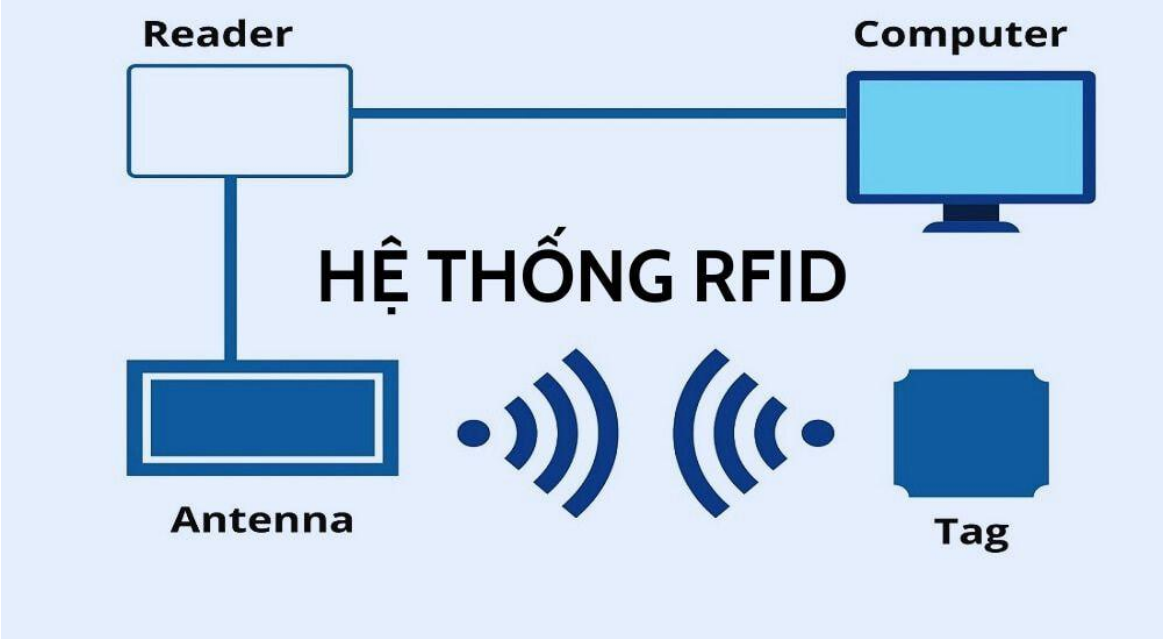
2.1. Nguyên lý hoạt động của hệ thống khóa cửa RFID

2.1.1. Tổng quan

Hệ thống khóa cửa RFID (Radio-Frequency IDentification) là hệ điều khiển cửa tự động cho phép mở/khóa dựa trên nhận dạng thẻ hoặc tag RFID. Trong mô hình này, mỗi người dùng sẽ được cấp một thẻ RFID mang mã định danh (UID – Unique Identifier). Khi thẻ được đưa vào vùng quét, module đọc RFID sẽ thu nhận thông tin và gửi đến bộ vi điều khiển để xử lý.

2.1.2. Cấu trúc hệ thống RFID

- Hệ thống RFID cơ bản bao gồm:
- **Thẻ RFID (Tag/Card)**: Chứa một mã UID cố định, không thể thay đổi. Thẻ hoạt động thụ động (không cần nguồn nuôi).
- **Đầu đọc RFID (RFID Reader – MFRC522)**: Phát sóng RF tần số 13.56 MHz để kích hoạt thẻ và thu thông tin UID.
- Bộ vi điều khiển xử lý dữ liệu (Arduino UNO).
- Cơ cấu chấp hành (Servo khóa cửa).



Hình 1. Hệ thống RFID

2.1.3. Nguyên lý giao tiếp giữa đầu đọc và thẻ

1. Reader tạo ra trường điện từ HF (13.56 MHz).
2. Thẻ RFID thụ động nhận năng lượng từ trường này → kích hoạt chip bên trong.
3. Thẻ sẽ phản hồi UID bằng kỹ thuật biến đổi trường (Load Modulation).
4. Reader thu tín hiệu phản hồi và giải mã để lấy UID.

5. Arduino nhận UID → so sánh với danh sách UID hợp lệ (lưu trong EEPROM hoặc nhận từ Web qua ESP).
6. Nếu hợp lệ → kích servo mở cửa.

2.1.4. Quy trình giao tiếp và xác thực

Quá trình xác thực diễn ra theo các bước nghiêm ngặt, đòi hỏi sự phối hợp giữa phần cứng và phần mềm:

1. **Khởi tạo (Power-up):** Đầu đọc Module RFID (ví dụ: RC522) liên tục phát sóng 13.56 MHz và chờ đợi.
2. **Kích hoạt (Activation):** Khi thẻ được đưa vào phạm vi hoạt động (thường là (< 10cm) năng lượng cảm ứng khởi động chip trên thẻ.
3. **Trao đổi dữ liệu (Data Exchange):**
    - Thẻ gửi một tín hiệu phản hồi chứa mã **UID (Unique Identifier)** duy nhất về đầu đọc. Mã UID này thường có độ dài 4 byte hoặc 7 byte.
    - Đầu đọc nhận và giải mã tín hiệu này, sau đó truyền mã UID đã xử lý qua giao tiếp **SPI** đến Vi điều khiển **Arduino Uno**.
4. **Xử lý và Truyền thông mạng (Network Communication):**
    - Arduino Uno nhận UID và chuyển tiếp mã này đến **Module ESP** thông qua giao tiếp **UART (Serial)**.
    - Module ESP (Web Client) đóng gói UID và các thông tin cần thiết khác, thiết lập kết nối **HTTP POST/GET** với **Web Quản lý** (Server).
5. **Xác thực phía Server (Server-side Authentication):**
    - Server nhận yêu cầu, truy vấn **Cơ sở dữ liệu (Database)** để so sánh UID nhận được với danh sách các UID hợp lệ (cùng với trạng thái và quyền hạn của User).
    - Server gửi lại phản hồi (ví dụ: mã **JSON** hoặc **text** đơn giản) cho ESP, bao gồm trạng thái xác thực (**Authorized/Denied**) và thông tin người dùng (tên).
6. **Điều khiển Cơ cấu chấp hành (Actuation):**
    - ESP Module chuyển tiếp kết quả xác thực cho Arduino Uno.
    - **Nếu Hợp lệ:** Arduino kích hoạt **Động cơ Servo** bằng tín hiệu **PWM** để mở khóa trong T giây, đồng thời hiển thị thông báo trên **LCD**.
    - **Nếu Không hợp lệ:** Hiển thị cảnh báo trên LCD, ghi log lỗi và giữ khóa ở trạng thái đóng.
7. **Quản lý Log (Logging):** Thông tin giao dịch (UID, thời gian, trạng thái) được lưu trữ trên Server qua kết nối mạng, phục vụ cho việc quản lý và báo cáo truy cập.

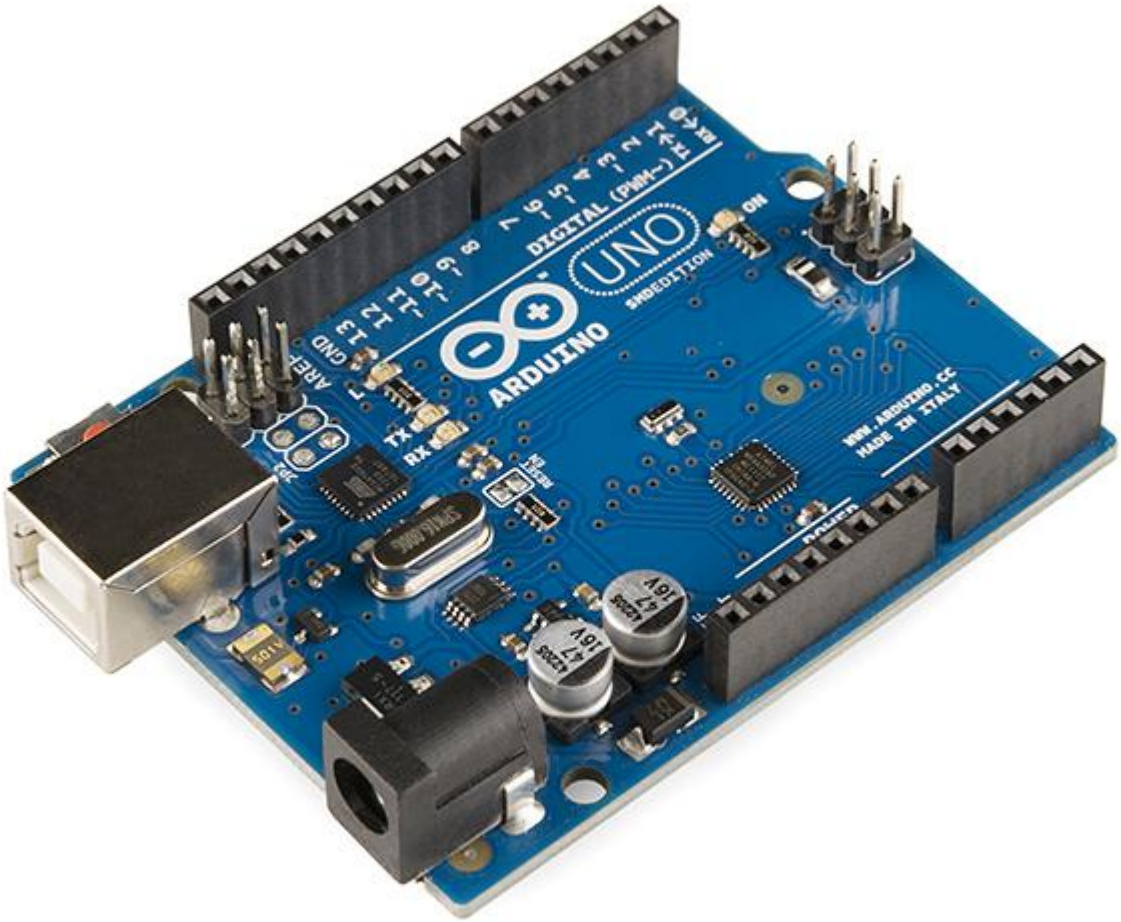
2.2. Vi điều khiển sử dụng trong hệ thống

Hệ thống khóa cửa thông minh của đồ án sử dụng cấu trúc **Vi điều khiển kép (Dual-Microcontroller Architecture)**, bao gồm **Arduino Uno** chịu trách nhiệm xử lý ngoại vi thời gian thực và **ESP Module** đảm nhiệm chức năng kết nối mạng Wi-Fi. Sự kết

hợp này tối ưu hóa hiệu suất và khả năng kết nối của toàn bộ hệ thống.

2.2.1. Arduino Uno (Bộ điều khiển logic và ngoại vi)

Arduino Uno đóng vai trò là **bộ điều khiển trung tâm (Main Controller)**, quản lý trực tiếp các thiết bị đầu vào/đầu ra (I/O) cần xử lý tốc độ cao và logic điều khiển chính xác.



Hình 2. Kit Arduino Uno R3

2.2.1.1. Kiến trúc và Cấu hình Phần cứng

Đặc tính	**Thông số Chi tiết**	**Ghi chú & Tầm quan trọng**
Vi điều khiển Chính	ATmega328P (8-bit AVR Microcontroller)	Lõi xử lý RISC (Reduced Instruction Set Computer), hiệu quả trong các tác vụ I/O.

Tốc độ Xung nhịp	16 MHz	Tốc độ đủ để xử lý giải mã RFID và tạo xung PWM điều khiển Servo đồng thời.
Bộ nhớ Flash	32 KB	Lưu trữ chương trình (firmware) của hệ thống.
SRAM (RAM tĩnh)	2 KB	Lưu trữ các biến, stack, và bộ đệm (buffer) dữ liệu UID, Serial.
EEPROM (Bộ nhớ không bay hơi)	1 KB	Có thể dùng để lưu trữ một số UID thẻ master cục bộ hoặc cấu hình mạng (nếu không dùng ESP để lưu).
Điện áp Hoạt động	5V	Tương thích trực tiếp với hầu hết các module ngoại vi.
Số lượng Chân I/O	14 chân Digital (6 chân PWM), 6 chân Analog	Đủ tài nguyên để kết nối RC522 (SPI), LCD (I2C), Servo (PWM), và ESP (UART).

Bảng 1. Cấu hình phần cứng vi điều khiển ATmega328P

2.2.1.2. Vai trò và chức năng chi tiết

Quản lý giao tiếp RFID:

- Sử dụng giao thức SPI để nhận mã UID từ Module RC522. Đây là giao thức đồng bộ, cần tốc độ cao để đảm bảo dữ liệu UID được đọc nhanh và chính xác ngay khi thẻ được đưa vào.
- Sau khi nhận UID, Arduino thực hiện bước kiểm tra sơ bộ (ví dụ: kiểm tra UID có phải là 0 hoặc lỗi đọc).

Giao tiếp và Điều khiển Hiển thị:

- Sử dụng giao tiếp I2C (tiết kiệm chân I/O) để gửi lệnh và dữ liệu hiển thị trạng thái lên Màn hình LCD.
- Hiển thị thông tin theo thời gian thực: Trạng thái kết nối (WiFi OK/FAIL), kết quả xác thực (Access Granted/Denied), Tên người dùng.

Điều khiển Cơ cấu Chấp hành (Servo):

- Dùng chức năng PWM (Pulse Width Modulation) để tạo tín hiệu điều khiển góc quay cho Động cơ Servo.
- Sau khi nhận phản hồi "Authorized" từ ESP, Arduino gửi xung PWM để Servo quay 90 độ (180 độ) để mở khóa, duy trì trong T giây, sau đó gửi xung PWM ngược lại để quay về vị trí đóng khóa.

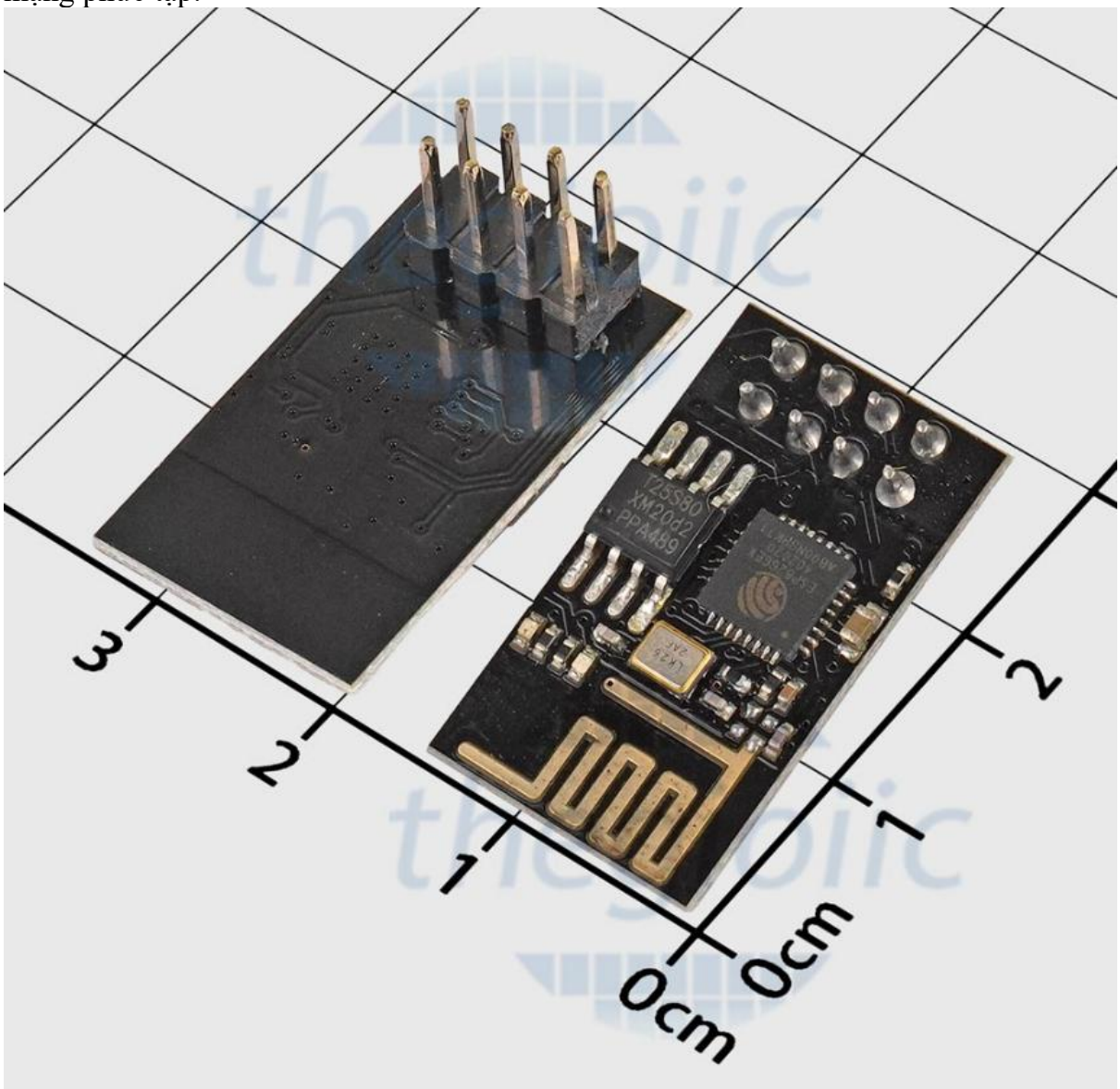
Điều phối Giao tiếp Mạng (UART - Serial):

- Arduino Uno giao tiếp với ESP Module thông qua cổng Serial (UART).

- Gửi dữ liệu: Khi có UID mới, Arduino gửi chuỗi ký tự UID này (thường dùng lệnh AT hoặc giao thức tùy chỉnh) đến ESP.
- Nhận phản hồi: Arduino chờ và nhận chuỗi phản hồi từ ESP (ví dụ: "AUTH_OK,TEN_USER", "AUTH_FAIL") để đưa ra quyết định điều khiển Servo và LCD.

2.2.2. ESP Module (Bộ điều khiển kết nối Wi-Fi)

ESP Module (thường là **ESP8266** hoặc **ESP-01**) được sử dụng như một bộ xử lý chuyên dụng cho các tác vụ kết nối mạng, giải phóng Arduino Uno khỏi các giao thức mạng phức tạp.



Hình 3. Module Wifi ESP8266

2.2.2.1. Kiến trúc và Khả năng Kết nối

Đặc tính	Thông số Chi tiết	Ghi chú & Tầm quan trọng

Vi điều khiển Chính	Tensilica L106 32-bit RISC	Lõi mạnh mẽ hơn ATmega328P, cần thiết cho việc xử lý TCP/IP Stack.
Tần số Hoạt động	80 MHz(hoặc 160 MHz	Tốc độ xử lý mạng nhanh.
Bộ nhớ Flash	512 KB đến 4 MB	Đủ để lưu trữ toàn bộ TCP/IP Stack và chương trình quản lý HTTP Client.
Kết nối Wi-Fi	Chuẩn 802.1b/g/n	Hỗ trợ kết nối mạng không dây 2.4GHz.
Giao thức Mạng	Tích hợp TCP/IP Stack	Khả năng tự xử lý mọi lớp mạng, từ vật lý đến ứng dụng (HTTP).

Bảng 2. Cấu hình phần cứng và khả năng kết nối Wi-Fi của ESP8266

2.2.2.2. Vai trò và Chức năng Chi tiết

1. **Quản lý Kết nối Wi-Fi:**
   - Khởi tạo kết nối với mạng Wi-Fi cục bộ (Station mode).
   - Duy trì kết nối, xử lý lỗi mất kết nối (Reconnection logic).
   - Quản lý thông tin đăng nhập Wi-Fi (SSID, Password).
2. **Thiết lập Giao thức HTTP/HTTPS:**
   - Đóng vai trò là **HTTP Client**. Nó tạo ra các yêu cầu **HTTP (POST/GET)** để gửi UID thẻ lên **Web Quản lý (Web Server)**.
   - Ví dụ về Yêu cầu: ESP gửi yêu cầu chứa UID đến một API cụ thể trên server:
     POST /api/authenticate_rfid với payload là {"uid": "E004B0000000"}.
3. **Xử lý Dữ liệu Phản hồi:**
   - Nhận và xử lý phản hồi từ Server (thường ở định dạng **JSON** hoặc **Plain Text**).
   - **Phân tích:** Trích xuất trạng thái xác thực (status: "OK" / "status: "DENIED") và các dữ liệu liên quan khác (ví dụ: tên người dùng).
   - **Rút gọn:** Chuyển đổi phản hồi phức tạp thành chuỗi dữ liệu ngắn gọn và dễ xử lý cho Arduino.
4. **Giao tiếp UART với Arduino:**
   - ESP hoạt động như một **Slave** hoặc **bộ chuyển đổi Serial-to-Network**.
   - Nhận UID thông qua Serial.
   - Gửi phản hồi xác thực đã được xử lý qua Serial ngược lại cho Arduino.

2.2.3. Sơ đồ Giao tiếp Vi điều khiển (Arduino <=> ESP)

Đây là điểm mấu chốt của cấu trúc Dual-Microcontroller:

Thành phần	Giao tiếp	Tốc độ Baud thường dùng	Chức năng Trao đổi Dữ liệu
Arduino Uno (TX/RX)	**UART (Serial)**	9600, 115200 bps	**Arduino gửi:** UID thẻ (VD: "UID:12345678") **ESP gửi:** Kết quả xác thực (VD: "OK:UserA" hoặc "FAIL")
ESP Module (RX/TX)	**UART (Serial)**	9600, 115200 bps	Kết nối trực tiếp chân TX của Arduino với RX của ESP và ngược lại.

Bảng 3. Đặc tả truyền thông Serial giữa Arduino Uno và ESP

Ưu điểm của cấu trúc này:

- **Tối ưu hóa Tài nguyên:** Arduino Uno tập trung vào logic thời gian thực và I/O (Servo, RFID), vốn là thế mạnh của kiến trúc AVR.
- **Xử lý Mạng Độc lập:** ESP Module chịu tải toàn bộ các giao thức TCP/IP nặng nề, tránh làm nghẽn quá trình xử lý I/O quan trọng trên Arduino.
- **Module hóa:** Nếu cần thay đổi nền tảng mạng (ví dụ: từ HTTP sang MQTT), chỉ cần thay đổi chương trình trên ESP mà không ảnh hưởng đến logic khóa cửa trên Arduino.

2.3. Thiết bị ngoại vi

Các thiết bị ngoại vi là cầu nối giúp Vi điều khiển (Arduino Uno và ESP) tương tác với thế giới vật lý, thực hiện các chức năng cốt lõi như nhận dạng người dùng, hiển thị trạng thái và điều khiển cơ chế khóa.

2.3.1. Màn hình LCD (Liquid Crystal Display) I2C

LCD là thiết bị **Đầu ra Hiển thị** (Display Output), cung cấp thông tin phản hồi trực quan và trạng thái hệ thống cho người dùng.

2.3.1.1. Đặc điểm và Cấu trúc

- **Loại phổ biến: LCD 16x2** (16 ký tự trên 2 dòng) hoặc **LCD 20x4** (20 ký tự trên 4 dòng).
- **Module I2C:** Sử dụng chip mở rộng I/O **PCF8574** gắn kèm.
- **Chức năng:** Hiển thị thông báo, hướng dẫn và kết quả xác thực.

2.3.1.2. Nguyên lý Hoạt động I2C

Việc sử dụng Module I2C là chiến lược quan trọng để tiết kiệm chân I/O trên Arduino Uno, từ 6 chân I/O tiêu chuẩn xuống còn 2 chân giao tiếp nối tiếp:

1. **Giao tiếp I2C:** Sử dụng giao thức nối tiếp hai dây:
    - **SDA (Serial Data):** Truyền dữ liệu.
    - **SCL (Serial Clock):** Xung đồng hồ đồng bộ.

2. **Chip PCF8574:** Chip này hoạt động như một **Bộ Mở rộng Cổng (Port Expander)**. Arduino gửi dữ liệu hiển thị theo gói I2C (bao gồm địa chỉ Slave của PCF8574) đến chip này.
3. **Chuyển đổi Dữ liệu:** PCF8574 nhận dữ liệu I2C, sau đó chuyển đổi thành tín hiệu song song 8 bit để điều khiển các chân của LCD (như Register Select, Enable, và các chân dữ liệu D4-D7), từ đó hiển thị ký tự lên màn hình.

Hình 4. LCD và module I2C

2.3.2. Động cơ Servo (Cơ cấu chấp hành)

Động cơ Servo đóng vai trò là **Cơ cấu Chấp hành** (Actuator), thực hiện lệnh vật lý đóng hoặc mở khóa cửa.

2.3.2.1. Đặc điểm và Cấu trúc

- **Loại thường dùng:** Servo chuẩn hoặc Micro Servo (ví dụ: SG90).
- **Cấu tạo:** Bao gồm một động cơ DC, một hộp số giảm tốc, một mạch điều khiển và một chiết áp (potentiometer) để theo dõi vị trí góc quay hiện tại.
- **Ngõ vào Điều khiển:** Nhận một tín hiệu **PWM** (Pulse Width Modulation) duy nhất.

2.3.2.2. Nguyên lý Điều khiển PWM

1. **Tín hiệu Đầu vào:** Arduino Uno sử dụng thư viện Servo.h để tạo ra chuỗi xung **PWM** trên một chân Digital (ví dụ: D9 hoặc D10). Tần số của xung PWM tiêu chuẩn cho Servo là 50 Hz, tương ứng với chu kỳ 20 ms.
2. **Xác định Góc quay:** Servo xác định góc quay theo **độ rộng (width)** của xung cao (High-pulse) trong chu kỳ 20 ms:
    - **Xung ≈ 1.0 ms:** Tương ứng với vị trí góc 0 độ (hoặc -90 độ tùy Servo).
    - **Xung ≈ 1.5 ms:** Tương ứng với vị trí góc 90 độ(Vị trí trung tâm).

- **Xung ≈ 2.0 ms:** Tương ứng với vị trí góc 180 độ (hoặc +90 độ).

3. **Vòng lặp Điều khiển:** Mạch điều khiển bên trong Servo so sánh độ rộng xung nhận được với vị trí góc hiện tại (đo bằng chiết áp) và điều chỉnh động cơ DC để quay trục đến vị trí góc mong muốn.

Hình 5. Servo

2.3.3. Cảm Biến Từ MC-38A (Magnetic Contact Sensor)

Cảm biến từ MC-38A là một thiết bị an ninh cơ bản, được sử dụng để xác định trạng thái vật lý của cửa (đang đóng hay đang mở).

2.3.3.1. Đặc điểm và Cấu tạo

- **Cấu tạo:** Cảm biến MC-38A bao gồm hai phần chính:
  1. **Phần Chính (Switch Unit):** Chứa một **công tắc lưỡi gà (Reed Switch)** được niêm phong trong một ống thủy tinh. Công tắc này có các lá kim loại mỏng, linh hoạt, được làm từ vật liệu từ tính.
  2. **Phần Từ tính (Magnet Unit):** Chứa một nam châm vĩnh cửu.
- **Loại NO (Normally Open):** Trong trạng thái bình thường (khi nam châm không ở gần), công tắc lưỡi gà **mở (Open)**, không dẫn điện. Khi nam châm ở gần, nó sẽ đóng lại và dẫn điện.
- **Ứng dụng trong Đồ án:** Phần nam châm được gắn vào cánh cửa, và phần công tắc lưỡi gà được gắn cố định vào khung cửa.

2.3.3.2. Nguyên lý Hoạt động

1. **Trạng thái Cửa Đóng:** Khi cánh cửa đóng lại, nam châm và công tắc lưỡi gà

nằm gần nhau. Trường từ tính của nam châm tác động lên các lá kim loại của công tắc, làm chúng **hút nhau và đóng mạch (Closed)**. Tín hiệu đầu ra từ cảm biến (hoặc điện áp trên chân của Arduino) sẽ thay đổi (ví dụ: chuyển từ HIGH sang LOW nếu được cấu hình Pull-up).

2. **Trạng thái Cửa Mở:** Khi cánh cửa mở ra, nam châm rời xa công tắc lưỡi gà. Không còn trường từ tính tác động, các lá kim loại sẽ trở lại vị trí ban đầu và **mở mạch (Open)**.
3. **Vai trò trong Hệ thống:**
    - **Giám sát An ninh:** Cung cấp thông tin quan trọng cho Vi điều khiển: Nếu cửa bị mở trong khi chưa được cấp quyền (sau khi quẹt thẻ), hệ thống có thể kích hoạt báo động.
    - **Logic Khóa:** Đảm bảo Servo chỉ khóa lại khi cửa thực sự đang ở trạng thái đóng (xác nhận từ MC-38A).

Hình 6. Cảm biến từ MC-38A

2.3.4. Bàn Phím Mềm Ma Trận Keypad 4x4

Bàn phím ma trận là thiết bị **Đầu vào Người dùng (User Input)** thứ cấp, cho phép người dùng nhập các mã số (ví dụ: mã PIN) để tăng cường bảo mật (xác thực hai yếu tố) hoặc nhập các lệnh cấu hình.

2.3.4.1. Đặc điểm và Cấu tạo

- **Cấu trúc Ma trận:** Bàn phím 4x4có tổng cộng 16 phím. Các phím này được

sắp xếp thành 4 hàng (Rows) và 4 cột (Columns).

- **Số chân I/O:** Mặc dù có 16 phím, nó chỉ cần 4 (Hàng) + 4 (Cột) = 8 chân I/O để giao tiếp với Arduino. Đây là ưu điểm chính của cấu trúc ma trận so với việc sử dụng 16 chân riêng biệt cho 16 phím.
- **Phím:** Thường bao gồm các phím số (0-9), các chữ cái (A-D) và các phím chức năng (* và #).

2.3.4.2. Nguyên lý Quét Phím (Keypad Scanning)

Vi điều khiển (Arduino) xác định phím nào được nhấn thông qua một quy trình gọi là **quét ma trận (Matrix Scanning)**:

1. **Thiết lập Đầu vào/Đầu ra:** Arduino cấu hình 4 chân (Cột) làm **Đầu vào (Input)** và 4 chân (Hàng) làm **Đầu ra (Output)**.
2. **Quy trình Quét:**
    - **Gửi Xung (Driving Rows):** Arduino lần lượt thiết lập từng chân Hàng ở trạng thái **LOW** (hoặc High, tùy thuộc vào logic). Ví dụ: thiết lập Hàng 1 LOW, và giữ Hàng 2, 3, 4 HIGH.
    - **Đọc Cột (Reading Columns):** Sau đó, Arduino đọc trạng thái của 4 chân Cột.
    - **Xác định Vị trí:**
        - Nếu một phím nằm trên Hàng 1 được nhấn, mạch điện sẽ được đóng.
        - Vì Hàng 1 đang ở LOW, chân Cột tương ứng sẽ kéo về **LOW**.
        - Arduino sẽ biết được: "Phím nhấn nằm ở **Hàng 1** và **Cột X**".
3. **Giải mã:** Bằng cách lặp lại quy trình này cho tất cả 4 Hàng, Arduino có thể xác định duy nhất vị trí (tọa độ) của bất kỳ phím nào được nhấn trong ma trận 4x4.
4. **Vai trò trong hệ thống:** Bàn phím được sử dụng chủ yếu cho tính năng xác thực hai yếu tố (2FA). Sau khi thẻ RFID được quẹt thành công, người dùng cần phải nhập thêm mã PIN hợp lệ qua bàn phím để truy cập, tăng cường đáng kể tính bảo mật khi thẻ bị đánh cắp. Ngoài ra, nó còn đóng vai trò là công cụ quản trị cục bộ, cho phép người dùng có mã Master nhập các lệnh cấu hình hoặc bảo trì hệ thống (Ví dụ: thay đổi mã PIN, thêm/xóa thẻ Master) mà không cần truy cập vào giao diện Web.

Hình 7. Keypad 4x4

CHƯƠNG 3. PHÂN TÍCH – THIẾT KẾ HỆ THỐNG

3.1. Phân tích hệ thống

3.1.1. Phân tích yêu cầu hệ thống

3.1.1.1. Chức năng chính

1. Xác thực người dùng
    - Hệ thống cho phép xác thực bằng:
        - Thẻ RFID (UID)
        - Mật khẩu nhập từ bàn phím
        - Lệnh mở cửa từ website
    - Mỗi phương thức xác thực đều được kiểm tra tính hợp lệ trước khi mở cửa
2. Điều khiển mở/đóng cửa
    - Khi xác thực thành công, hệ thống kích hoạt relay hoặc khóa điện để mở cửa
    - Cửa sẽ đóng sau khi người dùng đóng cửa
3. Ghi nhận lịch sử ra vào
    - Mỗi lần mở cửa đều được lưu lại thông tin:
        - Thời gian
        - Phương thức mở cửa
        - UID hoặc Password (đã được xử lý)
    - Dữ liệu được gửi về server để lưu trữ và hiển thị trên website

3.1.1.2. Chức năng phụ

- Quản lý danh sách thẻ được phép truy cập
- Cho phép thay đổi mật khẩu trực tiếp trên hệ thống
- Cập nhật thẻ Master
- Cảnh báo khi truy cập trái phép
- Giám sát trạng thái hoạt động của hệ thống

3.1.2. Phân tích module phần cứng

3.1.2.1. Module vi điều khiển trung tâm (Arduino UNO R3)

- Đóng vai trò là bộ xử lý trung tâm của hệ thống
- Nhận dữ liệu từ các module ngoại vi
- Xử lý logic xác thực và điều khiển
- Giao tiếp với server và website

3.1.2.2. Module RFID

- Đọc UID của thẻ RFID
- Gửi UID về vi điều khiển để so sánh
- Đảm bảo thời gian phản hồi nhanh và độ ổn định cao

3.1.2.3. Module bàn phím và hiển thị

- Bàn phím dùng để nhập mật khẩu và thao tác quản trị

- Màn hình LCD hiển thị:
  - Trạng thái hệ thống
  - Thông báo xác thực
  - Hướng dẫn người dùng

3.1.2.4. Module chấp hành (Khóa cửa/Servo)

- Bao gồm Servo
- Nhận tín hiệu điều khiển từ vi điều khiển
- Đảm bảo cửa chỉ mở khi có lệnh hợp lệ

3.1.2.5. Module truyền thông (ESP8266)

- Kết nối vi điều khiển với server
- Gửi và nhận dữ liệu phục vụ cho chức năng web
- Đảm bảo truyền thông ổn định

3.1.3. Phân tích module phần mềm

3.1.3.1. Lớp Driver

- Điều khiển trực tiếp phần cứng:
  - RFID
  - LCD
  - Keypad
  - Servo
- Cung cấp các hàm đọc/ghi cơ bản cho lớp trên

3.1.3.2. Lớp Middleware

- Xử lý giao thức truyền thông
- Quản lý dữ liệu gửi/nhận
- Đồng bộ dữ liệu giữa vi điều khiển và server

3.1.3.3. Lớp Ứng dụng

- Thực hiện toàn bộ logic nghiệp vụ:
  - Kiểm tra UID
  - So sánh mật khẩu
  - Xử lý lệnh mở cửa từ web
- Ghi nhận và gửi lịch sử ra vào
- Quản lý quyền truy cập

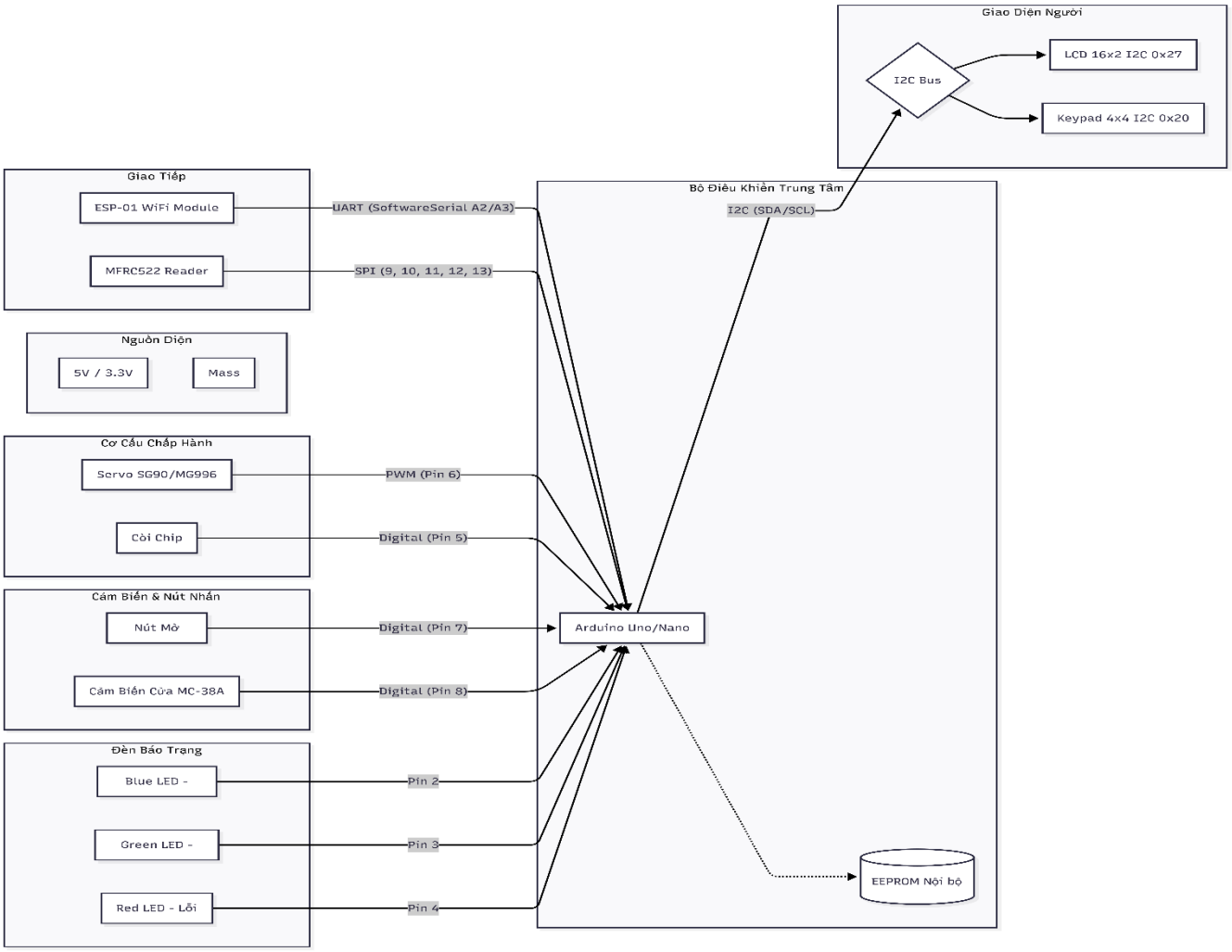
3.2. Thiết kế hệ thống

3.2.1. Thiết kế tổng thể hệ thống

Hệ thống được thiết kế theo mô hình **Client – Server – Embedded System**, trong đó:

- Website đóng vai trò giao diện người dùng
- Server xử lý dữ liệu và lưu trữ lịch sử
- Vi điều khiển thực hiện xác thực và điều khiển cửa

3.2.2. Sơ đồ khối phần cứng



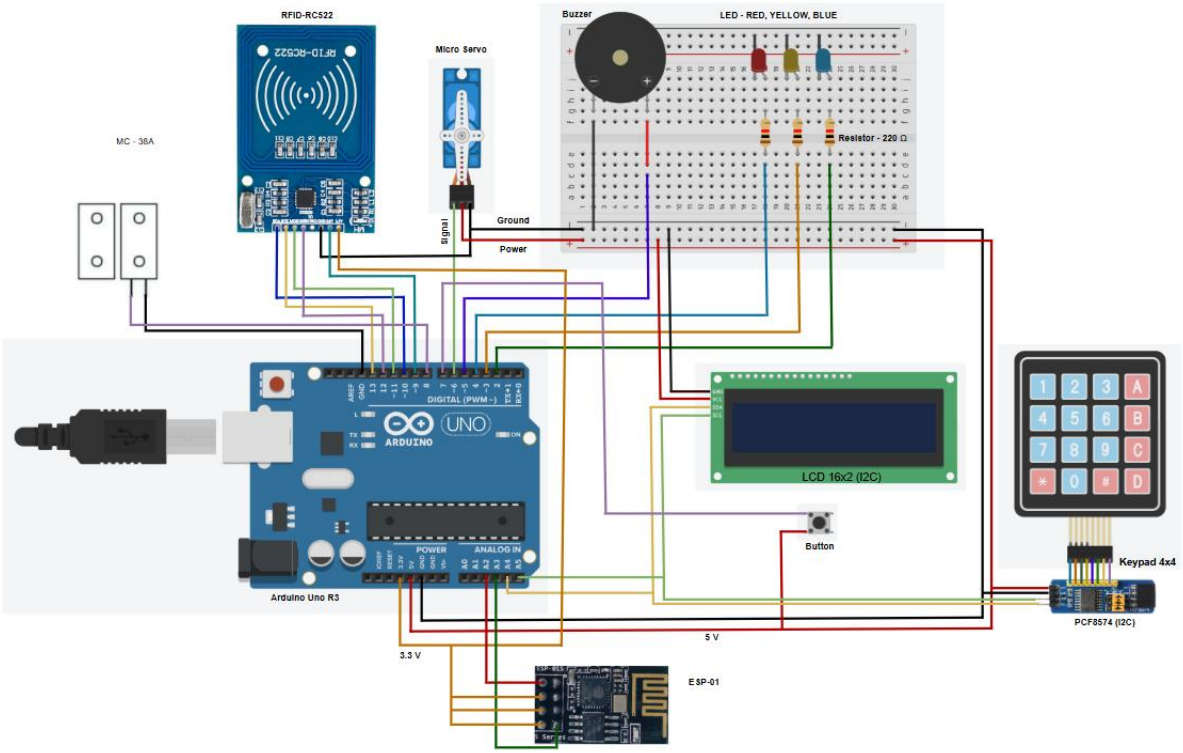
Hình 9. Sơ đồ khối phần cứng

3.2.3. Sơ đồ kết nối mạch

3.2.3.1. Sơ đồ

Sơ đồ kết nối mạch giúp thể hiện kết nối phần cứng chi tiết giữa các linh kiện trong hệ thống khóa cửa RFID, bao gồm:

- Vi điều khiển trung tâm (Arduino UNO R3)
- Module đọc thẻ RFID RC522
- Keypad 4x4
- Thiết bị đầu ra (servo, LED, buzzer)
- Thiết bị hiển thị (LCD 16x2 I2C)
- Mạch nguồn 5V – 3.3V cấp cho toàn hệ thống



Hình 10. Sơ đồ kết nối mạch

3.2.3.2. Kết nối Arduino UNO – RC522 (chuẩn SPI)

Chân RC522	Chức năng	Chân Arduino UNO
SDA (SS)	Chip Select	D10
SCK	Clock SPI	D13
MOSI	Master Out Slave In	D11
MISO	Master In Slave Out	D12
RST	Reset	D9
VCC	3.3V	3.3V
GND	Ground	GND

Bảng 4. Sơ đồ chân kết nối Arduino Uno – RC522

3.2.3.3. Kết nối Arduino UNO – LCD 16x2 (chuẩn I2C)

Chân LCD I2C	Chức năng	Chân Arduino UNO
SDA	Truyền dữ liệu	A4
SCL	Xung đồng hồ	A5
VCC	Nguồn 5V	5V
GND	Mass chung	GND

Bảng 5. Sơ đồ chân kết nối Arduino UNO – LCD 16x2

3.2.3.4. Kết nối thiết bị điều khiển và cảnh báo

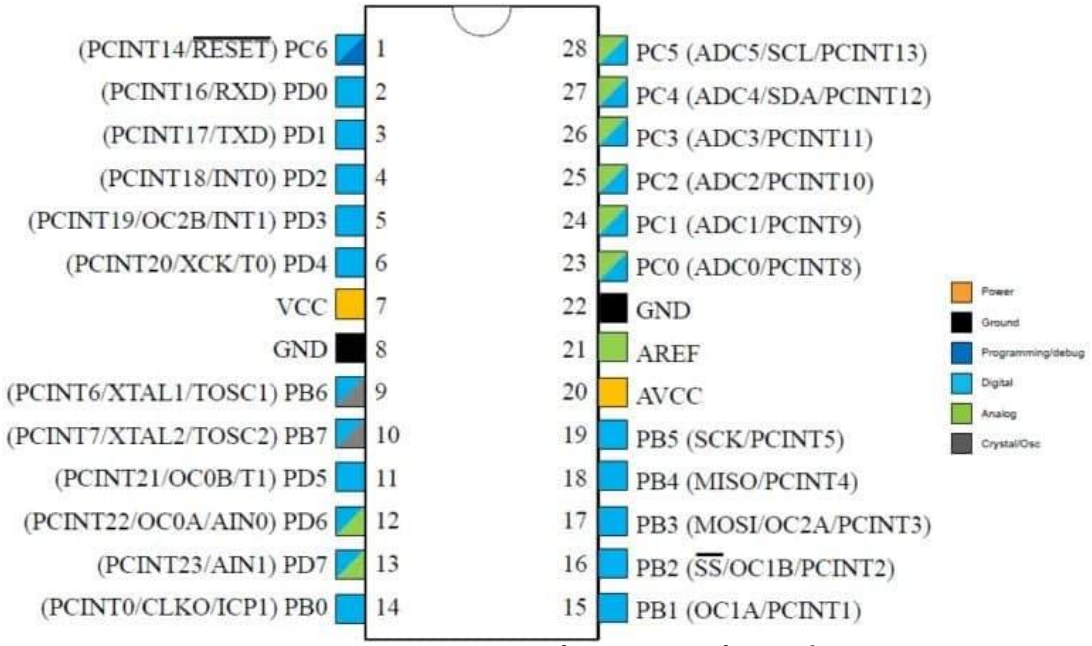
Thiết bị	Chức năng	Chân Arduino	Ghi chú
Servo SG90	Điều khiển chốt cửa	D7	PWM Output
LED xanh	Báo hợp lệ	D4	Có điện trở 220Ω nối tiếp
LED đỏ	Báo sai	D5	Có điện trở 220Ω nối tiếp
Buzzer	Cảnh báo	D6	Có thể dùng transistor điều khiển

Bảng 6. Phân công chân điều khiển Servo, LED và Buzzer

3.2.3.5. Mạch nguồn

- Arduino UNO được cấp nguồn 5V qua cổng USB hoặc adapter.
- RC522 lấy nguồn 3.3V từ Arduino.
- Các thiết bị khác (servo, LED, buzzer, LCD) dùng nguồn 5V.
- Mạch nối chung mass (GND) để đảm bảo tín hiệu ổn định.

3.2.4. Sơ đồ chân vi điều khiển



Hình 11. Sơ đồ chân vi điều khiển

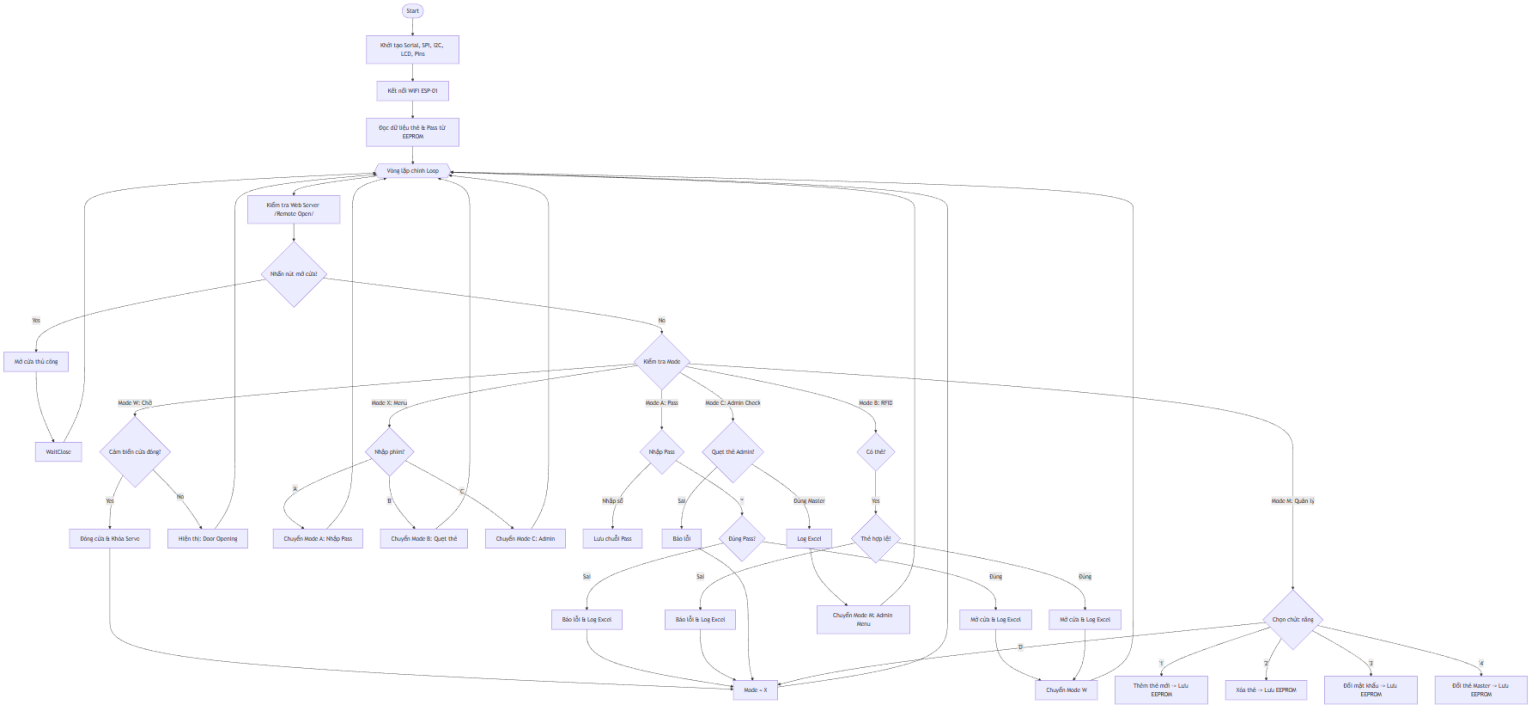
Các nhóm chân chính:

Nhóm	Chức năng	Ghi chú
Nguồn	VCC, GND, AVCC, AREF	Cấp điện 5V và tham chiếu ADC
UART	PD0 (RX) – PD1 (TX)	Giao tiếp nối tiếp (Serial Monitor)

I2C	PC4 (SDA) – PC5 (SCL)	Dùng cho LCD 16x2 I2C hoặc module khác
SPI	PB3 (MOSI), PB4 (MISO), PB5 (SCK), PB2 (SS)	Giao tiếp với RC522 RFID
PWM (tín hiệu điều chế độ rộng xung)	PD3, PD5, PD6, PB1, PB2	Dùng điều khiển servo hoặc LED
ADC (Analog Input)	PC0 → PC5 (A0–A5)	Đo tín hiệu analog nếu cần
Reset	PC6	Reset phần cứng ATmega328P

Bảng 7. Chức năng và phân loại chân của Arduino Uno

3.2.5. Thiết kế lưu đồ chương trình (Firmware)



Hình 12. Lưu đồ chương trình

CHƯƠNG 4. HIỆN THỰC HỆ THỐNG

4.1. Các thư viện cần dùng và định nghĩa các biến toàn cục

```
#include <EEPROM.h>
#include <I2CKeyPad.h>
#include <LiquidCrystal_I2C.h>
#include <MFRC522.h>
#include <SPI.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#include <Wire.h>

// ======= ESP-01 WiFi =======
SoftwareSerial myESP(A2, A3); // RX, TX (A2 nối TX của ESP, A3 nối RX của ESP)
String wifiSSID = "";
String wifiPass = "";
String devId = "v2158880920ACB6C";
// ======= RFID =======
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
String MasterTag = "97 AE 6C 05"; // Thẻ admin
String UIDCard = "";

// Danh sách thẻ được phép (tối đa 10 thẻ)
#define MAX_CARDS 10
String allowedCards[MAX_CARDS];
int cardCount = 0;

// ======= EEPROM Addresses =======
#define EEPROM_INIT_FLAG 0    // 1 byte - cờ khởi tạo
#define EEPROM_CARD_COUNT 1   // 1 byte - số lượng thẻ
#define EEPROM_CARDS_START 2  // Bắt đầu lưu thẻ (10 thẻ x 12 bytes = 120 bytes)
#define CARD_UID_LENGTH 12    // Độ dài UID tối đa
#define EEPROM_PASSWORD 122   // 10 bytes cho password
#define EEPROM_MASTER_TAG 132 // 12 bytes cho master tag
#define PASSWORD_MAX_LENGTH 10
```

Hình 13. Các thư viện sử dụng trong hệ thống

```
// ======= ESP-01 WiFi =======
SoftwareSerial myESP(A2, A3); // RX, TX (A2 nối TX của ESP, A3 nối RX của ESP)
String wifiSSID = "";
String wifiPass = "";
String devId = "v2158880920ACB6C";
// ======= RFID =======
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
String MasterTag = "97 AE 6C 05"; // Thẻ admin
String UIDCard = "";

// Danh sách thẻ được phép (tối đa 10 thẻ)
#define MAX_CARDS 10
String allowedCards[MAX_CARDS];
int cardCount = 0;

// ======= EEPROM Addresses =======
#define EEPROM_INIT_FLAG 0    // 1 byte - cờ khởi tạo
#define EEPROM_CARD_COUNT 1   // 1 byte - số lượng thẻ
#define EEPROM_CARDS_START 2  // Bắt đầu lưu thẻ (10 thẻ x 12 bytes = 120 bytes)
#define CARD_UID_LENGTH 12    // Độ dài UID tối đa
#define EEPROM_PASSWORD 122   // 10 bytes cho password
#define EEPROM_MASTER_TAG 132 // 12 bytes cho master tag
#define PASSWORD_MAX_LENGTH 10

// ======= LCD & Servo =======
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo servo;

// ======= LED & Buzzer =======
#define BlueLED 2
#define GreenLED 3
#define RedLED 4
```

Hình 14. Định nghĩa các biến toàn cục cụ thể để dễ dàng lập trình

4.2. Khởi tạo các ngoại vi và các hàm cần thiết

```
66  void setup() {
67    Serial.begin(9600);
68    myESP.begin(9600);
69    connectWifi();
70    SPI.begin();
71    mfrc522.PCD_Init();
72
73    // Khởi tạo I2C và keypad
74    Wire.begin();
75    if (keypad.begin() == true) {
76      Serial.println(F("Keypad found"));
77      keypad.loadKeyMap(keymap);
78    } else {
79      Serial.println(F("ERROR: Keypad not found"));
80    }
81
82    // Khởi tạo LCD
83    lcd.init();
84    lcd.backlight();
85    lcd.clear();
86
87    // Khởi tạo Servo
88    servo.attach(6);
89    servo.write(90); // Cửa khóa ban đầu (180 độ)
90
91    // Khởi tạo các chân GPIO
92    pinMode(GreenLED, OUTPUT);
93    pinMode(BlueLED, OUTPUT);
94    pinMode(RedLED, OUTPUT);
95    pinMode(Buzzer, OUTPUT);
96    pinMode(Open_Door_Button, INPUT_PULLUP);
97    pinMode(Door_Sensor, INPUT_PULLUP); // MC-38A với pull-up
98
```

*Hình 15. Khởi tạo các ngoại vi như **I2C, Keypad, LCD, Servo, GPIO***

```
124 void sendToExcel(String uid, String action) {
125   lcd.setCursor(0, 1);
126   lcd.print(F("Syncing Proxy..."));
127   action.replace(" ", "");
128   // Xóa bộ đệm
129   while (myESP.available())
130     myESP.read();
131
132   // Kết nối tới PushingBox (Cổng 80 - Rất nhẹ, ít tốn điện)
133   myESP.println("AT+CIPSTART=\"TCP\",\"api.pushingbox.com\",80");
134   delay(3000);
135
136   if (myESP.find("CONNECT")) {
137     Serial.println(F("Connected to PushingBox"));
138
139     // Tạo URL gửi cho PushingBox
140     // Cấu trúc: /pushingbox?devid=...&uid=...&action=...
141     String url =
142         "/pushingbox?devid=" + devId + "&uid=" + uid + "&action=" + action;
143
144     String cmd = "GET " + url + " HTTP/1.1\r\n" +
145                  "Host: api.pushingbox.com\r\n" + "Connection: close\r\n\r\n";
146
147     myESP.print(F("AT+CIPSEND="));
148     myESP.println(cmd.length());
149     delay(500); // Đợi ngắn thôi
150
151     if (myESP.find(">")) {
152       myESP.print(cmd);
153       Serial.println(F("Data Sent via HTTP! Success."));
154       long timeout = millis();
155       while (millis() - timeout < 5000) {
156         while (myESP.available()) {
157           Serial.write(myESP.read());
```

Hình 16. sendToExcel() giúp gửi dữ liệu lên Google Sheet

```
174  void connectWifi() {
175    myESP.println(F("AT+RST"));
176    delay(2000);
177    myESP.println(F("AT+CWMODE=1"));
178    delay(1000);
179
180    String cmd = "AT+CWJAP=\"" + wifiSSID + "\",\"" + wifiPass + "\"";
181    myESP.println(cmd);
182
183    lcd.clear();
184    lcd.print(F("Connecting Wifi"));
185    // Chờ kết nối (khoảng 5-8s)
186    delay(5000);
187    lcd.clear();
188    myESP.println(F("AT+CIPMUX=1")); // Cho phép nhiều kết nối
189    delay(1000);
190    myESP.println(F("AT+CIPSERVER=1,80")); // Mở cổng 80
191    delay(1000);
192
193    // Lấy IP để hiển thị cho bạn biết
194    myESP.println(F("AT+CIFSR"));
195
196    Serial.println(F("Dang lay IP..."));
197
198    // 2. Lắng nghe ESP trả lời và in ra màn hình
199    long timeout = millis();
200    while (millis() - timeout < 5000) { // Đợi 2 giây
201      while (myESP.available()) {
202        // Đọc từng chữ từ ESP và in sang Serial Monitor
203        Serial.write(myESP.read());
204      }
205    }
206    lcd.clear();
```

Hình 17. ***connectWifi()*** *thiết lập chế độ hoạt động cho ESP-01 để kết nối Wifi*

```
213  void checkWebServer() {
214    if (myESP.available()) {
215      if (myESP.find("+IPD,")) {
216        delay(300); // Đợi dữ liệu về hết
217
218        String request = "";
219        // Đọc tối đa 100 ký tự để tìm lệnh (đỡ tràn bộ nhớ)
220        for (int i = 0; i < 100; i++) {
221          if (myESP.available())
222            request += (char)myESP.read();
223        }
224
225
226        // Kiểm tra xem có lệnh "OPEN" không
227        if (request.indexOf("OPEN") != -1) {
228          Serial.println(F("WEB REQUEST: OPEN DOOR"));
229          openDoorSuccess();
230          sendToExcel("Web_User", "Remote_Open"); // Ghi log
231        }
232
233        // Đóng kết nối ngay để giải phóng tài nguyên
234        // Thử đóng ID 0 và 1 (thường là 0)
235        myESP.println(F("AT+CIPCLOSE=0"));
236        myESP.println(F("AT+CIPCLOSE=1"));
237      }
238    }
239  }
```

Hình 18. ***checkWebServer()*** *liên tục kiểm tra yêu cầu HTTP gửi đến máy chủ Web cục bộ*

4.3. Vòng lặp chính của hệ thống – loop()

```
242   // ======= KIỂM TRA CẢM BIẾN CỬA =======
243   // MC-38A NO: LOW khi gần nhau, HIGH khi xa nhau
244   checkWebServer();
245   if (mode == 'W') {
246     // Đang chờ cửa đóng
247     if (digitalRead(Door_Sensor) == HIGH) {
248       // Cảm biến chạm nhau -> đóng cửa
249       closeDoor();
250       return;
251     }
252
253     // Hiển thị trạng thái
254     lcd.setCursor(0, 0);
255     lcd.print(F("Door Opening    "));
256     lcd.setCursor(0, 1);
257     lcd.print(F("Push to close..."));
258     delay(100);
259     return;
260   }
261
262   // ======= NÚT MỞ THỦ CÔNG =======
263   if (digitalRead(Open_Door_Button) == LOW) {
264     manualOpen();
265     return;
266   }
267
```

Hình 19. Kiểm tra trạng thái cửa cửa và kiểm tra có tương tác mở cửa thủ công không

```
// ======= MENU CHÍNH =======
if (mode == 'X') {
  char key = getKey();
  if (key == 'A') {
    mode = 'A';
    lcd.clear();
    lcd.print(F("Enter Password:"));
    lcd.setCursor(0, 1);
    inputPass = "";
  } else if (key == 'B') {
    mode = 'B';
    lcd.clear();
    lcd.print(F("Scan Your Card"));
    lcd.setCursor(0, 1);
    lcd.print(F("To Open..."));
  } else if (key == 'C') {
    mode = 'C';
    lcd.clear();
    lcd.print(F("Admin Mode"));
    lcd.setCursor(0, 1);
    lcd.print(F("Scan Admin Card"));
  }
}
```

Hình 20. Menu chính của chương trình

```
291
292  // ======= CHẾ ĐỘ NHẬP MẬT KHẨU =======
293  else if (mode == 'A') {
294    char key = getKey();
295    if (key) {
296      if (key == '#') {
297        if (inputPass.length() > 0)
298          inputPass.remove(inputPass.length() - 1);
299        updatePassDisplay();
300      } else if (key == '*') {
301        checkPassword();
302      } else if (key == 'D') {
303        mode = 'X';
304        inputPass = "";
305        showMainMenu();
306      } else if (isDigit(key)) {
307        inputPass += key;
308        updatePassDisplay();
309      }
310    }
311  }
312
313  // ======= CHẾ ĐỘ RFID =======
314  else if (mode == 'B') {
315    char key = getKey();
316    if (key == 'D') {
317      mode = 'X';
318      showMainMenu();
319      return;
320    }
321
322    if (getUID()) {
323      lcd.clear();
324      lcd.setCursor(0, 0);
325      lcd.print(F("UID: "));
```

Hình 21. Đa dạng chế độ nhập mật khẩu để có thể mở cửa

4.4. Hàm mở cửa và đóng cửa

```
405  void openDoorSuccess() {
406    digitalWrite(GreenLED, HIGH);
407    digitalWrite(BlueLED, LOW);
408    digitalWrite(RedLED, LOW);
409
410    // Mở khóa: servo quay về 90 độ
411    servo.write(180);
412    doorLocked = false;
413    doorOpening = true;
414
415    for (int i = 0; i < 2; i++) {
416      tone(Buzzer, 2000);
417      delay(200);
418      noTone(Buzzer);
419      delay(200);
420    }
421
422    // Chuyển sang chế độ chờ đóng cửa
423    mode = 'W';
424
425    lcd.clear();
426    lcd.setCursor(0, 0);
427    lcd.print(F("Door Unlocked!"));
428    delay(1000);
429
430    digitalWrite(GreenLED, LOW);
431    digitalWrite(BlueLED, HIGH);
432  }
```

Hình 22. ***openDoorSuccess()*** *thực hiện chức năng mở cửa*

```
void closeDoor() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("Door Closing..."));
  // Khóa cửa: servo quay về 180 độ
  servo.write(90);
  doorLocked = true;
  doorOpening = false;

  tone(Buzzer, 1800);
  delay(300);
  noTone(Buzzer);

  delay(1000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("Door Closed!"));
  delay(1500);

  // Trở về menu chính
  mode = 'X';
  inputPass = "";
  showMainMenu();
}
```

Hình 23. closeDoor() thực hiện chức năng đóng cửa

```
460  void manualOpen() {
461    lcd.clear();
462    lcd.print(F("Manual Override"));
463    sendToExcel("Manual", "Open");
464    lcd.setCursor(0, 1);
465    lcd.print(F("Unlocking..."));
466
467    digitalWrite(GreenLED, HIGH);
468    digitalWrite(BlueLED, LOW);
469    digitalWrite(RedLED, LOW);
470
471    // Mở khóa: servo quay về 90 độ
472    servo.write(180);
473    doorLocked = false;
474    doorOpening = true;
475
476    for (int i = 0; i < 2; i++) {
477      tone(Buzzer, 2000);
478      delay(200);
479      noTone(Buzzer);
480      delay(200);
481    }
482
483    // Chuyển sang chế độ chờ đóng cửa
484    mode = 'W';
485
486    lcd.clear();
487    lcd.setCursor(0, 0);
488    lcd.print(F("Door Unlocked!"));
489    delay(1000);
490
491    digitalWrite(GreenLED, LOW);
492    digitalWrite(BlueLED, HIGH);
493  }
```

*Hình 24. **manualOpen()** thực hiện chức năng mởcửa thủ công*

4.5. Các hàm liên quan đến EEPROM

```
589  // Ghi chuỗi vào EEPROM
590  void writeStringToEEPROM(int address, String data) {
591    int len = data.length();
592    if (len > CARD_UID_LENGTH - 1)
593      len = CARD_UID_LENGTH - 1;
594
595    for (int i = 0; i < len; i++) {
596      EEPROM.write(address + i, data[i]);
597    }
598    EEPROM.write(address + len, '\0'); // Null terminator
599  }
```

*Hình 25. Hàm bổ trợ **ghi** lần lượt vào các ô nhớ cụ thể trong EEPROM*

```
601  // Đọc chuỗi từ EEPROM
602  String readStringFromEEPROM(int address) {
603    String data = "";
604    char c;
605
606    for (int i = 0; i < CARD_UID_LENGTH; i++) {
607      c = EEPROM.read(address + i);
608      if (c == '\0')
609        break;
610      data += c;
611    }
612
613    return data;
614  }
```

*Hình 26. Hàm bổ trợ **đọc** các ký tự từ các ô nhớ liên tiếp trong EEPROM*

```
498  void loadCardsFromEEPROM() {
499    // Kiểm tra cờ khởi tạo
500    byte initFlag = EEPROM.read(EEPROM_INIT_FLAG);
501
502    if (initFlag != 0xAA) {
503      // Lần đầu tiên sử dụng, khởi tạo EEPROM
504      Serial.println(F("First time init EEPROM"));
505
506      // Lưu password và master tag mặc định VÀO EEPROM
507      writeStringToEEPROM(EEPROM_PASSWORD, password);
508      writeStringToEEPROM(EEPROM_MASTER_TAG, MasterTag);
509
510      // Thêm master tag vào danh sách thẻ
511      allowedCards[0] = MasterTag;
512      cardCount = 1;
513      saveCardsToEEPROM();
514
515      // Đánh dấu đã khởi tạo
516      EEPROM.write(EEPROM_INIT_FLAG, 0xAA);
517
518      Serial.print(F("Saved default password: "));
519      Serial.println(password);
520      Serial.print(F("Saved default master tag: "));
521      Serial.println(MasterTag);
522    } else {
523      // Đọc password từ EEPROM
524      String loadedPassword = readStringFromEEPROM(EEPROM_PASSWORD);
525      if (loadedPassword.length() >= 4) {
526        password = loadedPassword;
527        Serial.print(F("Loaded password: "));
528        Serial.println(password);
529      } else {
530        Serial.println(F("Invalid password in EEPROM, using default: 1234"));
```

*Hình 27. **localCardsFromEEPROM()** kiểm tra trạng thái khởi tạo của bộ nhớ; nạp mật khẩu, mã thẻ Master và danh sách các thẻ RFID hợp lệ từ EEPROM*

```
572  void saveCardsToEEPROM() {
573    // Lưu số lượng thẻ
574    EEPROM.write(EEPROM_CARD_COUNT, cardCount);
575
576    // Lưu từng thẻ
577    for (int i = 0; i < cardCount; i++) {
578      writeStringToEEPROM(EEPROM_CARDS_START + (i * CARD_UID_LENGTH),
579                          allowedCards[i]);
580      Serial.print(F("Saved card "));
581      Serial.print(i);
582      Serial.print(F(": "));
583      Serial.println(allowedCards[i]);
584    }
585
586    Serial.println(F("All cards saved to EEPROM"));
587  }
```

*Hình 28. **saveCardsToEEPROM()** lưu lại số lượng thẻ hiện tại và danh sách mã UID của các thẻ hợp lệ vào EEPROM*

4.6. Các hàm phụ trợ thực hiện chức năng hiển thị

```
void showMainMenu() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("A:Pass B:RFID"));
  lcd.setCursor(0, 1);
  lcd.print(F("C:Admin Mode"));
}
```

*Hình 29. **showMainMenu()** hiển thị Menu chính của chương trình*

```
void showAdminMenu() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(F("1:Add 2:Remove"));
  lcd.setCursor(0, 1);
  lcd.print(F("3:PW 4:Master"));
}
```

*Hình 30. **showAdminMenu()** hiển thị Menu Admin*

```
void updatePassDisplay() {
  lcd.setCursor(0, 1);
  lcd.print(F("                "));
  lcd.setCursor(0, 1);
  for (unsigned int i = 0; i < inputPass.length(); i++)
    lcd.print("*");
}
```

*Hình 31. **updatePassDisplay()** mã hóa hiển thị Password khi người dung nhập mật khẩu*

```
643  void checkPassword() {
644    lcd.clear();
645    lcd.print(F("Checking..."));
646    delay(500);
647    lcd.clear();
648    lcd.setCursor(2, 0);
649    lcd.print(F("Permission"));
650    lcd.setCursor(0, 1);
651    if (inputPass == password) {
652      lcd.print(F("Access Granted!"));
653      sendToExcel("Pass_Entry", password);
654      openDoorSuccess();
655    } else {
656      lcd.print(F("Access Denied!"));
657      sendToExcel("Pass_Entry", "Denied");
658      accessDenied();
659      delay(2000);
660      inputPass = "";
661      mode = 'X';
662      showMainMenu();
663    }
664  }
```

Hình 32. ***checkPassword()*** *thông báo trạng thái xác thực cho người dùng*

```
666  void accessDenied() {
667    for (int i = 0; i < 3; i++) {
668      digitalWrite(RedLED, HIGH);
669      tone(Buzzer, 1500);
670      delay(300);
671      digitalWrite(RedLED, LOW);
672      noTone(Buzzer);
673      delay(200);
674    }
675    digitalWrite(BlueLED, HIGH);
676  }
```

Hình 33. ***accessDenied()*** *tạo ra các tín hiệu* ***cảnh báo bằng thị giác và âm thanh*** *để gây chú ý khi có sự xâm nhập không hợp lệ*

4.7. Các hàm quản lý thẻ

```
681 boolean isCardAllowed(String uid) {
682   for (int i = 0; i < cardCount; i++) {
683     if (allowedCards[i] == uid) {
684       return true;
685     }
686   }
687   return false;
688 }
```

*Hình 34. **isCardAllowed()** gikiểm tra UID có hợp lệ hay không*

```
690 void addCard() {
691   if (cardCount >= MAX_CARDS) {
692     lcd.clear();
693     lcd.print(F("Full! Cannot"));
694     lcd.setCursor(0, 1);
695     lcd.print(F("add more cards"));
696     delay(2000);
697     showAdminMenu();
698     return;
699   }
700
701   lcd.clear();
702   lcd.print(F("Scan new card"));
703   lcd.setCursor(0, 1);
704   lcd.print(F("to add..."));
705
706   unsigned long startTime = millis();
707   while (millis() - startTime < 10000) {
708     if (getUID()) {
709       if (isCardAllowed(UIDCard)) {
710         lcd.clear();
711         lcd.print(F("Card exists!"));
712         tone(Buzzer, 1000);
713         delay(500);
714         noTone(Buzzer);
715         delay(1500);
716         showAdminMenu();
717         return;
718       }
719
720       // Thêm thẻ mới
721       allowedCards[cardCount] = UIDCard;
722       cardCount++;
723
724       // Lưu vào EEPROM
```

*Hình 35. **addCard()** giúp người dùng có thể thêm thẻ mới vào*

```
758  void removeCard() {
759    if (cardCount <= 1) {
760      lcd.clear();
761      lcd.print(F("Cannot delete!"));
762      lcd.setCursor(0, 1);
763      lcd.print(F("Need 1+ card"));
764      delay(2000);
765      showAdminMenu();
766      return;
767    }
768
769    lcd.clear();
770    lcd.print(F("Scan card to"));
771    lcd.setCursor(0, 1);
772    lcd.print(F("remove..."));
773
774    unsigned long startTime = millis();
775    while (millis() - startTime < 10000) {
776      if (getUID()) {
777        if (UIDCard == MasterTag) {
778          lcd.clear();
779          lcd.print(F("Cannot delete"));
780          lcd.setCursor(0, 1);
781          lcd.print(F("Admin card!"));
782          tone(Buzzer, 1000);
783          delay(500);
784          noTone(Buzzer);
785          delay(1500);
786          showAdminMenu();
787          return;
788        }
789
790        boolean found = false;
791        for (int i = 0; i < cardCount; i++) {
```

Hình 36. ***removeCard()*** *giúp người dùng có thể xóa bỏ thẻ*

```
849  void changePassword() {
850    lcd.clear();
851    lcd.print(F("New Password:"));
852    lcd.setCursor(0, 1);
853
854    String newPassword = "";
855    unsigned long startTime = millis();
856
857    while (millis() - startTime < 30000) {
858      char key = getKey();
859
860      if (key) {
861        startTime = millis();
862
863        if (key == '#') {
864          if (newPassword.length() > 0) {
865            newPassword.remove(newPassword.length() - 1);
866            lcd.setCursor(0, 1);
867            lcd.print("                ");
868            lcd.setCursor(0, 1);
869            for (unsigned int i = 0; i < newPassword.length(); i++)
870              lcd.print("*");
871          }
872        } else if (key == '*') {
873          if (newPassword.length() >= 4) {
874            password = newPassword;
875            writeStringToEEPROM(EEPROM_PASSWORD, password);
876            sendToExcel("Pass_Changed", password);
877            lcd.clear();
878            lcd.print(F("Completed!"));
879            lcd.setCursor(0, 1);
880            lcd.print(F("Pass changed"));
881
882            tone(Buzzer, 2000);
883            delay(200);
```

Hình 37. ***changePassword()*** *giúp người dùng có thể thay đổi mật khẩu*

```
924  void changeMasterCard() {
925    lcd.clear();
926    lcd.print(F("Scan new"));
927    lcd.setCursor(0, 1);
928    lcd.print(F("Master card..."));
929
930    unsigned long startTime = millis();
931    while (millis() - startTime < 10000) {
932      if (getUID()) {
933        boolean isDuplicate = false;
934        for (int i = 0; i < cardCount; i++) {
935          if (allowedCards[i] == UIDCard && allowedCards[i] != MasterTag) {
936            isDuplicate = true;
937            break;
938          }
939        }
940
941        if (isDuplicate) {
942          lcd.clear();
943          lcd.print(F("Card is already"));
944          lcd.setCursor(0, 1);
945          lcd.print(F("in user list!"));
946          tone(Buzzer, 1000);
947          delay(500);
948          noTone(Buzzer);
949          delay(1500);
950          showAdminMenu();
951          return;
952        }
953
954        // Cập nhật master tag trong danh sách
955        for (int i = 0; i < cardCount; i++) {
956          if (allowedCards[i] == MasterTag) {
957            allowedCards[i] = UIDCard;
```

Hình 38. ***changeMasterCard()*** *cập nhật lại master tag trong danh sách*

4.8. Hàm thực hiện việc đọc RFID

```
1001  boolean getUID() {
1002    if (!mfrc522.PICC_IsNewCardPresent())
1003      return false;
1004    if (!mfrc522.PICC_ReadCardSerial())
1005      return false;
1006
1007    UIDCard = "";
1008    for (byte i = 0; i < mfrc522.uid.size; i++) {
1009      UIDCard.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
1010      UIDCard.concat(String(mfrc522.uid.uidByte[i], HEX));
1011    }
1012    UIDCard.toUpperCase();
1013    UIDCard = UIDCard.substring(1);
1014    mfrc522.PICC_HaltA();
1015    return true;
1016  }
```

*Hình 39. **getUID()** giúp ta lấy được giá trị của các tag*

4.9. Hàm đọc phím từ KEYPAD I2C

```
1021  char getKey() {
1022    if (keypad.isPressed()) {
1023      char key = keypad.getChar();
1024      delay(200);
1025      return key;
1026    }
1027    return 0;
1028  }
```

*Hình 40. **getKey()** đọc tín hiệu từ keypad*

CHƯƠNG 5. KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

5.1. Mục tiêu thực nghiệm

Mục tiêu của quá trình thực nghiệm nhằm:

- Kiểm tra tính đúng đắn và ổn định của hệ thống khóa cửa thông minh.
- Đánh giá khả năng xác thực người dùng thông qua các phương thức: RFID, mật khẩu (Keypad) và điều khiển từ Web.
- Đánh giá khả năng ghi nhận và quản lý lịch sử ra vào trên hệ thống Web.
- Phân tích các ưu điểm, hạn chế và khả năng ứng dụng thực tế của hệ thống.

5.2. Môi trường và điều kiện thực nghiệm

5.2.1. Phần cứng sử dụng

Hệ thống được lắp ráp và thử nghiệm với các thiết bị sau:

- Arduino Uno (bộ điều khiển trung tâm).
- Module RFID RC522 (13.56 MHz).
- Thẻ RFID MIFARE.
- Keypad ma trận 4×4 (giao tiếp I2C).
- LCD 1602 I2C (PCF8574).
- Động cơ Servo SG90 (cơ cấu khóa).
- Module Wi-Fi ESP8266 ESP-01.
- Relay 5V (mô phỏng cơ cấu đóng/ngắt).
- Breadboard và nguồn cấp 5V ổn định.

5.2.2. Phần mềm và nền tảng

- Firmware: Arduino IDE.
- Web quản lý: HTML (kết hợp backend xử lý xác thực và lưu log).
- Giao thức truyền thông: UART (Arduino ↔ ESP), HTTP (ESP ↔ Web Server).

5.3. Kết quả thực nghiệm theo từng chức năng

5.3.1. Chức năng mở cửa bằng thẻ RFID

Quy trình kiểm tra:

- Đưa thẻ RFID hợp lệ vào vùng quét của module RC522.
- Arduino đọc UID và gửi đến Web thông qua ESP8266.
- Web xác thực UID và trả kết quả về hệ thống.

Kết quả:

- Thẻ hợp lệ → Servo quay mở khóa, LCD hiển thị “Access Granted”.
- Thẻ không hợp lệ → Cửa không mở, còi/buzzer cảnh báo, LCD hiển thị “Access Denied”.
- Thời gian phản hồi trung bình: **< 1 giây**.

Nhận xét:

- Nhận dạng thẻ nhanh, ổn định.
- Không xảy ra tình trạng đọc sai UID trong quá trình thử nghiệm.

5.3.2. Chức năng mở cửa bằng mật khẩu (Keypad)

Quy trình kiểm tra:

- Người dùng nhập mật khẩu thông qua Keypad.
- Mật khẩu được che ký tự (*) trên LCD.
- Hệ thống so sánh mật khẩu nhập với mật khẩu lưu trữ.

Kết quả:

- Mật khẩu đúng → Servo mở cửa, ghi log thành công.
- Mật khẩu sai → Báo lỗi, bật LED/buzzer cảnh báo.
- Cho phép nhập lại sau khi sai.

Nhận xét:

- Giao diện LCD trực quan, dễ sử dụng.
- Hệ thống xử lý chính xác, không bị treo khi nhập sai nhiều lần.

5.3.3. Chức năng điều khiển và giám sát qua Web

Quy trình kiểm tra:

- Người dùng truy cập Web (HTML).
- Thực hiện lệnh mở cửa từ xa.
- Theo dõi lịch sử ra vào.

Kết quả:

- Có thể mở cửa từ Web trong phạm vi mạng Wi-Fi.
- Web hiển thị lịch sử gồm:
    - Thời gian
    - Phương thức truy cập (RFID / Password / Web)
    - UID hoặc mật khẩu (hoặc trạng thái Denied)
- Dữ liệu được cập nhật theo thời gian thực.

Nhận xét:

- Web hoạt động ổn định, dễ theo dõi.
- Phù hợp cho mục đích quản lý, giám sát.

5.4. Đánh giá độ ổn định và độ tin cậy

Tiêu chí	**Kết quả**
Độ ổn định hệ thống	Tốt
Thời gian phản hồi	Nhanh (< 1 giây)
Độ chính xác xác thực	Cao
Khả năng mở rộng	Tốt (có thể thêm user, log)
Khả năng sử dụng thực tế	Khả thi

Bảng 8. Kết quả đánh giá hệ thống sau thực nghiệm

Hệ thống hoạt động liên tục trong nhiều chu kỳ thử nghiệm mà không xảy ra lỗi nghiêm trọng.

5.5. Đánh giá tổng thể

Qua quá trình thực nghiệm, hệ thống khóa cửa thông minh đã:

- Đáp ứng đầy đủ các yêu cầu đề ra ban đầu.

- Hoạt động ổn định, chính xác.
- Thể hiện rõ tính ứng dụng trong các mô hình nhà thông minh và hệ thống kiểm soát ra vào quy mô nhỏ.

Kết quả đạt được cho thấy hệ thống hoàn toàn phù hợp cho mục đích học tập, nghiên cứu và có tiềm năng phát triển trong thực tế.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Chương này tổng kết lại những thành tựu đã đạt được của đồ án, đánh giá các điểm mạnh và hạn chế của hệ thống, đồng thời đề xuất những hướng nâng cấp và mở rộng khả năng hoạt động của hệ thống trong tương lai.

6.1. Kết Luận

Đồ án **Hệ thống khóa cửa thông minh sử dụng RFID và Wi-Fi** đã được hiện thực hóa thành công, đạt được các mục tiêu đặt ra ban đầu và cung cấp một giải pháp kiểm soát truy cập hiệu quả và chi phí hợp lý dựa trên nền tảng Vi điều khiển Arduino.

6.1.1. Tóm tắt Kết quả Đạt được

1. **Thiết kế và Triển khai Cấu trúc Vi điều khiển Kép:** Đã sử dụng thành công cấu trúc **Arduino Uno và ESP Module**, trong đó Arduino xử lý tác vụ I/O thời gian thực (RFID, Servo) và ESP chuyên trách giao tiếp Wi-Fi, tối ưu hóa hiệu suất tổng thể của hệ thống.
2. **Hệ thống Xác thực RFID Hoàn chỉnh:** Hệ thống có khả năng đọc và giải mã mã UID từ thẻ RFID 13.56 MHz một cách nhanh chóng và chính xác.
3. **Tích hợp IoT thông qua Google Sheets:** Đã thiết lập thành công cơ chế giao tiếp mạng không cần Server riêng, sử dụng **Google Sheets** làm nền tảng cốt lõi cho:
    - **Xác thực:** Kiểm tra tính hợp lệ của thẻ (UID) dựa trên dữ liệu từ Sheet.
    - **Ghi Log:** Ghi lại nhật ký ra vào (thời gian, trạng thái, thẻ) theo thời gian thực.
    - **Điều khiển Từ xa:** Thiết lập thành công kênh lệnh hai chiều (Web⇔Sheet ⇔ESP ⇔Arduino) cho phép mở cửa từ xa.
4. **Hoàn thành Cơ cấu Chấp hành:** Đã điều khiển thành công Động cơ Servo bằng tín hiệu PWM để thực hiện thao tác đóng/mở khóa cửa vật lý theo logic điều khiển và kết quả xác thực.
5. **Giao diện Giám sát Web:** Đã xây dựng giao diện Web cơ bản cho phép người quản lý giám sát nhật ký ra vào và thực hiện lệnh điều khiển từ xa, nâng cao khả năng quản lý.

6.1.2. Đánh giá Ưu điểm và Hạn chế

Phân loại	Ưu điểm (Strengths)	Hạn chế (Limitations)
Kỹ thuật	Sử dụng linh kiện phổ biến (Arduino, RC522), chi phí thấp, dễ dàng bảo trì. Phân tách tác vụ rõ ràng giữa Arduino và ESP.	Tốc độ kết nối và polling lệnh (trên ESP) bị giới hạn bởi tốc độ của Google Apps Script và mạng Wi-Fi.

Bảo mật	Mã UID được xác thực tập trung trên Web (Google Sheets), dễ dàng vô hiệu hóa thẻ đã mất ngay lập tức.	Quá trình truyền UID và lệnh qua HTTP có thể không được mã hóa hoàn toàn (tùy thuộc vào cấu hình GAS), cần SSL/HTTPS cho bảo mật cao hơn.
Khả năng Mở rộng	Cơ chế lưu trữ tập trung trên Google Sheets cho phép mở rộng số lượng người dùng dễ dàng mà không giới hạn bộ nhớ trên Arduino.	Phụ thuộc vào dịch vụ Google Sheets; có thể có giới hạn về số lượng yêu cầu truy cập/ngày hoặc độ trễ truy cập (Latency).
Hoạt động	Có thể điều khiển cửa từ xa thông qua Web và theo dõi log bất kỳ lúc nào có kết nối Internet.	Servo chỉ phù hợp cho khóa cửa mẫu hoặc khóa nhẹ; không đủ lực cho các loại cửa an ninh cao.

Bảng 9. Phân tích ưu – nhược điểm của hệ thống khóa cửa RFID

6.2. Hướng Phát triển và Nâng cấp Hệ thống trong Tương lai

Để nâng cấp hệ thống thành một sản phẩm thương mại hoặc một giải pháp kiểm soát truy cập mạnh mẽ hơn, cần tập trung vào các hướng phát triển sau:

6.2.1. Nâng cấp Phần cứng và Cơ cấu Chấp hành

- **Thay thế Vi điều khiển:** Thay thế cấu trúc Arduino Uno + ESP Module bằng một vi điều khiển tích hợp mạnh mẽ hơn như **ESP32**. ESP32 có lõi kép, Wi-Fi và Bluetooth tích hợp, giúp xử lý các tác vụ I/O và mạng đồng thời hiệu quả hơn, loại bỏ giao tiếp Serial giữa hai board.
- **Cơ cấu Khóa An toàn:** Thay thế Động cơ Servo bằng **Khóa điện từ (Solenoid Lock)** hoặc **Khóa cửa motor điện DC** công suất lớn hơn. Điều này yêu cầu bổ sung **Module Relay** để điều khiển nguồn điện cao áp/cao dòng cho khóa.
- **Bổ sung Pin Dự phòng (UPS):** Tích hợp mạch sạc và pin Lithium-ion để hệ thống có thể hoạt động trong trường hợp mất điện lưới, đảm bảo tính liên tục của an ninh.

6.2.2. Cải thiện Giao tiếp và Bảo mật Mạng

- **Chuyển sang Giao thức IoT Chuyên dụng (MQTT):** Thay vì sử dụng cơ chế Polling (đọc liên tục) từ Google Sheets, chuyển sang giao thức **MQTT (Message Queuing Telemetry Transport)**. MQTT sử dụng mô hình Publish/Subscribe (Xuất bản/Đăng ký) giúp việc truyền lệnh từ Web đến ESP trở nên tức thời (Real-time) và hiệu quả năng lượng hơn.
- **Sử dụng API Server Chuyên nghiệp:** Chuyển từ Google Sheets sang Web Server tự quản lý (VPS/Cloud) với Database (MySQL/MongoDB) và API được mã hóa **HTTPS/SSL** để đảm bảo bảo mật dữ liệu đầu cuối.

- **Mã hóa Dữ liệu Truyền:** Thực hiện mã hóa (ví dụ: AES) cho các lệnh và dữ liệu UID quan trọng được truyền giữa ESP và Server.

6.2.3. Mở rộng Chức năng Kiểm soát Truy cập

- **Giao tiếp Bluetooth (BLE):** Tích hợp giao tiếp Bluetooth (sử dụng ESP32) để cho phép người dùng mở khóa bằng điện thoại thông minh qua ứng dụng di động thay vì chỉ dùng thẻ vật lý.
- **Quản lý Thời gian:** Triển khai logic kiểm soát truy cập theo thời gian (Time-based Access Control): chỉ cho phép một số thẻ truy cập vào các khoảng thời gian cụ thể trong ngày hoặc trong tuần.
- **Đa Phương thức Xác thực:** Kết hợp với các Module xác thực khác (ví dụ: Module vân tay – Fingerprint Sensor) để nâng cao tính bảo mật (yêu cầu Thẻ **và** Vân tay).
- **Thông báo Tức thì:** Tích hợp dịch vụ thông báo (ví dụ: Telegram, Email) để gửi cảnh báo ngay lập tức cho người quản lý khi có thẻ không hợp lệ hoặc khi cửa được mở từ xa.

6.3. Tài liệu tham khảo

- Arduino Official Documentation – Arduino Language Reference https://www.arduino.cc/reference/en/
- Arduino UNO Technical Specifications https://docs.arduino.cc/hardware/uno-rev3
- MFRC522 RFID Module Datasheet https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
- LiquidCrystal I2C Library Documentation https://github.com/johnrickman/LiquidCrystal_I2C
- ESP8266 AT Command Set Documentation https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf
- Arduino EEPROM Library Reference https://www.arduino.cc/reference/en/libraries/eeprom/
- HTTP Protocol – RFC 2616 https://www.rfc-editor.org/rfc/rfc2616
- PushingBox API Documentation https://www.pushingbox.com/api.php
- Smart Door Lock System using Arduino – ResearchGate https://www.researchgate.net/publication/339867421
- IoT-Based Smart Door Lock System – IEEE https://ieeexplore.ieee.org/document/8459704