

Đại học Quốc gia Thành phố Hồ Chí Minh  
Trường Đại học Bách Khoa  
Khoa Khoa học và Kỹ thuật Máy tính



# BÁO CÁO ĐỒ ÁN THIẾT KẾ LUẬN LÝ (CO3091)

Đề bài:

Xây dựng hệ thống giao tiếp, điều khiển  
module RFID, LED RGB và WiFi ESP32  
sử dụng vi điều khiển STM32

Nhóm 09 – HK241  
Giảng viên hướng dẫn: Nguyễn Thành Lộc

Sinh viên thực hiện	MSSV	Diểm số
Lê Nguyễn Phúc Thịnh	2213279	
Nguyễn Kim Thuận	2213357	
Phan Huy Trung	2213709	

Tp.Hồ Chí Minh, Tháng 12 / 2024



# Mục lục

<b>1</b>	<b>Phần mở đầu</b>	<b>3</b>
<b>2</b>	<b>Giới thiệu</b>	<b>4</b>
2.1	Yêu cầu và chức năng của hệ thống . . . . .	4
2.2	Công cụ hiện thực hệ thống . . . . .	5
2.2.1	Phần cứng và thông số kỹ thuật . . . . .	5
2.2.2	Phần mềm . . . . .	14
<b>3</b>	<b>Thiết kế mạch</b>	<b>16</b>
3.1	Tổng quát và mô hình hóa các khối module . . . . .	16
3.1.1	Tổng quát về các chuẩn giao tiếp . . . . .	16
3.1.2	Tổng quát hóa các khối module . . . . .	18
3.1.3	Mô hình hóa các khối module . . . . .	18
3.2	Vai trò và Chức năng các khối module . . . . .	19
3.2.1	Module vi điều khiển STM32 . . . . .	19
3.2.2	Module vi điều khiển WiFi ESP32 . . . . .	19
3.2.3	Module RFID RC522 . . . . .	20
3.2.4	Module LED RGB . . . . .	20
3.2.5	Module Buzzer . . . . .	21
3.2.6	Module LCD 1602 . . . . .	21
3.3	Máy trạng thái của hệ thống . . . . .	21
3.4	Hiện thực source code . . . . .	22
3.4.1	Hiện thực cho vi điều khiển STM32 . . . . .	22
3.4.2	Hiện thực cho vi điều khiển ESP32 . . . . .	32
3.4.3	Phân tích độ phức tạp thuật toán . . . . .	36
3.5	Mạch kết nối các module . . . . .	38
<b>4</b>	<b>Tổng kết</b>	<b>39</b>
4.1	Kết luận . . . . .	39
4.2	Đánh giá và phương hướng phát triển hệ thống . . . . .	40
4.2.1	Đánh giá và nhận xét . . . . .	40
4.2.2	Phương hướng phát triển hệ thống . . . . .	40
4.3	Kết quả đánh giá . . . . .	42
<b>5</b>	<b>Tài liệu tham khảo</b>	<b>43</b>



# 1 Phần mở đầu

Trong thời đại hiện nay, thế giới đang chứng kiến một sự phát triển vượt bậc của công nghệ, đặc biệt là trong các lĩnh vực điện tử, Internet vạn vật (IoT), và trí tuệ nhân tạo (AI). Những tiến bộ này không chỉ cải thiện chất lượng cuộc sống mà còn thúc đẩy các ngành công nghiệp, y tế, giáo dục, và giao thông vận tải phát triển với tốc độ chưa từng có.

Việc ứng dụng các thiết bị điện tử thông minh đã trở thành xu thế tất yếu, từ các thiết bị gia dụng như khóa cửa thông minh, đèn tự động, cho đến các hệ thống quản lý lớn như kiểm soát ra vào tại các tòa nhà, trường học, và bệnh viện. Trong bối cảnh đó, nhu cầu về hệ thống giao tiếp giữa các module phần cứng để xử lý và truyền tải thông tin một cách chính xác và hiệu quả ngày càng trở nên quan trọng.

Hệ thống mà nhóm báo cáo giới thiệu bao gồm các module RFID RC522, ESP32, và STM32 là một giải pháp tiêu biểu nhằm đáp ứng những yêu cầu trên. Với khả năng quét thẻ RFID để nhận diện đối tượng, truyền thông tin đến ESP32, đèn LED RGB để báo hiệu tình trạng quét thẻ, và cuối cùng gửi dữ liệu lên Web Server, hệ thống không chỉ cung cấp khả năng quản lý và giám sát thời gian thực mà còn mang lại sự tiện lợi cho người quản lý và an toàn cho người sử dụng. Ngoài ra, trong hệ thống của nhóm báo cáo còn có những module bổ sung như màn hình LCD 16x2, và loa buzzer nhằm cải thiện và tối ưu hóa trải nghiệm người dùng.

Những đặc điểm này giúp hệ thống trở thành một công cụ hữu ích trong nhiều lĩnh vực:

- **Quản lý an ninh:** Kiểm soát ra vào tại các khu vực quan trọng.
- **Theo dõi và quản lý tài sản:** Ứng dụng trong các kho hàng, nhà máy.
- **Tích hợp IoT:** Dữ liệu từ hệ thống có thể được kết nối với các hệ sinh thái thông minh để phân tích hoặc ra quyết định tự động.

Việc xây dựng hệ thống không chỉ góp phần vào việc tối ưu hóa quy trình quản lý mà còn mở ra tiềm năng phát triển các giải pháp tích hợp dựa trên công nghệ RFID và IoT. Đây là nền tảng vững chắc để hướng tới một xã hội hiện đại, nơi công nghệ hỗ trợ tối đa cho nhu cầu của con người, từ việc giảm bớt công việc thủ công đến nâng cao tính chính xác và bảo mật.

Hệ thống này không chỉ dừng lại ở việc cung cấp một giải pháp cụ thể mà còn đóng vai trò như một bước tiến trong việc nghiên cứu và phát triển các ứng dụng công nghệ cao, góp phần thúc đẩy xu hướng số hóa và tự động hóa trong đời sống hiện đại.

## 2 Giới thiệu

### 2.1 Yêu cầu và chức năng của hệ thống

Hệ thống được thiết kế nhằm đáp ứng các yêu cầu và tiêu chí sau:

- **Tính chính xác cao trong nhận diện và xử lý dữ liệu:** Hệ thống cần đảm bảo khả năng nhận diện chính xác thẻ RFID, truyền tải dữ liệu (qua các chuẩn giao tiếp) phải không xảy ra sai sót qua các module, và hiển thị thông tin chính xác trên các thiết bị đầu cuối.
- **Khả năng hoạt động thời gian thực:** Dữ liệu quét thẻ và trạng thái của hệ thống cần được cập nhật tức thì, đảm bảo người dùng có thể theo dõi và quản lý nhanh chóng.
- **Tính bảo mật:** Dữ liệu người dùng và quá trình xử lý thông tin phải được bảo vệ chặt chẽ để ngăn chặn truy cập trái phép.
- **Dễ dàng sử dụng và tích hợp:** Hệ thống phải thân thiện với người dùng, dễ dàng cài đặt, vận hành, và tích hợp với các hệ thống quản lý khác hoặc nền tảng IoT.
- **Độ bền và tính ổn định:** Các module phần cứng cần đảm bảo độ bền trong quá trình sử dụng lâu dài, đồng thời hệ thống phải vận hành ổn định trong các điều kiện hoạt động khác nhau.
- **Mở rộng và bảo trì:** Cần thiết kế một cách có hệ thống để có thể dễ dàng thêm các tính năng, các module trong tương lai nhằm nâng cao trải nghiệm người dùng, hoặc chỉnh sửa, bảo trì hệ thống mà không ảnh hưởng đến cấu trúc hiện tại.

Hệ thống thực hiện các chức năng chính như sau:

- **Nhận diện đối tượng thông qua RFID RC522:** Hệ thống sử dụng module RFID để quét và xác nhận danh tính của người dùng hoặc đối tượng.
- **Xử lý dữ liệu và truyền thông tin giữa các module:** STM32 và ESP32 đóng vai trò trung gian, xử lý thông tin từ RFID RC522 và các module khác, sau đó gửi dữ liệu đến Web Server.
- **Cảnh báo và hiển thị trạng thái:** LED RGB hiển thị trạng thái hệ thống như quét thành công, thất bại hoặc trạng thái đang chờ thẻ. Trong khi Loa Buzzer có nhiệm vụ phát tín hiệu âm thanh khi quét thẻ hoặc xảy ra lỗi, tích hợp thêm màn hình LCD tên người dùng hoặc không tìm thấy dữ liệu người dùng, giúp cải thiện và tối ưu trải nghiệm người dùng.
- **Kết nối và quản lý dữ liệu từ xa:** Hệ thống sử dụng module WiFi ESP32 để truyền dữ liệu lên Web Server, cho phép quản lý từ xa và tích hợp với các hệ thống IoT.
- **Tích hợp IoT và mở rộng:** Hệ thống cung cấp khả năng thu thập và lưu trữ dữ liệu trên các nền tảng thông minh để phục vụ phân tích hoặc tự động hóa quy trình.

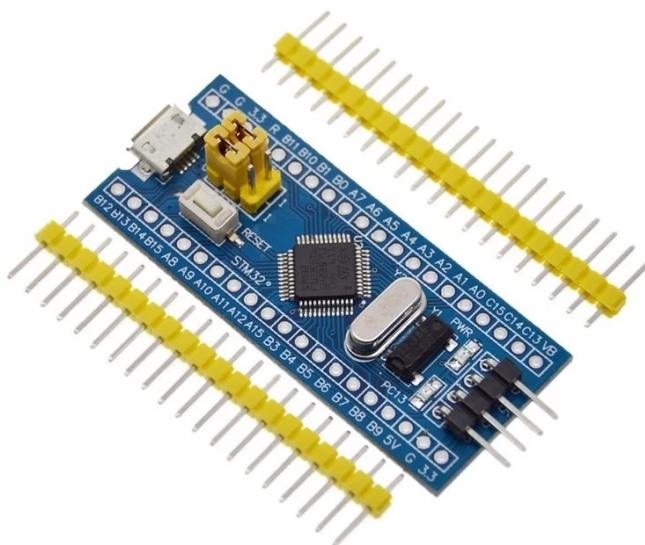
## 2.2 Công cụ hiện thực hệ thống

### 2.2.1 Phần cứng và thông số kỹ thuật

Hệ thống sử dụng một chip vi điều khiển STM32F103C8T6 và ESP32 WROOM 32 (module WiFi ESP32) để giao tiếp với các module chính như RFID RC522 (module đọc thẻ RFID), Led SMD RGB (module LED RGB) và các module khác như MKE-M03 buzzer module (module buzzer) và LCD text LCD1602 (module màn hình LCD).

Bây giờ cùng phân tích thông số kỹ thuật của từng linh kiện được sử dụng trong hệ thống.

#### 2.2.1.1 Module Vi điều khiển STM32F103C8T6



Hình 1: Kit phát triển STM32F103C8T6

#### Thông số kỹ thuật:

- Vi điều khiển: STM32F103C8T6.
- Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vi điều khiển chính.
- Tích hợp sẵn thạch anh 8Mhz và 32Khz cho các ứng dụng RTC.
- Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,...
- Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.
- Kích thước: 53.34 x 15.24mm.

### 2.2.1.2 Module WiFi ESP32 WROOM 32



Hình 2: Kit RF thu phát wifi bluetooth ESP32 Type C

#### Thông số kỹ thuật:

- Loại: Wifi + Bluetooth Module
- Cổng nạp: Type C || Micro (tùy chọn trong phần phân loại)
- Mô hình: ESP32 38 chân
- Điện áp nguồn (USB): 5V DC
- Đầu vào/Đầu ra điện áp: 3.3V DC
- Công suất tiêu thụ:  $5\mu\text{A}$  trong hệ thống treo chế độ
- Hiệu suất: lên đến 600 DMIPS
- Tần số: lên đến 240MHz
- Wifi: 802.11 B/g/n/E/I (802.11N @ 2.4 GHz lên đến 150 Mbit/s)
- Bluetooth: 4.2 BR/EDR BLE 2 chế độ điều khiển
- Bộ nhớ: 448 Kbyte ROM, 520 Kbyte SRAM, 6 Kbyte SRAM trên RTC và QSPI Hỗ trợ đèn flash / SRAM chip
- Chip USB-Serial: CP2102

- Ăng ten: PCB
- GPIO kỹ thuật số: 24 chân (một số chân chỉ làm đầu vào)
- Kỹ thuật số Analog: 12bit SAR loại ADC, hỗ trợ các phép đo trên lên đến 18 kênh, một số chân hỗ trợ một bộ khuếch đại với lập trình tăng
- Bảo mật: IEEE 802.11, bao gồm cả WFA, WPA/WPA2 và WAPI
- Phần cứng tăng tốc mật mã học: AES, SHA-2, RSA, hình elip mật mã Đường Cong (ECC), số ngẫu nhiên Máy phát điện (RNG)
- Cân nặng: 11g

#### 2.2.1.3 Module RFID - RC522



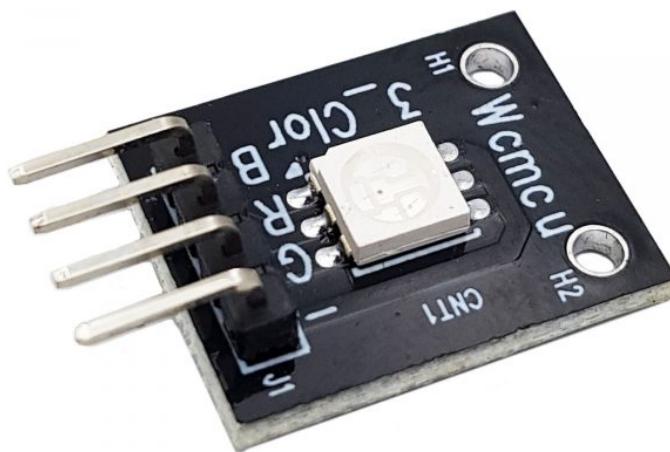
Hình 3: Mạch RFID NFC 13.56MHz RC522

#### Thông số kỹ thuật:

- Nguồn sử dụng: 3.3 VDC
- Dòng điện: 13 - 26 mA
- Tần số hoạt động: 13.56 Mhz
- Khoảng cách hoạt động: 0 - 60 mm
- Chuẩn giao tiếp: SPI
- Tốc độ truyền dữ liệu: tối đa 10 Mbit/s

- Các loại card RFID hỗ trợ: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire
- Kích thước:  $40 \times 60$  (mm)

#### 2.2.1.4 Module Led RGB

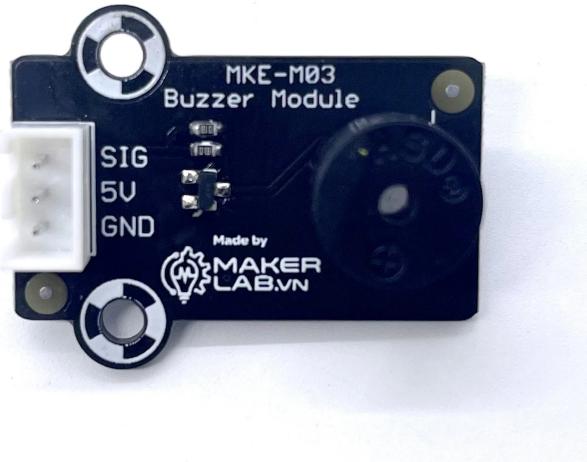


Hình 4: Module Led SMD RGB KY-009

#### Thông số kỹ thuật:

- Dương chung (anode)
- Điện áp led đỏ: từ 1.8 V đến 2.4 V
- Điện áp led xanh lá: từ 2.8 V đến 3.6 V
- Điện áp xanh dương: từ 2.8 V đến 3.6 V
- Dải dòng: từ 20 mA đến 30 mA
- Nhiệt độ làm việc: từ -25°C đến 85°C
- Kích thước PCB:  $15.5 \times 19 \times 3$  (mm)
- Lỗ bắt vít M2: 10 mm

### 2.2.1.5 Module còi buzzer



Hình 5: Mạch còi báo MKE-M03 buzzer module

#### Thông số kỹ thuật:

- Điện áp hoạt động: 5 VDC
- Chuẩn giao tiếp: Digital
- Điện áp giao tiếp: TTL 3.3 VDC / 5 VDC
- Sử dụng trực tiếp an toàn với các board mạch giao tiếp ở cả hai mức điện áp 3.3VDC và 5VDC như: Arduino, Raspberry Pi, Jetson Nano, Micro:bit,....
- Bổ sung thêm các thiết kế ổn định, chống nhiễu.
- Chuẩn kết nối: Connector XH2.54 3 Pins

### 2.2.1.6 Module màn hình LCD

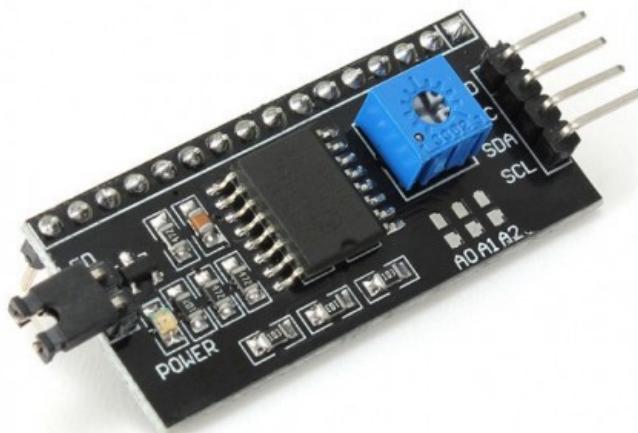
Với module màn hình LCD, nhóm báo cáo dùng màn hình LCD 1602 tích hợp thêm mạch chuyển giao tiếp LCD 1602 sang I2C để dễ dàng trong việc hiện thực giao tiếp với vi điều khiển STM32.



Hình 6: Màn hình LCD text LCD1602 xanh lá

### Thông số kỹ thuật:

- Điện áp hoạt động là 5 V.
- Kích thước: 80 x 36 x 12.5 (mm)
- Chữ đen, nền xanh lá
- Khoảng cách giữa hai chân kết nối là 0.1 inch tiện dụng khi kết nối với Breadboard.
- Tên các chân được ghi ở mặt sau của màn hình LCD hỗ trợ việc kết nối, đi dây điện.
- Có đèn led nền, có thể dùng biến trở hoặc PWM điều chỉnh độ sáng để sử dụng ít điện năng hơn.
- Có thể được điều khiển với 6 dây tín hiệu



Hình 7: Mạch chuyển giao tiếp LCD1602, LCD1604, LCD2004 sang I2C

## Thông số kỹ thuật:

- Sử dụng các loại LCD có driver là HD44780 (LCD 1602, LCD 2004,...)
  - Chỉ cần 2 chân (SDA và SCL) của MCU kết nối với 2 chân (SDA và SCL) của module là đã có thể hiển thị thông tin lên LCD.
  - Ngoài ra có thể điều chỉnh được độ tương phản bởi biến trở gắn trên module.

#### 2.2.1.7 Bộ cấp nguồn cho hệ thống

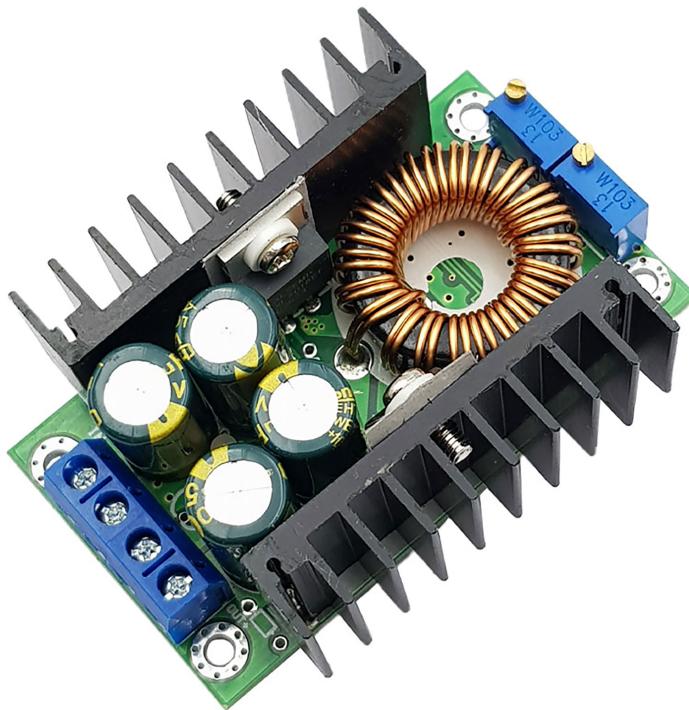
Bộ cấp nguồn mà nhóm báo cáo sử dụng bao gồm bộ nguồn tổ ong 12V 5A và một mạch giảm áp DC-DC Buck từ đó chuyển dòng điện xoay chiều 220V sang dòng một chiều 3,3V để cấp nguồn cho hệ thống.



Hình 8: Nguồn tản nhiệt 12V 5A

#### Thông số kỹ thuật:

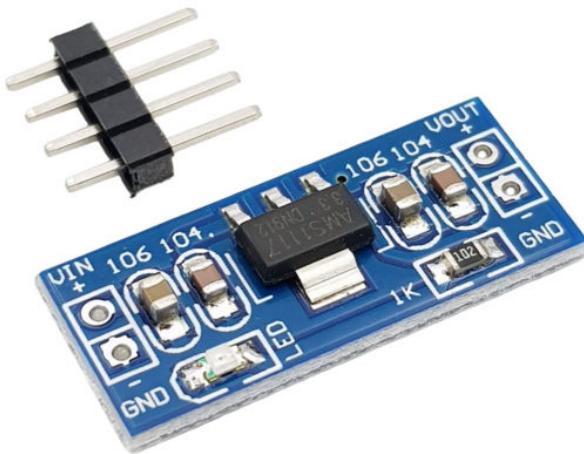
- Điện áp đầu vào: 180V - 240V
- Tần số hoạt động: 47Hz - 63Hz
- Công suất: 60W
- Điện áp đầu ra: 12V
- Dòng điện tối đa: 5A
- Điện áp điều chỉnh:  $\pm 10\%$
- Hiệu suất  $\geq 85\%$
- Điều chỉnh điện áp (Đầy tải)  $\leq 0.3\%$
- Bảo vệ quá áp 105% - 150% điện áp định mức
- Nhiệt độ làm việc:  $-20^{\circ}\text{C} \rightarrow 60^{\circ}\text{C}$
- Nhiệt độ bảo quản  $-40^{\circ}\text{C} \rightarrow 85^{\circ}\text{C}$
- Kích thước: 110 x 78 x 36 (mm)



Hình 9: Mạch giảm áp DC-DC Buck 12A

#### Thông số kỹ thuật:

- Điện áp đầu vào: 7 - 32VDC
- Điện áp đầu ra: 0.8 - 28VDC
- Dòng điện đầu ra: 0.2 - 12A
- Công suất ngõ ra: 100W (có thể lên tới 200W ở điều khiển tản nhiệt tốt)
- Tần số hoạt động: 300KHz
- Hiệu suất chuyển đổi: 95%
- Tích hợp LED báo hiệu quá dòng
- Kích thước: 60 x 51 x 22 (mm)



Hình 10: Mạch giảm áp 3.3V

#### Thông số kỹ thuật:

- Điện áp vào: DC 4.5V – 7V (Điện áp vào tối thiểu phải cao hơn điện áp ra 1V).
- Điện áp ra: 3.3V, 800mA (Dòng điện thực tế có thể vượt hơn 800mA).
- Kích thước bảng mạch: 2,5 cm \* 1.1cm.

#### 2.2.2 Phần mềm

Trong hệ thống, nhóm báo cáo sử dụng hai phần mềm chính để lập trình và hiện thực các module: **STM32CubeIDE phiên bản 1.7.0** cho STM32F103C8T6 và **Arduino IDE phiên bản 4.3.2** cho ESP32-S3. Hai phần mềm này được lựa chọn dựa trên khả năng tương thích cao với phần cứng, môi trường phát triển mạnh mẽ và nhiều ưu điểm hỗ trợ lập trình viên.



(a) STM32CubeIDE



(b) Arduino IDE

Hình 11: Phần mềm lập trình cho hệ thống

### 2.2.2.1 STM32CubeIDE

#### Ưu điểm

- Tích hợp đầy đủ công cụ phát triển: STM32CubeIDE là môi trường phát triển tích hợp (IDE) mạnh mẽ của STMicroelectronics, cung cấp các công cụ lập trình, biên dịch, và debug trực tiếp trên một nền tảng duy nhất.
- Khả năng hỗ trợ các dòng vi điều khiển STM32 vượt trội: Phần mềm tối ưu cho tất cả dòng vi điều khiển STM32, đặc biệt với chip STM32F103C8T6. Nó tự động cấu hình các chân GPIO, tích hợp cấu hình các bộ ngoại vi như UART, I2C, SPI,...
- Giao diện trực quan và dễ sử dụng: Công cụ cấu hình CubeMX đi kèm cho phép người dùng thiết kế sơ đồ kết nối pin và các ngoại vi một cách trực quan, tiết kiệm thời gian trong việc thiết lập phần cứng.
- Hỗ trợ thư viện HAL (Hardware Abstraction Layer): Các thư viện HAL được tích hợp giúp việc lập trình trở nên đơn giản hơn, giảm tải việc viết mã trực tiếp trên thanh ghi (registers).
- Debug mạnh mẽ: Hỗ trợ nhiều công cụ debug như SWD, JTAG, và các bộ mô phỏng, giúp phát hiện lỗi dễ dàng trong quá trình phát triển.
- Cộng đồng hỗ trợ rộng lớn: Là sản phẩm chính thức của STMicroelectronics, STM32CubeIDE có tài liệu phong phú và cộng đồng đông đảo, giúp giải quyết các vấn đề nhanh chóng.

#### Lý do chọn phần mềm

**STM32CubeIDE phiên bản 1.7.0** được nhóm báo cáo lựa chọn vì khả năng tối ưu hóa phát triển trên nền tảng STM32. Đây là phần mềm lý tưởng cho việc hiện thực hóa các tính năng phức tạp của STM32F103C8T6, bao gồm việc giao tiếp với các module LCD 1602, keypad 4x4 và buzzer, xử lý dữ liệu, và truyền tải thông tin sang ESP32-S3. Khả năng debug mạnh mẽ giúp đảm bảo độ ổn định của hệ thống trong quá trình phát triển.

### 2.2.2.2 Arduino IDE

#### Ưu điểm

- Dễ sử dụng và phổ biến: Arduino IDE là môi trường lập trình đơn giản, phù hợp cả với người mới bắt đầu và chuyên gia. Giao diện thân thiện và cú pháp lập trình đơn giản giúp tiết kiệm thời gian học tập và phát triển.
- Hỗ trợ đa nền tảng: Arduino IDE hỗ trợ nhiều loại vi điều khiển, đặc biệt với ESP32-S3 nhờ tích hợp thư viện riêng của Espressif.
- Thư viện phong phú: Hệ sinh thái Arduino IDE cung cấp hàng ngàn thư viện sẵn có, giúp lập trình viên dễ dàng tích hợp các giao thức giao tiếp (WiFi, Bluetooth, MQTT, v.v.) và tính năng đặc thù của ESP32.
- Tích hợp chức năng upload code qua giao thức USB: Việc tải chương trình xuống ESP32S3 đơn giản và nhanh chóng thông qua kết nối USB.
- Cộng đồng rộng lớn: Với hàng triệu người dùng toàn cầu, Arduino IDE có tài liệu và diễn đàn hỗ trợ phong phú, dễ dàng tìm kiếm giải pháp cho các vấn đề kỹ thuật.
- Khả năng mở rộng: Hỗ trợ cài đặt các plugin hoặc gói mở rộng cho ESP32, giúp tích hợp thêm các tính năng nâng cao mà không cần đổi sang môi trường khác.

#### Lý do chọn phần mềm

**Arduino IDE phiên bản 4.3.2** được nhóm báo cáo lựa chọn cho ESP32-S3 vì tính tiện lợi và hỗ trợ mạnh mẽ cho các ứng dụng IoT. Với các thư viện WiFi và Web Server tích hợp sẵn, Arduino IDE giúp hiện thực hóa chức năng kết nối mạng và truyền dữ liệu từ ESP32-S3 lên Web Server một cách nhanh chóng. Điều này rất phù hợp với yêu cầu giao tiếp thời gian thực và tính linh hoạt của hệ thống.

## 3 Thiết kế mạch

### 3.1 Tổng quát và mô hình hóa các khối module

#### 3.1.1 Tổng quát về các chuẩn giao tiếp

Trong quá trình hiện thực hệ thống, việc truyền và nhận tín hiệu giữa các khối module là yếu tố cốt lõi đảm bảo hệ thống hoạt động chính xác và hiệu quả. Do đó, các giao thức truyền thông nối tiếp (Serial Communication) đóng vai trò không thể thiếu. Trong khuôn khổ hệ thống này, nhóm sử dụng ba giao thức chính: **UART**, **SPI**, và **I2C**. Mỗi giao thức được chọn dựa trên đặc điểm kỹ thuật và vai trò phù hợp của các module liên quan.



### 3.1.1.1 Giao thức UART

UART (Universal Asynchronous Receiver/Transmitter) là một trong những giao thức nối tiếp lâu đời và đơn giản nhất, được thiết kế để trao đổi dữ liệu giữa hai thiết bị. UART hoạt động dựa trên việc truyền dữ liệu không đồng bộ, chỉ sử dụng hai dây chính là:

- TX (Transmit): Dây truyền dữ liệu.
- RX (Receive): Dây nhận dữ liệu.

Dữ liệu được truyền qua UART với các bit khởi đầu, bit dừng, và tùy chọn các bit chẵn lẻ để đảm bảo tính toàn vẹn. Điểm đặc biệt của giao thức này là không yêu cầu tín hiệu đồng hồ dùng chung; thay vào đó, các thiết bị phải thống nhất tốc độ truyền dữ liệu và định dạng trước khi giao tiếp.

Với sự đơn giản và đáng tin cậy, UART được nhóm sử dụng để truyền và nhận tín hiệu giữa **ESP32** và **STM32**, đảm bảo thông tin từ thẻ RFID được xử lý và phản hồi chính xác.

### 3.1.1.2 Giao thức SPI

SPI (Serial Peripheral Interface) là một giao thức truyền thông tốc độ cao, lý tưởng cho các hệ thống nhúng có yêu cầu trao đổi dữ liệu nhanh và liên tục. Giao thức này hoạt động theo kiến trúc master-slave, cho phép một thiết bị chính (master) giao tiếp với một hoặc nhiều thiết bị phụ (slave). SPI hỗ trợ giao tiếp song công hoàn toàn (full-duplex), cho phép cả hai thiết bị truyền và nhận dữ liệu đồng thời.

SPI không có giao thức cứng nhắc, mang lại sự linh hoạt cao trong việc cấu hình tốc độ truyền, số lượng bit, và tín hiệu điều khiển. Điều này làm cho SPI trở thành lựa chọn lý tưởng trong các ứng dụng cần băng thông cao hoặc yêu cầu dữ liệu theo thời gian thực.

Trong hệ thống, nhóm sử dụng giao thức SPI để kết nối **RC522** với **ESP32**, nhằm đảm bảo dữ liệu từ thẻ RFID được truyền nhanh chóng và chính xác đến bộ xử lý trung tâm.

### 3.1.1.3 Giao thức I2C

I2C (Inter-Integrated Circuit) là một giao thức nối tiếp đồng bộ, được thiết kế để kết nối nhiều thiết bị trên cùng một bus giao tiếp hai dây:

- SCL (Serial Clock Line): Dây đồng hồ nối tiếp.
- SDA (Serial Data Line): Dây dữ liệu nối tiếp.

I2C hỗ trợ kiến trúc đa chủ/đa nô lệ, mang lại tính linh hoạt cao trong việc kết nối nhiều thiết bị trên cùng một hệ thống. Tuy nhiên, giao thức này có nhược điểm là tốc độ thấp hơn SPI và yêu cầu các điện trỏ kéo lên để duy trì tín hiệu trên bus.

Trong hệ thống, I2C được nhóm sử dụng để giao tiếp giữa **STM32** và **LCD 1602**, đảm bảo việc hiển thị thông tin người dùng một cách rõ ràng và chính xác. Đây là sự lựa chọn phù hợp nhờ tính mở rộng cao và khả năng quản lý hiệu quả nhiều thiết bị.

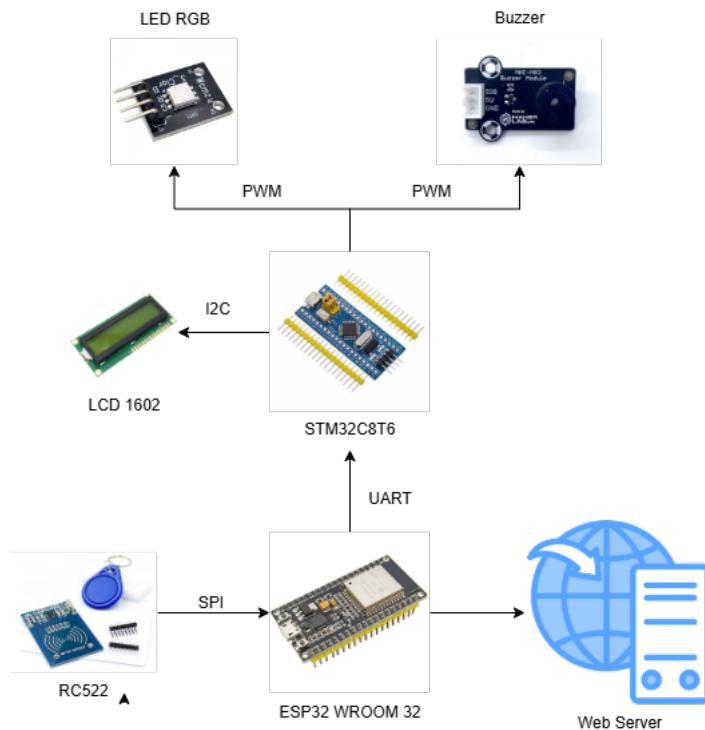
### 3.1.2 Tổng quát hóa các khối module

Hệ thống được thiết kế để nhận diện và quản lý thông tin thẻ RFID một cách hiệu quả, tích hợp các module phần cứng nhằm đảm bảo tính chính xác, tiện lợi và khả năng phản hồi nhanh. Các khối module chính trong hệ thống được kết nối và phối hợp hoạt động như sau:

- RFID RC522: Là thành phần đầu vào, chịu trách nhiệm quét và thu thập thông tin ID từ thẻ RFID. Dữ liệu thẻ sẽ được gửi đến ESP32 để xử lý.
- ESP32: Giao tiếp với RFID RC522 để nhận thông tin thẻ và truyền dữ liệu lên Web Server. Đồng thời, ESP32 gửi tín hiệu xác nhận (qua UART) đến STM32, thông báo trạng thái thẻ (hợp lệ hoặc không hợp lệ).
- STM32: Đóng vai trò là bộ điều khiển trung tâm, quản lý các module khác như LCD 1602, LED RGB, và Buzzer. STM32 điều khiển hiển thị thông tin, trạng thái và phản hồi từ hệ thống, dựa trên tín hiệu nhận được từ ESP32.
- LCD 1602: Hiển thị các thông điệp liên quan, như “Welcome <Tên người dùng>”, “Unknown User”.
- LED RGB: Thể hiện trạng thái hoạt động của hệ thống thông qua các màu sắc.
- Buzzer: Phát âm báo để thông báo trạng thái, giúp người dùng nhận biết khi thao tác thành công hoặc thất bại.

### 3.1.3 Mô hình hóa các khối module

Dựa trên phân tóm quát hóa các khối module trên, ta dễ dàng mô hình hóa, trực quan hóa bằng sơ đồ khát quát giữa các khối module của hệ thống như sau



Hình 12: Sơ đồ khái quát giữa các khối module

### 3.2 Vai trò và Chức năng các khối module

#### 3.2.1 Module vi điều khiển STM32

Vi điều khiển STM32 đóng vai trò trung tâm điều khiển các thiết bị ngoại vi trong hệ thống. Với hiệu năng mạnh mẽ, tốc độ xử lý cao và khả năng hỗ trợ nhiều chuẩn giao tiếp (UART, GPIO, I<sup>2</sup>C), STM32 đảm bảo thực hiện các nhiệm vụ sau:

- Quản lý các thiết bị ngoại vi: STM32 điều khiển để xử lý đầu vào và hiển thị lên màn hình LCD 1602.
- Tích hợp tín hiệu từ ESP32: Khi nhận dữ liệu từ ESP32 qua giao thức UART, STM32 thực hiện các phản hồi tương ứng bằng cách thay đổi trạng thái LED RGB, hiển thị thông báo trên màn hình LCD, và kích hoạt module Buzzer.
- Phản hồi trạng thái hệ thống: STM32 đóng vai trò cung cấp tín hiệu tức thời cho các thiết bị hiển thị hoặc cảnh báo khi cần (như đổi màu LED RGB hoặc phát âm báo từ Buzzer).

Nhờ khả năng linh hoạt trong lập trình và hỗ trợ đa dạng giao tiếp, STM32 là cốt lõi xử lý dữ liệu và điều phối hoạt động của toàn bộ hệ thống.

#### 3.2.2 Module vi điều khiển WiFi ESP32

ESP32 là bộ xử lý chính cho việc kết nối không dây của hệ thống. Được trang bị Wi-Fi tích hợp, ESP32 đảm nhiệm các vai trò sau:

- Xử lý và truyền dữ liệu từ module RFID RC522: Sau khi nhận thông tin thẻ qua giao thức SPI, ESP32 gửi dữ liệu đến vi điều khiển STM32 qua UART để xử lý cục bộ.
- Kết nối Web Server: ESP32 gửi dữ liệu thẻ lên máy chủ Web Server thông qua Wi-Fi, giúp kiểm tra và quản lý cơ sở dữ liệu người dùng. Kết quả từ Web Server (hợp lệ hay không hợp lệ) được truyền ngược lại STM32 để phản hồi cho người dùng.
- Điều khiển các tác vụ IoT: Với khả năng lập trình linh hoạt, ESP32 có thể mở rộng hệ thống, chẳng hạn tích hợp thêm tính năng cập nhật dữ liệu thời gian thực hoặc lưu trữ lịch sử truy cập thẻ trên đám mây.

Nhờ tích hợp Wi-Fi, ESP32 giúp hệ thống có khả năng kết nối và mở rộng dễ dàng với các ứng dụng kết nối mạng.

### 3.2.3 Module RFID RC522

Module RFID RC522 là thành phần đảm nhiệm chức năng quét thẻ RFID, có các nhiệm vụ chính sau:

- Đọc dữ liệu từ thẻ RFID: Module sử dụng giao tiếp SPI để truyền thông tin từ thẻ sang ESP32. Các dữ liệu này bao gồm mã số thẻ duy nhất (UID), giúp định danh người dùng.
- Xử lý tín hiệu thẻ nhanh và hiệu quả: Với khả năng giao tiếp tốc độ cao, RC522 đảm bảo việc quét thẻ diễn ra chính xác và không bị gián đoạn trong môi trường nhiều nhiễu.

RC522 là thiết bị đầu vào cốt lõi của hệ thống, cung cấp dữ liệu để xác minh danh tính người dùng.

### 3.2.4 Module LED RGB

LED RGB là thiết bị hiển thị trạng thái trực quan của hệ thống, giúp người dùng nhận biết tình trạng hoạt động mà không cần hiểu biết chuyên sâu. Các trạng thái hiển thị gồm:

- Màu xanh lá: Báo hiệu quá trình quét thẻ thành công. Sau khi thẻ được xác minh hợp lệ, STM32 điều khiển LED chuyển sang màu xanh lá trong 3 giây trước khi trở về trạng thái màu vàng.
- Màu đỏ: Khi thẻ không có trong cơ sở dữ liệu của hệ thống, LED RGB sẽ chuyển sang màu đỏ, giúp người dùng dễ dàng nhận biết lỗi.
- Màu vàng: Trạng thái chờ hoặc không có tín hiệu quét thẻ.

LED RGB đóng vai trò hỗ trợ giao tiếp giữa hệ thống và người dùng thông qua hiển thị màu sắc, góp phần cải thiện trải nghiệm sử dụng.

### 3.2.5 Module Buzzer

Module Buzzer cung cấp cảnh báo âm thanh trong hệ thống, với các chức năng chính:

- Thông báo quét thẻ thành công: Khi hệ thống xác định thẻ hợp lệ, Buzzer phát một âm thanh ngắn để thông báo cho người dùng.
- Báo lỗi: Khi thẻ không hợp lệ hoặc xảy ra sự cố trong hệ thống, Buzzer có thể phát âm cảnh báo khác biệt để thu hút sự chú ý của người dùng.

Buzzer là một thành phần đơn giản nhưng hiệu quả, giúp tăng khả năng phản hồi tức thời từ hệ thống.

### 3.2.6 Module LCD 1602

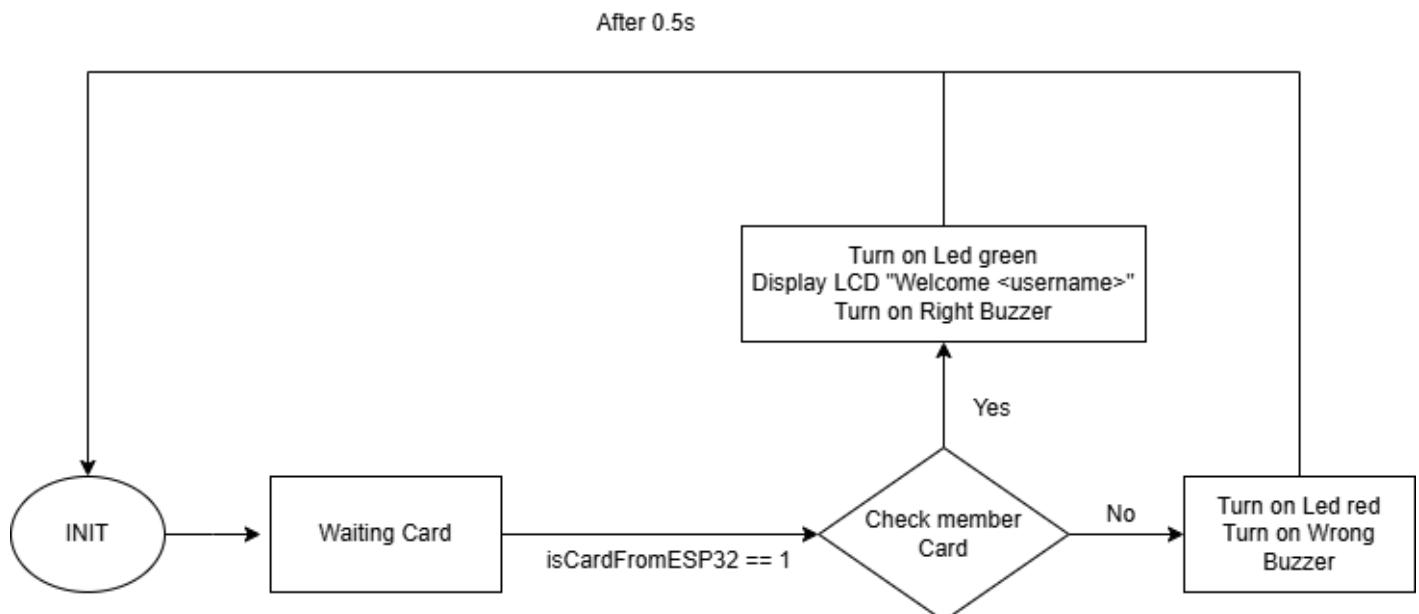
Màn hình LCD 1602 là thiết bị hiển thị thông tin, giúp người dùng tương tác trực tiếp với hệ thống. Các chức năng của LCD bao gồm:

- Hiển thị trạng thái: Sau khi quét thẻ thành công, LCD sẽ hiện thông báo "Welcome <tên người dùng>". Nếu quét thẻ thất bại, màn hình sẽ hiện thông báo "Unknown".

LCD 1602 đóng vai trò là giao diện chính giữa hệ thống và người dùng, giúp thông báo kết quả và hỗ trợ thao tác dễ dàng hơn.

## 3.3 Máy trạng thái của hệ thống

Dựa vào sơ đồ khái quát và vai trò, chức năng của các khối module, nhóm báo cáo xây dựng và thiết kế lưu đồ máy trạng thái của hệ thống như sau:



Hình 13: Lưu đồ máy trạng thái của hệ thống



- INIT: Là trạng thái đầu của chu kì máy trạng thái. Trạng thái này sẽ reset hệ thống, tắt các đèn LED, còi Buzzer, reset LCD về trạng thái ban đầu.
- WAITING CARD: Là trạng thái chờ đọc thẻ. Hệ thống sẽ chờ cho đến khi nào có tín hiệu thẻ được quét thì mới chuyển sang trạng thái kế tiếp.
- CHECK MEMBER CARD: Là trạng thái kiểm tra thẻ có trong Database hay không.
  - SIGNAL\_MEMBER: Là trạng thái nếu thẻ được quét có trong Database của hệ thống. Trạng thái này STM32 sẽ gửi các tín hiệu cho các module khác như: Bật đèn xanh cho LED RGB, bật còi Buzzer và gửi thông điệp cho màn hình LCD là "Welcome <Username>".
  - SIGNAL\_UNKNOWN: Là trạng thái nếu thẻ được quét không có trong Database của hệ thống. Trạng thái này STM32 sẽ gửi các tín hiệu cho các module khác như: Bật đèn đỏ cho LED RGB, bật còi Buzzer và gửi thông điệp cho màn hình LCD là "Unknown user".

Sau 0,5 giây, hệ thống sẽ quay lại trạng thái INIT và tiếp tục chu kì máy trạng thái kế tiếp.

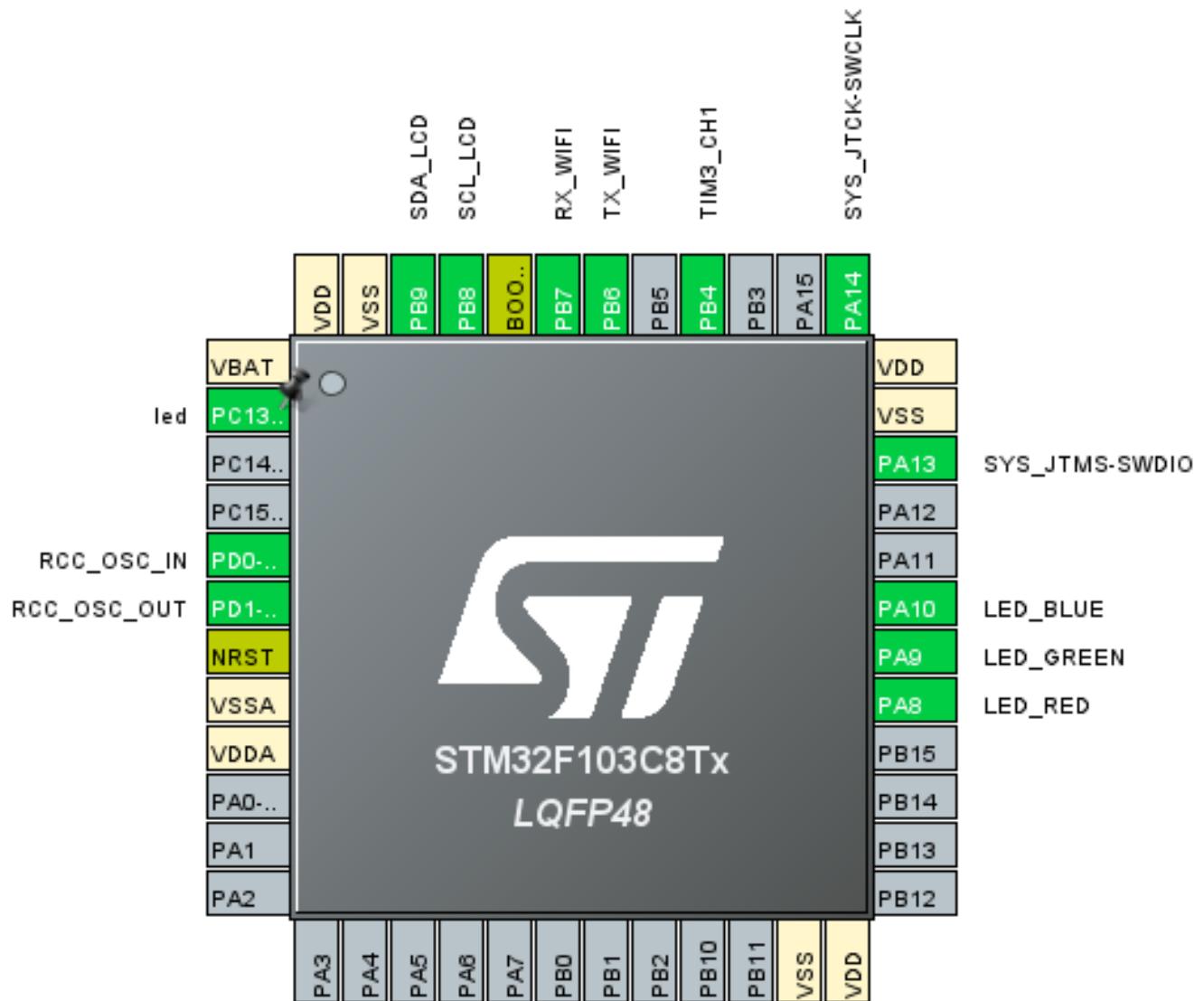
## 3.4 Hiện thực source code

### 3.4.1 Hiện thực cho vi điều khiển STM32

#### 3.4.1.1 Cấu hình cho vi điều khiển STM32

Để hiện thực một hệ thống cho vi điều khiển, trước tiên, ta phải cấu hình các chân GPIO, các chân giao tiếp, các thông số về Clock, Timer,... Nhóm báo cáo cấu hình cho vi điều khiển STM32 như sau:

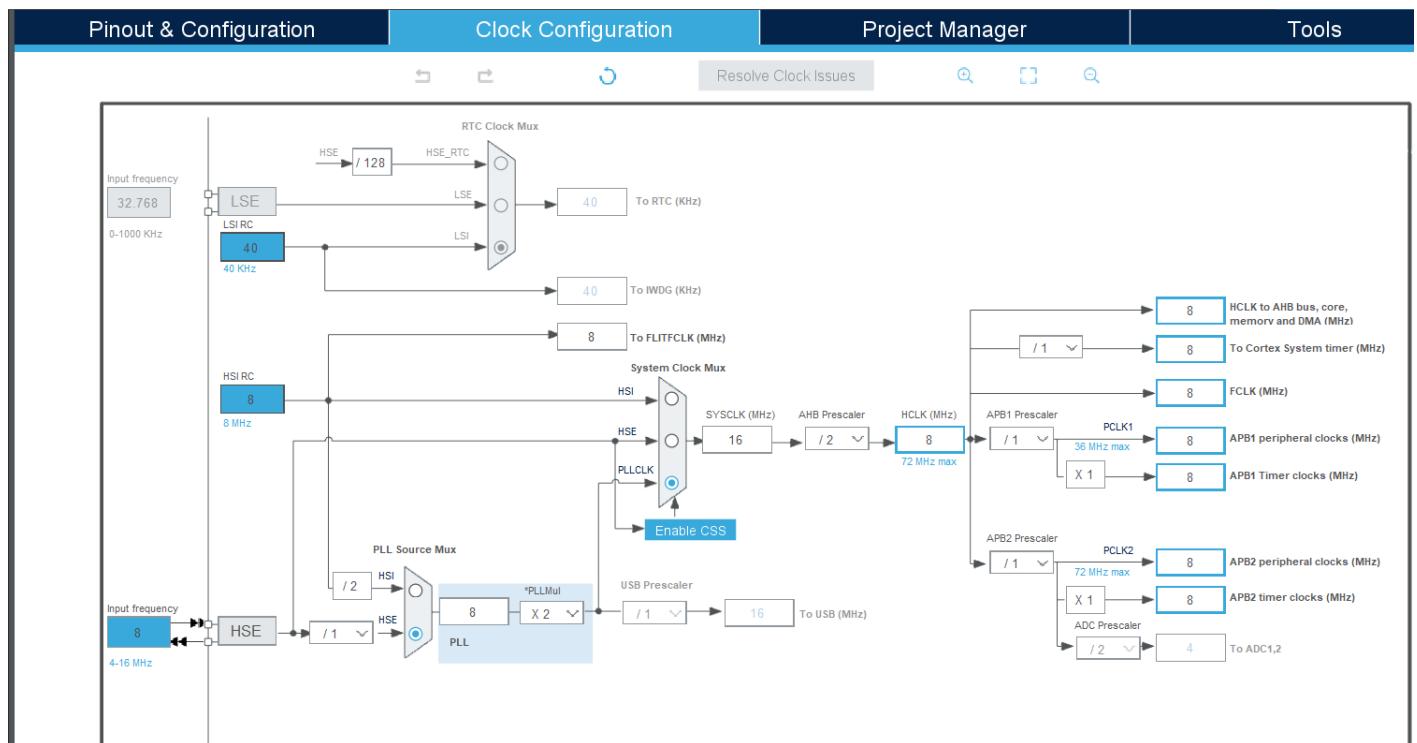
#### Cấu hình tổng thể



Hình 14: Cấu hình tổng thể cho STM32

### Cấu hình Clock và Timer

Về cấu hình Clock và Timer, nhóm báo cáo sử dụng tần số 8 MHz, nên để gọi hàm ngắt timer mỗi 10 ms, nhóm sẽ sử dụng cặp giá trị Prescaler và Counter Period lần lượt là 7999 và 9.



Hình 15: Cấu hình Clock

**TIM2 Mode and Configuration**

**Mode**

- Clock Source: Internal Clock
- Channel1: Disable
- Channel2: Disable
- Channel3: Disable
- Channel4: Disable
- Combined Channels: Disable

Use ETR as Clearing Source

XOR activation

One Pulse Mode

**Configuration**

Reset Configuration

User Constants	NVIC Settings	DMA Settings
Parameter Settings		

Configure the below parameters :

Search (Ctrl+F)

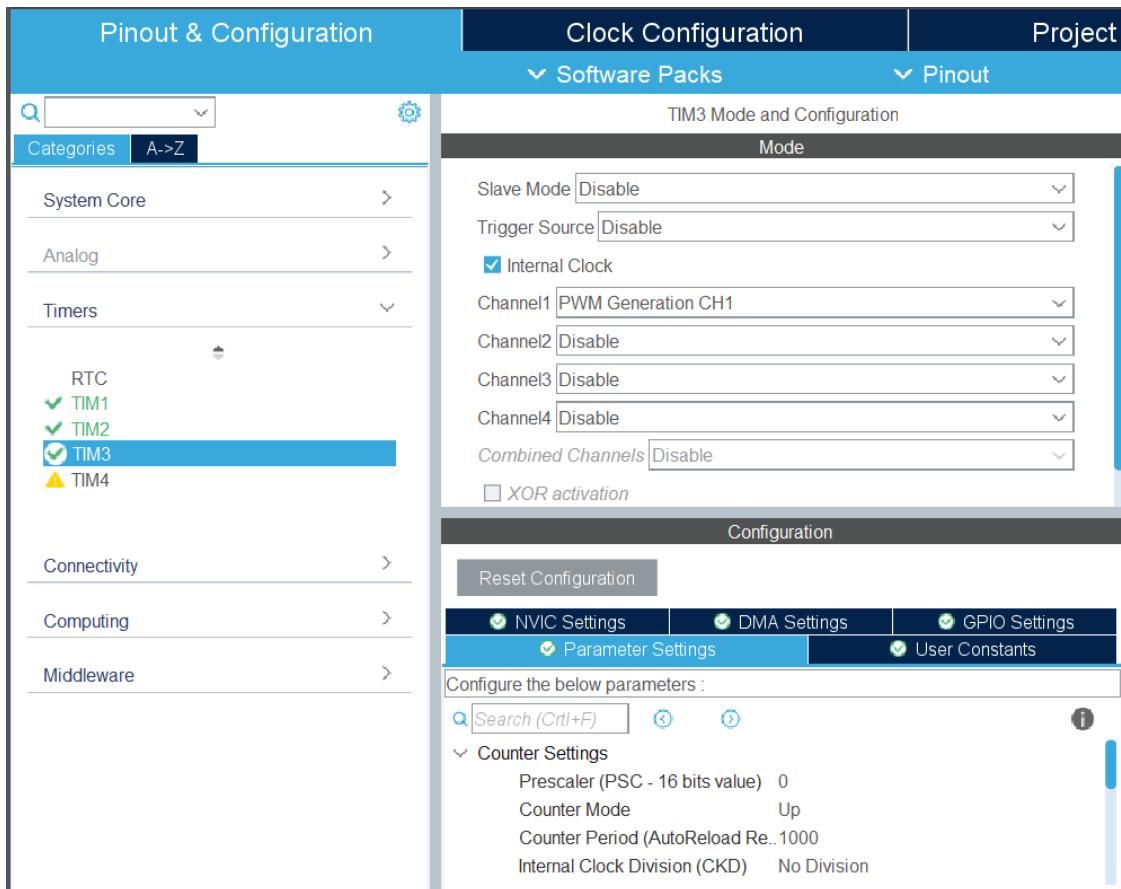
**Counter Settings**

- Prescaler (PSC - 16 bits value) : 7999
- Counter Mode : Up
- Counter Period (AutoReload Reg.) : 9
- Internal Clock Division (CKD) : No Division

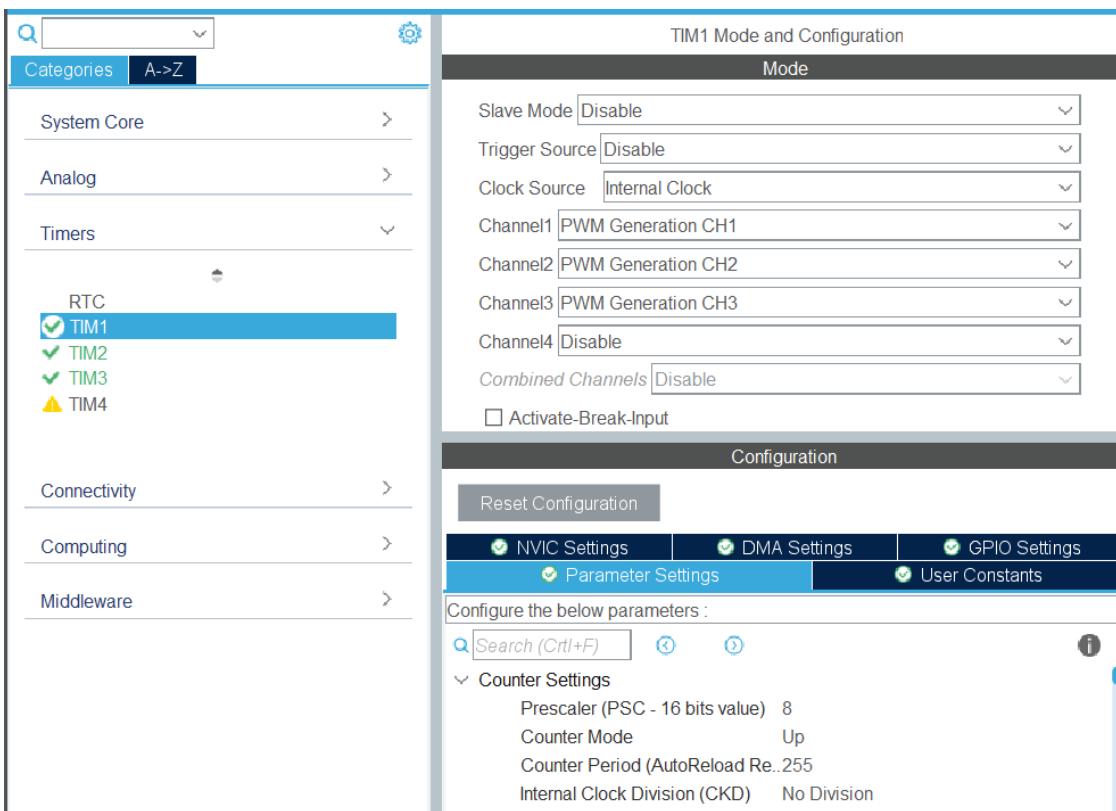
Hình 16: Cấu hình Timer

## Cấu hình PWM (Cho module Buzzer và Led RGB)

Nhóm báo cáo sử dụng TIM3 để hiện thực cho module Buzzer và TIM1 để hiện thực LED RGB (với 3 kênh tương ứng với 3 màu của LED).



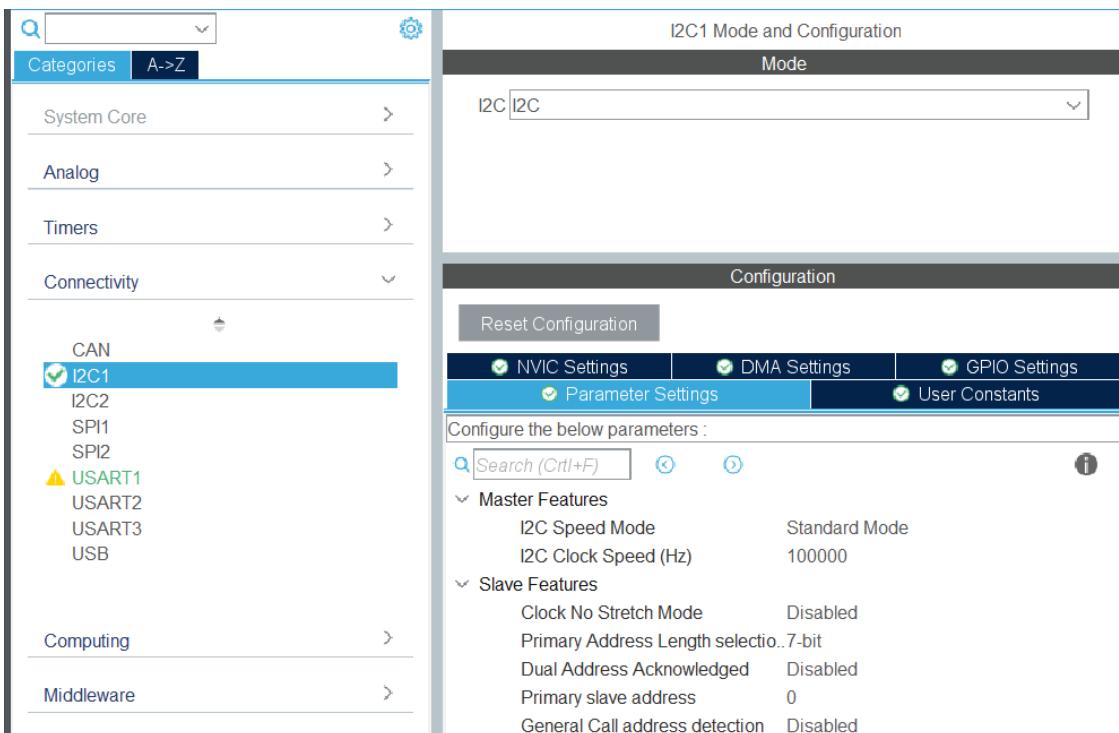
Hình 17: Cấu hình PWM cho Buzzer



Hình 18: Cấu hình PWM cho LED RGB

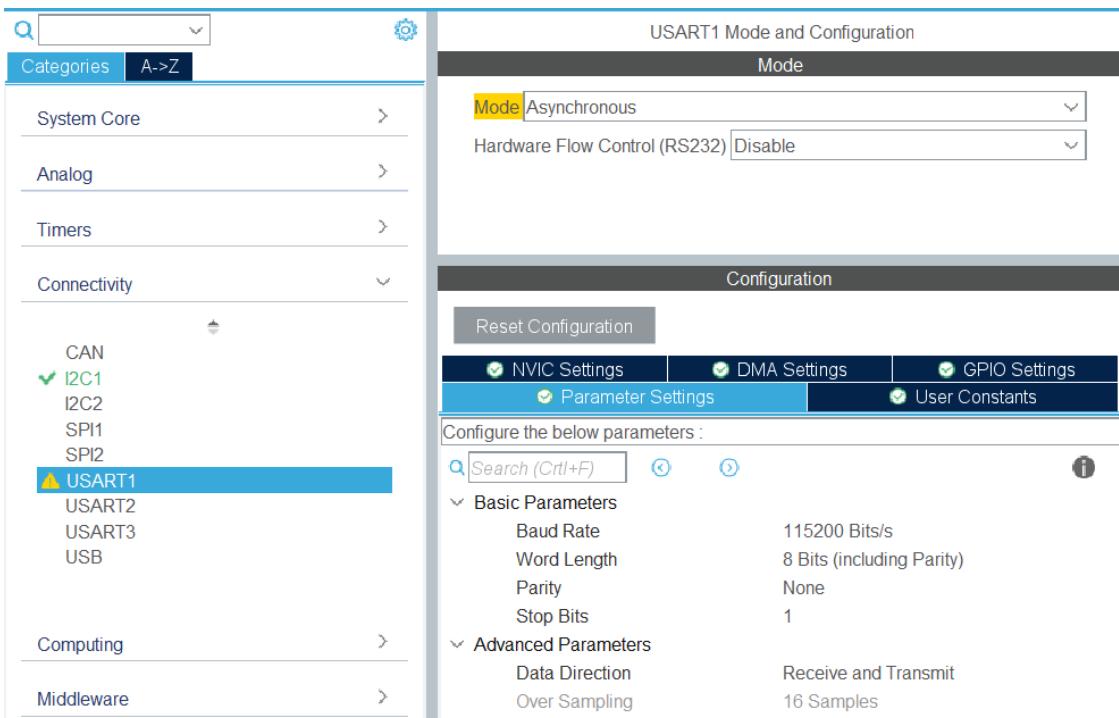
### Cấu hình giao tiếp I2C và UART (cho module LCD và ESP32)

Đối với cấu hình giao tiếp I2C cho LCD 1602, nhóm báo cáo sẽ cấu hình I2C1, với chế độ của Clock Speed là Standard mode, tần số là 100 KHz



Hình 19: Cấu hình I2C cho LCD 1602

Đối với cấu hình giao tiếp UART cho ESP32, nhóm báo cáo cấu hình USART1, với chế độ bất đồng bộ (Asynchronous), Baud Rate là 115200 Bits/s, chiều dài của Word là 8 Bits (đã bao gồm Parity bit) và Stop Bits là 1.



Hình 20: Cấu hình UART cho ESP32

### 3.4.1.2 Kiến trúc Scheduler và cấu trúc dữ liệu Linked List

Để hiện thực một hệ thống cho vi điều khiển, cụ thể là những hệ thống phức tạp đòi hỏi xử lý nhiều module và nhiều tác vụ (Task), ta không thể không nhắc tới các kiến trúc tổ chức chương trình và quản lý các tác vụ nhằm đảm bảo hệ thống thực thi đúng thời điểm, đúng thứ tự và không xảy ra xung đột giữa các tác vụ. Trong hệ thống này, nhóm báo cáo sử dụng kiến trúc Scheduler cho STM32, một kiến trúc giúp tổ chức và quản lý các tác vụ rất hiệu quả tích hợp thêm cấu trúc dữ liệu Linked List để lưu trữ các tác vụ giúp tăng hiệu năng và giảm độ phức tạp thuật toán.

Kiến trúc Scheduler và cấu trúc dữ liệu Linked List nhóm báo cáo hiện thực như sau:

```
1  typedef struct TaskNode
2  {
3      void (*pTask)(void);
4      uint32_t Delay;
5      uint32_t Period;
6      struct TaskNode* next;      //Next pointer
7 } TaskNode;
8
9 typedef struct
10 {
11     TaskNode* head;           //Head Node
12 } LinkedList;
13
14 LinkedList SCH_LL = {
15     .head = NULL
16 }; //Init Scheduler Linked List
17
18 void SCH_Init(void);           //Init Scheduler
19
20 void SCH_Add_Task(
21     void (*pFunction)(void),    //Task
22     uint32_t DELAY,          //Task's Delay
23     uint32_t PERIOD         //Task's Period
24 ); //Add Task Node to Scheduler Linked List
25
26 void SCH_Update(void);   //Update Delay of Task
27
28 void SCH_Dispatch_Tasks(void); //Perform the task when it's time !
29
30 void SCH_Delete_Task(void); //Delete the task after performing it
```

Listing 1: Scheduler của hệ thống

Với kiến trúc Scheduler được hiện thực như trên, ta sẽ gọi hàm thêm tác vụ trước Infinite loop để thêm hàm máy trạng thái (sẽ được nói rõ trong mục 3.4.1.3), hàm để thực hiện tác vụ ở Infinite Loop trong main của chương trình và hàm cập nhật Delay của tác vụ ở hàm ngắt timer (sẽ được hệ thống gọi mỗi 10 ms) như sau:

```
1 #include "main.h"
2 /*Includes some other header files*/
3 int main(void)
4 {
5
6     HAL_Init();
7     SystemClock_Config();
8
9     SCH_Add_Task(fsm_run, 0, 1); //Add the finite state machine
10    /* Infinite loop */
11    while (1)
12    {
13        SCH_Dispatch_Tasks();
14        //Just call only this function in the infinite loop
15    }
16 }
17
18 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
19 {
20     if(htim->Instance == TIM2)
21     {
22         SCH_Update();           //Update Scheduler every 10 ms
23     }
24 }
```

Listing 2: Infinite Loop và hàm ngắt Timer

Dể hiện thực một cách tối ưu cho kiến trúc Scheduler, do hàm `SCH_Update` sẽ được hệ thống gọi mỗi 10ms để thực hiện việc cập nhật Scheduler, nhóm báo cáo sẽ tối ưu kiến trúc Scheduler bằng cách giảm tối thiểu độ phức tạp của hàm này.

Ở hàm `SCH_Add_Task`, nhóm báo cáo sẽ thêm task vào Linked List và sắp xếp các Node theo độ tăng dần DELAY của các task. Sau đó, hàm `SCH_Dispatch_Tasks`, được gọi ở `while(1)`, sẽ kiểm tra DELAY ở `head`, nếu DELAY bằng 0 sẽ thực hiện task đó, xóa task và thêm lại task vào vị trí theo PERIOD của task hoặc xóa task nếu là one-shot task. Như vậy, ở hàm `SCH_Update`, ta chỉ cần giảm DELAY của task ở `head`. Từ đó, nhóm báo cáo đã tối ưu và giảm tối thiểu độ phức tạp thuật toán cho kiến trúc Scheduler.



### 3.4.1.3 Hiện thực máy trạng thái của hệ thống

Dựa vào lưu đồ máy trạng thái ở mục 3.3, nhóm báo cáo hiện thực mã nguồn cho máy trạng thái như sau:

```
1 void fsm_run(){
2     switch(status){ //Switch case for system status
3         case INIT:
4             signal_init(); //Init for LCD, Buzzer, LED RGB
5             timerReset = 100; //Count down this timer and reset system
6             status = WAITING_CARD;
7             cardFlag = 0;
8             userID = UNKNOWN;
9             break;
10        case WAITING_CARD:
11            signal_waiting(); //Turn On LED yellow
12            if(isCardFromESP32()) status = PROCESSING_CARD;
13            break;
14        case PROCESSING_CARD: //Case for check RFID Card
15            if(userID == UNKNOWN){
16                status = SIGNAL_UNKNOWN;
17                signal_reset();
18            }
19            else{
20                status = SIGNAL_MEMBER;
21                signal_reset();
22            }
23            break;
24        case SIGNAL_MEMBER: //If your card is already registered
25            signal_correct(); //Include LED RGB, LCD and Buzzer signal
26            if(timerReset > 0){
27                timerReset--;
28                if(timerReset <= 0) status = INIT;
29            }
30            break;
31        case SIGNAL_UNKNOWN: //If your card is not registered yet
32            signal_wrong(); //Include LED RGB, LCD and Buzzer signal
33            if(timerReset > 0){
34                timerReset--;
35                if(timerReset <= 0) status = INIT;
36            }
37            break;
38        default:
39            break;
40    }
41 }
```

Listing 3: Hàm máy trạng thái của hệ thống



### 3.4.2 Hiển thực cho vi điều khiển ESP32

#### 3.4.2.1 Cấu hình cho vi điều khiển ESP32

Để hiển thực cho vi điều khiển ESP32, trước hết, ta cần phải cấu hình các chân giao tiếp SPI với module RFID - RC522 và giao tiếp UART với vi điều khiển STM32.

```
1 //Configuration for UART
2 static const int RX_BUF_SIZE = (1024 * 2);
3 #define TXD_PIN (GPIO_NUM_17)
4 #define RXD_PIN (GPIO_NUM_16)
5 #define uart_num (UART_NUM_2)
6
7 //Configuration for SPI
8 #define SS_PIN 5
9 #define RST_PIN 0
```

Listing 4: Cấu hình các giao tiếp cho ESP32

#### 3.4.2.2 Kiến trúc chương trình ESP32

Viết chương trình cho ESP32, nhóm báo cáo chọn code style theo Arduino, giúp cho việc hiện thực chương trình và thực hiện các task dễ dàng và dễ tiếp cận cho người mới. Kiến trúc chương trình gồm 2 hàm `setup` và `loop`.

- Hàm `void setup`: Đây là hàm dùng để khởi tạo các chức năng của hệ thống. Trong hàm này, nhóm báo cáo sẽ setup các chức năng như Wifi, Web Server, module RFID - RC522, giao tiếp UART và hàm thời gian thực.
- Hàm `void loop`: Tương tự như cấu trúc `while(1)` hiện thực cho STM32, đây là hàm lặp vô hạn để chạy, thực thi các tác vụ đã khởi tạo ở hàm `setup`.

```
1 #include "wifi_config.h"
2 #include "web_server.h"
3 #include "rfid_config.h"
4 #include "time_config.h"
5 #include "uart_to_STM32.h"

6
7 void setup() {
8     Serial.begin(115200); //Setup Baud Rate
9     setupWiFi();          // Init Wifi
10    setupTime();          // Init time
11    setupRFID();          // Init RFID - RC522
12    setupWebServer();     // Init Web Server

13
14    Serial.println("Place RFID card near the reader:");
15    init_ua();             // Init UART
16 }

17
18 void loop() {
19    handleWebServer();     // Handle the Web Server
20    handleRFID();          // Perform RFID
21 }
```

Listing 5: Hiện thực chương trình cho ESP32

### 3.4.2.3 Xử lý front-end và back-end cho Web Server

#### Front-end cho Web Server

Để hiện thực giao diện cho Web Server, nhóm báo cáo sử dụng các ngôn ngữ lập trình chuyên dùng cho thiết kế giao diện và front-end cho Web Server: HTML, CSS và JavaScript. Ba file mã nguồn này sẽ được lưu trữ trong bộ nhớ Flash của ESP32 và sẽ được upload mỗi khi khởi động Web Server, bằng cách dùng thư viện LittleFS.

Để gọi 3 file mã nguồn HTML, CSS và JavaScript, ta dùng các hàm trong thư viện LittleFS như sau:



```
1 // Route for serving the main page
2 server.on("/", HTTP_GET, []() {
3     serveFile("/index.html", "text/html");
4 });
5
6 // Route for serving the CSS file
7 server.on("/styles.css", HTTP_GET, []() {
8     serveFile("/styles.css", "text/css");
9 });
10
11 // Route for serving the JavaScript file
12 server.on("/scripts.js", HTTP_GET, []() {
13     serveFile("/scripts.js", "application/javascript");
14 });
15
16 server.on("/images/logo.png", HTTP_GET, []() {
17     serveFile("/images/logo.png", "image/png");
18 });
19
```

Listing 6: Lưu trữ và upload file giao diện Web Server

### Back-end cho Web Server

Để hiện thực phần back-end cho Web Server, nhóm báo cáo sẽ khai báo mảng array dùng để lưu trữ các tên người dùng, mảng array lưu trữ các ID card, một struct CardInfo chứa thông tin thẻ, tên người dùng cuối cùng là một mảng chứa lịch sử của các thẻ được quét như sau:

```
1 String memberDatabase[] = {"51293B2", "34211E74", "849F2D74"};  
2 String memberNames[] = {  
3     "Lê Nguyễn Phúc Thịnh",  
4     "Nguyễn Kim Thuận",  
5     "Phan Huy Trung"  
6 };  
7  
8 struct CardInfo {  
9     String cardID;  
10    String name;  
11    String date;  
12    String time;  
13    bool isMember;  
14 };  
15 CardInfo scanHistory[10];
```

Listing 7: Database của hệ thống

Khi đã có hệ thống database rồi, ta sẽ hiện thực hàm `fetchTable()` trong mã nguồn JavaScript. Hàm này có nhiệm vụ khi có tín hiệu thẻ từ module RFID - RC522, đưa thông tin thẻ lên Web Server.

```
1 async function fetchTable() {  
2     try {  
3         const response = await fetch("/table");  
4         const tableContent = await response.text();  
5         document.getElementById("tableBody").innerHTML = tableContent;  
6     } catch (error) {  
7         console.error("Error fetching table data:", error);  
8     }  
9 }  
10 setInterval(fetchTable, 2000);  
11 fetchTable();
```

Listing 8: Hàm đưa thông tin thẻ lên Web Server khi quét

Kết hợp front-end và back-end, ta sẽ có một giao diện Web Server với chức năng là hiện lịch sử và thông tin người dùng mỗi khi quét thẻ RFID.



Hình 21: Giao diện của Web Server

### 3.4.3 Phân tích độ phức tạp thuật toán

#### 3.4.3.1 Độ phức tạp Big O của STM32

Với độ phức tạp Big O của vi điều khiển STM32, nhóm báo cáo chủ yếu sẽ đi phân tích thuật toán của một số hàm chính trong Scheduler.

##### SCH\_Add\_Task

- Độ phức tạp trong trường hợp xấu nhất (worst case) khi thêm task mới vào Linked List là phải duyệt qua danh sách có  $n$  task.
- Độ phức tạp:  $O(n)$ .

##### SCH\_Dispatch\_Tasks

- Ta cũng duyệt toàn bộ danh sách và thực hiện xử lý  $n$  task
- Độ phức tạp:  $O(n)$ .

##### fsm\_run

- Hàm này sẽ hoạt động như một state machine với số lượng trạng thái hữu hạn và cố định.
- Độ phức tạp:  $O(1)$ .

##### SCH\_Update

```
1 void SCH_Update()
2 {
3     if(SCH_LL.head == NULL) return;
4     if(SCH_LL.head->Delay > 0){
5         SCH_LL.head->Delay--;
6     }
7 }
```

Listing 9: Hàm update Scheduler

- Vì hàm này ở trong hàm ngắt Timer, sẽ được hệ thống gọi mỗi 10ms, nên ta sẽ phải có thuật toán để tối ưu hàm này thì hệ thống sẽ tăng năng suất lên đáng kể.
- Hàm này sẽ truy cập trực tiếp vào phần tử head của Linked List, giảm giá trị Delay nếu nó lớn hơn 0.
- Độ phức tạp:  $O(1)$ .

### 3.4.3.2 Độ phức tạp Big O của ESP32

Với độ phức tạp Big O của vi điều khiển ESP32, độ phức tạp thuật toán sẽ xoay quanh các hàm hiện thực việc đọc thẻ, hàm tạo bảng mỗi khi có thẻ quét vào module RFID - RC522 và hàm đọc file giao diện website.

#### processCard

- Hàm processCard sẽ gọi vòng lặp for loop để tìm kiếm trong memberDatabase và duyệt qua  $m$  phần tử (với  $m$  là số lượng thành viên trong cơ sở dữ liệu).
- Độ phức tạp là:  $O(m)$ .

#### serveFile

- Hàm serveFile sẽ gọi LittleFs.open và server.streamFile, đọc và gửi tệp có kích thước là  $s$ .
- Độ phức tạp là:  $O(s)$ .

#### generateTable

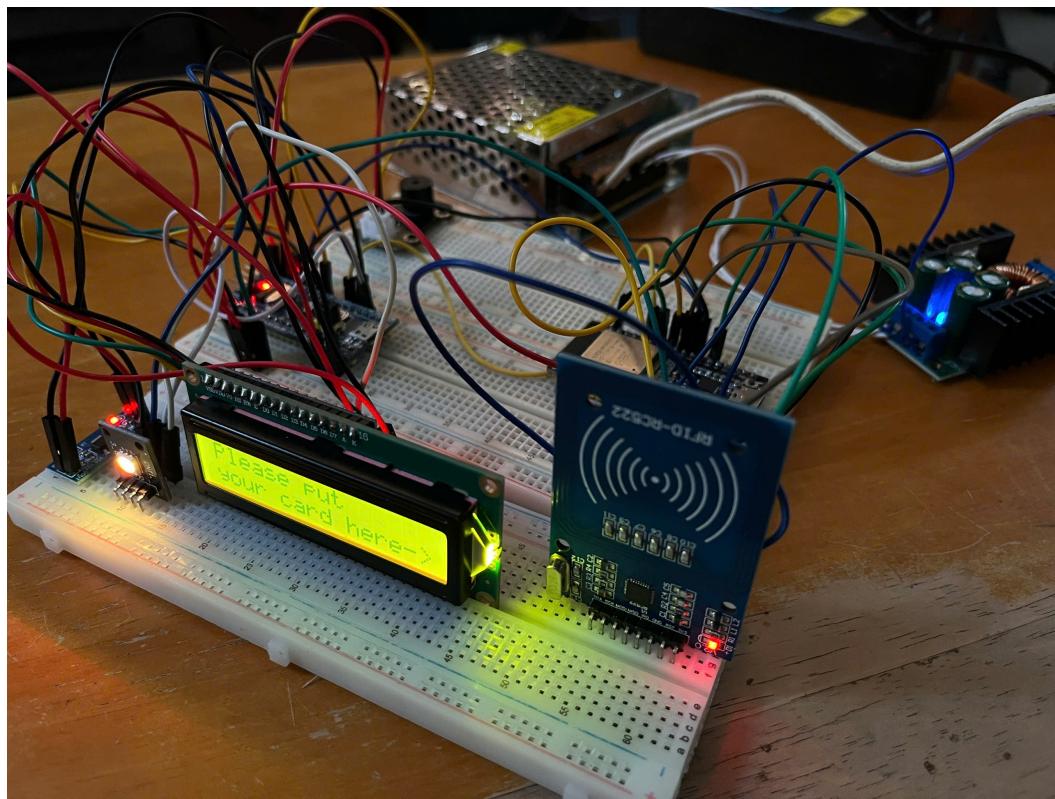
- Hàm generateTable sẽ gọi vòng lặp tạo bảng HTML từ mảng scanHistory:
  - Duyệt qua  $k$  phần tử, với  $k = \min(scanCount, 10)$ .
  - Giả sử mỗi thao tác chuỗi có độ phức tạp là  $O(l)$ , với  $l$  là chiều dài chuỗi (thường nhỏ)
- Tổng độ phức tạp:  $O(k \times l)$ .

### 3.4.3.3 Tổng hợp Big O Notation hệ thống

- STM32:  $O(n)$ , với  $n$  là số task được thêm vào.
- ESP32:  $O(n + m + k \times l)$ , với:
  - $n$ : Kích thước UID RFID.
  - $m$ : Số thành viên trong cơ sở dữ liệu.
  - $k$ : Số lịch sử quét.
  - $l$ : Kích thước chuỗi.

## 3.5 Mạch kết nối các module

Sau phần hiện thực code trên, ta tiến hành lắp mạch, kết hợp các module lại với nhau, ta sẽ có một hệ thống mạch kết nối các module như sau:



Hình 22: Mạch kết nối hoàn chỉnh

Về demo sản phẩm, nhóm báo cáo mời thầy truy cập đường link này để xem demo:

- Link demo sản phẩm:  
[https://youtu.be/dIH5JbN1\\_S8](https://youtu.be/dIH5JbN1_S8)
- Link demo sản phẩm + thuyết trình của nhóm:  
<https://youtu.be/WI9vyIaUOB0>

## 4 Tổng kết

### 4.1 Kết luận

Hệ thống được thiết kế nhằm đáp ứng các yêu cầu cơ bản về quản lý truy cập thông minh, đảm bảo tính tiện lợi, hiệu quả và khả năng mở rộng trong nhiều ứng dụng thực tế. Trong quá trình triển khai, hệ thống đã đạt được nhiều kết quả tích cực, tuy nhiên vẫn còn tồn tại một số hạn chế cần được khắc phục để hoàn thiện hơn. Dưới đây là các ưu điểm và nhược điểm của hệ thống:

#### Ưu điểm

Hệ thống được thiết kế với nhiều tính năng nổi bật và đem lại hiệu quả thực tế cao trong ứng dụng quản lý truy cập. Các ưu điểm chính bao gồm:

- **Tính linh hoạt:** Hệ thống hỗ trợ phương pháp xác thực hiện đại là thẻ RFID giúp tăng tính bảo mật và tiện lợi cho người dùng.
- **Khả năng kết nối mạng:** Nhờ sử dụng module ESP32 với Wi-Fi tích hợp, hệ thống có thể giao tiếp với máy chủ Web Server để quản lý cơ sở dữ liệu tập trung, giúp mở rộng quy mô và tích hợp IoT dễ dàng.
- **Phản hồi nhanh và trực quan:** Với LED RGB, màn hình LCD và Buzzer, hệ thống có khả năng thông báo trạng thái hoạt động một cách rõ ràng và tức thời, cải thiện trải nghiệm người dùng.
- **Chi phí thấp:** Hệ thống sử dụng các linh kiện phổ biến, giá cả phải chăng nhưng vẫn đảm bảo hiệu năng, phù hợp cho các ứng dụng thực tiễn quy mô nhỏ và vừa.
- **Tính mở rộng cao:** Hệ thống được xây dựng theo kiến trúc module, cho phép thêm hoặc thay đổi chức năng mà không cần thiết kế lại từ đầu.

#### Nhược điểm

Bên cạnh các ưu điểm kể trên, hệ thống vẫn tồn tại một số hạn chế cần được khắc phục:

- **Khả năng bảo mật:** Hiện tại, hệ thống chỉ dựa vào mã thẻ RFID, chưa có cơ chế bảo vệ mạnh mẽ như mã hóa dữ liệu hoặc xác thực nhiều yếu tố (MFA), dẫn đến nguy cơ bị giả mạo hoặc đánh cắp dữ liệu.
- **Giới hạn băng thông:** Do sử dụng giao thức UART và I2C, tốc độ truyền dữ liệu giữa các module bị giới hạn, có thể ảnh hưởng khi xử lý khối lượng dữ liệu lớn hoặc tốc độ cao.
- **Khả năng chống nhiễu:** Một số module như RFID và I2C dễ bị nhiễu trong môi trường công nghiệp hoặc nơi có nhiều tín hiệu không dây.
- **Hiển thị hạn chế:** Màn hình LCD 1602 chỉ có thể hiển thị thông tin cơ bản, chưa hỗ trợ đồ họa hoặc các giao diện phức tạp hơn.

- Hệ thống database sơ khai: Theo độ phức tạp thuật toán đã phân tích ở trên, database mà nhóm trình bày chỉ là hiện thực các mảng array đơn giản. Từ đó, việc truy xuất dữ liệu trong mảng trong trường hợp xấu nhất (worst-case) có thể rất phức tạp và tốn nhiều thời gian và tài nguyên nhất là khi database chứa càng nhiều thông tin người dùng.

## 4.2 Đánh giá và phương hướng phát triển hệ thống

### 4.2.1 Đánh giá và nhận xét

Hệ thống đã hoàn thành các mục tiêu đặt ra, bao gồm việc tích hợp nhiều module phần cứng khác nhau như RFID RC522, vi điều khiển STM32, Wi-Fi ESP32, LED RGB và các module phụ trợ như LCD 1602, Keypad 4x4, Buzzer. Kết quả đạt được chứng minh sự hiệu quả của mô hình trong việc quản lý truy cập thông minh thông qua cả thẻ từ RFID và mã PIN. Hệ thống vận hành ổn định, giao tiếp giữa các module thông qua các chuẩn giao tiếp UART, SPI và I2C đã đáp ứng yêu cầu kỹ thuật, đồng thời thể hiện tính linh hoạt khi triển khai.

Tuy nhiên, hệ thống vẫn còn một số hạn chế cần được lưu ý. Đầu tiên, khả năng bảo mật vẫn còn đơn giản và dễ bị khai thác nếu kẻ xấu có ý định tấn công, đặc biệt trong môi trường thực tế nơi yêu cầu bảo mật cao hơn. Ví dụ, giao thức UART và SPI không được mã hóa, dẫn đến nguy cơ dữ liệu có thể bị nghe lén hoặc giả mạo. Thứ hai, tốc độ xử lý dữ liệu, đặc biệt khi hệ thống cần xử lý nhiều yêu cầu đồng thời, có thể gặp hạn chế do khả năng xử lý của phần cứng. Thứ ba, giao diện người dùng trên màn hình LCD và Keypad còn khá cơ bản, chưa hỗ trợ thao tác thân thiện hay hiển thị chi tiết hơn, đặc biệt khi cần mở rộng quy mô hoặc tích hợp thêm các tính năng khác. Cuối cùng, hệ thống Database vẫn còn khá đơn giản, chỉ phù hợp với những dự án nhỏ với số lượng thông tin người dùng hạn chế, chưa đáp ứng được xu hướng phát triển của thế giới nhất là trong thời đại dữ liệu lớn (Big Data).

### 4.2.2 Phương hướng phát triển hệ thống

Dựa trên những đánh giá trên, hệ thống có thể được nâng cấp theo các hướng sau:

#### 1. Tăng cường bảo mật

- Áp dụng các phương pháp mã hóa cho giao tiếp giữa các module, đặc biệt là giao tiếp UART và SPI. Ví dụ, có thể tích hợp các thuật toán mã hóa như AES hoặc RSA.
- Sử dụng các giao thức bảo mật hơn như HTTPS hoặc MQTT cho việc truyền thông tin giữa ESP32 và Web Server.
- Xây dựng cơ chế xác thực mạnh hơn, chẳng hạn như kết hợp thẻ từ RFID với mã PIN hoặc sinh trắc học (vân tay, nhận diện khuôn mặt).

#### 2. Cải thiện tốc độ và hiệu suất

- Nâng cấp phần cứng: Thay thế vi điều khiển STM32 bằng các vi điều khiển hoặc bộ xử lý mạnh hơn nếu cần xử lý đa nhiệm.
- Tối ưu hóa mã nguồn và luồng dữ liệu, giảm thiểu độ trễ giữa các module.

#### 3. Nâng cấp giao diện người dùng



- Thay thế màn hình LCD 1602 bằng các loại màn hình đồ họa (OLED hoặc TFT) để hiển thị thông tin trực quan hơn.
- Tích hợp giao diện người dùng trên ứng dụng di động, giúp quản lý và theo dõi hệ thống dễ dàng hơn.

#### 4. Mở rộng tính năng

- Hỗ trợ nhiều người dùng với quyền hạn khác nhau, cho phép phân cấp truy cập.
- Lưu trữ dữ liệu quét thẻ và mật khẩu trên cơ sở dữ liệu đám mây để thuận tiện theo dõi và phân tích.
- Tích hợp các cảm biến khác như cảm biến chuyển động hoặc camera để tăng cường khả năng giám sát và bảo mật.
- Mở rộng thêm tính năng cho phép người dùng đăng ký ID card và thông tin người dùng thông qua giao diện trang Web hoặc hệ thống phần cứng.

#### 5. Tăng tính thực tế và ứng dụng

- Đánh giá hoạt động trong môi trường thực tế để kiểm tra tính ổn định và khắc phục các lỗi phát sinh.
- Thiết kế vỏ hộp bảo vệ toàn bộ hệ thống, đảm bảo độ bền cơ học và khả năng hoạt động trong các điều kiện môi trường khác nhau.

#### 6. Nâng cấp và cải thiện hệ thống Database

- Có thể cân nhắc thay thế các mảng array đơn giản thành các cấu trúc dữ liệu cao cấp hơn như Danh sách liên kết (Linked List), Ngăn xếp (Stack), Hàng đợi (Queue), hay các cấu trúc dữ liệu có phân cấp như Cây (Tree) đi kèm với các giải thuật tối ưu hơn như giải thuật sắp xếp, chèn, xóa, đệ quy,... giúp tăng hiệu suất và giảm độ phức tạp cho hệ thống.
- Thực tế hơn, ta có thể thay thế cơ chế lưu trữ mảng array bằng các hệ quản trị cơ sở dữ liệu hiện đại như SQLite, MySQL hoặc Firebase đi kèm với các phương pháp truy vấn tối ưu để quản lý dữ liệu người dùng một cách hiệu quả hơn nhất là khi hệ thống phải xử lý với số lượng người dùng lớn.
- Xây dựng và phát triển thêm các tính năng quản lý cơ sở dữ liệu thông minh, như tìm kiếm người dùng theo ID hoặc tên, cập nhật thông tin cá nhân và xóa dữ liệu khi cần thiết.
- Tích hợp thêm các cơ chế sao lưu và phục hồi dữ liệu, đảm bảo không mất dữ liệu trong trường hợp hệ thống gặp sự cố.
- Kết nối Database với hệ thống đám mây để đồng bộ hóa dữ liệu và hỗ trợ truy cập từ xa thông qua Web Server hoặc ứng dụng di động.

Những cải tiến này không chỉ giúp hệ thống trở nên toàn diện hơn mà còn đảm bảo khả năng phát triển dài hạn, phù hợp với các yêu cầu ngày càng cao của ứng dụng thực tế trong môi trường công nghiệp hoặc dân dụng.



### 4.3 Kết quả đánh giá

Nhóm báo cáo xin tổng kết kết quả đánh giá từng thành viên trong nhóm.

STT	Họ và tên	MSSV	Nội dung công việc	Mức độ hoàn thành
1	Nguyễn Kim Thuận	2213357	<ul style="list-style-type: none"><li>Hiện thực giao tiếp UART cho ESP32</li><li>Hiện thực PWM cho LED RGB</li></ul>	100%
2	Phan Huy Trung	2213709	<ul style="list-style-type: none"><li>Hiện thực giao tiếp I2C cho LCD</li><li>Hiện thực PWM cho Buzzer</li><li>Làm slide thuyết trình</li></ul>	100%
3	Lê Nguyễn Phúc Thịnh	2213279	<ul style="list-style-type: none"><li>Hiện thực giao tiếp SPI cho RFID - RC522</li><li>Hiện thực giao diện và tính năng cho Web Server</li><li>Hiện thực máy trạng thái và Scheduler cho STM32</li><li>Làm báo cáo</li></ul>	100%



## 5 Tài liệu tham khảo

[1] Electronic Wings. *RFID RC522 Interfacing with ESP32*.

Truy cập từ: <https://www.electronicwings.com/esp32/rfid-rc522-interfacing-with-esp32>

[2] Valvano, J. W. (2012). *Introduction to Arm Cortex-M Microcontrollers*. Nhà xuất bản CreateSpace Independent Publishing Platform.

[3] Texas Instruments. (2015). *Understanding the I2C Bus*

Truy cập từ <https://www.ti.com/lit/an/slva704/slva704.pdf>

[4] Analog Devices. (n.d.). *Introduction to SPI Interface*

Truy cập từ <https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html>

[5] Analog Devices. (n.d.). *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter*

Truy cập từ <https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>

[6] Espressif. *ESP-AT Command Firmware*

Truy cập từ <https://www.espressif.com/en/products/sdks/esp-at/overview>