
TÓM LƯỢC

Nhóm nghiên cứu đang tiến hành một dự án quan trọng với mục tiêu phát triển và cung cấp một kit thí nghiệm toàn diện, đáp ứng các yêu cầu từ các kiến thức cơ bản đến những công nghệ và kỹ thuật tiên tiến nhất trong lĩnh vực lập trình Hệ thống nhúng và Internet of Things (IoT). Dự án này đặt ra mục tiêu chính là tạo ra một công cụ linh hoạt và mạnh mẽ để hỗ trợ cả quá trình học và ứng dụng thực tế.

Kit thí nghiệm và bộ bài thí nghiệm, server backend đi kèm không chỉ là một bộ công cụ giáo dục, mà còn tạo nên một môi trường thực hành độc đáo. Nó sẽ cung cấp các kiến thức từ cơ bản đến nâng cao, giúp người sử dụng hiểu thêm về các khái niệm lập trình nhúng và triển khai chúng trong các ứng dụng IoT. Qua việc sử dụng Kit và bộ bài thí nghiệm, người học sẽ có cơ hội thực hành và làm quen với các kỹ thuật cụ thể như lập trình vi điều khiển, giao tiếp với cảm biến, và quản lý dữ liệu từ IoT đến máy chủ. Điều này không chỉ giúp họ học lý thuyết mà còn trang bị kỹ năng thực tế, sẵn sàng áp dụng trong nhiều lĩnh vực, từ nghiên cứu đến phát triển sản phẩm.

Nhằm hỗ trợ người dùng hiện thực hóa các dự án IoT một cách đơn giản và hiệu quả, nhóm sẽ xây dựng một server backend hỗ trợ giao thức HTTP. Server này không chỉ giúp người dùng dễ dàng kết nối và quản lý các project, data mà còn cung cấp một nền tảng mạnh mẽ cho việc thu thập và xử lý dữ liệu từ các thiết bị.

Dể hoàn thiện đề tài này một cách hiệu quả, nhóm chúng tôi đã phân chia các công việc cụ thể, đảm bảo phù hợp với tiến trình thực hiện. Việc này không chỉ giúp tối ưu hóa thời gian và nguồn lực mà còn đảm bảo mỗi thành viên có thể tập trung vào thế mạnh của mình. Dưới đây là chi tiết phân công công việc cho đề tài:

- Khảo sát, tìm hiểu công nghệ ARM trong hệ thống nhúng IoT.
- Thiết kế, hiện thực phần cứng KIT thí nghiệm.

-
- Xây dựng bộ bài thí nghiệm vi điều khiển trên KIT để khảo sát các chức năng và phục vụ việc học tập kiến thức về Vi điều khiển, mở rộng một phần ra hệ thống nhúng và IoT.
 - Thu thập ý kiến phản hồi của người dùng và cải tiến KIT về phần cứng cũng như tài liệu hướng dẫn.
 - Hiện thực server backend hỗ trợ http phục vụ cho các bài thí nghiệm liên quan đến IoT.

Dự án đặt ra nhiệm vụ quan trọng để phát triển một Kit thí nghiệm có thể đáp ứng hầu hết các nhu cầu cơ bản trong lĩnh vực Hệ thống nhúng và IoT. Đầu tiên, nhóm chúng tôi sẽ tiến hành một quá trình tìm hiểu sâu rộng, khảo sát chi tiết về Hệ thống nhúng và IoT để xác định rõ yêu cầu cụ thể cho Kit thí nghiệm. Sau đó, chúng tôi sẽ đề xuất một mô hình tổng quan và đặc tả chi tiết về chức năng của Kit, dựa trên nhu cầu cụ thể từ cộng đồng người học và nghiên cứu. Tiếp đó, chúng tôi sẽ thiết kế, hiện thực và kiểm nghiệm Kit để đảm bảo tính hiệu quả và đáp ứng mọi yêu cầu đã đề xuất. Đây là một quá trình toàn diện nhằm tạo ra một công cụ học tập và thử nghiệm linh hoạt, đáp ứng nhu cầu đa dạng của cộng đồng trong lĩnh vực này. Trong bước tiếp theo, chúng tôi sẽ xây dựng các bộ tài liệu học tập dựa trên Kit thí nghiệm và các bộ thư viện hỗ trợ lập trình. Sau đó, nhóm hướng tới sẽ phát triển thêm server backend IoT hỗ trợ http. Việc xây dựng một server IoT riêng sẽ tạo ra khả năng kết hợp linh hoạt với Kit thí nghiệm và bộ tài liệu hướng dẫn để người học có thể dễ dàng luyện tập phát triển ứng dụng nhúng và IoT theo mong muốn của mình.

Mục lục

Chapter 1. ĐẶT VẤN ĐỀ VÀ KHẢO SÁT NHU CẦU	1
1.1 Đặt vấn đề	1
1.2 Hệ thống nhúng và IoT ở Việt Nam	2
1.2.1 Hệ thống nhúng	2
1.2.2 Internet of Things (IoT)	3
1.2.3 Nhu cầu về nhân lực	4
1.2.4 Khảo sát Kit thí nghiệm trên thị trường	4
1.2.5 So sánh Kit thí nghiệm	7
1.2.6 Khảo sát server IoT	9
1.3 Đề xuất yêu cầu của sản phẩm	12
1.4 Cấu trúc của đề tài	13
Chapter 2. TỔNG QUAN TÀI LIỆU	15
2.1 Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển .	15
2.1.1 Vi điều khiển (MCU)	15
2.1.2 Lõi kiến trúc ARM	16
2.1.3 Hệ thống hướng sự kiện	18
2.1.4 Hệ thống hướng thời gian	19
2.1.5 Máy trạng thái hữu hạn	20
2.1.6 Cơ chế DMA	21
2.2 Kiến thức về các giao thức truyền thông logic	22
2.2.1 Giao tiếp UART	23
2.2.2 Giao tiếp SPI	24
2.2.3 Giao tiếp I2C	26
2.3 Kiến thức về các chuẩn giao tiếp vật lý	28
2.3.1 RS232 và RS485	28
2.3.2 SDcard	29
2.3.3 Ethernet	29
2.3.4 CAN bus	30

2.4	Kiến thức về xây dựng tài liệu	31
2.5	Kiến thức về Database và Backend	32
2.5.1	RESTful API	32
2.5.2	JSON	33
2.5.3	NODEJS	34
2.5.4	MONGODB	35
Chapter 3. MỤC TIÊU - PHƯƠNG PHÁP THIẾT KẾ HỆ THỐNG 37		
3.1	MỤC TIÊU THIẾT KẾ PHẦN CỨNG	37
3.1.1	Thiết kế khối vi điều khiển trung tâm	39
3.1.2	Khối tương tác với người dùng	43
3.1.3	Thiết kế khối đọc ADC	47
3.1.4	Thiết kế khối đồng hồ thời gian thực	49
3.1.5	Các khối giao tiếp	50
3.1.6	Khối kết nối internet	54
3.1.7	Bộ nhớ	57
3.1.8	Khối nguồn	60
3.2	MỤC TIÊU THIẾT KẾ TÀI LIỆU HƯỚNG DẪN	62
3.3	MỤC TIÊU THIẾT KẾ MÁY CHỦ	63
3.3.1	Mô tả nghiệp vụ server Iot	63
3.3.2	Sơ đồ thiết kế	65
3.3.3	Workflow của hệ thống	69
3.4	VỀ PHƯƠNG PHÁP THIẾT KẾ PHẦN CỨNG	70
3.4.1	Giới thiệu	70
3.4.2	Thi công board mạch	71
3.5	VỀ PHƯƠNG PHÁP HƯỚNG DẪN SỬ DỤNG	73
3.5.1	Các kiến thức sử dụng trong các bài thí nghiệm	74
3.5.2	Quy trình thiết kế bài thí nghiệm	74
3.5.3	Các nội dung của bài thí nghiệm	75
3.6	PHƯƠNG PHÁP THI CÔNG SERVER	77
3.6.1	Thiết kế API	77
Chapter 4. KIỂM QUẢ - THẢO LUẬN 79		
4.1	Kết quả kiểm thử và đánh giá phần cứng	79
4.2	Kết quả kiểm thử kiến thức có trong bộ bài thí nghiệm	81
4.3	Kết quả kiểm thử server backend	81
4.3.1	Một vài kịch bản kiểm thử	82
4.3.2	Kiểm thử hiệu suất	87

4.4 Triển khai thực tế	90
Chapter 5. KẾT LUẬN - ĐỀ NGHỊ	91
5.1 Kết luận về kết quả đạt được	91
5.2 Những hạn chế còn tồn tại	93
5.3 Những đề nghị về hướng phát triển của hệ thống	93
Chapter 6. TÀI LIỆU THAM KHẢO	95

Danh sách hình ảnh

1.1	Arduino Uno	5
1.2	Arduino Mega	5
1.3	Board NUCLEO STM32F103RB	6
1.4	Kit STM32F407 Discovery	6
1.5	Mạch tách rời module Yolobit	7
2.1	Tổng quan MCU (Nguồn: Max Embedded)	16
2.2	Mô hình kiến trúc arm	17
2.3	Cơ chế xử lý ngắn	19
2.4	Một ví dụ về FSM (Nguồn: xiaoyunyang)	21
2.5	Sơ đồ khối tổng quan DMA (Nguồn: Open4Tech)	22
2.6	Sơ đồ giao tiếp giữa 2 thiết bị qua UART	23
2.7	Các điểm cần lưu ý về UART	24
2.8	Dataframe của UART	24
2.9	Sơ đồ giao tiếp SPI	25
2.10	Kết nối I2C giữa Master và Slave	26
2.11	Dataframe của I2C	27
3.1	Ứng dụng thực tế của LED 7 đoạn và ma trận phím	38
3.2	STM32F407ZGT6	39
3.3	Sơ đồ nguyên lý của STM32F407ZGT6	40
3.4	Thạch anh ngoài tạo xung cho Vi điều khiển	40
3.5	Sơ đồ nguyên lý của LSE crystal	41
3.6	Bộ lọc nguồn cho Vi điều khiển	41
3.7	Vi điều khiển sử dụng làm chip nạp	42
3.8	Thạch anh ngoài tạo xung cho Vi điều khiển	42
3.9	Bộ lọc nguồn cho chip nạp	43
3.10	Sơ đồ nguyên lý ma trận phím	44
3.11	Module 4 led 7 đoạn	45
3.12	Sơ đồ nguyên lý led 7 đoạn	45

3.13 LCD touch 3.2in 320x240	46
3.14 Sơ đồ nguyên lý màn hình LCD touch	46
3.15 Mạch chia áp (Nguồn: All About Circuits)	47
3.16 Sơ đồ nguyên lý của khối cảm biến	48
3.17 Các cảm biến trên kit thí nghiệm	49
3.18 Sơ đồ nguyên lý của khối thời gian thực	49
3.19 Hình ảnh pin CR1220	50
3.20 Sơ đồ nguyên lý của kết nối RS232 trên mạch	50
3.21 Cổng kết nối DB9	51
3.22 Sơ đồ nguyên lý của kết nối RS485 trên mạch	52
3.23 Sơ đồ nguyên lý của kết nối CAN trên mạch	53
3.24 Sơ đồ nguyên lý của input trên mạch	53
3.25 Sơ đồ nguyên lý của output trên mạch	54
3.26 ESP8266 nhóm sử dụng trên mạch	55
3.27 ESP8266 nhóm sử dụng trên mạch	55
3.28 Sơ đồ nguyên lý của CH340 trên mạch	56
3.29 Sơ đồ nguyên lý của kết nối module sim	57
3.30 Module Sim A7670C	57
3.31 SRAM sử dụng cho LCD touch	58
3.32 Sơ đồ nguyên lý Eeprom	58
3.33 Sơ đồ nguyên lý bộ nhớ Flash	59
3.34 Sơ đồ nguyên lý thẻ microSD	59
3.35 Khe cắm thẻ microSD	60
3.36 Thiết kế nguồn	60
3.37 Mạch hạ áp	61
3.38 Sơ đồ ERD	65
3.39 Lược đồ các bảng dữ liệu	66
3.40 Flow work của User	69
3.41 Flow work của Device	70
3.42 Các yêu cầu cơ bản mà nhóm định hướng	71
3.43 Mô phỏng của mạch trên Altium Designer	71
3.44 Hình ảnh hoàn thiện mạch của Kit thí nghiệm	72
3.45 Hình ảnh của kit thí nghiệm	72
3.46 Kết nối các sản phẩm có trong bộ Kit thí nghiệm	73
3.47 Cấu trúc bài thí nghiệm	74
3.48 Nội dung tài liệu hướng dẫn	75
3.49 Cấu trúc thư mục xây dựng server	78

4.1	Cấu trúc thư mục postman	82
4.2	Thư mục kiểm thử 1	83
4.3	Kết quả kiểm thử 1	84
4.4	Kết quả kiểm thử 2	85
4.5	Kết quả kiểm thử 3	87
4.6	Kết quả kiểm thử tải 1	88
4.7	Kết quả kiểm thử tải 2	88
4.8	Kết quả kiểm thử tải 3	89
4.9	Kết quả kiểm thử tải 4	89

Danh sách bảng

1.1	Bảng cấu trúc đề tài	14
3.1	Thông số STM32F407ZGT6	39
3.2	Bảng USER	67
3.3	Bảng DEVICE	67
3.4	Bảng USER_DEVICE_MAPPING	68
3.5	Bảng PROJECT	68
3.6	Bảng API	68
3.7	Bảng DATALOG	69
4.1	Dánh giá, phản hồi sản phẩm	90

CHƯƠNG 1

ĐẶT VẤN ĐỀ VÀ KHẢO SÁT NHU CẦU

1.1 Đặt vấn đề

Hiện nay, nhu cầu về nguồn nhân lực cho lĩnh vực Hệ thống nhúng và Ứng dụng IoT ngày càng lớn. Trong khi đó, nhu cầu về ứng dụng IoT của người dùng và số lượng các doanh nghiệp phát triển về IoT ngày càng tăng mạnh, đặc biệt trong bối cảnh Việt Nam đang đẩy mạnh chủ trương chuyển đổi số từ năm 2020 của Chính phủ [?]. Mặc dù trên thị trường đã xuất hiện nhiều sản phẩm tương tự, tuy nhiên, nhiều trong số chúng lại không tích hợp sẵn các module, gây khó khăn trong việc kết nối và sử dụng. Việc phải sử dụng dây bus và breadboard không chỉ làm giảm tính thẩm mỹ mà còn tăng độ phức tạp của quá trình học. Vấn đề này đặt ra một thách thức cụ thể cần giải quyết là phát triển kit thí nghiệm tích hợp hầu hết các tính năng cơ bản để học lập trình trở nên thuận tiện và hiệu quả, phù hợp với cả những người mới bắt đầu và cả những người đang nghiên cứu, kiểm thử các ứng dụng thực tế. Việc phát triển kit thí nghiệm và bộ bài thí nghiệm có thể giúp tạo ra môi trường học tập năng động và sáng tạo, khuyến khích sự yêu thích và tò mò cho người học.

Để giúp cho người học, người nghiên cứu dễ dàng tiếp cận tới các công nghệ này một cách dễ dàng hơn, có thể tích hợp tìm hiểu nhiều công nghệ mới trên thị trường, đồng thời vẫn giữ được những bài toán, những cách xử lý hay của những dòng kit thí nghiệm trước nhóm chúng tôi quyết định chọn đề tài "*Phát triển kit thí nghiệm Hệ thống nhúng và ứng dụng IOT dựa trên lõi kiến trúc ARM*" để nghiên cứu trong đề tài lần này, với mong muốn từ những người dùng có kiến thức từ cơ bản nhất về lập trình cho đến những người đã lập trình lâu năm đều tìm thấy được những ứng dụng cụ thể khi sử dụng kit. Hơn thế nữa, những người học, người làm có thể ứng dụng kit vào công việc, học tập và nghiên cứu.

1.2 Hệ thống nhúng và IoT ở Việt Nam

1.2.1 Hệ thống nhúng

Hệ thống nhúng là tổ hợp phần cứng và phần mềm được thiết kế để giải quyết các vấn đề cụ thể, chuyên biệt của một hệ thống lớn hơn trong ngành truyền thông, quan trắc, tự động hóa,... [?]

Tính chất của hệ thống nhúng:

- Là một hệ thống phức tạp nằm trong một hệ thống lớn mà nó được giao nhiệm vụ điều khiển.
- Chỉ được thiết kế để giải quyết 1 số nhiệm vụ chuyên dụng, không được thiết kế với mục đích phổ biến.
- Bị giới hạn về mặt tài nguyên, hoạt động dưới các ràng buộc về mặt thời gian, phần cứng, năng lượng,...
- Tương tác với thế giới bên ngoài bằng nhiều phương pháp như cảm nhận môi trường bằng các hệ thống cảm biến, tác động trở lại môi trường bằng các thiết bị thực thi. Giao diện người dùng có thể là đèn LED, nút nhấn, màn hình LCD cảm ứng, hoặc có thể không có giao diện.
- Cần hoạt động chính xác và ổn định trong thời gian dài trong các môi trường được xác định trước. Một khi xảy ra lỗi sẽ gây ra những hậu quả nghiêm trọng.

Một số hệ thống nhúng trong thực tế:

- Ô tô hiện đại là một hệ thống bao gồm nhiều thiết bị nhúng (đôi khi lên tới hàng trăm) được thiết kế để thực hiện các nhiệm vụ khác nhau trong xe. Một số phục vụ các tính năng cơ bản để điều khiển xe và một số khác cung cấp các chức năng truyền thông giải trí cho người dùng. Một số thiết bị nhúng khác trong xe đảm nhiệm các nhiệm vụ bao gồm kiểm soát hành trình, cảm biến vật cản, hệ thống định vị, hệ thống hỗ trợ người lái, hệ thống đảm bảo an toàn,...
- Các hệ thống máy công nghiệp có thể chứa các hệ thống nhúng, như cảm biến bản thân chúng có thể là các hệ thống nhúng. Máy công nghiệp thường có hệ thống tự động hóa nhúng thực hiện các chức năng giám sát, điều khiển cụ thể.

- Các thiết bị y tế có thể chứa các hệ thống nhúng như cảm biến và cơ chế điều khiển. Thiết bị y tế phải có giao diện thân thiện với người dùng và hoạt động chính xác, ổn định để không gây ảnh hưởng tới sức khỏe con người.
- Ngoài ra, hệ thống nhúng có thể xuất hiện khắp mọi nơi trong cuộc sống từ trong các thiết bị gia dụng như máy lạnh, tivi, lò vi sóng... ; trong các hệ thống quan trắc môi trường; hay trong các hệ thống phức tạp như điện thoại, máy tính,...

1.2.2 Internet of Things (IoT)

Internet of Things (Internet vạn vật) đề cập tới một mạng lưới mà trong đó các hệ thống thông minh có thể kết nối với và chia sẻ dữ liệu thông qua internet. Các thiết bị này có thể là một hệ thống nhúng bao gồm các cảm biến, thiết bị điều khiển, phần mềm, máy chủ,...[?]

Tiềm năng của IoT là rất lớn, tác động của nó đã được nhận thấy trong nhiều ngành công nghiệp, bao gồm sản xuất, vận tải, chuỗi cung ứng, y tế và nông nghiệp. Khi số lượng thiết bị kết nối internet tiếp tục tăng, IoT có thể sẽ đóng vai trò ngày càng quan trọng trong việc định hình thế giới của chúng ta và thay đổi cách chúng ta sống, làm việc và tương tác với nhau.

Một ví dụ đã áp dụng thành công IoT là Amazon. Các thiết bị IoT được nhúng trong hàng hóa giúp cải thiện khả năng quản lý kho hàng và truy xuất nguồn gốc trong thời gian thực. Bằng cách có những thông tin bổ ích như vậy, các tổ chức có thể tối ưu hóa toàn bộ quy trình cung ứng sản phẩm. Tính minh bạch này cũng giúp cải thiện sự hài lòng của khách hàng.

Một ví dụ khác trong lĩnh vực thiết bị đeo, các hãng Apple, Samsung, Adidas, Motorola đã dẫn đầu trong việc ứng dụng IoT lên chiếc đồng hồ thông minh. Ngoài các tính năng cơ bản như một chiếc đồng hồ thông thường, các loại đồng hồ thông minh hiện nay có thể hoạt động như một chiếc điện thoại thông minh, cho phép kết nối Internet và đặc biệt là khả năng theo dõi các thông số sức khỏe của người sử dụng và đưa ra các cảnh báo cho người sử dụng cũng như người thân nhờ có mạng lưới IoT.

1.2.3 Nhu cầu về nhân lực

Theo báo cáo từ Research and Markets, quy mô thị trường IoT tại Việt Nam ước đạt hơn 3,7 tỷ USD vào năm 2023, dự kiến cán mốc hơn 15,2 tỷ USD vào năm 2030 với tốc độ tăng trưởng kép hàng năm (CAGR) khoảng 22,1% trong giai đoạn 2023-2030[?].

Trong khi đó, theo số liệu từ Statista, doanh thu trên thị trường IoT tại Việt Nam được dự đoán sẽ đạt mốc 6,23 tỷ USD vào năm 2023, tăng lên so với mức 4,94 tỷ USD vào năm 2022 và 3,79 tỷ USD vào năm 2021. Trong đó, thị trường quan trọng nhất đối với lĩnh vực IoT tại Việt Nam là ngành ô tô, với doanh thu ước tính đối với lĩnh vực IoT cho ngành ô tô trong năm 2023 rơi vào khoảng 2,18 tỷ USD.

Doanh thu từ lĩnh vực IoT tại Việt Nam dự kiến sẽ tăng trưởng với tốc độ tăng trưởng kép hàng năm là 16,04% trong giai đoạn 2023 – 2028, qua đó ước đạt tổng doanh thu 13,11 tỷ USD vào năm 2028. Trên toàn cầu, phần lớn doanh thu từ thị trường IoT sẽ được tạo ra ở Mỹ (172,3 tỷ USD vào năm 2023).

Theo ông Nguyễn Thiện Nghĩa, Phó cục trưởng phụ trách Cục Công nghiệp thông tin và truyền thông (Bộ TT-TT), quy mô thị trường IoT Việt Nam đã đạt hơn 2 tỉ USD vào năm 2019, dự kiến có thể đạt 7 tỉ USD vào năm 2025.

Tuy nhiên, việc triển khai IoT còn đối mặt với nhiều thách thức như nhân lực, hạ tầng kỹ thuật yếu, chi phí đầu tư cao và an ninh thông tin.

Để bắt kịp với sự bùng nổ của IoT, thị phần này cần sự đáp ứng về nguồn nhân lực chất lượng cao trong lĩnh vực hệ thống nhúng và IoT. Thế nhưng, vì đây là lĩnh vực mới, sự hiểu biết ban đầu của sinh viên về lĩnh vực này không nhiều.Thêm vào đó, để có thể nắm vững các kỹ thuật trong lĩnh vực này thì sinh viên cần có những hạ tầng để hỗ trợ cho việc nghiên cứu và thực hành. Vì những lý do trên, cơn "khát" nhân lực trong lĩnh vực này vẫn chưa thể được giải quyết.

1.2.4 Khảo sát Kit thí nghiệm trên thị trường

Để đáp ứng nhu cầu học về lập trình nhúng, trên thị trường hiện nay đã xuất hiện rất nhiều Kit thí nghiệm. Trong đó Arduino là một nền tảng rất phổ biến và

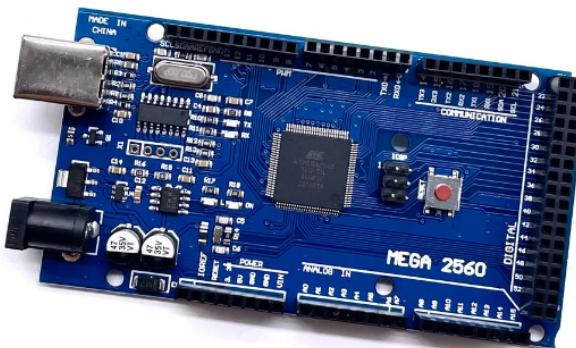
được tin tưởng lựa chọn. Trong đó phổ biến nhất phải kể đến Kit Arduino Uno.



Hình 1.1: Arduino Uno

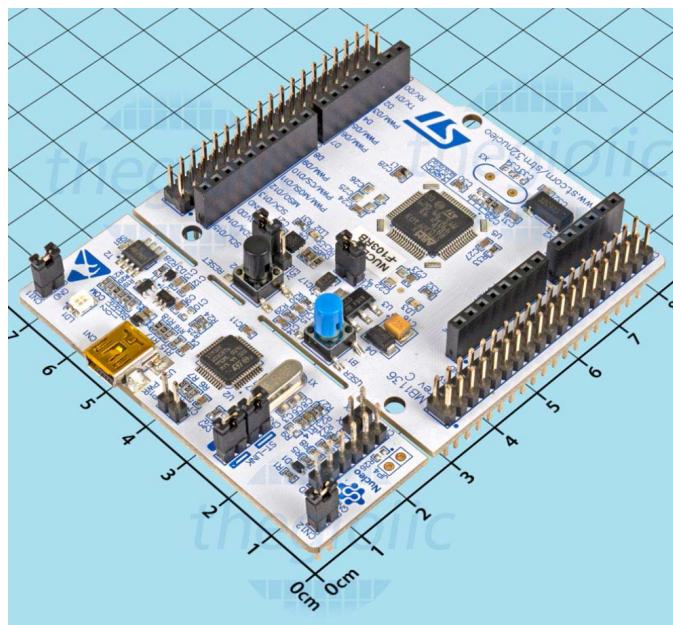
Kit Arduino Uno sử dụng vi điều khiển 8-bit ATmega328 với tần số tối đa là 16Mhz. Đối với các Kit Arduino, nên tảng lập trình phổ biến được sử dụng là Arduino IDE. Đây là nền tảng mã nguồn mở giúp người dùng có thể xây dựng các chương trình nhúng và nạp xuống Kit thí nghiệm. Arduino đã tạo ra một cộng đồng to lớn giúp đóng góp đa dạng về mặt thư viện cho người dùng. Các thư viện này giúp những người học chưa có nhiều kiến thức chuyên sâu về lập trình nhúng cũng có thể dễ dàng lập trình và sử dụng.

Một lựa chọn khác đối với dòng Kit này đó chính là Arduino Mega, sử dụng vi điều khiển ATmega2560. So với Arduino Uno thì Arduino Mega cung cấp số lượng chân lớn hơn, cũng như dung lượng các bộ nhớ Flash, SRAM, Eeprom nhiều hơn.



Hình 1.2: Arduino Mega

Ngoài Arduino, các dòng Kit thí nghiệm sử dụng vi điều khiển 32-bit STM32 cũng dần trở nên phổ biến. Các Kit STM32 sẽ có hiệu năng cao hơn và phù hợp hơn với những học viên tìm hiểu chuyên sâu về vi điều khiển cũng như hệ thống nhúng. Có nhiều công cụ lập trình cho vi điều khiển STM32, một trong số đó là STM32Cube IDE, do chính nhà sản xuất ST cung cấp. Công cụ này vừa hỗ trợ lập trình, vừa hỗ trợ chức năng cho phép người dùng thiết lập các cấu hình thông qua UI sau đó tự động chuyển đổi thành mã nguồn. Một đại diện cho dòng Kit thí nghiệm này là Kit Nucleo STM32F103RB.



Hình 1.3: Board NUCLEO STM32F103RB

Một Kit khác cũng được sử dụng phổ biến là STM32F407 Discovery Kit. Đây là một sự lựa chọn mạnh mẽ với vi xử lý hiệu năng cao, cung cấp nhiều chân I/O cũng như nhiều tính năng nâng cao. Ngoài ra trên Kit còn tích hợp các đèn led, cảm biến gia tốc, bộ xử lý âm thanh,...



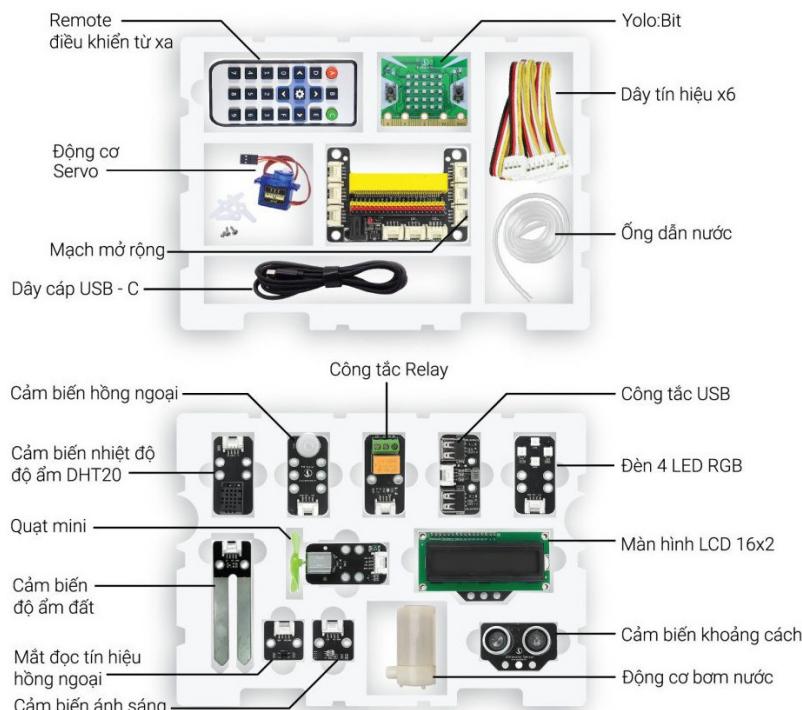
Hình 1.4: Kit STM32F407 Discovery

Các sản phẩm nói trên đều có lợi thế là nhỏ gọn, có thể kết hợp với nhiều thiết bị ngoại vi bên ngoài tùy theo mong muốn của người sử dụng. Ngoài ra đi kèm với các Kit thí nghiệm là một cộng đồng đông đảo người sử dụng và đóng góp, từ đó thuận lợi để người mới tiếp xúc có thể học tập. Tuy nhiên, nhược điểm của các Kit thí nghiệm nói trên là sẽ phải kết nối nhiều ngoại vi thông qua các chân header nếu muốn xây dựng thử nghiệm một hệ thống nhúng hoặc IoT hoàn chỉnh. Điều này sẽ gây khó khăn cho người sử dụng vì việc kết nối nhiều ngoại vi thông qua các header bằng dây bus có thể khiến tín hiệu không ổn định, dẫn đến khó khăn để xây dựng một chương trình nhúng hoàn chỉnh.

1.2.5 So sánh Kit thí nghiệm

Trong lĩnh vực vi điều khiển và hệ thống nhúng, các bộ kit thí nghiệm được chia thành hai loại chính: kit thí nghiệm tách rời và kit thí nghiệm tích hợp module

1. Kit thí nghiệm tách rời



Hình 1.5: Mạch tách rời module Yolobit

- Thành phần: Kit thí nghiệm tách rời bao gồm các thành phần riêng lẻ như vi điều khiển, mạch in (PCB), cảm biến, linh kiện điện tử (diện trở, tụ điện, transistor, v.v.), và các thiết bị đầu vào/đầu ra khác. Mỗi thành

phần có thể được sử dụng độc lập hoặc kết hợp với nhau để thực hiện các thí nghiệm khác nhau.

- Linh hoạt: Do các thành phần được cung cấp riêng lẻ, người sử dụng có thể linh hoạt trong việc lắp ráp và cấu hình các thí nghiệm theo nhu cầu cụ thể. Điều này cho phép người dùng tạo ra các thiết kế tùy chỉnh và học sâu hơn về cách từng thành phần hoạt động và tương tác.
- Giáo dục: Kit thí nghiệm tách rời rất hữu ích trong môi trường giáo dục vì chúng giúp sinh viên hiểu rõ hơn về cấu trúc và chức năng của từng phần tử trong hệ thống nhúng.
- Ví dụ: Các bộ kit thí nghiệm với các vi điều khiển phổ biến như ATmega328P (Arduino), PIC, hoặc STM32, nơi người dùng phải tự lắp ráp các mạch điện từ các linh kiện riêng lẻ.

2. Kit thí nghiệm tích hợp module

- Thành phần: Kit thí nghiệm tích hợp module bao gồm các module đã được tích hợp sẵn, thường có giao diện chuẩn để kết nối với nhau. Mỗi module có thể thực hiện một chức năng cụ thể như cảm biến, giao tiếp mạng, hiển thị, điều khiển động cơ, v.v.
- Tiện lợi: Các kit này thường dễ sử dụng hơn và nhanh chóng để lắp ráp, vì các module đã được thiết kế để làm việc cùng nhau mà không cần nhiều cấu hình. Điều này giúp tiết kiệm thời gian và giảm thiểu lỗi trong quá trình lắp ráp.
- Hiệu quả: Do tính chất tích hợp, các kit này giúp tiết kiệm thời gian và đảm bảo độ chính xác cao trong việc triển khai các thí nghiệm và dự án.
- Ví dụ: Các bộ kit Discovery hoặc Raspberry Pi, nơi các module cảm biến, module hiển thị, và các module khác có thể dễ dàng kết nối và lập trình để thực hiện các thí nghiệm và dự án nhúng khác nhau.

Từ đó có thể thấy, Kit thí nghiệm tách rời thích hợp cho những ai muốn tìm hiểu sâu về từng thành phần trong hệ thống nhúng và có nhu cầu tùy chỉnh cao trong thí nghiệm. Đây là lựa chọn tốt cho việc học tập và nghiên cứu cơ bản về vi điều khiển và các linh kiện điện tử. Trong khi đó, Kit thí nghiệm tích hợp module thích hợp cho những ai cần sự tiện lợi, dễ dàng lắp ráp và triển khai nhanh chóng các thí nghiệm và dự án nhúng. Đây là lựa chọn tốt cho việc phát triển ứng dụng nhanh chóng và các dự án có yêu cầu cụ thể về thời gian. Với những lý do đó, nhóm đã chọn kit thí nghiệm tích hợp module để phụ thuộc cho mục tiêu học tập và kiểm thử trong các dự án thực tế một cách nhanh chóng.

1.2.6 Khảo sát server IoT

Nền tảng IoT (IoT Platform) là công nghệ quan trọng trong việc quản lý, tự động hóa và thu thập dữ liệu từ các thiết bị, cảm biến và mạng lưới trong một hệ thống. Với nhiều tùy chọn kết nối khác nhau, nền tảng IoT đảm bảo tính bảo mật và hiệu quả trong việc vận hành các hệ thống IoT. Đây là thành phần không thể thiếu trong bất kỳ hệ thống IoT nào, giúp giảm thiểu rủi ro, giảm chi phí phát triển và đẩy nhanh thời gian triển khai, từ đó nâng cao khả năng cạnh tranh và tăng lợi nhuận cho doanh nghiệp.

Nền tảng IoT ngày càng được chú trọng phát triển với nhiều tính năng hiện đại, tích hợp trên cả phần mềm và phần cứng. Những tính năng này bao gồm giao diện người dùng thân thiện, khả năng xử lý và phân tích dữ liệu trên nhiều thiết bị, và triển khai dựa trên công nghệ đám mây. Những cái tiến này giúp người dùng dễ dàng hơn trong việc điều hành và quản lý các hệ thống IoT phức tạp. Một nền tảng IoT thường bao gồm các thành phần chính sau:

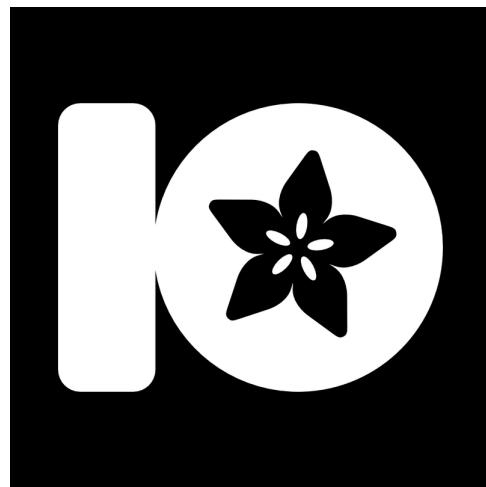
- **Thiết bị:** Bao gồm các cảm biến, máy móc hoặc các thiết bị khác có chức năng truyền hoặc nhận dữ liệu, thu thập thông tin từ môi trường xung quanh. Các thiết bị này là yếu tố kết nối chính trong hệ thống IoT.
- **Phương thức kết nối:** Được lựa chọn dựa trên thông số kỹ thuật của các thiết bị và yêu cầu của server hoặc đám mây. Các phương thức kết nối phổ biến bao gồm Wi-Fi, Bluetooth, Zigbee, và mạng di động.
- **Bộ phận xử lý dữ liệu:** Thường là server hoặc dịch vụ đám mây có nhiệm vụ nhận dữ liệu từ các thiết bị, sau đó phân tích và đưa ra các kế hoạch hành động. Quá trình xử lý dữ liệu này là yếu tố then chốt giúp tối ưu hóa hiệu suất của hệ thống.
- **Giao diện:** Là nơi người dùng có thể quan sát và quản lý toàn bộ quá trình vận hành của hệ thống. Giao diện người dùng cung cấp khả năng tương tác với dữ liệu, giúp người dùng dễ dàng theo dõi và điều chỉnh các thông số cần thiết.

Hiện nay, có nhiều nền tảng IoT được sử dụng rộng rãi nhờ vào tính năng và giao diện thân thiện với người dùng. Một số nền tảng tiêu biểu bao gồm: Eclipse, Mainflux, DSA, Thingsboard, Adafruit. Trong phần tiếp theo, nhóm sẽ tập trung vào hai nền tảng IoT phổ biến và được đánh giá cao về giao diện người dùng:

Adafruit và Thingsboard.

Adafruit

Adafruit là một nền tảng đám mây linh hoạt. Chúng ta có thể dễ dàng kết nối với Adafruit qua Internet để lưu trữ và truy xuất dữ liệu với nhiều tính năng hữu ích. Adafruit giúp biến dữ liệu thành thông tin có giá trị, tập trung vào tính dễ sử dụng và cho phép kết nối dữ liệu một cách đơn giản mà không cần nhiều kỹ năng lập trình.



Adafruit cung cấp các thư viện người dùng tích hợp với API REST và MQTT. Hệ thống này được xây dựng trên nền tảng Ruby on Rails và Node.js. Những tính năng nổi bật của Adafruit bao gồm:

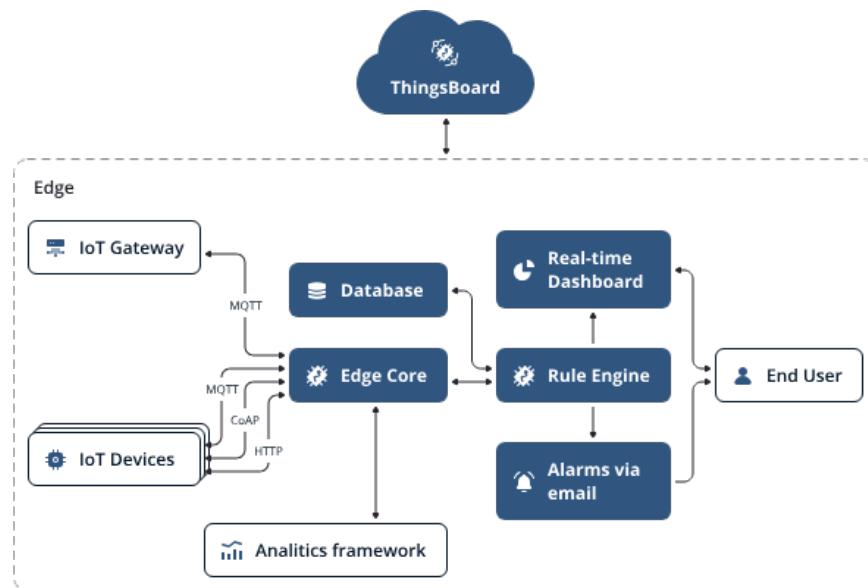
- Hiển thị dữ liệu trực tuyến theo thời gian thực.
- Kết nối các dự án với Internet, cho phép điều khiển động cơ và đọc dữ liệu cảm biến.
- Tích hợp các dự án với các dịch vụ web như Twitter, nguồn cấp dữ liệu RSS, dịch vụ thời tiết, v.v.
- Kết nối với các thiết bị hỗ trợ Internet khác.
- Đặc biệt, tất cả những tính năng này đều miễn phí với Adafruit.io.

Adafruit còn cung cấp một Bảng điều khiển (Dashboard) tích hợp, cho phép người dùng lập biểu đồ, vẽ đồ thị, đánh giá, ghi nhật ký và hiển thị dữ liệu. Chúng ta có thể truy cập trang tổng quan từ bất kỳ đâu trên thế giới.

ThingsBoard

ThingsBoard là một nền tảng Internet vạn vật (IoT) mạnh mẽ và mã nguồn mở, cho phép các nhà phát triển và doanh nghiệp tạo, triển khai và quản lý các ứng dụng IoT một cách hiệu quả. Được xây dựng như một giải pháp linh hoạt và có khả năng mở rộng, ThingsBoard hỗ trợ nhiều ứng dụng công nghiệp và thương mại, bao gồm nông nghiệp thông minh, năng lượng thông minh, thành phố thông minh và tự động hóa công nghiệp.

Các tính năng chính



- Quản lý thiết bị:** ThingsBoard cung cấp các khả năng quản lý thiết bị mạnh mẽ, cho phép người dùng kết nối, cấu hình và giám sát một loạt các thiết bị. Nó hỗ trợ các giao thức kết nối như MQTT, CoAP và HTTP, làm cho nó tương thích với nhiều thiết bị IoT. Người dùng có thể thực hiện cập nhật firmware qua mạng, quản lý cấu hình thiết bị và giám sát trạng thái thiết bị theo thời gian thực, đảm bảo vận hành và bảo trì thiết bị liền mạch.
- Thu thập và xử lý dữ liệu:** Một trong những điểm mạnh cốt lõi của ThingsBoard là khả năng thu thập và xử lý lượng lớn dữ liệu từ các thiết bị kết nối. Nền tảng này cung cấp các công cụ trực quan hóa dữ liệu tích hợp giúp người dùng tạo bảng điều khiển và tiện ích để hiển thị dữ liệu thời gian thực và xu hướng lịch sử. Với công cụ quy tắc của ThingsBoard, người dùng có thể định nghĩa các quy trình xử lý dữ liệu phức tạp, kích hoạt cảnh báo và thực hiện các hành động tùy chỉnh dựa trên các điều kiện cụ thể, nâng cao tự động hóa và trí thông minh của các ứng dụng IoT.

3. **Bảo mật và khả năng mở rộng:** Bảo mật là điều quan trọng trong bất kỳ triển khai IoT nào, và ThingsBoard giải quyết điều này với các cơ chế bảo mật mạnh mẽ, bao gồm xác thực thiết bị, mã hóa dữ liệu và kiểm soát truy cập dựa trên vai trò. Những tính năng này đảm bảo dữ liệu được truyền tải an toàn và chỉ truy cập bởi những người được ủy quyền.Thêm vào đó, ThingsBoard được thiết kế để mở rộng ngang, cho phép nó xử lý hàng triệu thiết bị và xử lý lượng lớn dữ liệu mà không ảnh hưởng đến hiệu suất.
4. **Tích hợp và tương tác:** ThingsBoard hỗ trợ tích hợp với nhiều hệ thống và dịch vụ bên ngoài, cho phép tương tác liền mạch trong hệ sinh thái IoT. Nó có thể kết nối với các dịch vụ đám mây bên thứ ba, cơ sở dữ liệu và các ứng dụng bên ngoài thông qua REST API và các broker MQTT. Sự linh hoạt này cho phép doanh nghiệp tận dụng hạ tầng hiện có và mở rộng chức năng của các giải pháp IoT của họ.
5. **Bảng Điều Khiển (Dashboard):** ThingsBoard cho phép người dùng tạo các bảng điều khiển (dashboard) phong phú để hiển thị dữ liệu và điều khiển thiết bị từ xa trong thời gian thực. Người dùng có thể xây dựng các bảng điều khiển tùy chỉnh cho dự án của mình, ví dụ như một bảng điều khiển cho dự án nông trại thông minh để hiển thị trực quan dữ liệu sản lượng, điều kiện thời tiết và các yếu tố sản xuất nông nghiệp khác. Điều này giúp người dùng dễ dàng giám sát và quản lý các thiết bị và dữ liệu IoT.

1.3 Đề xuất yêu cầu của sản phẩm

Từ những vấn đề nêu trên, nhóm đề xuất xây dựng một nền tảng, một công cụ hỗ trợ sinh viên trong việc học tập, tìm hiểu về hệ thống nhúng và IoT bao gồm Kit thí nghiệm và các tài liệu, cũng như dịch vụ máy chủ đi kèm. Kit thí nghiệm hướng tới sẽ thiết kế các khối chức năng cơ bản và thông dụng để sinh viên có thể sử dụng trong suốt quá trình học tập ở bậc đại học. Để đáp ứng nhu cầu về giáo dục hệ thống nhúng và IoT, nhóm đặt ra những yêu cầu sau cho sản phẩm:

1. Yêu cầu tính năng:

- Kit thí nghiệm cung cấp đủ các khối giúp học tập kiến thức cơ bản về vi điều khiển, hệ thống nhúng, IoT.
- Kit thí nghiệm cung cấp các chuẩn giao tiếp RS232, RS485, CAN.

- Kit thí nghiệm cung cấp khả năng kết nối Internet thông qua Wifi, Ethernet, 4G/LTE.
- Cung cấp bộ tài liệu học tập dựa trên các khối của Kit thí nghiệm.
- Cung cấp bộ thư viện mẫu của Kit thí nghiệm.
- Cung cấp máy chủ Iot hỗ trợ HTTP.
- Máy chủ Iot hỗ trợ các tính năng quản lý tài khoản, quản lý thiết bị, quản lý dự án, quản lý dữ liệu.

2. Yêu cầu phi tính năng:

- Kích thước Kit thí nghiệm tối đa 20cm x 20cm.
- Kit thí nghiệm có độ bền ít nhất 4 năm.
- Kit thí nghiệm có khả năng xếp chồng tối đa 5 bộ lên nhau vững chắc.
- Bộ tài liệu và thư viện hướng dẫn cho phép người dùng có khả năng sử dụng Kit thí nghiệm trong vòng 72 giờ.
- Giá thành sản xuất Kit thí nghiệm dưới 1,500,000 VND.

1.4 Cấu trúc của đề tài

Báo cáo của nhóm gồm có 6 chương, các chương được sắp xếp theo thứ tự nối tiếp, liền mạch nhau nhằm mục đích trình bày hệ thống nhóm triển khai một cách khách quan và tường minh nhất có thể. Cụ thể như sau:

Chương 1	Giới thiệu tổng quan về đề tài, đề xuất các chức năng sẽ hiện thực, trình bày mục tiêu, phạm vi và các giai đoạn thực hiện đề tài. Với những kiến thức tổng quan và các chức năng đề xuất nhóm sẽ bắt đầu đi sâu hơn về ý tưởng thiết kế. Khảo sát các kit trên thị trường, khảo sát server phổ biến hiện nay.
Chương 2	Trình bày các cơ sở lý thuyết được sử dụng trong đề tài như kiến thức về vi điều khiển, truyền thông, các chuẩn giao tiếp,... Trình bày về các nội dung lý thuyết, các bài toán kinh điển hướng tới trong các bài thí nghiệm. Trình bày các lý thuyết về cơ sở dữ liệu, backend.
Chương 3	Trình bày về các tính toán và thiết kế hệ thống: nêu sơ đồ nguyên lý toàn mạch, và thiết kế mô hình phần cứng, lựa chọn linh kiện. Trình bày kiến trúc xây dựng trong các bài thí nghiệm. Các sơ đồ thực thể, lược đồ và workflow được đề xuất nhằm đảm bảo tính nhất quán và dễ sử dụng cho người dùng.
Chương 4	Trình bày quá trình thi công KIT thí nghiệm, kết quả mô phỏng và thực tế, đóng gói Kit. Trình bày quá trình xây dựng bộ bài thí nghiệm. Trình bày hệ thống server backend, tài liệu API mô tả chi tiết các endpoint có sẵn, bao gồm các phương thức HTTP (GET, POST, PUT, DELETE), các tham số yêu cầu và phản hồi mẫu.
Chương 5	Kiểm thử và đánh giá về quá trình hiện thực, chỉnh sửa Kit thí nghiệm và bộ bài thí nghiệm đi kèm. Kiểm thử về logic và tải của server backend.
Chương 6	Tổng kết quá trình thực hiện đề tài với các kết quả đạt được, những đóng góp về KIT thí nghiệm, bài thí nghiệm cũng như server từ đó hoàn thiện các hạn chế và hướng phát triển của dự án trong tương lai.

Bảng 1.1: Bảng cấu trúc đề tài

CHƯƠNG 2

TỔNG QUAN TÀI LIỆU

Trong chương này, nhóm nêu những giải pháp khoa học đã được nhóm sử dụng để giải quyết vấn đề, các cơ sở lý thuyết về MCU và các kiến thức về module, những kiến thức này hỗ trợ nhóm trong quá trình nghiên cứu cũng như góp phần vào việc thiết kế mạch, xây dựng bộ bài thí nghiệm, xây dựng server. Cụ thể, bao gồm các kiến thức về vi điều khiển, thiết kế mạch, các kiến thức về truyền thông, các chuẩn giao tiếp,... Đồng thời, trong chương này cũng sẽ trình bày về các nội dung lý thuyết, các bài toán kinh điển hướng tới trong các bài thí nghiệm. Trình bày các lý thuyết về cơ sở dữ liệu, backend.

2.1 Khối kiến thức về lập trình hệ thống nhưng sử dụng vi điều khiển

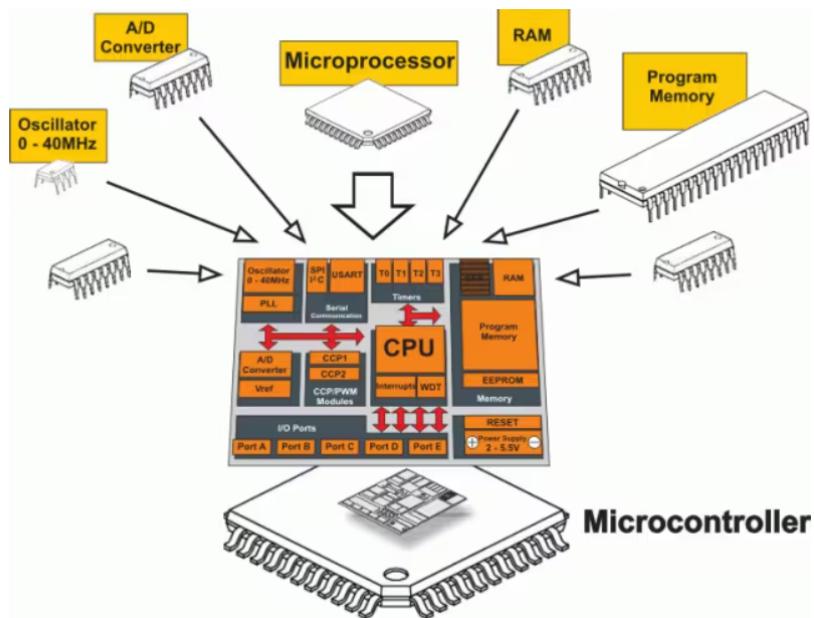
2.1.1 Vi điều khiển (MCU)

Trong hệ thống nhúng, có rất nhiều hình thái bộ xử lý được sử dụng. Chúng bao gồm từ rất nhỏ, chậm, rẻ tiền, tiêu thụ điện năng thấp, cho đến các thiết bị có mục đích đặc biệt, hiệu suất cao. Trong đó, Vi điều khiển là một lựa chọn phổ biến để làm bộ xử lý cho hệ thống nhúng.

Theo thống kê, hơn một nửa số CPU được bán trên toàn thế giới đều là các Vi điều khiển. Các Vi điều khiển thường được dùng trong các ứng dụng yêu cầu dung lượng bộ nhớ nhỏ và xử lý các tác vụ điều khiển đơn giản (so với các ứng dụng đòi hỏi nhiều hiệu năng). Chúng có thể tiêu thụ một lượng năng lượng cực kỳ nhỏ và thường bao gồm chế độ ngủ giúp giảm mức tiêu thụ điện năng xuống mức nanowatt.

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

Vi điều khiển đóng một vai trò quan trọng trong hệ thống nhúng bằng cách cung cấp khả năng xử lý và điều khiển cho các chức năng cụ thể của hệ thống. Một vi điều khiển thường bao gồm một bộ xử lý trung tâm (CPU), bộ nhớ truy cập ngẫu nhiên (RAM), bộ nhớ chỉ đọc (ROM), cổng I/O, bộ đếm thời gian, bộ chuyển đổi ADC, bộ chuyển đổi DAC, cổng giao tiếp nối tiếp.



Hình 2.1: Tổng quan MCU (Nguồn: Max Embedded)

2.1.2 Lõi kiến trúc ARM

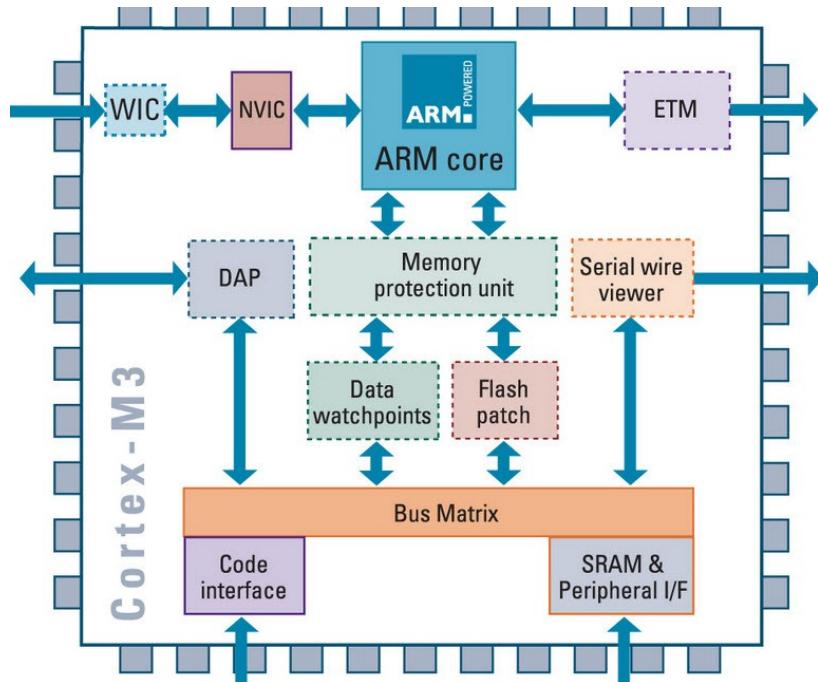
ARM[?] là viết tắt của Advanced RISC Machine là một trong những lõi xử lý được ứng dụng rộng rãi nhất hiện nay. Arm được giới thiệu lần đầu tiên tại Đại học Cambridge vào năm 1978. Các bộ xử lý này được sử dụng đặc biệt trong các thiết bị di động như máy ảnh kỹ thuật số, điện thoại di động, các module mạng gia đình, các công nghệ truyền thông không dây và các hệ thống nhúng khác vì các lợi ích như tiêu thụ điện năng thấp, hiệu suất hợp lý, v.v. Trong các sản phẩm kỹ thuật số tiên tiến, ARM đóng vai trò như một trái tim.

Các dòng Arm nổi bật ở thời điểm hiện tại là Cortex, bao gồm ba họ kiến trúc: Cortex-A, Cortex-R và Cortex-M[?].

- Họ Cortex-A là bộ xử lý ứng dụng có hỗ trợ hệ điều hành và ứng dụng của bên thứ ba. Điều này có nghĩa là chúng có thể xuất hiện trên các điện thoại thông minh có nhiều ứng dụng hoặc thậm chí cả máy chủ. Cortex-A có

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

các loại 32-bit (Armv7-A) và 64-bit (Armv8-A). Raspberry Pi 3, sử dụng lõi Cortex-A53 Arm, triển khai kiến trúc Armv8-A.



Hình 2.2: Mô hình kiến trúc arm

- Họ Cortex-R, kiến trúc Armv7-R và Armv8-R, được tối ưu hóa cho các ứng dụng thời gian thực hiệu suất cao. Những bộ xử lý này có khả năng chịu lõi tốt hơn và hoạt động tốt trong các ứng dụng chú trọng về tính an toàn như thiết bị y tế, hệ thống điều khiển công nghiệp và hệ thống thiết bị an toàn.
- Họ Cortex-M là một nhóm các lõi Arm 32-bit có chi phí thấp, thu nhỏ bao gồm Armv6-M, Armv7-M và Armv8-M. Họ bộ xử lý này hướng tới các ứng dụng vi điều khiển, ASIC, FPGA và SoC. Trong cuộc cạnh tranh trực tiếp với thị trường MCU 8 bit, lõi Cortex-M 32 bit được nhúng vào SoC lớn hơn có thể là một sự kết hợp cực kỳ mạnh mẽ. Cortex-M cũng đã tìm được chỗ đứng trong các ứng dụng IoT với các nền tảng như MCU.

Ưu điểm của vi điều khiển lõi Arm (Cortex-M):

- Tiêu thụ điện năng thấp: Vì bộ vi điều khiển ARM được thiết kế để sử dụng ít năng lượng nên chúng rất lý tưởng cho các sản phẩm chạy bằng pin. Để giảm mức sử dụng năng lượng, họ sử dụng các kĩ thuật tiết kiệm năng lượng bao gồm clock gating và power.
- Sức mạnh xử lý cao: Bộ vi điều khiển ARM nổi tiếng vì có nhiều sức mạnh xử lý, điều này khiến chúng trở nên hoàn hảo cho các ứng dụng hiệu suất cao.

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

Chúng có khả năng xử lý lượng lớn dữ liệu một cách nhanh chóng và hiệu quả.

- Tính linh hoạt: Bộ vi điều khiển ARM rất linh hoạt và có thể được lập trình để thực hiện nhiều chức năng khác nhau. I2C, SPI, UART, USB, Ethernet và CAN là một số giao diện và giao thức truyền thông mà chúng hỗ trợ.

2.1.3 Hệ thống hướng sự kiện

Hướng sự kiện (event-driven) trong lập trình vi điều khiển là một phương pháp lập trình giúp xử lý các sự kiện xảy ra tại các thời điểm cụ thể, thay vì việc thực hiện các tác vụ theo một luồng thực thi tuần tự. Trong ngữ cảnh vi điều khiển, các sự kiện có thể bao gồm ngắn từ cảm biến, tín hiệu từ các giao tiếp ngoại vi, hoặc các tín hiệu từ người dùng như nhấn nút.

Khi sử dụng lập trình Hướng sự kiện trong vi điều khiển, lập trình viên thường định nghĩa các hàm xử lý sự kiện (event handler) để xử lý các sự kiện cụ thể. Khi một sự kiện xảy ra, MCU sẽ gọi các hàm xử lý sự kiện tương ứng để thực hiện các hành động phù hợp. Điều này giúp tăng tính linh hoạt, phản ứng nhanh chóng và hiệu suất của hệ thống.

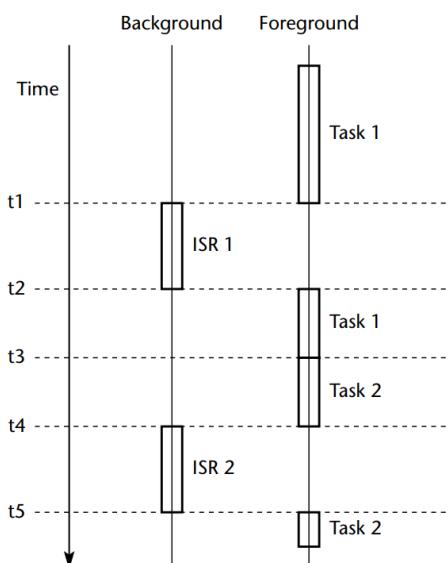
Trong vi điều khiển, lập trình Hướng sự kiện cho phép hệ thống phản ứng linh hoạt đối với các sự kiện bất ngờ và sắp xảy ra. Ví dụ, khi một ngắn từ cảm biến được kích hoạt, hệ thống có thể ngay lập tức chuyển đổi sang xử lý sự kiện này mà không cần phải đợi đến khi xử lý tuần tự đến phần này. Điều này cực kỳ hữu ích trong việc xử lý các tác vụ cần phản ứng nhanh chóng và không thể dự đoán trước.

Một ưu điểm lớn của lập trình Hướng sự kiện trong vi điều khiển là khả năng quản lý tác vụ một cách hiệu quả. Thay vì phải quản lý một luồng thực thi phức tạp, lập trình viên có thể tập trung vào viết các hàm xử lý sự kiện riêng biệt mà không lo lắng về sự chồng chéo và đồng bộ hóa các tác vụ.

Tuy nhiên, cũng cần lưu ý rằng việc sử dụng lập trình Hướng sự kiện cũng đòi hỏi tạo ra các thách thức liên quan đến quản lý bộ nhớ và đồng bộ hóa dữ liệu khi có nhiều sự kiện xảy ra đồng thời. Lập trình viên cần phải xem xét cẩn thận để đảm bảo tính nhất quán và an toàn trong việc xử lý các sự kiện bất đồng bộ.

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

Trong hệ thống nhúng, các hành vi hướng sự kiện được thực hiện thông qua cơ chế ngắn. Từ góc độ cấp thấp, ngắn là một cơ chế phần cứng được sử dụng để thông báo bộ xử lý rằng một 'sự kiện' đã diễn ra: những sự kiện đó có thể là sự kiện 'nội bộ' (chẳng hạn như tràn bộ đếm thời gian) hoặc các sự kiện 'bên ngoài' (chẳng hạn nhận được một byte dữ liệu từ cổng giao tiếp nối tiếp). Nhìn từ góc độ cấp cao, các ngắn cung cấp một cơ chế để tạo một ứng dụng đa nhiệm: đó là những ứng dụng dường như thực hiện nhiều hơn một nhiệm vụ tại một thời điểm bằng cách sử dụng một bộ xử lý duy nhất. Hình 2.3 thể hiện cơ chế xử lí ngắn của MCU.



Hình 2.3: Cơ chế xử lí ngắn

2.1.4 Hệ thống hướng thời gian

Hướng thời gian (time-driven) trong lập trình vi điều khiển là một phương pháp lập trình mà các tác vụ được thực hiện dựa trên thời gian thực hoặc theo các chu kỳ thời gian cố định. Kỹ thuật này được sử dụng rộng rãi trong vi điều khiển để lập lịch và quản lý các tác vụ cần thực thi theo chu kỳ nhất định, đảm bảo sự đồng nhất và dự đoán được trong việc thực hiện các chức năng cần thiết.

Trong phát triển phần mềm vi điều khiển, lập trình hướng thời gian đóng vai trò quan trọng trong việc xử lý các chức năng như lấy mẫu cảm biến, gửi dữ liệu qua mạng, kiểm soát thiết bị v.v. bằng cách thiết lập các hẹn giờ cụ thể để thực hiện các tác vụ này theo đúng chu kỳ. Điều này giúp hệ thống hoạt động một cách đồng nhất và dự đoán được, đồng thời giảm thiểu sự phụ thuộc vào các sự kiện

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

bất đồng bộ.

Một trong những lợi ích lớn nhất của lập trình hướng thời gian trong vi điều khiển là tính nhất quán và dự đoán trong việc thực hiện các chức năng. Ví dụ, trong các hệ thống điều khiển tự động, việc lập lịch các tác vụ như điều khiển động cơ, kiểm tra cảm biến, và gửi dữ liệu điều khiển, theo các chu kỳ thời gian cố định giúp đảm bảo tính ổn định và chính xác của hệ thống.

Tuy nhiên, cũng cần phải lưu ý rằng việc quản lý thời gian và đồng bộ hóa các tác vụ trong time-driven programming có thể trở thành một thách thức đối với việc lập trình và xử lý ngoại lệ. Đảm bảo tính đồng nhất của các chu kỳ thời gian và xử lý các sự kiện ngoại lệ một cách hiệu quả là những yếu tố phức tạp mà lập trình viên vi điều khiển cần phải xem xét khi sử dụng phương pháp này.

2.1.5 Máy trạng thái hữu hạn

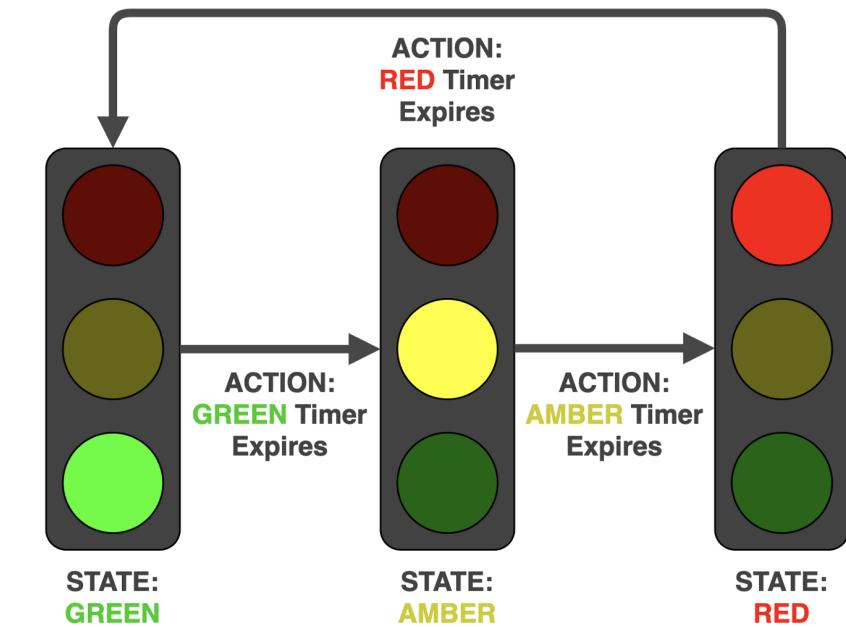
Finite State Machine (FSM) là một mô hình toán học được sử dụng rộng rãi trong lập trình vi điều khiển để mô tả hành vi của các hệ thống với số lượng hữu hạn trạng thái có thể. Trong ngữ cảnh vi điều khiển, FSM thường được sử dụng để điều khiển các thiết bị hoặc quyết định hành vi dựa trên trạng thái hiện tại của hệ thống và các sự kiện xảy ra.

FSM bao gồm một số trạng thái (state) có thể, các sự kiện (events) và các chuyển đổi trạng thái (transitions) giữa các trạng thái. Mỗi trạng thái đại diện cho một tình huống hoặc hành vi cụ thể của hệ thống. Khi một sự kiện xảy ra, hệ thống sẽ chuyển đổi từ trạng thái hiện tại sang trạng thái mới dựa trên các quy tắc xác định trước. Việc này giúp mô tả và quản lý các hành vi phức tạp một cách hiệu quả và dễ dàng hiểu.

Trong lập trình vi điều khiển, việc sử dụng FSM giúp quản lý và điều khiển các thiết bị hoặc hệ thống có các trạng thái khác nhau một cách dễ dàng. Ví dụ, FSM có thể được sử dụng để điều khiển một robot di động, trong đó các trạng thái có thể bao gồm "đi thẳng", "rẽ trái", "rẽ phải", "dừng lại" v.v. Khi nhận được các tín hiệu từ cảm biến, robot sẽ chuyển đổi trạng thái để điều khiển hành vi di chuyển theo các quy tắc đã được xác định trước.

2.1. Khối kiến thức về lập trình hệ thống nhúng sử dụng vi điều khiển

Một ưu điểm lớn của việc sử dụng FSM là tính dự đoán và tính nhất quán trong quản lý trạng thái của hệ thống. FSM giúp lập trình viên dễ dàng xác định và hiểu cấu trúc của hệ thống, từ đó tạo ra mã nguồn dễ bảo trì và mở rộng. Tuy nhiên, việc thiết kế FSM phù hợp và hiệu quả đòi hỏi sự tinh tế và kinh nghiệm của lập trình viên để đảm bảo tính linh hoạt và tái sử dụng.



Hình 2.4: Một ví dụ về FSM (Nguồn: xiaoyunyang)

2.1.6 Cơ chế DMA

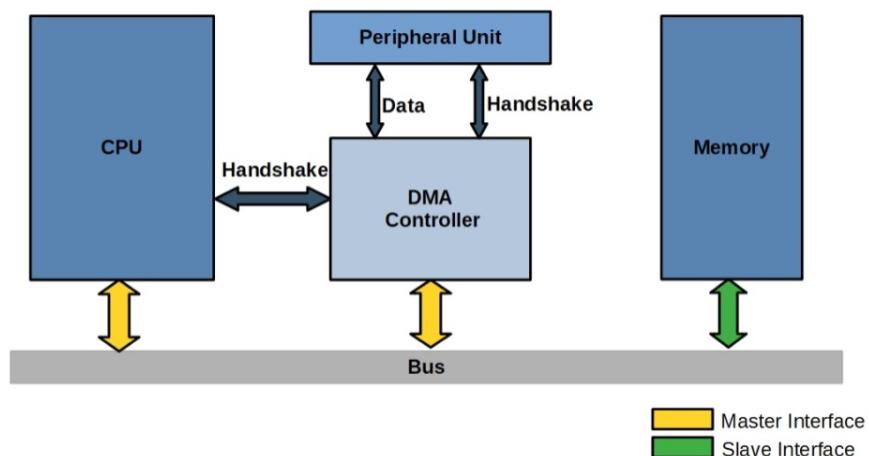
DMA (Direct Memory Access) là một tính năng quan trọng trong vi điều khiển, cho phép các thiết bị ngoại vi truy cập trực tiếp vào bộ nhớ hệ thống mà không cần sự can thiệp của CPU. Khi một thiết bị như card mạng, card âm thanh, hoặc bộ chuyển đổi dữ liệu cần truy cập dữ liệu trong bộ nhớ, DMA có thể được kích hoạt để quản lý quá trình truyền dữ liệu trực tiếp giữa bộ nhớ và thiết bị ngoại vi.

Việc sử dụng DMA mang lại hai lợi ích chính. Đầu tiên, nó giúp giảm tải cho CPU bởi vì CPU không cần phải tham gia trực tiếp trong quá trình truyền dữ liệu. Thay vào đó, DMA có thể quản lý việc truyền dữ liệu một cách độc lập. Lợi ích thứ hai là tăng hiệu suất hệ thống bởi vì việc truyền dữ liệu trực tiếp giữa bộ nhớ và thiết bị ngoại vi giúp giảm thời gian chờ đợi và tăng tốc độ truyền dữ liệu.

Các ứng dụng của DMA trong vi điều khiển rất đa dạng, từ việc truyền dữ liệu

từ bộ nhớ đến thiết bị ngoại vi (output), đọc dữ liệu từ thiết bị ngoại vi đến bộ nhớ (input), đến việc trao đổi dữ liệu giữa các thiết bị ngoại vi mà không cần sự can thiệp của CPU. DMA đặc biệt hữu ích trong việc xử lý dữ liệu lớn, truyền dữ liệu liên tục như âm thanh, video và mạng, và các ứng dụng đòi hỏi tốc độ và độ ổn định cao.

Việc sử dụng DMA đòi hỏi việc cấu hình phức tạp và quản lý tài nguyên một cách cẩn thận để tránh xung đột dữ liệu và lỗi hệ thống, nhưng với những lợi ích mà nó mang lại, DMA là một phần không thể thiếu trong nhiều hệ thống vi điều khiển hiện đại.



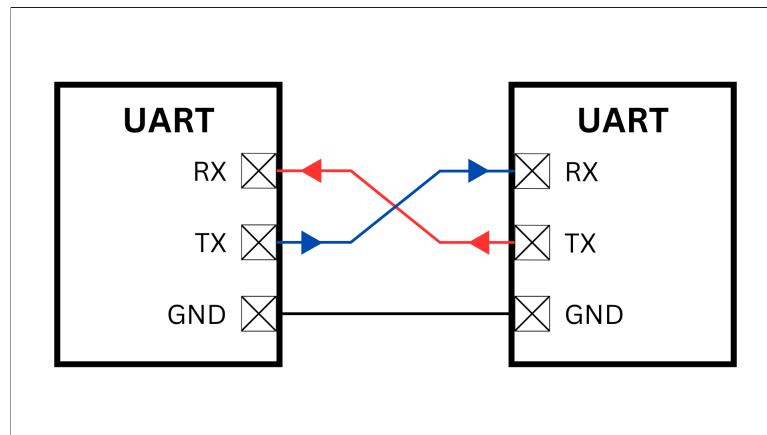
Hình 2.5: Sơ đồ khái niệm DMA (Nguồn: Open4Tech)

2.2 Kiến thức về các giao thức truyền thông logic

Trong phần này, nhóm sẽ đề cập đến các kiến thức về giao thức truyền thông được nhóm tìm hiểu để áp dụng trên mạch, cụ thể là UART, I2C, SPI. Đây là những giao thức truyền thông rất phổ biến mà những người tiếp cận Hệ thống nhúng và phát triển IoT đều sẽ tiếp cận và sử dụng thường xuyên, thậm chí là hầu hết tất cả các ứng dụng đều sẽ sử dụng đến. Những giao thức không chỉ là những phương tiện kết nối quan trọng mà còn đóng vai trò quyết định trong việc truyền thông giữa các linh kiện và thiết bị.

2.2.1 Giao tiếp UART

UART [?] (Universal Asynchronous Receiver/Transmitter) là một trong những giao thức truyền thông đơn giản nhất hiện có, nó là một loại giao tiếp bất đồng bộ. Dùng như tên gọi, nó vừa gửi vừa nhận dữ liệu. Trong giao tiếp UART, hai thiết bị UART giao tiếp trực tiếp với nhau. Nó xác định một bộ quy tắc để trao đổi dữ liệu nối tiếp giữa hai thiết bị. Trong uart sử dụng 2 dây để giao tiếp. Dây phát (TX) trên thiết bị 1 được kết nối với dây thu (RX) của thiết bị 2. Tương tự, dây phát (TX) trên thiết bị 2 được kết nối với dây thu (RX) của thiết bị 1. Nối đất (GND) dây là cần thiết để giữ cho cả hai thiết bị ở cùng một điện áp tham chiếu. Dây này hầu như luôn hiện diện trong mọi loại hình giao tiếp giữa các thiết bị.



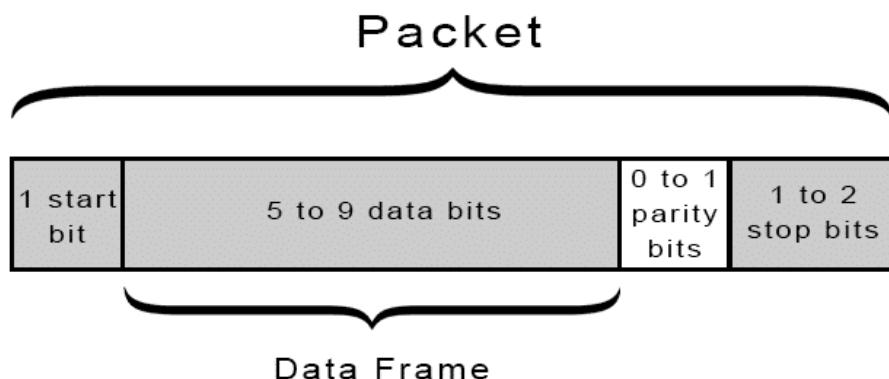
Hình 2.6: Sơ đồ giao tiếp giữa 2 thiết bị qua UART

Khi thiết bị UART nhận diện được bit khởi đầu, nó bắt đầu đọc bit theo một tần số cụ thể, được gọi là baudrate. Baudrate là chỉ số đo tốc độ truyền dữ liệu, thể hiện bằng bit/giây(bps). Cả hai thiết bị UART đều phải hoạt động ở cùng một tốc độ truyền. Sự khác biệt về tốc độ truyền giữa thiết bị truyền và thiết bị nhận chỉ có thể chênh lệch khoảng 10% [?]. Cả hai thiết bị UART cũng cần được cấu hình để truyền và nhận cùng một cấu trúc gói dữ liệu.

Wires Used	2
Maximum Speed	Any speed up to 115200 baud, usually 9600 baud
Synchronous or Asynchronous?	Asynchronous
Serial or Parallel?	Serial
Max # of Masters	1
Max # of Slaves	1

Hình 2.7: Các điểm cần lưu ý về UART

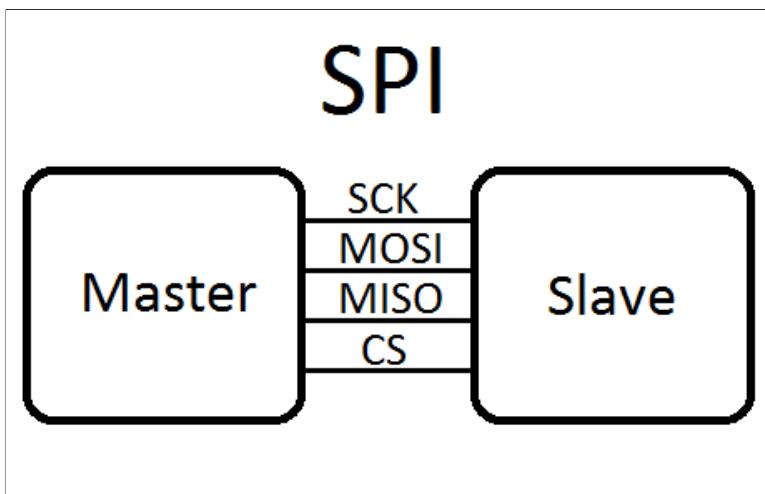
Dữ liệu được truyền trong giao tiếp UART được tổ chức thành các gói (Packets). Mỗi Packets chứa 1 bit Start, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), 1 bit Parity và 1 hoặc 2 bit Stop:



Hình 2.8: Dataframe của UART

2.2.2 Giao tiếp SPI

SPI[?](Serial Peripheral Interface) là giao tiếp thường được sử dụng trong máy tính và hệ thống nhúng để hỗ trợ giao tiếp khoảng cách ngắn giữa bộ vi điều khiển với các thiết bị ngoại vi. SPI là giao tiếp đồng bộ, bất cứ quá trình nào cũng đều được đồng bộ với xung clock sinh ra bởi thiết bị Master. Do đó, không cần phải lo lắng về tốc độ truyền dữ liệu.



Hình 2.9: Sơ đồ giao tiếp SPI

Hoạt động của giao tiếp SPI: sử dụng 4 đường giao tiếp nên đôi khi được gọi là chuẩn truyền thông “4 dây”. 4 đường đó là :

- SCK (Serial Clock): Thiết bị Master tạo xung tín hiệu SCK và cung cấp cho Slave. Xung này giữ nhịp cho giao tiếp SPI và truyền dữ liệu. Điều này giúp giảm lỗi và tăng tốc độ truyền.
- MISO (Master Input Slave Output): Tín hiệu tạo bởi thiết bị Slave và nhận bởi thiết bị Master. Đường MISO phải được kết nối giữa thiết bị Master và Slave.
- MOSI (Master Output Slave Input): Tín hiệu tạo bởi thiết bị Master và nhận bởi thiết bị Slave. Đường MOSI phải được kết nối giữa thiết bị Master và Slave.
- CS (Chip Select) hoặc SS(Slave Select): Chọn Slave để giao tiếp. Master kéo chân CS xuống mức 0 (Low) để chọn Slave. Vi điều khiển có thể tạo chân CS bằng cách cấu hình 1 chân GPIO chế độ Output.

Trong giao thức SPI, thiết bị Master (thường là vi điều khiển) kiểm soát và gửi lệnh cho thiết bị Slave (cảm biến, màn hình, chip nhớ). Hệ thống có thể đơn giản với một Master và một Slave, nhưng Master có thể kiểm soát nhiều hơn một Slave. Mỗi Slave chỉ phản hồi khi được Master yêu cầu, tạo nên môi trường linh hoạt trong việc giao tiếp giữa các thiết bị.

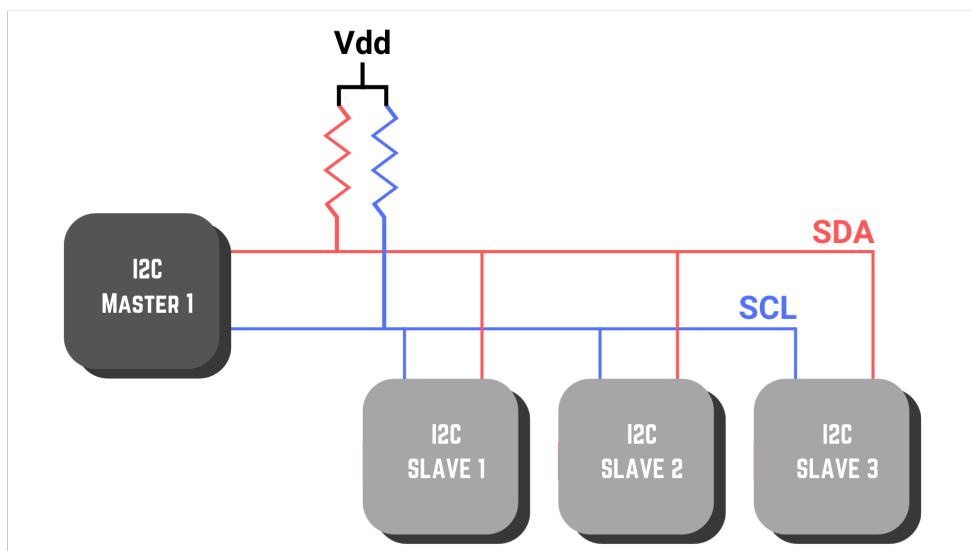
2.2.3 Giao tiếp I2C

I2C [?](Inter-Integrated Circuit) là một giao thức kết nối giao diện bus được tích hợp vào các thiết bị để giao tiếp nối tiếp. I2C là sự kết hợp các tính năng tốt nhất của SPI và UART. Giống như giao tiếp UART, I2C chỉ sử dụng hai dây để truyền dữ liệu giữa các thiết bị. Với I2C cho phép thể kết nối nhiều Slave với một Master duy nhất (như SPI) và cũng có thể có nhiều Master điều khiển một hoặc nhiều Slaver.

I2C thường được sử dụng để giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại vi điều khiển, cảm biến, EEPROM, DS3231 (RTC),...

I2C sử dụng 2 đường truyền tín hiệu:

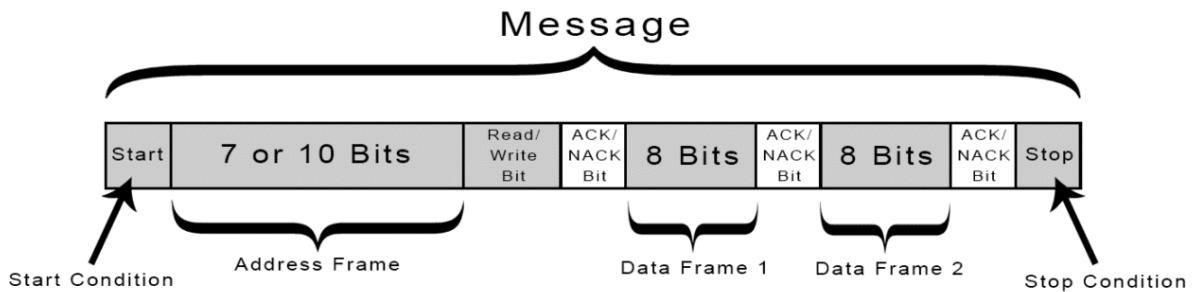
- SCL: Tạo xung nhịp đồng hồ do Master phát đi.
- SDA: Đường truyền nhận dữ liệu.



Hình 2.10: Kết nối I2C giữa Master và Slave

Giao tiếp I2C bao gồm quá trình truyền nhận dữ liệu giữa các thiết bị chủ (Master) và thiết bị phụ (Slave). Thiết bị Master là một vi điều khiển, có nhiệm vụ điều khiển đường tín hiệu SCL và gửi/nhận dữ liệu hoặc lệnh thông qua đường SDA đến các thiết bị khác. Thiết bị Slave thường là các IC, hoặc thậm chí là vi điều khiển. Hai đường bus SCL và SDA đều hoạt động ở chế độ Open Drain, tức là bất kỳ thiết bị nào kết nối với mạng I2C cũng chỉ có thể kéo 2 đường bus này xuống mức thấp (LOW), nhưng không thể kéo chúng lên mức cao.

Dataframe của I2C:



Hình 2.11: Dataframe của I2C

Quá trình truyền nhận dữ liệu:

- Start: Thiết bị Master sẽ gửi một xung Start bằng cách hạ lần lượt các dây SDA, SCL từ mức 1 xuống 0.
- Sau đó, Master sẽ gửi 7 bit địa chỉ đến Slave mà nó muốn giao tiếp, kèm theo bit Read/Write.
- Slave sẽ so sánh địa chỉ vật lý của nó với địa chỉ vừa nhận được. Nếu khớp, Slave sẽ xác nhận bằng cách hạ dây SDA xuống 0 và đặt bit ACK/NACK là '0'. Nếu không khớp, dây SDA và bit ACK/NACK mặc định sẽ là '1'.
- Thiết bị Master sẽ tiếp tục gửi hoặc nhận khung bit dữ liệu. Nếu Master gửi đến Slave, bit Read/Write sẽ ở mức 0. Ngược lại, nếu Master nhận dữ liệu, bit này sẽ ở mức 1.
- Nếu khung dữ liệu được truyền đi thành công, bit ACK/NACK sẽ được đặt về mức 0 để báo hiệu cho Master tiếp tục.
- Khi tất cả dữ liệu đã được gửi thành công đến Slave, Master sẽ phát tín hiệu Stop để thông báo cho các Slave rằng quá trình truyền đã kết thúc, bằng cách chuyển lần lượt SCL, SDA từ mức 0 lên mức 1.

Bên cạnh đó, giao tiếp I2C giúp chúng ta có thể trao đổi dữ liệu giữa các thiết bị Master và Slave một cách dễ dàng. Tuy nhiên, một số trường hợp có thể xảy ra xung đột hoặc lỗi dữ liệu. Một cách để giảm thiểu vấn đề này là mỗi thiết bị Master nên kiểm tra trạng thái của đường SDA trước khi bắt đầu truyền dữ liệu.

2.3 Kiến thức về các chuẩn giao tiếp vật lý

Trong hệ thống cũng như IoT, việc giao tiếp với các thiết bị ngoại vi, thu thập dữ liệu là một điều không thể tránh khỏi. Để giải quyết các thách thức liên quan đến IoT một cách hiệu quả, khái kiến thức về giao tiếp vật lý là một trong những điểm khá quan trọng. Kiến thức về giao tiếp vật lý đặt ra các yêu cầu quan trọng với các thành phần của hệ thống, bao gồm cả cảm biến, bộ điều khiển, và các ngoại vi. Trong kit thí nghiệm, nhóm hướng tới những giao tiếp phổ biến trên thị trường hiện nay như:

2.3.1 RS232 và RS485

RS232 [?] là một tiêu chuẩn giao tiếp định nghĩa giao diện giữa thiết bị truyền dữ liệu và thiết bị giao tiếp dữ liệu bằng cách trao đổi dữ liệu nhị phân theo chuỗi. Nó là một hình thức truyền thông chuỗi, cho phép truyền dữ liệu giữa các thiết bị. RS232 hỗ trợ cả chế độ truyền dữ liệu không đồng bộ và đồng bộ, nhưng thường được sử dụng không đồng bộ khi kết nối với máy tính cá nhân và các thiết bị khác. Mặc dù đã tồn tại lâu, RS232 vẫn được sử dụng trong nhiều ứng dụng công nghiệp và hệ thống, đặc biệt khi cần giao tiếp với các thiết bị chuyển động, bảng điều khiển và các thiết bị đo lường.

RS485 [?] là một tiêu chuẩn giao tiếp điện tử sử dụng tín hiệu cân bằng, hỗ trợ hệ thống điểm đa. Được xuất bản bởi TIA/EIA, RS-485 cho phép triển khai mạng truyền thông số ở khoảng cách xa và trong môi trường nhiễu điện. Với khả năng kết nối nhiều máy thu theo dạng bus multidrop, RS-485 thường được sử dụng trong kiểm soát công nghiệp. Hỗ trợ tốc độ truyền dữ liệu lên đến 10 Mbit/s và có thể hoạt động ổn định ở khoảng cách lên đến 1.200 m, RS-485 là lựa chọn phổ biến cho các ứng dụng đòi hỏi giao tiếp đa điểm và trong môi trường công nghiệp.

RS232 sử dụng chênh lệch điện áp với mức đất, trong khi RS485 chỉ sử dụng 2 dây với nguyên lý Logic 0 và 1. RS232 chỉ cho phép truyền điểm-điểm giữa 2 thiết bị, còn RS485 hỗ trợ kết nối đa điểm với nhiều thiết bị trên một đường truyền. RS232 có tốc độ nhanh nhưng khoảng cách truyền ngắn, trong khi RS485 có khoảng cách truyền xa nhưng tốc độ đáp ứng chậm hơn. Điều này làm cho RS232 phù hợp cho kết nối gần, còn RS485 thích hợp cho hệ thống đa điểm với khoảng cách truyền dài, thường ứng dụng trong lĩnh vực công nghiệp và kiểm soát.

2.3.2 SDcard

Thẻ nhớ Secure Digital card (SD card) [?] là một loại thẻ nhớ flash nhỏ được thiết kế cho dung lượng lớn, phổ biến trong nhiều thiết bị di động như điều hướng ô tô, điện thoại di động, máy ảnh, và máy tính cá nhân. Năm 1999, SanDisk, Panasonic và Toshiba hợp tác phát triển và tiếp thị tiêu chuẩn SD, cải tiến từ thẻ MultiMediaCard (MMC).

Tính năng tốc độ truyền dữ liệu cao của thẻ SD là một ưu điểm quan trọng, giúp chuyển đổi và truyền dữ liệu giữa thiết bị và thẻ nhanh chóng. Điều này rất quan trọng trong các tình huống đòi hỏi xử lý và ghi dữ liệu nhanh như quay video chất lượng cao hay chụp ảnh liên tục.

Ngoài ra, thẻ SD được xây dựng với tính năng tiêu thụ điện năng thấp, giúp tiết kiệm năng lượng và làm tăng tuổi thọ của pin trong các thiết bị di động. Điều này làm cho thẻ SD trở thành lựa chọn phổ biến trong các ứng dụng di động và đặc biệt hữu ích trong các thiết bị di động cần duy trì thời lượng pin lâu dài.

Cuối cùng, thẻ SD không chỉ là một phương tiện lưu trữ thông thường mà còn trở thành một tiêu chuẩn giao tiếp chung giữa các thiết bị. Sự tương thích cao này giúp người dùng dễ dàng chia sẻ dữ liệu giữa các thiết bị và sử dụng thẻ SD một cách linh hoạt trong nhiều tình huống khác nhau.

2.3.3 Ethernet

Ethernet là một phương thức truy cập mạng máy tính nội bộ (LAN), đặc trưng bởi việc sử dụng chuẩn đầu tiên và phổ biến nhất xuất phát từ chuẩn 802.3 của IEEE. Đây là một môi trường truyền thông được chia sẻ giữa các thiết bị trong mạng LAN, nơi mọi trạm đều sử dụng tổng băng thông của mạng, có thể là 10 Mbps, 100 Mbps và 1000 Mbps tùy thuộc vào phiên bản.

Ethernet ngày nay đã trở thành công nghệ mạng LAN phổ biến và được ứng dụng rộng rãi trên toàn thế giới, thường là lựa chọn hàng đầu khi nói đến kết nối mạng. Sự linh hoạt của Ethernet cho phép nó được tích hợp vào nhiều loại thiết bị và môi trường mạng khác nhau.

Sử dụng Ethernet trong môi trường công nghiệp mang lại nhiều lợi ích. Công

nghệ này cho phép áp dụng các kiến trúc điều khiển mới như điều khiển phân tán, điều khiển giám sát, và chẩn đoán lỗi từ xa. Cấu trúc liên kết giữa các thiết bị công nghiệp trở nên đơn giản hóa, tiết kiệm dây nối và công suất thiết kế, đồng thời nâng cao độ tin cậy và độ chính xác của thông tin. Nó cũng giúp tăng cường tính linh hoạt và tính mở của hệ thống.

Một điểm đặc biệt là với Ethernet công nghiệp, việc tham số hóa, chẩn đoán, và vận hành có thể thực hiện từ xa thông qua một trạm kỹ thuật trung tâm. Điều này cung cấp khả năng quản lý và điều khiển hệ thống từ xa một cách hiệu quả, giảm độ phức tạp và chi phí trong quá trình bảo trì và quản lý.

2.3.4 CAN bus

Bus CAN (Controller Area Network) [?] là một hệ thống giao tiếp chuẩn quốc tế, được phát triển ban đầu để đáp ứng nhu cầu trong ngành công nghiệp ô tô. Mục tiêu chính của Bus CAN là thay thế việc sử dụng dây cáp phức tạp bằng một bus hai dây đơn giản và hiệu quả. Chuẩn này không chỉ giúp giảm độ phức tạp của hệ thống dây cáp mà còn mang lại khả năng chống nhiễu điện từ cao, khả năng tự chẩn đoán và sửa lỗi dữ liệu.

Bus CAN đã nhanh chóng trở thành chuẩn giao tiếp phổ biến không chỉ trong ngành ô tô mà còn trong nhiều lĩnh vực khác như tự động hóa xây dựng, y tế và sản xuất. Khả năng chịu nhiễu và độ tin cậy cao của Bus CAN đã làm cho nó trở thành lựa chọn ưu việt trong các môi trường công nghiệp khắc nghiệt.

Giao thức truyền thông của Bus CAN mô tả cách thông tin được truyền giữa các thiết bị trên một mạng và tuân theo mô hình Open Systems Interconnection (OSI) được đặc tả trong các lớp. Việc giao tiếp giữa các thiết bị qua phương tiện vật lý được xác định bởi lớp vật lý của mô hình OSI.

Bus CAN được chia thành hai phiên bản chính dựa trên đặc điểm thông số kỹ thuật của Bosch. Phiên bản 2.0 của CAN được chia thành hai phần: phiên bản tiêu chuẩn sử dụng ID (Identifier) 11 bit và phiên bản mở rộng sử dụng ID 29 bit. Điều này mang lại tính linh hoạt cao, cho phép sự mở rộng và tích hợp dễ dàng vào các hệ thống phức tạp và đa dạng.

2.4 Kiến thức về xây dựng tài liệu

Trong phần xây dựng bộ bài thí nghiệm, chúng tôi mong muốn người dùng có thể học tập vi điều khiển từ con số 0. Để đạt được mục tiêu này, nhóm sẽ biên soạn một tài liệu hướng dẫn nhanh (quick start guide) nhằm giúp người dùng nhanh chóng cấu hình và sử dụng kit thí nghiệm. Tài liệu này được thiết kế để phù hợp với cả những người chưa có kỹ năng về đọc datasheet, giúp họ dễ dàng bắt đầu học tập và thực hành mà không gặp quá nhiều khó khăn ban đầu.

Bộ bài thí nghiệm sẽ bắt đầu với những hướng dẫn cơ bản nhất. Người dùng sẽ được học cách sử dụng GPIO để điều khiển đèn LED, lập trình quét LED theo các mẫu khác nhau, và sử dụng ngắn (sâu hơn là timer interrupt) để tạo ra các tác vụ đáp ứng tính real time. Những bài học này sẽ giúp người dùng làm quen với việc lập trình và điều khiển các thiết bị ngoại vi cơ bản.

Sau khi nắm vững các khái niệm cơ bản, người dùng sẽ tiến tới các bài học phức tạp hơn như cách sử dụng bộ chuyển đổi tương tự sang số (ADC) để đọc giá trị từ các cảm biến analog, và điều chế độ rộng xung (PWM) để điều khiển độ sáng đèn LED hoặc tốc độ động cơ, trên kit thì nghiệm sẽ dùng PWM để điều khiển buzzer. Những kỹ năng này sẽ rất hữu ích cho việc xây dựng các ứng dụng điều khiển trong thực tế.

Tiếp theo, người dùng sẽ được hướng dẫn về các chuẩn giao tiếp như UART, SPI, và I2C. Các hướng dẫn về phần này sẽ giúp họ hiểu cách kết nối và trao đổi dữ liệu giữa vi điều khiển và các thiết bị khác, chẳng hạn như cảm biến, màn hình, và bộ nhớ ngoài. Đây là các chuẩn giao tiếp hết sức thông dụng và phổ biến trên thị trường nhúng hiện nay.

Đặc biệt, nhóm sẽ cung cấp các bài hướng dẫn chi tiết về việc tích hợp module ESP để phát triển các dự án IoT, bao gồm việc thu thập dữ liệu từ cảm biến và truyền thông qua mạng Internet. Đây là một bước quan trọng để người dùng có thể xây dựng các ứng dụng IoT thông minh và kết nối. Bên cạnh đó, chúng tôi sẽ hướng dẫn người dùng cách làm việc với hệ điều hành thời gian thực (RTOS). Các bài hướng dẫn cơ bản về RTOS trên STM32 Cube IDE, đây là một điều còn khá mới, nhưng nó cũng giúp người học có thể thấy được sức mạnh của vi điều khiển hiện nay.

Bộ bài thí nghiệm của chúng tôi không chỉ dừng lại ở lý thuyết mà còn bao gồm nhiều dự án thực hành cụ thể. Người dùng sẽ được tham gia vào các dự án

từ đơn giản đến phức tạp, từ việc lập trình cơ bản cho đến xây dựng các hệ thống IoT hoàn chỉnh. Những dự án này sẽ giúp họ áp dụng những kiến thức đã học vào thực tế, từ đó nắm vững kỹ năng lập trình và phát triển ứng dụng trên vi điều khiển. Bộ bài thí nghiệm được thiết kế để cung cấp một lộ trình học tập rõ ràng và toàn diện, giúp người dùng phát triển kỹ năng và tự tin trong việc lập trình và sử dụng vi điều khiển. Chúng tôi hy vọng rằng, qua bộ bài thí nghiệm này, người dùng sẽ tìm thấy niềm vui và sự hứng thú trong việc học tập và khám phá thế giới vi điều khiển và hệ thống nhúng.

2.5 Kiến thức về Database và Backend

Trong phần này, nhóm sẽ trình bày các công nghệ nhóm sử dụng để xây dựng database và server backend. Nhóm đã chọn sử dụng MongoDB và Node.js để xây dựng cơ sở dữ liệu và server backend cho dự án. MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) phổ biến, được ưa chuộng với tính linh hoạt và khả năng mở rộng. Trong khi đó, Node.js là một môi trường thực thi mã nguồn mở, được sử dụng chủ yếu cho việc xây dựng ứng dụng web có hiệu suất cao, có khả năng xử lý bất đồng bộ. Sự kết hợp giữa MongoDB và Node.js hứa hẹn tạo ra một hệ thống mạnh mẽ và linh hoạt, phục vụ tốt cho mục tiêu và yêu cầu của dự án.

2.5.1 RESTful API



RESTful API là một kiến trúc thiết kế API dựa trên nguyên tắc của REST (Representational State Transfer), được sử dụng để xây dựng các dịch vụ web linh hoạt và mạnh mẽ. Đây là một phương pháp tiêu biểu trong việc tạo ra các ứng dụng web hoặc di động, cho phép tương tác với cơ sở dữ liệu một cách hiệu quả. Các đặc điểm và tính năng chính của RESTful API bao gồm:

- HTTP Methods: Sử dụng các phương thức HTTP như GET, POST, PUT, PATCH, và DELETE để thực hiện các thao tác trên tài nguyên. Mỗi phương thức có ý nghĩa và mục đích riêng.
- Stateless Communication: Giao tiếp trong RESTful API là stateless, điều này có nghĩa là mỗi yêu cầu từ client chứa đủ thông tin cần thiết cho server xử lý mà không phụ thuộc vào trạng thái trước đó.
- Resource-Based: Mọi thứ trong RESTful API được xem như một tài nguyên và được định danh bằng các URI (Uniform Resource Identifier). Điều này giúp dễ dàng quản lý và tìm kiếm các tài nguyên.
- Representation: Dữ liệu của tài nguyên được truyền dưới dạng các biểu diễn như JSON hoặc XML, giúp tương thích và linh hoạt.
- HATEOAS (Hypermedia As The Engine Of Application State): RESTful API có thể cung cấp các liên kết hypermedia trong các phản hồi, giúp client hiểu được các hành động mà nó có thể thực hiện tiếp theo.
- Versioning: RESTful API thường hỗ trợ quản lý phiên bản để đảm bảo tính tương thích giữa các phiên bản khác nhau của API.
- Authentication và Authorization: Hỗ trợ các phương thức xác thực và phân quyền để đảm bảo an toàn và bảo mật thông tin.
- Caching: Sử dụng các cơ chế caching để tối ưu hóa hiệu suất và giảm thời gian đáp ứng của hệ thống.

Nhờ vào tính linh hoạt và khả năng mở rộng, RESTful API đã trở thành một tiêu chuẩn phổ biến trong việc xây dựng các dịch vụ web và ứng dụng di động.

2.5.2 JSON



JSON (JavaScript Object Notation) là một định dạng dữ liệu phổ biến được sử dụng để truyền tải và lưu trữ dữ liệu dưới dạng văn bản. Đây là một định dạng dữ liệu nhẹ, dễ đọc và dễ hiểu cho cả con người và máy tính. JSON thường được

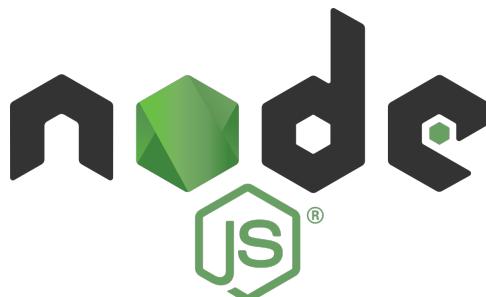
sử dụng trong các ứng dụng web, API và các hệ thống phân tán.

Dặc điểm của JSON bao gồm:

- Dữ liệu Dạng Key-Value Pairs: JSON sử dụng cặp key-value để biểu diễn dữ liệu. Mỗi cặp key-value được phân tách bằng dấu hai chấm (:), và các cặp này được phân tách bằng dấu phẩy (,).
- Dạng Dữ Liệu Dựa Trên Text: JSON sử dụng các ký tự văn bản để biểu diễn dữ liệu, bao gồm các chuỗi ký tự, số, boolean, null, mảng và đối tượng.
- Định Dạng Đồng Nhất: JSON có một cấu trúc dữ liệu đồng nhất và dễ hiểu, giúp cho việc truyền tải và xử lý dữ liệu trở nên dễ dàng.
- Khả Năng Đa Dạng Hóa Dữ Liệu: JSON hỗ trợ các kiểu dữ liệu đa dạng như số nguyên, số thực, chuỗi ký tự, boolean, mảng và đối tượng, giúp phù hợp với nhiều loại dữ liệu khác nhau.
- Tương Thích Với Nhiều Ngôn Ngữ và Nền Tảng: JSON không phụ thuộc vào bất kỳ ngôn ngữ lập trình cụ thể nào và có thể được sử dụng trong hầu hết các ngôn ngữ lập trình và nền tảng.

JSON thường được sử dụng trong các ứng dụng web và API để truyền tải dữ liệu giữa client và server, và đã trở thành một phần không thể thiếu của việc phát triển các ứng dụng phân tán và dịch vụ web hiện đại.

2.5.3 NODEJS



Node.js là một nền tảng phát triển dựa trên JavaScript, được xây dựng trên nền tảng V8 JavaScript Engine của Google Chrome. Nó cho phép viết mã JavaScript chạy ở phía máy chủ, thường được sử dụng để xây dựng các ứng dụng web và dịch vụ mạng.

- JavaScript Everywhere: Node.js cho phép viết mã JavaScript không chỉ ở phía máy khách (trình duyệt), mà còn ở phía máy chủ. Điều này giúp các nhà phát triển web sử dụng cùng một ngôn ngữ lập trình trên cả hai phía, giảm thiểu sự phân tán và tăng tính nhất quán của mã nguồn.
- Khả năng Xử lý Đa Luồng: Node.js sử dụng mô hình không đồng bộ và sự kiện (event-driven) trên một luồng duy nhất, giúp tối ưu hóa hiệu suất khi xử lý các yêu cầu I/O như đọc/ghi tệp, truy vấn cơ sở dữ liệu, và giao tiếp mạng.
- Hệ sinh thái Mô-đun Nguồn Mở: Node.js có một cộng đồng lớn và mạnh mẽ, cung cấp hàng ngàn các mô-đun mã nguồn mở trên npm (Node Package Manager), từ các thư viện cơ bản đến các framework và công cụ phát triển.
- Công Cụ Phát Triển Mạnh Mẽ: Node.js đi kèm với một số công cụ phát triển mạnh mẽ như npm, nodemon, eslint, và debug, giúp tối ưu hóa quy trình phát triển, kiểm tra, và triển khai ứng dụng.
- Phát triển Ứng Dụng Thời Gian Thực: Nhờ mô hình không đồng bộ và sự kiện, Node.js thích hợp cho việc xây dựng các ứng dụng thời gian thực, như ứng dụng chat, trò chơi trực tuyến, và các ứng dụng streaming.
- Khả năng Mở Rộng và Mô-đun: Node.js hỗ trợ cơ chế mô-đun để phân tách mã nguồn thành các phần nhỏ, dễ quản lý và tái sử dụng. Nó cũng có thể được mở rộng một cách linh hoạt, từ các ứng dụng nhỏ đến các hệ thống phân tán lớn.

Với những đặc điểm và lợi ích trên, Node.js đã trở thành một trong những nền tảng phát triển phổ biến nhất cho việc xây dựng các ứng dụng web hiện đại và dịch vụ mạng.

2.5.4 MONGODB



MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) mã nguồn mở, phát triển bởi MongoDB Inc. Nó được thiết kế để lưu trữ dữ liệu dưới dạng tài liệu linh hoạt, mà không cần tuân theo cấu trúc cố định như trong các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Dưới đây là một số điểm chi tiết về MongoDB:

- Hướng Tài Liệu: MongoDB lưu trữ dữ liệu dưới dạng tài liệu JSON-like gọi là BSON (Binary JSON), cho phép lưu trữ dữ liệu linh hoạt và phong phú với các trường động và tập hợp lồng nhau. Điều này phù hợp với các ứng dụng có yêu cầu dữ liệu động và thay đổi thường xuyên.
- Khả Năng Mở Rộng và Phân Tán: MongoDB hỗ trợ khả năng mở rộng ngang (horizontal scaling) thông qua việc phân tán dữ liệu trên nhiều máy chủ (nodes). Điều này cho phép mở rộng hệ thống để xử lý lượng dữ liệu lớn và tải cao mà không giảm đi hiệu suất.
- Query Mạnh Mẽ: MongoDB cung cấp một ngôn ngữ truy vấn linh hoạt và mạnh mẽ, cho phép thực hiện các truy vấn phức tạp và hiệu quả trên dữ liệu. Nó hỗ trợ các phép toán truy vấn như so sánh, phủ định, lựa chọn, sắp xếp, và tổng hợp.
- Khả Năng Tích Hợp: MongoDB tích hợp tốt với nhiều ngôn ngữ lập trình và framework phổ biến như JavaScript, Python, Node.js, và Django. Nó cũng hỗ trợ các công cụ như Mongoose để tạo ra các mô hình dữ liệu và thực hiện truy vấn trong ứng dụng Node.js.
- Tính Năng Tương Thích ACID: Mặc dù MongoDB là một cơ sở dữ liệu NoSQL, nhưng nó vẫn hỗ trợ các tính năng như Atomicity, Consistency, Isolation, và Durability (ACID) thông qua việc sử dụng các giao thức Write-Ahead Logging (WAL) và cơ chế phục hồi dữ liệu.
- Dễ Dàng Quản Lý: MongoDB cung cấp các công cụ quản lý dễ sử dụng như MongoDB Compass và MongoDB Atlas để giúp quản lý cơ sở dữ liệu, theo dõi hiệu suất, và sao lưu dữ liệu một cách dễ dàng.

MongoDB đã trở thành một lựa chọn phổ biến cho các ứng dụng web, di động, IoT và phân tích dữ liệu, nhờ vào tính linh hoạt, mở rộng và hiệu suất của nó.

CHƯƠNG 3

MỤC TIÊU - PHƯƠNG PHÁP THIẾT KẾ HỆ THỐNG

3.1 MỤC TIÊU THIẾT KẾ PHẦN CỨNG

Trong phần này sẽ trình bày thiết kế về các khối chức năng của Kit thí nghiệm. Vì đây là sản phẩm mang tính giáo dục nên cần cân nhắc lựa chọn các khối chức năng sao cho học viên vừa có thể tiếp xúc với đầy đủ các tính năng cơ bản, vừa có khả năng học hỏi các công nghệ mới, áp dụng trong công nghiệp mà sản phẩm vẫn có các tính năng bắt mắt, hấp dẫn đối với người học. Sau đây là một số khái niệm nổi bật.

Đối với những học viên bắt đầu học về lập trình vi điều khiển, việc làm quen với GPIO là điều bắt buộc. GPIO là ngoại vi cơ bản, và quan trọng nhất để học viên có nền tảng để tiếp xúc với các ngoại vi nâng cao hơn. Để việc học tập các chân GPIO được trực quan nhất, Kit thí nghiệm sẽ được thiết kế với các LED đơn và nút nhấn. Đối với LED, ngoài việc sử dụng các LED đơn, nhóm còn tích hợp thêm LED đồng hồ. Vì đây là linh kiện có thể giúp học viên tiếp xúc với kỹ thuật quét LED, cũng như sử dụng linh kiện này để xây dựng một giao diện tương tác với người dùng qua các con số, điều được sử dụng phổ biến trong các hệ thống ngày nay.

Đối với các nút nhấn, nhóm sẽ thiết kế một ma trận phím gồm các nút nhấn được ký hiệu số và các ký hiệu điều khiển cơ bản. Ma trận phím cũng là thiết kế đơn giản được thường xuyên sử dụng trong các thiết bị, phục vụ nhu cầu nhập thông tin, điều khiển thiết bị.



Hình 3.1: Ứng dụng thực tế của LED 7 đoạn và ma trận phím

Tiếp theo đó, để tạo sự thu hút cũng như tính hấp dẫn của Kit thí nghiệm, nhóm sẽ tích hợp một màn hình LCD Graphic có hỗ trợ khả năng cảm ứng chạm. Màn hình này sẽ đóng vai trò giúp trực quan hóa hệ thống, khiến cho việc học lập trình hệ thống nhúng trở nên sinh động hơn.

Trong thực tế, việc đọc thông tin từ các cảm biến là yêu cầu bắt buộc đối với hệ thống nhúng. Do đó, khối cảm biến sẽ được thiết kế sử dụng các cảm biến xuất tín hiệu dạng tương tự giúp học viên có thể khảo sát tính năng chuyển đổi tín hiệu tương tự sang tín hiệu số trên vi điều khiển. Ngoài ra các ứng dụng thực tế đều sử dụng IC thời gian thực (RTC) với mục đích quản lý thời gian hệ thống. Điều này sẽ giúp giảm tải cho Vi điều khiển so với việc sử dụng RTC nội bộ tích hợp sẵn. Nhóm sẽ học hỏi và thiết kế khôi RTC ngoại kèm với nguồn cấp riêng để thời gian vẫn có thể được cập nhật kể cả khi Kit thí nghiệm không được cấp nguồn. Ngoài ra việc sử dụng IC RTC sẽ giúp học viên học cách giao tiếp với nó qua các chuẩn giao tiếp cơ bản.

Trong hệ thống nhúng, việc giao tiếp với các bo mạch, các thiết bị khác trong hệ thống là cực kì quan trọng. Vì vậy, trong quá trình thiết kế, nhóm đã thêm các khối chức năng với các chuẩn giao tiếp công nghiệp thông dụng như RS232, RS485, CAN và các ngõ vào, ngõ ra cho thiết bị công suất.

Để đạt được mục đích học tập về IoT, việc kết nối Internet là thực sự cần thiết. Nhóm sẽ thiết kế thêm khôi gồm esp8266 với khả năng kết nối Internet thông qua Wifi và Ethernet. Ngoài ra, Kit thí nghiệm sẽ bao gồm khôi Sim hỗ trợ việc kết

nối mạng thông qua 4G/LTE.

Thêm vào đó, trong ứng dụng thực tế, khả năng lưu trữ dữ liệu là vấn đề cần được quan tâm. Vì vậy các bộ nhớ sẽ được tích hợp cho Kit thí nghiệm bao gồm RAM, EEPROM, FLASH và thẻ nhớ SD.

3.1.1 Thiết kế khối vi điều khiển trung tâm

3.1.1.1 Vi điều khiển STM32F407ZGT6

Trên kit thí nghiệm, nhóm sử dụng STM32F407ZGT6 làm vi điều khiển trung tâm.



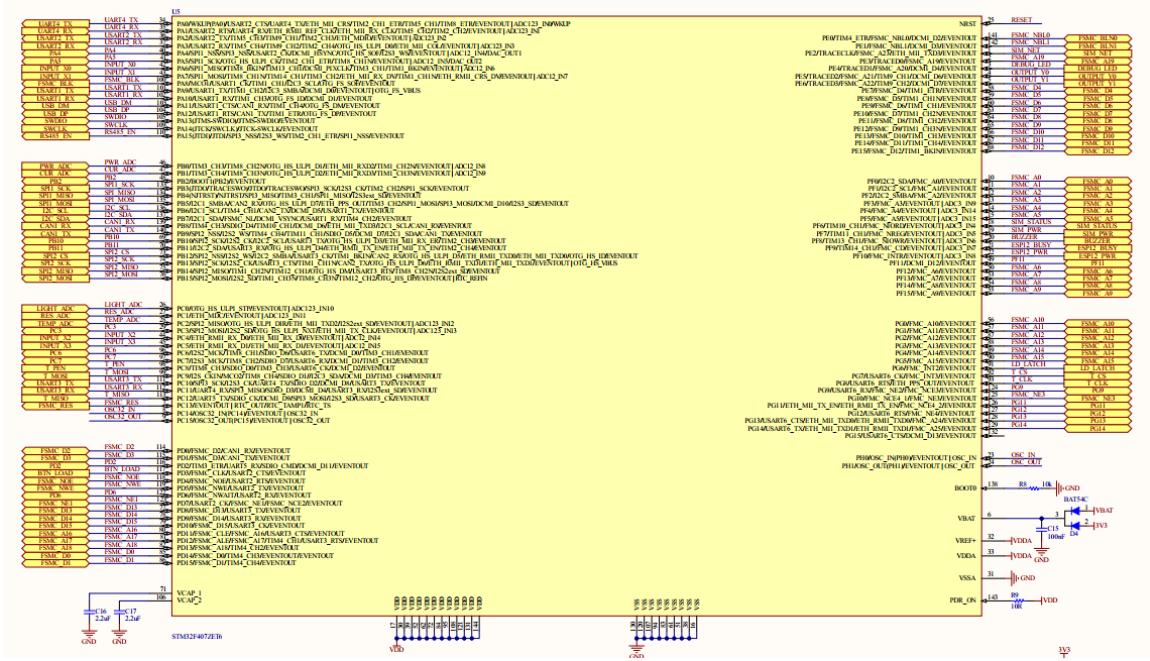
Hình 3.2: STM32F407ZGT6

Lõi	Arm Cortex - M4
Điện áp cấp khuyên dùng	3.3V
Tần số tối đa	168MHz
Đóng gói	LQFP144
Bộ nhớ flash	1024 KB
RAM	192 KB
Dòng ra tối đa mỗi chân I/O	25 mA
Tổng dòng ra	240 mA

Bảng 3.1: Thông số STM32F407ZGT6

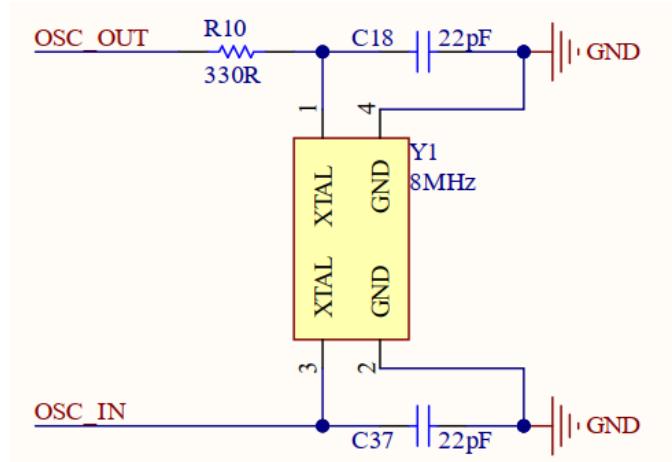
Sơ đồ nguyên lý của STM32F407ZGT6:

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



Hình 3.3: Sơ đồ nguyên lý của STM32F407ZGT6

Theo như sơ đồ, ta có thể thấy gần như toàn bộ chân Vi điều khiển đều được nhóm sử dụng. Với mong muốn tận dụng triệt để các chân của vi điều khiển để sử dụng cho việc học tập cũng như làm việc, nhóm đã sử dụng các chân vi điều khiển để chia vào các khối chức năng như là: tương tác với người dùng thông qua nút nhấn, led 7 đoạn, ADC, RTC, giao tiếp RS232, RS485,... Ngoài ra để đảm bảo hoạt động của Vi điều khiển được ổn định với một tốc độ cố định, ta cần cấu hình sử dụng thạch anh ngoài để tạo xung cho Vi điều khiển, có thể đẩy tần số của vi điều khiển lên tối đa (168MHz). Kết nối thạch anh với Vi điều khiển nhóm thiết kế như sau:

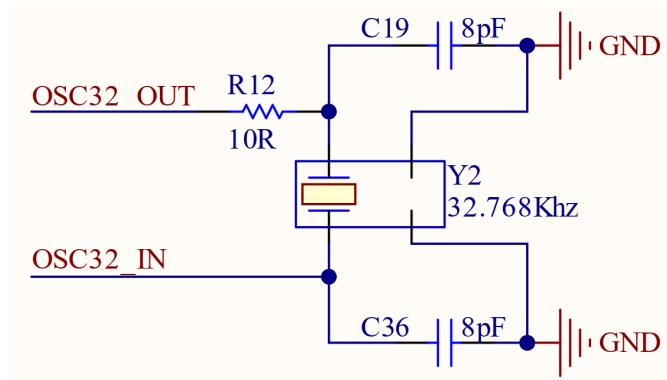


Hình 3.4: Thạch anh ngoài tạo xung cho Vi điều khiển

Để hỗ trợ cho đồng hồ thời gian thực nội, nhóm sẽ sử dụng thêm một thạch

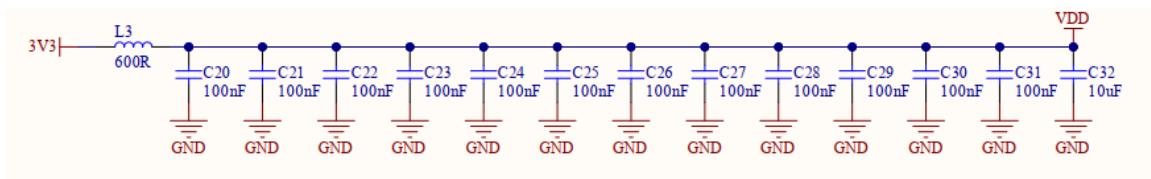
3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

anh ngoại 32.768 kHz low-speed (LSE crystal). Thạch anh này giúp tăng độ chính xác và có tính ổn định cao, làm cho nó phù hợp cho các ứng dụng yêu cầu đồng hồ chính xác trong thời gian dài. Thạch anh này cũng giúp tiết kiệm năng lượng vì tần số của nó là tốc độ thấp, giúp tối ưu hóa tiêu thụ năng lượng trong các ứng dụng di động hoặc pin dự trữ.



Hình 3.5: Sơ đồ nguyên lý của LSE crystal

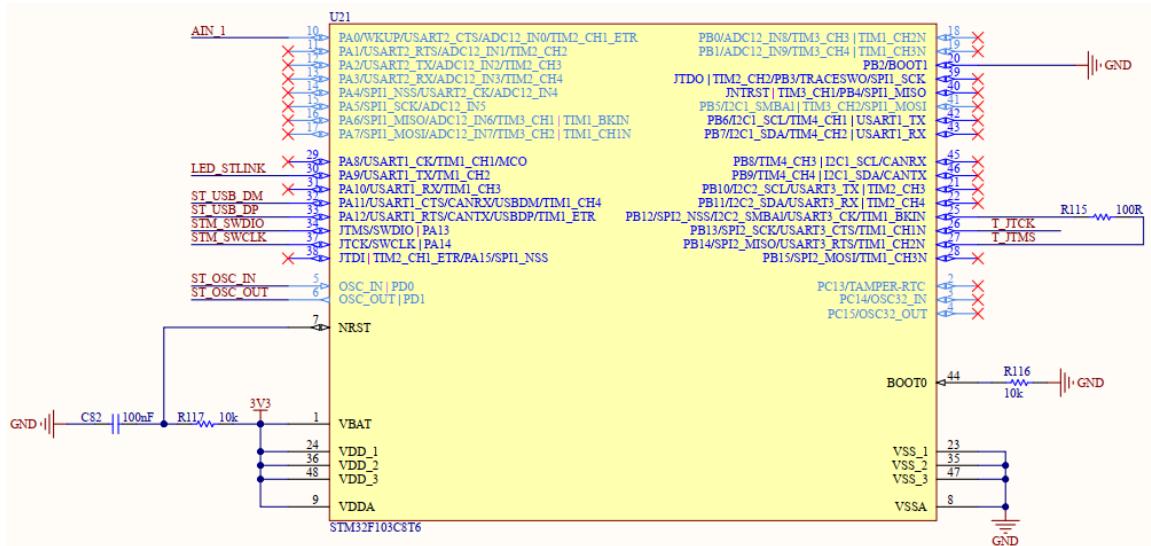
Và một số bộ phận đảm nhận chức năng lọc nguồn để đảm bảo cho nguồn 3.3V cấp vào Vi điều khiển ổn định, tránh nhiễu điện làm giảm tuổi thọ của vi điều khiển. Trong phần này nhóm đã thiết kế các tụ song song với mục đích tăng hiệu suất lọc, cải thiện điện áp đầu ra, tăng tổng dung lượng tụ của bộ lọc có thể giúp đảm bảo rằng mạch nguồn có đủ năng lượng để đáp ứng đòi hỏi của tải.



Hình 3.6: Bộ lọc nguồn cho Vi điều khiển

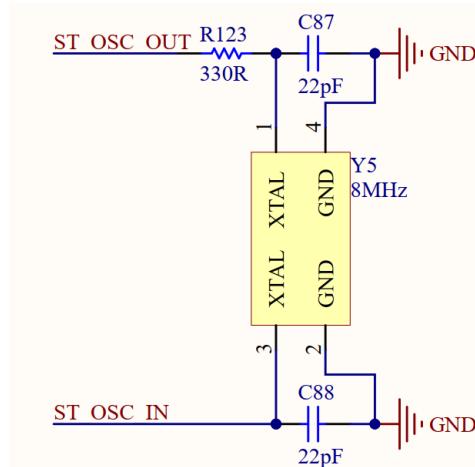
3.1.1.2 Khối nạp code - STlink

Để nạp code cho vi điều khiển chính, ta cần thiết kế một khối nạp. Khối nạp này sẽ được lấy ý tưởng từ mạch ST-LINK V2. Chương trình nạp sẽ được chạy bởi vi điều khiển STM32F103C8T6 và nạp code cho vi điều khiển chính thông qua chuẩn SWD. Để thuận tiện cho người sử dụng, khối nạp code được thiết kế với cổng Type-C.



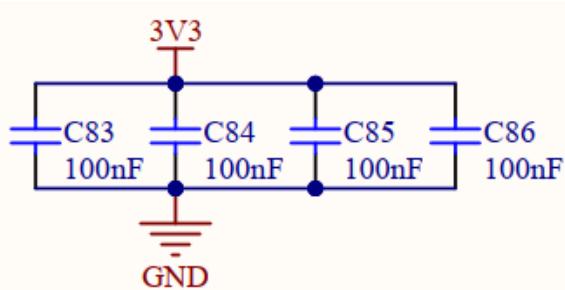
Hình 3.7: Vi điều khiển sử dụng làm chip nạp

Tương tự như thạch anh 8MHz chúng ta đã sử dụng cho chip chính, ở chip nạp chúng ta cũng sử dụng một thạch anh tương tự với cùng mục đích như trên



Hình 3.8: Thạch anh ngoài tạo xung cho Vi điều khiển

Để đảm bảo ổn định cho chip nạp theo quy định của nhà sản xuất, nhóm thiết kế thêm bộ lọc nguồn cho chip nạp như sau:



Hình 3.9: Bộ lọc nguồn cho chip nạp

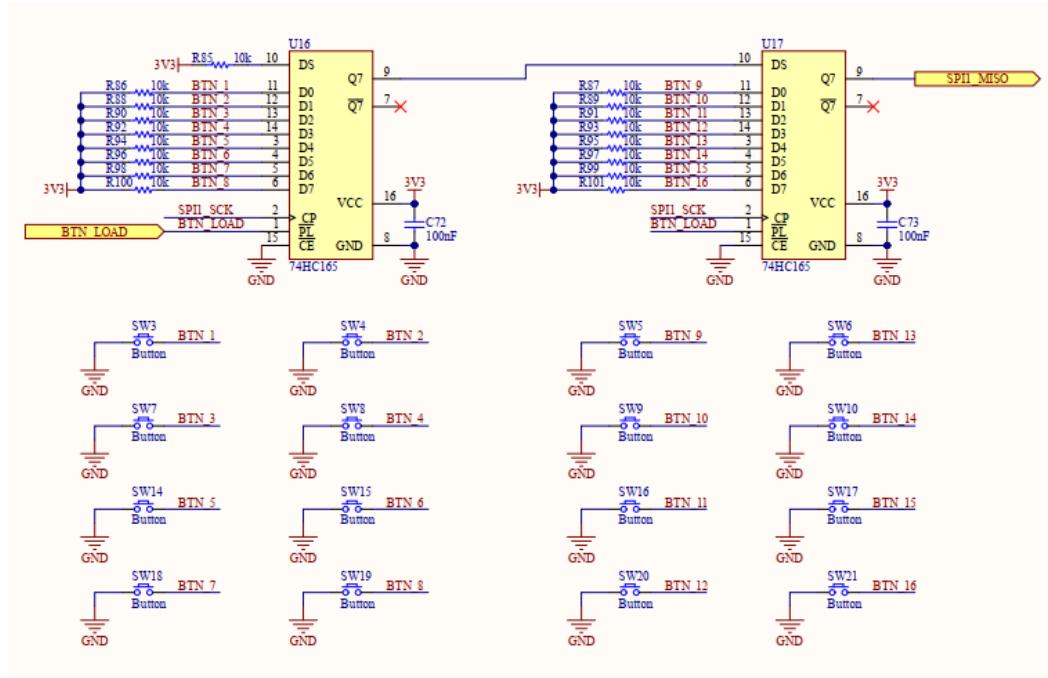
3.1.2 Khối tương tác với người dùng

Để người dùng có thể tiếp cận với các chức năng inout cơ bản, dễ dàng tương tác với mạch cũng như các ứng dụng IoT bên ngoài, nhóm sử dụng ma trận phím, led 7 đoạn, các led đơn và màn hình LCD touch với mong muốn có thể đáp ứng đầy đủ nhu cầu của người dùng và giúp người có thể học tập, nghiên cứu một cách trực quan hơn so với các cách truyền thống trước đây.

3.1.2.1 Ma trận phím

Nút nhấn là một thành phần thường xuyên xuất hiện trên hầu hết các hệ thống, khái niệm nút nhấn có thể hiểu theo nhiều dạng khác nhau nhưng chung quy lại nó sẽ là Input được người dùng nhập vào để có thể tương tác với hệ thống. Ma trận phím của chúng tôi gồm 16 nút nhấn với các chức năng cơ bản và quen thuộc của hệ thống. Các nút nhấn được chúng tôi mô tả như sau: 1 nút Enter, 1 nút Back, 2 nút điều chỉnh lên xuống (Up/Down), 2 nút điều chỉnh trái phải (Left/Right) và 10 nút chữ/số có thể đáp ứng được tối đa nhu cầu sử dụng của người dùng. Sơ đồ nguyên lý nhóm thiết kế như sau:

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



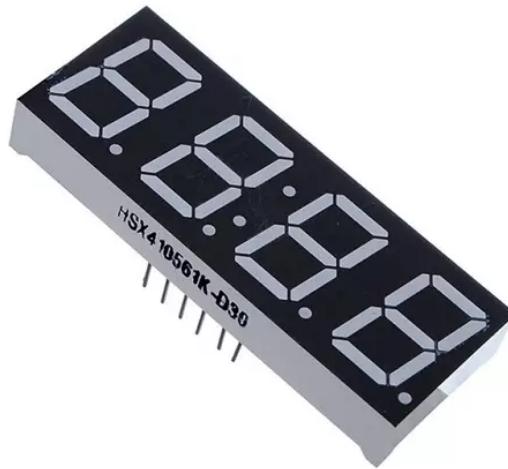
Hình 3.10: Sơ đồ nguyên lý ma trận phím

Nếu chỉ kết nối trực tiếp nút nhấn với vi điều khiển thì ta cần tới 16 chân GPIO của Vi điều khiển để đọc các Input này, điều đó thật sự lãng phí tài nguyên của Vi điều khiển, do đó, ta cần dùng thêm các IC thanh ghi dịch để lưu trạng thái của các nút nhấn vào một thanh ghi sau đó trả dữ liệu về Vi điều khiển. Với nguyên lý hoạt động của 74HC165, giao tiếp giữa Nút nhấn và Vi điều khiển có thể được hiện thực thông qua giao tiếp SPI. Thông qua ma trận phím này người học có thể tìm hiểu các kiến thức về xử lý nút nhấn cũng như giao tiếp SPI, đây là kiến thức hết sức cơ bản trong hệ thống nhúng.

3.1.2.2 Led 7 đoạn

Led 7 đoạn là thành phần điện tử khá phổ biến hiện nay, để dễ dàng hiển thị hoặc cho người dùng thấy được trạng thái hiện tại của hệ thống, một trong những ứng dụng nổi bật nhất của module led 7 đoạn là hiển thị thời gian thực. Bài toán RTC là một bài toán cơ bản trong hệ thống nhúng mà người học thường sẽ được tiếp cận qua. Trong bài này chúng ta sẽ sử dụng module Led 7 đoạn Anode chung.

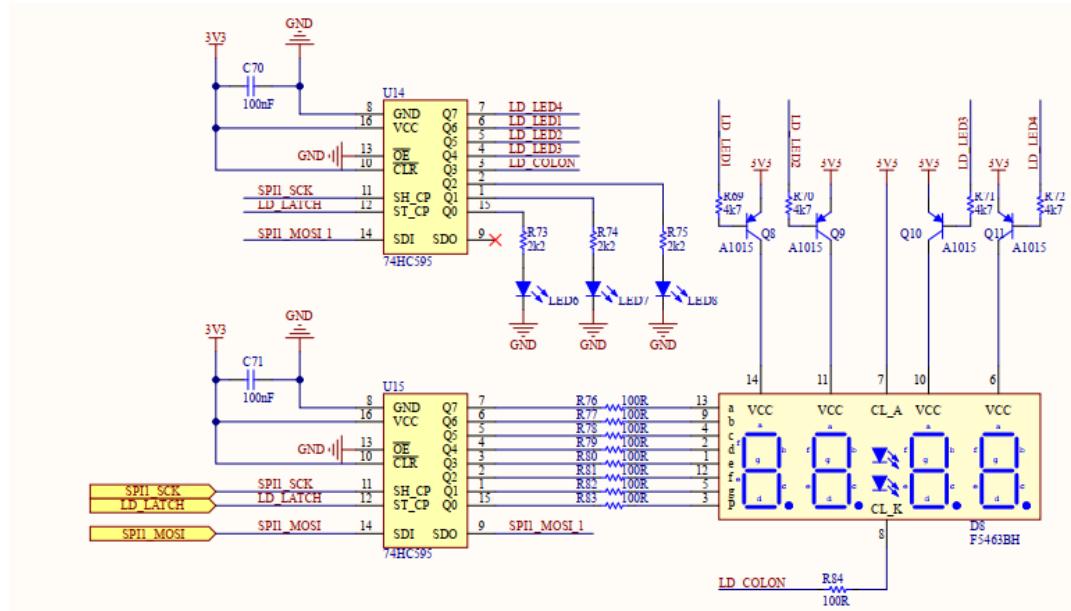
3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



Hình 3.11: Module 4 led 7 đoạn

Để tiết kiệm chân cho Vi điều khiển, khối Led 7 đoạn sẽ được điều khiển thông qua IC dịch 74HC595 với SPI. Với 2 IC 74HC595, ta có thể điều khiển tới 16 Output, trong khi để điều khiển khối Led 7 đoạn chỉ cần 13 Output. Với 3 tín hiệu Output còn lại nhóm sẽ thiết kế để điều khiển 3 Led đơn.

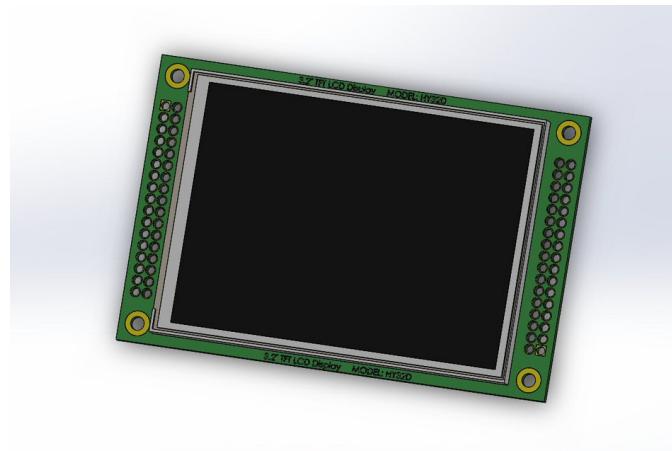
Dưới đây là sơ đồ nguyên lý của led 7 đoạn trên mạch:



Hình 3.12: Sơ đồ nguyên lý led 7 đoạn

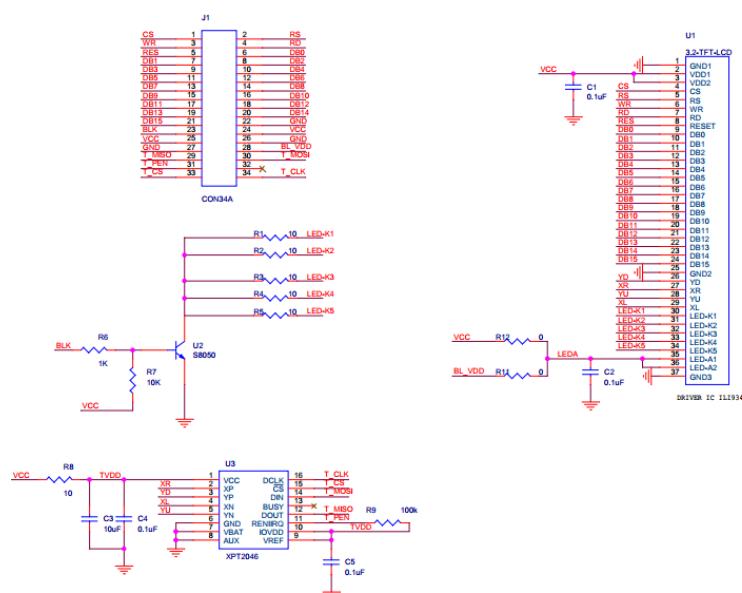
3.1.2.3 Màn hình LCD touch

Để người dùng có thể nâng cao tương tác với các thiết bị trong quá trình học lập trình nhúng, màn hình LCD touch có một số lợi ích khá quan trọng. Màn hình LCD touch cho phép người dùng mở rộng khả năng tương tác, giúp người dùng có thể phát triển các ứng dụng với giao diện thân thiện hơn, trực quan hơn. Trên Kit thí nghiệm, để tối ưu về cả chi phí lẫn diện tích, nhóm quyết định lựa chọn màn hình LCD TFT 3.2in với độ phân giải 320x240.



Hình 3.13: LCD touch 3.2in 320x240

Theo các thông tin từ nhà sản xuất, màn hình LCD TFT 3.2in hoạt động ở điện áp 3.3V với chế độ màu RGB 16bit, được điều khiển bởi IC ILI9341 và cảm ứng chạm thông qua IC XPT2046.



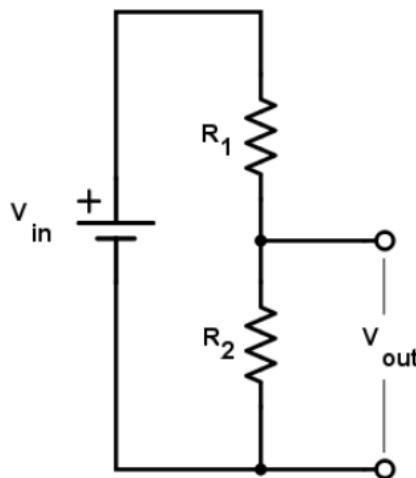
Hình 3.14: Sơ đồ nguyên lý màn hình LCD touch

3.1.3 Thiết kế khối đọc ADC

Trong hệ thống nhúng và phát triển ứng dụng IoT, ADC (Analog-to-Digital Converter) đóng một vai trò khá quan trọng trong việc chuyển đổi các tín hiệu tương tự từ thế giới ngoại vi, như cảm biến, thành dạng số để xử lý bởi vi xử lý số hoặc vi điều khiển.

Bộ ADC bên trong STM32 sẽ đọc tín hiệu điện áp từ chân ADC và chuyển sang tín hiệu số. Vậy nên việc học tập, khảo sát ADC sẽ dẫn đến nhu cầu người dùng thay đổi điện áp vào chân Vi điều khiển. Điều trên có thể được hiện thực đơn giản bằng cách sử dụng một biến trở có thể điều chỉnh dễ dàng bằng tay.

Bộ chia điện áp hai điện trở là một trong những mạch phổ biến và hữu ích được các kỹ sư thường xuyên sử dụng. Mục đích chính của mạch này là giảm điện áp đầu vào xuống giá trị thấp hơn dựa trên tỷ số của hai điện trở. Việc này giúp xác định điện áp đầu ra của mạch chia dựa trên điện áp đầu vào (hoặc nguồn) và các giá trị điện trở. Một điểm lưu ý là điện áp đầu ra trong các mạch thực tế có thể khác nhau, vì dung sai điện trở và điện trở tải (nơi kết nối điện áp đầu ra) trở thành các hệ số.

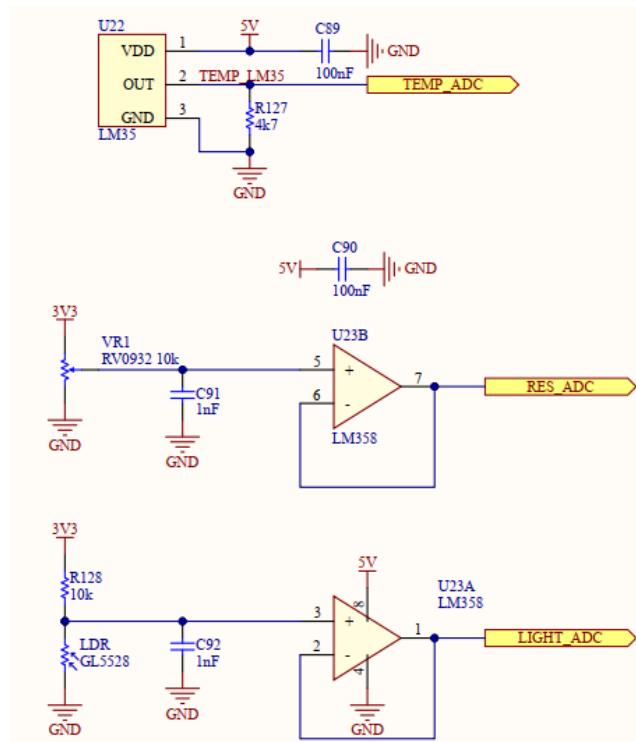


Hình 3.15: Mạch chia áp (Nguồn: All About Circuits)

$$V_{Out} = V_{In} \times \frac{R_2}{R_1 + R_2}$$

Để người học có thể dễ dàng nghiên cứu về ADC trong môi trường học tập, làm việc ở bất kỳ nơi nào, nhóm sử dụng 3 thiết bị là biến trở, cảm biến ánh sáng và cảm biến nhiệt độ.

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



Hình 3.16: Sơ đồ nguyên lý của khối cảm biến

- Như trên schematic, nhóm sử dụng cảm biến nhiệt độ LM35, cảm biến nhiệt độ LM35 có điện áp Analog đầu ra tuyến tính theo nhiệt độ thường được sử dụng để đo nhiệt độ của môi trường hoặc theo dõi nhiệt độ của thiết bị,...
- Biến trở trên Kit thí nghiệm sẽ được cấu hình thành mạch chia áp để có thể thay đổi điện áp đầu ra bằng cách thay đổi giá trị biến trở. Với thiết kế của mạch, chia áp, nhóm cần áp dụng thêm một Opamp được cấu hình hồi tiếp âm để dễ dàng cố định điện áp vào Vi điều khiển.
- Quang trở LDR là loại cảm biến ánh sáng được sử dụng phổ biến nhất trong mạch cảm biến ánh sáng, còn được gọi là điện trở phụ thuộc vào ánh sáng (LDR), Cảm biến này hoạt động theo nguyên lý: Khi ánh sáng chiếu vào bề mặt của cảm biến, các diot tử điện trong vật liệu bán dẫn bên trong cảm biến được kích thích và tạo ra điện tử tự do. Sự tăng cường của điện tử tự do dẫn đến giảm điện trở của cảm biến. Điều này đồng nghĩa với việc mức độ ánh sáng càng cao, điện trở càng thấp, và ngược lại. Quang trở sẽ được cấu hình thành một mạch chia áp và kết hợp với Omamp để có điện áp đầu vào ổn định cho Vi điều khiển.

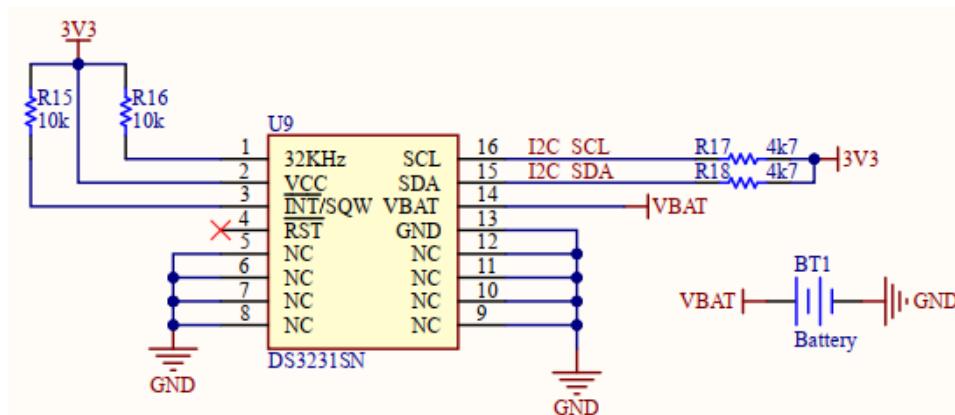


Hình 3.17: Các cảm biến trên kit thí nghiệm

3.1.4 Thiết kế khối đồng hồ thời gian thực

Trong quá trình học tập và nghiên cứu về hệ thống nhúng và phát triển ứng dụng IoT, việc thiết lập đồng hồ thời gian thực là công việc khá phổ biến hiện nay. Đồng hồ thời gian thực giúp chúng ta tiết kiệm tài nguyên của Vi điều khiển khi giờ đây, nhiệm vụ tính toán ngày giờ đã được một bộ phận bên ngoài đảm nhận. Với nguồn năng lượng thay thế, các mô-đun thời gian thực có thể tiếp tục giữ thời gian trong khi nguồn năng lượng chính bị ngắt hoặc không có.

Nhận thấy sự cần thiết của module này, nhóm đã sử dụng IC thời gian thực DS3231 để thiết kế khối chức năng này. DS3231 là IC thời gian thực thông dụng trong công nghiệp, với độ bền cao và sai số vô cùng thấp (chỉ khoảng 2s/năm khi được lắp đúng thạch anh). Kết nối với DS3231 được thiết kế như sau:



Hình 3.18: Sơ đồ nguyên lý của khối thời gian thực

Bên cạnh đó, IC DS3231 giao tiếp với chip chính thông qua giao thức I2C, đây cũng là một trong những giao thức truyền thông cơ bản trong IoT, từ đây người học có thể có thêm cơ hội để biết cách sử dụng giao thức I2C. Ngoài ra, đồng hồ

thời gian phải luôn hoạt động thì mới có ý nghĩa, điều đó có nghĩa là ta phải thiết kế cấp nguồn cho DS3231 qua pin để đồng hồ có thể hoạt động kể cả khi mạch không được cấp nguồn. Nhóm lựa chọn pin CR1220 để sử dụng vì kích thước nhỏ gọn và dễ dàng thay thế.



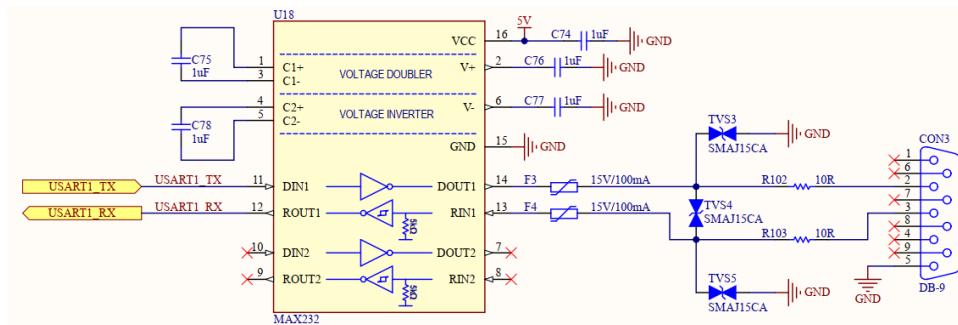
Hình 3.19: Hình ảnh pin CR1220

3.1.5 Các khối giao tiếp

Trong hệ thống nhúng, việc tương tác với các thiết bị khác trong hệ thống là điều cần thiết. Vì vậy, Kit thí nghiệm sẽ được tích hợp các khối cho phép giao tiếp bằng các chuẩn giao tiếp công nghiệp thông dụng như là RS232, RS485, CAN, INPUT-OUTPUT 12-24V, USB OTG.

3.1.5.1 RS232

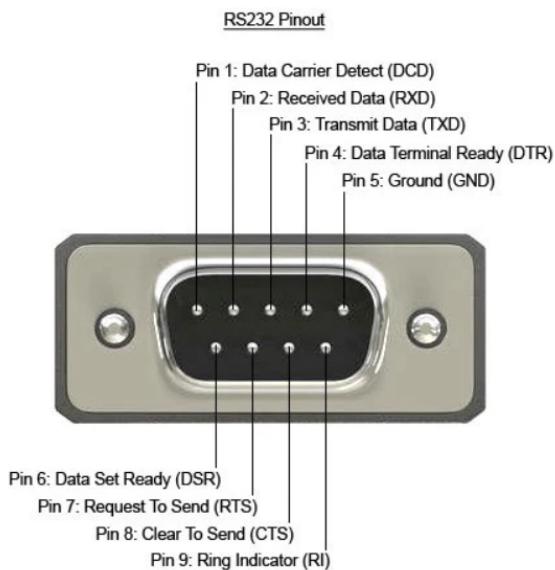
Trên kit thí nghiệm, nhóm sử dụng IC MAX232, đây là một loại IC thường được sử dụng để chuyển đổi tín hiệu RS232 từ mức cấp điện $\pm 12V$ xuống mức cấp điện $+5V$ phù hợp với hầu hết các vi xử lý và vi điều khiển hiện đại. Giao tiếp thông qua RS232 được coi là một trong những công việc đầu tiên và cơ bản khi người học tìm hiểu về UART. Dưới đây là sơ đồ nguyên lý của phần RS232.



Hình 3.20: Sơ đồ nguyên lý của kết nối RS232 trên mạch

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

Kết nối này nhóm dùng 2 chân UART, các thành phần khác như diot, cầu chì, trở và tụ được thiết kế theo yêu cầu của nhà sản xuất nhằm đảm bảo độ ổn định của tín hiệu và bảo vệ mạch.



Hình 3.21: Cổng kết nối DB9

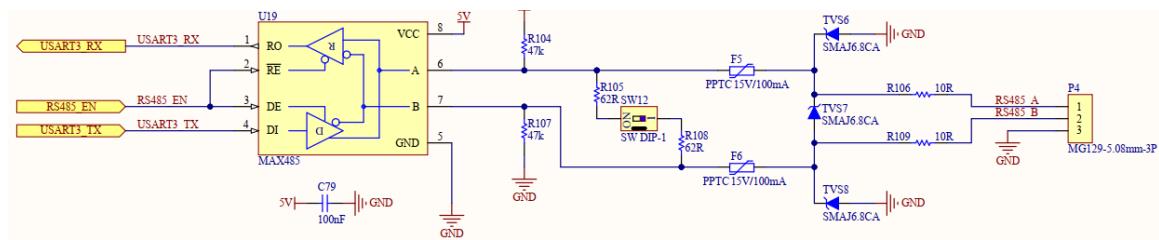
Khối RS232 được thiết kế ra chân với cổng DB9. Kết hợp với dây cáp chuyển tín hiệu từ RS232 sang USB, khối này sẽ giúp học viên học tập giao tiếp UART, và ứng dụng kiểm thử bằng cách truyền nhận dữ liệu giữa Kit thí nghiệm và máy tính thông qua RS232.

Trong thực tế, các bo mạch nhúng đa số đều hỗ trợ giao tiếp RS232 thông qua cổng DB9 với mục đích làm công cụ gỡ lỗi trong quá trình phát triển.

3.1.5.2 RS485

Như đã đề cập ở phần cơ sở lý thuyết, tùy vào mục đích của người học, nhóm đã thiết kế thêm phần kết nối RS485 với mong muốn người học có thể sử dụng kit trong các ứng dụng thực tế hơn. Trong phần này, nhóm sử dụng IC MAX485, một IC được thiết kế đặc biệt cho các ứng dụng truyền thông từ xa trong môi trường công nghiệp. Ngoài ra, trong thiết kế khối này, nhóm cũng bổ sung thêm các thành phần như các tụ trở, cầu chì, diot tương tự như ở phần RS232 để bảo vệ mạch và đảm bảo hoạt động hoạt động ổn định.

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



Hình 3.22: Sơ đồ nguyên lý của kết nối RS485 trên mạch

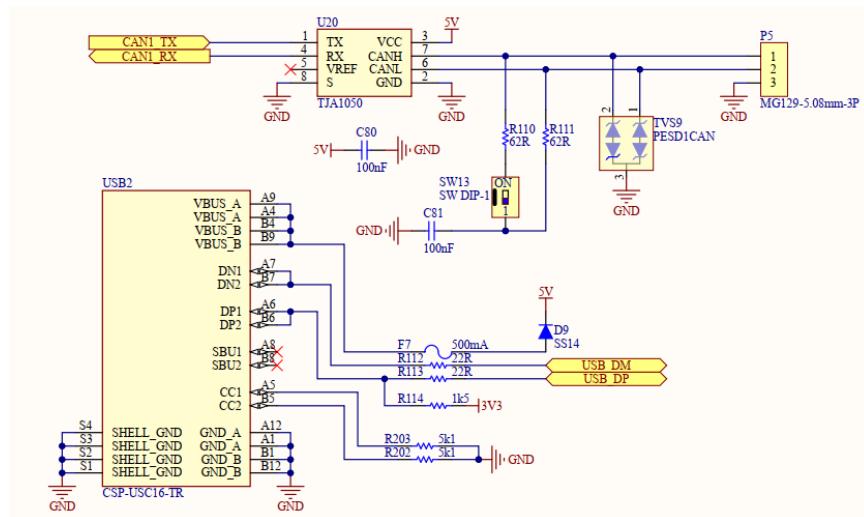
Trong phần này, nhóm thiết kế thêm switch với mục đích kiểm soát quá trình gửi nhận của dữ liệu.

3.1.5.3 CAN

Với mong muốn người dùng cũng có thể tiếp cận những giao tiếp mới phù hợp với thị trường nhóm đã thêm giao tiếp CAN vào mạch. Để hiện thực giao tiếp này, nhóm đã sử dụng IC TJA1050, đây là một IC chuyển đổi CAN chuyên dụng được sử dụng để kết nối bộ điều khiển CAN với bus vật lý trong các ứng dụng mạng điều khiển (CAN network). TJA1050 thường tích hợp trớ kéo (pull-up) và điều chỉnh nguồn cung cấp, giúp đơn giản hóa việc tích hợp vào mạch. Bên cạnh đó, IC này có khả năng điều chỉnh tốc độ truyền dữ liệu CAN để phù hợp với yêu cầu cụ thể của ứng dụng. Ngoài ra, TJA1050 còn hỗ trợ chế độ sleep và các chế độ tiết kiệm năng lượng, giảm tiêu thụ điện năng khi không có hoạt động truyền thông.

Trong phần này, nhóm sử dụng thêm IC PESD1CAN, đây là một thành phần được thiết kế đặc biệt để cung cấp bảo vệ cho hai CAN bus trong hệ. Chức năng chính của PESD1CAN là ngăn chặn thiệt hại do phóng tĩnh điện (ESD) và xung đột biến (transient) có thể gây ra. IC này được thiết kế dành riêng cho ô tô - một thị trường nhúng đang phổ biến ở Việt Nam, với độ tin cậy cao. Vì vậy nhóm cũng mong muốn người dùng có thể tiếp cận nó để phù hợp với nhu cầu thị trường. Dưới đây là sơ đồ nguyên lý của kết nối CAN trên mạch:

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

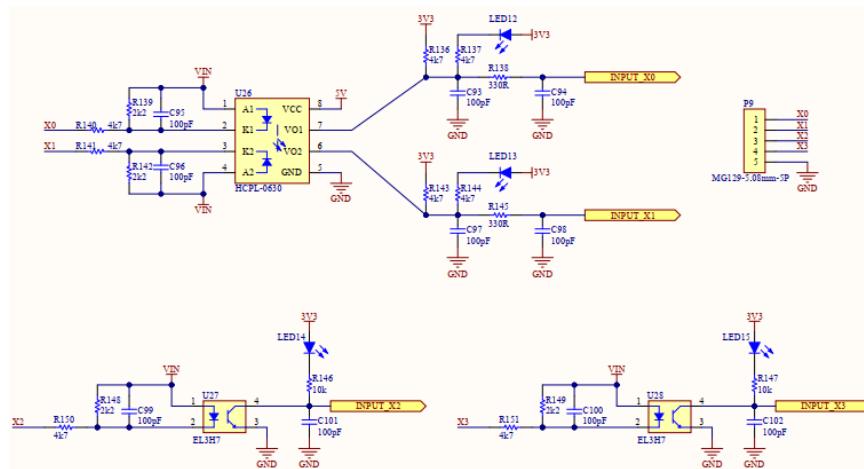


Hình 3.23: Sơ đồ nguyên lý của kết nối CAN trên mạch

3.1.5.4 INPUT - OUTPUT 24V

Với mong muốn phát triển các ứng dụng IoT, việc tích hợp thêm các input, output trên mạch phù hợp và thân thiện với nhiều thiết bị, tuân theo các chuẩn công nghiệp là một điều khá là phù hợp với người học. Để người dùng có thể thêm các thiết bị tương tác với mạch đồng thời, người dùng cũng có thể sử dụng như một bộ điều khiển rồi thông qua output, xuất ra các tín hiệu để điều khiển các thiết bị, cách mạch khác,...

Về phần Input của mạch, nhóm đã thiết kế sơ đồ nguyên lý như sau:



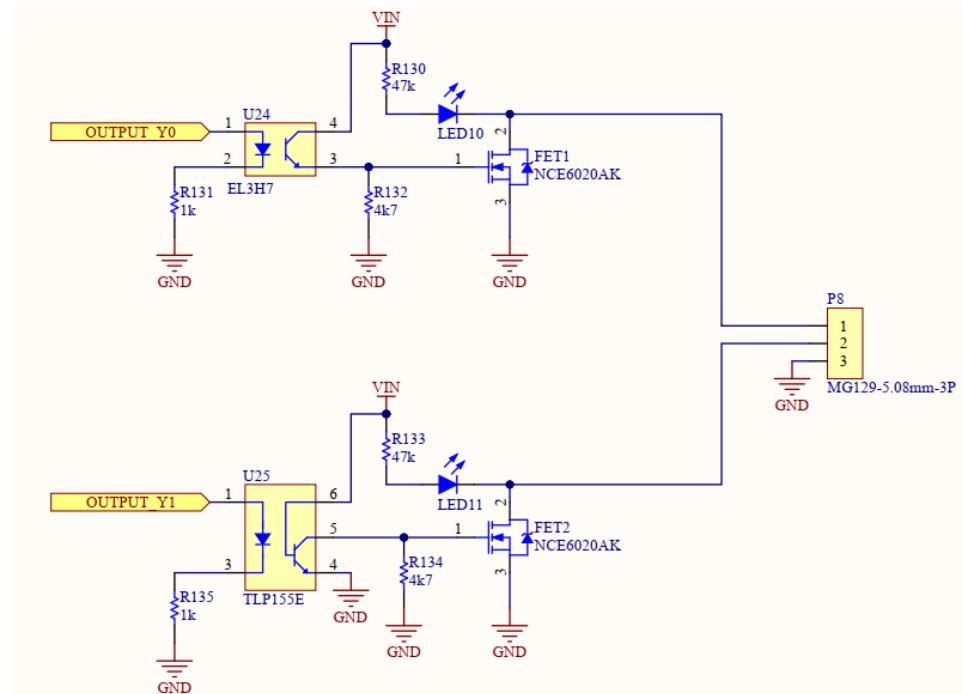
Hình 3.24: Sơ đồ nguyên lý của input trên mạch

Trong phần này, nhóm đã sử dụng 2 opto EL3H7 và 1 opto HCPL0630 để người dùng có thể tiếp cận được nhiều loại opto. Opto EL3H7 là một linh kiện quang điện tử chuyên dùng để truyền tín hiệu điều khiển giữa hai mạch điện có sự chênh

lệch cao về điện áp thông qua ánh sáng mà không cần liên hệ với nhau bằng tín hiệu điện. Opto HCPL0603 là một loại opto phù hợp cho giao tiếp logic tốc độ cao, nó sử dụng để dệm đầu vào đầu ra. Opto này được có thể hoạt động tốt trong môi khắc nghiệt.

Về phần Output trên mạch, nhóm đã sử dụng thêm Opto TLP155E, opto loại này bao gồm một tia hồng ngoại diot phát xạ và tích hợp độ lợi cao. TLP155E phù hợp trực tiếp mạch điều khiển cổng cho IGBT hoặc MOSFET điện. Opto TLP155E thường được ứng dụng trong bảng hiển thị plasma (PDP), biến tần công nghiệp, trong các mosfet.

Bên cạnh đó, để dễ dàng trong việc sử dụng, ở mỗi inout, nhóm đều sử dụng các đèn led để báo trạng thái cho người dùng tiện sử dụng. Một điểm đặc biệt trong khối inout này là nhóm đã cấu hình cho các chân khi tích cực sẽ nối đất, ngược lại sẽ thả nổi. Cách cấu hình này phù hợp với nguyên tắc điều khiển trong công nghiệp (điều khiển PLC).



Hình 3.25: Sơ đồ nguyên lý của output trên mạch

3.1.6 Khối kết nối internet

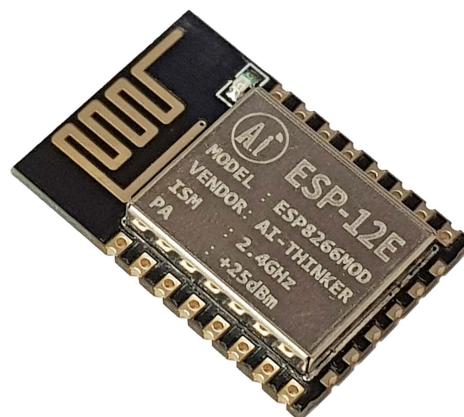
Trong phần này, nhóm sử dụng ESP8266, một module khá dễ tìm kiếm trên thị trường để người có thể tìm hiểu về quá trình kết nối Wifi. Bên cạnh đó, để người dùng có thể tiếp cận với kết nối 4G, nhóm sử dụng module Sim A7670C.

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

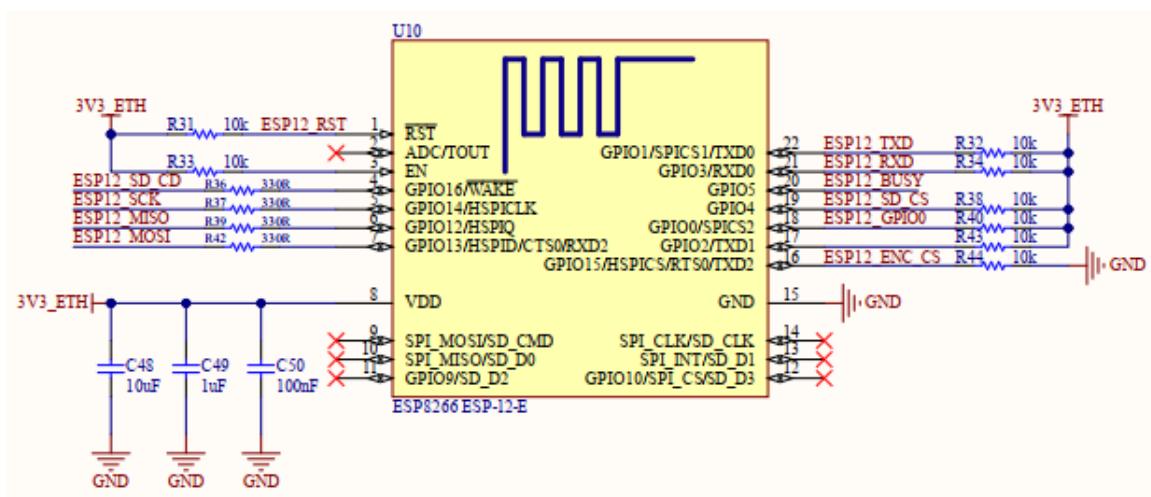
Với các khối kết nối Internet, học viên có thể dễ dàng học tập các kiến thức liên quan đến truyền nhận dữ liệu thông qua kết nối mạng trong hệ thống IoT. Cụ thể hơn, người dùng sẽ học cách giao tiếp với các server IoT thông qua các giao thức truyền thông phổ biến như HTTP và MQTT.

3.1.6.1 Module ESP8266

Mạch thu phát Wifi ESP8266 ESP-12E của Ai-Thinker là một giải pháp Wifi tiết kiệm và hiệu quả cho các ứng dụng liên quan đến Internet và Wifi. Nó tích hợp thêm bộ khuếch đại công suất (PA) và bộ lọc tín hiệu RF (LNA), cải thiện khả năng thu phát sóng Wifi. Mạch có kích thước nhỏ gọn, chất lượng đáng tin cậy, vỏ kim loại chống nhiễu và anten Wifi PCB tích hợp. ESP8266 cũng hỗ trợ giao tiếp SPI, I2C, UART và các chức năng GPIO.



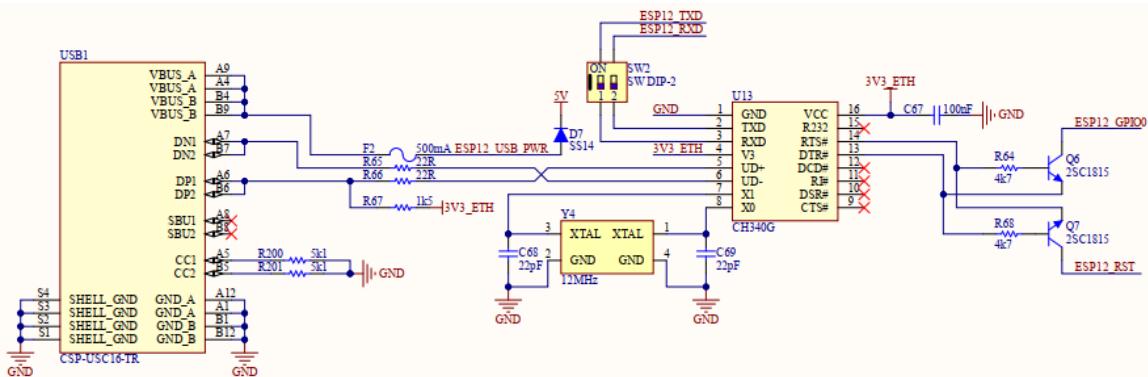
Hình 3.26: ESP8266 nhóm sử dụng trên mạch



Hình 3.27: ESP8266 nhóm sử dụng trên mạch

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

Trong khối ESP8266, nhóm đã thiết kế thêm các thành phần như khối nạp code cho ESP, sử dụng IC CH340G. CH340 là bộ chuyển đổi TTL (nối tiếp) sang USB và ngược lại. Các linh kiện liên quan còn lại được thiết kế theo nhà thiết kế.



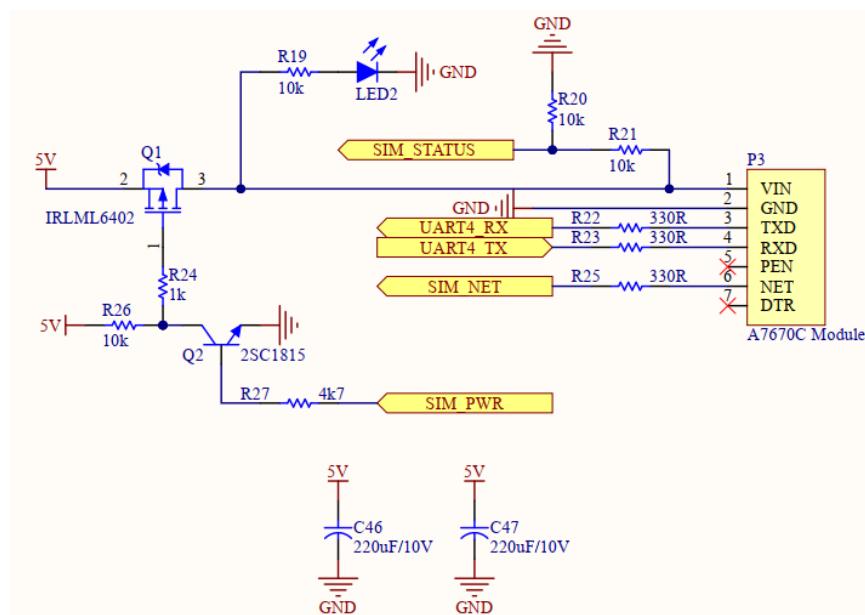
Hình 3.28: Sơ đồ nguyên lý của CH340 trên mạch

Ngoài kết nối wifi thì trên mạch có thiết kế thêm một cổng Ethernet để người dùng có thể sử dụng kết nối mạng. Để cổng này giao tiếp với mạch, nhóm sử dụng IC ENC28J60, đây là một IC Ethernet phổ biến được sử dụng trong các ứng dụng mạng nhúng. Cung cấp các tính năng quản lý mạng cơ bản như MAC (Media Access Control) và PHY (Physical Layer), có bộ nhớ đệm tích hợp để lưu trữ gói tin và dữ liệu. Đây là một IC phổ biến thường được sử dụng trong các ứng dụng mạng nhúng như vi điều khiển, máy tính nhúng, thiết bị IoT và các ứng dụng khác yêu cầu kết nối mạng.

3.1.6.2 Module sim A7670C

A7670 là Module sim được sử dụng rộng rãi trong các ứng dụng IoT với khả năng kết nối 4G tốc độ cao, và ăng ten được thiết kế sẵn. Trong khối này, nhóm sử dụng mosfet IRLM6402 để kiểm soát nguồn điện cho module Sim A7670C, giúp thêm khả năng điều khiển cấp nguồn cho module Sim thông qua Vi điều khiển chính, từ đó giúp người học viên có thể học cách tiết kiệm năng lượng khi sử dụng module Sim vì khối này tiêu thụ một lượng điện năng rất lớn. Việc điều khiển module Sim sẽ giúp người học có thể luyện tập việc giao tiếp với một thiết bị thông qua các qui tắc được đặt ra bởi nhà sản xuất thiết bị đó, cụ thể trong trường hợp này là bộ AT Command.

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG



Hình 3.29: Sơ đồ nguyên lý của kết nối module sim



Hình 3.30: Module Sim A7670C

3.1.7 Bô nhớ

Trên Kit thí nghiệm sẽ được thiết kế các bộ nhớ cơ bản cần thiết cho một ứng dụng nhúng hoàn chỉnh.

- SRAM: là viết tắt của Static Random-Access Memory, một loại bộ nhớ trong các hệ thống nhúng. Nó cung cấp khả năng lưu trữ dữ liệu và cho phép truy cập nhanh chóng đến dữ liệu đó mà không cần thời gian đợi. SRAM được sử dụng rộng rãi trong các ứng dụng nhúng như vi xử lý nhúng, viễn thông,

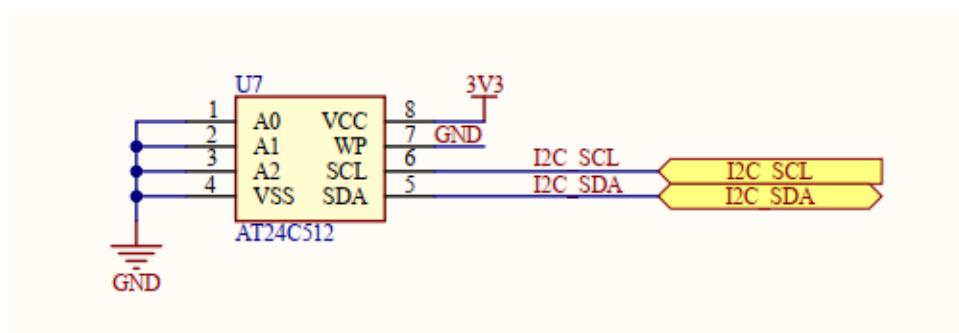
3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

thiết bị di động, và nhiều ứng dụng khác có yêu cầu về tốc độ truy xuất cao và tiêu thụ điện năng thấp. Trong hệ thống nhúng, SRAM thường được sử dụng như bộ nhớ đệm (cache memory) cho vi xử lý và bộ điều khiển. Việc sử dụng SRAM như vậy giúp giảm thời gian truy xuất dữ liệu từ bộ nhớ chính, cải thiện hiệu suất tổng thể của hệ thống và giảm tải cho bộ vi xử lý chính.



Hình 3.31: SRAM sử dụng cho LCD touch

- Eeprom: Trong thực tế, với đặc tính không mất dữ liệu lưu trữ khi ngừng cấp điện, Eeprom thường được lưu các thông tin vừa có nhu cầu lưu giữ lâu dài, vừa thay đổi được khi cần thiết như các cấu hình ban đầu của chương trình, thông số hiệu chỉnh của các cảm biến, phiên bản của chương trình,... Trong thiết kế này nhóm sẽ sử dụng Eeprom AT24C512 với khả năng lưu trữ là 512KB. IC trên sẽ giao tiếp với Vi điều khiển chính thông qua giao tiếp I2C.

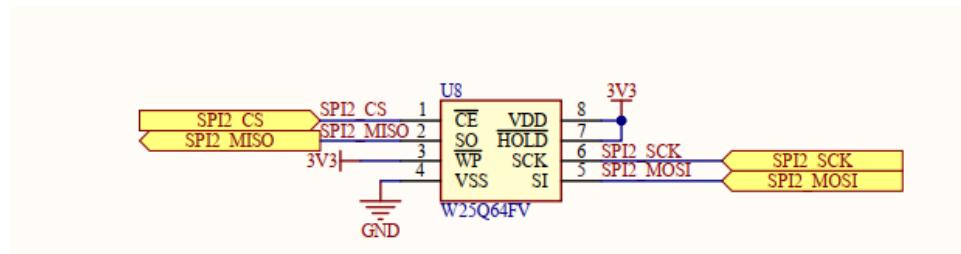


Hình 3.32: Sơ đồ nguyên lý Eeprom

- Flash: Bộ nhớ flash trong hệ thống nhúng có thể được sử dụng cho nhiều mục đích khác nhau. Đầu tiên, bộ nhớ flash ngoại có thể được sử dụng để mở rộng bộ nhớ flash bên trong MCU nhằm tăng bộ nhớ khả dụng cho mã ứng dụng. Thứ hai, Flash ngoại có thể được sử dụng để lưu trữ thông tin cấu hình hoặc dữ liệu ứng dụng. Memory map của bộ vi điều khiển có thể được cấu hình bên trong Flash ngoại để người lập trình có thể truy cập nó dễ dàng

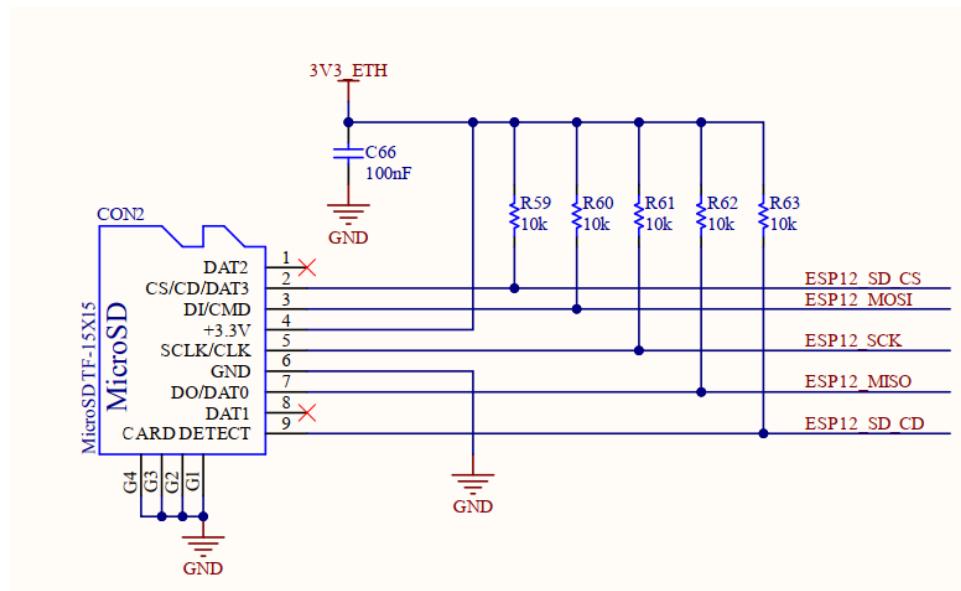
3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

hơn. Cuối cùng, Flash có thể được sử dụng để lưu trữ dữ liệu cho ứng dụng và thông tin payload.

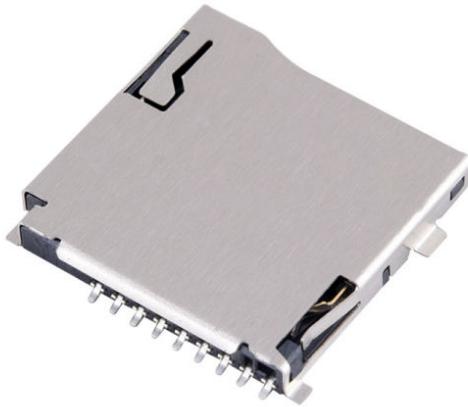


Hình 3.33: Sơ đồ nguyên lý bộ nhớ Flash

- Thẻ microSD: là một loại thẻ nhớ được sử dụng rộng rãi trong các thiết bị điện tử như máy ảnh kỹ thuật số, điện thoại di động, máy tính bảng, và các thiết bị lưu trữ khác. Thẻ microSD được thiết kế để lưu trữ dữ liệu số như hình ảnh, video, và tệp tin. Trong dự án này, khe đọc thẻ microSD sẽ được thiết kế để học viên có thể kiểm thử và chọn lựa như là một phương pháp lưu trữ bên ngoài. Việc đọc dữ liệu từ thẻ SD sẽ được ESP8266 đảm nhận thông qua giao tiếp SPI.



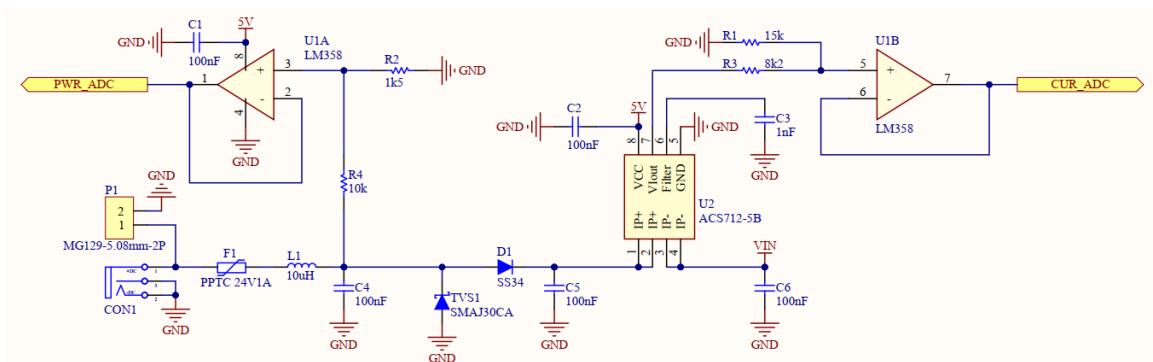
Hình 3.34: Sơ đồ nguyên lý thẻ microSD



Hình 3.35: Khe cắm thẻ microSD

3.1.8 Khối nguồn

Với các khối kẽ trên, cần phải thiết kế nguồn 12-24V đầu vào để có thể xuất tín hiệu điều khiển cho các tải. Các khối chức năng còn lại đều yêu cầu nguồn cấp 5V hoặc 3.3V. Nhóm sẽ thiết kế nguồn được cấp vào là nguồn một chiều từ Domino nguồn hoặc từ jack cắm DC. Dòng điện vào sẽ đi qua cầu chì tự phục hồi và các diode để bảo vệ mạch. Ngoài ra, nhóm sẽ cung cấp thêm các khối để phục vụ cho việc đọc giá trị điện áp và dòng cấp thông qua bộ ADC của vi điều khiển. Đối với chức năng đọc điện áp, nhóm thiết kế mạch chia áp kết hợp với OpAmp như đã trình bày ở phần khái niệm ADC. Đối với chức năng đọc dòng điện, nhóm sử dụng IC cảm biến dòng điện ACS712 để chuyển giá trị dòng điện đầu vào thành giá trị điện áp đầu ra tương ứng.

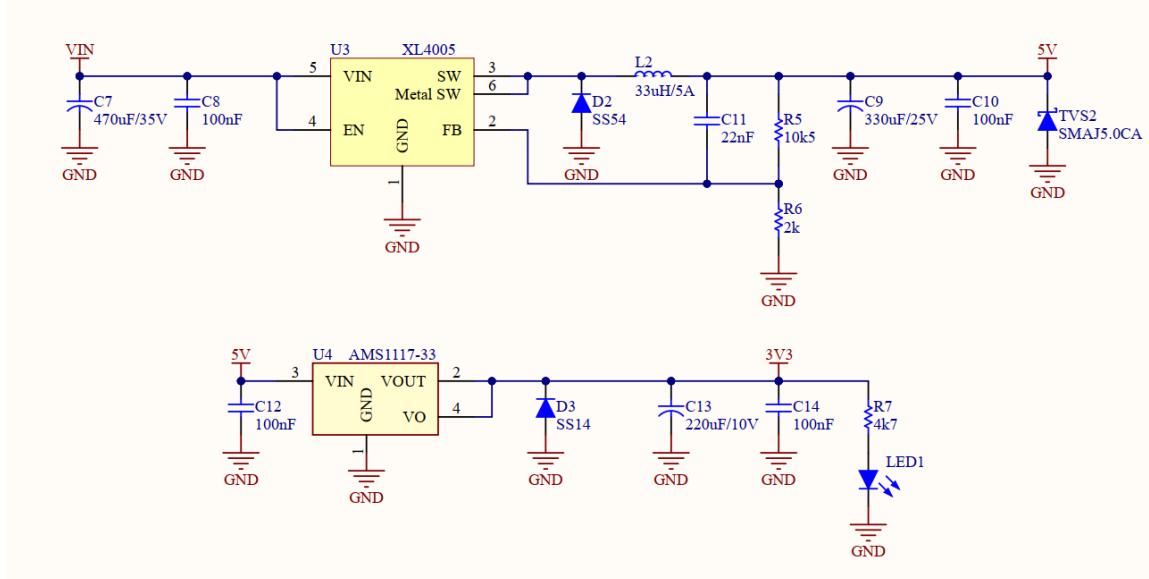


Hình 3.36: Thiết kế nguồn

Đối với nguồn cấp 5V, nhóm sử dụng IC hạ áp XL4005 với điện áp vào 5-32V và đầu ra 5V vì đây là thiết bị có thể cung cấp dòng đầu ra tối đa 5A, đảm bảo

3.1. MỤC TIÊU THIẾT KẾ PHẦN CỨNG

nguồn ổn định khi sử dụng tất cả khối chức năng trên. Sau đó nguồn 5V sẽ được vào IC hạ áp AMS1117-3V3 để tạo điện áp 3V3



Hình 3.37: Mạch hạ áp

3.2 MỤC TIÊU THIẾT KẾ TÀI LIỆU HƯỚNG DẪN

Thiết kế tài liệu hướng dẫn là quá trình tạo ra tài liệu chi tiết và dễ hiểu để hướng dẫn người dùng sử dụng Kit một cách hiệu quả. Đây là công cụ quan trọng giúp người dùng nắm được cách sử dụng sản phẩm một cách tự tin và dễ dàng. Các điểm quan trọng mà nhóm hướng tới của quá trình này bao gồm:

1. Xác định mục tiêu: Mục tiêu nhóm hướng tới là giúp cả người mới bắt đầu và những người có kinh nghiệm có thể dễ dàng tiếp cận và sử dụng kit thí nghiệm một cách hiệu quả. Bằng cách cung cấp ví dụ thực tế và hướng dẫn chi tiết, tài liệu sẽ giúp người dùng nắm vững kiến thức và kỹ năng cần thiết để thực hiện các thí nghiệm và dự án IoT của riêng họ.
2. Phân loại tài liệu: chúng tôi đã phân loại các loại tài liệu khác nhau dựa trên nội dung và mục tiêu sử dụng. Đầu tiên là Quick start guide, đây có thể là các hướng dẫn sơ bộ, gồm các bước cơ bản để config và chạy thử nghiệm với kit thí nghiệm một cách nhanh chóng, thích hợp với những người chưa có kinh nghiệm đọc datasheet và cả những người đã có nhiều kinh nghiệm, cần config nhanh chóng. Tài liệu cho vi điều khiển: Tài liệu này tập trung vào các khái niệm và kỹ thuật cơ bản của vi điều khiển về gpio, quét led, timer interrupt, về adc-pwm, uart, spi, i2c, mở rộng trong phần này là bài hướng dẫn về iot thông qua esp8266. Về tài liệu cho RTOS (Real-Time Operating System): Đây là các tài liệu tập trung vào việc sử dụng hệ điều hành thời gian thực (RTOS) để phát triển các ứng dụng nhúng. Nó cung cấp cho người dùng những khái niệm cơ bản về RTOS trên stm32, tài liệu này hiện đang trong quá trình hoàn thiện nên còn khá cơ bản và chưa có nhiều chỉnh sửa, góp ý sâu sắc.
3. Về phần Sử dụng hình ảnh và minh họa: Trong quá trình tạo tài liệu hướng dẫn, việc sử dụng hình ảnh và minh họa đóng vai trò quan trọng để làm cho nội dung trở nên sinh động và dễ hiểu hơn. Chúng tôi sẽ sử dụng hình ảnh và biểu đồ chi tiết để minh họa từng bước cấu hình và chạy kiểm thử, giúp người đọc có cái nhìn rõ ràng và cụ thể. Các hình ảnh sẽ tập trung vào từng ví dụ cụ thể, bao gồm cả các bước cài đặt phần mềm hỗ trợ và các bước chạy kiểm thử. Điều này giúp người đọc dễ dàng hình dung và thực hiện các thao tác một cách chính xác và hiệu quả.

4. Kiểm tra và cập nhật: Trong quá trình làm tài liệu, nhóm đã kiểm tra kỹ lưỡng tài liệu, nhờ giảng viên hướng dẫn kiểm tra tài liệu trước khi phát hành và cập nhật thường xuyên để đảm bảo nó luôn phản ánh đúng thông tin mới nhất.
5. Quá trình thiết kế tài liệu hướng dẫn đòi hỏi sự cẩn nhắc và công phu để đảm bảo rằng người dùng có thể tận dụng sản phẩm một cách tối ưu và không gặp khó khăn trong quá trình sử dụng, tuy nhiên, do yêu cầu về tính chính xác và tính dễ hiểu cao nên hiện tại, chỉ có quick start guide và tài liệu cho vi điều khiển(có mở rộng IoT) đã đưa vào giảng dạy, còn tài liệu của RTOS vẫn đang trong quá trình xây dựng.

3.3 MỤC TIÊU THIẾT KẾ MÁY CHỦ

Trong phần này nhóm sẽ trình bày về quá trình nhóm thiết kế, xây dựng một server backend để hỗ trợ người dùng trong việc hiện thực các project IoT.

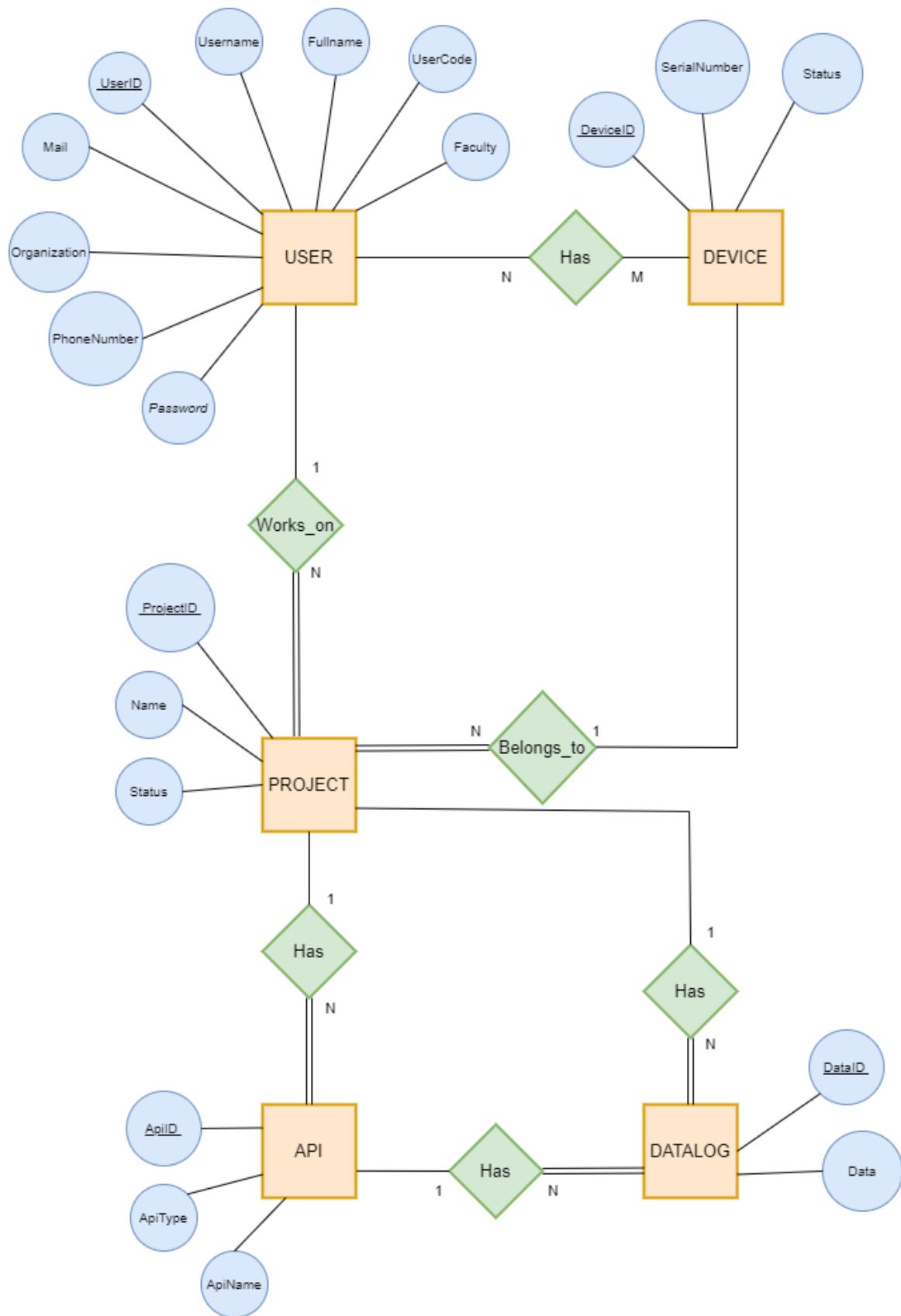
3.3.1 Mô tả nghiệp vụ server IoT

- Cơ sở dữ liệu lưu trữ các thông tin về users, projects, devices, data, apis.
- Cơ sở dữ liệu lưu trữ thông tin về USER như sau: Username (độc nhất), Fullname, Password, Mail (độc nhất),UserCode, Faculty, Organization, PhoneNumber (độc nhất). Một USER có thể tạo nhiều PROJECTs. Một USER có thể sở hữu nhiều DEVICEs
- Cơ sở dữ liệu lưu trữ thông tin về DEVICE như sau: SerialNumber, Status (trạng thái của sản phẩm: tồn kho, đã bán, khóa). Một DEVICE có thể thuộc về nhiều USERs. Một DEVICE có thể được sử dụng trong nhiều PROJECTs.
- Cơ sở dữ liệu lưu trữ thông tin về PROJECT như sau: ProjectName, Status (kích hoạt, ngưng kích hoạt, đã xóa). Một PROJECT bắt buộc phải thuộc về một USER. Một PROJECT bắt buộc phải thuộc về một USER. Một PROJECT bắt buộc phải thuộc về một DEVICE. Một PROJECT có thể có nhiều DATALOGs. Một PROJECT có thể có nhiều APIs.
- Cơ sở dữ liệu lưu trữ thông tin về DATALOG như sau: Data. Một DATALOG bắt buộc phải thuộc về một PROJECT. Một DATALOG bắt buộc phải thuộc về một API. Một PROJECT bắt buộc phải thuộc về một DEVICE.

3.3. MỤC TIÊU THIẾT KẾ MÁY CHỦ

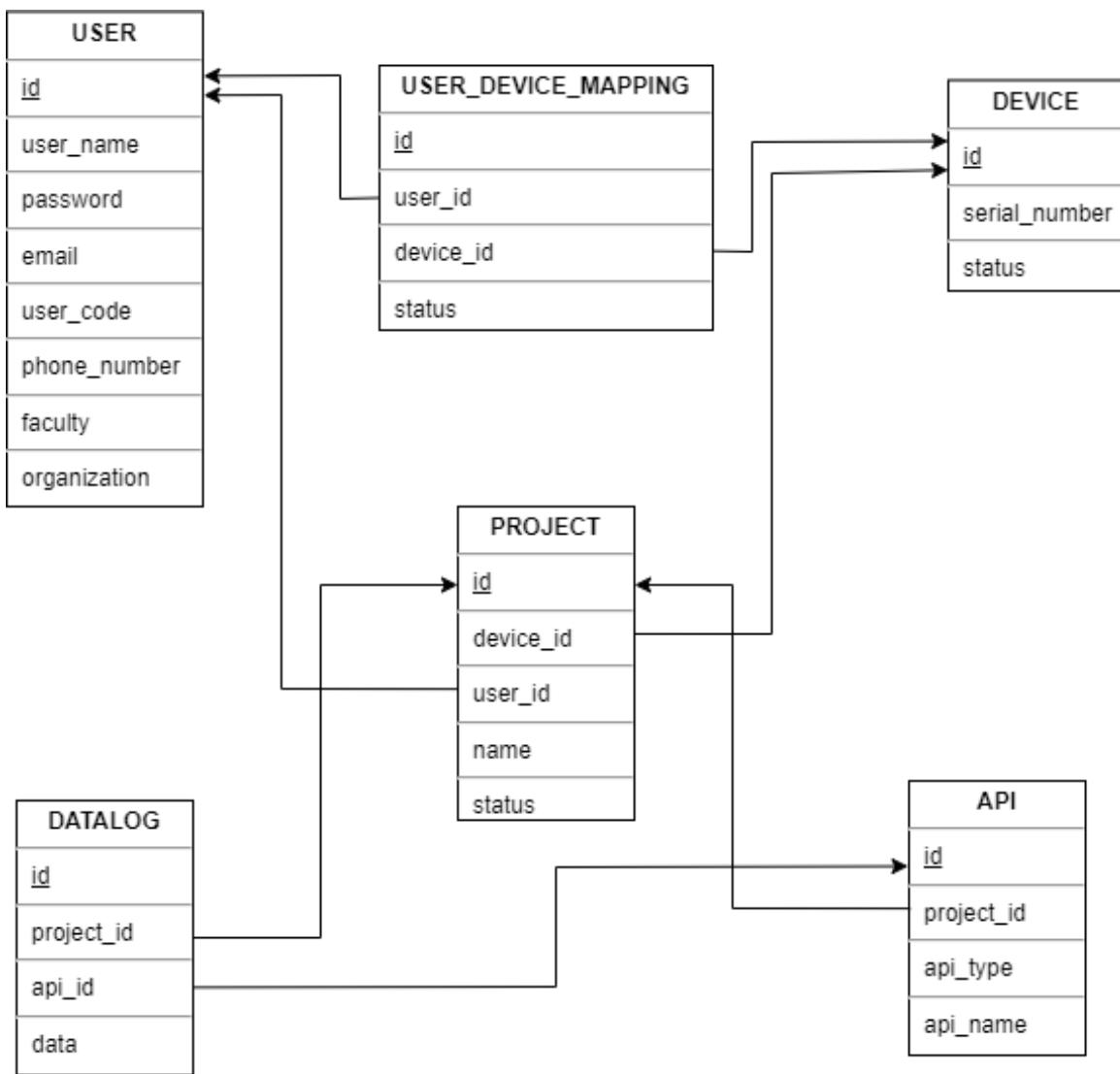
- Cơ sở dữ liệu lưu trữ thông tin về API như sau: ApiType, ApiName. Một API bắt buộc phải thuộc về một PROJECT. Một API có thể có nhiều DATALOGs.

3.3.2 Sơ đồ thiết kế



Hình 3.38: Sơ đồ ERD

3.3. MỤC TIÊU THIẾT KẾ MÁY CHỦ



Hình 3.39: Lược đồ các bảng dữ liệu

3.3. MỤC TIÊU THIẾT KẾ MÁY CHỦ

Mô tả bảng USER:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
user_name	String	Khác NULL Duy nhất	Tên đăng nhập
password	String	Khác NULL	Mật khẩu người dùng
email	String	Khác NULL Duy nhất	Email người dùng
full_name	String	Khác NULL	Họ và tên người dùng
user_code	String	Duy nhất	Mã số sinh viên
phone_number	String	Khác NULL Duy nhất	Số điện thoại
faculty	String		Khoa
organization	String		Tổ chức

Bảng 3.2: Bảng USER

Mô tả bảng DEVICE:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
serial_number	String	Khác NULL Duy nhất	Số seri của thiết bị
status	Integer	Khác NULL	Trạng thái của thiết bị

Bảng 3.3: Bảng DEVICE

3.3. MỤC TIÊU THIẾT KẾ MÁY CHỦ

Mô tả bảng USER_DEVICE_MAPPING:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
user_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng USER
device_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng DEVICE

Bảng 3.4: Bảng USER_DEVICE_MAPPING

Mô tả bảng PROJECT:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
user_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng USER
device_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng DEVICE
name	String	Khác NULL	Tên dự án
status	String	Khác NULL	Trạng thái của dự án

Bảng 3.5: Bảng PROJECT

Mô tả bảng API:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
api_type	String	Khác NULL	Loại API
project_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng PROJECT
api_name	String	Khác NULL	Tên API

Bảng 3.6: Bảng API

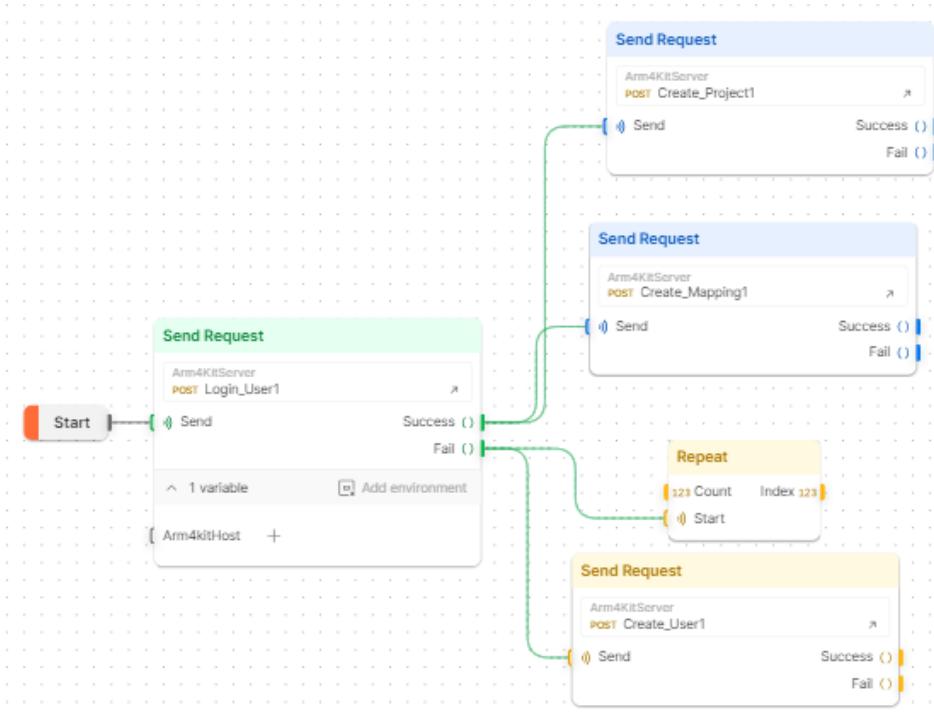
Mô tả bảng DATALOG:

Thuộc tính	Kiểu dữ liệu	Ràng buộc	Ý nghĩa
id	Integer	Khác NULL Duy nhất	Khóa chính, dùng để phân biệt giữa các collection
api_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng API
project_id	Integer	Khác NULL	Khóa ngoại liên kết tới bảng PROJECT
data	JSON	Khác NULL	Dữ liệu người dùng gửi lên

Bảng 3.7: Bảng DATALOG

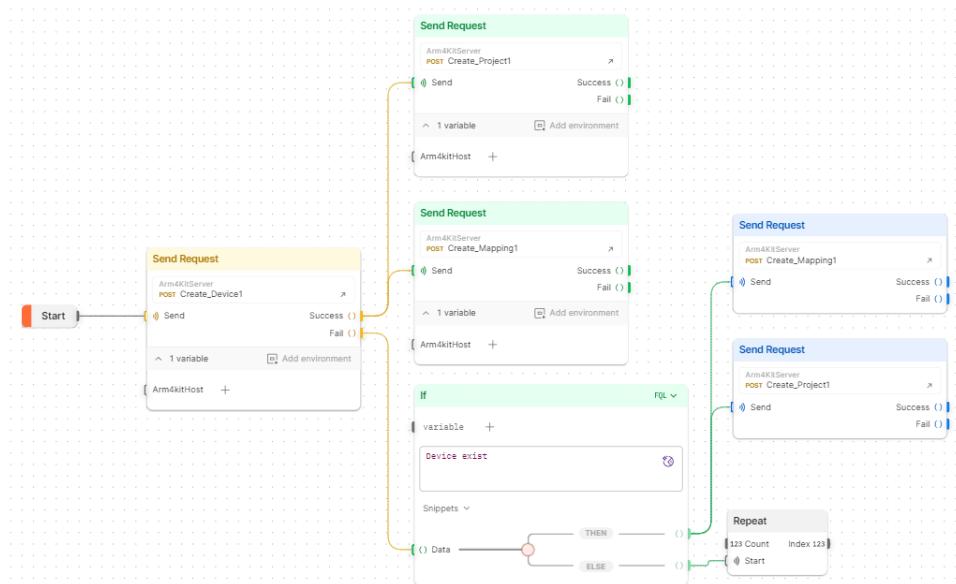
3.3.3 Workflow của hệ thống

Flow của User



Hình 3.40: Flow work của User

Flow của Device



Hình 3.41: Flow work của Device

3.4 VỀ PHƯƠNG PHÁP THIẾT KẾ PHẦN CỨNG

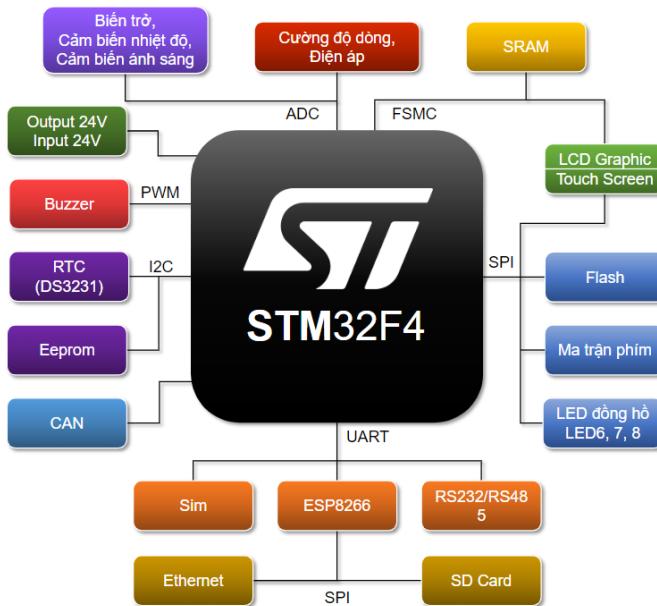
3.4.1 Giới thiệu

Sau khi đã tính toán và lựa chọn thiết bị cụ thể, ta sẽ bước sang giai đoạn sau cùng là thi công hệ thống. Về phần cứng, nhóm thực hiện thiết kế trên phần mềm Altium Designer, đây là phần mềm thiết kế mạch điện và PCB phổ biến nhất hiện nay. Phần lớn linh kiện và mô-đun có trong hệ thống đều có sẵn trên thị trường, riêng phần mạch điện phải đặt làm bởi một bên thứ ba ở Đài Loan nên việc chỉnh sửa sau thiết kế khá khó khăn.

Đối với phần Firmware, nhóm đã sử dụng công cụ lập trình chuyên dụng cho các dòng chip STM32 đó là STM32CubeIDE, công cụ này có hỗ trợ bộ thư viện HAL giúp người dùng có thể tương tác dễ dàng với các chức năng của Vi điều khiển đồng thời cấu hình Input/Output như mong muốn. Ngoài ra để có thể lập trình cho mô-đun ESP8266, nhóm đã sử dụng Arduino IDE để tiến hành dễ dàng hơn.

Dưới đây là diagram về các phần tử có trên mạch:

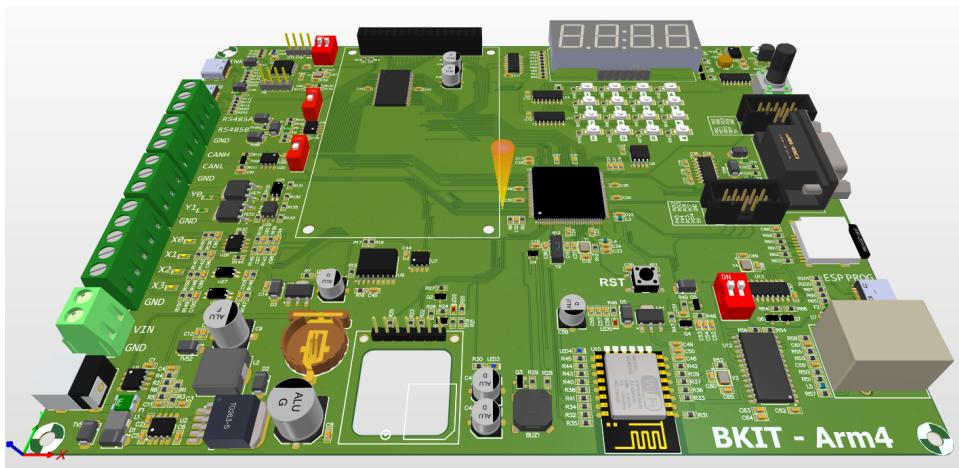
3.4. VỀ PHƯƠNG PHÁP THIẾT KẾ PHẦN CỨNG



Hình 3.42: Các yêu cầu cơ bản mà nhóm định hướng

3.4.2 Thi công board mạch

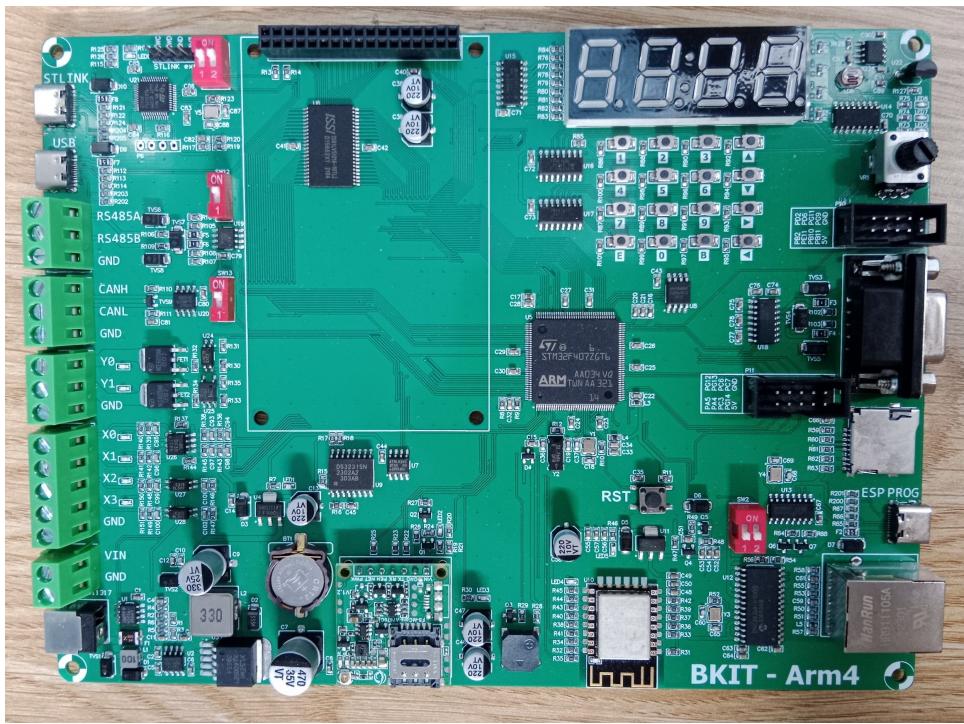
Từ sơ đồ nguyên lý đã được thiết kế ở Chương 3, cùng với sự hỗ trợ của phần mềm Altium Designer nhóm đã có thể thiết kế được PCB cơ bản của Kit thí nghiệm bao gồm đầy đủ các chức năng đã nêu và được trực quan hóa trong hình 3.42. Mạch mà nhóm thiết kế là mạch 2 lớp và đặt linh kiện hoàn toàn trên một mặt, dưới đây là hình ảnh



Hình 3.43: Mô phỏng của mạch trên Altium Designer

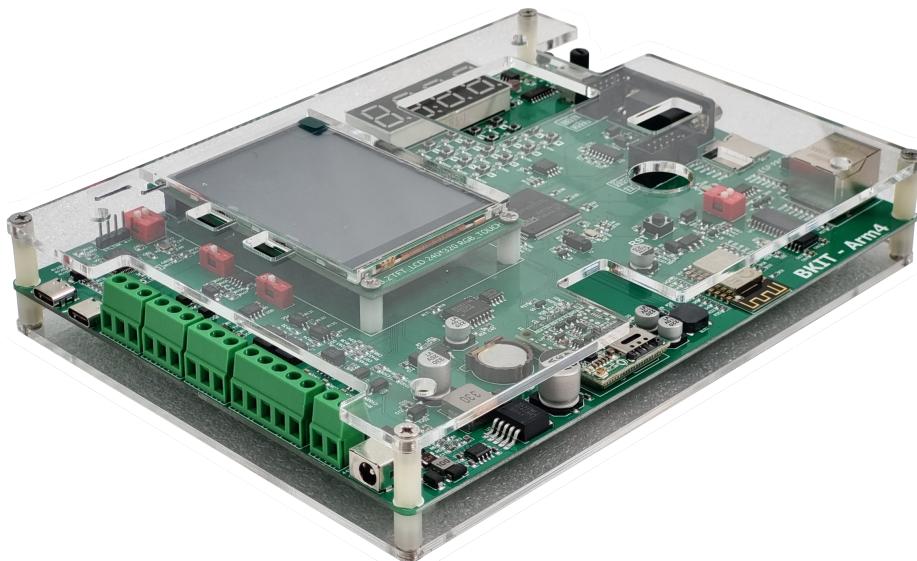
Hình ảnh của Kit thí nghiệm sau khi hoàn thiện xong:

3.4. VỀ PHƯƠNG PHÁP THIẾT KẾ PHẦN CỨNG



Hình 3.44: Hình ảnh hoàn thiện mạch của Kit thí nghiệm

Để bảo vệ mạch trước các tác động vật lý, tránh các va đập và dễ dàng vận chuyển đồng thời để những người mới sử dụng không làm chập mạch một cách đáng tiếc, nhóm quyết định đóng gói mạch bên trong hai lớp mica bảo vệ. Hình ảnh của kit thí nghiệm sau khi đóng gói:



Hình 3.45: Hình ảnh của kit thí nghiệm

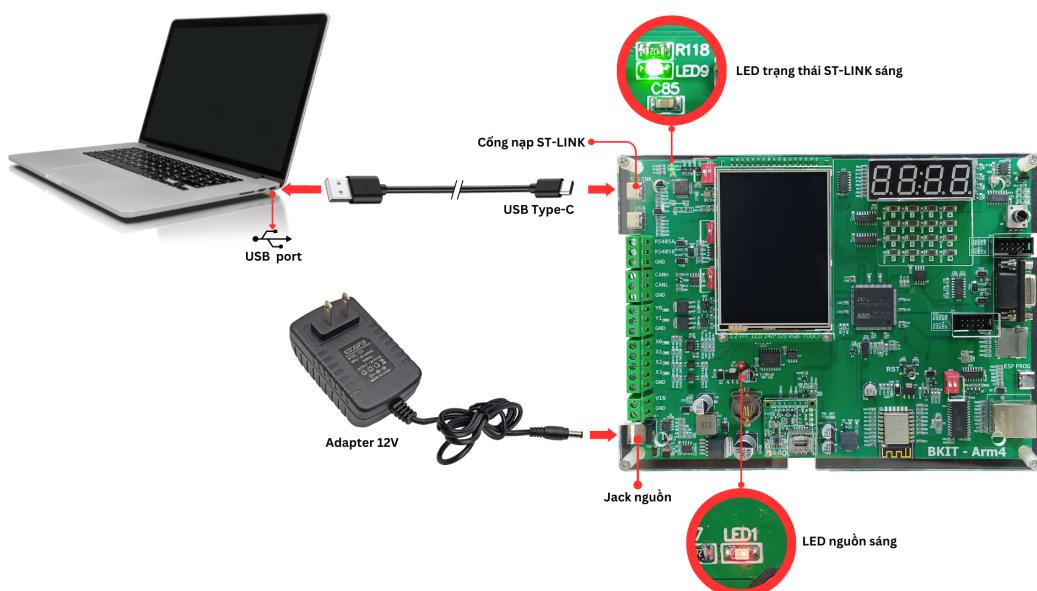
3.5 VỀ PHƯƠNG PHÁP HƯỚNG DẪN SỬ DỤNG

Trong phần này, để người dùng có những cái nhìn tổng quan nhất về việc sử dụng kit, nhóm đã viết một hướng dẫn sử dụng nhanh mô tả chi tiết về các thành phần chính trên mạch, cách config các chân của chip chính phù hợp với những người đã có kiến thức về vi điều khiển từ trước. Từ mô tả này, người dùng có thể thấy được chi tiết những linh kiện ở trên mạch một cách nhanh chóng, đồng thời có thể sử dụng ngay mạch chỉ trong 26 trang hướng dẫn.

Trong phần hướng dẫn này nhóm cũng đã đề cập đến đầy đủ về cách kết nối nguồn, cách cài đặt driver, hiện thực project đầu tiên trên kit thí nghiệm. Qua hướng dẫn này thì người dùng cũng có thể biết được tất cả các cách để kết nối với ngoại vi thông qua UART, SPI, RS232, RS485,... thì phải config thế nào.

Việc viết hướng dẫn này giúp người dùng tiếp cận kit một cách nhanh chóng và dễ dàng, đồng thời giúp họ tự tin hơn khi thực hiện các dự án và thí nghiệm của mình. Chi tiết về hướng dẫn nhóm sẽ để trong mục tài liệu tham khảo [?]

Dưới đây là đầy đủ một bộ Kit khi được hoàn thiện:



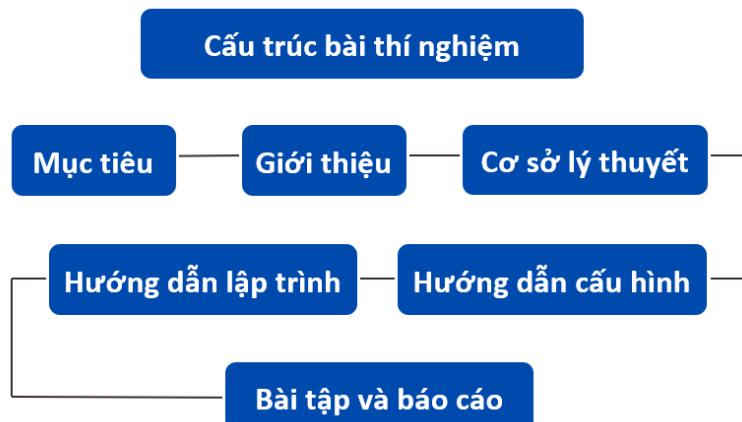
Hình 3.46: Kết nối các sản phẩm có trong bộ Kit thí nghiệm

3.5.1 Các kiến thức sử dụng trong các bài thí nghiệm

Phần này sẽ giải thích và mô tả những kiến thức quan trọng mà bài thí nghiệm hướng tới. Từ những khái niệm cơ bản đến những ứng dụng phức tạp, người học sẽ có cơ hội hiểu rõ và áp dụng kiến thức thu được từ mỗi bài thí nghiệm.

3.5.2 Quy trình thiết kế bài thí nghiệm

Để người học có thể tiếp cận được toàn diện kiến thức, nhóm sẽ thiết kế bài thí nghiệm dựa trên cấu trúc như sau:



Hình 3.47: Cấu trúc bài thí nghiệm

- Mục tiêu: Bài thí nghiệm đặt ra những mục tiêu rõ ràng và chi tiết, nhấn mạnh vào những kỹ năng và kiến thức mà người học sẽ đạt được sau khi hoàn thành. Cụ thể, mục tiêu có thể bao gồm việc hiểu rõ về vi điều khiển, phát triển kỹ năng lập trình, và khả năng áp dụng kiến thức vào các ứng dụng với mong muốn có cái nhìn rõ ràng hơn trong lĩnh vực hệ thống nhúng và IoT.
- Giới thiệu: giới thiệu về bài thí nghiệm, giúp người dùng hiểu rõ hơn những vấn đề cần tìm hiểu trong bài thí nghiệm thậm chí là những linh kiện thường xuyên được sử dụng trong thực tế mà nhóm đặt trong bài thí nghiệm.
- Cơ sở lý thuyết: đề cập đến các kiến thức chính, giải thích chi tiết về các kiến thức đó, giúp người học có thể dễ dàng hiểu được tường tận và chi tiết các nội dung có trong bài thí nghiệm.
- Hướng dẫn cấu hình: trong phần này, người dùng sẽ tiến hành cấu hình cho IC chính thông qua STM32CubeMX. Thông qua đây người dùng sẽ thấy được

3.5. VỀ PHƯƠNG PHÁP HƯỚNG DẪN SỬ DỤNG

những lý thuyết ở thực tế sẽ áp dụng vào IC sẽ được sử dụng cụ thể như thế nào.

- Hướng dẫn lập trình: người dùng sẽ sử dụng STM32CubeIDE và thư viện HAL để lập trình cho Kit thí nghiệm. Là một trong những phần quan trọng mà những người tiếp cận với hệ thống nhúng và phát triển IoT đều phải tìm hiểu. Thông qua đây người dùng sẽ biết cách làm thế nào để vi điều khiển sẽ giao tiếp được với các module trên mạch, các thiết bị IoT bên ngoài.
- Bài tập và báo cáo: Trong phần này sẽ có những yêu cầu để người học có thể luyện tập thêm, hiểu rõ thêm các vấn đề, nội dung của bài thí nghiệm, từ đó tạo điều kiện cho người dùng có thể sáng tạo trên kit thí nghiệm.

3.5.3 Các nội dung của bài thí nghiệm

Hiện tại, nhóm đã thiết kế Bộ 8 bài thí nghiệm với các nội dung xoay quanh các vấn đề về Vi điều khiển cơ bản, tiền dồn tới Hệ thống nhúng và phát triển IoT:



Hình 3.48: Nội dung tài liệu hướng dẫn

- GPIO: Trong bài này, nhóm chúng tôi sẽ hướng dẫn người dùng sử dụng STM32 Cube IDE để xây dựng ứng dụng trên vi điều khiển, người dùng sẽ biết cách config và lập trình các chân GPIO và biết cách điều khiển các ngoại vi liên quan đến GPIO trên kit thí nghiệm để mô phỏng các bài toán thực tế như bài toán đèn giao thông.
- Timer Interrupt and LED Scanning: Trong bài này, nhóm hướng tới người dùng sẽ tìm hiểu về khái niệm interrupt, biết cách sử dụng một loại interrupt nổi bật đó là timer, ứng dụng timer vào các bài toán thời gian thực, trước

3.5. VỀ PHƯƠNG PHÁP HƯỚNG DẪN SỬ DỤNG

hết là trên kit thí nghiệm. Led 7 đoạn là một module phổ biến trong thực tế, gần gũi với các bài toán IoT, nhóm mong muốn ngay từ bài thứ 2 này người dùng có thể biết cách cấu hình và sử dụng thư viện LED bảy đoạn trên kit thí nghiệm.

- LCD and Button matrix: Trong bài này, nhóm yêu cầu người dùng hiểu về nguyên lý hoạt động của nút nhấn sử dụng điện trở kéo lên, nguyên lý hoạt động và cách điều khiển LCD, biết cách cấu hình sử dụng ma trận phím và LCD trên Kit thí nghiệm. Nút nhấn và xử lý nhiều cho nút nhấn là những bài toán quan trọng mà người học về Hệ thống nhúng và phát triển IoT cần nắm được.
- Real Time Clock(RTC): Đối với vấn đề này, nhóm sẽ giới thiệu cho người dùng về IC RTC DS3231. Như ở phần cơ sở lý thuyết đã đề cập, IC này giao tiếp với chip chính thông qua giao thức I2C. Ở bài này, nhóm yêu cầu người dùng sẽ tìm hiểu kiến thức về máy trạng thái, một khái niệm phổ biến trong lập trình vi điều khiển.
- UART: RS232, một giao thức truyền thông lâu đời, vẫn được sử dụng phổ biến trong các ứng dụng truyền thông dữ liệu. Trên kit thí nghiệm, người dùng sẽ được hướng dẫn cách sử dụng RS232 để thiết lập giao tiếp UART giữa vi xử lý và máy tính. Thông qua việc sử dụng RS232, người dùng có thể trải nghiệm giao tiếp UART và áp dụng kiến thức này trong các dự án nhúng của mình. Các ứng dụng hỗ trợ như Serial Debug Assistant hay Hercules Terminal sẽ giúp người dùng theo dõi và gửi dữ liệu một cách dễ dàng. Việc này không chỉ mang lại sự thuận tiện trong quá trình thử nghiệm mà còn giúp người dùng hiểu rõ hơn về truyền thông dữ liệu và cách nó hoạt động trong các ứng dụng thực tế.
- ADC-PWM: Tìm hiểu về khái niệm adc và pwm, đây là những lý thuyết cơ bản trong Hệ thống nhúng hay phát triển IoT, thậm chí ngay cả khi tiếp xúc với vi điều khiển, đây cũng là những khái niệm quan trọng. Trong bài thí nghiệm này, nhóm sẽ giới thiệu về hai khái niệm này và hướng dẫn người dùng sử dụng chúng trong mạch trong những ứng dụng cụ thể trên các cảm biến nhiệt độ, ánh sáng, biến trở, trên buzzer. Nhóm cũng xây dựng một yêu cầu nâng cao để người dùng có thể kết hợp kiến thức nhiều bài lại với nhau.
- LCD Touch: Trong phần này, người dùng sẽ được tìm hiểu về 2 loại touch, về bộ nhớ EEPROM AT24C. Người dùng sẽ tiếp cận với LCD từ cơ bản nhất như hiển thị đến phức tạp như các tương tác chạm. Để màn hình có thể hoạt

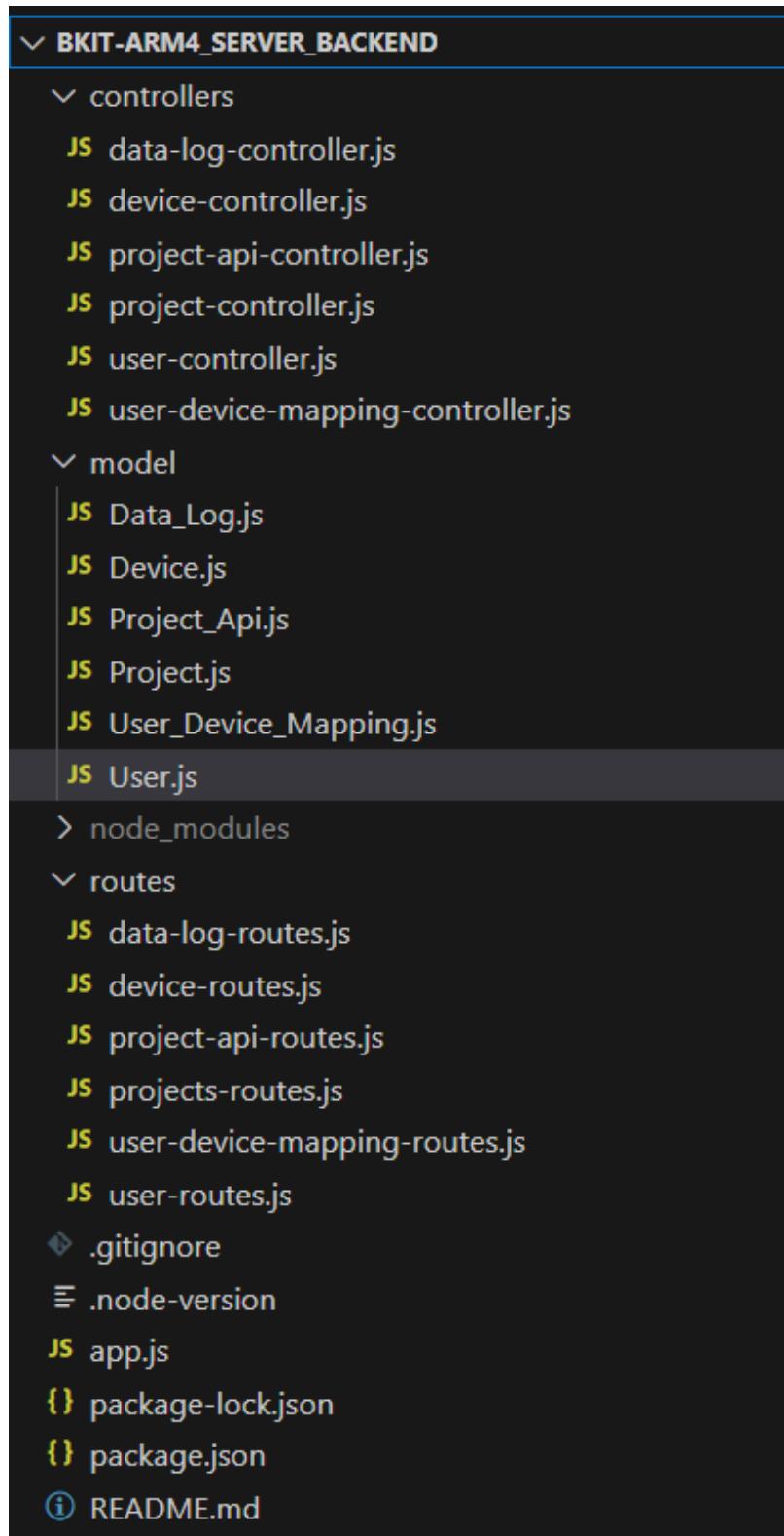
động tốt hơn, nhóm đã sử dụng thêm bộ nhớ, đây cũng là cơ hội để người dùng tiếp cận với bộ nhớ và có thêm một phần kiến thức về bộ nhớ và các ứng dụng liên quan. Bài thí nghiệm này yêu cầu người dùng thiết kế trò chơi con rắn, từ đó làm cho người dùng cho thêm hứng thú tìm hiểu cũng như có thêm khả năng sáng tạo trên Kit thí nghiệm để đưa ra nhiều ứng dụng thú vị và thực tế hơn.

- ESP8266 - WIFI: Ở bài thí nghiệm cuối cùng này nhóm sẽ giới thiệu đến người dùng về ESP8266 ESP-12 của Ai-Thinker, sử dụng ESP8266 trên mạch và biết cách giao tiếp với chip chính. Hiện thực kết nối Wifi và điều khiển thông qua Adafruit bằng giao thức MQTT một cách đơn giản, từ đó người nhìn sẽ có những khái niệm rõ ràng hơn về IoT. Trong phần này nhóm cũng giới thiệu kiến trúc 5 lớp của IoT, từ kiến trúc này người dùng sẽ có cái nhìn sâu sắc và tổng quan hơn về IoT.
- RTOS: Trong phần hướng dẫn này, chúng tôi sẽ trình bày một cách chi tiết và sâu rộng về Hệ điều hành thời gian thực, hay còn được biết đến với tên gọi là RTOS. Chúng tôi sẽ cung cấp cho bạn những kiến thức lý thuyết cần thiết, đồng thời hướng dẫn thực hành thông qua các bài tập trình cơ bản. Các chủ đề chính mà chúng tôi sẽ tập trung vào bao gồm: Task (nhiệm vụ), Queue (hàng đợi), Mutex (đối tượng đồng bộ hóa đa luồng), và Semaphore (bộ đếm).

3.6 PHƯƠNG PHÁP THI CÔNG SERVER

3.6.1 Thiết kế API

Tài liệu về các API có thể được xem tại đây .



Hình 3.49: Cấu trúc thư mục xây dựng server

CHƯƠNG 4

KIỂM QUẢ - THẢO LUẬN

4.1 Kết quả kiểm thử và đánh giá phần cứng

Kết quả kiểm thử phần cứng (tất cả kết quả dưới đây thu được khi kiểm thử với điều kiện bo mạch được cấp nguồn 12V-2A, tần số hoạt động của STM32F4 là 168MHz)

1. LED 7 đoạn:

- Kịch bản kiểm thử: Lập trình điều khiển LED 7 đoạn. Thay đổi tần số truyền dữ liệu với IC thanh ghi dịch, từ lớn tới nhỏ cho tới khi LED 7 đoạn hiển thị theo ý muốn. Thay đổi tần số quét tăng dần từ nhỏ đến lớn cho đến khi khởi LED 7 đoạn hoạt động ổn định. Mục đích để xác định tần số quét hợp lý.
- Kết quả: Tần số quét LED hợp lý là 1KHz.

2. RTC DS3231:

- Kịch bản kiểm thử: Cấp nguồn và lập trình lưu thời gian hiện tại vào bộ nhớ của DS3231. Sau 72 giờ kiểm tra sự sai lệch của thời gian so với đồng hồ trên điện thoại di động.
- Kết quả: Sai lệch 1 giây.

3. Màn hình LCD:

- Kịch bản kiểm thử: Lập trình hiển thị toàn màn hình lần lượt một số màu. Thay đổi tần số quét của LCD từ nhỏ tới lớn. Mục đích kiểm tra các điểm ảnh, tần số quét tối đa, tần số hoạt động ổn định.
- Kết quả: Các điểm ảnh hoạt động tốt, tần số quét tối đa là 119Hz, tần số quét ổn định là 90Hz.

4. RS232:

- Kịch bản kiểm thử: Lập trình phần mềm trên máy tính gửi lần lượt 180000 byte dữ liệu ngẫu nhiên từ máy tính sang Kit thí nghiệm với baudrate 115200, chu kì gửi là 50ms, lập trình nhận dữ liệu và gửi lại dữ liệu đã nhận thông qua RS232. Phần mềm trên máy tính sẽ kiểm tra số gói dữ liệu phản hồi chính xác. Kiểm tra tính ổn định của khối RS232.
- Kết quả: Dữ liệu nhận chính xác 180000/180000.

5. RS485:

- Kịch bản kiểm thử: Lập trình phần mềm trên máy tính gửi lần lượt 180000 byte dữ liệu ngẫu nhiên từ máy tính sang Kit thí nghiệm với baudrate 115200, chu kì gửi là 50ms, lập trình nhận dữ liệu và gửi lại dữ liệu đã nhận thông qua RS485. Phần mềm trên máy tính sẽ kiểm tra số gói dữ liệu phản hồi chính xác. Kiểm tra tính ổn định của khối RS485.
- Kết quả: Dữ liệu nhận chính xác 180000/180000.

6. CAN:

- Kịch bản kiểm thử: Lập trình phần mềm trên máy tính gửi lần lượt 180000 byte dữ liệu ngẫu nhiên từ máy tính sang Kit thí nghiệm với baudrate 375000, chu kì gửi là 50ms, lập trình nhận dữ liệu và gửi lại dữ liệu đã nhận thông qua CAN. Phần mềm trên máy tính sẽ kiểm tra số gói dữ liệu phản hồi chính xác. Kiểm tra tính ổn định của khối CAN.
- Kết quả: Dữ liệu nhận chính xác 180000/180000.

7. USB:

- Kịch bản kiểm thử: Lập trình phần mềm trên máy tính gửi lần lượt 180000 byte dữ liệu ngẫu nhiên từ máy tính sang Kit thí nghiệm với chu kì gửi là 50ms, lập trình nhận dữ liệu và gửi lại dữ liệu đã nhận thông qua USB. Phần mềm trên máy tính sẽ kiểm tra số gói dữ liệu phản hồi chính xác. Kiểm tra tính ổn định của khối USB.
- Kết quả: Dữ liệu nhận chính xác 180000/180000.

8. Cảm biến điện áp:

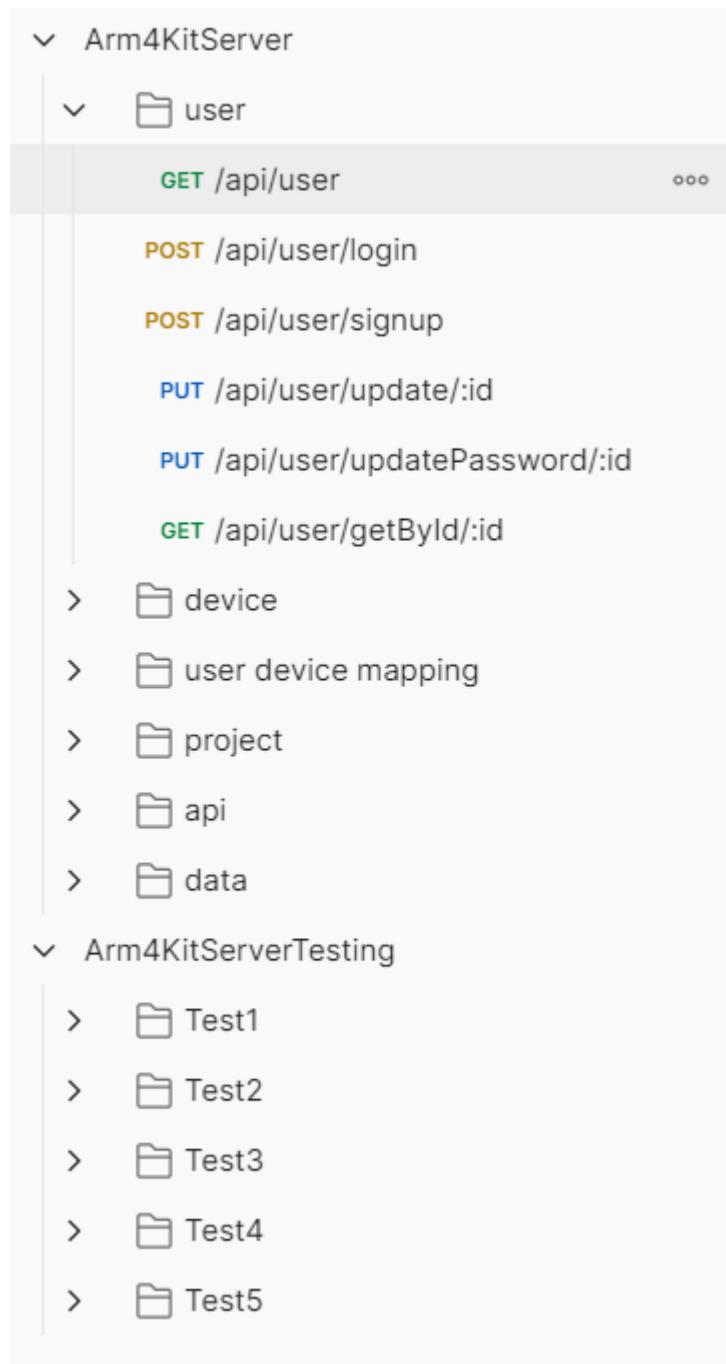
- Kịch bản kiểm thử: So sánh điện áp đo được qua cảm biến và điện áp đo được bởi VOM.
- Kết quả: với nguồn cấp từ adapter 12V, điện áp đo được từ VOM và cảm biến đều trong khoảng từ 11.83 đến 11,98.

4.2 Kết quả kiểm thử kiến thức có trong bộ bài thí nghiệm

1. Kiểm thử tính chính xác của các kiến thức: Để có thể khẳng định tính đúng đắn của tài liệu, nhóm đã dựa trên bộ bài thí nghiệm Vi xử lý, nhờ sử review của Ths Phan Đình Thế Duy - giáo viên trực tiếp hướng dẫn đề tài của chúng tôi và Ths. Vũ Trọng Thiên - giảng viên dạy môn Hệ thống nhúng - khoa Khoa học và Kỹ thuật Máy tính trường Đại Học Bách Khoa - Đại học quốc gia Tp.Hồ Chí Minh.
2. Kiểm thử độ phù hợp với đối tượng người học: Để kiểm thử độ phù hợp của kit thí nghiệm, giáo viên hướng dẫn đã hỗ trợ nhóm đưa 25 kit thí nghiệm vào bộ môn Hệ thống nhúng và Đồ án thiết kế luận lý của trường và 35 kit vào lớp vi điều khiển mà thầy Duy tổ chức, từ đó giúp nhóm có thể kiểm tra được sự phù hợp của kit thí nghiệm với môi trường học tập và có căn cứ để chỉnh sửa.
3. Kiểm thử tính tương tác và thực hành: Để kiểm tra sự tương tác của người dùng, nhóm đã tiến hành làm nhiều phiên bản khác nhau và làm nhiều lần, thay đổi các phần như cồng nạp, nút nhấn, header ra chân,...

4.3 Kết quả kiểm thử server backend

Trong phần này, nhóm đã sử dụng Postman để kiểm thử tính đúng đắn của các API[?].



Hình 4.1: Cấu trúc thư mục postman

4.3.1 Một vài kịch bản kiểm thử

Mô tả kịch bản 1: Kiểm tra tính đúng đắn của các API tương tác với thông tin của một người dùng.

1. Đăng nhập khi chưa có tài khoản.
Kết quả mong đợi: status code = 404.

2. Đăng ký tài khoản mới với các trường hợp lệ.

Kết quả mong đợi: status code = 201.

3. Đăng nhập với email và mật khẩu vừa đăng ký.

Kết quả mong đợi: status code = 200.

4. Cập nhật username mới với tài khoản vừa đăng ký

Kết quả mong đợi: status code = 200, username được cập nhật.

5. Cập nhật mật khẩu mới với tài khoản vừa đăng ký, nhập sai mật khẩu cũ.

Kết quả mong đợi: status code = 400.

6. Cập nhật mật khẩu mới với tài khoản vừa đăng ký, nhập đúng mật khẩu cũ.

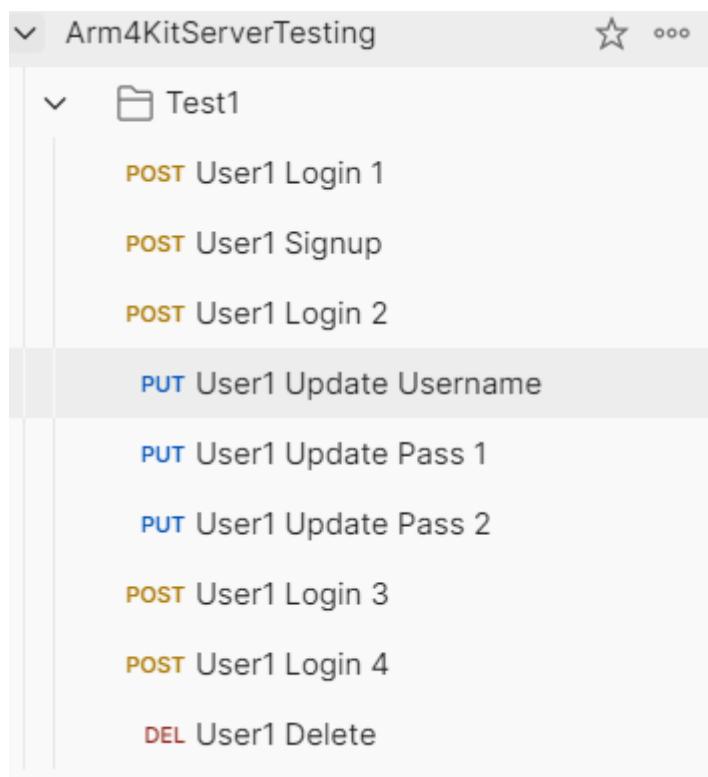
Kết quả mong đợi: status code = 200.

7. Đăng nhập tài khoản với mật khẩu trước khi đổi.

Kết quả mong đợi: status code = 400.

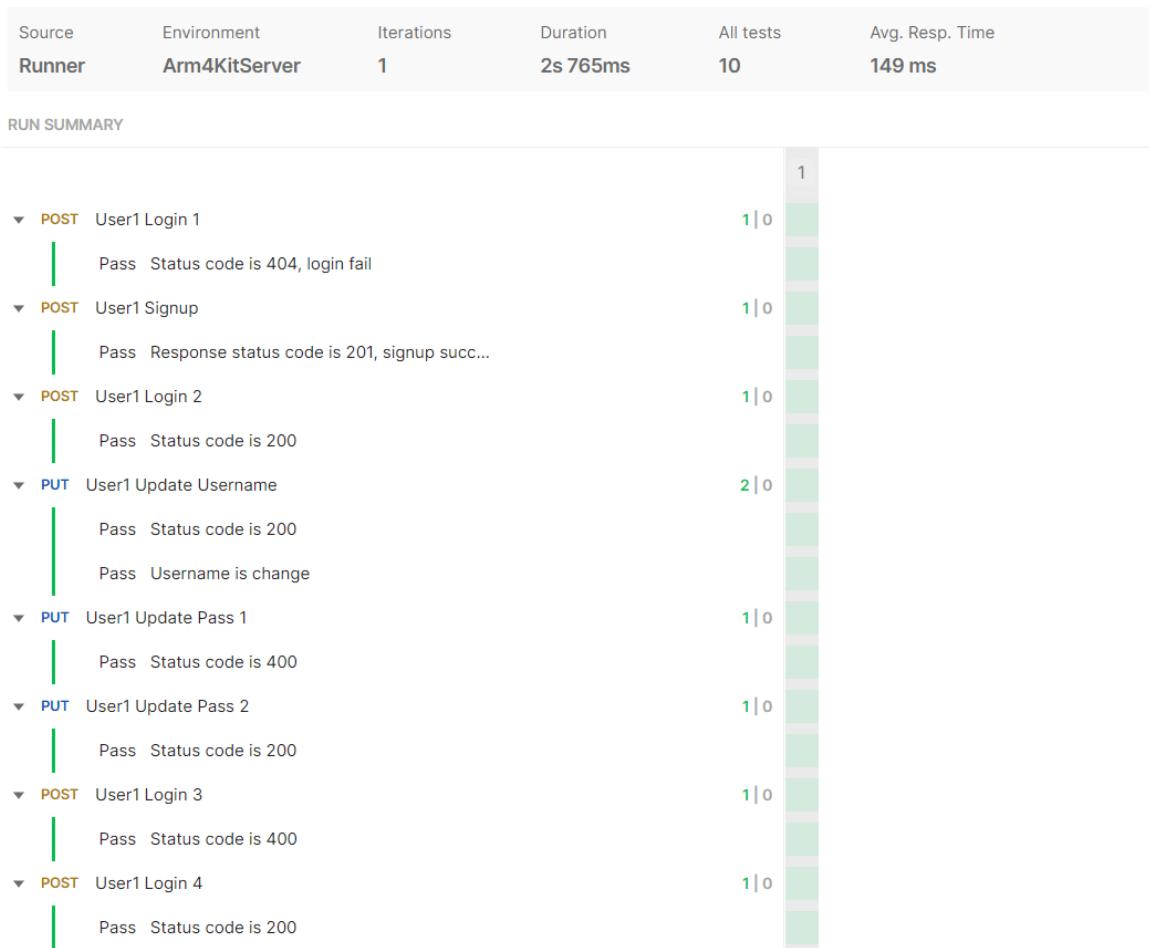
8. Đăng nhập tài khoản với mật khẩu sau khi đổi.

Kết quả mong đợi: status code = 200.



Hình 4.2: Thư mục kiểm thử 1

4.3. Kết quả kiểm thử server backend



Hình 4.3: Kết quả kiểm thử 1

Mô tả kịch bản 2: Kiểm tra tính đúng đắn về các ràng buộc của người dùng.

1. Người dùng 1 đăng ký với đầy đủ các thuộc tính.

Kết quả mong đợi: status code = 201.

2. Lần lượt các trường hợp người dùng 2 gửi yêu cầu đăng ký nhưng thiếu các trường bắt buộc.

Kết quả mong đợi: status code = 400.

3. Người dùng 2 đăng ký với đầy đủ các trường bắt buộc nhưng mật khẩu nhỏ hơn 6 kí tự.

Kết quả mong đợi: status code = 400.

4. Người dùng 2 đăng ký với trường username trùng với người dùng 1.

Kết quả mong đợi: status code = 400.

5. Người dùng 2 đăng ký với trường email trùng với người dùng 1.

Kết quả mong đợi: status code = 400.

4.3. Kết quả kiểm thử server backend

6. Người dùng 2 đăng ký với trường số điện thoại trùng với người dùng 1.

Kết quả mong đợi: status code = 400.

7. Người dùng 2 đăng ký tài khoản với đầy đủ các trường bắt buộc và khác với thông tin của người dùng 1.

Kết quả mong đợi: status code = 201.

8. Người dùng 2 đăng nhập sai mật khẩu (dùng mật khẩu của người dùng 1).

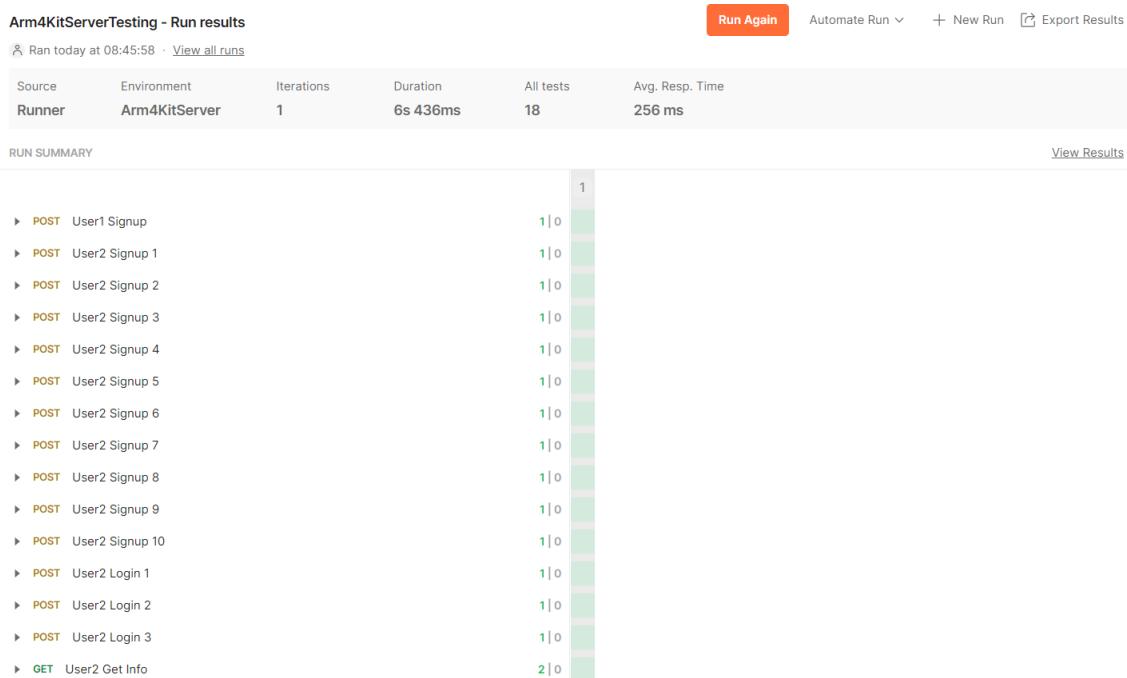
Kết quả mong đợi: status code = 400.

9. Người dùng 2 đăng nhập sai email (dùng email của người dùng 1).

Kết quả mong đợi: status code = 400.

10. Yêu cầu lấy thông tin của người dùng 2.

Kết quả mong đợi: status code = 200, các thông tin lấy được chính xác.



Hình 4.4: Kết quả kiểm thử 2

Mô tả kịch bản 3: Kiểm thử luồng sử dụng để gửi dữ liệu của người dùng

1. Người dùng 1 đăng ký với đầy đủ các thuộc tính thỏa mãn ràng buộc.

Kết quả mong đợi: status code = 201.

2. Người dùng 1 đăng nhập với đúng email và mật khẩu.

Kết quả mong đợi: status code = 201.

3. Đăng ký thiết bị 1 với đầy đủ các thuộc tính thỏa yêu cầu.

Kết quả mong đợi: status code = 201.

4. Đăng ký thiết bị 2, số seri trùng với thiết bị 1.

Kết quả mong đợi: status code = 400.

5. Cập nhật trạng thái của thiết bị 1.

Kết quả mong đợi: status code = 200, thông tin trạng thái được cập nhật đúng.

6. Tạo dự án 1 thuộc về người dùng 1 và thiết bị 1, nhưng chưa liên kết người dùng 1 và thiết bị 1.

Kết quả mong đợi: status code = 400.

7. Tạo mối liên kết giữa người dùng 1 và thiết bị 1.

Kết quả mong đợi: status code = 201.

8. Tạo dự án 1 thuộc về người dùng 1 và thiết bị 1.

Kết quả mong đợi: status code = 201.

9. Tạo dự án 2 thuộc về người dùng 1 và thiết bị 1, tên dự án 2 giống với tên dự án 1.

Kết quả mong đợi: status code = 400.

10. Tạo dự án 2 thuộc về người dùng 1 và thiết bị 1, tên dự án khác với tên các dự án khác của người dùng 1.

Kết quả mong đợi: status code = 201.

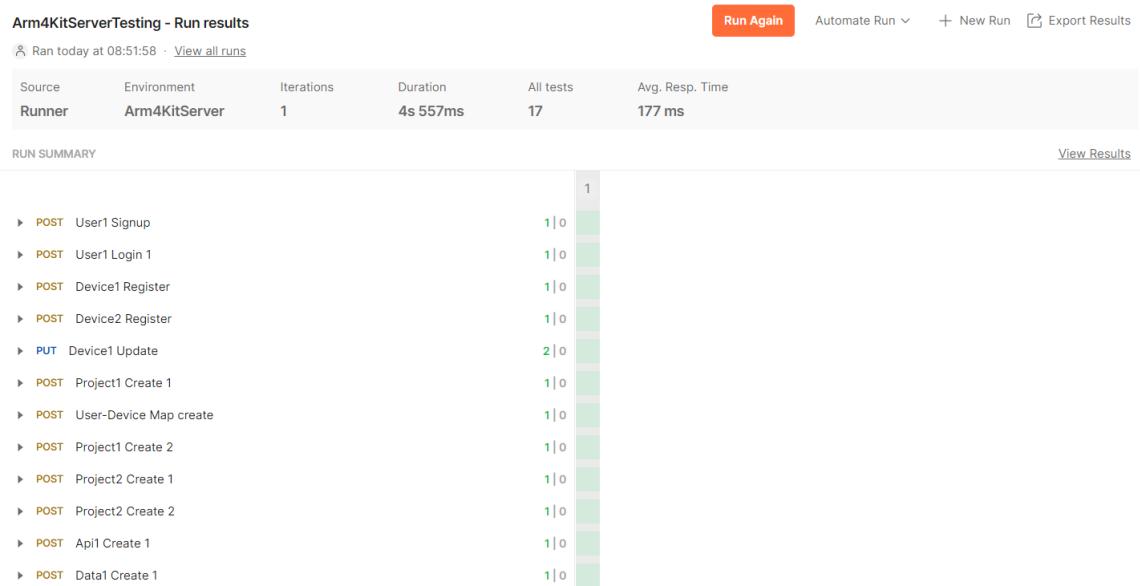
11. Tạo API 1 thuộc về dự án 1.

Kết quả mong đợi: status code = 201.

12. Tạo điểm dữ liệu 1 thuộc về dự án 1 và API 1, dữ liệu được lưu trữ bằng cấu trúc json.

Kết quả mong đợi: status code = 201.

4.3. Kết quả kiểm thử server backend



Hình 4.5: Kết quả kiểm thử 3

4.3.2 Kiểm thử hiệu suất

Điều kiện:

- Mạng ổn định (20Mbps).
- Cấu hình server: RAM 16GB, ổ cứng SSD 256 GB
- Thực hiện 100 requests cho mỗi chức năng để tính thời gian trung bình.

Kết quả:

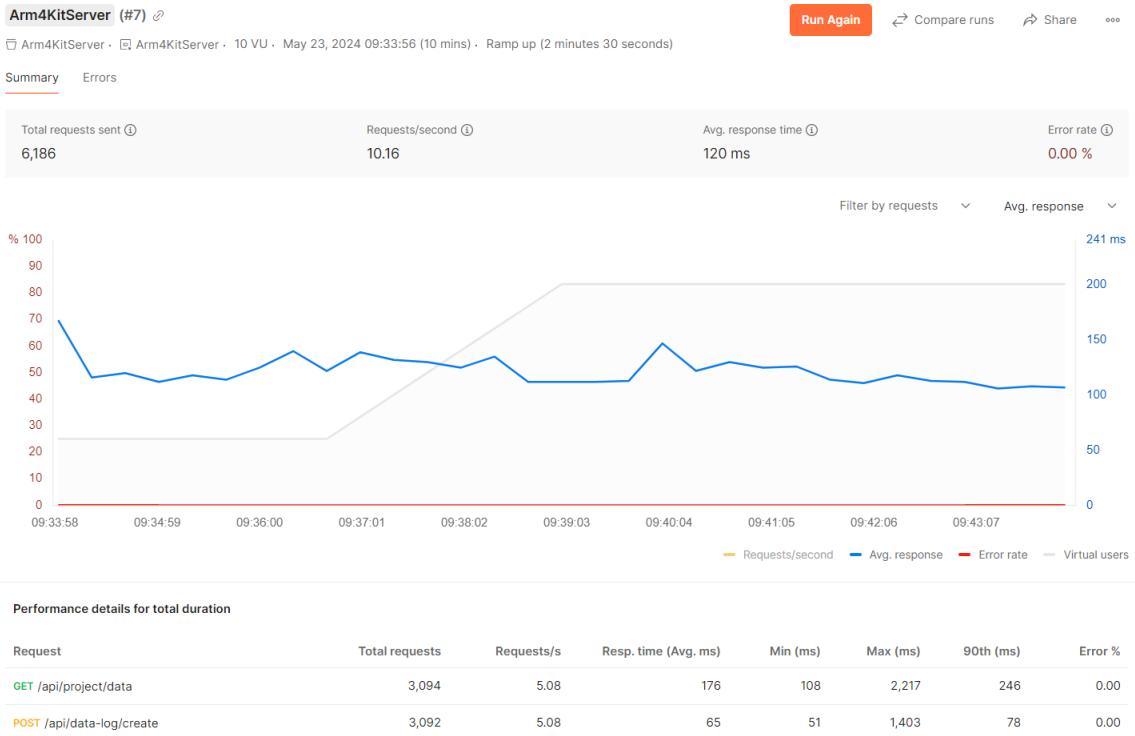
- Thời gian phản hồi trung bình khi tạo điểm dữ liệu mới: 60ms.
- Thời gian phản hồi trung bình khi lấy tất cả dữ liệu của 1 dự án (số lượng 800 trong tổng 7500 điểm dữ liệu): 261ms.

Một số mô phỏng kiểm thử tải trên Postman

- Thời gian mô phỏng: 10 phút.
- Số lượng người dùng mô phỏng 10.
- Các request mô phỏng: thêm điểm dữ liệu, lấy tất cả dữ liệu của 1 dự án.

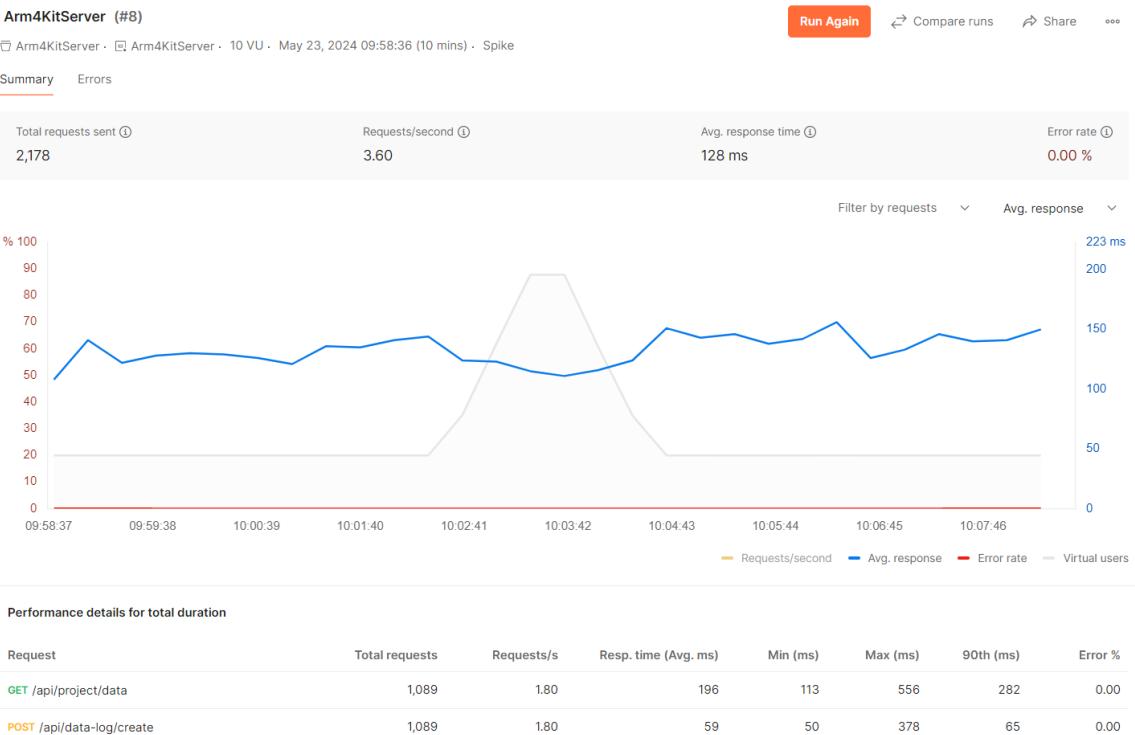
4.3. Kết quả kiểm thử server backend

Kết quả kiểm thử 1



Hình 4.6: Kết quả kiểm thử tải 1

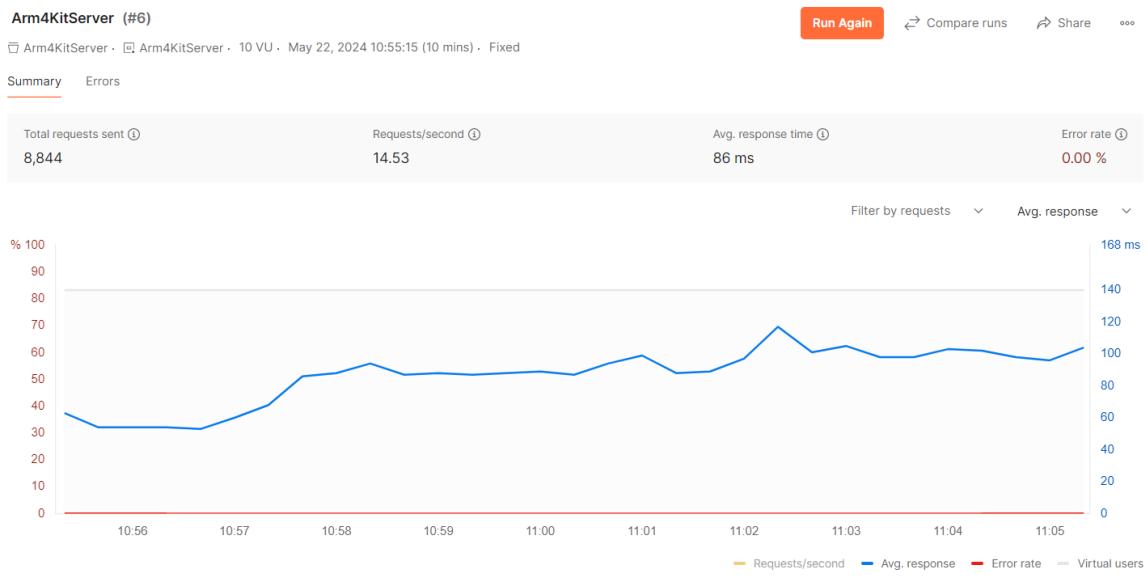
Kết quả kiểm thử 2



Hình 4.7: Kết quả kiểm thử tải 2

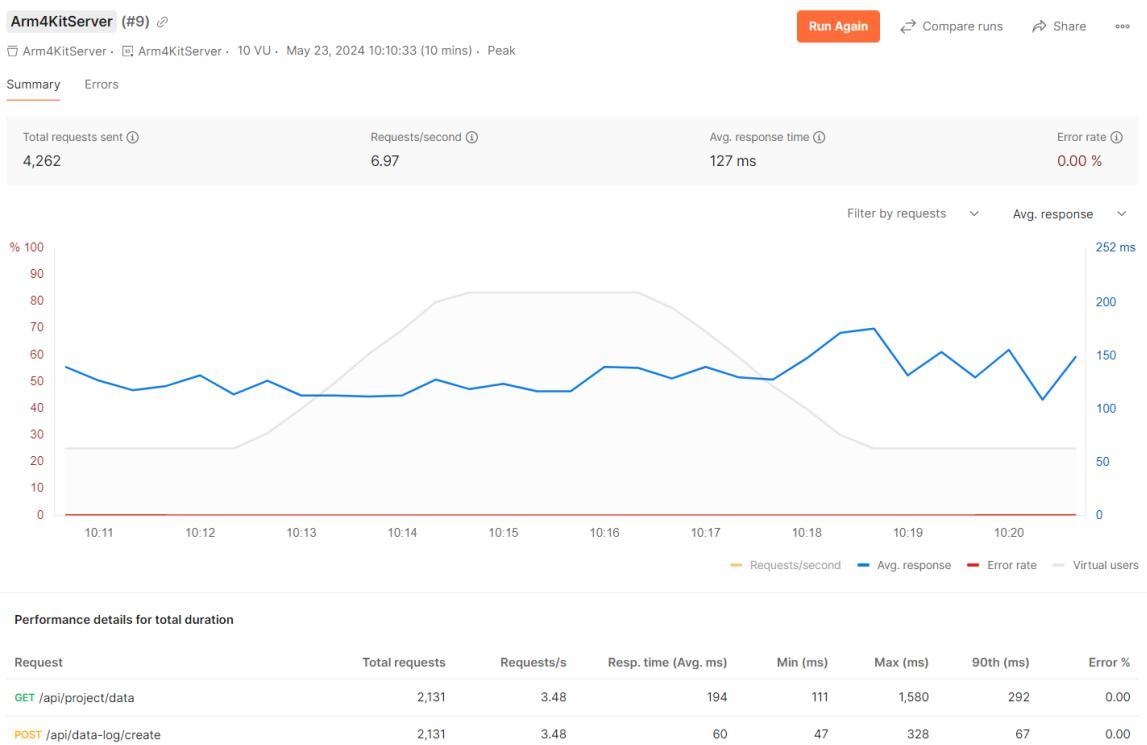
4.3. Kết quả kiểm thử server backend

Kết quả kiểm thử 3



Hình 4.8: Kết quả kiểm thử tải 3

Kết quả kiểm thử 4



Hình 4.9: Kết quả kiểm thử tải 4

4.4 Triển khai thực tế

Trong giai đoạn đề tài, dưới sự đồng ý của Khoa và của giáo viên hướng dẫn, nhóm đã triển khai các bộ Kit thí nghiệm và bài thí nghiệm cho một số môn học của Khoa để khảo sát và thu thập phản hồi về tính hiệu quả và hạn chế của sản phẩm. Cụ thể:

- 25 bộ Kit triển khai cho môn học Thí nghiệm hệ thống nhúng - Thầy Phạm Hoàng Anh.
- 35 bộ Kit triển khai cho khóa học vi điều khiển - Thầy Phan Đình Thế Duy.

Ngoài ra, nhóm cũng áp dụng Kit thí nghiệm để khảo sát, kiểm thử trong dự án xây dựng mạch điều khiển và hiển thị màn hình thang máy, giao tiếp với CAN trên mạch của khóa học Bosch tổ chức tại trường, khảo sát dự án smart home sử dụng KNX.

Trong quá trình triển khai thực tế, nhóm tóm tắt các phản hồi và đánh giá sản phẩm như sau:

Mục phản hồi	Phản hồi
Màn hình LCD	Hoạt động tốt, cảm ứng nhạy
Ma trận phím	Nhỏ, khó tương tác
Header ra chân	Không có chân ra nguồn 3V3, khó kết hợp các module
USB Type-C	Bị lỏng sau nhiều lần sử dụng
Các bài thí nghiệm	Cung cấp đầy đủ các kiến thức về vi điều khiển Giúp khảo sát thí nghiệm được các chức năng trên Kit Số lượng bài tập chưa đủ nhiều để luyện tập

Bảng 4.1: Đánh giá, phản hồi sản phẩm

CHƯƠNG 5

KẾT LUẬN - ĐỀ NGHỊ

5.1 Kết luận về kết quả đạt được

Trong quá trình nghiên cứu và thực hiện đề tài, với những nỗ lực và cỗ gắng của các thành viên trong nhóm, cùng với sự hướng dẫn, góp ý tận tình của ThS. Phan Đình Thế Duy và ThS Vũ Trọng Thiên, đề tài đã hoàn thành kịp tiến độ quy định theo yêu cầu đặt ra là PHÁT TRIỂN KIT THÍ NGHIỆM HỆ THỐNG NHÚNG DỰA TRÊN LÕI KIẾN TRÚC ARM.

Trong gai suốt quá trình thực hiện đề tài, nhóm đã hoàn thành được các yêu cầu sau:

- Thiết kế và hiện thực phần cứng KIT thí nghiệm với các khối chức năng sau:
 - Vi điều khiển chính STM32F407 và khôi nạp ST-Link.
 - Khối nguồn 12-24V.
 - Khối hiển thị LED đồng hồ và khôi ma trận phím.
 - Khối hiển thị LCD graphic hỗ trợ cảm ứng.
 - Khối giao tiếp thông qua RS232, RS485, USB, INPUT-OUTPUT 12-24V.
 - Khối cảm biến gồm biến trở, cảm biến ánh sáng, cảm biến nhiệt độ, cảm biến dòng điện và điện áp.
 - Khối bộ nhớ gồm DS3231, SRAM, Eeprom, Flash, thẻ SD.
 - Khối kết nối Internet thông qua Wifi, Ethernet sử dụng ESP8266.
 - Khối module sim 4G/LTE.
- Thiết kế bộ bài thí nghiệm đầy đủ kiến thức về vi xử lí và khảo sát chức năng bo mạch. Các bài thí nghiệm được thiết kế theo các chủ đề:

- GPIO.
 - Timer interrupt.
 - Đọc ma trận phím và điều khiển LCD.
 - Giao tiếp I2C và Real Time Clock DS3231.
 - Giao tiếp UART và RS232.
 - Đọc cảm biến qua ADC và điều khiển PWM.
 - Tương tác với màn hình cảm ứng.
 - Kết nối Wifi thông qua ESP8266.
 - RTOS và các kiến thức liên quan như queue, mutex, semaphore,...
- Thiết kế và kiểm thử server backend hỗ trợ giao thức HTTP: Trong phần này, chúng tôi thiết kế và kiểm thử server backend hỗ trợ giao thức HTTP. Mục tiêu chính của việc này là để tạo ra một hệ thống hỗ trợ hiệu quả cho việc học. Server này giúp người học có thể dễ dàng thực hiện các project, các bài thực hành liên quan đến IoT. Nó không chỉ giúp họ hiểu rõ hơn về cách hoạt động của IoT mà còn cung cấp cho họ các kỹ năng thực tế cần thiết để họ có thể áp dụng vào các dự án thực tế.

Với Kit thí nghiệm, bộ bài thí nghiệm và server backend đã xây dựng, người dùng có thể bước đầu sử dụng Kit thí nghiệm để học tập nghiên cứu về lập trình vi điều khiển cơ bản. Ngoài ra Kit thí nghiệm cũng là cầu nối để người dùng có thể ứng dụng vào việc mô phỏng, kiểm thử một số hệ thống từ điều khiển, giao tiếp đơn giản cho đến các hệ thống IoT phức tạp hơn.

Tuy chưa có nhiều kinh nghiệm trong thực tế công nghiệp và cũng gặp nhiều khó khăn trong quá trình thiết kế nhưng nhóm chúng tôi cũng đã đạt được những kết quả sau:

- Thiết kế và thi công hoàn thiện bộ bài thí nghiệm khảo sát board mạch và giảng dạy về vi xử lí, tổng hợp các chức năng phù hợp với người học tập, nghiên cứu lên đã board mạch, không chỉ đơn thuần là các kiến thức học, nhóm còn hướng tới người dùng có thể ứng dụng Kit trong tương lai về cả mặt công nghiệp.
- Qua đợt tài lần này, nhóm cũng đã có thêm rất nhiều kinh nghiệm trong quá trình sử dụng phần mềm thiết kế mạch Altium Designer và rất nhiều bộ thư viện liên quan. Đây là một kỹ năng có thể tạo cho các thành viên trong nhóm có thêm nhiều cơ hội trong tương lai.

- Qua quá trình xây dựng những bài code đơn giản và hiện thực trên mạch để kiểm thử các tính năng của mạch, nhờ quá trình này nhóm đã có thêm nhiều kinh nghiệm trong test mạch, kiểm thử mạch và sửa chữa đưa ra phiên bản đầu tiên được chạy ổn định như hiện tại.
- Hoàn thiện các bộ bài tài liệu hướng dẫn để người dùng có thể bước đầu sử dụng mạch. Nhờ quá trình này nhóm đã có cơ hội chiêm nghiệm lại những kiến thức đã học, tìm ra được sự liên kết giữa chúng và hiểu thêm được về quá trình những kiến thức này ra đời, hiểu thêm những kiến thức đã được học.
- Tuy chưa từng có kinh nghiệm trong việc làm server hay tiếp xúc nhiều với việc xây dựng server nhưng nhóm cũng đã cố gắng để tạo ra một server backend hỗ trợ http để người dùng có thể thực hiện các dự án IoT.

5.2 Những hạn chế còn tồn tại

Trong suốt giai đoạn đề tài, hệ thống của nhóm đã đạt được một số thành quả và đóng góp nhất định, song vẫn còn tồn tại một số hạn chế cần khắc phục trong tương lai như:

- Chưa hỗ trợ các tính năng truyền thông thông dụng như Bluetooth.
- Chưa có thiết kế hộp giúp đóng gói Kit thí nghiệm một cách bắt mắt hơn.
- Các bài thí nghiệm và công cụ lập trình phù hợp với người có định hướng học chuyên sâu về hệ thống nhúng, nhưng vẫn còn gây khó khăn với những người mới bắt đầu tìm hiểu về lập trình vi điều khiển.
- Hệ thống server chưa hỗ trợ UI và các tính năng trên UI như dashboard.
- Hệ thống server chưa hỗ trợ MQTT.
- Hệ thống server chưa được hosting để áp dụng và kiểm nghiệm thực tế.

5.3 Những đề nghị về hướng phát triển của hệ thống

Trong tương lai, nhóm mong muốn phát triển hệ thống để hướng tới mục đích xây dựng một hệ sinh thái phục vụ cho nhu cầu học lập trình nhúng và IoT gồm:

5.3. Những đề nghị về hướng phát triển của hệ thống

- Kit thí nghiệm: Tiến hành khảo sát, chỉnh sửa một số chức năng phần cứng để phù hợp hơn với nhu cầu người sử dụng, thêm các thành phần hoặc giảm bớt thành phần để phù hợp hơn với nhiều tệp khách hàng.
- Bộ bài thí nghiệm: Xây dựng đầy đủ bộ, hoàn thiện bài thí nghiệm về kĩ thuật truyền số liệu, hệ thống nhúng với hệ điều hành thời gian thực, IoT. Đồng thời, tiếp nhận góp ý, chỉnh sửa hoàn thiện hơn cho tài liệu.
- Tài liệu hướng dẫn: Thiết lập thêm nhiều hình thức hướng dẫn hơn để gây thêm hứng thú cho người dùng, chẳng hạn như: trang web hướng dẫn, xây dựng các video hướng dẫn.
- Máy chủ: Để hướng tới việc người dùng có trải nghiệm, ứng dụng Kit thí nghiệm để học tập về Iot, nhóm sẽ phải xây dựng một máy chủ Iot hoàn chỉnh hơn, có giao diện đẹp hơn nữa, giúp người học có thể dễ dàng tiếp cận và thực hành với khái niệm trong Iot.

CHƯƠNG 6

TÀI LIỆU THAM KHẢO

- [1] N. A. Văn, “Nhu cầu nhân lực ngành iot ngày càng tăng mạnh.” <https://vneconomy.vn/nhu-cau-nhan-luc-nganh-iot-ngay-cang-tang-manh.htm>, 2022.
- [2] B. Lutkevich, “What is an embedded system?.” <https://www.techtarget.com/iotagenda/definition/embeded-system>, 2020.
- [3] IBM, “What is internet of Things?.”<https://www.ibm.com/topics/internet-of-things>, 2023.
- [4] D. Chính, “Tiềm năng thị trường iot việt nam: Ước tạo doanh thu hàng tỷ usd mỗi năm, ngành ô tô và tiêu dùng chiếm tỷ trọng lớn,” vietnambiz, 2023.
- [5] WatElectronics.com, “Arm architecture.” <https://www.watelectronics.com/arm-processor-architecture-working/>, 2019.
- [6] S. S. Michael, “The arm architecture explained.” <https://www.allaboutcircuits.com/technical-articles/arm-architecture-explained/>, 2019.
- [7] I. technology, “Uart communication – how it works?!” <https://insemitech.com/?s=uart>, 2022.
- [8] Soldered, “What is the uart communication protocol.” <https://www.linkedin.com/pulse/uart-communication-how-works-insemi-technology-services-private>, 2023.
- [9] R. Sheldon, “What is a serial peripheral interface(spi)?.”<https://www.techtarget.com/whatis/definition/serial-peripheral-interface-SPI>, 2023.

-
- [10] S. Campbell, "Basics of the i2c communication protocol." <https://www.circuit-basics.com/basics-of-the-i2c-communication-protocol/>, 2023.
- [11] Shawn, "What is rs232 and how to get started." <https://www.seeedstudio.com/blog/2019/12/12/what-is-rs232-and-how-to-get-started/>, 2019.
- [12] Lorric, "What is rs485: 5 minutes to understand it." <https://www.lorric.com/en/WhyLORRIC/Flowmeter/what-is-rs485>, 2023.
- [13] S. Lelii, "Secure digital card (sd card)." <https://www.techtarget.com/search-storage/definition/Secure-Digital-card>, 2017.
- [14] S. Corrigan, "Introduction to the controller area network (can)." https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1702745807585ref_url=https%253A%252F%252Fwww.google.com%252F, 2016.
- [15] ABCSolutions, "Tài liệu hướng dẫn nhanh về kit thí nghiệm bkit arm4." <https://abcsolutions.com.vn/index.php/kit-thi-nghiem-bkit-arm4/>, 2023.
- [16] K. Osmereshone, "How to automate api testing with postman." <https://www.smashingmagazine.com/2020/09/automate-api-testing-postman/>, 2020
- [17] Dr.Le Trong Nhan, "Microcontroller Labs"
- [18] Dr.Le Trong Nhan, "Build your own IoT Gateway with Python"