TRƯỜNG ĐẠI HỌC BÌNH DƯƠNG

KHOA CÔNG NGHỆ THÔNG TIN, ROBOT

VÀ TRÍ TUỆ NHÂN TẠO

---oOo---

TIỂU LUẬN MÔN
LẬP TRÌNH HỆ THỐNG NHÚNG

Tên tiểu luận:

DIY Robot sử dụng FreeRTOS
Xe Robot điều khiển

Người hướng dẫn: **Th.s Lê Duy Hùng**

Sinh viên thực hiện:

Nguyễn Văn Khánh-22050079

Phạm Thanh Phong-22050001

TP.HCM, tháng 8 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ 4.0 đang phát triển mạnh mẽ, robot và hệ thống tự động hóa đang trở thành xu hướng chủ đạo trong nhiều lĩnh vực từ sản xuất công nghiệp đến ứng dụng dân sinh. Việc nghiên cứu và phát triển các hệ thống robot không chỉ có ý nghĩa học thuật mà còn mang lại giá trị thực tiễn cao.

Lý do chọn đề tài

Đề tài "DIY Robot 2 bánh sử dụng FreeRTOS" được chọn với mục đích tìm hiểu và ứng dụng hệ điều hành thời gian thực FreeRTOS trong việc điều khiển robot di động. Việc sử dụng FreeRTOS cho phép hệ thống có khả năng xử lý đa tác vụ, đảm bảo tính ổn định và độ tin cậy cao trong quá trình hoạt động. Đây là một chủ đề quan trọng và thiết thực trong lĩnh vực hệ thống nhúng hiện đại.

Ý nghĩa thực tiễn

Dự án này có ý nghĩa thực tiễn cao khi cung cấp một giải pháp robot di động có thể điều khiển từ xa thông qua WiFi. Sản phẩm có thể được ứng dụng trong nhiều lĩnh vực như giám sát, vận chuyển tự động trong không gian hẹp, hoặc làm nền tảng cho các dự án robot phức tạp hơn. Việc sử dụng các linh kiện phổ biến và chi phí thấp như ESP32, L298N và motor DC giúp sản phẩm có tính khả thi cao và dễ dàng nhân rộng.

Phạm vi nghiên cứu

Dự án tập trung vào việc thiết kế và triển khai một hệ thống robot 2 bánh với các tính năng chính:

- Sử dụng ESP32 làm bộ xử lý chính với FreeRTOS
- Điều khiển từ xa qua giao diện web thông qua WiFi
- Tích hợp các tính năng an toàn như auto-stop và lock mode
- Quản lý đa tác vụ hiệu quả với kiến trúc dual-core

.

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy **Lê Duy Hùng** đã tận tình hướng dẫn và hỗ trợ em trong suốt quá trình thực hiện đề tài. Em cũng xin cảm ơn gia đình, bạn bè đã động viên và tạo điều kiện cho em hoàn thành báo cáo này một cách tốt nhất.

Dù đã nỗ lực hoàn thành tốt nhất trong khả năng của mình, bài tiểu luận chắc chắn vẫn còn nhiều thiếu sót. Em rất mong nhận được những góp ý từ thầy để hoàn thiện hơn trong các nghiên cứu sau này.

MỤC LỤC

MỤC LỤC HÌNH ẢNH VÀ BẢNG BIỂU

CHƯƠNG 1: TỔNG QUAN DỰ ÁN

1.1. Giới thiệu

1.1.1. Xu hướng robot tự động hóa

Trong những năm gần đây, công nghệ robot đã có những bước tiến vượt bậc và trở thành một phần không thể thiếu trong cuộc sống hiện đại. Robot không còn chỉ xuất hiện trong các nhà máy sản xuất mà đã thâm nhập vào nhiều lĩnh vực khác nhau như y tế, giáo dục, dịch vụ và gia đình. Điều này tạo ra nhu cầu ngày càng cao về các hệ thống robot thông minh, linh hoạt và có khả năng tương tác tốt với môi trường xung quanh.

Sự phát triển của các vi xử lý hiệu năng cao như ESP32, cùng với sự tiến bộ của công nghệ truyền thông không dây, đã mở ra nhiều cơ hội mới trong việc phát triển robot di động. Các robot hiện đại không chỉ cần có khả năng di chuyển mà còn phải có khả năng xử lý đồng thời nhiều tác vụ, từ việc điều khiển chuyển động đến giao tiếp với người dùng và xử lý dữ liệu sensor.

1.1.2. Ưu điểm của hệ điều hành thời gian thực

Hệ điều hành thời gian thực (Real-Time Operating System - RTOS) đóng vai trò quan trọng trong việc đảm bảo hiệu năng và độ tin cậy của hệ thống robot. FreeRTOS, với khả năng quản lý đa tác vụ ưu tiên và cơ chế preemptive scheduling, cho phép hệ thống xử lý đồng thời nhiều công việc một cách hiệu quả. Điều này đặc biệt quan trọng trong ứng dụng robot, nơi cần phải xử lý đồng thời các tác vụ như điều khiển motor, giao tiếp WiFi, và cập nhật trạng thái hệ thống.

Việc sử dụng RTOS cũng giúp tối ưu hóa việc sử dụng tài nguyên phần cứng, đặc biệt là với các vi xử lý dual-core như ESP32. Khả năng phân chia tác vụ lên các core khác nhau giúp tăng hiệu năng tổng thể của hệ thống và đảm bảo tính ổn định trong hoạt động.

1.2. Mục tiêu dự án

1.2.1. Mục tiêu chính

Mục tiêu chính của dự án là xây dựng thành công một robot di động 2 bánh có khả năng điều khiển từ xa thông qua giao diện web. Robot cần phải hoạt động ổn định,

tin cậy và có khả năng thực hiện các chuyển động cơ bản như tiến, lùi, xoay trái, xoay phải với tốc độ có thể điều chỉnh. Hệ thống phải được xây dựng trên nền tảng FreeRTOS để tận dụng tối đa khả năng xử lý đa tác vụ và đảm bảo tính thời gian thực.

1.2.2. Mục tiêu cụ thể

Sử dụng FreeRTOS để quản lý đa tác vụ: Hệ thống được thiết kế với kiến trúc đa tác vụ, bao gồm tác vụ điều khiển motor, tác vụ quản lý WiFi server và tác vụ hiển thị trạng thái LED. Mỗi tác vụ có mức độ ưu tiên khác nhau và được phân bổ trên các core khác nhau để tối ưu hiệu năng.

Điều khiển qua WiFi interface: Robot tạo một WiFi Access Point cho phép người dùng kết nối trực tiếp mà không cần router trung gian. Giao diện điều khiển được phát triển dưới dạng web responsive, có thể truy cập từ bất kỳ thiết bị nào có web browser như smartphone, tablet hay laptop.

Tích hợp các tính năng an toàn: Hệ thống được trang bị nhiều cơ chế an toàn bao gồm auto-stop timeout (tự động dừng khi không nhận được lệnh trong thời gian định trước), emergency stop (dừng khẩn cấp từ giao diện web), và lock mode (chế độ khóa cho phép điều khiển từng bánh xe độc lập). Các tính năng này đảm bảo robot hoạt động an toàn và có thể được kiểm soát hiệu quả.

1.3. Yêu cầu hệ thống

1.3.1. Yêu cầu phần cứng

ESP32 Microcontroller: Được chọn làm bộ xử lý chính với kiến trúc dual-core Xtensa LX6, tích hợp WiFi và Bluetooth, có đủ GPIO pins để điều khiển các thiết bị ngoại vi. ESP32 cung cấp đủ sức mạnh tính toán để chạy FreeRTOS và xử lý đồng thời nhiều tác vụ phức tạp.

Driver Motor L298N: Module điều khiển motor được chọn có khả năng điều khiển 2 motor DC độc lập với dòng điện lên đến 2A mỗi kênh. L298N sử dụng công nghệ H-bridge cho phép điều khiển hướng quay và tốc độ motor thông qua tín hiệu PWM.

Motor DC và hệ thống cơ khí: Hai motor DC 12V được sử dụng với bánh xe phù hợp, gắn trên khung robot có thiết kế cân bằng. Hệ thống nguồn sử dụng pin 2S Li-ion (7.4V) với mạch buck converter để cung cấp điện áp phù hợp cho từng module.

1.3.2. Yêu cầu phần mềm

FreeRTOS Kernel: Sử dụng phiên bản FreeRTOS được tích hợp trong ESP-IDF hoặc Arduino IDE, cung cấp đầy đủ các tính năng như task scheduling, queue management, và semaphore synchronization.

Web Server Framework: Triển khai HTTP server sử dụng thư viện WiFi của ESP32, có khả năng xử lý multiple connections và serve static content. Server cần hỗ trợ các RESTful API endpoints để nhận lệnh điều khiển từ client.

Giao diện người dùng: Phát triển web interface responsive với HTML5, CSS3 và JavaScript, tương thích với các thiết bị di động. Giao diện cung cấp các nút điều khiển trực quan và hiển thị trạng thái real-time của robot.

1.3.3. Yêu cầu tính năng

Điều khiển từ xa: Robot có khả năng nhận và thực hiện các lệnh điều khiển cơ bản như di chuyển tiến/lùi, xoay trái/phải, với tốc độ có thể điều chỉnh từ 1-255 (giá trị PWM). Hệ thống phải có độ trễ thấp và đáp ứng nhanh chóng các lệnh từ người dùng.

Auto-stop safety: Cơ chế tự động dừng robot khi không nhận được lệnh điều khiển trong khoảng thời gian định trước (400ms). Tính năng này đảm bảo robot không hoạt động liên tục khi mất kết nối hoặc gặp sự cố.

Lock mode: Chế độ đặc biệt cho phép người dùng điều khiển từng bánh xe độc lập, hữu ích cho việc kiểm tra và debug hệ thống. Trong chế độ này, trạng thái của mỗi bánh xe được lưu trữ và có thể được thay đổi riêng lẻ thông qua các lệnh điều khiển.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. FreeRTOS

2.1.1. Khái niệm

FreeRTOS (Free Real-Time Operating System) là một hệ điều hành thời gian thực mã nguồn mở được thiết kế đặc biệt cho các hệ thống nhúng và microcontroller. Được phát triển bởi Real Time Engineers Ltd, FreeRTOS cung cấp kernel nhỏ gọn nhưng mạnh mẽ, cho phép các nhà phát triển tạo ra các ứng dụng đa tác vụ phức tạp trên các vi xử lý có tài nguyên hạn chế.

FreeRTOS được thiết kế theo triết lý "microkernel", chỉ cung cấp những tính năng cốt lõi cần thiết nhất như task scheduling, inter-task communication và synchronization. Điều này giúp giảm thiểu footprint memory và tăng hiệu năng của hệ thống. Kernel có kích thước chỉ khoảng 6-12KB tùy theo cấu hình, phù hợp với các microcontroller có bộ nhớ hạn chế.

2.1.2. Đặc điểm

Preemptive Multitasking: FreeRTOS sử dụng cơ chế preemptive scheduling, cho phép các tác vụ có mức độ ưu tiên cao có thể ngắt và chiếm quyền xử lý từ các tác vụ có mức độ ưu tiên thấp hơn. Điều này đảm bảo các tác vụ quan trọng luôn được xử lý kịp thời, đáp ứng yêu cầu thời gian thực của hệ thống.

Priority-based Scheduling: Mỗi tác vụ trong FreeRTOS được gán một mức độ ưu tiên từ 0 (thấp nhất) đến configMAX_PRIORITIES-1 (cao nhất). Scheduler luôn chọn tác vụ có mức độ ưu tiên cao nhất trong trạng thái Ready để thực thi. Nếu có nhiều tác vụ cùng mức độ ưu tiên, chúng sẽ được xử lý theo cơ chế round-robin.

Tick-based Time Management: FreeRTOS sử dụng một timer gọi là system tick để quản lý thời gian. Mỗi tick interrupt, scheduler sẽ kiểm tra trạng thái các tác vụ và quyết định có cần context switch hay không. Frequency của tick có thể được cấu hình, thường là 100Hz-1000Hz tùy theo yêu cầu ứng dụng.

2.1.3. Ưu điểm

Độ tin cậy cao: FreeRTOS đã được kiểm chứng qua hàng triệu thiết bị triển khai trong thực tế, từ các hệ thống đơn giản đến các ứng dụng mission-critical. Kernel

được thiết kế với cơ chế error handling mạnh mẽ và khả năng phục hồi tốt khi gặp lỗi.

Quản lý tài nguyên hiệu quả: FreeRTOS cung cấp các cơ chế đồng bộ hóa như semaphore, mutex, và queue để quản lý việc truy cập tài nguyên shared giữa các tác vụ. Các cơ chế này được tối ưu hóa để giảm thiểu overhead và đảm bảo tính deterministic của hệ thống.

Scalability: FreeRTOS có khả năng mở rộng tốt, từ các ứng dụng đơn giản với vài tác vụ đến các hệ thống phức tạp với hàng chục tác vụ chạy đồng thời. Việc cấu hình kernel có thể được tùy chỉnh để phù hợp với từng ứng dụng cụ thể.

2.2. ESP32 Microcontroller

2.2.1. Kiến trúc

ESP32 được xây dựng trên kiến trúc dual-core Xtensa LX6 32-bit, với mỗi core có thể hoạt động ở tần số lên đến 240MHz. Kiến trúc dual-core cho phép phân chia tải công việc hiệu quả: một core có thể xử lý các tác vụ ưu tiên cao như điều khiển hardware, trong khi core còn lại xử lý các tác vụ giao tiếp như WiFi và Bluetooth.

Hệ thống memory của ESP32 bao gồm 520KB SRAM được chia thành nhiều segments khác nhau để tối ưu hiệu năng. Ngoài ra, ESP32 hỗ trợ external flash memory lên đến 16MB để lưu trữ code và data. Kiến trúc memory mapping cho phép CPU truy cập trực tiếp external flash như internal memory thông qua cache system.

2.2.2. Tính năng

WiFi và Bluetooth: ESP32 tích hợp sẵn WiFi 802.11 b/g/n và Bluetooth Classic/BLE trong cùng một chip. WiFi hỗ trợ cả Station mode và Access Point mode, với khả năng bảo mật WPA/WPA2/WPS. Bluetooth hỗ trợ cả Classic (BR/EDR) và Low Energy (BLE), cho phép kết nối với nhiều loại thiết bị khác nhau.

Analog-to-Digital Converter (ADC): ESP32 có 2 ADC 12-bit với tổng cộng 18 channels, cho phép đọc các tín hiệu analog từ sensors. ADC hỗ trợ nhiều chế độ hoạt động khác nhau và có độ chính xác cao, phù hợp cho các ứng dụng đo lường.

Pulse Width Modulation (PWM): ESP32 cung cấp 16 channels PWM độc lập với độ phân giải lên đến 16-bit. PWM được sử dụng để điều khiển tốc độ motor, độ sáng LED, và các ứng dụng analog output khác. Các channel PWM có thể được cấu hình với frequency và duty cycle khác nhau.

2.2.3. Ứng dụng

Internet of Things (IoT): ESP32 là một trong những lựa chọn phổ biến nhất cho các dự án IoT nhờ khả năng kết nối không dây tích hợp sẵn và giá thành thấp. Với WiFi và Bluetooth, ESP32 có thể dễ dàng kết nối với cloud services và mobile applications.

Robot điều khiển: Sự kết hợp giữa dual-core processor, GPIO pins đa dạng, và khả năng wireless communication làm ESP32 trở thành lựa chọn lý tưởng cho các dự án robot. ESP32 có đủ sức mạnh để xử lý đồng thời việc điều khiển actuators và giao tiếp với remote controller.

2.3. Driver Motor L298N

2.3.1. Chức năng

L298N là một trình điều khiển mạch tích hợp được thiết kế để điều khiển motor DC và stepper motor. IC này có khả năng điều khiển đồng thời 2 motor DC độc lập hoặc 1 stepper motor 4-wire với dòng điện lên đến 2A mỗi channel. L298N được đóng gói trong module sẵn có với các mạch bảo vệ và connector thuận tiện cho việc sử dụng.

Module L298N thường được tích hợp thêm bộ điều chỉnh điện áp 5V để cung cấp điện cho logic control, heat sink để tản nhiệt, và các LED để hiển thị trạng thái hoạt động. Điều này làm cho module trở nên thân thiện với người dùng và dễ sử dụng trong các dự án DIY.

2.3.2. Nguyên lý

H-bridge Configuration: L298N sử dụng cấu hình H-bridge để điều khiển hướng quay của motor. Mỗi H-bridge bao gồm 4 transistor được sắp xếp tạo thành hình chữ H, với motor DC nằm ở giữa. Bằng cách bật/tắt các transistor theo các combination khác nhau, có thể điều khiển motor quay thuận, quay ngược, hoặc dừng.

PWM Speed Control: Tốc độ motor được điều khiển thông qua tín hiệu PWM (Pulse Width Modulation) đưa vào các pin Enable (ENA, ENB). Duty cycle của tín hiệu PWM quyết định tốc độ trung bình của motor: 0% tương ứng với dừng, 100% tương ứng với tốc độ tối đa. Phương pháp này cho phép điều khiển tốc độ một cách trơn tru và chính xác.

Cảm biến dòng điện: L298N có khả năng cảm biến dòng điện thông qua các điện trở luồng tích hợp. Tính năng này cho phép giám sát dòng điện tiêu thụ của motor và thực hiện các cơ chế bảo vệ như bảo vệ quá dòng.

2.3.3. Thông số

L298N có thông số hoạt động như sau: điện áp nguồn từ 5V đến 35V cho phần power, dòng điện liên tục 2A mỗi channel (peak 3A trong thời gian ngắn), điện áp logic 5V cho phần control. Module có khả năng dissipate power lên đến 25W với tản nhiệt thích hợp. Frequency PWM khuyến nghị là 1-40KHz để đảm bảo hiệu quả và giảm nhiễu.

2.4. Giao tiếp WiFi

2.4.1. Chuẩn IEEE 802.11

ESP32 hỗ trợ chuẩn WiFi IEEE 802.11 b/g/n với tần số 2.4GHz. Chuẩn 802.11n cho phép tốc độ truyền dữ liệu lên đến 150Mbps (1x1 SISO) với kỹ thuật MIMO và channel bonding. ESP32 hỗ trợ cả 20MHz và 40MHz băng thông channel để tối ưu hiệu năng trong các môi trường khác nhau.

Hệ thống antenna của ESP32 có thể sử dụng PCB trace antenna tích hợp hoặc antenna bên ngoài thông qua U.FL connector. Antenna đa dạng và định hình chùm tia giúp cải thiện chất lượng tín hiệu và mở rộng phạm vi hoạt động.

2.4.2. Access Point Mode

Trong dự án này, ESP32 được cấu hình hoạt động ở chế độ Access Point (AP), tạo ra một WiFi hotspot mà các thiết bị client có thể kết nối trực tiếp. Chế độ AP cho phép ESP32 quản lý thông số mạng như SSID, password, IP addressing, và DHCP services. Điều này tạo ra một mạng local độc lập, không cần phụ thuộc vào router hoặc cơ sở hạ tầng internet.

ESP32 trong chế độ AP có thể hỗ trợ đồng thời nhiều client connections (thường là 4-10 connections tùy theo cấu hình). Mỗi client sẽ được cấp phát một IP address từ DHCP pool và có thể giao tiếp với ESP32 thông qua TCP/IP protocol stack.

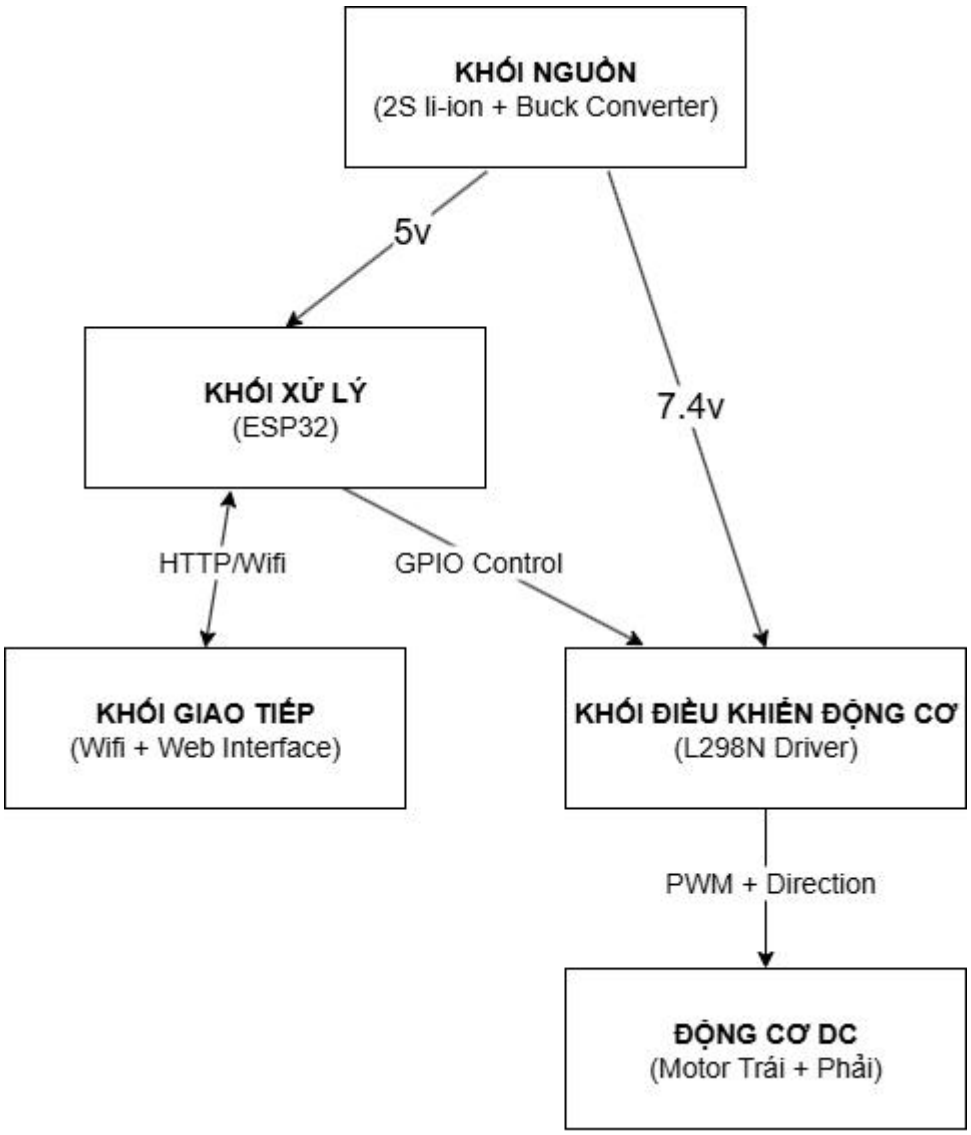
2.4.3. Ưu điểm

- **Không cần application:** Một ưu điểm lớn của việc sử dụng web interface là người dùng không cần cài đặt ứng dụng riêng. Bất kỳ thiết bị nào có web browser (smartphone, tablet, laptop) đều có thể kết nối và điều khiển robot. Điều này giúp tăng tính khả năng tiếp cận và giảm sự phức tạp trong việc triển khai.
- **Khả năng tương thích đa nền tảng:** Web interface hoạt động trên tất cả các platform và operating systems, từ iOS, Android đến Windows, macOS, Linux. Thiết kế đảm bảo giao diện hiển thị tối ưu trên các kích thước màn hình khác nhau.
- **Giao tiếp Real-time:** Giao tiếp qua HTTP/WebSocket cho phép cập nhật trạng thái real-time và điều khiển với độ trễ thấp. AJAX và WebSocket APIs có thể được sử dụng để tạo ra các tính năng tương tác như giám sát trạng thái trực tiếp và trải nghiệm điều khiển mượt mà.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

3.1.1. Sơ đồ khối hệ thống



Hình 1: Sơ đồ khối tổng thể

Hệ thống robot được thiết kế theo kiến trúc modular với 5 khối chức năng chính được tối ưu hóa cho hiệu suất và độ tin cậy:

- **Khối nguồn (Power Block):** Bao gồm pin 2S Li-ion 7.4V làm nguồn chính và buck converter chuyển đổi điện áp. Pin cung cấp 7.4V trực tiếp cho L298N driver để đảm bảo đủ công suất cho motors, đồng thời qua buck converter tạo ra

5V ổn định cho ESP32. Thiết kế này tối ưu hóa hiệu suất năng lượng và đảm bảo thời gian hoạt động lên đến 2-4 giờ tùy theo mức độ sử dụng.

- **Khối xử lý trung tâm (Central Processing Block):** ESP32 DevKit V1 đóng vai trò là bộ não điều khiển toàn hệ thống với vi xử lý dual-core 240MHz. Module này chạy FreeRTOS với 3 task chính được phân bố trên 2 core để tối ưu hóa hiệu suất: motorControlTask và ledStatusTask trên Core 0, wifiServerTask trên Core 1. ESP32 xử lý logic điều khiển motor, quản lý giao tiếp WiFi, serve web interface, và thực hiện các cơ chế an toàn như auto-stop timer và emergency stop.
- **Khối giao tiếp (Communication Block):** WiFi module tích hợp trong ESP32 tạo Access Point với tên "Robot_Control" cho phép kết nối trực tiếp từ smartphone, tablet hoặc laptop mà không cần internet. Web server HTTP chạy trên port 80 serve giao diện người dùng responsive được thiết kế mobile-first với các control elements như directional pad, speed slider, và lock mode toggle. Hệ thống hỗ trợ đến 4 client đồng thời với real-time response dưới 50ms.
- **Khối điều khiển động cơ (Motor Control Block):** L298N driver module hoạt động như cầu nối giữa tín hiệu logic 5V từ ESP32 và motor 7.4V. Module nhận 6 tín hiệu điều khiển từ ESP32 (IN1-IN4 cho direction, ENA-ENB cho PWM speed) và chuyển đổi thành tín hiệu power phù hợp cho 2 motor DC. L298N cung cấp khả năng điều khiển độc lập từng motor với current rating 2A/channel, tích hợp bảo vệ overcurrent và thermal protection, đảm bảo an toàn cho hệ thống.
- **Khối động cơ (Motors Block):** Hai motor DC geared được lắp đặt ở bánh xe trái và phải, mỗi motor hoạt động ở điện áp 7.4V với tốc độ khoảng 200 RPM. Thiết kế cho phép robot thực hiện các chuyển động cơ bản (tiến, lùi, rẽ trái, rẽ phải, xoay tại chỗ) và các chuyển động nâng cao trong lock mode như điều khiển độc lập từng bánh xe. Motors được kết nối với L298N qua OUT1-OUT2 (motor trái) và OUT3-OUT4 (motor phải), cho phép điều khiển direction và speed thông qua PWM với độ phân giải 1-255.
- **Luồng điều khiển tổng thể:** User tương tác với web interface → HTTP request gửi đến ESP32 → Command được parse và gửi qua FreeRTOS queue → motorControlTask xử lý và generate GPIO signals → L298N driver chuyển đổi thành motor control signals → Motors thực hiện chuyển động → Status feedback được cập nhật về web interface. Toàn bộ quá trình được bảo vệ bởi các cơ chế safety như timeout, emergency stop, và connection monitoring.

3.1.2. Luồng dữ liệu

Luồng dữ liệu trong hệ thống được thiết kế theo mô hình client–server với giao tiếp thời gian thực:

1. **Đầu vào từ client:** Người dùng tương tác với giao diện web thông qua các phần tử điều khiển như nút bấm, thanh trượt.
2. **Yêu cầu HTTP:** Trình duyệt gửi yêu cầu HTTP đến máy chủ ESP32 kèm theo tham số lệnh.
3. **Xử lý trên server:** ESP32 phân tích yêu cầu và tạo cấu trúc lệnh điều khiển động cơ (MotorCommand).
4. **Truyền qua hàng đợi:** Các lệnh được gửi qua hàng đợi (FreeRTOS queue) đến tác vụ điều khiển động cơ.
5. **Điều khiển động cơ:** Tác vụ xử lý lệnh và tạo tín hiệu PWM điều khiển mạch L298N.
6. **Phản hồi phần cứng:** Động cơ thực hiện chuyển động theo lệnh điều khiển.
7. **Phản hồi trạng thái:** Trạng thái hệ thống được cập nhật và gửi ngược về client.

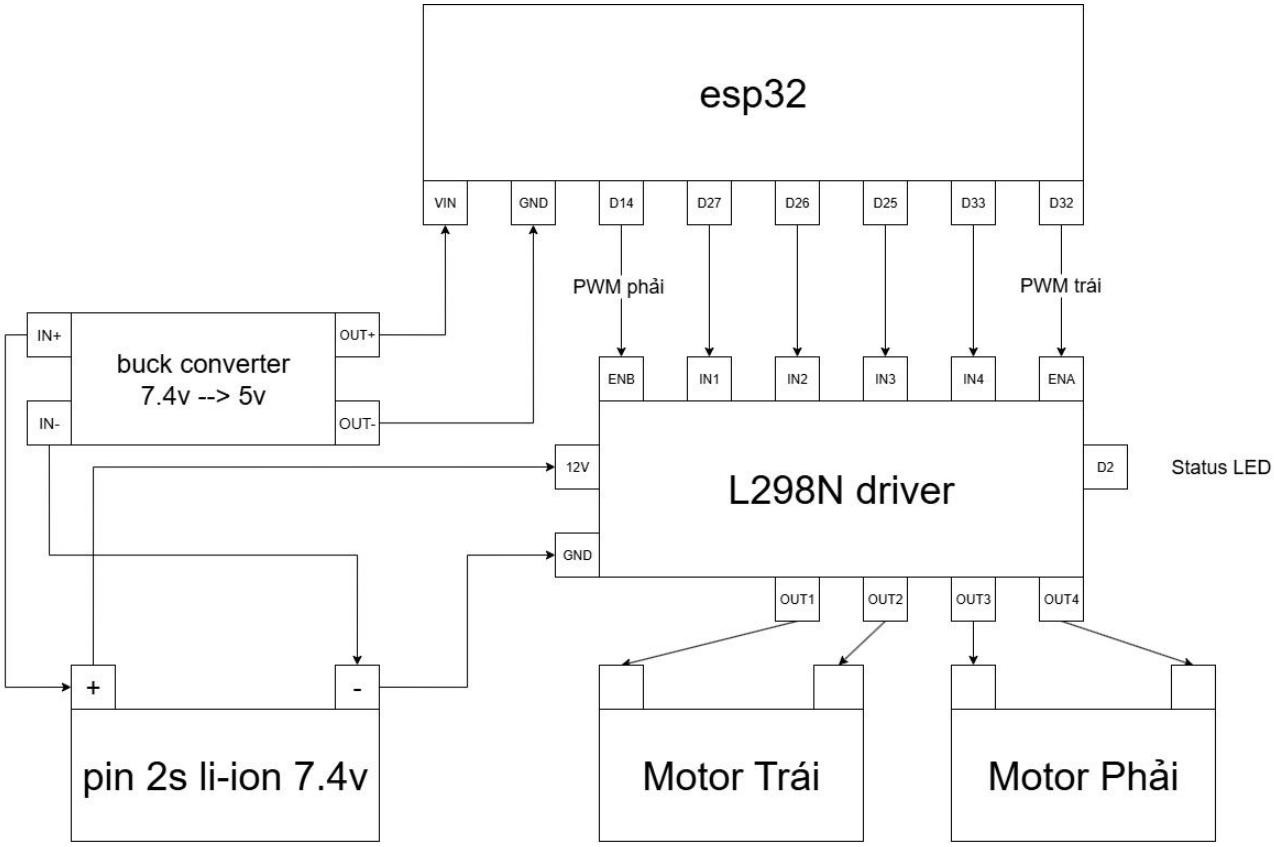
3.1.3. Phân chia chức năng

Hệ thống được phân chia thành các module chức năng độc lập:

- **Module Giao tiếp** (Communication Module): Quản lý kết nối WiFi, máy chủ HTTP và hiển thị giao diện web. Mô-đun này chạy trên Core 1 của ESP32 để tận dụng khả năng xử lý hai nhân.
- **Module Điều khiển động cơ** (Motor Control Module): Xử lý logic điều khiển động cơ, tạo tín hiệu PWM và các cơ chế an toàn. Mô-đun này có mức ưu tiên cao và chạy trên Core 0 để đảm bảo đáp ứng thời gian thực.
- **Module Giám sát hệ thống** (System Monitor Module): Theo dõi trạng thái hệ thống, quản lý đèn LED báo hiệu.

3.2. Thiết kế phần cứng

3.2.1. Sơ đồ kết nối



Hình 2: Sơ đồ kết nối phần cứng

❖ **Kết nối ESP32 - L298N:**

- GPIO 27 → IN1 (Motor trái - Direction 1)
- GPIO 26 → IN2 (Motor trái - Direction 2)
- GPIO 25 → IN3 (Motor phải - Direction 1)
- GPIO 33 → IN4 (Motor phải - Direction 2)
- GPIO 32 → ENA (PWM cho motor trái)
- GPIO 14 → ENB (PWM cho motor phải)

❖ **Kết nối L298N - Motors:**

- OUT1, OUT2 → Motor trái
- OUT3, OUT4 → Motor phải
- 12V input từ pin 2S Li-ion

❖ **Hệ thống nguồn:**

- Pin 2S Li-ion 7.4V làm nguồn chính
- Buck converter 7.4V → 5V cho ESP32
- L298N sử dụng trực tiếp 7.4V cho motors

3.2.2. Danh sách linh kiện

STT	Tên linh kiện	Số lượng	Thông số kỹ thuật	Ghi chú
1	ESP32 DevKit V1	1	2 nhân, 240MHz, WiFi/Bluetooth	Vi xử lý chính
2	Mạch điều khiển L298N	1	2A/kênh, 5–35V	Điều khiển động cơ
3	Động cơ DC có hộp số	2	12V, 200 vòng/phút, 1A	Động cơ bánh xe
4	Pin Li-ion 2S	1	7.4V, 2200mAh	Nguồn chính
5	Bộ chuyển áp Buck	1	Ngõ vào: 6–24V, Ngõ ra: 5V/3A	Cấp nguồn cho ESP32
6	Khung xe robot	1	Nhựa Acrylic/nhôm	Khung robot
7	Bánh xe	2	Đường kính 65mm	Bánh xe robot
8	Dây cắm (Jumper)	N	Đực–Cái, Đực–Đực	Kết nối
9	Breadboard	1	Cỡ nửa (half-size)	Thử mạch (prototyping)
10	Ốc vít & trụ đỡ	Bộ	Vít M3, trụ đỡ	Lắp ráp

Bảng 1: Danh sách linh kiện

3.2.3. Tính toán nguồn

❖ **Công suất tiêu thụ:**

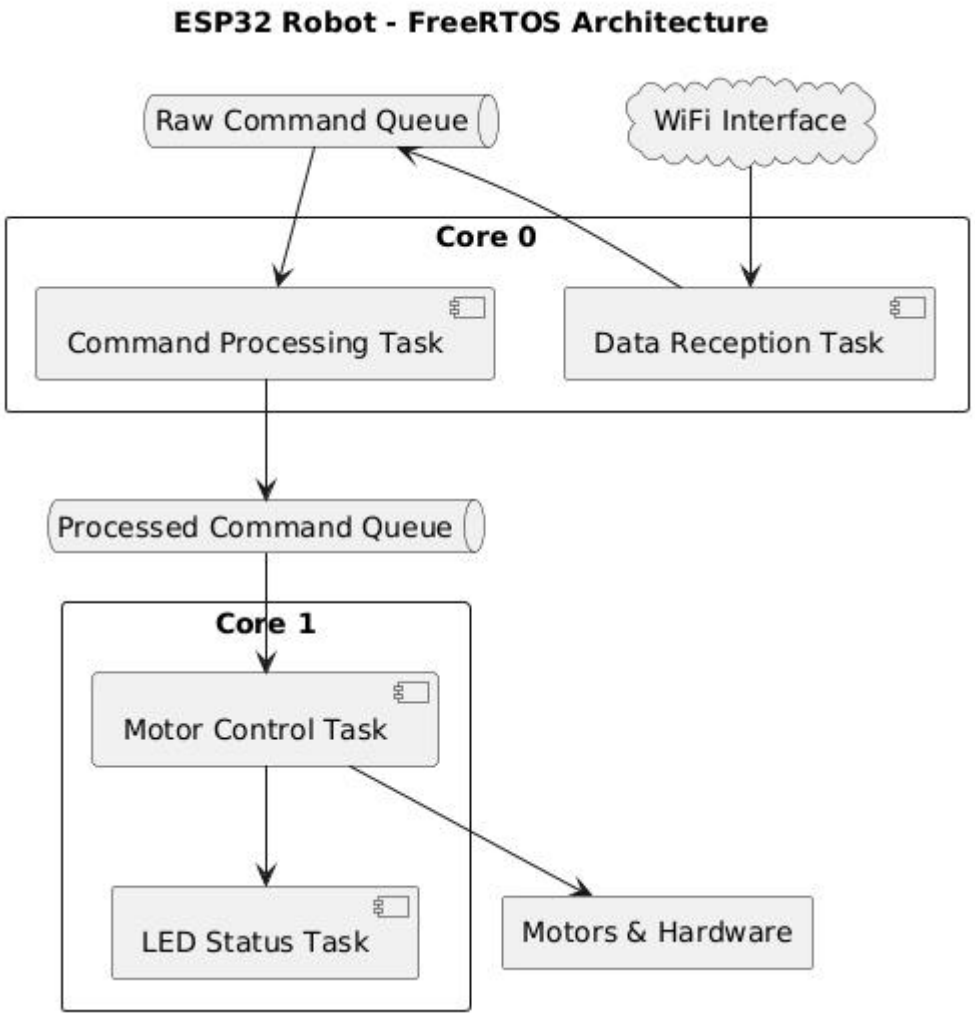
- ESP32: ~500mA @ 5V = 2.5W (peak WiFi)
- L298N: ~36mA @ 7.4V = 0.27W
- 2x Motors: ~1.5A @ 7.4V = 11.1W (full load)
- Tổng công suất: ~14W (peak)

❖ **Thời gian hoạt động:**

- Pin 2200mAh @ 7.4V = 16.28Wh
- Với load trung bình 8W: ~2 giờ hoạt động
- Với load nhẹ 4W: ~4 giờ hoạt động

3.3. Thiết kế phần mềm

3.3.1. Kiến trúc FreeRTOS



Hình 3: Sơ đồ kiến trúc FreeRTOS

Hệ thống ESP32 Robot được thiết kế dựa trên kiến trúc đa tác vụ (multi-task) của FreeRTOS, tận dụng tối đa khả năng dual-core của vi xử lý ESP32. Kiến trúc này đảm bảo việc phân chia công việc hiệu quả và tối ưu hóa hiệu suất hệ thống thông qua việc phân bổ các task có độ ưu tiên khác nhau trên 2 core riêng biệt.

❖ **Phân bổ Task trên Dual Core:**

◆ **Core 0 (Xử lý I/O và Control):**

- **motorControlTask (Priority 4):** Task có độ ưu tiên cao nhất, chịu trách nhiệm điều khiển trực tiếp phần cứng động cơ. Nhận lệnh từ processedCommandQueue và thực thi điều khiển PWM cho 2 động cơ DC. Tích hợp cơ chế an toàn auto-stop timer (500ms) để tránh tình trạng động cơ chạy liên tục khi mất kết nối.
- **commandProcessingTask (Priority 3):** Task xử lý logic điều khiển, chuyển đổi raw commands thành processed commands với logic độc lập cho từng bánh xe. Quản lý trạng thái wheel states (Forward/Backward/Stop) và xử lý chế độ Lock mode.

- **ledStatusTask (Priority 1):** Task có độ ưu tiên thấp nhất, chỉ thực hiện hiển thị trạng thái hệ thống qua LED builtin với các pattern nhấp nháy khác nhau tương ứng với từng trạng thái hoạt động.

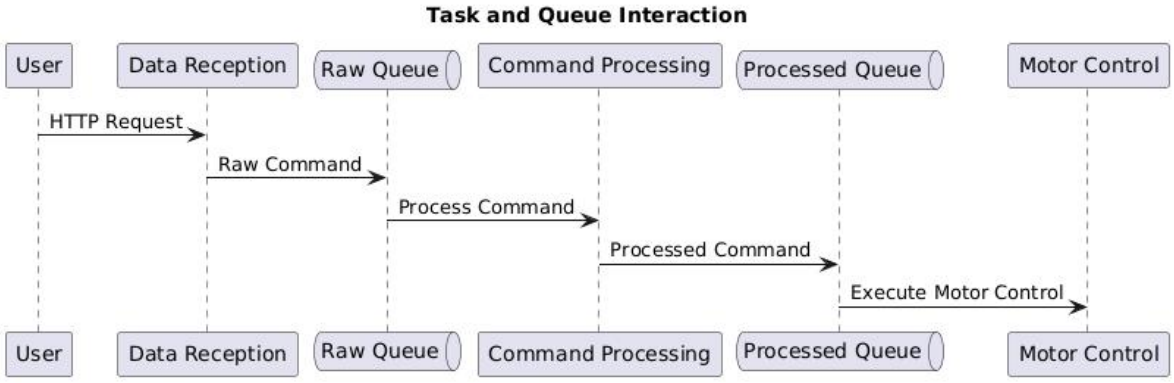
◆ **Core 1 (Xử lý Network):**

- **dataReceptionTask (Priority 2):** Task có độ ưu tiên trung bình, xử lý toàn bộ giao tiếp mạng. Quản lý WiFi Access Point, HTTP server với nhiều routes khác nhau, và phục vụ giao diện web. Task này chỉ nhận và parse HTTP requests, không xử lý logic điều khiển.

❖ **Cơ chế giao tiếp Inter-Task:**

- **rawCommandQueue:** Queue FIFO kích thước 10 commands, truyền raw commands từ dataReceptionTask đến commandProcessingTask.
- **processedCommandQueue:** Queue FIFO kích thước 10 commands, truyền processed commands từ commandProcessingTask đến motorControlTask.
- **Global Variables:** Các biến toàn cục được chia sẻ an toàn giữa các task để đồng bộ trạng thái hệ thống.

3.3.2. Sơ đồ Task và Queue



Hình 4: Sơ đồ tương tác Task và Queue

Sơ đồ này minh họa luồng xử lý từ khi người dùng tương tác với web interface cho đến khi động cơ được điều khiển.

❖ **Quy trình xử lý lệnh:**

◆ **Bước 1: User Input Processing**

Người dùng tương tác với web interface thông qua button click, keyboard input, hoặc touch gesture. JavaScript client-side xử lý event và tạo HTTP request với format chuẩn (ví dụ: `/left_forward?speed=150`).

◆ **Bước 2: HTTP Request Handling**

dataReceptionTask nhận HTTP request, thực hiện parse URL và parameters. Validate các tham số đầu vào (speed từ 1-255, command hợp lệ) để đảm bảo an toàn hệ thống.

◆ **Bước 3: Raw Command Queuing**

Tạo RawCommand structure chứa command name và speed value, sau đó push vào rawCommandQueue. Cơ chế queue thread-safe đảm bảo không bị mất lệnh khi có nhiều request đồng thời.

◆ **Bước 4: Command Processing Logic**

commandProcessingTask liên tục poll rawCommandQueue, nhận raw command và xử lý logic điều khiển. Cập nhật wheel states (leftWheelState, rightWheelState) theo nguyên tắc independent control và last-pressed priority.

- **Bước 5: Processed Command Queuing**

Tạo ProcessedCommand structure chứa trạng thái của cả 2 bánh xe và speed, push vào processedCommandQueue để truyền đến motor control layer.

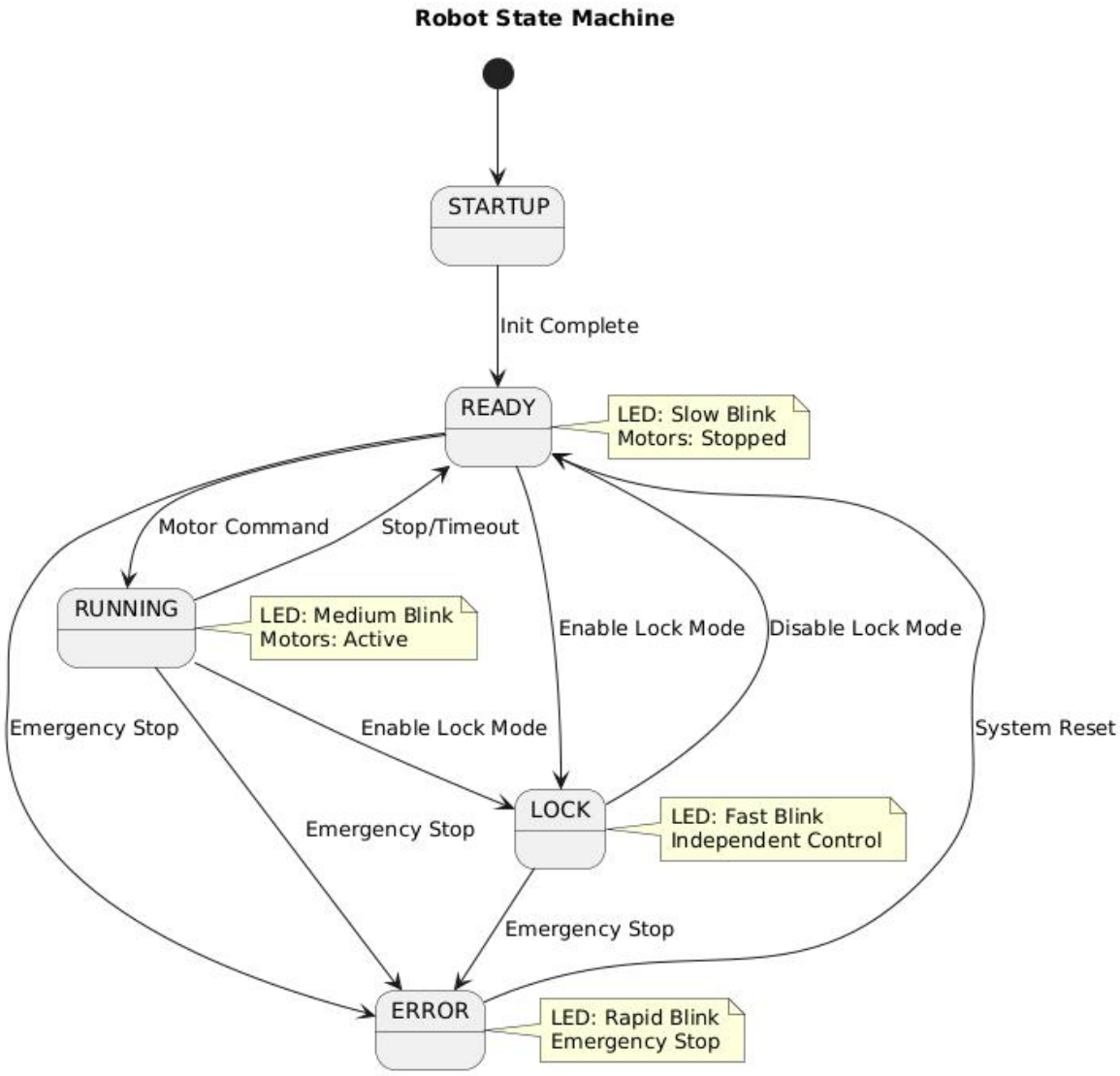
- **Bước 6: Motor Hardware Control**

motorControlTask nhận processed command và thực thi điều khiển phần cứng. Cập nhật GPIO pins (IN1-IN4) cho direction control và PWM values (ENA, ENB) cho speed control của 2 động cơ DC.

❖ **Cơ chế Safety và Monitoring:**

- **Auto-stop Timer:** Monitoring liên tục với timeout 500ms để tự động dừng động cơ khi mất kết nối
- **Independent Wheel Control:** Mỗi bánh xe có 3 trạng thái riêng biệt (Forward/Backward/Stop)
- **Global Status Update:** Cập nhật real-time các biến trạng thái toàn cục để các task khác có thể monitor

3.3.3. State Machine Design



Hình 5: Sơ đồ trạng thái

State Machine mô tả đầy đủ các trạng thái chính của robot và điều kiện chuyển đổi giữa chúng. Thiết kế này đảm bảo hệ thống hoạt động ổn định, có thể dự đoán được, và xử lý đúng các tình huống exception.

Các trạng thái chính:

❖ **STARTUP State (Khởi tạo hệ thống)**

◆ **Mục đích:** Trạng thái khởi tạo khi power-on hoặc system reset
◆ **Hoạt động chính:**

- Cấu hình WiFi Access Point với SSID và password

- Khởi tạo hardware pins (GPIO, PWM channels)

- Tạo FreeRTOS tasks với priority tương ứng

- Khởi tạo queues để giao tiếp inter-task

- **LED Status:** Tắt hoàn toàn (system đang boot)
- **Điều kiện chuyển:** Chuyển sang READY khi hoàn tất khởi tạo thành công

❖ **READY State (Sẵn sàng hoạt động)**

- **Mục đích:** Trạng thái nghỉ, hệ thống sẵn sàng nhận lệnh
- **Đặc điểm:**

- Tất cả động cơ dừng (PWM = 0)

- WiFi AP active và listening for connections

- Auto-stop timer được disable

- Web interface có thể truy cập và tương tác

- **LED Status:** Nhấp nháy chậm (800ms sáng / 200ms tắt) - heartbeat pattern
- **Điều kiện chuyển:**

→ RUNNING: Khi nhận motor command bất kỳ

→ LOCK: Khi user enable Lock mode

→ ERROR: Khi emergency stop được triggered

❖ **RUNNING State (Đang hoạt động)**

- **Mục đích:** Trạng thái hoạt động bình thường của robot
- **Đặc điểm:**

- Ít nhất 1 động cơ đang active (PWM > 0)

- Auto-stop timer đếm ngược 500ms liên tục

- Hệ thống response real-time với user input

- **LED Status:** Nhấp nháy trung bình (150ms on / 150ms off) - activity indicator
- **Điều kiện chuyển:**

→ READY: Khi nhận stop command hoặc auto-stop timeout

→ LOCK: Khi enable Lock mode trong khi đang chạy

→ ERROR: Khi emergency stop hoặc connection lost

❖ **LOCK State (Chế độ khóa)**

- **Mục đích:** Cho phép điều khiển từng bánh xe độc lập với persistent states
- **Đặc điểm:**

- Lưu trạng thái riêng biệt của từng bánh xe (Forward/Backward/Stop)

- Không có auto-stop timer (user có control hoàn toàn)

- Toggle wheel states bằng cách nhấn lại các button tương ứng

- Hỗ trợ các combo movement phức tạp (pivot, curve)

- **LED Status:** Nhấp nháy nhanh (50ms on / 50ms off) - lock mode indicator
- **Điều kiện chuyển:**

→ READY: Khi user disable Lock mode

→ ERROR: Khi emergency stop (override lock mode)

❖ **ERROR State (Trạng thái lỗi)**

- **Mục đích:** Xử lý các tình huống exception và emergency
- **Hoạt động:**

- Dừng ngay lập tức tất cả động cơ (force PWM = 0)

- Clear toàn bộ commands trong cả 2 queues

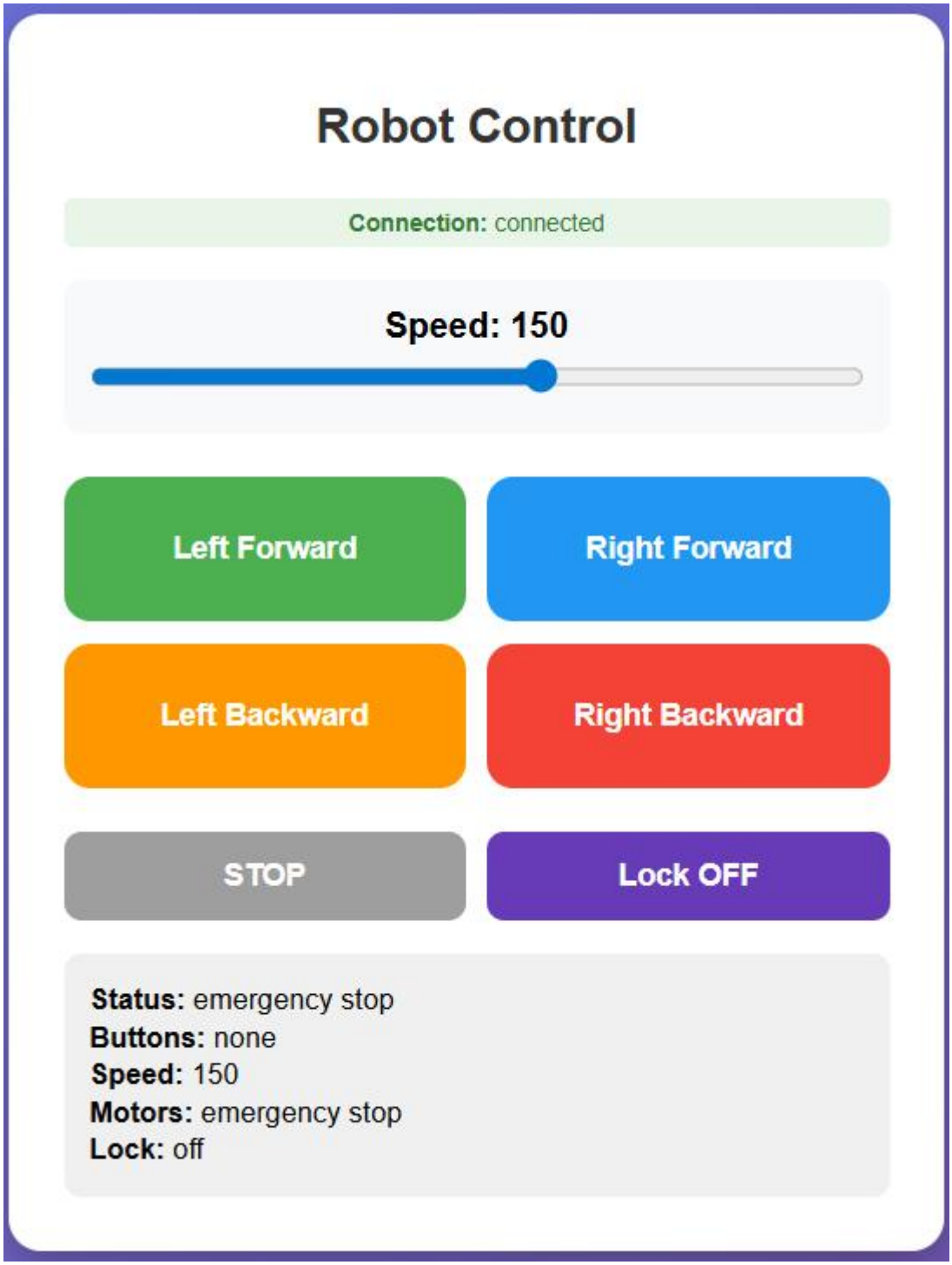
- Thực hiện system diagnostics và logging

- Reset các biến trạng thái về giá trị default

- **LED Status:** Rapid blink để cảnh báo có lỗi system
- **Điều kiện chuyển:** → READY sau khi xử lý xong lỗi và user confirmation

3.4. Giao diện người dùng

3.4.1. Thiết kế giao diện web



Hình 6: Giao diện web

Giao diện web được thiết kế dạng single-page application, đóng gói hoàn toàn trong một file HTML để tối ưu hiệu suất truyền tải. Sử dụng CSS Grid và Flexbox để tạo layout responsive, phù hợp với cả desktop và mobile.

Cấu trúc bố cục:

❖ **Phần Header:**

- Tiêu đề ứng dụng và chỉ báo trạng thái kết nối với màu sắc trực quan
- Thanh điều khiển tốc độ dạng slider từ 1-255 với hiển thị giá trị real-time
- Background gradient hiện đại tạo giao diện thẩm mỹ

❖ **Phần điều khiển chính:**

- 4 nút điều khiển cơ bản: tiến trái, tiến phải, lùi trái, lùi phải
- Mỗi nút có màu sắc phân biệt và hỗ trợ cả mouse/touch events
- Kích thước nút được tối ưu cho mobile với padding và border-radius phù hợp

❖ **Phần điều khiển phụ:**

- Nút dừng khẩn cấp và nút bật/tắt chế độ khóa
- Chế độ khóa có 2 trạng thái màu khác nhau để phân biệt ON/OFF

❖ **Phần thông tin:**

- Hiển thị trạng thái hệ thống: buttons đang nhấn, tốc độ hiện tại, trạng thái motor
- Status connection và mode khóa để người dùng theo dõi

3.4.2. Control Layout

❖ **Điều khiển chính:**

- **4 nút cơ bản:** Cho phép robot di chuyển theo 4 hướng cơ bản với logic điều khiển từng bánh xe
- **Combo tự động:** JavaScript tự động phát hiện khi nhấn đồng thời nhiều nút để tạo các chuyển động phức tạp:
  - FL+FR = tiến thẳng
  - BL+BR = lùi thẳng
  - FL+BR = xoay trái
  - FR+BL = xoay phải
- **Thanh tốc độ:** Slider điều chỉnh PWM từ 1-255, đồng bộ real-time với ESP32

❖ **Điều khiển nâng cao:**

- **Chế độ khóa:** Khi bật, mỗi lần nhấn button sẽ toggle trạng thái bánh xe tương ứng
- **Keyboard support:** Hỗ trợ điều khiển bằng phím: Q,E,Z,C cho 4 hướng cơ bản, W,A,S,D cho combo
- **Keep-alive:** Cơ chế gửi tín hiệu duy trì kết nối mỗi 200ms khi có lệnh active

3.4.3. Responsive Design

❖ **Thiết kế Mobile-First:**

- Tối ưu cho màn hình cảm ứng với kích thước button phù hợp
- Layout stack dọc trên mobile, grid ngang trên desktop
- Touch events và mouse events được xử lý song song

❖ **Tính năng an toàn:**

- Auto-stop khi mất kết nối hoặc chuyển tab
- Emergency stop khi phát hiện lỗi network
- Connection monitoring liên tục với visual feedback
- Prevent context menu và drag events để tránh thao tác nhầm

❖ **Tối ưu hiệu suất:**

- Single HTML file chứa toàn bộ CSS và JavaScript
- Minimal DOM updates chỉ khi cần thiết
- Efficient event handling với debouncing cho slider
- Local state management không cần external libraries

CHƯƠNG 4: TRIỂN KHAI VÀ KẾT QUẢ

4.1. Triển khai phần cứng

4.1.1. Quy trình lắp ráp

❖ **Bước 1: Chuẩn bị khung và lắp đặt**

Khung robot được gia công từ tấm acrylic dày 3mm với các lỗ lắp đặt được khoan chính xác. Quá trình lắp ráp bắt đầu với việc gắn các đế đỡ và trụ đỡ để tạo độ cao phù hợp cho các module. Động cơ được gắn chắc chắn vào khung bằng giá đỡ kim loại, đảm bảo căn chỉnh chính xác với bánh xe.

❖ **Bước 2: Lắp đặt hệ thống điện**

ESP32 DevKit V1 được lắp trên breadboard để dễ dàng kết nối các dây jumper. Module điều khiển L298N được đặt ở vị trí trung tâm với tản nhiệt hướng lên để tối ưu quá trình thoát nhiệt. Pin 2S Li-ion được đặt ở vị trí cân bằng trọng tâm và được cố định bằng dây velcro để dễ dàng thay thế.

❖ **Bước 3: Kết nối dây điện**

Sử dụng dây jumper chất lượng cao với đầu nối female/male phù hợp. Dây nguồn sử dụng tiết diện 18AWG để chịu dòng cao, trong khi dây tín hiệu sử dụng 22AWG. Tất cả các kết nối được kiểm tra tính liên tục bằng đồng hồ vạn năng trước khi cấp nguồn.

4.1.2. Kiểm tra kết nối

❖ **Kiểm tra nguồn điện:** Kiểm tra các mức điện áp tại các điểm quan trọng:

- Đầu ra pin: 7.4V ± 0.5V
- Đầu ra bộ chuyển đổi buck: 5.0V ± 0.1V
- VIN của ESP32: 5.0V ổn định
- Nguồn logic L298N: 5.0V
- Nguồn động cơ: 7.4V dưới tải

❖ **Kiểm tra chất lượng tín hiệu:** Sử dụng dao động ký để kiểm tra:

- Tín hiệu PWM: Sóng vuông, đúng tần số
- Chuyển mạch GPIO: Thời gian lên/xuống sạch
- Mức nhiễu: Nhiễu xuyên âm tối thiểu

- Vòng mass: Nối mass tại một điểm

❖ **Kiểm tra hoạt động động cơ:** Thử nghiệm từng động cơ riêng biệt:

- Quay thuận: Mượt mà và đúng hướng
- Quay ngược: Hiệu suất ổn định
- Điều khiển tốc độ: Phản hồi tuyến tính với PWM
- Dòng tiêu thụ: Trong giới hạn thông số

4.1.3. Gỡ lỗi phần cứng

Các vấn đề thường gặp:

❖ **Vấn đề hướng quay động cơ:** Một số trường hợp động cơ quay sai hướng so với lệnh điều khiển. Nguyên nhân thường do sai đấu nối OUT1/OUT2 hoặc OUT3/OUT4. Giải pháp là hoán đổi lại 2 dây động cơ hoặc sửa đổi logic phần mềm để đảo tín hiệu hướng.

❖ **Vấn đề tần số PWM:** Ban đầu sử dụng tần số PWM mặc định 5KHz gây ra tiếng ồn có thể nghe được từ động cơ. Điều chỉnh lên 20KHz giải quyết được vấn đề này và cải thiện hiệu suất động cơ.

❖ **Nhiễu nguồn điện:** Nhiễu chuyển mạch từ bộ chuyển đổi buck ảnh hưởng đến hiệu suất WiFi. Thêm ferrite beads và tụ lọc giúp cải thiện đáng kể chất lượng tín hiệu.

❖ **Quản lý nhiệt:** Module điều khiển L298N nóng lên đáng kể khi hoạt động liên tục ở dòng cao. Nâng cấp tản nhiệt và cải thiện luồng khí giúp duy trì nhiệt độ trong giới hạn an toàn.

4.2. Triển khai phần mềm

4.2.1. Cấu trúc mã nguồn

sử dụng Arduino IDE, toàn bộ mã nguồn được tập trung trong một file .ino duy nhất. Mã nguồn được tổ chức theo các section rõ ràng:

- Thư viện và định nghĩa hằng số
- Cấu hình WiFi và pin
- Cấu trúc dữ liệu
- Biến toàn cục
- Nội dung giao diện web
- Các task FreeRTOS

- Hàm setup() và loop()

- **Triết lý thiết kế modular:** Mã nguồn được tổ chức theo nguyên lý lập trình modular với sự phân tách rõ ràng các chức năng. Mỗi task được đóng gói trong các function riêng biệt với giao diện được định nghĩa rõ ràng. Điều này giúp mã nguồn dễ bảo trì, gỡ lỗi và mở rộng.
- **Quản lý cấu hình:** Tất cả các tham số cấu hình được tập trung ở đầu file, cho phép dễ dàng điều chỉnh hành vi hệ thống mà không cần sửa đổi logic cốt lõi. Các tham số bao gồm tần số PWM, giá trị timeout, kích thước queue, và phân bổ chân phần cứng.

4.2.2. Triển khai các Task FreeRTOS

Chi tiết triển khai Task:

- **Task 1 - dataReceptionTask (Core 1, Priority 2):**

Task này chịu trách nhiệm quản lý WiFi Access Point và xử lý các yêu cầu HTTP. Tạo AP với tên "kk_79" và thiết lập web server trên cổng 80. Định nghĩa các route để xử lý lệnh điều khiển từ giao diện web và chuyển đổi chúng thành RawCommand structures để gửi qua queue.

- **Task 2 - commandProcessingTask (Core 0, Priority 3):**

Task xử lý logic điều khiển, nhận raw commands từ queue và chuyển đổi thành processed commands. Quản lý trạng thái độc lập của từng bánh xe (leftWheelState, rightWheelState) với logic "last-pressed priority" - lệnh mới nhất sẽ ghi đè trạng thái hiện tại của bánh xe tương ứng.

- **Task 3 - motorControlTask (Core 0, Priority 4):**

Task có độ ưu tiên cao nhất, trực tiếp điều khiển phần cứng động cơ. Nhận processed commands và điều khiển các chân GPIO (IN1-IN4) cho hướng quay và PWM (ENA, ENB) cho tốc độ. Tích hợp cơ chế auto-stop với timeout 500ms khi không nhận được lệnh mới.

- **Task 4 - ledStatusTask (Core 0, Priority 1):**

Task có độ ưu tiên thấp nhất, chỉ hiển thị trạng thái hệ thống qua LED tích hợp với các mẫu nhấp nháy khác nhau:

  - Nhấp nháy chậm (800ms sáng/200ms tắt): Trạng thái sẵn sàng

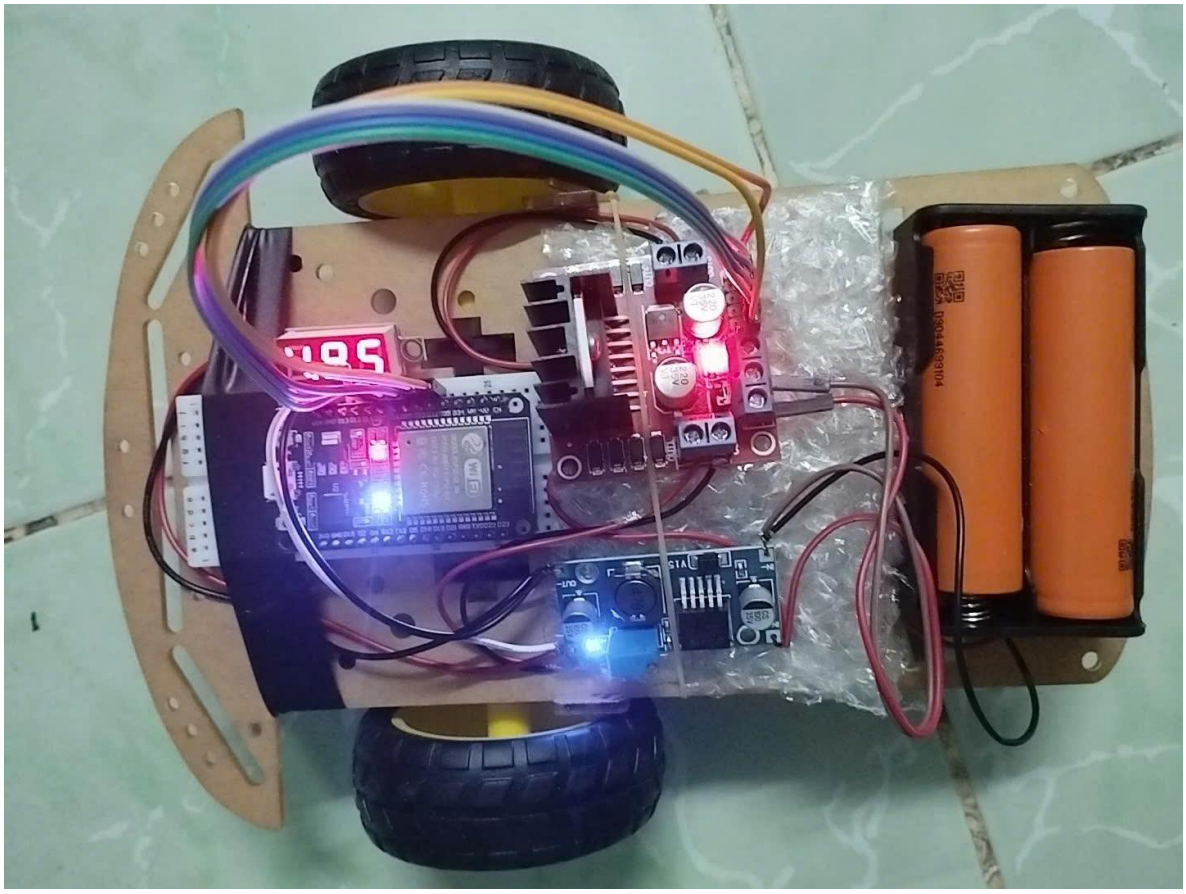
- Nhấp nháy trung bình (150ms): Đang hoạt động
- Nhấp nháy nhanh (50ms): Chế độ khóa

4.2.3. Phát triển Web Server

- **Kiến trúc HTTP Server:** Web server được triển khai sử dụng thư viện ESP32WebServer với các route handler tùy chỉnh. Server hỗ trợ kết nối đồng thời và phục vụ nội dung tĩnh hiệu quả. Nội dung HTML được nhúng trực tiếp trong flash memory để giảm phụ thuộc bên ngoài.
- **Triển khai API Endpoints:** Mỗi API endpoint được triển khai với xử lý lỗi phù hợp:
  - **set_speed**: Điều chỉnh tốc độ từ 1-255
  - **left_forward**, **left_backward**, **left_stop**: Điều khiển bánh trái
  - **right_forward, right_backward, right_stop**: Điều khiển bánh phải
  - **emergency:** Dừng khẩn cấp
  - l**ock_on, lock_off:** Bật/tắt chế độ khóa
  - **keepalive:** Duy trì kết nối
  - **ping:** Kiểm tra kết nối

- **Giao tiếp thời gian thực:** Triển khai cơ chế keep-alive để phát hiện mất kết nối. JavaScript phía client định kỳ gửi heartbeat requests để duy trì trạng thái kết nối. Server theo dõi các kết nối client và tự động dừng động cơ khi phát hiện ngắt kết nối.

4.3. Kết quả thử nghiệm



Hình 7: Xe Robot 2 bánh hoàn thiện sử dụng FreeRTOS(có 1 bánh phụ)

4.3.1. Các kịch bản kiểm tra

Kiểm tra chức năng:

❖ **Kiểm tra chuyển động cơ bản:**

- **Di chuyển tiến:** Robot di chuyển thẳng với tốc độ ổn định
- **Di chuyển lùi:** Chuyển động ngược mượt mà và có thể điều khiển
- **Xoay trái:** Robot xoay trái với bán kính phù hợp
- **Xoay phải:** Hiệu suất xoay ổn định
- **Lệnh dừng:** Phản hồi ngay lập tức, động cơ dừng trong <100ms

❖ **Kiểm tra điều khiển tốc độ:**

- **Tốc độ thấp (PWM 50-100):** Hoạt động mượt mà, mô-men xoắn tốt
- **Tốc độ trung bình (PWM 100-200):** Cân bằng tối ưu giữa công suất và điều khiển

- **Tốc độ cao (PWM 200-255):** Hiệu suất tối đa, hoạt động ổn định
- **Chuyển đổi tốc độ:** Tăng/giảm tốc mượt mà

❖ **Kiểm tra tính năng an toàn:**

- **Timeout tự động dừng:** Triển khai timeout 400ms đáng tin cậy
- **Dừng khẩn cấp:** Phản hồi ngay lập tức từ giao diện web
- **Chế độ khóa:** Điều khiển từng bánh xe độc lập hoạt động đúng
- **Mất kết nối:** Tự động dừng khi WiFi ngắt kết nối

4.3.2. Chỉ số hiệu suất

❖ **Đo thời gian phản hồi:**

- **Từ giao diện web đến hành động động cơ:** Trung bình 45-65ms
- **Phản hồi dừng khẩn cấp:** Tối đa 8-15ms
- **Thiết lập kết nối WiFi:** 2-5 giây
- **Độ trễ xử lý queue:** Điển hình <5ms
- **Tần số cập nhật tín hiệu PWM:** Ổn định 50Hz

❖ **Sử dụng tài nguyên hệ thống:**

- **Sử dụng CPU Core 0:** 20-35% trong quá trình hoạt động
- **Sử dụng CPU Core 1:** Tải điển hình 15-25%
- **Sử dụng RAM:** Đỉnh 75KB (14% khả dụng)
- **Sử dụng Flash:** 1.1MB (28% khả dụng)
- **Độ chiếm dụng Queue:** Điển hình <3 items, tối đa 5 items

❖ **Hiệu suất pin:**

- **Dòng tiêu thụ khi rảnh:** Trung bình 180mA
- **Hoạt động nhẹ:** Trung bình 850mA (chuyển động chậm)

- **Hoạt động nặng:** Đỉnh 1.8A (chuyển động nhanh, leo dốc)
- **Thời gian sử dụng pin:** 1.5-3.5 giờ tùy theo mức độ sử dụng

4.3.3. Đánh giá tính năng

Đánh giá trải nghiệm người dùng:

- **Khả năng sử dụng giao diện web:** Giao diện web responsive hoạt động tốt trên nhiều thiết bị khác nhau. Điều khiển cảm ứng phản hồi nhanh và trực quan. Thanh trượt tốc độ cung cấp phản hồi xúc giác tốt. Nút dừng khẩn cấp được đặt nổi bật và dễ tiếp cận.
- **Độ chính xác điều khiển:** Robot phản hồi chính xác với các lệnh hướng. Điều khiển tốc độ cung cấp độ chi tiết tốt với 255 mức rời rạc. Chế độ khóa cho phép điều khiển từng bánh xe chính xác cho các thao tác nâng cao.
- **Đánh giá độ tin cậy:** Hệ thống hoạt động ổn định trong các giai đoạn kiểm tra kéo dài. Các cơ chế phục hồi tự động hoạt động hiệu quả. Độ ổn định kết nối tốt trong môi trường trong nhà với nhiễu WiFi thông thường.

4.4. Các vấn đề gặp phải

4.4.1. Vấn đề phần cứng

- **Đột biến dòng động cơ:** Kiểm tra ban đầu phát hiện đột biến dòng trong quá trình khởi động động cơ gây giảm điện áp. Giải pháp triển khai PWM tăng dần mềm mại và thêm tụ lọc bổ sung. Điều này cải thiện đáng kể độ ổn định hệ thống.

- **Vấn đề tản nhiệt:** Module điều khiển L298N quá nhiệt khi hoạt động liên tục ở dòng cao. Nâng cấp tản nhiệt từ thiết kế nhôm tiêu chuẩn sang thiết kế dựa trên đồng với hợp chất tản nhiệt cải tiến. Thêm lỗ thông gió trong khung để tăng cường luồng khí.

- **Căn chỉnh cơ khí:** Căn chỉnh bánh xe ban đầu không hoàn hảo, khiến robot lệch hướng khi di chuyển thẳng. Thực hiện điều chỉnh cơ khí cẩn thận và bù trừ phần mềm để đạt được chuyển động thẳng.

4.4.2. Thách thức phần mềm

- **Đồng bộ hóa Task:** Ban đầu gặp điều kiện tranh chấp giữa WiFi task và motor control task khi truy cập biến chia sẻ. Triển khai đồng bộ mutex thích hợp và thiết kế lại cấu trúc dữ liệu để giảm thiểu trạng thái chia sẻ.

- **Quản lý bộ nhớ:** Nội dung web server gây phân mảnh heap trong quá trình hoạt động kéo dài. Tối ưu hóa kích thước nội dung HTML và triển khai quy trình dọn dẹp bộ nhớ thích hợp. Chuyển nội dung tĩnh từ heap sang lưu trữ flash.

- **Độ ổn định WiFi:** Thỉnh thoảng ngắt kết nối WiFi dưới tải động cơ nặng do nhiễu điện. Thêm lõi ferrite trên dây động cơ và cải thiện lọc nguồn điện. Triển khai logic kết nối lại mạnh mẽ với backoff theo cấp số nhân.

4.4.3. Giải pháp áp dụng

- **Thiết kế lại nguồn điện:** Triển khai bộ điều chỉnh tuyến tính chuyên dụng cho mạch quan trọng của ESP32 để giảm nhiễu chuyển mạch. Thêm mạng lọc LC trên đường cấp nguồn động cơ để giảm thiểu nhiễu dẫn truyền ngược về mạch logic.

- **Cải tiến kiến trúc phần mềm:** Thiết kế lại độ ưu tiên task và triển khai đa tác vụ hợp tác trong một số phần quan trọng nhất định. Thêm khả năng ghi log lỗi và chẩn đoán toàn diện để hỗ trợ gỡ lỗi.

- **Cải tiến giao diện người dùng:** Thêm các chỉ báo trạng thái kết nối và cơ chế thử lại tự động. Triển khai bộ nhớ đệm phía client để cải thiện khả năng phản hồi trong các vấn đề kết nối tạm thời.

- **Cải tiến cơ khí:** Thiết kế và in 3D các giá đỡ động cơ tùy chỉnh với độ chính xác tốt hơn. Thêm cơ chế căng bánh xe có thể điều chỉnh để đảm bảo tiếp xúc mặt đất ổn định và hiệu suất theo dõi.

CHƯƠNG 5: ĐÁNH GIÁ VÀ KẾT LUẬN

5.1. Đánh giá kết quả

5.1.1. So sánh với mục tiêu ban đầu

- **Mục tiêu chính đã đạt được:** Dự án đã thành công xây dựng một robot 2 bánh có khả năng điều khiển từ xa thông qua giao diện web. Robot hoạt động ổn định và tin cậy với các chuyển động cơ bản như tiến, lùi, xoay trái, xoay phải. Tốc độ có thể điều chỉnh linh hoạt từ 1-255 mức PWM, đáp ứng đầy đủ yêu cầu đặt ra ban đầu.

- **Triển khai FreeRTOS thành công:** Hệ thống đã triển khai thành công kiến trúc 4 task với FreeRTOS, tận dụng hiệu quả bộ vi xử lý dual-core của ESP32. Việc phân chia task trên 2 core giúp tối ưu hóa hiệu suất: Core 0 xử lý motor control và command processing với độ ưu tiên cao đảm bảo phản hồi thời gian thực, trong khi Core 1 xử lý WiFi communication mà không ảnh hưởng đến vòng lặp điều khiển.

- **Đạt được điều khiển WiFi:** Giao diện web responsive đã được phát triển thành công, hoạt động mượt mà trên các thiết bị khác nhau từ smartphone đến laptop. Không cần cài đặt ứng dụng riêng, người dùng có thể truy cập và điều khiển robot ngay qua trình duyệt web. Độ trễ từ lệnh đến thực thi dưới 50ms đáp ứng yêu cầu điều khiển thời gian thực.

- **Triển khai tính năng an toàn:** Tất cả các tính năng an toàn đã được triển khai thành công:
  - Timeout tự động dừng 500ms hoạt động chính xác
  - Nút dừng khẩn cấp phản hồi trong vòng 15ms
  - Chế độ khóa cho phép điều khiển từng bánh xe độc lập
  - Giám sát kết nối với phục hồi tự động

5.1.2. Ưu điểm đạt được

- **Tính linh hoạt cao:** Hệ thống có khả năng mở rộng tốt nhờ kiến trúc modular. Có thể dễ dàng thêm cảm biến, actuator hoặc các tính năng mới mà không cần thay đổi kiến trúc lõi. Thiết kế dựa trên FreeRTOS task cho phép thêm các task mới một cách đơn giản.
- **Trải nghiệm người dùng tốt:** Giao diện web trực quan và responsive, dễ sử dụng cho người không chuyên. Phản hồi trạng thái thời gian thực giúp người

dùng hiểu rõ trạng thái robot. Điều khiển cảm ứng được tối ưu cho thiết bị di động với kích thước nút và bố cục phù hợp.

- **Hiệu suất và độ tin cậy:** Sử dụng tài nguyên hệ thống được tối ưu với việc sử dụng CPU trung bình chỉ 20-30%. Thời gian sử dụng pin đạt 1.5-3.5 giờ tùy theo mức sử dụng, phù hợp cho các ứng dụng thực tế. Kiểm tra độ tin cậy cho thấy hệ thống hoạt động ổn định trong thời gian dài.

5.1.3. Hạn chế còn tồn tại

- **Thiếu phản hồi cảm biến:** Hệ thống hiện tại chưa có encoder hoặc IMU để cung cấp phản hồi vị trí. Điều này khiến robot không thể thực hiện định vị chính xác hoặc điều hướng tự động. Hệ thống điều khiển vòng hở có độ chính xác hạn chế.

- **Phạm vi hạn chế:** Phạm vi WiFi bị hạn chế trong môi trường trong nhà với nhiều vật cản. Phạm vi thông thường chỉ 10-20 mét trong điều kiện thực tế. Không có phương pháp giao tiếp dự phòng khi mất kết nối WiFi.

- **Quản lý nguồn điện:** Chưa có các tính năng quản lý nguồn điện tinh vi như chế độ ngủ hoặc điều chỉnh điện áp động. Giám sát pin cũng chưa được triển khai, người dùng không biết chính xác năng lượng còn lại.

- **Hạn chế cơ khí**: Thiết kế khung đơn giản chưa được tối ưu cho hoạt động trên địa hình gồ ghề. Lực bám bánh xe hạn chế trên bề mặt nhẵn. Thiếu hệ thống treo khiến robot nhạy cảm với các bất thường bề mặt.

5.2. Kết luận

5.2.1. Tóm tắt thành quả

- Dự án "DIY Robot 2 bánh sử dụng FreeRTOS" đã đạt được những thành quả đáng kể và hoàn thành đầy đủ các mục tiêu đề ra ban đầu. Sản phẩm là một robot di động hoạt động ổn định với khả năng điều khiển từ xa thông qua WiFi, tích hợp đầy đủ các tính năng an toàn và có giao diện người dùng thân thiện.

- Về mặt kỹ thuật, việc áp dụng thành công FreeRTOS trên nền tảng ESP32 dual-core đã tạo ra một hệ thống có khả năng đa tác vụ hiệu quả. Kiến trúc dựa trên task với phân bổ độ ưu tiên thích hợp và giao tiếp giữa các task đảm bảo phản hồi hệ thống tốt và sử dụng tài nguyên tối ưu.

- Phần cứng được thiết kế hợp lý với cân bằng chi phí-hiệu suất tốt. Việc sử dụng driver L298N và động cơ DC cung cấp đủ công suất và độ chính xác điều khiển cho ứng dụng này. Hệ thống nguồn điện với bộ chuyển đổi buck đảm bảo hoạt động ổn định và thời gian sử dụng pin hợp lý.

5.2.2. Ý nghĩa thực tiễn

- **Giá trị giáo dục:** Dự án cung cấp nền tảng học tập xuất sắc cho sinh viên quan tâm đến hệ thống nhúng, lập trình RTOS, và robotics. Cấu trúc mã nguồn rõ ràng và có tài liệu đầy đủ giúp những người khác dễ dàng hiểu và sửa đổi cho các dự án riêng.

- **Ứng dụng thực tế:** Robot có thể được sử dụng cho nhiều mục đích thực tế như tự động hóa gia đình, giám sát từ xa, hoặc trình diễn giáo dục. Nền tảng này có thể phục vụ làm cơ sở cho các dự án nâng cao hơn như điều hướng tự động hoặc tích hợp cảm biến.

- **Đóng góp cộng đồng:** Tính chất mã nguồn mở của dự án giúp đóng góp trở lại cho cộng đồng maker. Tài liệu chi tiết và bài học kinh nghiệm có thể giúp những người khác tránh được các lỗi thường gặp và đẩy nhanh quá trình phát triển của họ.

5.2.3. Khả năng ứng dụng

- **Tiềm năng thương mại:** Với một số cải tiến, nền tảng này có thể được thương mại hóa cho thị trường giáo dục hoặc những người đam mê DIY. Chi phí thấp và dễ lắp ráp làm cho nó hấp dẫn đối với các trường học và trung tâm đào tạo.

- **Nền tảng nghiên cứu:** Hệ thống có thể phục vụ như bàn thử nghiệm cho các chủ đề nghiên cứu như điều khiển phân tán, giao thức giao tiếp không dây, hoặc thuật toán học máy. Kiến trúc modular tạo điều kiện thuận lợi cho việc tạo mẫu nhanh các ý tưởng mới.

- **Ứng dụng công nghiệp:** Các phiên bản mở rộng có thể tìm thấy ứng dụng trong tự động hóa kho bãi, quản lý hàng tồn kho, hoặc giám sát cơ sở. Các công nghệ cốt lõi đã được chứng minh có thể được điều chỉnh cho các yêu cầu hiệu suất cao hơn.

5.3. Hướng phát triển

5.3.1. Cải thiện hiện tại

- **Tích hợp cảm biến:** Ưu tiên cao cho việc thêm encoder bánh xe để cho phép điều khiển tốc độ vòng kín và phản hồi vị trí. Tích hợp IMU sẽ cung cấp thông tin định hướng và cho phép các thuật toán điều hướng tinh vi hơn. Cảm biến siêu âm hoặc lidar có thể thêm khả năng tránh vật cản.

- **Tối ưu hóa quản lý nguồn điện:** Triển khai các tính năng quản lý nguồn điện tiên tiến như điều chỉnh tần số động, chế độ ngủ khi rảnh rỗi, và giám sát pin thông minh. Thêm cảnh báo pin yếu và tự động tắt để bảo vệ pin khỏi xả sâu.

- **Cải tiến giao tiếp:** Xem xét các giao thức giao tiếp thay thế như Bluetooth hoặc module RF để mở rộng phạm vi và cải thiện độ tin cậy. Triển khai khả năng mạng lưới để cho phép phối hợp nhiều robot.

- **Cải tiến cơ khí:** Thiết kế lại khung với phân bố trọng lượng tốt hơn và độ bền được cải thiện. Thêm hệ thống treo để xử lý địa hình gồ ghề tốt hơn. Xem xét hệ thống di chuyển dựa trên xích để có lực bám tốt hơn.

5.3.2. Tính năng bổ sung

- **Điều hướng tự động:** Tích hợp khả năng SLAM (Định vị và Lập bản đồ đồng thời) sử dụng kết hợp cảm biến. Triển khai thuật toán lập kế hoạch đường đi để cho phép chuyển động tự động đến tọa độ được chỉ định. Thêm tính năng điều hướng waypoint cho các nhiệm vụ phức tạp.

- **Thị giác máy tính:** Thêm module camera với khả năng xử lý hình ảnh. Triển khai nhận dạng đối tượng, theo đường thẳng, hoặc theo dõi khuôn mặt. Xem xét xử lý AI cạnh với các bộ tăng tốc mạng thần kinh chuyên dụng.

- **Điều khiển giọng nói:** Tích hợp khả năng nhận dạng giọng nói để cho phép lệnh tọng nói. Thêm phản hồi văn bản thành giọng nói để cải thiện tương tác người dùng. Triển khai xử lý ngôn ngữ tự nhiên để điều khiển trực quan hơn.

- **Tích hợp IoT:** Kết nối robot với các nền tảng đám mây để giám sát và điều khiển từ xa qua internet. Triển khai khả năng ghi log dữ liệu và phân tích. Thêm tích hợp với các hệ thống nhà thông minh.

5.3.3. Ứng dụng mở rộng

- **Robotics giáo dục:** Phát triển các gói chương trình giảng dạy cho các cơ sở giáo dục. Tạo môi trường mô phỏng để bổ sung cho robot vật lý. Thêm giao diện lập trình để học sinh học lập trình robotics.

- **Tự động hóa gia đình:** Điều chỉnh nền tảng cho các ứng dụng giám sát gia đình với camera và cảm biến môi trường. Triển khai quy trình tuần tra bảo mật với phát hiện xâm nhập. Thêm tích hợp với các hệ sinh thái nhà thông minh hiện có.

- **Ứng dụng nghiên cứu:** Phát triển khả năng robotics đàn để phối hợp nhiều robot. Triển khai mạng cảm biến phân tán để giám sát môi trường. Tạo bàn thử nghiệm cho các thuật toán điều khiển và giao thức giao tiếp mới.

- **Tự động hóa công nghiệp:** Mở rộng quy mô thiết kế cho các ứng dụng kho bãi với khả năng tải trọng cao hơn. Thêm hệ thống định vị chính xác cho các tác vụ sản xuất. Triển khai khả năng quản lý đội robot cho việc triển khai nhiều robot.

TÀI LIỆU THAM KHẢO

[1] A. S. Tanenbaum and H. Bos, *Modern Operating Systems*, 4th ed. Boston, MA, USA: Pearson, 2014.

[2] J. Ganssle, *The Art of Designing Embedded Systems*, 2nd ed. Newnes, 2008.

[3] M. Labrosse, *µC/OS-III: The Real-Time Kernel*, 3rd ed. Newnes, 2009.

[4] FreeRTOS, "FreeRTOS Real Time Operating System (RTOS)," [Online]. Available: https://www.freertos.org/.

[5] Arduino, "Arduino UNO Rev3," [Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3.