

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH  
KHOA ĐIỆN - ĐIỆN TỬ  
BỘ MÔN ĐIỆN TỬ



THIẾT KẾ HỆ THỐNG NHÚNG  
BÁO CÁO BÀI TẬP LỚN

THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN PI  
ĐỘNG CƠ DC

GVHD: BÙI QUỐC BẢO

TN01. NHÓM 2

Nguyễn Đình Quyền 2312890

Đặng Hà Minh Tuấn 2213764

TPHCM, tháng 11 năm 2025

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA TP HỒ CHÍ MINH  
KHOA ĐIỆN - ĐIỆN TỬ  
BỘ MÔN ĐIỆN TỬ



THIẾT KẾ HỆ THỐNG NHÚNG  
BÁO CÁO BÀI TẬP LỚN

THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN PI  
ĐỘNG CƠ DC

Tên	MSSV	Phân công	Đánh giá
Nguyễn Đình Quyền	2312890	Phân tích yêu cầu hệ thống Thiết kế phần cứng Làm báo cáo	50%
Đặng Hà Minh Tuấn	2213764	Phân tích yêu cầu hệ thống Thiết kế firmware Thiết kế giao diện người dùng Chế tạo mẫu thử	50%

TPHCM, tháng 11 năm 2025

## TÓM TẮT

Đề tài xây dựng một hệ thống điều khiển tốc độ động cơ DC dựa trên thuật toán PID, kết hợp thiết kế phần cứng, firmware và giao diện giám sát trên máy tính. Hệ thống sử dụng vi điều khiển STM32F103C8T6 làm trung tâm xử lý, bộ cầu H L298N làm mạch công suất và encoder để đo phản hồi tốc độ theo thời gian thực. Dữ liệu vận tốc được gửi lên máy tính thông qua UART và hiển thị bằng phần mềm được xây dựng trên nền tảng .NET Framework. Toàn bộ hệ thống được triển khai trên PCB hai lớp với các khối chức năng được tách biệt nhằm hạn chế nhiễu và tăng độ ổn định. Kết quả đo đạc cho thấy hệ thống đạt được tốc độ đáp ứng nhanh, sai số nhỏ và vận hành ổn định trong điều kiện tải thực tế, đáp ứng đầy đủ các yêu cầu đã đặt ra.

# MỤC LỤC

1. GIỚI THIỆU .....	1
1.1. Lý do chọn đề tài .....	1
1.2. Ý nghĩa .....	1
1.3. Ứng dụng .....	1
1.4. Mục tiêu .....	2
2. Tổng quan lý thuyết .....	3
2.1. Tổng quan về hệ thống nhúng .....	3
2.2. Nguyên lý động cơ DC và điều khiển .....	4
2.3. Điều khiển vòng kín .....	4
2.4. Giao tiếp UART .....	5
2.5. Vai trò MCU trong hệ thống nhúng .....	5
2.6. Kiến trúc phần mềm .....	6
3. Yêu cầu hệ thống .....	9
3.1. Yêu cầu hệ thống .....	9
3.2. Yêu cầu phần cứng .....	11
3.3. Yêu cầu phần mềm .....	14
4. Thiết kế .....	16
4.1. Thiết kế Schematic .....	16
4.1.1. Sơ đồ khối .....	16
4.1.2. Nguồn cấp .....	17
4.1.3. MCU .....	19
4.1.4. Motor Diver .....	21
4.1.5. UART .....	22
4.1.6. LED và BUTTON .....	24
4.1.7. Tiểu kết Schematic .....	25
4.2. Thiết kế PCB .....	26
4.2.1. Bố trí linh kiện .....	26
4.2.2. Đặt xếp lớp (stack layer) và luật (rule) .....	28
4.2.3. Triển khai PCB .....	32
4.3. Thiết kế Firmware (Firmware design) .....	39
4.3.1. Sơ đồ trạng thái (Finite State Machine - FSM) .....	39
4.3.2. Vòng lặp điều khiển PID (PID Loop) .....	40

4.3.3. Lưu đồ giải thuật .....	40
4.3.4. Cấu hình Timer và PWM .....	41
4.4. Thiết kế Giao diện người dùng - phần mềm giám sát .....	42
4.4.1. Kiến trúc phần mềm .....	42
4.4.2. Thiết kế giao diện người dùng .....	42
4.4.3. Xử lý Đa luồng và Truyền thông .....	43
4.4.4. Định nghĩa Giao thức truyền thông .....	43
4.4.5. Lưu đồ giải thuật .....	45
4.5. Thiết kế mẫu thử và đo đặc .....	48
4.5.1. Cấu hình phần cứng mẫu thử .....	48
4.5.2. Quy trình đấu nối và sơ đồ lắp ráp .....	49
4.5.3. Phương pháp đo đặc và thu thập dữ liệu .....	50
5. Kết quả và thảo luận .....	52
5.1. Kết quả tổng quát .....	52
5.2. Hiệu suất động cơ (Open-Loop Test) .....	53
5.3. Kiểm tra hiệu suất thuật toán (Closed-Loop PI Test) .....	53
5.4. Các khó khăn và vấn đề gặp phải .....	54
5.5. Thảo luận .....	54
6. Kết Luận .....	55

## DANH SÁCH HÌNH ẢNH

Hình 2.1	Sơ đồ mạch cầu H .....	4
Hình 3.1	Kiến trúc hệ thống .....	10
Hình 4.1	Block diagram của hệ thống .....	16
Hình 4.2	Sơ đồ phân phối nguồn .....	17
Hình 4.3	Sơ đồ nguyên lý phần nguồn .....	18
Hình 4.4	Sơ đồ nguyên lý MCU .....	19
Hình 4.5	.....	21
Hình 4.6	Sơ đồ nguyên lý L298N .....	21
Hình 4.7	Sơ đồ nguyên lý UART .....	22
Hình 4.8	.....	24
Hình 4.9	Sơ đồ nguyên lý Button và Led .....	24
Hình 4.10	Sơ đồ phác thảo bố trí linh kiện .....	26
Hình 4.11	Đặt stack layer .....	28
Hình 4.12	Đặt Clearance rule .....	29
Hình 4.13	Đặt trace width rule .....	30
Hình 4.14	Đặt Routing Vias Rule .....	30
Hình 4.15	Đặt Routing Corners Rule .....	31
Hình 4.16	Bố cục PCB ở chế độ layout 2D .....	32
Hình 4.17	Đi dây Top Layer .....	33
Hình 4.18	Đi dây Bottom Layer .....	33
Hình 4.19	Bố cục 3D: Mặt trước .....	34
Hình 4.20	Bố cục 3D: Mặt sau .....	34
Hình 4.21	Bố cục 3D: Mặt nghiên .....	35
Hình 4.22	Bố cục 2D: Đi dây GND .....	36
Hình 4.23	Design Rule Verication .....	37
Hình 4.24	Lưu đồ giải thuật .....	45
Hình 4.25	Mẫu thử thực tế .....	49
Hình 4.26	Sơ đồ đấu nối mẫu thử .....	50
Hình 4.27	Giao diện người dùng .....	51

# GIỚI THIỆU

## 1.1. Lý do chọn đề tài

Động cơ điện một chiều (DC Motor) là thành phần chấp hành phổ biến nhất trong công nghiệp và dân dụng. Tuy nhiên, đặc tính tự nhiên của động cơ DC là tốc độ thay đổi theo tải trọng (open-loop). Để duy trì tốc độ ổn định bất chấp thay đổi tải, cần áp dụng các thuật toán điều khiển vòng kín (Closed-loop control).

Hạn chế của hệ điều khiển khác: Các bộ điều khiển công nghiệp (PLC, Biến tần chuyên dụng) thường có chi phí cao, cồng kềnh và khó tùy biến cho các ứng dụng nhỏ hoặc robot di động. Các mạch điều khiển rẻ tiền thì thường thiếu tính năng giám sát trực quan và độ chính xác thấp.

## 1.2. Ý nghĩa

Việc giải quyết bài toán điều khiển chính xác tốc độ động cơ là nền tảng cho các hệ thống tự động hóa phức tạp như robot tự hành (AGV), cánh tay robot, và băng chuyền sản xuất. Trong bối cảnh Công nghiệp 4.0, yêu cầu về việc thu thập dữ liệu (Telemetry) và điều khiển giám sát từ xa (SCADA) ngày càng cao. Đề tài này không chỉ dừng lại ở việc quay động cơ mà còn tích hợp hệ thống nhúng với phần mềm máy tính, phản ánh xu hướng IoT và Smart Factory.

## 1.3. Ứng dụng

Sản phẩm của đề tài có thể ứng dụng trong:

- Hệ thống băng tải thông minh (tự ổn định tốc độ khi có hàng hóa).
- Bánh xe robot dò line hoặc robot thám hiểm địa hình.
- Các hệ thống quạt tản nhiệt thông minh điều chỉnh theo nhiệt độ.
- Giáo cụ trực quan cho việc giảng dạy môn Lý thuyết điều khiển tự động.

## 1.4. Mục tiêu

Mục tiêu của đề tài là thiết kế và triển khai một hệ thống điều khiển tốc độ động cơ DC hoàn chỉnh, bao gồm phần cứng, phần mềm nhúng và phần mềm giám sát. Cụ thể:

1. Thiết kế hệ thống nhúng có khả năng điều khiển tốc độ và chiều quay của động cơ DC bằng phương pháp điều chế độ rộng xung (PWM).
2. Xây dựng bộ điều khiển vòng kín PI, cho phép duy trì tốc độ ổn định với sai số nhỏ và khả năng đáp ứng nhanh khi thay đổi tải hoặc tốc độ đặt.
3. Thiết kế mạch PCB tích hợp các khối: nguồn, vi điều khiển, driver động cơ, UART-USB và giao tiếp ngoại vi.
4. Phát triển firmware thực hiện đọc encoder, tính toán điều khiển, bảo vệ hệ thống và truyền thông UART.
5. Xây dựng giao diện người dùng (GUI) để nhập tốc độ đặt, theo dõi tốc độ thực và quan sát dữ liệu theo thời gian thực.
6. Đánh giá hệ thống thực tế, bao gồm độ ổn định, sai số, tốc độ đáp ứng và khả năng hoạt động lâu dài.

# Tổng quan lý thuyết

## 2.1. Tổng quan về hệ thống nhúng

Hệ thống nhúng (Embedded systems) được xác định là các hệ thống xử lý thông tin tương tác với các quá trình vật lý và được nhúng (embedded) vào một sản phẩm lớn hơn. Bản thân hệ thống nhúng là một thiết bị bao gồm một máy tính có thể lập trình được nhưng không phải là một máy tính đa năng. Theo một định nghĩa khác, một hệ thống nhúng là hệ thống có phần cứng máy tính với phần mềm được nhúng trong đó như một trong những thành phần quan trọng nhất, và thành phần quan trọng nhất của hệ thống nhúng là Bộ vi xử lý. Hệ thống nhúng được đặc trưng bởi các thuộc tính cơ bản như: chức năng đơn lẻ, chỉ thực thi một chương trình duy nhất và lặp lại,; bị ràng buộc chặt chẽ bởi các giới hạn như chi phí thấp, năng lượng thấp, kích thước nhỏ và tốc độ nhanh,; và có tính chất phản ứng và thời gian thực , liên tục phản ứng với thay đổi môi trường và phải tính toán kết quả trong thời gian thực mà không có sự chậm trễ,.

Về mặt kiến trúc, mô hình hệ thống nhúng được xác định thông qua quá trình phân chia hệ thống (system partitioning),. Quá trình này có thể phân chia hệ thống thành Phần cứng và Phần mềm,, hoặc chi tiết hơn là thành ba lớp (layer) cấu thành chính,,,:;

1. Lớp Phần cứng (Hardware layer): Cung cấp nền tảng vật lý cho hệ thống nhúng. Lớp này bao gồm các thành phần cốt lõi như Bộ vi xử lý (Processors), bộ nhớ (memory), các thiết bị ngoại vi (peripherals), thiết bị đầu vào/đầu ra (input/output devices), và PCB. Các thành phần khác thuộc lớp này còn có vi điều khiển (microcontrollers).
2. Lớp Phần mềm Hệ thống (System software layer): Đảm nhận vai trò cung cấp môi trường hoạt động và quản lý tài nguyên. Các thành phần chính của lớp này là Hệ điều hành (Operating system), trình điều khiển hệ thống (system drivers) và thư viện hệ thống (system library).
3. Lớp Phần mềm Ứng dụng (Application software layer): Là lớp trên cùng, chứa các chương trình được viết để thực hiện các mục đích chuyên biệt của sản phẩm. Các thành phần điển hình trong lớp này bao gồm chương trình ứng dụng (application programs), giao diện người dùng đồ họa (GUI) và các chương trình điều khiển.

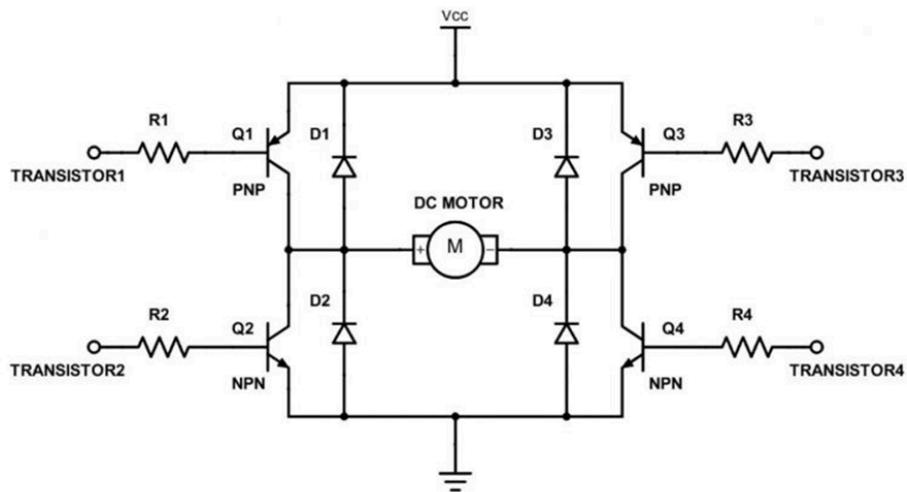
Việc phân chia hệ thống thành các lớp này là bước thiết yếu trong tiến trình thiết kế hệ thống nhúng.

## 2.2. Nguyên lý động cơ DC và điều khiển

Động cơ DC hoạt động dựa trên lực Lorentz. Khi cấp điện áp vào cuộn dây đặt trong từ trường, lực từ làm quay rotor. Tốc độ động cơ tỷ lệ thuận với điện áp cấp vào phần ứng (Armature Voltage). Thay vì thay đổi điện áp analog (khó thực hiện), ta sử dụng phương pháp điều chế độ rộng xung (PWM). Bằng cách đóng cắt nhanh nguồn điện với chu kỳ cố định, điện áp trung bình cấp cho động cơ thay đổi theo công thức:

$$V_{avg} = V_{source} \times \text{DutyCycle}$$

Để điều khiển chiều quay động cơ, ta cần đảo chiều dòng điện. Mạch cầu H sử dụng 4 khóa bán dẫn (Transistor/MOSFET) để thay đổi cực tính điện áp đặt lên động cơ.



Hình 2.1: Sơ đồ mạch cầu H

## 2.3. Điều khiển vòng kín

Hệ thống sử dụng bộ điều khiển PID (Proportional - Integral - Derivative) để tự động điều chỉnh PWM dựa trên sai số giữa tốc độ đặt (Setpoint) và tốc độ thực đo được từ Encoder.

- P: Tác động nhanh theo sai số hiện tại.
- I: Khử sai số xác lập (cộng dồn sai số quá khứ).
- D: Dự đoán xu hướng lỗi (giảm vọt lỗi).

Tuy nhiên trong project lần này nhóm sẽ chỉ triển khai bộ điều khiển PI.

## 2.4. Giao tiếp UART

Giao thức truyền thông nối tiếp không đồng bộ (UART – Universal Asynchronous Receiver/Transmitter) được sử dụng để trao đổi dữ liệu giữa vi điều khiển (MCU) và máy tính. UART hoạt động theo cơ chế truyền từng bit trên một đường truyền duy nhất, không sử dụng xung nhịp chung, vì vậy dữ liệu được đồng bộ dựa trên bit bắt đầu (Start bit) và bit kết thúc (Stop bit). Trong hệ thống, UART đảm nhiệm hai chức năng chính: gửi dữ liệu tốc độ thực từ MCU lên máy tính để giám sát và thu các lệnh điều khiển từ máy tính đưa xuống MCU.

Tốc độ Baud được cấu hình là 115200 baud, nghĩa là đường truyền có khả năng trao đổi 115200 bit dữ liệu mỗi giây. Đây là tốc độ cao, phù hợp cho các ứng dụng cần phản hồi nhanh và truyền dữ liệu liên tục, đảm bảo giao tiếp ổn định giữa thiết bị và máy tính.

## 2.5. Vai trò MCU trong hệ thống nhúng

Vi điều khiển (Microcontroller – MCU) là thành phần cốt lõi của hệ thống nhúng và thuộc lớp phần cứng trong kiến trúc tổng thể của hệ thống. Khác với máy tính đa năng, MCU được thiết kế chuyên dụng để thực hiện các nhiệm vụ điều khiển trong thời gian thực. Trong một hệ thống nhúng, MCU đóng vai trò như bộ xử lý trung tâm, chịu trách nhiệm điều khiển các thiết bị đầu vào, cảm biến, cơ cấu chấp hành và các bộ phận hiển thị; đồng thời xử lý những tác vụ, thuật toán và thực hiện giao tiếp với các hệ thống hoặc thiết bị bên ngoài. Các dòng vi điều khiển phổ biến bao gồm 8051, PIC16F, ARM Cortex-M và nhiều họ MCU khác được sử dụng rộng rãi trong công nghiệp và sản phẩm dân dụng.

Về mặt kiến trúc, MCU được phân loại theo chiều rộng xử lý của ALU, chẳng hạn như vi điều khiển 8-bit, 16-bit hoặc 32-bit. Sự khác biệt chủ yếu nằm ở khả năng thực thi các phép toán số học và logic: ALU có thể xử lý dữ liệu 8-bit, 16-bit hoặc 32-bit chỉ trong một câu lệnh, từ đó ảnh hưởng trực tiếp đến tốc độ xử lý, độ chính xác và mức độ phức tạp của ứng dụng. Công tác lựa chọn MCU vì vậy đóng vai trò quan trọng trong giai đoạn thiết kế phần cứng của hệ thống nhúng. Việc lựa chọn thường bắt đầu bằng việc xác định các giao diện phần cứng cần thiết như GPIO, ADC, DAC hoặc các ngoại vi giao tiếp. Tiếp theo là đánh giá yêu cầu của phần mềm, bao gồm thuật toán xử lý, nhu cầu tính toán dấu phẩy động, sự cần thiết của phần cứng hỗ trợ như FPU hay DMA, cùng các yếu tố liên quan đến cơ chế ngắn và bộ định thời. Sau khi xác định

được nhu cầu phần cứng và phần mềm, kiến trúc phù hợp — 8/16-bit hoặc 32-bit — sẽ được lựa chọn, đồng thời kiểm tra khả năng hỗ trợ thư viện và hệ sinh thái phát triển của kiến trúc đó.

Bên cạnh đó, việc ước tính dung lượng bộ nhớ chương trình và bộ nhớ dữ liệu là yếu tố không thể thiếu, bởi dung lượng này phải đủ để chứa cấu trúc dữ liệu, thuật toán và hệ điều hành hoặc thư viện đi kèm. Các tiêu chí thực tế như chi phí linh kiện, khả năng tiêu thụ năng lượng thấp (low-power) và khả năng cung ứng linh kiện cũng là những yếu tố quan trọng, đặc biệt đối với các thiết kế sản xuất số lượng lớn hoặc các thiết bị chạy bằng pin. MCU giữ vai trò trung tâm trong quá trình phân chia hệ thống (system partitioning), nơi các chức năng được tách thành phần cứng và phần mềm nhằm đảm bảo toàn bộ hệ thống hoạt động ổn định, tối ưu và đáp ứng đúng yêu cầu của ứng dụng nhúng.

## 2.6. Kiến trúc phần mềm

Hệ thống phần mềm nhúng trên vi điều khiển STM32 được thiết kế dựa trên mô hình Kiến trúc Tiền nền - Hậu nền (Foreground-Background Architecture), hay còn gọi là mô hình Siêu vòng lặp kết hợp Ngắt (Super Loop with Interrupts). Kiến trúc này đảm bảo sự cân bằng giữa khả năng đáp ứng thời gian thực (Real-time response) cho các tác vụ quan trọng và khả năng xử lý logic phức tạp.

Hệ thống được chia làm hai tầng xử lý chính:

**a. Tầng xử lý ngắt (Foreground - Interrupt Service Routines):** Đây là tầng có độ ưu tiên cao nhất, chịu trách nhiệm xử lý các sự kiện cần tính tức thời và định thời chính xác. Các tác vụ trong tầng này bao gồm:

- Ngắt Timer (Timer Interrupt): Được cấu hình định kỳ mỗi 100ms (Sampling time). Nhiệm vụ của ngắt này là kích hoạt cờ báo (Flag) để báo hiệu cho vòng lặp chính thực hiện lấy mẫu vận tốc và tính toán PID. Việc sử dụng ngắt Timer giúp chu kỳ lấy mẫu Ts luôn cố định, đảm bảo tính chính xác cho các phép toán tích phân và đạo hàm trong bộ điều khiển.
- Ngắt ngoài (External Interrupt - EXTI): Dùng để bắt sườn lên/xuống của tín hiệu xung từ Encoder. Mỗi khi có ngắt, biến đếm xung (pulse\_count) sẽ được tăng lên. Đây là phương pháp đếm xung không đồng bộ giúp không bỏ sót tín hiệu từ động cơ quay nhanh.

- Ngắt UART (UART Receive Interrupt): Xử lý việc nhận từng byte dữ liệu từ máy tính và lưu vào bộ đệm vòng (Ring buffer) hoặc bộ đệm tuyến tính, đảm bảo dữ liệu truyền xuống không bị mất mát khi MCU đang bận xử lý tác vụ khác.

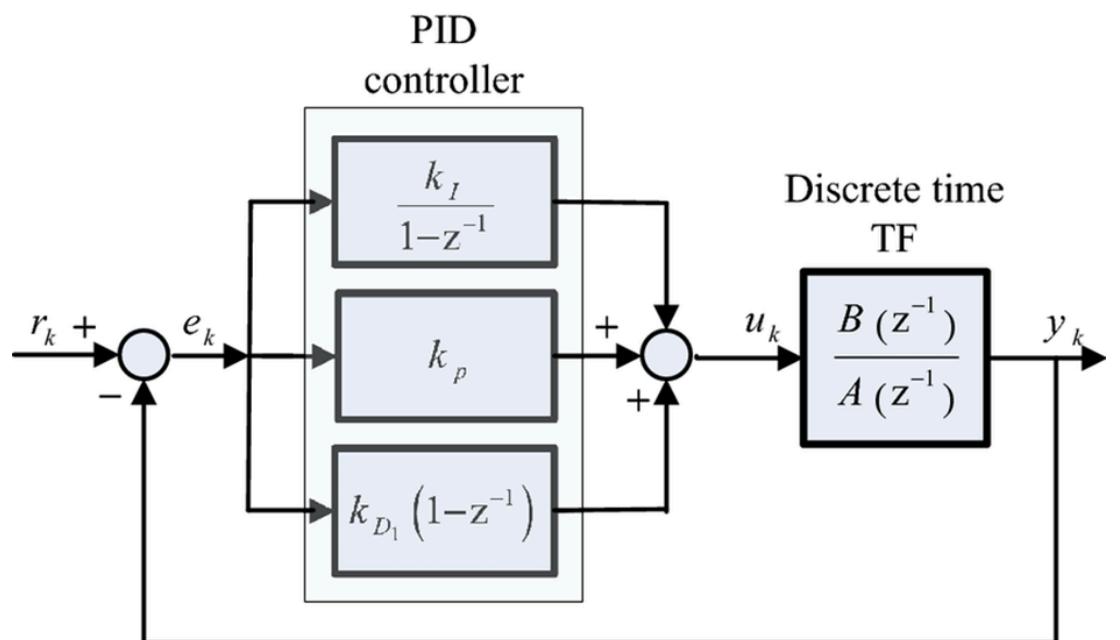
### b. Tầng vòng lặp chính (Background - Super Loop)

Đây là tầng xử lý các tác vụ có độ ưu tiên thấp hơn, diễn ra liên tục trong hàm main(). Các tác vụ được thực hiện tuần tự theo máy trạng thái:

Xử lý máy trạng thái (State Machine): Quản lý trạng thái hoạt động của hệ thống (IDLE, RUNNING, BRAKING). Dựa trên trạng thái hiện tại và tín hiệu điều khiển, hệ thống sẽ quyết định hành vi tiếp theo (ví dụ: đang chạy thì gấp lệnh dừng -> chuyển sang trạng thái hãm phanh).

Tính toán điều khiển (Control Loop): Khi nhận được cờ báo từ ngắt Timer (100ms), vòng lặp chính sẽ thực hiện:

1. Tính vận tốc thực tế (RPM) từ số xung đếm được.
2. So sánh với vận tốc đặt (Setpoint) để tìm sai số  $e(t)$ .
3. Thực thi thuật toán PID rồi rạc:



4. Quy đổi kết quả ra độ rộng xung PWM và cập nhật thanh ghi Timer.

Xử lý truyền thông: Phân tích chuỗi lệnh nhận được từ bộ đệm UART (Parsing) để cập nhật tham số (Kp, Ki, Setpoint) và đóng gói dữ liệu vận tốc hiện tại để gửi ngược lên máy tính (Telemetry).

**c. Cơ chế đồng bộ dữ liệu:**

Để đảm bảo toàn vẹn dữ liệu giữa tầng Ngắt và vòng lặp chính, các biến chia sẻ (Shared Variables) như pulse\_count, timer\_flag được khai báo với từ khóa volatile. Điều này ngăn trình biên dịch tối ưu hóa quá mức, đảm bảo CPU luôn đọc giá trị mới nhất từ bộ nhớ RAM thay vì từ thanh ghi đệm.

## Yêu cầu hệ thống

### 3.1. Yêu cầu hệ thống

**SYS-1: Hệ thống phải điều khiển được tốc độ và chiều quay của động cơ DC.**

SYS-1.1: Hệ thống phải có khả năng đo tốc độ động cơ.

SYS-1.2: Hệ thống phải có khả năng thay đổi công suất cấp để điều chỉnh tốc độ.

SYS-1.3: Hệ thống phải có khả năng đảo chiều quay theo lệnh điều khiển.

**SYS-2: Hệ thống phải cho phép người dùng nhập tốc độ mong muốn và quan sát tốc độ thực tế.**

SYS-2.1: Giao diện phải cho phép nhập giá trị tốc độ mục tiêu từ 0% set point cho tới 100% set point

SYS-2.2: Giao diện phải hiển thị tốc độ thực tế của động cơ.

SYS-2.3: Giao diện phải có khả năng kết nối với thiết bị giám sát bên ngoài, thông qua giao tiếp UART, tốc độ baud 115200

**SYS-3: Hệ thống phải duy trì tốc độ trong giới hạn sai số cho phép và phản ứng nhanh khi thay đổi tốc độ đặt.**

SYS-3.1: Sai số xác lập không vượt quá  $\pm 5\%$  so với tốc độ đặt.

SYS-3.2: Thời gian đáp ứng phải nhỏ hơn hoặc bằng 200 ms khi thay đổi tốc độ đặt.

**SYS-4: Hệ thống phải sẵn sàng hoạt động trong vòng 3 giây sau khi bật nguồn.**

SYS-4.1: Sau khi cấp nguồn, hệ thống phải vào trạng thái sẵn sàng điều khiển trong  $\leq 3$  giây.

**SYS-5: Hệ thống phải hoạt động liên tục tối thiểu 8 giờ mà không xảy ra lỗi hoặc reset ngoài ý muốn.**

SYS-5.1: Trong suốt thời gian hoạt động, hệ thống không được tự khởi động lại ngoài ý muốn.

**SYS-6: Hệ thống phải có cơ chế bảo vệ khi xảy ra sự cố (quá dòng, mất tín hiệu, lỗi phản hồi).**

SYS-6.1: Hệ thống phải phát hiện tình trạng quá dòng.

SYS-6.2: Hệ thống phải phát hiện tình trạng mất tín hiệu phản hồi.

SYS-6.3: Khi xảy ra sự cố, hệ thống phải ngắt tín hiệu điều khiển để bảo vệ phần cứng.

SYS-6.4: Khi lỗi nghiêm trọng xảy ra, hệ thống phải ngắt nguồn cấp cho động cơ ngay lập tức.

**SYS-7: Hệ thống phải được thiết kế để hạn chế nhiễu điện giữa phần công suất và mạch điều khiển.**

SYS-7.1: Đường nguồn động cơ phải được tách biệt với đường tín hiệu điều khiển.

SYS-7.2: Mạch phải có biện pháp lọc nhiễu phù hợp.

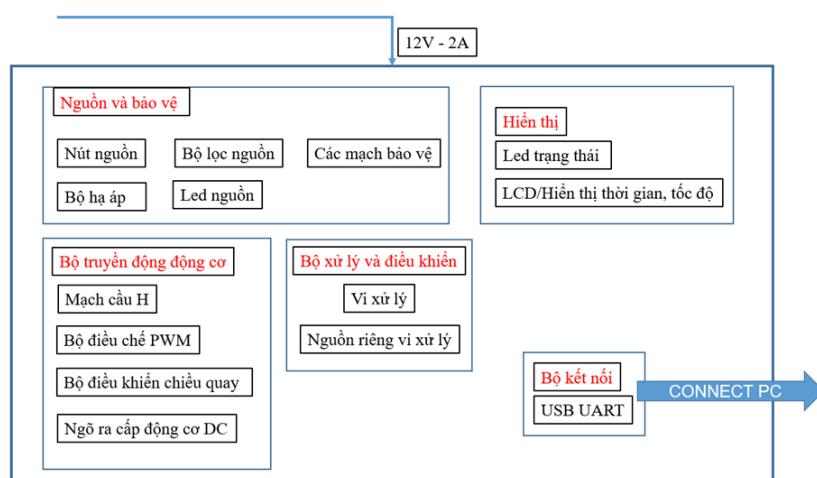
**SYS-8: Hệ thống phải hoạt động ổn định trong điều kiện thông thường.**

SYS-8.1: Hệ thống phải vận hành với điện áp ổn định 12V DC.

SYS-8.2: Dòng tiêu thụ tối đa không vượt quá 3A.

SYS-8.3: Hệ thống phải hoạt động ổn định trong nhiệt độ môi trường từ 25 độ C đến 50 độ C.

### Kiến trúc hệ thống



Hình 3.1: Kiến trúc hệ thống

### 3.2. Yêu cầu phần cứng

#### 1. Nguồn và Bảo vệ

HR-1.1: Hệ thống phải được cấp nguồn từ pin DC 12V, dung lượng đủ để vận hành liên tục  $\geq 8$  giờ.

HR-1.2: Bộ ổn áp tuyến tính hoặc xung phải cung cấp điện áp ổn định 5V/3.3V cho vi điều khiển và mạch điều khiển, sai số không vượt quá  $\pm 2\%$ .

HR-1.3: Mạch bảo vệ quá dòng phải được tích hợp, giới hạn dòng tối đa 3A, đảm bảo ngắt tải khi vượt ngưỡng trong  $\leq 10$  ms.

HR-1.4: Công tắc nguồn động cơ phải cho phép đóng/ngắt độc lập với mạch điều khiển.

HR-1.5: Mạch lọc nhiễu EMI phải được thiết kế để giảm ảnh hưởng xuyên nhiễu giữa phần công suất và tín hiệu điều khiển, với bộ lọc LC hoặc RC chuyên dụng.

HR-1.6: Hệ thống phải hoạt động ổn định trong dải nhiệt độ môi trường từ 25 °C đến 50 °C, không làm suy giảm chức năng điều khiển hoặc gây reset ngoài ý muốn.

#### 2. Truyền động Động cơ

HR-2.1: Mạch cầu H (H-Bridge) phải hỗ trợ điều khiển động cơ DC một chiều, công suất định mức  $\geq 36W$  (12V–3A).

HR-2.2: Bộ điều chế PWM phải hoạt động ở tần số từ 10 kHz – 20 kHz để hạn chế nhiễu âm thanh, với độ phân giải điều chế  $\geq 8$  bit.

HR-2.3: Bộ điều khiển chiều quay phải cho phép đảo chiều động cơ trong thời gian  $\leq 100$  ms sau khi nhận lệnh.

HR-2.4: Ngõ ra cấp cho động cơ phải có khả năng chịu tải liên tục 3A và xung khởi động tối đa 5A trong 1 giây.

#### 3. Xử lý và Điều khiển

HR-3.1: Vi điều khiển trung tâm phải có bộ tạo PWM tích hợp, tối thiểu 2 kênh độc lập.

HR-3.2: Bộ xử lý phải hỗ trợ điều khiển thời gian thực, đảm bảo chu kỳ xử lý  $\leq 1$  ms.

HR-3.3: Hệ thống phải khởi động và sẵn sàng trong vòng  $\leq 3$  giây sau khi cấp nguồn.

HR-3.4: Mạch phát hiện bắt thường phải giám sát tín hiệu dòng, điện áp, và phản hồi từ encoder/feedback.

HR-3.5: Khi xảy ra lỗi nghiêm trọng, phần điều khiển phải ngay lập tức gửi tín hiệu ngắt đến mạch công suất để bảo vệ phần cứng.

#### **4. Truyền thông UART**

HR-4.1: Khối UART-USB phải hỗ trợ tốc độ baud chuẩn 115200, 8-N-1.

HR-4.2: Hệ thống phải truyền/nhận dữ liệu trạng thái và lệnh điều khiển qua UART với độ tin cậy  $\geq 99.9\%$ .

HR-4.3: Khung truyền dữ liệu phải tuân theo định dạng: [Header]–[Command/Data]–[Checksum], đảm bảo phát hiện lỗi truyền.

HR-4.4: Thời gian phản hồi cho một lệnh UART không vượt quá 50 ms trong điều kiện hoạt động bình thường.

## Hardware consideration

Component	Quantity	MCU	Interface	Part	Note	
					Interface	Qty
					Number	
(1) STM32	1		PWM, ADC,	10+	STM32	
F103C8T6			UART,		F103C8T6	
(MCU)			GPIO			
(2) L298N	1		PWM,	4	L298N	Điều khiển
(Motor			GPIO		Module	tốc độ /
Driver)						chiều động
						cơ DC
(3) DC	1		–	–	DC12V	Động cơ DC
Motor					Motor	12V, dòng ≤
						3A
(4) Rotary	1		Timer/	2	Incremental	Cảm biến
Encoder			Encoder		Encoder	tốc độ và vị
			GPIO			trí trực
(5) Power	1		–	–	12V 3A PSU	Cấp nguồn
Supply 12V						cho driver và
DC						động cơ
(6) Push	2		GPIO	2	Generic	Nhập tốc
Button					Button	độ hoặc đổi
						chiều
(7) UART	1		UART	1	USB-TTL /	Kết nối PC,
Connector					3-pin header	baud 115200
(TTL/USB)						
(8) LED	3		GPIO	3	3mm LED	Hiển thị
						trạng thái và
						lỗi

Bảng 3.1: Hardware consideration

### 3.3. Yêu cầu phần mềm

Để đảm bảo khả năng vận hành chính xác và tính tương tác cao của hệ thống, các yêu cầu đối với phần mềm được chia thành hai mảng riêng biệt: phần mềm nhúng (Firmware) và phần mềm ứng dụng trên máy tính (Host Software).

#### a. Phần mềm nhúng (Firmware) trên Vi điều khiển:

- Môi trường phát triển tích hợp (IDE): Sử dụng Keil uVision 5 hoặc STM32CubeIDE. Đây là các công cụ tiêu chuẩn trong công nghiệp, cung cấp trình biên dịch tối ưu (Compiler optimization), công cụ nạp (Flasher) và trình gỡ lỗi (Debugger) thời gian thực.
- Ngôn ngữ lập trình: Sử dụng ngôn ngữ Embedded C (theo chuẩn C99). Ngôn ngữ C được lựa chọn nhờ khả năng truy cập phần cứng mức thấp, tốc độ thực thi nhanh và kiểm soát bộ nhớ chặt chẽ, phù hợp với tài nguyên hạn chế của vi điều khiển.
- Thư viện phần cứng: Sử dụng thư viện STM32 HAL (Hardware Abstraction Layer). Việc sử dụng HAL giúp trừu tượng hóa các thao tác thanh ghi phức tạp, tăng tính di động (portability) của mã nguồn và giảm thời gian phát triển, đồng thời đảm bảo mã nguồn tuân thủ các chuẩn lập trình của nhà sản xuất STMicroelectronics.
- Yêu cầu chức năng: Firmware phải đảm bảo tính đơn định (determinism) và đáp ứng thời gian thực (real-time) thông qua cơ chế ngắt (Interrupt). Cụ thể, các module Timer, UART, GPIO phải được cấu hình và khởi tạo chính xác để thực hiện chu trình điều khiển vòng kín.

#### b. Phần mềm Giám sát và Điều khiển (GUI):

- Nền tảng phát triển: Xây dựng trên nền tảng Microsoft .NET Framework bằng môi trường Visual Studio. Ứng dụng thuộc dạng Windows Forms Application để đảm bảo tính tương thích cao với hệ điều hành Windows.
- Ngôn ngữ lập trình: Sử dụng C Sharp. Đây là ngôn ngữ hướng đối tượng mạnh mẽ, hỗ trợ tốt cho việc thiết kế giao diện người dùng (UI/UX) và xử lý các sự kiện bất đồng bộ (Asynchronous events).
- Thư viện đồ họa: Tích hợp thư viện mã nguồn mở ZedGraph. Thư viện này cung cấp khả năng vẽ đồ thị hiệu năng cao, cho phép hiển thị dữ liệu đáp ứng

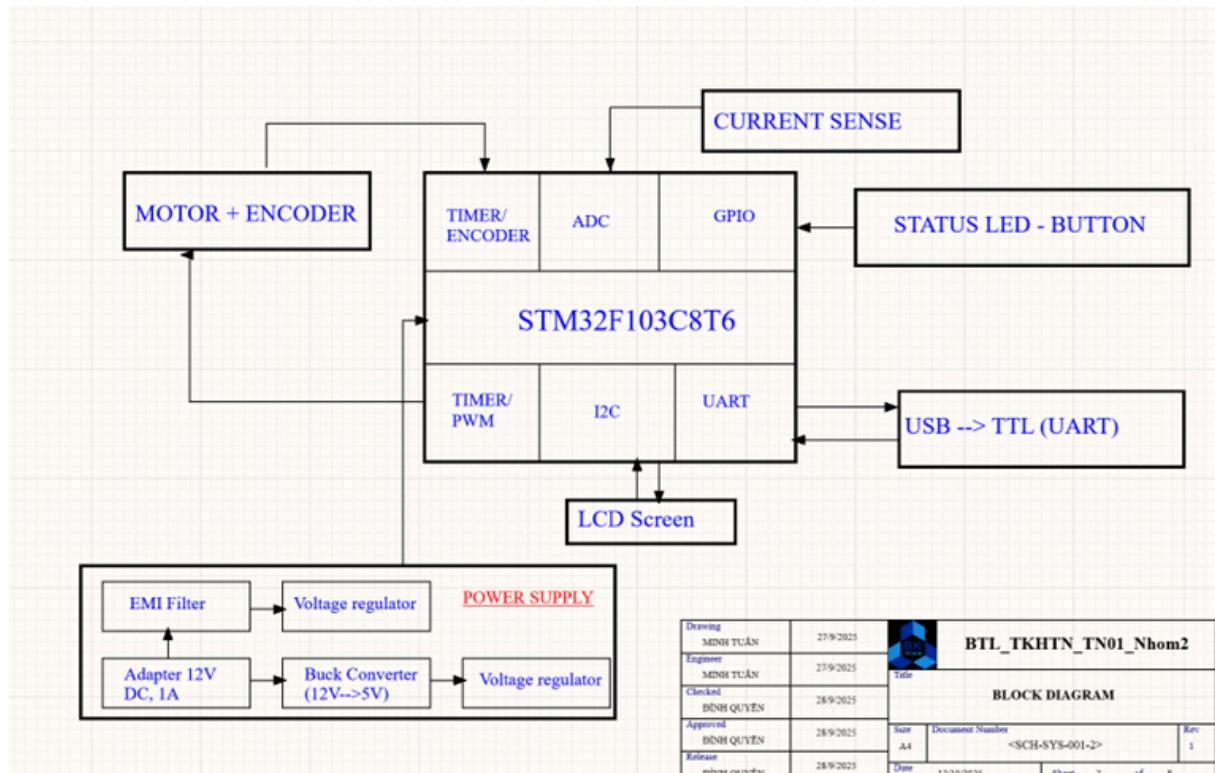
tốc độ theo thời gian thực (Real-time plotting) với độ trễ thấp, hỗ trợ các tính năng như Zoom, Pan và Auto-scaling trực tọa độ.

- Giao thức truyền thông: Phần mềm phải quản lý đối tượng SerialPort để thiết lập kênh truyền thông nối tiếp tin cậy với vi điều khiển. Cần triển khai cơ chế xử lý đa luồng (Multi-threading) hoặc Invoke để việc nhận dữ liệu từ cổng COM không làm “đóng băng” giao diện người dùng.

# Thiết kế

## 4.1. Thiết kế Schematic

### 4.1.1. Sơ đồ khối



Hình 4.1: Block diagram của hệ thống

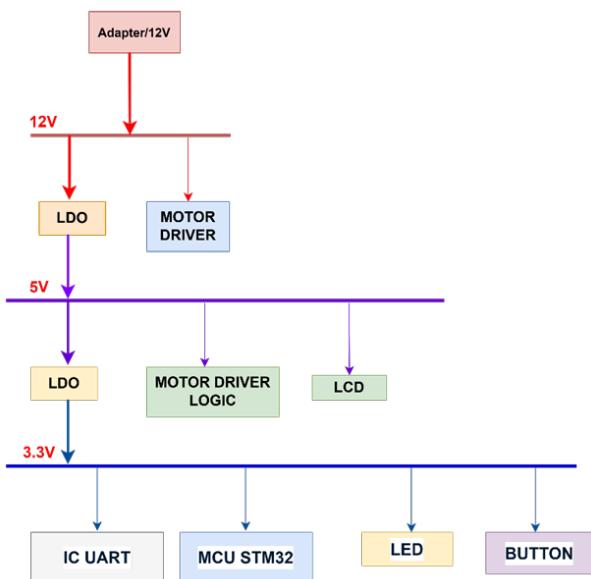
Sơ đồ khối của hệ thống cho thấy vi điều khiển STM32F103C8T6 đóng vai trò trung tâm, thực hiện toàn bộ tác vụ điều khiển, đo lường và giao tiếp. Động cơ đi kèm bộ mã hóa (encoder) được kết nối trực tiếp vào các timer/encoder của MCU, cho phép đo tốc độ và vị trí theo thời gian thực. Tín hiệu điều khiển động cơ được tạo ra thông qua các kênh PWM của timer, đảm bảo điều khiển đóng/mở công suất chính xác. Hệ thống cũng tích hợp mạch current sense, đưa tín hiệu dòng điện về ADC của MCU để giám sát tải và bảo vệ động cơ.

Các thành phần giao tiếp phụ trợ gồm LED trạng thái và nút nhấn, kết nối thông qua các chân GPIO để hiển thị và nhận lệnh người dùng. Giao tiếp với máy tính được thực hiện qua USB-TTL sử dụng UART, giúp nạp lệnh, kiểm tra dữ liệu hoặc truyền nhật ký hoạt động của hệ thống. Ngoài ra, một màn hình LCD được nối bằng giao tiếp

I2C, cho phép hiển thị các thông số quan trọng như tốc độ, dòng điện hay trạng thái hệ thống.

Khối nguồn bao gồm bộ adapter 12V, đi qua EMI filter để giảm nhiễu, sau đó được hạ áp bằng buck converter xuống 5V, tiếp tục qua các voltage regulator để tạo ra các mức 5V và 3.3V ổn định cho từng khối mạch. Cách bố trí nguồn theo tầng giúp đảm bảo toàn bộ hệ thống hoạt động ổn định, cách ly nhiễu tốt và phù hợp với yêu cầu của cả động cơ lẫn vi điều khiển.

#### 4.1.2. Nguồn cấp

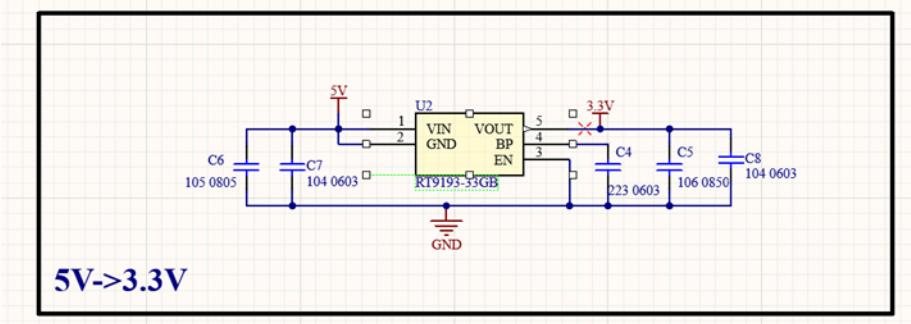
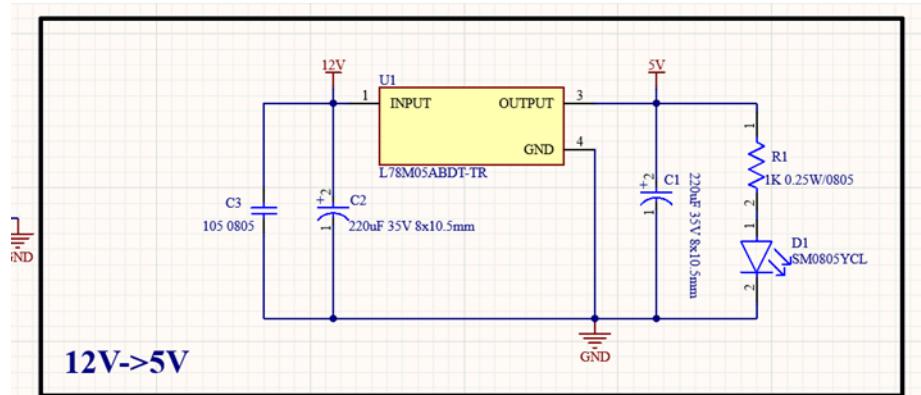


Hình 4.2: Sơ đồ phân phối nguồn

Nguồn 12V được cung cấp từ adapter thông qua terminal và đóng vai trò là nguồn công suất chính cho mạch điều khiển động cơ (Motor Driver). Điện áp này đồng thời được sử dụng làm đầu vào cho các tầng ổn áp tuyến tính.

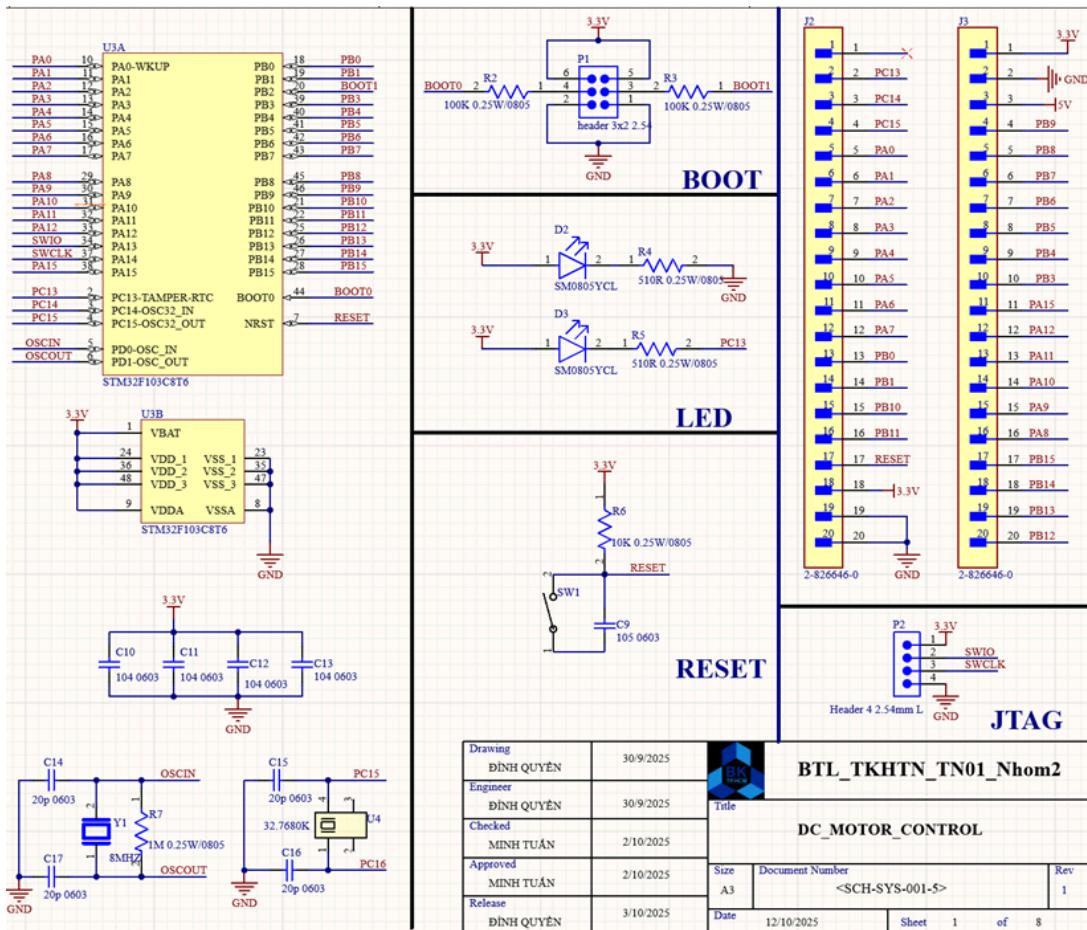
Đầu tiên, điện áp 12V được hạ xuống 5V nhờ bộ ổn áp tuyến tính L7805. Mức 5V này dùng để cấp nguồn cho khối logic của mạch lái công suất và màn hình LCD. Tiếp theo, điện áp 5V tiếp tục được hạ xuống 3.3V thông qua bộ ổn áp RT9193, một LDO có đặc tính nhiễu thấp, nhằm cung cấp nguồn cho vi điều khiển (MCU), IC UART, cũng như các led và nút nhấn.

Cấu trúc phân tầng như trên giúp đảm bảo từng khối mạch nhận được mức điện áp phù hợp và ổn định, đồng thời hạn chế nhiễu lan truyền giữa các phần công suất và phần điều khiển.



Hình 4.3: Sơ đồ nguyên lý phần nguồn

### 4.1.3. MCU



Hình 4.4: Sơ đồ nguyên lý MCU

Khối MCU sử dụng vi điều khiển STM32F103C8T6, đóng vai trò là trung tâm xử lý và điều khiển toàn bộ hệ thống. MCU được cấp nguồn 3.3 V từ LDO RT9193 để đảm bảo hoạt động ổn định, độ nhiễu thấp và phù hợp với mức điện áp hoạt động của dòng STM32.

Để hỗ trợ lập trình và debug, mạch tích hợp header 4 chân JTAG/SWD, cho phép nạp chương trình trực tiếp vào MCU qua SWCLK, SWDIO, GND và 3.3 V. Đồng thời, hệ thống bố trí một LED nguồn 3.3 V (kéo từ 3.3 V xuống GND) để hiển thị trạng thái có nguồn, giúp người vận hành dễ dàng kiểm tra tình trạng cấp nguồn của khối MCU.

Về mặt thời gian thực, MCU sử dụng hai khối dao động riêng biệt:

- Thạch anh 8 MHz cho hệ thống chính (System Clock), phục vụ xử lý và giao tiếp tốc độ cao.

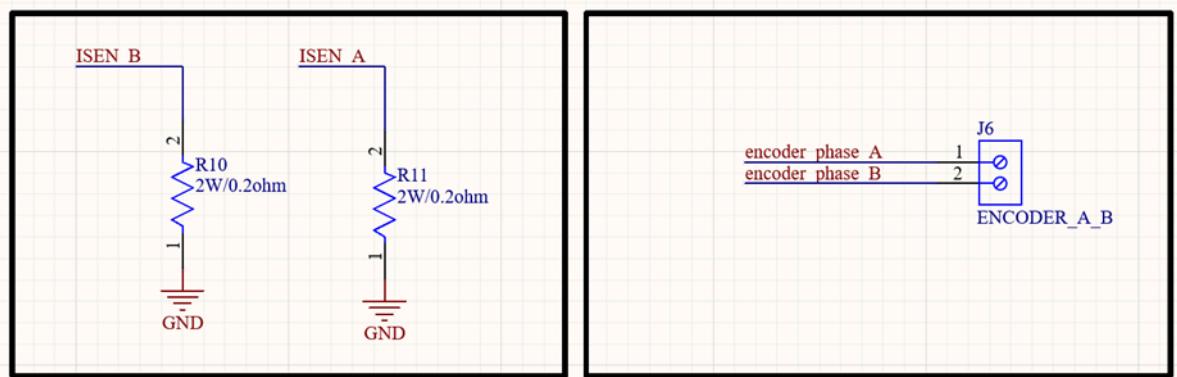
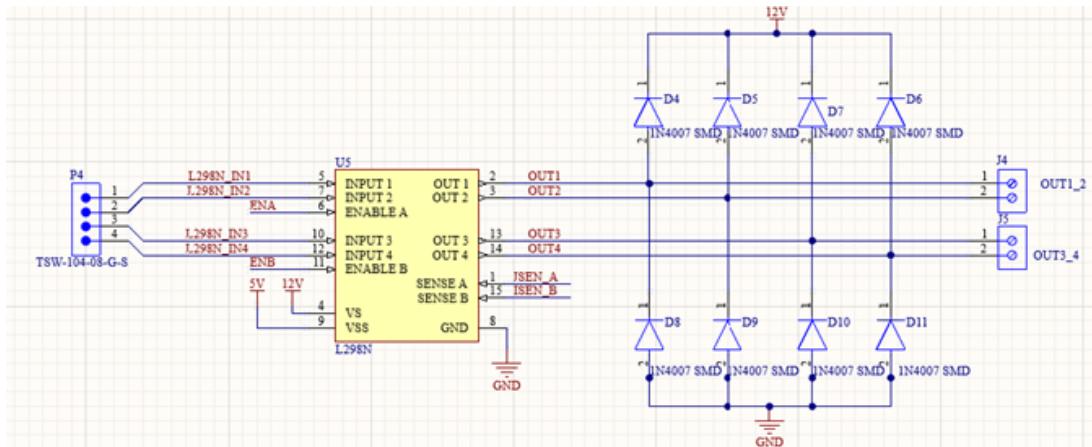
- Thạch anh 32.768 kHz cho RTC, đảm bảo độ chính xác khi cần đo thời gian thực hoặc vận hành chế độ tiết kiệm năng lượng.

Ngoài ra, hệ thống có một LED xuất (LED trạng thái) điều khiển trực tiếp từ chân MCU, dùng để báo hiệu hoạt động hoặc phục vụ debug trong quá trình thử nghiệm. Toàn bộ các chân của vi điều khiển được đưa ra header 20x2 nhằm hỗ trợ phát triển, thử nghiệm hoặc mở rộng chức năng sau này. Cấu trúc đưa ra toàn bộ chân giúp dễ dàng kết nối với module cảm biến, ngoại vi hoặc các khối mở rộng khác mà không cần thay đổi thiết kế phần cứng.

Mạch còn bố trí nút nhấn Reset nhằm khởi động lại MCU khi cần thiết trong quá trình lập trình hoặc chạy thử. Hai chân BOOT0 và BOOT1 được đưa ra cùng jump hoặc header tùy chỉnh, cho phép lựa chọn chế độ boot (Flash, RAM, hoặc ISP) sau khi reset, thuận tiện cho việc nạp chương trình qua UART hoặc cấu hình đặc biệt.

Toàn bộ khối MCU được bố trí tách biệt với khu vực công suất nhằm giảm ảnh hưởng nhiều từ motor và driver. Các tụ lọc nhiễu được đặt gần từng chân nguồn của MCU theo đúng khuyến nghị từ datasheet để đảm bảo độ ổn định, hạn chế nhiễu và tăng độ tin cậy của hệ thống.

#### 4.1.4. Motor Diver



Hình 4.6: Sơ đồ nguyên lý L298N

Khối điều khiển động cơ sử dụng IC L298N, một bộ cầu H kép cho phép điều khiển hai động cơ DC một cách độc lập. IC được cấp nguồn công suất từ đường 12 V, và nguồn logic 5 V từ LDO 7805 của hệ thống, đảm bảo mức điều khiển ổn định và tương thích với tín hiệu từ MCU.

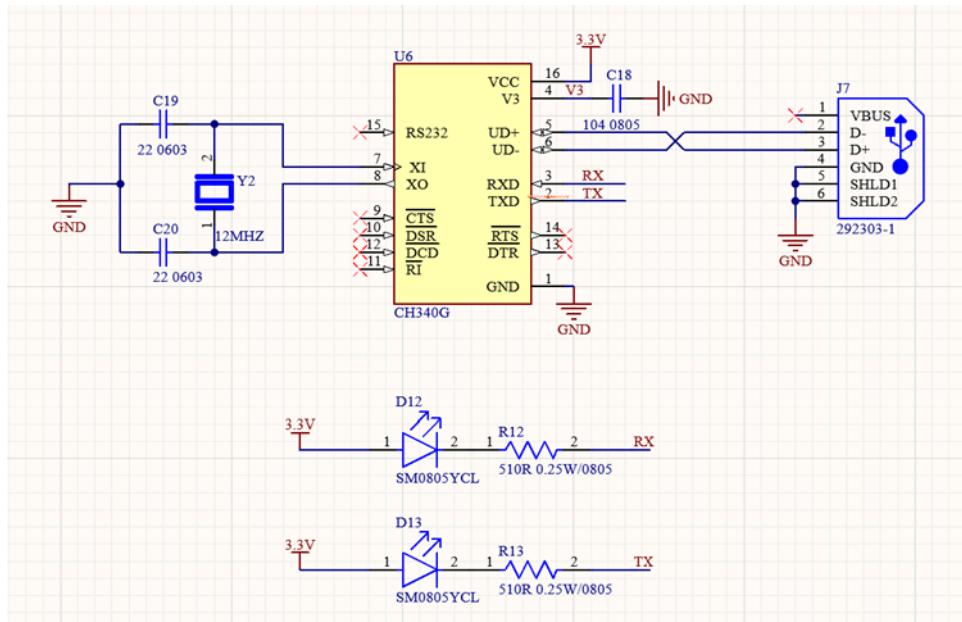
Các chân điều khiển IN1-IN4 của L298N được đưa ra cả MCU và đồng thời được bố trí thêm header test. Việc đưa các chân điều khiển ra header giúp dễ dàng kiểm tra, mô phỏng, hoặc thay đổi kết nối trong quá trình phát triển phần cứng mà không cần chỉnh sửa mạch in.

Để hỗ trợ giám sát dòng tải, hai chân ISEN A và ISEN B của L298N được mắc qua điện trở shunt  $0.2\ \Omega$ . Tín hiệu điện áp sụt trên shunt có thể được kéo về chân ADC của MCU, cho phép đo dòng tiêu thụ của động cơ theo thời gian thực. Thông tin này được dùng để:

- Phát hiện trạng thái tải (kết nối, tăng tải, không tải)
    - Kích hoạt bảo vệ quá dòng
    - Hoặc phục vụ thuật toán điều khiển thông minh.

Ngoài điều khiển công suất, mạch còn tích hợp terminal đầu nối encoder, cho phép kết nối trực tiếp với encoder gắn trên động cơ. Tín hiệu encoder (A, B) được đưa về MCU để đo tốc độ hoặc góc quay, qua đó hỗ trợ triển khai thuật toán điều khiển PID chính xác hơn, đặc biệt trong các ứng dụng robot di chuyển hoặc điều tốc vòng kín. Để giảm nhiễu do chuyển mạch dòng lớn, mạch sử dụng diode hồi dòng theo đúng khuyến nghị của datasheet và bố trí tụ lọc gần chân Vs để hạn chế sụt áp khi động cơ khởi động. Khối L298N được đặt cách biệt với khối MCU nhằm giảm nhiễu điện từ và đảm bảo hoạt động ổn định của toàn hệ thống. Thiết kế này giúp L298N không chỉ điều khiển tốc độ và hướng quay của động cơ mà còn cho phép giám sát và phản hồi dòng, kết hợp với encoder để tạo thành một hệ thống điều khiển vòng kín tin cậy và hiệu quả.

#### 4.1.5. UART



Hình 4.7: Sơ đồ nguyên lý UART

Khối giao tiếp sử dụng IC CH340G nhằm chuyển đổi tín hiệu UART của MCU sang chuẩn USB để có thể kết nối với máy tính phục vụ mục đích giao tiếp dữ liệu, giám sát trạng thái hoặc debug trong quá trình vận hành. IC CH340G được cấp nguồn 3.3 V từ LDO của hệ thống, đảm bảo tương thích mức logic với MCU STM32F103C8T6.

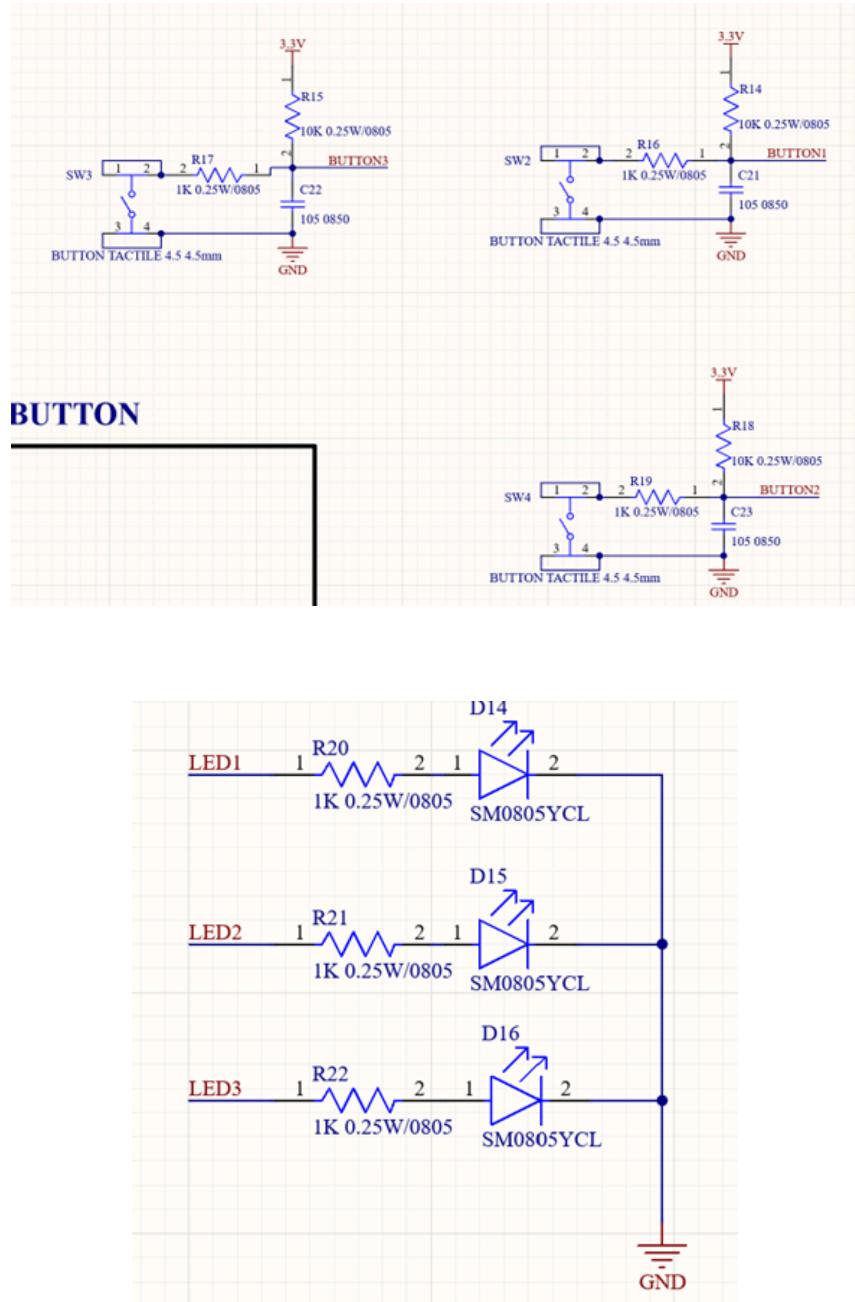
Đường truyền USB được đưa ra thông qua cổng USB Type-A, giúp dễ dàng kết nối với máy tính thông thường mà không cần cáp chuyển đổi đặc biệt. Mặc dù chân USB có đường 5 V, nguồn này không được sử dụng để nuôi hệ thống hoặc nạp MCU. Toàn bộ mạch chỉ dùng cổng USB cho mục đích truyền dữ liệu, tránh việc cấp nguồn ngược từ máy tính gây xung đột nguồn hoặc hỏng mạch.

Để hỗ trợ quan sát hoạt động giao tiếp, mạch tích hợp hai LED báo trạng thái RX và TX, được nối phù hợp để hiển thị khi có dữ liệu nhận hoặc truyền. Điều này giúp việc debug và theo dõi hoạt động UART trực quan hơn.

CH340G được kết nối với MCU thông qua các chân TXD và RXD, với mức tín hiệu đã được đảm bảo phù hợp nhờ sử dụng nguồn 3.3 V. Các tụ bypass và thạch anh dao động 12 MHz được bố trí theo đúng khuyến nghị của datasheet để đảm bảo độ ổn định của giao tiếp USB.

Thiết kế khối CH340G cho phép hệ thống giao tiếp dễ dàng với PC mà không ảnh hưởng đến phần nguồn chính, đồng thời đảm bảo mức logic chính xác và hoạt động ổn định trong môi trường nhiều từ khói công suất.

#### 4.1.6. LED và BUTTON



Hình 4.9: Sơ đồ nguyên lý Button và Led

Một khối nhỏ bao gồm LED và nút nhấn được bố trí gần MCU để tiện quan sát và thao tác. LED dùng để báo trạng thái hoạt động của hệ thống (kích khi mức cao), trong khi nút nhấn dùng cho cấu hình, test hoặc demo, với điện trở kéo lên để khi nhấn, mức tín hiệu xuống 0, đảm bảo MCU đọc đúng trạng thái.

#### 4.1.7. Tiêu kết Schematic

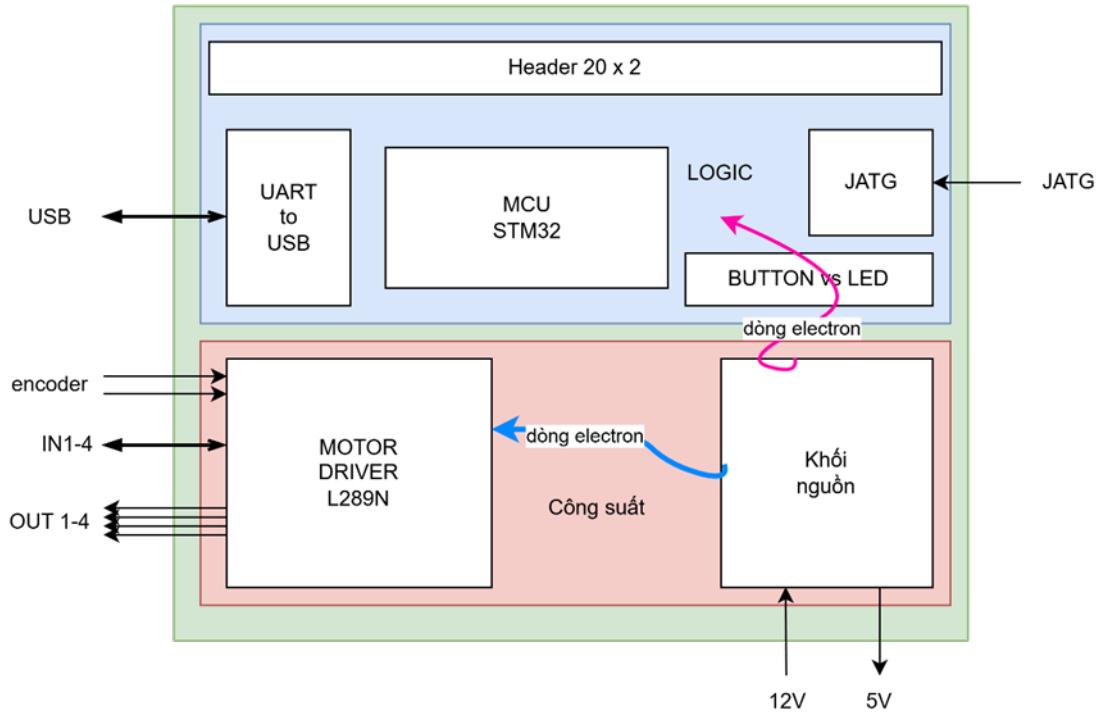
Toàn bộ hệ thống được thiết kế dựa trên nguyên tắc tối ưu tính ổn định, dễ phát triển và khả năng hoạt động tin cậy trong môi trường có nhiều từ tải động cơ. Khối nguồn phân tầng từ 12 V xuống 5 V và 3.3 V giúp đảm bảo mỗi phần mạch nhận được mức điện áp phù hợp, đồng thời giảm thiểu nhiễu xuyên giữa các tầng công suất và điều khiển. Khối MCU STM32F103C8T6 được bố trí đầy đủ các ngoại vi cần thiết như giao tiếp SWD, thạch anh hệ thống, mạch RTC, các header mở rộng và cơ chế lựa chọn chế độ boot, đảm bảo tính linh hoạt và thuận tiện khi phát triển phần mềm.

Khối Motor Driver L298N không chỉ điều khiển công suất mà còn tích hợp khả năng đo dòng qua điện trở shunt và hỗ trợ đọc encoder, cho phép xây dựng các thuật toán điều khiển vòng kín như PID, nâng cao độ chính xác và độ tin cậy của hệ thống. Khối giao tiếp UART-USB sử dụng CH340G giúp hệ thống dễ dàng kết nối với máy tính để debug và trao đổi dữ liệu mà không ảnh hưởng đến nguồn nuôi chính.

Tổng thể thiết kế cho thấy sự kết hợp hợp lý giữa các khối chức năng, đảm bảo tính tương thích, độ bền vững và dễ dàng mở rộng. Đặc biệt, toàn bộ ý tưởng và lựa chọn linh kiện đều dựa trên những module phổ biến trên thị trường, đã được kiểm chứng về độ ổn định, giúp tối ưu đồng thời chất lượng, chi phí, và tính thực tiễn trong triển khai.

## 4.2. Thiết kế PCB

### 4.2.1. Bố trí linh kiện



Hình 4.10: Sơ đồ phác thảo bố trí linh kiện

Trong quá trình thiết kế PCB cho hệ thống điều khiển động cơ sử dụng STM32 và driver L298N, bố trí linh kiện được tổ chức theo hướng phân vùng chức năng rõ ràng nhằm tối ưu hóa khả năng hoạt động, giảm nhiễu và đảm bảo tính thuận tiện khi thao tác, lập trình và kiểm thử. Toàn bộ board mạch được chia thành hai vùng chính: vùng logic ở nửa trên và vùng công suất ở nửa dưới. Cách sắp xếp này dựa trên nguyên tắc tách biệt xử lý tín hiệu và xử lý dòng lớn, từ đó nâng cao độ ổn định của MCU và giảm thiểu nhiễu từ động cơ truyền ngược lên hệ thống điều khiển.

Ở phía trên của board là khối logic, nơi tập trung các linh kiện xử lý và giao tiếp tốc độ thấp. Thành phần quan trọng nhất là vi điều khiển STM32, được đặt gần trung tâm nhằm giúp các đường tín hiệu phân bổ đều, ngắn gọn, dễ định tuyến và giảm thiểu độ trễ hay xuyên nhiễu. Các nút nhấn và LED được bố trí ngay sát MCU vì đây là các phần tử tương tác, báo trạng thái và cấu hình; việc đặt gần giúp tín hiệu ổn định và đảm bảo độ trực quan khi quan sát hoạt động. Phía bên trái của vùng logic là cổng JTAG, được đưa sát mép ngoài để việc lập trình, debug hoặc nạp firmware trở nên

thuận tiện, đồng thời không gây cản trở khi kết nối các thành phần khác trên board. Bên cạnh đó, header mở rộng 20x2 được đặt ở cạnh trên cùng nhằm tạo khả năng kết nối linh hoạt với các module ngoại vi như LCD, UART, SPI, I2C hoặc các mạch mở rộng khác; việc bố trí mép trên giúp việc thao tác dây cáp trở nên gọn gàng và dễ dàng hơn trong môi trường thử nghiệm. Ở phía phải của vùng logic là khối USB-UART dùng chip CH340G, đặt gần cạnh board để dễ kết nối với máy tính. CH340G chỉ sử dụng nguồn 3.3V ổn định từ hệ thống; nguồn 5V trên cổng USB chỉ phục vụ giao tiếp mà không cấp cho tải hoặc MCU để tránh xung ngược và đảm bảo an toàn nguồn. Hai LED TX/RX được bố trí ngay cạnh Vùng công suất nằm ở nửa dưới PCB, nơi xử lý dòng lớn cấp cho động cơ. Thành phần chính của vùng này là driver động cơ L298N, được đặt sát cạnh dưới để có thể dễ dàng gắn thêm tản nhiệt hoặc tiếp xúc thoáng khí khi tải dòng cao. Các cổng OUT1-OUT4 được bố trí hướng ra ngoài để việc kết nối với động cơ thuận tiện, giảm chiều dài dây dẫn và hạn chế nhiễu điện từ. Các chân điều khiển IN1-IN4 được đưa lên khu vực test giúp quan sát và đo đặc tín hiệu khi cần thiết. Khu vực này cũng bao gồm cổng Encoder, được đặt gần phần driver để tránh kéo dây xung quá dài, vốn có thể gây nhiễu hoặc sai lệch xung khi tốc độ quay cao. Phần nguồn đầu vào 12V được bố trí ở rìa vùng công suất để dễ đấu nối và để đường nguồn ra ngắn nhất có thể. Nguồn này được chia thành hai nhánh: một nhánh cấp cho vùng logic (thông qua hệ thống lọc, ổn áp và bảo vệ), một nhánh cấp cho vùng công suất. Việc tách riêng hai luồng dòng electron và tách mặt đất của hai vùng giúp giảm nhiễu cao tần từ động cơ ảnh hưởng ngược lên MCU, đồng thời hạn chế rung nhiễu gây reset hệ thống. Ngoài ra, board còn tích hợp terminal 5V output để cấp cho các ngoại vi nhẹ bên ngoài.

Hầu hết các linh kiện thụ động được bố trí ở mặt sau của PCB nhằm tiết kiệm diện tích mặt trước, tối ưu bố trí và tăng tính thẩm mỹ của board. Trong khi đó, bộ chuyển đổi USB-UART được đặt ở mặt trước, với các LED TX/RX bố trí ngay cạnh, giúp người dùng dễ dàng quan sát hoạt động truyền nhận dữ liệu trong quá trình vận hành và debug. Toàn bộ bố trí linh kiện tuân theo nguyên tắc các cổng giao tiếp (USB, JTAG, Header mở rộng, cổng motor) được đưa ra mép board, MCU nằm ở vùng trung tâm để tối ưu định tuyến, còn khối công suất nằm riêng biệt để đảm bảo tính ổn định và an toàn. Các ý tưởng bố trí và lựa chọn linh kiện đều được tham khảo từ các module thương mại phổ biến trên thị trường nhằm tối ưu chất lượng, chi phí và tính thực tiễn, giúp board mạch dễ sản xuất, dễ sử dụng và phù hợp để học tập, nghiên cứu hay triển khai demo thực tế.

#### 4.2.2. Đặt xếp lớp (stack layer) và luật (rule)

Trong thiết kế PCB cho hệ thống điều khiển động cơ, việc xác định thông số xếp lớp (stack-up) và các quy tắc thiết kế là bước quan trọng nhằm đảm bảo tính sản xuất được, độ tin cậy điện, khả năng chịu dòng và giới hạn nhiễu. Board mạch được thực hiện với cấu trúc hai lớp tiêu chuẩn, sử dụng vật liệu FR-4, hướng đến độ cứng cơ khí tốt và dễ dàng trong gia công.

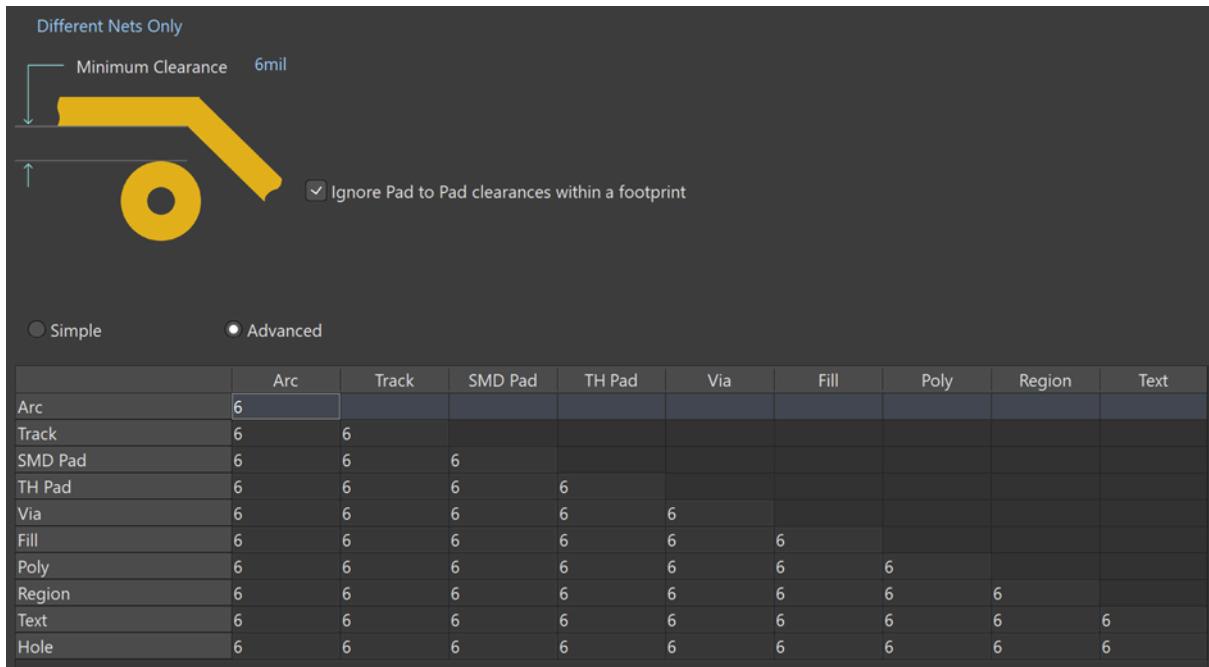
##### Stack layer

#	Name	Material	Type	Weight	Thickness	Dk
	Top Overlay		Overlay			
	Top Solder	Solder Resist	Solder Mask		0.4mil	3.5
1	Top Layer		Signal	1oz	1.4mil	
	Dielectric 1	FR-4	Dielectric		59.4mil	4.8
2	Bottom Layer		Signal	1oz	1.4mil	
	Bottom Solder	Solder Resist	Solder Mask		0.4mil	3.5
	Bottom Overlay		Overlay			

Hình 4.11: Đặt stack layer

PCB được thiết kế theo dạng 2-layer, gồm lớp Top và Bottom đều là lớp tín hiệu. Cấu trúc cụ thể như sau: Số lớp: 2 lớp (Top Layer – Bottom Layer) Độ dày đồng (Copper thickness): 1 oz (tương đương 2.8 mil) cho cả hai lớp Lớp điện môi (Dielectric): FR-4 với độ dày 59.4 mil Tổng độ dày PCB: khoảng 63 mil (tương đương 1.6 mm), hoàn toàn phù hợp với tiêu chuẩn sản xuất phổ biến trong công nghiệp và tương thích với hầu hết socket, connector và thiết bị cơ khí liên quan. Việc sử dụng đồng 1 oz là lựa chọn cân bằng giữa chi phí và khả năng chịu dòng, đặc biệt khi hệ thống có chứa đường truyền công suất cho động cơ với dòng có thể lên đến 2–4 A. Vật liệu FR-4 đảm bảo độ ổn định cơ học, độ bền nhiệt và tính cách điện tốt.

## Clearance rule



Hình 4.12: Đặt Clearance rule

Về quy tắc thiết kế, khoảng cách an toàn (clearance) giữa các đường mạch được đặt ở mức 6 mil trên toàn bộ board, nhằm đảm bảo khả năng sản xuất tốt ngay cả với các xưởng chế tạo có tiêu chuẩn trung bình. Quy tắc này áp dụng cho clearance giữa các đường tín hiệu thuộc các net khác nhau và phù hợp với khả năng công nghệ của PCB 2 lớp. Riêng clearance giữa pad-to-pad trong cùng một footprint không áp đặt giới hạn này vì thuộc cùng net, tuy nhiên hệ thống thiết kế vẫn chú ý để không vi phạm khả năng gia công. Đây là một điểm cần cân nhắc trong giai đoạn kiểm tra DRC, nhưng nhìn chung vẫn nằm trong tiêu chuẩn cho phép của các nhà sản xuất phổ thông.

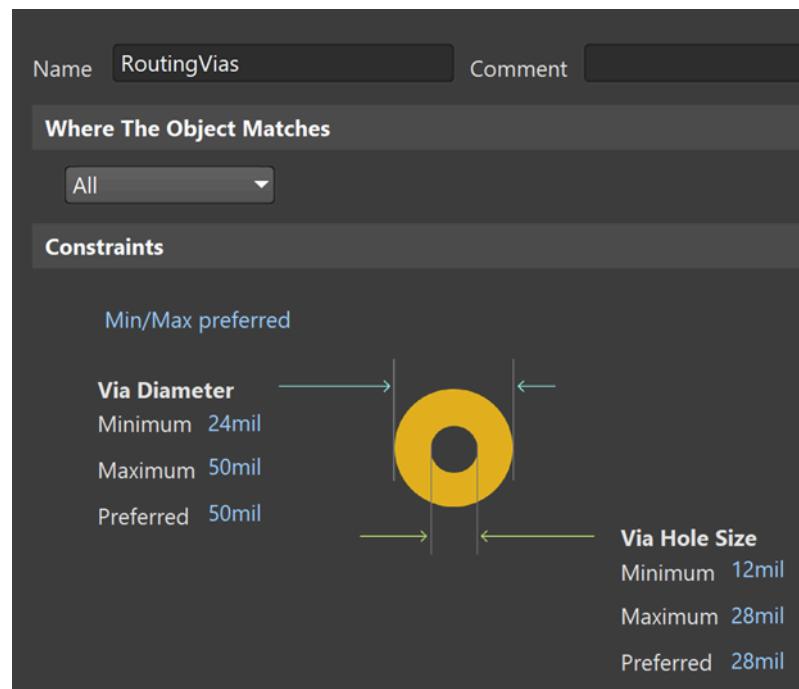
## Trace width rule

Name	Priority	Enabled	Type	Category	Scope	Attributes
~GND	1	<input checked="" type="checkbox"/>	Width	Routing	InNet('GND')	Pref Width = 20mil Min Width = 10mil Max Width = 60mil
~5V	2	<input checked="" type="checkbox"/>	Width	Routing	InNet('5V')	Pref Width = 20mil Min Width = 10mil Max Width = 30mil
~OUT	3	<input checked="" type="checkbox"/>	Width	Routing	InNetClass('OUT')	Pref Width = 30mil Min Width = 10mil Max Width = 60mil
~12V	4	<input checked="" type="checkbox"/>	Width	Routing	InNet('12V')	Pref Width = 20mil Min Width = 10mil Max Width = 60mil
~All	5	<input checked="" type="checkbox"/>	Width	Routing	All	Pref Width = 10mil Min Width = 10mil Max Width = 30mil

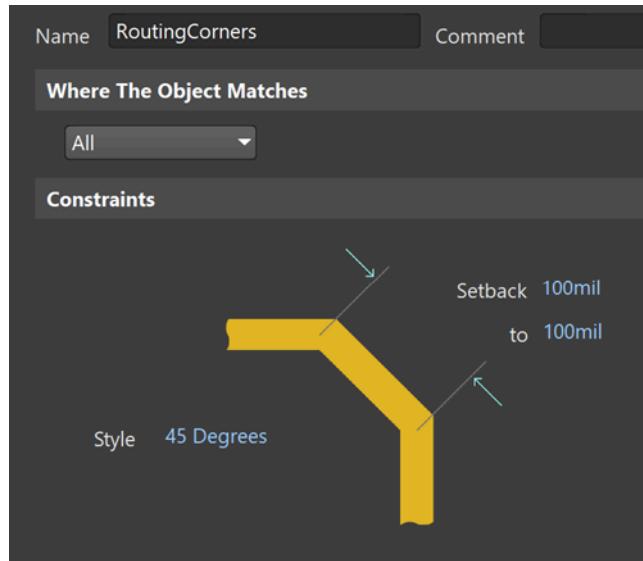
Hình 4.13: Đặt trace width rule

Chiều rộng đường mạch (trace width) được thiết kế theo nguyên tắc phù hợp với mức dòng từng khối chức năng. Đối với các đường tín hiệu thông thường, chiều rộng tối thiểu là 10 mil, đảm bảo độ ổn định và dễ dàng trong quá trình định tuyến. Các đường GND được chọn linh hoạt từ 10 mil đến 20 mil và có thể mở rộng lên 60 mil tại các đoạn chịu dòng cao để giảm điện trở và giảm sụt áp. Đối với đường nguồn 5V, chiều rộng sử dụng trong khoảng từ 10 đến 30 mil, phù hợp với dòng tiêu thụ của khối logic. Các đường công suất thuộc ngõ ra driver động cơ (OUT1–OUT4), nơi dòng có thể đạt đến 2 A, được mở rộng từ 10 mil ở vùng nhỏ, lên 30 hoặc tối đa 60 mil tại các đoạn chịu tải lớn để đảm bảo an toàn nhiệt. Đường 12V và GND chính, nơi tổng dòng có thể lên đến 4 A, luôn được ưu tiên sử dụng độ rộng lớn nhất 60 mil để đảm bảo khả năng dẫn dòng và tránh phát nhiệt trong vùng công suất.

## Via rule và Corners rule



Hình 4.14: Đặt Routing Vias Rule

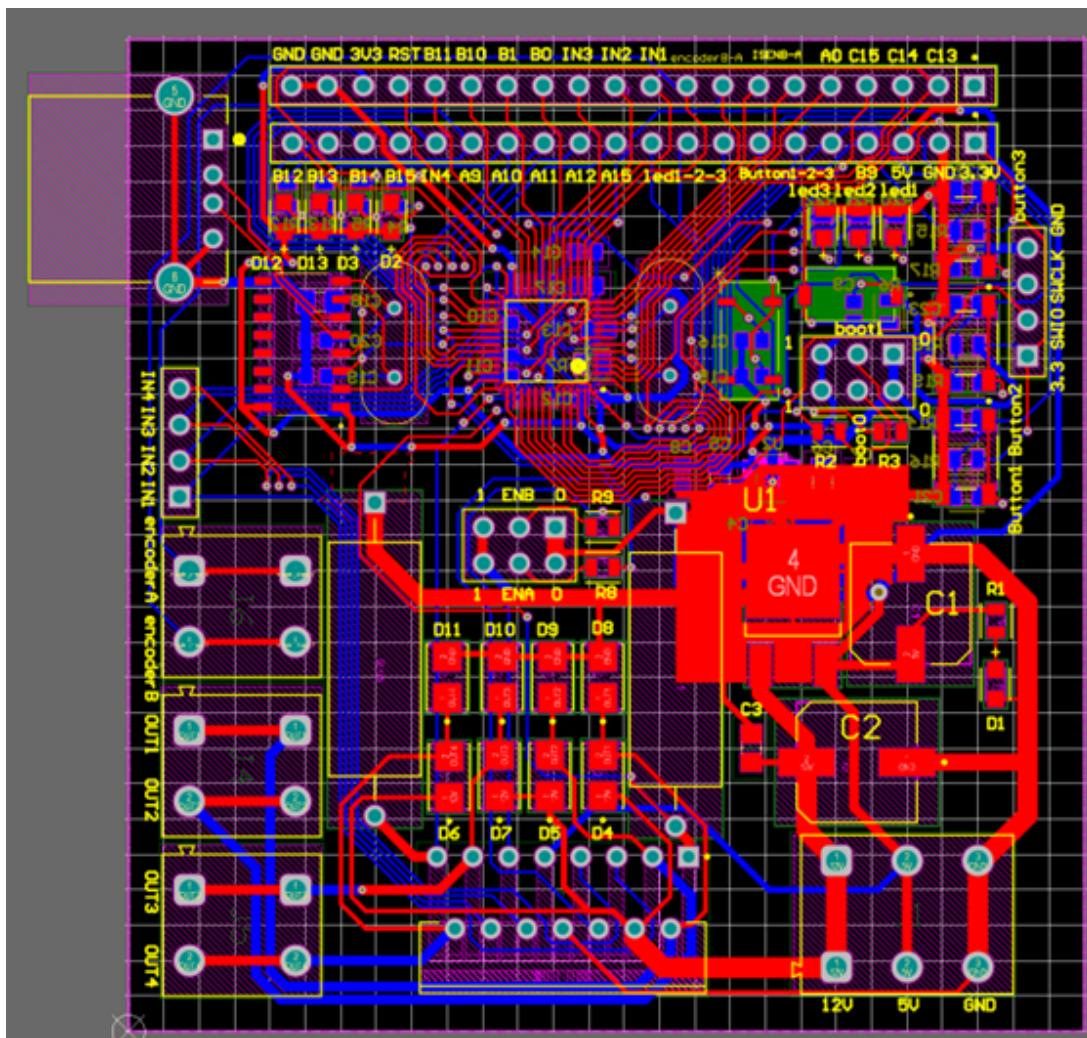


Hình 4.15: Đặt Routing Corners Rule

Thiết kế via cũng được phân thành hai loại tùy theo chức năng. Loại phổ thông sử dụng kích thước 12/28/28 mil (đường kính khoan/lò xo tổng/đệm), phù hợp với hầu hết tín hiệu và dễ dàng cho sản xuất hàng loạt. Riêng các đường công suất hoặc các điểm giao tiếp cần độ bền cơ học lớn hơn, sử dụng loại via kích thước 24/50/50 mil, giúp tăng khả năng dẫn dòng và đảm bảo độ tin cậy trong quá trình vận hành lâu dài. Việc định tuyến được thực hiện trên cả hai lớp top và bottom, đảm bảo bố trí tối ưu nhất cho từng phân vùng mạch và giảm giao cắt không cần thiết. Các góc cua của đường mạch tuân theo quy tắc bo 45 độ nhằm giảm phản xạ tín hiệu, hạn chế tập trung dòng và cải thiện tính thẩm mỹ chung của board. Ngoài ra, phần annular expansion (expand) trên pad và via được sử dụng theo giá trị mặc định 4 mil, phù hợp với khả năng chế tạo và giúp pad dễ bám đồng trong quá trình ăn mòn. Việc sử dụng giá trị tiêu chuẩn này đảm bảo PCB đạt độ tin cậy cao mà không làm tăng chi phí sản xuất. Toàn bộ cấu trúc lớp, quy tắc thiết kế và phương pháp định tuyến được tối ưu để vừa đảm bảo tính an toàn điện, vừa tạo điều kiện sản xuất dễ dàng, chi phí hợp lý và giữ được độ bền vững khi hoạt động trong môi trường thực tế, đặc biệt là các mạch có tải công suất như driver động cơ.

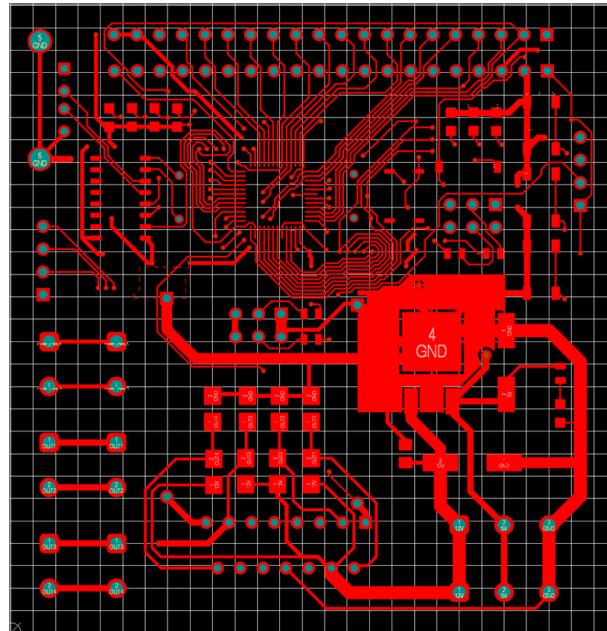
#### 4.2.3. Triển khai PCB

Hình dưới đây thể hiện tổng thể bố cục PCB ở chế độ layout 2D. Board được chia thành hai vùng rõ rệt: phần logic ở phía trên gồm MCU, USB-UART, header mở rộng và các khối điều khiển; phần công suất nằm phía dưới gồm bộ nguồn và Motor Driver L298N. Cách bố trí này đảm bảo tách nhiễu giữa tải động cơ và mạch điều khiển, đồng thời thuận tiện trong thao tác test và mở rộng hệ thống.

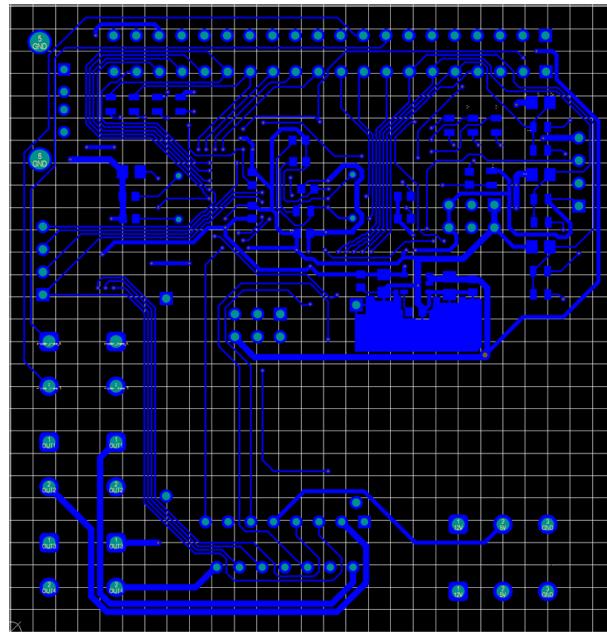


Hình 4.16: Bố cục PCB ở chế độ layout 2D.

## Đi dây



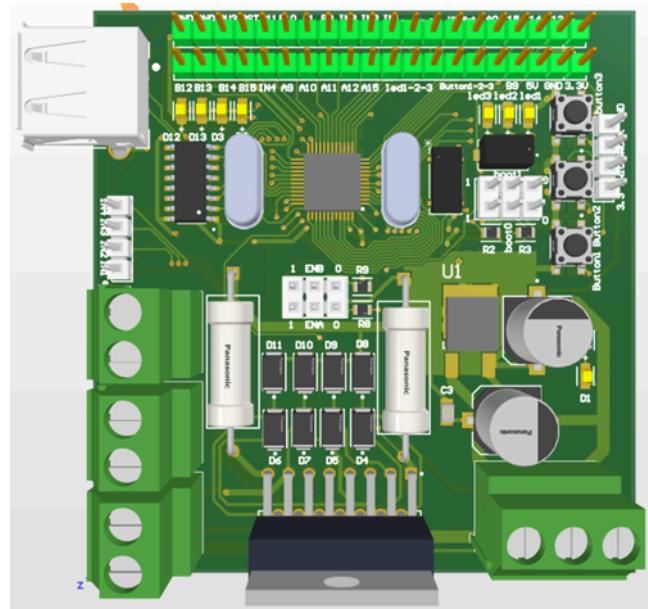
Hình 4.17: *Đi dây Top Layer*



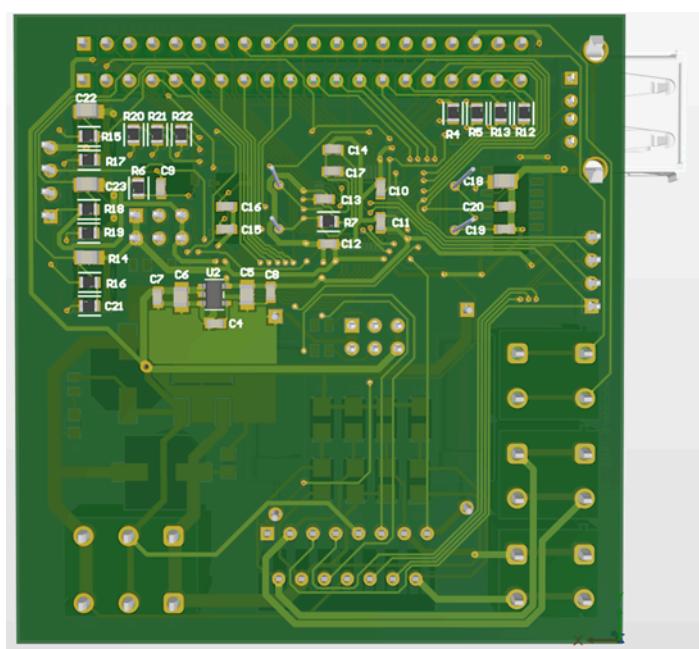
Hình 4.18: *Đi dây Bottom Layer*

Hình minh họa dạng 2D của board mạch ở mặt Top và bottom, thể hiện rõ bố cục các khối chức năng, đường tín hiệu và vùng đất GND. Các đường công suất được bố trí rộng, các đường logic được định tuyến ngắn và tách biệt để giảm nhiễu.

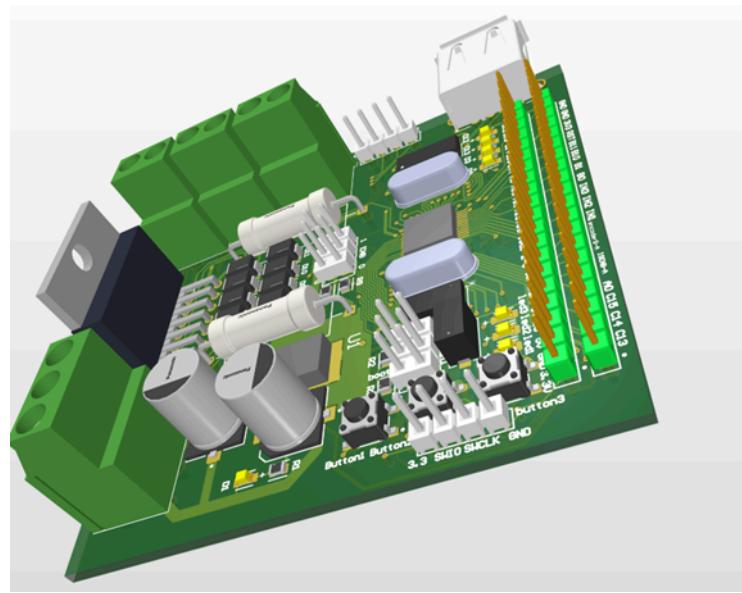
3D



Hình 4.19: Bố cục 3D: Mặt trước



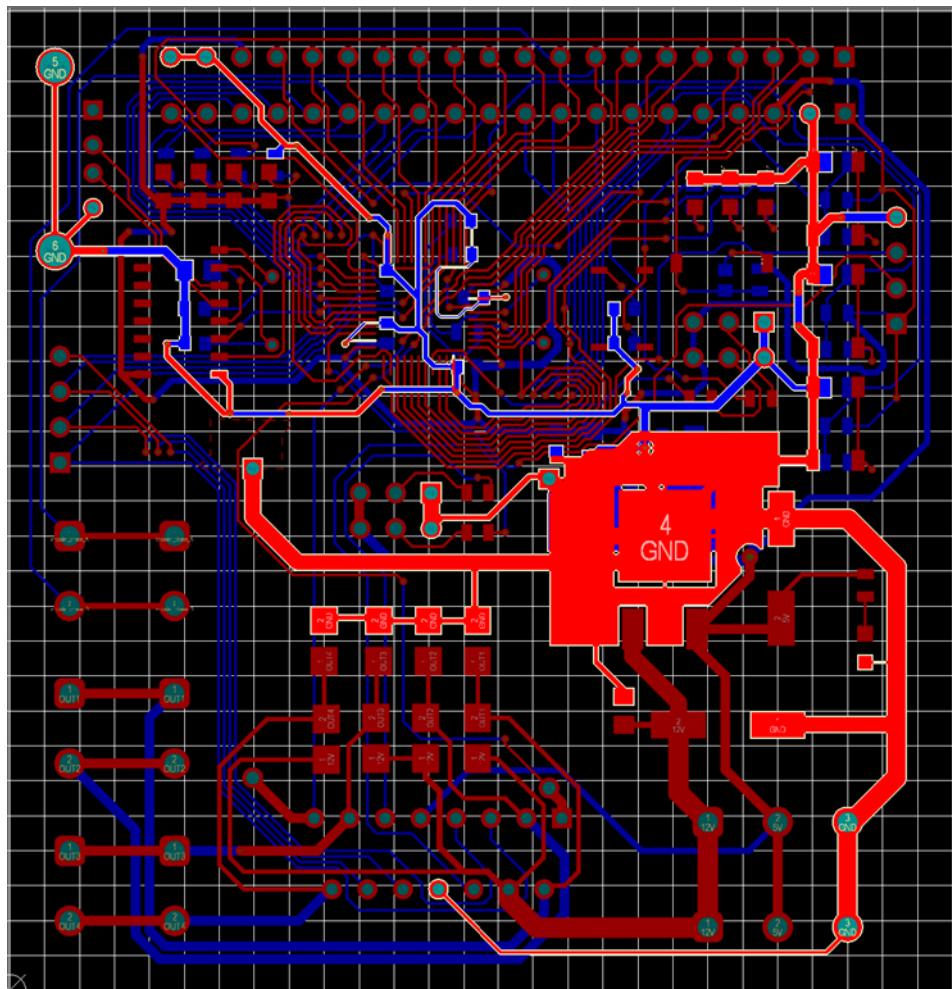
Hình 4.20: Bố cục 3D: Mặt sau



Hình 4.21: Bô cục 3D: Măt nghiên

Hình 3D dưới đây thể hiện bố cục trực quan của PCB sau khi lắp ráp linh kiện. Các khối logic (MCU, USB-UART, header mở rộng, nút nhấn, LED) nằm ở phần trên của board, trong khi khối công suất (nguồn, Motor Driver L298N) nằm ở phần dưới nhằm tách nhiễu và tối ưu khả năng thoát nhiệt.

## GND



Hình 4.22: Bố cục 2D: Đি dây GND

Từ điểm nối đất chính, các nhánh được dẫn tới từng khối chức năng: MCU, USB-UART, khối logic và khối công suất. Chiến lược phân nhánh này giúp giảm nhiễu cao tần, tránh Ground Loop và vẫn đảm bảo dẫn dòng đủ lớn cho các mạch công suất. Các nhánh GND được thiết kế với chiều rộng tương ứng với mức dòng từng khối, từ 10 mil cho tín hiệu nhỏ tới 60 mil cho đường GND chính chịu dòng lớn, đảm bảo an toàn và độ ổn định của hệ thống.

## Design Rule Verification



### Design Rule Verification Report

**Date:** 12/10/2025  
**Time:** 2:10:38 AM  
**Elapsed Time:** 00:00:01  
**Filename:** C:\Users\MyLaptop\Desktop\TKHT\_NHUNG\_SLIDE\DC MOTOR CONTROL\PCB.PcbDoc

**Warnings:** 0  
**Rule Violations:** 96  
**PCB Health Issues:** 0

#### Summary

Warnings	Count
<b>Total</b>	<b>0</b>

Rule Violations	Count
<a href="#">Clearance Constraint (Gap=6mil) (All) (All)</a>	0
<a href="#">Short-Circuit Constraint (Allowed=No) (All) (All)</a>	0
<a href="#">Un-Routed Net Constraint (All) (All)</a>	0
<a href="#">Modified Polygon (Allow modified: No) (Allow shelved: No)</a>	0
<a href="#">Width Constraint (Min=10mil) (Max=60mil) (Preferred=20mil) (InNet('GND'))</a>	0
<a href="#">Width Constraint (Min=10mil) (Max=60mil) (Preferred=20mil) (InNet('12V'))</a>	0
<a href="#">Width Constraint (Min=10mil) (Max=30mil) (Preferred=10mil) (All)</a>	0
<a href="#">Width Constraint (Min=10mil) (Max=30mil) (Preferred=20mil) (InNet('5V'))</a>	0
<a href="#">Width Constraint (Min=10mil) (Max=60mil) (Preferred=30mil) (InNetClass('OUT'))</a>	0
<a href="#">Routing Topology Rule(Topology=Shortest) (All)</a>	0
<a href="#">Power Plane Connect Rule(Relief Connect) (Expansion=20mil) (Conductor Width=10mil) (Air Gap=10mil) (Entries=4) (All)</a>	0
<a href="#">Hole Size Constraint (Min=1mil) (Max=100mil) (All)</a>	0
<a href="#">Pads and Vias to follow the Drill pairs settings</a>	0
<a href="#">Hole To Hole Clearance (Gap=10mil) (All) (All)</a>	0
<a href="#">Silk To Solder Mask (Clearance=10mil) (IsPad) (All)</a>	42
<a href="#">Silk to Silk (Clearance=10mil) (All) (All)</a>	54
<a href="#">Net Antennae (Tolerance=0mil) (All)</a>	0
<a href="#">Height Constraint (Min=0mil) (Max=1000mil) (Preferred=500mil) (All)</a>	0
<b>Total</b>	<b>96</b>

PCB Health Issues	Count
<b>Total</b>	<b>0</b>

Hình 4.23: Design Rule Verification

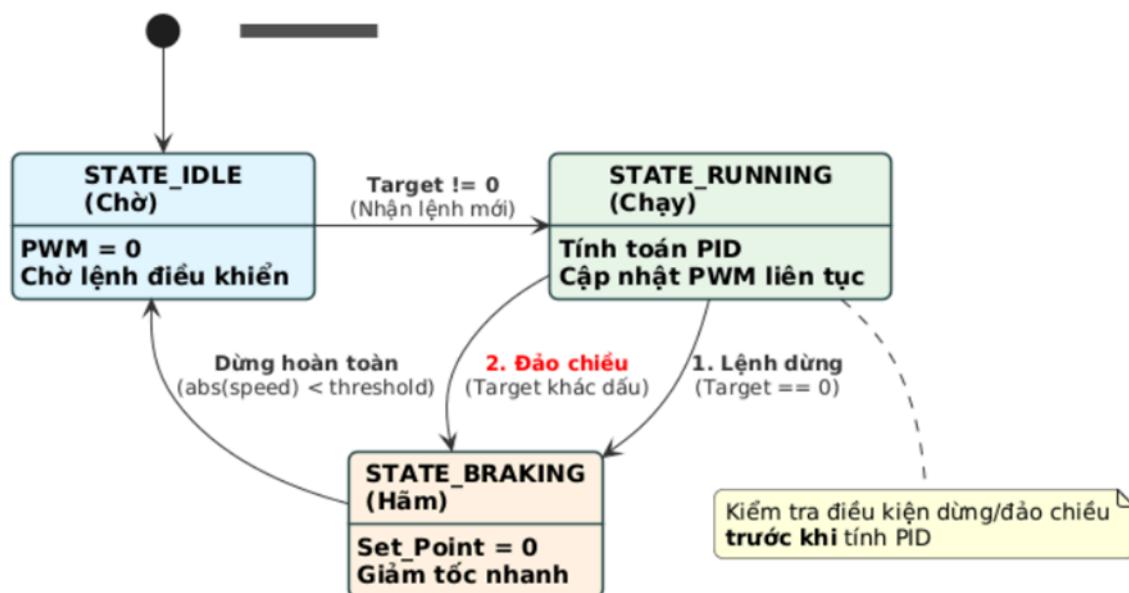
Sau khi hoàn tất quá trình bố trí linh kiện và định tuyến, toàn bộ PCB được chạy kiểm tra Design Rule Check (DRC) trong Altium Designer nhằm xác nhận việc tuân thủ các quy tắc thiết kế đã đặt ra. Kết quả cho thấy không có cảnh báo nghiêm trọng nào liên quan tới mạch, và không có vấn đề về sức khỏe PCB (PCB Health Issues = 0), khẳng định toàn bộ board đáp ứng tiêu chuẩn sản xuất và vận hành. Các quy tắc về clearance, width, hole size, via, routing topology và kết nối plane đều được thực hiện chính xác. Cụ thể, khoảng cách giữa các đường mạch (clearance) được đảm bảo 6 mil, đáp ứng an toàn sản xuất; các đường dẫn GND, 12V, 5V và OUT được tuân theo giới hạn chiều rộng tối thiểu, tối đa và chiều rộng ưu tiên, từ 10 mil đến 60 mil tùy theo dòng chịu tải. Toàn bộ các đường công suất và logic đều được kiểm tra, không có net nào bị bỏ sót hay nối ngắn (short-circuit), đảm bảo tín hiệu và dòng điện chạy đúng theo thiết kế. Một số vi phạm nhỏ chỉ liên quan đến silk-to-solder mask clearance và silk-to-silk clearance, với tổng cộng 96 điểm, đây hoàn toàn là vấn đề về thẩm mỹ hoặc chữ in trên PCB, không ảnh hưởng đến hoạt động điện của board. Việc này cũng là phổ biến trong quá trình thiết kế khi chữ ký hiệu hoặc ký hiệu linh kiện gần mép pad, và không phải lỗi nghiêm trọng. Kết quả DRC khẳng định rằng PCB đã được tối ưu hóa và kiểm soát tốt, tuân thủ các tiêu chuẩn kỹ thuật, sẵn sàng cho bước sản xuất và lắp ráp. Điều này cũng chứng minh rằng các quy tắc thiết kế, từ bố trí linh kiện, phân vùng logic – công suất, định tuyến đường tín hiệu và công suất, đến cách phân nhánh GND kiểu “tán cây”, đều đã được áp dụng một cách hiệu quả và chính xác.

### 4.3. Thiết kế Firmware (Firmware design)

Hệ thống Firmware được xây dựng dựa trên kiến trúc hướng sự kiện (Event-driven), sử dụng máy trạng thái hữu hạn (FSM) để quản lý hành vi động cơ và bộ định thời (Timer) để đảm bảo chu kỳ lấy mẫu cố định cho thuật toán điều khiển.

#### 4.3.1. Sơ đồ trạng thái (Finite State Machine - FSM)

Hệ thống điều khiển được mô hình hóa thành 3 trạng thái hoạt động chính để đảm bảo an toàn cho phần cứng (Driver và Động cơ) khi đảo chiều đột ngột.



Mô tả chuyển đổi trạng thái:

- STATE\_IDLE (Chờ): Động cơ dừng hoàn toàn ( $PWM = 0$ ). Hệ thống chờ lệnh từ người dùng.

Điều kiện chuyển: Khi nhận được  $target\_set\_point \neq 0$ .

- STATE\_RUNNING (Chạy): Động cơ hoạt động ổn định, thuật toán PID liên tục tính toán sai số và cập nhật PWM.

Điều kiện chuyển 1: Nếu người dùng gửi lệnh dừng ( $target\_set\_point == 0$ ) → chuyển sang BRAKING.

Điều kiện chuyển 2: Nếu phát hiện lệnh đảo chiều (Target và Current Setpoint khác dấu) → chuyển sang BRAKING (để tránh dòng ngược làm hỏng mạch cầu H).

- STATE\_BRAKING (Hãm): Động cơ được gán set\_point = 0 để giảm tốc độ về 0 nhanh chóng.

Điều kiện chuyển: Khi tốc độ thực tế đo được xấp xỉ 0 → chuyển về IDLE (để sẵn sàng cho chu trình mới).

### 4.3.2. Vòng lặp điều khiển PID (PID Loop)

Thuật toán PID được thực thi ròng rạc hóa theo thời gian. Chu kỳ lấy mẫu (Ts): 100ms (được định thời bởi Timer 1). Quy trình xử lý:

1. Đo lường: Đọc số xung Encoder tích lũy trong 100ms.
2. Tính toán vận tốc:

$$V_{\text{rpm}} = \frac{\text{PulseCount} \times 60}{\text{PPR} \times \text{GearRatio} \times T_s}$$

(Trong đó: PPR = 11, GearRatio = 21.3, Ts = 0.1s)

3. Tính sai số:  $e(t) = \text{Setpoint} - V_{\text{rpm}}$
4. Thuật toán PID:

$$u(t) = K_p \cdot e(t) + K_i \sum e(t)T_s + K_d \frac{e(t) - e(t-1)}{T_s}$$

5. Bão hòa (Saturation): Giá trị ngõ ra u(t) được giới hạn trong khoảng [0, 100] (tương ứng 0-100% Duty Cycle).

### 4.3.3. Lưu đồ giải thuật

Chương trình chính (main) hoạt động theo cơ chế Super Loop, liên tục kiểm tra các cờ sự kiện (Flags) được bật từ các trình phục vụ ngắn (ISR). Giải thích lưu đồ:

1. Khởi tạo (Init): Cấu hình Clock hệ thống, GPIO, Timer, UART và các thông số PID ban đầu.
2. Vòng lặp chính (Super Loop):

- Kiểm tra cờ Timer (100ms): Nếu Timer tràn, thực hiện tính toán vận tốc động cơ và reset biến đếm xung.
- Kiểm tra cờ UART: Nếu bộ đếm nhận đầy đủ gói tin, thực hiện tách chuỗi (Parse) để cập nhật thông số điều khiển (Start/Stop, Kp, Ki, Setpoint).
- Máy trạng thái: Quyết định trạng thái hiện tại (Running/Braking/Idle).
- PID Calculation: Tính toán giá trị PWM dựa trên sai số.

- Cập nhật phần cứng: Xuất xung PWM ra chân PA0 và tín hiệu chiều quay ra PA1/PA12.

#### 4.3.4. Cấu hình Timer và PWM

Hệ thống sử dụng vi điều khiển STM32F103 chạy ở xung nhịp nội HSI 8 MHz (dựa theo cấu hình RCC\_OSCILLATORTYPE\_HSI trong code).

##### a. Timer 2 - Tạo xung PWM (Motor Speed Control)

Chức năng: Tạo xung PWM điều khiển tốc độ động cơ.

Cấu hình:

- Prescaler (PSC): 7
- Counter Period (ARR): 999

Tính toán tần số PWM

$$F_{\text{PWM}} = \frac{F_{\text{CLK}}}{(\text{PSC} + 1)(\text{ARR} + 1)} = \frac{80000000}{(7 + 1)(999 + 1)} = 1000(\text{Hz})$$

Tần số PWM là 1 kHz, phù hợp với driver động cơ DC thông thường, giúp giảm tiếng ồn tần số cao và hạn chế tổn hao chuyển mạch trên MOSFET.

##### b. Timer 1 - Định thời lấy mẫu (Sampling Time)

Chức năng: Tạo ngắt định thời gian chính xác để tính toán vận tốc. Cấu hình:

- Prescaler (PSC): 799
- Counter Period (ARR): 999

Tính toán tần số ngắt

$$F_{\text{update}} = \frac{8000000}{(799 + 1)(999 + 1)} = 10\text{Hz}$$

-> Chu kỳ lấy mẫu là  $T = 100\text{ms}$

##### c. Timer 3 - Định thời gửi dữ liệu (Telemetry)

Chức năng: Gửi dữ liệu lên máy tính với tần số thấp hơn để tránh quá tải đường truyền UART.

Cấu hình:

- Prescaler: 799

- Period: 2499

Tính toán: Tần số ngắn là 4 Hz  $\Rightarrow$  Gửi dữ liệu mỗi 250ms.

## 4.4. Thiết kế Giao diện người dùng - phần mềm giám sát

Phần mềm giám sát trên máy tính (Host Software) đóng vai trò là giao diện người - máy (HMI), cho phép người dùng tương tác với hệ thống nhúng, cài đặt tham số điều khiển và quan sát đáp ứng của động cơ một cách trực quan.

### 4.4.1. Kiến trúc phần mềm

Phần mềm được xây dựng trên nền tảng Microsoft .NET Framework sử dụng Windows Forms (WinForms). Kiến trúc chương trình được thiết kế theo mô hình Hướng sự kiện (Event-Driven Programming), trong đó các hành động của hệ thống được kích hoạt bởi các sự kiện từ người dùng (nhấn nút, nhập liệu) hoặc sự kiện từ phần cứng (nhận dữ liệu từ cổng COM). Cấu trúc chương trình bao gồm 3 tầng xử lý chính:

1. Tầng Giao diện (Presentation Layer): Các nút nhấn, ô nhập liệu, và biểu đồ ZedGraph.
2. Tầng Logic điều khiển (Logic Layer): Xử lý đóng gói lệnh, phân tích dữ liệu nhận được (Parsing), và cập nhật đồ thị.
3. Tầng Truyền thông (Communication Layer): Quản lý đối tượng Serial Port, thiết lập kết nối và đệm dữ liệu thô.

### 4.4.2. Thiết kế giao diện người dùng

**Giao diện chính (Form1) được chia thành các khu vực chức năng:**

Khu vực Kết nối (Connection Panel): Cho phép chọn cổng COM và thực hiện kết nối/ngắt kết nối. Trạng thái kết nối được hiển thị qua Label và ProgressBar.

Khu vực Cài đặt (Control Panel): Các TextBox cho phép nhập tham số bộ điều khiển (Kp, Ki), Tốc độ đặt (Setpoint), và Giới hạn biên độ (Amp\_max, Amp\_min).

**Khu vực Hiển thị (Visual Panel):**

- Biểu đồ 1 (Speed Response): Hiển thị đồng thời hai đường tín hiệu theo thời gian thực: Tốc độ đặt (Màu xanh) và Tốc độ thực (Màu đCo).
- Biểu đồ 2 (Control Signal): Hiển thị tín hiệu điều khiển (Duty cycle) để giám sát trạng thái bão hòa của bộ điều khiển.

#### 4.4.3. Xử lý Đa luồng và Truyền thông

Một thách thức lớn trong lập trình giao diện giám sát là xử lý dữ liệu tốc độ cao từ vi điều khiển mà không làm “treo” giao diện người dùng (UI Freezing).

##### a. Cơ chế nhận dữ liệu (Data Reception)

Sử dụng sự kiện DataReceived của lớp SerialPort. Sự kiện này chạy trên một luồng (Thread) riêng biệt, khác với luồng giao diện chính (Main UI Thread). • Vấn đề: Không thể cập nhật trực tiếp các điều khiển giao diện (TextBox, ZedGraph) từ luồng DataReceived (gây lỗi Cross-thread operation). • Giải pháp: Sử dụng phương thức BeginInvoke (Delegate) để đẩy việc cập nhật dữ liệu về lại luồng UI chính một cách an toàn và bất đồng bộ.

##### b. Giải thuật phân tích gói tin (Packet Parsing)

Dữ liệu từ vi điều khiển gửi lên dưới dạng chuỗi ASCII: “Setpoint,ActualSpeed,DutyCyclern”.

Quy trình xử lý tại máy tính:

- Đọc dòng dữ liệu (ReadLine).
- Tách chuỗi dựa vào ký tự phân cách dấu phẩy (,).
- Chuyển đổi chuỗi sang số thực (double.TryParse).
- Thêm điểm dữ liệu mới vào danh sách RollingPointPairList của ZedGraph.
- Cập nhật trực thời gian (Time Axis) để tạo hiệu ứng cuộn (Scrolling Graph).

#### 4.4.4. Định nghĩa Giao thức truyền thông

Để đảm bảo tính toàn vẹn dữ liệu, nhóm quy định khung truyền dữ liệu (Data Frame) như sau:

**Chiều Máy tính -> Vi điều khiển (Command Frame)** Gói tin là một chuỗi ký tự chứa đầy đủ các tham số cấu hình, kết thúc bằng ký tự ‘R’.

Cấu trúc: MotorState,CtrlType,InputType,Kp,Ki,Gp,GI,Max,Min,Setpoint,R

Ví dụ: 1,0,0,1.5,0.5,0,0,100,0,150R

- 1: Bật động cơ.

- 0: Chế độ PID.

- 0: Input là hằng số (Setpoint).
- 1.5, 0.5: Tham số K\_p, K\_i.
- 150: Tốc độ đặt 150 RPM.
- R: Ký tự kết thúc lệnh (Terminator).

### Chiều Vi điều khiển -> Máy tính (Telemetry Frame)

Gói tin chứa dữ liệu giám sát, gửi định kỳ mỗi 250ms.

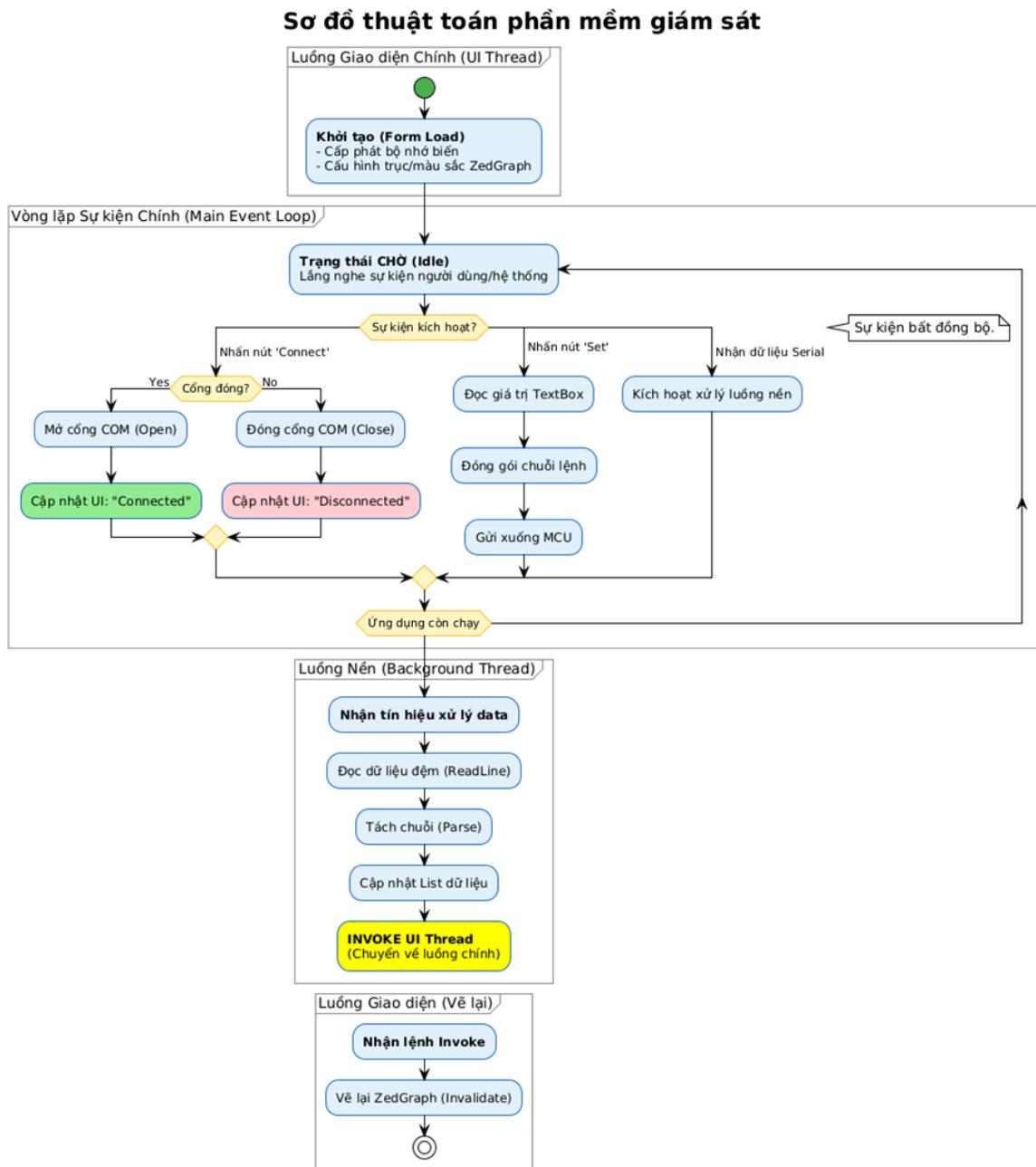
Cấu trúc: Setpoint, CurrentSpeed, ControlSignalrn

Ví dụ: 150.00,148.50,45.20 Ví dụ: 1,0,0,1.5,0.5,0,0,100,0,150R

- Tốc độ thực: 148.5 RPM.
- Duty Cycle: 45.2%.

#### 4.4.5. Lưu đồ giải thuật

Dưới đây là sơ đồ khối (Flowchart) được thiết kế chuyên nghiệp để bạn đưa vào báo cáo, kèm theo phần giải thích chi tiết mang tính kỹ thuật.



Hình 4.24: Lưu đồ giải thuật

Phần mềm được thiết kế theo mô hình Lập trình hướng sự kiện (Event-Driven Programming). Chương trình không chạy tuần tự từ trên xuống dưới rồi kết thúc, mà sau khi khởi tạo sẽ rơi vào trạng thái “Chờ” (Idle) để lắng nghe các tác động từ người dùng hoặc hệ thống. Quá trình hoạt động được chia làm 3 giai đoạn chính:

## **Giai đoạn 1: Khởi tạo (Initialization)**

Ngay khi phần mềm được khởi động (Form\_Load), hệ thống thực hiện:

- Khởi tạo biến: Cấp phát bộ nhớ cho các biến toàn cục và danh sách dữ liệu (RollingPointPairList) dùng để lưu trữ các điểm vẽ đồ thị.
- Cấu hình ZedGraph: Thiết lập các thuộc tính cho biểu đồ như tên trục (Time, Speed), đơn vị, giới hạn trục Y (Scale Min/Max), màu sắc đường nét (Setpoint màu xanh, Tốc độ thực màu đỏ) và độ dày nét vẽ.

## **Giai đoạn 2: Lắng nghe sự kiện (Event Listener)**

Chương trình duy trì trạng thái chờ. Bất kỳ hành động nào cũng sẽ kích hoạt một hàm xử lý sự kiện (Event Handler) tương ứng:

### **1. Sự kiện Kết nối (Connection Event)**

- Nếu cổng đang đóng: Thiết lập Baudrate (115200), Parity, StopBits và thực hiện lệnh Open(). Sau đó cập nhật trạng thái giao diện sang “Connected”.
- Khi người dùng nhấn nút “Connect”, chương trình kiểm tra trạng thái cổng COM.
- Nếu cổng đang mở: Thực hiện lệnh Close() để ngắt kết nối an toàn.

### **2. Sự kiện Điều khiển (Control Event):**

- Khi người dùng nhấn nút “Set”, chương trình đọc toàn bộ giá trị từ các ô nhập liệu (TextBox).
- Dữ liệu được đóng gói (Packing) thành một chuỗi ký tự theo giao thức đã định nghĩa (Ví dụ: 1,0,0,Kp,Ki,Setpoint...R).
- Chuỗi lệnh được gửi xuống vi điều khiển thông qua phương thức SerialPort.WriteLine().

### **3. Sự kiện Nhận dữ liệu (Data Reception Event):**

- Đây là sự kiện bắt đồng bộ, được kích hoạt tự động khi bộ đệm máy tính nhận được dữ liệu từ MCU.
- Xử lý đa luồng: Dữ liệu được đọc và xử lý trên một luồng nền (Background Thread).
- Tách dữ liệu (Parsing): Chuỗi nhận được (VD: “100,98.5,45”) được tách thành các giá trị số thực riêng biệt: Tốc độ đặt, Tốc độ thực, Duty Cycle.

- Cập nhật giao diện: Sử dụng cơ chế Invoke để chuyển dữ liệu từ luồng nền về luồng giao diện chính (UI Thread), thêm điểm mới vào đồ thị và ra lệnh vẽ lại (Invalidate) để tạo hiệu ứng đồ thị trôi theo thời gian thực.

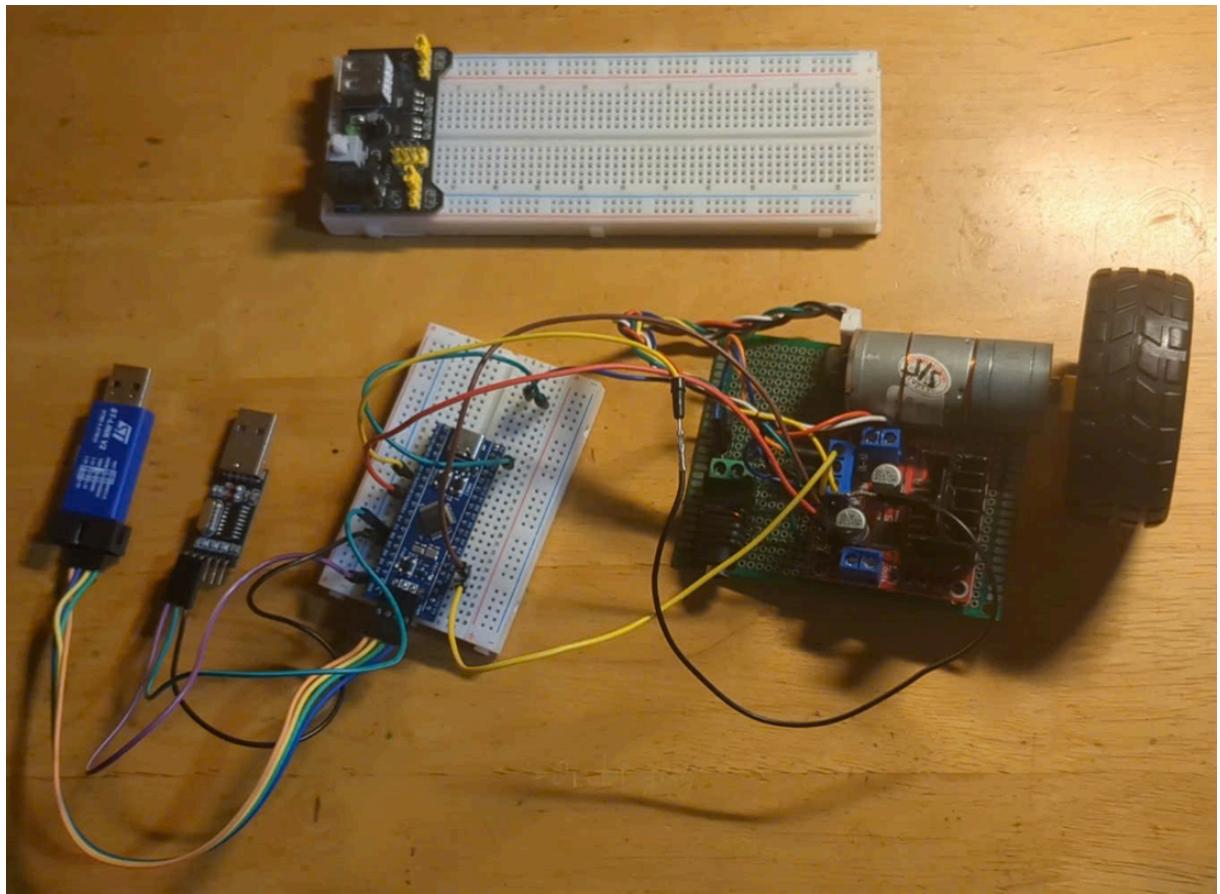
## 4.5. Thiết kế mẫu thử và đo đặc

Sau khi hoàn tất thiết kế lý thuyết và mô phỏng, nhóm tiến hành xây dựng mẫu thử nghiệm (Prototype) trên Testboard (Breadboard) để kiểm chứng nguyên lý hoạt động và độ tin cậy của phần mềm trước khi thi công mạch in (PCB).

### 4.5.1. Cấu hình phần cứng mẫu thử

Mẫu thử được xây dựng dựa trên các linh kiện chính đã lựa chọn ở phần thiết kế, bao gồm:

1. Khối điều khiển trung tâm: Kit phát triển STM32F103C8T6 (Blue Pill). Đây là module chính thực thi thuật toán PID và giao tiếp dữ liệu.
2. Khối công suất (Driver): Module L298N (hoặc TB6612FNG tùy thực tế nhóm bạn). Module này đóng vai trò mạch cầu H, nhận tín hiệu PWM 3.3V từ MCU và đóng cát nguồn động lực 12V cho động cơ.
3. Khối chấp hành: Động cơ DC Gear Motor (loại GA25 hoặc tương đương) tích hợp hộp số giảm tốc và Encoder từ tính (Hall Effect Encoder) 2 kênh A/B để phản hồi vị trí/tốc độ.
4. Nguồn cấp: Sử dụng Adapter 12V-2A cấp nguồn động lực. Nguồn cho MCU được hạ áp thông qua cổng USB hoặc mạch hạ áp 5V tích hợp.
5. Giao tiếp máy tính: Module chuyển đổi USB-to-TTL (CP2102) để tạo cổng COM ảo kết nối với phần mềm giám sát C#



Hình 4.25: Mẫu thử thực tế

#### 4.5.2. Quy trình đấu nối và sơ đồ lắp ráp

Để đảm bảo an toàn và giảm thiểu nhiễu trong quá trình thử nghiệm, quy trình đấu nối được thực hiện theo các bước sau:

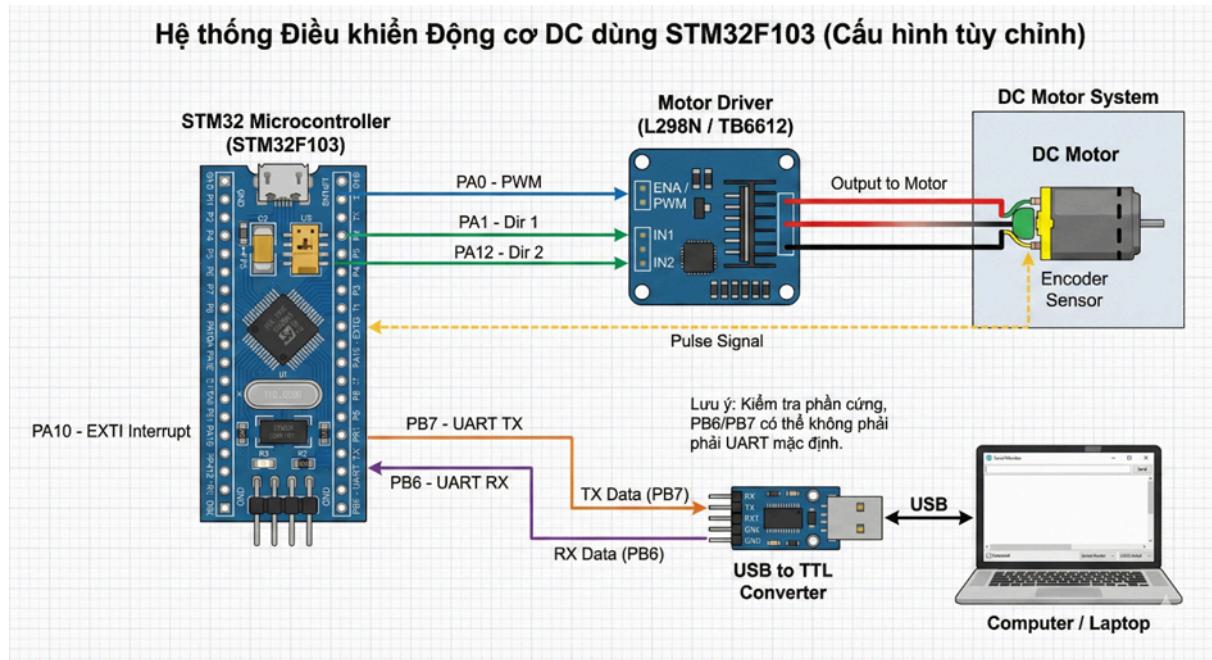
##### a. Phân tách hệ thống nguồn (Power Distribution):

- Hệ thống sử dụng chung cực âm (Common Ground - GND) giữa nguồn động lực 12V và nguồn điều khiển 5V/3.3V. Đây là yêu cầu bắt buộc để tín hiệu điều khiển có giá trị tham chiếu đúng.
- Nguồn 12V được cấp trực tiếp vào chân 12V và GND của Driver L298N.
- Nguồn 3.3V từ STM32 được cấp cho Encoder của động cơ.

##### b. Kết nối tín hiệu điều khiển (Control Signals):

- Dựa trên cấu hình Firmware (main.c), các chân tín hiệu được nối như sau:
- PWM: Chân PA0 (STM32) → Chân ENA (Driver).

- Chiều quay: Chân PA1, PA12 (STM32) → Chân IN1, IN2 (Driver).
- Phản hồi Encoder: Chân PA10 (STM32) → Chân Phase A (Encoder). Chân Phase B được nối vào một chân ngắt khác hoặc để trống nếu chỉ đo tốc độ đơn giản (tuy nhiên trong thiết kế tối ưu nên nối cả 2 pha vào Timer Encoder Mode).



Hình 4.26: Sơ đồ đấu nối mẫu thử

#### 4.5.3. Phương pháp đo đặc và thu thập dữ liệu

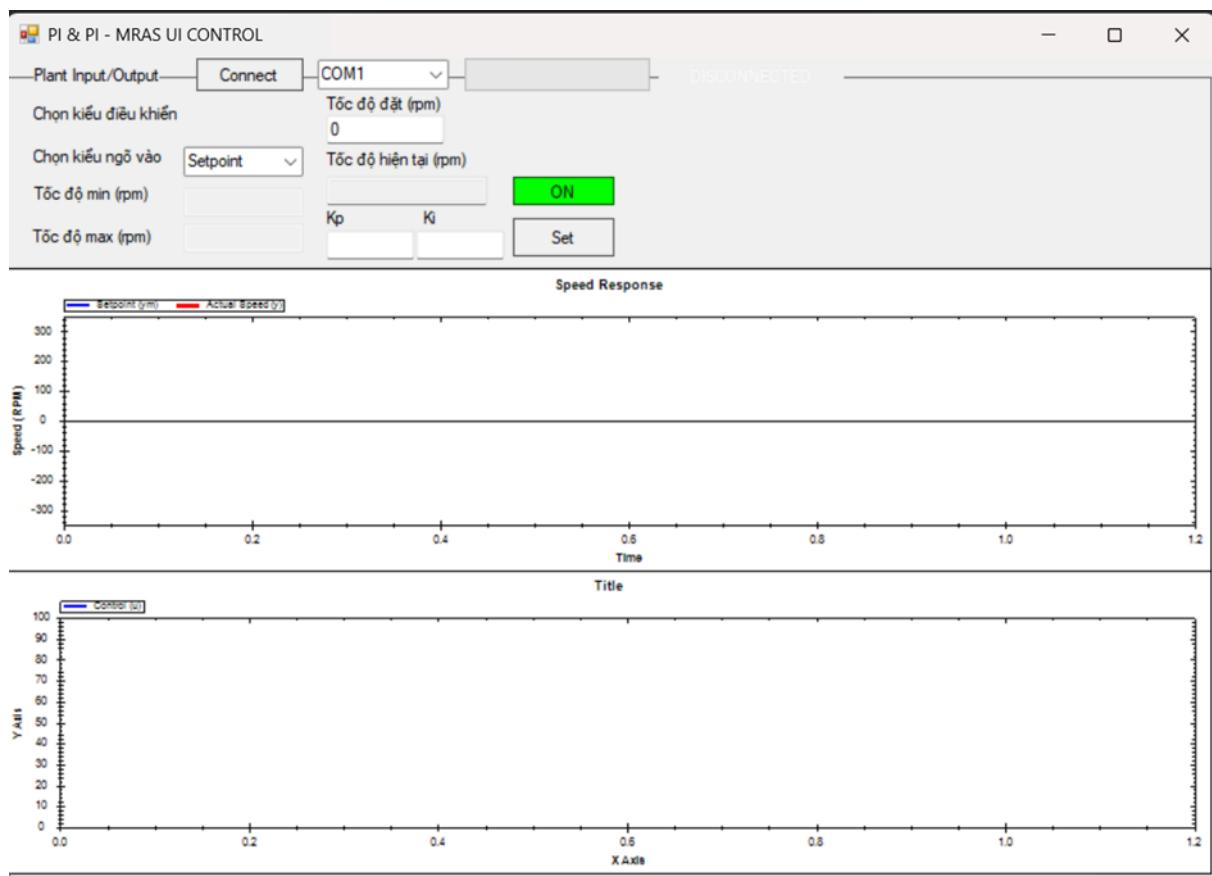
Để kiểm chứng hoạt động của hệ thống, nhóm không sử dụng các thiết bị đo cơ học bên ngoài (như máy đo tốc độ cầm tay) vì độ trễ cao, mà sử dụng phương pháp Đo lường từ xa (Telemetry) thông qua chính vi điều khiển và phần mềm máy tính. Quy trình thiết lập đo đặc:

1. Thiết lập lấy mẫu (Sampling Setup):
  - Vi điều khiển được lập trình để đếm số xung Encoder trong chu kỳ  $T_s = 100\text{ms}$ .
  - Tốc độ tức thời (RPM) được tính toán ngay trong ngắt Timer.
2. Truyền dữ liệu (Data Transmission):
  - Sử dụng giao thức UART với tốc độ Baud 115200 bps.
  - Gói tin chứa dữ liệu tốc độ thực và tốc độ đặt được gửi lên máy tính với tần số 4Hz (250ms/lần) để đảm bảo không gây nghẽn băng thông xử lý của luồng PInh

### 3. Trực quan hóa (Visualization):

- Sử dụng phần mềm C# (đã thiết kế ở mục 4.3) đóng vai trò như một “Oscilloscope số”.
- Phần mềm sẽ vẽ lại các điểm dữ liệu nhận được lên biểu đồ theo trực thời gian thực.
- Công cụ kiểm tra tín hiệu: Ngoài ra, để kiểm tra chất lượng xung PWM và độ sạch của tín hiệu Encoder, nhóm sử dụng Oscilloscope (nếu có) kẹp vào chân PA0 và chân Phase A của Encoder để quan sát dạng sóng của tín hiệu

Cách tiếp cận đo đạc này cho phép thu thập dữ liệu liên tục trong thời gian dài, giúp dễ dàng quan sát được các đặc tính quá độ (transient response) như độ vọt lố và thời gian xác lập mà mắt thường không thể nhìn thấy được.



Hình 4.27: Giao diện người dùng

## Kết quả và thảo luận

Sau quá trình thiết kế và thi công mẫu thử, nhóm đã tiến hành các kịch bản thử nghiệm để đánh giá khả năng hoạt động của toàn hệ thống. Kết quả được ghi nhận trực tiếp thông qua phần mềm giám sát trên máy tính và quan sát thực tế.

### 5.1. Kết quả tổng quát

Trước khi đi vào đánh giá chi tiết thuật toán, các khái niệm cơ bản của hệ thống đã được kiểm tra hoạt động (Functional Testing):

#### 1. Khối Nguồn:

- Hệ thống hoạt động ổn định với nguồn đầu vào 12V DC.
  - Điện áp 5V cấp cho MCU và Encoder ổn định, không bị sụt áp quá mức khi động cơ khởi động (đã kiểm tra bằng đồng hồ VOM).
  - Đèn LED báo trạng thái trên mạch hoạt động đúng thiết kế.

#### 2. Khối PWM và Driver:

- Xung PWM được tạo ra từ Timer 2 của STM32 hoạt động chính xác với tần số 1 kHz.
- Driver động cơ điều khiển được chiều quay (Thuận/Nghịch) và thay đổi được điện áp trung bình cấp cho động cơ tương ứng với độ rộng xung (Duty Cycle).

#### 3. Khối Động cơ và Encoder:

- Động cơ quay êm, không có tiếng rít bất thường ở tần số PWM 1 kHz.
- Encoder đọc được xung phản hồi, MCU đếm xung và tính toán ra vận tốc (RPM) chính xác (đã đổi chiều thủ công bằng cách đếm số vòng quay trong 1 phút ở tốc độ thấp).

#### 4. Giao tiếp UART:

- Kết nối giữa STM32 và phần mềm CC# qua cổng COM ảo (USB-TTL) ổn định ở Baudrate 115200.
- Dữ liệu gửi lên (Telemetry) liên tục, không bị mất gói tin.
- Lệnh điều khiển từ máy tính (Set Kp, Ki, Setpoint) được MCU nhận và phản hồi tức thì.

## 5.2. Hiệu suất động cơ (Open-Loop Test)

Nhóm đã thử nghiệm chạy động cơ ở chế độ vòng hở (không có PID) để xác định đặc tính tuyến tính của hệ thống.

- Vùng chết (Dead-zone): Động cơ bắt đầu quay khi Duty Cycle đạt khoảng 15-20%. Dưới ngưỡng này, ma sát tĩnh của hộp số lớn hơn mô-men khởi động nên động cơ không quay nhưng có tiếng rung nhẹ.
- Tốc độ tối đa: Tại Duty Cycle 100% (nguồn 12V), động cơ đạt tốc độ khoảng X RPM (ví dụ: 300 RPM).
- Độ tuyến tính: Tốc độ động cơ tăng tương đối tuyến tính theo độ rộng xung trong khoảng hoạt động từ 20% đến 100%.

## 5.3. Kiểm tra hiệu suất thuật toán (Closed-Loop PI Test)

Đây là phần quan trọng nhất, đánh giá chất lượng của bộ điều khiển PI được cài đặt trên STM32. Kịch bản thử nghiệm: Đặt tốc độ mục tiêu (Setpoint) từ 0 lên 150 RPM (Step Input) và quan sát đáp ứng trên đồ thị CC#. Kết quả phân tích đáp ứng quá độ:

1. Thời gian đáp ứng (Rise Time): Là thời gian để tốc độ tăng từ 10% lên 90% giá trị đặt. Kết quả đo được khoảng 0.4s - 0.6s. Hệ thống phản hồi nhanh, độ trễ thấp.
2. Độ vọt lố (Overshoot): Tốc độ thực tế vượt quá giá trị đặt khoảng 5% - 10% tại đỉnh đầu tiên trước khi ổn định lại. Mức vọt lố này nằm trong giới hạn cho phép, đảm bảo động cơ không bị giật quá mạnh.
3. Sai số xác lập (Steady-state Error): Sau khi ổn định, tốc độ thực tế dao động quanh giá trị đặt trong khoảng 2-5 RPM.
4. Khả năng bù tải: Khi dùng tay giữ nhẹ trực động cơ (tạo tải giả lập), tốc độ bị giảm xuống nhất thời, nhưng bộ điều khiển PID lập tức tăng Duty Cycle (quan sát được trên đồ thị 2 của phần mềm) để kéo tốc độ trở lại giá trị đặt. Đây là ưu điểm lớn nhất so với điều khiển vòng hở.

## 5.4. Các khó khăn và vấn đề gặp phải

Trong quá trình thực hiện, nhóm đã gặp phải và xử lý một số vấn đề kỹ thuật:

**Vấn đề nhiệt độ:** IC Driver L298N tỏa nhiệt khá lớn khi động cơ hoạt động ở tải cao trong thời gian dài do hiệu suất thấp (sụt áp qua BJT lớn).

**Độ phân giải thấp ở tốc độ chậm:** Do Encoder có độ phân giải thấp (11 xung/vòng xỉ số truyền), khi chạy ở tốc độ rất thấp (< 30 RPM), số lượng xung đếm được trong chu kỳ lấy mẫu 100ms rất ít (chỉ vài xung). Điều này làm cho phép tính vận tốc bị “răng cưa” (bước nhảy lớn), khiến bộ điều khiển PI hoạt động kém mượt mà.

## 5.5. Thảo luận

### Những điểm làm tốt (Ưu điểm)

- Hệ thống tích hợp hoàn chỉnh: Đã xây dựng thành công một vòng khép kín từ Phần cứng -> Firmware -> Software.
- Giao diện trực quan: Phần mềm C# hoạt động mượt mà, tính năng vẽ đồ thị Real-time giúp việc tinh chỉnh thông số PID (Kp, Ki) trở nên rất dễ dàng và trực quan, không cần phải nạp lại code nhiều lần.
- Thuật toán ổn định: Bộ điều khiển PID đã triệt tiêu được sai số xác lập, duy trì tốc độ ổn định ngay cả khi có biến động tải nhỏ.

### Những điểm cần cải tiến (Nhược điểm & Hướng khắc phục)

- Cải thiện thuật toán đo tốc độ: Ở dải tốc độ thấp, thay vì đếm số xung trong một khoảng thời gian cố định (Frequency measurement), nên chuyển sang đo khoảng thời gian giữa các xung (Period measurement) để tăng độ chính xác.
- Xử lý nhiễu: Cần bổ sung bộ lọc số (Digital Filter), ví dụ như bộ lọc trung bình trượt (Moving Average) hoặc bộ lọc thấp (Low-pass filter) cho tín hiệu vận tốc trước khi đưa vào bộ điều khiển PID.
- Nâng cấp phần cứng: Thay thế L298N bằng các Driver MOSFET hiện đại hơn (như TB6612, DRV8833) để giảm nhiệt và tăng hiệu suất. Sử dụng Encoder quang có độ phân giải cao hơn (ví dụ 100-400 xung/vòng) để điều khiển chính xác hơn.

## Kết Luận

Dựa trên những phân tích và thảo luận tại mục 5.5, nhóm đề xuất các giải pháp cải tiến cụ thể để nâng cao chất lượng hệ thống trong các phiên bản kế tiếp. Về mặt thuật toán, để khắc phục hiện tượng nhiễu tín hiệu Encoder và dao động ở tốc độ thấp, cần tích hợp thêm bộ lọc số (Digital Filter) như bộ lọc trung bình trượt (Moving Average) hoặc bộ lọc thông thấp (Low-pass Filter) vào firmware trước khi đưa dữ liệu vào vòng lặp PID. Bên cạnh đó, thay vì sử dụng phương pháp đếm xung đơn thuần trong khoảng thời gian cố định (Frequency Measurement), hệ thống nên chuyển sang phương pháp đo khoảng thời gian giữa các xung (Period Measurement) để tăng độ phân giải và độ mượt mà khi động cơ vận hành ở dải tốc độ thấp. Về phần cứng, việc thay thế Driver L298N bằng các dòng Driver MOSFET hiệu suất cao (như TB6612, DRV8833) và sử dụng Encoder quang có độ phân giải cao hơn sẽ giải quyết triệt để vấn đề nhiệt độ và sai số lượng tử hóa.

Về hướng phát triển trong tương lai, đề tài mở ra nhiều tiềm năng ứng dụng và nâng cấp công nghệ. Nhóm định hướng mở rộng bài toán từ điều khiển tốc độ (Velocity Control) sang điều khiển vị trí (Position Control), biến động cơ DC thông thường thành một Servo Motor để ứng dụng trong các cánh tay robot hoặc máy CNC. Ngoài ra, hệ thống có thể được nâng cấp khả năng kết nối không dây bằng cách tích hợp các module WiFi/Bluetooth (như ESP32), cho phép giám sát và điều khiển từ xa qua ứng dụng di động hoặc Web Server, bắt kịp xu hướng Internet vạn vật (IoT). Các thuật toán điều khiển hiện đại hơn như PID thích nghi (Adaptive PID) hoặc Fuzzy Logic cũng sẽ được nghiên cứu để giúp hệ thống tự động nhận dạng và thích ứng với sự thay đổi của tải trọng mà không cần tinh chỉnh lại tham số thủ công.

Tổng kết lại, đồ án “Thiết kế và Chế tạo Hệ thống Điều khiển Tốc độ Động cơ DC” đã hoàn thành tốt các mục tiêu đề ra ban đầu. Nhóm đã thực hiện trọn vẹn quy trình kỹ thuật từ khâu xác định yêu cầu hệ thống, thiết kế nguyên lý, thi công mạch in (PCB), đến lập trình Firmware cho vi điều khiển STM32 và xây dựng phần mềm giám sát C # trên máy tính. Hệ thống thực tế hoạt động ổn định, đáp ứng nhanh với các thay đổi tốc độ đặt và duy trì sai số xác lập ở mức thấp nhờ bộ điều khiển PID. Kết quả của đồ án không chỉ tạo ra một mô hình thực nghiệm hoàn chỉnh mà còn giúp nhóm cung cấp kiến thức về lý thuyết điều khiển tự động, kỹ thuật vi xử lý và kỹ năng thiết kế hệ thống nhúng thực.

