ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

BÁO CÁO BÀI TẬP LỚN
HỆ THỐNG ĐIỀU KHIỂN NHÚNG (EE3067)

Điều khiển tốc độ động cơ DC

Giảng viên hướng dẫn: TS. Nguyễn Vĩnh Hảo

Nhóm 6

STT	Họ và tên	MSSV	Đóng góp
1	Võ Quế Long	2211910	100%
2	Nguyễn Ngọc Quỳnh Nhi	2212438	100%
3	Lê Gia Uy	2213895	100%
4	Nguyễn Vinh	2213968	100%

Thành phố Hồ Chí Minh, tháng 12 năm 2025

Mục lục

1 Tổng quan đề tài

1.1 Đặt vấn đề

Trong nền công nghiệp hiện đại và tự động hóa, động cơ điện một chiều (DC Motor) đóng vai trò vô cùng quan trọng, xuất hiện phổ biến trong các hệ thống robot, dây chuyền băng tải, và các thiết bị cơ điện tử. Yêu cầu đặt ra cho các hệ thống này không chỉ là làm cho động cơ quay, mà còn phải điều khiển chính xác tốc độ, đảm bảo độ ổn định và có khả năng giám sát trạng thái hoạt động theo thời gian thực.

Việc điều khiển động cơ bằng vi điều khiển (Microcontroller) là giải pháp tối ưu về chi phí và hiệu năng. Dòng vi điều khiển STM32F4 (ARM Cortex-M4) với khả năng xử lý mạnh mẽ, tích hợp nhiều ngoại vi cao cấp như Timer, UART và đặc biệt là bộ truy cập bộ nhớ trực tiếp (DMA), cho phép thực hiện các tác vụ điều khiển phức tạp với độ chính xác cao.

Tuy nhiên, thách thức lớn trong các hệ thống nhúng điều khiển là vấn đề giao tiếp dữ liệu. Khi hệ thống cần gửi/nhận lượng lớn dữ liệu giám sát liên tục (tốc độ, dòng điện, tham số PID) về máy tính, việc sử dụng phương thức ngắt (Interrupt) truyền thống có thể gây tiêu tốn tài nguyên CPU, làm ảnh hưởng đến chu kỳ lấy mẫu và độ ổn định của thuật toán điều khiển động cơ.

Xuất phát từ thực tế đó, nhóm thực hiện đề tài "Thiết kế hệ thống điều khiển tốc độ động cơ DC sử dụng STM32F407 với giao tiếp UART/DMA và giao diện giám sát trên máy tính". Đề tài tập trung vào việc ứng dụng kỹ thuật DMA để tối ưu hóa luồng dữ liệu, đồng thời xây dựng giao diện người dùng (GUI) để quan sát đáp ứng của hệ thống một cách trực quan.

1.2 Mục tiêu đề tài

Đề tài hướng tới việc giải quyết các bài toán kỹ thuật sau:

1. **Về phần cứng:** Kết nối và điều khiển thành công động cơ DC có gắn Encoder sử dụng board mạch STM32F407 và mạch công suất (cầu H) L298N.

2. **Về thuật toán điều khiển:**
   - Tạo xung PWM để điều khiển tốc độ động cơ.
   - Có khả năng thay đổi linh hoạt tần số PWM (Frequency) và độ rộng xung (Duty Cycle) để khảo sát ảnh hưởng của tần số đến hoạt động của động cơ.
   - Đọc tính hiệu từ Encoder để tính toán tốc độ thực tế.

3. **Về giao tiếp dữ liệu:**
   - Thiết lập giao tiếp UART giữa STM32 và máy tính.
   - Ứng dụng **DMA (Direct Memory Access)** để truyền nhận dữ liệu tốc độ cao mà không làm gián đoạn CPU, đảm bảo tính thời gian thực (Real-time).

4. **Về giao diện giám sát:** Xây dựng phần mềm GUI trên máy tính cho phép:
   - Cài đặt tốc độ mong muốn (Set Speed) và tần số PWM.
   - Hiển thị đồ thị đáp ứng tốc độ (Response Plot) theo thời gian thực.

1.3 Đối tượng và phạm vi nghiên cứu

- **Đối tượng nghiên cứu:**
  - Vi điều khiển STM32F407VGT6.
  - Động cơ DC Servo (có Encoder).
  - Giao thức truyền thông nối tiếp UART và cơ chế DMA.
  - Phương pháp điều chế độ rộng xung (PWM).
- **Phạm vi nghiên cứu:**
  - Hệ thống tập trung vào điều khiển tốc độ (Speed Control), chưa đi sâu vào điều khiển vị trí (Position Control).
  - Giao diện GUI được thiết kế trên nền tảng Windows.
  - Nghiên cứu ảnh hưởng của việc thay đổi tần số PWM và tham số điều khiển đến đáp ứng đầu ra.

1.4 Phương pháp nghiên cứu

• **Phương pháp lý thuyết:** Nghiên cứu tài liệu Datasheet của STM32F4 Reference Manual, lý thuyết về điều khiển tự động và xử lý tín hiệu số.

• **Phương pháp thực nghiệm:**
  – Mô phỏng giải thuật.
  – Lắp ráp mô hình phần cứng thực tế.
  – Viết code Firmware trên Keil C hoặc STM32CubeIDE và phần mềm GUI.
  – Tiến hành đo đạc, thu thập dữ liệu đáp ứng và tinh chỉnh thông số.

1.5 Bố cục báo cáo

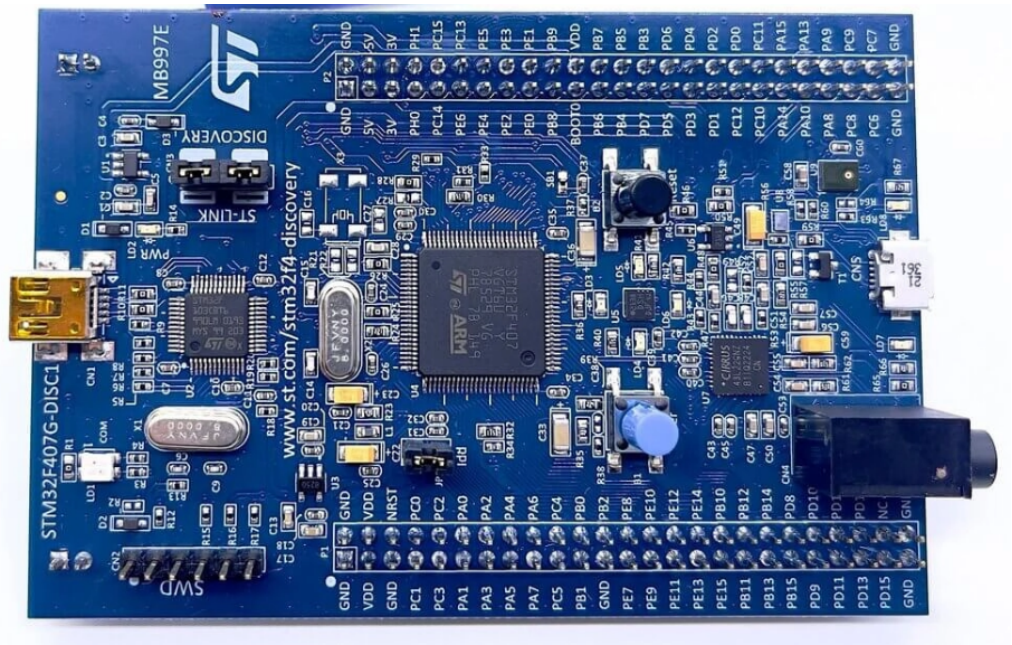
Báo cáo được chia thành 5 phần chính:

1. Tổng quan đề tài.
2. Cơ sở lý thuyết (Giới thiệu STM32, PWM, UART, DMA, Encoder).
3. Thiết kế hệ thống (Sơ đồ khối, sơ đồ nguyên lý, lưu đồ thuật toán, thiết kế giao thức truyền tin).
4. Kết quả thực hiện và Đánh giá (Hình ảnh mô hình, giao diện GUI, đồ thị)
5. Kết luận và hướng phát triển.

2 Cơ sở lý thuyết

2.1 KIT STM32F407 Discovery

Kit phát triển **STM32F4 Discovery** được thiết kế dựa trên dòng vi điều khiển hiệu năng cao STM32F407, cho phép người dùng dễ dàng phát triển các ứng dụng xử lý âm thanh. Kit tích hợp **công cụ debug ST-LINK/V2-A**, một cảm biến gia tốc kỹ thuật số MEMS, một microphone kỹ thuật số, một bộ DAC âm thanh kèm driver loa Class D, các đèn LED, nút nhấn và cổng USB OTG Micro-AB.



Hình 1: KIT STM32F407 Discovery.

Ngoài ra, kit hỗ trợ kết nối các bo mở rộng chuyên dụng thông qua các chân header mở rộng.

Các tính năng của board:

- **Vi điều khiển STM32F407VGT6**:
  - Lõi xử lý Arm Cortex-M4 32-bit với FPU.
  - 1 MB Flash.
  - 192 KB RAM.

  - Đóng gói LQFP100.

- USB OTG Full-Speed.
- Cảm biến gia tốc MEMS 3 trục.
- Cảm biến âm thanh MEMS (microphone số đa hướng).
- Bộ DAC âm thanh với driver loa Class D tích hợp.
- Nút nhấn người dùng và nút reset.
- **Hệ thống đèn LED**:
  - LD1 (đỏ/xanh lá): trạng thái giao tiếp USB.
  - LD2 (đỏ): báo nguồn 3.3 V.
  - Bốn LED người dùng: LD3 (cam), LD4 (xanh), LD5 (đỏ), LD6 (xanh dương).
  - Hai LED USB OTG: LD7 (xanh, VBUS), LD8 (đỏ, quá dòng).
- **Các cổng kết nối**:
  - USB Micro-AB.
  - Jack xuất âm thanh stereo (tai nghe).
  - Header mở rộng chuẩn 2.54 mm, cung cấp toàn bộ chân I/O của LQFP100.
- **Nguồn cấp linh hoạt**:
  - USB từ ST-LINK.
  - Cổng USB ngoài.
  - Nguồn ngoài (3 V hoặc 5 V).
- **Phần mềm hỗ trợ miễn phí**:

  - Ví dụ mẫu trong gói STM32CubeF4 MCU Package.
  - STSW-STM32068 sử dụng thư viện chuẩn (legacy).

- **Debugger/Programmer ST-LINK/V2-A**:
  - Mass storage.
  - Virtual COM port.
  - Debug port.

- **Hỗ trợ nhiều IDE**:
  - IAR Embedded Workbench.
  - MDK-ARM.
  - STM32CubeIDE.

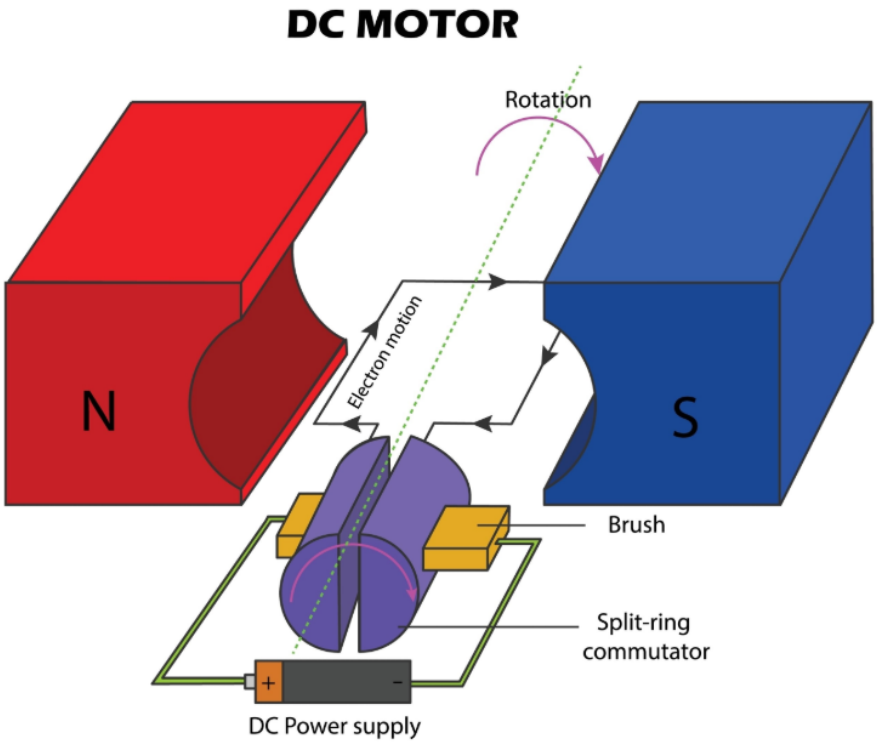
2.2 Động cơ DC và cảm biến Encoder

2.2.1 Động cơ điện 1 chiều (DC motor)

Động cơ DC là cơ cấu chấp hành biến đổi điện năng thành cơ năng dựa trên hiện tượng cảm ứng điện từ. Cấu tạo cơ bản gồm hai phần:

- **Stator (Phần tĩnh):** Thường là nam châm vĩnh cửu tạo ra từ trường.
- **Rotor (Phần quay):** Các cuộn dây quấn quanh lõi sắt từ. Khi có dòng điện chạy qua cuộn dây đặt trong từ trường, lực từ (Lực Lorentz) sẽ sinh ra ngẫu lực làm quay rotor.

Vận tốc quay của động cơ tỉ lệ thuận với điện áp đặt vào hai đầu cực. Do đó, để điều khiển tốc độ, ta sử dụng phương pháp thay đổi điện áp trung bình (thông qua PWM).



Hình 2: Động cơ DC (DC motor).

2.2.2 Cảm biến tốc độ (Incremental Encoder)

Để thực hiện điều khiển vòng kín (Closed-loop), hệ thống sử dụng **Encoder tương đối (Incremental Encoder)** gắn ở trục động cơ.

• **Nguyên lý:** Encoder bao gồm một đĩa quay có các rãnh và hệ thống thu phát quang (LED và Phototransistor). Khi đĩa quay, ánh sáng bị cắt ngắt quãng tạo ra các chuỗi xung vuông.

• **Xác định chiều quay:** Encoder xuất ra hai kênh tín hiệu A và B lệch pha nhau 90 độ điện (Quadrature output).

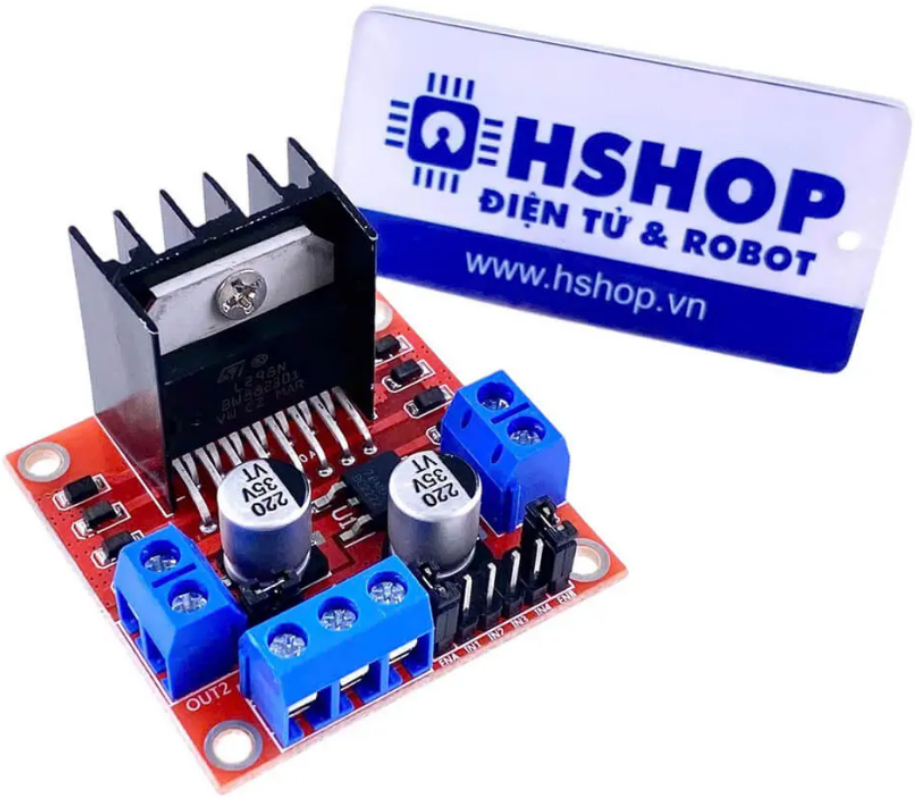
– Nếu xung A sớm pha hơn B → Động cơ quay thuận.

– Nếu xung A trễ pha hơn B → Động cơ quay nghịch.

• **Đo tốc độ:** Vi điều khiển sẽ đếm số lượng xung (sườn lên/xuống) trong một khoảng thời gian nhất định để tính ra vận tốc (RPM).

2.3 Mạch điều khiển động cơ (H-Bridge L298N)

Do dòng điện từ chân vi điều khiển (GPIO) rất nhỏ (khoảng 20mA), không thể chạy trực tiếp động cơ, nên cần một mạch công suất trung gian. Nhóm sử dụng module L298N *(Hình 3)*.



Hình 3: Module L298N.

- **Cấu trúc:** L298N tích hợp hai mạch cầu H (H-Bridge) bên trong. Mạch cầu H cho phép đảo chiều dòng điện chạy qua động cơ, từ đó đảo chiều quay.
- **Nguyên lý hoạt động:**
  - Module có các chân điều khiển `IN1`, `IN2` và chân cho phép `ENA`.
  - Khi `IN1 = High`, `IN2 = Low` → Dòng điện chạy từ A sang B → Quay thuận.
  - Khi `IN1 = Low`, `IN2 = High` → Dòng điện chạy từ B sang A → Quay nghịch.
  - Tín hiệu PWM được cấp vào chân `ENA` để đóng ngắt dòng điện liên tục, giúp điều chỉnh tốc độ.

2.4 Các giao thức và thuật toán điều khiển

2.4.1 Kỹ thuật điều chế độ rộng xung (Pulse Width Modulation)

PWM (Pulse Width Modulation) là phương pháp thay đổi điện áp trung bình đặt lên tải bằng cách thay đổi độ rộng của chuỗi xung vuông liên tiếp. Thay vì thay đổi biên độ điện áp (điều này gây tiêu hao năng lượng trên linh kiện bán dẫn), PWM thực hiện đóng ngắt nguồn điện cấp vào động cơ với tần số cao.

Tham số quan trọng nhất của PWM là **Duty Cycle (Chu kỳ nhiệm vụ - D)**, được xác định bởi công thức:

$$D = \frac{T_{on}}{T} \times 100\% \tag{1}$$

Điện áp trung bình cấp cho động cơ sẽ tỷ lệ thuận với Duty Cycle:

$$V_{out} = V_{in} \times D \tag{2}$$

Trong đó:

- V_{out}: Điện áp trung bình (V).
- V_{in}: Điện áp nguồn cấp (V).
- D: Chu kỳ nhiệm vụ (%).

2.4.2 Giao thức UART (Universal Asynchronous Receiver - Transmitter)

UART là giao thức truyền thông nối tiếp không đồng bộ, đóng vai trò cầu nối trao đổi dữ liệu giữa vi điều khiển STM32 và máy tính (PC). Do là giao thức không đồng bộ, nó không cần dây tín hiệu xung nhịp (Clock) đi kèm, giúp tiết kiệm số lượng dây dẫn (chỉ cần 2 dây: TX để truyền và RX để nhận).

Để quá trình truyền tin thành công, cả hai thiết bị phải thống nhất các tham số sau:

• **Baud rate (Tốc độ baud):** Số bit truyền được trong một giây. Trong đề tài này, hệ thống sử dụng tốc độ 115200 bps để đảm bảo truyền lượng dữ liệu lớn nhanh chóng.

- **Khung truyền (Data Frame):** Cấu trúc của một gói dữ liệu cơ bản bao gồm:
  - **Start Bit:** 1 bit báo hiệu bắt đầu (mức thấp).
  - **Data Bits:** 8 bit dữ liệu chính.
  - **Stop Bit:** 1 bit báo hiệu kết thúc (mức cao).

Trong hệ thống này, UART nhận lệnh điều khiển từ GUI xuống STM32 và gửi ngược lại thông số tốc độ thực tế để vẽ đồ thị đáp ứng.

2.4.3 Cơ chế truy cập bộ nhớ trực tiếp (Direct Memory Access - DMA)

Truy cập bộ nhớ trực tiếp (Direct Memory Access - DMA) là một tính năng chuyên dụng của vi điều khiển, cho phép truyền dữ liệu tốc độ cao giữa ngoại vi (như UART, ADC) và bộ nhớ (RAM) hoặc giữa bộ nhớ với bộ nhớ mà không cần sự can thiệp của CPU. Đây là kỹ thuật nâng cao được áp dụng để tối ưu hóa hiệu năng của hệ thống thời gian thực (Real-time).

Thông thường, khi UART nhận được một byte dữ liệu, nó sẽ gửi yêu cầu ngắt (Interrupt) đến CPU. CPU phải tạm dừng công việc hiện tại, lưu ngữ cảnh, thực hiện hàm ngắt để sao chép dữ liệu vào RAM, sau đó mới quay lại làm việc tiếp. Nếu dữ liệu truyền liên tục với tốc độ cao, CPU sẽ bị quá tải vì phải xử lý ngắt liên tục, dẫn đến việc tính toán thuật toán điều khiển động cơ bị trễ hoặc sai lệch.

Giải pháp DMA: DMA hoạt động như một bộ điều khiển trung chuyển dữ liệu độc lập.

- **Nguyên lý:** Khi UART nhận được dữ liệu, DMA sẽ tự động "gắp" dữ liệu đó từ thanh ghi của UART và chuyển thẳng vào vùng nhớ đệm (Buffer) trong RAM mà không cần thông qua CPU.
- **Chế độ Circular (Vòng tròn):** Hệ thống sử dụng DMA ở chế độ Circular kết hợp với bộ đệm vòng. Khi bộ đệm đầy, DMA tự động quay lại ghi đè vào đầu bộ đệm. Điều này đảm bảo luồng dữ liệu từ máy tính xuống vi điều khiển luôn được tiếp nhận liên tục và không bao giờ bị mất mát, trong khi CPU hoàn toàn rảnh rỗi để tập trung xử lý thuật toán PID.

2.4.4 Thuật toán điều khiển PID

Để đảm bảo động cơ quay ổn định đúng tốc độ đặt bất chấp tải trọng thay đổi, hệ thống sử dụng bộ điều khiển hồi tiếp PID (Proportional - Integral - Derivative).

Sơ đồ khối hệ thống điều khiển như sau:

- **Input (Setpoint):** Tốc độ mong muốn cài đặt từ GUI.
- **Feedback:** Tốc độ thực tế đo được từ Encoder.
- **Error (e):** Sai số giữa tốc độ đặt và tốc độ thực ($e = Setpoint - Actual$).

Bộ điều khiển PID tính toán giá trị điều khiển $u(t)$ dựa trên sai số $e(t)$ thông qua 3 thành phần:

1. **Khâu Tỷ lệ (Proportional - K_p):** Tạo ra tín hiệu điều khiển tỷ lệ với sai số hiện tại. Giúp hệ thống phản ứng nhanh, nhưng nếu K_p quá lớn sẽ gây dao động (vọt lố).

2. **Khâu Tích phân (Integral - K_i):** Cộng dồn sai số theo thời gian. Thành phần này có vai trò triệt tiêu sai số xác lập (sai số tĩnh), giúp tốc độ thực tế bằng đúng tốc độ đặt về lâu dài.

3. **Khâu Vi phân (Derivative - K_d):** Phản ứng với tốc độ thay đổi của sai số. Giúp giảm độ vọt lố và làm hệ thống ổn định nhanh hơn khi có thay đổi đột ngột.

Công thức toán học của bộ điều khiển PID rời rạc:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{3}$$

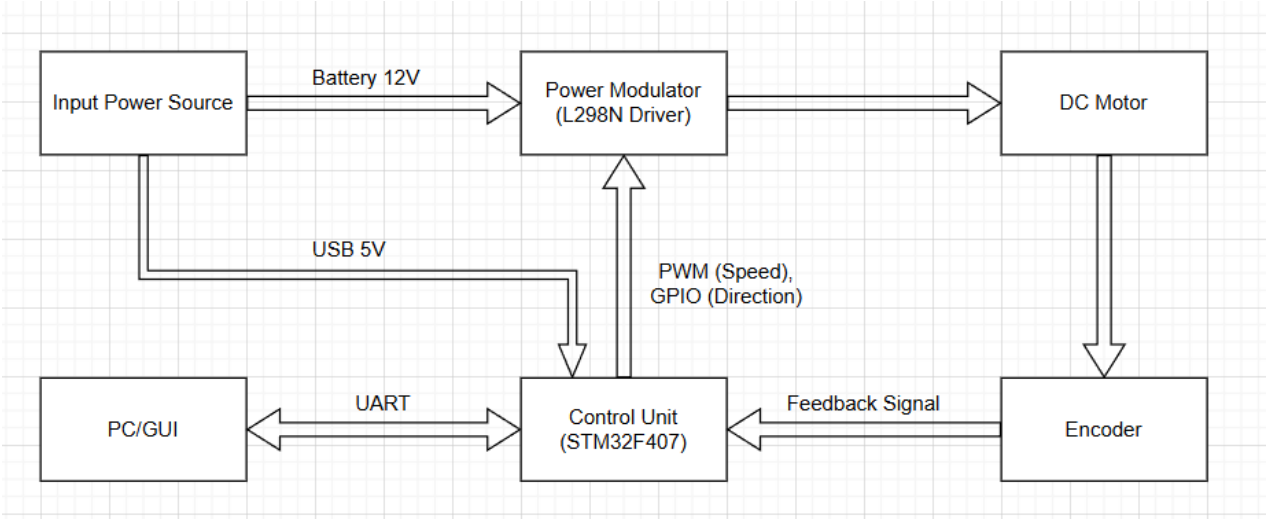
Trong đó:

- K_p: Hệ số khâu Tỷ lệ (Proportional).
- K_i: Hệ số khâu Tích phân (Integral).
- K_d: Hệ số khâu Vi phân (Derivative).
- $e(t)$: Sai số tại thời điểm t ($Setpoint - Actual$).

3 Thiết kế hệ thống

Hệ thống được thiết kế theo **mô hình điều khiển vòng kín (Closed-Loop Control System)**, bao gồm 05 khối chức năng chính tương tác với nhau qua các luồng tín hiệu và luồng năng lượng như hình 4.

3.1 Sơ đồ khối hệ thống (System Block Diagram)



Hình 4: Sơ đồ khối cho toàn bộ hệ thống (System Block Diagram).

• **Khối nguồn (Input Power Source):** Hệ thống sử dụng 02 nguồn cấp độc lập để đảm bảo an toàn và chống nhiễu:

– Nguồn USB 5V: Cung cấp năng lượng cho khối xử lý trung tâm (STM32F407) hoạt động ổn định.

– Nguồn Battery 12V: Cung cấp dòng điện lớn cho khối điều chế công suất (L298N) để vận hành động cơ.

– Hai nguồn này được nối chung cực âm (Common GND) để đồng bộ mức tham chiếu tín hiệu.

• **Khối xử lý trung tâm (Control Unit - STM32F407):** Đóng vai trò là "bộ não" của hệ thống, thực hiện các nhiệm vụ:

– Tiếp nhận lệnh điều khiển từ máy tính qua giao tiếp UART.

– Đọc tín hiệu xung từ Encoder để tính toán tốc độ thực tế.

– Thực thi thuật toán PID để tính toán sai số và xuất tín hiệu điều khiển.

• **Khối điều chế công suất (Power Modulator - L298N Driver:)** Đóng vai trò trung gian khuếch đại dòng điện.

– Nhận tín hiệu điều khiển công suất thấp từ vi điều khiển: Tín hiệu PWM (quyết định tốc độ) và tín hiệu GPIO (quyết định chiều quay của động cơ).

– Đóng ngắt nguồn 12V từ pin để cấp dòng điện tương ứng cho động cơ hoạt động.

• **Khối chấp hành (DC Motor) và Cảm biến (Encoder):**

– Động cơ DC chuyển đổi điện năng thành cơ năng.

– Encoder gắn trên trục động cơ đóng vai trò cảm biến hồi tiếp, gửi các chuỗi xung (Feedback Signal) về vi điều khiển để giám sát tốc độ.

• **Khối giao diện người dùng (PC/GUI):** Phần mềm trên máy tính cho phép người dùng nhập các tham số cài đặt và hiển thị đồ thị đáp ứng tốc độ theo thời gian thực.

• **Luồng hoạt động của hệ thống:**

1. Người dùng gửi lệnh cài đặt tốc độ từ PC xuống KIT STM32F407 Discovery qua chuẩn giao tiếp UART.

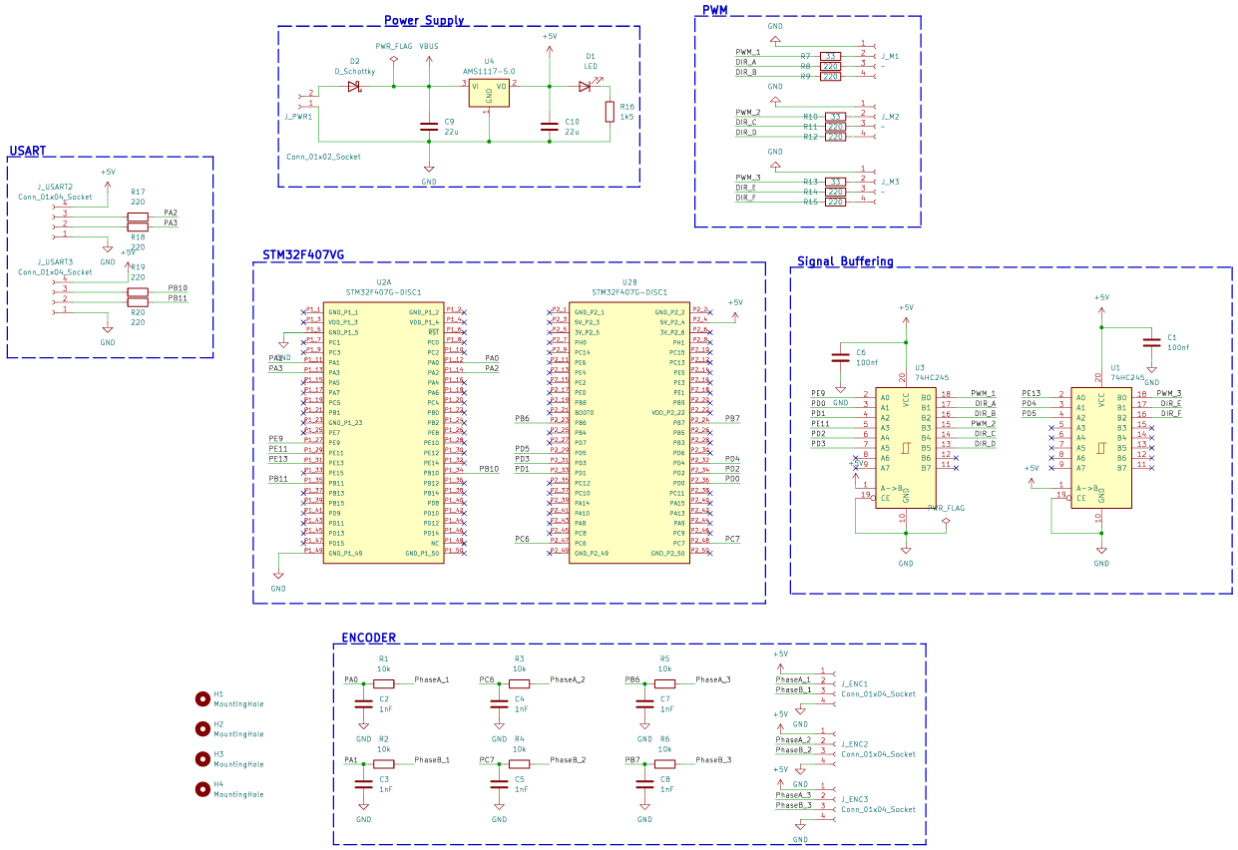
2. STM32 so sánh tốc độ đặt với tốc độ thực tế đo được từ Encoder.

3. Thuật toán PID tính toán và điều chỉnh độ rộng xung PWM xuất ra mạch L298N.

4. Mạch L298N điều khiển dòng điện vào động cơ DC để đạt tốc độ mong muốn, hình thành 1 vòng điều khiển khép kín ổn định.

3.2 Sơ đồ nguyên lý và kết nối phần cứng

3.2.1 Sơ đồ nguyên lý và mạch PCB



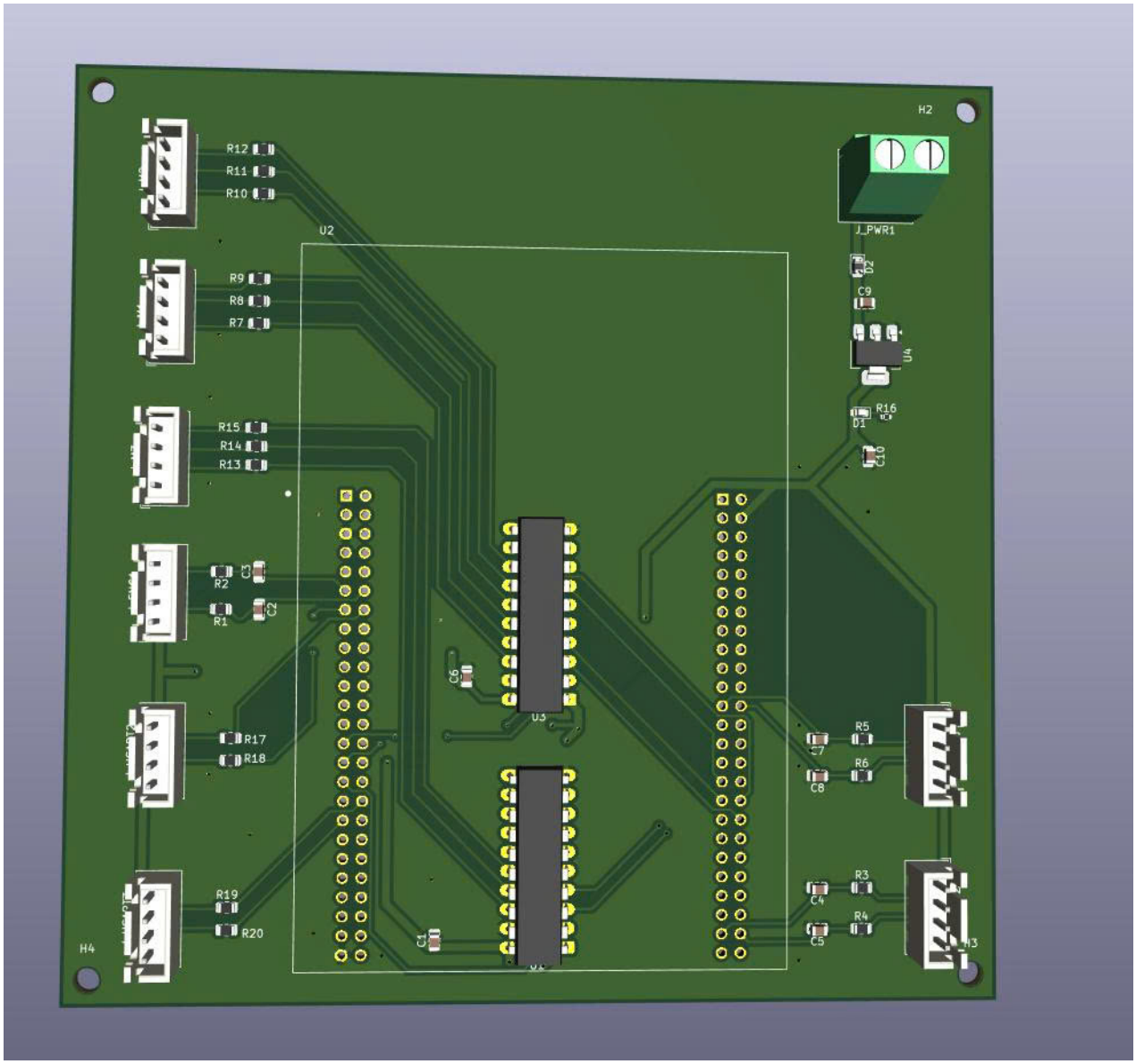
Hình 5: Sơ đồ nguyên lý mạch toàn bộ hệ thống.

Dựa trên yêu cầu điều khiển ổn định và chống nhiễu, sơ đồ nguyên lý hệ thống được thiết kế bao gồm các khối chức năng chính như Hình 5. Điểm nổi bật của thiết kế này bao gồm:

1. **Khối Signal Buffering và chuyển mức Logic (Level Shifter):** Hệ thống sử dụng IC đệm dòng 74HC245 nằm giữa vi điều khiển và mạch công suất L298N.
   - Chức năng 1: Chuyển đổi mức logic từ $3.3V$ của STM32 lên $5V$ của L298N để đảm bảo kích mở MOSFET trong cầu H bão hòa hoàn toàn.
   - Chức năng 2: Bảo vệ các chân GPIO của STM32 khỏi các xung điện áp ngược có thể sinh ra từ phía động cơ.

2. **Khối lọc nhiễu Encoder (Noise Filtering):** Tín hiệu xung pha A và pha B từ Encoder không đi trực tiếp vào vi điều khiển mà đi qua mạch lọc thụ động RC Low-pass Filter (Gồm điện trở nối tiếp và tụ điện xuống mass). Mạch này giúp loại bỏ các gai nhiễu cao tần (Glitch) sinh ra do tia lửa điện chổi than động cơ, giúp bộ đếm Timer của STM32 hoạt động chính xác.

3. **Khối nguồn (Power Supply):** Sử dụng IC ổn áp tuyến tính AMS1117-5.0 để tạo ra điện áp tham chiếu 5V ổn định cho các cảm biến và IC đệm.



Hình 6: Thiết kế mạch PCB.

3.2.2 Kết nối phần cứng

Dựa trên đặc tả kỹ thuật của KIT STM32F407 Discovery và yêu cầu đề tài, sơ đồ kết nối chân (Pin Mapping) được thiết lập như sau:

Chức năng	**Pin STM32**	**Kết nối ngoại vi**	**Mô tả**
PWM Output	**PE9** (TIM1)	L298N - ENA	Tín hiệu điều xung tốc độ.
Encoder A	**PC6** (TIM3)	Encoder Phase A	Kênh đếm xung A.
Encoder B	**PC7** (TIM3)	Encoder Phase B	Kênh đếm xung B.
UART TX	**PA2** (USART2)	USB-TTL RX	Truyền dữ liệu lên máy tính.
UART RX	**PA3** (USART2)	USB-TTL TX	Nhận dữ liệu từ máy tính.
Motor Dir 1	**PD0**	L298N - IN1	Tín hiệu điều khiển chiều quay 1.
Motor Dir 2	**PD1**	L298N - IN2	Tín hiệu điều khiển chiều quay 2.

Bảng 1: Bảng phân bố chân tín hiệu.

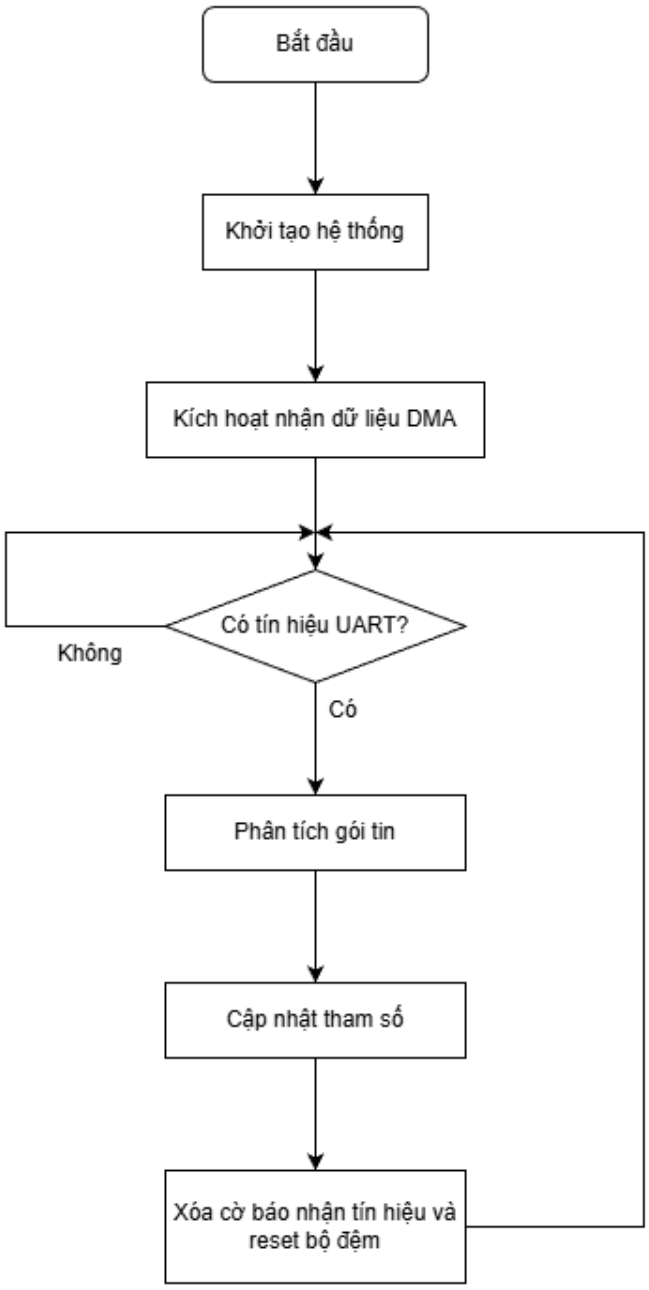
3.3 Phân bố tài nguyên (Resource Allocation)

Để hệ thống hoạt động đa nhiệm (vừa nhận dữ liệu từ GUI, vừa điều khiển động cơ, vừa đọc Encoder), vi điều khiển được cấu hình sử dụng các tài nguyên sau:

- **TIM1:** Cấu hình chế độ **PWM Generation**.
- **TIM3:** Cấu hình chế độ **Encoder Mode.**
- **TIM2:** Cấu hình **Timer Interrupt** với chu kỳ lấy mẫu $T_s = 10ms$. Đây là chu kỳ thực hiện thuật toán PID rời rạc.
- **UART2 và DMA1:** Cấu hình chế độ truyền nhận **DMA Circular.**

3.4 Lưu đồ thuật toán (Flowcharts)

3.4.1 Lưu đồ thuật toán cho chương trình chính



Hình 7: Lưu đồ thuật toán trong chương trình chính.

Lưu đồ thuật toán trong chương trình chính mô tả trình tự khởi động và chu trình xử lý sự kiện của hệ thống, được minh họa như Hình 7

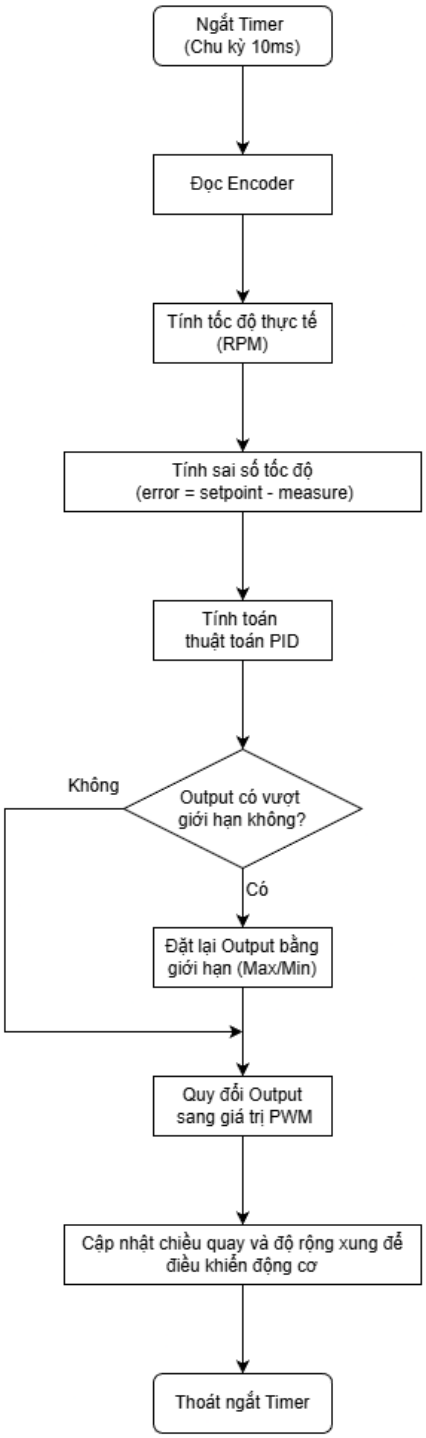
1. **Khởi tạo hệ thống:** Ngay sau khi cấp nguồn và nạp code, vi điều khiển thực hiện cấu hình xung nhịp (System Clock), khởi tạo các ngoại vi (GPIO), Timer (cho PWM và Encoder), và thiết lập thông số cho module UART.

2. **Kích hoạt DMA:** Hệ thống gọi hàm kích hoạt bộ thu UART ở chế độ **DMA (Direct Memory Access)** với cơ chế vòng tròn (Circular). Việc này đảm bảo dữ liệu đến sẽ được tự động chuyển vào bộ nhớ đệm mà không cần CPU can thiệp tức thời.

3. **Vòng lặp chính** (While(1) trong hàm main()): Hệ thống đi vào vòng lặp vô hạn để liên tục giám sát trạng thái hoạt động:
   • **Kiểm tra tín hiệu:** Vi điều khiển liên tục kiểm tra cờ báo hiệu trạng thái nhận dữ liệu `flag_uart_rx`.
   • **Nếu không có tín hiệu:** Hệ thống tiếp tục duy trì trạng thái chờ hoặc thực hiện các tác vụ nền khác, sau đó lặp lại bước kiểm tra.
   • **Nếu có tín hiệu** `flag_uart_rx == 1`: Chương trình sẽ gọi hàm xử lý ngắt `uart_rx_handler()` để phân tích gói tin nhận được. Tại đây, chuỗi dữ liệu sẽ được tách thành mã lệnh (Command) và giá trị tham số (Data) để cập nhật cho các biến điều khiển.

4. **Kết thúc chu trình xử lý:** Sau khi cập nhật tham số thành công, hệ thống tiến hành xóa cờ báo hiệu và làm sạch bộ đệm để sẵn sàng cho phiên giao tiếp tiếp theo, sau đó quay trở lại ban đầu vòng lặp.

3.4.2 Lưu đồ thuật toán điều khiển PID



Hình 8: Lưu đồ thuật toán điều khiển PID.

Lưu đồ thuật toán điều khiển PID (Hình 8) mô tả quá trình xử lý tín hiệu hồi tiếp và tính toán điều khiển, được thực hiện bên trong trình phục vụ ngắt định thời (Timer Interrupt Service Routine).

1. **Kích hoạt ngắt (Timer Interrupt):** Bộ định thời được cấu hình để tạo ra ngắt định kỳ với chu kỳ lấy mẫu cố định $T_s = 10ms$.

2. **Thu thập dữ liệu (Data Acquisition:)**
   - **Đọc Encoder:** Vi điều khiển đọc giá trị xung đếm được từ Timer Encoder.
   - **Tính tốc độ:** Dựa trên độ chênh lệch xung so với chu kỳ trước, hệ thống tính toán vận tốc tức thời của động cơ (đơn vị RPM).

3. **Tính toán điều khiển (PID Computation):**
   - **Tính sai số** (e)**:** Xác định độ lệch giữa tốc độ mong muốn (Setpoint) và tốc độ thực tế.
   - **Thuật toán PID:** Tính toán ba thành phần Tỷ lệ (P), Tích phân (I), Vi phân (D) và tổng hợp thành tín hiệu điều khiển ngõ ra (u).

4. **Khâu bão hòa (Saturation):** Kiểm tra và giới hạn tín hiệu điều khiển trong khoảng cho phép để bảo vệ hệ thống và chống hiện tượng bão hòa tích phân.

5. **Cập nhật chấp hành (Actuation Update):**
   - **Quy đổi:** Chuyển đổi giá trị điều khiển PID sang giá trị nạp vào thanh ghi PWM.
   - **Xuất tín hiệu:** Cập nhật trạng thái các chân GPIO (chiều quay của động cơ) và độ rộng xung PWM để điều khiển mạch công suất L298N.

6. **Thoát ngắt:** Kết thúc chu trình điều khiển, vi điều khiển quay lại thực hiện chương trình chính cho đến khi chu kỳ 10ms tiếp theo đến.

3.5 Thiết kế giao thức truyền thông (UART Protocol Design)

1. **Kiến trúc khung truyền (Frame Architecture)**

   Để đảm bảo tính đồng bộ và tận dụng tối đa khả năng của bộ điều khiển DMA (Direct Memory Access), hệ thống sử dụng kiến trúc **khung truyền độ dài cố định (Fixed-Length Frame)**. Mỗi gói tin gửi từ máy tính xuống vi điều khiển luôn có kích thước cố định là 30 Bytes.

   Cấu trúc gói tin bao gồm hai thành phần chính:

Trường (Field)	**Kích thước**	**Mô tả**
CMD	5 Bytes	Mã lệnh (Mnemonic). Ví dụ: `M_STR`, `M_PLT`.
DATA	25 Bytes	Giá trị tham số dạng ASCII (Zero-padded).
TỔNG	**30 Bytes**	Tổng độ dài khung truyền cố định.

Bảng 2: Cấu trúc khung truyền dữ liệu (Frame Structure.)

2. **Kỹ thuật đệm 'space' vào chuỗi**

   Do DMA được cấu hình để kích hoạt ngắt khi nhận đủ số lượng byte cố định (30 bytes), các dữ liệu có giá trị nhỏ phải được chèn thêm các ký tự `'space'` vào phía trước để lấp đầy khung truyền.

   - *Sai:* Gửi `M_STP` (Chỉ 5 bytes → DMA chưa ngắt, hệ thống treo).
   - *Đúng:* Gửi `M_STP + 25 [space]` (Đủ 30 bytes → DMA ngắt xử lý ngay).

3. **Bảng tập lệnh điều khiển (Command Table)**

   Hệ thống hỗ trợ các tập lệnh điều khiển và giám sát được liệt kê chi tiết trong bảng dưới đây:

Chức năng	**Mã lệnh**	**Ví dụ gói tin**	**Ý nghĩa**
Start	`M_STR`	`M_STRKp Ki Kd SP ...[space]`	Bắt đầu đặt Kp Ki Kd và chạy thuật toán điều khiển.
Stop	`M_STP`	`M_STP...[space]`	Dừng động cơ ngay lập tức.
Inverse	`M_INV`	`M_INV...[space]`	Đảo chiều quay động cơ.
Set Freq	`M_FRE`	`M_FRE20000...[space]`	Cài đặt tần số PWM (VD: 20kHz).

Bảng 3: Bảng tập lệnh điều khiển (Command Table).

4 Kết quả thực hiện và Đánh giá

4.1 Giao diện giám sát hệ thống (GUI)

Nhóm đã xây dựng thành công phần mềm giao diện trên máy tính (GUI) để phục vụ quá trình điều khiển và giám sát. Giao diện được thiết kế trực quan, bao gồm các khu vực chức năng chính:

- **Connection Settings:** Cài đặt thông số cổng COM và Baudrate (115200).
- **PID Controller:** Cho phép tinh chỉnh 3 tham số K_p, K_i, K_d và cài đặt tốc độ mong muốn (Setpoint) ngay khi hệ thống đang chạy (Online Tuning).
- **Graph Panel:** Đồ thị hiển thị song song hai đường đặc tính: Tốc độ đặt (Màu xanh) và Tốc độ thực tế (Màu đỏ) theo thời gian thực.

Hình 9: Giao diện phần mềm điều khiển khi ở chế độ chờ

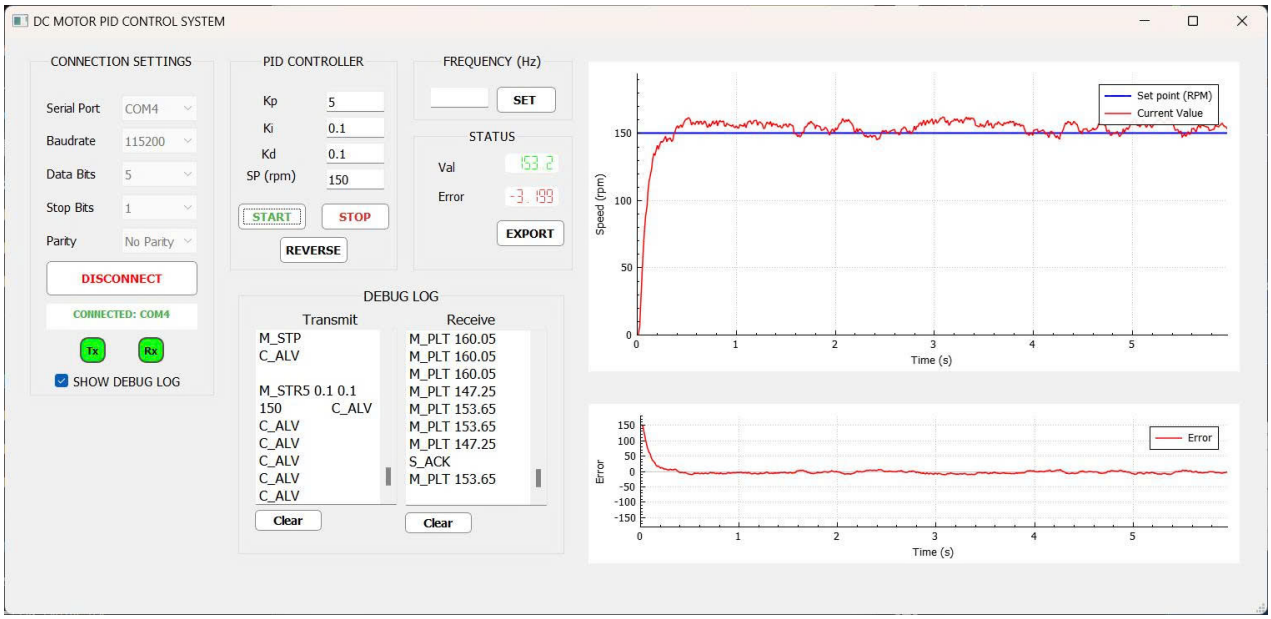
4.2 Kết quả đáp ứng tốc độ (PID Response)

Tiến hành thử nghiệm hệ thống với động cơ DC không tải. Các tham số bộ điều khiển được thiết lập ban đầu: $K_p = 5.0, K_i = 0.1, K_d = 0.1$. Tốc độ đặt là **150 RPM**. Kết quả thực nghiệm trên đồ thị (Hình 10) cho thấy:

1. **Thời gian đáp ứng:** Động cơ tăng tốc rất nhanh, đạt được tốc độ đặt chỉ sau khoảng thời gian ngắn (dưới 0.5 giây).

2. **Độ vọt lố (Overshoot):** Có xuất hiện vọt lố nhẹ (khoảng 160 RPM) do khâu K_p lớn giúp đáp ứng nhanh, nhưng sau đó nhanh chóng dao động tắt dần nhờ khâu K_d.

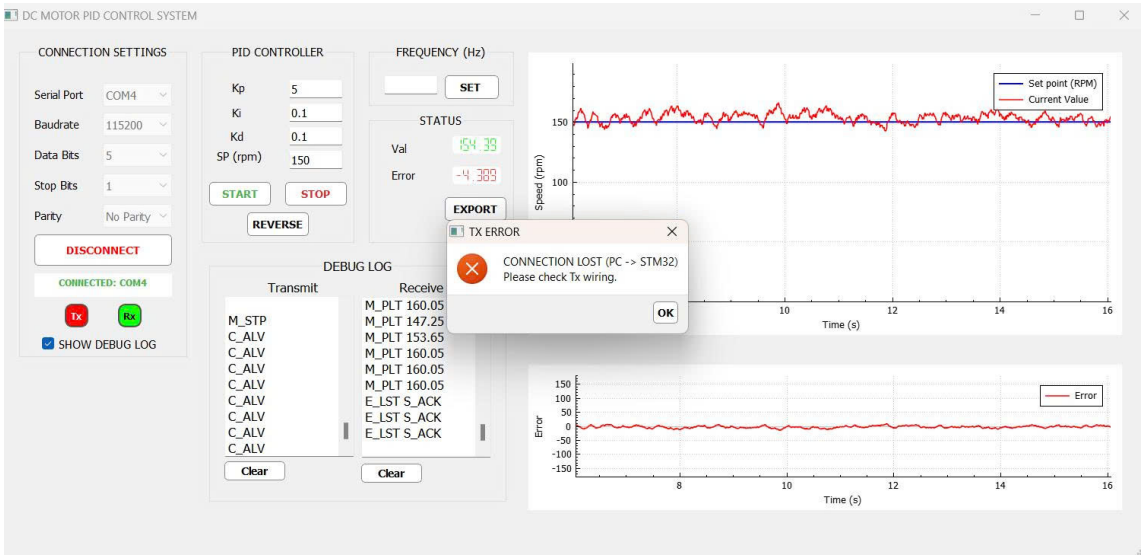
3. **Sai số xác lập:** Đường màu đỏ (Tốc độ thực) bám sát đường màu xanh (Setpoint). Sai số tĩnh gần như bằng 0 nhờ tác động của khâu K_i.



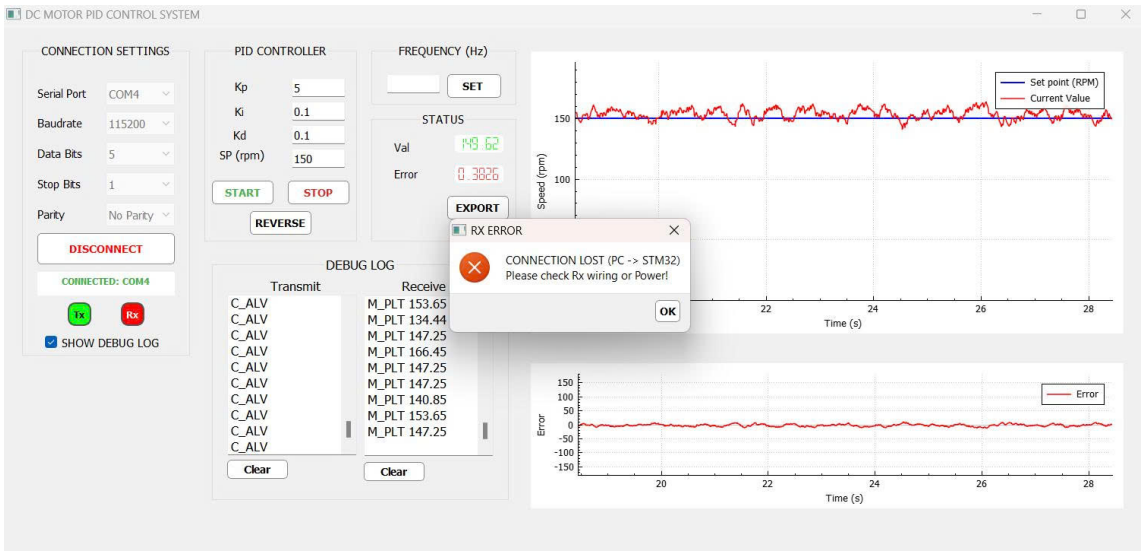
Hình 10: Đáp ứng tốc độ động cơ tại Setpoint = 150 RPM

4.3 Khả năng phát hiện lỗi (Error Handling)

Hệ thống được tích hợp tính năng giám sát đường truyền. Khi kết nối vật lý (dây TX/RX) bị ngắt hoặc nguồn cấp cho vi điều khiển bị mất, phần mềm sẽ tự động phát hiện và gửi cảnh báo "Connection Lost" tới người dùng, đồng thời dừng cập nhật đồ thị để tránh sai lệch dữ liệu *(như Hình 11 và Hình 12)*.



Hình 11: Cảnh báo mất kết nối đường truyền (TX Error)



Hình 12: Cảnh báo mất kết nối đường truyền (RX Error).

5 Kết luận và Hướng phát triển

5.1 Kết luận

Nhóm đã hoàn thành các mục tiêu đề ra ban đầu:

- Thiết kế và thi công thành công phần cứng điều khiển động cơ DC sử dụng STM32F407 và Driver L298N.
- Xây dựng giao thức truyền thông UART sử dụng cơ chế DMA để giao tiếp giữa GUI và board STM32F407.
- Áp dụng thành công thuật toán PID để ổn định tốc độ động cơ với đáp ứng nhanh và sai số thấp.

5.2 Hướng phát triển

- Nâng cấp mạch công suất để chịu dòng tải lớn hơn.
- Thực hiện điều khiển vị trí (Position Control) bên cạnh điều khiển tốc độ.
- Tối ưu hóa thuật toán PID (Auto-tuning) để tự động tìm tham số tối ưu cho từng loại tải khác nhau.

6 Tài liệu tham khảo

https://www.st.com/en/evaluation-tools/stm32f4discovery.html

https://hshop.vn/kit-stm32f4-discovery-armcortex-m4-dsp-core

https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

https://cdn.sparkfun.com/assets/7/1/d/6/c/Full-Bridge_Motor_Driver_Dual_-_L298N.pdf

https://assets.nexperia.com/documents/data-sheet/74HC_HCT245.pdf