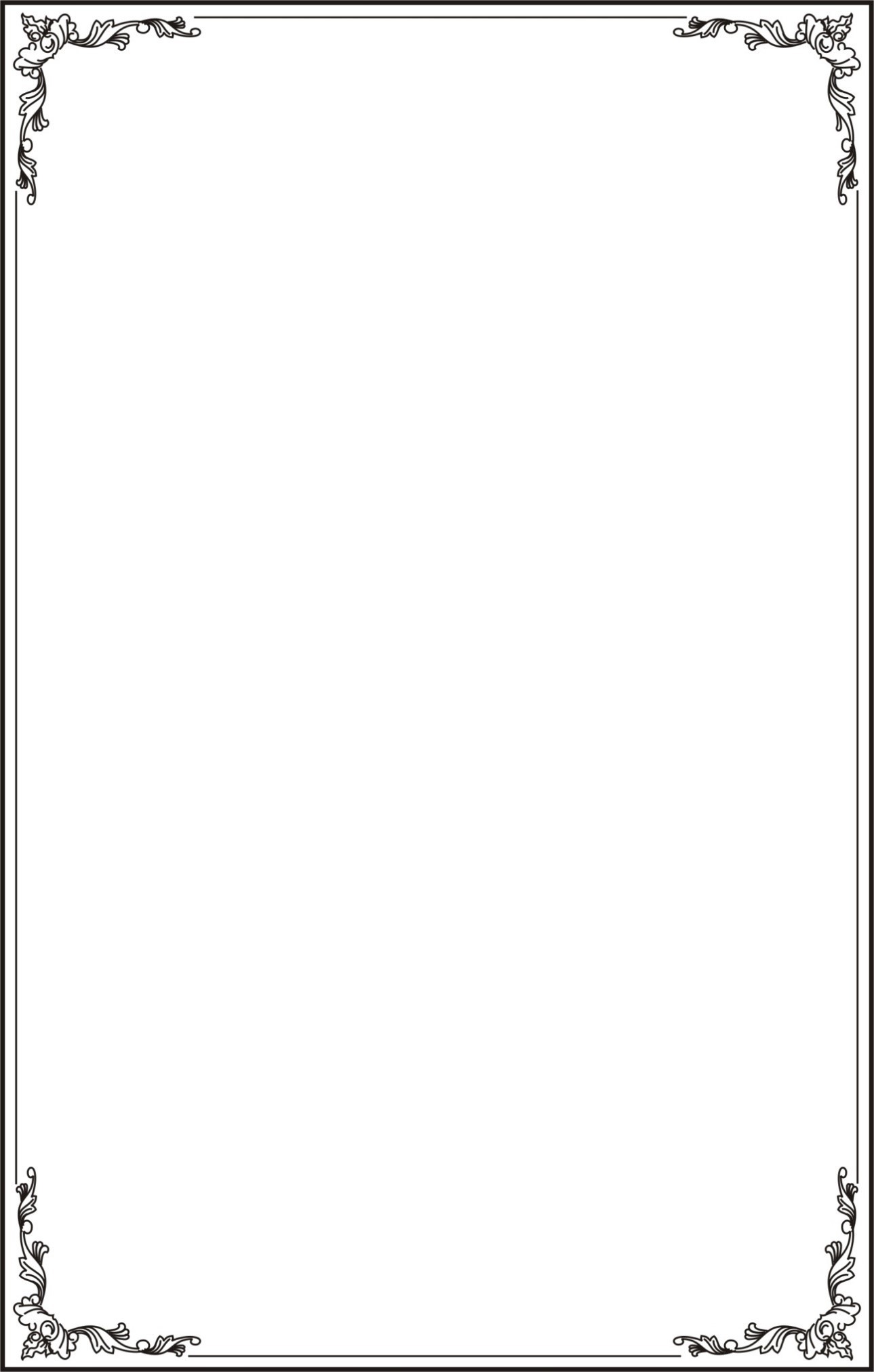
|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | **Võ Lâm Chương** |
| **Sinh viên thực hiện:** |  |
| **Trần Xuân Hoàng** | **22146311** |
| **Nguyễn Thành Danh** |  |
| **Nguyễn Huỳnh Anh** |  |
| **Đặng Thành Tài** |  |

*TP Hồ Chí Minh, Tháng … Năm 2025*

**BÁO CÁO THỰC TẬP SERVO**



**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ KỸ THUẬT TP.HCM**

**KHOA CƠ KHÍ CHẾ TẠO MÁY**

**BỘ MÔN CƠ ĐIỆN TỬ**

**---------------🙡 🕮 🙣---------------**

**Mục Lục**

[BÀI THỰC HÁNH SỐ 1: HỆ THỐNG SERVO ĐA TRỤC SỬ DỤNG ĐỘNG CƠ AC SERVO 4](#_Toc218346965)

[1.1. Kết cấu phần cứng: 4](#_Toc218346966)

[1.2. Sơ đồ đấu nối hệ thống: 5](#_Toc218346967)

[1.3. Điều khiển chế độ Test-mode (bộ điều khiển SSC): 7](#_Toc218346968)

[1.4. Chương trình nội suy hình trên modul FX3U điều khiển đa trục (XY): 9](#_Toc218346969)

[1.5. Chương trình PLC 10](#_Toc218346970)

[1.6. Kết quả thực hiện (các bài thực nghiệm): 14](#_Toc218346971)

[BÀI THỰC HÁNH SỐ 2: HỆ THỐNG THỦY LỰC SỬ DỤNG PLC 16](#_Toc218346972)

[2.1. Kết cấu phần cứng: 16](#_Toc218346973)

[2.2. Sơ đồ đấu nối hệ thống thủy lực: 17](#_Toc218346974)

[2.3. Chương trình PLC điều khiển hệ thống thủy lực: 18](#_Toc218346975)

[2.4. Kết quả thực hiện: 21](#_Toc218346976)

[BÀI THỰC HÁNH SỐ 3: ĐIỀU KHIỂN TỐC ĐỘ ĐỘNG CƠ DC SERVO 22](#_Toc218346977)

[3.1. Tổng quan lý thuyết: 22](#_Toc218346978)

[3.2. Sơ đồ đấu nối và cấu hình phần cứng: 27](#_Toc218346979)

[3.3. Cấu hình dự án trên STM32 CubeMX: 27](#_Toc218346980)

[3.4. Chương trình điều khiển STM32: 28](#_Toc218346981)

[3.5. Tính toán thông số PID: 29](#_Toc218346982)

[3.6. Kết quả mô phỏng và kết quả thực tế: 29](#_Toc218346983)

[BÀI THỰC HÁNH SỐ 4: ĐIỀU KHIỂN VỊ TRÍ ĐỘNG CƠ DC SERVO SỬ DỤNG VI ĐIỀU KHIỂN STM32 31](#_Toc218346984)

[4.1. Kết cấu phần cứng: 31](#_Toc218346985)

[4.2. Sơ đồ đấu nối phần cứng: 31](#_Toc218346986)

[4.3. Chương trình STM32: 32](#_Toc218346987)

[4.4. Tính toán thông số PID: 33](#_Toc218346988)

[4.5. Mô phỏng trên MATLAB/Simulink và kết quả thực tế: 35](#_Toc218346989)

# HỆ THỐNG SERVO ĐA TRỤC SỬ DỤNG ĐỘNG CƠ AC SERVO

Mục tiêu: Vận hành và điều khiển hệ servo nhiều trục (servo XY) sử dụng động cơ AC servo và module điều khiển chuyên dụng. Thực hành chế độ điều khiển Test mode của module điều khiển servo Mitsubishi, lập trình PLC để thực hiện các chuyển động đa trục (nội suy đường thẳng và cung tròn), quan sát và đánh giá kết quả chuyển động.

## Kết cấu phần cứng:

Hệ thống servo đa trục gồm bộ điều khiển (tủ điều khiển) và cơ cấu chấp hành servo 2 trục. Bộ điều khiển sử dụng PLC Mitsubishi FX3U kết hợp module điều khiển vị trí đa trục FX3U-20SSC-H (SSC Net) dành cho servo. Module này có khả năng điều khiển đồng thời tới 2 trục servo, hỗ trợ các chế độ điều khiển vị trí, tốc độ và nội suy đường tròn/đường thẳng. Trong hệ thống thí nghiệm, module FX3U-20SSC-H được cấu hình để điều khiển 2 động cơ AC servo loại Mitsubishi HF-KP23 (200 W) gắn trên mô hình bàn máy phẳng dạng XY. Trục thứ 3 trong hệ thống không tham gia chuyển động ở các bài thực hành (có thể dự phòng cho trục Z nhưng không sử dụng trong các ví dụ). Hình dưới đây thể hiện rõ phần tủ điện từ bên ngoài, bên trong tủ điện, và phần cơ cấu chấp hành.

A close-up of a machine

AI-generated content may be incorrect.

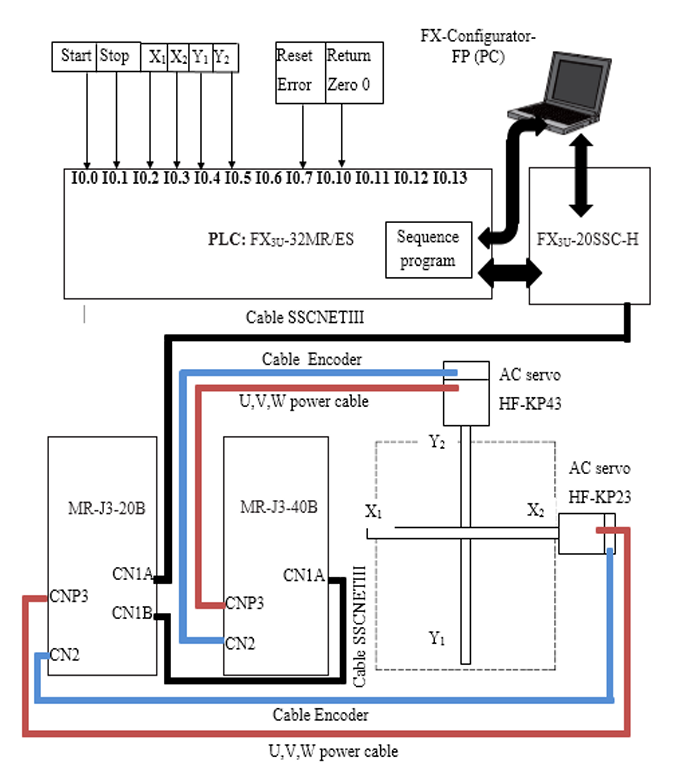
Động cơ AC servo HF-KP23 có tích hợp encoder phản hồi độ phân giải 131.072 xung/vòng. Các động cơ này được điều khiển bởi driver (bộ khuếch đại servo) Mitsubishi MR-J3 tương ứng. Driver servo giao tiếp với module FX3U-20SSC-H qua mạng SSCNET nên việc kết nối dây rất gọn và đơn giản (chỉ cần cáp quang). Module SSC điều khiển vị trí/tốc độ servo dựa trên lệnh từ PLC, đồng thời nhận phản hồi encoder từ driver servo. Tủ điều khiển có các nút nhấn và công tắc để khởi động, dừng khẩn và chọn chế độ vận hành.

## Sơ đồ đấu nối hệ thống:

A diagram of electrical components

AI-generated content may be incorrect.Các động cơ servo AC và driver được nối với module FX3U-20SSC-H theo hướng dẫn của hãng Mitsubishi. Mỗi driver MR-J3 kết nối với module SSC bằng một cáp quang (cổng CH1, CH2 tương ứng trục X, Y). PLC FX3U kết nối với module SSC qua thanh nối mở rộng bên hông. Ngoài ra, một số cảm biến hành trình (nếu có) có thể nối vào các ngõ vào PLC để làm tín hiệu gốc (homing) cho các trục.

Trên mô hình thực, hệ thống đã được đấu nối sẵn theo sơ đồ từ nhà sản xuất, đảm bảo an toàn. Người thực hành không cần thay đổi đấu nối phần cứng, chỉ cần thiết lập cấu hình và viết chương trình điều khiển phù hợp.



## Điều khiển chế độ Test-mode (bộ điều khiển SSC):

Module FX3U-20SSC-H cho phép điều khiển các trục servo thông qua các lệnh trong chương trình PLC. Tuy nhiên, Mitsubishi cũng cung cấp chế độ Test mode – một chế độ vận hành bằng giao diện phần mềm giúp thử nghiệm nhanh chuyển động các trục trước khi lập trình đầy đủ. Trong bài này, sinh viên sẽ thực hành điều khiển các trục servo ở chế độ Test mode để làm quen với hệ thống.

Các bước thực hiện chế độ Test mode như sau:

*Kết nối và khởi tạo module:* Trên phần mềm (ví dụ GX Works hoặc phần mềm chuyên dụng cho SSC), tạo file dự án mới, thiết lập kết nối cổng COM giữa PC và PLC (chọn đúng cổng COM và tốc độ Baud của bộ lập trình). Sau đó khởi tạo module FX3U-20SSC-H (chọn số trục điều khiển, cấu hình thông số động cơ như số xung encoder, tỷ số truyền cơ khí, v.v.).

Cấu hình cho FX3U-20SSC-H thông qua các bước như trong tài liệu hướng dẫn. Cấu hình thông số hệ thống: Thiết lập các tham số chuyển động cho từng trục trong chế độ Test (tốc độ di chuyển, gia tốc/giảm tốc mặc định, đơn vị mm hay deg...). Các thông số này thường lấy từ datasheet động cơ và cơ cấu. A close-up of a test

AI-generated content may be incorrect.

Thực hiện lệnh di chuyển: Giao diện Test mode cung cấp sẵn các lệnh di chuyển cơ bản. Mỗi chương trình chạy trong Test mode luôn bắt đầu bằng lệnh Incremental address specification (chỉ định địa chỉ gia số) và kết thúc bằng lệnh End. Người dùng có thể chọn các lệnh như:

A screenshot of a computer program

AI-generated content may be incorrect.

“X-axis positioning at 1-step speed”: di chuyển trục X một đoạn với tốc độ 1-step (tốc độ đặt trước).

“Y-axis positioning at 1-step speed”: di chuyển trục Y một đoạn với tốc độ 1-step.

“XY-axis positioning at 1-step speed”: di chuyển đồng thời trục X và Y (nội suy đường thẳng).

A screenshot of a computer

AI-generated content may be incorrect.

“Circular interpolation”: nội suy cung tròn (đồng thời 2 trục tạo chuyển động theo cung tròn). A screenshot of a computer

AI-generated content may be incorrect.

Quan sát giao diện điều khiển: Sau khi thiết lập lệnh, nhấn thực thi để module gửi lệnh đến các driver servo. Người dùng có thể quan sát bàn trượt servo di chuyển theo lệnh đã chọn. Giao diện phần mềm sẽ hiển thị trạng thái vị trí hiện tại của các trục, các bước lệnh đang thực hiện.

## Chương trình nội suy hình trên modul FX3U điều khiển đa trục (XY):

Sau khi thử nghiệm thành công ở chế độ Test mode, bước tiếp theo là viết chương trình PLC để điều khiển các trục servo tự động. Chương trình PLC được viết bằng ngôn ngữ dạng ladder (hình thang). Trong bài này, nhóm đã lập trình một đoạn chương trình để thực hiện lần lượt các chuyển động như trong chế độ Test mode (Ví dụ 1 đến 5 ở mục 3.5).

Chương trình sử dụng các lệnh chuyên dụng của module FX3U-20SSC-H để điều khiển servo. Một số điểm chính trong chương trình như sau:

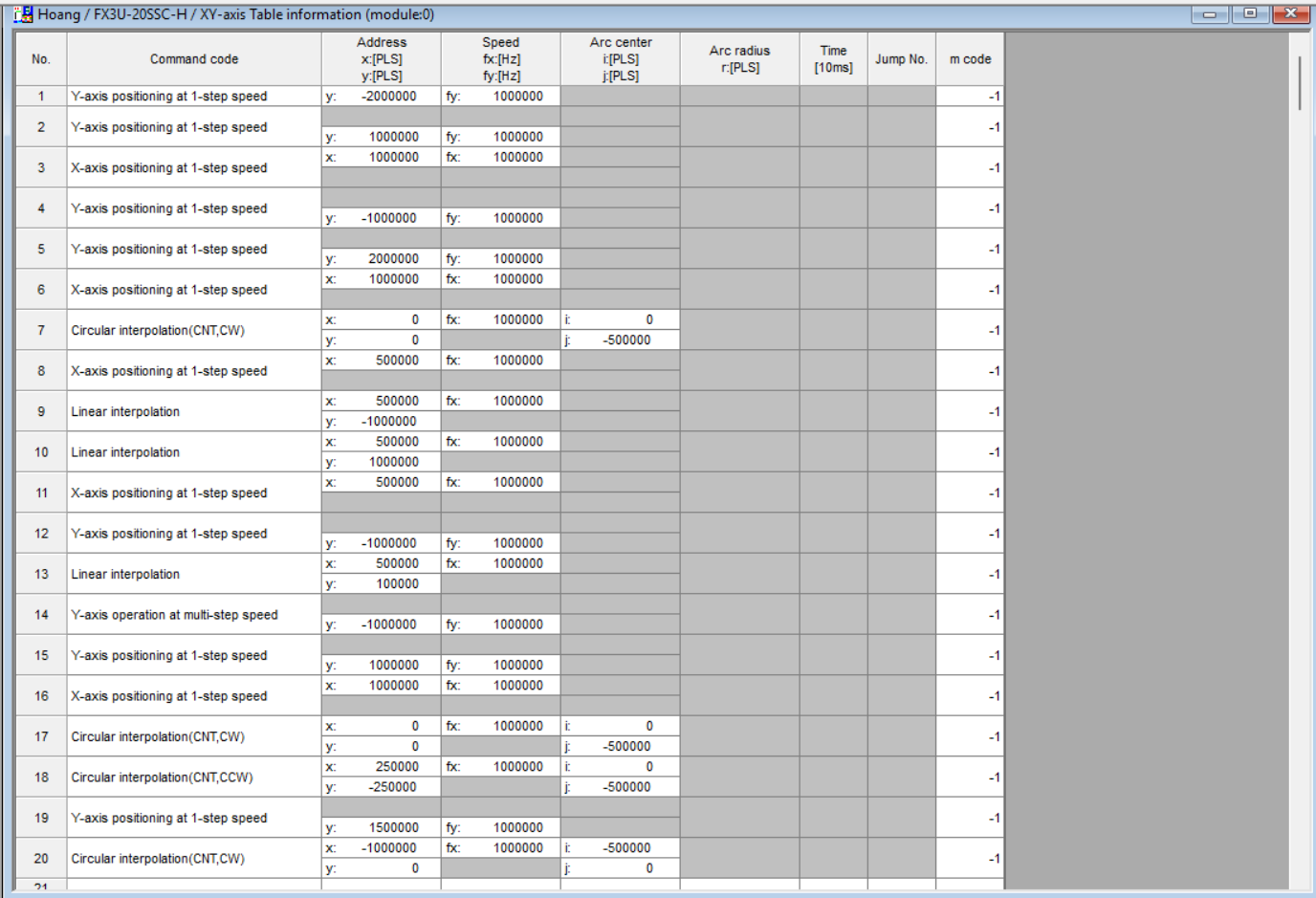
Khởi động module SSC: Khi bắt đầu, chương trình gửi lệnh INCMOV (Incremental Move) để chuẩn bị chế độ định địa chỉ gia số cho các trục. Đây tương đương với việc đặt hệ trục toạ độ tương đối cho các lệnh di chuyển sau. Mỗi lần chương trình bắt đầu chu trình mới cũng lặp lại bước này.

Điều khiển trục X, Y đơn lẻ: Sử dụng các lệnh vị trí cho từng trục, ví dụ lệnh DDRF (Drive axis Forward) kèm tham số tốc độ và quãng đường cho trục X hoặc Y. Mỗi lệnh thực thi xong thì chương trình đợi tín hiệu hoàn thành từ module (servo đến vị trí đích).

Điều khiển đồng thời (nội suy): Sử dụng lệnh nội suy vòng tròn hoặc đường thẳng, ví dụ CIRC (Circular interpolation) để hai trục X, Y chuyển động đồng thời tạo thành cung tròn. Tham số bán kính hoặc góc cung được tính toán trước theo yêu cầu bài toán (ví dụ cung tròn 90 độ hay 180 độ).

Kết thúc chu trình: Sau khi thực hiện xong các lệnh di chuyển, chương trình ladder kết thúc bằng lệnh END để dừng chương trình đúng chuẩn.

Dưới đây là hình ảnh chương trình ladder PLC của nhóm, trong đó thể hiện các bước lệnh kể trên cho trục X và Y (bao gồm lệnh đặt địa chỉ gia số, các lệnh điều khiển vị trí trục X, Y và lệnh nội suy tròn, cùng tín hiệu kết thúc):



*Hình 15. Một phần chương trình PLC (dạng ladder) điều khiển 2 trục servo X và Y: bao gồm các lệnh điều khiển vị trí cho trục X, trục Y, lệnh nội suy cung tròn XY và logic điều khiển trình tự thực hiện các lệnh.*

## Chương trình PLC

Chương trình PLC giúp đọc nút nhấn và gửi các lệnh tương ứng xuống modul FX3U để thực thi. Chương trình PLC của nhóm em được viết như hình bên dưới đây, tham khảo theo tài liệu thực tập servo mà nhóm đã được học:

A screenshot of a computer

AI-generated content may be incorrect. A diagram of a computer

AI-generated content may be incorrect. A screenshot of a computer program

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

## Kết quả thực hiện (các bài thực nghiệm):

Sau khi lập trình PLC, nhóm đã cho hệ thống servo đa trục thực hiện tuần tự 5 bài thực nghiệm (Ví dụ 1 đến Ví dụ 5) như sau:

Ví dụ 1: Di chuyển trục X một đoạn cố định với tốc độ đặt trước, trong khi trục Y giữ cố định. Kết quả: bàn trượt di chuyển dọc theo phương X, đúng quãng đường yêu cầu (Hình 16).

Ví dụ 2: Di chuyển trục Y một đoạn cố định với tốc độ đặt trước (trục X không đổi). Kết quả: bàn trượt di chuyển dọc theo phương Y như mong muốn (Hình 17).

Ví dụ 3: Di chuyển đồng thời cả hai trục X và Y một đoạn bằng nhau tại cùng tốc độ (nội suy thẳng). Kết quả: bàn trượt di chuyển theo đường chéo 45° trên mặt phẳng XY (Hình 18).

Ví dụ 4: Chuyển động nội suy cung tròn trên mặt phẳng XY. Trong thí nghiệm này, nhóm lập trình cho bàn trượt di chuyển theo cung tròn góc 90° (phần tư của vòng tròn). Kết quả: bàn trượt vẽ một cung tròn kết nối hai điểm đặt ra, chuyển động trơn tru (Hình 19).

Ví dụ 5: Thực hiện một chuyển động phức tạp hơn kết hợp nhiều đoạn thẳng và cung tròn. Cụ thể, nhóm lập trình cho bàn trượt vẽ một đường zigzag hoặc một đường tròn khép kín (tùy chọn). Kết quả: hệ thống servo phối hợp các trục để tạo quỹ đạo hình học yêu cầu (Hình 20).

Dưới đây là hình ảnh chụp quỹ đạo chuyển động tương ứng của bàn trượt servo cho các ví dụ 1–5:

Qua các bài thực nghiệm, có thể nhận thấy hệ thống servo đa trục hoạt động đúng theo yêu cầu lệnh đặt ra. Các chuyển động tuyến tính (theo trục X, Y hoặc đường chéo) đều chính xác về quãng đường và phương hướng. Chuyển động nội suy tròn tương đối mượt, bàn trượt di chuyển đồng thời hai trục không bị rung giật. Sai số cuối cùng (sai lệch vị trí khi dừng) hầu như không đáng kể, nhờ có hệ thống servo AC có độ chính xác cao (encoder độ phân giải lớn) và thuật toán điều khiển của driver servo hoạt động hiệu quả. Điều này cho thấy việc sử dụng module điều khiển chuyên dụng (SSCNET) kết hợp PLC đã đơn giản hóa việc lập trình, đồng thời đảm bảo độ đồng bộ khi điều khiển đa trục.

# HỆ THỐNG THỦY LỰC SỬ DỤNG PLC

Mục tiêu: Vận hành mô hình hệ thống thủy lực sử dụng van servo và xi-lanh thủy lực, điều khiển đóng vòng bằng bộ điều khiển PID lập trình trên PLC. Khảo sát việc sử dụng module chuyển đổi ADC/DAC trên PLC để đọc tín hiệu cảm biến và xuất tín hiệu điều khiển tương tự.

## Kết cấu phần cứng:

A diagram of a machine

AI-generated content may be incorrect.A close-up of a machine

AI-generated content may be incorrect.Hệ thống thủy lực thí nghiệm gồm một cụm trạm thủy lực (bồn dầu, bơm, van, xi-lanh) và tủ điều khiển PLC. Trạm này gồm mô hình điều khiển, bao gồm PLC Mitsubishi FX3U gắn các module mở rộng ADC/DAC để giao tiếp tín hiệu analog. Cơ cấu chấp hành là mô hình trạm thủy lực với xi-lanh tác động kép được điều khiển bởi van servo. Trên xi-lanh có gắn cảm biến đo vị trí (hoặc cảm biến áp suất tùy cấu hình bài) để phản hồi tín hiệu về PLC.

A grey rectangular object with wires

AI-generated content may be incorrect.

Cụ thể, PLC FX3U được gắn module FX2N-2AD (2 kênh Analog-Digital) để đọc tín hiệu tương tự từ cảm biến, và module FX2N-4DA (4 kênh Digital-Analog) để xuất tín hiệu điều khiển van servo. Van servo trong hệ thống là loại van tỷ lệ điện, điều khiển bằng điện áp analog ±10 V (xuất từ module 4DA). Cảm biến vị trí xi-lanh (hoặc cảm biến áp suất) xuất tín hiệu 0-10 V hoặc 4-20 mA, được chuyển đổi về PLC qua module 2AD. Bộ PLC đọc giá trị cảm biến (ví dụ hành trình xi-lanh) liên tục và so sánh với giá trị đặt, tính toán bộ PID để điều khiển van servo đóng/mở nhằm đạt vị trí/áp suất mong muốn.

## Sơ đồ đấu nối hệ thống thủy lực:

Module 2AD (FX2N-2AD): Module này nhận tín hiệu analog 0-10 V từ cảm biến. Trong PLC, hai thanh ghi dữ liệu (ví dụ D100 cho kênh CH1, D101 cho CH2) được dùng để lưu giá trị chuyển đổi (giá trị số 12-bit tương ứng mức áp). Cần cấu hình module 2AD ở chế độ đọc liên tục hai kênh analog.

Module 4DA (FX2N-4DA): Module này có 4 kênh output analog. Chọn chế độ ±10 V cho cả 4 kênh để điều khiển van. Thanh ghi D0-D3 dùng để chứa giá trị số xuất ra 4 kênh tương ứng CH1-CH4 (giá trị 12-bit, 0 tương ứng -10 V, 2000h tương ứng +10 V). Phải cấu hình các thanh ghi này trước khi vận hành.

Việc đấu nối dây: tín hiệu cảm biến (0-10 V) được đấu vào CH1 của module 2AD; tín hiệu điều khiển van servo lấy từ CH1 của module 4DA (±10 V) và nối vào cuộn điều khiển van.

A diagram of a machine

AI-generated content may be incorrect.

## Chương trình PLC điều khiển hệ thống thủy lực:

Chương trình PLC được viết để đọc giá trị cảm biến, tính toán PID và xuất giá trị điều khiển tương ứng. Dưới đây là trích đoạn chương trình dạng lệnh (toán hạng) kèm giải thích cho các bước chính:

Đọc module 2AD (chuyển A/D): Module 2AD cần được kích hoạt chuyển đổi liên tục. Trong PLC, sử dụng lệnh TO để ghi từ K1 (module 2AD ở vị trí 1) giá trị cấu hình H1 vào thanh ghi K17 của module (bật chế độ chuyển A/D liên tục cho cả 2 kênh). Lệnh TO K1 K17 H1 K1 cấu hình kênh bắt đầu, và TO K1 K17 H3 K1 cấu hình số kênh chuyển đổi. Kết quả, module 2AD sẽ liên tục chuyển giá trị analog sang cặp nhớ M1000-M1007 và M1008-M1015 (ghép thành thanh ghi 16 bit). Sau đó dùng lệnh FROM để đọc giá trị đó vào thanh ghi dữ liệu: FROM K1 K0 K2 M100 K2 (đọc 2 thanh ghi từ module 2AD, lưu vào D100 và D101). Cuối cùng, lệnh MOV K3M100 D100 được dùng để ghép các bit và lưu giá trị analog đã đọc được vào thanh ghi D100 (giá trị số tương ứng cảm biến).

Xuất module 4DA (chuyển D/A): Trước khi xuất giá trị analog, cần cấu hình module 4DA bằng lệnh TO K0 K0 H0 K1 (cấu hình phạm vi ±10V cho 4 kênh) và TO K0 K4 D0 K1 (bắt đầu truyền dữ liệu từ D0 đến 4 kênh output). Trong đó K0 (cột 1) là vị trí module 4DA (gắn trực tiếp vào PLC, vị trí 0), K0 (cột 2) chọn thanh ghi cấu hình (địa chỉ 0), H0 đặt chế độ ±10V cho cả 4 kênh, và D0 là thanh ghi đầu tiên chứa giá trị cần xuất (sẽ lấy D0->CH1, D1->CH2, D2->CH3, D3->CH4). Sau khi cấu hình, chỉ cần ghi giá trị 12-bit tương ứng điện áp điều khiển vào D0, module sẽ tự động xuất ra CH1.

Thuật toán PID: Bộ PLC Mitsubishi cung cấp lệnh PID tích hợp. Lệnh PID D200 D128 D300 D0 được sử dụng, với ý nghĩa: lấy giá trị đặt từ thanh ghi D200 (setpoint), giá trị đo từ D128 (process variable), bắt đầu vùng thông số PID tại D300, và xuất kết quả điều khiển vào D0. Ở đây D0 chính là thanh ghi output analog CH1 (điều khiển van servo), do đó lệnh PID sẽ tự động tính và ghi giá trị điều khiển vào D0. Các tham số PID (Kp, Ki, Kd, chu kỳ lấy mẫu) cần được nạp vào các thanh ghi bắt đầu từ D300 trước khi chạy lệnh. Nhóm đã thiết lập các thông số PID dựa trên tính toán và tinh chỉnh: ví dụ nạp giá trị $K\_c = 7,85$ vào D300, $T\_d = 0,13$ s vào D301, v.v (cụ thể phương pháp tính tương tự bài 2, không nêu chi tiết ở đây). Lệnh MOV K150 D300 cũng được sử dụng để nạp nhanh một hằng số (0x96 = 150) vào vùng thanh ghi D300 nhằm thử nghiệm đáp ứng (ví dụ cho setpoint = 150 đơn vị).

Tóm lại, chương trình PLC thực hiện vòng lặp: đọc cảm biến (0-10V) vào D128, so sánh với setpoint D200, lệnh PID tính toán và cập nhật liên tục giá trị D0 (xuất analog ±10V ra van servo). Nhờ đó tạo thành vòng điều khiển kín cho hệ thống thủy lực.

*Trích đoạn chương trình:*

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

## Kết quả thực hiện:

Hệ thống thủy lực được thử nghiệm với chế độ điều khiển PID như trên. Thông số mục tiêu có thể là vị trí xi-lanh hoặc áp suất trong xi-lanh, tùy cấu hình cảm biến. Trong thí nghiệm này, giả sử hệ thống điều khiển vị trí xy-lanh về một vị trí đặt cố định (tương ứng giá trị đặt 150 trong thanh ghi D200). Hình 22 cho thấy kết quả đáp ứng của hệ thống: xi-lanh di chuyển đến vị trí đặt với một độ vọt lố nhỏ và ổn định tại vị trí yêu cầu sau khoảng 2 giây. Đường màu xanh biểu diễn vị trí đặt, đường màu đỏ là vị trí thực tế của xi-lanh thu thập từ cảm biến (đã qua chuyển đổi ADC). Có thể thấy bộ điều khiển PID giúp hệ thống đạt vị trí mục tiêu với sai số rất nhỏ, sai lệch được triệt tiêu hoàn toàn sau khi ổn định.

*Hình 22.*

Trong quá trình điều khiển, giá trị điều khiển (điện áp van servo) thay đổi tương ứng: ban đầu van mở lớn để xi-lanh di chuyển nhanh, sau đó giảm dần khi gần tới vị trí đặt nhằm tránh vượt quá. Điều này cho thấy thuật toán PID hoạt động đúng vai trò – hiệu chỉnh liên tục dựa trên sai lệch để đưa hệ thống về trạng thái cân bằng.

Nhìn chung, qua bài thực hành này, sinh viên đã nắm được cách tích hợp module analog với PLC để điều khiển một hệ thống liên tục (thủy lực). Bộ điều khiển PID số chạy trên PLC đã chứng minh hiệu quả trong việc ổn định vị trí/áp suất hệ thủy lực. Đây là cơ sở để có thể thiết kế các hệ thống điều khiển tương tự cho các quá trình công nghiệp (ví dụ: điều khiển vị trí xi-lanh trong robot thủy lực, điều khiển áp suất trong hệ thống thủy lực tự động, ...).

# ĐIỀU KHIỂN TỐC ĐỘ ĐỘNG CƠ DC SERVO

Mục tiêu: Thiết kế bộ điều khiển PI để ổn định tốc độ quay của động cơ DC servo quanh giá trị đặt, sử dụng vi điều khiển STM32F103C8. Khảo sát đặc tính động học của động cơ DC servo, tính toán tham số bộ điều khiển PI theo phương pháp IMC, so sánh đáp ứng mô phỏng và thực nghiệm.

## Tổng quan lý thuyết:

Đối với động cơ DC servo, mô hình toán học cơ bản gồm hai phần: hệ thống điện (cuộn dây phần ứng) và hệ thống cơ (rotor + tải).

Trong motor DC, mô-men điện từ tạo bởi dòng phần ứng được mô tả bởi:



Trong đó: (N·m): mô-men điện từ trên trục moto; (N·m/A): hằng số mô-men; (A): dòng điện cảm ứng

Sức điện động phản kháng (Back-EMF)

Khi rotor quay với vận tốc góc , motor sinh ra sức điện động phản kháng chống lại điện áp đặt vào:



Trong đó: (V): điện áp ngược; (V·s/rad): hằng số điện áp ngược; (rad/s): vận tốc góc rotor.

Tiếp theo là mối quan hệ giữa và (theo bảo toàn năng lượng):

Công suất:



Trong đó là điện áp. Thành phần công suất có ích liên quan đến chuyển đổi điện–cơ là:



Công suất cơ trên trục:



Theo nguyên lý bảo toàn năng lượng (phần công có ích điện–cơ), suy ra:



Quan hệ đúng khi sử dụng hệ đơn vị SI và xét chuyển đổi lý tưởng.

Mô hình toán học của motor

Phương trình cơ học (Newton–Euler)

Phương trình cân bằng mô-men trên trục motor:



trong đó: (kg·): mô-men quán tính rotor; (N·m): mô-men tải tổng quát.

Với hệ motor kéo cánh quạt, mô-men tải khí động thường được xấp xỉ dạng phi tuyến theo bình phương tốc độ như phương trình

Thay phương trình vào :



Phương trình điện (Kirchhoff)

Theo định luật Kirchhoff cho mạch phần ứng:



Trong bài toán điều khiển tốc độ của động cơ UAV, do nhỏ hơn nhiều so với động học động cơ do đó có thể bỏ qua ảnh hưởng cuộn cảm, khi đó: 

Phương trình động lực học tổng hợp (dạng phi tuyến)

Thế phương trình vào phương trình :



Tuyến tính hoá quanh điểm làm việc và hàm truyền

Tại điểm cân bằng (hover): , , . Từ phương trình :



Đặt :



Thế vào , khai triển và bỏ qua thành phần bậc hai nhỏ , thu được:



Chia hai vế cho đưa về dạng chuẩn bậc nhất:



với: 

#### Hàm truyền động cơ

Lấy biến đổi Laplace (với điều kiện đầu bằng 0):



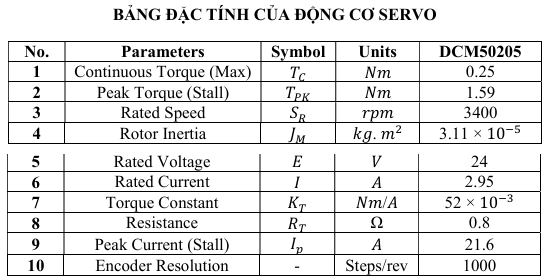
Khi đó ta tìm được hàm truyền bật nhất (xấp xỉ) của động cơ như sau:



**A diagram of a circuit

AI-generated content may be incorrect.2.2. Kết cấu phần cứng:**

Phần cứng của bài toán điều khiển tốc độ sử dụng cùng mô hình động cơ DC servo một trục như bài 1. Hình đưới đây chụp lại mô hình thực tế của động cơ DC servo và bộ driver một trục trong phòng thí nghiệm. Các thông số của động cơ servo DCM50205 được liệt kê ở bảng bên dưới. Động cơ này có đặc tính moment-tốc độ như nhau, sử dụng chung cho cả bài vận tốc và bài vị trí. Tải quán tính trên trục động cơ không đổi.

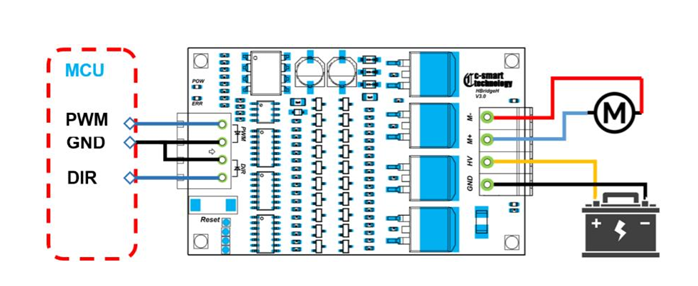


A blue circuit board with white text

AI-generated content may be incorrect.A green circuit board with black and white components

AI-generated content may be incorrect.

## Sơ đồ đấu nối và cấu hình phần cứng:

Sơ đồ đấu nối phần cứng cho bài toán điều khiển vận tốc đưới mô tả như hình bên dưới. Vi điều khiển STM32F103C8 vẫn sử dụng các kênh PWM và ngắt như đã cấu hình trang tài liệu hướng dẫn. Cụ thể, động cơ DC servo được nối với cầu H L298N, tín hiệu PWM điều khiển tốc độ từ STM32 (Timer 3, chân PC7). Encoder động cơ nối vào các chân PC3, PC7 (EXTI) để đo tốc độ vòng quay. Ngoài ra, để thuận tiện cho việc giám sát, vi điều khiển truyền dữ liệu tốc độ hiện tại qua UART (USART1) về máy tính (ví dụ hiển thị trên terminal). 

## Cấu hình dự án trên STM32 CubeMX:

Để lập trình vi điều khiển STM32 một cách hiệu quả, nhóm đã sử dụng công cụ STM32 CubeMX để cấu hình ban đầu các chân và ngoại vi. Các bước cấu hình chính gồm:

Tạo project mới: Chọn vi điều khiển STM32F103C8, cấu hình clock hệ thống 72 MHz (bật PLL và đặt nguồn HSE nếu có).

Kích hoạt các ngắt ngoài (External Interrupt): Bật chức năng ngắt ngoài trên các chân PC3 và PC7 (tương ứng EXTI3 và EXTI9\_5) để đọc xung encoder.

Cấu hình Timer3 ở chế độ PWM: Timer 3 kênh 2 được cấu hình Output Compare PWM Mode, tần số PWM ~20 kHz, chân output là PC7.

Cấu hình UART: Bật UART1 ở tốc độ baud 115200 để truyền dữ liệu debug (nếu cần). Chân TX (PA9), RX (PA10).

Sau khi cấu hình CubeMX, mã khởi tạo HAL được tạo, đảm bảo các timer, ngắt, UART sẵn sàng cho lập trình điều khiển.

*A screenshot of a computer

AI-generated content may be incorrect.*

## Chương trình điều khiển STM32:

Chương trình điều khiển vận tốc trên STM32 về cơ bản chỉ cần dung 1 vòng điều khiển PI là đủ. Vi điều khiển vẫn đếm xung encoder bằng ngắt ngoài, nhưng thay vì tích lũy để tính vị trí, chương trình sẽ tính toán tốc độ từ số xung thu được trong khoảng thời gian nhất định.

Cụ thể, trong ngắt Timer định kỳ (Timer4, chu kỳ T = 10 ms), vi điều khiển đọc số xung encoder đếm được trong khoảng T (hiệu giữa count hiện tại và count lần trước). Từ đó tính ra tốc độ góc dựa trên số xung/chu kỳ và độ phân giải encoder. Giá trị tốc độ thực đo này được so sánh với tốc độ đặt (setpoint), tính sai lệch và thực hiện thuật toán PI để điều chỉnh độ rộng PWM đầu ra.

Các bước chính của chương trình:

1. Đọc encoder: Sử dụng ngắt EXTI trên PC3, PC7 để tăng/giảm biến đếm count tương tự bài 1.
2. Tính tốc độ trong ngắt Timer4: Mỗi chu kỳ Timer4 (10 ms),
3. Điều khiển PI
4. Cập nhật PWM: Chuyển giá trị điều khiển thành độ rộng xung PWM (giới hạn 0-100%).

Chương trình cũng gửi giá trị tốc độ thực qua UART về PC để vẽ đồ thị kiểm chứng.

## Tính toán thông số PID:

Nhóm đã tiến hành thực nghiệm để xác định tham số đối tượng cho điều khiển tốc độ. Tín hiệu PWM đầu vào được thay đổi từ 20% lên 96%, và đo tốc độ động cơ tương ứng. Kết quả: tốc độ tăng từ khoảng 28 rad/s lên 204 rad/s. Suy ra: $\Delta \omega \approx 176,2$ rad/s với $\Delta u = 76%$. Thời gian đạt 63% $\Delta \omega$ khoảng 0,062 s. Từ đó mô hình hệ thống tốc độ là $G\_\omega(s) = \frac{2,3189}{0,062s+1}$ (như đã tính ở mục 1.4).

Với mô hình này, áp dụng phương pháp IMC thiết kế bộ điều khiển PI tốc độ:

* Chọn $\tau\_c = \tau\_p/5 = 0,0124$ s.
* Tính $K\_c = \frac{\tau\_c}{K\_p \tau\_p} = \frac{0,0124}{2,3189 \times 0,062} \approx 0,0863$.
* Chọn $\tau\_i = \tau\_p = 0,062$ s (đặt thời gian tích phân bằng hằng số thời gian quá trình).
* Tính $K\_i = \frac{K\_c}{\tau\_i} = \frac{0,0863}{0,062} \approx 1,39\ \text{s}^{-1}$.

Bộ PI điều khiển tốc độ có $K\_c \approx 0,0863$, $K\_i \approx 1,39$. Lưu ý đây là các giá trị trong mô hình chuẩn hóa. Khi lập trình, cần chuyển đổi $K\_c$ và $K\_i$ sang dạng phù hợp (ví dụ tính toán theo đơn vị xung encoder và chu kỳ tính toán 10 ms). Sau khi hiệu chỉnh nhỏ cho phù hợp thực tế, nhóm sử dụng $K\_p^{*} = 0,1$ và $K\_i^{*} = 1,4$ trong chương trình (các giá trị gần với tính toán lý thuyết).

## Kết quả mô phỏng và kết quả thực tế:

Nhóm đã mô phỏng hệ thống điều khiển tốc độ trên MATLAB/Simulink trước khi thử nghiệm thực tế. Hình 9 thể hiện mô hình mô phỏng, bao gồm khâu động cơ (mô hình bậc nhất) và bộ điều khiển PI với các tham số thiết kế. Đầu vào đặt là một tốc độ góc không đổi (hàm bước tốc độ), mô phỏng quá trình động cơ vọt lên đến tốc độ đặt rồi giữ ổn định.

A diagram of a mathematical equation

AI-generated content may be incorrect.

Kết quả mô phỏng (Hình 10) cho thấy tốc độ động cơ (đường màu vàng) đáp ứng hàm bước từ 0 lên 150 rad/s một cách nhanh chóng và hầu như không có sai số duy trì. Thời gian vọt lố rất ngắn và giá trị vọt lố nhỏ. Sau khoảng 0,1 s, tốc độ đã tiến rất gần tới giá trị đặt (đường màu xanh lá cây) và ổn định.

*Hình 10. Đáp ứng mô phỏng tốc độ động cơ (vạch vàng) đối với bước thay đổi tốc độ đặt (vạch xanh lá cây) – bộ điều khiển PI giúp động cơ đạt tốc độ mong muốn nhanh, vượt ít và ổn định.*

Tiến hành thí nghiệm trên hệ thống thực, nhóm cho động cơ quay ở một tốc độ đặt tương ứng (~150 rad/s). Dữ liệu tốc độ thực tế được thu thập qua UART và vẽ thành đồ thị (Hình 11). Kết quả thực tế cho thấy đáp ứng tốc độ của động cơ (đường màu trắng) tương đồng với mô phỏng: động cơ đạt tới tốc độ đặt (đường màu xanh dương) trong khoảng 0,1–0,2 s và giữ ổn định. Có một chút dao động nhỏ ban đầu do nhiễu và độ không tuyến tính, nhưng sau đó tốc độ duy trì khá đều quanh giá trị đặt. Điều này chứng tỏ bộ điều khiển PI đã thiết kế hoạt động hiệu quả trên đối tượng thực.

*Hình 11. Đáp ứng tốc độ của động cơ servo trên hệ thực nghiệm (đường màu trắng) khi áp dụng bước thay đổi tốc độ đặt (đường màu xanh dương). Hệ thống đạt tốc độ yêu cầu với sai số rất nhỏ và ổn định.*

# ĐIỀU KHIỂN VỊ TRÍ ĐỘNG CƠ DC SERVO SỬ DỤNG VI ĐIỀU KHIỂN STM32

Mục tiêu: Điều khiển vị trí trục động cơ DC servo bằng bộ điều khiển PID hiện thực trên vi điều khiển STM32F103C8. Động cơ servo DC một chiều tích hợp encoder phản hồi được điều khiển thông qua mạch cầu H. Vi điều khiển STM32 đọc tín hiệu encoder để xác định vị trí trục, tính toán bộ điều khiển PID và xuất PWM để điều khiển động cơ đến vị trí mong muốn.

## Kết cấu phần cứng:

Hình 1 mô tả mô-đun điều khiển sử dụng vi điều khiển STM32F103C8 trên board mạch và động cơ DC servo một trục trong hệ thống thí nghiệm. Động cơ servo DC loại DCM50205 có gắn encoder độ phân giải 1000 xung/vòng, được điều khiển thông qua driver cầu H trên board STM32 (sử dụng IC công suất L298N). Một trục quay của động cơ được nối với tải mô phỏng. Bảng 1 liệt kê các thông số kỹ thuật chính của động cơ servo DC này.

*Hình 1. (a) Board STM32F103C8 tích hợp driver cầu H điều khiển động cơ; (b) Động cơ DC servo một trục kèm encoder.*

Khối lượng tải nối với trục động cơ là khoảng *m* = 0,81774 kg (đĩa thép), bán kính *R* = 0,025 m. Quán tính tải quy về trục động cơ tính theo công thức $J = \frac{1}{2}mR^2$. Thay số liệu vào được:

Giá trị quán tính này nhỏ so với quán tính rotor động cơ (khoảng $3,11×10^{-5}$ kg·m²), do đó ảnh hưởng quán tính tải là không đáng kể trong mô hình hệ thống.

## Sơ đồ đấu nối phần cứng:

Board STM32F103C8 kết nối với các phần tử ngoại vi như sau: động cơ DC servo được điều khiển bởi cầu H tích hợp (kênh PWM điều khiển từ STM32), encoder gắn trên động cơ nối vào các chân ngắt ngoại vi của STM32 để đọc xung phản hồi, và cổng SWD dùng để nạp chương trình cho vi điều khiển. Hình 2 trình bày sơ đồ các chân I/O chính được sử dụng trên vi điều khiển STM32 và sơ đồ chân của cổng nạp SWD.

Cụ thể, các kênh Timer và ngắt được cấu hình như sau: tín hiệu PWM điều khiển động cơ được cấu hình trên chân **PC7** (ngõ ra Timer 3), các ngắt ngoài **EXTI** được sử dụng ở các chân **PC3** và **PC7** để đếm xung encoder (kênh A, B của encoder động cơ). Vi điều khiển nhận xung encoder và tính toán vị trí hiện tại của trục động cơ dựa trên số xung đếm được. Nguồn cấp cho động cơ và driver là 24 VDC; vi điều khiển STM32 được cấp nguồn 5 VDC qua cổng USB hoặc mạch ổn áp trên board.

## Chương trình STM32:

Chương trình điều khiển trên STM32F103C8 được viết bằng ngôn ngữ C (sử dụng STM32 HAL). Chức năng chính của chương trình gồm: đọc xung encoder để xác định vị trí, thực hiện thuật toán PID tính toán độ lệch và điều khiển PWM, và giao tiếp UART (USART1) để truyền dữ liệu (ví dụ gửi giá trị vị trí thực tế về PC để giám sát).

Một số phần chính trong chương trình:

* **Đọc tín hiệu encoder:** Encoder của động cơ cung cấp hai kênh tín hiệu vuông pha lệch (A và B). Các chân **PC3** và **PC7** của STM32 được cấu hình ngắt ngoài (EXTI) để bắt mỗi xung lên/xuống của encoder. Trong trình phục vụ ngắt (ISR), vi điều khiển kiểm tra trạng thái hai pha A/B để xác định hướng quay và tăng/giảm biến đếm vị trí count. Nhờ đó, biến count luôn chứa giá trị xung encoder tương ứng với vị trí góc của trục động cơ.
* **Điều khiển PWM:** Tín hiệu điều khiển động cơ là PWM được xuất bởi Timer 3 trên chân PC7 (kết nối đến driver cầu H). Chu kỳ PWM được cấu hình cố định (ví dụ 20 kHz), còn độ rộng xung (duty cycle) được điều chỉnh theo đầu ra bộ PID. Giá trị tính toán của bộ PID (điều khiển) sẽ được chuyển đổi sang độ rộng PWM phù hợp (trong khoảng 0-100% tương ứng 0-255 hoặc 0-1000 tùy độ phân giải) và cập nhật vào thanh ghi so sánh của Timer để thay đổi điện áp trung bình cấp cho động cơ.

Chương trình bao gồm các hàm xử lý ngắt và một vòng lặp chính. Trong vòng lặp chính (main loop), vi điều khiển có thể gửi dữ liệu vị trí hiện tại qua UART để hiển thị hoặc giám sát nếu cần. Bộ điều khiển PID cho vị trí được thực hiện trong ngắt Timer định kỳ (sử dụng Timer 4):

// Hàm xử lý ngắt Timer4 (chu kỳ 10ms):

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) {

if (htim->Instance == TIM4) {

// Đọc vị trí hiện tại từ encoder (count)

currentPos = count;

// Tính sai lệch vị trí

error = setpoint - currentPos;

// Tính toán điều khiển PID (position)

integral += error;

derivative = error - prevError;

output = Kp\*error + Ki\*integral + Kd\*derivative;

prevError = error;

// Giới hạn output trong [min,max]

if (output > MAX\_DUTY) output = MAX\_DUTY;

if (output < MIN\_DUTY) output = MIN\_DUTY;

// Cập nhật PWM điều khiển động cơ

\_\_HAL\_TIM\_SET\_COMPARE(&htim3, TIM\_CHANNEL\_2, output);

}

}

*Chú thích:* Trong đoạn code trên, setpoint là vị trí đặt (giá trị mong muốn, đơn vị xung encoder), currentPos là vị trí đo được (đếm xung encoder), error là sai lệch vị trí. Bộ điều khiển PID tính toán biến điều khiển output (giá trị độ rộng PWM). Biến output sau đó được dùng để cập nhật thanh ghi so sánh của Timer 3 kênh 2 (ứng với chân PC7 PWM). Chu kỳ ngắt Timer4 được chọn đủ nhanh (ví dụ 10 ms) để điều khiển vị trí nhạy và ổn định.

## Tính toán thông số PID:

Để thiết kế bộ điều khiển PID vị trí, nhóm đã áp dụng phương pháp thực nghiệm dựa trên đáp ứng hệ thống hở (vận tốc) và dùng quy trình điều chỉnh theo tiêu chuẩn Internal Model Control (IMC).

Trước tiên, tiến hành khảo sát đặc tính động học của trạm vận tốc (bỏ vòng điều khiển vị trí, chỉ giữ vòng điều khiển tốc độ bên trong). Tín hiệu điều khiển (biên độ PWM) được tăng từ mức 20 lên 96 (giá trị tương đối từ 0 đến 100%) và ghi nhận tốc độ góc trục động cơ trước và sau khi tăng. Kết quả đo được: tốc độ ban đầu $\omega\_1 \approx 27,96$ rad/s và tốc độ cuối $\omega\_2 \approx 204,20$ rad/s. Như vậy độ biến thiên tốc độ $\Delta \omega \approx 176,24$ rad/s khi biên độ PWM tăng $\Delta u = 76$ (đơn vị phần trăm PWM).

* **Hệ số khuếch đại quá trình:** $K\_p = \dfrac{\Delta \omega}{\Delta u} = \dfrac{176,24}{76} \approx 2,3189$ (rad/s trên mỗi đơn vị điều khiển).
* **Hằng số thời gian của hệ thống:** Thời gian đạt 63% độ biến thiên tốc độ khoảng $\tau\_{63%} = 0,062$ s (từ đồ thị đáp ứng tốc độ). Xấp xỉ hằng số thời gian quá trình $\tau\_p = 0,062$ s.
* **Mô hình hàm truyền của hệ thống vận tốc:** Với động cơ DC servo dạng hệ bậc nhất, hàm truyền hở (từ tín hiệu PWM $U(s)$ đến tốc độ $\omega(s)$) được xấp xỉ:

Đối với hệ thống điều khiển **vị trí**, do vị trí là tích phân của tốc độ, hàm truyền hở từ tín hiệu điều khiển đến vị trí $\Theta(s)$ sẽ có thêm một bậc tích phân (1/s):

Hơn nữa, cần đổi đơn vị tốc độ góc (rad/s) sang vị trí tuyến tính (mm) nếu quan tâm đến di chuyển thẳng của cơ cấu (trục vít me chuyển đổi quay thành tịnh tiến). Thí nghiệm cho thấy trục vít me có bước vít khoảng 10 mm cho mỗi vòng (2$\pi$ rad). Vì vậy, ta có thể nhân thêm hệ số chuyển đổi $\frac{10}{2\pi}$ để biểu diễn vị trí tính bằng mm. Khi đó hàm truyền vị trí xấp xỉ:

Với mô hình đối tượng trên, nhóm sử dụng tiêu chuẩn **IMC** để suy ra thông số điều khiển PID. Chọn hằng số thời gian mạch kín $\tau\_c = \frac{\tau\_p}{5} = \frac{0,062}{5} \approx 0,0124$ s (mong muốn hệ thống đáp ứng nhanh hơn nhiều so với thời gian trễ của quá trình). Bộ điều khiển PI (do hệ thống hạng nhất với tích phân nên chỉ cần PI) được thiết kế với các tham số:

* **Kc** (hệ số khuếch đại của bộ điều khiển) tính theo công thức IMC:
* **$\tau\_i$** (thời gian tích phân) thường chọn bằng $\tau\_p$ để bù trễ hệ thống, nên $\tau\_i \approx 0,062$ s.
* **Ki** (hệ số tích phân của bộ PI) được tính là $K\_c/\tau\_i$, do đó:

Như vậy, bộ điều khiển PI vị trí được thiết lập với $K\_c \approx 0,0863$, $K\_i \approx 1,39$ (và $K\_d = 0$ do không cần phần vi phân). Các thông số này sẽ được lập trình vào vi điều khiển STM32 để thực hiện điều khiển vị trí servo.

## Mô phỏng trên MATLAB/Simulink và kết quả thực tế:

Để kiểm chứng thiết kế, nhóm đã mô phỏng hệ thống servo vị trí trên MATLAB/Simulink với mô hình đối tượng động cơ và bộ điều khiển PID như tính toán ở trên. Hình 3 là mô hình mô phỏng, bao gồm khâu động cơ (hàm truyền đã xấp xỉ) và bộ điều khiển PID vị trí. Đầu vào đặt là một hàm bước vị trí (bước tín hiệu vị trí mục tiêu), và đầu ra theo dõi là vị trí trục động cơ.

*Hình 3. Mô hình mô phỏng hệ thống điều khiển vị trí động cơ DC servo trên Simulink.*

Kết quả mô phỏng cho thấy động cơ đạt được vị trí đặt nhanh với sai số xác lập xấp xỉ bằng 0. Cụ thể, Hình 4 biểu diễn đáp ứng vị trí của hệ thống khi áp dụng đầu vào bước (từ 0 lên vị trí đặt nhất định). Đường màu xanh lá cây thể hiện vị trí động cơ servo, có độ vọt lố nhỏ và ổn định về giá trị đặt (đường đứt nét màu đỏ) trong khoảng thời gian ngắn (vài trăm ms).

*Hình 4. Kết quả mô phỏng đáp ứng vị trí của động cơ (vạch xanh) đối với tín hiệu bước đặt vị trí (vạch đỏ); bộ PID được thiết kế giúp hệ thống đạt vị trí mong muốn nhanh và ổn định.*

Nhóm cũng tiến hành thực nghiệm trên hệ servo một trục thực tế với cùng thông số điều khiển PID. Kết quả thu được (Hình 5) cho thấy hệ thống thực tế đạt được vị trí đặt với thời gian đáp ứng khoảng dưới 0,5 s và hầu như không có sai số xác lập. Đường phản hồi vị trí đo được có dạng tương tự kết quả mô phỏng, mặc dù có dao động nhẹ hơn do ma sát và nhiễu ngoài trong hệ thống thực. Kết quả thực tế khẳng định bộ điều khiển PID thiết kế hoạt động tốt, động cơ servo dừng đúng vị trí yêu cầu.