ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

ROBOT SƠN CHỈ ĐƯỜNG DÙNG THỊ GIÁC MÁY TÍNH

Giáo viên hướng dẫn: **TS. NGUYỄN HOÀNG MAI**

Sinh viên thực hiện:

1. **ĐINH PHÚ GIANG – MSSV: 105200297 – LỚP: 20TDH1**
2. **ĐỖ TRỌNG VINH – MSSV: 105200396 – LỚP: 20TDHCLC1**
3. **NGÔ ĐẠT HUY – MSSV: 105200364 – LỚP: 20TDHCLC1**

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: Robot sơn chỉ đường dùng thị giác máy tính

Sinh viên thực hiện: Đinh Phú Giang

Số thẻ SV: 105200297 Lớp: 20TDH1

Sinh viên thực hiện: Đỗ Trọng Vinh

Số thẻ SV: 105200396 Lớp: 20TDHCLC1

Sinh viên thực hiện: Ngô Đạt Huy

Số thẻ SV: 105200364 Lớp: 20TDHCLC1

Nội dung tóm tắt:

Đề tài tập trung vào việc thiết kế và triển khai một hệ thống robot tự động sơn vạch chỉ đường dựa trên công nghệ thị giác máy tính. Hệ thống sử dụng hai camera để nhận diện đường line thép được bố trí sẵn trên mặt đất. Thông tin thu được từ camera được xử lý để xác định chính xác vị trí, hướng đi của xe và định tuyến đường sơn cần thực hiện. Robot được điều hướng theo dữ liệu hình ảnh nhằm bám sát tuyến đường, đồng thời cánh tay phun sơn sẽ tự động điều chỉnh để bù độ lệch do chuyển động của xe, đảm bảo vạch sơn song song và bám sát dây thép.

Toàn bộ hệ thống xử lý hình ảnh được lập trình bằng Python, sử dụng thư viện OpenCV để nhận diện đường line và tính toán quỹ đạo. Cánh tay robot được điều khiển thông qua vi điều khiển kết hợp thuật toán định vị và điều chỉnh động học. Đề tài mang tính ứng dụng cao trong các hệ thống đánh dấu, phân làn hoặc sơn đường trong môi trường công nghiệp và giao thông.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HÔI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Đinh Phú Giang	105200297	20TDH1	Kỹ thuật điều khiển và tự động hóa
	Đỗ Trọng Vinh	105200396	20TDHCLC1	Kỹ thuật điều khiển và tự động hóa (CLC)
	Ngô Đạt Huy	105200364	20TDHCLC1	Kỹ thuật điều khiển và tự động hóa (CLC)

1. *Tên đề tài đồ án:*

   Robot sơn chỉ đường dùng thị giác máy tính

2. *Đề tài thuộc diện: ☐Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*
4. *Nội dung các phần thuyết minh và tính toán:*

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Đinh Phú Giang	Tìm hiểu yêu cầu của đề tài, các phương pháp điều khiển cho từng thành phần của Xe robot Tìm hiểu quy trình vận hành tổng quát của Xe robot Lựa chọn các thiết bị cần thiết cho hệ thống Triển khai, thực nghiệm và hoàn thiện mô hình. Hoàn thiện báo cáo
2	Đỗ Trọng Vinh	
3	Ngô Đạt Huy	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
	Đinh Phú Giang	- Tìm hiểu về thị giác máy tính, thực hiện thu thập dữ liệu, train và test mô hình thị giác máy tính - Triển khai kết nối và hiệu phần cứng của hệ thống thị giác - Tìm hiểu về cách truyền dữ liệu từ laptop sang arduino - Tính toán và lập trình xác định được dữ liệu cần để truyền sang arduino điều khiển Xe robot
	Ngô Đạt Huy	- Tìm hiểu về điều hướng xe Robot

		- Triển khai kết nối phần cứng (Động cơ DC điều hướng xe, encoder vào các thành phần điều khiển) - Tính toán phương trình động học động cơ DC điều hướng, thực hiện mô phỏng trên Matlab - Tính toán, lập trình bộ điều khiển PID và hiệu chỉnh thực nghiệm động cơ DC điều hướng
	Đỗ Trọng Vinh	- Tìm hiểu về cánh tay 2 bậc tự do - Triển khai kết nối phần cứng (Động cơ bước vào các thành phần điều khiển) - Tính toán phương trình động học của cánh tay robot - Tính toán, lập trình bộ điều khiển vòng hở và hiệu chỉnh thực nghiệm cánh tay robot

5. Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):

a. Phần chung:

TT	Họ tên sinh viên	Nội dung

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Ngô Đạt Huy	- Sơ đồ khối hệ thống - Sơ đồ đấu dây chi tiết

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
Nguyễn Hoàng Mai	- Hướng dẫn, định hướng phát triển mô hình - Hướng dẫn tính toán bài toán điều khiển cánh tay Robot - Hiệu chỉnh báo cáo

7. Ngày giao nhiệm vụ đồ án: 21/2/2025

8. Ngày hoàn thành đồ án: 17/6/2025

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Đinh Phú Giang_____ Số thẻ SV: 105200297_____

Họ tên sinh viên: Đỗ Trọng Vinh_____Số thẻ SV: 105200396_____

Họ tên sinh viên: Ngô Đạt Huy_____ Số thẻ SV: 105200364_____

Tên đề tài ĐATN: Robot sơn chỉ đường dùng thị giác máy tính_____

Họ tên người HD: TS. Nguyễn Hoàng Mai___ Đơn vị: Khoa Điện – Trường Đại học Bách Khoa – Đại học Đà Nẵng

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	24/2	Tìm hiểu tổng quan về đề tài		
2	3/3	Tìm hiểu sơ lược về thị giác máy tính, xe tự hành và cánh tay robot		
3	10/3	Đặt ra bài toán dựa trên mô hình xe Robot có sẵn và mục tiêu mong muốn đạt được		
4		Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	17/3	Hiệu chỉnh lại phần cơ khí để phù hợp với điều kiện thực tế	5%	
6	24/3	Tiến hành lựa chọn và mua các thiết bị còn thiếu và đấu nối phần cứng cho bộ điều khiển	10%	
7	31/3	Tìm hiểu cách train mô hình thị giác máy tính, thực hiện tính toán động lực học cho từng thành phần và tiến hành mô phỏng Matla	15%	
8		Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	7/4	Train và test mô hình thị giác, vẽ lưu đồ thuật toán và lập trình điều khiển	20%	

10	14/4	- Tính toán và lập trình để tìm dữ liệu và góc lệch của dây thép so với xe dựa vào mô hình thị giác máy tính đã train - Thực hiện truyền giả lập dữ liệu để thực nghiệm điều hướng xe và điều khiển cánh tay robot	30%	
11	21/4	- Tiến hành thực nghiệm ở điều kiện thực tế	40%	
12		Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	28/4	Tiến hành thực nghiệm, cải thiện mô hình thị giác, hiệu chỉnh các thông số điều khiển các động cơ	60%	
14	5/5	Tiếp tục thực nghiệm, tối ưu hóa mô hình	80%	
15	5/6	Thực hiện viết báo cáo và tiếp tục hiệu chỉnh mô hình	100%	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trong suốt quá trình thực hiện và hoàn thành đề tài tốt nghiệp "Sơn chỉ đường dùng thị giác máy tính", em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Nguyễn Hoàng Mai – người đã trực tiếp hướng dẫn, tận tình chỉ bảo và đồng hành cùng em trong suốt quá trình nghiên cứu. Những góp ý chuyên môn và sự định hướng của Thầy là yếu tố quan trọng giúp em từng bước vượt qua các khó khăn để hoàn thiện đề tài.

Em cũng xin trân trọng cảm ơn quý thầy cô trong Khoa Điện – Trường Đại học Bách Khoa – Đại học Đà Nẵng đã truyền đạt cho em những kiến thức quý báu trong suốt thời gian học tập, là nền tảng giúp em có thể thực hiện và hoàn thành đề tài này.

Mặc dù đã có nhiều nỗ lực và cố gắng, đề tài vẫn không tránh khỏi những thiếu sót nhất định. Em rất mong nhận được sự góp ý từ quý thầy cô để có thể hoàn thiện hơn và nâng cao năng lực bản thân trong các nghiên cứu tiếp theo.

Em xin chân thành cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Em xin cam đoan đề tài “Sơn chỉ đường dùng thị giác máy tính” là kết quả nghiên cứu và làm việc của chính em dưới sự hướng dẫn của Thầy Nguyễn Hoàng Mai.

Toàn bộ nội dung, số liệu, hình ảnh và kết quả trong báo cáo là trung thực, không sao chép hay sử dụng trái phép từ bất kỳ công trình nào khác. Các tài liệu tham khảo được trích dẫn đầy đủ và tuân thủ đúng quy định.

Người cam đoan

Sinh viên 1 Sinh viên 2 Sinh viên 3

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

MỞ ĐẦU

Đề tài *"Robot sơn chỉ đường dùng thị giác máy tính"* được thực hiện nhằm thiết kế và chế tạo một robot tự hành có khả năng hoạt động ngoài trời trên mặt đường nhựa, tự động vẽ các vạch chỉ đường giao thông thông qua nhận dạng hình ảnh từ camera. Mục tiêu của đề tài là xây dựng một hệ thống robot tích hợp được khả năng xử lý ảnh, bám theo đường tham chiếu và điều khiển cơ cấu sơn chính xác, từ đó góp phần hiện đại hóa và tự động hóa quy trình sơn vạch giao thông vốn còn thủ công và nguy hiểm hiện nay.

Phạm vi nghiên cứu tập trung vào môi trường ngoài trời với bề mặt đường nhựa và điều kiện ánh sáng thực tế. Đối tượng nghiên cứu gồm robot di động điều khiển bằng vi điều khiển, hệ thống thị giác máy tính dùng camera, thuật toán điều khiển hướng đi và cơ cấu sơn. Phương pháp nghiên cứu được sử dụng bao gồm khảo sát thực tiễn, nghiên cứu lý thuyết, thiết kế – chế tạo phần cứng, lập trình điều khiển và thực nghiệm đánh giá hiệu quả.

Cấu trúc của đồ án được trình bày qua bốn chương chính:

Chương 1: Tổng quan đề tài;

Chương 2: Cấu trúc hệ thống và nguyên lý hoạt động;

Chương 3: Cơ sở lý thuyết và các tính toán thiết kế;

Chương 4: Mô hình thực nghiệm và đánh giá.

Chương 1: TỔNG QUAN ĐỀ TÀI

1.1. Bối cảnh chung và hiện trạng giao thông tại Việt Nam

Trong những năm gần đây, Việt Nam chứng kiến tốc độ đô thị hóa nhanh chóng, cùng với sự phát triển mạnh mẽ về hạ tầng giao thông và phương tiện vận tải. Tuy nhiên, đi kèm với đó là nhiều thách thức đáng kể về quản lý giao thông đô thị, an toàn lao động trong thi công, và tối ưu hóa hiệu quả thi công hạ tầng. Một trong những công đoạn nhỏ nhưng đóng vai trò quan trọng là việc sơn vạch kẻ đường, giúp phân luồng, điều tiết và đảm bảo an toàn giao thông.

Hiện nay, công tác kẻ vạch đường tại Việt Nam chủ yếu vẫn được thực hiện theo phương pháp thủ công hoặc bán tự động. Công nhân phải vận hành trực tiếp máy móc, điều khiển hướng di chuyển và kiểm soát lượng sơn. Quá trình này diễn ra ngay trên lòng đường đang lưu thông, tiềm ẩn nhiều rủi ro về tai nạn giao thông và tiếp xúc với hóa chất độc hại trong môi trường làm việc.

Theo Nghị định số 168/2021/NĐ-CP ban hành ngày 31/12/2021 của Chính phủ, các đơn vị thi công cần tuân thủ nghiêm ngặt quy định về an toàn giao thông, không gây cản trở hoặc mất an toàn cho người và phương tiện lưu thông trong quá trình thi công. Tuy nhiên, trên thực tế, nhiều công trình kẻ vạch không đáp ứng được các tiêu chuẩn nêu trên do thiếu thiết bị hiện đại và phụ thuộc quá nhiều vào yếu tố con người.

Một số hạn chế cụ thể trong thi công kẻ vạch đường:

- Tính thủ công cao: Phụ thuộc vào tay nghề và sự phối hợp giữa các công nhân, dễ xảy ra sai sót, đặc biệt ở các đoạn cong hoặc giao lộ.
- Tốc độ thi công chậm: Một nhóm công nhân chỉ có thể kẻ một đoạn đường ngắn mỗi ngày, gây ảnh hưởng đến tiến độ tổng thể.
- Nguy cơ mất an toàn cao: Công nhân phải làm việc giữa dòng xe cộ, tiếp xúc với sơn hóa học có chứa dung môi dễ bay hơi.
- Thiếu tính linh hoạt: Khó thay đổi thiết kế vạch theo yêu cầu mới, cần thời gian dừng máy và điều chỉnh thủ công.
- Chi phí nhân công cao: Do yêu cầu về số lượng người lao động, thiết bị bảo hộ, thời gian thi công kéo dài.

Hình 1.1. Kẻ vạch đường thủ công

Hình 1.2. Kẻ vạch đường bán tự động

Theo báo cáo của Bộ Lao động – Thương binh và Xã hội năm 2022, ngành giao thông vận tải là một trong những ngành có số vụ tai nạn lao động cao, với hàng trăm ca mỗi năm, trong đó không ít tai nạn xảy ra trong quá trình thi công sơn vạch đường. Tại các đô thị lớn như Hà Nội, TP.HCM, nơi có mật độ giao thông dày đặc, rủi ro càng tăng cao.

1.2. Cơ sở khoa học và xu thế công nghệ

Trong thời đại công nghiệp 4.0, robot tự hành, thị giác máy tính (computer vision) và hệ thống điều khiển thông minh đang được ứng dụng ngày càng rộng rãi trong các

lĩnh vực như sản xuất công nghiệp, nông nghiệp, giao thông, và dịch vụ đô thị. Thị giác máy tính cho phép máy móc "nhìn thấy" và phân tích hình ảnh tương tự như con người, trong khi các hệ thống điều khiển có thể xử lý dữ liệu và ra quyết định theo thời gian thực.

Robot tự hành trong giao thông là xu hướng tất yếu nhằm giảm tải cho lao động con người, đồng thời tăng hiệu suất và độ chính xác. Các quốc gia phát triển đã áp dụng robot vào việc sơn vạch, rửa đường, thu gom rác, kiểm tra mặt đường,... Việc xây dựng một hệ thống xe kẻ vạch tự động tại Việt Nam không chỉ phù hợp với xu thế công nghệ, mà còn có tính thực tiễn cao, dễ triển khai tại các địa phương.

Công nghệ lõi đề tài ứng dụng bao gồm:

- Camera kép định vị vạch (dây thép): Sử dụng thuật toán phát hiện đường cong hoặc biên sắc nét để định vị dây thép làm đường chuẩn.
- Bộ điều khiển trung tâm: Phân tích hình ảnh từ camera, tính toán sai số và gửi lệnh điều khiển đến các động cơ.
- Thuật toán PID: Điều khiển chính xác góc lái và tốc độ động cơ DC để xe bám theo đường.
- Cánh tay robot điều khiển động cơ bước: Phun sơn theo điều hướng và độ cong của đường dẫn, đảm bảo chính xác vị trí, tốc độ và lượng sơn.
- Giao tiếp thời gian thực PC–Arduino: Kết hợp giữa xử lý mạnh của máy tính và phản hồi nhanh của vi điều khiển.

1.3. Mục tiêu nghiên cứu

Đề tài hướng đến việc xây dựng một hệ thống xe kẻ vạch tự hành dựa trên mô hình đã có sẵn có khả năng nhận diện đường đi, điều khiển chính xác, linh hoạt và có thể hoạt động độc lập, không cần sự can thiệp của con người trong quá trình thi công.

Các mục tiêu cụ thể:

- Phá triển hệ thống điều khiển tích hợp giữa camera, bộ xử lý trung tâm và động cơ.
- Ứng dụng thuật toán xử lý ảnh thời gian thực để nhận diện vạch hướng dẫn.
- Tính toán bộ điều khiển, xây dựng thuật toán và lập trình điều khiển 2 động cơ DC điều hướng và 2 động cơ bước điều khiển cánh tay robot tối ưu
- Đảm bảo khả năng hoạt động trong nhiều điều kiện khác nhau, bao gồm đường cong, gấp khúc, bề mặt không đồng đều.
- Đánh giá, thử nghiệm và tối ưu hệ thống dựa trên tiêu chí hiệu suất, độ chính xác và độ tin cậy.

1.4. Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học:

- Góp phần vào nghiên cứu và ứng dụng thị giác máy tính trong điều khiển tự động.

- Áp dụng thực tế thuật toán PID vào hệ thống cơ điện tử phức hợp.
- Tạo tiền đề cho các nghiên cứu mở rộng về xe tự hành, robot thi công, và hệ thống điều khiển phân tán.

Ý nghĩa thực tiễn:

- Tăng năng suất thi công, giảm chi phí nhân lực và nguy cơ tai nạn lao động.
- Cung cấp giải pháp hiệu quả cho các đơn vị quản lý hạ tầng giao thông, đặc biệt ở các đô thị lớn.
- Có thể mở rộng ứng dụng vào các công việc khác như vẽ vạch sân thể thao, đánh dấu khu công nghiệp, chăm sóc cây xanh theo hàng.
- Góp phần thúc đẩy tiến trình hiện đại hóa ngành giao thông theo định hướng phát triển đô thị thông minh tại Việt Nam.

1.5. Phạm vi nghiên cứu và định hướng phát triển

Phạm vi:

- Nghiên cứu được triển khai ở cấp mô hình thực nghiệm, trong môi trường kiểm soát (phòng lab, sân thử).
- Vạch đường được mô phỏng bằng dây thép màu nổi bật (mô hình đại diện).
- Cánh tay robot hoạt động trên mặt phẳng cố định, chưa có điều kiện thử nghiệm trên địa hình phức tạp thực tế.

Định hướng phát triển:

- Tích hợp thêm cảm biến khoảng cách (LiDAR, siêu âm) để hỗ trợ phát hiện vật cản.
- Tăng độ chính xác bằng cách sử dụng AI (machine learning) trong phân tích ảnh.
- Sử dụng GPS để định vị và lập lịch trình thi công theo bản đồ số.
- Tăng tính cơ động: thêm khả năng leo dốc, xử lý địa hình gồ ghề.
- Xây dựng giao diện điều khiển từ xa bằng ứng dụng di động hoặc nền tảng web.

1.6. Kết luận

Trong bối cảnh chuyển đổi số và hiện đại hóa hạ tầng giao thông tại Việt Nam, việc áp dụng các công nghệ tiên tiến như thị giác máy tính, robot tự hành và điều khiển tự động vào thi công là một hướng đi tất yếu. Đề tài "Xe kẻ vạch đường tự động sử dụng thị giác máy tính phát hiện dây thép" là một bước tiến quan trọng, không chỉ giúp nâng cao hiệu suất lao động, mà còn góp phần cải thiện chất lượng, độ chính xác và an toàn trong lĩnh vực thi công giao thông.

Đây là mô hình nghiên cứu có tính ứng dụng cao, phù hợp với điều kiện thực tế tại Việt Nam, đồng thời tạo nền tảng cho các nghiên cứu tiếp theo trong lĩnh vực robot tự hành, thị giác máy tính và tự động hóa thông minh.

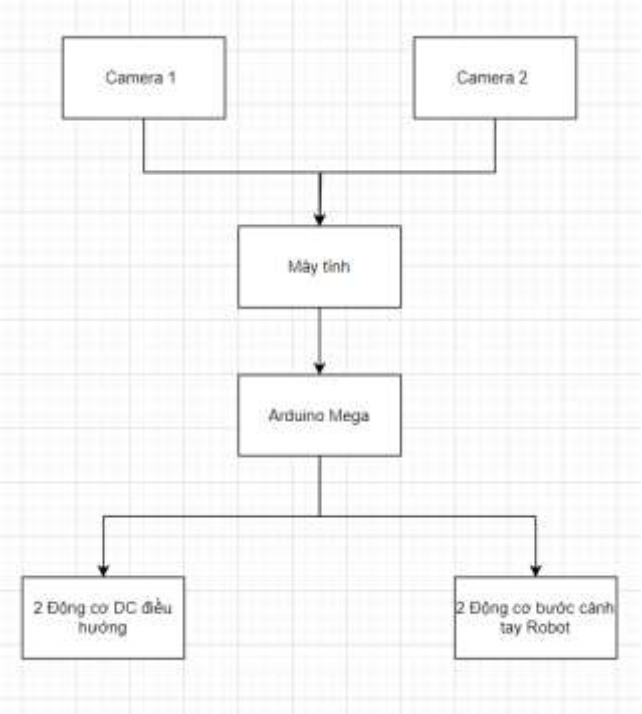
Chương 2: CẤU TRÚC HỆ THỐNG VÀ NGUYÊN LÝ HOẠT ĐỘNG

2.1. Cấu trúc hệ thống

Sơ đồ khối hệ thống

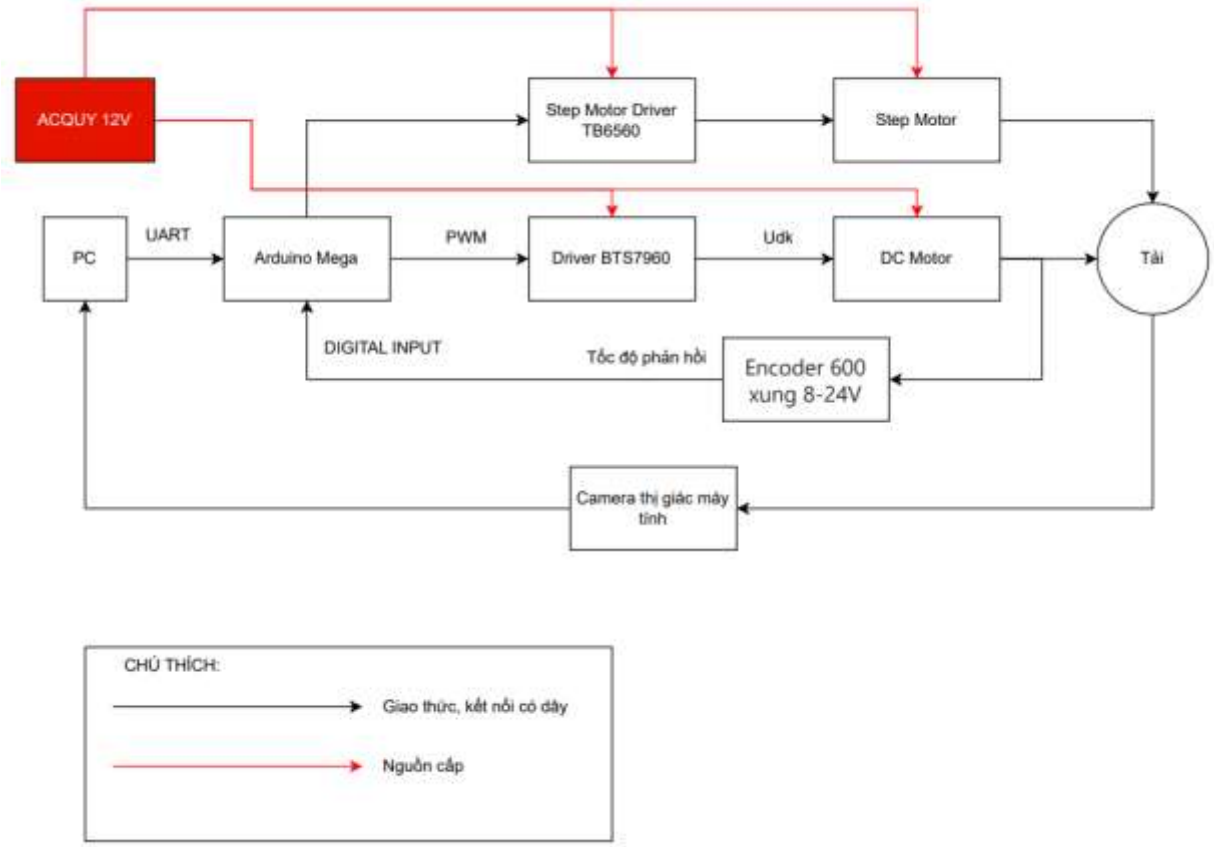
- Xe di chuyển tự hành:
  - 4 bánh xe với 2 động cơ DC điều hướng (điều khiển trái/phải).
- Thị giác máy tính:
  - 2 camera trước và sau thu thập dữ liệu vị trí dây thép.
- Cánh tay robot kẻ vạch:
  - 2 động cơ bước điều khiển cánh tay 2 bậc tự do.
- Bộ xử lý trung tâm:
  - Máy tính phân tích hình ảnh, tính toán tọa độ sai lệch và gửi lệnh điều chỉnh qua Serial tới Arduino Mega.
- Arduino Mega:
  - Điều khiển động cơ DC bằng PID.
  - Điều khiển động cơ bước bằng thuật toán nội suy tuyến tính kết hợp điều khiển bằng Timer.

- Sơ đồ mô tả:



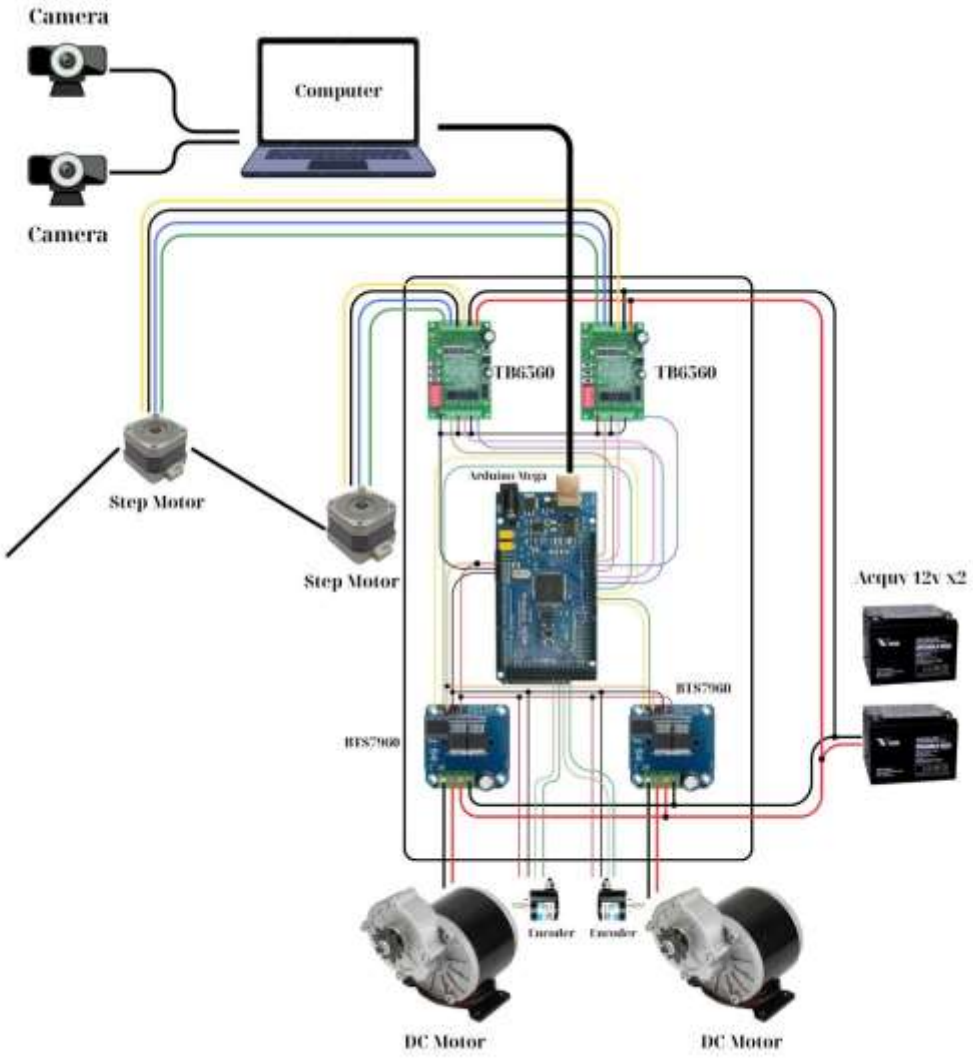
Hình 2.1. Sơ đồ mô tả hệ thống

Sơ đồ khối :



Hình 2.2. Sơ đồ khối hệ thống

Sơ đồ đấu dây chi tiết:



Hình 2.3. Sơ đồ đấu dây chi tiết

2.2. Nguyên lý hoạt động

Hệ thống sử dụng hai camera được lắp đặt ở đầu và cuối xe để liên tục thu thập các khung hình từ môi trường. Các hình ảnh này được truyền về máy tính nhúng, nơi tích hợp mô hình đã được huấn luyện bằng công cụ YOLOv8 nhằm phát hiện chính xác vị trí dây thép định hướng. Bên cạnh đó, các thuật toán xử lý ảnh được áp dụng để tính toán góc lệch và khoảng cách lệch giữa xe và dây thép.

Sau khi xử lý, dữ liệu bao gồm góc lệch và khoảng cách lệch sẽ được truyền qua giao thức Serial tới Arduino Mega, chịu trách nhiệm điều khiển toàn bộ hệ thống cơ điện của xe.

Xe robot sử dụng hai động cơ DC gắn vào hai bánh trước, đảm nhiệm việc kiểm soát tốc độ và điều hướng. Hai bánh sau được thiết kế tự do, có khả năng xoay 360 độ, nhằm tăng tính linh hoạt trong quá trình di chuyển, đồng thời đảm bảo sự ổn định khi xe thay đổi hướng.

Ngoài ra, hệ thống được trang bị một cánh tay robot hai bậc tự do, với gốc đặt tại trung tâm khung xe. Cánh tay sử dụng hai động cơ bước cho phép điều chỉnh linh hoạt các khớp, đảm bảo đầu phun sơn luôn bám sát vị trí dây thép cần vẽ, ngay cả khi quỹ đạo di chuyển thay đổi.

Dữ liệu góc lệch và khoảng cách lệch sau khi được Arduino xử lý sẽ được sử dụng để điều chỉnh tốc độ tương đối của hai động cơ DC, giúc xe tự động căn chỉnh và bám sát đường dây thép. Đồng thời, hệ thống cũng tính toán vị trí đích cho đầu phun sơn thông qua các thuật toán nghịch động học, nhằm đảm bảo độ chính xác cao trong quá trình kẻ vạch.

2.3. Nguyên lý điều khiển

2.3.1. Động cơ DC điều hướng

- Ban đầu cho một tốc độ đặt cố định để xe bắt đầu di chuyển, sau khi xe đã di chuyển bắt đầu sử dụng bộ điều khiển PID vòng kín để điều khiển tốc độ cho mỗi động cơ ở mỗi bánh xe để có thể điều khiển hướng đi của xe. Camera thị giác máy tính sẽ thu thập dữ liệu và đưa về bộ điều khiển để tính toán ra tốc độ đặt cho mỗi động cơ của mỗi bánh cho phù hợp với tình huống hiện tại.

- Dữ liệu thu về bao gồm khoảng cách 1 và khoảng cách 2, chúng ta sử dụng luân phiên cả 2 dữ liệu thu về.

- Trường hợp 1: Khoảng 1 thu về được giá trị như sau

+ Nếu d1 > 0 → Xe lệch trái → Tốc độ bánh trái sẽ tăng lên, giảm tốc độ bánh phải để giữ ổn định tốc độ trung bình của xe.

+ Nếu d1 < 0 → Xe lệch phải → Tốc độ bánh phải sẽ tăng lên, giảm tốc độ bánh trái để giữ ổn định tốc độ trung bình của xe.

- Trường hợp 2: Camera 1 bị lệch ra khỏi đường dây và không gửi dữ liệu khoảng cách 1 về bộ điều khiển, sử dụng khoảng cách 2:

+ Nếu d2 < 0 → Xe lệch trái → Tốc độ bánh trái sẽ tăng lên, giảm tốc độ bánh phải để giữ ổn định tốc độ trung bình của xe.

+ Nếu d2 > 0 → Xe lệch phải → Tốc độ bánh phải sẽ tăng lên, giảm tốc độ bánh trái để giữ ổn định tốc độ trung bình của xe.

- Sử dụng bộ PID để điều khiển tốc độ đầu ra cho động cơ:

+ Đặt tốc độ mong muốn là Setpoint

+ Encoder sẽ trả về tốc độ hiện tại của động cơ và tính ra sai số

+ Bộ điều khiển PID sẽ tính toán ra xung cần thiết để cấp cho động cơ

+ Khi động cơ thay đổi tốc độ, Encoder sẽ đọc tốc độ và trả về lại tiếp tục điều khiển vòng kín.

2.3.2. Thuật toán điều khiển cánh tay Robot

Cánh tay gồm hai khâu với tham số Denavit-Hartenberg (DH):

Frame No.	ai	αi	di	qi
1	L1	0	0	q1
2	L2	0	0	q2

Trong đó:

- L1 và L2 là độ dài của khâu 1 và khâu 2.

- q1 và q2 là góc khớp quay của khâu 1 và khâu 2, điều khiển hướng và vị trí của chấp hành cuối.

Ma trận biến đổi thuần nhất

Từ gốc đến khớp 1:

$$^0T_1 = \begin{bmatrix} cos\ q_1 & -sin\ q_1 & 0 & L_1 cos\ q_1 \\ sin\ q_1 & cos\ q_1 & 0 & L_1 sin\ q_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

Từ khớp 1 đến khớp 2:

$$^1T_2 = \begin{bmatrix} cos\ q_2 & -sin\ q_2 & 0 & L_2 cos\ q_2 \\ sin\ q_2 & cos\ q_2 & 0 & L_2 sin\ q_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Ma trận tổng hợp: $^0T_2 = {}^0T_1\ {}^1T_2$

Dựa trên $^0T_2$ ta có vị trí đầu cuối của robot:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} L_1 cos\ q_1 + L_2 \cos(q_1 + q_2) \\ L_1 sin\ q_1 + L_2 \sin(q_1 + q_2) \end{bmatrix} \quad (2.3)$$

Nguyên lý điều khiển cho cánh tay robot (dùng động cơ bước - stepper motor, điều khiển vị trí end effector):

Điều khiển hở (Open-loop):

Dùng động cơ bước để đếm số bước quay → xác định góc quay và vị trí của từng khớp.

Mỗi bước tương ứng với 1 góc quay nhỏ cố định, sử dụng Microstep để

Lệnh điều khiển là số bước cần quay từ vị trí hiện tại đến vị trí mong muốn.

Sử dụng động học nghịch để tính ra góc q1, q2 cần thiết để quay , sau đó chuyển đổi thành số bước cần thiết để động cơ có thể quay bằng công thức :

step1=q1×STEPS_PER_DEGREE.

Trong đó : STEPS_PER_DEGREE là số bước động cơ cần di chuyển để quay được 1 độ.

2.4. Phương trình động lực học

2.4.1. Động cơ DC điều hướng

- Xe trong quá trình di chuyển được mô hình hóa như một hệ thống động học bậc hai, mô hình của xe khi chuyển động tịnh tiến:

$$m \times a = F_k - \left(F_{ms} + F_{truot}\right) \tag{2.4}$$

Trong đó:

m: Khối lượng xe (kg)

a: Gia tốc của xe (m/s^2)

F_k : Lực kéo do động cơ tạo ra tại 2 bánh trước

F_{ms} : Lực ma sát bánh xe

F_{truot} : Lực trượt trên mặt dốc khi di chuyển trên mặt phẳng nghiêng góc α

- Lực kéo từ bánh cố định (F_L, F_R): Lực do động cơ tạo ra tại hai bánh trái/ phải.

$$F_L = \frac{\tau_L}{R} \tag{2.5}$$

$$F_R = \frac{\tau_R}{R} \tag{2.3}$$

Trong đó: τ_L, τ_R: Mô-men xoắn động cơ trái/phải (Nm).

R: Bán kính bánh xe cố định (m)

- Lực ma sát giữa bánh xe và mặt đường gây ra:

$$F_{ms} = \mu mg \tag{2.6}$$

Trong đó: μ: Hệ số ma sát của bánh tự do.

m: Khối lượng xe (kg).

g: Gia tốc trọng trường (9.81 m/s^2).

- Lực trượt trên mặt dốc ($F_{trượt}$): Nếu di chuyển trên mặt phẳng nghiêng với góc nghiêng α

$$F_{truot} = mg \sin\alpha \tag{2.5}$$

- Công suất yêu cầu của xe được tính:

$$P = F_t \times v \tag{2.6}$$

Mô hình toán học của động cơ:

Phương trình điện áp phần ứng:

$$V_a(t) = e_a(t) + R_a i_a(t) + L_a \frac{di_a(t)}{dt} \tag{2.7}$$

Biến đổi Laplace:

$$V_a(s) = E_a i_a(s) + L_a s I_a(s) \tag{2.8}$$

Nên:

$$I_a(s) = \frac{V_a(s) - E_b(s)}{L_a(s) + R_a} \tag{2.9}$$

Lực điện động của động cơ:

$$e_a(t) = K_e \phi \omega(t) \tag{2.10}$$

Biến đổi Laplace:

$$E_a(s) = K_e \phi \omega(s) \tag{2.11}$$

Phương trình mô-men xoắn điện từ:

$$M_e(t) = K_M \phi I_a(t) \tag{2.12}$$

Biến đổi Laplace:

$$M_e(s) = K_M \phi I_a(s) \tag{2.13}$$

Phương trình đặc tính cơ:

$$M_e(s) - M_c(s) = J \frac{d\omega(t)}{dt} \tag{2.14}$$

Biến đổi Laplace:

$$M_e(s) - M_c(s) = Js\omega(s) \tag{2.15}$$

Nên:

$$\omega(s) = \frac{K_M \phi I_a}{dt} - \frac{1}{J} F(s) \tag{2.16}$$

2.4.2. Cánh tay Robot 2DOF

Phương trình động lực học Lagrange

$$K_1 = \frac{1}{2} I_1 \dot{q}_1^2 = \frac{1}{6} m_1 l_1^2 \dot{q}_1^2$$

$$\begin{cases} x_2 = l_1 \cos q_1 + \frac{1}{2} l_2 \cos q_2 \\ y2 = l_1 \sin q_1 + \frac{1}{2} l_2 \sin q_2 \end{cases}$$

$$\Rightarrow \begin{cases} \dot{x}_2 = -l_1 \dot{q}_1 \sin q_1 - \frac{1}{2} l_2 \dot{q}_2 \sin q_2 \\ \dot{y}_2 = l_1 \dot{q}_1 \cos q_1 + \frac{1}{2} l_2 \dot{q}_2 \cos q_2 \end{cases}$$

$$\begin{cases} x_3 = l_1\cos q_1 + l_2\cos q_2 \\ y_3 = l_1\sin q_1 + l_2\sin q_2 \end{cases} \Rightarrow \begin{cases} \dot{x}_3 = -l_1\dot{q}_1\sin q_1 - l_2\dot{q}_2\sin q_2 \\ \dot{y}_3 = l_1\dot{q}_1\cos q_1 + l_2\dot{q}_2\cos q_2 \end{cases}$$

$$K_2 = m_2 l_1^2\dot{q}_1^2 + \frac{7}{24}m_2 l_2^2\dot{q}_2^2 + \frac{1}{2}m_2 l_1\dot{q}_1 l_2\dot{q}_2\cos(q_1 - q_2) + m_3 l_1^2\dot{q}_1^2 + m_3 l_2^2\dot{q}_2^2$$

$$+\frac{1}{2}m_3 l_2^2\dot{q}_1^2 + \frac{1}{2}m_2 l_1 l_2\cos q_2\dot{q}_1^2 + m_3 l_1 l_2\cos q_2\dot{q}_1^2 + m_3 l_1\dot{q}_1 l_2\dot{q}_2\cos(q_1 - q_2) + \frac{1}{8}m_2 l_2^2\dot{q}_1^2$$

(2.17)

Phương trình Lagrange:

$$L = K_1 + K_2$$

$$= m_2 l_1^2\dot{q}_1^2 + \frac{7}{24}m_2 l_2^2\dot{q}_2^2 + \frac{1}{2}m_2 l_1\dot{q}_1 l_2\dot{q}_2\cos(q_1 - q_2) + m_3 l_1^2\dot{q}_1^2 + m_3 l_2^2\dot{q}_2^2 + m_3 l_1\dot{q}_1 l_2\dot{q}_2\cos(q_1 - q_2)$$

$$+\frac{1}{8}m_2 l_2^2\dot{q}_1^2 + \frac{1}{2}m_2 l_1 l_2\cos q_2\dot{q}_1^2 + m_3 l_1 l_2\cos q_2\dot{q}_1^2 + \frac{1}{6}m_1 l_1^2\dot{q}_1^2$$

(2.18)

Từ đó ta có phương trình động lực trên hai khớp robot:

$$F_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{q}_1}\right) - \frac{\partial L}{\partial q_1}$$

$$= \left(\frac{1}{3}m_1 l_1^2 + 2m_2 l_1^2 + 2m_3 l_1^2 + \frac{1}{4}m_2 l_2^2 + m_3 l_2^2 + m_2 l_1 l_2\cos q_2 + 2m_3 l_1 l_2\cos q_2\right)\ddot{q}_1$$

$$+\left(\frac{1}{2}m_2 l_1 l_2\cos(q_1 - q_2) + m_3 l_1 l_2\cos(q_1 - q_2)\right)\ddot{q}_2 - \frac{1}{2}m_2 l_1 l_2\dot{q}_2(\dot{q}_1 - \dot{q}_2)\sin(q_1 - q_2)$$

$$-m_3 l_1 l_2\dot{q}_2(\dot{q}_1 - \dot{q}_2)\sin(q_1 - q_2) - m_2 l_1 l_2\dot{q}_1\dot{q}_2\sin q_2 - 2m_3 l_1 l_2\dot{q}_1\dot{q}_2\sin q_2 + l_1\dot{q}_1 l_2\dot{q}_2\sin(q_1 - q_2)\left[m_3 + \frac{1}{2}m_2\right]$$

(2.19)

$$F_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial\dot{q}_2}\right) - \frac{\partial L}{\partial q_2}$$

$$= \left(\frac{1}{2}m_2 l_1\cos(q_1 - q_2) + m_3 l_1 l_2\cos(q_1 - q_2)\right)\ddot{q}_1 + \left(\frac{7}{12}m_2 l_2^2 + 2m_3 l_2^2\right)\ddot{q}_2$$

$$-\frac{1}{2}m_2 l_1\dot{q}_1 l_2(\dot{q}_1 - \dot{q}_2)\sin(q_1 - q_2) - m_3 l_1\dot{q}_1 l_2(\dot{q}_1 - \dot{q}_2)\sin(q_1 - q_2)$$

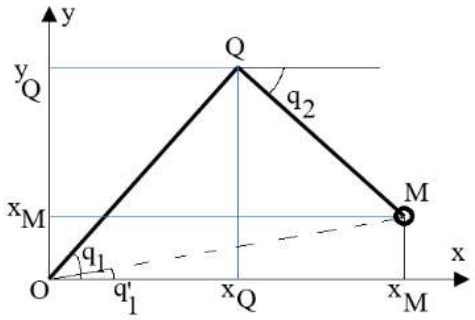
$$-\left(\frac{1}{2}m_2 + m_3\right)l_1\dot{q}_1 l_2\dot{q}_2\sin(q_1 - q_2) + \left(\frac{1}{2}m_2 + m_3\right)l_1 l_1 l_2\sin q_2\dot{q}_1^2$$

(2.20)

Ta có thể viết lại các phương trình động lực theo dạng như sau:

$$F_r = M_r(q)\ddot{q} + N_r(q,\dot{q}) + G_r(q) \quad (2.21)$$

Phương trình động học nghịch:



Dựa theo hình l_1 = OQ, l_2 = QM, với qui ước $q_1 = q_1'' + q_1'$, khi biết trước tọa độ x_M, y_M, ta sẽ có:

$$q_1' = arctan\left(\frac{y_M}{x_M}\right); q_1'' = arccos\left(\frac{x_M^2+y_M^2+l_1^2-l_2^2}{2l_1\sqrt{x_M^2+y_M^2}}\right), \quad (2.22)$$

Từ đó ta suy ra tọa độ của điểm Q sẽ là:

$$x_Q = l_1 \cos q_1 ; y_Q = l_1 \sin q_1 \quad (2.23)$$

Và góc q_2 :

$$q_2 = arctan\left(\frac{y_Q - y_M}{x_Q - x_M}\right) \quad (2.24)$$

Như vậy khi xác định được phương trình của quỹ đạo đặt M, ta sẽ hoàn toàn xác định được các góc điều khiển các khớp robot tương ứng.

2.5. Đề xuất phần cứng và phần mềm cho đề tài

Bảng 2.1. Các đề xuất phần cứng

Thành phần	***Mục đích***	***Ghi chú***
Khung xe 4 bánh	Di chuyển tự hành	2 bánh trước chủ động, 2 bánh sau tự do
Động cơ DC (x2)	Điều khiển hướng di chuyển	Kèm Encoder phản hồi tốc độ
Bộ điều khiển động cơ DC BTS7960 (x2)	Điều khiển động cơ DC	Công suất lớn, giao tiếp với Arduino
Động cơ bước (x2)	Điều khiển cánh tay 2 bậc tự do	Tùy chọn dòng động cơ theo tải trọng
Driver TB6560	Điều khiển động cơ bước	
Arduino Mega	Bộ điều khiển trung tâm phần cơ điện	Nhiều cổng I/O, serial
Camera USB (x2)	Thị giác máy tính phát hiện dây thép	Độ phân giải đủ rõ
PC	Xử lý hình ảnh	Yêu cầu cài máy hợp lý
Encoder gắn động cơ DC	Phản hồi tốc độ thực tế	

Bảng 2.2. Các đề xuất phần mềm

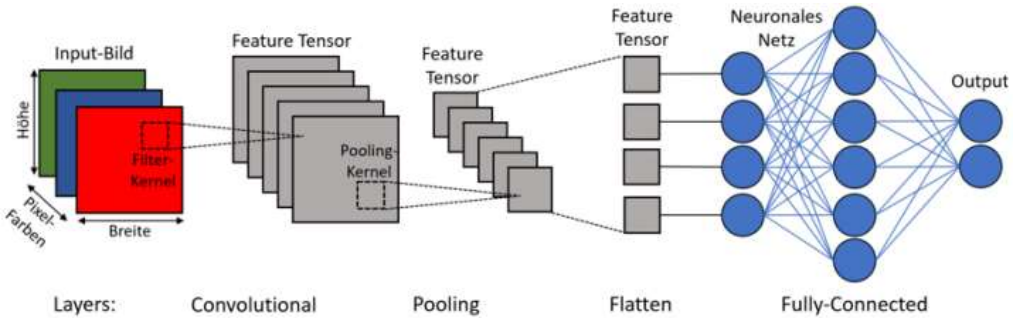
Thành phần	***Mục đích***	***Ghi chú***
Python (OpenCV, YOLOv8)	Xử lý hình ảnh từ 2 camera	Phát hiện dây thép, tính toán lệch
Thư viện PySerial	Giao tiếp PC/ Raspberry Pi với Arduino	Gửi dữ liệu alpha1, d1, alpha2, d2
Arduino IDE	Lập trình cho Arduino Mega	Viết code PID, điều khiển động cơ
YOLOv8	Mô hình AI phát hiện vật thể	Cần huấn luyện hoặc fine-tune lại

Chương 3: CƠ SỞ LÝ THUYẾT VÀ CÁC TÍNH TOÁN THIẾT KẾ

1.1. Thị giác máy tính

3.1.1. CNN – Cơ sở hình thành YoloV8-segmentation:

CNN là một loại mạng nơ-ron nhân tạo được thiết kế để xử lý dữ liệu hình ảnh, hoạt động thông qua các lớp tích chập (Convolutional layers) và lớp pooling nhằm trích xuất đặc trưng hiệu quả. CNN có ba thành phần chính: lớp tích chập (Convolution Layer), Lớp pooling (MaxPooling Layer), Lớp fully connected (FC Layer).



Sử dụng bài toán phân đoạn ảnh (Segmentation): trong đó mỗi pixel trong ảnh đầu vào được phân loại là thuộc về dây kim loại hoặc không. Mô hình sử dụng hàm kích hoạt sigmoid để xác định xác suất từng pixel thuộc vào dây kim loại.

3.1.1.1. Dữ liệu và tiền xử lý

- Thu thập ảnh:

* Chụp 500-1000 ảnh dây thép ở các góc độ, khoảng cách và điều kiện ánh sáng khác nhau

* Đảm bảo ảnh chứa đầy đủ các trường hợp: dây thẳng, cong, bị mất một phần

Hình 3.1. Thu thập dữ liệu ảnh dây thép với mặt đường

- Gắn nhãn trên Roboflow:

Sử dụng công cụ Polygon để đánh dấu chính xác đường viền dây thép

Định dạng nhãn: Mask segmentation theo chuẩn COCO

Hình 3.2. Ảnh đã được gắn nhãn nhờ công cụ Polygon của Roboflow

Tập dữ liệu: tập mẫu ảnh mặt đường có dây kim loại và tập ảnh mask tương ứng.

Tiền xử lý:

+ Chuyển đổi ảnh sang dạng RGB, chuẩn hóa về kích thước 128x128 pixels.

+ Chuẩn hóa giá trị pixel về khoảng [0,1].

+ Tách dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%).

3.1.1.2. Kiến trúc mô hình CNN

Bộ mã hóa (Encoder - Downsampling): Phần bộ mã hóa của mô hình CNN có nhiệm vụ trích xuất đặc trưng từ ảnh đầu vào thông qua các lớp tích chập và giảm kích thước dần bằng các lớp MaxPooling. Cấu trúc gồm 3 khối chính:

+ Một lớp tích chập (Conv2D) với số kênh tăng dần: 8 → 16 → 32.

+ Hàm kích hoạt ReLU giúp mô hình học được các đặc trưng phi tuyến.

+ Lớp MaxPooling2D với kernel (2×2) để giảm kích thước ảnh còn một nửa.

+ Sau khối cuối cùng, có thêm lớp Dropout (0.15) để giảm overfitting.

Các lớp này giúp mô hình tập trung vào các đặc trưng quan trọng của dây kim loại, đồng thời giảm kích thước ảnh để tối ưu tính toán.

Bộ giải mã: Sau khi trích xuất đặc trưng, bộ giải mã khôi phục kích thước ảnh về lại 128×128 pixels bằng cách sử dụng Conv2DTranspose (chuyển tích chập). Cấu trúc gồm 3 khối chính:

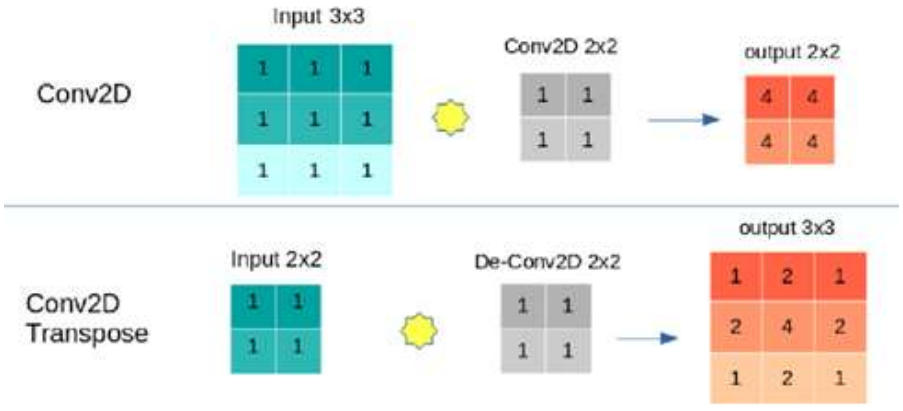
+ Một lớp chuyển tích chập (Conv2DTranspose) với số kênh giảm dần: 32 → 16 → 8.

+ Hàm kích hoạt ReLU để đảm bảo tính phi tuyến trong việc tái tạo ảnh.

+ Mỗi bước Upsampling làm tăng gấp đôi kích thước ảnh.

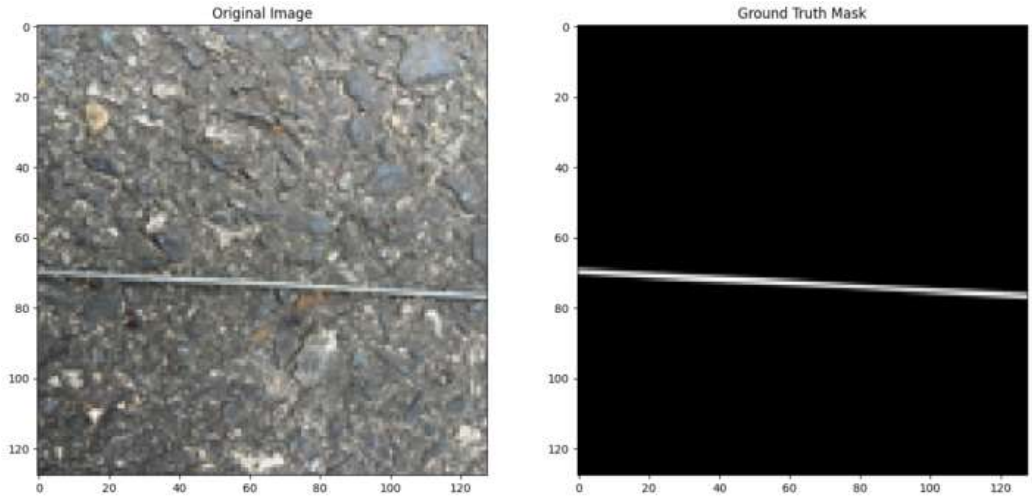
+ Cuối cùng, lớp đầu ra có: Lớp tích chập (Conv2D) với kernel (1×1) và hàm kích hoạt sigmoid, giúp mô hình phân loại từng pixel với giá trị từ 0 đến 1.

Phần này giúp tái tạo lại hình dạng của dây kim loại, từ đó tạo ra mặt nạ phân đoạn chính xác.

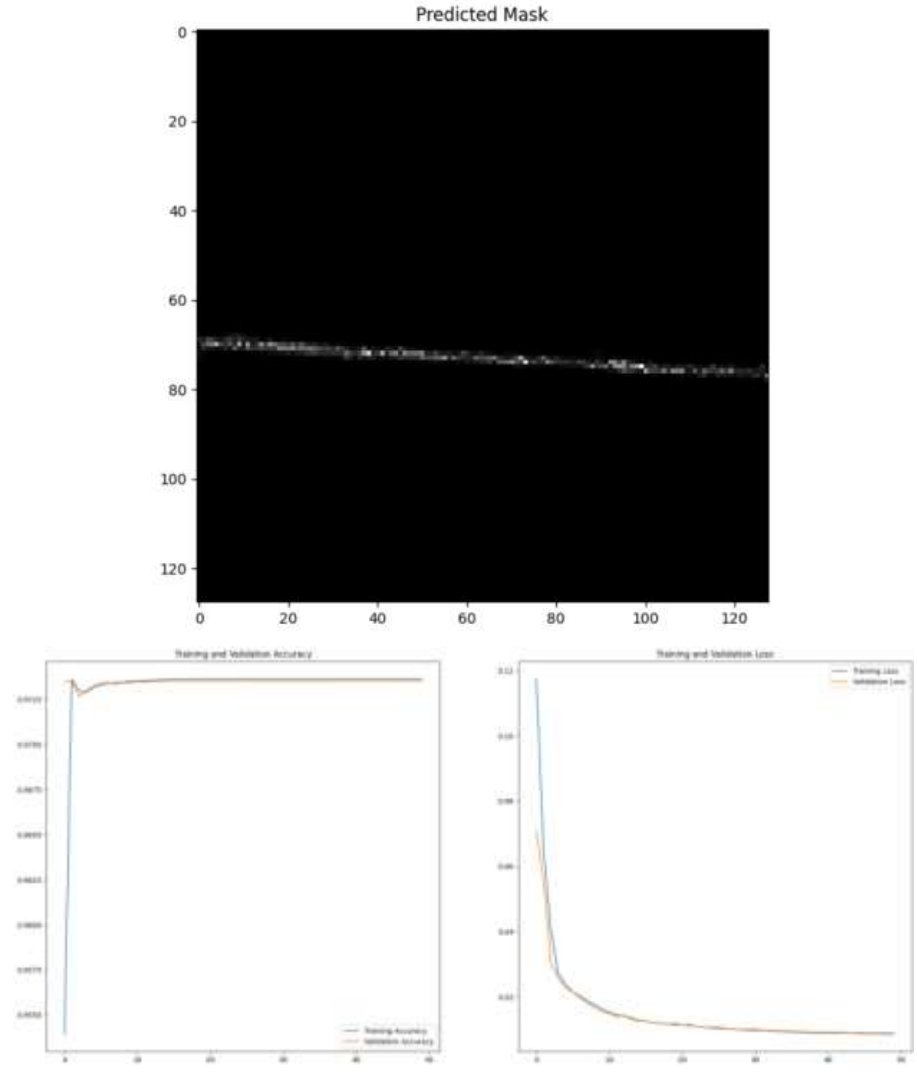


3.1.1.3. Kết quả:

Input:



Output:



Hình 3.3. Đồ thị độ chính xác tập huấn luyện so với tập kiểm tra

Nhìn vào đồ thị ta có thể thấy độ chính xác trên tập huấn luyện và tập kiểm tra gần như trùng nhau, với accuracy: 0.9733 - loss: 0.0127, điều này cho thấy mô hình hội tụ tốt, không có dấu hiệu về vấn đề underfitting hoặc overfitting.

3.1.2. Hiệu chỉnh camera 2D vật lý

3.1.2.1. Xác định tỷ lệ cm/pixel

Thiết lập:

Đặt thước đo chuẩn song song với mặt phẳng làm việc

Sử dụng thước đo độ để đảm bảo góc lệch giữa camera với mặt phẳng làm việc là 0 độ

Thu ảnh từ camera ở độ cao đã cố định trên xe

Sử dụng OpenCV của Python để tính toán được tỷ lệ cm/pixel và hiệu chỉnh sao cho đường thẳng đi qua tâm của 2 camera song song với than xe

Tính toán:

Đo khoảng cách L_{pixel} giữa 2 điểm đã có khoảng cách thực đo được

Tỷ lệ hiệu chỉnh:

$$S = \frac{L_{real}(cm)}{L_{pixel}} \text{ (cm/pixel)} \tag{3.1}$$

3.1.3. Tính toán góc lệch và khoảng cách

3.1.3.1. Phương pháp xác định phương trình đường thẳng

Hệ thống sử dụng phương pháp bình phương tối thiểu (Least Squares Fitting) thông qua hàm cv2.fitLine() của OpenCV để xác định phương trình đường thẳng đại diện cho dây thép. Phương pháp này tối ưu hóa sai số tổng bình phương khoảng cách từ tất cả các điểm biên (contour) đến đường thẳng cần tìm.

Công thức toán học:

Cho tập hợp điểm biên $\{(x_i, y_i)\}_{i=1}^{N}$ từ mask của dây thép

Tìm vector pháp tuyến $v = (v_x, v_y)$ và điểm (x_0, y_0) thuộc đường thẳng sao cho:

$$\min_{v,(x_0,y_0)} \sum_{i=1}^{N} \left(v_y (x_i - x_0) - v_x (y_i - y_0) \right)^2 \tag{3.2}$$

Phương trình tham số đường thẳng:

$$\begin{cases} x = x_0 + v_x.t \\ y = y_0 + v_y.t \end{cases} \tag{3.3}$$

Dạng tổng quát: $v_y(x-x_0)-v_x(y-y_0)=0$ (3.4)

3.1.3.2. Tính góc lệch giữa dây thép và trục ngang khung hình camera

Góc lệch α giữa dây thép và trục ngang (trục X) của camera được xác định như sau:

Xác định vector phương của dây thép

Từ phương trình đường thẳng thu được bằng cv2.fitLine():

Vector pháp tuyến: $v=\left(v_x,v_y\right)$

Vector phương của dây thép (song song với đường thẳng):

$u=\left(v_y,-v_x\right)$

Góc lệch α được tính bằng công thức:

$$\alpha_{radian}=\arctan\left(\frac{u_y}{u_x}\right)=\arctan\left(\frac{-v_x}{v_y}\right) \quad (3.5)$$

$$\alpha_{\deg}=\frac{180}{\pi}.\alpha \quad (3.6)$$

3.1.3.3. Tính khoảng cách lệch:

Tìm giao điểm giữa đường thẳng và trục đứng giữa khung hình $\left(x_c=\frac{\mathrm{W}}{2}\right)$:

$$y_{\mathrm{i}}=\frac{v_y}{v_x}\left(\frac{\mathrm{W}}{2}-x_0\right)+y_0 \quad (3.7)$$

Khoảng cách lệch từ tâm ảnh đến giao điểm:

$d_{pixel}=y_i-\frac{H}{2}$ (>0 nếu lệch phải, <0 nếu lệch trái) (3.8)

Chuyển đổi sang đơn vị cm:

$d_{cm}=d_{pixel}.S$ (S : tỷ lệ cm/pixcel) (3.9)

3.1.4. Truyền dữ liệu sang Arduino qua USB-to-TTL

3.1.4.1. Thiết lập giao tiếp UART

Giao thức và thông số kỹ thuật:

Giao thức: UART (Universal Asynchronous Receiver/Transmitter)

Cấu hình:

Baud rate: 115200 (tốc độ truyền phải khớp giữa PC và Arduino)

Data bits: 8 bit

Stop bits: 1 bit

Parity: None (không kiểm tra chẵn lẻ)

Flow control: None

3.1.4.2. Đóng gói dữ liệu

Cấu trúc gói tin:

Dữ liệu được đóng gói thành chuỗi ASCII có định dạng:

<góc_trái>,<khoảng_cách_trái>,<góc_phải>,<khoảng_cách_phải>\n

3.1.4.3. Truyền nhận dữ liệu:

Bước 1	Laptop (Python)	Arduino (C++)
1	Mở kết nối COMx (x tùy vào setup Port của Laptop)	Chờ dữ liệu
2	Gửi chuỗi dữ liệu có cấu trúc như trên	Nhận toàn bộ chuỗi
3	Chờ 100ms	Tách dữ liệu dựa vào bố trí dấu phẩy
4	Gửi gói tiếp theo	Cập nhật giá trị điều khiển

Lựa chọn 2 Camera USB:

Webcam Logitech C270 HD 720P:

Thông số kỹ thuật:

Hãng sản xuất	Logitech
Tên sản phẩm	Webcam HD C270
Độ phân giải	- Chụp ảnh ở chế độ thực 3 Megapixel; Phần mềm 15 Megapixel - Ghi lại hình ảnh ở chế độ Full HD 1080, Video Call ở chế độ HD 720
Góc quay (FOV)	78°
Điểm nhấn	- Tích hợp 1 Micro Stereo ứng dụng công nghệ RightSound cho âm thanh rõ ràng và trong veo. - Ứng dụng công nghệ Righ Light 2 - tự động điều chỉnh để lấy được ánh sáng tối ưu, và cho ảnh tốt ngay cả trong điều kiện ánh sáng mập mờ. - Ống kính Carl Zeiss . Tự động autofocus

Mạch chuyển USB to TLL CP2102:

Mạch chuyển USB to TTL CP2102 là module chuyển đổi tín hiệu USB sang tín hiệu Serial tuần tự theo chuẩn TTL. Mạch sử dụng chip CP2102

CP2102 là chip USB to UART của Silabs. CP2102 có kích thước nhỏ gọn và yêu cầu rất ít thành phần bên ngoài để hoạt động được ngay. CP2102 không sử dụng thạch anh ngoài như các chip PL2303.

Module có sẵn ngõ ra điện áp 3.3V với dòng tải tối đa là 100mA.

Trên mạch có 5 cổng đầu ra: 3.3V 5V Tx Rx Gnd

Thông số kỹ thuật:

Phạm vi nhiệt độ: -40Cto + 85C

Hỗ trợ windows vista / xp / server 2003/200, Mac OS-X / OS-9, Linux

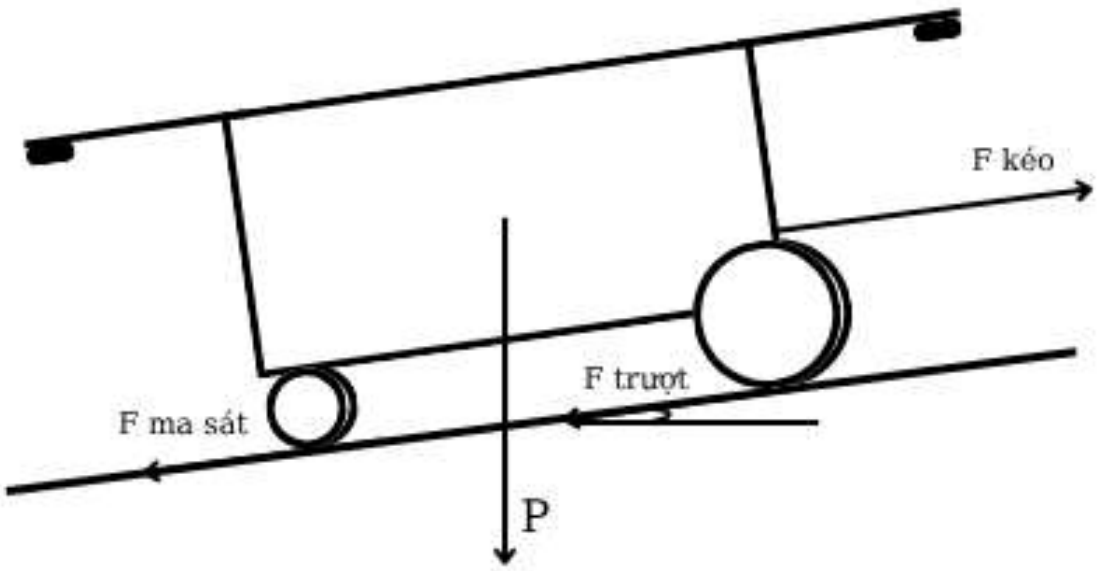
Kích thước: chiều dài (Không bao gồm USB): 30 mm

USB để lấy nguồn, dẫn đến giao diện bao gồm 3,3V (<40mA), 5V, GND, TX, RX, mức pin tín hiệu là 3,3V, logic dương

Hỗ trợ tốc độ truyền trong khoảng 300bps ~ 1Mbps

3.2. Phân tích mô hình động học của xe

3.2.1. Phương trình động lực học



Hình 3.4. Mô hình xe

- Xe trong quá trình di chuyển được mô hình hóa như một hệ thống động học bậc hai, mô hình của xe khi chuyển động tịnh tiến:

$$m \times a = F_k - \left(F_{ms} + F_{truot}\right) \tag{3.10}$$

Trong đó:

m: Khối lượng xe (kg)

a: Gia tốc của xe (m/s^2)

F_k : Lực kéo do động cơ tạo ra tại 2 bánh trước

F_{ms} : Lực ma sát bánh xe

F_{truot} : Lực trượt trên mặt dốc khi di chuyển trên mặt phẳng nghiêng góc α

3.2.2. Phân tích các lực cản chính

3.2.2.1. Lực ma sát lăn (F_{ms})

- Lực ma sát giữa bánh xe và mặt đường gây ra:

$$F_{ms} = \mu mg \tag{3.11}$$

Trong đó: μ: Hệ số ma sát của bánh tự do.

m: Khối lượng xe (kg).

g: Gia tốc trọng trường (9.81 m/s^2).

3.2.2.2. Lực trượt trên mặt dốc:

- Lực trượt trên mặt dốc ($F_{trượt}$): Nếu di chuyển trên mặt phẳng nghiêng với góc nghiêng α

$$F_{truot} = mg\sin\alpha \tag{3.12}$$

3.2.3. Tổng lực cản và công suất yêu cầu:

3.2.3.1. Tổng lực cản.

- Lực ma sát giữa bánh xe và mặt đường. Ta có:

  - $\mu = 0.018$: Hệ số ma sát của bánh xe với mặt đường. (giả định)
  - $m = 30\,\text{kg}$: Khối lượng toàn bộ xe.
  - $g = 9.81\,\text{m s}^{-2}$: Gia tốc trọng trường.

$$F_{ms} = \mu mg = 0.018 \times 30 \times 9.81 = 5.2974N$$

- Lực trượt trên mặt dốc nếu di chuyển trên mặt phẳng nghiêng (giả định dốc 5%)(góc $\approx \arctan(0.05) \approx 2.8624^\circ$, $\sin(\alpha) \approx 0.05$):

$$F_{truot} = mg\sin\alpha = 30 \times 9.81 \times 0.05 = 14.715N$$

- Tổng lực cản: $F_t = F_{ms} + F_{truot} = 5.2974 + 14.715 = 20.0124N$

3.2.3.2. Công suất yêu cầu:

- Công suất yêu cầu của xe được tính:

$$P = F_t \times v \tag{3.13}$$

- Với tốc độ v = 5 m/s

$$\Rightarrow P = F_t \times v = 20.0124 \times 5 = 100.06W$$

3.2.3.3. Lựa chọn phần cứng:

- Lựa chọn động cơ một chiều có công suất định mức 250W trở lên, điện áp hoạt động 24V, đảm bảo đáp ứng đủ mô-men và công suất trong mọi tải và điều kiện vận hành. Và nhóm chọn động cơ MY1016Z.

- Lựa chọn Driver BTS7960 để điều khiển công suất, điều khiển dòng điện cấp cho động cơ DC dựa trên tín hiệu PWM từ Arduino

- *Động cơ DC MY1016Z*

Thông số	**Giá trị**
Model	MY1016Z
Điện áp hoạt động	24 VDC
Công suất danh định	350 W
Dòng điện định mức	18 A
Tốc độ quay định mức	330 vòng/phút (RPM)
Moment định mức	110 kg·cm ≈ 10.78 Nm
Moment tối đa (Stall Torque)	350 kg·cm ≈ 34.3 Nm
Hộp số giảm tốc tích hợp	Có
Hệ số giảm tốc (gear ratio)	9.78 : 1

Đánh giá:

- Mô-men: Cung cấp 10.78Nm, đáp ứng được yêu cầu 0.105Nm (dốc 5%)

- Công suất: 350W đáp ứng được yêu cầu 77.0331W, đảm bảo có thể dự phòng.

- *Driver BTS7960*

Tiêu chí	**Yêu cầu hệ thống**	**BTS7960**	**Đánh giá**
Dòng liên tục	≥ 15A	43A	Đáp ứng
Dòng đỉnh	≥ 30A	> 43A (ngắn hạn)	Dư tải
Điện áp hoạt động	24V	24V	Phù hợp
Logic điều khiển	PWM 5V	3.3–5V	Tương thích với Arduino Mega
Bảo vệ mạch	Cần có	Có	An toàn

Phân tích đánh giá:

Động cơ MY1016Z: 18 A, BTS7960 (43 A) đáp ứng.

- Dòng khởi động cần ~30 A, BTS7960 (>43 A) đủ.

3.3. Triển khai điều khiển

3.3.1. Điều khiển điều hướng xe:

- Công thức tính tốc độ góc mong muốn: (đơn vị rad/s)

$$\omega = K_d \times d \tag{3.14}$$

- Trong đó:

+ Kd: Hệ số điều chỉnh khoảng cách (d)

+ d sẽ được lấy theo trường hợp 1 hoặc 2

- Tính tốc độ từng bánh xe: (đơn vị km/h)

$$v_{left} = v_{base} - \frac{\omega \times L}{2} \tag{3.15}$$

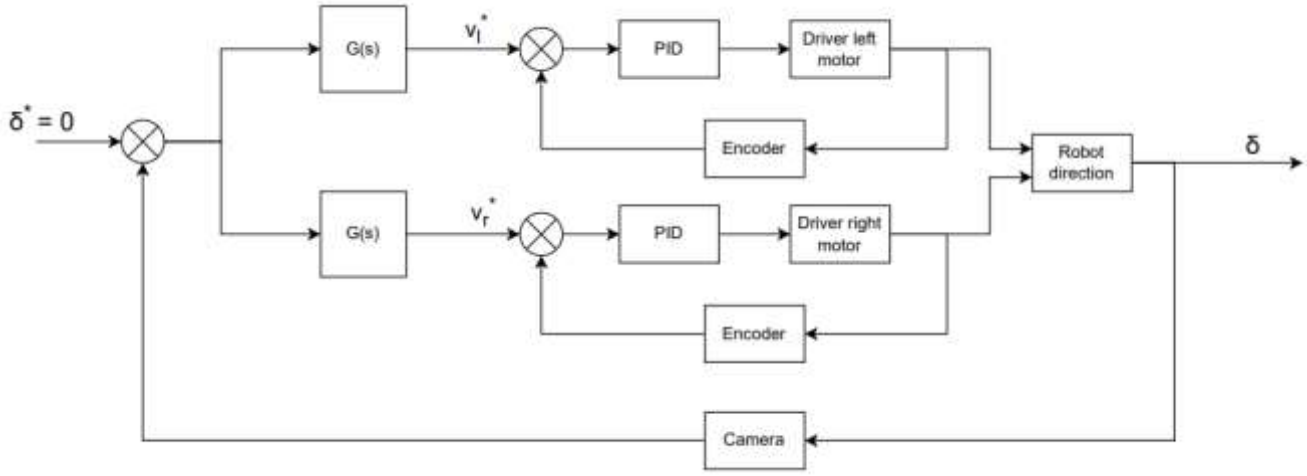
$$v_{right} = v_{base} + \frac{\omega \times L}{2} \tag{3.16}$$

- Tính tốc độ RPM cấp cho động cơ:

$$RPM_{left} = \frac{\frac{v_{left} \times 1000}{60}}{d \times 3.14} \tag{3.17}$$

$$RPM_{right} = \frac{\frac{v_{right} \times 1000}{60}}{d \times 3.14} \tag{3.18}$$

- Sơ đồ điều khiển vòng kín:



3.3.2. Điều khiển cánh tay Robot:

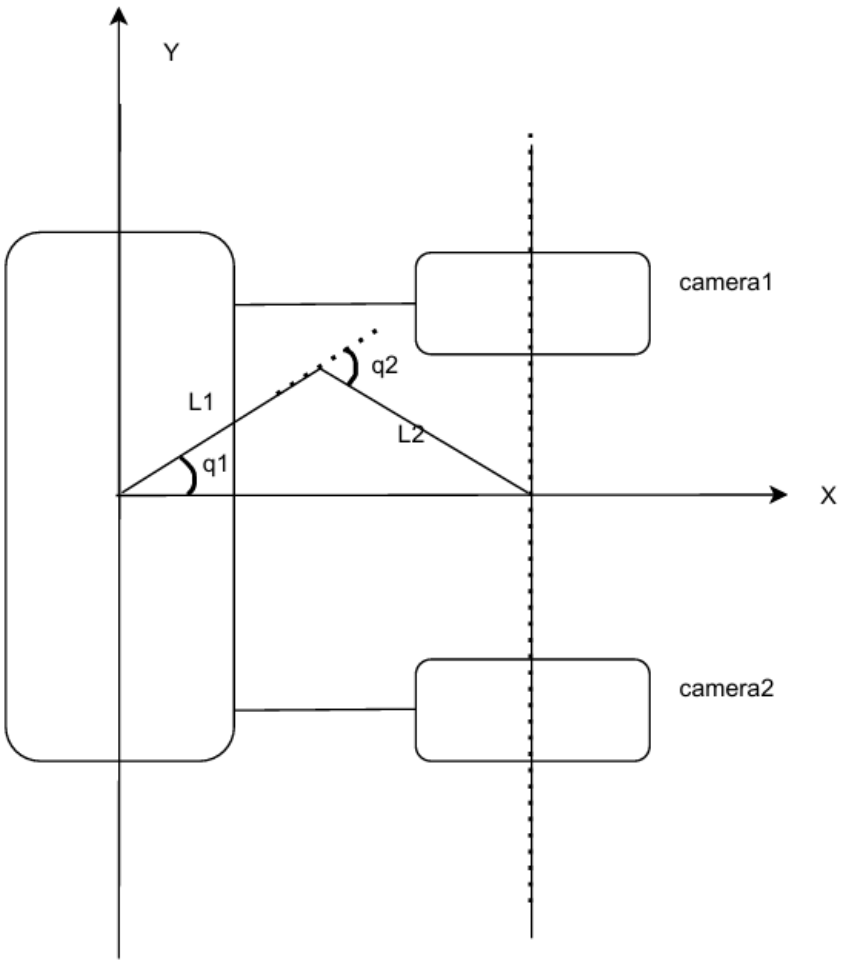
3.3.2.1. Thuật toán chính cho cánh tay Robot 2DOF:

- **Chia tổ hợp các trường hợp theo 2 đại lượng:**

Hướng: Trái, phải , đúng

Vị trí trọng tâm: Lệch trái, lệch phải, đúng

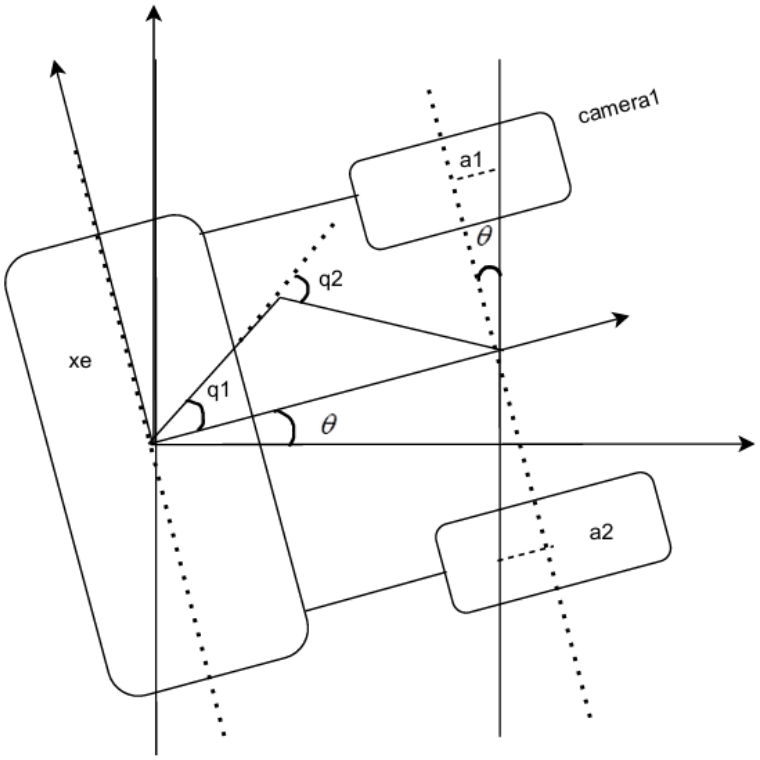
- **Trường hợp 1:** Vị trí trọng tâm (đúng) , hướng (đúng).



Từ vị trí y_d =0 và x_d =R , với R là chiều dài từ tâm xe đến trung điểm khoảng cách từ tâm vùng quan sát camera 1 đến tâm vùng quan sát camera 2.

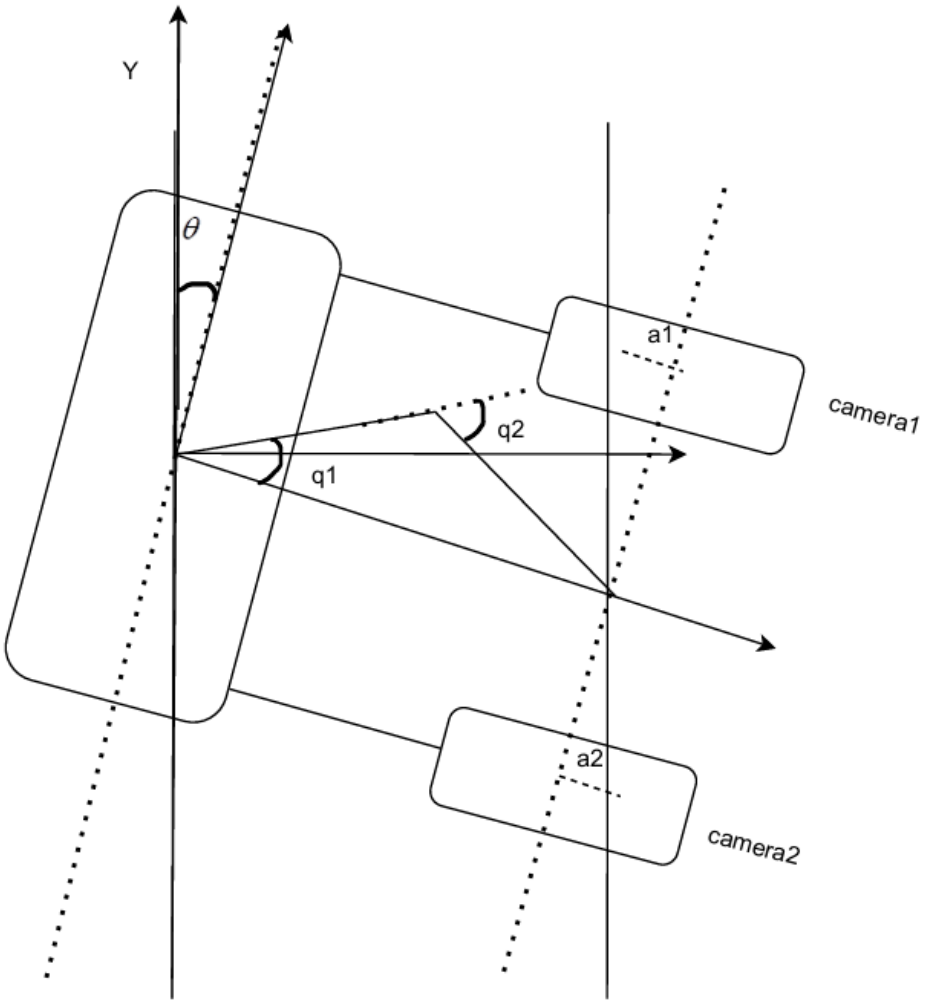
- > Từ vị trí điểm cuối end-effector mà ta muốn cánh tay Robot chạm đến d(x_d , y_d) , ta sử dụng Inverse Kinematics tính ra được góc q_1 , q_2 cần quay để robot có thể đạt đến vị trí mong muốn .

- **Trường hợp 2:** Vị trí trọng tâm (đúng) , hướng (trái) .



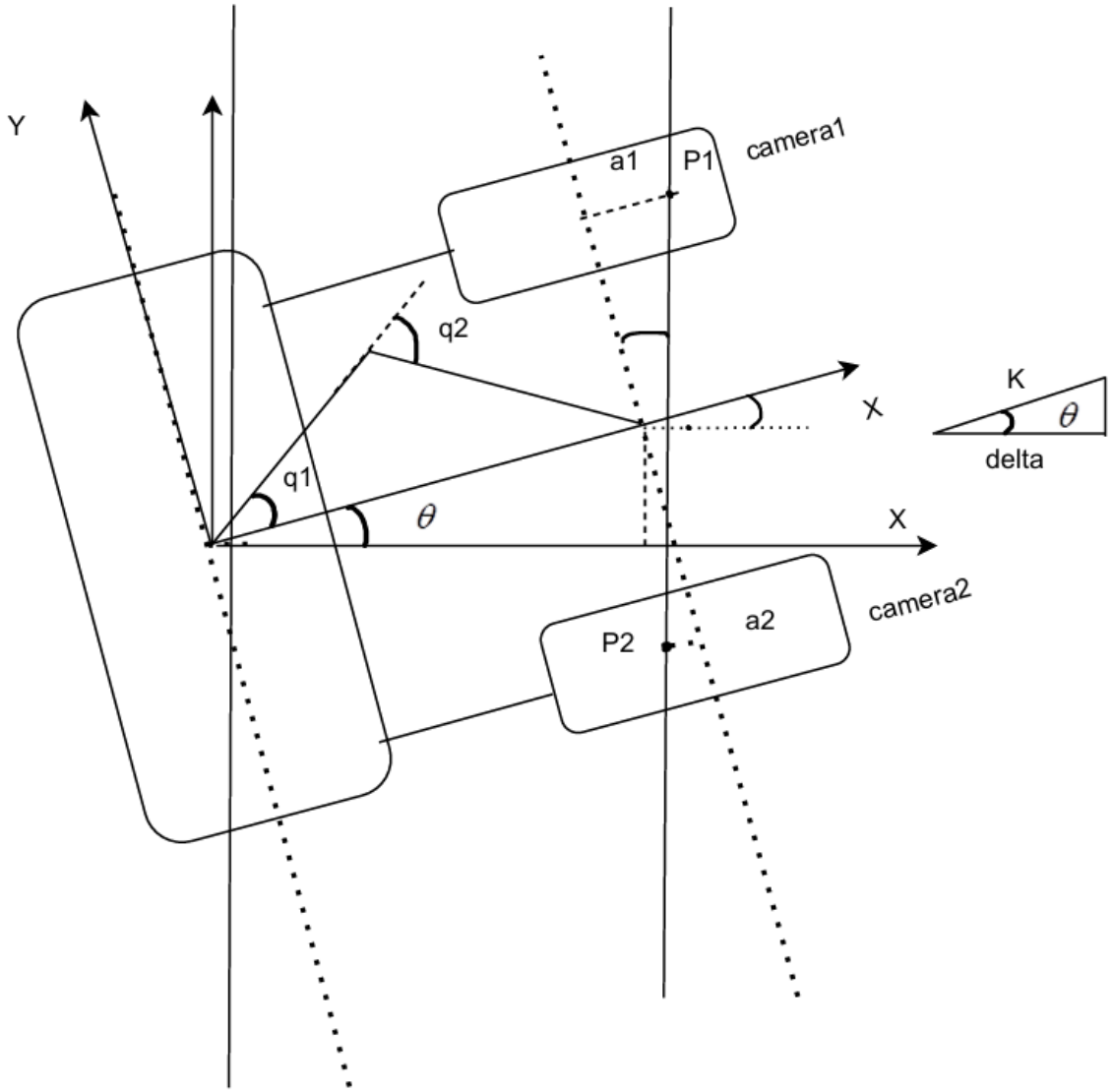
Ở trường hợp này, khi xe bị nghiêng sang bên trái, hệ tọa độ xe được gắn tại tâm xe bị xoay đi 1 góc θ, trong khi đó hệ tọa độ robot trùng với hệ tọa độ xe khi xoay . Như vậy, vị trí điểm cuối của robot đã bị di chuyển đi, để cho điểm cuối vẫn nằm ở vị trí ban đầu , ta thực hiện xoay hệ tọa độ q_1 một góc θ, tức là q_1 sau điều chỉnh = q_1 - θ .

Trường hợp 3: Hướng (phải), vị trí (đúng).



Trường hợp này tương tự như trường hợp 2 nhưng ngược lại, vì hệ toạ độ cánh tay robot gắn với trọng tâm xe đã bị quay đi 1 góc bằng q_1 - θ , do đó để q_1 không đổi ta tính lại q_1 sau điều chỉnh = q_1 + θ

Trường hợp 4: Vị trí trọng tâm (lệch trái) , hướng (Trái).



Trong trường hợp này, ngoài việc xe bị nghiêng sang bên trái, nó còn bị lệch hẳn về bên trái. Do đó, ta sẽ sử dụng độ lệch của tâm xe làm chuẩn để xác định mức độ lệch của xe so với trục quy chiếu ban đầu, so với trường hợp 2 thì xe chỉ bị xoay đi 1 góc bằng θ nên khi ta thực hiện việc xoay hệ toạ độ của robot ngược lại chính bằng θ thì điểm cuối của Robot sẽ chạm đến vị trí cần chạm. Nhưng ở trường hợp này, xe còn bị lệch đi 1 đoạn bằng delta, do đó điểm cuối của robot vốn dĩ cũng đã bị lệch đi 1 đoạn chính bằng đoạn xe lệch.Vậy ta có thể giải quyết bài toán trên như sau:

Ta đặt tâm của xe (cũng chính là gốc toạ độ của cánh tay Robot) là gốc toạ độ 0,dùng suy luận hình học 2D thì ta có thể xác định được độ lệch trọng tâm xe dựa vào vùng nhìn của 2 Camera quan sát 2 đầu của dây thép,đoạn lệch đó là delta.

Từ hệ toạ độ gốc đặt tại tâm xe, ta xác định được hệ toạ độ của vùng quan sát camera:

$$C1:(R,\frac{L}{2}) => P_1:(R+a_1,\frac{L}{2}) \quad (3.19)$$

$$C2:(R,\frac{-L}{2}) => P_2:(R-a_2,\frac{-L}{2}) \quad (3.20)$$

Từ 2 điểm P_1 và P_2 thì ta sẽ xác định được hệ số góc m:

$$m=\frac{y-y_0}{x-x_0}=\frac{\frac{-L}{2}-\frac{L}{2}}{R-a_2-R-a_1}=\frac{L}{a_1+a_2} \quad (3.21)$$

Ta có phương trình đường thẳng :

$$y-y_0=m(x-x_0) \quad (3.22)$$

Vì phương trình đường thẳng đi qua điểm $P_1:(R+a_1,\frac{L}{2})$, thay vào phương trình

$$=> y-\frac{L}{2}=(\frac{L}{a_1-a_2})\times(x-R-a_1) \quad (3.23)$$

Để tìm giao điểm của đường thẳng này với trục x, ta tiếp tục thay y=0 vào phương trình

$$=> \frac{-L}{2}=(\frac{L}{a_1+a_2})\times(x-R-a_1) \quad (3.24)$$

Rút ra : $x=R+\frac{a_1-a_2}{2}$ (*) (3.25)

Mà: $k=|x-R| => delta=k\times\cos(\theta)$ (3.26)

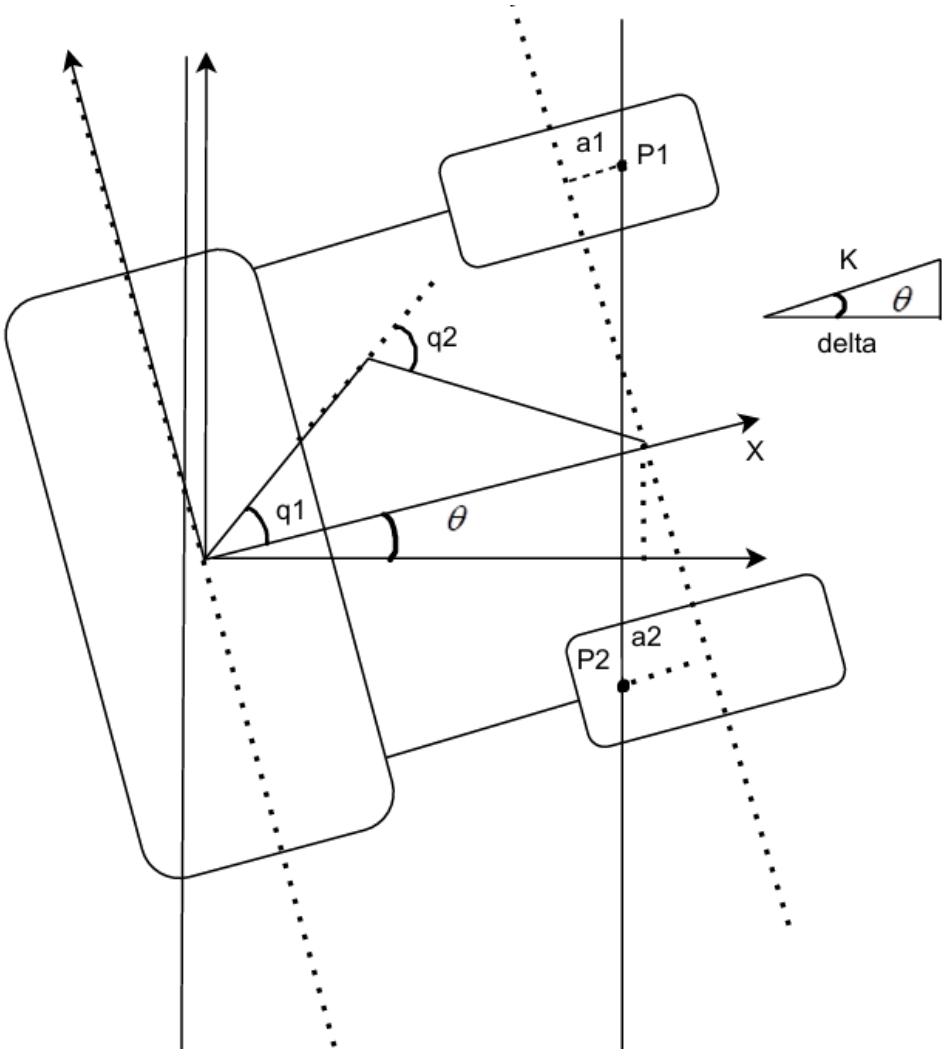
Vì trọng tâm xe bị lệch sang bên trái, do đó ta cần cánh tay di chuyển thêm 1 đoạn bằng delta về bên phải.

$$=> x_d=L+delta \ , \ y_d=0 \quad (3.27)$$

Dùng động học nghịch tính ra q_1 và q_2 sau đó xoay hệ toạ độ của Robot lại vị trí ban đầu khi chưa lệch hướng bằng cách tính lại $q_1 = q_1 - \theta$.

Với R là khoảng cách từ tâm xe đến tâm khoảng cách từ vùng quan sát camera 1 và camera 2.

Trường hợp 5: Vị trí trọng tâm (lệch phải), hướng (trái).



Trường hợp này cách tính tương tự như trường hợp 4 , nhưng khác ở chỗ vì trọng tâm xe bị lệch sang bên phải, do đó ta cần cánh tay di chuyển bớt 1 đoạn bằng delta về bên trái.

Khi xe lệch trọng tâm sang bên phải $a_1 < a_2$, do đó phương trình (*) giá trị x tính ra nhỏ hơn x ở trường hợp 4, tức $x - R$ là 1 số âm $\Rightarrow$ $delta = k \times \cos(\theta)$ mang giá trị âm biểu diễn cho việc điểm cuối cần xê dịch lại so với điểm ban đầu để bù lệch trọng tâm . Ta đặt tâm của xe (cũng chính là gốc toạ độ của cánh tay Robot) là gốc toạ độ 0,dùng suy luận hình học 2D thì ta có thể xác định được độ lệch trọng tâm xe dựa vào vùng nhìn của 2 Camera quan sát 2 đầu của dây thép,đoạn lệch đó là delta.

Từ hệ toạ độ gốc đặt tại tâm xe, ta xác định được hệ toạ độ của vùng quan sát camera:

$$C1 : (R, \frac{L}{2}) => P_1 : (R + a_1, \frac{L}{2}) \quad (3.28)$$

$$C2 : (R, \frac{-L}{2}) => P_2 : (R - a_2, \frac{-L}{2}) \quad (3.29)$$

Từ 2 điểm P_1 và P_2 thì ta sẽ xác định được hệ số góc m:

$$m = \frac{y - y_0}{x - x_0} = \frac{\frac{-L}{2} - \frac{L}{2}}{R - a_2 - R - a_1} = \frac{L}{a_1 + a_2} \quad (3.30)$$

Ta có phương trình đường thẳng :

$$y - y_0 = m(x - x_0) \quad (3.31)$$

Vì phương trình đường thẳng đi qua điểm $P_1 : (R + a_1, \frac{L}{2})$, thay vào phương trình

$$=> y - \frac{L}{2} = (\frac{L}{a_1 - a_2}) \times (x - R - a_1) \quad (3.32)$$

Để tìm giao điểm của đường thẳng này với trục x, ta tiếp tục thay y=0 vào phương trình

$$=> \frac{-L}{2} = (\frac{L}{a_1 + a_2}) \times (x - R - a_1) \quad (3.33)$$

Rút ra : $x = R + \frac{a_1 - a_2}{2}$ (3.34)

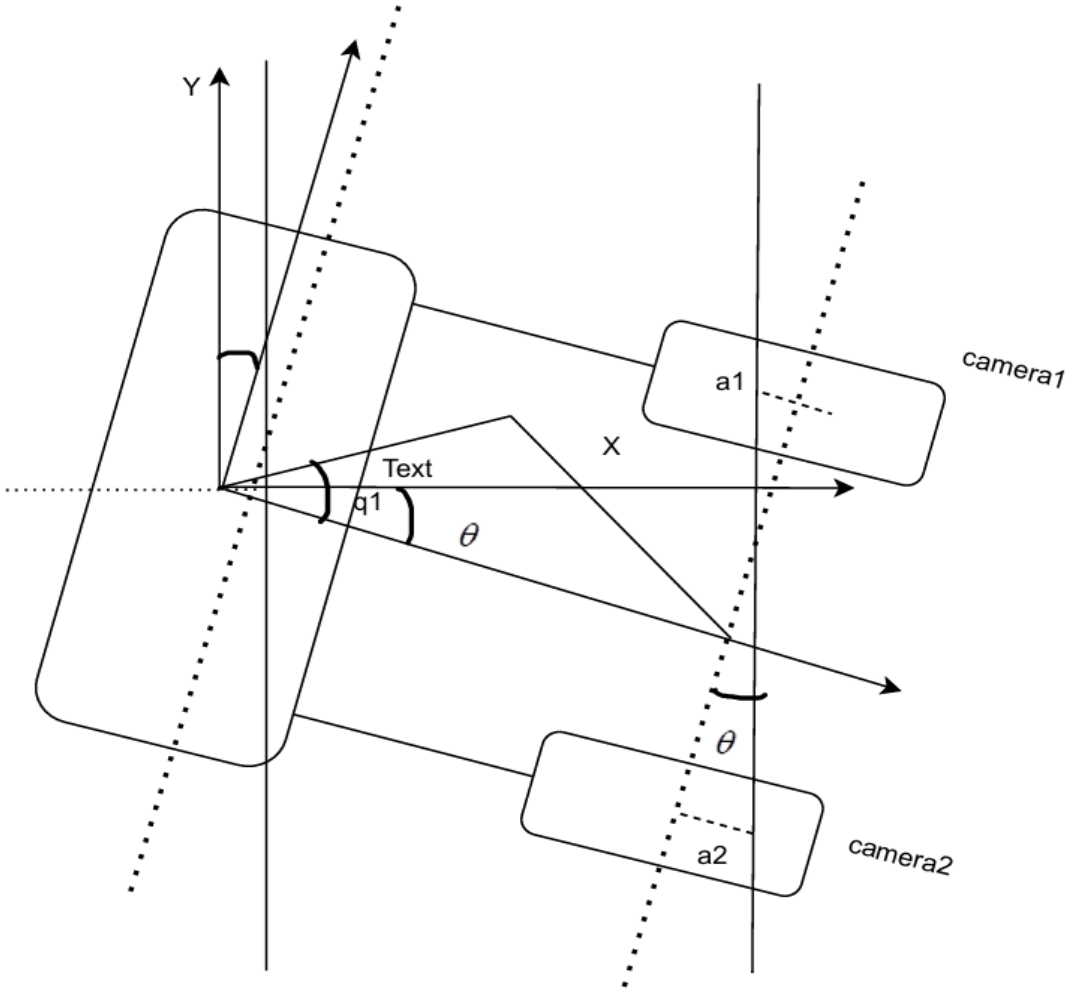
Mà: $k = x - R => delta = k \times \cos(\theta)$ (3.35)

Vì trọng tâm xe bị lệch sang bên trái, do đó ta cần cánh tay di chuyển thêm 1 đoạn bằng delta về bên phải.

$$=> x_d = L - delta \ , \ y_d = 0 \quad (3.36)$$

Dùng động học nghịch tính ra q_1 và q_2 sau đó xoay hệ toạ độ của Robot lại vị trí ban đầu khi chưa lệch hướng bằng cách tính lại $q_1 = q_1 - \theta$.

Trường hợp 6 : Vị trí trọng tâm (Lệch trái) , hướng (phải).



Từ hệ toạ độ gốc đặt tại tâm xe, ta xác định được hệ toạ độ của vùng quan sát camera:

$$C1:(R,\frac{L}{2}) => P_1:(R-a_1,\frac{L}{2}) \quad (3.37)$$

$$C2:(R,\frac{-L}{2}) => P_2:(R+a_2,\frac{-L}{2}) \quad (3.38)$$

Từ 2 điểm P_1 và P_2 thì ta sẽ xác định được hệ số góc m:

$$m=\frac{y-y_0}{x-x_0}=\frac{\frac{-L}{2}-\frac{L}{2}}{R+a_2-R-a_1}=\frac{-L}{a_1+a_2} \quad (3.39)$$

Ta có phương trình đường thẳng :

$$y-y_0=m(x-x_0) \quad (3.40)$$

Vì phương trình đường thẳng đi qua điểm $P_1:(R-a_1,\frac{L}{2})$, thay vào phương trình

$$=> y-\frac{L}{2}=(\frac{L}{a_1-a_2})\times(x-R-a_1) \quad (3.41)$$

Để tìm giao điểm của đường thẳng này với trục x, ta tiếp tục thay y=0 vào phương trình

$$=> \frac{-L}{2} = (\frac{-L}{a_1 + a_2}) \times (x - R + a_1) \quad (3.42)$$

Rút ra : $x = R + \frac{-a_1 + a_2}{2}$ (3.43)

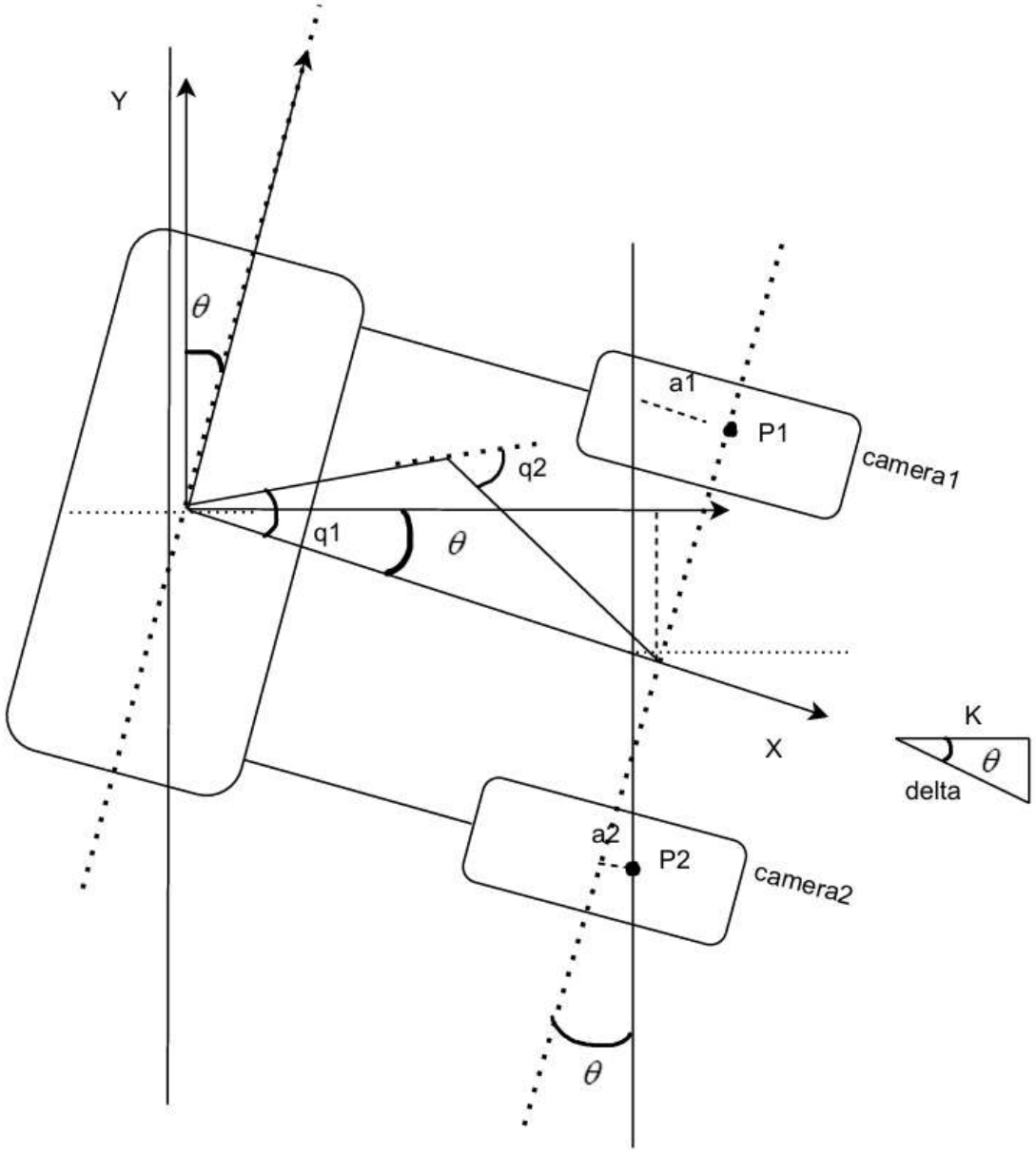
Mà: $k = |x_d - R| => delta = k \times \cos(\theta)$ (3.44)

Vì trọng tâm xe bị lệch sang bên trái nên ta cần cánh tay di chuyển thêm 1 đoạn bằng delta về bên phải.

$$=> x_d = L + delta \ , \ y_d = 0 \quad (3.45)$$

Dùng động học nghịch tính ra q_1 và q_2 sau đó xoay hệ toạ độ của Robot lại vị trí ban đầu khi chưa lệch hướng bằng cách tính lại $q_1 = q_1 + \theta$.

Trường hợp 7: Vị trí trọng tâm (Lệch phải) , hướng (phải).



Từ hệ toạ độ gốc đặt tại tâm xe, ta xác định được hệ toạ độ của vùng quan sát camera:

$$C1:(R,\frac{L}{2}) => P_1:(R-a_1,\frac{L}{2}) \quad (3.46)$$

$$C2:(R,\frac{-L}{2}) => P_2:(R+a_2,\frac{-L}{2}) \quad (3.47)$$

Từ 2 điểm P_1 và P_2 thì ta sẽ xác định được hệ số góc m:

$$m=\frac{y-y_0}{x-x_0}=\frac{\frac{-L}{2}-\frac{L}{2}}{R+a_2-R-a_1}=\frac{-L}{a_1+a_2} \quad (3.48)$$

Ta có phương trình đường thẳng :

$$y-y_0=m(x-x_0) \quad (3.49)$$

Vì phương trình đường thẳng đi qua điểm $P_1:(R-a_1,\frac{L}{2})$, thay vào phương trình

$$=> y-\frac{L}{2}=(\frac{L}{a_1-a_2})\times(x-R-a_1) \quad (3.50)$$

Để tìm giao điểm của đường thẳng này với trục x, ta tiếp tục thay y=0 vào phương trình

$$=> \frac{-L}{2}=(\frac{-L}{a_1+a_2})\times(x-R+a_1) \quad (3.51)$$

Rút ra : $x=R+\frac{-a_1+a_2}{2}$ (3.52)

Mà: $k=|x_d-R| => delta=k\times\cos(\theta)$ (3.53)

Vì trọng tâm xe bị lệch sang bên trái nên ta cần cánh tay di chuyển thêm 1 đoạn bằng delta về bên phải.

$$=> x_d=L-delta \ , \ y_d=0 \quad (3.54)$$

Dùng động học nghịch tính ra q_1 và q_2 sau đó xoay hệ toạ độ của Robot lại vị trí ban đầu khi chưa lệch hướng bằng cách tính lại $q_1 = q_1+\theta$.

3.3.2.2. Tính toán cho cánh tay 2DOF:

Động năng thanh 1 (K_1):

$$K_1=\frac{1}{2}I_1\dot{q}_1^{\,2}=\frac{1}{6}m_1L_1^{\,2}\dot{q}_1^{\,2} \quad (3.55)$$

I_1 :Mô men quán tính của thanh 1 quanh trục quay (tại khớp 1).

$\dot{q}_1$:Vận tốc góc của thanh 1.

Công thức trên có thể diễn giải ra là:

$$I_1=\frac{1}{3}m_1L_1^{\,2} \quad (3.56)$$

Điều này phù hợp với momen quán tính của một thanh đồng nhất quay quanh đầu khớp 1

$$I = \frac{1}{3} m_1 L^2 \tag{3.57}$$

Thay số vào ta được : $k_1 = \frac{1}{2}.\frac{2}{3}.q_1^{\cdot 2} = \frac{1}{6}.2.1^2.q_1^{\cdot 2} = \frac{1}{3}\dot{q}_1$ (3.58)

Động năng của thanh 2 (K2)

Động năng của thanh 2 phức tạp hơn vì nó bao gồm cả chuyển động tịnh tiến của tâm khối và chuyển động quay quanh tâm khối. Giả sử thanh 2 là thanh đồng nhất, tâm khối nằm ở giữa thanh (cách khớp 2 một khoảng L2/2).

Tọa độ và vận tốc tâm khối của thanh 2

$$x_2 = L_1 \cos q_1 + \frac{1}{2} L_2 \cos q_2 \tag{3.59}$$

$$y_2 = L_1 \sin q_1 + \frac{1}{2} L_2 \sin q_2 \tag{3.60}$$

Sau đó lấy đạo hàm theo thời gian:

$$\dot{x_2} = -L_1 \dot{q_1} \sin q_1 - \frac{1}{2} L_2 q_2^{\cdot 2} \sin q_2 \tag{3.61}$$

$$\dot{y_2} = L_1 \dot{q_1} \cos q_1 + \frac{1}{2} L_1 \dot{q_2} \sin q_2 \tag{3.62}$$

Lấy bình phương vận tốc tâm khối:

$$\dot{x_2}^2 = (-L_1 \dot{q_1} \sin q_1 - \frac{1}{2} L_2 \dot{q_2} \sin q_2)^2 = -L_1^2 \dot{q_1}^2 \sin^2 q_1 + \frac{1}{4} L_2^2 \dot{q_2}^2 \sin^2 q_2 + L_1 \dot{q_1}.\frac{1}{2} L_2 \dot{q_2} \sin q_2 \sin q_1 \tag{3.63}$$

$$\dot{y_2}^2 = (L_1 \dot{q_1} \cos q_1 + \frac{1}{2} L_2 \dot{q_2} \cos q_2)^2 = L_1^2 \dot{q_1}^2 \cos^2 q_1 + \frac{1}{4} L_2^2 \dot{q_2}^2 \cos^2 q_2 + L_1 \dot{q_1} \frac{1}{2} L_2 \dot{q_2} \cos q_2 \tag{3.64}$$

Sau đó lấy tổng vận tốc bình phương:

$$\begin{aligned} \dot{x_2}^2 + \dot{y_2}^2 &= -L_1^2 \dot{q_1}^2 \sin^2 q_1 + \frac{1}{4} L_2^2 \dot{q_2}^2 \sin^2 q_2 + L_1 \dot{q_1}.\frac{1}{2} L_2 \dot{q_2} \sin q_2 \sin q_1 \\ &+ L_1^2 \dot{q_1}^2 \cos^2 q_1 + \frac{1}{4} L_2^2 \dot{q_2}^2 \cos^2 q_2 + L_1 \dot{q_1} \frac{1}{2} L_2 \dot{q_2} \cos q_2 \end{aligned} \tag{3.65}$$

Sử dụng $\sin^2\theta + \cos^2\theta = 1$ và $\sin q_1 \sin q_2 + \cos q_1 \cos q_2 = \cos(q_1 - q_2)$:

$$\dot{x_2}^2 + \dot{y_2}^2 = L_1^2 \dot{q_1}^2 + \frac{1}{4} L_2^2 \dot{q_2}^2 + L_1 L_2 \dot{q_1} \dot{q_2} \cos(q_1 - q_2) \tag{3.66}$$

Tổng động năng của thanh 2:

$$K_2 = K_{2,tt} + K_{2,quay} = \frac{1}{2} m_2 L_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 . \frac{1}{4} L_2^2 \dot{q}_2^2 + \frac{1}{2} m_2 L_1 L_2 \dot{q}_1 \dot{q}_2 \cos(q_1 - q_2) + \frac{1}{24} m_2 L_2^2 \dot{q}_2^2$$

(3.71)

Hệ số $\dot{q}_2^2$

$$\frac{1}{2}m_2.\frac{1}{4}L_2^2+\frac{1}{24}m_2L_2^2=\frac{1}{6}m_2L_2^2 \quad (3.67)$$

Vậy:

$$K_2=\frac{1}{2}m_2L_1^2\dot{q}_1^2+\frac{1}{6}m_2L_2^2\dot{q}_2^2+\frac{1}{2}m_2L_1L_2\dot{q}_1\dot{q}_2\cos(q_1-q_2) \quad (3.68)$$

Tổng động năng của hệ:

$$K=K_1+K_2$$

$$=\frac{1}{6}m_1L_1^2\dot{q}_1^2+\frac{1}{2}m_2L_1^2\dot{q}_1^2+\frac{1}{6}m_2L_2^2\dot{q}_2^2+\frac{1}{2}m_2L_1L_2\dot{q}_1\dot{q}_2\cos(q_1-q_2) \quad (3.69)$$

Hệ số của $\dot{q}_1^2$:

$$\frac{1}{6}m_1L_1^2+\frac{1}{2}m_2L_1^2=L_1^2\left(\frac{1}{6}m_1+\frac{1}{2}m_2\right)$$

Vậy :

$$K=L_1^2\left(\frac{1}{6}m_1+\frac{1}{2}m_2\right)\dot{q}_1^2+\frac{1}{6}m_2L_2^2\dot{q}_2^2+\frac{1}{2}m_2L_1L_2\dot{q}_1\dot{q}_2\cos(q_1-q_2) \quad (3.70)$$

Phương trình động lực:

$$\tau=M(q)\ddot{q}+C(q,\dot{q})\dot{q} \quad (3.71)$$

Trong đó $C(q,\dot{q})$ chứa các lực Corioolis và ly tâm, nhưng để nhưng để chọn động cơ, ta tập trung vào mô men quán tính khi gia tốc tối đa, giả sử vận tốc ban đầu nhỏ (C~0).

Tính mô men tối đa để chọn động cơ:

Giả sử yêu cầu gia tốc góc tối đa cho mỗi khớp là $\alpha=10rad/s^2$.

Mô men tại khớp 1 (τ_1):

Khi $\ddot{q}_1=\alpha,\ddot{q}_2=0$ (chuyển động khớp 1, khớp 2 giữ cố định):

$$\tau_1=M_{11}\ddot{q}_1=\frac{5}{3}.10=16.67Nm$$

$$\tau_2=M_{12}\ddot{q}_1=0.25\cos(q_1-q_2).10=2.5\cos(q_1-q_2)Nm$$

$|\tau_2|\le 2.5Nm$ (tối đa khi $\cos(q_1-q_2)=1$).

Mô men tại khớp 2(τ_2):

Khi $\ddot{q}_2 = \alpha$, $\ddot{q}_1$=0(chuyển động khớp 2, khớp 1 giữ cố định):

$$\tau_2 = M_{22}\ddot{q}_2 = \frac{1}{12}.10 = 0.833Nm$$

$$\tau_1 = M_{12}\ddot{q}_2 = 0.25\cos(q_1 - q_2).10 = 2.5\cos(q_1 - q_2)Nm$$

$$|\tau_1| \le 2.5Nm.$$

Tuy nhiên, động cơ tại khớp 1 phải chịu tải của cả hai thanh, đặc biệt khi cánh tay duỗi thẳng $(q_1 - q_2 = 0)$:

$$\tau_1 = (M_{11} + M_{12})\alpha = \left(\frac{5}{3} + 0.25\right).10 = 1,9167.10 = 19.17Nm$$

Động cơ tại khớp 2 chịu mô men lớn nhất khi khớp 1 chuyển động:

$\tau_2 = 2.5Mm$ (Từ $M_{12}\alpha$)

Mô men tối đa:

- Khớp 1:$1,5.19,17 \approx 28.76$Nm
- Khớp 2: $1,5.2,5 = 3.75Nm$

Đề xuất:

- Động cơ cho khớp 1: Chọn động cơ bước có mô men giữ tối thiểu 28.76 Nm (có thể làm tròn thành 30 Nm để dễ tìm trên thị trường).
- Động cơ cho khớp 2: Chọn động cơ bước có mô men giữ tối thiểu 3.75 Nm (có thể chọn 4–5 Nm).

3.3.3. Lựa chọn phần cứng:

Động cơ cho khớp 1: NEMA 57 (mô men giữ tối thiểu 30 Nm):

Thông số kỹ thuật chi tiết của động cơ bước NEMA 57:

- Kích thước: 57x57mm (mặt bích). Chiều dài thân có thể khác nhau tùy theo loại động cơ, ví dụ 56mm, 76mm, 80mm, 82mm, 102mm, 112mm.
- Số pha: Thường là 2 pha.
- Số bước: 1 bước 1.8 độ.
- Dòng điện: 3A đến 4.5A, tùy thuộc vào loại động cơ.
- Mô-men xoắn: 1.2Nm đến 3Nm, tùy thuộc vào loại động cơ.
- Đường kính trục: 6.35mm hoặc 8mm, tùy thuộc vào loại động cơ.
- Chiều dài trục: Tùy theo loại động cơ, có thể là 20mm hoặc 33mm.
- Số dây: Thường là 4 dây.

Động cơ cho khớp 2: NEMA 34:

Thông số kỹ thuật điển hình của động cơ NEMA 34:

- Kích thước mặt bích: 86x86mm
- Chiều dài thân: Các kích thước khác nhau, ví dụ: 78mm, 114mm, 156mm
- Moment xoắn: Các mức khác nhau, ví dụ: 4Nm, 8Nm, 12Nm
- Dòng điện: 4A
- Góc bước: 1.8°/step
- Đường kính trục: 14mm
- Các thông số khác:
    - Số pha: 2 pha
    - Số dây: 4 dây
    - Độ dài trục: 35mm
    - Dòng tải: 4A/5.8A

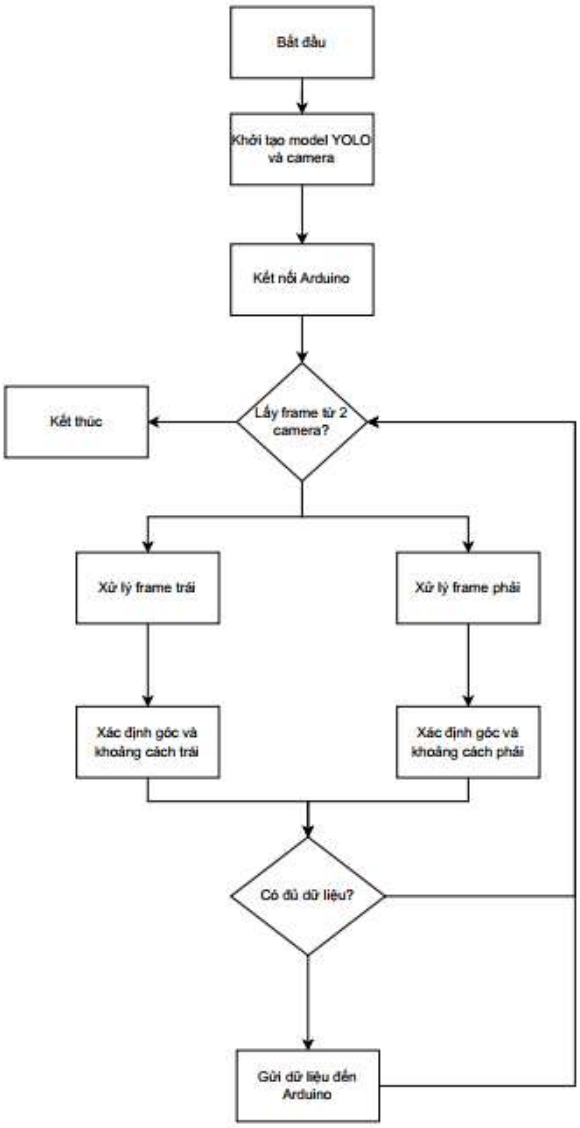
Lựa chọn Driver cho stepper với các thông số sau:

Thông số kỹ thuật:

- **Điện áp hoạt động:** 10-35VDC.
- **Dòng tải tối đa:** 3A, dòng đỉnh 3.5A.
- **Opto cách ly:** Tích hợp 6N137 để cách ly tín hiệu điều khiển.
- **Tản nhiệt:** Tích hợp tản nhiệt nhôm lớn.
- **Dòng điện tối đa:** 3A, dòng đỉnh 3.5A.
- **Vi bước:** Có thể điều chỉnh vi bước (1:1, 1:2, 1:8, 1:16).
- **Decay:** Có chế độ Decay (lực giữ vị trí cố định).
- **Thích hợp:** Với động cơ bước 43, 57, 86 | 2 hoặc 4 pha | 4 dây hoặc 6 dây.
- **Kích thước:** 75x50x35 mm.
- **Công tắc:** Có các công tắc để thiết lập dòng tải, vi bước và chế độ Decay.

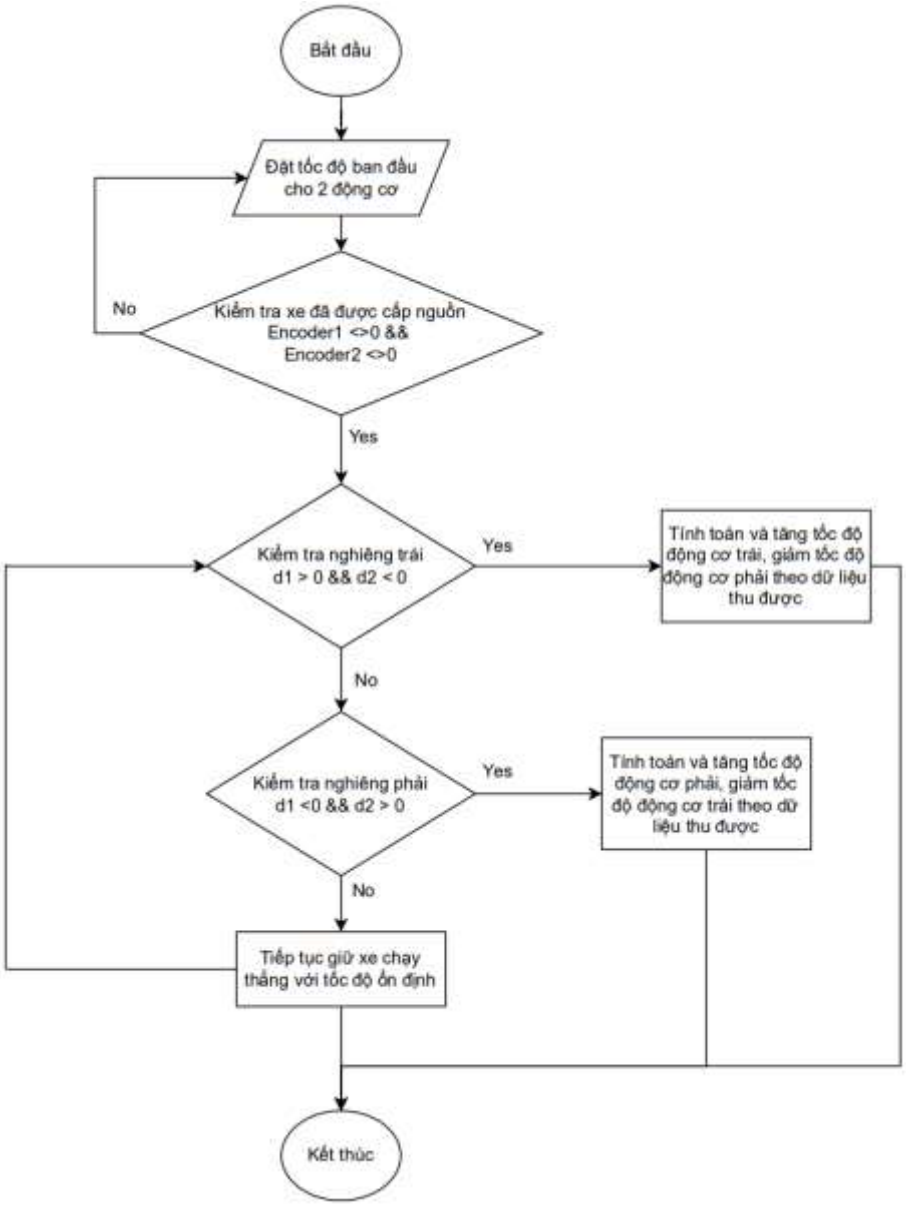
3.4. Lưu đồ thuật toán:

Khối thị giác máy tính:



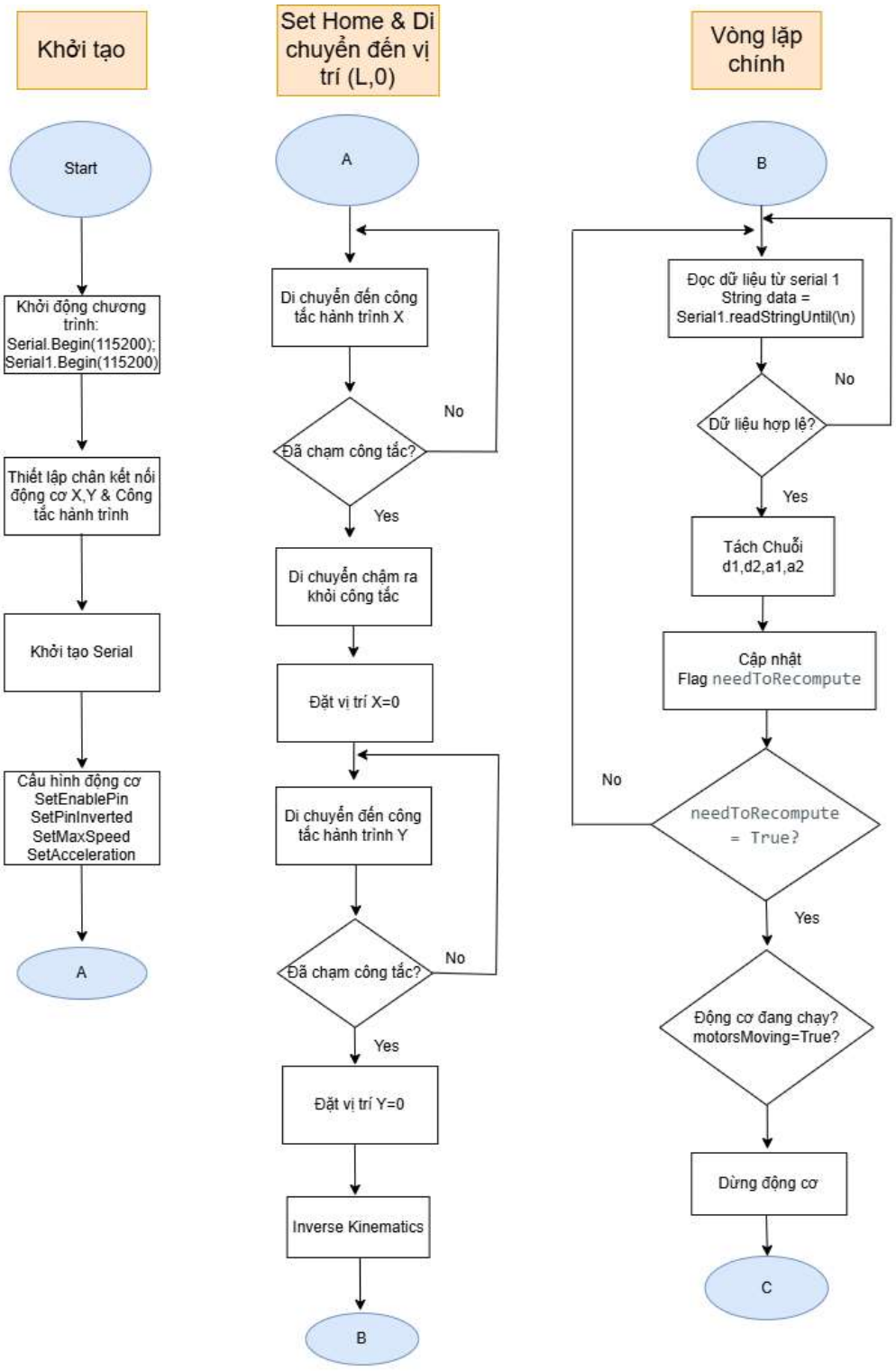
Hình 3.5. Lưu đồ thuật toán khối thị giác máy tính

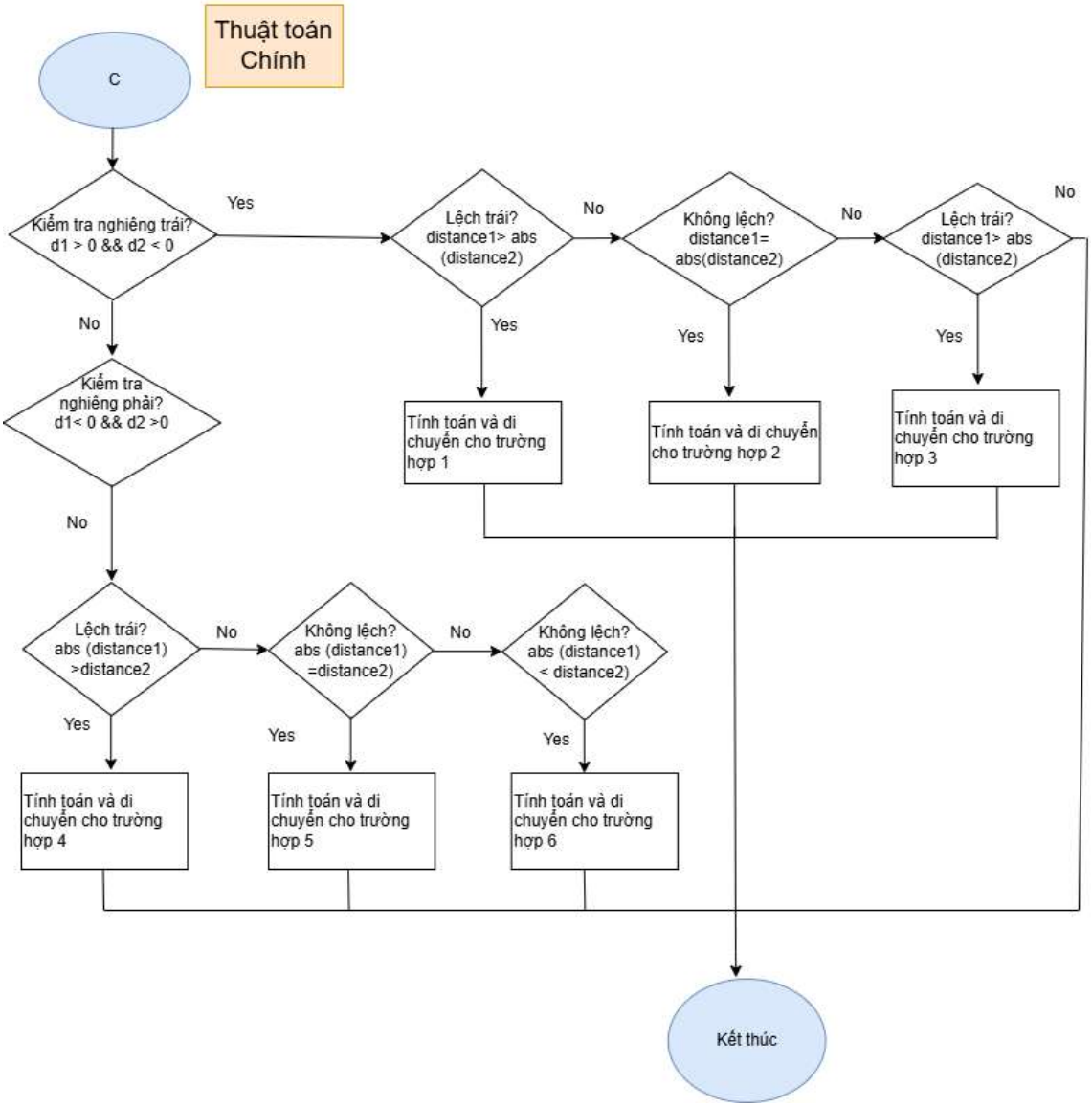
Điều hướng:



Hình 3.6. Lưu đồ thuật toán Điều hướng xe

Cánh tay Robot:


Khởi tạo
Start
Khởi động chương trình: Serial.Begin(115200); Serial1.Begin(115200)
Thiết lập chân kết nối động cơ X,Y & Công tắc hành trình
Khởi tạo Serial
Cấu hình động cơ SetEnablePin SetPinInverted SetMaxSpeed SetAcceleration
A
Set Home & Di chuyển đến vị trí (L,0)
A
Di chuyển đến công tắc hành trình X
Đã chạm công tắc?
No
Yes
Di chuyển chậm ra khỏi công tắc
Đặt vị trí X=0
Di chuyển đến công tắc hành trình Y
Đã chạm công tắc?
No
Yes
Đặt vị trí Y=0
Inverse Kinematics
B
Vòng lặp chính
B
Đọc dữ liệu từ serial 1 String data = Serial1.readStringUntil(\n)
Dữ liệu hợp lệ?
No
Yes
Tách Chuỗi d1,d2,a1,a2
Cập nhật Flag needToRecompute
needToRecompute = True?
No
Yes
Động cơ đang chạy? motorsMoving=True?
Dừng động cơ
C



Hình 3.7. Lưu đồ thuật toán cánh tay Robot

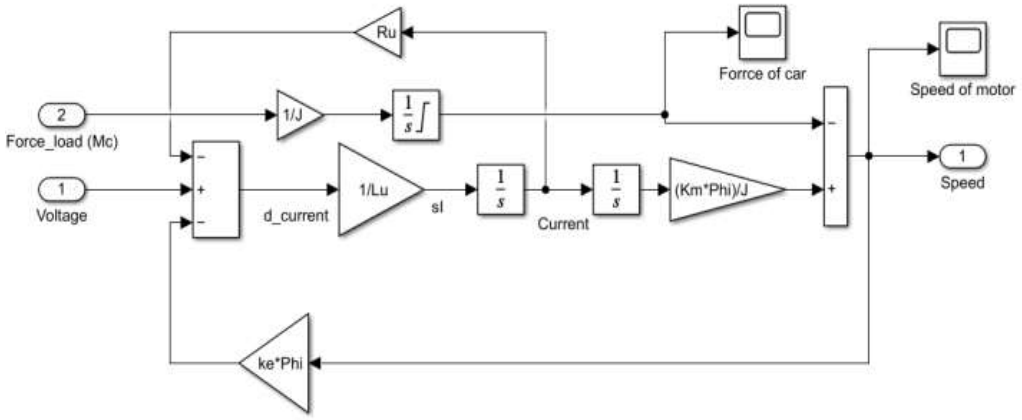
Chương 4: MÔ HÌNH THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Mô phỏng động học xe :

4.1.1. Mục tiêu mô phỏng:

- Xe di chuyển đúng theo quỹ đạo dây thép

4.1.2. Mô phỏng Matlab và nhận xét:

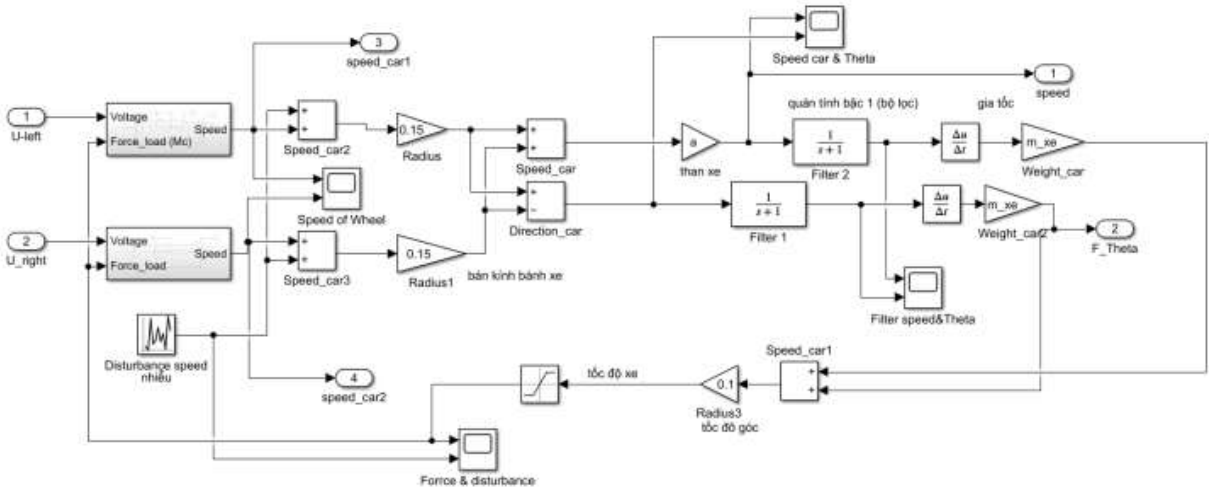


Hình 4.1. Sơ đồ khối mô hình hóa động cơ một chiều

Lực tải tác động lên hệ thống:

$$M_c = F(s) = m_{xe}.a \tag{4.1}$$

Với M_c gồm $F_{\text{resistance}}$, F_{noise}, M_{quay}



Hình 4.2. Mô hình tính M_c, tốc độ xe, M_{quay}

Hàm truyền hệ hở của động cơ:

$$G(s) = \frac{\mathrm{w}(s)}{V_a(s)} = \frac{K_m}{(sJ)(sL_a + R_a) + K_m K_e} \quad (4.2)$$

Dựa vào datasheet của động cơ MY1016Z2-350W

$P_{dm} = 350W$

$U_{dm} = 24V$

$\omega_{dm} = 3500RPM$ (No gear box)

Sau khi đi qua hộp số:

$\omega_{Idm} = 350RPM$

$I_{dm} = 13.4A$

$R_a = 0.7(\Omega)$

$L_a = 0.00047(H)$

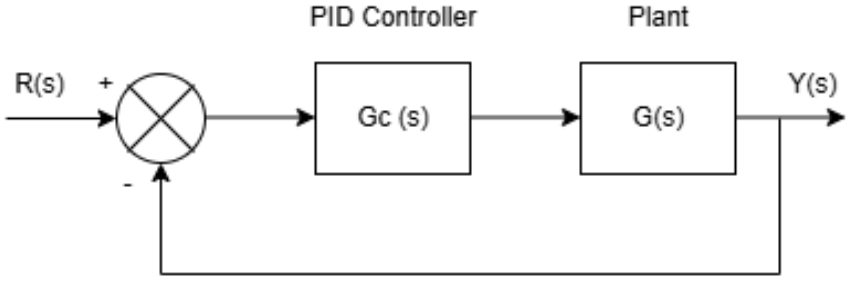
$$K_M = \frac{T}{I} = \frac{P/\omega'}{I} = \frac{P/\frac{\omega 2\pi}{60}}{I} = 0.07Nm/A$$

$$K_e = \frac{V}{\omega'} = \frac{V}{\frac{\omega 2\pi}{60}} = 0.07Vs/rad$$

$J = 0,00015Kg.m^2$

Ta có: $$G(s) = \frac{0,06}{7,05.10^{-7}s^2 + 1,05.10^{-4}s + 4,87.10^{-3}}$$

Bộ điều khiển PID được thiết kế bằng phương pháp điều chỉnh Zeigler-Nichols:



Hình 4.3. Sơ đồ khối vòng kín

$$G_c(s) = K_p + \frac{K_i}{s} + K_p s = K_p(1 + \frac{T_i}{s} + Tds)$$

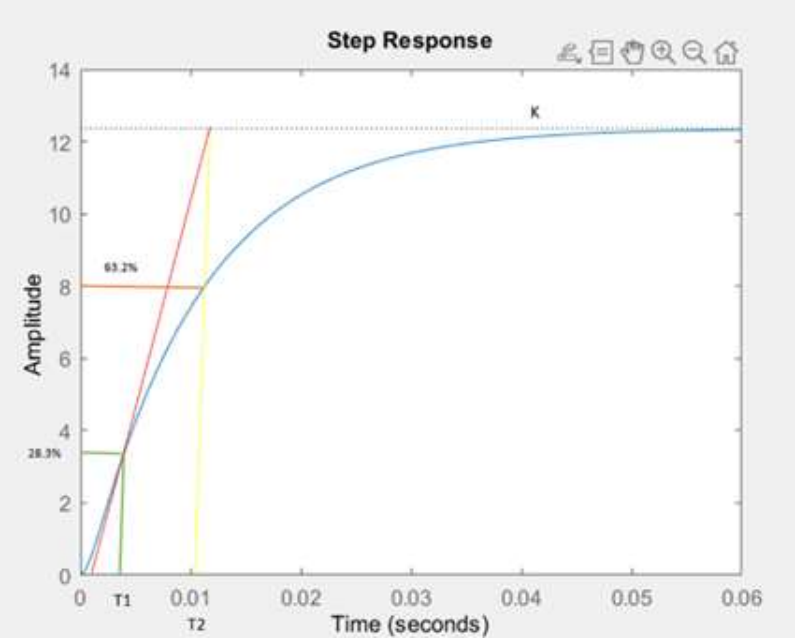
- Chạy mô phỏng MATLAB với đầu vào dưới dạng hàm bước đơn vị. Sử dụng phương pháp tham chiếu hai điểm:

- Tại hai điểm cụ thể trên phản hồi bước:

t1: Thời gian tương ứng với 28.3% thay đổi đầu ra (Δy).

t2: Thời gian tương ứng với 63,2% thay đổi đầu ra (Δy).

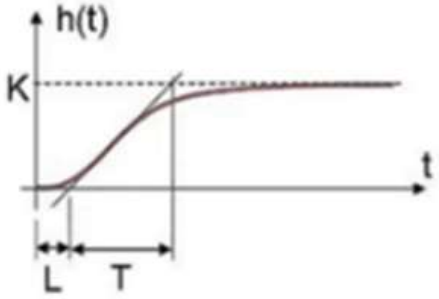
- Thời gian trễ: $L = 1.5(t_1 - \frac{t_2}{3}) = t_2 - T$



- Từ biểu đồ, giá trị cuối cùng của phản hồi:

y∞= K= 12,7

vậy: T = 0.009, L = 0.001

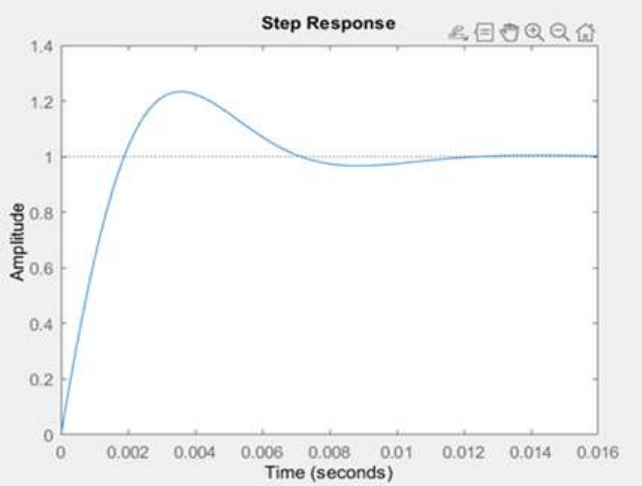


	K_P	K_I	K_D
P	T/L	0	0
PI	0.9 T/L	0.3/L	0
PID	1.2 T/L	0.5/L	0.5L

- Thông số bộ điều khiển PID:

Kp = 10.8, Ki = 500, Kd = 0.0005

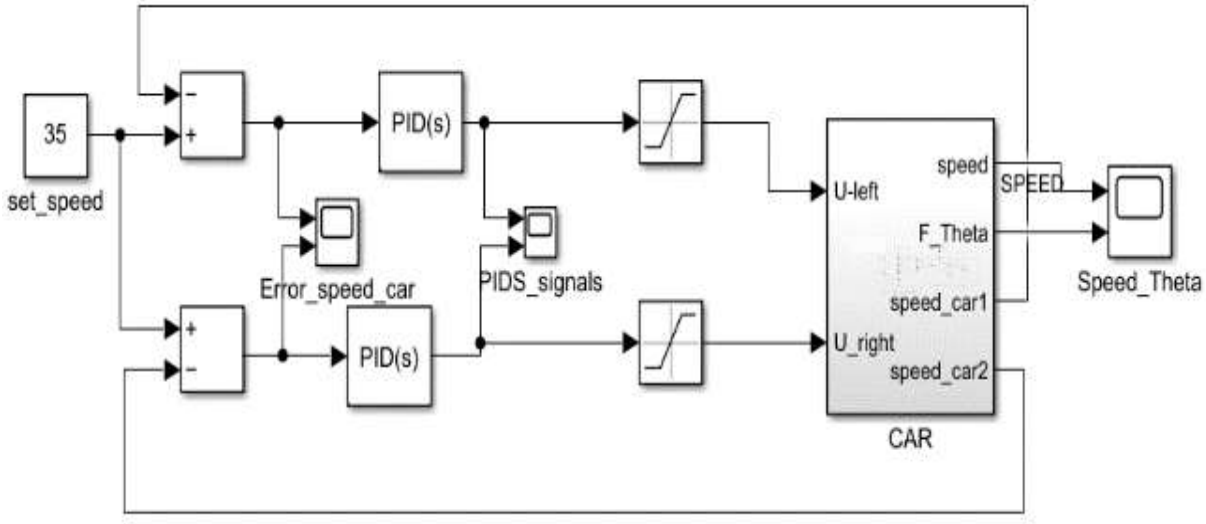
$$Gc(s) = 10.8 + \frac{500}{s} + 0.0005s$$



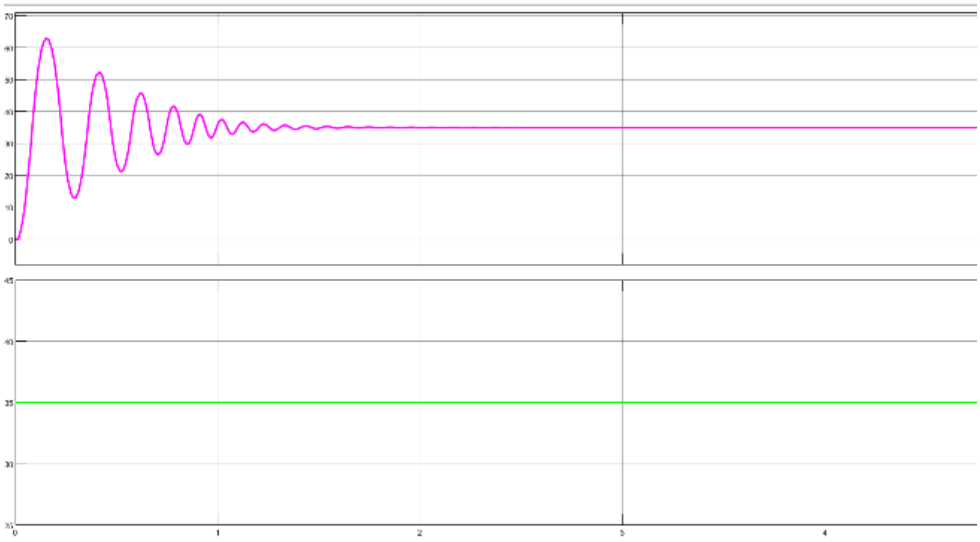
Hình 4.4. Đáp ứng từng bước của hệ thống với điều khiển vòng kín

Hệ thống có phản hồi, giúp điều chỉnh lỗi và cung cấp phản hồi đầu ra nhanh hơn.

- Đường cong phản hồi thể hiện quá độ và dao động nhẹ trước khi ổn định.
- Thời gian xác lập nhanh hơn so với hệ thống hở vòng còn giao động nhỏ xung quanh giá trị mong muốn

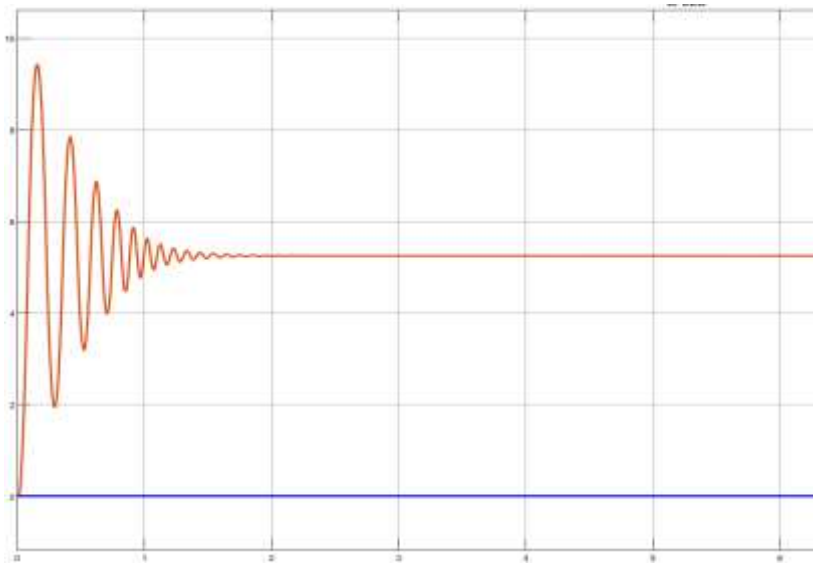


Hình 4.5. Mô phỏng MATLAB cho động cơ



- Tốc độ thực của động cơ có thời gian xác lập khoảng 2 giây để ổn định về giá trị mong muốn là 35. Sau thời gian này, tốc độ bám sát tốt giá trị đặt với sai số gần như bằng không. Tuy nhiên, hệ thống có độ lố lớn, đạt giá trị cực đại khoảng 70. Trong giai đoạn quá độ, hệ thống dao động mạnh nhưng tắt dần sau khoảng 2 giây trước khi ổn định hoàn toàn. Nhìn chung, hệ thống đảm bảo bám sát giá trị mong muốn sau quá trình quá độ.

$$speed_{xe} = R_{wheel}.\omega = 0.15.35 = 5.25(rad/s)$$



- Tốc độ thực của động cơ có thời gian xác lập khoảng 2 giây để đạt giá trị mong muốn là 5 rad/s. Hệ thống có độ lố lớn, với giá trị cực đại đạt khoảng 10 rad/s. Trong giai đoạn quá độ, tốc độ dao động mạnh với biên độ giảm dần và ổn định hoàn toàn sau khoảng 2 giây. Sau khi ổn định, tốc độ bám sát giá trị mong muốn với sai số gần như bằng không. Đường màu xanh biểu diễn mô-men quay giữ giá trị không đổi theo thời gian, điều này cho thấy mô-men tác động đến hệ thống không thay đổi và không bị ảnh hưởng bởi dao động của tốc độ động cơ. Nhìn chung, hệ thống đáp ứng tốt yêu cầu bám giá trị mong muốn nhưng có độ quá điều chỉnh cao

4.2. Mô hình thực nghiệm

4.2.1. Cấu tạo hệ thống

Hệ thống thực nghiệm gồm các thành phần chính:

- Khung xe: Khung xe lắp 2 động cơ DC bánh trước, 2 bánh tự do phía sau.
- Camera: 2 camera USB đặt trước và sau xe, mỗi camera cách thân xe 40 cm, khoảng cách giữa hai camera là 130 cm.
- Bộ điều khiển:
    - Máy tính xử lý hình ảnh, sử dụng YOLOv8-seg nhận dạng dây.
    - Arduino Mega điều khiển động cơ, nhận dữ liệu qua Serial.

Động cơ:

- 2 động cơ DC điều khiển tốc độ bằng PID.
- Cánh tay robot hai bậc tự do gắn động cơ bước
- Nguồn cấp: Hai ắc-quy 12V mắc nối tiếp và nguồn 5V trực tiếp từ laptop cho camera và Arduino.

Hình 4.6. Tổng quan mô hình xe

4.2.2. Quá trình thực nghiệm

Bước 1: Gắn các thành phần lên xe, kiểm tra kết nối cơ khí và điện tử.

Bước 2: Chạy thử phần xử lý hình ảnh, kiểm tra độ chính xác phát hiện dây.

Bước 3: Truyền dữ liệu góc và khoảng cách sang Arduino, kiểm tra phản hồi điều khiển động cơ.

Bước 4: Cho xe chạy thử trên dây thép thật, ghi nhận kết quả.

4.2.3. Sự cố và biện pháp xử lý:

Sự cố gặp phải	Nguyên nhân chính	Biện pháp khắc phục
Dữ liệu không truyền được từ Python	Lỗi cổng Serial hoặc tốc độ baud	Kiểm tra cáp USB, thiết lập baud đúng
Xe lắc lư mạnh khi điều chỉnh	PID chưa được tinh chỉnh	Giảm Kp, tăng Ki để giảm dao động
Hình ảnh bị mờ hoặc mất kết nối camera	Cáp USB kém chất lượng hoặc nhiễu nguồn	Dùng cáp chống nhiễu, cấp nguồn riêng
Xe không nhận diện được dây cong	YOLOv8 chưa đủ dữ liệu huấn luyện	Gắn thêm ảnh dây cong vào tập train

4.3. Kết quả thực nghiệm:

4.3.1. Đánh giá khả năng điều hướng của xe robot:

- Kết quả thực nghiệm được đo lường sau 5 lần chạy thử:

Lần chạy	Góc lệch ban đầu (độ)	Khoảng cách ban đầu (cm)	Thời gian ổn định (s)	Sai số góc cuối (độ)	Sai số vị trí cuối (cm)
1	12	18	4,5	1,5	3,2
2	-8	22	3,9	1,1	2,1
3	10	15	3,2	0,8	2,0
4	0	10	2,0	0,3	0,9
5	-5	20	3,6	1,0	1,5

- Nhận xét:
    - Hệ thống thực tế hoạt động tương đối ổn định, sai số góc và vị trí đều nhỏ hơn 3 độ và 3,5 cm.
    - Thời gian ổn định trung bình khoảng 3,5 giây, phù hợp với yêu cầu điều khiển thời gian thực.
    - So với mô phỏng, sai số có cao hơn nhẹ do nhiễu, ma sát và sai số cơ khí.

Đánh giá hoạt động của cánh tay robot sơn chỉ đường:

- Cánh tay được điều khiển theo hai góc:
    - θ_1 (khớp gốc): quay theo trục dọc thân xe.

- θ_2 (khớp giữa): điều chỉnh độ vươn tới điểm cần sơn.

- Thông số đánh giá gồm:
  - Góc yêu cầu: (θ_{1_yc}, θ_{2_yc})
  - Góc thực tế đo được sau điều khiển: (θ_{1_tt}, θ_{2_tt})
  - Vị trí đầu bút sơn theo hệ tọa độ xe (x, y)
  - Sai số vị trí cuối cùng
  - Thời gian điều khiển

Lần thử	θ_1_yc	θ_2_yc	θ_1_tt	θ_2_tt	**Sai số vị trí (cm)**	**Thời gian phản hồi (s)**
1	45	60	39.5	52.2	7.3	2.3
2	30	45	26.1	39.4	5.6	2.0
3	60	75	53.4	66.0	7.9	2.6
4	15	30	12	24.7	5.1	1.7
5	90	60	82.8	52.5	8.0	2.9

- Nhận xét
  - Sai số vị trí đầu mút dao động trong khoảng 5–8 cm, lớn hơn so với yêu cầu chính xác khi sơn vạch đường.
  - Nguyên nhân có thể đến từ:
    - Độ rơ cơ khí tại khớp nối và động cơ.
    - Chưa hiệu chỉnh chính xác hệ tọa độ điều khiển và tính toán góc.
    - Tốc độ quay động cơ vượt quá khả năng kiểm soát, gây bỏ bước.
  - Thời gian phản hồi vẫn dưới 3 giây, cho thấy hệ thống điều khiển thời gian thực hoạt động tốt về tốc độ, nhưng cần cải thiện độ chính xác định vị.

Hệ thống có thể sử dụng cho các ứng dụng không đòi hỏi chính xác cao, nhưng chưa đủ ổn định để sơn vạch chính xác nếu không được hiệu chỉnh kỹ hơn.

KẾT LUẬN

1. Kết luận chung

- Sau quá trình nghiên cứu, thiết kế, mô phỏng và thực nghiệm, đề tài "Robot sơn chỉ đường dùng thị giác máy tính" đã hoàn thành mục tiêu đặt ra. Hệ thống được xây dựng bao gồm mô hình điều hướng xe bám dây thép sử dụng camera kép, kết hợp thuật toán YOLOv8-seg để nhận diện vật thể và tính toán hình học tuyến để xác định vị trí – góc lệch. Dữ liệu này được truyền đến Arduino để điều khiển động cơ DC bằng PID, đồng thời kết hợp với cánh tay robot hai bậc tự do sử dụng động cơ bước để thực hiện nhiệm vụ sơn vạch theo thời gian thực.

2. Những kết quả đạt được

- Khả năng điều hướng tương đối chính xác: Hệ thống có thể tự căn chỉnh theo dây thép với sai số góc < 5° và sai số vị trí < 5 cm sau khi ổn định.
- Xử lý ảnh hiệu quả: Mô hình YOLOv8-seg huấn luyện riêng cho dây thép đạt độ chính xác cao trong điều kiện thực tế.
- Truyền và điều khiển thời gian thực: Giao tiếp giữa máy tính và Arduino ổn định qua Serial, đảm bảo dữ liệu liên tục phục vụ điều khiển.
- Cánh tay robot hai bậc tự do hoạt động tương đối: Điều khiển động cơ bước tương đối, sai số vị trí đầu mút < 8 cm, cần phải hiệu chỉnh thêm.
- Hệ thống hoàn chỉnh có tính ứng dụng cao: Hệ thống phù hợp cho các ứng dụng sơn vạch tạm thời, thi công đánh dấu khu vực tạm thời, hoặc ứng dụng tự động hóa trong công nghiệp.

3. Những đóng góp của đề tài

- Tích hợp thị giác máy tính và điều khiển nhúng: Kết hợp hai lĩnh vực quan trọng để tạo ra một giải pháp thông minh trong robot tự hành.
- Xây dựng mô hình mô phỏng và thực nghiệm đồng bộ: Từ đó dễ dàng kiểm chứng và tối ưu hệ thống.
- Xây dựng bộ dữ liệu chuyên biệt cho bài toán nhận diện dây thép bằng YOLOv8-seg, có thể dùng làm tham khảo cho các ứng dụng tương tự.
- Giải pháp điều khiển cánh tay robot theo thời gian thực dựa vào dữ liệu từ camera, mở ra khả năng ứng dụng vào robot công nghiệp, robot thi công ngoài hiện trường.

4. Đề xuất và kiến nghị

- Tăng cường độ chính xác xử lý ảnh: Bổ sung nhiều trường hợp dây cong, ánh sáng phức tạp vào bộ dữ liệu huấn luyện để tăng tính ổn định cho mô hình.
- Nâng cấp cơ khí và động cơ: Thay động cơ bước thường bằng loại có encoder hoặc dùng servo để tăng độ chính xác.

- Tối ưu thuật toán điều khiển: Tích hợp điều khiển thích nghi hoặc học máy để tự điều chỉnh PID theo môi trường.
- Triển khai phần mềm điều khiển giao diện người dùng: Giúp theo dõi trạng thái hệ thống và điều chỉnh dễ dàng trong quá trình vận hành.
- Thử nghiệm hệ thống ngoài trời: Để đánh giá độ bền và tính ứng dụng thực tiễn trong điều kiện thời tiết thật.

TÀI LIỆU THAM KHẢO

1. Radim Hercik, Radek Byrtus, Rene Jaros, Jiri Koziorek. Implementation of Autonomous Mobile Robot in SmartFactory.

2. Xingshuai Dong, Massimiliano L. Cappuccio. Applications of Computer Vision in Autonomous Vehicles: Methods, Challenges and Future Directions

3. Siddhi Lahange, Prashansa Nalawade, Pramod Bide, Deep Nayak, Atharva Mohite. Computer Vision Techniques in Autonomous Vehicles: A Survey.

PHỤ LỤC

PHỤ LỤC 1: PHẦN MỀM ARDUINO IDE

Arduino IDE (Integrated Development Environment - Môi trường Phát triển Tích hợp) là một phần mềm mã nguồn mở được thiết kế để lập trình cho các vi điều khiển thuộc hệ sinh thái Arduino. Được phát triển bởi cộng đồng Arduino, phần mềm này cho phép người dùng, từ những người mới bắt đầu đến các kỹ sư chuyên nghiệp, viết và nạp mã lệnh (firmware) vào các bo mạch Arduino như Arduino Uno, Arduino Mega, Arduino Nano, và nhiều biến thể khác. Arduino IDE được sử dụng rộng rãi trong các dự án điện tử, IoT (Internet of Things), robot, và các ứng dụng sáng tạo khác.

Phần mềm Arduino IDE hỗ trợ nhiều hệ điều hành phổ biến, bao gồm Windows, macOS và Linux, giúp nó trở thành một công cụ linh hoạt và dễ tiếp cận. Giao diện đơn giản, dễ sử dụng cùng với ngôn ngữ lập trình dựa trên C/C++ đã làm cho Arduino IDE trở thành lựa chọn hàng đầu cho các nhà phát triển, sinh viên và những người đam mê công nghệ muốn khám phá thế giới vi điều khiển.

Phiên bản mới nhất của Arduino IDE tại thời điểm hiện tại (tháng 6/2025) là Arduino IDE 2.x, ra mắt với nhiều cải tiến so với phiên bản 1.x, bao gồm giao diện hiện đại hơn, hiệu suất cải thiện, và tích hợp các tính năng như tự động hoàn thành mã, gỡ lỗi trực tiếp, và hỗ trợ tốt hơn cho các thư viện bên thứ ba.

2. Các tính năng chính của Arduino IDE

Arduino IDE cung cấp một bộ tính năng mạnh mẽ, giúp đơn giản hóa quá trình lập trình và phát triển các dự án dựa trên vi điều khiển. Dưới đây là những tính năng nổi bật:

- Giao diện đơn giản và thân thiện với người dùng: Arduino IDE có giao diện tối giản với các thành phần chính như trình soạn thảo mã, cửa sổ thông báo, và các nút điều khiển (như nạp mã, kiểm tra mã). Điều này giúp người mới bắt đầu dễ dàng làm quen mà không cần kiến thức sâu về lập trình.
- Hỗ trợ ngôn ngữ lập trình C/C++ đơn giản hóa: Arduino IDE sử dụng một phiên bản đơn giản của C/C++, được gọi là Arduino Language, với các hàm và thư viện được thiết kế sẵn để điều khiển các thành phần phần cứng như cảm biến, động cơ, đèn LED, và màn hình.
- Quản lý thư viện mạnh mẽ: Arduino IDE tích hợp Library Manager, cho phép người dùng dễ dàng cài đặt và quản lý các thư viện mã nguồn mở để hỗ trợ các cảm biến, module, hoặc giao thức giao tiếp như I2C, SPI, và Wi-Fi.
- Board Manager: Tính năng này cho phép người dùng thêm hỗ trợ cho các bo mạch Arduino mới hoặc các bo mạch tương thích từ các nhà sản xuất khác như ESP32, ESP8266, hoặc STM32.
- Trình biên dịch và nạp mã: Arduino IDE tự động biên dịch mã nguồn thành mã máy và nạp trực tiếp vào vi điều khiển thông qua cổng USB hoặc các giao diện khác. Quá trình này được thực hiện chỉ bằng một cú nhấp chuột.
- Hỗ trợ gỡ lỗi (Debugging): Trong phiên bản 2.x, Arduino IDE đã cải thiện khả năng gỡ lỗi, cho phép người dùng kiểm tra mã nguồn, phát hiện lỗi logic, và tối ưu hóa chương trình.
- Hỗ trợ đa nền tảng: Arduino IDE hoạt động trên nhiều hệ điều hành, đảm bảo tính tương thích và khả năng sử dụng rộng rãi.

- Cộng đồng hỗ trợ lớn: Arduino IDE được hỗ trợ bởi một cộng đồng đông đảo, cung cấp hàng ngàn tài liệu, hướng dẫn, và mã nguồn mẫu. Người dùng có thể tìm thấy các dự án mẫu, diễn đàn, và thư viện do cộng đồng đóng góp.

PHỤ LỤC 2: PHẦN MỀM VISUAL STUDIO CODE (VSCODE)

1. Tổng quan về Visual Studio Code

Visual Studio Code (VSCode) là một trình soạn thảo mã nguồn (source-code editor) mã nguồn mở, được phát triển bởi Microsoft và ra mắt lần đầu vào năm 2015. VSCode được thiết kế để hỗ trợ lập trình viên phát triển ứng dụng trên nhiều nền tảng, từ các dự án web, ứng dụng di động, đến lập trình nhúng và IoT. Với giao diện hiện đại, khả năng tùy chỉnh cao, và hỗ trợ nhiều ngôn ngữ lập trình, VSCode đã nhanh chóng trở thành một trong những trình soạn thảo mã phổ biến nhất trên thế giới, được sử dụng bởi cả lập trình viên chuyên nghiệp lẫn người mới học lập trình.

Không giống như Visual Studio (một IDE đầy đủ), VSCode là một trình soạn thảo nhẹ, tập trung vào tốc độ và tính linh hoạt. Tuy nhiên, nhờ hệ thống extension (phần mở rộng) phong phú, VSCode có thể được tùy chỉnh để hoạt động như một IDE mạnh mẽ cho hầu hết các ngôn ngữ lập trình như Python, JavaScript, C++, Java, Go, và thậm chí cả lập trình vi điều khiển như Arduino hoặc ESP32.

VSCode hỗ trợ đa nền tảng, chạy trên Windows, macOS, và Linux, đồng thời cung cấp các phiên bản cho cả máy tính để bàn và trình duyệt (qua VSCode for Web). Tính đến tháng 6/2025, VSCode tiếp tục được cập nhật thường xuyên với các cải tiến về hiệu suất, tích hợp AI (như GitHub Copilot), và hỗ trợ tốt hơn cho các công nghệ mới.

2. Các tính năng chính của VSCode

VSCode nổi bật với bộ tính năng mạnh mẽ và khả năng tùy chỉnh, đáp ứng nhu cầu của nhiều đối tượng lập trình viên. Dưới đây là các tính năng chính:

- Hỗ trợ đa ngôn ngữ lập trình: VSCode hỗ trợ hàng trăm ngôn ngữ lập trình thông qua các phần mở rộng, từ các ngôn ngữ phổ biến như Python, JavaScript, và C++ đến các ngôn ngữ chuyên biệt như Rust, Julia, hoặc VHDL.
- Hệ thống phần mở rộng (Extensions): Kho Extensions Marketplace của VSCode chứa hàng ngàn phần mở rộng, cho phép người dùng thêm các tính năng như gỡ lỗi, tích hợp Git, linter, hoặc hỗ trợ lập trình cho các nền tảng cụ thể (ví dụ: Arduino, PlatformIO).
- Tích hợp Git và quản lý phiên bản: VSCode tích hợp sẵn Git, cho phép người dùng thực hiện các thao tác như commit, push, pull, hoặc giải quyết xung đột trực tiếp từ giao diện. Hỗ trợ các nền tảng như GitHub, GitLab, và Bitbucket cũng được tích hợp chặt chẽ.
- Gỡ lỗi (Debugging) mạnh mẽ: VSCode cung cấp công cụ gỡ lỗi tích hợp, cho phép đặt điểm dừng (breakpoints), kiểm tra biến, và theo dõi luồng thực thi mã. Các phần mở rộng như Debugger for Chrome hoặc Python Debugger giúp tăng cường khả năng này.
- Tự động hoàn thành mã (IntelliSense): IntelliSense cung cấp gợi ý mã, thông tin tham số, và tài liệu tham khảo nhanh, giúp tăng tốc độ viết mã và giảm lỗi cú pháp.
- Giao diện tùy chỉnh: Người dùng có thể tùy chỉnh giao diện VSCode bằng cách thay đổi theme (chủ đề), bố cục, phím tắt, và cài đặt. Các theme phổ biến như Dracula, One Dark Pro, hoặc Material Theme mang lại trải nghiệm cá nhân hóa.
- Terminal tích hợp: VSCode có terminal tích hợp, hỗ trợ PowerShell, Command Prompt, Bash, hoặc các shell khác, giúp lập trình viên thực hiện lệnh mà không cần rời khỏi trình soạn thảo.

- Hỗ trợ phát triển web và đám mây: VSCode cung cấp các tính năng như Live Server (xem trước web trực tiếp), Remote Development (làm việc trên máy chủ từ xa), và tích hợp với các nền tảng đám mây như AWS hoặc Azure.
- Tích hợp AI và công cụ hỗ trợ lập trình: Các phần mở rộng như GitHub Copilot hoặc Tabnine cung cấp gợi ý mã dựa trên AI, giúp tăng năng suất lập trình.
- Hỗ trợ lập trình nhúng: Với các phần mở rộng như PlatformIO hoặc Arduino, VSCode có thể được sử dụng để lập trình vi điều khiển, thay thế hoặc bổ sung cho Arduino IDE.

3. Cách sử dụng Visual Studio Code

Để bắt đầu sử dụng VSCode, người dùng cần thực hiện các bước cơ bản sau:

- Tải và cài đặt VSCode:
    - Truy cập trang web chính thức (code.visualstudio.com) để tải phiên bản phù hợp với hệ điều hành (Windows, macOS, hoặc Linux).
    - Cài đặt phần mềm theo hướng dẫn, quá trình này thường nhanh chóng và đơn giản.
- Cấu hình môi trường:
    - Mở VSCode và cài đặt các phần mở rộng cần thiết thông qua Extensions Marketplace (Ctrl+Shift+X hoặc Cmd+Shift+X trên macOS).
    - Ví dụ: Cài đặt Python extension cho lập trình Python hoặc PlatformIO cho lập trình nhúng.
    - Tùy chỉnh giao diện và phím tắt trong menu File > Preferences > Settings.
- Mở hoặc tạo dự án:
    - Mở một thư mục dự án bằng cách vào File > Open Folder hoặc tạo tệp mới với các phần mở rộng như .py, .js, hoặc .cpp.
    - VSCode tự động nhận diện ngôn ngữ và cung cấp các tính năng như gợi ý mã hoặc định dạng tự động.
- Viết và chạy mã:
    - Viết mã trong trình soạn thảo. Ví dụ, một chương trình Python đơn giản:

python

print("Hello, World!")

    - Chạy mã bằng cách sử dụng terminal tích hợp hoặc phần mở rộng như Code Runner. Với Python, nhấn Run Python File (nút tam giác ở góc trên bên phải).
- Gỡ lỗi mã:
    - Vào tab Run and Debug (Ctrl+Shift+D), thiết lập cấu hình gỡ lỗi (debug configuration) cho ngôn ngữ bạn đang sử dụng.
    - Đặt các điểm dừng và chạy chương trình để kiểm tra lỗi.
- Quản lý mã nguồn với Git:
    - Kết nối với kho lưu trữ Git bằng cách vào tab Source Control (Ctrl+Shift+G).

    - Thực hiện các thao tác như commit, push, hoặc pull trực tiếp từ giao diện.
- Cài đặt phần mở rộng cho vi điều khiển (nếu cần):
    - Cài đặt PlatformIO hoặc Arduino extension để lập trình bo mạch như Arduino hoặc ESP32.
    - Cấu hình bo mạch và cổng tương tự như Arduino IDE, sau đó biên dịch và nạp mã.

PHỤ LỤC 3 : PHẦN MỀM YOLOV8 VÀ SỬ DỤNG TRÊN GOOGLE COLAB

1. Tổng quan về YOLOv8:

YOLOv8 (You Only Look Once version 8) là phiên bản mới nhất trong dòng mô hình YOLO, được phát triển bởi Ultralytics, công ty đứng sau YOLOv5. Ra mắt vào năm 2023, YOLOv8 là một mô hình học sâu (deep learning) tiên tiến, được thiết kế cho các tác vụ thị giác máy tính như phát hiện đối tượng (object detection), phân đoạn hình ảnh (instance segmentation), phân loại hình ảnh (image classification), theo dõi đối tượng (object tracking), và ước lượng tư thế (pose estimation). YOLOv8 cải thiện đáng kể về tốc độ, độ chính xác, và tính dễ sử dụng so với các phiên bản trước, đặc biệt là YOLOv5.

YOLOv8 được xây dựng dựa trên framework PyTorch, cung cấp một giao diện thân thiện với người dùng thông qua Python API và Command Line Interface (CLI). Mô hình này có các biến thể từ nhỏ gọn (YOLOv8n - nano) đến lớn (YOLOv8x - extra-large), phù hợp cho cả thiết bị nhúng (edge devices) và máy chủ GPU mạnh mẽ. YOLOv8 được cấp phép dưới AGPL-3.0 (mã nguồn mở) và Enterprise License (thương mại), với mã nguồn công khai trên GitHub.

Google Colab là một nền tảng đám mây miễn phí của Google, cung cấp môi trường Jupyter Notebook để chạy mã Python với tài nguyên GPU/TPU miễn phí (giới hạn với người dùng miễn phí). Sử dụng YOLOv8 trên Google Colab là một lựa chọn phổ biến vì nó không yêu cầu cài đặt phức tạp trên máy cục bộ, tận dụng sức mạnh tính toán của đám mây, và tích hợp tốt với Google Drive để lưu trữ dữ liệu và mô hình.

2. Các tính năng chính của YOLOv8 :

YOLOv8 được thiết kế với nhiều cải tiến so với YOLOv5, mang lại hiệu suất vượt trội và tính linh hoạt:

- Hiệu suất cao: YOLOv8 đạt độ chính xác cao hơn (mAP cao hơn trên tập COCO) và tốc độ suy luận nhanh, với các mô hình nano đạt hàng trăm FPS trên GPU mạnh.
- Hỗ trợ đa tác vụ: Ngoài phát hiện đối tượng, YOLOv8 hỗ trợ phân đoạn, phân loại, theo dõi, và ước lượng tư thế, tất cả trong một framework thống nhất.
- Kiến trúc cải tiến: Sử dụng các khối C2f (CSP Bottleneck with 2 convolutions) thay vì C3 của YOLOv5, cải thiện hiệu quả tính toán và giảm tham số.
- Tăng cường dữ liệu nâng cao: Áp dụng Mosaic Augmentation (tắt trước 10 epoch cuối), MixUp, và Cutout để tăng độ bền vững của mô hình.
- Tối ưu hóa huấn luyện: Hỗ trợ Ultralytics Tuner để tự động điều chỉnh siêu tham số (hyperparameters) và các kỹ thuật như early stopping để tránh quá khớp (overfitting).
- Hỗ trợ xuất mô hình: Xuất sang các định dạng như ONNX, CoreML, TFLite, TensorRT, và OpenVINO, phù hợp cho triển khai trên nhiều nền tảng.
- Tích hợp dễ dàng: Hỗ trợ PyTorch Hub, CLI, và tích hợp với các công cụ như Roboflow, ClearML, và Comet để quản lý dữ liệu và theo dõi huấn luyện.
- Cải thiện trải nghiệm người dùng: CLI và Python API đơn giản, cùng tài liệu chi tiết từ Ultralytics, giúp người dùng dễ dàng bắt đầu.

3. Sử dụng YOLOv8 trên Google Colab

Google Colab là môi trường lý tưởng để huấn luyện và suy luận với YOLOv8, đặc biệt cho người dùng không có GPU mạnh. Dưới đây là hướng dẫn chi tiết từng bước để sử dụng

YOLOv8 trên Google Colab, bao gồm cài đặt, suy luận, và huấn luyện trên tập dữ liệu tùy chỉnh.

3.1. Thiết lập môi trường trên Google Colab:

- Mở Google Colab:
    - Truy cập colab.research.google.com và đăng nhập bằng tài khoản Google.
    - Tạo một notebook mới bằng cách nhấp vào File > New Notebook.
- Kích hoạt GPU:
    - Chuyển sang runtime GPU: Nhấp vào Runtime > Change runtime type, chọn GPU (thường là NVIDIA T4 với tài khoản miễn phí) và lưu.
    - Kiểm tra GPU bằng lệnh:

bash

!nvidia-smi

- Cài đặt Ultralytics YOLOv8:
    - Cài đặt thư viện Ultralytics qua pip:

bash

!pip install ultralytics

    - Kiểm tra cài đặt:

python

from ultralytics import YOLO

print("Ultralytics installed successfully!")

- Kết nối Google Drive (tùy chọn):
    - Để lưu trữ dữ liệu và mô hình, kết nối Google Drive:

python

from google.colab import drive

drive.mount('/content/drive')

    - Sau khi chạy, cấp quyền truy cập và sao chép mã xác thực vào ô nhập liệu.

3.2. Suy luận với mô hình YOLOv8 được huấn luyện trước

- Tải mô hình YOLOv8:
    - Sử dụng mô hình được huấn luyện trước (pre-trained), ví dụ YOLOv8n (nano):

python

from ultralytics import YOLO

model = YOLO("yolov8n.pt") # Tải mô hình nano

- Chạy suy luận trên ảnh hoặc video:
    - Suy luận trên một ảnh từ URL:

bash

!yolo predict model=yolov8n.pt source="https://ultralytics.com/images/zidane.jpg"

- Hoặc trên tệp cục bộ/video/webcam (lưu ý: webcam không được hỗ trợ trực tiếp trên Colab):

python

results = model.predict(source="/content/sample_image.jpg", save=True)

results[0].show() # Hiển thị kết quả

- Kết quả (ảnh/video với bounding box) được lưu trong thư mục runs/detect/predict.

3.3. Huấn luyện YOLOv8 trên tập dữ liệu tùy chỉnh

- Chuẩn bị tập dữ liệu:
  - Tập dữ liệu cần ở định dạng YOLOv8, bao gồm:
    - Thư mục images chứa ảnh (train/val/test).
    - Thư mục labels chứa tệp .txt với thông tin bounding box (định dạng: class_id x_center y_center width height).
    - Tệp data.yaml định nghĩa cấu trúc dữ liệu, ví dụ:

yaml

path: /content/drive/MyDrive/Colab Notebooks/dataset

train: images/train

val: images/val

names:

 0: person

 1: car

  - Có thể sử dụng Roboflow hoặc Kaggle để tạo/tải tập dữ liệu tương thích YOLOv8.
- Tải tập dữ liệu lên Google Drive:
  - Tải thư mục dữ liệu lên Google Drive, ví dụ: /content/drive/MyDrive/Colab Notebooks/dataset.
- Huấn luyện mô hình:
  - Chạy lệnh huấn luyện với CLI:

bash

!yolo train model=yolov8n.pt data=/content/drive/MyDrive/Colab Notebooks/dataset/data.yaml epochs=50 imgsz=640 batch=16

  - Hoặc sử dụng Python API:

python

from ultralytics import YOLO

model = YOLO("yolov8n.pt")

model.train(data="/content/drive/MyDrive/Colab Notebooks/dataset/data.yaml", epochs=50, imgsz=640, batch=16)

- Các tham số quan trọng:
  - epochs: Số vòng lặp huấn luyện.
  - imgsz: Kích thước ảnh đầu vào (mặc định 640x640).
  - batch: Kích thước batch (phụ thuộc VRAM GPU, thường 16 cho T4).
- Kết quả huấn luyện (mô hình best.pt, last.pt) được lưu trong runs/detect/train.

- Đánh giá mô hình:
  - Đánh giá trên tập kiểm tra:

bash

!yolo val model=runs/detect/train/weights/best.pt data=data.yaml

- Kết quả bao gồm các chỉ số như mAP (mean Average Precision), precision, recall, và confusion matrix.

- Lưu và tải lại mô hình:
  - Mô hình được lưu tự động trong runs/detect/train/weights.
  - Để tiếp tục huấn luyện:

python

model = YOLO("runs/detect/train/weights/best.pt")

model.train(data="data.yaml", epochs=10) # Huấn luyện thêm 10 epochs

- Sao chép mô hình vào Google Drive để tránh mất dữ liệu khi phiên Colab kết thúc:

bash

!cp runs/detect/train/weights/best.pt /content/drive/MyDrive/Colab Notebooks/models/

PHỤ LỤC 4: ĐỘNG CƠ BƯỚC NEMA 57 (NEMA 23)

1. Tổng quan về động cơ bước NEMA 57 (NEMA 23)

Động cơ bước NEMA 57, thường được gọi là NEMA 23, là một loại động cơ bước (stepper motor) có kích thước mặt trước là 57 x 57 mm (2.3 x 2.3 inch). Đây là một trong những kích thước động cơ bước phổ biến nhất trong dòng NEMA, được sử dụng rộng rãi trong các ứng dụng yêu cầu độ chính xác cao và mô-men xoắn (torque) mạnh, chẳng hạn như máy in 3D, máy CNC, robot, và các hệ thống tự động hóa. Tên gọi "NEMA 57" xuất phát từ tiêu chuẩn của Hiệp hội Các nhà sản xuất Điện Quốc gia (National Electrical Manufacturers Association), trong đó số 57 biểu thị kích thước mặt trước tính bằng milimet.

NEMA 23 là động cơ bước lai (hybrid stepper motor), kết hợp giữa động cơ nam châm vĩnh cửu (permanent magnet) và động cơ từ trở (variable reluctance), mang lại hiệu suất cao với khả năng điều khiển vị trí chính xác mà không cần hệ thống phản hồi (feedback) phức tạp. Động cơ này thường có góc bước (step angle) là 1.8° (200 bước/vòng) hoặc 0.9° (400 bước/vòng) cho độ chính xác cao hơn, với mô-men xoắn từ 0.4 Nm đến 3.0 Nm (55–425 oz-in), tùy thuộc vào model cụ thể.

NEMA 23 có thể là loại bipolar (4 dây hoặc 8 dây) hoặc unipolar (6 dây), với dòng điện định mức từ 0.44A đến 4.2A và điện áp từ 1.6V đến 3.6V, tùy thuộc vào cấu hình. Động cơ này thường được điều khiển bằng các driver bước (stepper motor driver) như TB6600, DM542, hoặc High-Power Stepper Motor Driver 36v4 từ Pololu, và có thể được lập trình thông qua các phần mềm như Arduino IDE hoặc VSCode.

2. Các tính năng chính của động cơ bước NEMA 57 (NEMA 23)

NEMA 23 được thiết kế với nhiều tính năng nổi bật, phù hợp cho các ứng dụng công nghiệp và sáng tạo:

- Mô-men xoắn cao: Cung cấp mô-men xoắn từ 0.4 Nm đến 3.0 Nm (55–425 oz-in), lý tưởng cho các ứng dụng yêu cầu lực mạnh như máy CNC hoặc máy in 3D.
- Góc bước chính xác: Góc bước 1.8° (200 bước/vòng) hoặc 0.9° (400 bước/vòng) cho phép điều khiển vị trí chính xác với sai số ±5%.
- Tùy chọn cấu hình dây: Hỗ trợ cấu hình bipolar (4 hoặc 8 dây) hoặc unipolar (6 dây), cho phép kết nối linh hoạt với các driver khác nhau.
- Kích thước tiêu chuẩn: Kích thước mặt trước 57 x 57 mm và chiều dài thân từ 41 mm đến 114 mm, tương thích với nhiều hệ thống lắp đặt.
- Độ bền cao: Nhiệt độ hoạt động từ -20°C đến 50°C, khả năng chịu nhiệt độ tăng tối đa 80°C, và điện trở cách điện 100MΩ (min) tại 500V DC.
- Tùy chỉnh linh hoạt: Có thể tùy chỉnh trục (D-shaft, trục vít, hoặc trục kép), độ dài dây, hoặc tích hợp bộ mã hóa (encoder) và hộp số (gearbox).
- Hiệu suất thấp tiếng ồn và rung: Sử dụng nam châm đất hiếm (rare earth magnets) để giảm rung và tiếng ồn, cải thiện hiệu quả vận hành.

3. Cách sử dụng động cơ bước NEMA 57 với Arduino IDE và VSCode

NEMA 23 thường được điều khiển bằng các driver bước và lập trình thông qua các phần mềm như Arduino IDE hoặc VSCode. Dưới đây là hướng dẫn chi tiết cách sử dụng động cơ này với hai môi trường lập trình.

3.1. Sử dụng với Arduino IDE

- Chuẩn bị phần cứng:
    - Động cơ NEMA 23: Chọn model phù hợp, ví dụ: 23HS22-2804S (1.26 Nm, 2.8A, 4 dây).
    - Driver bước: Sử dụng driver như TB6600 hoặc A4988, đảm bảo dòng điện và điện áp tương thích.
    - Bo mạch Arduino: Arduino Uno hoặc Mega thường được sử dụng.
    - Nguồn điện: Nguồn DC 24–48V cho NEMA 23 (khuyến nghị).
- Kết nối phần cứng:
    - Kết nối 4 dây của động cơ (đen, xanh lá, đỏ, xanh dương) với driver theo đúng cặp cuộn dây (black-green, red-blue).
    - Nối driver với Arduino: Ví dụ, chân STEP và DIR của driver TB6600 với các chân digital của Arduino (như chân 8 và 9).
    - Kết nối nguồn điện với driver.
- Lập trình trong Arduino IDE:
    - Cài đặt thư viện AccelStepper hoặc Stepper để điều khiển động cơ bước:

bash

```
#include <AccelStepper.h>
// Định nghĩa driver và chân
AccelStepper stepper(AccelStepper::DRIVER, 8, 9); // STEP: 8, DIR: 9
void setup() {
  stepper.setMaxSpeed(1000); // Tốc độ tối đa
  stepper.setAcceleration(500); // Gia tốc
}
void loop() {
  stepper.moveTo(200); // Quay 200 bước (1 vòng với góc 1.8°)
  stepper.runToPosition(); // Chạy đến vị trí
  delay(1000);
  stepper.moveTo(0); // Quay về vị trí ban đầu
  stepper.runToPosition();
  delay(1000);
}
```

    - Biên dịch và nạp mã vào Arduino qua Arduino IDE.
- Kiểm tra và tinh chỉnh:
    - Sử dụng Serial Monitor để theo dõi trạng thái động cơ.
    - Điều chỉnh microstepping trên driver để tăng độ mịn (ví dụ: 1/16 hoặc 1/32 bước).

3.2. Sử dụng với Visual Studio Code (VSCode)

- Cài đặt môi trường:
    - Cài đặt VSCode và phần mở rộng PlatformIO hoặc Arduino để lập trình Arduino.
    - Cấu hình dự án Arduino trong VSCode, thêm thư viện AccelStepper qua PlatformIO.
- Viết và nạp mã:
    - Tạo dự án mới trong PlatformIO, chọn bo mạch (Arduino Uno/Mega).
    - Sử dụng mã tương tự như trong Arduino IDE, nhưng tận dụng tính năng IntelliSense của VSCode để gợi ý mã và gỡ lỗi.
    - Nạp mã qua cổng USB bằng lệnh:

bash

pio run --target upload

- Ưu điểm của VSCode:
    - Hỗ trợ gỡ lỗi (debugging) và tích hợp Git.
    - Quản lý dự án phức tạp hơn Arduino IDE, phù hợp với các hệ thống lớn.

3.3. Tích hợp với YOLOv5/YOLOv8 (nếu cần)

Trong các dự án kết hợp thị giác máy tính (ví dụ: robot tự hành sử dụng YOLOv8 để phát hiện đối tượng và NEMA 23 để di chuyển), bạn có thể:

- Sử dụng YOLOv8 trên Google Colab để huấn luyện mô hình phát hiện đối tượng (như trong Phụ lục 4).
- Kết nối đầu ra của YOLOv8 (vị trí đối tượng) với Arduino/VSCode để điều khiển động cơ NEMA 23 di chuyển đến vị trí mong muốn.
- Ví dụ: Phát hiện một đối tượng bằng YOLOv8, tính toán góc quay cần thiết, và gửi lệnh đến Arduino để điều khiển NEMA 23.

4. Ưu điểm của động cơ bước NEMA 57 (NEMA 23)

- Mô-men xoắn cao: Phù hợp cho các ứng dụng công nghiệp như CNC, in 3D, hoặc robot.
- Độ chính xác cao: Góc bước 1.8° hoặc 0.9° đảm bảo điều khiển vị trí chính xác mà không cần bộ mã hóa.
- Tương thích đa dạng: Hoạt động với nhiều driver và bo mạch điều khiển như Arduino, Raspberry Pi, hoặc PLC.
- Tùy chỉnh linh hoạt: Có thể thêm trục kép, bộ mã hóa, hoặc hộp số để đáp ứng nhu cầu cụ thể.
- Độ bền và đáng tin cậy: Hoạt động ổn định trong môi trường khắc nghiệt (-20°C đến 50°C).
- Chi phí hợp lý: Giá thành cạnh tranh, đặc biệt với các model từ các nhà cung cấp như STEPPERONLINE hoặc Pololu.

5. Hạn chế của động cơ bước NEMA 57 (NEMA 23)

- Rung và tiếng ồn: Ở chế độ toàn bước (full-step), động cơ có thể rung mạnh, cần microstepping để giảm rung.
- Hiệu suất giảm ở tốc độ cao: NEMA 23 hoạt động tốt nhất ở tốc độ dưới 1000 RPM (1000–3000 PPS tại 0.9°).
- Yêu cầu driver phù hợp: Cần driver mạnh để cung cấp dòng điện và điện áp chính xác, đặc biệt với các model có dòng cao (4.2A).
- Không phù hợp với tải quán tính lớn: Cần hộp số hoặc động cơ kích thước lớn hơn (NEMA 34) cho các ứng dụng có quán tính cao.

6. Ứng dụng thực tế của động cơ bước NEMA 57 (NEMA 23)

NEMA 23 được sử dụng trong nhiều lĩnh vực nhờ mô-men xoắn cao và độ chính xác:

- Máy in 3D: Điều khiển chuyển động của đầu in và bàn in.
- Máy CNC: Điều khiển trục X, Y, Z trong các máy phay, cắt laser, hoặc plasma.
- Robot: Sử dụng trong các cánh tay robot hoặc xe tự hành để điều khiển chuyển động chính xác.
- Tự động hóa công nghiệp: Ứng dụng trong máy gắp và đặt (pick-and-place), băng chuyền, hoặc máy đóng gói.
- Thiết bị y tế: Điều khiển vị trí trong các thiết bị như máy quét CT hoặc robot phẫu thuật.
- Hệ thống IoT: Kết hợp với YOLOv5/YOLOv8 trong các dự án như camera thông minh hoặc robot giám sát.

7. Tài nguyên và hỗ trợ:

- Nhà cung cấp: STEPPERONLINE (www.omc-stepperonline.com) (www.omc-stepperonline.com), Pololu (www.pololu.com) (www.pololu.com), Oyostepper (www.oyostepper.com) (www.oyostepper.com), và Motion Control Products (www.motioncontrolproducts.co.uk) (www.motioncontrolproducts.co.uk) cung cấp động cơ NEMA 23 với nhiều tùy chọn.
- Tài liệu kỹ thuật: Datasheet từ các nhà sản xuất như Pololu hoặc Lin Engineering cung cấp thông số chi tiết.
- Cộng đồng: Diễn đàn Arduino, Reddit, hoặc Stack Overflow hỗ trợ lập trình và điều khiển NEMA 23.
- Hướng dẫn trực tuyến: Các video trên YouTube và bài viết trên Adafruit (learn.adafruit.com) hướng dẫn cách kết nối và lập trình.

8. Kết luận:

Động cơ bước NEMA 57 (NEMA 23) là một giải pháp mạnh mẽ và linh hoạt cho các ứng dụng yêu cầu điều khiển vị trí chính xác và mô-men xoắn cao. Với khả năng tích hợp dễ dàng với Arduino IDE, VSCode, và các driver bước phổ biến, NEMA 23 phù hợp cho cả người mới bắt đầu và các kỹ sư chuyên nghiệp. Trong các dự án kết hợp thị giác máy tính, NEMA 23 có thể được sử dụng cùng YOLOv5/YOLOv8 để tạo ra các hệ thống thông minh như robot hoặc máy tự động hóa. Mặc dù có một số hạn chế về rung và hiệu suất ở tốc độ cao, NEMA 23 vẫn là lựa chọn hàng đầu trong nhiều ứng dụng công nghiệp và sáng tạo nhờ độ bền, chi phí hợp lý, và tính tương thích cao.

PHỤ LỤC 5: ĐỘNG CƠ MY1016Z2-350W

1. Tổng quan về động cơ MY1016Z2-350W

Động cơ MY1016Z2-350W là một loại động cơ DC chổi than (brushed DC motor) được thiết kế với cơ chế giảm tốc (geared motor), thường được sử dụng trong các ứng dụng xe điện như xe đạp điện, xe scooter, xe điện trẻ em, và các dự án tự chế (DIY) như robot hoặc xe tự hành. Động cơ này thuộc dòng MY1016Z của các nhà sản xuất Trung Quốc, chẳng hạn như Changzhou Lunyee Electromechanical Manufacturing Co., Ltd., và được phân phối rộng rãi trên các nền tảng như Amazon, Shopee, và các nhà cung cấp linh kiện điện tử như Zbotic hoặc KitsGuru.

MY1016Z2-350W hoạt động ở điện áp định mức 24V (hoặc đôi khi 36V tùy biến thể), với công suất 350W, tốc độ quay khoảng 330–370 RPM sau giảm tốc, và mô-men xoắn cao, phù hợp cho các ứng dụng yêu cầu lực kéo mạnh ở tốc độ thấp. Động cơ được tích hợp hộp số giảm tốc (gearbox) với tỷ lệ giảm tốc khoảng 23.2:1, sử dụng bánh răng xích (#410, thường 9 hoặc 16 răng) để truyền lực, giúp tăng mô-men xoắn và giảm tốc độ quay. Động cơ này nổi bật với thiết kế nhỏ gọn, độ bền cao, và chi phí hợp lý, khiến nó trở thành lựa chọn phổ biến cho các dự án xe điện và robot.

So với động cơ bước NEMA 23 (được đề cập trong Phụ lục 5), MY1016Z2-350W không cung cấp khả năng điều khiển vị trí chính xác như động cơ bước, nhưng bù lại, nó có công suất lớn hơn và phù hợp cho các ứng dụng yêu cầu chuyển động liên tục với lực mạnh, chẳng hạn như di chuyển xe hoặc tải nặng.

2. Các tính năng chính của động cơ MY1016Z2-350W

MY1016Z2-350W được thiết kế với nhiều tính năng nổi bật, phù hợp cho các ứng dụng xe điện và dự án DIY:

- Công suất mạnh mẽ: Công suất định mức 350W, cung cấp mô-men xoắn cao (khoảng 0.98–1.93 Nm tùy cấu hình), lý tưởng cho xe đạp điện, scooter, hoặc robot tải trọng 180–220 kg.
- Tốc độ giảm tốc: Tốc độ định mức khoảng 330 RPM (sau giảm tốc từ 2400–3700 RPM không tải), phù hợp cho các ứng dụng yêu cầu lực kéo lớn ở tốc độ thấp.
- Hộp số tích hợp: Tỷ lệ giảm tốc 23.2:1, sử dụng bánh răng xích #410 (9 hoặc 16 răng), tăng mô-men xoắn và đảm bảo truyền lực hiệu quả.
- Điện áp linh hoạt: Hoạt động ở 24V hoặc 36V (tùy model), với dòng định mức khoảng 13.7–14.8A, hiệu suất động cơ đạt 70–76%.
- Thiết kế bền bỉ: Sử dụng 100% dây đồng, vỏ thép chắc chắn, và bảo vệ thân máy loại kín (enclosed type), phù hợp cho các môi trường khắc nghiệt như xe địa hình (ATV) hoặc robot ngoài trời.
- Tùy chỉnh linh hoạt: Hỗ trợ các tùy chọn như trục dài (55 mm), bánh xe răng tự do (freewheel), hoặc đai ròng rọc (belt pulley), phù hợp cho nhiều ứng dụng khác nhau.
- Tiếng ồn thấp: Động cơ hoạt động êm ái, giảm rung và tiếng ồn nhờ thiết kế bánh răng xoắn (helical gear) và nam châm vĩnh cửu (PMDC).
- Dễ lắp đặt: Kích thước nhỏ gọn (đường kính trục 17 mm, chiều dài tổng thể khoảng 100–150 mm), dễ tích hợp vào xe đạp, scooter, hoặc robot.

3. Cách sử dụng động cơ MY1016Z2-350W với Arduino IDE và VSCode

Động cơ MY1016Z2-350W là động cơ DC chổi than, thường được điều khiển bằng bộ điều khiển tốc độ (motor controller) như IC điều tốc (ví dụ: IC 24V 500W hoặc bộ điều khiển PWM) thay vì driver bước như NEMA 23. Dưới đây là hướng dẫn chi tiết cách sử dụng động cơ này với Arduino IDE và VSCode, cùng cách tích hợp với YOLOv5/YOLOv8 nếu cần.

3.1. Sử dụng với Arduino IDE

- Chuẩn bị phần cứng:
    - Động cơ MY1016Z2-350W: Điện áp 24V, công suất 350W, bánh răng xích 9 răng (#410).
    - Bộ điều khiển tốc độ: Sử dụng IC điều tốc 24V 500W hoặc module PWM như L298N (cho thử nghiệm công suất thấp) hoặc ESC (Electronic Speed Controller) cho xe điện.
    - Bo mạch Arduino: Arduino Uno hoặc Mega, với nguồn 24V DC (pin hoặc adapter).
    - Phụ kiện: Chân ga, công tắc đảo chiều (nếu cần), và dây đấu nối.
- Kết nối phần cứng:
    - Kết nối động cơ với bộ điều khiển tốc độ: Động cơ MY1016Z2-350W có 2 dây (đỏ và đen) nối với đầu ra của IC điều tốc.
    - Nối IC điều tốc với Arduino: Chân PWM (như chân 9) điều khiển tốc độ, chân digital (như chân 8) điều khiển chiều quay (nếu có đảo chiều).
    - Kết nối chân ga (nếu sử dụng): Dây đỏ (5V), đen (GND), và xanh (tín hiệu) của chân ga nối với IC điều tốc.
    - Kết nối nguồn 24V với IC điều tốc và GND chung với Arduino.
- Lập trình trong Arduino IDE:
    - Sử dụng PWM để điều khiển tốc độ động cơ:

cpp

```
const int pwmPin = 9; // Chân PWM điều khiển tốc độ
const int dirPin = 8; // Chân điều khiển chiều (nếu có)

void setup() {
  pinMode(pwmPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  analogWrite(pwmPin, 0); // Tốc độ ban đầu = 0
  digitalWrite(dirPin, HIGH); // Chiều quay thuận
}

void loop() {
  // Tăng tốc độ từ 0 đến 100%
  for (int speed = 0; speed <= 255; speed += 5) {
```

```
  analogWrite(pwmPin, speed);
  delay(100);
 }
 delay(2000);
 // Giảm tốc độ về 0
 for (int speed = 255; speed >= 0; speed -= 5) {
  analogWrite(pwmPin, speed);
  delay(100);
 }
 delay(2000);
}
```

- Biên dịch và nạp mã qua Arduino IDE.

- Kiểm tra và tinh chỉnh:
  - Sử dụng Serial Monitor để theo dõi tốc độ và trạng thái.
  - Điều chỉnh giá trị PWM (0–255) để đạt tốc độ mong muốn.
  - Nếu sử dụng chân ga, đọc tín hiệu analog từ chân ga (A0) để điều khiển tốc độ:

cpp

```
int speed = map(analogRead(A0), 0, 1023, 0, 255);
analogWrite(pwmPin, speed);
```

3.2. Sử dụng với Visual Studio Code (VSCode)

- Cài đặt môi trường:
  - Cài đặt VSCode và phần mở rộng PlatformIO hoặc Arduino.
  - Tạo dự án Arduino trong PlatformIO, chọn bo mạch (Arduino Uno/Mega).
- Viết và nạp mã:
  - Sử dụng mã tương tự như trong Arduino IDE, tận dụng IntelliSense và gỡ lỗi của VSCode.
  - Nạp mã qua cổng USB bằng lệnh:

bash

```
pio run --target upload
```

- Ưu điểm của VSCode:
  - Hỗ trợ quản lý dự án lớn, tích hợp Git, và gỡ lỗi.
  - Phù hợp cho các hệ thống phức tạp kết hợp nhiều thành phần (ví dụ: động cơ và YOLOv8).

3.3. Tích hợp với YOLOv5/YOLOv8 (nếu cần)

Trong các dự án như robot tự hành hoặc xe điện thông minh, MY1016Z2-350W có thể được điều khiển dựa trên đầu ra từ YOLOv5/YOLOv8 (được huấn luyện trên Google Colab, như trong Phụ lục 3 và 4):

- Quy trình:
    - Sử dụng YOLOv8 trên Colab để phát hiện đối tượng (ví dụ: người, xe, chướng ngại vật).
    - Truyền tọa độ hoặc lệnh (qua giao tiếp UART hoặc Wi-Fi) từ máy tính/Raspberry Pi chạy YOLOv8 đến Arduino.
    - Arduino điều khiển MY1016Z2-350W để di chuyển, dừng, hoặc đổi hướng dựa trên lệnh.
- Ví dụ: Nếu YOLOv8 phát hiện chướng ngại vật ở phía trước, gửi lệnh "STOP" đến Arduino, Arduino đặt PWM = 0 để dừng động cơ.
- Mã mẫu (Arduino nhận lệnh từ Serial):

cpp

```
if (Serial.available() > 0) {
  String command = Serial.readStringUntil('\n');
  if (command == "STOP") {
    analogWrite(pwmPin, 0);
  } else if (command == "FORWARD") {
    analogWrite(pwmPin, 255);
    digitalWrite(dirPin, HIGH);
  }
}
```

4. Ưu điểm của động cơ MY1016Z2-350W

- Công suất mạnh mẽ: 350W, phù hợp cho xe điện tải trọng 180–220 kg hoặc robot nặng.
- Mô-men xoắn cao: Hộp số giảm tốc tăng lực kéo, lý tưởng cho xe địa hình, scooter, hoặc xe kéo mini.
- Chi phí hợp lý: Giá thành từ 900.000–1.500.000 VNĐ, rẻ hơn nhiều so với động cơ BLDC cùng công suất.
- Dễ lắp đặt: Thiết kế nhỏ gọn, bánh răng xích tiêu chuẩn, dễ tích hợp vào xe đạp, scooter, hoặc robot.
- Độ bền cao: Vỏ thép, dây đồng 100%, và bảo vệ kín, chịu được môi trường khắc nghiệt.
- Ứng dụng đa dạng: Phù hợp cho xe điện, robot, máy tời mini, hoặc máy phát điện tự chế.

5. Hạn chế của động cơ MY1016Z2-350W

- Chổi than hao mòn: Là động cơ DC chổi than, chổi than cần thay định kỳ, giảm tuổi thọ so với động cơ không chổi than (BLDC).

- Tiếng ồn và rung: Mặc dù được tối ưu, động cơ vẫn có tiếng ồn từ hộp số và chổi than, không êm ái như BLDC.
- Hiệu suất trung bình: Hiệu suất 70–76%, thấp hơn so với động cơ BLDC (thường >85%).
- Không điều khiển vị trí chính xác: Không phù hợp cho các ứng dụng cần điều khiển vị trí như động cơ bước NEMA 23.
- Yêu cầu bộ điều khiển phù hợp: Cần IC điều tốc hoặc ESC mạnh để xử lý dòng 13.7–14.8A, tăng chi phí hệ thống.
- Bảo hành ngắn: Một số nhà cung cấp chỉ bảo hành 15 ngày cho lỗi sản xuất, cần kiểm tra kỹ trước khi sử dụng.

6. Ứng dụng thực tế của động cơ MY1016Z2-350W

MY1016Z2-350W được sử dụng rộng rãi trong nhiều lĩnh vực nhờ công suất mạnh và chi phí thấp:

- Xe đạp điện và scooter: Nâng cấp hoặc chế tạo xe đạp điện, scooter với tốc độ 25–30 km/h, quãng đường 15–17 km (với pin 7.5Ah).
- Xe điện trẻ em: Điều khiển ô tô điện, xe máy đồ chơi với tải trọng lên đến 100 kg.
- Robot và xe tự hành: Sử dụng trong robot chiến đấu (combat robots), robot ATV, hoặc xe tự hành kết hợp với YOLOv5/YOLOv8.
- Dự án DIY: Máy tời mini, máy phát điện tự chế, hoặc xe kéo nhỏ.
- Công nghiệp nhẹ: Điều khiển băng chuyền nhỏ hoặc thiết bị vận chuyển trong xưởng sản xuất.
- Ứng dụng IoT: Kết hợp với Arduino/Raspberry Pi và YOLOv8 để tạo xe điện thông minh hoặc robot giám sát.

7. Tài nguyên và hỗ trợ

- Nhà cung cấp: Các nền tảng như Shopee (shopee.vn/motor2hand), Amazon (www.amazon.com) (www.amazon.com), Zbotic (zbotic.in), và KitsGuru (kitsguru.com) cung cấp MY1016Z2-350W với giá cạnh tranh.
- Tài liệu kỹ thuật: Datasheet từ nhà sản xuất (như Changzhou Lunyee) hoặc tài liệu từ Motor2Hand (www.motor2hand.com) (www.motor2hand.com) cung cấp thông số chi tiết.
- Cộng đồng: Diễn đàn Arduino, nhóm Facebook Motor2Hand (www.facebook.com/motor2handhn) (www.facebook.com/motor2handhn), hoặc phutungxedapdien.net hỗ trợ về lắp đặt và điều khiển.
- Hướng dẫn trực tuyến: Video YouTube từ Motor2Hand hoặc bài viết trên xedienmotordien.com hướng dẫn chế xe điện với MY1016Z2-350W.

PHỤ LỤC 6: BỘ ĐIỀU KHIỂN ĐỘNG CƠ BTS7960

1. Tổng quan về bộ điều khiển động cơ BTS7960

- BTS7960 là một bộ điều khiển động cơ H-cầu (H-bridge motor driver) công suất cao, được thiết kế bởi Infineon Technologies, một công ty bán dẫn hàng đầu có trụ sở tại Đức. Module BTS7960 sử dụng chip BTS7960 (thuộc dòng NovalithIC™) bao gồm một MOSFET kênh-p phía cao (p-channel high-side) và một MOSFET kênh-n phía thấp (n-channel low-side), tích hợp với một mạch điều khiển (driver IC) trong một gói duy nhất. Module này thường được cấu hình với hai chip BTS7960 để tạo thành một H-cầu hoàn chỉnh, cho phép điều khiển động cơ DC công suất lớn, như động cơ MY1016Z2-350W (đã đề cập trong Phụ lục 6), với dòng điện lên đến 43A và điện áp từ 6V đến 27V.
- BTS7960 được thiết kế để điều khiển các động cơ DC, động cơ bước (stepper motor), hoặc động cơ BLDC ba pha (3-phase BLDC) trong các ứng dụng yêu cầu dòng cao, chẳng hạn như xe điện, robot, máy CNC, hoặc các dự án tự động hóa. Module này hỗ trợ điều chế độ rộng xung (PWM) với tần số lên đến 25 kHz, cho phép điều khiển tốc độ và hướng quay của động cơ một cách mượt mà. Ngoài ra, BTS7960 tích hợp các tính năng bảo vệ như chống quá nhiệt, quá áp, thấp áp, quá dòng, và ngắn mạch, đảm bảo an toàn cho cả động cơ và vi điều khiển.
- BTS7960 thường được sử dụng với các vi điều khiển như Arduino, ESP32, hoặc Raspberry Pi, và có thể được lập trình thông qua Arduino IDE hoặc VSCode. Trong các dự án phức tạp, BTS7960 có thể được tích hợp với các hệ thống thị giác máy tính như YOLOv5/YOLOv8 (được huấn luyện trên Google Colab) để điều khiển động cơ dựa trên kết quả phát hiện đối tượng.

2. Các tính năng chính của BTS7960

- BTS7960 nổi bật với nhiều tính năng tiên tiến, phù hợp cho các ứng dụng điều khiển động cơ công suất cao:
- Dòng điện cao: Hỗ trợ dòng điện đỉnh 43A (với tản nhiệt phù hợp), phù hợp cho động cơ như MY1016Z2-350W hoặc các động cơ DC khác.
- Điện áp rộng: Hoạt động từ 6V đến 27V, tương thích với nguồn 12V hoặc 24V thường dùng trong xe điện và robot.
- H-cầu tích hợp: Sử dụng hai chip BTS7960 để tạo H-cầu, cho phép điều khiển cả tốc độ và hướng quay (thuận/ngược).
- PWM tần số cao: Hỗ trợ PWM lên đến 25 kHz, đảm bảo điều khiển tốc độ mượt mà và giảm tiếng ồn động cơ.
- Bảo vệ toàn diện: Bao gồm bảo vệ chống quá nhiệt, quá áp, thấp áp (tắt tại 5.4V, bật lại tại 5.5V), quá dòng, và ngắn mạch, tăng độ an toàn và tuổi thọ.
- Tích hợp cảm biến dòng: Cung cấp tín hiệu chẩn đoán dòng (current sense) qua các chân R_IS và L_IS, giúp giám sát trạng thái động cơ.
- Giao tiếp vi điều khiển dễ dàng: Hỗ trợ mức logic 3.3V–5V, tương thích với Arduino, ESP32, hoặc Raspberry Pi.
- Tản nhiệt hiệu quả: Module đi kèm tản nhiệt lớn, nhưng cần bổ sung tản nhiệt phụ khi hoạt động ở dòng trên 25A liên tục.

- Cách ly tín hiệu: Sử dụng chip 74HC244 để cách ly vi điều khiển và động cơ, giảm nhiễu điện từ (EMI).
- Kích thước nhỏ gọn: Kích thước khoảng 50 x 50 x 43 mm, dễ tích hợp vào các dự án robot hoặc xe điện.

3. Cách sử dụng BTS7960 với Arduino IDE, VSCode, và MY1016Z2-350W

- BTS7960 được sử dụng để điều khiển các động cơ DC công suất cao như MY1016Z2-350W. Dưới đây là hướng dẫn chi tiết cách sử dụng module này với Arduino IDE và VSCode, cùng cách tích hợp với YOLOv5/YOLOv8 nếu cần.

PHỤ LỤC 7: BỘ ĐIỀU KHIỂN ĐỘNG CƠ BTS7960

1. Tổng quan về bộ điều khiển động cơ BTS7960

- BTS7960 là một bộ điều khiển động cơ H-cầu (H-bridge motor driver) công suất cao, được thiết kế bởi Infineon Technologies, một công ty bán dẫn hàng đầu có trụ sở tại Đức. Module BTS7960 sử dụng chip BTS7960 (thuộc dòng NovalithIC™) bao gồm một MOSFET kênh-p phía cao (p-channel high-side) và một MOSFET kênh-n phía thấp (n-channel low-side), tích hợp với một mạch điều khiển (driver IC) trong một gói duy nhất. Module này thường được cấu hình với hai chip BTS7960 để tạo thành một H-cầu hoàn chỉnh, cho phép điều khiển động cơ DC công suất lớn, như động cơ MY1016Z2-350W (đã đề cập trong Phụ lục 6), với dòng điện lên đến 43A và điện áp từ 6V đến 27V.
- BTS7960 được thiết kế để điều khiển các động cơ DC, động cơ bước (stepper motor), hoặc động cơ BLDC ba pha (3-phase BLDC) trong các ứng dụng yêu cầu dòng cao, chẳng hạn như xe điện, robot, máy CNC, hoặc các dự án tự động hóa. Module này hỗ trợ điều chế độ rộng xung (PWM) với tần số lên đến 25 kHz, cho phép điều khiển tốc độ và hướng quay của động cơ một cách mượt mà. Ngoài ra, BTS7960 tích hợp các tính năng bảo vệ như chống quá nhiệt, quá áp, thấp áp, quá dòng, và ngắn mạch, đảm bảo an toàn cho cả động cơ và vi điều khiển.
- BTS7960 thường được sử dụng với các vi điều khiển như Arduino, ESP32, hoặc Raspberry Pi, và có thể được lập trình thông qua Arduino IDE hoặc VSCode. Trong các dự án phức tạp, BTS7960 có thể được tích hợp với các hệ thống thị giác máy tính như YOLOv5/YOLOv8 (được huấn luyện trên Google Colab) để điều khiển động cơ dựa trên kết quả phát hiện đối tượng.

2. Các tính năng chính của BTS7960

- BTS7960 nổi bật với nhiều tính năng tiên tiến, phù hợp cho các ứng dụng điều khiển động cơ công suất cao:
- Dòng điện cao: Hỗ trợ dòng điện đỉnh 43A (với tản nhiệt phù hợp), phù hợp cho động cơ như MY1016Z2-350W hoặc các động cơ DC khác.
- Điện áp rộng: Hoạt động từ 6V đến 27V, tương thích với nguồn 12V hoặc 24V thường dùng trong xe điện và robot.
- H-cầu tích hợp: Sử dụng hai chip BTS7960 để tạo H-cầu, cho phép điều khiển cả tốc độ và hướng quay (thuận/ngược).
- PWM tần số cao: Hỗ trợ PWM lên đến 25 kHz, đảm bảo điều khiển tốc độ mượt mà và giảm tiếng ồn động cơ.
- Bảo vệ toàn diện: Bao gồm bảo vệ chống quá nhiệt, quá áp, thấp áp (tắt tại 5.4V, bật lại tại 5.5V), quá dòng, và ngắn mạch, tăng độ an toàn và tuổi thọ.
- Tích hợp cảm biến dòng: Cung cấp tín hiệu chẩn đoán dòng (current sense) qua các chân R_IS và L_IS, giúp giám sát trạng thái động cơ.
- Giao tiếp vi điều khiển dễ dàng: Hỗ trợ mức logic 3.3V–5V, tương thích với Arduino, ESP32, hoặc Raspberry Pi.
- Tản nhiệt hiệu quả: Module đi kèm tản nhiệt lớn, nhưng cần bổ sung tản nhiệt phụ khi hoạt động ở dòng trên 25A liên tục.

- Cách ly tín hiệu: Sử dụng chip 74HC244 để cách ly vi điều khiển và động cơ, giảm nhiễu điện từ (EMI).
- Kích thước nhỏ gọn: Kích thước khoảng 50 x 50 x 43 mm, dễ tích hợp vào các dự án robot hoặc xe điện.

3. Cách sử dụng BTS7960 với Arduino IDE, VSCode, và MY1016Z2-350W

- BTS7960 được sử dụng để điều khiển các động cơ DC công suất cao như MY1016Z2-350W. Dưới đây là hướng dẫn chi tiết cách sử dụng module này với Arduino IDE và VSCode, cùng cách tích hợp với YOLOv5/YOLOv8 nếu cần.

3.1. Sử dụng với Arduino IDE

- Chuẩn bị phần cứng:
- Module BTS7960: Module H-cầu đôi với dòng tối đa 43A.
- Động cơ: MY1016Z2-350W (24V, 350W, 13.7A).
- Bo mạch Arduino: Arduino Uno hoặc Mega.
- Nguồn điện: Nguồn DC 24V, công suất tối thiểu 400W (đảm bảo dòng >15A).
- Dây nối: Dây chịu tải cao cho động cơ và dây tín hiệu cho Arduino.
- Kết nối phần cứng:
- Nguồn động cơ: Nối VM+ (B+) và GND (B-) của BTS7960 với nguồn 24V.
- Nguồn logic: Nối VCC (5V) và GND của BTS7960 với 5V và GND của Arduino (hoặc dùng nguồn 5V tích hợp trên module).
- Động cơ: Nối hai dây của MY1016Z2-350W với M+ và M- của BTS7960.
- Tín hiệu điều khiển:
- RPWM (PWM thuận) → Chân PWM Arduino (ví dụ: chân 5).
- LPWM (PWM ngược) → Chân PWM Arduino (ví dụ: chân 6).
- R_EN (kích hoạt thuận) → Chân digital Arduino (ví dụ: chân 7).
- L_EN (kích hoạt ngược) → Chân digital Arduino (ví dụ: chân 8).
- R_IS và L_IS (cảm biến dòng, tùy chọn) → Chân analog Arduino (ví dụ: A0, A1) nếu cần giám sát dòng.
- Sơ đồ kết nối (tham khảo từ Instructables):
- Lập trình trong Arduino IDE:
- Sử dụng PWM để điều khiển tốc độ và hướng quay của MY1016Z2-350W:
- cpp
- Sử dụng Serial Monitor để theo dõi trạng thái.
- Điều chỉnh tần số PWM (mặc định Arduino ~970Hz, có thể tăng lên ~3.92kHz trên Arduino Mega bằng cách cấu hình timer, như trong mã mẫu từ Instructables).
- Kiểm tra tản nhiệt: Nếu dòng vượt 25A, cần thêm tản nhiệt phụ trên chip BTS7960.

3.2. Sử dụng với Visual Studio Code (VSCode)

- Cài đặt môi trường:

- Cài đặt VSCode và phần mở rộng PlatformIO hoặc Arduino.
- Tạo dự án Arduino trong PlatformIO, chọn bo mạch (Arduino Uno/Mega).
- Viết và nạp mã:
- Sử dụng mã tương tự như trong Arduino IDE, tận dụng IntelliSense và gỡ lỗi của VSCode.
- Nạp mã qua cổng USB bằng lệnh:

bash

```
pio run --target upload
```

PHỤ LỤC 8: BỘ ĐIỀU KHIỂN ĐỘNG CƠ TB6560

1. Tổng quan về TB6560

- TB6560 là một bộ điều khiển động cơ bước (stepper motor driver) công suất trung bình, sử dụng chip TB6560AHQ của Toshiba, được thiết kế để điều khiển động cơ bước 2/4 pha, 4/6 dây, như NEMA 23 (đã đề cập trong Phụ lục 5). Module này hỗ trợ dòng điện tối đa 3.5A (đỉnh), điện áp 10–35V DC, và các chế độ microstepping (1/1, 1/2, 1/8, 1/16), phù hợp cho các ứng dụng như máy CNC, in 3D, hoặc robot. TB6560 tích hợp bảo vệ quá dòng, quá nhiệt, và thấp áp, nhưng không có bảo vệ đảo ngược điện áp, đòi hỏi cẩn thận khi kết nối nguồn.
- TB6560 được điều khiển qua các tín hiệu STEP, DIR, và ENABLE, tương thích với vi điều khiển như Arduino hoặc Raspberry Pi, lập trình bằng Arduino IDE hoặc VSCode. So với BTS7960 (Phụ lục 7), TB6560 dành riêng cho động cơ bước, không phù hợp cho động cơ DC như MY1016Z2-350W.

2. Tính năng chính

- Dòng điện và điện áp: Hỗ trợ dòng tối đa 3.5A, điện áp 10–35V, phù hợp cho NEMA 23 (dòng ≤3A).
- Microstepping: Chế độ 1/1, 1/2, 1/8, 1/16, tăng độ mịn và giảm rung.
- Bảo vệ: Chống quá dòng, quá nhiệt, và thấp áp, nhưng thiếu bảo vệ đảo ngược điện áp.
- Opto-coupler: Tích hợp 6N137 tốc độ cao, cách ly tín hiệu STEP/DIR, giảm nhiễu.
- Chế độ giảm dòng: Giảm dòng khi động cơ dừng, tiết kiệm năng lượng và giảm nhiệt.
- Kích thước nhỏ: 50 x 75 x 35 mm, dễ lắp đặt.
- Cài đặt linh hoạt: Sử dụng công tắc DIP để điều chỉnh dòng, microstepping, và chế độ suy giảm (decay mode).

3. Cách sử dụng với Arduino IDE và VSCode

- TB6560 được sử dụng để điều khiển động cơ bước như NEMA 23, không tương thích với động cơ DC như MY1016Z2-350W.

3.1. Sử dụng với Arduino IDE

- Chuẩn bị phần cứng:
- TB6560: Module với dòng tối đa 3.5A.
- Động cơ: NEMA 23 (≤3A, ví dụ: 23HS22-2804S, 2.8A).
- Arduino: Uno hoặc Mega.
- Nguồn: 24V DC (>100W).
- Kết nối:
- Nguồn: VCC và GND của TB6560 với nguồn 24V.
- Động cơ: Nối A+, A-, B+, B- với đầu ra TB6560 (theo datasheet động cơ).
- Tín hiệu:
- CLK+ (STEP) → Chân PWM Arduino (chân 9).
- CW+ (DIR) → Chân digital Arduino (chân 8).

- CLK-, CW- → GND Arduino.
- EN+, EN-: Để ngỏ (enable mặc định).
- Cài đặt DIP switch: Ví dụ, S1-S2 (dòng 2.8A), S3-S4 (1/8 microstepping).
- Lập trình:
- Sử dụng thư viện AccelStepper:
- cpp

```
#include <AccelStepper.h>
AccelStepper stepper(AccelStepper::DRIVER, 9, 8); // STEP: 9, DIR: 8
void setup() {
  stepper.setMaxSpeed(1000);
  stepper.setAcceleration(500);
}
void loop() {
  stepper.moveTo(1600); // 1 vòng (200 bước x 8 microsteps)
  stepper.runToPosition();
  delay(1000);
  stepper.moveTo(0);
  stepper.runToPosition();
  delay(1000);
}
```

Nạp mã qua Arduino IDE.

- Kiểm tra:
- Sử dụng Serial Monitor để theo dõi.
- Điều chỉnh microstepping và dòng qua DIP switch để giảm rung.

3.2. Sử dụng với VSCode

- Cài đặt PlatformIO trong VSCode, tạo dự án Arduino.
- Sử dụng mã tương tự như Arduino IDE, tận dụng IntelliSense và gỡ lỗi.
- Nạp mã bằng lệnh: pio run --target upload.
- VSCode phù hợp cho dự án lớn, tích hợp với YOLOv8 hoặc Git.
- 3.3. Tích hợp với YOLOv5/YOLOv8
- Trong dự án robot, YOLOv8 (huấn luyện trên Google Colab, Phụ lục 4) phát hiện đối tượng (ví dụ: mục tiêu di chuyển).
- Đầu ra YOLOv8 (tọa độ) được gửi qua UART/Wi-Fi đến Arduino, điều khiển TB6560 và NEMA 23 để xoay hoặc di chuyển.
- Ví dụ: Nếu YOLOv8 phát hiện mục tiêu, Arduino ra lệnh quay 90° (500 bước với 1/8 microstepping).

4. Ưu điểm

- Hiệu quả chi phí: Giá ~300.000–600.000 VNĐ, rẻ hơn TB6600 hoặc DM542.
- Hiệu suất tốt: Hỗ trợ microstepping, giảm rung và tiếng ồn.
- Bảo vệ tích hợp: Chống quá dòng, quá nhiệt, và thấp áp.
- Tương thích rộng: Điều khiển NEMA 23, 42, 57, 86 (dòng ≤3A).
- Dễ sử dụng: Công tắc DIP đơn giản hóa cài đặt dòng và microstepping.

5. Hạn chế

- Thiếu bảo vệ đảo ngược: Cần kiểm tra nguồn cẩn thận.
- Hiệu suất kém hơn driver số: So với DM542/DM556, TB6560 (analog driver) có tiếng ồn và rung cao hơn.
- Tản nhiệt hạn chế: Cần tản nhiệt phụ khi dòng >2.5A.
- Hạn chế microstepping: Chỉ hoạt động tốt ở 1/8 và 1/16, chế độ full/half step có thể không ổn định.
- Không phù hợp động cơ DC: Không dùng được với MY1016Z2-350W.

6. Ứng dụng thực tế

- Máy CNC: Điều khiển trục X, Y, Z.
- Máy in 3D: Di chuyển đầu in hoặc bàn in.
- Robot: Điều khiển cánh tay hoặc bánh xe với NEMA 23.
- Tự động hóa: Hệ thống băng chuyền, cửa tự động.
- IoT và thị giác máy tính: Kết hợp với YOLOv8 trong robot giám sát hoặc điều hướng.

PHỤ LỤC 9: CÁC MÃ CODE SỬ DỤNG TRONG DỰ ÁN:

- **Code Arduino IDE**

1. Thêm các thư viện và chân

```
#include <Arduino.h>
#include <AccelStepper.h>
#include <SoftwareSerial.h>
#include <PID_v1.h>
#include <Encoder.h>

#define DEBUG_MODE  // ← Chỉ cần comment dòng này lại là tắt debug
// Định nghĩa chân kết nối
int step_X = 11; int Dir_X = 12; int ena_X = 13;
int step_Y = 8; int Dir_Y = 9; int ena_Y = 10;

// Khai báo các chân Động cơ
#define PWM_A 6
#define PWM_B 7

// Định nghĩa công tắc hành trình
#define end_X  A9
#define end_Y  A8
// Khai báo các chân encoder
#define EncoderA 2
#define EncoderB 3
#define EncoderC 4
#define EncoderD 5
```

2. Khai báo, đặt các giá trị ban đầu :

```
// Cấu hình thông số xe
double Duong_Kinh_Banh_Xe = 0.19; // Đơn vị: m
int Xung_Encoder_A = 600;
int Xung_Encoder_B = 600;

// Khai báo dữ liệu cho PID
double W_Feed_Back_From_DC_A;
double PWM_Output_DC_A;
double W_Feed_Back_From_DC_B;
double PWM_Output_DC_B;

double Initial_PWM =40 ; // Giảm xung ban đầu để quay nhẹ

// Các biến set giá trị
double Set_Speed_DC_A = 0; // Đơn vị: m/s
double Set_RPM_A = 0;      // Đơn vị: vòng/phút
double Set_Speed_DC_B = 0; // Đơn vị: m/s
double Set_RPM_B = 0;      // Đơn vị: vòng/phút
double Set_Speed = 0.8;    // Đơn vị: km/h

// Biến lưu giá trị đếm encoder
volatile int var_Encoder_A = 0;
volatile int var_Encoder_B = 0;

// Hệ số điều chỉnh cho distance1
double K_DISTANCE = 0.8;  // Hệ số điều chỉnh tốc độ bánh xe dựa trên distance1

// Khai báo chân encoder
Encoder myEncoder1(EncoderA, EncoderB);
Encoder myEncoder(EncoderC, EncoderD);

double Sampling_Time_DC = 100; // Thời gian lấy mẫu 100ms
double Sampling_Time_Camera = Sampling_Time_DC;

// Tham số PID tối ưu hóa
double Kp_A = 0.05, Ki_A = 0.01, Kd_A = 0.01;
double Kp_B = 0.12, Ki_B = 0.01, Kd_B = 0.05;

// Tham số PID khởi động nhẹ nhàng - giảm tích phân để tránh vọt lố
double Warmup_Kp_A = 0, Warmup_Ki_A = 0, Warmup_Kd_A = 0;
double Warmup_Kp_B = 0, Warmup_Ki_B = 0, Warmup_Kd_B = 0;

// Hệ số điều chỉnh từ khoảng cách và góc
double KD = 0.7;     // Hệ số khoảng cách
double KTHETA = 2.0; // Hệ số góc

// Khai báo đối tượng PID
PID myPID_A(&W_Feed_Back_From_DC_A, &PWM_Output_DC_A, &Set_RPM_A, Kp_A, Ki_A, Kd_A, DIRECT);
PID myPID_B(&W_Feed_Back_From_DC_B, &PWM_Output_DC_B, &Set_RPM_B, Kp_B, Ki_B, Kd_B, DIRECT);
```

```
// Thông số robot

#define STEPS_PER_DEGREE 4.44 // Số bước trên mỗi độ
#define L1 40.0 // Chiều dài đoạn 1
#define L2 31.5 // Chiều dài đoạn 2
#define R 46.5   // Khoảng cách từ tâm xe đến line cần vẽ (cm)
#define L 115.0 //Chiều dài của xe (cm)

// TỐI ƯU SIÊU TỐC - Tăng tốc độ động cơ
#define MAX_SPEED 1000      // Tăng từ 1000 lên 5000
#define MAX_ACCELERATION 500 // Tăng từ 500 lên 4000
#define STEPS_PER_LOOP 1000000    // Số bước thực hiện mỗi lần trong runMotors

// Khởi tạo động cơ bước
AccelStepper Step_X(1, step_X, Dir_X);
AccelStepper Step_Y(1, step_Y, Dir_Y);

// Biến lưu trữ dữ liệu
float angle1 = 0.0;
float distance1 = 0.0;
float angle2 = 0.0;
float distance2 = 0.0;
```

3. Hàm khởi tạo ban đầu khi cấp nguồn của xe và set up giá trị ban đầu :

```
713  void setup()
714  {
715    float q1, q2;
716    // TỐI ƯU: Tăng tốc độ truyền thông
717    Serial.begin(115200);   // Tăng từ 9600 lên 115200
718    Serial1.begin(115200);  // Tăng từ 9600 lên 115200
719    Serial1.setTimeout(5);  // TỐI ƯU: Giảm timeout từ 10ms xuống 5ms
720    Serial.println("SUPER BEAST MODE ACTIVATED!");
721
722    pinMode(PWM_A, OUTPUT);
723    pinMode(PWM_B, OUTPUT);
724
725    pinMode(EncoderA, INPUT_PULLUP);
726    attachInterrupt(digitalPinToInterrupt(EncoderA), Interrupt_Encoder_A, CHANGE);
727    pinMode(EncoderB, INPUT_PULLUP);
728    attachInterrupt(digitalPinToInterrupt(EncoderB), Interrupt_Encoder_B, CHANGE);
729    pinMode(EncoderC, INPUT_PULLUP);
730
731    // Đặt PID ở chế độ MANUAL và reset I-term
732    myPID_A.SetMode(MANUAL);
733    myPID_B.SetMode(MANUAL);
734    myPID_A.SetOutputLimits(0, 0);
735    myPID_B.SetOutputLimits(0, 0);
736    myPID_A.Compute();
737    myPID_B.Compute();
738    myPID_A.SetOutputLimits(0, 255);
```

```
// Khởi tạo PWM = 0
analogWrite(PWM_A, 0);
analogWrite(PWM_B, 0);

// Tính toán tham số ban đầu
Tinh_toan_tham_so();


// Cấu hình chân GPIO
pinMode(end_X, INPUT_PULLUP);
pinMode(end_Y, INPUT_PULLUP);

// Cấu hình động cơ
Step_X.setEnablePin(ena_X);
Step_X.setPinsInverted(false, false, true);
Step_X.setMaxSpeed(MAX_SPEED);       // TỐI ƯU: Tăng tốc độ tối đa
Step_Y.setEnablePin(ena_Y);
Step_Y.setPinsInverted(false, false, true);

home_X(); // Tìm vị trí gốc của động cơ X
home_Y(); // Tìm vị trí gốc của động cơ Y
delay(3000);
// homeY(); // Tìm vị trí gốc của động cơ Y

Step_X.setAcceleration(MAX_ACCELERATION);   // TỐI ƯU: Tăng gia tốc
Step_Y.setMaxSpeed(MAX_SPEED);       // TỐI ƯU: Tăng tốc độ tối đa
Step_Y.setAcceleration(MAX_ACCELERATION);   // TỐI ƯU: Tăng gia tốc
```

Khởi tạo động lực học cho ROBOT sau khi đã set Home:

```
387
388  void initialKinematicSETUP(float x, float y, float &q1, float &q2) {
389    float distToTarget = sqrt(x*x + y*y);
390    if (distToTarget > (L1 + L2)) {
391      // TỐI ƯU: Tính toán nhanh hơn và bỏ Serial.println
392      x = x * (L1 + L2 - 1) / distToTarget;
393      y = y * (L1 + L2 - 1) / distToTarget;
394    }
395
396    float c2 = (x*x + y*y - L1*L1 - L2*L2) / (2 * L1 * L2);
397
398    // Giới hạn c2 trong khoảng [-1, 1] để tránh lỗi math
399    c2 = constrain(c2, -1.0, 1.0);
400
401    // Sử dụng -sqrt để có q2 âm (elbow-down configuration)
402    q2 = atan2(-sqrt(1 - c2*c2), c2); // Góc q2 sẽ luôn âm
403
404    float s2 = sin(q2);
405    float k1 = L1 + L2 * c2;
406    float k2 = L2 * s2;
407    q1 = atan2(y, x) - atan2(k2, k1); // Góc q1
408
409    // Đảm bảo q1 luôn dương
410    if (q1 < 0) {
411      q1 += 2 * PI;
412    }
413
```

```
765   Step_X.setAcceleration(MAX_ACCELERATION);   // TỐI ƯU: Tăng gia tốc
766   Step_Y.setMaxSpeed(MAX_SPEED);       // TỐI ƯU: Tăng tốc độ tối đa
767   Step_Y.setAcceleration(MAX_ACCELERATION);   // TỐI ƯU: Tăng gia tốc
768
769   // Bật động cơ
770   Step_X.enableOutputs();
771   Step_Y.enableOutputs();
772
773   // Khởi tạo vị trí ban đầu khi xe chưa lệch
774   initialKinematicSETUP(R, 0, q1, q2);
775
776   // Chuyển đổi góc q1, q2 sang bước (step)
777   int step1 = (int)(q1 * STEPS_PER_DEGREE);
778   int step2 = (int)(q2 * STEPS_PER_DEGREE);
779
780   // Đặt vị trí đích cho cả hai động cơ
781   Step_X.moveTo(step1);
782   Step_Y.moveTo(step2);
783
784   // TỐI ƯU: Di chuyển về vị trí ban đầu nhanh hơn
785   while (Step_X.distanceToGo() != 0 || Step_Y.distanceToGo() != 0) {
786     if (Step_X.distanceToGo() != 0) Step_X.run();
787     if (Step_Y.distanceToGo() != 0) Step_Y.run();
788   }
789
790   #ifdef DEBUG_MODE
791   Serial.print("Moving to steps: X=");
```

4. Hàm set vị trí ban đầu cho động cơ step điều khiển cơ cấu Robot 2 DOF:

```
347
348 void home_X() {
349   Step_X.setMaxSpeed(200);      // TỐI ƯU: Tăng tốc độ homing để nhanh hơn
350   Step_X.setAcceleration(100);  // TỐI ƯU: Tăng gia tốc homing
351   Step_X.enableOutputs();
352   Serial.println("Homing X axis...");
353
354   Step_X.moveTo(-4000); // Di chuyển về bên trái
355
356   unsigned long startTime = millis();
357   const unsigned long timeout = 5000; // TỐI ƯU: Giảm timeout từ 10s xuống 5s
358
359   while (digitalRead(end_X) == 1 && (millis() - startTime < timeout)) {
360     Step_X.run();
361     // TỐI ƯU: Bỏ delayMicroseconds để tăng tốc độ
362   }
363
364   Step_X.setCurrentPosition(0);
365   Serial.println("X axis homed!");
366 }
367
```

```
368
369 void home_Y() {
370   Step_Y.setMaxSpeed(500);       // TỐI ƯU: Tăng tốc độ homing để nhanh hơn
371   Step_Y.setAcceleration(300);   // TỐI ƯU: Tăng gia tốc homing
372   Step_Y.enableOutputs();
373   Serial.println("Homing Y axis...");
374
375   Step_Y.moveTo(100000); // Di chuyển về bên trái (hướng về endstop)
376
377   unsigned long startTime = millis();
378   const unsigned long timeout = 5000; // TỐI ƯU: Giới hạn thời gian homing
379
380   while (digitalRead(end_Y) == HIGH && (millis() - startTime < timeout)) {
381     Step_Y.run();
382   }
383
384   Step_Y.setCurrentPosition(0);
385   Serial.println("Y axis homed!");
386 }
387
```

5 / Tính toán giá trị output để điều khiển động cơ bánh xe :

```
267       last_pwm_A = PWM_Output_DC_A;
268       last_pwm_B = PWM_Output_DC_B;
269     } else {
270       // Giai đoạn hoạt động bình thường với tham số PID đầy đủ
271       myPID_A.Compute();
272       myPID_B.Compute();
273     }
274
275     // Giới hạn giá trị PWM trong phạm vi hợp lệ
276     PWM_Output_DC_A = constrain(PWM_Output_DC_A, 0, 255);
277     PWM_Output_DC_B = constrain(PWM_Output_DC_B, 0, 255);
278   } else {
279     // Động cơ không hoạt động, giữ PWM ở mức thấp
280     PWM_Output_DC_A = Initial_PWM;
281     PWM_Output_DC_B = Initial_PWM * 1.15;
282   }
283
284   // Tính toán và điều chỉnh tốc độ dựa trên dữ liệu camera
285   if (runEvery(Sampling_Time_Camera, 1) && Is_Power_On) {
286
287     double omega = KD * distance2;
288     //distance > 0 -> Xe nghiêng trái, tăng tốc bánh trái (DC_A), giảm bánh phải (DC_B)
289     //distance < 0 -> Xe nghiêng phải, tăng tốc bánh phải (DC_B), giảm bánh trái (DC_A)
290     //Giữ nguyên tốc độ trung bình của xe
291     double base_speed_ms = Set_Speed * 1000.0 / 3600.0;
292
293     Set_Speed_DC_A = base_speed_ms + (omega * 0.6 / 2.0);
294     Set_Speed_DC_B = base_speed_ms - (omega * 0.6 / 2.0);
```

```
22
23  void Control_DC() {
24    // Xuất giá trị PWM đến động cơ
25    analogWrite(PWM_A, PWM_Output_DC_A);
26    analogWrite(PWM_B, PWM_Output_DC_B);
27
28    // Debug information
29    Serial.print("FB_A:"); Serial.print(W_Feed_Back_From_DC_A);
30    Serial.print(" | Set_A:"); Serial.print(Set_RPM_A);
31    Serial.print(" | PWM_A:"); Serial.print(PWM_Output_DC_A);
32    Serial.print(" | FB_B:"); Serial.print(W_Feed_Back_From_DC_B);
33    Serial.print(" | Set_B:"); Serial.print(Set_RPM_B);
34    Serial.print(" | PWM_B:"); Serial.print(PWM_Output_DC_B);
35
36    // Thêm trạng thái khởi động vào debug
37    if (Is_Warming_Up) {
38      Serial.print(" | Warming-up: ");
39      Serial.print((millis() - Warmup_Start_Time) / 1000.0);
40      Serial.println("s");
41    } else if (Is_Power_On) {
42      Serial.println(" | Normal");
43    } else {
44      Serial.println(" | Idle");
45    }
46  }
47
```

6. Hàm nhận giá trị từ YOLO:

```
457  bool PC_To_Slave() {
458    if (Serial1.available() > 0) {
459      String data = Serial1.readStringUntil('\n');
460      data.trim();  // Xoá ký tự \r hoặc khoảng trắng
461
462      // TỐI ƯU: Kiểm tra nhanh định dạng, bỏ đếm dấu phẩy
463      int i1 = data.indexOf(',');
464      int i2 = data.indexOf(',', i1 + 1);
465      int i3 = data.indexOf(',', i2 + 1);
466
467      if (i1 <= 0 || i2 <= i1 || i3 <= i2) {
468        // Định dạng không đúng
469        return false;
470      }
471
472      // Tách dữ liệu
473      String a1_str = data.substring(0, i1);
474      String d1_str = data.substring(i1 + 1, i2);
475      String a2_str = data.substring(i2 + 1, i3);
476      String d2_str = data.substring(i3 + 1);
477
478      // Chuyển sang float nhanh hơn
479      float a1 = a1_str.toFloat();
480      float d1 = d1_str.toFloat();
481      float a2 = a2_str.toFloat();
482      float d2 = d2_str.toFloat();
```

7. Hàm thuật toán điều khiển cho cánh tay Robot:

```
568  */
569  void LechTraiHuongTrai(){
570    // Tính giá trị delta từ khoảng cách (đơn vị cm)
571    float Xp = R + (distance1 + distance2)/2.0 ;
572    float K = abs(Xp - R);
573
574    float delta = cos(angle1 * PI / 180.0) * K;
575
576
577    // Tính toán vị trí X và Y
578    float X = R + delta;
579    float Y = 0;
580
581    // Sử dụng hàm moveRobot với góc bù -angle1
582    moveRobot(X, Y, -angle1);
583  }
584
585  void KhonglechHuongTrai(){
586    moveRobot(R, 0, -angle1);
587  }
588
```

```
584
585  void KhonglechHuongTrai(){
586    moveRobot(R, 0, -angle1);
587  }
588
589  void LechPhaiHuongTrai(){
590    float Xp = R + (distance1 + distance2)/2.0 ;
591    float K = abs(Xp - R);
592
593    float delta = cos(angle1 * PI / 180.0) * K;
594    // Tính toán vị trí X và Y
595    float X = R - delta;
596    float Y = 0;
597
598
599    // Sử dụng hàm moveRobot với góc bù -angle1
600    moveRobot(X, Y, -angle1);
601  }
602
603  void LechTraiHuongPhai(){
604    // Tính giá trị delta từ khoảng cách (đơn vị cm)
605    float Xp = R + (distance1 + distance2)/2.0 ;
606    float K = abs(Xp - R);
607
608    float delta = cos(angle1 * PI / 180.0) * K;
609
610    // Tính toán vị trí X và Y
--
14     // Sử dụng hàm moveRobot với góc bù angle1
15     moveRobot(X, Y, angle1);
16   }
17
18   void KhongLechHuongPhai(){
19     moveRobot(R, 0, angle1);
20   }
21
22   void LechPhaiHuongPhai(){
23     // Tính giá trị delta từ khoảng cách (đơn vị cm)
24     float Xp = R + (distance1 + distance2)/2.0;
25     float K = abs(Xp - R);
26
27     float delta = cos(angle1 * PI / 180.0) * K;
28
29     // Tính toán vị trí X và Y
30     float X = R - delta;
31     float Y = 0;
32
33     // Sử dụng hàm moveRobot với góc bù angle1
34     moveRobot(X, Y, angle1);
35   }
36
```

Hàm điều khiển tổng:

```
void controlRobot() {
  // Kiểm tra các điều kiện để điều khiển robot

  // TỐI ƯU: Sắp xếp lại điều kiện để kiểm tra nhanh hơn
  if (distance1 > 0 && distance2 < 0) {
    // Xe nghiêng bên trái
    if (distance1 > abs(distance2)) {
      LechTraiHuongTrai();
    } else if (distance1 == abs(distance2)) {
      KhonglechHuongTrai();
    } else if (distance1 < abs(distance2)) {
      LechPhaiHuongTrai();
    }
  } else if (distance1 < 0 && distance2 > 0) {
    // Xe nghiêng bên phải
    if (abs(distance1) < distance2) {
      LechTraiHuongPhai();
    } else if (abs(distance1) == distance2) {
      KhongLechHuongPhai();
    } else if (abs(distance1) > distance2) {
      LechPhaiHuongPhai();
    }
  } else {
    Serial.println(" ❌ Dữ liệu không hợp lệ: Không thỏa mãn bất kỳ điều kiện nào!");
  }
}
```

Hàm tính toán động học sau khi đã tính toán lại giá trị mới nhờ thuật toán:

```
void calculateInverseKinematics(float x, float y, float &q1, float &q2) {
  // TỐI ƯU: Sử dụng các phép tính trực tiếp, giảm biến trung gian
  float distToTarget = sqrt(x*x + y*y);
  if (distToTarget > (L1 + L2)) {
    // TỐI ƯU: Bỏ Serial.println để tăng tốc độ
    float ratio = (L1 + L2 - 1) / distToTarget;
    x *= ratio;
    y *= ratio;
  }

  float c2 = (x*x + y*y - L1*L1 - L2*L2) / (2 * L1 * L2);

  // Giới hạn c2 trong khoảng [-1, 1] để tránh lỗi math
  c2 = constrain(c2, -1.0, 1.0);

  // TỐI ƯU: Sử dụng phép tính toán một lần
  q2 = atan2(-sqrt(1 - c2*c2), c2); // Góc q2 sẽ luôn âm

  float s2 = sin(q2);
  float k1 = L1 + L2 * c2;
  float k2 = L2 * s2;
  q1 = atan2(y, x) - atan2(k2, k1); // Góc q1

  // Đảm bảo q1 luôn dương
  if (q1 < 0) {
    q1 += 2 * PI;
  }
```

- Code trên google colab:

1. Cài đặt thư viện:

```
!pip install ultralytics roboflow numpy opencv-python pillow
```

2. Tải dữ liệu đã được gắn nhãn từ Roboflow:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="MoplLnzxuUfJrAklyekT")
project = rf.workspace("datn-phwgz").project("datn-3eouf")
version = project.version(8)
dataset = version.download("yolov8")
```

3. Tiến hành train mô hình thị giác máy tính:

```
from ultralytics import YOLO

# Tạo mô hình
model = YOLO('yolov8n-seg.pt')  # Phiên bản segmentation

# Training (không cần chỉ định boxes/mask ở đây)
results = model.train(
    data='/content/DATN-8/data.yaml',
    epochs=100,
    imgsz=640,
    batch=16,
    device=0  # GPU
)
```

- Code trên Visual Studio:

1. Tải các thư viện cần thiết:

```
import cv2
import numpy as np
import math
import serial
import time
from ultralytics import YOLO
```

2. Khởi tạo mô hình và camera:

```
model = YOLO('final.pt')  # Model đã train
cap_left = cv2.VideoCapture(0, cv2.CAP_DSHOW)  # Camera trái

cap_right = cv2.VideoCapture(1, cv2.CAP_DSHOW)  # Camera phải
```

3. Khai báo các biến:

```
    angle_left = None
    distance_mm_left = None
    angle_right = None
    distance_mm_right = None

    # Tỷ lệ mm/pixel cho từng camera (cần hiệu chỉnh riêng)
    MM_PER_PIXEL_LEFT = 0.0909   # Ví dụ: 1 pixel = 0.1mm (camera trái)
    MM_PER_PIXEL_RIGHT = 0.0633 # Ví dụ: 1 pixel = 0.095mm (camera phải)
```

4. Vẽ trục tọa độ và hiển thị khoảng cách:

```
def draw_visual_guides(frame, wire_intersect_y=None, mm_per_pixel=None):
    """Vẽ trục tọa độ và hiển thị khoảng cách (mm) - hỗ trợ tỷ lệ riêng"""
    h, w = frame.shape[:2]
    center_x, center_y = w // 2, h // 2

    # Vẽ trục tọa độ (giữ nguyên)
    cv2.line(frame, (center_x, 0), (center_x, h), (0, 255, 0), 1)
    cv2.line(frame, (0, center_y), (w, center_y), (255, 0, 0), 1)
    cv2.drawMarker(frame, (center_x, center_y), (255, 255, 255), cv2.MARKER_CROSS, 15, 1)

    # Hiển thị khoảng cách bằng mm nếu có tỷ lệ và giao điểm
    if wire_intersect_y is not None and mm_per_pixel is not None:

        distance_px = wire_intersect_y - center_y
        distance_mm = distance_px * mm_per_pixel

        cv2.circle(frame, (center_x, int(wire_intersect_y)), 8, (0, 255, 255), -1)
        cv2.putText(frame, f"{abs(distance_mm):.1f}mm",
                    (center_x + 10, int(wire_intersect_y) + 20),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)
```

5. Hàm xử lý ảnh, phát hiện dây và xử lý góc lệch:

```python
def process_frame(frame):
    """Xử lý frame và trả về góc lệch, khoảng cách"""
    results = model(frame, conf=0.5)
    if results[0].masks is not None:
        mask = results[0].masks[0].data[0].cpu().numpy()
        contours, _ = cv2.findContours((mask*255).astype(np.uint8), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        if contours:
            largest_contour = max(contours, key=cv2.contourArea)
            [vx, vy, x, y] = cv2.fitLine(largest_contour, cv2.DIST_L2, 0, 0.01, 0.01)
            vx, vy, x, y = vx[0], vy[0], x[0], y[0]

            # Tính góc lệch [-90°, 90°]
            angle_rad = math.atan2(vx, vy)
            angle_deg = math.degrees(angle_rad) - 90
            angle_deg = abs(angle_deg - 360 if angle_deg > 90 else angle_deg + 360 if angle_deg < -270 else angle_deg)

            # Tính giao điểm với trục dọc giữa (thay vì trục ngang)
            h, w = frame.shape[:2]
            center_x = w // 2
            if vx != 0:  # Tránh chia cho 0
                slope = vy / vx
                y_intercept = y - slope * x
                y_intersect = slope * center_x + y_intercept
            else:  # Đường thẳng đứng
                y_intersect = y

            return angle_deg, y_intersect, (vx, vy, x, y)
    return None, None, None
```

6. Khởi tạo kết nối Arduino:

```python
# Kết nối với Arduino qua USB-to-TTL (COM tùy máy bạn, ví dụ COM8)
arduino = serial.Serial('COM7', 115200 , timeout=1)  # CHỈ CẦN ĐỔI COM NÀY
time.sleep(2)  # Đợi Arduino khởi động
last_sent_time = time.time()
SEND_INTERVAL = 0.1  # Giây
```

7. Vòng lặp chính-Đọc camera, xử lý ảnh, vẽ hình và gửi dữ liệu:

```python
# Vòng lặp chính
while True:
    ret_left, frame_left = cap_left.read()
    ret_right, frame_right = cap_right.read()
    if not ret_left or not ret_right:
        break

    # Xử lý frame và nhận kết quả
    angle_left, y_intersect_left, coords_left = process_frame(frame_left)
    angle_right, y_intersect_right, coords_right = process_frame(frame_right)

    # Xử lý cho camera TRÁI
    draw_visual_guides(frame_left, y_intersect_left, MM_PER_PIXEL_LEFT)
    if angle_left is not None:
        vx, vy, x, y = coords_left
        cv2.line(frame_left, (int(x - vx*1000), int(y - vy*1000)),
                 (int(x + vx*1000), int(y + vy*1000)), (0, 0, 255), 2)

        distance_mm_left = (y_intersect_left - frame_left.shape[0]//2) * MM_PER_PIXEL_LEFT if y_intersect_left is not None else None
        cv2.putText(frame_left, f"L: {angle_left:.1f}° | {distance_mm_left:.1f}mm",
                    (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)
```

```
# Xử lý cho camera PHẢI
draw_visual_guides(frame_right, y_intersect_right, MM_PER_PIXEL_RIGHT)
if angle_right is not None:
    vx, vy, x, y = coords_right
    cv2.line(frame_right, (int(x - vx*1000), int(y - vy*1000)),
             (int(x + vx*1000), int(y + vy*1000)), (0, 0, 255), 2)

    distance_mm_right = (y_intersect_right - frame_right.shape[0]//2) * MM_PER_PIXEL_RIGHT if y_intersect_right is not None else None
    cv2.putText(frame_right, f"R: {angle_right:.1f}° | {distance_mm_right:.1f}mm",
                (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)


    # Truyền dữ liệu nếu đủ
    if None not in (angle_left, distance_mm_left, angle_right, distance_mm_right):
        current_time = time.time()
        if current_time - last_sent_time >= SEND_INTERVAL:
            data_str = f"{angle_left:.1f},{distance_mm_left:.1f},{angle_right:.1f},{distance_mm_right:.1f}\n"
            arduino.write(data_str.encode())
            last_sent_time = current_time
            print(f"Đã gửi: {data_str.strip()}")


else:
    print("Skipping frame due to missing data")
```

8. Hiển thị và thoát:

```
    cv2.imshow("Left Camera", frame_left)
    cv2.imshow("Right Camera", frame_right)

    if cv2.waitKey(1) == 27:
        break
```

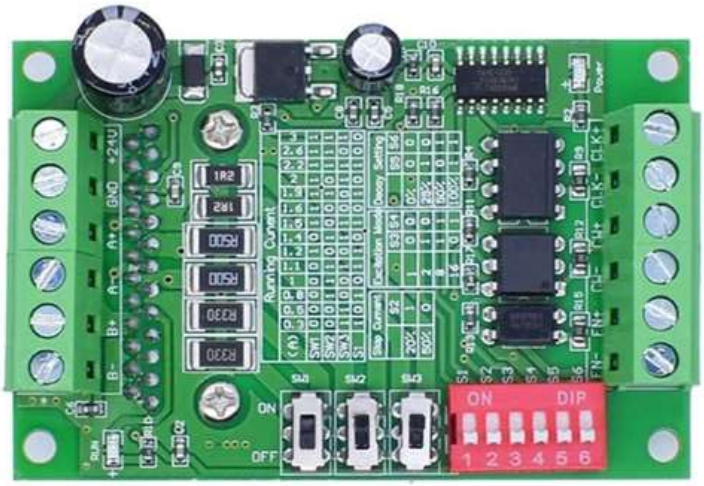
PHỤ LỤC 10: DANH SÁCH LINH KIỆN, THIẾT BỊ

- Arduino MEGA

- Driver BTS7960

- Động cơ DC MY1016Z

- Driver Step Motor TB6560



- Động cơ Step

- Encoder HN3806-AB-600G

- Ắc quy 12V

- Webcam

- USB to TTL