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:

NGHIÊN CỨU VÀ THIẾT KẾ BỘ ĐIỀU KHIỂN THÍCH NGHI DỰA TRÊN LOGIC MỜ CHO XE TỰ HÀNH

Giảng viên hướng dẫn: **TS. TRẦN THỊ MINH DUNG**
Cán bộ hướng dẫn: **ÔNG LƯU ĐỨC HÒA**
Sinh viên thực hiện:
1. NGUYỄN DOÃN QUANG – MSSV: 105200341 – LỚP: 20TDH2
2. ĐÀO NGUYÊN ĐÁN – MSSV: 105200326 – LỚP: 20TDH2

Đà Nẵng, 06/2025

TÓM TẮT

Tên đề tài: NGHIÊN CỨU VÀ THIẾT KẾ BỘ ĐIỀU KHIỂN THÍCH NGHI DỰA TRÊN LOGIC MỜ CHO XE TỰ HÀNH

Sinh viên thực hiện:

NGUYỄN DOÃN QUANG – MSSV: 105200341 – LỚP: 20TDH2

ĐÀO NGUYÊN ĐÁN – MSSV: 105200326 – LỚP: 20TDH2

Đề tài nhóm nghiên cứu tập trung vào việc thiết kế và phát triển bộ điều khiển thích nghi kết hợp với logic mờ cho hệ thống xe tự hành. Phương pháp điều khiển thích nghi cho phép hệ thống tự điều chỉnh tham số trong quá trình hoạt động khi có sự thay đổi từ môi trường. Logic mờ khai thác khả năng suy luận xấp xỉ và mô tả mối quan hệ đầu vào – đầu ra bằng các luật IF–THEN, đóng vai trò xử lý các thông tin không chính xác, mơ hồ vốn rất phổ biến trong các hệ thống thực tế. Việc kết hợp 2 phương pháp này giúp ta xây dựng được một bộ điều khiển vừa linh hoạt cũng như có khả năng chỉnh định giúp hệ thống đảm bảo được tính ổn định, nâng cao độ chính xác, khả năng tự dộng hóa – giảm sai lệch khi có nhiễu, phản ứng chính xác trước sự thay đổi của môi trường. Đây là điều cần thiết mà đề tài nhóm nghiên cứu muốn hướng đến nhằm đáp ứng được yêu cầu của các hệ thống xe tự hành trong công nghiệp.

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

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Nguyễn Doãn Quang	105200341	20TDH2	Kỹ thuật Điều khiển và Tự động hóa
2	Đào Nguyên Đán	105200326	20TDH2	Kỹ thuật Điều khiển và Tự động hóa

1. Tên đề tài đồ án:

Phân tích hệ thống điều khiển theo công nghệ thổi LDPE tại Công ty Cổ phần BMP GROUP

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	Nguyễn Doãn Quang	Tìm hiểu chung về AGV Robot, các phương pháp điều khiển và bộ điều khiển Fuzzy - PID
2	Đào Nguyên Đán	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Doãn Quang	- Thiết kế bộ điều khiển PID cho từng cơ sở - Tổng hợp bộ điều khiển Fuzzy – PID cho toàn bộ AGV Robot - Mô phỏng hệ thống trên Matlab – Simulink - Thi công mô hình (đấu nối phần cứng, lập trình)
2	Đào Nguyên Đán	- Nghiên cứu mô hình động học, động lực học của AGV Robot - Mô hình hóa hệ thống - Tính chọn thiết bị - Thi công mô hình (đấu nối, lập trình)

5. Các bản vẽ, đồ thị

a. Phần chung:

TT	Họ tên sinh viên	Nội dung

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Nguyễn Doãn Quang	Bản vẽ lưu đồ thuật toán
2	Đào Nguyên Đán	Bản vẽ sơ đồ công nghệ

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Trần Thị Minh Dung	Định hướng thiết kế bộ điều khiển Fuzzy - PID
Ông Lưu Đức Hòa	Tổng quan của bộ điều khiển thích nghi

7. Ngày giao nhiệm vụ đồ án: 16/2/2025

8. Ngày hoàn thành đồ án: 16/6/2025

Đà Nẵng, ngày 16 tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa **Giảng viên hướng dẫn**

TS. Giáp Quang Huy TS. Trần Thị Minh Dung

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: Nguyễn Doãn Quang Số thẻ SV: 105200341

Họ tên sinh viên: Đào Nguyên Đán Số thẻ SV: 105200326

Tên đề tài ĐATN: Nghiên cứu và thiết kế bộ điều khiển thích nghi dựa trên Logic mờ cho xe tự hành

Họ tên người HD: TS. Trần Thị Minh Dung Đơn vị: Khoa Điện, Trường Đại học Bách Khoa

Họ tên người HD: Lưu Đức Hòa Đơn vị: Công ty Hòa Phát An

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1				
2				
3				
4		Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5				
6				
7				
8		Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9				
10				
11				
12		Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13				

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trong xu thế phát triển mạnh mẽ của cuộc cách mạng công nghiệp 4.0, ứng dụng hệ thống điều khiển tự động và robot thông minh vào hoạt động sản xuất và vận chuyển nội bộ trong doanh nghiệp ngày càng trở nên thiết yếu. Đặc biệt, các hệ thống vận hành linh hoạt như robot tự hành AGVđang được quan tâm và phát triển rộng rãi nhờ khả năng tối ưu hóa năng suất, giảm thiểu sức lao động thủ công và tăng tốc độ an toàn trong môi trường sản xuất.

Xuất phát từ thực tiễn đó, đồ án tốt nghiệp với đề tài "Nghiên cứu và thiết kế bộ điều khiển thích nghi dựa trên logic mờ cho xe tự hành" được thực hiện nhằm mục tiêu xây dựng mô hình điều khiển hai cấp: sử dụng Fuzzy-PID để điều khiển tốc độ tổng thể và PID để điều khiển độ chính xác của động cơ. Nội dung sơ đồ tập trung vào các bước chính như: khảo sát mô hình động học – động lực học xe Mecanum, nhận dạng hệ thống từ dữ liệu mô phỏng, thiết kế – so sánh các bộ điều khiển và tiến trình mô phỏng kiểm tra tính ổn định của hệ thống trong các điều kiện tải khác nhau.

Để thực hiện được đồ án tốt nghiệp này, nhóm chúng em xin gửi lời cảm ơn chân thành đến:

- **Cô Trần Thị Minh Dung** – Giảng viên Khoa Điện – Trường Đại học Bách Khoa – Đại học Đà Nẵng, người đã tận dụng tình hướng dẫn, hỗ trợ chuyên môn và đóng góp ý kiến xuyên suốt quá trình thực hiện đề tài.

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Nhóm chúng em xin cam đoan mọi nội dung, bản vẽ, hình ảnh và biểu đồ trong báo cáo đều được xây dựng dựa trên quá trình khảo sát, mô phỏng, thiết kế và phân tích thực tế của nhóm, không sao chép từ bất kỳ báo cáo nào khác.

Trong quá trình thực hiện đề tài, nhóm em luôn tuân thủ các quy định về đạo đức học thuật, đảm bảo tính trung thực trong việc trích dẫn tài liệu tham khảo cũng như không vi phạm bản quyền đối với bất kỳ sản phẩm trí tuệ nào.

Nhóm em xin cam kết hoàn toàn trách nhiệm trước Khoa, Nhà trường và các bên liên quan nếu phát hiện ra hành vi vi gian học thuật, làm giả số liệu, hoặc sử dụng trái phép nội dung từ các nguồn không được phép.

Sinh viên thực hiên	Sinh viên thực hiện
Đào Nguyên Đán	Nguyễn Doãn Quang

MỤC LỤC

TÓM TẮT ..... **i**

LỜI NÓI ĐẦU VÀ CẢM ƠN ..... **v**

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT ..... **vi**

DANH SÁCH CÁC HÌNH VẼ ..... **ix**

DANH SÁCH CÁC BẢNG ..... **xi**

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT ..... **xii**

MỞ ĐẦU **1**

Chương 1: TỔNG QUAN ĐỀ TÀI **4**

1.1 Tổng quan về AGV Robot **4**

1.2 Các loại AGV Robot hiện nay trên thị trường **6**

1.3 AGV Robot tải hàng (AGV Robot Unit Load) **8**

1.4 Những thách thức AGV Robot tải hàng (AGV Robot Unit Loading) **9**

1.5 Đề xuất giải pháp **9**

Chương 2: NGHIÊN CỨU VÀ THIẾT KẾ AGV ROBOT **10**

2.1 Đặt vấn đề **10**

2.2 Giải pháp **10**

2.3 Sơ đồ công nghệ **11**

2.4 Thiết kế kiến trúc điều khiển hệ thống **12**

2.3.1 Điều khiển cấp cao 12

2.3.2 Điều khiển cấp thấp 13

2.4 Sơ đồ khối điều khiển **13**

2.5 Thiết kế cấu trúc hệ thống AGV Robot **15**

2.5.1 Thiết kế AGV Robot mong muốn 15

2.5.2 AGV Robot trong đề tài 16

Chương 3: MÔ HÌNH HÓA HỆ THỐNG VÀ THIẾT KẾ BỘ ĐIỀU KHIỂN FUZZY – PID **17**

3.1 Mô hình hóa hệ thống.....17
3.1.1 Mô hình động học của AGV Robot.....17
3.1.2 Mô hình động lực học của AGV Robot.....24
3.2 Thiết kế bộ điều khiển.....26
3.2.1 Thiết kế bộ điều khiển PID của động cơ DC.....26
3.2.2 Thiết kế bộ điều khiển Fuzzy – PID.....30
Chương 4: THI CÔNG MÔ HÌNH.....46
4.1 Sơ đồ kết nối phần cứng.....47
4.2 Lưu đồ thuật toán.....48
4.3 Thi công mô hình.....49
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....50
TÀI LIỆU THAM KHẢO.....52
PHỤ LỤC.....1

DANH SÁCH CÁC HÌNH VẼ

Hình 1. 1 Ví dụ điển hình Robot AGV trong công nghiệp 4
Hình 1. 2 AGV Robot vận chuyển hàng trong kho và bệnh viện.....6
Hình 1. 3 AGV Robot kéo hàng ITA – 01 của Intech Group.....6
Hình 1. 4 AGV kéo hàng (Towing AGV) 7
Hình 1. 5 AGV chở hàng (Unit Load AGV) 7
Hình 1. 6 AGV dạng xe nâng (Forklift AGV).....7
Hình 1. 7 AGV dạng robot di động nhỏ (Small Mobile Robot).....8
Hình 1. 8 AGV định vị dẫn đường 8

Hình 2. 1 Sơ đồ công nghệ của hệ thống.....11
Hình 2. 2 Sơ đồ về cấu trúc điều khiển của hệ thống.....12
Hình 2. 3 Sơ đồ khối điều khiển của hệ thống 13
Hình 2. 4 AGV Robot trong đề tài nhóm thực hiện 16

Hình 3. 1 Chuyển động đa hướng của AGV Robot.....18
Hình 3. 2 Sơ đồ khối bộ điều khiển PID 26
Hình 3. 3 Kết quả mô phỏng bộ điều khiển PID của động cơ DC 29
Hình 3. 4 Kết quả mô phỏng bộ điều khiển PID động cơ DC.....29
Hình 3. 5 Cấu trúc tổng quát của một hệ mờ.....30
Hình 3. 6 Nhiệm vụ của khâu giải mờ.....35
Hình 3. 7 Cơ sở toán học của phương pháp điều khiển mờ 35
Hình 3. 8 Các khối chứng năng của bộ điều khiển mờ 35
Hình 3. 9 Cấu trúc bộ điều khiển Fuzzy của hệ thống trên Matlab – Simulink 37
Hình 3. 10 Các biến ngôn ngữ và giá trị đầu vào 38
Hình 3. 11 Các biến ngôn ngữ và giá trị đầu ra.....38
Hình 3. 12 Luật hợp thành của bộ điều khiển Fuzzy.....43
Hình 3. 13 Bộ điều khiển PID 43
Hình 3. 14 Bộ điều khiển Fuzzy – PID của hệ thống.....44
Hình 3. 15 Động lực học của hệ thống.....44
Hình 3. 16 Kết quả mô phỏng giữa 2 bộ điều khiển PID và Fuzzy – PID 44

Hình 4. 1 Sơ đồ kết nối.....47
Hình 4. 2 Lưu đồ thuật toán.....48
Hình 4. 3 Khung xe và giá lắp động cơ.....49
Hình 4. 4 Hoàn thiện hệ thống cơ khí của hệ thống.....49
Hình 4. 5 Hoàn thiện mô hình.....49

DANH SÁCH CÁC BẢNG

Bảng 3. 1 Tham số hệ thống.....22
Bảng 3. 2 Một số phép giao thường dùng (phép hội).....32
Bảng 3. 3 Một số phép hợp thường dùng (phép tuyển).....33
Bảng 3. 4 Luật hợp thành 34
Bảng 3. 5 Luật điều khiển Kp.....39
Bảng 3. 6 Luật điều khiển Ki.....41
Bảng 3. 7 Luật điều khiển Kd.....42
Bảng 3. 8 Bảng so sánh kết quả giữa 2 phương pháp 45

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU:

x, y, θ	Vị trí của robot tại tọa độ (x, y) và θ là góc định hướng (góc là góc giữa hệ trục X và X_R)
XGY	Hệ quy chiếu quán tính, (x, y) là hệ tọa độ Đề các gắn với chuyển động của thân xe
S_i, P_i, E_i	Hệ tọa độ gắn với bánh xe thứ i tại tâm điểm P_i của bánh xe
O, P_i	Là cơ sở quán tính của xe trong hệ quy chiếu xe và $P_i = \{X_{P_i}, Y_{P_i}\}$ tại tâm trục quay của bánh xe
$\overrightarrow{OP_i}$	Vector chỉ khoảng cách giữa tâm xe và tâm của bánh xe thứ
l_{ix}, l_{iy}	l_{ix} là một nửa khoảng cách giữa hai bánh trước và l_{iy} là một nửa khoảng cách giữa bánh trước và bánh sau
l_i	Khoảng cách giữa bánh xe và tâm của xe
r_i	Bán kính của bánh xe thứ i (hay khoảng cách giữa trục quay của bánh xe và tâm của con lăn)
r_r	Bán kính của con lăn gắn trên bánh xe
r_{OL}	Khoảng cách từ con lăn đến trục bánh xe i
α_i	Góc giữa $\overrightarrow{OP_i}$ trục X_R của hệ tọa độ xe
β_i	Góc giữa S_i và X_R
γ_i	Góc giữa v_{ir} và E_i
ω_z	Vận tốc góc tuyến tính của xe
v_x, v_y	Vận tốc tuyến tính của xe
$[v_{si}\ \mathrm{v}_{Ei}\ \omega_i]^T$	Vận tốc tổng quát của điểm P_i trong hệ trục tọa độ $X_R O Y_R$
$[w_{si}\ \mathrm{w}_{Ei}\ \omega_i]^T$ là	Vận tốc tổng quát của điểm P_i trong hệ trục tọa độ $S_i P_i E_i$
v_{ir}	Vận tốc của con lăn thụ động trong bánh xe i

$v_{i\omega}$	Vận tốc dài của bánh xe i
ω_i	Vận độ góc của bánh xe i
$X_R O Y_R$	Hệ trục tọa độ xoay quanh thân xe
$S_i P_i E_i$	Hệ trục tọa độ xoay quanh bánh xe

CHỮ VIẾT TẮT:

AGV: Automated Guided Vehicle

PID: Proportional–Integral–Derivative

FIS: Fuzzy Inference System

MỞ ĐẦU

1. Mục đích thực hiện đề tài

Mục đích chính của đề tài là nghiên cứu và thiết kế một bộ điều khiển thích nghi sử dụng logic mờ kết hợp với điều khiển PID nhằm cải thiện khả năng ổn định vận tốc của xe tự hành trong các điều kiện thay đổi về tải trọng và môi trường làm việc. Qua đó, xây dựng được nền tảng điều khiển có khả năng thích nghi, phù hợp với yêu cầu thực tiễn trong công nghiệp hiện đại, nơi các hệ thống vận chuyển cần tính linh hoạt và ổn định cao.

2. Mục tiêu đề tài

- Thiết kế mô hình xe tự hành điều khiển độc lập bằng 4 động cơ riêng biệt.

- Phân tích và xây dựng mô hình động học, động lực học của hệ thống.

- Thiết kế bộ điều khiển thích nghi điều khiển dựa trên logic mờ

- Triển khai mô hình thực nghiệm thu nhỏ có tích hợp cảm biến, hiển thị và điều khiển từ xa, giúp kiểm chứng nguyên lý điều khiển.

3. Phạm vi và đối tượng nghiên cứu

- Đối tượng nghiên cứu: Hệ thống xe tự hành với điều khiển động cơ độc lập và bộ điều khiển logic mờ – PID.

- Phạm vị nghiên cứu:

+ Tập trung vào điều khiển vận tốc và phản ứng của hệ thống khi có thay đổi tải trọng.

+ Mô hình thực nghiệm sử dụng các linh kiện mô phỏng thu nhỏ so với bản thiết kế lý thuyết ban đầu, phù hợp với quy mô đồ án.

4. Phương pháp nghiên cứu

- Phương pháp thu thập tài liệu: Tổng hợp từ các tài liệu khoa học, giáo trình, bài báo, và ứng dụng công nghiệp liên quan đến điều khiển thích nghi và bánh xe Mecanum.

- Phân tích – mô hình hóa: Dựa trên nguyên lý động học, động lực học để xây dựng mô hình toán học của hệ thống.

- Mô phỏng: Thực hiện trên MATLAB/Simulink để kiểm tra và so sánh hiệu quả giữa bộ điều khiển PID và Fuzzy-PID.

- Thực nghiệm: Triển khai mô hình thu nhỏ bằng Arduino, cảm biến, động cơ thực tế, đo lường và hiển thị thông số để kiểm nghiệm tính khả thi của mô hình điều khiển đã thiết kế.

5. Ý nghĩa khoa học và thực tiễn

- Ý nghĩa khoa học: Cung cấp một giải pháp hiện đại cho bài toán điều khiển động cơ, đồng thời đóng góp vào sự phát triển của các hệ thống điều khiển thông minh. Phát triển các thuật toán điều khiển thích nghi có thể mở rộng áp dụng cho các hệ thống phức tạp hơn. Qua đó góp phần làm rõ và mở rộng các nội dung nghiên cứu trong lĩnh vực điều khiển thông minh cho hệ thống phi tuyến, cụ thể là:

+ Xây dựng và hoàn thiện mô hình điều khiển cho xe tự hành bốn bánh sử dụng điều khiển thích nghi kết hợp với logic mờ – một hướng tiếp cận hiện đại, phù hợp với đặc tính thay đổi và phi tuyến của hệ thống robot di động.

+ Kết hợp lý thuyết điều khiển thích nghi và hệ mờ để tạo ra một bộ điều khiển có khả năng tự điều chỉnh thông minh, đóng vai trò như một lớp suy luận logic giúp hệ thống thích ứng với sai số và nhiễu động trong quá trình di chuyển.

+ Góp phần khẳng định tính khả thi và hiệu quả của phương pháp điều khiển lai (hybrid control) trong các bài toán thực tế, đặc biệt là điều khiển ổn định tốc độ và bám quỹ đạo cho xe tự hành.

+ Tạo nền tảng cho các nghiên cứu tiếp theo trong lĩnh vực điều khiển thông minh, điều khiển học máy (machine learning-based control) và robot tự động.

- Ý nghĩa thực tiễn: Giúp tối ưu hóa hiệu suất vận hành của các hệ thống sử dụng động cơ. Đáp ứng nhu cầu ổn định tốc độ động cơ trong nhiều ứng dụng thực tiễn, từ công nghiệp, giao thông đến thiết bị gia dụng.

+ Tăng độ ổn định và độ chính xác khi điều khiển xe tự hành trong các môi trường có đặc tính thay đổi như địa hình không bằng phẳng, tải trọng biến đổi, nhiễu cảm biến hoặc điều kiện không lý tưởng.

+ Góp phần nâng cao khả năng tự chủ và thông minh hóa cho các thiết bị robot di động, phục vụ trong nhiều lĩnh vực như: vận chuyển trong nhà máy (AGV), xe tự hành ngoài trời, robot nông nghiệp, xe dịch vụ đô thị, ...

+ Mô hình điều khiển được mô phỏng trong môi trường MATLAB/Simulink có thể triển khai nhanh trên nền tảng phần cứng thực tế, phù hợp với xu hướng phát triển hệ thống nhúng thông minh trong công nghiệp 4.0.

+ Là tài liệu tham khảo và ứng dụng hữu ích cho các kỹ sư, sinh viên, nhà nghiên cứu trong lĩnh vực tự động hóa, điều khiển và robot học.

6. Cấu trúc của đồ án tốt nghiệp

CHƯƠNG 1 TỔNG QUAN ĐỀ TÀI

CHƯƠNG 2 NGHIÊN CỨU VÀ THIẾT KẾ AGV ROBOT

CHƯƠNG 3 MÔ HÌNH HÓA VÀ THIẾT KẾ BỘ ĐIỀU KHIỂN FUZZY - PID

CHƯƠNG 4 THI CÔNG MÔ HÌNH

Chương 1: TỔNG QUAN ĐỀ TÀI

1.1 Tổng quan về AGV Robot

- Trong thời đại phát triển không ngừng của công nghệ, AGV Robot nổi lên với những khả năng hiện đại tự động di chuyển và thực hiện nhiều nhiệm vụ cùng một lúc như vận chuyển hàng hóa, nguyên vật liệu, sản phẩm… mà không cần quá nhiều sự can thiệp của con người. Chúng là giải pháp tự động hóa vận chuyển đang được nhiều doanh nghiệp áp dụng trong sản xuất và logistics. Với khả năng hoạt động chính xác, liên tục và an toàn, AGV Robot giúp tối ưu hóa quy trình nội bộ, giảm phụ thuộc vào nhân công và nâng cao hiệu quả vận hành. Thiết bị này phù hợp với nhiều mô hình nhà máy thông minh, từ quy mô vừa đến lớn.

- AGV Robot là một loại phương tiện tự hành không người lái, được thiết kế để vận chuyển hàng hóa trong môi trường sản xuất hoặc kho vận. AGV di chuyển theo tuyến đường định sẵn bằng các công nghệ điều hướng như vạch từ, laser, mã QR hoặc điều hướng tự nhiên. Thiết bị này hoạt động hoàn toàn tự động, giúp giảm chi phí lao động và tăng hiệu quả vận hành trong các hệ thống logistics và sản xuất công nghiệp.

Hình 1. 1 Ví dụ điển hình Robot AGV trong công nghiệp

- Các ưu điểm và hạn chế của AGV Robot:

+ Ưu điểm:

- Giảm chi phí lao động: Vận hành tự động, không cần người điều khiển; Hoạt động liên tục 24/7, tăng năng suất và giảm thời gian gián đoạn.
- Tăng độ an toàn: Trang bị cảm biến (siêu âm, hồng ngoại, camera, lidar) để tránh va chạm; Giảm nguy cơ tai nạn, cải thiện môi trường làm việc; Có thể làm việc trong môi trường độc hại, nguy hiểm.
- Tăng độ chính xác và năng suất: Loại bỏ sai sót do con người gây ra; Tích hợp hệ thống quản lý kho giúp tối ưu hóa quy trình kiểm kê, đặt hàng.
- Tính mở rộng và linh hoạt: Dễ mở rộng quy mô bằng cách bổ sung AGV mà không cần thay đổi hạ tầng; Thích ứng linh hoạt với thay đổi của nhà xưởng, dây chuyền; Tương thích với các hệ thống tự động khác như robot, máy CNC.

+ Hạn chế:

- Chi phí đầu tư ban đầu cao: Đầu tư thiết bị và hệ thống AGV có chi phí lớn, gây khó khăn cho doanh nghiệp nhỏ.
- Không phù hợp với nhiệm vụ không lặp lại: Hoạt động hiệu quả nhất với các nhiệm vụ lặp lại, có tuyến đường cố định; Hạn chế trong môi trường thay đổi liên tục hoặc phức tạp.
- Chi phí bảo trì cao: Yêu cầu kỹ thuật cao để bảo trì và sửa chữa; Công nghệ phức tạp dễ bị hư hỏng, gây tốn kém về thời gian và chi phí.
- Khả năng thích nghi chưa tốt với sự thay đổi của môi trường làm việc trong các điều kiện cụ thể mà AGV gặp phải

- Trên thế giới, các tập đoàn lớn như Amazon, Alibaba, Siemens, BMW, Tesla đã triển khai rất thành công hệ thống AGV trong kho bãi và nhà máy. Đơn cử, Amazon Robotics sử dụng hàng chục ngàn robot AGV để tự động hóa toàn bộ quá trình vận chuyển hàng hóa trong kho, giảm thiểu sai sót và nâng cao năng suất đáng kể. Tương tự, BMW sử dụng AGV trong dây chuyền lắp ráp linh kiện xe hơi, cho phép hệ thống tự động đưa linh kiện đến đúng vị trí, đúng thời điểm mà không cần sự can thiệp của con người. Ở các quốc gia như Nhật Bản, Hàn Quốc, Trung Quốc, AGV còn được ứng dụng trong bệnh viện, khách sạn, thậm chí trong nông nghiệp thông minh.

Hình 1. 2 AGV Robot vận chuyển hàng trong kho và bệnh viện

- Tại Việt Nam, tuy việc ứng dụng AGV chưa phổ biến rộng rãi như các nước phát triển, nhưng đã bắt đầu xuất hiện trong một số nhà máy quy mô lớn như VinFast, Samsung Việt Nam, THACO, và một số kho bãi logistics hiện đại tại TP. Hồ Chí Minh và Hà Nội. Ngoài ra, các viện nghiên cứu, trường đại học cũng đã bước đầu triển khai các đề tài nghiên cứu về AGV trong phòng thí nghiệm và các mô hình thử nghiệm nhỏ. Tuy nhiên, hầu hết các hệ thống AGV hiện nay vẫn còn nhập khẩu, hoặc mới chỉ dừng lại ở mức điều khiển cơ bản, chưa tối ưu về khả năng thích nghi trong điều kiện vận hành thực tế như thay đổi tải trọng, bề mặt di chuyển hoặc môi trường làm việc phức tạp.

Hình 1. 3 AGV Robot kéo hàng ITA – 01 của Intech Group

1.2 Các loại AGV Robot hiện nay trên thị trường

- Trong lĩnh vực công nghiệp hiện đại, AGV được phân loại dựa trên chức năng, hình thức di chuyển và phương pháp điều hướng. Một số loại phổ biến bao gồm:

+ AGV kéo hàng (Towing AGV): Có khả năng kéo theo nhiều xe hàng phía sau như đoàn tàu, thường sử dụng trong vận chuyển khối lượng lớn trong nhà máy, kho xưởng.

Hình 1. 4 AGV kéo hàng (Towing AGV)

+ AGV chở hàng (Unit Load AGV): Dùng để vận chuyển các đơn vị hàng hóa, pallet hoặc thùng chứa. Thường được trang bị hệ thống nâng để tự động lấy và đặt hàng hóa.

Hình 1. 5 AGV chở hàng (Unit Load AGV)

+ AGV dạng xe nâng (Forklift AGV): Tích hợp cơ cấu nâng như xe nâng truyền thống, tự động thực hiện nhiệm vụ nâng – hạ hàng từ giá kệ hoặc sàn.

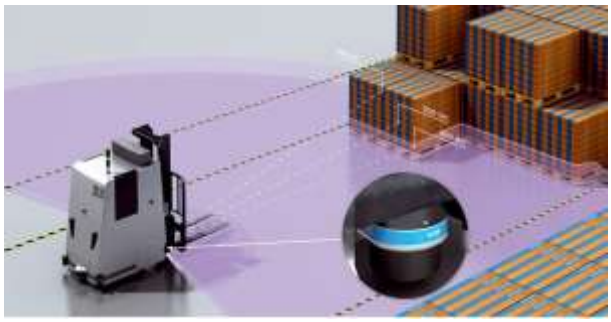
Hình 1. 6 AGV dạng xe nâng (Forklift AGV)

+ AGV dạng robot di động nhỏ (Small Mobile Robot): Thường ứng dụng trong các kho hàng tự động, có thể vận chuyển vật nhẹ hoặc linh kiện nhỏ.

Hình 1. 7 AGV dạng robot di động nhỏ (Small Mobile Robot)

+ AGV dẫn đường từ tính/quang học/LiDAR: Phân loại theo công nghệ điều hướng, gồm:

- Dẫn đường bằng vạch từ (magnetic tape)
- Dẫn đường bằng mã vạch, phản quang (QR, vision)
- Dẫn đường bằng laser, LiDAR (SLAM)
- Điều hướng dựa trên bản đồ ảo và cảm biến (AMR – Autonomous Mobile Robot)



Hình 1. 8 AGV định vị dẫn đường

1.3 AGV Robot tải hàng (AGV Robot Unit Load)

- AGV Unit Load là một dạng xe tự hành chuyên dùng để vận chuyển tải đơn vị (unit load) như pallet, khay, hộp hoặc container trong môi trường sản xuất hoặc kho vận. Thay vì kéo hoặc chở nhiều hàng hóa cùng lúc, loại AGV này thường xử lý một đơn vị tải tại một thời điểm, giúp đảm bảo độ chính xác và an toàn trong quá trình vận chuyển.

- Ưu điểm:

+ Tự động hóa cao, giảm nhu cầu lao động thủ công.

+ Chính xác và đáng tin cậy, đặc biệt khi xử lý hàng hóa giá trị cao hoặc cần vận chuyển cẩn thận.

+ Dễ tích hợp với hệ thống quản lý kho (WMS) hoặc hệ thống sản xuất (MES).

+ Có thể hoạt động 24/7, tăng hiệu suất vận hành.

- Ứng dụng:

+ Trong nhà máy sản xuất linh kiện điện tử, để vận chuyển khay linh kiện giữa các công đoạn.

+ Trong kho tự động, để di chuyển pallet từ khu vực lưu trữ đến khu vực xuất hàng.

+ Trong bệnh viện, để vận chuyển thuốc, dụng cụ y tế hoặc vật tư.

1.4 Những thách thức AGV Robot tải hàng (AGV Robot Unit Loading)

- AGV Robot tải hàng thường gặp phải những thách thức sau:

+ Khi vận chuyển hàng hóa bằng AGV Robot tải hàng hóa dễ vỡ (thủy tinh, đồ điện tử, dược phẩm trong lọ) có thể bị dao động và rung lắc

+ Cần kiểm soát vận tốc một cách ổn định liên tục.

+ Hệ số ma sát thay đổi và ảnh hưởng bám đường của bánh xe: Di chuyển qua nhiều loại bề mặt (gạch, bê tông, nhựa...) với hệ số ma sát khác nhau gây ra độ trượt bánh không đồng đều, làm hệ thống khó duy trì quỹ đạo ổn định, dẫn đến rung chấn hoặc mất cân bằng tải, ảnh hưởng đến hàng hóa bên trên.

+ Tải trọng đặt lên hệ thống AGV Robot có thể thay đổi khi thêm hoặc bớt hàng hóa.

1.5 Đề xuất giải pháp

- Dựa trên các thách thức đã nêu trong mục trước, nhóm nghiên cứu đề xuất một giải pháp điều khiển AGV sử dụng bộ điều khiển kết hợp giữa PID truyền thống và logic mờ (Fuzzy), nhằm giải quyết các vấn đề thực tế về tải trọng thay đổi, địa hình không đồng đều, khả năng vận hành ổn định và hợp lí về giá cả, ổn định tốc độ. Việc kết hợp giữa 2 phương pháp PID và logic mờ (Fuzzy) đã tận dụng được ưu điểm của 2 phương pháp này so với các bộ điều khiển khác.

- AGV Robot được thiết kế sử dụng 4 bánh xe Mecanum có khả năng di chuyển đa hướng, gia tăng độ linh hoạt và chính xác, ứng dụng hoạt động để thay đổi tải trọng.

Chương 2: NGHIÊN CỨU VÀ THIẾT KẾ AGV ROBOT

2.1 Đặt vấn đề

- Trong thực tế vận hành tại các môi trường sản xuất và kho bãi hiện đại, việc ứng dụng các hệ thống xe tự hành (AGV) nhằm tự động hóa quá trình vận chuyển vật liệu đang ngày càng phổ biến. Điều này không chỉ giúp giảm thiểu sự phụ thuộc vào lao động thủ công mà còn góp phần nâng cao hiệu suất và tính nhất quán trong hoạt động logistics. Tuy nhiên, các yếu tố thực tế như thay đổi tải trọng, điều kiện mặt sàn không đồng đều, hoặc các tình huống bất ngờ trong vận hành có thể gây ảnh hưởng lớn đến sự ổn định và hiệu quả chuyển động của hệ thống AGV.

- Đặc biệt, đối với các robot sử dụng bánh xe Mecanum – vốn có khả năng di chuyển đa hướng – thì việc điều khiển vận tốc thân xe trở nên phức tạp hơn do phải phối hợp đồng thời nhiều động cơ một cách chính xác. Sự thay đổi tải trọng hoặc ma sát bánh có thể dẫn đến sai lệch vận tốc, ảnh hưởng đến độ chính xác trong vận hành và tiềm ẩn rủi ro đối với hàng hóa, nhất là khi vận chuyển các vật dễ vỡ hoặc có giá trị cao.

2.2 Giải pháp

- Để giải quyết được bài toán điều khiển tốc độ ổn định cho AGV Robot sử dụng bánh xe Mecanum, nhóm đưa ra giải pháp nhằm nâng cao khả năng thích nghi và ổn định của hệ thống là một hệ thống gồm 2 tầng điều khiển:

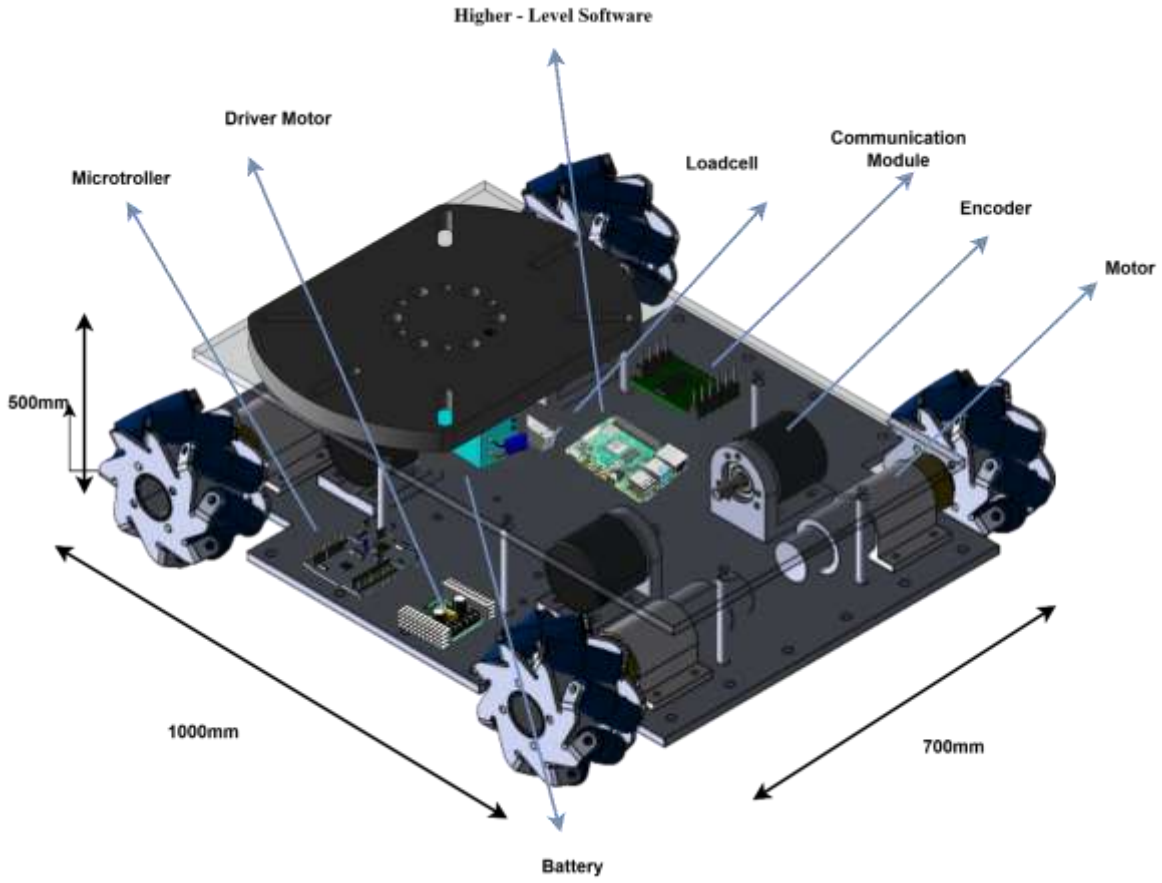
+ Tầng điều khiển đầu tiên là một bộ điều khiển mờ – PID (Fuzzy-PID), có vai trò làm nhiệm vụ điều chỉnh tốc độ tốc độ xe dựa trên các yếu tố như sai lệch tốc độ. Khác với bộ truyền PID với hệ số cố định, bộ Fuzzy-PID sử dụng hệ suy luận mờ để tự động điều chỉnh các hệ số Kp, Ki, Kd.

+ Tầng điều khiển thứ hai là các bộ PID riêng biệt cho từng động cơ, đảm bảo đảm bảo tốc độ góc của từng bánh xe góp theo tốc độ mong muốn làm tầng trên đặt ra. Với xe sử dụng bánh Mecanum, mỗi bánh xe đóng góp vào chuyển động sạch tiến trình và quay tổng thể, vì vậy việc điều khiển chính xác từng bánh là yếu tố rồi chốt để đảm bảo động chuyển thân xe ổn định.

- Việc sử dụng hệ thống điều khiển hai cấp cấu trúc vừa tận dụng tính chính xác cao của PID trong bộ điều khiển cục bộ, vừa kết hợp khả năng thích nghi và phi tuyến của logic

mờ trong điều kiện thay đổi phức tạp. Đây là hướng dẫn tiếp cận phù hợp trong bối cảnh yêu cầu cao về độ tin cậy và tính linh hoạt trong các ứng dụng AGV hiện đại, đặc biệt khi hướng đến môi trường công nghiệp có nhiều yếu tố nhiễu và không xác định.

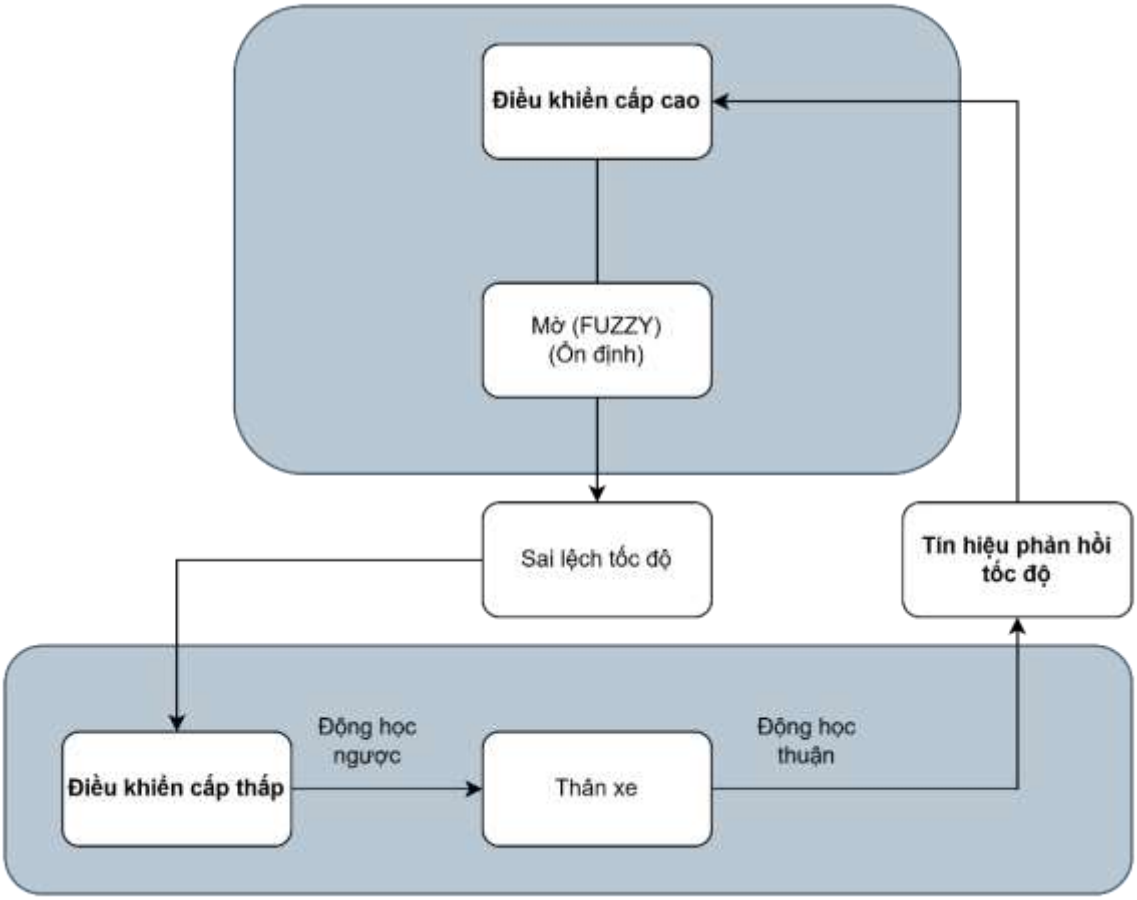
2.3 Sơ đồ công nghệ



Hình 2. 1 Sơ đồ công nghệ của hệ thống

Hệ thống xe tự hành sử dụng 4 bánh Mecanum cho phép di chuyển linh hoạt theo mọi hướng nhờ thiết kế bánh đặc biệt. Xe được trang bị các động cơ gắn trực tiếp vào từng bánh, điều khiển thông qua bộ vi điều khiển và mạch driver. Bộ vi điều khiển nhận dữ liệu phản hồi từ encoder để điều chỉnh tốc độ, đồng thời giao tiếp với phần mềm điều khiển cấp cao thông qua mô-đun truyền thông. Hệ thống còn tích hợp loadcell để đo tải trọng và pin làm nguồn cấp chính. Toàn bộ thiết kế hướng đến khả năng tự động hóa và vận hành chính xác trong môi trường công nghiệp.

2.4 Thiết kế kiến trúc điều khiển hệ thống



Hình 2. 2 Sơ đồ về cấu trúc điều khiển của hệ thống

2.3.1 Điều khiển cấp cao

- Trong đề tài này, điều khiển cấp cao có nhiệm vụ thiết lập các giá trị đầu vào điều khiển mong muốn cho xe tự hành, cụ thể là lực và vận tốc theo các phương x, y và moment quay quanh trục z. Các giá trị tham chiếu này được đặt trước (setpoint) hoặc có thể thay đổi theo kịch bản mô phỏng (ví dụ: chuyển động thẳng, rẽ trái, tăng tải, lên dốc, v.v.). Nó không xử lý bản đồ hay định vị toàn cục mà chỉ tập trung cung cấp mục tiêu vận tốc cho hệ thống điều khiển ở tầng thấp.

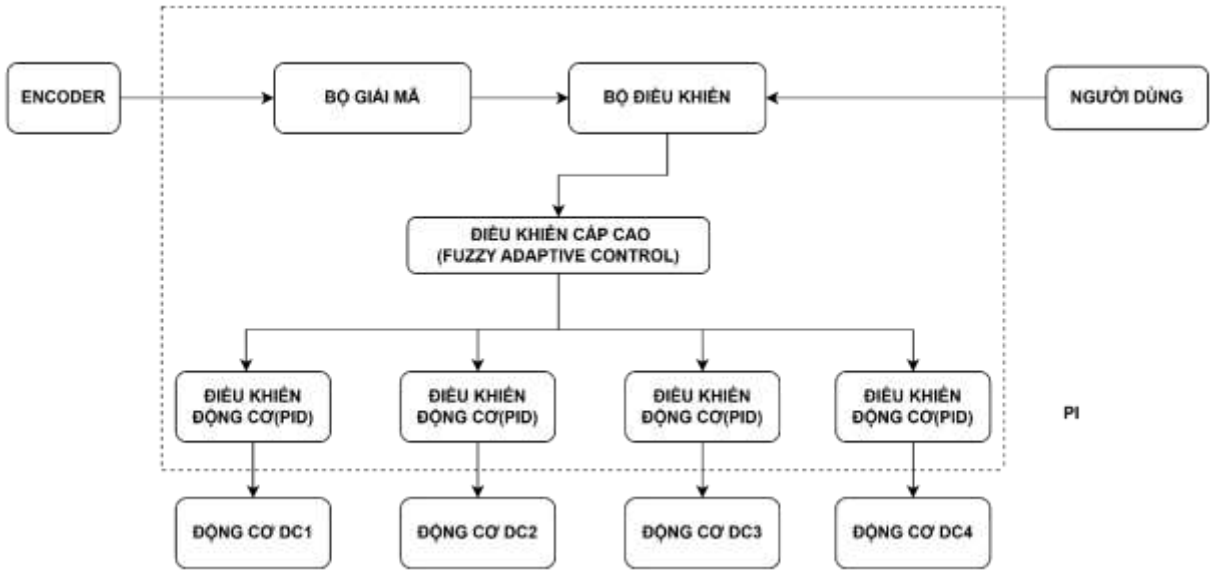
- Đặc biệt, do xe sử dụng cơ cấu bánh Mecanum, khả năng chuyển động đa hướng (omni-directional) được tận dụng để thử nghiệm khả năng ổn định trong nhiều tình huống khác nhau. Các tham số như vận tốc thân xe và moment quay được truyền trực tiếp xuống bộ điều khiển tầng thấp, nơi các thuật toán điều khiển PID và Fuzzy-PID xử lý và phân phối lực cho từng bánh xe.

- Điều khiển cấp cao được xây dựng trong môi trường MATLAB/Simulink để thuận tiện cho việc mô phỏng và đánh giá hiệu suất hệ thống. Các tín hiệu đầu vào như tải trọng, độ nghiêng mặt dốc, hoặc hệ số ma sát có thể được điều chỉnh trực tiếp trong quá trình mô phỏng. Điều này giúp đánh giá khả năng thích nghi của bộ điều khiển Fuzzy-PID khi các điều kiện vận hành thay đổi.

2.3.2 Điều khiển cấp thấp

- Ở phần này, điều khiển cấp thấp có chức năng nhận tham chiếu tín hiệu tốc độ từ tầng điều khiển cấp cao, điều khiển PID từng động cơ bánh xe dựa trên vận tốc mong muốn, tính toán động học thuận từ vận tốc bánh xe thực tế để suy ra vận tốc thực tế của xe. Sau đó, gửi thông tin tốc độ phản hồi lên tầng điều khiển cao để phục vụ quá trình điều khiển ổn định

2.4 Sơ đồ khối điều khiển



Hình 2. 3 Sơ đồ khối điều khiển của hệ thống

- ENCODER:

+ Cảm biến đo tốc độ quay của hệ thống

+ Xuất xung tín hiệu tương ứng với vận tốc hoặc vị trí quay.

+ Thông tin này dùng cho phản hồi điều khiển PID.

- BỘ GIẢI MÃ:

+ Giải mã tín hiệu từ encoder (dạng xung vuông) sang dạng số hoặc vận tốc dễ xử lý.

+ Gửi tín hiệu đã xử lý đến BỘ ĐIỀU KHIỂN để sử dụng cho PID

- BỘ ĐIỀU KHIỂN:

+ Là cầu nối giữa tầng điều khiển cao và các bộ điều khiển động cơ.

+ Nhận lệnh từ ĐIỀU KHIỂN CẤP CAO (High Level Controller) và phản hồi từ ENCODER.

+ Thực hiện các thuật toán điều khiển thấp như PID tốc độ cho từng động cơ.

+ Xuất tín hiệu PWM để điều khiển mô-men xoắn động cơ.

- NGƯỜI DÙNG:

+ Tác động vào hệ thống thông qua giao diện điều khiển.

+ Nhập điểm đến, chế độ hoạt động (manual/auto), theo dõi trạng thái AGV.

+ Giao tiếp với hệ thống qua phần mềm GUI, mobile app, hoặc web.

- ĐIỀU KHIỂN CẤP CAO (FUZZY ADAPTIVE CONTROL):

+ Thực hiện các thuật toán điều khiển thông minh như Fuzzy Logic, PID tổng thể, tính toán động học/nghịch đảo động học.

+ Nhận dữ liệu từ người dùng và cảm biến → xử lý → xuất lệnh điều khiển vận tốc từng bánh xe.

- ĐIỀU KHIỂN ĐỘNG CƠ (PID):

+ Là phần tử công suất (driver) như L298N, BTS7960, VNH2SP30, v.v.

+ Nhận tín hiệu PWM từ vi điều khiển và điều chỉnh điện áp cấp cho động cơ.

+ Có thể đảo chiều và điều chỉnh tốc độ động cơ DC.

+ Mỗi động cơ cần một bộ điều khiển riêng.

- ĐỘNG CƠ DC1 → ĐỘNG CƠ DC4:

+ Cung cấp lực quay cho từng bánh xe.

+ Nhận điện từ ĐIỀU KHIỂN ĐỘNG CƠ để quay với vận tốc mong muốn.

+ Gắn trực tiếp với các bánh xe

2.5 Thiết kế cấu trúc hệ thống AGV Robot

2.5.1 Thiết kế AGV Robot mong muốn

- AGV Robot có khả năng chịu được tải trọng, đảm bảo khả năng chở hàng với tải trọng từ 20 kg đến 30 kg, phù hợp với nhu cầu vận chuyển hàng hóa nhẹ trong nhà máy như linh kiện điện tử, thực phẩm đóng gói, thiết bị y tế, v.v.

- Khả năng thích nghi môi trường làm việc: Xe phải hoạt động ổn định trong các điều kiện thay đổi thực tế như:

+ Sàn có hệ số ma sát thay đổi (do dầu, nước, bụi...).

+ Địa hình có dốc nhẹ (khoảng 5–10 độ nghiêng).

- Kích thước và trọng lượng giới hạn: Để di chuyển linh hoạt trong không gian hẹp, kích thước tổng thể của xe được giới hạn như sau:

+ Chiều dài: ≤ 1000 mm

+ Chiều rộng: ≤ 800 mm

+ Chiều cao: ≤ 500 mm

+ Khối lượng AGV Robot: khoảng 25 kg

- Thời gian làm việc và sạc:

+ Xe cần đảm bảo khả năng vận hành liên tục 20 giờ/ngày.

+ Thời gian sạc pin: không quá 4 giờ. Có thể tích hợp sạc nhanh hoặc cơ chế thay pin.

- Sử dụng bánh xe Mecanum: Việc sử dụng bánh xe Mecanum là yêu cầu bắt buộc trong đề tài. Loại bánh xe này có các con lăn nghiêng 45 độ cho phép xe có thể:

+ Di chuyển theo mọi hướng: tiến, lùi, sang ngang, chéo, quay tại chỗ, ...

- Tiết kiệm năng lượng: Hệ thống điều khiển phải có khả năng tiết kiệm năng lượng, tự động chuyển sang chế độ nghỉ khi không hoạt động, và tối ưu hóa công suất động cơ trong quá trình di chuyển.

- Chuyển đổi chế độ điều khiển: Xe phải hỗ trợ chuyển đổi linh hoạt giữa chế độ tự động (Auto) và bằng tay (Manual) để thuận tiện trong quá trình bảo trì, điều chỉnh, hoặc xử lý các tình huống bất ngờ.

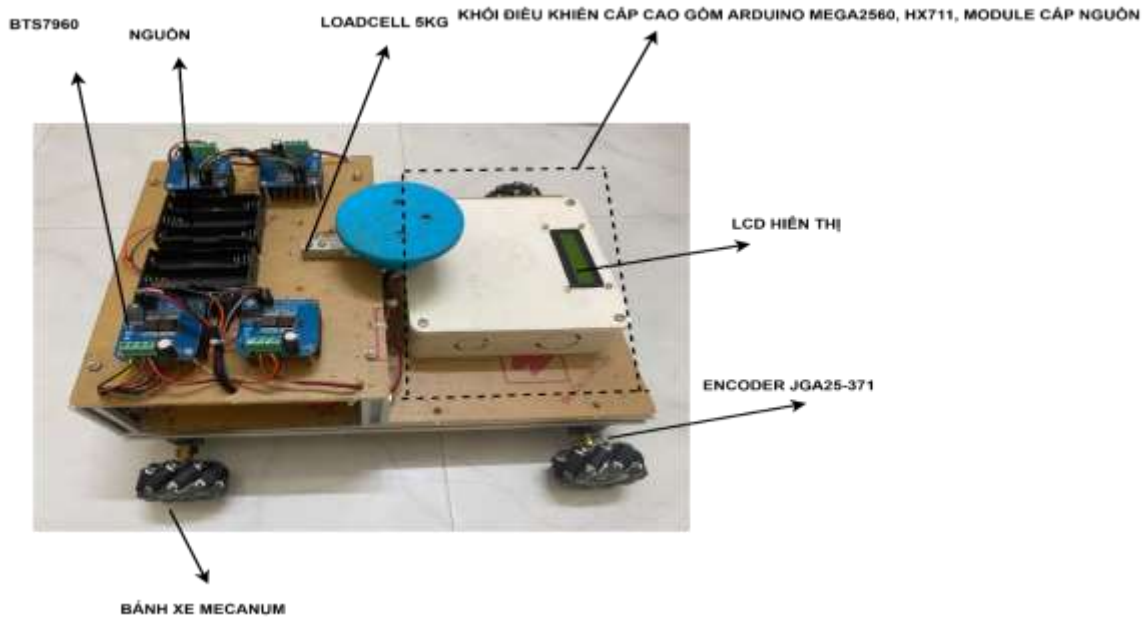
- Với các yêu cầu về kỹ thuật như trên, nhóm nghiên cứu đã tính chọn ra các thiết bị phù hợp với AGV Robot mong muốn:

Bảng 2.1 Tính chọn thiết bị cho hệ thống

Thiết bị	Mã/ loại cụ thể
Encoder	JGA25-371 12V 60RPM (Bộ mã hóa 2CH)
Loadcell	Cảm biến lực 50kg + Module HX711
Bộ xử lý cấp cao	Raspberry Pi 4 (RAM 2–4GB)
Điều khiển cấp thấp	STM32F103C8T6

2.5.2 AGV Robot trong đề tài

- Các thiết bị được tính chọn cho AGV Robot mong muốn nêu trên, dựa vào các yêu cầu thực tế đối với một hệ thống xe tự hành ứng dụng trong công nghiệp, đáp ứng tốt các yêu cầu của 1 AGV Robot Công nghiệp thực hiện được nhiều nhiệm vụ ở quy mô vừa và lớn. Còn về AGV Robot của đề tài nhóm thực hiện thì đã có sự thu hẹp lại về quy mô hệ thống bao gồm các cấp điều khiển sẽ đơn giản hơn so với AGV Robot mong muốn nhằm mục đích đơn giản hóa việc tính toán, thiết kế và gia công cũng như giảm nhẹ chi phí.



Hình 2. 4 AGV Robot trong đề tài nhóm thực hiện

Chương 3: MÔ HÌNH HÓA HỆ THỐNG VÀ THIẾT KẾ BỘ ĐIỀU KHIỂN FUZZY – PID

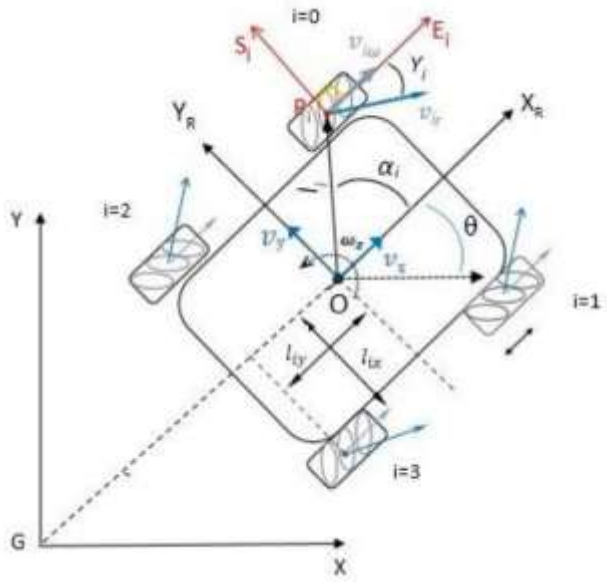
3.1 Mô hình hóa hệ thống

3.1.1 Mô hình động học của AGV Robot

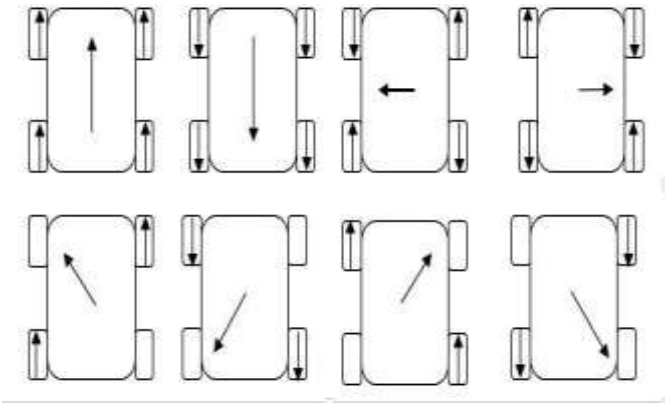
- Trong việc mô hình hóa một hệ thống tự hành sử dụng bánh xe Mecanum, việc xây dựng mô hình động học là bước đầu tiên và quan trọng để hiểu mối quan hệ giữa các động cơ mong muốn của thân xe và điều khiển tín hiệu tại mỗi bánh xe, giúp thiết lập mối quan hệ học giữa tốc độ của robot (vận tốc tuyến tính và góc quay) với tốc độ quay của từng bánh xe (vận tốc góc). Mô hình này đóng vai trò kết nối giữa bài toán điều khiển cao (lực, vận tốc thân xe) và trình độ (điều khiển từng bánh xe).

- Động học thuận (Forward Kinematics): mô tả cách tính toán vận chuyển tốc độ của thân xe dựa trên vận tốc quay của các bánh xe. Đây là mô hình dùng để xác định xe đang chuyển hướng như thế nào trong không gian (hướng, tốc độ, quay tại chỗ, vv) từ đầu vào là tốc độ của từng bánh. Phương thức học tập thuận lợi thường được sử dụng trong mô phỏng, đánh giá phản hồi thực tế từ bộ mã hóa hoặc phân tích hợp nhất vào định vị thuật toán.

- Động học ngược (Động học nghịch đảo): Ngược lại, động học ngược xác định tốc độ yêu cầu của từng bánh xe sao cho thân xe di chuyển theo một tốc độ mong muốn đã cho (ví dụ: đi thẳng, sang trái, quay tròn). Mô hình học ngược đóng vai trò quan trọng trong hệ thống điều khiển, đặc biệt khi áp dụng các bộ điều khiển như Fuzzy-PID để sinh lực hoặc chạy mục tiêu tốc độ cho thân xe.



Hình 3.1 Mô tả minh họa về tâm quay của hệ thống xe tự hành



Hình 3. 1 Chuyển động đa hướng của AGV Robot

- Đề tài nghiên cứu của nhóm sử dụng bốn bánh xe Mecanum trong dự án. Cấu trúc bố trí bánh xe giống như Hình 3.1. Hướng và tốc độ của các bánh xe chéo được điều khiển độc lập. Việc sử dụng cùng một tốc độ cho tất cả các bánh xe cùng lúc trong quá trình vận hành giúp đạt được 8 hướng chuyển động khác nhau của robot mà không làm thay đổi hướng quay của nó. Bằng cách thay đổi tốc độ của các bánh xe chéo, chúng tôi có thể đạt được chuyển động theo mọi hướng từ 0° đến 360°.

- Ví dụ, để thực hiện chuyển động ngang sang phải, các bánh xe bên phải được quay ngược chiều nhau vào trong, trong khi các bánh xe bên trái quay ngược chiều nhau ra ngoài (Hình 3.2). Bằng cách sử dụng kỹ thuật này, chúng tôi đạt được tất cả tám chuyển động khác nhau như được thể hiện trong (Hình 3.2)

3.1.1.1 Động học thuận của AGV Robot

- Ta có thể tính toán vận tốc của bánh xe i và vận tốc tiếp tuyến của con lăn tự do tiếp xúc với mặt đất theo công thức sau:

$$v_{ir} = \frac{1}{cos\,45} r_r \omega \quad (3.1)$$

$$w_{Ei} = r_i \omega_i, \; i = 0,1,2,3 \quad (3.2)$$

Và xét các phương trình *(3.1) và (3.2),* vận tốc của bánh xe i trong hệ tọa độ $S_i P_i E_i$ suy ra:

$$v_{S_i} = v_{ir}\, sin\, \gamma_i \quad (3.3)$$

$$v_{Ei} = \omega_i r_i + v_{ir}\, cos\, \gamma_i \quad (3.4)$$

$$\begin{bmatrix} v_{S_i} \\ v_{E_i} \end{bmatrix} = \begin{bmatrix} 0 & \sin\gamma_i \\ r_i & \cos\gamma_i \end{bmatrix} \begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} = {}^{w_i} T_{P_i} \begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} \quad (3.5)$$

- Ma trận biến đổi vận tốc từ bánh xe thứ i là:

$$w_i T_{P_i} = \begin{bmatrix} 0 & \sin\gamma_i \\ r_i & \cos\gamma_i \end{bmatrix} \quad (3.6)$$

- Vận tốc của tâm bánh xe được chuyển sang hệ tọa độ $X_R O Y_R$ được phương trình sau:

$$\begin{bmatrix} v_{iX_R} \\ v_{iX_Y} \end{bmatrix} = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix} \begin{bmatrix} v_{S_i} \\ v_{E_i} \end{bmatrix} = {}^{w_i} T_{P_i}{}^{P_i} T_R \begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} \quad (3.7)$$

- Sau đó, ma trận chuyển đổi từ vận tốc bánh xe sang hệ tọa độ là phương trình:

$$P_i T_{T_R} = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix} \quad (3.8)$$

Lưu ý: Vì chuyển động của robot là chuyển động phẳng (2D), nên ta có các điều kiện riêng cho việc tính toán ma trận vận tốc và các biến thiên góc trong mặt phẳng.

- Ta thu được phương trình động học thuận:

$$\begin{bmatrix} v_{iX_R} \\ v_{iX_Y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix} \begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_{X_R} \end{bmatrix} = T' \begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_{X_R} \end{bmatrix} \quad (3.9)$$

Trong đó: $T' = \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix}$

(Trích dẫn từ *Kinematic Model of a Four Mecanum Wheeled Mobile Robot – Hamid Taheri, Bing Qiao, Nurallah Ghaeminezhad)*

3.1.1.2 Động học nghịch của AGV Robot

- Từ phương trình *(3.6)* và *(3.8),* ta thu được phương trình động học ngược:

$$w_i T_{P_i}^{P_i} T_{T_R} \begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} = T' \begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_{X_R} \end{bmatrix}, i = 0,1,2,3 \qquad (3.10)$$

Vì As$r_i \neq 0, 0 < |\gamma_i| < \frac{\pi}{2}, det(P_i^T T^R) \neq 0$ và $det(\omega_i^T P_i) \neq 0$, nên bằng cách kết hợp phương trình *(3.7)* và *(3.9)* ta được:

$$\begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} = {}^{w_i} T_{P_i}^{-1} . {}^{P_i} T_{T_R}^{-1} . T' \begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_{X_R} \end{bmatrix}, i = 0,1,2,3 \qquad (3.11)$$

- Theo phương trình (3.6) và phương trình (3.7), tồn tại một mối quan hệ giữa các biến trong hệ tọa độ của từng bánh xe của xe và tâm của nó. Với bài toán động học nghịch vận tốc của xe có thể được xác định bằng cách sử dụng v_{ir} là vận tốc tuyến tính và ω_i là tốc độ quay của bánh xe thứ *i* trong phương trình *(3.12),* và ngược lại trong phương trình *(3.13):*

$$\begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_z \end{bmatrix} = T^+ \begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} \qquad (3.12)$$

$$\begin{bmatrix} \omega_i \\ v_{ir} \end{bmatrix} = T \begin{bmatrix} v_{X_R} \\ v_{Y_R} \\ \omega_{X_R} \end{bmatrix} \qquad (3.13)$$

Trong đó: $T = {}^{w_i} T_{P_i}^{-1} . {}^{P_i} T_{T_R}^{-1} . T'$, $T^+ = (T^T T)^{-1} T^T$

$$T = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix}^{-1} . \begin{bmatrix} 0 & \sin\gamma_i \\ r_i & \cos\gamma_i \end{bmatrix}^{-1} . \begin{bmatrix} 1 & 0 & -l_{iy} \\ 0 & 1 & l_{ix} \end{bmatrix} \qquad (3.14)$$

Xét $l_{ix} = l_i \cos\alpha_i$ và$l_{iy} = l_i \sin\alpha_i$ (các bánh xe có cùng kích thước), ma trận chuyển đổi sẽ là:

$$T = \frac{1}{-r}\begin{bmatrix} \frac{cos(\beta_i - y_i)}{sin(y_i)} & \frac{sin(\beta_i - y_i)}{sin(y_i)} & \frac{l_i\, sin(-\alpha_i + \beta_i - y_i)}{sin(y_i)} \\ -\frac{r\, cos(\beta_i)}{sin(y_i)} & -\frac{r\, sin(\beta_i)}{sin(y_i)} & -\frac{l_i\, sin(-\alpha_i + \beta_i) r}{sin(y_i)} \end{bmatrix} \quad (3.15)$$

$$T^{+} = \frac{1}{li^2 + 1}\begin{bmatrix} -\frac{1}{2}\left(li^2 \sin(\beta_i) - li^2 \sin(-\beta_i + 2\alpha_i) + 2\sin(\beta_i)\right) & \frac{1}{2} li^2 \sin(-y_i + \beta_i + 2\alpha_i) - \frac{1}{2}\sin(-y_i + \beta_i) li^2 - \sin(-y_i + \beta_i) \\ \frac{1}{2} r\left(li^2 \cos(\beta_i) - li^2 \cos(-\beta_i + 2\alpha_i) + 2\cos(\beta_i)\right) & -\frac{1}{2} li^2 \cos(y_i - \beta_i + 2\alpha_i) + \frac{1}{2}\cos(-y_i + \beta_i) li^2 + \cos(-y_i + \beta_i) \\ \cos(\alpha_i - \beta_i) & \cos(\alpha_i - \beta_i + y_i) li \end{bmatrix} \quad (3.16)$$

- ω_i và v_{ir} có mối quan hệ độc lập trong mỗi khớp và vận tốc góc và vận tốc tuyến tính của xe, động học nghịch đảo có phương trình sau:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{-1}{r}\begin{bmatrix} \frac{cos(\beta_1 - \gamma_1)}{sin\,\gamma_1} & \frac{sin(\beta_1 - \gamma_1)}{sin\,\gamma_1} & \frac{l_1\, sin(\beta_1 - \gamma_1 - \alpha_1)}{sin\,\gamma_1} \\ \frac{cos(\beta_2 - \gamma_2)}{sin\,\gamma_2} & \frac{sin(\beta_2 - \gamma_2)}{sin\,\gamma_2} & \frac{l_2\, sin(\beta_2 - \gamma_2 - \alpha_2)}{sin\,\gamma_2} \\ \frac{cos(\beta_3 - \gamma_3)}{sin\,\gamma_3} & \frac{sin(\beta_3 - \gamma_3)}{sin\,\gamma_3} & \frac{l_3\, sin(\beta_3 - \gamma_3 - \alpha_3)}{sin\,\gamma_3} \\ \frac{cos(\beta_4 - \gamma_4)}{sin\,\gamma_4} & \frac{sin(\beta_4 - \gamma_4)}{sin\,\gamma_4} & \frac{l_4\, sin(\beta_4 - \gamma_4 - \alpha_4)}{sin\,\gamma_4} \end{bmatrix}\begin{bmatrix} v_X \\ v_y \\ \omega_z \end{bmatrix} \quad (3.17)$$

và ma trận Jacobian cho động học nghịch đảo của hệ thống:

$$T = -\frac{1}{r}\begin{bmatrix} \frac{cos(\beta_1 - \gamma_1)}{sin\,\gamma_1} & \frac{sin(\beta_1 - \gamma_1)}{sin\,\gamma_1} & \frac{l_1\, sin(\beta_1 - \gamma_1 - \alpha_1)}{sin\,\gamma_1} \\ \frac{cos(\beta_2 - \gamma_2)}{sin\,\gamma_2} & \frac{sin(\beta_2 - \gamma_2)}{sin\,\gamma_2} & \frac{l_2\, sin(\beta_2 - \gamma_2 - \alpha_2)}{sin\,\gamma_2} \\ \frac{cos(\beta_3 - \gamma_3)}{sin\,\gamma_3} & \frac{sin(\beta_3 - \gamma_3)}{sin\,\gamma_3} & \frac{l_3\, sin(\beta_3 - \gamma_3 - \alpha_3)}{sin\,\gamma_3} \\ \frac{cos(\beta_4 - \gamma_4)}{sin\,\gamma_4} & \frac{sin(\beta_4 - \gamma_4)}{sin\,\gamma_4} & \frac{l_4\, sin(\beta_4 - \gamma_4 - \alpha_4)}{sin\,\gamma_4} \end{bmatrix} \quad (3.18)$$

và với động học thuận theo phương trình *(3.12),* ta được:

$$\begin{bmatrix} v_X \\ v_y \\ \omega_z \end{bmatrix} = T^+ \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (3.19)$$

Bảng 3. 1 Tham số hệ thống

i	Bánh xe	α_i	β_i	y_i	l_i	l_{ix}	l_{yx}
0	1	$\frac{\pi}{4}$	$\frac{\pi}{2}$	$-\frac{\pi}{4}$	l	l_x	l_y
1	2	$-\frac{\pi}{4}$	$-\frac{\pi}{2}$	$\frac{\pi}{4}$	l	l_x	l_y
2	3	$\frac{3\pi}{4}$	$\frac{\pi}{2}$	$\frac{\pi}{4}$	l	l_x	l_y
3	4	$-\frac{3\pi}{4}$	$-\frac{\pi}{2}$	$-\frac{\pi}{4}$	l	l_x	l_y

- Bằng cách thay các thông số trong Bảng 3.1 vào ma trận phương trình *(3.17)* và phương trình *(3.18)* ta thua được:

$$T = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_{ix}+l_{iy}) \\ 1 & 1 & (l_{ix}+l_{iy}) \\ 1 & 1 & -(l_{ix}+l_{iy}) \\ 1 & -1 & (l_{ix}+l_{iy}) \end{bmatrix} \quad (3.20)$$

$$T^+ = \frac{r}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \quad (3.21)$$

- Theo các phương trình *(3.12)* và *(3.13)* về động học thuận và động học ngược ta có:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_x+l_y) \\ 1 & 1 & (l_x+l_y) \\ 1 & 1 & -(l_x+l_y) \\ 1 & -1 & (l_x+l_y) \end{bmatrix}\begin{bmatrix} v_X \\ v_y \\ \omega_z \end{bmatrix} \quad (3.22)$$

$$\begin{cases} \omega_1 = \frac{1}{r}(v_x - v_y - (l_x + l_y)\omega), \\ \omega_2 = \frac{1}{r}(v_x + v_y + (l_x + l_y)\omega), \\ \omega_3 = \frac{1}{r}(v_x + v_y - (l_x + l_y)\omega), \\ \omega_4 = \frac{1}{r}(v_x - v_y + (l_x + l_y)\omega), \end{cases} \quad (3.23)$$

Và

$$\begin{bmatrix} v_X \\ v_y \\ \omega_z \end{bmatrix} = \frac{r}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (3.24)$$

- Vận tốc theo trục x:

$$v(x) = (\omega_1 + \omega_2 + \omega_3 + \omega_4)\frac{r}{4} \quad (3.25)$$

- Vận tốc theo trục y:

$$v(y) = (-\omega_1 + \omega_2 + \omega_3 - \omega_4)\frac{r}{4} \quad (3.26)$$

- Vận tốc góc:

$$\omega_z(t) = (-\omega_1 + \omega_2 - \omega_3 + \omega_4)\frac{r}{4(l_x+l_y)} \quad (3.27)$$

- Vận tốc tổng hợp và góc định hướng trong hệ tọa độ đứng yên (x, y, z) có thể được tính toán như sau:

+ Hướng chuyển động (góc):

$$\rho = tan^{-1}\left(\frac{v_y}{v_x}\right) \quad (3.28)$$

+ Độ lớn của vận tốc tổng hợp:

$$v_R = \sqrt{{v_x}^2 + {v_y}^2} \quad (3.29)$$

- Phương trình động học của hệ:

$$\omega = T \times^{P_i} {T_{T_R}}^T \times q_f \quad (3.30)$$

Trong đó:

$\boldsymbol{\omega} = [\boldsymbol{\omega_1} \quad \boldsymbol{\omega_2} \quad \boldsymbol{\omega_3} \quad \boldsymbol{\omega_4}]^{\boldsymbol{T}}$

$$T = \frac{1}{r}\begin{bmatrix} 1 & -1 & -(l_{ix}+l_{iy}) \\ 1 & 1 & l_{ix}+l_{iy} \\ 1 & 1 & -(l_{ix}+l_{iy}) \\ 1 & -1 & l_{ix}+l_{iy} \end{bmatrix}$$

$$P_i T_{T_R} = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix}$$

$\dot{q}_f = [\dot{x} \quad \dot{y} \quad \dot{\varphi}]^T = [v_x \quad v_y \quad \omega_z]^T$ với ω_z là vận tốc góc của hệ thống, $\dot{q}_f$ là vector vị trí tại tâm hình học của hệ thống xe tự hành trong hệ quy chiếu XGY.

(Trích dẫn từ *Kinematic Model of a Four Mecanum Wheeled Mobile Robot – Hamid Taheri, Bing Qiao, Nurallah Ghaeminezhad)*

3.1.2 Mô hình động lực học của AGV Robot

- Phương trình động lực học:

$$E\tau = [MJ({}^{B}\dot{T}_{T_R}{}^{T}\dot{q} + {}^{B}T_{T_R}{}^{T}\ddot{q}) - Hf_{Lms} - Kf_{ms}] \qquad (3.31)$$

Trong đó:

$$M = \begin{bmatrix} A+B+J_{ob} & -B & A-B & B \\ -B & A+B+J_{ob} & B & A-B \\ A-B & B & A+B+J_{ob} & -B \\ B & A-B & -B & A+B+J_{ob} \end{bmatrix} \qquad (3.32)$$

Với:

- $A = \frac{1}{8}Mr^2; B = \frac{Jr^2}{4(2l_{ix}+2l_{iy})}$
- J_{ob} là mô-men quán tính quay quanh trục z (tức là phương vuông góc với mặt phẳng chuyển động) của toàn bộ robot khi xét đến chuyển động quay.
- $M = m_b + 4m_w$: là khối lượng của toàn bộ hệ thống

  m_b, m_w là lượt là khối lượng của thân hệ thống và các bánh xe

- $J = J_{GB} + m_w(4{l_{ix}}^2 + 4{l_{iy}}^2) + J_{GW}$ là momen quán tính của hệ thống xe tự hành

  Trong đó: J_{GB} là Mô-men quán tính của thân robot (body) quanh trục z

  J_{GW} Mô-men quán tính của bánh xe quanh trục quay của nó.

$$E = \begin{bmatrix} 1-a & a & -b & b \\ a & 1-a & b & -b \\ -b & b & 1-a & a \\ b & -b & a & 1-a \end{bmatrix} \qquad (3.33)$$

Với: $a = \frac{r(2l_{ix}+4l_{iy})}{4\sqrt{2}r_{OL}(2l_{ix}+2l_{iy})}$ $b = \frac{2rl_{ix}}{4\sqrt{2}r_{OL}(2l_{ix}+2l_{iy})}$, r_{OL} là khoảng cách từ trục con lăn đến trục bánh xe

$$H = \begin{bmatrix} -\frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & \frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & -\frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & \frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} \\ \frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & -\frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & \frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & -\frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} \\ -\frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & \frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & -\frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & \frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} \\ \frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & -\frac{2rl_{ix}}{4(2l_{ix}+2l_{ix})} & \frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} & -\frac{r(2l_{ix}+4l_{iy})}{4(2l_{ix}+2l_{ix})} \end{bmatrix} \quad (3.34)$$

$$K = [-r \quad -r \quad -r \quad -r]^T \quad (3.35)$$

$$\tau = [\tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4]^T \quad (3.36)$$

là momen quán tính điều khiển của các bánh xe

$$f_{Lms} = [f_{Lms1} \quad f_{Lms2} \quad f_{Lms3} \quad f_{Lms4}]^T \quad (3.37)$$

là lực ma sát lăn của con lăn trên bánh xe

$$f_{ms} = [f_{ms1} \quad f_{ms2} \quad f_{ms3} \quad f_{ms4}]^T \quad (3.38)$$

là lực ma sát lăn của các bánh xe

Với: $F_{msi} = N_i\mu_i\, sgn\, \omega_i$ và $F_{msi} = N_i\mu_i\, sgn\, \omega_{Li}$ lần lượt là các thành phần ma sát tại điểm tiếp xúc của con lăn và các bánh xe đối với mặt đường, N_i, μ_i lần lượt là phản lực từ mặt đường tác dụng lên các bánh xe và hệ số ma sát lăn

- Từ (3.30) và (3.31) suy ra:

$$E\tau = MJ^{P_i}{T_{T_R}}^T \ddot{q} + MJ^{P_i}\dot{T}_{T_R}^T \dot{q} - G\,\mathrm{sgn}(\dot{q}) \quad (3.39)$$

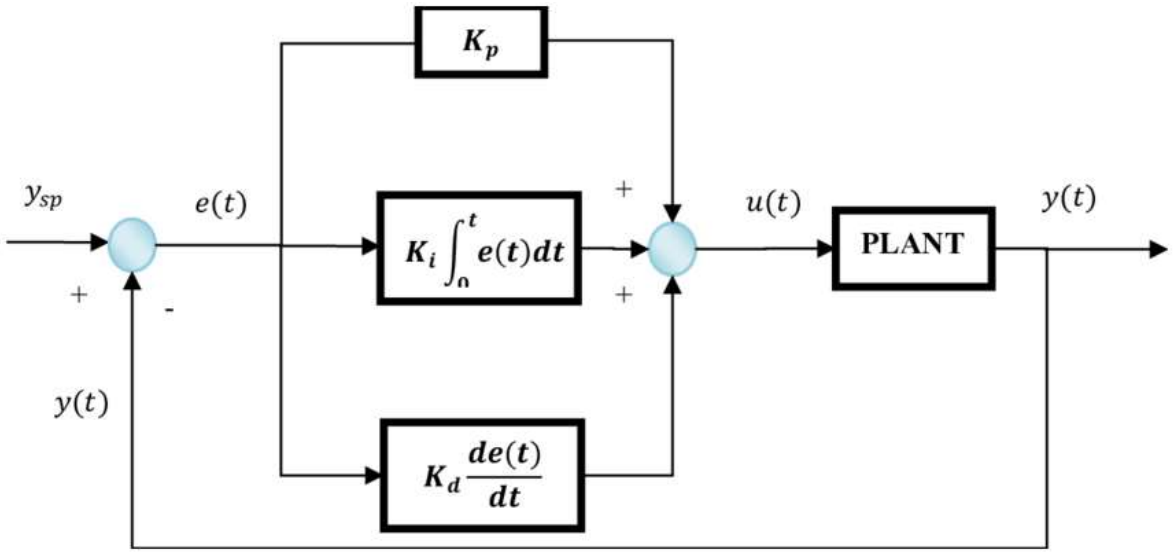
Với: $\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\varphi}]^T = [v_x + v_y + \omega_z]^T$ là vận tốc của hệ thống

$$G = (H - K)J^{P_i}{T_{T_R}}^T N\mu \quad (3.40)$$

(Trích dẫn từ *Trajectory tracking control for the mecanum wheel mobile robot the dynamic surface controller connected to the neural network – Nguyễn Thị Thanh, Võ Thu Hà)*

3.2 Thiết kế bộ điều khiển

3.2.1 Thiết kế bộ điều khiển PID của động cơ DC



Hình 3. 2 Sơ đồ khối bộ điều khiển PID

Trong đó:

- Thành phần P (Tỉ lệ) xử lý sai số hiện tại.
- Thành phần I (Tích phân) xử lý tổng sai số theo thời gian, giúp loại bỏ sai số tĩnh.
- Thành phần D (Vi phân) phản ánh tốc độ thay đổi của sai số, giúp cải thiện đáp ứng động.

- Trong hệ thống xe tự hành, mỗi bánh được dẫn động độc lập bởi một động cơ DC, cho phép xe có khả năng di chuyển linh hoạt theo nhiều hướng. Bốn động cơ này có cùng loại, cùng đặc tính kỹ thuật và cùng yêu cầu điều khiển về tốc độ, do đó quá trình thiết kế bộ điều khiển có thể thực hiện cho một động cơ. Cấu trúc điều khiển là tương tự và áp dụng cho ba động cơ còn lại.

- Phương trình động lực của động cơ:

$$R_a i_a(t) + L_a \frac{di_a(t)}{dt} + E_b = e_a(t) \quad (4.5)$$

$$J_m \frac{d^2\theta_m}{dt^2} + D_m \frac{d\theta_m}{dt} = T_m(t) \quad (4.6)$$

$$T_m = K_t i_a(t) \quad (4.7)$$

$$E_b(t) = K_b \frac{d\theta_m}{dt} \quad (4.8)$$

+ Chuyển sang miền Laplace:

$$R_a I_a(s) + L_a s I_a(s) + K_b s\theta_m(s) = E_a(s) \quad (4.9)$$

$$J_m s^2 \theta_m(s) + D_m s\theta_m = K_t I_a(s) \quad (4.10)$$

+ Hàm truyền đạt:

$$G(s) = \frac{V(s)}{E_a(s)} = \frac{K_t}{L_a J_m s^2 + (R_a J_m + L_a D_m)s + (R_a D_m + K_b K_t)} \quad (4.11)$$

Trong đó:

R_a là điện trở phần ứng

L_a là điện cảm cuộn dây

J_m là momen quán tính ($kg.m^2$)

D_m là hệ số ma sát

K_t là hằng số momen (Nm/A)

K_b là hằng số điện áp ($V/(rad/s)$)

- Dựa vào datasheet của động cơ Encoder JGA25-371 ta có các thông số kĩ thuật cần thiết sau:

+ $L_a = 0.1214$ H

+ $R_a = 12\ \Omega$

+ $J_m = 0.0495$ kg.cm^2

+ $D_m = 0.002953$

+ $K_t = 0.412$ Nm/A

+ $K_b = 0.91$ V/(rad/s)

- Ta có hàm truyền của động cơ DC:

$$G(s) = \frac{0.412}{0.006009s^2 + 0.5944s + 0.4104}$$

- Tiến hành thiết kế bộ điều khiển PID cho động cơ DC:

+ Ta có phương trình sau:

$$1 + PI(z)G(z) = 0 \qquad (4.12)$$

$$1 + \left(K_P + \frac{K_I T}{2}\frac{z+1}{z-1}\right)\frac{0.06076z + 0.006673}{z^2 - 0.9329z + 5.059e-05} = 0$$

$$\Rightarrow 2(z^2 - 0.9329z + 5.059 \times 10^{-5})(z-1) + (2K_P(z-1) + K_I T(z + 1))(0.06076z + 0.006673) = 0$$

$$\Rightarrow (2z^3 - 3.8658z^2 + 1.8659z - 0.00010118)$$
$$+(0.12152K_P z^2 - 0.108174K_P z - 0.013346K_P + 0.06076K_I T z^2 + 0.067433K_I T z + 0.006673K_I T) = 0$$

$$\Rightarrow 2z^3 + (-3.8658 + 0.12152K_P + 0.06076K_I T)z^2$$
$$+(1.8659 - 0.108174K_P + 0.067433K_I T)z + (-0.00010118 - 0.013346K_P + 0.006673K_I T) = 0$$

$$\Rightarrow 2z^3 + (0.12152K_P + 0.06076K_I T - 3.8658)z^2$$
$$+(-0.108174K_P + 0.067433K_I T + 1.8659)z + (-0.013346K_P + 0.006673K_I T - 0.00010118) = 0$$

+ Với các tiêu chuẩn ổn định, ta có:

$\zeta = 0.707; \omega_n = 10\ \text{rad/s}; \text{T} = 0.1\text{s}$

$z_{1,2}^* = re^{\pm j\emptyset}$

$r = e^{-T\zeta\omega_n} = e^{-0.1\times0.707\times10} = 0.493$

$\emptyset = T\omega_n\sqrt{1-\zeta^2} = 0.1 \times 10 \times \sqrt{1-0.707^2} = 0.707$

$z_{1,2}^* = r \times (cos\,\emptyset \pm j\,sin\,\emptyset)$

$= \text{rcos}\emptyset \pm \text{rjsin}\emptyset$

$= 0.493\text{cos}(0.707) \pm \text{j}0.493\text{sin}(0.707)$

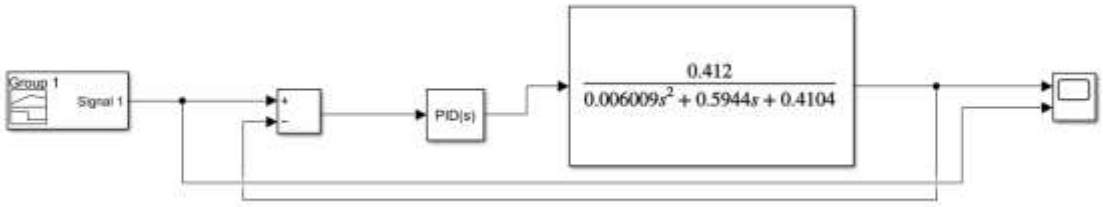
$= 0.375 \pm \text{j}0.32$

Lấy $z_3^* = 1$

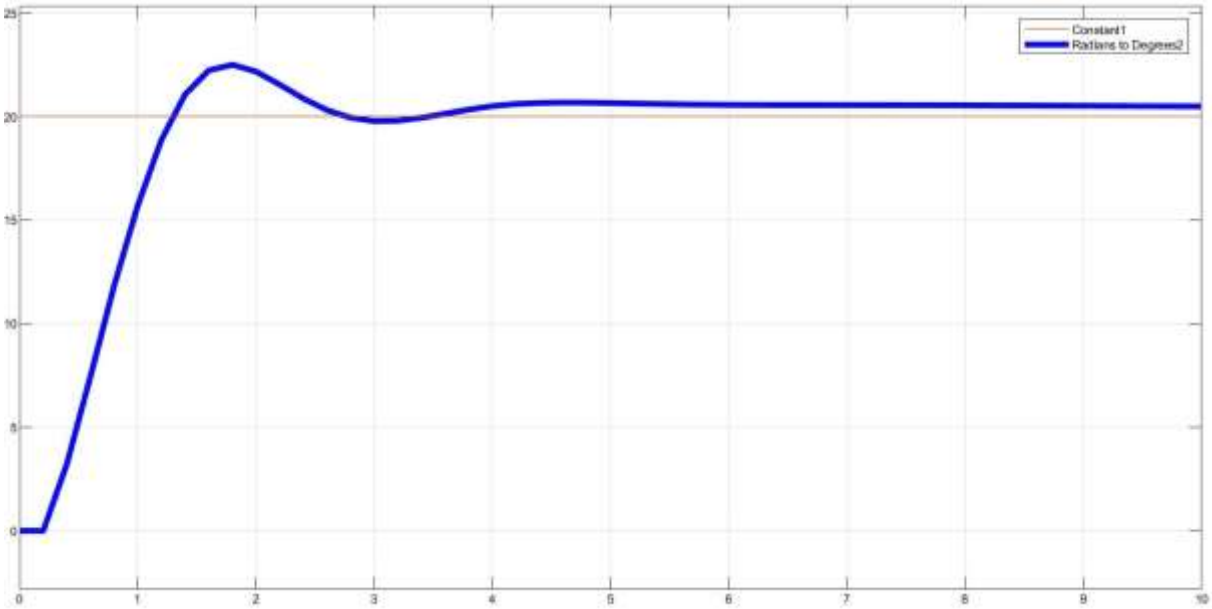
$(z - z_1^*)(z - z_2^*)(z - z_3^*)$

$$= (z^2 - (z_1^* + z_2^*)z + z_1^* z_2^*)(z - z_3^*)$$
$$= z^3 - z_3^* z^2 - (z_1^* + z_2^*)z^2 + (z_1^* + z_2^*)z_3^* z + z_1^* z_2^* z - z_1^* z_2^* z_3^*$$
$$= z^3 - (z_1^* + z_2^* + z_3^*)z^2 + [(z_1^* + z_2^*)z_3^* + z_1^* z_2^*]z - z_1^* z_2^* z_3^*$$
$$= z^3 - (z_1^* + z_2^* + 1)z^2 + [(z_1^* + z_2^*) + z_1^* z_2^*]z - z_1^* z_2^*$$
$$\rightarrow z^3 - 1.7500z^2 + 0.9933z - 0.2433 = 0$$
$$\rightarrow 2z^3 - 3.5000z^2 + 1.9866z - 0.4866 = 0$$
$$\begin{cases} 0.12152K_P + 0.06076K_I T - 3.8658 = -3.5000 \\ -0.108174K_P + 0.067433K_I T + 1.8659 = 1.9866 \\ -0.013346K_P + 0.006673K_I T - 0.00010118 = -0.4866 \end{cases}$$
$$K_P = 1.173$$
$$K_I = 36.74$$

+ Mô phỏng trên Matlab – Simulink:



Hình 3. 3 Kết quả mô phỏng bộ điều khiển PID của động cơ DC



Hình 3. 4 Kết quả mô phỏng bộ điều khiển PID động cơ DC

Nhận xét và đánh giá kết quả: Kết quả mô phỏng hệ thống điều khiển phản hồi PID cho cơ sở DC cho thấy khả năng điều khiển ổn định và hiệu quả. Khi tác động tín hiệu điều khiển dưới dạng cấp bậc, hệ thống nhanh chóng đưa tốc độ đầu ra đạt tới giá trị với thời gian quá ngắn (khoảng 1 giây). Sau thời gian quá độ, hệ thống nhanh chóng đạt trạng thái ổn định, không xuất hiện dao động dư và sai số xác lập gần như bằng không.

3.2.2 Thiết kế bộ điều khiển Fuzzy – PID

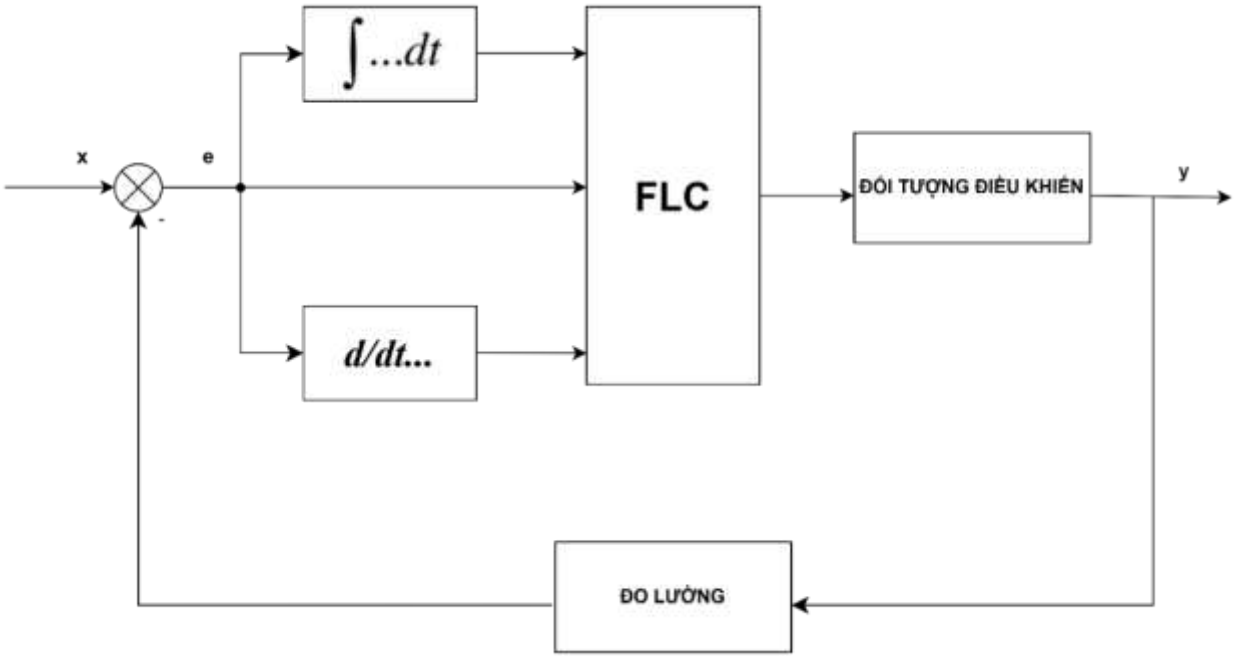
3.2.2.1 Lý thuyết về Logic mờ

3.2.2.1.1 Tổng quan về logic mờ

- Logic mờ được thể hiện bằng ngôn ngữ con người. Dựa trên logic mờ, bộ điều khiển mờ chuyển đổi một chiến lược điều khiển ngôn ngữ thành một chiến lược điều khiển và các quy tắc mờ được xây dựng bởi kinh nghiệm của một chuyên gia hoặc cơ sở dữ liệu.

- Sai số (e) và sự thay đổi của sai số (de) của tốc độ là các đầu vào biến đổi của bộ điều khiển logic mờ. Đầu ra biến đổi của bộ điều khiển logic mờ điều chỉnh các tham số của bộ điều khiển PID, sau đó bộ điều khiển PID tính toán đầu ra điều khiển.

- Cấu trúc tổng quát của một hệ điều khiển mờ:



Hình 3. 5 Cấu trúc tổng quát của một hệ mờ

3.2.2.1.2 Các thành phần của một hệ suy luận mờ (FIS)

3.2.2.1.2.1 Biến ngôn ngữ

- Biến ngôn ngữ là biến không có giá trị số cụ thể mà được diễn tả bằng ngôn ngữ tự nhiên.

- Giá trị ngôn ngữ là các từ. Giá trị ngôn ngữ chứa đựng thông tin không chính xác, do đó có thể mô tả giá trị ngôn ngữ bằng tập mờ.

- Ví dụ:

Biến nhiệt độ có thể nhận giá trị *"lạnh", "ấm", "nóng"* thay vì số độ C cụ thể.

Biến tốc độ có thể nhận giá trị *"chậm", "trung bình", "nhanh"* thay vì km/h cụ thể.

Xét tốc độ của xe, ta có thể phát biểu xe đang chạy:

+ Rất chậm (VS)

+ Chậm (S)

+ Trung bình (M)

+ Nhanh (F)

+ Rất nhanh (VF)

Những phát biểu như vậy được gọi là biến ngôn ngữ của tập mờ. Gọi x là giá trị của biến tốc độ, ví dụ $x = 10km/h$, $x = 60km/h \ldots$ Hàm liên thuộc tương ứng của các biến ngôn ngữ trên được kí hiệu là:

$$\mu_{VS}(x), \mu_{S}(x), \mu_{M}(x), \mu_{F}(x), \mu_{VF}(x)$$

- Biến tốc độ sẽ có 2 miền giá trị:

+ Miền giá trị ngôn ngữ:

N = {rất chậm, chậm, trung bình, nhanh, rất nhanh}

+ Miền giá trị vật lý:

$$V = \{x \in B \mid x \geq 0\}$$

- Biến tốc độ được xác định trên miền ngôn ngữ N được gọi là biến ngôn ngữ

- Với mỗi $x \in B$ ta có hàm liên thuộc:

$$x \to \mu_x = \{\mu_{VS}(x), \mu_S(x), \mu_M(x), \mu_F(x), \mu_{VF}(x)\}$$

3.2.2.1.2.2 Các dạng hàm liên thuộc

- Các dạng hàm liên thuộc (membership function) trong logic mờ: Gaussian, PI-shape, S-shape, Sigmoidal, Z-shape…

3.2.2.1.2.3 Các phép toán trên tập mờ

- Phép giao: giao của hai tập mờ X và Y có cùng tập nền B là một tập mờ $X \cap Y$ cũng xác định trên nền cơ sở B có hàm liên thuộc thỏa mãn $\mu_{X\cap Y}(x)$ thỏa mãn các tính chất sau:

a) $\mu_{X\cap Y}(x)$ chỉ phụ thuộc vào $\mu_X(x)$ và $\mu_Y(x)$

b) Nếu có $\mu_X(x) = 1$, với mọi x thì cũng có $\mu_{X\cap Y}(x) = \mu_X(x)$

c) $\mu_{X\cap Y}(x) = \mu_{Y\cap X}(x)$, tức là có tính chất giao hoán

d) $\mu_{(X\cap Y)\cap Z}(x) = \mu_{X\cap(Y\cap Z)}(x)$, tức là có tính chất kết hợp

e) Nếu có $\mu_{X1}(x) \le \mu_{X2}(x)$ thì cũng có $\mu_{X1\cap Y}(x) \le \mu_{X2\cap Y}(x)$, tức là có tính chất không giảm

Ta có thể thấy được nhiều công thức khác nhau được dùng để thực hiện tính toán hàm liên thuộc $\mu_{X\cap Y}(x)$ cho hai tập mờ. Ví dụ như 5 công thức sau đều sử dụng để định nghĩa hàm liên thuộc $\mu_{X\cap Y}(x)$ của phép giao hai tập mờ:

Bảng 3. 2 Một số phép giao thường dùng (phép hội*)*

STT	Tên gọi	Công thức mô tả
1	Luật Min	$\mu_{X\cap Y}(x) = \min\{\mu_X(x), \mu_Y(x)\}$
2	prod (tích)	$\mu_{X\cap Y}(x) = \mu_X(x)\mu_Y(x)$
3	Tích yếu	$\mu_{X\cap Y} = \begin{cases} \min\{\mu_X(x), \mu_Y(x)\} if \max\{\mu_X(x), \mu_Y(x)\} = 1 \\ 0 if \max\{\mu_X(x), \mu_Y(x)\} \neq 1 \end{cases}$
4	Lukasiewicz	$\mu_{X\cap Y}(x) = \{0, \mu_X(x) + \mu_Y(x) - 1\}$
5	Tích Einstein	$\mu_{X\cap Y}(x) = \dfrac{\mu_X(x)\mu_Y(x)}{2 - (\mu_X(x) + \mu_Y(x)) - \mu_X(x)\mu_Y(x)}$

- Phép hợp: hợp của hai tập mờ X và Y có cùng tập nền B là một tập mờ $X \cup Y$ cũng xác định trên nền cơ sở B có hàm liên thuộc thỏa mãn $\mu_{X\cup Y}(x)$ thỏa mãn các tính chất sau:

f) $\mu_{X\cup Y}(x)$ chỉ phụ thuộc vào $\mu_X(x)$ và $\mu_Y(x)$

g) Nếu có $\mu_X(x) = 0$, với mọi x thì cũng có $\mu_{X\cup Y}(x) = \mu_X(x)$

h) $\mu_{X\cup Y}(x) = \mu_{Y\cup X}(x)$, tức là có tính chất giao hoán

i) $\mu_{(X\cup Y)\cup Z}(x) = \mu_{X\cup(Y\cup Z)}(x)$, tức là có tính chất kết hợp

k) Nếu có $\mu_{X1}(x) \le \mu_{X2}(x)$ thì cũng có $\mu_{X1\cup Y}(x) \le \mu_{X2\cup Y}(x)$, tức là có tính chất không giảm

Ta có thể thấy được nhiều công thức khác nhau được dùng để thực hiện tính toán hàm liên thuộc $\mu_{X\cup Y}(x)$ cho hai tập mờ. Ví dụ như 5 công thức sau đều sử dụng để định nghĩa hàm liên thuộc $\mu_{X\cup Y}(x)$ của phép hợp hai tập mờ:

Bảng 3. 3 Một số phép hợp thường dùng (phép tuyển)

STT	Tên gọi	Công thức mô tả
1	Luật Max	$\mu_{X\cup Y}(x) = \max\{\mu_X(x), \mu_Y(x)\}$
2	Tổng trực tiếp	$\mu_{X\cup Y}(x) = \mu_X(x) + \mu_Y(x) - \mu_X(x)\mu_Y(x)$
3	Tổng yếu	$\mu_{X\cup Y} = \begin{cases} \max\{\mu_X(x), \mu_Y(x)\} if \max\{\mu_X, \mu_Y\} = 0 \\ 0 if \min\{\mu_X, \mu_Y\} \neq 0 \end{cases}$
4	Lukasiewicz	$\mu_{X\cup Y}(x) = \min\{1, \mu_X(x) + \mu_Y(x)\}$
5	Tổng Einstein	$\mu_{X\cup Y}(x) = \frac{\mu_X(x) + \mu_Y(x)}{1 + \mu_X(x) + \mu_Y(x)}$

3.2.2.1.2.4 Luật hợp thành

- Định lý Mamdani:

"Độ phụ thuộc của kết luận không được lớn hơn độ phụ thuộc điều kiện"

Nếu hệ thống có nhiều đầu vào và nhiều đầu ra thì mệnh đề suy diễn có dạng tổng quát như sau:

If $N = n_i$ *and* $M = m_i$ *and* … *Then* $R = r_i$ *and* $K = k_i$ *and* …

- Suy luận mờ: phép suy luận mờ hình thành dựa trên khi lý thuyết logic mờ được ứng dụng trong điều khiển, tương tự với phép tính kéo theo ở lý thuyết logic kinh điển khi biểu diễn mối quan hệ X → Y giữa hai tập X và Y thì phép tính này cũng được chuyển sang phép tính kéo theo của hai tập mờ không cần có chung một tập nền nhưng vẫn phải tính toán một cách máy móc để thực hiện mệnh đề hợp thành.

+ Định nghĩa phép suy luận mờ khi kết hợp tính chất kéo theo kinh điển và đề xuất Mamdani: Hai hàm thuộc $\mu_{Xk}(x)$, $\mu_{Yk}(x)$ của hai tập mờ X_k và Y_k là một tập mờ cùng nền với giá trị ngôn ngữ Yk có hàm liên thuộc $\mu_{A_k \Rightarrow B_k}(y)$ thỏa mãn:

a) $\mu_{X_k \Rightarrow Y_k}(y) \le \mu_{X_k}(x)$

b) Nếu có $\mu_{Y_k}(y) = 0$ thì cũng có $\mu_{X_k \Rightarrow Y_k}(y) = 0$

c) Nếu có $\mu_{X_k}(x) \le \mu_{X_i}(x)$ thì cũng có $\mu_{X_k \Rightarrow Y}(y) \le \mu_{X_i \Rightarrow Y}(x), \forall Y$

d) Nếu có $\mu_{X_k}(x) \le \mu_{X_i}(x)$ thì cũng có $\mu_{X \Rightarrow Y_k}(y) \le \mu_{X \Rightarrow Y_i}(x), \forall X$

Bảng 3. 4 Luật hợp thành

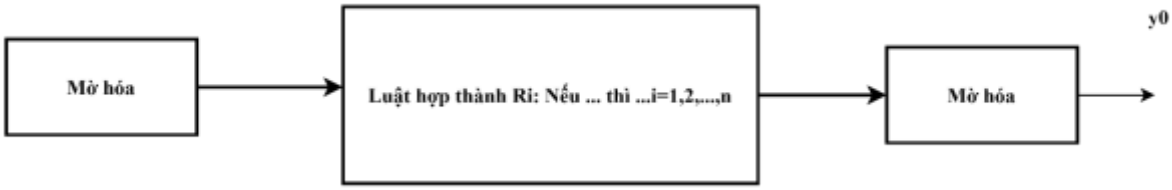
Luật hợp thành	**Phép hợp (phép tuyển)**	**Phép giao (phép hội)**	**Phép suy luận mờ**
max – min	*Luật max*	*Luật min*	*Luật min*
max – prod	*Luật max*	*Luật prod*	*Luật min*
sum – min	*Tổng trực tiếp*	*Luật min*	*Luật min*
sum – prod	*Tổng trực tiếp*	*Luật prod*	*Luật min*

3.2.2.1.2.5 Giải mờ

- Từ một giá trị rõ x_0 đầu vào, sau khi qua khối luật hợp thành ta có tập mờ đầu ra Y' và cần phải xác định rõ giá trị đầu ra y_0

- Quá trình giải mờ chính là quá trình để ta thực hiện tìm ra giá trị đầu ra y_0, giá trị này có thể chấp nhận được từ hàm liên thuộc $\mu_{Y'}(y)$ của tập mờ Y'.

- Có 2 phương pháp giải mờ chính: phương pháp cực đại và phương pháp điểm trọng tâm



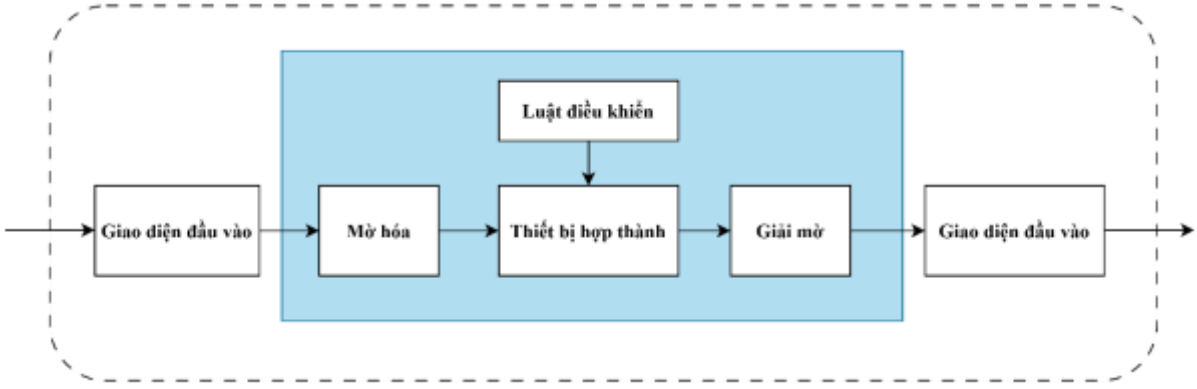
Hình 3. 6 Nhiệm vụ của khâu giải mờ

3.2.2.2 Điều khiển mờ

3.2.2.2.1 Cấu trúc của bộ điều khiển mờ

Hình 3. 7 Cơ sở toán học của phương pháp điều khiển mờ

- Một bộ điều khiển mờ bao gồm 3 khối cơ bản: khối mờ hóa, khối thiết bị hợp thành và khối giải mờ. Ngoài ra còn có khối giao diện vào và khối giao diện ra.



Hình 3. 8 Các khối chứng năng của bộ điều khiển mờ

+ Khối mờ hóa: có chức năng chuyển mỗi giá trị rõ của biến ngôn ngữ đầu vào thành vector μ có số phần tử bằng số tập mờ đầu vào.

+ Khối thiết bị hợp thành: bản chất của nó là sự triển khai luật hợp thành R được xây dựng dựa trên cơ sở của luật điều khiển.

+ Khối giải mờ: có chức năng chuyển tập mờ đầu ra thành giá trị rõ y_0 (ứng với mỗi giá trị rõ x_0 để điều khiển đối tượng).

+ Khối giao diện đầu vào: có chức năng thực hiện việc tổng hợp và chuyển đổi tín hiệu vào (từ tín hiệu tương tự sang tín hiệu số), ngoài ra còn có thể có thêm các khâu phụ trợ để thực hiện bài toán như tích phân, vi phân…

+ Khối giao diện đầu ra: có chức năng thực hiện chuyển đổi tín hiệu ra (từ tín hiệu số sang tín hiệu tương tự) để điều khiển đối tượng.

Nguyên tắc tổng hợp bộ điều khiển mờ hoàn toàn dựa vào những phương pháp toán học dựa trên cơ sở định nghĩa các biến ngôn ngữ vào/ra và sự lựa chọn những luật điều khiển. Do các bộ điều khiển mờ có khả năng xử lý các giá trị vào/ra biểu diễn dưới dạng dấu phẩy động với độ chính xác cao nên chúng hoàn toàn đáp ứng được các yêu cầu của một bài toán điều khiển "rõ ràng" và "chính xác".

3.2.2.2.2 Tổng hợp bộ điều khiển mờ

- Các bước tổng hợp bộ điều khiển mờ:

+ Khảo sát đối tượng: định nghĩa các biến vào – ra và miền xác định của chúng.

+ Mờ hóa các biến ngôn ngữ vào – ra: ở bước này ta cần xác định số lượng tập mờ và hình dạng các hàm liên thuộc cho mỗi biến ngôn ngữ, số lượng các tập mờ và hình dạng các hàm liên thuộc được chọn tùy ý. Tuy nhiên nếu chọn quá ít thì việc điều chỉnh sẽ không mịn còn nếu chịn quá nhiều thì sẽ khó khăn khi cài đặt luật hợp thành, quá trình tính toán lâu, hệ thống dễ mất ổn định. Hình dạng các hàm liên thuộc có thể chọn các hình tam giác, hình thang, hàm Gauss…

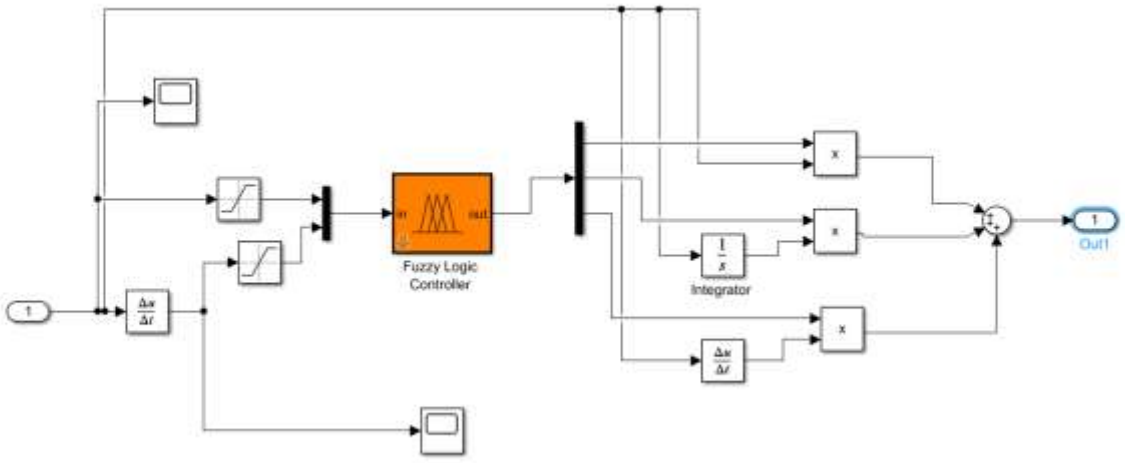
+ Xây dựng các luật điều khiển (mệnh đề hợp thành): đây là bước quan trọng và khó khăn nhất trong quá trình thiết kế bộ điều khiển mờ. Việc xây dựng luật điều khiển phụ thuộc rất nhiều vào tri thức và kinh nghiệm vận hành hệ thống của các chuyên gia.

+ Chọn các thiết bị hợp thành (MIN – PROD hoặc MAX – PROD hoặc SUM – MIN hoặc SUM – PROD) và chọn nguyên tắc giải mờ (trung bình, cận trái, cận phải, điểm trọng tâm, độ cao)

+ Tối ưu hệ thống: sau khi thiết kế xong bộ điều khiển mờ, ta cần mô hình hóa và mô phỏng hệ thống để kiểm tra kết quả, đồng thời chỉnh định lại một số tham số để có chế độ làm việc tối ưu. Các tham số có thể điều chỉnh trong bước này là: thêm, bớt luật điều khiển; thay đổi trọng số các luật; thay đổi hình dạng và miền xác định của hàm liên thuộc.

3.2.2.3 Thiết kế bộ điều khiển Fuzzy – PID cho hệ thống

- Bộ điều khiển Fuzzy-PID là sự kết hợp giữa điều khiển mờ và PID truyền thống nhằm cải thiện khả năng thích nghi và độ ổn định cho hệ thống phi tuyến hoặc có nhiễu. Cấu trúc của bộ điều khiển này gồm hai phần chính: bộ mờ (Fuzzy logic) và bộ PID. Trong đó, bộ mờ nhận đầu vào là sai số và đạo hàm sai số, sau đó xử lý qua quá trình mờ hóa, suy luận theo luật IF–THEN và giải mờ để điều chỉnh các tham số Kp, Ki, Kd của bộ PID. Bộ PID sau đó tính toán tín hiệu điều khiển dựa trên các tham số đã được hiệu chỉnh. Nhờ cấu trúc này, Fuzzy-PID có khả năng tự điều chỉnh linh hoạt và phản ứng tốt với điều kiện hoạt động thay đổi.



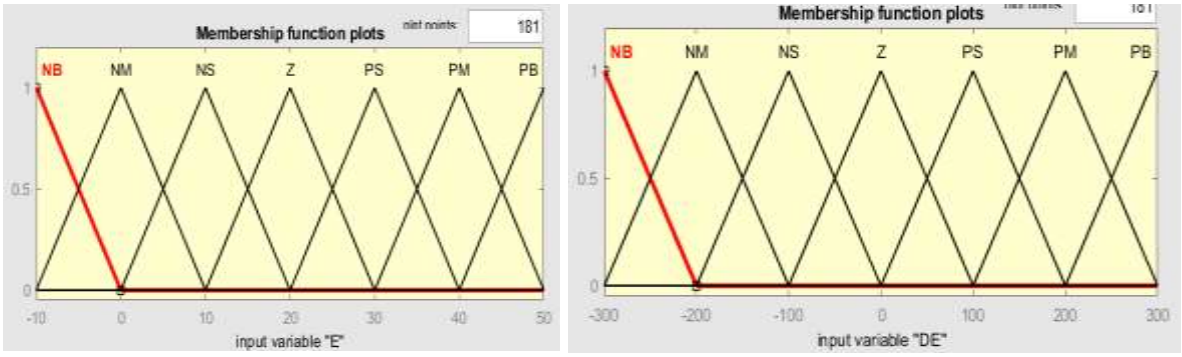
Hình 3. 9 Cấu trúc bộ điều khiển Fuzzy của hệ thống trên Matlab – Simulink

- Bộ điều khiển Fuzzy lấy sai số tốc độ và đạo hàm sai số tốc độ để điều chỉnh hệ số Kp, Ki, Kd. Từ đó, các hệ số Kp, Ki, Kd thay đổi theo thời gian được đưa vào bộ điều khiển PID tính toán ra giá trị điều khiển.

- Các biến ngôn ngữ của hệ thống: Tập mờ có 7 giá trị đầu vào của sai số vận tốc(E) và đạo hàm sai số vận tốc(dE):

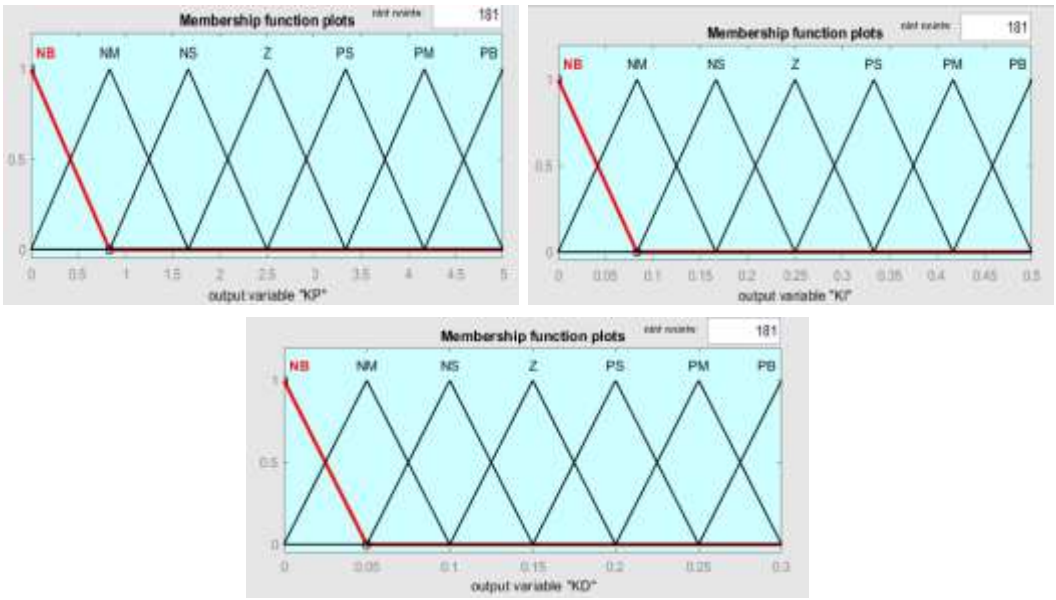
- **NB**: Âm lớn
- **NM**: Âm vừa

- **NS**: Âm nhỏ
- **Z**: Không
- **PS**: Dương nhỏ
- **PM**: Dương vừa
- **PB**: Dương lớn



Hình 3. 10 Các biến ngôn ngữ và giá trị đầu vào

Tương tự cũng có 7 giá trị đầu ra cho mỗi hệ số Kp, Ki, Kd



Hình 3. 11 Các biến ngôn ngữ và giá trị đầu ra

** Luật điều khiển Mờ PID (Fuzzy - PID):*

- Luật điều khiển cho hệ số Kp:

+ Mục tiêu của Kp là làm hệ phản hồi nhanh hơn khi sai số lớn nhưng nếu Kp quá lớn hệ dễ bị giao động (không ổn định).

+ Sai số càng lớn thì tăng Kp để đáp ứng nhanh hơn. Trường hợp khi tăng Kp:

- Khi sai số vận tốc lớn, ví dụ: bánh sau bên phải bị kẹt → vận tốc thấp → e âm lớn.
- Hệ thống cần tăng lực kéo nhanh để bánh đó đuổi kịp các bánh khác.

+ Sai số gần 0 thì giữ hoặc giảm Kp

Bảng 3. 5 Luật điều khiển Kp

E \ dE	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
NB	PB	PB	PM	PM	PS	Z	Z
NM	PB	PM	PM	PS	Z	Z	NS
NS	PM	PM	PS	Z	NS	NM	NM
Z	PM	PS	Z	Z	Z	NS	NM
PS	PS	Z	NS	Z	NS	NM	NM
PM	Z	NS	NM	NM	NM	NM	NB
PB	Z	NM	NM	NM	NB	NB	NB

Ví dụ 1:

e = NB, **de = NB** → Sai số rất âm và đang tăng nhanh theo hướng âm.

- Điều này cho thấy hệ đang lệch mạnh và nhanh khỏi giá trị mong muốn.
- Giải pháp: Tăng mạnh Kp để hệ phản ứng ngay và đưa sai số về lại → **ΔKp = PB**.

Ví dụ 2:

- **e = Z**, **de = Z** → Sai số nhỏ, hệ ổn định.
- Không cần điều chỉnh mạnh, tránh làm hệ dao động → **ΔKp = Z** (không đổi).

Ví dụ 3:

- **e = PB**, **de = PB** → Hệ đang tăng nhanh về sai số dương.
- Giống trường hợp vượt quá, nên giảm Kp để tránh tăng quá đà → **ΔKp = NB** hoặc **NM**.

+Tóm lại:

- Sai số càng lớn → tăng Kp.
- Sai số nhỏ, gần 0 → giảm Kp để tránh dao động.
- Đạo hàm lớn → giảm Kp để tránh phản ứng thái quá khi sai số biến động nhanh.

- Luật điều khiển cho hệ số Ki:

+ Hệ số tích lũy sai số theo thời gian giúp loại bỏ sai số tĩnh

+ Tuy nhiên, nếu Ki quá lớn, sai số tích lũy quá nhanh → dễ gây dao động mạnh và trễ.

+ Trường hợp cần giảm Ki:

- Khi xe đang dao động liên tục, bánh vượt quá – lùi lại – rồi vượt tiếp.
- dE lớn (sai số biến động mạnh) → giảm Ki để tránh tích lũy quá nhiều.

+ Khi điều khiển:

- Khi hệ đang dao động mạnh (de lớn), giảm Ki để tránh tích lũy thêm.
- Khi hệ ổn định nhưng còn sai số nhỏ, tăng Ki để triệt tiêu sai số lâu dài.

Bảng 3. 6 Luật điều khiển Ki

E\ dE	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
NB	NB	NB	NM	NM	NS	Z	Z
NM	NB	NM	NM	NS	Z	Z	PS
NS	NM	NM	NS	Z	PS	PM	PM
Z	NM	NS	Z	Z	Z	PS	PM
PS	NS	Z	PS	Z	PS	PM	PM
PM	Z	PS	PM	PM	PM	PM	PB
PB	Z	PM	PM	PM	PB	PB	PB

Ví dụ 1:

- **e = Z**, **de = Z**: Hệ ổn định, sai số nhỏ → có thể tăng nhẹ Ki để loại bỏ sai số nhỏ đó → **ΔKi = Z** hoặc **PS**.

Ví dụ 2:

- **e = NB**, **de = NB**: Sai số âm lớn đang tăng → nên giảm Ki mạnh để tránh tích lũy quá mức → **ΔKi = NB**.

Ví dụ 3:

- **e = PB**, **de = Z**: Sai số dương lớn, không đổi → nên tăng Ki để triệt tiêu sai số tĩnh → **ΔKi = PB**.

+ Tóm lại:

- Hệ đang dao động mạnh → giảm Ki.
- Hệ ổn định nhưng còn sai số → tăng Ki.
- Hệ ổn định hoàn toàn → giữ nguyên Ki.

- Luật điều khiển cho hệ số Kd:

+ Hệ số dự đoán sai số, phản ứng sớm để giảm vượt quá và dao động

+ Kd hoạt động như bộ "phanh" – nếu thấy sai số đang tăng nhanh, nó giảm điều khiển để tránh quá đà.

+ Trường hợp cần tăng hệ số Kd

- Khi sai số đang tăng nhanh (de lớn), nghĩa là bánh bắt đầu chệch rất nhanh.
- Tăng Kd để phản ứng sớm → không để sai số tiếp diễn.

+ Khi điều khiển:

- Nếu de lớn → tăng Kd để phản ứng sớm (chống dao động).
- Nếu de nhỏ → giảm Kd để hệ không quá chậm hoặc bị triệt tiêu tín hiệu điều khiển.

Bảng 3. 7 Luật điều khiển Kd

E\ dE	**NB**	**NM**	**NS**	**Z**	**PS**	**PM**	**PB**
NB	PS	Z	NS	NM	NM	NB	NB
NM	Z	Z	NM	NM	NB	NB	NB
NS	NS	NM	NB	NB	NB	NB	NB
Z	NS	NM	NB	Z	PB	PM	PS
PS	PS	PM	PB	PB	PB	PB	PB
PM	PM	PB	PB	PB	PB	PB	PB
PB	PB	PB	PB	PB	PB	PB	PB

Ví dụ 1:

- **e = Z**, **de = PB**: Sai số đang thay đổi rất nhanh về dương → tăng Kd để phanh → **ΔKd = PB**.

Ví dụ 2:

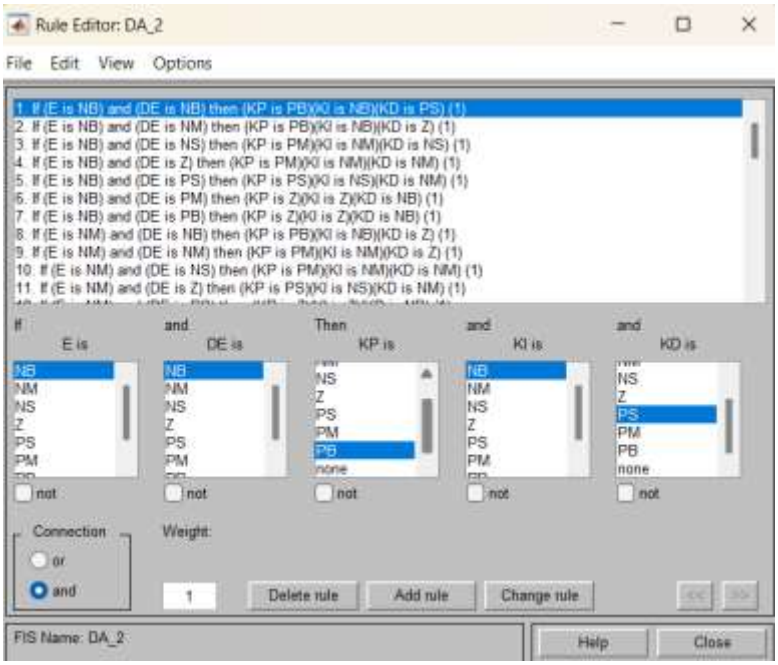
- **e = NS**, **de = Z**: Sai số nhỏ, ổn định → không cần vi phân mạnh → **ΔKd = NB** hoặc **Z**.

Ví dụ 3:

- **e = NB**, **de = NB**: Hệ đang chuyển động mạnh ngược lại mong muốn → tăng nhẹ Kd để ổn định → **ΔKd = PS** hoặc **Z**.

Tóm lại:

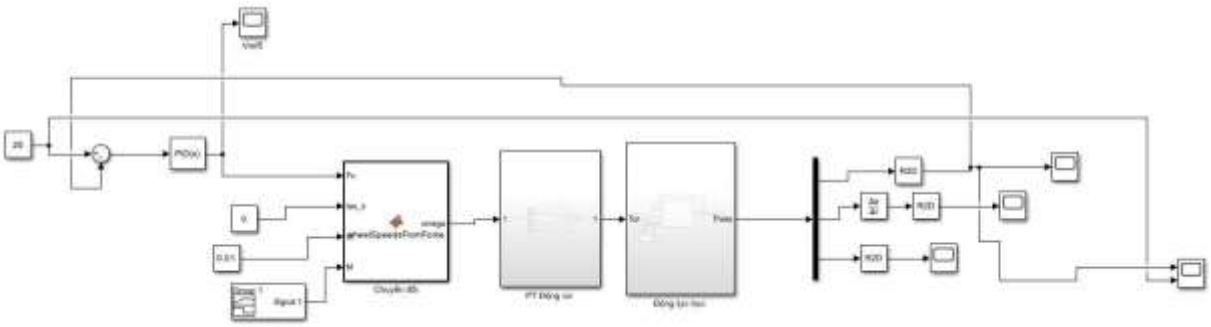
- Dao động mạnh (de lớn) → tăng Kd.
- Hệ ổn định (de nhỏ) → giảm hoặc giữ Kd nhỏ.



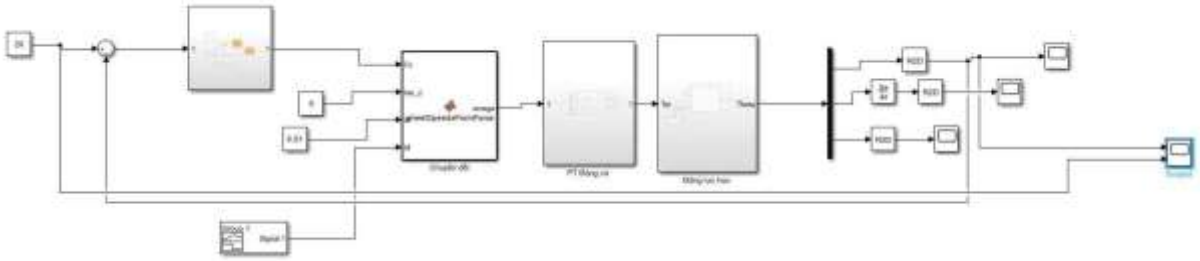
Hình 3. 12 Luật hợp thành của bộ điều khiển Fuzzy

- Trong bộ điều khiển Fuzzy-PID, luật hợp thành Mamdani được sử dụng để thực hiện quá trình suy diễn mờ, giúp hệ thống điều chỉnh linh hoạt và thích nghi tốt với các điều kiện vận hành thay đổi. Phương pháp này sử dụng các luật dạng "Nếu... thì..." (IF–THEN) để mô tả mối quan hệ giữa các biến vào (như sai số và đạo hàm sai số) và biến ra (như hệ số điều chỉnh Kp, Ki, Kd). Quá trình suy diễn gồm các bước: mờ hóa biến vào, áp dụng luật bằng phép min để xác định mức độ kích hoạt, tổng hợp đầu ra bằng phép max và giải mờ để đưa ra giá trị điều khiển thực. Với khả năng mô phỏng gần đúng suy luận của con người, luật Mamdani đặc biệt hiệu quả trong các hệ thống điều khiển phi tuyến và phức tạp như điều khiển ổn định tốc độ cho xe tự hành sử dụng bánh Mecanum.

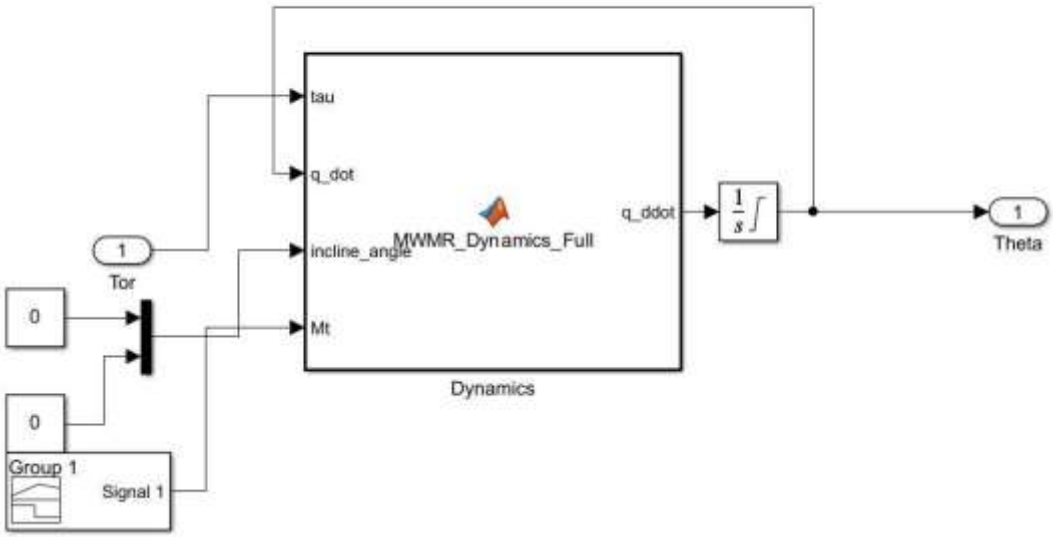
** Mô phỏng và đánh giá kết quả:*



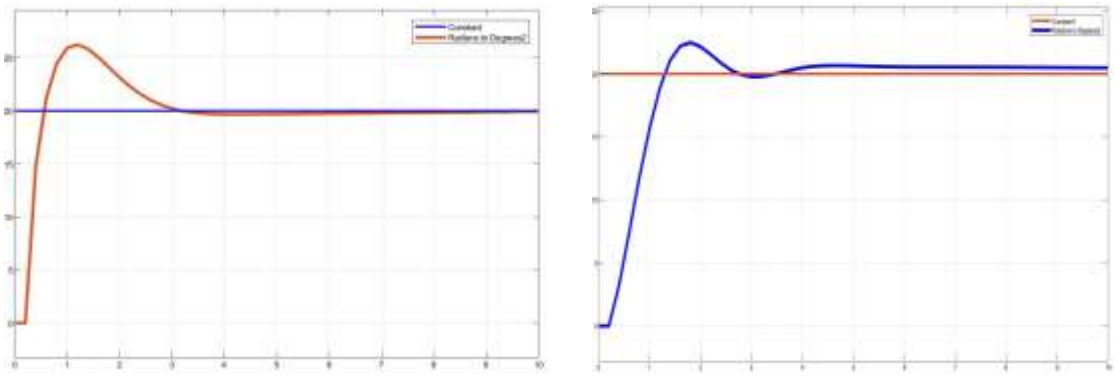
Hình 3. 13 Bộ điều khiển PID



Hình 3. 14 Bộ điều khiển Fuzzy – PID của hệ thống



Hình 3. 15 Động lực học của hệ thống



Hình 3. 16 Kết quả mô phỏng giữa 2 bộ điều khiển PID và Fuzzy – PID

- Nhận xét:

+ So sánh kết quả:

Bảng 3. 8 Bảng so sánh kết quả giữa 2 phương pháp

Bộ điều khiển / Đánh giá	PID	Fuzzy - PID
Độ vọt lố	Khá lớn	Cải thiện đáng kể hơn
Thời gian xác lập	Tương đối lâu để ổn định và duy trình ở giá trị mong muốn, dao động lớn	Đạt được trạng thái ổn định nhanh hơn, ít dao động
Sai số xác lập	Sai số xác lập bằng 0 sau khi ổn định	Sai số xác lập bằng 0 sau khi ổn định

+ Kết luận: Từ hai biểu đồ, có thể thấy bộ điều khiển Fuzzy-PID (bên phải) cho hiệu suất vượt trội hơn so với bộ điều khiển PID kinh điển (bên trái) trong trường hợp này, đặc biệt là ở các khía cạnh nêu trên. Bộ điều khiển Fuzzy-PID thể hiện hiệu quả hơn trong việc điều khiển hệ thống, mang lại đáp ứng nhanh hơn, ít vọt lố hơn và ổn định hơn so với bộ điều khiển PID kinh điển trong các điều kiện được thể hiện qua hai biểu đồ này. Điều này cho thấy sự kết hợp của logic mờ (Fuzzy Logic) có thể cải thiện đáng kể hiệu suất của bộ điều khiển PID truyền thống, đặc biệt là trong việc xử lý các hệ thống phi tuyến tính hoặc có độ bất định cao.

- Khi thay đổi tải trọng đặt lên hệ thống, bộ điều khiển Fuzzy - PID, các hệ thống PID được điều chỉnh theo thời gian thực dựa trên luật mờ đầu vào là: sai lệch tốc độ (e), đạo hàm sai số (de/dt).

Ví dụ:

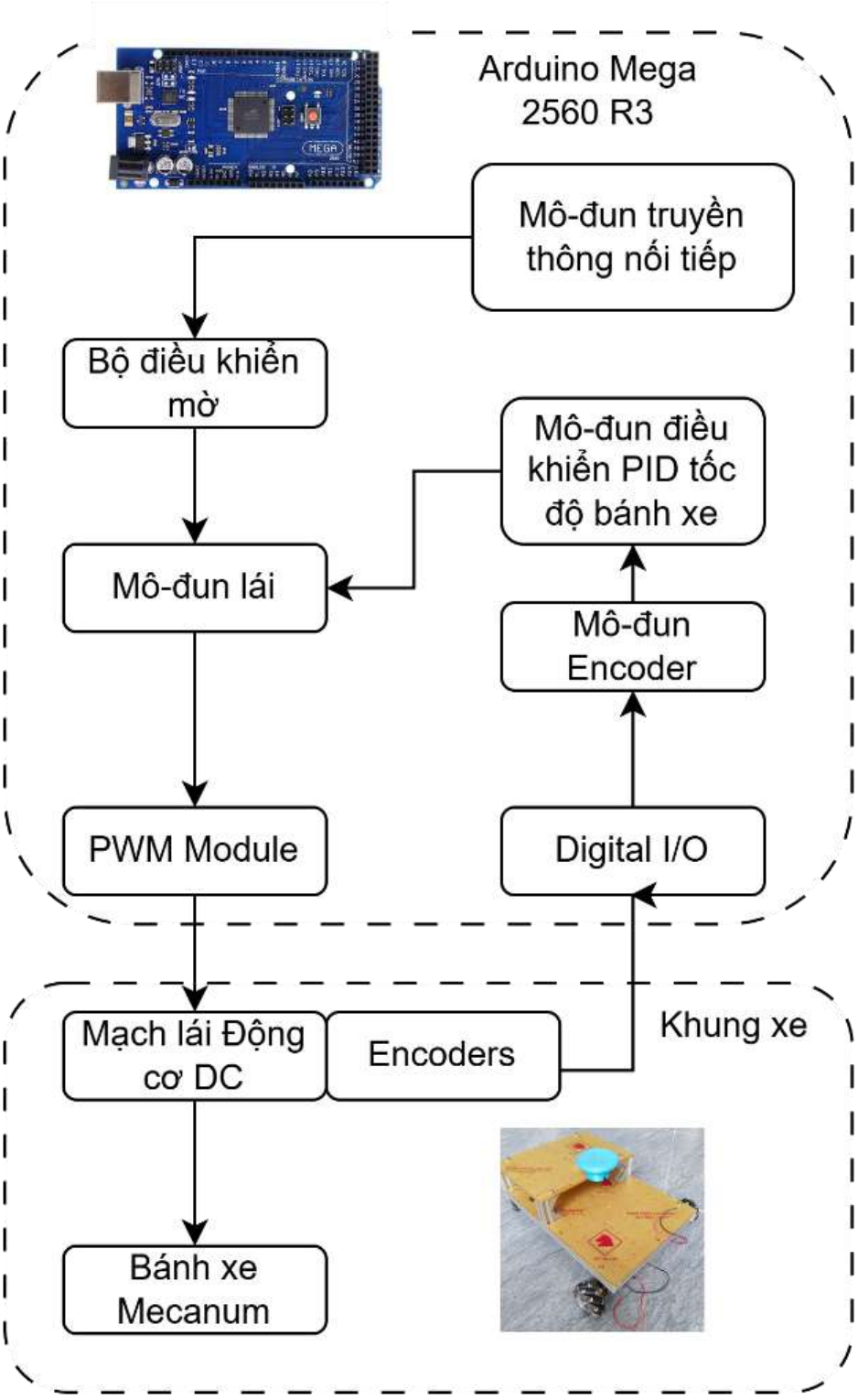
- Khi xe bắt đầu tăng thêm tải trọng, hệ thống nhận thấy vận tốc giảm đột ngột → sai số lớn và đạo hàm âm → hệ thống mờ tăng Kp và Ki tức thời → tác động mạnh hơn đến tín hiệu điều khiển → giúp cơ nhanh chóng vượt qua tính năng và ổn định trở lại.
- Ngược lại, khi giảm tải (tháo hàng xuống), nếu hệ thống không giảm tác động điều khiển → có thể gây ra quá tốc độ. Bộ mờ sẽ nhanh chóng giảm Kp, tăng nhẹ Kd để hoàn thiện dao động, giúp quá trình ổn định nhanh hơn.

Chương 4: THI CÔNG MÔ HÌNH

- Ở chương 2 của đồ án, việc tính chọn các thiết bị và linh kiện được tiến hành dựa trên các yêu cầu thực tế đối với một hệ thống xe tự hành (AGV) ứng dụng trong công nghiệp. Các yêu cầu này bao gồm khả năng chở tải trọng lớn (20–30 kg), hoạt động ổn định trên các địa hình phẳng và có độ dốc nhẹ, kích thước phù hợp với không gian làm việc trong nhà máy hoặc kho xưởng, khả năng hoạt động liên tục trong thời gian dài (lên đến 20 giờ/ngày), thời gian sạc ngắn, cùng khả năng tiết kiệm năng lượng và dễ dàng bảo trì. Do đó, các thiết bị được lựa chọn trong giai đoạn này có công suất cao, độ bền cơ khí tốt, khả năng giao tiếp công nghiệp mạnh mẽ (như CAN, RS485), và mức chi phí phù hợp với các tiêu chuẩn công nghiệp. Ví dụ, các động cơ công suất lớn, bộ mã hóa (encoder) chính xác cao, bộ điều khiển công nghiệp chuyên dụng, và nguồn cấp đáp ứng ổn định dòng tải lớn đều được cân nhắc sử dụng. Mục tiêu của giai đoạn này là đảm bảo tính khả thi và tối ưu hóa về mặt kỹ thuật của một hệ thống AGV thực tế.

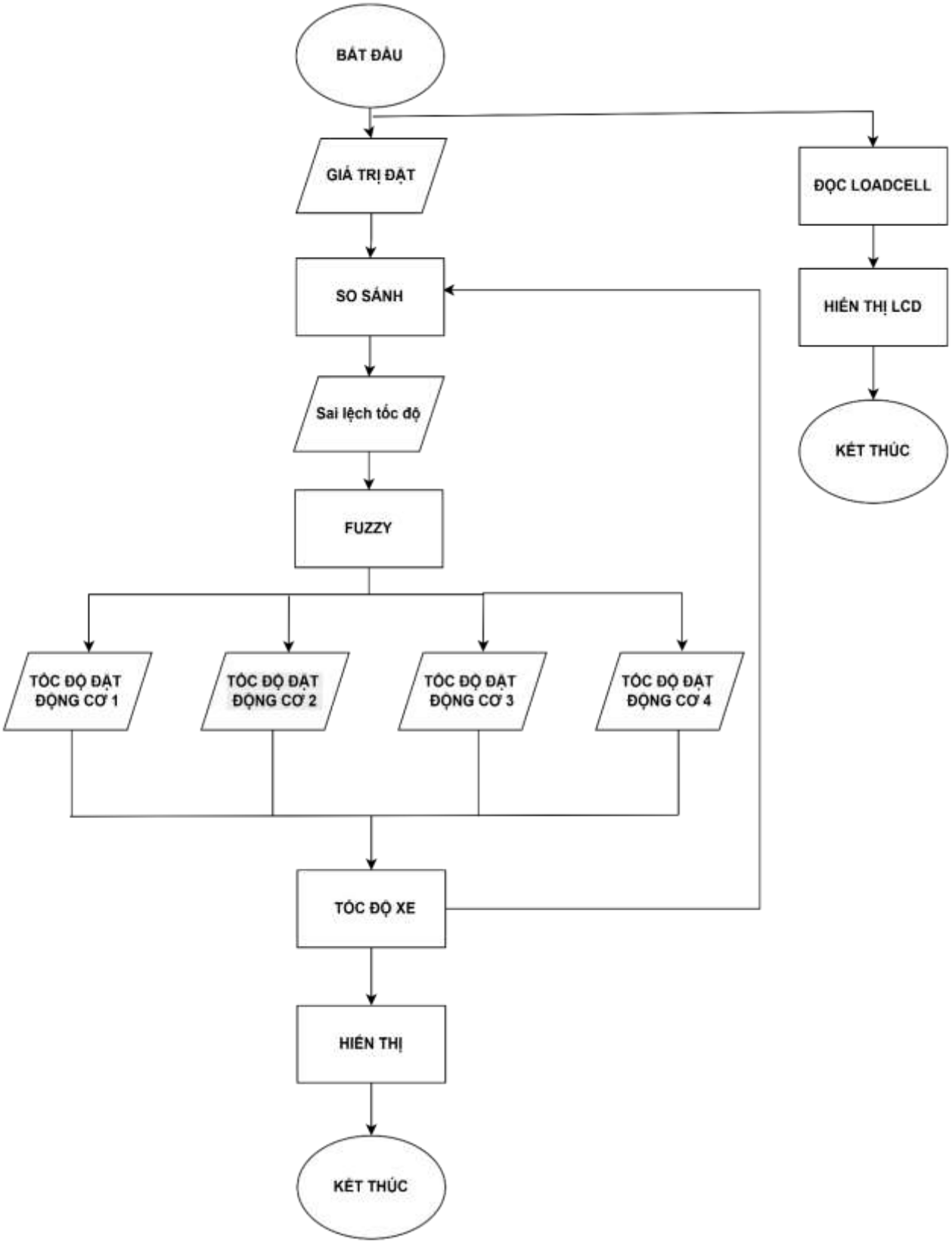
- Tuy nhiên, để phù hợp với phạm vi và điều kiện thực hiện, trong bối cảnh tài nguyên giới hạn về kinh phí, thời gian và không gian triển khai, hệ thống được thiết kế lại dưới dạng mô hình thu nhỏ. Trong mô hình này, các thiết bị được lựa chọn sao cho vẫn đảm bảo nguyên lý hoạt động và cấu trúc điều khiển cơ bản, nhưng có chi phí thấp hơn, dễ tiếp cận và thuận tiện trong việc thi công, lắp ráp cũng như lập trình. Cụ thể, hệ thống sử dụng các động cơ DC có encoder loại JGA25-371 công suất nhỏ, mỗi động cơ được điều khiển riêng biệt bằng driver BTS7960, vi điều khiển sử dụng nền tảng Arduino Mega 2560 thay cho PLC hoặc máy tính công nghiệp, kết hợp cùng các thiết bị ngoại vi như module hiển thị LCD I2C, cảm biến lực (loadcell) kết nối qua mạch chuyển đổi HX711, và tay điều khiển không dây qua sóng RF.

4.1 Sơ đồ kết nối phần cứng



Hình 4. 1 Sơ đồ kết nối

4.2 Lưu đồ thuật toán

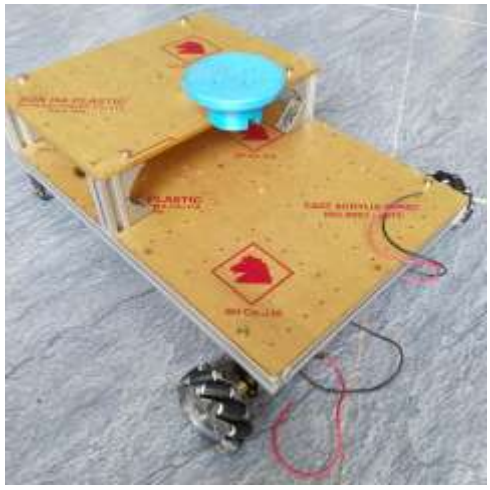


Hình 4. 2 Lưu đồ thuật toán

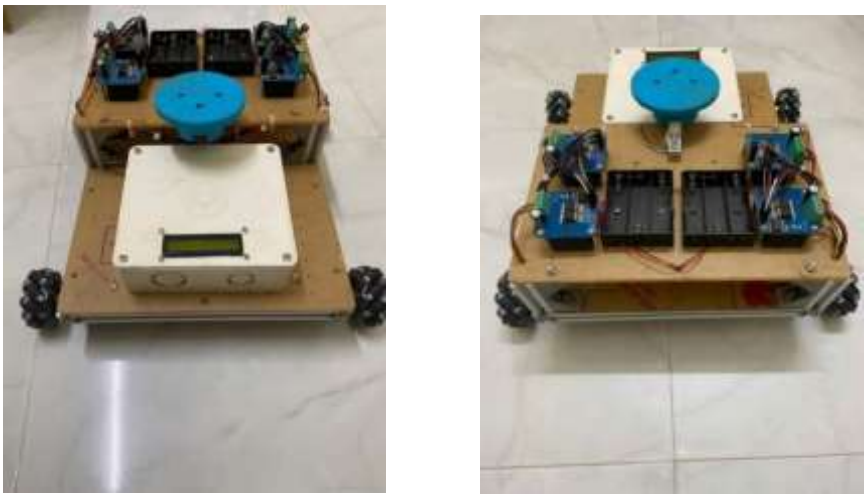
4.3 Thi công mô hình

- Các thành phần cơ khí:

Hình 4. 3 Khung xe và giá lắp động cơ



Hình 4. 4 Hoàn thiện hệ thống cơ khí của hệ thống



Hình 4. 5 Hoàn thiện mô hình

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Sau quá trình thực hiện đồ án tốt nghiệp với đề tài **"Nghiên cứu và thiết kế bộ điều khiển thích nghi dựa trên logic mờ cho xe tự hành"** nhóm chúng em đã có cơ hội tiếp cận và vận dụng tổng hợp nhiều kiến thức chuyên ngành đã học vào một hệ thống ứng dụng thực tế, từ lý thuyết điều khiển cho đến triển khai mô hình phần cứng.

Thông qua quá trình nghiên cứu, sinh viên đã nắm vững được các kiến thức cơ bản và nâng cao liên quan đến hệ thống điều khiển, bao gồm: động học và động lực học của xe sử dụng bánh xe Mecanum, kỹ thuật điều khiển PID truyền thống và những hạn chế của nó, cũng như nguyên lý xây dựng và ứng dụng bộ điều khiển thích nghi dựa trên logic mờ. Ngoài ra, việc tiến hành nhận dạng hệ thống để xây dựng mô hình hàm truyền đạt và thực hiện mô phỏng điều khiển trên phần mềm MATLAB/Simulink đã giúp sinh viên hiểu sâu hơn về mối liên hệ giữa mô hình toán học và thực tiễn vận hành hệ thống.

Về mặt thực hành, sinh viên đã thiết kế và lắp ráp thành công mô hình thu nhỏ của hệ thống xe tự hành sử dụng 4 bánh Mecanum. Hệ thống có khả năng vận hành với tốc độ đặt và ghi nhận sự thay đổi tải trọng, từ đó hiển thị thông tin lên màn hình và phục vụ cho việc đánh giá chất lượng điều khiển. Tuy nhiên, trong khuôn khổ đồ án, do hạn chế về thời gian, chi phí và thiết bị, đề tài mới chỉ dừng lại ở việc điều khiển vận tốc tuyến tính trong điều kiện đơn giản. Các yếu tố như điều hướng thông minh, định vị trong không gian, bám quỹ đạo chính xác hay điều khiển lực tổng hợp (Fx, Fy, τz) chưa được tích hợp đầy đủ. Việc triển khai thuật toán fuzzy-PID cũng mới dừng lại ở mức độ mô phỏng, chưa áp dụng thực nghiệm trực tiếp lên mô hình.

Tổng kết lại, đồ án đã giúp sinh viên hiểu rõ quy trình thiết kế một hệ thống điều khiển hoàn chỉnh từ khâu lý thuyết đến thực nghiệm, đồng thời rèn luyện được các kỹ năng nghiên cứu, tư duy hệ thống, khả năng xử lý kỹ thuật và lập trình nhúng. Đây là bước chuẩn bị quan trọng cho việc tiếp cận các hệ thống tự động hóa phức tạp hơn trong môi trường công nghiệp thực tế sau này.

2. Hướng phát triển

Sau khi hoàn thành dự án "Nghiên cứu và thiết kế bộ điều khiển thích nghi dựa trên logic mờ cho xe tự hành", nhóm nhận thấy vẫn còn nhiều tiềm năng để tiếp tục mở rộng, hoàn thiện và nâng cao tính năng thực thi ứng dụng của hệ thống. Một số khả năng phát triển định hướng trong các thành phần tương lai:

Tích hợp định vị và điều hướng thông minh (SLAM, GPS, Camera): Mở rộng hệ thống bằng cách tích hợp các định vị thuật toán và bản đồ hóa đồng thời (SLAM) kết hợp với biến cảm ứng Lidar hoặc camera, cho phép xe tự động xác định vị trí và chuyển trong môi trường mở rộng, phức tạp mà không cần cố định đường dẫn.

- Ưu tiên điều khiển thông minh: Hiện tại hệ thống sử dụng hai cấp độ điều khiển Fuzzy-PID, có thể phát triển các phương pháp nâng cao điều khiển tối ưu như:

+ Tự động điều chỉnh mờ

+ Kết hợp mờ Mạng thần kinh hoặc GA (Thuật toán di truyền) để tự động tối ưu hóa tham số.

+ MPC (Model Predictive Control) kết hợp Fuzzy nhắm mục tiêu mong đợi và xử lý tốt hơn các vấn đề có tốc độ chậm, tải thay đổi.

- Ứng dụng vào mô hình thực tế quy mô lớn hơn: Không chỉ dừng lại ở mô hình thử nghiệm nhỏ, hệ thống có thể được phát triển để ứng dụng vào các kho hàng thông minh, xưởng sản xuất tự động hóa, với khả năng điều phối nhiều AGV cùng lúc thông qua mạng truyền thông (CAN, MQTT, ROS...).

- Phát triển giao diện điều khiển từ xa và giám sát thời gian thực: Xây dựng một giao diện giám sát (HMI/Web app) để theo dõi trạng thái xe, vị trí, cảnh báo lỗi… từ xa thông qua mạng WiFi hoặc 4G.

- Năng lượng ưu tiên: Thử nghiệm bổ sung năng lượng quản lý thuật toán và chế độ ngủ thông minh, nâng cao thời gian vận hành của xe tối ưu mà không cần tăng dung lượng pin.

- Nâng cấp phần cứng – truyền hệ thống: Thay thế động cơ DC thông thường bằng động cơ có tốc độ tiếp theo cao hơn (BLDC + Chính xác bộ mã hóa), giúp tăng khả năng phản ứng của hệ thống trong môi trường điều kiện thay đổi xung đột.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Thị Thanh, Võ Thu Hà, Trajectory tracking control for the mecanum wheel mobile robot the dynamic surface controller connected to the neural network, Sep 2024)

[2] Hamid Taheri, Bing Qiao, Nurallah Ghaeminezhad, Kinematic Model of a Four Mecanum Wheeled Mobile Robot, International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 3, March 2015)

[3] Nguyễn Doãn Phước, Lý thuyết điều khiển tuyến tính, Assoc. Prof. of Department of Automatic Control, Hanoi University of Technology, 2009)

[4] O. Diegel, A. Badve, G. Bright, J. Potgieter, and S. Tlale, "Improved Mecanum Wheel Design for Omni-directional Robots," no. November, pp. 27–29, 2002.

[5] T. A. Baede, "Motion control of an omnidirectional mobile robot," 2006.

[6] X. Li and A. Zell, "Motion control of an omnidirectional mobile robot," 2006.

[7] P. F. Muir, C. P. Neuman, Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. Autonomous robot vehicles, New York, 1999.

[9] Nguyễn Hồng Thái, et al., "Trajectory tracking control of AGV mobile robot by time-varying parameter PID controller," in National Conference on Dynamics and Control, Hanoi University of Science and Technology, 2022.

[10] Trinh Thi Khanh Ly, et al., "Dynamic Simulation of Differential-Driven Mobile Robot Taking into Account the Friction Between the Wheel and the Road Surface," in International Conference on Material, Machines and Methods for Sustainable Development, 2023.

[11] Qing Xu, Jiangming Kan, Shanan Chen and Shengqi Yan, Fuzzy PID Based Trajectory Tracking Control of Mobile Robot and its Simulation in Simulink, School of Technology, Beijing Forestry University, 2014

[12] Prof. D. Prattichizzo, Leader-Follower Formation Control and Visibility Maintenance of Nonholonomic Mobile Robots, Ph.D. Thesis, University of Siena, March 2009.

[13] Jun Qian, Bin Zi, Daoming Wang, Yangang Ma and Dan Zhang, The Design and Development of an Omni-Directional Mobile Robot Oriented to an Intelligent Manufacturing System, 10 September 2017

PHỤ LỤC

```
#include <PID_v1.h>          // Thư viện PID của Brett Beauregard
#include <HX711.h>           // Thư viện cho Loadcell Module HX711
#include <Wire.h>            // Thư viện cho giao tiếp I2C
#include <LiquidCrystal_I2C.h> // Thư viện cho LCD I2C
#include <PS2X_lib.h.h>      // Thư viện cho tay cầm PS2 (Đảm bảo tên file chính xác)

// --- ĐỊNH NGHĨA CÁC CHÂN & THÔNG SỐ CƠ BẢN ---
// Chân Encoder (ví dụ cho 4 bánh) - Sử dụng Interrupts cho độ chính xác cao
const byte ENCODER_FL_A = 2; // Front Left - Chân Interrupt 0 (Digital Pin 2)
const byte ENCODER_FL_B = 20; // Chân Digital (PCINT hoặc đọc thường)
const byte ENCODER_FR_A = 3; // Front Right - Chân Interrupt 1 (Digital Pin 3)
const byte ENCODER_FR_B = 21;
const byte ENCODER_RL_A = 18; // Rear Left - Chân Interrupt 5 (Digital Pin 18)
const byte ENCODER_RL_B = 22;
const byte ENCODER_RR_A = 19; // Rear Right - Chân Interrupt 4 (Digital Pin 19)
const byte ENCODER_RR_B = 23;

// Chân điều khiển động cơ (ví dụ dùng PWM và hướng)
```

```
const byte MOTOR_FL_PWM = 4;
const byte MOTOR_FL_DIR1 = 5;
const byte MOTOR_FL_DIR2 = 6;

const byte MOTOR_FR_PWM = 7;
const byte MOTOR_FR_DIR1 = 8;
const byte MOTOR_FR_DIR2 = 9;

const byte MOTOR_RL_PWM = 10;
const byte MOTOR_RL_DIR1 = 11;
const byte MOTOR_RL_DIR2 = 12;

const byte MOTOR_RR_PWM = 13;
const byte MOTOR_RR_DIR1 = 14;
const byte MOTOR_RR_DIR2 = 15;

// Chân cho Loadcell HX711
const byte LOADCELL_DOUT_PIN = 24;
const byte LOADCELL_SCK_PIN = 26;

// Địa chỉ I2C của LCD và kích thước
const int LCD_I2C_ADDRESS = 0x27;
const int LCD_COLS = 16;
const int LCD_ROWS = 2;
```

```
// Chân cho tay cầm PS2
#define PS2_DAT 31 // DI (Data)
#define PS2_CMD 30 // DO (Command)
#define PS2_SEL 28 // CS (Attention)
#define PS2_CLK 29 // CLK (Clock)

// Chế độ tay cầm PS2 (analog hoặc digital)
#define PS2_MODE_ANALOG 0x01
#define PS2_MODE_DIGITAL 0x00

// Thông số xe Mecanum
const float WHEEL_RADIUS = 0.05;
const float ROBOT_LENGTH = 0.3;
const float ROBOT_WIDTH = 0.25;
const float ENCODER_CPR = 480.0;

// Giới hạn tốc độ động cơ và robot (có thể điều chỉnh)
const float MAX_WHEEL_SPEED_RAD_S = 10.0; // Max speed for each wheel in rad/s
const float MAX_ROBOT_VX = 0.5; // Max robot forward/backward speed in m/s
const float MAX_ROBOT_VY = 0.5; // Max robot strafe speed in m/s
const float MAX_ROBOT_OMEGA = 2.0; // Max robot angular speed in rad/s
```

```
// --- BIẾN TOÀN CỤC ---
// Biến đếm Encoder
volatile long encoderCountFL = 0;
volatile long encoderCountFR = 0;
volatile long encoderCountRL = 0;
volatile long encoderCountRR = 0;

// Thời gian đo tốc độ
unsigned long prevMillis = 0;
const unsigned long MEASURE_INTERVAL_MS = 50; // Tần suất đo tốc độ và cập nhật PID (ms)
const unsigned long LOADCELL_READ_INTERVAL_MS = 500; // Tần suất đọc loadcell (ms)
unsigned long prevLoadcellMillis = 0;
const unsigned long PS2_READ_INTERVAL_MS = 50; // Tần suất đọc PS2 (ms)
unsigned long prevPS2Millis = 0;

// Tốc độ thực tế (rad/s)
double actualSpeedFL = 0;
double actualSpeedFR = 0;
double actualSpeedRL = 0;
double actualSpeedRR = 0;

// Tốc độ mong muốn cho từng bánh (đầu ra của Fuzzy-PID hoặc PS2)
double desiredSpeedFL = 0;
```

```
double desiredSpeedFR = 0;
double desiredSpeedRL = 0;
double desiredSpeedRR = 0;

// Các biến cho PID cấp thấp (cho từng động cơ)
double Kp_low = 2.0, Ki_low = 0.5, Kd_low = 0.1;
double outputFL, outputFR, outputRL, outputRR;

// Khởi tạo các đối tượng PID cho từng bánh xe
PID pidFL(&actualSpeedFL, &outputFL, &desiredSpeedFL, Kp_low, Ki_low, Kd_low, DIRECT);
PID pidFR(&actualSpeedFR, &outputFR, &desiredSpeedFR, Kp_low, Ki_low, Kd_low, DIRECT);
PID pidRL(&actualSpeedRL, &outputRL, &desiredSpeedRL, Kp_low, Ki_low, Kd_low, DIRECT);
PID pidRR(&actualSpeedRR, &outputRR, &desiredSpeedRR, Kp_low, Ki_low, Kd_low, DIRECT);

// Các biến cho Fuzzy-PID (cấp cao)
double vehicle_vx_desired = 0.0;
double vehicle_vy_desired = 0.0;
double vehicle_omega_desired = 0.0;

double current_error_x = 0;
double current_error_y = 0;
```

```
double current_error_theta = 0;

double prev_error_x = 0;
double prev_error_y = 0;
double prev_error_theta = 0;

double Kp_fuzzy_x, Ki_fuzzy_x, Kd_fuzzy_x;
double Kp_fuzzy_y, Ki_fuzzy_y, Kd_fuzzy_y;
double Kp_fuzzy_omega, Ki_fuzzy_omega, Kd_fuzzy_omega;

// Đối tượng Loadcell
HX711 scale;
float calibration_factor = -104500; // HIỆU CHUẨN LẠI GIÁ TRỊ NÀY
float current_load_kg = 0.0;

// Đối tượng LCD
LiquidCrystal_I2C lcd(LCD_I2C_ADDRESS, LCD_COLS, LCD_ROWS);

// Đối tượng PS2X
PS2X ps2x;
int PS2_error = 0; // 0 = no error, 1 = no controller, 2 = no communication
bool ps2_connected = false; // Trạng thái kết nối PS2
bool autonomous_mode = true; // Chế độ tự hành (true) hoặc điều khiển bằng tay (false)
```

```
// --- CÁC HÀM XỬ LÝ INTERRUPT CHO ENCODER ---
void ISR_Encoder_FL_A() {
  if (digitalRead(ENCODER_FL_B) == LOW) {
    encoderCountFL++;
  } else {
    encoderCountFL--;
  }
}

void ISR_Encoder_FR_A() {
  if (digitalRead(ENCODER_FR_B) == HIGH) {
    encoderCountFR++;
  } else {
    encoderCountFR--;
  }
}

void ISR_Encoder_RL_A() {
  if (digitalRead(ENCODER_RL_B) == LOW) {
    encoderCountRL++;
  } else {
    encoderCountRL--;
  }
```

```
}

void ISR_Encoder_RR_A() {
  if (digitalRead(ENCODER_RR_B) == HIGH) {
    encoderCountRR++;
  } else {
    encoderCountRR--;
  }
}

// --- HÀM KHỞI TẠO ---
void setup() {
  Serial.begin(115200);

  lcd.init();
  lcd.backlight();
  lcd.print("Robot Init...");
  lcd.setCursor(0, 1);
  lcd.print("Loading modules");

  // Cấu hình chân động cơ
  pinMode(MOTOR_FL_PWM, OUTPUT); pinMode(MOTOR_FL_DIR1, OUTPUT); pinMode(MOTOR_FL_DIR2, OUTPUT);
  pinMode(MOTOR_FR_PWM, OUTPUT); pinMode(MOTOR_FR_DIR1, OUTPUT); pinMode(MOTOR_FR_DIR2, OUTPUT);
```

```
  pinMode(MOTOR_RL_PWM, OUTPUT); pinMode(MOTOR_RL_DIR1, OUTPUT); pinMode(MOTOR_RL_DIR2, OUTPUT);

pinMode(MOTOR_RR_PWM, OUTPUT); pinMode(MOTOR_RR_DIR1, OUTPUT); pinMode(MOTOR_RR_DIR2, OUTPUT);


  // Cấu hình chân Encoder và gắn Interrupts

  pinMode(ENCODER_FL_A, INPUT_PULLUP); pinMode(ENCODER_FL_B, INPUT_PULLUP); attachInterrupt(digitalPinToInterrupt(ENCODER_FL_A), ISR_Encoder_FL_A, RISING);

  pinMode(ENCODER_FR_A, INPUT_PULLUP); pinMode(ENCODER_FR_B, INPUT_PULLUP); attachInterrupt(digitalPinToInterrupt(ENCODER_FR_A), ISR_Encoder_FR_A, RISING);

  pinMode(ENCODER_RL_A, INPUT_PULLUP); pinMode(ENCODER_RL_B, INPUT_PULLUP); attachInterrupt(digitalPinToInterrupt(ENCODER_RL_A), ISR_Encoder_RL_A, RISING);

  pinMode(ENCODER_RR_A, INPUT_PULLUP); pinMode(ENCODER_RR_B, INPUT_PULLUP); attachInterrupt(digitalPinToInterrupt(ENCODER_RR_A), ISR_Encoder_RR_A, RISING);


  // Thiết lập chế độ cho các bộ PID cấp thấp

  pidFL.SetMode(AUTOMATIC); pidFR.SetMode(AUTOMATIC); pidRL.SetMode(AUTOMATIC); pidRR.SetMode(AUTOMATIC);

  pidFL.SetOutputLimits(-255, 255); pidFR.SetOutputLimits(-255, 255); pidRL.SetOutputLimits(-255, 255); pidRR.SetOutputLimits(-255, 255);

  pidFL.SetSampleTime(MEASURE_INTERVAL_MS);
pidFR.SetSampleTime(MEASURE_INTERVAL_MS);
pidRL.SetSampleTime(MEASURE_INTERVAL_MS);
pidRR.SetSampleTime(MEASURE_INTERVAL_MS);
```

```
  // Khởi tạo các tham số PID mờ ban đầu
  Kp_fuzzy_x = 1.0; Ki_fuzzy_x = 0.01; Kd_fuzzy_x = 0.0;
  Kp_fuzzy_y = 1.0; Ki_fuzzy_y = 0.01; Kd_fuzzy_y = 0.0;
  Kp_fuzzy_omega = 1.0; Ki_fuzzy_omega = 0.01; Kd_fuzzy_omega = 0.0;

  // Khởi tạo HX711
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(calibration_factor);
  scale.tare();

  // Khởi tạo PS2 Controller
  PS2_error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL,
PS2_DAT, true, false); // Analog mode, no rumble
  if (PS2_error == 0) {
    Serial.println("PS2 Controller found and configured.");
    ps2_connected = true;
    lcd.setCursor(0, 1);
    lcd.print("PS2 Ready       ");
    delay(500);
  } else if (PS2_error == 1) {
    Serial.println("No PS2 Controller found. Check wiring.");
    lcd.setCursor(0, 1);
    lcd.print("PS2 not found!  ");
```

```
  } else if (PS2_error == 2) {
    Serial.println("No communication with PS2. Check power/mode.");
    lcd.setCursor(0, 1);
    lcd.print("PS2 No Comm!    ");
  }
  delay(1000); // Đợi để thông báo PS2
  lcd.clear();
  lcd.print("System Ready!");
  lcd.setCursor(0,1);
  lcd.print("Mode: Autonomous");

  Serial.println("System Initialized.");
}

// --- HÀM VÒNG LẶP CHÍNH ---
void loop() {
  unsigned long currentMillis = millis();

  // 1. Cập nhật tốc độ thực tế của từng bánh (luôn chạy)
  if (currentMillis - prevMillis >= MEASURE_INTERVAL_MS) {
    noInterrupts();
    long currentEncoderFL = encoderCountFL; long currentEncoderFR = encoderCountFR;
    long currentEncoderRL = encoderCountRL; long currentEncoderRR = encoderCountRR;
```

```
encoderCountFL = 0; encoderCountFR = 0; encoderCountRL = 0; encoderCountRR = 0;
  interrupts();

  actualSpeedFL = (double)currentEncoderFL / ENCODER_CPR * (1000.0 / MEASURE_INTERVAL_MS) * (2 * PI);
  actualSpeedFR = (double)currentEncoderFR / ENCODER_CPR * (1000.0 / MEASURE_INTERVAL_MS) * (2 * PI);
  actualSpeedRL = (double)currentEncoderRL / ENCODER_CPR * (1000.0 / MEASURE_INTERVAL_MS) * (2 * PI);
  actualSpeedRR = (double)currentEncoderRR / ENCODER_CPR * (1000.0 / MEASURE_INTERVAL_MS) * (2 * PI);

  prevMillis = currentMillis;

  // --- Cập nhật vận tốc mong muốn dựa trên chế độ ---
  if (autonomous_mode) {
    // Chế độ tự hành: Sử dụng logic Fuzzy-PID
    // Bạn cần có thông tin current_vehicle_vx, vy, omega thực tế từ IMU/Odometry ở đây
    double current_vehicle_vx = 0.0; // Placeholder
    double current_vehicle_vy = 0.0; // Placeholder
    double current_vehicle_omega = 0.0; // Placeholder

    // Để demo, xe tự hành muốn tiến thẳng với tốc độ 0.2 m/s
    // Hoặc dựa trên thông tin tải
```

```
    if (current_load_kg > 0.1) {
       vehicle_vx_desired = 0.1;
    } else {
       vehicle_vx_desired = 0.2;
    }
    vehicle_vy_desired = 0.0;
    vehicle_omega_desired = 0.0;

    current_error_x = vehicle_vx_desired - current_vehicle_vx;
    current_error_y = vehicle_vy_desired - current_vehicle_vy;
    current_error_theta = vehicle_omega_desired - current_vehicle_omega;

    double delta_error_x = current_error_x - prev_error_x;
    double delta_error_y = current_error_y - prev_error_y;
    double delta_error_theta = current_error_theta - prev_error_theta;

    prev_error_x = current_error_x;
    prev_error_y = current_error_y;
    prev_error_theta = current_error_theta;

    calculateFuzzyPIDGains(current_error_x, delta_error_x, &Kp_fuzzy_x,
&Ki_fuzzy_x, &Kd_fuzzy_x);
    calculateFuzzyPIDGains(current_error_y, delta_error_y, &Kp_fuzzy_y,
&Ki_fuzzy_y, &Kd_fuzzy_y);
```

```
    calculateFuzzyPIDGains(current_error_theta, delta_error_theta, &Kp_fuzzy_omega, &Ki_fuzzy_omega, &Kd_fuzzy_omega);


    double output_vx = Kp_fuzzy_x * current_error_x + Ki_fuzzy_x * (current_error_x + prev_error_x) * (MEASURE_INTERVAL_MS / 1000.0) + Kd_fuzzy_x * delta_error_x / (MEASURE_INTERVAL_MS / 1000.0);

    double output_vy = Kp_fuzzy_y * current_error_y + Ki_fuzzy_y * (current_error_y + prev_error_y) * (MEASURE_INTERVAL_MS / 1000.0) + Kd_fuzzy_y * delta_error_y / (MEASURE_INTERVAL_MS / 1000.0);

    double output_omega = Kp_fuzzy_omega * current_error_theta + Ki_fuzzy_omega * (current_error_theta + prev_error_theta) * (MEASURE_INTERVAL_MS / 1000.0) + Kd_fuzzy_omega * delta_error_theta / (MEASURE_INTERVAL_MS / 1000.0);


    // --- Động học nghịch đảo (Inverse Kinematics) của Mecanum ---

desiredSpeedFL = (1.0 / WHEEL_RADIUS) * (output_vx - output_vy - (ROBOT_LENGTH + ROBOT_WIDTH) * output_omega);

    desiredSpeedFR = (1.0 / WHEEL_RADIUS) * (output_vx + output_vy + (ROBOT_LENGTH + ROBOT_WIDTH) * output_omega);

    desiredSpeedRL = (1.0 / WHEEL_RADIUS) * (output_vx + output_vy - (ROBOT_LENGTH + ROBOT_WIDTH) * output_omega);

    desiredSpeedRR = (1.0 / WHEEL_RADIUS) * (output_vx - output_vy + (ROBOT_LENGTH + ROBOT_WIDTH) * output_omega);


  } else {

    // Chế độ điều khiển bằng tay (PS2): desiredSpeedFL/FR/RL/RR đã được tính ở phần đọc PS2

    // Không cần tính toán Fuzzy-PID ở đây, chỉ cần dùng giá trị từ PS2
```

```
    // Đảm bảo vehicle_vx_desired, vy, omega = 0 để tránh xung đột với điều khiển tự hành
    vehicle_vx_desired = 0;
    vehicle_vy_desired = 0;
    vehicle_omega_desired = 0;
  }

  // Giới hạn tốc độ mong muốn của bánh xe
  desiredSpeedFL = constrain(desiredSpeedFL, -MAX_WHEEL_SPEED_RAD_S, MAX_WHEEL_SPEED_RAD_S);
  desiredSpeedFR = constrain(desiredSpeedFR, -MAX_WHEEL_SPEED_RAD_S, MAX_WHEEL_SPEED_RAD_S);
  desiredSpeedRL = constrain(desiredSpeedRL, -MAX_WHEEL_SPEED_RAD_S, MAX_WHEEL_SPEED_RAD_S);
  desiredSpeedRR = constrain(desiredSpeedRR, -MAX_WHEEL_SPEED_RAD_S, MAX_WHEEL_SPEED_RAD_S);

  // --- 3. Bộ điều khiển PID (Cấp thấp) cho từng động cơ ---
  pidFL.Compute();
  pidFR.Compute();
  pidRL.Compute();
  pidRR.Compute();

  // --- 4. Điều khiển động cơ ---
```

```
  setMotorSpeed(MOTOR_FL_PWM, MOTOR_FL_DIR1, MOTOR_FL_DIR2, outputFL);
  setMotorSpeed(MOTOR_FR_PWM, MOTOR_FR_DIR1, MOTOR_FR_DIR2, outputFR);
  setMotorSpeed(MOTOR_RL_PWM, MOTOR_RL_DIR1, MOTOR_RL_DIR2, outputRL);
  setMotorSpeed(MOTOR_RR_PWM, MOTOR_RR_DIR1, MOTOR_RR_DIR2, outputRR);

  // --- Gỡ lỗi (Debug) trên Serial ---
  Serial.print("Mode: "); Serial.print(autonomous_mode ? "AUTO" : "MANUAL");
  Serial.print("\tDesired FL: "); Serial.print(desiredSpeedFL, 2);
  Serial.print("\tActual FL: "); Serial.print(actualSpeedFL, 2);
  Serial.print("\tOutput FL: "); Serial.println(outputFL, 2);

  // Serial.print("KpX: "); Serial.print(Kp_fuzzy_x, 3);
  // Serial.print("\tKiX: "); Serial.print(Ki_fuzzy_x, 3);
  // Serial.print("\tKdX: "); Serial.println(Kd_fuzzy_x, 3);
  // Serial.println("---");
 }

 // 5. Đọc và hiển thị giá trị Loadcell (ít thường xuyên hơn)
 if (currentMillis - prevLoadcellMillis >= LOADCELL_READ_INTERVAL_MS) {
  if (scale.is_ready()) {
```

```
      current_load_kg = scale.get_units(5);
      // Serial.print("Load: "); Serial.print(current_load_kg, 3); Serial.println(" kg");

      // Hiển thị lên LCD
      lcd.setCursor(0, 0);
      lcd.print("Load: ");
      lcd.print(current_load_kg, 2);
      lcd.print(" kg ");

      lcd.setCursor(0, 1);
      lcd.print("Mode: ");
      lcd.print(autonomous_mode ? "AUTO" : "MANUAL");
lcd.print("  "); // Xóa ký tự thừa
    } else {
      // Serial.println("HX711 not ready!");
      lcd.setCursor(0, 1);
      lcd.print("HX711 error!");
    }
    prevLoadcellMillis = currentMillis;
  }

  // 6. Đọc tín hiệu từ PS2 Controller
  if (ps2_connected && (currentMillis - prevPS2Millis >= PS2_READ_INTERVAL_MS)) {
```

```
  ps2x.read_gamepad(false, 0); // Đọc gamepad, không rumble

  // Chuyển đổi chế độ điều khiển (ví dụ: nút START)
  if (ps2x.ButtonPressed(PSB_START)) {
    autonomous_mode = !autonomous_mode;
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Mode Switched");
    lcd.setCursor(0,1);
    lcd.print(autonomous_mode ? "Autonomous" : "Manual Control");
    delay(1000); // Đợi để người dùng nhìn thấy thông báo
    lcd.clear();
  }

  if (!autonomous_mode) { // Chỉ điều khiển bằng PS2 khi ở chế độ Manual
    // Đọc giá trị từ cần analog (Left stick for movement, Right stick for rotation)
    // Giá trị từ 0-255, cần chuyển về -127 đến 127 hoặc -1 đến 1
    int LX = ps2x.Analog(PSS_LX); // Trái/Phải của cần trái
    int LY = ps2x.Analog(PSS_LY); // Lên/Xuống của cần trái
    int RX = ps2x.Analog(PSS_RX); // Trái/Phải của cần phải

    // Chuyển đổi từ 0-255 sang -1.0 đến 1.0 (trừ đi 127 để có 0 ở giữa)
    // Và sau đó nhân với MAX_ROBOT_SPEED để có vận tốc mong muốn thực
tế
```

```
    double cmd_vx = map(LY, 0, 255, 127, -127) / 127.0 * MAX_ROBOT_VX; // LY: 127 = stop, 0 = max fwd, 255 = max bwd

    double cmd_vy = map(LX, 0, 255, -127, 127) / 127.0 * MAX_ROBOT_VY; // LX: 127 = stop, 0 = max left, 255 = max right

    double cmd_omega = map(RX, 0, 255, -127, 127) / 127.0 * MAX_ROBOT_OMEGA; // RX: 127 = stop, 0 = max CCW, 255 = max CW


    // Xử lý Deadzone cho cần analog (tránh trôi robot khi cần ở vị trí giữa)

    if (abs(LX - 127) < 10) cmd_vy = 0;

    if (abs(LY - 127) < 10) cmd_vx = 0;

    if (abs(RX - 127) < 10) cmd_omega = 0;


    // Tính toán tốc độ bánh xe từ vận tốc robot mong muốn (Động học nghịch đảo)

    desiredSpeedFL = (1.0 / WHEEL_RADIUS) * (cmd_vx - cmd_vy - (ROBOT_LENGTH + ROBOT_WIDTH) * cmd_omega);

    desiredSpeedFR = (1.0 / WHEEL_RADIUS) * (cmd_vx + cmd_vy + (ROBOT_LENGTH + ROBOT_WIDTH) * cmd_omega);

    desiredSpeedRL = (1.0 / WHEEL_RADIUS) * (cmd_vx + cmd_vy - (ROBOT_LENGTH + ROBOT_WIDTH) * cmd_omega);

    desiredSpeedRR = (1.0 / WHEEL_RADIUS) * (cmd_vx - cmd_vy + (ROBOT_LENGTH + ROBOT_WIDTH) * cmd_omega);


    // In ra giá trị PS2 để debug

    // Serial.print("PS2 LX: "); Serial.print(LX); Serial.print(" LY: "); Serial.print(LY); Serial.print(" RX: "); Serial.println(RX);
```

```
    // Serial.print("Cmd VX: "); Serial.print(cmd_vx); Serial.print(" VY: "); Serial.print(cmd_vy); Serial.print(" Omega: "); Serial.println(cmd_omega);
   } else {
// Khi ở chế độ tự hành, đảm bảo các biến điều khiển thủ công bằng 0
      desiredSpeedFL = 0;
      desiredSpeedFR = 0;
      desiredSpeedRL = 0;
      desiredSpeedRR = 0;
   }
   ps2x.cleardata(); // Xóa bộ đệm dữ liệu PS2
   prevPS2Millis = currentMillis;
  }

  // Nhận lệnh từ Serial/Bluetooth để điều khiển xe hoặc thay đổi chế độ
  if (Serial.available() > 0) {
   String command = Serial.readStringUntil('\n');
   command.trim();
   if (command == "forward") {
    autonomous_mode = true; // Chuyển sang tự hành
    vehicle_vx_desired = 0.5;
    vehicle_vy_desired = 0;
    vehicle_omega_desired = 0;
   } else if (command == "stop") {
    vehicle_vx_desired = 0;
```

```
      vehicle_vy_desired = 0;
      vehicle_omega_desired = 0;
    } else if (command == "tare_loadcell") {
       if (scale.is_ready()) {
           scale.tare();
           Serial.println("Loadcell tared!");
           lcd.clear(); lcd.print("Loadcell Tared!"); delay(1000); lcd.clear();
       } else { Serial.println("HX711 not ready to tare!"); }
    } else if (command == "manual") {
      autonomous_mode = false;
      lcd.clear(); lcd.print("Mode: Manual"); delay(500); lcd.clear();
    } else if (command == "auto") {
      autonomous_mode = true;
      lcd.clear(); lcd.print("Mode: Autonomous"); delay(500); lcd.clear();
    }
  }
}

// --- HÀM ĐIỀU KHIỂN ĐỘNG CƠ ---
void setMotorSpeed(byte pwmPin, byte dir1Pin, byte dir2Pin, double pwmValue)
{
  int absPwm = abs((int)pwmValue);
  absPwm = constrain(absPwm, 0, 255);
```

```
  if (pwmValue > 0) {
    digitalWrite(dir1Pin, HIGH);
    digitalWrite(dir2Pin, LOW);
  } else if (pwmValue < 0) {
    digitalWrite(dir1Pin, LOW);
    digitalWrite(dir2Pin, HIGH);
  } else {
    digitalWrite(dir1Pin, LOW);
    digitalWrite(dir2Pin, LOW);
  }
  analogWrite(pwmPin, absPwm);
}

// --- HÀM GIẢ ĐỊNH LOGIC MỜ ĐỂ ĐIỀU CHỈNH Kp, Ki, Kd ---
void calculateFuzzyPIDGains(double error, double delta_error, double *Kp_out, double *Ki_out, double *Kd_out) {
  // Logic mờ dựa trên error và delta_error
  if (abs(error) < 0.05 && abs(delta_error) < 0.01) {
    *Kp_out = 1.0; *Ki_out = 0.05; *Kd_out = 0.1;
  } else if (abs(error) > 0.5) {
    *Kp_out = 2.5; *Ki_out = 0.01; *Kd_out = 0.2;
  } else {
    *Kp_out = 1.5; *Ki_out = 0.02; *Kd_out = 0.15;
  }
```

```
  // Điều chỉnh thêm dựa trên tải trọng
  const float LIGHT_LOAD_THRESHOLD = 0.1;
  const float MEDIUM_LOAD_THRESHOLD = 1.0;
  const float HEAVY_LOAD_THRESHOLD = 5.0;

  if (current_load_kg > LIGHT_LOAD_THRESHOLD) {
     if (current_load_kg <= MEDIUM_LOAD_THRESHOLD) {
        *Kp_out *= 1.1; *Ki_out *= 1.05; *Kd_out *= 0.95;
     } else if (current_load_kg > MEDIUM_LOAD_THRESHOLD && current_load_kg <= HEAVY_LOAD_THRESHOLD) {
        *Kp_out *= 1.2; *Ki_out *= 1.1; *Kd_out *= 0.9;
     } else {
        *Kp_out *= 1.3; *Ki_out *= 1.15; *Kd_out *= 0.8;
     }
  }
*Kp_out = constrain(*Kp_out, 0.5, 5.0);
  *Ki_out = constrain(*Ki_out, 0.001, 0.5);
  *Kd_out = constrain(*Kd_out, 0.0, 1.0);
}
```

```
function q_ddot = MWMR_Dynamics_Full(tau, q_dot, incline_angle,Mt)
    % Inputs:
    %   tau: 4x1 vector - mômen điều khiển từng bánh xe
    %   q_dot: 3x1 vector - vận tốc [x_dot; y_dot; phi_dot]
    %   incline_angle: 2x1 vector - góc nghiêng [alpha; beta] (radian)

    %% Thông số vật lý
    mb = Mt +10 ;          % khối lượng thân robot (kg)
    mw = 1 ;           % khối lượng mỗi bánh xe (kg)
    r = 0.05;          % bán kính bánh xe (m)
    L = 0.3;           % nửa chiều dài robot (m)
    d = 0.2;           % nửa chiều rộng robot (m)
    Jw = 0.01;         % moment quán tính bánh xe (kg.m^2)
    Jb = 1.5;          % moment quán tính thân robot (kg.m^2)
    g = 9.81;          % gia tốc trọng trường (m/s^2)
    mu = 0.1;          % hệ số ma sát

    %% Tính các đại lượng
    % A, B
    A = (mw * r^2 * (L + d)^2) / (4 * (L + d)^2);
    B = (mw * r^2 * L * (L + d)) / (4 * (L + d)^2);

    % Momen quán tính toàn hệ
```

```
J = Jb + 4 * mw * (L^2 + d^2) + 4 * Jw;

%% Ma trận M (động học động lực học)
Job = mw * (L^2 + d^2);
M = [A + B + Job,    -B,         -A,          B;
    -B,         A + B + Job,      B,         -A;
    -A,              B,     A + B + Job,     -B;
     B,             -A,         -B,     A + B + Job];

%% Ma trận H
factor = 1 / (4 * (L + d));
H = factor * r * [
    (L + 2*d), (L + 2*d),   L,   L;
    (L + 2*d), (L + 2*d),   L,   L;
    L,         L,     (L + 2*d), (L + 2*d);
    L,         L,     (L + 2*d), (L + 2*d)
];

%% Ma trận K
K = diag([-r, -r, -r, -r]);

%% Ma trận J_R
J_R = (1 / r) * [
    1, -1, -(L + d);
```

```
    1,  1,  (L + d);
    1,  1, -(L + d);
    1, -1,  (L + d)
];

%% Vận tốc bánh xe
phi_dot = J_R * q_dot;   % 4x1

%% Ma sát
N = (mb + 4*mw) * g / 4;     % tải trọng mỗi bánh
Fms = N * mu * sign(phi_dot);       % ma sát tiếp xúc
FLms = N * mu * sign(phi_dot);      % ma sát con lăn

fms = Fms;    % 4x1
fLms = FLms;  % 4x1

%% Các lực
friction_force = H * fLms + K * fms;  % 4x1
control_force = tau;                  % 4x1

%% Lực tổng trong hệ tọa độ robot
net_force_body = J_R' * (control_force - friction_force);  % 3x1

%% Lực trọng lực theo mặt nghiêng
```

```
    alpha = incline_angle(1);   % rad
    beta  = incline_angle(2);   % rad
    M_total = mb + 4*mw;
    gravity_force = [M_total*g*sin(alpha); M_total*g*sin(beta); 0];  % 3x1

    %% Ma trận khối lượng robot
    Mr = diag([M_total, M_total, J]);

    %% Gia tốc q_ddot
    q_ddot = Mr \ (net_force_body + gravity_force);  % 3x1
end
```

```
function omega = wheelSpeedsFromForce(Fx, tau_z, dt,M)
%#codegen
% Mô phỏng vận tốc bánh xe từ lực PID, khởi đầu từ 0, tăng dần nếu lực dương, giảm dần nếu âm

  %% --- Trạng thái vận tốc ---
  persistent v_body initialized
  if isempty(initialized)
     v_body = zeros(3,1);  % [v_x; v_y; omega_z] khởi đầu = 0
     initialized = true;
  end

  %% --- Thông số robot ---
  m_b = 10+M;     % kg - khối lượng thân xe
  m_w = 1;      % kg - khối lượng bánh
  r   = 0.05;   % m  - bán kính bánh xe
  L   = 0.3;    % m  - nửa chiều dài
  d   = 0.2;   % m  - nửa chiều rộn
  J_b = 1.5;    % kg.m^2 - moment quán tính thân
  J_w = 0.01;   % kg.m^2 - moment quán tính mỗi bánh
  Fy= 0;
```

```
%% --- Tính tổng khối lượng và quán tính ---
M_total = m_b + 4 * m_w;
J_total = J_b + 4 * (J_w + m_w * (L^2 + d^2));
M_dyn = diag([M_total, M_total, J_total]);

%% --- Gia tốc tuyến tính & góc ---
F = [Fx; Fy; tau_z];
a_body = M_dyn \ F;

%% --- Cập nhật vận tốc thân xe ---
for i = 1:3
    delta_v = a_body(i) * dt;

    if v_body(i) == 0
        if delta_v > 0
            v_body(i) = v_body(i) + delta_v;
        end
    else
        if sign(delta_v) == sign(v_body(i))
            v_body(i) = v_body(i) + delta_v;
        else
            if abs(delta_v) < abs(v_body(i))
                v_body(i) = v_body(i) + delta_v;
            else
```

```
                v_body(i) = 0;
            end
        end
    end
end

%% --- Ma trận động học đảo ---
a = L + d;
T_inv = (1/r) * [
    1, -1, -a;
    1,  1,  a;
    1,  1, -a;
    1, -1,  a
];

%% --- Tính tốc độ góc từng bánh ---
omega = T_inv * v_body;
end

function tau = ComputeTorque1Wheel(omega)
% Tính mômen của 1 bánh từ vận tốc góc
% Đầu vào:
%   omega - vận tốc góc hiện tại của bánh xe (rad/s)
% Đầu ra:
```

```
%   tau   - mômen (Nm)

    % Thông số động học bánh xe
    Jw = 0.001;   % moment quán tính của bánh xe (kg.m^2)
    dt = 0.01;    % thời gian mẫu (s)

    % Biến lưu trạng thái omega trước đó
    persistent omega_prev

    if isempty(omega_prev)
        omega_prev = 0;
    end

    % Tính gia tốc góc
    alpha = (omega - omega_prev) / dt;

    % Tính mômen từ alpha
    tau = Jw * alpha;

    % Cập nhật omega trước đó
    omega_prev = omega;
end

function q_ddot = MWMR_Dynamics_Full(tau, q_dot, incline_angle,Mt)
```

```
% Inputs:
%   tau: 4x1 vector - mômen điều khiển từng bánh xe
%   q_dot: 3x1 vector - vận tốc [x_dot; y_dot; phi_dot]
%   incline_angle: 2x1 vector - góc nghiêng [alpha; beta] (radian)

%% Thông số vật lý
mb = Mt +10 ;          % khối lượng thân robot (kg)
mw = 1 ;           % khối lượng mỗi bánh xe (kg)
r = 0.05;          % bán kính bánh xe (m)
L = 0.3;           % nửa chiều dài robot (m)
d = 0.2;           % nửa chiều rộng robot (m)
Jw = 0.01;         % moment quán tính bánh xe (kg.m^2)
Jb = 1.5;          % moment quán tính thân robot (kg.m^2)
g = 9.81;          % gia tốc trọng trường (m/s^2)
mu = 0.1;          % hệ số ma sát

%% Tính các đại lượng
% A, B
A = (mw * r^2 * (L + d)^2) / (4 * (L + d)^2);
B = (mw * r^2 * L * (L + d)) / (4 * (L + d)^2);

% Momen quán tính toàn hệ
J = Jb + 4 * mw * (L^2 + d^2) + 4 * Jw;
```

```
%% Ma trận M (động học động lực học)
Job = mw * (L^2 + d^2);
M = [A + B + Job,    -B,         -A,          B;
    -B,         A + B + Job,      B,         -A;
    -A,              B,     A + B + Job,     -B;
     B,             -A,         -B,     A + B + Job];

%% Ma trận H
factor = 1 / (4 * (L + d));
H = factor * r * [
    (L + 2*d), (L + 2*d),   L,   L;
    (L + 2*d), (L + 2*d),   L,   L;
    L,         L,     (L + 2*d), (L + 2*d);
    L,         L,     (L + 2*d), (L + 2*d)
];

%% Ma trận K
K = diag([-r, -r, -r, -r]);

%% Ma trận J_R
J_R = (1 / r) * [
    1, -1, -(L + d);
    1,  1,  (L + d);
    1,  1, -(L + d);
```

```
    1, -1,  (L + d)
];

%% Vận tốc bánh xe
phi_dot = J_R * q_dot;   % 4x1

%% Ma sát
N = (mb + 4*mw) * g / 4;     % tải trọng mỗi bánh
Fms = N * mu * sign(phi_dot);       % ma sát tiếp xúc
FLms = N * mu * sign(phi_dot);      % ma sát con lăn

fms = Fms;    % 4x1
fLms = FLms;  % 4x1

%% Các lực
friction_force = H * fLms + K * fms;  % 4x1
control_force = tau;                  % 4x1

%% Lực tổng trong hệ tọa độ robot
net_force_body = J_R' * (control_force - friction_force);  % 3x1

%% Lực trọng lực theo mặt nghiêng
alpha = incline_angle(1);   % rad
beta  = incline_angle(2);   % rad
```

```
    M_total = mb + 4*mw;
    gravity_force = [M_total*g*sin(alpha); M_total*g*sin(beta); 0];  % 3x1

    %% Ma trận khối lượng robot
    Mr = diag([M_total, M_total, J]);

    %% Gia tốc q_ddot
    q_ddot = Mr \ (net_force_body + gravity_force);  % 3x1
end
```

Danh sách thiết bị

Các linh kiện sử dụng trong AGV Robot mong muốn

1. Động cơ Encoder JGA25-371

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
Điện áp định mức	12 V	
Dòng điện không tải	46 mA	
Dòng điện định mức	300 mA	
Tốc độ quay định mức	100 rpm	
Hằng số momen	0.412 N.m	
Hằng số điện áp	0.91 V/(rad/s)	

2. Loadcell 50kg và mạch chuyển đổi HX711

cơ Encoder JGA25-371

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
Tải trọng tối đa	50kg	 
Tín hiệu đầu ra	khoảng 1mV/V – 2mV/V	
Nguồn hoạt động	5V DC	
Độ phân giải ADC của HX711	24 bits	
Giao tiếp với vi điều khiển	Giao tiếp nối tiếp (Serial) 2 dây	

3. Raspberry Pi 4 (RAM 2–4GB)

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
Bộ xử lý (CPU)	Broadcom BCM2711, Lõi tứ Cortex-A72 (ARM v8) 64-bit @ 1,5 GHz	
Bộ xử lý đồ họa	Broadcom VideoCore VI, hỗ trợ OpenGL ES 3.0	
Cổng USB	2xUSB 3.0, 2xUSB 2.0	
GPIO	40 chân GPIO đa năng (tương thích với Pi trước đó)	
Nguồn cấp	5V DC qua USB-C (khuyến nghị nguồn 5V 3A)	
Kích thước	85,6mm x 56,5mm x 17mm	

4. STM32F103C8T6

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
CPU	ARM Cortex-M3, RISC 32 bit @ 72 MHz	
GPIO	37 chân I/O	
Giao tiếp	2 x I2C, 3 x USART, 2 x SPI, 1 x USB 2.0 tốc độ đầy đủ	
ADC	10 kênh ADC 12-bit, tốc độ chuyển đổi 1 µs	
Điện áp hoạt động	2.0 V đến 3.6 V	
PWM	Hỗ trợ xuất tín hiệu điều khiển từ các bộ hẹn giờ, dùng để điều khiển tốc độ cơ sở dữ liệu DC	

Các linh kiện ở AGV Robot trong đề tài

1. Arduino Mega 7960

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
Xung nhịp hoạt động	16MHz	
Kích thước PCB	101,5mm x 53,3mm	
Giao tiếp	Có hỗ trợ SPI, I2C	
Số chân analog	Đầu vào analog 16 chân (độ phân giải 10 bit)	
Điện áp hoạt động	7-12V DC	
Dòng tiêu thụ	Khoảng cách 70 mA (không tải)	

2. BTS7960

Thông số kỹ thuật	Giá trị	Hình ảnh thực tế
Điện áp hoạt động	5 – 27V DC (phiên bản tùy chọn tối đa 40V)	
Dòng tải liên tục	10A – 15A (có nhiệt nhiệt tốt), tối đa lên đến 43A	
Tín hiệu điều khiển	TTL 3.3V hoặc 5V (tương thích với Arduino, STM32, Raspberry Pi, ...)	
Chế độ điều khiển	điều chỉnh tốc độ, chân L/R điều chỉnh chiều	
Bảo vệ tích hợp	Bảo vệ mạch ngắn, quá nhiệt, quá áp, mất pha	
Kích thước PCB	~50 mm x 50 mm (tuỳ mô-đun)	