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 ỨNG DỤNG AI ĐIỀU KHIỂN XE TỰ HÀNH

Người hướng dẫn: **TS. Trần Thị Minh Dung**
Sinh viên thực hiện:
1.Trần Văn Quốc Bảo – MSSV: 105200355 – LỚP: 20TDHCLC1
2.Lê Tuấn Sơn – MSSV: 105200468 – LỚP: 20TDHCLC3

Đà Nẵng, 06/2025

TÓM TẮT

Tên đề tài: Ứng dụng AI điều khiển xe tự hành

Sinh viên thực hiện:

1) Trần Văn Quốc Bảo – MSSV: 105200355- Lớp: 20TDHCLC1
2) Lê Tuấn Sơn – MSSV: 105200468 - Lớp: 20TDHCLC3

Nội dung tóm tắt:

Trong bối cảnh yêu cầu cấp thiết về tự động hóa vận chuyển y tế nhằm giảm thiểu nguy cơ lây nhiễm trong khu vực cách ly, đề tài này tập trung vào thiết kế và triển khai hệ thống robot di động tự hành (Autonomous Mobile Robot – AMR) ứng dụng trí tuệ nhân tạo. Mục tiêu chính là phát triển một nền tảng robot có khả năng vận chuyển thuốc và vật tư y tế một cách tự động, an toàn và hiệu quả trong môi trường y tế phức tạp.

Hệ thống được xây dựng trên nền tảng ROS 2, kết hợp các công nghệ hiện đại như bản đồ hóa và định vị SLAM, xác định vị trí bằng thuật toán AMCL, hoạch định đường đi với thuật toán A*, và phát hiện vật thể động sử dụng mô hình học sâu YOLOv8. Robot sử dụng cấu trúc dẫn động vi sai với bộ điều khiển bám quỹ đạo Backstepping và điều khiển tốc độ động cơ DC bằng thuật toán PID. Toàn bộ hệ thống được mô phỏng và kiểm thử trong môi trường ảo Gazebo và Rviz2.

Kết quả mô phỏng cho thấy robot có khả năng di chuyển ổn định, tránh vật cản, nhận diện người và theo dõi lộ trình hiệu quả trong môi trường mô phỏng tương đương khu cách ly thực tế. Nghiên cứu góp phần khẳng định tính khả thi và tiềm năng ứng dụng của AMR trong lĩnh vực y tế, đặc biệt trong các kịch bản đòi hỏi vận hành liên tục và hạn chế tiếp xúc con người.

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	Trần Văn Quốc Bảo	105200355	20TDHCLC1	Kỹ thuật điều khiển và Tự động hóa
2	Lê Tuấn Sơn	105200468	20TDHCLC3	Kỹ thuật điều khiển và Tự động hóa

1. *Tên đề tài đồ án:* Nghiên cứu ứng dụng AI điều khiển xe tự hành
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:*

Nội dung các phần thuyết minh và tính toán:

4. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ): Bản vẽ Inventor, khổ A4.*

5. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Trần Thị Minh Dung	– Hướng dẫn tư vấn giải pháp, lựa chọn công nghệ cho dự án – Hướng dẫn làm thuyết trình báo cáo dự án – Theo dõi tiến độ của dự án

6. *Ngày giao nhiệm vụ đồ án:* *22/2/2025*
7. *Ngày hoàn thành đồ án:* *24/06/2025*

Đà Nẵng, ngày tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa

TS. Giáp Quang Huy

Người hướng dẫn

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

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

TT	Họ tên sinh viên	Số thẻ SV	Lớp	Ngành
1	Trần Văn Quốc Bảo	105200355	20TDHCLC1	Kỹ thuật điều khiển và Tự động hóa
2	Lê Tuấn Sơn	105200468	20TDHCLC3	Kỹ thuật điều khiển và Tự động hóa

8. Tên đề tài đồ án: Nghiên cứu ứng dụng AI điều khiển xe tự hành

9. *Đề tài thuộc diện: ☐Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*

10. Các số liệu và dữ liệu ban đầu:

11. Nội dung các phần thuyết minh và tính toán:

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Trần Văn Quốc Bảo	Chương 5: Kết Quả thực nghiệm Chương 6: Kết luận, hướng phát triển Tìm nguồn tài liệu tham khảo, hình ảnh minh họa, kiểm tra chính tả định dạng, nội dung và thiết kế hình ảnh mô hình. Thuyết minh, vận hành và kiểm tra. Trình bày báo cáo, mở đầu và kết thúc.
2	Lê Tuấn Sơn	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
1	Lê Tuấn Sơn	Chương 1: TỔNG QUAN VỀ AGV
2	Lê Tuấn Sơn	Chương 2: VẤN ĐỀ NGHIÊN CỨU VÀ THIẾT KẾ KIẾN TRÚC HỆ THỐNG AGV
3	Trần Văn Quốc Bảo	Chương 3: MÔ HÌNH HÓA VÀ PHƯƠNG PHÁP ĐIỀU KHIỂN AGV
4	Trần Văn Quốc Bảo	Chương 4: XÂY DỰNG HỆ THỐNG ĐIỀU HƯỚNG VÀ NÉ VẬT CẢN CHO XE TỰ HÀNH AGV

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

a. Phần chung:

TT	Họ tên sinh viên	Nội dung
1	Trần Văn Quốc Bảo	

TT	Họ tên sinh viên	Nội dung
2	Lê Tuấn Sơn	

b. Phần riêng:

TT	Họ tên sinh viên	Nội dung
	Lê Tuấn Sơn	Bản vẽ Thiết kế, chế tạo xe tự hành, Khung xe, A4
	Lê Tuấn Sơn	Bản vẽ Thiết kế chế tạo xe tự hành, Xe tự hành, A4
	Lê Tuấn Sơn	Bản vẽ CAD kỹ thuật chi tiết khung xe tự hành
	Trần Văn Quốc Bảo	Lưu đồ thuật toán toàn bộ hệ thống

13. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Trần Thị Minh Dung	– Hướng dẫn làm thuyết trình báo cáo dự án – Theo dõi tiến độ của dự án

14. Ngày giao nhiệm vụ đồ án: 22/2/2025

15. Ngày hoàn thành đồ án: 24/6/2025

Đà Nẵng, ngày tháng 6 năm 2025

Trưởng Bộ môn Tự động hóa	**Người 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:

1. Trần Văn Quốc Bảo – MSSV: 105200355 – Lớp: 20TDHCLC1
2. Lê Tuấn Sơn – MSSV: 105200468 – Lớp: 20TDHCLC3

Tên đề tài ĐATN: **Nghiên cứu ứng dụng AI điều khiển xe tự hành**

Họ tên người HD: Trần Thị Minh Dung Đơn vị: ĐHBK-ĐHĐN

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	22/03/2025	5 %	95%	
2	29/03/2025	10%	90%	
3	05/04/2025	20%	80%	
4	12/04/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành 30 %: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	19/04/2025	40%	60%	
6	26/04/2025	50%	55%	
7	03/05/2025	60%	40%	
8	10/05/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành 70 %: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	17/05/2025	80%	20%	
10	24/05/2025	90%	10%	
11	31/05/2025	95%	5%	

12	07/06/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành 90 %: Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	14/06/2025	100%	0%	

LỜI NÓI ĐẦU VÀ CẢM ƠN

Trước tiên, em xin chân thành gửi lời cảm ơn sâu sắc đến quý thầy cô trong khoa Điện, bộ môn Tự động hóa, trường Đại học Bách Khoa, đã tận tình giảng dạy và trang bị cho em những kiến thức quý báu trong suốt quá trình 5 năm học tập, là hành trang vững chắc cho em trên con đường sự nghiệp sau này.

Đặc biệt, em xin bày tỏ lòng biết ơn chân thành đến Cô Trần Thị Minh Dung là người đã trực tiếp hướng dẫn, tận tình chỉ bảo em trong suốt quá trình thực hiện đồ án. Với sự kiên nhẫn, cô đã không ngừng động viên, hỗ trợ và định hướng giúp em vượt qua những khó khăn, hoàn thành đồ án một cách tốt nhất.

Do hạn chế về mặt kiến thức, thời gian và kinh nghiệm thực tế, trong quá trình thực hiện đồ án, nhóm em khó tránh khỏi những thiếu sót mong được quý Thầy/Cô góp ý, chỉ bảo để em rút kinh nghiệm và có điều kiện hoàn thiện thêm kiến thức cho mình, nhằm phục vụ tốt hơn công việc sau này.

Nhóm xem xin chân thành cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Chúng tôi xin được cam đoan đề tài: "Nghiên cứu ứng dụng AI điều khiển xe tự hành" là sự nghiên cứu dưới sự hướng dẫn của giảng viên TS. Trần Thị Minh Dung. Ngoài ra không có bất cứ sự sao chép của người khác. Nội dung báo cáo là sản phẩm mà các thành viên trong nhóm đã nghiên cứu thực hiện trong quá trình thực hiện đồ án. Các số liệu, kết quả trong báo cáo là hoàn toàn trung thực, nhóm xin hoàn toàn chịu trách nhiệm, kỷ luật của bộ môn, khoa và nhà trường nếu có vấn đề xảy ra.

Sinh viên thực hiện

Lê Tuấn Sơn

Sinh viên thực hiện

Trần Văn Quốc Bảo

MỤC LỤC

DANH MỤC HÌNH ẢNH CHƯƠNG 1

DANH MỤC HÌNH ẢNH CHƯƠNG 2

DANH MỤC HÌNH ẢNH CHƯƠNG 3

DANH MỤC HÌNH ẢNH CHƯƠNG 4

DANH MỤC HÌNH ẢNH CHƯƠNG 5

DANH MỤC BẢNG CHƯƠNG 2

DANH MỤC BẢNG CHƯƠNG 3

DANH MỤC BẢNG CHƯƠNG 4

DANH MỤC BẢNG CHƯƠNG 5

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU:

b: khoảng cách từ tâm trục đến 2 bánh xe

R: bán kính bánh xe

d: là khoảng cách tâm trục bánh xe đến tâm Robot

θ: góc quay của robot

X_w, Y_w: hệ tọa độ toàn cục

X_r, Y_r : hệ tọa độ robot

x, y: ví trí của tâm trục bánh xe trong hệ toạ độ toàn cục

ω: tốc độ góc quay của xe

V_x: vận tốc theo hướng đi của xe

V_y: vận tốc ngang

$\overrightarrow{r_{CA}}$, $\overrightarrow{r_{BA}}$: véc-tơ vị trí từ điểm C đến điểm A và vector vị trí từ điểm B đến điểm A

$\overrightarrow{V_A}$: vận tốc tiếp tuyến tâm bánh xe

$\overrightarrow{V_B}$: vận tiếp tuyến tại tâm bánh xe phải

$\overrightarrow{V_C}$: vận tiếp tuyến tại tâm bánh xe phải

$\dot{\phi}_r$: vận tốc góc bánh phải

$\dot{\phi}_L$vận tốc góc bánh trái

$\dot{x}, \dot{y}, \dot{\theta}$: vận tốc tuyến tính và góc quay của robot trong hệ tọa độ thế giới

$\overrightarrow{V_G}$: vận tốc robot khi tâm xe không nằm ở trục bánh xe

L: hàm Lagrange của hệ

q: vector toạ độ tổng quát

C(q): ma trận ràng buộc Pfaffian

λ: nhân tử Lagrange

Q là lực tổng quát tác động của hệ

$\dot{q}$: vector vận tốc tổng quát

K_c: động năng khung xe

I_G: momen quán tính của robot (khung xe) quanh trục z qua trọng tâm (G)

m: khối lượng robot

Ω_ω: vận tốc góc của bánh xe khi bao gồm chuyển động quay của robot

I_ω: tensor quán tính toàn cục

I_ω^G: tensor quán tính gắn với vật

K_R: động năng bánh phải

K_L: động năng bánh trái

m_T: tổng khối lượng xe khi chưatair

m_ω: khối lượng bánh xe

m_G: khối lượng khung xe

I_T: tổng momen quán tính quay

τ_1, τ_2là lực tác động lên bánh xe

R_a: điện trở phần ứng

L_a : điện cảm cuộn dây

E_a: suất điện động cảm ứng

I_a : dòng điện đặt trong từ trường

K_e : hệ số cấu tạo

ϕ : từ thông

M_{dt}: momen điện từ

M_c: momen cản

q_r: tọa độ mẫu

e_p: sai lệch tọa độ

F_f: lực ma sát lăn

μ: Hệ số ma sát lăn với bề mặt sàn bê tông

g: Gia tốc trọng trường

G_x: thành phần trọng lực theo phương chuyển động

F_i: lực quán tính khi tang tốc

F_t: tổng lực kéo cần thiết

P: công suất

$(x_{t-1}, y_{t-1}, \theta_{t-1})$: tráng thái trước của Robot

z_t^i: khoảng cách và góc giữa robot và landmark

δx là bước cập nhật nhỏ.

$\overline{X_t}$: mẫu Particle tạm thời

ω_{short}: trọng số ngắn hạn

ω_{long}: trọng số dài hạn

P_{random}: xác suất thêm Particle ngẫu nhiên

(x_{lidar}, y_{lidar}): vị trí vật cản trong hệ robot

(x_{lidar}, y_{lidar}): vị trí vật cản trong hệ robot

$T_{robot \to map}$: ma trận đồng nhất 4x4 mô tả vị trí và hướng của robot trong bản đồ

CHỮ VIẾT TẮT:

AV: Autonomous Vehicle (Xe tự hành)

ROS: Robot Operating System (Hệ thống vận hành Robot)

PID: Proportional Integral Derivative (Bộ điều khiển vi tích phân tỉ lệ)

SLAM: Simultaneous Localization And Mapping (Định vị và vẽ bản đồ đồng thời)

AMCL: Adaptive Monte Carlo Localization (Định vị Monte – Carlo thích nghi)

MCL: Monte Carlo Localization (Định lí Monte – Carlo)

LIDAR: Light Detection And Ranging (Xác định ánh sáng và phạm vi)

PWM: Pulse Width Modulation (Điều chế độ rông xung)

DC: Direct Current (Điện một chiều)

OMG: Occupancy Grid Maps (Bản đồ lưới chiếm dụng)

UART: Universal Asynchronous Receiver/Transmitter (Giao thức gói tin người dung)

PF: Particle Filter

AGV: Automation Guided Vehicle (Xe tự hành theo đường đi cố định)

AMR: Autonomous Mobile Robot (Robot di động tự động)

MQTT: Messeage Queing Telemtry Transport (Giao thức truyền thông MQTT)

KBBC: Kinematics Based Backstepping Controller

YOLO: You only look one

DDMR: DIFFERENTIAL DRIVE MOBILE ROBOT (Robot di động vi sai)

AI: Artificial Intelligence

CSI: Camera Serial Interface

Rviz2: Ros Visualization 2

CPU: Central Processing Unit

IOT: Internet of things

MỞ ĐẦU

1. Mục đích thực hiện đề tài

Trong bối cảnh các dịch bệnh truyền nhiễm nguy hiểm như COVID-19, Ebola hay cúm A/H5N1 luôn tiềm ẩn nguy cơ tái bùng phát, việc đảm bảo an toàn tuyệt đối trong các khu vực cách ly trở thành một yêu cầu bắt buộc, không chỉ đối với người bệnh mà còn đối với toàn bộ lực lượng y tế. Một trong những thách thức lớn nhất là làm sao duy trì được chuỗi cung ứng thuốc men, vật tư y tế và nhu yếu phẩm một cách liên tục, chính xác mà không tạo thêm nguy cơ lây nhiễm chéo thông qua tiếp xúc người – người.

Đại dịch COVID-19 đã phơi bày rõ những giới hạn trong cách thức tổ chức vận hành khu cách ly truyền thống: hàng ngàn nhân viên y tế phải tiếp xúc trực tiếp với bệnh nhân chỉ để thực hiện các công việc lặp đi lặp lại như đưa thuốc, giao thực phẩm, chuyển mẫu xét nghiệm; điều đó làm gia tăng mức độ lây nhiễm chéo. Khi sức người là hữu hạn, và rủi ro lây nhiễm là hiện hữu, việc áp dụng hệ thống xe tự hành để đảm nhận các tác vụ vận chuyển không còn là lựa chọn phụ, mà là một lời giải thiết yếu. Trong bối cảnh đó, đề tài hướng tới "Thiết kế và triển khai xe tự hành thông minh phục vụ vận chuyển thuốc và vật tư y tế trong khu vực cách ly" được hình thành với một mục tiêu cấp thiết: giảm thiểu tối đa sự tiếp xúc trực tiếp giữa người với người, qua đó bảo vệ sự an toàn của nhân viên y tế, đồng thời đảm bảo chuỗi cung ứng thuốc và vật tư y tế diễn ra liên tục, chính xác và hiệu quả.

Không chỉ đơn thuần là một phương tiện vận chuyển, xe tự hành trong đề tài này là sự kết tinh của các công nghệ tiên tiến: trí tuệ nhân tạo (AI) giúp nhận diện và ra quyết định thông minh, SLAM để xây dựng bản đồ và định vị trong không gian hẹp, thuật toán điều hướng A* đảm bảo tìm đường đi tối ưu ngay cả trong môi trường phức tạp, và cảm biến LiDAR hỗ trợ phát hiện vật cản với độ chính xác cao. Tất cả hội tụ thành một hệ thống thông minh, có thể hoạt động bền bỉ trong môi trường đòi hỏi độ tin cậy và an toàn tuyệt đối.

Lựa chọn đề tài này không chỉ xuất phát từ khát vọng chinh phục công nghệ, mà còn là sự cam kết của nhóm thực hiện đối với trách nhiệm xã hội – mang công nghệ đến đúng nơi cần nhất, phục vụ con người trong hoàn cảnh khó khăn nhất. Đề tài hướng đến tương lai mà công nghệ không chỉ hiện diện trong các dây chuyền sản xuất hiện đại, mà còn

hiện hữu ở những nơi đang cần sự hỗ trợ thầm lặng, nhưng mang ý nghĩa sống còn. với tính ứng dụng cao, định hướng rõ ràng và nền tảng công nghệ vững chắc, đề tài này được kỳ vọng sẽ góp phần thiết thực vào việc hiện đại hóa hoạt động y tế, đặc biệt trong bối cảnh các thách thức về an toàn và nhân lực ngày càng gia tăng.

2. Mục tiêu đề tài

- Nghiên cứu thành công đề tài xe tự hành.
- Tạo ra một sản phẩm hoạt động linh hoạt có khả năng thích ứng với môi trường, ứng dụng được thực tế trong đời sống hằng ngày.
- Thiết kế robot tự hành có thể hoạt động được với các thuật toán trên Ros2 để: xây dựng bản đồ, di chuyển tới vị trí bản đồ theo điểm đặt, tránh vật cản, di chuyển theo đường đi ngắn nhất, có thể sử dụng AI trong đồ án.

3. Phạm vi và đối tượng nghiên cứu

Phạm vi:

Nghiên cứu và chế tạo xe tự hành. Robot tự hành hoạt động trong điều kiện địa hình bằng phẳng, không trượt:

- Khối lượng tổng của Robot ước tính: 11.5kg
- Vận tốc di chuyển tối đa: 0.8 (m/s)
- Thời gian làm việc ước tính: 5 (giờ)

Đối tượng nghiên cứu:

- Hệ thống robot tự hành sử dụng nền tảng **ROS 2**.
- Các thuật toán điều khiển và điều hướng hiện đại phù hợp cho robot DDMR trong môi trường thực tế.
- Các công nghệ hỗ trợ như: cảm biến LIDAR, camera AI, vi điều khiển nhúng, bộ truyền động DC encoder.
- Các phần mềm hỗ trợ phát triển như Gazebo, RViz, ROS 2 packages, YOLOv8, và phần mềm mô phỏng mô hình điều khiển.

4. Phương pháp nghiên cứu

- Khảo sát lý thuyết và công nghệ: Tìm hiểu tổng quan về xe tự hành cùng các công nghệ liên quan như hệ thống điều khiển tự động, điện tử, cơ khí và cảm

biến. Đồng thời, nhóm cũng nghiên cứu các mô hình AMR đã được triển khai và ứng dụng thực tế.

- Nghiên cứu tài liệu chuyên ngành: Trong suốt quá trình thực hiện, nhóm thường xuyên tham khảo các tài liệu học thuật, sách kỹ thuật, bài báo và các công trình nghiên cứu có liên quan để nâng cao kiến thức chuyên môn và hỗ trợ quá trình thử nghiệm thực tế.
- Thiết kế: Sử dụng phần mềm Inventor thiết kế mô hình 3D của xe và khung xe AMR.
- Gia công và chế tạo: Dựa trên mô hình thiết kế trên Inventor. Tiến hành vẽ bản vẽ cắt nhựa mica bằng Auto Cad sau đó tiến hành gia công và lắp ráp toàn bộ linh kiện để hoàn thiện xe.
- Lựa chọn công nghệ và giải pháp kỹ thuật: So sánh và đánh giá các công nghệ khả thi để thiết kế và chế tạo AMR như loại động cơ phù hợp, giao thức truyền thông, cảm biến và nền tảng phần mềm điều khiển.
- Xác định mục tiêu nghiên cứu: Làm rõ định hướng và đặt ra những mục tiêu cụ thể cho quá trình thực hiện đồ án tốt nghiệp.
- Thử nghiệm và đánh giá: Tiến hành các thí nghiệm và thử nghiệm trên môi trường ảo và thực tế để kiểm tra và đánh giá hiệu suất của công nghệ và giải pháp xe tự hành
- Phân tích và định hướng phát triển: Nhóm đánh giá các kết quả đã đạt được, xác định những điểm còn hạn chế và đề xuất các giải pháp cải tiến hoặc định hướng mở rộng trong tương lai.

5. Bố cục đồ án

CHƯƠNG 1: TỔNG QUAN VỀ XE TỰ HÀNH

Trình bày khái niệm, vai trò và ứng dụng thực tiễn của xe tự hành; phân tích các hạn chế của AGV truyền thống và xu hướng phát triển sang AMR cùng các công nghệ nền tảng liên quan. Tham khảo một số nghiên cứu trong và ngoài nước nhằm bổ sung góc nhìn tổng quát và làm cơ sở định hướng cho đề tài.

CHƯƠNG 2: NGHIÊN CỨU VÀ THIẾT KẾ KIẾN TRÚC HỆ THỐNG XE TỰ HÀNH

Xác định bài toán nghiên cứu, phân tích các thách thức kỹ thuật trong điều hướng, cảm biến và điều khiển. Đề xuất giải pháp công nghệ và thiết kế kiến trúc tổng thể của hệ thống xe tự hành thông minh.

CHƯƠNG 3: MÔ HÌNH HÓA VÀ CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN

Mô hình hóa động học, động lực học và động cơ. Trình bày các phương pháp điều khiển như PID và Backstepping nhằm đảm bảo ổn định và chính xác cho hệ thống.

CHƯƠNG 4: XÂY DỰNG CHỨC NĂNG ĐIỀU HƯỚNG VÀ NÉ VẬT CẢN CHO XE TỰ HÀNH

Xây dựng hệ thống điều hướng dựa trên camera,LiDAR và sử dụng các thuật toán như YOLO, SLAM, A*,Costmap,AMCL. Tích hợp thuật toán điều khiển để xe tự hành có thể bám đường đi và phản ứng nhanh với môi trường để đảm bảo khả năng tự hành hiệu quả.

CHƯƠNG 5: THI CÔNG

Trình bày quá trình lựa chọn thiết bị, triển khai phần mềm điều khiển và tiến hành kiểm thử mô hình xe trong cả môi trường thực tế và mô phỏng.

CHƯƠNG 6: KẾT LUẬN

Tổng kết các nội dung đã thực hiện, nêu rõ ưu điểm và hạn chế của nghiên cứu. Đề xuất các hướng phát triển trong tương lai nhằm nâng cao hiệu quả, độ chính xác và khả năng ứng dụng của hệ thống xe tự hành.

CHƯƠNG 1: TỔNG QUAN VỀ XE TỰ HÀNH

1.1 Tổng quan về xe tự hành

1.1.1 Xe tự hành là gì?

Xe tự hành (Autonomous Vehicle – AV) là phương tiện có khả năng tự di chuyển và thực hiện nhiệm vụ mà không cần sự điều khiển trực tiếp từ con người. Thông qua việc tích hợp các công nghệ hiện đại như cảm biến (LiDAR, camera, GPS), trí tuệ nhân tạo (AI) và các thuật toán điều khiển tự động, xe tự hành có thể tự nhận biết môi trường xung quanh, phát hiện vật cản, lập kế hoạch đường đi và di chuyển đến vị trí mong muốn một cách chính xác và an toàn. Trong bối cảnh công nghiệp 4.0, xe tự hành đang trở thành một giải pháp thông minh, góp phần tự động hóa các quy trình vận chuyển trong nhiều lĩnh vực như logistics, y tế, nông nghiệp và sản xuất.

Hình 1.1: Robot di động tự hành (Autonomous Mobile Robot – AMR)

1.1.2 Phạm vi ứng dụng và vai trò hiện tại của xe tự hành

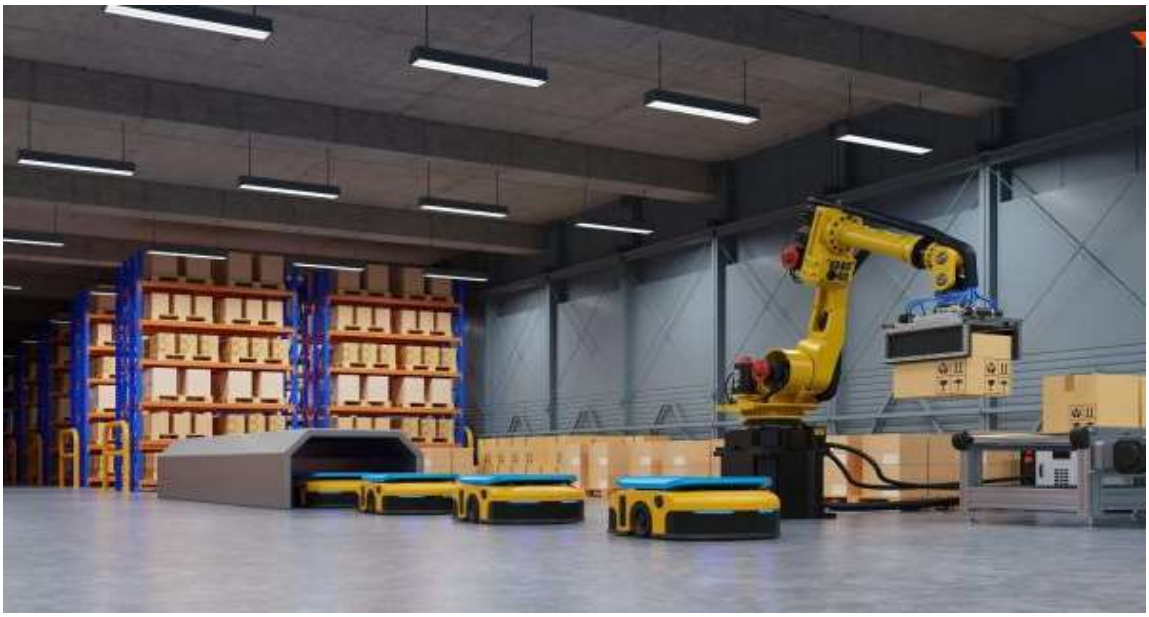
1.1.2.1 Khả năng ứng dụng của xe tự hành

Xe tự hành có tiềm năng ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau:

- **Giao thông công cộng:** Xe buýt hoặc taxi tự hành giúp giảm ùn tắc và nâng cao mức độ an toàn giao thông đô thị.

Hình 1.2: Mercedes - Benz Future Bus – Xe buýt tự hành đầu tiên trên thế giới

- **Vận tải hàng hóa:** Xe tải tự hành được sử dụng trong lĩnh vực logistics và các khu công nghiệp để vận chuyển hàng hóa, giảm chi phí lao động và tăng hiệu quả vận hành.



Hình 1.3: Ứng dụng của xe tự hành trong nhà kho thông minh

- **Quân sự:** AV (Autonomous Vehicle) phục vụ các nhiệm vụ nguy hiểm như trinh sát, vận chuyển vật tư trong vùng chiến sự, giảm rủi ro cho con người.

Hình 1.4: Robot tự hành sử dụng trong quân sự

- **Y tế:** Xe tự hành hỗ trợ vận chuyển thuốc, trang thiết bị, thậm chí bệnh nhân trong bệnh viện hoặc xe cấp cứu tự lái cho các tình huống khẩn cấp.

Hình 1.5: Vibot - Robot y tế hiện đại được lắp đặt tại bệnh viện Bạch Mai khu vực cách ly bệnh nhân covid-19

- **Cá nhân:** Các hệ thống xe tự hành hỗ trợ người cao tuổi, người khuyết tật di chuyển an toàn và thuận tiện hơn trong sinh hoạt hàng ngày.

Hình 1.6: Xe lăn điện Avest tại Tokyo

1.1.2.2 Mục đích sử dụng chính của AV trong đời sống

Mục đích chính của xe tự hành (AV) là thay thế con người trong các nhiệm vụ di chuyển, giám sát và vận chuyển tự động, đặc biệt trong các môi trường lặp đi lặp lại, nguy hiểm hoặc đòi hỏi độ chính xác cao. Với khả năng tự định vị, tự lập kế hoạch đường đi, tránh vật cản và phản ứng linh hoạt, AV ngày càng được ứng dụng rộng rãi nhằm:

- **Tự động hóa quy trình** vận chuyển trong kho xưởng, nhà máy, bệnh viện, sân bay và các tòa nhà thông minh.
- **Tăng hiệu suất lao động**, tối ưu hóa thời gian xử lý và giảm chi phí nhân công.
- **Nâng cao mức độ an toàn**, hạn chế sự can thiệp của con người vào các khu vực độc hại, nguy hiểm hoặc không gian hẹp.
- **Giám sát không gian**, tuần tra tự động, phát hiện sự cố và hỗ trợ thu thập dữ liệu tại các vị trí khó tiếp cận hoặc nguy hiểm.
- **Tăng tính linh hoạt trong vận hành**, dễ dàng thay đổi lộ trình mà không cần sửa đổi cơ sở hạ tầng vật lý.
- **Tối ưu quản lý vận hành trong hệ thống thông minh**, khi được kết nối với IoT, hệ thống điều phối hoặc phần mềm quản lý trung tâm.

Tóm lại, xe tự hành được ứng dụng để làm cho các hoạt động trở nên tự động, an toàn và hiệu quả hơn trong đời sống hiện đại.

1.1.3 Triển vọng công nghệ và mở rộng ứng dụng của xe tự hành

Xe tự hành (Autonomous Vehicle – AV) là khái niệm bao quát nhiều loại phương tiện tự động di chuyển, trong đó phổ biến nhất hiện nay là AGV (Automated Guided Vehicle) và AMR (Autonomous Mobile Robot). AGV là thế hệ AV truyền thống, trong khi AMR được xem là xu hướng phát triển hiện đại nhờ khả năng điều hướng linh hoạt và thông minh hơn.

1.1.3.1 Hạn chế của AGV truyền thống

AGV (Automated Guided Vehicles) là phương tiện tự động sử dụng đường dẫn cố định như băng từ, dây dẫn hoặc mã QR để vận chuyển vật liệu trong nhà máy, kho bãi. Dù giúp tăng hiệu suất và giảm chi phí lao động, AGV truyền thống vẫn còn nhiều hạn chế, thúc đẩy sự phát triển của các công nghệ tự hành linh hoạt hơn.

a. Đường đi cố định, phụ thuộc vào vạch từ, mã QR hoặc dải từ tính

AGV truyền thống sử dụng các hệ thống dẫn đường cố định như băng từ, dây dẫn, hoặc mã QR để điều hướng. Điều này giới hạn khả năng linh hoạt của chúng trong việc thích nghi với các thay đổi về bố trí hoặc môi trường làm việc. Khi bố trí nhà máy hoặc kho bãi thay đổi, toàn bộ hệ thống dẫn đường cần được tái cấu hình, gây tốn kém thời gian và chi phí [1].

Hình 1.7: Các công nghệ điều hướng được sử dụng trong robot di động

b. Không phản ứng tốt với môi trường linh hoạt

Do phụ thuộc vào các đường dẫn cố định, AGV truyền thống khó thích nghi khi có thay đổi bất ngờ trên đường đi, như vật cản đột xuất hoặc thay đổi trong bố trí mặt bằng [1]. Trong các trường hợp này, hệ thống thường cần con người can thiệp để xử lý, gây gián đoạn hoạt động – đặc biệt trong môi trường động như kho hàng hoặc nhà máy đông đúc.

Bên cạnh đó, AGV truyền thống thường sử dụng các bộ điều khiển đơn giản, không khai thác đầy đủ mô hình động học và trạng thái vận hành tức thời của xe [1]. Điều này dẫn đến phản ứng chậm hoặc thiếu chính xác trước các tình huống bất định, làm giảm độ ổn định và chính xác của quá trình di chuyển.

c. Chi phí đầu tư cao cho một số mô hình chuyên biệt

Đối với các ứng dụng yêu cầu cấu hình đặc biệt như tải trọng lớn, môi trường khắc nghiệt hoặc điều kiện hoạt động liên tục, việc triển khai AGV truyền thống đòi hỏi các thiết kế riêng biệt về cơ khí, cảm biến và phần mềm điều khiển [2]. Điều này làm gia tăng đáng kể chi phí đầu tư ban đầu.

Bên cạnh đó, hạ tầng đi kèm như hệ thống dẫn hướng, mạng cảm biến cố định hoặc trung tâm điều phối thường phức tạp và khó mở rộng. Khi có nhu cầu thay đổi quy trình hoặc mặt bằng sản xuất, việc tái cấu hình AGV và hạ tầng hỗ trợ đi kèm cũng tốn nhiều thời gian và chi phí, làm giảm tính linh hoạt trong dài hạn [2].

d. Thiếu khả năng tương tác thông minh

AGV truyền thống thiếu khả năng nhận diện vật thể hoặc điều kiện bất thường mà không có sự hỗ trợ của AI, làm hạn chế khả năng tự chủ và hiệu quả [3]. Chúng không thể đưa ra quyết định dựa trên dữ liệu môi trường thời gian thực, dẫn đến việc không thể thích ứng với các tình huống phức tạp [3].

Tóm lại, mặc dù AGV truyền thống đã đóng vai trò quan trọng trong tự động hóa công nghiệp, nhưng các hạn chế về tính linh hoạt, khả năng phản ứng, chi phí triển khai và tích hợp thông minh đang dần bộc lộ rõ. Những yếu tố này tạo ra nhu cầu cấp thiết cho các giải pháp xe tự hành tiên tiến hơn như AMR – vốn có khả năng tự định vị, tránh vật cản, tương tác linh hoạt và dễ dàng thích nghi với môi trường hiện đại.

1.1.3.2 Xu hướng phát triển và triển vọng trong tương lai

a. Xu hướng phát triển AGV sang AMR

Trong bối cảnh Công nghiệp 4.0, các hệ thống robot di động tự hành (AMR) đang dần thay thế AGV truyền thống nhờ khả năng thích nghi linh hoạt và mức độ tự động hóa cao. Một số xu hướng phát triển nổi bật của AMR hiện nay bao gồm:

- **Tăng mức độ tự chủ:** AMR ngày càng thông minh nhờ tích hợp trí tuệ nhân tạo (AI), cho phép tự định vị, tránh vật cản, học hỏi và ra quyết định tối ưu theo thời gian thực.
- **Kết nối hệ thống toàn diện:** AMR có khả năng tích hợp với các nền tảng quản lý như WMS, ERP, giúp tối ưu luồng vận hành và điều phối.
- **Mở rộng cảm biến:** AMR hiện đại tích hợp thêm các cảm biến môi trường (nhiệt độ, khí độc, độ ẩm...) để phục vụ nhiều tác vụ hơn ngoài việc di chuyển.
- **Thiết kế mô-đun, dễ tùy biến:** Cấu trúc mở cho phép dễ dàng thay đổi phần cứng hoặc bổ sung tính năng theo yêu cầu ứng dụng.
- **Ứng dụng đa ngành:** AMR không chỉ giới hạn trong công nghiệp mà còn mở rộng sang y tế, nông nghiệp, bán lẻ, hậu cần và an ninh.

Xu hướng chung là hướng đến các hệ thống AMR có tính linh hoạt, thông minh, kết nối mạnh và khả năng thích nghi cao, đáp ứng tốt các môi trường vận hành ngày càng phức tạp và biến động.

b. Triển vọng tương lai

Trong tương lai, robot di động tự hành (AMR) được kỳ vọng sẽ đóng vai trò trung tâm trong quá trình tự động hóa và chuyển đổi số ở nhiều lĩnh vực khác nhau:

- **Trong công nghiệp:** AMR sẽ dần thay thế lao động thủ công trong các tác vụ vận chuyển nội bộ, góp phần tối ưu hóa chi phí, nâng cao hiệu suất và tăng khả năng thích ứng với thay đổi trong sản xuất [3].
- **Trong y tế:** AMR có thể hỗ trợ vận chuyển mẫu bệnh phẩm, vật tư y tế và thực hiện các nhiệm vụ khử trùng trong môi trường vô trùng, hạn chế tiếp xúc trực tiếp giữa người với môi trường rủi ro [4].

- **Trong đô thị thông minh:** Các robot AMR sẽ đảm nhiệm vai trò giao hàng, tuần tra an ninh hoặc hướng dẫn trong các trung tâm thương mại, sân bay và tòa nhà lớn [5].
- **Trong giám sát môi trường:** AMR tích hợp cảm biến môi trường (nhiệt độ, khí độc, độ ẩm...) để hoạt động hiệu quả tại các khu vực nguy hiểm hoặc khó tiếp cận như hầm mỏ, nhà máy hóa chất hoặc vùng thiên tai [6].

Với khả năng học hỏi, thích nghi và mở rộng chức năng, AMR được xem là nền tảng cốt lõi của các hệ thống robot thông minh thế hệ mới, hứa hẹn góp phần nâng cao chất lượng sống, hiệu quả vận hành và mức độ an toàn trong nhiều lĩnh vực của đời sống hiện đại.

1.2 Các nghiên cứu quốc tế và trong nước về xe tự hành

1.2.1 Các nghiên cứu ngoài nước đã được triển khai

Xe tự hành hiện đã được triển khai trong nhiều ứng dụng thực tế, bao gồm:

- **Y tế:** Trong nghiên cứu của Cheng và cộng sự [7], đã chỉ ra: Robot di động tự hành (AMR) vận chuyển thuốc, mẫu xét nghiệm, hoặc vật tư y tế trong bệnh viện, giảm khối lượng công việc cho y tá, sử dụng ánh sáng UV-C để khử trùng phòng bệnh, giảm nguy cơ lây nhiễm (đặc biệt trong đại dịch COVID-19) tại Trung Quốc, hỗ trợ bệnh nhân trong các bài tập vật lý trị liệu, giảm căng thẳng cho nhân viên y tế.
- **Giám sát môi trường:** Goutam Mukherjee [8] AMR tích hợp cảm biến (nhiệt độ, khí độc, độ ẩm, chất ô nhiễm) để giám sát chất lượng không khí, nước, và đất ở các khu vực nguy hiểm như nhà máy hóa chất, hầm mỏ, hoặc vùng ô nhiễm.
- **Quản lý tài nguyên thiên nhiên:** Drone giám sát các khu vực có nguy cơ cao (sạt lở đất, núi lửa) để cảnh báo sớm.AMR trên mặt đất kiểm tra khí metan trong mỏ than, giảm nguy cơ cháy nổ [8].

1.2.2 Các nghiên cứu trong nước về xe tự hành

Tại Việt Nam, nghiên cứu về xe tự hành đang ở giai đoạn khởi đầu, với sự tham gia của các trường đại học, viện nghiên cứu và doanh nghiệp công nghệ. Dù còn khoảng cách so với các quốc gia phát triển, một số đơn vị trong nước đã đạt được những bước tiến đáng chú ý:

- **Phenikaa Group:** Năm 2021, Phenikaa ra mắt xe tự hành cấp độ 4 đầu tiên tại Việt Nam, có khả năng nhận diện biển báo, người đi bộ và tự điều chỉnh tốc độ

theo tuyến đường. Dự án do Phenikaa Group, ĐH Phenikaa và Phenikaa-X phối hợp thực hiện, đã thử nghiệm tại nhiều khu vực như Khu đô thị Bình Dương [9].

- **FPT Software:** Từ 2017, FPT phát triển xe tự hành cấp độ 2 và hiện tiếp tục hoàn thiện các tính năng như định vị GPS, phản hồi điều khiển, phanh tự động. Công ty định hướng phát triển xe SDV ứng dụng AI và đã được cấp phép thử nghiệm tại các khu công nghệ cao [10].

Việt Nam hiện vẫn trong quá trình xây dựng khung pháp lý phù hợp cho xe tự hành, với trọng tâm nghiên cứu hiện nay là các giải pháp kỹ thuật và đảm bảo an toàn trong môi trường thử nghiệm.

1.3 Tích hợp AI, SLAM, bộ điều khiển phi tuyến, thuật toán A* và các công nghệ hiện đại: Nền tảng công nghệ cho xe tự hành thông minh

Trong các hệ thống AMR hiện đại, việc tích hợp AI, SLAM, bộ điều khiển phi tuyến, thuật toán A* và nhiều công nghệ bổ trợ khác là chìa khóa giúp robot di chuyển tự chủ, linh hoạt và an toàn trong môi trường phức tạp như bệnh viện, nhà máy, kho logistics.

a. Trí tuệ nhân tạo (AI) – Nhận thức và ra quyết định thông minh

AI đóng vai trò trung tâm trong việc xử lý dữ liệu cảm biến và đưa ra hành vi di chuyển:

- Nhận diện người, vật thể, pallet... từ camera, LiDAR nhờ các mô hình học sâu (YOLOv8, CNN).
- Dự đoán tình huống, tránh va chạm và điều chỉnh hành vi theo thời gian thực.
- Phân tích dữ liệu hoạt động để bảo trì dự đoán và tối ưu vận hành.

b. SLAM – Định vị và lập bản đồ thời gian thực

Trong môi trường không có GPS, SLAM giúp AMR định vị và cập nhật bản đồ:

- Xác định chính xác vị trí và xây dựng bản đồ môi trường không có GPS.
- Cập nhật bản đồ khi môi trường thay đổi, hỗ trợ AI và thuật toán A* trong điều hướng.

c. Bộ điều khiển phi tuyến – Ổn định và thích nghi

Bộ điều khiển phi tuyến giúp AMR hoạt động hiệu quả trong điều kiện không lý tưởng:

- Xử lý điều kiện bất định như ma sát, tải trọng, mặt sàn không đều.
- Duy trì quỹ đạo chính xác, giảm tiêu hao năng lượng và nâng cao độ bền thiết bị.

d. Thuật toán A – Lập kế hoạch đường đi tối ưu*

A* là thuật toán điều hướng hiệu quả cho AMR trong môi trường động:

- Tìm đường ngắn nhất với độ an toàn cao.
- Kết hợp với bản đồ SLAM để thích nghi với môi trường thay đổi liên tục.

e. Sensor Fusion (Tổng hợp cảm biến)

Để nâng cao độ chính xác trong nhận thức và định vị, AMR sử dụng cơ chế tổng hợp dữ liệu từ nhiều loại cảm biến khác nhau:

- Kết hợp LiDAR, camera RGB-D, IMU, siêu âm… để tăng độ chính xác nhận thức môi trường.
- Cảm biến encoder và IMU giúp cải thiện định vị và kiểm soát chuyển động.

f. Hệ thống giao tiếp và điều phối

Hệ thống kết nối và điều phối giữ vai trò quan trọng trong việc đảm bảo AMR hoạt động liên tục, ổn định và tích hợp với các nền tảng điều hành chung:

- ROS/ROS2: Nền tảng mã nguồn mở để tích hợp các thành phần robot.
- Kết nối không dây (WiFi 6, 5G): Giao tiếp nhanh, ổn định trong môi trường lớn.
- Giao thức MQTT, OPC-UA: Tương thích với hệ thống giám sát công nghiệp hoặc quản lý bệnh viện.

g. Điều phối đa robot & lập lịch thông minh

Khi triển khai nhiều AMR cùng lúc, cần có hệ thống điều phối để đảm bảo hiệu suất và an toàn:

- Hệ thống điều phối tập trung hoặc phân tán giúp các AMR tránh va chạm và tối ưu hiệu suất.
- Ứng dụng Swarm Intelligence hoặc Task Assignment Algorithms

1.4 Cơ sở định hướng cho đề

Trong quá trình hiện đại hóa công tác vận hành nội bộ, các hệ thống AGV (Automated Guided Vehicle) truyền thống từng đóng vai trò quan trọng, song đang dần bộc lộ nhiều điểm hạn chế: phụ thuộc vào đường dẫn cố định, thiếu khả năng thích ứng với môi trường thay đổi, phản ứng chậm trước tình huống bất ngờ, và khó tích hợp với các nền

tảng điều phối thông minh. Những nhược điểm này khiến AGV khó đáp ứng được các yêu cầu ngày càng linh hoạt và phức tạp trong môi trường dịch vụ, đặc biệt là bệnh viện.

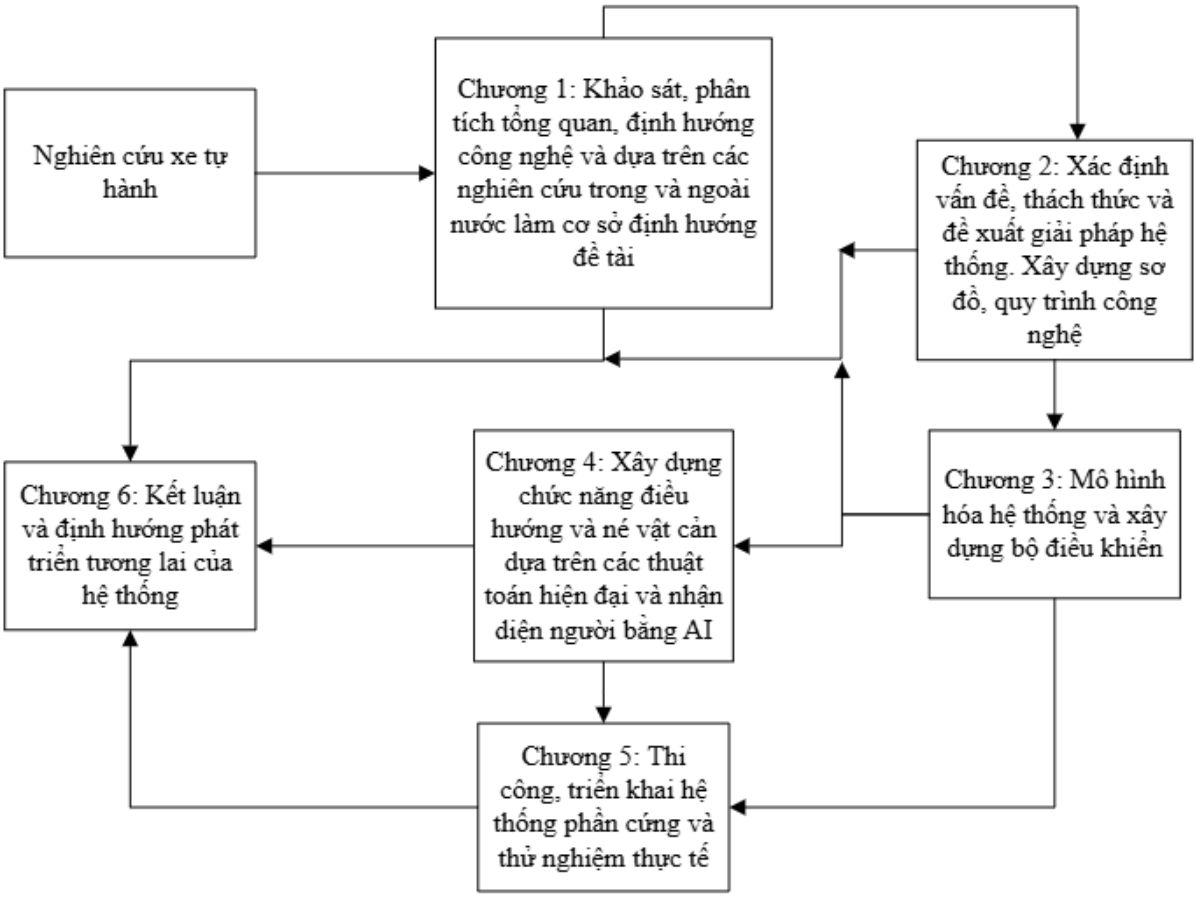
Ngược lại, robot di động tự hành (AMR – Autonomous Mobile Robot) nổi bật nhờ khả năng **tự định vị**, **tự tránh vật cản** và **lập kế hoạch đường đi tối ưu**. Đặc biệt, khi được tích hợp các công nghệ lõi như **AI** (trí tuệ nhân tạo), **SLAM** (Simultaneous Localization and Mapping), thuật toán tìm đường A*, bộ điều khiển phi tuyến và nền tảng ROS, AMR có thể hoạt động hiệu quả, an toàn và thích nghi trong môi trường y tế nhạy cảm – nơi yêu cầu khắt khe về chính xác, vệ sinh và giảm tiếp xúc con người.

Từ thực tiễn triển khai AMR tại các cơ sở y tế quốc tế và phân tích nhu cầu tại các khu điều trị cách ly trong nước, nhóm nghiên cứu xác định hướng phát triển một hệ thống AMR thông minh phục vụ **vận chuyển thuốc**, **vật tư y tế và mẫu bệnh phẩm**. Hệ thống này sẽ **giải quyết bài toán nhân lực**, **giảm nguy cơ lây nhiễm chéo**, đồng thời **tận dụng hiệu quả các công nghệ hiện đại để vận hành tự động**, linh hoạt và có khả năng mở rộng trong tương lai.

Mục tiêu cuối cùng là thiết kế và triển khai một giải pháp robot y tế có khả năng tự hành thực sự, hoạt động ổn định trong các khu cách ly – góp phần nâng cao hiệu quả vận hành, tăng cường năng lực phòng chống dịch và hiện đại hóa hạ tầng bệnh viện trong kỷ nguyên chuyển đổi số.

1.5 Xây dựng quy trình nghiên cứu hệ thống xe tự hành

Để đảm bảo quá trình thiết kế và phát triển hệ thống xe tự hành diễn ra một cách khoa học, hiệu quả và có tính ứng dụng thực tế cao, nhóm nghiên cứu đã xây dựng một quy trình nghiên cứu gồm nhiều giai đoạn kế tiếp nhau, từ khảo sát thực tiễn, phân tích yêu cầu, đến thiết kế, phát triển và thử nghiệm hệ thống. Quy trình này được xây dựng dựa trên các nguyên tắc nghiên cứu – phát triển (R&D) hiện đại, kết hợp giữa lý thuyết nền tảng và thực tiễn triển khai trong môi trường ứng dụng cụ thể là khu vực cách ly y tế.



Hình 1.8: Quy trình nghiên cứu và phát triển hệ thống xe tự hành.

CHƯƠNG 2: NGHIÊN CỨU VÀ THIẾT KẾ KIẾN TRÚC HỆ THỐNG XE TỰ HÀNH

2.1 Đặt vấn đề

Trong hệ thống chăm sóc sức khỏe hiện đại, việc vận chuyển thuốc, vật tư y tế và mẫu bệnh phẩm giữa các khu vực chức năng – đặc biệt tại các **khu cách ly hay khu điều trị bệnh truyền nhiễm** – là một mắt xích quan trọng nhưng thường không được chú trọng đúng mức. Tuy chỉ là công việc lặp đi lặp lại với yêu cầu kỹ thuật đơn giản, nhưng nhiệm vụ này đòi hỏi tính chính xác cao, đúng thời gian và đảm bảo tuyệt đối về an toàn sinh học.

Hiện nay, phần lớn hoạt động vận chuyển trong bệnh viện vẫn **do nhân viên thực hiện** thủ công – một **mô hình tiêu tốn nhân lực và tiềm ẩn nhiều rủi ro**, đặc biệt là nguy cơ **lây nhiễm chéo** qua tiếp xúc trực tiếp. Áp lực càng gia tăng trong bối cảnh các khu cách ly phải hoạt động liên tục 24/7, trong khi **nguồn nhân lực lại hạn chế**.

Dù khối lượng vận chuyển mỗi lần không lớn (thường chỉ từ 1÷2 kg), nhưng loại hình hàng hóa lại rất đa dạng – từ thuốc theo toa, vật tư tiêu hao như bông gạc, nhiệt kế, đến nhu yếu phẩm hàng ngày như khẩu trang, đồ ăn đóng gói, nước uống, quần áo bệnh nhân. Dù là vật phẩm gì, yêu cầu về độ chính xác, đúng giờ và hạn chế tối đa sai sót luôn là ưu tiên hàng đầu.

Trong bối cảnh đó, việc ứng dụng **robot di động tự hành** (Autonomous Mobile Robot – AMR) đang dần trở thành **một giải pháp khả thi**. Tuy nhiên, trong môi trường bệnh viện, đặc biệt là khu cách ly rất khác so với các nhà máy hay kho hàng nơi AMR thường được triển khai**. Robot cần khả năng di chuyển linh hoạt** trong hành lang hẹp, tự tránh vật cản, phản ứng nhanh với sự xuất hiện bất ngờ của con người, và thích nghi với sự thay đổi liên tục về mặt bằng cũng như quy trình vận hành.

Điều này đòi hỏi **một hệ thống điều hướng thông minh** – có thể tự định vị, xây dựng bản đồ, lập kế hoạch lộ trình tối ưu và duy trì hoạt động ổn định, an toàn trong một môi trường thực tế nhiều ràng buộc như bệnh viện.

2.2 Các thách thức của hệ thống xe tự hành trong đề tài

Để xây dựng được một hệ thống AMR đáp ứng các yêu cầu khắt khe về độ ổn định, an toàn và khả năng mở rộng trong môi trường thực tế, nhóm nghiên cứu phải giải quyết nhiều bài toán kỹ thuật mang tính liên ngành. Các thách thức này không chỉ đến từ đặc thù môi trường triển khai, mà còn từ yêu cầu về tính thông minh, độ chính xác và hiệu quả vận hành. Cụ thể, hệ thống cần vượt qua những khó khăn sau:

- **Điều hướng trong không gian bệnh viện phức tạp, thiếu giám sát trực tiếp:** Các khu vực như hành lang bệnh viện, phòng cấp phát thuốc, phòng chứa mẫu virus hoặc tầng hầm kỹ thuật thường có cấu trúc không gian hẹp, nhiều ngã rẽ, vật cản cố định và ánh sáng yếu. AMR cần khả năng tự định vị, xây dựng bản đồ môi trường và hoạch định đường đi tối ưu trong điều kiện thông tin hạn chế và không có sự giám sát thường xuyên từ nhân viên y tế.
- **Xử lý và né tránh vật cản động trong môi trường bất định:** Trong các khu vực cách ly hoặc phòng chức năng đặc biệt, có thể xuất hiện các tình huống bất ngờ như nhân viên y tế di chuyển khẩn cấp, thiết bị y tế tạm thời để trên đường đi hoặc vật cản do rơi đổ. Hệ thống robot phải có khả năng phát hiện nhanh, nhận dạng chính xác và phản ứng kịp thời để đảm bảo an toàn tuyệt đối cho cả con người lẫn robot.
- **Điều khiển chính xác trong hệ dẫn động phi tuyến:** Robot sử dụng cơ cấu dẫn động vi sai – vốn mang tính phi tuyến và nonholonomic, gây khó khăn trong điều khiển chính xác khi di chuyển ở tốc độ thấp, quay đầu tại hành lang hẹp hoặc tiếp cận vị trí giao – nhận vật tư. Đề tài lựa chọn bộ điều khiển backstepping như một giải pháp điều khiển phi tuyến hiệu quả, giúp đảm bảo độ ổn định và mượt mà trong quá trình vận hành.
- **Độ lệch giữa mô phỏng và thực tế trong điều kiện vận hành y tế:** Môi trường bệnh viện có thể phát sinh các yếu tố nhiễu như tín hiệu cảm biến không ổn định do nhiệt độ, độ ẩm cao hoặc vật liệu phản xạ không đồng nhất. Ngoài ra, độ trễ truyền tín hiệu hoặc sai số định vị cũng ảnh hưởng đến hiệu suất vận hành thực tế nếu không được xử lý thích hợp. Đây là rào cản cần khắc phục để hệ thống hoạt động ổn định như khi mô phỏng.

2.3 Giải pháp đề xuất

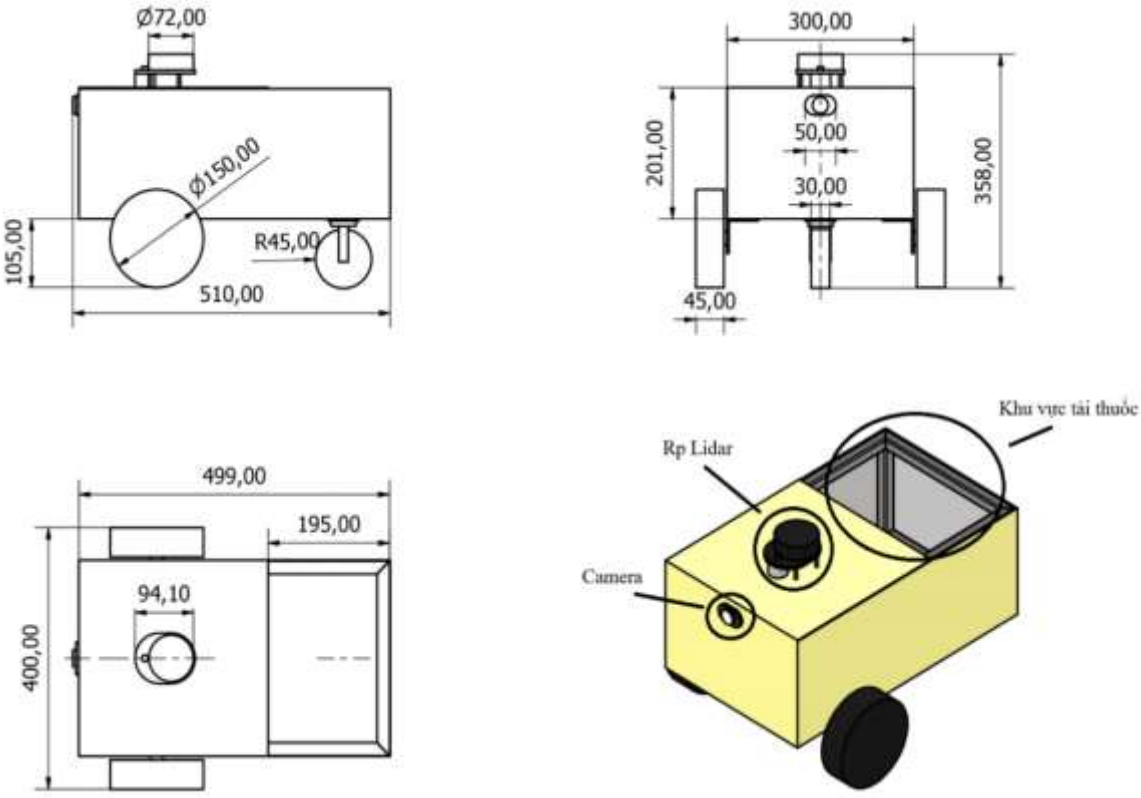
Để giải quyết các thách thức trên, nhóm xây dựng hệ thống AMR tích hợp nền tảng điều phối, điều khiển phi tuyến, nhận thức môi trường và lập kế hoạch di chuyển. Các giải pháp chính gồm:

- **ROS 2 - nền tảng điều phối linh hoạt:** ROS 2 Humble là trung tâm điều phối hệ thống xe tự hành, hỗ trợ xử lý thời gian thực, phân tán và đa luồng. Nhờ đó, các mô-đun như SLAM, A*, xử lý ảnh và điều khiển động cơ được tích hợp hiệu quả. ROS 2 còn hỗ trợ kiểm thử mô phỏng và đồng bộ với phần cứng, giúp giảm rủi ro khi triển khai thực tế.
- **SLAM - bản đồ hóa và định vị động:** SLAM cho phép xe tự hành vừa xây dựng bản đồ vừa định vị trong môi trường thay đổi. Kết hợp với Lidar và ROS 2 (slam_toolbox), hệ thống định vị và cập nhật bản đồ theo thời gian thực.
- **AI - nhận diện vật cản động:** Mô hình học máy chạy trên camera Raspberry Pi giúp phát hiện người và vật thể di chuyển, hỗ trợ cập nhật bản đồ động và điều chỉnh hướng đi. Kết hợp Lidar giúp phát hiện vật cản tĩnh và tăng độ chính xác định vị.
- **Thuật toán A - lập kế hoạch đường đi:** A* tính toán lộ trình ngắn nhất, an toàn nhất dựa trên bản đồ hiện tại, hỗ trợ né tránh vật cản và tái lập kế hoạch khi môi trường thay đổi.
- **Bộ điều khiển KBBC - bám quỹ đạo chính xác:** Dựa trên cơ cấu dẫn động vi sai, bộ điều khiển backstepping (KBBC) được sử dụng để đảm bảo điều khiển ổn định, khử nhiễu tốt và phản ứng hiệu quả với sai lệch giữa mô phỏng và thực tế.

Tổng thể, sự kết hợp giữa ROS 2, SLAM, AI, A*, Lidar và bộ điều khiển KBBC tạo nên hệ thống AMR, thích ứng linh hoạt và điều khiển chính xác trong môi trường thực tế.

2.4 Thiết kế kiến trúc hệ thống điều khiển xe tự hành

2.4.1 Sơ đồ công nghệ



Hình 2.1: Sơ đồ công nghệ

Trong phạm vi đề tài, nhóm thiết kế và triển khai một mô hình robot tự hành thông minh (AMR) với kích thước 50 × 30 × 35 cm, có khoang chở hàng phía sau, nhằm **phục vụ vận chuyển thuốc và vật tư y tế tại các bệnh viện, khu cách ly hoặc bệnh viện dã chiến**. Đây là những môi trường đặc thù, nơi cần giảm thiểu tối đa tiếp xúc trực tiếp giữa người với người, hạn chế nguy cơ lây nhiễm chéo và duy trì hoạt động vận chuyển liên tục, chính xác trong không gian chật hẹp và tiềm ẩn rủi ro.

Robot được thiết kế để **di chuyển linh hoạt trong các hành lang hẹp, tự động tránh vật cản, nhận diện người và thích ứng với các tình huống bất định thường gặp trong môi trường y tế.** Khoang chứa phía sau được bố trí để chở các loại vật tư nhẹ như thuốc dạng chai, bộ test nhanh, găng tay, ống nghiệm, dụng cụ y tế... với tổng khối lượng khoảng 1–2 kg đủ để phục vụ các chuyến vận chuyển nội viện thông thường mà không ảnh hưởng đến độ ổn định khi di chuyển.

Về mặt cơ khí, khung xe sử dụng nhôm định hình kết hợp với các chi tiết dạng hộp nhằm đảm bảo độ cứng vững, nhẹ và thuận tiện cho việc gia công. Hệ dẫn động vi sai sử dụng hai động cơ DC có tích hợp encoder, cho phép điều khiển độc lập từng bánh xe, đồng thời hỗ trợ quay đầu trong không gian hẹp – một yêu cầu quan trọng khi hoạt động trong môi trường hành lang nhiều ngã rẽ.

Về cảm biến, hệ thống gồm:

- LiDAR 360° để tạo bản đồ môi trường, phát hiện và tránh vật cản tĩnh.
- Camera độ phân giải cao phục vụ xử lý hình ảnh và nhận diện người bằng mô hình YOLO.
- Encoder cảm nhận và đo đạc các đại lượng vật lý như vị trí, vận tốc, hướng quay, sau đó chuyển thành tín hiệu điện.

Hệ thống xử lý chia thành hai tầng:

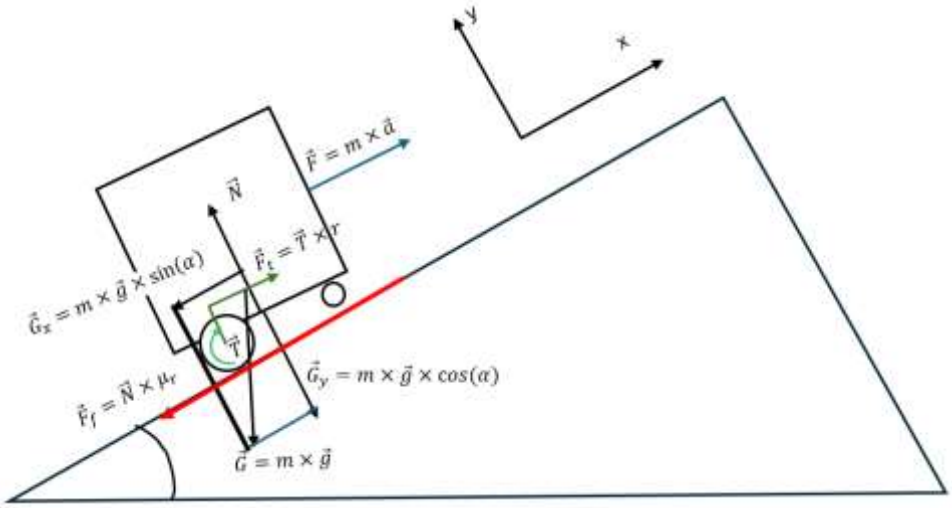
- High-Level: đảm nhiệm các thuật toán định vị, điều hướng và xử lý hình ảnh.
- Low-Level: thu thập dữ liệu từ các cảm biến và điều khiển động cơ thông qua các tín hiệu PWM.

Về điều khiển chuyển động, hệ thống ứng dụng bộ điều khiển Backstepping phi tuyến, phù hợp với đặc tính non-holonomic của dẫn động vi sai, giúp xe bám sát quỹ đạo, di chuyển ổn định và phản ứng tốt trong điều kiện môi trường thực tế.

Toàn bộ mô hình đóng vai trò như một nền tảng thử nghiệm thu gọn, phục vụ đánh giá khả năng tự hành, tránh vật cản và nhận diện người trong môi trường y tế thực – một bước quan trọng hướng tới việc ứng dụng AMR trong thực tiễn bệnh viện hiện đại.

2.4.2 Tính chọn động cơ cho mô hình thiết kế

Dựa trên cơ sở nghiên cứu của Crenganis [11], ta có:



Hình 2.2: Các lực tác động lên robot

Ta có thông số thiết kế ban đầu:

- Khối lượng robot: m =12 Kg
- Gia tốc trọng trường: g = 9.81 m/s^2
- Gia tốc mục tiêu của robot: a = 0.5 m/s^2
- Vận tốc đích của robot: v = 0.8 m/s
- Bán kính bánh xe: r = 0.075 m
- Hệ số ma sát lăn với bề mặt sàn bê tông: $\mu = 0.01$
- Góc nghiêng mặt phẳng chuyển động: $\alpha = 0°$

Ta có công thức tính lực ma sát lăn:

$$F_f = \mu N = \mu \times m \times g \times \cos(\alpha) = 0.01 \times 12 \times 9.81 \times \cos(0) = 1.1772\ N \quad (2.1)$$

Thành phần trọng lực theo phương chuyển động

$$G_x = m \times g \times \sin(\alpha) = 12 \times 9.81 \times \sin(0) = 0\ N \quad (2.2)$$

Lực quán tính khi tang tốc F_i:

$$F_i = m \times a = 12 \times 0.5 = 6\ N \quad (2.3)$$

Tổng lực kéo cần thiết F_t:

$$F_t = F_i + G_x + F_f = 6 + 1.1772 = 7.1772\ N \quad (2.4)$$

Tính momen xoắn yêu cầu T:

Lực kéo liên hệ với momen xoắn thông qua bán kính bánh xe:

$$F_t = \frac{T}{r} \rightarrow T = F_t \times r \quad (2.5)$$

Vì robot sử dụng 2 bánh chủ động, mỗi bánh chịu một nửa lực kéo:

$$T = \frac{F_t \times r}{2} = \frac{7.1772 \times 0.075}{2} = 0.269145 \text{ N.m} \quad (2.6)$$

Tính tốc độ góc của trục bánh xe:

$$\omega = \frac{v}{r} = \frac{0.8}{0.075} = 10.667\ rad/s \quad (2.7)$$

Chuyển sang tốc độ quay trục động cơ:

$$n = \frac{\omega \times 60}{2\pi} = \frac{10.667 \times 60}{2\pi} = 101.94\ rpm \quad (2.8)$$

Tính công suất cơ học yêu cầu P:

$$P = T \times \omega = 0.17943 \times 12 = 2.868\ W \quad (2.9)$$

Từ đây chọn động cơ:

Series 2619 012SR hộp số là: 33:1

Hình 2.3: Động cơ Series 2619 012SR

Bảng 2.1: Thông số động cơ Series 2619 012SR

Series 2619 ... SR

Values at 22°C and nominal voltage	2619 S	006 SR	012 SR	024 SR	
Nominal voltage	U_N	6	12	24	Volt
Terminal resistance	R	8,2	36,5	128	Ω
No-load speed (motor)	n_o	6 600	5 900	6 200	min^{-1}
Speed constant	k_n	1 111	500	261	min^{-1}/V
Back-EMF constant	k_E	0,9	2	3,83	mV/min^{-1}
Torque constant	k_M	8,59	19,09	36,54	mNm/A
Current constant	k_I	0,116	0,052	0,027	A/mNm
Slope of n-M curve	$\Delta n/\Delta M$	1 055	957	917	min^{-1}/mNm
Rotor inductance	L	465	2 200	8 400	µH
Rotor inertia	J	0,68	0,68	0,68	gcm^2

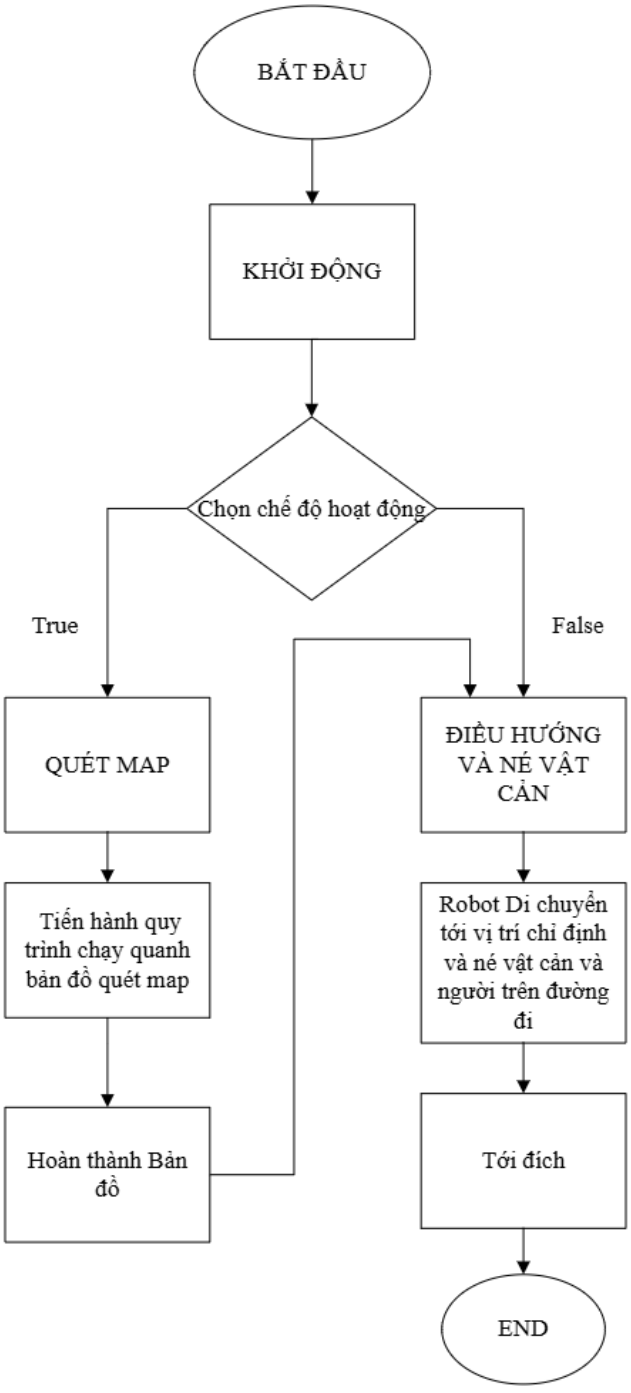
Housing material		plastic		
Geartrain material		metal		
Backlash, at no-load	≤	4		°
Bearings on output shaft		brass / ceramic bearings	ball bearings, preloaded	
Shaft load max.:		(standard)	(optional)	
– radial (5 mm from mounting face)	≤	3,5	10,5	N
– axial	≤	2	5	N
Shaft press fit force, max.	≤	10	10	N
Shaft play:				
– radial (5 mm from mounting face)	≤	0,07	0,03	mm
– axial	≤	0,25	0	mm
Operating temperature range		– 25 ... + 80		°C

Specifications

reduction ratio (rounded)	output speed up to n_{max} min^{-1}	weight with motor g	output torque continuous operation M_{max} mNm	output torque intermittent operation M_{max} mNm	direction of rotation (reversible)	efficiency %
8 : 1	635	25	9	30	=	81
22 : 1	223	26	23	75	≠	73
33 : 1	151	26	30	100	=	66
112 : 1	44	27	93	180	≠	59
207 : 1	24	27	100	180	=	53
361 : 1	14	27	100	180	=	53
814 : 1	6	28	100	180	=	43
1 257 : 1	4	29	100	180	=	43

2.4.3 Quy trình công nghệ

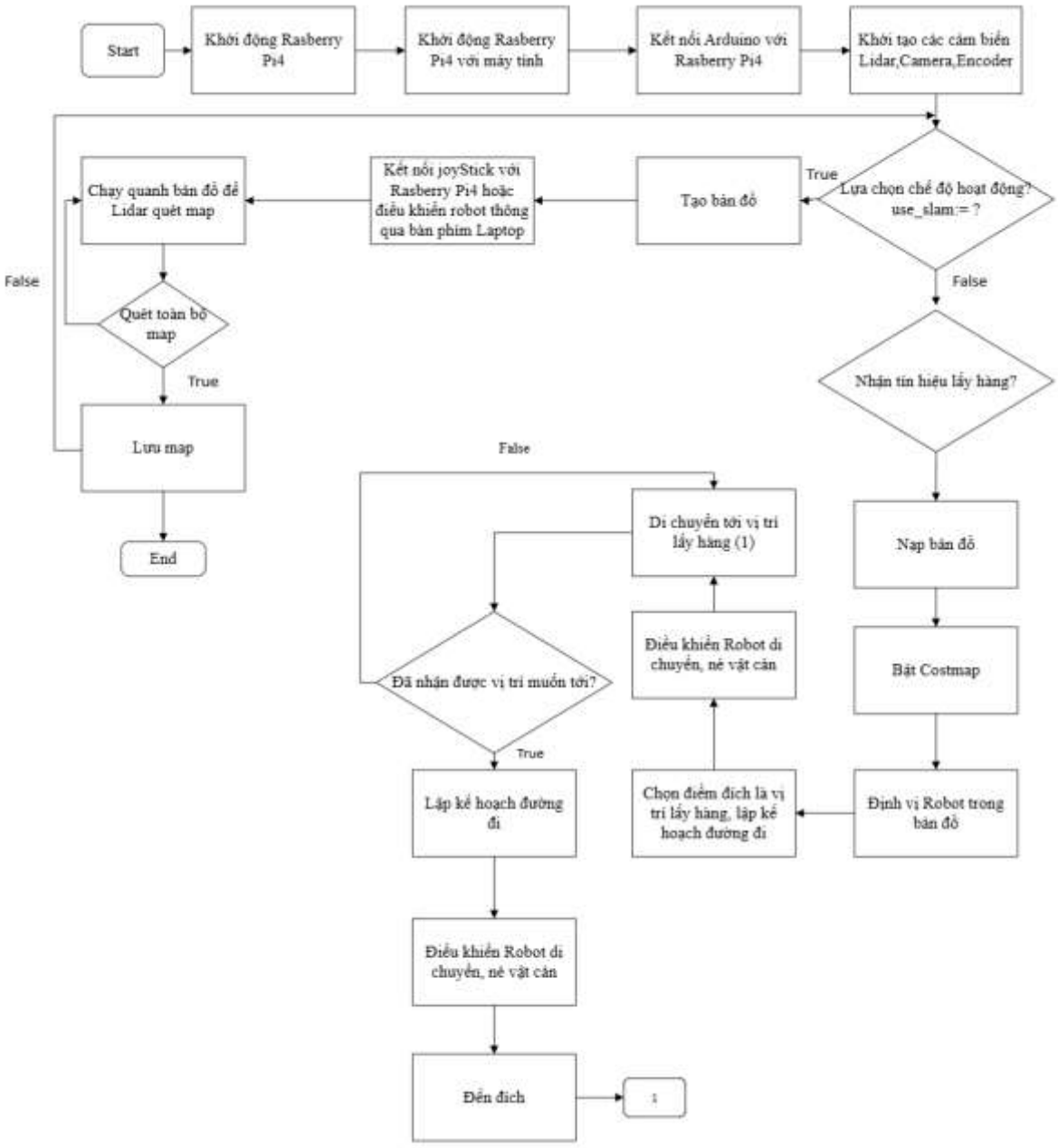
Nhằm hiện thực hóa mục tiêu thiết kế xe tự hành thông minh trong môi trường y tế, nhóm nghiên cứu xây dựng một quy trình công nghệ phù hợp với điều kiện thực tế như sau:



Hình 2.4: Quy trình công nghệ

2.4.4 Lưu đồ thuật toán hoạt động toàn hệ thống

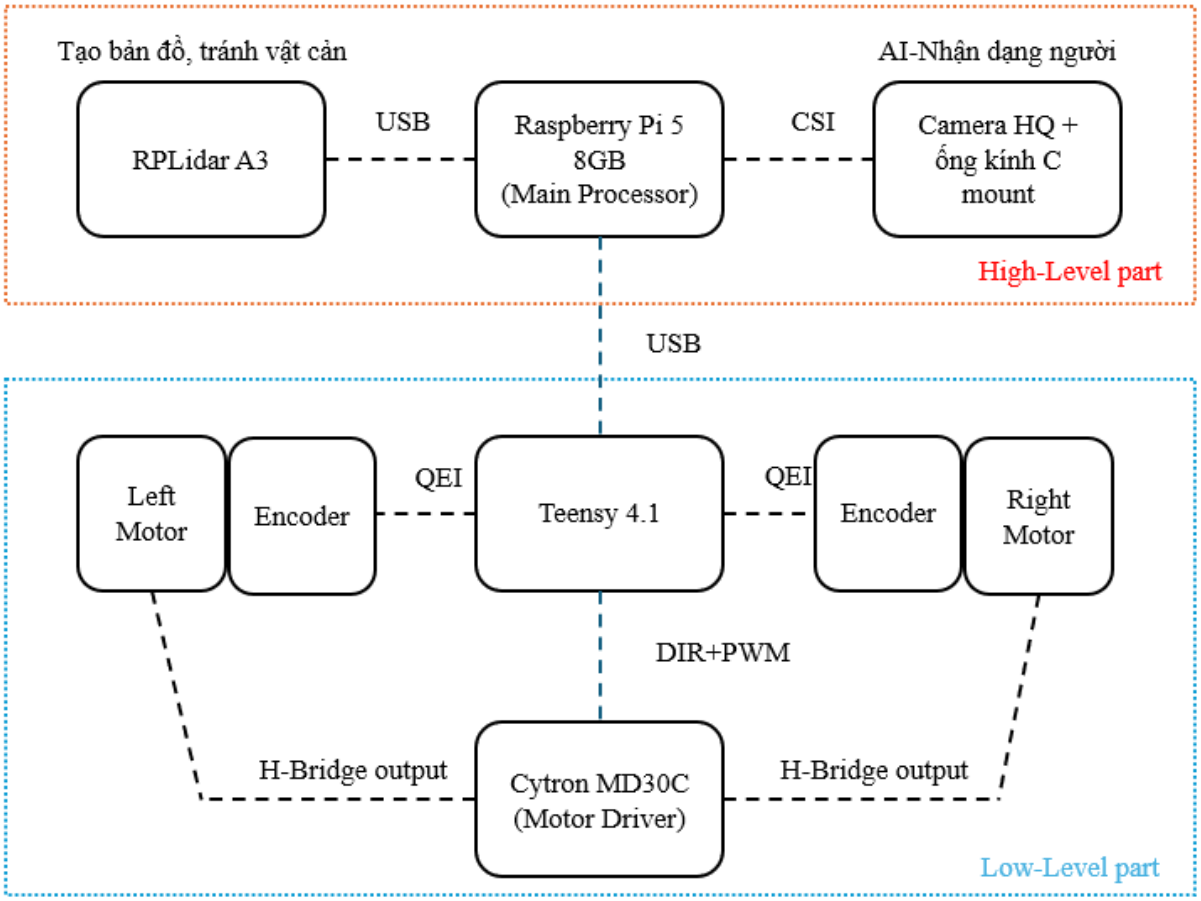
Để đảm bảo hoạt động chính xác và linh hoạt của hệ thống xe tự hành, nhóm đã xây dựng lưu đồ thuật toán tổng thể được thể hiện ở **Hình 2.5**, mô tả rõ quy trình điều khiển từ khâu khởi động, xử lý cảm biến đến điều hướng và điều khiển chuyển động.



Hình 2.5: Lưu đồ thuật toán toàn hệ thống

2.4.5 Phân tầng điều khiển của hệ thống

Hệ thống điều khiển của xe tự hành được thiết kế theo mô hình phân tầng như **Hình 2.6**, nhằm đảm bảo tính linh hoạt, dễ mở rộng và tối ưu hiệu suất xử lý.



Hình 2.6: Mô hình phân tầng điều khiển của hệ thống

2.4.5.1 High-Level part

Tầng điều khiển cao được xử lý bởi Raspberry Pi 5, đóng vai trò là bộ xử lý trung tâm (Main Processor) của hệ thống. Các chức năng chính gồm:

- Xử lý nhận thức và định vị: Raspberry Pi thu thập dữ liệu từ cảm biến LiDAR RPLidar A3 thông qua giao tiếp USB, phục vụ cho việc xây dựng bản đồ (SLAM) và xác định vị trí tương đối của robot trong không gian.
- Xử lý hình ảnh: Camera HQ (IMX477) kết nối với Pi qua giao tiếp CSI, được sử dụng trong các tác vụ thị giác máy như phát hiện người, vật thể, vạch kẻ đường.
- Quản lý lộ trình và giao tiếp ROS 2: Hệ điều hành Ubuntu 22.04 + ROS 2 Humble hoạt động trên Raspberry Pi 5 để thực hiện lập kế hoạch đường đi, tránh vật cản và điều hướng. Các lệnh điều khiển tốc độ và hướng được gửi xuống tầng thấp thông qua UART (USB Serial) đến vi điều khiển Teensy 4.1.

- UART (Universal Asynchronous Receiver Transmitter): Chuẩn giao tiếp nối tiếp không đồng bộ sử dụng hai dây: TX (truyền) và RX (nhận). Đây là kênh liên lạc chính giữa Raspberry Pi và Teensy để truyền lệnh điều khiển.

2.4.5.2 Low-Level part

Tầng điều khiển thấp chịu trách nhiệm thực thi lệnh điều hướng từ tầng cao một cách chính xác, bao gồm các chức năng:

- Teensy 4.1 đóng vai trò là Platform Control Unit (PCU), xử lý tín hiệu phản hồi từ encoder và thực hiện điều khiển động cơ thông qua thuật toán PID được lập trình sẵn.
- Đọc tín hiệu từ encoder: Mỗi động cơ 2619 SR tích hợp encoder, cung cấp 2 kênh xung A và B dạng QEI (Quadrature Encoder Interface). Teensy sử dụng các chân digital có hỗ trợ interrupt để đọc chính xác số xung từ encoder, phục vụ cho việc đo tốc độ và vị trí.
- Xuất tín hiệu PWM + DIR: Sau khi tính toán tốc độ cần thiết, Teensy xuất tín hiệu điều khiển gồm:
    - PWM (xung điều tốc)
    - DIR (xác định chiều quay)

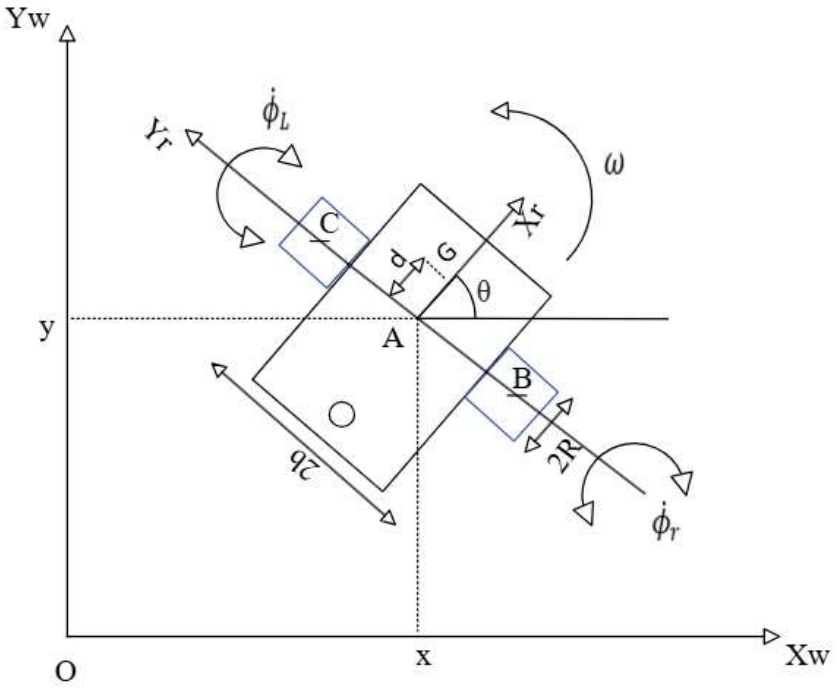
Các tín hiệu này được truyền đến mạch điều khiển động cơ Cytron MD30C.

- Cytron MD30C Motor Driver: Nhận tín hiệu điều khiển từ Teensy và cấp dòng điện điều khiển 2 động cơ DC thông qua 2 cặp đầu ra dạng cầu H đơn. Driver có khả năng xử lý dòng cao (tới 30A), phù hợp với yêu cầu mô-men của động cơ.

CHƯƠNG 3: MÔ HÌNH HÓA VÀ CÁC PHƯƠNG PHÁP ĐIỀU KHIỂN

3.1 Mô hình hóa hệ thống

3.1.1 Xây dựng phương trình động học xe tự hành



Hình 3.1: Minh họa mối quan hệ động học

Với:

- 2b là khoảng cách tâm bánh trái đến tâm bánh phải
- R là bán kính bánh xe
- A là điểm giữa của tâm bánh trái và bánh phải
- G là tâm của Robot
- d là khoảng cách tâm trục bánh xe đến tâm Robot
- θ là góc quay của robot
- X_w, Y_w là hệ tọa độ toàn cục
- X_r , Y_r là hệ tọa độ robot

Trạng thái của xe tự hành được đặc trưng bởi vị trí và hướng di chuyển trong không gian.

Cụ thể nó được mô tả [12] bằng:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad (3.1)$$

Trong đó:

- x, y xác định ví trí của tâm trục bánh xe trong hệ toạ độ toàn cục
- θ biểu diễn góc quay của xe so với trục x, cho biết hướng di chuyển hiện tại

Ngoài ra, vận tốc của xe trong hệ quy chiếu gắn với thân xe [12] là:

$$V = \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (3.2)$$

Với:

- ω là tốc độ góc quay của xe
- $V_x$là vận tốc theo hướng đi của xe
- $V_y$là vận tốc ngang

Do hệ toạ độ thân xe quay theo góc θ so với hệ toạ độ toàn cục [13], ta có:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = R_z^T(\theta) \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (3.3)$$

Vì xe không có vận tốc ngang, theo như Hirpo và Boru Diriba [13] ta rút gọn như sau:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ 0 \\ \omega \end{bmatrix} \quad (3.4)$$

Theo lí thuyết chuyển động song phẳng của vật rắn [14], ta có:

$$\overrightarrow{V_A} = \overrightarrow{V_C} + \vec{\omega} \times \vec{r}_{CA} \quad (3.5)$$

$$\overrightarrow{V_A} = \overrightarrow{V_B} + \vec{\omega} \times \vec{r}_{BA} \quad (3.6)$$

Với:

$\overrightarrow{V_B}$ *l*à vận tiếp tuyến tại tâm bánh xe phải

$\overrightarrow{V_C}$ *l*à vận tiếp tuyến tại tâm bánh xe phải

$\overrightarrow{V_A}$ là vận tốc tiếp tuyến tâm bánh xe

$\overrightarrow{r_{CA}}, \overrightarrow{r_{BA}}$ là véc-tơ vị trí từ điểm C đến điểm A và vector vị trí từ điểm B đến điểm A

Cộng (3.5) *và* (3.6) ta được:

$$2\overrightarrow{V_A} = \overrightarrow{V_C} + \overrightarrow{V_B} \rightarrow \overrightarrow{V_A} = \frac{r_R \times \dot{\phi}_r + r_L \times \dot{\phi}_L}{2} \quad (3.\ 7)$$

Trừ (3.5) cho (3.6) ta có:

$$(V_C - V_B)\overrightarrow{X_r} = (\ \vec{r}_{CA} - \ \vec{r}_{CB}\ ) \times \vec{\omega} \quad (3.\ 8)$$

Từ đây ta có:

$$\omega = \frac{r_R \times \dot{\phi}_r - r_L \times \dot{\phi}_L}{2b} \quad (3.\ 9)$$

Từ (3.7), (3.9) ta có thể viết lại là:

$$\begin{bmatrix} V \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r_R}{2} & \frac{r_L}{2} \\ \frac{r_R}{2b} & \frac{-r_L}{2b} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_L \end{bmatrix} \quad (3.\ 10)$$

Từ các phương trình (3.4), (3.10) ta có phương trình động học của xe khi tâm tại điểm A là:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r_R \cos(\theta)}{2} & \frac{r_L \cos(\theta)}{2} \\ \frac{r_R \sin(\theta)}{2} & \frac{r_L \sin(\theta)}{2} \\ \frac{r_R}{2b} & -\frac{r_L}{2b} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_L \end{bmatrix} \quad (3.\ 11)$$

Từ đây ta sẽ có 5 trường hợp:

- $\dot{\phi}_L = \dot{\phi}_r$ xe đi thẳng: $V_C = V_A = V_B = \dot{\varphi} \times \mathrm{r}$
- $\dot{\phi}_L > \dot{\phi}_r$ *thì rẽ phải*
- $\dot{\phi}_L < \dot{\phi}_r$ *thì rẽ trái*
- $\dot{\phi}_L < 0, \dot{\phi}_r < 0$ xe đi lùi $V_C = V_A = V_B = -\dot{\varphi} \times \mathrm{r}$
- $\dot{\phi}_L = -\dot{\phi}_r$ xe quay vòng: $V_A = 0, \omega = \frac{\dot{\varphi} \times \mathrm{r}}{2b}$

Xét tâm xe khi không nằm ở tâm trục bánh xe mà nằm ở điểm G, theo Shin [14], ta có:

$$\overrightarrow{V_G} = \overrightarrow{V_A} + \overrightarrow{\omega_A} \times \vec{r}_{AG} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix} \times \begin{bmatrix} dcos(\theta) \\ dsin(\theta) \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x} - \dot{\theta} dcos(\theta) \\ \dot{y} + \dot{\theta} dsin(\theta) \\ 0 \end{bmatrix} \quad (3.\ 12)$$

$$\overrightarrow{\omega_G} = [0 \quad 0 \quad \dot{\theta}]^T$$

Từ phương trình *(3.4), (3.11), (3.12)* ta có phương trình động học của robot tại điểm G cách tâm trục bánh xe 1 khoảng d là:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{R(b\cos(\theta)-d\sin(\theta))}{2b} & \frac{R(b\cos(\theta)+d\sin(\theta))}{2b}\\ \frac{R(b\sin(\theta)+d\cos(\theta))}{2b} & \frac{R(b\sin(\theta)-d\cos(\theta))}{2b}\\ \frac{R}{2b} & \frac{R}{2b}\end{bmatrix}\begin{bmatrix}\dot{\phi}_r\\ \dot{\phi}_L\end{bmatrix} \quad (3.13)$$

3.1.2 Xây dựng phương trình động lực học của xe tự hành

Cấu hình robot được biểu diễn thông qua 5 toạ độ [15], tổng quát như sau:

$$q = [x\ y\ \theta\ \varphi_R\ \varphi_L]^T \quad (3.14)$$

Ta có vận tốc tiếp tuyến tại tâm bánh xe phải là:

$$V_B = \begin{bmatrix}\dot{x} + b\dot{\theta}\cos(\theta)\\ \dot{y} + b\dot{\theta}\sin(\theta)\\ 0\end{bmatrix} \quad (3.15)$$

Vận tốc thẳng tiếp tuyến tại tâm bánh xe trái là:

$$V_C = \begin{bmatrix}\dot{x} - b\dot{\theta}\cos(\theta)\\ \dot{y} - b\dot{\theta}\sin(\theta)\\ 0\end{bmatrix} \quad (3.16)$$

Trong hệ cơ học Non-holonomic, khi chúng ta có các tham số ràng buộc lẫn nhau ta có thể viết chúng dưới dạng ràng buộc Pfaffian như sau:

$$C(q)\dot{q} = 0 \quad (3.17)$$

Trong trường hợp này, ta không thể tìm được tập hợp tọa độ tổng quát độc lập mà phải sử dụng nhân tử Lagrange (Lagrangian Multipliers) để đưa các ràng buộc này vào phương trình Euler-Lagrange, thu được:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] - \frac{\partial L}{\partial q} - C(q)^T\lambda = Q \quad (3.18)$$

Với:

- L là hàm Lagrange của hệ
- q là vector toạ độ tổng quát
- C(q) là ma trận ràng buộc Pfaffian
- λ là nhân tử Lagrange
- Q là lực tổng quát tác động của hệ

Ta có [15]:

$$\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\theta} \quad \dot{\varphi}_R \quad \dot{\varphi}_L] \tag{3. 19}$$

Với:

- $\dot{\varphi}_R$ là vận tốc góc bánh xe bên phải
- $\dot{\varphi}_L$là vận tốc góc bánh xe bên trái

Từ phương trình động học trước ta có:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} - \begin{bmatrix} \frac{R(bcos(\theta)-dsin(\theta))}{2b} & \frac{R(bcos(\theta)+dsin(\theta))}{2b} \\ \frac{R(bsin(\theta)+dcos(\theta))}{2b} & \frac{R(bsin(\theta)-dcos(\theta))}{2b} \\ \frac{R}{2b} & \frac{R}{2b} \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix} = 0 \tag{3. 20}$$

Từ đây ta suy ra được phương trình C(q) theo dữ liệu $C(q)\dot{q} = 0$ [15], như sau:

$$C(q) = \begin{bmatrix} \sin(\theta) & \cos(\theta) & -d & 0 & 0 \\ \cos(\theta) & \sin(\theta) & b & -R & 0 \\ \cos(\theta) & \sin(\theta) & -b & 0 & -R \end{bmatrix} \tag{3. 21}$$

Giờ ta xét động năng của hệ thống ta có:

Động năng của hệ sẽ gồm 3 thành phần: Khung xe và 2 bánh xe

Xét động năng của khung xe [16]:

$$K_c = \frac{m_G}{2} \left\| \overrightarrow{V_G} \right\|^2 + \overrightarrow{\omega_G}^T I_G \overrightarrow{\omega_G}$$

$$= \frac{m_G}{2}(\dot{x}^2 + \dot{y}^2 + d^2\dot{\theta}^2 + 2d\dot{\theta}(\dot{y}\cos(\theta) - \dot{x}sin\theta)) + \frac{I_G}{2}\dot{\theta}^2 \tag{3. 22}$$

Chọn khung xe hình hộp chữ nhật ta có:

$$I_G = \frac{1}{12}m(a^2 + b^2) \tag{3. 23}$$

Đối với động năng của bánh xe, gọi độ dày bánh xe là t, vận tốc và ma trận quán tính của bánh xe bên phải có thể biểu diễn như sau:

Ta có vận tốc góc của bánh xe không chỉ bao gồm $\dot{\varphi}_R$ mà còn chịu ảnh hưởng của chuyển động quay robot [16], từ đây ta có:

$$\overrightarrow{\Omega_\omega} = \dot{\phi}_R \begin{bmatrix} \sin(\theta) \\ cos(\theta) \\ 0 \end{bmatrix} + \dot{\theta} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \dot{\phi}_R sin(\theta) \\ \dot{\phi}_R \cos(\theta) \\ \dot{\theta} \end{bmatrix} \tag{3. 24}$$

$$I_\omega^G = \begin{bmatrix} I_{xx}^G & 0 & 0 \\ 0 & I_{yy}^G & 0 \\ 0 & 0 & I_{zz}^G \end{bmatrix} = \begin{bmatrix} \frac{m_\omega(3r^2+t^2)}{12} & 0 & 0 \\ 0 & \frac{m_\omega r^2}{2} & 0 \\ 0 & 0 & \frac{m_\omega(3r^2+t^2)}{12} \end{bmatrix} \quad (3.25)$$

Chuyển tensor quán tính từ hệ quy chiếu gắn với vật sang hệ quy chiếu toàn cục ta có:

$$I_\omega = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{xx}^G & 0 & 0 \\ 0 & I_{yy}^G & 0 \\ 0 & 0 & I_{zz}^G \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.26)$$

Từ đây ta có phương trình động năng bánh phải [17]:

$$K_R = \frac{m_\omega}{2} \left\| \overrightarrow{V_B} \right\|^2 + \overrightarrow{\Omega_\omega}^T I_\omega \overrightarrow{\Omega_\omega}$$

$$= \frac{m_\omega}{2} (\dot{x}^2 + \dot{y}^2 + b^2\dot{\theta}^2 + 2b\dot{\theta}(\dot{y}\cos\theta + \dot{x}sin\theta)) + \frac{I_{zz}^G}{2}\dot{\theta}^2 + \frac{I_{yy}^G}{2}\dot{\phi}_R{}^2 \quad (3.27)$$

Tương tự cho động năng bánh trái [17], ta có:

$$K_L = \frac{m_\omega}{2} \left\| \overrightarrow{V_C} \right\|^2 + \overrightarrow{\Omega_{\omega 2}}^T I_\omega \overrightarrow{\Omega_{\omega 2}}$$

$$= \frac{m_\omega}{2} (\dot{x}^2 + \dot{y}^2 + b^2\dot{\theta}^2 - 2b\dot{\theta}(\dot{y}\cos\theta + \dot{x}sin\theta)) + \frac{I_{zz}^G}{2}\dot{\theta}^2 + \frac{I_{yy}^G}{2}\dot{\phi}_L{}^2 \quad (3.28)$$

Từ 3 phương trình động năng trên ta rút ra được phương trình:

$$K_{total} = \frac{(2m_\omega + m_G)(\dot{x}^2 + \dot{y}^2)}{2} + \frac{(I_G + m_G d^2 + 2m_\omega b^2 + 2I_{zz}^G)\dot{\theta}^2}{2}$$

$$+ m_G d\dot{\theta}(\dot{y}\cos\theta - \dot{x}sin\theta) + \frac{I_{yy}^G}{2}(\dot{\phi}_L{}^2 + \dot{\phi}_R{}^2) \quad (3.29)$$

Gọi:

- $m_T = m_G + 2m_\omega$: là tổng khối lượng của xe khi chưa tải
- $I_T = (I_G + m_G d^2 + 2m_\omega b^2 + 2I_{zz}^G$: là tổng momen quán tính quay

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{q}}\right] - \frac{\partial L}{\partial q} - C(q)^T\lambda = Q = [0\ 0\ 0\ \tau_1\tau_2] \quad (3.30)$$

với τ_1, τ_2là lực tác động lên bánh xe, theo Ahmad Abu Hatab [17]:

$$\frac{d}{dt}\begin{bmatrix} m_T\dot{x} - m_G d\dot{\theta}(sin\theta) \\ m_T\dot{y} + m_G d\dot{\theta}(sin\theta) \\ I_T\dot{\theta} + m_G d\dot{\theta}(\dot{y}\cos\theta - \dot{x}sin\theta) \\ \frac{I_{yy}^G}{2}\dot{\phi}_R \\ \frac{I_{yy}^G}{2}\dot{\phi}_R \end{bmatrix} + m_G d\dot{\theta}\begin{bmatrix} 0 \\ 0 \\ \dot{y}\cos\theta + \dot{x}sin\theta \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -sin\theta & cos\theta & cos\theta \\ cos\theta & sin\theta & sin\theta \\ -d & L & -L \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix}\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \tau_1 \\ \tau_2 \end{bmatrix} \quad (3.31)$$

Ta có thể viết lại thành:

$$M(q)\ddot{\vec{q}} + B\left(\vec{q}, \dot{\vec{q}}\right) - C^T(q)\vec{\lambda} = \vec{Q} \quad (3.32)$$

Từ đây ta có:

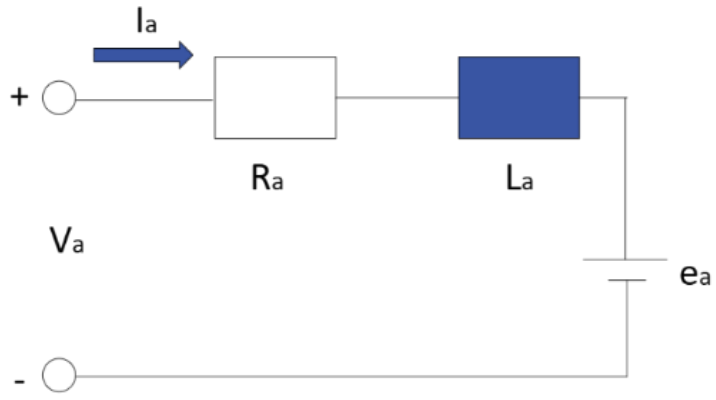
$$M(\vec{q}) = \begin{bmatrix} m_T & 0 & -m_G d(sin\theta) & 0 & 0 \\ 0 & m_T & m_G d(cos\theta) & 0 & 0 \\ -m_G d(sin\theta) & m_G d(cos\theta) & I_T & 0 & 0 \\ 0 & 0 & 0 & I_{yy}^G & 0 \\ 0 & 0 & 0 & 0 & I_{yy}^G \end{bmatrix} \quad (3.33)$$

$$B\left(\vec{q}, \dot{\vec{q}}\right) = m_G d\dot{\theta}^2\begin{bmatrix} \cos\theta \\ sin\theta \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.34)$$

$$C(q) = \begin{bmatrix} \sin(\theta) & \cos(\theta) & -d & 0 & 0 \\ \cos(\theta) & \sin(\theta) & b & -R & 0 \\ \cos(\theta) & \sin(\theta) & -b & 0 & -R \end{bmatrix} \quad (3.35)$$

3.1.3 Hàm truyền động cơ DC

Ta có Sơ đồ thay thế của mạch phần ứng động cơ điện một chiều kích từ độc lập



Hình 3.2: Sơ đồ thay thế của mạch phần ứng động cơ điện 1 chiều kích từ độc lập

Phương trình điện áp của mạch phần ứng:

$$V_a(s) = E_a(s) + R_a I_a(s) + L_a s\, I_a(s) \quad (3.36)$$

Với:

- R_a là điện trở phần ứng
- L_a là điện cảm cuộn dây
- E_a là suất điện động cảm ứng
- I_a là dòng điện đặt trong từ trường

Suất điện động cảm ứng của động cơ:

$$E_a(s) = K_e \phi \omega(s) \quad (3.37)$$

Với:

- K_e là hệ số cấu tạo, phụ thuộc vào các hằng số như là kích thước của mạch,chiều dài thanh dẫn
- ϕ là từ thông
- ω là tốc độ quay

Phương trình momen điện từ:

$$M_{dt}(s) = K_M \phi I_a(s) \quad (3.38)$$

- K_M là hệ số cấu tạo
- I_a là dòng phần ứng

Phương trình mô tả quan hệ điện-cơ:

$$M_{dt}(t) - M_c(t) = J\frac{d\omega(t)}{dt} \Rightarrow M_{dt}(s) - M_c(s) = Js\omega(s) \quad (3.39)$$

Với:

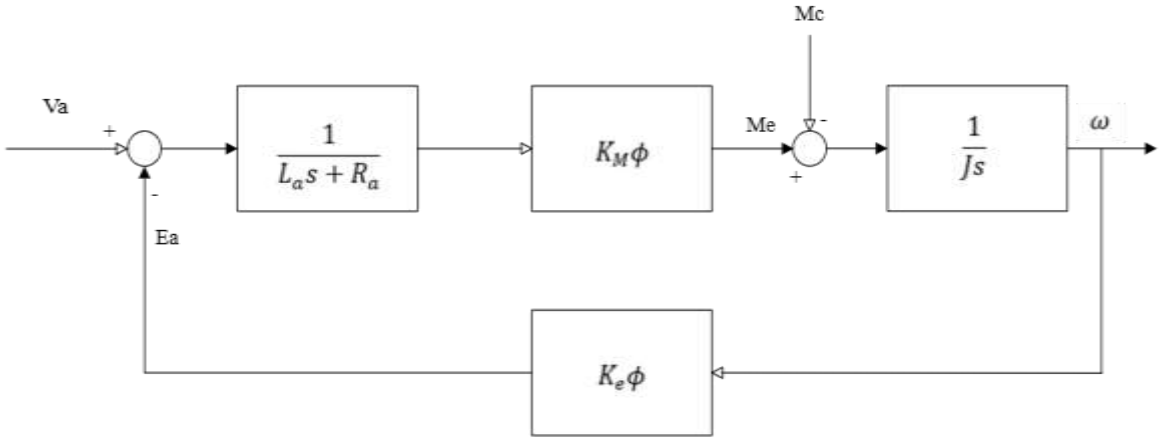
- $M_c(t)$ là momen cản

Kết hợp các phương trình trên rút ra:

$$I_a(s) = \frac{V_a(s) - K_e \phi \omega(s)}{L_a s + R_a} \quad (3.40)$$

$$\omega(s) = \frac{K_M \phi I_a(s) - M_c(s)}{Js} \quad (3.41)$$

Từ các phương trình trên ta vẽ sơ đồ khối mô tả toán học của động cơ 1 chiều:



Hình 3.3: Sơ đồ khối mô tả toán học của động cơ

Ta có hàm truyền đạt của động cơ 1 chiều:

Giả sử $M_c(s) = 0$ (trường hợp không tải) ta có:

$$\omega(s)(Js(L_a s + R_a) + K_M\phi K_e\phi) = K_M\phi V_a(s) \quad (3.42)$$

$$\frac{\omega(s)}{V(s)} = \frac{K_M\phi}{Js(L_a s + R_a) + K_M\phi K_e\phi} \quad (3.43)$$

Từ các giá trị trong datasheet ta tổng kết lại:

Bảng 3.1: Bảng dữ liệu động cơ

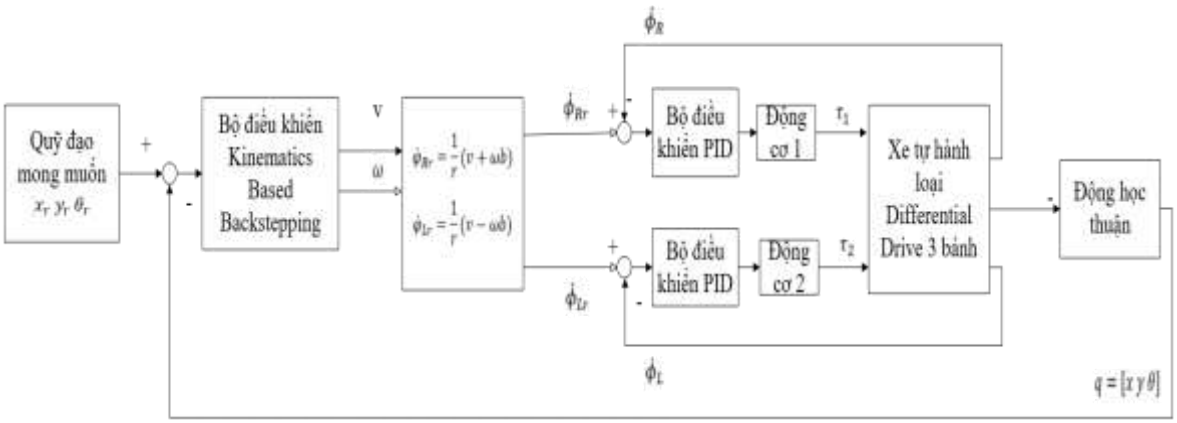
Tham số	Giá trị
Điện áp	12V
Điện trở phần ứng	36.5Ω
Hằng số momen K_m	0.01909 Nm/A
Hằng số phản điện động K_e	0.0277 V.s/rad
Độ tự cảm phần ứng L_a	2.2 mH
Momen quán tính rotor J_m	6.8×10^{-8} kg.m^2

Từ đây ta có hàm truyền đạt:

$$\frac{\omega(s)}{\mathrm{V}(s)} = \frac{0.01909}{1.496\times10^{-10}s^2+2.482\times10^{-6}\,s+5.28953\times10^{-4}} \qquad (3.44)$$

3.2 Phương pháp điều khiển

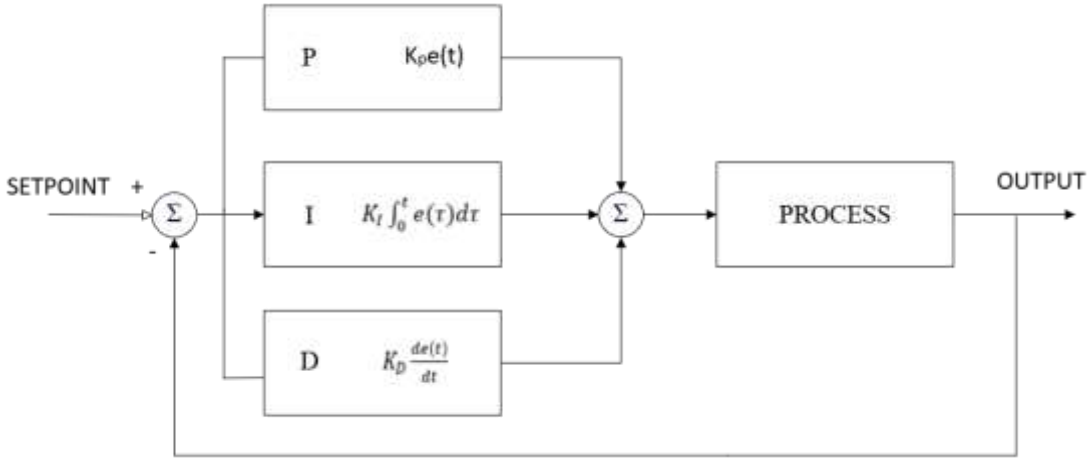
Sơ đồ tổng quan phương pháp điều khiển bám vị trí của hệ thống:



Hình 3.4: Sơ đồ khối hệ thống điều khiển robot vi sai 3 bánh

3.2.1 Bộ điều khiển PID của động cơ DC

PID là bộ điều khiển hồi tiếp vòng kín được sử dụng nhiều nhất trong công nghiệp, là sự kết hợp của 3 bộ điều khiển: tỉ lệ, tích phân, và vi phân. Có khả năng triệt tiêu sai số xác lập, tăng tốc độ đáp ứng, giảm độ vọt lố nếu thông số của bộ điều khiển được lựa chọn thích hợp



Hình 3.5: Bộ điều khiển PID

PID là một cơ chế phản hồi vòng điều khiển và được sử dụng nhiều nhất trong các hệ thống điều khiển có tín hiệu phản hồi vòng kín, giá trị sai lệch sẽ được tính toán thông

qua bộ điều khiển là kết quả của hiệu số giữa giá trị mong muốn và giá trị phản hồi. Bộ điều khiển sẽ làm giảm sai số bằng cách điều chỉnh các giá trị điều khiển đầu ra. Các thông số PID sử dụng trong tính toán phải điều chỉnh theo tính chất của hệ thống điều khiển, phải phụ thuộc vào đặc thù tính chất của hệ thống điều khiển để kết quả đạt như mong đợi.

3.2.1.1 Ý nghĩa tham số PID

a. Khâu tỉ lệ:

Hàm truyền K(s)= K_P

Đặc tính thời gian: Y(s)= K_PG(s)E(s)

- $\boldsymbol{K_P}$ càng lớn thì tốc độ đáp ứng càng nhanh
- $\boldsymbol{K_P}$ càng lớn thì sai số xác lập càng nhỏ (nhưng không thể triệt tiêu)
- $\boldsymbol{K_P}$ càng lớn thì các cực của hệ thống có xu hướng di chuyển ra xa trục thức → Hệ thống càng dao động và độ vọt lố càng cao
- Nếu $\boldsymbol{K_P}$ tăng quá giá trị giới hạn thì hệ số không tắt dần → Hệ thống mất ổn định

b. Khâu tích phân

Tín hiệu ngõ ra được xác định bởi sai số

- $\boldsymbol{K_I}$ càng lớn thì đáp ứng quá độ càng chậm
- $\boldsymbol{K_I}$ càng lớn thì sai số xác lập càng nhỏ. Đặc biệt hệ số khuếch đại của khâu tích phân bằng vô cùng khi tần số = 0 → triệt tiêu sai số xác lập với hàm nâng
- $\boldsymbol{K_I}$ càng lớn thì độ vọt lố càng cao

c. Khâu vi phân

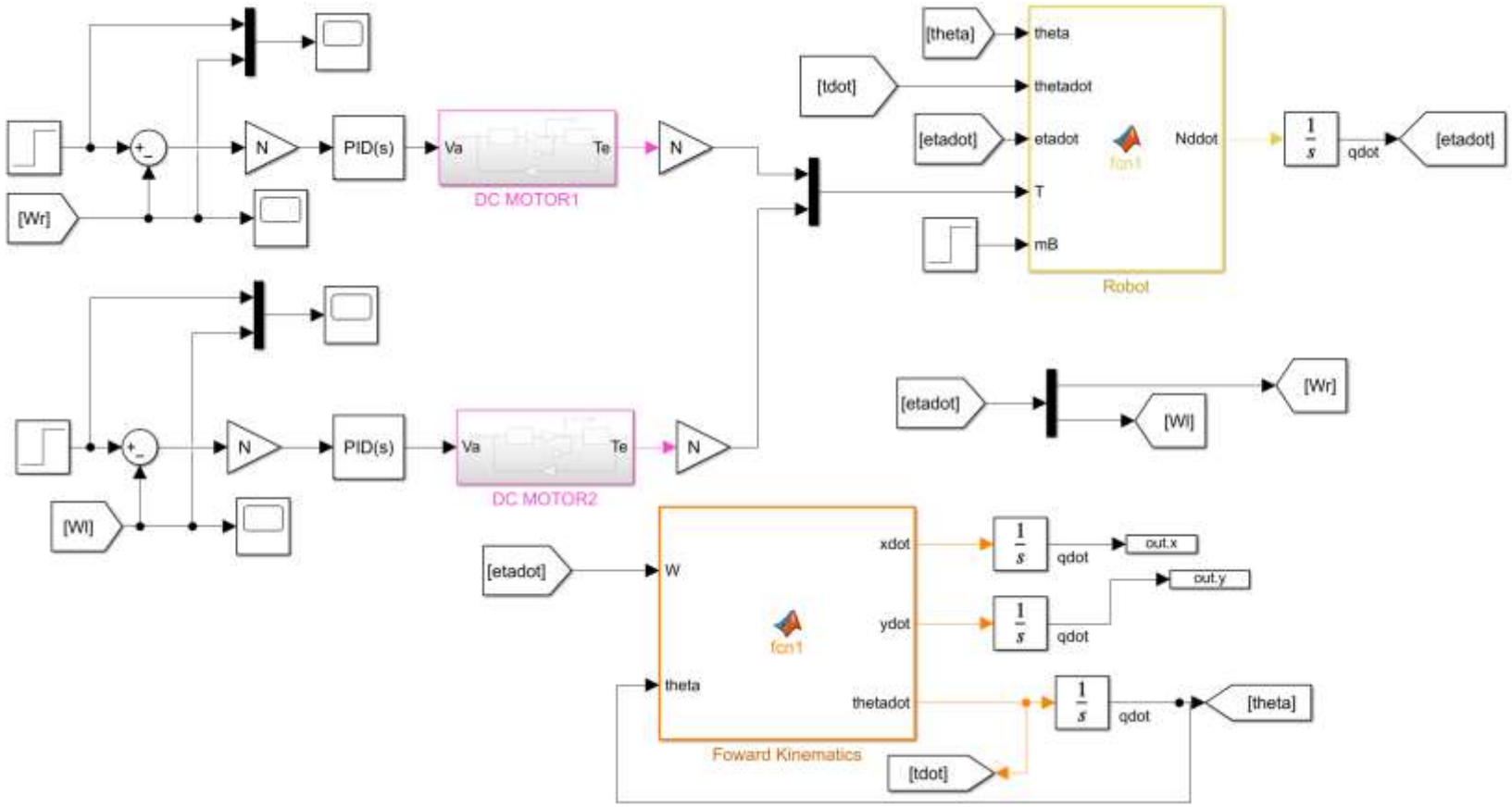
- $\boldsymbol{K_D}$ càng lớn thì đáp ứng quá độ càng nhanh
- $\boldsymbol{K_D}$ càng lớn thì độ vọt lố càng nhỏ
- Hệ số khuếch đại tại tần số cao là vô cùng lớn nêu khâu hiệu chỉnh D rất nhạy với nhiễu tần số cao.

3.2.1.2 Phương pháp điều chỉnh PID

PID có nhiều phương pháp để tìm thông số như là phương pháp Ziegler-Nichols, phương pháp điều chỉnh theo tối ưu hóa, phương pháp triển khai theo dạng mô hình xử lý sau đó chọn P, I, D dựa trên các thông số của mô hình động học….

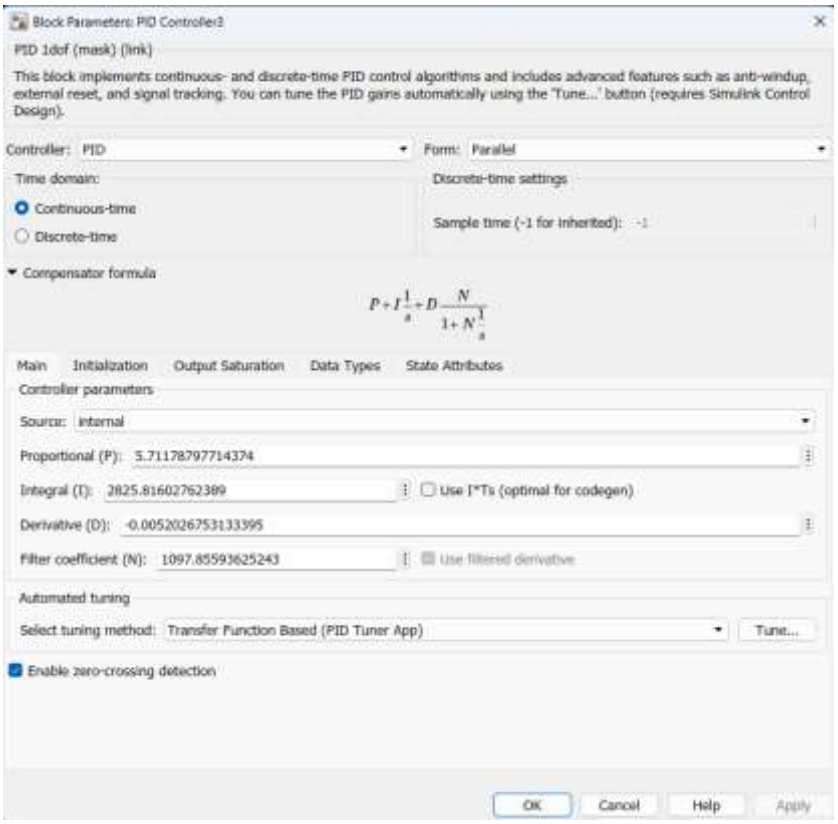
Tuy nhiên trong đề tài này nhóm quyết định sử dụng phương pháp tunning tham số PID bằng khối PID controller trong matlab simulink.

Sau đây là sơ đồ mô phỏng:



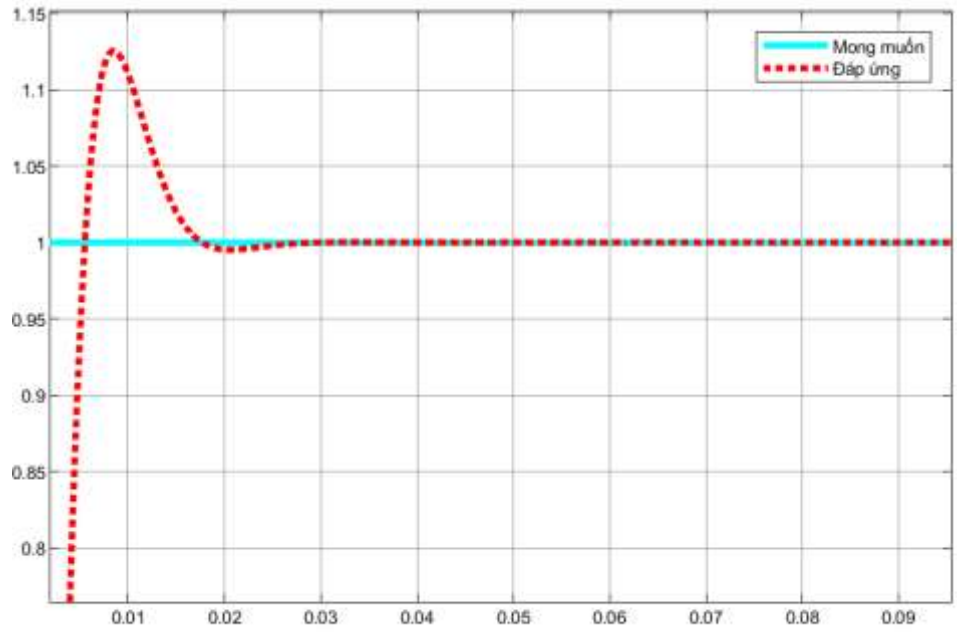
Hình 3.6: Sơ đồ mô phỏng PID

Giá trị PID khi tunning:

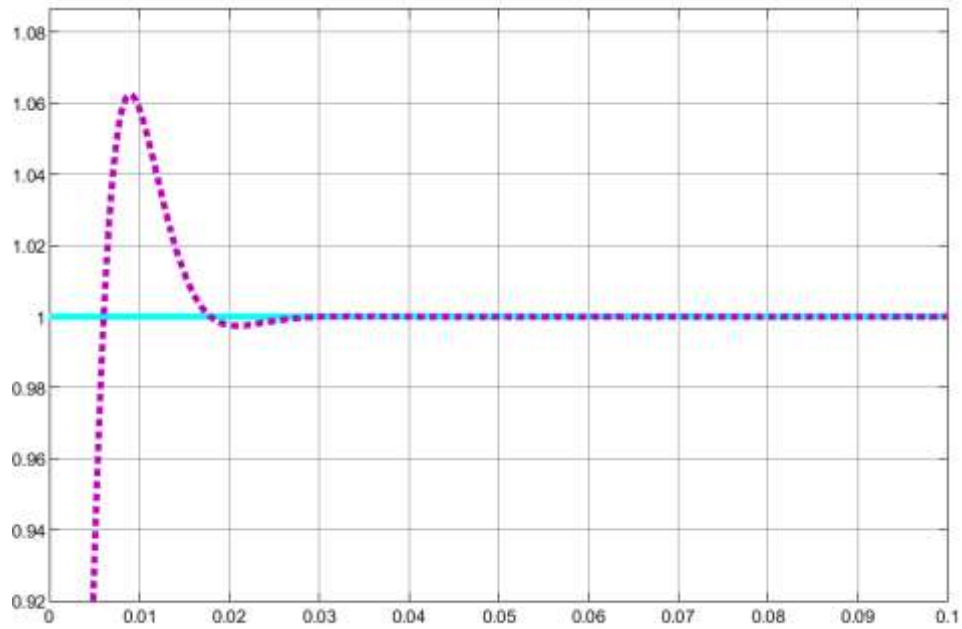


Hình 3.7: Thông số PID

Kết quả mô phỏng:



Hình 3.8: Kết quả mô phỏng của động cơ phải



Hình 3.9: Kết quả mô phỏng động cơ trái

Kết luận:

Ta nhận thấy hệ thống có phản hồi nhanh và đạt được trạng thái ổn định trong thời gian ngắn. Tuy nhiên, độ vọt lố (POT) vẫn còn tương đối cao, đạt khoảng 11.55%, tức giá trị cực đại của đáp ứng đạt xấp xỉ 1.155 so với giá trị mong muốn là 1. Mặc dù thông số này vẫn nằm trong ngưỡng chấp nhận được (thường < 20%), nhưng cần cân nhắc điều chỉnh *để cải thiện hơn nữa* độ chính xác và hiệu suất tổng thể của hệ thống.

3.2.2 Bộ điều khiển bám vị trí Backstepping Controller

3.2.2.1 Lý thuyết

Trong bộ điều khiển này, sẽ sử dụng hai tư thế [18] (vị trí và góc: tư thế tham chiếu)

$$q_r = [x_r \; y_r \; \theta_r] \tag{3. 45}$$

và tư thế hiện tại của robot $q = [x \; y \; \theta]$ để có thể điều khiển

Tư thế sai số trong hệ tọa độ gắn với robot [18], sẽ được định nghĩa như sau:

$$e_p = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{3.46}$$

– Với $(x_r \; y_r \; \theta_r)$: tọa độ mục tiêu trong hệ toàn cục.

– Với $(x\ y\ \theta)$: tọa độ hiện tại của robot trong hệ tọa độ toàn cục

Bài toán điều khiển ở đây nhằm xác định luật điều khiển sao cho phép ước lượng các vận tốc sao cho hệ thống đạt được tính ổn định tiệm cận, ta lựa chọn luật điều khiển dựa trên mô hình động học được đề xuất [18], như sau:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_r \cos(e_\theta) + K_x e_x \\ \omega_r + v_r\left(K_y e_y + K_\theta \sin(e_\theta)\right) \end{bmatrix} \quad (3.47)$$

Ta có:

$$e_p = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} = \begin{bmatrix} (x_r - x)\cos(\theta) + (y_r - y)\sin(\theta) \\ (x_r - x)\sin(\theta) + (y_r - y)\cos(\theta) \\ \theta_r - \theta \end{bmatrix} \quad (3.48)$$

$$\dot{e}_x = (\dot{x}_r - \dot{x})\cos(\theta) + (\dot{y}_r - \dot{y})\sin(\theta) - (x_r - x)\dot{\theta}\sin(\theta) + (y_r - y)\dot{\theta}\cos(\theta) \quad (3.49)$$

$$\dot{e}_y = -(\dot{x}_r - \dot{x})\sin(\theta) + (\dot{y}_r - \dot{y})\cos(\theta) - (x_r - x)\dot{\theta}\cos(\theta) - (y_r - y)\dot{\theta}\sin(\theta)$$

$$= -e_x\omega - \dot{x_r}\sin(\theta_r - \theta) + \dot{y_r}\cos(\theta_r - \theta) = -\omega e_x + v_r \sin(e_\theta), \quad (3.50)$$

$$\dot{e_\theta} = \omega_r - \omega \quad (3.51)$$

$$\begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \dot{e_p} = \dot{f}(t, e_p) = \begin{bmatrix} \omega e_y - v + v_r \cos(e_\theta) \\ -\omega e_x + v_r \sin(e_\theta) \\ \omega_r - \omega \end{bmatrix} \quad (3.52)$$

Dựa vào phương trình trên, có thể nói rằng khi vận tốc tham chiếu $v_r > 0$, $e_p = 0$ là 1 điểm cần bằng ổn định.

Ta chọn hàm V ứng với hàm Lyapunov cho hệ thống là:

$$V = \frac{1}{2}\left(e_x^2 + e_y^2\right) + \frac{(1-\cos(e_\theta))}{K_y} \quad (3.53)$$

Ta có, rõ ràng $V \geq 0, e_p = 0 \rightarrow V = 0$ *và nếu* $e_p \neq 0 \rightarrow V > 0$, do đó V là 1 hàm xác định dương, xét đạo hàm V ta có:$\dot{V} = \dot{e_x}e_x + \dot{e_y}e_y + \frac{\dot{e_\theta}\sin(e_\theta)}{K_y}$ (3.54)

$$\dot{V} = \left[\left(\omega_r + v_r\left(K_y e_y + K_\theta \sin(e_\theta)\right)\right)e_y - K_x e_x\right]e_x + \left[-\left(\omega_r + v_r\left(K_y e_y + K_\theta \sin(e_\theta)\right)\right)e_x + v_r \sin(e_\theta)\right]e_y + \frac{[-v_r(K_y e_y + K_\theta \sin(e_\theta))]\sin(e_\theta)}{K_y} = -K_x {e_x}^2 - \frac{-v_r K_\theta \sin(e_\theta)^2}{K_y} \leq 0, \quad (3.55)$$

Từ đây ta thấy đạo hàm của hàm Lyapunov V là một hàm xác định âm, điều này có nghĩa là hệ thống ạt được tính ổn định tiệm ận đều xung quanh $e_p = 0$ với điều kiện $v_r, \omega_r, K_x, K_y, K_\theta$ bị chặn và v_r, ω_r là các hàm liên tục.

Kết luận:

Luật điều khiển mà nhóm sử dụng để điều khiển xe bám theo đường đi mong muốn sẽ là:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_r \cos(e_\theta) + K_x e_x \\ \omega_r + v_r\left(K_y e_y + K_\theta \sin(e_\theta)\right) \end{bmatrix} \quad (3.\ 56)$$

Với K_x, K_y, K_θ là các hằng số dương. Các hằng số này là hệ số khuếch đại của bộ điều khiển Backstepping dựa trên mô hình động học.

Và tham số mà nhóm sử dụng để điều khiển sẽ là: $K_x = 4, K_y = 12, K_\theta = 9$

3.2.2.2 Mô phỏng Matlab

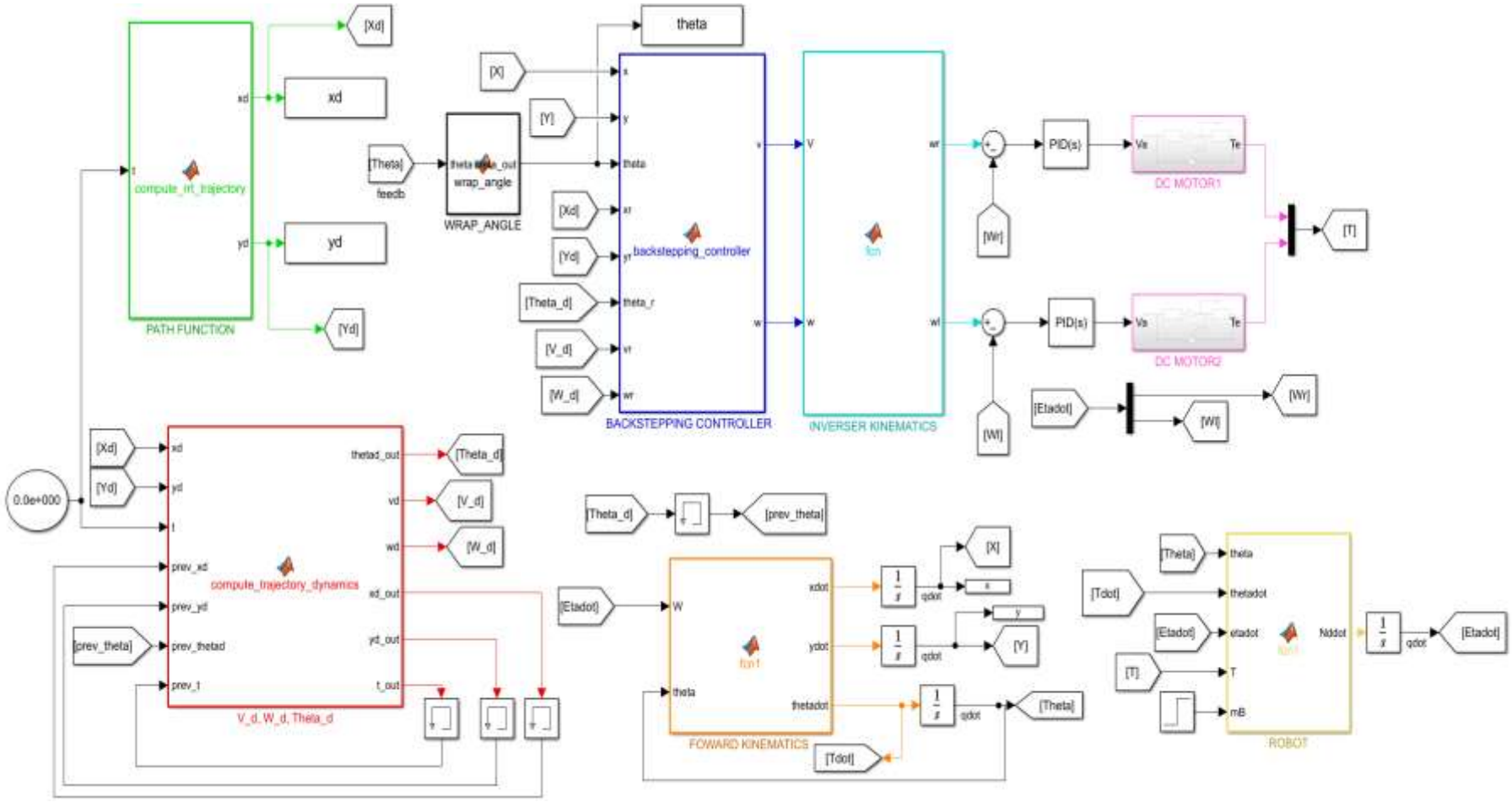
Đầu vào của hệ thống: 1 quỹ đạo bao gồm tập hợp các điểm (x_d, y_d, θ_d) mà robot muốn bám theo.

Đầu ra của hệ thống: 1 tập hợp các điểm x, y, θ thực tế mà robot di chuyển được.

Trong hệ thống này sẽ gồm 2 bộ điều khiển:

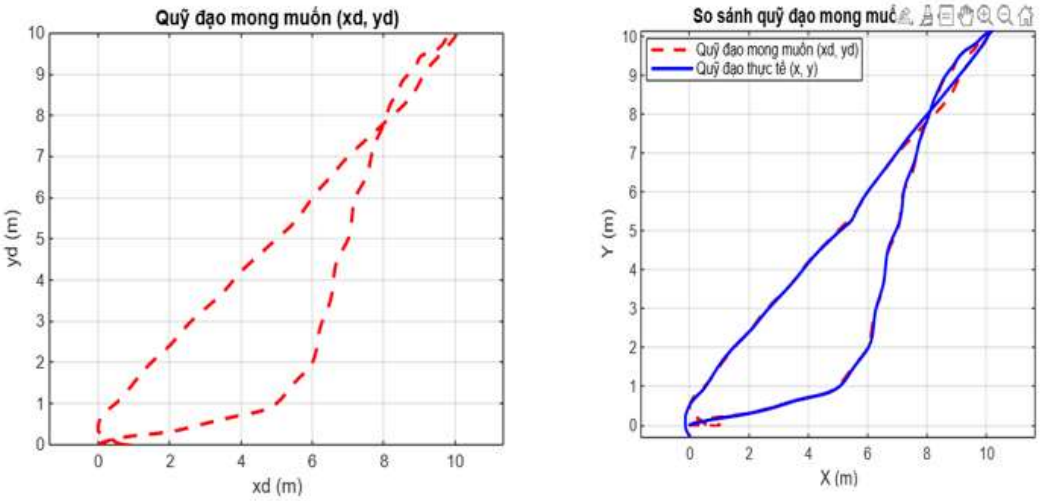
- Bộ điều khiển bậc cao (Backstepping Controller): Bộ điều khiển này nhận đầu vào là trạng thái thực tế của robot (x, y, θ) cùng với thông tin quỹ đạo tham chiếu, gồm vị trí (x_r, y_r) từ khối Path Function và vận tốc tham chiếu (v_d, ω_d) từ khối compute_trajectory, bộ điều khiển so sánh trạng thái thực tế với các giá trị tham chiếu này để tính toán bộ điều khiển vận tốc (v, ω) giúp robot bám sát quỹ đạo mong muốn, giảm sai số vị trí và góc. Sau đó, 2 giá trị (v, ω) này được chuyển đổi thành vận tốc mục tiêu từng bánh xe trái/phải, rồi đưa vào bộ điều khiển động cơ DC.
- Bộ điều khiển PID đảm nhiệm việc điều khiển động cơ, đảm bảo hai bánh xe đạt đúng vận tốc đã tính toán từ bộ điều khiển cấp cao, sau đó dung động học thuận để cập nhật vị trí hiện tại của robot đưa lên lại bộ điều khiển bậc cao để so sánh.

Mô phỏng matlab hệ thống:



Hình 3.10:Mô phỏng Matlab hệ thống

Kết quả mô phỏng:

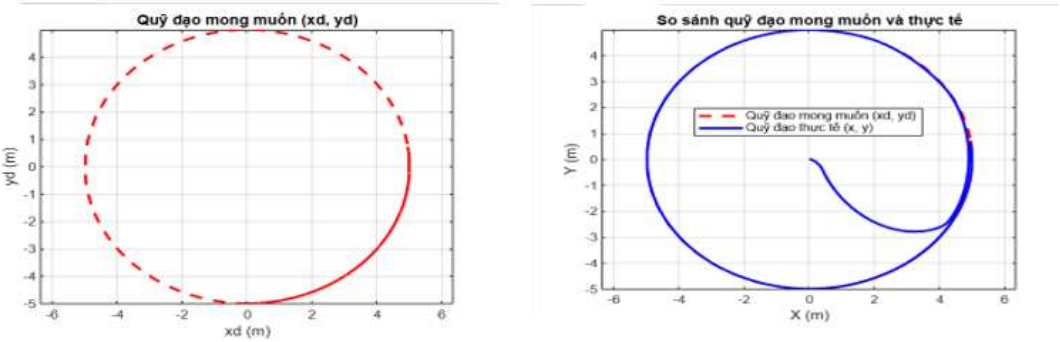


Hình 3.11: Quỹ đạo bất kì và Đáp ứng của hệ thống

Nhận xét:

Ta thấy bộ điều khiển điều khiển robot bám quỹ đạo khá tốt, tuy nhiên ở các đoạn cong hẹp thì thuật toán điều khiển chưa thực sự hiệu quả. Robot chưa bám sát được các đường cong mà có xu hướng cắt góc, gây sai lệch tại các khúc cua, tuy nhiên là không đáng kể.

Mô phỏng với quỹ đạo hình tròn:



Hình 3.12: Quỹ đạo mong muốn và đáp ứng của hệ thống

Nhận xét:

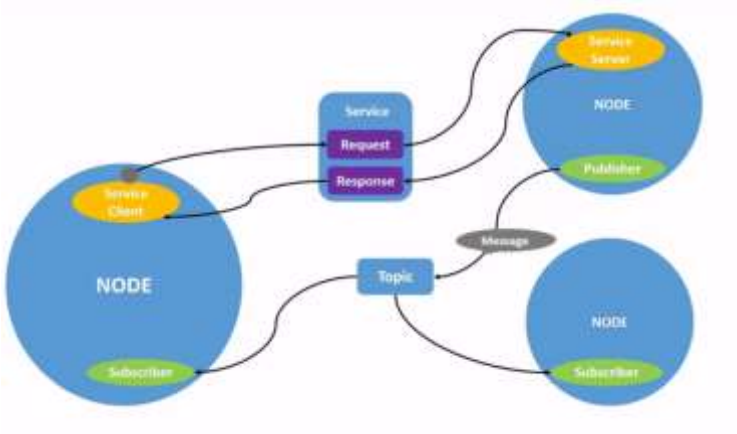
Sau một khoảng thời gian ban đầu, dù chạy từ tâm đường tròn nhưng sau 1 khoảng thời gian quỹ đạo thực tế (x, y) đã bám gần sát quỹ đạo mong muốn (xd, yd) điều này chứng tỏ bộ điều khiển có khả năng điều chỉnh sai lệch tốt và robot bám đường đi khá ổn định.

Chương 4: XÂY DỰNG CHỨC NĂNG ĐIỀU HƯỚNG VÀ NÉ VẬT CẢN CHO XE TỰ HÀNH

4.1 Nền tảng tích hợp thuật toán trong hệ thống robot tự hành

Trong bối cảnh các hệ thống robot tự hành ngày càng trở nên phức tạp và đòi hỏi tính linh hoạt, hiệu năng cao, Robot Operating System 2 (ROS 2) là một nền tảng quan trọng tạo điều kiện thuận lợi cho sự phát triển về lĩnh vực này. Đây là một framework chịu trách nhiệm đồng bộ hóa ác module phần mềm của Robot, trừu tượng các chi tiết phần cứng đối với lập trình sinh viên, ngoài ra ROS2 còn cung cấp công cụ và trực quan hóa mô hình robot trong môi trường ảo như Rviz2, Gazebo, hơn thế nữa ROS2 có cộng đồng phát triển lớn mạnh với hàng ngàn gói, thư viện mở, giúp đẩy nhanh quá trình phát triển và tùy chỉnh hệ thống robot. Nhờ vậy nhóm có thể thuận tiện thực hiện các dự án trong giai đoạn triển khai, áp dụng và kiểm thử.

ROS 2 kế thừa cấu trúc phân chia theo node từ ROS 1, đồng thời cải tiến đáng kể với khả năng giao tiếp thời gian thực, độ trễ thấp, hỗ trợ đa luồng, hoạt động tốt trong môi trường phân tán và công nghiệp. Ros2 có các đặc tính thiết yếu của 1 hệ điều hành như khả năng thực hiện các task song song, giao tiếp, trao đổi dữ liệu giữa các task, quản lí dữ liệu, hơn thế nữa Ros còn được phát triển riêng biệt về cá công cụ dành cho việc thu nhập, xử lý hiển thị điều khiển.



Hình 4.1: Sơ đồ kiến trúc giao tiếp Ros2

Trong khuôn khổ đề tài này, nhóm đã lựa chọn ROS 2 làm nền tảng triển khai toàn bộ hệ thống robot tự hành, kết hợp với các thuật toán trí tuệ nhân tạo như Graph SLAM

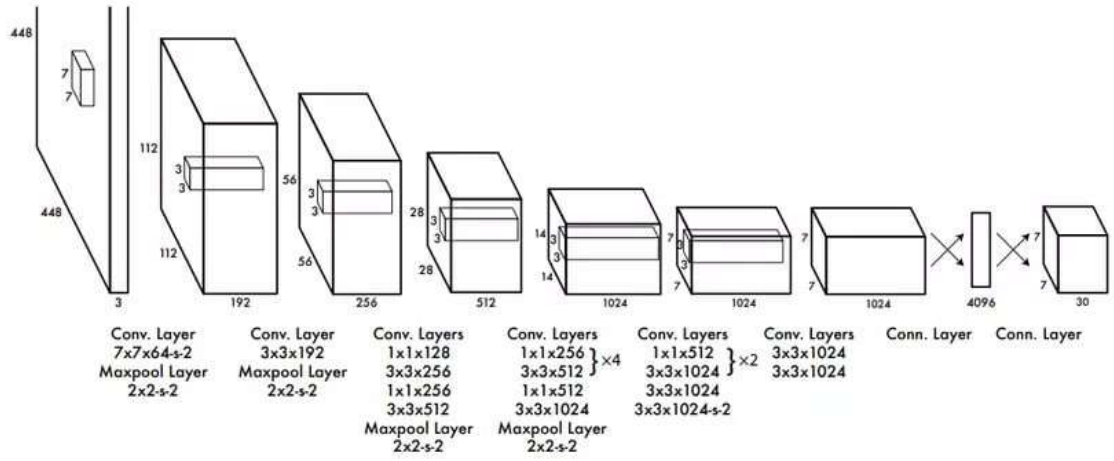
để vẽ bản đồ, AMCL để định vị, A* để hoạch định đường đi, Costmap để xử lý vật cản, và bộ điều khiển tối ưu để điều hướng robot. Tất cả các thành phần đều được phát triển, mô phỏng và kiểm thử trong môi trường ROS 2, sử dụng các công cụ trực quan như Rviz2 và Gazebo, đảm bảo tính thực tiễn, chính xác và khả năng ứng dụng cao.

4.2 Mô hình AI nhận dạng

4.2.1 Tổng quan về thuật toán YOLO

Trong môi trường vận hành thực tế của AMR tại các khu vực cách ly, phòng khám, an toàn khi robot di chuyển cùng con người luôn là ưu tiên hàng đầu. Việc phát hiện người bằng trí tuệ nhân tạo (AI) cần đảm bảo vừa nhanh, vừa chính xác, lại phù hợp với phần cứng giới hạn của robot. Do đó, nhóm đã lựa chọn thuật toán YOLO - một giải pháp phát hiện vật thể thời gian thực nổi bật hiện nay để tăng hiệu quả nhận diện và đảm bảo an toàn hệ thống.

YOLO (You Only Look Once) là mô hình mạng nơ-ron tích chập (CNN) xử lý toàn bộ ảnh chỉ trong một lần duy nhất, nhờ đó đạt tốc độ nhanh vượt trội so với các phương pháp truyền thống như RCNN. YOLO kết hợp giữa các lớp tích chập để trích xuất đặc trưng và các lớp kết nối đầy đủ để dự đoán vị trí, xác suất các đối tượng trên ảnh. Với ưu điểm vừa nhanh vừa chính xác, YOLO rất phù hợp cho các ứng dụng thời gian thực như xe và robot tự hành.



Hình 4.2: Kiến trúc YOLO

4.2.2 Lựa chọn mô hình nhận diện

Nhóm đã lựa chọn mô hình YOLOv8, làm giải pháp nhận diện con người vì các ưu điểm nổi bật sau:

- Độ chính xác cao: Mô hình YOLOv8 đạt mức độ chính xác (mAP - Mean Average Precision) cao trên nhiều bộ dữ liệu phổ biến như COCO, giúp phát hiện đối tượng tin cậy.
- Hiệu năng cao, thời gian đáp ứng thấp: YOLOv8 có khả năng xử lý nhanh, đáp ứng yêu cầu phát hiện đối tượng trong thời gian thực phù hợp với phần cứng Rasberry pi4 4GB mà nhóm sử dụng.
- Đa dạng mô hình: Cung cấp nhiều lựa chọn về kích thước mô hình (nano, small, medium), dễ dàng tùy chỉnh phù hợp với điều kiện và yêu cầu nhận diện phát hiện người của nhóm

YOLOv8 hỗ trợ 5 biến thể tương ứng với kích thước model từ nhỏ nhất đến lớn nhất theo size là n, s, m, l, x. Sau đây là chỉ số khi thực hiện detection trên tập dữ liệu COCO:

Bảng 4.1: Các biến thể của YOLOv8

Model	Kích thước ảnh đầu vào(pixels)	Map (50-95)	Speed CPU ONNX (ms)	Speed A100 TensorRT(ms)	Số lượng param	FLOPs(B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.2	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

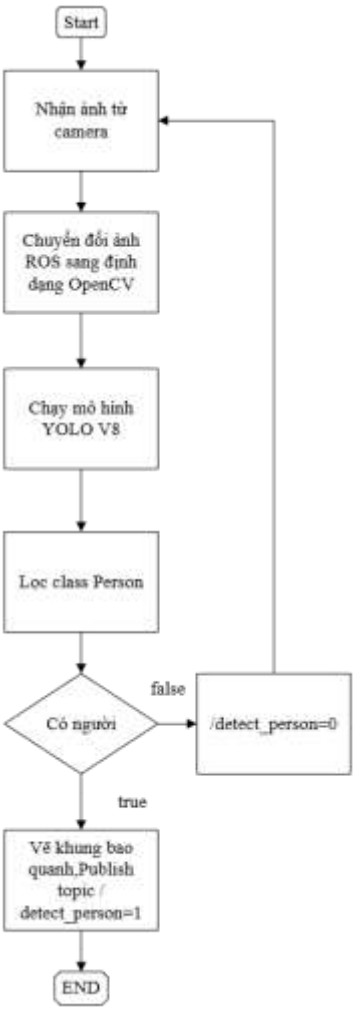
Trong đó:

- mAP 50-95: Giá trị mAP (Mean Average Precision) đo lường độ chính xác của mô hình. Giá trị này có thể được tính dựa trên các ngưỡng xác suất từ 50% đến 95%
- Speed CPU ONNX (ms): Thời gian xử lý trung bình của mô hình trên CPU khi sử dụng định dạng ONNX. Đây là thời gian tính bằng mili giây (ms) để mô hình dự đoán trên một ảnh.

- Speed A100 TensorRT (ms): Thời gian xử lý trung bình của mô hình trên GPU A100 khi sử dụng TensorRT. Đây là thời gian tính bằng mili giây (ms) để mô hình dự đoán trên một ảnh.
- FLOPs (B): Số lượng phép toán dấu chấm động (Floating Point Operations) tính toán bởi mô hình, đơn vị tính là tỷ (B). Đây là một đánh giá về độ phức tạp tính toán của mô hình.

Sau khi đánh giá các mô hình pretrain, để tiết kiệm tài nguyên và mà vẫn giữ được độ chính xác tương đối, đáp ứng nhanh với thời gian thực trong đồ án này sẽ sử dụng phiên bản nano cho nhiệm vụ person detection.

Sau khi đã chọn được phiên bản mô hình YOLO, để có thể áp dụng mô hình vào hệ thống, nhóm đề xuất lưu đồ thuật toán sau:



Hình 4.3: Lưu đồ thuật toán cách áp dụng mô hình YOLO trong hệ thống

Nguyên lý:

Hệ thống phát hiện người sử dụng mô hình YOLOv8 hoạt động theo chu trình xử lý tuần tự trên từng khung hình ảnh thu được từ camera. Đầu tiên, hệ thống nhận ảnh trực tiếp từ camera gắn trên thiết bị và chuyển đổi định dạng từ ROS (sensor_msgs/Image) sang ảnh OpenCV (cv::Mat) thông qua CvBridge, giúp thuận tiện cho xử lý bằng thư viện thị giác máy tính. Sau đó, ảnh được đưa trực tiếp vào mô hình YOLOv8 để suy luận. Trong quá trình này, thư viện Ultralytics sẽ tự động thực hiện các bước tiền xử lý cần thiết như thay đổi kích thước (resize) và chuẩn hóa giá trị pixel (normalize) theo đúng định dạng yêu cầu của mạng nơ-ron.

Kết quả đầu ra của mô hình là danh sách các hộp bao (bounding boxes) ứng với các vật thể phát hiện được trong ảnh, kèm theo nhãn lớp. Hệ thống tiếp tục lọc ra các đối tượng có nhãn là "person" (lớp người). Nếu không phát hiện người, hệ thống sẽ xuất tín hiệu 0 lên topic /detect_person. Ngược lại, nếu có người, hệ thống sẽ vẽ khung bao quanh người trong ảnh và xuất tín hiệu 1 lên topic đó để thông báo đến các node điều khiển khác.

4.2.3 Tiến hành mô phỏng, Kết quả mô phỏng

4.2.3.1 Tiến hành mô phỏng

Mô phỏng bao gồm hai thành phần chính: node camera đầu vào và node nhận diện người sử dụng YOLOv8.

- Đối với node camera đầu vào:

Đầu tiên, node camera được khởi tạo bằng cách sử dụng package v4l2_camera trên nền tảng ROS 2, cho phép đọc ảnh trực tiếp từ Raspberry Pi Camera Module V2.1 thông qua lệnh:

```
ros2 run v4l2_camera v4l2_camera_node --ros-args -p image_size:="[640, 480]"
```

- Đối với node nhận diện người:

Để thực hiện việc nhận diện người node cần phải có các chức năng sau:

a. Nhận ảnh từ topic camera

Node đăng ký làm subscriber của topic /image_raw, nơi mà ảnh từ camera được xuất ra dưới dạng ROS message kiểu sensor_msgs/Image. Đây là định dạng chuẩn trong ROS nhưng không thể thao tác trực tiếp bằng các thư viện xử lý ảnh thông dụng trong Python như OpenCV hoặc NumPy.

b. Chuyển đổi ảnh sang định dạng OpenCV

Để có thể xử lý ảnh bằng mô hình học sâu, ảnh ROS cần được chuyển đổi sang định dạng ảnh số kiểu mảng NumPy (ndarray) do ROS lưu ảnh dưới dạng message, không phải mảng số học, nếu đưa trực tiếp vào hoạt động với YOLOv8 sẽ không hoạt động vì OpenCV và YOLOv8 yêu cầu ảnh ở dạng NumPy để truy cập, xử lý pixel và đưa vào mô hình. Việc chuyển đổi này được thực hiện bằng công cụ CvBridge, giúp chuyển đổi dữ liệu từ sensor_msgs/Image sang định dạng ảnh BGR chuẩn của OpenCV:

frame = bridge.imgmsg_to_cv2(data, 'bgr8')

c. Chạy mô hình YOLOv8n

Sau khi có ảnh định dạng OpenCV, ảnh được đưa trực tiếp vào mô hình YOLOv8 bằng lệnh:

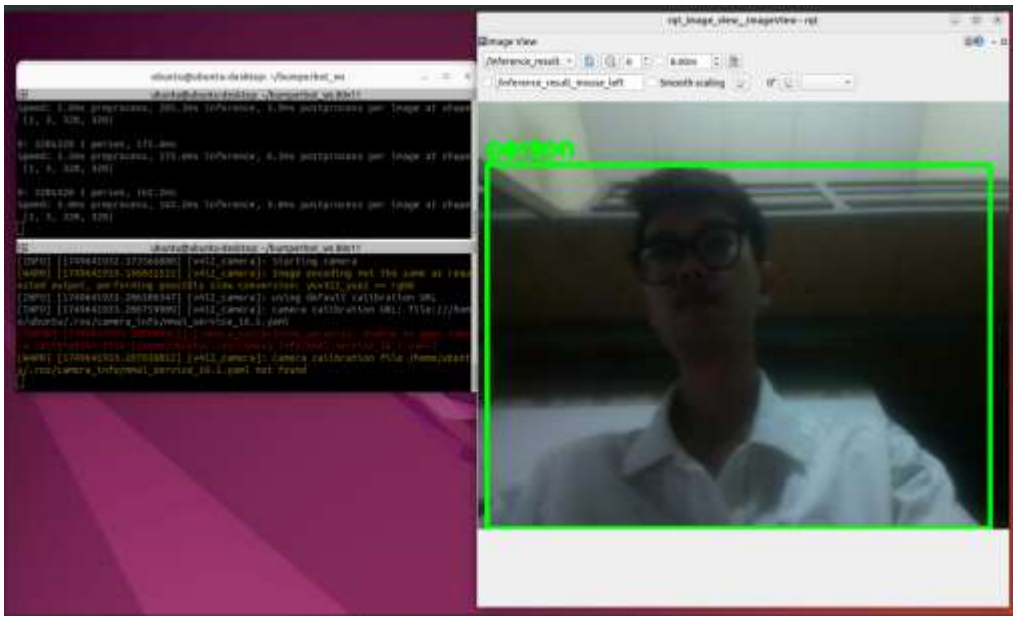
results = self.model(frame)

d. Xử lý kết quả đầu ra

Sau khi suy luận, mô hình YOLOv8 trả về danh sách các đối tượng được phát hiện trong ảnh, bao gồm tọa độ bounding box, class ID và độ tin cậy (confidence). Node sẽ duyệt qua danh sách này để kiểm tra có người xuất hiện hay không (class ID = 0). Nếu phát hiện người, hệ thống sẽ đánh dấu trạng thái và vẽ khung bao quanh đối tượng trong ảnh và nó được thực hiện thông qua câu lệnh sau:

```
if cls_id == 0:
person_found = True
coords = box.xyxy[0].cpu().numpy().astype(int)
x1, y1, x2, y2 = coords
# Draw rectangle and label
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.putText(frame, 'person', (x1, y1 - 6), cv2.FONT_HERSHEY_SIMPLEX,0.5, (0, 255, 0), 2
```

4.2.3.2 Kết quả mô phỏng



Hình 4.4: Kết quả mô phỏng nhận diện người

Nhận xét:

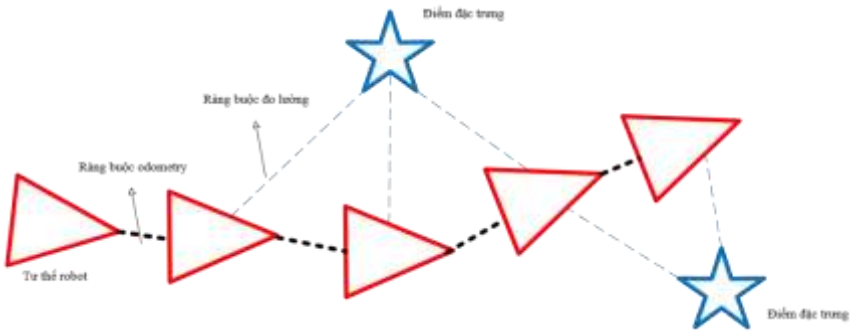
Qua quan sát kết quả nhận diện từ hình ảnh đầu ra, có thể thấy mô hình YOLOv8 hoạt động khá hiệu quả trong việc phát hiện người, với độ chính xác tốt và khả năng vẽ bounding box đúng vị trí đối tượng. Tuy nhiên, thời gian suy luận ghi nhận dao động từ khoảng 100ms đến 800ms cho mỗi khung hình, cho thấy tốc độ xử lý vẫn còn tương đối chậm so với yêu cầu thời gian thực lý tưởng. Nguyên nhân chủ yếu đến từ hạn chế về hiệu năng phần cứng của Raspberry Pi 4 (RAM 4GB), vốn không được tối ưu cho các tác vụ AI nặng, tuy nhiên vẫn đủ để đáp ứng cho đề tài này.

4.3 Thuật toán vẽ bản đồ Graph Slam

4.3.1 Lí Thuyết

4.3.1.1 Tổng quan về Graph Slam

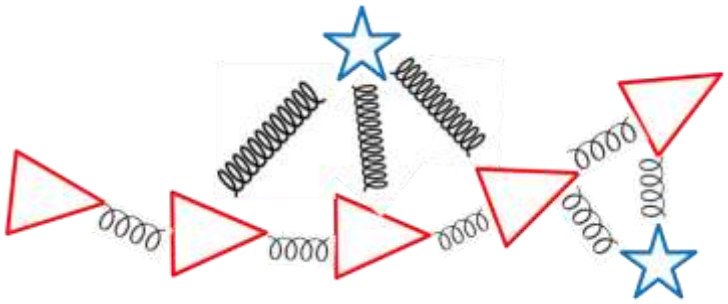
Graph Slam biểu diễn bài toán SLAM dưới dạng một đồ thị, với các nút (nodes) thể hiện tư thế robot (poses) và các điểm đặc trưng của môi trường (landmarks). Các cạnh (edges) của đồ thị thể hiện mối quan hệ (ràng buộc) giữa các nút, bao gồm thông tin về phép đo từ các cảm biến như lidar và odometry.



Hình 4.5: Đồ thị ràng buộc pose–landmark trong Graph SLAM

Khi robot di chuyển tới một vị trí mới, nút mới được thêm vào đồ thị và kết nối với nút trước bằng cạnh chứa thông tin từ odometry. Đồng thời, khi robot phát hiện một landmark mới thông qua các cảm biến, nút mới cũng được thêm vào và kết nối với tư thế của robot bằng cạnh đo lường (measurement constraint).

Quá trình robot quay lại một khu vực đã từng di chuyển qua được gọi là phát hiện đóng vòng lặp (loop closure). Việc thêm các cạnh đóng vòng lặp vào đồ thị giúp hiệu chỉnh lại các sai số tích lũy và cải thiện đáng kể độ chính xác của bản đồ



Hình 4.6: Đồ thị Graph SLAM với ràng buộc pose, landmark và loop closure

4.3.1.2 Cách hoạt động của thuật toán Graph Slam

a. Motion Model

Mô tả cách trạng thái của robot (vị trí và hướng) thay đổi theo thời gian khi nhận các lệnh điều khiển. Dựa vào vị trí, hướng hiện tại và các vận tốc điều khiển, mô hình sẽ dự đoán robot sẽ ở đâu sau một khoảng thời gian ngắn tiếp theo [19].

Đầu vào:

- Trạng thái trước đó
- Tín hiệu điều khiển: Các lệnh như vận tốc tịnh tiến, vận tốc góc hoặc dữ liệu odometry

- Khoảng thời gian: thời gian tác động của lệnh điều khiển

Đầu ra:

- Dự đoán tư thế mới

Công thức:

$$x_t = x_{t-1} + v\cos(\theta_{t-1})\,\Delta t \quad (4.1)$$

$$y_t = y_{t-1} + v\sin(\theta_{t-1})\,\Delta t \quad (4.2)$$

$$\theta_t = \theta_{t-1} + \omega\Delta t \quad (4.3)$$

Với:

- Δt là thời gian tác động của lệnh điều khiển.
- (v, ω) là vận tốc tịnh tiến và vận tốc góc.
- $(x_{t-1}, y_{t-1}, \theta_{t-1})$ là tráng thái trước của Robot.

b. Sensor Model

Dự đoán các giá trị đo được từ các cảm biến của robot, dựa trên trạng thái hiện tại của robot và vị trí các mốc trong môi trường.

Công thức:

Sensor model (Lidar): đo khoảng cách và góc giữa robot và landmark:

$$z_t^i = h(x_t, m_i) + v_t^i \; với \; h(x_t, m_i) = \begin{bmatrix} \sqrt{(m_i x - x_t x)^2 + ((m_i y - x_t y)^2} \\ arctan2(m_i y - x_t y, m_i x - x_t x) - x_t \theta \end{bmatrix} \quad (4.4)$$

c. Hàm lỗi và hàm mục tiêu

Mỗi cạnh (edge) trong đồ thị Graph SLAM đều có một hàm lỗi đặc trưng:

Hàm sai số giữa đo thực tế và đo dự đoán

$$e_{ij} = \begin{Bmatrix} x_j - f(x_i, u_i), \text{đối } với \; odometry \\ z_t^j - h(x_t, m_i), \text{đối } với \; observation \end{Bmatrix} \quad (4.5)$$

Tổng các lỗi này được tổng hợp thành hàm mục tiêu cần tối ưu:

$$J(\{x_t\}, \{m_i\}) = \sum_t \|x_t - f(x_{t-1}, u_{t-1})\|^2_{\sum t} + \sum_{t,i} \left\|z_t^i - h(x_t, m_i)\right\|^2_{\Delta_t^{-1}} \quad (4.6)$$

Trong đó, $\sum_t, \Lambda_t$ là ma trận hiệp phương sai cho từng phép đo:

Vì hàm mục tiêu trên là phi tuyến, quá trình giải quyết gồm:

Tuyến tính hóa hàm lỗi quanh nghiệm hiện tại:

$$e(x + \delta x) \approx e(x) + J\delta x \tag{4.7}$$

Trong đó e(x) là vector lỗi hiện tại:

- J là ma trận Jacobian của e(x).
- δx là bước cập nhật nhỏ.

Tiếp theo ghép các hàm lỗi tuyến tính hóa thành hệ phương trình tổng quát, tìm δx:

$$(A^T A)\delta x = A^T b \tag{4.8}$$

- A là ma trận Jacobian tổng hợp từ tất cả các ràng buộc.
- B là vector phần còn lại từ các sai số hiện tại.

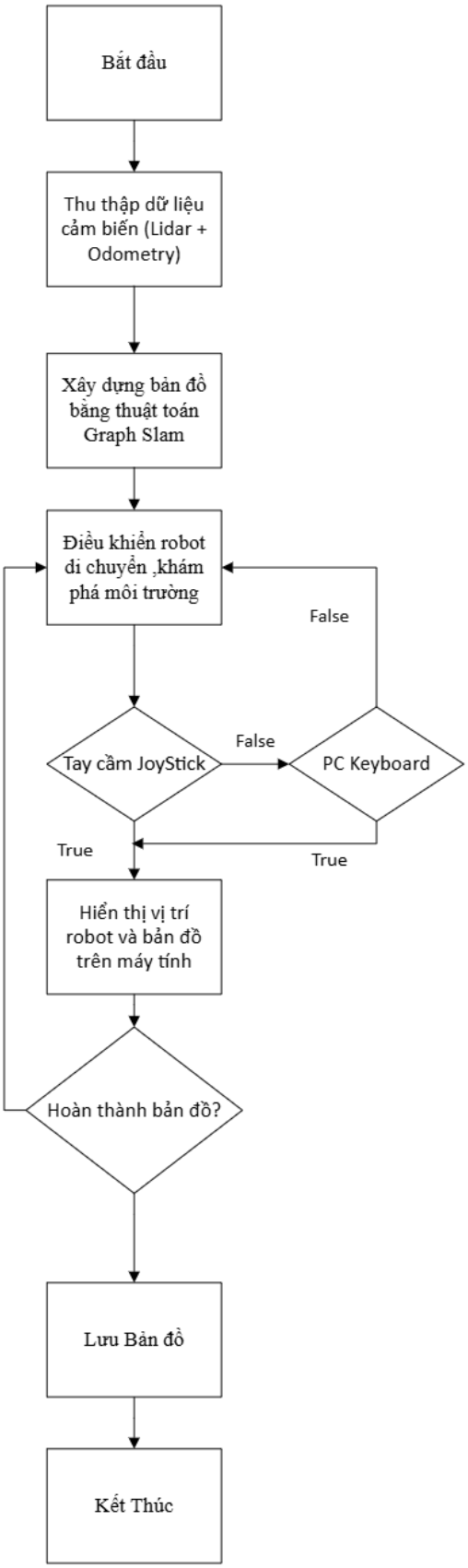
Sau đó cập nhật các giá trị đo lường và lặp lại cho đến khi hội tụ:

$$x_t^{k+1} = x_t^{(k)} + \delta x_t, m_i^{k+1} = m_i^{(k)} + \delta m_i \tag{4.9}$$

Lưu đồ thuật toán sau sẽ biểu diễn tổng quan cách hệ thống tạo bản đồ thông qua thuật toán Graph Slam đã trình bày ở trên:

$$\dot{\varphi}_{Rr} = \frac{1}{r}(v + \omega b)$$

$$\dot{\varphi}_{Lr} = \frac{1}{r}(v - \omega b)$$



Hình 4.7: Lưu đồ thuật toán Tạo bản đồ cho hệ thống

4.3.2 Tiến hành mô phỏng Graph Slam

Cấu hình và khởi tạo node Slam Toolbox:

Để áp dụng thuật toán Graph SLAM vào hệ thống robot AMR, nhóm đã lựa chọn gói mã nguồn mở Slam Toolbox, một trong những giải pháp phổ biến và hiệu quả trong cộng đồng ROS 2. Gói này hỗ trợ đầy đủ cả hai chế độ vận hành: xây dựng bản đồ (mapping) và định vị (localization). Trong quá trình thử nghiệm, hệ thống được cấu hình ở chế độ mapping, cho phép robot vừa di chuyển vừa đồng thời xây dựng và tối ưu hóa bản đồ trong thời gian thực.

Cấu hình hệ thống:

- Slam Toolbox được khởi tạo thông qua file launch và file cấu hình YAML, với các tham số quan trọng như:
  - mode: mapping: bật chế độ xây dựng bản đồ trực tiếp.
  - scan_topic: /scan, odom_frame: odom, base_frame: base_footprint: khai báo các topic và khung tọa độ đầu vào.
  - transform_publish_period: 0.02: cập nhật biến đổi TF ở tần số cao, giúp tăng độ chính xác khi kết hợp dữ liệu.
  - map_update_interval: 5.0: bản đồ được cập nhật mỗi 5 giây.
- Luồng dữ liệu cảm biến và xử lý đầu vào:

Hệ thống nhận dữ liệu từ:

- Cảm biến Lidar (topic /scan) cung cấp các tập dữ liệu dạng tia quét 2D (LaserScan).
- Odometry (topic /odom) lấy từ bộ điều khiển diff_drive_controller, phản ánh chuyển động tương đối của robot.

Slam Toolbox sử dụng scan matching (use_scan_matching: true) để định vị tương đối robot giữa các lần quét, và cập nhật tư thế thông qua bộ lọc tích lũy từ dữ liệu odometry.

- Tối ưu hóa bản đồ và đóng vòng lặp:

Để tăng độ chính xác của bản đồ, hệ thống sử dụng kỹ thuật đóng vòng lặp (loop closure), cho phép nhận diện các vị trí robot đã từng đi qua. Các tham số liên quan bao gồm:

- do_loop_closing: true: bật chức năng đóng vòng lặp.
- loop_search_maximum_distance: 3.0: chỉ thực hiện tìm vòng lặp trong khoảng cách tối đa 3m.
- loop_match_minimum_response_fine: 0.45: ngưỡng tín cậy để xác nhận một vòng lặp.

Khi một vòng lặp được phát hiện, Slam Toolbox sử dụng solver Ceres (solver_plugin: solver_plugins::CeresSolver) với thuật toán tối ưu hóa Levenberg - Marquardt để hiệu chỉnh toàn bộ đồ thị, từ đó giảm sai số tích lũy.

- Các kỹ thuật tăng cường hiệu suất và độ ổn định được cấu hình thông qua các tham số sau:
  - scan_buffer_size: 10, scan_buffer_maximum_scan_distance: 10.0: giới hạn lượng scan lưu trữ để xử lý.
  - minimum_travel_distance: 0.5, minimum_travel_heading: 0.5: chỉ thêm node mới khi robot đã di chuyển đủ xa hoặc xoay đủ lớn.
  - correlation_search_space_dimension: 0.5: thu hẹp không gian tìm kiếm tương quan, giúp tăng tốc quá trình scan matching.

Các thông số như angle_variance_penalty, distance_variance_penalty cũng được hiệu chỉnh để hạn chế nhiễu và cải thiện độ tin cậy của các liên kết trong đồ thị.

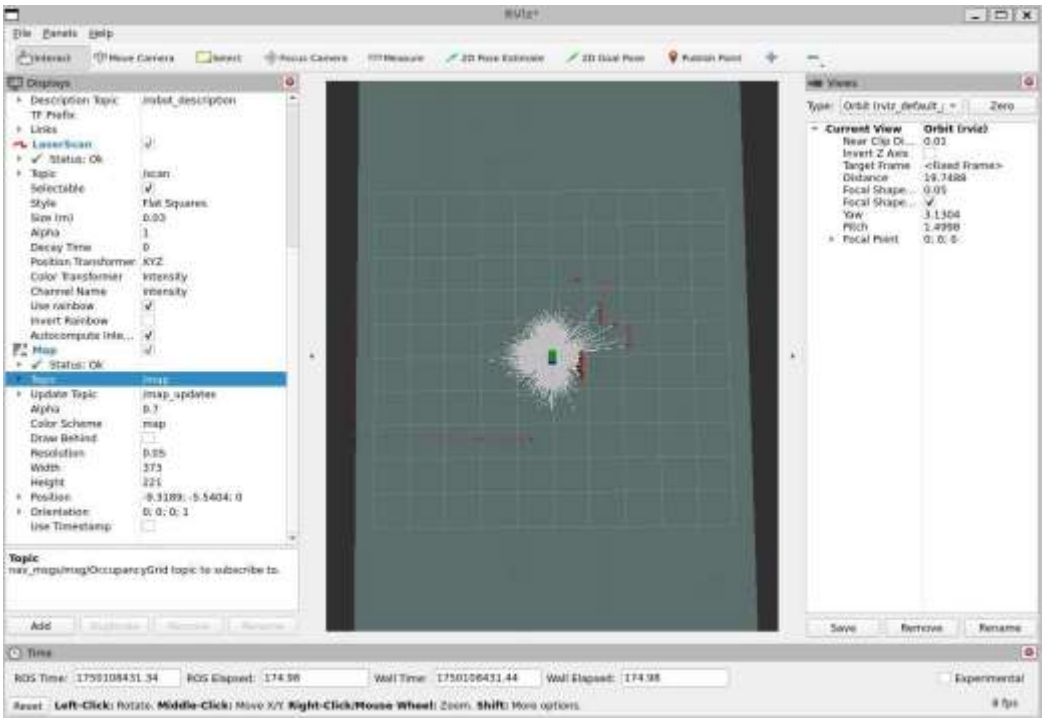
– Lưu bản đồ và sử dụng trong định vị:

Khi quá trình tạo bản đồ kết thúc nhóm sẽ lưu bản đồ bằng lệnh:

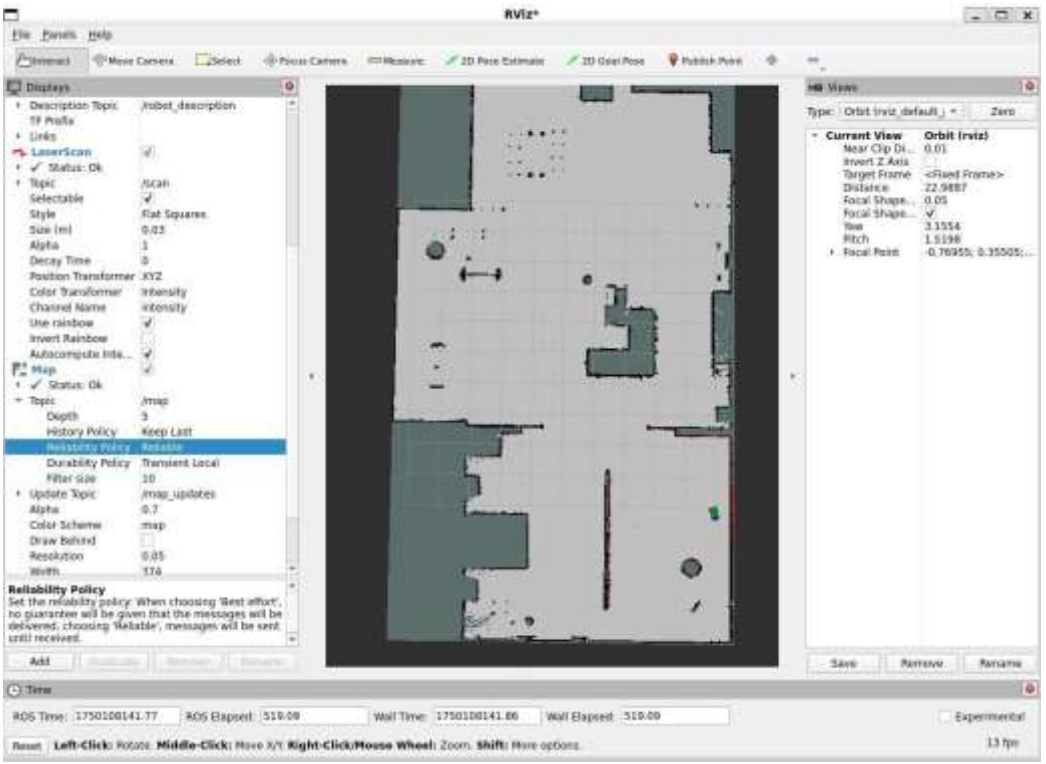
ros2 run nav2_map_server map_saver_cli -t /map -f my_map --free 0.196 --occ 0.65

Sau khi nghiên cứu chi tiết về thuật toán Graph SLAM và cách slam_toolbox thực hiện tối ưu hóa đồ thị từ dữ liệu Lidar và Odometry, nhóm đã tiến hành triển khai thực tế quá trình tạo bản đồ trên hệ thống robot AMR.

Sau đây là kết quả mô phỏng:



Hình 4.8: Bắt đầu chạy Graph Slam



Hình 4.9: Hoàn thành bản đồ

Nhận xét:

Ta thấy việc tạo bản đồ bằng thuật toán Graph SLAM do gói slam_toolbox hỗ trợ khá tốt,và điểm đặc biệt là khi chạy tạo bản đồ thỉnh thoảng xuất hiện hiện tượng robot trên RViz2 bị lệch nhẹ so với vị trí thực tế ngoài đời có thể do nhiễu cảm biến hoặc tích lũy sai số odometry, đặc biệt khi robot di chuyển qua các khu vực nhiều đặc trưng tương tự nhau hoặc mới vào vùng chưa từng khám phá. Tuy nhiên, nhờ đặc tính tối ưu hóa toàn cục và khả năng tự động phát hiện, hiệu chỉnh lỗi (loop closure) của thuật toán Graph SLAM, chỉ sau một thời gian ngắn di chuyển và thu thập thêm dữ liệu, hệ thống nhanh chóng hiệu chỉnh lại, giúp vị trí robot trên bản đồ đồng bộ chính xác với thực tế.

4.4 Xác định vị trí robot với thuật toán AMCL

Trong bài toán định vị lần này thì nhóm sẽ định vị robot trên bản đồ biết trước, và có 2 phương pháp thường được dung là MCL và AMCL là bản mở rộng của nó

4.4.1 Lí thuyết

4.4.1.1 Nguyên lí định vị Monte Carlo

Trong bài toán định vị robot với bản đồ đã có thuật toán Monte Carlo Localization là một giải pháp phổ biến dựa trên nguyên lý bộ lọc xác suất dạng Particle Filter. Ý tưởng chính là sử dụng một tập hợp các hạt để biểu diễn các giả định về vị trí hiện tại của robot trên bản đồ.

Tại thời điểm ban đầu, khi chưa có thông tin gì về robot, các particle $\{x_0^{[i]}\}_{i=1}^{N}$được phát sinh ngẫu nhiên với phân bố đều trên toàn bộ bản đồ sau đó robot nhận được quan sát cảm biến $z_t$có được từ việc quét lidar, thuật toán thực hiện các bước sau:

Bước 1: Cập nhật trọng số

Mỗi particle $x_t^{[i]}$được đánh giá bằng cách tính trọng số:

$$\omega_t^{[i]} = p\left(z_t \middle| x_t^{[i]}, m\right) \qquad (4.10)$$

Trong đó $x_t^{[i]}$là vị trí giả định của particle thứ i

- m là bản đồ môi trường
- z_t là phép đo từ cảm biến

Bước 2: Dự đoán chuyển động:

Khi robot di chuyển với lệnh điều khiển u_t vị trí của từng particle được cập nhật bằng mô hình động học có nhiễu:

$$x_t^{[i]} \sim p\left(x_t \middle| x_{t-1}^{[i]}, u_t\right) \quad (4.11)$$

Bước 3: Tái lấy mẫu:

Các particle có trọng số cao được giữ lại và nhân bản còn các particle sai lệch bị loại bỏ

4.4.1.2 Thuật toán AMCL

Mặc dù MCL là một thuật toán hiệu quả về mặt lý thuyết, nhưng khi áp dụng trong thực tế, nó gặp một số hạn chế:

- Hiệu suất kém: MCL sử dụng số lượng particle cố định, kể cả khi robot đã hội tụ, gây lãng phí tính toán.
- Không tự phục hồi khi mất định vị: Nếu robot bị "lạc", MCL không có cơ chế thêm particle ngẫu nhiên để tìm lại vị trí.
- Thiếu thích nghi: MCL không điều chỉnh hành vi theo độ tin cậy cảm biến hoặc chất lượng đo đạc.

Để khắc phục vấn đề trên AMCL được phát triển như 1 bản mở rộng, để thực hiện thì sẽ bao gồm các bước sau [20]:

Bước 1: Khởi tạo tập particle rỗng.

Bước 2: với mỗi particle $x_t^{[i]}$thực hiện:

Lấy mẫu chuyển động dựa trên odometry:

$$x_t^{[i]} \sim p\left(x_t \middle| x_{t-1}^{[i]}, u_t\right) \quad (4.12)$$

Tính trọng số dựa trên quan sát:

$$\omega_t^{[i]} = p\left(z_t \middle| x_t^{[i]}, m\right) \quad (4.13)$$

Cập nhật tập Particle tạm thời:

$$\overline{X_t} \leftarrow \overline{X_t} \cup \left\{x_t^{[i]}, \omega_t^{[i]}\right\} \quad (4.14)$$

Tính trung bình trọng số:

$$\overline{\omega} \leftarrow \frac{1}{M}\sum_{i=1}^{m} \omega_t^{[i]} \quad (4.15)$$

Bước 3: cập nhật trọng số ngắn hạn và dài hạn

Trọng số ngắn hạn (Trung bình trọng số mới nhất):

$$\omega_{short} \leftarrow \omega_{short} + \alpha_{short}(\overline{\omega} - \omega_{short}) \quad (4.16)$$

Trọng số dài hạn (Trung bình ổn định theo thời gian):

$$\omega_{long} \leftarrow \omega_{long} + \alpha_{long}\left(\overline{\omega} - \omega_{long}\right) \quad (4.17)$$

Trong đó α_{short}, α_{long} là hệ số điều chỉnh học trung bình

Bước 4: Xác suất them particle ngẫu nhiên

Tính xác suất them Particle ngẫu nhiên

$$P_{random} = \max\left(0, 1 - \frac{\omega_{short}}{\omega_{long}}\right) \quad (4.18)$$

Nếu P > 0 bổ sung một số lượng particle ngẫu nhiên rải trên toàn bộ bản đồ

Bước 5: Resample tập particle chính thức: Lấy mẫu lại từ X_t

4.4.2 Tiến hành mô phỏng AMCL

Ở phần trên nhóm đã trình bày tổng quan về cách áp dụng AMCL để định vị robot trong bản đồ có sẵn, nhưng may mắn thay trong thực tế triển khai, nhờ sự phát triển của cộng đồng mã nguồn mở ROS 2, việc hiện thực hóa AMCL đã trở nên đơn giản và thuận tiện hơn. Cụ thể, ROS 2 cung cấp sẵn gói nav2_amcl – một node mạnh mẽ cho phép thực hiện định vị toàn cục (global localization) bằng cách tích hợp các yếu tố như bản đồ tĩnh (occupancy grid), dữ liệu quét Lidar, và thông tin odometry.

Do đó, thay vì phải lập trình thuật toán AMCL từ đầu, nhóm đã tận dụng gói nav2_amcl và tiến hành cấu hình các tham số cần thiết thông qua file .yaml, Node amcl trong ROS 2 được khởi tạo qua launch file Python và sử dụng file YAML để thiết lập các tham số hoạt động. File amcl.yaml được chia thành 4 nhóm chính: ROS2 Interface, Odometry motion model, Laser model, Resampling.

Đối với Ros 2 interface:

Các tham số định nghĩa cách amcl.yaml tương tác với khung tọa độ trong hệ thống ros2:

- base_frame_id: "base_footprint": Khung tọa độ gắn vào thân robot.

- global_frame_id: "map": Khung toàn cục cần định vị.
- odom_frame_id: "odom" – Khung tọa độ theo dõi chuyển động tương đối.
- scan_topic: "/scan": Chủ đề chứa dữ liệu từ Lidar 2D.
- tf_broadcast: true: Cho phép AMCL phát TF từ map → base_footprint, chính là kết quả định vị.
- transform_tolerance: 1.0: Độ trễ tối đa cho phép trong việc xử lý biến đổi tọa độ.

Đối với Odometry Motion Model: Mô hình này giúp AMCL dự đoán chuyển động của robot dựa trên dữ liệu odometry

robot_model_type: nav_amcl::DifferentialMotionModel: Dạng robot bánh vi sai.

Các hệ số nhiễu:

- alpha1, alpha2, alpha3, alpha4: 0.2 – Mức độ nhiễu cho từng thành phần chuyển động (tiến, xoay).

Đối với Sensor Model: Dựa trên so sánh giữa dữ liệu Lidar và bản đồ để đánh giá độ tin cậy vị trí:

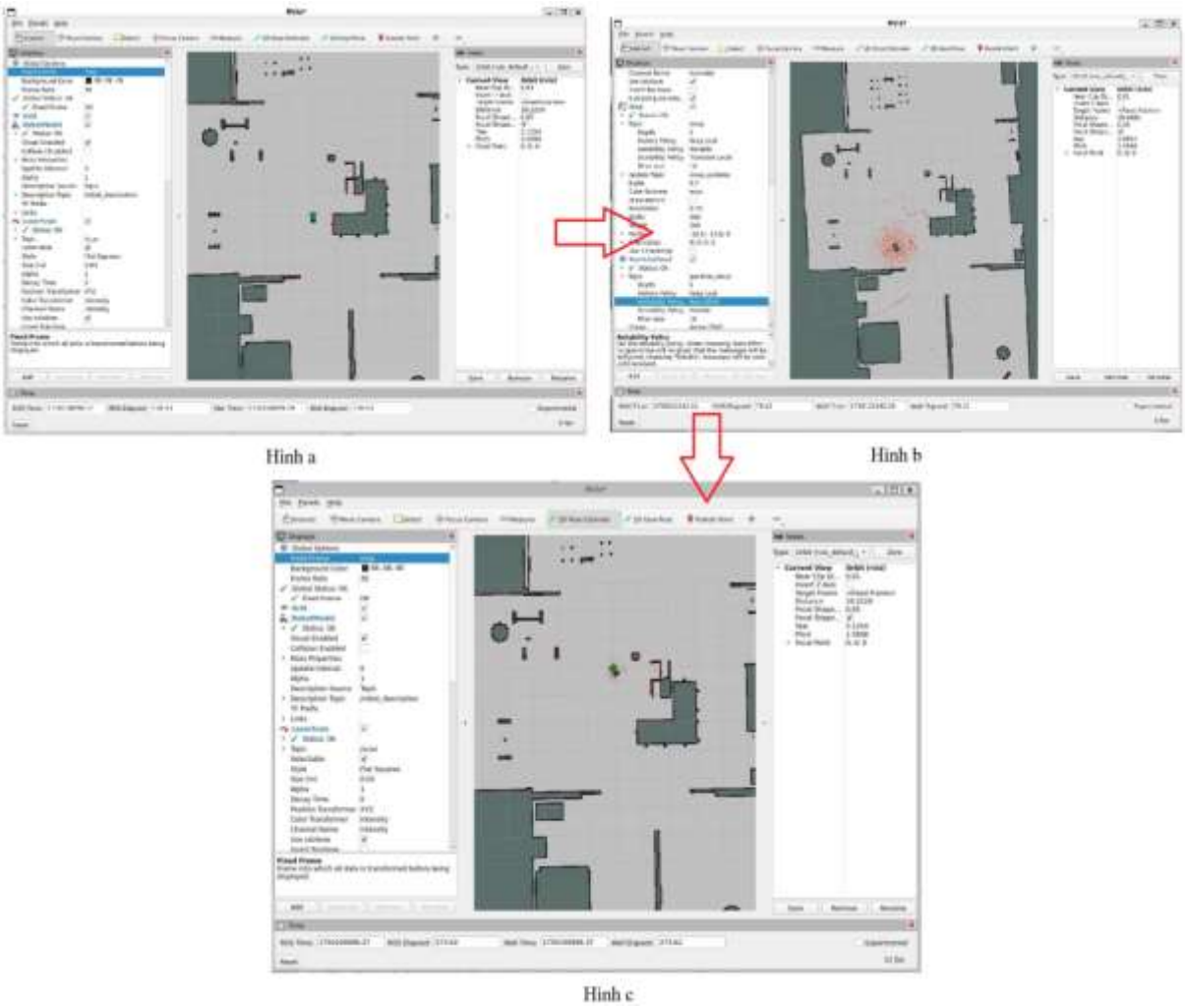
- laser_model_type: "likelihood_field"
- z_hit: 0.95, z_rand: 0.05 – Mô hình kết hợp xác suất đo đúng và nhiễu ngẫu nhiên.
- sigma_hit: 0.2 – Độ lệch chuẩn của phép đo.

Đối với Cơ chế cập nhật và tái lấy mẫu particle:

Hệ thống sử dụng particle filter để liên tục cập nhật ước lượng vị trí robot:

- resample_interval: 1 – Tái lấy mẫu ở mỗi chu kỳ.
- min_particles: 500, max_particles: 2000 – Giới hạn số lượng particle.
- update_min_d: 0.25 (m) – Cập nhật nếu robot di chuyển ít nhất 25cm.
- update_min_a: 0.2 (rad) – Cập nhật nếu xoay tối thiểu 0.2 rad.
- recovery_alpha_fast, recovery_alpha_slow: 0.0 – Có thể kích hoạt để tăng khả năng phục hồi khi robot bị "lạc".

Sau đây là kết quả mô phỏng:



Hình 4.10: Kết quả mô phỏng AMCL

Nhận xét:

Ở **Hình 4.10a,** ta sử dụng bản đồ môi trường đã tạo từ thuật toán Graph SLAM ở phần trước. Khi bắt đầu khởi động trong RViz2, robot được gán vào một vị trí gốc (0, 0) cố định trên bản đồ, do đó các điểm laserpoint (màu tím) trùng khớp khá tốt với các đặc trưng của môi trường xung quanh trên bản đồ tĩnh. Sang **Hình 4.10b**, nếu ta di chuyển robot vật lý đến vị trí khác bằng cách nhấc xe lên để chỗ khác hoặc để yên robot và dùng tính năng 2D Pose Estimate để đặt lại vị trí robot trên bản đồ, các laserpoint sẽ xuất hiện lệch so với bản đồ tĩnh. Điều này phản ánh thực tế là robot trên RViz2 và robot thật không còn đồng bộ vị trí với nhau. Ở **Hình 4.10c**, khi bật thuật toán AMCL (Adaptive Monte Carlo Localization), hệ thống bắt đầu sinh ra các particle màu đỏ biểu diễn các giả định về vị trí robot trên bản đồ. Sau một thời gian robot di chuyển trong môi trường,

thuật toán AMCL sử dụng dữ liệu quét Lidar và odometry để hội tụ các particle về đúng vị trí thực tế của robot. Kết quả là robot trên RViz2 được đồng bộ trở lại với robot thực ngoài đời – thể hiện qua việc các laserpoint lại trùng khớp với các đặc trưng bản đồ. Qua đó ta thấy thuật toán AMCL hoạt động khá tốt.

4.5 Costmap

4.5.1 Tổng quan về costmap

Costmap là một mô hình bản đồ không gian hai chiều được ứng dụng rộng rãi trong hệ thống điều hướng robot di động, đặc biệt trong các nền tảng sử dụng ROS 2. Trong mô hình này, môi trường được chia thành một lưới ô (cell), mỗi ô đại diện cho một khu vực nhỏ và mang một giá trị chi phí phản ánh mức độ an toàn hoặc rủi ro khi robot di chuyển qua khu vực đó.

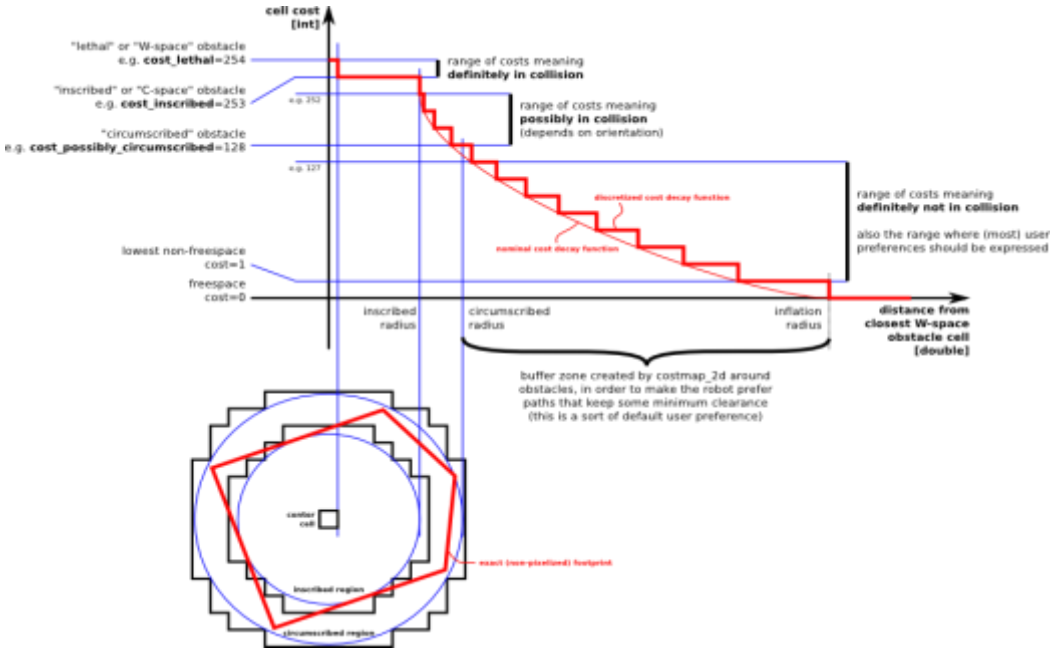
Giá trị chi phí của mỗi ô được cập nhật liên tục dựa trên thông tin tổng hợp từ nhiều nguồn:

- Bản đồ tĩnh từ map_server: Thể hiện các đối tượng cố định như tường, vách ngăn.
- Cảm biến động (Lidar, sonar, camera): Phát hiện vật cản thời gian thực.
- Lớp lạm phát (inflation_layer): Mở rộng vùng nguy hiểm xung quanh vật cản, đảm bảo robot giữ khoảng cách an toàn khi di chuyển.

Costmap phân loại không gian thành năm vùng theo mức độ rủi ro:

- Lethal Obstacle (254–255): Đại diện cho ô có vật cản nguy hiểm (vật thể rắn như tường, hộp, người…). Robot tuyệt đối không được di chuyển vào các ô này, vì điều đó đồng nghĩa với va chạm chắc chắn.
- Inscribed Inflated Obstacle (~253): Các ô nằm trong bán kính va chạm so với vật cản. Nếu tâm robot nằm trong vùng này, khả năng va chạm là rất cao. Thường được định nghĩa là chi phí xấp xỉ 253, gần sát ngưỡng nguy hiểm.
- Possibly Circumscribed (~128–252): Vùng lạm phát – chi phí giảm dần theo khoảng cách từ vật cản. Trong vùng này, robot vẫn có thể đi qua, nhưng cần đánh giá kỹ hướng di chuyển và kích thước robot.
- Free Space (0): Đại diện cho khu vực an toàn tuyệt đối, không có vật cản, robot có thể tự do di chuyển.

– Unknown Space (255) hoặc -1 (NO_INFORMATION): Vùng chưa có dữ liệu cảm biến (ví dụ: ngoài phạm vi Lidar, chưa từng được quét). Robot cần thận trọng khi đi vào, vì không xác định được mức độ an toàn.



Hình 4.11: Các giá trị của bản đồ trọng số ảnh hưởng tới hoạt động của robot

4.5.2 Tiến hành mô phỏng Costmap

Trong ROS 2, gói nav2_costmap_2d đã hỗ trợ đầy đủ cho việc xây dựng và quản lý costmap thông qua cơ chế plugin. Các lớp trong costmap có thể dễ dàng được cấu hình thông qua tệp YAML, mang lại sự linh hoạt trong quá trình thiết kế hệ thống điều hướng.

Vì vậy, trong đề tài này, nhóm đã sử dụng nav2_costmap_2d để triển khai hệ thống costmap, phục vụ cho cả quá trình lập kế hoạch toàn cục (global planner) và tránh vật cản cục bộ (local planner). Cấu hình costmap gồm ba lớp chính: static_layer, obstacle_layer và inflation_layer, được điều chỉnh để phù hợp với robot thực nghiệm của nhóm.

4.5.2.1 Lớp static_layer

– Nhiệm vụ: Có nhiệm vụ tải bản đồ tĩnh từ topic /map được phát bởi node map_server. Bản đồ này thường được xây dựng trước bằng các thuật toán SLAM hoặc nạp sẵn từ file .pgm/.yaml, và chứa thông tin về các chướng ngại vật cố định như tường, cột, vách ngăn,…

– Cấu hình: static_layer:

- plugin: "nav2_costmap_2d::StaticLayer"
- map_topic: /map
- map_subscribe_transient_local: true

Tham số map_subscribe_transient_local được bật nhằm đảm bảo các node subscriber mới luôn nhận được bản đồ hiện tại, kể cả khi chủ đề đã được phát trước đó. Điều này giúp hệ thống duy trì tính nhất quán trong quá trình cập nhật và sử dụng costmap.

4.5.2.2 Lớp obstacle_layer

Nhiệm vụ: Chịu trách nhiệm xử lý dữ liệu cảm biến thời gian thực từ. Dữ liệu này được sử dụng để đánh dấu (marking) các vật cản mới xuất hiện và xóa (clearing) các vật cản không còn trong tầm quét. Cơ chế này giúp bản đồ chi phí phản ánh kịp thời sự thay đổi của môi trường xung quanh robot.

Cấu hình: obstacle_layer:

- plugin: "nav2_costmap_2d::ObstacleLayer"
- enabled: true
- observation_sources: scan
- scan:
- topic: /scan
- data_type: "LaserScan"
- marking: true
- clearing: true
- obstacle_min_range: 0.0
- obstacle_max_range: 2.5
- raytrace_min_range: 0.0
- raytrace_max_range: 3.0
- max_obstacle_height: 2.0

Các thông số như obstacle_max_range và raytrace_max_range xác định phạm vi phát hiện và xử lý vật thể, trong đó, raytracing được dùng để xác định vùng không còn vật cản nhằm cập nhật lại costmap. Tham số max_obstacle_height giới hạn chiều cao của

vật thể cần xét, giúp loại bỏ các đối tượng không liên quan như vật thể treo cao hoặc thấp hơn tầm di chuyển của robot.

c. Lớp Inflation

Nhiệm vụ: Mở rộng vùng nguy hiểm xung quanh các vật cản đã được phát hiện, nhằm tạo ra một vùng đệm an toàn giúp robot duy trì khoảng cách hợp lý trong quá trình di chuyển. Đây là lớp hỗ trợ quan trọng để tránh va chạm trong các không gian hẹp hoặc khi có sai số điều khiển.

Cấu hình: inflation_layer:

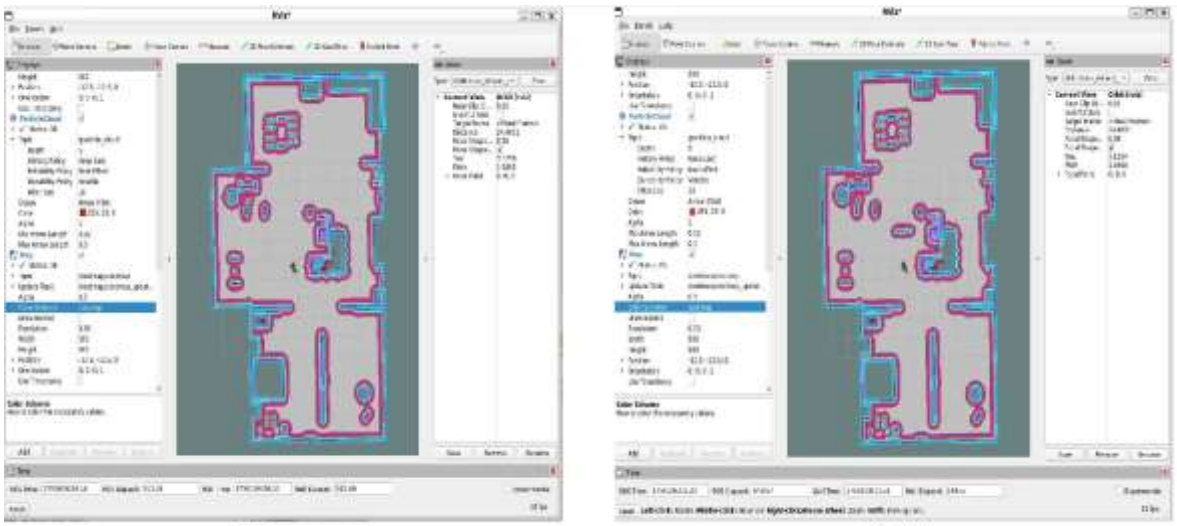
- plugin: "nav2_costmap_2d::InflationLayer"
- inflation_radius: 0.35
- cost_scaling_factor: 3.0

Tham số inflation_radius xác định bán kính tối đa mà chi phí được mở rộng tính từ vật cản (ở đây là 0.35m).

cost_scaling_factor điều chỉnh tốc độ giảm dần của chi phí theo khoảng cách – hệ số càng cao thì chi phí giảm càng nhanh và vùng chi phí cao sẽ thu hẹp lại.

Nhờ lớp lạm phát, các thuật toán điều hướng có thể đưa ra lộ trình tránh xa các vùng có nguy cơ cao, đảm bảo an toàn cho robot trong cả môi trường tĩnh và môi trường có chướng ngại vật động

<u>Kết quả mô phỏng:</u>



Hình a Hình b

Hình 4.12: Kết quả mô phỏng Costmap

Nhận xét:

Ở **Hình 4.12a** là khi ta bật costmap với bản đồ tĩnh (bản đồ có sẵn đã được tạo từ thuật toán Graph Slam ở trên) và **Hình 4.12b** là bản đồ costmap với vật cản động đặt ở phía sau robot và tổng quan thì bản đồ sẽ có 4 màu:

- Màu trắng xám: Là khu vực tự do
- Màu hồng đỏ là: Vùng inflation chi phí thấp
- Màu xanh dương nhạt: Vùng inflation chi phí cao gần như không tạo đường đi qua đó được, vùng xanh lá ngoài bản đồ là vùng không xác định

4.6 Thuật toán A* hoạch định đường đi

*4.6.1 Tổng quan thuật toán A**

Trong điều hướng cho robot, để robot đến được vị trí mong muốn, ta phải đưa ra một kế hoạch đường đi từ điểm bắt đầu (vị trí hiện tại) đến điểm kết thúc đường đi, ta cần phải chọn đường đi ngắn nhất giữa 2 điểm đó, và đối với đề tài này nhóm sử dụng thuật toán A* để tìm đường đi từ điểm bắt đầu tới điểm đích một cách ngắn và hiệu quả nhất, đây là một thuật toán tìm kiếm theo cơ chế heuristic, tức là nó sử dụng thông tin dự đoán để hướng dẫn quá trình tìm kiếm. Thuật toán này kết hợp giữa tìm kiếm theo chi phí thấp nhất (like Dijkstra) và tìm kiếm theo chiều sâu (depth-first search), giúp tìm ra đường đi ngắn nhất một cách nhanh chóng. Thuật toán A* **(**Lỗi! Không tìm thấy nguồn tham chiếu.**)** là một trong những giải pháp hiệu quả và phổ biến nhất để tìm đường đi tối ưu có chi phí thấp nhất trên đồ thị, trong đó, chi phí đường đi được định nghĩa là các tổng các trọng số dương của các cạnh trên đường đi [21].

Thuật toán A*:

1: OPEN ← {1}

2: past_cost[1] ← 0, past_cost[node] ← infinity for node ∈ {2, ..., N}

3: **while** OPEN is not empty **do**

4: current ← first node in OPEN, remove from OPEN

5: add current to CLOSED

6: **if** current is in the goal set **then**

7: **return** SUCCESS and the path to current

8: **end if**

9: **for** each nbr of current not in CLOSED do

10: tentative_past_cost ← past_cost[current] + cost[current, nbr]

11: **if** tentative_past_cost < past_cost[nbr] then

12: past_cost[nbr] ← tentative_past_cost

13: parent[nbr] ← current

14: put (or move) nbr in sorted list OPEN according to

est_total_cost[nbr] ← past_cost[nbr] + heuristic_cost_to_go(nbr)

15: **end if**

16: **end for**

17: end while

18: return FAILURE

Giả sử đồ thị được mô tả bằng tập các nút $N = \{1, 2, \dots., N\}$ trong đó nút 1 là khởi đầu. bộ cạnh được lưu trong ma trận $cost[node1, node2]$, với mỗi phần tử đại diện cho chi phí di chuyển từ node 1 đến node2.Nếu không có cạnh nối giữa 2 nút, phần tử tương ứng sẽ mang giá trị âm.

A* sử dụng các cấu trúc dữ liệu chính sau:

- Open: Danh sách các nút đang chờ được mở rộng được sắp xếp theo tổng chi phí ước lượng nhỏ nhất
- Closed: Danh sách các nút đã được mở rộng
- Past_cost[node]: Chi phí nhỏ nhất tìm được cho đến hiện tại để đi từ nút khởi đầu đến node
- Parent[node]: Mảng thể hiện cây tìm kiếm trong đó mỗi nút lưu lại nút cha gần nhất để truy vết đường đi sau cùng.

Quá trình tạo đường đi:

Khởi tạo: Open: Khởi tạo với nút bắt đầu 1, Past_cost[1]=0 tất cả cá past_cost [node≠ 1) được đặt là vô cùng, Parent[1] để trống hoặc gắn null

- Tiến trình thuật toán:

1. Tại mỗi vòng lặp lấy nút đầu tiên trong danh sách OPEN (nút có tổng chi phí ước lượng nhỏ nhất) và đặt là current.
2. Nếu current nằm trong tập nút đích, thuật toán kết thúc, đường đi tối ưu được truy vết ngược từ current về start thông qua mảng parent
3. Nếu không them current vào Closed, duyệt qua tất cả các nút láng giềng nbr của current chưa nằm trong CLOSED.
4. Tính chi phí tạm thời để đi đến nbr:

$$tentative\ past\ cost = pastcost[current] + cost[current, nbr]$$

5. Nếu tentative_past_cost < past_cost[nbr], cập nhật:

$$pastcost[nbr] = tentative\ past\ cost$$

$$parent[nbr] = current$$

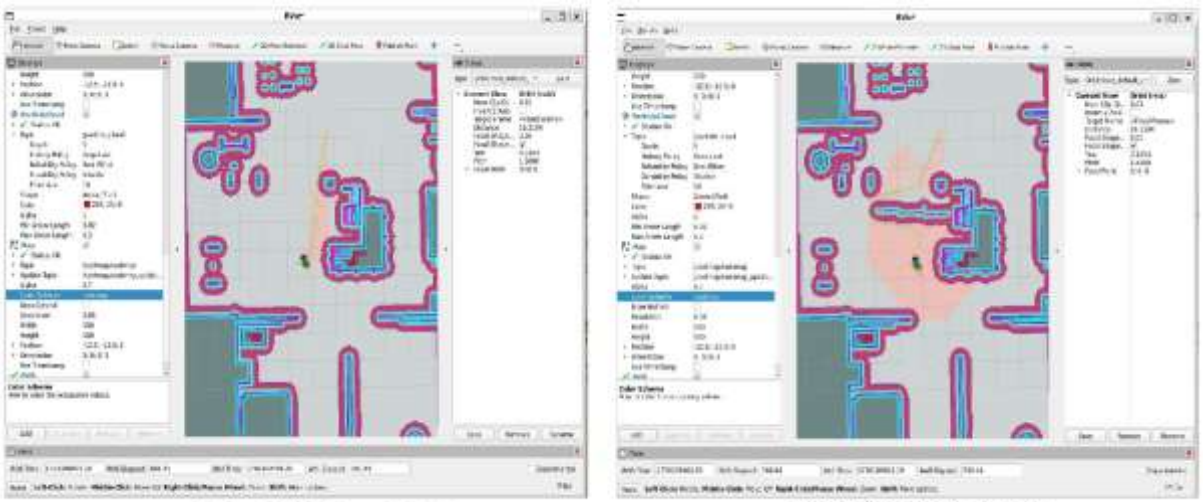
6. Thêm hoặc di chuyển nbr vào OPEN, được sắp xếp theo tổng chi phí ước lược:

$$est\ total\ cost[nbr] = pastcost[nbr] + heuristic\_cost[nbr]$$

Hàm heuristic là ước lượng chi phí thấp nhất từ node đến nút đích. Để đảm bảo tìm ra đường đi tối ưu, Heuristic cần thỏa mãn tính không đánh giá cao(admissible) tức là luôn nhỏ hơn hoặc bằng chi phí thực tế.

*4.6.2 Tiến hành mô phỏng thuật toán A**

Kết quả mô phỏng thuật toán A* khi có vật cản và không có vật cản:



Hình a: Khi không có vật cản

Hình b: Khi có vật cản động

Hình 4.13: Kết quả mô phỏng thuật toán A*

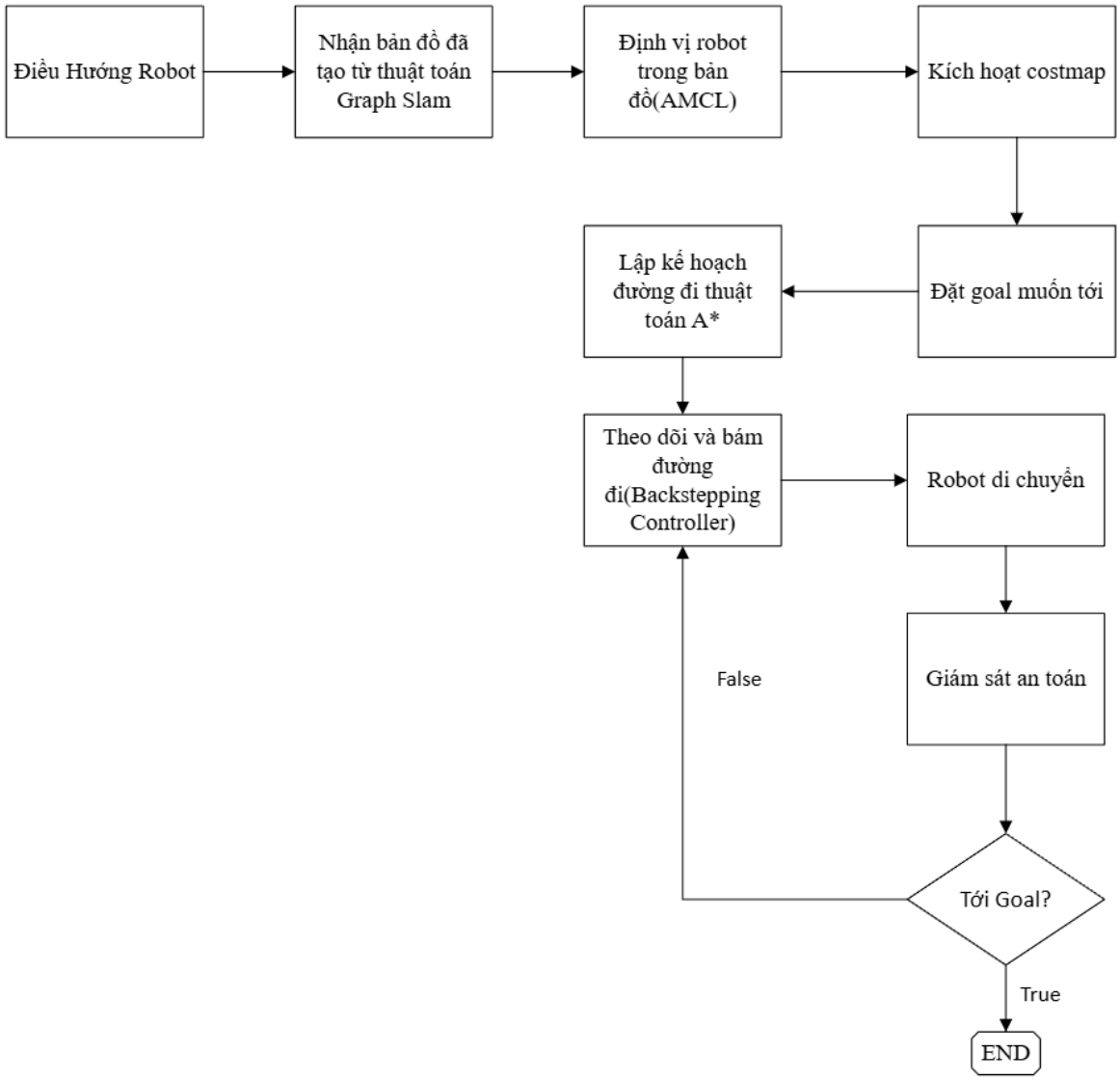
Nhận xét:

Để có thể tạo đường đi A* như kết quả ở **Hình 4.13**, trước tiên ta cần phải chọn điểm đích mà robot muốn đến bằng cách dung 2D goal Pose trong rviz2, sau khi chọn thuật toán sẽ vẽ đường đi từ robot tới vị trí đích, đường đi đó được biểu diễn là đường màu xanh lá cây trong hình, ta nhận thấy đường được tạo ra có hơi cong nhưng khá mượt, giúp robot dễ dàng có thể bám theo.

4.7 Điều khiển bám đường đi và Né vật cản

4.7.1 Điều khiển bám đường đi

Hệ thống điều hướng robot sử dụng thuật toán A* để lập kế hoạch đường đi và thuật toán Backstepping Controller (kết hợp PID) để bám đường (**Hình 4.14**) và điều khiển robot di chuyển an toàn trong môi trường có vật cản.



Hình 4.14: Lưu đồ quy trình điều hướng robot bám đường đi

Nguyên lý hoạt động:

Hệ thống điều hướng robot được thiết kế với chu trình xử lý tuần tự từ khởi tạo bản đồ đến điều khiển và giám sát an toàn trong môi trường thực. Đầu tiên, robot sử dụng bản đồ đã được xây dựng từ trước bằng thuật toán Graph SLAM, bản đồ này được nạp vào hệ thống qua map_server. Sau đó, robot tiến hành định vị trong bản đồ thông qua thuật toán AMCL (Adaptive Monte Carlo Localization) sử dụng dữ liệu LiDAR và odometry để xác định chính xác vị trí hiện tại của robot trên bản đồ tĩnh.

Tiếp theo, hệ thống kích hoạt costmap, trong đó bao gồm cả global_costmap và local_costmap. Costmap sẽ kết hợp giữa bản đồ tĩnh và các thông tin cảm biến động từ LiDAR để phát hiện vật cản thời gian thực. Người dùng sau đó sẽ đặt một điểm đến

(goal), và hệ thống sẽ sử dụng thuật toán A* để lập kế hoạch đường đi toàn cục từ vị trí hiện tại đến vị trí mục tiêu, dựa trên global_costmap.

Đường đi toàn cục sau khi được lập bởi thuật toán A* sẽ được truyền đến bộ điều khiển chuyển động. Tại đây, hệ thống sử dụng Backstepping Controller để điều khiển robot bám sát quỹ đạo đã định, đảm bảo chuyển động mượt mà, ổn định và chính xác trong suốt quá trình di chuyển. Trong khi thực thi lộ trình, robot được giám sát liên tục bởi hệ thống Safety Supervisor, với nhiệm vụ theo dõi các nguy cơ tiềm ẩn trong môi trường xung quanh. Hệ thống này khai thác dữ liệu từ cảm biến LiDAR và tín hiệu phát hiện người từ mô hình YOLOv8, nhằm phát hiện kịp thời các tình huống nguy hiểm như vật cản di động hoặc con người xuất hiện bất ngờ trong vùng hoạt động. Khi phát hiện sự cố, robot sẽ lập tức đưa ra phản ứng phù hợp đảm bảo quá trình vận hành luôn an toàn và thích ứng với môi trường động.

Kết quả mô phỏng bám đường đi của xe:

Hình 4.15: Quá trình mô phỏng bám đường đi

Nhận xét:

Robot bám đường khá tốt , tuy nhiên ở đường cong thì robot lúc đầu bám tâm robot hơi lệch so với đường đi nhưng rất nhanh chóng có thể ổn định lại.

4.7.2 Né vật cản

Nhóm sẽ điều khiển robot né vật cản dựa vào hai tác động chính:

- Con người xuất hiện trong vùng làm việc.
- Vật cản động bất ngờ xuất hiện trên đường đi do thuật toán A lập kế hoạch, được phát hiện bởi LIDAR.

4.7.2.1 Đối với con người

Như đã trình bày ở phần áp dụng AI, thuật toán YOLOv8 sẽ liên tục phân tích hình ảnh từ camera để nhận diện con người xuất hiện trong khu vực làm việc của robot. Khi phát hiện người, hệ thống sẽ gửi tín hiệu tới topic /person_detected, node safety_supervisor sau đó sẽ subcribe để xử lí.

4.7.2.2 Đối với vật cản động bất ngờ (từ LIDAR)

Robot sử dụng dữ liệu LIDAR để kiểm tra xem vật cản có xuất hiện gần đường đi (path) do thuật toán A* tạo ra hay không, thông qua các bước tính toán sau:

- Tính toán vị trí vật cản trong hệ robot:

$$x_{lidar} = r \times \cos(\alpha)\ , y_{lidar} = r \times \sin(\alpha) \qquad (4.19)$$

Với r là khoảng cách đo được và α là góc quét từng tia lidar

- Tiếp theo chuyển sang hệ bản đồ (map):

Sử dụng phép biến đổi tọa độ (TF), điểm vật cản được chuyển sang hệ tọa độ toàn cục

$$\begin{bmatrix} x_{map} \\ y_{map} \\ z_{map} \\ 1 \end{bmatrix} = T_{robot \to map} \begin{bmatrix} x_{lidar} \\ y_{lidar} \\ 0 \\ 1 \end{bmatrix} \qquad (4.20)$$

Trong đó:

- (x_{lidar}, y_{lidar}) là vị trí vật cản trong hệ robot.
- (x_{map}, y_{map}) là vị trí vật cản trong hệ bản đồ toàn cục.

- $T_{robot \to map}$ là ma trận đồng nhất 4x4 mô tả vị trí và hướng của robot trong bản đồ.

Gom nhóm các điểm laserpoint gần nhau:

Nếu các điểm laserpoint có khoảng cách nhỏ hơn một ngưỡng xác định chúng sẽ được gom lại thành 1 cụm để tránh nhận diện trùng lặp thông thường ta chọn điểm gần robot nhất trong ụm đề đại diện vị trí vật cản:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (4.21)$$

$d_{ij} < d_{cluster}$ thì hai điểm I và j thuộc cùng một cụm

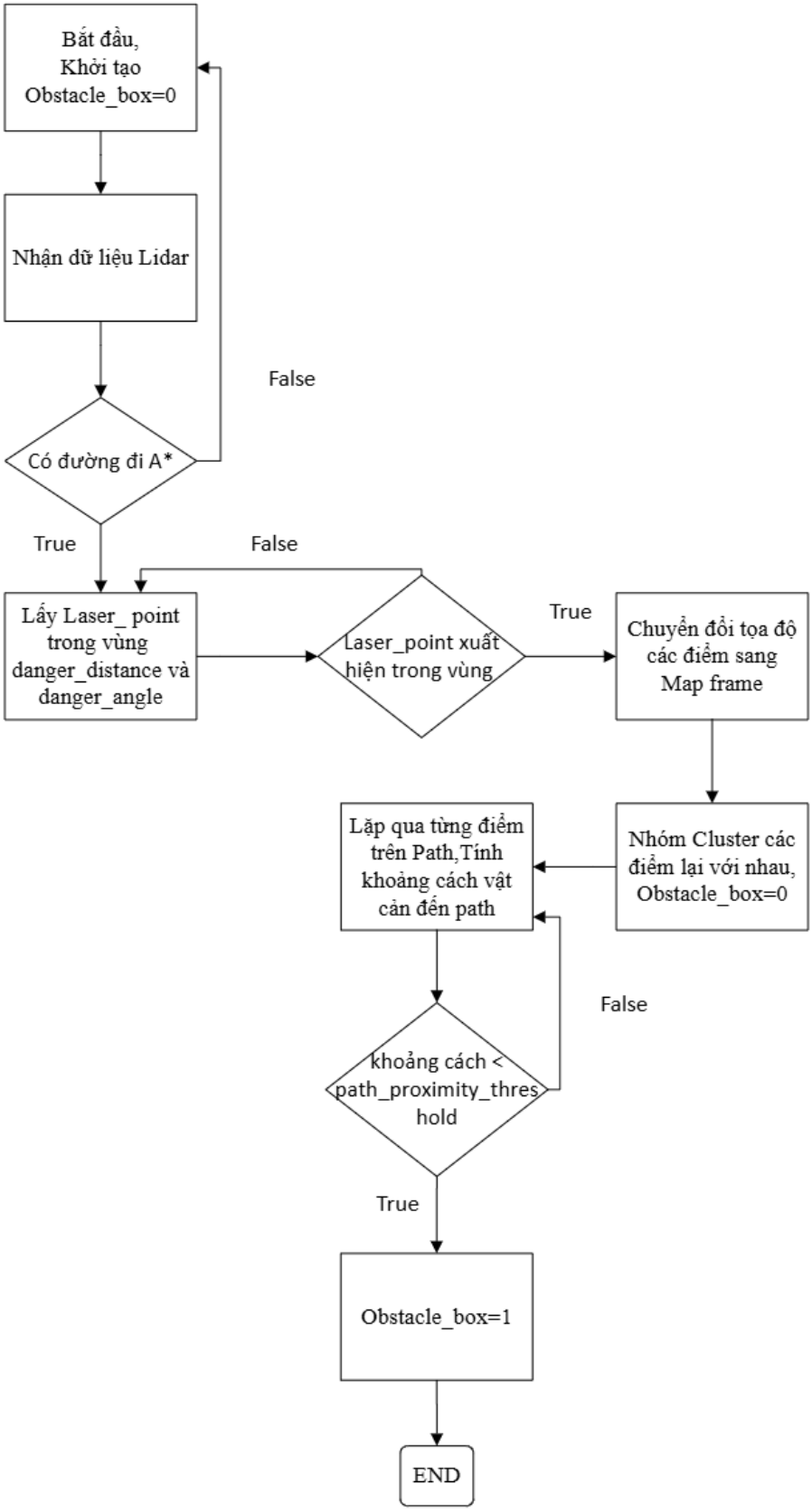
- Kiểm tra vị trí vật cản so với đường đi:

Với mỗi vật cản đại diện tính khoảng cash từng điểm trên đường đi đã lập kế hoạch

$$d_{path} = min\sqrt{(x_{map} - x_k)^2 + (y_{map} - y_k)^2} \quad (4.22)$$

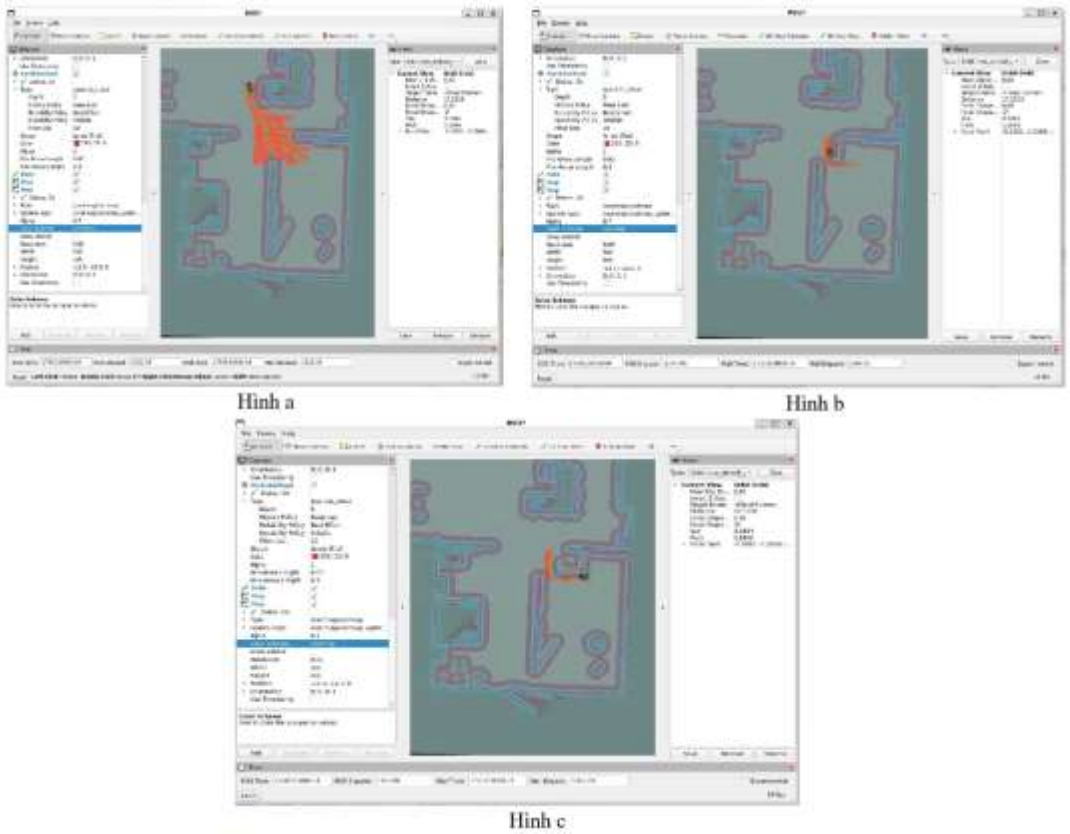
Nếu $d_{path} < d_{threshold}$, robot kích hoạt chế độ né vật cản gửi tín hiệu vào topic obstacle_box để safety_supervisor xử lí.

Và quá trình xử lí nhận dạng vật cản gần đường đi sẽ được biểu diễn dưới dạng lưu đồ thuật toán như sau:



Hình 4.16: Lưu đồ thuật toán xử lý nhận dạng vật cản

Kết quả mô phỏng né vật cản bất ngờ xuất hiện trên đường đi

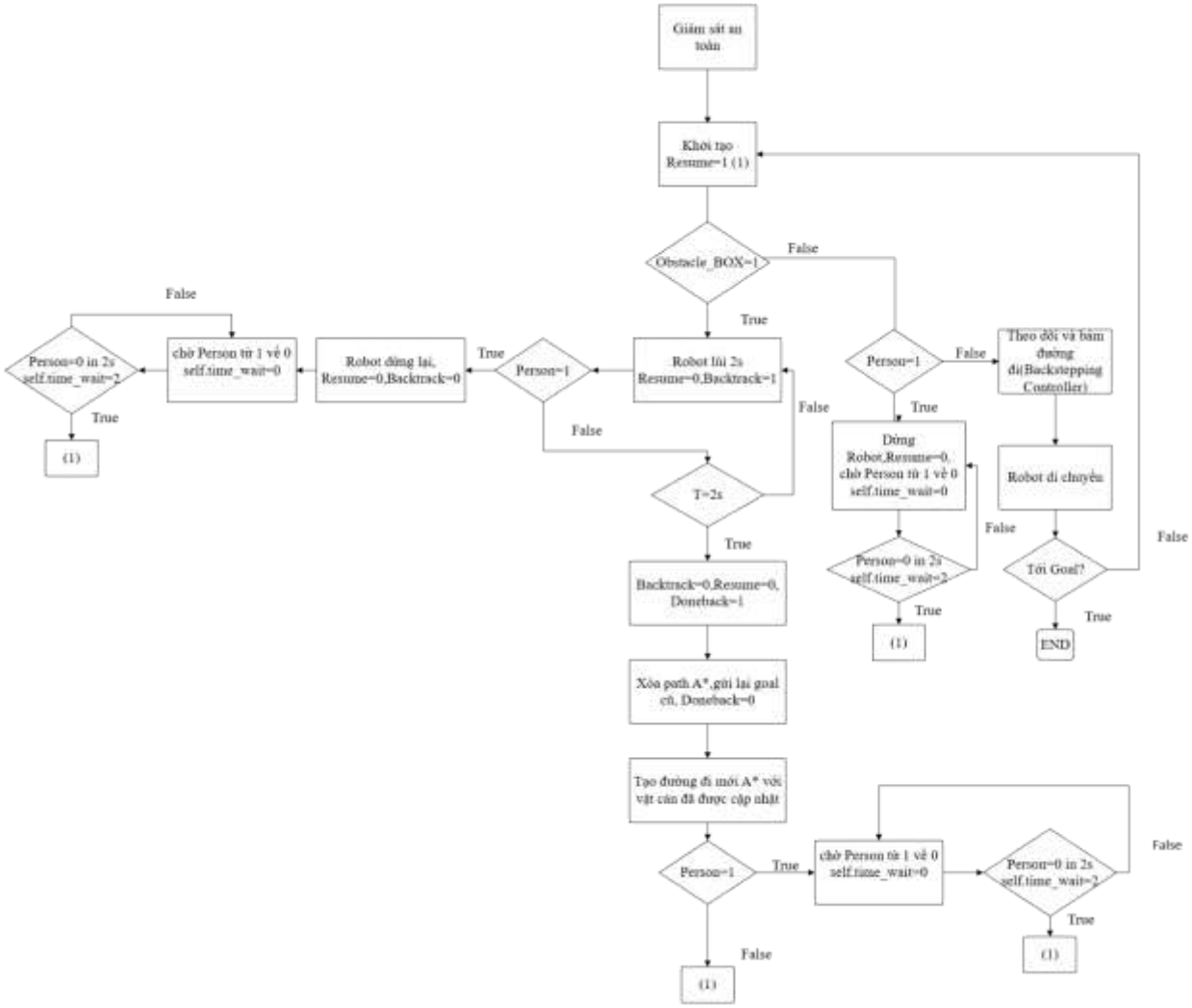


Hình 4.17: Kết quả mô phỏng né vật cản bất ngờ

Nhận xét:

Mô phỏng cho thấy xe có khả năng phát hiện và né tránh vật cản bất ngờ một cách hiệu quả, nhờ kết hợp cảm biến LiDAR và thuật toán điều hướng thời gian thực. Hệ thống phản hồi nhanh, duy trì được quỹ đạo an toàn mà không cần can thiệp thủ công, thể hiện tính linh hoạt và độ tin cậy cao trong môi trường phức tạp.

Sau khi hệ thống đã tích hợp khả năng phát hiện vật cản bằng Lidar và nhận diện người bằng camera sử dụng mô hình YOLOv8, toàn bộ logic né tránh vật cản sẽ được thể hiện thông qua lưu đồ thuật toán sau:



Hình 4.18: Lưu đồ thuật toán Nguyên lí né vật cản robot

Nguyên lý hoạt động hệ thống điều hướng và né vật cản:

Hệ thống điều hướng và né vật cản của robot **Hình 4.18,** được thiết kế dựa trên sự phối hợp giữa các module lập kế hoạch đường đi (A* planner), điều khiển bám đường (Backstepping Controller), định vị (AMCL), nhận diện vật cản bằng Lidar, nhận diện người bằng YOLO và giám sát an toàn tổng thể thông qua Safety Supervisor. Sau khi khởi động, robot nhận bản đồ môi trường xây dựng sẵn từ thuật toán Graph SLAM và xác định vị trí hiện tại nhờ AMCL. Khi người dùng đặt vị trí đích, thuật toán A* sẽ lập kế hoạch, tạo ra đường đi tối ưu từ vị trí hiện tại tới đích, gửi sang bộ điều khiển bám đường. Bộ điều khiển Backstepping liên tục điều chỉnh vận tốc để robot bám sát quỹ đạo tham chiếu, đồng thời cập nhật bản đồ động quanh robot dựa trên dữ liệu Lidar để phát hiện vật cản mới xuất hiện.

Mọi quá trình di chuyển đều được giám sát bởi node Safety Supervisor. Nếu phát hiện người xuất hiện trong vùng nguy hiểm thông qua camera YOLO, robot sẽ dừng lại ngay lập tức và chỉ tiếp tục di chuyển khi vùng nguy hiểm không còn người trong một khoảng thời gian cooldown đảm bảo an toàn. Trong trường hợp phát hiện vật cản bất ngờ trên đường đi (phân tích dữ liệu Lidar, so với vị trí trên quỹ đạo), robot sẽ lùi lại trong thời gian ngắn để tránh va chạm, sau đó dừng lại, xoá đường đi cũ và gửi lại goal cho planner lập lại đường đi mới với bản đồ đã cập nhật vật cản. Trong mọi tình huống, nếu đồng thời phát hiện cả người lẫn vật cản, robot luôn ưu tiên dừng vì sự an toàn của con người. Quá trình bám đường, phát hiện và xử lý vật cản, nhận diện người và lập lại đường đi sẽ lặp lại liên tục cho đến khi robot di chuyển đến vị trí đích với sai số cho phép, lúc này robot sẽ tự động dừng lại.

CHƯƠNG 5: THI CÔNG VÀ KẾT QUẢ THỰC NGHIỆM

5.1 Thiết kế phần cứng hệ thống xe tự hành

Trong đề tài này, nhóm quyết định tiếp cận theo hướng đơn giản hóa mô hình thay vì triển khai theo thiết kế ban đầu với kích thước thực tế. Cụ thể, thay vì phát triển một xe tự hành cỡ lớn có khả năng vận chuyển khối lượng lớn vật tư y tế trong môi trường bệnh viện hoặc cơ sở y tế, nhóm lựa chọn xây dựng một phiên bản thu gọn với kích thước nhỏ hơn, chỉ tích hợp các chức năng cốt lõi như: bám quỹ đạo định sẵn, phát hiện và tránh người di chuyển trong khu vực hoạt động, né tránh vật cản xuất hiện bất ngờ trên đường đi, và tìm kiếm đường đi tối ưu trong môi trường hẹp, nhiều chướng ngại vật.

Cách tiếp cận này cho phép nhóm tập trung toàn lực vào việc nghiên cứu, phát triển và tinh chỉnh các thuật toán điều hướng, xử lý dữ liệu từ cảm biến và điều khiển chuyển động – những yếu tố then chốt quyết định mức độ thông minh, độ ổn định và hiệu quả hoạt động của hệ thống robot tự hành. Việc loại bỏ những yếu tố phức tạp liên quan đến kích thước lớn và khả năng tải trọng cao không chỉ giúp mô hình trở nên gọn nhẹ và dễ kiểm soát hơn mà còn giúp đẩy nhanh quá trình chế tạo, giảm thời gian lắp ráp, dễ dàng thử nghiệm trong không gian hạn chế của phòng thí nghiệm và tiết kiệm đáng kể chi phí phát triển.

Hơn nữa, trong bối cảnh nhóm nghiên cứu đối mặt với những giới hạn về nguồn lực kỹ thuật và tài chính, lựa chọn triển khai mô hình ở quy mô nhỏ là một hướng đi hợp lý và khả thi. Dù kích thước được tối giản, mô hình vẫn thể hiện đầy đủ các nguyên lý hoạt động cơ bản, các khả năng cốt lõi mà một hệ thống AMR (Autonomous Mobile Robot) cần có, đồng thời mở ra khả năng mở rộng và nâng cấp trong tương lai. Đây được xem là bước đi quan trọng mang tính chiến lược, không chỉ giúp giảm thiểu rủi ro trong giai đoạn phát triển ban đầu mà còn tạo nền tảng vững chắc để tiến tới xây dựng các phiên bản robot tự hành hoàn chỉnh hơn trong tương lai

5.1.1 Chọn động cơ cho hệ thống

Nhóm chọn động cơ JGB37-520 vì moment lớn, có encoder và phù hợp dẫn động vi sai khi xe mang tải với các thông số kỹ thuật của động cơ JGB37-520 được trình bày trong.

Hình 5.1: Động Cơ DC giảm tốc JGB37-520 12V

Bảng 5.1: Thông số kỹ thuật động cơ JGB37-520

Điện áp hoạt động	12V
Công suất	15W
Dòng không tải	< 1A
Dòng chịu đựng tối đa khi có tải	2.3A
Tốc độ không tải	110RPM
Tốc độ chịu đựng tối đa khi có tải	85RPM
Moment xoắn định mức của động cơ	10Kg.cm
Lực léo Moment tối đa	15Kg.cm
Chiều dài hộp số L	24mm
Số xung Encoder mỗi kênh trên 1 vòng quay trục chính	11 x 90 = 990 xung

5.1.2 Lựa chọn bánh xe

Nhóm chọn bánh chủ động 65mm và một bánh đa hướng trước**,** các thông số kỹ thuật của bánh xe được thể hiện trong **bảng 5.1,** để đảm bảo cân bằng, dễ điều hướng và phù hợp dẫn động vi sai.

Hình 5.2: Bánh xe 65mm khớp lục giác 12mm và bánh xe đa hướng 38x32x33mm

Bảng 5.2: Thông số kỹ thuật bánh xe

Đường kính bánh cao su	65 mm
Bề rộng bánh cao su	27 mm
Bánh xe đa hướng	38x32x33 mm
Chất liệu	Nhựa, mút, cao su

5.1.3 Lựa chọn nguồn điện

Hệ thống sử dụng pin sạc dự phòng iValue Y1-5 (10.000mAh, sạc nhanh 3A–15W) để cấp nguồn cho Raspberry Pi 4, đảm bảo hoạt động ổn định 2–3 tiếng. Ba cell Li-ion 18650 (3.7 V) mắc nối tiếp được dùng cấp nguồn cho hai động cơ qua driver L298N.

Hình 5.3: Pin dự phòng I.value và khay pin 3 cell 18650

5.1.4 Raspberry Pi 4 Model B

Raspberry Pi 4 được lựa chọn nhờ hiệu năng xử lý mạnh, hỗ trợ tốt cho ROS 2 và các tác vụ AI trong **bảng 5.3**, phù hợp triển khai các thuật toán SLAM, A* và nhận diện hình ảnh trong thời gian thực.

Hình 5.4: Raspberry Pi 4 Model B

Bảng 5.3: Thông số kỹ thuật Raspberry Pi 4

Vi xử lý	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM	4GB LPDDR4-2400 SDRAM
Wi-Fi	IEEE 802.11ac, băng tần 2.4GHz và 5.0GHz
Bluetooth	Bluetooth 5.0, BLE
Cổng mạng	Gigabit Ethernet
USB	2 x USB 3.0, 2 x USB 2.0
GPIO	40 chân, tương thích với các phiên bản trước
Cổng AV	AV 4 chân (âm thanh và video)
Lưu trữ	Khe cắm thẻ Micro-SD
Nguồn cấp	5V – 3A qua cổng USB-C, hoặc 5V qua GPIO header (tối thiểu 3A)

Vai trò của Raspberry Pi 4 Model B:

- Xử lý hình ảnh từ camera để nhận diện vật cản, vạch đường hoặc đối tượng cần tracking bằng AI.
- Chạy mô hình AI (như YOLO, TensorFlow Lite) phục vụ định hướng và ra quyết định.

- Giao tiếp với các vi điều khiển như Arduino để gửi lệnh điều khiển bánh xe, cảm biến.
- Quản lý hệ thống: kết nối Wi-Fi, truyền nhận dữ liệu, lưu trữ log.
- Cài hệ điều hành Ubuntu 22.04 kèm ROS2 Humble

5.1.5 Arduino UNO R3

Arduino Uno được chọn vì dễ lập trình, ổn định và tương thích tốt với encoder, cảm biến và driver động cơ.

Hình 5.5: Arduino UNO R3

Bảng 5.4: Thông số kỹ thuật Arduino Uno R3

Vi điều khiển chính	ATmega328P
Điện áp hoạt động	5V
Điện áp đầu vào (khuyến nghị)	7-12V
Số chân Digital I/O	14 (trong đó 6 chân PWM)
Số chân Analog Input	6
Tần số hoạt động	16 MHz
Bộ nhớ Flash	32 KB (0.5 KB dùng cho bootloader)

Vai trò của Arduino UNO R3:

- Đọc tín hiệu encoder từ động cơ JGB37-520
- Điều khiển động cơ thông qua mạch L298N (bật/tắt, đảo chiều, PWM điều tốc)
- Nhận lệnh điều khiển từ Raspberry Pi 4 và thực thi các tác vụ phần cứng theo thời gian thực

– Xử lý tín hiệu ở tầng thấp, đảm bảo phản hồi nhanh và ổn định.

5.1.6 Mạch điều khiển động cơ L298N

Mạch điều khiển động cơ L298N được chọn, vì có khả năng điều khiển hai động cơ DC độc lập, chịu dòng lớn (tối đa 2A/motor), dễ tích hợp với Arduino và phù hợp cho hệ dẫn động vi sai.

Hình 5.6: Mạch điều khiển động cơ L298N

Bảng 5.5: Thông số kỹ thuật L298N

Module điều khiển	2A L298N
Chip điều khiển	Cặp H-Bridge L298N
Điện áp cấp cho động cơ (Tối đa)	46V
Dòng điện cấp động cơ (tối đa)	2A
Điện áp logic	5V
Điện áp hoạt động của IC	5÷35V
Dòng điện hoạt động IC	2A
Dòng logic	0÷36mA
Công suất tối đa (W)	25W

Vai trò của Mạch điều khiển động cơ L298N:

– Điều khiển tốc độ và chiều quay của động cơ thông qua tín hiệu PWM từ Arduino
– Cấp dòng lớn cho động cơ (lên đến 2A mỗi kênh)
– Bảo vệ mạch điều khiển nhờ tích hợp diode chống nhiễu và IC ổn áp 5V
– Hỗ trợ điều khiển 2 động cơ DC hoặc 1 động cơ bước.

5.1.7 Cảm biến RPLIDAR A1M8

Cảm biến RPLIDAR A1M8 được chọn vì có thông số kỹ thuật phù hợp trong **bảng 5.6** như khả năng quét 360°, tầm quét xa tới 12m, tốc độ quét cao (6÷10Hz), phù hợp cho bản đồ hóa và tránh vật cản trong không gian hẹp của môi trường kho.

Hình 5.7: Cảm biến RPLIDAR A1M8

Bảng 5.6: Thông số kỹ thuật RPLIDAR A1M8

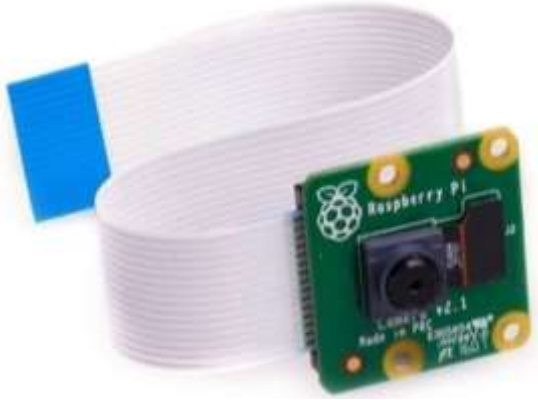
Phương pháp đo khoảng cách	Phép tam giác (Triangulation)
Loại động cơ quay	Động cơ DC chổi than
Điện áp hệ thống	5VDC
Giao tiếp đầu ra	UART Serial (mức điện áp 3.3V)
Phạm vi đo	0.15÷12 mét
Tần số lấy mẫu	8000 mẫu/giây (8KHz)
Tốc độ quay	5.5 vòng/giây (Hz)
Độ phân giải góc	≤ 1°
Góc quét	360°

Vai trò của cảm biến RPLIDAR A1M8:

- Quét và phát hiện vật cản xung quanh xe theo mọi hướng
- Tạo bản đồ môi trường phục vụ định vị và SLAM
- Hỗ trợ hệ thống AI xác định đường đi an toàn và tránh va chạm
- Cung cấp dữ liệu độ sâu chính xác hơn cảm biến siêu âm trong phạm vi rộng

5.1.8 Raspberry Pi Camera Module V2.1 – 8MP

Camera Raspberry Pi V2.1 được lựa chọn nhờ khả năng ghi hình sắc nét, phù hợp cho các tác vụ nhận diện, theo dõi và xử lý hình ảnh trong hệ thống xe tự hành.



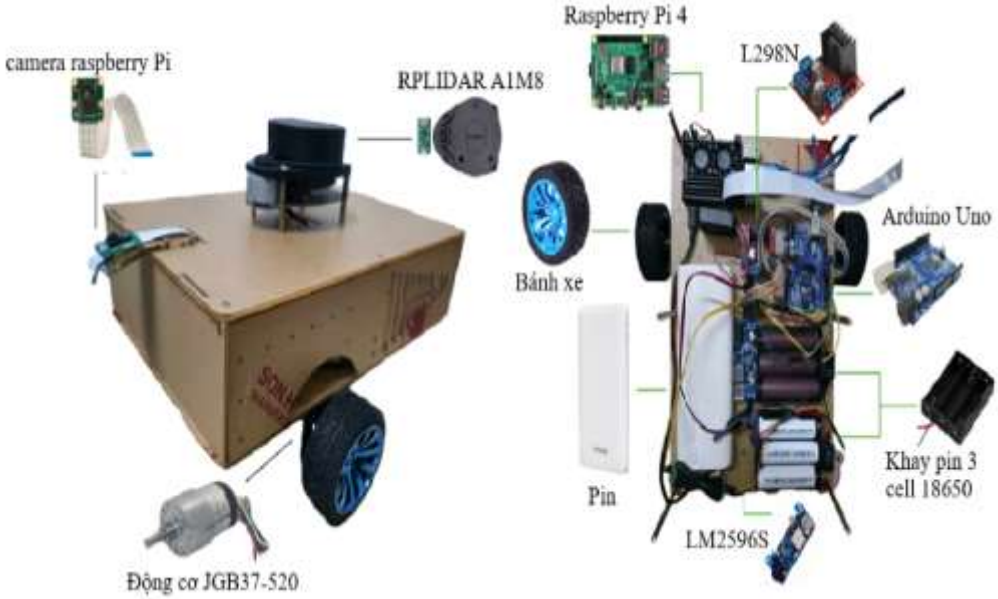
Hình 5.8: Raspberry Pi Camera Module V2.1 – 8MP

Bảng 5.7: Thông số kỹ thuật camera Raspberry Pi

Cảm biến	Sony IMX219
Độ phân giải	8MP (3280 x 2464 pixels)
Hỗ trợ video	1080p30, 720p60, 480p90
Ống kính	Cố định (fixed - focus)
Góc nhìn (FOV)	62.2° ngang, 48.8° dọc
Khoảng lấy nét	~1m
Kết nối	CSI (Camera Serial Interface) bằng cáp ribbon
Nguồn cấp	Qua cổng CSI – không cần nguồn riêng
Hệ điều hành hỗ trợ	Raspberry Pi OS, hỗ trợ libcamera, picamera2, OpenCV

5.1.9 Thiết kế kiến trúc hệ thống điều khiển trong đề tài

Dựa trên định hướng từ thiết kế lý tưởng nêu ở mục **2.4.1** và lựa chọn lại linh kiện. Để minh họa rõ hơn cách bố trí các thành phần trên mô hình xe tự hành được thiết kế trong đề tài, mặt cắt **Hình 5.9** thể hiện vị trí lắp đặt tương đối của các phần tử chính như: Bộ vi điều khiển (Arduino), máy tính nhúng (Raspberry Pi 4), pin, mô-đun cảm biến LiDAR, camera và khung động cơ.



Hình 5.9: Mặt cắt và vị trí linh kiện trong xe

5.2 Chạy thực nghiệm và nhận xét

5.2.1 Môi trường thử nghiệm

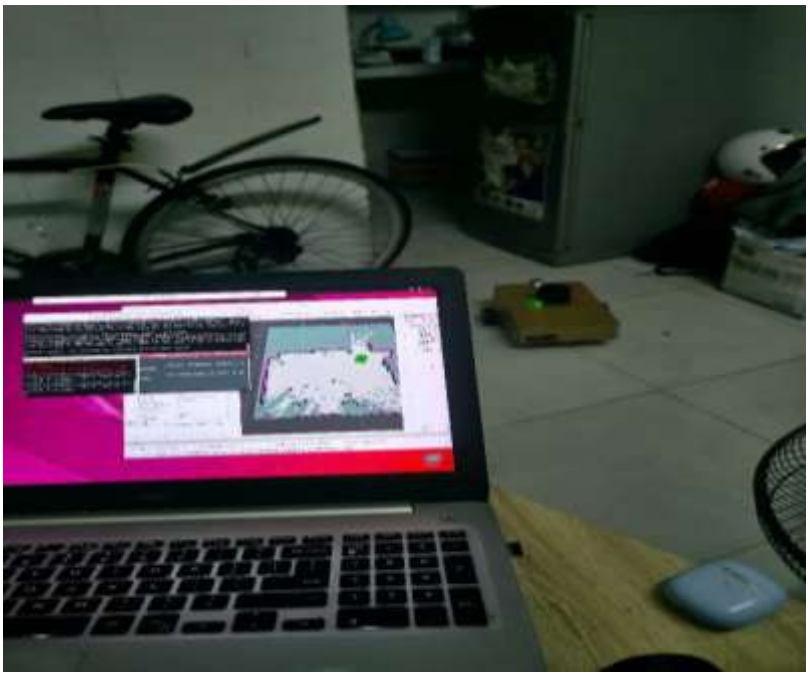
Thử nghiệm được tiến hành trong môi trường nhà kín với diện tích 3.5 x 3.5 m^2 nền lát gạch hoa địa hình di chuyển phù hợp với robot đã thiết kế.

Ảnh chụp căn phòng thực tế dung để thử nghiệm robot:

Hình 5.10: Không gian thực tế thử nghiệm robot

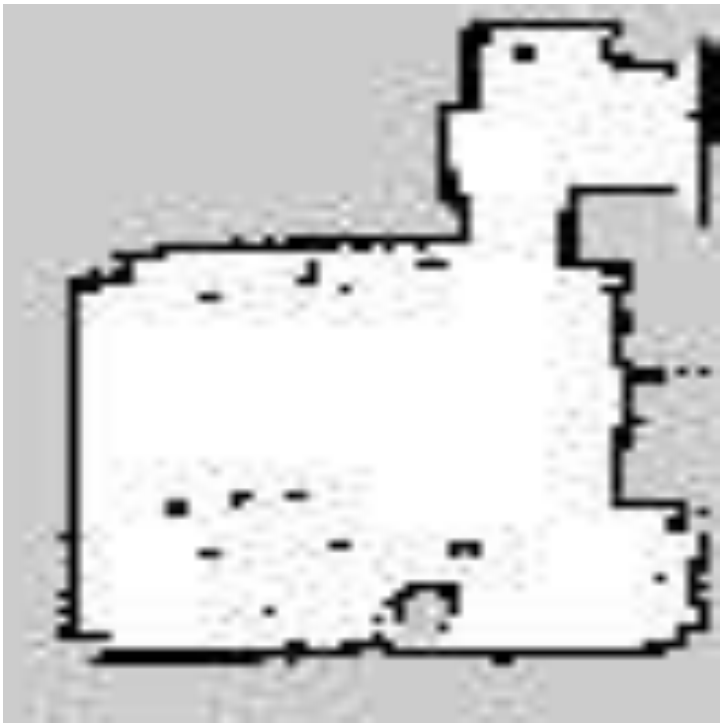
5.2.2 Quá trình quét bản đồ

Dưới đây là hình ảnh minh họa quá trình tạo bản đồ môi trường bằng cảm biến Lidar, giúp hệ thống xây dựng bản đồ và định vị trong không gian làm việc thực tế.



Hình 5.11: Quá trình tạo bản đồ

Dữ liệu bản đồ được tạo sau khi quét hoàn tất.



Hình 5.12: Bản đồ 2D thu được sau khi quét môi trường

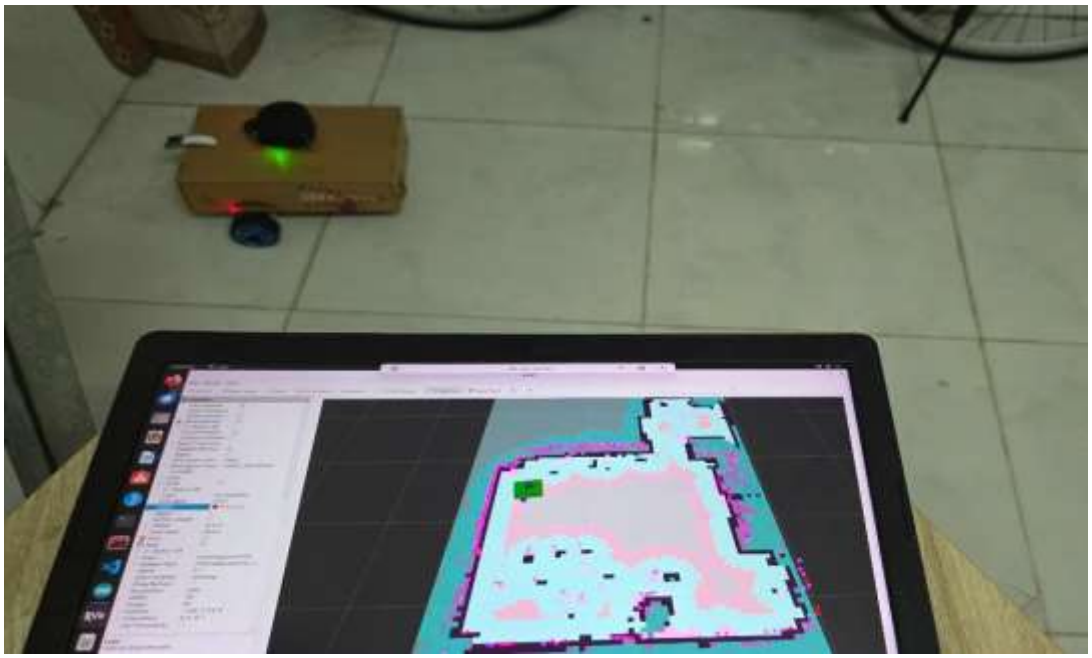
Nhận xét:

Bản đồ quét Lidar thể hiện rõ ràng cấu trúc môi trường và các vật cản lớn, tuy nhiên vẫn tồn tại hạn chế trong việc phát hiện các vật thể nhỏ, mảnh hoặc nằm ngoài phạm vi quét (như dưới tầm quét). Điều này có thể ảnh hưởng đến độ tin cậy khi di chuyển trong môi trường phức tạp.

5.2.3 Quá trình điều hướng

5.2.3.1 Điều khiển bám đường đi

Vị trí bắt đầu của xe: Xe khởi động tại góc trên bên trái bản đồ, đây là điểm xuất phát để thực hiện quá trình SLAM và điều hướng.



Hình 5.13: Vị trí bắt đầu của xe

Vị trí khi xe bám được nửa đường: Xe di chuyển đến khoảng giữa lộ trình dự kiến, thể hiện qua việc quỹ đạo di chuyển ổn định và hệ thống giữ được hướng đi chính xác, cho thấy quá trình điều hướng và bám đường diễn ra hiệu quả.

Hình 5.14: Vị trí xe khi di chuyển được 1 nửa lộ trình

Vị trí kết thúc của xe: Xe đã di chuyển đến đúng vị trí đích theo kế hoạch ban đầu, dừng lại ổn định với sai số nhỏ so với quỹ đạo mong muốn. Điều này chứng tỏ hệ thống điều khiển hoạt động hiệu quả, đảm bảo bám đường và định vị chính xác trong suốt quá trình di chuyển.

Hình 5.15: Xe tới vị trí chỉ định

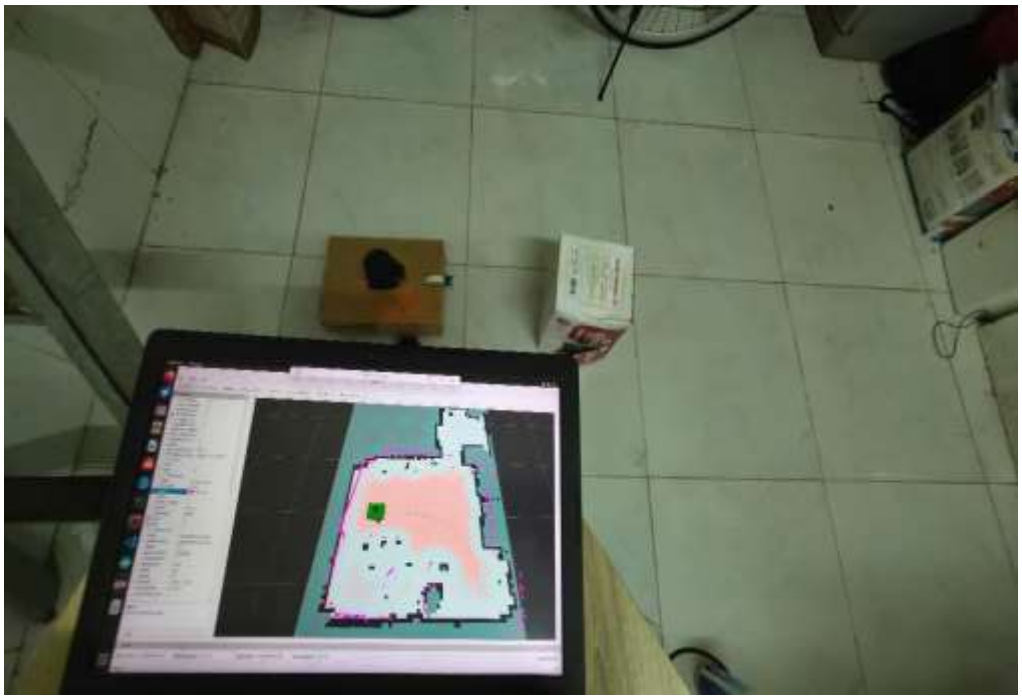
Nhận xét:

Xe tự động chạy đến điểm đích được chỉ định trên bản đồ hiển thị trên màn hình máy tính, khá mượt và chính xác, khi điều hướng qua những chỗ không trong tầm quét của Lidar thì xe đôi lúc có thể bị thất bại vì vướng bánh xe, dẫn tới lệch bản đồ.

5.2.3.2 Điều khiển né vật cản và nhận diện người

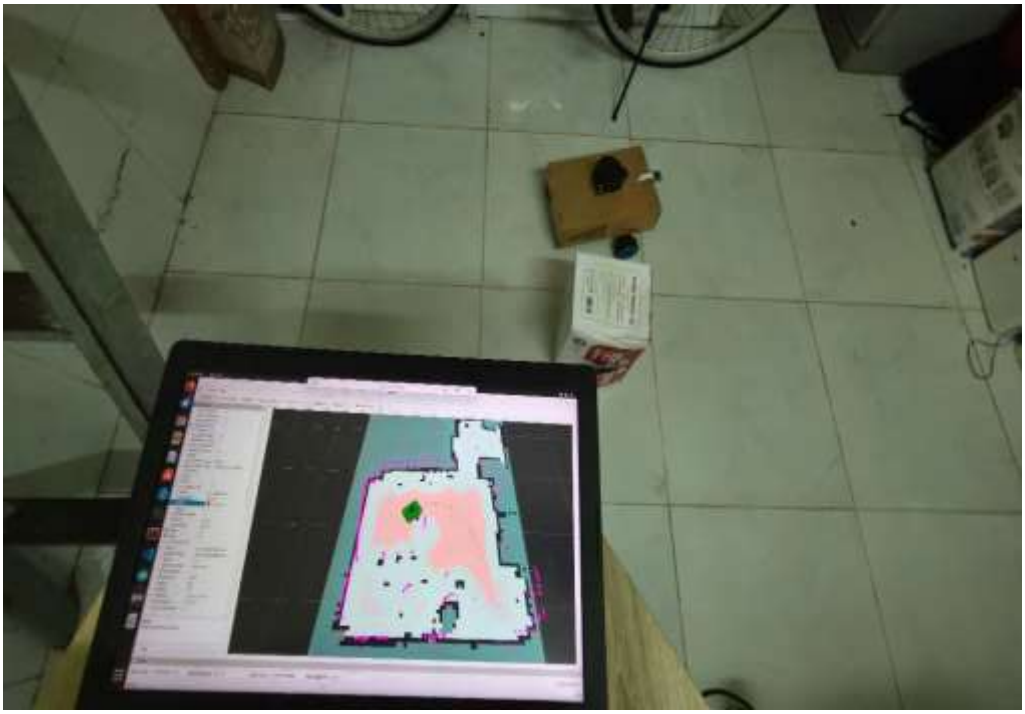
a. Xử lý vật cản trên đường tối ưu

Xe bắt đầu di chuyển theo đường đi tối ưu. Khi xuất hiện vật cản bất ngờ, hệ thống dừng để đảm bảo hành trình an toàn.



Hình 5.16: Xe dừng khi xuất hiện vật cản bất ngờ

Khi phát hiện vật cản, hệ thống lập tức nhận diện và tính toán lại đường đi mới để tiếp tục di chuyển an toàn.



Hình 5.17: Xe đã tái lập kế hoạch thành công và tiếp tục di chuyển theo đường đi tối ưu mới

Xe đã hoàn thành đường đi tối ưu mới và tới vị trí đích đã chỉ định.

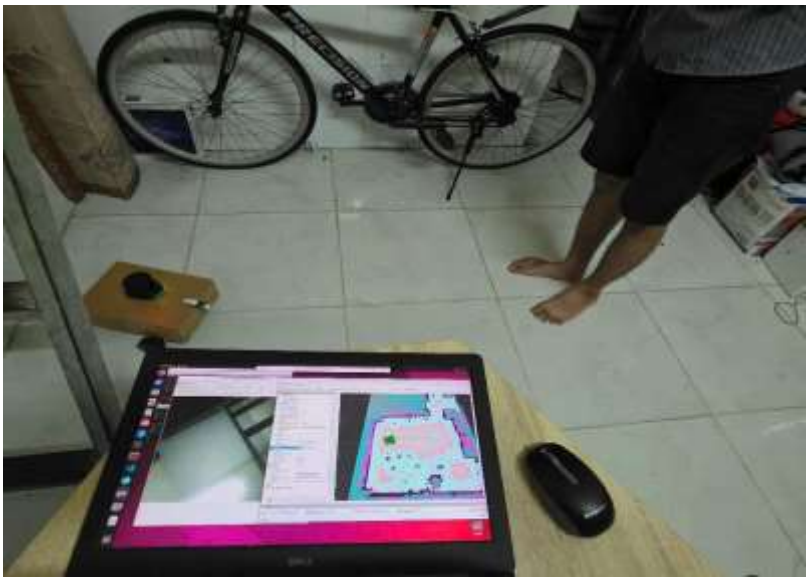
Hình 5.18: Xe đã hoàn thành lộ trình và dừng tại vị trí đích như mong muốn

Nhận xét:

Xe có khả năng nhận diện và phản ứng với vật cản động, đồng thời tái tạo lại đường đi tối ưu để tiếp tục đến vị trí đích hiệu quả. Hệ thống xử lý ổn định với các vật cản nằm trong vùng quét của Lidar, nhưng vẫn còn hạn chế với vật nhỏ hoặc có tiết diện hẹp, dễ bị Lidar bỏ sót.

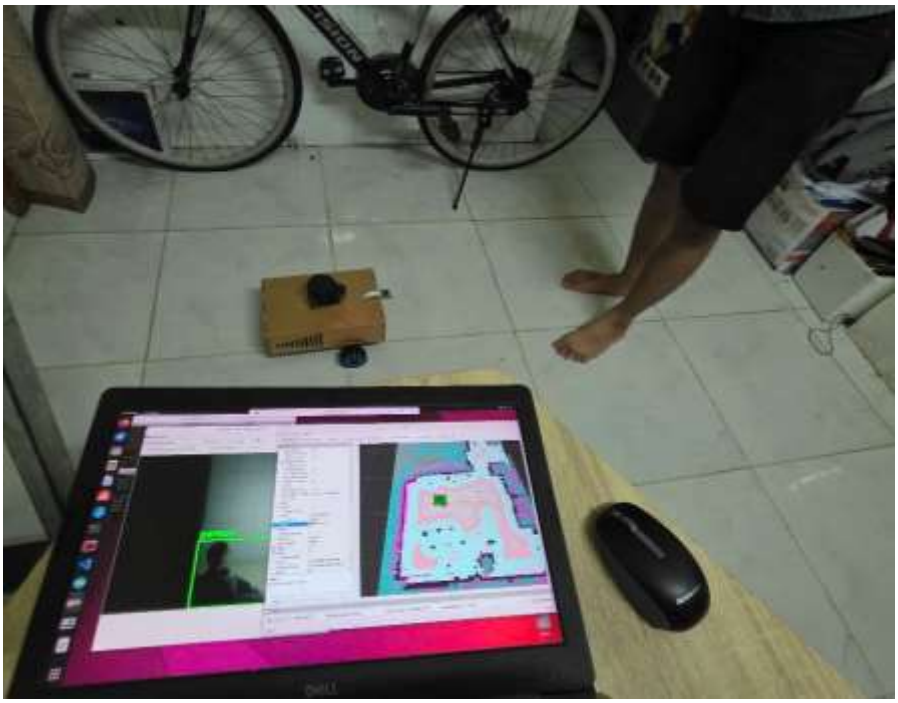
b. Quá trình nhận diện người và điều chỉnh hướng di chuyển

Khi hệ thống chưa phát hiện người hay vật cản, xe sẽ tự động di chuyển theo đường đi tối ưu đã tính toán trước đó trên bản đồ.



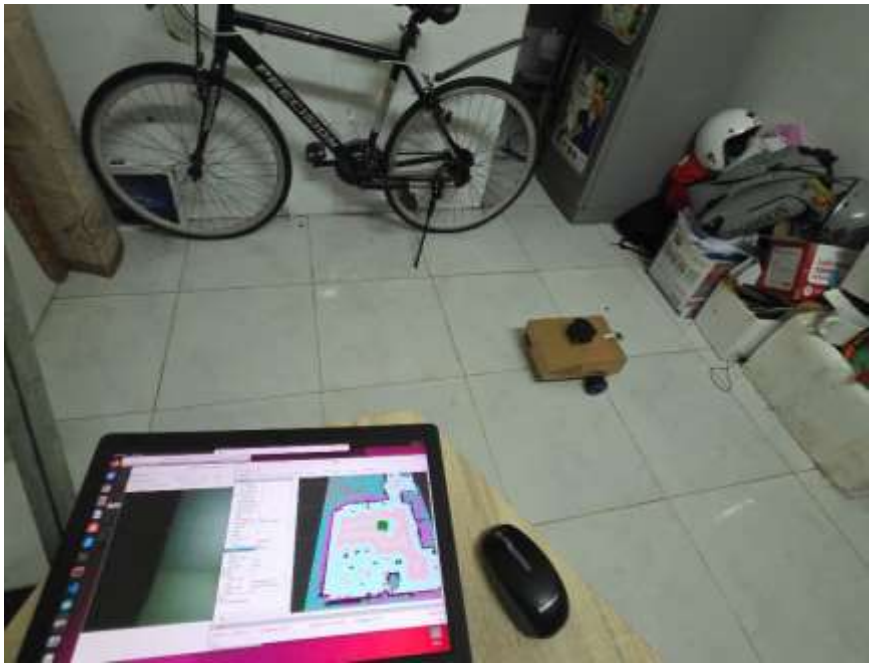
Hình 5.19: Xe di chuyển theo đường đi tối ưu khi chưa phát hiện người

Khi đang di chuyển, hệ thống phát hiện người xuất hiện trong vùng quét, xe lập tức dừng lại để đảm bảo an toàn.



Hình 5.20: Xe dừng lại khi nhận diện được người

Sau khi người rời khỏi vùng nhận diện, xe tiếp tục tính toán và di chuyển theo lộ trình tối ưu đã được thiết lập trước đó để hoàn thành nhiệm vụ.



Hình 5.21: Xe hoàn thành lộ trình khi người đã di chuyển ra khỏi vùng quét

Nhận xét:

Thử nghiệm thực tế cho thấy YOLOv8 có khả năng nhận diện người với độ chính xác cao ngay cả trong điều kiện ánh sáng phòng bình thường và từ nhiều góc độ khác nhau. Tuy nhiên, hệ thống vẫn gặp hạn chế về tốc độ xử lý, gây ra một độ trễ nhất định trong phản ứng. Dù vậy, sau khi không còn phát hiện người, xe vẫn xử lý tình huống an toàn và tiếp tục di chuyển ổn định đến đích, đảm bảo tính tin cậy trong quá trình vận hành tự động.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

6.1.1 Ưu điểm của nghiên cứu

Nghiên cứu này mang lại một số ưu điểm nổi bật, góp phần nâng cao hiệu quả và tính khả thi khi triển khai hệ thống AMR trong thực tế, cụ thể gồm:

- Xe có khả năng tự tìm đường đi tối ưu nhờ thuật toán A* kết hợp SLAM.
- Tích hợp camera nhận diện người, hỗ trợ Lidar phát hiện và tránh vật cản động hiệu quả.
- Bám đường tốt nhờ bộ điều khiển backstepping xử lý chính xác đặc tính phi tuyến của xe.

6.1.2 Nhược điểm của nghiên cứu

Bên cạnh những ưu điểm, nghiên cứu vẫn tồn tại một số hạn chế cần được cải thiện trong các giai đoạn tiếp theo, bao gồm:

- Độ trễ nhận diện người còn cao do giới hạn phần cứng camera và hiệu suất Raspberry Pi
- Điều khiển phụ thuộc Wifi, gây khó khăn ở nơi kết nối yếu hoặc không ổn định
- Chưa có cảm biến ở các góc khuất của Lidar, hạn chế khả năng xử lý linh hoạt ở nơi ngoài tầm quét của Lidar.
- Tốc độ di chuyển chưa cao, chưa đáp ứng yêu cầu tải trọng.
- Hệ thống chưa có khả năng kéo tải thực tế nặng.
- Chưa thích nghi tốt với với nhiều loại môi trường.

6.2 Hướng phát triển tương lai

Nếu có thêm thời gian và điều kiện, nhóm định hướng phát triển hệ thống theo các hướng sau:

- Cải thiện độ chính xác trong định vị và điều khiển.
- Tối ưu hiệu suất, giảm độ trễ xử lý của camera và mô hình AI.
- Huấn luyện mô hình AI nâng cao, hướng đến khả năng chạy ở tốc độ cao và phản ứng chính xác hơn.

- Củng cố cơ khí, gia cường khung gầm để xe có thể kéo tải nặng.
- Phát triển giao diện giao tiếp trực quan (màn hình cảm ứng, giọng nói tổng hợp, hoặc tín hiệu đèn) giúp xe tương tác linh hoạt hơn với nhân viên y tế.
- Xây dựng hệ thống nhiều xe hoạt động song song với khả năng chia sẻ bản đồ, tránh nhau và phối hợp luân phiên thực hiện nhiệm vụ để tối ưu hóa luồng vận chuyển trong bệnh viện.
- Phát triển khả năng kết nối xe với phần mềm quản lý bệnh viện để nhận lịch vận chuyển, theo dõi trạng thái đơn hàng và cập nhật tiến độ theo thời gian thực.
- Gắn thêm các cảm biến siêu âm hoặc Time-of-Flight ở những vị trí góc khuất mà LiDAR chưa bao phủ được, giúp nâng cao khả năng tránh va chạm toàn diện, đặc biệt trong khu vực có nhiều thiết bị y tế sát sàn nhà.

KẾT LUẬN

Trong khuôn khổ đồ án tốt nghiệp, nhóm đã nghiên cứu và triển khai thành công một hệ thống xe tự hành tích hợp nhiều công nghệ hiện đại như trí tuệ nhân tạo (AI), thuật toán điều khiển phi tuyến (Backstepping), lập kế hoạch đường đi A*, định vị AMCL, xây dựng bản đồ bằng Graph SLAM, và điều hướng tránh vật cản trong môi trường thực tế.

Hệ thống được phát triển trên nền tảng ROS 2 Humble với kiến trúc phân tầng, cho phép xử lý ảnh thời gian thực bằng mô hình YOLOv8, điều khiển động cơ thông qua bộ điều khiển PID trên vi điều khiển Arduino. Nhờ đó, xe có khả năng tự động lập bản đồ, định vị, nhận diện con người, tránh vật cản, và bám theo lộ trình tối ưu.

Thông qua quá trình triển khai, nhóm không chỉ củng cố kiến thức chuyên môn mà còn phát triển tư duy thiết kế hệ thống và kỹ năng giải quyết các vấn đề thực tiễn. Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, đáp ứng tốt yêu cầu điều hướng trong không gian có người và vật cản bất ngờ.

Hệ thống xe tự hành này sở hữu tiềm năng ứng dụng cao trong thực tế, đặc biệt tại các bệnh viện hoặc phòng khám thông minh – nơi có nhu cầu vận chuyển hàng hóa, vật tư y tế hoặc thuốc men một cách chính xác, an toàn và hạn chế tiếp xúc. Giải pháp này không chỉ giúp giảm phụ thuộc vào nhân lực mà còn nâng cao độ chính xác và an toàn trong vận hành.

Tuy vẫn còn một số hạn chế nhất định do giới hạn về phần cứng và thời gian triển khai, đồ án đã đặt nền móng vững chắc cho việc tiếp tục cải tiến, mở rộng quy mô và hướng tới ứng dụng thực tiễn hoặc thương mại hóa trong tương lai gần.

TÀI LIỆU THAM KHẢO

[1] "AGV Navigation: Methods, Comparison, Pros and Cons - Illustrated Guide." Accessed: Jun. 17, 2025. [Online]. Available: https://www.agvnetwork.com/types-of-navigation-systems-automated-guided-vehicles

[2] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, May 2006, doi: 10.1016/j.ejor.2004.09.020.

[3] D. Bechtsis, N. Tsolakis, D. Vlachos, and J. S. Srai, "Intelligent Autonomous Vehicles in digital supply chains: A framework for integrating innovations towards sustainable value networks," *Journal of Cleaner Production*, vol. 181, pp. 60–71, Apr. 2018, doi: 10.1016/j.jclepro.2018.01.173.

[4] G. Fragapane, H.-H. Hvolby, F. Sgarbossa, and J. O. Strandhagen, "Autonomous Mobile Robots in Hospital Logistics," in *Advances in Production Management Systems. The Path to Digital Transformation and Innovation of Production Management Systems*, vol. 591, B. Lalic, V. Majstorovic, U. Marjanovic, G. Von Cieminski, and D. Romero, Eds., in IFIP Advances in Information and Communication Technology, vol. 591. , Cham: Springer International Publishing, 2020, pp. 672–679. doi: 10.1007/978-3-030-57993-7_76.

[5] R. Chand, B. Sharma, and S. A. Kumar, "Systematic review of mobile robots applications in smart cities with future directions," *Journal of Industrial Information Integration*, vol. 45, p. 100821, May 2025, doi: 10.1016/j.jii.2025.100821.

[6] L. Marques, A. Martins, and A. T. De Almeida, "Environmental monitoring with mobile robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alta., Canada: IEEE, 2005, pp. 3624–3629. doi: 10.1109/IROS.2005.1545133.

[7] Q. Chen, H. Cheng, R. Huang, J. Qiu, and X. Chen, "Learning and planning of stair ascent for lower-limb exoskeleton systems," *IR*, vol. 46, no. 3, pp. 421–430, May 2019, doi: 10.1108/IR-03-2018-0054.

[8] A. Goutam Mukherjee *et al.*, "A review on modern and smart technologies for efficient waste disposal and management," *Journal of Environmental Management*, vol. 297, p. 113347, Nov. 2021, doi: 10.1016/j.jenvman.2021.113347.

[9] "Việt Nam's first autonomous vehicle debuts," vietnamnews.vn. Accessed: Jun. 17, 2025. [Online]. Available: https://vietnamnews.vn/economy/914897/viet-nam-s-first-autonomous-vehicle-debuts.html

[10] "The Automotive Revolution: Why Vietnam is a Rising Hub for AI-First Software-Defined Vehicles (SDVs)." Accessed: Jun. 17, 2025. [Online]. Available: https://fptsoftware.com/resource-center/blogs/the-automotive-revolution-why-vietnam-is-a-rising-hub-for-ai-first-software-defined-vehicles

[11] M. Crenganis, C. Biris, and C. Girjob, "Mechatronic Design of a Four-Wheel drive mobile robot and differential steering," *MATEC Web Conf.*, vol. 343, p. 08003, 2021, doi: 10.1051/matecconf/202134308003.

[12] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st ed. Cambridge University Press, 2017. doi: 10.1017/9781316661239.

[13] B. D. Hirpo and W. Zhongmin, "Design and Control for Differential Drive Mobile Robot," *International Journal of Engineering Research*, vol. 6, no. 10, 2017.

[14] M. Shin, "Chuyển Động Song Phẳng Của Vật Rắn," Bộ môn Kỹ thuật Cơ khí - HUMG. Accessed: Jun. 17, 2025. [Online]. Available: https://ktck-humg.com/chuyen-dong-song-phang-cua-vat-ran/

[15] S. YiGiT and A. SezgiN, "Trajectory Tracking via Backstepping Controller with PID or SMC for Mobile Robots," *Sakarya University Journal of Science*, vol. 27, no. 1, pp. 120–134, Feb. 2023, doi: 10.16984/saufenbilder.1148158.

[16] J. Wiley, "Mark W. Spong, Seth Hutchinson, and M. Vidyasagar".

[17] R. D. Ahmad Abu Hatab, "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework," *Adv Robot Autom*, vol. 02, no. 02, 2013, doi: 10.4172/2168-9695.1000107.

[18] M. Kalyoncu and F. DemiRbaş, "DIFFERENTIAL DRIVE MOBILE ROBOT TRAJECTORY TRACKING WITH USING PID AND KINEMATIC BASED BACKSTEPPING CONTROLLER," *Scitech*, vol. 5, no. 1, pp. 1–15, Mar. 2017, doi: 10.15317/Scitech.2017.65.

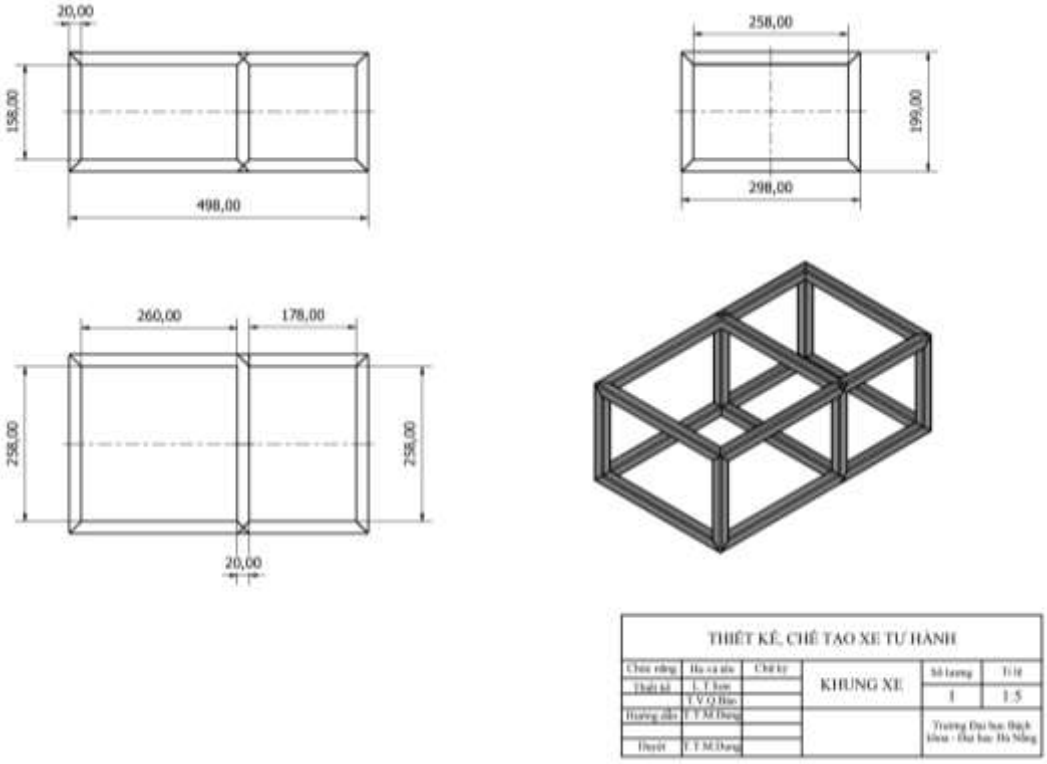
[19] F. Sarrocco, "Graph SLAM: From Theory to Implementation," Federico Sarrocco. Accessed: Jun. 17, 2025. [Online]. Available: https://federicosarrocco.com/blog/graph-slam-tutorial

[20] T. Lauttia, "Adaptive Monte Carlo Localization in ROS".

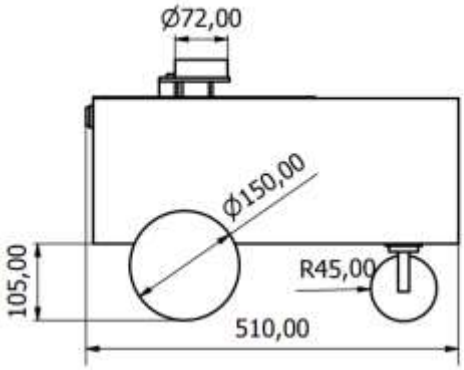
[21] S. Thrun, W. Burgard, and D. Fox, "PROBABILISTIC ROBOTICS".

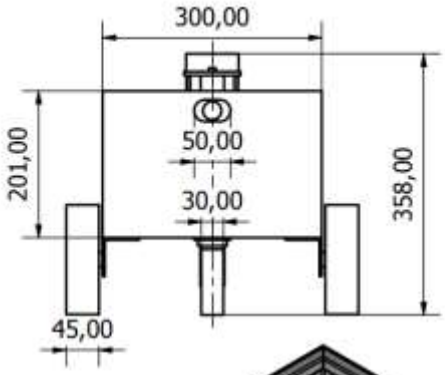
PHỤ LỤC

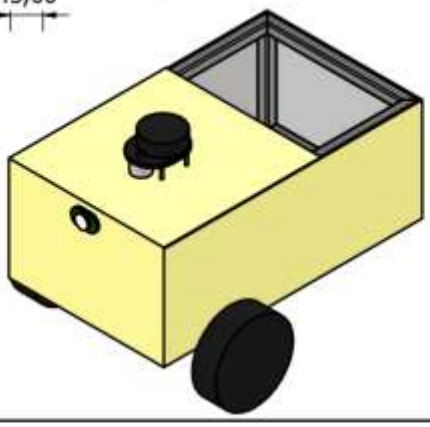
PHỤ LỤC 1

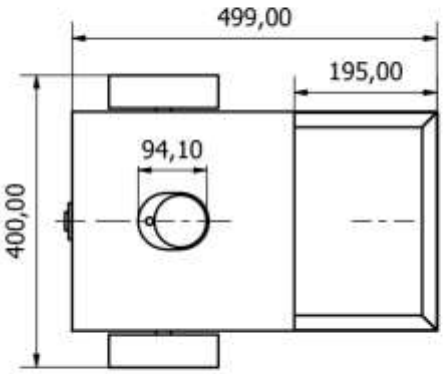


PHỤ LỤC 2:









THIẾT KẾ, CHẾ TẠO XE TỰ HÀNH					
Chức năng	Họ và tên	Chữ ký	XE TỰ HÀNH	Số lượng	
Thiết kế	L.T.Sơn			1	
	T.V.Q.Bảo				
Hướng dẫn	T.T.M.Dung			Trường Đại học	
				khoa - Đại học Đ	
Duyệt	T.T.M.Dung				