

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

DƯƠNG NGỌC HOÀN
NGUYỄN VÕ HỒNG MỸ HIỀN

ĐỒ ÁN TỐT NGHIỆP
XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN
ĐỘNG 4 BÁNH TOÀN THỜI GIAN

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

TP. HỒ CHÍ MINH, 2024

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

DƯƠNG NGỌC HOÀN - 2010020

NGUYỄN VŨ HỒNG MỸ HIỀN - 2010260

ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN
ĐỘNG 4 BÁNH TOÀN THỜI GIAN

BUILD AND CONTROL A FULL-TIME FOUR WHEEL DRIVE
ELECTRIC VEHICLE MODEL

CỬ NHÂN NGÀNH KỸ THUẬT ĐIỀU KHIỂN & TỰ ĐỘNG HÓA

GIẢNG VIÊN HƯỚNG DẪN
TS. NGUYỄN VĨNH HẢO

TP. HỒ CHÍ MINH, 2024

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI
TRƯỜNG ĐẠI HỌC BÁCH KHOA -ĐHQG -HCM**

Cán bộ hướng dẫn Khóa luận tốt nghiệp :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 1 :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 2 :
(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Khóa luận tốt nghiệp được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG
Tp.HCM ngày tháng năm

Thành phần Hội đồng đánh giá khóa luận tốt nghiệp gồm:
(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ khóa luận tốt
nghiệp)

1.
2.
3.
4.
5.

Xác nhận của Chủ tịch Hội đồng đánh giá khóa luận tốt nghiệp và Chủ nhiệm
Bộ môn sau khi luận văn đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG

CHỦ NHIỆM BỘ MÔN.....

TP. HCM, ngày....tháng.....năm.....

**NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
CỦA CÁN BỘ HƯỚNG DẪN**

Tên luận văn:

**XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN ĐỘNG 4 BÁNH
TOÀN THỜI GIAN**

Nhóm Sinh viên thực hiện:

Dương Ngọc Hoàn
Nguyễn Võ Hồng Mỹ Hiền

Cán bộ hướng dẫn:

2010020 TS. Nguyễn Vĩnh Hảo
2010260 TS. Nguyễn Vĩnh Hảo

Đánh giá Đồ án

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....
.....

2. Về nội dung đồ án:

.....
.....
.....
.....
.....

3. Về tính ứng dụng:

.....
.....
.....

4. Về thái độ làm việc của sinh viên:

Đánh giá chung: Đồ án đạt/không đạt yêu cầu của một đồ án tốt nghiệp cử nhân, xếp loại
Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Dương Ngọc Hoàn/10

Nguyễn Võ Hồng Mỹ Hiền :...../10

Cán bộ hướng dẫn

(Ký tên và ghi rõ họ tên)

TP. HCM, ngày....tháng.....năm.....

**NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
CỦA CÁN BỘ PHẢN BIỆN**

Tên luận văn:

**XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN ĐỘNG 4 BÁNH
TOÀN THỜI GIAN**

Nhóm Sinh viên thực hiện:

Dương Ngọc Hoàn
Nguyễn Võ Hồng Mỹ Hiền

Cán bộ phản biện:

2010020 TS. Nguyễn Vĩnh Hảo
2010260 TS. Nguyễn Vĩnh Hảo

Đánh giá Đồ án

5. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

.....
.....
.....
.....
.....

6. Về nội dung đồ án:

.....
.....
.....
.....
.....

7. Về tính ứng dụng:

.....
.....
.....

8. Về thái độ làm việc của sinh viên:

Đánh giá chung: Đồ án đạt/không đạt yêu cầu của một đồ án tốt nghiệp cử nhân, xếp loại
Giỏi/ Khá/ Trung bình

Điểm từng sinh viên:

Dương Ngọc Hoàn/10

Nguyễn Võ Hồng Mỹ Hiền :...../10

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trong quá trình nghiên cứu và thực hiện đề tài tại Trường Đại học Bách Khoa – Đại học Quốc gia Hồ Chí Minh, bằng sự biết ơn và kính trọng, nhóm chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô Khoa Điện – Điện Tử đã nhiệt tình hướng dẫn, giảng dạy và tạo mọi điều kiện thuận lợi giúp đỡ em trong suốt quá trình học tập, nghiên cứu và hoàn thiện đề tài này.

Chúng em cũng xin bày tỏ lòng biết ơn sâu sắc đến thầy Nguyễn Vĩnh Hảo – Trưởng bộ môn Điều khiển Tự động, người đã trực tiếp hướng dẫn, dành thời gian cho chúng em với những lời khuyên, định hướng và hướng dẫn trong quá trình nghiên cứu đề tài. Không những hỗ trợ về kiến thức, thầy còn tiếp thêm tinh thần, hỗ trợ về không gian phòng Lab, giúp chúng em vượt qua những khó khăn trong quá trình thực hiện. Nếu như không có sự giúp đỡ của thầy, nhóm đã không thể có được kết quả nghiên cứu như ngày hôm nay. Có được sự dẫn dắt của thầy là một may mắn rất lớn trên con đường học tập, đồng thời là một bàn đạp tốt trên con đường phát triển sự nghiệp của chúng em sau này.

Chân thành cảm ơn các anh/bạn làm việc tại Phòng 207B3 - phòng thí nghiệm Điều Khiển Tự Động – Bộ môn Tự Động, Phòng 301B1 đã đồng hành và tạo điều kiện về không gian nghiên cứu để nhóm chúng em có thể hoàn thành tốt đề tài.

Cuối cùng xin dành lời cảm ơn đến gia đình và người thân đã chăm sóc, ủng hộ động viên trong suốt thời gian chúng em học tập và nghiên cứu, tạo điều kiện cho chúng em có một tương lai tươi sáng phía trước.

Tuy nhiên điều kiện về năng lực bản thân còn hạn chế, việc thực hiện đề tài chắc chắn không tránh khỏi những thiếu sót. Kính mong nhận được sự đóng góp ý kiến của các thầy cô giáo, bạn bè để bài nghiên cứu của chúng em được hoàn thiện hơn.

Chúng em xin trân trọng cảm ơn!

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN:

XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN ĐỘNG 4 BÁNH TOÀN THỜI GIAN

Cán bộ hướng dẫn: TS. Nguyễn Vĩnh Hảo

Thời gian thực hiện: Từ ngày 30/11/2023 đến ngày 1/5/2024

Sinh viên thực hiện:

Dương Ngọc Hoàn - 2010020

Nguyễn Võ Hồng Mỹ Hiền - 2010260

Nội dung đề tài:

Mục tiêu:

Xây dựng mô hình robot thực tế di chuyển trong nhà, sử dụng các cảm biến Lidar, IMU và Encoder để bám quỹ đạo đã hoạch định trước.

Phương pháp thực hiện:

- Thiết kế phần cứng, xây dựng mô hình robot truyền động 4 bánh độc lập.
- Nghiên cứu, xây dựng các thuật toán và bộ điều khiển để thực thi robot bám quỹ đạo.
- Xây dựng giao tiếp giữa các module trong mô hình.

Kết quả mong đợi:

- Xây dựng thành công mô hình robot truyền động 4 bánh độc lập và dễ dàng cải tiến (nếu cần thiết).
- Thực hiện thành công điều khiển mô hình bám quỹ đạo cho trước một cách

chính xác với mong muốn giảm thiểu sai số bám quỹ đạo, đồng thời tối ưu tốc độ xử lý.

- Kết quả của bộ điều khiển vận tốc và bộ điều khiển góc có sai số không quá 1 cm/s và 1° đồng thời độ vọt lỗ dưới 10% và thời gian xác lập nhỏ hơn 3s.
- Kết quả của thuật toán bám quỹ đạo có sai số khoảng cách không quá 10 cm.

Kế hoạch thực hiện:

Dương Ngọc Hoàn	Nguyễn Võ Hồng Mỹ Hiền
<p>Giai đoạn 1: Tìm hiểu đề tài và cấu trúc tổng thể của robot</p> <ul style="list-style-type: none"> - Lên ý tưởng hình dạng, kích thước của robot và mô phỏng trên SolidWorks - Cài hệ điều hành, setup cho máy tính nhúng Jetson - Tìm hiểu ROS2, giao thức uROS để giao tiếp giữa máy tính nhúng và vi điều khiển <p>Giai đoạn 2: Thiết kế mô hình robot thực tế</p> <ul style="list-style-type: none"> - Hàn mạch và thực hiện lắp ráp robot <p>Giai đoạn 3: Tìm hiểu các thuật toán bám quỹ đạo</p> <ul style="list-style-type: none"> - Tìm hiểu các thuật toán bám quỹ đạo phổ biến hiện nay: Pure Pursuit, Stanley Steering Controller, Model Predictive Control-MPC 	<p>Giai đoạn 1: Tìm hiểu đề tài và cấu trúc tổng thể của robot</p> <ul style="list-style-type: none"> - Tìm hiểu thông tin và cách sử dụng từng module: IMU, Motor, Encoder, Raspberry Pico, Driver motor - Tìm hiểu cách kết nối toàn bộ hệ thống phần cứng và tính toán vấn đề về công suất <p>Giai đoạn 2: Thiết kế mô hình robot thực tế</p> <ul style="list-style-type: none"> - Thiết kế mạch Driver motor điều khiển 4 động cơ 4 DC - Thiết kế mạch điều khiển chính để kết nối vi điều khiển với các sensor - Hàn mạch và test từng module phần cứng <p>Giai đoạn 3: Lập trình Firmware cho vi điều khiển</p> <ul style="list-style-type: none"> - Đọc và xử lý tín hiệu cho từng

<ul style="list-style-type: none"> - Chọn sử dụng thuật toán Stanley Steering vì: chi phí thấp, dễ thực hiện, độ chính xác ổn định, chấp nhận được, tốc độ xử lí nhanh <p>Giai đoạn 4: Xác định vị trí robot trong không gian và thực hiện truyền thông dữ liệu</p> <ul style="list-style-type: none"> - Tìm hiểu thuật toán Iterative Closets Point (ICP) - Sử dụng Lidar A1 để xác định vị trí robot trong không gian 2D - Thực hiện truyền dữ liệu giữa các node điều khiển trong môi trường ROS2 <p>Giai đoạn 5: Quan sát kết quả thực nghiệm, thay đổi các thông số và đánh giá thuật toán.</p> <ul style="list-style-type: none"> - Tiến hành chạy thực nghiệm, thu thập số liệu để kiểm tra và đánh giá thuật toán - Điều chỉnh thông số của thuật toán cho phù hợp với mô hình - Tiến hành viết báo cáo 	<ul style="list-style-type: none"> cảm biến IMU, Encoder <ul style="list-style-type: none"> - Viết chương trình thực hiện giải thuật điều khiển cho từng khối: quỹ đạo tham chiếu, điều khiển vận tốc PID và điều khiển góc PD Fuzzy <p>Giai đoạn 4: Thực nghiệm chạy mô hình và tinh chỉnh tham số cho các bộ điều khiển PID, PD Fuzzy</p> <ul style="list-style-type: none"> - Lựa chọn các bộ số K_p, K_I, K_D cho từng động cơ - Lựa chọn các hệ số K_e, $K_{\dot{e}}$, K_{ω} và các giá trị ngôn ngữ của tập mờ <p>Giai đoạn 5: Quan sát kết quả thực nghiệm, thay đổi các thông số và đánh giá thuật toán.</p> <ul style="list-style-type: none"> - Viết chương trình Matlab xử lý và phân tích dữ liệu thu thập. Vẽ đồ thị và đánh giá sai số - Tiến hành chạy thực nghiệm, thu thập số liệu để kiểm tra và đánh giá thuật toán - Tiến hành viết báo cáo
<p>Xác nhận của Cán bộ hướng dẫn (Ký tên và ghi rõ họ tên)</p>	<p>TP. HCM, ngàythángnăm.....</p> <p>Sinh viên (Ký tên và ghi rõ họ tên)</p>

DANH SÁCH HỘI ĐỒNG BẢO VỆ ĐỒ ÁN

Hội đồng chấm đồ án tốt nghiệp, thành lập theo Quyết định số ngày của Hiệu trưởng Trường Đại học Bách khoa TP.HCM.

1. – Chủ tịch.
2. – Thư ký.
3. – Ủy viên.
4. – Ủy viên.
5. – Ủy viên.

MỤC LỤC

TÓM TẮT ĐỒ ÁN.....	1
Chương 1. TỔNG QUAN VỀ ĐỀ TÀI.....	3
1.1. Lý do chọn đề tài.....	3
1.2. Nội dung và mục tiêu của đề tài.....	4
1.2.1. Nội dung đề tài.....	4
1.2.2. Mục tiêu đề tài.....	4
1.3. Cấu trúc của đề tài.....	4
Chương 2. CƠ SỞ LÝ THUYẾT.....	5
2.1. Thuật toán ICP xây dựng bản đồ 2D	5
2.1.1. Thuật toán ICP.....	6
2.1.2. Thuật toán PL-ICP	7
2.2. Thuật toán Stanley.....	9
2.3. Bộ điều khiển PID.....	12
2.4. Bộ điều khiển PD Fuzzy	14
2.4.1. Cấu trúc bộ điều khiển Fuzzy	14
2.4.2. Bộ điều khiển PD Fuzzy	15
2.5. Nền tảng ROS2	16
2.5.1. Tầng ROS File System	17
2.5.2. ROS2 Graph	19
2.5.3. Tầng ROS Community	20
2.5.4. Package TF trong ROS.....	21
2.5.5. Một số phần mềm mô phỏng trong ROS.....	21
2.5.6. Giao tiếp Micro-ROS (uROS)	22

Chương 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG.....	25
3.1. Cấu trúc tổng quát hệ thống	25
3.1.1. RPLLidar A1	26
3.1.2. Module cảm biến gia tốc và góc quay IMU.....	27
3.1.3. Máy tính nhúng Jetson Nano B01	29
3.1.4. Vi điều khiển Raspberry Pico	30
3.1.5. Driver motor, mainboard (tự thiết kế)	30
3.1.6. DC Servo JGB37-520 có Encoder	31
3.1.7. Pin sạc 18650 Li-on và mạch sạc pin 18650 4S	32
3.1.8. Mạch giảm áp LM1596S 3A và XL4015 5A.....	33
3.2. Sơ đồ công suất phần cứng	34
3.3. Thi công phần cứng.....	35
3.3.1. Thiết kế mạch điều khiển	35
3.3.1.1. Mạch điều khiển chính chứa vi điều khiển	35
3.3.1.2. Mạch điều khiển động cơ.....	38
3.3.2. Thiết kế mô hình	39
Chương 4. THIẾT KẾ HỆ THỐNG PHẦN MỀM	42
4.1. Cấu trúc phần mềm	42
4.2. Khối quỹ đạo tham chiếu	46
4.3. Khối điều khiển bám quỹ đạo	51
4.4. Khối điều khiển góc	54
4.5. Khối mô hình động học của robot.....	59
4.6. Khối điều khiển vận tốc	61
Chương 5. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ	63

5.1. Kết quả xây dựng mô hình robot.....	63
5.2. Môi trường tiến hành thực nghiệm.....	65
5.3. Kết quả điều khiển vận tốc.....	66
5.4. Kết quả điều khiển góc.....	72
5.5. Kết quả điều khiển bám quỹ đạo.....	76
Chương 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	86
6.1. Kết luận.....	86
6.2. Hướng phát triển.....	86
TÀI LIỆU THAM KHẢO.....	88
PHỤ LỤC	89

DANH MỤC HÌNH VẼ

Hình 2.1: Minh họa khoảng cách từ robot đến các điểm mốc trên hệ tọa độ	5
Hình 2.2: Minh họa phép biến đổi giữa 2 lần di chuyển của robot so với các điểm mốc trên hệ tọa độ	5
Hình 2.3: Chọn các điểm tương ứng trên hai tập dữ liệu	6
Hình 2.4: Minh họa lần lượt các bước lặp dự đoán để so khớp giữa 2 vị trí	7
Hình 2.5: Dữ liệu điểm-điểm xấp xỉ khoảng cách đến bề mặt tốt hơn so với số liệu điểm-điểm được sử dụng trong ICP vanilla	8
Hình 2.6: Minh họa các dự đoán	8
Hình 2.7: Minh họa phương pháp tính point-to-point	8
Hình 2.8: Minh họa phương pháp tính point-to-line	9
Hình 2.9: Mô hình sử dụng trong thuật toán Staley Controller	10
Hình 2.10: Sơ đồ khối bộ điều khiển PID	12
Hình 2.11: Cấu trúc bộ điều khiển Fuzzy	14
Hình 2.12: Sơ đồ khối bộ điều khiển PD Fuzzy	15
Hình 2.13: Cấu trúc File System trên ROS	17
Hình 2.14: Truyền nhận message trong ROS	18
Hình 2.15: Service trong ROS	18
Hình 2.16: ROS2 graph	19
Hình 2.17: Vai trò của ROS Master	19
Hình 2.18: Trao đổi dữ liệu giữa các Node qua Topic trên ROS	20
Hình 2.19: Cấu trúc tầng ROS Community	20
Hình 2.20: Cấu trúc TF của một Robot	21
Hình 2.21: Mô hình có được sau khi xuất file URDF từ phần mềm Solidwork ..	22
Hình 2.22: Kiến trúc của uROS	23
Hình 2.23: minh họa uROS kết nối giữa CPU và MCU	24
Hình 3.1: cấu trúc tổng quát hệ thống	25
Hình 3.2: RPLidar A1	26

Hình 3.3: Các packet truyền khi Lidar quét 360°	26
Hình 3.4: IMU sử dụng cảm biến MEMS ADIS16488	27
Hình 3.5: NVIDIA Jetson Nano B01	29
Hình 3.6: Raspberry Pico.....	30
Hình 3.7: Mainboard.....	30
Hình 3.8: Driver motor	31
Hình 3.9: DC Servo JBG37-520	31
Hình 3.10: Pin sạc Li-ion 18650	32
Hình 3.11: Mạch sạc pin 18650 4S	32
Hình 3.12: Mạch giảm áp DC LM2596S3	33
Hình 3.13: Mạch giảm áp DC XL4015 5A.....	33
Hình 3.14: Sơ đồ khối công suất phần điều khiển.....	34
Hình 3.15: Sơ đồ khối công suất phần động lực.....	35
Hình 3.16: Bảng vẽ schematic của Mainboard.....	36
Hình 3.17: PCB top và bottom layer của Mainboard.....	37
Hình 3.18: Bản vẽ schematic của Driver.....	39
Hình 3.19: PCB top và bottom layer của Driver	39
Hình 3.20: Kích thước tầng dưới (đơn vị mm)	40
Hình 3.21: Kích thước tầng giữa (đơn vị mm).....	40
Hình 3.22: Kích thước tầng trên (đơn vị mm)	41
Hình 3.23: Mô hình robot thiết kế dùng SoildWorks.....	41
Hình 4.1: Cấu trúc phần mềm của hệ thống.....	42
Hình 4.2: Chạy thành công chương trình tạo node trên raspberry pico W	42
Hình 4.3: Sơ đồ các khối thuật toán sử dụng trong hệ thống.....	46
Hình 4.5: Quỹ đạo đường thẳng	47
Hình 4.6: Góc tham chiếu của quỹ đạo đường thẳng	47
Hình 4.7: Quỹ đạo đường tròn	48
Hình 4.8: Góc tham chiếu của quỹ đạo đường tròn	49

Hình 4.9: Quỹ đạo hình thang.....	50
Hình 4.10: Góc tham chiếu của quỹ đạo hình thang.....	50
Hình 4.11: Sơ đồ khối thuật toán Stanley.....	51
Hình 4.12: Hình vẽ minh họa xét dấu của $\Delta\theta$	53
Hình 4.13: Sơ đồ khối giải thuật bộ điều khiển PD Fuzzy.....	54
Hình 4.14: Tập mờ cho biến ngõ vào e	55
Hình 4.15: Tập mờ cho biến ngõ vào \dot{e}	56
Hình 4.16: Tập mờ cho biến ngõ ra u	56
Hình 4.17: Minh họa một số quy tắc điển hình	58
Hình 4.18: Vận tốc của một bánh xe	59
Hình 4.19: Mô hình mô tả động học của robot.....	60
Hình 4.20: Sơ đồ khối giải thuật bộ điều khiển PID.....	61
Hình 5.1: Mặt trước và mặt bên trái của robot.....	63
Hình 5.2: Mặt sau và mặt bên phải của robot.....	63
Hình 5.3: Mặt trên và mặt dưới của robot	64
Hình 5.4: Thông số kích thước mô hình robot.....	64
Hình 5.5: Môi trường tiến hành thực nghiệm	65
Hình 5.6: Đáp ứng điều khiển vận tốc M1 có tải, $K_p = 0.1, K_I = 0.6, K_D = 0$	66
Hình 5.7: Đáp ứng điều khiển vận tốc M2 có tải, $K_p = 0.2, K_I = 0.7, K_D = 0$	67
Hình 5.8: Đáp ứng điều khiển vận tốc M3 có tải, $K_p = 0.1, K_I = 0.6, K_D = 0$	68
Hình 5.9: Đáp ứng điều khiển vận tốc M4 có tải, $K_p = 0.1, K_I = 0.6, K_D = 0$	69
Hình 5.10: Đáp ứng vận tốc tuyến tính khi robot đi thẳng.....	70
Hình 5.11: Đáp ứng vận tốc góc khi robot đi thẳng	70
Hình 5.12: Đáp ứng góc của robot trường hợp góc đặt là hàm nắc.....	72
Hình 5.13: Sai số và tín hiệu điều khiển của PD Fuzzy trường hợp góc đặt là hàm nắc	73
Hình 5.14: Đáp ứng góc của robot trường hợp góc đặt là hàm sin	74

Hình 5.15: Sai số và tín hiệu điều khiển của PD Fuzzy trường hợp góc đặt là hàm sin	75
Hình 5.16: Đáp ứng quỹ đạo với ref là đường thẳng 0°	76
Hình 5.17: Sai số vị trí với ref là đường thẳng 0°	77
Hình 5.18: Đáp ứng góc với ref là đường thẳng 0°	77
Hình 5.19: Sai số góc với ref là đường thẳng 0°	78
Hình 5.20: Đáp ứng quỹ đạo với ref là đường thẳng 45°	79
Hình 5.21: Sai số vị trí với ref là đường thẳng 45°	79
Hình 5.22: Đáp ứng góc với ref là đường thẳng 45°	80
Hình 5.23: Sai số góc với ref là đường thẳng 45°	80
Hình 5.24: Đáp ứng quỹ đạo với ref là đường tròn.....	81
Hình 5.25: Sai số vị trí với ref là đường tròn	81
Hình 5.26: Đáp ứng góc với ref là đường tròn	82
Hình 5.27: Sai số góc với ref là đường tròn.....	82
Hình 5.28: Đáp ứng quỹ đạo với ref là hình thang	83
Hình 5.29: Sai số vị trí với ref là hình thang.....	83
Hình 5.30: Đáp ứng góc với ref là hình thang.....	84
Hình 5.31: Sai số góc với ref là hình thang	84
Hình 0.1: Đáp ứng vận tốc của 4 motor và robot khi chạy với vận tốc tối đa	89
Hình 0.2: Đáp ứng vận tốc của M1 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s..	90
Hình 0.3: Đáp ứng vận tốc của M2 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s..	90
Hình 0.4: Đáp ứng vận tốc của M3 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s..	91
Hình 0.5: Đáp ứng vận tốc của M4 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s..	91
Hình 0.6: Đáp ứng vận tốc robot với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s	92

DANH MỤC BẢNG

Bảng 2.1: Ảnh hưởng của các hệ số lên bộ điều khiển PID.....	14
Bảng 3.1: Thông số kỹ thuật của Module IMU.....	28
Bảng 3.2: Frame truyền dữ liệu của IMU.....	28
Bảng 4.1: Ý nghĩa của các topic và tần số truyền nhận dữ liệu của chúng	44
Bảng 4.2: Ý nghĩa của các node	45
Bảng 4.3: Các qui tắc mờ.....	58
Bảng 5.1: Kết quả điều khiển vận tốc PID	71
Bảng 5.2: Kết quả điều khiển bám quỹ đạo	85

DANH MỤC TỪ VIẾT TẮT

Từ khóa	Tên tiếng Anh
ROS	Robot Operating System
uROS	Micro-ROS
ICP	Iterative Closest Point
PL-ICP	Point to Line Iterative Closets Point
MSE	Mean Square Error
PID	Proportional Integral Derivative
MPC	Model Predictive Control
IMU	Intertial Measurement Unit

TÓM TẮT ĐỒ ÁN

Đề tài mô tả quá trình thiết kế, xây dựng và lập trình mô hình robot truyền động bốn bánh độc lập để bám quỹ đạo cho trước. Robot được thiết kế sử dụng vi điều khiển Raspberry Pico để điều khiển 4 động cơ động lập và kết nối với máy tính nhúng Jetson Nano thông qua giao thức uROS. Máy tính nhúng thu thập dữ liệu vị trí và hướng của robot bằng lidar A1 và IMU, sau đó sử dụng thuật toán Stanley Controller để xác định góc di chuyển cần thiết để bám quỹ đạo. Bộ điều khiển PD Fuzzy sau đó xử lý giá trị góc cho ra giá vận tốc góc để điều khiển robot đạt được góc mong muốn. Từ vận tốc góc và vận tốc tuyến tính, ta có thể tính toán được vận tốc của từng bánh xe, và mỗi bánh xe được điều khiển thông qua một bộ PID riêng biệt. Các giá trị điều khiển đầu ra mỗi bộ PID được truyền từ máy tính nhúng xuống Raspberry Pico qua giao thức uROS. Các thuật toán, quá trình truyền nhận và xử lý dữ liệu được xây dựng trên nền tảng hệ điều hành ROS2.

Kết quả của đồ án cho thấy mô hình robot truyền động 4 bánh độc lập có thể di chuyển với vận tốc tối đa 2.1m/s. Với mục tiêu của việc điều khiển robot trong môi trường trong nhà, nhóm đã tiến hành thực hiện cho robot bám các quỹ đạo đã cho trước với vận tốc tuyến tính 0.3 m/s. Sai số khi bám các quỹ đạo có kích thước nhỏ thu được dao động từ 1cm đến 8cm cho thấy khả năng di chuyển và xoay linh hoạt của robot trong các không gian hạn chế cùng với tốc độ xử lý nhanh chóng.

ABSTRACT

This project describes the design, construction, and programming of an independent four-wheeled drive robot model for predefined path tracking. The robot construction utilized a Raspberry Pico microcontroller to control four independent motors and interfaced with a Jetson Nano embedded computer via the uROS protocol. The embedded computer collected position and orientation data of the robot using an A1 lidar and IMU, then employed the Stanley Controller algorithm to determine the necessary steering angle for path tracking. Subsequently, the PD Fuzzy controller then processes the angle value to obtain the angular velocity output for controlling the robot to achieve the desired angle. Based on the angular and linear velocities, wheel speeds were calculated, and each wheel was controlled by a separate PID controller. The output control values of each PID controller were transmitted from the embedded computer to the Raspberry Pico using the uROS protocol. The algorithms, data transmission, and processing were developed on the ROS2 operating system platform.

The results of the thesis demonstrate that the independent four-wheeled drive robot model can achieve a maximum speed of 2.1 m/s. With the goal of controlling the robot in indoor environments, the team conducted experiments to have the robot track predefined trajectories at a linear speed of 0.3 m/s. The trajectory tracking error ranged from 1 cm to 8 cm, indicating the robot's flexible movement capabilities in confined spaces with rapid processing speeds.

Chương 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

Robot ngày càng được sử dụng rộng rãi trong nhiều lĩnh vực của cuộc sống hàng ngày, thay thế vai trò của con người trong các công việc trước đây thường được thực hiện bằng tay. Từ nông nghiệp đến hoạt động khai thác mỏ, và từ sản xuất trong nhà máy đến chăm sóc bệnh nhân trong các bệnh viện, sự áp dụng của robot đã tăng cường hiệu suất, hiệu quả và an toàn trong các nhiệm vụ trước đây được xem là mệt nhọc, bẩn thỉu hoặc nguy hiểm đối với con người.

Nhờ sự phát triển liên tục của công nghệ, cánh tay robot có thể di chuyển với tốc độ và độ chính xác cao để thực hiện các hoạt động lặp đi lặp lại như sơn hoặc hàn,... Tuy nhiên, những robot này có một bất lợi lớn là thiếu tính cơ động, có phạm vi hoạt động hạn chế và phụ thuộc vào vị trí của nó trong dây chuyền lắp ráp. Ngược lại, một robot di động có thể di chuyển tự do linh hoạt qua lại trong nhà máy sản xuất và có thể đến bất cứ nơi nào nó được yêu cầu. Lợi thế lớn này đòi hỏi phải có một hệ thống điều khiển cho phép robot điều hướng di chuyển từ điểm A đến điểm B một cách hiệu quả, an toàn và có thể thực hiện các thao tác quay đầu nhanh chóng và xoay linh hoạt trong không gian hạn chế. Vì thế nhóm quyết định thiết kế mô hình robot truyền động 4 bánh có dạng skid steer car vì khả năng di chuyển linh hoạt vượt bậc của nó so với các loại xe lái vi sai thông thường, đồng thời sử dụng bộ điều khiển Stanley cho mục đích bám quỹ đạo vì chi phí triển khai và thực hiện của bộ điều khiển thấp hơn so với nhiều giải pháp khác, ngoài ra cũng đảm bảo độ chính xác ổn định và tốc độ xử lý nhanh, cùng với việc dễ dàng tích hợp vào hệ thống điều khiển tổng thể. Những yếu tố trên đã tạo động lực cho nhóm nghiên cứu và thực hiện đề tài “XÂY DỰNG VÀ ĐIỀU KHIỂN MÔ HÌNH XE ĐIỆN TRUYỀN ĐỘNG 4 BÁNH TOÀN THỜI GIAN”.

1.2. Nội dung và mục tiêu của đề tài

1.2.1. Nội dung đề tài

Xây dựng mô hình robot thực tế di chuyển trong nhà, sử dụng các cảm biến Lidar, IMU và Encoder để bám quỹ đạo đã hoạch định trước.

Thực hiện bộ điều khiển Stanley, PD Fuzzy, PID và thuật toán ICP để xác định vị trí của robot trong không gian và điều khiển robot đi theo quỹ đạo một cách chính xác. Với mong muốn giảm thiểu sai số bám quỹ đạo, đồng thời tối ưu tốc độ xử lý.

1.2.2. Mục tiêu đề tài

Kết quả của bộ điều khiển vận tốc và bộ điều khiển góc có sai số không quá 1 cm/s và 1° đồng thời độ vọt lố dưới 10% và thời gian xác lập nhỏ hơn 3s.

Kết quả của thuật toán bám quỹ đạo có sai số khoảng cách không quá 10 cm.

1.3. Cấu trúc của đề tài

Đề tài gồm có 6 chương với nội dung như sau:

Chương 1: Giới thiệu tính cấp thiết và lý do chọn đề tài, đồng thời trình bày nội dung, mục tiêu và cấu trúc đề tài.

Chương 2: Trình bày cơ sở lý thuyết của các bộ điều khiển, các thuật toán sử dụng trong đề tài.

Chương 3: Giới thiệu các thiết bị sử dụng trong đề tài. Trình bày chi tiết về cấu trúc, chức năng và nhiệm vụ của từng khối trong phần cứng.

Chương 4: Trình bày hệ thống phần mềm sử dụng cho thực nghiệm.

Chương 5: Trình bày kết quả thực nghiệm và đánh giá sai số.

Chương 6: Nêu ra kết luận và hướng phát triển của đề tài.

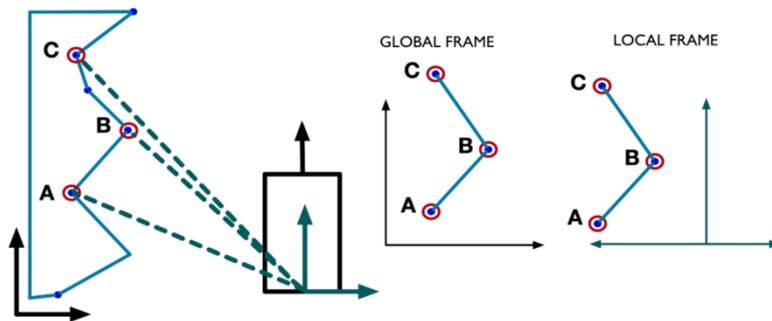
Chương 2. CƠ SỞ LÝ THUYẾT

2.1. Thuật toán ICP xây dựng bản đồ 2D

Thuật toán Scan matching

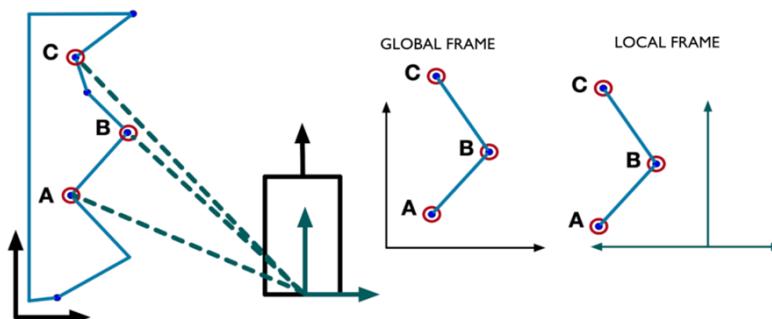
Đây là thuật toán hỗ trợ robot có trang bị cảm biến lidar, giúp tối ưu hóa sự liên kết giữa các chùm điểm cuối (end-points) trên bản đồ và các chùm điểm cuối vừa mới thu được từ cảm biến lidar.

Đặt vấn đề: Robot đang ở trong môi trường nào đó với 3 điểm mốc A, B, C. Tại thời điểm $t=0$, ta đo được khoảng cách từ robot đến từng điểm A, B, C và thể hiện trên tọa độ toàn bộ và tọa độ cục bộ tương ứng như sau:



Hình 2.1: Minh họa khoảng cách từ robot đến các điểm mốc trên hệ tọa độ

Tại thời điểm $t=1$, robot di chuyển một khoảng chưa rõ, khi này khoảng cách từ robot đến các điểm đánh dấu bị thay đổi. Chúng ta cần tìm một phép biến đổi R mà biến đổi 2 tập hợp điểm là gần nhất:



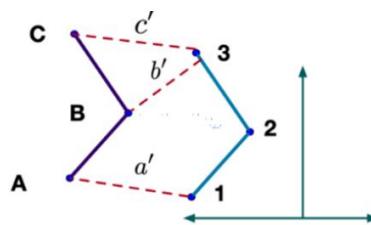
Hình 2.2: Minh họa phép biến đổi giữa 2 lần di chuyển của robot so với các điểm mốc trên hệ tọa độ

Nếu chúng ta biết chính xác các điểm mốc A, B, C thì việc tìm R là đơn giản. Tuy nhiên chúng ta lại không thể thực hiện các phép đo chính xác các điểm mốc này, nghĩa là ta không thể xác định được các cặp điểm tương ứng trong hệ tọa độ robot giữa các lần di chuyển. Dưới đây ta có hai hướng tiếp cận:

2.1.1. Thuật toán ICP

Thuật toán ICP lựa chọn biến đổi giữa 2 chuyển động dựa trên đo lường từ điểm đến điểm (point – to – point), cụ thể là so sánh hàm tổng khoảng cách giữa các điểm từ dữ liệu ban đầu đến điểm tương ứng ở tập dữ liệu mới, các điểm tương ứng được lựa chọn là các tập điểm trên 2 tập dữ liệu có khoảng cách gần nhau nhất.

Điều kiện tối ưu 2-norm: trong đó m là số điểm thu thập được, là khoảng cách giữa các điểm tương ứng:



Hình 2.3: Chọn các điểm tương ứng trên hai tập dữ liệu

Các bước thực hiện thuật toán:

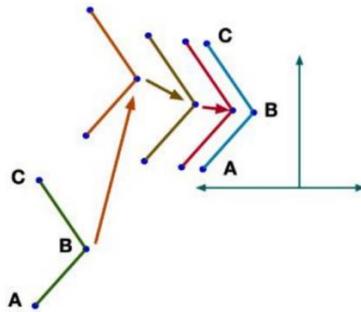
Bước 1: Khởi tạo giá trị error và giá trị ngưỡng threshold (error > threshold)

Bước 2: Định nghĩa các cặp điểm tương ứng bằng cách chọn các cặp điểm dữ liệu mới và cũ gần nhau nhất.

Bước 3: Tính toán vector xoay R và vector dịch chuyển T giữa giá trị dữ liệu cũ so với dữ liệu lidar mới thu thập được.

Bước 4: Dùng giá trị R và T để tính giá trị sai số $error = \sqrt{\sum_i^m \Delta x_i^2}$

Bước 5: Nếu giá trị error giảm và lớn hơn mức ngưỡng threshold thì ta quay lại bước 2, ngược lại xuất giá trị R và T để xác định vị trí robot trong bản đồ.



Hình 2.4: Minh họa lần lượt các bước lặp dự đoán để so khớp giữa 2 vị trí

Các hạn chế của thuật toán:

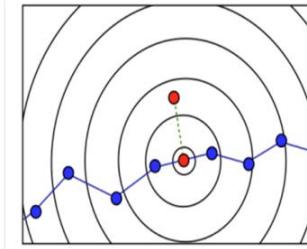
Thuật toán hoạt động tốt nhất với các giả định sau nhưng lại ít xảy ra trong thực tế: Cần tốc độ quét cực nhanh để có: sự chồng chéo đáng kể tồn tại giữa các lần quét liên tiếp và tồn tại các điểm có sự tương ứng cao. Giả định bề mặt nhẵn và đồng nhất.

Thiết lập dự đoán là quan trọng trong ICP để hội tụ.

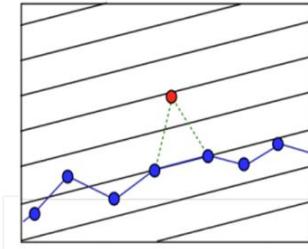
Việc lặp lại tính toán giữa từng điểm tốn nhiều tài nguyên và thời gian, dẫn tới không được real-time và gây ra nhiều sai số nếu quá trình chuyển động nhanh.

2.1.2. Thuật toán PL-ICP

Thuật toán PL-ICP là một cải tiến trong việc “tìm kiếm tương ứng” để mang lại hiệu quả hơn. Chuyển từ việc tính toán giữa Điểm tới điểm (point-to-point) sang tính toán giữa Điểm tới đường (Point-to-line). Với point-to-point, các đường viền của tập cấp là đường tròn đồng tâm và tập trung tại điểm chiếu. Với point-to-line, các đường viền của tập cấp dạng đường. Do đó, mức thiết lập bề mặt gần đúng tốt hơn. Điều này có nghĩa là chúng ta có một xấp xỉ chính xác hơn của lỗi, giảm không gian biến đổi, kết quả hội tụ nhanh hơn.



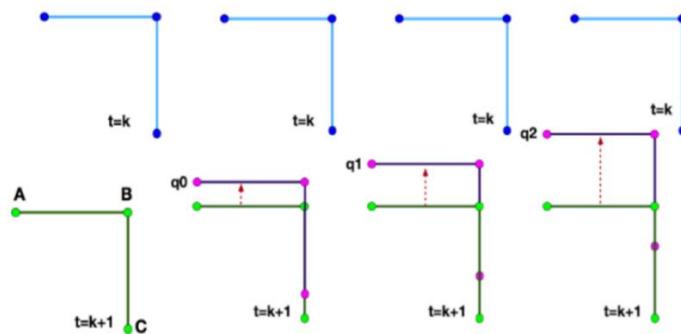
Point-to-point (vanilla ICP)



Point-to-line (PL-ICP)

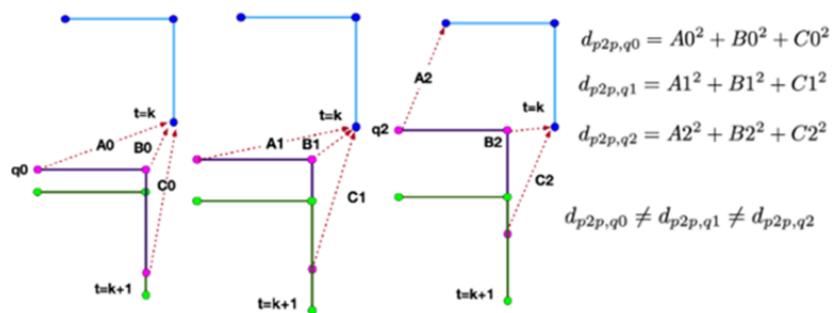
Hình 2.5: Dữ liệu điểm-điểm xấp xỉ khoảng cách đến bề mặt tốt hơn so với số liệu điểm-điểm được sử dụng trong ICP vanilla¹

Ta lấy ví dụ minh họa, để so khớp thời điểm $t=k$ và thời điểm $t=k+1$, lần lượt có các dự đoán q_0, q_1, q_2 cần tính toán sự tương ứng:



Hình 2.6: Minh họa các dự đoán

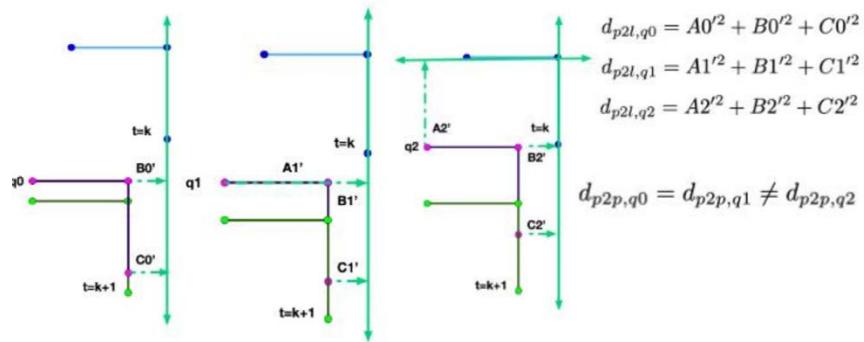
Đối với phương pháp tính toán point-to-point, ta phải lặp đủ số lần do tính toán các khoảng cách so khớp của các dự đoán q_0, q_1, q_2 khác nhau



Hình 2.7: Minh họa phương pháp tính point-to-point

¹ Andrea Censi, “An ICP variant using a point-to-line metric”, in 2008 IEEE International Conference on Robotics and Automation, 2008.

Với point-to-line, chúng ta do kết quả tính toán bằng nhau, số lần lặp được giảm bớt (lần lặp tiếp theo từ q_0 sang q_2 , bỏ qua lần lặp q_1 do kết quả giống nhau)



Hình 2.8: Minh họa phương pháp tính point-to-line

2.2. Thuật toán Stanley

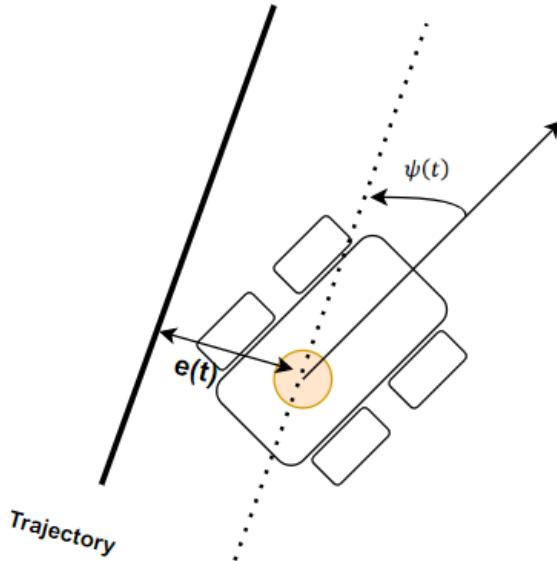
Bộ điều khiển Stanley, được phát triển bởi Tiến sĩ Gabe Hoffman của Đại học Stanford, là một bộ điều khiển đơn giản nhưng vô cùng hữu ích cho người máy tự động và các loại xe tự hành. Bộ điều khiển này đã chứng tỏ sức mạnh của mình khi giúp đội đua Stanford đoạt chiến thắng trong sự kiện Darpa Grand Challenge lần thứ hai. Trong quá trình phát triển, mục tiêu chính của bộ điều khiển Stanley là tạo ra các tín hiệu điều khiển để cải thiện vị trí và hướng đi, dựa trên việc phát hiện các sai sót như lỗi đường đi chéo và sai số góc so với quỹ đạo tham chiếu của robot tự hành. Các điều kiện hình thành cơ sở cho việc hình thành luật điều khiển đối với mô hình xe đang điều khiển là:

Sử dụng tâm của trục Lidar làm điểm tham chiếu trên robot, do đó thuật ngữ vị trí hiện tại của robot được dùng như là vị trí hiện tại của điểm tham chiếu trên robot.

Sử dụng cả sai số trong hướng chuyển động so với quỹ đạo tham chiếu và sai số ở vị trí hiện tại đến điểm gần nhất trên quỹ đạo của robot để tạo ra tín hiệu điều khiển.

Với cơ cấu phần cứng của robot đề xuất, đầu ra góc điều khiển của bộ điều khiển Stanley không bị giới hạn (đầu ra có thể -180° đến 180°).

Crosstrack error được đo từ vị trí hiện tại của robot đến quỹ đạo tham chiếu:



Hình 2.9: Mô hình sử dụng trong thuật toán Stanley Controller

Với $\psi(t)$ là góc giữa quỹ đạo và hướng của xe, $\delta(t)$ là góc điều khiển cần suất để điều khiển xe. Đầu tiên, để loại bỏ lỗi của hướng chuyển động so với quỹ đạo, góc điều khiển được đặt bằng với hướng chuyển động:

$$\delta(t) = \psi(t)$$

Sau đó, để loại bỏ crosstrack error, một bộ điều khiển tỷ lệ được thêm vào, mức tăng của nó tỷ lệ nghịch với vận tốc và tỷ lệ thuận với crosstrack error. Bộ điều khiển sau đó được đưa qua một hàm \tan^{-1} để tín hiệu điều khiển nằm trong khoảng từ $-\pi$ đến π :

$$\psi'(t) = \tan^{-1}\left(\frac{k * e(t)}{v(t)}\right)$$

Với cơ cấu phần cứng của robot đề xuất, đầu ra góc điều khiển của bộ điều khiển Stanley không bị giới hạn (đầu ra có thể -180° đến 180°)

$$\psi'(t) \in [-180^\circ, 180^\circ]$$

Từ đó ta có luật điều khiển cuối cùng:

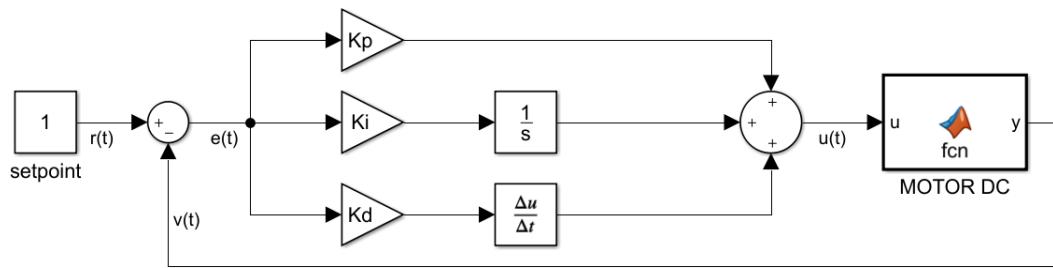
$$\delta(t) = \psi(t) + \psi'(t) = \psi(t) + \tan^{-1}\left(\frac{k^* e(t)}{v_f(t)}\right)$$

Tuy nhiên, trên thực tế, bộ điều khiển Stanley vẫn là bộ điều khiển bám quỹ đạo hình học (geometric path tracking controller) và do đó không xem xét nhiều khía cạnh khác nhau của robot tự hành thực sự. Ví dụ, nó không xem xét các nhiễu phép đo, động lực học cơ cấu hoặc hiệu ứng lực căng của lốp xe, tất cả đều có thể gây ra các đặc điểm không mong muốn trong quá trình điều khiển. Tuy nhiên, có thể thực hiện một vài điều chỉnh cho bộ điều khiển giúp giảm thiểu một số hiệu ứng không mong muốn này. Trong quá trình vận hành ở tốc độ thấp, vì thuật ngữ vận tốc nằm trong mẫu số của phân số bên trong \tan^{-1} , những giá trị sai số nhỏ trong vận tốc có xu hướng được khuếch đại trong tín hiệu điều khiển, điều này dẫn đến sự thay đổi lớn của góc điều khiển mong muốn. Vì vậy, để thoát khỏi vấn đề này và để tăng tính ổn định ở vận tốc thấp, chúng ta thêm một hằng số k_{soft} (softening constant) dương để đảm bảo mẫu số luôn có một giá trị nhỏ nhất. Hằng số k_{soft} này có thể được điều chỉnh trong thực nghiệm.

$$\delta(t) = \psi(t) + \psi'(t) = \psi(t) + \tan^{-1}\left(\frac{k^* e(t)}{k_{\text{soft}} + v_f(t)}\right)$$

2.3. Bộ điều khiển PID

Vận dụng thuật toán bộ điều khiển PID trong việc điều khiển tốc độ động cơ DC, sử dụng hàm truyền của bộ điều khiển PID rời rạc từ đó tìm được phương trình điều khiển.



Hình 2.10: Sơ đồ khối bộ điều khiển PID

Trong đó:

$r(t)$: giá trị vận tốc mong muốn

$v(t)$: giá trị vận tốc thực tế của động cơ

$e(t)$: sai số vận tốc giữa giá trị đặt và thực tế

Ta có hàm truyền của bộ điều khiển PID rời rạc có dạng như sau:

$$G(z) = \frac{U(z)}{E(z)} = K_p + \frac{K_I T}{2} \frac{z+1}{z-1} + \frac{K_D}{T} \frac{z-1}{z}$$

Viết lại phương trình trên ta được:

$$G(z) = \frac{\left(K_p + \frac{K_I T}{2} + \frac{K_D}{T}\right) + \left(-K_p + \frac{K_I T}{2} - \frac{2K_D}{T}\right)z^{-1} + \left(\frac{K_D}{T}\right)z^{-2}}{1 - z^{-1}}$$

Đặt:

$$\begin{aligned} a_0 &= K_p + \frac{K_I T}{2} + \frac{K_D}{T} \\ a_1 &= -K_p + \frac{K_I T}{2} - \frac{2K_D}{T} \\ a_2 &= \frac{K_D}{T} \end{aligned}$$

Ta được:

$$G(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}}$$

Từ đó, ta tính được giá trị ngõ ra của bộ điều khiển PID $u(k)$ khi giá trị ngõ vào là $e(k)$ như sau:

$$u(k) = G(z)e(k) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 - z^{-1}} e(k)$$

Thay các giá trị a_0 , a_1 và a_2 vào phương trình trên và áp dụng tính chất dời thời gian theo phép biến đổi Z ta có:

$$u(k) = u(k-1) + \left(K_p + \frac{K_I T}{2} + \frac{K_D}{T} \right) e(k) + \left(-K_p + \frac{K_I T}{2} - \frac{2K_D}{T} \right) e(k-1) + \frac{K_D}{T} e(k-2)$$

Viết lại phương trình trên ta tìm được giá trị tại các khâu P, I và D như sau:

$$\begin{aligned} P_{Part} &= K_p (e(k) - e(k-1)) \\ I_{Part} &= \frac{K_I T}{2} (e(k) + e(k-1)) \\ D_{Part} &= \frac{K_D}{T} (e(k) - 2e(k-1) + e(k-2)) \end{aligned}$$

Từ đó ta có được phương trình bộ điều khiển PID rời rạc dùng cho điều khiển động cơ DC như sau:

$$u(k) = u(k-1) + P_{Part} + I_{Part} + D_{Part}$$

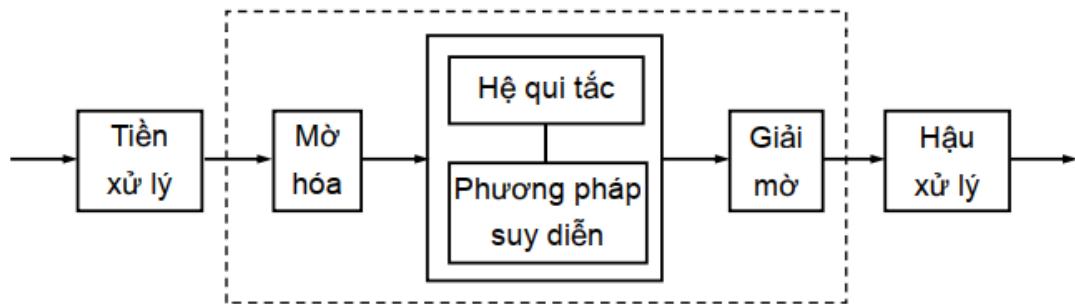
Điều khiển tỉ lệ K_p có ảnh hưởng làm giảm thời gian lên và sẽ làm giảm nhưng không loại bỏ sai số xác lập. Điều khiển tích phân K_I sẽ loại bỏ sai số xác lập nhưng có thể là đáp ứng bị vọt lố. Điều khiển vi phân K_D có tác dụng là tăng sự ổn định của hệ thống, giảm vọt lố và cải thiện đáp ứng quá độ. Ảnh hưởng của mỗi bộ điều khiển K_p , K_I và K_D lên hệ thống vòng kín được tóm tắt ở bảng dưới đây.

Đáp ứng vòng kín	Thời gian lên	Vọt lố	Thời gian xác lập	Sai số xác lập
K_p	Giảm	Tăng	Thay đổi nhỏ	Giảm
K_I	Giảm	Tăng	Tăng	Loại bỏ
K_D	Thay đổi nhỏ	Giảm	Giảm	Thay đổi nhỏ

Bảng 2.1: Ảnh hưởng của các hệ số lên bộ điều khiển PID

2.4. Bộ điều khiển PD Fuzzy

2.4.1. Cấu trúc bộ điều khiển Fuzzy



Hình 2.11: Cấu trúc bộ điều khiển Fuzzy

Một bộ điều khiển Fuzzy cơ bản gồm 5 khối:

Khối tiền xử lý: Tín hiệu vào bộ điều khiển thường là các giá trị rõ từ các mạch đo, bộ tiền xử lý có chức năng xử lý các giá trị đo này trước khi đưa vào bộ điều khiển mờ. Khối tiền xử lý có thể:

Ví phân, tích phân tín hiệu.

Chuẩn hóa, lượng tử hóa.

Lọc nhiễu.

Khối mờ hóa: Chuyển giá trị rõ phản hồi từ ngõ ra của đối tượng thành giá trị mờ để hệ quy tắc có thể suy luận được.

Khối hệ quy tắc mờ: Có thể được xem như mô hình toán học biểu diễn tri thức, kinh nghiệm của con người trong việc giải quyết bài toán dưới dạng các phát biểu ngôn ngữ. Các hệ quy tắc mờ thường dùng: Quy tắc mờ mamdani và quy tắc mờ sugeno. Hai phương pháp suy diễn Max – Min và Max – Prob là 2 phương pháp dùng trong suy luận từ hệ quy tắc mờ.

Khối giải mờ: Chuyển giá trị giải mờ suy luận được ở ngõ ra của khối hệ quy tắc mờ thành giá trị rõ để điều khiển đối tượng.

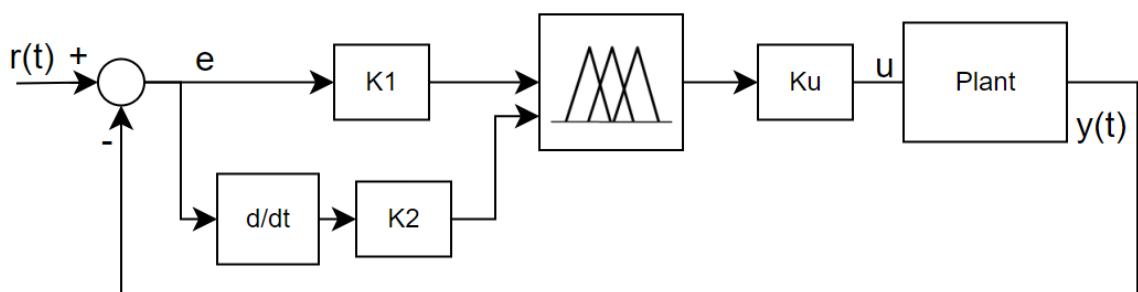
- Giải mờ (defuzzification) là chuyển đổi giá trị mờ ở ngõ ra của hệ mờ thành giá trị rõ.

- Các phương pháp giải mờ có thể quy vào 2 nhóm:

- + Giải mờ dựa theo độ cao: thường dùng trong các bài toán phân nhóm.
- + Giải mờ dựa vào điểm trọng tâm: thường dùng trong các bài toán điều khiển.

Khối hậu xử lý: Khuếch đại tín hiệu giải mờ chuẩn hóa thành giá trị vật lý.

2.4.2. Bộ điều khiển PD Fuzzy



Hình 2.12: Sơ đồ khối bộ điều khiển PD Fuzzy

Bộ điều khiển PD Fuzzy thường được sử dụng trong các trường hợp sau:

- Đối tượng có khâu tích phân lý tưởng.

- Ổn định hóa trạng thái của đối tượng xung quanh điểm cân bằng (\bar{u}, \bar{x}), trong đó $\bar{u} = 0$.

Trình tự thiết kế bộ điều khiển PD Fuzzy:

Bước 1: Dựa trên sơ đồ khối thiết kế bộ điều khiển PD Fuzzy, xác định tầm giá trị của:

- Biến vào: sai số (e) và vi phân sai số (e')
- Biến ra: tín hiệu điều khiển (u)

Bước 2: Xác định các hệ số chuẩn hóa giá trị biến vào, biến ra về miền giá trị $[-1, 1]$.

Bước 3: Định nghĩa các giá trị ngôn ngữ cho biến vào và biến ra, định lượng các giá trị ngôn ngữ bằng tập mờ.

Bước 4: Xây dựng hệ quy tắc mờ bằng cách vẽ hình minh họa, dựa trên đặc tính của đối tượng đang điều khiển mà đưa ra những một số quy tắc điển hình, sau đó có thể áp dụng tính đối xứng và tính liên tục của hệ mờ để đưa ra các quy tắc còn lại.

Bước 5: Chọn phương pháp suy diễn (Max – Min hay Max – Prod).

Bước 6: Chọn phương pháp giải mờ (trọng tâm hay trung bình có trọng số).

Bước 7: Mô phỏng hoặc thực nghiệm để đánh giá kết quả, tinh chỉnh thông số của bộ điều khiển để đạt được chất lượng mong muốn.

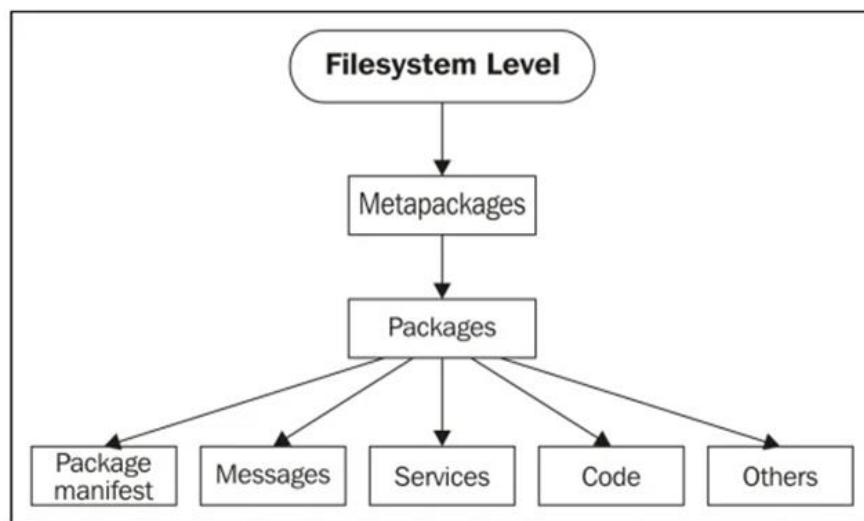
2.5. Nền tảng ROS2

Robot Operating System (ROS) ngày càng dần phổ biến trong cộng đồng robotics. ROS phiên bản đầu tiên (ROS1) đã phát triển không ngừng nhưng cuối cùng cũng đã ngừng hỗ trợ để nhường lại sự phát triển cho ROS2, một nền tảng đầy hứa hẹn. ROS2 sẽ khắc phục được nhiều nhược điểm mà ROS1 chưa thực hiện được, đồng thời cũng mở rộng khả năng của mình để có thể đi ở nhiều lĩnh vực, nhiều khía cạnh từ nghiên cứu đến công nghiệp.

Mặc dù chưa hẳn là một hệ điều hành hoàn chỉnh mà chính xác hơn là platform, nhưng ROS2 có các yếu tố cấu thành một hệ điều hành: trừu tượng hóa phần cứng, chuyển thông tin giữa các tiến trình, quản lý dữ liệu theo các gói, ... Mô hình ROS2 được khái quát như sau.

2.5.1. Tầng ROS File System

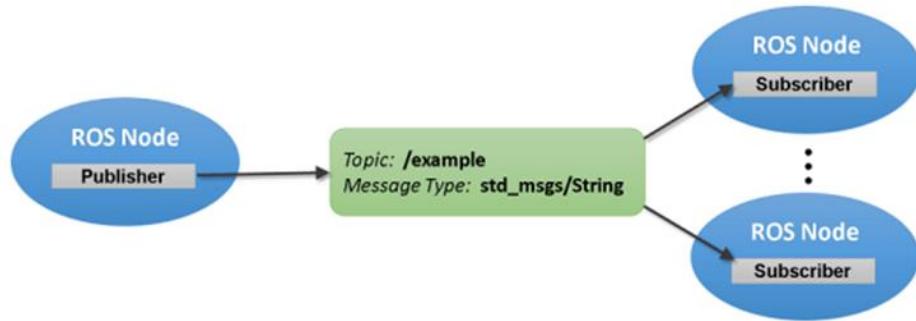
Filesystem chủ yếu là các tài nguyên trên phần cứng, cấu trúc thư mục và các tập tin tối thiểu để ROS hoạt động.



Hình 2.13: Cấu trúc File System trên ROS

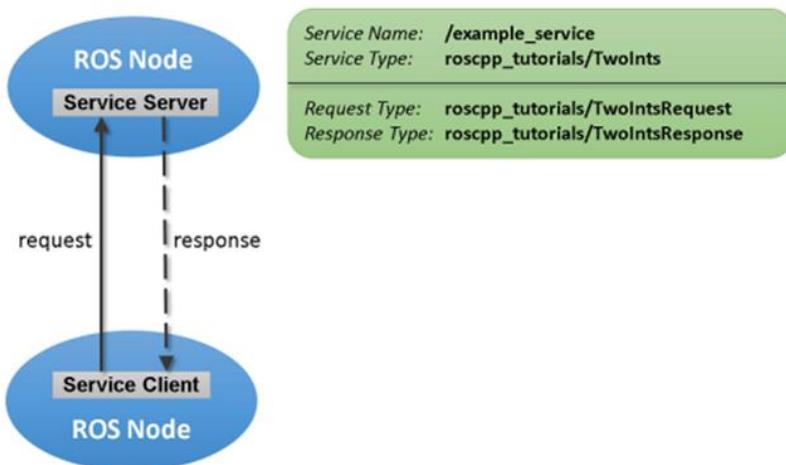
Packages: ROS Package là thành phần cơ bản nhất trên ROS. Nó bao gồm các node, thư viện, và các tệp tin cấu hình, ... được sắp xếp với nhau thành một thành phần duy nhất. Mục đích của gói mã nguồn là tạo ra chương trình tổng quát nhất để dễ dàng sử dụng lại cũng như phát triển.

Message: chứa cấu trúc thông tin theo một dạng cố định để giao tiếp giữa các node. Việc truyền/nhận messages dựa theo cơ chế publish/subscribe. Node Publisher sẽ gửi yêu cầu lấy thông tin từ node Subscriber, sau khi kết nối giữa các node thành công, quá trình gửi và nhận sẽ diễn ra liên tục.



Hình 2.14: Truyền nhận message trong ROS

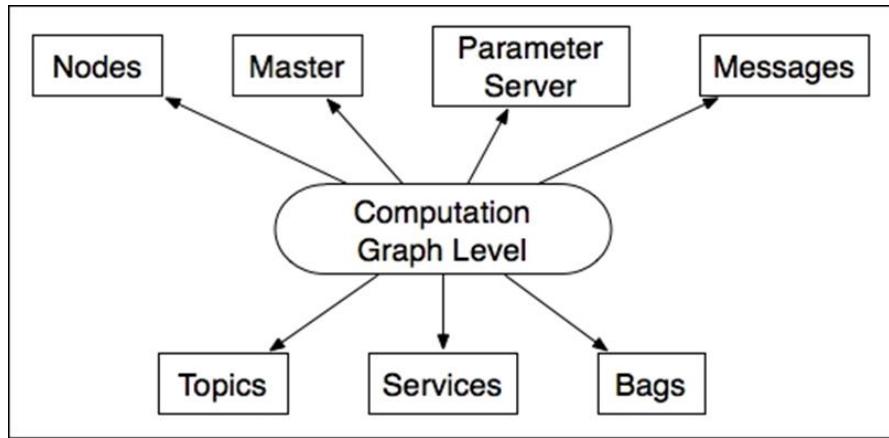
Services: tương tự như messages, services được dùng để trao đổi thông tin giữa các node. Tuy nhiên services hoạt động theo cơ chế request/respond.



Hình 2.15: Service trong ROS

2.5.2. ROS2 Graph

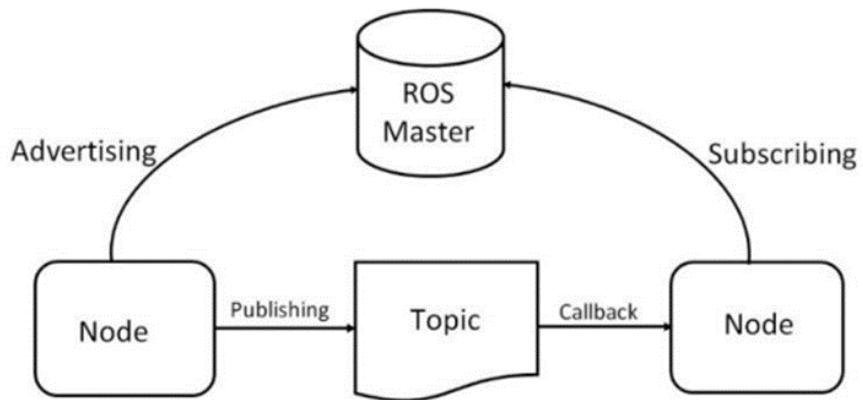
ROS graph là một mạng các phần tử ROS2 xử lý dữ liệu cùng nhau tại một thời điểm, là nơi mà các quy trình trong ROS được kết nối với nhau. Nó bao gồm tất cả các tệp thực thi (executables) và các kết nối giữa chúng nếu chúng ta muốn sắp xếp và thể hiện thị chúng. Tầng này bao gồm các khái niệm cơ bản sau:



Hình 2.16: ROS2 graph

Node: đơn vị thực thi cơ bản nhất. Mỗi node thực hiện một nhiệm vụ cố định là xử lý và truyền/nhận dữ liệu, nhờ vậy các tác vụ lớn, cần nhiều tính toán dễ dàng thực hiện bằng việc liên kết các node lại với nhau.

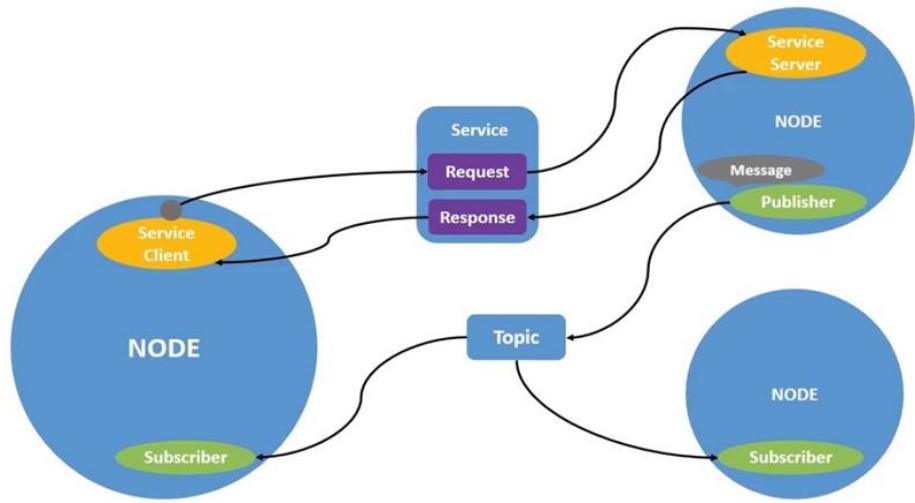
Master: đóng vai trò kết nối các node với nhau. Do đó, master luôn được khởi động đầu tiên, sau đó thì bạn có thể gọi bất kỳ các node nào trong hệ thống.



Hình 2.17: Vai trò của ROS Master

Message: Đây là cấu trúc dữ liệu được các node sử dụng để trao đổi với nhau tương tự như các kiểu dữ liệu double, int trong các ngôn ngữ lập trình. Các node tương tác với nhau bằng cách send và receive ROS message.

Topics: là phương pháp giao tiếp trao đổi dữ liệu giữa hai node, nó bao gồm nhiều cấp bậc thông tin mà chúng có thể giao tiếp thông qua ROS message. Hai phương thức trong topic bao gồm publish và subscribe.



Hình 2.18: Trao đổi dữ liệu giữa các Node qua Topic trên ROS

2.5.3. Tầng ROS Community

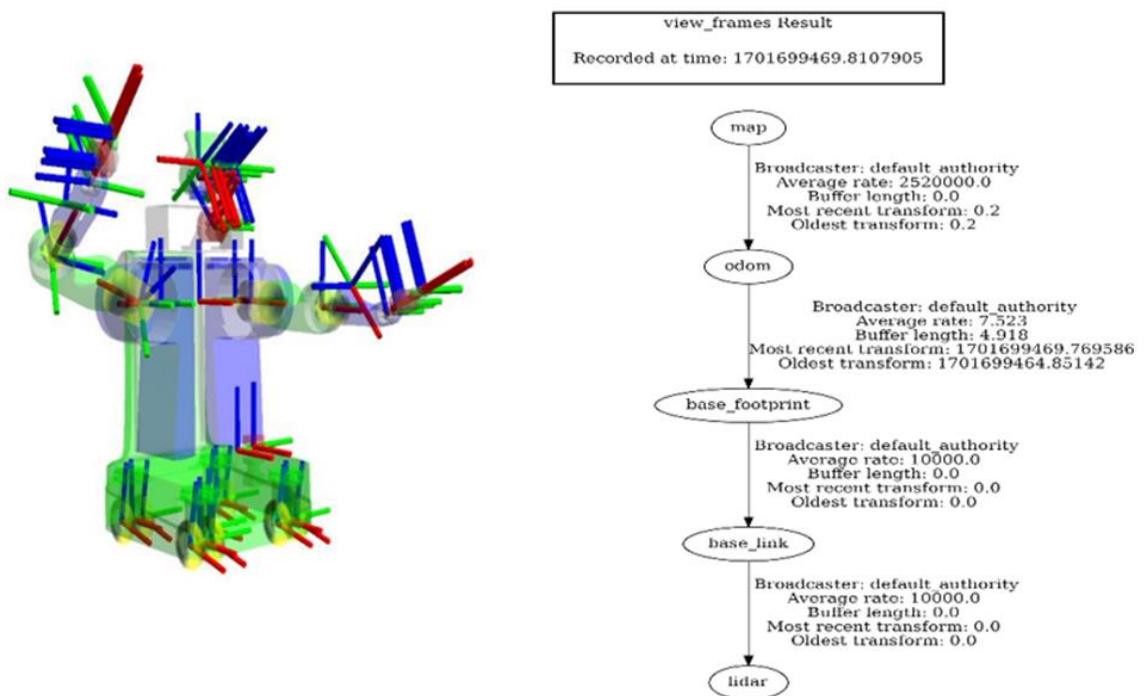
ROS Community là nguồn tài nguyên của ROS được cộng đồng người dùng trao đổi với nhau về phần mềm và kiến thức



Hình 2.19: Cấu trúc tầng ROS Community

2.5.4. Package TF trong ROS

TF là một package giúp người dùng quản lý được nhiều khung tọa độ qua thời gian. TF giữ được mối quan hệ giữa các khung tọa độ trong một cấu trúc dạng cây (tree structure) được đếm vào theo thời gian, cho phép người dùng chuyển đổi (transform) các điểm, vector, ... giữa 2 khung tọa độ bất kỳ thời điểm nào. Robot được trang bị nhiều cảm biến, việc tìm ra mối quan hệ vị trí giữa cảm biến so với robot là rất quan trọng cho việc phân tích ý nghĩa của dữ liệu sau này. Thư viện TF được ROS cung cấp giúp người dùng dễ dàng định nghĩa mô hình vật lý cho robot.



Hình 2.20: Cấu trúc TF của một Robot

2.5.5. Một số phần mềm mô phỏng trong ROS

RVIZ và Gazebo là hai phần mềm nổi tiếng nhất trong ROS. Đây là một giao diện đồ họa của ROS, cung cấp khả năng mô phỏng chính xác và hiệu quả robot trong môi trường phức tạp cả trong nhà lẫn ngoài trời, cho phép hiển thị dữ

liệu 3 chiều của các loại cảm biến hoặc các robot sử dụng phương thức mô tả mô hình vật lý Unified Robot Description Format (URDF).



Hình 2.21: Mô hình có được sau khi xuất file URDF từ phần mềm Solidwork

2.5.6. Giao tiếp Micro-ROS (uROS)

Ở ROS2, thay vì sử dụng giao thức truyền thông ROS1 (ROS1 Communication System), ROS2 đã chuyển sang sử dụng giao thức truyền thông mới được gọi là "uROS" (Micro-ROS Communication System). Giao thức uROS được thiết kế để hỗ trợ ROS 2 chạy trên các hệ thống có tài nguyên hạn chế, như các thiết bị nhúng, vi điều khiển, và các ứng dụng có yêu cầu về kích thước và hiệu suất. Dưới đây là một số điểm chính về giao thức uROS trong ROS2:

Thiết kế cho Hệ thống Nhúng: giao thức uROS được tối ưu hóa cho các hệ thống có tài nguyên hạn chế, với mục tiêu chạy trên các thiết bị nhúng và vi điều khiển với bộ nhớ và xử lý giới hạn.

Hiệu suất và kích thước nhỏ: uROS được thiết kế để có hiệu suất tốt và kích thước nhỏ, giúp giảm yêu cầu về băng thông và tài nguyên.

Hỗ trợ các Middleware Nhúng: uROS có thể tích hợp với các middleware nhúng như Micro XRCE-DDS (Data Distribution Service) để cung cấp các tính

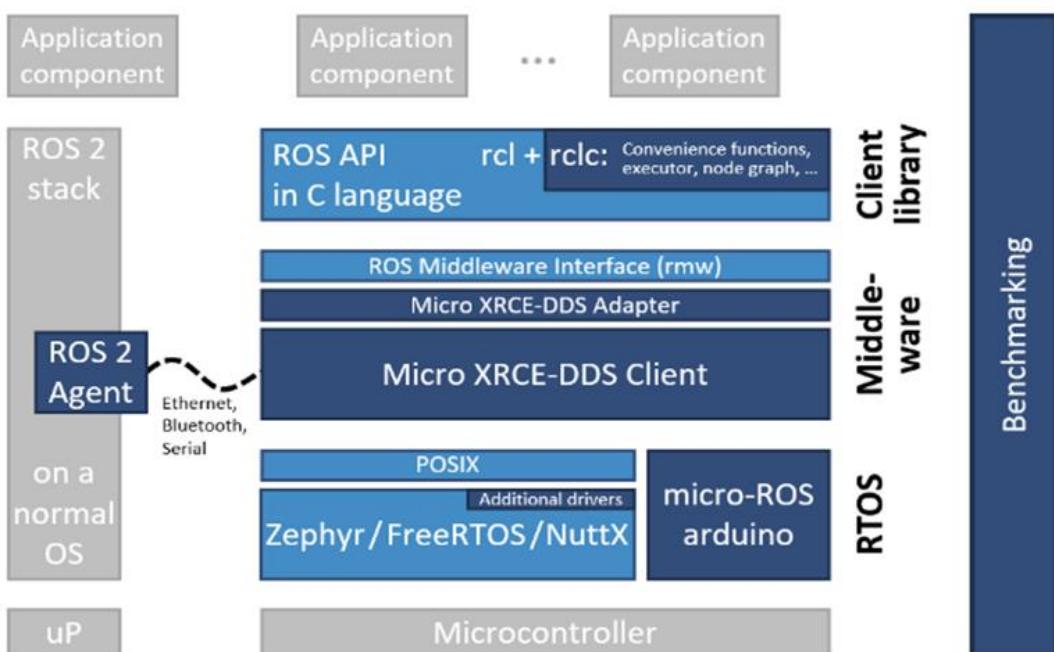
năng như truyền thông, đồng bộ dữ liệu, và phát hiện nhanh chóng trong môi trường có hạn.

Phần Mềm Bảo Mật: giao thức uROS hỗ trợ các tính năng bảo mật như mã hóa và xác thực để đảm bảo an toàn trong quá trình truyền thông dữ liệu.

Hỗ trợ ROS2: uROS không thay thế ROS2 mà là một phần của ROS2, đặc biệt là được tích hợp với ROS2 trong hệ thống ROS2 chung.

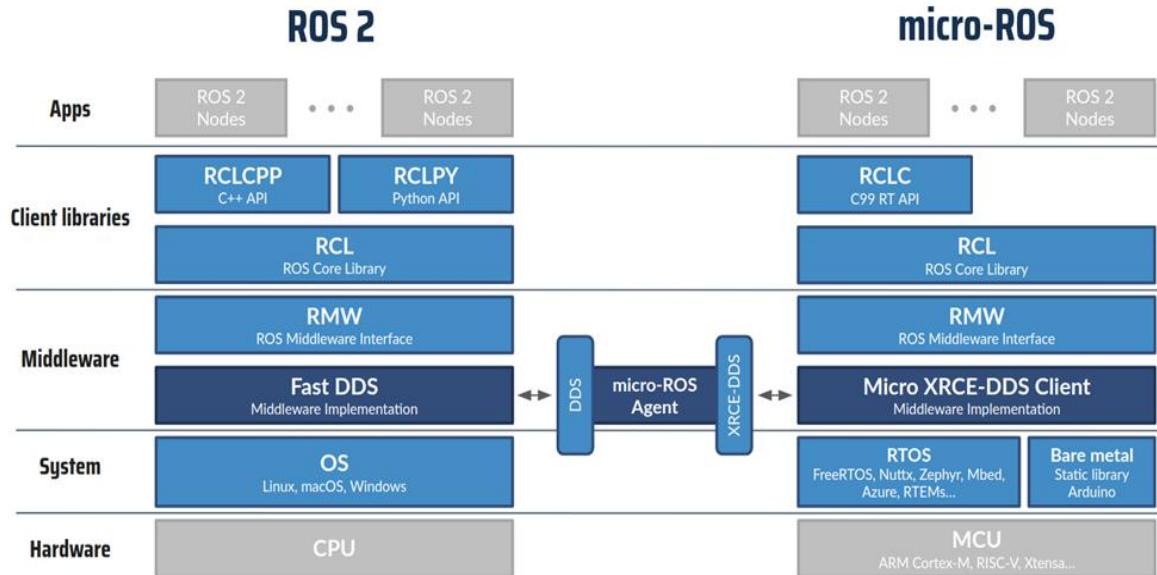
Hỗ trợ Các Ngôn Ngữ Lập Trình: Giao thức uROS hỗ trợ việc viết ứng dụng với nhiều ngôn ngữ lập trình khác nhau, giúp linh hoạt khi phát triển ứng dụng trên các nền tảng nhúng.

Đơn Giản Hóa cho Hệ thống Nhúng: mục tiêu của uROS là đơn giản hóa việc triển khai ROS 2 trên các hệ thống có tài nguyên hạn chế, giúp nhà phát triển dễ dàng tích hợp ROS 2 vào các ứng dụng nhúng.



Hình 2.22: Kiến trúc của uROS

Các thành phần màu xanh đậm được phát triển riêng cho micro-ROS. Các thành phần màu xanh nhạt được lấy từ ngăn xếp ROS 2 tiêu chuẩn.



Hình 2.23: minh họa uROS kết nối giữa CPU và MCU²

² Nguồn: micro-ROS architecture, đường dẫn:
<https://micro.ros.org/docs/overview/features/>

Chương 3. THIẾT KẾ VÀ THI CÔNG PHẦN CỨNG

3.1. Cấu trúc tổng quát hệ thống

Cấu trúc tổng quát gồm 2 phần: Embedded PC và Microcontroller. Ngoài ra, để truy cập máy tính nhúng từ xa trong quá trình robot vận hành, laptop sử dụng giao thức SSH để kết nối với máy tính nhúng để chạy và giám sát các Node ROS.

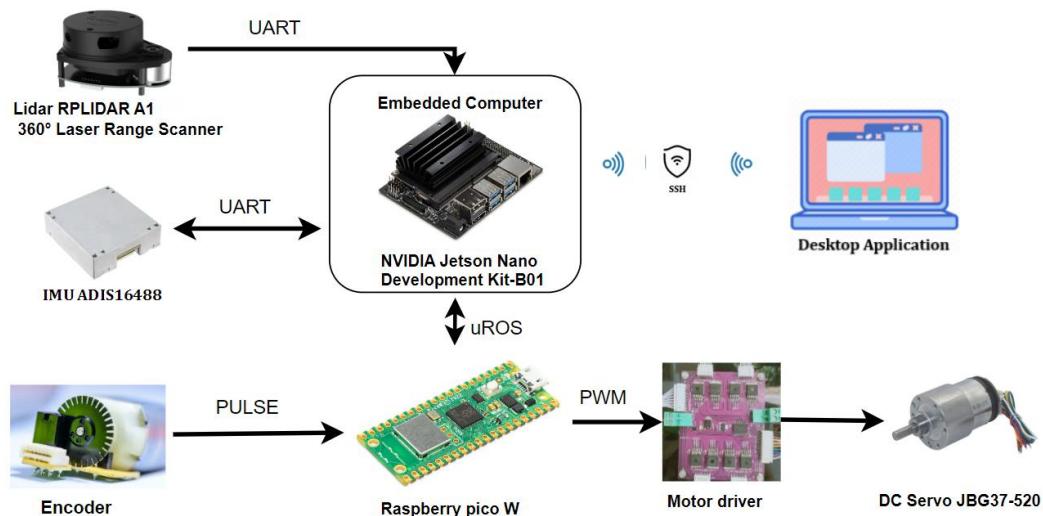
Embedded PC:

Sử dụng bộ kit phát triển NVIDIA Jetson Nano B01 là trung tâm điều khiển chính của robot. Nơi thu thập dữ liệu từ cảm biến IMU, Lidar và Encoder cũng như thực hiện các thuật toán điều khiển Stanley, PD Fuzzy và PID và giải thuật Iterative Closest Point. Các chương trình được viết trên nền tảng ROS2 sử dụng ngôn ngữ lập trình C/C++.

Về phần truyền thông, Jetson được kết nối với cảm biến IMU và Lidar bằng giao thức UART. Đồng thời Jetson sẽ nhận vận tốc thật của robot và truyền tín hiệu điều khiển động cơ với vi xử lý thông qua giao thức uROS.

Microcontroller:

Sử dụng vi điều khiển Raspberry Pico được xây dựng dựa trên chip RP-2040. Nơi đọc cảm biến Encoder, cấp tín hiệu điều khiển PWM cho Driver motor board và giao tiếp với máy tính nhúng thông qua giao thức uROS.



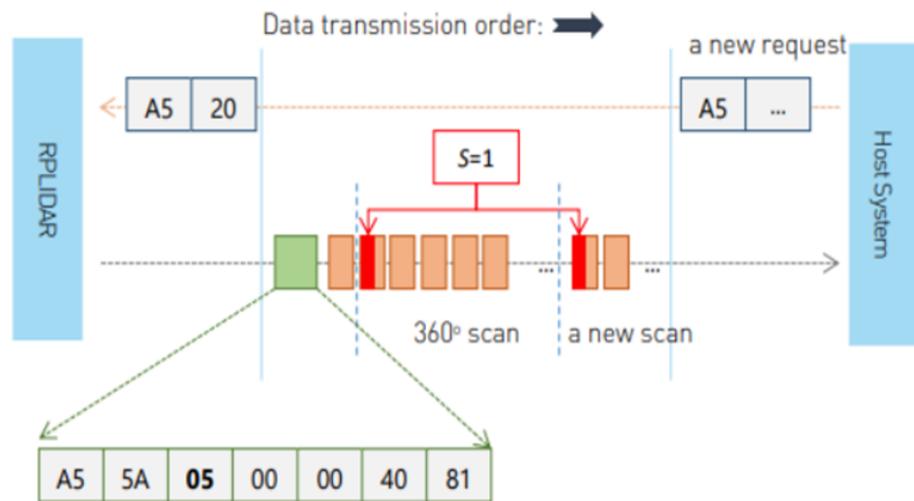
Hình 3.1: cấu trúc tổng quát hệ thống

3.1.1. RPLidar A1

Cảm biến Laser Radar RPLIDAR A1 12m của hãng SLAMTEC sử dụng giao tiếp UART, kết nối máy tính qua mạch chuyển USB-UART và phần mềm đi kèm. Dữ liệu truyền về sẽ là các point clouds. Các point cloud này được sử dụng để vẽ bản đồ 2D.

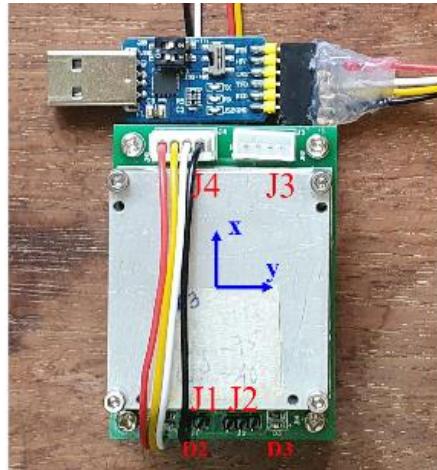


Hình 3.2: RPLidar A1



Hình 3.3: Các packet truyền khi Lidar quét 360°

3.1.2. Module cảm biến tốc độ và góc quay IMU



Hình 3.4: IMU sử dụng cảm biến MEMS ADIS16488

Module sử dụng cảm biến MEMS ADIS16488 là một hệ thống cảm biến quán tính hoàn chỉnh trong đó bao gồm 3 trực gyroscope (con quay hồi chuyển), 3 trực accelerometer (gia tốc kế), 3 trực đo magnetometer (từ trường kế) và một cảm biến đo áp lực (pressure sensor).

Gyroscope: Tầm đo $450^\circ/s$, Bias 6.250/h.

Accelerometer: Tầm đo $\pm 18g$, Bias 3.5mg.

Magnetometer: Tầm đo ± 2.5 gauss.

Thông số kỹ thuật	Mô tả
Nguồn cấp	5VDC, 0.4A
Sai số góc nghiêng tĩnh	0.1° (RMS)
Sai số góc nghiêng động	0.2° (RMS)
Sai số hướng tĩnh	0.1° (RMS)
Sai số hướng động	0.3° (RMS)
Độ phân giải đo tốc độ	$0.001^\circ / s$
Độ phân giải góc	$0.001^\circ / s$
Dải đo vị trí 3 trực:	(liên tục)

+ Roll	từ -180° tới 180°
+Pitch	từ -90° tới 90°
+Yaw	từ -180° tới 180°
Hoạt động đảm bảo ở dải tần số	0 tới 5Hz
Hoạt động đảm bảo ở dải nhiệt độ	từ $20^\circ C$ tới $85^\circ C$
Tần số cập nhật ngõ ra tối đa	500 Hz
Chuẩn truyền thông RS232	từ 115.2 tới 921.6 Kbps
Thời gian khởi động	30s
Độ trễ tín hiệu	<1ms
Kích thước	64x46x3 mm
Độ trôi góc nghiêng (roll, pitch) động liên tục	0.3° / 10 phút
Độ trôi góc hướng (yaw) tĩnh khi không sử dụng cảm biến từ trường	0.1° / 10 phút
Độ trôi góc hướng (yaw) động khi không sử dụng cảm biến từ trường	2° / 10 phút

Bảng 3.1: Thông số kỹ thuật của Module IMU

0x0A	roll	pitch	yaw	ω_x	ω_y	ω_z
1byte	7bytes	7bytes	7bytes	7bytes	7bytes	7bytes

ax	ay	az	0x0D
6bytes	6bytes	6bytes	1byte

Bảng 3.2: Frame truyền dữ liệu của IMU

Mỗi giá trị cách nhau bởi ký tự khoảng trắng (0x20).

Tổng số byte truyền: 71 (không từ trường), 89 (có từ trường).

Tổng số byte truyền có thể thay đổi theo yêu cầu lựa chọn các giá trị gửi lên.

3.1.3. Máy tính nhúng Jetson Nano B01

Các giải thuật và thuật toán điều khiển đều được thực hiện trên ROS, vì vậy cần có một máy tính chạy hệ điều hành này gắn trên robot. Bộ kit phát triển NVIDIA Jetson Nano là một máy tính nhúng nhỏ nhưng rất mạnh mẽ, cho khả năng tính toán cao, phù hợp với các ứng dụng nhúng. Máy tính nhúng có tích hợp nhiều cổng đọc được dữ liệu từ camera, lidar, USB UART, ... và sử dụng nguồn 5V – 4A.



Hình 3.5: NVIDIA Jetson Nano B01

3.1.4. Vi điều khiển Raspberry Pico

Raspberry Pico là mạch vi điều khiển hiệu năng cao, chi phí thấp được xây dựng dựa trên chip RP2040-chip vi điều khiển được thiết kế bởi chính Raspberry Pi. Nó có thể được lập trình lại dễ dàng qua USB từ Raspberry Pi hoặc máy tính khác sử dụng C/C++ hoặc MicroPython.

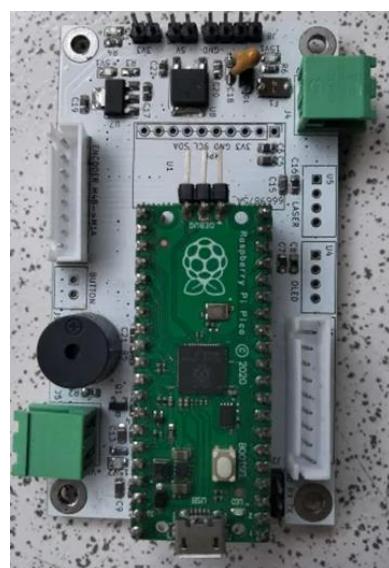


Hình 3.6: Raspberry Pico

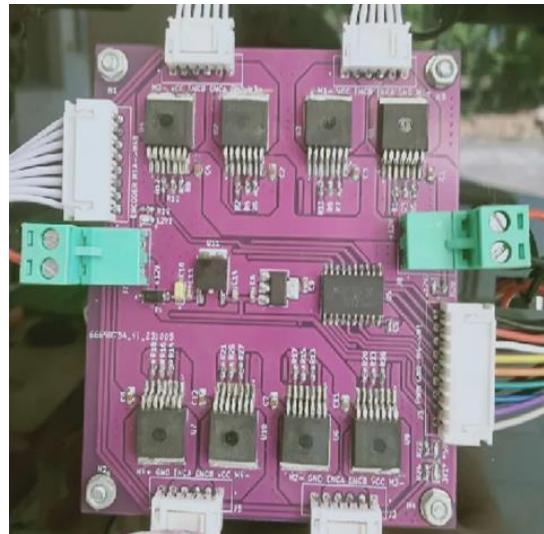
3.1.5. Driver motor, mainboard (tự thiết kế)

Board mạch cầu H dùng IC BTS7960, driver sử dụng nguồn 12V-4A, dùng để điều khiển 4 động cơ DC. Driver nhận đầu vào là 8 xung điều khiển PWM, đầu ra là 8 chân nguồn cấp cho 4 động cơ.

Mainboard chứa vi điều khiển Raspberry Pico, 2 module đọc adc, nguồn LDO, và các header để kết nối cảm biến. Sử dụng nguồn 15V-1A.



Hình 3.7: Mainboard



Hình 3.8: Driver motor

3.1.6. DC Servo JGB37-520 có Encoder

DC Servo JGB37-520 sử dụng nguồn 12V-1A (max) , 333RPM (không tải) và 250RPM (có tải).

Encoder có cảm biến từ trường Hall, có 2 kênh AB lệch nhau giúp xác định chiều quay và vận tốc của động cơ, đĩa Encoder trả ra 11 xung/1 kênh/ 1 vòng (nếu đo tín hiệu đồng thời của cả hai kênh sẽ thu được tổng 22 xung / 1 vòng quay của Encoder). Số xung Encoder mỗi kênh trên 1 vòng quay trực chính là $11 \times 30 = 330$ xung. Điện áp cấp cho Encoder hoạt động: 3.3~5VDC.



Hình 3.9: DC Servo JBG37-520

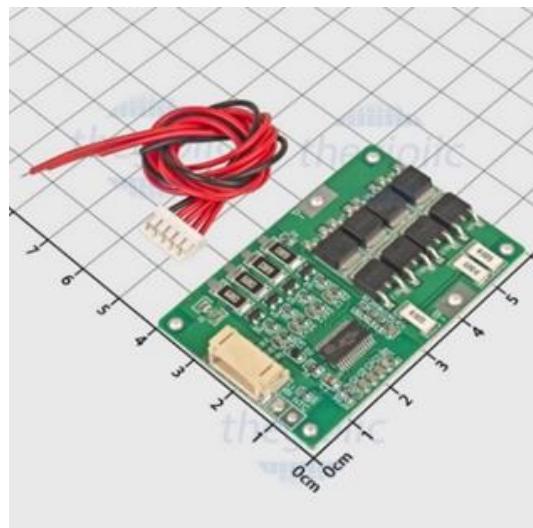
3.1.7. Pin sạc 18650 Li-ion và mạch sạc pin 18650 4S

Pin sạc Li-ion 18650 2000mAh 3.7V 3C thường được sử dụng để làm sạc dự phòng, cấp nguồn cho mạch điện hoặc các cấu trúc robot đơn giản.

JH-996040 8 MOS mạch sạc pin có điện áp sạc là 16.8V có thể chịu được dòng điện tức thời lên tới 40A, bảo vệ an toàn hiệu quả cho pin với thiết kế 4s.



Hình 3.10: Pin sạc Li-ion 18650



Hình 3.11: Mạch sạc pin 18650 4S

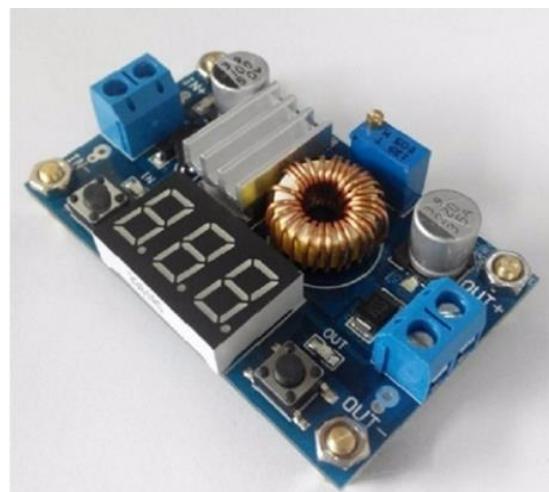
3.1.8. Mạch giảm áp LM1596S 3A và XL4015 5A

Mạch giảm áp DC-DC LM2596S 3A có kích thước nhỏ gọn có khả năng giảm áp từ 30VDC xuống còn 1.5VDC mà vẫn đạt hiệu suất cao (92%), thích hợp cho các ứng dụng chia nguồn, hạ áp,...

Mạch giảm áp XL4015 5A mạch có tích hợp đồng hồ Led và phím chức năng chọn hiển thị áp đầu vào (4-38VDC) và đầu ra (1.25-36VDC) để tiện theo dõi. Dễ dàng điều khiển điện áp bằng biến trở tinh chỉnh để thông số chính xác.



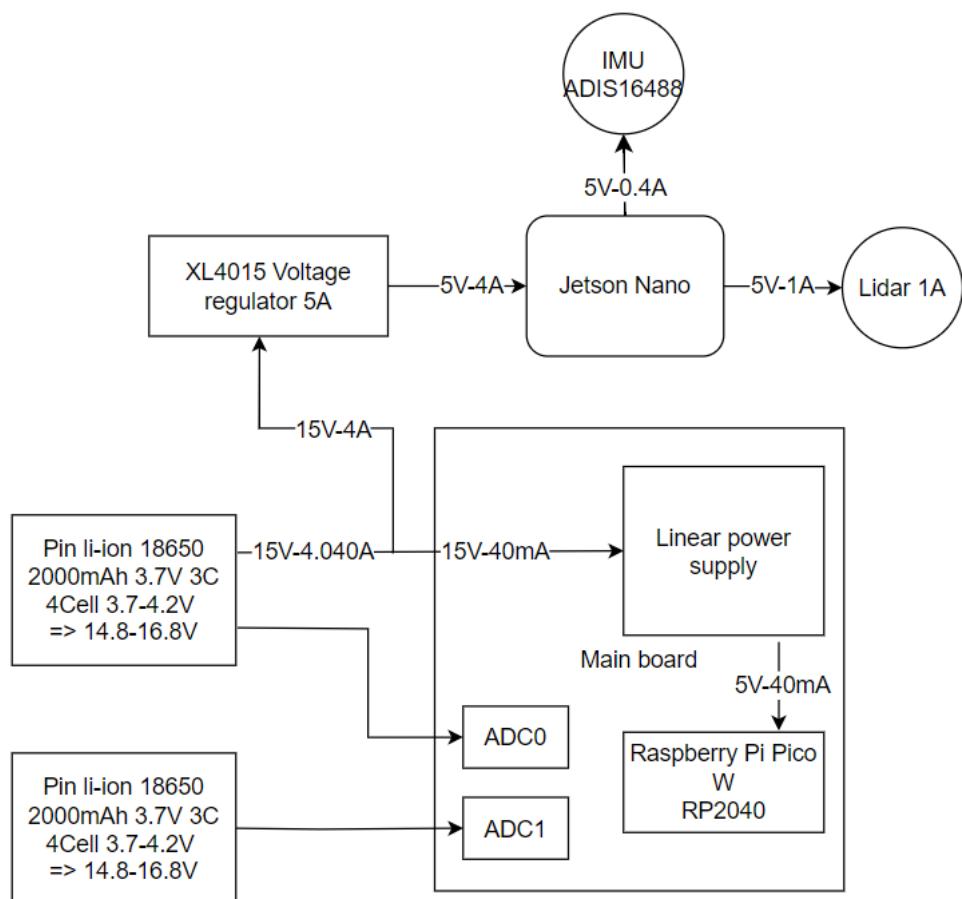
Hình 3.12: Mạch giảm áp DC LM2596S3



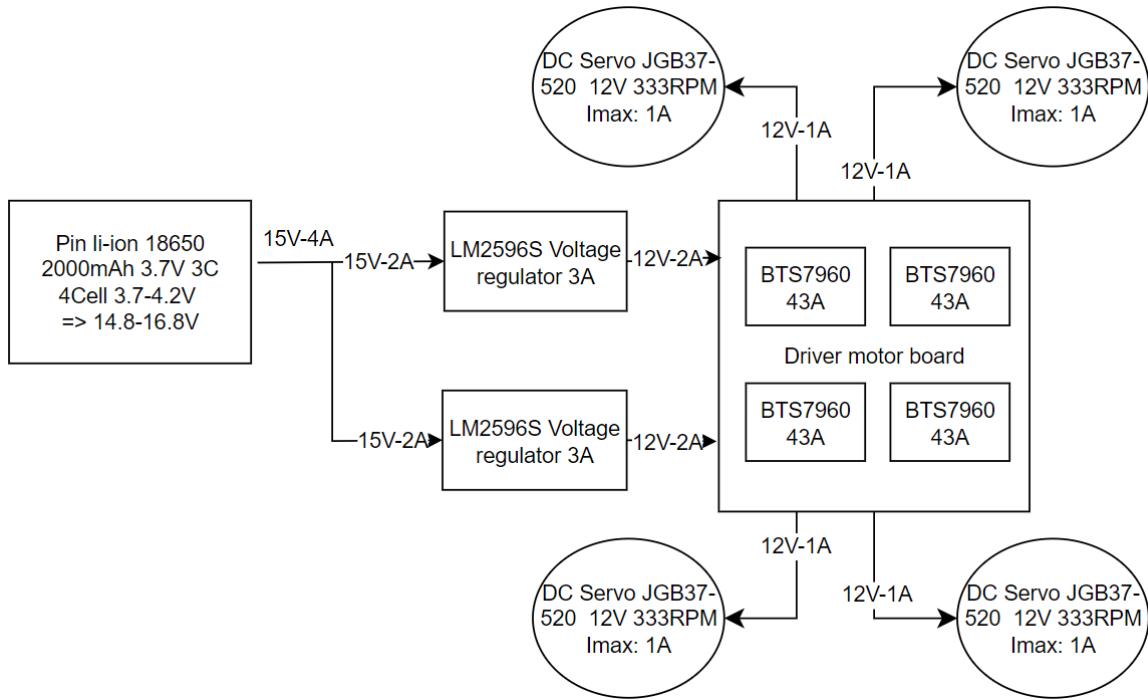
Hình 3.13: Mạch giảm áp DC XL4015 5A

3.2. Sơ đồ công suất phần cứng

Toàn bộ năng lượng của robot được cấp từ hai nguồn chính. Mỗi nguồn là 4 viên pin li-ion 2000mAh 3.7V 3C. Nhóm quyết định sử dụng 2 nguồn riêng vì muốn cách ly phần công suất khôi điều khiển và khôi động lực. Vì driver cầu H điều khiển động cơ với dòng ra liên tục lớn, mỗi động cơ có thể kéo dòng tối đa 1A ở tốc độ và moment cao nhất. Ngoài ra 4cell pin cấp điện áp từ 14.8-16.8V phù hợp với điện áp điều khiển động cơ 12V khi đã qua mạch giảm áp. Nguồn còn lại được cấp cho vi điều khiển, máy tính nhúng và các module cảm biến với tổng dòng tiêu thụ tối đa là 4A (khi jetson khởi động).



Hình 3.14: Sơ đồ khôi công suất phần điều khiển



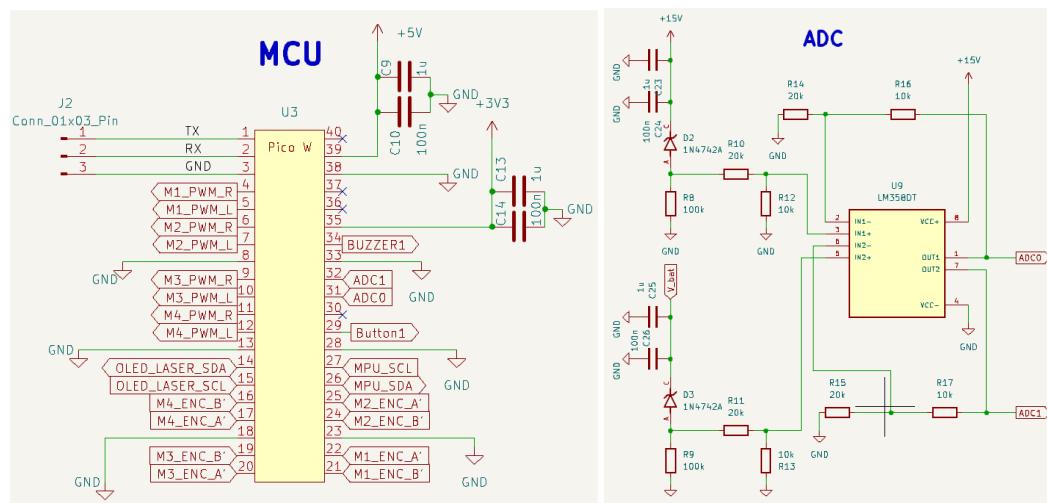
Hình 3.15: Sơ đồ khái niệm công suất phần động lực

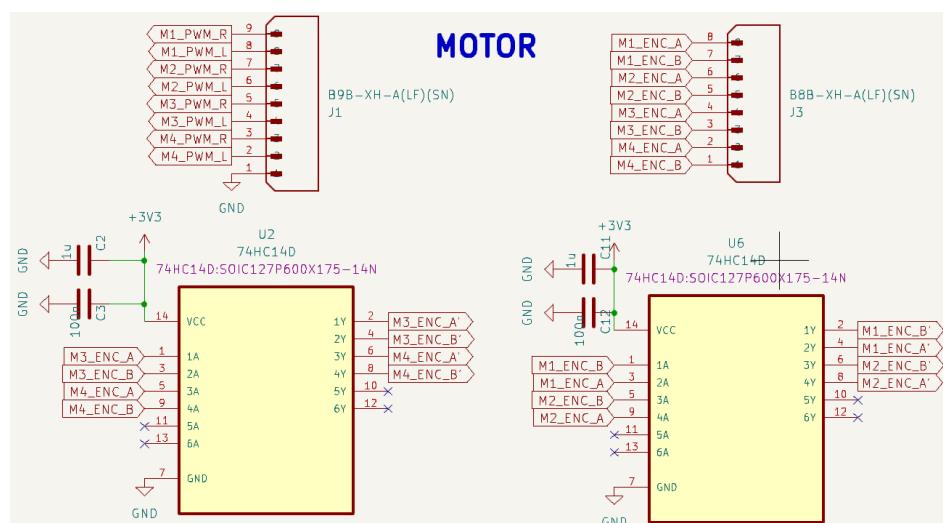
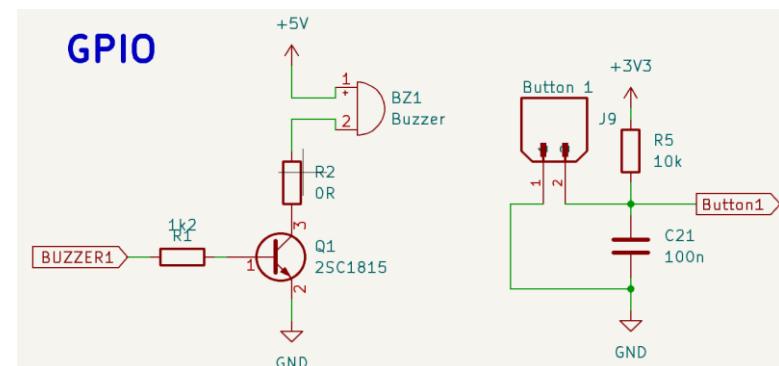
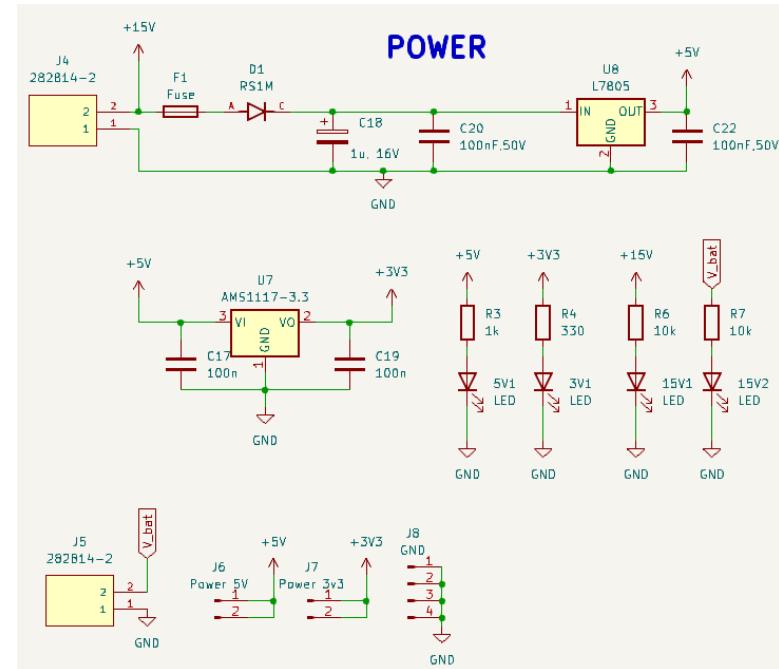
3.3. Thi công phần cứng

3.3.1. Thiết kế mạch điều khiển

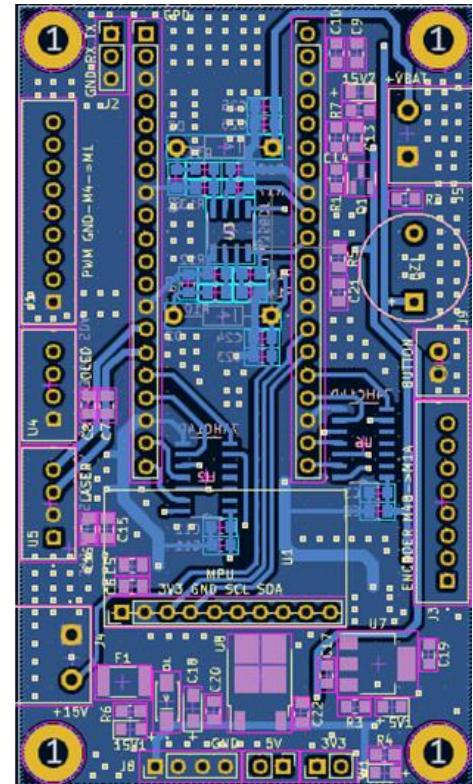
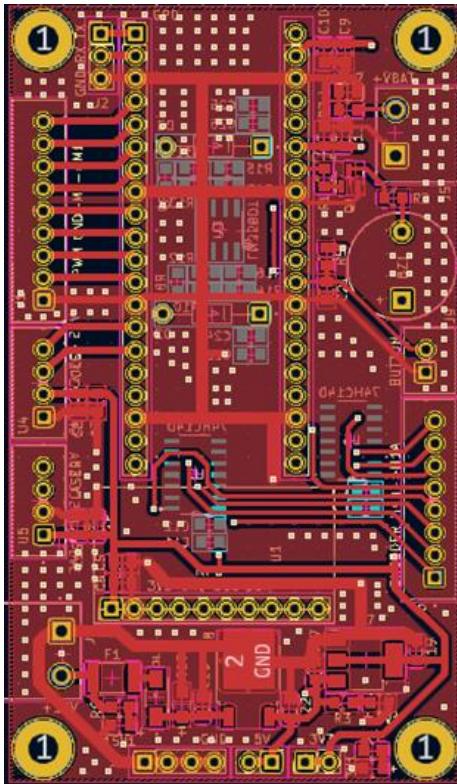
3.3.1.1. Mạch điều khiển chính chứa vi điều khiển

Mainboard dùng để kết nối vi điều khiển Raspberry Pico với các ngoại vi khác như mạch cầu H, encoder, ... Mặt khác, trên mainboard có khối nguồn LDO và 2 khối ADC để đo dung lượng pin của 2 nguồn cấp chính của robot.





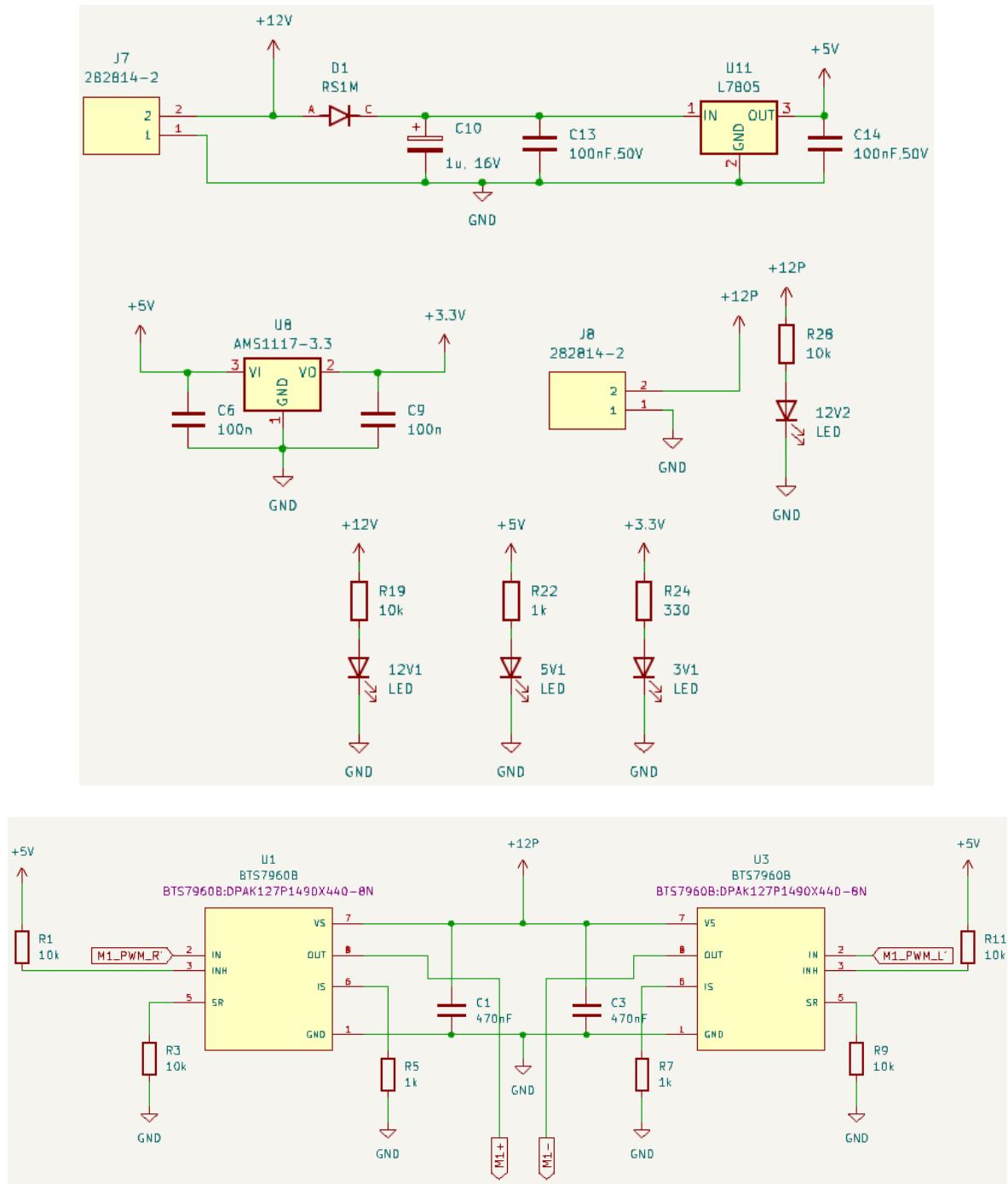
Hình 3.16: Bảng vẽ schematic của Mainboard

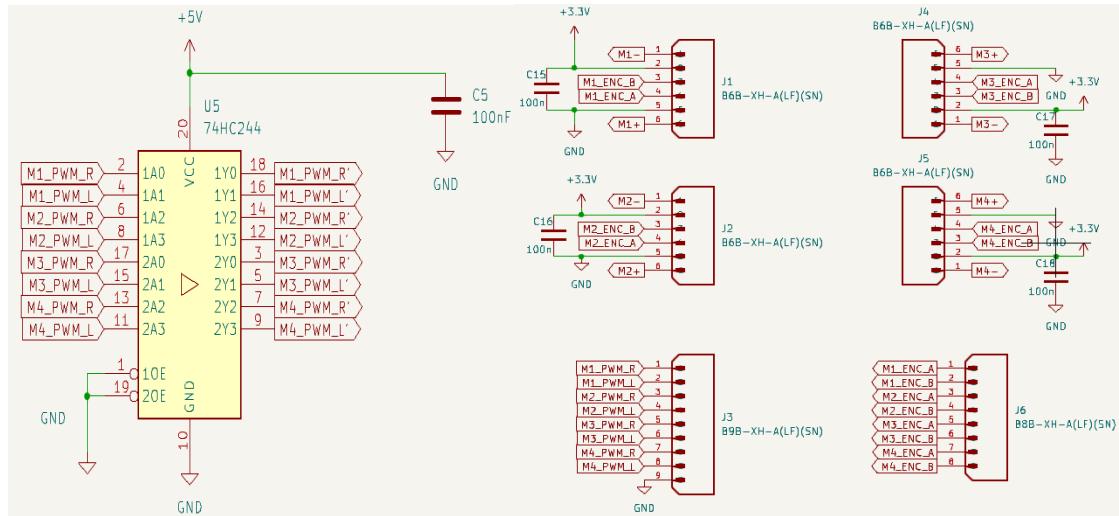


Hình 3.17: PCB top và bottom layer của Mainboard

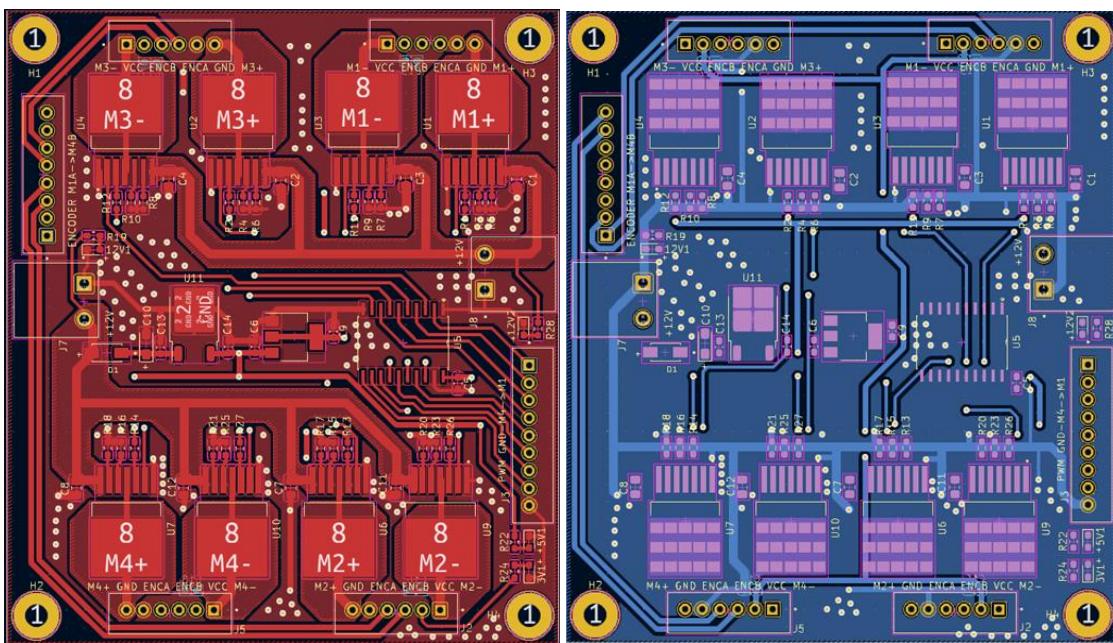
3.3.1.2. Mạch điều khiển động cơ

Motor driver board dùng BTS7960 có tần số PWM và dòng điện lên tới 25KHz và 43A. Board có 4 mạch cầu H dùng để điều khiển 4 động cơ. Mặt khác trên board còn chứa khối xử lý tín hiệu cho Encoder sử dụng IC 74HC244.





Hình 3.18: Bản vẽ schematic của Driver

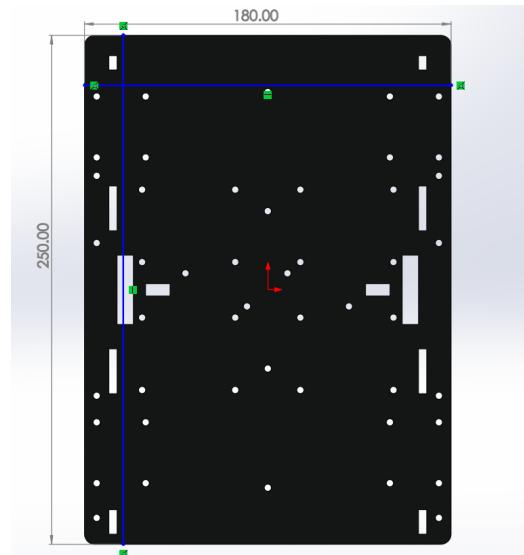


Hình 3.19: PCB top và bottom layer của Driver

3.3.2. Thiết kế mô hình

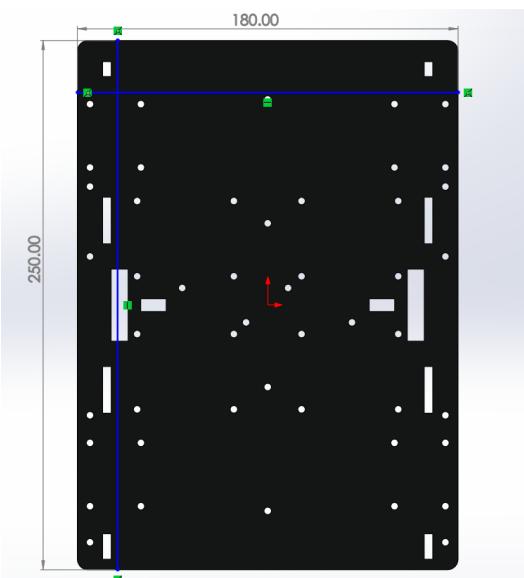
Nhóm sử dụng phần mềm vẽ 3D SolidWorks để vẽ mô hình robot, tự thiết kế các tầng robot đáp ứng các nhu cầu lắp ráp cần thiết cho các thành phần cứng được trình bày ở mục 2 chương 3. Robot sẽ được vẽ với 3 tầng nhằm thực hiện 3 nhiệm vụ riêng biệt với nhau gồm:

Tầng dưới: sẽ được sử dụng để lắp 4 motor có bánh xe và driver điều khiển, đồng thời đây là nơi cố định nguồn điện và chứa mạch sạc. Nhiệm vụ của tầng này là cung cấp nguồn, sạc pin và nhận tín hiệu điều khiển motor để di chuyển robot.



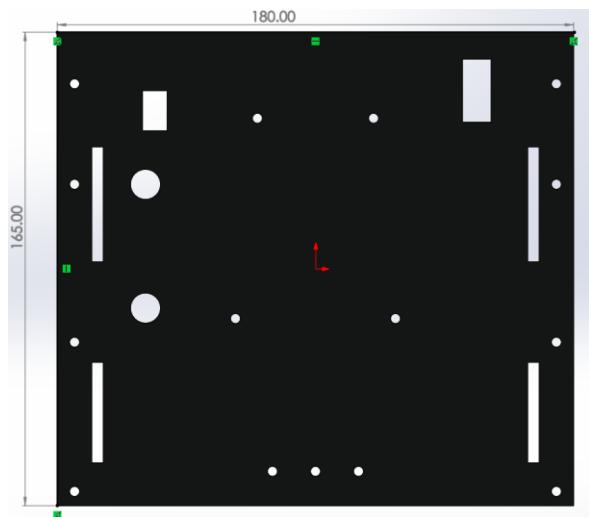
Hình 3.20: Kích thước tầng dưới (đơn vị mm)

Tầng giữa: sẽ lắp mainboard, máy tính nhúng Jetson Nano. Có thể nói nơi đây là bộ não của robot. Nhiệm vụ của tầng này là thu thập thông tin từ các cảm biến và thực hiện các giải thuật để đưa ra các tín hiệu điều khiển.



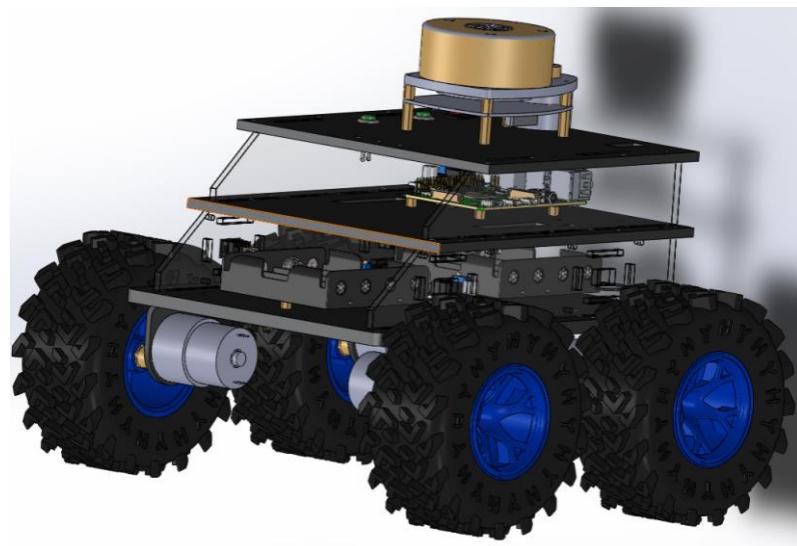
Hình 3.21: Kích thước tầng giữa (đơn vị mm)

Tầng trên: chứa Lidar và IMU, đây là nơi thu thập dữ liệu môi trường.



Hình 3.22: Kích thước tầng trên (đơn vị mm)

Sau quá trình tính toán kỹ lưỡng và vẽ nhiều mẫu thiết kế khác nhau, nhóm đã quyết định chọn thiết kế mô hình cuối cùng như hình dưới đây để có thể đáp ứng được các yêu cầu đặt ra, thực hiện các tác vụ mà robot cần làm.

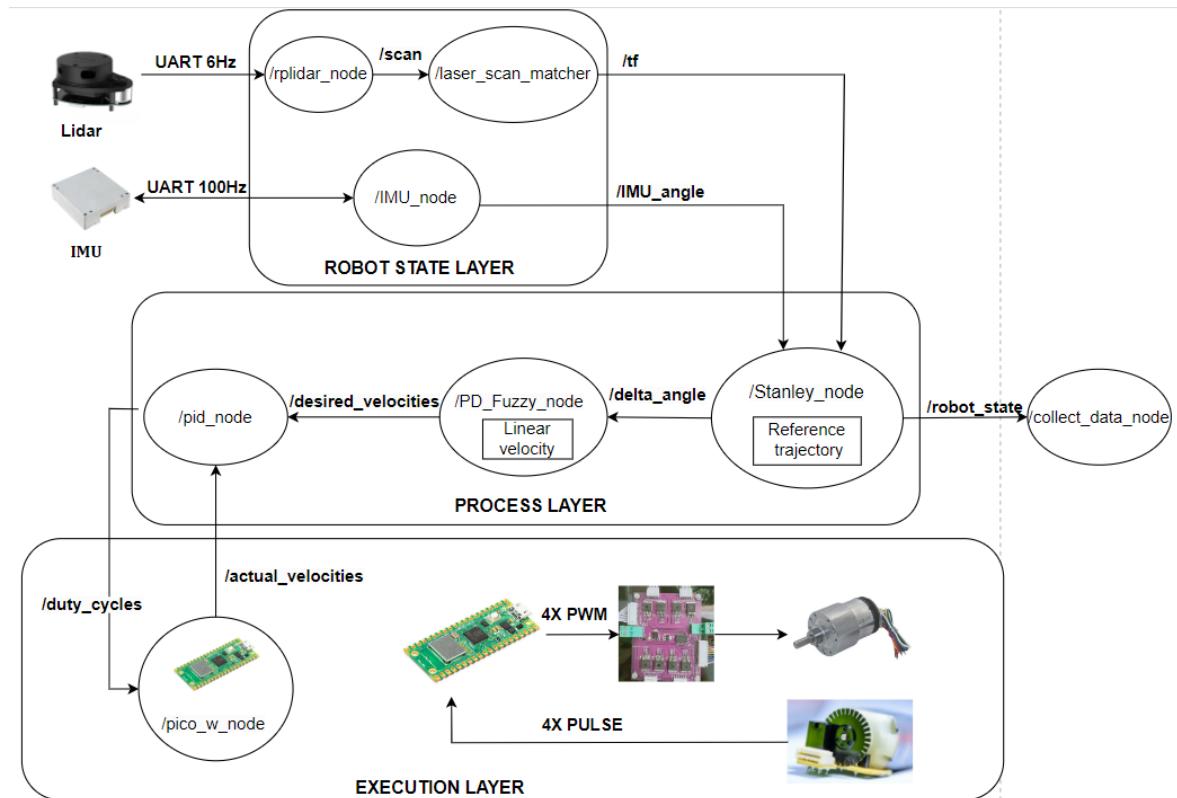


Hình 3.23: Mô hình robot thiết kế dùng SolidWorks

Chương 4. THIẾT KẾ HỆ THỐNG PHẦN MỀM

4.1. Cấu trúc phần mềm

Các node được lập trình và dữ liệu truyền giữa chúng được quản lý trong môi trường hệ điều hành ROS2:



Hình 4.1: Cấu trúc phần mềm của hệ thống

Tạo node tên /pico_w_node trên raspberry pico W và sử dụng phương thức uROS để subscribe đến topic /duty_cycles và publish topic /actual_velocities giao tiếp với máy tính nhúng:

```
[1704445823.081160] info | TermiosAgentLinux.cpp | init | running...
[1704445823.813064] info | Root.cpp | create_client | create
[1704445823.813217] info | SessionManager.hpp | establish_session | session established
[1704445823.892487] info | ProxyClient.cpp | create_participant | participant created
[1704445823.896388] info | ProxyClient.cpp | create_topic | topic created
[1704445823.898513] info | ProxyClient.cpp | create_publisher | publisher created
[1704445823.901254] info | ProxyClient.cpp | create_datawriter | datawriter created
[1704445823.904259] info | ProxyClient.cpp | create_topic | topic created
[1704445823.907053] info | ProxyClient.cpp | create_subscriber | subscriber created
[1704445823.909269] info | ProxyClient.cpp | create_datareader | datareader created
[1704445823.913064] info | Root.cpp | create | fd: 3
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, session_id: 0x81
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, address: 0
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, participant_id: 0x000(1)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, topic_id: 0x000(2), participant_id: 0x000(1)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, publisher_id: 0x000(3), participant_id: 0x000(1)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, topic_id: 0x001(2), participant_id: 0x000(1)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, subscriber_id: 0x000(4), participant_id: 0x000(1)
[1704445823.913217] info | SessionManager.hpp | establish_session | client key: 0x58C4376F, datareader_id: 0x000(6), subscriber_id: 0x000(4)
```

Hình 4.2: Chạy thành công chương trình tạo node trên raspberry pico W

Tên topic	Ý nghĩa	Tần số truyền dữ liệu trung bình
/scan	Chứa tập dữ liệu điểm của không gian xung quanh lidar quét được.	6.843Hz
/tf	Trả về vị trí x, y của robot trong không gian với gốc tọa độ là vị trí bắt đầu chạy thuật toán scan matching.	6.461Hz
/IMU_angle	Trả về góc heading của robot từ cảm biến IMU.	100.225Hz
/delta_angle	Sai số góc mà robot cần xoay để tới được hướng mong muốn.	10.011Hz
/desired_velocities	Vận tốc tuyến tính và vận tốc góc mong muốn của robot.	9.975Hz
/robot_state	Trả về các giá trị hướng, vị trí hiện tại và mong muốn của robot để thực hiện lưu trữ và thu thập dữ liệu.	6.647Hz
/duty_cycles	Giá trị duty cycles đầu ra của 4 bộ PID của mỗi động cơ để xe đạt được	9.870Hz

	vận tốc tuyến tính và vận tốc góc mong muốn.	
/actual_velocities	Vận tốc di chuyển thực tế của 4 bánh xe	10.005Hz

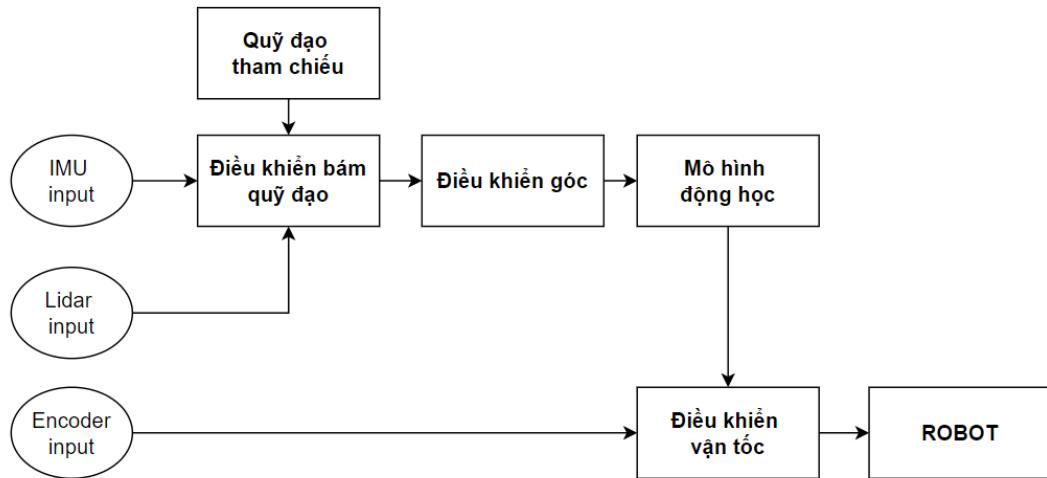
Bảng 4.1: Ý nghĩa của các topic và tần số truyền nhận dữ liệu của chúng

Tên node	Ý nghĩa hoạt động
/rplidar_node	Thực hiện các cấu hình để nhận được tập dữ liệu điểm từ lidar.
/laser_scan_matcher	Thực hiện thuật toán scan matching từ tập dữ liệu điểm của topic /scan để cho ra vị trí của robot trong không gian.
/IMU_node	Thực hiện các cấu hình để truyền lệnh xuống và nhận góc heading hiện tại của robot từ cảm biến IMU. Sau đó truyền đến /Stanley_node.
/pid_node	Nhận vận tốc tuyến tính và góc mong muốn từ /PD_Fuzzy_node để tính ra vận tốc mong muốn của 4 bánh xe. Sau đó kết hợp vận tốc thực tế của 4 bánh xe từ /pico_w_node để tính ra giá trị duty cycle tương ứng của mỗi động cơ và sau đó truyền lại cho /pico_w_node.
/PD_Fuzzy_node	Nhận sai số góc robot cần xoay từ

	/Stanley_node để tính toán và đưa ra giá trị vận tốc góc của robot. Sau đó truyền giá trị này cho /pid_node.
/Stanley_node	Nhận vị trí hiện tại của robot từ /laser_scan_matcher và góc heading từ /IMU_node để tính toán và đưa ra giá trị sai số góc mà robot cần xoay để đảm bảo robot bám được đúng hướng và quỹ đạo. Sau đó truyền giá trị góc này cho /PD_Fuzzy_node, đồng thời truyền vị trí, góc thực tế và mong muốn cho /collect_data_node.
/collect_data_node	Nhận giá trị từ topic /robot_state để lưu trữ dữ liệu trong tệp mở rộng .txt hoặc .csv
/pico_w_node	Nhận giá trị duty cycles từ /pid_node và gửi tín hiệu điều khiển tương ứng cho mỗi động cơ.

Bảng 4.2: Ý nghĩa của các node

Các khối thuật toán sử dụng trong hệ thống gồm:



Hình 4.3: Sơ đồ các khối thuật toán sử dụng trong hệ thống

Khối điều khiển bám quỹ đạo nhận vị trí hiện tại của robot trong không gian dựa vào tín hiệu của Lidar và hướng hiện tại của robot dựa vào góc heading từ cảm biến IMU, cùng với góc và vị trí mong muốn mà robot cần bám từ khối quỹ đạo tham chiếu. Đầu ra của khối điều khiển bám quỹ đạo là sai số góc mà robot cần phải xoay so với vị trí hiện tại của nó để đảm bảo bám được quỹ đạo và hướng mong muốn, góc này cũng chính là đầu vào của khối điều khiển góc. Mục tiêu của khối điều khiển góc chính là đưa ra giá trị vận tốc góc mà robot phải quay. Từ giá trị vận tốc góc và vận tốc tuyến tính, khối mô hình động học sẽ tính toán ra vận tốc của từng bánh xe cần chạy. Sau đó, khối điều khiển vận tốc sẽ nhận giá trị vận tốc của 4 bánh xe và vận tốc trả về từ các encoder để điều khiển vận tốc của động cơ.

4.2. Khối quỹ đạo tham chiếu

Quỹ đạo đường thẳng

Phương trình quỹ đạo đường thẳng:

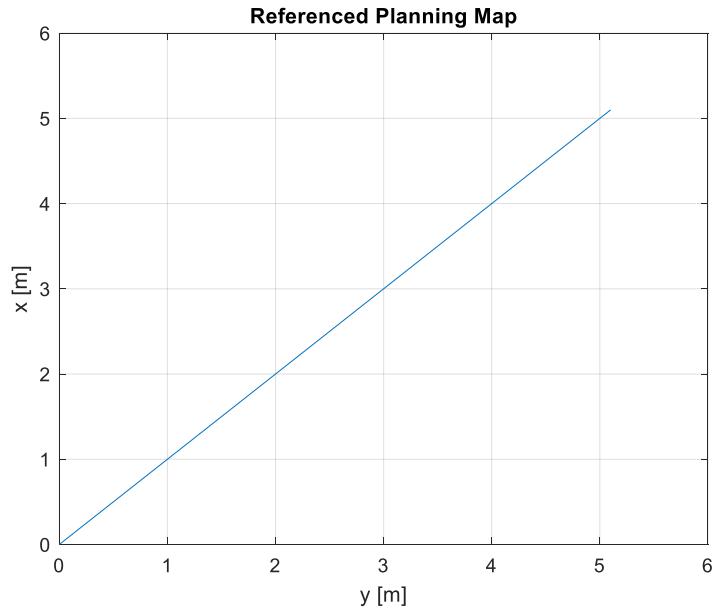
$$\begin{aligned}
 x(t) &= Ay(t) + B \\
 y(t) &= y(t-1) + STEP \\
 \theta(t) &= \arctan 2(y(t) - y(t-1), x(t) - x(t-1))
 \end{aligned}$$

Trong đó:

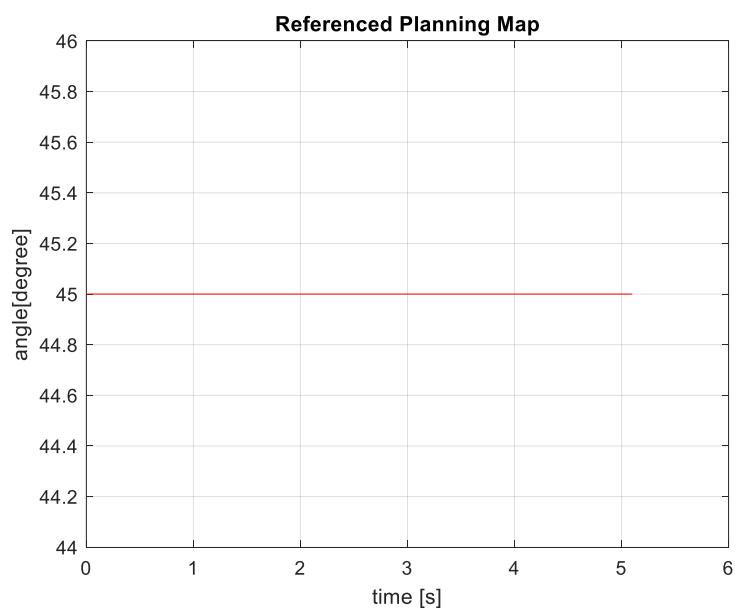
A, B: là hệ số của đường thẳng

STEP: là bước nhảy của x sau mỗi chu kì lấy mẫu

$\theta(t)$: góc của đường thẳng so với trục Ox



Hình 4.4: Quỹ đạo đường thẳng



Hình 4.5: Góc tham chiếu của quỹ đạo đường thẳng

Quỹ đạo đường tròn

Phương trình quỹ đạo đường tròn:

$$x(t) = X_0 + R \sin(\varphi t)$$

$$y(t) = Y_0 - R \cos(\varphi t)$$

$$\theta(t) = \arctan 2(Y_0 - y(t), X_0 - x(t)) - \frac{\pi}{2}$$

Trong đó:

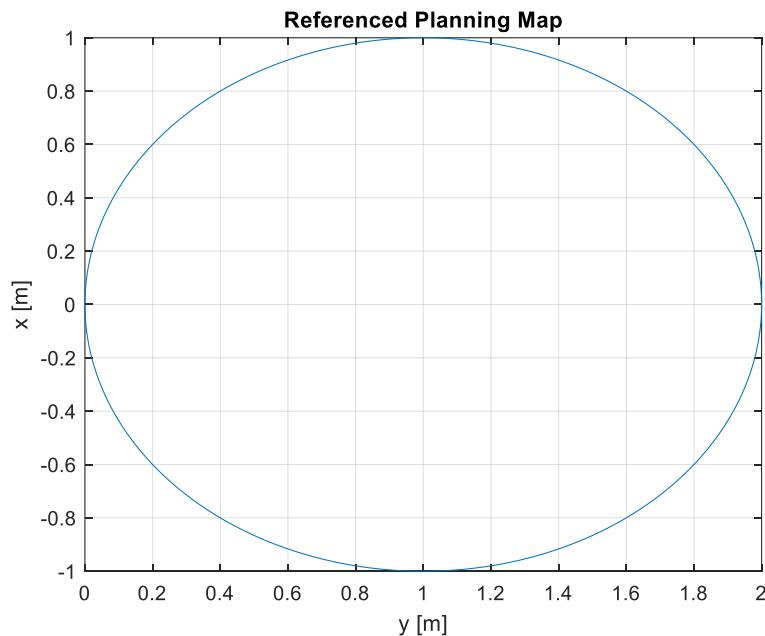
X_0, Y_0 : vị trí của tâm đường tròn

R : bán kính của đường tròn

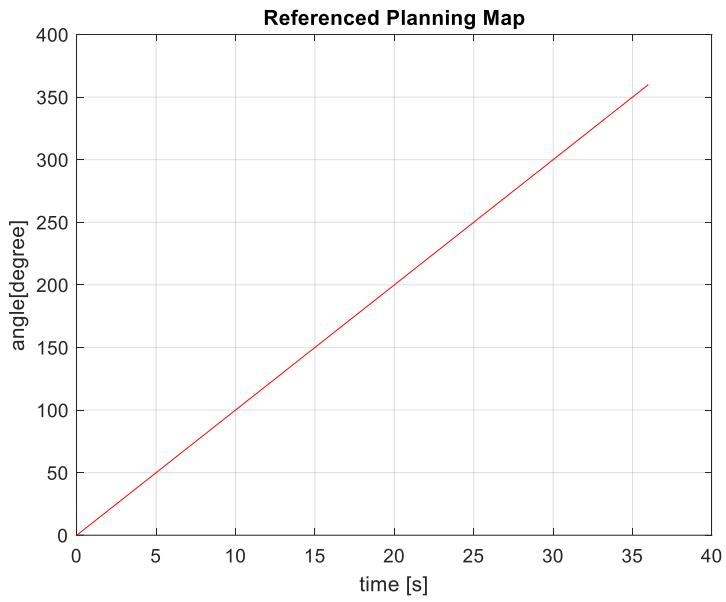
$\theta(t)$: góc hợp bởi tiếp tuyến của đường tròn tại từng điểm của quỹ đạo

và trục Ox

φ : là sự thay đổi góc của $\theta(t)$ sau mỗi chu kì lấy mẫu



Hình 4.6: Quỹ đạo đường tròn



Hình 4.7: Góc tham chiếu của quỹ đạo đường tròn

Quỹ đạo hình thang

Phương trình quỹ đạo hình thang:

$$x(t) = x(t-1) + STEP$$

$$y(t) = \begin{cases} 0; & \text{if } x(t) \leq A \\ \frac{x(t) - A}{B - A}; & \text{if } x(t) \geq A \text{ and } x(t) \leq B \\ 1; & \text{if } x(t) \geq B \text{ and } x(t) \leq C \\ \frac{D - x(t)}{D - C}; & \text{if } x(t) \geq C \text{ and } x(t) \leq D \\ 0; & \text{if } x(t) > D \end{cases}$$

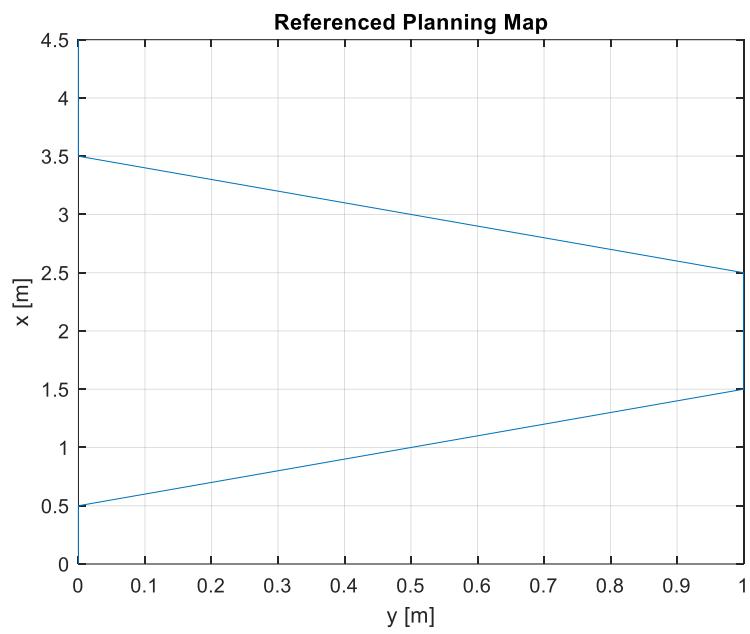
$$\theta(t) = \arctan 2(y(t) - y(t-1), x(t) - x(t-1))$$

Trong đó:

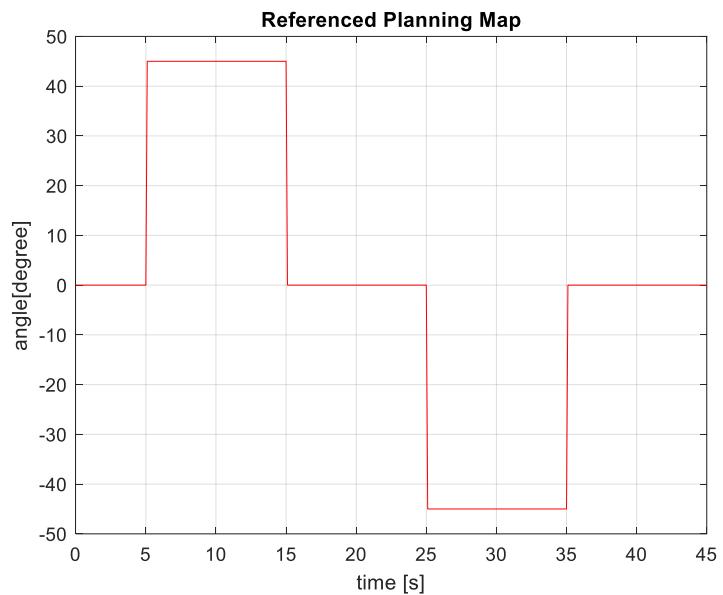
A, B, C, D: là các thông số thể hiện kích thước của hình thang

STEP: là bước nhảy của x sau mỗi chu kì lấy mẫu

$\theta(t)$: là góc hợp bởi trục Ox và hướng của đường thẳng hợp bởi điểm trước và sau mỗi chu kì lấy mẫu



Hình 4.8: Quỹ đạo hình thang



Hình 4.9: Góc tham chiếu của quỹ đạo hình thang

4.3. Khối điều khiển bám quỹ đạo

Bộ điều khiển bám quỹ đạo dựa trên giải thuật Stanley. Tần số điều khiển 10Hz.

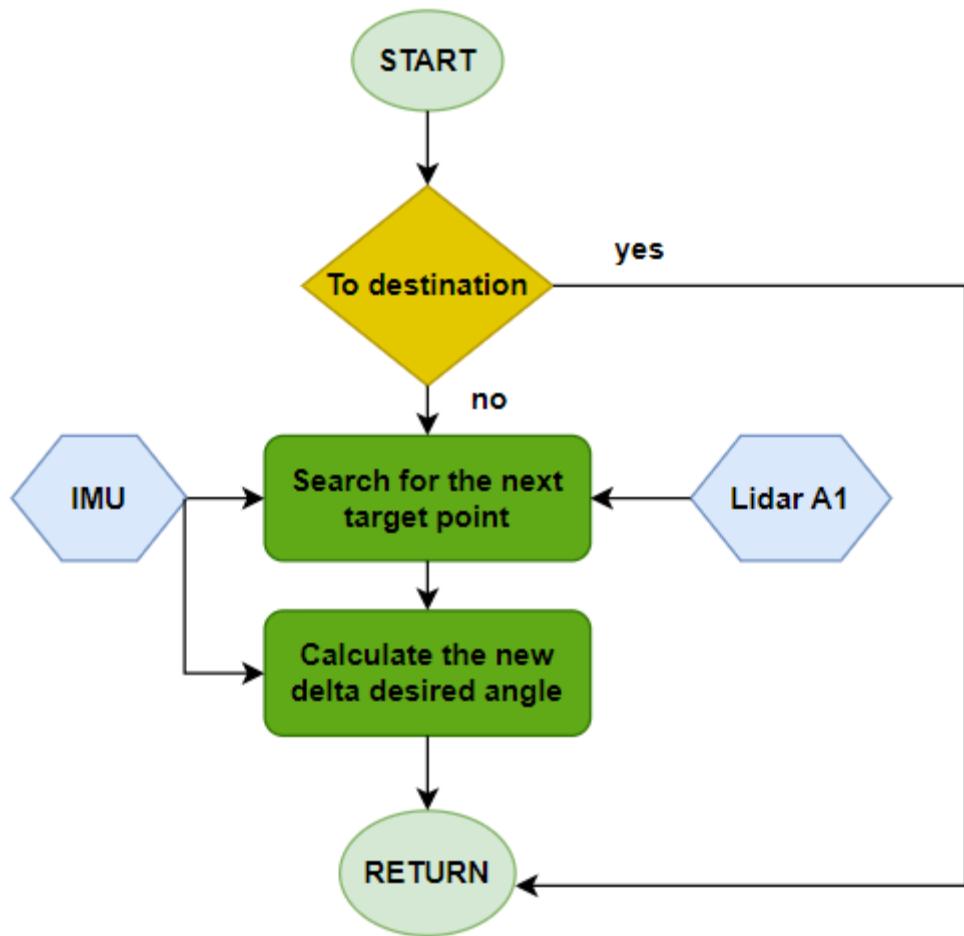
Input:

$\{Xref(j), Yref(j), \theta_{ref}(j)\}$ là tọa độ điểm đang tham chiếu đến và góc tiếp tuyến tại đó.

$\{X(t), Y(t), \theta(t)\}$ lần lượt là tọa độ hiện tại của robot lấy từ cảm biến Lidar A1 và góc heading của robot tính toán từ cảm biến IMU.

Output: $\Delta\theta$ là độ lệch góc giữa góc điều khiển δ và góc heading θ .

Sơ đồ giải thuật:



Hình 4.10: Sơ đồ khối thuật toán Stanley

Bước 1: Tính bán kính tới mục tiêu là khoảng cách giữa vị trí hiện tại của robot và điểm cuối cùng của quỹ đạo. Nếu bán kính này nhỏ hơn 0.05m thì xác định đã hoàn thành quỹ đạo, dừng robot. Nếu chưa đạt được điều kiện này thì tiếp tục thực hiện bước tính toán tiếp theo.

Bước 2: Xác định điểm tham chiếu tiếp theo trên quỹ đạo dựa trên vị trí hiện tại của robot. Giả sử i là điểm trên quỹ đạo mà robot đã tham chiếu đến trước đó, ta tính các khoảng cách từ vị trí hiện tại của robot với 10 (search offset) điểm tiếp theo (từ $i \rightarrow i + 10$) và chọn điểm có khoảng cách ngắn nhất để làm tham chiếu để tính các bước tiếp theo. Giả sử điểm tìm được là điểm j .

Bước 3: Tính $e(t)$ là crosstrack error được tính từ vị trí hiện tại của robot đến quỹ đạo tham chiếu (là khoảng cách từ vị trí hiện tại của robot đến điểm j ở bước 2):

$$e(t) = \sqrt{(X(t) - X_{ref}(j))^2 + (Y(t) - Y_{ref}(j))^2}$$

Tính $\Delta\theta$ là góc lệch giữa góc tiếp tuyến tại điểm tham chiếu và góc heading của robot trả về từ IMU, thành phần này giúp giữ góc lái của xe song song với quỹ đạo

$$\Delta\theta = \theta_{ref}(j) - \theta(t)$$

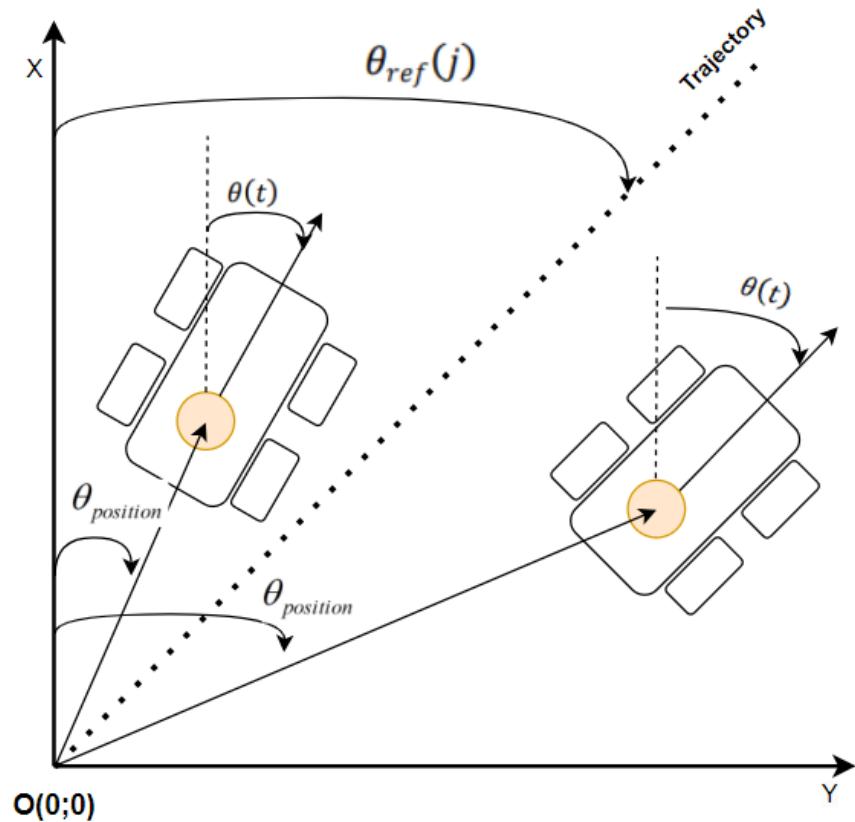
Tính θ_d là góc lệch thêm vào so với $\Delta\theta$ để đẩy xe vào quỹ đạo thay vì chạy song song với nó. Trong đó k là hệ số khuếch đại cho $e(t)$, k lớn làm tăng góc lệch đưa robot vào quỹ đạo nhanh nhưng dễ gây vọt lỗ và kéo theo dao động. Còn k_{soft} là hệ số điều chỉnh được thêm vào mẫu số, cho phép khả năng điều khiển góc ở vận tốc thấp, vì khi vận tốc dần về 0 thì góc thay đổi là rất lớn, tuy nhiên k_{soft} lớn sẽ làm giảm góc lệch cần thiết làm cho robot chậm tiến vào quỹ đạo.

$$\theta_d = \arctan 2\left(\frac{k * e(t)}{k_{soft} + v(t)}\right)$$

Do đó, kết hợp thêm θ_d ta có $\Delta\theta$ là độ biến thiên góc điều khiển cần thiết truyền vào bộ điều khiển PD Fuzzy để đưa robot bám theo quỹ đạo mà vẫn có hướng mong muốn, $\Delta\theta$ có phạm vi $[-180, 180]$:

$$\Delta\theta = \begin{cases} \theta_{ref}(j) - \theta(t) + \arctan 2\left(\frac{k * e(t)}{k_{soft} + v(t)}\right); \theta_{position} < \theta_{ref}(j) \\ \theta_{ref}(j) - \theta(t) - \arctan 2\left(\frac{k * e(t)}{k_{soft} + v(t)}\right); \theta_{position} > \theta_{ref}(j) \end{cases}$$

Xét dấu của θ_d trong biểu thức của $\Delta\theta$:



Hình 4.11: Hình vẽ minh họa xét dấu của $\Delta\theta$

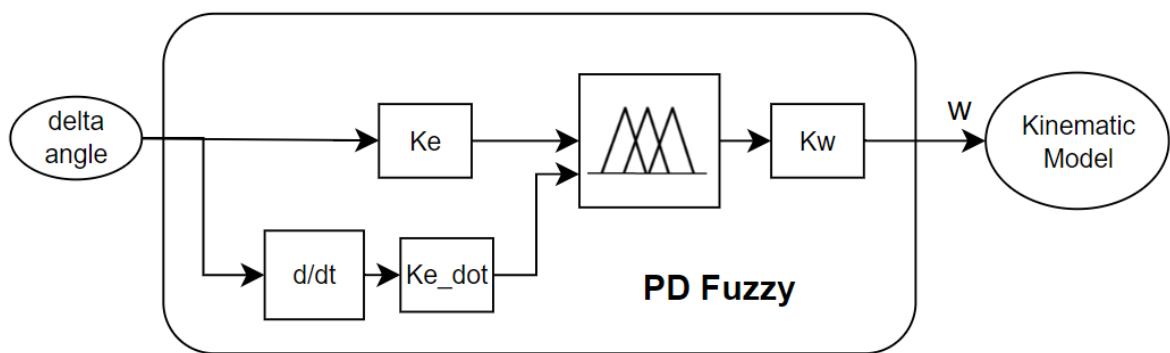
Trong đó: tại mỗi vị trí của robot trong không gian, ta xác định được hướng của vị trí robot như sau:

$$\theta_{position} = \arctan 2(Y(t); X(t))$$

4.4. Khối điều khiển góc

Khối điều khiển góc sử dụng bộ điều khiển PD Fuzzy. Mục tiêu của khối này là tính toán ra giá trị vận tốc góc đặt ω cho robot dựa vào góc lệch (giữa góc điều khiển δ và góc heading θ) được tính từ khối điều khiển bám quỹ đạo. Nói theo cách khác, input và output của khối này lần lượt là $\Delta\theta, \omega$.

Do đối tượng là robot có khâu tích phân lý tưởng nên chọn bộ điều khiển PD Fuzzy.



Hình 4.12: Sơ đồ khối giải thuật bộ điều khiển PD Fuzzy

Thiết kế bộ điều khiển PD Fuzzy:

-Input:

e : sai số góc (độ lệch giữa góc δ và θ) (rad)

\bar{e} : tốc độ biến thiên sai số góc (rad/s)

-Output:

ω : giá trị vận tốc góc reference của robot (rad/s)

Bước 1: Xác định tầm giá trị cho biến vào, ra và chuẩn hóa các biến vào, ra về miền $[-1,1]$

Xác định tầm giá trị của e :

Ta có: $0 \leq \delta \leq 2\pi$ và $0 \leq \theta \leq 2\pi$

Suy ra: $-2\pi \leq e \leq 2\pi$ (rad)

Chọn hệ số: $K_e = \frac{1}{2\pi}, \bar{e}(k) = K_e e(k)$

Xác định tầm giá trị của \bar{e} : tốc độ biến thiên sai số góc phụ thuộc vào tốc độ quay của động cơ, thực nghiệm cho thấy với mỗi chu kỳ lấy mẫu 100ms

$$\text{Ta có: } -\frac{\pi}{10} \leq \bar{e} \leq \frac{\pi}{10} \text{ (rad/s)}$$

$$\text{Chọn hệ số: } K_{\dot{e}} = \frac{\pi}{10}, \quad \bar{e}(k) = K_{\dot{e}} \dot{e}(k)$$

Xác định tầm giá trị tốc độ góc của robot: $-17 \leq \omega \leq 17$ (rad/s)

$$\text{Chọn hệ số: } K_{\omega} = 17$$

Dựa vào thực nghiệm các hệ số được tinh chỉnh và có giá trị như sau:

$$K_e = 0.159, \quad K_{\dot{e}} = 3.123, \quad K_{\omega} = 17$$

Bước 2: Tính toán giá trị đầu vào cho bộ điều khiển Fuzzy

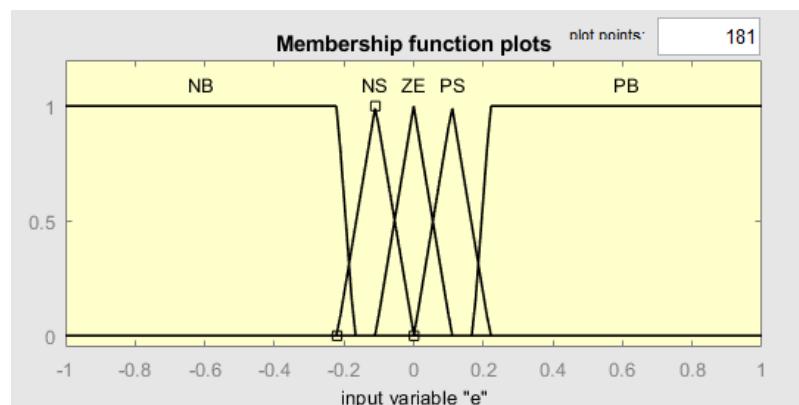
$$P_{Part} = K_e e(k)$$

$$D_{Part} = K_{\dot{e}} \frac{e(k) - e(k-2)}{T_s}$$

Giá trị P_{Part} và D_{Part} được cho qua khâu bão hòa để đảm bảo nằm trong miền giá trị $[-1;1]$.

Bước 3: Định nghĩa các giá trị ngôn ngữ và định lượng các giá trị ngôn ngữ bằng tập mờ:

Chọn 5 giá trị ngôn ngữ cho biến e : NB, NS, ZE, PS, PB

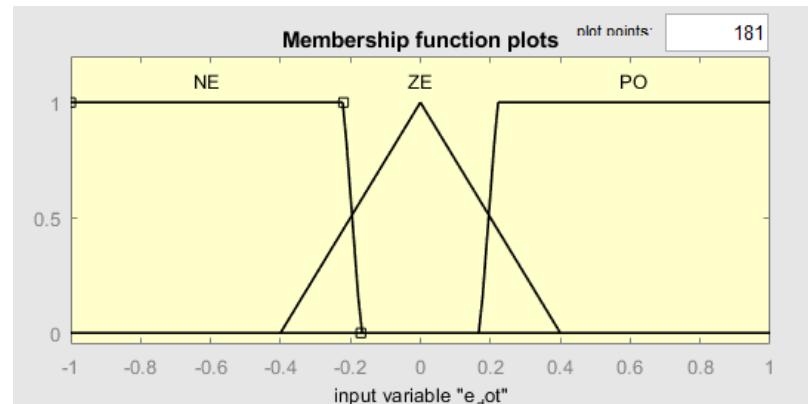


Hình 4.13: Tập mờ cho biến ngữ vào e

Định lượng giá trị cho biến e

$$\begin{cases} \mu_{NB}(e) = \text{trapf}(-2, -1, -0.22, -0.17) \\ \mu_{NS}(e) = \text{trimf}(-0.22, -0.11, 0.001) \\ \mu_{ZE}(e) = \text{trimf}(-0.11, 0, 0.11) \\ \mu_{PS}(e) = \text{trimf}(0.001, 0.11, 0.22) \\ \mu_{PB}(e) = \text{trapf}(0.17, 0.22, 1.2) \end{cases}$$

Chọn 3 giá trị ngôn ngữ cho biến \dot{e} : NE, ZE, PO

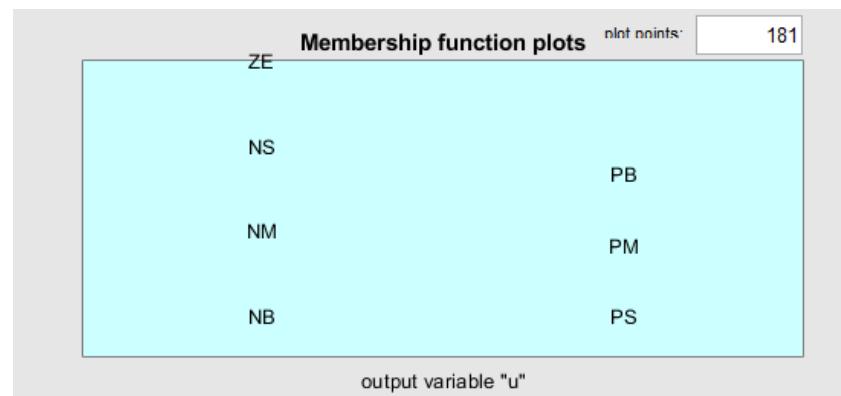


Hình 4.14: Tập mờ cho biến ngữ vào \dot{e}

Định lượng giá trị cho biến \dot{e}

$$\begin{cases} \mu_{NE}(\dot{e}) = \text{trapf}(-2, -1, -0.22, -0.17) \\ \mu_{ZE}(\dot{e}) = \text{trimf}(-0.4, 0, 0.4) \\ \mu_{PO}(\dot{e}) = \text{trapf}(0.17, 0.22, 1.2) \end{cases}$$

Chọn 7 giá trị ngôn ngữ cho biến u : NB, NM, NS, ZE, PS, PM, PB



Hình 4.15: Tập mờ cho biến ngữ ra u

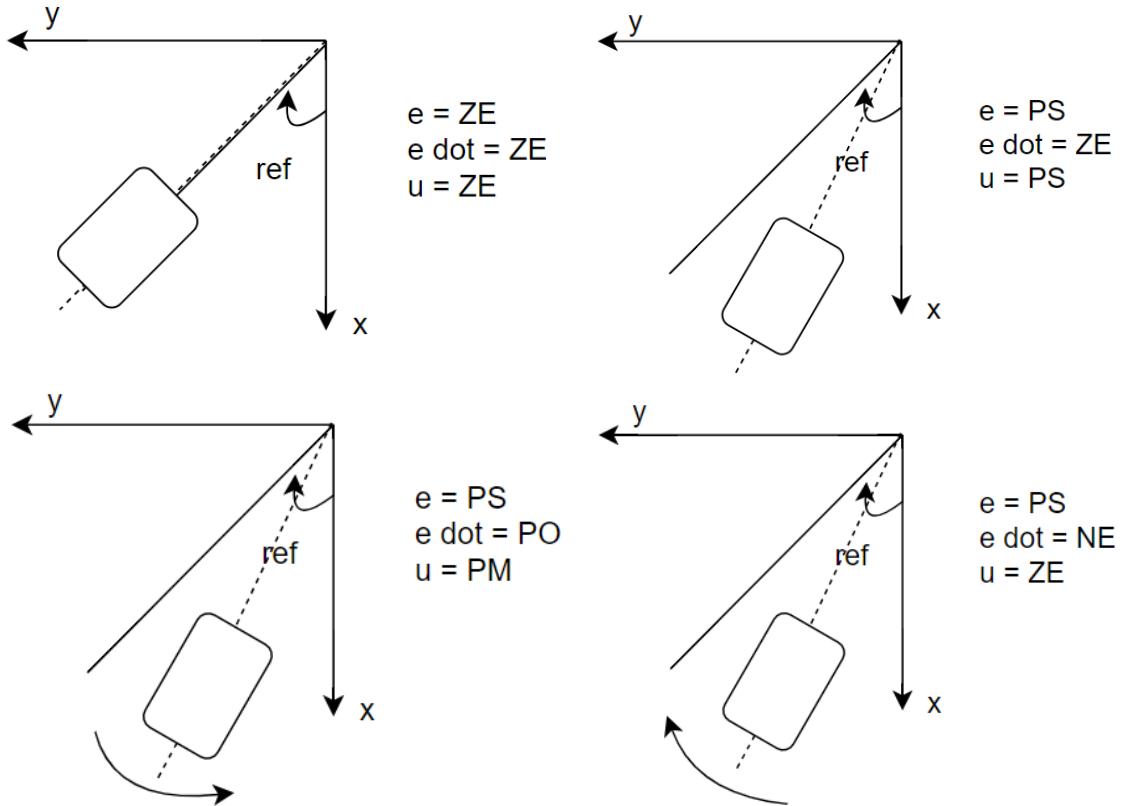
Định lượng cho biến ngõ ra u

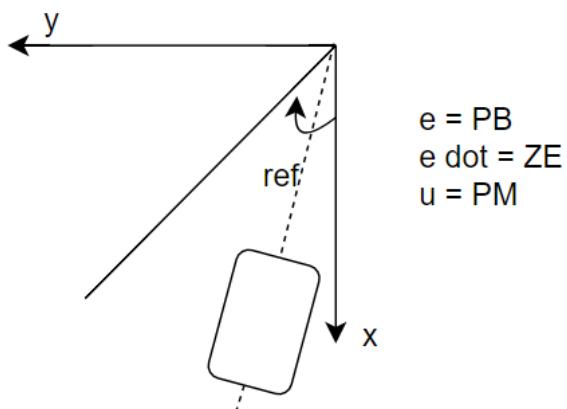
$$\begin{cases} \mu_{NB}(u) = -0.95 \\ \mu_{NM}(u) = -0.75 \\ \mu_{NS}(u) = -0.5 \\ \mu_{ZE}(u) = 0 \\ \mu_{PS}(u) = 0.5 \\ \mu_{PM}(u) = 0.75 \\ \mu_{PB}(u) = 0.95 \end{cases}$$

Các giá trị biến ngôn ngữ được chọn dựa trên kinh nghiệm điều khiển

Các hàm thành viên của từng biến ngôn ngữ được chọn và điều chỉnh thông số dựa trên thực tế để có được đáp ứng tốt.

Bước 4: Xây dựng hệ quy tắc mờ





Hình 4.16: Minh họa một số quy tắc điển hình

Áp dụng tính liên tục của hệ mờ và tính đối xứng ta có hệ các quy tắc mờ đầy đủ.

u		e				
		NB	NS	ZE	PS	PB
\dot{e}	NE	NB	NM	NS	ZE	PS
	ZE	NM	NS	ZE	PS	PM
	PO	NS	ZE	PS	PM	PB

Bảng 4.3: Các qui tắc mờ

Bước 5: Giải mờ

Sử dụng phương pháp suy diễn Max-Min và phương pháp giải mờ trung bình có trọng số.

Ta có ngõ ra của bộ Fuzzy được tính như sau:

$$u = \frac{\sum_k y_k \beta_k}{\sum_k \beta_k}$$

Từ đó ta tính được tốc độ góc reference của robot:

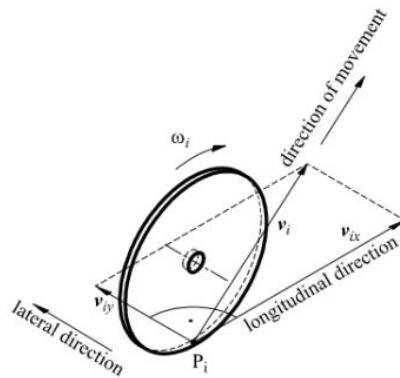
$$\omega = K_\omega u$$

Nếu $\omega > 0$ quay phải, $\omega < 0$ quay trái.

Giá trị của ω kết hợp với v cho qua khối **Kinematic Model** ta được vận tốc quay phải và trái của robot.

4.5. Khối mô hình động học của robot

Mục tiêu của khối này là tìm ra mối liên hệ giữa vận tốc của robot và vận tốc của từng bánh xe.



Hình 4.17: Vận tốc của một bánh xe³

Giả sử bỏ qua bất kỳ sự trượt dọc của bánh xe và bề mặt tiếp xúc ta có mối liên hệ sau:

$$v_{ix} = r_i \omega_i$$

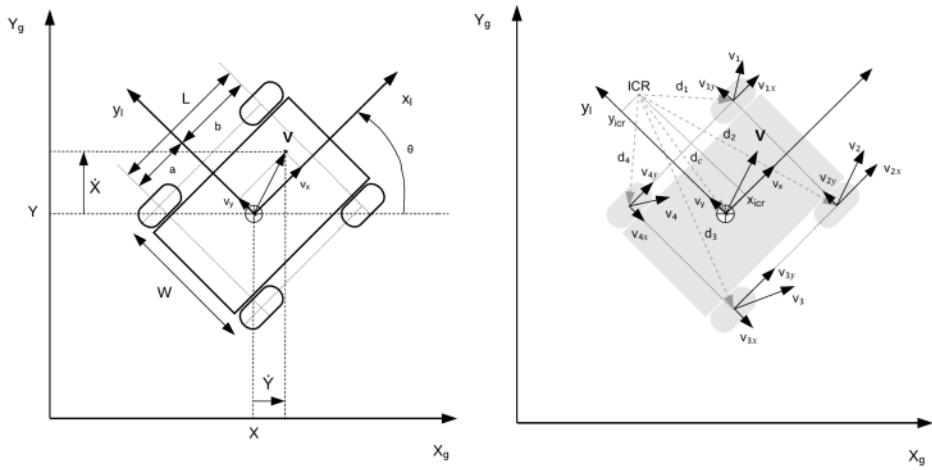
Trong đó:

v_{ix} : là thành phần vận tốc dọc của vector vận tốc vi tương ứng với bánh xe thứ i.

r_i : bán kính của bánh xe thứ i

ω_i : vận tốc góc của bánh xe thứ i

³ Modelling and Control of a Skid-Steering Mobile Robot for Indoor Trajectory Applications



Hình 4.18: Mô hình mô tả động học của robot⁴

Ta có mối liên hệ giữa tâm quay tức thời ICR và các thành phần vận tốc đối với local frame:

$$\frac{\|v_i\|}{\|d_i\|} = \frac{\|V\|}{\|d_c\|} = |\omega_z|$$

Tọa độ của tâm quay tức thời đối với local frame:

$$ICR(x_{ICR}, y_{ICR}) = (-d_{xc}, -d_{yc})$$

Mặc khác ta có:

$$\begin{aligned} d_{1y} &= d_{4y} = d_{cy} + \frac{W}{2} \\ d_{2y} &= d_{3y} = d_{cy} - \frac{W}{2} \end{aligned}$$

Từ các phương trình trên ta được mối liên hệ giữa vận tốc 4 bánh xe và vận tốc của robot:

$$v_L = v_{1x} = v_{4x}$$

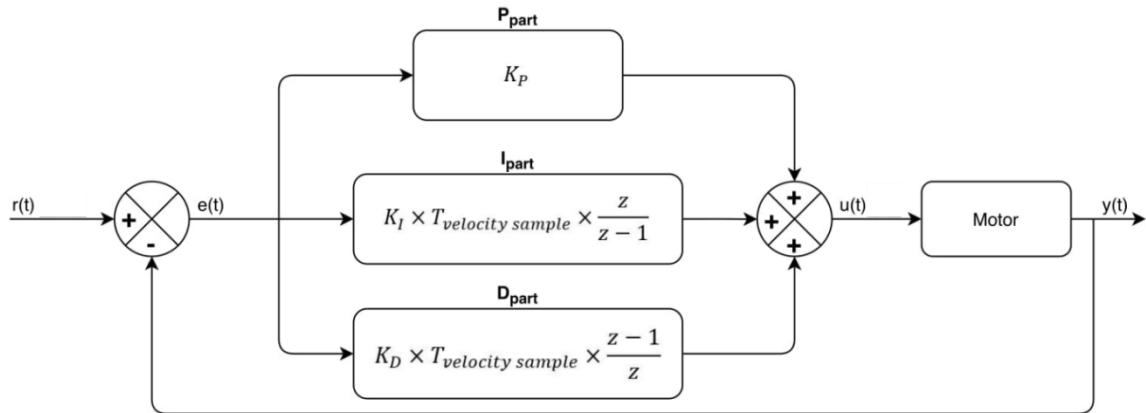
$$v_R = v_{2x} = v_{3x}$$

$$\begin{bmatrix} v_L \\ v_R \end{bmatrix} = \begin{bmatrix} 1 & -\frac{W}{2} \\ 1 & \frac{W}{2} \end{bmatrix} \begin{bmatrix} v_x \\ \omega_z \end{bmatrix}$$

⁴ Modelling and Control of a Skid-Steering Mobile Robot for Indoor Trajectory Applications

4.6. Khối điều khiển vận tốc

Khối điều khiển vận tốc sử dụng bộ điều khiển PID. Ta thấy từ vận tốc đặt cho từng bánh nhận được từ khối động học, chúng ta cần điều khiển vận tốc động cơ sử dụng bộ điều khiển PID rời rạc, với thời gian lấy mẫu 100 ms.



Hình 4.19: Sơ đồ khối giải thuật bộ điều khiển PID

Cách tinh chỉnh các hệ số K_P , K_I , K_D

Cho K_I , K_D bằng 0, tăng dần K_P đến khi hệ thống đạt giá trị xác lập và có thể có vọt lố.

Tăng dần K_I và tinh chỉnh K_P đến khi hệ thống đạt giá trị xác lập tại setpoint (sai số bằng 0).

Tinh chỉnh K_D để giảm độ vọt lố và chỉnh K_P để có được thời gian xác lập mong muốn.

Bước 1: Tính toán vận tốc từng bánh dựa vào tín hiệu đọc được từ Encoder.

Vận tốc tại thời điểm lấy mẫu của động cơ được tính dựa trên số xung trên một vòng của Encoder, thời gian lấy mẫu và số xung chênh lệch đọc được giữa 2 chu kỳ lấy mẫu và bán kính bánh xe, được tính theo công thức:

$$v = 2\pi R \frac{cnt(k) - cnt(k-1)}{T_s \text{Pulses}}$$

Trong đó:

$cnt(k)$: giá trị số xung hiện tại đọc được từ Encoder

$cnt(k-1)$: giá trị số xung đọc được từ chu kì trước đó

R : bán kính bánh xe (m)

T_s : thời gian lấy mẫu (s)

$Pulses$: tổng số xung trên 1 vòng (xem trên datasheet của Encoder)

Bước 2: Cập nhật luật điều khiển PID rời rạc, công thức luật điều khiển

$$P_{Part} = K_p(e(k) - e(k-1))$$

$$I_{Part} = \frac{K_I T}{2} (e(k) + e(k-1))$$

$$D_{Part} = \frac{K_D}{T} (e(k) - 2e(k-1) + e(k-2))$$

$$u(k) = u(k-1) + P_{Part} + I_{Part} + D_{Part}$$

Ngõ ra $u(k)$ được cho qua khâu bão hòa $[-1;1]$ để tránh trường hợp vượt lố và vượt tầm giá trị của PWM.

Bước 3: Xuất xung điều khiển PWM

Vận tốc động cơ được điều chỉnh dựa vào giá trị độ rộng xung PWM trong 1 chu kì với tần số phù hợp với động cơ đang sử dụng.

Để tài chọn điều khiển với tần số 25KHz. Ta tính được tổng số xung trong 1 chu kì xuất xung với duty cycle 100% là

$$N = 25000T_s \text{ (xung)}$$

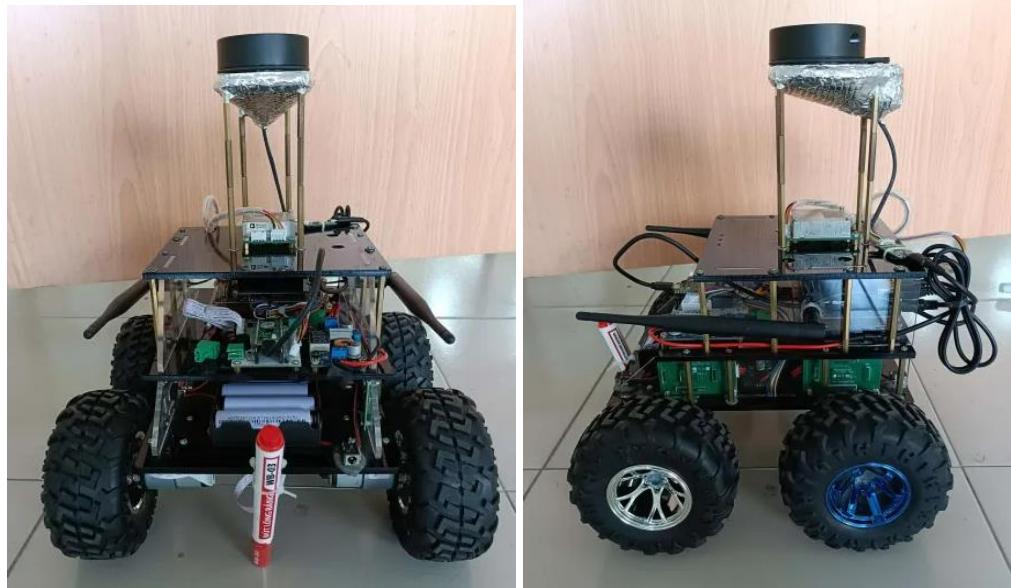
Từ đó tính được số xung xuất ra trong 1 chu kì với tín hiệu điều khiển uk từ bộ điều khiển PID là

$$PWM = u_k N \text{ (xung)}$$

Chương 5. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ

5.1. Kết quả xây dựng mô hình robot

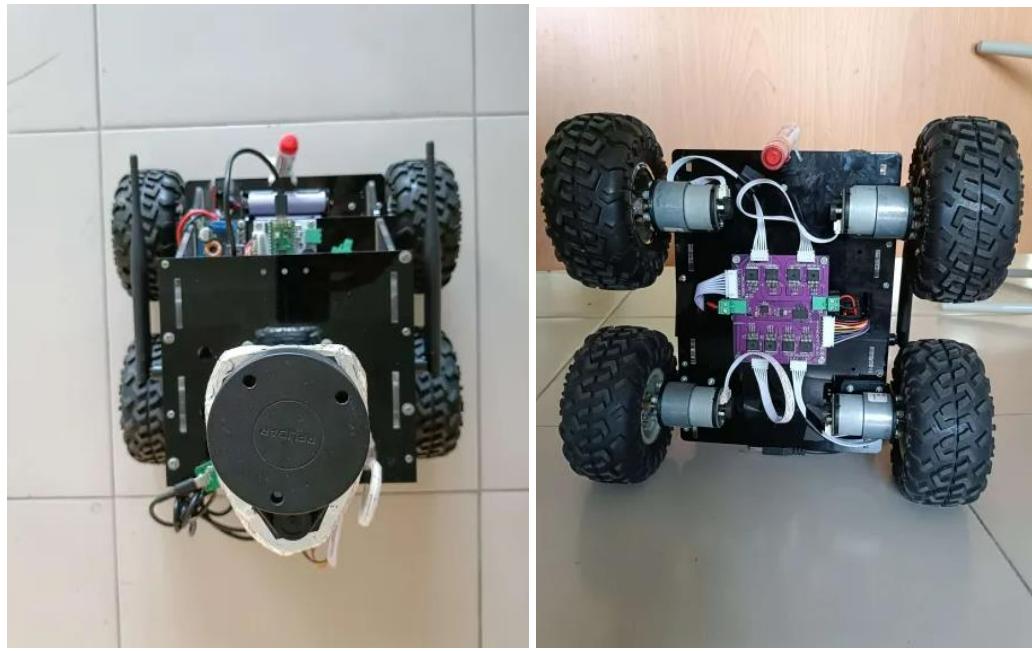
Sau khi thực hiện vẽ mô hình robot trên SolidWorks, nhóm tiến hành cắt mica và lắp ráp. Kết quả thu được như hình dưới đây gồm 3 tần như đã thiết kế ở mục 3.3.



Hình 5.1: Mặt trước và mặt bên trái của robot



Hình 5.2: Mặt sau và mặt bên phải của robot



Hình 5.3: Mặt trên và mặt dưới của robot

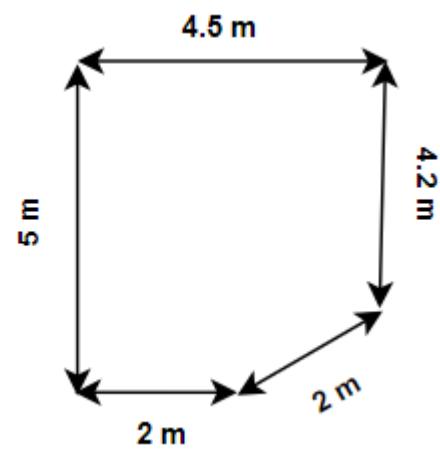
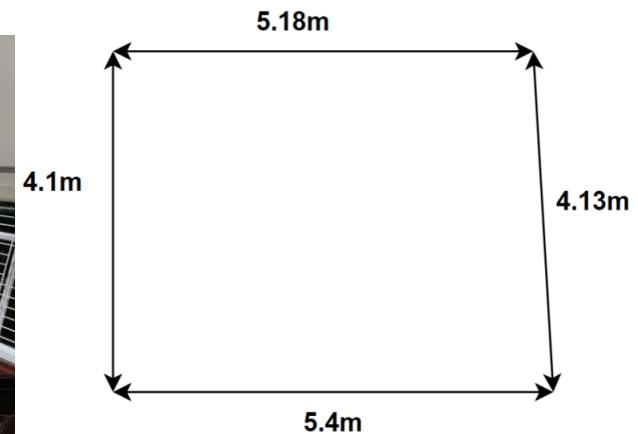
Loại thông số	Thiết kế	Thực tế
Chiều dài (cm)	25	25
Chiều rộng (cm)	18	18
Chiều cao (cm)	21	21
Cân nặng (kg)	3	3.5
Bán kính bánh xe (cm)	6.25	6.33

Hình 5.4: Thông số kích thước mô hình robot

Mô hình robot thực tế được chỉnh sửa trong quá trình thực hiện, nâng cấp cải tiến và thay đổi một số thông số đã thiết kế, và đã đáp ứng được các yêu cầu đề ra gồm khả năng di chuyển, khả năng chịu tải và khả năng quan sát môi trường.

5.2. Môi trường tiến hành thực nghiệm

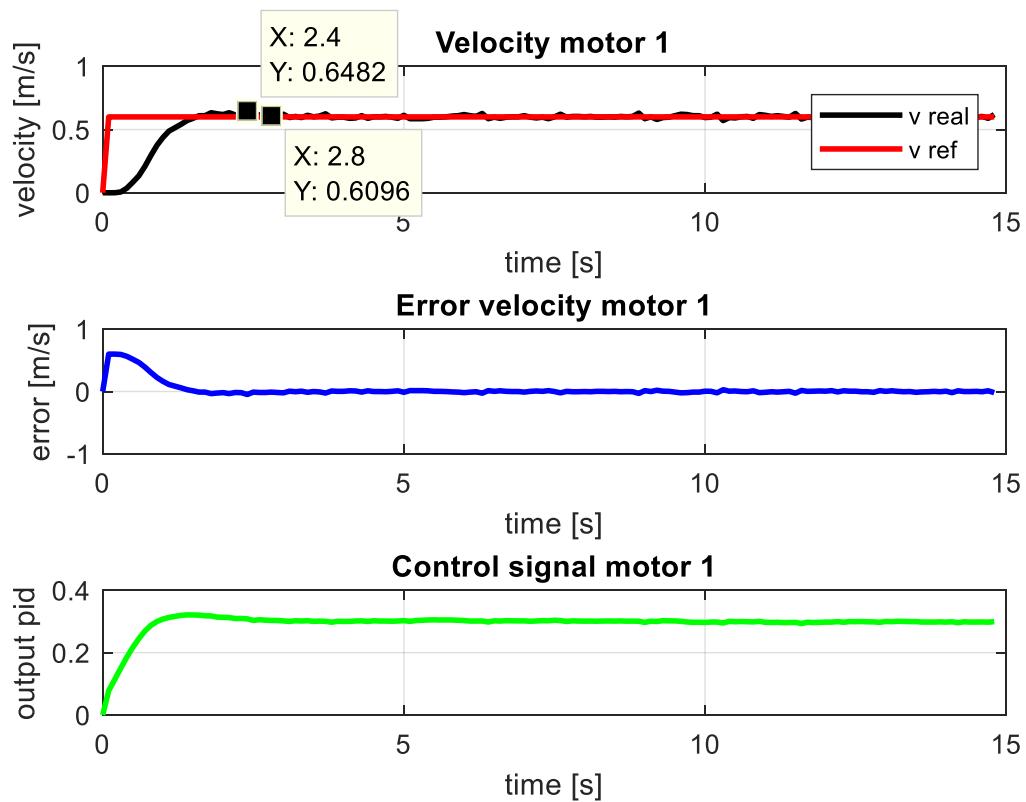
Các kết quả thu được với môi trường thực nghiệm sau:



Hình 5.5: Môi trường tiến hành thực nghiệm

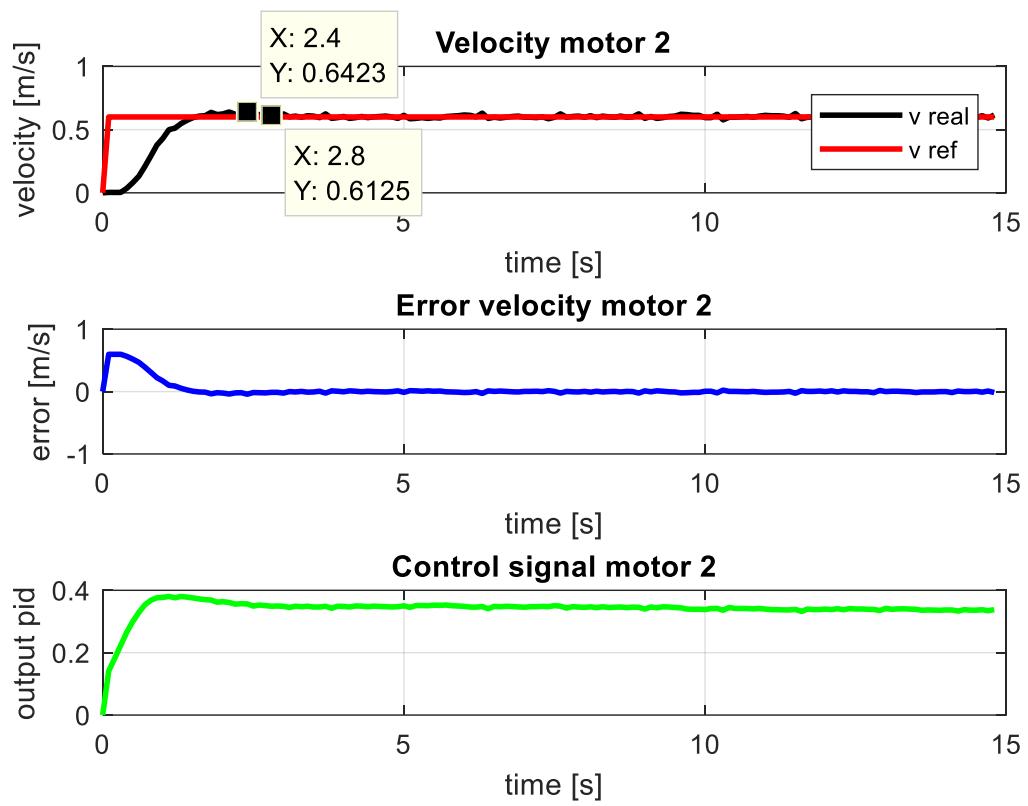
5.3. Kết quả điều khiển vận tốc

Đồ thị đáp ứng vận tốc, sai số, tín hiệu điều khiển của 4 động cơ, vận tốc tuyến tính và vận tốc góc của robot với vận tốc tuyến tính là 0.6 m/s và vận tốc góc là 0 rad/s.



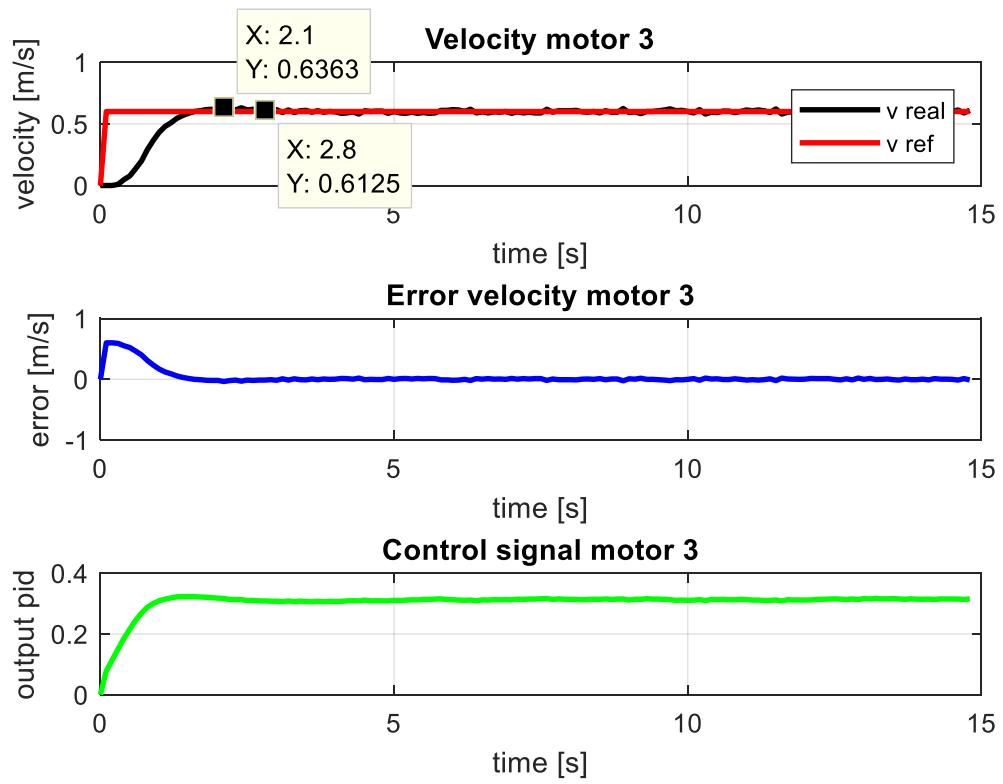
Hình 5.6: Đáp ứng điều khiển vận tốc M1 có tải, $K_p = 0.1, K_I = 0.6, K_D = 0$

Đáp ứng vận tốc của động cơ M1 có sai số bình phương trung bình là 0.0088 m/s, độ vọt lố 8.1% và thời gian xác lập là 2.8s.



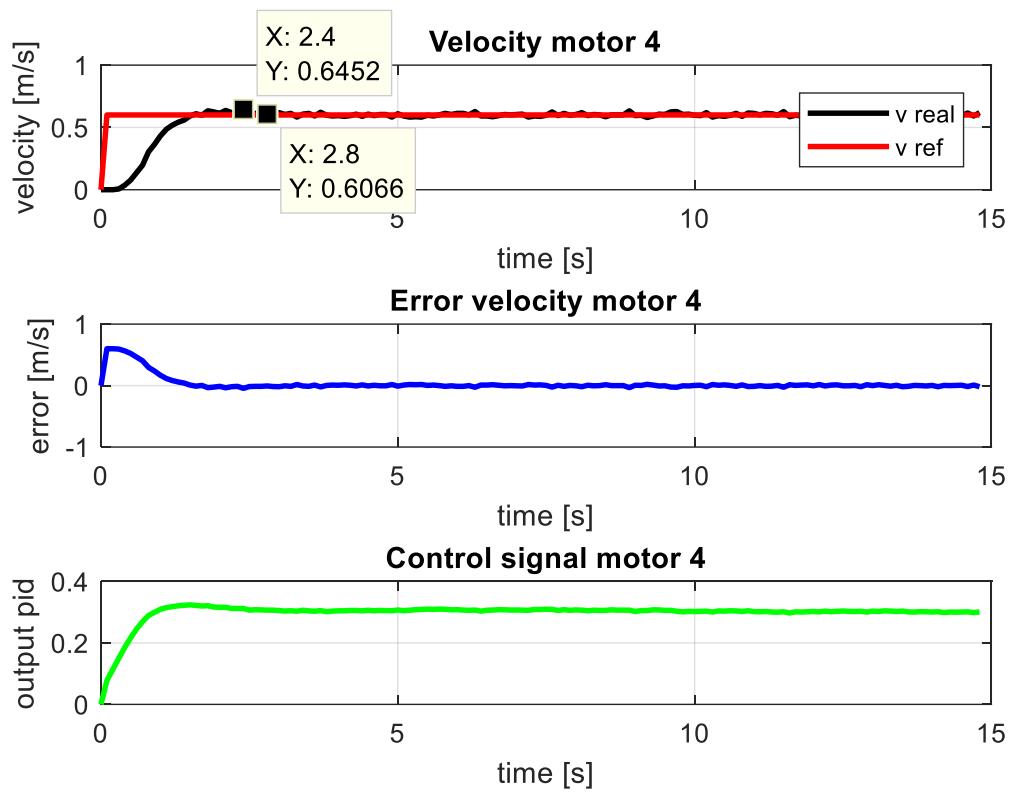
Hình 5.7: Đáp ứng điều khiển vận tốc M2 có tải, $K_P = 0.2, K_I = 0.7, K_D = 0$

Đáp ứng vận tốc của động cơ M2 có sai số bình phương trung bình là 0.0083 m/s, độ vọt lố 7.05% và thời gian xác lập là 2.8s.



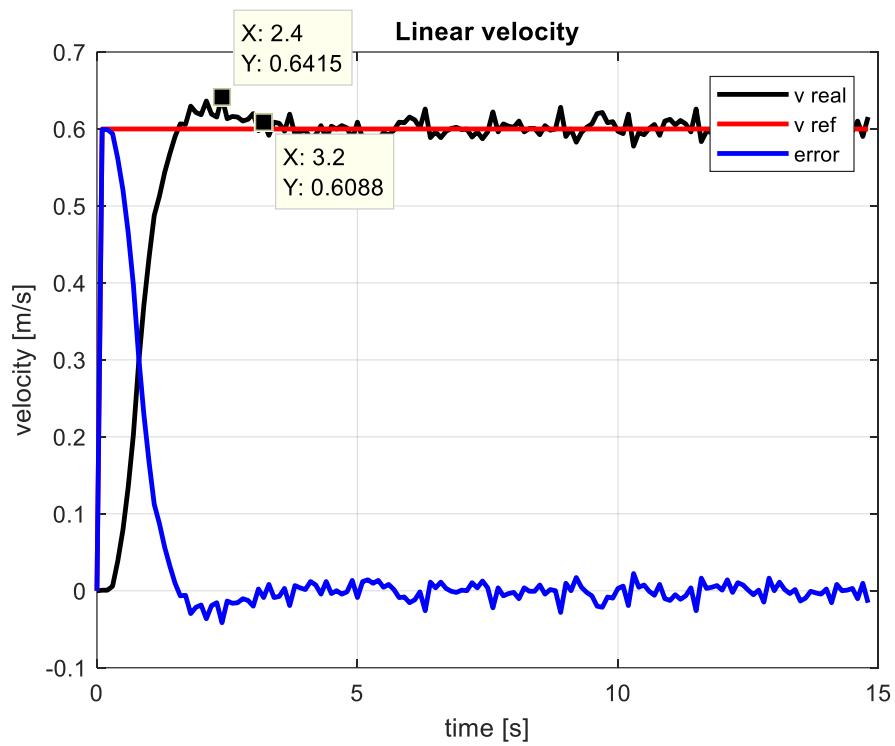
Hình 5.8: Đáp ứng điều khiển vận tốc M3 có tải, $K_P = 0.1, K_I = 0.6, K_D = 0$

Đáp ứng vận tốc của động cơ M3 có sai số bình phương trung bình là 0.0083 m/s, độ vọt lố 6.05% và thời gian xác lập là 2.8s.

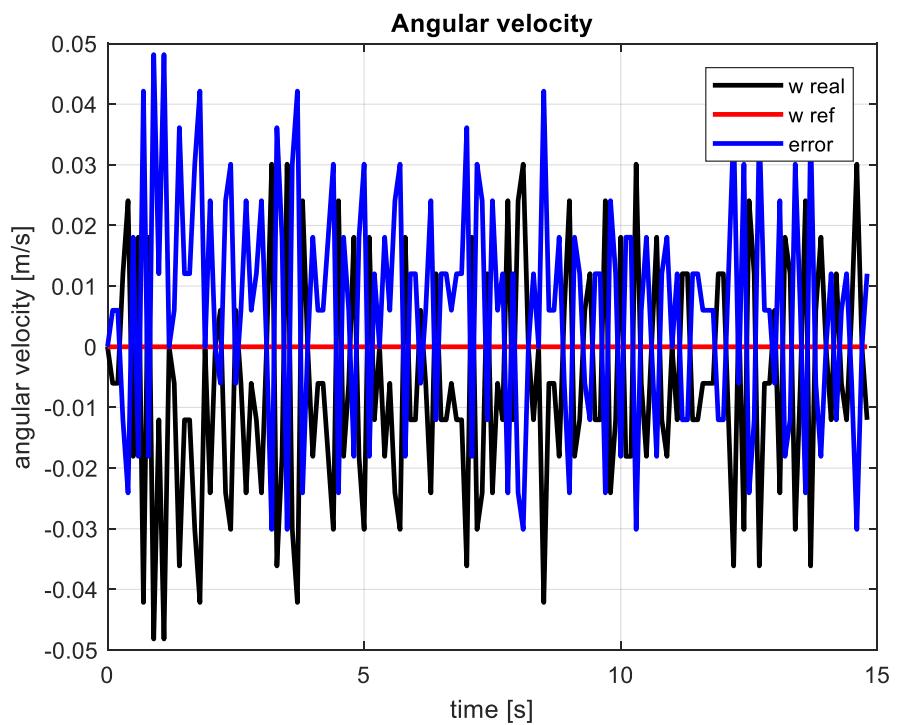


Hình 5.9: Đáp ứng điều khiển vận tốc M4 có tải, $K_p = 0.1, K_I = 0.6, K_D = 0$

Đáp ứng vận tốc của động cơ M4 có sai số bình phương trung bình là 0.0095 m/s, độ vọt lố 7.53% và thời gian xác lập là 2.8s.



Hình 5.10: Đáp ứng vận tốc tuyến tính khi robot đi thẳng



Hình 5.11: Đáp ứng vận tốc góc khi robot đi thẳng

	Motor 1	Motor 2	Motor 3	Motor 4	v	ω (rad/s)
MSE (m/s)	0.0088	0.0083	0.0083	0.0095	0.0083	0.0143
POT(%)	8.1	7.05	6.05	7.53	6.92	
t_{xl} (s)	2.8	2.8	2.8	2.8	3.2	

Bảng 5.1: Kết quả điều khiển vận tốc PID

Trong đó: MSE (Mean Square Error) được tính bằng công thức:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Nhận xét:

Khi có tải (trọng lượng của robot) robot vẫn đáp ứng được yêu cầu điều khiển vận tốc với thời gian xác lập khoảng 3s và độ vọt lỗ dưới 10% và sai số rất bé khoảng 0.008 m/s và 0.0143 rad/s.

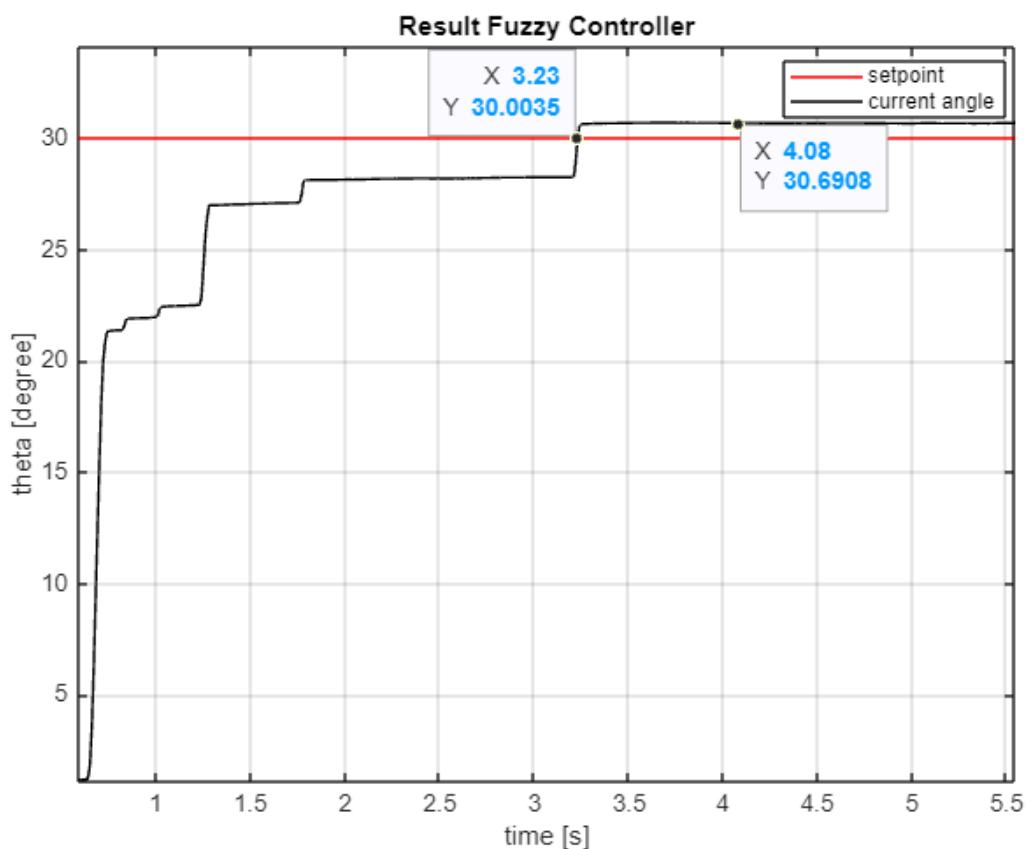
Vì bài toán cần điều khiển chính xác góc lái của robot, nên yêu cầu chọn các hệ số PID của từng động cơ sao cho đáp ứng của 4 động cơ càng giống nhau càng tốt. Vì nếu khi vận tốc 4 bánh xe khác nhau khi cùng một giá trị đặt thì robot sẽ chạy lệch về một hướng, ánh hưởng đến góc lái mong muốn.

Tuy có cùng thông kỹ thuật trên lý thuyết nhưng 4 động cơ hoàn toàn có thể có đáp ứng khác nhau trong thực tế. Điều này đòi hỏi phải dựa vào thực tế để tinh chỉnh các hệ số trong bộ điều khiển PID cho từng động cơ chứ không dùng chung một bộ thông số PID cho cả 4 động cơ.

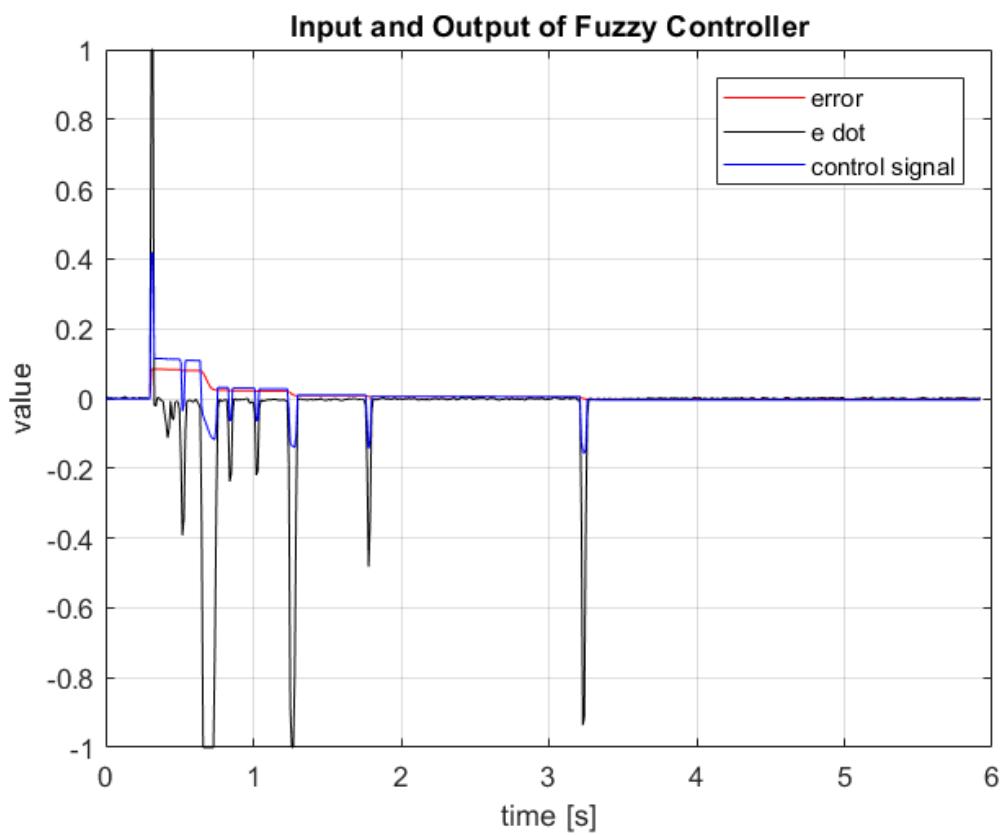
5.4. Kết quả điều khiển góc

Output của thuật toán Stanley cho ta một góc delta angle $\Delta\theta$ để có thể đưa robot vào quỹ đạo mong muốn, góc delta angle $\Delta\theta$ này được qua bộ điều khiển PD Fuzzy để đưa góc heading θ trở về góc heading mong muốn, kết quả của bộ điều khiển trong thực nghiệm được thể hiện bởi các hình sau:

Trường hợp 1: delta angle $\Delta\theta = 30^\circ$, yêu cầu robot xoay phải một góc 30° quanh tâm của robot

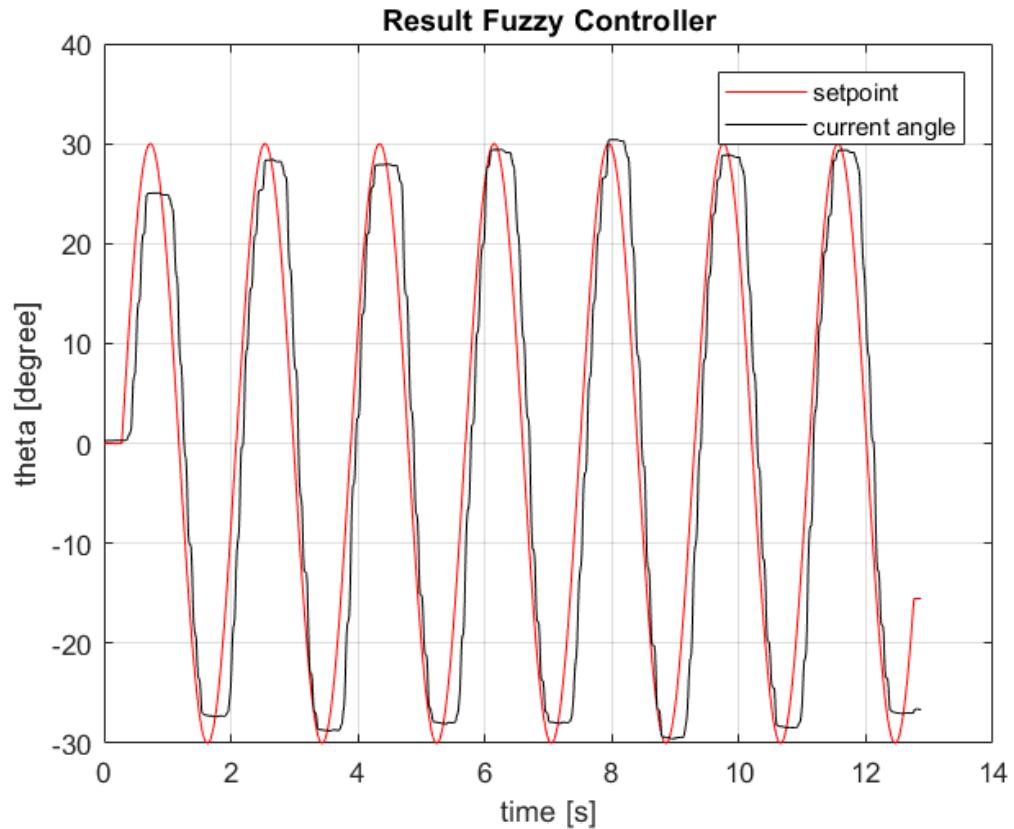


Hình 5.12: Đáp ứng góc của robot trường hợp góc đặt là hàm nấc

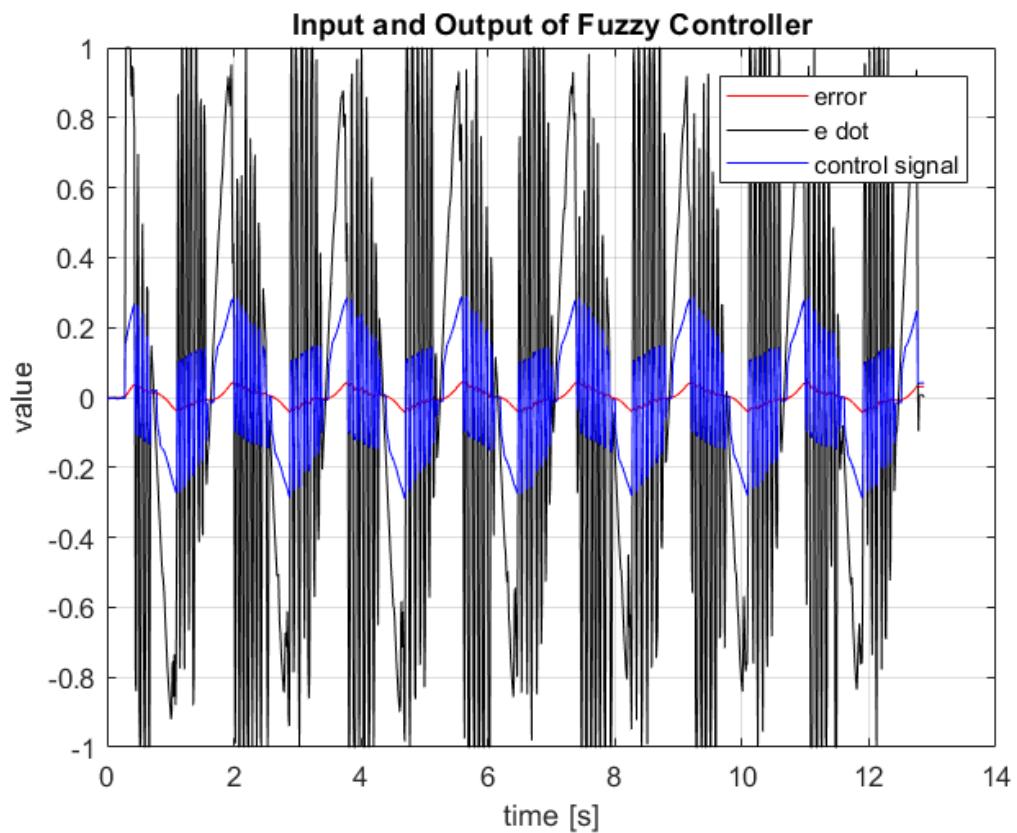


Hình 5.13: Sai số và tín hiệu điều khiển của PD Fuzzy trường hợp góc đặt là hàm nắc

Trường hợp 2: delta angle $\Delta\theta$ thay đổi theo dạng hình sin có biên độ là 30° có phương trình như sau delta theta $\Delta\theta = 30\sin(\frac{\pi t}{180})$ yêu cầu robot dao động quanh góc heading ban đầu với biên độ là 30° .



Hình 5.14: Đáp ứng góc của robot trường hợp góc đặt là hàm sin



Hình 5.15: Sai số và tín hiệu điều khiển của PD Fuzzy trường hợp góc đặt là hàm sin

Nhận xét:

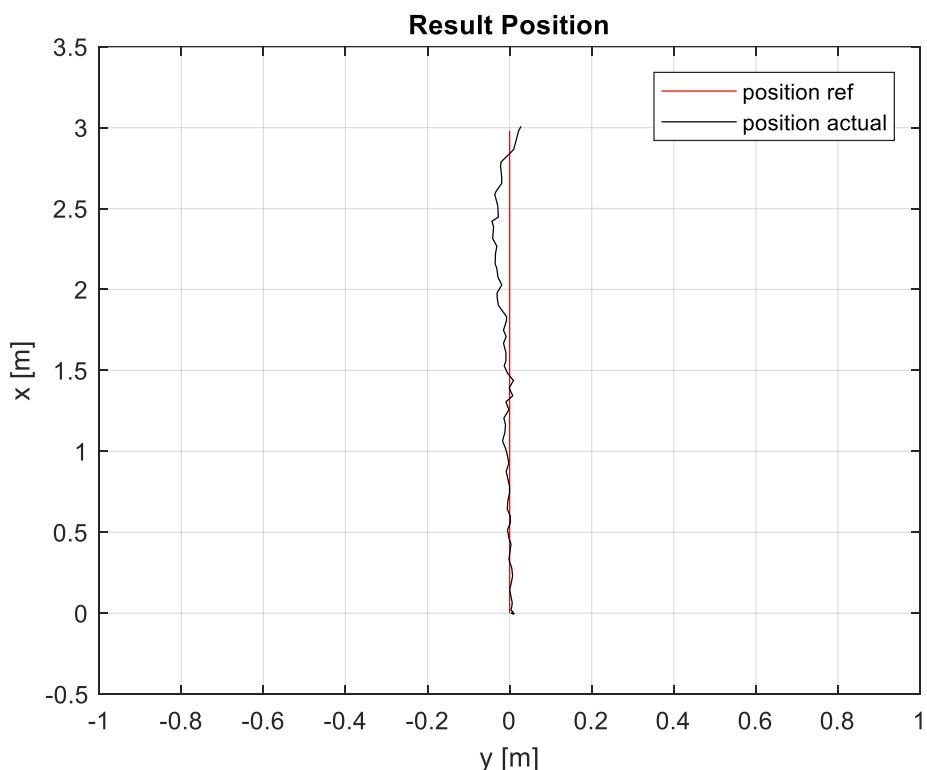
Đáp ứng góc không bị vọt lố, sai số 0.69 độ và thời gian xác lập là 3.23s. Trong đó thời gian xác lập sẽ phụ thuộc một phần vào vận tốc tối đa của động cơ, khi tốc độ quay động cơ càng nhanh thì đáp ứng góc càng nhanh nhưng cũng có thể gây ra hiện tượng vọt lố.

Thực nghiệm được tiến hành trên 2 trường hợp góc đặt là hàm nắc và hình sin đều cho ra đáp ứng tốt chứng minh rằng bộ điều khiển có thể điều khiển tốt khi robot cần cua gắt hoặc xoay với một góc nhỏ.

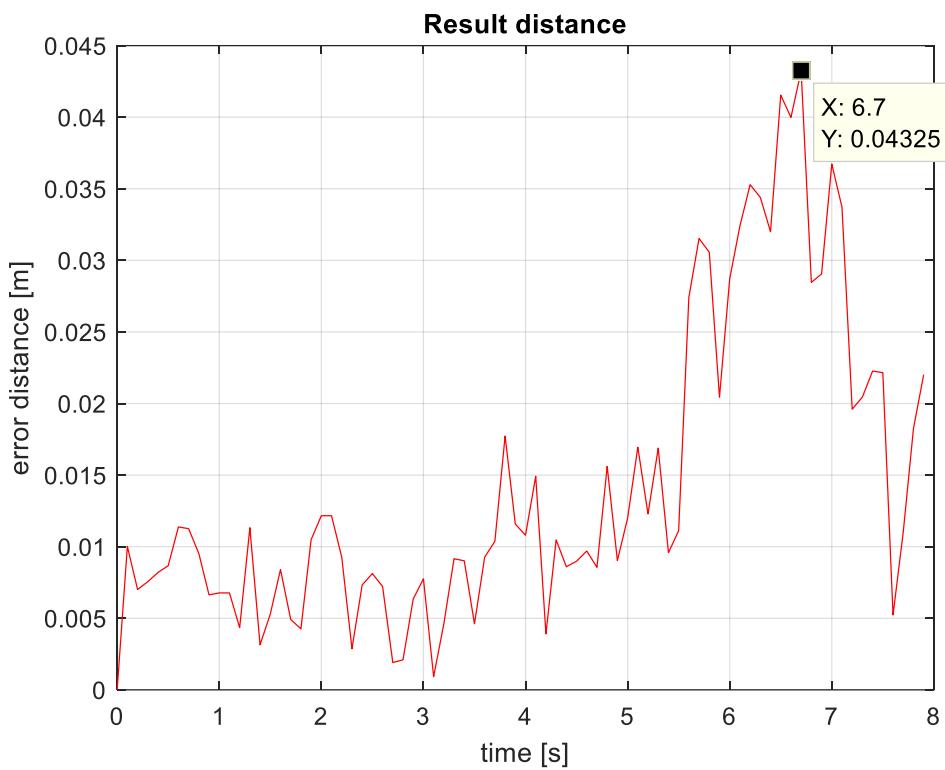
5.5. Kết quả điều khiển bám quỹ đạo

Để khảo sát chất lượng điều khiển bám quỹ đạo của thuật toán Stanley, các thí nghiệm được thực hiện lần lượt bằng cách khảo sát đáp ứng với những quỹ đạo khác nhau với vận tốc không đổi là 0.3m/s và $K=0.575$.

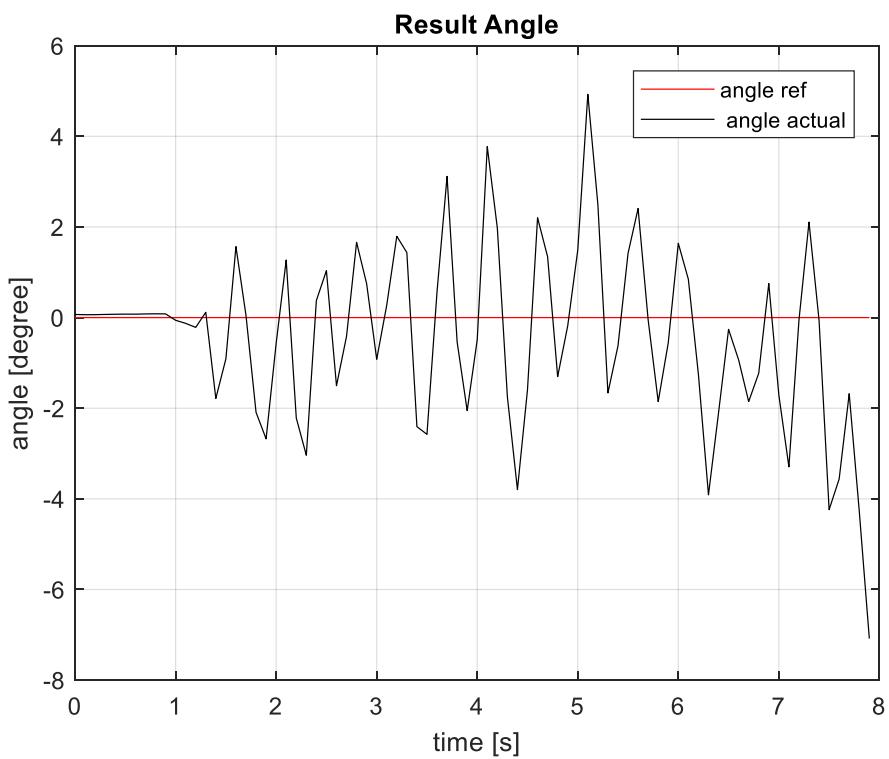
Trường hợp 1: Quỹ đạo đường thẳng với góc hợp bởi trục Ox là 0°



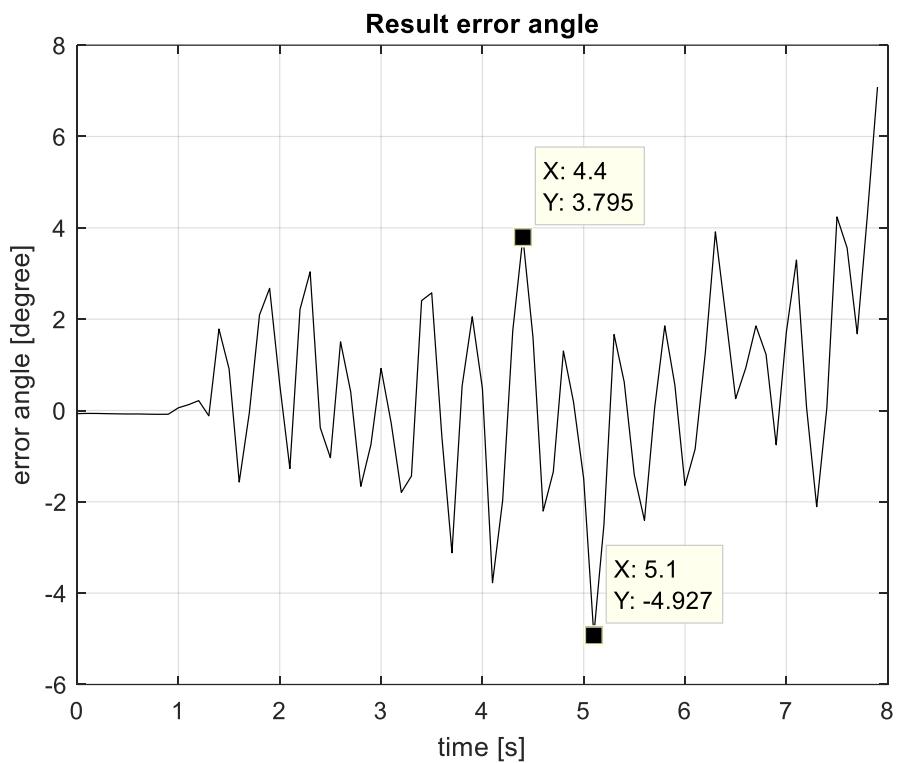
Hình 5.16: Đáp ứng quỹ đạo với ref là đường thẳng 0°



Hình 5.17: Sai số vị trí với ref là đường thẳng 0°



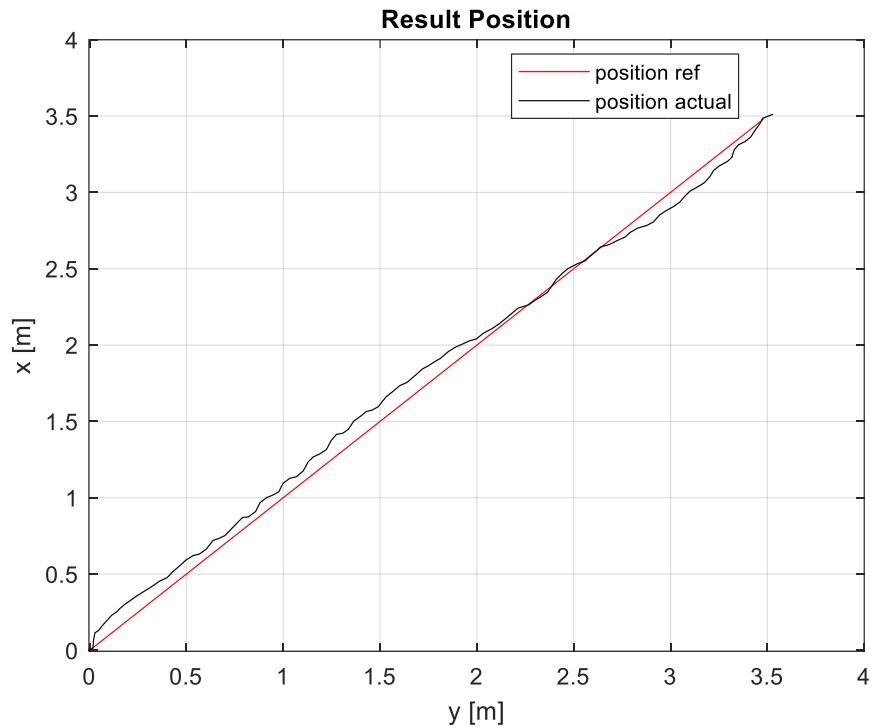
Hình 5.18: Đáp ứng góc với ref là đường thẳng 0°



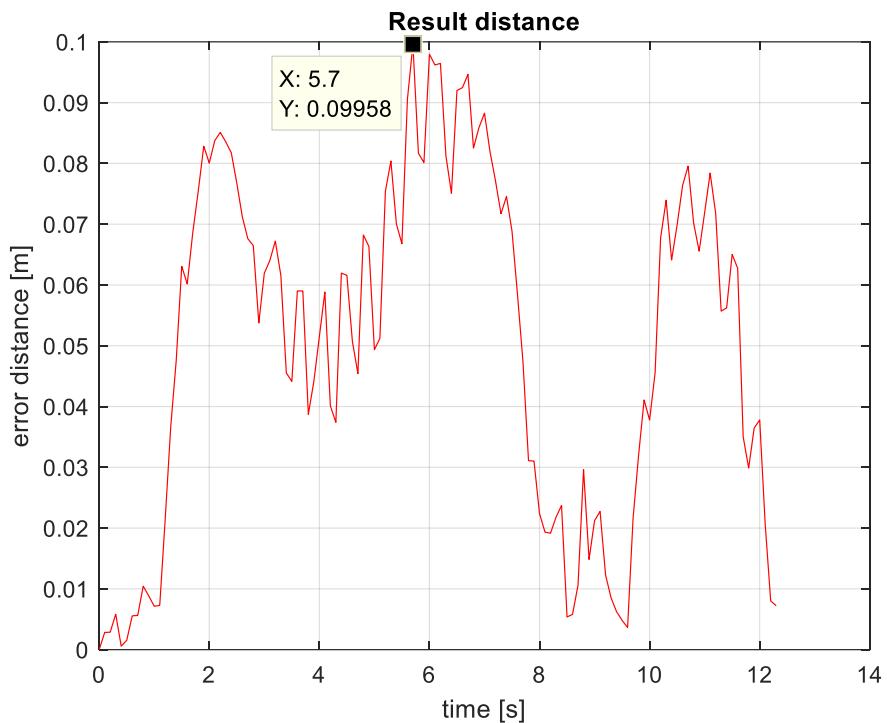
Hình 5.19: Sai số góc với ref là đường thẳng 0°

Robot bám quỹ đạo đường thẳng 0° với sai số trung bình là 1.43 cm và sai số lớn nhất là 4.33 cm.

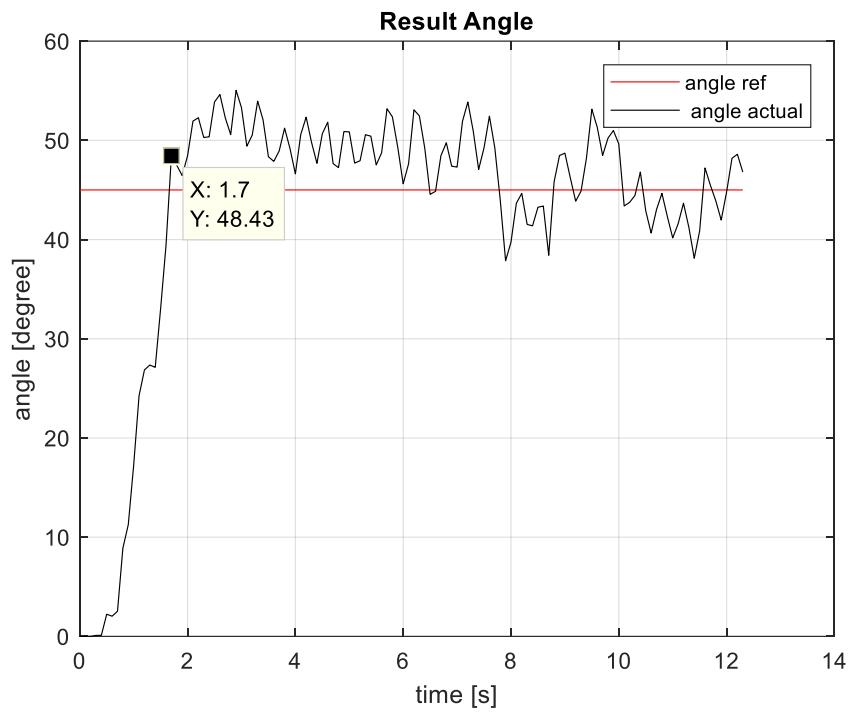
Trường hợp 2: Quỹ đạo đường thẳng với góc hợp bởi trục Ox là 45°



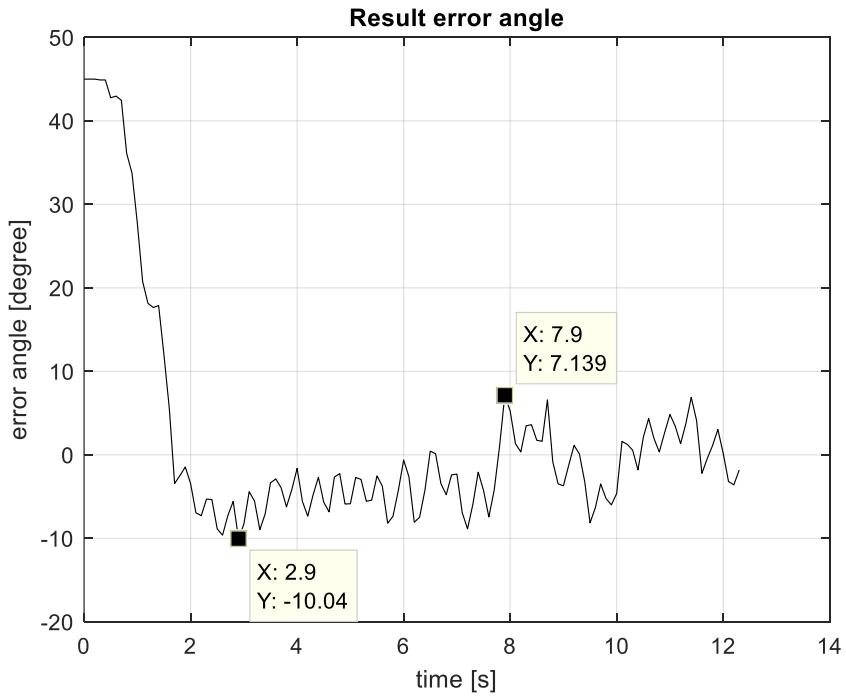
Hình 5.20: Đáp ứng quỹ đạo với ref là đường thẳng 45°



Hình 5.21: Sai số vị trí với ref là đường thẳng 45°



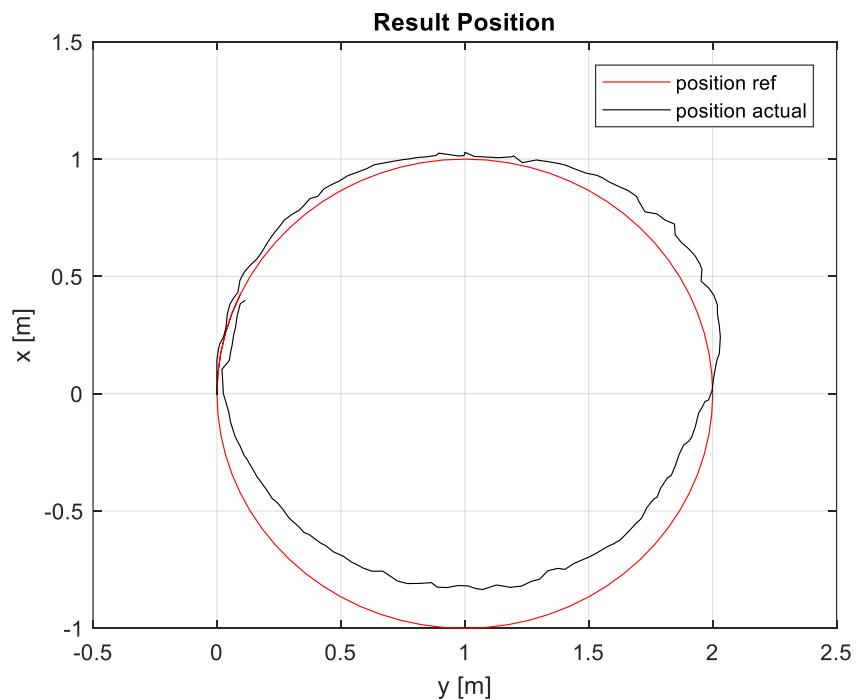
Hình 5.22: Đáp ứng góc với ref là đường thẳng 45°



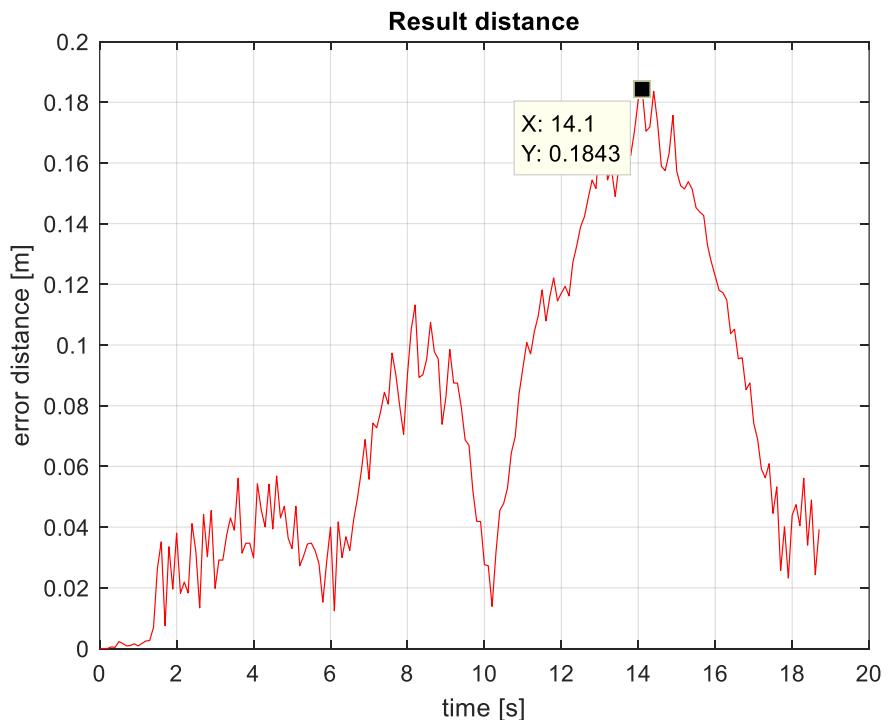
Hình 5.23: Sai số góc với ref là đường thẳng 45°

Robot bám quỹ đạo đường thẳng 45° với sai số trung bình là 5.05 cm và sai số lớn nhất là 9.96 cm.

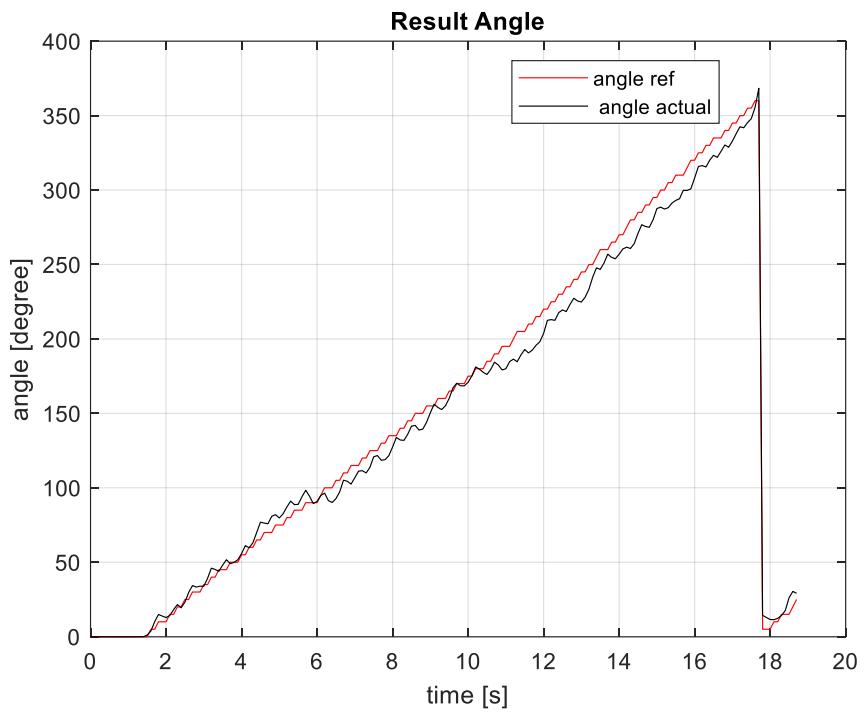
Trường hợp 3: Quỹ đạo đường tròn



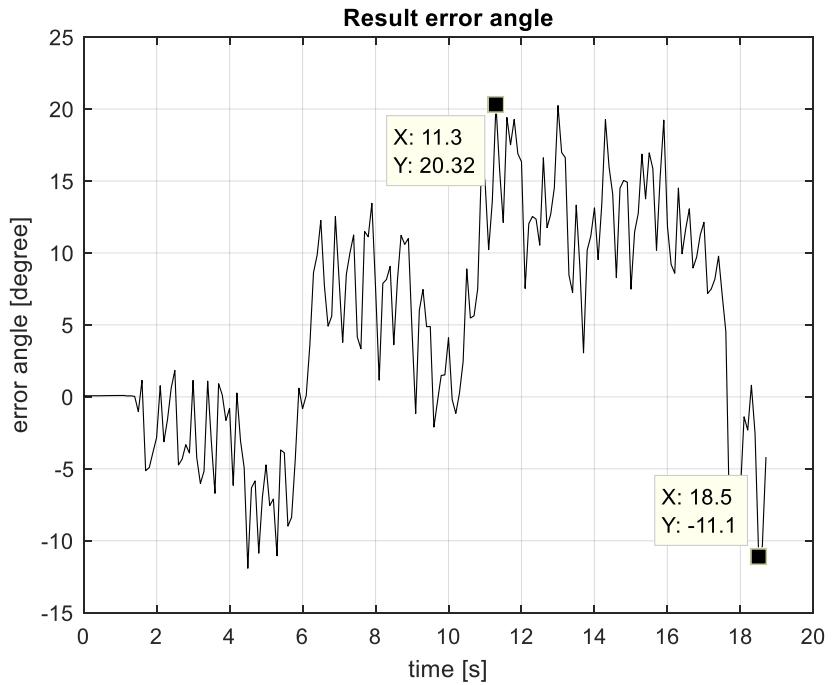
Hình 5.24: Đáp ứng quỹ đạo với ref là đường tròn



Hình 5.25: Sai số vị trí với ref là đường tròn



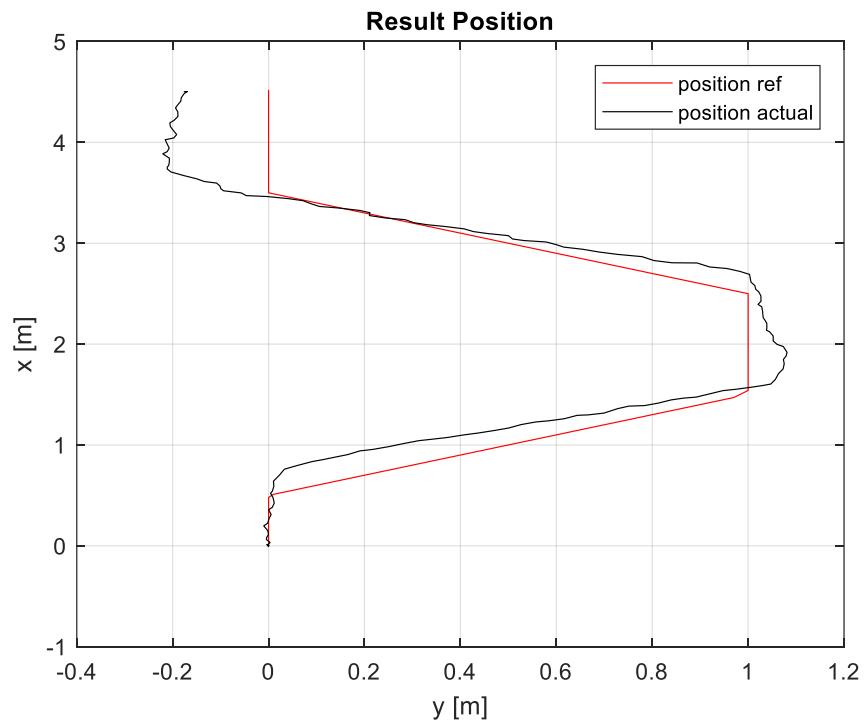
Hình 5.26: Đáp ứng góc với ref là đường tròn



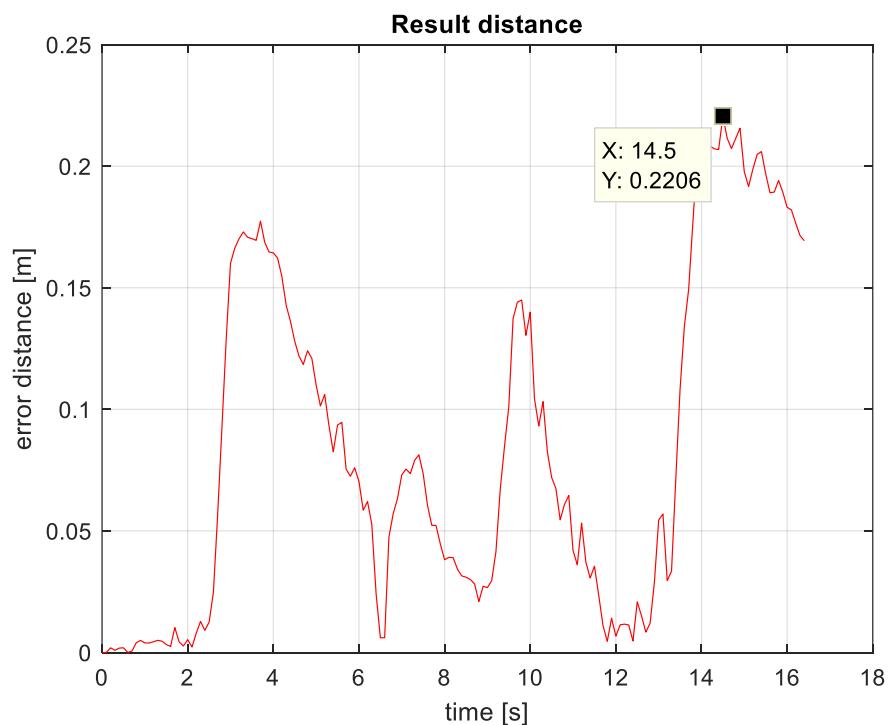
Hình 5.27: Sai số góc với ref là đường tròn

Robot bám quỹ đạo đường tròn với sai số trung bình là 7.46 cm và sai số lớn nhất là 18.43 cm.

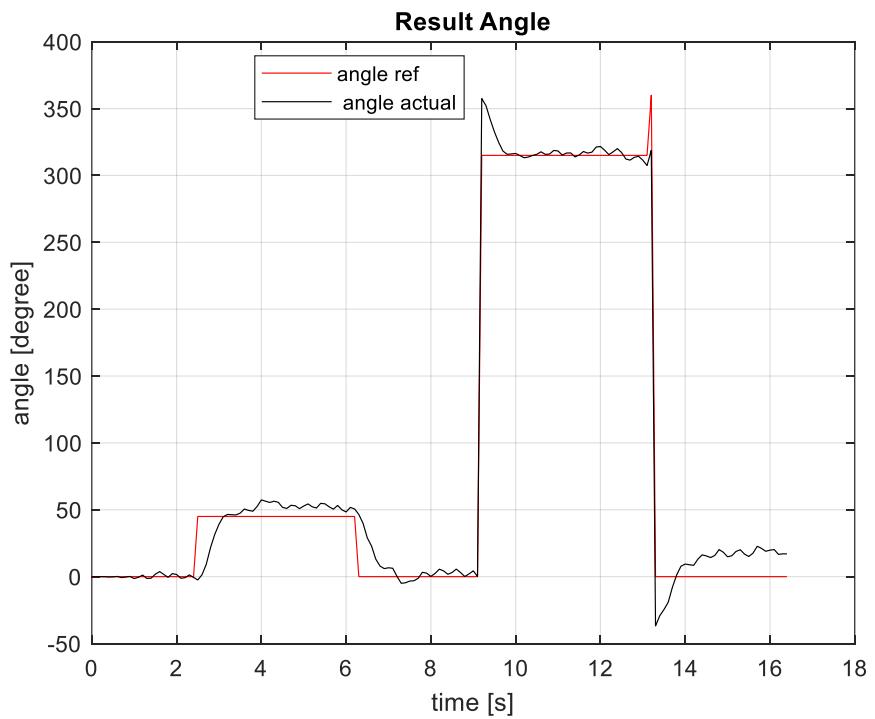
Trường hợp 4: Quỹ đạo hình thang



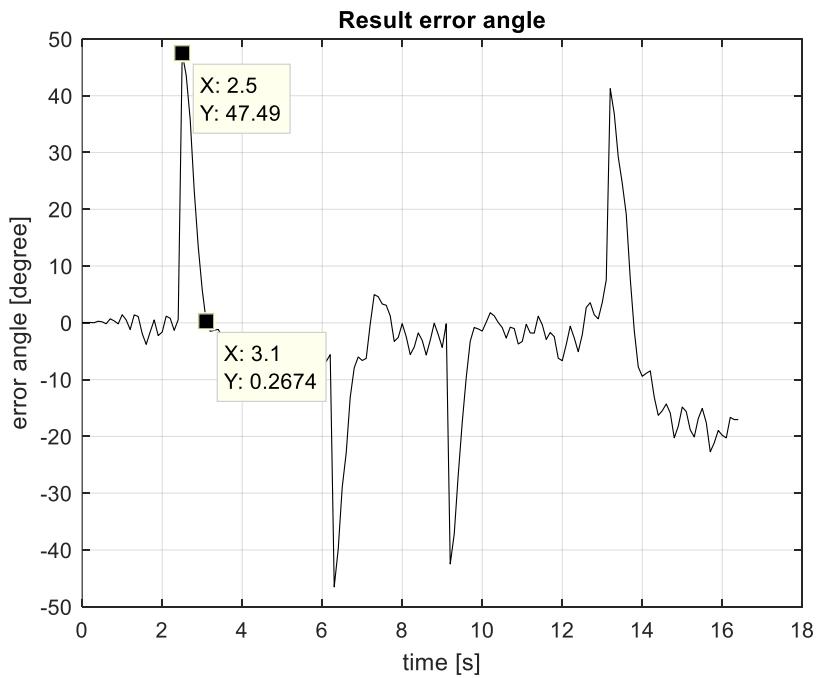
Hình 5.28: Đáp ứng quỹ đạo với ref là hình thang



Hình 5.29: Sai số vị trí với ref là hình thang



Hình 5.30: Đáp ứng góc với ref là hình thang



Hình 5.31: Sai số góc với ref là hình thang

Robot bám quỹ đạo đường hình thang với sai số trung bình là 8.46 cm và sai số lớn nhất là 22.06 cm.

	Traight 0	Traight 45	Circle	Trapezoid
Sai số trung bình (m)	0.0143	0.0505	0.0746	0.0846
Sai số lớn nhất (m)	0.0433	0.0996	0.1843	0.2206

Bảng 5.2: Kết quả điều khiển bám quỹ đạo

Nhận xét:

Kết quả bám quỹ đạo với sai số trung bình từ 1 cm đến 8 cm, sai số này có thể do dữ liệu laser quét các vật xung quanh bị nhiễu và vận tốc feedback sai lệch do nhiễu Encoder,... Tuy nhiên, bộ điều khiển đã hoạt động tốt với sai số chấp nhận được khi bám các quỹ đạo khác nhau trong môi trường nhỏ hẹp và nhiều khúc cua.

Tại những khúc cua gắt (ví dụ như 45 độ ở thực nghiệm bám quỹ đạo hình thang) cho thời gian đáp ứng ngắn (khoảng 0.6s). Từ đó cho thấy được thế mạnh của mô hình robot truyền động 4 bánh độc lập của đề tài so với các robot có cơ cấu lái vi sai. Hướng xe có thể thay đổi đột ngột mà không chiếm nhiều diện tích không gian và thời gian.

Chương 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Luận văn đã hoàn thành mục tiêu đề ra, xây dựng mô hình robot thực tế di chuyển trong nhà, sử dụng các cảm biến Lidar, IMU và Encoder để bám quỹ đạo đã hoạch định trước. Sử dụng các bộ điều khiển Stanley, PD Fuzzy, PID và thuật toán ICP để điều khiển robot đi theo quỹ đạo một cách chính xác. Với kết quả của bộ điều khiển vận tốc và bộ điều khiển góc có sai số không quá 1 cm/s và 1° đồng thời độ vọt lỗ dưới 10% và thời gian xác lập nhỏ hơn 3s; kết quả của thuật toán bám quỹ đạo có sai số khoảng cách trung bình không quá 10 cm.

Nhiều thử nghiệm đã được thực hiện với các quỹ đạo và môi trường khác nhau để chứng minh sự hiệu quả của các bộ điều khiển. Tuy nhiên, đề tài vẫn còn hạn chế đó là các thử nghiệm được thực hiện với vận tốc tuyến tính của robot không thay đổi dẫn đến tại những khúc cua robot sẽ bị vọt lỗ khỏi quỹ đạo mong muốn.

Hệ thống phần cứng và phần mềm được xây dựng để thu thập dữ liệu và đánh giá hệ thống có thể tận dụng lại hệ thống để thử nghiệm với nhiều giải thuật khác nhau hoặc dùng để đánh giá các giải thuật mới.

6.2. Hướng phát triển

Cải tiến thuật toán bám quỹ đạo: kết hợp Stanley Controller với Reinforcement Learning, MPC với Reinforcement Learning. Điều khiển robot với vận tốc tuyến tính thay đổi theo quỹ đạo tham chiếu để giảm thiểu sai số và tăng thời gian đáp ứng.

Sử dụng các thuật toán tối ưu như Differential Evolution (DE), Particle Swarm Optimisation (PSO) để ước lượng các thông số của bộ điều khiển phù hợp với từng môi trường thử nghiệm khác nhau một cách tự động, giúp mô hình có tính thích nghi cao.

Bổ sung thêm tính năng tránh vật cản và bám vật thể chuyển động bằng Camera.

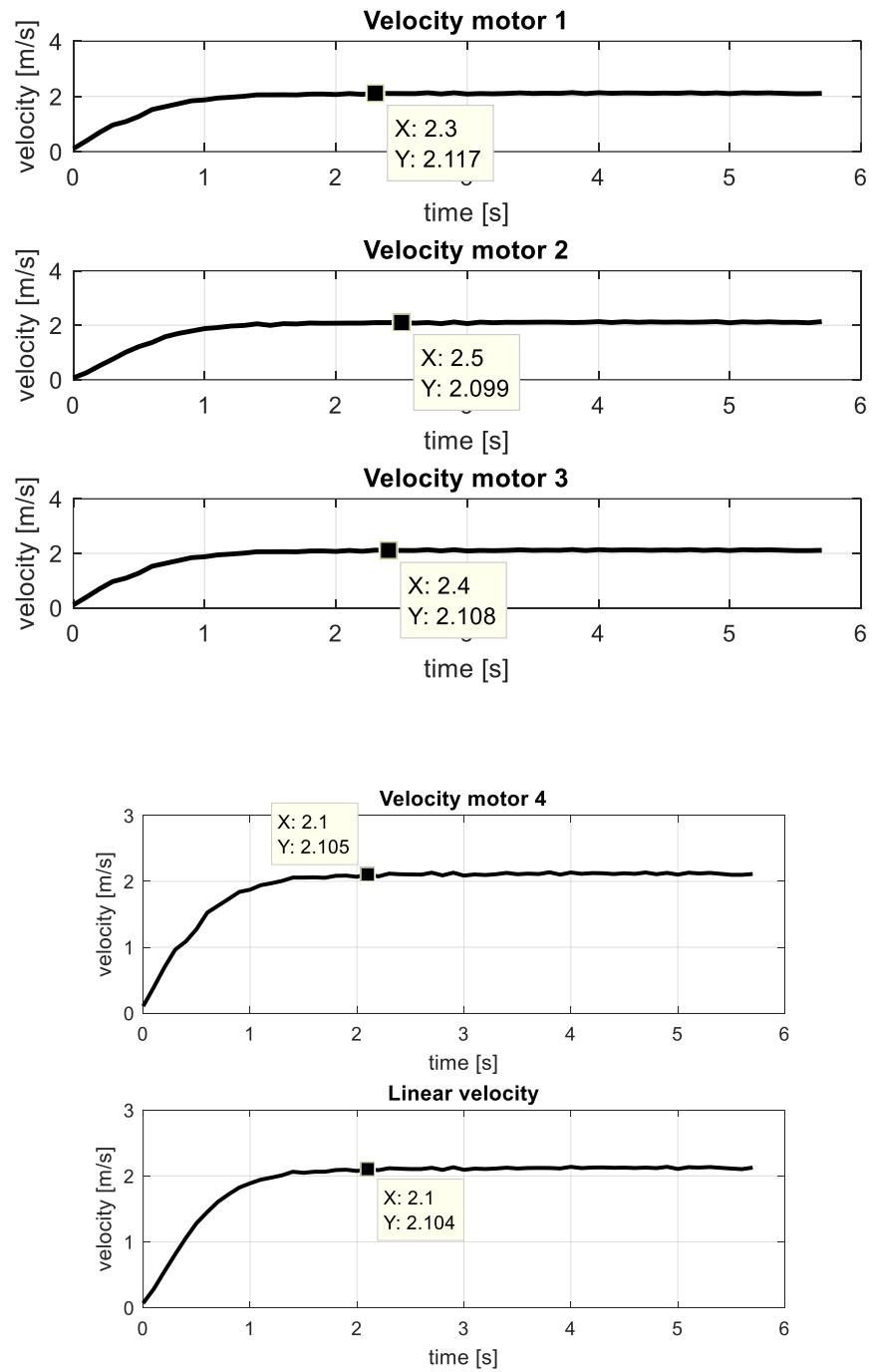
Cải tiến mô hình cơ khí và bổ sung thêm các cảm biến cần thiết như GPS, Lora, Camera để robot có thể di chuyển cả hai môi trường trong nhà và ngoài trời.

TÀI LIỆU THAM KHẢO

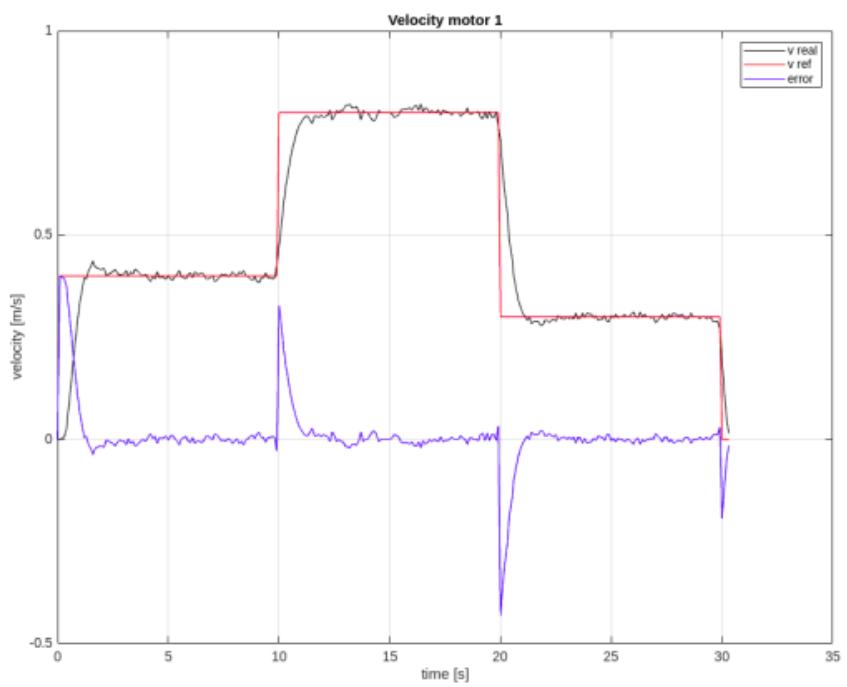
- [1] H. T. Hoàng, Bài giảng môn Cơ sở điều khiển tự động, khoa Điện-Điện tử, Đại học Bách Khoa TPHCM.
- [2] H. T. Hoàng, "Chương 3 Điều khiển mờ," in *Bài giảng môn Nhập môn điều khiển thông minh*, khoa Điện-Điện tử, Đại học Bách Khoa TPHCM.
- [3] A. Censi, "An ICP Variant Using A Point-to-line Metric," *the IEEE International Conference On Robotics And Automation (ICRA)*, May 2008.
- [4] P. E. C. Prof. Giorgio Guglieri, "Prof. Giorgio Guglieri, Pro.ssa Elisa Capello," Master Degree in Aerospace Engineering, April 2020.
- [5] S. M. Thrun, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, p. <https://doi.org/10.1002/rob.20147>, 2006.
- [6] "Micro-ROS architecture," [Online]. Available: <https://micro.ros.org/docs/overview/features/>.
- [7] "Wiki ROS," [Online]. Available: <http://wiki.ros.org/>.

PHỤ LỤC

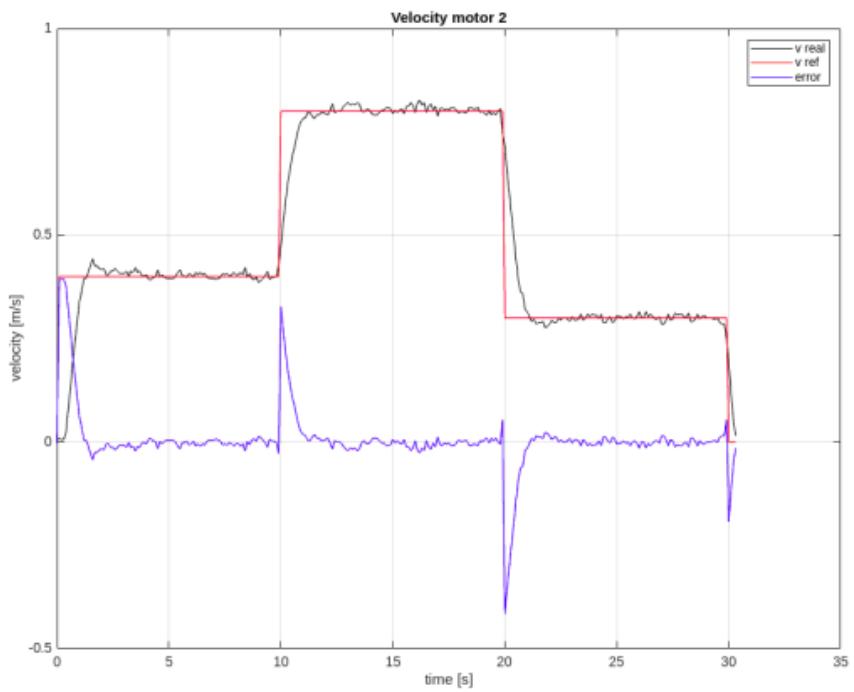
Vận tốc tối đa mà robot có thể di chuyển được là 2.1 m/s



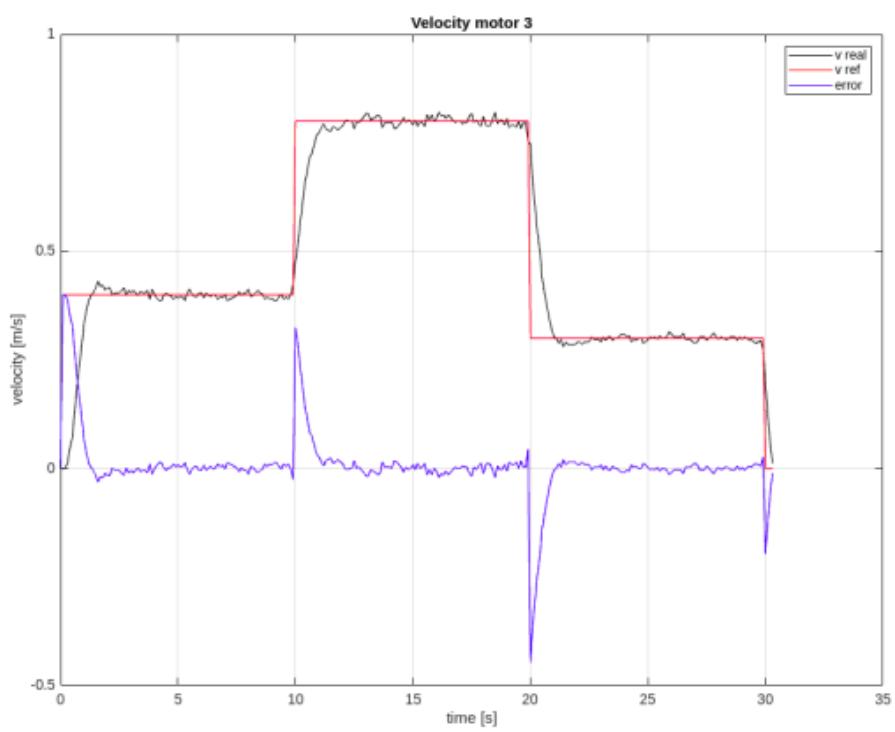
Hình 0.1: Đáp ứng vận tốc của 4 motor và robot khi chạy với vận tốc tối đa
Robot có thể di chuyển ổn định với nhiều giá trị vận tốc khác nhau



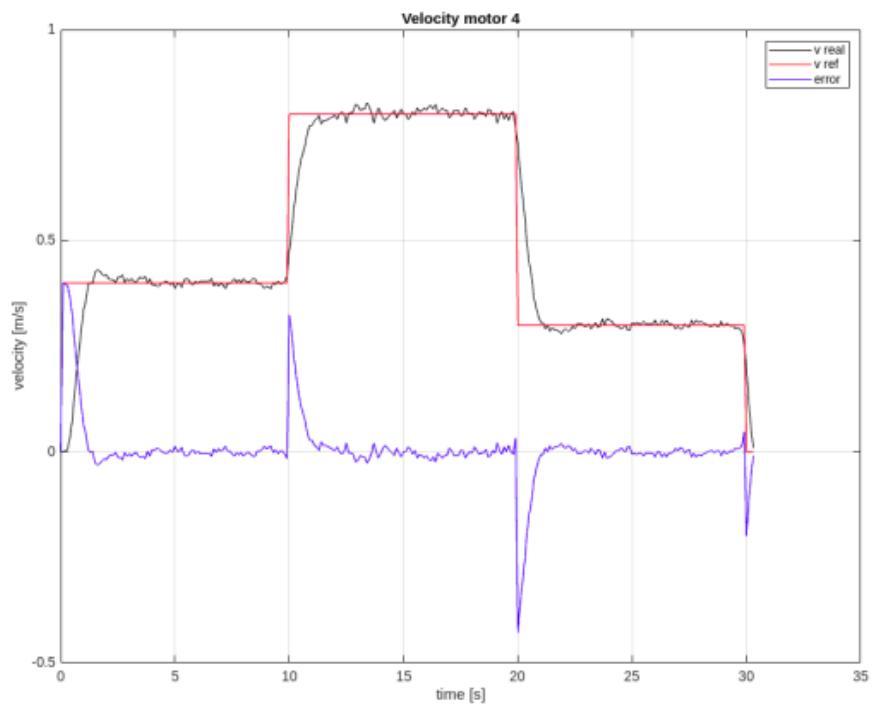
Hình 0.2: Đáp ứng vận tốc của M1 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s



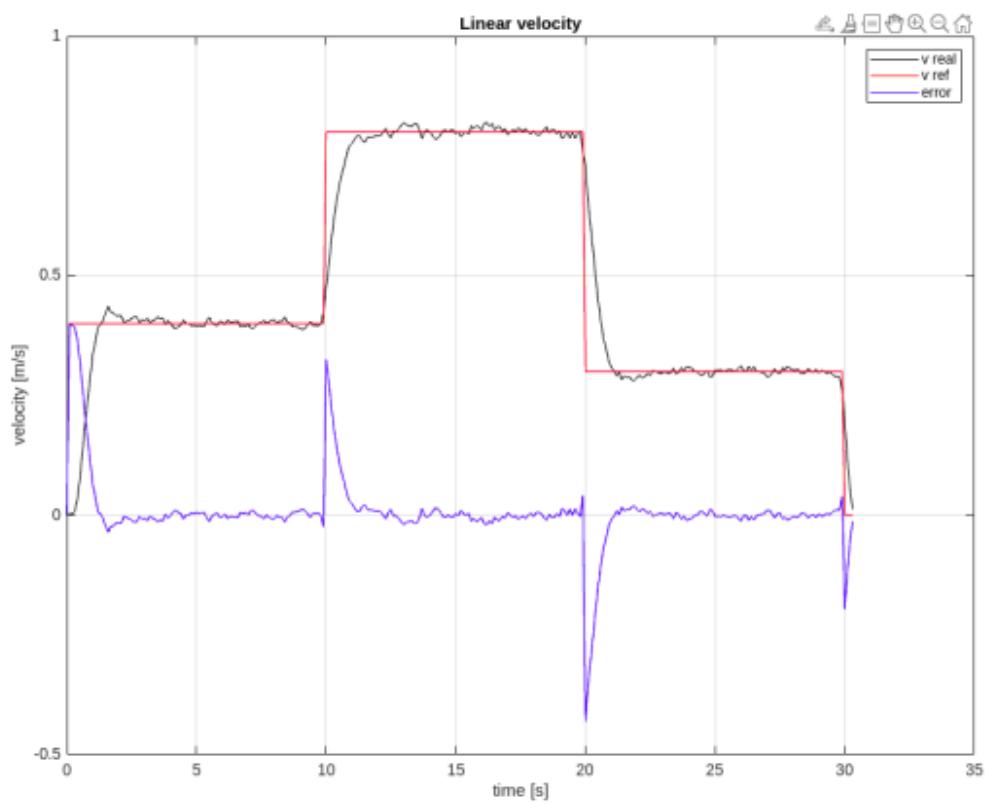
Hình 0.3: Đáp ứng vận tốc của M2 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s



Hình 0.4: Đáp ứng vận tốc của M3 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s



Hình 0.5: Đáp ứng vận tốc của M4 với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s



Hình 0.6: Đáp ứng vận tốc robot với setpoint là 0.4 m/s, 0.8m/s và 0.3m/s