

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
(Dành cho Giảng viên hướng dẫn)

Tên đề tài:

Thiết kế chế tạo robot giao hàng linh hoạt dùng bánh xe quay swerve drive

Họ và tên SV: Đàm Quốc Khánh	Lớp: ME1-03-K63
Họ và tên SV: Hà Lực Minh Vũ	Lớp: ME-E1-01-K63
Họ và tên SV: Ngô Xuân Thành	Lớp: ME1-04-K63
Họ và tên SV: Nguyễn Hoàng Long	Lớp: ME-E1-02-K63

Chuyên ngành: Kỹ thuật Cơ điện tử

Giảng viên hướng dẫn: TS. Bùi Định Bá

NỘI DUNG NHẬN XÉT CỦA GIẢO VIÊN HƯỚNG DẪN

I. Tác phong làm việc

Nhóm làm đồ án có tinh thần làm việc tốt, chăm chỉ, hoàn thành tốt những yêu cầu được đề ra.

II. Những kết quả đạt được

Nhóm đồ án đã xây dựng được hệ để linh hoạt cho AGV và mobile robot sử dụng bánh xe quay swerve drive . Thiết lập hệ thống điều khiển vị trí , tốc độ dùng PID và lập trình tự vẽ bản đồ và tìm đường trên ROS.

III. Hạn chế của đồ án

Đồ án cần cải thiện thêm các tính năng cũng như tích hợp thêm các cảm biến an toàn để phù hợp hơn với môi trường làm việc của robot.

IV. Kết luận

Người hướng dẫn đề nghị cho phép sinh viên được bảo vệ đề tài tốt nghiệp trước Hội đồng chấm đồ án tốt nghiệp.

Đánh giá: 10 điểm

Hà Nội, ngày 12 tháng 8 năm 2023

Giảng viên hướng dẫn



NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
(Dành cho Giảng viên phản biện)

Tên đề tài:

Thiết kế chế tạo robot giao hàng linh hoạt dùng bánh xe quay swerve drive

Họ và tên SV: Đàm Quốc Khánh	Lớp: ME1-03-K63
Họ và tên SV: Hà Lực Minh Vũ	Lớp: ME-E1-01-K63
Họ và tên SV: Ngô Xuân Thành	Lớp: ME1-04-K63
Họ và tên SV: Nguyễn Hoàng Long	Lớp: ME-E1-02-K63

Chuyên ngành: Kỹ thuật Cơ điện tử

Giảng viên hướng dẫn: TS. Bùi Đình Bá

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

I. Những kết quả đạt được

- Nhóm đồ án nắm được nguyên lý, kết cấu và chế tạo được cơ cấu dẫn động swerve drive
- Chế tạo được mô hình robot giao hàng có khả năng di chuyển linh hoạt
- Thiết lập được các giao thức để kết nối, điều khiển phần cứng và phần mềm

II. Hạn chế của đồ án

- Robot được thiết kế để giao hàng cả trong nhà và ngoài trời nhưng nhóm đồ án mới chỉ nắm được phần điều khiển với môi trường trong nhà.
- Robot chưa có các cảm biến hỗ trợ an toàn để phù hợp với việc hoạt động trong môi trường làm việc cùng với con người.
- Thuật toán tìm đường chưa được tối ưu với hệ dẫn động.

III. Kết luận

Người duyệt đồng ý đề sinh viên được bảo vệ để tài tốt nghiệp trước Hội đồng
chấm đồ án tốt nghiệp.

Danh giá: điểm

Hà Nội, ngày 10 tháng 8 năm 2023

Giáo viên phản biện



Dương Văn Lạc

ĐẠI HỌC BÁCH KHOA HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP

**Thiết kế chế tạo robot giao hàng linh hoạt dùng
bánh xe quay swerve drive**

ĐÀM QUỐC KHÁNH

khanh.dq184499@sis.hust.edu.vn

HÀ LỤC MINH VŨ

vu.hlm185317@sis.hust.edu.vn

NGÔ XUÂN THÀNH

thanh.nx184621@sis.hust.edu.vn

NGUYỄN HOÀNG LONG

long.nh185274@sis.hust.edu.vn

Ngành Kỹ thuật Cơ điện tử

Giảng viên hướng dẫn: TS. Bùi Đình Bá **Chữ ký của GVHD**

Khoa: Cơ điện tử

Trường: Cơ khí

HÀ NỘI, 8/2023

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP
(NGÀNH KỸ THUẬT CƠ ĐIỆN TỬ)

1. Thông tin về sinh viên:

Họ và tên SV: Đàm Quốc Khanh	Lớp: ME1-03-K63	ĐT: 0972 546 290
Họ và tên SV: Hà Lực Minh Vũ	Lớp: ME-E1-01-K63	ĐT: 0962 378 584
Họ và tên SV: Ngô Xuân Thành	Lớp: ME1-04-K63	ĐT: 0359 657 985
Họ và tên SV: Nguyễn Hoàng Long	Lớp: ME-E1-02-K63	ĐT: 0333 187 370

Email (đại diện): khanh.dq184499@sis.hust.edu.vn

Hệ đào tạo: Chính quy Chuyên ngành: Cơ điện tử

Đồ án tốt nghiệp được thực hiện tại: Thư viện TQB phòng 809

Thời gian làm ĐATN: Từ ngày 15/03/2023 đến 03/08/2023

2. Tên đề tài:

Thiết kế chế tạo robot giao hàng linh hoạt dùng bánh xe quay swerve drive

Thông số đề tài:

- Phân tích nguyên lý: Tổng quan về hệ thống, nguyên lý hoạt động.
- Thiết kế và chế tạo xe: Thiết kế hệ thống bánh xoay độc lập, khung xe,...
- Xây dựng bản vẽ thiết kế: Xây dựng bản vẽ lắp 2D/3D
- Thiết kế hệ thống điều khiển
- Lập trình điều khiển: truyền nhận dữ liệu, giao diện điều khiển.

Hà Nội, ngày tháng năm 2023

Giảng viên hướng dẫn

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
(*Dành cho Giảng viên hướng dẫn*)

Tên đề tài:

Thiết kế chế tạo robot giao hàng linh hoạt dùng bánh xe quay swerve drive

Họ và tên SV: **Đàm Quốc Khánh** Lớp: ME1-03-K63

Họ và tên SV: **Hà Lực Minh Vũ** Lớp: ME-E1-01-K63

Họ và tên SV: **Ngô Xuân Thành** Lớp: ME1-04-K63

Họ và tên SV: **Nguyễn Hoàng Long** Lớp: ME-E1-02-K63

Chuyên ngành: Kỹ thuật Cơ điện tử

Giảng viên hướng dẫn: TS. Bùi Đình Bá

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

I. Tác phong làm việc

Thành viên nhóm đồ án có tinh thần làm việc tốt, chăm chỉ, hoàn thành tốt những yêu cầu được đề ra

II. Những kết quả đạt được

Nhóm đồ án đã xây dựng được một hệ thống robot tự hành hoàn chỉnh trong thời gian quy định của đồ án. Năm được những kiến thức về các thành phần của một hệ thống cơ điện tử

III. Hạn chế của đồ án

Đồ án cần cải thiện thêm các tính năng cũng như tích hợp thêm các cảm biến an toàn để phù hợp hơn với môi trường làm việc của robot.

IV. Kết luận

Người hướng dẫn đề nghị cho phép sinh viên được bảo vệ đề tài tốt nghiệp trước Hội đồng chấm đồ án tốt nghiệp.

Đánh giá: điểm

Hà Nội, ngày tháng năm 2023

Giảng viên hướng dẫn

NHẬN XÉT ĐỒ ÁN TỐT NGHIỆP
(Dành cho Giảng viên phản biện)

Tên đề tài:

Thiết kế chế tạo robot giao hàng linh hoạt dùng bánh xe quay swerve drive

Họ và tên SV: **Đàm Quốc Khánh** Lớp: ME1-03-K63

Họ và tên SV: **Hà Lực Minh Vũ** Lớp: ME-E1-01-K63

Họ và tên SV: **Ngô Xuân Thành** Lớp: ME1-04-K63

Họ và tên SV: **Nguyễn Hoàng Long** Lớp: ME-E1-02-K63

Chuyên ngành: Kỹ thuật Cơ điện tử

Giảng viên hướng dẫn: TS. Bùi Đình Bá

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

I. Những kết quả đạt được

- Nhóm đồ án nắm được nguyên lý, kết cấu và chế tạo được cơ cấu dẫn động swerve drive
- Chế tạo được mô hình robot giao hàng có khả năng di chuyển linh hoạt
- Thiết lập được các giao thức để kết nối, điều khiển phần cứng và phần mềm

II. Hạn chế của đồ án

- Robot được thiết kế để giao hàng cả trong nhà và ngoài trời nhưng nhóm đồ án mới chỉ nắm được phần điều khiển với môi trường trong nhà.
- Robot chưa có các cảm biến hỗ trợ an toàn để phù hợp với việc hoạt động trong môi trường làm việc cùng với con người.
- Thuật toán tìm đường chưa được tối ưu với hệ dẫn động.

III. Kết luận

Người duyệt đồng ý đề sinh viên được bảo vệ đề tài tốt nghiệp trước Hội đồng
chấm đồ án tốt nghiệp.

Đánh giá: điểm

Hà Nội, ngày tháng năm 2023

Giáo viên phản biện

LỜI MỞ ĐẦU

Thời đại công nghệ phát triển từng ngày tạo ra những nhu cầu mới, những vấn đề mới cần giải quyết. Với các công nghệ về trí tuệ nhân tạo và Robotics, chúng ta đã có những đột phá trong nhiều lĩnh vực trong cuộc sống. Hiện nay, rất nhiều công việc trong cuộc sống đã được thay thế bởi robot tự động, với những ưu điểm về độ chính xác và khả năng hoạt động trong thời gian dài. Một trong số những ứng dụng của robot có thể kể đến là lĩnh vực giao nhận hàng hóa, công việc đòi hỏi tính chính xác và hiệu quả. Với sự phát triển của thương mại điện tử và mô hình kinh doanh trực tuyến thì nhu cầu vận chuyển hàng hóa cũng tăng theo.

Trong bối cảnh đó, đề tài về “Robot giao hàng tự động” đã thu hút sự quan tâm đặc biệt của nhóm đồ án. Nhóm đồ án nhận thấy rằng việc sử dụng robot để thực hiện tác vụ vận chuyển hàng hóa không những giúp tiết kiệm thời gian và công sức của con người mà còn đảm bảo tính chính xác và an toàn trong quá trình vận chuyển.

Đề tài của nhóm đồ án được thực hiện dưới sự hướng dẫn của TS. Bùi Đình Bá. Cảm ơn thầy đã đồng hành cùng nhóm đồ án trong quá trình nghiên cứu và hoàn thành đồ án. Đồng thời nhóm đồ án cũng cảm ơn các thầy cô giúp đỡ trong quá trình học tập và nghiên cứu tại Đại học Bách khoa Hà Nội.

Báo cáo của nhóm đồ án sẽ trình bày chi tiết về quá trình nghiên cứu và triển khai đồ án. Với một dự án có khối lượng kiến thức lớn như đồ án tốt nghiệp, nhóm đồ án đã cố gắng áp dụng những kiến thức, kinh nghiệm mình đã có và đã tìm hiểu để hoàn thành đề tài đúng thời hạn. Tuy nhiên, do hạn chế về những kiến thức nền trong quá trình làm việc sẽ có những sai sót, nhóm đồ án mong muốn sẽ nhận được những đánh giá và những góp ý để nhóm có thể hoàn thiện sản phẩm của mình. Hy vọng đề tài này sẽ đem lại những thông tin hữu ích cũng như cung cấp cái nhìn sâu hơn về tiềm năng và ứng dụng của robot trong đời sống của con người.

Xin chân thành cảm ơn sự quan tâm và hỗ trợ từ phía các thầy cô, bạn bè và gia đình trong suốt quá trình thực hiện đồ án tốt nghiệp này.

Tóm tắt nội dung đồ án

Trong đồ án tốt nghiệp này, nhóm đồ án đặt mục tiêu nghiên cứu, thiết kế và triển khai một hệ thống Mobile robot ứng dụng trong việc giao và nhận hàng trong khuôn viên vừa và nhỏ như trường học hoặc khu dân cư. Với mục tiêu đã đề ra, nhóm đồ án đã đặt ra những yêu cầu sau:

- Robot có cảm biến lidar để tạo bản đồ và tránh vật cản.
- Có khả năng di chuyển linh hoạt trong các khu vực hẹp như ngõ ngách hay hành lang.
- Robot có thiết kế cơ động phù hợp với địa hình ngoài trời lẫn trong nhà.

Từ các yêu cầu đã đặt ra nhóm tiến hành hoàn thiện đồ án với các công việc được đặt ra cho từng thành viên và hoàn thành với sản phẩm cuối cùng đạt được những tiêu chí đúng với yêu cầu đề ra:

- Thiết kế cơ khí và hệ thống điện sử dụng phần mềm Solidworks và Autocad.
- Lập trình vi điều khiển.
- Lập trình và mô phỏng hệ thống robot sử dụng framework Robot Operating System (ROS).
- Gia công và lắp ráp sản phẩm.

Với đồ án mà nhóm đã chọn thì cũng có nhiều mặt hạn chế do khả năng gia công và lắp ráp nên sản phẩm chưa có được sự hoàn thiện về mặt thẩm mỹ. Độ chính xác cũng như đồ tin cậy không cao do giới hạn về các thiết bị phần cứng còn hạn chế.

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI	1
1.1. Giới thiệu chung.....	1
1.2. Tổng quan về robot tự hành	1
1.3. Các thành phần quan trọng của robot tự hành.....	2
1.4. Ứng dụng của robot tự hành.....	3
1.5. Những lợi ích mà robot tự hành mang lại	5
1.6. Đặt vấn đề	5
CHƯƠNG 2: THIẾT KẾ MÔ HÌNH ROBOT.....	7
2.1. Giới thiệu chương	7
2.2. Yêu cầu thiết kế.....	7
2.2.1. Ứng dụng của robot	7
2.2.2. Yêu cầu về kỹ thuật	7
2.3. Thiết kế cơ khí.....	7
2.3.1. Phân tích và lựa chọn hệ thống dẫn động cho mô hình robot	8
2.3.2. Các thành phần của hệ thống dẫn động swerve drive.....	15
2.3.3. Thiết kế hệ thống giảm xóc cho module swerve drive	16
2.3.4. Thông số động cơ bánh xe chủ động	22
2.3.5. Thiết kế hệ thống chuyển hướng cho module swerve drive	23
2.3.6. Thiết kế hệ thống khung và kiểm tra	30
2.3.7. Thiết kế tổng thể của robot	31
CHƯƠNG 3: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN.....	34
3.1. Giới thiệu chung.....	34
3.2. Các thành phần của hệ thống điều khiển.....	34
3.2.1. Vi điều khiển.....	34
3.2.2. Driver điều khiển động cơ DC.....	36
3.2.3. Động cơ encoder DC	39
3.2.4. Công tắc hành trình.....	39
3.2.5. Driver động cơ BLDC	40
3.2.6. Động cơ BLDC	41

3.3. Lập trình điều khiển	42
3.3.1. Động học ngược của robot swerve drive.....	42
3.3.2. Động học thuận của robot swerve drive	45
3.3.3. Xác định vị trí của robot.....	46
3.3.4. Thực hiện trên vi điều khiển (Microcontroller):	47
3.3.5. Điều khiển vị trí của động cơ encoder DC	48
3.3.6. Điều khiển động cơ BLDC.....	55
3.3.7. Giao thức kết nối giữa vi điều khiển và PC.....	56
3.3.8. Khung truyền (Message Frame)	57
3.3.9. Chu trình điều khiển của hệ thống.....	57
CHƯƠNG 4: ROS VÀ THUẬT TOÁN ĐIỀU KHIỂN.....	59
4.1. Tổng quan về Linux và Robot Operating System (ROS)	59
4.2. Tổng quan về navigation stack	62
4.3. Tổng quan về SLAM	67
4.3.1. Phương pháp global planner.....	70
4.3.2. Phương pháp local planner	76
4.4. Ứng dụng các thuật toán điều khiển và thực nghiệm	77
4.4.1. Xây dựng bản đồ sử dụng Slam gmapping.....	78
4.4.2. Cấu hình Navigation.....	79
4.5. Xây dựng giao diện người dùng cho robot	81
4.5.1. Ý tưởng về giao diện	81
4.5.2. Thiết kế giao diện	82
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	89
5.1. Kết luận.....	89
5.2. Hướng phát triển	89
TÀI LIỆU THAM KHẢO	91

DANH MỤC HÌNH VẼ

Hình 1.1 Robot tự vận hành trong nhà kho	3
Hình 1.2 Robot tự hành ứng dụng trong các dịch vụ	3
Hình 1.3 Ứng dụng trong công nghệ sinh học	4
Hình 1.4 Mô hình robot điện hình trong việc nghiên cứu nâng cao	5
Hình 2.1:Robot Differential Drive	8
Hình 2.2: Robot Skid Steering	9
Hình 2.3: Robot dạng Ackerman	10
Hình 2.4: Robot đa hướng	11
Hình 2.5: Robot đa hướng sử dụng Swerve drive	13
Hình 2.6: Mô hình hệ thống swerve drive	16
Hình 2.7: Hệ thống giảm xóc trên oto	17
Hình 2.8: Hình học của hệ thống giảm xóc	18
Hình 2.9: Hệ thống giảm xóc của xe máy Vespa	18
Hình 2.10: Mô tả nguyên lý hoạt động của giảm xóc xe Vespa	19
Hình 2.11: Hệ thống giảm xóc được sử dụng trong mô hình của MRV	19
Hình 2.12: Thiết kế hệ thống giảm xóc của nhóm thiết kế	20
Hình 2.13: Thông số hình học của tay đòn hệ thống giảm xóc	21
Hình 2.14: Động cơ BLDC	22
Hình 2.15: Kết quả momen quán tính từ Solidworks	23
Hình 2.16: Vị trí hệ tọa độ và trực quay dùng để tính momen quán tính	24
Hình 2.17: Mô tả tiếp xúc giữa 2 vật thể theo tài liệu	25
Hình 2.18: Động cơ JBG37-545	26
Hình 2.19: Mô hình hệ thống chuyển hướng	30
Hình 2.20: Hệ thống khung xe	31
Hình 2.21: Mô hình tổng thể của robot	32
Hình 2.22: Mô hình thực tế của robot	33
Hình 3.1: Những loại vi điều khiển phổ biến	34
Hình 3.2: STM32F401CCU	35
Hình 3.3: Sơ đồ cấu hình của STM32F401CCU	36
Hình 3.4: Driver BTS7960	37

Hình 3.5: Sơ đồ kết nối của driver	38
Hình 3.6: Sơ đồ kết nối của động cơ encoder	39
Hình 3.7: Công tắc hành trình quang	40
Hình 3.8: Driver BLDC	41
Hình 3.9: Động cơ BLDC	41
Hình 3.10: Vị trí các bánh xe	42
Hình 3.11: Tính toán hướng và tốc độ cho mỗi module bánh xe	43
Hình 3.12: Liên hệ vận tốc tại tâm với vận tốc của từng bánh xe	47
Hình 3.13: Giới hạn góc quay của bánh xe	47
Hình 3.14: Tối ưu hóa góc trong chương trình	48
Hình 3.15: Giới hạn vận tốc	48
Hình 3.16: Mô hình hóa của bộ điều khiển PID	49
Hình 3.17: Đáp ứng của hệ thống khi không sử dụng PID	52
Hình 3.18: Đáp ứng của hệ thống với bộ điều khiển PID thích hợp	52
Hình 3.19: Nguyên lý của encoder	53
Hình 3.20: Nguyên lý điều khiển độ rộng xung	54
Hình 3.21: Trường từ Rotor và Trường từ Stator	55
Hình 3.22: Driver BLDC	56
Hình 3.23: Khung thông điệp MCU gửi cho driver	56
Hình 3.24: Khung thông điệp driver gửi cho MCU	56
Hình 3.25: Khung thông điệp từ PC tới MCU	57
Hình 3.26: Phản hồi từ MCU tới PC	57
Hình 3.27: Sơ đồ thuật toán điều khiển robot	58
Hình 4.1: FILESYSTEM LEVEL của ROS	59
Hình 4.2: Cây thư mục trong ROS	60
Hình 4.3: Các thành phần chính của ROS	60
Hình 4.4: Giao tiếp giữa các node trong ROS	61
Hình 4.5: Sơ đồ khối move_base	63
Hình 4.6: Local plan (lục) global plan (lam)	64
Hình 4.7: Global costmap và local cost map	64
Hình 4.8: Mô tả AMCL	66
Hình 4.9: Vị trí robot x_t và bản đồ m	68

Hình 4.10: Tổng quát về Hector Mapping and Navigation stack	70
Hình 4.11: Dijkstra Path Planning	71
Hình 4.12: Ví dụ về thuật toán Dijkstra trong bản đồ dạng lưới	72
Hình 4.13: Mô tả thuật toán Dijkstra	72
Hình 4.14: Kết quả mô phỏng thuật toán Dijkstra	73
Hình 4.15: A* Path Planning	74
Hình 4.16: Mô tả thuật toán A*	75
Hình 4.17: Kết quả mô phỏng thuật toán A*	76
Hình 4.18: Dữ liệu nhận được từ Lidar.....	78
Hình 4.19: Rviz mô phỏng lại dữ liệu nhận được từ scan	78
Hình 4.20: Map xây dựng được từ Slam gmapping.....	79
Hình 4.21: Kế hoạch đường đi với map có sẵn.....	80
Hình 4.22: Màn hình bắt đầu.....	82
Hình 4.23: Màn hình lựa chọn dịch vụ	83
Hình 4.24: Màn hình thanh toán	83
Hình 4.25: Màn hình xác nhận dịch vụ	84
Hình 4.26: Sơ đồ mô tả	87
Hình 5.1: Mô hình concept của robot giao hàng.....	90

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu chung

Chương này giới thiệu tổng quan về robot, đưa ra các nhiệm vụ robot thực hiện và yêu cầu kỹ thuật đồng thời trình bày nguyên lý hoạt động của robot.

1.2. Tổng quan về robot tự hành

Robot tự vận hành cũng giống như con người, cũng có khả năng đưa ra quyết định của riêng mình và sau đó thực hiện một hành động tương ứng. Robot tự vận hành có thể nhận thức môi trường của nó, đưa ra quyết định dựa trên những gì nó nhận thức và đã được lập trình để nhận ra các điều kiện và sau đó thực hiện một chuyển động hoặc thao tác trong môi trường đó.

Trong 15-20 năm qua, việc sử dụng các robot di động điều khiển từ xa được trang bị camera quan sát những thứ ngoài tầm với hoặc các ứng dụng công nghiệp rất phổ biến.

Ví dụ những sản phẩm robot tự vận hành như:

- Xe tự hành được sử dụng để di chuyển vật liệu trong nhà máy và nhà kho.
- Robot tự bay cũng đang được sử dụng để ứng phó với thảm họa.
- Robot tự hành dưới nước đang được sử dụng để tìm kiếm và phát hiện những con tàu đắm ở vùng sâu nhất độ sâu của đại dương của chúng ta.
- Robot tự hành trên bề mặt sao hỏa.
- Robot humanoid hỗ trợ vào các khu vực nguy hiểm.
- Robot chó tự vận hành hỗ trợ bảo vệ con người.
- Robot phục vụ tự vận hành hỗ trợ trong người trong các bệnh viện, robot phục vụ cửa hàng café.

Robot tự hành hiện nay thực sự là những cỗ máy thông minh có thể thực hiện các nhiệm vụ và hoạt động trong môi trường một cách độc lập, không có sự điều khiển hoặc can thiệp của con người. Mức độ tự chủ này mang lại cho lực lượng lao động khả năng giao các nhiệm vụ nhảm chán, nguy hiểm hoặc bẩn thỉu cho robot để họ có thể dành nhiều thời gian hơn để thực hiện những phần thú vị, hấp dẫn và có giá trị trong công việc mà họ có đủ khả năng duy nhất.

Robot tự vận hành ngày nay là sản phẩm không thể thiếu đối với con người chúng ta hiện nay, robot hỗ trợ còn người trong nhiều khía cạnh trong cuộc sống.

1.3. Các thành phần quan trọng của robot tự hành

Nhận thức của robot tự vận hành:

Đối với robot, nhận thức được thể hiện dưới dạng các cảm biến. Máy quét laser, camera nhìn nội (mắt), cảm biến va chạm (da và tóc), cảm biến mô-men xoắn (căng cơ) và thậm chí cả quang phổ kế (mùi) được sử dụng làm thiết bị đầu vào cho robot để giúp robot “nhìn thấy” và nhận biết môi trường. Với cả con người và robot, giờ đây chúng ta có thể nghĩ đến các loại đầu vào thông tin khác, như nguồn cung cấp dữ liệu vô tận từ internet.

Phán quyết để robot tự hành ra quyết định hoạt động:

Bộ não của robot thường là một máy tính và đưa ra quyết định dựa trên nhiệm vụ của nó và thông tin mà nó nhận được trên đường đi. Nhưng robot cũng có một khả năng tương tự như hệ thống thần kinh ở người, nơi hệ thống an toàn của chúng hoạt động nhanh hơn và không cần sự cho phép của não. Trên thực tế, trong robot, bộ não hoạt động với sự cho phép của hệ thống an toàn. Trong một robot tự động, nhóm đồ án gọi hệ thống “thần kinh” đó là một hệ thống nhúng. Nó hoạt động nhanh hơn và có quyền hạn cao hơn so với máy tính đang thực hiện kế hoạch nhiệm vụ và phân tích dữ liệu. Đây là cách robot có thể quyết định dừng lại nếu nó nhận thấy có chướng ngại vật cản đường mình, nếu nó tự phát hiện ra sự cố hoặc nếu nút dừng khẩn cấp được nhấn.

Hành động của robot tự hành:

Robot có thể có tất cả các loại thiết bị truyền động, và một số loại động cơ thường nằm ở trung tâm của thiết bị truyền động. Cho dù đó là bánh xe, thiết bị truyền động tuyến tính hay ram thủy lực, luôn có một động cơ chuyển đổi năng lượng thành chuyển động. Vì vậy, robot tự hành là robot tự quyết định hành động của nó dựa trên thông tin mà nó nhận được.

1.4. Ứng dụng của robot tự hành

Robot tự hành trong nhà kho



Hình 1.1: Robot tự vận hành trong nhà kho

Robot tự vận hành được sử dụng trong các ứng dụng kho bãi để nâng và vận chuyển hàng hóa nặng trong không gian. Thực hiện các robot tự vận hành với các hoạt động lưu kho cơ bản giúp giảm thời gian công nhân đi lại trong nhà kho, cho phép họ thực hiện nhiều nhiệm vụ có giá trị gia tăng hơn.

Robot tự hành ứng dụng trong các dịch vụ



Hình 1.2: Robot tự hành ứng dụng trong các dịch vụ

Ngày nay, có nhiều robot tự hành được sử dụng trong lĩnh vực như chăm sóc sức khỏe và ứng trong các dịch vụ trong các bệnh viện quán ăn, quán cà phê với các nhiệm vụ khác nhau.

Robot tự vận hành là một công cụ hữu ích để hợp lý hóa việc vận chuyển các món ăn đồ uống, vật tư và thuốc trong toàn bộ cơ sở chăm sóc sức khỏe. Điều này thậm chí còn quan trọng hơn trong các đơn vị bệnh truyền nhiễm, vì nó giúp y tá và người phục vụ không phải tiếp xúc thường xuyên với các chất gây ô nhiễm tiềm

ân, nhưng vẫn đảm bảo rằng bệnh nhân được điều trị thích hợp. Thứ hai, robot tự hành cũng có thể được sử dụng trong vệ sinh - robot khử trùng có thể được trang bị đèn UV diệt vi rút hoặc bình xịt khử nhiễm để làm sạch phòng hoặc không gian mà không làm cho con người có nguy cơ bị tổn thương.

Ứng dụng trong công nghệ sinh học

Robot di động tự động kết hợp với cánh tay robot có thể được sử dụng để kiểm soát các đầu vào có giá trị của quá trình, thực hiện các nhiệm vụ giám sát thường xuyên và quản lý an toàn việc loại bỏ chất thải khỏi dây chuyền sản xuất.

Với robot tự hành xử lý các công việc lặp đi lặp lại, người lao động có thể tập trung vào các bước quan trọng trong quy trình sản xuất được phẩm sinh học như theo dõi các thông số tăng trưởng, kiểm tra liên tục và thực hiện các điều chỉnh cần thiết khi quá trình phát triển tiến triển.



Hình 1.3: Ứng dụng trong công nghệ sinh học

Nghiên cứu và phát triển

Đối với nghiên cứu và phát triển, robot tự hành được sử dụng để giảm thiểu các nhiệm vụ vận chuyển tẻ nhạt liên quan đến thử nghiệm lặp đi lặp lại hoặc các yêu cầu kỹ thuật khác. Robot tự hành ngày càng trở thành một phần của chính nghiên cứu.



Hình 1.4: Mô hình robot điển hình trong việc nghiên cứu nâng cao

1.5. Những lợi ích mà robot tự hành mang lại

Với nhiều điểm mạnh đáng kể, robot tự vận hành mang lại cho chúng ta những lợi ích to lớn sau đây:

- Giảm chi phí vận hành.
- Đảm bảo tính đồng nhất và tăng chất lượng sản phẩm.
- Tăng năng suất.
- Tăng tính linh hoạt trong sản xuất.
- Tiết kiệm nguyên vật liệu và hạ giá thành sản phẩm.
- Tiết giảm chi phí, cải thiện chất lượng môi trường lao động.
- Nâng cao hiệu quả đầu tư của doanh nghiệp.
- Nâng cao uy tín thương hiệu, tăng sức cạnh tranh của doanh nghiệp.

1.6. Đặt vấn đề

Hệ thống robot giao hàng mang lại nhiều tiềm năng và lợi ích trong việc tăng cường hiệu suất và hiệu quả trong ngành giao nhận. Tuy nhiên, nó cũng đối mặt với một số vấn đề và thách thức, bao gồm:

- An ninh và độ an toàn: Robot giao hàng di chuyển trong môi trường công cộng và tương tác với con người. Vấn đề an ninh và độ an toàn trở thành một ưu tiên quan trọng, đảm bảo rằng robot không gây nguy hiểm cho con người và không trở thành mục tiêu của hành vi xấu.
- Pháp lý và quy định: Việc sử dụng robot giao hàng đòi hỏi tuân thủ các quy định và quyền lực pháp lý địa phương và quốc gia. Cần thiết phải có

khung pháp lý rõ ràng để đảm bảo hoạt động của robot tuân thủ các quy tắc và tiêu chuẩn an toàn liên quan.

- Tương tác với con người: Robot giao hàng thường phải tương tác với con người, bao gồm việc giao nhận hàng hóa, trao đổi thông tin và đáp ứng các yêu cầu. Để tạo ra trải nghiệm tương tác thuận lợi và tránh sự nhầm lẫn, robot cần có khả năng giao tiếp hiệu quả và nhận biết được các tình huống xã hội phức tạp.
- Đa dạng môi trường: Robot giao hàng phải hoạt động trong môi trường đa dạng, bao gồm nhiều loại địa hình, điều kiện thời tiết và cơ sở hạ tầng khác nhau. Cần phải thiết kế robot và hệ thống điều khiển sao cho phù hợp với các môi trường khác nhau và có khả năng thích ứng.
- Quản lý năng lượng và sạc pin: Robot giao hàng yêu cầu nguồn năng lượng để hoạt động. Việc quản lý năng lượng và sạc pin hiệu quả là một thách thức quan trọng, đảm bảo robot có thể hoạt động liên tục và hiệu quả trong quá trình giao hàng.
- Độ tin cậy và bảo trì: Robot giao hàng cần được thiết kế để đảm bảo độ tin cậy cao và ít lỗi. Đồng thời, việc bảo trì và sửa chữa robot cũng là một yếu tố quan trọng để đảm bảo hoạt động liên tục và giảm thời gian chết.

Trong giới hạn của đồ án này nhóm đồ án sẽ trình bày quy trình nhóm xây dựng một mô hình hệ thống robot tự hành với ứng dụng dùng trong công việc vận chuyển hàng hóa. Qua đề tài này nhóm đồ án mong muốn có thể mang đến một ứng dụng mới trong lĩnh vực mobile robot và đồng thời đề xuất những ý tưởng để giải quyết những vấn đề mà những hệ thống mobile robot đang gặp phải.

CHƯƠNG 2: THIẾT KẾ MÔ HÌNH ROBOT

2.1. Giới thiệu chương

Với một hệ thống robot tự động, độ chính xác của robot phần lớn đến từ phần cứng của robot. Chương này sẽ trình bày cách nhóm thiết kế các thành phần phần cứng của robot giao hàng tự động từ tính chọn các chi tiết cơ khí, kiểm tra các thành phần đánh giá về ứng suất và lực, thiết kế hệ thống điện, cách chọn các thành phần điện tử,...

2.2. Yêu cầu thiết kế

2.2.1. Ứng dụng của robot

Trong các khu vực như trường học hay những khu đô thị, việc các dịch vụ giao hàng bị giới hạn do những yêu cầu về an ninh của khu vực, hay như việc giao nhận các gói hàng trong các khu vực bị cách ly do các yếu tố về dịch bệnh hay môi trường ô nhiễm,... Những yêu cầu trên đặt ra việc thiết kế một robot có thể thay thế con người làm việc trong những môi trường kể trên. Robot cần phải đạt được những yêu cầu sau:

- Robot có khả năng di chuyển linh hoạt dưới điều kiện ngoài trời lẫn trong nhà.
- Robot có khả năng tải lớn, có sức chứa lớn để giao được nhiều hàng hóa.
- Robot có khả năng tránh vật cản, di chuyển theo tuyến đường được đề ra.

2.2.2. Yêu cầu về kỹ thuật

Với những yêu cầu đã đặt ra nhóm thiết kế robot giao hàng gồm các thông số về kỹ thuật như sau:

- Tải trọng tối đa 30 kg
- Tốc độ tối đa: 6 km/h
- Hệ thống dẫn động đa hướng
- Có hệ thống giảm xóc phù hợp với điều kiện làm việc ngoài trời với địa hình không bằng phẳng
- Các cảm biến Lidar dùng để mapping và navigation

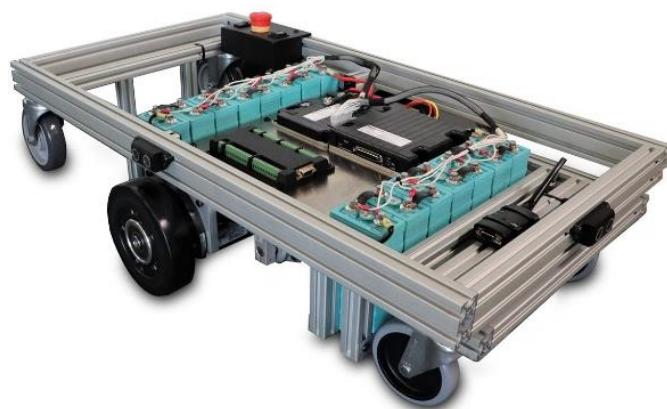
2.3. Thiết kế cơ khí

2.3.1. Phân tích và lựa chọn hệ thống dẫn động cho mô hình robot

Robot tự hành di chuyển bằng hệ thống bánh xe. Do đó, phần lớn sự linh hoạt cũng như độ chính xác của robot ảnh hưởng tương tác giữa bánh xe và bề mặt của địa hình di chuyển. Đã có rất nhiều hệ thống dẫn động đã được sử dụng trong lĩnh vực robot tự hành, nhóm đã tiến hành nghiên cứu và đánh giá các hệ thống đó và đưa ra những ưu và nhược điểm của từng hệ thống.

a. Hệ dẫn động 2 bánh vi sai (Differential Drive Robot)

Hệ thống dẫn động Differential Drive là một loại hệ thống dẫn động phổ biến được sử dụng trong robot di động và xe tự lái. Hệ thống này sử dụng cơ cấu Differential để điều khiển hai bánh xe riêng biệt, mỗi bánh xe có thể quay với tốc độ khác nhau và hướng di chuyển độc lập.



Hình 2.1: Robot Differential Drive

Ưu điểm:

- Đơn giản và kinh tế: Hệ thống Differential Drive có cấu trúc đơn giản và ít thành phần, giúp giảm chi phí sản xuất và bảo trì.

- Dễ điều khiển: Với hai bánh xe có thể quay độc lập, hệ thống Differential Drive cho phép điều khiển khá chính xác hướng di chuyển và vị trí của robot. Điều này giúp robot thực hiện các chuyển động quay, tiến lùi và di chuyển đa hướng một cách linh hoạt.
- Hiệu suất di chuyển tốt: Hệ thống Differential Drive cho phép robot di chuyển mượt mà và linh hoạt trên địa hình bằng phẳng. Nó cung cấp khả năng xoay chính xác trên chỗ và vượt qua các rào cản trong môi trường.

Nhược điểm:

- Hạn chế di chuyển: Robot sử dụng hệ thống hai bánh xe khó di chuyển trên mặt phẳng không bằng phẳng hoặc vượt qua các rào cản cao. Điều này hạn chế ứng dụng của hệ thống trong nhiều môi trường.

b. Hệ dẫn động skid steering

Dạng robot này cũng có cấu tạo gần giống với robot Differential Drive nhưng có thêm 2 bánh chủ động nữa, với hệ dẫn động 4 bánh thì động học của robot có phần khác so với cấu hình 2 bánh.



Hình 2.2: Robot Skid Steering

Ưu điểm:

- Khả năng cân bằng tốt.
- Khả năng di chuyển địa hình tốt, phù hợp với nhiều tác vụ [1].
- Hệ dẫn động đơn giản nên robot có cấu tạo vững chắc [1].

Nhược điểm:

- Do chuyển động của robot gồm các chuyển động trượt của các bánh xe chủ động nên việc tính toán, dự đoán vị trí của robot trở nên khó khăn trong những điều kiện khác nhau [2].
- Công suất động cơ lớn tùy thuộc vào tải và hệ số ma sát của bánh xe với địa hình [2].

c. Hệ dẫn động dạng Ackerman

Hệ thống dẫn động Ackerman là một hệ thống dẫn động được sử dụng phổ biến trong các phương tiện di chuyển có bánh lái, chẳng hạn như ô tô. Hệ thống này dựa trên nguyên tắc Ackerman, trong đó các bánh xe được điều khiển sao cho các đường kính của chúng đạt được một điểm giao chung, gọi là tâm quay.



Hình 2.3: Robot dạng Ackerman

Ưu điểm:

- Đơn giản và đáng tin cậy: Hệ thống dẫn động Ackerman có thiết kế đơn giản, với các thành phần chủ yếu bao gồm cơ cấu bánh lái và các khớp nối. Điều này giúp hệ thống đạt được tính tin cậy cao và ít gặp sự cố hỏng hóc.
- Điều khiển ổn định: Hệ thống Ackerman cho phép điều khiển phương tiện một cách ổn định và dễ dàng, đặc biệt khi thực hiện các chuyển động quay và quay đầu.
- Khả năng vận hành tốt ở tốc độ cao: Hệ thống Ackerman được tối ưu hóa để hoạt động tốt ở tốc độ cao, giúp duy trì sự ổn định và khả năng kiểm soát trong các tình huống di chuyển nhanh.
- Tích hợp dễ dàng: Với thiết kế đơn giản, hệ thống dẫn động Ackerman dễ dàng tích hợp vào các phương tiện và các hệ thống điều khiển tự động.

Nhược điểm:

- Hạn chế di chuyển bên trong: Do nguyên lý Ackerman, các bánh xe không thể quay tới góc cạnh như nhau khi di chuyển trong không gian hẹp, như khi thực hiện quay đầu hoặc đi qua các góc nhỏ. Điều này có thể làm hạn chế khả năng di chuyển trong một số tình huống cụ thể.
- Mòn bánh xe: Vì các bánh xe di chuyển với bán kính khác nhau khi thực hiện các chuyển động quay, nên có thể gây mòn không đồng đều và nhanh chóng trên bệ mặt liên hệ của bánh xe.
- Hạn chế trong điều khiển tuyến tính: Hệ thống Ackerman có đặc điểm điều khiển phi tuyến tính, có thể tạo ra sai số khi điều khiển chính xác theo góc và tốc độ.

d. Hệ dẫn động đa hướng (Omnidirectional robot)

Robot đa hướng là loại robot sử dụng các bánh xe đặc biệt (Omni hoặc Mecanum) được thiết kế giúp robot có khả năng di chuyển theo nhiều hướng.



Hình 2.4: Robot đa hướng

Ưu điểm:

- Độ linh hoạt vượt trội so với các dạng robot sử dụng bánh xe bình thường.
- Di chuyển đa hướng: Hệ thống này cho phép robot di chuyển ở nhiều hướng khác nhau, bao gồm di chuyển tiến, lùi, quay trái/phải và chuyển động chéo. Điều này tạo ra khả năng linh hoạt và đa dạng trong việc điều hướng và vận chuyển hàng hóa trong không gian hẹp. Có khả năng di chuyển qua các khu vực hẹp nhờ khả năng di chuyển đa hướng.

Nhược điểm:

- Robot thực hiện các chuyển động đa hướng nhờ ma sát trượt giữa con lăn và mặt đường nên độ ổn định không cao, có hiện tượng rung do kết cấu của các bánh xe.
- Khó có thể di chuyển trên các bề mặt phức tạp do giới hạn của các bánh xe đa hướng.
- Hệ thống bánh xe cầu tạo phức tạp nên có chi phí cao hơn so với loại bánh xe thông thường.

e. Hệ dẫn động 4 bánh độc lập (Swerve drive robot)



Hình 2.5: Robot đa hướng sử dụng Swerve drive

Hệ dẫn động Swerve Drive là một loại hệ thống dẫn động đặc biệt được sử dụng trong robot để cung cấp khả năng di chuyển linh hoạt và điều khiển chính xác trên nhiều địa hình. Hệ thống này bao gồm bốn bánh xe độc lập, mỗi bánh có thể quay và di chuyển một cách độc lập với các góc và tốc độ khác nhau. Điều này cho phép robot thực hiện các chuyển động điều hướng đa hướng, di chuyển ngang và xoay 360 độ một cách thuận tiện.

Ưu điểm:

- Độ cơ động cao: Hệ thống Swerve Drive cung cấp khả năng di chuyển tốt trên mọi địa hình, bao gồm cả địa hình gồ ghề, đồng thời cho phép robot thực hiện các chuyển động phức tạp như đi lại, quay đầu và di chuyển đa hướng.
- Điều khiển chính xác: Các bánh xe độc lập của Swerve Drive cho phép điều khiển chính xác hướng di chuyển và vị trí của robot. Điều này rất hữu ích trong các ứng dụng yêu cầu định vị chính xác và quỹ đạo di chuyển linh hoạt.

- **Khả năng vượt rào cản:** Với khả năng quay và di chuyển độc lập, robot sử dụng Swerve Drive có thể vượt qua các rào cản và tránh những vật cản trong môi trường.
- **Linh hoạt trong tương tác:** Swerve Drive cho phép robot di chuyển qua các hành lang nhỏ và thực hiện các chuyển động xoay 360 độ một cách dễ dàng. Điều này tạo ra khả năng tương tác linh hoạt và đa dạng trong các tác vụ cụ thể.

Nhược điểm:

- Phức tạp và tốn kém: hệ thống swerve drive gồm nhiều các thành phần hơn đáng kể so với các hệ thống khác.
- Tiêu thụ năng lượng: hệ thống Swerve Drive có thể tiêu tốn nhiều năng lượng hơn so với các hệ thống dẫn động khác, do phải cung cấp sức mạnh cho nhiều động cơ và các cơ cấu di chuyển độc lập.
- Độ bền và bảo trì: Vì tính phức tạp của hệ thống, Swerve Drive có thể đòi hỏi sự bảo trì và kiểm tra định kỳ để đảm bảo hoạt động ổn định và tránh hỏng hóc.

Sau quá trình nghiên cứu kỹ lưỡng và tìm hiểu sâu về các loại robot có sử dụng các hệ dẫn động khác nhau, cùng việc đánh giá kỹ các ưu điểm và nhược điểm của từng hệ thống dẫn động, nhóm đồ án đã quyết định lựa chọn hệ dẫn động 4 bánh độc lập Swerve Drive với niềm tin rằng hệ thống này sẽ mang lại nhiều lợi ích vượt trội.

Ý tưởng chính khi nhóm chọn hệ dẫn động Swerve Drive là khai thác sự linh hoạt và động lực của cơ cấu này, giúp robot có khả năng di chuyển trên nhiều địa hình đa dạng một cách hiệu quả. Hệ thống 4 bánh độc lập cho phép robot di chuyển một cách linh hoạt và chính xác, cho phép robot của nhóm đồ án vượt qua các rào cản và khó khăn trong môi trường mà robot phải hoạt động.

Đặc biệt, hệ dẫn động Swerve Drive cho phép robot di chuyển đa hướng một cách thuận tiện và mạnh mẽ. Nhờ vào khả năng này, robot của nhóm đồ án có thể đi qua các hành lang nhỏ và thậm chí thực hiện các chuyển động xoay 360 độ mà không gặp khó khăn. Điều này tạo ra sự linh hoạt tuyệt vời cho robot và mở ra nhiều khả năng trong việc vận hành và tương tác với môi trường xung quanh.

Với việc sử dụng hệ dẫn động Swerve Drive, nhóm đồ án tin rằng robot của nhóm sẽ được trang bị một công nghệ di chuyển tiên tiến và linh hoạt, cho phép chúng thích ứng và hoạt động tốt trên nhiều địa hình và trong các tình huống phức tạp. Sự lựa chọn này đồng thời có thể là một ý tưởng giúp mở ra nhiều tiềm năng và cơ hội phát triển trong lĩnh vực robot hóa và tự động hóa trong tương lai.

2.3.2. Các thành phần của hệ thống dẫn động swerve drive

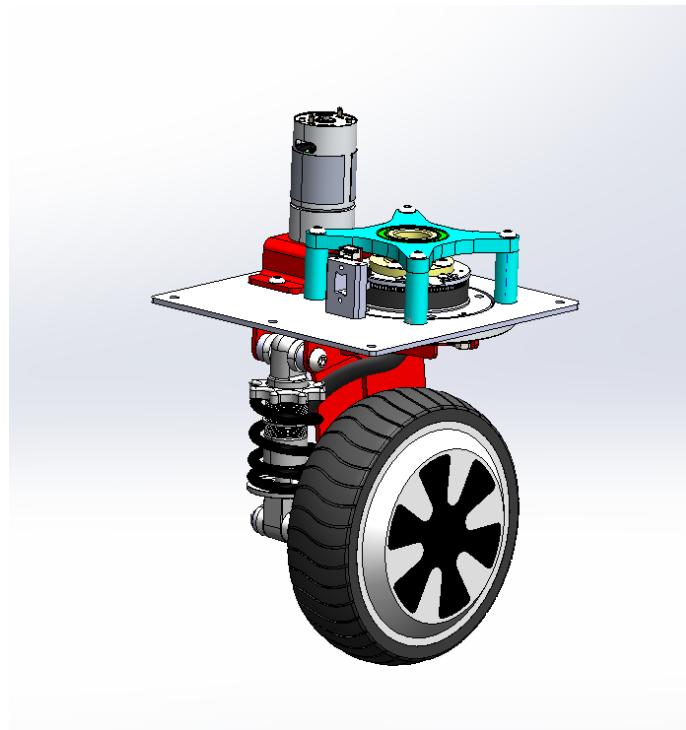
Hệ thống dẫn động swerve drive là một hệ thống khá phức tạp gồm nhiều thành phần và mỗi thành phần đều đóng vai trò quan trọng trong khả năng hoạt động chính xác và linh hoạt của toàn hệ thống. Qua việc tìm hiểu và nghiên cứu các hệ thống swerve drive đã có trước đó nhóm đã thống kê những thành phần cần phải có của một hệ thống swerve drive.

Cấu tạo của một hệ thống swerve drive sẽ thường bao gồm 3 đến 4 module swerve drive và mỗi module sẽ gồm những thành phần sau:

- Động cơ lái: Động cơ này được sử dụng để điều khiển góc lái của module. Nó cho phép module quay độc lập và thay đổi hướng di chuyển.
- Động cơ dẫn động: Động cơ dẫn động cung cấp công suất và mô-men xoắn cần thiết để đẩy module và cả robot. Nó chuyển đổi năng lượng điện thành chuyển động cơ học.
- Encoder: Hai encoder được sử dụng để đo chính xác hướng quay và vận tốc của bánh xe động lực. Encoder cung cấp thông tin phản hồi cho hệ thống điều khiển, cho phép điều khiển chính xác các chuyển động của module.
- Vòng bi: Vòng bi bánh xe đảm bảo sự chính xác cũng như sự mượt mà cho những chuyển động quay của module.
- Hệ thống truyền động: Hệ thống truyền động bao gồm các bánh răng, pulley hoặc đai truyền động chuyển động từ động cơ dẫn động sang bánh xe. Nó cung cấp tốc độ và mô-men xoắn cần thiết cho chuyển động của module.
- Khung bánh xe dẫn động: Khung bánh xe dẫn động giữ bánh xe và kết nối nó với hệ thống truyền động. Nó cung cấp sự hỗ trợ và ổn định cho toàn bộ module.

- Khung giữ hệ thống lái: Khung giữ hệ thống lái hỗ trợ động cơ lái và cho phép nó quay độc lập. Nó tạo điểm quay cho chuyển động lái.

Nhóm đã thiết kế hệ thống swerve drive như sau:



Hình 2.6: Mô hình hệ thống swerve drive

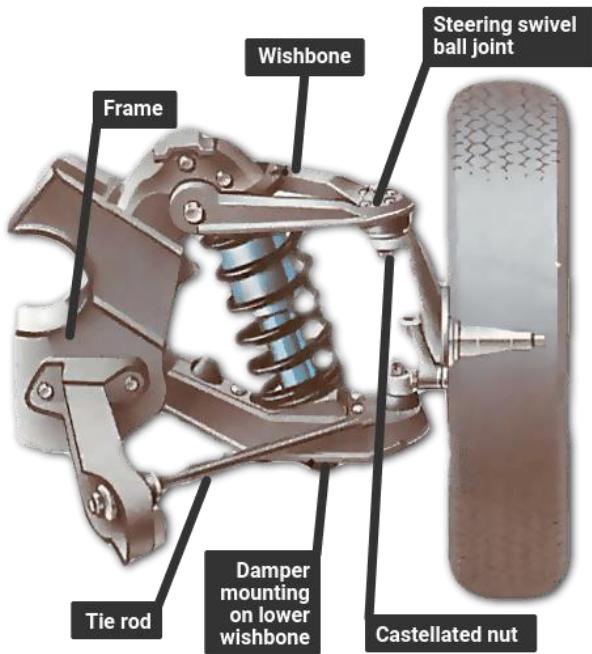
2.3.3. Thiết kế hệ thống giảm xóc cho module swerve drive

a. Lựa chọn cơ cấu giảm xóc phù hợp

Với yêu cầu robot có thể vận hành được trong những môi trường khác nhau nhóm đã tiến hành nghiên cứu và chế tạo hệ thống giảm xóc cho module swerve drive. Hệ thống giảm xóc giúp cho 4 bánh của robot luôn chạm đất để giảm thiểu tình trạng trượt từ đó cải thiện độ chính xác cho robot.

Nhóm đã tiến hành nghiên cứu các hệ thống giảm xóc đã được thiết kế trước đó. Nhóm nhận thấy có 2 thiết kế giảm xóc mà nhóm đồ án nhận thấy có thể áp dụng cho mô hình của mình.

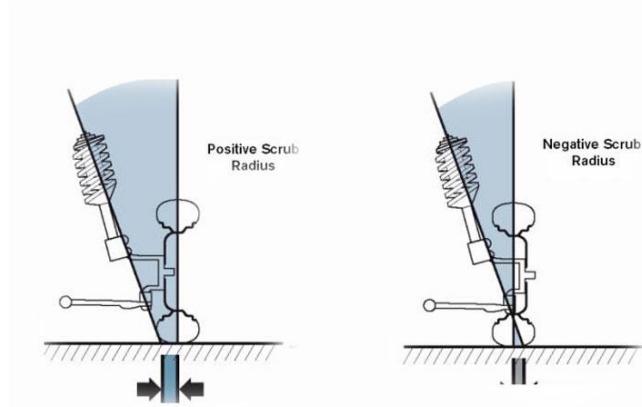
Đầu tiên là hệ thống giảm xóc dạng tay đòn đôi. Hệ thống này giống với hệ thống giảm xóc thường thấy trong lĩnh vực ô tô. Với nguyên lý hệ 4 khâu 4 khớp.



Hình 2.7: Hệ thống giảm xóc trên ô tô

Với ứng dụng trong ngành công nghiệp ô tô, hệ thống này có độ ổn định cao các hệ thanh liên kết với nhau tạo nên độ vững chắc tương đối cho bánh xe và tối đa độ bám đường của bánh xe với mặt đường. Nhưng hệ thống này đem đến sự phức tạp khi tích hợp với mô hình robot tự hành do kích thước của toàn bộ hệ thống tay đòn cũng như hệ thống giảm chấn cũng như số lượng các bộ phận sẽ làm tăng độ phức tạp cũng như chi phí để thiết kế và lắp ráp.

Ngoài ra hệ thống giảm xóc loại này có giới hạn về hình học. Điểm tiếp xúc của bánh xe với mặt đường không nằm trên trực quay của hệ thống lái. Điều này gây ảnh hưởng đến khả năng lái do các lực tác động vào bánh xe gây nên những ngoại lực tác dụng lên động cơ lái cũng như những sai lệch không mong muốn trong quá trình robot di chuyển.



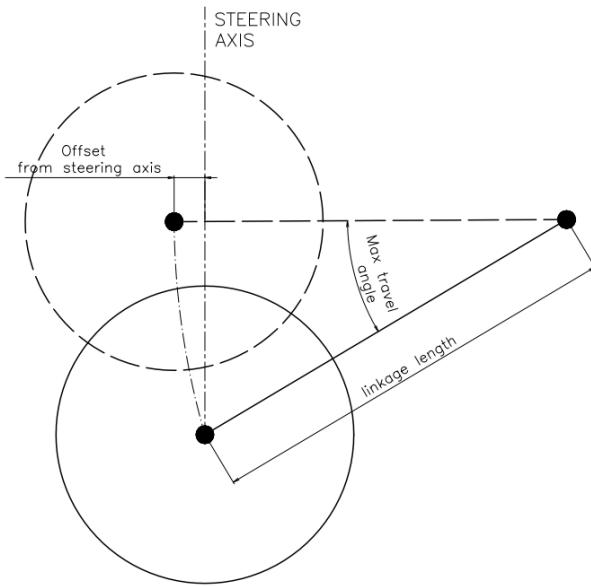
Hình 2.8: Hình học của hệ thống giảm xóc

Tiếp đến là hệ thống giảm xóc của xe Vespa, nguyên lý của hệ thống giảm xóc này là bánh xe được gắn trên đầu của một thanh đòn, độ cao của bánh xe được thay đổi với chuyển động hình vòng cung của tay đòn.



Hình 2.9: Hệ thống giảm xóc của xe máy Vespa

So với hệ thống giảm xóc dạng tay đòn đôi, hệ giảm xóc tay đòn đơn cần ít bô phận hơn. Hơn nữa phần trực lái của bánh xe cũng được thiết kế giao với điểm tiếp xúc của xe với mặt đường. Nhưng với việc chỉ có 1 tay đòn thì chuyển động của trực bánh xe sẽ có dạng cung tròn và khí biên độ của giảm xóc thay đổi trực bánh xe cũng sẽ xảy ra hiện tượng lệch khỏi trực lái. Sai lệch này tương tự với hệ giảm xóc thanh đòn đôi. Sai lệch này tăng lên với độ dài của thanh đòn và hành trình của giảm xóc.



Hình 2.10: Mô tả nguyên lý hoạt động của giảm xóc xe Vespa

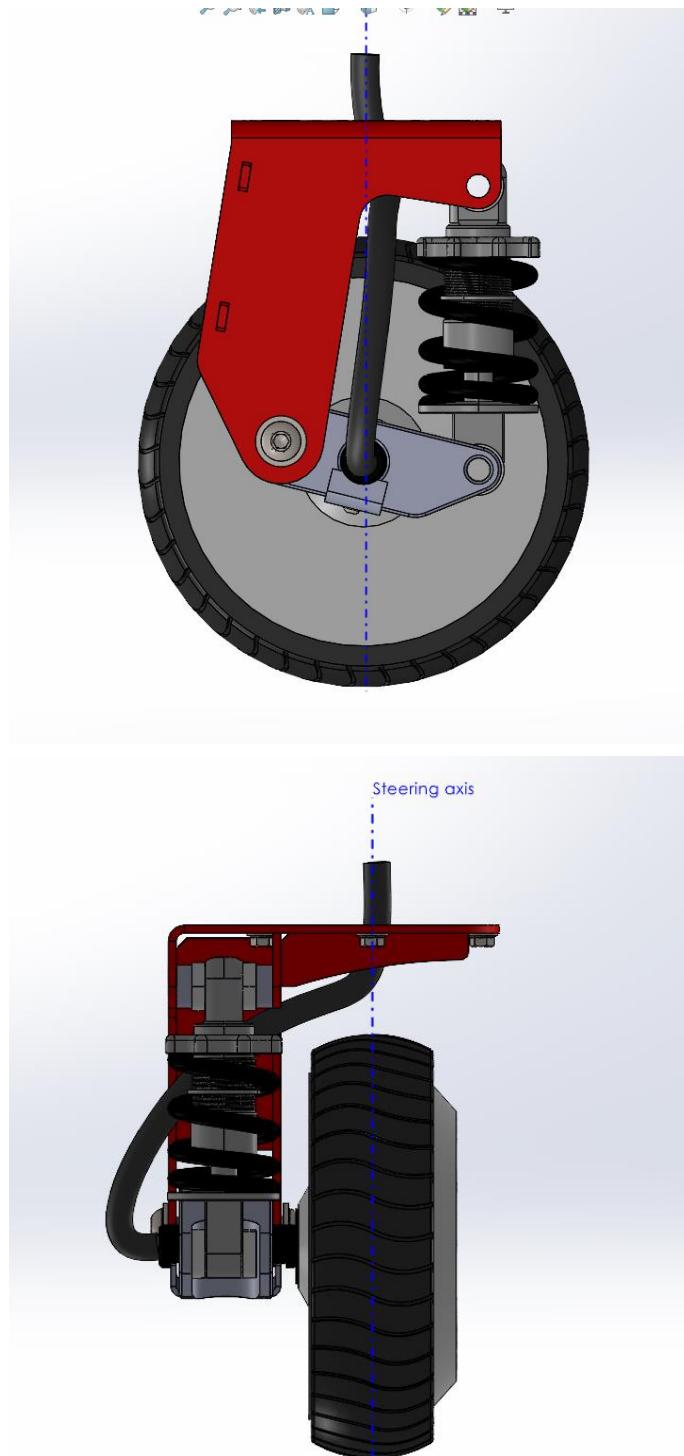
Một thiết kế mà nhóm đã tham khảo được có sử dụng nguyên lý tương tự với hệ giảm xóc của xe vespa đó là mô hình xe MRV-Modular Robotic Vehicle của NASA.



Hình 2.11: Hệ thống giảm xóc được sử dụng trong mô hình của MRV

Với những thiết kế đã tham khảo và đánh giá, nhóm đã quyết định xe thiết kế hệ thống giảm xóc của nhóm dựa trên hai mẫu thiết kế dạng tay đòn đơn. Với lý do là các hệ thống này có sự đơn giản phù hợp với những phương pháp gia công mà nhóm có thể thực hiện được.

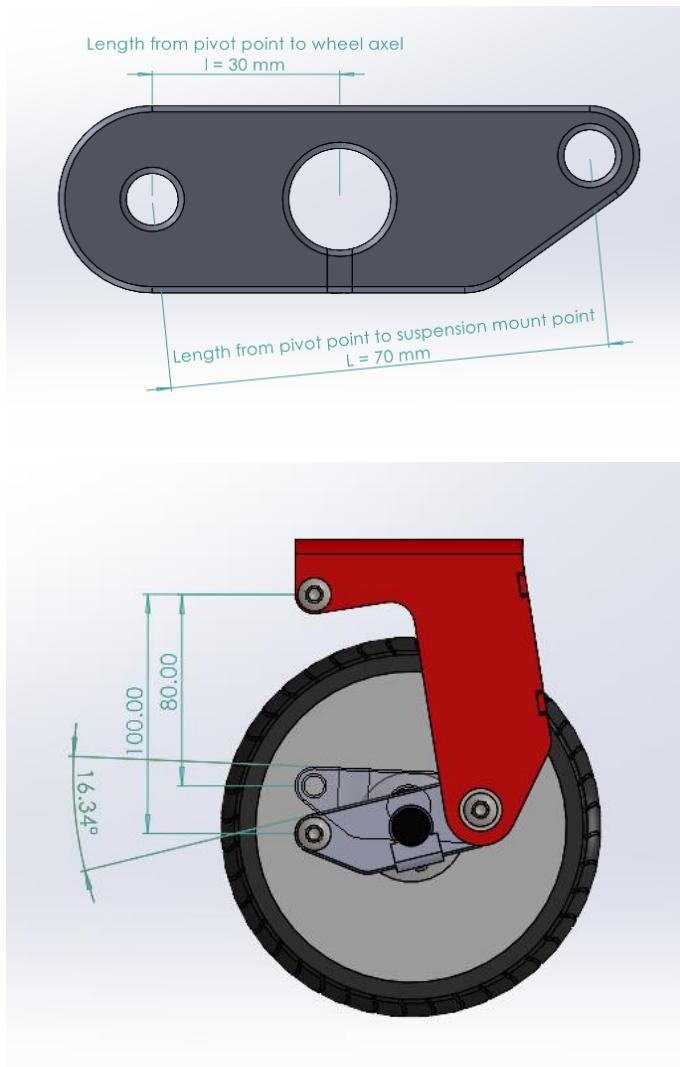
Mô hình của nhóm được thiết kế như sau:



Hình 2.12: Thiết kế hệ thống giảm xóc của nhóm thiết kế

b. Tính chọn thông số của giảm xóc

Với mô hình mà nhóm đã lựa chọn, nhóm thiết kế hệ thống giảm xóc với các thông số như sau:



Hình 2.13: Thông số hình học của tay đòn hệ thống giảm xóc

Với

- Khoảng cách từ tâm quay đến tâm bánh xe: $l = 30 \text{ mm}$
- Khoảng cách từ tâm quay đến vị trí gắn giảm xóc: $L = 70 \text{ mm}$
- Hành trình tối đa của giảm xóc: $\Delta l = 20\text{mm}$
- Góc quay tối đa của tay đòn: $\Theta_{\max} = 16.34^\circ$

Từ các thông số trên ta tính được:

- Độ lệch của trục bánh xe với trục đánh lái khi trọng tải đạt tối đa :

$$R_{\text{offset}} = l - l \cdot \cos(\Theta) = 1.21 \text{ mm} \quad (2.1)$$

Tính chọn thông số cho lò xo giảm xóc:

- Khối lượng tối đa của xe khi có tải: $M_{\max} = 30 \text{ kg}$

- Hành trình còn lại của lò xo khi có tải: $L_c = 10 mm$
- Khối lượng của 4 hệ thống swerve drive: $M_{wheel} = 16 kg$
- Khối lượng mà 1 lò xo phải chịu : $m = \frac{M_{max} - M_{wheel}}{4} = 3.5 kg$
- Với thông số hình học của tay đòn thì lực tác dụng lên lò xo:
- $F_{td} = \frac{m \cdot g}{i} = 80,12 N$ (với i là hệ số đòn bẩy của tay đòn Hình 2.13)
- Hệ số của lò xo $K = \frac{F_{td}}{L_c} = 8,01 N/mm$

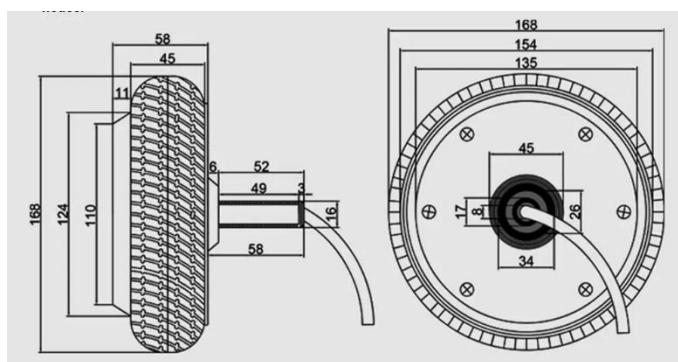
Với các thông số trên và dựa vào các kích thước thực tế nhóm chọn được lò xo có thông số như sau:

- Đường kính của mặt cắt lò xo: $wd = 4 mm$
- Đường kính vòng ngoài của lò xo: $OD = 30 mm$
- Chiều dài khi không tải: $L = 100 mm$
- Số vòng lò xo $n = 6$

2.3.4. Thông số động cơ bánh xe chủ động

Động cơ nhóm lựa chọn sử dụng là loại động cơ không chổi than BLDC liền bánh xe với các thông số :

- Đường kính bánh xe: 6.5 inch
- Tải trọng tối đa: 108 kg
- Điện áp: 36 VDC
- Điện áp cảm biến hall: 5 VDC
- Công suất: 350 W
- Tốc độ không tải: 600 rpm
- Trọng lượng: 3.1 kg



Hình 2.14: Động cơ BLDC

Với ứng dụng ban đầu là trong xe cân bằng hoverboard thì động cơ này hoàn toàn phù hợp với ứng dụng của nhóm đồ án, Hơn nữa việc động cơ đã gồm cả bánh xe nên việc tích hợp với mô hình của nhóm cũng trở nên dễ dàng hơn.

2.3.5. Thiết kế hệ thống chuyển hướng cho module swerve drive

a. Tính chọn động cơ

Do đặc tính của hệ giảm xóc đã được nói đến ở phần trên (Hình 2.13). Hệ thống tải sẽ phải chịu thêm tác động của momen do lực ma sát ở bánh xe. Momen này lớn nhất khi giảm xóc ở cuối hành trình.

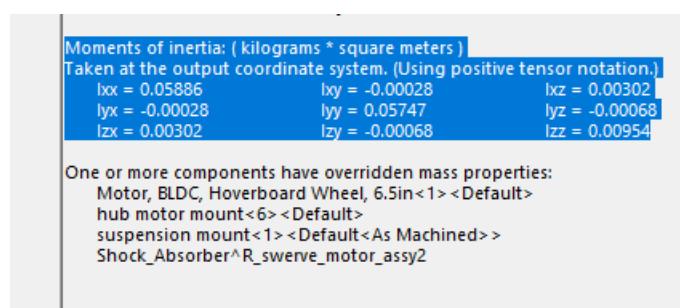
Với các thông số hình học của hệ thống giảm xóc và thông số về tải trọng đã có, nhóm tính toán các thông số liên quan đến hệ thống chuyển hướng.

Với:

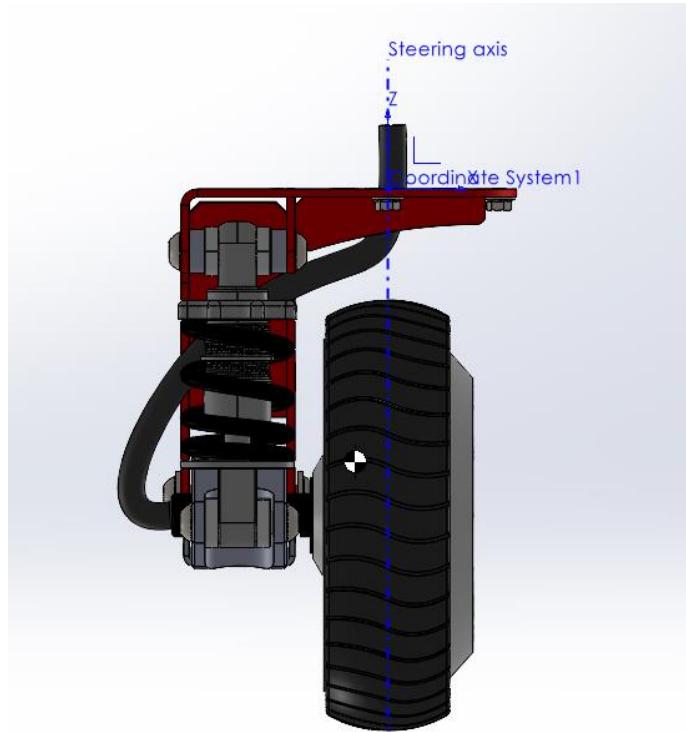
- Hệ số ma sát giữa bánh xe với mặt đường nhựa $\mu = 0.72$
- Khối lượng dồn lên 1 bánh xe $m = 7.5 kg$
- Mô men tác động lên trực lái do sai lệch hình học:

$$T_{offset} = \mu \cdot m \cdot g \cdot R_{offset} = 106.83 Nmm = 0.10683 Nm \quad (2.2)$$

- Với tốc độ lớn nhất mà hệ thống đạt được:
- $\omega_{max} = 90deg/s = 1.57rad/s$
- Chu kỳ của hệ thống lái: $t = 1s$
- Gia tốc góc của hệ thống $\beta = \frac{\Delta\theta}{\Delta t} = 1.57 rad/s^2$
- Momen quán tính tại trực quay của hệ thống lái: $I_{zz} = 0.00954 kg.m^2$



Hình 2.15: Kết quả momen quán tính từ Solidworks



Hình 2.16: Vị trí hệ tọa độ và trục quay dùng để tính momen quán tính

- Momen quán tính đối với trục lái:

$$T_{zz} = I_{zz} \cdot \beta = 0.015Mm \quad (2.3)$$

- Momen do ma sát:

$$T_{friction} = \mu \cdot N \cdot a \quad (2.4)$$

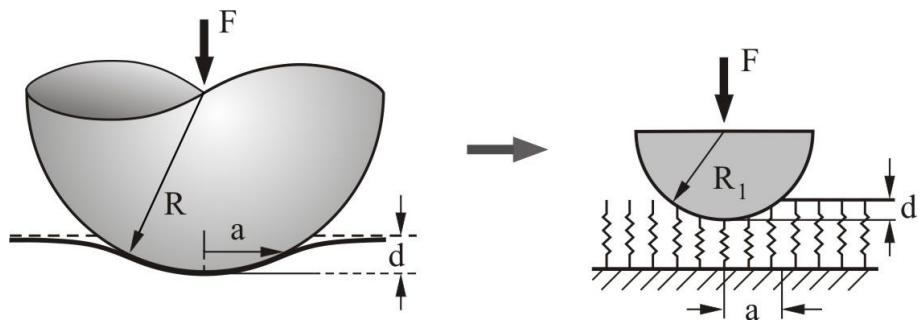
Với:

+ μ : Hệ số ma sát giữa bánh xe và đường nhựa

+ N : Phản lực của mặt đường tác dụng lên bánh xe

Dưới tác dụng của các lực đặt lên trục bánh xe, phần lốp xe bằng cao su bị biến dạng. Nhóm đã coi phần tiếp xúc giữa bánh xe giữa mặt đường có dạng hình tròn có bán kính là a .

Nhóm tham khảo cách tính bán kính a theo tài liệu [3]



Hình 2.17: Mô tả tiếp xúc giữa 2 vật thể theo tài liệu

Với:

- + $F = 73,60 \text{ N}$ - Lực tác dụng lên bánh xe
- + $E_1 = 10 \text{ MPa}$ - Úng suất Young của lốp cao su
- + $\nu_1 = 0.49$ – Hệ số poisson của lốp cao su
- + $E_2 = 17 \text{ GPa}$ – Úng suất Young của đường nhựa
- + $\nu_2 = 0.15$ – Hệ số poisson của đường nhựa
- + $\mu_d = 0.63$ – Hệ số ma sát động
- + $\mu_s = 0.72$ – Hệ số ma sát nghỉ
- + $R = 39.4 \text{ mm}$ – Bán kính của bờ mặt bánh xe tiếp xúc với mặt đường
- + d – Độ biến dạng của bờ mặt khi có lực tác dụng
- Độ cứng Young tương đương tiếp xúc Hertzian [4]

$$E' = \left(\frac{1 - \nu_1^2}{E_1} + \frac{1 - \nu_2^2}{E_2} \right)^{-1} = 13,15 \cdot 10^6 \text{ N/m}^2 \quad (2.5)$$

- Lực tác dụng [4]

$$F = \frac{4}{3} E' \cdot \sqrt{R \cdot d^3} \quad (2.6)$$

Từ (2.6), ta tính được độ biến dạng $d = 0,76 \cdot 10^{-3} \text{ m}$

- Bán kính diện tích tiếp xúc [4]

$$a = \sqrt{R \cdot d} = 0.005 \text{ m} \quad (2.7)$$

- Momen do ma sát [3] từ (2.4) trở thành:

$$T_{friction} = \int_0^a \mu_s \cdot F da = \mu_s \cdot F \cdot a = 0.26 \text{ Nm} \quad (2.8)$$

- Tổng momen ngoại lực tác dụng lên trực lái:

$$T_{steer} = \sum M = T_{offset} + T_{zz} + T_{friction} = 0.38 \text{ Nm} \quad (2.9)$$

- Công suất trên trục lái:

$$P = T \cdot \omega = 0,38 \cdot 1,57 = 0.6 W \quad (2.10)$$

Từ các thông số trên nhóm đã chọn động cơ **JGB37-545-208rpm-1:30**



Hình 2.18: Động cơ JBG37-545

Động cơ đã chọn có thông số như sau

- Công suất: 4W
- Momen khi có tải: $2.4 \text{ kg.cm} = 0.23 \text{ Nm}$
- Tốc độ động cơ khi có tải: $176 \text{ rpm} = 18.43 \text{ rad/s}$
- Số xung encoder: 480

b. Tính chọn hệ truyền động

Cách tính dưới đây được nhóm tham khảo dựa theo tài liệu của **MISUMI** –

1st Edit_PulleyCatalog.

Với các thông số:

- Tốc độ ở đầu động cơ: $n = 176 \text{ rpm}$
- Công suất động cơ $P_t = 4 \text{ W}$
- Hệ số an toàn: $K_s = K_o + K_r$
- Tỷ số truyền: 1:5

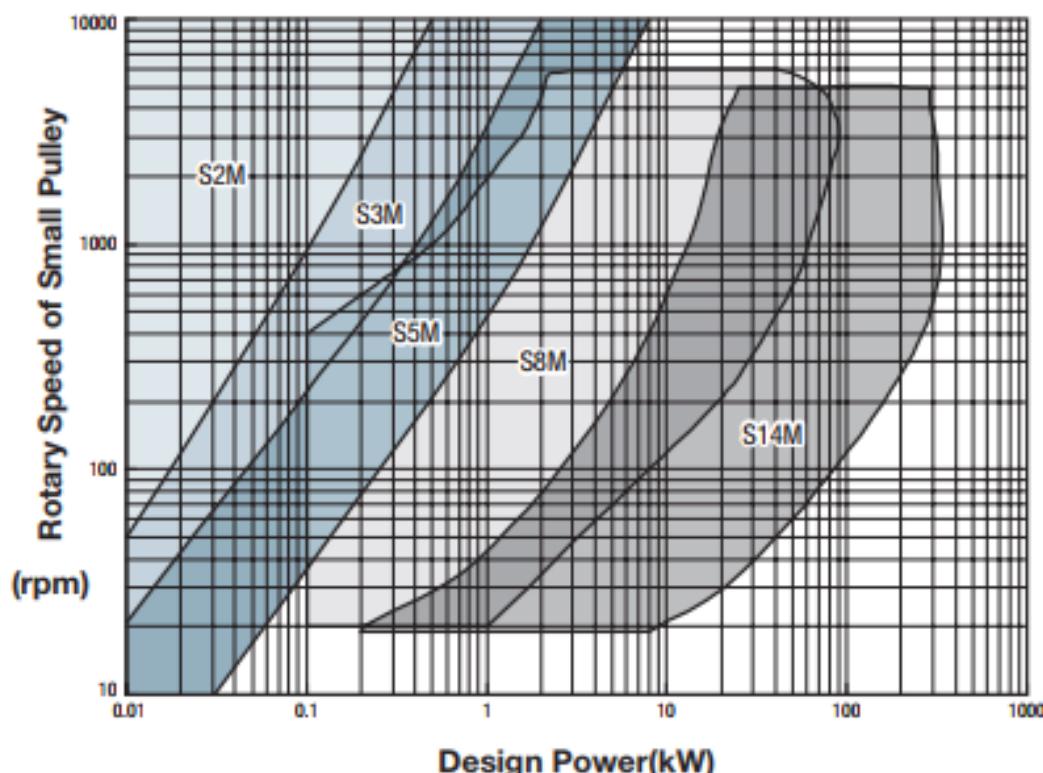
Công suất bộ truyền:

$$P_d = P_t \cdot K_s = 10.4 \text{ W} \quad (2.11)$$

Với:

- P_d : Công suất đầu ra
- P_t : Công suất động cơ
- K_o : Hệ số tải, chọn $K_o = 2.2$ (theo bảng 1 trang 124 [5])
- K_r : Hệ số giảm tốc, chọn $K_r = 0.4$ (theo bảng 2 trang 125 [5])

Table 19. Selection Guide Table 2 (S_M Series)



Bảng 2.1: Bảng chọn cõi đai theo tốc độ và công suất

Dựa vào bảng 19 trang 128 [5], ta chọn được cõi đai của bộ truyền:

- Với vận tốc ở đầu động cơ và công suất bộ truyền thì S2M sẽ là cõi đai phù hợp.

Kích thước của đai S2M:

- Module đai $m = 2$
- Bước đai $p = 2 \text{ mm}$

Xác định kích thước pulley chủ động và pulley bị động:

- Với pulley nhỏ ta chọn theo tài liệu, số răng pulley nhỏ: $T_1 = 20$

Table 25. Min. Number of Teeth of Pulley

Rotary Speed of Small Pulley (rpm)	Type of Belt, Minimum Number of Teeth							
	MXL	XL	L	H	S2M	S3M	S5M	S8M
900 or Less	12	10	12	14	14	14	14	22
Over 900	1200 or Less	12	10	12	16	14	14	24
Over 1200	1800 or Less	14	11	14	18	16	16	26
Over 1800	3600 or Less	16	12	16	20	18	18	28
Over 3600	4800 or Less	—	16	20	24	20	20	30
Over 4800	10000 or Less	—	—	—	—	20	20	—

Bảng 2.2: Bảng kích thước giới hạn của pulley nhỏ

- Với tỷ số truyền là 1:5 thì số răng pulley lớn: $T_2 = 100$

Tính sơ bộ chu vi của đai:

$$L'_p = 2C' + \frac{\pi(Dp + dp)}{2} + \frac{(Dp - dp)^2}{4C'} = 275,3 \text{ mm} \quad (2.12)$$

Chọn đai S2M chu vi 280 mm $\Rightarrow L_p = 280 \text{ mm}$

Với:

- L'_p : Chu vi đai dự tính
- C' : Khoảng cách trực tiếp dự tính, $C' = 76$
- Dp : Đường kính vòng lăn của pulley lớn, $Dp = 60.9 \text{ mm}$
- dp : Đường kính vòng lăn của pulley nhỏ $dp = 12.73 \text{ mm}$

Tính chuẩn khoảng cách trực C:

$$C = \frac{b + \sqrt{b^2 - 8(Dp - dp)^2}}{8} = 78,47 \text{ mm} \quad (2.13)$$

$$b = 2L_p - \pi(Dp + dp) = 328.7 \text{ mm} \quad (2.14)$$

Tính sơ bộ độ rộng của đai:

$$Bw' = \frac{Pd}{P_s \cdot Km} \cdot Wp = 2,2 \text{ mm} \quad (2.15)$$

\Rightarrow Chọn đai có độ rộng $Bw' = 6 \text{ mm}$

Với

- Ps: Công suất tiêu chuẩn của cõi đai, theo bảng 36, trang 134 của tài liệu [5]

$$\Rightarrow Ps = 19W$$

- Km: Hệ số ăn khớp răng (Hệ số Km liên quan đến số răng ăn khớp trên pulley nhỏ Zm)

$$Zm = \frac{Zd \cdot \theta}{360} = 8; \quad (2.16)$$

$$\theta = 180 - \frac{57,3 \cdot (Dp - dp)}{C} = 144,82 \quad (2.17)$$

- Zd: Số răng pulley nhỏ
- θ : Góc ăn khớp
- Với số răng ăn khớp là 8, theo bảng 26, trang 130 của tài liệu [5]
 $\Rightarrow Km = 1,0$
- Wp: Hệ số độ rộng đai, theo bảng 27, trang 130 của tài liệu [5]
 $\Rightarrow Wp = 4$
- Kb: Hệ số chỉnh độ rộng đai = 2.84

Xác định lực căng đai của bộ truyền:

Khác với các truyền động đai dẹt, đai thang và đai nhiều chẽm cần phải mắc đai lên bánh đai với lực căng ban đầu đủ lớn để tạo ra lực ma sát cần thiết, trong truyền động răng, lực căng ban đầu chỉ nhằm khắc phục khe hở khi ăn khớp và đảm bảo cho đai tiếp xúc tốt với bánh đai. Nó chỉ cần lớn hơn lực căng do li tâm sinh ra:

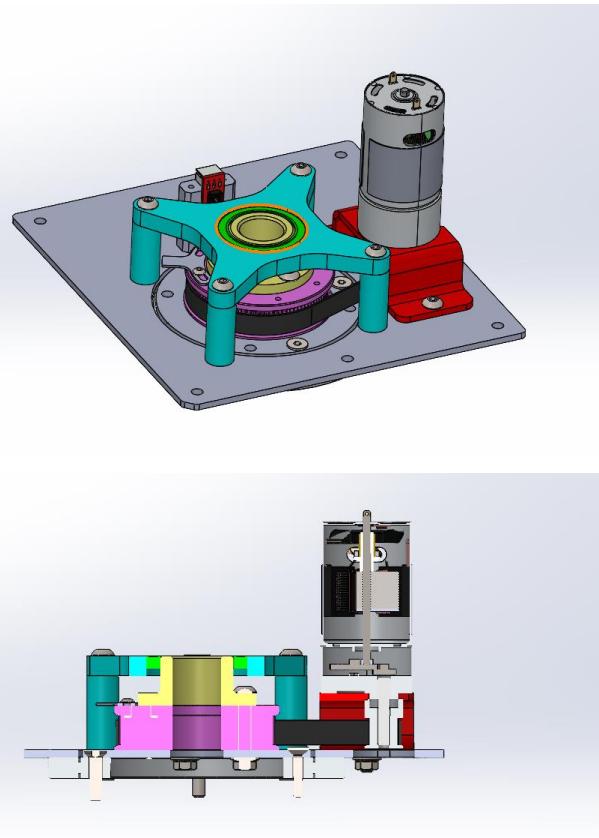
$$F_0 = 1,3 F_v = 1,3 \cdot q_m \cdot v^2 = 62,55 N \quad (2.18)$$

Với

- v - tốc độ trên đai (m/s), $v = n_1 \cdot \frac{d_{a1}}{2 \cdot 10^{-3}} = 0,12 m/s.$
- d_{a1} - đường kính của pulley nhỏ, $d_{a1} = 12,73 mm$
- n_1 - tốc độ quay của pulley nhỏ $n_1 = 18,43 rad/s$
- q_m - khối lượng 1 mét đai có chiều rộng 1 mm, $q_m = 0,0032 kg/mm$

c. Thiết kế hệ thống chuyển hướng

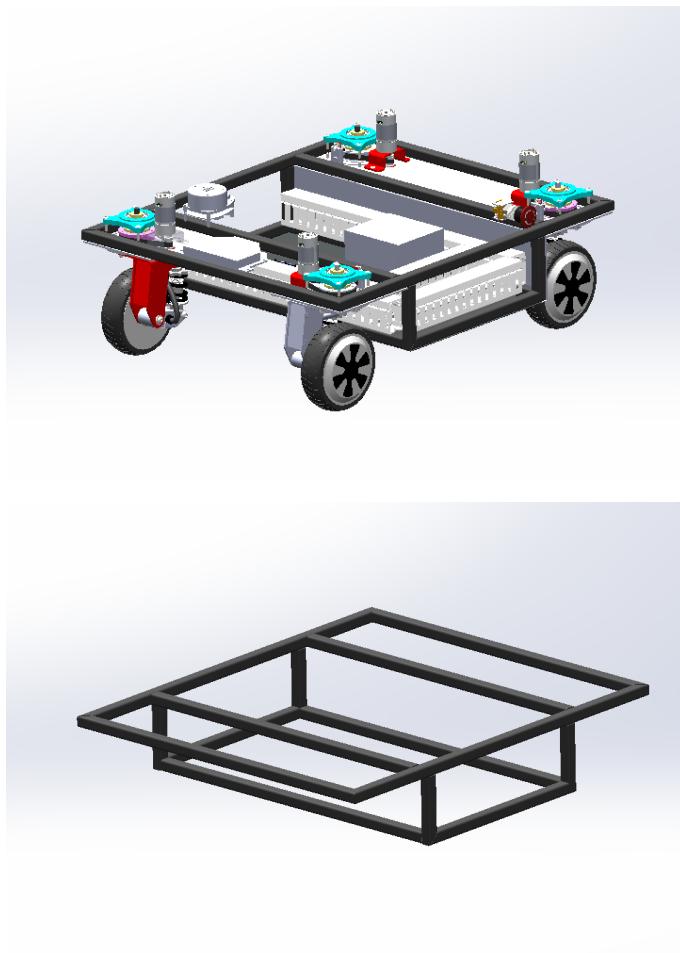
Hệ thống chuyển hướng được nhóm tính toán và thiết kế như mô tả dưới đây



Hình 2.19: Mô hình hệ thống chuyển hướng

2.3.6. Thiết kế hệ thống khung và kiểm tra

Hệ thống khung xe là thành phần giữ các bộ phận khác của xe lại, đem lại sự vững chắc và chính xác cho toàn bộ hệ thống. Khung xe cần đạt yêu cầu về độ cứng vững và khả năng chịu tải phù hợp với những yêu cầu mà đè tài đặt ra.

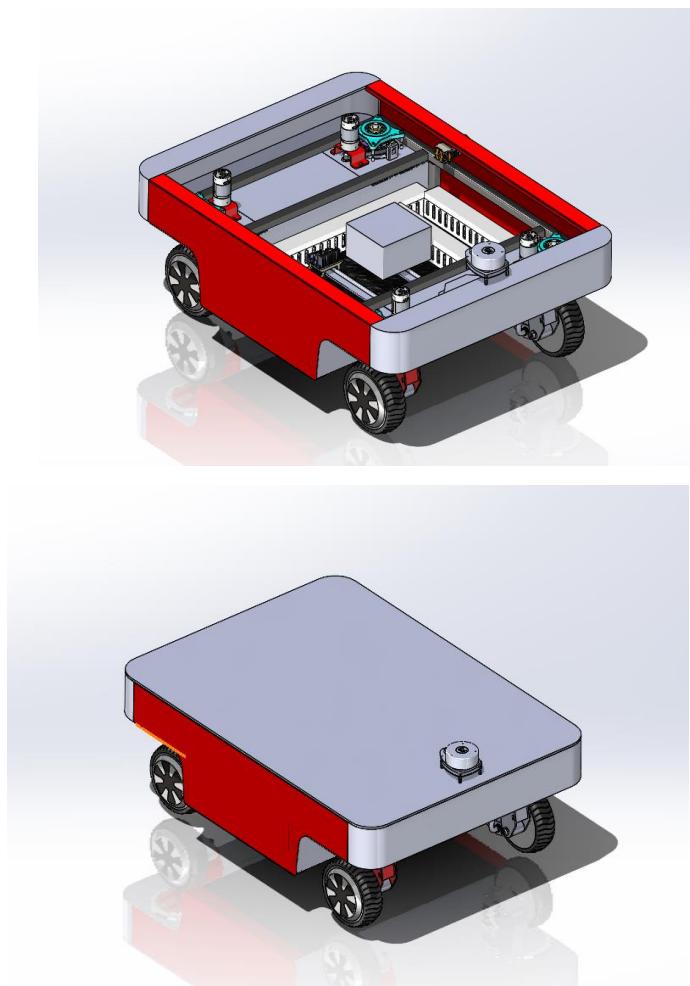


Hình 2.20: Hệ thống khung xe

Nhóm thiết kế hệ thống khung xe phù hợp với kết cấu của hệ thống swerve drive mà nhóm đã thiết kế trước đó. Vật liệu được lựa chọn là ống thép 20x20x0.8 mm.

2.3.7. Thiết kế tổng thể của robot

Với những tính toán và thiết kế của từng thành phần phần cứng cơ khí của robot nhóm tiến hành lắp ráp các bộ phận thành mô hình robot tổng thể.



Hình 2.21: Mô hình tổng thể của robot



Hình 2.22: Mô hình thực tế của robot

Robot sau khi lắp đặt các bộ phận sẽ có thông số như sau:

- Kích thước tổng thể (Dài x rộng x cao): 850 x 650 x 380 mm
- Khối lượng: 25 kg

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN

3.1. Giới thiệu chung

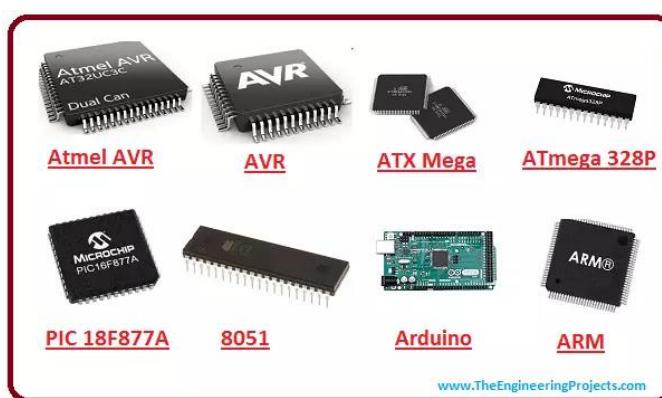
Việc điều khiển các thiết bị phần cứng của hệ thống được đảm nhiệm bởi các vi điều khiển. Chương này trình bày các thành phần của hệ thống điều khiển của robot

3.2. Các thành phần của hệ thống điều khiển

3.2.1. Vi điều khiển

Một vi điều khiển (Microcontroller - MCU) là một máy tính nhỏ trên một mạch tích hợp duy nhất được thiết kế để điều khiển các nhiệm vụ cụ thể trong các hệ thống điện tử. Nó kết hợp các chức năng của một đơn vị xử lý trung tâm (CPU), bộ nhớ và các giao diện nhập/xuất, tất cả trên một chip duy nhất.

Vi điều khiển được sử dụng rộng rãi trong các hệ thống nhúng, như các thiết bị gia dụng, hệ thống ô tô, thiết bị y tế và hệ thống điều khiển công nghiệp. Chúng cũng được sử dụng trong các sản phẩm điện tử tiêu dùng, chẳng hạn như điện thoại di động, máy tính bảng và đồng hồ thông minh. Vi điều khiển là những máy tính nhỏ trên một mạch tích hợp duy nhất (IC) được thiết kế để điều khiển các thiết bị cụ thể hoặc thực hiện các nhiệm vụ cụ thể. Chúng trở thành một phần không thể thiếu trong điện tử hiện đại nhờ giá thành thấp, tiêu thụ năng lượng thấp và tính linh hoạt.

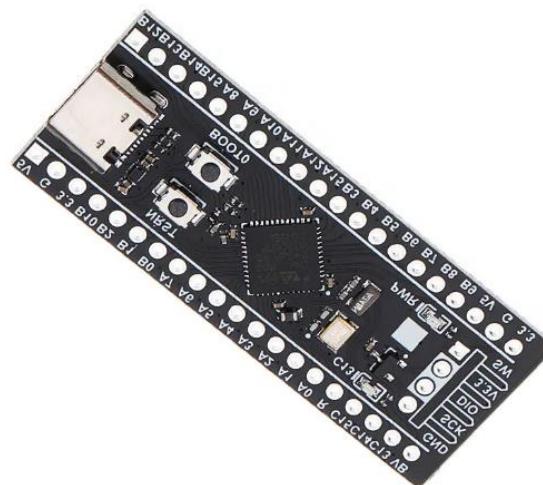


Hình 3.1: Những loại vi điều khiển phổ biến

Một trong những lợi ích chính của vi điều khiển là khả năng hoạt động trong thời gian thực. Điều này có nghĩa là chúng có thể phản ứng với các tín hiệu đầu vào và thực hiện nhiệm vụ ngay lập tức mà không cần sự can thiệp của con người.

Chúng cũng có thể được lập trình để hoạt động độc lập, giải phóng nguồn lực con người cho các nhiệm vụ khác.

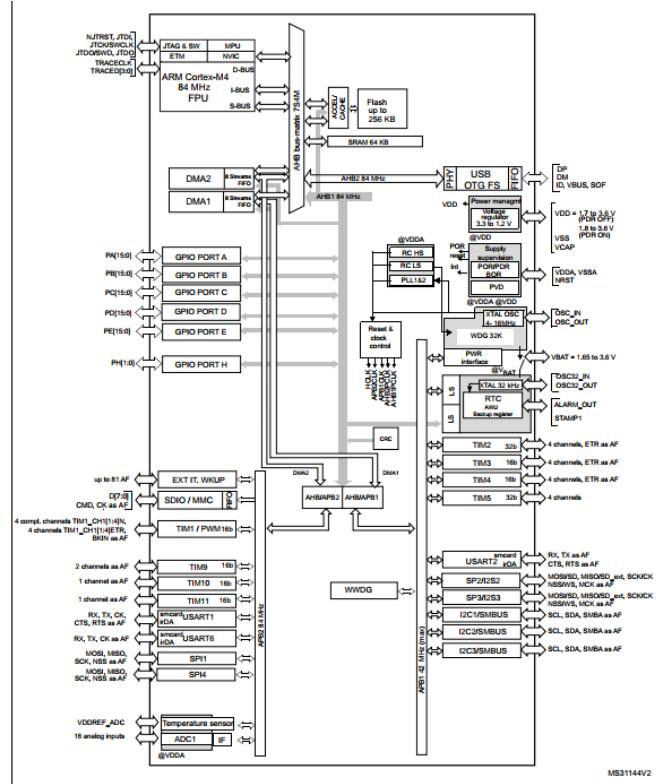
Trong đề tài này, nhóm lựa chọn vi điều khiển STM32. STM32 là một trong số những vi điều khiển được sinh viên quen thuộc và được hỗ trợ bởi nhiều nhóm và cộng đồng khác nhau. Đó là lý do tại sao nhóm của nhóm đồ án đã chọn nó cho dự án của mình.



Hình 3.2: STM32F401CCU

Cấu hình của STM32f401CCU:

- Bộ nhớ:
 - + Flash memory lên đến 256 Kbytes.
 - + Bộ nhớ OTP (One-Time Programmable) 512 bytes.
 - + SRAM lên đến 64 Kbytes.
- Giao diện ngoại vi:
 - + Lên đến 81 cổng I/O có khả năng ngắt.
 - + Lên đến 11 bộ đếm: lên đến sáu bộ đếm 16-bit, hai bộ đếm 32-bit với tần số lên đến 84 MHz, mỗi bộ đếm có tối đa 4 chế độ IC/OC/PWM hoặc bộ đếm xung và đầu vào mã xung đa chiều (quadrature) (đầu vào bộ giải mã tăng). Ngoài ra, vi điều khiển còn có hai bộ đếm thời gian xem (watchdog).
 - + Vi điều khiển có các bộ đếm thời gian (independent và window) và một bộ đếm thời gian SysTick.



Hình 3.3: Sơ đồ cấu hình của STM32F401CCU

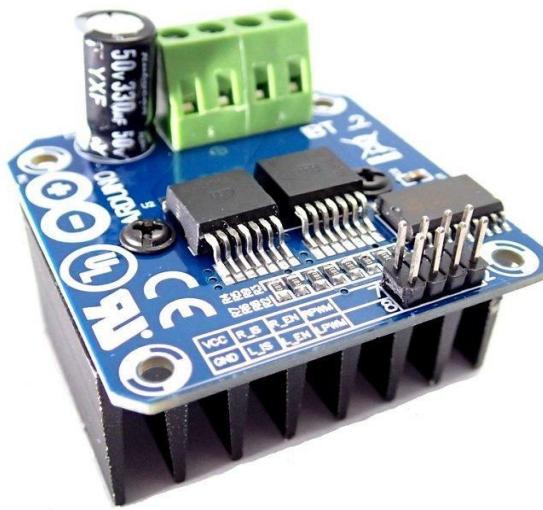
- Giao tiếp:

- + Lên đến 3 giao diện I2C (1Mbit/s, SMBus/PMBus).
- + Lên đến 3 giao diện USART (2 x 10.5 Mbit/s, 1 x 5.25 Mbit/s), giao diện ISO 7816, LIN, IrDA, modem control).
- + Lên đến 4 giao diện SPI (lên đến 42 Mbits/s khi fCPU = 84 MHz).
- + Giao diện SDIO.

STM32 là một loại vi điều khiển mà sinh viên đã quen thuộc. Sau khi xem xét khả năng của GPIO, hơn 8 bộ đếm thời gian và 3 giao diện UART, nhóm đồ án đã quyết định chọn vi điều khiển này cho dự án của mình.

3.2.2. Driver điều khiển động cơ DC

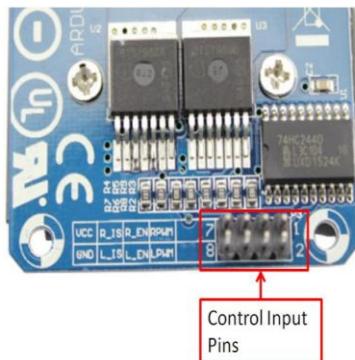
Với những yêu cầu về kỹ thuật, nhóm đã lựa chọn driver cho động cơ DC có mã BTS7960. Driver đã chọn dễ dàng giao tiếp với vi điều khiển thông qua trình điều khiển tích hợp sẵn trong IC, đáp ứng đầy đủ các tính năng cảm biến dòng điện, tạo thời gian chết, chống quá nhiệt, quá áp, quá dòng, tụt áp và ngắn mạch.



Hình 3.4: Driver BTS7960

Thông số kỹ thuật:

- Nguồn điện: 6 ~ 27V.
- Dòng tải tối đa: 43A (Tải trớ) hoặc 15A (Tải từ).
- Tín hiệu logic điều khiển: 3.3 ~ 5V.
- Tần số điều khiển tối đa: 25KHz.
- Tự động tắt nguồn khi điện áp thấp: để tránh điều khiển động cơ ở điện áp thấp, thiết bị sẽ tự động tắt nguồn. Nếu điện áp < 5.5V, trình điều khiển sẽ tự động ngắt kết nối và sẽ mở lại sau khi điện áp > 5.5V.
- Bảo vệ quá nhiệt: BTS7960 có tính năng bảo vệ quá nhiệt thông qua cảm biến nhiệt tích hợp sẵn. Đầu ra sẽ bị cắt khi quá nhiệt xảy ra.
- Kích thước: 40 x 50 x 12mm.



Pin No	Function	Description
1	RPWM	Forward Level or PWM signal, Active High
2	LPWM	Reverse Level or PWM signal, Active High
3	R_EN	Forward Drive Enable Input, Active High/ Low Disable
4	L_EN	Reverse Drive Enable Input, Active High/ Low Disable
5	R_IS	Forward Drive, Side current alarm output
6	L_IS	Reverse Drive, Side current alarm output
7	Vcc	+5V Power Supply microcontroller
8	Gnd	Ground Power Supply microcontroller

Hình 3.5: Sơ đồ kết nối của driver

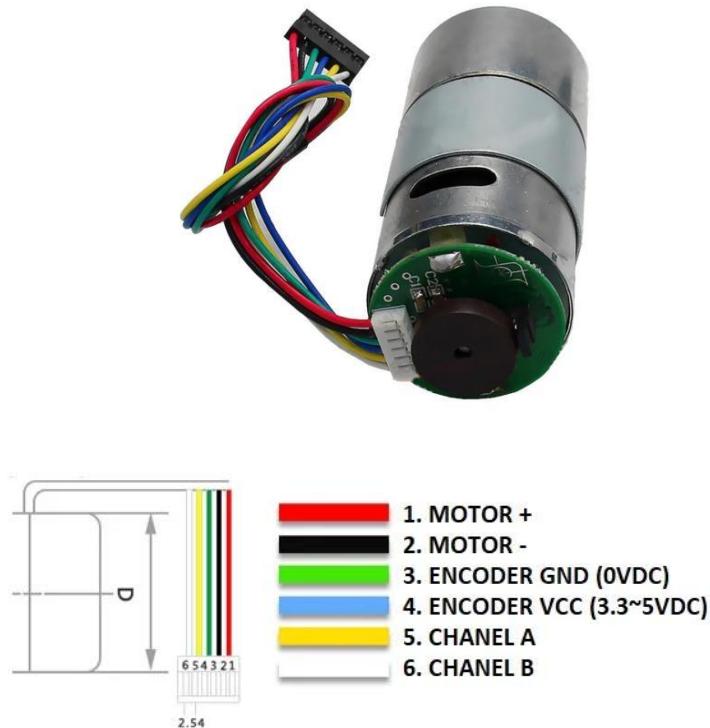
Sơ đồ chân kết nối:

- VCC: Nguồn điện cho műc logic điều khiển (5V - 3V3).
- GND : Nối đất.
- R_EN = 0: Vô hiệu hóa nửa H-bridge bên phải. R_EN = 1: Kích hoạt nửa H-bridge bên phải.
- L_EN = 0: Vô hiệu hóa nửa H-bridge bên trái. L_EN = 1: Kích hoạt nửa H-bridge bên trái.
- RPWM và LPWM: Các chân điều khiển đảo chiều và tốc độ động cơ.
 - + RPWM = 1 và LPWM = 0: Động cơ quay về phía trước (forward).
 - + RPWM = 0 và LPWM = 1: Động cơ quay ngược (reverse rotation).
 - + RPWM = 1 và LPWM = 1 hoặc RPWM = 0 và LPWM = 0: Dừng lại (stop).
 - + R_IS và L_IS: kết hợp với các resistor để giới hạn dòng đi qua nửa H-bridge.

Nguyên lý điều khiển:

Trong các ứng dụng thông thường, RPWM và LPWM được kết nối với GPIO (ví dụ: chân số 2, 3) để điều khiển hướng quay của động cơ. Chân R_EN và L_EN được kết nối với nhau và sau đó được kết nối với PWM (ví dụ: chân số 5) để điều khiển tốc độ của động cơ.

3.2.3. Động cơ encoder DC



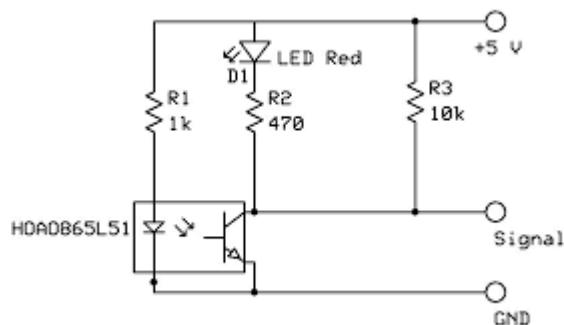
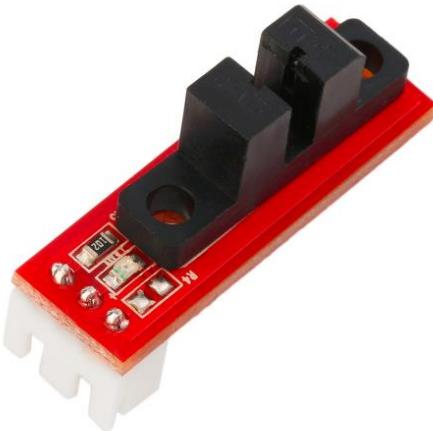
Hình 3.6: Sơ đồ kết nối của động cơ encoder

Sơ đồ chân kết nối:

- Dây đỏ, dây đen: Điện áp điều khiển của động cơ DC (12VDC).
- Dây xanh lá, dây xanh lam: Điện áp cấp cho cảm biến encoder (3.3~5VDC).
- Dây vàng, dây trắng: Tín hiệu của 2 kênh encoder A và B.

3.2.4. Công tắc hành trình

Công tắc hành trình quang học (optical end stops) bao gồm một công quang điện, khi có vật cản chặn tia sáng ở giữa công quang điện module sẽ thay đổi mức logic của tín hiệu.



Hình 3.7: Công tắc hành trình quang

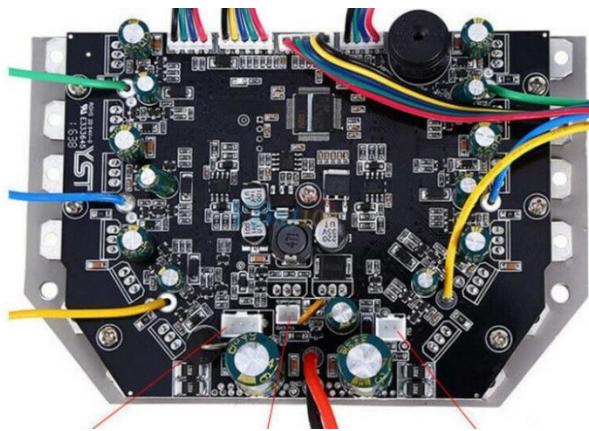
Sơ đồ chân kết nối:

- VCC, GND: Điện áp cấp cho module (5VDC).
- Signal: Tín hiệu logic của module.

3.2.5. Driver động cơ BLDC

Để điều khiển 4 động cơ của robot nhóm sử dụng driver điều khiển đi kèm với động cơ BLDC của xe cân bằng 2 bánh. Driver này cung cấp các kết nối để điều khiển 2 động cơ BLDC và 2 cảm biến HALL. Để điều khiển driver, nhóm sử dụng giao thức UART có sẵn trên driver để điều khiển tốc độ của 2 động cơ.





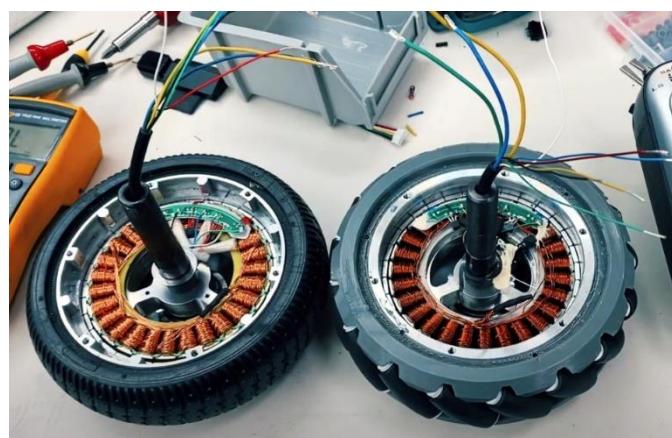
Hình 3.8: Driver BLDC

3.2.6. Động cơ BLDC

Động cơ BLDC là loại động cơ đồng bộ, khác với động cơ DC thường thấy, động cơ BLDC trong cấu tạo không có chổi than. Điều này khiến cho động cơ BLDC có cấu tạo đơn giản và hoạt động hiệu quả.

Động cơ BLDC thường được điều khiển với driver và có phản hồi để điều khiển vòng kín. Việc phản hồi của động cơ có vai trò quan trọng đến hiệu suất của động cơ, vì nguyên lý của động cơ là thay đổi từ trường trong cuộn dây stator để thay đổi vị trí của rotor nên bộ điều khiển cần nắm được vị trí chính xác của rotor để từ đó điều chỉnh từ trường trong cuộn dây để đạt được yêu cầu về tốc độ.

Động cơ mà nhóm sử dụng là loại BLDC có phần stator liền với trực động cơ, bánh xe được gắn với rotor quay phía bên ngoài.

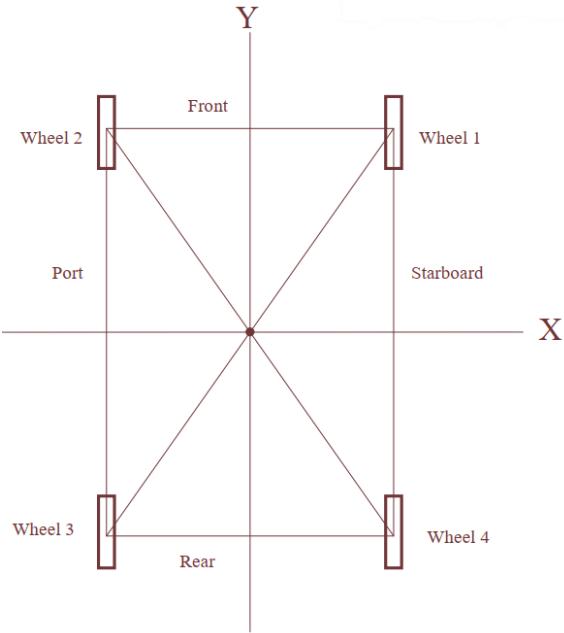


Hình 3.9: Động cơ BLDC

3.3. Lập trình điều khiển

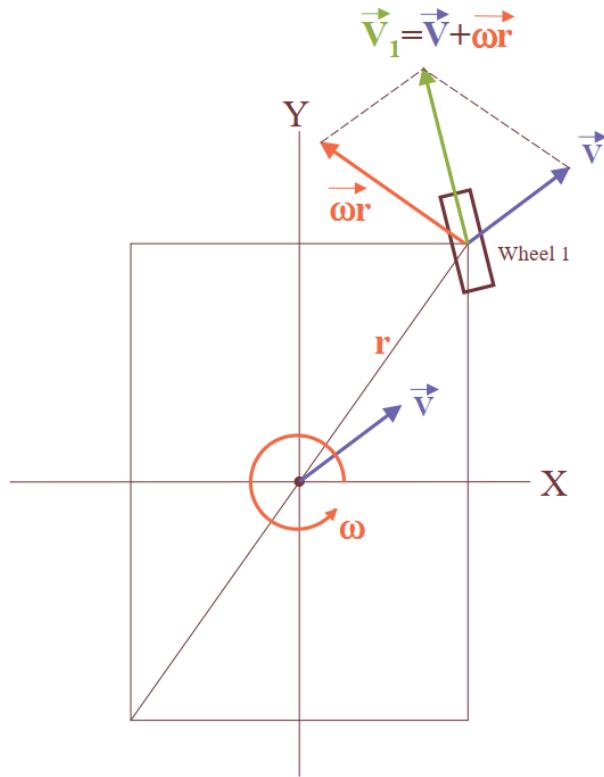
3.3.1. Động học ngược của robot swerve drive

Để thuận tiện cho quá trình giải bài toán động học của robot, nhóm quy định thứ tự của mỗi bánh xe như sau.



Hình 3.10: Vị trí các bánh xe

Với robot này, chúng ta quy các chuyển động của robot làm 2 thành phần: chuyển động tuyến tính và chuyển động quay tại tâm của robot, vận tốc tuyến tính và vận tốc góc tương ứng được ký hiệu là V và ω . Vận tốc tuyến tính của mỗi bánh xe là như nhau khi chúng làm việc cùng nhau. Đối với thành phần góc, chúng ta lấy vận tốc tiếp tuyến ($\omega \cdot r$) tại mỗi bánh xe tương ứng với ω . Vận tốc (bao gồm tốc độ và hướng) của mỗi bánh xe là tổng của \vec{V} và $\overrightarrow{\omega \cdot r}$.



Hình 3.11: Tính toán hướng và tốc độ cho mỗi module bánh xe

Ví dụ lấy cho module bánh xe số 1, chúng ta tính tốc độ cho các thành phần x và y:

$$\begin{bmatrix} v_1x \\ v_1y \end{bmatrix} = \begin{bmatrix} V_x & L/2 \\ V_y & -H/2 \end{bmatrix} \begin{bmatrix} 1 \\ \omega \end{bmatrix}$$

Với

- L là khoảng cách giữa 2 bánh xe theo phương y.
- H là khoảng cách giữa 2 bánh xe theo phương x.
- $r = \frac{\sqrt{L^2 + H^2}}{2}$
- v_1 : Vector vận tốc của bánh xe số 1.
- V: Vận tốc tuyến tính của robot.

Thực hiện theo quy trình này, chúng ta thu được tập hợp các vector vận tốc cho 4 bánh xe.

$$\begin{bmatrix} v_1x \\ v_1y \end{bmatrix} = \begin{bmatrix} V_x & L/2 \\ V_y & -H/2 \end{bmatrix} \begin{bmatrix} 1 \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} v_2x \\ v_2y \end{bmatrix} = \begin{bmatrix} V_x & L/2 \\ V_y & H/2 \end{bmatrix} \begin{bmatrix} 1 \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} v_3x \\ v_3y \end{bmatrix} = \begin{bmatrix} V_x & -L/2 \\ V_y & H/2 \end{bmatrix} \begin{bmatrix} 1 \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} v_4x \\ v_4y \end{bmatrix} = \begin{bmatrix} V_x & -L/2 \\ V_y & -H/2 \end{bmatrix} \begin{bmatrix} 1 \\ \omega \end{bmatrix}$$

Chúng ta có thể suy ra tốc độ và hướng như sau:

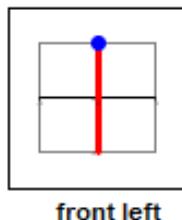
$$v_n = \sqrt{V_{nx}^2 + V_{ny}^2}$$

$$\theta_n = \text{atan2}(V_{ny}, V_{nx})$$

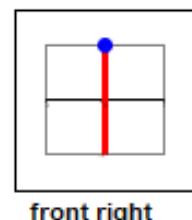
Với n= 1,2,3,4.

Mô tả các chuyển động cơ bản của robot:

- Chuyển động với vận tốc $v_x = v$

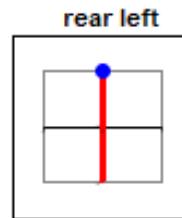


front left

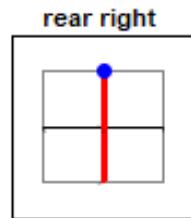


front right

Top View

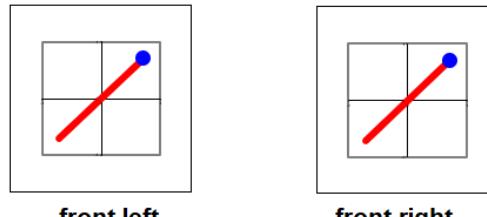


rear left

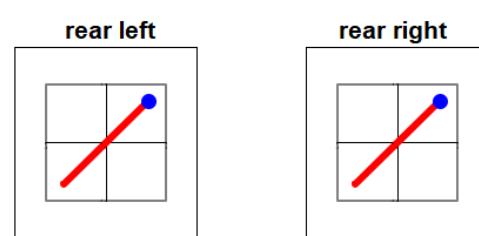


rear right

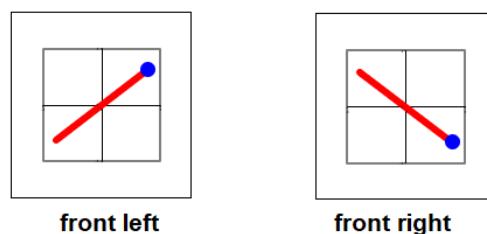
- Chuyển động của robot với vận tốc $v_x = v_y = v$



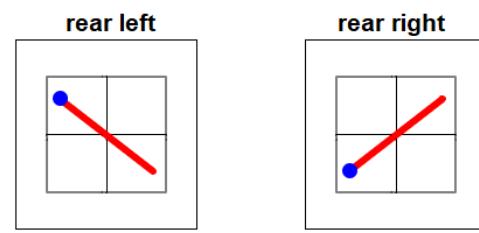
Top View



- Chuyển động của robot với vận tốc $\omega_z = \omega$



Top View



3.3.2. Động học thuận của robot swerve drive

Từ vận tốc và góc quay của từng bánh xe ta có thể quy ngược lại được về vận tốc dài và vận tốc quay tại tâm của robot:

Với vận tốc của các bánh xe v_n và góc quay của các bánh xe θ_n đã biết, ta có thể suy ngược được về vận tốc dài V và vận tốc góc ω_z tại vị trí tâm của robot

Với thành phần v_n của bánh xe được phân tách thành v_{nx} và v_{ny} . Từ động học ngược ta rút ra được phương trình sau:

$$\begin{bmatrix} v_{1x} \\ v_{1y} \\ v_{2x} \\ v_{2y} \\ v_{3x} \\ v_{3y} \\ v_{4x} \\ v_{4y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & L/2 \\ 0 & 1 & -H/2 \\ 1 & 0 & L/2 \\ 0 & 1 & H/2 \\ 1 & 0 & -L/2 \\ 0 & 1 & H/2 \\ 1 & 0 & -L/2 \\ 0 & 1 & -H/2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z/2 \end{bmatrix} \quad (3.1)$$

$$[B] = [A][X]$$

Việc giải bài toán động học ngược được quy về giải hệ phương trình dưới điều kiện (3.1). Nhóm tham khảo cách giải bằng phương pháp phân rã QR để giải phương trình trên.

Đặt $[A] = [Q][R]$.

Nhân cả 2 vế với $[A]^T$:

$$[A]^T[A][X] = [A]^T[B] \quad (3.2)$$

Với $[A] = [Q][R]$, ta được:

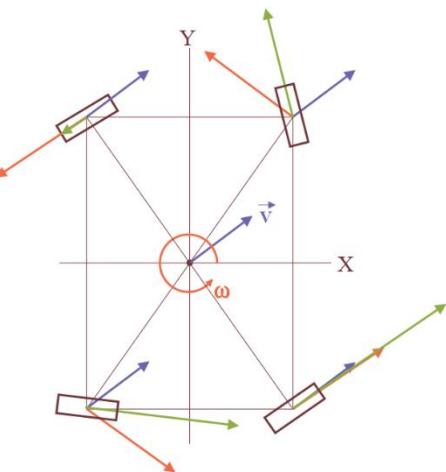
$$\begin{aligned} [Q]^T[R]^T[Q][R][X] &= [Q]^T[R]^T[B] \\ \Leftrightarrow [R][X] &= [Q]^T[B] \end{aligned} \quad (3.3)$$

Với phương trình 3.3 đã xác định, vận tốc tại tâm của robot v_x, v_y, ω_z được xác định.

3.3.3. Xác định vị trí của robot

Trong không gian hoạt động của robot, ta đặt tọa độ của robot ở thời điểm ban đầu là:

$$x_0 = 0; y_0 = 0; \theta_0 = 0$$



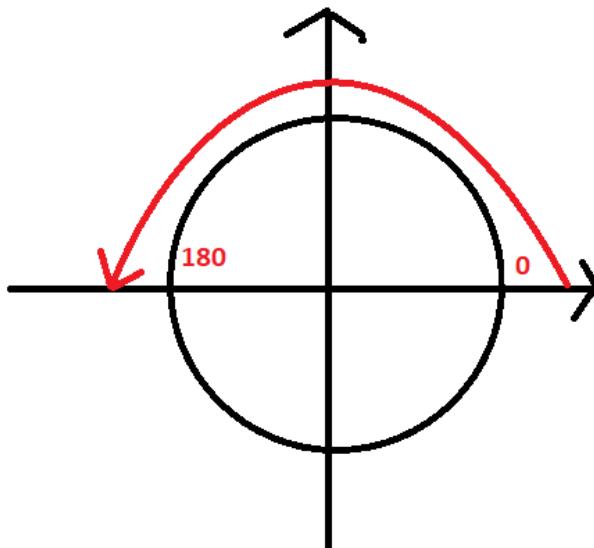
Hình 3.12: Liên hệ của vận tốc tại tâm xe với vận tốc của từng bánh xe

Tọa độ của robot tại thời điểm Δt , được xác định bởi:

$$\begin{aligned}x &= x_0 + [v_x \cos \theta - v_y \sin \theta] \cdot \Delta t \\y &= y_0 + [v_x \sin \theta + v_y \cos \theta] \cdot \Delta t \\ \theta &= \theta_0 + \omega_z \cdot \Delta t\end{aligned}\quad (3.4)$$

3.3.4. Thực hiện trên vi điều khiển (Microcontroller):

Do các giới hạn cơ học, góc (Theta) của mỗi module phải được giới hạn trong khoảng $[0, 180]$.



Hình 3.13: Giới hạn góc quay của bánh xe

Đối với các góc không nằm trong khoảng $[0, 180]$, chúng ta lấy phần dư 180 độ của góc đó và đảo ngược tốc độ như code sau:

```

int Swerve_reverseSpeedFlag[4] ;
int Swerve_angleOptimization(float inputAngle,uint8_t wheelN0)
{
    int suitableAngle;
    if((inputAngle >= -180) && (inputAngle<0))
    {
        suitableAngle = inputAngle + 180;
        Swerve_reverseSpeedFlag[wheelN0-1] = -1;
    }
    else
    {
        suitableAngle = inputAngle;
        Swerve_reverseSpeedFlag[wheelN0-1] = 1;
        //sModule1Params.reverseVel
    }
    return (int)suitableAngle;
}

```

Hình 3.14: Tối ưu hóa góc trong chương trình.

Tốc độ đầu ra được giới hạn ở mức 1000 vòng/phút.

```

// Limit
sModule1Params.speed = (sModule1Params.speed > 1000) ? 1000 : sModule1Params.speed;
sModule2Params.speed = (sModule2Params.speed > 1000) ? 1000 : sModule2Params.speed;
sModule3Params.speed = (sModule3Params.speed > 1000) ? 1000 : sModule3Params.speed;
sModule4Params.speed = (sModule4Params.speed > 1000) ? 1000 : sModule4Params.speed;

sModule1Params.speed = (sModule1Params.speed < -1000) ? -1000 : sModule1Params.speed;
sModule2Params.speed = (sModule2Params.speed < -1000) ? -1000 : sModule2Params.speed;
sModule3Params.speed = (sModule3Params.speed < -1000) ? -1000 : sModule3Params.speed;
sModule4Params.speed = (sModule4Params.speed < -1000) ? -1000 : sModule4Params.speed;

```

Hình 3.15: Giới hạn vận tốc

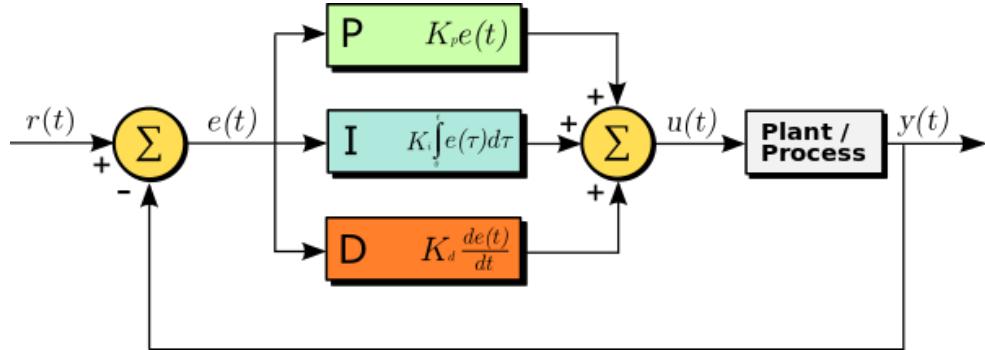
3.3.5. Điều khiển vị trí của động cơ encoder DC

Với yêu cầu điều chỉnh hướng của 4 bánh xe dẫn động, nhóm sử dụng 4 động cơ DC với encoder để điều khiển góc quay của 4 bánh xe này.

a. Bộ điều khiển PID

Với mục đích điều khiển vị trí của động cơ DC, bộ điều khiển PID là lựa chọn rất phổ biến.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt},$$



Hình 3.16: Mô hình hóa của bộ điều khiển PID

Tham số P tỷ lệ với giá trị hiện tại của sai số SP-PV $e(t)$. Ví dụ, nếu sai số lớn, đầu ra điều khiển sẽ tăng theo tỷ lệ bằng cách sử dụng hệ số tỷ lệ "K_p". Sử dụng chỉ điều khiển tỷ lệ sẽ dẫn đến sai số giữa điểm thiết lập và giá trị quá trình vì bộ điều khiển yêu cầu một sai số để tạo ra phản hồi đầu ra tỷ lệ. Trong điều kiện ổn định của quá trình, đạt được một trạng thái cân bằng, với một "offset" ổn định giữa SP-PV.

Tham số I tích lũy các giá trị quá khứ của sai số SP-PV và tích phân chúng theo thời gian để tạo ra tham số I. Ví dụ, nếu còn lại sai số SP-PV sau khi áp dụng điều khiển tỷ lệ, tham số tích phân sẽ có găng loại bỏ sai số còn lại bằng cách thêm hiệu ứng điều khiển do giá trị tích lũy lịch sử của sai số. Khi sai số được loại bỏ, tham số tích phân sẽ không còn tăng lên nữa. Điều này sẽ dẫn đến hiệu ứng tỷ lệ giảm khi sai số giảm, nhưng được bù đắp bằng hiệu ứng tích phân ngày càng lớn.

Tham số D là một ước lượng tốt nhất về xu hướng tương lai của sai số SP-PV, dựa trên tốc độ thay đổi hiện tại của nó. Đôi khi được gọi là "điều khiển tiên đoán", vì nó hiệu quả tìm cách giảm thiểu tác động của sai số SP-PV bằng cách tạo ra một ảnh hưởng điều khiển dựa trên tốc độ thay đổi của sai số. Càng nhanh chóng thay đổi, hiệu ứng điều khiển hoặc giảm dần càng lớn.

Cách điều chỉnh PID:

- Về cơ bản, chúng ta xem xét những yếu tố sau:

- + Tham số P được áp dụng để tăng tốc độ phản hồi (tốc độ phản hồi nhanh). Hành động P được đẩy quá sẽ dẫn đến dao động.
- + Tham số I được áp dụng để đạt được phản hồi ổn định mong muốn. Nhược điểm là phản hồi dao động cao hơn trong một khoảng thời gian dài.
- + Tham số D được áp dụng cho mục đích giảm rung. Nhược điểm là khả năng dao động ở tần số cao hơn và nhạy cảm với nhiễu.
- + Các tham số đầu vào cho PID trong trường hợp của chúng ta là giá trị xung hiện tại và giá trị mục tiêu (set-point) của bộ đếm encoder, hàm được triển khai như sau: PID(current_pulse, target_pulse).
- + Giá trị đầu ra trả về là giá trị PWM để điều khiển động cơ sao cho nó đạt được vị trí mục tiêu ngay cả khi động cơ bị ảnh hưởng bởi các điều kiện bên ngoài.
- Điều quan trọng là điều chỉnh các tham số Ki, Kp, Kd và DT (thời gian giữa hai lần tính toán) sao cho nó thích nghi với đầu ra mong đợi, và lưu ý rằng đầu ra nên nằm trong giới hạn tối thiểu và tối đa.
- Trong nghiên cứu này, việc áp dụng bộ điều khiển PID giúp động cơ quay đến vị trí mong đợi ngay cả khi bị ảnh hưởng bởi các điều kiện bên ngoài. Ví dụ, trục động cơ phải duy trì vị trí của nó trong khi xe đang di chuyển.

Triển khai bộ điều khiển PID trên vi điều khiển:

```
uint16_t PID(float ref, float pitch, uint8_t pid_flag)
{
    float P = 0, I = 0, D = 0, pid_pwm = 0;
    float lastError = 0;
    float error = 0;
    static float i_err = 0;
    uint16_t out_pwm = 0;
    //calculate error
    if(ref > pitch)
    {
        error = (ref - pitch);
    }
    else
    {
        error = (pitch - ref);
    }
    P = Kp * error;

    i_err+= error;
    if(pid_flag==1)
        I = Ki* i_err;
    else
        I = 0.5*i_err; // If the robot has to move the Ki

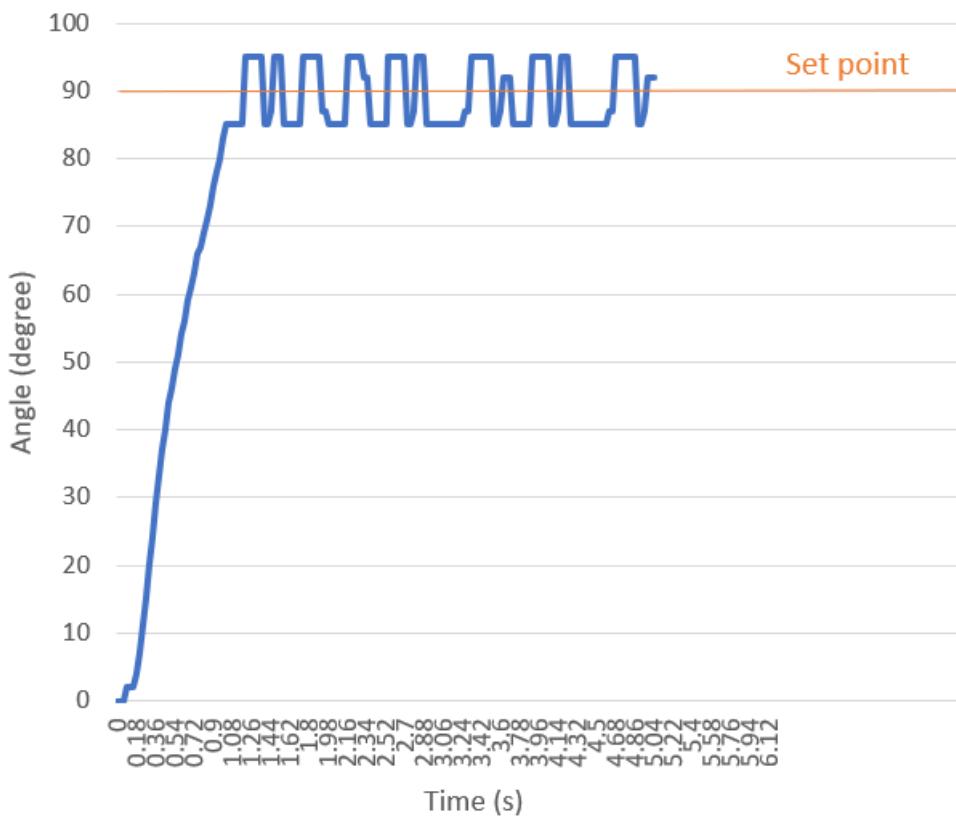
    if (I > MAX_PWM)
        I = MAX_PWM;
    else if (I<MIN_PWM)
    {
        I = MIN_PWM;
    }
    //calculate Derivative term
    D = Kd * (error - lastError);

    // If the robot has to move the control low is PI so
    if(pid_flag == 0)
    {
    {
        D = 0;
    }
    //total PID value
    pid_pwm = P + I + D;
    //max sure pwm is bound between allowed min/
    out_pwm = (int)(pid_pwm);

    if (pid_pwm > MAX_PWM)
        out_pwm = MAX_PWM;
    else if (pid_pwm < MIN_PWM)
        out_pwm = MIN_PWM;
    lastError = error;

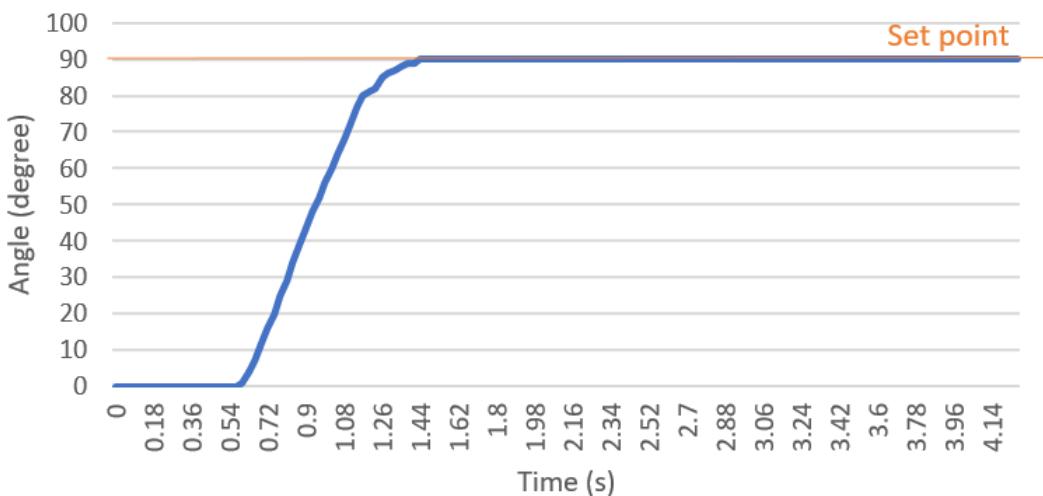
    return out_pwm;
}
```

Kết quả thông số của bộ điều khiển PID:



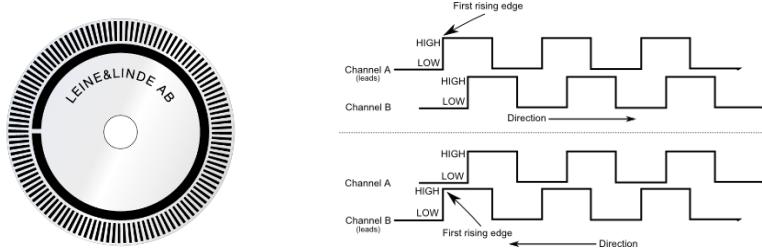
Hình 3.17: Đáp ứng của hệ thống khi không sử dụng PID

$$P = 1; I = 0.0000011; D = 10$$



Hình 3.18: Đáp ứng của hệ thống với bộ điều khiển PID thích hợp

b. Xác định vị trí của trục động cơ bằng encoder



Hình 3.19: Nguyên lý của encoder

Vị trí tương đối của động cơ có thể được xác định bằng cách đếm các bước đo (increments) từ một điểm gốc nào đó. Hướng được xác định bằng cách phát hiện xung của tín hiệu đến đầu tiên như đã được trình bày ở trên.

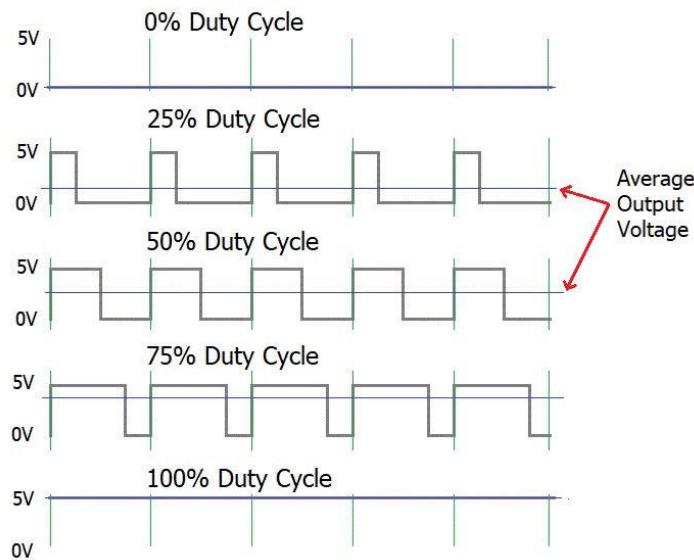
Trong trường hợp này, nhóm đồ án sử dụng chế độ input capture để liên kết với việc đọc một chân GPIO. Mức tín hiệu của GPIO giúp chúng ta xác định hướng quay của động cơ. Biến "count" được tăng lên mỗi khi động cơ quay theo chiều kim đồng hồ và giá trị được giảm trong hướng ngược lại.

```
86 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
87 {
88     /*Axis 1: channel 1, pin A8*/
89     if (htim->Instance == htim2.Instance)
90     {
91         /*Axis 1: channel 1, pin A8*/
92         axis1.dir = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_8);
93         if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
94         {
95             if (!axis1.dir)
96                 cnt1++;
97             else
98             {
99                 /*Avoiding floating value*/
100                cnt1 = (cnt1 > MIN_AXIS_VAL) ? (cnt1 - 1) : 0;
101            }
102        }
103    }
104 }
```

c. Điều khiển tốc độ của động cơ

Để điều khiển tốc độ của động cơ ta có thể thay đổi chế độ rộng xung (Pulse-width modulation - PWM) hay còn gọi là điều chế độ rộng xung thời gian, là một phương pháp điều khiển công suất trung bình của một tín hiệu điện. Giá trị trung bình của điện áp được cung cấp cho tải được điều khiển bằng cách chuyển đổi nguồn cấp giữa 0% và 100% với tốc độ nhanh hơn so với thời gian tải thay đổi đáng kể.

Trong đồ án này, nhóm áp dụng kỹ thuật PWM để điều khiển tốc độ của động cơ.



Hình 3.20: Nguyên lý điều khiển độ rộng xung

d. Xác định vị trí ban đầu của động cơ

Do đặc tính của encoder tương đối, ta chỉ có thể xác định được vị trí của động cơ từ vị trí bắt đầu, nếu như vị trí ban đầu của động cơ bị thay đổi thì vị trí sau đó cũng sẽ bị thay đổi theo. Vậy để có định vị trí ban đầu của động cơ nhóm sử dụng 1 cảm biến hành trình. Khi bắt đầu khởi động, robot chạy chương trình về gốc để xác định vị trí ban đầu của bánh xe.

```
/*Turn 4 motors till reaching end stop*/
for(i = 0; i < HOME_SPEED; i+=20)
{
    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, 0);
    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, i);

    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, 0);
    HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_4, i);

    HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, 0);
    HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, i);

    HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_3, 0);
    HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, i);
}
```

```

while((axis1_flag != 1) || (axis2_flag != 1) || (axis3_flag != 1) || (axis4_flag != 1))
{
    /*Check each motor if reaching end stop*/
    if(HAL_GPIO_ReadPin (GPIOA, GPIO_PIN_15) == 1)
    {
        __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, 0);
        axis1_flag = 1;
    }
    if(HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_3) == 1)
    {
        __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_4, 0);
        axis2_flag = 1;
    }
    if(HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_4) == 1)
    {
        __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_2, 0);
        axis3_flag = 1;
    }
    if(HAL_GPIO_ReadPin (GPIOB, GPIO_PIN_5) == 1)
    {
        __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_4, 0);
        axis4_flag = 1;
    }
}

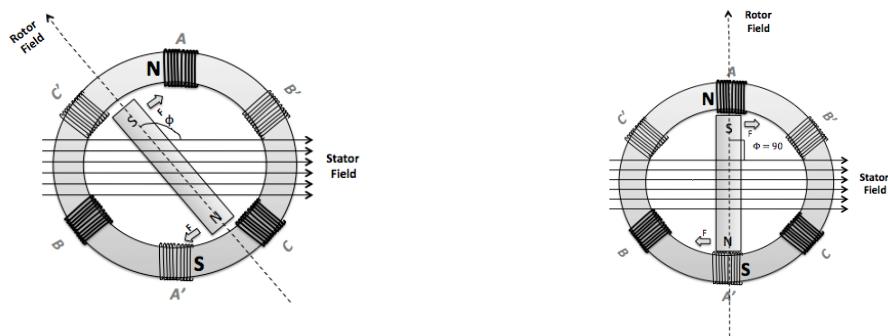
```

Chu trình khởi động của robot sẽ quay tất cả các động cơ cho đến khi đạt được vị trí cuối cùng của cảm biến hành trình (end stop), sau đó tắt cả các động cơ trực dừng lại khi đạt được vị trí gốc (home position) bằng cách đọc tín hiệu đầu vào từ các cảm biến hành trình.

3.3.6. Điều khiển động cơ BLDC

a. Phương pháp FOC (Field Oriented Control)

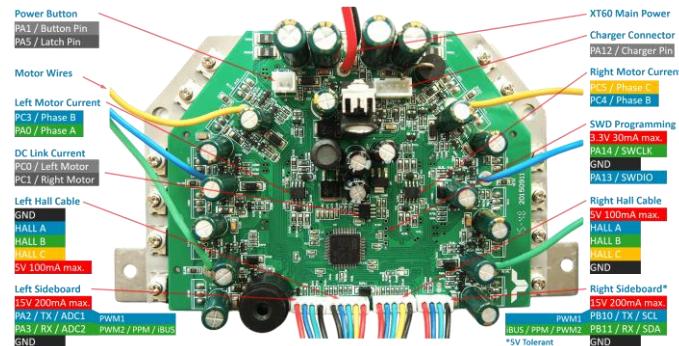
Phương pháp FOC là phương pháp điều khiển tiên tiến sử dụng trong động cơ BLDC và động cơ bước để cung cấp hiệu suất tốt hơn ở tốc độ cao và kiểm soát vị trí chính xác hơn. Nó dựa vào điều khiển các dòng điện trong động cơ để tối ưu hóa vector dòng chéo và tạo mô-men xoắn cực đại. Điều này giúp động cơ hoạt động hiệu quả hơn và tiết kiệm năng lượng.



Hình 3.21: Trường từ Rotor và Trường từ Stator

b. Sử dụng driver BLDC để điều khiển động cơ

BLDC Hover board sử dụng phương pháp FOC để điều khiển động cơ BLDC bằng vi xử lý ARM tích hợp sẵn. Để điều khiển mỗi cặp động cơ BLDC sau và trước, chúng ta gửi lệnh tới vi xử lý ARM này thông qua giao tiếp UART.



Hình 3.22: Driver BLDC

Trước khi sử dụng driver này trong dự án, nhóm đã thay đổi một số điều trong phần mềm firmware của bảng đạp. Điều đầu tiên là thay đổi hình thức lệnh giao tiếp để có thể điều khiển mỗi động cơ một cách độc lập. Thứ hai, chúng ta thay đổi chế độ của FOC sang chế độ tốc độ.

Hình dưới đây là thông tin được truyền cho driver động cơ BLDC để điều chỉnh tốc độ của 2 bánh và thông tin được driver trả lại

Byte Number	0	1	2	3
Content	0xAB	0xCD	V_right (High Byte)	V_right(Low Byte)
	4	5	6	7
	V_left (High Byte)	V_left(Low Byte)	Checksum (High)	Checksum (Low)

Hình 3.23: Khung thông điệp MCU gửi cho driver

0	1	2	3	4	5	6	7	8
0xABCD	cmd1	cmd2	Vr	VL	BatVol	BatTemp	cmdLed	Checksum

Hình 3.24: Khung thông điệp driver gửi cho MCU

3.3.7. Giao thức kết nối giữa vi điều khiển và PC

Trong dự án này, nhóm đã sử dụng RS485 để truyền dữ liệu giữa PC và MCU. RS485 là tiêu chuẩn truyền thông linh hoạt nhất trong loạt tiêu chuẩn được định nghĩa bởi EIA, vì nó hoạt động tốt trên cả bốn điểm. Đó là lý do tại sao RS485 hiện đang là giao diện truyền thông rộng rãi được sử dụng trong các ứng dụng thu thập dữ liệu và điều khiển, nơi nhiều nút giao tiếp với nhau.

Vì yêu cầu của robot có thể bị ảnh hưởng bởi từ trường, hoạt động trên đường gò ghè và cần truyền thông liên tục giữa MCU và PC, việc sử dụng truyền thông RS485 là lựa chọn phù hợp.

3.3.8. Khung truyền (Message Frame)

Để đảm bảo dữ liệu được truyền liên tục, chúng ta tích hợp một số thông tin như tốc độ, góc lái, gia tốc... vào một khung thông điệp duy nhất.

Byte Number	0	1 - 4	5 - 8	9-12	10	11
Content	0x24	Vx	Vy	Angular velocity	0x0D	0x0A

Hình 3.25: Khung thông điệp từ PC tới MCU

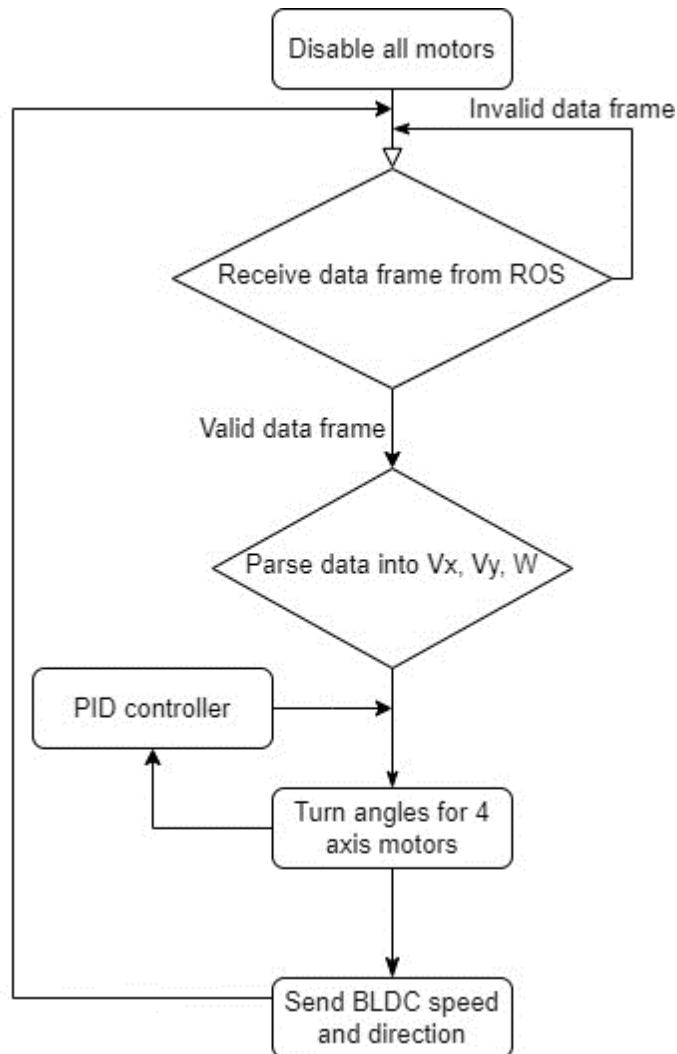
Byte Number	0	1 - 4	5 - 8	9-12	10	11
Content	0x24	acceleration	Vx real	Vy real	0x0D	0x0A

Hình 3.26: Phản hồi từ MCU tới PC

Để giảm và tránh thông điệp không đảm bảo trong quá trình truyền thông, nhóm sử dụng hai byte 0x0D và 0x0A. Cách này giúp đảm bảo tính toàn vẹn của thông điệp và định dạng chính xác cho dữ liệu được gửi từ MCU đến PC.

3.3.9. Chu trình điều khiển của hệ thống

Chương trình của robot bao gồm nhận các gói tin của máy tính gửi thông qua giao thức UART và xử lý gói tin này. Các gói tin chứa các thông tin về vận tốc của robot, từ đó vi điều khiển sẽ xử lý và điều khiển các động cơ để đạt được vận tốc mong muốn.



Hình 3.27: Sơ đồ thuật toán điều khiển robot

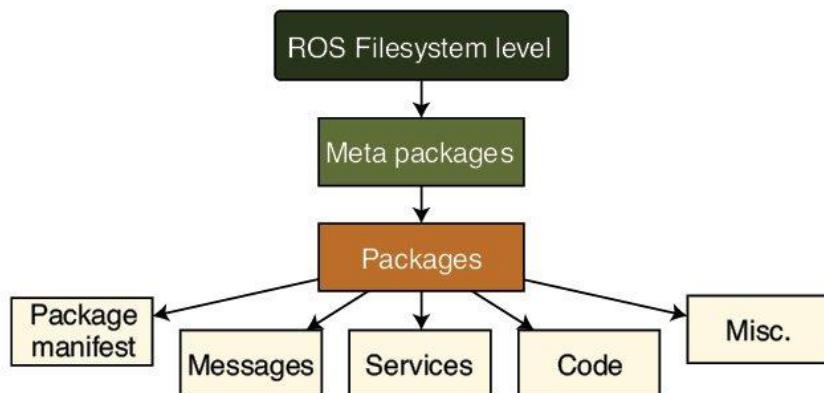
CHƯƠNG 4: ROS VÀ THUẬT TOÁN ĐIỀU KHIỂN

4.1. Tổng quan về Linux và Robot Operating System (ROS)

Linux là một hệ điều hành mã nguồn mở. Phát triển từ năm 1991, cho đến ngày nay Linux đã có rất nhiều phiên bản Ubuntu, Mint, Debian,... và được rất nhiều người ưa chuộng bởi tính miễn phí và khả năng chạy trên các máy tính cấu hình yếu.

ROS (Robot Operating System) là một hệ thống phần mềm mã nguồn mở phổ biến trên hệ điều hành Linux cho phép người dùng lập trình và phát triển robot. Là một nền tảng phần mềm phổ biến cho những người xây dựng và phát triển robot, ROS cho phép người dùng chia sẻ mã nguồn và ý tưởng của mình dễ dàng hơn và điều đó trở thành một ưu điểm của nó: người dùng không cần bỏ nhiều thời gian viết tắt cả cơ sở hạ tầng phần mềm trước khi robot có thể di chuyển.

Hình 4.1 bên dưới mô tả hoạt động của ROS gồm các thành phần chính là các tệp tin được bố trí từ bên trên xuống bên dưới theo thứ tự và hoạt động như sau: Metapackages, Packages, Packages manifest, Misc, Messages, Services và Codes.



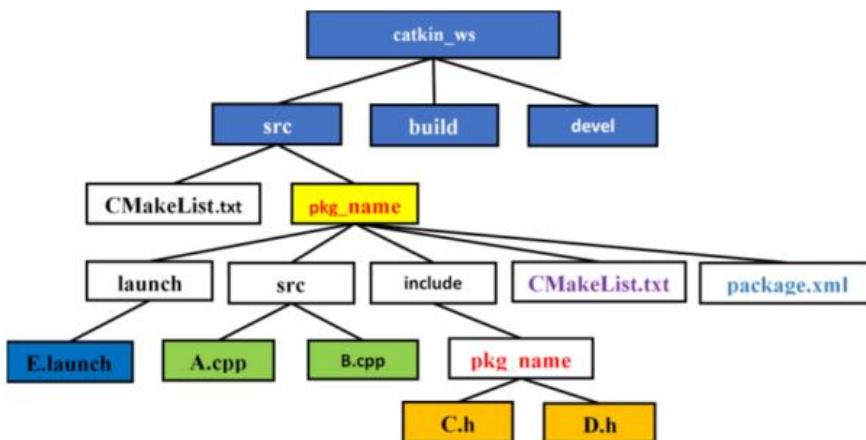
Hình 4.1: FILESYSTEM LEVEL của ROS

Trong đó gói tổng (**Metapackages**) là một nhóm các gói (**packages**) có liên quan tới nhau. Ví dụ, trong ROS có một gói tổng tên là **Navigation**, gói này có chứa tất cả các gói liên quan tới việc di chuyển robot, bao gồm di chuyển thân, bánh, các thuật toán liên quan như **Kalman**, **Particle filter**, ... Khi cài đặt gói tổng, nghĩa là tất cả các gói con trong nó cũng được cài đặt.

Gói (**Packages**), khái niệm gói rất quan trọng, chúng ra có thể nói rằng các gói chính là các nguyên tử cơ bản nhất tạo nên ROS. Trong một gói gồm có: ROS node, datasets, configuration files, source files, tất cả được gói lại trong một “gói”. Tuy nhiên, mặc dù có nhiều thành phần trong một gói, nhưng để làm việc, chúng ta chỉ cần quan tâm đến 2 thành phần, đó chính là src folder, chứa source code của chúng ta, và file Cmake.txt, đây là nơi ta khai báo những thư viện cần thiết để thực thi (**compile**) code.

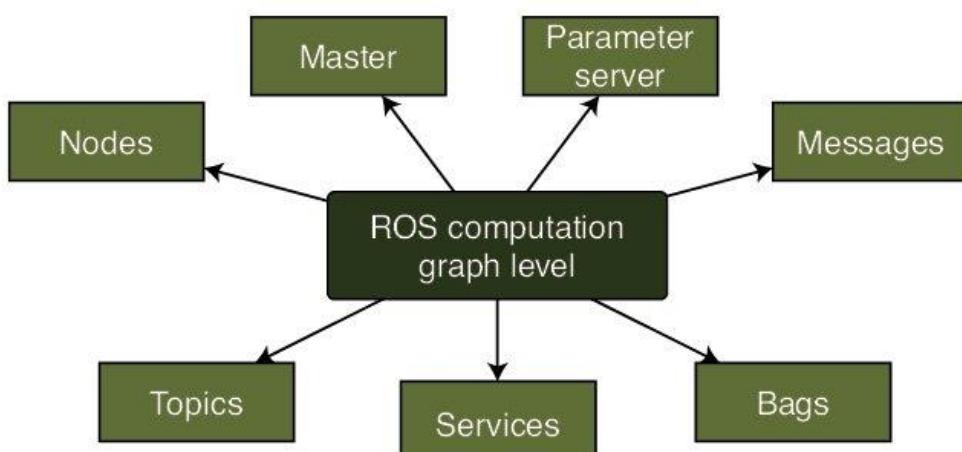
Trong dự án này, linux và ROS đã được cài đặt trên bo mạch chủ máy tính nhúng mini với phiên bản Ubuntu 20.04 và ROS-Noetic.

Để làm việc với ROS ta có thể sử dụng một trong hai công cụ có thể tạo gói đó là catkin. Với cây thư mục làm việc có dạng sau.



Hình 4.2: Cây thư mục trong ROS

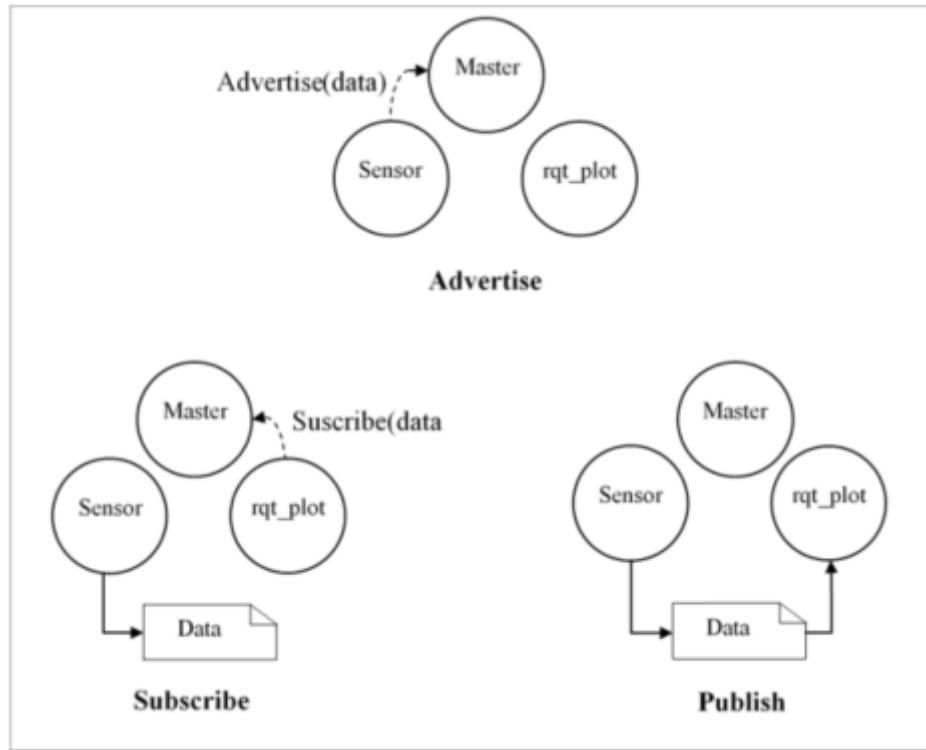
Các thành phần chính của ROS là mạng của tất cả các tác vụ ROS trong một hệ thống robot xử lý và trao đổi dữ liệu giữa các tác vụ đó.



Hình 4.3: Các thành phần chính của ROS

Các node là các thành phần chính trong ROS, chúng là những tập tin thực thi được dùng để giao tiếp với các node khác. Việc giao tiếp giữa các node được mô tả trong hình 4.4.

Dữ liệu giao tiếp qua lại giữa các node được mô tả bằng các thành phần trong ROS như: parameter service, message, topic, service,... Ngoài ra ROS còn cung cấp một công cụ cho phép ghi lại những dữ liệu trên theo thời gian và giữ nguyên mối tương tác giữa chúng, công cụ đó là rosbag.



Hình 4.4: Giao tiếp giữa các node trong ROS

Nodes: là đơn vị cơ bản nhằm hỗ trợ giao tiếp với các thành phần cấu thành nên robot. Ví dụ như một con robot thường có các node như Laser scanner, Camera, Node gửi vận tốc đến thiết bị điều khiển động cơ. Các nodes này có thể giao tiếp và tương tác với nhau qua Master.

Master: đóng vai trò kết nối các node với nhau. Do đó, master luôn được khởi động đầu tiên bằng câu lệnh roscore sau đó thì bạn có thể gọi bất kỳ các node nào trong hệ thống. Sau khi gọi xong, các node có thể kết nối và tương tác với nhau.

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.

Publish là gửi thông tin.

Subscribe: là đăng ký nhận thông tin.

Services: Là một giao tiếp trao đổi dữ liệu/ thông tin giữa hai node thông qua phương thức request và response. Thường được áp dụng trong trường hợp việc thực hiện một lệnh cần nhiều thời gian xử lý nên dữ liệu tính toán được lưu ở server và sẽ dùng khi cần xử lý.

Bags: là một định dạng tệp trong ROS dùng để lưu trữ dữ liệu message với phần mở rộng là .bag có vai trò quan trọng và có thể được xử lý, phân tích bởi các công cụ mô phỏng trong ROS như Rviz.

4.2. Tổng quan về navigation stack

Để điều khiển xe cần có rất nhiều gói hỗ trợ cho các công việc khác nhau như: Tạo quỹ đạo di chuyển, tránh vật cản, xác định vị trí xe, quét bản đồ,... và thật may mắn các gói này đã được cộng đồng phát triển ROS nghiên cứu tập hợp thành một gói lớn mang tên (metapackage) navigation.

Navigation stack: là một gói điều hướng 2D lấy thông tin từ mô hình, luồng cảm biến, đặt mục tiêu và đưa ra vận tốc điều khiển xe. Thông tin chi tiết về các gói cũng như mã nguồn có thể tìm thấy trên trang wiki phần phụ lục [1].

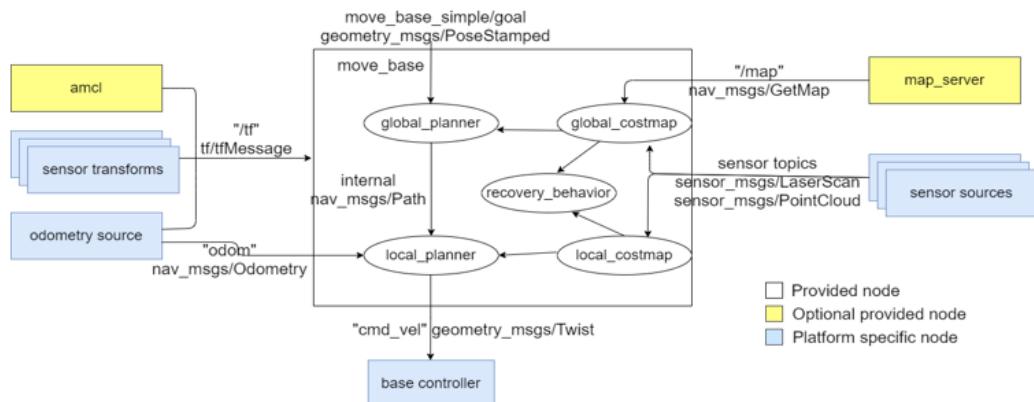
Sau đây là giới thiệu sơ lược về một gói được dùng trong quá trình hoạt động của robot.

Move_base: Cung cấp vận tốc cho xe khi có một mục tiêu (goal point) được chỉ ra. Gói này sẽ liên kết với các gói quỹ đạo để thực hiện nhiệm vụ điều hướng cho xe di chuyển.

- Theo dõi các topic trạng thái:
 - + Move_base/goal: điểm đích trên bản đồ.
 - + Move_base/cancel: yêu cầu hủy bỏ điểm đích vừa gọi.
- Xuất bản các topic trạng thái:
 - + Move_base/feedback: phản hồi chứa vị trí hiện tại với gốc tọa độ trên bản đồ.

- + Move_base/status: cung cấp thông tin trạng thái về các mục tiêu được gửi đến move_base.
- Các topic theo dõi:
 - + Move_base_simple/goal: cung cấp cho người dùng không quan tâm đến việc theo dõi trạng thái thực hiện mục tiêu của họ.
- Các topic xuất bản:
 - + Cmd-vel: luồng lệnh vận tốc điều khiển vận tốc dài và vận tốc góc xe.

NAVIGATION STACK SETUP



Hình 4.5: Sơ đồ khái niệm move_base

Global_planner: gói này cung cấp triển khai một kế hoạch toàn cục, nội suy điều hướng xe. Gói này sử dụng 2 thuật toán Dijkstra's và A* để thực hiện việc tạo quỹ đạo. Người dùng cần phải cấu hình để lựa chọn giữa một trong 2 thuật toán này:

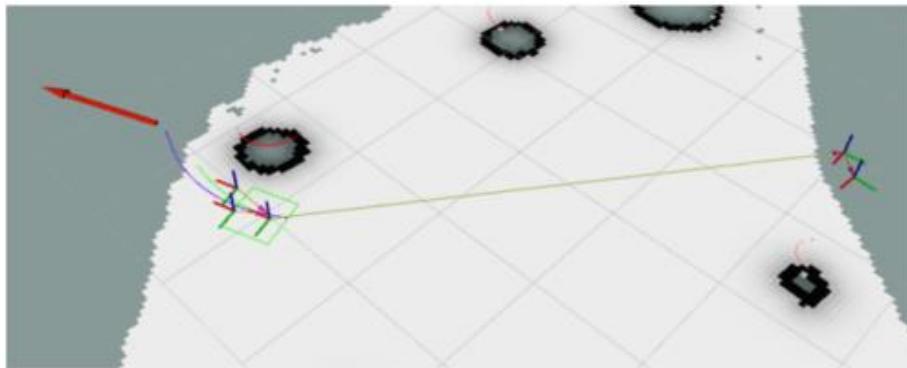
- Dijkstra's: Là thuật toán giải quyết bài toán đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng không có cạnh mang trọng số âm.
- A*: Thuật toán tìm kiếm đường đi từ một nút khởi đầu tới một nút đích cho trước. Thuật toán sử dụng một đánh giá heuristics để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó.
- Node này sẽ nhận tín hiệu từ node global_costmap.

Xuất bản các topic:

- ~<name>/plan: quỹ đạo cuối cùng được tính toán có dạng mảng chứa vị trí của xe(đối với gốc tọa độ trên bản đồ): vị trí theo tọa độ x,y,z và góc quay đối với hệ trục tọa độ gốc.

Local_planner: Sử dụng thuật toán Dynamic Window Approach (dwa_local_planner) cung cấp cho robot một quỹ đạo di chuyển trong phạm vi gần,

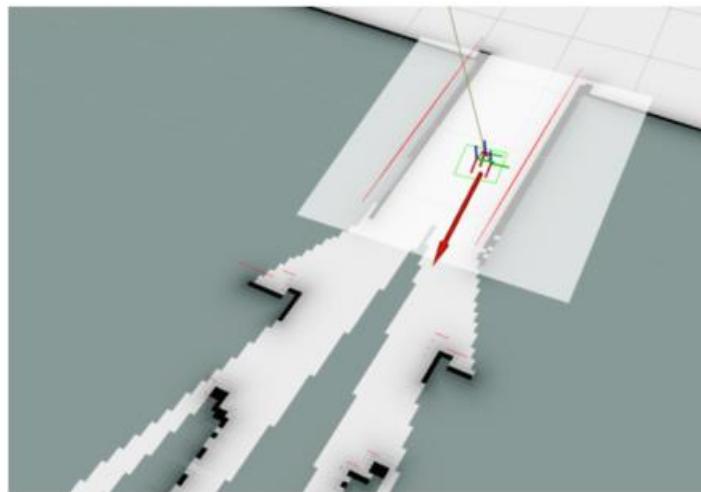
cho phép nó phản ứng nhanh với các vật cản và đưa ra quỹ đạo di chuyển tránh vật phù hợp.



Hình 4.6: Local plan (lục) global plan (lam)

Về phần thuật toán sẽ được trình bày cụ thể trong phần sau.

Costmap_2d: Gói này bao gồm global_costmap và local_costmap. Được sử dụng với mục đích giúp việc điều khiển xe dựa trên tín hiệu bản đồ lưới thuận tiện hơn. Cụ thể, việc xây dựng ra costmap tức là định nghĩa một khoảng trên bản đồ (dựa trên các đường biên vật cản hoặc tín hiệu cảm biến laser) mà nếu khói tâm của robot ở trong khoảng đó tức là đã xảy ra va chạm. Từ đó, ta có thể đơn giản xác định khi nào robot va chạm với các vật thể chỉ nhòe vị trí khói tâm của robot.



Hình 4.7: Global costmap và local cost map

Như trong hình, Global costmap là những viền mờ ở sát vật cản trên toàn bản đồ, local costmap là những viền mờ ở sát vật cản nằm trong vùng local costmap (vùng vuông sáng).

Tf: tf là một gói đặc biệt cần thiết trong việc thiết kế và lập trình robot trong ROS. Nó cho phép theo dõi nhiều trực tọa độ khác nhau cùng một lúc và hỗ trợ chuyển đổi tọa độ của một điểm từ hệ tọa độ này sang hệ tọa độ khác. Việc thiết

lập mối quan hệ giữa các hệ trục tọa độ được thực hiện trong file mô tả về phần cứng của robot(thường có dạng .urdf).

Amcl(Adaptive Monte Carlo localization): là một gói giúp định vị robot nhờ thuật toán xác xuất thống kê trong môi trường 2D. Gói này được sử dụng nhằm mục đích định vị chính xác robot trong một khu vực ta đã có bản đồ từ trước. Gói không thể sử dụng trong các môi trường quá đơn giản(quá ít vật cản để có thể xác định vị trí thật của robot như đường hầm, hoặc căn phòng trông rộng lớn). Thuật toán của amcl có thể được mô tả giản lược qua các bước sau:

- Bước 1: Chuẩn bị. Trước hết để sử dụng thuật toán này ta cần có bản đồ lưới dạng (topic/map), vị trí khởi tạo ước lượng, tf (để chuyển đổi tín hiệu lấy được từ cảm biến từ hệ trục cảm biến đưa về hệ trục tọa độ tại khói tâm xe) và tín hiệu cảm biến.
- Bước 2 : Thuật toán sẽ tạo ra hàng loạt các vector phủ trên toàn bản đồ (số lượng vector tạo ra có thể cấu hình được).
- Bước 3: Điều khiển xe di chuyển (bằng thuật toán tự động hoặc qua bàn phím). Các vector được tạo ra cũng di chuyển cùng vận tốc dài và vận tốc góc so với khói tâm robot. Thuật toán sẽ lọc ra các vector có vị trí tương đối với vật cản trên bản đồ tương ứng với tín hiệu cảm biến laser đang thu được. Từ đó lọc ra được vị trí chính xác của xe.





Hình 4.8: Mô tả AMCL

Thuật toán kết thúc khi sai số của nó nhỏ hơn một ngưỡng nào đó mà người dùng đặt ra.

Thuật toán này theo dõi các topic:

- Scan: Laser scan
- Initialpose: bao gồm cả vị trí và hướng (tính theo quaternion)
- Map: bản đồ dưới dạng lưới

Thuật toán xuất ra các topic:

- Amcl_pose: vị trí tương đối dựa trên xác suất đi kèm với sai số.
- Particlecloud: bộ các vector (vị trí dự đoán của robot) được duy trì bởi bộ lọc của thuật toán.
- Tf: cập nhật lại vị trí của robot so với điểm gốc tọa độ trên bản đồ.

Base_controller: Là một gói dùng để trực tiếp cấp lệnh điều khiển đến robot. Gói này được người lập trình robot tự phát triển tùy thuộc vào dạng robot của mình. Trong bài đồ án này, chức năng của gói này là nhận dữ liệu điều khiển từ move_base dưới dạng vận tốc dài và vận tốc góc của khói tâm robot. Sau khi tính toán, phân tích liền đưa ra tín hiệu điều khiển dạng xung PWM xuống MCU để điều khiển vận tốc từng bánh xe.

Odometry: Odometry được dùng để tính toán vị trí tương đối của robot trong môi trường với thông tin đầu vào là dữ liệu đến từ những cảm biến như: encoder và imu. Từ đó tính ra được vị trí hiện tại so với vị trí ban đầu của robot. Gói này cũng được người lập trình phát triển dựa trên cấu hình robot và mô hình động học của từng robot. Tuy nhiên, định dạng đầu ra được chuẩn hóa bao gồm vị trí và hướng của robot, kèm theo sai số và một số hệ thời gian.

Để viết chương trình odometry, trước hết ta cần phải tính toán xong bài toán động học của robot. Sau đó dựa vào tín hiệu từ các cảm biến đê thu được vị trí các khớp, từ đó theo công thức tìm ra được vị trí của robot. Công việc đơn giản còn lại là đẩy thông tin ra topic tên là odom.

Lidar_ros: Đây là gói cho phép giao tiếp giữa lidar và ROS. Với gói này, tín hiệu đầu ra của lidar sẽ chứa tín hiệu vật cản và mặc định đẩy vào topic/scan/

Tham số của gói:

- Port: tên serial port, mặc định là /dev/USB1
- Baudrate: mặc định là 115200
- Frame ID: tên hệ tọa độ cảm biến laser: laser_frame
- Angle_max/angle_min: góc thu tín hiệu: 180/-180 đối với cảm biến 360 độ
- Range_min/range_max: khoảng cách gần nhất/xa nhất đến vật cản mà cảm biến có thể phát hiện (0.08 -16 m)
- Samp_rate: tần số lấy mẫu, mặc định là 4
- Frequency: tần số quét, mặc định là 7

4.3. Tổng quan về SLAM

SLAM (Simultaneous localization and mapping) là hệ thống sử dụng thông tin ảnh thu được từ camera, hoặc Lidar hay 1 số loại cảm biến khác để tái tạo môi trường bên ngoài bằng cách đưa thông tin môi trường vào một map (2D hoặc 3D), từ đó thiết bị (robot, camera, xe) có thể định vị (localization) đang ở đâu, trạng thái, tư thế của nó trong map để tự động thiết lập đường đi (path planning) trong môi trường hiện tại.

Điều khiển tự động thiết bị robot chia làm 3 vấn đề chính:

- Định vị (localization).
- Tái tạo môi trường (mapping).
- Hoạch định đường đi (path planning).

Ban đầu, hai vấn đề định vị và tái tạo môi trường được nghiên cứu độc lập, tuy nhiên, sau khi nhận thấy:

- Định vị: cần xác định vị trí hiện tại của robot dựa vào bản đồ tái tạo.
- Tái tạo bản đồ: cần xác định vị trí của đối tượng trong bản đồ, để xây dựng bản đồ chính xác nhất, ít sai số.

Các vấn đề SLAM phát sinh khi robot không có quyền truy cập vào bản đồ của môi trường, nó cũng không có quyền truy cập vào các tư thế riêng của nó. Thay vào đó, tất cả những gì nó được cung cấp là các phép đo thông tin về thành phần của môi trường ứng với từng tư thế của robot $z_{1:t}$ và các phép điều khiển $u_{1:t}$.

Có hai dạng chính của thuật toán SLAM, cả hai đều quan trọng như nhau. Thứ nhất được gọi là SLAM trực tuyến, thứ hai được gọi là SLAM đầy đủ.

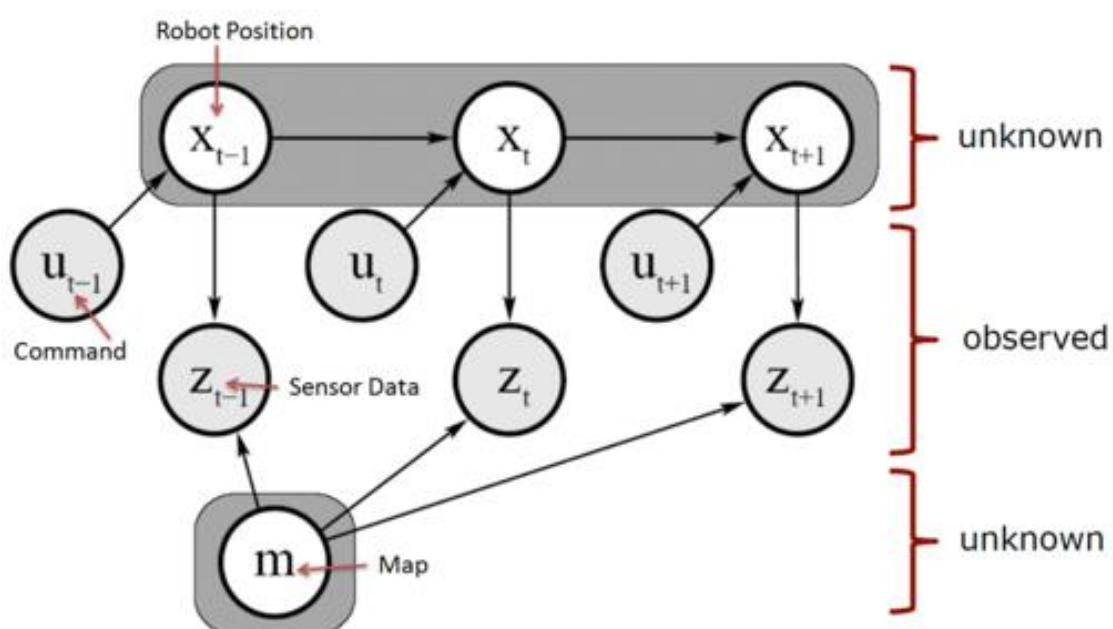
- SLAM trực tuyến:

SLAM trực tuyến liên quan đến việc ước lượng hậu quả của tư thế nhất thời cùng với bản đồ: $p(x_t, m | z_{1:t}, u_{1:t})$.

Trong đó: x_t là tư thế tại thời điểm t , m là bản đồ, và $z_{1:t}$ và $u_{1:t}$ lần lượt là các phép đo và phép điều khiển. Bài toán này được gọi là bài toán SLAM trực tuyến vì nó chỉ liên quan đến việc ước lượng tư thế của robot tại thời điểm t .

- SLAM đầy đủ:

Trong SLAM đầy đủ là tìm cách tính toán trên toàn bộ đường đi $x_{1:t}$ cùng với bản đồ, thay vì chỉ tư thế x_t hiện tại. Nói cách khác, với SLAM đầy đủ, chúng ta có thể hình dung như quỹ đạo của robot $p(x_{1:t}, m | z_{1:t}, u_{1:t})$.



Hình 4.9: Vị trí robot x_t và bản đồ m

Hình 4.9 cho thấy tổng quan về thuật toán SLAM, u_t là lệnh nhóm đồ án đưa ra cho robot hoặc dữ liệu dự đoán. trong khi z_t là dữ liệu mà robot nhận được từ

máy quét laser hoặc máy ảnh Kinect, hoặc chúng ta có thể gọi chúng là dữ liệu đo lường. Từ những đầu vào này, nhóm đồ án có thể tạo một bản đồ và ước tính vị trí tư thế của robot trong bản đồ này.

Để robot có thể thực hiện được nhiệm vụ thì ta cần cung cấp cho robot một bản đồ của môi trường. Việc có một bản đồ chính xác sẽ cho phép hệ thống có thể hoạt động trong các môi trường phức tạp mà chỉ cần dựa vào cảm biến có sẵn trên robot mà không bị phụ thuộc vào các hệ thống tham chiếu bên ngoài như GPS.

SLAM là một vấn đề khó giải quyết vì phản ánh giữa các không gian được quan sát và bản đồ là không xác định, bản đồ và vị trí ước tính tương quan với nhau. Ngoài ra, việc chọn sai các liên kết dữ liệu có thể gây ra hậu quả khó lường vì robot sẽ cập nhật vị trí của nó dựa trên thông tin sai. Một tính năng chính trong SLAM là phát hiện các khu vực đã thăm trước đó để giảm lỗi bản đồ.

- SLAM Gmapping

Hector là viết tắt của cụm từ "Heterogeneous Generating Team of Robots", tên của một nhóm nghiên cứu người máy tại TU Darmstadt. Slam gmapping sử dụng nút hector_mapping để thiết lập bản đồ của môi trường và đồng thời ước tính tư thế vị trí 2D của nền tảng máy quét laser. Không giống như Gmapping, Slam gmapping không dựa trên bộ lọc Rao Blackwellized Particle để lập bản đồ.

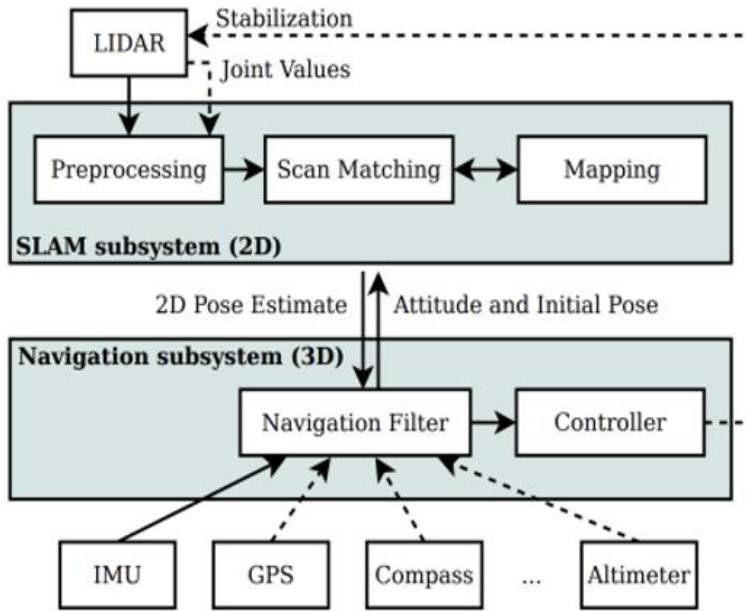
Thuật toán có gắng tìm sự liên kết tối ưu của các điểm cuối của quá trình quét laser với bản đồ đã được xây dựng bằng cách tìm phép biến đổi phần cứng ξ^* :

$$\xi^* = \operatorname{argmin} \sum_i^n [1 - M(S_i(\xi))]^2 \quad (4.1)$$

Trong đó: Hàm $M(S_i(\xi))$ là giá trị bản đồ tại $S_i(\xi)$ là tọa độ trong không gian thật của điểm cuối của quá trình quét laser. Với ước lượng trước là ξ , biến đổi bước $\Delta\xi$ được ước lượng bằng cách tối ưu hóa sai số sao cho:

$$\xi^* = \operatorname{argmin} \sum_i^n [1 - M(S_i(\xi + \Delta\xi))]^2 \quad (4.2)$$

Hình dưới cung cấp tổng quan về ngăn xếp thuật toán Hector. Dữ liệu được xử lý trước từ LiDAR được chuyển tiếp đến một thuật toán đối sánh quét.



Hình 4.10: Tổng quát về Hector Mapping and Navigation stack

- Thuật toán di chuyển tránh vật cản được sử dụng trong đồ án này nằm trong gói move_base (thuộc gói liên kết navigation) áp dụng cả hai phương pháp tránh vật cản để xây dựng ra chiến lược di chuyển tránh vật của robot. Đầu tiên phương pháp global sẽ được sử dụng một lần để tạo ra chiến lược di chuyển tối ưu đến điểm đích, sau đó robot sẽ sử dụng phương pháp local để di chuyển tránh vật cản bám theo chiến lược tối ưu ban đầu.

4.3.1. Phương pháp global planner

a. Thuật toán Dijkstra

Giới thiệu

Như đã trình bày ở trên, Thuật toán Dijkstra là thuật toán tìm đường đi từ điểm xuất phát đến 1 (hoặc các đỉnh) còn lại trong đồ thị trọng số.

Để thực hiện lập kế hoạch đường đi, các gói ROS đã có 1 số hỗ trợ về thuật toán tìm đường. Một trong số đó là navfn (navigation function), một chức năng điều hướng nội suy nhanh được tạo ra cho robot. Người lập kế hoạch giả định một rô bót và hoạt động trên costmap để tìm kế hoạch chi phí tối thiểu từ điểm bắt đầu đến điểm kết thúc trong lưới. Thuật toán được áp dụng trong việc lập kế hoạch này là thuật toán tìm đường nổi tiếng của Dijkstra đối với môi trường tĩnh (không có vật cản động).

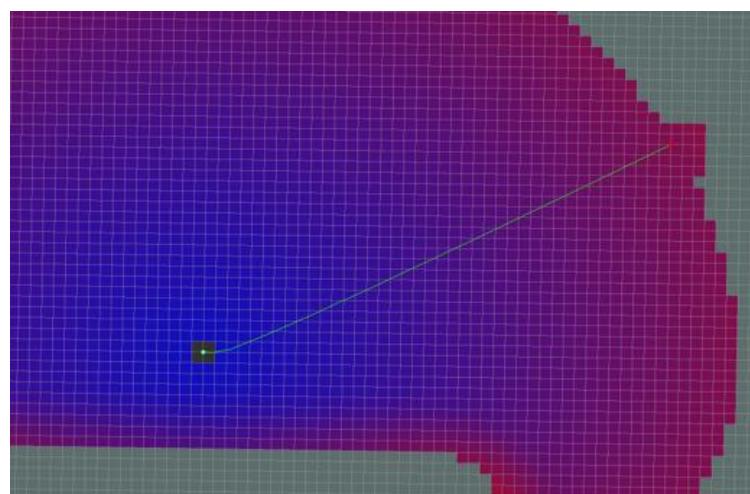
Path Planning và Dijkstra trong ROS

Để di chuyển robot theo quỹ đạo đặt trước, ta sử dụng package move_base của ROS. Package này sẽ kết nối với global_planner để đưa ra các lệnh di chuyển cho robot. Global_planner sử dụng thuật toán Dijkstra như phần giới thiệu bên trên để đưa ra quãng đường đi.

Để thuật toán Dijkstra có thể hoạt động được ta cần có file cấu hình một số thông số quan trọng của costmap hiện tại trong file global_costmap_param.yaml:

- Inflation_layer: Phần scale tỷ lệ các vật cản để robot không chạm vào.
- Cost_scaling_factor: 5.0 (tỷ lệ theo cấp số nhân mà tại đó chi phí chướng ngại vật giảm xuống).
- Inflation_radius: 0.5 (khoảng cách tối đa từ chướng ngại vật mà từ đó phát sinh chi phí cho việc lập kế hoạch đường đi).
- Resolution: độ phân giải của bản đồ (m/cell).
- Publish_scale: tỷ lệ khuếch đại cho các vị trí tiềm năng.
- Lethal_cost = 253 (default).
- Neutral_cost = 50 (default).
- Cost_factor = 3.0 (default).

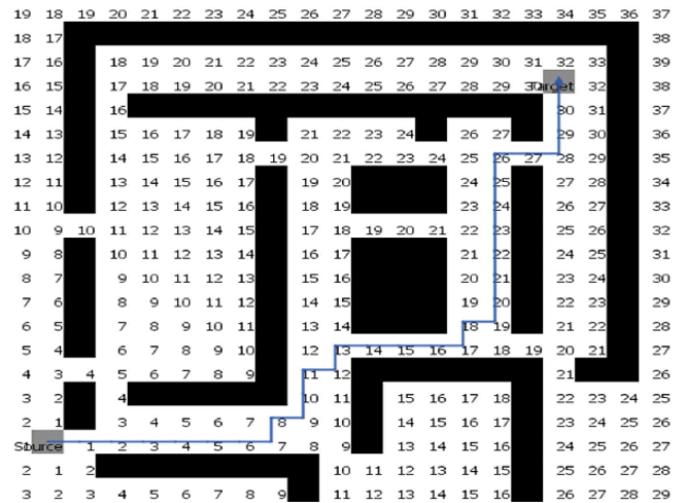
Thuật toán sẽ chia nhỏ bản đồ thành các ô (grid map) để tiến hành tính toán.



Hình 4.11: Dijkstra Path Planning

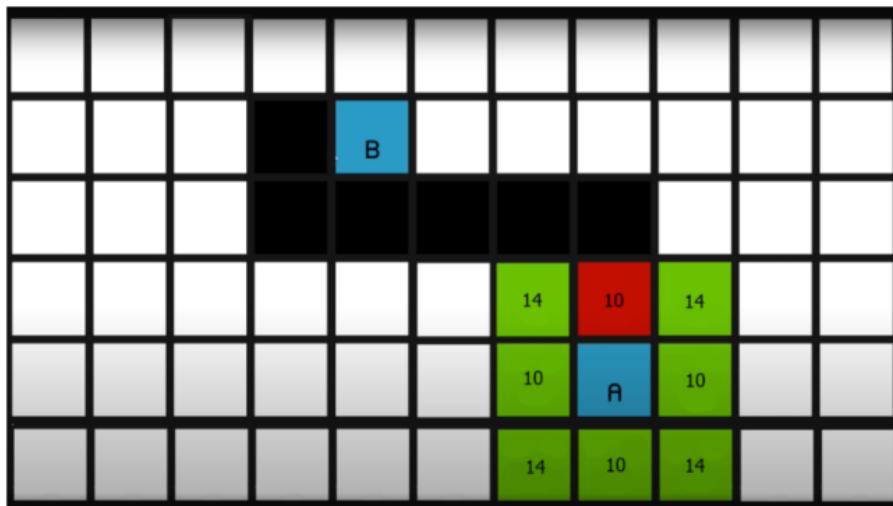
Thuật toán được sử dụng trong trường hợp:

- Robot có sẵn bản đồ.
- Robot đã được định vị trong bản đồ đó.
- Robot có thể di chuyển tự do lên xuống, trái phải.



Hình 4.12: Ví dụ về thuật toán Dijkstra trong bản đồ dạng lưới

Ví dụ về thuật toán Dijkstra

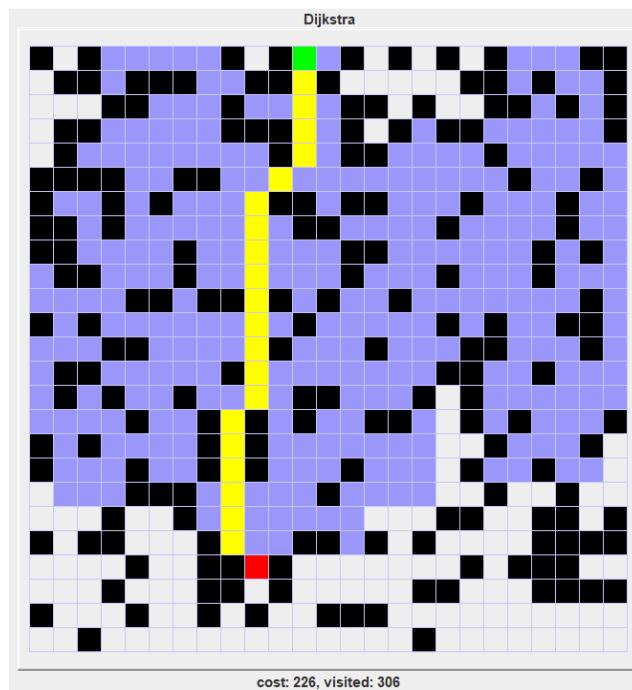


Hình 4.13: Mô tả thuật toán Dijkstra

Giả sử robot hiện đang ở điểm A và đích đến của nó là điểm B. Ta xét 8 điểm xung quanh vị trí điểm A và tính ra được khoảng cách từ các điểm này đến A là giá trị được viết trong ô. Chọn một ô có giá trị nhỏ nhất (ô số 10 phía trên) và tiếp xúc xác định giá trị các ô trống xung quanh nó bằng cách lấy giá trị hiện tại trong ô (100 cộng với khoảng cách từ ô đó đến ô xung quanh, nếu giá trị này nhỏ hơn giá trị hiện tại trong ô trống thì giá trị ô trống được cập nhật). Trường hợp trên hình, khi cập nhật giá trị xung quanh ô đó thì giá trị ô ngay trái nó tính được là $10 + 10 = 20 > 14$ nên giá trị không được cập nhật. Tiếp tục xét tiếp các ô có giá trị nhỏ nhất còn lại và lặp lại cho đến khi ô B được xét tới (tức là có giá trị nhỏ nhất trong

tất cả các ô). Các ô đi qua tạo nên giá trị nhỏ nhất cho B là quãng đường đi ngắn nhất cần tìm.

Kết quả mô phỏng thuật toán Dijkstra



Hình 4.14: Kết quả mô phỏng thuật toán Dijkstra

Hình 4.14 thể hiện kết quả mô phỏng của thuật toán Dijkstra, các ô và đường màu đen là các vật cản, ô màu cam là điểm đầu, ô màu xanh da trời là điểm đích (điểm mà robot muốn đi tới), các ô màu đỏ là các ô mà thuật toán đã quét để tìm điểm đích, các ô màu xanh lá là các ô dẫn hướng thực hiện để quét tìm điểm đích. Khi các ô màu xanh lá tìm và chạm vào được ô màu xanh da trời thuật toán sẽ cho kết quả đường đi ngắn nhất mà ta cần tìm là đường các ô màu vàng.

b. Thuật toán A*

Giới thiệu thuật toán A*

A* là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là hàm Heuristic.

Từ trạng thái hiện tại A* xây dựng tất cả các đường đi có thể đi dùng hàm ước lượng khoảng cách (hàm Heuristic) để đánh giá đường đi tốt nhất có thể đi. Tùy theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế. A* lưu giữ một tập các đường đi qua đồ thị, từ đỉnh bắt đầu đến đỉnh kết thúc, tập các đỉnh có thể đi tiếp được lưu trong tập Open.

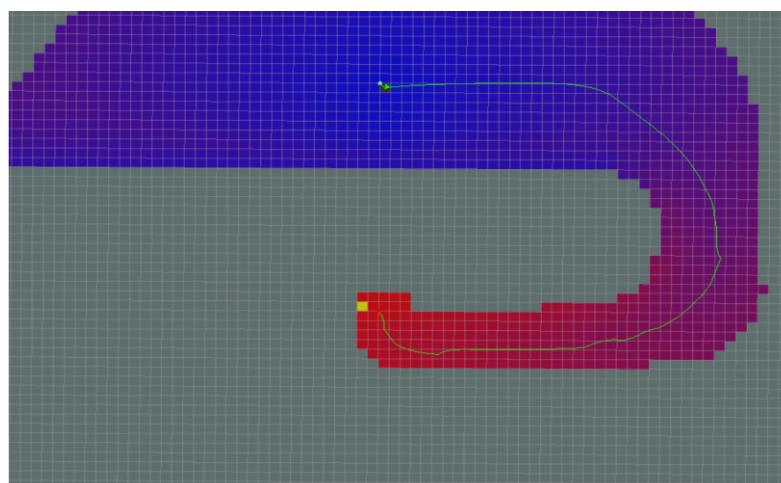
Thuật toán A* trong ROS

Để di chuyển robot theo quỹ đạo đặt trước, ta sử dụng package move_base của ROS. Package này sẽ kết nối với global_planner để đưa ra các lệnh di chuyển cho robot. Global Planner sử dụng thuật toán A* như phần giới thiệu bên trên để đưa ra quãng đường đi ngắn nhất.

Để thuật toán A* có thể hoạt động được ta cần có file cấu hình một số thông số quan trọng của costmap hiện tại trong file global_costmap_param.yaml:

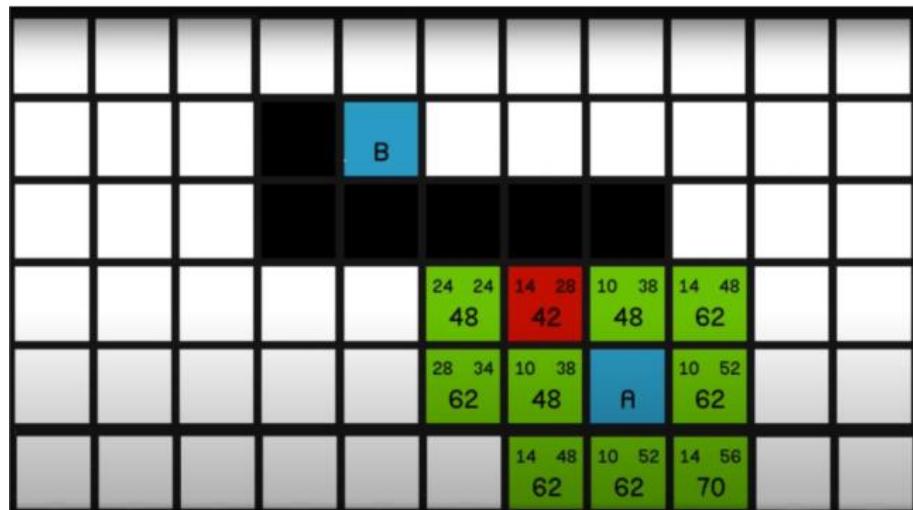
- Inflation_layer: Phần scale tỷ lệ các vật cản để robot không chạm vào.
- Cost_scaling_factor: 5.0 (tỷ lệ theo cấp số nhân mà tại đó chi phí chướng ngại vật giảm xuống).
- Inflation_radius: 0.5 (khoảng cách tối đa từ chướng ngại vật mà từ đó phát sinh chi phí cho việc lập kế hoạch đường đi).
- Resolution: độ phân giải của bản đồ (m/cell).
- Publish_scale: tỷ lệ khuếch đại cho các vị trí tiềm năng.
- Lethal_cost = 253 (default): chi phí “gây chết”, hay chi phí cho chướng ngại vật.
- Neutral_cost = 50 (default).
- Cost_factor = 3.0 (default).

Thuật toán sẽ chia nhỏ bản đồ thành các ô (grid map) để tiến hành tính toán.



Hình 4.15: A* Path Planning

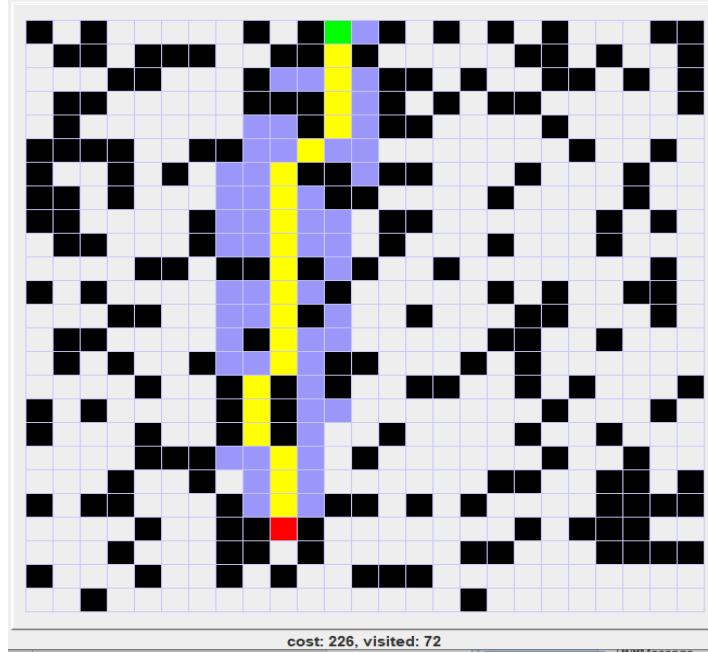
Ví dụ về thuật toán A*



Hình 4.16: Mô tả thuật toán A*

Thuật toán A* yêu cầu phải biết khoảng cách của từng ô đối với điểm đích. Xét điểm A, tính toán giá trị của 8 ô trống xung quanh nó được 3 giá trị: góc trên bên trái là khoảng cách từ ô đó đến A, góc trên bên phải là khoảng cách từ ô đó đến đích B, và trung tâm là tổng của 2 giá trị trên. Sau khi tính toán giá trị các ô xung quanh A, chọn một ô có giá trị nhỏ nhất chưa xét (ô đỏ) và tiếp tục xác định giá trị các ô trống xung quanh nó (ô trống là ô không phải vật cản). Nếu giá trị mới nhỏ hơn giá trị cũ trong ô được tính toán thì cần cập nhật giá trị của ô đó. Lặp lại việc chọn ô có giá trị tổng nhỏ nhất và xét giá trị, cập nhật giá trị các ô xung quanh đến khi ô được chọn để xét là đích B. Danh sách các ô được chọn để B được xét là đường đi ngắn nhất cần tìm.

Kết quả mô phỏng thuật toán A*



Hình 4.17: Kết quả mô phỏng thuật toán A*

Hình 4.17 thể hiện kết quả mô phỏng của thuật toán A*, các ô và đường màu đen là các vật cản, ô màu xanh là điểm đầu, ô màu đỏ là điểm đích (điểm mà robot muốn đi tới), các ô màu tím là các ô mà thuật toán đã quát để tìm điểm đích, các ô màu vàng là các ô dẫn hướng thực hiện để quét tìm điểm đích. Khi các ô màu xanh lá tìm và chạm vào được ô màu xanh da trời thuật toán sẽ cho kết quả đường đi ngắn nhất mà ta cần tìm là đường các ô màu vàng.

4.3.2. Phương pháp local planner

Thuật toán Dynamic Window Approach (DWA) là dùng để tìm ra một tín hiệu điều khiển hợp lý gửi xuống robot nhằm mục đích điều khiển nó đến đích an toàn, nhanh chóng dựa trên global planner đã hoạch định từ trước. Thuật toán này gồm hai bước chính: tìm không gian vector và tối ưu chiến lược.

Tìm không gian vector bước đầu tiên là tập hợp các chiến lược có thể có của robot và lưu vào mảng 2 chiều chứa giá trị v , ω , mảng này gọi là trường V_s . Ánh xạ trường V_s trong một thời gian t để thu được quỹ đạo di chuyển của robot. Để robot di chuyển an toàn, tức là có thể dừng lại trước khi đâm vào vật cản thì chỉ chiến lược của robot phải chấp nhận biểu thức sau:

$$v_a = \{(v, \omega) \mid v \cdot \sqrt{2 \cdot dist(v, \omega) \cdot v_b} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \omega_b}\} \quad (4.3)$$

Trong đó:

- v_a : là chuỗi các giá trị vận tốc (v, ω) cho phép robot dừng trước vật cản mà không có sự va chạm.
- $dist(v, \omega)$: là khoảng cách nhỏ nhất mà robot dừng trước vật cản để không có sự va chạm.
- v_b, ω_b : là vận tốc của vận tốc thẳng và vận tốc xoay tối đa nếu robot di chuyển sẽ gây va chạm với vật cản.

Dynamic window: nhằm hạn chế vận tốc cho phép đối với những vận tốc có thể đạt được trong khoảng thời gian mà trước với vận tốc tối đa của robot. Để Δt là khoảng thời gian mà trong đó vận tốc v, ω sẽ được thực thi để (v_a, ω_a) là vận tốc thực được gửi xuống robot. Từ đó, vận tốc V_d sẽ được định nghĩa như sau.

$$\begin{aligned} V_d = \{(v, \omega) | v \in [v_a - \dot{v} \cdot \Delta t, v_a + \dot{v} \cdot \Delta t] \wedge \omega \\ \in [\omega_a - \dot{\omega} \cdot \Delta t, \omega_a + \dot{\omega} \cdot \Delta t]\} \end{aligned} \quad (4.4)$$

Kết thúc ba bước trên thì ta tìm được không gian tìm kiếm.

$$V_r = V_s \cap V_a \cap V_d \quad (4.5)$$

Tối ưu hóa chiến lược là bước thực hiện sau khi đã có trường các chiến lược V_r nhằm “chấm điểm” các chiến lược trong V_r dựa theo những tiêu chí: hướng về phía mục tiêu, khoảng cách đến vật cản gần nhất, tốc độ di chuyển đến điểm đích. Điểm số của chiến lược được xác định dựa trên biểu thức sau:

$$G(v, \omega) = \alpha \cdot \text{heading}(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot vel(v, \omega) \quad (4.6)$$

Trong đó (α, β, γ) là trọng số của các tiêu chí

- σ là tham số điều chỉnh độ mềm của dãy “điểm số”
- $heading(v, \omega)$ là hàm tiêu chí đánh giá độ lệch giữa góc khi kết thúc quỹ đạo và góc của robot tại điểm đích, giá trị trả về là giá trị góc.
- $dist(v, \omega)$ là hàm tiêu chí đánh giá độ gần giữa quỹ đạo và vật cản gần nhất, trả về giá trị khoảng cách nhỏ nhất đến vật cản khi robot di chuyển theo quỹ đạo.
- $velocity(v, \omega)$ là hàm tiêu chí đánh giá tốc độ di chuyển đến mục tiêu, trả về giá trị vận tốc theo hướng từ vị trí cuối của quỹ đạo đến mục tiêu.

4.4. Ứng dụng các thuật toán điều khiển và thực nghiệm

4.4.1. Xây dựng bản đồ sử dụng Slam gmapping

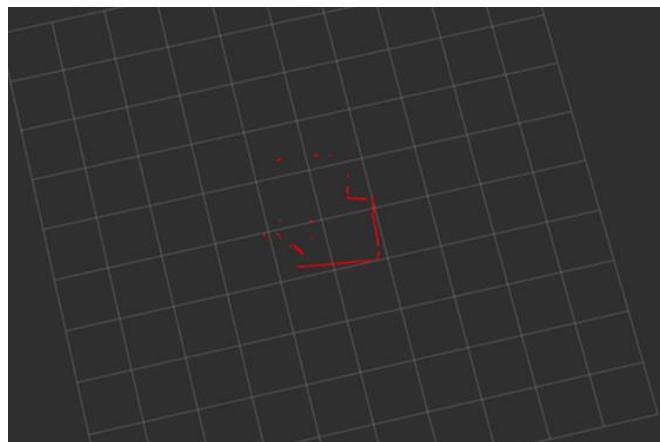
Slam gmapping là một thuật toán sử dụng dữ liệu quét laser để tạo bản đồ. Ưu điểm của Slam gmapping so với các kỹ thuật SLAM khác là nó chỉ yêu cầu dữ liệu quét laser để thực hiện công việc của mình mà không cần tới dữ liệu odometry.

a. Dữ liệu của RPLIDAR A1

RPLidar là cảm biến LIDAR chi phí thấp phù hợp cho ứng dụng SLAM robot trong nhà. Nó cung cấp trường quét 360 độ, với khoảng cách phát hiện vật cản tối đa là 12m. Để quét cần 360 mẫu mỗi vòng quay, có thể đạt được tần số quét 10hz. Người dùng có thể tùy chỉnh tần số quét từ 2hz đến 10hz tùy ý bằng cách điều khiển tốc độ của động cơ quét. Trình điều khiển sẽ đọc tín hiệu từ LIDAR sau đó xử lí và publish dữ liệu nhận được lên topic sensor_msgs /Scan.

Hình 4.18: Dữ liệu nhận được từ Lidar

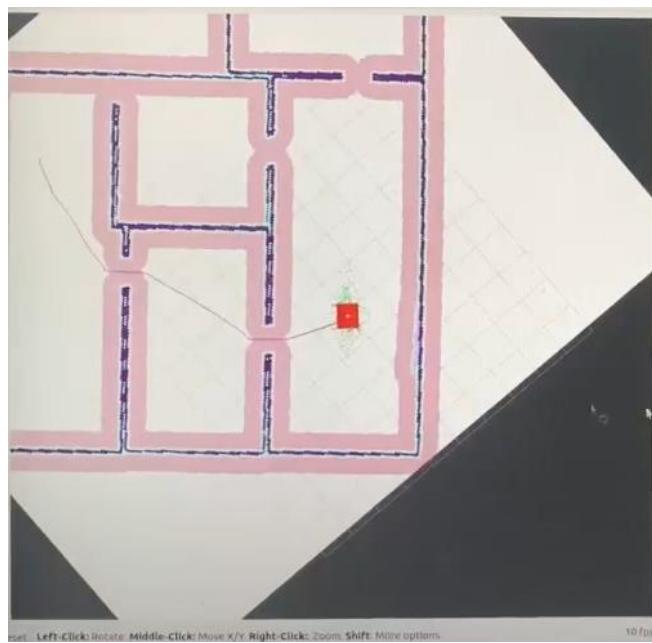
Dữ liệu trả về dưới dạng khoảng cách và được hiển thị qua topic scan như hình dưới đây. Các chấm đỏ chính là vị trí các vật cản mà lidar quét được, nếu không gặp vật cản, khoảng cách sẽ là inf.



Hình 4.19: Rviz mô phỏng lại dữ liệu nhận được từ scan

b. Thiết lập bản đồ

Ta có bản đồ dựa trên thuật toán Slam gmapping như hình 4.20



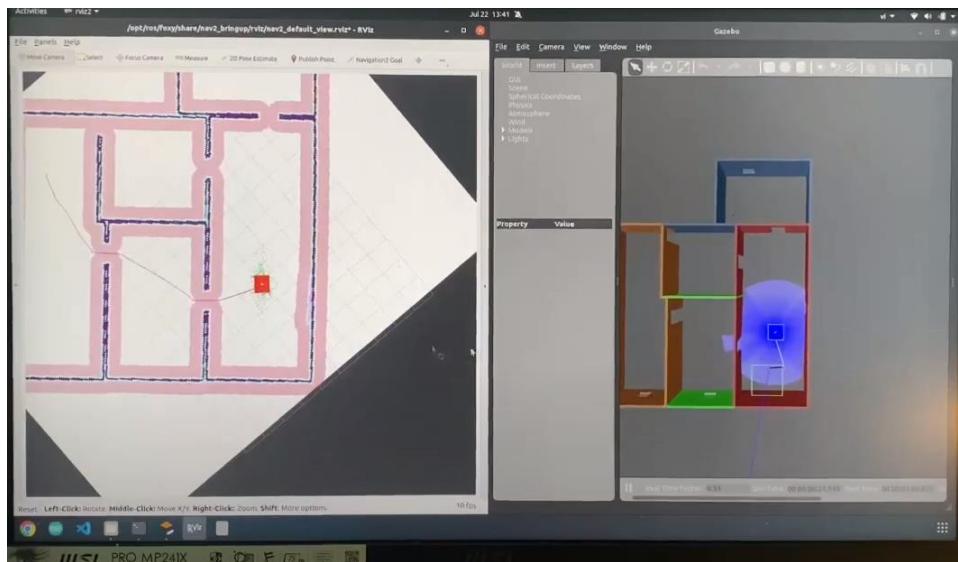
Hình 4.20: Map xây dựng được từ Slam gmapping

4.4.2. Cấu hình Navigation

Gói Navigation cần được cấu hình đầy đủ để có thực hiện nhiệm vụ điều hướng tự động cho robot. Hình sau mô tả cấu hình gói Navigation được mô tả trên trang chủ của ROS.

Trước hết, ta cần cấu hình costmap. Costmap có thể có ba lớp: lớp tĩnh, lớp vật cản và lớp lạm phát. Thường thì global costmap sử dụng cả ba lớp, còn local costmap chỉ sử dụng hai lớp là lớp vật cản và lớp lạm phát. Lớp tĩnh chính là bản đồ thu được từ Slam gmapping, được biểu diễn ở hình 4.20. Ngoài ra, đối với global costmap được biểu diễn ở hình 4.20, cần xác định hệ tọa độ của nó là "map". Tương tự đối với local costmap được biểu diễn ở hình 4.21, hệ tọa độ ở đây là "odom". Tần số cập nhật cho mỗi bản đồ được lựa chọn là 1 Hz.

Sau đó, bộ lập kế hoạch đường đi được cấu hình như sau: Bộ lập kế hoạch đường đi toàn cục được lựa chọn là Djisktra. Trong bộ lập kế hoạch đường đi cục bộ, cần thiết lập thông số cho vận tốc quay lớn nhất và vận tốc tịnh tiến lớn nhất để phù hợp với yêu cầu bài toán đặt ra. Hình 4.21 trực quan hóa kết quả của bộ lập kế hoạch đường đi bằng công cụ RVIZ.



Hình 4.21: Kế hoạch đường đi với map có sǎn

4.5. Xây dựng giao diện người dùng cho robot

4.5.1. Ý tưởng về giao diện

Giao diện người dùng được hiển thị trên màn hình của robot, chức năng của giao diện là giúp người dùng chọn địa điểm giao hàng và các thông tin của người nhận hàng,...

Trong đồ án này, nhóm đã xây dựng giao diện dựa trên PyQt5 và QT Designer. PyQt5 cung cấp giao diện đồ họa chuyên nghiệp và đa dạng các thành phần giao diện để tạo ra giao diện người dùng đẹp và chuyên nghiệp. Qt Designer cho phép thiết kế giao diện người dùng đồ họa một cách trực quan bằng cách kéo và thả các thành phần giao diện. Nó tích hợp dễ dàng với PyQt5, cho phép tạo tệp UI để sử dụng trong mã PyQt5. Qt Designer cũng cho phép tùy chỉnh giao diện bằng cách thay đổi thuộc tính và kiểu dáng của các thành phần. Bạn có thể tái sử dụng các thành phần tùy chỉnh và quản lý chúng trong thư viện để dễ dàng duy trì và sử dụng lại trong các dự án khác. Qt Designer cũng hỗ trợ đa ngôn ngữ, giúp bạn tạo các phiên bản ứng dụng đa ngôn ngữ.



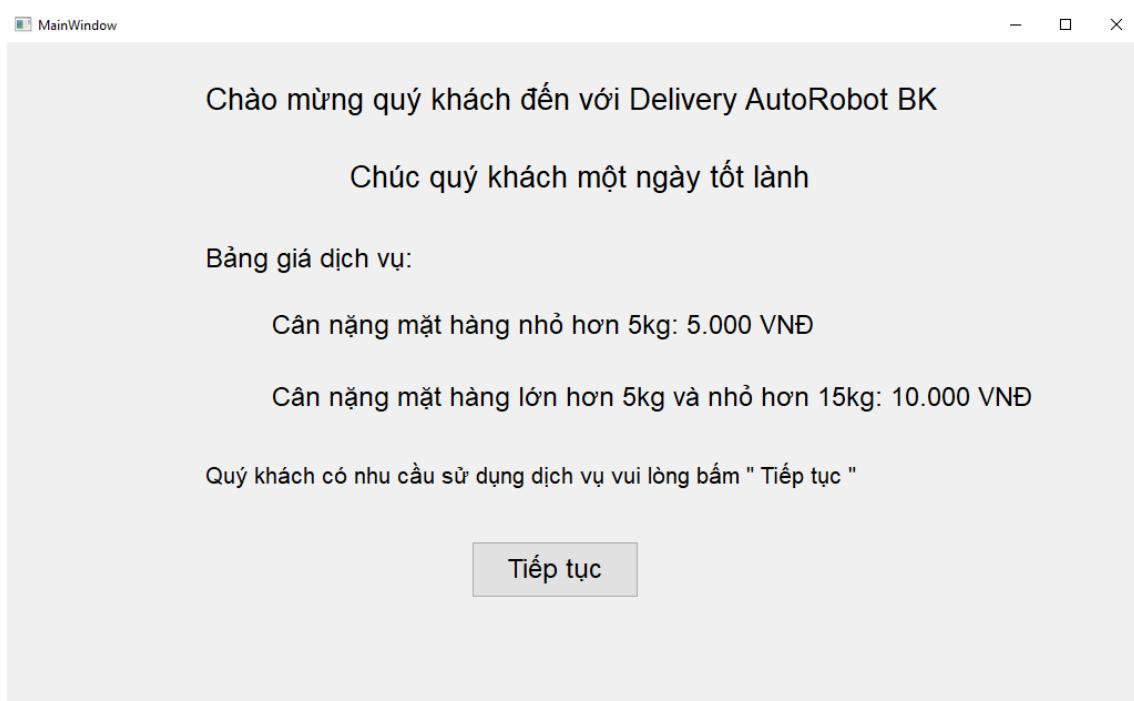
Hình 4..22: Vị trí đặt màn hình trên robot

4.5.2. Thiết kế giao diện

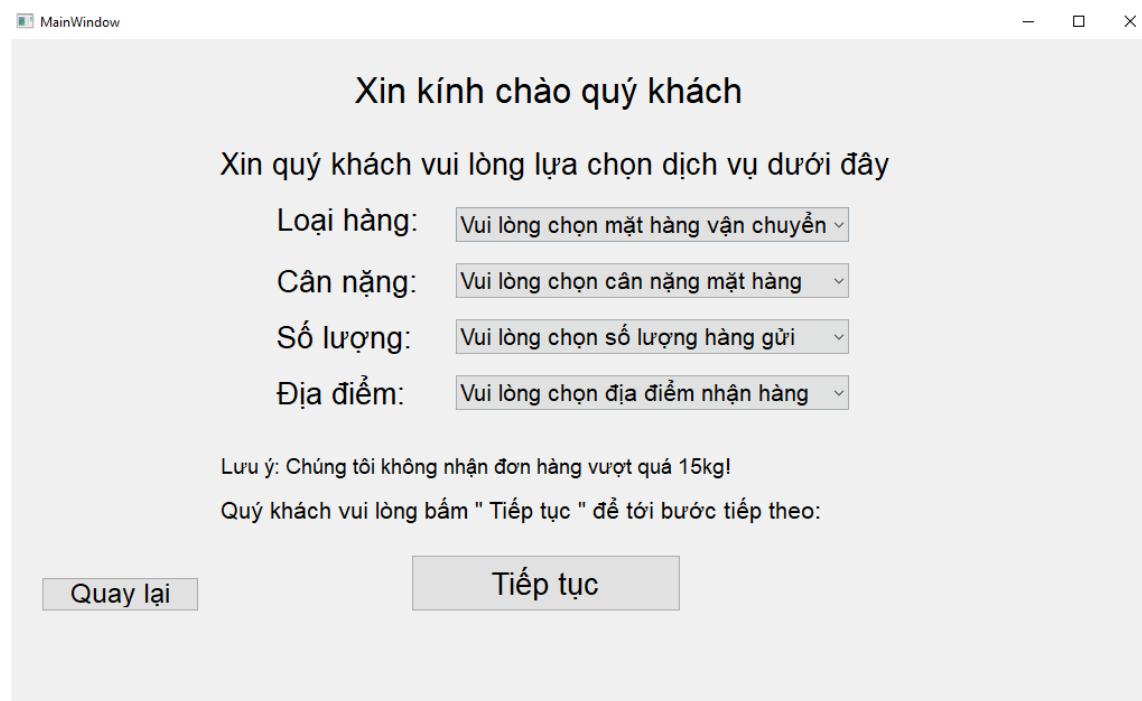
Nhóm đồ án sẽ thiết kế giao diện theo các bước. Đầu tiên là xây dựng bối cảnh, hình ảnh cho giao diện. Tiếp theo là xây dựng tương tác giữa người dùng với giao diện. Cuối cùng là xây dựng tương tác giữa giao diện với Robot.

a. Bối cảnh, hình ảnh

Để tương tác với khách hàng một cách có tuần tự và hiệu quả. Nhóm đồ án thiết kế giao diện có 4 Page. Trong đó cửa sổ chính được xây dựng bằng lớp QWidget. Trong lớp QWidget em xây dựng thêm lớp con QStackWidget để thiết kế trên đó. Dưới đây là lần lượt các Page mà nhóm đồ án thiết kế.



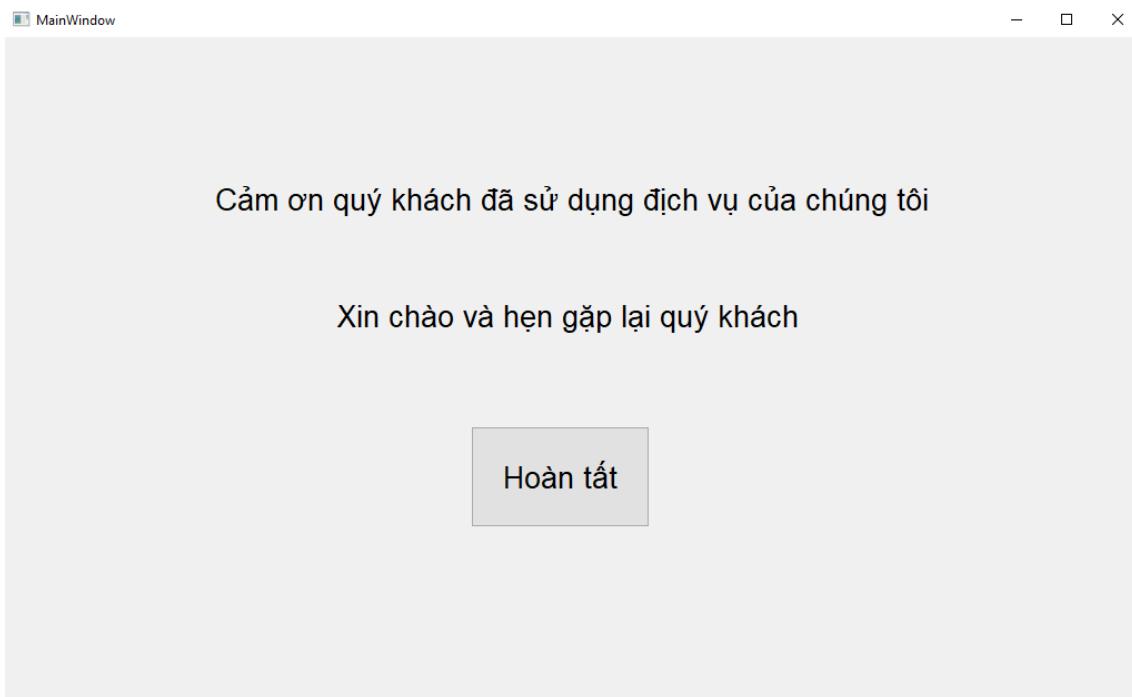
Hình 4.22: Màn hình bắt đầu



Hình 4.23: Màn hình lựa chọn dịch vụ



Hình 4.24: Màn hình thanh toán



Hình 4.25: Màn hình xác nhận dịch vụ

b. Tương tác với người dùng

Sau khi đã có file `Ui_MainWindow.py`, nhóm đồ án tạo thêm một file `MainWindow.py` để lập trình tương tác giữa các Button với giao diện.

Đầu tiên nhóm đồ án import những thư viện cần thiết để tương tác giữa PyQt5 và Qt Designer cùng với đó là file `Ui_MainWindow.py` để tiến hành lập trình trên giao diện đã thiết kế.

```

1  import sys
2  import rospy
3  from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QComboBox
4  from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
5  import actionlib
6  from PyQt5.QtWidgets import QApplication
7  from PyQt5.QtWidgets import QMainWindow
8  from Ui_MainWindow import Ui_MainWindow

```

Tạo một class `MainWindow` và hàm khởi tạo `__init__()` để tạo một cửa sổ chính cho ứng dụng và truyền vào đó `Ui_MainWindow`.

```

10  class MainWindow:
11      def __init__(self):
12          self.main_win = QMainWindow()
13          self.ui = Ui_MainWindow()
14          self.ui.setupUi(self.main_win)

```

Sau đó sử dụng các câu lệnh trong ảnh dưới để thiết lập sự kết nối giữa sự kiện "click" trên một nút (button) và một phương thức xử lý sự kiện tương ứng trong lớp `MainWindow`.

```

25     self.ui.nut_tieptuc.clicked.connect(self.showpage_chondichvu)
26     self.ui.nut_tieptuc2.clicked.connect(self.showpage_thanhtoan)
27     self.ui.nut_xacnhanthanhtoan.clicked.connect(self.showpage_camon)
28     self.ui.nut_hoantat.clicked.connect(self.showpage_chaomung)
29     self.ui.nut_quaylai1.clicked.connect(self.showpage_chaomung)
30     self.ui.nut_quaylai1_2.clicked.connect(self.showpage_chondichvu)

```

Ví dụ, khi chúng ta click chuột vào button có tên là nut_hoantat thì sẽ thực hiện chuyển từ trang hiện tại sang một trang khác có tên là page_chaomung. Cùng với đó là đặt lại các giá trị của các combo box về giá trị mặc định.

Thiết lập các câu lệnh và các phương thức để có thể di chuyển qua lại giữa các Page và chức năng thanh toán.

c. Tương tác với ROS

Các thư viện cần thiết cho việc thiết lập giao diện người dùng của nhóm đồ án thành một Node trong Ros là Rospy, actionlib và move_base_msgs.msg.

Chương trình này tương tác với robot xây dựng trên ROS bằng cách sử dụng giao thức ROS (Robot Operating System) để gửi các tín hiệu và nhận thông tin từ robot.

Cụ thể, chương trình sử dụng gói move_base_msgs để tạo ra các tin nhắn di chuyển (MoveBaseAction) và mục tiêu di chuyển (MoveBaseGoal) trong ROS. Điều này cho phép chương trình gửi yêu cầu di chuyển tới robot mô phỏng thông qua actionlib.SimpleActionClient.

Phương thức def movebase_client() được nhóm đồ án viết để tương tác với Ros về vị trí và các yêu cầu di chuyển.

```

43     def movebase_client(self):
44         # kiem tra neu flag hoantat_press la True thi moi chay chuong trinh
45         if not self.hoantat_press:
46             return
47         vitri = self.ui.box_diemnhanhang.currentText()
48
49         client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
50         client.wait_for_server()
51
52         goal = MoveBaseGoal()
53         goal.target_pose.header.frame_id = "map"
54         goal.target_pose.header.stamp = rospy.Time.now()
55
56         if vitri == "Khu vực 1":
57             goal.target_pose.pose.position.x = -2.554
58             goal.target_pose.pose.position.y = -1.771
59             goal.target_pose.pose.orientation.z = -0.3
60             goal.target_pose.pose.orientation.w = 0.701
61         elif vitri == "Khu vực 2":
62             goal.target_pose.pose.position.x = 1.5
63             goal.target_pose.pose.position.y = -1.5
64             goal.target_pose.pose.orientation.z = -0.05
65             goal.target_pose.pose.orientation.w = 0.05
66         elif vitri == "Khu vực 3":
67             goal.target_pose.pose.position.x = 2.752
68             goal.target_pose.pose.position.y = -1.538
69             goal.target_pose.pose.orientation.z = 0.016
70             goal.target_pose.pose.orientation.w = 1
71         elif vitri == "Khu vực 4":
72             goal.target_pose.pose.position.x = 0.0
73             goal.target_pose.pose.position.y = 0.0
74             goal.target_pose.pose.orientation.z = 0.0
75             goal.target_pose.pose.orientation.w = 0.01
76         else:
77             return
78
79         client.send_goal(goal)

```

Đầu tiên nhóm đồ án tạo một client actionlib để gửi các mục tiêu đến move_base:

client = actionlib.SimpleActionClient('move_base', MoveBaseAction): Đây là cách tạo một đối tượng client actionlib với tên là "move_base" và kiểu là MoveBaseAction. Đối tượng client này sẽ được sử dụng để gửi mục tiêu đến move_base server.

client.wait_for_server(): Dòng này đảm bảo rằng client đã kết nối thành công với move_base server trước khi tiếp tục thực hiện các thao tác khác. Nó đợi đến khi move_base server sẵn sàng trước khi tiếp tục thực hiện mã.

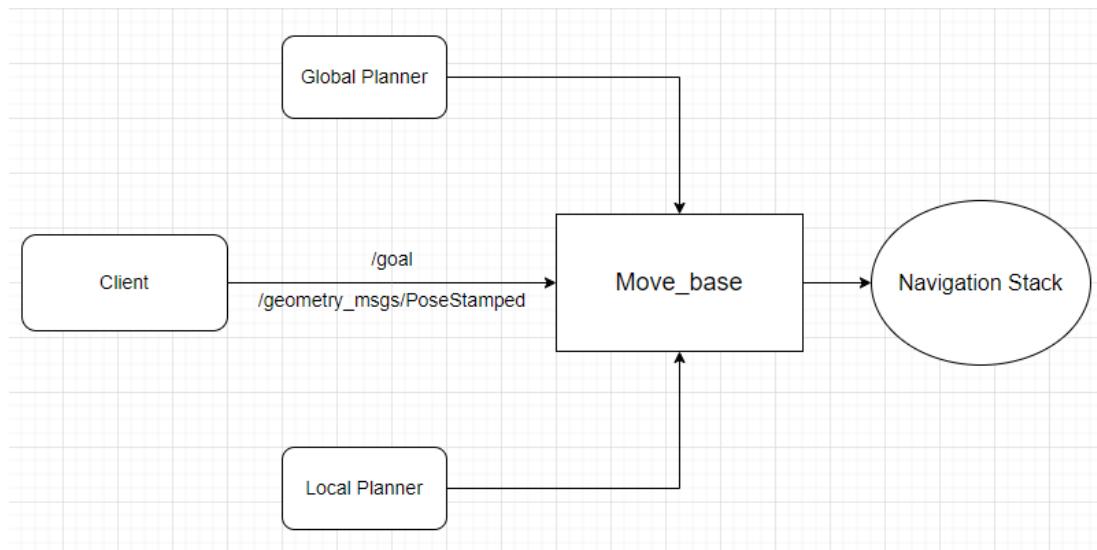
goal = MoveBaseGoal(): Đây là cách tạo một đối tượng MoveBaseGoal, đại diện cho một mục tiêu di chuyển (move goal) trong move_base. Đối tượng này chứa thông tin về vị trí và hướng di chuyển mà robot cần đạt được.

goal.target_pose.header.frame_id = "map": Dòng này đặt giá trị của trường frame_id trong goal.target_pose.header là "map". Đây là frame of reference mà tọa

độ mục tiêu được định rõ trong. Map có được qua SLAM hoặc các thiết bị như GPS.

`goal.target_pose.header.stamp = rospy.Time.now()`: Dòng này đặt thời gian của `goal.target_pose.header` là thời điểm hiện tại, sử dụng `rospy.Time.now()`. Thời gian này chỉ ra thời điểm mà mục tiêu di chuyển được gửi đi.

Dòng lệnh `client.send_goal(goal)` trong đoạn mã trên được sử dụng để gửi mục tiêu (`goal`) cho `move_base` server thông qua client `actionlib`. Khi gọi `send_goal(goal)`, client sẽ gửi mục tiêu đến `move_base` server để yêu cầu robot di chuyển tới vị trí và hướng di chuyển được chỉ định trong `goal`. Mục tiêu này sẽ được gửi đi thông qua kênh giao tiếp đã thiết lập giữa client và server.



Hình 4.26: Sơ đồ mô tả

Thiết lập biến vị trí theo giá trị hiện tại trong Combo box `box_diemnhanhang`. Với mỗi giá trị vị trí khác nhau sẽ truyền vào một goal khác nhau.

Điều kiện để robot di chuyển là đã được chọn vị trí kết thúc và button `nut_hoantat` đã được nhấn để tránh trường hợp vừa chọn biến vị trí xong thì xe đã di chuyển. Vì vậy nhóm đồ án tạo ra 2 biến cờ `self.is_location_selected` và `self.hoantat_press` để xác định điều kiện chạy cho phương thức `def movebase_client()`, cả hai cờ đều phải có giá trị True thì phương thức mới được chạy.

Cuối cùng, trong hàm `if __name__ == '__main__'`, nhóm đồ án thêm câu lệnh `rospy.init_node('giaodien_py')` được sử dụng để khởi tạo một nút trong mạng ROS với tên là 'giaodien_py', cho phép chương trình tương tác và giao tiếp với các thành phần khác trong mạng ROS.

Tổng kết lại, giao diện người dùng mà nhóm đồ án xây dựng có các chức năng tương tác trực tiếp với khách hàng là:

- Di chuyển giữa các Page bằng các nút bấm.
- Các box lựa chọn tương tác với khách hàng: chọn mặt hàng, chọn cân nặng hàng, chọn số lượng gói hàng và chọn địa điểm nhận hàng.
- Chức năng tính giá tiền và hỗ trợ thanh toán qua QR Code.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

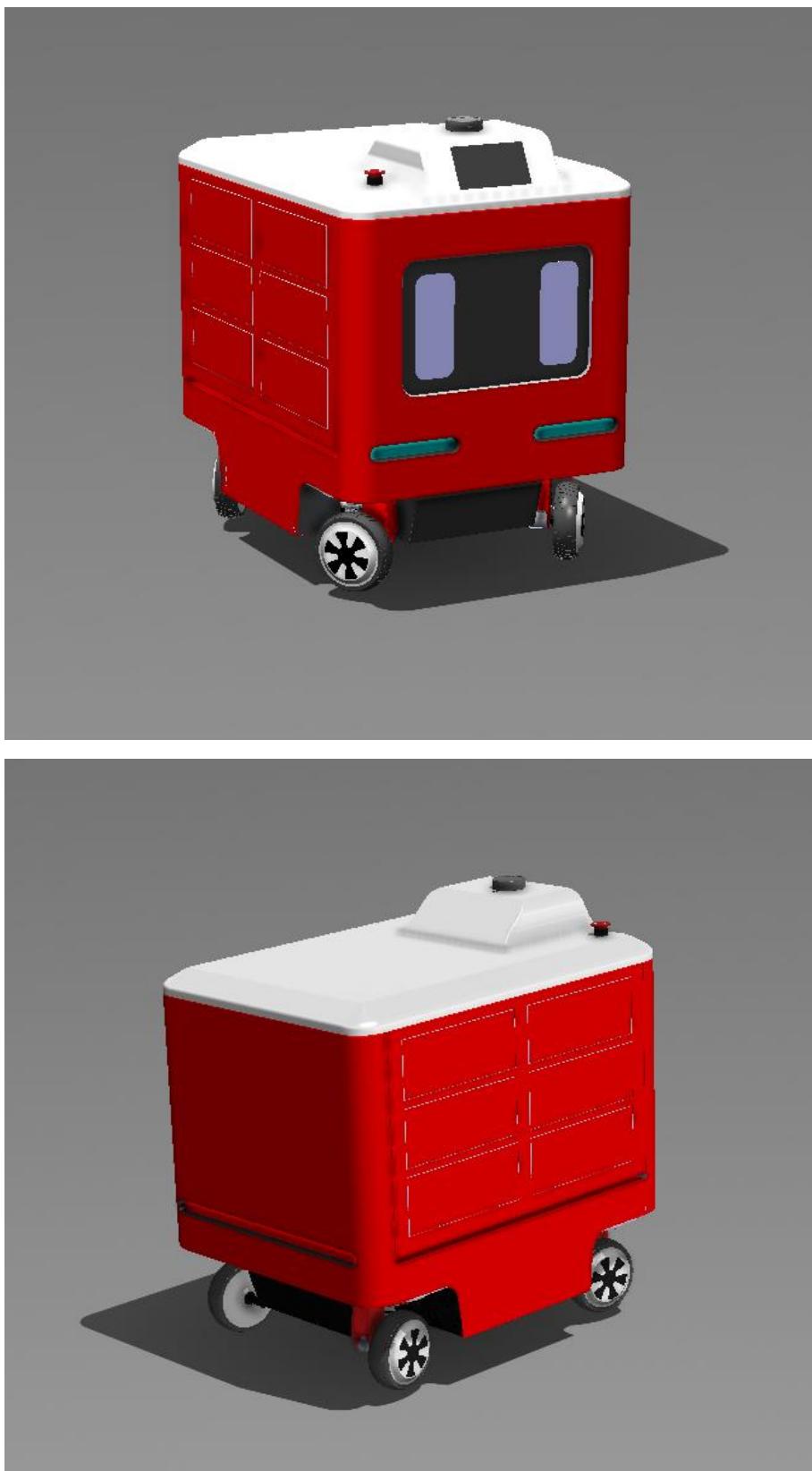
Với đề tài “Thiết kế và điều khiển robot tự hành ứng dụng trong dịch vụ giao hàng”, nhóm đã đạt được những kết quả nhất định:

- Ứng dụng hệ thống lái swerve drive 4 bánh lái độc lập trong mô hình mobile robot.
- Nghiên cứu, thiết kế và chế tạo các thành phần của module swerve drive.
- Nắm được nguyên lý của hệ thống lái độc lập 4 bánh.
- Ứng dụng framework ROS trong việc điều khiển tự động robot, áp dụng được những thuật toán tìm đường cũng như định vị trong mobile robot.
- Tích hợp được cách hệ thống phần cứng và cảm biến trong mô hình robot.

5.2. Hướng phát triển

Đề tài của nhóm ứng dụng được hệ thống dẫn động khá mới trong mobile robot, với những nghiên cứu sau nhóm hy vọng có thể đạt được:

- Cải thiện độ chính xác, cũng như độ ổn định của hệ thống dẫn động mới.
- Tối ưu hóa các thành phần của hệ thống điều khiển.
- Tối ưu được các thuật toán tìm đường để phù hợp hơn với cấu hình của mô hình robot.
- Thiết kế thêm hệ thống thùng hàng và màn hình thao tác.
- Tích hợp thêm các cảm biến để phục vụ mục đích an toàn khi di chuyển.



Hình 5.1: Mô hình concept của robot giao hàng

TÀI LIỆU THAM KHẢO

- [1] B. Shamah, "Experimental Comparison of Skid Steering", 1999.
- [2] R. M. F. R. A. a. M. N. Khan, ""Comprehensive study of skid-steer wheeled mobile robots: development and challenges," in *Industrial Robot*, 2021, pp. Vol. 48 No. 1, pp. 142-156..
- [3] C. Savant, Design of Driveline for Mobile Robot, KTH Industrial Engineering and Management.
- [4] "wikipedia.org," [Online]. Available:
https://en.wikipedia.org/wiki/Contact_mechanics.
- [5] MISUMI, "1st Edition_PulleyCatalog," pp. 124-150.
- [6] L. Bies, "RS485, specifications and in-depth tutorial," 2021. [Online]. Available: <https://www.lammertbies.nl/comm/info/rs-485> .
- [7] C. Delphi, "Derivation of Inverse Kinematics for Swerve," 2011. [Online]. Available:
<https://www.chiefdelphi.com/uploads/default/original/3X/8/c/8c0451987d09519712780ce18ce6755c21a0acc0.pdf..>
- [8] EFeru, "hoverboard-firmware-hack-FOC," 2019. [Online]. Available:
<https://github.com/EFeru/hoverboard-firmware-hack-FOC> .
- [9] D. Goodwin, "Field-oriented Control (Vector Control) for Brushless DC Motors," 2023. [Online]. Available: <https://control.com/technical-articles/field-oriented-control-vector-control-for-brushless-dc-motors>.
- [10] "wikipedia.org," [Online]. Available:
https://en.wikipedia.org/wiki/PID_controller.
- [11] "discuss.saleae.com," [Online]. Available:
<https://discuss.saleae.com/t/quadrature-encoder-analyser/877>.
- [12] "leinelinde.com," [Online]. Available:
<https://www.leinelinde.com/support/tech-corner/encoder-technology/how>-

incremental-encoders-work/.

- [13] Yunxiang Ye, Long He, Qin Zhang, "Steering Control Strategies for a Four-Wheel-Independent-Steering Bin Managing Robot," 2016. [Online]. Available: <https://doi.org/10.1016/j.ifacol.2016.10.008..>

Part No.	Description
1	Front wheel hub
2	Front wheel hub
3	Front wheel hub
4	Front wheel hub
5	Front wheel hub
6	Front wheel hub
7	Front wheel hub
8	Front wheel hub
9	Front wheel hub
10	Front wheel hub
11	Front wheel hub
12	Front wheel hub
13	Front wheel hub

