

BỘ GIÁO DỤC VÀ ĐÀO TẠO

BỘ NÔNG NGHIỆP VÀ PTNT

TRƯỜNG ĐẠI HỌC THỦY LỢI



LÊ THÀNH SƠN

**THIẾT KẾ, XÂY DỰNG VÀ LẬP TRÌNH HỆ THỐNG ĐIỀU
KHIỂN ROBOT PHÁT THUỐC TRONG BỆNH VIỆN SỬ DỤNG
LIDAR DẪN HƯỚNG**

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

BỘ NÔNG NGHIỆP VÀ PTNT

TRƯỜNG ĐẠI HỌC THỦY LỢI

LÊ THÀNH SƠN

**THIẾT KẾ, XÂY DỰNG VÀ LẬP TRÌNH HỆ THỐNG ĐIỀU
KHIỂN ROBOT PHÁT THUỐC TRONG BỆNH VIỆN SỬ DỤNG
LIDAR DẪN HƯỚNG**

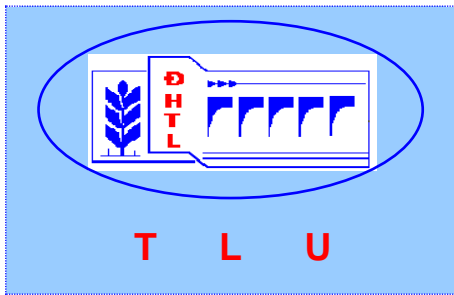
Ngành : Cơ Điện Tử

Mã số : TLA120

NGƯỜI HƯỚNG DẪN: 1. TH.S NGUYỄN TIẾN THỊNH

2. TH.S NGUYỄN QUANG HUY

HÀ NỘI, NĂM 2024



CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc



NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: LÊ THÀNH SON

Lớp: 61CĐTĐNB

Khoa: CƠ KHÍ

Hệ đào tạo : CHÍNH QUY

Ngành: CƠ ĐIỆN TỬ

1 - TÊN ĐỀ TÀI: THIẾT KẾ, XÂY DỰNG VÀ LẬP TRÌNH HỆ THỐNG ĐIỀU KHIỂN
ROBOT PHÁT THUỐC TRONG BỆNH VIỆN SỬ DỤNG LIDAR DẪN HƯỚNG

2 - CÁC TÀI LIỆU CƠ BẢN:

[1] Effective Robotics Programming with ROS - Third Edition của Anil Mahtani, Luis Sanchez, Enrique Fernandez, và Aaron Martinez

[2] PID Controllers: Theory, Design, and Tuning của Karl Johan Åström, Tore Hägglund

[3] Learning ROS for Robotics Programming 2015 của Enrique Fernández, Luis Sánchez Crespo, Anil Mahtani, Aaron Martinez

[4] Mastering ROS for Robotics Programming của Lentin Joseph

[5] PID Control in the Third Millennium: Lessons Learned and New Approaches của Antonio Visioli và Qing-Guo Wang

[6] Programming Robots with ROS: A Practical Introduction to the Robot Operating System của Morgan Quigley, Brian Gerkey, và William D. Smart

3 - NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN LẬP TRÌNH

Nội dung Thuyết minh và tính toán lập trình	Tỉ lệ
Chương 1. Giới thiệu về robot	20%
Chương 2. Mô hình hóa	
Chương 3. Lựa chọn thiết điều khiển	
Chương 4. Hệ thống điện và thuật toán lập trình điều khiển	
Chương 5. Kết luận và hướng phát triển đề tài	

4 - BẢN VẼ VÀ BIỂU ĐỒ

Bản vẽ sơ đồ nguyên lý hệ thống điều khiển – A3

Lưu đồ thuật toán hệ thống điều khiển – A3

Sơ đồ khối hệ thống điều khiển – A3

Bản tổng thể – A3

5. GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN

Phần	Họ tên giáo viên hướng dẫn
Chương 1, chương 3 và chương 5	Th.s Nguyễn Tiến Thịnh
Chương 2 và chương 4	Th.s Nguyễn Quang Huy

6. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày tháng năm 20

Trưởng Bộ môn
(Ký và ghi rõ Họ tên)

Giáo viên hướng dẫn chính
(Ký và ghi rõ Họ tên)

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày.tháng.năm 20..

Chủ tịch Hội đồng

(Ký và ghi rõ Họ tên)

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày... tháng... năm 20...

Sinh viên làm Đồ án tốt nghiệp

(Ký và ghi rõ Họ tên)

LỜI CAM ĐOAN

Tôi xin cam đoan đây là Đồ án tốt nghiệp của bản thân tôi. Các kết quả trong Đồ án tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào. Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

Tác giả ĐATN

Lê Thành Sơn

MỤC LỤC

LỜI CAM ĐOAN	i
MỤC LỤC	ii
DANH MỤC HÌNH ẢNH.....	iv
DANH MỤC BẢNG BIỂU	vii
DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ'	ix
TÓM TẮT CHƯƠNG	x
CHƯƠNG 1. GIỚI THIỆU VỀ ROBOT.....	1
1.1 Tổng quan về Robot.....	1
1.2 Các dạng Robot tự hành.....	7
1.3 Tổng quan về Robot AMR.....	11
1.4 Đặt vấn đề	13
1.5 Mục tiêu nghiên cứu của đề án	14
1.6 Phạm vi và phương pháp nghiên cứu về Robot phát thuốc trong bệnh viện	14
CHƯƠNG 2. MÔ HÌNH HÓA	16
2.1 Mô hình hóa	16
2.2 Kiến trúc chi tiết hệ thống.....	20
2.3 Phần mềm hỗ trợ và phần mềm chính xây dựng Robot.....	23
2.4 Hệ điều hành cho ROS.....	28

2.5	Truyền thông từ máy tính đến Robot	31
2.6	Quy trình tạo và hiển thị bản đồ.....	35
2.7	Kết luận chung	45
CHƯƠNG 3. LỰA CHỌN THIẾT BỊ ĐIỀU KHIỂN		46
3.1	Trung tâm xử lý.....	46
3.2	Khối xây dựng map.....	56
3.3	Cơ cấu chấp hành khối chuyển động:	59
3.4	Khối năng lượng	63
3.5	Sơ đồ chi tiết thiết bị trong hệ thống.....	69
CHƯƠNG 4. HỆ THỐNG ĐIỆN VÀ THUẬT TOÁN LẬP TRÌNH ĐIỀU KHIỂN		70
4.1	Hệ thống điện	70
4.2	Sơ đồ nguyên lí	71
4.3	Xây dựng hệ thống Robot	73
4.4	Mã nguồn cho Arduino	78
4.5	Kết quả hoạt động Robot	84
KẾT LUẬN		86
Tài liệu tham khảo		90
Phụ lục 1		91
Phụ lục 2		99

DANH MỤC HÌNH ẢNH

Hình 1.1 Rô bốt trí tuệ nhân tạo tại một sân bay Nhật Bản.....	1
Hình 1.2 Robot Unimate, Robot công nghiệp đầu tiên trong nhà xưởng.....	2
Hình 1.3 Robot chăm sóc bệnh nhân tại Italy	3
Hình 1.4 Robit Moxi.....	5
Hình 1.5 Robot TUG thay y tá vận chuyển vật dụng	5
Hình 1.6 ZenZoe đang loại bỏ Covid-19 trong một bệnh viện	6
Hình 1.7 Đưa Robot y tế hỗ trợ khu cách ly Covid-19 tại Hà Nam.....	6
Hình 1.8 SBC MIO-5373 của Advantech.....	7
Hình 1.9 Robot AGV VTPost.....	8
Hình 1.10 Máy bay không người lái (UAV).....	8
Hình 1.11 THEMIS của Đức	9
Hình 1.12 A9-M / AUV thăm dò đại dương.....	9
Hình 1.13 Một Robot Starship Technologies giao một bưu kiện Starship Technologies..	10
Hình 2.1 Mô hình hóa hệ thống.....	16
Hình 2.2 Sơ đồ chi tiết hệ thống của Robot.....	20
Hình 2.3 Mô tả kiến trúc chi tiết hệ thống.....	23
Hình 2.4 Logo ROS Noetic một phiên bản của ROS	24
Hình 2.5 Giao diện khi dùng lệnh ping google.com thành công.....	31
Hình 2.6 Giao diện khi dùng lệnh ping google.com không thành công.....	31

Hình 2.7 Màn hình giao diện Ubuntu Mate.....	33
Hình 2.8 Xây dựng bản đồ bằng SLAM trên Rviz.....	36
Hình 2.9 Xây dựng bản đồ theo bản vẽ kiến trúc.....	36
Hình 3.1 Raspberry Pi 4 model B 4G.....	47
Hình 3.2 Sơ đồ chân Raspberry Pi 4	48
Hình 3.3 Sơ đồ chân Arduino Uno R3 ATmega328	50
Hình 3.4 Module DC BTS7960 43A.....	52
Hình 3.5 MODULE Relay 12VDC 4 Kênh.....	54
Hình 3.6 Sơ đồ nguyên lí hoạt động.....	56
Hình 3.7 RPLIDAR A1M8 360.....	57
Hình 3.8 Cảm biến siêu âm HC-SR04.....	59
Hình 3.9 DC Servo JGB37-545 (12V/110RPM).....	60
Hình 3.10 AD16-22DS 12VDC/AC	62
Hình 3.11 Mạch Hạ Áp DC XL4015.....	63
Hình 3.12 Mạch tăng áp 12-220V 150W	65
Hình 3.13 Pin Li-ion.....	67
Hình 3.14 Sơ đồ chi tiết thiết bị trong hệ thống	69
Hình 4.1 Lưu đồ thuật toán hệ thống Robot tạo bản đồ - tiếng việt.....	74
Hình 4.2 Lưu đồ thuật toán hệ thống Robot tạo bản đồ - thuật ngữ chuyên ngành	74
Hình 4.3 Lưu đồ thuật toán điều khiển Robot trên vi điều khiển.....	78

Hình 4.4 Sơ đồ khối của bộ điều khiển PID	80
Hình 4.5 Lần chạy thử nghiệm lần 2	84

DANH MỤC BẢNG BIỂU

Bảng 2.1 So sánh phần mềm xây dựng Robot.....	23
Bảng 2.2 So sánh ưu nhược điểm của hệ điều hành.....	28
Bảng 3.1 So sánh các bo mạch nhúng	46
Bảng 3.2 Thông số kỹ thuật của Raspberry Pi 4 Model B	48
Bảng 3.3 So sánh các loại vi điều khiển.....	49
Bảng 3.4 Thông số kỹ thuật của Arduino Uno R3 ATmega328	51
Bảng 3.5 So sánh Drive điều khiển động cơ	51
Bảng 3.6 Thông số kỹ thuật của DC BTS7960 43A	53
Bảng 3.7 Thông số kỹ thuật module relay 12vdc bốn kênh	55
Bảng 3.8 So sánh các loại LIDAR.....	56
Bảng 3.9 Thông số kỹ thuật của LIDAR RPLIDAR A1M8.....	58
Bảng 3.10 Thông số kỹ thuậtDC Servo JGB37-545	61
Bảng 3.11 Thông số kỹ thuật mạch hạ áp DC XL4015.....	64
Bảng 3.12 Thông số kỹ thuật mạch tăng áp 12-220V 150W	66
Bảng 3.13 So sánh nguồn năng lượng	67
Bảng 4.1 Tính toán dự tính lượng điện cho toàn bộ hệ thống	70
Bảng 4.2 Các mức nguồn năng lượng cho hệ thống	71
Bảng 4.3 Sơ đồ chân kết nối thiết bị ngoại vi với Raspberry Pi 4	71
Bảng 4.4 Sơ đồ chân kết nối vi điều khiển với cảm biến	71

Bảng 4.5 Sơ đồ chân kết nối vi điều khiển với mạch điều khiển	72
Bảng 4.6 Sơ đồ chân kết nối vi điều khiển với relay.....	72
Bảng 4.7 Sơ đồ chân kết nối relay với LED.....	72
Bảng 4.8 Sơ đồ chân kết nối mạch điều khiển với động cơ	72
Bảng 4.9 Sơ đồ chân kết nối RPLIDAR A1M8 với mạch chuyển đổi micro – USB.....	73
Bảng 4.10 Sơ đồ chân kết nối cảm biến với vi điều khiển	73
Bảng 4.11 Xây dựng hệ thống	73

DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ

ROS: Robot Operating System

OROCOS: Open Robot Control Software

MOOS: Mission Oriented Operating Suite

AMR (Autonomous Mobile Robot)

AGV (Automated Guided Vehicle)

UAV: Unmanned Aerial Vehicle

UGV: Unmanned Ground Vehicle

AUV: Autonomous Underwater Vehicle

SLAM: Simultaneous Localization and Mapping

LIDAR: Light Detection and Ranging

HT: Hệ Thống

AMCL: Adaptive Monte Carlo Localization

COM (chung), NO (thường mở) và NC (thường đóng)

TÓM TẮT CHƯƠNG

CHƯƠNG 1. GIỚI THIỆU VỀ ROBOT

CHƯƠNG 2. MÔ HÌNH HÓA

CHƯƠNG 3. LỰA CHỌN THIẾT ĐIỀU KHIỂN

CHƯƠNG 4. HỆ THỐNG ĐIỆN VÀ THUẬT TOÁN LẬP TRÌNH ĐIỀU KHIỂN

CHƯƠNG 1. GIỚI THIỆU VỀ ROBOT

1.1 Tổng quan về Robot

1.1.1 *Khái niệm về Robot*

Robot hoặc Rôbốt, Rô-bô (tiếng Anh: Robot) là một loại máy có thể thực hiện những công việc một cách tự động bằng sự điều khiển của máy tính hoặc các vi mạch điện tử được lập trình. Robot là một tác nhân cơ khí, nhân tạo, thường là một hệ thống cơ khí-điện tử.



Hình 1.1 Rô bốt trí tuệ nhân tạo tại một sân bay Nhật Bản

Từ ngữ "Robot" thường được hiểu với hai nghĩa: Robot cơ khí và phần mềm tự hoạt động. Do sự đa dạng mức độ tự động của hệ thống cơ điện tử mà ranh giới phân chia Robot với phần còn lại không được rõ ràng, thể hiện ở quan niệm về định nghĩa Robot.

1.1.2 Lịch sử phát triển của Robot

1.1.2.1 Lịch sử phát triển của Robot



Hình 1.2 Robot Unimate, Robot công nghiệp đầu tiên trong nhà xưởng

Cỗ máy tự động cổ đại và trung cổ (Trước năm 1700): Khái niệm về Robot có thể truy nguyên từ thời cổ đại. Người Hy Lạp, ví dụ, có những truyền thuyết về những người hầu cơ học do các vị thần tạo ra. Trong thời kỳ trung cổ, có nhiều loại cỗ máy tự động cơ học tại châu Âu và những thiết bị cơ học ở thế giới Hồi giáo, được sử dụng cho mục đích giải trí hoặc thực tế.

Cách mạng Công nghiệp (Thế kỷ 18-19): Cách mạng Công nghiệp mang lại những tiến bộ đáng kể trong máy móc và tự động hóa. Giai đoạn này chứng kiến sự phát triển của các loại máy dệt cơ học và máy dệt Jacquard, sử dụng thẻ đục lỗ để kiểm soát họa tiết. Những máy móc này đã đặt nền móng cho sự phát triển sau này của các máy có thể lập trình.

Đầu thế kỷ 20: Thuật ngữ "Robot" được sử dụng lần đầu tiên vào năm 1920 trong vở kịch "R.U.R." (Rossum's Universal Robots) của Karel Čapek, mô tả một tương lai với những người nhân tạo. Khái niệm về Robot như chúng ta biết bắt đầu hình thành, chịu ảnh hưởng từ khoa học viễn tưởng và những tiến bộ công nghệ mới nổi.

Robot học giữa thế kỷ 20: Sự phát triển của những Robot lập trình đầu tiên bắt đầu vào thời kỳ này. Năm 1954, George Devol phát minh ra Robot đầu tiên được vận hành kỹ thuật số và có thể lập trình, Unimate, sau đó hoạt động trong dây chuyền lắp ráp của General Motors. Thuật ngữ "Robot học" được sử dụng lần đầu tiên bởi Isaac Asimov, một nhà văn khoa học viễn tưởng, người cũng đề xuất Ba Định luật về Robot.

Cuối thế kỷ 20: Giai đoạn này chứng kiến sự phát triển nhanh chóng trong lĩnh vực Robot học, được thúc đẩy bởi cuộc cách mạng số hóa. Robot trở nên ngày càng tinh vi, với các cảm biến, bộ điều khiển và AI được cải thiện.

Thế kỷ 21: Kỷ nguyên hiện tại của Robot học được đánh dấu bởi những tiến bộ chưa từng có trong AI và học máy. Robot đang trở nên tự động hóa hơn, linh hoạt hơn và hòa nhập sâu vào đời sống con người. Chúng được sử dụng trong các lĩnh vực đa dạng, từ phẫu thuật y tế đến khám phá vũ trụ, và tiếp tục mở rộng giới hạn của công nghệ Robot học.

1.1.2.2 Lịch sử phát triển của Robot y tế



Hình 1.3 Robot chăm sóc bệnh nhân tại Italy

Những thí nghiệm và khái niệm ban đầu (1950s-1970s): Ý tưởng sử dụng Robot trong y học bắt đầu hình thành vào giữa thế kỷ 20, chịu ảnh hưởng từ những tiến bộ trong lĩnh vực Robot và công nghệ máy tính. Các khái niệm và thí nghiệm ban đầu chủ yếu mang tính lý thuyết, khám phá cách hệ thống Robot có thể hỗ trợ trong các tình huống y tế.

Sự ra đời của phẫu thuật hỗ trợ bằng máy tính (1980s): Ứng dụng đáng kể đầu tiên của Robot trong y học xuất hiện với sự phát triển của phẫu thuật hỗ trợ bằng Robot. Vào năm 1985, cánh tay phẫu thuật Robot, PUMA 560, được sử dụng để thực hiện sinh thiết thần kinh với độ chính xác cao hơn. Điều này đánh dấu một bước ngoặt quan trọng, thể hiện tiềm năng của Robot trong việc tăng cường độ chính xác của phẫu thuật.

Hệ thống phẫu thuật Robot (1990s): Thập niên 1990 chứng kiến sự tiến bộ nhanh chóng với sự ra đời của các Robot phẫu thuật phức tạp hơn. Năm 1992, hệ thống ROBODOC được sử dụng cho các ca phẫu thuật thay khớp hông, mang lại độ chính xác cao hơn trong việc cắt xương. Sự phát triển quan trọng nhất là sự ra đời của Hệ thống Phẫu thuật da Vinci vào năm 1998, một hệ thống Robot cho phép các bác sĩ thực hiện các thủ thuật ít xâm lấn với kiểm soát và thị lực tốt hơn.

Sự mở rộng và đa dạng hóa (2000s): Đầu thế kỷ 21 chứng kiến sự mở rộng và đa dạng hóa của Robot y tế. Ngoài phẫu thuật, Robot bắt đầu được sử dụng trong phục hồi chức năng, hiện diện từ xa và tự động hóa được phòng.

Tích hợp AI và tự động hóa nâng cao (2010s-Hiện tại): Những phát triển gần đây nhất trong Robot y tế bao gồm việc tích hợp trí tuệ nhân tạo (AI) và học máy, dẫn đến tự động hóa và hiệu quả cao hơn. Những tiến bộ này đã tạo điều kiện cho sự hỗ trợ Robot cá nhân hóa và thích ứng hơn trong phẫu thuật, phục hồi chức năng và chăm sóc bệnh nhân. Hiện nay, Robot được sử dụng cho các nhiệm vụ như khử trùng phòng bệnh, giao thuốc, và hỗ trợ trong các thủ thuật phẫu thuật phức tạp với độ chính xác và kiểm soát cao.

1.1.3 Tình trạng sử dụng trên thế giới

Moxi: Moxi là một Robot phát thuốc đang được phát triển bởi Diligent Robotics. Robot này có khả năng di chuyển trong bệnh viện và thực hiện nhiệm vụ như phát thuốc và lấy mẫu máu.



Hình 1.4 Robot Moxi

TUG: TUG là một hệ thống Robot đa năng của Aethon, có khả năng phát thuốc, giao đồ ăn, và vận chuyển các vật dụng trong bệnh viện.



Hình 1.5 Robot TUG thay y tá vận chuyển vật dụng

ZenZoe: ZenZoe là một Robot AMR của ASTI Mobile Robotics, có thể khử trùng các khu vực trong bệnh viện bằng ánh sáng cực tím. Robot này có thể diệt khuẩn 99,99% các vi khuẩn và virus, bao gồm cả virus SARS-CoV-2.



Hình 1.6 ZenZoe đang loại bỏ Covid-19 trong một bệnh viện

1.1.4 Tình trạng sử dụng ở Việt Nam



Hình 1.7 Đưa Robot y tế hỗ trợ khu cách ly Covid-19 tại Hà Nam

Bệnh viện Bạch Mai đã đưa vào sử dụng hệ thống Robot y tế vận chuyển từ tháng 5 năm 2021. Hệ thống này gồm 4 Robot AMR có khả năng chở được 300 kg hàng hóa, di chuyển với tốc độ 1,5 m/s và hoạt động liên tục trong 8 giờ. Các Robot được kết nối với trung tâm giám sát, điều khiển và có thể giao tiếp từ xa với người bệnh.

1.2 Các dạng Robot tự hành

1.2.1 Giới thiệu các dạng Robot

Với sự phát triển của công nghệ, có nhiều dạng Robot hiện đại được thiết kế để phục vụ trong nhiều lĩnh vực khác nhau, từ công nghiệp, y tế đến giáo dục và giải trí. Dưới đây số dạng Robot hiện đại ngày nay:

AMR (Autonomous Mobile Robot):



Hình 1.8 SBC MIO-5373 của Advantech

- Mô tả: Robot di động tự động có khả năng tự định hình đường đi và tránh vật cản mà không cần hệ thống dẫn đường cố định.
- Ứng Dụng: Giao hàng tự động, dịch vụ khách hàng, và quản lý nhà máy thông minh.

AGV (Automated Guided Vehicle):



Hình 1.9 Robot AGV VTPost

- Mô tả: Robot di động được thiết kế để tự động di chuyển trong môi trường cố định, thường sử dụng trong nhà máy hoặc kho hàng để vận chuyển hàng hóa.
- Ứng Dụng: Logistik và quản lý kho hàng tự động.

UAV (Unmanned Aerial Vehicle):



Hình 1.10 Máy bay không người lái (UAV)

- Mô tả: Robot không người lái có khả năng di chuyển trong không gian 3 chiều, thường được sử dụng trong quân sự, giám sát, nông nghiệp thông minh.
- Ứng Dụng: Giám sát môi trường, quản lý đám cháy, và giao hàng từ không gian.

UGV (Unmanned Ground Vehicle):



Hình 1.11 THEMIS của Đức

- Mô tả: Robot không người lái được thiết kế để di chuyển trên mặt đất, sử dụng trong nhiều môi trường khác nhau.
- Ứng Dụng: Nhiệm vụ quân sự, khảo sát môi trường, và công việc nguy hiểm.

AUV (Autonomous Underwater Vehicle):



Hình 1.12 A9-M / AUV thăm dò đại dương

- Mô tả: Robot không người lái hoạt động dưới nước mà không cần sự can thiệp của con người.
- Ứng Dụng: Nghiên cứu, thăm dò dưới đại dương, và giám sát môi trường biển.

Delivery Robots:



Hình 1.13 Một Robot Starship Technologies giao một bưu kiện Starship Technologies

- Mô tả: Robot được sử dụng để tự động giao hàng trong các môi trường đô thị hoặc công nghiệp.
- Ứng Dụng: Giao hàng tự động và dịch vụ giao hàng nhanh.

1.2.2 Kết luận lựa chọn dạng Robot

Dựa trên tình trạng sử dụng của các loại Robot y tế tại thế giới và tại Việt Nam như Moxi, TUG, ZenZoe, có thể kết luận rằng Robot dạng AMR là lựa chọn phù hợp cho việc cải thiện hiệu suất và an toàn trong lĩnh vực y tế. Điều này đặc biệt quan trọng trong bối cảnh đại dịch COVID-19 và nhu cầu cần giảm thiểu tiếp xúc giữa nhân viên y tế và bệnh nhân.

Robot AMR có nhiều lợi thế so với các loại Robot khác như AGV, UAV, UGV... Một số lợi thế của Robot AMR là:

- Robot AMR không cần cài đặt cơ sở hạ tầng cố định như đường ray, dây nam châm, bóng đèn... để hướng dẫn Robot di chuyển. Robot AMR có thể sử dụng

các cảm biến, hệ thống định vị và thuật toán điều hướng để tìm đường đi tối ưu và thích ứng với môi trường thay đổi.

- Robot AMR có kích thước nhỏ gọn, tốc độ nhanh và độ chính xác cao, phù hợp với môi trường bệnh viện, nơi có nhiều người, thiết bị và hành lang hẹp. Robot AMR có thể di chuyển an toàn và hiệu quả trong bệnh viện, tránh va chạm và gây phiền nhiễu cho bệnh nhân và nhân viên y tế.
- Robot AMR có sự linh hoạt và mở rộng cao, cho phép thay đổi và cải tiến theo nhu cầu và điều kiện thực tế. Robot AMR có thể được lập trình để thực hiện nhiều nhiệm vụ khác nhau trong bệnh viện, như vận chuyển thực phẩm, rác, khăn trải, dụng cụ y tế, thuốc... Robot AMR cũng có thể được tích hợp thêm các chức năng khác như khử trùng, lau dọn, giao tiếp... để nâng cao hiệu quả và chất lượng dịch vụ.

Vì những lý do trên, Robot AMR là một lựa chọn tốt cho việc tự động hóa các công việc vận chuyển trong bệnh viện. Robot AMR giúp giảm thiểu chi phí nhân công, tăng năng suất, cải thiện môi trường làm việc và chăm sóc bệnh nhân.

1.3 Tổng quan về Robot AMR

1.3.1 Khái niệm:

- Robot AMR là viết tắt của Autonomous Mobile Robots, tức là Robot di động tự động, có khả năng tự điều hướng và thích ứng với môi trường xung quanh mà không cần sự can thiệp của con người.
- Robot AMR sử dụng các bộ cảm biến, camera, LIDAR, GPS và các thuật toán trí tuệ nhân tạo để xác định vị trí, phát hiện và tránh vật cản, tìm đường đi tối ưu và thực hiện các nhiệm vụ được giao.
- Robot AMR có thể được trang bị các thiết bị phụ trợ như cánh tay Robot, khay đựng, giỏ hàng, xe đẩy, máy khử trùng, máy chà sàn, máy hút bụi, máy phun

thuốc, máy bắn đinh, máy phát hiện bom, máy bay không người lái, v.v. để phục vụ cho các ứng dụng khác nhau.

1.3.2 Lợi ích:

- Robot AMR có thể giúp tăng hiệu quả, tiết kiệm chi phí, giảm rủi ro và nâng cao chất lượng công việc trong nhiều lĩnh vực như sản xuất, kho vận, bán lẻ, nông nghiệp, quân sự, y tế, giáo dục, giải trí, v.v.
- Robot AMR có thể làm việc liên tục, chính xác, nhanh chóng và an toàn hơn con người trong các môi trường khắc nghiệt, nguy hiểm hoặc khó tiếp cận.
- Robot AMR có thể hỗ trợ, hợp tác và tương tác với con người một cách linh hoạt, thân thiện và thông minh, giúp nâng cao năng suất, sáng tạo và trải nghiệm của con người.

1.3.3 Thách thức:

Robot AMR cũng đối mặt với nhiều thách thức trong quá trình phát triển và triển khai, như:

- Sự đầu tư ban đầu cao để mua sắm, lắp đặt và bảo trì Robot.
- Hỗ trợ kỹ thuật và đào tạo cho nhân viên y tế để sử dụng Robot hiệu quả.
- Thích nghi và hợp tác của con người với Robot, đảm bảo sự an toàn và tuân thủ các quy định khi làm việc cùng Robot.
- Sự giám sát và kiểm soát của con người để ngăn chặn các trường hợp lạm dụng hoặc sai lầm của Robot.
- Tuân thủ các tiêu chuẩn kỹ thuật, an toàn và pháp lý khi triển khai Robot trong bệnh viện.
- Xây dựng niềm tin và chấp nhận của bệnh nhân và nhân viên y tế đối với Robot.

Robot AMR trong y tế là một xu hướng công nghệ mới đang được phát triển và ứng dụng tại Việt Nam. Robot AMR có thể mang lại nhiều lợi ích cho các bệnh viện, như nâng cao hiệu quả, tiết kiệm chi phí, giảm tải cho nhân viên y tế và bảo vệ sức khỏe của cộng đồng.

Tuy nhiên, Robot AMR cũng cần được quản lý, điều khiển và bảo trì một cách chuyên nghiệp để đảm bảo an toàn và chất lượng dịch vụ y tế.

1.4 Đặt vấn đề

Trong môi trường bệnh viện, việc phân phối thuốc chính xác và kịp thời cho bệnh nhân không chỉ là nhiệm vụ quan trọng giúp cải thiện chất lượng chăm sóc sức khỏe mà còn tăng cường niềm tin và sự hài lòng của bệnh nhân. Hiện tại, các quy trình phát thuốc truyền thống thường đối mặt với nhiều thách thức như sai sót trong việc phân phối, nguy cơ nhiễm khuẩn chéo, và tốn kém về thời gian và nguồn nhân lực.

Trong bối cảnh này, sự xuất hiện của Robot tự hành, đặc biệt là Robot AMR (Autonomous Mobile Robot), mở ra một hướng tiếp cận mới và hiệu quả. Robot AMR, với khả năng tự định hình đường đi và tránh vật cản, mang đến giải pháp tối ưu hóa quy trình phân phối thuốc. Các ưu điểm của Robot AMR bao gồm:

- Giảm nguy cơ sai sót: Nhờ vào hệ thống nhận diện mã vạch và quét thông tin thuốc, Robot AMR giảm thiểu nguy cơ nhầm lẫn, quên lãng hoặc sai sót trong việc phát thuốc, tăng cường độ chính xác và an toàn.
- Tiết kiệm thời gian và nguồn lực: Robot AMR có khả năng hoạt động liên tục và tự động, giúp giảm đáng kể thời gian và công sức mà nhân viên y tế phải bỏ ra cho việc phân phối thuốc.

Mục tiêu chính của việc nghiên cứu và phát triển Robot AMR trong lĩnh vực y tế là cải thiện chất lượng dịch vụ y tế, nâng cao hiệu quả và an toàn trong việc phát thuốc cho bệnh nhân, đồng thời giảm bớt gánh nặng cho nhân viên y tế. Đồng thời, việc này cũng góp phần thúc đẩy sự phát triển của công nghệ Robot và trí tuệ nhân tạo, mở rộng khả năng ứng dụng của chúng trong lĩnh vực y tế, một bước tiến quan trọng trong việc chăm sóc sức khỏe toàn diện và hiện đại.

1.5 Mục tiêu nghiên cứu của đề án

- Thiết kế và xây dựng một hệ thống điện Robot AMR: [Arduino Uno], [Raspberry Pi], [Battery], [LED AD], [MODULE Relay 12VDC 4 Kênh], [Động Cơ DC Servo JGB37], [DC BTS7960 43A], ...
- Tích hợp các cảm biến và thiết bị điều khiển để Robot AMR có thể phát hiện được các vật thể để xây dựng bản đồ.
- Lập trình và cài đặt các thuật toán, để Robot AMR có thể nhận và xử lý các yêu cầu phát thuốc.
- Lập trình, cài đặt các thuật toán và gói package để Robot có thể di chuyển khi điều khiển bằng tay.

1.6 Phạm vi và phương pháp nghiên cứu về Robot phát thuốc trong bệnh viện

Thời gian nghiên cứu:

Nghiên cứu sẽ diễn ra từ ngày 16 tháng 10 năm 2023 đến ngày 21 tháng 1 năm 2024, cho phép thời gian đủ để thiết kế, triển khai và đánh giá hệ thống.

Đối tượng nghiên cứu:

Nghiên cứu sẽ tập trung vào việc triển khai và kiểm tra hệ thống Robot AMR trong việc:

- Phân phối thuốc cho 15 bệnh nhân tại các bệnh viện khác nhau.
- Khả năng tương tác điều khiển giữa bác sĩ với Robot.

Giới hạn và phương pháp nghiên cứu:

Báo cáo này tập trung vào việc phát triển một hệ thống Robot (AMR) cho môi trường bệnh viện, bao gồm việc xây dựng, lập trình, và cài đặt các thuật toán cùng giao thức kết nối. Nghiên cứu chủ yếu tập trung vào khả năng di chuyển của Robot, quét bản đồ 2D, cùng với việc đánh giá hiệu năng và tính khả thi của Robot trong điều kiện thực tế của bệnh viện.

Phạm vi hoạt động của Robot:

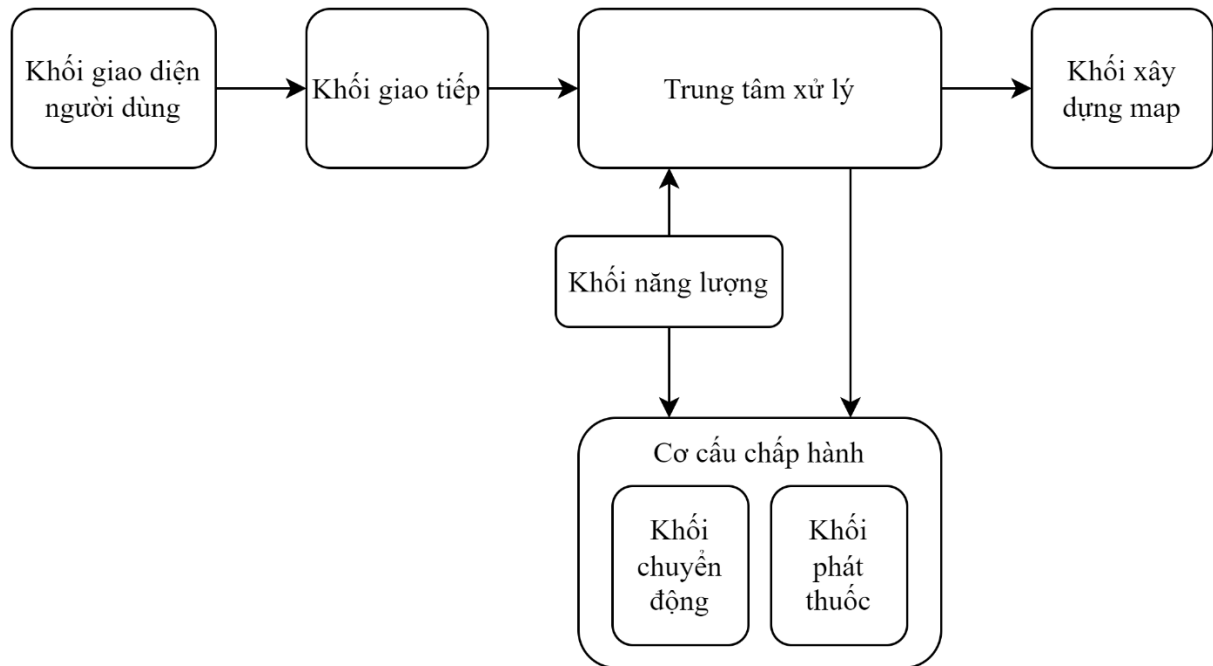
Robot sẽ được kiểm tra trong môi trường giới hạn là một tầng của bệnh viện, với phạm vi hoạt động trong vùng phủ sóng wifi.

Kết nối mạng:

Robot sẽ hoạt động trên cùng một mạng wifi, không chuyển đổi giữa các mạng khác nhau, để đảm bảo sự ổn định trong giao tiếp và truyền dữ liệu.

CHƯƠNG 2. MÔ HÌNH HÓA

2.1 Mô hình hóa



Hình 2.1 Mô hình hóa hệ thống

Hệ thống Robot (AMR) được xây dựng dựa trên nền tảng phát triển Robot với mục đích tối ưu hóa các tác vụ tự động hóa. Sự linh hoạt và hiệu quả là hai yếu tố chủ đạo trong thiết kế hệ thống, được thể hiện qua sự phối hợp giữa các khối chức năng thông qua một cơ chế giao tiếp và xử lý tập trung, đảm bảo hoạt động ổn định và hiệu quả của Robot.

2.1.1 Khối giao diện người dùng:

Khối giao diện người dùng đóng vai trò cực kỳ quan trọng trong việc tạo điều kiện giao tiếp giữa người dùng và hệ thống robot. Nó không chỉ cho phép người dùng nhập lệnh điều khiển trực tiếp và theo dõi quá trình thực hiện nhiệm vụ của robot như di chuyển và phát thuốc, mà còn có khả năng hiển thị các vật cản mà robot gặp phải trong quá trình hoạt động.

Qua giao diện này, người vận hành có thể xác định vị trí của Robot trên bản đồ đã được tạo sẵn hoặc đang được xây dựng, và thực hiện các điều chỉnh hoặc gửi lệnh mới một cách nhanh chóng và hiệu quả, tăng cường khả năng tương tác giữa người và máy.

2.1.2 Khối giao tiếp:

Khối giao tiếp là một thành phần cốt lõi của hệ thống Robot AMR, đóng vai trò cầu nối giữa hệ thống và môi trường xung quanh. Với sự hỗ trợ của công nghệ WiFi, khối giao tiếp này trao quyền cho Robot thực hiện các tác vụ giao tiếp không dây, qua đó đem lại khả năng linh hoạt cực kỳ cao trong việc di chuyển và thực hiện nhiệm vụ một cách độc lập.

Giao thức an ninh mạng SSH (Secure Shell) là một phần không thể thiếu trong khối giao tiếp, giúp mã hóa tất cả dữ liệu truyền đi, đảm bảo rằng mọi thông tin trao đổi giữa Robot và các điểm kiểm soát từ xa luôn được bảo vệ an toàn.

Thêm vào đó, sử dụng giao thức SSH cũng cho phép các nhà điều hành thực hiện các tác vụ điều khiển và bảo trì từ xa, bao gồm cả việc cập nhật phần mềm và firmware cho Robot mà không cần phải có mặt tại vị trí thực tế của chúng. Điều này không chỉ tăng cường khả năng tiếp cận và quản lý hệ thống robot một cách hiệu quả mà còn giảm thiểu thời gian chết và tối ưu hóa chu kỳ làm việc của robot.

Nhờ vào khối giao tiếp này, Robot có thể liên tục cập nhật thông tin và trạng thái đến người vận hành, cũng như nhận lệnh và phản hồi một cách nhanh chóng, giúp cho việc quản lý và điều hành hệ thống trở nên mượt mà và đồng bộ hơn. Đây là một bước tiến quan trọng trong việc nâng cao khả năng tự chủ và hiệu quả của hệ thống robot tự động..

2.1.3 Trung tâm xử lý:

Trung tâm xử lý trong hệ thống Robot Tự Động Di Chuyển (AMR) có vai trò trung tâm trong việc quản lý và điều phối hoạt động của robot. Đây là nơi xử lý mọi thông tin đầu vào từ các cảm biến, hệ thống định vị, và các yêu cầu từ người dùng, cũng như thông tin đầu ra để điều khiển cơ chế vận hành của robot. Nó đóng vai trò như một bộ não thông

minh, có khả năng phân tích dữ liệu, đưa ra quyết định nhanh chóng về hướng di chuyển hay các tác vụ cần thực hiện, và đảm bảo sự phối hợp liền mạch giữa các khối chức năng khác nhau của hệ thống.

Trung tâm xử lý được trang bị các thuật toán phức tạp cho việc tự động hóa, như thuật toán định tuyến. Nó liên tục tổng hợp và phân tích dữ liệu trong thời gian thực để đảm bảo rằng robot có thể thích ứng với môi trường đang thay đổi, cũng như thực hiện các nhiệm vụ được giao một cách chính xác và hiệu quả.

2.1.4 Khối xây dựng map:

Khối xây dựng map đảm nhận trách nhiệm tạo dựng và cập nhật bản đồ môi trường làm việc của Robot, là một yếu tố cốt lõi cho việc định vị và điều hướng trong không gian làm việc. Công việc của khối này bao gồm việc thu thập dữ liệu từ các cảm biến và camera, xây dựng một mô hình 2D chi tiết của môi trường xung quanh, và liên tục cập nhật nó để phản ánh bất kỳ thay đổi nào trong không gian làm việc.

Bản đồ này không chỉ đơn giản là một hình ảnh tĩnh; nó là một mô hình động được sử dụng để định vị chính xác vị trí của Robot. Khối xây dựng map cũng phối hợp chặt chẽ với các khối khác như khối giao tiếp và trung tâm xử lý để đảm bảo thông tin về môi trường luôn được cập nhật và chính xác.

Ngoài ra, khối xây dựng map còn sử dụng các thuật toán tiên tiến như SLAM (Simultaneous Localization and Mapping) để Robot có thể định vị bản thân mình trong khi đồng thời tạo ra bản đồ môi trường. Điều này cho phép Robot hoạt động một cách độc lập, thậm chí trong các môi trường chưa được biết đến trước đây, làm tăng khả năng tự chủ và hiệu quả của hệ thống AMR trong việc thực hiện nhiệm vụ và thích nghi với các điều kiện mới.

2.1.5 Cơ cấu chấp hành

2.1.5.1 Khối chuyển động:

Khối chuyển động trong hệ thống hoạt động như trái tim của hệ thống động lực học, nơi kiểm soát và quản lý mọi hoạt động di chuyển của robot. Nó bao gồm một loạt các thành phần như động cơ, bánh xe, và cơ cấu chấp hành, cùng với các bộ điều khiển và phần mềm cần thiết để thực hiện các tác vụ điều hướng nâng cao.

Khối chuyển động cũng phối hợp với các khối xử lý khác để thích ứng với các tình huống không ngờ tới trong môi trường làm việc, như thay đổi địa hình hoặc sự xuất hiện của vật cản mới. Sử dụng các thuật toán cảm biến tiên tiến và phản hồi từ các cảm biến, khối này liên tục điều chỉnh đường đi và cách thức vận hành để đạt được hiệu quả tối đa, đồng thời đảm bảo an toàn cho cả robot và môi trường xung quanh nó.

Sự chính xác trong việc thực hiện chuyển động không chỉ cải thiện hiệu suất tổng thể của AMR mà còn giúp tăng cường độ tin cậy và giảm thiểu rủi ro trong quá trình robot thực hiện các nhiệm vụ.

2.1.5.2 Khối Phát Thuốc:

Đây là chức năng đặc biệt của Robot, nhấn mạnh vào khả năng cung cấp thuốc cho bệnh nhân, một ứng dụng cụ thể trong lĩnh vực y tế. Khối này phải vận hành với độ chính xác cao để đảm bảo an toàn cho bệnh nhân.

Hệ thống AMR được mô hình hóa trong báo cáo này cho thấy sự tiên tiến trong công nghệ tự động hóa và tiềm năng lớn trong việc ứng dụng vào đời sống thực tế và các ngành công nghiệp khác nhau.

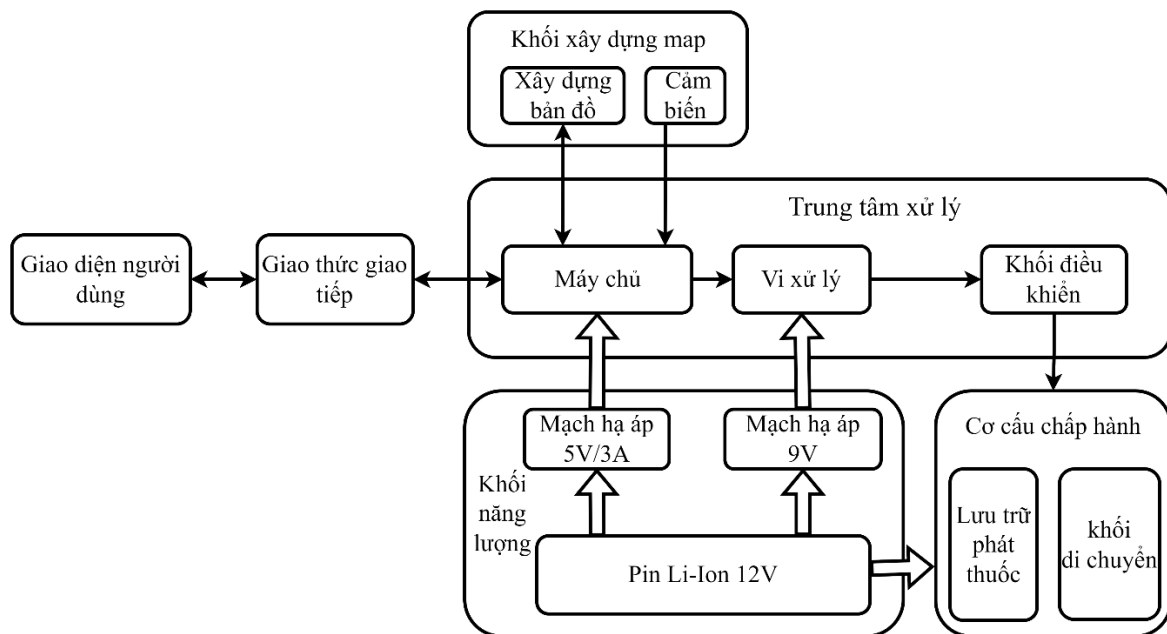
Khối này không chỉ bao gồm các cơ cấu cơ khí để lấy và giao thuốc mà còn cần có hệ thống phần mềm để kiểm soát việc cung cấp thuốc một cách chính xác. Sự tương tác này đòi hỏi sự chính xác không chỉ ở mức độ cơ khí mà còn ở mức độ phần mềm và dữ liệu.

2.1.6 Khối năng lượng

Khối năng lượng này có thể bao gồm các thành phần như pin (thường là pin Li-ion hoặc Li-Po do đặc tính năng lượng cao và sự linh hoạt trong kích thước), bộ quản lý pin để đảm bảo an toàn trong quá trình sạc và phóng điện, cũng như các mạch chuyển đổi điện áp để cung cấp nguồn phù hợp cho các thành phần điện tử khác nhau trong Robot. Đây là một hệ thống cần thiết để Robot có thể duy trì hoạt động trong thời gian dài mà không cần đến sự can thiệp của con người.

Trong một hệ thống Robot tự hành, quản lý năng lượng là cực kỳ quan trọng bởi nó không chỉ ảnh hưởng đến thời gian hoạt động của Robot mà còn đảm bảo hiệu quả và độ tin cậy của Robot khi thực hiện các nhiệm vụ được giao. Việc tối ưu hóa việc sử dụng năng lượng và quản lý sạc có thể giúp tăng cường khả năng tự chủ của Robot và giảm thiểu thời gian nghỉ do sạc pin.

2.2 Kiến trúc chi tiết hệ thống



Hình 2.2 Sơ đồ chi tiết hệ thống của Robot

2.2.1 Tổng quan

Sơ đồ biểu diễn kiến trúc hệ thống của một Robot được điều khiển bởi máy tính thông qua kết nối mạng không dây. Hệ thống này tích hợp các thành phần điện tử, cơ khí và phần mềm, tạo điều kiện cho Robot thực hiện các tác vụ tự động một cách chính xác và linh hoạt trong môi trường y tế.

2.2.2 Mô tả ngắn gọn chi tiết

1. Giao diện người dùng:

- **Người dùng:** Là người vận hành Robot, thường là bác sĩ, tương tác với Robot thông qua máy tính để nhập lệnh và nhận dữ liệu phản hồi.
- **Máy tính:** Thiết bị người dùng sử dụng để gửi lệnh và nhận dữ liệu từ Robot, chạy phần mềm điều khiển và giám sát Robot.

2. Giao thức giao tiếp:

- **Kết nối mạng không dây:** Cung cấp kết nối không dây giữa máy tính và Robot, cho phép truyền dữ liệu hai chiều.
- **SSH (Secure Shell):** Phương thức kết nối an toàn sử dụng để truy cập vào máy chủ từ xa qua mạng không dây.

3. Trung tâm xử lý:

- **Máy chủ:** Máy tính nhúng trong Robot, xử lý lệnh từ người dùng và điều khiển các thành phần khác của Robot.
- **Vi điều khiển:** Nhận lệnh từ máy chủ và đảm bảo cơ cấu chấp hành hoạt động chính xác.

4. Khối xây dựng map:

- **Cảm biến:** Sử dụng LIDAR để đo khoảng cách và giúp Robot tạo bản đồ 3D của môi trường xung quanh.
- **Xây dựng bản đồ:** Áp dụng thuật toán SLAM để tạo và cập nhật bản đồ môi trường.

5. Khối năng lượng:

- **Pin:** Nguồn năng lượng chính cho Robot, cung cấp điện cho máy chủ và các thành phần điện tử khác.
- **Mạch hạ áp:** Chuyển đổi điện áp từ pin để cung cấp nguồn phù hợp cho máy chủ và vi điều khiển.

6. Cơ cấu chấp hành:

6.1 Khối chuyển động:

Sử dụng 2 bánh xe để di chuyển và hệ thống đèn để báo dừng (đèn đỏ) và đi (đèn xanh lá).

6.2 Khối phát thuốc:

Bao gồm 3 băng tải với 15 ngăn thuốc, mỗi ngăn dành cho một bệnh nhân. Đèn xanh dương báo hiệu việc có thuốc sẵn sàng để phát.

2.2.3 Quy trình hoạt động

Bước 1: Bác sĩ nhập lệnh vào máy tính.

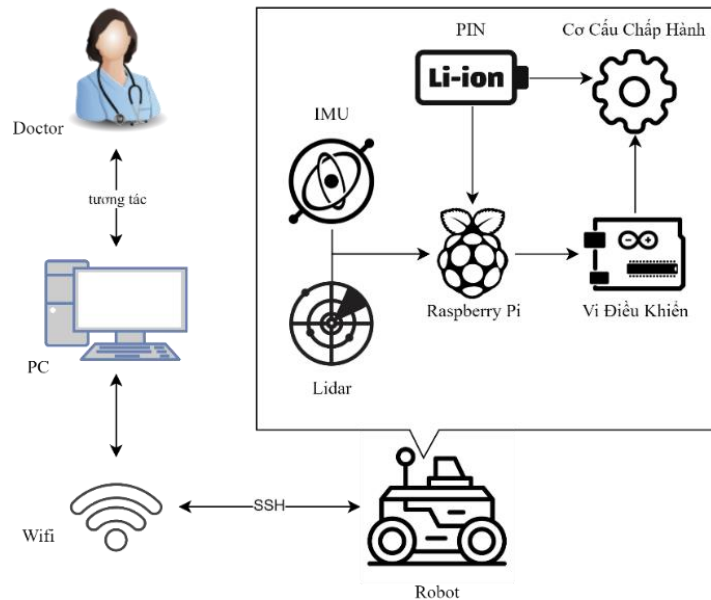
Bước 2: Máy tính gửi lệnh đến Robot qua SSH và Wi-Fi.

Bước 3: Máy chủ trong Robot xử lý lệnh, giao tiếp với cảm biến LIDAR và vi điều khiển.

Bước 4: LIDAR thu thập dữ liệu và gửi về máy chủ để xây dựng bản đồ.

Bước 5: Máy chủ điều khiển cơ cấu chấp hành thông qua vi điều khiển dựa trên lệnh và dữ liệu từ cảm biến.

Bước 6: Robot sử dụng năng lượng từ pin Li-ion để thực hiện các tác vụ như di chuyển và phát thuốc.



Hình 2.3 Mô tả kiến trúc chi tiết hệ thống

2.3 Phần mềm hỗ trợ và phần mềm chính xây dựng Robot

Các phần mềm trong xây dựng một hệ thống cho Robot

Tiêu Chí	ROS	OROCOS	MOOS
Ưu Điểm	Tính linh hoạt cao. Hỗ trợ đa nền tảng và phần cứng. Cộng đồng lớn, nhiều tài nguyên.	Đảm bảo tính thời gian thực. Mở rộng cao. Tích hợp được với ROS.	Nhẹ và đơn giản. Hỗ trợ đặc biệt cho Robot dưới nước. Tương thích với các hệ thống khác.
Nhược Điểm	Khó khăn trong cài đặt và cấu hình. Vấn đề về độ trễ và đồng bộ hóa.	Khó sử dụng. Chỉ hỗ trợ C++. Cộng đồng nhỏ.	Hạn chế về tính năng. Hỗ trợ ngôn ngữ giới hạn. Tài liệu không đầy đủ.
Ứng Dụng	Lựa chọn hàng đầu cho dự án phức tạp, tích hợp nhiều thành phần.	Thích hợp cho ứng dụng cần độ chính xác cao, thời gian thực.	Lý tưởng cho dự án Robot dưới nước hoặc HĐH nhẹ, ít phức tạp.

Bảng 2.1 So sánh phần mềm xây dựng Robot

Lí do chọn phần mềm ROS

Sau khi xem xét các tiêu chí so sánh giữa ROS, OROCOS và MOOS, ROS nổi bật như là lựa chọn tối ưu cho việc xây dựng và phát triển hệ thống Robot. Mặc dù có thể gặp phải một số thách thức trong cài đặt và cấu hình ban đầu, các ưu điểm của ROS phù hợp với một loạt các yêu cầu phức tạp và đa dạng trong lĩnh vực Robot.

ROS



Hình 2.4 Logo ROS Noetic một phiên bản của ROS

ROS (Robot Operating System) vượt trội so với các hệ thống khác nhờ vào khả năng phát triển các gói độc lập và linh hoạt, mặc dù nó đòi hỏi một khoảng thời gian đáng kể để nghiên cứu và phát triển. Các gói phần mềm của ROS có thể phức tạp và cần nhiều thời gian để hiểu rõ, nhưng chính sự phức tạp này lại chứng minh sự mạnh mẽ và đa dạng của nền tảng. Mỗi gói trong ROS được thiết kế để giải quyết một bộ phận cụ thể của vấn đề Robot và có thể được kết hợp với nhau để tạo nên một hệ thống hoàn chỉnh.

Ưu điểm của việc sử dụng các gói độc lập là cho phép các nhà phát triển chọn lựa và tích hợp chỉ những gì họ cần, giúp tối ưu hóa cả về hiệu suất lẫn tài nguyên. Điều này cũng tạo điều kiện cho việc cộng tác và chia sẻ giữa các nhóm nghiên cứu, mỗi nhóm có thể đóng góp vào hệ thống một cách độc lập mà không làm ảnh hưởng đến công việc của nhau.

Xét trên tất cả các phương diện, từ tính năng, mức độ hỗ trợ, cho đến cộng đồng và tài nguyên, ROS là lựa chọn đáng tin cậy và mạnh mẽ nhất cho phát triển hệ thống Robot. Đây là nền tảng sẽ không chỉ hỗ trợ cho đồ án hiện tại mà còn có khả năng thích ứng với các thách thức và nhu cầu tương lai trong lĩnh vực Robot.

2.3.1 ROS Noetic

1) Cách hỗ trợ trong việc điều khiển Robot AMR.

ROS Noetic – phiên bản ổn định của ROS cung cấp một loạt các công cụ và thư viện hỗ trợ trong việc điều khiển Robot AMR (Autonomous Mobile Robot). Dưới đây là một số cách mà ROS Noetic hỗ trợ trong lĩnh vực này:

- Gói Navigation Stack: Gói này giúp Robot di động tự động trong môi trường không gian 2D hoặc 3D. Nó tích hợp các phương pháp điều hướng như DWA (Dynamic Window Approach) để tránh vật cản và đạt được mục tiêu.
- Công cụ giả lập Gazebo: ROS Noetic tích hợp với Gazebo, một môi trường mô phỏng 3D mạnh mẽ. Điều này giúp trong quá trình phát triển và kiểm thử chương trình điều khiển Robot mà không cần sử dụng thiết bị thực tế.
- Gói SLAM (Simultaneous Localization and Mapping): Gói này cho phép Robot xây dựng bản đồ của môi trường xung quanh và đồng thời xác định vị trí của mình trên bản đồ đó.
- Bộ lọc Kalman và các gói liên quan: ROS Noetic hỗ trợ các bộ lọc Kalman và các phương pháp ước lượng trạng thái khác. Điều này quan trọng trong việc nâng cao độ chính xác của vị trí và trạng thái của Robot.
- ROS Control: Gói này cung cấp các công cụ để điều khiển cảm biến và actuator trên Robot. Nó giúp tạo ra các giao thức điều khiển chung giữa các loại Robot khác nhau.

- RViz và công cụ diễn giải dữ liệu: RViz là công cụ giúp theo dõi trực quan trạng thái của Robot trong môi trường 3D. Nó hỗ trợ hiển thị dữ liệu từ các cảm biến và giúp định vị Robot trong không gian.
- Cộng đồng lớn và tài liệu phong phú: ROS Noetic được sử dụng rộng rãi và có một cộng đồng lớn. Người dùng có thể tận dụng tài nguyên, hỏi đáp và chia sẻ kinh nghiệm thông qua các diễn đàn và trang web chuyên ngành.
- Hỗ trợ đa nền tảng: ROS Noetic hỗ trợ nhiều hệ điều hành như Ubuntu, Debian và Windows, tạo điều kiện thuận lợi cho phát triển trên nhiều loại thiết bị và máy tính nhúng.

Tóm lại, ROS Noetic cung cấp một hệ sinh thái đầy đủ cho việc phát triển, kiểm thử và triển khai Robot AMR. Sự hỗ trợ từ cộng đồng và sự kết hợp với các công cụ mô phỏng giúp tăng tính linh hoạt và giảm rủi ro khi triển khai vào môi trường thực tế.

2.3.2 Visual Studio Code

1) Lý do tại sao chọn sử dụng.

Visual Studio Code (VSCode) là một trình soạn thảo mã nguồn mở và mạnh mẽ được phát triển bởi Microsoft. Dưới đây là một số lý do nên chọn sử dụng Visual Studio Code cho dự án ROS (Robot Operating System):

- Hỗ trợ ngôn ngữ rộng rãi: VSCode hỗ trợ nhiều ngôn ngữ lập trình, bao gồm C++, Python và nhiều ngôn ngữ khác được sử dụng phổ biến trong phát triển ROS. Ngoài ra, VSCode còn có các tiện ích mở rộng và trình môi trường phát triển tích hợp (IDE) chất lượng cao cho các ngôn ngữ này.
- Tiện ích mở rộng ROS: VSCode có các tiện ích mở rộng cho ROS như "ROS for Visual Studio Code" giúp tối ưu hóa quy trình làm việc với ROS. Các tiện ích mở rộng này cung cấp các tính năng như xem biểu đồ ROS, quản lý gói ROS và hỗ trợ tự động hoàn thành mã.

- Nhẹ nhàng và linh hoạt: VSCode là một ứng dụng nhẹ và nhanh chóng, không làm tốn tài nguyên hệ thống quá mức. Có thể mở rộng chức năng của VSCode thông qua cộng đồng các tiện ích mở rộng.
- Tích hợp terminal: VSCode có terminal tích hợp giúp thực hiện các lệnh ROS một cách thuận tiện từ trong môi trường VSCode.
- Tích hợp debugging: VSCode hỗ trợ tích hợp các công cụ debugging, giúp theo dõi và gỡ lỗi mã nguồn ROS một cách hiệu quả.
- Hỗ trợ đa nền tảng: VSCode hỗ trợ đa nền tảng, chạy trên Windows, macOS và Linux, giúp đảm bảo tính tương thích trên nhiều môi trường phát triển ROS.
- Cộng đồng và cập nhật liên tục: VSCode có một cộng đồng lớn và được cập nhật thường xuyên với các tính năng mới và sửa lỗi.

2) Cách cài đặt

Lệnh cài đặt trên terminal:

```
sudo apt update
sudo apt install software-properties-common apt-transport-https wget
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor >
microsoft.gpg
sudo mv microsoft.gpg /etc/apt/trusted.gpg.d/microsoft.gpg
sudo add-apt-repository "deb [arch=amd64]
https://packages.microsoft.com/repos/vscode stable main"
sudo apt update
sudo apt install code
```

2.4 Hệ điều hành cho ROS

Hệ Điều Hành	Ưu Điểm	Nhược Điểm
Ubuntu	Hỗ trợ đầy đủ từ cộng đồng ROS. Dễ sử dụng, giao diện thân thiện. Tốt cho cả người mới và chuyên gia. Đa dạng phần mềm và thư viện hỗ trợ.	Yêu cầu phần cứng tương đối cao so với các HĐH nhẹ khác.
Fedora	Cập nhật nhanh chóng các phần mềm mới. Hỗ trợ tốt cho các ứng dụng doanh nghiệp.	Ít được cộng đồng ROS sử dụng hơn. Có thể không ổn định như Ubuntu.
Debian	Ổn định và bảo mật cao. Tốt cho các máy có cấu hình thấp.	Không thường xuyên cập nhật. Ít thân thiện với người mới bắt đầu.
Windows (với WSL)	Thuận tiện cho người dùng quen với Windows. Hỗ trợ WSL cho phép chạy ROS.	Cần cấu hình thêm. Hiệu suất không tốt như khi chạy trên Linux.
Arch Linux	Cấu hình linh hoạt và tùy chỉnh cao.	Không dành cho người mới. Cần cấu hình và bảo trì thường xuyên.

Bảng 2.2 So sánh ưu nhược điểm của hệ điều hành

2.4.1 Chọn hệ điều hành Ubuntu

Sử dụng hệ điều hành Ubuntu cho một dự án ROS (Robot Operating System) có nhiều lợi ích. Dưới đây là một số lý do mà nhiều nhà phát triển ROS lựa chọn Ubuntu:

- Hỗ trợ ROS chính thức: ROS cung cấp phiên bản chính thức và hỗ trợ chính thức cho Ubuntu. Điều này đảm bảo tính tương thích và ổn định giữa ROS và hệ điều hành.
- Quy trình cài đặt đơn giản: Ubuntu cung cấp các gói cài đặt cho ROS, giúp quá trình cài đặt và cấu hình trở nên đơn giản hơn so với nhiều hệ điều hành khác.
- Cộng đồng lớn: Ubuntu có một cộng đồng lớn và tích cực, mang lại lợi ích trong việc tìm kiếm giải pháp cho vấn đề cụ thể và chia sẻ kiến thức.
- Hỗ trợ tích hợp cho phần cứng: Nhiều driver và phần mềm hỗ trợ cho các thiết bị phần cứng được phát triển và kiểm thử chủ yếu trên Ubuntu.
- Hệ thống gói Debian: Ubuntu sử dụng hệ thống gói Debian, giúp quản lý và cập nhật các gói phần mềm một cách dễ dàng.
- Đa nhiệm và ổn định: Ubuntu được biết đến với tính ổn định và khả năng đa nhiệm, quan trọng khi phát triển và triển khai các ứng dụng ROS phức tạp.
- Hỗ trợ đa nền tảng: Ubuntu có sẵn cho nhiều kiến trúc x86 và ARM, giúp tương thích với nhiều loại phần cứng, từ máy tính cá nhân đến thiết bị nhúng.
- Sự linh hoạt và tích hợp: Ubuntu hỗ trợ một loạt các công nghệ và giao thức, giúp tích hợp dễ dàng với các phần cứng và phần mềm khác.
- Cập nhật liên tục: Hệ điều hành Ubuntu nhận các cập nhật định kỳ, đảm bảo tính mới mẻ và bảo mật.
- Đội ngũ phát triển mạnh mẽ: Ubuntu được phát triển và duy trì chủ yếu bởi Canonical, một công ty với đội ngũ phát triển lớn và nhiều kinh nghiệm.

2.4.2 Cài đặt hệ điều hành Ubuntu

Cài đặt Ubuntu bằng Raspberry Pi Imager

Hãy truy cập trang web Raspberry Pi Foundation và tải xuống phiên bản chính xác của Raspberry Pi Imager cho hệ điều hành.

Khi Raspberry Pi Imager được cài đặt:

1. Lắp thẻ SD vào laptop hoặc máy tính.
2. Khởi chạy ứng dụng Raspberry Pi Imager.
3. Nhấp vào Choose OS. Bây giờ Imager sẽ truy xuất và hiển thị thông tin về các hệ điều hành khác nhau tương thích với Raspberry Pi.

Nếu cài Ubuntu Server:

Đầu tiên, cập nhật Raspberry Pi:

```
sudo apt update  
sudo apt upgrade
```

Sau đó, có thể cài đặt môi trường desktop ưa thích. Ví dụ:

```
sudo apt install ubuntu-mate
```

Sau khi cài đặt môi trường desktop, hãy khởi động lại Pi bằng lệnh sau:

```
sudo reboot
```

Khi Raspberry Pi khởi động, môi trường desktop sẽ sẵn sàng để bạn sử dụng

2.5 Truyền thông từ máy tính đến Robot

2.5.1 Cài đặt wifi cho Raspberry Pi 4 Model B.

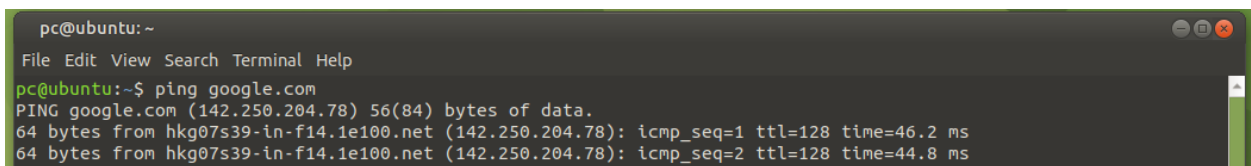
Cài đặt kết nối WiFi cho Raspberry Pi 4 Model B là bước quan trọng để thiết lập truyền thông giữa máy tính và Robot. Quá trình này bao gồm các bước sau:

Kết nối với WiFi từ thiết bị đầu cuối trong Ubuntu:

Đầu tiên, cần xác định Raspberry Pi 4 có kết nối internet không bằng cách sử dụng lệnh:

```
ping google.com
```

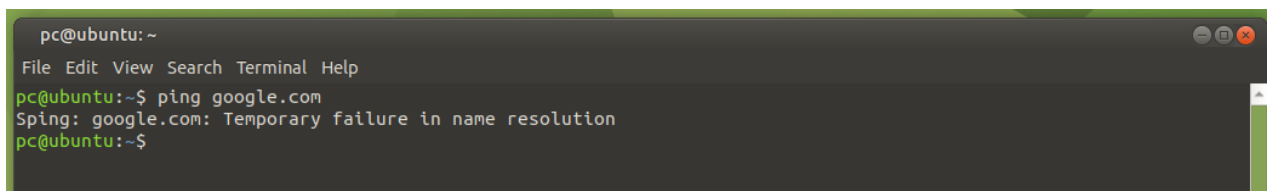
Khi có mạng: Raspberry Pi sẽ hiển thị các phản hồi từ google.com, cho thấy đã kết nối thành công với internet.



```
pc@ubuntu: ~  
File Edit View Search Terminal Help  
pc@ubuntu:~$ ping google.com  
PING google.com (142.250.204.78) 56(84) bytes of data.  
64 bytes from hkg07s39-in-f14.1e100.net (142.250.204.78): icmp_seq=1 ttl=128 time=46.2 ms  
64 bytes from hkg07s39-in-f14.1e100.net (142.250.204.78): icmp_seq=2 ttl=128 time=44.8 ms
```

Hình 2.5 Giao diện khi dùng lệnh *ping google.com* thành công

Khi không có mạng: Sẽ không nhận được phản hồi, chỉ ra rằng cần thiết lập kết nối mạng.



```
pc@ubuntu: ~  
File Edit View Search Terminal Help  
pc@ubuntu:~$ ping google.com  
Sping: google.com: Temporary failure in name resolution  
pc@ubuntu:~$
```

Hình 2.6 Giao diện khi dùng lệnh *ping google.com* không thành công

Chỉnh sửa tập tin cấu hình Netplan:

```
sudo nano /etc/netplan/50-cloud-init.yaml
```

Thêm vào tập tin cấu hình như sau:

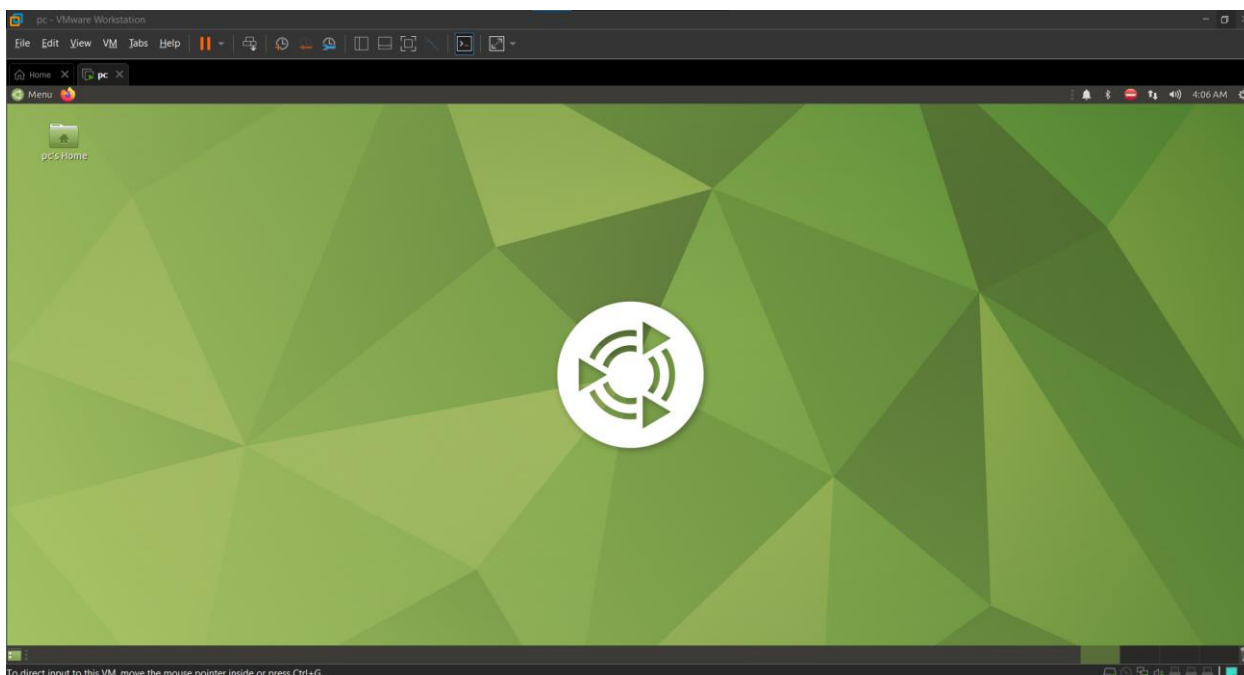
```
network:
  ethernets:
    eth0:
      dhcp4: true
      optional: true
  version: 2
  wifis:
    wlan0:
      dhcp4: true
      optional: true
      access-points:
        "SSID_name":
          password: "WiFi_password"
```

Sau khi chỉnh sửa, áp dụng cấu hình bằng lệnh:

```
sudo netplan apply
```

Kiểm tra lại kết nối: Cuối cùng, kiểm tra lại kết nối internet bằng lệnh ping google.com để đảm bảo Raspberry Pi đã kết nối thành công với WiFi.

2.5.2 Cài đặt máy ảo trên PC



Hình 2.7 Màn hình giao diện Ubuntu Mate

Cài đặt máy ảo:

Tải về file iso > Chọn Create a New Virtual Machine trên VMware > Chọn Installer disc image file (iso) và bấm Browse > Chọn file iso và bấm Open > Chọn Next > Thiết lập thông tin người dùng và bấm Next > Cài đặt ổ cứng ảo và bấm Next > Thiết lập dung lượng cho ổ cứng ảo và bấm Next > Check vào ô Power on this virtual machine after creation và bấm Finish.

Cài đặt hệ điều hành:

Chọn layout bàn phím và bấm Continue > Chọn Normal installation, check vào ô Download updates while installing Ubuntu và bấm Continue > Chọn Erase disk and install Ubuntu và Install now > Chọn Continue > Chọn múi giờ và Continue > Thiết lập thông tin người dùng và bấm Continue > Chọn Restart Now.

2.5.3 Giao thức mạng SSH

Trong bối cảnh sử dụng ROS để phát triển robot, việc thiết lập một kênh giao tiếp bảo mật và đáng tin cậy giữa máy tính cá nhân (PC) và Robot là cực kỳ quan trọng. SSH (Secure Shell) được chọn làm phương tiện giao tiếp chính do khả năng mã hóa mạnh mẽ, đảm bảo an toàn thông tin truyền giữa các thiết bị. Qua SSH, người dùng có thể thực hiện các lệnh từ xa, truyền tệp và cấu hình hệ thống mà không lo ngại về sự xâm nhập. Báo cáo này mô tả chi tiết cách thiết lập và sử dụng SSH trong môi trường ROS, cũng như giải thích các lợi ích và phương pháp giải quyết vấn đề khi xảy ra.

Thiết lập SSH và Cấu hình Môi trường cho Robot Operating System (ROS)

Để thiết lập một kết nối an toàn giữa máy tính và Robot thông qua SSH và cấu hình môi trường cho ROS, các bước sau cần được thực hiện:

Khởi động Dịch vụ SSH:

Đầu tiên, cần khởi động dịch vụ SSH trên máy chủ (trong trường hợp này là máy tính cá nhân hoặc Raspberry Pi) bằng lệnh:

```
sudo systemctl start ssh
```

Cấu hình SSH:

Sửa đổi tập tin cấu hình SSH để cho phép đăng nhập bằng tài khoản root:

```
sudo nano /etc/ssh/sshd_config
```

Thay đổi hoặc thêm dòng sau vào tập tin:

PermitRootLogin yes

Cấu hình Môi trường cho ROS:

Sửa đổi tập tin ~/.bashrc để cài đặt các biến môi trường cần thiết cho ROS:

```
nano ~/.bashrc
```

Thêm các dòng sau vào cuối tập tin:

```
ROS_MASTER_URI = http://IP_OF_PC:11311
```

```
ROS_HOSTNAME = IP_OF_ROBOT
```

Hoặc, nếu máy tính cá nhân là ROS Master:

```
ROS_MASTER_URI = http://IP_OF_PC:11311
```

```
ROS_HOSTNAME=IP_OF_PC
```

Kết nối SSH đến Robot:

Sử dụng SSH để kết nối từ máy tính cá nhân đến Robot:

```
ssh NAME_OF_ROBOT@IP_OF_ROBOT
```

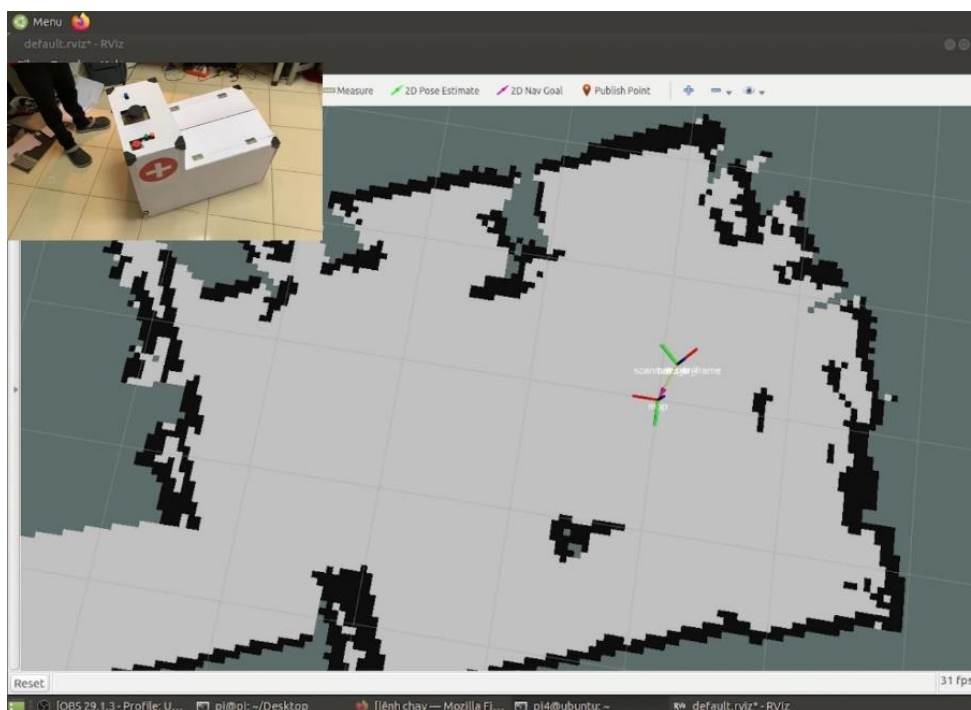
Trong đó

NAME_OF_ROBOT là tên người dùng trên Robot

IP_OF_ROBOT là địa chỉ IP của Robot.

2.6 Quy trình tạo và hiển thị bản đồ

Bản đồ tạo từ sơ đồ tầng hoặc bản vẽ kiến trúc khác với bản đồ tạo bằng SLAM (Simultaneous Localization and Mapping) trong ROS ở một số điểm chính:



Hình 2.8 Xây dựng bản đồ bằng SLAM trên Rviz



Hình 2.9 Xây dựng bản đồ theo bản vẽ kiến trúc

Nguồn dữ liệu: Bản đồ từ sơ đồ tầng dựa trên các bản vẽ hoặc kế hoạch đã được định trước, trong khi bản đồ SLAM được tạo ra dựa trên dữ liệu cảm biến thực tế (như từ LIDAR hoặc camera).

Độ chính xác và khả năng thích ứng: Bản đồ SLAM phản ánh môi trường hiện tại và có thể thích ứng với sự thay đổi hoặc chướng ngại vật, trong khi bản đồ từ bản vẽ có thể không tính đến những thay đổi. Chúng là các vật mà ngoài tầm LIDAR quét được trên một mặt phẳng, các vật này thường được đặt thấp hơn so với độ cao của LIDAR.

Mục đích và cách sử dụng: Bản đồ từ sơ đồ tầng thường được sử dụng cho việc lập kế hoạch ban đầu hoặc mô phỏng, trong khi bản đồ SLAM thiết yếu cho việc điều hướng và ra quyết định thời gian thực.

2.6.1 Không gian làm việc trong ROS (Workspace) .

- **Chuẩn bị**

Cài đặt hệ điều hành Ubuntu 20.04 cho Raspberry Pi 4 Model B

Cài đặt ROS Noetic

- **Cài đặt không gian làm việc**

Mở Terminal nhập:

```
mkdir ~/catkin_ws/src
```

Không gian làm việc Catkin

```
cd ~/catkin_ws/src
```

Truy cập chứa mã nguồn của các gói ROS

catkin_make - biên dịch và xây dựng các gói ROS trong không gian làm việc Catkin.

2) Gói (Package).

Trong ROS (Robot Operating System) 1, tạo một gói (package) bằng cách thực hiện các bước sau:

- 1) Mở một cửa sổ terminal.
- 2) Di chuyển đến thư mục src của workspace.

Thông thường, workspace sẽ nằm ở đường dẫn /path/to/your/workspace/src.

- 3) Sử dụng lệnh `catkin_create_pkg` để tạo gói.

Ví dụ:

```
catkin_create_pkg my_package_name rospy std_msgs
```

Trong đó:

- `my_package_name` là tên của gói bạn muốn tạo. Thư mục `my_package_name` trong thư mục src của workspace, thư mục này chứa một số tệp và thư mục mẫu, bao gồm tệp `CMakeLists.txt` và `package.xml` cho gói.

- `rospy` và `std_msgs` là các phụ thuộc (dependencies) của gói.

3) Có thể chỉnh sửa tệp

`CMakeLists.txt` và `package.xml` để cấu hình gói.

4) Chỉnh sửa gói

Sử dụng lệnh `catkin_make` để xây dựng lại toàn bộ workspace để gói mới được biên dịch.

• Gói quét LIDAR

```
https://github.com/robopeak/RPLIDAR\_ros.git
```

`robopeak/RPLIDAR_ros`: một gói ROS để sử dụng cảm biến LIDAR RPLIDAR A1/A2/A3. Đọc kết quả quét thô từ RPLIDAR bằng SDK của RPLIDAR và chuyển đổi thành thông

điệp LaserScan của ROS. Gói này được tạo ra bởi RoboPeak Team, một nhóm nghiên cứu và phát triển công nghệ Robot của công ty Slamtec.

Chức Năng: Gói RPLIDAR_ros được sử dụng để tương tác với cảm biến LIDAR RPLIDAR A1/A2/A3. Nó đọc dữ liệu quét thô từ RPLIDAR và chuyển đổi chúng thành thông điệp LaserScan trong ROS, được tích hợp vào Robot để giúp nhận dạng và điều hướng môi trường xung quanh.

```
rostopic list  
rostopic echo /scan
```

- **Gói SLAM**

```
https://github.com/tu-darmstadt-ros-pkg/hector\_slam.git
```

tu-darmstadt-ros-pkg/hector_slam là một gói ROS liên quan đến việc thực hiện SLAM trong các môi trường không có cấu trúc như những kịch bản Tìm kiếm và Cứu hộ Đô thị (USAR) của cuộc thi RoboCup Rescue. Gói này được tạo ra bởi Hector Team, một nhóm nghiên cứu và thi đấu Robot của trường Đại học Kỹ thuật Darmstadt, Đức.

Chức Năng: Gói hector_slam cung cấp các công cụ cho việc thực hiện SLAM (Simultaneous Localization and Mapping) trong môi trường không cấu trúc. hector_slam được sử dụng để tạo ra bản đồ động của bệnh viện, cho phép Robot cập nhật vị trí của mình trong thời gian thực và điều hướng một cách chính xác.

- **Chạy rviz**

Lưu bản đồ

```
mkdir ~/catkin_ws/maps  
cd ~/catkin_ws/maps  
roscore
```

Bây giờ tải bản đồ. Trong cửa sổ terminal mới, gõ:

```
roslaunch map_server map_server my_map.yaml
```

Các file YAML chứa dữ liệu bản đồ thường chỉ chứa văn bản và có kích thước khá nhỏ, thường chỉ vài KB đến vài M

Mở rviz trong một thiết bị đầu cuối khác.

```
rviz
```

Nhấp vào Thêm ở dưới cùng bên trái và thêm màn hình Bản đồ.

Trong Chủ đề bên dưới phần Bản đồ, chọn /map.

2.6.2 Tích hợp SLAM và LIDAR

Sử dụng LIDAR để thu thập dữ liệu cho Robot. LIDAR (Light Detection and Ranging) là một công nghệ cảm biến chủ chốt trong việc thu thập dữ liệu môi trường cho Robot. Công nghệ này sử dụng tia laser để đo khoảng cách và tạo ra một bức tranh chi tiết 3D về môi trường xung quanh. Phương pháp này đo lường thời gian mà tia laser mất để đi từ nguồn phát đến vật thể và quay trở lại, từ đó xây dựng dữ liệu chiều sâu.

Tự định vị (Localization): LIDAR giúp Robot xác định vị trí của mình trong môi trường bằng cách liên tục cập nhật dữ liệu về các điểm đặc trưng xung quanh. Thuật toán tự định vị sử dụng dữ liệu này để điều chỉnh vị trí của Robot trên bản đồ.

Xây dựng bản đồ (Mapping): Dữ liệu LIDAR được sử dụng để tạo bản đồ 2D hoặc 3D của môi trường. Các điểm dữ liệu chiều sâu được tích hợp và liên kết với vị trí của Robot, tạo nên bản đồ chi tiết và chính xác.

2.6.3 SLAM trong ROS

(SLAM): SLAM là quá trình đồng thời xây dựng bản đồ và tự định vị Robot trong môi trường không biết. ROS tích hợp nhiều gói SLAM như Gmapping, Hector SLAM và Cartographer.

Các bước thực hiện SLAM trong ROS:

- Thu thập dữ liệu: Robot sử dụng LIDAR để thu thập dữ liệu về môi trường xung quanh.
- Xử lý dữ liệu: Dữ liệu từ LIDAR được xử lý để tạo ra bức tranh chi tiết về môi trường.
- Ánh xạ đặc trưng: Các điểm đặc trưng trên bức tranh được sử dụng để xác định vị trí của Robot.
- Xây dựng bản đồ: Dữ liệu vị trí và chiều sâu được tích hợp để xây dựng bản đồ 2D hoặc 3D.
- Duy trì bản đồ: Bản đồ được liên tục cập nhật khi Robot di chuyển trong môi trường.

2.6.4 Quy trình tạo bản đồ:

1. Thu thập dữ liệu từ LIDAR:

- LIDAR được sử dụng để quét môi trường xung quanh Robot, thu thập dữ liệu về khoảng cách và hình dạng của các đối tượng.
- Dữ liệu LIDAR thường chứa các điểm 3D đại diện cho bề mặt và cấu trúc không gian xung quanh.

2. Tiền xử lý dữ liệu:

- Dữ liệu LIDAR thường cần được tiền xử lý để loại bỏ nhiễu và các điểm dữ liệu không chính xác.
- Các bước tiền xử lý có thể bao gồm lọc nhiễu, giảm độ phân giải, và điều chỉnh mức độ nhạy cảm của LIDAR.

3. Tích hợp dữ liệu vào thuật toán SLAM:

- Dữ liệu LIDAR được tích hợp vào thuật toán SLAM trong ROS hoặc các hệ thống khác để xác định vị trí của Robot và xây dựng bản đồ 3D đồng thời.
- Các gói phần mềm như Cartographer hoặc Hector SLAM trong ROS thường được sử dụng để thực hiện quá trình này.

4. Xây dựng bản đồ 2D:

- Dữ liệu từ LIDAR được sử dụng để đặt điểm trong không gian 2D, tạo nên bản đồ chi tiết về cấu trúc và hình dạng của môi trường.
- Bản đồ 2D được liên tục cập nhật khi Robot di chuyển, giúp theo dõi sự biến động trong môi trường.

5. Tối ưu và cải thiện bản đồ:

- Điều chỉnh thông số SLAM: Các thông số của thuật toán SLAM như độ chính xác, độ tin cậy, hoặc kích thước bước nhảy có thể được điều chỉnh để cải thiện chất lượng bản đồ.
- Lọc Nhiễu: Sử dụng các thuật toán lọc như Kalman hoặc Particle Filter để loại bỏ nhiễu từ dữ liệu LIDAR, giúp tạo ra bản đồ ổn định và chính xác hơn.

6. Kiểm tra và đánh giá:

- Bản đồ 2D được kiểm tra và đánh giá bằng cách so sánh với bản đồ thực tế hoặc bản đồ đã biết trước để đảm bảo độ chính xác và khả năng tự định vị của Robot.

7. Lặp lại quy trình (Loop Closure):

- Các kỹ thuật lặp lại quy trình giúp cải thiện độ chính xác bản đồ bằng cách nhận diện và sửa lỗi trong vị trí ước lượng của Robot khi quay lại các vị trí đã đi qua.

8. Bảo trì và cập nhật bản đồ:

- Bản đồ 2D được duy trì và cập nhật khi Robot di chuyển trong môi trường, đồng thời áp dụng các kỹ thuật như incremental mapping để giảm áp lực tính toán.

- Quy trình này tích hợp nhiều bước khác nhau, từ thu thập dữ liệu LIDAR đến xử lý và tối ưu hóa bản đồ 2D, đảm bảo rằng Robot có thể tự định vị và hiểu rõ môi trường xung quanh một cách chính xác và độ tin cậy.

2.6.5 *Hiển thị bản đồ trên Rviz*

Trong một hệ thống robot sử dụng ROS, dữ liệu từ robot thường được thu thập qua các cảm biến và gửi về máy tính thông qua giao tiếp mạng. Dữ liệu này có thể bao gồm thông tin từ cảm biến LIDAR, camera, hoặc các cảm biến khác. Để xem dữ liệu này, chúng ta sử dụng công cụ rviz, một ứng dụng trực quan hóa 3D trong ROS.

Rviz cho phép người dùng cấu hình các loại dữ liệu khác nhau để hiển thị, bao gồm nhưng không giới hạn ở bản đồ, đám mây điểm, và dữ liệu hình ảnh. Người dùng có thể tương tác với dữ liệu được hiển thị để phân tích và đánh giá hiệu suất cũng như hành vi của robot trong môi trường thực tế hoặc mô phỏng. Cấu hình rviz thường được lưu dưới dạng các hồ sơ, cho phép tái sử dụng và chia sẻ dễ dàng giữa các thành viên trong nhóm phát triển.

2.6.5.1 *Giao diện RViz:*

1. RViz trong ROS:

RViz là công cụ trực quan hóa mạnh mẽ trong ROS, giúp người dùng hiển thị và phân tích dữ liệu từ Robot bao gồm bản đồ và dữ liệu cảm biến.

2. Cách sử dụng RViz:

- **Khởi động RViz:** Người dùng có thể khởi động RViz thông qua terminal.
- **Chọn cấu hình hiển thị:** RViz cung cấp nhiều cấu hình mặc định cho việc hiển thị bản đồ, dữ liệu cảm biến và thông tin vị trí Robot.
- **Thêm các mục hiển thị:** Người dùng có thể thêm bản đồ, dữ liệu LIDAR, camera, và nhiều mục khác vào không gian làm việc của RViz.

3. Tương tác với RViz:

- **Zoom và pan:** Người dùng có thể thu phóng và di chuyển trong không gian 3D.
- **Chọn và theo dõi đối tượng:** Cho phép chọn và theo dõi Robot hoặc đối tượng trong thời gian thực.
- **Hiển thị thông tin chi tiết:** Cung cấp thông tin chi tiết về các đối tượng được chọn.

2.6.5.2 Tích hợp và đồng bộ:

1. Tích hợp dữ liệu từ SLAM và LIDAR:

- **Topic truyền dữ liệu:** Dữ liệu từ SLAM và LIDAR được gửi đến RViz qua các topic trong ROS như `"/map"` và `"/scan"`.

2. Đồng bộ hóa dữ liệu với RViz:

- **TF (transform frames):** RViz sử dụng TF để đồng bộ hóa các frame tham chiếu và hiển thị dữ liệu một cách chính xác.
- **Marker và mảng hiển thị:** Cung cấp biểu diễn trực quan cho các đối tượng và bản đồ.

3. Tương tác với dữ liệu:

- **Sử dụng interactive markers:** Cho phép người dùng tương tác trực tiếp với bản đồ và điều chỉnh vị trí đối tượng.
- **Phân tích dữ liệu:** Các topic trong ROS truyền dữ liệu giữa các nút và RViz, hỗ trợ phân tích dữ liệu trong môi trường đồng nhất và tương tác.

Kết luận: Quy trình hiển thị bản đồ trên Rviz trong ROS là công cụ hữu ích và mạnh mẽ, cung cấp cho người phát triển khả năng trực quan hóa và tương tác chi tiết với dữ liệu Robot. Tích hợp dữ liệu từ SLAM và LIDAR cùng với khả năng đồng bộ hóa dữ liệu qua TF, RViz tạo nên hệ thống hiển thị linh hoạt và chính xác. Người dùng có thể dễ dàng quản lý, phân tích và tối ưu hóa quy trình làm việc của Robot trong không gian 2D.

2.7 Kết luận chung

Hệ thống này tạo nên một Robot có khả năng nhận lệnh từ người dùng, xử lý dữ liệu môi trường qua cảm biến, và thực hiện các tác vụ thông qua cơ cấu chấp hành. Sự kết hợp của phần mềm và phần cứng này cho phép Robot hoạt động một cách linh hoạt trong nhiều ứng dụng khác nhau.

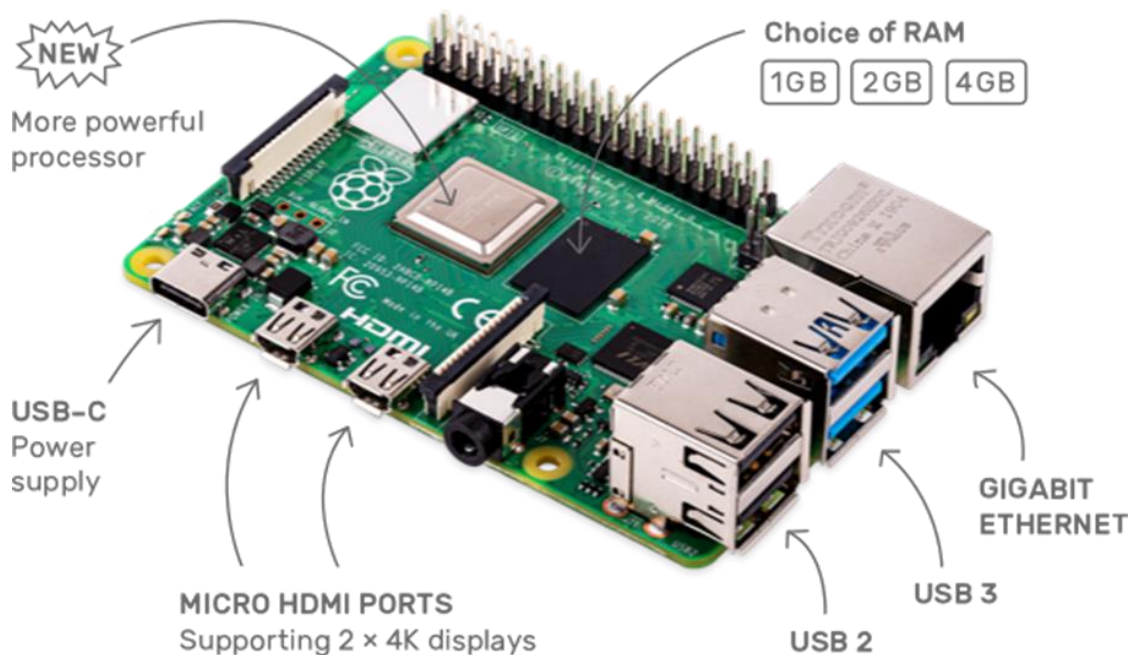
CHƯƠNG 3. LỰA CHỌN THIẾT BỊ ĐIỀU KHIỂN

3.1 Trung tâm xử lý

3.1.1 Bảng so sánh bo mạch nhúng

Đặc Điểm	Raspberry Pi 4	Odroid XU4	NVIDIA Jetson Nano	BeagleBone Black
CPU	Broadcom BCM2711, Quad-core Cortex-A72 @ 1.5 GHz	Samsung Exynos 5422, Octa-core (4×Cortex-A15 @ 2.0 GHz & 4×Cortex-A7 @ 1.4 GHz)	Quad-core ARM Cortex-A57 @ 1.43 GHz	AM335x 1GHz ARM® Cortex-A8
RAM	4GB LPDDR4	2GB LPDDR3	4GB LPDDR4	512MB DDR3
Khả năng kết nối	Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0	Gigabit Ethernet, Wi-Fi (via module), USB 3.0	Gigabit Ethernet, Wi-Fi (via module)	Ethernet, Wi-Fi (via module)
Hỗ trợ đồ họa	Broadcom VideoCore VI	Mali-T628 MP6 GPU	NVIDIA Maxwell GPU	PowerVR SGX530 GPU
Cổng I/O	GPIO, SPI, I2C, UART, USB 3.0, USB 2.0	GPIO, I2C, UART, SPI, USB 3.0, USB 2.0	GPIO, I2C, SPI, UART, USB 3.0	GPIO, I2C, SPI, UART, analog input
Giá cả (VNĐ)	1.800.000	1.500.000	5.200.000	1.800.000

Bảng 3.1 So sánh các bo mạch nhúng



Hình 3.1 Raspberry Pi 4 model B 4G

3.1.1.1 Lý do chọn lựa Raspberry Pi 4.

Sự cân bằng giữa hiệu suất và chi phí: Raspberry Pi và NVIDIA Jetson là hai trong số những lựa chọn tốt nhất cho các dự án sử dụng Ubuntu và ROS, các phiên bản Ubuntu Server và Ubuntu Core chuyên biệt cho Raspberry Pi, nhờ vào sự cân bằng giữa hiệu suất và khả năng tương thích. Raspberry Pi 4 là lựa chọn cho hầu hết các dự án nhờ vào sự linh hoạt và cộng đồng hỗ trợ lớn, trong khi NVIDIA Jetson phù hợp với các dự án cần khả năng xử lý hình ảnh và AI/ML và giá của NVIDIA Jetson cao hơn so với mặt bằng trung.

Tương thích tốt với cảm biến LIDAR: Raspberry Pi 4 có khả năng kết nối tốt với nhiều loại cảm biến, bao gồm cả LIDAR, nhờ vào cổng USB 3.0 và các tùy chọn I/O khác.

Khả năng kết nối mạng và giao tiếp: Raspberry Pi 4 cung cấp nhiều tùy chọn kết nối mạng, bao gồm Gigabit Ethernet, Wi-Fi 802.11ac và Bluetooth 5.0, hỗ trợ truyền tải dữ liệu và điều khiển từ xa.

3.1.1.2 Thông số kỹ thuật

Thông Số Kỹ Thuật	Chi Tiết
CPU	Broadcom BCM2711, Quad core Cortex-A72 (ARMv8) 1.5GHz
RAM	4GB LPDDR4-2400 SDRAM
Kết nối không dây	Wifi 2.4 GHz và 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE
Cổng kết nối	Gigabit Ethernet, 2 USB 3.0, 2 USB 2.0
GPIO	40 chân GPIO, tương thích với các phiên bản trước
Hỗ trợ hiển thị	2 cổng Micro HDMI, độ phân giải lên tới 4K
Cổng giao tiếp	MIPI DSI/ MIPI CSI/ AV 4 chân
Xử lý hình ảnh và video	H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode), OpenGL ES 3.0 graphics
Lưu trữ	Khe cắm Micro-SD
Nguồn điện	5V – 3A DC chuẩn USB-C
Hỗ trợ PoE	Có, cần có PoE HAT

Bảng 3.2 Thông số kỹ thuật của Raspberry Pi 4 Model B

Voltage	Pull	Funcio	Name	Physical	Name	Funcio	Pull	Voltage
3V3				1	2			5V
HIGH	UP	SDA1	GPIO2	3	4			5V
HIGH	UP	SCL1	GPIO3	5	6			GND
HIGH	UP	IN	GPIO4	7	8	GPIO14	IN	DOWN LOW
			GND	9	10	GPIO15	IN	DOWN HIGH
LOW	DOWN	IN	GPIO17	11	12	GPIO18	IN	DOWN LOW
LOW	DOWN	IN	GPIO27	13	14			GND
LOW	DOWN	IN	GPIO22	15	16	GPIO23	IN	DOWN LOW
			3V3	17	18	GPIO24	IN	DOWN LOW
LOW	DOWN	IN	GPIO10	19	20			GND
LOW	DOWN	IN	GPIO9	21	22	GPIO25	IN	DOWN LOW
LOW	DOWN	IN	GPIO11	23	24	GPIO8	IN	UP HIGH
			GND	25	26	GPIO7	IN	UP HIGH
HIGH	UP	IN	GPIO0	27	28	GPIO1	IN	UP HIGH
LOW	UP	IN	GPIO5	29	30			GND
HIGH	UP	OUT	GPIO6	31	32	GPIO12	IN	DOWN LOW
LOW	DOWN	IN	GPIO13	33	34			GND
LOW	DOWN	IN	GPIO19	35	36	GPIO16	IN	DOWN LOW
HIGH	DOWN	IN	GPIO26	37	38	GPIO20	IN	DOWN LOW
			GND	39	40	GPIO21	IN	DOWN HIGH

Hình 3.2 Sơ đồ chân Raspberry Pi 4

3.1.2 Các loại vi điều khiển

Đặc điểm	Arduino Uno R3	ESP32	ESP8266	STM32F103C8T6
CPU	ATmega328	Tensilica Xtensa LX6	Tensilica Xtensa L106	ARM Cortex-M3
Tốc độ xung nhịp	16 MHz	Up to 240 MHz	80 MHz	72 MHz
RAM	2 KB	520 KB	160 KB	20 KB
Bộ nhớ Flash	32 KB	Up to 16 MB	4 MB	64 KB
Kết nối mạng	Không	Wi-Fi, Bluetooth	Wi-Fi	Không

Bảng 3.3 So sánh các loại vi điều khiển

3.1.2.1 Lý do chọn lựa Arduino Uno R3 ATmega328

Dựa trên bảng đặc điểm của các bảng mạch điều khiển như Arduino Uno R3, ESP32, ESP8266, và STM32F103C8T6 trong hệ thống Robot tự hành sử dụng Raspberry Pi để xử lý kết nối mạng, Arduino Uno R3 được chọn làm bộ điều khiển cho các drive, đảm bảo không làm hỏng Raspberry Pi do chập điện khi kết nối với thiết bị ngoại vi. Arduino Uno R3 mang lại sự cân bằng giữa chức năng và sự đơn giản, không yêu cầu các chức năng kết nối mạng phức tạp, và nó hoàn toàn phù hợp để thực hiện các tác vụ điều khiển cơ bản trong Robot, bảo vệ hệ thống chính khỏi các rủi ro về điện.

Độ tin cậy và ổn định: Arduino Uno R3 sử dụng ATmega328, một microcontroller 8-bit từ Atmel. Làm việc ở tần số 16 MHz, cung cấp 32 KB bộ nhớ flash và 2 KB SRAM. Với việc sử dụng các linh kiện đơn giản và chắc chắn, Arduino Uno R3 ít phải đối mặt với các vấn đề như quá nhiệt hoặc quá tải. Arduino Uno được sử dụng rộng rãi trong giáo dục, nghiên cứu và thậm chí trong các dự án thương mại nhỏ. Điều này phản ánh độ tin cậy cao của nó trong các môi trường khác nhau.

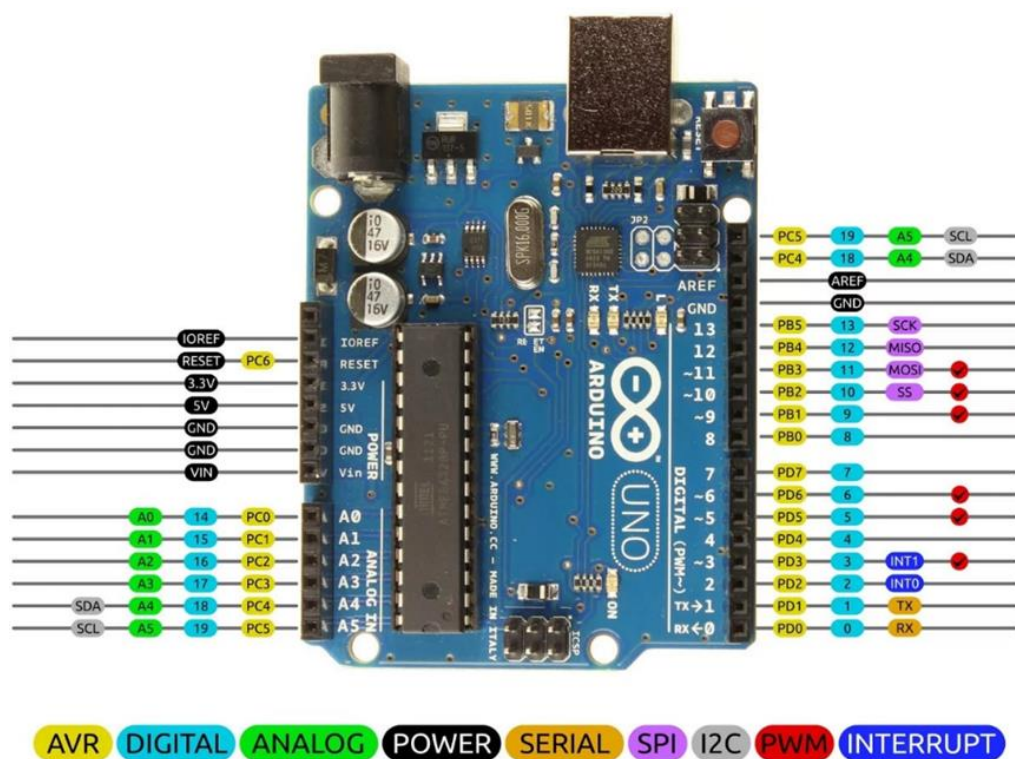
Giao tiếp đơn giản với Raspberry Pi: Arduino Uno có thể dễ dàng kết nối với Raspberry Pi thông qua cổng USB, tạo điều kiện cho giao tiếp đơn giản và hiệu quả.

Tương thích với ROS: Arduino Uno tương thích với ROS thông qua rosserial, cho phép trao đổi thông điệp ROS qua cổng serial.

Đễ dàng điều khiển động cơ: Arduino Uno có khả năng điều khiển động cơ hiệu quả thông qua các shield động cơ và các module liên quan.

Phát triển linh hoạt: Arduino Uno cho phép phát triển linh hoạt với khả năng thêm hoặc sửa đổi các thành phần một cách dễ dàng, điều này rất quan trọng trong quá trình thử nghiệm và cải tiến của dự án.

Vì những lý do trên, sử dụng Arduino Uno R3 ATmega328 cho hệ thống Robot phát thuốc trong bệnh viện. Arduino Uno R3 ATmega328 là một lựa chọn phù hợp và hiệu quả cho hệ thống.



Hình 3.3 Sơ đồ chân Arduino Uno R3 ATmega328

3.1.2.2 Thông số kỹ thuật

Thông Số Kỹ Thuật	Giá Trị
Vi điều khiển chính	ATmega328
Điện áp hoạt động	5VDC
Điện áp vào	7~12VDC
Điện áp vào giới hạn	6~20VDC
Số chân Digital	14 (6 chân PWM)
Số chân vào Analog	6
Dòng DC trên mỗi chân	40mA
Dòng DC trên chân 3.3V	50mA
Bộ nhớ Flash	32 KB (0.5KB dùng cho bootloader)

Bảng 3.4 Thông số kỹ thuật của Arduino Uno R3 ATmega328

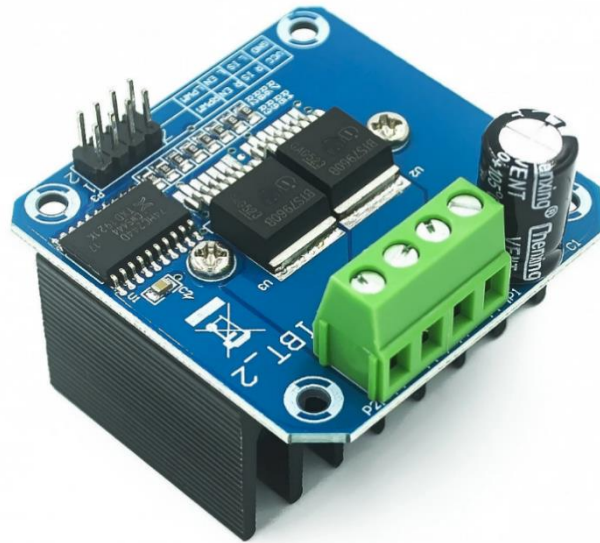
3.1.3 Khối điều khiển (Drive)

3.1.3.1 Các loại module điều khiển

Tính Năng/Model	DC BTS7960	DC L298N	TB6612FNG	L293D
Dòng Điện Tối Đa	43A	2A (4A peak)	1.2A (3.2A peak)	0.6A (1.2A peak)
Điện Áp Hoạt Động	5.5V - 27V	5V - 35V	2.5V - 13.5V	4.5V - 36V
Số Lượng Kênh	1	2	2	2
Dòng Điện Tối Đa	43A	2A (4A peak)	1.2A (3.2A peak)	0.6A (1.2A peak)
Tản Nhiệt	Hiệu quả cao	Trung bình	Tốt	Trung bình

Bảng 3.5 So sánh Drive điều khiển động cơ

3.1.3.2 Lý do chọn lựa DC BTS7960 43A



Hình 3.4 Module DC BTS7960 43A

Trong bảng so sánh, ta thấy có nhiều loại drive khác nhau bao gồm DC BTS7960, DC L298N, TB6612FNG, và L293D, mỗi loại có các thông số và ưu điểm riêng. Đối với ứng dụng của Robot phân phát thuốc trong bệnh viện, yêu cầu là sử dụng động cơ DC có khả năng chịu tải lớn, kiểm soát tốc độ và hướng quay một cách chính xác, cùng với khả năng bảo vệ chống lại quá tải dòng điện, quá nhiệt và ngắn mạch. Phù hợp với những tiêu chí này, nhóm đã quyết định chọn mạch điều khiển động cơ DC BTS7960 43A để kết hợp với Arduino, cho phép kiểm soát hiệu quả các động cơ DC Servo có dòng điện cao.

Một số ưu điểm của mạch điều khiển động cơ DC BTS7960 43A là:

- Điện áp cung cấp: 6 ~ 27 VDC, phù hợp với nguồn điện của hệ thống Robot.
- Tần số điều khiển tối đa 25KHz, cho phép điều khiển tốc độ và chiều quay của động cơ một cách chính xác và linh hoạt bằng cách cấp xung PWM.
- Tự động ngắt khi áp dưới 5.5V, và mở lại sau khi áp lớn hơn 5.5V, giúp bảo vệ mạch và động cơ khỏi sự cố sụt áp.

- Bảo vệ quá nhiệt: BTS7960 bảo vệ chống quá nhiệt bằng cảm biến nhiệt tích hợp bên trong, giúp ngắt mạch khi nhiệt độ vượt quá ngưỡng cho phép.
- Bảo vệ quá dòng: BTS7960 có chức năng giới hạn dòng bằng cách đo dòng qua một điện trở đo dòng và so sánh với một ngưỡng cài đặt, giúp ngắt mạch khi dòng vượt quá ngưỡng cho phép.
- Bảo vệ quá áp: BTS7960 có chức năng bảo vệ quá áp bằng cách đo điện áp cấp và so sánh với một ngưỡng cài đặt, ngắt mạch khi điện áp vượt quá ngưỡng cho phép.
- Bảo vệ ngắn mạch: BTS7960 có chức năng bảo vệ ngắn mạch bằng cách đo điện áp trên các MOSFET và so sánh với một ngưỡng cài đặt, giúp ngắt mạch khi có ngắn mạch xảy ra.

Như vậy, mạch điều khiển động cơ DC BTS7960 43A là một lựa chọn phù hợp cho hệ thống Robot phát thuốc trong bệnh viện, vì nó có khả năng điều khiển động cơ DC có dòng cao, điều khiển tốc độ và chiều quay chính xác, bảo vệ chống quá dòng, quá nhiệt, quá áp và ngắn mạch.

3.1.3.3 Thông số kỹ thuật

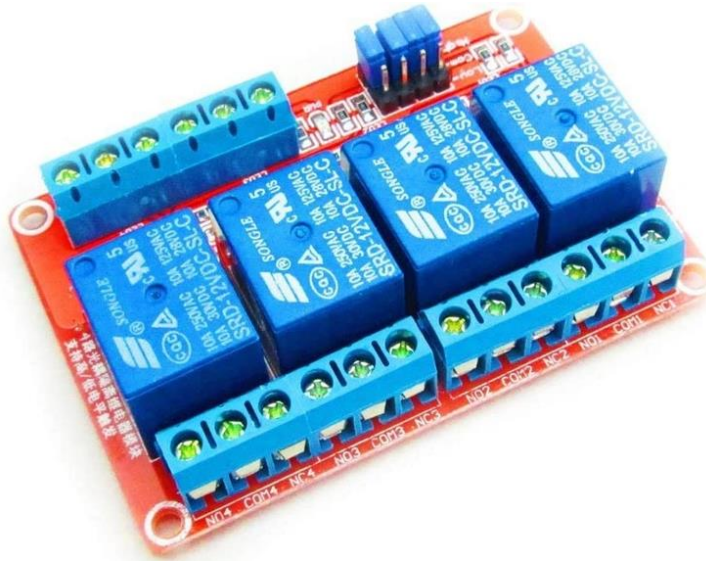
Thông Số Kỹ Thuật	Giá Trị
Điện áp đầu vào	6~27V
Dòng điện tải mạch	43A
Tín hiệu logic điều khiển	3.3 ~ 5V
Tần số điều khiển tối đa	25KHz
Bảo vệ quá nhiệt	Có, bằng cảm biến nhiệt tích hợp, tự ngắt khi quá nhiệt
Kích thước	40 x 50 x 12mm
Sơ đồ chân	VCC, GND, R_EN, L_EN, RPWM, LPWM, R_IS, L_IS

Bảng 3.6 Thông số kỹ thuật của DC BTS7960 43A

3.1.3.4 Nguyên lý hoạt động

- Mạch điều khiển động cơ DC BTS7960 43A là một mạch cầu H dòng cao tích hợp đầy đủ cho các ứng dụng truyền động động cơ cần sử dụng dòng cao. Mạch này chứa một MOSFET bên cao kênh p và một MOSFET bên thấp kênh n, với một vi mạch trình điều khiển tích hợp trong một gói. Việc giao tiếp với vi điều khiển được thực hiện dễ dàng nhờ vi mạch điều khiển tích hợp có các đầu vào mức logic, bảo vệ chống quá nhiệt, quá áp, quá dòng và ngắn mạch.
- Tự động shutdown khi điện áp thấp: Để tránh điều khiển động cơ ở mức điện áp thấp thiết bị sẽ tự shutdown. Nếu điện áp $< 5.5V$, mạch điều khiển động cơ DC BTS7960 sẽ tự ngắt điện và sẽ mở lại sau khi điện áp $> 5.5V$.

3.1.4 MODULE Relay 4 Kênh



Hình 3.5 MODULE Relay 12VDC 4 Kênh

Mô-đun role có thể chịu được dòng điện và điện áp cao, cho phép nó điều khiển các thiết bị có công suất lớn mà không làm hỏng các linh kiện điện tử nhạy cảm. Role cung cấp sự tách biệt giữa mạch điều khiển (thấp công suất) và mạch tải (cao công suất). Điều này giúp bảo vệ mạch điều khiển khỏi các tác động đột ngột từ mạch tải.

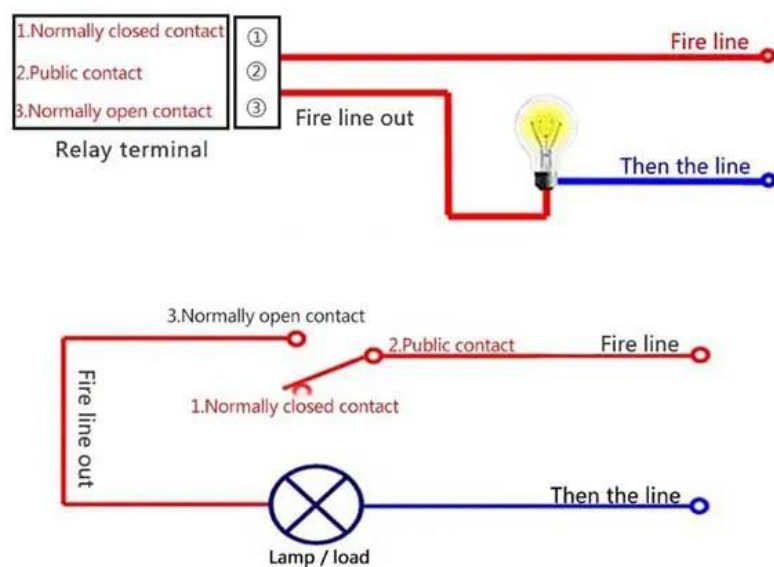
3.1.4.1 Thông số kỹ thuật

Thông số kỹ thuật	Giá Trị
Tải tối đa	AC 250V/10A, DC 30V/10A
Cách ly SMD OPTOCOUPLER	kích hoạt dòng điện 5mA
Điện áp mô-đun	12V
Thiết lập kích hoạt	Có thể thiết lập cao hoặc thấp bởi jumper
Thiết kế chịu lỗi	Ngay cả khi đường điều khiển bị hỏng, role sẽ không hoạt động
Đèn báo trạng thái role	4 đèn màu đỏ
Kích thước mô-đun	73mm x 50mm x 18,5mm (L x W x H)
Lỗ bắt bu lông	Có bốn lỗ, 3.1mm, khoảng cách 67mm x 44.5mm
DC +	Nguồn điện tích cực (có lựa chọn 5V, 9V, 12V và 24V)
DC-	Kết nối nguồn âm
IN1-IN4	Mỗi cách có thể cao hoặc thấp để điều khiển role
NO1 - NO4, COM1 - COM4, NC1 - NC4	Giao diện role thường mở, thường đóng và chung
S1-S4	Chuyển đổi lựa chọn kích hoạt đường cao và thấp cho mỗi cách

Bảng 3.7 Thông số kỹ thuật module relay 12vdc bốn kênh

3.1.4.2 Nguyên lý hoạt động

Module relay 12vdc bốn kênh sử dụng nguyên lý relay để đóng ngắt các mạch điện. Relay là một thiết bị điện từ, có một cuộn dây và một tiếp điểm. Khi có dòng điện chạy qua cuộn dây, nó tạo ra từ trường, kéo tiếp điểm đóng hoặc mở mạch điện. Module này có bốn relay, mỗi relay có ba tiếp điểm: COM (chung), NO (thường mở) và NC (thường đóng). Khi không có dòng điện kích, tiếp điểm COM sẽ nối với NC, khi có dòng điện kích, tiếp điểm COM sẽ nối với NO.



Hình 3.6 Sơ đồ nguyên lý hoạt động

3.2 Khôi xây dựng map

Các loại LIDAR có thể sử dụng trong đồ án

Tính Năng/Model	RPLIDAR A1M8 360	Hokuyo UST-10/20LX	Velodyne LIDAR Sensors	Sick LMS111	Leica BLK360
Phạm Vi Đo Lường	6 - 12 m	10 - 20 m	100 m	20 m	60 m
Độ Phân Giải	Cao	Cao	Rất cao	Cao	Rất cao
Tốc Độ Quét	5.5 - 10 Hz	40 Hz	20 Hz	50 Hz	Thấp
Ứng Dụng Phổ Biến	Robot di động	Robot di động	Xe tự lái	Công nghiệp	Quét 3D
Khoảng Giá USD	106	2,070	8,000	6,501	26,550

Bảng 3.8 So sánh các loại LIDAR



Hình 3.7 RPLIDAR A1M8 360

3.2.1 Lý do chọn lựa RPLIDAR A1M8 360

1. Chi phí hiệu quả

RPLIDAR A1M8 360 có mức giá tốt, phù hợp với các dự án có ngân sách hạn chế hoặc đang trong giai đoạn thử nghiệm.

2. Tích hợp dễ dàng với ROS

Thiết bị dễ dàng tích hợp với ROS, hỗ trợ tăng tốc quá trình phát triển và triển khai các ứng dụng Robot. Cảm biến sử dụng giao tiếp UART, dễ dàng kết nối với vi điều khiển, máy tính nhúng, hay máy tính qua mạch chuyển USB-UART.

3. Độ chính xác và độ phân giải

RPLIDAR A1M8 cung cấp dữ liệu quét chính xác, rất quan trọng cho các ứng dụng như điều hướng và tránh vật cản. Cảm biến có tần số lấy mẫu cao, từ 4000 đến 8000 lần/giây, cho phép Robot có được dữ liệu chính xác và nhanh chóng. Cảm biến có tần số quét có thể từ 2 đến 10Hz, cho phép Robot thích ứng với các điều kiện môi trường khác nhau.

4. Kích thước và môi trường hoạt động

Thiết bị nhỏ gọn và nhẹ, thuận lợi cho việc tích hợp vào nhiều loại Robot, đặc biệt là Robot di động. Cảm biến có thiết kế OPTMAG, giúp kéo dài tuổi thọ của cảm biến. Cảm biến cũng có khả năng hoạt động tốt trong các môi trường trong nhà và ngoài trời không có ánh sáng mặt trời.

5. Phạm vi đo lường.

Phạm vi đo lường từ 6 đến 12 mét phù hợp với nhiều môi trường khác nhau, từ trong nhà đến một số ứng dụng ngoài trời. Cảm biến có khả năng quét 360° môi trường xung quanh với khoảng cách từ 0.15 đến 12 mét, phù hợp cho việc xây dựng bản đồ, làm SLAM.

3.2.2 Thông số kỹ thuật

Thông Số	Giá Trị
Model	RPLIDAR A1M8
Điện áp hệ thống	5VDC
Dòng hệ thống	100mA
Công suất tiêu thụ	0.5W
Đầu ra	UART Serial (mức điện áp 3.3V)
Phạm vi đo lường	0.15~12m
Tần số lấy mẫu	8K
Tốc độ quay	5.5Hz
Độ phân giải góc	$\leq 1^\circ$
Phạm vi góc	360°
Độ phân giải phạm vi	$\leq 1\%$ ($\leq 12\text{m}$), $\leq 2\%$ ($12 \sim 16\text{m}$)
Độ chính xác	1% ($\leq 3\text{m}$), 2% (35m), 2.5% (512m)

Bảng 3.9 Thông số kỹ thuật của LIDAR RPLIDAR A1M8

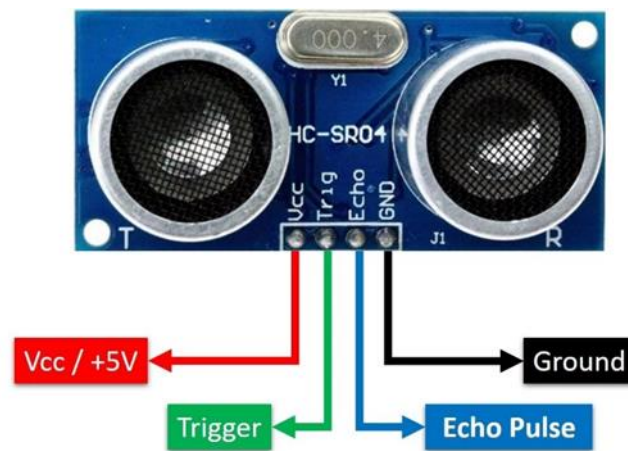
3.2.3 Nguyên lý hoạt động

LIDAR RPLIDAR A1M8 360 sử dụng nguyên lý tam giác laser để đo khoảng cách. LIDAR bao gồm một đầu phát laser, một đầu thu ánh sáng và một động cơ quay. Đầu phát laser phát ra một xung laser có bước sóng hồng ngoại và công suất thấp. Xung laser này sẽ chiếu vào một vật thể nào đó trong môi trường và phản xạ lại. Đầu thu ánh sáng sẽ nhận được ánh sáng phản xạ và chuyển đổi thành tín hiệu điện. Từ tín hiệu điện này, LIDAR có thể tính được khoảng cách từ LIDAR đến vật thể bằng cách sử dụng công thức $v = d/t$, trong đó v là tốc độ ánh sáng, d là khoảng cách cần tìm và t là thời gian từ khi phát ra xung laser

đến khi nhận được ánh sáng phản xạ. Đồng thời, LIDAR cũng có thể xác định được góc của vật thể so với LIDAR bằng cách sử dụng một bộ mã hóa quang để đo góc quay của động cơ. Như vậy, LIDAR có thể tạo ra một điểm 2D gồm khoảng cách và góc của vật thể đối với LIDAR. Bằng cách quét liên tục các xung laser xung quanh 360 độ, LIDAR có thể tạo ra một đám mây điểm 2D về môi trường xung quanh.

3.3 Cơ cấu chấp hành khối chuyển động:

3.3.1 Cảm biến siêu âm



Hình 3.8 Cảm biến siêu âm HC-SR04

3.3.1.1 Lý do chọn lựa HC-SR04

Mặc dù có LIDAR và map server để giúp Robot AMR nhận biết được vật cản nhưng để tăng độ tin cậy cho hệ thống khi xe di chuyển vào góc hẹp hoặc có vật cản động ngoài tầm quét của LIDAR thì HC-SR04 là một sự lựa chọn tốt.

3.3.1.2 Thông số kỹ thuật

- HC-SR04 có 4 chân là Vcc, Trig, Echo và GND.
- Cảm biến có thể đo khoảng cách từ 2cm đến 450cm với độ chính xác cao.
- Hoạt động với điện áp 5V DC, dòng hoạt động nhỏ hơn 2mA, góc đo được bao phủ nhỏ hơn 15 độ và tần số hoạt động là 40Hz.

3.3.1.3 Nguyên lý hoạt động

Phát tín hiệu siêu âm: HC-SR04 phát ra một xung siêu âm. Điều này được thực hiện bằng cách sử dụng một bộ phận phát siêu âm, thường là một loại loa nhỏ, có khả năng tạo ra sóng âm ở tần số cao (vượt quá ngưỡng nghe của con người, thường là khoảng 40 kHz).

Phản hồi tín hiệu: Sau khi phát sóng siêu âm, cảm biến chuyển sang chế độ nhận để chờ đợi tín hiệu phản xạ trở lại. Khi sóng siêu âm gặp một vật thể, nó sẽ được phản xạ trở lại cảm biến.

Tính toán khoảng cách: Cảm biến đo thời gian từ khi xung siêu âm được phát ra cho đến khi nó nhận được tín hiệu phản hồi. Khoảng cách đến vật thể được tính bằng cách sử dụng tốc độ truyền âm trong không khí (khoảng 343 mét/giây) và thời gian phản hồi.

Giao tiếp với bộ vi xử lý: HC-SR04 thường có 4 chân: VCC (điện áp cấp), Trig (để kích hoạt phát xung siêu âm), Echo (để nhận tín hiệu phản hồi), và GND (mát). Bộ vi xử lý hoặc vi điều khiển sẽ gửi một tín hiệu đến chân Trig để bắt đầu một lần đo lường và đọc tín hiệu từ chân Echo để tính toán khoảng cách.

3.3.2 Lý do chọn lựa DC Servo JGB37-545 (12V/110RPM)



Hình 3.9 DC Servo JGB37-545 (12V/110RPM)

Tính ổn định và độ tin cậy: Động cơ DC servo JGB37-545 được thiết kế để có tính ổn định cao và độ tin cậy trong quá trình hoạt động. Điều này đảm bảo rằng động cơ sẽ hoạt động một cách đáng tin cậy và không gây ra các vấn đề kỹ thuật không mong muốn.

Tốc độ và công suất: Động cơ DC servo JGB37-545 có tốc độ 110 vòng/phút và dòng điện định mức 3A. Điều này cho phép nó cung cấp đủ công suất và tốc độ để thực hiện các nhiệm vụ điều khiển và di chuyển của Robot.

Điều chỉnh tốc độ: Động cơ này được điều khiển bằng PWM (Pulse Width Modulation), cho phép điều chỉnh tốc độ của động cơ một cách linh hoạt.

Động cơ có encoder: Động cơ này được trang bị encoder, cho phép đọc vị trí của động cơ. Điều này hữu ích trong việc xác định và điều khiển chính xác vị trí của Robot.

3.3.2.1 Thông số kỹ thuật

Thông Số	Giá Trị	Chú Thích
Điện áp sử dụng	12VDC	-
Encoder	Cảm biến từ trường Hall	Có 2 kênh AB
Điện áp cấp cho Encoder	3.3~5VDC	Cấp quá áp hỏng Encoder
Loại động cơ	12VDC 110RPM	-
Dòng không tải	200mA	-
Dòng chịu đựng tối đa	3A	Khi có tải
Tốc độ không tải	37RPM	37 vòng/phút
Tốc độ chịu đựng tối đa	30RPM	30 vòng/phút khi có tải
Moment định mức	32KG.CM	Lực kéo tối ưu
Moment tối đa	90KG.CM	Lực kéo tối đa có thể đạt được
Số xung Encoder/vòng quay	1848 xung	Số xung mỗi kênh trên một vòng quay của trục chính

Bảng 3.10 Thông số kỹ thuật DC Servo JGB37-545

3.3.2.2 Nguyên lý hoạt động

- Cảm biến từ trường Hall, có 2 kênh AB lệch nhau giúp xác định chiều quay và vận tốc của động cơ, đĩa Encoder trả ra 11 xung/1 kênh/ 1 vòng (nếu đo tín hiệu đồng thời của cả hai kênh sẽ thu được tổng 22 xung / 1 vòng quay của Encoder).
- Cách tính số xung của mỗi kênh trên 1 vòng quay của trục chính động cơ = Tỉ số truyền x số xung của Encoder, ví dụ tỉ số 150:1 thì số xung Encoder trả ra cho 1 vòng quay của trục chính động cơ sẽ là $11 \times 150 = 1650$ xung / 1 kênh.

3.3.3 Đèn báo AD16-22DS 12VDC/AC



Hình 3.10 AD16-22DS 12VDC/AC

Trong sơ đồ hệ thống Robot, đèn giao thông trong khối chuyển động có tác dụng thông báo trạng thái hoạt động của xe cho mọi người xung quanh. Thông qua việc sử dụng các tín hiệu màu sắc giống như đèn giao thông thông thường, đèn này giúp chỉ dẫn cho người điều khiển và những người khác trong cùng môi trường biết được xe đang dừng lại hoặc đang di chuyển, từ đó tăng cường an toàn và hiệu quả trong vận hành.

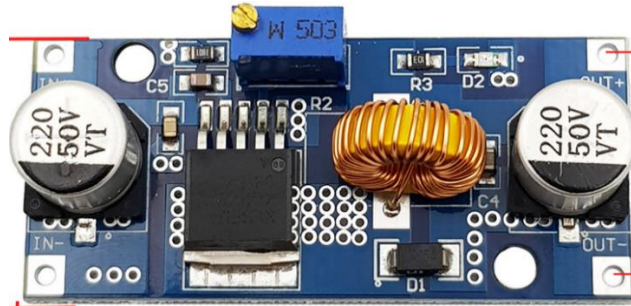
3.3.3.1 Thông số kỹ thuật

Thông Số	: Giá Trị
Dòng điện làm việc	: < 20mA
Độ sáng	: $\geq 100 \text{ cd/m}^2$
Đường kính lỗ lắp đặt	: 22 mm
Đường kính đèn báo	: 28.3 mm
Tổng chiều dài	: 50.3 mm
Tuổi thọ	: 30.000 giờ

3.4 Khả năng lượng

3.4.1 Mạch Hạ Áp DC XL4015

Arduino là một nền tảng vi điều khiển phổ biến, được sử dụng rộng rãi trong nhiều dự án điện tử và Robot. Đối với mô-đun này, điện áp đầu vào an toàn và ổn định là quan trọng để đảm bảo hoạt động hiệu quả và an toàn.



Hình 3.11 Mạch Hạ Áp DC XL4015

3.4.1.1 Lý do chọn lựa.

- Tối Ưu Hóa Hiệu Suất

Arduino hoạt động tốt nhất với nguồn điện từ 7V đến 12V. Tuy nhiên, việc sử dụng ở mức điện áp gần giới hạn trên (12V) có thể dẫn đến sự nóng lên của bộ điều chỉnh điện áp trên bo mạch. Điện áp 9V cung cấp một giải pháp cân bằng, giúp giảm thiểu nhiệt độ hoạt động mà vẫn đảm bảo đủ năng lượng cho hệ thống.

- Tăng Độ Tin Cậy

Sử dụng điện áp 9V giúp giảm rủi ro quá tải và hỏng hóc, đặc biệt trong các ứng dụng liên tục hoặc lâu dài như trong Robot AMR. Điều này tăng cường độ tin cậy và giảm thiểu cần bảo trì thường xuyên.

- Tính Tương Thích

Mạch hạ áp XL4015 cho phép điều chỉnh điện áp đầu ra một cách chính xác, đảm bảo tương thích tốt với nhu cầu năng lượng của Arduino.

- Bảo Vệ Hệ Thống

Mạch hạ áp DC XL4015 giúp bảo vệ Arduino khỏi những tác động tiêu cực của sự dao động điện áp, chập mạch, đặc biệt trong môi trường có nguồn điện không ổn định.

3.4.1.2 Thông số kỹ thuật

Thông Số	Giá Trị
IC chính	XL4015
Điện áp đầu vào	8~36VDC
Điện áp đầu ra	1.25~32VDC
Dòng đầu ra tối đa	5A
Hiệu suất	96%
Tần số xung	180KHz
Tích hợp Mosfet	Đóng ngắt tần số cao
Maximum Duty Cycle	100%
Minimum Drop Out	0.3VDC
Nhiệt độ làm việc	-40~125 độ C
Trọng lượng	25g
Kích thước	67.1 x 26.2 x 15mm

Bảng 3.11 Thông số kỹ thuật mạch hạ áp DC XL4015

3.4.2 Mạch tăng áp 12-220V 150W

Việc cung cấp nguồn điện ổn định và an toàn cho các thành phần chính như Raspberry Pi 4 là cực kỳ quan trọng. Raspberry Pi 4 yêu cầu một nguồn điện 5.1V – 3A để hoạt động hiệu quả. Trong trường hợp này, sử dụng mạch tăng áp 12-220V 150W để chuyển đổi điện áp từ 12V DC lên 220V AC, sau đó sử dụng nguồn chính hãng của Raspberry Pi 4 để cung cấp điện áp phù hợp tăng độ tin cậy đảm bảo điện áp cho trung tâm xử lý.



Hình 3.12 Mạch tăng áp 12-220V 150W

3.4.2.1 Lý do chọn lựa.

1. Phù Hợp với Nhu Cầu Điện Áp và Dòng Điện:

Mạch tăng áp 12-220V 150W cung cấp giải pháp hiệu quả để chuyển đổi điện áp từ 12V DC lên 220V AC, mức điện áp phổ biến cho các thiết bị gia dụng. Điều này giúp tạo ra nguồn điện ổn định và an toàn cho Raspberry Pi, đặc biệt quan trọng khi không có nguồn cấp 5VDC sẵn có.

2. Tính Di Động và Linh Hoạt:

Sử dụng mạch tăng áp giúp Raspberry Pi trở nên linh hoạt và di động hơn, cho phép sử dụng trong nhiều môi trường khác nhau, từ nhà ở đến các ứng dụng ngoại vi. Điều này rất hữu ích cho các dự án cần di chuyển hoặc khi làm việc tại những khu vực không có nguồn điện ổn định.

3. Độ An Toàn và Độ Tin Cậy:

Mạch tăng áp này được thiết kế để đảm bảo an toàn với các tính năng như chống quá tải và bảo vệ quá áp. Điều này quan trọng để bảo vệ Raspberry Pi khỏi các sự cố điện có thể xảy ra, như sự cố quá tải điện áp hoặc ngắn mạch. Hơn nữa việc sử dụng Official Raspberry Pi Power Supply 5.1VDC 3A USB-C làm nguồn cấp tăng độ tin cậy vì đây là sản phẩm chính hãng thiết kế dành riêng cho Raspberry Pi 4B.

3.4.2.2 Thông số kỹ thuật

Thông số	Giá Trị
Điện áp vào	8-15VDC
Điện áp ra	180-220VAC, có thể điều chỉnh
Công suất	Liên tục 70W, 100W
Tần số ra	50-60HZ, có thể điều chỉnh
Dạng sóng đầu ra	Xung vuông

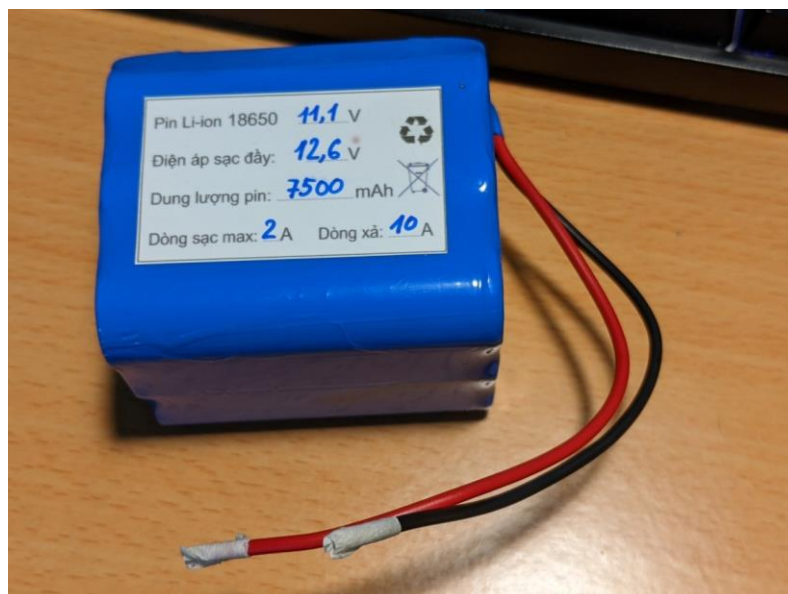
Bảng 3.12 Thông số kỹ thuật mạch tăng áp 12-220V 150W

3.4.3 Lựa chọn pin

Nguồn năng lượng	Ưu điểm	Nhược điểm	Thích hợp cho AMR
Pin Li-ion	Dung lượng cao Tuổi thọ dài Hiệu suất ổn định	Giá thành cao	Có
Pin NiMH	An toàn hơn Li-ion Ít ảnh hưởng môi trường	Dung lượng thấp hơn Nặng hơn	Không
Pin Lead-Acid (Ắc Quy)	Chi phí thấp Dễ tái chế, bền bỉ	Nặng, tuổi thọ ngắn Dung lượng thấp	Có
Pin NiCad	Chịu được xả sâu Độ bền cao	Hiệu suất thấp Tác động môi trường	Không
Supercapacitors	Sạc nhanh Tuổi thọ xả/sạc cao	Dung lượng thấp Rất đắt	Không

Bảng 3.13 So sánh nguồn năng lượng

3.4.3.1 Lý do Lựa Chọn Pin Li-ion



Hình 3.13 Pin Li-ion

Xét từ bảng 3.5 việc chọn pin Li-ion làm nguồn năng lượng cho AMR là lựa chọn phù hợp nhất vì chúng cung cấp dung lượng cao và độ ổn định lớn, rất quan trọng đối với hoạt động liên tục của Robot. Pin Li-ion cũng có tỷ lệ năng lượng-trọng lượng tốt, làm cho chúng trở thành lựa chọn tối ưu cho các ứng dụng cần giảm trọng lượng và tối đa hóa thời gian hoạt động. Mặc dù giá thành có thể cao hơn và có một số rủi ro liên quan đến an toàn, nhưng với công nghệ hiện đại, những rủi ro này có thể được quản lý hiệu quả, làm cho pin Li-ion trở thành lựa chọn phổ biến cho các hệ thống Robot hiện đại.

3.4.3.2 Một số ưu điểm của pin Li-ion

- Mật độ năng lượng cao: Pin Li-ion cung cấp mật độ năng lượng cao, cho phép Robot hoạt động lâu hơn trước khi cần sạc lại. Điều này quan trọng đối với hiệu suất liên tục của Robot trong môi trường bệnh viện.
- Kích thước và trọng lượng: So với ắc quy, pin Li-ion nhẹ hơn và nhỏ gọn hơn. Điều này làm giảm tổng trọng lượng của Robot, giúp dễ dàng di chuyển và ít tốn năng lượng hơn.
- Tuổi thọ và độ bền: Pin Li-ion có tuổi thọ chu kỳ sạc/xả dài hơn so với ắc quy. Điều này giúp giảm chi phí thay thế và bảo trì pin trong thời gian dài.
- Tự xả thấp: Pin Li-ion có tỷ lệ tự xả thấp, giúp bảo quản năng lượng tốt hơn trong các giai đoạn không hoạt động.
- Không yêu cầu bảo dưỡng: Pin Li-ion không yêu cầu bảo dưỡng định kỳ, trong khi ắc quy đôi khi cần được bảo dưỡng, như kiểm tra mức nước cất.
- Tác động môi trường: Pin Li-ion ít tác động tiêu cực đến môi trường hơn so với ắc quy, đặc biệt là ắc quy chì-axit.

3.4.3.3 Kết luận

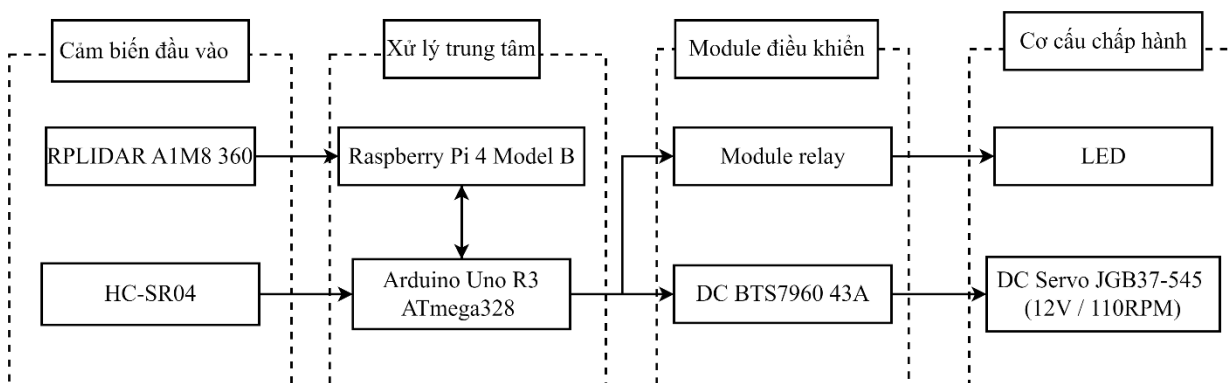
Chọn pin Li-ion có điện áp là 12V và dung lượng là 7500mAh.

$$W = U * I = 12V * 7.5Ah = 90Wh$$

Như vậy, năng lượng của pin chọn là 90Wh.

3.5 Sơ đồ chi tiết thiết bị trong hệ thống

Dựa vào phân chọn thiết bị ở các mục trên ta có sơ đồ kết nối các thiết bị trong hệ thống:



Hình 3.14 Sơ đồ chi tiết thiết bị trong hệ thống

CHƯƠNG 4. HỆ THỐNG ĐIỆN VÀ THUẬT TOÁN LẬP TRÌNH ĐIỀU KHIỂN

4.1 Hệ thống điện

4.1.1 Tính toán dự tính lượng điện cho hệ thống

Bảng dưới đây tổng hợp thông tin chi tiết về các thiết bị và linh kiện điện tử được sử dụng trong các hệ thống Robot. Bảng này bao gồm các thông số kỹ thuật cơ bản như số lượng (SL), điện áp hoạt động (Vdc), dòng điện, mô-men xoắn (Moment) động cơ và hệ thống, và tổng năng lượng tiêu thụ tính.

Thiết bị	SL	Điện áp (Vdc)	Dòng điện (A)	Moment (KG.CM)		Năng lượng (W)	
				max	HT	max	HT
Raspberry Pi 4	1	5	3			7,28	
RPLIDAR A1M8	1	5	0.1				0.5
Arduino Uno	1	9	0.2				1,8
ESP32	1	5	0,79				3,95
LED AD16 DC	3	12	0.02				0,86
HC-SR04	3	5	0.015				0,225
JGB37-545 (12V/110RPM)	2	12	3	90	52.51	72	38,6
GA25-370 (12V/60RPM)	3	12	2	12,5	12	24	23,04

Bảng 4.1 Tính toán dự tính lượng điện cho toàn bộ hệ thống

Tổng điện áp dự tính của hệ thống = **76,255 Wh**

4.1.2 Thời gian hoạt động của Robot

Với pin chọn có năng lượng là **90Wh**, hệ thống có thể hoạt động được trong khoảng **1 giờ 10 phút** với năng lượng tiêu thụ là **76,255 Wh**

4.2 Sơ đồ nguyên lí

4.2.1 Khối năng lượng

Module	Vin	Vout	Thiết bị
Pin Li-ion	12,6V/2A	12,6V/7,5A	Channel relay A DC BTS7960 43A
Mạch hạ áp DC XL4015	12,6V/7,5A	9V	Arduino Uno R3
Mạch tăng áp 12-220V 150W	12,6V/7,5A	220V	R/Pi 4 Power Supply
Raspberry Pi 4 Power Supply	220V	5V/3A	Raspberry Pi 4

Bảng 4.2 Các mức nguồn năng lượng cho hệ thống

4.2.2 Trung tâm xử lí

Thành Phần kết nối với Raspberry Pi 4	Loại kết nối	Mục đích truyền
LIDAR A1M8	Micro đến USB	Truyền dữ liệu vật đã quét
Arduino Uno R3	USB Type A-B	Truyền dữ điều khiển

Bảng 4.3 Sơ đồ chân kết nối thiết bị ngoại vi với Raspberry Pi 4

Arduino Uno R3	Cảm biến siêu âm 1	Cảm biến siêu âm 2
5V	VCC	VCC
GND	GND	GND
A0	Trig	-
A1	Echo	-
A2	-	Trig
A3	-	Echo

Bảng 4.4 Sơ đồ chân kết nối vi điều khiển với cảm biến

Chân Arduino Uno R3	DC BTS7960 43A B1	DC BTS7960 43A B2
5V	VCC	VCC
GND	GND	GND
D5	RPWM	-
D6	LPWM	-
D7	L_EN	-
D7	R_EN	-
D8	-	L_EN
D8	-	R_EN
D9	-	RPWM
D10	-	LPWM

Bảng 4.5 Sơ đồ chân kết nối vi điều khiển với mạch điều khiển

Arduino Uno R3	Channel relay A
D4_A	IN4
D11_A	IN3

Bảng 4.6 Sơ đồ chân kết nối vi điều khiển với relay

4.2.3 Module điều khiển

Pin Li-ion	LED	Channel relay A
12,6V	V+	COM4/COM3
GND	V-	NC4/NC3

Bảng 4.7 Sơ đồ chân kết nối relay với LED

DC BTS7960 43A	JGB37-545
VCC	M+
GND	M-

Bảng 4.8 Sơ đồ chân kết nối mạch điều khiển với động cơ

4.2.4 Cảm biến

RPLIDAR A1M8	Adapter micro - USB cable
VCC	VCC
RX	TX
TX	RX
GND	GND
VCC	VCC
HIGH	HIGH

Bảng 4.9 Sơ đồ chân kết nối RPLIDAR A1M8 với mạch chuyển đổi micro – USB

Cảm biến siêu âm 1	Cảm biến siêu âm 2	Arduino Uno R3
VCC	VCC	5V
GND	GND	GND
Trig	-	A0
Echo	-	A1
-	Trig	A2
-	Echo	A3

Bảng 4.10 Sơ đồ chân kết nối cảm biến với vi điều khiển

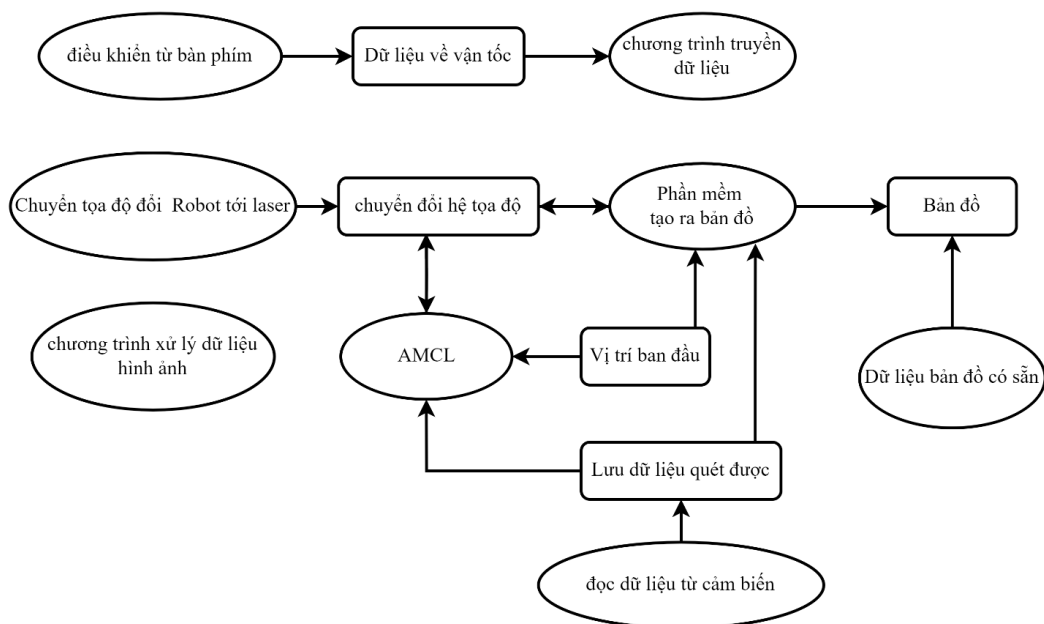
4.3 Xây dựng hệ thống Robot

4.3.1 Hệ thống robot đã xây dựng được

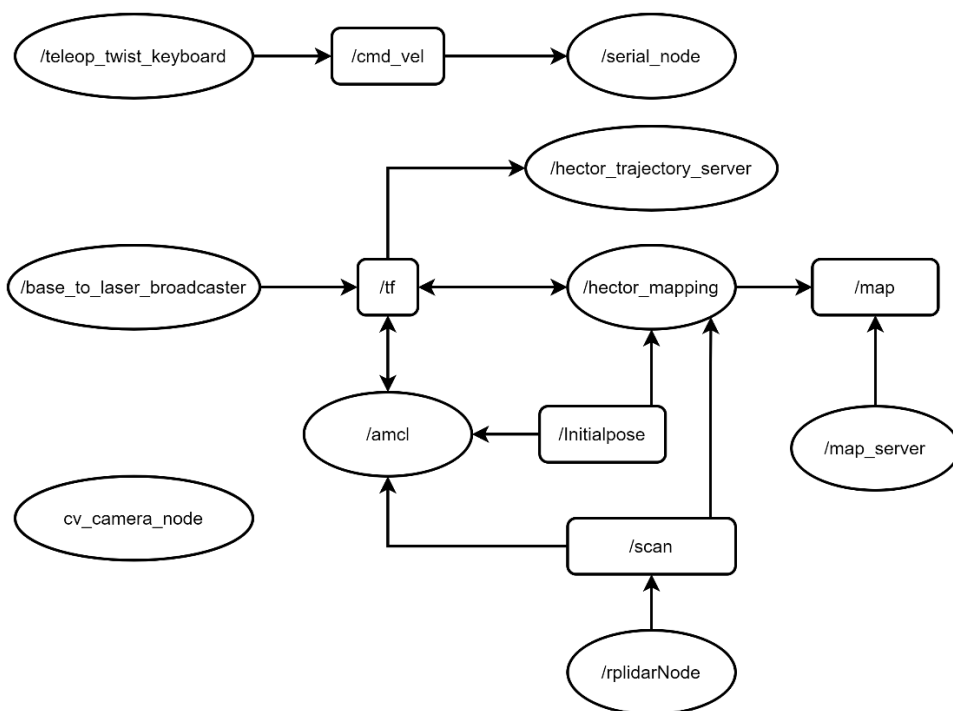
Tính năng	Tạo không gian làm việc cho hệ thống	Giao tiếp giữa PC và Robot	Phát hiện vật cản	Tạo bản đồ 2D	Điều khiển bằng tay	Lên kế hoạch đường đi tự tránh vật cản
Mức độ	Hoàn thành	Hoàn thành	Hoàn thành	Hoàn thành	Hoàn thành	Chưa hoàn thành
Nội dung	Mục 2.3/2.4/2.6.1	Mục 2.5	Mục 2.5	Mục 2.5	Mục 4.4	

Bảng 4.11 Xây dựng hệ thống

4.3.2 Tổng Quan Sơ Đồ Nodes Và Topics



Hình 4.1 Lưu đồ thuật toán hệ thống Robot tạo bản đồ - tiếng việt



Hình 4.2 Lưu đồ thuật toán hệ thống Robot tạo bản đồ - thuật ngữ chuyên ngành

4.3.2.1 Tổng quan

Biểu đồ được cung cấp miêu tả một hệ thống Robot được điều phối qua Robot Operating System (ROS). Biểu đồ thể hiện mạng lưới giao tiếp giữa các *node* độc lập và *topic*, mỗi cái đóng vai trò riêng biệt trong việc xử lý và điều khiển Robot. Trong một dự án có thể nhiều *node* và *topic* liên kết với nhau và các dự án có thể phát triển độc lập.

4.3.2.2 Chi Tiết Hệ Thống:

1. Định nghĩa

Node:

- Một "node" trong ROS là một đơn vị chức năng nhỏ, thường là một chương trình nhỏ thực hiện một nhiệm vụ cụ thể.
- Ví dụ, trong một robot, có thể có một node dành cho việc đọc dữ liệu từ cảm biến, một node khác để điều khiển các động cơ, và một node nữa để xử lý dữ liệu hình ảnh.
- Các node tương tác với nhau qua một hệ thống giao tiếp mạng phân tán, cho phép chúng chia sẻ thông tin và phối hợp các nhiệm vụ.

Topic:

- "Topic" là một kênh giao tiếp mà qua đó các node trong ROS có thể gửi và nhận thông tin.
- Một node gửi thông tin ra một topic được gọi là "publisher" (người xuất bản), trong khi một node nhận thông tin từ một topic được gọi là "subscriber" (người đăng ký).
- Thông tin được gửi qua các topic thường được định dạng dưới dạng "messages" (thông điệp), có cấu trúc dữ liệu cố định.

Ví dụ, một topic có thể được dùng để truyền dữ liệu cảm biến từ node cảm biến đến node xử lý, hoặc từ node điều khiển đến node động cơ.

Cách thức hoạt động của các node và topic trong ROS giúp tạo nên một hệ thống mô-đun và linh hoạt, cho phép các nhà phát triển xây dựng các ứng dụng robot phức tạp từ các thành phần nhỏ hơn, dễ quản lý hơn.

2. Các node và chức năng:

- Điều khiển từ bàn phím (teleop_twist_keyboard): Cho phép người dùng sử dụng bàn phím máy tính để điều khiển hướng và tốc độ của robot.
- Chương trình truyền dữ liệu (serial_node): Đây là cầu nối cho phép robot nhận và gửi dữ liệu qua cổng kết nối.
- Chuyển tọa độ đổi Robot tới laser (base_to_laser_broadcaster): Điều chỉnh thông tin từ robot sang hệ thống đo khoảng cách bằng laser của nó, giúp robot hiểu được môi trường xung quanh.
- Xây dựng bản đồ (hector_mapping): Tạo ra bản đồ môi trường dựa trên dữ liệu từ cảm biến như laser mà không cần dữ liệu vận động.
- Thuật toán AMCL: Giúp robot xác định vị trí của nó trên bản đồ dựa vào thông tin từ cảm biến.
- RPLIDARNode: Đọc thông tin từ cảm biến khoảng cách laser và chuyển thành dữ liệu để robot có thể hiểu được.
- Chương trình xử lý dữ liệu hình ảnh (cv_camera_node): Xử lý hình ảnh từ camera.

3. Các topics và dữ liệu truyền:

- Dữ liệu về vận tốc (/cmd_vel): Chứa thông tin về tốc độ và hướng di chuyển mà robot cần để di chuyển.
- Chuyển đổi hệ tọa độ (/tf): Một hệ thống thông tin giúp robot nhận biết vị trí và hướng của các bộ phận trên cơ thể nó.
- Bản đồ (/map): Chứa bản đồ môi trường mà robot đang hoạt động.
- Vị trí ban đầu (/initialpose): Cho robot biết điểm bắt đầu của nó khi bắt đầu một nhiệm vụ.
- Dữ liệu bản đồ có sẵn (/map_server): Cung cấp bản đồ đã được tạo sẵn cho robot.

- Lưu dữ liệu quét được (/scan): Chứa thông tin quét từ cảm biến laser, giúp robot "nhìn" môi trường xung quanh.

4.3.2.3 Kết luận:

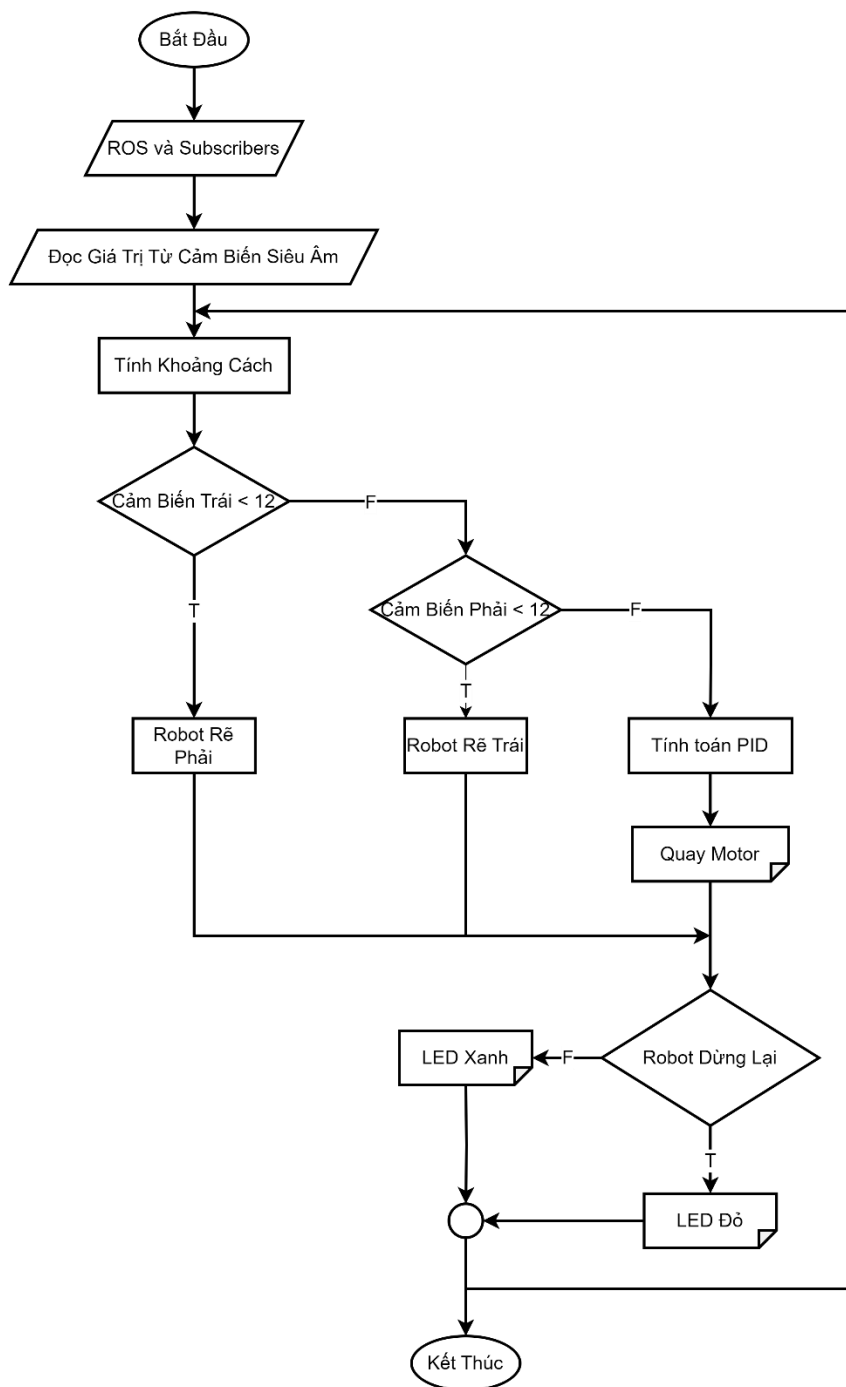
Biểu đồ cung cấp cái nhìn tổng quát về cách thức giao tiếp và hợp tác giữa các phần của hệ thống để thực hiện nhiệm vụ. Điểm nổi bật là sự đa dạng và phức tạp trong giao tiếp và sự linh hoạt của hệ thống, cho phép Robot thực hiện đa dạng nhiệm vụ với khả năng điều chỉnh cao.

4.3.3 File khởi chạy

1. Thiết lập hệ thống slam
 - *Sử dụng Hector_SLAM*: Tích hợp launch file của hector_slam từ gói hector_slam_launch để tạo map và định vị.
 - *Cấu Hình*: Các thiết lập cần thiết đã được định nghĩa trong file *tutorial.launch*.
2. Cấu hình camera
 - *Node Camera*: Sử dụng gói *cv_camera* để thiết lập node camera.
 - *Thiết lập độ phân giải*: Độ phân giải của camera được cấu hình là 640x480.
3. Kết nối với arduino
 - *Node Arduino*: Sử dụng gói *rosserial_python* để tạo kết nối serial với Arduino.
 - *Cấu hình cổng và baudrate*: Cổng */dev/ttyACM0* và baudrate là *115200*.
4. Cấu hình LIDAR
 - *Node LIDAR*: Sử dụng gói *RPLIDAR_ros* để tích hợp cảm biến LIDAR.
 - *Thiết lập cổng và baudrate*: Cổng */dev/ttyUSB0* và baudrate *115200*.
5. Thiết lập map server và amcl
 - *Map Server*: Sử dụng map server để tải bản đồ từ file YAML.
 - *AMCL*: Tích hợp AMCL cho việc định vị dựa trên bản đồ.

4.4 Mã nguồn cho Arduino

4.4.1 Sơ đồ thuật toán



Hình 4.3 Lưu đồ thuật toán điều khiển Robot trên vi điều khiển

1. ROS và Subscribers: Robot sử dụng ROS và đăng ký (subscribers) để nhận thông tin từ các nút khác trong hệ thống.
2. Đọc Giá Trị Từ Cảm Biến Siêu Âm: Robot sử dụng cảm biến siêu âm để đo khoảng cách đến các vật cản.
3. Tính Khoảng Cách: Từ giá trị đọc được từ cảm biến, Robot tính toán khoảng cách đến vật cản.
4. Cảm Biến Trái < 12: Kiểm tra xem khoảng cách đo được từ cảm biến bên trái có nhỏ hơn 12 không.
 - Nếu Đúng (T): Robot sẽ thực hiện hành động Robot Rẽ Phải.
 - Nếu Sai (F): Quá trình tiếp tục kiểm tra cảm biến phía bên phải.
5. Cảm Biến Phải < 12: Tương tự như bước trước, nhưng đối với cảm biến bên phải.
 - Nếu Đúng (T): Robot sẽ thực hiện hành động Robot Rẽ Trái.
 - Nếu Sai (F): Quá trình tiếp tục với việc tính toán PID.
6. Tính toán PID: Robot sử dụng thuật toán PID (Proportional, Integral, Derivative) để điều chỉnh động cơ sao cho có thể điều hướng một cách mượt mà và chính xác.
7. Quay Motor: Robot điều khiển động cơ để thực hiện chuyển động.
8. Robot Dừng Lại: Kiểm tra xem Robot có cần phải dừng lại không.
 - Nếu Đúng (T): Bật LED Xanh để chỉ thị Robot đang trong trạng thái dừng.
 - Nếu Sai (F): Bật LED Đỏ và Robot tiếp tục chuyển động.
9. Kết Thúc: Kết thúc quá trình điều khiển của Robot.
10. Trong quá trình này, LED xanh và đỏ được sử dụng như một phần của hệ thống thông báo trạng thái, có thể để người dùng biết Robot đang ở trạng thái nào: dừng lại hay tiếp tục chuyển động. Thuật toán PID đóng một vai trò quan trọng trong việc đảm bảo Robot di chuyển một cách ổn định và hiệu quả.

4.4.2 Thiết lập bộ điều khiển PID cho động cơ

Bộ điều khiển PID (Proportional-Integral-Derivative) là một trong những phương pháp điều khiển tự động phổ biến được sử dụng trong nhiều ứng dụng, bao gồm điều khiển động cơ cho Robot. Bộ điều khiển PID hoạt động dựa trên ba thành phần chính: P (Tỷ lệ), I (Tích phân), và D (Đạo hàm). Mỗi thành phần này tương ứng với một khía cạnh khác nhau của quá trình điều khiển

Đặt Tín Hiệu Ngõ Vào (Input Signal):

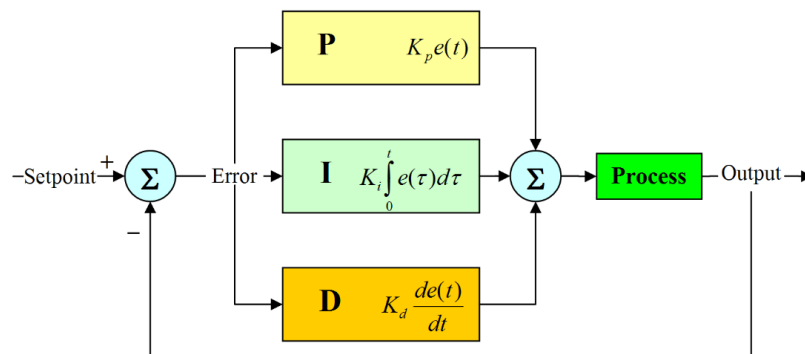
Trong trường hợp của động cơ lái bánh xe Robot, tín hiệu ngõ vào thường là số xung xác lập, biểu thị vận tốc mong muốn của động cơ (ví dụ: vòng/phút).

Xác Định Tín Hiệu Ngõ Ra (Output Signal):

Giá trị ngõ ra từ bộ điều khiển PID thường là vận tốc động cơ, được điều chỉnh dưới dạng độ rộng xung. Điều này quyết định mức năng lượng được cung cấp cho động cơ.

Phản Hồi Tín Hiệu (Feedback Signal):

Tín hiệu phản hồi thường đến từ các thiết bị như encoder, cung cấp thông tin về số xung thực tế (vòng/phút) mà động cơ đang sản sinh. Điều này giúp bộ điều khiển PID điều chỉnh ngõ ra để khớp với ngõ vào mong muốn.



Hình 4.4 Sơ đồ khối của bộ điều khiển PID

Phương trình cơ bản của bộ điều khiển PID có thể được biểu diễn như sau:

$$u(t) = P_{out} + I_{out} + D_{out}$$

Khâu tỉ lệ

$$P_{out} = K_p e(t)$$

- P_{out} : Thừa số tỉ lệ của đầu ra
- K_p : Hệ số tỉ lệ, thông số điều chỉnh
- e : Sai số = SP – PV
- t : Thời gian hay thời gian tức thời (hiện tại)

Khâu tích phân

$$I_{out} = K_i \int_0^t e(\tau) d\tau$$

- I_{out} : Thừa số tích phân của đầu ra
- K_i : Độ lợi tích phân, 1 thông số điều chỉnh
- e : Sai số = SP – PV
- t : Thời gian hay thời gian tức thời (hiện tại)
- τ : Một biến tích phân trung gian

Khâu đạo hàm

$$D_{out} = k_d \frac{d}{dt} e(t)$$

- D_{out} : Thừa số tích phân của đầu ra
- K_d : Độ lợi tích phân, 1 thông số điều chỉnh
- e : Sai số = SP – PV
- t : Thời gian hay thời gian tức thời (hiện tại)

Khâu tỉ lệ, tích phân, vi phân được cộng lại với nhau để tính toán đầu ra của bộ điều khiển PID. Định nghĩa rằng $u(t)$ là đầu ra của bộ điều khiển, biểu thức cuối cùng của giải thuật PID là:

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

Quá Trình Thiết Lập:

1. Tuning (Điều Chỉnh):

Điều chỉnh ba hệ số K_p K_i K_d để đạt được hiệu suất điều khiển tốt nhất. Quá trình này yêu cầu kiểm tra và điều chỉnh dựa trên phản hồi thực tế từ hệ thống.

2. Thử Nghiệm và Tối Ưu:

Thực hiện các thử nghiệm để đánh giá hiệu suất của hệ thống điều khiển. Điều chỉnh các hệ số PID dựa trên kết quả thử nghiệm để tối ưu hóa hiệu suất.

3. Ứng Dụng Trên Hệ Thống Thực Tế:

Sau khi tối ưu, áp dụng cài đặt PID trên hệ thống động cơ thực tế của Robot và tiếp tục quan sát, điều chỉnh nếu cần.

4.4.3 Thiết lập điều khiển HCSR 04 và DC BTS7960 43A

4.4.3.1 Thư viện và Khai báo Lớp

Motor.h: Định nghĩa lớp Motor, dùng để điều khiển động cơ.

Ultrasonic.h: Định nghĩa lớp Ultrasonic, dùng để làm việc với cảm biến siêu âm.

4.4.3.2 Lớp Motor

Constructor: Khởi tạo đối tượng Motor với các chân điều khiển và mã hóa (encoder).

Các biến plus, minus, và pin được khởi tạo để quản lý các chân động cơ.

Phương thức rotate(int value): Điều khiển động cơ quay.

Sử dụng digitalWrite() và analogWrite() để điều khiển hướng và tốc độ của động cơ.

Sử dụng map() để chuyển đổi giá trị đầu vào sang phạm vi PWM.

4.4.3.3 Lớp Ultrasonic

Constructor: Khởi tạo đối tượng Ultrasonic với chân truyền (trig) và nhận (echo).

Phương thức getDistance(): Đo khoảng cách sử dụng cảm biến siêu âm.

Gửi tín hiệu siêu âm và đo thời gian phản hồi để tính khoảng cách.

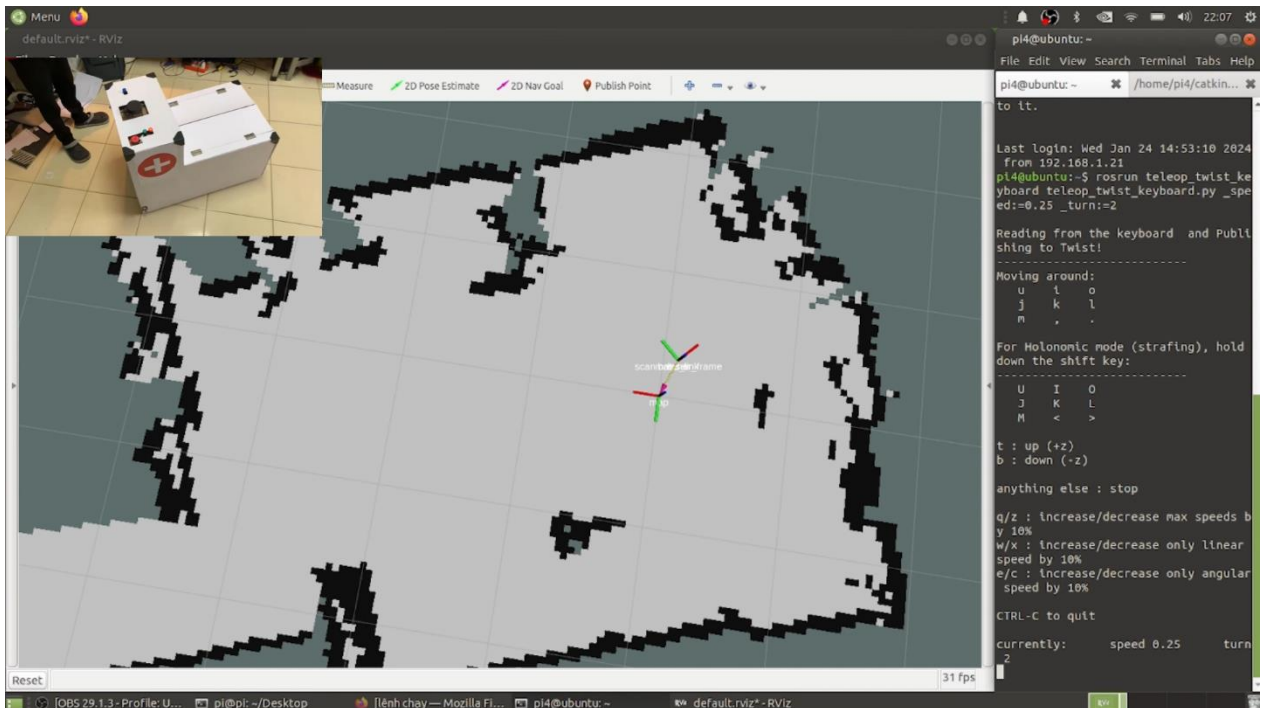
pulseIn() được sử dụng để đo thời gian phản hồi.

Kết Luận:

Chương trình này cung cấp cơ sở vững chắc cho việc điều khiển Robot với khả năng điều khiển động cơ và đọc dữ liệu từ cảm biến siêu âm. Các lớp Motor và Ultrasonic được thiết kế để dễ dàng tích hợp và sử dụng trong các dự án Arduino, đặc biệt là trong các ứng dụng Robot. Mỗi phần của mã đều được thiết kế để tối ưu hóa hiệu suất và độ chính xác của hệ thống điều khiển Robot.

4.5 Kết quả hoạt động Robot

Robot đã thành công trong việc tạo ra bản đồ của môi trường xung quanh như được hiển thị trên giao diện RViz, một công cụ trực quan hóa trong ROS. Bản đồ này phản ánh các đặc điểm của không gian làm việc, bao gồm vị trí của các bức tường và các chướng ngại vật khác. Việc sử dụng tay cầm điều khiển từ xa cho thấy robot có khả năng được điều khiển bằng tay, cho phép người dùng can thiệp và điều hướng nó một cách linh hoạt.



Hình 4.5 Làn chạy thử nghiệm lần 2

Giao diện hiển thị một robot với màu sắc khác biệt (các trục màu xanh, đỏ, và xanh lá cây) định vị ở trung tâm bản đồ là tọa độ x, y, z , cho thấy vị trí và hướng hiện tại của nó. Sự xuất hiện của các điểm màu xám và đen trên bản đồ chỉ ra những khu vực mà robot đã quét và phát hiện chướng ngại vật, trong khi các khu vực trắng là những nơi chưa được khám phá hoặc không có chướng ngại vật.

Robot được trang bị nhiều cảm biến và phần cứng điện tử, được lắp đặt trên một cấu trúc có bánh xe, cho phép nó di chuyển và thực hiện các nhiệm vụ một cách độc lập. Sự kết hợp

giữa phần cứng và phần mềm trong hình ảnh cho thấy một hệ thống tự động hóa tích hợp và phức tạp, có khả năng thực hiện các tác vụ như mapping và di chuyển.

KẾT LUẬN

I. Kết quả đạt được

1. Thiết kế và xây dựng robot AMR:

Đã thành công trong việc thiết kế và xây dựng một hệ thống điện cho Robot AMR, sử dụng các thành phần như Arduino Uno, Raspberry Pi, Battery, LED AD, MODULE Relay 12VDC 4 Kênh, Động Cơ DC Servo JGB37, DC BTS7960 43A, v.v.

2. Tích hợp cảm biến và thiết bị điều khiển:

Robot AMR đã được trang bị khả năng phát hiện vật thể và xây dựng bản đồ, một bước tiến quan trọng trong việc nâng cao khả năng tự động hóa.

3. Lập trình và cài đặt thuật toán:

Phần mềm của robot đã được lập trình để Robot có khả năng di chuyển linh hoạt khi được điều khiển bằng tay, nhờ vào các thuật toán và gói package được cài đặt.

4. So sánh với mục tiêu ban đầu:

So với mục tiêu ban đầu là tạo ra một Robot AMR có khả năng tự động hóa cao trong việc phát hiện vật thể, xây dựng bản đồ, kết quả đạt được đã phản ánh một mức độ thành công đáng kể. Đặc biệt, khả năng tích hợp cảm biến và phần mềm lập trình cho thấy sự tiến bộ rõ rệt so với yêu cầu ban đầu.

5. Nhận xét và đánh giá cá nhân:

Dự án đã cho thấy một bước tiến đáng kể trong việc áp dụng công nghệ robot vào lĩnh vực y tế. Mặc dù có những thách thức về nguồn lực và kỹ thuật, nhưng khả năng sáng tạo và giải quyết vấn đề đã được thể hiện rõ nét. Các bài học kinh nghiệm từ dự án này sẽ hữu ích cho các nghiên cứu tương lai.

Hạn chế

1. Giới hạn về nguồn lực:

Mô hình cơ khí lớn đòi hỏi chi phí cao, cùng với việc máy tính nhúng và LIDAR có giá thành đắt đỏ.

2. Thách thức trong thu thập và phân tích dữ liệu:

Gặp khó khăn trong việc thu thập dữ liệu vị trí cần thiết, điều này ảnh hưởng đến quá trình phân tích và xử lý dữ liệu.

3. Khó khăn về mặt kỹ thuật hoặc công nghệ:

Đầu vào dữ liệu cho phần tự động hóa chưa được chính xác, ảnh hưởng đến hiệu quả của hệ thống.

4. Hạn chế về phạm vi ứng dụng:

Robot AMR có thể chưa đủ linh hoạt để ứng dụng trong mọi tình huống hoặc môi trường làm việc, nhất là trong các điều kiện không dự đoán trước được hoặc phức tạp.

5. Giới hạn về quy mô và độ chính xác:

Kích thước của robot và độ chính xác trong việc thực hiện các nhiệm vụ cụ thể có thể chưa đáp ứng được yêu cầu cao nhất như lên kế hoạch đường đi tự tránh vật cản

II. Hướng phát triển đề tài

Trong bối cảnh hiện nay, việc ứng dụng công nghệ Robot vào các lĩnh vực như y tế đang ngày càng trở nên quan trọng. Robot AMR đã được phát triển với khả năng tạo map 2D và điều khiển bằng tay, nhưng để nâng cao giá trị ứng dụng của nó, chúng ta cần mở rộng các khả năng của Robot. Dưới đây là một số đề xuất cho việc mở rộng này.

1. Chế độ tự động hóa hoàn toàn

Mục tiêu: Phát triển chức năng tự định tuyến và di chuyển không cần sự can thiệp của con người.

Phương pháp: Tích hợp các thuật toán SLAM để Robot có khả năng tự định vị và tạo map môi trường một cách độc lập.

Lợi ích: Tăng cường độ chính xác và hiệu quả trong việc di chuyển và thực hiện nhiệm vụ.

2. Tối ưu giao diện người dùng

Mục tiêu: Cải thiện giao diện người dùng, tạo trải nghiệm thân thiện và dễ dàng sử dụng.

Phương pháp: Phát triển ứng dụng với khả năng hiển thị map 2D và thông tin trạng thái hoạt động của Robot.

Lợi ích: Tăng cường khả năng tương tác và giám sát từ xa, giúp người dùng dễ dàng quản lý và điều khiển Robot.

3. Tích hợp hệ thống điều hướng đa cấp

Mục tiêu: Nâng cấp hệ thống điều hướng cho phép xử lý tình huống phức tạp.

Phương pháp: Tích hợp công nghệ nhận diện và tránh vật cản, khả năng di chuyển trong môi trường đa tầng.

Lợi ích: Tăng cường sự linh hoạt và độ an toàn trong quá trình hoạt động của Robot.

4. Kết nối và tích hợp với các hệ thống tự động phát thuốc

Mục tiêu: Tích hợp với hệ thống tự động phát thuốc trong bệnh viện.

Phương pháp: Phát triển khả năng kết nối và làm việc với hệ thống phát thuốc, bao gồm quét thẻ, nhận thuốc và ghi lại thông tin.

Lợi ích: Tối ưu hóa quy trình phát thuốc, giảm thiểu sai sót và tăng cường hiệu quả trong quản lý thuốc.

5. Ứng dụng trí tuệ nhân tạo và học máy

Mục tiêu: Cải thiện sự linh hoạt và hiệu quả của Robot thông qua AI và học máy.

Phương pháp: Phát triển khả năng giao tiếp và tương tác đơn giản với bệnh nhân thông qua công nghệ AI.

Lợi ích: Tăng cường khả năng tương tác và hỗ trợ bệnh nhân, giảm bớt áp lực công việc cho nhân viên y tế.

6. Kết luận

Những đề xuất mở rộng này không chỉ giúp tăng cường khả năng hoạt động độc lập của Robot mà còn tạo ra giá trị ứng dụng thực tế trong môi trường bệnh viện. Sự kết hợp giữa công nghệ tiên tiến và yếu tố con người sẽ mở ra những cơ hội mới cho việc cải thiện chất lượng dịch vụ y tế.

Tài liệu tham khảo

- [1] Effective Robotics Programming with ROS - Third Edition của Anil Mahtani, Luis Sanchez, Enrique Fernandez, và Aaron Martinez
- [2] PID Controllers: Theory, Design, and Tuning của Karl Johan Åström, Tore Hägglund
- [3] Learning ROS for Robotics Programming 2015 của Enrique Fernández, Luis Sánchez Crespo, Anil Mahtani, Aaron Martinez
- [4] Mastering ROS for Robotics Programming của Lentin Joseph
- [5] PID Control in the Third Millennium: Lessons Learned and New Approaches của Antonio Visioli và Qing-Guo Wang
- [6] Programming Robots with ROS: A Practical Introduction to the Robot Operating System của Morgan Quigley, Brian Gerkey, và William D. Smart

Phụ lục 1

Code Arduino chính

```
#include "Motor.h"
#include "ultrasonic.h"
#include <ros.h>
#include <geometry_msgs/Twist.h>
#include <PID_v1.h>
#include <geometry_msgs/Vector3Stamped.h>
ros::NodeHandle nh;
#define LOOPTIME 10
Motor right(9, 10, 8);
Motor left(5, 6, 7);
Ultrasonic leftSensor(A0, A1);
Ultrasonic rightSensor(A2, A3);
int ledr = 4; int ledg = 11;
double left_kp = 3, left_ki = 0, left_kd = 0.0;
double right_kp = 3, right_ki = 0, right_kd = 0.0;
double right_input = 0, right_output = 0, right_setpoint = 0;
PID rightPID(&right_input, &right_output, &right_setpoint, right_kp, right_ki,
right_kd, DIRECT);
double left_input = 0, left_output = 0, left_setpoint = 0;
PID leftPID(&left_input, &left_output, &left_setpoint, left_kp, left_ki, left_kd,
DIRECT);
float demandx = 0;
float demandz = 0;
double demand_speed_left;
```

```

double demand_speed_right;
unsigned long currentMillis;
unsigned long prevMillis;
float speed_act_left = 0;
float speed_act_right = 0;
int left_pwm_value = 0;
int right_pwm_value = 0;
int a = 100;
unsigned long ledDelayStart = 0;
const unsigned long ledDelayDuration = 1000;
geometry_msgs::Vector3Stamped ticks_msg;
ros::Publisher left_ticks_pub("left_ticks", &ticks_msg);
ros::Publisher right_ticks_pub("right_ticks", &ticks_msg);
geometry_msgs::Vector3Stamped pwm_msg;
ros::Publisher left_pwm_pub("left_pwm", &pwm_msg);
ros::Publisher right_pwm_pub("right_pwm", &pwm_msg);
void cmd_vel_cb(const geometry_msgs::Twist &twist)
{
    demandx = twist.linear.x;
    demandz = twist.angular.z;
}
ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel", cmd_vel_cb);
void publishTicksAndPWM(double time)
{
    pwm_msg.header.stamp = nh.now();
    pwm_msg.vector.x = left_pwm_value;
    pwm_msg.vector.y = right_pwm_value;
    pwm_msg.vector.z = time / 1000;
    left_pwm_pub.publish(&pwm_msg);
}

```

```

    right_pwm_pub.publish(&pwm_msg);
}

void setup()
{
    nh.getHardware()->setBaud(115200);
    nh.initNode();
    nh.subscribe(sub);
    nh.advertise(left_pwm_pub);
    nh.advertise(right_pwm_pub);
    rightPID.SetMode(AUTOMATIC);
    rightPID.SetSampleTime(1);
    rightPID.SetOutputLimits(-a, a);
    leftPID.SetMode(AUTOMATIC);
    leftPID.SetSampleTime(1);
    leftPID.SetOutputLimits(-a, a);
    pinMode(ledr, OUTPUT);
    pinMode(ledg, OUTPUT);
    digitalWrite(ledg, LOW);
    digitalWrite(ledr, LOW);
}

void loop()
{
    currentMillis = millis();
    if (currentMillis - prevMillis >= LOOPTIME)
    {
        prevMillis = currentMillis;
        int leftDistance = leftSensor.getDistance();
        int rightDistance = rightSensor.getDistance();
    }
}

```

```

int temp_left_pwm = left_output;
int temp_right_pwm = right_output;
if (leftDistance < 12)
{
    left_pwm_value = 80;
    right_pwm_value = -110;
}
else if (rightDistance < 12)
{
    left_pwm_value = -110;
    right_pwm_value = 80;
}
else
{
    demand_speed_left = demandx - (demandz * 0.1075);
    demand_speed_right = demandx + (demandz * 0.1075);
    left_setpoint = demand_speed_left * 40;
    right_setpoint = demand_speed_right * 40;
    leftPID.Compute();
    rightPID.Compute();
}
left.rotate(left_pwm_value);
right.rotate(right_pwm_value);
if (left_pwm_value == 0 && right_pwm_value == 0)
{
    digitalWrite(ledr, HIGH);
    digitalWrite(ledg, LOW);

    if (millis() - ledDelayStart >= ledDelayDuration)

```

```

{
    ledDelayStart = millis(); // Reset the timer
}
}
else
{
    digitalWrite(ledg, HIGH);
    digitalWrite(ledr, LOW);

    if (millis() - ledDelayStart >= ledDelayDuration)
    {
        ledDelayStart = millis(); // Reset the timer
    }
}

left_pwm_value = map(left_output, -a, a, -255, 255);
right_pwm_value = map(right_output, -a, a, -255, 255);
int m = 100;
int h = 80;
if (left_pwm_value >= 140) {
    left_pwm_value = m;
    right_pwm_value = h;
}
if (right_pwm_value >= 140) {
    right_pwm_value = m;
    left_pwm_value = h;
}
if (left_pwm_value <= -140) {
    left_pwm_value = -m;

```

```

    right_pwm_value = -h;
}
if (right_pwm_value <= -140) {
    right_pwm_value = -m;
    left_pwm_value = -h;
}
publishTicksAndPWM(LOOPTIME);
nh.spinOnce();
}
}

```

Code C++ Motor

```

#include "Arduino.h"
#include "Motor.h"
Motor::Motor(int plus, int minus, int pin) {
    pinMode(pin, OUTPUT);
    pinMode(plus, OUTPUT);
    pinMode(minus, OUTPUT);
    pinMode(en_a, INPUT_PULLUP);
    pinMode(en_b, INPUT_PULLUP);
    Motor::plus = plus;
    Motor::minus = minus;
    Motor::pin = p
}
void Motor::rotate(int value) {
    digitalWrite(pin, HIGH);
    if (value >= 0) {
        int out = map(value, 0, 120, 0, 120);
    }
}

```

```

    analogWrite(plus,out);
    analogWrite(minus,0);
}else{
    //Max Voltage with 12.5V battery with 12V required
    //(12/12.5)*255 ~=244
    int out = map(value, 0, -120, 0, 120);
    analogWrite(plus,0);
    analogWrite(minus,out);
}
}

```

Code C Motor

```

#ifndef Motor_h
#define Motor_h
#include "Arduino.h"
class Motor {
public:
    //Constructor - Plus and Minus are the Motor output / en_a and en_b are the encoder
    inputs
    Motor(int plus, int minus, int pin);
    //Spin the motor with a percentage value
    void rotate(int value);
    //Motor Outputs - plus is one direction and minus is the other
    int plus;
    int minus;
    int pin;
};
#endif

```


Code C++ Ultrasonic

```
#include "ultrasonic.h"
Ultrasonic::Ultrasonic(int trigPin, int echoPin) {
    this->trigPin = trigPin;
    this->echoPin = echoPin;
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}
long Ultrasonic::getDistance() {
    digitalWrite(trigPin, LOW);delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    long duration = pulseIn(echoPin, HIGH);
    return duration / 29.1 / 2;
}
```

Code C Ultrasonic

```
#ifndef ULTRASONIC_H
#define ULTRASONIC_H
#include <Arduino.h>
class Ultrasonic {
public:
    Ultrasonic(int trigPin, int echoPin);
    long getDistance();
private:
    int trigPin; int echoPin;
};
#endif // ULTRASONIC_H
```

Phụ lục 2

1. Bản vẽ sơ đồ nguyên lý hệ thống điều khiển – A3
2. Lưu đồ thuật toán hệ thống điều khiển – A3
3. Sơ đồ khối hệ thống điều khiển – A3
4. Bản tổng thể – A3