

ĐẠI HỌC BÁCH KHOA HÀ NỘI



TRƯỜNG CƠ KHÍ - KHOA CƠ ĐIỆN TỬ



ĐỒ ÁN TỐT NGHIỆP

THIẾT KẾ ROBOT TỰ HÀNH TRONG NGÀNH DỊCH VỤ AUTONOMOUS SERVICE ROBOT

Hoàng Gia Cường

Cuong.hg184367@sis.hust.edu.vn

Chuyên ngành Cơ điện tử

Giảng viên hướng dẫn: PGS.TS. Bùi Hải Lê

Chữ ký của GVHD

Khoa: Cơ điện tử
Trường: Cơ khí

Hà Nội, 8/2023

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

1. Thông tin về sinh viên:

Họ và tên SV: **Hoàng Gia Cường** Lớp: Cơ điện tử 07 ĐT: 0705506103
Email (đại diện): **cuong.hg184367@sis.hust.edu.vn**
Hệ đào tạo: Chính quy Chuyên ngành: Cơ điện tử
Đồ án tốt nghiệp được thực hiện tại: Phòng 809 thư viện Tạ Quang Bửu
Thời gian làm ĐATN: Từ ngày 01/04/2023 đến ngày 11/08/2023

2. Tên đề tài:

**THIẾT KẾ ROBOT TỰ HÀNH TRONG NGÀNH DỊCH VỤ
AUTONOMOUS SERVICE ROBOT**

3. Thông số đề tài:

- Không gian hoạt động: Trong nhà hàng nhỏ
- Vận tốc di chuyển: ~ 0.5 m/s
- Khối lượng hành hóa: 12 kg
- Có thể tạo bản đồ và lập kế hoạch đường đi .
- Có thể thao tác qua giao diện HDMI, giao diện web, app.
- Ứng dụng trí tuệ nhân tạo điều khiển robot.

Hà Nội, ngày tháng năm 2023
Giáo viên hướng dẫn

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

Tên đề tài: Thiết kế robot trong ngành dịch vụ-Autonomous service robot.
Họ và tên SV: **Hoàng Gia Cường** Lớp: Cơ điện tử 07
Chuyên ngành: Cơ điện tử
Giáo viên hướng dẫn: PGS.TS. Bùi Hải Lê

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

I. Tác phong làm việc

- Tích cực tham gia làm đồ án, hòa đồng làm việc nhóm tốt.
- Chăm chỉ có tinh thần tìm hiểu, nghiên cứu ứng dụng các công nghệ trong quá trình thực hiện đồ án tốt nghiệp.

II. Những kết quả đạt được

- Thiết kế được mô hình robot đảm bảo các yêu cầu giao hàng, tốc độ và độ bền.
- Robot có thể lập bản đồ, xây dựng đường đi gần tối ưu trong cả mô phỏng và thực nghiệm.
- Robot có thể giao tiếp và được điều khiển thông qua nhiều giao diện: giao diện HDMI trên robot, giao diện web, app.
- Ứng dụng trí tuệ nhân tạo trong nhận diện đồ uống bằng YOLO, điều khiển bằng cử chỉ tay.

III. Hạn chế của đồ án

- Robot còn sai lệch về hướng và vị trí.
- Nhận diện điều khiển bằng tay còn có độ trễ.

IV. Kết luận

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

Đánh giá: điểm

Hà Nội, ngày tháng năm 2023
Giáo viên hướng dẫn

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

Tên đề tài: Thiết kế robot trong ngành dịch vụ-Autonomous service robot.
Họ và tên SV: **Hoàng Gia Cường** Lớp: Cơ điện tử 07
Chuyên ngành: Cơ điện tử

NỘI DUNG NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Những kết quả đạt được

- Đề tài đã thiết kế, điều khiển robot giao hàng. Robot đảm bảo được các yêu cầu giao hàng, tốc độ và độ bền.
- Robot có khả năng tạo bản đồ và xây dựng đường đi.
- Điều khiển, giao tiếp qua nhiều giao diện: giao diện HMI trên robot, giao diện web và app.
- Robot sử dụng trí tuệ nhân tạo để nhận diện đồ uống, điều khiển robot bằng cử chỉ tay

Hạn chế của đồ án

- Robot còn sai lệch về hướng và vị trí.
- Nhận diện điều khiển bằng tay còn có độ trễ.
- Phần thiết kế cơ khí cần tính toán thêm.

Kết luận

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

Đánh giá: điểm

Hà Nội, ngày tháng năm 2023
Giáo viên phản biện
(Ký và ghi rõ họ tên)

Lời cảm ơn

Lời cảm ơn đầu tiên chân tình và sâu sắc nhất con xin được gửi tới ba mẹ, những người luôn yêu thương con vô điều kiện, tha thứ thật nhiều sai lầm, luôn ủng hộ và động viên con hoàn thành chặng đường sinh viên. Con cũng cảm ơn gia đình lớn, là nguồn động lực cho con, ủng hộ những gì con làm.

Em xin được gửi lời cảm ơn chân thành tới PGS.TS Bùi Hải Lê luôn tạo điều kiện cho em hoàn thành đồ án. Em xin cảm ơn thầy TS. Bùi Đình Bá người trực tiếp hướng dẫn, chỉ bảo từ những việc nhỏ nhất, giúp em có những trải nghiệm đáng nhớ thời sinh viên. Em chúc thầy luôn giữ được nhiệt huyết, vui vẻ tiếp tục đồng hành cùng các thế hệ sinh viên cơ điện tử tiếp theo.

Mình xin gửi thêm nhiều lời cảm ơn tới nhóm đồ án: Lê Ngọc Duy, Trần Văn Lợi, Phạm Minh Hoàng, cảm ơn chúng ta vì luôn đoàn kết, lạc quan dù nhiều lúc khó khăn và bất lực, nhưng cuối cùng cũng đã cùng nhau vượt qua hoàn thành tốt đồ án. Mình cũng cảm ơn những người bạn ở bên, cổ vũ là điểm tựa tinh thần cho mình.

Cuối cùng xin được gửi lời cảm ơn tới chính bản thân đã không bỏ cuộc, đã luôn cố gắng vượt qua khó khăn, vượt qua thật nhiều nỗi sợ, tự ti để trưởng thành hơn, điềm tĩnh hơn.

Cảm ơn Bách Khoa! Cảm ơn tất cả!

Tóm tắt nội dung đồ án

Nội dung đồ án bao gồm có 6 chương bao gồm: Tổng quan đề tài; tính toán và thiết kế hệ thống cơ khí; tính toán thiết kế hệ thống điều khiển; lập trình điều khiển hệ thống và kết quả thực nghiệm, thiết kế giao diện điều khiển robot; kết luận và định hướng phát triển mở rộng.

Đồ án Thiết kế robot trong ngành dịch vụ-Autonomous service robot giúp giải quyết vấn đề nhân lực, từ đó tiết kiệm được chi phí, mang tới cho người sử dụng sự tiện lợi trong công việc. Nhiệm vụ chính của xe là vận chuyển đồ uống hoặc thức ăn từ vị trí của nhân viên đến vị trí của khách hàng. Robot có khả năng giao tiếp với người quản lý thông qua thông qua giao diện trên xe, giao diện web app và qua cử chỉ tay để vận hành một cách chính xác. Sử dụng phương pháp lập bản đồ, kết hợp với một số cảm biến để có thể lập kế hoạch đường đi, tối ưu hóa đường đi và di chuyển tránh vật cản. Robot sử dụng cảm biến Lidar để định vị và xây dựng bản đồ, kết hợp các thuật toán tìm đường trên hệ điều hành ROS, giao tiếp với khách hàng qua màn hình HDMI với giao diện thiết kế thông qua ngôn ngữ python và thư viện Tkinter, khách hàng cũng có thể giao tiếp với robot thông qua giao diện trên web, app hoặc dùng cử chỉ tay để điều khiển robot tới vị trí mong muốn để lấy đồ ăn.

Kết quả của đồ án là robot đã có thể xây dựng bản đồ cũng như tìm được đường đi gần tối ưu trong cả mô phỏng lẫn thực nghiệm. Robot có thể được điều khiển qua giao diện HMI, web, app và cử chỉ tay. Do vậy, đề tài này hoàn toàn là khả thi và có thể dẫn đầu xu hướng trong tương lai.

Qua đồ án này, em đã học được rất nhiều kỹ năng làm việc nhóm cũng như các kiến thức chuyên môn như sử dụng các thuật toán tìm đường trên hệ điều hành ROS, thiết kế giao diện sử dụng ngôn ngữ python kết hợp thư viện Tkinter, giao diện web sử dụng html, css, js, giao diện app sử dụng flutter nhận dạng đối tượng sử dụng YOLO, nhận dạng cử chỉ tay sử dụng thư viện mediapipe hand. Các kiến thức trên đã giúp em tăng thêm nhiều kiến thức cũng như kỹ năng sử dụng trong công việc và cuộc sống sau này.

Sinh viên thực hiện
Ký và ghi rõ họ tên

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI	1
1.1 Giới thiệu.....	1
1.2 Tổng quan về robot tự hành.....	1
1.3 Các thành phần của robot tự hành.....	1
1.3.1 Nhận thức của robot (see).....	2
1.3.2 Quyết định của robot (think).....	2
1.3.3 Hành động của robot (act).....	2
1.4 Ứng dụng của robot tự hành.....	2
1.4.1 Robot tự hành trong kho hàng.....	2
1.4.2 Robot tự hành dùng trong lĩnh vực y tế.....	3
1.4.3 Robot tự hành dùng trong lĩnh vực quân sự.....	4
1.5 Đặt vấn đề.....	5
1.6 Đề xuất mô hình robot.....	6
CHƯƠNG 2. THIẾT KẾ MÔ HÌNH ROBOT	7
2.1 Giới thiệu.....	7
2.2 Yêu cầu kỹ thuật.....	7
2.3 Thiết kế cơ khí.....	7
2.3.1 Lựa chọn kết cấu của mô hình xe giao hàng.....	7
2.3.2 Mô hình xây dựng cho kết cấu đã lựa chọn.....	10
2.4 Tính toán và lựa chọn động cơ.....	11
2.4.1 Thông số yêu cầu.....	11
2.4.2 Tính chọn động cơ 2 bánh dẫn động.....	12
2.5 Thiết kế sơ bộ khung và kiểm nghiệm.....	14
2.5.1 Thiết kế sơ bộ khung.....	14
2.5.2 Kiểm nghiệm khung.....	15
CHƯƠNG 3. THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN	26
3.1 Giới thiệu.....	26
3.2 Thiết kế hệ thống điều khiển phần cứng.....	26
3.2.1 NVIDIA Jetson Nano Developer Kit.....	26
3.2.2 RPLiDAR A1M8.....	31
3.2.3 Hoverboard Hardware.....	32
3.2.4 Mạch phát âm thanh DY-SV5W.....	34

3.2.5	Loa Toàn Dải 2inch - 6Ω 5W.....	36
3.2.6	Nút dừng khẩn cấp.....	36
3.2.7	Pin.....	37
3.2.8	Màn hình HDMI LCD 7 inch.....	38
3.2.9	Webcam Full HD 1080P.....	40
3.3	Bản vẽ mạch điện.....	41
CHƯƠNG 4. LẬP TRÌNH ĐIỀU KHIỂN HỆ THỐNG		42
4.1	Giới thiệu.....	42
4.2	Tổng quan về ROS.....	42
4.2.1	ROS là gì?.....	42
4.2.2	FILE-SYSTEM LEVEL của ROS.....	42
4.2.3	Các thành phần của ROS.....	43
4.2.4	Dữ liệu từ RPLidar A1.....	44
4.2.5	Hector SLAM.....	45
4.2.6	Thiết lập bản đồ.....	46
4.3	Bài toán hoạch định đường đi.....	47
4.3.1	Tổng quan.....	47
4.3.2	Thuật toán Dynamic Window Approach (DWA).....	48
4.3.3	Tổng quan về thuật toán tìm đường đi ngắn nhất.....	51
4.3.4	Kết quả thực nghiệm.....	54
4.4	Bài toán nhận dạng đồ uống sử dụng YOLO.....	55
4.4.1	Tổng quan về mạng YOLO.....	55
4.4.2	Kết quả.....	60
4.5	Ứng dụng trí tuệ nhân tạo điều khiển robot bằng cử chỉ tay.....	64
4.5.1	Công nghệ nhận diện cử chỉ tay trong robot.....	64
4.5.2	Mediapipie hand recognige.....	64
4.5.3	Mô hình.....	65
4.5.4	Train nhận dạng cử chỉ tay.....	69
4.5.5	Kết nối và gửi dữ liệu điều khiển tới robot.....	70
CHƯƠNG 5. THIẾT KẾ GIAO DIỆN		73
5.1	Giới thiệu.....	73
5.2	Thiết kế giao diện web, app order món ăn.....	73
5.2.1	HTML.....	73
5.2.2	CSS.....	76

5.2.3	JavaScript.....	77
5.2.4	Giao diện web.....	79
5.2.5	Flutter.....	80
5.2.6	Rosbridge và thư viện Rosjs.....	86
5.3	Kết nối và điều khiển Robot từ giao diện web.....	89
5.4	Thiết kế giao diện trên robot.....	93
5.4.1	Giới thiệu.....	93
5.4.2	Nguyên lý hoạt động.....	93
5.4.3	Kết quả.....	93
CHƯƠNG 6. KẾT LUẬN		96
6.1	Kết luận.....	96
6.2	Hướng phát triển tương lai của đề tài.....	96
TÀI LIỆU THAM KHẢO		97
PHỤ LỤC		98

DANH MỤC HÌNH VẼ

Hình 1.1 Các thành phần của robot tự hành	2
Hình 1.2 Robot tự hành trong kho hàng	3
Hình 1.3 Robot tự hành dùng trong lĩnh vực y tế	3
Hình 1.4 Robot tự hành dùng trong lĩnh vực quân sự	4
Hình 1.5 Robot dạng người phục vụ trong nhà hàng ở Pháp	5
Hình 2.1 Hình ảnh một số loại robot DDR	7
Hình 2.2 Mô hình thiết kế của nhóm	8
Hình 2.3 Mô hình động học của robot	8
Hình 2.4 Mô hình robot được thiết kế trong Solidworks	10
Hình 2.5 Bản vẽ kỹ thuật mô hình robot	11
Hình 2.6 Hoverboard bldc Motor	13
Hình 2.7 Thông tin về nhôm 20X20	14
Hình 2.8 Thiết kế khung trên solidworks	15
Hình 2.9 Thiết kế khung trên solidworks	16
Hình 2.10 Sau khi đã thiết lập xong các ràng buộc cố định cho khung xe	18
Hình 2.11 Khung xe sau khi đặt lực phân bố	21
Hình 2.12 Khung xe sau khi được tạo Mesh	22
Hình 2.13 Kết quả phân tích ứng suất	22
Hình 2.14 Kết quả phân tích chuyển vị	23
Hình 2.15 Kết quả phân tích hệ số an toàn	23
Hình 2.16 Kết quả phân tích ứng suất	24
Hình 2.17 Kết quả phân tích chuyển vị	24
Hình 2.18 Kết quả phân tích hệ số an toàn	25
Hình 3.1 Jetson Nano Developer Kit	26
Hình 3.2 Mô hình 3D và các cổng Input/Output	27
Hình 3.3 Cổng input thẻ nhớ	28
Hình 3.4 RPLiDAR A1M8	31
Hình 3.5 Hoverboard Hardware	32
Hình 3.6 Sơ đồ các chân Hoverboard Hardware	32
Hình 3.7 Sơ đồ chân STM32F103RCT6	33
Hình 3.8 Mạch phát âm thanh DY-SV5W	35
Hình 3.9 Loa toàn dải 2 inch - 6Ω 5W	36
Hình 3.10 Nút dừng khẩn cấp	37
Hình 3.11 Pin	38
Hình 3.12 Màn Hình LCD 7 inch Cảm ứng HDMI	38
Hình 3.13 Webcam Full HD 1080P	40

Hình 3.14 Bản vẽ mạch điện	41
Hình 4.1 FILESYSTEM LEVEL của ROS	42
Hình 4.2 Các thành phần chính của ROS	43
Hình 4.3 Một phần dữ liệu nhận được từ Lidar	44
Hình 4.4 Rviz mô phỏng lại dữ liệu nhận được từ scan	45
hình 4.5 Tổng quán về Hector Mapping and Navigation stack	46
Hình 4.6 Hình ảnh bản đồ xây dựng từ thuật toán Hector	46
Hình 4.7 Sơ đồ khối Navigation Stack	47
Hình 4.8 Vận tốc cho phép V_a trong DWA	49
Hình 4.9 Vận tốc trong cửa sổ động V_d trong DWA	50
Hình 4.10 Heading của robot trong DWA	50
hình 4.11 Sơ đồ nút và trọng số trên mỗi cạnh đồ thị	52
Hình 4.12 Cấu trúc của Yolo	55
Hình 4.13 Nguyên lý hoạt động của YOLO	55
Hình 4.14 Hàm tính IoU	57
Hình 4.15 Ví dụ về Localization loss	58
Hình 4.16 Công thức ước lượng bounding box từ anchor box	59
Hình 4.17 Non-max suppression	60
Hình 4.18 Tập dữ liệu về hình ảnh để thực hiện quá trình train	61
Hình 4.19 Quá trình gán nhãn cho tập dữ liệu trên labelImg	61
Hình 4.20 Label của các ảnh sau khi gán nhãn	62
Hình 4.21 Setup thư viện và tải gói pretrain weights	62
Hình 4.22 Giải nén file dataset	62
Hình 4.23 Tạo file yaml	62
Hình 4.24 Quá trình huấn luyện	63
Hình 4.25 Kết quả của YOLO trong việc phát hiện đối tượng trong real-time	63
Hình 4.26 Mediapipe hand nhận diện khớp ngón tay	64
hình 4.27 điểm nhận dạng trên bàn tay của mô hình landmark	64
hình 4.28 SSD: Single Shot MultiBox Detector	66
hình 4.29 Kiến trúc mạng SSD dựa trên base là VGG-16	66
Hình 4.30 21 điểm tọa độ khớp ngón tay sau khi nhận dạng qua mô hình.	67
Hình 4.31 Mô hình phân loại cử chỉ tay với 2 lớp hidden Layers	68
hình 4.32 Gắn nhãn cho dữ liệu	69
hình 4.33 Ghi dữ liệu bằng video.	69
hình 4.34 Tọa độ các điểm của keypoint với điểm gốc tọa độ là cổ tay ID 0.	70
hình 4.35 Quá trình train	70
Hình 4.36 Lưu đồ hoạt động điều khiển robot bằng cử chỉ tay	70

hình 4.37 Giao tiếp giữa các node, topic trong rqt_Graph Ros	71
hình 4.38 Kết quả điều khiển Robot bằng cử chỉ tay.	72
hình 5.1 Web xây dựng từ html	73
hình 5.2 Các dạng css selector	77
hình 5.3 Trang quản lý order cho nhân viên	80
hình 5.4 Trang gọi món dành cho khách hàng	80
hình 5.5 Quá trình debug và chạy máy ảo	81
hình 5.6 Giao diện công cụ chuẩn đoán Flutter Doctor	83
hình 5.7 Cài đặt plugin Dart	84
hình 5.8 Cài đặt plugin Flutter	85
hình 5.9 Khởi động lại IDE	85
hình 5.10 Giao diện khách hàng, giao diện nhân viên	86
hình 5.11 Mô tả Rosbridge	87
hình 5.12 Ví dụ về dữ liệu JSON	87
Hình 5.13 Hoạt động của Websocket	88
hình 5.14 Các thuộc tính của websocket	88
hình 5.15 Các node/topic giao tiếp với rosbridge	89
hình 5.16 Lưu đồ giải thuật giao hàng bằng web	90
Hình 5.17 Khách hàng chọn món	91
Hình 5.18 Nhân viên đăng nhập xác nhận giao đồ	91
Hình 5.19 Mô phỏng kết quả giao đồ tới từng bàn.	92
hình 5.20 Giao diện nhân viên	93
hình 5.21 Giao diện khách hàng	94
hình 5.22 Giao diện với bảng các đồ uống đã chọn	94

DANH MỤC BẢNG BIỂU

Bảng 2.1 Thông số kỹ thuật của động cơ	14
Bảng 2.2 Các bước thực hiện tạo một study mới	17
Bảng 2.3 Các bước thiết lập thống số vật liệu	18
Bảng 2.4 Các bước thiết lập các ràng buộc cố định	19
Bảng 2.5 Các bước đặt lực lên khung xe	20
Bảng 2.6 Các bước tạo MESH	21
Bảng 3.1 Hiệu suất của Jetson Nano khi chạy các mạng ML so với các nền tảng khác	29
Bảng 3.2 Sơ đồ chân trên Jetson Nano	30
Bảng 4.1 Kết quả sai số trên thực nghiệm	54

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1 Giới thiệu.

Chương này sẽ giới thiệu tổng quan về robot tự hành, các thành phần và ứng dụng của robot tự hành, yêu cầu về thiết kế và kỹ thuật của robot.

1.2 Tổng quan về robot tự hành.

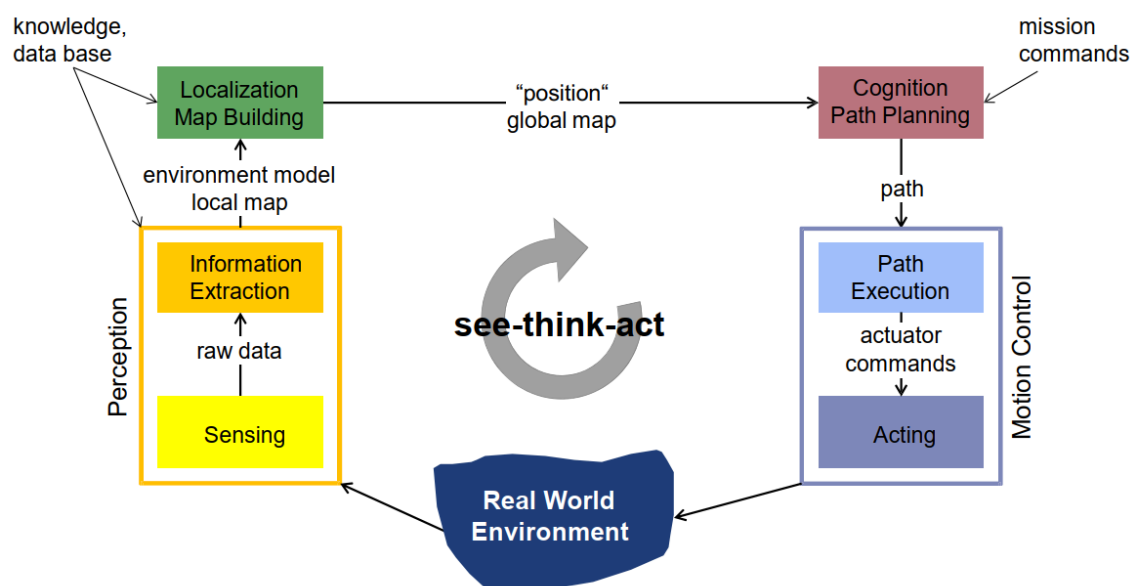
Robot tự hành là một loại robot có khả năng đưa ra quyết định và thực hiện hành động mà không cần sự can thiệp của con người. Chúng có khả năng nhận thức môi trường xung quanh và dựa trên thông tin đó để lựa chọn và thực hiện các hành động phù hợp. Trong những năm gần đây, các công nghệ như máy học, học sâu và trí tuệ nhân tạo đã được áp dụng rộng rãi vào robot tự vận hành, đặc biệt là trong lĩnh vực robot di động và ứng dụng công nghiệp.

Một số ví dụ về các loại robot tự vận hành bao gồm:

- + Xe tự hành (Automation Guided Vehicle - AGV): Được sử dụng để vận chuyển vật liệu trong nhà máy và nhà kho một cách tự động.
- + Máy bay không người lái: Được sử dụng để ứng phó với các thảm họa hoặc nhiệm vụ khó khăn trong môi trường không thể tiếp cận được bằng con người.
- + Robot tự hành dưới nước: Được sử dụng để tìm kiếm và phát hiện các con tàu đắm ở những vùng sâu nhất của đại dương.
- + Robot dạng người: Được sử dụng trong các khu vực nguy hiểm như tác vụ cứu hộ hoặc làm việc trong môi trường gần liền với con người.
- + Robot chó tự hành: Được sử dụng nhằm mục đích giải trí cho con người.
- + Robot phục vụ tự hành: Được sử dụng trong nhà hàng hoặc quán cà phê để hỗ trợ người trong việc phục vụ khách hàng.

Việc sử dụng robot tự vận hành mang lại nhiều lợi ích cho con người. Chúng giúp giải phóng lao động khỏi những nhiệm vụ nhàm chán, nguy hiểm hoặc bẩn thỉu, để con người có thể tập trung vào các công việc sáng tạo và có giá trị hơn. Các sản phẩm robot tự vận hành, như robot hút bụi và lau nhà, đã trở thành những thiết bị không thể thiếu trong cuộc sống hiện đại của chúng ta, mang lại sự tiện ích và tiết kiệm thời gian cho gia đình và cá nhân.

1.3 Các thành phần của robot tự hành.



Hình 1.1 Các thành phần của robot tự hành

1.3.1 Nhận thức của robot (see).

Đối với con người, nhận thức được thể hiện qua các giác quan (mắt, mũi, da...). Với robot tự hành cũng vậy, giác quan của robot chính là các cảm biến như camera 3D (mắt), Lidar, Ultrasonic Sensor, Encoder, IMU... Các cảm biến giúp robot có thể cảm nhận môi trường thực tế để đưa ra quyết định chính xác hơn.

1.3.2 Quyết định của robot (think).

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

1.3.3 Hành động của robot (act).

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

1.4 Ứng dụng của robot tự hành.

1.4.1 Robot tự hành trong kho hàng.



Hình 1.2 Robot tự hành trong kho hàng

Robot tự hành trong nhà kho là các hệ thống robot được thiết kế để tự động hoá quá trình di chuyển và vận chuyển hàng hóa trong môi trường nhà kho. Chúng được sử dụng để tăng cường hiệu suất và hiệu quả của các hoạt động nhà kho, giảm thiểu sự phụ thuộc vào lao động con người và giảm nguy cơ sai sót.

1.4.2 Robot tự hành dùng trong lĩnh vực y tế.



(a)



(b)



(c)



(d)

Hình 1.3 Robot tự hành dùng trong lĩnh vực y tế

Robot tự hành trong lĩnh vực y tế là một loại robot được thiết kế để tự động hoá và hỗ trợ các tác vụ y tế trong môi trường bệnh viện và các cơ sở chăm sóc sức khỏe. Robot này có khả năng thực hiện các công việc như di chuyển, giao tiếp, thu thập dữ liệu, phân tích và cung cấp hỗ trợ cho các chuyên gia y tế.

Các lợi ích của việc sử dụng robot tự hành trong lĩnh vực y tế bao gồm tăng cường hiệu quả công việc, giảm thiểu sai sót, giảm thiểu sự tiếp xúc giữa con người và các chất độc hại và nâng cao chất lượng chăm sóc sức khỏe. Tuy nhiên, việc triển khai robot tự hành trong lĩnh vực y tế còn đối mặt với thách thức về độ tin cậy, an toàn và đạo đức trong việc áp dụng công nghệ này.

1.4.3 Robot tự hành dùng trong lĩnh vực quân sự.



Hình 1.4 Robot tự hành dùng trong lĩnh vực quân sự

Robot tự hành trong lĩnh vực quân sự là một công nghệ đột phá được sử dụng để tăng cường khả năng quân sự và thực hiện các nhiệm vụ quân sự mà người không thể thực hiện hoặc làm mất quá nhiều rủi ro. Robot tự hành trong lĩnh vực quân sự được thiết kế để hoạt động độc lập hoặc được điều khiển từ xa bởi các nhà quân sự để thực hiện nhiệm vụ quân sự.

Việc sử dụng robot tự hành trong lĩnh vực quân sự có nhiều lợi ích như giảm thiểu nguy hiểm cho người lính, tăng cường sức mạnh quân sự, và giảm thiểu số lượng binh sĩ cần thiết trong các nhiệm vụ nguy hiểm. Tuy nhiên, việc triển khai robot tự hành trong lĩnh vực quân sự cũng đặt ra nhiều thách thức về độ tin cậy, bảo mật và đạo đức trong việc sử dụng công nghệ này.

1.5 Đặt vấn đề.

Sử dụng robot phục vụ trong nhà hàng mang lại nhiều lợi ích và tiềm năng trong việc cung cấp dịch vụ tốt hơn và nâng cao trải nghiệm của khách hàng. Dưới đây là một số lý do vì sao sử dụng robot phục vụ trong nhà hàng:

- + Nâng cao hiệu suất: Robot phục vụ có khả năng làm việc liên tục mà không cần nghỉ ngơi, không mắc sai sót như con người và không bị ảnh hưởng bởi tình trạng mệt mỏi. Điều này giúp tăng hiệu suất làm việc và đảm bảo dịch vụ được cung cấp một cách liên tục và nhanh chóng.

- + Giảm thiểu sai sót: Robot phục vụ thực hiện các tác vụ một cách chính xác và không gặp những vấn đề như lỗi đơn đặt hàng, việc đem nhầm món ăn hay việc tính toán sai hóa đơn. Điều này giúp giảm thiểu sai sót và tăng cường chất lượng dịch vụ.

- + Tạo trải nghiệm độc đáo: Sử dụng robot phục vụ trong nhà hàng tạo ra một trải nghiệm mới mẻ và hấp dẫn cho khách hàng. Robot có thể có thiết kế độc đáo, di chuyển một cách linh hoạt và tương tác với khách hàng một cách thú vị. Điều này giúp tạo ra sự khác biệt và gây ấn tượng cho khách hàng.

- + Tăng cường sự an toàn và vệ sinh: Robot phục vụ giúp giảm tiếp xúc trực tiếp giữa nhân viên và thực phẩm, giúp tăng cường vệ sinh và đảm bảo sự an toàn thực phẩm. Đồng thời, robot cũng có thể được lập trình để tuân thủ các quy tắc vệ sinh và an toàn trong việc thực hiện các tác vụ.

- + Giảm chi phí nhân lực: Sử dụng robot phục vụ có thể giảm chi phí nhân lực do không cần trả lương, không cần bảo hiểm và không gặp các vấn đề như mệt mỏi hay nghỉ việc. Điều này giúp giảm chi phí hoạt động của nhà hàng trong dài hạn.

Trên thực tế, cũng đã có rất nhiều nhà hàng đã đưa robot vào trong quá trình phục vụ để tiết kiệm chi phí cũng như tăng năng suất, ngoài ra còn đem lại trải nghiệm mới lạ và thú vị cho khách hàng. Hình 1.5 mô tả robot dạng người sử dụng trong các nhà hàng tại Pháp.



Hình 1.5 Robot dạng người phục vụ trong nhà hàng ở Pháp

Việc sử dụng robot ở Việt Nam đã bắt đầu nhận được sự quan tâm và phát triển trong nhiều lĩnh vực khác nhau, tuy nhiên chủ yếu trong lĩnh vực công nghiệp hoặc phục vụ trong gia đình, ngược lại robot phục vụ trong các nhà hàng còn khá ít và khá mới mẻ. Tuy nhiên, với sự phát triển của công nghệ và nhu cầu tăng cường hiệu quả công việc, dự kiến sẽ có nhiều cơ hội và tiềm năng cho việc sử dụng robot trong tương lai.

Với những tính năng và ưu điểm như trên, robot hoàn toàn có thể thay thế con người trong phục vụ nhà hàng. Do vậy, đề tài này hoàn toàn là khả thi và có thể dẫn đầu xu hướng trong tương lai.

1.6 Đề xuất mô hình robot.

Robot được thiết kế để có thể thực hiện 3 nhiệm vụ sau:

- + Nhiệm vụ 1: Xây dựng bản đồ phòng ăn:
 - Nhân viên điều khiển robot đi xung quanh phòng để lập bản đồ phòng cho robot.
 - Robot lưu trữ bản đồ phòng vào thư mục để sử dụng cho các lần sau.
 - Đánh số bàn trên bản đồ đã vẽ.
- + Nhiệm vụ 2: Vận chuyển các món ăn từ khu vực quầy, bếp ra đến các bàn theo yêu cầu của khách hàng.
 - Vị trí thường trực của robot là ở gần các quầy. Robot đã lưu trữ bản đồ của phòng được xây dựng trước trong thư mục.
 - Chu trình làm việc của robot bắt đầu khi có khách hàng gọi món ăn, bàn ăn tại web app.
 - Nhân viên đặt đồ ăn lên robot ,nhấn xác nhận qua GUI trên robot hoặc trên web,app.
 - Robot di chuyển đến bàn ăn, trên GUI của robot và web,app có hiện trạng thái: “Đồ ăn đã đến mời quý khách xác nhận” và có hệ thống loa thông báo trên robot để thông báo tới khách hàng.
 - Khi tới nơi khách hàng có thể điều khiển robot bằng cử chỉ tay để tới vị trí thuận tiện nhất lấy đồ ăn và nhấn xác nhận trên màn hình robot hoặc web,app.
 - Sau khi xác nhận xong robot sẽ di chuyển tới bàn khác lặp lại chu trình trên và trở về quầy nếu giao xong và thông báo với nhân viên qua web,app.
- + Nhiệm vụ 3: Nhận diện và phân loại đồ uống.
 - Robot sẽ nhận dạng các loại đồ uống bao gồm: Cocacola, Pepsi, nước lọc,...

CHƯƠNG 2. THIẾT KẾ MÔ HÌNH ROBOT

2.1 Giới thiệu.

Khi thiết kế một robot, đầu tiên chúng ta sẽ thiết kế phần cơ khí của robot. Chương này sẽ trình bày việc tính chọn hình dạng, kích thước của robot.

2.2 Yêu cầu kỹ thuật.

Đối với robot phục vụ nhà hàng vấn đề đặt ra là sự tự định hướng bằng các cảm biến, camera, ngoài ra còn phải tương tác với khách hàng. Tìm ra đường đi ngắn, đảm bảo phục vụ kịp thời. Hơn nữa phải đáp ứng được khối lượng công việc thay thế cho 2-3 nhân viên. Phục vụ được nhiều bàn, nhiều khách hàng cùng một lúc, điều mà một nhân viên không thể làm được.

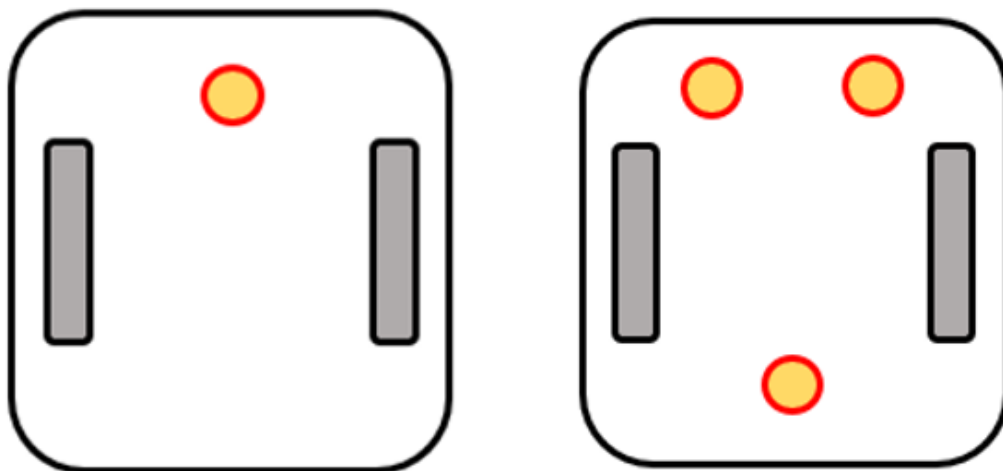
Để đáp ứng được những vấn đề ở trên thì robot phải đáp ứng được:

- + Robot phải chịu được tải trọng lớn.
- + Kích thước nhỏ gọn để đảm bảo di chuyển dễ dàng trong nhà hàng cũng như tránh va chạm với con người. (kích thước 519x516x1282mm)
- + Tốc độ robot phải đủ nhanh để phục vụ cho khách hàng nhưng không được quá nhanh nếu không nước uống sẽ bị rơi. (tốc độ khoảng 0.5 – 1.2 m/s)
- + Đảm bảo vệ sinh an toàn thực phẩm, sử dụng loại vật liệu phù hợp.
- + Robot có khả năng xây dựng bản đồ của nhà hàng.
- + Robot có khả năng tìm đường đến đích nhanh trong khi tránh vật cản.
- + Tương tác với khách hàng một cách thân thiện thông qua giao diện và loa thông báo.

2.3 Thiết kế cơ khí.

2.3.1 Lựa chọn kết cấu của mô hình xe giao hàng.

a) Giới thiệu robot 2 bánh (Differential Drive Robot - DDR).

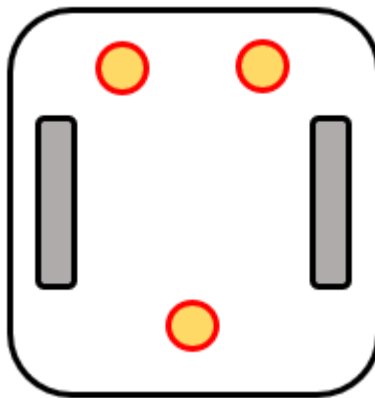


Hình 2.1 Hình ảnh một số loại robot DDR

Robot DDR là một loại mobile robot di chuyển nhờ 2 động cơ riêng biệt được đặt ở 2 bên của robot. Nó có thể thay đổi hướng nhờ việc thay đổi tốc độ của 2 bánh. Thông thường robot loại này thường có thêm từ 1 hoặc nhiều hơn bánh phụ (bánh bị động) hay còn gọi là bánh dẫn hướng dùng để dẫn hướng và giữ thẳng bằng cho robot.

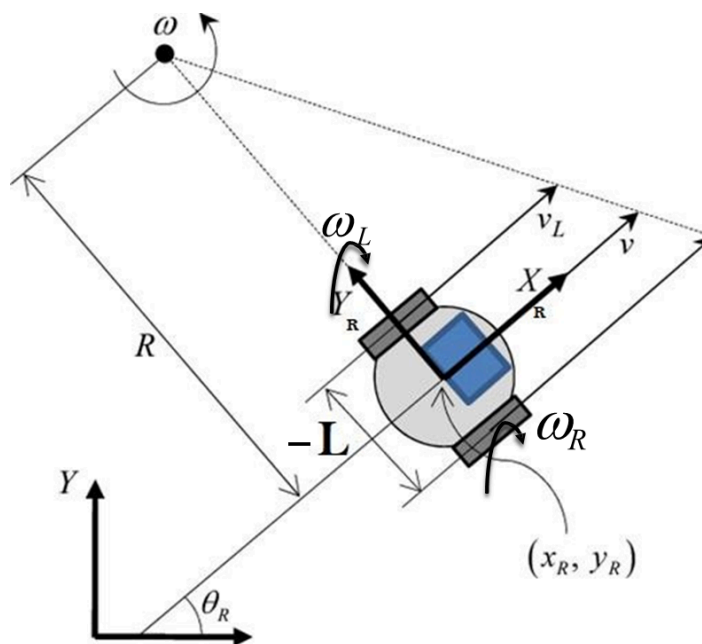
Ở hình trên, 2 bánh màu đen là 2 bánh chủ động, được gắn với động cơ để điều khiển robot. Bánh màu vàng là bánh dẫn hướng. Các robot có thêm 3 bánh dẫn hướng sẽ giữ thẳng bằng tốt hơn tuy nhiên 5 bánh sẽ khó đồng phẳng nhau hơn so với loại 3 bánh, nhưng với nhiệm vụ giao đồ ăn thức uống cần phải thẳng bằng tốt, ổn định tốc độ, không xô lệch thì 5 bánh là lựa chọn tối ưu hơn.

b) Cấu trúc robot.



Hình 2.2 Mô hình thiết kế của nhóm

Dựa vào những đặc điểm về cách thiết kế khung xe và bánh xe như trên thì ta có sơ đồ robot như sau:



Hình 2.3 Mô hình động học của robot

Robot bao gồm 2 bánh chủ động và 1 bánh dẫn hướng như trên:

- + (x_R, y_R, θ_R) là vị trí và góc quay của robot so với hệ tọa độ cố định.
- + L là khoảng cách giữa 2 bánh xe.
- + d, r tương ứng là đường kính và bán kính của bánh xe.
- + R là bán kính quay từ tâm vận tốc tức thời ICC đến tâm robot.

Động học thuận:

- + Đầu vào: Vận tốc góc của 2 bánh xe w_L, w_R
- + Đầu ra: Vận tốc góc và vận tốc dài của robot v, w

$$\text{Ta có : } w = \frac{v}{R} = \frac{v_L}{R-L/2} = \frac{v_R}{R+L/2} = \frac{v_R - v_L}{L} = \frac{(w_R - w_L)r}{L}$$

$$\text{Tương tự ta có: } \frac{v}{R} = \frac{v_L}{R-L/2} = \frac{v_R}{R+L/2} = \frac{v_L + v_R}{2R}$$

$$\Rightarrow v = \frac{v_L + v_R}{2} = \frac{(w_R + w_L)r}{2}$$

Động học ngược :

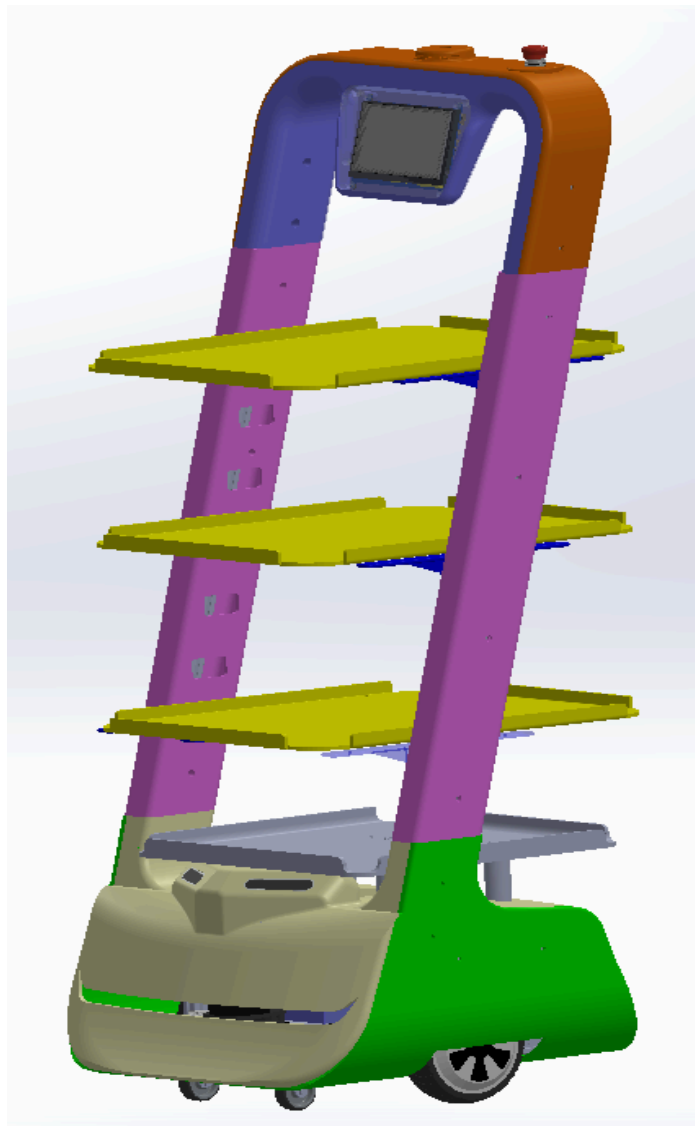
- + Đầu vào: Vận tốc góc và vận tốc dài của robot v, w
- + Đầu ra: Vận tốc góc của 2 bánh xe w_L, w_R

$$\text{Ta có: } w_L = \left(v + \frac{wL}{2}\right) \frac{1}{r}$$

$$w_R = \left(v - \frac{wL}{2}\right) \frac{1}{r}$$

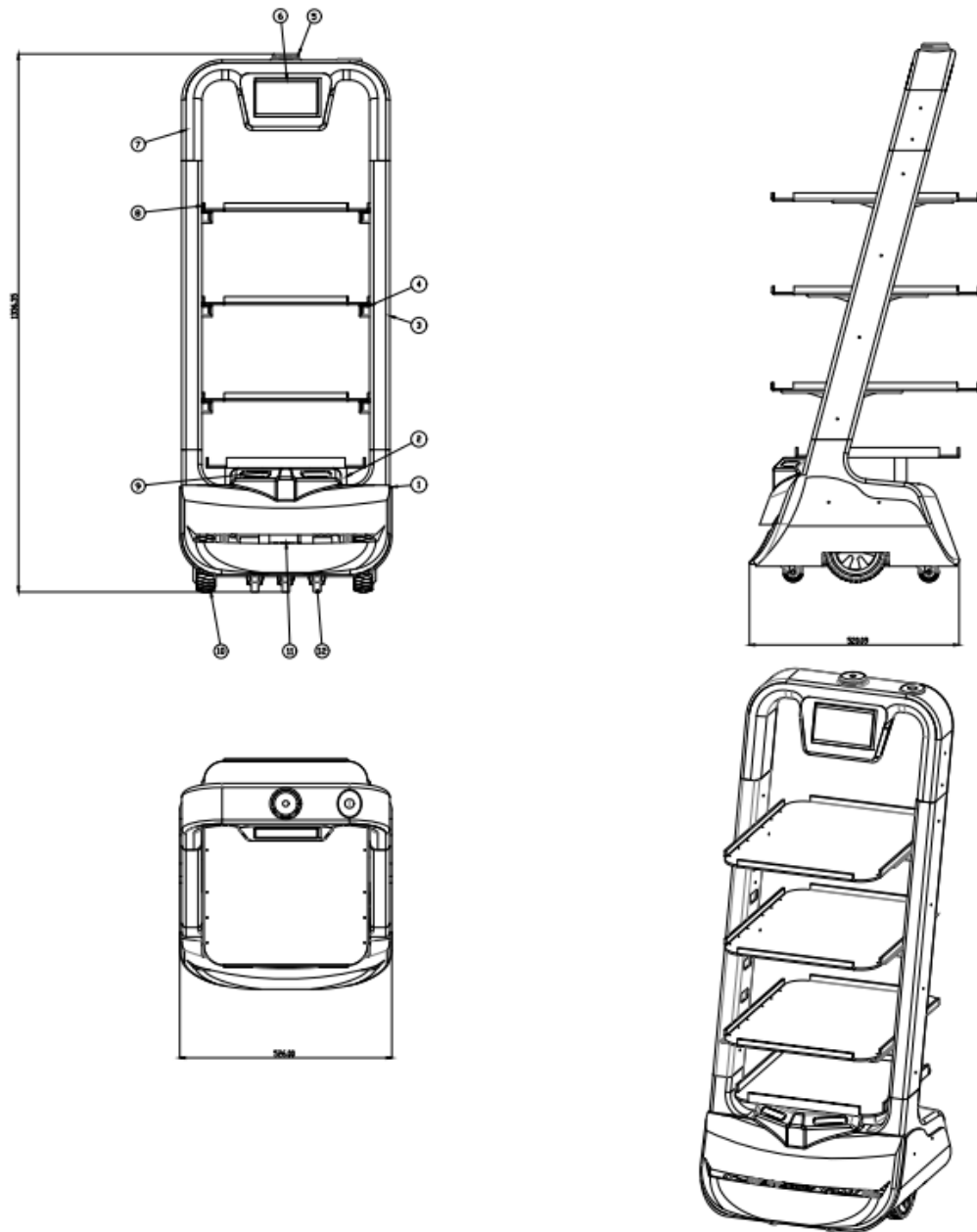
2.3.2 Mô hình xây dựng cho kết cấu đã lựa chọn

a) Mô hình xây dựng



Hình 2.4 Mô hình robot được thiết kế trong Solidworks

b) Bản vẽ cơ khí



Hình 2.5 Bản vẽ kỹ thuật mô hình robot

2.4 Tính toán và lựa chọn động cơ

2.4.1 Thông số yêu cầu

- + Khối lượng robot (bao gồm cả khối lượng bánh xe): $M = 10 \text{ (kg)}$
- + Kích thước robot: 519x516x1282mm
- + Tốc độ tối đa không tải: $v = 0.5 \left(\frac{m}{s}\right)$
- + Khối lượng bánh xe: 1kg x 2
- + Đường kính bánh xe: 165 mm

- + Độ dày bánh xe: 50mm
- + Khả năng chịu tải: 2.8kg / Tray ~ 8 lon coca trên 1 tray.
- + Thời gian sử dụng: 10 – 14 h
- + Thời gian sạc: 4h
- + Hệ số ma sát $\mu_{ms} = 0.3$
- + Hiệu suất $\eta = 0.9$

2.4.2 Tính chọn động cơ 2 bánh dẫn động

- + Lực cản tác động lên robot trong quá trình di chuyển:

$$F = M \times g \times \mu_{ms}$$

Trong đó:

M : tổng khối lượng của robot (kg)

$g = 9.81(m/s^2)$: gia tốc trọng trường

μ_{ms} : ma sát lăn giữa bánh xe và bề mặt chuyển động

- + Momen xoắn T (Nm) của từng động cơ để robot có thể di chuyển:

$$T = \frac{F \times R}{2 \times i \times \eta}$$

Trong đó:

F : lực cản (N)

i : tỷ số truyền

η : hiệu suất

R : bán kính bánh xe chủ động (m)

- + Công suất của động cơ khi có tải P (W):

$$P = \frac{F \times v}{2}$$

Trong đó:

F : lực cản (N)

v : vận tốc lớn nhất của robot (m/s^2)

- + Số vòng quay của động cơ RPM (vòng / phút):

$$RPM = \frac{60 \times v}{\pi \times D}$$

Trong đó:

v : là vận tốc lớn nhất của robot (m/s^2)

D : đường kính của bánh chủ động (m)

Từ các thông số đầu vào ta tính được:

+ Lực cản:

$$F = (M + M_{tai}) \times g \times \mu_{ms} = (10 + 11.2) \times 9.81 \times 0.3 = 62.3916(N)$$

+ Momen xoắn:

$$T = \frac{F \times R}{2 \times i \times \eta} = \frac{62.3916 \times \frac{0.165}{2}}{2 \times 1 \times 0.9} = 2.8596(Nm)$$

+ Công suất động cơ:

$$P = \frac{F \times v}{2} = \frac{62.3916 \times 0.5}{2} = 15.5979 (W)$$

+ Số vòng quay:

$$RPM = \frac{60 \times v}{\pi \times D} = \frac{60 \times 0.5}{\pi \times 0.165} = 57.8745 (vòng/phút)$$

Từ các giá trị tính được ở trên, chúng ta phải chọn động cơ đáp ứng các yêu cầu sau:

- + Momen xoắn đủ lớn.
- + Đáp ứng yêu cầu về vận tốc, khả năng tăng tốc cũng như khả năng hãm động cơ.
- + Phù hợp với điện áp sử dụng.
- + Thích hợp với môi trường làm việc, làm việc chính xác, hoạt động yên tĩnh.
- + Giá thành phù hợp.

Từ những yêu cầu trên lựa chọn mẫu **Hoverboard bldc Motor**, được thể hiện ở hình 2.6 ở dưới.



Hình 2.6 Hoverboard bldc Motor

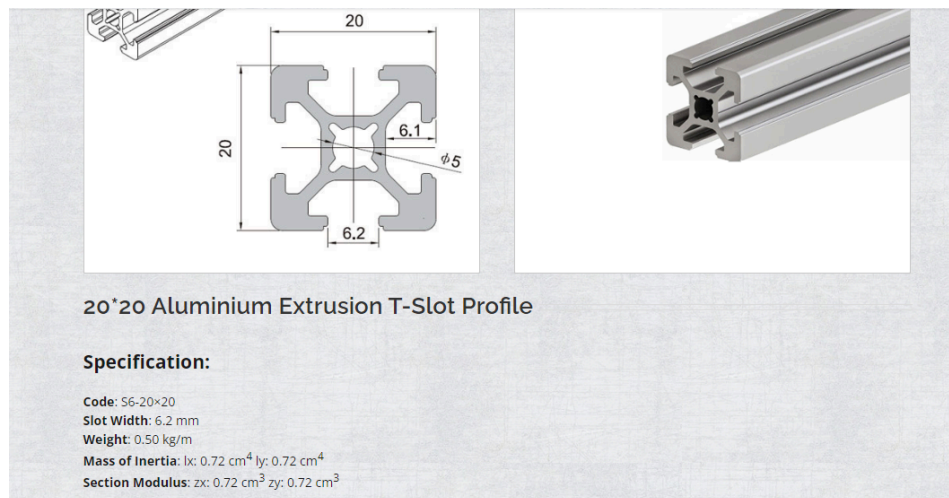
Bảng 2.1 Thông số kỹ thuật của động cơ

Loại động cơ	Không chổi than
Điện áp	36v
Công suất đầu ra	250w
Đường kính trục	13.5mm & 16mm
Chiều dài trục	53mm
Opening size	110mm
Tốc độ tối đa	26km/h
Trọng lượng	~3kg
Momen xoắn	8Nm

2.5 Thiết kế sơ bộ khung và kiểm nghiệm

2.5.1 Thiết kế sơ bộ khung

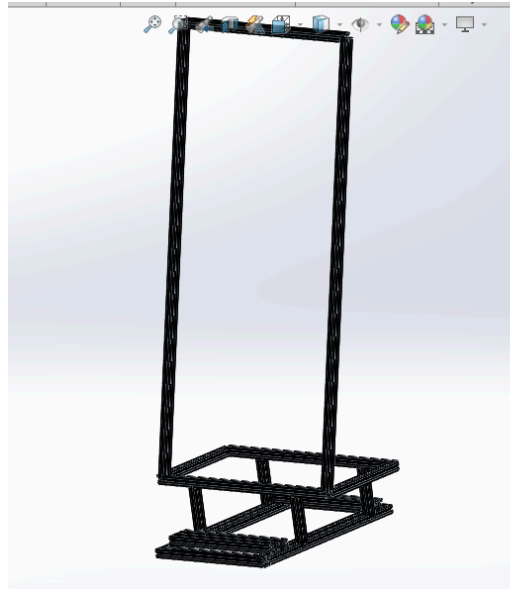
+ Vật liệu: Nhôm định hình 30X30



Hình 2.7 Thông tin về nhôm 20X20

- + Tính năng: Nhôm định hình 20X20
- + Khả năng chống tĩnh điện tốt.
- + Khung nhôm định hình được chế tạo bằng nhôm định hình được anodized hóa sáng bóng bề mặt, chống xước, chịu lực tốt .
- + Độ bền cao, được sản xuất theo phương pháp đùn ép, nguyên liệu là nhôm A6063-T5, với độ nguyên chất cao, đảm bảo cho nhôm định hình công nghiệp có độ bền cao.
- + Khung nhôm định hình với trọng lượng nhẹ có thể tháo lắp, di chuyển dễ dàng.

- + Tính linh hoạt trong lắp ráp: Khung nhôm định hình sử dụng các phụ kiện nhôm định hình như con trượt, ke góc để lắp ráp, vì thế có tính linh hoạt cao, có thể tái sử dụng nhiều lần.



Hình 2.8 Thiết kế khung trên solidworks

2.5.2 Kiểm nghiệm khung

a) Giới thiệu về SolidWorks Simulation

Kiểm nghiệm khung là một quá trình quan trọng trong quá trình thiết kế và sản xuất các thiết bị, đặc biệt là trong ngành công nghiệp. Kiểm tra bền giúp đảm bảo rằng khung xe có đủ khả năng chịu được các tải trọng, áp lực và các yếu tố khác mà nó có thể gặp phải trong quá trình vận hành. Điều này đảm bảo rằng không có sự suy yếu hoặc hỏng hóc trong khung xe.

Hiện nay, Rất nhiều phần mềm có khả năng mô phỏng và kiểm tra bền cho khung máy ở nhiều điều kiện và tải trọng khác nhau, từ đơn giản đến phức tạp. Một vài công cụ, phần mềm có thể kể đến như: ANSYS Mechanical, Abaqus, SolidWorks Simulation, COMSOL Multiphysics, MSC Nastran,... Trong đồ án này, chúng em đã sử dụng SolidWorks Simulation để mô phỏng và kiểm nghiệm bền cho khung xe.

SolidWorks Simulation là một phần mềm mô phỏng và phân tích kỹ thuật được tích hợp trong nền tảng SolidWorks CAD 3D. Nó cung cấp các công cụ mạnh mẽ để mô phỏng và kiểm tra bền cho các sản phẩm và cấu trúc khác nhau.

Với SolidWorks Simulation, chúng ta có thể thực hiện các phân tích cơ bản và nâng cao để đánh giá hiệu suất và độ bền của các thành phần, khung máy, hệ thống máy móc và các sản phẩm khác.

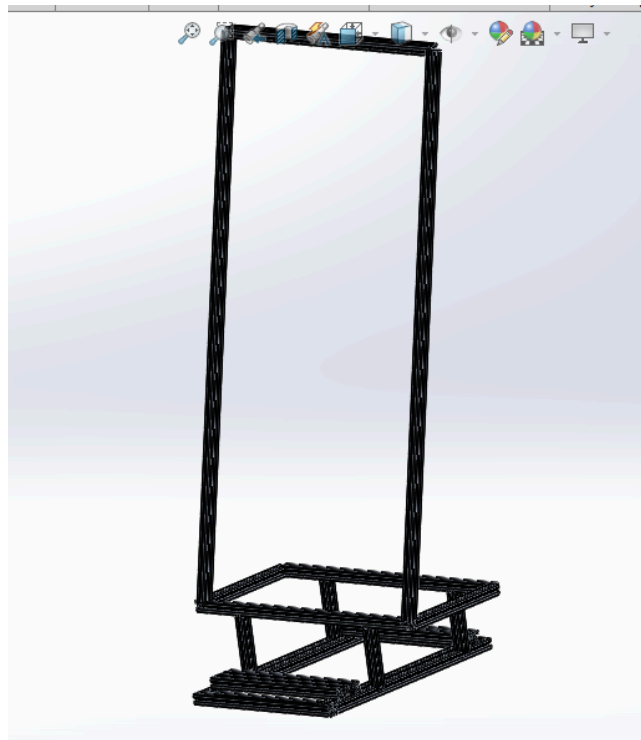
Một trong những ứng dụng chính của SolidWorks Simulation là phân tích cơ học. SolidWorks Simulation cho phép bạn mô phỏng và kiểm tra bền cho các phân tích cơ học, bao gồm tính toán độ bền, biến dạng, tải trọng tĩnh và động. Ta có thể đánh giá các yếu tố như áp lực, lực tác động, tải trọng, độ chịu tải và độ bền tĩnh.

b) Các bước thực hiện kiểm nghiệm bền bằng SolidWorks Simulation.

Để kiểm nghiệm bền cho khung xe, chúng em đã thực hiện lần lượt các bước sau:

Bước 1: Thiết lập mô hình

Sau khi sử dụng các công cụ thiết kế trong solidword để thiết kế khung xe, chúng em được mô hình khung xe 3D như hình 2.9

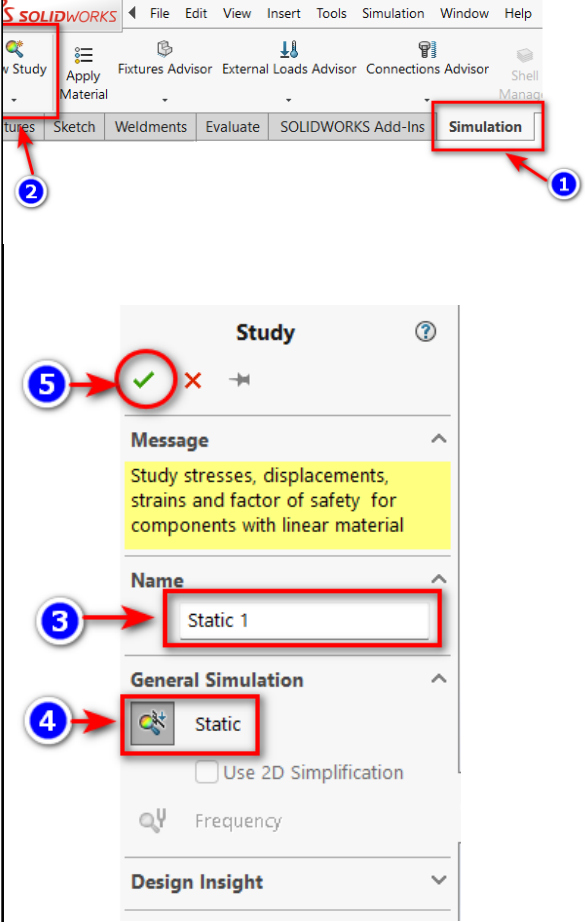


Hình 2.9 Thiết kế khung trên solidworks

Bước 2: Khởi tạo New Study

Khởi tạo một New Study trong SolidWorks Simulation cho phép bắt đầu một phân tích mới hoặc tạo một phân tích bổ sung dựa trên mô hình hiện tại. Khởi tạo một New Study cho phép tùy chỉnh thiết lập phân tích, chọn phương pháp phân tích, áp dụng các điều kiện biên và thiết lập các thông số cụ thể cho phân tích. Điều này giúp tối ưu hóa quy trình phân tích và thu thập kết quả chính xác và đáng tin cậy. Các bước thực hiện được mô tả trong bảng 2.2

Bảng 2.2 Các bước thực hiện tạo một study mới

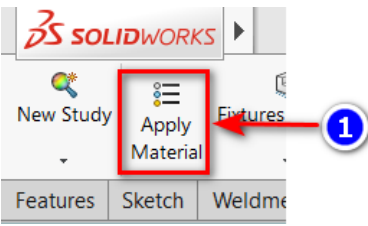
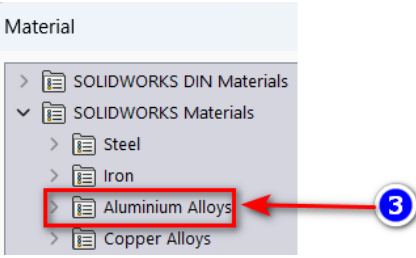
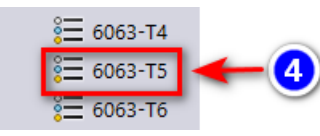
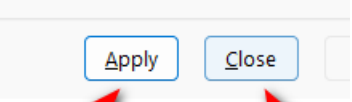
TẠO MỘT STUDY MỚI	
Click vào tab Simulation trên thanh công cụ. Các tùy chọn trong Simulation sẽ hiện lên trên thanh Menu .	
Click vào New Study	
Trong Property Manager , dưới Name , điền tên Study mới.	
Trong General Simulation , click Static .	
Click dấu tích xanh.	
Phần mềm tạo Study trong cây Simulation study.	

Bước 3: Thiết lập thông số vật liệu cho khung xe.

Đặt các thông số vật liệu cho các thành phần trong mô hình là bước rất quan trọng trong kiểm nghiệm bền khung xe. SolidWorks Simulation cung cấp một thư viện vật liệu phong phú để lựa chọn, có thể nhập liệu trực tiếp các thông số vật liệu cụ thể.

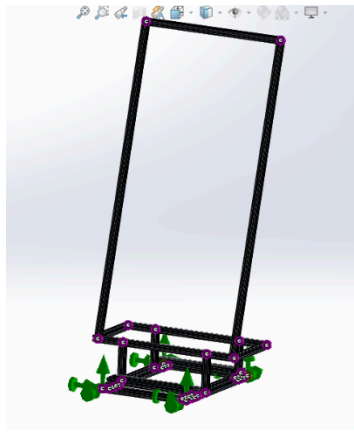
Với nhôm định hình 20X20 được sản xuất theo phương pháp đùn ép, nguyên liệu là nhôm A6063-T5. Nên trong simulation ta sẽ chọn vật liệu là Nhôm 6063-T5. Các thực hiện được mô tả trong 2.3.

Bảng 2.3 Các bước thiết lập thông số vật liệu

THIẾT LẬP THÔNG SỐ VẬT LIỆU	
Trên thanh Menu của tab Simualton , click Apply Material .	
Xuất hiện hộp thoại Material .	
Click Aluminium Alloys , sẽ hiển thị các mã vật liệu thuộc nhóm này.	
Tìm và chọn 6063-T5	
Click Apply .	
Click Close để đóng hộp thoại	

Bước 4: Thiết lập các ràng buộc cố định

Để phân tích tĩnh, ta phải áp dụng đầy đủ những ràng buộc cố định để ổn định mô hình. Đối với kiểm nghiệm bền khung xe, ta sẽ đặt các ràng buộc cố định tại vị trí các bánh xe. Kết quả sau khi đã thiết lập các ràng buộc cố định như Hình 2.10.



Hình 2.10 Sau khi đã thiết lập xong các ràng buộc cố định cho khung xe

Các bước thực hiện được mô tả trong 2.4.

Bảng 2.4 Các bước thiết lập các ràng buộc cố định

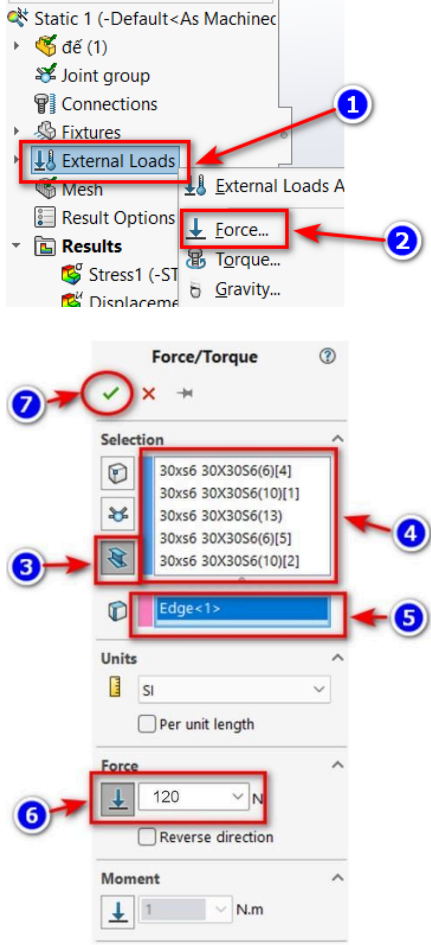
THIẾT LẬP CÁC RÀNG BUỘC CỐ ĐỊNH	
Trong cây Simulation study, click Fixtues. Xuất hiện hộp thoại, Click chọn Fixed Geometry. Trong Standard(Fixed Geometry), click chọn Fixed Geometry. Click vào các vị trí muốn đặt ràng buộc trên khung xe 3D Click vào dấu tích xanh để hoàn tất ràng buộc cố định.	1 → 2 → 3 → 4 → 5

Bước 5: Đặt lực lên khung xe

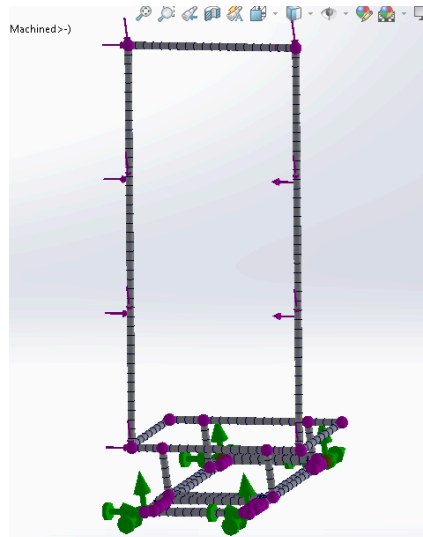
Đặt lực lên mô hình trong SolidWorks Simulation là một cách để mô phỏng và kiểm tra bền các sản phẩm và cấu trúc. Bằng cách áp dụng các lực và tải trọng đúng, ta có thể đánh giá hiệu suất và độ bền của mô hình trong các điều kiện thực tế.

Với khung xe này, tổng khối lượng sản phẩm bên trên tối đa là 12 kg. Từ đó ta xác định được lực tác dụng lên bề mặt phía trên khung xe là 120 N. Để đơn giản, ta sẽ đặt lực phân bố trên các thanh nhôm phía trên khung xe với độ lớn là 120 N. Các bước thực hiện được mô tả trong 2.5.

Bảng 2.5 Các bước đặt lực lên khung xe

ĐẶT LỰC LÊN KHUNG XE	
Trong cây Simulation study, click External Loads. Xuất hiện hộp thoại, click chọn Force. Trong Selection, click Beams. Click vào các thanh dầm cần đặt lực phân bố Chọn một cạnh hoặc một đường làm hướng của lực tác dụng. Click vào Force và điền giá trị của lực. Click vào dấu tích xanh để hoàn tất đặt lực.	

Kết quả sau khi đặt lực phân bố lên khung xe



Hình 2.11 Khung xe sau khi đặt lực phân bố

Bước 6: Tạo Mesh và chạy phân tích.

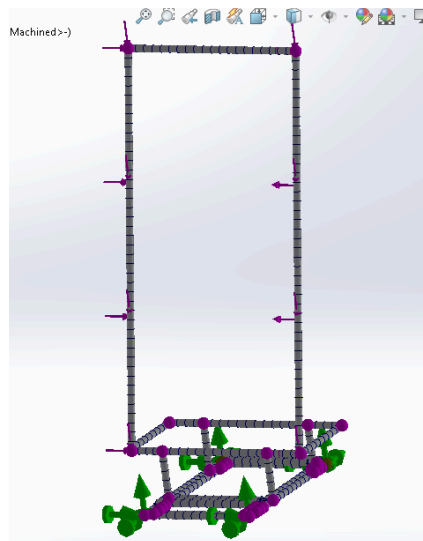
Trong SolidWorks Simulation, tạo mesh (lưới mạng) là một bước quan trọng trong quá trình chuẩn bị mô hình để thực hiện phân tích. Tạo mesh là quá trình chia mô hình thành các phần tử nhỏ hơn để tính toán các giá trị chính xác hơn cho các biến dạng, áp lực, căng thẳng và các thông số khác trong phân tích.

Các bước thực hiện được mô tả trong Bảng 2.6.

Bảng 2.6 Các bước tạo MESH

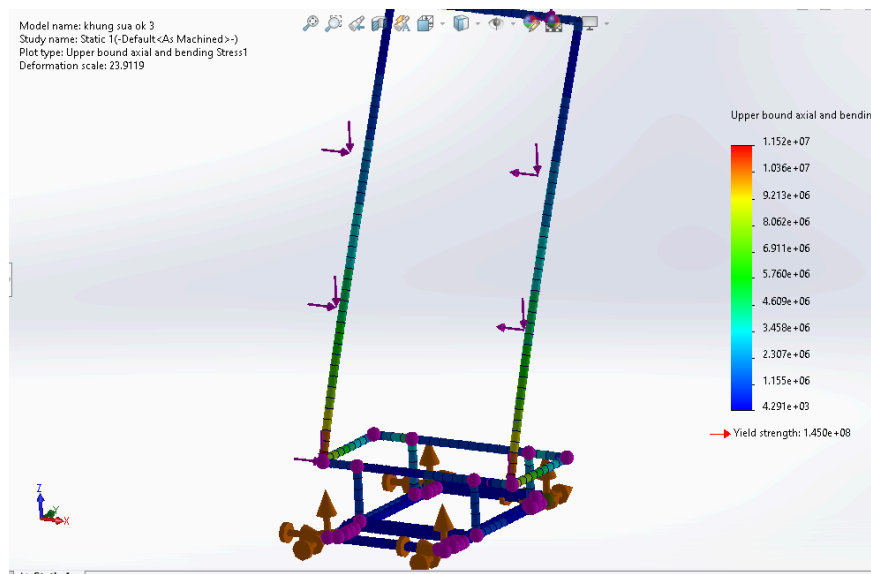
TẠO MESH	
Trong cây Simulation study , click Mesh .	
Xuất hiện hộp thoại, click chọn Create Mesh .	
Phần mềm sẽ tự động chia lưới cho khung xe.	
Trên thanh Menu của tab Simualton , click Run This Study để chạy phân tích.	
Phần mềm sẽ tự động tính toán và trả ra kết quả.	

Kết quả sau khi tạo Mesh cho khung xe Hình 2.12.

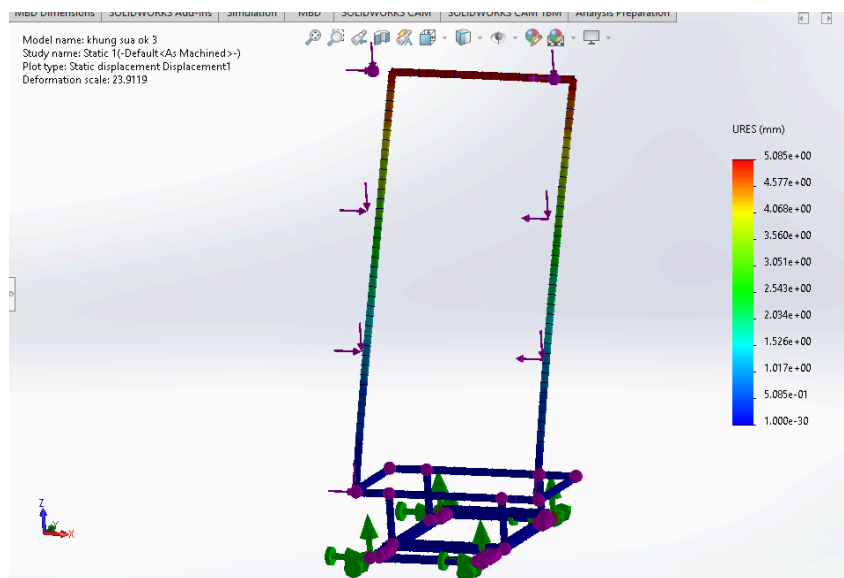


Hình 2.12 Khung xe sau khi được tạo Mesh

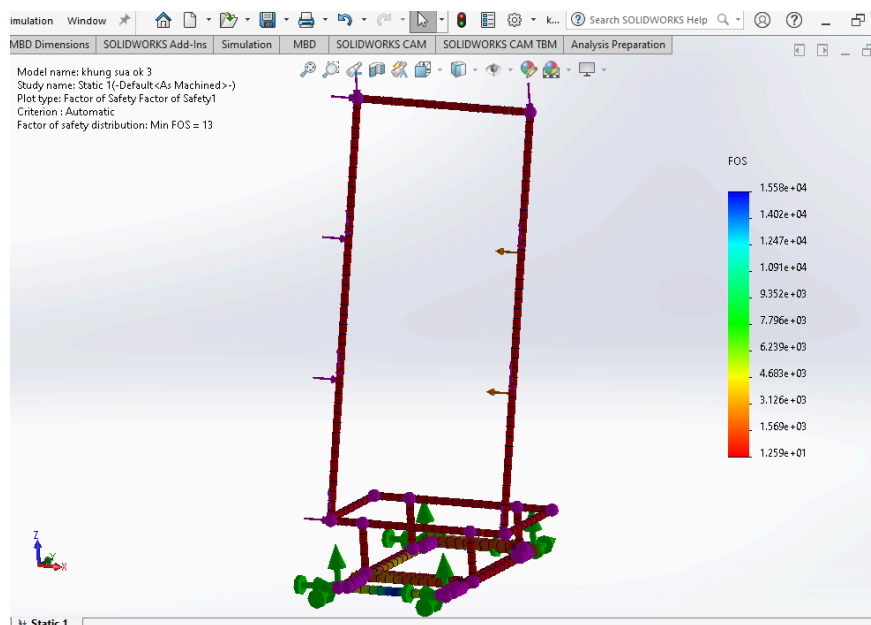
Sau khi click **Run This Study** phần mềm sẽ tự động trả ra kết quả ứng suất, chuyển vị, hệ số an toàn ...



Hình 2.13 Kết quả phân tích ứng suất



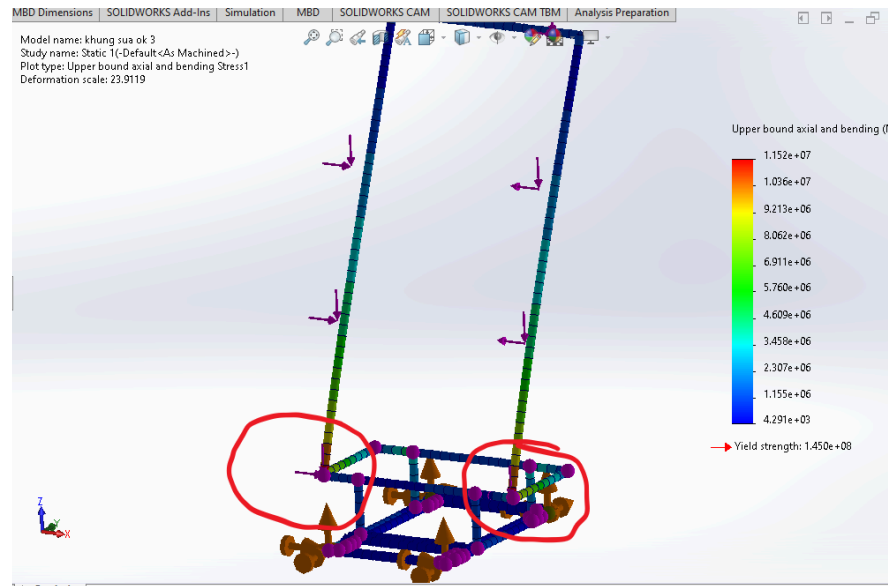
Hình 2.14 Kết quả phân tích chuyển vị



Hình 2.15 Kết quả phân tích hệ số an toàn

d) Phân tích, đánh giá kết quả kiểm nghiệm bền.

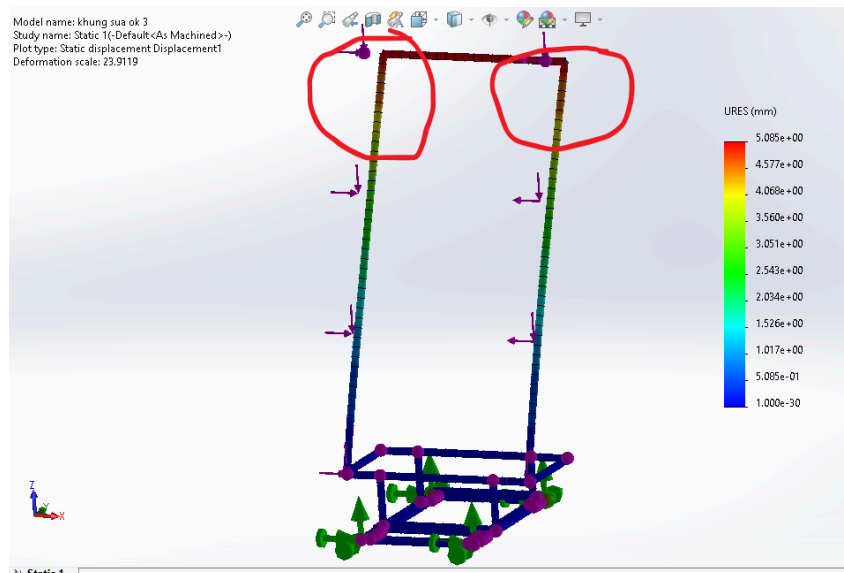
1. Đánh giá, phân tích ứng suất



Hình 2.16 Kết quả phân tích ứng suất

Vị trí dễ bị phá hủy nhất là 2 vị trí được khoanh tròn màu đỏ. Ứng suất tại 2 vị trí này đang là $1,152.10^7$ nhỏ hơn nhiều so với ứng suất cho phép là $1,45.10^8$. Với kết quả này, khung xe đã thỏa mãn bền về ứng suất. Để tăng độ bền tại 2 vị trí này, chúng em đã tăng thêm ke góc để tăng bền cho 2 vị trí này.

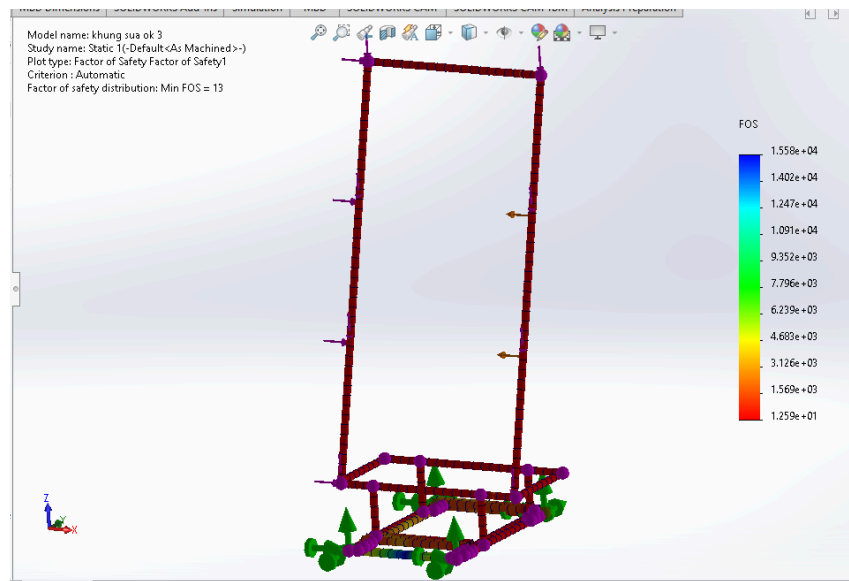
2. Đánh giá, phân tích chuyển vị



Hình 2.17 Kết quả phân tích chuyển vị

Sau khi tác dụng lực, vị trí chuyển vị nhiều nhất được khoanh đỏ. Chuyển vị lớn nhất tại vị trí này là 5 mm. Kết quả này rất nhỏ, có thể chấp nhận được và thỏa mãn bền về chuyển vị.

3. Đánh giá hệ số an toàn của khung xe



Hình 2.18 Kết quả phân tích hệ số an toàn

Từ kết quả trả ra, ta thấy hệ số an toàn nhỏ nhất là $13 > 3$. Với kết quả này, khung xe thỏa mãn điều kiện bền về hệ số an toàn.

Kết luận: Sau khi đánh giá kết quả simulation về ứng suất, chuyển vị, hệ số an toàn thì khung xe đã thỏa mãn điều kiện bền.

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN

3.1 Giới thiệu

Trong chương này, ta sẽ đi xây dựng và lựa chọn các module điều khiển cho hệ thống và giới thiệu về các cấu hình, chức năng của từng phần tử trong hệ thống robot. Từ đó, ta sẽ đi xây dựng và tìm hiểu về các thuật toán điều khiển được ứng dụng.

3.2 Thiết kế hệ thống điều khiển phần cứng

3.2.1 NVIDIA Jetson Nano Developer Kit

a) Giới thiệu về NVIDIA Jetson Nano Developer Kit

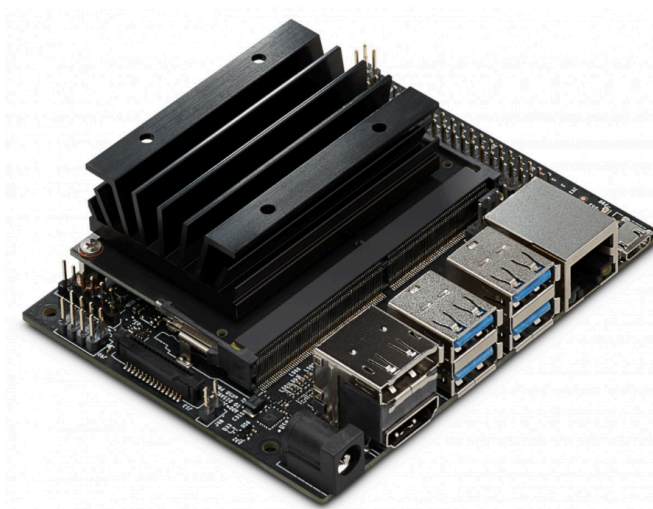
NVIDIA Jetson Nano Developer Kit là một máy tính nhỏ nhưng rất mạnh mẽ cho phép bạn chạy song song nhiều mạng neural sử dụng cho các ứng dụng như phân loại hình ảnh, phát hiện đối tượng, phân đoạn và xử lý giọng nói. Tất cả trong cùng một nền tảng dễ sử dụng cũng như tiêu tốn ít hơn 5 watts.

Jetson Nano cũng cấp 472 GFLOPS để chạy các thuật toán AI hiện đại một cách nhanh chóng, với một CPU quad-core 64-bit ARM, một NVIDIA GPU 128-core được tích hợp trên board mạch, cũng như bộ nhớ 4GB LPDDR4. Có thể chạy song song nhiều mạng neural và xử lý đồng thời một số cảm biến có độ phân giải cao.

Jetson Nano cũng được hỗ trợ bởi NVIDIA JetPack. Bao gồm các gói hỗ trợ board (BSP), CUDA, cuDNN và thư viện phần mềm TensorRT cho deep learning, computer vision, GPU computing, multimedia processing và nhiều ứng dụng khác. SDK cũng bao gồm khả năng cài đặt frameworks Machine Learning (ML) mã nguồn mở như TensorFlow, PyTorch, Caffe / Caffe2, Keras và MXNet, cho phép các nhà phát triển tích hợp các model AI/ framework yêu thích của họ vào các sản phẩm một cách nhanh chóng và dễ dàng.

Đặc biệt: NVIDIA Jetson Nano đã có phiên bản mới nhất B01, có nâng cấp thêm một cổng CSI hỗ trợ sử dụng đồng thời 2 Camera.

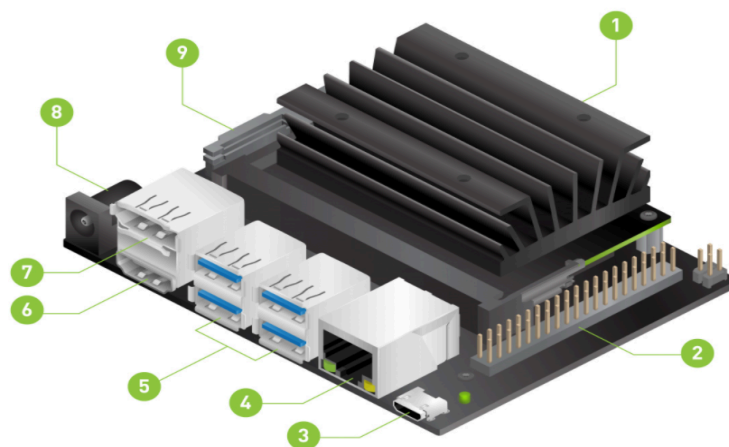
b) Thông số kỹ thuật của Jetson Nano Developer Kit



Hình 3.1 Jetson Nano Developer Kit

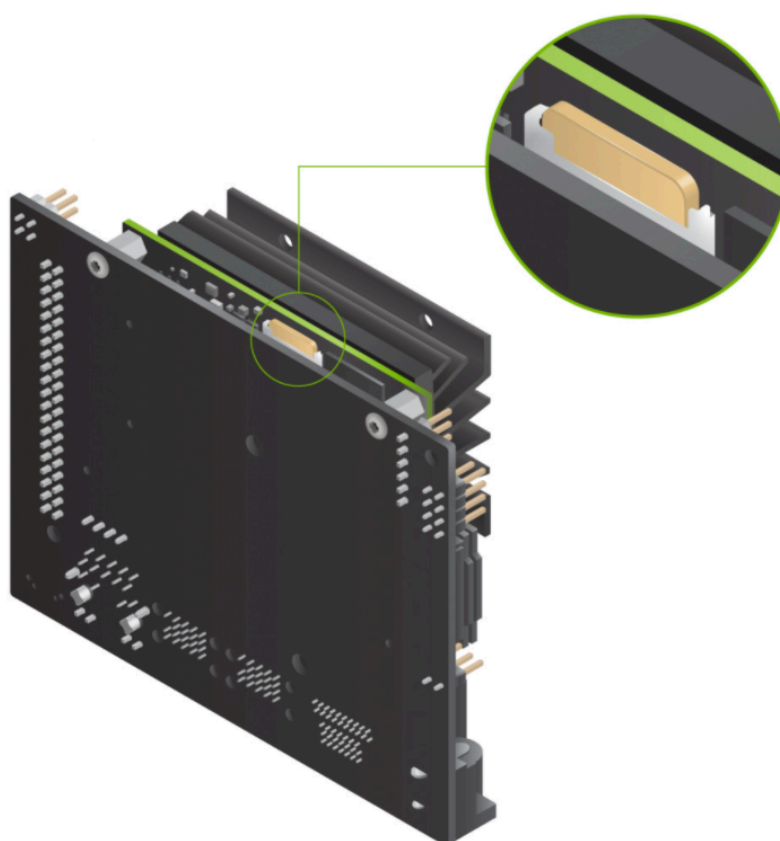
- + GPU: 128-core Maxwell™ GPU
- + CPU: quad-core ARM® Cortex®-A57 CPU
- + RAM : 4GB 64-bit LPDDR4
- + Bộ nhớ : khe cắm thẻ microSD (devkit) hoặc 16GB eMMC flash (Từ nhà sản xuất)
- + Video:
 - Encode: 4K @ 30 (H.264/H.265)
 - Decode: 4K @ 60 (H.264/H.265)
- + Các giao diện :
 - Ethernet: 10/100/1000BASE-T self-negotiation
 - Camera: 12-ch (3x4 OR 4x2) MIPI CSI-2 DPHY 1.1 (1.5Gbps)
 - Display: HDMI 2.0, DP (DisplayPort)
 - USB: 4x USB 3.0, USB 2.0 (Micro USB)
 - Others: GPIO, I2C, I2S, SPI, UART
- + Nguồn cấp :
 - Micro USB (5V 2.5A)
 - DC jack (5V 4A)
- + Kích thước :
 - Core module: 69.6 mm × 45 mm
 - Whole kit: 100mm × 80mm × 29mm

c) Mô hình 3D và các cổng Input/Output của Jetson Nano Developer Kit



-
- 1) microSD card slot for main storage
 - 2) 40-pin expansion header

- 3) Micro-USB port for 5V power input or for data
- 4) Gigabit Ethernet port
- 5) USB 3.0 ports (×4)
- 6) HDMI output port
- 7) DisplayPort connector
- 8) CD Barrel jack for 5V power input
- 9) MIPI CSI camera connector



d) Khả năng xử lý của NVIDIA Jetson Nano so với các loại máy tính nhúng khác

NVIDIA Jetson Nano có thể chạy rất nhiều mạng tiên tiến và mới nhất, bao gồm toàn bộ các framework Machine Learning phổ biến như: Tensorflow, Keras, Pytorch, Caffè/Caffe2, Từ đó có thể áp dụng để xây dựng, thiết kế các mô hình, các thuật toán phức tạp của hệ thống Trí tuệ nhân tạo bằng cách tận dụng khả năng tính toán mạnh mẽ đến từ phần cứng của Jetson Nano để nhận dạng và xử lý hình ảnh, phát hiện, định vị đối tượng.

Bảng 3.1 sau đây sẽ cung cấp đầy đủ hơn hiệu suất của Jetson Nano khi chạy các mạng Machine Learning đồng thời so sánh với các nền tảng khác như Raspberry Pi 3, Intel Neural Compute Stick 2 và Google TPU Coral Dev Board.

Bảng 3.1 Hiệu suất của Jetson Nano khi chạy các mạng ML so với các nền tảng khác

Model	Application	Framework	NVIDIA Jetson Nano	Raspberry Pi 3	Raspberry Pi 3 + Intel Neural Compute Stick 2	Google Edge TPU Dev Board
ResNet-50 [224×224]	Classification	TensorFlow	36 FPS	1.4 FPS	16 FPS	DNR
MobileNet-v2 [300×300]	Classification	TensorFlow	64 FPS	2.5 FPS	30 FPS	130 FPS
SSD ResNet-18 [960×544]	Object Detection	TensorFlow	5 FPS	DNR	DNR	DNR
SSD ResNet-18 [480×272]	Object Detection	TensorFlow	16 FPS	DNR	DNR	DNR
SSD ResNet-18 [300×300]	Object Detection	TensorFlow	18 FPS	DNR	DNR	DNR
SSD Mobilenet-V2 [960×544]	Object Detection	TensorFlow	8 FPS	DNR	1.8 FPS	DNR
SSD Mobilenet-V2 [480×272]	Object Detection	TensorFlow	27 FPS	DNR	7 FPS	DNR
SSD Mobilenet-V2 [300×300]	Object Detection	TensorFlow	39 FPS	1 FPS	11 FPS	48 FPS
Inception V4 [299×299]	Classification	PyTorch	11 FPS	DNR	DNR	9 FPS
Tiny YOLO V3 [416×416]	Object Detection	Darknet	25 FPS	0.5 FPS	DNR	DNR
OpenPose [256×256]	Pose Estimation	Caffe	14 FPS	DNR	5 FPS	DNR
VGG-19 [224×224]	Classification	MXNet	10 FPS	0.5 FPS	5 FPS	DNR
Super Resolution [481×321]	Image Processing	PyTorch	15 FPS	DNR	0.6 FPS	DNR
Unet [1x512x512]	Segmentation	Caffe	18 FPS	DNR	5 FPS	DNR

Kết quả DNR (Did Not Run) xảy ra thường xuyên do dung lượng của bộ nhớ bị hạn chế, các mạng Machine Learning không được hỗ trợ hoặc là bị giới hạn về phần cứng và phần mềm. Như trên bảng thống kê ở trên, Jetson Nano có thể chạy vô số mạng khác nhau với độ phân giải full HD và có FPS khá cao so với các nền tảng máy tính nhúng khác. Bên cạnh đó, kiến trúc CUDA của Jetson Nano còn có thể sử dụng cho lĩnh vực thị giác máy tính, xử lý tín hiệu số, các thuật toán FFT, BLAS, LAPACK, cùng các nhân CUDA do người dùng tự định nghĩa.

e) Giao tiếp GPIO trên Jetson Nano

Jetson Nano là lý tưởng cho các ứng dụng học tập và AI sâu nhưng nó cũng có tiêu đề mở rộng 40 chân mang theo một số chân GPIO có thể được sử dụng để thực hiện các tác vụ cấp vi điều khiển tương tự như các bảng Raspberry Pi.

Bảng 3.2 Sơ đồ chân trên Jetson Nano

Sysfs	Name	Pin	Pin	Name	Sysfs
	3.3V DC	1	2	5V DC	
	I2C_2_SDA	3	4	5V DC	
	I2C_2_SCL	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX	
	GND	9	10	UART_2_RX	
gpio50	UART_2_RTS	11	12	I2S_4_CLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3V DC	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_2_CS0	gpio19
	GND	25	26	SPI_2_CS1	gpio20
	IC2_1_SDA	27	28	I2C_1_SCL	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZO	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Để sử dụng thiết bị đầu cuối để truy cập và sửa đổi các giá trị trên các chân ta ứng dụng thư viện Jetson.GPIO Python để giao tiếp với các thiết bị ngoại vi kết nối với Jetson nano. Gói Linux cho Tegra cung cấp thư viện Jetson.GPIO Python để giao tiếp theo chương trình với các chân GPIO.

Để sử dụng thư viện ta phải cài đặt và setup thư viện trên nền tảng python3 và hoạt động trên quy tắc GPIO udev.

f) Kết luận

Trong đồ án này, ta sử dụng NVIDIA Jetson Nano với vai trò là trung tâm xử lý chính, nhận dữ liệu từ RPLiDAR và camera cũng như truyền tín hiệu xuống mạch điều khiển động cơ.

3.2.2 RPLiDAR A1M8

a) Giới thiệu chung về RPLiDAR A1M8



Cảm biến Laser Radar (Lidar) RPLiDAR A1M8 360° Laser Range Scanner được sản xuất bởi hãng SLAMTEC được sử dụng cho các ứng dụng phát hiện vật cản, lập bản đồ bằng tia Laser trong xe, robot tự hành, hệ thống chống trộm,... cảm biến có độ ổn định và độ chính xác cao.

Cảm biến Laser Radar (Lidar) RPLiDAR A1M8 360° Laser Range Scanner sử dụng giao tiếp UART nên có thể dễ dàng giao tiếp với vi điều khiển, máy tính nhúng hoặc kết nối máy tính qua mạch chuyển USB-UART và phần mềm đi kèm, cảm biến có khả năng quét xa với khoảng cách lên đến 12m, tần số tối đa 10Hz với 8000 samples per time, phù hợp cho vô số các ứng dụng khác nhau.

b) Thông số kỹ thuật của RPLiDAR A1M8

- + Điện áp sử dụng: 5VDC
- + Chuẩn giao tiếp: UART
- + Phương pháp phát hiện vật cản: Laser
- + Khoảng cách phát hiện vật cản tối đa: 12m
- + Góc quay: 360°
- + Tốc độ lấy mẫu tối đa: 8000 Samples per time.
- + Tần số quét tối đa: 10Hz
- + Kích thước: 71 x 97mm

c) Kết luận

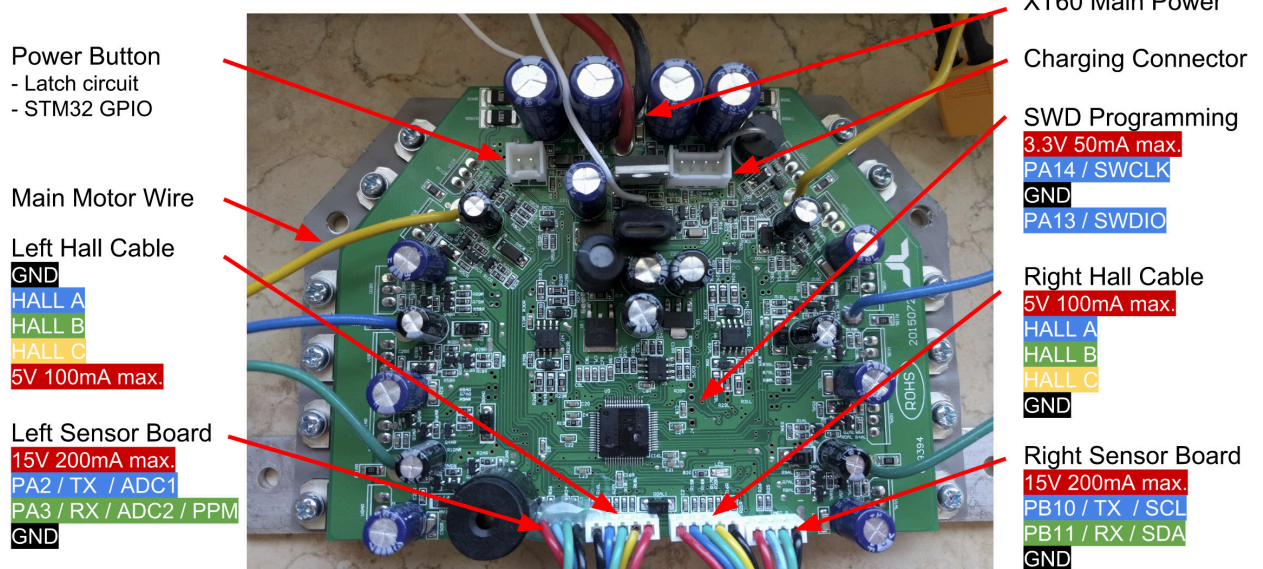
Trong đồ án này, RPLiDAR A1M8 có vai trò quét và lập bản đồ trên không gian 2D, xác định vật cản và định hướng đường đi cho robot và truyền tín hiệu cho Jetson Nano.

3.2.3 Hoverboard Hardware

a) Giới thiệu chung về Hoverboard Hardware

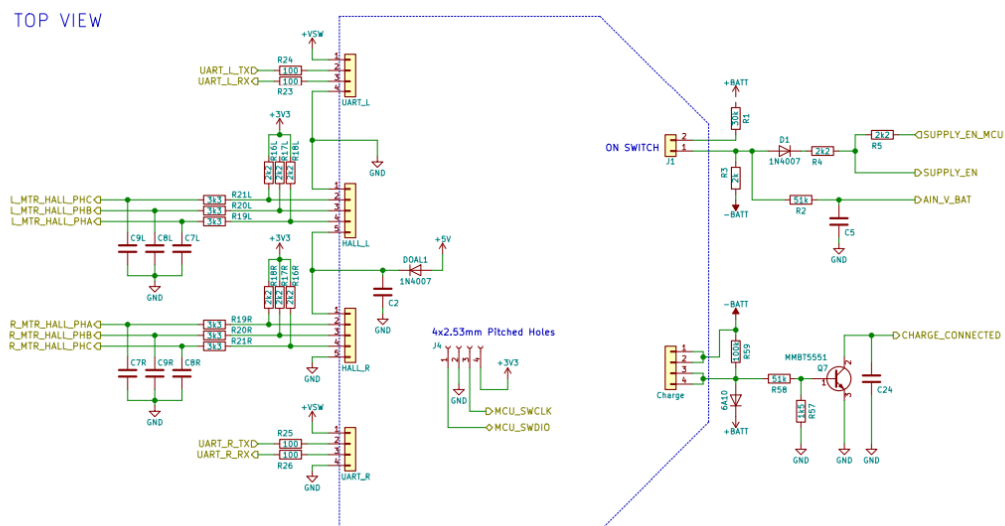
Điều khiển tốc độ và chiều quay của động cơ là 2 nhiệm vụ chính khi động cơ hoạt động. Trong đề tài, nhóm chúng em sử dụng Hoverboard Hardware trong hình 2.26 để giải quyết vấn đề này.

Connectors of the Mainboard



Hình 3.5 Hoverboard Hardware

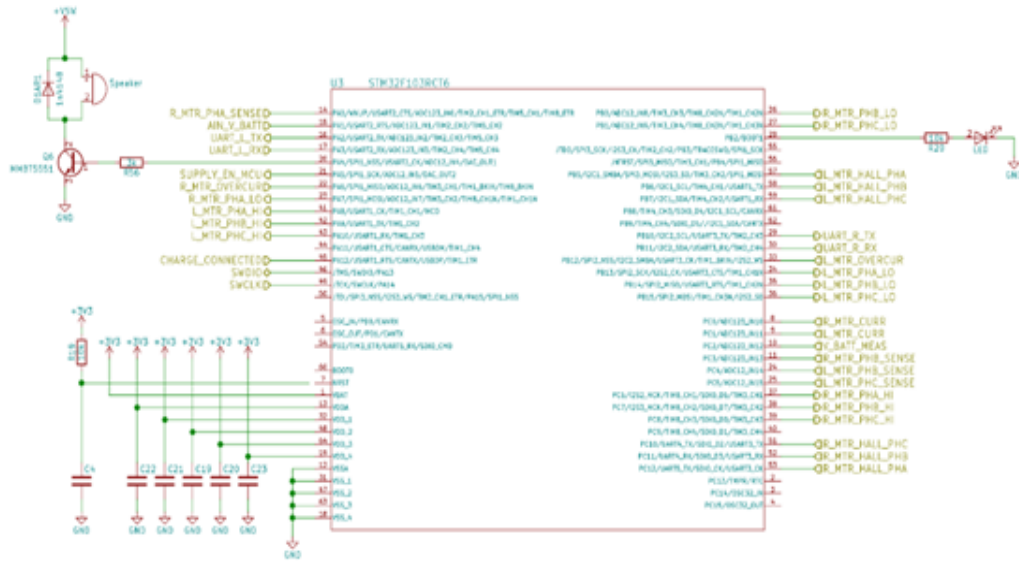
b) Thông số kỹ thuật



Hình 3.6 Sơ đồ các chân Hoverboard Hardware

STM32F103RCT6:

Đây là vi điều khiển chính được sử dụng để lập trình và chạy tác vụ cho hệ thống. Đây là bộ não của hệ thống để điều khiển tất cả các thiết bị khác trên mạch.



Hình 3.7 Sơ đồ chân STM32F103RCT6

Nguồn DC:

Cấp nguồn cho Hoverboard 36VDC qua cổng “XT60 Main Power”.

Chân sensor:

Mạch được tích hợp sẵn 2 cable đọc tín hiệu cảm biến từ 3 pha của 2 động cơ.

Giao tiếp:

Cả 2 cổng USART đều hỗ trợ đầu vào UART, PWM, PBM và iBUS. Ngoài ra, USART2 có thể được sử dụng làm ADC 12bit, trong khi USART3 có thể được sử dụng cho I2C. Lưu ý rằng mặc dù USART3 (cáp bên phải) có khả năng chịu 5V, USART2 (cáp bên trái) không chịu được 5V.

c) Nguyên lý hoạt động

Trong mạch này có sẵn 3 loại điều khiển, nó có thể được đặt trong tệp config.h thông qua tham số CTRL_TYP_SEL:

- + Commutation (COM_CTRL)
- + Sinusoidal (SIN_CTRL)
- + Field Oriented Control (FOC_CTRL) với 3 chế độ điều khiển sau có thể được đặt trong tệp config.h với tham số CTRL_MOD_REQ:
 - CHẾ ĐỘ ĐIỆN ÁP (VLT_MODE): ở chế độ này, bộ điều khiển áp dụng Điện áp không đổi cho động cơ. Được đề xuất cho các ứng dụng rô bốt hoặc các ứng dụng yêu cầu phản ứng động cơ nhanh.
 - CHẾ ĐỘ TỐC ĐỘ (SPD_MODE): trong chế độ này, bộ điều khiển vòng kín nhận ra mục tiêu tốc độ đầu vào bằng cách loại bỏ bất kỳ nhiễu nào (tải điện trở) được áp dụng cho động cơ. Được đề xuất cho các ứng dụng rô bốt hoặc các ứng dụng tốc độ không đổi.
 - TORQUE MODE (TRQ_MODE): trong chế độ này, mục tiêu mô-men đầu vào được thực hiện. Chế độ này cho phép động cơ "quay vòng tự do" khi mục tiêu mô-men xoắn là 0. Được đề xuất cho hầu hết các ứng dụng với người lái xe ngồi.

d) Các chế độ mẫu:

- + **VARIANT_ADC**: Động cơ được điều khiển bởi hai chiết áp được kết nối với cặp cảm biến Trái (có dây dài)
- + **VARIANT_USART**: Động cơ được điều khiển thông qua giao thức nối tiếp (ví dụ: trên cặp cảm biến bên phải USART3, cáp có dây ngắn). Các lệnh có thể được gửi từ Arduino.
- + **VARIANT_NUNCHUK**: Wii Nunchuk cung cấp khả năng điều khiển ga, phanh và lái bằng một tay. Đây là một trong những thiết bị đầu vào đầu tiên được sử dụng cho ghế bành điện hoặc thùng chai.
- + **VARIANT_PPM**: Điều khiển từ xa RC với tín hiệu PPM Sum.
- + **VARIANT_PWM**: Điều khiển từ xa RC với tín hiệu PWM.
- + **VARIANT_IBUS**: Điều khiển từ xa RC với giao thức Flysky iBUS được kết nối với cặp cảm biến Trái.
- + **VARIANT_HOVERCAR**: Động cơ được điều khiển bởi hai bàn đạp phanh và ga.
- + **VARIANT_HOVERBOARD**: Mainboard đọc dữ liệu của hai sideboard. Các bảng phụ cần được flash với phiên bản hack. Bộ điều khiển cân bằng chưa được triển khai.
- + **VARIANT_TRANSPOTTER**: Đây là hệ thống vận tải dành cho thiết bị di chuyển, là một hệ thống giao thông dựa trên ván trượt
- + **VARIANT_SKATEBOARD**: Cái này dành cho việc xây dựng ván trượt, được điều khiển bằng điều khiển từ xa RC với tín hiệu PWM được kết nối với cặp cảm biến bên phải.

e) Kết luận

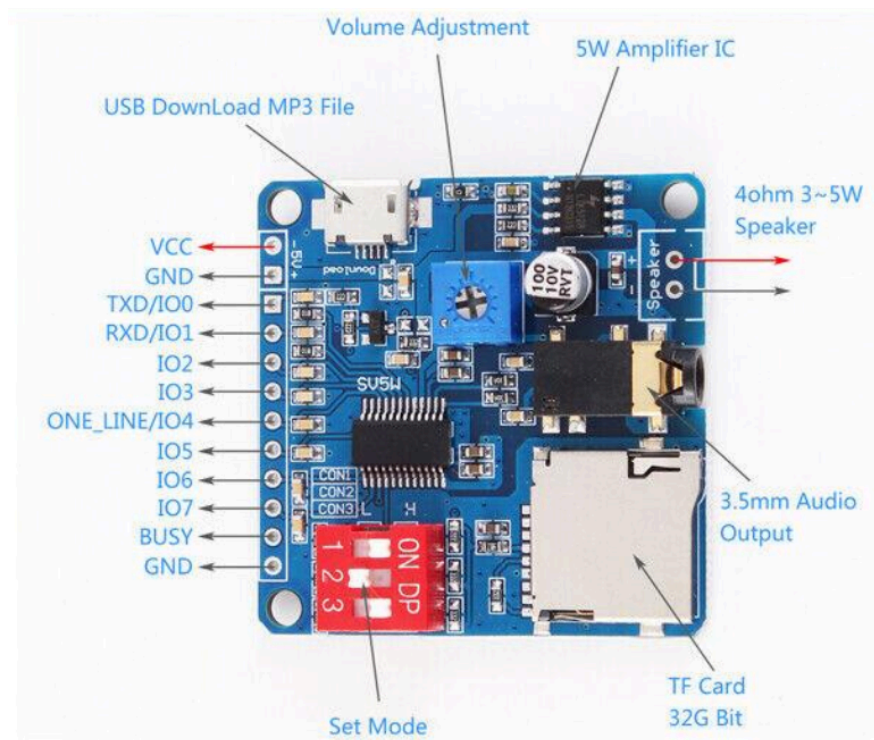
Trong đồ án này, Hoverboard Hardware được sử dụng để điều khiển tốc độ và chiều quay của động cơ.

3.2.4 Mạch phát âm thanh DY-SV5W

a) Giới thiệu chung

Mạch phát âm thanh DY-SV5W MP3 WAV Player UART tích hợp Amply 5W được sử dụng để phát âm thanh MP3 WAV từ thẻ nhớ MicroSD, mạch có thể điều khiển qua giao tiếp UART hoặc kích bằng các chân tín hiệu trên mạch, mạch hỗ trợ định dạng thẻ nhớ FAT16 / FAT32 tối đa lên đến 32GB với 2 chuẩn âm thanh thông dụng là MP3 và WAV.

Mạch phát âm thanh DY-SV5W MP3 WAV Player UART tích hợp Amply 5W với Amply 5W nên có thể phát trực tiếp ra loa có trở kháng dưới 4Ω , mạch còn tích hợp biến trở chỉnh âm lượng (volume), cổng MicroUSB (cấp nguồn và đọc thẻ nhớ) và ngõ ra âm thanh chuẩn 3.5mm, thích hợp với các ứng dụng phát âm thanh, Robot, Smarthome,...



Hình 3.8 Mạch phát âm thanh DY-SV5W

b) Thông số kỹ thuật Mạch phát âm thanh DY-SV5W:

- + Điện áp sử dụng: 5VDC từ chân cấp nguồn hoặc cổng Micro USB.
- + Hỗ trợ chuẩn âm thanh MP3 / WAV.
- + Hỗ trợ Sampling Rate (KHz): 8/11.025/12/16/22.05/24/32/44.1/48.
- + Hỗ trợ ngõ ra DAC 24-bit, dynamic range support 90dB, signal-to-noise ratio support 85dB.
- + Hỗ trợ thẻ nhớ MicroSD (TF Card) định dạng thẻ nhớ FAT16/FAT32, bộ nhớ tối đa lên đến 32GB.
- + Tích hợp Amply 5W Class D phát trực tiếp ra loa có trở kháng dưới 4Ω.
- + Hỗ trợ giao tiếp UART điều khiển hầu hết các chức năng: Play, Back, Pause, Selection, Volume, ... điều khiển lên đến 65535 bài hát, tốc độ giao tiếp Baud rate 9600bit/s.
- + Hỗ trợ điều khiển qua các IO trên mạch với các chế độ độc lập, kết hợp để điều khiển lên đến 255 bài hát.
- + Hỗ trợ điều khiển qua chỉ 1 dây giao tiếp tại IO4 với các chức năng: Play, Back, Pause, Selection, Volume, ...
- + Tích hợp 3 công tắc để chuyển 7 chế độ điều khiển khác nhau.
- + Tích hợp cổng Micro USB để cấp nguồn và có chức năng đọc thẻ nhớ MicroSD (TF Card) trên mạch.

- + Kích thước: 40 x 40mm

c) Kết luận

Trong đồ án này, sử dụng mạch Mạch phát âm thanh DY-SV5W để thông báo cho khách hàng để chọn và nhận đồ uống.

3.2.5 Loa Toàn Dải 2inch - 6Ω 5W

a) Giới thiệu



Hình 3.9 Loa toàn dải 2 inch - 6Ω 5W

Thông số kỹ thuật:

- + Trở kháng loa: 6Ω
- + Công suất: 5W
- + Tần số đáp ứng: 167Hz - 20KHz
- + Đường kính ngoài tối đa: 64mm
- + Đường kính mở: 58mm
- + Khoảng cách đường chéo: 73mm
- + Chiều cao: 30mm

b) Kết luận

Trong đồ án này, loa được nối với mạch phát âm thanh để phát những âm thanh đã lưu sẵn trong mạch để thông báo với người sử dụng.

3.2.6 Nút dừng khẩn cấp

a) Giới thiệu

Nút dừng khẩn cấp (Emergency Stop Button) là một thành phần quan trọng trong các hệ thống điều khiển và tự động hóa, được sử dụng để dừng ngay lập tức hoạt động của hệ thống trong tình huống khẩn cấp hoặc nguy hiểm. Nó đảm bảo an toàn cho nhân viên và thiết bị trong trường hợp cần dừng lại một cách nhanh chóng.



Hình 3.10 Nút dừng khẩn cấp

b) Thông số kỹ thuật

- + Điện áp: (điện áp tối đa): 600V
- + Dòng: (dòng điện tối đa): 10A
- + Đường kính ren gắn tủ: 22mm
- + Tiếp điểm 1NO + 1NC
- + Nhiệt độ hoạt động: -25~70 độ C.
- + Chất liệu: nhựa, kim loại

c) Nguyên lý hoạt động

- + Ngắt mạch điện: Nút dừng khẩn cấp được kết nối với mạch điện của hệ thống. Khi nút này được nhấn, nó ngắt mạch điện, ngắt kết nối nguồn điện hoặc tín hiệu điều khiển đến các thiết bị và máy móc trong hệ thống. Điều này dẫn đến việc dừng hoạt động của các thiết bị đó.
- + Nguyên tắc khẩn cấp: Nút dừng khẩn cấp có tính năng khẩn cấp, nghĩa là nó được thiết kế để được nhấn một cách nhanh chóng và dễ dàng trong trường hợp xảy ra sự cố hoặc tình huống nguy hiểm. Với một lực nhấn nhất định, nút dừng khẩn cấp sẽ được nhấn xuống, ngắt mạch điện và dừng hoạt động của hệ thống ngay lập tức.
- + Thiết kế có thể khóa lại: Nút dừng khẩn cấp thường có thiết kế có thể khóa lại (latching design). Sau khi được nhấn, nút này sẽ ở trạng thái khẩn cấp và chỉ có thể được trả về trạng thái bình thường bằng cách quay, kéo hoặc đảo nút trở lại vị trí ban đầu. Điều này giúp ngăn ngừa việc vô tình đặt lại hoặc mở nút dừng khẩn cấp, đảm bảo rằng hệ thống không khởi động lại mà không có ý định.
- + Tín hiệu và phản hồi: Khi nút dừng khẩn cấp được nhấn, thông thường nó sẽ tạo ra một tín hiệu điện hoặc lệnh điều khiển để thông báo rằng hệ thống đã dừng lại. Điều này có thể được thực hiện bằng cách gửi tín hiệu đến các bộ điều khiển, hiển thị trạng thái hoặc hệ thống.

d) Kết luận

Trong đồ án này, nút dừng khẩn cấp được sử dụng để ngay lập tức dừng hoạt động của hệ thống trong trường hợp khẩn cấp hoặc nguy hiểm.

3.2.7 Pin

a) Giới thiệu



Hình 3.11 Pin

Loại pin: Lithium (Li-ion). Hàng chính hãng chuyên dụng cho xe điện cân bằng 2 bánh, xe drift...

b) Thông số kỹ thuật

- + Loại pin: Lithium (Li-ion).
- + Dung lượng: 4400mAh
- + Điện thế pin: 36V
- + Dòng xả pin: 20A.
- + Công suất khởi động: 720W
- + Công suất hoạt động liên tục: 540W.
- + Chuẩn kết nối: XT60.

c) Kết luận

Trong đồ án này, Pin Lithium được sử dụng để cấp nguồn cho mạch Jetson Nano cũng như nguồn cho Hoverboard.

3.2.8 Màn hình HDMI LCD 7 inch

a) Giới thiệu



Hình 3.12 Màn Hình LCD 7 inch Cảm ứng HDMI

Màn hình 7 inch HDMI LCD là một màn hình hiển thị LCD nhỏ gọn với kết nối HDMI. Được thiết kế cho sự tiện lợi và độ phân giải cao, nó thường được sử

dụng trong các ứng dụng như máy tính nhúng, máy tính nhỏ gọn, máy chơi game di động, camera, và các dự án điện tử khác .

Màn hình 7 inch HDMI LCD là một lựa chọn phổ biến trong cộng đồng điện tử DIY và người dùng đam mê công nghệ, nhờ kích thước nhỏ gọn, độ phân giải cao, và tính linh hoạt trong kết nối và tương tác.

b) Thông số kỹ thuật

- + Model: 7 inch HDMI LCD (C).
- + Độ phân giải: 1024 x 600 pixel
- + Loại cảm ứng: điện trở (kết nối qua cổng microUSB)
- + Cổng kết nối: HDMI.
- + Tương thích: Raspberry PI, PI2, ...
- + Hỗ trợ Banana Pi / Banana Pro, comes with Lubuntu, Raspbian images
- + Giao tiếp hiển thị: HDMI
- + Điều khiển chạm: giao tiếp USB
- + Điều chỉnh độ sáng đèn nền với chế độ tiết kiệm công suất
- + Kích thước: 163.8 x 97 x 2.6 (mm)
- + Trọng lượng sản phẩm: 175 (g)

c) Nguyên lý hoạt động

- + Màn hình LCD: Màn hình sử dụng công nghệ LCD (Liquid Crystal Display), trong đó ánh sáng từ nguồn chiếu phía sau sẽ đi qua các lớp cấu thành màn hình để tạo ra hình ảnh. Cấu trúc của màn hình LCD gồm nhiều lớp, bao gồm hai tấm kính polaroid với một lớp chất lỏng điện dẫn (liquid crystal) nằm giữa chúng.
- + Đèn nền (Backlight): Đèn nền cung cấp nguồn sáng cho màn hình LCD. Trong trường hợp màn hình 7 inch HDMI LCD (C), có thể sử dụng đèn nền LED (Light Emitting Diode) hoặc CCFL (Cold Cathode Fluorescent Lamp).
- + HDMI (High-Definition Multimedia Interface): HDMI là một giao diện đa phương tiện cao cấp cho truyền tải âm thanh và hình ảnh số chất lượng cao từ một nguồn tới màn hình hoặc thiết bị khác. Đầu vào HDMI trên màn hình 7 inch HDMI LCD (C) cho phép bạn kết nối với các nguồn tín hiệu HDMI như máy tính, máy chơi game, thiết bị đa phương tiện và các nguồn tương tự.
- + Điều khiển và xử lý tín hiệu: Màn hình 7 inch HDMI LCD (C) đi kèm với một bộ điều khiển và xử lý tín hiệu để nhận và hiển thị tín hiệu đầu vào từ HDMI. Bộ điều khiển này có nhiệm vụ giải mã tín hiệu đầu vào HDMI và điều chỉnh các thông số như độ phân giải, tần số làm mới, độ sáng, tương phản và màu sắc để tạo ra hình ảnh trên màn hình.
- + Điều chỉnh và hiển thị hình ảnh: Sau khi bộ điều khiển xử lý tín hiệu, màn hình LCD sẽ hiển thị hình ảnh lên màn hình. Các tấm kính polaroid và lớp chất lỏng điện dẫn trong màn hình LCD sẽ thay đổi sự hiển thị của ánh sáng đi qua để tạo ra các màu sắc và hình ảnh.

d) Kết luận

Trong đồ án này, màn hình HDMI LCD 7 inch sẽ được sử dụng để hiển thị giao diện người dùng cho nhân viên và khách hàng.

3.2.9 Webcam Full HD 1080P

a) Giới thiệu



Hình 3.13 Webcam Full HD 1080P

b) Tính năng

- + Cảm biến hình ảnh CMOS 2MP, nhận dạng khuôn mặt nhanh.
- + Chất lượng hình ảnh sắc nét với độ phân giải full HD 1080p ở tốc độ 30 khung hình/giây.
- + Tích hợp micro lọc tạp âm, giảm tiếng ồn hiệu quả.
- + Góc quay rộng, nắm bắt được nhiều không gian hơn.
- + Cắm và sử dụng không cần cài đặt trình điều khiển.
- + Tương thích với các phần mềm Zoom, Skype, Facebook, Youtube, Video Call.
- + Hỗ trợ giao diện USB 2.0.

c) Thông số kỹ thuật

- + Tốc độ khung hình : 30 khung hình/giây
- + Độ phân giải video : 720P - 1080P
- + Độ phân giải hình chụp : 10MP
- + Tốc độ khung hình quay : 60FPS
- + Microphone : Có
- + Cổng kết nối: USB 2.0

d) Kết luận

Trong đồ án này, webcam có tác dụng ghi hình ảnh thời gian thực cho các bài toán nhận diện đồ uống hay điều khiển robot qua cử chỉ.

Hình 3.14 Bản vẽ mạch điện

CHƯƠNG 4. LẬP TRÌNH ĐIỀU KHIỂN HỆ THỐNG

4.1 Giới thiệu

Trong chương này sẽ giới thiệu tổng quan về hệ điều hành ROS và các bài toán đối với robot tự hành bao gồm: định vị và xây dựng bản đồ, tìm đường đi và nhận dạng vật thể đồng thời cũng giới thiệu về ứng dụng trí tuệ nhân tạo nhận diện đồ uống, cử chỉ tay để điều khiển robot. Từ đó, ta sẽ đưa ra các kết quả, vấn đề gặp phải, hướng giải quyết trong thực nghiệm.

4.2 Tổng quan về ROS

4.2.1 ROS là gì?

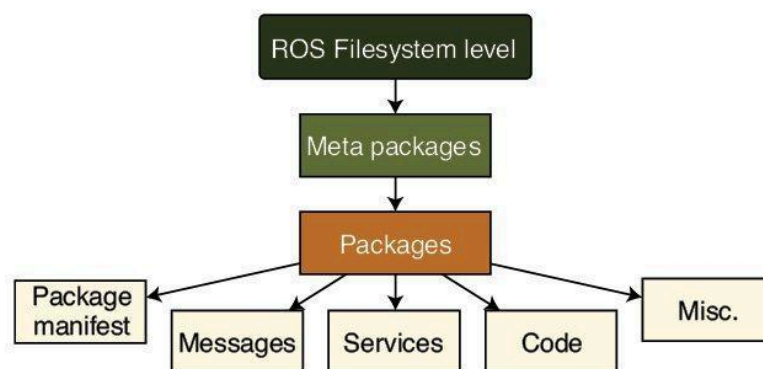
Robot operating system (ROS) là một hệ thống phần mềm có tính linh hoạt và chuyên dụng cao dùng để lập trình và điều khiển robot. ROS bao gồm các thư viện, công cụ hỗ trợ lập trình, công cụ đồ họa, các công cụ hỗ trợ giao tiếp điều khiển trực tiếp với phần cứng cũng như các thư viện hỗ trợ việc lấy dữ liệu từ các cảm biến và các thuật toán phổ biến trong lập trình điều khiển robot. ROS là gói công cụ được cài đặt trên Ubuntu.

ROS giúp người dùng có thể dễ dàng kết nối các máy tính ở trên robot (Raspberry pi, Jetson Nano, các loại máy tính nhúng, mini computer,..) với laptop để điều khiển hoặc thao tác từ xa.

ROS giúp đơn giản và tiết kiệm thời gian việc lập trình các hành vi phức tạp của robot. ROS như là một mạng lưới điểm (node) được kết nối bởi nhiều node mà mỗi node có nhiệm vụ riêng tương ứng với các thành phần cấu tạo nên robot. Mỗi node riêng có thể được tùy chỉnh phát triển và lập trình theo ý tưởng của người phát triển mà không cần quan tâm đến việc phải thống nhất sử dụng cùng một ngôn ngữ lập trình, điều đó có nghĩa là node này bạn phát triển trên ngôn ngữ C++ nhưng node khác bạn có thể phát triển trên ngôn ngữ Python. Python và C++ là hai ngôn ngữ nền tảng mà bạn cần phải học trước khi làm quen với ROS.

4.2.2 FILE-SYSTEM LEVEL của ROS

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



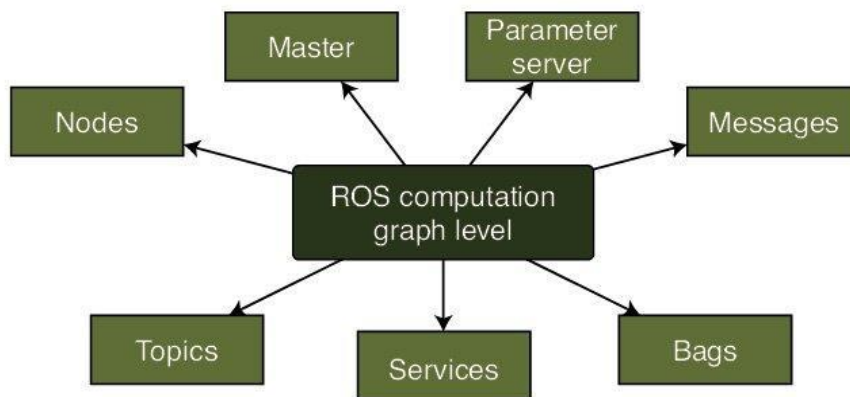
Hình 4.1 FILESYSTEM LEVEL của ROS

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

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

4.2.3 Các thành phần của ROS

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



Hình 4.2 Các thành phần chính của ROS

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

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

Parameter Server: Là một cấu trúc nhiều tham số có thể truy cập trong lúc chạy ROS. Các node sử dụng cấu trúc này nhằm lưu trữ và truy xuất các thông số trong thời gian chạy. Do nó không có hiệu suất cao nên thường dùng với kiểu dữ liệu tĩnh, chẳng hạn như các thông số cấu hình, thời gian hệ thống.

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

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

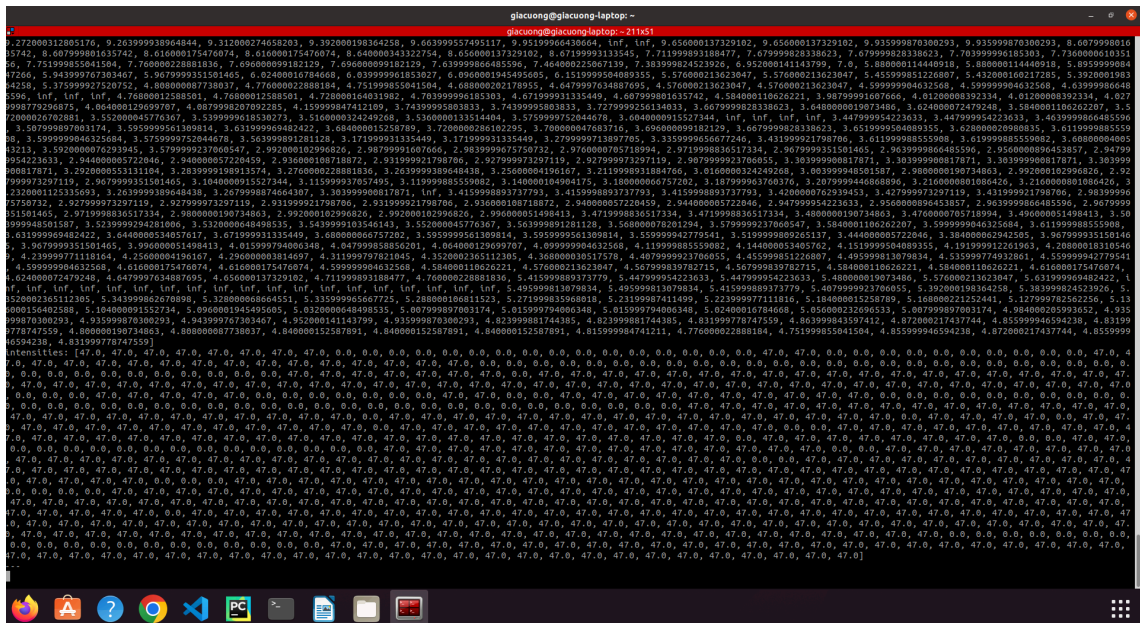
- + Publish là đăng kí gửi thông tin
- + Subscribe là đăng ký nhận thông tin

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

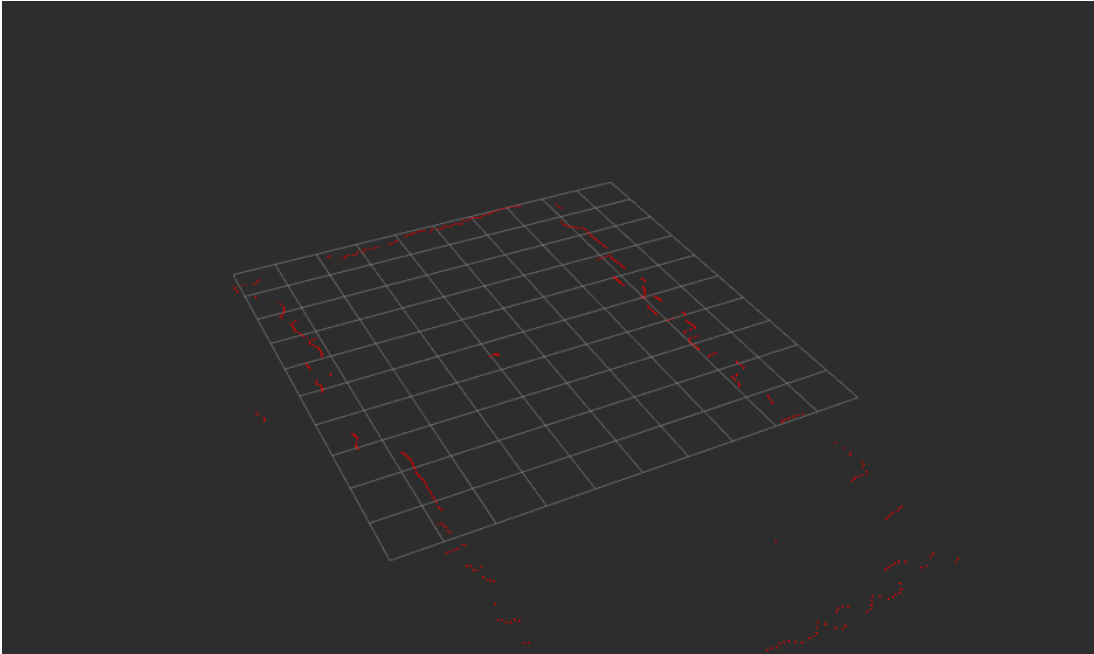
Bags: là một định dạng tệp trong ROS dùng để lưu trữ dữ liệu message với phần mở rộng là .bag có vai trò quan trọng và có thể được xử lý, phân tích bởi các công cụ mô phỏng trong ROS như Rviz.

4.2.4 Dữ liệu từ RPLidar A1

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



Hình 4.3 Một phần dữ liệu nhận được từ Lidar



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

4.2.5 Hector SLAM

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

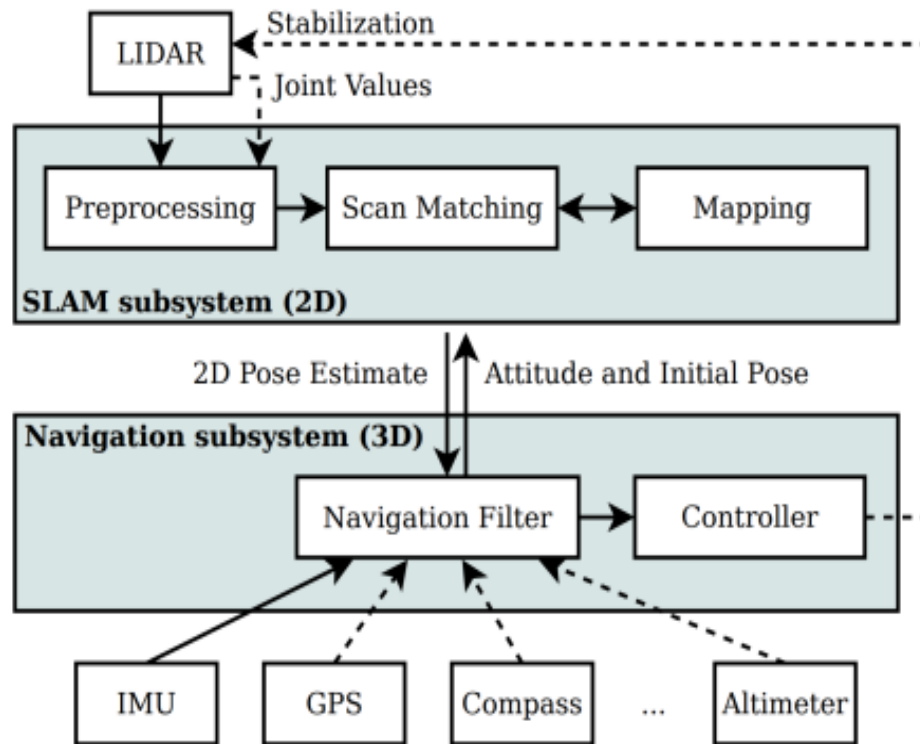
Thuật toán cố gắng tìm sự liên kết tối ưu của các điểm cuối của quá trình quét laser với bản đồ đã được xây dựng bằng cách tìm phép biến đổi phân cứng ξ^* :

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^n [1 - M(S_i(\xi))]^2$$

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

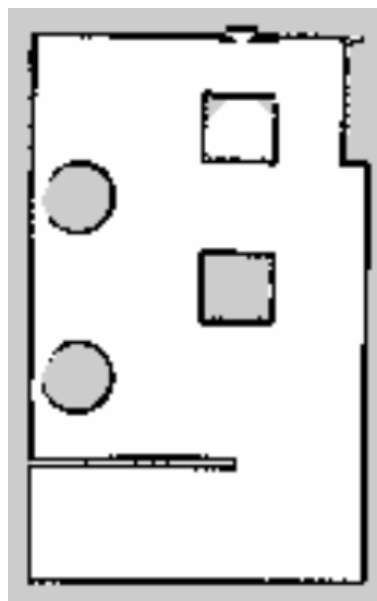
$$\sum_{i=1}^n [1 - M(S_i(\xi + \Delta\xi))]^2 \rightarrow 0$$

Hình 4.5 cung cấp tổng quan về ngăn xếp thuật toán Hector. Dữ liệu được xử lý trước từ LiDAR được chuyển tiếp đến một thuật toán đối sánh quẻ



4.2.6 Thiết lập bản đồ

Bản đồ thiết lập từ thuật toán SLAM Hector sẽ có 2 phần gồm một file `<map_name>.pgm` là một file chứa hình ảnh bản đồ và một file `<map_name>.yaml` chứa các thông số về bản đồ dùng cho mô phỏng.



Hình 4.6 Hình ảnh bản đồ xây dựng từ thuật toán Hector

File `<map_server>.yaml` chứa thông tin định dạng bản đồ sẽ có dạng như sau:

- + image: /home/hoang247/restaurant.pgm
- + resolution: 0.1
- + origin: [0.0, 0.0, 0.0]
- + occupied_thresh: 0.65
- + free_thresh: 0.196
- + negate: 0

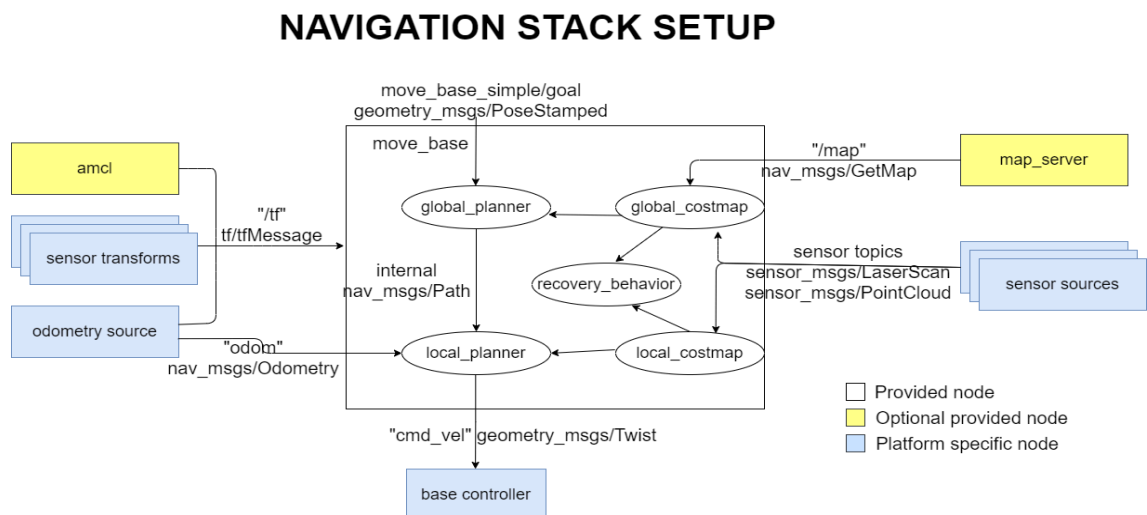
Trong đó:

- + **image**: Đường dẫn đến file chứa ảnh bản đồ.
- + **resolution**: Độ phân giải của bản đồ, meters / pixel.
- + **origin**: Tọa độ 2D của pixel phía dưới bên trái của bản đồ (x, y, yaw).
- + **occupied_thresh**: Các điểm ảnh với tỉ lệ chiếm chỗ lớn hơn ngưỡng này được coi là có vật cản.
- + **free_thresh**: Các điểm ảnh với tỉ lệ chiếm chỗ lớn hơn ngưỡng này được coi là trống.
- + **negate**: Thông số xác định có đảo ngược màu trắng với đen không (thông thường negate là 0)

4.3 Bài toán hoạch định đường đi

4.3.1 Tổng quan

Navigation: Tìm ra 1 đường đi (nếu tồn tại con đường đó) để robot đi đến đích cho trước một con robot đã được định vị trong một môi trường biết trước.



Hình 4.7 Sơ đồ khối Navigation Stack

amcl: là một hệ thống bản đồ hóa xác suất cho một robot di chuyển ở chế độ 2D, sử dụng bộ lọc hạt (Particle filter) dựa trên dữ liệu từ các cảm biến laser như lidar để đưa ra vị trí ước lượng của robot.

sensor transforms: Sau khi nhận dữ liệu từ các cảm biến như siêu âm, encoder, ... thông qua phép biến đổi các hệ tọa độ TF giúp định vị chính xác hơn vị trí của robot trong môi trường. Trong đề tài này sử dụng duy nhất 1 cảm biến để định vị là Lidar.

odometry source: thông tin về tốc độ của robot (đọc từ encoder) hoặc IMU để ước tính vị trí của robot so với vị trí trước đó.

move_base_simple/goal: Xác định điểm (vị trí) cuối robot cần thực hiện di chuyển tới.

map_server: chứa dữ liệu về bản đồ đã được lập sẵn trong phần mapping cho robot.

sensor source: lấy dữ liệu các cảm biến từ các node.

global costmap: hệ thống bản đồ xây dựng từ lidar và được thêm 1 số thành phần tỷ lệ để điều chỉnh hệ thống map.

local costmap: hệ thống bản đồ hiện tại khi robot đang ở trong môi trường và so sánh với hệ thống global costmap

global planner: kế hoạch di chuyển xây dựng từ global costmap

local planner: kế hoạch di chuyển xây dựng từ local costmap.

4.3.2 Thuật toán Dynamic Window Approach (DWA)

Thuật toán **Dynamic Window Approach (DWA)** là dùng để tìm ra một tín hiệu điều khiển hợp lý gửi xuống robot nhằm mục đích điều khiển nó đến đích an toàn, nhanh chóng dựa trên global planner đã hoạch định từ trước. Thuật toán này gồm hai bước chính là cắt giảm không gian tìm kiếm (search space) của vận tốc và tìm được vận tốc tối ưu trong không gian tìm kiếm đó.

Ý tưởng cơ bản của thuật toán Dynamic Window Approach (DWA) như sau:

- + Lấy mẫu rời rạc vận tốc trong không gian điều khiển của robot $V(v, w)$
- + Đối với mỗi vận tốc V được lấy mẫu, hãy thực hiện mô phỏng chuyển tiếp từ trạng thái hiện tại của robot để dự đoán điều gì sẽ xảy ra nếu vận tốc được lấy mẫu được áp dụng trong một khoảng thời gian ngắn Δt nào đó.
- + Đánh giá và tính trọng số từng quỹ đạo do mô phỏng chuyển tiếp, sử dụng số liệu kết hợp các đặc điểm như: gần chướng ngại vật, gần mục tiêu, gần đường do global planner đưa ra và tốc độ. Loại bỏ các quỹ đạo lỗi (những quỹ đạo va chạm với chướng ngại vật).
- + Chọn quỹ đạo với trọng số cao nhất và gửi vận tốc này đến robot.
- + Xóa các trọng số và lặp lại.

Không gian tìm kiếm:

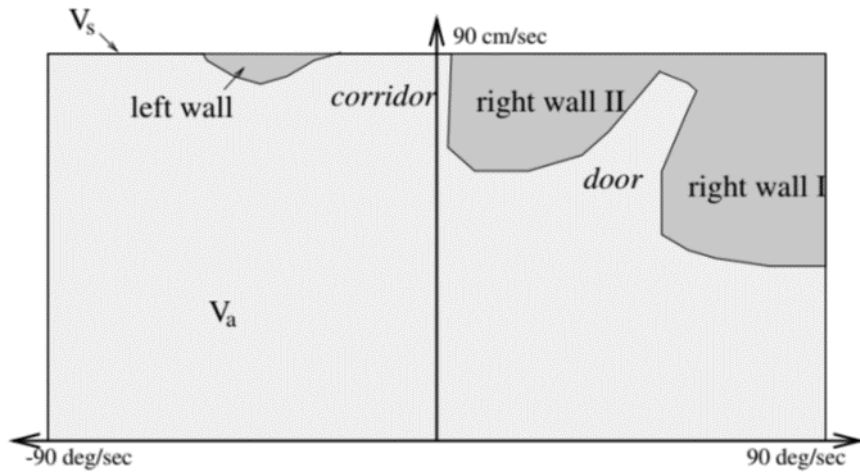
Các vận tốc có thể điều khiển được trong không gian tìm kiếm được cắt giảm theo ba bước sau:

- + **Quỹ đạo tròn**: thuật toán DWA chỉ xét đến quỹ đạo là hình tròn (đường cong) được xác định duy nhất bởi một cặp vận tốc dài và vận tốc góc (v , ω).
- + **Vận tốc cho phép**: nhằm tạo ra một quỹ đạo an toàn cho robot để tránh vật cản. Một cặp vận tốc (v , ω) được cho phép là khi robot có thể dừng trước vật cản gần nhất mà không có sự va chạm trên đường cong tương ứng với vận tốc đó. Vận tốc cho phép được định nghĩa như sau:

$$Va = \{(v, \omega) | v \leq \sqrt{2 \cdot dist(v, \omega) \cdot vb} \wedge \omega \leq \sqrt{2 \cdot dist(v, \omega) \cdot \omega b}\}$$

Trong đó:

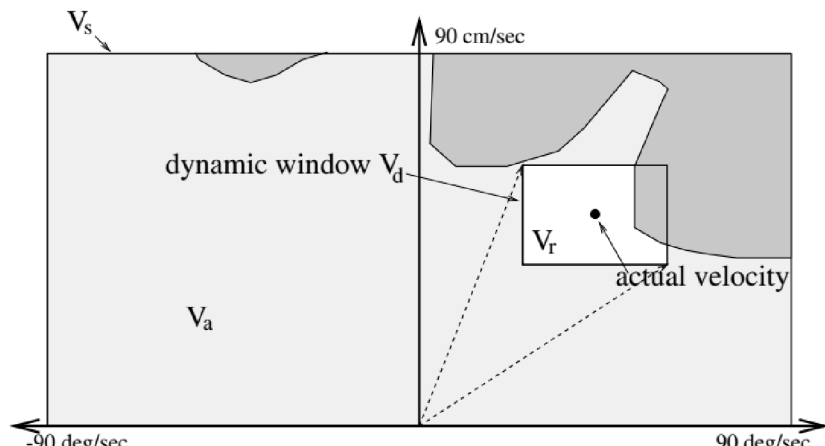
- + **Va**: là chuỗi các giá trị vận tốc (v, ω) cho phép robot dừng trước vật cản mà không có sự va chạm.
- + **dist(v, ω)**: là khoảng cách nhỏ nhất mà robot dừng trước vật cản để không có sự va chạm.
- + **vb, ωb**: là gia tốc của vận tốc thẳng và vận tốc xoay tối đa nếu robot di chuyển sẽ gây va chạm với vật cản.



Hình 4.8 Vận tốc cho phép Va trong DWA

Dynamic window: nhằm hạn chế vận tốc cho phép đối với những vận tốc có thể đạt được trong khoảng chu kỳ cho trước với gia tốc tối đa của robot. Để Δt là khoảng thời gian mà trong đó gia tốc $\dot{v}$, $\dot{\omega}$ sẽ được thực thi để (va, ω_a) là vận tốc thực được gửi xuống robot. Từ đó, vận tốc Vd sẽ được định nghĩa như sau

$$Vd = \{(v, \omega) | v \in [va - \dot{v} \cdot \Delta t, va + \dot{v} \cdot \Delta t] \wedge \omega \in [\omega_a - \dot{\omega} \cdot \Delta t, \omega_a + \dot{\omega} \cdot \Delta t]\}$$



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

$$Vr = Vs \cap Va \cap Vd$$

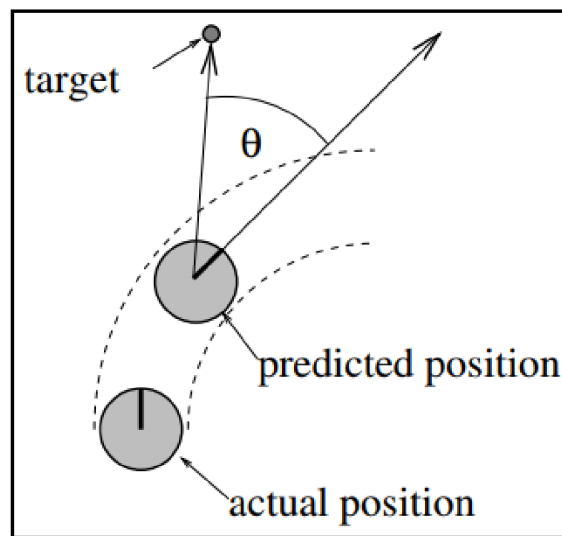
Tối ưu

Ta có hàm mục tiêu được định nghĩa như sau :

$$G(v, \omega) = \alpha \cdot heading(v, \omega) + \beta \cdot dist(v, \omega) + \gamma \cdot vel(v, \omega)$$

Để có thể tối ưu hóa vận tốc ngõ ra thì hàm mục tiêu phải có giá trị tối đa. Để thực hiện được điều này, ta thực hiện các bước sau:

Target heading: heading là giá trị đo tiến độ hướng đến đích của robot. Giá trị sẽ mang giá trị tối đa khi robot di chuyển trực tiếp về phía đích. Giá trị của $heading(v, \omega)$ được tính bởi công thức $180 - \theta$, với θ là góc giữa hướng của robot và điểm đích.



Không gian trống (clearance): hàm $dist(v, \omega)$ thể hiện khoảng cách tính từ robot đến vật cản gần nhất nằm trên quỹ đạo cong của nó. Giá trị này sẽ rất lớn nếu không có vật cản nằm trên quỹ đạo cong di chuyển của nó. Giá trị này càng nhỏ thì việc nó đối mặt với vật cản càng cao, khi đó nó sẽ di chuyển xung quanh vật cản ấy.

Vận tốc: hàm $vel(v, \omega)$ là vận tốc di chuyển thẳng của robot và hỗ trợ di chuyển nhanh hơn. Các hệ số α, β, γ được chọn sao cho phù hợp với đặc tính của robot và môi trường hoạt động. Khi hàm mục tiêu có giá trị lớn nhất thì quỹ đạo tối ưu sẽ được chọn với vận tốc (v, ω) tốt nhất và là kết quả của thuật toán.

4.3.3 Tổng quan về thuật toán tìm đường đi ngắn nhất

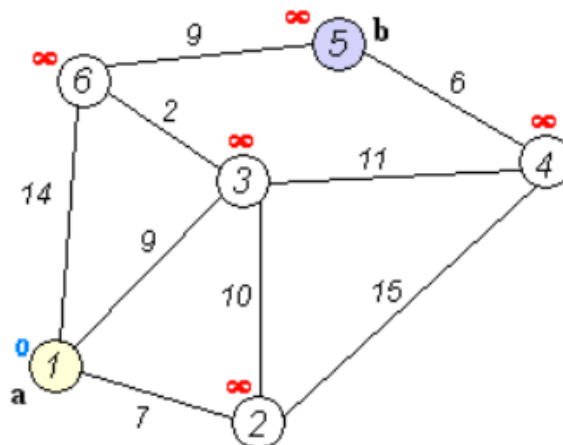
Trong điều hướng cho robot để robot đến được vị trí mong muốn, ta phải đưa ra một hoạch định đường đi từ điểm bắt đầu (vị trí hiện tại của robot) đến điểm kết thúc (vị trí đích mong muốn). Nhằm tối ưu hóa cho đường đi, ta cần phải lựa chọn đường đi ngắn nhất giữa hai điểm ấy. Với yêu cầu này, thuật toán Dijkstra là một trong những thuật toán đơn giản để tìm được đường đi tối ưu nhất. Bài toán đặt ra cho thuật toán Dijkstra là có một đồ thị $G = (V, E)$ (với V là tập hợp các đỉnh và E là các cạnh vô hướng hoặc có hướng), mỗi cạnh sẽ có trọng số luôn

đương và một đỉnh nguồn s . Chúng ta cần tính toán được đường đi ngắn nhất từ đỉnh nguồn s đến mỗi đỉnh trên đồ thị.

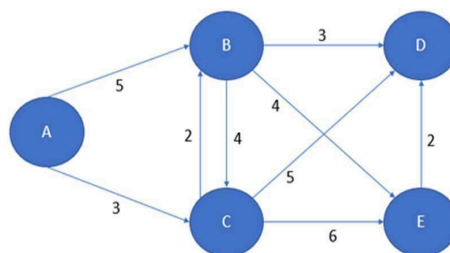
Ta có một đồ thị gồm 6 nút, 9 cạnh, các trọng số trên mỗi cạnh và có điểm nguồn a , điểm đích b . Vậy chúng ta cần tìm đường đi ngắn nhất từ a đến b . Để thực hiện được thuật toán này, ta sẽ có phương pháp giải như sau:

- + Trước khi bắt đầu, ta cần gán mỗi nút một giá trị khởi tạo: bằng 0 đối với nút bắt đầu và bằng vô cùng cho tất cả các nút còn lại.
- + Đưa tất cả các nút vào trong một tập dữ liệu Q để ta tính khoảng cách từ điểm khởi tạo đến điểm đó.
- + Đối với điểm nút hiện tại, lần lượt xét các điểm nút xung quanh chưa từng xét trước đó và tính khoảng cách dự kiến của nó từ trọng số của nút hiện tại. Nếu giá trị dự kiến nhỏ hơn giá trị hiện tại của nút đó thì gán lại giá trị của nút đó bằng giá trị dự kiến.
- + Đối với mỗi nút đã xét, ta loại bỏ nút này khỏi tập Q để không cần phải xét lại.
- + Chúng ta lần lượt kiểm tra các nút trong tập Q . Nếu nút đích đã được kiểm tra thì dừng lại và kết thúc thuật toán.

Từ phương pháp giải trên, chúng ta sẽ thành lập đoạn mã giả (pseudo code) sau để có thể tìm được đường đi ngắn nhất.

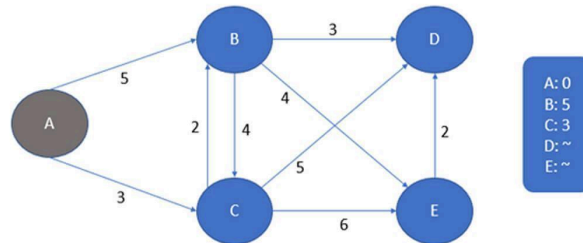


Qua thuật toán trên, ta có thể dễ dàng tìm được đường đi ngắn nhất từ điểm bắt đầu đến điểm đích. Chúng ta sẽ lấy thêm 1 ví dụ về thuật toán này. Giả sử xuất phát từ nút A .

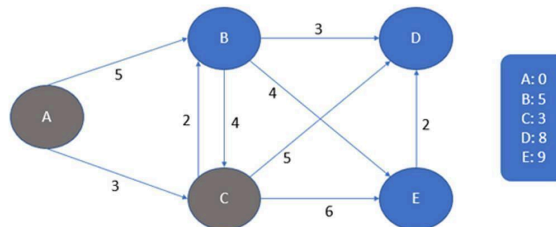


Xuất phát từ A và có 2 con đường:

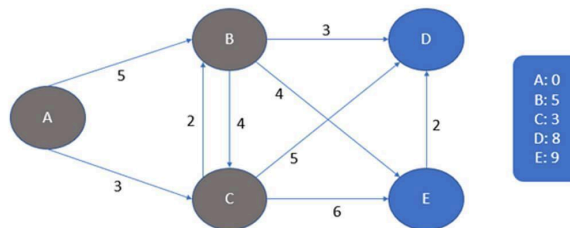
- + Từ A tới B với độ dài là 5
- + Từ A tới C với độ dài là 3



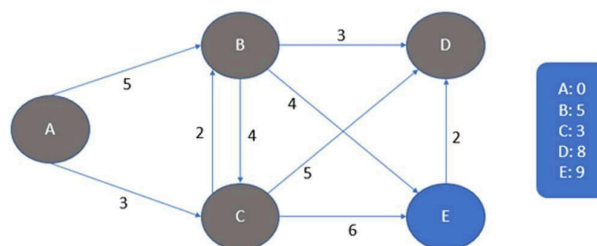
Do quãng đường từ A tới C ngắn hơn từ A tới B nên ta chọn con đường từ A tới C



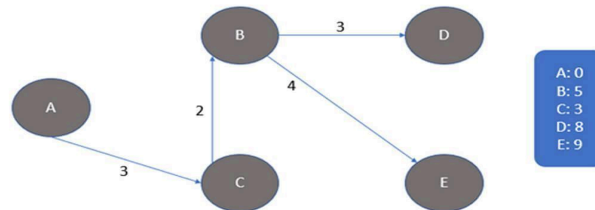
Khi ở C, ta có 3 đường đi khác nhau: Từ C $\rightarrow$ B, C $\rightarrow$ D, C $\rightarrow$ E, ghi vào danh sách 2 đỉnh mới và chọn con đường ngắn nhất là từ C đến B



Tại B lại có 3 con đường, từ B $\rightarrow$ D, B $\rightarrow$ E, B $\rightarrow$ C, chọn từ B đến D



Khi tới đây không có đường đi mới nào từ D đến E, quay lại đỉnh trước đó để kiểm tra đường đi ngắn nhất. Bây giờ có 1 đường với độ dài 4 đến E và 1 đường đi đến C, ở đây ta chọn bất kì đường nào chúng ta thích. Cuối cùng ta thấy bất kì phương án nào đi trên đường từ A đến E đều có trọng số như nhau vì các đường đi ngắn nhất được ghi trong danh sách



Từ đó ta có kết quả các đường đi ngắn nhất:

Từ A đến B: A->B, từ A đến C: A->C, từ A đến D: A->C->D, từ A đến E: A->C->B->E

4.3.4 Kết quả thực nghiệm

Cho Robot chạy tự động từ điểm A đến điểm B xong đến điểm C, với sai số đặt trước là:

- + Sai số khoảng cách: 0.2 m
- + Sai số góc (hướng): 0.2 rad

Bảng 4.1 Kết quả sai số trên thực nghiệm

Sai số Lần thứ	Điểm A		Điểm B		Điểm C	
	Khoảng cách h	Góc	Khoảng cách h	Góc	Khoảng cách h	Góc
1	0.1947	0.0541	0.1841	0.0239	0.1534	0.011
2	0.1874	0.0342	0.1674	0.0005	0.175	0.0001
3	0.1636	0.1475	0.1225	0.0207	0.124	0.0389
4	0.1705	0.0171	0.116	0.1794	0.1234	0.1734
5	0.1584	0.0758	0.1186	0.0091	0.086	0.0088
6	0.1317	0.1035	0.1533	0.0127	0.143	0.0145
7	0.153	0.0065	0.1301	0.0485	0.112	0.0444
8	0.156	0.1408	0.1401	0.0196	0.145	0.0134
9	0.178	0.0499	0.0984	0.0622	0.0911	0.0655
10	0.1856	0.008	0.1512	0.0006	0.1345	0.0011

Nhận xét: Sai số thực tế chạy ngẫu nhiên trong khoảng 0 đến 0.2 nằm trong khoảng sai số cho phép khi đặt trước, sai số do nhiễu từ Lidar, sai số từ sensor vị trí động cơ.

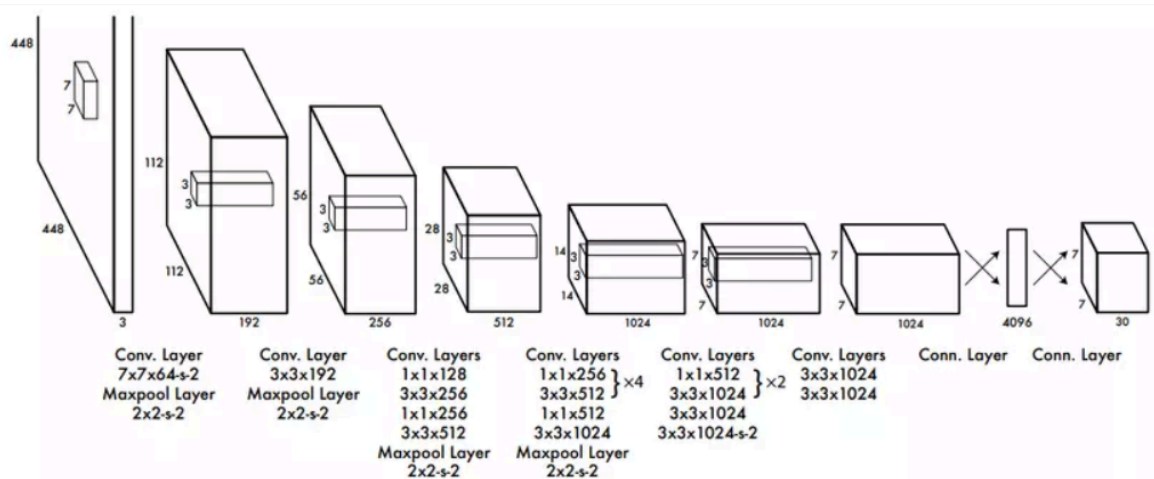
4.4 Bài toán nhận dạng đồ uống sử dụng YOLO

4.4.1 Tổng quan về mạng YOLO

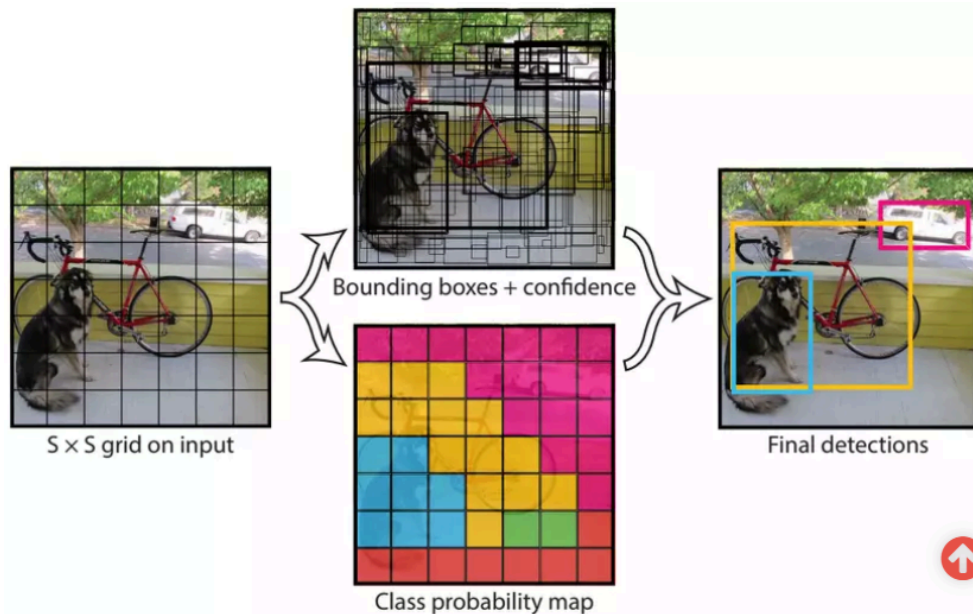
a) Khái niệm

YOLO (You Only Look Once) là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong các convolutional layers sẽ trích xuất ra các feature của ảnh còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

YOLO có ứng dụng đa dạng như: Đếm số lượng vật thể, chấm công tự động, phát hiện vật thể nguy hiểm,...



b) Nguyên lý hoạt động của YOLO



Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành thành $S \times S$ times $S \times S$ ô thường thì sẽ là 3×3 times 3×3 , 7×7 times 7×7 , 9×9 times 9×9 ... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng, mình xin trình bày ở phần sau.

- Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước $S \times S \times (5 \times N + M)$ times $S \times S \times (5 \times N + M)$ với số lượng tham số mỗi ô là $(5 \times N + M)$ times $(5 \times N + M)$ times $(5 \times N + M)$ với N và M lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán.

Ví dụ với hình ảnh trên chia thành 7×7 times 7×7 ô, mỗi ô cần dự đoán 2 bounding box và 3 object : con chó, ô tô, xe đạp thì output là $7 \times 7 \times 13$ times $7 \times 7 \times 13$, mỗi ô sẽ có 13 tham số, kết quả trả về $(7 \times 7 \times 2 = 98)$ times $(7 \times 7 \times 2 = 98)$ times $(7 \times 7 \times 2 = 98)$ bounding box. Chúng ta sẽ cùng giải thích con số $(5 \times N + M)$ times $(5 \times N + M)$ times $(5 \times N + M)$:

- Dự đoán mỗi bounding box gồm 5 thành phần : $(x, y, w, h, \text{prediction})$ với (x, y) là tọa độ tâm của bounding box, (w, h) lần lượt là chiều rộng và chiều cao của bounding box, prediction được định nghĩa.

$\text{Pr}(\text{Object}) * \text{IOU}(\text{pred}, \text{truth})$ times $\text{Pr}(\text{Object}) * \text{IOU}(\text{pred}, \text{truth})$ times $\text{Pr}(\text{Object}) * \text{IOU}(\text{pred}, \text{truth})$ được trình bày sau. Với hình ảnh trên như ta tính mỗi ô sẽ có 13 tham số, ta có thể hiểu đơn giản như sau tham số thứ 1 sẽ chỉ ra ô đó có chứa đối tượng nào hay không $P(\text{Object})$, tham số (2, 3, 4, 5) sẽ trả về (x, y, w, h) của Box1. Tham số (6, 7, 8, 9, 10) tương tự sẽ Box2, tham số (11, 12, 13) lần lượt là xác suất ô đó có chứa object1 ($P(\text{chó}|\text{object})$), object2 ($P(\text{ô tô}|\text{object})$), object3 ($P(\text{xe đạp}|\text{object})$). Lưu ý rằng tâm của bounding box nằm ở ô nào thì ô đó sẽ chứa đối tượng, cho dù đối tượng có thể ở các ô khác thì cũng sẽ trả về là 0. Vì vậy việc mà 1 ô chứa 2 hay nhiều tâm của bounding box hay đối tượng thì sẽ không thể detect được, đó là một hạn chế của mô hình YOLO1, vậy ta cần phải tăng số lượng ô chia trong 1

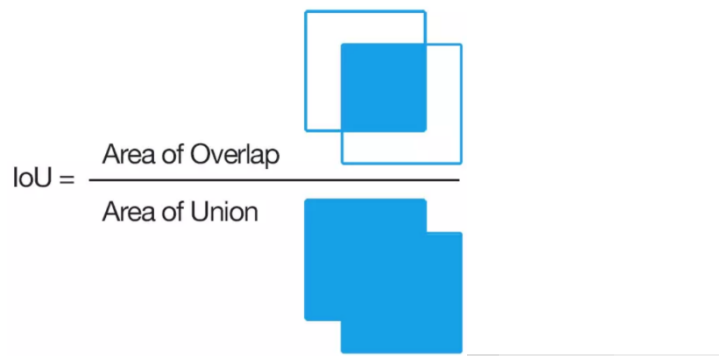
ảnh lên đó là lí do vì sao mình nói việc chia ô có thể làm ảnh hưởng tới việc mô hình phát hiện đối tượng.

c) Hàm tính IOU

Trên ta có đề cập prediction được định nghĩa

$\text{Pr(Object)} * \text{IOU(pred, truth)} \text{Pr(Object)}$ * $\text{IOU(pred, truth)} \text{Pr(Object)}$ * IOU(pred, truth) , ta sẽ làm rõ hơn IOU(pred, truth) là gì. IOU (INTERSECTION OVER UNION) là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ thể. IOU được tính bằng:

Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với growth-truth bounding box, còn Area of Union là diện tích phần hợp giữa predicted bounding box với growth-truth bounding box. Những bounding box được đánh nhãn bằng tay trong tập training set và test set. Nếu $\text{IOU} > 0.5$ thì prediction được đánh giá là tốt.



d) Loss Function

Hàm lỗi trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính. Cụ thể hơn nó là tổng độ lỗi của 3 thành phần con sau :

- + Độ lỗi của việc dự đoán loại nhãn của object – Classification loss
- + Độ lỗi của dự đoán tọa độ tâm, chiều dài, rộng của boundary box (x, y, w, h) – Localization loss
- + Độ lỗi của việc dự đoán bounding box đó chứa object so với nhãn thực tế tại ô vuông đó – Confidence loss

e) Classification loss

Classification loss – độ lỗi của việc dự đoán loại nhãn của object, hàm lỗi này chỉ tính trên những ô vuông có xuất hiện object, còn những ô vuông khác ta không quan tâm. Classification loss được tính bằng công thức sau:

$$L_{\text{classification}} = \sum_{i=0}^{S^2} l_i^{\text{obj}} \sum_{c \in \text{class}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Trong đó:

- + l_i^{obj} : Bằng 1 nếu ô vuông đang xét có object, ngược lại bằng 0

- + $\hat{p}_i(c)$: là xác suất có điều của lớp c tại ô vuông tương ứng mà mô hình dự đoán

f) Localization loss

Localization loss là hàm lỗi dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của so với vị trí thực tế từ dữ liệu huấn luyện của mô hình. Lưu ý rằng chúng ta không nên tính giá trị hàm lỗi này trực tiếp từ kích thước ảnh thực tế mà cần phải chuẩn hóa về $[0, 1]$ so với tâm của bounding box. Việc chuẩn hóa này kích thước này giúp cho mô hình dự đoán nhanh hơn và chính xác hơn so với đề giá trị mặc định của ảnh.

Ví dụ:



Hình 4.15 Ví dụ về Localization loss

Giá trị hàm Localization loss được tính trên tổng giá trị lỗi dự đoán tọa độ tâm (x, y) và (w, h) của predicted bounding box với ground-truth bounding box. Tại mỗi ô có chứa object, ta chọn 1 boundary box có IOU (Intersect over union) tốt nhất, rồi sau đó tính độ lỗi theo các boundary box này.

Giá trị hàm lỗi dự đoán tọa độ tâm (x, y) của predicted bounding box và $(\hat{x}, \hat{y})$ là tọa độ tâm của truth bounding box được tính như sau :

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B l_{ij}^{obj} \left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2$$

Giá trị hàm lỗi dự đoán (w, h) của predicted bounding box so với truth bounding box được tính như sau:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B l_{ij}^{obj} \left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2$$

Với ví dụ trên thì $S=7$, $B=2$, còn λ_{coord} là trọng số thành phần trong paper gốc tác giả lấy giá trị là 5.

g) Confidence loss

Confidence loss là độ lỗi giữa dự đoán boundary box đó chứa object so với nhãn thực tế tại ô vuông đó. Độ lỗi này tính trên cả những ô vuông chứa object và không chứa object.

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^B l_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobject} \sum_{i=0}^{S^2} \sum_{j=0}^B l_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2$$

Với ví dụ trên thì $S = 7$, $B = 2$, còn $\lambda_{noobject}$ là trọng số thành phần trong paper gốc tác giả lấy giá trị là 0.5. Đối với các hộp j của ô thứ i nếu xuất hiện object thì $C_i = 1$ và ngược lại.

h) Total loss

Tổng lại chúng ta có hàm lỗi là tổng của 3 hàm lỗi trên:

$$L_{total} = L_{classification} + L_{localization} + L_{confidence}$$

i) Dự báo bounding box

Để dự báo bounding box cho một vật thể chúng ta dựa trên một phép biến đổi từ anchor box và cell.

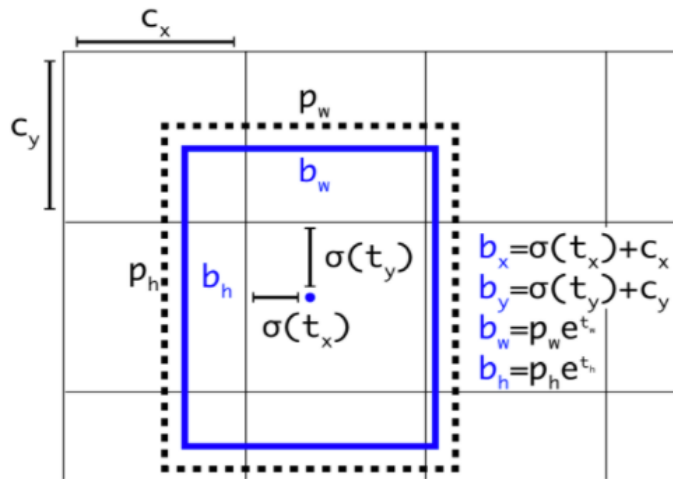
YOLO dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước (p_w, p_h) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (c_x, c_y) , mô hình dự đoán 4 tham số (t_x, t_y, t_w, t_h) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (b_x, b_y) và kích thước (b_w, b_h)

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned}$$

thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

Ngoài ra do các tọa độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng $[0, 1]$. Do đó khi áp dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.



Hình 4.16 Công thức ước lượng bounding box từ anchor box

Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (p_w, p_h) . Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLO version 1.

j) Non-max suppression

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap là rất cao. Trong trường hợp đó YOLO sẽ cần đến non-max suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể.



Hình 4.17 Non-max suppression

Các bước của non-max suppression:

- + Bước 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.

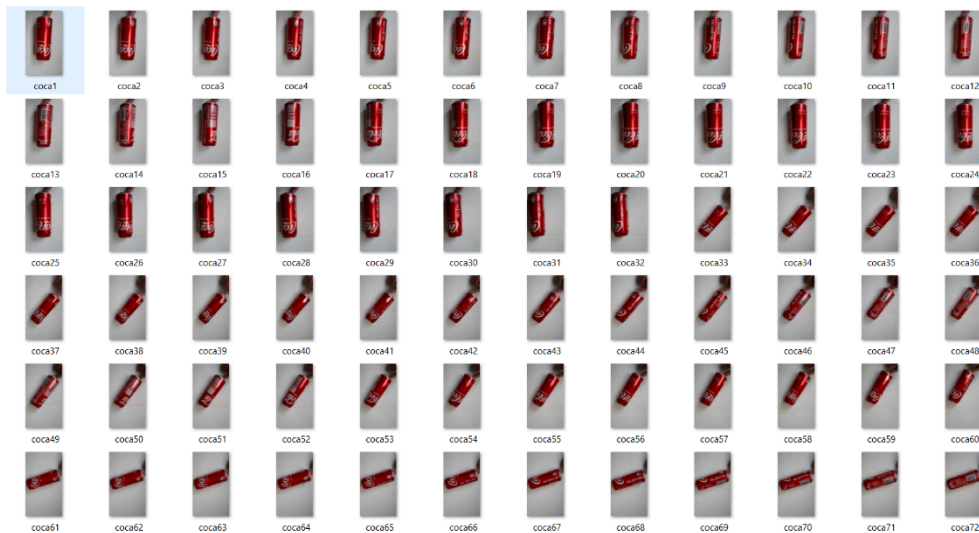
- + Bước 2: Đối với các bounding box giao nhau, non-max suppression sẽ lựa chọn ra một bounding box có xác suất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại.

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding có xác suất thấp hơn và giữ lại bounding box có xác suất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

4.4.2 Kết quả

Để nhận diện được các loại đồ uống trong mạng YOLO ta thực hiện các bước train như sau:

- + Thu thập dữ liệu:



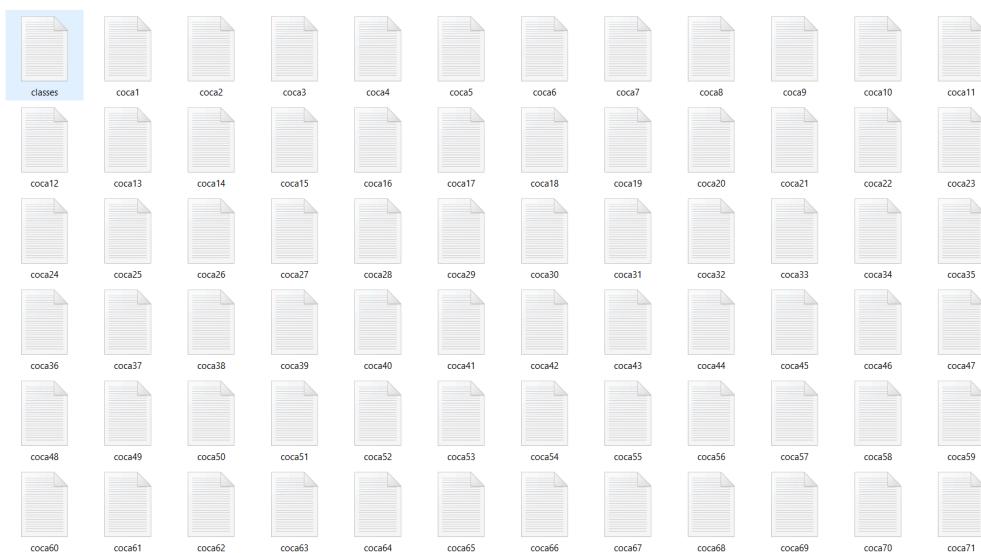
Hình 4.18 Tập dữ liệu về hình ảnh để thực hiện quá trình train

Để thực hiện quá trình gán nhãn và huấn luyện ta cần chuẩn bị tập dữ liệu khoảng 200 ảnh về 4 loại vật (Coca, Fanta, Mirinda, Spite), với mỗi hình sẽ có 1 file gán nhãn đi kèm. Dưới đây là một số hình ảnh về tập dữ liệu.



Hình 4.19 Quá trình gán nhãn cho tập dữ liệu trên labelingImg

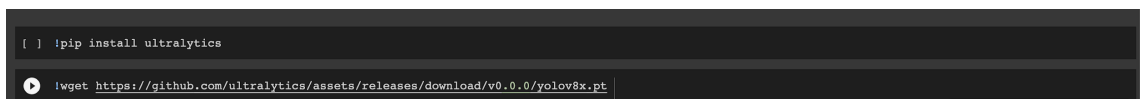
Sử dụng công cụ labeling và gán nhãn cho những file đó nhằm đa dạng hoá tập dữ liệu để giúp cho quá trình huấn luyện được chính xác hơn.



Hình 4.20 Label của các ảnh sau khi gán nhãn

+ Quá trình train:

Sử dụng nền tảng có sẵn trên Google Colab kết hợp với models YOLOv8 để tiến hành quá trình huấn luyện.



Hình 4.21 Setup thư viện và tải gói pretrain weights

Sau đó ta giải nén file dataset tải lên Google Colab

```
[ ] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

!unzip /content/coco.zip
```

Hình 4.22 Giải nén file dataset

Sau đó tạo file yaml để khai báo

```
!rm /content/coco128.yaml # nếu có file thì xóa
!echo 'train: /content/coco/images/train2017' >> /content/coco128.yaml
!echo 'val: /content/coco/images/train2017' >> /content/coco128.yaml
!echo 'nc: 5' >> /content/coco128.yaml
!echo 'names: ['coca','fanta','nuoc loc','mirinda','sprite']' >> /content/coco128.yaml
```

Hình 4.23 Tạo file yaml

Sau khi tạo file yaml xong thì train model

```
# Train với CLI
!yolo task=detect mode=train model=yolov8x.pt data=/content/coco128.yaml epochs=200 imgsz=640
```

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
181/200	14.1G	0.4201	0.2173	0.93	34	640: 100% 17/17 [00:20<00:00, 1.23s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.29it/s]
	all	267	267	1	1	0.995 0.951
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
182/200	14.1G	0.401	0.2134	0.9247	31	640: 100% 17/17 [00:20<00:00, 1.22s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.29it/s]
	all	267	267	1	1	0.995 0.95
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
183/200	14.2G	0.3944	0.2084	0.9244	28	640: 100% 17/17 [00:21<00:00, 1.24s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.30it/s]
	all	267	267	1	1	0.995 0.955
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
184/200	14.2G	0.3968	0.2166	0.9274	29	640: 100% 17/17 [00:20<00:00, 1.22s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.31it/s]
	all	267	267	0.999	1	0.995 0.952
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
185/200	14.1G	0.4067	0.2069	0.9211	39	640: 100% 17/17 [00:21<00:00, 1.25s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.29it/s]
	all	267	267	0.999	1	0.995 0.954
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
186/200	14.1G	0.4151	0.2046	0.9361	30	640: 100% 17/17 [00:20<00:00, 1.22s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.34it/s]
	all	267	267	0.999	1	0.995 0.956
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
187/200	14.2G	0.3904	0.2079	0.9161	38	640: 100% 17/17 [00:21<00:00, 1.24s/it]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 9/9 [00:06<00:00, 1.34it/s]
	all	267	267	0.999	1	0.995 0.955
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
188/200	14.2G	0.4048	0.2116	0.929	27	640: 100% 17/17 [00:21<00:00, 1.25s/it]

Hình 4.24 Quá trình huấn luyện

Kết quả của quá trình huấn luyện YOLO:



Hình 4.25 Kết quả của YOLO trong việc phát hiện đối tượng trong real-time

4.5 Ứng dụng trí tuệ nhân tạo điều khiển robot bằng cử chỉ tay

4.5.1 Công nghệ nhận diện cử chỉ tay trong robot

Một hệ thống có thể hiểu cử chỉ tay của con người có vô vàn ứng dụng thực tế. Ví dụ như trong chế tạo rô-bốt, nó có thể giúp chuyển đổi nhận dạng cử chỉ thành tín hiệu điều khiển khiến rô-bốt.

Điều khiển rô-bốt: Hệ thống nhận dạng cử chỉ tay có thể chuyển đổi các cử chỉ tay của con người thành tín hiệu điều khiển để rô-bốt thực hiện các hành động tương ứng. Ví dụ, nếu người sử dụng muốn rô-bốt di chuyển sang phải, họ có thể thực hiện cử chỉ tay phù hợp và rô-bốt sẽ nhận ra cử chỉ đó và di chuyển theo hướng tương ứng.

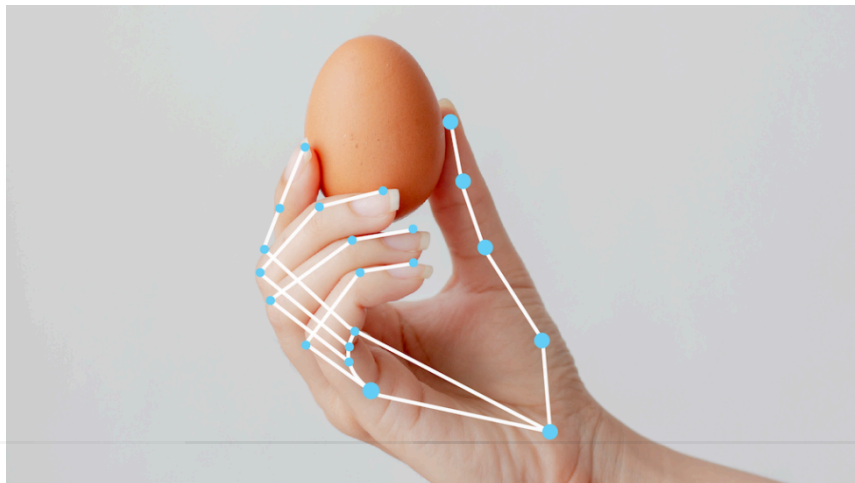
Tương tác người-máy: Hệ thống cử chỉ tay cũng có thể được sử dụng trong các giao diện người-máy để thực hiện các tác vụ cụ thể. Thay vì sử dụng bàn phím hoặc chuột, người dùng có thể điều khiển máy tính hay các thiết bị khác bằng cách thực hiện cử chỉ tay phù hợp.

Thiết kế giao diện người dùng dễ sử dụng: Hệ thống cử chỉ tay có thể giúp thiết kế các giao diện người dùng dễ sử dụng hơn và tiện lợi hơn. Thay vì phải tìm kiếm các nút hoặc menu, người dùng có thể thực hiện cử chỉ tay để thực hiện các tác vụ một cách nhanh chóng và hiệu quả hơn.

4.5.2 Mediapipe hand recogninige

a) Giới thiệu

Mediapipe hand là gói máy học (ML) mã nguồn mở của Google có thể giúp nhận dạng và theo dõi (track) bàn tay người bằng hình ảnh. Về cơ bản, Mediapipe hand kết hợp hai mô hình (model) học máy: một mô hình palm detection (nhận diện bàn tay) để tìm vị trí của bàn tay và một mô hình mốc (landmark) bàn tay để định vị chính xác 21 tọa độ khớp ngón tay từ vị trí tìm được ở palm detection). Nó được train với khoảng 30 nghìn dữ liệu thực có gắn nhãn (label) và chạy rất nhanh.



b) Đầu vào

Hand Landmarker chấp nhận đầu vào của một trong các loại dữ liệu sau:

- + Ảnh tĩnh
- + Khung hình video.
- + Nguồn cấp dữ liệu video trực tiếp

c) Đầu ra

Hand Landmarker đưa ra các kết quả sau:

- + Bàn tay của bàn tay được phát hiện
- + Các mốc của bàn tay được phát hiện trong tọa độ cục bộ
- + Các mốc của bàn tay được phát hiện trong tọa độ chung

4.5.3 Mô hình

a) Mô hình phát hiện lòng bàn tay:

Model Type: Convolutional Neural Network (CNN)

Kiến trúc mô hình (Model Architecture): Single-shot detector model(SSD).

Cách thức hoạt động của SSD có thể được tóm tắt như sau:

- + **Base Convolutional Network:** SSD sử dụng mạng neural convolutional (CNN) làm cơ sở để trích xuất các đặc trưng từ hình ảnh. Mạng CNN này thường là một mạng được huấn luyện trước, ví dụ như VGG, ResNet, hoặc MobileNet, được đề xuất để trích xuất các đặc trưng cấp cao từ hình ảnh đầu vào.
- + **Multiscale Feature Maps:** SSD tạo ra một loạt các bản đồ đặc trưng (feature -maps) có tỷ lệ khác nhau từ các lớp của mạng CNN. Các bản đồ đặc trưng này có các đặc trưng có kích thước và tỷ lệ khác nhau, giúp phát hiện các đối tượng ở nhiều kích thước khác nhau trong hình ảnh.
- + **Default Boxes (Anchor Boxes):** SSD sử dụng các hộp mẫu mặc định (default boxes) hoặc còn gọi là anchor boxes, được đặt trên các vị trí khác nhau của bản đồ đặc trưng. Các default boxes có kích thước và tỷ lệ khác nhau để có thể phát hiện đối tượng có kích thước và tỷ lệ đa dạng.
- + **Dự đoán lớp và hộp giới hạn:** Trong mỗi default box, SSD sử dụng các lớp phân loại và hồi quy để dự đoán xác suất của từng lớp đối tượng có thể có trong hộp và điều chỉnh các hộp giới hạn để tốt hơn phù hợp với đối tượng thực tế.
- + **Non-maximum Suppression (NMS):** Để giảm bớt các hộp giới hạn trùng lặp và giữ lại các hộp có độ tin cậy cao, SSD sử dụng thuật toán non-maximum suppression (NMS) để làm sạch đầu ra và đưa ra các hộp kết quả cuối cùng.

c) Mô hình nhận dạng khớp ngón tay

Kiểu mô hình: Convolutional Neural Network (CNN)

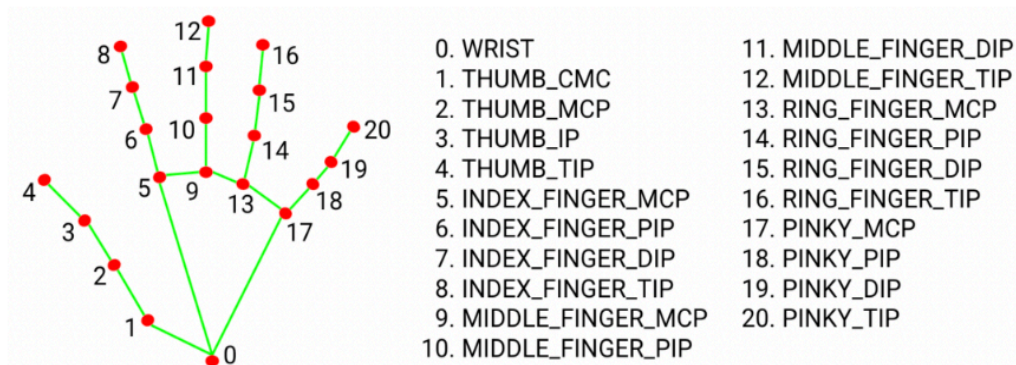
Kiến trúc mô hình: Regression model (Mô hình hồi quy)

Đầu vào: Một cắt ảnh (crop) từ khung hình video hoặc một hình ảnh, được biểu diễn dưới dạng tensor $224 \times 224 \times 3$. Kênh màu được sắp xếp theo thứ tự RGB và giá trị của pixel nằm trong khoảng $[0.0, 1.0]$.

Đầu ra:

- + Một số thực (scalar) biểu diễn xác suất (presence score) về sự hiện diện của bàn tay trong hình ảnh đầu vào. Kết quả này chỉ được coi là hợp lệ (valid) khi điểm số hiện diện (presence score) vượt qua ngưỡng (threshold).
- + 21 điểm đặc trưng trên màn hình, mỗi điểm có 3 giá trị tọa độ (tổng cộng 63 giá trị) được biểu thị bởi tensor 1×63 và chuẩn hóa theo kích thước hình ảnh. Kết quả dự đoán này chỉ được coi là hợp lệ khi điểm số hiện diện (presence score) vượt qua ngưỡng.
- + Một số thực (scalar) biểu diễn xác suất (handedness) về việc tay được dự đoán là tay trái hay tay phải. Kết quả này chỉ được coi là hợp lệ khi điểm số hiện diện (presence score) vượt qua ngưỡng.

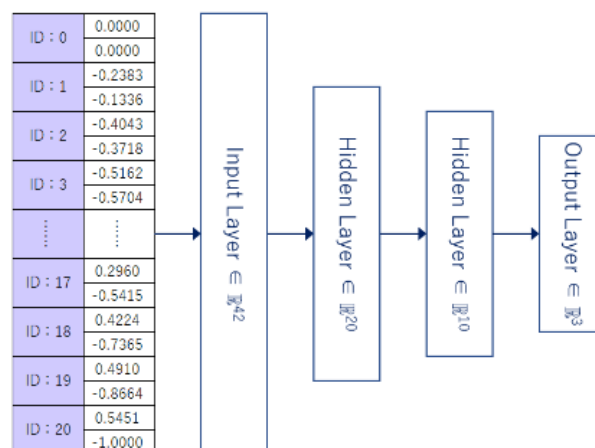
Mô hình này được sử dụng để dự đoán xác suất có mặt của bàn tay, vị trí và chiều trái/phải của tay trong ảnh, video đầu vào. Các dự đoán này chỉ coi là hợp lệ khi xác suất hiện diện (presence score) đạt đủ mức tin cậy.



Sau khi phát hiện lòng bàn tay trên toàn bộ hình ảnh, mô hình mốc bàn tay tiếp theo thực hiện định vị chính xác điểm chính của 21 tọa độ khớp ngón tay 3D bên trong vùng bàn tay được phát hiện thông qua hồi quy, đó là dự đoán tọa độ trực tiếp. Mô hình học cách thể hiện tư thế bàn tay bên trong nhất quán và mạnh mẽ ngay cả khi bàn tay có thể nhìn thấy một phần và tự che phủ. Model được train bằng 30000 dữ liệu tay được chú thích thủ công.

d) Mô hình nhận dạng cử chỉ tay

Sau khi phát hiện bàn tay và 21 điểm thì cần thêm một model để nhận dạng cử chỉ tay. Mô hình hai bước bao gồm embed mode và sau đó là classification model. Mô hình nhúng học biểu diễn chiều thấp của dữ liệu đầu vào, sau đó mô hình phân loại dự đoán lớp của các điểm đặc trưng dựa trên biểu diễn đã học.



+ Embed model

Loại mô hình: Mạng neural kết nối đầy đủ (Fully Connected Neural Network) với các khối kiến trúc dư (residual blocks).

Kiến trúc mô hình: Mô hình hồi quy (Regression model).

Đầu vào: Là đầu ra của mô hình mốc bàn tay.

Đầu ra: Một tensor số thực (float) kích thước 128x1 biểu diễn nhúng (embedding) dự đoán đại diện cho các điểm đặc trưng của bàn tay. Đầu ra này

sau đó được sử dụng tiếp trong mô hình phân loại (classification model) mô tả ở phần tiếp theo.

+ Classification model

Loại mô hình: Mạng neural kết nối đầy đủ (Fully Connected Neural Network).

Kiến trúc mô hình: Mô hình phân loại (Classification model).

Đầu vào: Đầu ra của embedded model

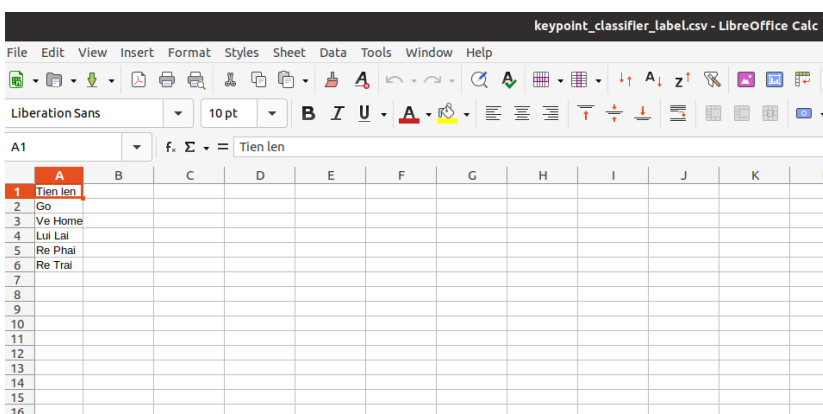
Đầu ra: Một vector gồm 6 phần tử, mỗi phần tử biểu thị xác suất cho mỗi lớp trong 6 lớp tương ứng với các cử chỉ tay: Tien len, Lui lai, Re trai, Re phai, Dung lai, Go.

4.5.4 Train nhận dạng cử chỉ tay

Các thư viện cần thiết để train mô hình: **tf-nightly**, **scikit-learn**, **matplotlib**

Các bước thực hiện:

- Bước 1: Thêm label cho kí hiệu tay cần nhận dạng vào file .csv:
keypoint_classifier_label.csv

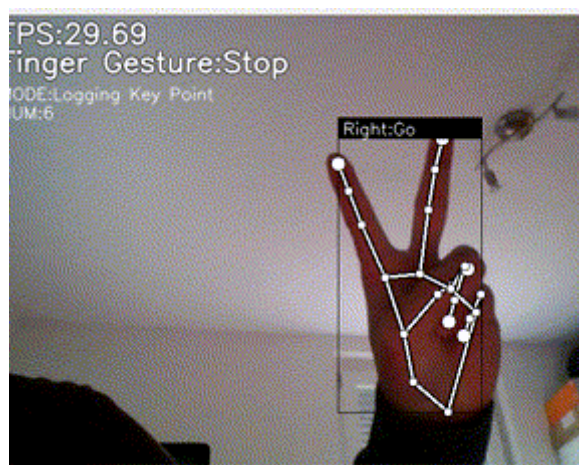


	A	B	C	D	E	F	G	H	I	J	K	L
1	Tien len											
2	Go											
3	Ve Home											
4	Lui Lai											
5	Re Phai											
6	Re Trai											
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												

Bước 2: Chúng ta cần ghi thêm các tọa độ của cử chỉ tay mới vào file keypoint.csv

Số đầu tiên trong mỗi dòng là class ID (số hiệu của lớp) tương ứng với danh sách trên, Sau đó là 42 số hoặc 21 cặp số biểu thị tọa độ của từng keypoint (tức các đốt ngón tay) đối với điểm gốc là cổ tay.

Thực hiện ghi dữ liệu bằng video:



File Edit View Insert Format Styles Sheet Data Tools Window Help												
Liberation Sans 10pt B I U A Z T												
A1	f. Σ. = 0											
1	A	B	C	D	E	F	G	H	I	J	K	L
2	0	0	0	0.2007874031574803	-0.0511811023622047	0.366141732283465	-0.181102362204724	0.484251968503937	-0.307086614173228	0.594488188976378	-0.381889763779528	0.263779527590555
3	0	0	0	0.206349206349206	-0.0436507936507937	0.376984126984127	-0.162698412698413	0.507936507936508	-0.273809523809524	0.615079365079365	-0.353174603174603	0.265873015873016
4	0	0	0	0.202380952380952	-0.0436507936507937	0.373015873015873	-0.162698412698413	0.5	-0.277777777777778	0.607142857142857	-0.353174603174603	0.261904761904762
5	0	0	0	0.207171314741036	-0.0398406374501992	0.382470119521912	-0.159362549800797	0.50996015936255	-0.274900398406375	0.617520880478088	-0.350597805061753	0.266932270916335
6	0	0	0	0.206349206349206	-0.0398406374501992	0.380952380952381	-0.162698412698413	0.507936507936508	-0.281746031746032	0.615079365079365	-0.353174603174603	0.265873015873016
7	0	0	0	0.206349206349206	-0.0476190476190476	0.376984126984127	-0.162698412698413	0.503986253986254	-0.273809523809524	0.611111111111111	-0.345238095238095	0.265873015873016
8	0	0	0	0.208	-0.044	0.384	-0.156	0.516	-0.264	0.62	-0.336	0.272
9	0	0	0	0.212	-0.04	0.388	-0.152	0.524	-0.26	0.632	-0.332	0.284
10	0	0	0	0.208835341365462	-0.0401606425702811	0.389558232931727	-0.14859437751004	0.526104417670683	-0.257028112449799	0.634538152610442	-0.333333333333333	0.281124497991968
11	0	0	0	0.208	-0.036	0.388	-0.148	0.524	-0.262	0.636	-0.328	0.296
12	0	0	0	0.208	-0.036	0.388	-0.144	0.524	-0.244	0.632	-0.316	0.276
13	0	0	0	0.208835341365462	-0.0321285140562249	0.389558232931727	-0.136546184738956	0.526104417670683	-0.240963854216867	0.634538152610442	-0.317269076365221	0.281124497991968
14	0	0	0	0.208	-0.028	0.388	-0.136	0.524	-0.236	0.632	-0.304	0.284

hình STYLEREF 1 \s 4. SEQ hình * ARABIC \s 1 34 Tọa độ các điểm của keypoint với điểm gốc tọa độ là cổ tay ID 0.

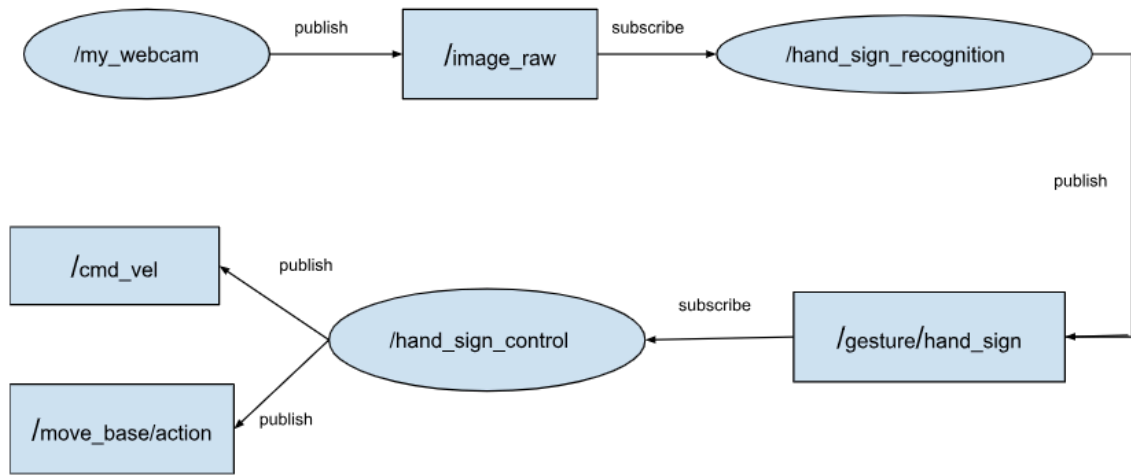
Bước 3: Thực hiện train bằng Jupyter notebook

```
"Epoch 178: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 6ms/step - loss: 0.3962 - accuracy: 0.8556 - val_loss: 0.1447 - val_accuracy: 0.9781\n",
"Epoch 179:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3868 - accuracy: 0.8582\n",
"Epoch 179: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 6ms/step - loss: 0.3820 - accuracy: 0.8592 - val_loss: 0.1366 - val_accuracy: 0.9713\n",
"Epoch 180:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.4006 - accuracy: 0.8500\n",
"Epoch 180: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 5ms/step - loss: 0.3985 - accuracy: 0.8498 - val_loss: 0.1487 - val_accuracy: 0.9787\n",
"Epoch 181:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3945 - accuracy: 0.8535\n",
"Epoch 181: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 5ms/step - loss: 0.3901 - accuracy: 0.8569 - val_loss: 0.1457 - val_accuracy: 0.9688\n",
"Epoch 182:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3847 - accuracy: 0.8654\n",
"Epoch 182: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 4ms/step - loss: 0.3864 - accuracy: 0.8623 - val_loss: 0.1377 - val_accuracy: 0.9719\n",
"Epoch 183:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3739 - accuracy: 0.8557\n",
"Epoch 183: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 5ms/step - loss: 0.3953 - accuracy: 0.8473 - val_loss: 0.1366 - val_accuracy: 0.9726\n",
"Epoch 184:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3958 - accuracy: 0.8530\n",
"Epoch 184: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 6ms/step - loss: 0.3948 - accuracy: 0.8552 - val_loss: 0.1403 - val_accuracy: 0.9726\n",
"Epoch 185:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3781 - accuracy: 0.8649\n",
"Epoch 185: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 6ms/step - loss: 0.3803 - accuracy: 0.8615 - val_loss: 0.1402 - val_accuracy: 0.9719\n",
"Epoch 186:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3858 - accuracy: 0.8648\n",
"Epoch 186: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 5ms/step - loss: 0.3890 - accuracy: 0.8635 - val_loss: 0.1471 - val_accuracy: 0.9694\n",
"Epoch 187:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3769 - accuracy: 0.8563\n",
"Epoch 187: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 5ms/step - loss: 0.3921 - accuracy: 0.8552 - val_loss: 0.1435 - val_accuracy: 0.9719\n",
"Epoch 188:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.3916 - accuracy: 0.8628\n",
"Epoch 188: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 4ms/step - loss: 0.3969 - accuracy: 0.8621 - val_loss: 0.1449 - val_accuracy: 0.9781\n",
"Epoch 189:1000\n",
"38/38 [=====] - ETA: 0s - loss: 0.4165 - accuracy: 0.8532\n",
"Epoch 189: saving model to /home/giacuong/catkln_ws/src/ros_hand_gesture_recognition/src/model/keypoint_classifier/keypoint_classifier.hdf5\n",
"38/38 [=====] - 0s 6ms/step - loss: 0.4119 - accuracy: 0.8517 - val_loss: 0.1485 - val_accuracy: 0.9781\n",
"Epoch 189: early stopping\n"
```

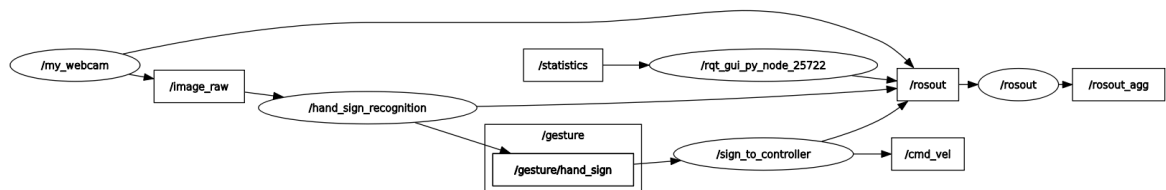
4.5.5 Kết nối và gửi dữ liệu điều khiển tới robot

a) Nguyên lý hoạt động:

Topic /image sẽ nhận dữ liệu hình ảnh trực tiếp từ camera sau đó topic /hand_sign_recognition nhận và xử lý nhận dạng kí hiệu tay publish lên topic /hand_sign. Node /hand_sign_control nhận dữ liệu cử chỉ tay và publish vận tốc, tọa độ để điều khiển Robot.



Hình 4.36 Lưu đồ hoạt động điều khiển robot bằng cử chỉ tay



/image_raw: dữ liệu hình ảnh từ webcam

/gesture/hand_sigh: chứa dữ liệu nhận diện cử chỉ

Các node:

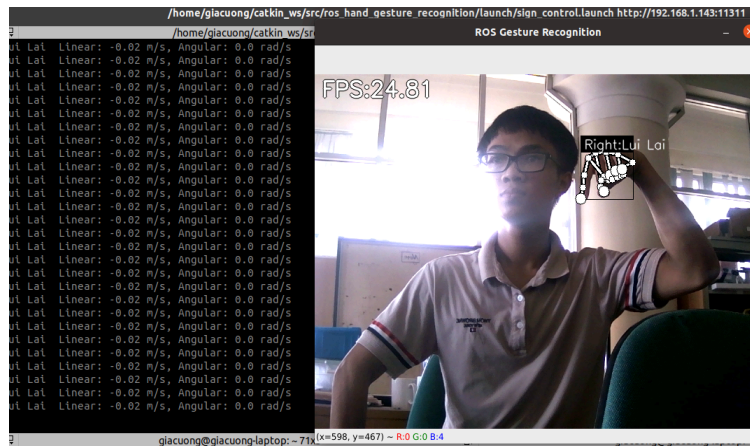
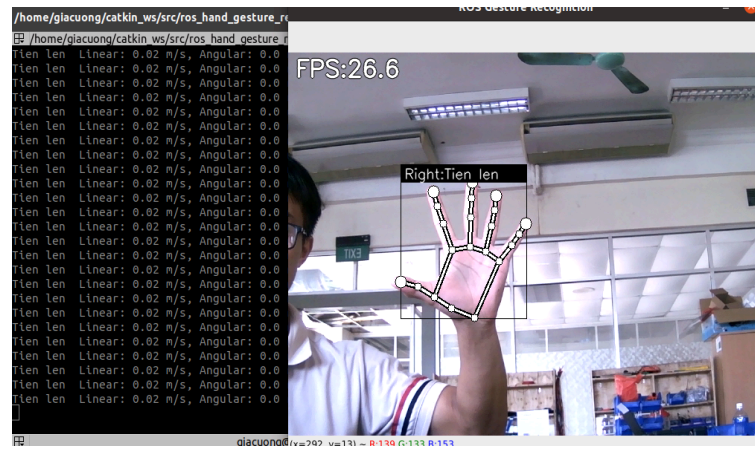
/my_webcam: publish hình ảnh

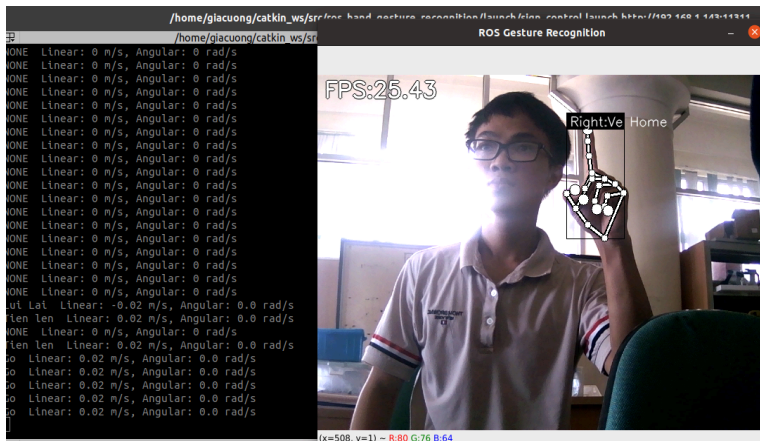
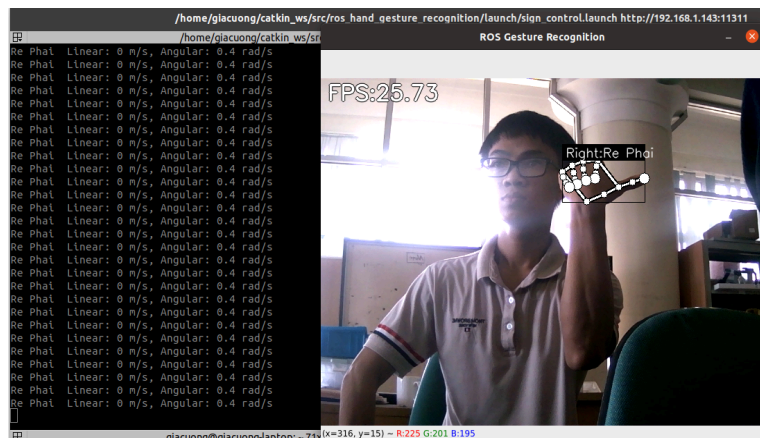
/hand_sign_recognition: subscribe hình ảnh và thực hiện nhận diện cử chỉ tay

/sign_to_controller: subscribe cử chỉ tay và gửi tín hiệu điều khiển xuống

/cmd_vel và /move_base_action

b) Kết quả:





hình STYLEREF 1 \s 4. SEQ hình * ARABIC \s 1 38
 Kết quả điều khiển Robot bằng cử chỉ tay.

CHƯƠNG 5. THIẾT KẾ GIAO DIỆN

5.1 Giới thiệu

Chương này sẽ giới thiệu các công nghệ ứng dụng để tạo giao diện người máy giúp giao tiếp, điều khiển robot bao gồm giao diện trên web, app, giao diện trên màn hình HDMI của robot. Các công nghệ sử dụng bao gồm html, css, js, tạo giao diện web, flutter tạo giao diện app kết nối với robot thông qua rosbigde, python thinker tạo giao diện HDMI.

5.2 Thiết kế giao diện web, app order món ăn

5.2.1 HTML

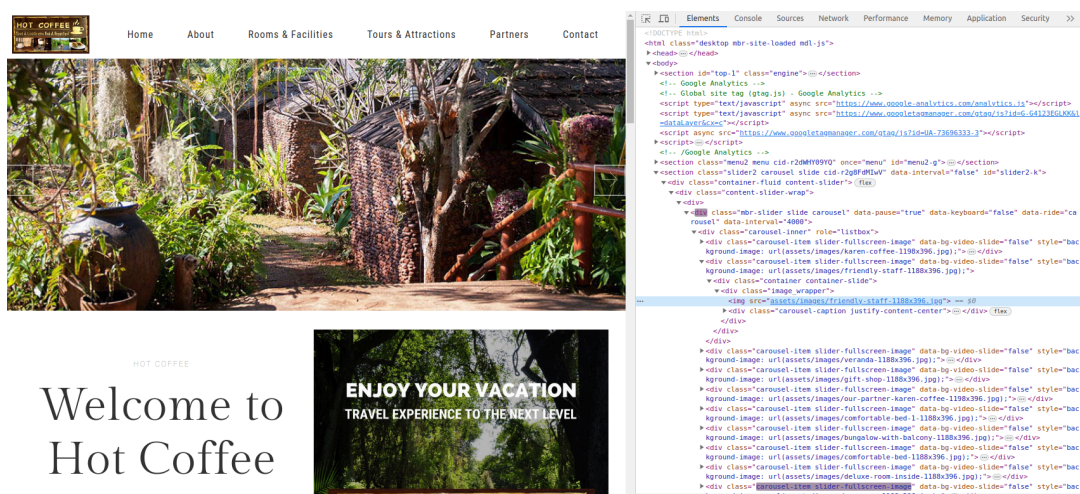
a) Khái niệm:

HTML (HyperText Markup Language) là một ngôn ngữ đánh dấu sử dụng để xây dựng và cấu trúc nội dung trên các trang web. HTML mô tả cấu trúc và ý nghĩa của các phần tử trên trang, bao gồm tiêu đề, đoạn văn bản, hình ảnh, bảng, liên kết, v.v. Nó sử dụng các thẻ (tag) để đánh dấu và phân loại các phần tử, và được trình duyệt web sử dụng để hiển thị nội dung một cách phù hợp.

Vào cuối năm 1991, phiên bản HTML đầu tiên do Tim Berners-Lee phát triển đã được công khai với tên HTML Tags. Phiên bản này có thiết kế vô cùng đơn giản, mô tả 18 phần tử. Tiếp đến vào năm 1995, IETF đã hoàn thành “HTML 2.0”. Sau đó phiên bản HTML 4.01 được công bố vào năm 1999. Đến năm 2000, các phiên bản HTML đã được các nhà phát triển thay thế bằng XHTML. Năm 2014, HTML được nâng cấp lên chuẩn HTML5 với sự cải tiến rõ rệt.

b) Cách hoạt động:

HTML document có đuôi file dạng .html hoặc .htm. Bạn có thể xem chúng bằng các trình duyệt web như Google Chrome, Firefox, Safari, v.v. Trình duyệt đọc những file HTML này và biến đổi chúng thành nội dung trực quan trên Internet để người dùng có thể xem và hiểu được.



Một trang web thường có nhiều HTML document, mỗi trang con có một tệp HTML riêng. Mỗi tài liệu HTML được xây dựng bằng cách sử dụng các thẻ (tags) để tạo cấu trúc tương tự cây thư mục với các tiêu đề, phần, đoạn văn bản,

v.v. Hầu hết các thẻ HTML đều có một thẻ mở và một thẻ đóng, ví dụ: `<tag></tag>`."

c) Các loại HTML hiện nay

- HTML4

HTML 4 ra đời vào năm 1997 là phiên bản thứ 4 của ngôn ngữ đánh dấu siêu văn bản. Phiên bản HTML này được xuất bản dưới dạng như một W3C Recommendation. HTML 4 áp dụng cho nhiều phần tử và thuộc tính khác nhau cho trình duyệt web.

- HTML5

HTML5 phiên bản thứ 5 của HTML được công bố bởi World Wide Web Consortium (W3C). HTML5 là sự kết hợp giữa HTML 4, XHTML, DOM cấp 2 và JavaScript. HTML5 hỗ trợ chạy trên mọi trình duyệt.

- XHTML

XHTML có tên đầy đủ là Extensible HyperText Markup Language, nghĩa là ngôn ngữ đánh dấu siêu văn bản mở rộng. Đây là một ngôn ngữ thay thế của HTML với cú pháp chặt chẽ hơn. Cụ thể, XHTML yêu cầu mọi phần tử được đóng bằng thẻ đóng hoặc cú pháp tự đóng riêng và phân biệt được chữ in hoa hoặc chữ in thường, trong khi đó HTML không có điều này.

d) Cấu trúc của HTML

Một file mã HTML được hình thành từ các phần tử HTML và các cặp thẻ. HTML bao gồm nhiều loại thẻ khác nhau và mỗi thẻ có nhiệm vụ và ý nghĩa riêng. Các thẻ được đặt trong cặp dấu ngoặc nhọn "<, >", và các từ giữa các dấu ngoặc này được gọi là phần tử. Cấu trúc cơ bản của một trang HTML bao gồm ba phần:

- + Phần khai báo loại file code với thẻ `<!DOCTYPE html>`: Được đặt ở đầu hoặc trên cùng của file HTML. Phần này cho biết loại tài liệu HTML và phiên bản HTML mà trang web sử dụng.
- + Phần khai báo ban đầu (head), chứa các khai báo về meta, link, JavaScript, CSS, v.v.: Bắt đầu bằng thẻ `<head>` và kết thúc bằng thẻ `</head>`. Phần này chứa tiêu đề trang web (được đặt giữa cặp thẻ `<title>` và `</title>`) và các khai báo khác như mô tả trang, liên kết đến các tệp CSS và JavaScript, và các thông tin khác liên quan đến trang web.
- + Phần chứa và hiển thị nội dung của trang web (body): Bắt đầu bằng thẻ `<body>` và kết thúc bằng thẻ `</body>`. Phần này chứa nội dung chính của trang web như tiêu đề, nội dung bài viết, hình ảnh, đường dẫn, các phần tử tạo bố cục, và các phần tử khác như liên kết, bảng, biểu đồ, v.v.

Các tag thông dụng nhất của HTML:

Block-level tags

Đây là loại tag cấp cao, sẽ chiếm toàn bộ không gian của trang web và luôn bắt đầu dòng mới của trang. 3 block-level tags mà tất cả các trang

HTML đầu cần có đó là: `<html></html>`, `<head></head>` và `<body></body>`.

Inline tags

Loại tag này chỉ chiếm một phần nhỏ trong không gian Website và cũng không bắt đầu dòng mới của trang. Inline tags thường được dùng để định dạng, tạo bố cục cho nội dung bên trong của block-level tags thường có 5 nhóm tag chính:

- + **Tag văn bản:** Tag văn bản dùng để tạo một đoạn văn bản trong code HTML. Để tạo văn bản, bạn cần dùng thẻ `<p>`. Tiếp đó, bạn có thể sử dụng chuỗi ký tự “ ” để tạo khoảng trắng giữa 2 ký tự. Khi muốn ngắt dòng thì dùng thẻ `<br>`.
- + **Tag định dạng:** Tag định dạng dùng để đánh dấu hay nhấn mạnh một số điều vào văn bản HTML.
- + **Tag hình ảnh:** Tag hình hay tag `<img />` được dùng để đưa một hình ảnh vào văn bản HTML. Loại tag này có 2 thuộc tính cần thiết là `src` và `alt`. Cấu trúc của tag hình ảnh như sau `<img src="" alt="" />`
- + **Tag liên kết:** Tag liên kết dùng để xác định mối quan hệ giữa một file HTML và các tài nguyên bên ngoài. Tag liên kết được dùng phổ biến nhất là link tới CSS. Một số cấu trúc của tag liên kết bao gồm `<link rel="stylesheet" type="text/css" href="filecss.css" />`, `<head>`
- + **Tag form:** Tag form được sử dụng để tạo một “biểu mẫu” HTML cho người dùng. Bên cạnh đó tag form còn được dùng để truyền tải dữ liệu lên máy chủ và không dùng để điều khiển layout. Cấu trúc của tag form bắt buộc phải có tham số `action`: `<form action=""></form>`. Ngoài ra, bên trong tag form có thể chứa một số tag khác.

e) Ưu nhược điểm HTML

HTML là một công cụ hữu hiệu giúp người dùng thiết kế và tạo nội dung nhanh chóng cho trang web. HTML sở hữu nhiều ưu điểm nổi bật như sau:

- + Kho tài nguyên khổng lồ với cộng đồng người dùng rộng lớn
- + Sử dụng mã nguồn mở nên người dùng có thể sử dụng miễn phí
- + Hoạt động mượt mà trên hầu hết các trình duyệt
- + Cách thức hoạt động đơn giản nên người học có thể dễ dàng nắm bắt được kiến thức để triển khai cho website của mình
- + Có thể tích hợp với nhiều ngôn ngữ khác nhau như PHP, Java, NodeJs, Ruby,... giúp người dùng xây dựng được một trang web với nhiều tính năng hấp dẫn
- + Sử dụng các markup ngắn gọn và có tính đồng nhất cao
- + Quy định theo một tiêu chuẩn nhất định và được vận hành bởi World Wide Web Consortium (W3C)

Bên cạnh những ưu điểm trên, HTML vẫn còn tồn tại một số nhược điểm:

- + Chỉ áp dụng được cho web tĩnh, không có sự tương tác với người dùng. Nếu muốn trang bị cho web các tính năng tự động thì cần phải sử dụng dịch vụ của bên thứ 3.

- + Một số trình duyệt còn cập nhật chậm để hỗ trợ các phiên bản mới của HTML, đặc biệt là HTML5
- + Việc kiểm soát cách đọc và hiển thị file HTML của trình duyệt khó thực hiện
- + Chỉ áp dụng với cấu trúc nhất định, không có khả năng sáng tạo

5.2.2 CSS

a) Giới thiệu

CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (ví dụ như HTML), có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,...thì CSS sẽ giúp chúng ta có thể thêm một chút “phong cách” vào các phần tử HTML đó như đổi màu sắc trang, đổi màu chữ, thay đổi cấu trúc,...rất nhiều.

Các loại CSS:

- + **CSS1:** CSS1 được công bố lần đầu tiên vào năm 1996 bởi W3C (World Wide Web Consortium). Nó định nghĩa các thuộc tính và các quy tắc đơn giản để kiểm soát kiểu dáng của các phần tử HTML. CSS1 đặt nền móng cho việc tách biệt nội dung và kiểu dáng trong phát triển web.
- + **CSS2:** CSS2 được công bố vào năm 1998 và cung cấp nhiều tính năng mới và cải tiến so với CSS1. CSS2 giới thiệu các khối mô-đun mới như bố cục, hiệu ứng, và trình đơn. Nó cũng mở rộng cú pháp CSS và hỗ trợ nhiều lựa chọn kiểu dáng hơn cho các phần tử HTML.
- + **CSS3:** CSS3 là phiên bản tiếp theo của CSS và đang tiếp tục được phát triển. CSS3 được chia thành nhiều mô-đun nhỏ, mỗi mô-đun chứa các tính năng và quy tắc kiểu dáng mới. CSS3 giới thiệu nhiều tính năng đáng chú ý như độ đàn hồi (flexbox), đồ họa (transitions, animations), viền (border-radius, box-shadow), v.v. CSS3 cung cấp khả năng tùy chỉnh và linh hoạt cao hơn cho phát triển web.

Tại đồ án này chúng em sử dụng **CSS3**

b) Hoạt động và cách sử dụng CSS

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.

Khái niệm **property, value, selector**:

- + **Property** là thuộc tính của phần tử HTML, ví dụ: font-size (kích cỡ chữ của phần tử), color (màu chữ của phần tử), background-color (màu nền của phần tử)
- + **Value** là giá trị của thuộc tính.
- + **Selector** (bộ chọn) là phần tử mà bạn muốn định dạng.

Selector	Ý Nghĩa	Ví Dụ
Universal Selector	Loại selector này sẽ tác động đến tất cả các thẻ HTML trên trang web	<code>*{ property: value; }</code>
Type Selector	Loại CSS này sẽ tác động đến các thẻ HTML mà chúng ta muốn tác động	<code>h1, h2, h3 {property: value;}</code>
Class Selector	Loại selector này sẽ tác động đến tất cả các thẻ HTML có class mà nó quy định	<code>.note {property: value;}; p.note {};</code>
ID Selector	Loại selectors này sẽ tác động đến tất cả các thẻ HTML có id mà nó quy định	<code>#introduction {property: value;}</code>
Child Selector	Loại css này sẽ cho phép chúng ta css vào các thẻ HTML theo cấp cha con	<code>li>a {property: value;}</code>
Descendant Selector	Chọn bất kì phần tử con nào trong phần tử cha	<code>p a {property: value;}</code>
Adjacent Sibling Selector	Nhằm chọn các phần tử anh chị em	<code>h1+p {property: value;}</code>
General Sibling Selector	Nhằm chọn các phần tử anh chị em mặc dù nó không nhất thiết phải là phần tử đứng trước trực tiếp	<code>h1~p {property: value;}</code>

Để định dạng cho một phần tử, ta sử dụng cú pháp như sau:

```
selector { property1: value;
property2: value; property3: value;
..... propertyN: value; }
```

c) Cách sử dụng CSS

Có 3 cách sử dụng CSS:

- + Inline CSS: sử dụng thuộc tính style trong các thẻ HTML để áp dụng các quy tắc CSS trực tiếp
- + External CSS: tạo một tệp CSS riêng và sau đó liên kết tệp CSS đó với tài liệu HTML bằng cách sử dụng thẻ <link> trong phần head của tài liệu HTML:

5.2.3 JavaScript

a) Khái niệm

JavaScript là ngôn ngữ lập trình website phổ biến hiện nay, nó được tích hợp và nhúng vào HTML giúp website trở nên sống động hơn. JavaScript đóng vai trò như là một phần của trang web, thực thi cho phép Client-side script từ phía người dùng cũng như phía máy chủ (Nodejs) tạo ra các trang web động.

Brendan Eich chính là người đã phát triển JS tại Netscape với tiền thân là Mocha. Sau đó, Mocha được đổi thành LiveScript và cuối cùng mới đổi thành JavaScript.

Năm 1998, JavaScript với phiên bản mới nhất là ECMAScript 2 phát hành và đến năm 1999 thì ECMAScript 3 được ra mắt.

Năm 2016, ứng dụng JavaScript đã đạt kỷ lục lên tới 92% website sử dụng, đồng thời cũng được đánh giá là một công cụ cực kỳ quan trọng đối với lập trình viên.

b) Hoạt động của js

- + **Thay đổi nội dung HTML:** Một trong số nhiều phương thức HTML JavaScript chính là `getElementById()`. Chúng được sử dụng để tìm một phần tử của HTML với `id="demo"` và dùng để thay đổi nội dung của phần tử (Internal HTML) sang thành "Hello JavaScript"
- + **Thay đổi giá trị thuộc tính HTML:** Tổng quan về javascript còn có thể sử dụng để thay đổi các giá trị của thuộc tính. Ví dụ: thay đổi thuộc tính `src` (source) của tag `<img>`.
- + **Thay đổi kiểu HTML:** Đây chính là một hoạt động biến thể của việc thay đổi thuộc tính của HTML ở trên. Ví dụ: `document.getElementById('demo').style.fontSize = '35px';`
- + **Ẩn các phần tử HTML:** Một hoạt động tiếp theo là Javascript có thể ẩn được các phần tử HTML. Chúng có thể được thực hiện thông qua hoạt động thay đổi kiểu hiển thị các phần tử HTML.
- + **Hiển thị các phần tử HTML:** Một điểm đặc biệt là JavaScript có thể hiển thị được các yếu tố HTML ẩn. Đồng thời, cũng có thể thực hiện được thông qua cách thay đổi kiểu hiển thị phần tử.

c) Ưu nhược điểm js

Ưu điểm:

- + Chương trình rất dễ học.
- + Những lỗi Javascript rất dễ để phát hiện, từ đó giúp bạn sửa lỗi một cách nhanh chóng hơn.
- + Những trình duyệt web có thể dịch thông qua HTML mà không cần sử dụng đến một compiler.
- + JS có thể hoạt động ở trên nhiều nền tảng và các trình duyệt web khác nhau.
- + Được các chuyên gia đánh giá là một loại ngôn ngữ lập trình nhẹ và nhanh hơn nhiều so với các ngôn ngữ lập trình khác.
- + JS còn có thể được gắn trên một số các element hoặc những events của các trang web.
- + Những website có sử dụng JS thì chúng sẽ giúp cho trang web đó có sự tương tác cũng như tăng thêm nhiều trải nghiệm mới cho người dùng.
- + Người dùng cũng có thể tận dụng JS với mục đích là để kiểm tra những input thay vì cách kiểm tra thủ công thông qua hoạt động truy xuất database.
- + Giao diện của ứng dụng phong phú với nhiều thành phần như Drag and Drop, Slider để cung cấp đến cho người dùng một Rich Interface (giao diện giàu tính năng).
- + Giúp thao tác với người dùng phía Client và tách biệt giữa các Client với nhau.

Nhược điểm

- + JS Code Snippet khá lớn.
- + JS dễ bị các hacker và scammer khai thác hơn.

- + JS cũng không có khả năng đa luồng hoặc đa dạng xử lý.
- + Có thể được dùng để thực thi những mã độc ở trên máy tính của người sử dụng.
- + Những thiết bị khác nhau có thể sẽ thực hiện JS khác nhau, từ đó dẫn đến sự không đồng nhất.
- + Vì tính bảo mật và an toàn nên các Client-Side Javascript sẽ không cho phép đọc hoặc ghi các file.
- + JS không được hỗ trợ khi bạn sử dụng ở trong tình trạng thiết bị được kết nối mạng.

5.2.4 Giao diện web

Giao diện tạo ra gồm hai trang chính:

- + Trang gọi món dành cho khách hàng: Gồm giao diện logo quán, thẻ nav, và giao diện order có hiển thị món đồ uống, giá, chọn bàn và gọi đồ uống
- + Trang quản lý order của nhân viên: bao gồm một bảng cho thấy số bàn khách đặt, đồ uống khách đã đặt, tính tổng giá, nút xác nhận và giao đồ ra lệnh cho robot.

COFFEE SHOP

MỜI QUÝ KHÁCH ORDER:

CHỌN BÀN

BÀN 1

BÀN 2

BÀN 3

BÀN 4

MENU



Cafe Capuchino
30.000đ



Baileys milk coffee
35.000đ



Caramel latte
40.000đ



cepresso
36.000đ



raspberry mocha
50.000đ



Irish
60.000đ



Frappe
40.000đ



Mocha
90.000đ

MÓN ĐÃ CHỌN :

TỔNG BILL:

GỌI MÓN

XÁC NHẬN

Tên bàn	Món ăn	Tổng giá	Xác nhận
Bàn 1			XÁC NHẬN
Bàn 2			XÁC NHẬN
Bàn 3			XÁC NHẬN
Bàn 4			XÁC NHẬN

GIAO ĐÓ

5.2.5 Flutter

a) Khái niệm

Flutter là mobile UI framework của Google để tạo ra các giao diện chất lượng cao trên iOS và Android trong khoảng thời gian ngắn. Flutter hoạt động với những code sẵn có được sử dụng bởi các lập trình viên, các tổ chức. Flutter hoàn toàn miễn phí và cũng là mã nguồn mở.

b) Ưu điểm của Flutter

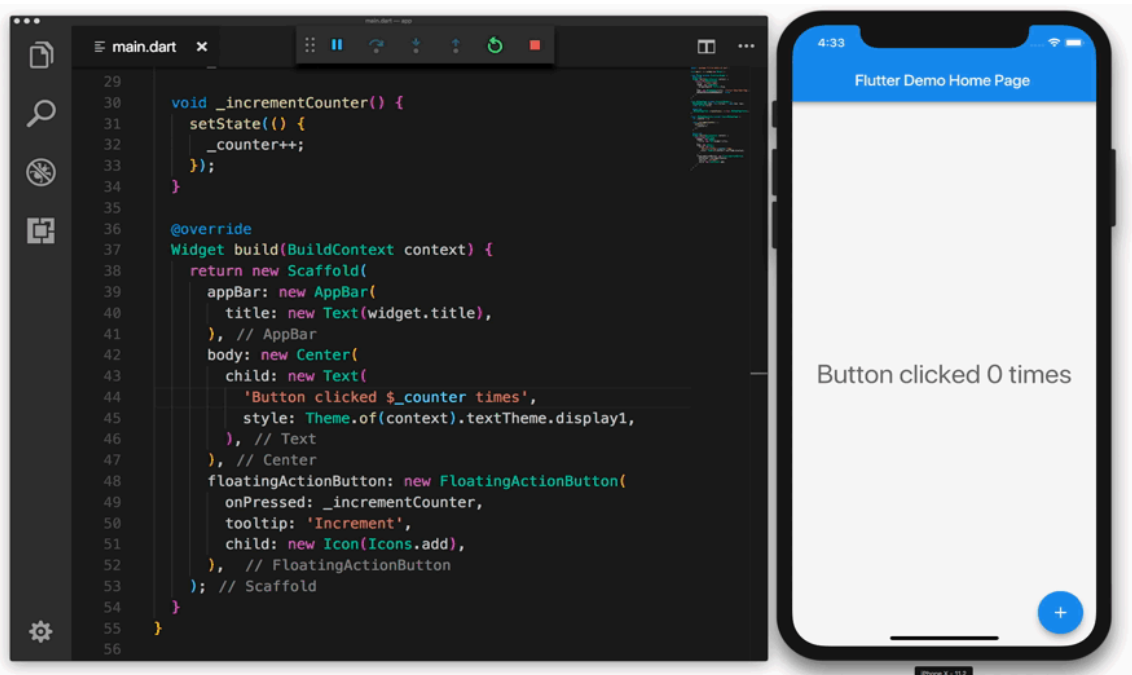
- + Nếu bạn đang tìm kiếm các phương pháp thay thế để phát triển ứng dụng Android, bạn nên cân nhắc thử Flutter của Google, một framework dựa trên ngôn ngữ lập trình Dart.
- + Các ứng dụng được xây dựng với Flutter hầu như không thể phân biệt với những ứng dụng được xây dựng bằng cách sử dụng Android SDK, cả về giao diện và hiệu suất. Hơn nữa, với những tinh chỉnh nhỏ, chúng có thể chạy trên thiết bị iOS.
- + Chạy ở 60 fps, giao diện người dùng được tạo ra với Flutter thực thi tốt hơn nhiều so với những ứng dụng được tạo ra với các framework phát triển đa nền

tảng khác chẳng hạn như React Native và Ionic. Một số lý do khiến bạn có thể hứng thú với Flutter:

- + Flutter sử dụng Dart, một ngôn ngữ nhanh, hướng đối tượng với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.
- + Flutter có các thành phần UI của riêng nó, cùng với một cơ chế để kết xuất chúng trên nền tảng Android và iOS. Hầu hết các thành phần giao diện người dùng, đều sẵn dùng, phù hợp với các nguyên tắc của Material Design.
- + Các ứng dụng Flutter có thể được phát triển bằng cách sử dụng IntelliJ IDEA, một IDE rất giống với Android Studio.

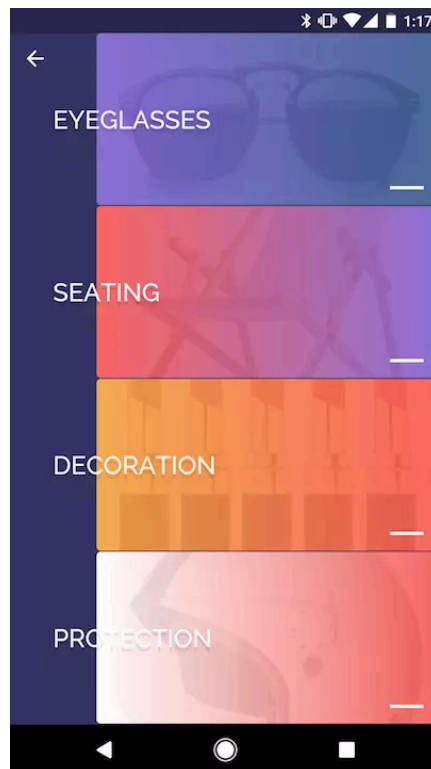
c) Tính năng của Flutter

Fast Development: Tính năng Hot Reload hoạt động trong milliseconds để hiển thị giao diện tới bạn. Sử dụng tập hợp các widget có thể customizable để xây dựng giao diện trong vài phút. Ngoài ra Hot Reload còn giúp bạn thêm các tính năng, fix bug tiết kiệm thời gian hơn mà không cần phải thông qua máy ảo, máy android hoặc iOS.



hình 5.5 Quá trình debug và chạy máy ảo

Expressive and Flexible UI: Có rất nhiều các thành phần để xây dựng giao diện của Flutter vô cùng đẹp mắt theo phong cách Material Design và Cupertino, hỗ trợ nhiều các APIs chuyển động, smooth, scrolling,...



Native Performance: Các widget của flutter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

d) Cài đặt flutter

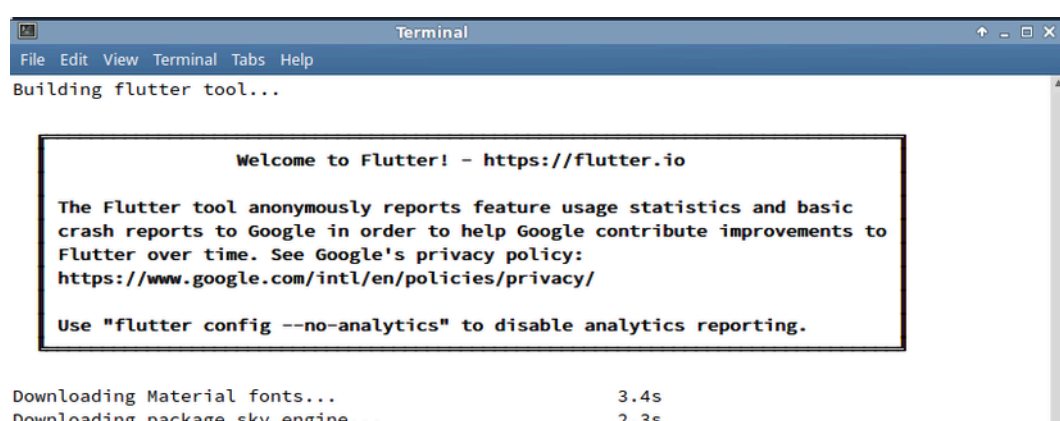
Có thể cài đặt bằng cách clone Flutter repository từ Github về:

```
git clone https://github.com/flutter/flutter.git
```

Tiếp theo bạn nên sử dụng công cụ chuẩn đoán của Flutter để kiểm tra các thành phần như Dart SDK, font Material Design:

```
cd flutter/bin
```

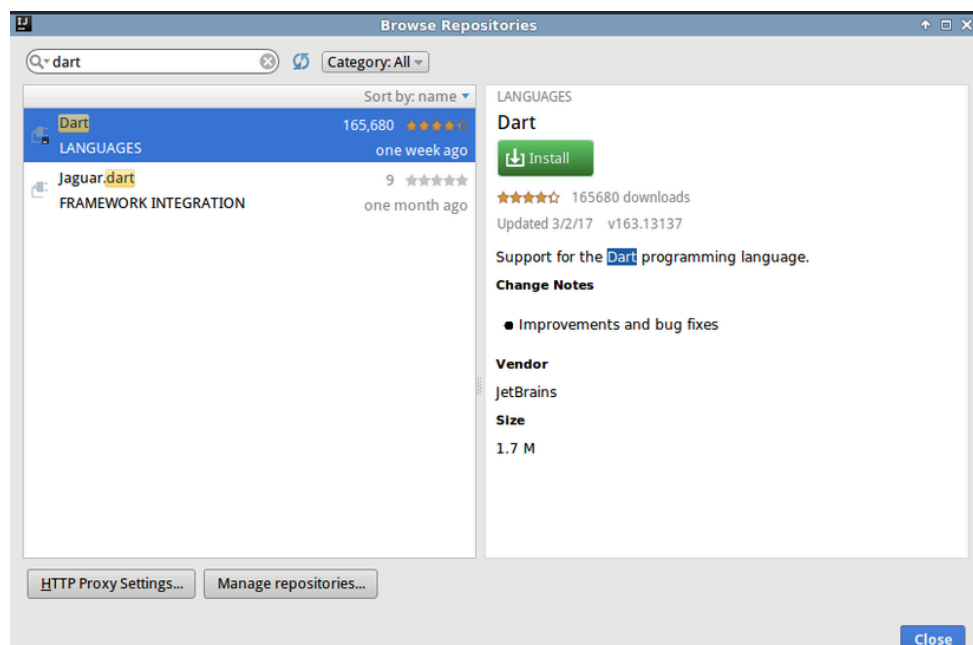
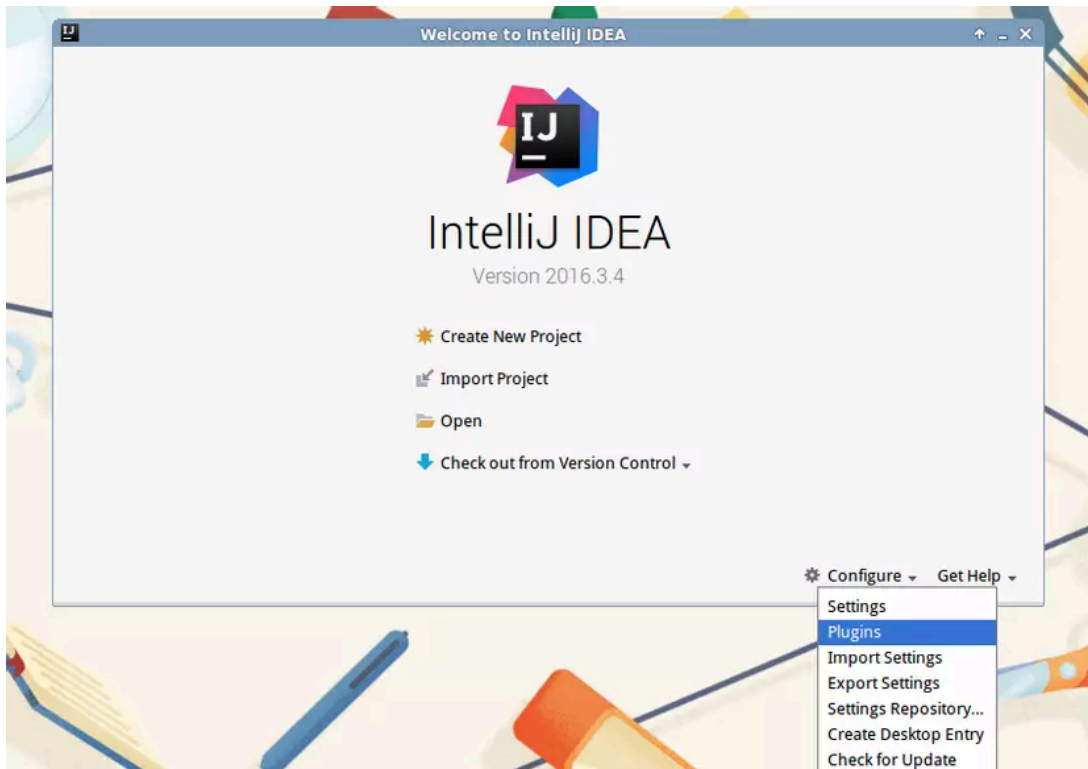
```
./flutter doctor
```

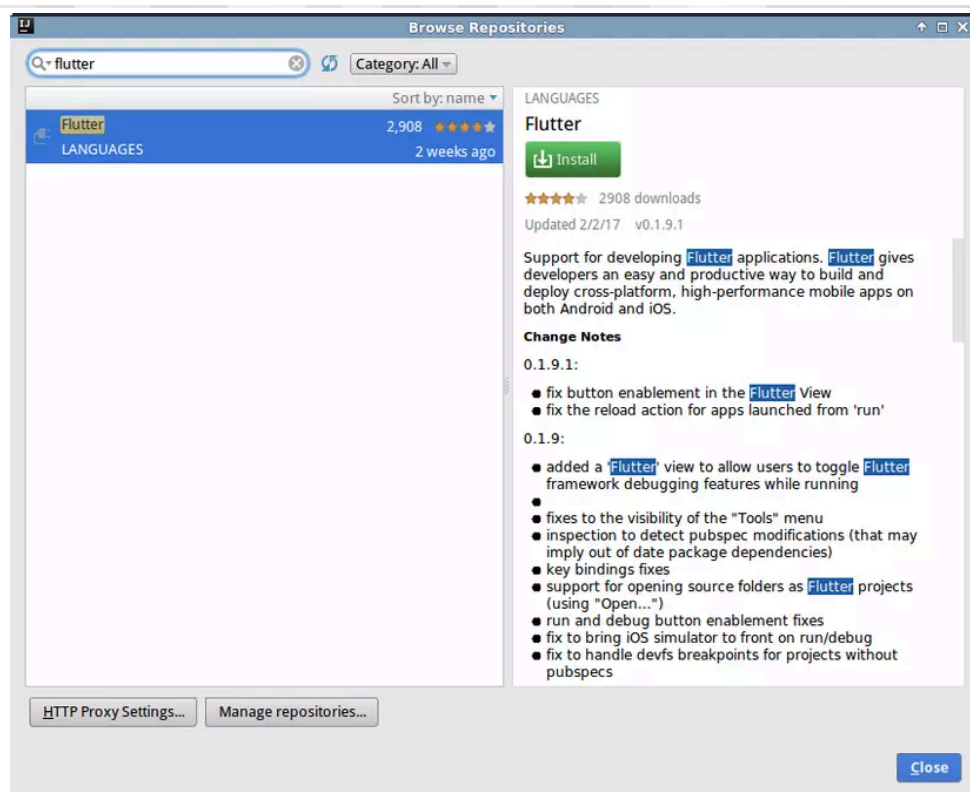


Để xây dựng ứng dụng Android, bạn phải trỏ Flutter tới thư mục của Android Studio

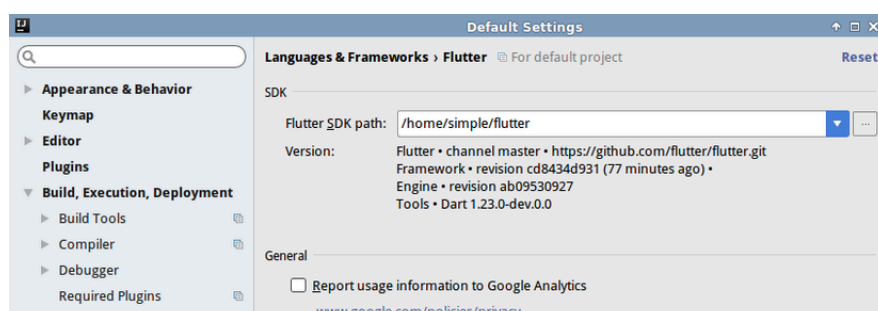
```
./flutter config --android-studio-dir ~/android-studio
```

Cấu hình với IntelliJ hoặc Android Studio, chọn Configure->Plugins tại màn hình khởi động. Sau đó cài đặt plugin Dart





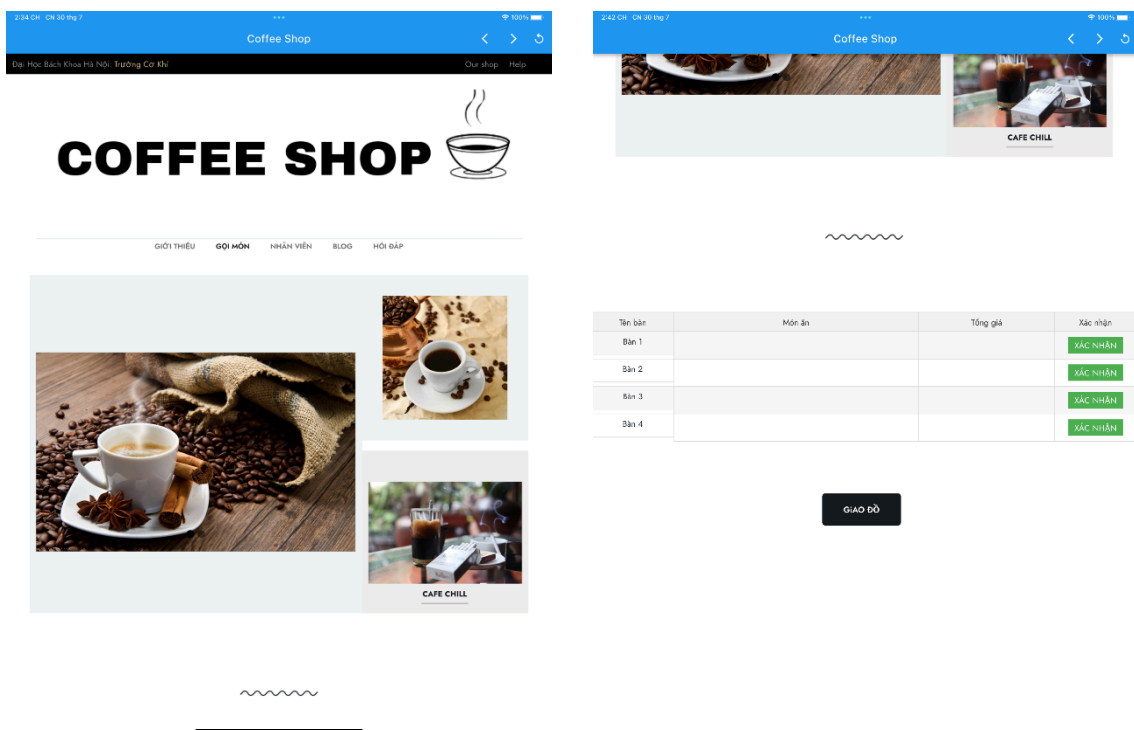
Bây giờ, bạn phải trở plugin Flutter đến thư mục mà bạn đã cài đặt **Flutter**. Để làm như vậy, hãy chọn **Configure > Settings** trong màn hình chào mừng và trong hộp thoại bật lên, điều hướng đến **Languages & Frameworks > Flutter**. Trong trường Flutter SDK path, gõ đường dẫn tuyệt đối của thư mục.



e) Giao diện bằng Flutter

Giao diện tạo ra gồm hai màn hình:

- + Màn hình gọi món dành cho khách hàng: Gồm giao diện logo quán, thẻ nav, và giao diện order có hiển thị món đồ uống, giá, chọn bàn và gọi đồ uống
- + Màn hình quản lý order của nhân viên: bao gồm một bảng cho thấy số bàn khách đặt, đồ uống khách đã đặt, tính tổng giá, nút xác nhận và giao đồ ra lệnh cho robot.



hình 5.10 Giao diện khách hàng, giao diện nhân viên

5.2.6 Rosbridge và thư viện Rosjs

a) Giới thiệu:

Rosbridge là một giao diện cung cấp khả năng giao tiếp giữa ROS (Robot Operating System) và các ứng dụng và dịch vụ bên ngoài như ứng dụng web, ứng dụng di động, hệ thống điều khiển từ xa và các hệ thống khác.

Rosbridge cho phép truyền thông giữa ROS và các ứng dụng bên ngoài bằng cách sử dụng giao thức JSON (JavaScript Object Notation) và giao thức WebSocket. Nó cung cấp một cầu nối để truyền tải dữ liệu ROS, gọi các dịch vụ ROS và điều khiển các robot và thiết bị thông qua các yêu cầu từ các ứng dụng không phải ROS.

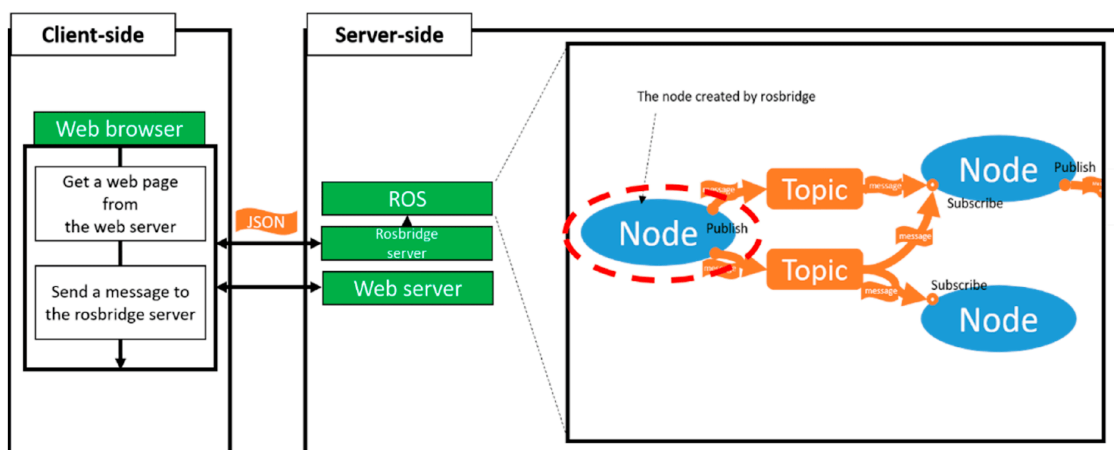
b) Rosbridge cung cấp một số chức năng chính:

Giao tiếp đa nền tảng: Rosbridge cho phép các ứng dụng từ nhiều nền tảng khác nhau (như web, di động) giao tiếp với ROS thông qua giao thức JSON và WebSocket.

Truyền tải dữ liệu ROS: Rosbridge cho phép truyền tải dữ liệu từ các chủ đề (topics) ROS, cho phép ứng dụng bên ngoài theo dõi và nhận dữ liệu từ robot và các thiết bị khác trong mạng ROS.

Gọi dịch vụ ROS: Rosbridge cho phép ứng dụng bên ngoài gọi các dịch vụ (services) ROS để thực hiện các tác vụ như kiểm soát robot, truy xuất dữ liệu, v.v.

Điều khiển robot: Rosbridge cho phép ứng dụng bên ngoài gửi các yêu cầu điều khiển đến robot thông qua giao thức ROS, cho phép điều khiển và tương tác với robot từ xa.



JSON (JavaScript Object Notation): JSON là một định dạng dữ liệu đơn giản dựa trên văn bản, được sử dụng để truyền tải và lưu trữ dữ liệu. Nó dựa trên cú pháp của JavaScript, nhưng đã trở thành một định dạng phổ biến và độc lập ngôn ngữ trong nhiều ngữ cảnh. JSON sử dụng cặp cấu trúc key-value (khóa-giá trị) để

biểu diễn dữ liệu và hỗ trợ các kiểu dữ liệu cơ bản như chuỗi (string), số (number), đối tượng (object), mảng (array), boolean và null. JSON rất phổ biến trong giao tiếp dữ liệu giữa máy chủ và client trong ứng dụng web.

```
package.json x
1 {
2   "name": "myapplication",
3   "description": "some description here",
4   "version": "0.0.1",
5   "private": true,
6   "scripts": {
7     "start": "node ./bin/www"
8   },
9   "dependencies": {
10    "express": "~4.12.2",
11    "jade": "~1.9.2"
12  }
13 }
```

hình 5.12 Ví dụ về dữ liệu JSON

WebSocket: WebSocket là một giao thức truyền thông hai chiều (full-duplex) trên giao thức HTTP, cho phép truyền thông tin liên tục và thời gian thực giữa máy chủ và trình duyệt web. WebSocket giúp xác lập một kết nối liên tục giữa máy chủ và trình duyệt, cho phép hai bên truyền tải dữ liệu một cách song song và không liên quan đến yêu cầu-trả lời truyền thống của giao thức HTTP. Điều này giúp truyền tải dữ liệu nhanh hơn và tiết kiệm tài nguyên mạng hơn so với các phương pháp giao tiếp trước đó. WebSocket thường được sử dụng trong các ứng dụng yêu cầu truyền thông liên tục và thời gian thực như trò chuyện trực tuyến, trò chơi trực tuyến, và ứng dụng web thời gian thực khác.

THUỘC TÍNH	MÔ TẢ
readyState	Diễn tả trạng thái kết nối. Nó có các giá trị sau: - Giá trị 0: kết nối vẫn chưa được thiết lập (WebSocket.CONNECTING) - Giá trị 1: kết nối đã thiết lập và có thể giao tiếp (WebSocket.OPEN) - Giá trị 2: kết nối đang qua handshake đóng (WebSocket.CLOSING) - Giá trị 3: kết nối đã được đóng (WebSocket.CLOSED)
bufferedAmount	Biểu diễn số byte của UTF-8 mà đã được xếp hàng bởi sử dụng phương thức send()

server

c) Thư viện rosjs

Thư viện rosjs là một thư viện JavaScript được sử dụng để kết nối và giao tiếp với ROS (Robot Operating System) trong môi trường trình duyệt web. Nó cung cấp các chức năng để truyền tải dữ liệu, đăng ký và nhận các chủ đề ROS, gọi các dịch vụ ROS và tương tác với ROS thông qua giao thức WebSocket.

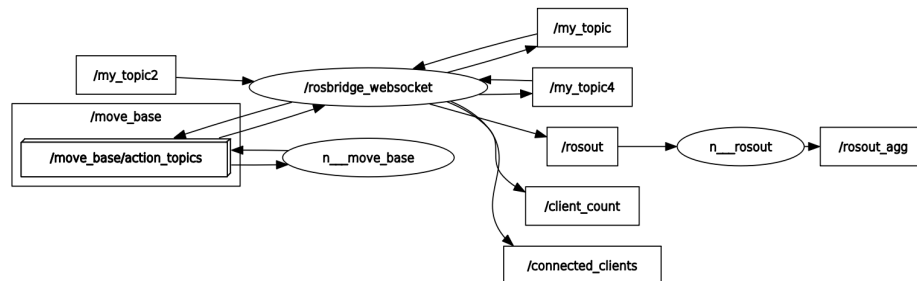
Rosjs cho phép ứng dụng web được viết bằng JavaScript tương tác với hệ thống ROS thông qua một giao diện đơn giản và dễ sử dụng. Nó cung cấp các lớp và phương thức để truyền tải dữ liệu từ các chủ đề ROS (topics) và gọi các dịch vụ ROS (services).

Một số phương thức Rosjs sử dụng để tương tác với ROS:

- + **ROS Object:** Đây là cấu trúc cơ bản để tạo và quản lý kết nối với ROS. Bạn có thể sử dụng ROS object để thiết lập kết nối với máy chủ ROSBridge, đăng ký các chủ đề (topics), gọi các dịch vụ (services) và thực hiện các hoạt động khác liên quan đến ROS.
- + **Topic Object:** Đối tượng Topic trong rosjs cho phép bạn đăng ký và nhận dữ liệu từ các chủ đề ROS. Bạn có thể đăng ký để nhận các cập nhật từ chủ đề và gửi tin nhắn đến chủ đề.
- + **Service Object:** Đối tượng Service trong rosjs cho phép bạn gọi các dịch vụ ROS. Bạn có thể tạo một Service object để gửi yêu cầu dịch vụ và nhận phản hồi từ dịch vụ.
- + **Action Object:** được sử dụng để tương tác với hành động (action) trong ROS.

5.3 Kết nối và điều khiển Robot từ giao diện web.

a) Giao tiếp với Rosbridge:



Giao tiếp giữa web và robot qua Rosbridge thông qua các topic:

- + **/mytopic:** chứa dữ liệu order của khách hàng bao gồm: số bàn, món ăn, giá tiền xuất bản từ trang gọi món của khách. Trang quản lý sẽ đăng kí topic và hiển thị lên bảng.
- + **/mytopic4:** chứa dữ liệu danh sách bàn ăn để robot di chuyển tới xuất bản bởi trang quản lý và đăng kí bởi trang nhân viên.
- + **/mytopic2:** chứa dữ liệu báo khi robot giao xong tất cả các bàn.
- + **/move_base/action_topics:** ra lệnh cho robot di chuyển tới tọa độ của các bàn.

+ Tại trang gọi món: khách hàng chọn bàn, chọn món ăn và nhấn nút gọi món

MỜI QUÝ KHÁCH ORDER:

CHỌN BÀN



MENU



Cafe Capuchino
30.000đ



Baileys milk coffee
35.000đ



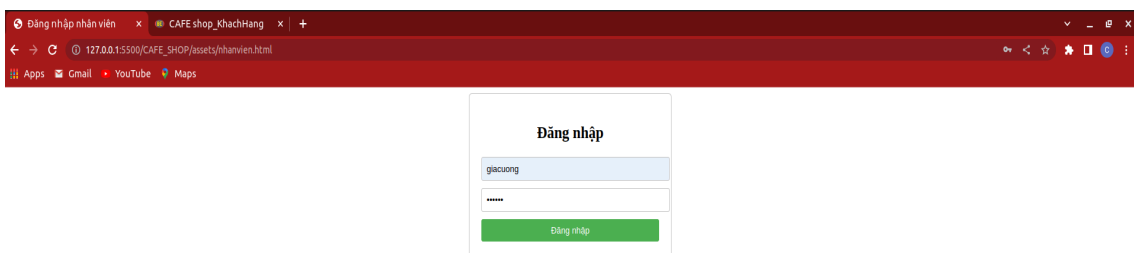
Caramel latte
40.000đ



cepresso
36.000đ

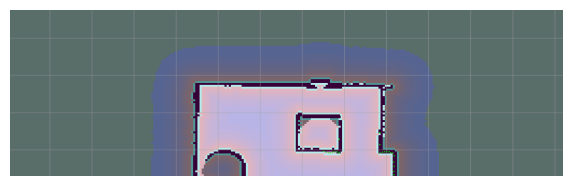
+ Sau khi gọi món danh sách món ăn sẽ được hiển thị tại trang của quản lý: quản lý đăng nhập và xem danh sách món khách gọi.

+ Sau khi nhân viên ấn xác nhận danh sách bàn ăn sẽ được lưu lại, nhân viên ra lệnh giao đồ thì Robot sẽ giao lần lượt tới các bàn đã xác nhận món ăn. Khi tới một bàn khách xác nhận lấy đồ ăn thì xong Robot mới giao tới bàn tiếp.



Tên bàn	Món ăn	Tổng giá	Xác nhận
Bàn 1	Cafe Capuchino, Baileys milk coffee	65000	XÁC NHẬN
Bàn 2	Irish, Frappe	90000	XÁC NHẬN
Bàn 3	Raspberry mocha	50000	XÁC NHẬN
Bàn 4	Irish, Frappe	90000	XÁC NHẬN

GIAO ĐỒ

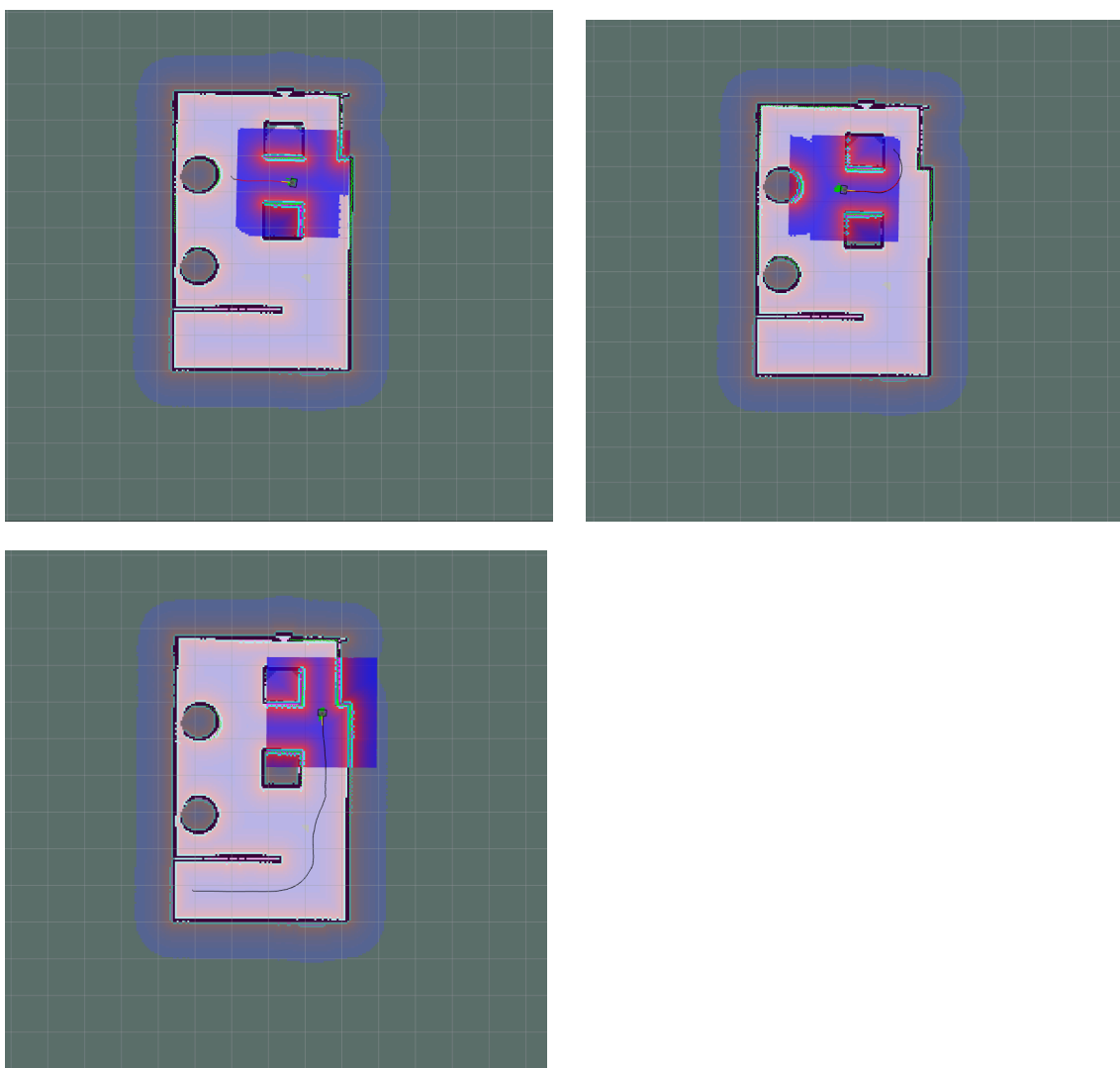


Hình STYLEREF 1 \s 5. SEQ hình * ARABIC \s 1 19 Mô phỏng kết quả giao đồ tới từng bàn.

5.4 Thiết kế giao diện trên robot

5.4.1 Giới thiệu

Giao diện trên robot là phương thức để tương tác giữa con người và robot. Giao diện trên robot là bộ mặt của máy tính. Nó là nơi người sử dụng nhập thông



tin vào hệ thống máy tính (đầu vào) và nhận thông tin phản hồi từ máy tính (đầu ra). Giao diện người dùng nói chung bao gồm người sử dụng, hệ thống, đầu vào và đầu ra.

5.4.2 Nguyên lý hoạt động

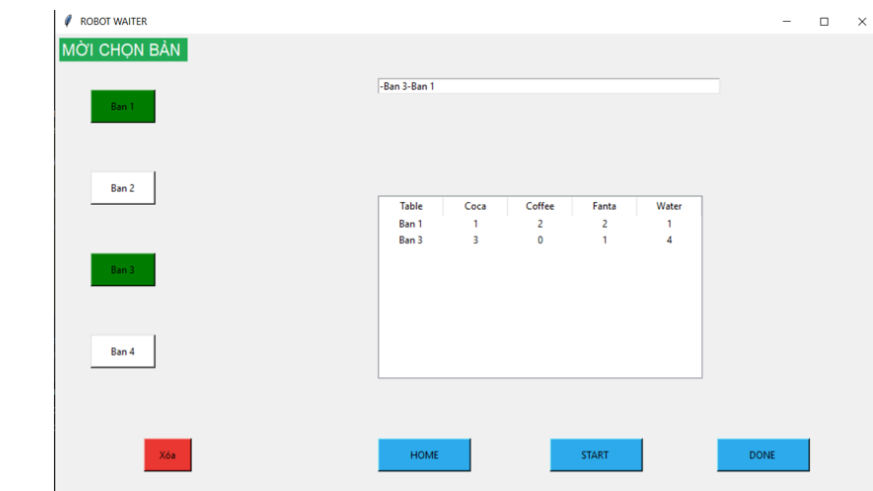
Giao diện robot hoạt động như sau:

- + Khi khách hàng đến và ngồi vào bàn, nhân viên sẽ dựa theo số bàn và chọn bàn từ 1-4. Sau đó nhân viên ấn nút Start, Robot sẽ di chuyển đến từng bàn một theo thứ tự.
- + Khi robot di chuyển đến vị trí bàn thứ #, màn hình sẽ chuyển sang giao diện khách hàng, loa của robot sẽ thông báo để khách hàng chọn đồ uống. Khách hàng chọn đồ uống, sau đó ấn Done để tính toán giá tiền. Nếu khách hàng cảm thấy không hợp lí sẽ chọn lại và ấn Order để kết thúc chọn đồ uống.
- + Robot di chuyển hết các bàn, sau đó quay về vị trí HOME. Nhân viên sẽ dựa theo bảng danh sách đồ uống trên giao diện nhân viên để đặt đồ uống lên từng khay, sau đó ấn Done, Robot sẽ giao đồ uống đến từng bàn và thông báo cho khách lấy đồ uống. Sau khi khách ấn Done, robot sẽ di chuyển sang bàn khác cho đến khi hết bàn, robot sẽ quay về vị trí HOME.

5.4.3 Kết quả

Trong đồ án này sử dụng ngôn ngữ lập trình python kết hợp với thư viện Tkinter để thiết kế giao diện. Tkinter là thư viện GUI tiêu chuẩn cho Python. Khi kết hợp với Tkinter, Python sẽ được cung cấp các công cụ một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng mạnh mẽ đến các bộ công cụ Tk GUI.

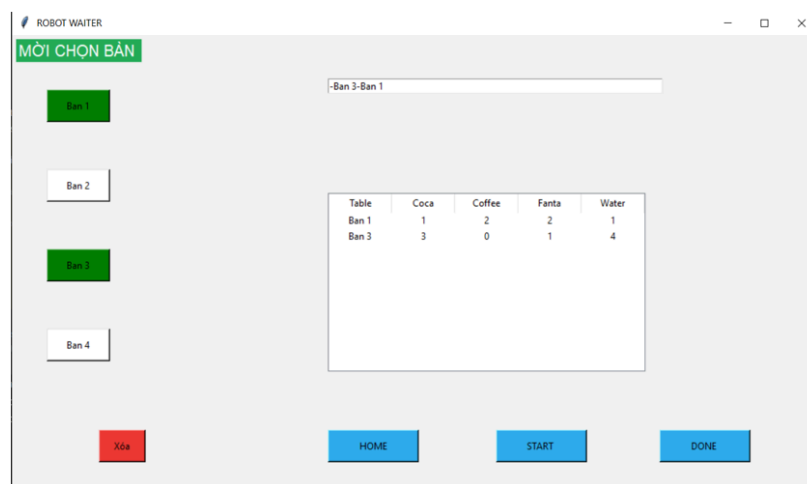
Giao diện người dùng sẽ bao gồm 2 giao diện chính là giao diện dành cho nhân viên và giao diện dành cho khách hàng:



Trước hết chúng ta sẽ sử dụng SLAM để tạo bản đồ và xác định các vị trí HOME và vị trí của các bàn để lưu vào giao diện.

Đầu tiên, nhân viên sẽ đặt robot về vị trí HOME – vị trí sạc pin của robot. Nhân viên sẽ ấn HOME để xác định vị trí ban đầu giúp robot. Sau khi có khách, nhân viên sẽ ấn vào các bàn mà có khách, sau đó ấn START, robot sẽ di chuyển đến từng bàn theo thứ tự. Khi robot đến bàn # sẽ chuyển sang giao diện khách hàng và loa sẽ phát âm thanh 00001.mp3 để thông báo cho khách chọn đồ uống. Khách hàng sẽ chọn đồ uống và ấn Done! để tính giá tiền, nếu không hợp lý thì có thể chọn lại đồ uống và ấn Done lại. Sau đó khách sẽ ấn Order! Robot sẽ chuyển sang bàn tiếp theo và cứ lặp lại cho đến khi hết bàn, robot sẽ quay về vị trí HOME.

Khi đó, trên màn hình sẽ hiện giao diện nhân viên với một bảng hiện các loại đồ uống cùng số bàn. Nhân viên chuẩn bị đồ uống xong sẽ ấn DONE để robot mang đồ uống đến từng bàn.



Khi đó robot sẽ phát âm thanh 00002.mp3 để thông báo khách hàng nhận đồ uống. Khách nhận xong ấn DONE, robot sẽ chuyển qua bàn tiếp theo đến hết bàn thì quay về HOME.

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Với đề tài “Thiết kế Robot tự hành trong ngành dịch vụ-Autonomous service robot”, nhóm đã đạt được những kết quả nhất định:

- + Ghép nối, kết hợp điều khiển các module, linh kiện điện tử hoạt động ổn định.

- + Tìm hiểu và ứng dụng được thuật toán điều khiển điều hướng cho robot trên nền tảng hệ điều ROS.
- + Ứng dụng được RPLidar và Jetson Nano vào trong quá trình xây dựng bản đồ, định vị cho robot trong quá trình di chuyển.
- + Ứng dụng ngôn ngữ lập trình python và thư viện tkinter trong việc thiết lập giao diện điều khiển người dùng và giao tiếp với các chân IO trên Jetson Nano để phát âm thanh.
- + Ứng dụng các ngôn ngữ lập trình web, app xây dựng giao diện điều khiển cho khách hàng và nhân viên.
- + Ứng dụng được công nghệ xử lý ảnh dựa trên mạng YOLO trong việc phát hiện đối tượng và nhận diện các loại đồ uống.
- + Ứng dụng trí tuệ nhân tạo điều khiển robot bằng cử chỉ tay.
- + Ứng dụng được các kiến thức đã học vào trong quá trình xây dựng và thiết lập các thuật toán cho đề tài.

Tuy nhiên do kiến thức còn nhiều thiếu sót và thời gian hạn chế nên còn tồn tại một số nhược điểm:

- + Khả năng tương tác giữa robot và con người còn thấp.
- + Vẫn còn sai lệch về hướng và vị trí của robot.
- + Xử lý ảnh và nhận dạng cử chỉ tay điều khiển còn có độ trễ.

6.2 Hướng phát triển tương lai của đề tài

- + Tăng tốc độ di chuyển của robot, nâng cao số lượng đồ ăn phục vụ.
- + Thiết kế thêm bộ sạc tự động sau mỗi phiên làm việc.
- + Tăng độ chính xác của robot bằng cách sử dụng các cảm biến như IMU, GPS
- + Xây dựng hệ thống hoàn toàn tự động gồm nhiều robot phục vụ để giảm thiểu tối đa nhân lực.
- + Phát triển thêm các tính năng cho web, app tích hợp CSDL, server.
- + Thay đổi phần cứng máy tính nhúng, tăng cấu hình tốc độ xử lý để hạn chế độ trễ cho robot.

TÀI LIỆU THAM KHẢO

- [1] Model Cards for Model Reporting (arxiv.org)
- [2] Hand landmarks detection guide | MediaPipe | Google for Developers
- [3] Model Card Hand Tracking (Lite/Full) with Fairness Oct 2021
(storage.googleapis.com)
- [4]
refer: http://www.isrr-2011.org/ISRR-2011/Program_files/Papers/Jenkins-I SRR-2011.pdf
- [5] <https://www.mdpi.com/1424-8220/21/4/1369>
- [6] <https://arxiv.org/abs/1512.02325>
- [7] <http://wiki.ros.org/ROS>
- [8] Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments | IEEE Journals & Magazine | IEEE Xplore

PHỤ LỤC

1. Yêu cầu về các hệ điều hành cho các thiết bị:

- Máy tính chủ: yêu cầu hệ điều hành Ubuntu 20.04 LTS
- Jetson Nano: yêu cầu hệ điều hành Ubuntu 20.04 LTS

2. Hướng dẫn cài đặt hệ điều hành ROS trên Ubuntu:

Đồ án sử dụng ROS phiên bản Noetic. Các bước cài đặt ROS trên các thiết bị sử dụng hệ điều hành Ubuntu là giống nhau và được trình bày sau đây:

Bước 1: Cài đặt sources.list cho thiết bị để thiết bị có thể chấp nhận phần mềm từ packages.ros.org:

```
▪ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

Bước 2: Cài đặt keys cho thiết bị:

```
▪ sudo apt install curl # if you haven't already installed curl
▪ curl -s
https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc |
sudo apt-key add -
```

Bước 3: Cài đặt ROS:

Trước hết, cần đảm bảo chỉ mục gói Debian là mới nhất:

```
▪ sudo apt update
```

Đối với Jetson Nano, nên cài đặt phiên bản ROS desktop:

```
▪ sudo apt install ros-noetic-desktop
```

Đối với Máy tính chủ, nên cài đặt phiên bản ROS desktop full (tích hợp thêm các gói mô phỏng 2D/3D, các gói perception 2D/3D):

```
▪ sudo apt install ros-noetic-desktop-full
```

Bước 4: Cài đặt môi trường:

Mỗi khi sử dụng ROS trong terminal, cần dẫn nguồn script sau /opt/ros/Noetic/setup.bash. Để mỗi terminal khi được mở lên tự động dẫn nguồn script đó, cần thực hiện các lệnh sau:

```
▪ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
▪ source ~/.bashrc
```

Bước 5: Cài đặt các phần phụ thuộc để xây dựng các gói:

```
▪ sudo apt install python-rosdep python-rosinstall  
python-rosinstall-generator python-wstool build-essential
```

```
▪ sudo apt install python-rosdep  
▪ sudo rosdep init  
▪ rosdep update
```

Bước 6: Khởi tạo Không gian làm việc cho ROS:

```
▪ mkdir -p ~/catkin_ws/src  
▪ cd ~/catkin_ws/  
▪ catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3  
▪ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc  
▪ source ~/.bashrc
```

3. Kết nối máy tính với Jetson Nano

- Trên máy chủ, tìm ROS_IP và ROSMASTER_URI là thông tin của cả hai máy cần liên lạc. Tìm ROS_IP (là địa chỉ IP của máy tính chính) bằng cách chạy lệnh ifconfig.

- Trên Robot

- *Set ROS_IP is the machine IP address.*
- *ROS_MASTER_URI is the robot's IP*

- Trên máy tính

- *Set ROS_IP is its own IP.*
- *SET ROS_MASTER_URI is the robot's IP*

- Ví dụ: Trong ví dụ này (IP có thể sẽ khác nhau trong mạng của bạn),

- Trên robot, ta đặt: `export ROS_IP=192.168.43.228, export ROS_MASTER_URI=http://192.168.43.228:11311.`

- Trên máy tính ta đặt: `export ROS_IP=192.168.43.123, export ROS_MASTER_URI=http://ubiquityrobot.local:11311.`

4. Kiểm tra Lidar

Để kiểm tra Lidar, ta chạy lần lượt các dòng lên sau trên terminal:

- `cd catkin_ws/src.`
- `git clone https://github.com/EAIROBOT/ydlidar_ros.git.`
- `catkin_make`
- `roscd ydlidar_ros/startup`

- `sudo chmod 777 ./*`
- `sudo sh initenv.sh`
- Quay lại môi trường `catkin_ws`, và chạy `source devel/setup.bash`.
- `git checkout G2`. Lệnh này sẽ chuyển bạn đến nhánh của mô hình LiDAR cụ thể của bạn.
- Chạy lại `catkin_make`.

5. Thiết lập bản đồ sử dụng Hector SLAM

Bước 1: Kết nối Lidar với Jetson Nano

Để kiểm tra ta chạy lệnh:

```
ls -l /dev | grep ttyUSB
```

Bước 2: Set Up Catkin Workspace và Cài đặt phần phụ thuộc cần thiết

```
sudo apt-get update
sudo apt-get install cmake python-catkin-pkg python-empy
python-nose python-setuptools libgtest-dev python-rosinstall
python-rosinstall-generator python-wstool build-essential git
```

Tạo catkin workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
gedit ~/.bashrc # dùng để chỉnh sửa .bashrc text file
Phải chắc chắn 2 dòng code này phải nằm dưới cùng của .bashrc file.
```

```
source /opt/ros/melodic/setup.bash
```

```
source ~/catkin_ws/devel/setup.bash
```

Sau đó chạy lệnh: `source ~/catkin_ws/devel/setup.bash`

Bước 3: Cài đặt RPLIDAR ROS Packages

Clone the RPLIDAR ROS package vào tệp src.

```
sudo git clone https://github.com/Slamtec/rplidar_ros.git
cd ~/catkin_ws/
catkin_make
```

Launch the RPLIDAR launch file, Mở terminal và chạy lệnh sau:

```
roslaunch rplidar_ros rplidar.launch
```

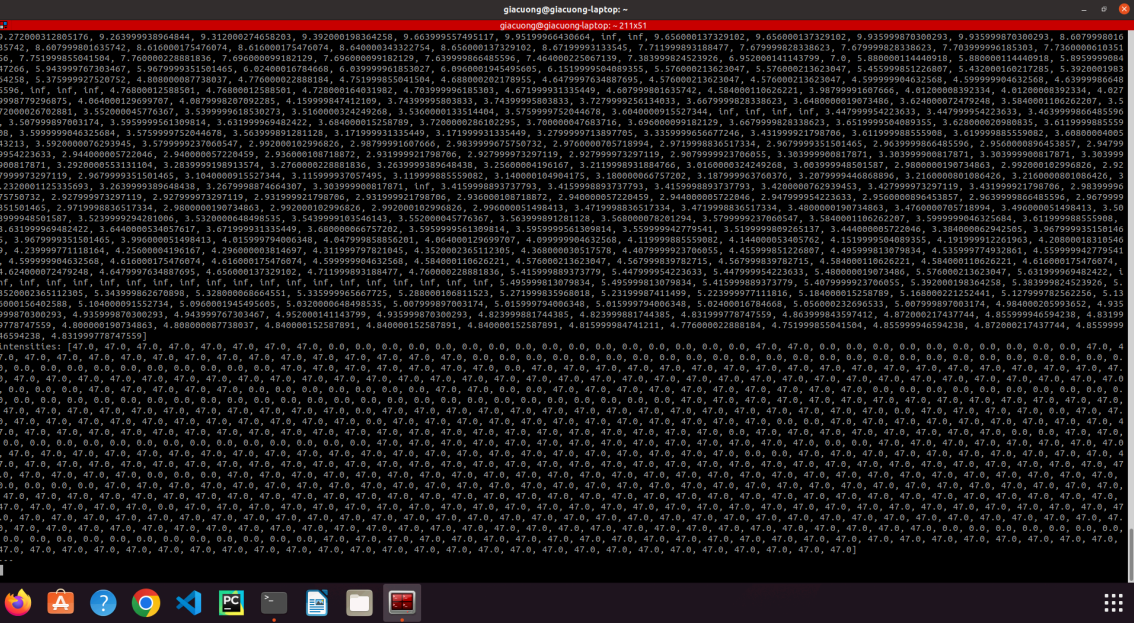
Kiểm tra topic, mở mới và chạy lệnh sau:

- rostopic list

Bắt đầu quá trình scan:

- rostopic echo /scan

Kết quả nhận được:

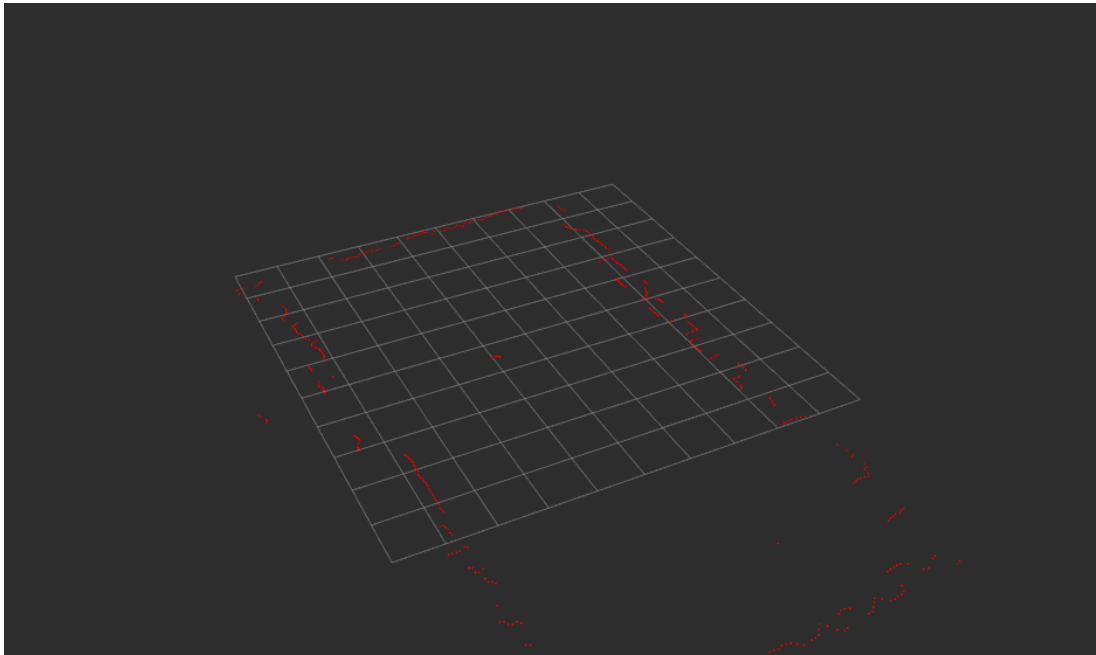


Kiểm tra kết quả scan trên rviz:

- roslaunch rplidar_ros rplidar.launch

Mở terminal mới và chạy: rviz

Kết quả scan trên rviz



Bước 4: Thiết lập bản đồ

Download the Hector-SLAM Package:

```
▪ cd ~/catkin_ws/src
▪ git clone
  https://github.com/tu-darmstadt-ros-pkg/hector_slam.git
```

Set the Coordinate Frame Parameters:

```
▪ sudo gedit
  ~/catkin_ws/src/hector_slam/hector_mapping/launch/mapping_defau
  lt.launch
```

Tìm kiếm các dòng code sau ở dòng thứ 5 và 6:

```
<arg name="base_frame" default="base_footprint"/>
```

```
<arg name="odom_frame" default="nav"/>
```

Và đổi thành:

```
<arg name="base_frame" default="base_link"/>
```

```
<arg name="odom_frame" default="base_link"/>
```

Sau đó vào dòng cuối của code (dòng thứ 54) đổi các code sau:

```
<!--<node          pkg="tf"          type="static_transform_publisher"
name="map_nav_broadcaster" args="0 0 0 0 0 0 map nav 100"/>-->
```

Thành:

```
<node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0 0 0 0 0 0 base_link laser 100"/>
```

Lưu lại và mở terminal mới và chạy:

```
▪ cd ~/catkin_ws/src/hector_slam/hector_slam_launch/launch
```

Mở tệp tutorial.launch:

```
▪ gedit tutorial.launch
```

Tìm và đổi dòng code thứ 7: `<param name="/use_sim_time" value="true"/>` thành `<param name="/use_sim_time" value="false"/>`.

Lưu lại và chạy:

```
▪ cd ~/catkin_ws/
▪ catkin_make
```

Khởi động lại Jetson Nano và chạy các lệnh sau:

```
▪ cd ~/catkin_ws/
▪ sudo chmod 666 /dev/ttyUSB0
▪ roslaunch rplidar_ros rplidar.launch
# Mở terminal mới và chạy:
▪ roslaunch hector_slam_launch tutorial.launch
```

Bước 5: Lưu bản đồ

```
▪ sudo apt-get install ros-melodic-map-server
▪ mkdir ~/catkin_ws/maps
▪ cd ~/catkin_ws/maps
▪ rosrun map_server map_saver -f my_map
```

6. Chạy Navigation và giao diện người dùng, giao diện trên web, app.

Bước 1: Kết nối SSH tới Jetson Nano

```
▪ ssh 192.168.1.136
```

Bước 2: Chạy các gói trên jetson nano để giao tiếp với cảm biến và rplidar

```
▪ roscore
▪ cd my_ws
```

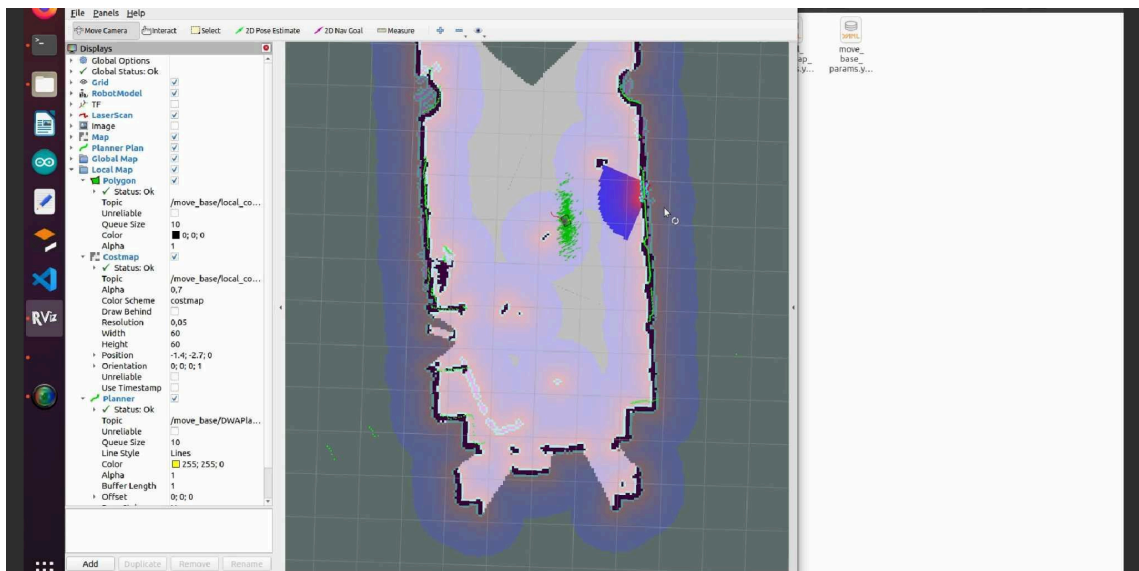
```
▪ source ./devel/setup.bash
```

```
▪ roslaunch node_example nano.launch
```

Bước 3: Trên PC chạy các gói và xác định vị trí của robot trên bản đồ

```
▪ cd my_ws  
▪ source ./devel/setup.bash  
▪ roslaunch navigation navigationjetsonbot.launch
```

- chọn vị trí robot trên bản đồ chọn 2D Pose Estimate và click vị trí trên bản đồ
Kết quả nhận được:



Bước 4: Set vị trí bàn ăn tại bản đồ

```
▪ rostopic echo /clicked_point
```

Click point trên bản đồ và lưu vị trí thấy được vào file giaodien_ros.py

Bước 5: Chạy giao diện người dùng

```
▪ cd my_ws  
▪ source ./devel/setup.bash  
▪ python3 main2.py
```

Bước 6: Chọn bàn và thao tác trên giao diện xe sẽ di chuyển tới đúng vị trí bàn đã chọn và có những giao tiếp với khách hàng. Khi hết bàn xe sẽ tự di chuyển về vị trí home.

Bước 7: Kết nối tới Rosbridge

```
▪ roslaunch rosbridge_server rosbridge_websocket.launch
```

Bước 8: Chạy điều khiển robot bằng cử chỉ tay

```
▪ roslaunch ros_hand_gesture_recognition sign_control.launch
```

Bước 9: Chạy web trên server local và chọn bàn, gọi món, nhân viên đăng nhập xác nhận để robot giao đồ ăn tới các bàn. Tới bàn ăn khách hàng có thể giao tiếp với robot qua cử chỉ tay, điều khiển robot, chọn nhận món ăn và xác nhận trên web hoặc app. Khi giao hết bàn robot sẽ tự động về quầy và thông báo trên web, app.