ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

----------o0o----------

ĐỒ ÁN 2

ỨNG DỤNG THUẬT TOÁN SCAN MATCHING ĐỂ XÂY DỰNG BẢN ĐỒ VÀ ĐIỀU HƯỚNG CHO ROBOT TỰ HÀNH TRONG NHÀ

GVHD: TS. NGUYỄN VĨNH HẢO

NGUYỄN VÕ HỒNG MỸ HIỀN MSSV: 2010260

DƯƠNG NGỌC HOÀN MSSV: 2010020

TP.HỒ CHÍ MINH, THÁNG 1 NĂM 2024

TRƯỜNG ĐẠI HỌC BÁCH KHOA
TP. HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

BỘ MÔN: ĐIỀU KHIỂN TỰ ĐỘNG

TP. HCM, ngày 30 tháng 12 năm 2023

ĐỀ CƯƠNG CHI TIẾT

TÊN LUẬN VĂN:
Cán bộ hướng dẫn: TS. NGUYỄN VĨNH HẢO
Thời gian thực hiện: Từ ngày 30/8 đến ngày 30/12
Sinh viên thực hiện: **Nguyễn Võ Hồng Mỹ Hiền - 2010260** **Dương Ngọc Hoàn - 2010020**
Nội dung đề tài: - Thiết kế mô hình phần cứng xe tự hành cho môi trường trong nhà. - Tìm hiểu thuật toán Scan Matching. - Dùng cảm biến lidar 2D và thuật toán Scan Matching để xây dựng bản đồ và điều hướng cho robot.
Kế hoạch thực hiện: Các công việc của nhóm được hai thành viên phân chia như sau:

Giai đoạn	Dương Ngọc Hoàn	Nguyễn Võ Hồng Mỹ Hiền
Giai đoạn 1	- Vẽ mô hình robot trên phần mềm SolidWorks và đặt cắt mica. - Xây dựng mô hình mô phỏng trên ROS2.	- Thiết kế mạch driver cho 4 DC motor. - Thiết kết mạch điều khiển chính. - Tính toán vấn đề công suất.
Giai đoạn 2	- Hàn mạch và lắp ráp robot.	- Hàn mạch và lắp ráp robot. - Test các module phần cứng.
Giai đoạn 3	- Tìm hiểu giao tiếp uROS. - Lập trình Firmware uROS giao tiếp giữ Jetson Nano và Raspberry Pico.	- Tìm hiểu mối liên hệ vận tốc bánh xe và vận tốc của robot. - Lập trình Firmware đọc cảm biến encoder và điều khiển PID cho động cơ.
Giai đoạn 5	- Tìm hiểu thuật toán Iterative Closets Point (IPC). - Sử dụng Lidar và điều khiển robot để mapping không gian.	- Lập trình Firmware đọc adc giám sát pin của robot. - Lập trình Firmware module scheduler.
Giai đoạn 6	- Tiến hành viết báo cáo.	- Tiến hành viết báo cáo.

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn tới Ban giám hiệu nhà trường, đoàn hội, khoa, tập thể giảng viên và công nhân viên chức nhà trường Đại học Bách Khoa – Đại học Quốc gia TP.HCM đã xây dựng cho chúng em một môi trường học tập và phát triển tốt. Đây là nền tảng cơ sở vững chắc cho chúng em có tiếp cận và rèn luyện được nhiều kỹ năng, kiến thức mới.

Tiếp theo, chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Vĩnh Hảo, người đã tận tình hướng dẫn chúng em thực hiện luận văn. Thầy không chỉ là điểm tựa tinh thần, giải đáp các vấn đề chuyên ngành và đưa ra những lời khuyên hết sức quý giá, mà còn tạo điều kiện vật chất, môi trường làm việc nghiên cứu thuận lợi nhất giúp nhóm hoàn thành luận văn với những mục tiêu đã đề ra.

Chúng em cũng chân thành cảm ơn các thầy cô Khoa Điện-Điện Tử, đặc biệt là những thầy cô trong bộ môn Điều Khiển và Tự Động Hóa. Các thầy cô luôn hết sức giảng dạy cho sinh viên một cách chu đáo, truyền đạt kiến thức nền tảng đến chuyên sâu cũng như những kinh nghiệm thực tế của bản thân, góp phần giúp chúng em nắm vững và áp dụng kiến thức vào luận văn, tạo động lực học tập nghiên cứu lâu dài.

Sau cùng, nhóm gửi lời cảm ơn đến các bạn cùng chuyên ngành, đặc biệt là các anh chị, các bạn ở phòng thí nghiệm 207B3 đã nhiệt tình giúp đỡ và đóng góp ý kiến để nhóm hoàn thành luận văn một cách hoàn thiện.

MỤC LỤC

DANH MỤC HÌNH ẢNH

TÓM TẮT ĐỒ ÁN 2

Đề tài này thực hiện thiết kế thi công phần cứng và thiết kế xây dựng phần mềm cho robot tự hành môi trường trong nhà. Robot thực hiện các tác vụ như: vẽ bản đồ, định vị. Bộ não của robot là một máy tính nhúng Jetson Nano, các cảm biến được sử dụng cho robot là Lidar 2D, IMU và Encoder. Dựa vào các dữ liệu cảm biến Lidar mà bản đồ 2D có thể được xây dựng bởi thuật toán RTAB-Map, từ đó thực hiện định vị và điều hướng cho robot (sẽ phát triển trong luận văn). Máy tính nhúng giao tiếp với bảng mạch nhúng có vi điều khiển Raspberry Pico để gửi tín hiệu điều khiển robot. Dữ liệu bản đồ vẽ trên máy tính nhúng có thể được hiển thị trên PC bất kỳ có cài đặt hệ điều hành ROS2 và phần mềm RVIZ thông qua chuẩn UDP không dây. UDP cũng giúp gửi các lệnh điều khiển trên PC xuống Jetson để hỗ trợ quá trình vận hành robot.

Odometry sử dụng cho việc vẽ bản đồ được lấy từ một trong hai nguồn cảm biến: Lidar hoặc IMU và Encoder.

CHƯƠNG 1. GIỚI THIỆU

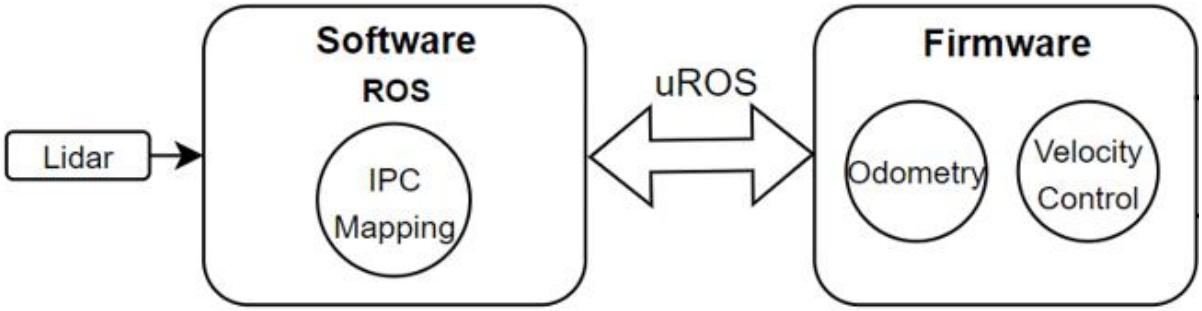
1.1. Tổng quan đề tài nghiên cứu

Hiện nay, công nghệ liên quan đến robot di động (Mobile Robot) như phương tiện tự hành (autonomous vehicles) hoặc robot phục vụ trong nhà đang được đưa rất nhiều vào trong nghiên cứu. Một thành phần thiết yếu trong ứng dụng robot là hệ thống điều hướng (navigation), nó giúp robot nhận dạng và tái tạo môi trường để việc di chuyển trở nên hiệu quả hơn. Một robot tự hành cần đáp ứng được các yêu cầu tất yếu: thứ nhất, cần phải xác định được vị trí của của bản thân trong hệ tọa độ tham chiếu; thứ hai, robot phải lập cho mình một kế hoạch tìm đường và tranh vật cản tự động; cuối cùng là robot phải nhận biết được môi trường xung quanh thông qua các cảm biến (nhận thức). SLAM (Simultaneous Localization and Mapping) là một hệ thống định vị và lập bản đồ trực quan thời gian thực giúp robot hoạt động ưu việt hơn.

Nhiều hệ thống SLAM khác nhau đang được nghiên cứu phát triển, hầu hết chúng sử dụng cảm biến quang, nổi bật trong số đó là Visual SLAM (camera-based) sử dụng thông tin ảnh thu được từ camera và Lidar SLAM (lidar-based) sử dụng thông tin laser từ cảm biến lidar. Với mục tiêu hoạt động ở môi trường trong nhà, đề tài nghiên cứu đồ án 2 được nhóm lựa chọn sử dụng cảm biến Lidar phục vụ nhiệm vụ SLAM thông qua một thuật toán mã nguồn mở giúp xây dựng bản đồ 2D.

Để xây dựng một hệ thống hoàn thiện thì đây là một đề tài khá rộng và cần có những chuyên gia có kiến thức, kinh nghiệm để có thể phát triển một hệ thống ổn định, hoạt động trôi chảy trên nhiều điều kiện môi trường khác nhau. Vì tính ứng dụng rộng lớn và tính phức tạp của vấn đề, đồ án 2 chỉ ở mức tìm hiểu thuật toán và ứng dụng các thuật toán cơ bản để có thể xây dựng được mô hình hoạt động hoàn thiện.

1.2. Mục tiêu đề tài



Hình 1. Tổng quan hệ thống đề tài đồ án

Đồ án 2 với mục tiêu thiết kế phần cứng, điều khiển vận tốc robot và sử dụng Lidar để mapping. Hình 1 thể hiện tổng quan hệ thống cần xây dựng của đề tài. ROS sẽ đọc dữ liệu từ cảm biến lidar và nhận tính hiệu điều khiển từ Firmware, sau đó robot sẽ di chuyển và xây dựng lại không gian 2D xung quanh. Nhóm đặt ra mục tiêu cần hoàn thành như sau:

Thiết kế mô hình phần cứng robot và mạch điều khiển.

Robot nhận tín hiệu điều khiển vận tốc và thực thi đúng với yêu cầu đặt ra.

Xây dựng bản đồ 2D của môi trường trong nhà chưa biết trước bằng cảm biến Lidar.

1.3. Phương pháp thực hiện

Dựa vào môi trường hoạt động của robot là ở trong nhà nên nhóm tự thiết kế và lắp ráp khung robot với động cơ, nguồn, vi điều khiển, cảm biến,…. sao cho khung robot chịu được tải trọng của robot và có hình dáng giúp robot dễ dàng di chuyển trong nhà, phần này được trình bày ở mục 3.1. Quá trình thiết kế khung xe cùng mạch điều khiển và cảm biến được thực hiện thông qua phần mềm hỗ trợ là SolidWorks và Kicad, phần này được trình bày ở mục 3.3 và 3.4. Với nhiệm vụ đặt ra nhóm tiến hành tìm hiểu lí thuyết về các thuật toán liên quan, phần này được trình bày ở chương 2. Việc thiết kế phần mềm cho mạch điều khiển bao gồm điều khiển động cơ DC bằng thuật toán PID và đọc encoder, đồng thời tích hợp giao thức uROS để giao tiếp với nền tảng ROS được trình bày ở mục 3.5. Robot sử

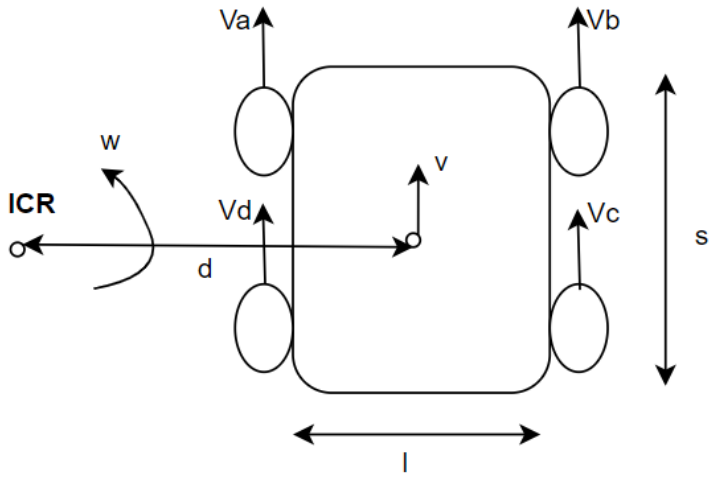
dụng máy tính nhúng Jetson Nano là bộ xử lí trung tâm, hoạt động với hệ điều hành Ubuntu có tích hợp ROS. Dựa trên đó, nhóm tiến hành thiết kế phần mềm xử lí chính trên ROS gồm việc đọc cảm biến Lidar và áp dụng thực toán Iterative Closets Point để xây dựng bản đồ 2D, được trình bày ở mục 3.6. Cuối cùng, nhóm thu thập kết quả tổng kết ở chương 4, từ đó đánh giá quá trình thực hiện, tìm hiểu cách khắc phục và hướng phát triển trong tương lai, được trình bày ở chương 5.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Điều khiển tốc độ động cơ

2.1.1. Mô hình Differential Drive

Khi robot được thiết kế theo mô hình Differential Drive, mối liên hệ giữa vận tốc của từng bánh xe v_l, v_r với vận tốc xoay ω và vận tốc tịnh tiến v của robot có thể được miêu tả rõ ràng bằng công thức.



Hình 2. Mô hình Differential Drive

Với ICR là tâm xoay tức thời khi robot di chuyển, l là khoảng cách giữa 2 bánh xe, s là chiều dài xe, d là khoảng cách từ trọng tâm xe đến ICR, v và ω lần lượt là vận tốc trọng tâm robot và vận tốc quay quay của robot đối với tâm quay tức thời, v_a, v_b, v_c, v_d là vận tốc của các bánh xe được xác định bằng công thức như sau:

$$v_a = v_d = v_{left} = \frac{2v - \omega l}{2}$$

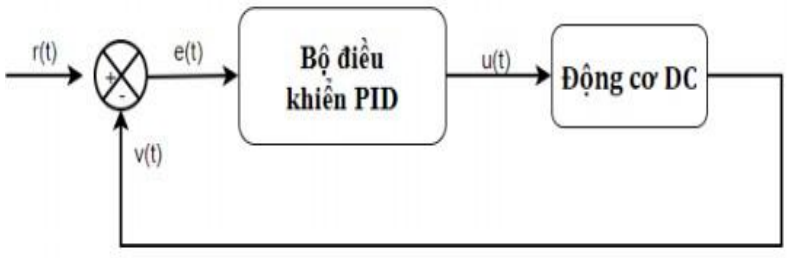
$$v_c = v_b = v_{right} = \frac{2v + \omega l}{2}$$

Phần chứng minh được trình bày ở phần phụ lục.

2.1.2. Bộ điều khiển PID

Bộ điều khiển PID là bộ điều khiển vòng kín có hồi tiếp được sử dụn phổ biến trong việc điều khiển động cơ DC. Với T là chu kì điều khiển, u(k) là tín hiệu điều khiển, r(k) là tín hiệu đặt, v(k) là tín hiệu hồi tiếp, e(k) là sai số giữa tín hiệu đặt và tín hiệu hồi tiếp e(k) = u(k) – r(k) ở chu kì k. Rời rạc hóa ta được phương trình cập nhật tín hiệu điều khiển như sau:

$$u(k) = u(k-1) + K_p(e(k) - e(k-1)) +$$
$$K_i \frac{T}{2}(e(k) - e(k-1)) +$$
$$K_d \frac{1}{T}(e(k) - 2e(k-1) + e(k-2))$$



Hình 3. Sơ đồ khối bộ điều khiển PID

Trong đó Kp, Ki, Kd là các hệ số của 3 khâu (khâu tỉ lệ P, tích phân I và vi phân D) giúp tinh chỉnh bộ điều khiển và có ảnh hưởng lên bộ điều khiển như sau:

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

Hình 4. Ảnh hưởng của các hệ số trong bộ điều khiển PID[1]

[1] Nguồn: Tài liệu thí nghiệm môn Cơ sở Điều khiển tự động, khoa Điện-Điện tử, trường Đại học Bách Khoa Thành phố Hồ Chí Minh.

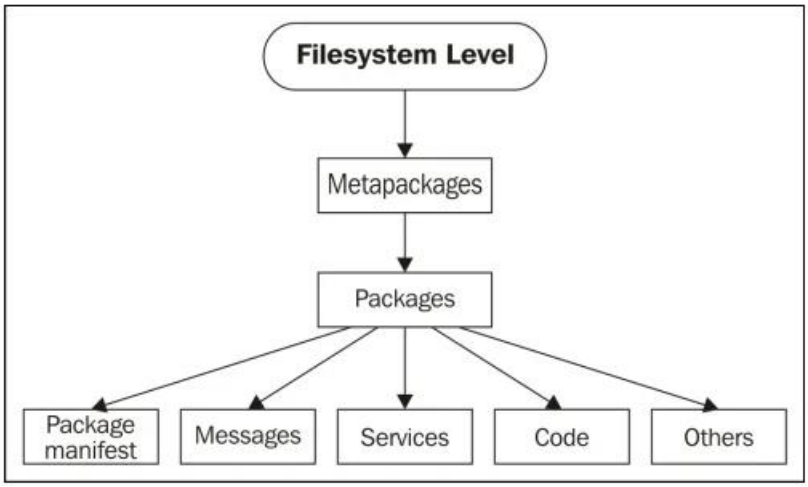
2.2. Nền tảng ROS2

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

Mặc dù chưa hẳn là một hệ điều hành hoàn chỉnh mà chính xác hơn là platform, nhưng ROS2có các yếu tố cấu thành một hệ điều hành: trừu tượng hóa phần cứng, chuyển thông tin giữa các tiến trình, quản lý dữ liệu theo các gói, … Mô hình ROS2 được khái quát như sau.

2.2.1. Tầng ROS File System

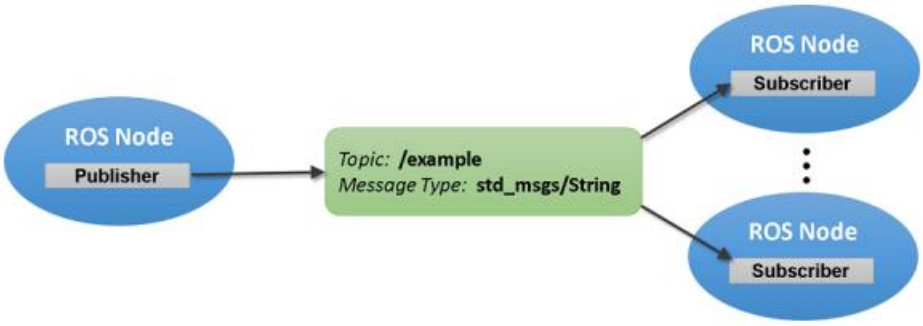
Filesystem chủ yếu là các tài nguyên trên phần cứng, cấu trúc thư mục và các tập tin tối thiểu để ROS hoạt động.



Hình 5 Cấu trúc File System trên ROS

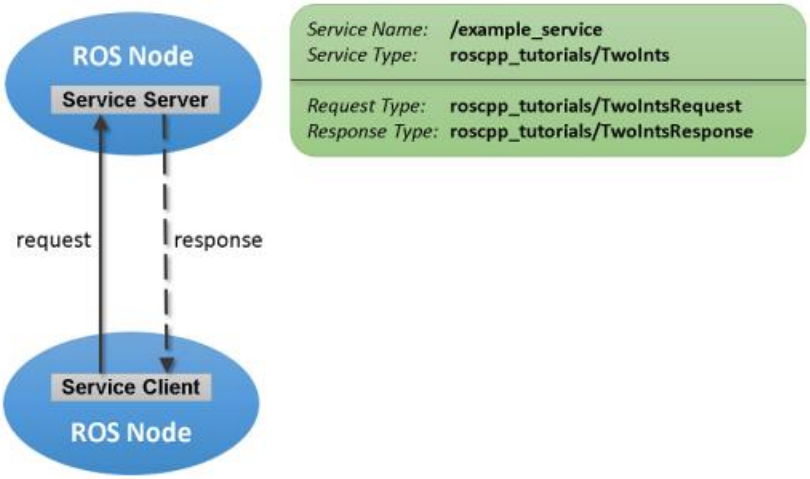
- Packages: ROS Package là thành phần cơ bản nhất trên ROS. Nó bao gồm các node, thư viện, và các tệp tin cấu hình, … được sắp xếp với nhau thành một thành phần duy nhất. Mục đích của gói mã nguồn là tạo ra chương trình tổng quát nhất để dễ dàng sử dụng lại cũng như phát triển.

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



Hình 6 Truyền nhận message trong ROS

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

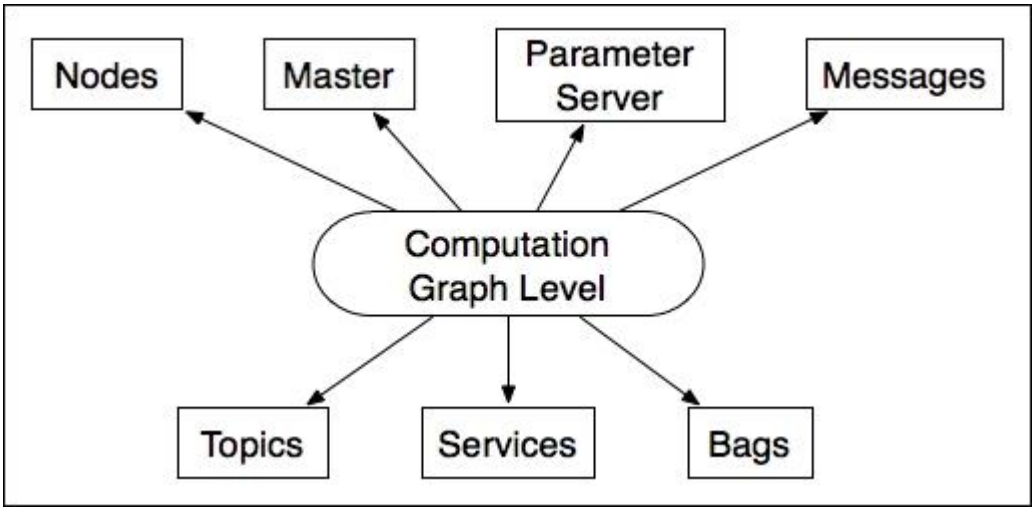


Hình 7 Service trong ROS

2.2.2. ROS2 Graph

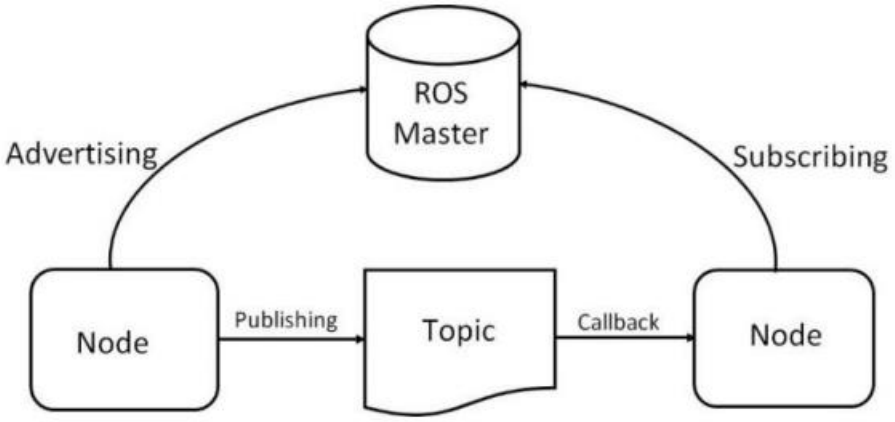
ROS graph là một mạng các phần tử ROS2 xử lý dữ liệu cùng nhau tại một thời điểm, là nơi mà các quy trình trong ROS được kết nối với nhau. Nó bao gồm tất cả các tệp thực

thi (executables) và các kết nối giữa chúng nếu chúng ta muốn sắp xếp và thể hiển thị chúng. Tầng này bao gồm các khái niệm cơ bản sau:



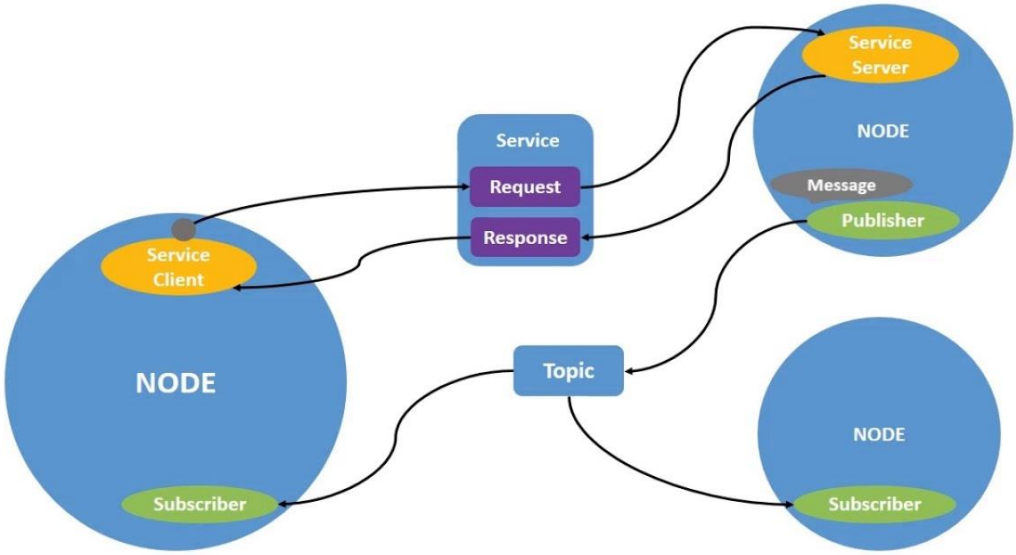
Hình 8 ROS2 graph

- Node: đơn vị thực thi cơ bản nhất. Mỗi node thực hiện một nhiệm vụ cố định là xử lý và truyền/nhận dữ liệu, nhờ vậy các tác vụ lớn, cần nhiều tính toán dễ dàng thực hiện bằng việc liên kết các node lại với nhau.
- Master: đóng vai trò kết nối các node với nhau. Do đó, master luôn được khởi động đầu tiên, sau đó thì bạn có thể gọi bất kỳ các node nào trong hệ thống.



Hình 9 Vai trò của ROS Master

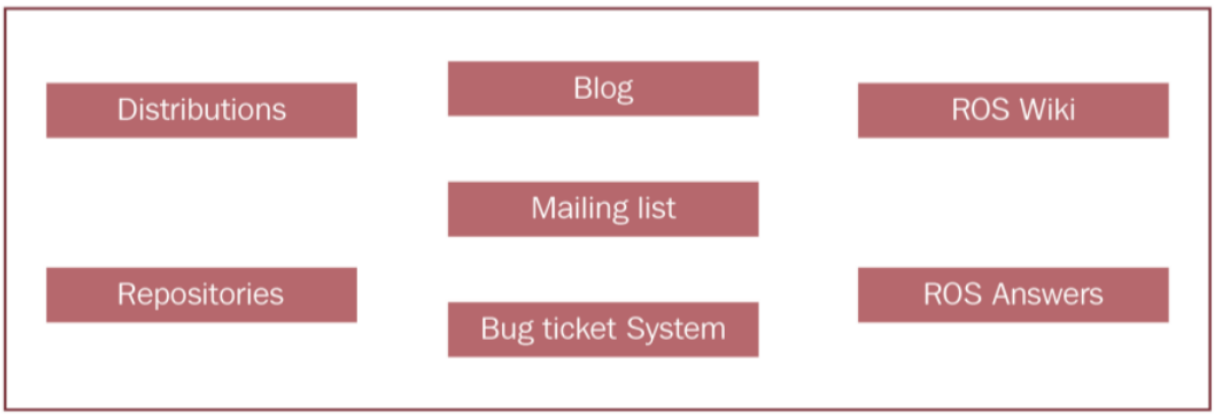
- Message: Đây là cấu trúc dữ liệu được các node sử dụng để trao đổi với nhau tương tự như các kiểu dữ liệu double, int trong các ngôn ngữ lập trình. Các node tương tác với nhau bằng cách send và receive ROS message.
- Topics: là phương pháp giao tiếp trao đổi dữ liệu giữa hai node, nó bao gồm nhiều cấp bậc thông tin mà chúng có thể giao tiếp thông qua ROS message. Hai phương thức trong topic bao gồm publish và subscribe.



Hình 10 Trao đổi dữ liệu giữa các Node qua Topic trên ROS

2.2.3. Tầng ROS Community

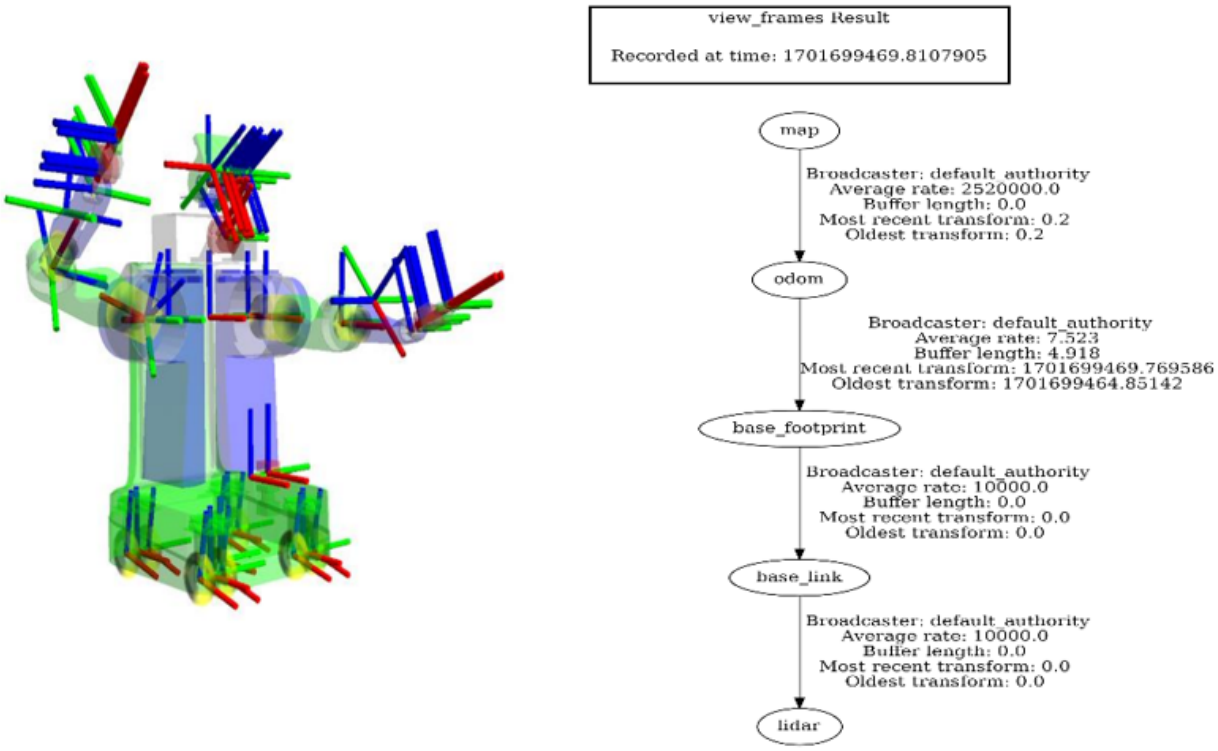
ROS Community là nguồn tài nguyên của ROS được cộng đồng người dùng trao đổi với nhau về phần mềm và kiến thức



Hình 11 Cấu trúc tầng ROS Community

2.2.4. Package TF trong ROS

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

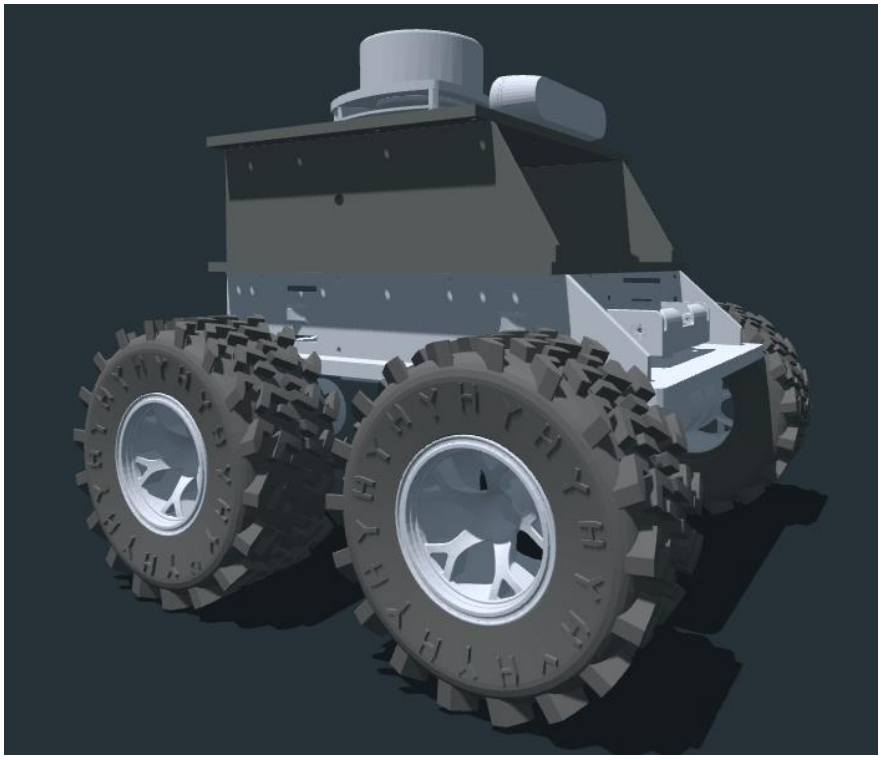


Hình 12 Cấu trúc TF của một Robot

2.2.5. Một số phần mềm mô phỏng trong ROS

RVIZ và Gazeo là hai phần mềm nổi tiếng nhất trong ROS. Đây là một giao diện đồ họa của ROS, cung cấp khả năng mô phỏng chính xác và hiệu quả robot trong môi trường phức tạp cả trong nhà lẫn ngoài trời, cho phép hiển thị dữ liệu 3 chiều của các loại cảm biến

hoặc các robot sử dụng phương thức mô tả mô hình vật lý Unified Robot Description Format (URDF).



Hình 13 Mô hình có được sau khi xuất file URDF từ phần mềm Solidwork

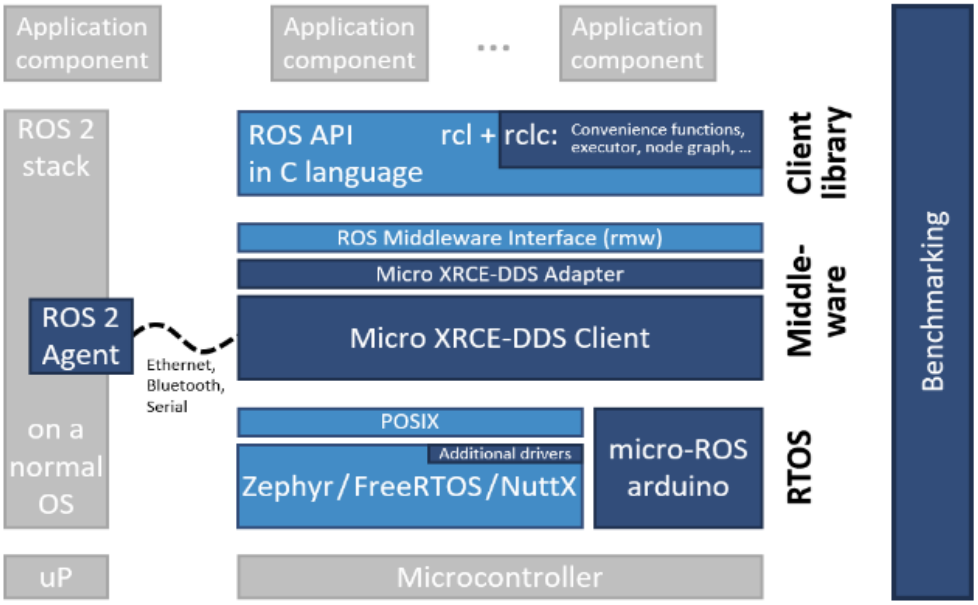
2.2.6 Giao tiếp Micro-ROS (uROS)

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

Dưới đây là một số điểm chính về giao thức uROS trong ROS 2:

- Thiết kế cho Hệ thống Nhúng: giao thức uROS được tối ưu hóa cho các hệ thống có tài nguyên hạn chế, với mục tiêu chạy trên các thiết bị nhúng và vi điều khiển với bộ nhớ và xử lý giới hạn.
- Hiệu Suất và Kích Thước Nhỏ: uROS được thiết kế để có hiệu suất tốt và kích thước nhỏ, giúp giảm yêu cầu về băng thông và tài nguyên.

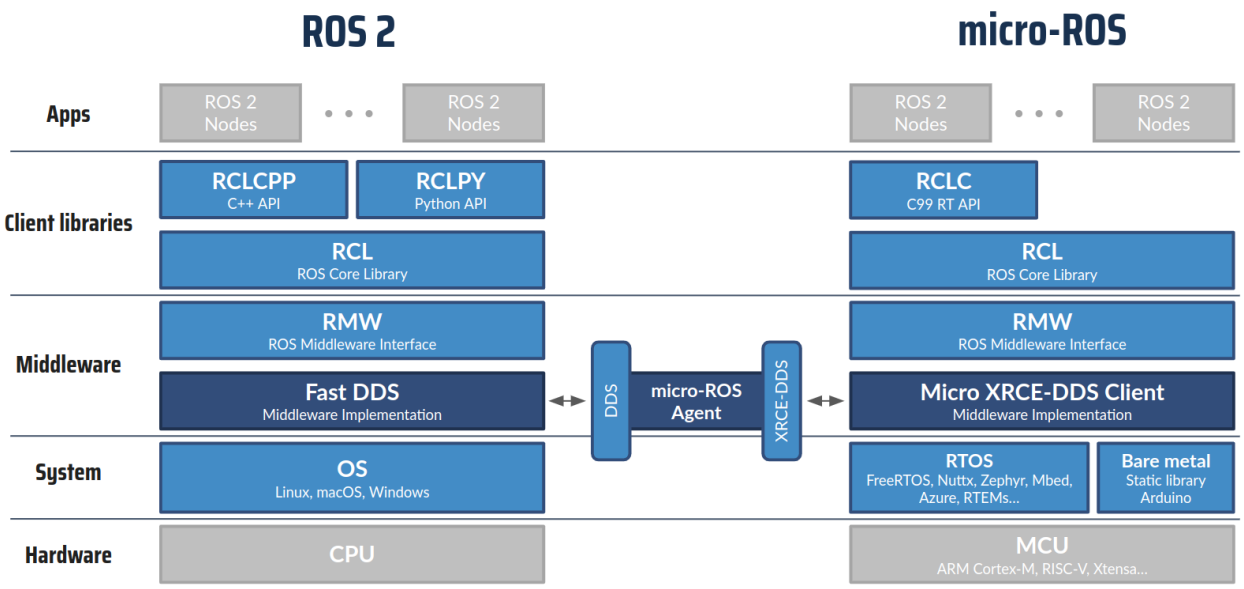
- Hỗ trợ các Middleware Nhúng: uROS có thể tích hợp với các middleware nhúng như Micro XRCE-DDS (Data Distribution Service) để cung cấp các tính năng như truyền thông, đồng bộ dữ liệu, và phát hiện nhanh chóng trong môi trường có hạn.
- Phần Mềm Bảo Mật: giao thức uROS hỗ trợ các tính năng bảo mật như mã hóa và xác thực để đảm bảo an toàn trong quá trình truyền thông dữ liệu.
- Hỗ trợ ROS2: uROS không thay thế ROS2 mà là một phần của ROS2, đặc biệt là được tích hợp với ROS2 trong hệ thống ROS2 chung.
- Hỗ trợ Các Ngôn Ngữ Lập Trình: Giao thức uROS hỗ trợ việc viết ứng dụng với nhiều ngôn ngữ lập trình khác nhau, giúp linh hoạt khi phát triển ứng dụng trên các nền tảng nhúng.
- Đơn Giản Hóa cho Hệ thống Nhúng: mục tiêu của uROS là đơn giản hóa việc triển khai ROS 2 trên các hệ thống có tài nguyên hạn chế, giúp nhà phát triển dễ dàng tích hợp ROS 2 vào các ứng dụng nhúng.



Hình 14 Kiến trúc của uROS[2]

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

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



Hình 15 Hình ảnh minh họa uROS kết nối giữa CPU và MCU

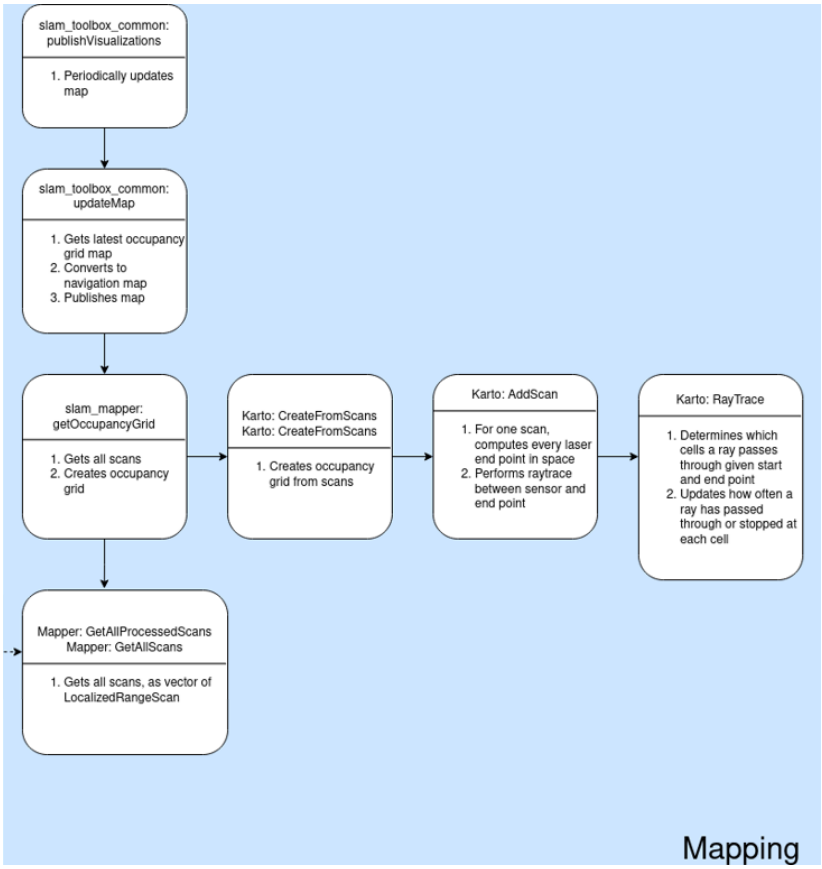
2.3. Thuật toán Slam

Simultaneous Localization and Mapping (SLAM) là một lĩnh vực quan trọng trong robot học, và nó giải quyết vấn đề của việc tự định vị robot và đồng thời xây dựng bản đồ của môi trường xung quanh robot khi nó di chuyển trong không gian không biết trước.

Trong quá trình SLAM, một bước quan trọng là "scan matching." Scan matching là quá trình so sánh dữ liệu điểm của các lần đo quét từ cảm biến với bản đồ hiện tại để xác định vị trí của robot trong bản đồ. Nói một cách đơn giản, scan matching giúp định vị robot bằng cách so sánh dữ liệu thực tế từ cảm biến với dữ liệu dự kiến từ bản đồ đã tạo ra trước đó.

Ceres Solver là một thư viện tối ưu hóa được sử dụng trong nhiều ứng dụng, bao gồm cả quá trình scan matching trong SLAM. Trong ngữ cảnh SLAM, Ceres Solver có thể được sử dụng để tối ưu hóa vị trí của robot dựa trên thông tin từ cảm biến. Nó giúp giải quyết bài toán tối ưu hóa để điều chỉnh vị trí và hình dạng của robot sao cho dữ liệu đo được từ cảm biến phù hợp nhất với bản đồ đã xây dựng.

Dùng thư viện Ceres Solver để thực hiện quá trình mapping với gói sam_toolbox trong ROS2.



Hình 16 Khối mapping

Trong slam_toolbox package, cấu hình cho thư viện Ceres Solver để thực hiện quá trình mapping như sau:

- Sử dụng Sparse Normal Cholesky là một phương pháp giải hệ tuyến tính sử dụng phân rã Cholesky cho ma trận thưa thớt (sparse matrix). Nó thích hợp cho các bài toán tối ưu hóa lớn và thưa thớt như trong các ứng dụng SLAM. Khi A là một ma trận hệ số thưa thớt, nghĩa là có nhiều phần tử 0, Sparse Normal Cholesky tận dụng cấu trúc thưa thớt để giảm độ phức tạp tính toán. Phương pháp này là một biến thể của phân rã Cholesky được tối ưu hóa cho ma trận thưa thớt.

- Sử dụng Phân rã Schur-Jacobi giải bài toán tối ưu và cụ thể là trong Ceres Solver để giải quyết các hệ phương trình tuyến tính trong quá trình tối ưu hóa. Dưới đây là mô tả chi tiết:

  Phương trình Schur-Jacobi: Phân rã Schur-Jacobi của một ma trận A được thực hiện bằng cách phân rã A thành một khối tam giác trên R và một khối tam giác dưới L theo công thức:

  $$A = R \bullet D \bullet L$$

  trong đó: R là ma trận trên, D là ma trận chéo và L là ma trận dưới.

- Sử dụng phương pháp Levenberg-Marquardt xác định chiến lược đánh giá độ tin cậy cho bước cải thiện trong quá trình tối ưu hóa. Đây là một phương pháp kết hợp giữa phương pháp Gauss-Newton và phương pháp đòi hỏi hệ số Levenberg, giúp đảm bảo tính ổn định của quá trình tối ưu hóa.

  Phương trình Levenberg-Marquardt: Thuật toán Levenberg-Marquardt cố gắng giải một bài toán tối ưu hóa dạng:

  $$\min_x \frac{1}{2} \| f(x) \|^2$$

  Trong đó f(x) là hàm sai số giữa các giá trị tính toán và giá trị thực tế, và x là vector biến cần tối ưu. Ở mỗi bước lặp của thuật toán, ta cần giải một hệ phương trình tuyến tính có dạng:

  $$(J^T J + \lambda I)\Delta x = -J^T f$$

  Trong đó J là ma trận Jacobi của f(x), λ là hệ số Levenberg-Marquardt, Δx là thay đổi ước lượng của x.

- Sử dụng phương pháp Dogleg là kết hợp giữa hai phương pháp tối ưu hóa: phương pháp Cauchy step và phương pháp Newton step để chọn bước di chuyển tốt nhất trong không gian tham số.

  Thuật toán Dogleg cố gắng giải bài toán tối ưu hóa dạng:

  $$\min_{\Delta x} m(\Delta x) = \| f(x) + J(x)\Delta x \|^2$$

Trong đó: f(x) là hàm sai số giữa các giá trị tính toán và giá trị thực tế, J(x) là ma trận Jacobi của f(x), Δx là bước di chuyển ước lượng của x. Ở mỗi bước lặp, Dogleg tìm kiếm giải bài toán con tối ưu: $\min_{\Delta x} m(\Delta x)$

- Ta giả sử dữ liệu là chính xác và không có nhiễu, do đó ta không sử hàm mất mát đặc biệt, có nghĩa là đang giả sử không có sự chênh lệch giữa giá trị ước lượng và giá trị thực tế.

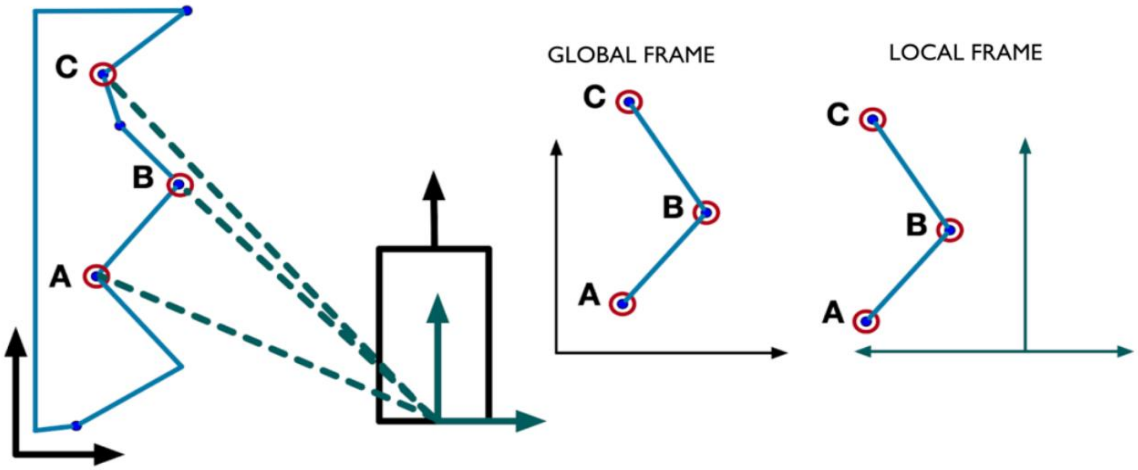
```
slam_toolbox:
  ros__parameters:
    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None
```

Sau khi chạy slam_toolbox package cùng với các config trên, ta được bản đồ mapping không gian và phép biến đổi tọa độ từ map sang odom.

2.4. Thuật toán scan matching.

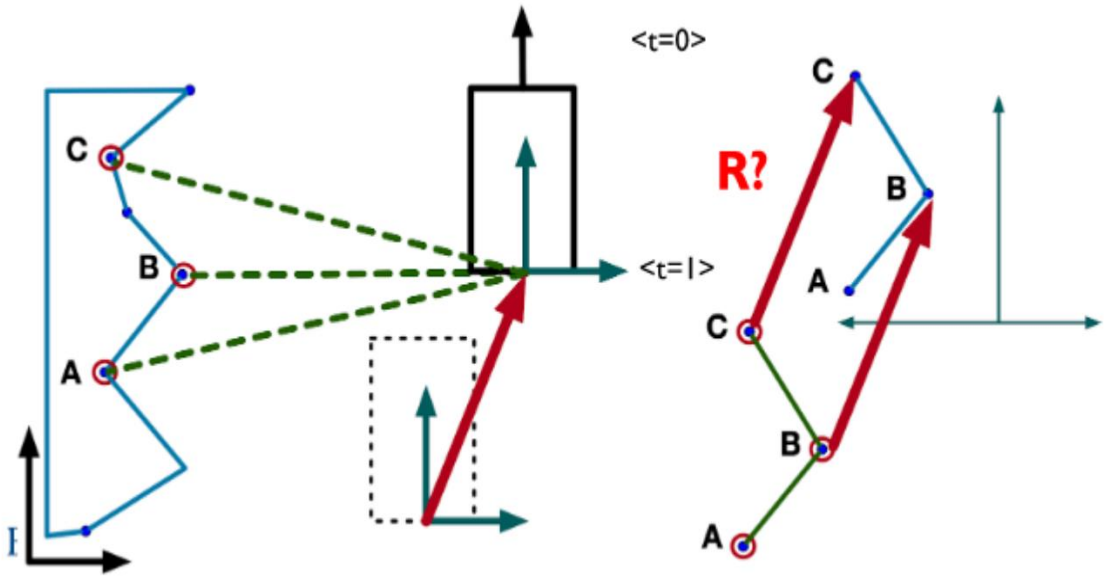
Đây là thuật toán hỗ trợ robot có trang bị cảm biến lidar, giúp tối ưu hóa sự liên kết giữa các chùm điểm cuối (end-points) trên bản đồ và các chùm điểm cuối vừa mới thu được từ cảm biến lidar.

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



Hình 17 Minh họa khoảng cách từ robot đến các điểm mốc trên hệ tọa độ

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



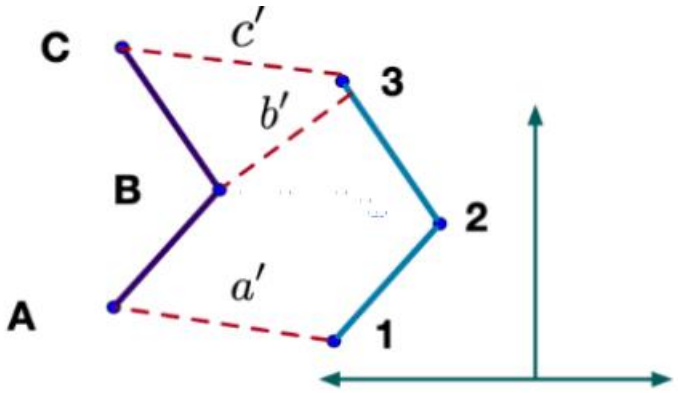
Hình 18 Minh họa phép biến đổi giữa 2 lần di chuyển của robot so với các điểm mốc trên hệ tọa độ

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

2.4.1. Thuật toán Iterative Closest Points ICP

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

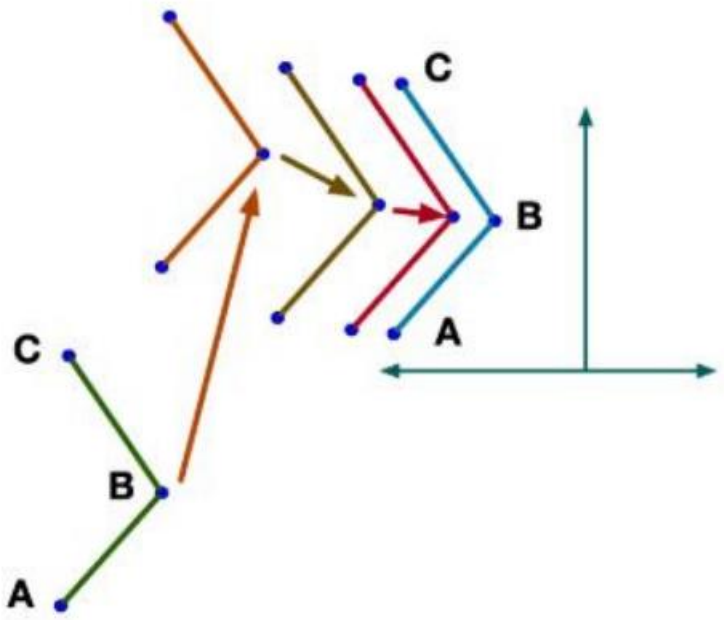
Điều kiện tối ưu 2-norm: $\| x \|_2 = \sqrt{\sum_i^m \Delta x_i^2}$ trong đó m là số điểm thu thập được, Δx_i là khoảng các giữa các điểm tương ứng:



Hình 19 Chọn các điểm tương ứng trên hai tập dữ liệu

Các bước thực hiện thuật toán:

1. Khởi tạo giá trị error và giá trị ngưỡng threshold (error > threshold)
2. Định nghĩa các cặp điểm tương ứng bằng cách chọn các cặp điểm dữ liệu mới và cũ gần nhau nhất.
3. Tính toán vector xoay R và vector dịch chuyển T giữa giá trị dữ liệu cũ so với dư liệu lidar mới thu thập được.
4. Dùng giá trị R và T để tính giá trị sai số error = $\| x \|_2 = \sqrt{\sum_i^m \Delta x_i^2}$
5. Nếu giá trị error giảm và lớn hơn mức ngưỡng threshold thì ta quay lại bước 2, ngược lại xuất giá trị R và T để xác định vị trí robot trong bản đồ.



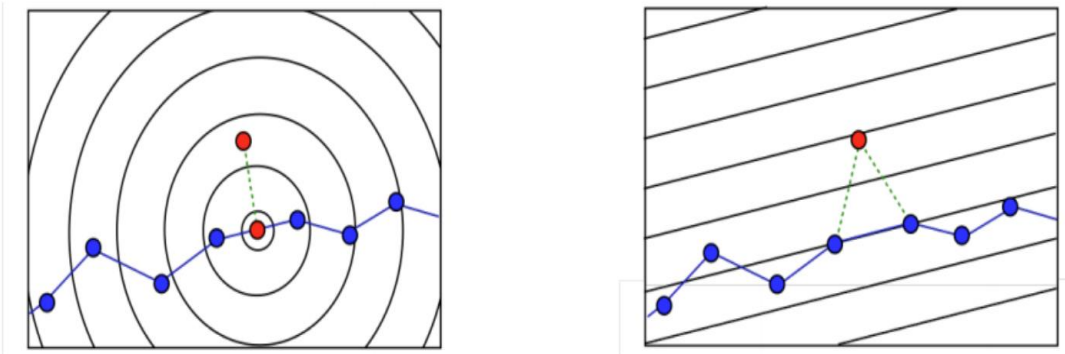
Hình 20 Minh họa lần lượt các bước lặp dự đoán để so khớp giữa 2 vị trí

Các hạn chế của thuật toán:

- Thuật toán hoạt động tốt nhất với các giả định sau nhưng lại ít xảy ra trong thực tế: Cần tốc độ quét cực nhanh để có: sự chồng chéo đáng kể tồn tại giữa các lần quét liên tiếp và tồn tại các điểm có sự tương ứng cao.Giả định bề mặt nhẵn và đồng nhất.
- Thiết lập dự đoán là quan trọng trong ICP để hội tụ
- Việc lặp lại tính toán giữa từng điểm tốn nhiều tài nguyên và thời gian, dẫn tối không được real-time và gây ra nhiều sai số nếu quá trình chuyển động nhanh.

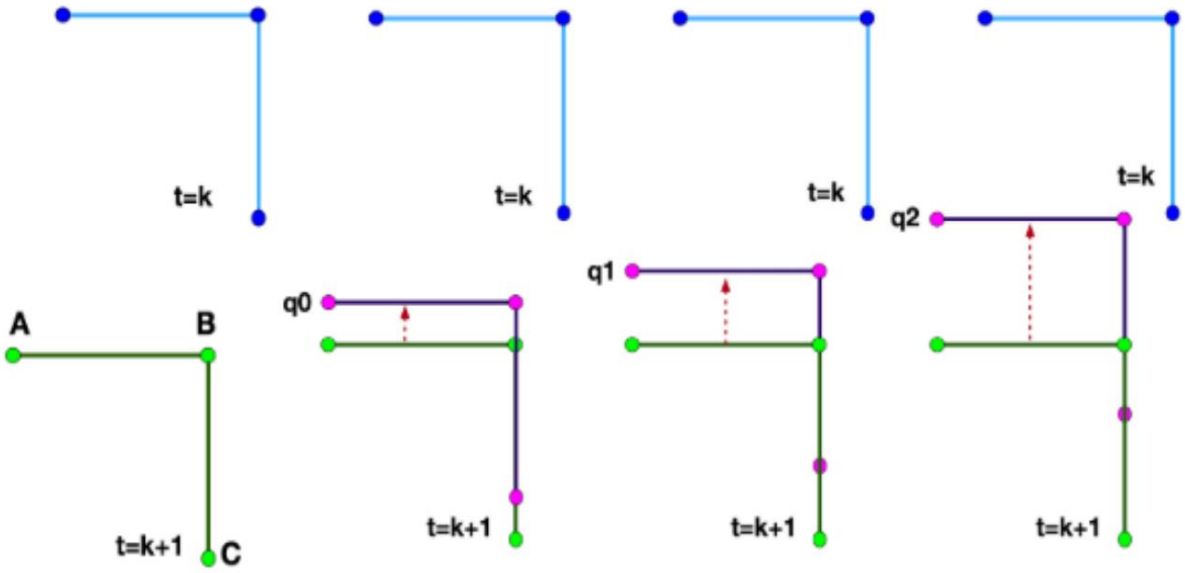
2.4.2. Thuật toán PL-ICP

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



Hình 21 Dữ liệu điểm-điểm xấp xỉ khoảng cách đến bề mặt tốt hơn so với số liệu điểm-điểm được sử dụng trong ICP vanilla.[3]

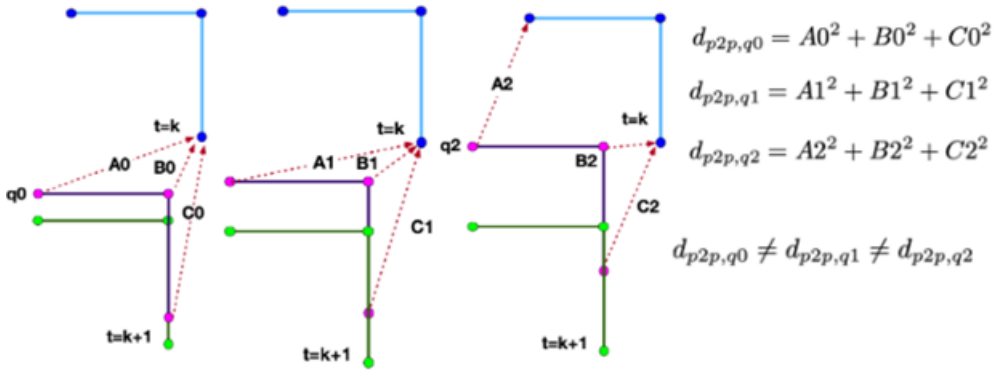
Ta lấy ví dụ minh họa, để so khớp thời điểm t=k và thời điểm t=k+1, lần lượt có các dự đoán q0, q1, q2 cần tính toán sự tương ứng:



Hình 22 Minh họa các dự đoán

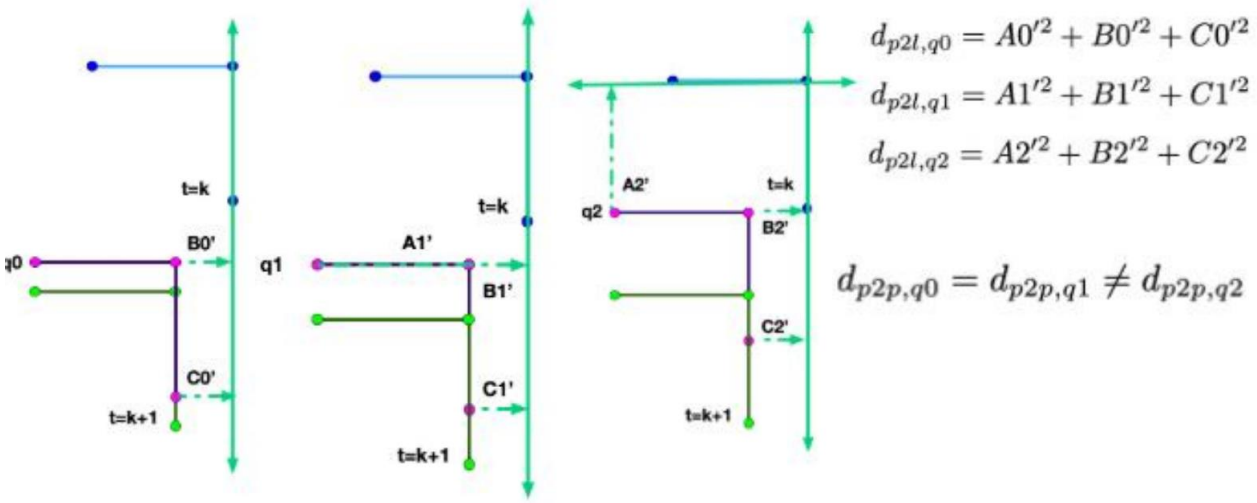
Đối với phương pháp tính toán point-to-point, ta phải lặp đủ số lần do tính toán các khoảng cách so khớp của các dự đoán q0,q1,q2 khác nhau

[3] Andrea Censi, "An ICP variant using a point-to-line metric", in 2008 IEEE International Conference on Robotics and Automation, 2008.



Hình 23. Minh họa phương pháp tính point-to-point

Với point-to-line, chúng ta do kết quả tính toán bằng nhau, số lần lặp được giảm bớt (lần lặp tiếp theo từ q0 sang q2, bỏ qua lần lặp q1 do kết quả giống nhau)



Hình 24 Minh họa phương pháp tính point-to-line

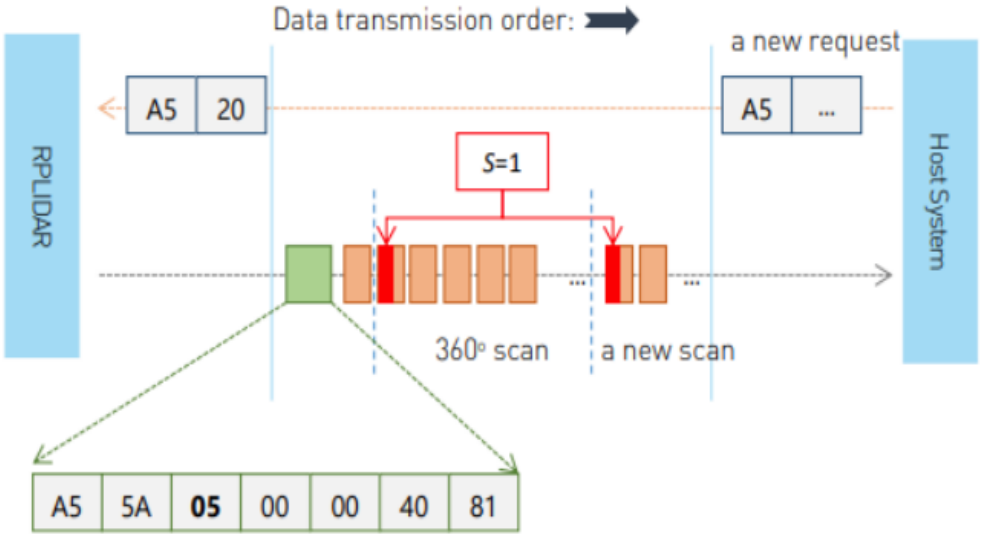
CHƯƠNG 3. NỘI DUNG THỰC HIỆN

3.1. Các module phần cứng

3.1.1. RPLidar A1

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

Hình 25. RPLidar A1



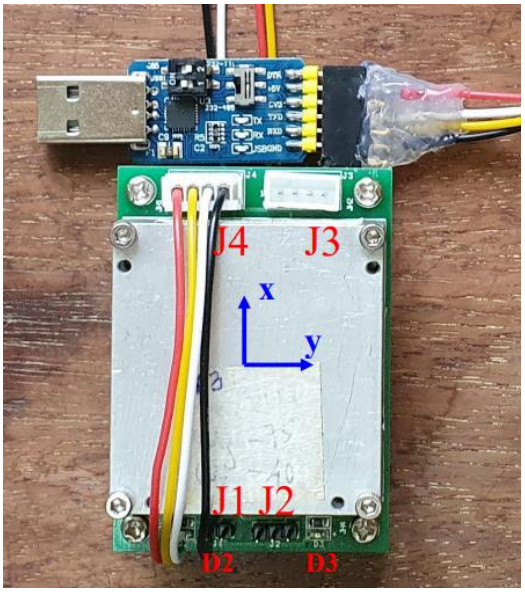
Hình 27 Các Packet truyền khi LiDAR quét 360 độ

3.1.2. Cảm biến IMU ADIS16488 được nhúng trên board mạch chứa vi xử lí STM32F405RG

16488 là một đơn vị đo quán tính (IMU) dựa trên công nghệ gia công vi mô (MEMS),

bao gồm con quay hồi chuyển MEMS hiệu suất cao và gia tốc MEMS và đưa ra ba vận tốc góc và ba gia tốc.

Đồng thời board nhúng hỗ trợ thuật toán ước lượng góc xây dựng trên cảm biến có hoặc không sử dụng từ trường, tự ước lượng giá trị nhiễu nền của cảm biến.



Hình 28. Cảm biến IMU ADIS16488

3.1.3. Máy tính nhúng Jetson Nano

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

Hình 26. NVIDIA Jetson Nano

3.1.4. Vi điều khiển Raspberry Pico

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

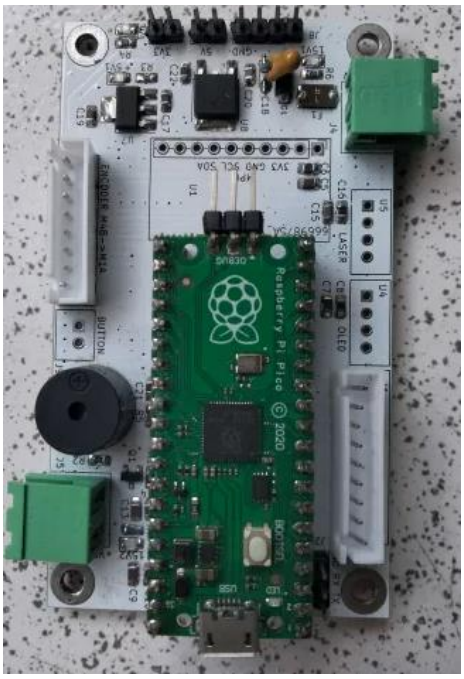
Hình 27. Raspberry Pico

3.1.5. Driver motor, main board và DC Servo JGB37-520 có Encoder

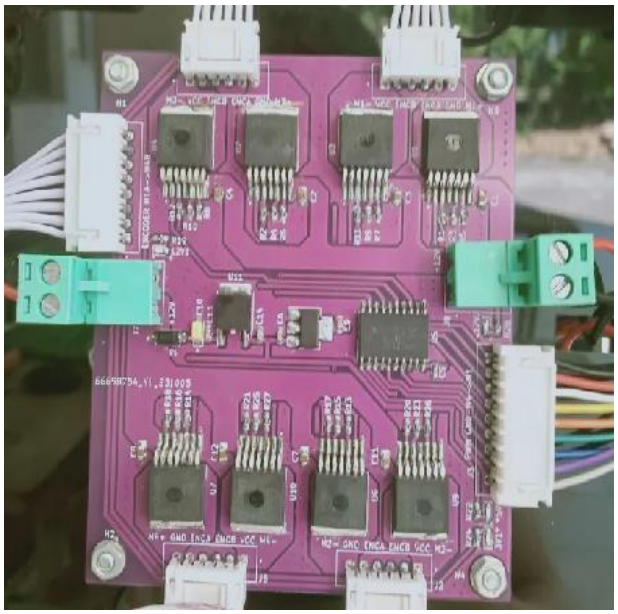
Board mạch cầu H dùng IC BTS7960, driver sử dụng nguồn 12V-4A, dùng để điều khiển 4 động cơ DC. Driver nhận đầu vào là 8 xung điều khiển PWM, đầu ra là 8 chân nguồn cấp cho 4 động cơ.

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

DC Servo JGB37-520 sử dụng nguồn 12V-1A (max) 250RPM (có tải)



Hình 28. Main board



Hình 29. Driver điều khiển motor

Hình 30. DC Servo JBG37-520

3.1.6. Pin sạc 18650 Li-ion

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

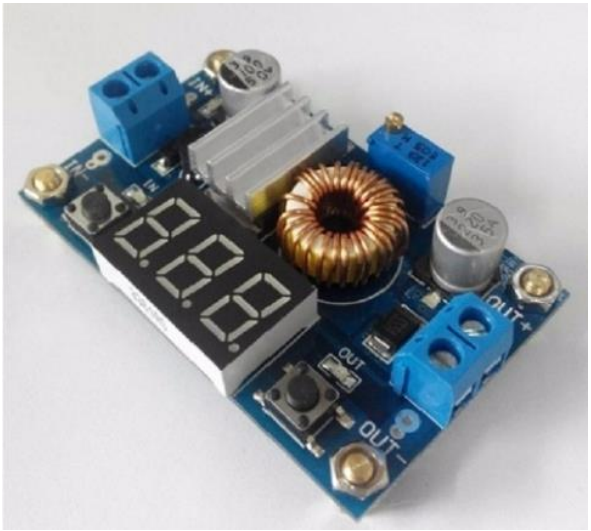
Hình 31. Pin sạc Li-ion 18650

3.1.7. LM2596S mạch giảm áp 3A và XL4015 mạch giảm áp 5A

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

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

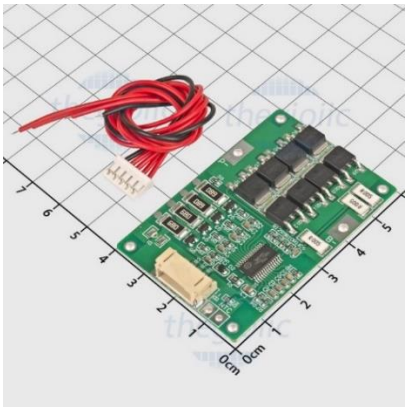
Hình 32. Mạch giảm áp DC LM2596S 3A



Hình 33. Mạch giảm áp DC XL4015 5A

3.1.8. Mạch sạc pin 18650 4S

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

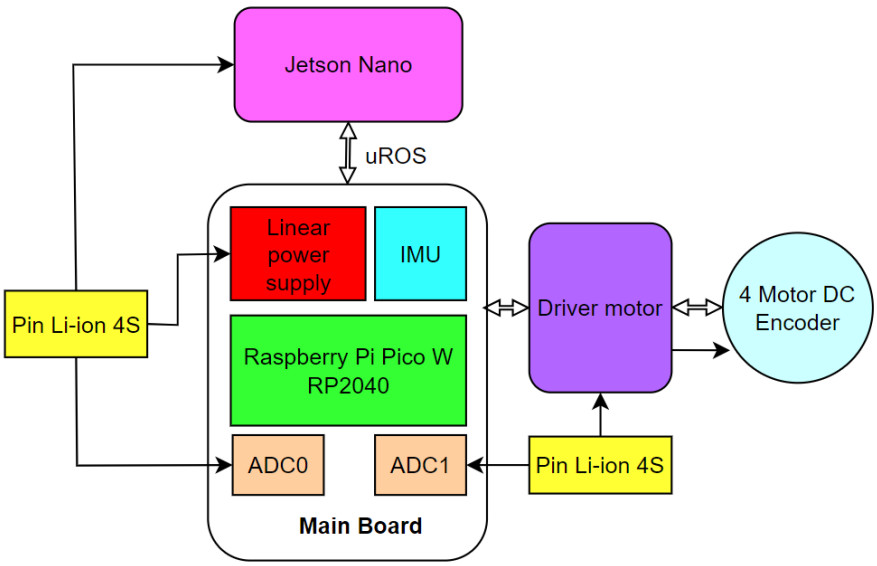


Hình 34. Mạch sạc pin 18650 4S

3.2. Kết nối phần cứng

3.2.1. Khối vi điều khiển

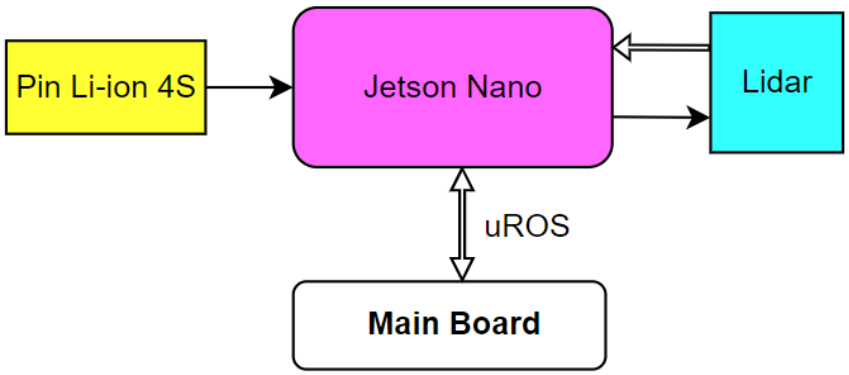
Main board được cấp nguồn bởi pin Li-ion 4S 15V đây cũng là nguồn cấp cho Jetson Nano. Khi nhận được thông tin điều khiển từ Jetson Nano (ROS) thông qua giao tiếp uROS, vi điều khiển dùng bộ điều khiển PID để xuất xung PWM cho mạch Driver điều khiển động cơ. Driver được cấp bằng nguồn Li-ion 4S 15V khác, nhằm tách riêng nguồn cho mạch công suất và nguồn cho mạch điều khiển.



Hình 35. Sơ đồ khối tổng quan mạch điều khiển

3.2.2. Khối máy tính nhúng

Hệ điều hành robot (ROS) được cài đặt trên máy tính nhúng Jetson Nano. Máy tính nhúng được cấp nguồn bởi pin Li-ion 4S 15V, qua mạch buck DC-DC XL4015 để có được nguồn 5V-4A. Cảm biến Lidar sẽ lấy nguồn từ máy tính nhúng thông qua cổng USB.

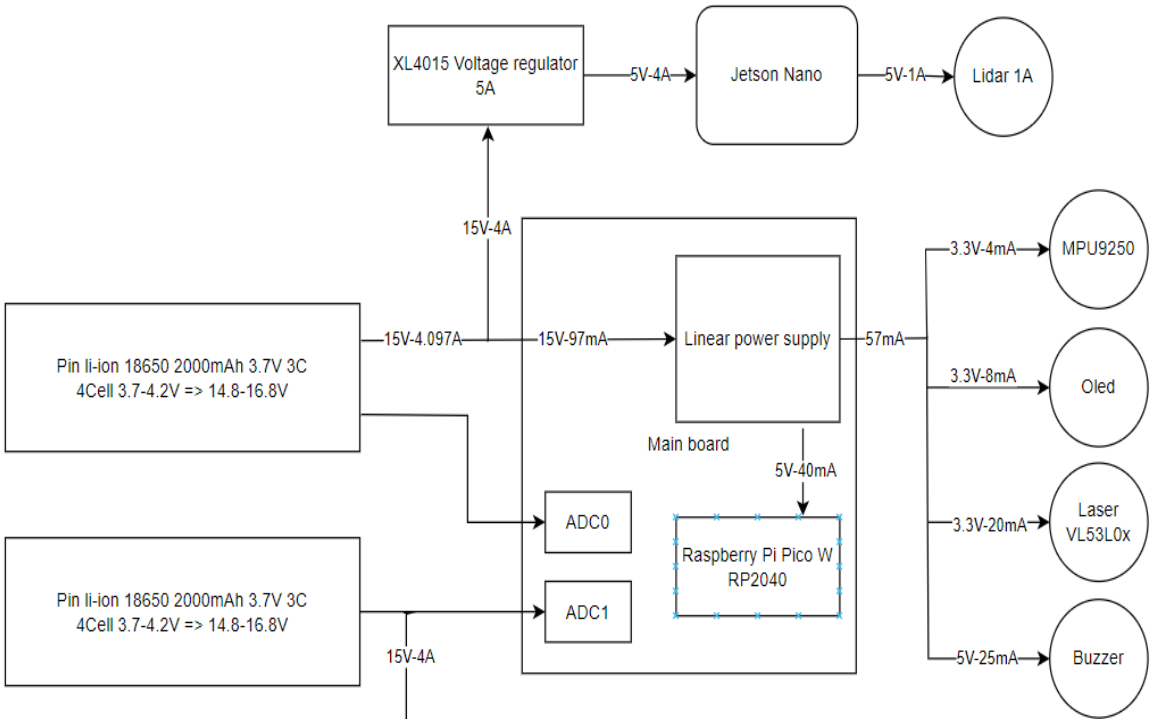


Hình 36. Sơ đồ khối máy tính nhúng

3.3. Thi công mạch điều khiển

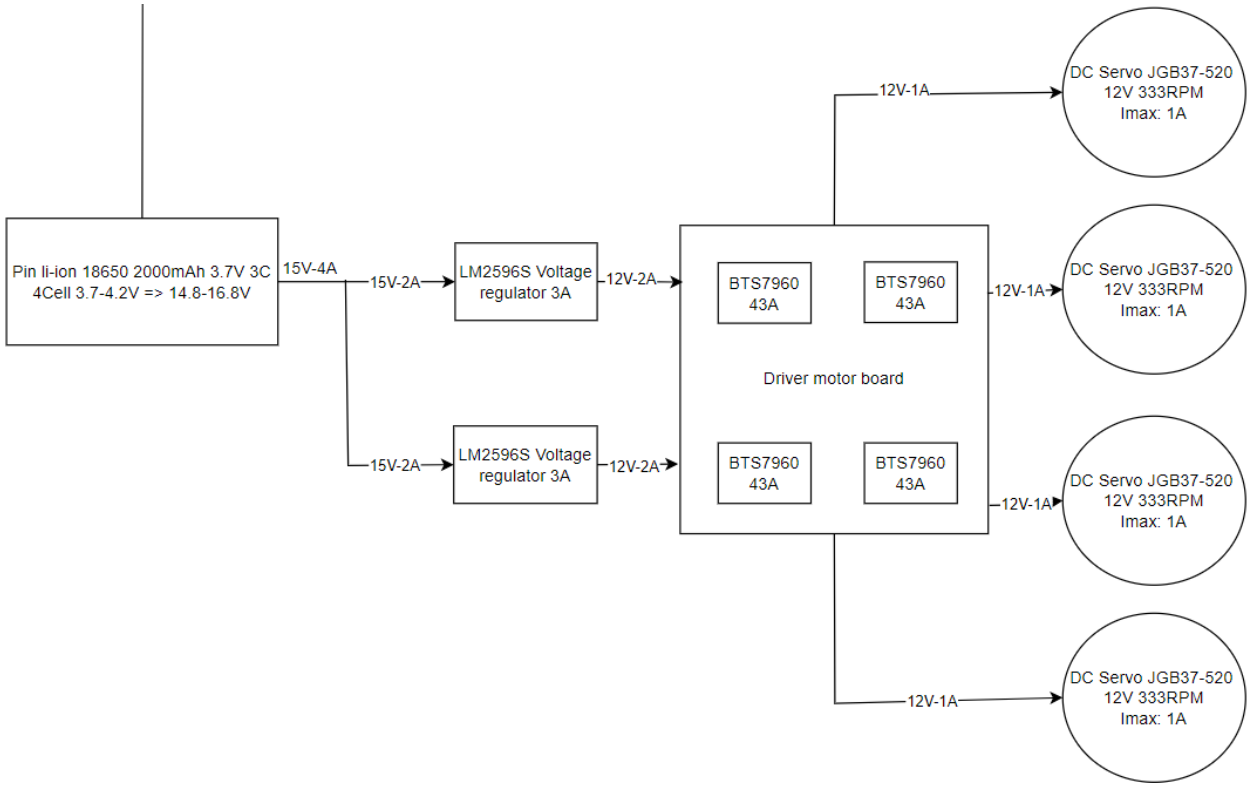
3.3.1. Power diagram của toàn hệ thống

Công suất phần khối điều khiển chính:



Hình 37. Power diagram của khối điều khiển

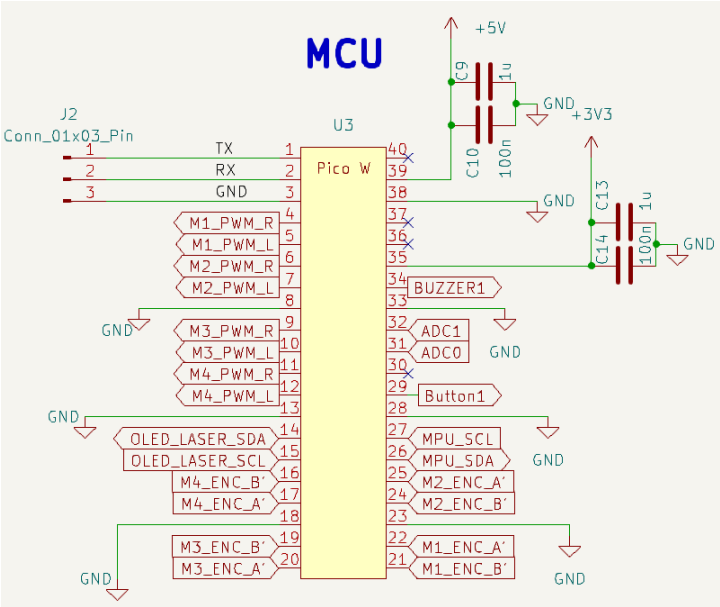
Công suất phần động lực



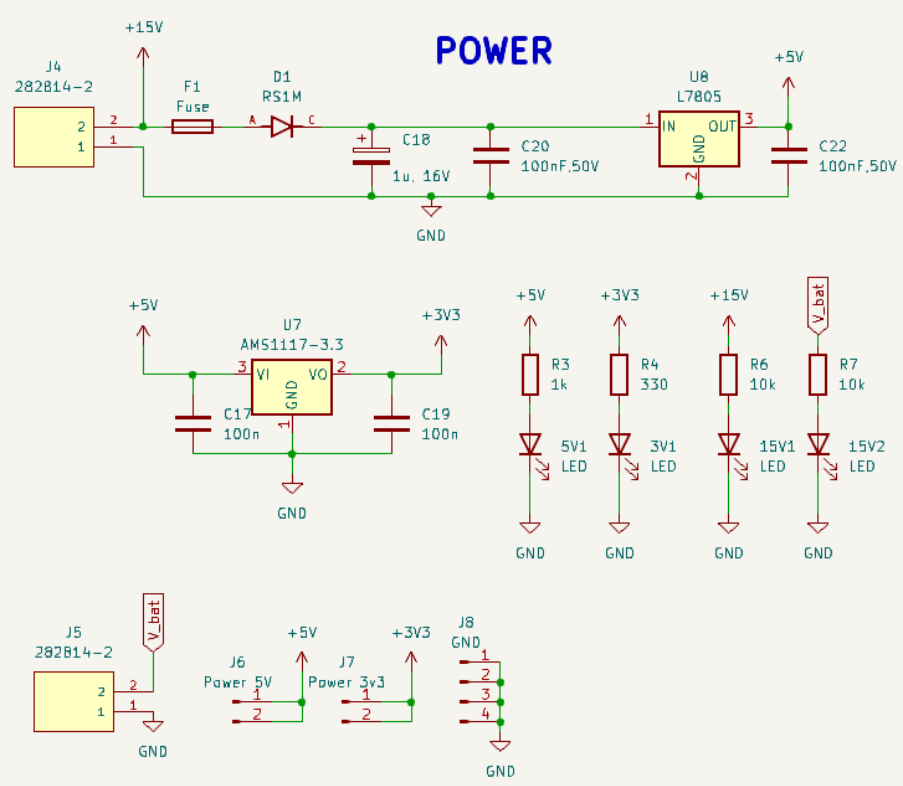
Hình 38. Power diagram phần động lực

3.3.2. Mạch điều khiển chính chứa vi điều khiển Raspberry Pico

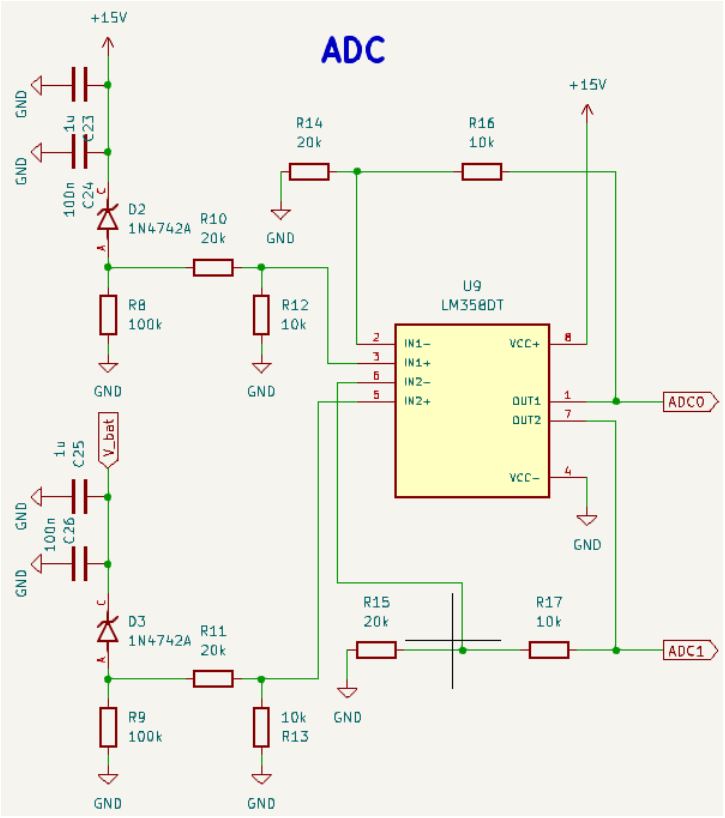
Schematic của từng khối trong main board



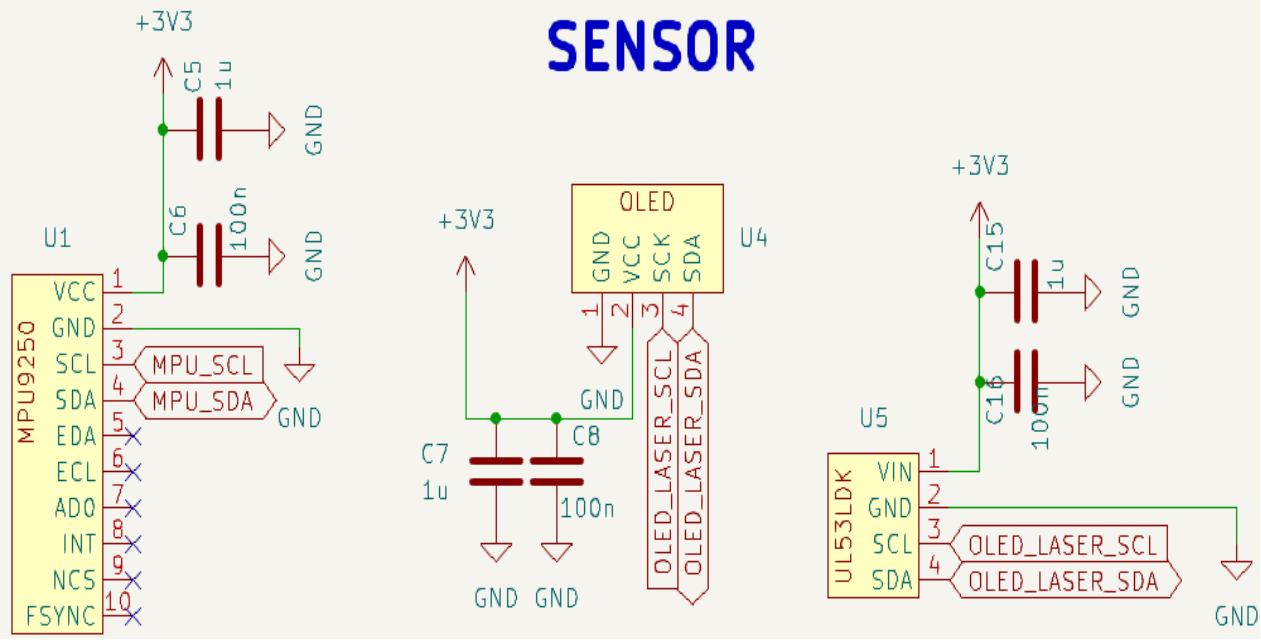
Hình 39. Schematic của khối vi điều khiển



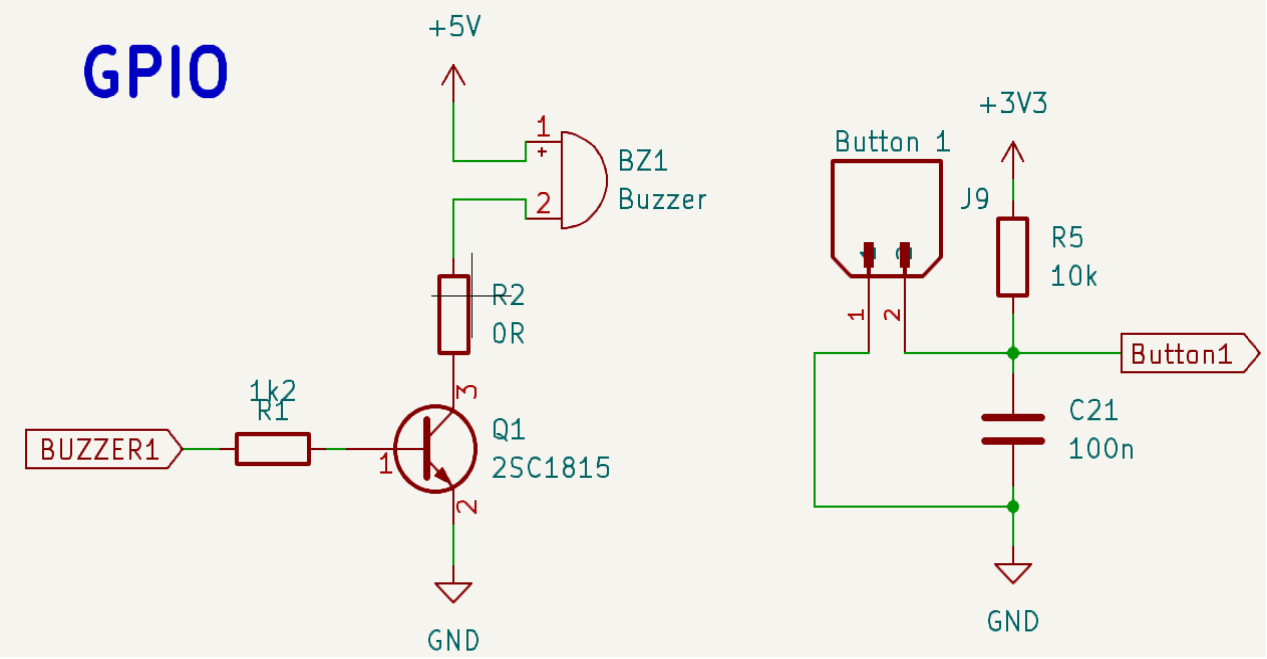
Hình 40. Schematic khối nguồn của main board



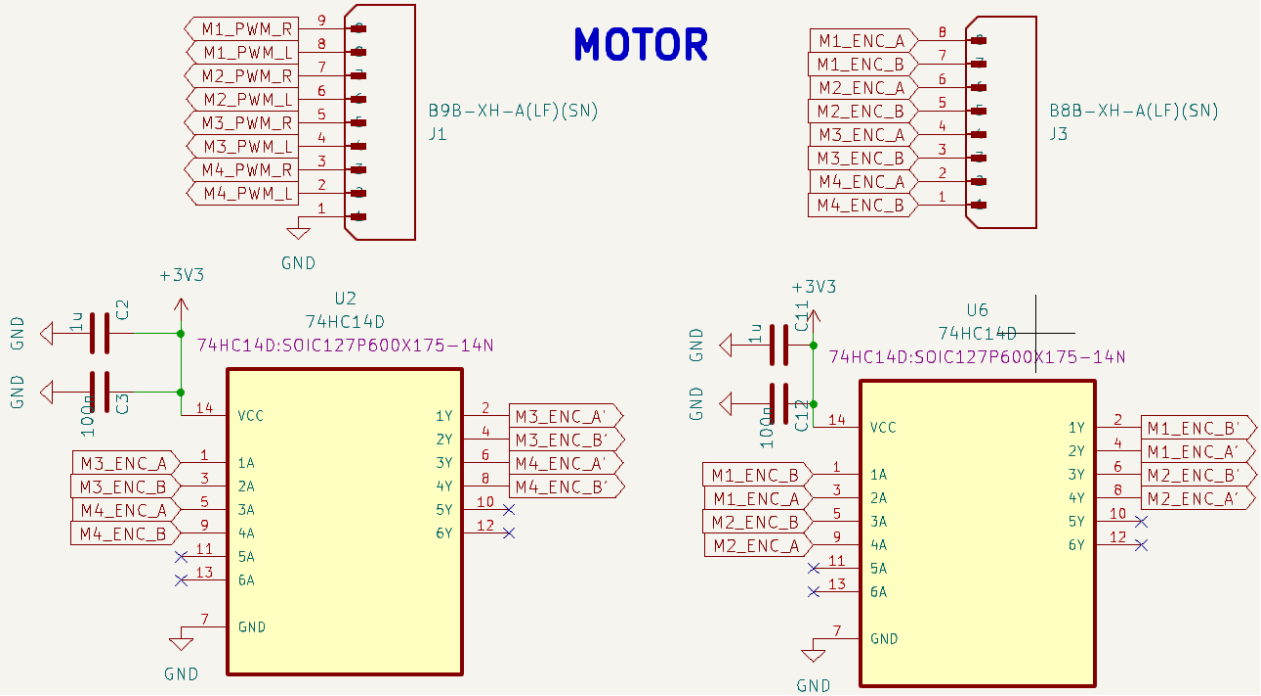
Hình 41. Schematic của khối adc



Hình 42. Schematic của khối cảm biến

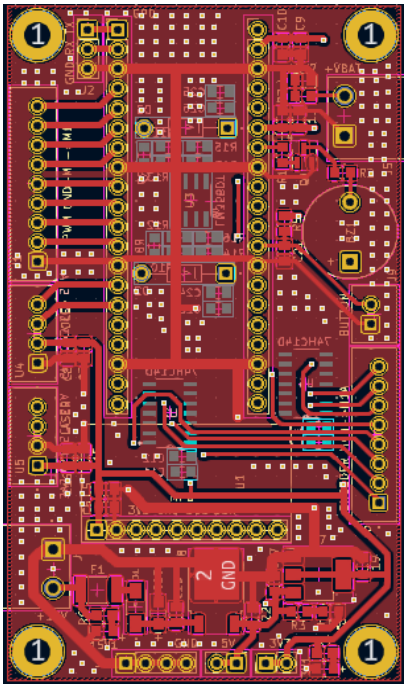


Hình 43. Schematic của khối GPIO

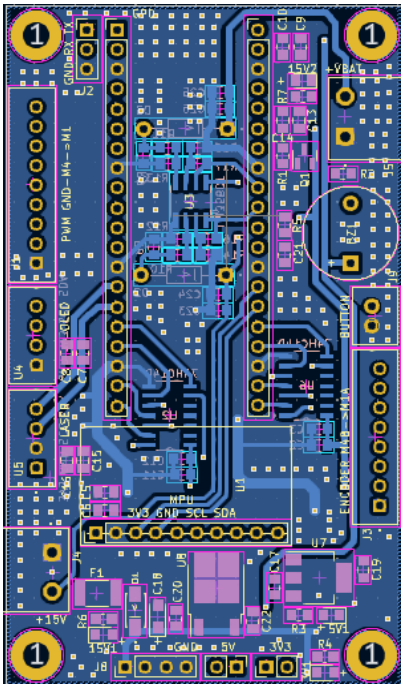


Hình 44. Schematic của khối motor

Layout của main board



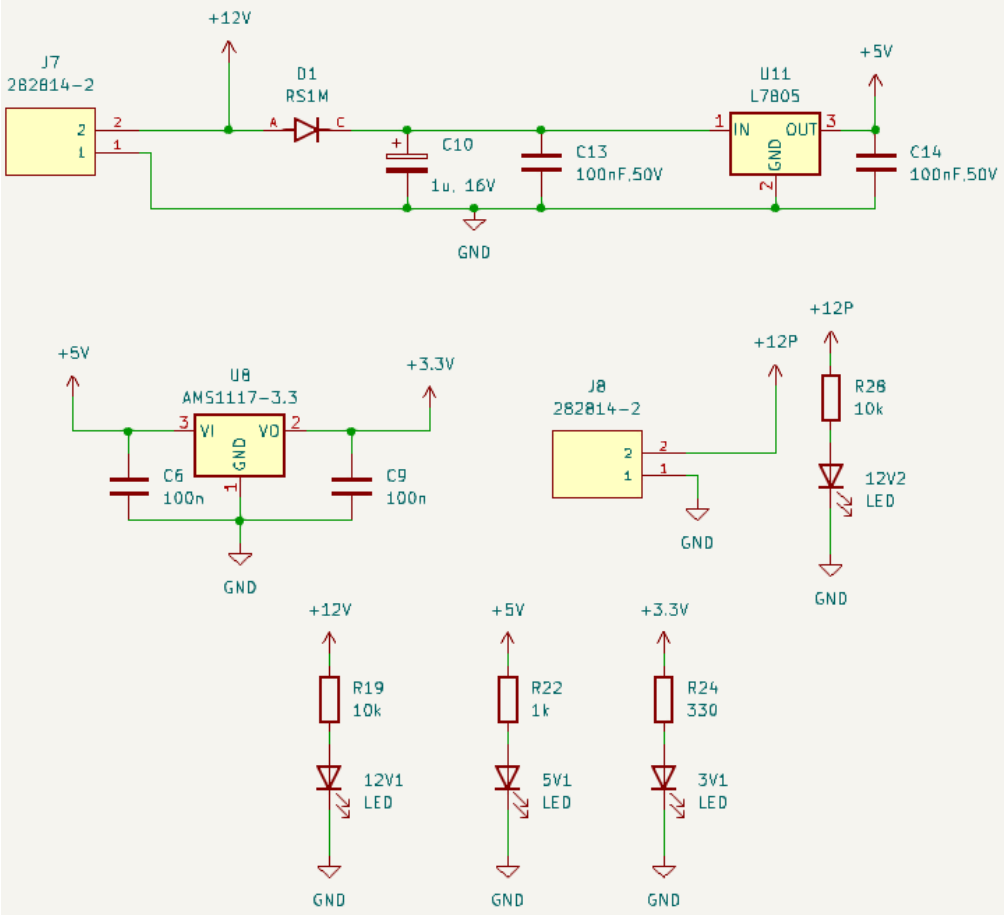
Hình 45. PCB top layer của main board



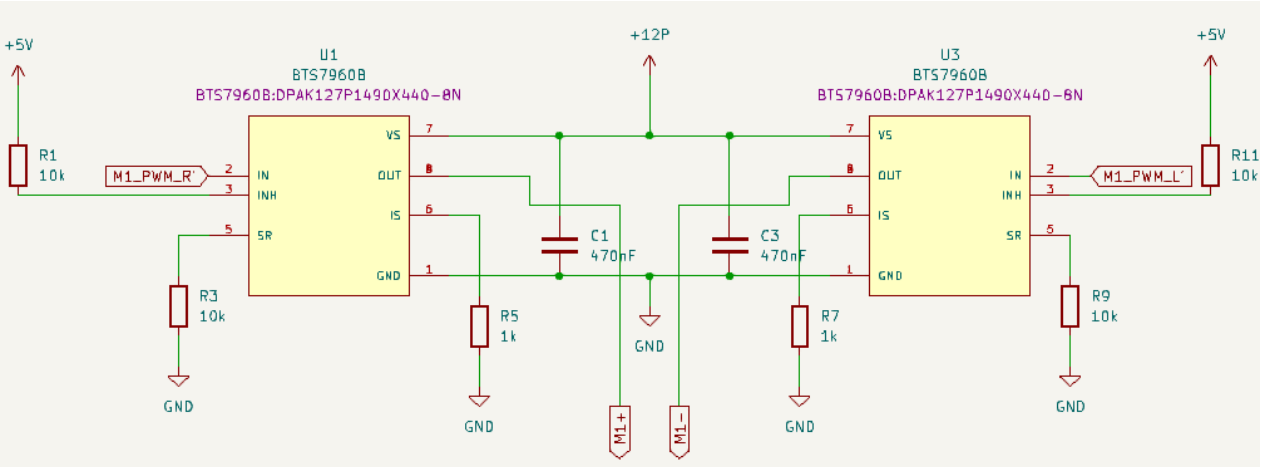
Hình 46. PCB bottom layer của main board

3.3.3. Mạch Driver điều khiển motor DC

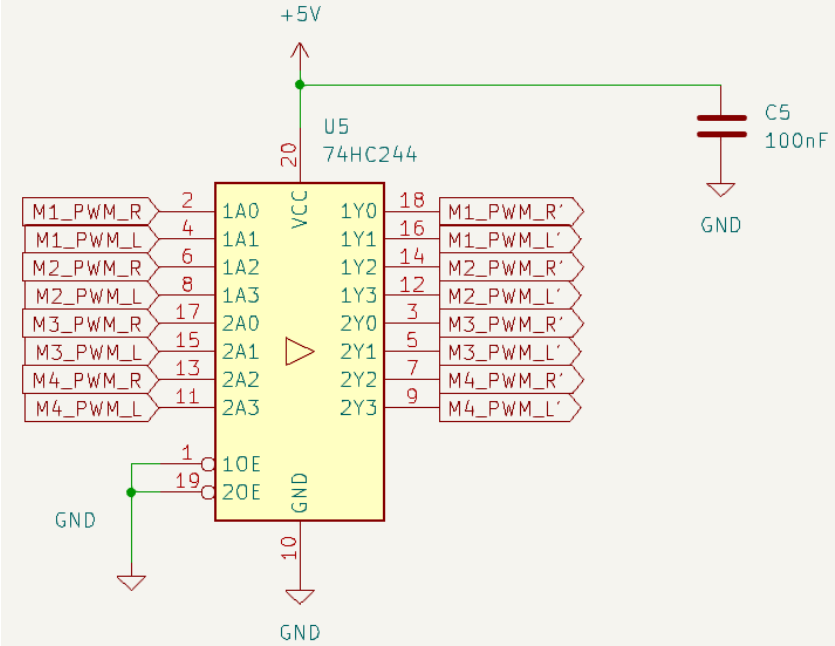
Shematic của mạch Driver điều khiển motor DC



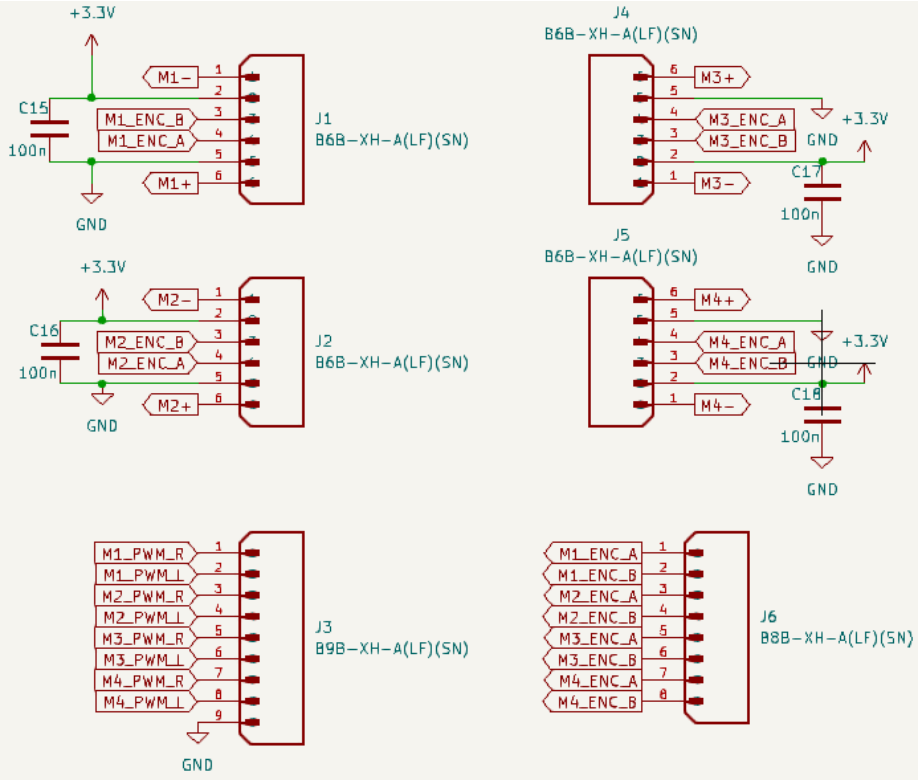
Hình 47. Schematic khối nguồn của Driver



Hình 48. Shematic của một cầu H (tương tự cho các mạch cầu còn lại)

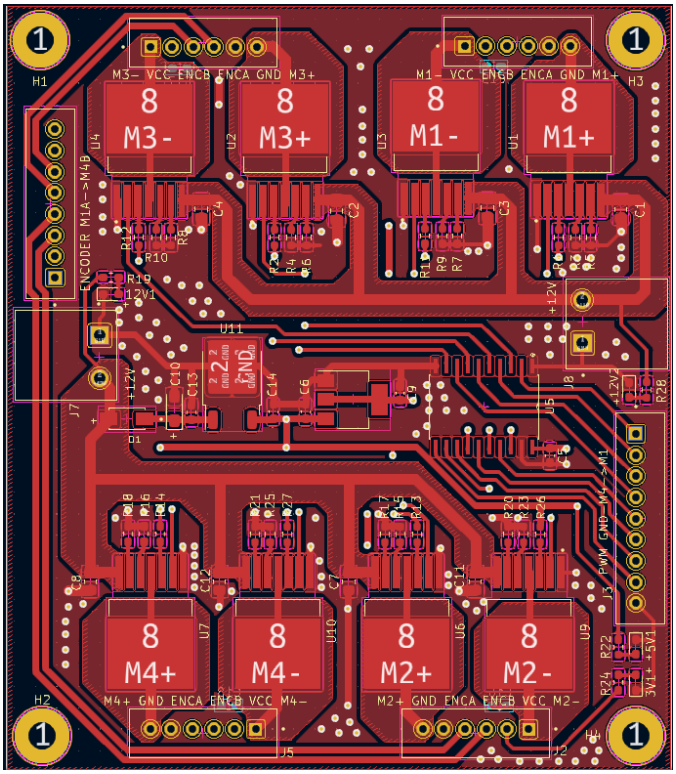


Hình 49. Schematic của IC đệm

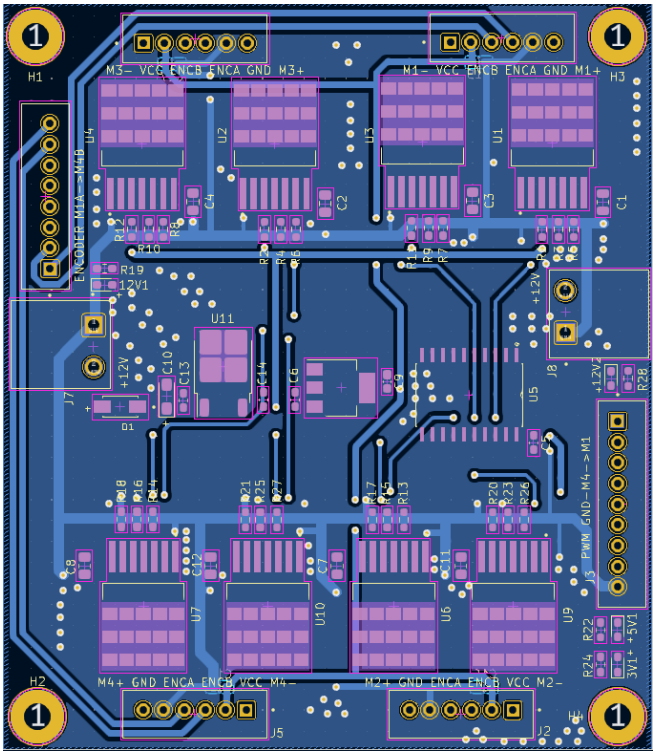


Hình 50. Schematic của khối connect

Layout của mạch Driver



Hình 51. PCB top layer của Driver



Hình 52. PCB bottom layer của Driver

3.4. Thiết kế mô hình

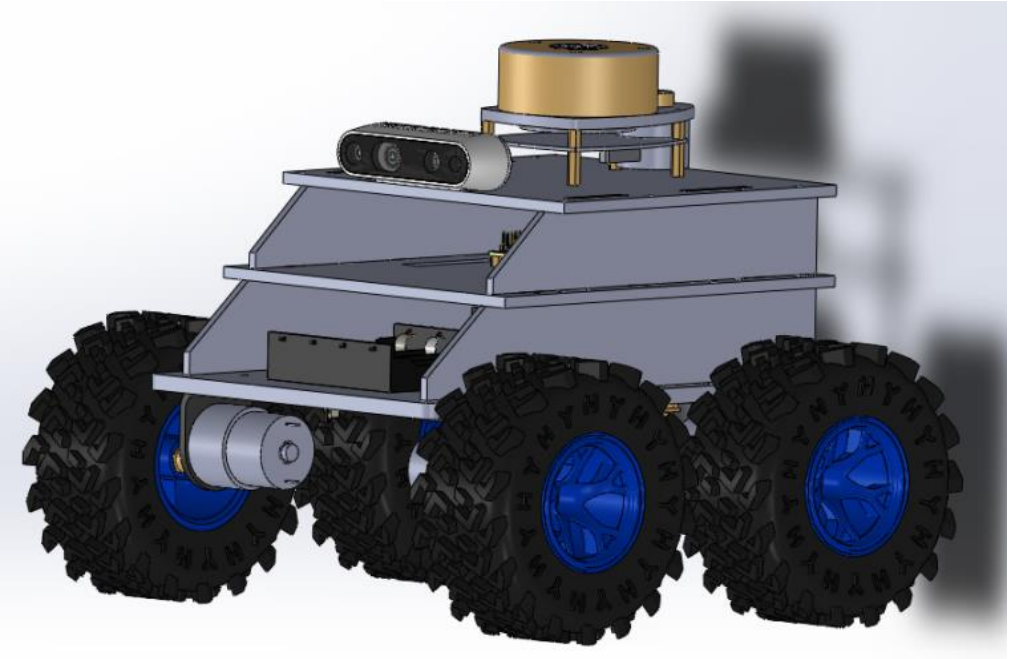
Nhóm sử dụng phần mềm vẽ 3D SolidWorks để vẽ mô hình robot, tự thiết kế các tầng robot đáp ứng các nhu cầu lắp ráp cần thiết cho các thành phần phần cứng được trình bày ở mục 3.1. Robot sẽ được vẽ với 3 tầng nhằm thực hiện 3 nhiệm vụ riêng biệt với nhau gồm:

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

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

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

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



Hình 53. Mô hình robot thiết kế dùng SolidWorks

3.5. Xây dựng Firmware

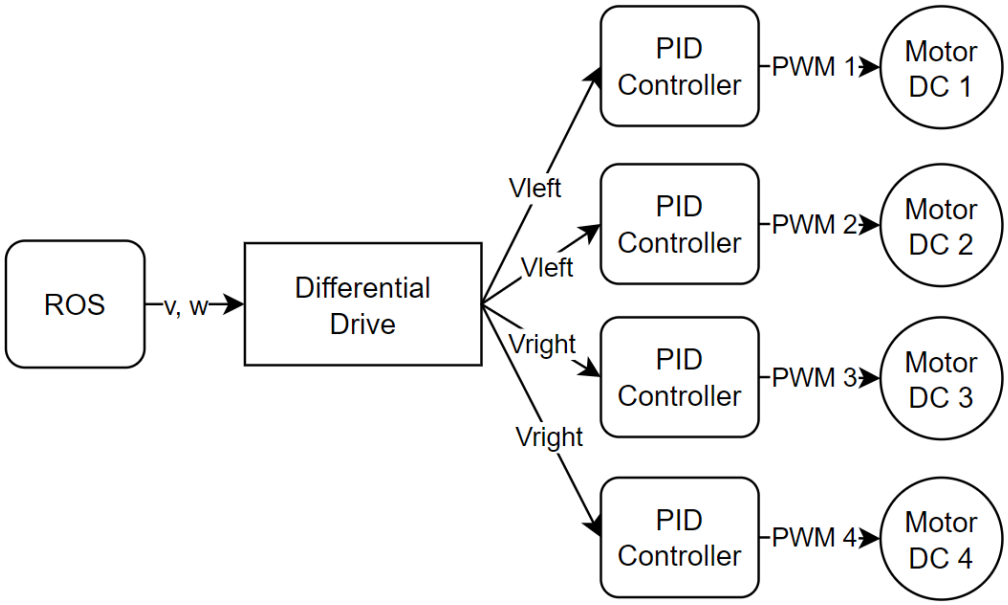
3.5.1. Điều khiển tốc độ động cơ

Việc điều khiển tay để robot di chuyển và mapping hay Navigation đều cần sự tương tác giữa máy tính nhúng và vi điều khiển. Máy tính nhúng sẽ gửi lại giá trị vận tốc tính tiến v và vận tốc xoay ω qua cổng nối tiếp bằng giao tiếp uROS. Do robot được thiết kế theo mô hình Diferential Drive, ta sẽ tính toán được giá trị vận tốc riêng cho từng bánh v_{right}, v_{left} để đạt được v và ω.

Khi có được vận tốc riêng cho từng bánh xe, đề tài sử dụng thuật toán PID để xuất xung PWM chính xác cho Driver , với mục tiêu chính là hạn chế tối thiểu sai số xác lập và độ vọt lố với thời gian xác lập ngắn.

Tín hiệu đặt	$r(t)$	v_{right} hoặc v_{left}	m/s
Tín hiệu hồi tiếp	$v(t)$	$\bar{v}_{right}$ hoặc $\bar{v}_{left}$	m/s
Tín hiệu điều khiển	$u(t)$	Chu kỳ điều khiển xung PWM	%

Bảng 1. Bảng thông tín tín hiệu bộ điều khiển PID



Hình 54. Sơ đồ khối điều khiển vận tốc 4 bánh xe

Ta có số xung khi quay một vòng của mỗi kênh encoder là 11x30 = 330 xung khi đọc encoder ở mode x4 thì encoder sẽ xuất được 1320 xung/vòng.

Giả sử chu kì đọc encoder là Tenc(s), số xung đọc được trong lần lấy mẫu là p(xung/Tenc) và bán kính của bánh xe là R(m) ta được công thức tính vận tốc của bánh xe v(m/s) như sau:

$$v = \frac{2\pi pR}{T_{enc}.10^{-3}.1320}$$

Tốc độ di chuyển trung bình của robot khoảng 20cm/s và bán kính bánh xe là 6.25cm, vì vậy nhóm lựa chọn chu kì đọc encoder là Tenc = 100 ms để trong một chu kì lấy mẫu thì số xung đọc được khoảng 67 xung, nhờ đó gai nhiễu của encoder sẽ không làm thay đổi quá lớn giá trị vận tốc của motor khi tính toán.

Động cơ đề tài sử dụng là DC Servo JGB37-520 với tốc độ tối đa khi có tải là 250 RPM. Bán kính bánh xe là 6.25cm, chiều rộng của robot là 24.69 cm và nhóm cho robot chạy với vận tốc tối đa là 70% vận tốc tối đa mà motor cho phép. Từ đó ta tìm được vận tốc tối đa của robot là 1.15 m/s và vận tốc xoay tối đa của robot là 9.32 rad/s, phần tính toán được trình bày ở phụ lục.

Dựa vào sự ảnh hưởng của các thông số lên bộ điều khiển ta có các bước tinh chỉnh thông số như sau:

Bước 1: Đặt Ki, Kd ban đầu bằng 0. Tăng Kp đến khi POT đạt 10%.

Bước 2: Tăng Ki đến khi $e_{xl} = 0$.

Bước 3: Tăng Kd đến khi POT đạt nhỏ nhất.

Chu kỳ điều khiển PID	T	100 ms
Chu kỳ đọc encoder	$T_{encoder}$	100 ms
Tần số xung PWM	T_{PWM}	25 khz
Hệ số khâu P	Kp	0.2
Hệ số khâu I	Ki	0.7
Hệ số khâu D	Kd	0

Hình 55. Bảng giá trị thông số điều khiển tốc độ motor

3.5.2. Đọc adc và xây dựng module scheduler

Raspberry Pico hỗ trợ đọc adc 12-bit từ 0-3.3V, giá trị adc đọc được từ vi điều khiển là n, điện áp tham chiếu là Vref (V), điện áp của diode zenner là Vzenner(V) và hệ số khuếch đại là K ta có công thức tính điện áp của pin như sau:

$$V_{bat} = \frac{V_{ref} n}{2^{12}} K + V_{zenner}$$

Với thiết kế mạch adc được trình bày ở mục 3.3.2 ta có bảng thông số như sau:

Điện áp tham chiếu	Vref	3.27 V
Hệ số khuếch đại	K	2
Điện áp diode Zenner	Vzenner	11.8 V

Bảng 2. Bảng giá trị thông số tính điện áp của pin

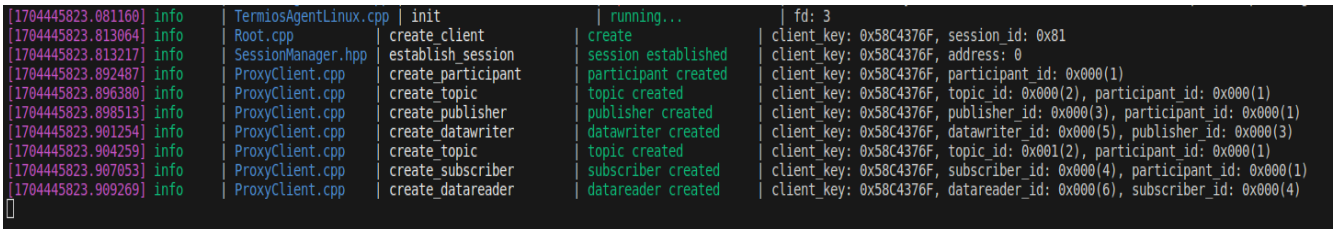
Module scheduler là module để các function đăng kí sự kiện thực hiện với chu kì T. Cấu trúc của scheduler gồm function pointer, chu kì thực hiện sự kiện và các thông số của sự kiện được khai báo như sau:

```
typedef void (*HE_CALLBACK_FUNC)(void *cxt);
typedef struct {
  HE_CALLBACK_FUNC tCallbackFunc;
  uint32_t u32Periodic;
  uint32_t u32Time;
  void *context;
} HandleEvent_t;
```

Bảng 3. Struct của module scheduler

3.5.3. Giao tiếp uROS

Sử dụng phương thức uROS, truyền dữ liệu theo giao thức UART (baudrate = 115200) tạo node tên “pico_node” trên raspberry pico subsribe đến topic /cmd_vel được tạo ra khi chạy file node teleop_twist_keyboard trên PC.



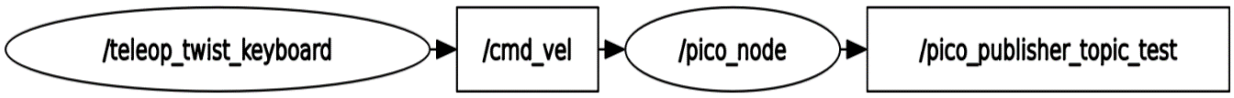
Hình 56 Chạy thành công chương trình tạo node trên raspberry pico

Kiểu message của /cmd_vel có dạng geometry_msgs/Twist:

```
Vector3 linear
float64 x
float64 y
float64 z
Vector3  angular
float64 x
float64 y
float64 z
```

Khi node "pico_node" trên raspberry pico subsribe dữ liệu từ topic /cmd_vel, sử dụng giá trị của linear.x là vận tộc tuyến tính và angular.z là vận tốc góc để điều khiển xe từ xa.

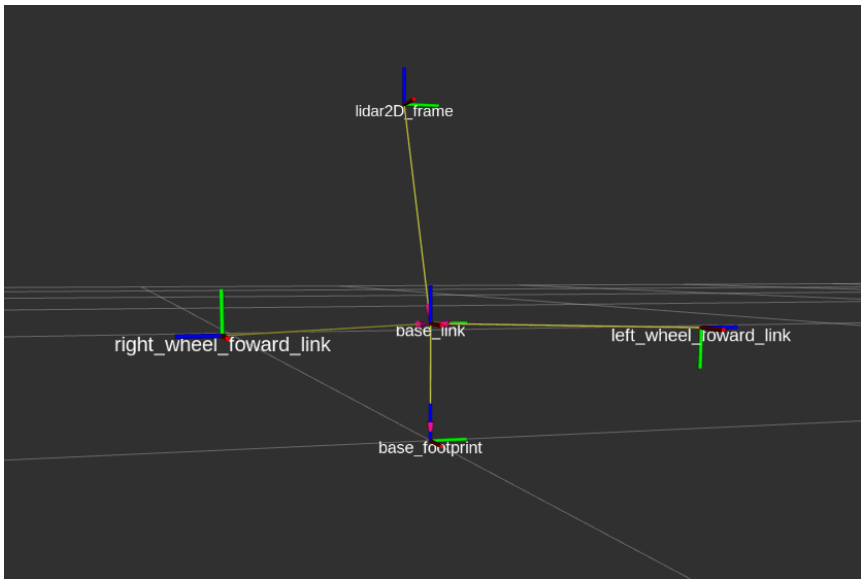
Rqt_graph:



Hình 57 rqt graph điều khiển robot từ xa

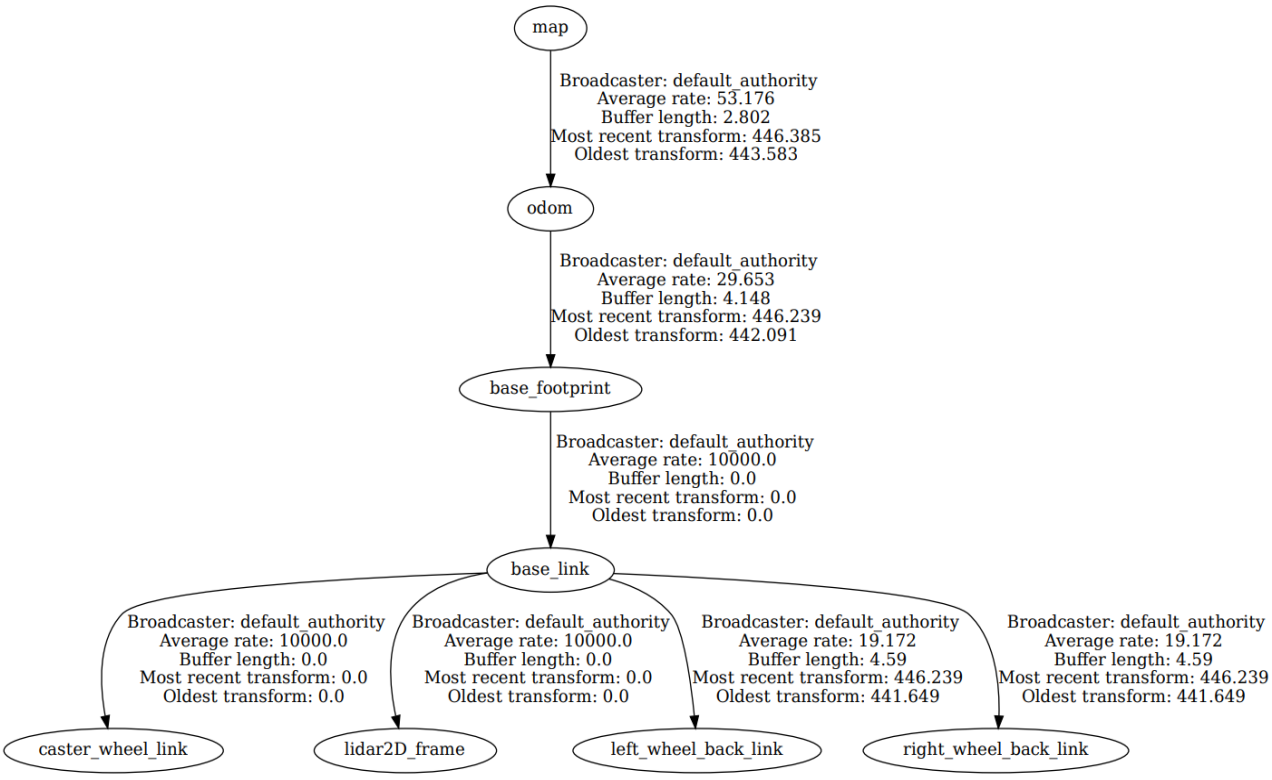
3.6. Xây dựng Software

Xây dựng file URDF và gán các trục tọa độ cho xe.



Hình 58 Các trục tọa độ được gắn trên robot

View tf:



Hình 59 Tf view của hệ thống khi robot mapping

Trong đó:

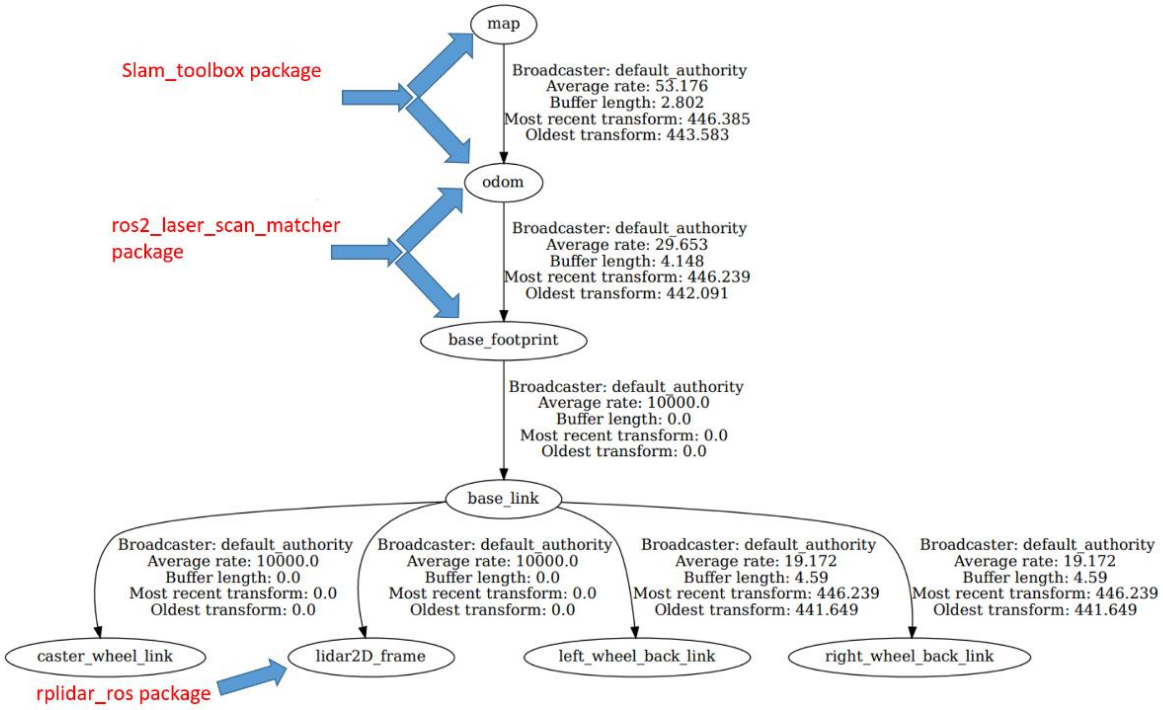
- lidar2d_frame: gốc tọa độ của lidar trên robot.
- base_link: gốc tọa độ gắn trên xe
- base_footprint: gốc tọa độ của base_link trên mềm phẳng robot di chuyển
- map: gốc tọa độ của không gian.
- Odom: gốc tọa độ được tính so với base_footprint khi dùng thuật toán scan_matching.

Sử dung rplidar_ros package đọc dữ liệu từ lidar, khi chạy package cho ra topic /scan chứa các giá trị quét của lidar.

Thuật toán scan_matching được triển khai trong ros2_laser_scan_matcher package dùng thuật toán PL-ICP cho ta được giá trị vị trí và hướng của robot của base_footprint trong hệ tọa độ odom.

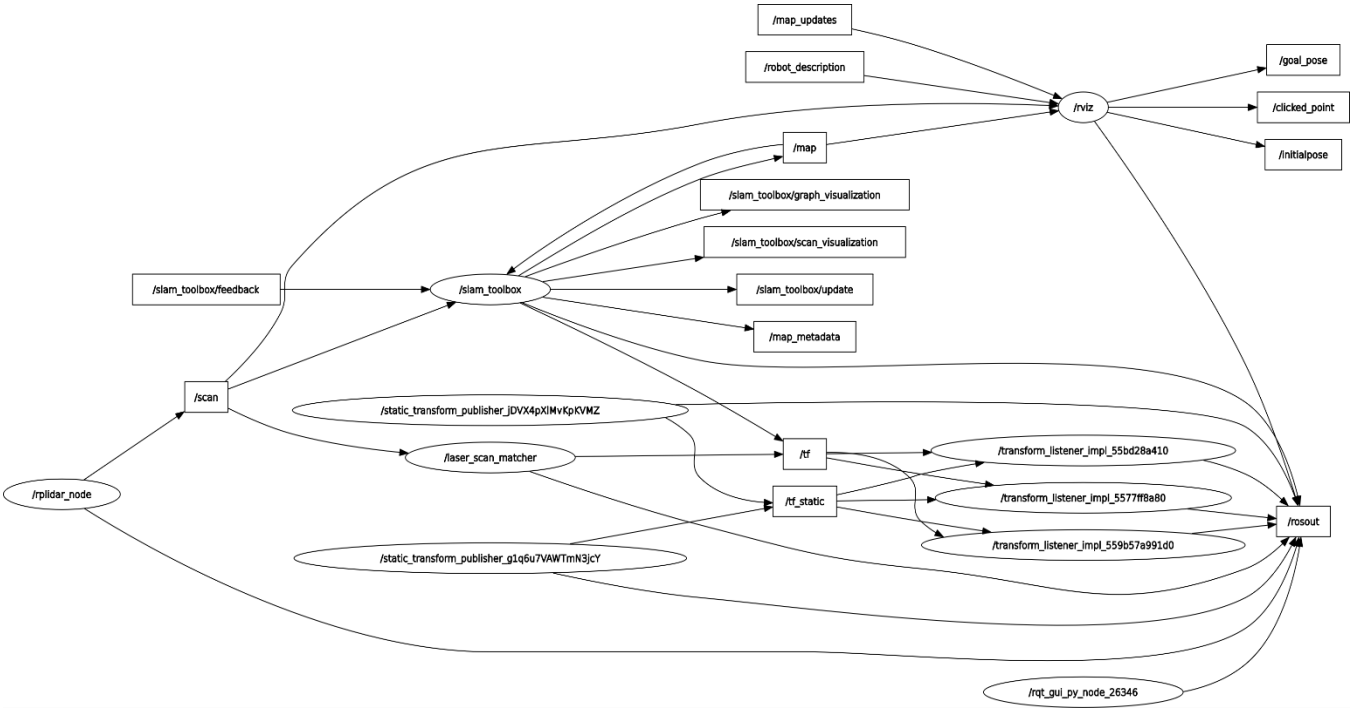
Chạy package slam_toolbox với các thông số bên dưới cho ra được giá trị của odom trong tọa độ map, và từ đó giúp mapping được không gian xung quanh

```
slam_toolbox:
  ros__parameters:
    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None
```



Hình 60 Chạy các package để tạo ra liên kết giữa các tạo độ hệ thống

Rqt_graph:



Hình 61 rqt graph của robot trong quá trình mapping

CHƯƠNG 4. KẾT QUẢ THỰC HIỆN

4.1. Kết quả xây dựng mô hình robot

4.1.1. Mô hình robot thực tế

Sau khi thực hiện vẽ mô hình robot trên SolicWorkd, nhóm tiến hành cắt mica và lắp ráp. Kết quả robto thu được như hình dưới đây, gồm 3 tầng như đã thiết kế ở mục 3.4.

Hình 62. Mô hình robot thực tế

4.1.2. Thông số mô hình và nhận xét

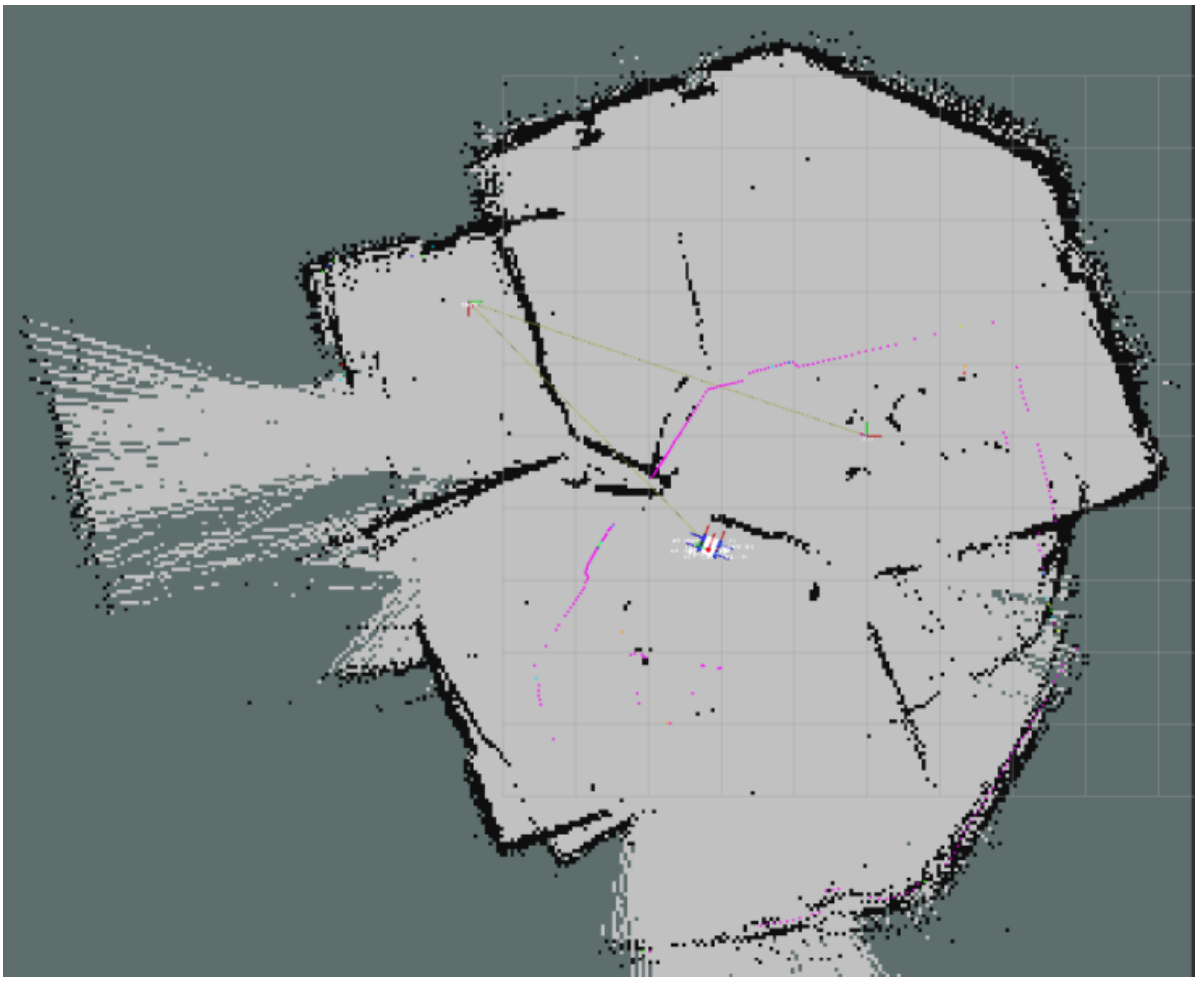
Loại thông số	Thiết kế (đơn vị)	Thực tế (đơn vị)
Chiều dài	25 cm	25 cm
Chiều rộng	18 cm	18 cm
Chiều cao	21 cm	20.8 cm
Cân nặng		
Bán kính bánh xe	6.25 cm	6.33 cm

Bảng 4. Thông số của mô hình robot

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

4.2. Xây dựng không gian môi trường

Kết quả mapping được quan sát qua phần mềm Rviz:



Hình 63 Kết quả mapping một căn phòng tự học

Nhận xét:

Từ kết quả mapping ta thấy giá trị tọa độ odom di chuyển quá xa so với giá trị tọa độ map, điều này xuất phát từ việc chọn trường hợp lý tưởng không có nhiễu, không chọn loss function trong slam_toolbox package (`ceres_loss_function: None`).

Việc chỉ sử dụng dữ liệu lidar để định vị robot và không có thêm các cảm biến khác để đối chiếu làm cho kết quả định vị không được chắc chắn và tối ưu.

Ngoài ra trong quá trình mapping do điều khiển xe di chuyển quá nhanh dẫn tới laser_scan_matching không cho ra được output kịp thời với tốc độ của xe.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Nhóm đã thực hiện các mục tiêu đã đề ra gồm quá trình thiết kế và xây dựng phần cứng, lập trình Firmware, xây dựng phần mềm trên nền tảng ROS, phát triển tác vụ mapping. Trong quá trình nghiên cứu thực hiện của mình, nhóm đã được các kết quả mong muốn cũng như các hạn chế gặp phải, từng bước tìm cách khắc phục và đã hoàn thành bước đầu của đề tài luận văn với các mục tiêu đề ra ban đầu. Sau khi hoàn thành đồ án 2, nhóm đã rút ra một số kết luận và kết quả như sau.

Về thiết kế phần cứng, nhóm lựa chọn thiết kế mô hình 4 bánh xe giúp robot có thể di chuyển linh hoạt và ổn định cho môi trường trong nhà. Nhóm đã sử dụng máy tính nhúng để tối ưu hóa không gian của robot, sắp xếp các linh kiện cảm biến một cách tối ưu và tận dụng tối đa hai mặt của mỗi tầng robot. Tuy nhiên, thiết kế này gặp phải một số hạn chế như việc đi dây kết nối trong robot giữa các tần là phức tạp. Đồng thời robot chạy bằng pin nên thời gian sử dụng robot bị hạn chế cần phải sạc robot thường xuyên.

Thuật toán scan matching dùng PL-ICP cho ta kết quả định vị robot còn sai số nhiều, chưa giống với lý thuyết đã đặt ra và đồ án 2 nhóm chưa đánh giá được kết quả sai số này.

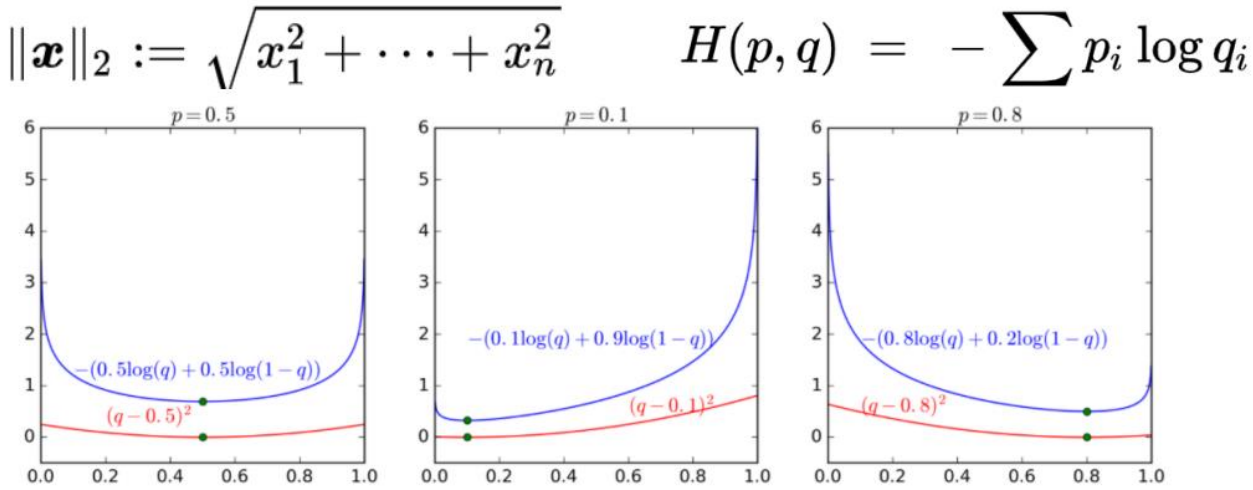
5.2. Hướng phát triển

Kết hợp các cảm biến encoder và IMU với giá trị đầu ra thuật toán scan matching dùng PL-ICP để tối thiểu sai số trong việc định vị vị trí của robot trong không gian.

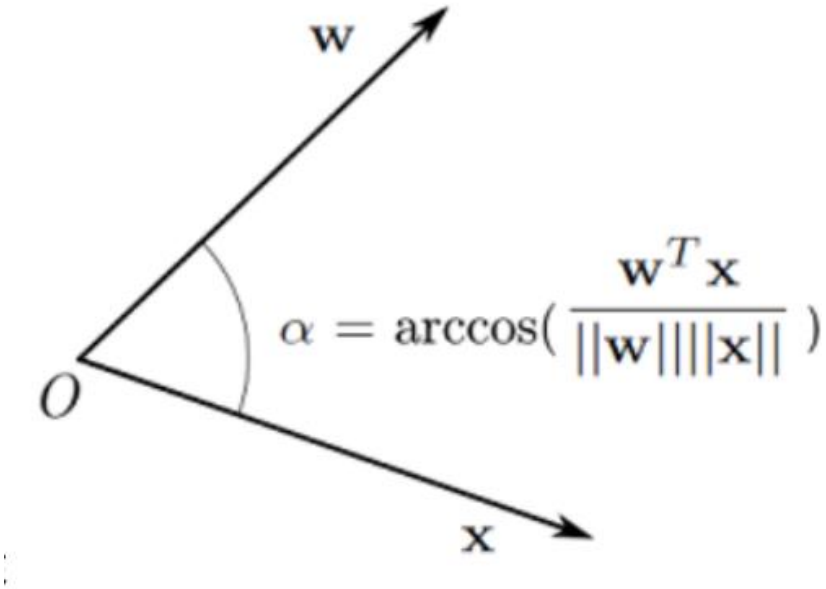
Thuật toán scan matching hiện tại nhóm đang dùng sử dụng hàm tối ưu là 2-norm $\| x \|_2 = \sqrt{\sum_i^m \Delta x_i^2}$ cho kết quả không thực sự được tốt với robot hiện tại và không đáp ứng được tốc độ mong muốn. Do đó, nhóm mong muốn nghiên cứu, thử nghiệm và đánh giá trong trường hợp dùng hàm tối ưu khác là: angle between 2

vectors và cross entropy, để từ đó có thể đánh giá, so sánh được sai số, số vòng lặp và thời gian xử lý dữ liệu với các hàm tối ưu khác nhau trên.

Cross entropy: $H(q,p) = -\sum_i p_i \log q_i$



Hình 64. So sánh đáp ứng giữa hai loss funtion: 2-norm và cross entropy[4]



Hình 65. Angle between 2 vectors[5]

[4] Pham Viet Cuong - Dept. Control Engineering & Automation, Artificial Neural Network lecture, FEEE, Ho Chi Minh City University of Technology

[5] Pham Viet Cuong - Dept. Control Engineering & Automation, Artificial Neural Network lecture, FEEE, Ho Chi Minh City University of Technology

TÀI LIỆU THAM KHẢO

[1] Tài liệu thí nghiệm môn Cơ sở Điều khiển tự động, khoa Điện-Điện tử, trường Đại học Bách Khoa Thành phố Hồ Chí Minh.

[2] Lin Xu, Jiaqiang Du, Baoye Song & Maoyong Cao, "A combined backstepping and fractional-order PID controller to trajectory tracking of mobile robots", Systems Science & Control Engineering, 2022.

[3] Andrea Censi, "An ICP variant using a point-to-line metric", in 2008 IEEE International Conference on Robotics and Automation**,** 2008.

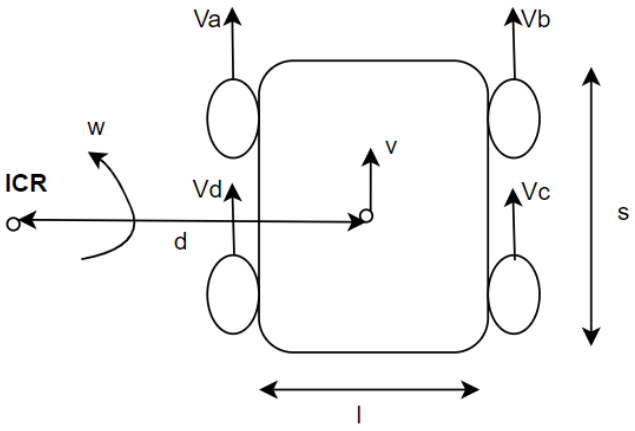
[4] Wei Guan, WenTao Li Yan Ren, "Point Cloud Registration Based on Improved ICP Algorithm", in 2018 Chinese Control And Decision Conference (CCDC)**,** 2018.

[5] Micro-ROS architecture, đường dẫn: https://micro.ros.org/docs/overview/features/

[6] Pham Viet Cuong - Dept. Control Engineering & Automation, Artificial Neural Network lecture, FEEE, Ho Chi Minh City University of Technology

PHỤ LỤC

Differential Drive

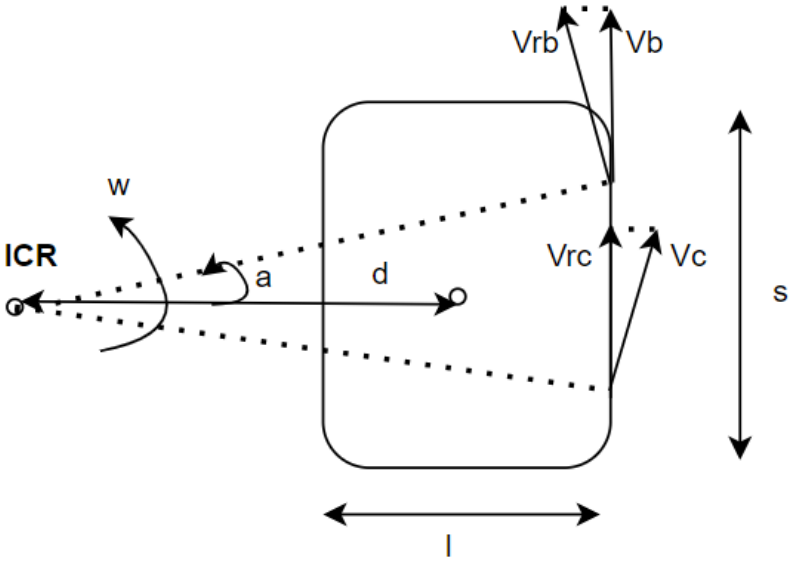


Hình 66. Mô hình Differential Drive

Ta có:

$$v = d\omega$$

$$\rightarrow d = \frac{v}{\omega}$$



Do tính đối xứng của khung robot nên ta có:

$$v_{rc} = v_{rb} = v_r$$
$$v_c = v_b = v_{right}$$
$$v_{right} = v_r \cos a$$

Ta có:

$$v_r = \omega\sqrt{(d + \frac{l}{2})^2 + (\frac{s}{2})^2}$$

$$\cos a = \frac{1}{\sqrt{1 + \tan^2 a}}, \tan a = \frac{\frac{s}{2}}{d + \frac{l}{2}}$$

Từ các phương trình trên ta được:

$$v_{right} = \frac{2v + \omega l}{2}$$

Chứng minh tương tự cho trường hợp 2 bánh xe bên trái ta được:

$$v_{left} = \frac{2v - \omega l}{2}$$

Vận tốc tối đa của robot

Ta có tốc độ tối đa của motor khi có tải là 250 RPM. Bán kính bánh xe là 6.25cm, chiều rộng của robot là 24.69 cm và nhóm cho robot chạy với vận tốc tối đa là 70% vận tốc tối đa mà motor cho phép.

Ta được vận tốc tối đa của robot là 1.15 m/s

$$\frac{0.7 * 250 * 2\pi * 0.0625}{60} = 1.15$$

Từ công thức của mô hình Differential Drive ta được vận tốc xoay tối đa của robot là 9.32 rad/s

$$\frac{1.15 * 2}{0.2469} = 9.32$$