ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

HUỲNH NGỌC HIỆP Ý

TRỊNH ĐĂNG DƯƠNG

KHOÁ LUẬN TỐT NGHIỆP

GIẢI THUẬT SLAM TRONG KHÔNG GIAN 3D ỨNG DỤNG LÊN ROBOT TỰ HÀNH LEO THANG

Research on 3D Mapping for SLAM applying to the autonomous stair-climbing robot

KỸ SƯ KỸ THUẬT MÁY TÍNH

TP. HỒ CHÍ MINH, 2023

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

HUỲNH NGỌC HIỆP Ý - 19522553

TRỊNH ĐĂNG DƯƠNG - 19521414

KHOÁ LUẬN TỐT NGHIỆP

GIẢI THUẬT SLAM TRONG KHÔNG GIAN 3D ỨNG DỤNG LÊN ROBOT TỰ HÀNH LEO THANG

Research on 3D Mapping for SLAM applying to the autonomous stair-climbing robot

KỸ SƯ KỸ THUẬT MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN

ThS. PHẠM MINH QUÂN

TP. HỒ CHÍ MINH, 2023

THÔNG TIN HỘI ĐỒNG CHẤM KHOÁ LUẬN TỐT NGHIỆP

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số 731/QĐ-ĐHCNTT ngày 20 tháng 07 năm 2023 của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thiện khoá luận tốt nghiệp, chúng em cảm thấy rất may mắn khi nhận được nhiều sự quan tâm, giúp đỡ từ quý thầy cô, các anh chị khoá trên, gia đình và bạn bè.

Trước tiên, chúng em xin gửi lời cảm ơn chân thành đến quý thầy cô Khoa Kỹ thuật Máy tính, Trường Đại học Công nghệ Thông tin đã tạo môi trường học tập và rèn luyện rất tốt, cung cấp cho chúng em những kiến thức quý báu và kĩ năng bổ ích trong suốt thời gian học tập tại trường.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới ThS. Phạm Minh Quân, người đã tận tình hướng dẫn, truyền đạt kiến thức và kinh nghiệm cho chúng em trong suốt quá trình nghiên cứu và thực hiện đề tài.

Sau cùng, chúng em xin gửi lời cảm ơn đến gia đình, bạn bè, đặc biệt là bác Huỳnh Thanh Hiệp, anh Bùi Hoàng Kha và anh Nguyễn Bá Văn đã luôn động viên, giúp đỡ, tạo điều kiện tốt nhất để chúng em có thể nỗ lực hoàn thành tốt bài nghiên cứu.

Một lần nữa, chúng em xin chân thành cảm ơn.

TP. Hồ Chí Minh, ngày 04 tháng 08 năm 2023

Nhóm sinh viên thực hiện

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
RTAB-MAP	Real-Time Appearance-Based Mapping
LIDAR	Light Detection And Ranging
DC	Direct Current

TÓM TẮT KHOÁ LUẬN

Bản báo cáo trình bày giải pháp thiết kế hệ thống định vị và tái tạo bản đồ cho robot tự hành sử dụng hệ điều hành robot (ROS) và ứng dụng thuật toán SLAM. Hệ thống định vị và tái tạo bản đồ sử dụng dữ liệu thu được từ các cảm biến camera, cảm biến quét laser và wheel encoder được tổng hợp qua thuật toán RTAB-MAP.

Thuật ngữ SLAM hay "Simultaneous Localization And Mapping" là công nghệ tọa độ hóa và tái tạo môi trường cho robot tự hành. Thuật toán được chọn để hiện thực SLAM là RTAB-Map, cách tiếp cận dựa trên đồ thị RGB-D và sử dụng trình phát hiện vòng lặp (Loop Closure Detector) để xác định khả năng một hình ảnh mới đến từ một vị trí trước đó hay vị trí mới thông qua phương pháp tiếp cận từ nhiều điểm.

Thiết kế robot phù hợp để leo cầu thang là mô hình 6 bánh. Động cơ sử dụng cho robot được ước tính là loại động cơ một chiều có công suất tối thiểu 25W. Nguồn năng lượng sử dụng cho robot là pin LiPo 12V với tổng dung lượng 10400mAh. Một máy tính Ubuntu được thiết lập môi trường làm việc tương tự như Jetson TX1 thực hiện tính toán vị trí toạ độ và vẽ bản đồ, trong khi Arduino UNO sẽ điều khiển động cơ bánh xe và đọc giá trị cảm biến encoder gửi về cho máy tính.

Mô hình robot được chia làm 2 phần. Phần chân robot cao ~520mm, các chân robot được đặt nghiêng một góc 45 độ so với phương thẳng đứng. Phần thân là bộ khung cố định vi điều khiển và các cảm biến có kích thước là 250×240×140mm.

Thiết lập RTAB-Map lên mô hình robot với 3 loại cảm biến camera, cảm biến quét laser 2D và cảm biến tốc độ encoder. Hệ thống được điều khiển bởi máy tính được cài đặt hệ điều hành ROS Melodic Morenia chạy trên nền hệ điều hành Ubuntu Bionic (18.04). Trường hợp kiểm thử chương trình có thể sử dụng laptop có cài đặt môi trường tương đương để thực hiện thuật toán.

Thời gian di chuyển trung bình của robot là 1 km/h, đáp ứng tốt cho nhu cầu sử dụng để thực hiện tái tạo môi trường (mapping) và bản địa hóa (localization).

LỜI MỞ ĐẦU

Tính từ thời điểm máy tính đầu tiên được ra đời cho đến nay, thế giới đã chứng kiến sự thống trị vượt trội của công nghệ thông tin, với sự có mặt của nó ở mọi lĩnh vực đời sống. Các sản phẩm thông minh đáp ứng nhu cầu của con người đang trở thành đối tượng được nghiên cứu nhiều nhất và được coi là trung tâm của cuộc cách mạng công nghệ hiện nay. Trong số các sản phẩm đó, robot dịch vụ đang nổi lên là một trong những sản phẩm có nhiều tiềm năng phát triển nhất và hoàn toàn có khả năng thương mại hóa. Rất nhiều loại robot dịch vụ đã được ra đời và trở nên phổ biến như robot lau nhà, robot hướng dẫn tại các nơi công cộng như bệnh viện, sân bay, nhà ga, trung tâm thương mại, … Các robot này đều hoạt động chủ yếu trong môi trường có diện tích giới hạn và có nhiều chướng ngại vật, do đó việc xác định chính xác vị trí tương đối của robot so với các đồ vật xung quanh sẽ làm giảm khả năng gây ra các va chạm khi robot di chuyển.

Trên thị trường hiện đã có rất nhiều sản phẩm robot được nghiên cứu và áp dụng nhiều giải thuật và phương pháp khác nhau trong việc dẫn đường robot trong không gian trong nhà. Bên cạnh những thuật toán và phương pháp thực hiện đa dạng, nhiều loại cảm biến khác nhau đến từ nhiều nhà cung cấp cũng được sử dụng để tạo ra các sản phẩm robot phục vụ cho nhiều mục đích khác nhau. Tuy nhiên, do đặc tính của từng loại cảm biến, các hệ thống robot thường sử dụng các loại cảm biến mang tính đặc thù để đảm bảo yếu tố về hình dạng, kiểu dáng đặc trưng. Vì vậy, tính mở rộng của hệ thống cũng là bài toán cần được quan tâm.

Mục tiêu của đề tài là nghiên cứu, phát triển một hệ thống định vị trong bản đồ 3D và thực hiện nhiệm vụ leo cầu thang. Các mục tiêu được đặt ra bao gồm: sử dụng hệ điều hành ROS thu nhận dữ liệu từ bất kỳ loại cảm biến laser 2D và camera 3D nào, sử dụng thuật toán RTAB-Map để xác định vị trí tương đối của robot so với vị trí gốc khi robot di chuyển. Kết quả của đề tài sẽ được sử dụng để phát triển robot dịch vụ có khả năng di chuyển trong các toà nhà cao tầng, ứng dụng trong robot dọn dẹp, robot dẫn đường, robot vận chuyển hàng …

CHƯƠNG 1. **TỔNG QUAN**

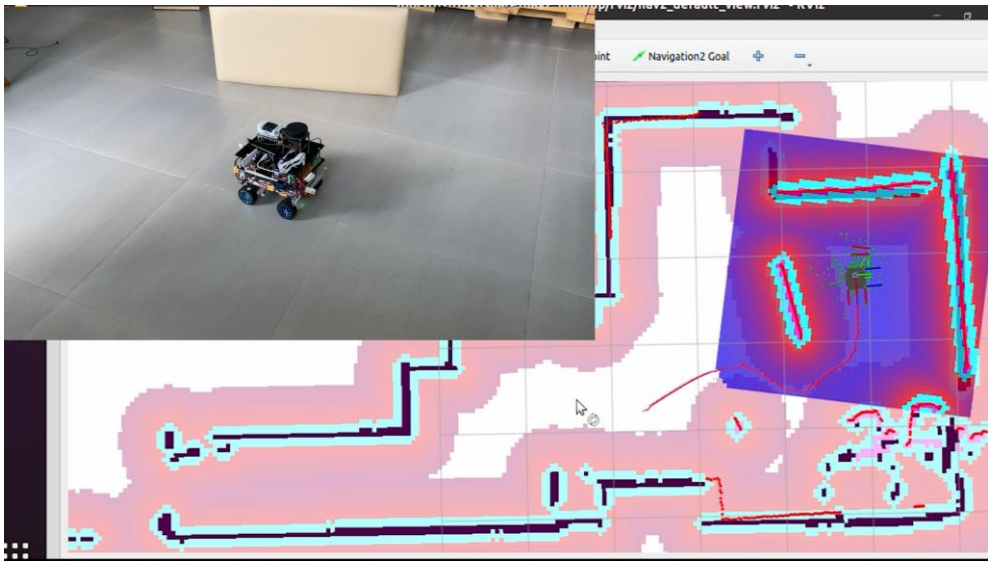
Phần Tổng quan sẽ phân tích, đánh giá các hướng nghiên cứu đã có của các tác giả trong và ngoài nước liên quan đến đề tài, từ đó chỉ ra những vấn đề mà đề tài cần tập trung nghiên cứu giải quyết.

1.1. Giới thiệu các nghiên cứu có liên quan

1.1.1. Nghiên cứu tích hợp ROS2 và RTOS cho robot tự hành

Đây là đề tài khoá luận tốt nghiệp của Phùng Văn Hảo và Phan Hữu Đạt, sinh viên khoa Kỹ thuật Máy tính, trường Đại học Công nghệ thông tin.

Hệ thống sử dụng kit Jetson TX2 với hệ điều hành ROS2 và STM32 sử dụng hệ điều hành RTOS tích hợp cùng với Micro-Ros.



Hình 1.1 Robot định vị và tìm đường đi trên bản đồ 2D đã tạo

Robot tái tạo môi trường và đinh vị trên bản đồ 2D bằng dữ liệu thu được từ cảm biến laser 2D và thực hiện nhiệm vụ di chuyển đến vị trí được chỉ định trên bản đồ, đồng thời đảm bảo né tránh các vật cản cố định hoặc di động.

Robot có thể di chuyển và xoay trên bề mặt phẳng. Điểm hạn chế là robot sẽ gặp khó khăn khi chạy trên bề mặt ghồ ghề.

1.1.2. Các sản phẩm robot có khả năng leo trèo trên bậc thang

Sau đây là 3 loại hình robot đáp ứng tốt nhu cầu leo trèo trên bậc thang:

- Mô hình “Legged Based Robot” (tạm dịch: robot có chân)
- Mô hình “Six-Wheel Robot” (tạm dịch: robot sáu bánh xe)
- Mô hình “Track Based Robot” (tạm dịch: robot với bánh xe xích)

Ví dụ cho mô hình Legged Based Robot là Spot đến từ Boston Dynamics.

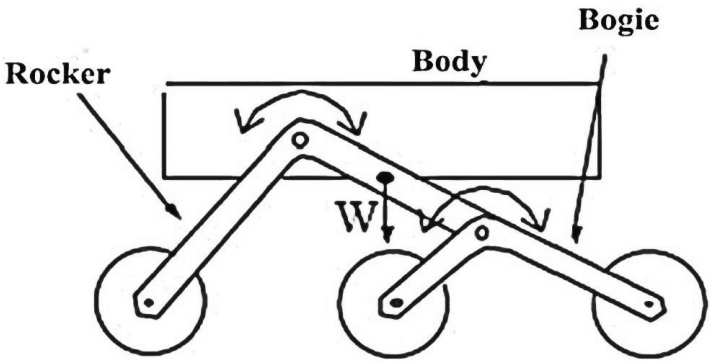
Hình 1.2 Robot Spot của Boston Dynamics

Robot Spot được hãng Boston Dynamics giới thiệu với mục tiêu chính là hỗ trợ con người trong ngành xe hơi. Spot có trọng lượng 32 kg, có khả năng mang vật nặng lên đến 14 kg và tốc độ trung bình khi di chuyển đạt khoảng 5-6 km/h.

Thiết kế của Spot trông giống một chú chó với 4 chân, phần thân và bộ điều khiển nằm phía trước. Đặc biệt, Spot không tạo tiếng ồn khi di chuyển bởi các đệm cao su được bố trí ở phần chân tiếp xúc và các khớp truyền động.

Phần thân của Spot có mang cảm biến Lidar để quét môi trường xung quanh và nhận diện vật thể. Khi gắn thêm camera 360^0, Spot có thể quan sát môi trường, hỗ trợ cho tính năng dự báo thời tiết.

Ví dụ cho mô hình Six-Wheel Robot là Rocker Bogie Rover đến từ IJSR[1] - Tạp chí Khoa học và Nghiên cứu Quốc tế.

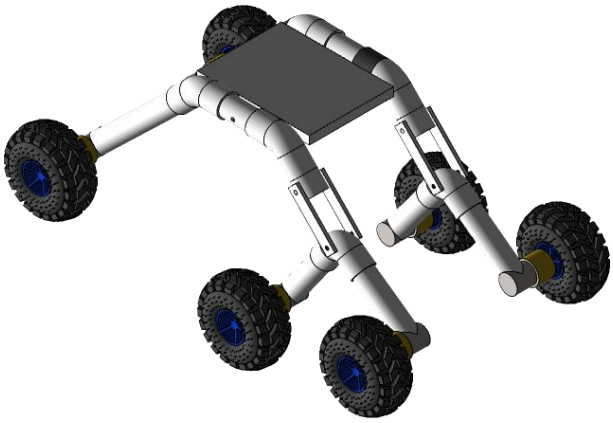


Hình 1.3 Cấu tạo chân của Rocker Bogie Rover

Điểm nổi bật của robot này nằm ở chân và bánh xe của robot. Phần bánh xe với chất liệu cao su có độ ma sát cao. Động cơ được thiết kế độc lập cho mỗi bánh xe. Bánh xe phía sau (Rear wheel) được kết nối vào một thanh nối "Rocker" có hình dáng chữ V và vì được cố định trên thân xe (Body) tại điểm gấp khúc nên thanh nối này mới có tên gọi là Rocker (nghĩa là bập bênh). Đầu còn lại của thanh bập bênh "Rocker" được nối vào một khớp xoay để kết nối với một thanh nối khác có tên gọi "Bogie" (nghĩa là giá chuyển hướng). Bogie được kết nối với cặp bánh xe trước (Middle wheel và Front wheel) có nhiệm vụ là các bánh lái của robot.

Cấu trúc này được sử dụng khá phổ biến trên hệ thống treo (suspension system) của xe tải sơ mi rơ moóc (semi-trailer truck).

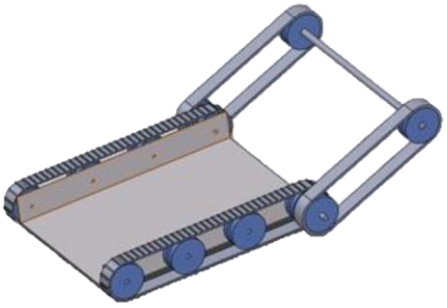
[1] International Journal of Science and Research (www.ijsr.net)



Hình 1.4 Mô hình 3D của Rocker Bogie Rover

Rocker Bogie có thể chịu được độ nghiêng lên đến 50^0 theo bất kỳ hướng nào mà không bị lật. Ngoài ra nhờ khung gầm thoáng nên nó có thể vượt qua bất kì chướng ngại vật nào có kích thước, độ cao gấp đôi đường kính bánh xe. Với cấu tạo chân và bánh xe như trên, việc thực hiện nhiệm vụ leo bậc thang sẽ dễ dàng do lực kéo cũng như lực đẩy được phân bố đều cho mỗi bánh xe.

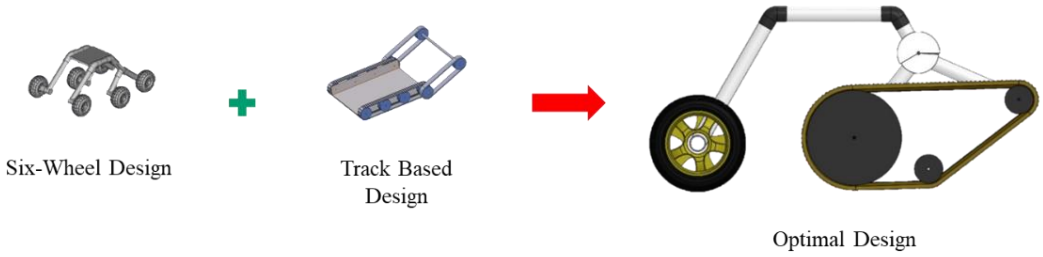
Mô hình robot còn lại là Track Based Robot, sử dụng 2 dây đai cao su để tăng độ bám dính khi bắt đầu leo lên bậc thang, tăng cường lực kéo lên cho toàn bộ robot; chuỗi bánh xe ở phía sau có nhiệm vụ làm lực đẩy robot về phía trước.



Hình 1.5 Mô phỏng mô hình robot bánh xe xích

Thiết kế này được ứng dụng trong rất nhiều loại xe cơ giới với phần bánh xe xích được thiết kế nâng lên 2 đầu để xe có thể dễ dàng leo trèo trên những địa hình có nhiều sự chênh lệch về độ cao hoặc có độ dốc lớn.

Để tăng độ bám mặt đường khi di chuyển trên bề mặt ghồ ghề, có độ dốc lớn hoặc bề mặt sụt lún, có thể kết hợp thiết kế 6 bánh xe và bánh xe xích như hình bên dưới [1].



Hình 1.6 Mô hình kết hợp thiết kế 6 bánh và bánh xe xích

Xét theo độ phức tạp khi xây dựng mô hình cũng như các vật liệu dễ tìm, đề tài sẽ sử dụng mô hình robot 6 bánh xe để xây dựng robot tự hành leo cầu thang.

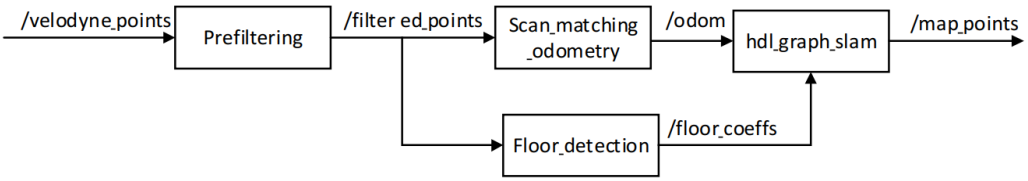
Một điểm lợi thế khi sử dụng mô hình này là giúp tăng độ linh hoạt khi robot di chuyển qua các bề mặt ghồ ghề bởi sự linh hoạt từ khớp xoay của 2 bánh xe trước và lực đẩy từ bánh sau được tận dụng tối đa.

1.1.3. Thuật toán HDL Graph SLAM

HDL Graph SLAM là thuật toán SLAM thời gian thực dành cho laser quét 3D. Nó dựa trên một thuật toán SLAM 3D Graph với một biến đổi phân phối chuẩn[2] là một quá trình hợp nhất kết quả quét để xác định quỹ đạo. Nó phù hợp cho hệ toạ độ 6DOF[3] gồm có 6 trục, trong đó có 3 trục toạ độ tịnh tiến và 3 trục chuyển động xoay. Thêm vào đó để scan-matching (so khớp), các đầu vào cảm biến như IMU hoặc GPS có thể được sử dụng như điều kiện biên cho việc xác định quỹ đạo. [2]

Hình 1.7 mô tả khái quát 4 node cấu thành HDL Graph SLAM, bao gồm node *Prefiltering*, node *Scan_matching_odometry*, node *Floor_detection* và node thực hiện thuật toán chính *hdl_graph_slam*. [3]

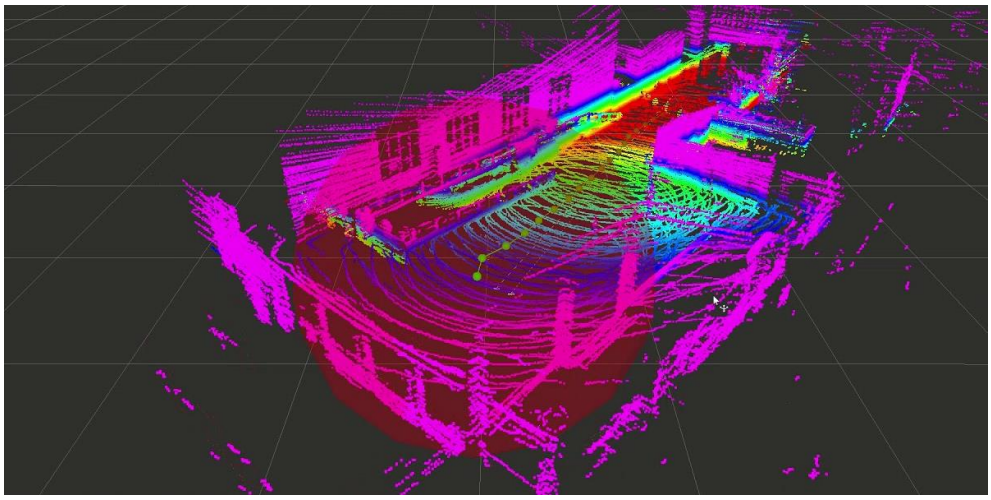
[2] NDT - Normal Distribution Transform
[3] Six Degrees of Freedom - 6DOF



Hình 1.7 Tổng quan các node của hdl_graph_slam

Đầu tiên, đám mây điểm đầu vào được lấy mẫu xuống bằng *Prefiltering*, sau đó được chuyển đến các node tiếp theo. Trong khi *Scan_matching_odometry* ước tính "tư thế" của cảm biến bằng cách lặp lại áp dụng quét khớp giữa các khung liên tiếp (nghĩa là ước tính đo đường), thì *Floor_detection* phát hiện các mặt phẳng sàn bằng RANSAC. Số đo ước tính và mặt phẳng sàn được phát hiện được gửi tới *hdl_graph_slam*. Node này sẽ thực hiện loop-closure detection (phát hiện vòng lặp) để bù cho lỗi tích lũy của quá trình *scan matching* (so khớp) và tối ưu hóa biểu đồ.

Dữ liệu môi trường thu được từ một cảm biến laser 3D với kiểu dữ liệu là point cloud (đám mây điểm). Dữ liệu này được chuyển đến node *Prefiltering*.



Hình 1.8 Dữ liệu point cloud thu được từ cảm biến laser 3D

Trong node *Prefiltering*, dữ liệu quét laser được xử lý trước. Các điểm đo quá gần hoặc quá xa đều bị loại bỏ. Điều này có thể được xác định thông qua các giá trị ngưỡng (threshold value) *distance_near_thresh* và *distance_far_thresh*. Hơn nữa,

các ngoại lệ (the outlires) có thể được loại bỏ khỏi các point cloud trong bước này, thông qua phương pháp bán kính hoặc phương pháp thống kê.

- Phương pháp bán kính (radius method) xem xét số lượng điểm lân cận (mật độ lân cận) của một điểm nằm trong bán kính đã cho. Nếu giá trị thấp hơn số lân cận tối thiểu (radius_min_neighbours) thì điểm sẽ bị xoá.
- Phương pháp thống kê (statistical method): các ngoại lệ có thể được xác định và loại bỏ thông qua độ lệch chuẩn (the standard deviation).

Sau khi xử lý đầu vào ở dạng các point cloud, bộ lọc sẽ được xử lý bởi node *Scan_matching odometry* để ước tính quỹ đạo và để dành cho việc nhận diện mặt phẳng (ground surface detection). Việc sử dụng đầu ra cho cả 2 node, quỹ đạo sau đó được tối ưu hoá trong node *hdl_graph_slam,* dẫn đến bản đồ đầu ra mong muốn (desired output map) bao gồm các điểm 3D đã ghi nhận.

Qua các bước tiền xử lý, scan matching (so khớp) để ước tính tư thế robot hiện tại (bao gồm vị trí toạ độ và góc xoay) và để kết hợp các khung hình liên tiếp để tạo ra một khung ảnh kéo dài, cùng lúc đó phát hiện mặt phẳng sàn.

Scan matching được thực hiện bằng cách sử dụng các điểm đã lọc. Vị trí cảm biến được ước tính bằng cách áp dụng phương pháp *scan matching* từ các khung liên tiếp (successive frames), từ đó quỹ đạo robot (trajectory) được xác định (ước tính đo lường - odometry estimate). Để scan matching, có thể chọn các phương pháp ICP, GICP, NDT và VGICP và có thể định cấu hình (configured) các tham số tương ứng cho các thuật toán.

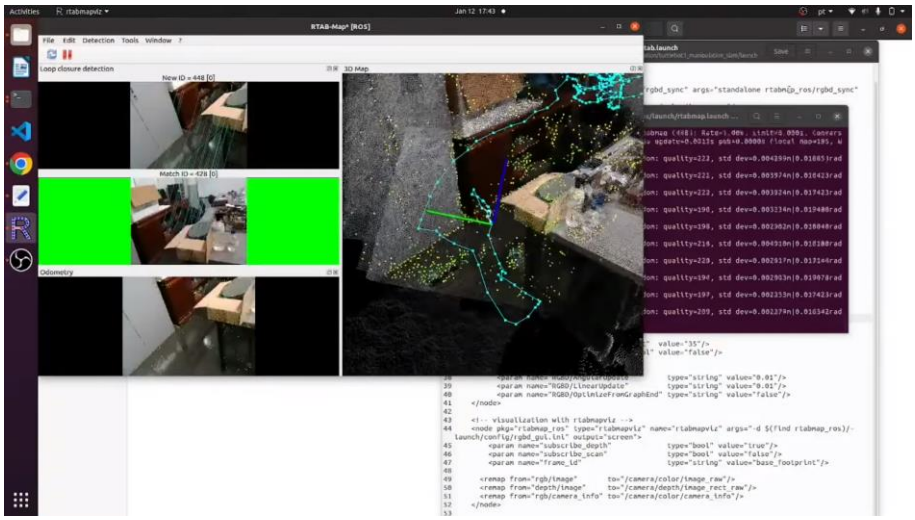
Floor detection (phát hiện mặt sàn) được tiến hành cùng lúc với *scan matching*. Trong các toà nhà có diện tích sàn lớn (trường học, bệnh viện, ...) khu vực xung qunah thường chỉ có bề mặt sàn phẳng. Do đó để tối ưu hoá biểu đồ tư thế (pose graph), một điều kiện biên bổ sung (additional boundary condition) được đưa ra có tính đến diện tích sàn được phát hiện. Với mục đích này, giả định rằng tất cả

các bề mặt sàn được phát hiện trong lần quét tương ứng đều nằm trên cùng một mặt phẳng. Đối với cấu hình, chiều cao cảm biến ước tính phải được chỉ định. Để ước tính mặt phẳng robot di chuyển, tất cả các điểm nằm trong một phạm vi nhất định trên hoặc dưới chiều cao mặt đất đều được sử dụng. Việc ước tính một mặt phẳng được thực hiện bằng RANSAC[4] - đồng thuận mẫu ngẫu nhiên. Cuối cùng thuật toán *loop closure detection* (phát hiện vòng lặp) sẽ tối ưu hoá bản đồ và làm giảm sai số.

Ưu điểm của thuật toán này là đạt được kết quả chính xác, logic của các phần xử lý dễ hiểu và phần lắp đặt dễ dàng. Nhược điểm là giá thành cảm biến khá cao.

1.1.4. RTAB-Map

RTAB-Map là dự án được phát triển bởi IntRoLab, Université De Sherbrooke (Québec, Canada). Đây là một đồ thị RGB-D SLAM dựa trên *loop closure detector* (bộ phát hiện vòng lặp kín), sử dụng phương pháp tiếp cận từ nhiều điểm để xác định hình ảnh mới có khả năng đến từ một vị trí cũ hay không.



Hình 1.9 Thuật toán RTAB-Map với Loop Closure Detector

[4] Random Sample Consensus - Đồng thuận mẫu ngẫu nhiên

Dữ liệu thu thập từ camera 3D và cảm biến quét laser 2D (RP-LiDAR) dùng để tạo bản đồ. Ngoài ra dữ liệu từ RP-LiDAR còn được dùng để so khớp nhằm mục đích ước tính bước di chuyển thay vì sử dụng IMU.

Ưu điểm: linh hoạt với nhiều lựa chọn cảm biến, có thể kết hợp nhiều loại cảm biến khác nhau nên có thể lựa chọn các cảm biến với giá phải chăng.

Ví dụ:

- Trường hợp có đầy đủ cảm biến (camera, odometry[5] và 2D laser): thuật toán được thực hiện đầy đủ các node và các dữ liệu đầu vào được lấy trực tiếp từ cảm biến tương ứng mà không cần thông qua bước chuyển đổi trung gian nào từ bất cứ một dữ liệu đầu vào nào khác.
- Trường hợp thiếu 2D laser: mô phỏng kết quả quét laser với hình ảnh độ sâu (depth image) bằng package *depthimage_to_laserscan*, sau đó thuật toán được thực hiện như trường hợp đầy đủ.
- Trường hợp thiếu odometry: tạo odometry bằng dữ liệu từ cảm biến quét laser 2D và thuật toán ICP[6], có thể sử dụng package *laser_scan_matcher* hoặc *hector_slam*. Khi đó một node *icp_odometry* sẽ được sinh ra bên trong sơ đồ node của thuật toán.
- Trường hợp thiếu camera: khi không có camera, rtabmap sẽ mất khả năng *loop closure detection* bằng hình ảnh. Khi đó phải build workspace của rtabmap với phụ thuộc (dependence) *libpointmatcher*.

Nhược điểm: thuật toán phức tạp, cần phải cấu hình để các node giao tiếp với nhau đúng cách. Bên cạnh đó, thuật toán này còn đang phát triển nên chưa có đầy đủ tài liệu chi tiết về thuật toán này từ tác giả.

[5] Odometry là kỹ thuật được sử dụng trong robot để xác định vị trí trong môi trường xung quanh bằng cách sử dụng dữ liệu từ các cảm biến chuyển động (motion sensors) để ước tính sự thay đổi vị trí theo thời gian.
[6] ICP (Iterative Closest Point) là thuật toán được ứng dụng nhiều trong việc đạt được một sự chênh lệch nhỏ nhất giữa hai đám mây điểm (point cloud).

1.2. Mục tiêu và giới hạn đề tài

1.2.1. Mục tiêu

Kết quả cần đạt được: Đặt robot trên bề mặt sàn nhà và thực hiện thuật toán SLAM bên trong căn phòng và thực hiện di chuyển đến vị trí được chỉ định.

Các mục tiêu được đặt ra bao gồm:

- Sử dụng hệ điều hành ROS (Robot Operating System) thu nhận dữ liệu từ cảm biến laser 2D và camera 3D.
- Sử dụng thuật toán RTAB-Map để xác định vị trí tương đối của Robot so với vị trí gốc khi robot di chuyển.
- Thực hiện di chuyển đến vị trí được chỉ định và thực hiện nhiệm vụ leo trèo trên các bậc thang từ chân cầu thang.

1.2.2. Giới hạn đề tài

Các giới hạn cho đề tài như sau:

- Không có khả năng nhảy (qua vật cản) hoặc bay.
- Tốc độ di chuyển trung bình 1 km/h, tốc độ tối đa là 5 km/h.
- Tốc độ khung hình cho SLAM: tối thiểu 15 fps.
- Sai số (khi dùng bộ lọc): tối đa 0.1m về trị trí và tối đa 1.0^0 về hướng.
- Robot tự quyết định hướng đi dựa trên các cảm biến và vị trí chỉ định.
- Không có bất cứ sự can thiệp từ con người dưới bất kỳ hình thức nào sau khi robot bắt đầu nhiệm vụ di chuyển hay leo lên các bậc thang.
- Robot phải mang theo nguồn năng lượng riêng (tổng dung lượng pin 10400mAh với thời gian sử dụng liên tục ước tính khoảng 15 phút).
- Khi thực hiện leo thang, robot phải bắt đầu từ chân cầu thang.
- Robot được thiết kế để leo tối thiểu 1 tầng lầu (tương đương 10 bậc thang có kích cỡ như các bậc thang ở sân tòa nhà B thuộc Trường Đại học Công nghệ thông tin, ĐHQG TP.HCM).

CHƯƠNG 2. **CƠ SỞ LÝ THUYẾT**

Phần này trình bày cơ sở lí thuyết, lí luận, giả thiết khoa học và phương pháp nghiên cứu đã được sử dụng trong khoá luận.

2.1. Hệ điều hành ROS

Hệ điều hành robot – ROS là một hệ điều hành mã nguồn mở, thu hút được sự quan tâm và đóng góp của cộng đồng trên khắp thế giới để xây dựng & phát triển các dự án robotics. Cho đến hiện tại đã có rất nhiều phiên bản được phát hành dưới dạng bản phân phối (ROS Distribution) cùng với các công cụ và thư viện hữu ích được cập nhật thường xuyên. Khi cài đặt ROS cần chọn bản phân phối phù hợp với nền tảng hệ điều hành hiện tại (target platform).

Hệ điều hành robot (ROS) đã được phát triển và cập nhật trong nhiều năm qua [4], cho phép trừu tượng hoá phần cứng và trình điều khiển thiết bị (driver) giúp việc thiết kế phần mềm trên robot trở nên dễ dàng hơn. ROS cung cấp cho lập trình viên đa dạng các công cụ và thư viện hữu ích cho việc xây dựng hệ thống, lập trình, biên dịch, chạy chương trình, thực hiện giao tiếp giữa hai và nhiều máy tính, … Ngôn ngữ lập trình được sử dụng phổ biến là C và Python. Hệ thống chương trình của ROS được tổ chức thành các node hoạt động đồng thời cùng cơ chế trao đổi dữ liệu giữa các node. Đặc biệt, phần mã nguồn chương trình của các node có thể được viết bằng các ngôn ngữ lập trình khác nhau, cho phép linh hoạt trong việc phát triển từng ứng dụng khác nhau trên ROS. [5]

2.1.1. Các khái niệm trong ROS

Master

Có thể hiểu Master là một server (máy chủ) có chức năng tạo sự kết nối giữa các chương trình và từ đó tạo ra nhiều phương thức giao tiếp giữa các chương trình với nhau. Câu lệnh để khởi động Master là `roscore`.

Master giao tiếp với các chương trình khác bằng một tính năng truyền dữ liệu có tên là XMLRPC (XML - Remote Procedure Call). Khi thao tác với master, nó được cấu hình với địa chỉ URI và một port (cổng) được cấu hình trong ROS_MASTER_URI. Các giá trị mặc định gồm địa chỉ URI là địa chỉ IP của máy và với port là 11311.

Node

Node là thành phần nhỏ nhất của ROS, nó xuất hiện trong ROS khi một chương trình được thực thi để thực hiện một chức năng hoặc nhiệm vụ nào đó.

Các thông tin cần thiết để xác định một node là tên node, kiểu message, địa chỉ URI, giá trị port. Tùy thuộc vào thông tin mà node được định nghĩa, node có thể đóng vai trò là một publisher, subscriber, service server hoặc service client. Các nodes có thể giao tiếp với nhau bằng cách sử dụng topics và services.

Khi giao tiếp với Master, Node sử dụng XMLRPC. Còn khi giao tiếp với các node khác, Node có thể sử dụng XMLRCP hoặc TCPROS của giao thức TCP/IP.

Package

Package (gói) là một thành phần cơ bản của ROS, có chức năng thi hành các node được chứa bên trong nó. Nội dung của package gồm có các file cấu hình, mã nguồn (source code) của các node, file launch, … Ngoài ra một package còn chứa nhiều file cần thiết khác cho việc thi hành package như các thư viện, các phụ thuộc (dependences), ...

Topic

Topic (chủ đề) là một tên gọi đại diện cho các message (thông điệp) mà các node gửi và nhận với nhau. Bởi vì trong một hệ thống của ROS có rất nhiều node, các node sẽ giao tiếp được với nhau nếu chúng cùng có liên kết đến một topic. Liên kết này có thể là publish (xuất bản, tạo ra thông điệp) hoặc subscribe (theo dõi, lắng nghe thông điệp).

Một số thao tác với topic:

1. rostopic bw: hiển thị băng thông (bandwidth) sử dụng bởi topic
2. rostopic delay: hiển thị độ trễ của topic
3. rostopic echo: in các tin nhắn ra màn hình
4. rostopic find: tìm topic bởi kiểu dữ liệu
5. rostopic hz: hiển thị tỉ lệ (rate) publish của topic
6. rostopic info: in ra thông tin của topic hoạt động
7. rostopic list: liệt kê các topic hoạt động
8. rostopic pub: publish dữ liệu đến topic
9. rostopic type: in ra kiểu dữ liệu của topic

Publisher và subscriber

Publisher là thuật ngữ dùng để chỉ một node đang truyền thông điệp, ngược lại với Subscriber chỉ một node đang lắng nghe thông điệp. Các node truyền nhận thông điệp qua một bước trung gian là Master. Một node vừa có thể truyền thông điệp đến topic này và cũng vừa có thể nhận thông điệp từ topic khác.

Message

Message (thông điệp) là nội dung trao đổi giữa các nodes. Có nhiều kiểu message khác nhau ví dụ như:

- Kiểu dữ liệu chuẩn: int, boolean, String, ...
- Kiểu dữ liệu đặc thù: geometry_msgs/Vector3, sensor_msgs/LaserScan

Service

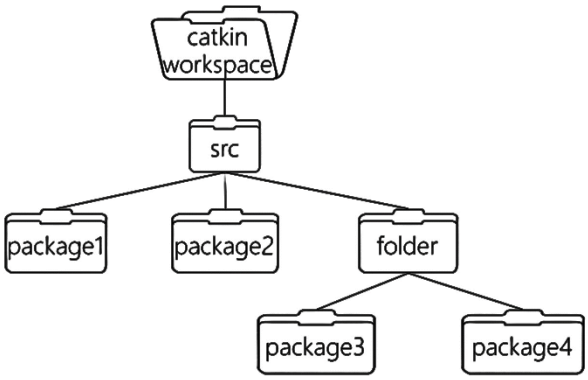
Service là một phương thức giao tiếp đồng bộ 2 chiều giữa client (máy khách) và server (máy chủ). Service của server sẽ tiếp nhận các yêu cầu từ service của client và hồi đáp lại yêu cầu đó, được thi hành trong node server. Ngược lại, service của client sẽ gửi yêu cầu tới server và sau đó nhận hồi đáp.

2.1.2. Một số thao tác cơ bản trên ROS

Tạo catkin workspace

catkin là hệ thống của ROS dùng dể build, hay cụ thể hơn đó là một nhóm các công cụ mà ROS hỗ trợ để tạo ra các chương trình khả thực thi, các thư viện và giao diện mà những ngôn ngữ lập trình khác (như C/C++ hay Python) có thể sử dụng.

Hình 2.1 mô tả cấu trúc cơ bản của một catkin workspace, với các thư mục là các package hoặc là thư mục đơn thuần chứa các package khác.



Hình 2.1 Cấu trúc của một catkin workspace

Để sử dụng **catkin**, đầu tiên cần thiết lập workspace (không gian làm việc).

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws && catkin_make
```

Sau khi đã khởi tạo workspace, lần lượt tạo các package như sau:

```
catkin_create_pkg package1 roscpp
```

Trong đó, `package1` là tên package cần tạo, `roscpp` là các phụ thuộc/thư viện mà package này cần sử dụng.

Tạo node

Di chuyển đến package cần tạo node bằng lệnh `cd`. Một package sẽ bao gồm các tệp/thư mục như CMakeList.txt, package.xml, thư mục include, thư mục script, thư mục src, ... Trong đó thư mục src là nơi chứa mã nguồn viết bằng ngôn ngữ C/C++, còn thư mục script là nơi chứa mã nguồn viết bằng ngôn ngữ Python.

Sử dụng Vim/Atom (hoặc bất cứ công cụ soạn thảo nào) trên Ubuntu để tạo các tệp tin chứa mã nguồn (ngôn ngữ C/C++ hoặc Python). Sau đó build mã nguồn bằng các công cụ gcc/g++/CMake, hoặc có thể trực tiếp sử dụng câu lệnh `catkin_make` để build toàn bộ workspace hoặc tuỳ chọn các package cần build bằng cách thêm các tên package vào câu lệnh như các tham số đầu vào sau tuỳ chọn `--only-pkg-with-deps`.

```
catkin_make --only-pkg-with-deps package1
```

Sau khi build thành công có thể thực thi node bằng lệnh `rosrun`.

Launch

Launch là thao tác thi hành nhiều node cùng lúc, cùng lúc đó có thể thiết lập các tham số, định nghĩa cho một remap, chỉ định biến môi trường, gộp nhóm bằng namespace, ... Tệp tin cấu hình cho thao tác launch có phần tên mở rộng là .launch, viết bằng tệp cấu hình XML.

Công cụ để launch trong ROS là `roslaunch`.

```
roslaunch package_name file.launch
```

Theo dõi kết quả trên Rviz

Rviz là công cụ trực quan hóa dữ liệu 3D dành cho ROS. Người dùng có thể gọi Rviz bằng lệnh `rviz` hoặc launch node rviz từ một file .launch

2.2. Giải thuật SLAM

Thuật ngữ SLAM là viết tắt của “Simultaneous Localization And Mapping”, tạm dịch là công nghệ tọa độ hóa và tái tạo môi trường cho robot tự hành. SLAM xây dựng bản đồ bằng cách thu thập dữ liệu môi trường từ cảm biến được cố định trên robot như cảm biến quét laser, camera, ... Nhiệm vụ định vị cần phải được thực hiện đồng thời với nhiệm vụ xây dựng bản đồ. Khi đó robot kết hợp dữ liệu tốc độ của bánh xe và góc xoay để tính toán bước di chuyển trong một đơn vị thời gian, cùng với đó là so sánh những phần trùng khớp trên những hình ảnh liên tiếp và đo khoảng cách đến các vật thể để tạo bản đồ 3D. Điều này sẽ giúp robot hạn chế những sai số liên quan đến độ trượt của bánh xe cũng như phân biệt được những chi tiết nhỏ giống nhau trên những khung hình liên tiếp.

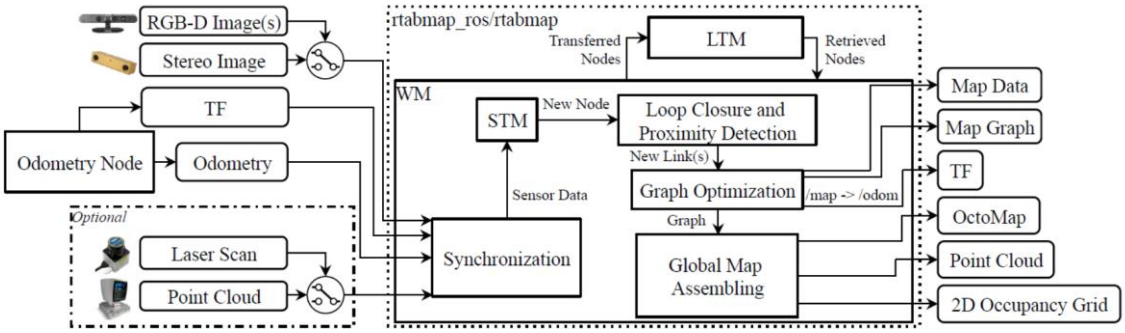
Ngoài ra giải thuật SLAM còn có thể phát hiện được các chướng ngại vật bất ngờ xuất hiện trong khi robot đang di chuyển giúp cho bài toán né tránh vật cản và hoạch định đường đi sau đó được giải quyết hiệu quả hơn. Tuy nhiên, việc xác định vị trí tương đối bằng các phép so sánh với phần bản đồ trước đó sẽ khiến thời gian đáp ứng phụ thuộc phần lớn vào tốc độ tính toán của hệ thống, điều này có khả năng gây sai lệch thông tin vị trí tọa độ và hướng di chuyển hiện tại của robot [6].

Visual SLAM

Visual SLAM (vSLAM) là công nghệ SLAM chỉ sử dụng hình ảnh để lập bản đồ từ môi trường thế giới thực và xác định vị trí của robot (hoặc người giám sát). So với các cảm biến được sử dụng trong thuật toán SLAM cổ điển, như GPS (hệ thống định vị toàn cầu) hay LIDAR thì camera có giá cả phải chăng hơn và có thể lấy thêm các thông tin khác từ môi trường như màu sắc, kết cấu và hình thức. Ngoài ra, các camera ngày nay có kích thước nhỏ gọn, giá thành thấp và mức tiêu thụ điện năng thấp. Ví dụ về việc sử dụng vSLAM gần đây như điều khiển robot hình người, máy bay không người lái (phương tiện trên không không người lái) và phương tiện trên bộ không người lái, robot thám hiểm, robot tự động dưới nước và nội soi.

2.3. Giải thuật RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) là một cách tiếp cận SLAM dựa trên đồ thị RGB-D thu được từ cảm biến RGB-D, là một loại thiết bị cảm biến độ sâu hoạt động cùng với cảm biến màu sắc. [7].



Hình 2.2 Sơ đồ khối thuật toán RTAB-Map

RTAB-Map sử dụng trình phát hiện vòng lặp (Loop Closure Detector) để xác định hình ảnh mới có khả năng đến từ một vị trí cũ hay không thông qua phương pháp tiếp cận từ nhiều điểm. Đầu vào của thuật toán là sự linh hoạt thay thế giữa cảm biến camera, cảm biến quét lidar và IMU (cảm biến encoder và góc quay). Từ đó các kỹ thuật (technique) hỗ trợ cho việc xác định odometry hay phát hiện vòng lặp cũng có thể sử dụng linh hoạt. Trong phần kiểm tra thuật toán sau đây sử dụng đầu vào từ camera, rplidar, vậy nên kỹ thuật xác định odometry chính là ICP lấy đầu vào là kết quả scan, còn kỹ thuật phát hiện vòng lặp được sử dụng là Visual (BoVW) lấy đầu vào là hình ảnh thu được từ camera [8]. Hình ảnh độ sâu (depth image) có thể không sử dụng do đã có kết quả scan từ rplidar.

Thể hiện của thuật toán này là package[7] *rtabmap_ros*. Cấu trúc của package này được trình bày như sơ đồ ở Hình 2.2. Sau khi các dữ liệu đầu vào trải qua bước đồng bộ hóa ở node *synchronization*, module bộ nhớ ngắn hạn (STM) sẽ tạo một

[7] ROS package là một thư mục chứa các tệp thực thi (executable file) và những tệp hỗ trợ để phục vụ một mục đích cụ thể.

node mới để ghi nhận các thông tin như odometry, dữ liệu cảm biến như các thông tin bổ sung cho các module tiếp theo.

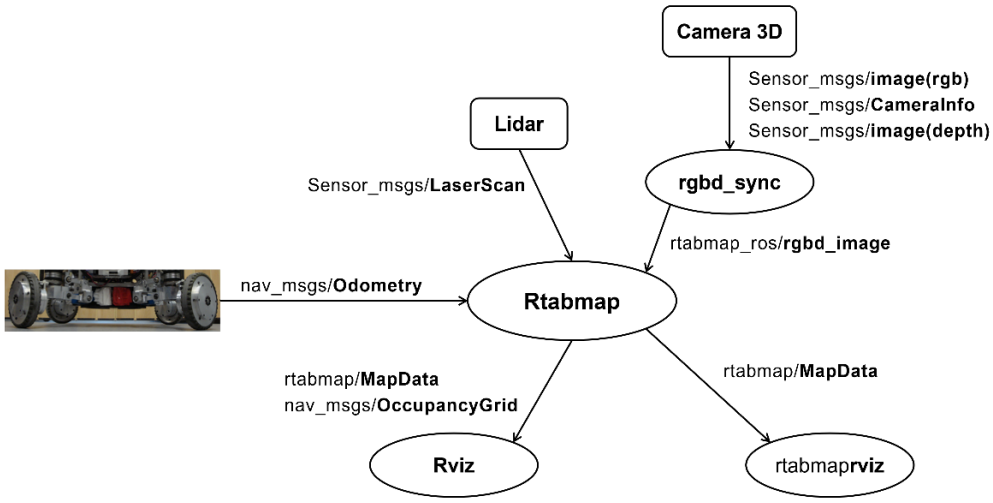
Các đầu vào được yêu cầu:

- TF (Transform) – xác định vị trí tọa độ của các cảm biến.
- Odometry (IMU).
- Hình ảnh ghi nhận được từ camera.
- Kết quả quét laser từ cảm biến Lidar.

Tất cả các thông tin này sau đó được đồng bộ hóa, ghi nhận bởi STM và chuyển đến module thực hiện đồ thị hóa trong thuật toán SLAM. Kết quả đầu ra là dữ liệu bản đồ chứa trong *Map Data*, dữ liệu đồ thị sau khi tối ưu hóa trong *Map Graph* (tuy nhiên ở thời điểm đầu sẽ không có dữ liệu nào), hiệu chỉnh tư thế (odometry) trong *TF*, và các tùy chọn hiển thị theo các dạng lưới 3D (OctoMap), Point Cloud và lưới 2D (2D Occupancy Grid).

Phương pháp *rtab-map* chạy trên một mô hình quản lý đồ thị, gồm các module *Loop Closure Proximity Detection, Graph Optimization* và *Global Map Assembling*. Các module này được sử dụng để tối ưu hóa kích thước của biểu đồ để hệ thống RTAB-Map có thể hoạt động lâu dài trong những không gian rộng lớn. Nếu không quản lý bộ nhớ, khi bản đồ được mở rộng, thời gian xử lý của các module kể trên có thể tốn một khoảng thời gian xử lý quá lớn. Để giải quyết bài toán mở rộng bản đồ, bộ nhớ của *rtab-map* được chia thành bộ nhớ làm việc (WM) và bộ nhớ dài hạn (LTM), khi một node được chuyển sang LTM thì nó sẽ không còn khả dụng cho các module bên trong WM. Khi thời gian cập nhật *rtab-map* vượt ngưỡng thời gian cho phép, nghĩa là WM đã thu được một số lượng node đến ngưỡng nhất định, các node trong WM sẽ được STM xác định chuyển sang LTM để làm giảm kích thước của WM và đồng thời làm giảm đi thời gian xử lý.

Package *rtab-map* ước tính vị trí tọa độ và tư thế của robot dựa trên những dữ liệu thu được từ cảm biến và các phép đo hình học cụ thể, từ đó xây dựng bản đồ.



Hình 2.3 Sơ đồ tín hiệu xây dựng và định vị đồng thời

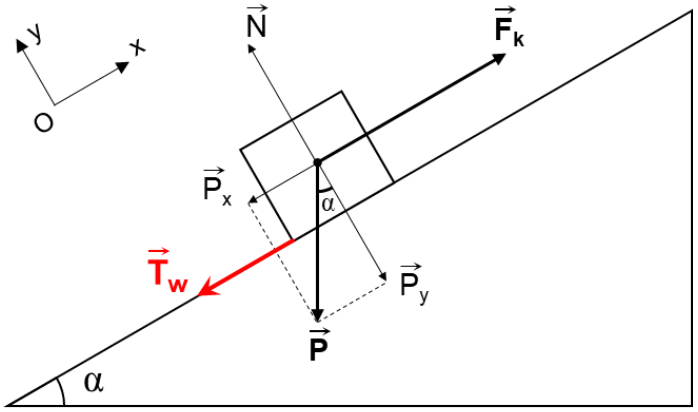
Trong hệ thống RTAB-Map xây dựng trên ROS, mỗi node có nhiệm vụ và chức năng độc lập. Chức năng của các node được mô tả chi tiết trong Bảng 2.1.

Bảng 2.1 Chức năng từng node của hệ thống xây dựng RTAB-Map trên ROS

STT	Tên node	Chức năng
1	Lidar	Đọc cảm biến Lidar và gửi dữ liệu quét laser (scan) đến node *rtabmap*.
2	Camera 3D	Đọc cảm biến camera 3D và gửi hình ảnh và thông tin cảm biến (camera info) đến node *rgbd_sync*.
3	Rgbd_sync	Đảm bảo các topic hình ảnh được đồng bộ với nhau trước khi đi vào node *rtabmap*.
4	Rtabmap	Đồng bộ hóa các topic đến từ tín hiện LaserScan và thông tin từ Camera 3D.
5	Nav_msgs/Odometry	Đưa thông tin về vị trí, vận tốc của robot hiện tại đến node *rtabmap*.
6	Rviz, rtabmaprviz	Giám sát hoạt động của robot và theo dõi kết quả.

2.4. Công suất động cơ

Công suất động cơ (engine power) là tổng năng lượng mà động cơ có thể tạo ra, thường được biểu thị bằng đơn vị kilowatt (kilô-oát) hay mã lực. Công suất của động cơ còn thể hiện tốc độ sinh ra momen xoắn của động cơ và đồng thời cũng thể hiện khả năng đạt tốc độ nhanh hay chậm của bánh xe.



Hình 2.4 Phân tích hợp lực tác động lên vật thể trên mặt phẳng nghiêng

Hình 2.4 thể hiện sơ đồ phân tích hợp lực, với $\vec{P}$ là vectơ trọng lực, $\overrightarrow{F_k}$ là lực kéo của động cơ, $\vec{N}$ là phản lực của mặt phẳng lên vật và $\overrightarrow{T_w}$ là lực đối ngẫu bánh xe.

Với gia tốc trọng trường g = 10 m/s^2, khối lượng của xe m = 5 kg.

Trọng lượng của xe là: **P** = m × g = 5 × 10 = 50 N.

Giả sử gia tốc a = 1 m/s^2.

Tỉ số truyền **N** của động cơ là thương số giữa tốc độ đầu vào và đầu ra:

$$N = \frac{w_{wheel}}{w_{motor}}$$

- Nếu N > 1: làm giảm tốc độ và tăng mô-men động cơ → có lợi về lực
- Nếu N ≤ 1: tăng tốc độ động cơ → có lợi về đường đi

Xét bài toán leo cầu thang, robot cần lực lớn, vì vậy cần chọn tỉ số truyền N > 1.

Quay trở lại sơ đồ hợp lực, ta có:

$$\overrightarrow{F_k} = \sum_{i=0}^{6} \tau_{motor} = 6 \times \mathrm{P}_{motor} \times \frac{9.55}{n} = 57.3 \times \frac{U_m \times I_m}{n}$$

Trong đó:

- $\mathrm{U_m}$ là giá trị hiệu điện thế sử dụng để động cơ có thể hoạt động (V)
- $\mathrm{I_m}$ là giá trị cường độ dòng điện đi qua động cơ (A)
- n là tốc độ động cơ hay tốc độ quay (vòng/phút)

Theo Định luật II Newton: $\vec{N} + \vec{P} + \overrightarrow{F_k} + \overrightarrow{f_{ms}} = m\vec{a}$

Chiếu lên trục Ox, ta được:

$$F_k - P_x - T_w = ma$$

$$\Leftrightarrow F_k - P \sin\alpha - T_w = ma \quad (1)$$

Chiếu lên trục Oy, ta được:

$$N = P_y = P \cos\alpha \quad (2)$$

Thay (2) vào (1):

$$F_k - P \sin\alpha - T_w = ma$$

$$\Leftrightarrow F_k = 50 \times \sin 45 + T_w + ma$$

$$\Leftrightarrow 57.3 \times \frac{U_m \times I_m}{n} = 25\sqrt{2} + T_w + 5$$

Với r là khoảng cách từ động cơ đến tâm bánh xe robot, ta có điều kiện ràng buộc sau: $T_{motor} > T_{wheel}$ (để thắng được lực đối ngẫu bánh xe và leo lên)

$$T_{wheel} = r \times T_{motor}$$

Từ biểu thức (1):

$$(1) \Leftrightarrow 6T_{motor} > 25\sqrt{2} + 5 + (6 \times 0.02 \times T_{motor})$$

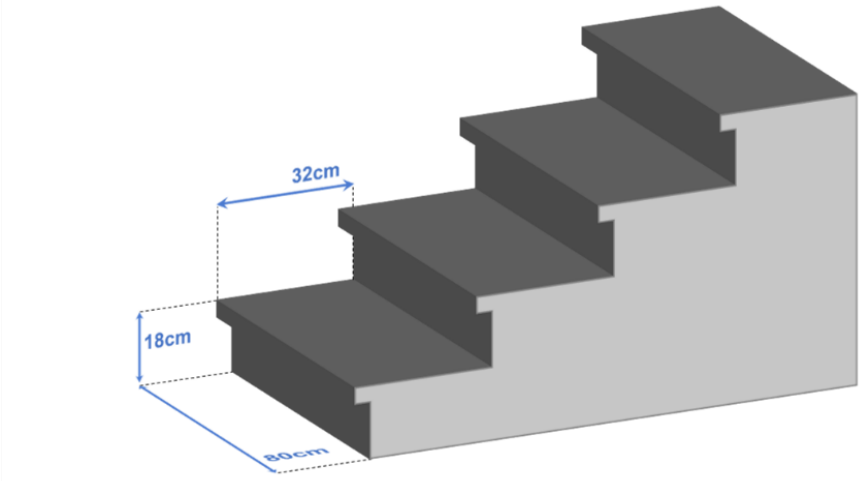
$$\Leftrightarrow T_{motor} > 6.86(N)$$

Vậy cần chọn động cơ có công suất và tốc độ quay (vòng/phút) sao cho lực kéo của motor phải thắng lực đối ngẫu của bánh xe (6.86N)

2.5. Khảo sát các mô hình cầu thang

Cầu thang là một dạng địa hình gồ ghề và khó di chuyển đối với robot, vì vậy việc khảo sát và quyết định mô hình cầu thang nào làm chuẩn để thực hiện tính toán kích thước bánh xe, độ cao khung gầm, độ cao trục nối bánh xe, ... sao cho robot có thể leo lên được cầu thang là cực kỳ cần thiết.

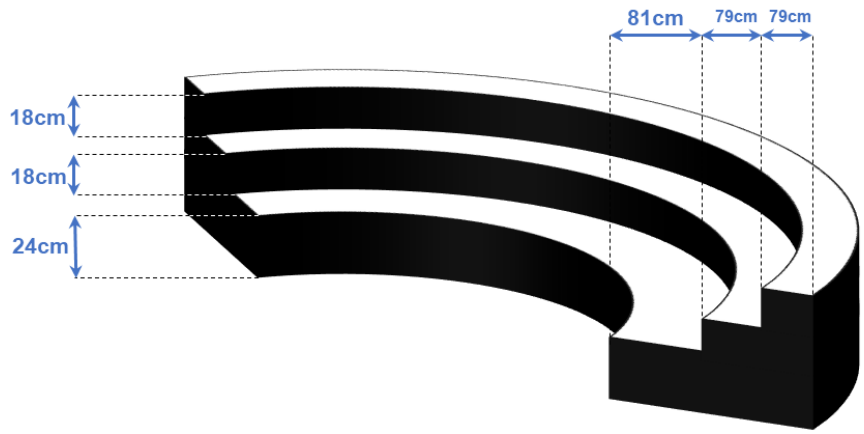
Thực tế có rất nhiều loại cầu thang, tuy vậy vẫn có một vài tiêu chuẩn về độ an toàn, tiết kiệm không gian, phù hợp với nhu cầu sử dụng. Các tiêu chuẩn này bao gồm các thông số như chiều cao mỗi bậc, chiều dài mỗi bậc, chiều rộng mỗi bậc, ... Kích thước phổ biến là chiều cao bậc 150mm, chiều rộng mỗi bậc 300mm.



Hình 2.5 Mô hình cầu thang của toà nhà E, UIT

Thực hiện việc khảo sát các cầu thang ở các toà nhà như toà nhà A, toà nhà B, toà nhà C và toà nhà E. qua đó nhận thấy kích thước cầu thang toà nhà A, B và E có chiều cao bậc là 180mm, chiều rộng bậc là 310mm, đối với toà nhà C thì mỗi bậc thang có chiều cao là 150mm và chiều rộng là 320mm.

Đối với các bậc thang bố trí ở các lối đi ngoài trời, độ rộng bậc thang có thể dài hơn để phù hợp với chiều dài bước chân của người trưởng thành. Cụ thể nhóm đã đo đạc các bậc thang ở sân trước toà nhà B.

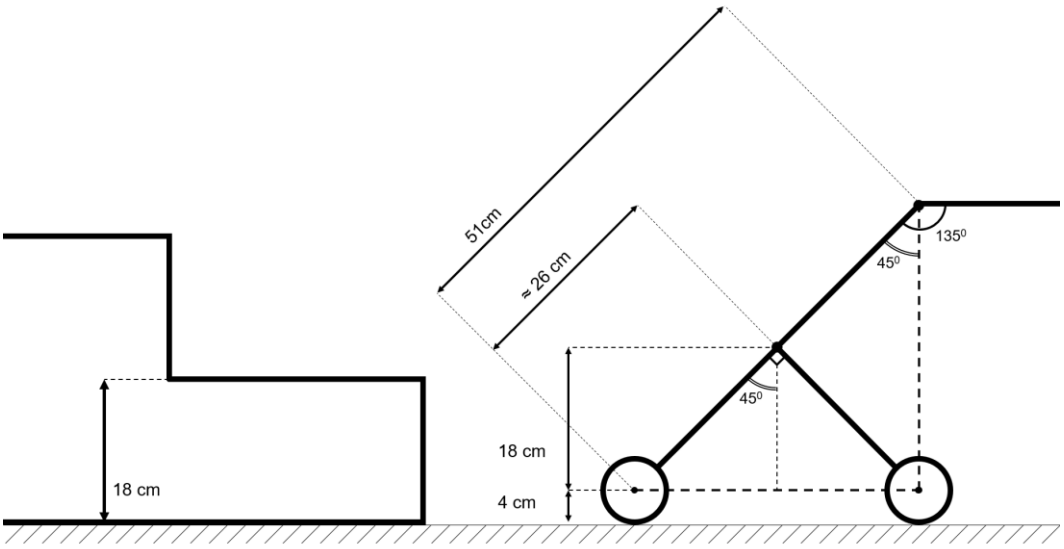


Hình 2.6 Mô hình bậc thang ở sân trước toà nhà B

Theo kết quả đo đạc, đối với các bậc thang ở sân trước toà nhà B, bậc thang đầu tiên có chiều cao 240mm và chiều rộng 810mm, thông số tương ứng với các bậc thang còn lại là 180mm và 790mm.

Do kích thước bậc thang lớn hơn kích thước cầu thang tiêu chuẩn, nhóm chọn kích thước bậc thang ở sân trước toà nhà B để thiết kế robot.

2.6. Mô hình Six-Wheel Robot



Hình 2.7 Tính toán độ dài các chân robot

Để đảm bảo robot có thể leo được bậc thang cao 18cm, chiều cao chân robot phải cao tối thiểu 18cm.

Cấu trúc chân robot là các khớp nối vuông góc và khớp nối 135 độ. Nối các đường vuông góc từ khớp nối vuông xuống mặt phẳng sẽ tạo ra các tam giác vuông cân.

Từ đó suy ra được các độ dài còn lại như sau:

- Chiều dài chân trước: khoảng 26cm
- Chiều dài chân sau: 51cm
- Chiều dài thanh đỡ cảm biến và vi điều khiển: 25cm
- Khoảng cách giữa chân trước và thanh đỡ cảm biến: khoảng 26cm

2.7. Dung lượng pin

Dung lượng pin là thông số thể hiện khả năng tích trữ điện năng của pin, phụ thuộc vào khối lượng và đặc tính riêng của vật liệu sử dụng làm chất điện phân. Thông số này được nhà sản xuất pin kiểm tra, đo đạc và con số được in trên sản phẩm là dung lượng tối đa pin có thể lưu trữ.

Thông số mAh (mili-ampe giờ) là đơn vị tính của dung lượng pin, biểu thị giá trị công suất điện theo thời gian. Dung lượng càng cao, khả năng lưu trữ của pin càng cao, cũng có nghĩa là thời gian sử dụng càng lâu. Kí hiệu "A" viết tắt cho ampere, đơn vị đo cường độ dòng điện.

Trên một số dòng sản phẩm pin, dung lượng còn được ghi bằng đơn vị Wh. Sự khác nhau giữa đơn vị này với mAh là nó cho biết công suất hoạt động trong 1 giờ.

Trong điều kiện lý tưởng, ta có thời gian sử dụng pin là:

$$time(h) = \frac{Ah \times U(Volt)}{P(Watt)}$$

Trong đó:

- Ah là dung lượng của pin, đo bằng đơn vị ampere-hour (đọc là ampe-giờ), nếu sử dụng đơn vị mAh (đọc là mili-ampe-giờ) thì phải đổi đơn vị thành Ah bằng cách nhân hệ số 10^{-3}.
- U(Volt) là hiệu điện thế ngõ ra của pin, tạo từ hai điện cực dương (+) và điện cực âm (-) của pin, đó là điện áp danh định mà pin tạo ra khi hoạt động, thường được tính bằng Volt (đọc là vôn, kí hiệu là V).
- P(Watt) là công suất tiêu thụ của mạch tiêu thụ. Nó được tính bằng Watt (đọc là oát, kí hiệu là W).

Tính công suất tiêu thụ của từng thiết bị trong trường hợp sử dụng tối đa:

- STM32: PSTM = 5V × 0.036A = 0.18W
- Jetson TX1: P_{TX1} = 19V × 5A = 95W
- Động cơ GB37-520 12V 7RPM: P_{M520} = 10W
- Động cơ JGB37-545 12V 37RPM: P_{M545} = 40W
- Driver điều khiển động cơ XY-160D: P_{Driver} = 5V × 7A = 35W

Tổng công suất tiêu thụ tối đa:

$$\sum P = P_{STM} + P_{TX1} + 4 \times P_{M520} + 2 \times P_{M545} + 6 \times P_{Driver}$$

$$\sum P = 0.18 + 95 + 4 \times 6 + 2 \times 60 + 6 \times 35 = 449.18\ (W)$$

Nguồn năng lượng sử dụng là 2 pin Lipo 12V dung lượng 5200mAh dòng xả 45C được ghép song song.

Tổng dung lượng pin là: 5200mAh × 2 = 10400mAh = 10400×10^{-3} Ah.

Số giờ sử dụng pin liên tục tối đa là:

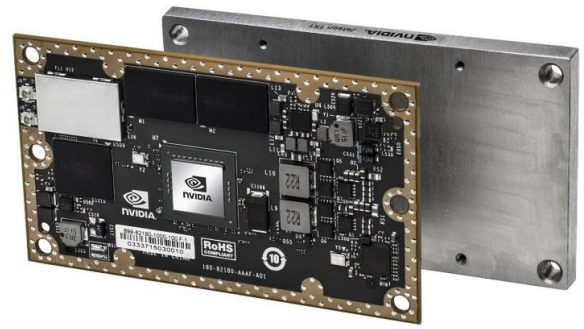
$$t = \frac{10400 \times 10^{-3}(Ah) \times 12(V)}{449.18(W)} \approx 0.2778(h) = 16.668(min)$$

Vậy, khi sử dụng 2 pin Lipo 12V 5200mAh ghép song song cho mạch tiêu thụ gồm 1 vi điều khiển STM32, 1 máy tính nhúng Jetson TX1, 4 động cơ GB37-520, 2 động cơ JGB37-545 và 6 mạch điều khiển động cơ XY-160D, thời gian sử dụng pin liên tục trong trường hợp sử dụng với công suất tối đa là khoảng 15 phút.

CHƯƠNG 3. **THIẾT LẬP ROBOT TỰ HÀNH**

3.1. Các thành phần phần cứng

3.1.1. NVIDIA Jetson TX1 Developer Kit



Hình 3.1 Jetson TX1 Module

NVIDIA Jetson là nhà cung cấp những giải pháp phần cứng hàng đầu trên thế giới dành cho các máy tự động (autonomous machines) và các ứng dụng nhúng có khả năng tích hợp trí tuệ nhân tạo (AI) và học sâu (Deep Learning).

Jetson TX1 là một dòng máy tính nhúng chạy trên nền tảng hệ điều hành Linux đầu tiên trên thế giới sử dụng Module. Jetson TX1 là công cụ thích hợp để phát triển cho mọi loại hệ thống nhúng, từ các hệ thống có hiệu suất thấp cho đến các hệ thống có hiệu suất hoạt động cao.

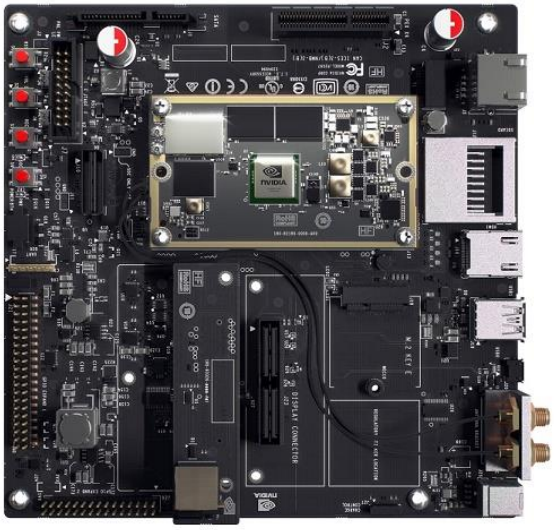
Thông số kỹ thuật chi tiết:

Bảng 3.1 Thông số kỹ thuật của Jetson TX1 Module

Điện áp sử dụng	3.3VDC
GPU – Bộ xử lý đồ hoạ	1 TFLOP/s, Maxwell 256 nhân
CPU – Vi xử lý	64-bit ARM Cortex-A57
RAM	4GB LPDDR4, 25.6 GB/s
Bộ nhớ lưu trữ	16GB eMMC

Video	4K
Camera	Hỗ trợ 1400 megapixel/giây
Wifi/Bluetooth	802.11ac 2x2 – có bluetooth
Kết nối mạng	1GB Ethernet
Hỗ trợ hệ điều hành	Linux phiên bản dành cho Tegra
Kích thước	50 × 87 mm

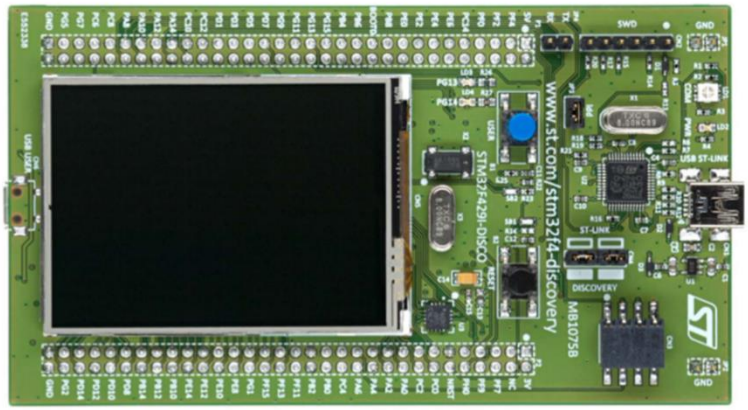
Jetson TX1 Developer Kit là một loại board phát triển, trang bị đầy đủ một hệ thống trên module thay vì chỉ riêng con chip. Nguồn điện sử dụng cho bộ kit là 19VDC 5A.



Hình 3.2 Kit phát triển Jetson TX1

Jetson TX1 đáp ứng tốt trong việc thiết kế và xây dựng nhiều loại máy khác nhau bằng cách nạp thêm vào các thuật toán để giúp máy giải quyết các bài toán trong thực tế như tránh các chướng ngại vật trên đường di chuyển. Jetson TX1 cũng có thể được kết hợp với AI - trí tuệ nhân tạo để huấn luyện cho các thiết bị đầu cuối, từ đó phát triển hệ thống phục vụ chế tạo các thiết bị bay không người lái hay robot tự hành.

3.1.2. STM32F429 Discovery Kit



Hình 3.3 STM32F429 Discovery Kit

Được STMicroelectronics công bố vào tháng 9/2013, STM32F429 Discovery Kit giúp lập trình viên khám phá hiệu suất cao của dòng sản phẩm STM32F4 và phát triển các ứng dụng cá nhân. Kit được thiết kế xung quanh bộ vi điều khiển STM32F429ZIT6 với 144 chân.

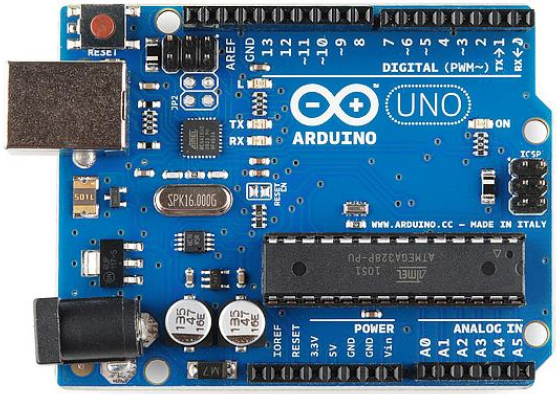
Thông số kỹ thuật chi tiết:

Bảng 3.2 Thông số kĩ thuật chi tiết của STM32F429 Discovery Kit

MCU	32-bit ARM Cortex-M4
Bộ nhớ Flash	2MB
RAM	256KB
Điện áp hoạt động	3.3 - 5V DC
Màn hình LCD	2.4 inch QVGA
SDRAM	64MBit
LED	6
GPIO	144 chân
Ngoại vi	Timer, SPI, USART, I2C, ...

3.1.3. Arduino UNO R3

Arduino UNO R3 là board mạch phát triển từ vi điều khiển ATmega328P.



Hình 3.4 Arduino UNO R3

Các chân IO của Arduino UNO R3 bao gồm:

- 14 chân digital (kỹ thuật số) đầu vào hoặc đầu ra tùy chọn thiết lập được đánh số từ 0 đến 13, trong đó có 6 chân có thể tạo đầu ra PWM.
- 6 chân analog (tín hiệu tương tự) đầu vào, kí hiệu chân từ A0 đến A5.

Thông số kỹ thuật chi tiết:

Bảng 3.3 Thông số kỹ thuật chi tiết của Arduino UNO R3

Vi điều khiển	ATmega328
Điện áp hoạt động	5V DC
Tần số hoạt động	16 MHz
Dòng tiêu thụ	Khoảng 30mA
Điện áp vào (Vin) khuyên dùng	7-12V DC
Điện áp vào (Vin) giới hạn	6-20V DC
Số chân digital I/O	14 (6 chân PWM)

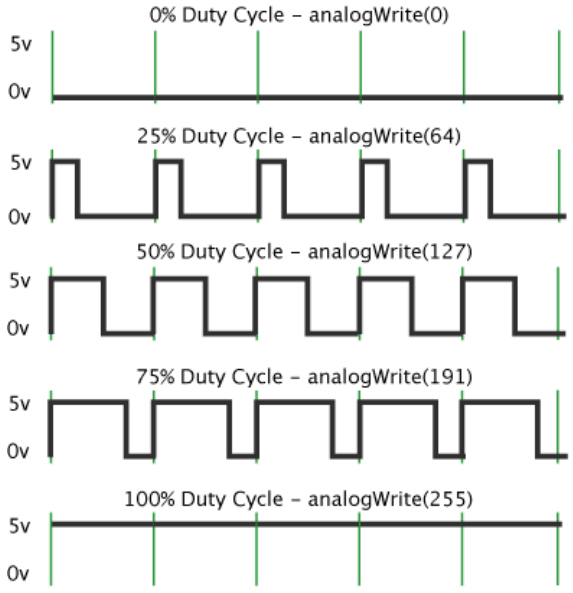
Số chân analog	6 (độ phân giải 10bit)
Dòng tối đa trên mỗi chân	30mA
Dòng tối đa trên chân 5V	500mA
Dòng tối đa trên chân 3.3V	50mA
Bộ nhớ flash	32 KB, trong đó 0.5 KB bootloader
SRAM	2 KB
EEPROM	1 KB

Arduino UNO R3 có thể tạo ra một xung tín hiệu điện áp một chiều có biên độ 5V, mỗi chu kỳ (duty cycle) nằm ở khoảng 2 mili-giây tương đương với tần số PWM của Arduino ở khoảng 500Hz.

Để tạo ra tín hiệu PWM ở chân output digital:

```
analogWrite(<x>, <y>);
```

Trong đó: <x> là số thứ tự chân, <y> là tham số có giá trị từ 0 đến 255 tương ứng với giá trị từ 0% đến 100% chu kỳ Duty Circle.



Hình 3.5 Tín hiệu đầu ra PWM tạo bởi Arduino UNO R3

3.1.4. Cảm biến RP-LiDAR A1M8

Cảm biến LiDAR A1M8 là sản phẩm độc quyền của SLAMTEC, một công ty chuyên về công nghệ định vị robot, cung cấp các giải pháp điều hướng (navigation) và bản địa hoá (localization) cho robot. Từ khả năng cung cấp bản đồ có độ phân giải cao, cảm biến LiDAR được ứng dụng trong các lĩnh vực đời sống như khảo cổ học, trắc địa, địa chất, dẫn đường bằng laser, vẽ bản đồ laser không ảnh (ALTM), đo độ cao, độ sâu, …

LiDAR, hay "Light Detection And Ranging" (phát hiện ánh sáng và phạm vi), còn được gọi là cảm biến quét laser. Nó là một phương pháp đo khoảng cách đến các đối tượng bằng cách bắn ra chùm xung laser và sau đó đo các xung phản xạ từ các đối tượng đó. Sự khác biệt về thời gian truyền về (returning times) và bước sóng (wave lengths) được dùng để tính toán khoảng cách phục vụ cho việc xây dựng bản đồ 2D/3D.

Hình 3.6 Cảm biến RP-LiDAR A1M8

Trong lĩnh vực xe tự hành và robot, cảm biến LiDAR được ứng dụng để có thể liên tục quét xung quanh và phát hiện các phương tiện giao thông khác, người đi bộ và các mảnh vụn hay vật chắn trên đường. Các loại robot hút bụi được dùng trong các gia đình hiện nay cũng ứng dụng công nghệ cảm biến LiDAR.

LiDAR A1M8 có thể giao tiếp bằng giao thức UART thông qua một module đi kèm, có thể dễ dàng giao tiếp với vi điều khiển hay máy tính nhúng qua cổng USB.

Thông số kỹ thuật chi tiết:

Bảng 3.4 Thông số kỹ thuật của cảm biến RP-LiDAR 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, tối thiểu 0.3m
Góc quét	360°
Tốc độ lấy mẫu tối đa	8000 mẫu một lần
Tần số quét tối đa	10Hz
Kích thước	70 × 98.5 mm

3.1.5. Cảm biến máy ảnh Astra 3D Camera

Hình 3.7 Cảm biến Astra 3D Camera

Được sản xuất bởi hãng ORBBEC, Astra 3D Camera là một máy ảnh RGB-D, một loại máy ảnh độ sâu (depth camera) cung cấp cả dữ liệu đầu ra thời gian thực về độ sâu (D) và màu sắc (RGB).

Astra 3D Camera được tối ưu hóa cho các trường hợp sử dụng tầm xa, điều này làm cho nó trở nên tuyệt vời cho các hệ thống tương tác, giải trí.

Đặc biệt, Astra 3D Camera cung cấp thị giác máy tính cho phép nhận dạng khuôn mặt, nhận dạng cử chỉ, theo dõi cơ thể người, đo lường ba chiều, nhận thức môi trường và tái tạo bản đồ 3D.

Thông số kỹ thuật chi tiết:

Bảng 3.5 Thông số kỹ thuật của cảm biến Astra 3D Camera

Phạm vi	0.6m – 8m
FOV[8]	60°H x 49.5°V x 73°D
Độ phân giải của ảnh RGB	640p x 480p (30fps)
Độ phân giải của ảnh độ sâu	640p x 480p (30fps)
Kích thước	165mm x 30mm x 40mm
Nhiệt độ hoạt động	0 – 40°C
Cổng kết nối	USB 2.0
Công suất tiêu thụ	< 2.4 W
Nền tảng mở rộng	Android / Linux / Windows7/8/10
SDK	Astra SDK, OpenNI2 hoặc SDK thứ ba
Sai số	± 1 – 3mm /1 m
Microphones	2 Built-in

[8] Field of View – Trường nhìn, phạm vi quan sát

3.1.6. Cảm biến tốc độ - Encoder

Encoder là một cảm biến tốc độ, có khả năng biến đổi chuyển động (tịnh tiến hoặc xoay quanh trục) tạo ra tín hiệu xung hoặc tín hiệu số đáp ứng với tốc độ chuyển động bằng cách mã hóa vòng quay của encoder. Độ phân giải của encoder là số xung ghi nhận được trên một vòng quay, ví dụ 16 xung/vòng.

Có hai loại encoder là encoder tuyệt đối (absolute encoder) và encoder tương đối (incremental encoder). So sánh hai loại có thể thấy encoder tương đối đo tốc độ quay của bánh xe dựa trên sự tăng giảm số xung trên từng đơn vị thời gian, còn encoder tuyệt đối bên cạnh tính năng tương tự còn có thể xác định được chính xác tọa độ của từng điểm trên vòng quay. Cụ thể, một vòng quay của encoder sẽ được mã hóa theo mã nhị phân ở từng vị trí và đơn vị thường là bao nhiêu bit, điều này có nghĩa là mỗi vị trí vòng quay của encoder đều có một tọa độ riêng biệt.

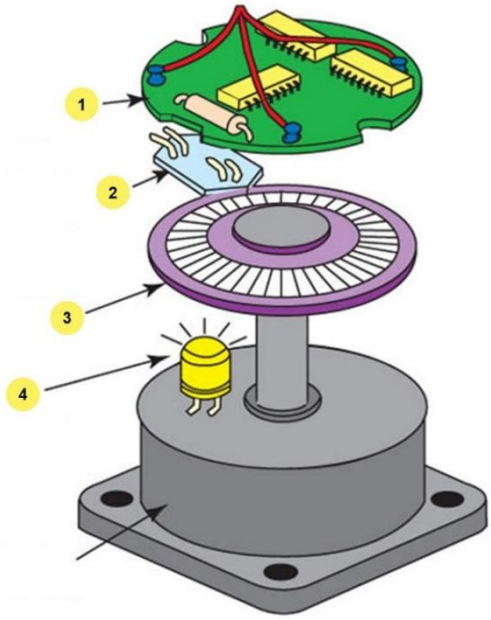
Ưu điểm của encoder tuyệt đối là giữ được giá trị tuyệt đối kể cả khi mất điện, đổi lại là giá thành cao vì phải chế tạo phức tạp hơn. Trong khi encoder tương đối thì có giá thành rẻ, chế tạo đơn giản và xử lý tín hiệu trả về dễ dàng; tuy nhiên lại dễ bị sai lệch về xung khi trả về, vì vậy sẽ có sai số tích lũy theo thời gian.

Encoder tuyệt đối được sử dụng trong các máy CNC vì yêu cầu độ chính xác cao và cần phải tiếp tục chạy tại vị trí dừng sau khi được cấp điện trở lại (trong trường hợp gặp sự cố mất điện trong khi sử dụng). Thông số vị trí này được lưu vào bộ nhớ được nuôi bằng nguồn điện là pin gắn rời.

Đề tài này sử dụng encoder tương đối được tích hợp trên động cơ một chiều.

Cấu tạo của encoder được trình bày như Hình 3.8. Trong đó bao gồm:

(1) - Bảng mạch điện giúp khuếch đại tín hiệu.
(2) - Mắt thu quang điện.
(3) - Đĩa quay có khoét lỗ gắn vào trục động cơ.
(4) - Đèn Led dùng làm nguồn phát sáng.



Hình 3.8 Cấu tạo encoder

Nguyên lí hoạt động

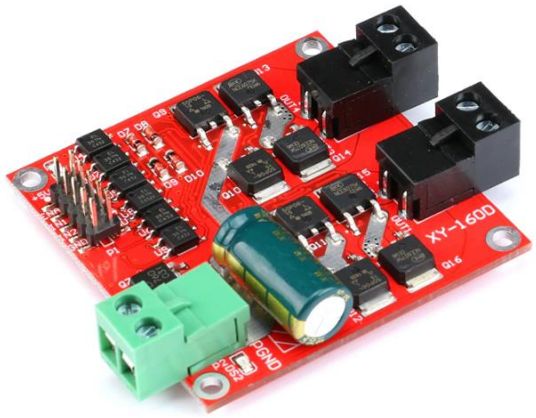
Khi trục động cơ xoay, encoder sẽ xoay theo, khi đó mắt thu quang điện sẽ ghi nhận tín hiệu ánh sáng từ đèn Led chiếu qua các lỗ khoét trên đĩa và số xung đếm được sẽ tăng lên khi mỗi lần tín hiệu đèn Led được nhận. Ví dụ: trên đĩa chỉ có một lỗ khoét, khi đĩa quay được một vòng thì mắt thu sẽ nhận được một tín hiệu, vậy suy ra khi nhận được một tín hiệu thì có nghĩa là đĩa đã quay được một vòng.

Đây chính là nguyên lý hoạt động của Encoder cơ bản, còn đối với nhiều loại encoder khác thì khi đĩa quay có nhiều lỗ hơn khi đó tín hiệu thu nhận sẽ khác hơn.

3.1.7. Mạch điều khiển động cơ

Mạch điều khiển động cơ (Motor Driver circruit) là mạch điện điều chỉnh tốc độ, mô-men xoắn và đầu ra vị trí của động cơ để đạt được công suất đầu ra mong muốn. Cụ thể, mạch điều khiển động cơ lấy tín hiệu dòng điện thấp (low-current signal) từ mạch điều khiển và khuếch đại nó thành tín hiệu dòng điện cao (high-current signal) để điều khiển động cơ chính xác.

Đề tài này sử dụng mạch điều khiển động cơ **XY-160D**.



Hình 3.9 Mạch điều khiển động cơ XY-160D

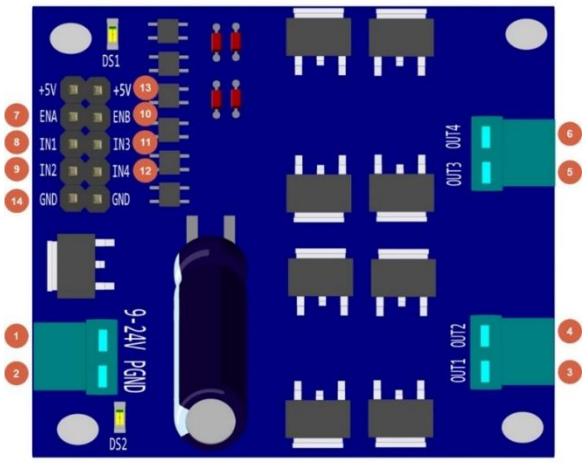
XY-160D được sử dụng để điều khiển cùng lúc 2 động cơ DC với những tín hiệu điều khiển riêng biệt. Điểm đặc biệt của mạch là sử dụng IC Driver bằng công nghệ MOSFET với optocoupler cách li quang và chống nhiễu bằng TVS Diode.

Thông số kỹ thuật chi tiết:

Bảng 3.6 Thông số kỹ thuật của mạch điều khiển động cơ XY-160D

Điện áp sử dụng	7 ~ 24 VDC (giới hạn: 6.5 ~ 27 VDC)
Công suất	Tối đa 160W cho mỗi động cơ
Mức tín hiệu điều khiển	Mức cao: 3.0 ~ 6.5V; Mức thấp: 0 ~ 0.8V
Dòng tín hiệu điều khiển	3 - 11mA
Tốc độ điều khiển xung PWM	0 - 10 KHz
Độ rộng xung tối thiểu	5us
Nhiệt độ hoạt động	–25 ~ 85 °C
Kích thước module	55 × 55 × 13 mm
Khối lượng	32g

Sơ đồ chân của driver XY-160D:



Hình 3.10 Sơ đồ chân của driver XY-160D

Thứ tự các chân và chức năng của từng chân được trình bày theo Bảng 3.7:

Bảng 3.7 Tên các chân và chức năng từng chân của driver XY-160D

STT	Tên chân	Mô tả
1	9~24 VDC	Cực dương nguồn cấp điện cho động cơ
2	PGND	Cực âm nguồn cấp điện cho động cơ
3, 4	OUT1, OUT2	Cực dương, cực âm đầu ra động cơ 1
5, 6	OUT3, OUT4	Cực dương, cực âm đầu ra động cơ 2
7	ENA	Chân Enable / điều khiển PWD động cơ 1
8, 9	IN1, IN2	Điều khiển đầu vào động cơ 1
10	ENB	Chân Enable / điều khiển PWD động cơ 2
11, 12	IN3, IN4	Điều khiển đầu vào động cơ 2
13	+5V	Cực dương nguồn cấp cho mạch điều khiển
14	GND	Cực âm nguồn cấp cho mạch điều khiển

Để điều khiển động cơ, thiết lập các mức tín hiệu cao hoặc thấp cho các chân.

Bảng 3.8 Phương pháp điều khiển động cơ

ENA / ENB	IN1 / IN3	IN2 / IN4	Chức năng cho động cơ 1 / 2
-	0	0	Phanh động cơ
-	1	1	Thả tự do động cơ
PWM	1	0	Đi tới với tốc độ tuỳ chỉnh
PWM	0	1	Đi lùi với tốc độ tuỳ chỉnh
1	1	0	Đi tới với tốc độ tối đa
1	0	1	Đi lùi với tốc độ tối đa

3.1.8. Động cơ một chiều

Động cơ một chiều – DC Motor là bất kỳ loại động cơ điện nào có thể chuyển đổi năng lượng điện từ dòng điện một chiều thành năng lượng cơ học. Đề tài sử dụng hai loại động cơ JGB37-545 và GB37-520 để đáp ứng được lực kéo/đẩy robot khi leo cầu thang.

Động cơ DC Servo JGB37-545

Động cơ JGB37-545 loại dùng cho điện áp 12V, tốc độ quay 37RPM[9] có thêm Encoder được cố định trên động cơ với hai kênh ngõ ra là channel A và channel B. Mỗi channel trả về 16 xung cho mỗi vòng quay của đĩa Encoder, nghĩa là khi đọc cả 2 xung sẽ nhận được tổng cộng 32 xung.

Động cơ được cấu tạo từ kim loại, đảm bảo độ bền và cho độ ổn định cao. Đường kính của phần vỏ ngoài động cơ Ø37mm, đường kính trục quay Ø6mm, chiều dài hộp số 26.5mm.

Loại động cơ này được sử dụng cho bánh xe giữa ở mỗi bên chân robot.

[9] RPM – Revolutions per minute (số vòng quay trên phút)

Hình 3.11 Động cơ DC Servo JGB37-545 12VDC 37RPM

Thông số chi tiết:

Bảng 3.9 Bảng thông số chi tiết của động cơ JGB37-545 12VDC 37RPM

Điện áp hoạt động	12VDC
Tỉ số truyền	168:1
Tốc độ không tải	36-37 RPM
Dòng không tải	0.26 A
Tốc độ tối đa khi có tải	29 RPM
Dòng định mức khi có tải	1.2 A
Lực kéo Moment định mức khi có tải	17 kg.cm
Dòng chịu đựng tối đa khi có tải	3.8 A
Lực kéo Momen tối đa khi có tải	35 kg.cm
Chiều dài hộp số	26.5 mm
Khối lượng	310 g
Công suất định mức	18 W
Công suất tối đa khi có tải	40 W

Động cơ và encoder được điều khiển qua các chân:

Hình 3.12 Các chân điều khiển động cơ JGB37-545 và encoder

Động cơ DC giảm tốc GB37-520 12VDC 7RPM

Động cơ GB37-520 loại 12V 7RPM là loại động cơ có hộp giảm tốc. Đường kính của phần vỏ ngoài động cơ là Ø37mm, đường kính trục quay là Ø6mm, chiều dài hộp số là 29mm. Đặc biệt lực kéo Moment tối đa khi có tải lên đến 45 kg.cm.

Hình 3.13 Động cơ DC giảm tốc GB37-520 12VDC 7RPM

Thông số chi tiết:

Bảng 3.10 Bảng thông số chi tiết của động cơ JGB37-520 12VDC 7RPM

Điện áp hoạt động	12VDC
Tỉ số truyền	810:1
Tốc độ không tải	7 RPM (vòng/phút)
Dòng không tải	0.05 A

Tốc độ tối đa khi có tải	5.6 RPM
Dòng định mức khi có tải	0.3 A
Lực kéo Moment định mức khi có tải	18 kg.cm
Dòng chịu đựng tối đa khi có tải	1.5A
Lực kéo Moment tối đa khi có tải	45 kg.cm
Chiều dài hộp số	29 mm
Khối lượng	170 g
Công suất định mức	3.6 W
Công suất tối đa khi có tải	10 W

Loại động cơ này được sử dụng cho cặp bánh xe trước (front wheel) và cặp bánh xe sau (rear wheel) của robot.

3.1.9. Cầu chì

Cầu chì là một thiết bị cơ bản được sử dụng trong các mạch điện để bảo vệ mạch điện khỏi dòng điện quá tải, hiện tượng ngắn mạch và các sự cố khác, từ đó hạn chế cháy nổ, gây hư hại cho mạch điện.

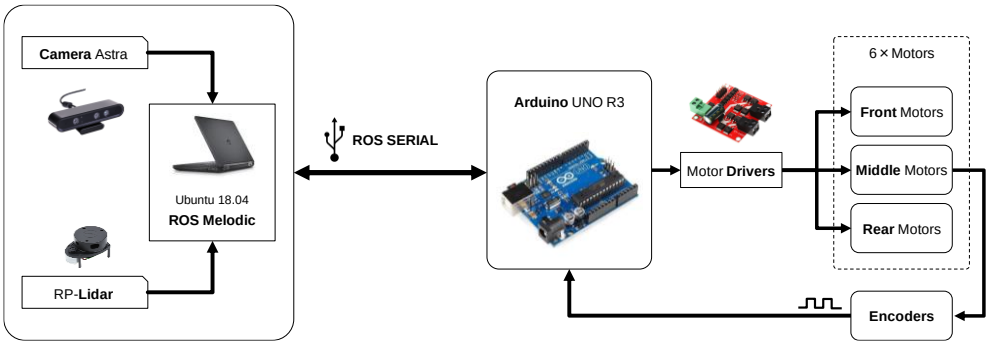
Hình 3.14 Cầu chì

Hoạt động theo nguyên lý nóng chảy và giãn nở vì nhiệt dẫn đến uốn cong và tách rời khỏi mạch khi cường độ dòng điện đi qua cầu chì tăng đến ngưỡng, vì vậy cầu chì có thể bảo vệ mạch khi dòng điện quá tải có khả năng dẫn đến hư hỏng.

Cầu chì được sử dụng cho mạch điều khiển động cơ khi cấp nguồn vào mạch và cấp nguồn ra cho động cơ. Các loại cầu chì sử dụng là cầu chì 10A và 15A.

3.2. Thiết kế phần cứng

3.2.1. Kết nối bộ xử lý và các cảm biến



Hình 3.15 Sơ đồ kết nối bộ xử lý và các cảm biến của robot

Sơ đồ hệ thống phần cứng được mô tả như Hình 3.15, gồm bộ điều khiển trung tâm ban đầu được sử dụng là một máy tính nhúng NVIDIA Jetson TX1 (phiên bản Development Kit) được cài hệ điều hành ROS với bản phân phối ROS Melodic. Tuy nhiên có thể thay thế TX1 bằng một laptop Ubuntu Bionic (18.04) để thiết lập môi trường làm việc (workspace) tương đương với TX1.

Cảm biến được sử dụng bao gồm RP-Lidar (cảm biến quét laser), Camera Astra (cảm biến hình ảnh và độ sâu) và encoder cố định trên động cơ một chiều. Camera Astra và RP-Lidar sẽ thu thập thông tin lấy được từ môi trường và gửi dữ liệu ở dạng point cloud (đám mây điểm) cho ROS thông qua Serial Port (cổng COM giao tiếp 2 chiều, có thể giúp máy tính và thiết bị ngoại vi truyền và nhận dữ liệu cùng lúc) và kết nối bằng cổng USB.

Vi điều khiển ban đầu được sử dụng là STM32F429 Discovery Kit (sau đây gọi tắt là STM32) sẽ điều khiển các động cơ bằng các mạch driver. Ngoài ra STM32 còn đóng vai trò xử lý dữ liệu thu được từ cảm biến tốc độ - encoder để gửi về ROS thông qua giao tiếp UART. Tuy nhiên do đặc thù hệ thống ROS Melodic trên Ubuntu 18.04 không thể nhận diện và đồng bộ được thiết bị STM32, vi điều khiển thay thế là ATmega328 trên Arduino UNO R3, với package `rosserial_arduino` và `rosserial_python` để tạo kết nối giữa Odometry (do Arduino publish) và ROS.

3.2.2. Mô hình robot thử nghiệm

Mô hình robot được thực hiện theo mô hình Rocker Bogie Rover.

Hình 3.16 Mô hình robot thử nghiệm

Phần mô hình gồm 2 phần chính: phần thân và phần chân.

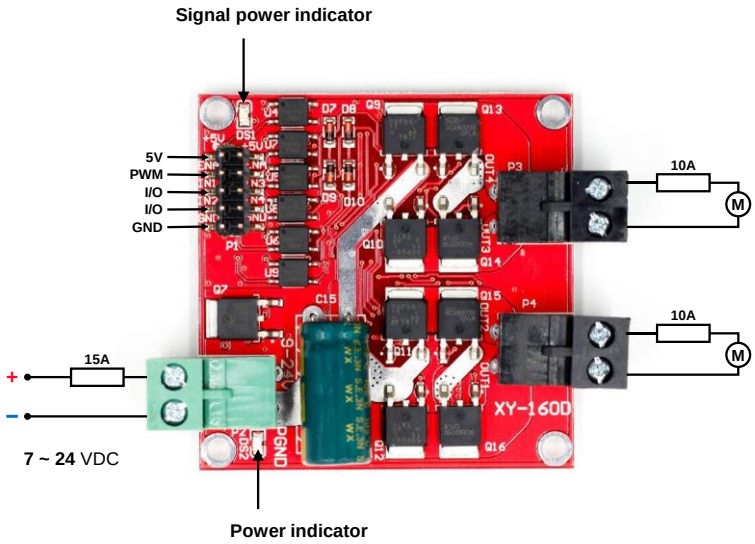
Phần thân bao gồm khung cố định, các cảm biến, vi điều khiển, mặt phẳng mica kết nối phần thân với phần chân, pin, dây dẫn, ... Phần khung cố định cảm biến có kích thước 25×24×14cm, được làm từ các thanh nhôm định hình 20x20. Các cảm biến và vi điều khiển được đặt trên một mặt phẳng mica cố định với khung nhôm.

Phần chân robot bao gồm bánh xe, động cơ, encoder, các thanh nối và khớp xoay chân trước. Bánh xe có phần lốp cao su có gai bám, đường kính tối đa là 85mm, độ rộng 38mm. Các động cơ có đường kính trục Ø6mm, được gắn với các bánh xe bằng khớp lục giác có cạnh 12mm. Encoder được gắn cố định trên động cơ ở hai bánh giữa. Các thanh nối được làm bằng vật liệu nhựa PVC, hình dạng hình trụ tròn rỗng ruột, kích thước Ø32mm, dày 2.2mm. Khớp xoay gồm hai thanh đỡ bằng nhôm có kích thước 17×4cm và các bu-lông để cố định vào phần khung.

3.3. Lập trình Arduino

3.3.1. Điều khiển động cơ

Nối nguồn điện 7~24VDC vào input của mạch điều khiển động cơ XY-160D với cầu chì 15A, nối hai đầu động cơ vào các cổng output trên mạch với cầu chì 10A để bảo vệ mạch không bị hư hỏng khi dòng quá tải.



Hình 3.17 Sơ đồ kết nối với mạch điều khiển động cơ

Sử dụng các chân trên Arduino UNO R3:

- Thiết lập chế độ cho các chân ở hàm `void setup()`:

```
pinMode(<x>, <y>);
```

Trong đó `<x>` *là kí hiệu của chân,* `<z>` *là tùy chọn* `INPUT` *hoặc* `OUTPUT`.

- Thiết lập mức High/Low cho các chân chế độ Output như sau:

```
digitalWrite(<x>, <y>);
```

Trong đó `<x>` *là kí hiệu của chân,* `<y>` *là giá trị* `HIGH` *hoặc* `LOW`.

- Thiết lập giá trị PWM trên một chân Output như sau:

```
analogWrite(<x>, <y>);
```

Trong đó `<x>` *là kí hiệu của chân,* `<y>` *là một số nguyên từ 0 đến 255.*

Để điều khiển robot, ta thiết lập các mức cao (HIGH) hoặc thấp (LOW) và độ rộng xung (Pulse width) cho các chân trên Arduino UNO R3 như sau:

Bảng 3.11 Tín hiệu điều khiển hành động của robot

Hành động của robot	Bánh xe bên trái			Bánh xe bên phải		
	ENA	IN1	IN2	ENA	IN1	IN2
Đi tới (tốc độ tối đa)	Quay theo chiều thuận					
	1	1	0	1	1	0
Đi lùi (tốc độ tối đa)	Quay ngược					
	1	0	1	1	0	1
Đi tới (tốc độ tuỳ chỉnh)	Quay theo chiều thuận					
	PWM	1	0	PWM	1	0
Đi lùi (tốc độ tuỳ chỉnh)	Quay ngược					
	PWM	0	1	PWM	0	1
Quay sang trái	Quay ngược			Quay theo chiều thuận		
	PWM	0	1	PWM	1	0
Quay sang phải	Quay theo chiều thuận			Quay lùi		
	PWM	1	0	PWM	0	1
Phanh (dừng lại)	Ngừng quay					
	-	0	0	-	0	0

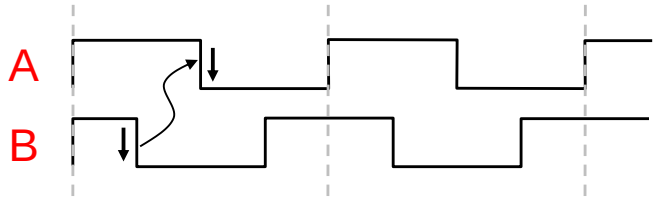
PWM là một số nguyên từ 0 đến giá trị 255. Ví dụ với tốc độ tối đa không tải của động cơ là 37RPM, nếu PWM = 48 thì tốc độ động cơ đầu ra tương ứng sẽ là:

$$\frac{48}{255 - 0 + 1} \times 37\text{RPM} \approx 6.96\text{RPM}$$

3.3.2. Đọc giá trị cảm biến tốc độ Encoder

Cảm biến tốc độ Encoder là loại gắn trực tiếp trên động cơ giảm tốc một chiều JGB37-545, có hai kênh (channel A và channel B) lệch nhau, đĩa Encoder trả về 16 xung trên mỗi vòng quay. Khi đọc cả 2 kênh thì nhận được 32 xung.

Tín hiệu xung của kênh A sẽ có cùng tần số với tín hiệu xung của kênh B, tuy nhiên do cảm biến của hai kênh được đặt lệch nhau nên tín hiệu kênh B thường sẽ lệch pha 90º so với tín hiệu kênh A. Từ đặc điểm này việc phối hợp đọc tín hiệu từ cả hai kênh có thể xác định được chiều quay của động cơ.



Hình 3.18 Sự lệch pha của hai kênh A và B trong encoder

Hình 3.18 là tín hiệu xung của hai kênh A và B khi động cơ quay theo chiều thuận (chiều quay mặc định khi cấp hai đầu điện áp đúng với thứ tự chân kết nối trên động cơ). Khi mắt thu của kênh A bị che (lúc tín hiệu ở kênh A đi từ mức cao xuống thấp) thì mắt thu của kênh B vẫn còn nhận được tín hiệu qua khe hở của đĩa quay encoder, tức là tín hiệu tại thời điểm đó ở kênh B đang ở mức thấp. Ngược lại, khi động cơ quay ngược chiều mặc định, tín hiệu xung của hai kênh sẽ đi từ phải sang trái theo như Hình 3.18, khi đó tại cạnh xuống của kênh A thì tín hiệu ở kênh B đang ở mức cao.

Để bắt được thời điểm tín hiệu ở kênh A đi từ cao xuống thấp (HIGH sang LOW), thiết lập ngắt (interrupt) trên Arduino ở chế độ RISING như sau:

```
attachInterrupt(interrupt, ISR, mode);
```

Trong đó:

- `interrupt` là số thứ tự ngắt (0 và 1 tương ứng với chân 2 và chân 3).

- `ISR` là tên hàm xử lí mà chương trình sẽ gọi tới khi có sự kiện ngắt.
- `mode` là kiểu kích hoạt ngắt, bao gồm 4 kiểu sau:
  - LOW: kích hoạt khi chân đọc tín hiệu ngắt ở mức thấp.
  - HIGH: kích hoạt khi chân đọc tín hiệu ngắt ở mức cao.
  - FALLING: kích hoạt khi chân đọc tín hiệu ngắt chuyển từ mức cao sang mức thấp.
  - RISING: kích hoạt khi chân đọc tín hiệu ngắt chuyển từ mức thấp sang mức cao.

Khi tín hiệu ở kênh A đi từ mức cao sang mức thấp, chương trình sẽ gọi đến hàm xử lý. Về cơ bản hàm này sẽ đọc tín hiệu ở kênh B để xác định chiều quay của động cơ, sau đó đếm lên hoặc đếm xuống giá trị `tick_count` tùy vào chiều quay của động cơ. Sự tăng giảm của giá trị này trong từng khoảng thời gian nhất định thể hiện cho tốc độ quay của động cơ tại những thời điểm đó.

Câu lệnh đọc tín hiệu digital như sau:

```
int val;
val = digitalRead(pinNumber);
```

Giá trị trả về là HIGH hoặc LOW và phải được lưu trong một biến kiểu số nguyên (int) và kiểu dữ liệu tương ứng trên ROS (để con số này có thể truyền đến các node trong ROS) là `std_msgs/Int16`.

3.3.3. Giao tiếp với ROS

Để có thể giao tiếp với Arduino, ROS cần được cài đặt các thư viện như sau:

```
sudo apt-get install ros-melodic-rosserial-arduino
sudo apt-get install ros-melodic-rosserial
```

Bên cạnh đó, môi trường của Arduino cũng cần phải được cài đặt thư viện `ros_lib`. Để thuận tiện nhất, Arduino IDE sẽ được cài đặt trên Ubuntu.

```
cd <sketchbook>/libraries
rosrun rosserial_arduino make_libraries.py .
```

Trong đó, `sketchbook` là thư mục mà Linux Arduino lưu các sketch (bản phác thảo, cụ thể là mã nguồn lập trình Arduino) có phần mở rộng là `.ino`. Mặc định, `sketchbook` sẽ nằm ở `~/Arduino/libraries`.

Để Arduino có thể gửi dữ liệu bên trong ROS, chương trình của Arduino phải publish một topic có kèm data lên ROS để các node có thể subcribe (lắng nghe). Phương tiện để thực hiện việc tạo một publisher hay subcriber là NodeHandle, vì vậy cần khởi tạo một NodeHandle:

```
#include <ros.h>
ros::NodeHandle nh;
```

Tiếp theo cần khởi tạo Publisher với tên của publisher, tên topic và dữ liệu:

```
std_msgs::Int16 tick_count;
ros::Publisher pub("ticks", &tick_count);
```

Sau đó, ở hàm `setup` cần thiết lập chung cho ROS NodeHandle:

```
nh.getHardware()->setBaud(57600);
nh.initNode();
nh.advertise(pub);
```

Cuối cùng ở hàm `loop`, publisher sẽ publish dữ liệu như sau:

```
void loop() {
  // ...
    pub.publish( &tick_count );
  // ...
  nh.spinOnce();
}
```

Theo ví dụ trên, trong hàm `loop`, node `serial_arduino` sẽ publish một dữ liệu là một số nguyên 16 bit `tick_count`.

3.4. Xây dựng hệ thống RTAB-Map trên ROS

3.4.1. Thiết lập môi trường

Trình tự thiết lập bao gồm:

Bước 1. Cài đặt hệ điều hành Ubuntu

Bước 2. Cài đặt hệ điều hành ROS

Bước 3. Cài đặt các thư viện, phụ thuộc (dependences), ...

Bước 4. Thiết lập workspace

Dưới đây sẽ trình bày các bước thiết lập môi trường cho thuật toán RTAB-Map sử dụng cảm biến quét laser RP-Lidar a1m8, cảm biến hình ảnh độ sâu (rgbd) camera Orbbec Astra 3D Camera và cảm biến tốc độ Encoder hoạt động trên hệ điều hành ROS mã phiên bản Melodic Morenia được cài đặt trên môi trường Linux với hệ điều hành Ubuntu Bionic (hay được biết đến là phiên bản Ubuntu 18.04).

Cài đặt hệ điều hành Ubuntu (trên một laptop)

- Bước 1: tải về file .iso của phiên bản Ubuntu 18.04 từ nhà sản xuất tại https://releases.ubuntu.com/18.04/ubuntu-18.04.6-desktop-amd64.iso.
- Bước 2: tạo một phân vùng (partition) có định dạng FAT32 có kích thước tối thiểu 3GB và tạo một khoảng trống (space) có kích thước tối thiểu 20GB (không tạo phân vùng).
- Bước 3: giải nén file .iso vào phân vùng có định dạng FAT32 vừa tạo.
- Bước 4: khởi động lại máy tính, vào chế độ BIOS, chọn khởi động bằng UEFI và chọn đúng phân vùng FAT32 đã tạo
- Bước 5: chọn Trying with Ubuntu và tiến hành các bước cài đặt trên giao diện của Ubuntu

Cài đặt hệ điều hành ROS

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

```
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc
 | sudo apt-key add –
sudo apt update
sudo apt install ros-melodic-desktop-full
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt install python-rosinstall python-rosinstall-generator
 python-rosdep python-wstool build-essential
sudo rosdep init && rosdep update
```

Cài đặt các phụ thuộc

```
sudo apt install ros-melodic-rtabmap ros-melodic-rtabmap-ros
```

Cài đặt thư viện RTABMAP

```
git clone https://github.com/introlab/rtabmap.git rtabmap
cd rtabmap/build && cmake ..
make -j6
sudo make install
```

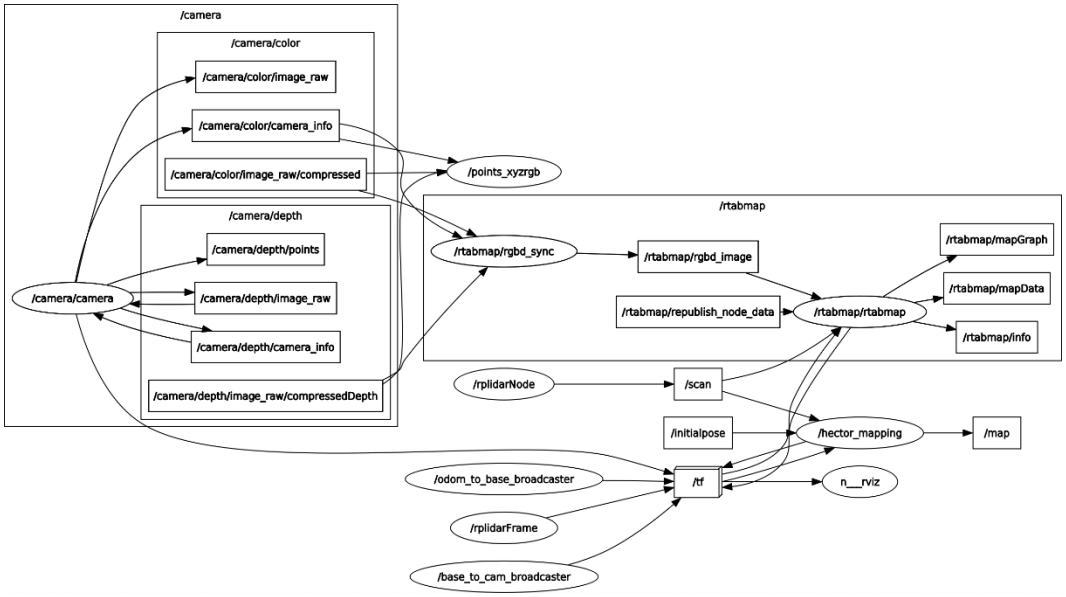
Thiết lập workspace

```
mkdir -p ~/catkin_ws && cd ~/catkin_ws
git clone https://github.com/introlab/rtabmap_ros.git src/rtabmap_ros
git clone https://github.com/orbbec/ros_astra_camera
git clone https://github.com/Slamtec/rplidar_ros
catkin_make -j4
```

3.4.2. Vận hành hệ thống

Để điều hướng cho robot di chuyển đến điểm đích như mong muốn, cần phải xác định tọa độ và góc quay khi robot di chuyển. Dữ liệu đầu vào của hệ thống chương trình trong đề tài này kết hợp dữ liệu đọc từ các cảm biến tốc độ encoder, cảm biến camera 3D và cảm biến quét laser 2D.

Hệ thống chương trình trên ROS được xây dựng, phát triển và thực thi trên các node riêng biệt với các chức năng khác nhau. Sự trao đổi dữ liệu giữa các node được thể hiện qua các message theo từng topic. Bên cạnh đó, ROS cung cấp các công cụ kiểm tra hoạt động và theo dõi kết quả thực hiện bởi các node một cách trực quan. Sơ đồ hệ thống hoạt động trên ROS gồm các node và cách truyền nhận dữ liệu giữa các node được mô tả như trong Hình 3.19 với các node được thể hiện bằng những hình ellipse, còn các topic được thể hiện bằng hình chữ nhật.



Hình 3.19 Sơ đồ tổ chức các node trên ROS Melodic

Trong hệ thống được thiết kế, thông tin của bản đồ cục bộ của robot được xác định qua các point cloud (đám mây điểm) từ cảm biến Laser và Camera thông qua node "/rplidarNode" và "/camera/camera", dữ liệu này được node *rtabmap* đối chiếu, so sánh với bản đồ có sẵn để tìm ra vị trí của robot trên bản đồ đó.

Trình tự vận hành như sau:

Bước 1. Khởi động Master bằng `roscore`

Bước 2. Launch node `camera` từ package `astra_camera`

Bước 3. Launch node `rplidarNode` từ pakage `rplidar_ros`

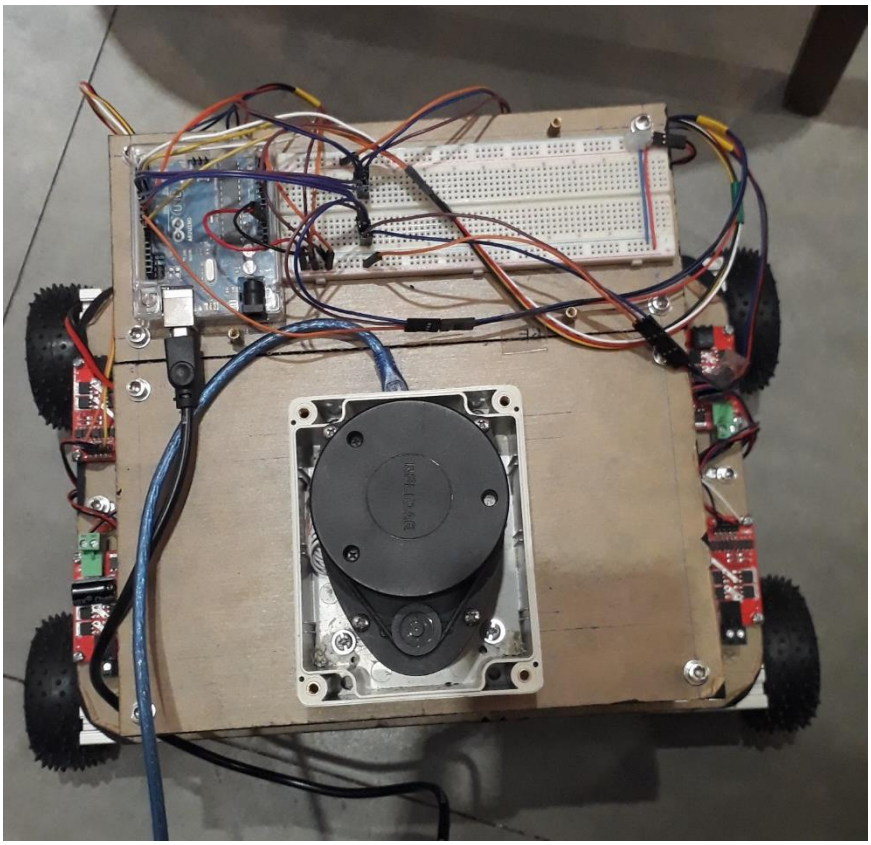
Bước 4. Launch các node liên quan đến thuật toán RTAB-Map (gồm node `rgbd_sync` từ package `nodelet`, node `rtabmap` từ package `rtabmap_slam`, node `rtabmap_viz` từ package `rtabmap_viz`, hoặc node `rviz` từ pakage `rviz` thay thế cho `rtabmap_viz`)

Bước 5. Cấu hình các thông tin như tên topic của cảm biến, tên frame hiển thị trên công cụ quan sát, …

Bước 6. Lưu bản đồ thành file `map.db`

3.5. Kết quả đạt được

3.5.1. Hoàn thiện robot



Hình 3.20 Robot sau khi đã hoàn thiện phần điện tử và thiết kế cơ khí

3.5.2. Đọc dữ liệu từ RPLidar

Để đọc dữ liệu từ cảm biến, ROS phải khởi động node "/rplidarNode" từ package **`rplidar_ros`** và thao tác này được cụ thể hóa bằng lệnh `roslaunch` như sau:

```
roslaunch rplidar_ros rplidar.launch
```

Để theo dõi quá trình đọc dữ liệu, ta có thể dùng công cụ Rviz. Ở cửa sổ bên trái, chọn Add → LaserScan với dữ liệu từ Lidar và thiết lập lại topic của Lidar là "/scan".

3.5.3. Giao tiếp Arduino với ROS

Để kiểm chứng giao tiếp giữa Arduino và ROS, thực hiện quan sát tốc độ quay của động cơ và điều khiển hướng di chuyển của robot thông qua các thao tác trên giao diện của Ubuntu.

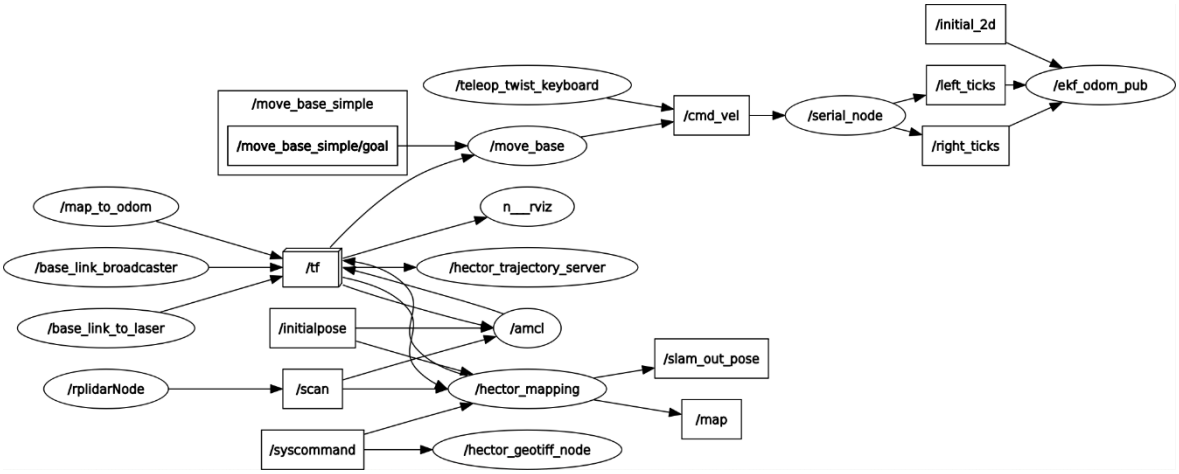
Arduino sẽ giao tiếp với ROS bằng **`rosserial`** thông qua kết nối USB. Phiên giao tiếp sẽ diễn ra hai chiều. Ở chiều giao tiếp thứ nhất, Arduino sẽ publish dữ liệu thu được từ encoders lên hai topic là "/left_ticks" và "/right_ticks". Từ phía ROS sẽ thực hiện tạo hai topic này, có thể kiểm tra xem ROS có nhận được dữ liệu hay chưa bằng lệnh sau:

```
rostopic echo /topic_name
```

Ở chiều giao tiếp còn lại, Arduino sẽ subscribe (lắng nghe) một topic có tên là "/cmd_vel" để nhận lệnh điều hướng từ giao diện của ROS. Từ phía ROS, khởi động một node điều khiển robot bằng bàn phím như sau:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Khi hệ thống được khởi động thành công, ta có tổ chức các node như sau:



Hình 3.21 Sơ đồ tổ chức các node khi thực hiện giao tiếp Arduino với ROS

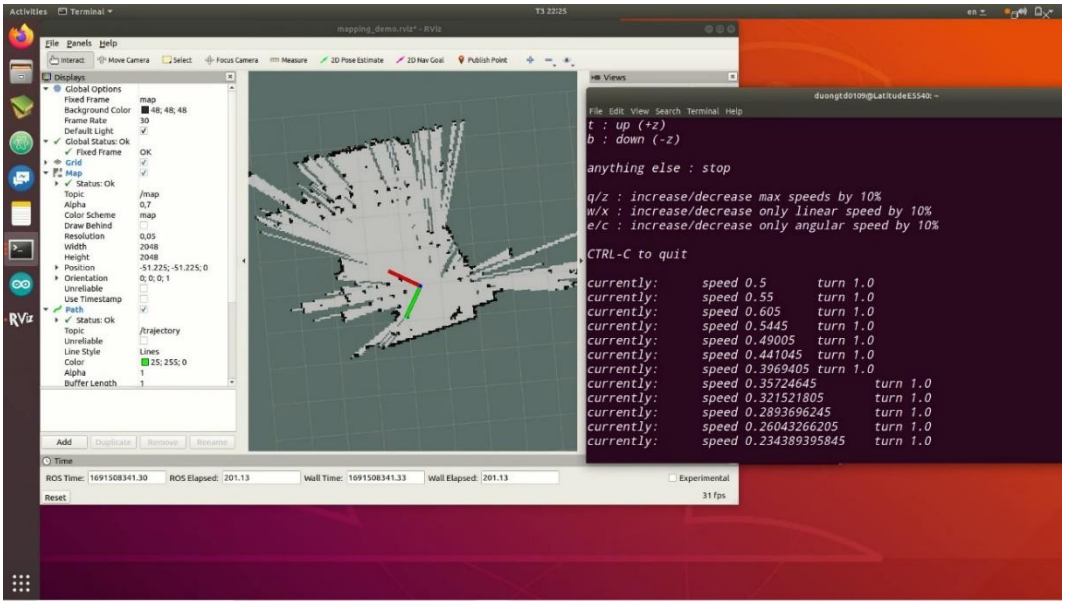
3.5.4. Mapping với RP-Lidar và wheel encoders

Như đã trình bày ở phần 1.1.4, RTAB-Map cho phép việc thực hiện thuật toán trở nên linh hoạt với nhiều lựa chọn cảm biến khác nhau mà không ảnh hưởng đến mục tiêu của thuật toán. Khi chỉ sử dụng RP-Lidar và Odometry để thực hiện giải thuật RTAB-Map mà không có camera, hệ thống sẽ mất đi khả năng phát hiện vòng lặp (detect loop closure) bằng hình ảnh, kết quả tối ưu hóa bản đồ (Graph Optimization) sẽ không hiệu quả bằng khi có camera, tuy nhiên nếu xử lý tốt phần điều khiển động cơ, điều hướng và tính toán Odometry (sai số đạt tối thiểu) thì kết quả cuối cùng sẽ không có nhiều sai lệch.

Để thực hiện mapping, sử dụng phương pháp điều khiển robot tương tự như đã trình bày ở phần 3.5.3, kết hợp với file `.launch` đã được cấu hình cho phù hợp với các thiết bị sử dụng như sau:

```
roslaunch hector_slam_launch tutorial.launch
```

Cửa sổ công cụ Rviz hiện ra, thực hiện quan sát kết quả mapping sau khi điều khiển robot đi xung quanh không gian trong nhà. Kết quả thu được trên Rviz như hình bên dưới. Khi đã thấy được hình ảnh mapping mong muốn, save map để sử dụng cho localization.



Hình 3.22 Kết quả thực hiện mapping với lidar và hector_slam

CHƯƠNG 4. **KẾT LUẬN**

Khoá luận đã trình bày giải pháp thiết kế hệ thống SLAM - tái tạo môi trường và định vị robot di động dựa trên hệ điều hành ROS, bao gồm cả phần cứng và phần mềm. Hệ thống thử nghiệm được tích hợp lên một robot tự hành leo cầu thang và thử nghiệm hoạt động ở môi trường ngoài trời và trong nhà. Các kết quả cho thấy, robot có khả năng thu thập dữ liệu từ môi trường xung quanh, xây dựng bản đồ 3D và định vị vị trí của robot trên bản đồ đã tạo. Các kết quả này là dữ liệu cơ sở cho các bước điều hướng di chuyển, tính toán quỹ đạo chuyển động cho robot thực hiện các nhiệm vụ cụ thể như robot tự hành hoạt động trong các nhà máy, ứng dụng trong công việc vận chuyển hàng hóa trong nhà, Logictis… Đặc biệt bên cạnh đó, bản đồ 3D có ý nghĩa với các robot tự hành có tích hợp các cơ cấu chấp hành, cánh tay máy robot … Bên cạnh đó, giải pháp thiết kế hệ thống trên hệ điều hành ROS cho phép người dùng có thể linh hoạt trong việc điều chỉnh, cấu trúc lại theo từng ứng dụng cụ thể.

Cụ thể, nhóm đã thu được các kết quả như sau:

- Hoàn thành việc nghiên cứu thuật toán SLAM 3D.
- Hoàn thành phần cơ khí robot.
- Sử dụng module mạch cầu điều khiển động cơ một chiều.
- Đọc dữ liệu cảm biến tốc độ encoder bằng STM32, Arduino.
- Giao tiếp Arduino với ROS.
- Đọc dữ liệu cảm biến quét laser, camera và thể hiện trên ROS.

CHƯƠNG 5. **HƯỚNG PHÁT TRIỂN**

Sau khi hoàn thiện mô hình, nhóm nhận thấy còn khá nhiều điểm có thể cải thiện để tăng khả năng chắc chắn cho mô hình cũng như tăng độ chính xác của thuật toán.

Những hướng phát triển robot leo cầu thang của nhóm như sau:

- Tiếp tục thực hiện localization.
- Thay thế các vật liệu nhẹ hơn, tối ưu kích thước robot tuỳ mục đích sử dụng.
- Cải tiến động cơ và bánh xe để có thể linh hoạt di chuyển trong nhiều địa hình khác nhau với độ bám tốt hơn và tăng tốc độ di chuyển.
- Tích hợp thêm cảm biến IMU, GPS, ... để tăng độ chính xác hoặc đáp ứng các mục đích khác.

TÀI LIỆU THAM KHẢO

[1] Sinan Oğuzhan Başkurt, Akif Ekrekli, Veysel Bitgül, Kaan Han Elmalı, "Autonomous Stair Climbing Robot Final Report," pp. 15-16, May 2020.

[2] Stefan Hensel, Marin B. Marinov, Markus Obert, "3D LiDAR Based SLAM System Evaluation with Low-Cost Real-Time Kinematics GPS Solution," *Computation 2022*, vol. 10, no. 9, p. 154, 2022.

[3] "koide3/hdl_graph_slam: 3D LIDAR-based Graph SLAM," [Online]. Available: https://github.com/koide3/hdl_graph_slam. [Accessed 10 May 2023].

[4] Tom Duckett, Stephen Marsland, Jonathan Shapiro, "Learning globally consistent maps by relaxation," in *IEEE*, San Francisco, 2000.

[5] Swas Oajsalee, Suradet Tantrairatn, Sorada Khaengkarn, "Study of ROS Based Localization and Mapping for Closed Area Survey," in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, Singapore, IEEE, 2019, pp. 24-28.

[6] Zhang Xuexi, Lu Guokun, Fu Genping, Xu Dongliang, Liang Shiliu, "SLAM Algorithm Analysis of Mobile Robot Based on Lidar," in *2019 Chinese Control Conference (CCC)*, Guangzhou, China, IEEE, 2019, pp. 4739-4745.

[7] MathieuLabbe, "Wiki: rtabmap_ros," 18 April 2023. [Online]. Available: http://wiki.ros.org/rtabmap_ros. [Accessed 10 May 2023].

[8] A. Lorente Rogel, "Component-based Slam in RTAB-Map," October 2021. [Online]. Available: http://essay.utwente.nl/88715/. [Accessed 10 May 2023].