**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**VÕ ANH TUẤN**

**TRẦN HOÀNG PHƯƠNG**

**ĐỒ ÁN MÔN HỌC**

**LIDAR MAPPING SYSTEM**

**TP. HỒ CHÍ MINH, 2019**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**

**VÕ ANH TUẤN – 16521388**

**TRẦN HOÀNG PHƯƠNG - 16520973**

**ĐỒ ÁN MÔN HỌC**

**LIDAR MAPPING SYSTEM**

**GIẢNG VIÊN HƯỚNG DẪN**

**PHAN ĐÌNH DUY**

**TP. HỒ CHÍ MINH, 2019**

MỤC LỤC

[Chương 1. TỔNG QUAN 6](#_Toc28558785)

[1.1. Lý do thực hiện đồ án: 6](#_Toc28558786)

[1.2. Mục tiêu đồ án: 6](#_Toc28558787)

[1.3. Phương pháp thực hiện: 6](#_Toc28558788)

[1.3.1. Dữ liệu đầu vào: 6](#_Toc28558789)

[1.3.2. Quá trình xữ lý: 6](#_Toc28558790)

[1.3.3 Dữ liệu đầu ra: 6](#_Toc28558791)

[1.4. Các nội dung của đồ án: 7](#_Toc28558792)

[1.4.1. Tìm hiểu kiến thức cơ bản về ROS: 7](#_Toc28558793)

[Chương 2. CƠ SỠ LÝ THUYẾT 7](#_Toc28558794)

[2.1. Tổng quan về ROS: 7](#_Toc28558795)

[2.1.1. Khái niệm: 7](#_Toc28558796)

[2.1.2. Cơ bản về hoạt động của ROS: 8](#_Toc28558797)

[2.2. Tổng quan về RPLIDAR và RASPBERRY: 11](#_Toc28558798)

[2.2.1. Giới thiệu : 11](#_Toc28558799)

[2.3. Sơ đồ kết nối và hình ảnh của Prototype: 14](#_Toc28558800)

[2.3.1. code Python: 14](#_Toc28558801)

[2.3.2. Vật liệu sữ dụng trong đồ án: 17](#_Toc28558802)

[Chương 3. PHÂN TÍCH VÀ HIỆN THỰC CHƯƠNG TRÌNH 19](#_Toc28558803)

[3.1.Phân tích lệnh thiết lập RPLidar và gói Hector SLAM: 19](#_Toc28558804)

[3.1.1. Phân tích lệnh thiết lập RPLidar: 19](#_Toc28558805)

[3.1.2. Phân tích lệnh thiết lập cho gói Hector SLAM: 22](#_Toc28558806)

[3.2.Tổng quan về hệ thống xây dựng bản đồ và định vị dựa trên LIDAR và gói Hector\_SLAM : 23](#_Toc28558807)

[3.3. Tổng quan về sơ đồ hoạt động của chương trình: 24](#_Toc28558808)

[Chương 4. KẾT QUẢ THỰC NGHIỆM 25](#_Toc28558809)

[4.1. Kết quả phần cứng 25](#_Toc28558810)

[4.2.Kết quả đầu ra 26](#_Toc28558811)

[4.2.1. Phân tích kết quả đầu ra: 26](#_Toc28558812)

[4.2.2. Các lỗi phát sinh: 30](#_Toc28558813)

[Chương 5. KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN 32](#_Toc28558814)

[5.1. Kết luận 32](#_Toc28558815)

[5.2. Phương hướng phát triển 32](#_Toc28558816)

**DANH MỤC HÌNH ẢNH**

[Hình 1 - Những robot được hỗ trợ, bắt đầu từ Pepper(a), REEM-C(b), Turtlebot(c), Robotnaut(d) và Universal robot(e). 7](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562789)

[Hình 2 - Sensor được hỗ trợ bởi ROS 8](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562790)

[Hình 3 - Bố trí trong ROS 8](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562791)

[Hình 4 - Các Node trong ROS 9](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562792)

[Hình 5 - RPLIDAR A1 12](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562793)

[Hình 6 - Kết quả trả về từ LIDAR được gắn dưới máy bay 13](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562794)

[Hình 7- RASPBERRY PI 3 MODEL B 14](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562795)

[Hình 8 - Sơ đồ mô tả code Python 16](#_Toc28562796)

[Hình 9 - Quạt tản nhiệt 24x24mm 17](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562797)

[Hình 10 - Module giảm áp XL4005 17](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562798)

[Hình 11 - Module cầu H mini MDL31 17](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562799)

[Hình 12 - Khung xe in 3D 18](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562800)

[Hình 13 - Pin Lipo 3 cell 18](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562801)

[Hình 14 - Thiết kế PCB 19](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562802)

[Hình 15 - Lệnh ls -l /dev |grep ttyUSB 20](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562803)

[Hình 16 - Lệnh roslaunch rplidar\_ros rplidar.launch 20](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562804)

[Hình 17 21](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562805)

[Hình 18 - Lệnh roslaunch hector\_slam\_launch tutorial.launch 22](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562806)

[Hình 19 - Hệ thống xây dựng bản đồ và định vị dựa trên LIDAR và gói Hector\_SLAM 23](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562807)

[Hình 20 - Sơ đồ hoạt động của chương trình 24](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562808)

[Hình 21 - Kết quả phần cứng 25](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562809)

[Hình 22 - Hình thực tế của hộ dân 26](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562810)

[Hình 23 26](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562811)

[Hình 24 27](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562812)

[Hình 25 27](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562813)

[Hình 26 - Kết quả trả về của sơ đồ hộ dân 28](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562814)

[Hình 27 28](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562815)

[Hình 28 - Khoảng cách giữa RPLIDAR và vật thể 29](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562816)

[Hình 29 - Phương pháp lượng giác 30](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562817)

[Hình 30 – Hình thực tế của hộ dân 31](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562818)

[Hình 31 – Lỗi phát sinh 31](file:///C:\Users\ADMIN\Desktop\LIDAR%20MAPPING%20SYSTEM.docx#_Toc28562819)

Chương 1. TỔNG QUAN

1.1. Lý do thực hiện đồ án:

* Xây dựng một robot dò vật cản sử dụng công nghệ LIDAR , tạo một map môi trường 2D, đồng thời kết hợp xử lý ảnh trong tương lai.

1.2. Mục tiêu đồ án:

* Tìm hiểu về các kiến thức cơ bản về hệ điều hành Linux/Ubuntu, ngôn ngữ lập trình Python, sử dụng các package của RPLIDAR.
* Áp dụng các kiến thức đã tìm hiểu được để xây dựng một nguyên mẫu robot, tạo một map môi trường 2D.

1.3. Phương pháp thực hiện:

1.3.1. Dữ liệu đầu vào:

* Dữ liệu đầu vào được lấy trực tiếp từ cảm biến vật cản RPLIDAR, trả về các điểm quét từ laze.

1.3.2. Quá trình xữ lý:

* Dữ liệu được cảm biến RPLIDAR quét được được đưa vào RASPBERRY PI 3 B+ để xử lý.
* RASPBERRY sẽ nhận diện được dữ liệu và hiển thị lên màn hình thông qua VNC hoặc SSH, qua phần mềm RVIZ của ROS.
* Ngoài ra, RASPBERRY được nạp code Python để điều khiển cả robot thông qua bàn phím máy tính, quét hết diện tích môi trường.

1.3.3 Dữ liệu đầu ra:

* Chương trình sẽ xuất ra khung hình đã quét được, đưa ra môi trường 2D của diện tích cần quét.

1.4. Các nội dung của đồ án:

1.4.1. Tìm hiểu kiến thức cơ bản về ROS:

* Tham khảo các trang mạng về ROS
* Tìm cách cài đặt ROS trên nền UBUNTU Mate 18.04 và 16.04
* Học các Node và cách vận hành của ROS

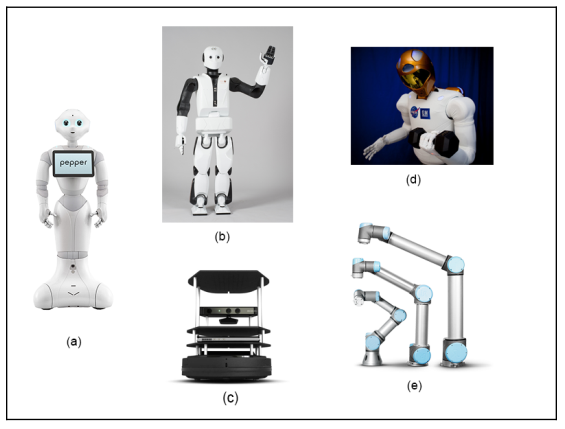
1.4.2. Tìm hiểu kiến thức cơ bản về RPLIDAR và RASPBERRY:

* Tìm hiểu trên trang chủ của RPLIDAR
* Tìm cách cài đặt UBUNTU Mate 18.04 và 16.04 trêm RASPBERRY PI3 B+

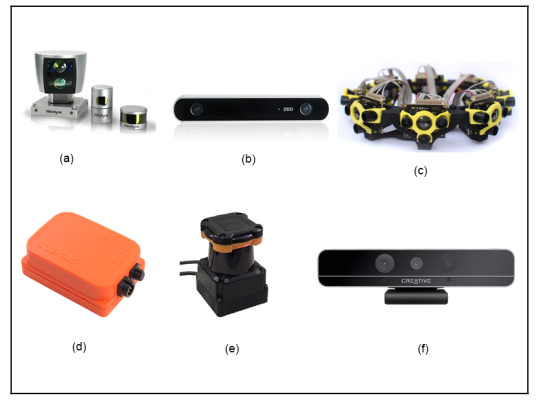
Chương 2. CƠ SỠ LÝ THUYẾT

2.1. Tổng quan về ROS:

2.1.1. Khái niệm:

* Robot operating system là một hệ thống phần mềm chuyên dụng để lập trình và điều khiển robot, bao gồm các công cụ để lập trình, hiển thị, tương tác trực tiếp với phần cứng, và kết nối cộng đồng robot trên toàn thế giới.ROS chạy trên nền Ubuntu, do đó để sử dụng ROS đầu tiền chúng ta phải cài Linux. Những Robot và sensor được hỗ trợ bởi ROS:

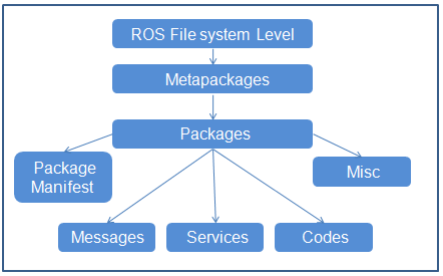
Hình 1 - Những robot được hỗ trợ, bắt đầu từ Pepper(a), REEM-C(b), Turtlebot(c), Robotnaut(d) và Universal robot(e).

* Ngoài ra, các sensor được hỗ trợ bởi ROS gồm có LIDAR, SICK laser lms1xx, lms2xx, Hokuyo, Kinect-v2, Velodyne..,

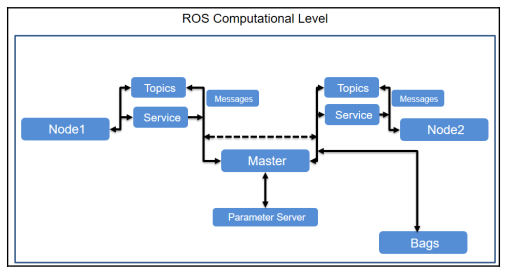
Hình 2 - Sensor được hỗ trợ bởi ROS

2.1.2. Cơ bản về hoạt động của ROS:

* Về cơ bản ROS files được bố trí và hoạt động như sau, từ trên xuống theo thứ tự như sau, metapackages, packages, packages manifest, Misc, messages, Services, codes:



Hình 3 - Bố trí trong ROS

* Trong đó gói tổng(Metapackages) là một nhóm các gói(packages) có liên quan tới nhau. Lấy ví dụ như, trong ROS có một gói tổng tên là ****Navigation****, gói này có chứa tất cả các gói liên quan tới việc di chuyển của robot, bao gồm di chuyển thân, bánh, các thuật toán liên quan như Kalman, Particle filter… Khi chúng ta cài đặt gói tổng, nghĩa là tất cả các gói con trong nó cũng được cài đặt. Gói(Packages), ở đây là mình dịch là gói cho dễ hiểu, khái niệm gói rất quan trọng, chúng ta có thể nói rằng cái gói chính là các nguyên tử cơ bản nhất tạo nên ROS. Trong một gói gồm có, ROSnode, datasets, configuration files, source files, tất cả được gói lại trong một “gói”. Tuy nhiên, mặc dù có nhiều thứ trong một, gói, nhưng để làm việc, chúng ta chỉ cần quan tâm 2 thứ trong 1 gói, đó chính là src folder, chứa source code của chúng ta, và file Cmake.txt, đây là nơi ta khai báo những thư viện cần thiết để thực thi(compile) code.
* Sự Tương tác giữa các node trong ROS

Hình 4 - Các Node trong ROS

ROS computation graph là bức tranh toàn cảnh về sự tương tác của các nodes, topics với nhau.

Ở hình trên ta có thể thấy Master chính là nodes kết nối tất cả các node còn lại, các nodes còn lại muốn giao tiếp với nhau phải thông qua node Master.

* Nodes: ROS nodes chỉ đơn giản là quá trình sử dụng ROS API để giao tiếp với nhau. Một robot có thể có rất nhiều nodes để thực hiện quá trình giao tiếp của nó. Ví dụ như, một con robot tự lái sẽ có những node sau, node đọc dữ liệu từ Laser scanner, Kinect camera, localization and mapping, node gửi lệnh vận tốc cho hệ thống bánh lái.
* Master: ROS master đóng vai trò như một node trung gian kết nối giữa các node khác nhau. Master bao quát thông tin về tất cả các nút chạy trong môi trường ROS. Nó sẽ trao đổi chi tiết của một nút với khác để thiết lập kết nối giữa chúng.

Sau khi trao đổi thông tin, giao tiếp sẽ bắt đầu giữa hai nút ROS. Khi bạn chạy một chương trình ROS, lúc nào ros\_master cũng phải chạy trước.

* Message: Các nút ROS có thể giao tiếp với nhau bằng cách gửi và nhận dữ liệu dưới dạng ROS mesage. ROS message là một cấu trúc dữ liệu được sử dụng bởi các nút ROS để trao đổi dữ liệu. Nó giống như là một protocol, định dạng thông tin gửi đi nữa các node, ví dụ như string, float, int…
* Topic: Một trong những phương pháp để giao tiếp và trao đổi tin nhắn giữa hai node được gọi là ROS Topic. ROS Topic giống như một kênh tin nhắn, trong kênh đó dữ liệu được trao đổi bằng ROS message. Mỗi chủ đề sẽ có một tên khác nhau tùy thuộc vào những thông tin mà nó sẽ phụ trách cung cấp. Một Node sẽ đăng bài(publish) thông tin cho một Topic

và một nút khác có thể đọc từ Topic bằng cách đăng ký(subcribe) với nó.

* Service: Service là một loại phương pháp giao tiếp khác với Topic. Topic sử dụng tương tác publish hoặc subcrible nhưng trong dịch vụ, nó tương tác theo kiểu request – response. Cái này y chang như bên network. Một nút sẽ hoạt động như một server,có một server chạy thường xuyên, chạy và khi Node client gửi yêu cầu dịch vụ cho server. Máy chủ sẽ thực hiện dịch vụ và gửi kết quả cho máy khách. Client node phải đợi cho đến khi máy chủ phản hồi với kết quả. Server đặc biệt hữu dụng khi chúng ta cần thực hiện một lệnh gì đó mà cần thời gian lâu để xử lý, vậy thì chúng ta để nó ở server, khi cần thì ta gọi.

2.2. Tổng quan về RPLIDAR và RASPBERRY:

2.2.1. Giới thiệu :

* Cảm biến Laser Radar
* Cảm biến Laser Radar (Lidar) RPLIDAR A1 được sản xuất bởi hãng SLAMTEC được sử dụng cho các ứng dụng phát hiện vật cản, lập bản đồ bằng tia Laser trong xe, robot tự hành, hệ thống chống trộm, ..., cảm biến có độ ổn định và độ chính xác cao.
* Cảm biến Laser Radar (Lidar) RPLIDAR A1 sử dụng giao tiếp UART nên có thể dễ dàng giao tiếp với Vi điều khiển, Máy tính nhúng hoặc kết nối máy tính qua mạch chuyển USB-UART và phần mềm đi kèm, cảm biến có khả năng quét xa với khoảng cách lên đến 12m, tần số tối đa 10Hz với 8000 samples per time, phù hợp cho vô số các ứng dụng khác nhau.

Thông số kỹ thuật:

• Điện áp sử dụng: 5VDC

• Chuẩn giao tiếp: UART

• Phương pháp phát hiện vật cản: Laser

• Khoảng cách phát hiện vật cản tối đa: 12m

• Góc quay: 360 độ.

• Tốc độ lấy mẫu tối đa: 8000 Samples per time.

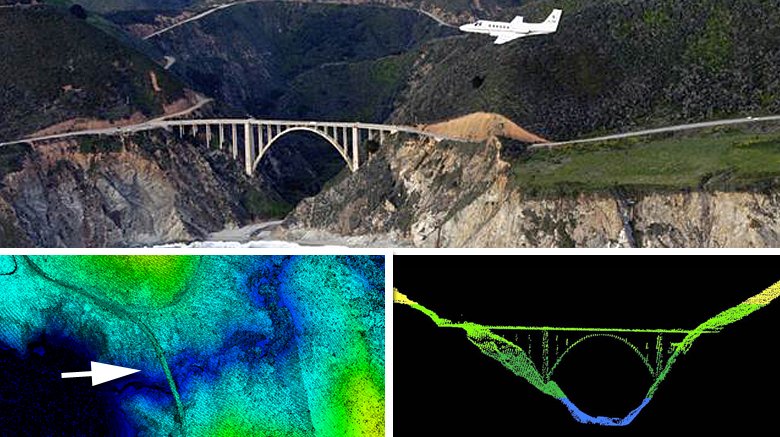
• Tần số quét tối đa: 10Hz

• Kích thước: 71 x 97mm

Hình 5 - RPLIDAR A1

* Công nghệ LIDAR (Laser Imaging, Detection, and Ranging)
* LIDAR có thể phát ra tối đa 200.000 xung laser trong mỗi giây. Một bộ lidar cơ bản bao gồm một máy phát laser, một máy scan, một bộ thu nhận GPS được tùy biến.Khi một chùm laser được chiều vào một điểm trên mặt đất, chùm sáng này sẽ bị phản xạ lại. Một cảm biến sẽ thu nhận thông tin của chùm phản xạ để đo khoảng cách dựa theo thời gian di chuyển của xung laser. Kết hợp với dữ liệu về vị trí và phương hướng từ hệ thống GPS cũng như bộ đo quán tính, bộ quét góc, dữ liệu sẽ được đưa ra thành một tập hợp các điểm, gọi là "point cloud". Mỗi point cloud sẽ có tọa độ xác định trong không gian ba chiều (bao gồm kinh độ, vĩ độ và cao độ) tương ứng với vị trí của nó trên bề mặt Trái

Đất. Các điểm này sau đó được đem đi dựng thành mô hình.



Hình 6 - Kết quả trả về từ LIDAR được gắn dưới máy bay

* RASPBERRY PI3 Module B

Thông số kỹ thuật chi tiết:

* Sản xuất tại: nhà máy Sony tại Anh (Made in UK), chính hãng RS Components.
* CPU Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
* GPU Broadcom Videocore-IV
* RAM 1GB LPDDR2 SDRAM
* Networking: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
* Bluetooth 4.2, Bluetooth Low Energy (BLE)
* Storage: Micro-SD
* GPIO: 40-pin GPIO header, populated
* Ports: HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
* Dimensions: 82mm x 56mm x 19.5mm, 50g



Hình 7- RASPBERRY PI 3 MODEL B

2.3. Sơ đồ kết nối và hình ảnh của Prototype:

2.3.1. code Python:

* Pygame: một thư viện miễn phí của ngôn ngữ lập trình python dùng để xây dựng ứng dụng đa phương tiện. Thư viện rất dễ sữ dụng và chỉ dùng cho mục đích đơn giản nhanh chống. Không cần GUI để sữ dụng hết tất cả các chức năng. Trong đồ án này thư viện chỉ có nhiệm vụ nhận sự kiện từ laptop.
* PRi.GPIO là một thư viện hổ trợ các lớp để điều khiển GPIO của Raspberry Pi đơn giản
* Sữ dụng pygame là một biện pháp thay thế đơn

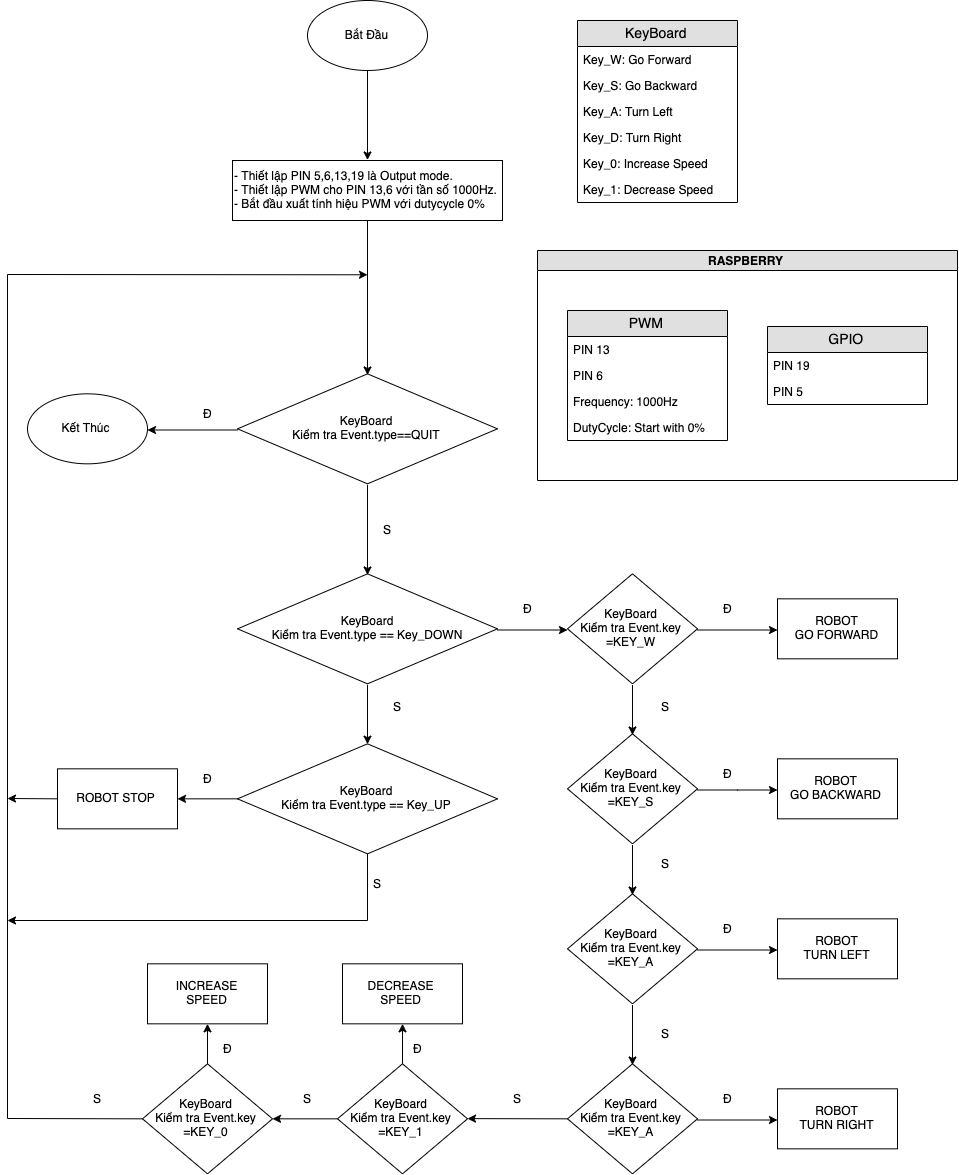
giản để điều khiển động cơ bằng Raspberry. Chương trình điều khiển động cơ cho phép điều khiển động cơ thông qua bàn phím mày tính. Đó là nhờ vào pygame đã nhận những sự kiện từ bàn phím từ đó dùng GPIO để điều xung PWM giúp động cơ quay nhanh chậm và theo chiều nhất định tuỳ theo phím được nhấn.

* Sau khi hoàn thành, Robot có thể tiến (phím W), lui

(phím S), rẻ trái (phím A) và rẻ phải (phím D). Do sữ dụng động cơ rẻ tiền và không có định hướng về độ chính xác khi điều khiển động cơ nên nhóm đã không hướng đến PID(Proportional, Integral, Derivative). Thay vào đó sữ dụng hai phím là phím “1” và phím “0” để tăng giãm vận tốc cho phù hợp.

* Sơ đồ mô tả code dưới đây mô tả hoạt động của việc di

chuyển của phương tiện điều khiển bằng bàn phím của laptop hoặc PC, màn hình hiển thị và điều khiển thông qua app NoMachine.

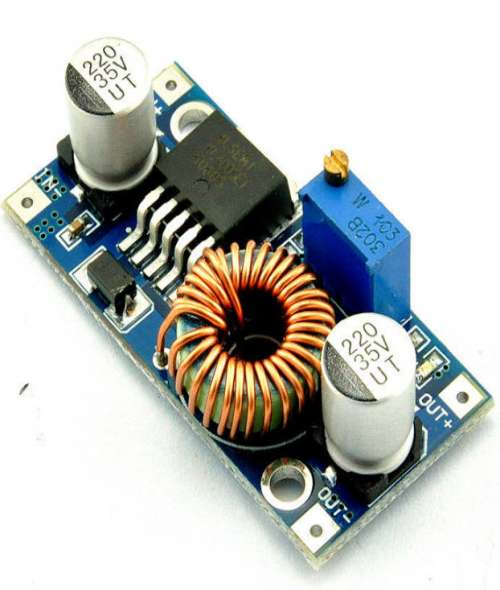


Hình 8 - Sơ đồ mô tả code Python

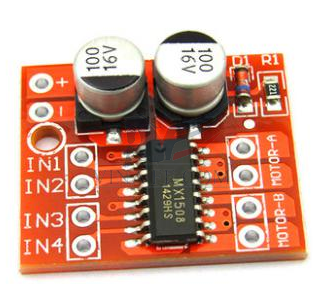
2.3.2. Vật liệu sữ dụng trong đồ án:

* Quạt mini 24\*24mm (tản nhiệt)

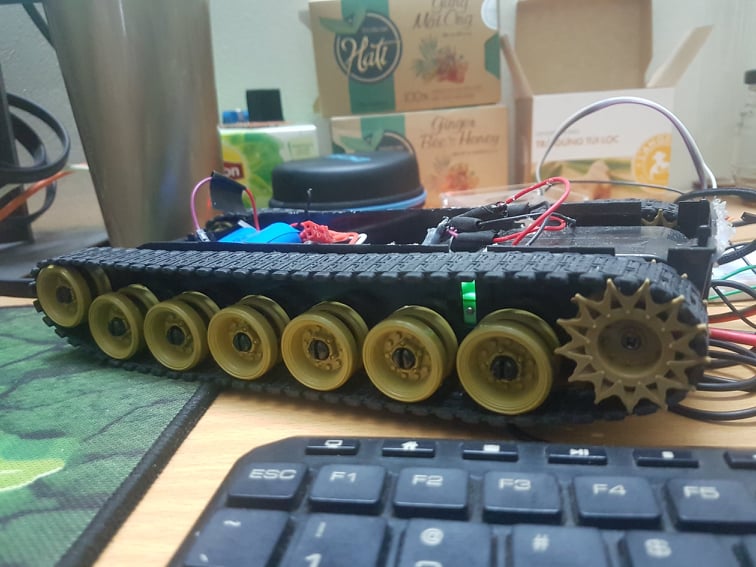
Hình 9 - Quạt tản nhiệt 24x24mm

* Module XL4005 (Chỉnh áp 5V cho toàn phương tiện)

Hình 10 - Module giảm áp XL4005

* Mạch cầu H mini MDL31 (Điều khiển động cơ)

Hình 11 - Module cầu H mini MDL31

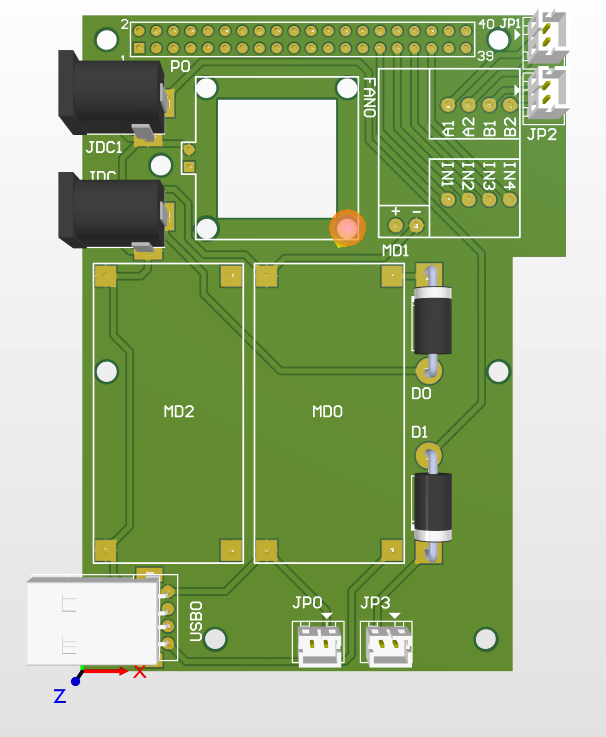
* Khung xe

Hình 12 - Khung xe in 3D

* Pin LIPO 3 cell



Hình 13 - Pin Lipo 3 cell

* Mạch PCB

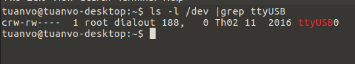
Hình 14 - Thiết kế PCB

Chương 3. PHÂN TÍCH VÀ HIỆN THỰC CHƯƠNG TRÌNH

3.1.Phân tích lệnh thiết lập RPLidar và gói Hector SLAM:

3.1.1. Phân tích lệnh thiết lập RPLidar:

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

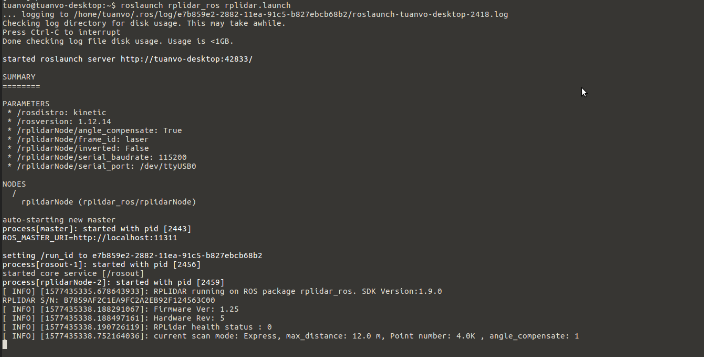
Lệnh kiểm tra cổng Serial nào đang được sử dụng trên RASPBERRY, trong trường hợp này là ttyUSB0.

Hình 15 - Lệnh ls -l /dev |grep ttyUSB

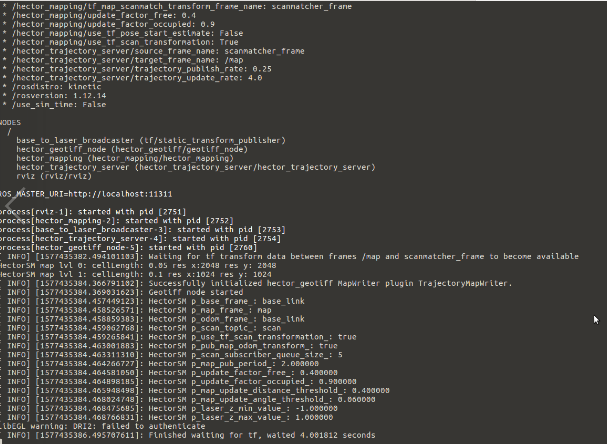
**sudo chmod 666 /dev/ttyUSB0**

Lệnh chuyển chế độ ghi, đọc hoặc thực thi file. Trong trường hợp này mode 666 chỉ cho đọc hoặc ghi file, không cho thực thi)

**roslaunch rplidar\_ros rplidar.launch**

Chạy node rpilidar

Hình 16 - Lệnh roslaunch rplidar\_ros rplidar.launch

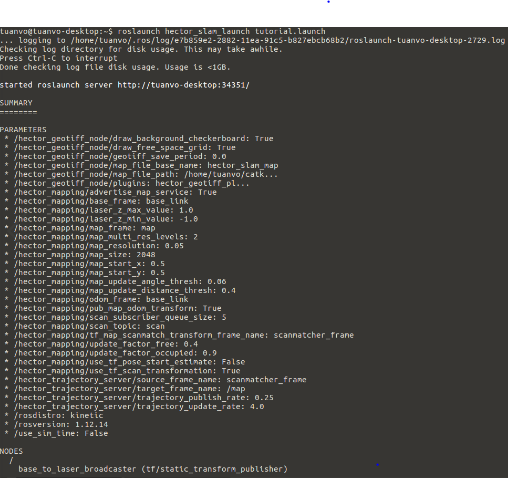
Sau khi thực thi lệnh, RASPBERRY sẽ tiến hành khởi động server roslaunch. Sau đó, thiết bị RPLIDAR được chạy trên nền của gói rplidar\_ros.

Hình 17

Ngoài ra còn có lệnh **roslaunch rplidar\_ros view\_rplidar.launch** để hiển thị kết quả quét không lưu điểm ảnh trong phầm mềm của ROS rviz.

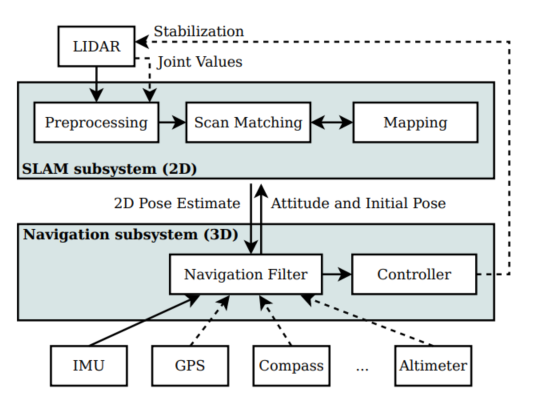
### 

### 3.1.2. Phân tích lệnh thiết lập cho gói Hector SLAM:

**roslaunch hector\_slam\_launch tutorial.launch**

Hình 18 - Lệnh roslaunch hector\_slam\_launch tutorial.launch

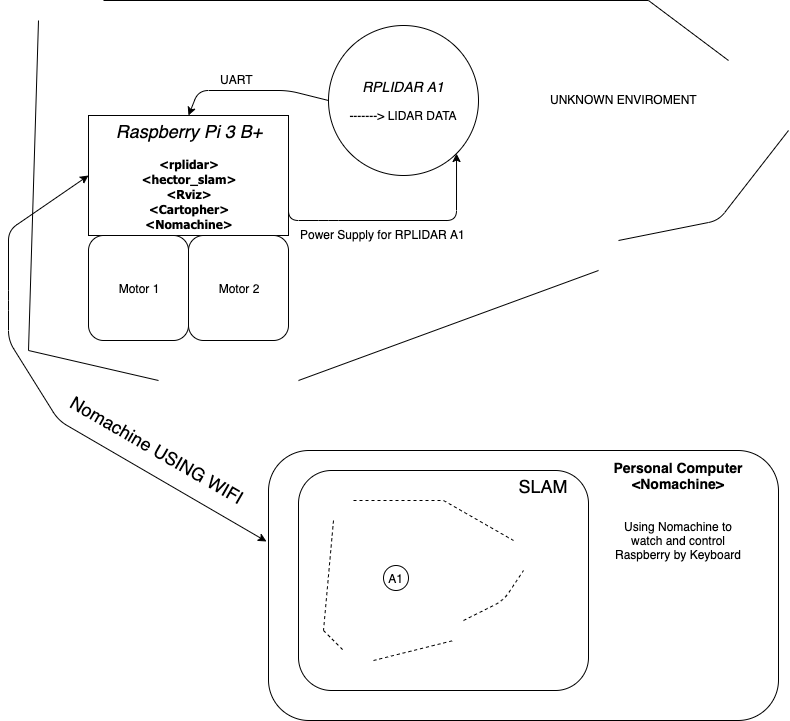
File launch sẽ chạy các node hector\_mapping, hector\_trajectory\_server và hector\_geotiff để xây dựng bản đồ GeoTiff hiển thị lên phần mềm rviz.

3.2.Tổng quan về hệ thống xây dựng bản đồ và định vị dựa trên LIDAR và gói Hector\_SLAM :

Hình 19 - Hệ thống xây dựng bản đồ và định vị dựa trên LIDAR và gói Hector\_SLAM

Hệ thống được kết hợp hệ thống 2D SLAM dựa trên sự tích hợp giữa quét laser (LIDAR) trong bản đồ 2D và hệ thống định vị 3D tích hợp, bao gồm đơn vị đo lường quán tính (IMU), GPS, la bàn,… Nhờ sự kết hợp giữa 2 phần nêu trên, hệ thống trả về một nguồn thông tin để xử lý map 2D do thiết bị laser quét được (Hình ). Trong khi SLAM thường được chạy trong hệ thống “Soft Real-time” , bắt sự kiện ở các lần cập nhật của thiết bị laser, thì giải pháp của hệ thống định vị 3D là tính toán thông tin trả về của các phần bao

gồm đã nêu trên trong hệ thống “Hard Real-time” và thường được cấu thành một phần của hệ thống điều khiển phương tiện.

3.3. Tổng quan về sơ đồ hoạt động của chương trình:

Hình 20 - Sơ đồ hoạt động của chương trình

Chương 4. KẾT QUẢ THỰC NGHIỆM

## 4.1. Kết quả phần cứng

Hình 21 - Kết quả phần cứng

RPLidar quét laser và gửi dữ liệu về board RASPBERRY thông qua cổng UART. Toàn bộ phương tiện được cấp nguồn bởi 2 nguồn pin Lipo, 1 nguồn cấp cho RASPBERRY và RPLidar, và 1 cấp cho động cơ. Phương tiện được điều khiển thông qua bàn phím và app NoMachine để hiển thị về laptop. Ngoài ra có 2 thiết bị hiển thị áp pin để theo dõi dung lượng. Động cơ được điều khiển thông qua module cầu H.

## 4.2.Kết quả đầu ra

### 4.2.1. Phân tích kết quả đầu ra:

Hình ảnh thật:

Hình 22 - Hình thực tế của hộ dân

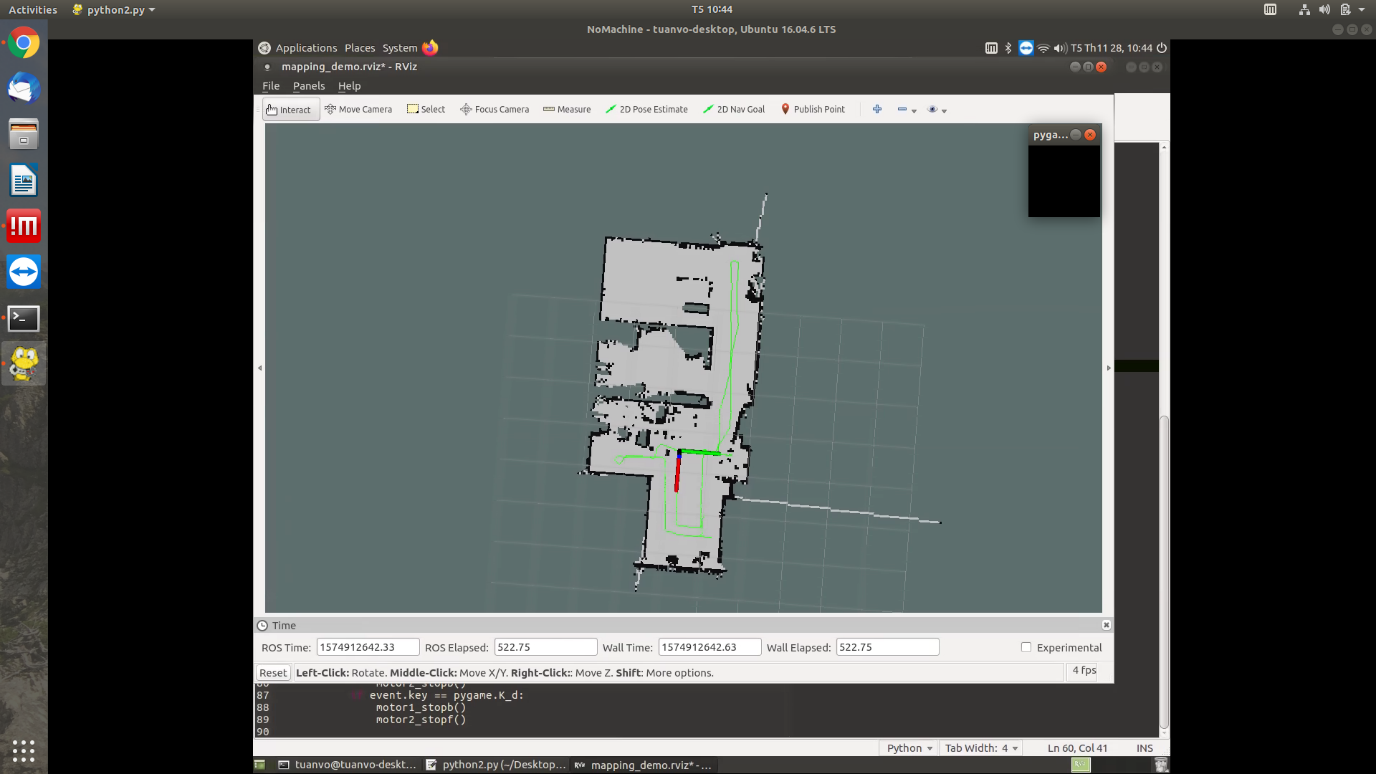


Hình 23

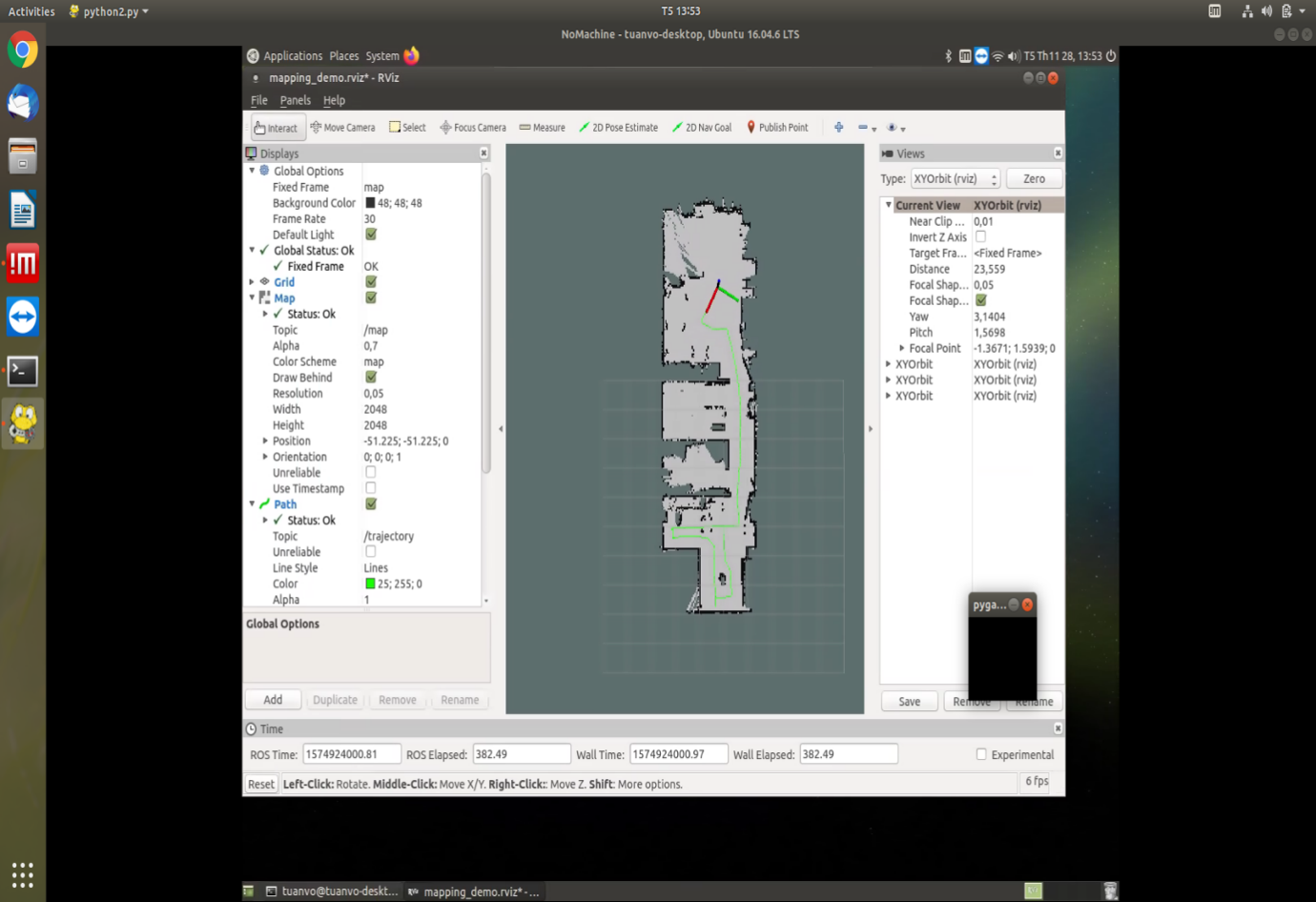


Hình 24

Hình 25

Hình ảnh kết quả:

Hình 26 - Kết quả trả về của sơ đồ hộ dân

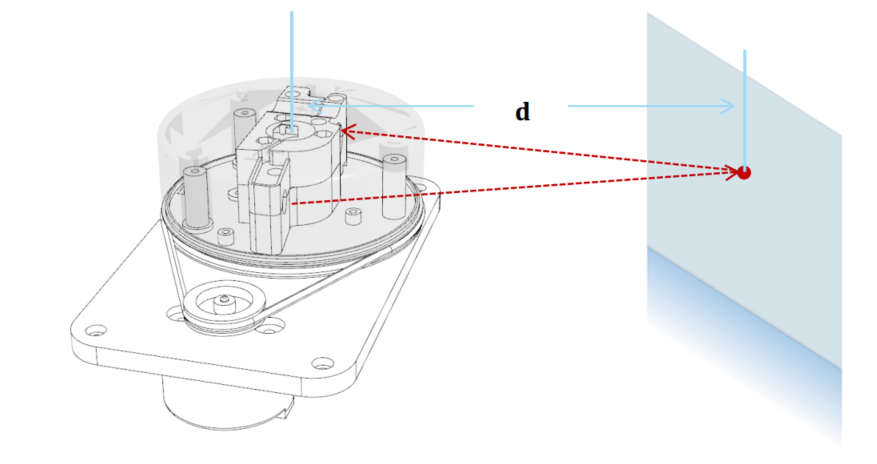


Hình 27

Kết quả đầu ra ở trên là sơ đồ 1 tầng lầu của một hộ dân. Đường màu đen thể hiện các mặt tiếp xúc cố định như tường,… Đường màu xanh lá thể hiện lịch sử hành trình của phương tiện thông qua kết quả trả về của hệ

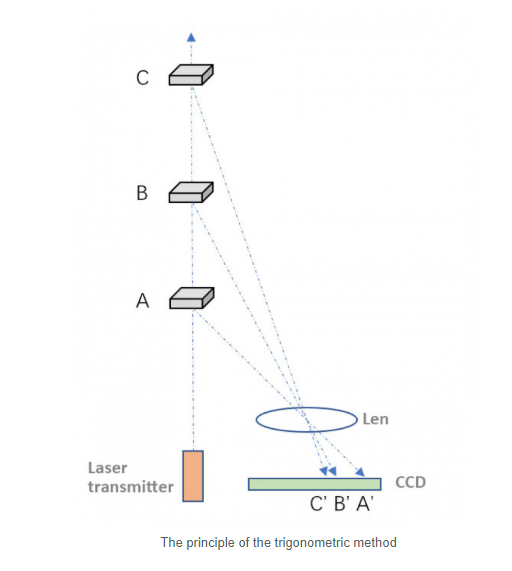
thống định vị 3D. Phương tiện có 3 trục được thể hiện ở 3 màu (màu đỏ trục y, màu xanh lá trục x và màu xanh dương là trục z).

Khi di chuyển , phương tiện đồng thời phát các tia laser và đo lường khoảng cách với thời gian lấy mẫu lớn nhất là 8000 sa/s, với góc chia nhỏ nhất là 1 độ và khoảng cách ngắn nhất là 0.2 cm, lớn nhất là 12 m. Sau mỗi vòng quay, thiết bị sẽ lấy mẫu khoảng 1450 điểm.



Hình 28 - Khoảng cách giữa RPLIDAR và vật thể

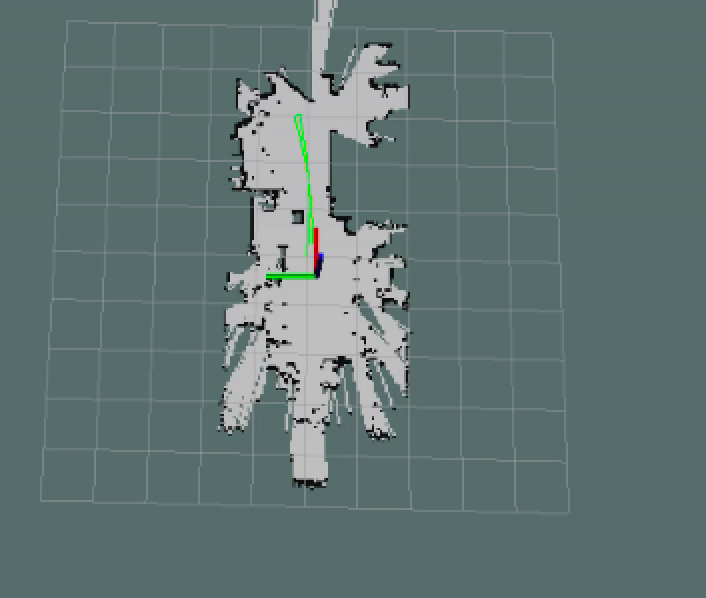
Bằng cách sử dụng phương pháp lượng giác, khi tia laser được phát ra và chạm đến vật cản và quay trở về cảm biến tuyến tính CCD (cảm biến chuyển đổi hình ảnh quang học sang tín hiệu điện) được tích hợp trong RPLIDAR. Trong khi bộ phát tia laser và bộ thu được thiết kế cách nhau môt khoảng cách nhất định (RPLIDAR khoảng 4 cm), hình ảnh các vật cản ở các vị trí khác nhau sẽ được thể hiện qua một thấu kính hội tụ và hiển thị ở cảm biến CCD . Khoảng cách các vật cản được tính bằng công thức lượng giác với một độ dài và một góc cho trước ở phần cứng của RPLIDAR.



Hình 29 - Phương pháp lượng giác

### 4.2.2. Các lỗi phát sinh:

Theo nhà phát triển của gói Hector SLAM, yêu cầu thiết bị quét, cụ thể là RPLIDAR phải được nằm trên một mặt phẳng, không gồ ghề vì có thể làm thay đổi vị trị định vị ban đầu của thiết bị, từ đó tạo lại một map khác đè chồng lên map cũ, làm sai lệch dữ liệu.



Hình 30 – Hình thực tế của hộ dân

Hình 31 – Lỗi phát sinh

Chương 5. KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

## 5.1. Kết luận

Với phương tiện đã làm , nhóm đã tạo ra một map 2D môi trường cơ bản, tạo tiền đề để phát triển robot tự hành về sau.

Robot có thể di chuyển bằng điều khiển thông qua bàn phím qua app NoMachine.

Dữ liệu số trả về từ RPLIDAR được dùng để đọc tính khoảng cách để sử dụng làm dữ liệu về sau.

## 5.2. Phương hướng phát triển

Kết hợp với xử lý ảnh và RPLIDAR để tạo ra robot tự hành hoàn thiện, có các chức năng đeo bám vật thể, vẽ bản đồ và tránh vật cản.

Nâng cấp board Main để xử lý tốt hơn, nhóm đang hướng tới board Jetson Nano, với CPU có 4 nhân ARM A57, GPU có 128 nhân Maxwell,… có thể được dùng để xử lý các tác vụ xử lý ảnh.

Để robot tự hành yêu cầu phải có PID, nên ở đồ án môn học 2 sẽ nâng cấp lên động cơ encoder.

Fix các bug như thay đổi độ cao của robot gây ra lỗi.

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE

* RPLIDAR A1-Low Cost 360 Degree Laser Range Scanner-Model: A1M8 [www.slamtec.com](http://www.slamtec.com)
* RPLIDAR -Low Cost 360 Degree Laser Range Scanner-Introduction to Standard SDK - [www.slamtec.com](http://www.slamtec.com)
* A Flexible and Scalable SLAM System with Full 3D Motion Estimation

Stefan Kohlbrecher, Oskar von Stryk, Johannes Meyer và Uwe Klingauf

* Hector SLAM - <http://wiki.ros.org/hector_slam>
* RPLIDAR - <http://wiki.ros.org/rplidar>
* TurtleBot - <https://www.turtlebot.com/>