**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ**

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

****

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**THIẾT KẾ MÔ HÌNH MÔ PHỎNG**

**XE LĂN TỰ HÀNH TRONG NHÀ**

**GVHD: TS. NGUYỄN LÝ THIÊN TRƯỜNG**

**SVTH: NGUYỄN DUY LINH**

**MSSV: 1711957**

**TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2021**

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

TRƯỜNG ĐẠI HỌC BÁCH KHOA Độc lập – Tự do – Hạnh phúc.

-----✩----- -----✩-----

Số: \_\_\_\_\_\_ /BKĐT

Khoa: **Điện – Điện tử**

Bộ Môn: **Điện Tử**

N**HIỆM VỤ LUẬN VĂN TỐT NGHIỆP**

1. HỌ VÀ TÊN : NGUYỄN DUY LINH MSSV: 1711957

1. NGÀNH: **ĐIỆN TỬ - VIỄN THÔNG** LỚP : DD17DV4
2. Đề tài:
3. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

1. Ngày giao nhiệm vụ luận văn: ...............................
2. Ngày hoàn thành nhiệm vụ: ...................................
3. Họ và tên người hướng dẫn: Phần hướng dẫn

................................................................. .....................................

................................................................. .....................................

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

*Tp.HCM, ngày…... tháng….. năm 20*

**CHỦ NHIỆM BỘ MÔN NGƯỜI HƯỚNG DẪN CHÍNH**

**PHẦN DÀNH CHO KHOA, BỘ MÔN:**

Người duyệt (chấm sơ bộ):.......................

Đơn vị:......................................................

Ngày bảo vệ : ...........................................

Điểm tổng kết: .........................................

Nơi lưu trữ luận văn: ...............................

***LỜI CẢM ƠN***

*Tp. Hồ Chí Minh, ngày tháng năm .*

**Sinh viên**

**TÓM TẮT LUẬN VĂN**

Luận văn này trình bày về …

* Thiết bị tự hành
* Visual SLAM
* Xe lăn tự hành ứng dụng Visual SLAM

**MỤC LỤC**

[1. GIỚI THIỆU 1](#_Toc86743761)

[1.1 Tổng quan 2](#_Toc86743762)

[1.2 Nhiệm vụ luận văn 2](#_Toc86743763)

[2. LÝ THUYẾT 2](#_Toc86743764)

[1.1 Nội dung 1 2](#_Toc86743765)

[a. Lý thuyết cơ bản lĩnh vực Probabilistic Robotics: 2](#_Toc86743766)

[b. Lý thuyết các khái niệm trong Visual SLAM 2](#_Toc86743767)

[1.1.1 Nội dung 3 2](#_Toc86743768)

[1.1.1.1 ROS Filesystem Level 2](#_Toc86743769)

[1.1.1.2 ROS Computation Graph Level 2](#_Toc86743770)

[1.1.1.3 ROS Community Level 2](#_Toc86743771)

[1.1.1.4 ROS TF Coordinate Transform 2](#_Toc86743772)

[1.1.2 Nội dung 5 2](#_Toc86743773)

[1.1.2.1 Tổng quan hệ thống 2](#_Toc86743774)

[1.1.2.2 Lý thuyết phần cứng 2](#_Toc86743775)

[a. Differential drive – Lái vi sai 2](#_Toc86743776)

[b. Pulse Width Modulation – PWM. 2](#_Toc86743777)

[c. Mạch Four-quadrant Chopper 2](#_Toc86743778)

[d. Bộ điều khiển PI (PI controller) 2](#_Toc86743779)

[3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG 2](#_Toc86743780)

[3.1 Tổng quan 2](#_Toc86743781)

[3.2 Khung 2](#_Toc86743782)

[3.3 Máy tính nhúng 2](#_Toc86743783)

[3.4 RGB-D camera. 2](#_Toc86743784)

[3.5 Mạch điều khiển động cơ. 2](#_Toc86743785)

[3.6 MCU 2](#_Toc86743786)

[3.7 Motors 2](#_Toc86743787)

[3.8 Nguồn 2](#_Toc86743788)

[4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM 2](#_Toc86743789)

[4.1 Phần mềm trên ROS 2](#_Toc86743790)

[**4.1.1** **TX Frame Parser trên Pi 4** 2](#_Toc86743791)

[**4.1.2** **TX frame parser trên STM32** 2](#_Toc86743792)

[4.2 Phần mềm trên STM32 2](#_Toc86743793)

[5. KẾT QUẢ THỰC HIỆN 2](#_Toc86743794)

[6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 2](#_Toc86743795)

[6.1 Kết luận 2](#_Toc86743796)

[6.2 Hướng phát triển 2](#_Toc86743797)

[7. TÀI LIỆU THAM KHẢO 2](#_Toc86743798)

[8. PHỤ LỤC 2](#_Toc86743799)

DANH SÁCH HÌNH MINH HỌA

[*Hình 1‑1 Robot kho vận thông minh là ứng dụng tiêu biểu cùa SLAM.* 1](#_Toc86742624)

[Hình 1‑2 Phân loại các giải thuật Path Planning [4]. 4](#_Toc86742625)

[Hình 4‑1 Tập hợp partiles thể hiện một phân bố xác suất ngẫu nhiên. 12](#_Toc86742626)

[Hình 4‑2 Minh hoạ phân bố xác suất của vị trí robot qua các bước. Biến k đại diện cho lần lặp thứ k, thay cho t. [6] 13](#_Toc86742627)

[Hình 4‑3 Một số features giống nhau xuất hiện trong hai ảnh khác nhau. 14](#_Toc86742628)

[Hình 4‑4 Minh hoạ khung toạ độ camera. [7] 15](#_Toc86742629)

[*Hình 4‑5 Minh hoạ vị trí 2 camera và điểm P. [7]* 16](#_Toc86742630)

[Hình 4‑6 Minh hoạ cho quá trình reprojection.[8] 18](#_Toc86742631)

[Hình 4‑7 a) Sai số tích luỹ dần trên bản đồ lập bởi SLAM; b) Sau khi phát hiện loop closure và thực hiện bundle adjustment. 19](#_Toc86742632)

[Hình 4‑8 Minh hoạ quá trình trích visual words cho mỗi ảnh. 20](#_Toc86742633)

[Hình 4‑9 Cấp quản lý Filesystem của ROS. 22](#_Toc86742634)

[Hình 4‑10 Các thành phần của cấp Computation Graph 23](#_Toc86742635)

[Hình 4‑11 Tên message type mang thông tin về package định nghĩa nó (Package name)và tên file chứa định nghĩa (Type Name) 24](#_Toc86742636)

[Hình 4‑12 Minh hoạ cơ chế truyền thông qua topic giữa hai node. Master quản lý việc quảng bá topic mới và quản lý đăng ký topic của các node. 24](#_Toc86742637)

[Hình 4‑13 Minh hoạ các khung toạ độ - coordinate frames của robot. 26](#_Toc86742638)

[Hình 4‑14 Minh hoạ vị trí tương đối giữa 3 khung toạ độ. robot\_base là khung toạ độ của thân robot. mouting\_point là khung toạ độ của ngàm giữ camera. camera\_center là khung toạ độ camera tương tự Hình 4-4. 27](#_Toc86742639)

[Hình 4‑15 TF tree cho thấy mối quan hệ giữa 5 khung toạ độ: /world, /turtle1, /turtle2, /carrot1, /carrot2. Trong đó /world là khung toạ độ tham chiếu toàn cục. Broadcaster là tên node publish thông tin của mỗi khung toạ độ. 29](#_Toc86742640)

[Hình 4‑16 Sơ đồ tổng quan của hệ thống. 30](#_Toc86742641)

[Hình 4‑17 ICC nằm trên đường thằng qua trục động cơ. 32](#_Toc86742642)

[Hình 4‑18 Minh hoạ cho duty circle và độ rộng xung tương ứng. 34](#_Toc86742643)

[Hình 4‑19 Dòng điện qua cuộn dây được giữ liên tục khi tần số PWM đủ lớn. 34](#_Toc86742644)

[Hình 4‑20 Mạch four-quadrant chopper điều khiển động cơ. 35](#_Toc86742645)

[Hình 4‑21 Bốn góc phần tư hoạt động của các mạch chopper. 35](#_Toc86742646)

[Hình 4‑22 Bộ điều khiển PI vòng kín có output feedback cho tín hiệu liên tục. 37](#_Toc86742647)

[Hình 4‑23 So sánh đáp ứng của bộ điều khiển khi có wind-up và khi đã áp dụng anti-windup. PV, SP lần lượt là giá trị feedback (measure process variable) và giá trị đặt (set point). CO là output của bộ điều khiển (control output). 39](#_Toc86742648)

[Hình 4‑24 Sơ đồ phần cứng chi tiết. 40](#_Toc86742649)

[Hình 4‑25 Kích thước mô hình. Dài 26.4 cm. Rộng 23.1 cm. Cao 27 cm khi đã bao gồm Kinect. 41](#_Toc86742650)

[Hình 4‑26 Thông số kỹ thuật của Raspberry Pi 4 Model B (4GB RAM). 42](#_Toc86742651)

[Hình 4‑27 Board Raspberry Pi 4 Model B. 42](#_Toc86742652)

[Hình 4‑28 Kinect V1. 43](#_Toc86742653)

[Hình 4‑29 Hình ảnh và kích thước thực của động cơ. 46](#_Toc86742654)

[Hình 4‑30 Sơ đồ phân phối nguồn đến các khối phần cứng. 48](#_Toc86742655)

[Hình 4‑31 Hai khối DMA truy cập trực tiếp tới bus bộ nhớ (SRAM, FLASH và FSMC) và bus ngoại vi (APB & AHB). 50](#_Toc86742656)

[Hình 4‑32 Lưu đồ hàm main() và hàm xử lý ngắt UART. 51](#_Toc86742657)

[Hình 4‑33 Minh hoạ chế độ PWM của STM32 timer. 52](#_Toc86742658)

[Hình 4‑34 Timer ở Encoder Mode đọc xung encoder. 53](#_Toc86742659)

[Hình 4‑35 Lưu đồ PI controller 53](#_Toc86742660)

**DANH SÁCH BẢNG SỐ LIỆU**

[Bảng 1 Thông số hệ thống 3](#_Toc310380293)

# GIỚI THIỆU

## Tổng quan

Trang thiết bị tự động được nghiên cứu và phát triển từng ngày để hỗ trợ con người trong sinh hoạt, sản xuất. Chúng ngày càng thông minh để thoả mãn được nhu cầu mạnh mẽ mà chúng ta đặt ra. Bây giờ, các thiết bị tự động đã có thể tự thực hiện những phần công việc độc hại, khó khăn thay con người một cách chính xác và năng suất hơn bao giờ hết. Hơn thế nữa, chúng có khả năng thực hiện cả những công việc mà con người không thể. Công nghệ không ngừng phát triển, mở rộng khả năng của máy móc đồng thời cũng kéo theo viễn cảnh lao động con người bị thay thế hoàn toàn.

Để đạt được tới viễn cảnh đó, trước hết, máy móc cần phải tự mình di chuyển mà không cần sự điều khiển từ con người, hay nói các khác là nó phải có khả năng tự hành. Đây là khả năng cơ bản nhất mà máy móc cần có để thực sự thế chân con người. Nhưng thực tế chưa có nghiên cứu nào đặt ra khả năng tự hành hoàn toàn ổn định trong mọi điều kiện vận hành cho máy móc. Dẫu vậy lĩnh vực nghiên cứu phương tiện tự hành vẫn đạt được nhiều thành tựu lớn. Những nỗ lực tiêu biểu trong nghiên cứu và ứng dụng khả năng tự hành là hệ thống trên xe tự lái của Tesla và robot Perseverance của NASA. Ngoài ra, rất nhiều robot nhỏ với khả năng tự hành trong môi trường đặc thù đang được ứng dụng hiệu quả như nhiều robot hút bụi trong gia đình, đã được thương mại hoá, hoặc robot kho bãi và giao hàng của công ty Alibaba.



*Hình 1‑1 Robot kho vận thông minh là ứng dụng tiêu biểu cùa SLAM.*

Bên cạnh việc nghiên cứu phát triển các thiết bị tự hành thay thế lao động con người, cũng có những nghiên cứu thiết bị như xe lăn tự hành hỗ trợ người khuyết tật về thị lực, hỗ trợ họ di chuyển dễ dàng hơn trong môi trường phức tạp hoặc không quen thuộc. Đề tài cũng chọn hướng ứng dụng như vậy [12][13]. Mục tiêu là thiết kế một thiết bị tương tự xe lăn có khả năng tự hành với điều kiện môi trường trong nhà. Vậy nên, đề cương luận văn này sẽ nghiên cứu, đánh giá và đưa vào ứng dụng một số giải thuật giải quyết bài toán lớn gồm 3 vấn đề sau đây:

1. Làm sao để thiết bị có khả năng tự lập bản đồ không gian xung quanh (Mapping).
2. Làm sao để thiết bị định vị được chính nó trong không gian đó (Localization).
3. Làm sao để thiết bị có thể tự lên kế hoạch di chuyển đến điểm cần đến. (Path Planning).

Bài toán lớn này là bài toán chung của lĩnh vực nghiên cứu thiết bị/robot tự hành. Trong phần tiếp theo, tình hình nghiên cứu tìm ra giải pháp cho chúng sẽ được trình bày.

Tình hình nghiên cứu trong và ngoài nước

Các nghiên cứu phương tiện tự hành đều làm cả hai việc: lập bản đồ và định vị thiết bị, cùng một lúc. Cái tên chung cho các giải thuật thực hiện hai việc như vậy là **S**imultaneous Localization And Mapping, hay gọi tắt là SLAM. Nhìn chung, các giải thuật giải quyết bài toán chia làm 2 hướng tiếp cận chính:

1. Dùng cảm biến khoảng cách thu thập dữ liệu về khoảng cách gữa thiết bị tới vật cản trong không gian.
2. Dùng camera như cảm biến chính. Hình ảnh từ một hay nhiều camera được dùng để trích xuất thông tin cần thiết.

Mỗi cách tiếp cận đều có ưu điểm và nhược điểm riêng.

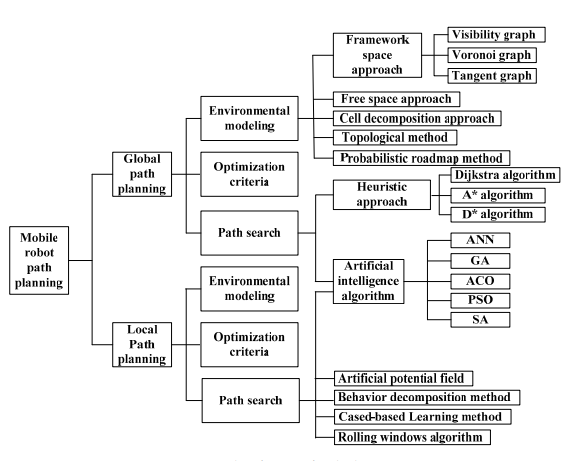
Nhóm giải thuật thứ nhất là những giải thuật SLAM cơ bản nhất và đã được nghiên cứu tương đối hoàn thiện. Dữ liệu đầu vào của các giải thuật loại này là khoảng cách tới các điểm không gian, một mặt phẳng 2D, vốn là dữ liệu trực tiếp và cần thiết nhất cho tác vụ lập bản đồ và định vị. Vì lý do chúng đều dùng dữ liệu ở dạng 2D nên được gọi chung là các giải thuật 2D-SLAM. Những giải thuật 2D-SLAM, vì vậy, không phải là tác vụ quá nặng so với các vi xử lý hiện có. 2D-SLAM lại có nhiều hướng nghiên cứu như Grid-based, Graph-based hoặc Feature-based. Grid-based SLAM vốn đã được ứng dụng thực tế trong gói phần mềm Gmapping (bởi nhóm nghiên cứu của Giorgio Grisetti [1]) được tích hợp trong hệ điều hành cho robot - ROS. 2D-SLAM dùng các loại cảm biến khoảng cách như RADAR, LiDAR hoặc ToF. Tuy nhiên những loại cảm biến kể trên thường phức tạp, chính xác cao nên giá thành là một rào cản để có thể đưa vào ứng dụng rộng rãi.

Các giải thuật ở nhóm thứ hai thì ngược lại. Camera thường là cảm biến chính được dùng và giá thành camera thì rất hợp lý, nhờ độ phổ biến cao. Giải thuật dùng dữ liệu ảnh trực quan nên được gọi là Visual SLAM hay V-SLAM. Các giải thuật Visual SLAM dùng thông tin trích xuất từ ảnh. Thông tin thường là khoảng cách tới các vật thể mà ảnh thu được và vị trí của chúng so với camera. Khác biệt giữa Visual SLAM so với 2D-SLAM là vị trí của các điểm so với camera là vị trí trong không gian 3D, thay vì một mặt cắt 2D so với 2D-SLAM. Vì thế, camera thu được nhiều thông tin về không gian xung quanh hơn. Số lượng điểm cần để Visual SLAM tính toán do vậy cũng trở nên rất nhiều để có được độ chính xác cần thiết. Trong khi đó, các phương pháp trích xuất khoảng cách từ ảnh cũng đòi hỏi phải trải qua nhiều bước dẫn đến khối lượng xử lý rất lớn khi triển khai Visual SLAM. Nhưng, vi xử lý ngày càng mạnh hơn theo định luật Moore và giải thuật thì luôn có thể cải tiến, Visual SLAM vẫn đang là lĩnh vực nghiên cứu năng động. Hai thành quả nghiên cứu tiêu biểu có thể kể tới là giải thuật ORB-SLAM 3 (do nhóm nghiên cứu của Carlos Campos [2]) và RTAB-MAP (do nhóm nghiên cứu của Mathieu Labbé [3]). Riêng RTABMAP là giải thuật đã được ứng dụng và được nhóm tác giả đóng thành gói phần mềm của hệ điều hành cho robot – ROS.

So sánh giữa 2D-SLAM và Visual SLAM, 2D-SLAM đang là hướng tiếp cận phổ biến và đã được ứng dụng, thương mại hoá. Lĩnh vực nghiên cứu 2D-SLAM có thể nói là đã đạt được những thành tựu tương đối. Trong khi đó, Visual SLAM phần nhiều vẫn còn là những giải thuật nghiên cứu mang tính lý thuyết, chưa được sử dụng rộng như 2D-SLAM. Nhưng Visual SLAM lại rất hứa hẹn khi dữ liệu ảnh mang được rất nhiều thông tin trực quan, có thể đem lại cho thiết bị khả năng ứng dụng rộng hơn nhờ những nghiên cứu mới của lĩnh vực Computer Vision. Và cũng do vậy mà lĩnh vực nghiên cứu Visual SLAM vẫn còn nhiều cơ hội để phát triển các thuật toán tốt hơn nữa. Lĩnh vực này cũng tạo ra thách thức để những kỹ sư có thể ứng dụng thành quả nghiên cứu vào những ứng dụng thực tế thay thế 2D-SLAM.

Bài toán SLAM giải quyết được hai vấn đề đầu tiên là lập bản đồ (mapping) và định vị (localization). Thiết bị phải có khả năng giải quyết thêm vấn đề thứ ba là lên kế hoạch di chuyển (navigation hay path planning) thì mới đạt được khả năng tự hành đúng nghĩa. Lĩnh vực nghiên cứu bài toán tìm đường đi tối ưu nhất đã có nhiều giải thuật tương đối hoàn thiện. Thành quả nghiên cứu từ lâu đã được ứng dụng rộng rãi và vẫn còn được phát triển xa hơn. Các thuật toán đã có thể lên kế hoạch di chuyển trong nhiều môi trường, từ đơn giản đến phức tạp.

Có hai hướng nghiên cứu cho bài toán tìm đường đi tối ưu dựa trên hai mục đích: *global navigation* và *local navigation*. Khả năng dựa vào vị trí hiện tại và vị trí đích đến trong bản đồ có sẵn để lên kế hoạch di chuyển tới đích gọi là global navigation. Trong khi đó, local navigation giúp thiết bị có khả năng ứng biến khi có thay đổi xảy ra trong quá trình di chuyển tới điểm đến. Những giải thuật cho global navigation tiêu biểu có thể kể đến như giải thuật Dijkstra cùng cải tiến như giải thuật A\*. Giải thuật RRT (Rapidly-Exploring Random Trees) và giải thuật cải tiến RRT\* cũng được ứng dụng cho mục đích global navigation nhờ ưu điểm nhanh và khối lượng tính toán nhẹ. Mỗi giải thuật sẽ đặc biệt hiệu quả trong mỗi môi trường khác nhau. Đối với local navigation, nhiều giải thuật như Dynamic Window Approach (DWA), D\*, .. tỏ ra tương đối hiệu quả và đã được tích hợp cho ứng dụng thực tế.



Hình 1‑2 Phân loại các giải thuật Path Planning [4].

## Nhiệm vụ luận văn

Nhận thấy cơ hội và thách thức trong việc ứng dụng Visual SLAM, đề tài quyết định chọn phương án này làm giải thuật chính cho mô hình.

Về mục tiêu tổng quan, đề tài sẽ dựa trên lý thuyết đưa ra phương án thiết kế và thi công một thiết bị (robot) mô phỏng xe lăn, có khả năng tự hành trong nhà. Thiết bị phải đạt được khả năng di chuyển từ điểm đầu tới điểm đích định sẵn và có khả năng ứng phó với các vật cản tĩnh và động.

Mục tiêu tổng quan của đề tài được phân tích và cụ thể hoá thành các nội dung như dưới đây như là kế hoạch cho từng giai đoạn phát triển đề tài.

* **Nội dung 1:** Tìm hiểu lý thuyết toán xác suất giải quyết bài toán SLAM và lý thuyết về xử lý ảnh, thị giác máy tính được dùng trong các giải thuật Visual SLAM. Lựa chọn giải thuật Visual SLAM phù hợp để triển khai.
* **Nội dung 2:** Tìm hiểu về các thuật toán tìm đường đi tối ưu cho global navigaton và local navigation. Lựa chọn thuật toán để thực hiện.
* **Nội dung 3:** Chọn ROS làm nền tảng cho SLAM và navigation. Tìm hiểu về cấu trúc của ROS và lý thuyết về các khối chức năng/cơ chế thông tin của ROS..
* **Nội dung 4:** Phác hoạ thiết kế các khối phần mềm để chạy trên ROS.
* **Nội dung 5:** Thiết kế và thực hiện mô hình mô phỏng xe lăn.
* **Nội dung 6:** Triển khai hệ thống phần mềm lên phần cứng thực tế, sửa lỗi và đánh giá kết quả đạt được so với mục tiêu ban đầu đã đề ra.

# LÝ THUYẾT

Phần 2 – Lý thuyết trình bày các lý thuyết liên quan của đề tài được đề cập tới trong các nội dung 1, 2, 4 và 5.

## Nội dung 1: Probabilistic Robotic, Visual SLAM và RTAB-MAP

## Lý thuyết cơ bản lĩnh vực Probabilistic Robotics:

Lĩnh vực *Probabilistic Robotics* là lĩnh vực nghiên cứu phát triển nhận thức, hành động của robot dựa trên lý thuyết xác suất. Nhận thức của robot về môi trường xung quanh thường dựa trên thông tin từ cảm biến, vốn bị ảnh hưởng bởi nhiễu. Vì vậy, lý thuyết xác suất được ứng dụng và biểu diễn thông tin thành một dạng phân phối xác suất để thể hiện mức độ không chắc chắn của thông tin do nhiễu. Dựa vào đó robot có thể tính toán và đưa ra quyết định linh hoạt hơn nhờ dự đoán được các khả năng có thể xảy ra do sực không chắc chắn đó. Hơn thế nữa, khi robot hành động nhờ vào thông tin dạng xác suất sẽ giảm được sự ảnh hưởng từ nhiều nguồn nhiễu [5, p.3].

Quy ước trình bày trong báo cáo về ký hiệu toán như sau:

Với một biến ngẫu nhiên xác định trong miền D, có không gian mẫu là Ω ↦D. là giá trị cụ thể nằm trong không gian mẫu Ω. Thì hàm phân bố xác suất của xác suất để biến cố xảy ra sẽ là sao cho:

|  |  |  |
| --- | --- | --- |
|  |  | (0) |

mang đủ tính chất của xác suất:

**Khái niệm cơ bản** ***Probabilistic Robotics***

1. State: Tất cả yếu tố liên quan đến môi trường xung quanh robot và bản thân robot là *state*. Chính vì vậy state bị ảnh hưởng bởi rất nhiều biến ngẫu nhiên trong và ngoài hệ thống. State cũng được xem là môt biến ngẫu nhiên, ký hiệu . Các biến cố của được kí hiệu là và biến cố xảy ra ở thời điểm sẽ là . Do robot thường là các hệ thống số nên rời rạc kéo theo state cũng không liên tục và các biến cố ảnh hưởng đến state cũng xảy ra không liên tục.

Một state gọi là hoàn chỉnh khi là dự đoán tốt nhất và các biến ngẫu nhiên ảnh hưởng đến trong quá khứ không giúp dự đoán tốt hơn trong tương lai. Khái niệm này được đưa ra để đơn giản hoá các xác suất điều kiện được sử dụng trong *Probabilistic Robotics*.

Trong báo cáo, thuật ngữ *trạng thái* được sử dụng thay cho thuật ngữ *state* trong một số trường hợp để dễ hiểu hơn.

1. Measurement: Measurement còn được đề cập là Observation hoặc Percept trong các bài báo để chỉ thông tin về trạng thái môi trường xung quanh mà robot thu thập được từ sensor. Measurement cũng là biến ngẫu nhiên, được ký hiệu là . Các biến cố của sẽ là . ở dạng rời rạc theo thời gian và thường được kí hiệu là .
2. Control: Control là thông tin về sự thay đổi của môi trường xung quanh do robot chủ động tác động. Ví dụ như robot di chuyển với tốc độ 10 cm/s trong khoảng thời gian 5 giây thì trạng thái mới của robot sẽ cách trạng thái cũ 50 cm. Vậy nên, về bản chất, control mang thông tin về sự thay đổi của state. Ký hiệu thông dụng dùng cho biến control là U và biến cố theo thời gian là là .
3. Belief: Belief là thông tin về state mà robot suy ra dựa trên dữ liệu mà nó đang có. Phân phối xác suất của belief thể hiện những khả năng có thể xảy ra với trạng thái thực. Belief tồn tại dưới dạng xác suất hậu nghiệm dựa trên tiền nghiệm là measurement và control:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

là kí hiệu của belief của . Về mặt ý nghĩa, chính là ước lượng của state dựa trên measurement và control trong quá khứ, kèm theo là control tại thời điểm : . Ước lượng của được thực hiện trước khi thu được measurement tại thời điểm (). Một khi đã biết , sẽ được chính xác hoá lại thành . chính là ước lượng hoàn chỉnh của robot về state tại thời điểm .

**Bayes Filter**

Ý tưởng về belief trong *Probabilistic Robotics* dựa trên nguyên lý của Bayes Filter.

Bayes filter dựa trên định lý Bayes cho xác suất có điều kiện, rằng:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

Nếu xem là state và là measurement thì công thức trên cho thấy ta có thể tính được xác suất của state khi đã có measurement (dựa trên xác suất thu được measurement tại state ). Dữ liệu đi kèm theo là xác suất tiền nghiệm , đại diện cho mọi ước lượng về state trước khi measurement được thực hiện. chỉ đóng vai trò chuẩn hoá cho phép nhân ở tử. Vậy nên có thể nói, quá trình tính xác suất hậu nghiệm thực tế là quá trình chính xác hoá belief trước đó là dựa trên xác suất thu được measurement tại state .

Thực tế, state bị ảnh hưởng bởi nhiều hơn một biến ngẫu nhiên. Thật vậy, trong lý thuyết *Probabilistic Robotics* tính tới hai yếu tố *measurement* và *control* nên định lý Bayes cho 3 biến ngẫu nhiên sẽ được quan tâm. Cụ thể, thêm biến ngẫu nhiên Z có biến cố là thì:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Công thức trên suy ra từ định lý Bayes cho 2 biến (2) và được ứng dụng cho như sau:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  | |  | (4) |

Trong đó: là , đóng vai trò là hệ số chuẩn hoá.

Nếu giả thuyết state tại thời điểm t là hoàn chỉnh, đồng nghĩa là hoàn chỉnh, thì xác suất dựa trên sẽ không phụ thuộc vào các biến cố trong quá khứ của các biến ngẫu nhiên khác. Vậy nên:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | = | | (5) | |
|  | |  | |  | |

Phương trình (5) cho thấy xác suất có được measurement chỉ phụ thuộc vào state , bỏ qua mọi thông tin về measurement và control trong quá khứ. Điều này giúp rút gọn phương trình (4) thành:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Có thể thấy từ phương trình (6), chính là xác suất hậu nghiệm của khi đã thu được measurement . là xác suất tiền nghiệm khi chưa thu được measurement . Nói cách khác, có vai trò là và là . Từ đó, phương trình (6) có thể được viết như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Trong đó:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Để tìm ra mối quan hệ giữa và state trước đó , ta cần phải đưa vào phương trình (8). Áp dụng tổng xác suất của trong không gian mẫu bằng 1 ().

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  |  |  | |
|  |  | (9) |

Tiếp tục áp dụng giả thuyết state là hoàn chỉnh, khi đã biết mọi thông tin khác không giúp dự đoán xác suất tốt hơn, nên ta có:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

Phương trình (10) cho thấy dự đoán chỉ phụ thuộc vào state trước đó và tín hiệu control . Từ (8) (9) và (10) có thể kết luận theo công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

Trong đó: là vì không thể tác động đến state trong quá khứ nên được bỏ qua. Cuối cùng trong Bayes filter được tính bằng công thức sau:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Tổng quan lại, từ phương trình (7) và (12), giải thuật của Bayes filter ước lượng state trên dữ liệu measurement và control theo phương pháp hồi quy được thể hiện qua pseudo code như sau:

|  |  |
| --- | --- |
| **Algorithm Bayes\_filter**(): | |
| 1 | for all do |
| 2 |  |
| 3 |  |
| 4 | endfor |
| 5 | return |

Bảng 2‑1 Pseudo code của giải thuật Bayes filter.

Dựa theo pseudo code có thể hiểu rằng, Bayes filter dựa trên 2 bước chính:

* + - Dự đoán: là quá trình ước lượng trạng thái , hay ,dựa trên xác suất hậu nghiệm của (là ) và control .
    - Cập nhật: là quá trình chính xác hoá lại dự đoán thành sau khi đã thu được measurement .

Tuy có thể biểu diễn dễ dàng dưới dạng pseudo code, nhưng Bayes filter về bản chất chỉ là thuật toán trên lý thuyết. Thực tế, hai giải thuật Particle filter và Kalman filter là hai giải thuật hiện thực hoá ý tưởng của Bayes filter, và cũng là hai giải thuật được ứng dụng nhiều nhất cho vấn đề localization của mobile robot.

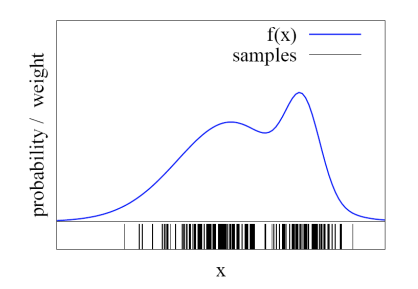
**Particle Filter**

Để ứng dụng Bayesian filter vào giải quyết bài toán dự đoán state của robot có hai phương án chính.

Truyền thống, Kalman Filter được sử dụng với giả thuyết rằng: các phân bố xác suất của các biến ngẫu nhiên đều là phân bố Gaussian và các biến ngẫu nhiên thay đổi theo một hàm tuyến tính. Vì thực tế biến ngẫu nhiên thay đổi bất tuyến tính như giả thuyết. Nên bộ lọc cải tiến Extended Kalman filter (EKF) được sử dụng thay và bỏ qua giả tuyết state thay đổi theo hàm tuyến tính. Thay vào đó, EKF dựa trên khai triển Taylor bậc 1 của hàm trạng thái (state). Bộ lọc EKF rất hiệu quả khi áp dụng trong các hệ thống có nhiễu theo phân bố Gaussian. Tuy nhiên, thực tế dễ gặp phải biến ngẫu nhiên có phân bố xác suất không phải là phân bố Gaussian (non-gaussian).

Do trên, các bộ lọc Particle filtersinh ra để giải quyết bài toán dự đoán trạng thái trong trường hợp phân bố xác suất bất kỳ. Loại bộ lọc này đang được sự dụng rộng rãi do mang nhiều ưu điểm khi ứng dụng thực tế.

Các bộ lọc thuộc loại Particle filter theo cơ chế: lấy một tập hợp hữu hạn mẫu (sample) ngẫu nhiên từ phân phối xác suất state trước đó. Các mẫu xác suất là rời rạc nên được gọi là particles. Do đó, từ mật độ của các particles, ta có thể thể hiện lại phân phối xác suất của state trước đó đó một cách gần đúng. Mỗi particle đi kèm trọng số thể hiện độ tin cậy của nó. Particle có trọng số cao nhất đồng nghĩa với xác suất xảy ra của particle này là cao nhất. Hình dưới đây thể hiện sự phân bố của các particles (samples) theo hàm phân bố xác suất f(x) ngẫu nhiên mà chúng đã được lấy mẫu.



Hình ‑ Tập hợp partiles thể hiện một phân bố xác suất ngẫu nhiên.

Phân bố xác suất trên cho thấy một số vùng rìa có xác suất thấp, xuất hiện ít particles. Để tối ưu, các particles ở vùng quanh chóp sẽ được giữ lại và particles vùng rìa sẽ bị loại bỏ nếu có trọng số thấp hơn ngưỡng định trước (threshold). Nói cách khác, bộ lọc sẽ chỉ giữ lại các particles có khả năng xảy ra cao nhất để tiếp tục quá trình ước lượng state. Nhờ đó mà giảm được một khối lượng tính toán lớn không cần thiết.

Dựa vào ưu điểm lấy mẫu rời rạc, khả năng thích ứng với phân bố xác suất bất kì và khối lượng tính toán hữu hạn, Particle filter trở thành lựa chọn tối ưu nhất để triển khai lên hệ thống số như robot. Thực tế hoá cho ý tưởng của Particle filter là giải thuật Monte Carlo filter dùng để ước lượng phân bố xác suất của state. Tên gọi của giải pháp này là Monte Carlo Localization.

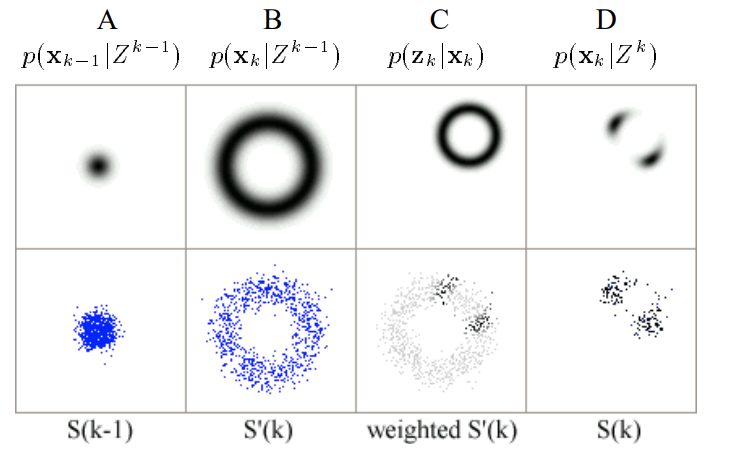
**Monte Carlo Localization**

Monte Carlo Localization [6] là giải thuật ước lượng state của mobile robot dựa trên Monte Carlo filter (Particle filter). Về tổng thể, Monte Carlo Localization dựa trên ý tưởng chung của nhóm bộ lọc Particle filter và Bayes filter. Như Bayes filter, Monte Carlo filter ước lượng trạng thái theo 2 bước:

* + - 1. Dự đoán: Dựa vào hàm phân bố xác suất , một nhóm particles là được lấy mẫu ngẫu nhiên. Mỗi particle kí hiệu là . Quá trình này được xem như áp dụng tín hiệu control vào các mẫu particle để đạt được các mẫu dự đoán . Gọi chung nhóm particle dự đoán là
      2. Cập nhật: Khi đã có dữ liệu measurement , các particles nhóm được cập nhật trọng số. Trong số này là xác suất để xảy ra tại particle . Gọi trọng số của particle là , ta có:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Cuối cùng, chỉ các particle có trọng số lớn nhất được duy trì để tiếp tục ước lượng state. Trải qua hữu hạn lần lặp hồi quy, các particle sẽ dần hội tụ tại giá trị state gần đúng với thực tế nhất. Nói cách khác, phương sai của hàm phân bố xác suất sẽ đạt giá trị cực tiểu sau vài lần lặp.



Hình ‑ Minh hoạ phân bố xác suất của vị trí robot qua các bước. Biến k đại diện cho lần lặp thứ k, thay cho t. [6]

## Lý thuyết các khái niệm trong Visual SLAM

**Feature – đặc trưng ảnh**

Trong lĩnh vực thị giác máy tính, feature (đặc trưng) là thông tin đặc biệt xuất hiện trong ảnh, thường là những vùng ảnh cụ thể mang một tính chất đặc trưng. Các feature có thể là những cấu trúc hình học đặc biệt, khả kiến, xuất hiện trong ảnh như điểm, cạnh, đường cong hoặc một vật thể nhất định. Feature cũng có thể là những cấu trúc phức tạp mà mắt thường không nhận ra được. Các feature được xác định bằng các phép toán đại số áp dụng lên ma trận ảnh. Các phép này được biết tới là phép trích đặc trưng ảnh – feature extraction. Các feature này do tính chất đặc trưng nên không thay đổi khi góc ảnh thay đổi hoặc ánh sáng thay đổi. Với toạ độ mỗi feature tìm được từ feature extraction, người ta cần biết đặc điểm của feature nhằm mục đích so sánh.



Hình ‑ Một số features giống nhau xuất hiện trong hai ảnh khác nhau.

Feature descriptor là yếu tố quan trọng góp phần vào sự hữu dụng của feature. Feature descriptor là dữ liệu (thường là vector) đại diện cho mỗi feature. Nhờ vậy người ta có thể tìm ra hai feature tương đồng qua việc tính “khoảng cách” của hai descriptor.

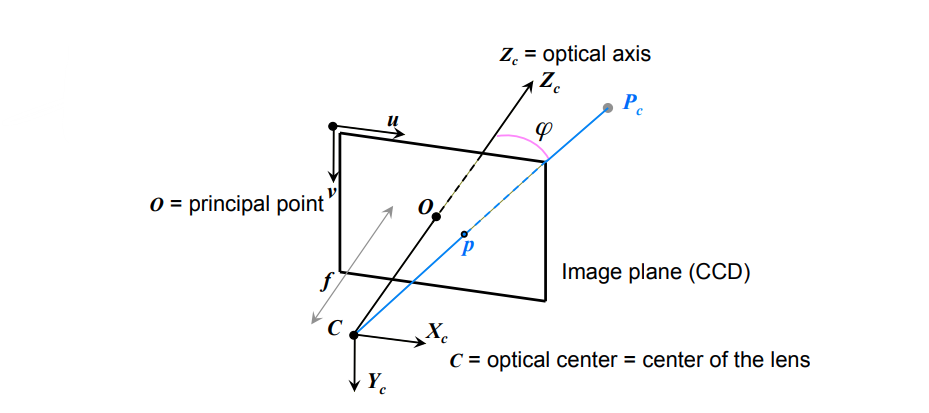
Mỗi loại feature thường kèm theo cách tính descriptor riêng. Về cơ bản, chúng đều tính toán dựa trên đặc điểm vùng ảnh quanh feature để lập thành vector đại diện cho feature đó.

Nhờ feature descriptor, các cặp feature giống nhau trong hai ảnh được tìm ra. Các cặp feature như thế đóng vai trò quan trọng cho quá trình phục hồi chuyển động của camera. Quá trình đó vốn là vấn đề visual odometry, một phần của bài toán Visual SLAM.

**Triangulation**

Triangulation là kỹ thuật ứng dụng lý thuyết về camera để tìm được của độ 3D của các feature xuất hiện trong hai ảnh. Tạm gọi **P** là điểm thực tế tạo nên feature xuất hiện trên hai ảnh. Triangulation khôi phục lại tọa độ 3D **[X,Y,Z]** của **P** khi đã biết hình chiếu của nó lên ảnh của nhiều camera, cũng có thể hiểu là nhiều ảnh của cùng một camera cùng chụp điểm **P**. Quy ước chung, mỗi ảnh được xem như được chụp từ một camera.

Toạ độ 3D của **P** được khôi phục dựa trên vị trí tương đối của ít nhất 2 camera với nhau. Thường mối quan hệ vị trí tương đối này sẽ diễn theo dạng (R | T). Trong đó, chọn một camera làm gốc và khung tọa độ của nó làm chuẩn. Khung toạ độ của camera sẽ có trục x và y cùng nằm trên mặt phẳng ảnh và trục z sẽ vuông góc với 2 trục còn lại tại gốc toạ độ là *quang tâm* của camera.



Hình ‑ Minh hoạ khung toạ độ camera. [7]

Trong ảnh minh họa trên, **C** là quan tâm của camera cũng là gốc toạ độ của khung. **O** là tâm của ảnh và đường thẳng **CO** là quang tâm của camera, cùng là **Zc** . **Pc**đại diện cho một điểm ảnh như **P** và **f** là tiêu cự của camera. Hai vector u và vector v đại diện cho hệ tọa độ 2D của ảnh có gốc toạ độ tại góc trên bên trái.

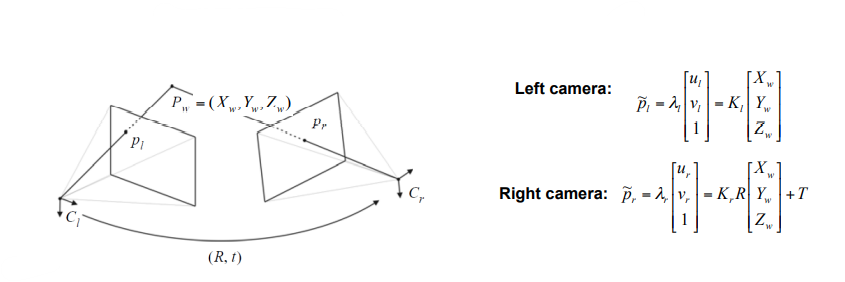
Xét trường hợp cụ thể, hai camera **Cl** (camera bên trái – left camera) và **Cr** (camera bên phải – right camera), ta có thể lập một hệ gồm hai phương trình mà trong đó toạ độ 3D [X, Y, Z] của P là ẩn. Chọn gốc tọa độ gốc tham chiếu trùng với khung toạ độ của camera bên trái, minh hoạ và biểu diễn của hai phương trình như sau:

Camera bên trái – left camera:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Camera bên phải – right camera:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |



*Hình 2‑5 Minh hoạ vị trí 2 camera và điểm P. [7]*

Trong hệ phương trình trên, Kl và Kr là hai ma trận kích thước (3x3), lần lượt mang thông số nội (intrinsic parameters) của camera trái và phải. Hai ma trận thông số này mang thông tin về tiêu cự và kích thước cảm biến, là những thông số đã biết. Còn lại, R và T là ma trận xoay (3x3) và ma trận chuyển vị (3x1). lần lượt là toạ độ hai điểm ảnh bên trái và bên phải (theo khung toạ độ camera mỗi bên).

Như vậy, với chuỗi nhiều ảnh, toạ độ thực của các feature sẽ được ước lượng sơ bộ để tối ưu lại về sau, phục vụ cho quá trình Mapping. Đồng thời, mối quan hệ về R và T giữa mỗi ảnh có thể được khôi phục lại một cách tương đối. Nhờ vậy mà quỹ đạo di chuyển ước lượng của camera (dựa trên R và T) sẽ góp một phần vào tiến trình Localization của giải thuật Visual SLAM.

**Bundle Adjustment**

Bundle Adjustment là quá trình tối ưu hoá lại những ước lượng có được từ Triangulation từng cặp ảnh trong chuỗi ảnh.

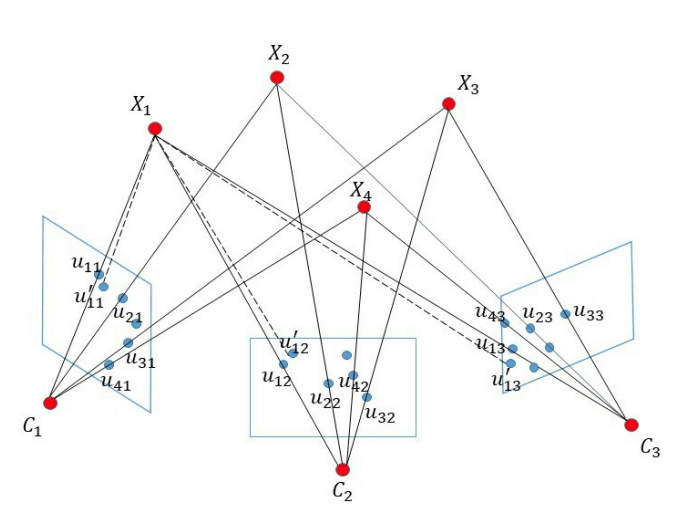
Bundle Adjustment về bản chất là quá trình tối thiểu hoá sai số. Cụ thể, một điểm có toạ độ 3D ước lượng là Xi. Khi ta tính lại hình chiếu của Xi lên ảnh (bằng phương trình (14) và (15)) thu được điểm . Quá trình tính lại hình chiếu của Xi gọi là reprojection. Do cả hai và đều là toạ độ trên hệ toạ độ ảnh () nên sai số sẽ tính theo khoảng các Euclide của hai điểm này. Sai số này là reprojection error:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Và min E:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Trong đó là toạ độ pixel của feature và là hình chiếu của điểm Xi lên camera Cj. là toạ độ pixel thu được từ phép tính reprojection.



Hình ‑ Minh hoạ cho quá trình reprojection.[8]

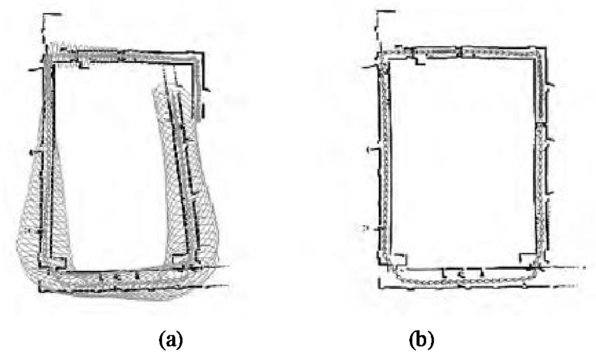
Để đạt được sai số cực tiểu, ma trận K (trong phương trình (14) và (15)) được điều chỉnh khi tính . Đây chính là sự điều chỉnh (adjustment) trong tên phương pháp.

Mặc dù phương trình (16) lấy tổng E từ camera C1 tới Cn, nhưng thực tế feature Xi không thể thấy được từ tất cả camera. Nói cách khác, chỉ một nhóm ảnh (bundle) có hình chiếu của Xi sẽ đóng góp vào quá trình điều chỉnh.

Có thể thấy phương trình (16) khi áp dụng cho *n* features và *m* ảnh thì sẽ tạo thành phương trình phi tuyến phức tạp. Tìm cực tiểu cho phương trình phi tuyến như thế vốn là lĩnh vực thuần toán nên đề cương luận văn sẽ không đi sâu. Bài báo [8] là review một cách chi tiết về mặt toán cho tiến trình Bundle Adjustment này.

**Loop Closure Detection**

Loop closure detection là phần sau cùng của các giải thuật Visual SLAM. Bước này duyệt qua toàn bộ những ảnh đã chụp trước đó nhằm phát hiện việc quay trở lại điểm đã đi qua trước đó. Mục đích của loop closure detection là để tối ưu hoá lại sai số tích luỹ của SLAM (dựa trên phương pháp Bundle Adjustment). Vì vậy, loop closure detection đặc biệt đóng vai trò quan trọng khi SLAM hoạt động trong thời gian dài và môi trường rộng lớn.

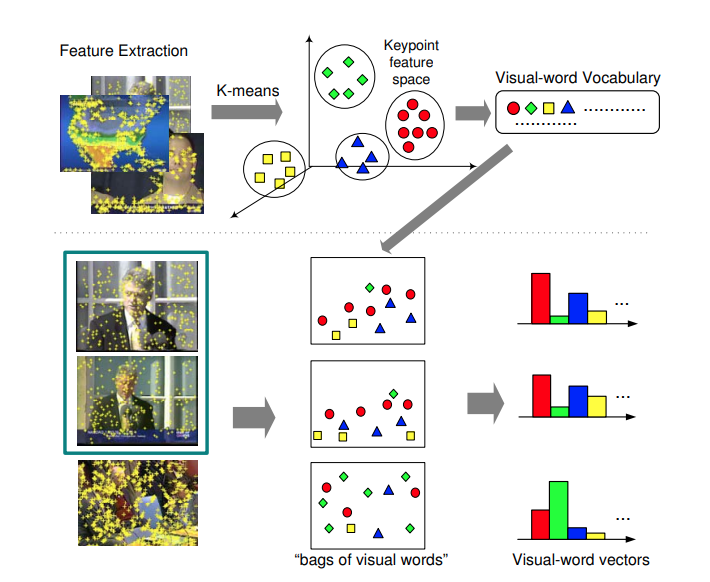


Hình 2‑7 a) Sai số tích luỹ dần trên bản đồ lập bởi SLAM; b) Sau khi phát hiện loop closure và thực hiện bundle adjustment.

Một số vấn đề phát sinh khi tìm loop closure. Về mặt ý tưởng, đơn giản nhất là so sánh ảnh cuối cùng với toàn bộ các ảnh trước đó để phát hiện vị trí đã đi qua. Nhưng thực tế, SLAM bị ràng buộc về bộ nhớ và thời gian xử lý. Việc so sánh như thế nào để thoả được hai ràng buộc đó là vấn đề được nghiên cứu khá nhiều.

Phương pháp kinh điển là dùng keyframe. Keyframe là ảnh được chọn ra dựa theo một số tiêu chí nhất định như cách nhau *n* giây hoặc có nhiều feature khác nhau. Các keyframe được lưu lại để so sánh và tìm loop closure.

Hiện tại, phương pháp tương đối hiệu quả và thông dụng là *Bag of Words (BoW).* Bag of Words dựa trên ý tưởng lập mô tả (descriptions) cho mỗi ảnh sao cho phép tính trừ hai mô tả thể hiện được độ tương đồng của hai ảnh. Cụ thể, BoW trích suất features, tính descriptor cho mỗi feature. Sau đó, chúng được phân nhóm theo thuật toán K-means clustering. Descriptor trung bình của nhóm được xem như một *visual word.*  Từ visual word đại diện nhóm và số feature trong nhóm ta có thể lập histogram. Mỗi ảnh đặc trưng bởi một histogram như thế. Chỉ có các ảnh giống nhau mới có histogram giống nhau.



Hình 2‑8 Minh hoạ quá trình trích visual words cho mỗi ảnh.

Trong Hình 4-8, có thể thấy ảnh số 1 và ảnh số 2 tương tự nhau nên histogram được minh hoạ giống nhau. Điều này xảy ra trong thực tế. Bằng cách trích xuất, lưu histogram cho mỗi keyframe, quá trình phát hiện loop closure đảm bảo được ràng buộc thời gian lẫn ràng buộc tài nguyên.

## RTAB-MAP: Real-Time Appearance-Based Mapping

Mathieu Labbé và Francois Michaud đã giới thiệu RTAB-MAP dựa trên hai tiêu chí là giải quyết những vấn đề thực tế của SLAM và đa dụng với nhiều loại cả biến. Tính thực tế của RTAB-MAP thể hiện ở việc vượt qua rào cản thực tế của các giải thuật Visual SLAM tốt nhất bấy giờ. Rào cản đó nằm ở hai hai giả định:

1. Camera không bị cản trở trong quá trình mapping
2. Môi trường có đủ feature (đặc trưng) để tính toán.

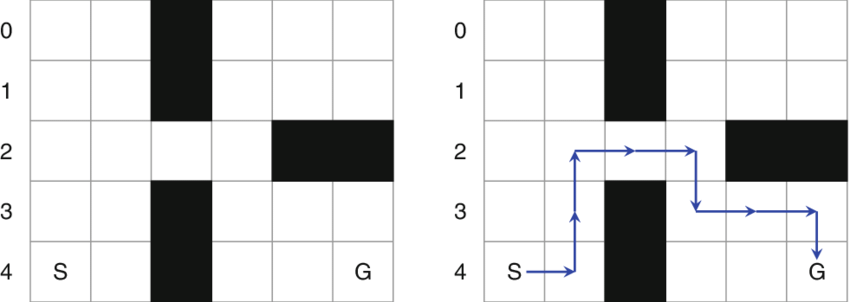
Hơn thế nữa, RTAB-MAP giải quyết được hai vấn đề khác đáng cân nhắc trong ứng dụng là: giới hạn tài nguyên tính toán khi mapping môi trường lớn.

## Nội dung 2: Djisktra và Digital Window Approach (DWM)

* + 1. **Thuật toán Dijkstra**

Tìm đường đi đến đích là tác vụ cần phải thực hiện sau khi bản đồ có được từ quá trình mapping nhờ RTAB-MAP. Thuật toán Dijkstra giải quyết bài toán tìm đường đi ngắn nhất được ứng dụng trong quá trình này.

Bằng cách chia nhỏ bản đồ thành các ô đơn vị (cell). Bản đồ được thể hiện bằng các cell là gridmap.



Hình ‑ Start (S), Goal (G) và đường đi ngắn nhất từ S->G trên gridmap đơn giản.

Mỗi cell được xem như các node/vertex hay được đề cập trong lý thuyết thuật toán Djisktra. Và như vậy, thuật tuán Dijsktra có thể tìm được đường đi ngắn nhất từ điểm đầu tới điểm đích bằng cách duyệt qua các cell và khoảng cách của nó so với điểm đầu.

|  |
| --- |
| **Dijkstra Pseudo code** |
| * For each node, n, in the graph:   + n.distance = Infinity * Create an empty list   + start.distance = 0, add start to list * While list not empty:   + Let current = node in the list with the smallest distance, remove current from list * For each node, n that is adjacent to current:   + if n.distance > current.distance + length of edge from n to current   + n.distance = current.distance + length of edge from n to current   + Add n to list if it is not there already |

Bảng ‑. Pseudo code của thuật toán Dijkstra.

* + 1. **Thuật toán Dynamic Window Approach (DWA)**

Sau khi có được đường đi tổng quát đến đích, mô hình cần hoạch định đường đi cục bộ (local path planning) nhằm đảm bảo không va phải các chướng ngại vật chuyển động xung quanh mô hình. Những vật cản này làm hạn chế khả năng di chuyển của robot do nhiều trường hợp. Một trường hợp dễ thấy là vật cản xuất hiện bất ngờ và nằm trên đường đi ngắn nhất đã hoạch định bời thuật toán Dijkstra.

Output của DWA là lệnh điều khiển vận tốc tức thời của mô hình, bao gồm hai thành phần là vận tốc dài và vận tốc góc. DWA tính toán dựa trên giới hạn vật lý (vận tốc) và ràng buộc tránh va chạm với vật cản. Các bước tính toán của DWA được trình bày ngắn gọn bằng 2 bước sau đây:

**Đặt giới hạn:**

Vì DWA hoạch định các mô hình di chuyển bằng cặp vận tốc dài và vận tốc góc nên DWA xem đường di chuyển là các cung tròn.

Do giới hạn vật lý, DWA bỏ qua các vị trí quá xa mà mô hình không thể đi tới trong một khoảng thời gian tính toán (time interval), thường là 0.25s. Các vị trí còn lại được đưa vào tính toán chỉ còn lại một vùng không gian xung quanh mô hình, gọi là “window”.

DWA bỏ qua các cặp vận tốc của cung tròn gây ra khả năng va chạm với vật cản. Chỉ giữ lại các cặp vận tốc thoả tiêu chí “có thể dừng lại trước va chạm”.

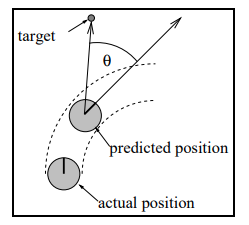
**Tối ưu hoá:**

Thuật toán đề ra hàm mục tiêu cho việc tối ưu hoá:

Trong đó

* là hướng của mô hình so với hướng tới đích. Thành phần này được đưa vào vì nó có thể dự đoán vị trí của mô hình trong time interval kế tiếp.
* đại diện cho khoảng cách từ mô hình tới vật cản gần nhất mà cung tròn đi qua. Nếu không có vật cản trên cung tròn thì thành phần này sẽ là một hằng số lớn.
* là thành phần đại diện vận tốc dài của mô hình.

Cuối cùng, output của thuật toán là cặp làm cho hàm mục tiêu lớn nhất.



Hình ‑ Heading của mô hình robot.

## Nội dung 3: Robotics Operating System (ROS)

*Tìm hiểu về cấu trúc của ROS và lý thuyết về các khối chức năng/cơ chế thông tin của ROS.*

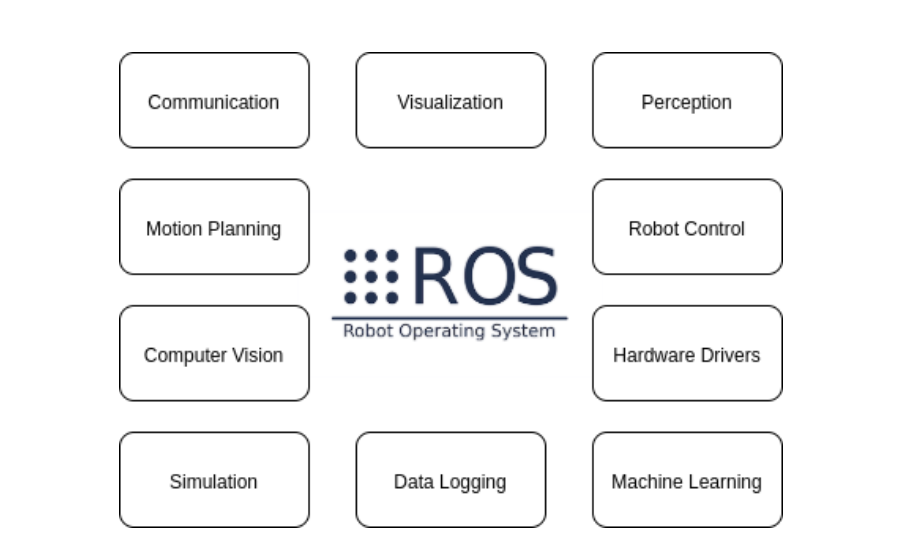
Robot Operating System – ROS là hệ thống bao gồm các giải thuật mã nguồn mở, driver phần cứng và rất nhiều công cụ phát triển phần mềm dành cho robot. Mặc dù được gọi là “hệ điều hành” (Operating System) nhưng ROS không có cấu trúc như một hệ điều hành. ROS dựa trên 3 concept chính chia theo 3 level:

1. *ROS Filesystem Level*: Thành phần này quản lý, phân cấp mã nguồn phần mềm và dữ liệu lưu trên bộ nhớ theo thứ tự, phương thức quy định bởi ROS. Nó tổ chức phần mềm/dữ liệu thành các Packages, Metapackages, Message types, Service types, …
2. *ROS Computation Graph Level*: Thành phần này quản lý việc trao đổi dữ liệu giữa các phần mềm và giữa các hệ thống trong ROS. Chúng được tổ chức thành các thành phần như Nodes, Master, Parameter Server, messages, services, topics, …
3. *ROS Community Level*: Thành phần này giúp quản lý sự phân tán trong mã nguồn mở. Nó giúp việc sử dụng, phát triển ROS trở nên thuận tiện hơn bằng cách quản lý việc phát triển các lớp (layers) phần mềm và tài liệu, đóng gói các phần mềm ổn định thành các “stack” trong ROS và phân phối tới các nhà phát triển thông qua GIT (một hệ thống quản lý source code). Mã nguồn, tài liệu phát triển của các stack, thư viện, framework được quản lý công khai qua GIT.

Nói chung, những phần mềm được tích hợp vào ROS vốn là những công cụ dành cho các lĩnh vực hẹp hơn như thị giác máy, xử lý ảnh, điều khiển, giả lập, ... Chúng được liên kết thông qua cơ chế riêng của ROS nhằm đưa vào ứng dụng trong lĩnh vực robotics.

Trong số đó, RVIZ là một phần mềm hạt nhân cực kỳ quan trọng của ROS. RVIZ giúp trực quan hoá số liệu, dữ liệu làm đơn giản hóa quá trình phát triển robot.

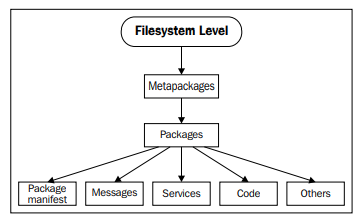
ROS cũng tích hợp thư viện xử lý ảnh OpenCV hỗ trợ phát triển robot ứng dụng thị giác máy tính. Các bản phân phối ROS mới ngày càng tích hợp nhiều frameworks, thư viện mã nguồn mở trong lĩnh vực Machine Learning hỗ trợ việc triển khai trí tuệ nhân tạo cho robot.



*Hình 4-4 Những gói phần mềm được tích hợp trong ROS.*

## ROS Filesystem Level

Cấp quản lý Filesystem của ROS đặt ra quy chuẩn trong cách tổ chức hệ thống file/folder khi phát triển phần mềm.

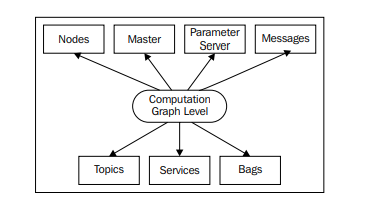


Hình 2‑11 Cấp quản lý Filesystem của ROS.

1. Packages: Packages là hạt nhân của ROS. Package có vai trò là thành phần tối thiểu cấu thành nên một chương trình trong ROS. Nó thường chứa các chương trình, file cấu hình, …
2. Package manifests: Package manifests chứa thông tin mô tả về package, về license, về các phần mềm phụ thuộc, về thông tin biên dịch, … Chúng được quản lý bới file package.xml
3. Metapackages: Là tập hợp của nhiều packages được đóng gói và biên dịch thành một Metapackage.
4. Metapackage manifests: Tương tự như package manifests về chức năng, cũng là file xml nhưng có một số ràng buộc nhất định về cấu trúc.
5. Message (msg) types: Message là gói thông tin trao đổi giữa các tiến trình (processes) đang hoạt động. ROS có nhiều loại message types, mỗi loại message type có cấu trúc riêng và được mô tả trong một file định dạng “.msg”. File mô tả cấu trúc cho các message types thường nằm trong folder “package/msg/”.
6. Service (src) types: Service types mô tả cấu trúc dữ liệu trong các gói thông tin dạng request-response được trao đổi giữa các tiến trình. Mô tả cho các Service types thường nằm trong folder “package/srv/” và có định dạng “.msg”.

## ROS Computation Graph Level

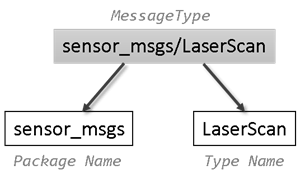
Đây là cấp quản lý các tiến trình chạy trong hệ thống. Cấp quản lý này tạo ra một mạng lưới kết nối chặt chẽ các thành phần để xử lý dữ liệu một cách khoa học.



Hình 2‑12 Các thành phần của cấp Computation Graph

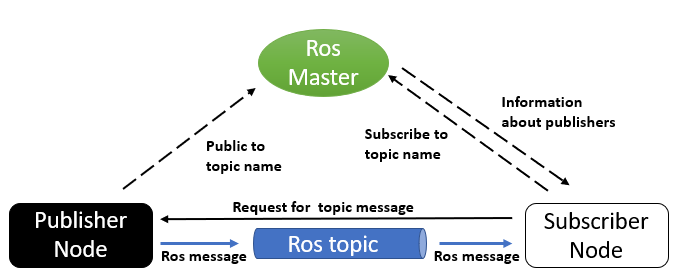
Ở cấp này, các thành phần bao gồm:

1. Nodes: Node là tiến trình (process) chạy trên CPU được tạo ra bới người dùng. Các nodes được liên kết với nhau thành một mạng thông tin liên lạc với messages và sevices. Mã nguồn của nodes phải được viết theo thư viện roscpp (C/C++) hoặc rospy (Python).
2. Master: Master là chương trình lõi của ROS. Master quản lý toàn bộ các node đang chạy. Nó thiết lập kết nối giữa các node. Chỉ cần một master cho hệ thống thì có thể quản lý nhiều nodes trên nhiều thiết bị, miễn là các thiết bị kết nối với nhau qua cơ chế TCP/IP.
3. Parameter server: Parameter server là chương công cụ quản lý thông số cấu hình của toàn bộ các node đang hoạt động. Nhờ parameter server, các thông số cấu hình của node sẽ không cố định mà có thể thay đổi khi node đang chạy, giúp cho node hoạt động linh hoạt hơn.
4. Messages: Các node thông tin với nhau thông qua messages. Message là các gói dữ liệu có cấu trúc thống nhất theo message type được định nghĩa trong mã nguồn của node.



Hình 2‑13 Tên message type mang thông tin về package định nghĩa nó (Package name)và tên file chứa định nghĩa (Type Name)

1. Topics: Topics là các luồng message được các nodes tạo ra và đăng kí tới Master. Master điều hướng messages theo cơ chế publish/subscribe. Nghĩa là, một node sẽ khởi tạo và gửi messages lên topic và có thể có một hoặc nhiều node khác subscribe tới topic đó. Khi có message mới được gửi (publish) lên thì tất cả các node đã subscribe đều nhận được message này. Mỗi topic chỉ trao đổi loại message theo message type của node tạo ra nó.



Hình 2‑14 Minh hoạ cơ chế truyền thông qua topic giữa hai node. Master quản lý việc quảng bá topic mới và quản lý đăng ký topic của các node.

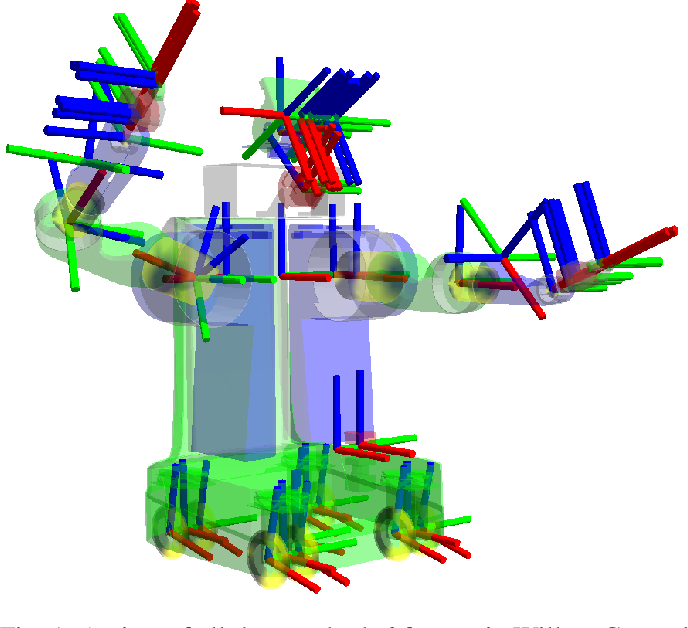
1. Services: Không giống như nodes, services là luồng thông tin được tạo bởi node theo cơ chế request/response. Nói cách khác, khi một node khởi tạo sevice, nó sẽ nhận các yêu cầu (requests) từ một node A và trả về chính node A đó các gói thông tin hồi âm (responses). Các responses tuân theo cấu trúc được mô tả trong các file service types.
2. Bags: Bags là định dạng file ghi lại nội dung, thời gian và nhiều thông tin khác của message. Nhờ vậy các thông tin quan trọng như dữ liệu cảm biến có thể được lưu lại phục vụ quá trình tìm lỗi hoặc thử nghiệm.

## ROS Community Level

1. Distributions: Distribution là tập hợp của các thành phần hạt nhân của ROS kèm theo các gói metapackages lớn được biên dịch sẵn để người dùng cài đặt.
2. Repositories: Là các địa chỉ lưu trữ mã nguồn giúp chia sẻ, đóng góp mã nguồn phần mềm ROS. Mã nguồn của nhiều giải thuật triển khai trên ROS được tổ chức thành các Repositories này.
3. The ROS Wiki: ROS Wiki là trang web chứa tài liệu phát triển, sử dụng của các phần mềm ROS.
4. Bug Ticket System: Đây là hệ thống thông báo lỗi của ROS.
5. ROS Answer: ROS Answer là forum nơi để thảo luận về các vấn đề liên quan tới ROS.
6. ROS Blog: ROS Blog là trang web tin tức về ROS và lĩnh vực Robotics.

## ROS TF Coordinate Transform

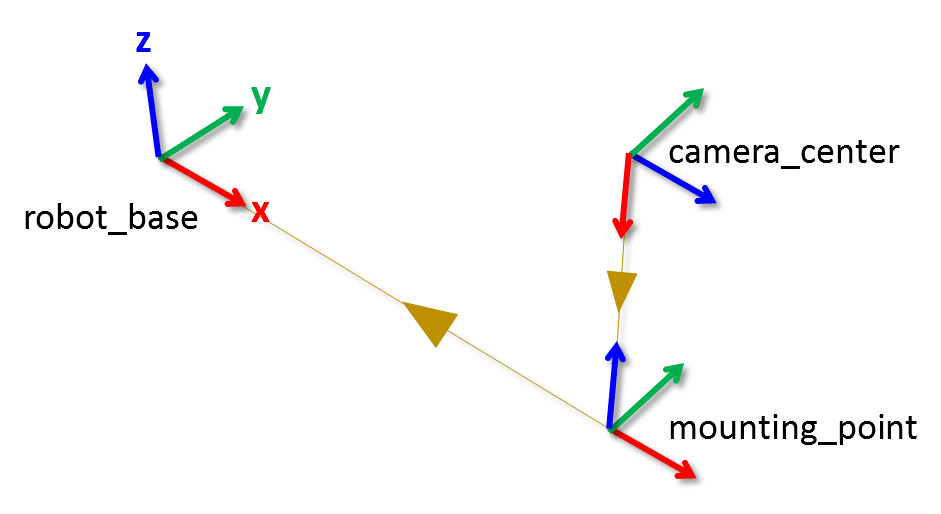
ROS TF là công cụ quản lý mối quan hệ về vị trí giữa các khung toạ độ cấu thành của robot. Mỗi bộ phận của robot được gán một khung toạ độ (coordinate frame) để dễ dàng cho việc tính toán vị trí tương đối giữa các thành phần trong quá trình làm việc (bài toán động học thuật và động học nghịch).



Hình 2‑15 Minh hoạ các khung toạ độ - coordinate frames của robot.

Trong phạm vi đề cương luận văn, mối quan hệ vị trí giữa cảm biến và thân robot rất có ý nghĩa khi giải quyết bài toán tự hành. Bởi phép đo khoảng cách tới vật thể là khoảng cách tới khung toạ độ của cảm biến chứ không phải khoảng cách tới khung toạ độ trên thân robot. Khi có nhiều cảm biến, TF giúp thống nhất dữ liệu của chúng về cùng góc nhìn từ khung toạ độ robot. Nói cách khác, TF chuyển đổi dữ liệu khi quan sát từ các khung toạ độ cảm biến sang khung toạ độ robot khi đã biết vị trí tương đối giữa các khung toạ độ này. Vị trí tương đối khung toạ độ thân robot và khung toạ độ camera được thể hiện trong Hình 4-14.

Ở phương diện toàn cục, đề tài quan tâm đến quá trình di chuyển của robot nhằm mục đích định vị (Localization). Giải thuật phục hồi quá trình di chuyển như vậy thường có output là ma trận chuyển vị giữa hai khung toạ độ của 2 ảnh liên tiếp. Do đó, việc phục hồi quỹ đạo về chung một khung toạ độ tham chiếu đòi hỏi phải nhân liên tục các ma trận chuyển vị. TF giúp giảm gánh nặng tính toán đó.



Hình 2‑16 Minh hoạ vị trí tương đối giữa 3 khung toạ độ. robot\_base là khung toạ độ của thân robot. mouting\_point là khung toạ độ của ngàm giữ camera. camera\_center là khung toạ độ camera tương tự Hình 4-4.

Trong ROS, mọi sự thay đổi của các khung toạ độ được truyền nhận qua topic /tf. Message trong topic /tf là dạng geometry\_msgs/TransformStamped[].

|  |  |
| --- | --- |
|  | *geometry\_msgs/TransformStamped[]* |
| 1  2  3 | Header  header # contain time stamp + parent\_frame\_id string  child\_frame\_id # the frame\_id of the child frame  Transform transform |

Bảng 2‑3 Cấu trúc geometry\_msgs/TranforStamped[].

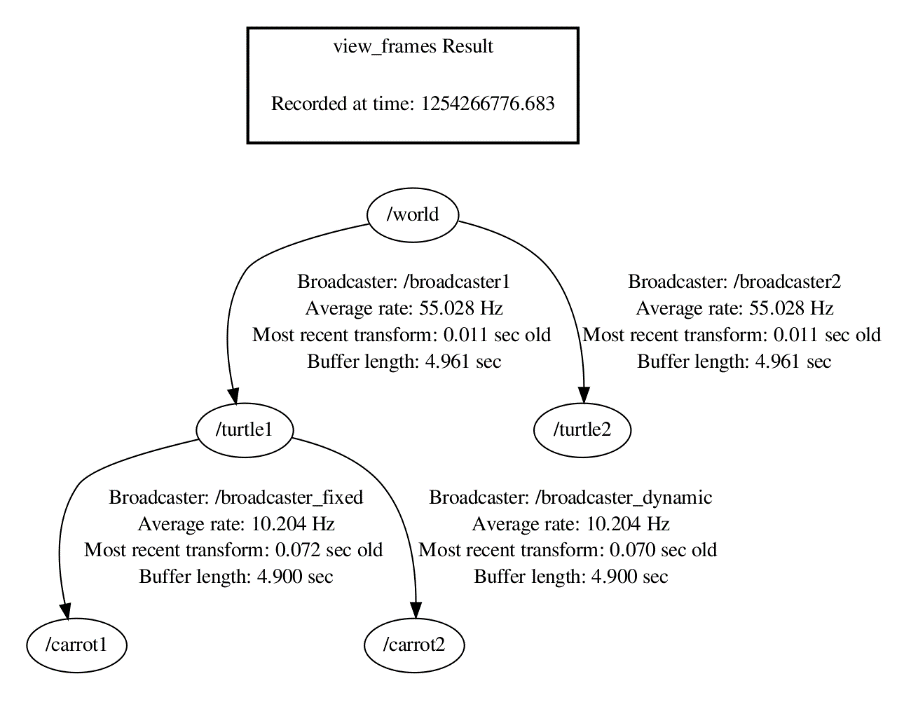
Trong đó, biến header mang thông tin về mốc thời gian và ID của khung toạ độ tham chiếu parent\_frame\_id (parent\_frame\_id đã được định nghĩa với ROS trước đó). Biến transform mang thông tin về vị trí tương đối của khung toạ độ child\_frame\_id so với khung toạ độ parent\_frame\_id. Transform là cấu trúc dữ liệu chứa 2 thành phần:

|  |  |
| --- | --- |
|  | geometry\_msgs/Transform[] |
| 1  2  3  4  5  6  7  8 | # This represents the transform between two coordinate frames in free space. Vector3 translation  # Vector3: [double64 x, double64 y, double64 z]  # rotation is presented by an orientation in free space in quaternion form.  Quaternion rotation  # Quaternion: [double64 x, double64 y, double64 z, double64 w] |

Bảng 2‑4 Cấu trúc geometry\_msgs/Transform[].

Định nghĩa cấu trúc dữ liệu của transfom *geometry\_msgs/Transform[]* dựa trên translation T[x, y, z] và rotation dạng quaternion Q[x, y, z, w]. Quaternion Q là dạng biểu diễn rotation một cách gọn hơn khi chỉ cần 4 phần [x, y, z, w]. Hình thức ngắn gọn của quarternion có ý nghĩa quan trọng khi tính toán. Do vậy, ROS dùng quaternion làm hình thức biểu diễn phép xoay. Trong khi đó, output của giải thuật SLAM là rotation R và translation T trong hệ toạ độ Decade. R là một ma trận kích thước 3x3. Việc chuyển đổi qua lại giữa R và Q được [9] trình bày chi tiết về mặt toán. Thư viện ROSCPP đã có hàm hỗ trợ chuyển đổi.

ROS hỗ trợ công cụ tf view\_frame để debug và trực quan hoá mối quan hệ giữa các khung toạ độ bằng sơ đồ gọi là tf\_tree. Nhiều thông tin hữu ích về các node subscribe tới topic /tf cũng được liệt kê trong sơ đồ tf\_tree.

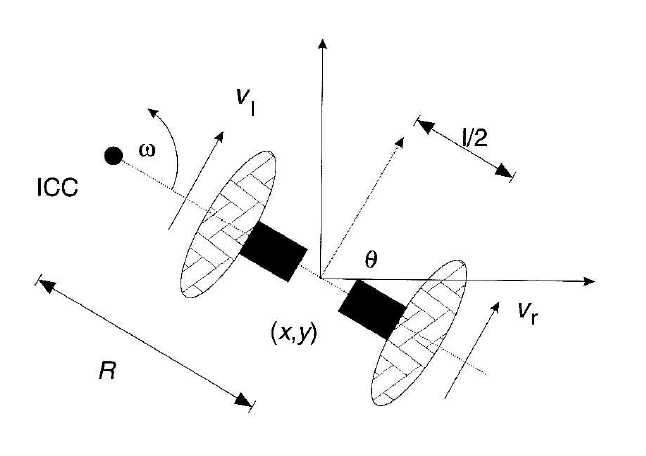


Hình 2‑17 TF tree cho thấy mối quan hệ giữa 5 khung toạ độ: /world, /turtle1, /turtle2, /carrot1, /carrot2. Trong đó /world là khung toạ độ tham chiếu toàn cục. Broadcaster là tên node publish thông tin của mỗi khung toạ độ.

## Nội dung 5: Lý thuyết phần cứng

## Differential drive – Lái vi sai

Cơ chế lái vi sai – differential drive là cơ chế di chuyển ứng dụng nhiều trong lĩnh vực Robotics. Cơ chế lái vi sai thường dùng hai motors có trục cùng nằm trên một đường thẳng. Mỗi motor được điều khiển tiến – lùi với vận tốc độc lập nhau. Do vậy, khi vận tốc mỗi bên bánh xe khác nhau sẽ dẫn đến robot di chuyển xoay quanh một điểm ICC nằm trên đường thẳng qua trục. ICC là viết tắt của thuật ngữ tiếng anh Instantaneous Center of Curvature.



Hình ‑ ICC nằm trên đường thằng qua trục động cơ.

Bằng cách thay đổi tốc độ quay của mỗi motor, ta có thể thay đổi cách robot chuyển động. Vì cùng nằm trên đường thẳng qua ICC, tốc độ góc tại ICC của hai bánh xe là như nhau, kí hiệu . Khi đó, vận tốc dài của hai bên được tính theo như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

Trong đó:

* : Khoảng cách giữa hai bánh xe.
* , : là lần lượt vận tốc dài của bánh xe bên phải và bánh xe bên trái.
* R: là khoảng cách từ ICC đến trung điểm giữa hai bánh xe.

Từ (18) có thể suy ra mối quan hệ giữa vận tốc dài của robot, vận tốc góc và vận tốc mỗi bánh xe. Mối quan hệ được biểu diễn như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (19) |
|  |  | (20) |

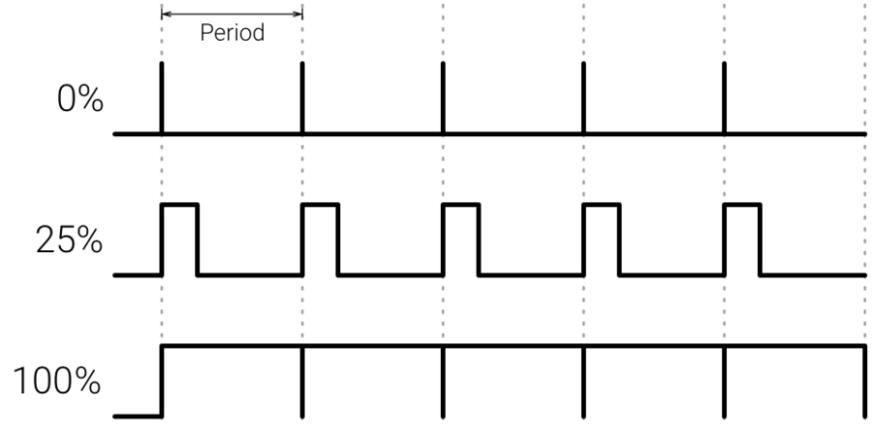
Hai công thức trên đặc biệt quan trọng cho vấn đề điều khiển robot. Bời vì message điều khiển gửi từ ROS luôn ở dạng tốc độ dài (m/s) và tốc độ góc (rad/s). Từ (19) và (20), tốc độ dài mỗi bánh xe sẽ được tính và quy đổi đơn vị sang vòng/phút để làm giá trị đặt cho PI controller.

## Pulse Width Modulation – PWM.

Pulse Width Modulation là kỹ thuật điều chế độ rộng xung vuông trong mỗi chu kì xung. Phương pháp PWM được áp dụng để điều chỉnh điện áp và dòng điện ngõ ra thông qua mức xung. Nói các khác, giá trị trung bình của điện áp (và dòng điện) cung cấp cho tải được kiểm soát bằng cách thay đổi việc đóng cắt giữa nguồn và tải với tốc độ rất nhanh. Tỉ lệ giữa thời gian đóng và chu kì xung càng lớn thì công suất cấp cho tải càng lớn. Thuật ngữ duty cycle đại diện cho tỉ lệ này. Duty cycle thường được viết dưới dạng phần trăm %.

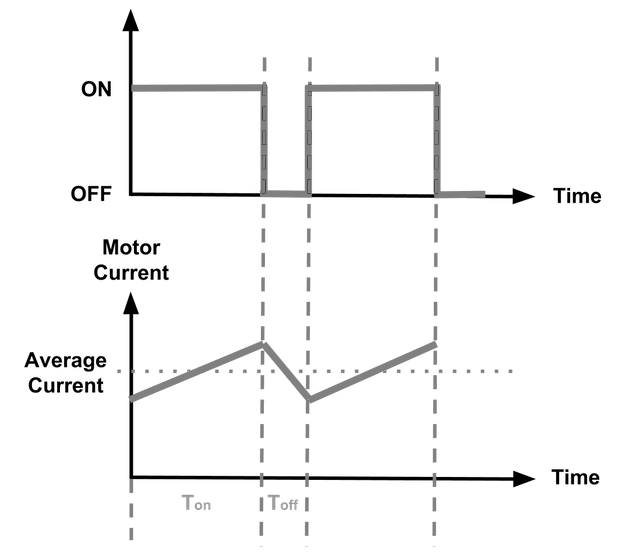
|  |  |  |
| --- | --- | --- |
|  | (%) | (21) |

Với là khoảng thời gian xung ở mức cao; là chu kì xung.



Hình ‑ Minh hoạ cho duty circle và độ rộng xung tương ứng.

Tần số của xung PWM tuỳ vào tải. Đối tượng điều khiển PWM của đề tài là động cơ DC. Do động cơ DC có mạch tương đương gốm 2 thành phần cuộn dây L và điện trở R, nên hoạt động như bộ lọc thông thấp. Tần số PWM phải đủ lớn sao cho chỉ thành phần hài cơ bản của điện áp đi qua động cơ. Nói cách khác, tần số PWM phải đủ lớn để đảm bảo dòng qua cuộn dây L là liên tục và ổn định.

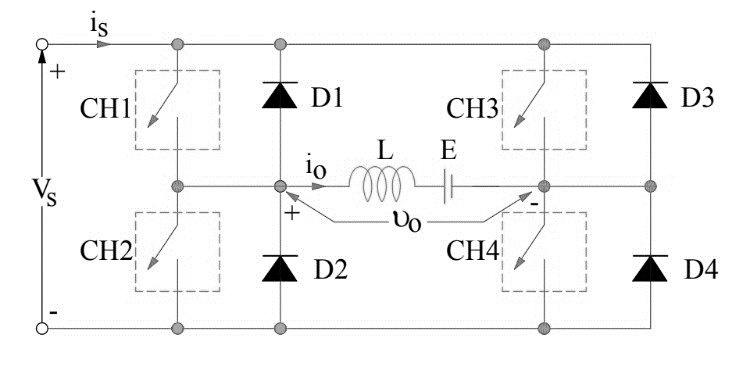


Hình ‑ Dòng điện qua cuộn dây được giữ liên tục khi tần số PWM đủ lớn.

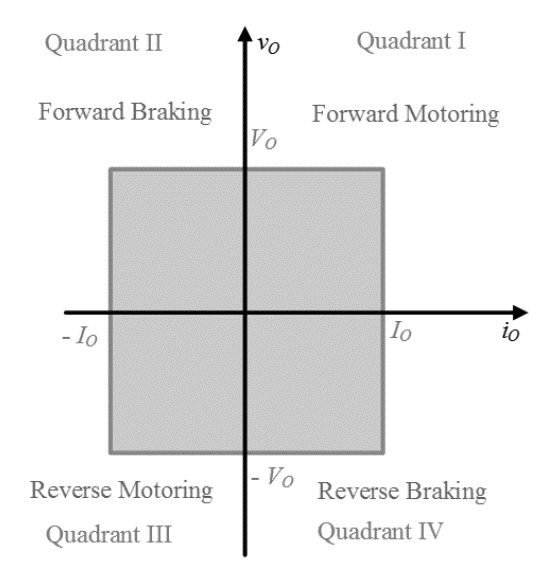
## Mạch Four-quadrant Chopper

Mạch chopper là mạch chuyển đổi điện áp DC-DC, trong đó, điện áp ngõ vào là hằng số và điện áp ngõ ra điều chỉnh được. Chopper điều chỉnh điện áp trung bình ngõ ra thông qua cơ chế đóng cắt tần số cao như PWM. Nhờ vậy, các mạch chopper thường dùng như mạch động lực dành cho điều khiển động cơ.

Các mạch chopper được phân loại theo góc phần tư hoạt động (quadrant), bao gồm 4 góc có đặc tính điện áp, dòng điện khác nhau. Mạch four-quadrant chopper làm việc trên cả 4 góc phần tư đó. Nhờ vậy, four-quadrant chopper là thông dụng nhất trong điều khiển động cơ.



Hình 2‑21 Mạch four-quadrant chopper điều khiển động cơ.



Hình 2‑22 Bốn góc phần tư hoạt động của các mạch chopper.

Hình (4-20) thể hiện mạch điều khiển động cơ với cuộn dây L và sức điện động tự cảm E. là điện áp và dòng điện từ nguồn DC. CH1, CH2, CH3 và CH4 là bốn công tắc bán dẫn, có thể là BJT, MOSFET hoặc IGBT.

Hình (4-21) đề cập đến 4 góc phần từ hoạt động (quadrant) của tải động cơ được điều khiển bởi mạch chopper ở mỗi quadrant. Trong đó lần lượt là điện áp và dòng điện trung bình ở ngõ ra chopper (có dấu phụ thuộc chiều quy ước như hình 4-20).

Cụ thể 4 góc phần tư có đặc tính như sau:

Forward Motoring: CH4 ON, CH2 và CH3 OFF. CH1 đóng mở theo PWM. Trong 1 chu kì PWM.

* + - * CH1 ON: và tăng dần qua CH1 và CH4.
      * CH1 OFF: giảm dần qua CH4 và D2.

Lặp lại cho kì PWM tiếp theo. Khi chu kì đủ nhỏ, dòng qua cuộn dây gần như không thay đổi. Quy ước chiều quay động cơ ở quadrant này là thuận chiều (forward motoring).

Forward Braking: Giả sử động cơ đang quay thuận chiều do tác động bên ngoài. CH1, CH3 và CH4 OFF. CH2 đóng mở theo PWM.

* + - * CH2 ON: Sức điện động E tạo dòng kín qua CH2 và D4, . Cuộn dây L tích năng lượng.
      * CH2 OFF: Biến thiên dòng điện trên cuộn L gây ra điện áp làm D1, D4 dẫn tạo dòng hồi năng lượng về nguồn.

Lúc này động cơ hoạt động như máy phát, chuyển lực tác động thành điện năng hồi về nguồn và cùng chiều quay quadrant I nên gọi là forward braking.

Reverse Motoring: CH1 và CH4 OFF, CH2 ON. CH3 đóng mở theo PWM.

* + - * CH3 ON: Điện áp ngõ ra , làm cho sức điện động tự cảm E đảo cực. Dòng điện đi từ nguồn ra tải , cuộn L tích năng lượng.
      * CH3 OFF: Chênh áp trên cuộn dây L làm D4 dẫn, tạo dòng , giảm về 0.

Điện áp qua tải đảo chiều nên động cơ quay chiều ngược lại so vơi quadrant I (reverse motoring).

Reverse Braking: CH1, CH2 và CH3 đều OFF. CH4 đóng mở theo PWM. Giải sử động cơ bị ngoại lực tác động xoay ngược chiều.

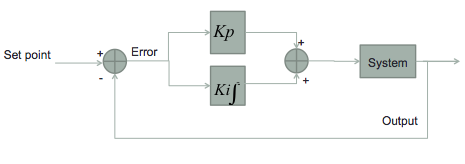
* CH4 ON: Sức điện động E đổi chiều, tạo dòng qua D2 và CH4. Khi đó . Cuộn dây tích năng lượng.
* CH4 OFF: Dòng giảm gây ra lệch áp , khiến D2, D4 dẫn gây ra dòng hồi về nguồn.

Động cơ xoay ngược chiều, chuyển ngoại lực thành năng lượng nạp về nguồn nên được gọi là reverse braking.

Do ưu điểm hoạt động được ở cả bốn góc phần tư trên, four-quadrant chopper trong điều khiển động cơ có thể tận dụng năng lượng của lực tác động lên động cơ để sạc lại pin. Các nguồn năng lượng có thể kể tới như quán tính, thế năng khi xuống dốc, …

## Bộ điều khiển PI (PI controller)

PI controller là biến thể của bộ điều khiển Proportional Integral Derivative (PID), trong đó PI chỉ sử dụng hai khâu thành phần là Proportional (khâu tỉ lệ) và Integral (khâu tích phân). PI controller là thuật toán điều khiển phổ biến nhất do cấu trúc đơn giản, thiết kế nhanh và dễ chọn hệ số khi tinh chỉnh.



Hình 2‑23 Bộ điều khiển PI vòng kín có output feedback cho tín hiệu liên tục.

Đề tài chọn sử dụng bộ điều khiển PI do những ưu điểm trên và bỏ qua thành phần Derivative (khâu vi phân) vì thành phần này nhạy với nhiễu.

PI controller dựa trên input là sai số giữa tín hiệu đặt SP (set point) và tín hiệu output của đối tượng điều khiển PV (measured process variable). Trong hệ thống rời rạc, kí hiệu sai số là

= SP – PV (22)

Khi đó công thức tính hai thành phần proportional và integral tại mẫu như sau:

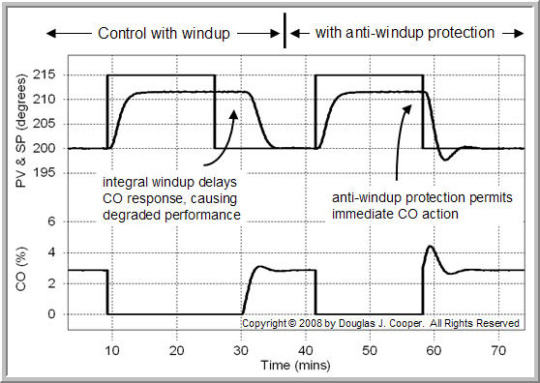
|  |  |  |
| --- | --- | --- |
|  |  | (23) |
|  |  | (24) |

Output của bộ điều khiển là tổng hai thành phần:

(25)

Do vận tốc động cơ có điểm bão hoà trên, sai số duy trì khi vận tốc đặt vượt quá giới hạn vật lý. Điều này làm nảy sinh vấn đề integrator wind-up. Wind-up xảy ra khi thành phần liên tục tích luỹ qua từng bước hồi quy kéo theo tăng không giới hạn. Một khi sai số giảm, giá trị vẫn quá lớn và mất một khoảng thời gian mới trở về giá trị bình thường làm đáp ứng chậm đáng kể.

Vấn đề wind-up được giải quyết bằng cách giới hạn output và bằng giá trị thực nghiệm sao cho hiện tượng trễ thoả tiêu chuẩn đặt ra.

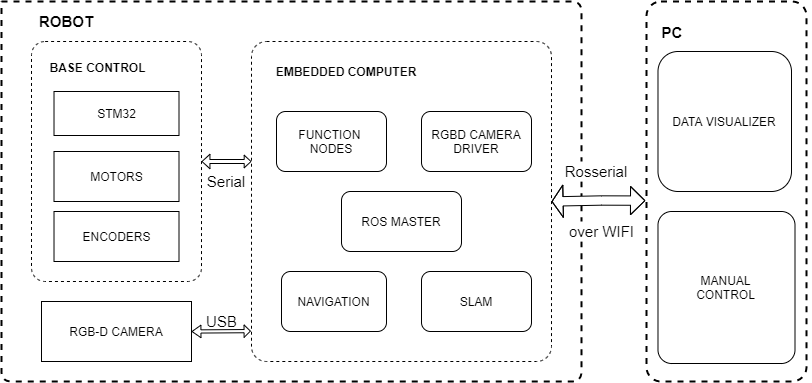


Hình 2‑24 So sánh đáp ứng của bộ điều khiển khi có wind-up và khi đã áp dụng anti-windup. PV, SP lần lượt là giá trị feedback (measure process variable) và giá trị đặt (set point). CO là output của bộ điều khiển (control output).

# THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

## Tổng quan hệ thống

Để có hướng đi đúng trong quá trình phát triển mô hình, cấu trúc tổng quan của hệ thống cần được thiết kế trước hết. Về chức năng hệ thống cấu thành từ các khối liên kết như sau:



Hình ‑ Sơ đồ tổng quan của hệ thống.

Về mặt chức năng hệ thống bao gồm 3 khối lớn:

Khối Embedded Computer là máy chủ chạy ROS và các giải thuật quan trọng. Khối này bao gồm các khối nhỏ là:

* + 1. Khối Function Nodes: là đại diện cho các node làm nhiệm vụ truyền nhận thông tin với Base Control và PC thông qua giao thức Rosserial.
    2. Khối RGB-D Camera Driver: là khối đại diện cho các node quản lý, xử lý thông tin được gửi về từ RGB-D Camera thông qua giao tiếp USB.
    3. Khối Navigation là khối chạy giải thuật global và local navigation dựa trên thông tin về bản đồ được cung cấp bởi giải thuật SLAM.
    4. Khối SLAM là khối triển khai giải thuật SLAM. Input là thông tin wheel odometry từ khối base control và dữ liệu từ RGB-D Camera. Output là thông tin về bản đồ, vật cản, vị trí robot được publish lên các topic liên quan.
    5. Khối ROS Master là trung tâm thông tin của toàn hệ thống. Các topics, services, nodes, ROS clients được quản lý bởi ROS Master.

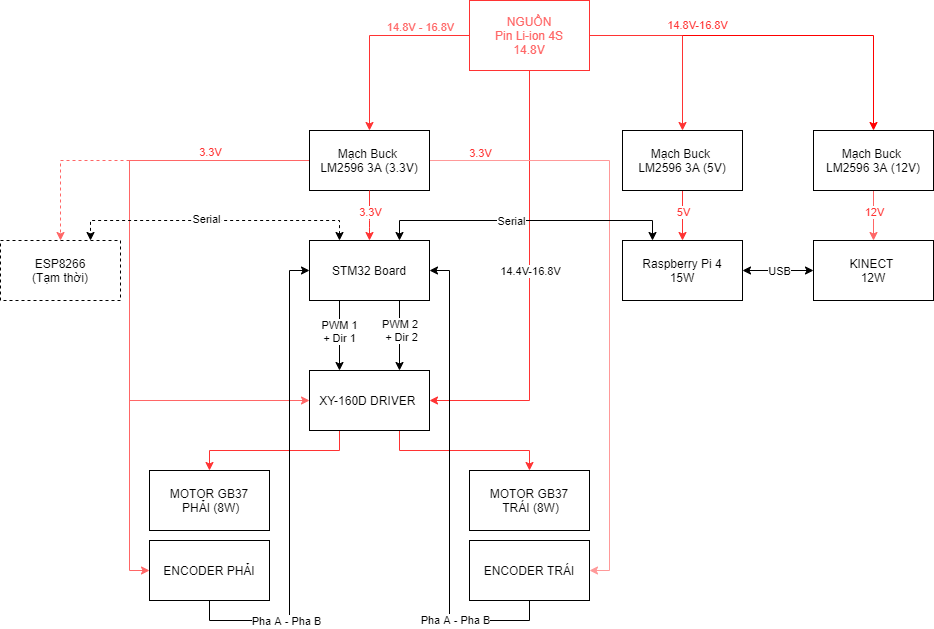
Khối Base Control điều khiển phần cứng. Input là thông tin về vận tốc mỗi bánh xe. Output là thông tin vận tốc thực thu được từ encoder.

1. Khối STM32 là khối phần cứng để triển khai thuật toán PI Controller điều khiuển tốc độ động cơ. STM32 cũng đảm nhiệm tính vận tốc bánh xe từ encoder.
2. Khối Motors là khối phần cứng bao gồm 2 motor để di chuyển robot theo cơ chế lái vi sai.
3. Khối Encoders bao gồm 2 encoder. Dữ liệu từ encoder là dữ liệu hồi tiếp cho PI Controller, cũng là dữ liệu để tính vận tốc phục vụ wheel odometry.

Khối PC là khối giao tiếp với người dùng, hiển thị trực quan dữ liệu về bản đồ và vị trí robot, đường đi và vật cản. Khối này có input là thông tin về bản đồ, vị trí robot, đường đi,.. Output là lệnh điều khiển base control trực tiếp từ người dùng hoặc đích đến được người dùng đặt trước.

1. Khối Data Visualizer chứa các phần mềm trực quan hoá dữ liệu như RVIZ phục vụ hiển thị bản đồ cho người dùng.
2. Khối Manual Control nhận thông tin điều khiển từ người dùng khi khối thuật toán navigation không hoạt động.

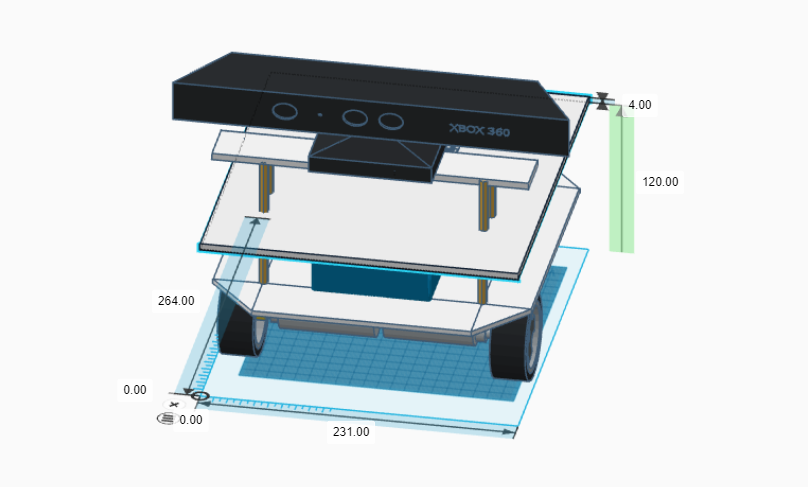
Từ cách phân tích tổng quan của hệ thống, đề tài thiết kế cách đơn vị phần cứng và mối tương quan chức năng của cúng như sau:



Hình 3‑2 Sơ đồ phần cứng chi tiết.

Thông qua sơ đồ phần cứng trên, các thành phần được lựa chọn dựa trên những yêu cầu và lý do được trình bày chi tiết ở các mục tiếp theo.

## Khung



Hình 3‑3 Kích thước mô hình. Dài 26.4 cm. Rộng 23.1 cm. Cao 27 cm khi đã bao gồm Kinect.

Khung robot thiết kế theo cơ chế lái vi sai gồm hai bánh xe chủ động, được điều khiển độc lập nhau, và một bánh thụ động.

## Máy tính nhúng

***Yêu cầu:***

* Máy tính chạy được Ubuntu 20.04.
* Cấu hình đảm bảo cho giải thuật SLAM và navigation.
* Có hỗ trợ GPIOs.
* Công suất tương đối nhỏ để có thể hoạt động với nguồn pin.

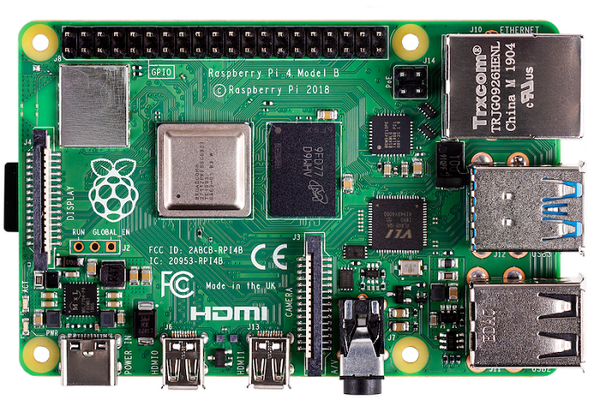
***Phân tích:***

ROS được lập trình để hoạt động trên nền tảng Ubuntu. Mỗi phiên bản ROS được phát hành riêng cho từng phiên bản Ubuntu LTS (Long Time Supported). Vì vậy, máy tính nhúng phải được hỗ trợ bởi một phiên bản Ubuntu LTS. Quan trọng hơn, đề tài cần một máy tính nhúng phải cân bẵng giữa hai tiêu chí đối lập nhau là cấu hình đủ mạnh và công suất tiêu thụ vừa phải.

Raspberry Pi 4 Model B phiên bản 4GB RAM dường như thoả được các yêu cầu trên với cấu hình củ thể như sau:

|  |  |
| --- | --- |
| Processor | Broadcom BCM2711, quad-core Cortex-A72 (ARM v8)  64-bit SoC @ 1.5GH |
| Memory | 4GB LPDDR4 |
| Connectivity | 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless  LAN, Bluetooth 5.0, BLE  Gigabit Ethernet  2 × USB 3.0 ports  2 × USB 2.0 ports. |
| GPIO | 40 pins |
| Input Power | 5V DC via USB-C connector (minimum 3A1)  5V DC via GPIO header (minimum 3A1)  Power over Ethernet (PoE)–enabled |
| Power Consumption | Maximum 15W |
| Size | 85 mm x 56 mm x 17 mm |

Hình 3‑4 Thông số kỹ thuật của Raspberry Pi 4 Model B (4GB RAM).



Hình 3‑5 Board Raspberry Pi 4 Model B.

## RGB-D camera.

***Yêu cầu:***

* Công suất tiêu thụ không quá cao phù hợp sử dụng nguồn pin.
* Khoảng cách tối đa của cảm biến chiều sâu phải đạt trên 2m.

***Thiết kế:***

Đề tài chọn camera Kinect V1. Kinect V1 là camera RGB-D được sử dụng tương đối phổ biến cho nhiều nghiên cứu và ứng dụng. Những giải thuật Visual SLAM cho RGB-D camera thường được thử nghiệm với dòng RGB-D camera Kinect.



Hình 3‑6 Kinect V1.

Thông số Kinect V1:

|  |  |
| --- | --- |
| Depth Sensing Technology | Triangulation Time of flight with structured light |
| Depth Sensing Resolution | 640x480 30fps  320x240 30fps  80x60 30fps |
| Color Image Resolution | 640x480 30fps  1280x960 12fps |
| Field of View | 43◦ vertical  57◦ horizonta |
| Depth Sensing Resolution | 0.4m - 3m (near mode)  0.8m - 4m (default mode) |
| Input Voltage | 12V |
| Power Consumption | < 12W |
| Connection | USB 2.0 |

Bảng 3‑1 Thông số kỹ thuật Kinect V1.

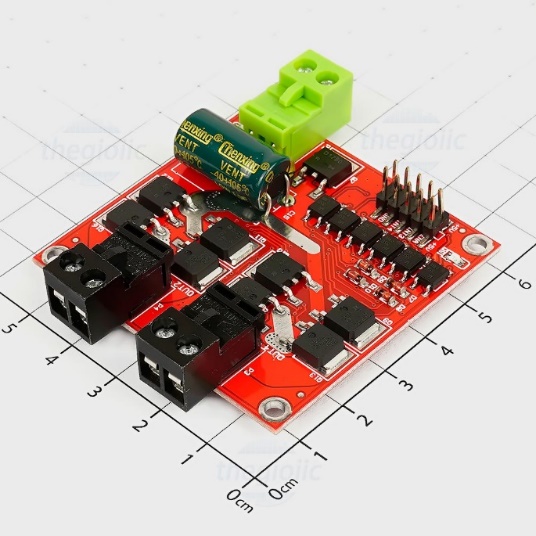
## Mạch điều khiển động cơ.

***Yêu cầu:***

* Mạch có điện áp hoạt động rộng.
* Mạch điều khiển hai động cơ công suất lớn hơn 9W mỗi động cơ.
* Mạch có cách ly phần điều khiển và động lực.

***Thiết kế:***

Đề tài chọn mạch điều khiển động cơ XY-160D. XY-160D có 2 kênh điều khiển động cơ với dòng tối đa 7A mỗi kênh, khoảng điện áp hoạt động rộng (6.5V – 27V) và có cách ly quang cho từng tín hiệu điều khiển.



Bảng 3‑2 Mạch điều khiển động cơ XY-160D.

Thông số kỹ thuật cảu XY-160D theo bảng sau:

|  |  |
| --- | --- |
| Nguồn | 7 – 24 VDC |
| Điện áp tín hiệu điều khiển | Logic high: 3 – 6.5 VDC  Logic low: 0 – 0.8 VDC |
| Số kênh điều khiển | 2 |
| Dòng liên tục tối đa mỗi kênh | 7A |
| Dòng đỉnh (không liên tục) | 25A |
| Tần số PWM tối đa | 10 KHZ |
| Kích thước (LxWxH) | 55mm x 55mm x 13mm |

Bảng 3‑3 Thông số kỹ thuật của mạch điều khiển động cơ.

## MCU

***Yêu cầu:***

* MCU có ít nhất 40 GPIO.
* Hỗ trợ DMA (Direct Memory Access).
* Có ít nhất 4 timer.
* Timer hỗ trợ mode xuất xung PWM.
* Timer hỗ trợ mode đọc encoder QEI (Quadrature Encoder Interface).

***Thiết kế:***

Dòng MCU STM32F103 của hãng STMicroelectronics thoả các yêu cầu trên. Cụ thể, phiên bản STM32F103C8 trên kit STM32 Blue Pill được đề tài sử dụng có thông số như sau:

|  |  |
| --- | --- |
| CPU Core | ARM®32-bit Cortex®-M3 |
| Operating voltage | 3.3V |
| Analog inputs | 10 |
| Digital I/O pins | 37 |
| DC source/sink from I/O pins | 6mA |
| Flash memory | 64KB |
| SRAM | 20KB |
| Frequency (clock speed) | 72MHz max. |
| Communication | 2 I2C interfaces (SMBus/PMBus),  2 SPI interfaces (18 Mbit/s),  3 USARTs,  1 CAN interface (2.0B Active),  1 USB 2.0 full-speed. |
| DMA | 7-channel DMA |
| Timer | 4 x 16-bit Timers support PWM/QEI/OC/IC  2 Watchdog Timers  1 Systick 24-bit Timer |

Bảng 3‑4 Thông số kỹ thuật của STM32F103C8.

## Motors

***Yêu cầu:***

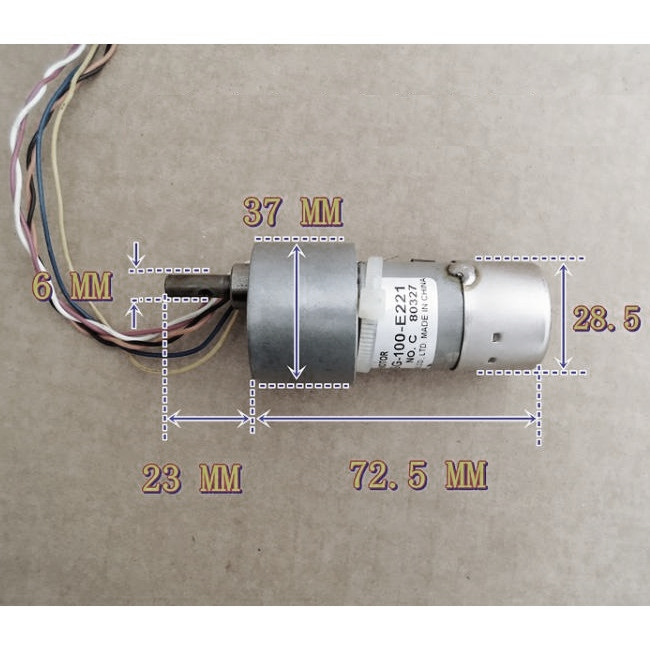
* Motor phù hợp cho điều khiển tốc độ di chuyển tối đa trong khoảng cm/s, phù hợp môi trường trong nhà.
* Số xung encoder phải đủ lớn để tính tốc độ quay trong mỗi vòng lặp lấy mẫu.

***Thiết kế:***

Dựa trên yêu cầu trên đề tài chọn dùng hai động cơ giảm tốc Tsukasa TG-47C-SG-100 đã kèm encoder. Thông số mỗi động cơ theo bảng sau:

|  |  |
| --- | --- |
| Kích thước motor (Φ mm) | 27.7 |
| Kích thước giảm tốc (Φ mm) | 37 |
| Tỉ số truyền | 100:1 |
| Tốc độ không tải (rpm) | 123.5 |
| Điện áp định mức (V) | 24 |
| Dòng định mức (mA) | 355 |
| Dòng không tải (mA) | 90 |
| Momen xoắn định mức (mN/m) | 294 |
| Trọng lượng (g) | 164 |
|  |  |

Bảng 3‑5 Thông số kỹ thuật của động cơ.



Hình 3‑7 Hình ảnh và kích thước thực của động cơ.

Động cơ đi kèm encoder có thông số:

|  |  |
| --- | --- |
| Điện áp hoạt động (V) | 3 – 5 |
| Số xung/vòng mỗi pha chưa qua giảm tốc | 4 |
| Số xung/vòng mỗi pha đã qua giảm tốc | 400 |

Bảng 3‑6 Thông số kỹ thuật của encoder.

Như vậy, ta thu được 400 xung/vòng với mỗi pha encoder. Kết hợp hai pha A và B, ta có khoảng 800 xung/xòng, tương đối lớn để đảm bảo tốc độ lấy mẫu tốc độ ở tần số cao.

## Nguồn

***Yêu cầu:***

* Nguồn có điện áp trên 12V.
* Thời lượng sử dụng trên 30 phút.
* Nguồn cần cung cấp đủ công suất cho các khối:

|  |  |
| --- | --- |
| **Phần cứng** | **Công suất tiêu thụ (maximum) (W)** |
| Raspberry Pi 4 | 15 |
| Kinect | 12 |
| Động cơ (2x) | 17.4 |
| MCU và các thành phần khác | 5 |
| **Tổng** | **49** |
|  |  |

Bảng 3‑7 Bảng liệt kê công suất tiêu thụ bởi phần cứng.

***Thiết kế:***

Theo yêu cầu, hệ thống cần nguồn có công suất tối thiểu 49W. Đề tài chọn pin li-ion có điện áp 3.7V 4.2V, dung lượng danh định 2700mAh, dòng xả tối đa 10C (27A).

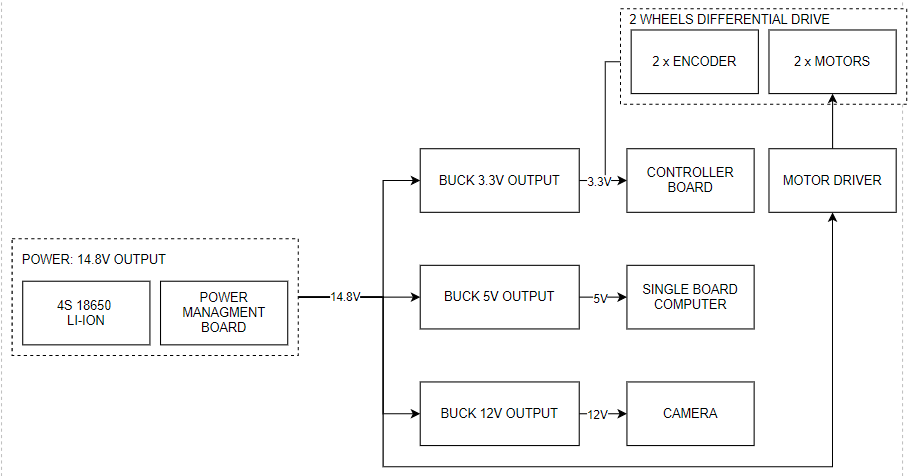
Nguồn bao gồm 4 pin mắc nối tiếp có khoảng điện áp 14.8 16.8 Volt, dung lượng 2700 mAh. Vậy, công suất tức thời cực đại mà nguồn có thể cung cấp là:

vượt xa công suất tối thiểu cho hệ thống.

Với công suất tiêu thụ tối đa là 49W ở điện áp 14.8V, dòng xả tối đa của nguồn khi có tải là:

Căn cứ vào dung lượng pin 2700 mAh và dòng xả như trên, dự kiến hệ thống hoạt động được ít nhất 0.8 giờ, tương đương khoảng 45 phút.

Nguồn đi kèm mạch bảo vệ pin chống quá dòng, ngắn mạch, bảo vệ cạn pin. Sơ đồ phân phối nguồn đến các thành phần theo sơ đồ sau:



Hình 3‑8 Sơ đồ phân phối nguồn đến các khối phần cứng.

# THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

## Phần mềm trên Pi 4

## TX Frame Parser trên Pi 4

Đề tài sử dụng UART kết nối base control và ROS thông qua cơ chế Rosserial. Phía ROS, đề tài viết một node đăng ký topic /cmd\_vel để nhận lệnh điều khiển theo tốc độ dài linear\_vel và tốc độ góc angular\_vel. Hai giá trị này dạng double, 8 bytes. Vậy nên để có thể truyền qua UART, bắt buộc phải chia nhỏ thành các bytes. Đề tài thiết kế một frame truyền dữ liệu như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TX Frame 1 (24 bytes)** | | | | |
| **Header** | **Data (big-endian)** | | **Checksum** | **End frame** |
| 0x01 & 0x02 (2 bytes) | Linear\_vel (8 bytes) | Angular\_vel (8 bytes) | Sum of frame (4 bytes) | CR and LF (2 bytes) |

Bảng 4‑1 Cấu trúc frame UART loại 1.

Node chuyển đổi hai số double là linear\_vel và angular\_vel thành 4 byte riêng lẻ và đẩy vào frame theo thứ tự big-edian. Frame được gửi qua UART đến STM32. STM32 phân giải frame và phục hồi hai giá trị double ban đầu.

## TX frame parser trên STM32

Tương tự vậy, tốc độ dài và tốc độ góc đo được từ encoder cũng được STM32 gửi ngược lại node thông qua UART. Node tạo message theo cấu trúc geometry\_msgs/TranformStamped[] (Bảng 4-2) mang gía trị và . Message được node publish lên topic /tf. Các message này giúp TF tính toán vị trí tương đối của robot trong khung toạ độ /odom. Ta có thể xem vị trí này một cách trực quan thông qua phần mềm RVIZ.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TX Frame 2 (32 bytes)** | | | | | |
| **Header** | **Data (big-endian)** | | | **Checksum** | **End frame** |
| 0x01 & 0x02 (2 bytes) | Linear\_vel (8 bytes) | Angular\_vel (8 bytes) | Dt  (8 bytes) | Sum of frame  (4 bytes) | CR and LF (2 bytes) |

Bảng ‑ Cấu trúc frame UART loại 2.

## Phần mềm trên STM32

STM32 có vai trò thực hiện thuật toán PI controller điều khiển tốc độ động cơ theo tốc độ nhận được từ ROS.

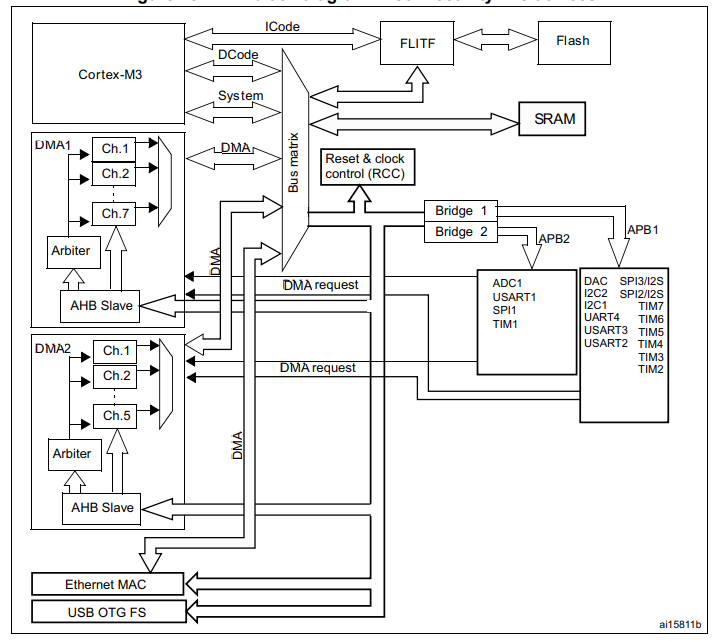
***GPIO***

4 GPIO Pin được cấu hình output nhằm điều khiển chiều quay động cơ:

* Động cơ phải: PB6 và PB7.
* Động cơ trái: PB4 và PB5.

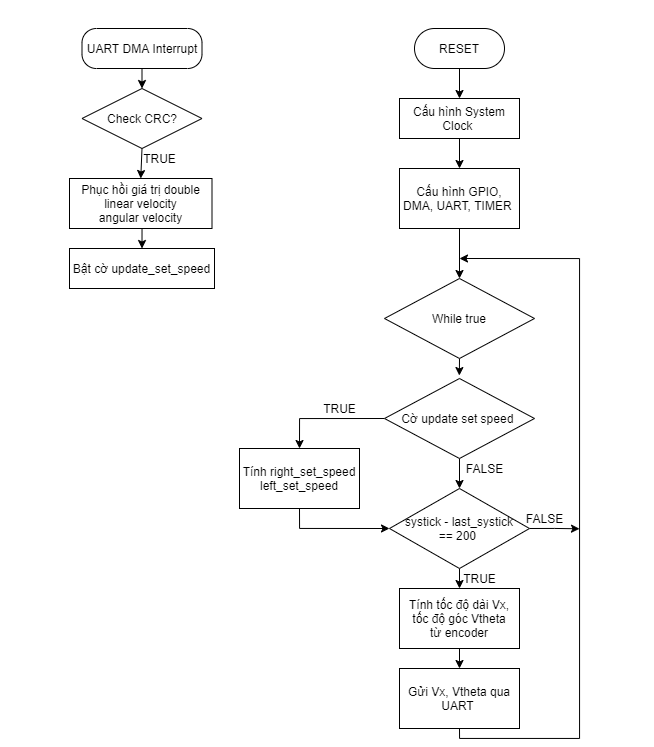
***UART***

UART của STM32F103 được DMA hỗ trợ. DMA, điều khiển bời DMA Controller, chuyển dữ liệu từ ngoại vi sang ngoại vi, ngoại vi sang bộ nhớ mà không cần thông qua CPU. Nhờ DMA dữ liệu có thể đi thẳng buffer của UART tới SRAM mà không tốn thời gian CPU.



Hình 4‑1 Hai khối DMA truy cập trực tiếp tới bus bộ nhớ (SRAM, FLASH và FSMC) và bus ngoại vi (APB & AHB).

UART ở chế độ DMA được cấu hình ngắt theo kích thước buffer. Mỗi khi UART nhận được 3 byte SYN liên tiếp, nó sẽ bật ngắt cho buffer kích thước 16 byte, bằng kích thước 1 frame. Quá trình xử lý ngắt như sau:



Hình 4‑2 Lưu đồ hàm main() và hàm xử lý ngắt UART.

Bằng hai công thức (19) và (20) (Phần 4.1.4.2a ), ta tính được tốc độ của bánh xe bên trái và bên phải (m/s) từ tốc độ dài linear\_vel và tốc độ góc angular\_vel. Thông qua chuyển đổi m/s rpm, ta thu được right\_set\_speed và left\_set\_speed là hai biến giá trị đặt của PI controller.

Ngoài ra, cứ 200 ms dựa trên systick timer, hệ thống gửi lại ROS tốc độ dài Vx và tốc độ góc Vtheta đã đo được. Hai tốc độ này được publish trên topic /tf phục vụ SLAM và navigation.

***TIMER***

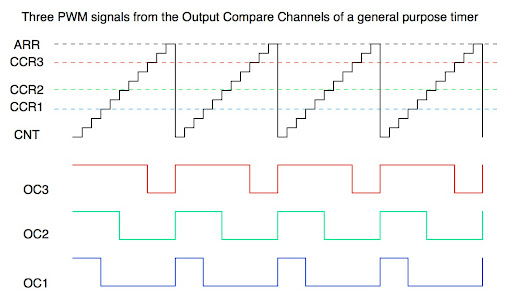
PI controller hoạt động dựa trên Timer 1 16-bit với thời gian ngắt là 10ms. Timer 1 là timer thuộc bus APB1 có clock là 74 MHz. Timer 1 với chế độ đếm lên theo giá trị phụ thuộc vào hai thanh ghi TIM1\_PSC (timer 1 prescaler) và TIM1\_ARR (timer 1 auto-reload). Cụ thể để timer ngắt mỗi 10ms, ta cần nạp giá trị cho TIM1\_PSC và TIM1\_ARR sao cho:

Chọn giá trị 7199 cho TIM1\_PSC và 99 cho TIM1\_ARR.

***PWM***

Với PWM, STM32F103 dùng Timer để xuất xung PWM. Mỗi timer của STM32 có 4 channel hoạt động độc lập. Mỗi channel có giá trị so sánh riêng được cấu hình qua thanh ghi TIMx\_CCRx (capture and compare register x). Đề tài chọn Timer 4 cho chức năng PWM. Channel 3 (cho motor trái) và channel 4 (cho motor phải) được cấu hình xuất xung PWM ra GPIO tương ứng là PB8 và PB9. PWM duty cycle có thể được thay đổi bằng cách ghi giá trị thanh ghi TIM4\_CCR3 cho channel 3 và TIM4\_CCR4 cho channel 4.

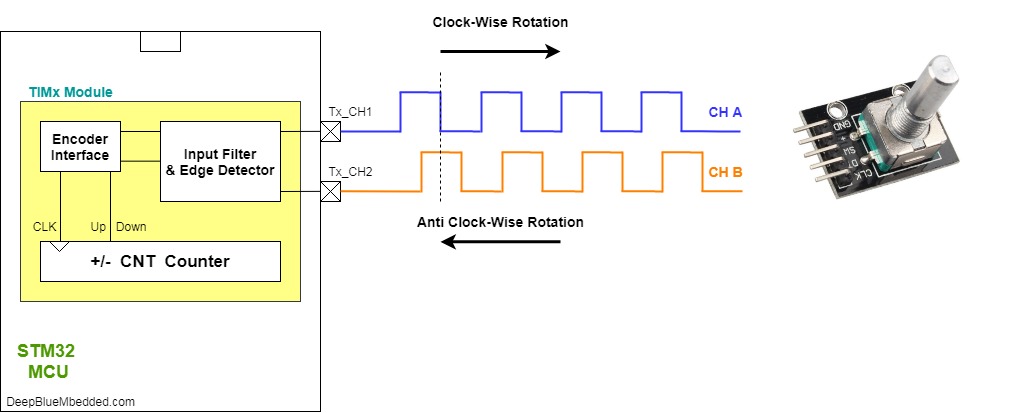
Đề tài chọn tần số PWM là 10KHz. Tần số này được cấu hình thông qua thanh hai thanh ghi TIM1\_PSC (timer 4 prescaler) và TIM1\_ARR (timer 4 auto-reload). Cách tính tương tự như đã trình bày cho Timer 1. Với tần số 10 KHz, chọn TIM1\_PSC = 71, TIM\_ARR = 99.



Hình 4‑3 Minh hoạ chế độ PWM của STM32 timer.

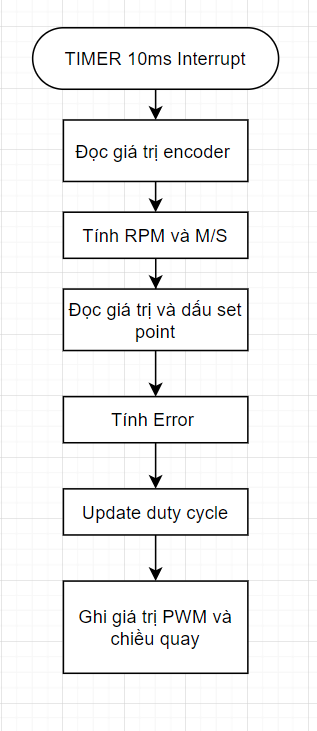
***Encoder***

Với Encoder, đề tài tận dụng chức năng Encoder Mode của timer 2 (cho động cơ bên trái) và timer 3 (cho động cơ bên phải). Với mode này, mỗi timer dùng 2 channel để đọc xung từ hai pha encoder và tăng/giảm giá trị thanh ghi TIMx\_CNT.



Hình 4‑4 Timer ở Encoder Mode đọc xung encoder.

***PI Controller***



Hình 4‑5 Lưu đồ PI controller

Từ các phần cứng đã cấu hình, PI controller được thực hiện dựa trên lưu đồ Hình 4-34.

|  |  |
| --- | --- |
|  |  |
| 1  2  3  4  6 | **PID\_PreProcess**(&str\_right\_motor, right\_set\_speed);  **PID\_PreProcess**(&str\_left\_motor, left\_set\_speed);  **PID\_ComputeOutput**(&str\_right\_motor);  **PID\_ComputeOutput**(&str\_left\_motor);  **PID\_SetDuty**(&str\_right\_motor );  **PID\_SetDuty**(&str\_left\_motor ); | |

Trong đó

* **PID\_Preprocess**() là hàm đọc giá trị encoder, tính tốc độ rpm, chuyển đổi đơn vị m/s và tính sai số đầu vào cho PI controller.
* **PID\_ComputeOutput**() là hàm tính PWM output dựa trên thuật toán PI (công thức (23) và (24)), có bước anti-windup.
* **PID\_SetDuty**() là hàm ghi giá trị PWM output lên thanh ghi TIM4\_CCR3 và TIM4\_CCR4. Đồng thời hàm này cũng thay đổi chiều quay mỗi động cơ qua 4 GPIO Pin điều khiển chiều quay động cơ.
* str\_right\_motor và str\_left\_motor là hai struct chứa tất cả thông tin về GPIO, Timer, duty cycle và thông số PID của từng động cơ.

# KẾT QUẢ THỰC HIỆN

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Sinh viên tóm tắt những điều rút ra được từ kết quả đề tài, những kinh nghiệm có được sau khi thực hiện đề tài. **Ưu và khuyết điểm** của kết quả nghiên cứu đề tài cũng được trình bày trong mục này. Sinh viên cần so sánh với mục tiêu đặt ra trong chương 1.

## Hướng phát triển

Sinh viên trình bày hướng phát triển và khả năng ứng dụng của đề tài

# TÀI LIỆU THAM KHẢO

Trong mục này, sinh viên liệt kê những tài liệu đã tham khảo khi thực hiện đề tài luận văn. Những nội dung trình bày ở mục trên có tham khảo tài liệu thì sinh viên cần ghi chú bằng chỉ số (ví dụ [1], [2]). Chỉ số này cần tương ứng danh mục tài liệu tham khảo. Sinh viên xem thêm hướng dẫn cách viết trích dẫn kiểu IEEE.

Ví dụ:

1. Tống Văn On, “Thiết kế mạch số với VHDL & Verilog”, Nhà xuất bản Lao động Xã Hội, 2007.
2. Altera Corp., “SDRAM Controller for Altera’s DE2/ DE1 boards”, [www.altera.com](http://www.altera.com)

# PHỤ LỤC

Trong phần này, sinh viên có thể trình bày:

* Những kết quả nghiên cứu bổ sung mà trong phần Kết quả luận văn chưa trình bày hết.
* Phần mã nguồn chương trình, sinh viên cũng có thể trình bày trong mục này. Để ngắn gọn, sinh viên chỉ đưa những mã nguồn chính vào phần Phụ lục.
* Sơ đồ toàn mạch chi tiết