**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

**BỘ MÔN ĐIỆN TỬ-VIỄN THÔNG**

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

****

**ĐỀ TÀI NGHIÊN CỨU KHOA HỌC**

**TELEPRESENCE ROBOT**

**GVHD: HÀ HOÀNG KHA**

**SVTH: TRẦN MINH LỢI**

**MSSV: 1412163**

**SVTH: HỒ VĂN BÔN**

**MSSV: 1410311**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2017**

**LỜI CẢM ƠN**

Nhóm đề tài chân thành gửi lời cảm ơn đến thầy Hà Hoàng Kha đã tận tình hướng dẫn, giúp đỡ về các kiến thức chuyên ngành trong suốt quá trình chúng em thực hiện đồ án.

Chúng em cũng thật sự biết ơn những thầy cô giáo trong khoa Điện – Điện Tử, đặc biệt là những thầy cô trong bộ môn Viễn Thông đã giảng dạy chu đáo, cung cấp những kiến thức nền tảng và chuyên sâu góp phần quan trọng để chúng em có thể hoàn thành đồ án.

Sau cùng nhóm cũng xin tri ân đến gia đình đã luôn là động lực để mỗi thành viên vượt qua những khó khăn thử thách trong suốt quá trình học tập và rèn luyện, cảm ơn các bạn đồng chuyên ngành đã nhiệt tình đóng góp ý kiến để đề tài được hoàn thiện.

*Tp. Hồ Chí Minh, ngày 20 tháng 6 năm 2017.*

**Sinh viên**

**Trần Minh Lợi**

**Hồ Văn Bôn**

**TÓM TẮT ĐỀ TÀI**

**DANH SÁCH TỪ VIẾT TẮT**

MỤC LỤC

[I. GIỚI THIỆU 1](#_Toc499062668)

[1.1 Tổng quan về Telepresence Robot: 1](#_Toc499062669)

[1.2 Tình hình nghiên cứu trong và ngoài nước 1](#_Toc499062670)

[**1.2.1** **Tình hình nghiên cứu trong nước** 1](#_Toc499062671)

[**1.2.2** **Tình hình nghiên cứu ngoài nước** 1](#_Toc499062672)

[1.3 Mục tiêu đề tài và phương pháp thực hiện 3](#_Toc499062673)

[**1.3.1** **Mục tiêu đề tài:** 3](#_Toc499062674)

[**1.3.2** **Phương pháp thực hiện** 4](#_Toc499062675)

[II. WebRTC 4](#_Toc499062676)

[2.1 Giới thiệu WebRTC: 4](#_Toc499062677)

[2.2 Xử lý luồng Media ở local peer: 5](#_Toc499062678)

[2.3 Báo hiệu thiết lập kênh truyền với remote peer 6](#_Toc499062679)

[2.4 Kênh truyền dữ liệu 9](#_Toc499062680)

[III. THIẾT KẾ HỆ THỐNG NHÚNG 9](#_Toc499062681)

[3.1 Thiết kế phần cứng của hệ thống 9](#_Toc499062682)

[3.1.1 Thiết kế phần cơ khí 10](#_Toc499062683)

[3.1.2 Lựa chọn động cơ DC, cầu H và nguồn điện cấp 11](#_Toc499062684)

[3.1.3 Cảm biến IMU 16](#_Toc499062685)

[3.1.4 Kit phát triển TM4C123GXL 18](#_Toc499062686)

[3.1.5 Kết nối các ngoại vi với vi điều khiển 20](#_Toc499062687)

[3.2 Lập trình giao tiếp các module ngoại vi với vi điều khiển 20](#_Toc499062688)

[3.2.1 Giới thiệu về thư viện Tivaware 20](#_Toc499062689)

[3.2.2 Cấu hình xung clock TM4C123GH6PM 21](#_Toc499062690)

[3.2.3 Đọc Encoder dùng module QEI 23](#_Toc499062691)

[3.2.4 Điều khiển động cơ DC sử dụng thuật toán PID 26](#_Toc499062692)

[3.2.5 Lập trình giao tiếp MPU6050 28](#_Toc499062693)

[IV. KẾT QUẢ 28](#_Toc499062694)

[V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 28](#_Toc499062695)

[5.1 Kết luận 28](#_Toc499062696)

[5.2 Hướng phát triển 28](#_Toc499062697)

[VI. TÀI LIỆU THAM KHẢO 28](#_Toc499062698)

[VII. PHỤ LỤC 28](#_Toc499062699)

# 

DANH SÁCH HÌNH MINH HỌA

[Hình 1- 1 So sánh SCF và DSCF 3](#_Toc485723563)

[Hình 1- 2 Các bước nhận dạng vân tay 6](file:///C:\Users\Windows%2010%20TIMT\OneDrive\Do%20an\Do_An_Official_May16.docx#_Toc485723564)

[Hình 3- 1 Quá trình phân đoạn 9](#_Toc485723575)

[Hình 3- 2 Ảnh xám 10](file:///C:\Users\Windows%2010%20TIMT\OneDrive\Do%20an\Do_An_Official_May16.docx#_Toc485723576)

[Hình 3- 3 Histogram ảnh khi chuẩn hóa 12](#_Toc485723577)

[Hình 3- 4 Vân tay sau khi được chuẩn hóa 12](#_Toc485723578)

[Hình 3- 5 Vân tay tăng cường bằng hàm lũy thừa 17](#_Toc485723579)

[Hình 4- 1 Ước lượng hướng vân tay bằng toán tử gradient 18](#_Toc485723594)

[Hình 4- 2 Ảnh vân tay một chiều 21](#_Toc485723595)

[Hình 4- 3 Ảnh vân tay một chiều đã làm đầy 21](#_Toc485723596)

[Hình 4- 4 Bộ lọc Gabor theo các hướng từ 3o – 180o 23](#_Toc485723597)

[Hình 4- 5 Ảnh vân tay sau khi lọc Gabor 24](#_Toc485723598)

[Hình 5- 1 Các điểm đặc trưng của vân tay 26](#_Toc485723613)

[Hình 5- 2 Sơ đồ quá trình trích đặc trưng 26](file:///C:\Users\Windows%2010%20TIMT\OneDrive\Do%20an\Do_An_Official_May16.docx#_Toc485723614)

[Hình 5- 3 Ảnh vân tay đã được làm mỏng 27](file:///C:\Users\Windows%2010%20TIMT\OneDrive\Do%20an\Do_An_Official_May16.docx#_Toc485723615)

[Hình 5- 4 Ảnh minutiae 31](#_Toc485723616)

[Hình 7- 1 Kết quả nhận dạng vân tay với SCF 33](#_Toc485723636)

[Hình 7- 2 Kết quả nhận dạng vân tay với DSCF 34](#_Toc485723637)

**DANH SÁCH BẢNG SỐ LIỆU**

[Bảng 1 Kết quả phương pháp của Stein et al. 4](#_Toc485711152)

[Bảng 2 Kết quả theo phương pháp của V.Piuri và F. Scotti 4](#_Toc485711153)

[Bảng 3 Kết quả phương pháp của Hiew et al. 5](#_Toc485711154)

[Bảng 4 Phân chia công việc trong nhóm 6](#_Toc485711155)

[Bảng 5 Kết quả nhận dạng vân tay với SCF 34](#_Toc485711156)

[Bảng 6 Kết quả nhận dạng vân tay với DSCF 34](#_Toc485711157)

# GIỚI THIỆU

## Tổng quan về Telepresence Robot:

Telepresence robots là một loại robot có khả năng thực hiện tác vụ dưới sự điều khiển của người sử dụng từ xa thông qua mạng internet, có khả năng streaming video và hiển thị ra một màn hình giao diện gắn trên robots, từ đó robot trở thành người đại diện cho người dùng khi vắng mặt tại nơi mà robot đang thực thi lệnh từ người dùng.

Hình 1- Một số sản phẩm TELEPRESENCE ROBOT

## Tình hình nghiên cứu trong và ngoài nước

* + 1. **Tình hình nghiên cứu trong nước**

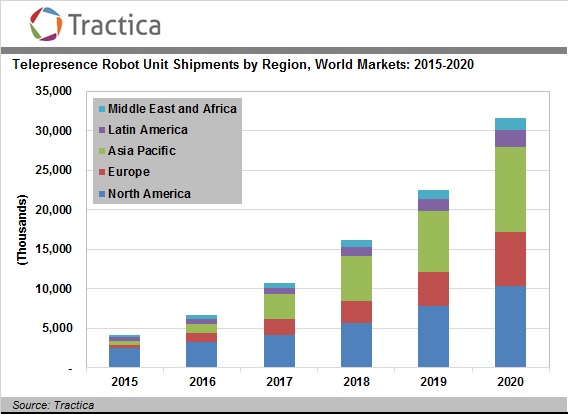
Telepresence robot là một hướng phát triển nghiên cứu còn khá mới mẻ ở Việt Nam, một sản phẩm hứa hẹn sẽ góp phần không nhỏ vào khả năng cải thiện vào chất lượng cuộc sống của người Việt, làm đa dạng hóa các sản phẩm IOT được nghiên cứu chế tạo ở nước ta

* + 1. **Tình hình nghiên cứu ngoài nước**

Telepresence robots bắt đầu được sử dụng trong nhiều lĩnh vực khác nhau:

* Trong quân đội, telepresence robots được sử dụng trong nhiệm vụ có tính rủi ro cao như vô hiệu hóa chất nổ.
* Trong y tế, ngày càng nhiều những ca điều trị từ xa được trợ giúp bởi telepresence robots giúp giảm thiểu thời gian đi lại của bệnh nhân, đảm bảo việc điều trị luôn kịp thời.
* Trong giáo dục, học sinh vẫn có thể theo dõi bài học, tham gia thảo luận nhóm trong khi đang nghỉ dưỡng bệnh ở bệnh viện.
  + Trong quản lý căn hộ, telepresence robots hỗ trợ quản lý thiết bị trong căn hộ thông minh, giao tiếp giữa thành viên ở nhà như trẻ nhỏ và người lớn tuổi với các thành viên đang đi làm, tracking theo đối tượng để báo động khi có nguy hiểm xảy ra.

Sau đây là biểu đồ dự đoán về sử dụng telepresence robots theo từng khu vực từ 2015 đến 2020:



Hình 1- Dự đoán thị phần của Telepresence Robot

Có thể thấy telepresence robots hiện tại là một lĩnh vực mới hiện tại với số thiết bị ít và tập trung chủ yếu ở Bắc Mỹ. Nhưng theo dự báo, số lượng telepresence robots sẽ tăng mạnh trong thời gian tới, đặc biệt ở khu vực châu Âu và châu Á - Thái Bình Dương.

## Mục tiêu đề tài và phương pháp thực hiện

### **Mục tiêu đề tài:**

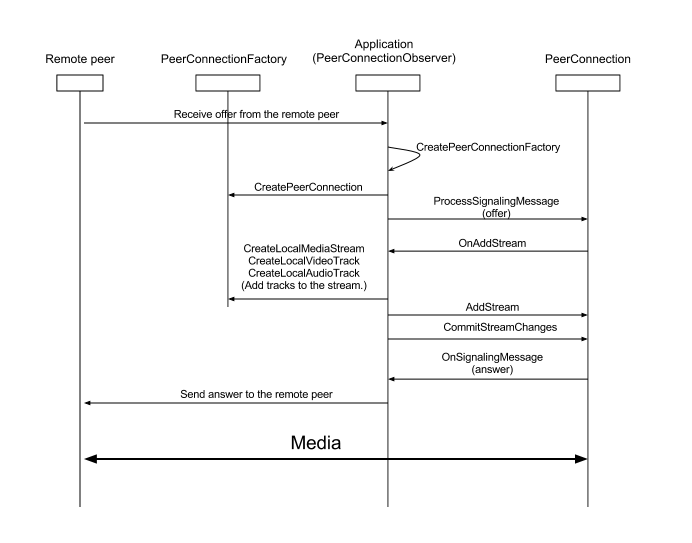
Mục tiêu chính của đề tài là tạo ra sản phẩm đáp ứng như cầu thực tế của người dùng. Telepresence robots ở đây được áp dụng vào lĩnh vực cụ thể là quản lý căn hộ. Các chức năng chính của Robots đã được giới thiệu ở mục “Một số telepresence robots trên thị trường quốc tế”. Tuy nhiên, đề tài nghiên cứu này chỉ tập trung vào một số tính năng cơ bản khi áp dụng telepresence robots vào quản lý căn hộ. Yêu cầu bắt buộc đầu tiên là khả năng tự vận hành và tránh vật cản. Robots di chuyển trong môi trường nhiều đồ nội thất và có mặt của trẻ nhỏ, người lớn tuổi phải đáp ứng được khả năng tránh vật cản. Kinect với hệ thống các cảm biến và camera đảm nhiệm vai trò xác định vật cản và tính toán khoảng cách.. Để giao tiếp từ xa, telepresence robots phải thực hiện chức năng streaming video hoặc audio qua Internet. Đề tài sẽ nghiên cứu những giao thức streaming data qua Internet, một máy tính sẽ đảm nhiệm vai trò này. Ngoài ra, đề tài còn phải nghiên cứu về điều khiển động cơ DC bằng giải thuật PID để điều hướng cho robots. Qua đề tài nghiên cứu, nhóm định hướng tìm hiểu về các giải thuật xử lý ảnh, bên cạnh đó là giao thức streaming dữ liệu và giải thuật điều khiển động cơ DC.

* + 1. **Phương pháp thực hiện**

# WebRTC

## Giới thiệu WebRTC

Webrtc tạo ra một chuẩn mở cho việc giao tiếp peer-to-peer thời gian thực bằng video và audio. WebRTC có thể hoạt động trực tiếp trên trình duyệt web trên các nền tảng khác nhau như Windows, Mac, Android,… Tiến trình thiết lập một cuộc gọi bằng WebRTC trên trình duyệt web được thể hiện ở hình 2-1. Tiến trình kết thúc một cuộc gọi được thể hiện ở hình 2-2.



Có ba bước chính trong thực hiện cuộc gọi bằng WebRTC là: Xử lý luồng media ở local peer, báo hiệu signaling với remote peer và thực hiện stream media.

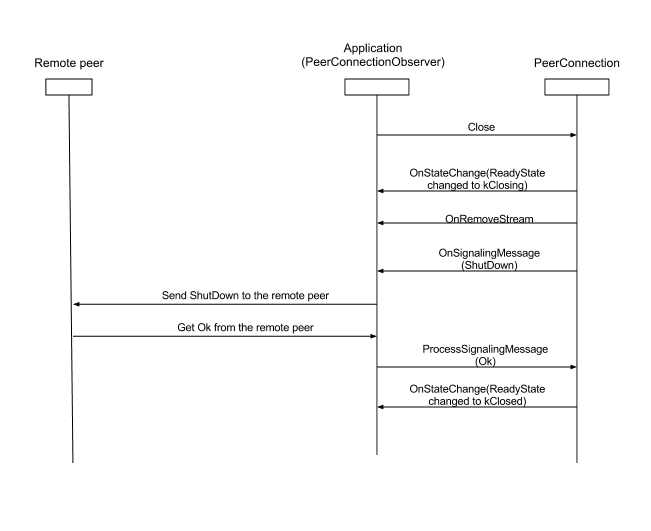
## Xử lý luồng Media ở local peer:

Việc đầu tiên khi thực hiện cuộc gọi bằng WebRTC là truy cập luồng video và audio của thiết bị. Webrtc sử dụng MediaStream API để truy cập vào camera và microphone một cách đồng bộ. Sử dụng phương pháp getVideoTracks() và getAudioTracks() trả về một mảng danh sách các camera và microphone đang được kết nối. Mỗi thiết bị có một ID khác nhau. Sau khi chọn thiết bị, phương thức getUserMedia()trả về luồng media để phát trực tiếp trên local hoặc streaming đến peer khác. WebRTC cũng hộ trợ một số ràng buộc về chất lượng của video như:

* MinWidth: chiều rộng tối thiểu tính theo pixel.

Hình 2- Thiết lập cuộc gọi với WebRTC

Có ba bước chính trong thực hiện cuộc gọi bằng WebRTC là: Xử lý luồng media ở local peer, báo hiệu signaling với remote peer và thực hiện stream media.



Hình 2- Kết thúc cuộc gọi với WebRTC

## 2.2 Xử lý luồng media ở local-peer

Việc đầu tiên khi thực hiện cuộc gọi bằng WebRTC là truy cập luồng video và audio của thiết bị. Webrtc sử dụng MediaStream API để truy cập vào camera và microphone một cách đồng bộ. Sử dụng phương pháp getVideoTracks() và getAudioTracks() trả về một mảng danh sách các camera và microphone đang được kết nối. Mỗi thiết bị có một ID khác nhau. Sau khi chọn thiết bị, phương thức getUserMedia()trả về luồng media để phát trực tiếp trên local hoặc streaming đến peer khác. WebRTC cũng hộ trợ một số ràng buộc về chất lượng của video như:

* MaxWidth: chiều rộng tối đa tính theo pixel.
* MinHeight: chiều cao tối thiểu tính theo pixel.
* MaxHeight: chiều cao tối đa tính theo pixel.
* MinAspectRatio: tỉ lệ màn hình tối thiểu, thường là 1 với hình vuông, 1.7777 với màn hình tỉ lệ 16/9 và 1.3333 với màn hình 4/3.
* MaxAspectRatio: tỉ lệ màn hình tối đa.
* MinFramerate: tần số khung hình tối thiểu (frame/s).
* MaxFramerate: tần số khung hình tối đa (frame/s).

WebRTC quy định một số chuẩn codec cho video và audio trước khi được truyền đi. Dải màu được ưu tiên sử dụng là sRGB. Trình duyệt web phải sử dụng codec VP8 và H.264. Audio của WebRTC sử dụng một số codec như sau:

* Opus.
* PCMA và PCMU.
* Comfort Noise.

## Báo hiệu thiết lập kênh truyền với remote-peer

### **2.3.1 WebSocket**

Trước khi thực hiện streaming media, WebRTC cần thực hiện bắt tay giữa hai peer thông qua các tín hiệu điều khiển hay còn gọi là signaling. WebRTC không ràng buộc cơ chế signaling riêng biệt, vì vậy các cơ chế signaling khác nhau có thể được sử dụng một cách linh hoạt. Một số cơ chế có thể được sử dụng cho signaling là SIP, XMPP hoặc Websocket. Thông tin của các peer được trao đổi thông qua signaling server thay vì nằm ở phía client. Vì vậy thông tin trạng thái của những lần truy cập khác nhau sẽ không được lưu ở trình duyệt web. Các chức năng chính của signaling server là:

* Tạo danh sách các peer đang kết nối tới server.
* Cho phép các peer chọn được peer muốn streaming data.
* Cho phép peer chấp nhận hoặc từ chối kết nối khi được yêu cầu.
* Trao đổi thông tin của các peer với nhau.

Đề tài sử dụng Websocket để thực hiện signaling. Websocket cung cấp full-duplex channel giữa peer va server. Với ứng dụng web chạy trong thời gian thực, cơ chế polling giữa peer và server yêu cầu trình duyệt web phải liên tục refresh để cập nhật thông tin. Để hạn chế việc gửi yêu cầu lên server liên tục, websocket sử dụng cơ chế AJAX long polling thay cho cơ chế polling thông thường. Với long polling, sau khi nhận được yêu cầu từ peer, server sẽ không trả lời ngay lập tức mà giữ kết nối mở trong một khoảng thời gian định sẵn. Trong thời gian mở kết nối, server trả lời cho peer liên tục mà không cần đóng kết nối. Nếu không có dữ liệu cần cập nhật thì sau một khoảng thời gian kết nối sẽ được đóng.

Hình 2- Signaling với WebSocket

**Websocket Server**

**Update Peer List**

**Peer A**

**Ready For Connect**

**Request For**

**Remote Peer**

**Send info**

**Send request**

**Respone Peer list**

**Ready for**

**remote peer call**

**Handle Request**

Theo hình 2-3, các peer khi kết nối với server sẽ gửi kèm thông tin ID của mình. Server sẽ lưu toàn bộ thông tin của các peer đang hoạt động. Khi một peer muốn kết nối với một peer khác sẽ gửi yêu cầu lên server và đợi trả lời chứa thông tin của các peer có thể kết nối. Sau khi có thông tin của peer cần kết nối, peer có thể thực hiện cuộc gọi video.

### **2.3.2 STUN server và TURN server**

STUN server cho phép các thiết bị ở sau NAT có thể biết được IP công cộng và loại NAT mà thiết bị đang kết nối vào. Thông tin này cần thiết để các thiết bị nằm ngoài LAN có thể thiết lập kết nối UDP với nhau. Giao thức của STUN là server-client được quy định ở chuẩn RFC3489.

Hình 2- Giao thức kết nối với STUN server



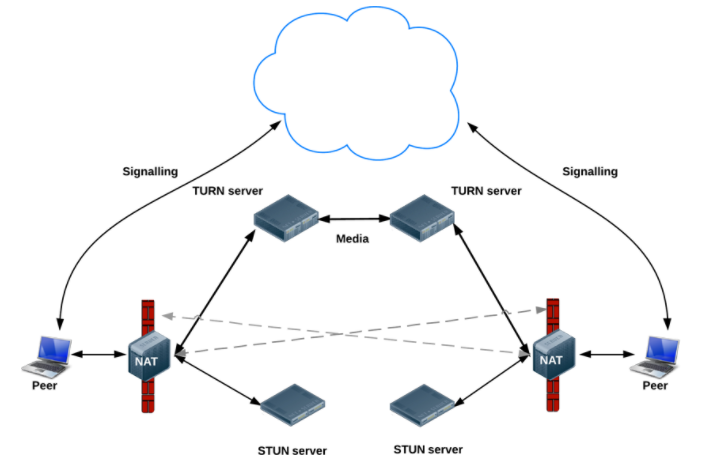
Theo hình 2-4, peer A có IP cá nhân là 192.168.1.3 khi muốn kết nối với STUN server 64.25.58.65 sẽ gửi yêu cầu đến Gateway 192.168.1.1. Tiếp theo Gateway sẽ chuyển tiếp yêu cầu đến STUN server. Cuối cùng STUN server sẽ trả lời cho peer A thông qua Gateway với thông tin IP công cộng là 212.128.56.125. Khi đó peer A biết được IP công cộng của mình. Vì STUN server chỉ cung cấp thông tin cho các thiết bị nên băng thông tiêu tốn giữa thiết bị và server là tương đối nhỏ, các STUN server thường được cung cấp miễn phí. Tuy nhiên trong một số cấu hình NATs thì thông tin cung cấp từ STUN server cũng không giúp các thiết bị trong NATs có thể giao tiếp trực tiếp với nhau bằng UDP được.

Ngoài STUN server, TURN server cũng được sử dụng khi một trong hai thiết bị ở phía sau NATs. Khác với STUN server chỉ cung cấp thông tin IP công cộng để thực hiện truyền dữ liệu point-to-point giữa hai thiết bị, TURN server cho phép chuyển tiếp dữ liệu song công giữa thiết bị và server. Nhờ mô hình client-server mà dữ liệu có thể truyền đến các thiết bị nằm trong NATs mà không bị chặn thông qua sự chuyển tiếp của TURN server. Tuy nhiên, việc sử dụng TURN server là tương đối tốn kém vì băng thông phải được cung cấp bởi server. Mặt khác thời gian delay khi sử chuyển tiếp dữ liệu đặc biệt là video tương đối lớn.

Để tận dụng ưu điểm, khắc phục hạn chế của STUN và TURN server, ICE server sử dụng cả hai cơ chế nào để truyền dữ liệu giữa hai thiết bị trong NATs. ICE server cùng với Signaling server tạo nên mô hình truyền dữ liệu hoàn chỉnh như hình 2-5.

Đầu tiên ICE server cố gắng truyền dữ liệu point-to-point giữa hai thiết bị. Khi 2 thiết bị ở sau NATs, ICE sử dụng STUN server để cung cấp IP công cộng cho các peer. Nếu kết nối thành công, các peer sẽ kết nối point-to-point để truyền media bằng các gói UDP. Cuối cùng nếu vẫn không thành công, dữ liệu chuyển từ point-to-point sang client –server thông qua TURN server.

Hình 2- ICE server và Signaling server



## Kênh truyền dữ liệu

# THIẾT KẾ HỆ THỐNG NHÚNG

## Thiết kế phần cứng hệ thống

Telepresence robot yêu cầu hệ thống có khung cơ khí, các ngoại vi và bo điều khiển đáp ứng được yêu cầu xử lý về stream camera, tự hành tránh vật cản trong nhà, điều khiển tự hành từ xa. Từ những yêu cầu đặt ra trên, các thành phần phần cứng được sử dụng gồm: khung xe, camera kinect, vi điều khiển giao tiếp với ngoại vi và máy tính, động cơ DC bao gồm encoder, cầu H để điều khiển động cơ, bo mạch kết nối các ngoại vi với vi điều khiển, cảm biến IMU đo góc quay của robot, nguồn điện cấp cho hệ thống, máy tính cài đặt hệ thống ROS trên nền tảng hệ điều hành Linux.

Vi điều khiển

PC

Motor 1

Motor2

Cầu H

Mạch đệm

Encoder1

Encoder 2

IMU

Kinect

Nguồn cấp

Hình 3- Sơ đồ khối của hệ thống

### **Thiết kế phần cơ khí**

Với chức năng chính của robot là tự hành tránh vật cản trong nhà phần cứng được thiết kế với hình dạng khung xe rùa. Với hình dạng này robot có thể dựa vào bán kính khung xe và các khoảng cách đến những vật cản xung quanh, sau đó máy tính thực hiện thuật toán tránh vật cản để đưa ra quyết định di chuyển cho robot. Trọng tải chính của robot gồm các bộ phận: nguồn acquy 12V-9Ah trọng lượng 2.78 kg, Laptop ASUS trọng lượng 1.5 kg, Kinect Xbox 360 trọng lượng gồm cả adapter là 1.5 kg, vậy tổng trọng tải đặt lên khung xe lên đến 5.78 kg. Từ đó vật liệu được dùng để làm khung robot được lựa chọn là mica 5ml, đặc trưng của vật liệu này là có tỉ trọng chỉ bằng ½ so với thủy tinh, chịu lực và chịu nhiệt tốt, hạn chế tác động của các hóa chất như xăng dầu, bền về mặt thời gian, chi phí vật liệu rẻ hơn so với các vật liệu như khung nhôm hay vật liệu in 3D.

Bản vẽ khung robot ….………………………………………………..

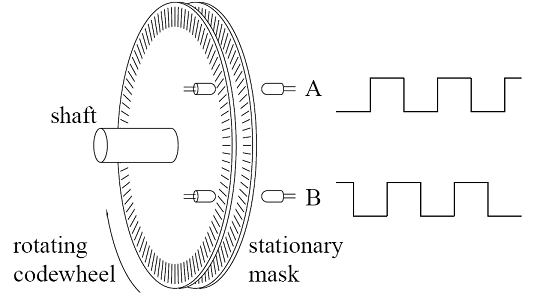
Kích thước của khung robot: Đường kính khung robot, chiều cao giữa 2 lớp,

### **Lựa chọn động cơ DC, cầu H và nguồn điện cấp**

* Lựa chọn động cơ DC:

Với trọng tải robot lên đến 5.78 kg trong khi không yêu cầu về tốc độ chạy cao, động cơ DC được yêu cầu phải có hộp số giảm tốc để tăng moment xoắn giúp việc kéo tải của động cơ điện được dễ dàng. Ngoài ra để điều khiển vị trí robot dựa vào số vòng quay của bánh xe yêu cầu động cơ cần có bộ phận phản hồi thông tin về số vòng quay mà bánh xe robot, bộ phận có thể đảm nhận chức năng được gọi là encoder.

Hình 3- Encoder tương đối gồm 2 kênh A và B



Nếu chia theo sóng sử dụng trong encoder gồm: encoder quang và encoder từ. Còn chia theo dạng tín hiệu ra của encoder ta có encoder tuyệt đối và encoder tương đối. Trong đó encoder tuyệt đối cho tín hiệu ngõ ra là chuỗi n bit báo động cơ đã quay được vòng quay. Encoder tương đối thì ngõ ra là 2 kênh tín hiệu A và B lệch pha nhau. Số xung trên mỗi vòng quay được chỉ định bởi nhà sản xuất, pha A sớm hay trễ hơn pha B cho ta biết chiều quay của động cơ. Động cơ được chọn là động cơ DC giảm tốc GRB37-3530 kèm encoder tương đối.

Hình 3- Động cơ DC giảm tốc GRB37-3530 kèm encoder



Thông số kĩ thuật động cơ DC và Encoder:

Bảng Động cơ DC giảm tốc GRB37-3530

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tỉ số truyền của hộp số | Điện áp hoạt động (Vdc) | Điện áp trung bình (V) | Tốc độ  cực đại không tải (rpm) | Dòng điện cực đại không tải (mA) | Dòng điện cực đại  có tải (mA) |
| 1:10 | 6 – 24 | 12 | 1000 | 90 | 480 |

Bảng Encoder

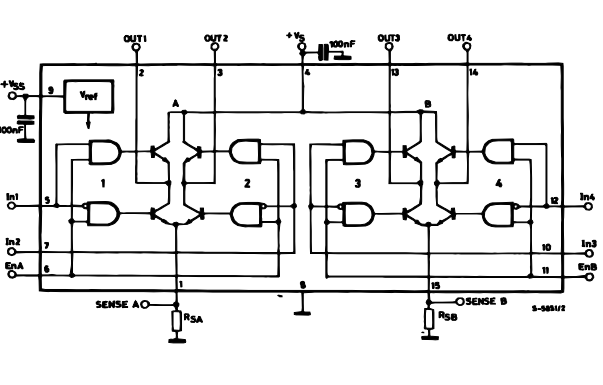
|  |  |  |
| --- | --- | --- |
| Số xung/ vòng  (CPR) | Điện áp  hoạt động  (Vdc) | Loại encoder |
| 2100 | 3.3 – 5 | Encoder  tương đối |

* Lựa chọn mạch cầu H:

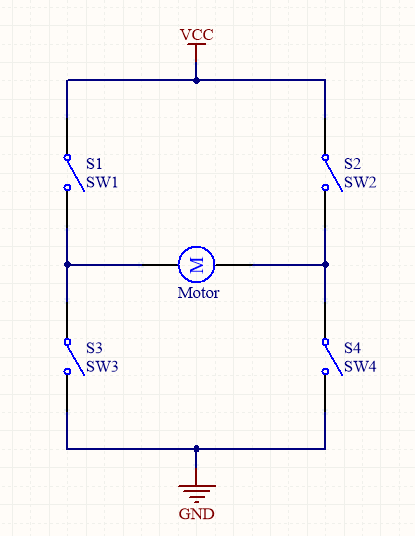
Để điều khiển chiều quay, tốc độ động cơ yêu cầu cần có một mạch điện có khả năng đảo chiều dòng điện cũng như có thể điều khiển cường độ dòng điện chảy qua động cơ, dưới đây là sơ đồ nguyên lý đơn giản của một mạch cầu H:

Thông qua việc điều khiển các khóa S1, S2, S3, S4 ta có thể thay đổi chiều dòng điện qua động cơ. Trên thực tế các khóa điện trên được thay thế bởi các linh kiện điện tử có khả năng đóng ngắt dòng điện như BJT hoặc MOSFET. Hai loại linh kiện trên có ưu và nhược điểm riêng. Điển hình là BJT có khả năng hoạt động với điện áp kích cao nhưng dòng điện ngõ ra thấp do trở kháng giữa 2 cực C-E lớn, đồng thời tần số đóng ngắt nhỏ hơn nhiều so với MOSFET. MOSFET có ưu điểm là trở kháng ngõ vào rất lớn (hàng MΩ), có khả năng chịu dòng tốt nhờ trở kháng Rds nhỏ, độ ổn định về nhiệt cao. Vì vậy MOSFET thường được tích hợp trong IC có tần số đóng ngắt cao, song MOSFET có mức điện áp hoạt động thường nhỏ hơn so với BJT do vậy dễ bị hư hỏng bởi các gai nhiễu. Dựa vào dòng điện có tải của động cơ DC giảm tốc GRB37-3530 , mạch cầu H được lựa chọn là mạch cầu H sử dụng IC L298N.

Hình 3- Sơ đồ khối IC L298N

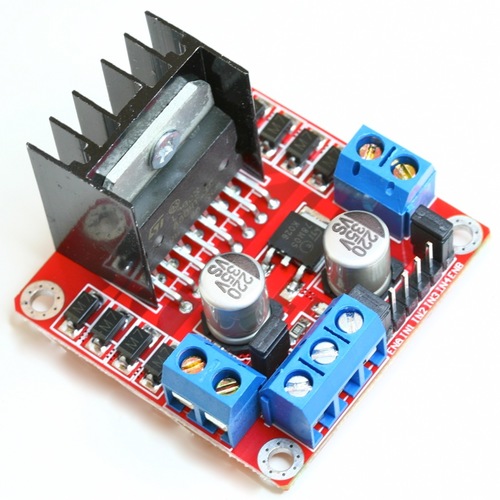


Hình 3- Sơ đồ nguyên lý cầu H



IC L298N được tích hợp 2 mạch cầu H với chuẩn điều khiển TTL, không có diode bảo vệ nội, dòng điện tải cực đại của mỗi cầu lên đến 2A, nguồn cấp lên đến 46V, có bảo vệ quá nhiệt. Module cầu H L298N được thiết kế với thành phần chính là IC L298N, diode bảo vệ cho 2 cầu H bên trong IC, đây là module được dùng rất thông dụng hiện nay.

Hình 3- Module cầu H L298N



* Lựa chọn nguồn điện cấp cho robot:

Để robot có thể tự hành, di chuyển một cách linh hoạt trong nhà cần một bộ nguồn điện cấp độc lập cho hệ thống. Với hệ thống này, hai nguồn điện chính để cấp cho hệ thống. Một nguồn được lấy trực tiếp từ pin của laptop ASUS cấp trực tiếp cho hoạt động của máy đồng thời ta có thể lấy nguồn 5V từ cổng USB máy tính cấp cho bo mạch phát triển Tiva TM4C123GXL. Nguồn còn lại được dùng cho việc cấp nguồn cho điều khiển động cơ và Kinect XBOX 360.

Điện áp cấp cho động cơ dao động từ 6 – 24 V với dòng điện tiêu thụ khi có tải là 0.480x2 = 0.96 A. Điện áp cấp cho Kinect là 12V DC với dòng tiêu thụ khi hoạt động là 1.5A. Tổng dòng điện tiêu thụ lớn nhất của 2 thành phần này khi hoạt động là 0.96 + 1.5 = 2.46A.

Acquy đã được lựa chọn để giải quyết vấn đề trên. Acquy là một loại pin có khả năng tái sử dụng nhiều lần bằng cách nạp điện thông qua mạch sạc điện. Acquy được chia thành nhiều loại khác nhau: acquy ướt, acquy kín khí, acquy khô. Mỗi loại acquy có ưu và nhược điểm riêng của mình, trong đó acquy khô là loại acquy có những ưu điểm vượt trội do được thiết kế với công nghệ kín khí chống bốc hơi, có khả năng tái tổ hợp nước nên không cần thêm nước từ bên ngoài. Acquy được sử dụng là dòng acquy Vision 12V-9AH CP1290 với những thông số kỹ thuật như sau:

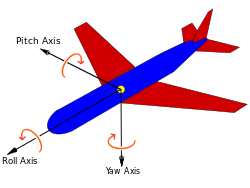
Bảng Thông số Acquy Vision 12V-9AH CP1290

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Hiệu điện thế  (Vdc) | Dung lượng (AH) | Kích thước  (LxWxH)  (mm) | Trọng lượng  (kg) | Số cell | Dòng điện cực đại ở 25°C  (A) | Dòng ngắn mạch  (A) | Tuổi thọ trung bình  (năm) |
| 12 | 9 | 151x65x94 | 2.78 | 6 | 135 | 450 | 7 |

Với việc dùng loại acquy này, ta thấy với dòng tiêu thụ khi hệ thống hoạt động là 2.46 A, ta có thể cho hệ thống hoạt động liên tục trong vòng 9/2.46 = 3.66 giờ, đáp ứng được yêu cầu đặt ra.

### **Cảm biến IMU**

IMU là một thiết bị điện tử kết hợp 2 loại cảm biến là cảm biến gia tốc (Accelerometer) và cảm biến con quay hồi chuyển (Gygroscope). IMU được sản xuất với công nghệ MEMS cho phép xác định góc quay, vận tốc, vị trí của các hệ thống chuyển động trong không gian 3 chiều một cách hiệu quả.



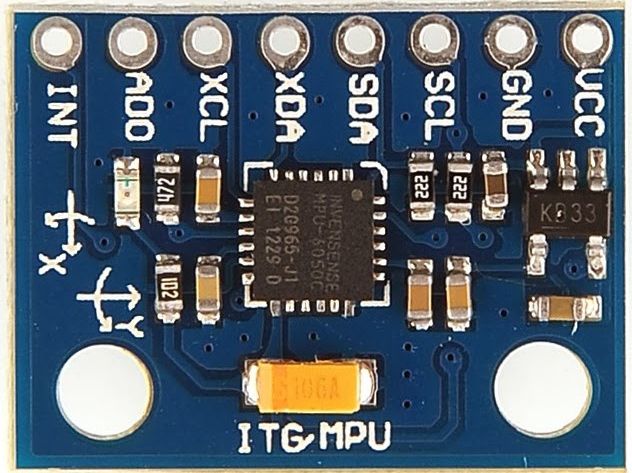
Hình 3- Các chiều trong không gian

Cảm biến con quay hồi chuyển cho phép đo sự biến thiên về vận tốc góc theo các trục khác nhau. Số trục của con quay hồi chuyển cho những ứng dụng phức tạp có thể lên đến 9 trục hoặc hơn, trong đó ba trục phổ biến là Roll, Pitch và Yall như trên hình 3-7. Từ vận tốc góc ta có thể suy ra góc quay của vật thể bằng phương pháp tích phân. Ưu điểm của con quay hồi chuyển là ít ảnh hưởng bởi rung động, tác động của từ trường xung quanh, song khuyết điểm của nó là có giá trị offset khi đứng yên, giá trị này chịu tác động tùy theo chất lượng và nhiệt độ làm việc của cảm biến đồng thời giá trị cũng bị trôi chậm theo thời gian.

Cảm biến gia tốc có khả năng đo độ lớn hình chiếu của gia tốc trọng trường lên hệ trục quán tính của vật thể gắn với cảm biến. Từ giá trị gia tốc đọc về ta có thể suy ra được giá trị vận tốc, góc quay bằng phương pháp tích phân. Điểm yếu của cảm biến này là nhạy với rung động cơ học và cũng có offset như cảm biến con quay hồi chuyển.

Đề tài sử dụng cảm biến MPU6050 tích hợp cảm biến con quay hồi chuyển 3 trục, cảm biến gia tốc 3 trục, 1 chip xử lý DMP (Digital Motion Processor) thu thập và xử lý dữ liệu với các thông số sau:

* Cảm biến con quay hồi chuyển có độ phân giải dữ liệu 16 bit, dải thang đo có thể lập trình được là: ±250, ±500, ±1000, ±2000 °/s (dps)
* Cảm biến gia tốc cũng có độ phân giải 16 bit, dải thang đo có thể lập trình được là: ±2, ±4, ±8, ±16 (g).
* Điện áp hoạt động: 2.375V - 3.46V
* Chuẩn giao tiếp: I2C 400 kHz



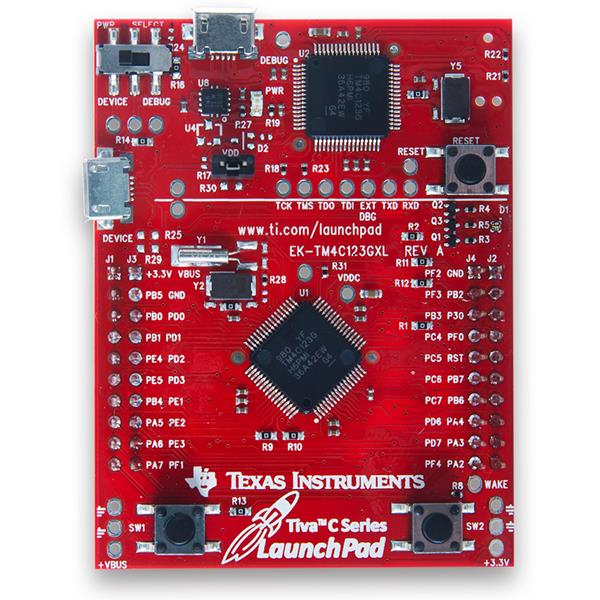
Hình 3- Module cảm biến MPU6050

### **Kit phát triển TM4C123GXL**

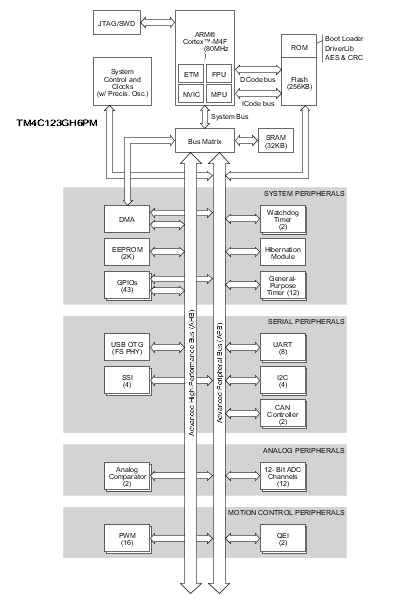
Tiva TM4C123GXL là kit phát triển được tích hợp sẵn vi điều khiển TM4C123GH6PM của hãng Texas Instrument. Đây là dòng vi điều khiển 32 bit -M4 80MHz, hỗ trợ tính toán dấu chấm động với các thông số:

* Tích hợp 256KB bộ nhớ Flash, 32KB SRAM và 2 KB EEPROM
* 8 UARTs (Universal Asynchronous Receivers/Transmitter)
* 4 SSI (Synchronous Serial Interface)
* 4 module I2C hỗ trợ truyền nhận tốc độ cao
* 2 module QEI (Quadrature Encoder Interface)
* 2 module PWM, mỗi module gồm 4 khối tạo PWM và một khối điều khiển, có tổng tổng tất cả 16 ngõ ra PWM
* 2 module ADC 12 bit, tốc độ lấy mẫu lến đến 1 triệu (sample/s)

Hình 3- Kit TM4C123GXL



Hình 3- Sơ đồ khối của TM4C123GH6PM



Hệ thống robot đang sử dụng có các module cần giao tiếp là encoder và IMU MPU6050 thông qua chuẩn I2C. Việc đọc xung encoder tốc độ cao tốn rất nhiều thời gian xử lý nếu sử dụng ngắt. Vi điều khiển TM4C123GH6PM hỗ trợ việc đọc và xử lý dữ liệu encoder thông qua module QEI độc lập với việc xử lý của CPU. Đây là một điểm rất nổi bật và mạnh mẽ của dòng chip này. Ngoài ra TM4C123GH6PM cũng hỗ trợ module giao tiếp I2C truyền nhận tốc độ cao nên hoàn toàn có thể giao tiếp với IMU MPU6050.

### **Kết nối ngoại vi với vi điều khiển**

Hình 3- Sơ đồ khối hệ thống kết nối với vi điều khiển

Mạch đệm (SN74-HC245)

Cầu H L298N

Encoder 1

Encoder 2

MPU6050

Động cơ 1

Động cơ 2

Tiva TM4C123GXL

PD7

PD7

PB6

PB7

PC5

PC6

PE4

PE5

PA6

PA7

Trong đó:

* PD6, PD7: được cấu hình để kết nối đến module QEI0
* PC5, PC6: được cấu hình để kết nối đến module QEI1
* PA6, PA7: được cấu hình để kết nối đén module I2C1(PA6: SCL và PA7: SDA)
* PB6, PB7: ngõ ra của module PWM0 khối tạo xung 0
* PE4, PE5: ngõ ra của module PWM0 khối tạo xung 2

## Lập trình giao tiếp các module ngoại vi với vi điều khiển

### Giới thiệu về thư viện Tivaware

Tivaware là một bộ thư viện được viết bởi Texas Instrument, bộ thư viện cho phép truy cập đến những module ngoại vi của dòng vi điều khiển Tiva một cách dễ dàng

Những điểm tích cực của bộ thư viện này là:

* Thư viện được viết hoàn toàn bằng ngôn ngữ C với cấu trúc hàm dễ hiểu, gợi nhớ
* Thư viện có sách hướng dẫn chi tiết cách sử dụng các ngoại vi ở các chế độ thông dụng
* Thư viện được thiết kế để sử dụng hiệu quả về bộ nhớ và luồng xử lý
* Thư viện được cập nhật thường xuyên để có thể tối ưu hóa việc sử dụng phần cứng của dòng Tiva

Từ những đặc điểm trên, việc lập trình giao tiếp các module ngoại vi của hệ thống dựa trên các hàm của thư viện Tivaware phiên bản 2.1.2.111

### Cấu hình xung clock TM4C123GH6PM

Các nguồn dao động cấp cho TM4C123GH6PM rất đa dạng, gồm có:

* Nguồn dao động nội chính xác (PIOSC): Đây là một nguồn dao động được tích hợp sắn trong chip tạo ra xung clock hệ thống 16MHZ chính xác với sai số chỉ +/- 1%, đáp ứng cho những ứng dụng yêu cầu về định thì
* Nguồn dao động chính: Nguồn dao động được lấy từ nguồn dao động đơn bên ngoài (external single – ended clock) cấp đến chân OSC0 hoặc dùng thạch anh hay nguồn tạo dao động kết nối đến OSC0 và OSC1, khi dùng PLL tần số cấp vào yêu cầu nằm trong khoảng 5Mhz đến 25Mhz, nếu không dùng PLL tần số nguồn dao động là 4Mhz – 25Mhz, độ chính xác tùy thuộc vào chất lượng nguồn dao động cấp vào
* Nguồn dao động nội tần số thấp: Nguồn dao động này được dùng trong chế độ ngủ sâu của vi điều khiển (Deep sleep mode)
* Nguồn dao động ngủ đông: Nguồn dao động này được lấy từ một nguồn dao động đơn 32.768 Khz cấp vào chân XOSC0, dùng cho chế độ ngủ sâu(Deep sleep) hoặc chế độ ngủ đông(Hibernate)(trong chế độ ngủ đông, nguồn cấp sẽ không cấp trực tiếp cho vi điều khiển mà được cấp cho module hibernate để quản lý năng lượng cho hệ thống vi điều khiển)

|  |
| --- |
|  |

Hình: Sơ đồ hệ thống xung clock vi điều khiển TM4C123GH6PM

* Việc cấu hình xung clock được thực hiện thông qua thư viện Tivaware sử dùng lệnh *SysCtlClockSet(uint32\_t ui32Config)*

Với *ui32Config* là đối số truyền vào, tham số này là kết quả OR của một vài đối số khác nhau dưới đây:

+ Giá trị bộ chia tần số dao động: **SYSCTL\_SYSDIV\_1**,  
**SYSCTL\_SYSDIV\_2**, **SYSCTL\_SYSDIV\_3**, ... **SYSCTL\_SYSDIV\_64**

+ Đối số chỉ ra dùng PLL hay không: **SYSCTL\_USE\_PLL** or **SYSCTL\_USE\_OSC** Nếu dùng PLL thì tần số chưa qua bộ chia được đẩy lên 400Mhz, còn không thì tần số bằng tấn số của nguồn cấp dao động

+ Giá trị tần số thạch anh ngoài: **SYSCTL\_XTAL\_4MHZ**, **SYSCTL\_XTAL\_4\_09MHZ**, **SYSCTL\_XTAL\_16MHZ,…**

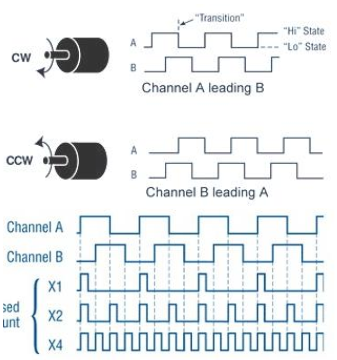
**+** Chọn nguồn dao dộng**: SYSCTL\_OSC\_MAIN**,  
**SYSCTL\_OSC\_INT**, **SYSCTL\_OSC\_INT4**, **SYSCTL\_OSC\_INT30**, or **SYSCTL\_OSC\_EXT32**

Câu lệnh sau được dùng để cấu hình xung clock sử dụng nguồn dao động chính với thạch anh ngoài 16 Mhz được gắn sắn trên bo mạch phát triển TM4C123GXL, không dùng PLL để tạo tần số dao động của hệ thống là 16 MHz

SysCtlClockSet(SYSCTL\_SYSDIV\_1 | SYSCTL\_USE\_OSC | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ)

### Đọc Encoder dùng module QEI

Encoder gồm có 2 pha A và B đặt lệch pha nhau thường là 90 độ, để xác định số vòng đã quay được cũng như là chiều quay hiện tại của động cơ



Khi pha A sớm pha hơn pha B, nếu quy định động cơ đang quay thuận thì trường hợp pha A trễ pha hơn pha B sẽ là quay nghịch

Có 4 chế độ đọc xung encoder là x1, x2 và x4. Chế độ x1 chỉ đọc cạnh lên của pha A, chế độ x2 đọc cạnh lên và cạnh xuống của pha A, chế độ x4 sẽ đọc cả cạnh lên và cạnh xuống của pha và pha B, chế độ x4 sẽ cho độ phân giải cao nhất

Điều kiện xét quay thuận(A sớm pha hơn B): A có cạnh lên trong khi B đang ở mức thấp và trường hợp ngược lại là B có cạnh lên trong khi A đang ở mức thấp. Khi quay thuận thì biến đếm ở các chế độ tăng lên 1 tại các thời điểm đọc và ngược lại

Từ lý thuyết đọc encoder trên, ta thấy rằng việc đọc xung encoder cho những loại encoder có CPR lớn như trong hệ thống này sử dụng (lên đến 8400 CPR) với phương pháp sử dụng ngắt ngoài bắt cạnh 2 pha A và B sẽ rất tốn thời gian xử lý việc đọc và tính toán giá trị encoder, ngoài ra còn có các yếu tố khác ảnh hưởng lớn đến quá trình đọc như nhiễu lên đường truyền, méo dạng xung, … sẽ gây ra sai số lớn, để khắc phục khuyết điểm trên , dòng Tiva C TM4C123GH6PM đã tích hợp 2 module QEI làm chức năng chuyên biệt đọc encoder, việc đọc xung này hoạt động độc lập với quá trình xử lý của CPU, truy cập giá trị encoder thông qua các thanh ghi của module, ngoài ra module còn hỗ trợ nắn dạng xung ngõ vào, tách lỗi sai trong quá trình đọc xung, hỗ trợ tính toán vận tốc tức thời của động cơ

|  |
| --- |
|  |

Module QEI có 2 chế độ đọc: một là chế độ *CAPMODE set,* đọc 2 pha A và pha B theo chế độ x4, với chế độ này yêu cầu encoder cung cấp 2 xung A và B lệch nhau 90 độ, dùng mối quan hệ giữ cạnh lên của xung để suy ra chiều quay của động cơ, nếu động cơ quay thuận thì biến đếm position tăng thêm 1. Chế độ đọc thứ 2 là chế độ *CAPMODE clear,* chế độ này yêu cầu encoder cung cấp 1 xung clock báo số vòng quay của động cơ và 1 tín hiệu chỉ hướng quay của động cơ. Ở hai chế độ, khi biến đếm vượt quá giá trị đặt(thanh ghi QEIMAXPOS), giá trị sẽ được đặt về 0.

Để tính được vận tốc tức thời của động cơ, module QEI sử dụng một timer và một thanh ghi đếm số xung đếm được trong khoảng thời gian được cấu hình cho tirmer đếm, giá trị nạp cho timer đếm thông qua thanh ghi QEILOAD, khi timer đếm về 0 sẽ xuất hiện một ngắt và hệ thống tự reload lại giá trị nạp QEILOAD cho timer, số xung đếm được trong chu kỳ trước được lưu vào thanh ghi QEISPEED, trong khi số xung đo được tại chu kỳ hiện tại được lưu trong thanh ghi QEICOUNT

|  |
| --- |
|  |

Hình: Giản đồ đọc xung encoder của module QEI

Công thức chuyển đồi giữa giá trị thanh ghi QEISPEED thành số vòng/ phút thông qua công thức sau:



Với RPM: tốc độ vòng/ phút

Clock: tần số dao động của hệ thống ( = 16 MHz)

: Hệ số scale vận tốc ( = 1)

SPEED: giá trị đọc vận tốc của thanh ghi QEISPEED  
 LOAD: giá trị nạp cho timer cấu hình đọc tốc độ

(LOAD =  = )

PPR: số xung trên một vòng encoder ( = 4xCPR)

EDGES: tùy theo chế độ đọc(4 cho chế độ *CAPMODE set* và 2 cho *CAPMODE clear*)

Từ đó, theo các giá trị cấu hình trên, tỷ lệ giữa RPM và SPEED sẽ là: hay 

### Điều khiển động cơ DC sử dụng thuật toán PID

PID là thuật toán điều khiển vi tích phân tỉ lệ được sử dụng để điều khiển những đối tượng như nhiệt độ lò nhiệt, tốc độ và vị trí dộng cơ, áp suất,… Thuật toán điều khiển PID là một thuật toán được sử phổ biến nhất trong các hệ thống điều khiển phản hồi công nghiệp hiện nay

Để có thể điều khiển được đối tượng, PID phải lấy được giá trị đầu ra của hệ thống tức hệ thống phải có phản hồi về giá trị điều khiển, sau đó được so sánh với giá trị đặt mong muốn để tìm ra được sai số e, từ sai số e được thuật toán PID xử lý để tìm ra được tác động u tác động vào đối tượng để có thể làm ngõ ra bám theo giá trị đặt



PID

System



*e*

Bộ điều khiển PID gồm 3 khâu điều khiển là tỷ lệ(P), tích phân(I) và vi phân(D), ngõ ra của bộ điều khiển PID chính là tổng của ngõ ra 3 khâu này



Khâu tỷ lệ là tích giữa sai số hiện tại với hằng số độ lợi : 

Khâu tích phân là tích giữa tổng sai số tức thời theo thời gian(tích phân sai số) với độ lợi : 

Khâu vi phân là tích giữa độ dốc sai số theo thời gian(đạo hàm bậc một của sai sô) với độ lợi vi phân : 

Vậy ngõ ra tác động 

Hàm truyền của bộ điều khiển PID liên tục:



Sự tác động của các hệ số , ,  đến hệ thống được thể hiện qua bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Thời gian khởi đông | Quá độ | Thời gian xác lập | Sai số ổn định |
|  | Giảm | Tăng | Tăng | Giảm |
|  | Giảm | Tăng | Thay đồi nhỏ | Giảm đáng kể |
|  | Giảm ít | Giảm ít | Giảm ít | Không tác động về lý thuyết |

Với sự phát triển của các hệ thống số yêu cầu đặt ra cho điều khiển cũng cần có những bộ điều khiển rời rạc, thuật toán điều khiển PID rời rạc được sử dụng rộng rãi trong nhiều hệ thống điều khiển số hiện nay, thuật toán hoàn toàn có thể được lập trình trên máy tính cũng như trong các hệ thống nhúng

Chuyển đổi về miền thời gian rời rạc cho từng khâu:

* Khâu tỷ lệ: 
* Khâu tích phân: 
* Khâu vi phân: 

Với T là chu kỳ lấy mẫu (giây)

Từ đó cho ta ngõ ra tác động u(kT) là:



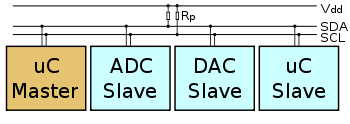
Để thực thi thuật toán này trong vi điều khiển, sử dụng một timer định thì thời gian lấy mẫu T, tại mỗi thời điểm kT tức timer đếm xong có thể dùng một ngắt timer để thực thi thuật toán PID này, dưới đây là thuật toán PID điều khiển vận tốc được nhúng trong vi điều khiển TM4C123GH6PM:

|  |
| --- |
| void PIDVerCalc\_motor\_0(int32\_t Speed, int32\_t MaxResponse)  {  PIDVelocity\_motor\_0.PIDError = PIDVelocity\_motor\_0.SetPoint - Speed;  PIDVelocity\_motor\_0.pPart = PIDVelocity\_motor\_0.Kp \* PIDVelocity\_motor\_0.PIDError;  PIDVelocity\_motor\_0.iPart += PIDVelocity\_motor\_0.Ki \* PIDVelocity\_motor\_0.PIDError\*time;  PIDVelocity\_motor\_0.dPart = PIDVelocity\_motor\_0.Kd \* (PIDVelocity\_motor\_0.PIDError - PIDVelocity\_motor\_0.PIDErrorTemp1)/time;  PIDVelocity\_motor\_0.PIDResult += (PIDVelocity\_motor\_0.pPart + PIDVelocity\_motor\_0.iPart + PIDVelocity\_motor\_0.dPart);  if (PIDVelocity\_motor\_0.PIDResult > MaxResponse)  PIDVelocity\_motor\_0.PIDResult = (float)MaxResponse;  if (PIDVelocity\_motor\_0.PIDResult < -1 \* MaxResponse)  PIDVelocity\_motor\_0.PIDResult = (float)(-1 \* (MaxResponse));  PIDVelocity\_motor\_0.PIDErrorTemp1 = PIDVelocity\_motor\_0.PIDError;  } |

### Lập trình giao tiếp MPU6050

* Giới thiệu chuẩn I2C:

Chuẩn giao tiếp I2C là một chuẩn giao tiếp truyền nhận dữ liệu 2 chiều thông qua 2 đường dây tín hiệu(đường tín hiệu SDA và đường clock đồng bộ SCL) để giao tiếp với các ngoại vi như RAM, ROM, RTC, IMU,...



Hình: Kết nối các thiết bị giao tiếp qua chuẩn I2C

Có thể có nhiều thiết bị giao tiếp trên cùng một bus I2C mà không bị xung đột với nhau do chúng có địa chỉ riêng của nó, địa chỉ này được phát đi dạng broadcast từ thiết bị làm master đến các thiết bị slave khác và chỉ thiết bị có địa chỉ này mới cho phép truyền nhận dữ liệu với master

Chuẩn I2C hỗ trợ truyền ở 3 tốc độ: tốc độ tiêu chuẩn(100Kbps), tốc độ nhanh(400Kbps), tốc độ cao(1.7Mbps hoặc 3.4Mbps)

|  |
| --- |
|  |

Hình: Quá trình truyền nhận dữ liệu theo chuẩn I2C

Đầu tiên, Master gửi tín hiệu START báo cho tất cả thiết bị khác rằng đang có một thiết bị muốn tạo kêt nối để truyền nhận dữ liệu, sau đó Master gửi tiếp địa chỉ của thiết bị mà nó muốn giao tiếp kèm theo cờ read/write và chờ response, khi thiết bị có địa chỉ này nhận được byte địa chỉ thì nó thực hiện gửi ACK về cho master biết rằng nó đã sẵn sàng truyền nhận dữ liệu với master, sau đó là quá trình truyền nhận dữ liệu giữa 2 bên, thiết bị truyền 8 bit đến thiết bị nhận và sau đó thiết bị nhận trả lời ACK báo đã nhận đúng, không đúng hay chưa nhận dữ liệu, kết thúc quá trình truyền nhận master gửi đi một tín hiệu STOP

* Cấu hình module I2C TM4C123GH6PM

TM4C123GH6PM có 4 module I2C hỗ trợ chức năng làm master và slave, hỗ trợ truyền nhận ở cả 3 tốc độ I2C, tạo các ngắt I2C sau khi tạo tín hiệu START và STOP, giao tiếp tương thích với các thiết bị theo chuẩn TTL(áp mức ngưỡng là 2.5V)

Cấu hình module I2C được thực hiện bởi tập lệnh hỗ trợ của thư viện Tivaware như sau:

|  |
| --- |
| SysCtlPeripheralEnable(SYSCTL\_PERIPH\_I2C1); // Enable I2C1 peripheral  SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA); // Enable GPIOA peripheral  // Use alternate function  GPIOPinConfigure(GPIO\_PA6\_I2C1SCL);  GPIOPinConfigure(GPIO\_PA7\_I2C1SDA);    GPIOPinTypeI2CSCL(GPIO\_PORTA\_BASE, GPIO\_PIN\_6); // Use pin with I2C SCL peripheral  GPIOPinTypeI2C(GPIO\_PORTA\_BASE, GPIO\_PIN\_7); // Use pin with I2C peripheral    I2CMasterInitExpClk(I2C1\_BASE, SysCtlClockGet(), true); // Enable and set frequency to 400 kHz |

* Giao tiếp với module MPU6050:

Module MPU6050 giao tiếp qua chuẩn I2C tần số nhanh (400Kbps), hỗ trợ ghi dữ liệu cấu hình thanh ghi, đọc dữ liệu từ thanh ghi ở 2 chế độ đọc ghi một byte và nhiều byte thanh ghi

|  |
| --- |
|  |

Hình: Chế độ đọc một byte và nhiều byte thanh ghi MPU6050 qua chuẩn I2C

Trong đó AD+W: Là địa chỉ của MPU6050 7 bit (0x68) và thêm một bit quy định đọc/ ghi R/W

RA (Register address): Địa chỉ thanh ghi

Trong chế độ đọc nhiều byte thanh ghi, MPU6050 tự động tăng địa chỉ thanh ghi lên 1 sau khi gửi ACK cho data trước

* Một số thanh ghi sử dụng để cấu hình MPU6050:

+ Thanh ghi chia tần số lấy mẫu **SMPRT\_DIV địa chỉ 0x19, quy định tần số lấy mẫu cho MPU6050**

|  |
| --- |
|  |



Với Gygro output rate = 8kHz khi DLPF trong thanh ghi CONFIG được disable(DLPF\_CFG = 0 hoặc bằng 7), hoặc bằng 1kHz DLPF được enable

+ Thanh ghi CONFIG có cấu trúc như sau:

|  |
| --- |
|  |

Thanh ghi CONFIG gồm 2 chức năng :

cấu hình chức năng đồng bộ dùng tín hiệu tham chiếu ngoại được kết nối đến chân FSYNC, tín hiệu này sẽ được lấy mẫu và giá trị sau khi lấy mẫu được ghi vào LSB cho thanh ghi data được quy định bởi 3 bit thanh ghi **FSYNC( external Frame Synchronization)**

|  |
| --- |
|  |

Chức năng thứ 2 của thanh ghi CONFIG là cấu hình bộ lọc thông thấp **DLPF(Digital Low Pass Filter)**cho cả hai thành phần tín hiệu Gygroscope và Accelerometer

|  |
| --- |
|  |

+ Thanh ghi cấu hình Gygroscope(**GYRO\_CONFIG, đại chỉ 0x1B**) và Accelerometer (**ACCEL\_CONFIG, địa chỉ 0x1C**)

|  |
| --- |
|  |

Thanh ghi cho phép kích hoạt cấu hình tự kiểm tra mô tả trong các thanh ghi **SELF\_TEST\_X, SELF\_TEST\_Y, SELF\_TEST\_Z**

Thanh ghi còn cho phép cấu hình thang đo(độ nhạy) của Gygroscope và Accelerometer theo 3 bit FS\_SEL và ASF\_SEL theo bảng dưới đây

|  |  |
| --- | --- |
|  |  |

+ Thanh ghi chứa dữ liệu đo Gygroscope và Accelerometer:

Thanh ghi dữ liệu đo Accelerometer có địa chỉ từ 0x3B – 0x40

|  |
| --- |
|  |

Thanh ghi dữ liệu đo Gygroscope có địa chỉ từ 0x43 – 0x48

|  |
| --- |
|  |

Với giá trị đo theo mỗi trục là 16 bit ADC

1. **ROS**

## Hệ thống với ROS

ROS không phải là hệ điều hành dành cho robot. ROS chỉ cung cấp giao tiếp giữa các tiến trình. Chức năng của robot được chia ra thành từng node độc lập. Các node thực hiện xử lý thông tin nhận được từ một hay nhiều node khác thông qua các topic. Mỗi topic sẽ mang thông tin của một message. Kiểu dữ liệu của message rất đa dạng như: thông tin vị trí robot, vận tốc, bản đồ,… Đề tài thiết kế robot theo mô hình sau:

Trong đó chức năng của các khối là:

Hình 4- Telepresence Robot theo mô hình ROS

(4) Differential

Transform

(6) Kinect

(9) Move Base

(7) Robot

State Publisher

(8) Map Server

(5) SLAM

/AMCL

(2) Twist

To Motor

(1) Telerobot

Teleop

(3) Launchpad

1. Telerobot Teleop: nhận thông tin điều khiển từ người dùng ở chế độ điều khiển bẳng tay hoặc từ node (9) ở chế độ tự hành. Thông tin điều khiển được chuyển thành kiểu dữ liệu [vận tốc tuyến tính ; vận tốc góc ].
2. Twist to motor: nhận thông tin vận tốc từ node (1), chuyển thành vận tốc điều khiển cho từng motor.
3. Launchpad: thực hiện giao tiếp giữa ROS và kit TivaC. Thông tin được truyền qua chuẩn UART. Node (3) truyền thông tin vận tốc điều khiển và nhận lại thông tin các cảm biến, encoder từ kit.
4. Diffterential transform: Nhận thông tin encoder từ node (3), tính toán chuyển về thông tin quỹ đạo của robot.
5. SLAM/AMCL: Ở lần đầu tiên hoạt động, robot được điều khiển thủ công và xây dựng bản đồ với SLAM. Sau khi đã có bản đồ, robot sẽ từ hành và xác định vị trí của mình trong bản đồ với AMCL.
6. Kinect: Cảm biến độ sâu của kinect được sử dụng để thu thập thông tin khoảng cách của vật cản trong môi trường, giúp xây dựng bản đồ và tránh vật cản khi robot tự vận hành.
7. Robot State Publisher: quản lý các bộ phận của robot như kích thước khung xe, khaongr cách giữa hai bánh trái phải, vị trí đặt cảm biến độ sâu,…
8. Map Server: cung cấp bản đồ cho robot trong quá trình vận hành. Bản đồ được chia thành hai loại:

* Bản đồ toàn cục: chứa thông tin toàn bộ vật cản trong môi trường hoạt động của robot, thường là bản đồ tĩnh đã được xây dựng từ AMCL ở node (5).
* Bản đồ cục bộ: chứa thông tin của vật cản trong một cửa sổ phía trước robot, thường là bản đồ động giúp robot xử lý các vật cản xuất hiện ngẫu nhiên như con người.

1. Move Base: đưa ra điều khiển đường đi cho robot thông qua thông tin bản đồ từ Map Server và vị trí từ AMCL. Thông tin vận tốc được truyền xuống node (2).

## Mô hình quỹ đạo

Vị trí của robot trong không gian không thể được đo một cách trực tiếp như sử dụng GPS được do môi trường hoạt động trong nhà và yêu cầu độ chính xác cao. Tuy nhiên quỹ đạo của robot có thể được tính một cách gián tiếp thông qua các cảm biến như encoder, cảm biến độ sâu,… Robot được điều khiển thông qua vận tốc tuyến tính   và vận tốc góc  như hình 4-2.

Hình 4- Các đại lượng vận tốc của robot



Trong đó: l là khoảng cách giữa hai bánh xe (m).

, là vận tốc của hai bánh xe phải và trái (m/s).

I là tâm ảo của đường tròn khi robot được chạy theo quỹ đạo cong.

là khoảng cách từ tâm ảo đến trọng tâm của robot.

Khi đó, hai giá trị vận tốc tuyến tính và góc được gửi đến node (2) để chuyển thành vận tốc động cơ trái và phải theo công thức sau:

(4.1)

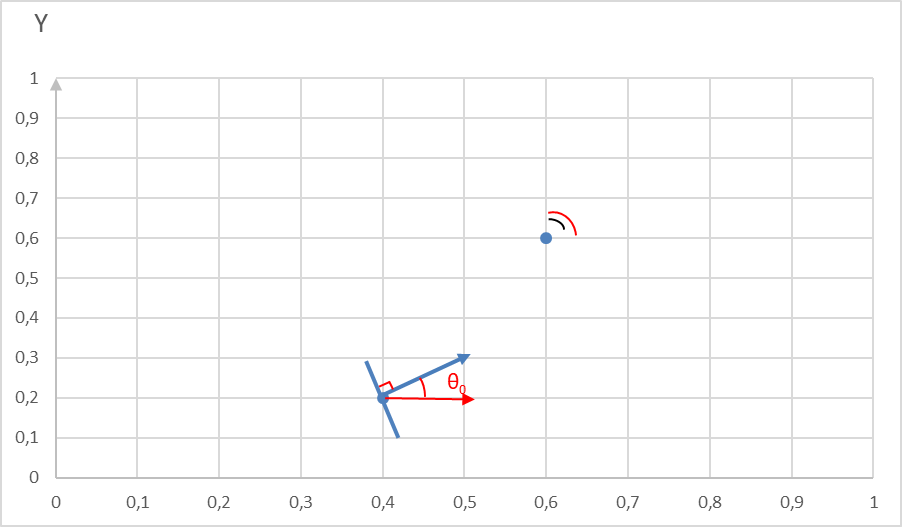
(4.2)

Vi điều khiển xử lý dữ liệu tốc độ nhận được xuất ra xung PWM điều khiển 2 động cơ thông qua cầu H, kết hợp với thuật toán PID đã được nhúng sẵn để điều khiển chính xác vận tốc đã đặt. Từ hai giá trị vận tốc trái và phải ta có thể tính được bán kính mà robot quay quanh tâm I thông qua công thức toán sau:

(4.3)

Đồng thời vi điều khiển cũng truyền giá trị của encoder cho Launchpad Node để tính quảng đường robot đã đi được. Vị trí của robot được biểu diễn qua bộ ba thông số , trong đó là tọa độ trong hệ tọa độ Decade và  là hướng của robot so với trục x. Quỹ đạo của robot sau thời gian t là . . Giả sử robot có vị trí đầu . Vị trí mới của robot sau một khoảng thời gian , di chuyển một đoạn  và xoay góc  như trên hình 4-3 được tính theo công thức (4.4)

x



θ

x0, y0

θ'

x’, y’

d

Hình 4- Cập nhật vị trí robot

 (4.4)

Trong đó, x, y và  được là khoảng cách di chuyển và góc xoay của robot trong khoảng , được tính theo công thức (4.5):

 (4.5)

Tốc độ góc  có thể được tính từ encoder của robot. Tuy nhiên trong quá trình thử nghiệm, việc tính tóc độ góc từ encoder có sai số lớn, gây ảnh hưởng bài toán xác định vị trí. Vì vậy, một IMU được dùng để đọc giá trị tốc độ góc , từ đó tính được góc quay của robot. Khoảng cách di chuyển  của robot được tính từ khoảng cách di chuyển độc lập của hai bánh xe trái và phải thông qua giá trị đọc về của encoder theo công thức (4.6):

 (4.6)

Trong đó:

, là số xung encoder đọc được trong khoảng (xung).

,  là khoảng cách hai bánh xe trái và phải đi được (m).

là số xung encoder khi bánh xe di chuyển được một mét (ppm).

Trên thực tế, vị trí của robot sau khi điều khiển không thể được tính toán một cách chính xác do nhiều yếu tố như:

- Mô hình toán của robot chưa chính xác do đơn giản hóa các phương trình.

- Nhiễu của hệ thống và môi trường.

- Sai số của các cảm biến, encoder,…

Kết quả là vị trí thực tế của robot có thể bị lệch so với kết quả robot nhận được từ encoder và IMU như trên hình 4-4.

Hình 4- Vị trí robot trên thực tế

Inititial Pose

Control

Final Pose

Tập hợp các vị trí có thể của robot ở thời điểm  là phân phối xác suất phụ thuộc vào vị trí thông tin điều khiển trước đó . Theo hình 4-4, robot có vị trí đầu là , sau khi nhận được điều khiển sẽ di chuyển đến vị trí là một trong các điểm màu xanh dương. Trong khi đó, robot chỉ đọc được 1 giá trị của encoder để suy ra vị trí là điểm màu đỏ. Mỗi vị trí  có thể xảy ra được gọi là một particle. Với mô hình quỹ đạo, vị trí của robot có thể được suy ra từ thông tin của encoder. Tuy nhiên, mô hình này có độ chính xác chưa cao và không sử dụng thông tin thu được từ môi trường. Một mô hình khác tận dụng thông tin khoảng cách đến vật cản trong môi trường là mô hình đo lường.

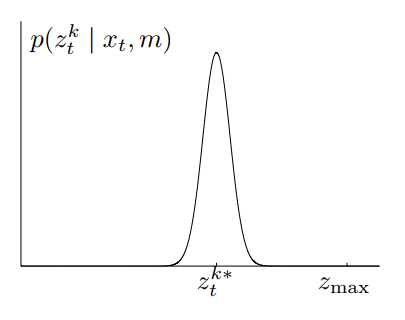
## Mô hình quan sát

Robot được trang bị Kinect với khả năng thu được bản đồ độ sâu của các vật gần nhất trong tầm hoạt động hiệu dụng. Giống như mô hình quỹ đạo, mô hình đo lượng cũng giải quyết vấn đề nhiễu bằng cách định nghĩa một phân phối mật độ xác suất , trong đó là phép đo của cảm biến, là vị trí của robot ở thời điểm và m là bản đồ của môi trường. Ở một thời điểm , phép đo thu được một mảng phần tử các giá trị đo độ sâu với . Giả sử các điểm đo là độc lập với nhau, ta có thể xấp xỉ:

 (4.7)

Giả sử khoảng cách đo đúng là  của phép đo  . Nhiễu có thể được biểu diễn bằng một phân phối Gaussian có trung bình  và độ lệch chuẩn  như hình 4-5.

Hình 4- Mô hình giám sát với nhiễu



Mô hình giám sát có độ lệch chuẩn nhỏ hơn mô hình quỹ đạo. Vì vậy mô hình này có thể được áp dùng để xác định vị trí của robot, đặc biệt là những robot có trang bị cảm biến đo khoảng cách với độ chính xác cao. M. Montemerlo avas S. Thrun [1] đưa ra phương pháp ước lượng vị trí dựa trên thông tin bản đồ trước đó và quan sát ở hiện tại:

 (4.8)

Trong đó là dự đoán ban đầu để thực hiện phép đo . Vùng lân cận quan sát được ở thời điểm được so sánh với bản đồ ở thời điểm . Vị trí  được suy ra từ vị trí của vùng lân cận trong bản đồ và vị trí đã dự đoán trước đó .

## Xây dựng bản đồ với SLAM

Xây dựng bản đồ và xác định vị trí của robot là hai vấn đề có quan hệ phụ thuộc lẫn nhau. Bản đồ *m* là kiểu bản đồ dựa trên vị trí. Mỗi điểm có thể có một trong ba trạng thái: có vật cản, không có vật cản và không có thông tin. Kiểu bản đồ trên thuận lợi cho việc tìm đường tránh vật cản. Quỹ đạo của robot được chọn đi qua những điểm trên bản đồ không có vật cản. Bộ lọc thường được xử dụng để giải quyết bài toán là EKF [2]. Bộ lọc xử dụng mô hình quỹ đạo để ước lượng vị trí ban đầu. Sau đó cảm biến đọc thông tin vật cản của môi trường. Tiếp theo, robot tiếp tục di chuyển và quan sát sự di chuyển của vật cản. Những ước lượng vị trí tin cậy sẽ quan sát được sự di chuyển của vật cản phù hợp với sự di chuyển của robot. Phương pháp này đòi hỏi phái trích được đặc trưng của vật cản. Đặc biệt những vật cản phải có đặc trưng duy nhất. Môi trường có nhiều vật cản cùng đặc tính với nhau thì phương pháp trên trở nên kém hiệu quả. G. Grisetti và các đồng nghiệp [3] đưa ra giải pháp sử dụng bộ lọc Rao-Blackwellized cùng những cải tiến để nâng cao độ chính xác của vị trí và bản đồ. Bộ lọc chia SLAM thành hai phần: xác định vị trí của robot và xây dựng bản đồ từ vị trí đã xác định theo phương trình (4.9):

 (4.9)

Bản đồ có thể được xây dựng dễ dàng khi đã biết phân phối xác suất của vị trí . Bộ lọc Rao-Blackwellized sử dụng phương pháp SIR. Các bước chính của một bộ SIR để ước lượng một phân phối mật độ xác suất là:

Bước 1: Lấy mẫu vị trí hiện tại từ thông tin vị trí trước đó  thông qua phân phối tham chiếu . Thông thường phân phối tham chiếu được chọn theo mô hình quỹ đạo đã trình bày ở phần 4.2.

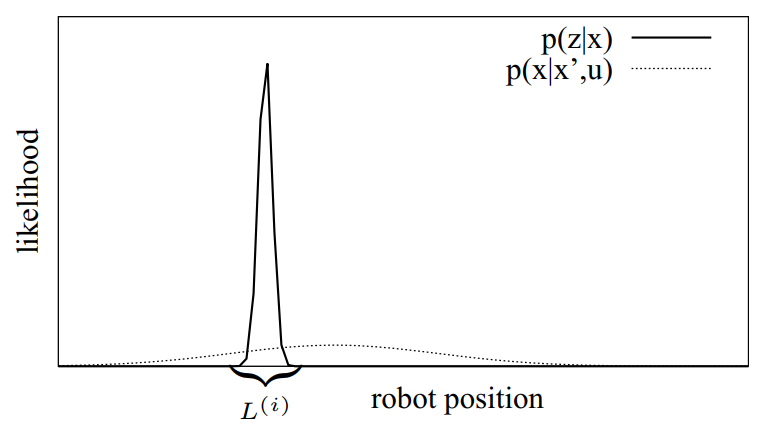
Bước 2: Tính trọng số cho mỗi particle theo công thức:

 (4.10)

Bước 3: Loại bỏ các particle có trọng số thấp giúp các mẫu từ phân phối tham chiếu xấp xỉ được phân phối cần tìm.

Bước 4. Xây dựng bản đồ từ vị trí và quan sát .

Hình 4- So sánh mô hình quỹ đạo và giám sát



Bộ lọc Rao-Blackwellized không qui ước phân phối tham chiếu ở bước 1 và bước 2. Đa số robot hoạt động với động cơ có thể sử dụng mô hình  để làm phân phối tham chiếu. Tuy nhiên G. Grisetti và đồng nghiệp [4] đã chỉ ra mô hình giám sát chiếm ưu thế hoàn toàn so với mô hình quỹ đạo khi robot được trang bị cảm biến đo khoảng cách với độ chính xác cao. Theo hình 4-6, Trong vùng lân cận , phân phối của mô hình quỹ đạo được xem như một hằng số, trong khi đó mô hình giám xác có thể được xem như một phân phối Gaussian. Vùng lân cận xác định bởi:

 (4.11)

Các particle  sẽ được lấy mẫu trong vùng lân cận này. Tổng hợp quá trình ước lượng với bộ lọc Rao-Blackwellized như sau:

Bước 1: Tập hợp particle hiện tại , tập hợp particle trước đó . Với mỗi particle , dùng mô hình quỹ đạo để tìm ra vị trí dự đoánsau khi điều khiển đã trình bày ở mục 4.2.

Bước 2: So sánh quan sát và bản đồ  thu được vị trí dự đoán đã trình bay ở mục 4.3.

Bước 3: Lấy K mẫu particle ở vùng lân cận : , với là độ rộng của vùng lân cận.

Bước 4: Ước lượng phân phối tham chiếu  trong vùng lân cận bằng K particle. Phân phối tham chiếu được xấp xỉ một phân phối Gaussian với các tham số:

- Trung bình

 (4.12)

- Phương sai:

 (4.13)

- Hệ số chuẩn hóa:

 (4.14)

Bước 5: Lấy mẫu particle:

. (4.15)

Bước 6: Tính trọng số

 (4.16)

Bước 7: Cập nhật bản đồ theo , và .

Bước 8: Cập nhật tập hợp particle ở thời điểm hiện tại:

 (4.17)

Bước 9: Tiến hành loại bỏ các particle có trọng số thấp nếu

 (4.18)

Trong đó  là trọng số chuẩn hóa từ . Bước 9 thể hiện quá trình loại bỏ mẫu có chọn lọc. Nếu phân phối cần tìm và phân phối tham chiếu giống nhau thì các trọng số của các particle sẽ bằng nhau. Khi đó sẽ lớn hơn mức ngưỡng T. Vì vậy việc loại bỏ particle chỉ xảy ra khi có sự khác biệt lớn giữa hai phân phối. Sau quá trình lấy mẫu lại, trọng số của tất cả các particle đều được đánh giá lại từ đầu. Quá trình trên được lập đi lập lại trong suốt quá trình xây dựng bản đồ.

## Xác định vị trí robot với AMCL

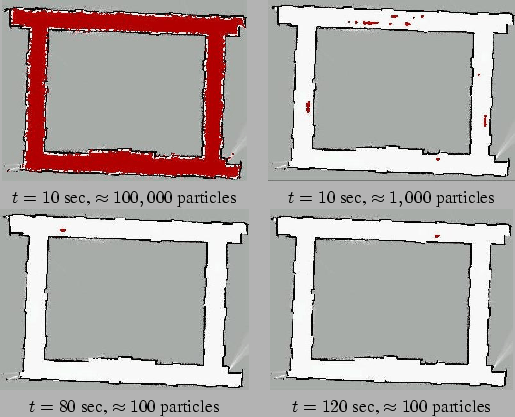
AMCL(Adapted Monte-Carlo Localization) là một thuật toán định vị robot trong môi trường 2 chiều dựa trên thông tin dữ liệu chuyển động của robot kết hợp với dữ liệu thu được từ cảm biến đo khoảng cách như sonar, laser range-finders

|  |
| --- |
|  |

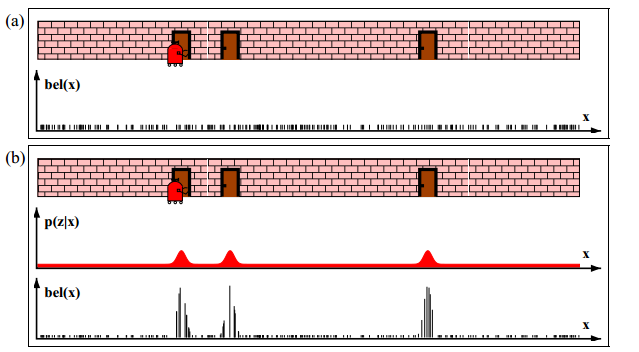
Thuật toán MCL được thực hiện như bảng trên , thuật toán này được xây dựng trên mô hình bộ lọc particle có ngõ vào gồm: tập hợp các hạt(mẫu) thu được từ chu kỳ lấy mẫu trước, là thông tin đã chuyển đổi về đơn vị vận tốc đọc từ encoder và IMU( do mô hình chuyển động được sử dụng là mô hình odometry), phép đo từ cảm biến đo khoảng cách và thông tin lấy từ một bản đồ đã quét *m,*  ngõ ra của thuật toán là tập hợp các hạt(mẫu) thể hiện việc ước lượng vị trí robot hiện tại trong bản đồ

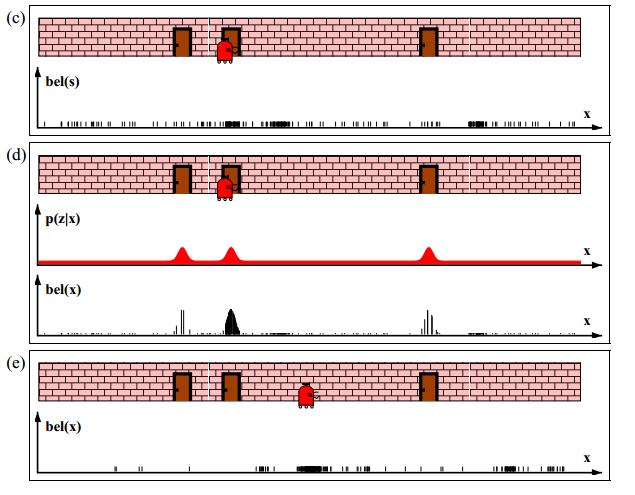
Dòng thứ 4 của thuật toán MCL tạo ra các mẫu thông qua thuật toán sample motion model odomnetry được phân tích ở phần trước, dòng thứ 5 tính trọng số cho từng mẫu qua dựa trên mô hình đo lường, dòng thư 8 – 11 thực hiện việc lấy mẫu lại(resampling), trước khi resampling các hạt phân bố theo hàm **, việc resampling thực chất là việc thưc hiện phép toán kết hợp các mẫu  với trọng số của nó, từ đó những hạt có trọng số nhỏ thì trong hàm phân phối  có giá trị rất nhỏ, ta có thể loại đi những hạt này trong tập hợp các hạt . Thuật toán này cho phép đối sánh dữ liệu thu được với một bản đồ có sẵn

Thuật toán có thể thích nghi vì nó có thể điều chỉnh được số lượng hạt(mẫu) sử dụng trong bộ lọc particle, khi vị trí của robot có độ không chắc chắn lớn số lượng hạt(mẫu) yêu cầu trong bộ lọc tăng lên làm cho tốc độ tính toán chậm, khi robot di chuyển và thu thập được nhiều dữ liệu môi trường xung quanh hơn từ cảm biến, số lượng hạt (mẫu) yêu cầu trong bộ lọc giảm xuống, tốc độ tính toán được cải thiện



Hình x: Sự hội tụ số hạt trong quá trình chạy thuật toán AMCL





Hình: Mô tả quá trình thực hiện thuật toán AMCL để định vị robot

Hình x mô tả quá trình thực hiện thuật toán AMCL trên hệ thống robot tự hành, lúc khởi tạo vị trí robot là hoàn toàn không chắc chắn khi chưa thực hiện phép quan sát môi trường xung quanh, do đó các hạt  được lấy mẫu ngẫu nhiên và đều trên khắp bản đồ, sau khi quan sát với phép đo zt dòng 5 của thuật toán AMCL thực hiện tính trọng số cho các hạt đã lấy mẫu trước đó thu được, hàm bel(xt) và tập hợp các hạt sau khi đã được resampling được thể hiện như hình x.b, sau đó mô hình di chuyển một đoạn u(t) dời tập hợp các hạt  theo biến u(t), tiếp tục thực hiện quan sát để tính trọng số cho tập hợp các mới cập nhật, sau đó lại tính hàm bel(x(t)) và resampling, quá trình được lặp lại cho những lần tiếp theo với số hạt giảm dần và việc định vị ví trí robot ngày càng chính xác

## Phương pháp tránh vật cản

### Tránh vật cản cố định từ bản đồ đã vẽ

Node (9) Move Base đảm bảo tìm đường đi trong bản đồ thu được từ bộ lọc Rao-Blackwellized bằng thuật toán A\*. Được sử dụng rộng rãi trong khoa học máy tính, A\* tìm đường qua các điểm rời rạc, gọi là các node, được phát triển bởi P. Hart và các đồng nghiệp [3] vào năm 1968. Thuật toán sử dụng một dự đoán về khoản cách để đánh đổi giữa khoảng cách tối ưu và tốc độ xử lý khi bản đồ có độ phân giải cao. Sau mỗi bước xử lý, thuật toán sẽ quyết định đường đi nào được tiếp tục mở rộng hướng đến đích dựa vào trọng số:

 (4.19)

Trong đó:

 là node cuối cùng trên đường đi đang được xử lý.

là quảng đường từ điểm xuất phát đến node .

 là ước lượng quảng đường từ node  đến đích.  chỉ phụ thuộc vào vị trí node  và đích, không phụ thuộc vào vật cản trong bản đồ. Thường có ba cách xác định  theo hình 4-7:

1.  (4.20)
2.  (4.21)
3.  (4.22)

Goal(x,y)

Node(x,y)

Node(x,y)

Node(x,y)

Goal(x,y)

Goal(x,y)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| a) | b) | c) |

Hình 4- Mô hình ước lượng 

Thuật toán A\* gồm các bước sau:

Bước 1: Khởi tạo danh sách quan sát chứa các node có thể tạo thành đường dẫn đến đích. Ban đầu danh sách quan sát chỉ chứa node xuất phát.

Bước 2: Khởi tạo danh sách đã kiểm tra chứa các node đã ước lượng đường đi. Ban đầu danh sách này rỗng.

Bước 3: Kiểm tra các node trong danh sách quan sát:

* Trích node  có trọng số nhỏ nhất trong danh sách quan sát, chuyển vào danh sách đã kiểm tra.
* Kiểm tra các node lân cận của , gọi là  với =1,2,…8.
* Với mỗi điểm  không nằm trong danh sách đã kiểm tra:
* Thêm vào danh sách quan sát.
* Nếu đường đi đến  thông qua  ngắn hơn đường đi đến  đã tìm thì cập nhật lại và ghi nhớ node  xuất phát từ .
* Nếu  là đích đến thì tìm ngược lại đường đi từ  đến điểm xuất phát.

Ví dụ tìm đường từ điểm xuất phát đến đích và tránh vật cản bằng thuật toán A\* thể hiện ở hình 4-8.

(2)

10 80

**90**

(8)

10 80

**90**

(9)

14 70

**84**

Node *q*: (1)

Danh sách quan sát:

(2) (8) (9)

Danh sách đã kiểm tra (1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Node *q*: (9)

Danh sách quan sát:

(2) (3) (8) (10) (16) (17)

Danh sách đã kiểm tra:

(1) (9)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (2)  10 80  **90**  (8)  10 80  **90**  (9)  14 70  **84**  (3)  28 70  **98**  (10)  24 60  **84**  (16)  24 60  **84**  (17)  28 50  **78** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (2)  10 80  **90**  (8)  10 80  **90**  (9)  14 70  **84**  (3)  28 70  **98**  (10)  24 60  **84**  (16)  24 60  **84**  (17)  28 50  **78**  (11)  42 50  **92**  (24)  38 40  **78**  (25)  42 30  **72** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Node *q*: (17)

Danh sách quan sát:

(2) (3) (8) (10) (11) (16) (24) (25)

Danh sách đã kiểm tra

(1) (9) (17)

Node *q*: (25)

Danh sách quan sát:

(2) (3) (8) (10) (11) (16) (19) (20) (24) (26)

Danh sách đã kiểm tra:

(1) (9) (17) (25)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (2)  10 80  **90**  (8)  10 80  **90**  (9)  14 70  **84**  (3)  28 70  **98**  (10)  24 60  **84**  (16)  24 60  **84**  (17)  28 50  **78**  (11)  42 50  **92**  (24)  38 40  **78**  (25)  42 30  **72**  (19)  56 30  **78**  (26)  52 20  **72** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Node q: (26)

Danh sách quan sát:

(2) (3) (8) (10) (11) (16) (19) (20) (24) (27)

Danh sách đã kiểm tra:

(1) (9) (17) (25) (26)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (2)  10 80  **90**  (8)  10 80  **90**  (9)  14 70  **84**  (3)  28 70  **98**  (10)  24 60  **84**  (16)  24 60  **84**  (17)  28 50  **78**  (11)  42 50  **92**  (24)  38 40  **78**  (25)  42 30  **72**  (19)  56 30  **78**  (26)  52 20  **72**  (20)  66 20  **78**  (27)  62 10  **72** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | (2)  10 80  **90** | (3)  28 70  **98** |  |  |  |  |
| (8)  10 80  **90** | (9)  14 70  **84** | (10)  24 60  **84** | (11)  42 50  **92** |  |  |  |
|  | (16)  24 60  **84** | (17)  28 50  **78** |  | (19)  56 30  **78** | (20)  66 20  **78** |  |
|  |  | (24)  38 40  **78** | (25)  42 30  **72** | (26)  52 20  **72** | (27)  62 10  **72** |  |

Hình 4- Tìm đường bằng A\*

Node *q*: (27)

Danh sách quan sát:

(2) (3) (8) (10) (11) (16) (19) (20) (24) (27)

Danh sách đã kiểm tra:

(1) (9) (17) (25) (26) (27)

Phát hiện đích ở node: (28)

**Chú thích:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Không có vật cản |  | Vùng lận cận *qi* |
|  |  |  |  |
|  | Có vật cản |  | Node trong danh sách đã quan sát |
| *n*  *gn*  *hn*  *fn*  (2)  10 80  **90** |  |  |  |
|  |  |  |  |
|  | Giá trị tại node *n* |  |  |

* + 1. Tránh vật cản xuất hiện ngẫu nhiên trong quá trình hoạt động

Trong quá trình di chuyển đến đích thông qua đường đi tìm được bằng thuật toán A\*, robot có thể gặp vật cản ngẫu nhiên như con người gây ra tình trạng tắc nghẽn. Một chuỗi hành vi được qui ước để robot giải quyết vấn đề trên:

Hình 4- Tránh vật cản ngẫu nhiên

**Có vật cản**

**Không có vật cản**

Xung quanh robot được chia thành ba vùng. Vùng thứ nhất là hình chữ nhật bao phủ vừa đủ kích thước của robot, vì vậy robot có thể xoay tự do trong khu vực này. Vùng thứ hai là hình chữ nhật lớn hơn do người dùng qui định kích thước để cảnh báo nguy cơ va chạm. Vùng thứ ba là vùng ngoài vùng có nguy cơ va chạm nhưng vẫn trong tầm quan sát của cảm biến. Đầu tiên, vật cản xuất hiện trong vùng thứ ba, robot đợi một khoảng thời gian để tiến hành loại bỏ vật cản ra khỏi bản đồ. Hành vi này có ý nghĩ khi có một người di chuyển nhanh qua tầm nhìn của robot. Nếu người này đứng lại, robot tiến hành xoay tại chỗ để đổi hướng đến khi không có vật cản. Trong trường hợp vẫn mắc kẹt, robot tiến hành xóa vật cản trong vùng thứ hai và quét lại bản đồ, thường đi kèm thao tác xoay tại chỗ. Nếu robot vẫn không tìm được đường tránh xem như đã bị mắc kẹt.

1. **KẾT QUẢ**

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

## Hướng phát triển

# TÀI LIỆU THAM KHẢO

1. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters
2. Mohammed Faisal, Ramdane Hedjar, Mansour Alsulaiman, Khalid Al-Mutabe, Hassan Mathkour, "Robot localization using extended kalman filter with infrared sensor", *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, vol. 00, no. , pp. 356-360, 2014.
3. P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, July 1968.

# PHỤ LỤC