ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CƠ KHÍ
BỘ MÔN CƠ ĐIỆN TỬ

š š & › ›

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

ROBOT DI ĐỘNG THEO DẤU TƯỜNG

Sinh viên thực hiện: DOÃN MINH ĐĂNG
MSSV: P9900012
Cán bộ hướng dẫn : TS. NGUYỄN TẤN TIẾN

CHƯƠNG TRÌNH ĐÀO TẠO KỸ SƯ CHẤT LƯỢNG CAO
KHÓA 1: 1999 - 2004
TP.Hồ Chí Minh, 07/2004

Lời cảm ơn

Để thực hiện đề tài, tác giả đã nhận được rất nhiều sự chỉ dẫn, giúp đỡ và động viên quý báu của nhiều người, thiếu một trong các sự giúp đỡ đó cũng có thể làm cho đề tài không đạt kết quả như hiện nay.

Trước hết, em xin bày tỏ lòng cảm ơn sâu sắc đối với thầy TS. Nguyễn Tấn Tiến, người thầy hướng dẫn đã tận tình chỉ cho em phương pháp nghiên cứu khoa học, thầy cũng đã cung cấp cho em rất nhiều kiến thức chuyên sâu để thực hiện đề tài.

Em cũng vô cùng cảm ơn cô Th.S Trần Thị Ngọc Dung và các thầy cô ở chương trình đào tạo Kỹ sư chất lượng cao, thầy TS. Nguyễn Văn Giáp và các thầy cô ở bộ môn Cơ Điện Tử, Khoa Cơ khí, Trường Đại Học Bách Khoa Tp.HCM đã tham gia quá trình đào tạo và hướng dẫn em trong suốt thời gian học đại học, nhờ các thầy cô mà em có đủ kiến thức và lòng tự tin để thực hiện đề tài nghiên cứu này cũng như các đề tài trong tương lai.

Bên cạnh đó, sự hợp tác và giúp đỡ của bạn bè và các thế hệ đàn anh cũng giúp tôi rất nhiều trong việc thực hiện đề tài này. Em xin được cảm ơn KS. Lưu Tuấn Anh, Khoa Công nghệ Vật Liệu, người đã hướng dẫn em đi vào nghiên cứu vè robot, Th.S Trần Văn Tùng và các bạn ở Phòng thí nghiệm Thiết Kế Máy đã tích cực giúp đỡ em trong thời gian thực hiện đề tài. Tôi cũng xin chân thành cảm ơn các bạn cùng học lớp Cơ Điện Tử – Việt Pháp 99, đặc biệt là các bạn Đoàn Hiệp, Nguyễn Anh Kiệt, Phạm Huỳnh Phong, Nguyễn Minh Trung, những người cùng nghiên cứu về robot di động đã cho tôi các ý kiến đóng góp quý giá!

Con cũng xin cảm ơn gia đình đã luôn chăm sóc và quan tâm đến việc học của con, con vô cùng cảm ơn và luôn tự hào vì có Bố, Mẹ, Chị luôn động viên con trong quá trình học tập.

Và cuối cùng, tôi xin gửi lời cảm ơn tới những người đã tham gia giúp đỡ tôi trong quá trình thực hiện luận văn mà tôi chưa nêu tên ở đây, sự giúp đỡ của họ dù ít hay nhiều cũng đóng góp một phần vào kết quả thực hiện đề tài tốt nghiệp này.

Tp. Hồ Chí Minh, ngày 19 tháng 7 năm 2004

Doãn Minh Đăng

Mục lục

Danh mục các hình vẽ

Danh mục các bảng

Tóm tắt đề tài

Trong thời đại công nghiệp ngày nay, Robot ngày càng được sử dụng phổ biến trong sản xuất cũng như trong cuộc sống của con người. Robot đã có một vị trí quan trọng khó có thể thay thế được, nó giúp con người để làm việc trong các điều kiện nguy hiểm, khó khăn. Ngoài ra, Robot còn được dùng vào các lĩnh vực thám hiểm không gian, quân sự, giải trí... Lĩnh vực Robot di động đang ngày càng chiếm được sự quan tâm của các nhà nghiên cứu và xã hội. Từ tình hình thực tế đó, việc xây dựng các chương trình hoạt động cho các Robot là điều thiết yếu đặc biệt đối với các Robot di động. Bài toán Robot di động bám tường (wall-following problem) là một trong các bài toán thường gặp của Robot kiểu phản xạ (reactive paradigm), nó đã được giải bằng nhiều cách khác nhau. Trong đề tài "Robot di động theo dấu tường", bài toán Robot di động bám tường được giải quyết bằng một bộ điều khiển hồi tiếp đầy đủ trạng thái mà kết quả đã được chứng minh bằng mô phỏng. Một cảm biến tiếp xúc dùng các encoder được tạo ra để sử dụng cho robot. Mô hình robot được chế tạo để tiến hành thí nghiệm nhằm kiểm chứng giải thuật của bộ điều khiển. Kết quả thí nghiệm là căn cứ để phát triển bộ điều khiển dành cho bài toán wall-following trong các Robot sau này.

Abstract

This project studies on control of a wall-following mobile robot. The wall is assumed unknown. A tactile sensor is constructed to measure the angle and the distance of mobile robot relatively to the wall that the mobile robot must follow. A nonlinear controller is built based on Lyapunov stability. The experiment has been carried out to verify the study. Based on this result, the proposed controller can be used for control of a wall-following mobile robot problem.

1
TỔNG QUAN VÀ ĐẶT VẤN ĐỀ

1.1 Giới thiệu chung về robot

Khái niệm Robot theo nghĩa chung thường được hiểu đồng nghĩa với khái niệm tự động hoá công nghiệp, điều này chỉ đúng một phần bởi vì: thứ nhất, Robot chỉ là một thành phần trong hệ thống tự động hoá, thứ hai là tự thân việc trình bày, miêu tả Robot trong sinh hoạt xã hội ít nhiều phóng đại.

Những Robot xuất hiện lần đầu tiên ở NewYork vào ngày 9/10/1922 trong vở kịch"Rossum's Universal Robot" của nhà soạn kịch người Tiệp Khắc là Karen Chapek, còn từ Robot là một cách gọi khác của từ Robota-theo tiếng Tiệp có nghĩa là công việc lao dịch. Khi đó, Karen Chapek cho rằng Robot là những người máy có khả năng làm việc nhưng không có khả năng suy nghĩ.

Gần một thế kỷ tiếp theo, khái niệm robot đã liên tục được phát triển, đóng góp thêm bởi nhiều nhà nghiên cứu, nhiều công ty chuyên về lĩnh vực robot. Dưới đây là bảng tóm tắt quá trình lịch sử hình thành và phát triển của công nghệ chế tạo robot, và những tác động của khoa học cũng như xã hội đối với từng thời kỳ [4].

Bảng 1.1 Tóm tắt lịch sử phát triển của công nghệ robot

Mốc thời gian	Nghiên cứu và phát triển	Ứng dụng trong công nghiệp	Kỹ thuật hỗ trợ	Các yếu tố ảnh hưởng
1920	Khái niệm robot xuất hiện trong tiểu thuyết			
1940	Phát minh ra cánh tay máy			
1950	Phát sinh khái niệm robot thông minh		Giới thiệu về bộ nhớ vòng	
1960	Giới thiệu về robot được điều khiển bằng máy tính Tăng cường nghiên cứu	Phát triển robot trong công nghiệp Ứng dụng ở NASA và NAVY	Máy tính dùng transitor Giới thiệu vi xử lý	
1970	Robot có trí thông minh nhân tạo	Sự bộc phát lần đầu tiên về robot	Phát triển vi xử lý	Sự hạn chế của nền kinh tế
1980	Robot dùng trong những công việc nguy hiểm (1983)	Robot công nghiệp thực tế và các ứng dụng rộng rãi khác	Kỹ thuật số Kỹ thuật quang	Nhu cầu tăng cường tự động
1990		Giới thiệu về robot thông minh trong sản xuất	Điều khiển logic Nghiên cứu về robot trí thông minh nhân tạo	Robot gây nên thất nghiệp
2000		Robot giống con người	Các tiến bộ về cơ khí	

Trước những năm 1970, người ta chỉ tập trung vào việc phát triển những robot tay máy hoạt động trong các nhà máy công nghiệp. Sau đó mới xuất hiện những khái niệm về robot thông minh, và các nghiên cứu bắt đầu tập trung hơn vào robot di động. Một trong những chuyên gia đầu ngành về robot di động là Hans P. Moravec (bắt đầu nghiên cứu từ năm 1964), và hiện nay, chuyên nghiên cứu về robot di động là Sebastien Thruns.

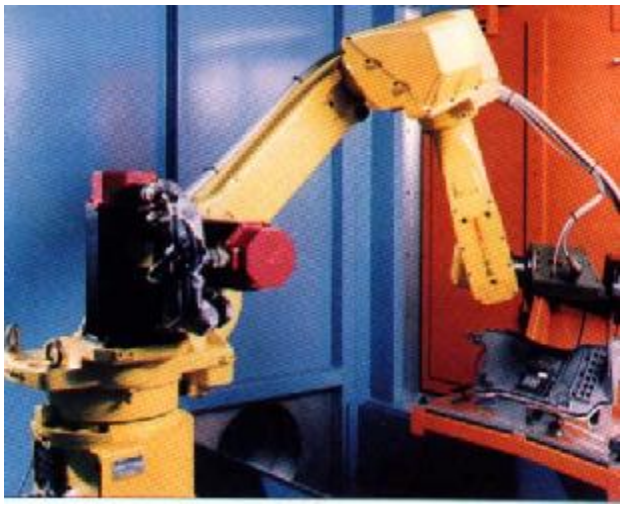
Các robot di động có người điều khiển đã được dùng cho các mục đích quân sự, các nhiệm vụ nguy hiểm như phá mìn, thăm dò đáy đại dương, hầm mỏ, kiểm tra các đường ống ngầm, hay thăm dò sao Hoả...

Sản phẩm robot di động được sản xuất đại trà và đưa vào thị trường lần đầu tiên là robot hút bụi Roomba và Trilobite của hãng Electrolux năm 2003.

Ở hình 1.1 là một số hình ảnh về các Robot và ứng dụng của nó:

a. Robot tự hành Sojourner thám hiểm sao Hoả

b. Robot dò mìn



c. Tay máy dùng trong công nghiệp

d. Robot hút bụi Trilobite

Hình 1.1 Một số hình ảnh về robot và các ứng dụng

1.2 Tổng quan về các bài toán của robot di động [5]

Vấn đề "**navigation**" (tạm dịch là "di chuyển") là vấn đề trọng tâm của robot di động. Để di chuyển được, robot phải thực hiện một loạt các tác vụ, mỗi tác vụ gắn với một bài toán nhỏ trong bài toán "navigation". Các bài toán đó gồm:

Mapping: là công việc lập bản đồ môi trường hoạt động của robot. Nếu không được cung cấp dữ liệu trước thì robot phải có khả năng lập bản đồ.

Positioning: là việc định vị, robot phải có khả năng biết được mình đang ở đâu trong bản đồ toàn cục hoặc địa phương.

Path planning: là việc hoạch định đường đi sắp tới của robot, sau khi nó biết được bản đồ và biết mình đang ở vị trí nào.

Motion control: là việc điều khiển cho robot di động, tức là điều khiển các cơ cấu để robot đi theo con đường thu được từ bài toán "path planning".

Obstacle avoidance: là nhiệm vụ tránh chướng ngại vật khi robot đang di chuyển.

1.3 Bài toán di chuyển theo tường và các nghiên cứu liên quan

Bài toán di chuyển theo tường:

Việc di chuyển theo tường (wall following) là một tác vụ thường thấy ở robot di động, trong các môi trường biết trước hoặc không biết trước. Tác vụ này được dùng với các nhiệm vụ: tránh chướng ngại vật, đi theo tường biết trước, đi theo tường không biết trước.

Một số nghiên cứu đã thực hiện:

Turennout et al., 1992 [6]: một bộ điều khiển hồi tiếp dùng bộ quan sát để ước lượng khoảng cách và góc giữa robot với tường, dùng cảm biến siêu âm.

Medromi et al., 1994 [7]: dùng một bộ quan sát để ước lượng trạng thái của hệ phi tuyến với đầu vào không biết trước.

Urzelai, J. et al., 1997 [8]: sử dụng bộ điều khiển mờ để điều khiển robot VEA-II dùng cảm biến siêu âm.

Bemporad et al., 1997 [9]: đưa ra hướng tiếp cận dùng sự chồng chất cảm biến để ước lượng toạ độ của robot, trong đó dùng một bộ lọc Kalman để kết hợp tín hiệu từ cảm biến siêu âm và cảm biến độ dịch chuyển.

Yata et al., 1998 [10]: đưa ra phương pháp bám tường sử dụng việc đo góc nhờ cảm biến siêu âm.

Chung Tan Lam et al, 2004 [11]: một bộ điều khiển hồi tiếp phi tuyến khi biết khoảng cách và góc giữa robot với tường, một bộ điều khiển dựa trên bộ quan sát khi chỉ biết khoảng cách giữa robot với tường, dùng cảm biến cơ. Các bộ điều khiển ổn định theo tiêu chuẩn Lyapunov, các kết quả mô phỏng và thực nghiệm đã chứng tỏ hiệu quả của các bộ điều khiển.

Nguyễn Viết Hiệp và Phạm Đình Anh Vũ [3]: thiết kế bộ điều khiển hồi tiếp phi tuyến và bộ điều khiển dựa trên bộ quan sát để điều khiển robot đi theo tường, mô phỏng các bộ điều khiển để kiểm chứng tính hội tụ và ổn định.

Các kết quả nghiên cứu trên cho thấy hướng giải quyết bài toán robot di chuyển theo tường bằng các bộ điều khiển hồi tiếp phi tuyến là một hướng đi thích hợp. Ở đề tài của Nguyễn Viết Hiệp và Phạm Đình Anh Vũ, các bộ điều khiển đã được đưa ra và chứng minh bằng lý thuyết và mô phỏng, nhưng chưa được kiểm nghiệm thực tế. Dựa trên thành quả đó, luận văn này có nhiệm vụ là: **Kiểm chứng lý thuyết nghiên cứu về bài toán robot di chuyển theo tường đã được xây dựng trong luận văn tốt nghiệp đi trước.**

Để thực hiện mục tiêu trên, luận văn này tập trung vào các vấn đề sau:

- Nghiên cứu các bộ điều khiển cho robot bám tường.

- Thiết kế và chế tạo một đế di chuyển (mobile platform).

- Thiết kế và chế tạo một cảm biến tiếp xúc gắn lên robot để do sai số giữa robot với tường.

- Thiết kế và thực hiện các mạch điều khiển cho robot.

- Lập trình cho robot để hiện thực các bộ điều khiển.

- Thí nghiệm và so sánh kết quả thí nghiệm với kết quả mô phỏng.

- Nhận xét kết quả và kết luận.

1.3.1 Giới thiệu bài toán

Thiết kế và thực hiện bộ điều khiển dùng để điều khiển Mobile Robot di chuyển dọc theo tường với vận tốc và khoảng cách từ Robot đến tường cho trước.

Giả thiết của bài toán:

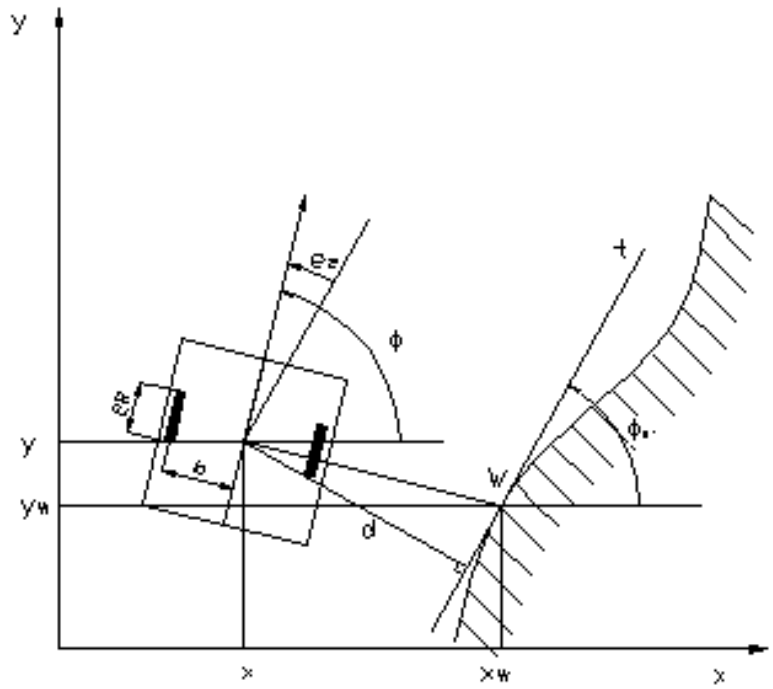
* Đo được khoảng cách d từ tường đến Robot.

* Đo được góc lệch giữa Robot và tường.

* Tường là đường cong trơn bất kỳ với bán kính cong của tường không nhỏ hơn khoảng cách d.

1.3.2 Mô hình toán học

Mô hình robot di động bám tường được cho như hình 1.2



Hình 1.2 Mô hình bài toán robot di động bám tường

Trong đó:

* (x,y): hệ trục toạ độ tuyệt đối.

* R: bán kính bánh xe của Mobile Robot.

* b: khoảng cách từ tâm Mobile Robot đến bánh xe.

* W:giao điểm của trục Y của Mobile Robot với tường.

* t: tiếp tuyến với tường tại điểm W.

* f : góc định hướng của Mobile Robot.

* f_w : góc nghiêng của tường.

* d :khoảng cách từ Mobile Robot đến tiếp tuyến t.

* d0 :khoảng cách yêu cầu từ Mobile Robot đến tường.

* e_2: góc lệch giữa mobile robot với tường.

Phương trình động học của Mobile robot đã được nghiên cứu bởi các công trình trước đây của nhiều tác giả [6]. Vận tốc được biểu diễn như sau:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{f} \end{bmatrix} = \begin{bmatrix} \cos f & 0 \\ \sin f & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \quad \textbf{[1-1]}$$

Với v: vận tốc dài của Robot.

w : vận tốc góc của Robot.

Mối quan hệ giữa v , w với vận tốc góc của hai bánh xe như sau:

$$\begin{bmatrix} w_{rw} \\ w_{lw} \end{bmatrix} = \begin{bmatrix} 1/R & b/R \\ 1/R & -b/R \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \quad \textbf{[1-2]}$$

với w_{lw} , w_{rw} : vận tốc góc của bánh xe trái và bánh xe phải của Mobile Robot.

1.3.3 Mục tiêu điều khiển

Yêu cầu của bài toán là phải điều khiển robot chạy theo biên dạng song song với tường, sao cho các sai số khoảng cách (e_1) và (e_2) hội tụ về 0, robot hoạt động ổn định trong một vùng lân cận điểm hội tụ.

1.4 Phương pháp giải quyết vấn đề

Để giải quyết bài toán robot di động đi theo tường, hai bộ điều khiển đã được trình bày trong đề tài luận văn tốt nghiệp ngành Cơ Điện Tử, trường ĐHBK Tp.HCM của hai sinh viên Nguyễn Viết Hiệp và Phạm Đình Anh Vũ. Một bộ điều khiển là hồi tiếp đầy đủ trạng thái (full-state feedback), một bộ điều khiển là dựa trên bộ quan sát (observer-based) [2]. Trong đề tài này, chúng tôi sử dụng các kết quả đó để thực hiện các bộ điều khiển, nhằm kiểm tra tính chính xác của kết quả mô phỏng.

Theo giả thiết của bài toán, ta biết được giá trị của cả 2 biến trạng thái là e_1 và e_2, như vậy ta có thể sử dụng một bộ hồi tiếp tất cả biến trạng thái (full-state feedback controller). Bộ điều khiển này sẽ được đề cập đến ở chương sau.

2
TÓM TẮT THUẬT TOÁN ĐIỀU KHIỂN

2.1 Mô hình bộ điều khiển

Công thức của bộ điều khiển full-state feedback:

$$\begin{cases} v = v_r \\ w = k_2[v_r - (e_1 + d_0)\hat{w}_w / \cos e_2]e_1 - k_1 \sin e_2 + \hat{w}_w \\ \dot{\hat{w}}_w = e_1(e_1 + d_0)\tan e_2 - \sin e_2 / k_2 \end{cases} \quad \textbf{[2-1]}$$

Trong đó:

v_r: vận tốc yêu cầu

d_0: khoảng cách yêu cầu

$\hat{w}$: ước lượng vận tốc góc của tường

k_1, k_2: các tham số của bộ điều khiển để ổn định Lyapunov

e_1, e_2: sai số về khoảng cách và góc giữa robot với tường

v: vận tốc dài của robot

w: vận tốc góc của robot (vận tốc của vectơ chỉ hướng robot)

2.2 Đặc tính bộ điều khiển (theo kết quả chứng minh và mô phỏng)

Qua kết quả chứng minh và mô phỏng, bộ điều khiển này có các tính chất sau:

- Ổn định theo tiêu chuẩn Lyapunov dạng 2.

- Các sai số e1 và e2 hội tụ về zero sau thời gian 8-10s, với các tham số mô phỏng lấy theo mô hình thực.

3
THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1 Kiến trúc robot

Robot được thiết kế theo kiểu kiến trúc SA (subsumption architect) [12], phân thành 2 lớp:

- Lớp dưới là các bộ điều khiển động cơ, bộ thu tín hiệu cảm biến.

- Lớp trên là bộ điều khiển trung tâm. Bộ điều khiển này không can thiệp vào hoạt động của các bộ phận ở lớp dưới.

Sơ đồ khối điều khiển được cho ở hình 3.1, gồm 01 khối xử lý chính (master module) và 02 khối xử lý phụ (slave module).

Master module có các chức năng:

- Đọc tín hiệu từ cảm biến, tính các sai số về khoảng cách và góc giữa robot với tường.

- Dùng giải thuật điều khiển full-state feedback để tìm các vận tốc dài và vận tốc góc mới cho robot, nhằm giảm các sai số.

- Chuyển vận tốc của robot thành vận tốc của 2 bánh xe, gửi các giá trị đó cho các slave module.

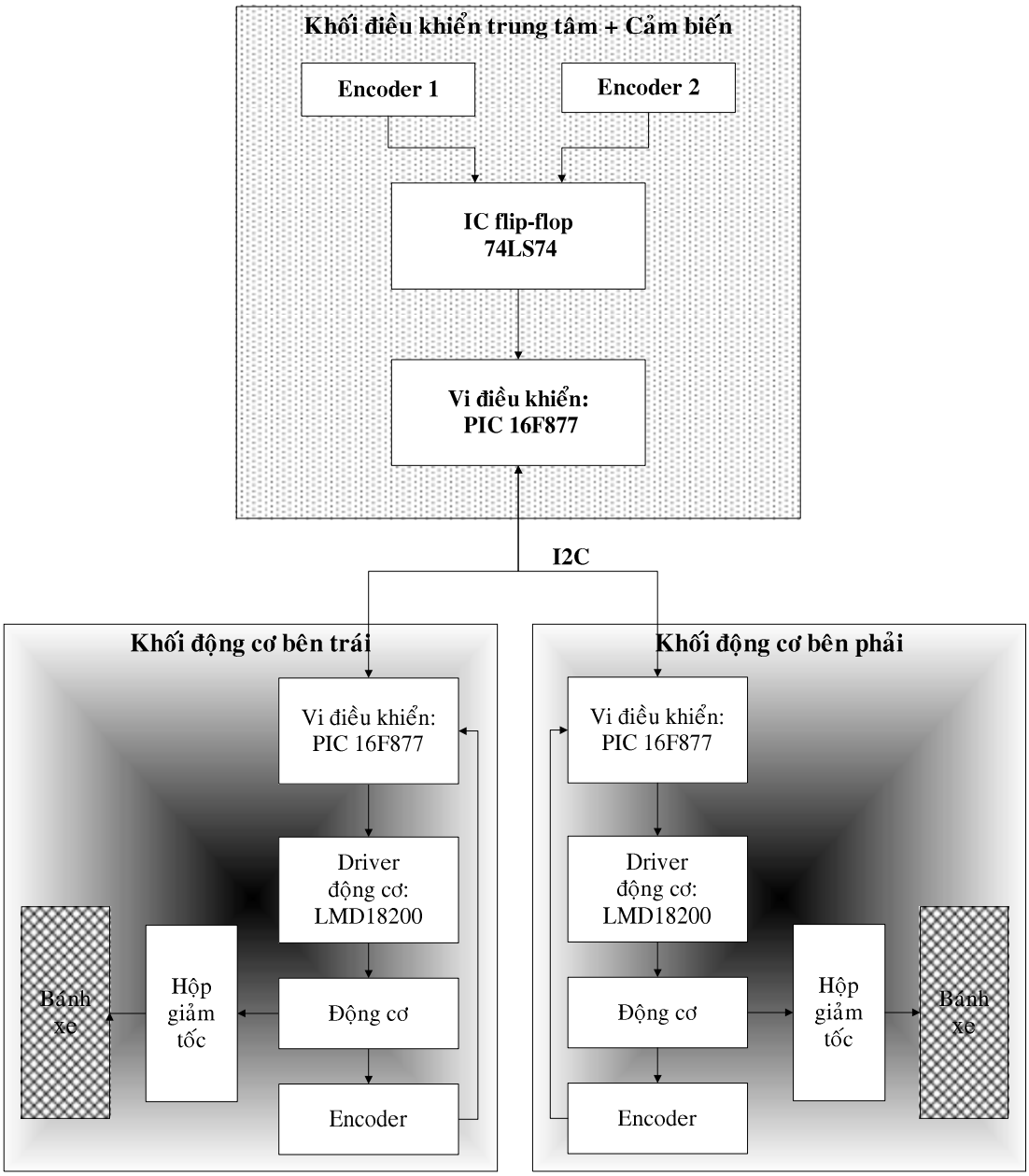
Slave module có các chức năng:

- Nhận lệnh từ master module (vận tốc mong muốn của mỗi bánh xe).

- Tính vận tốc hiện thời của robot qua số xung hồi tiếp trong 1 chu kỳ.

- Dùng giải thuật điều khiển PID để tính giá trị PWM, nhằm điều khiển bánh xe quay với vận tốc được ra lệnh.

Giao tiếp giữa master module và 2 slave module được thực hiện dựa trên chuẩn giao tiếp I2C, chuẩn này sẽ được giải thích rõ hơn ở đoạn dưới.



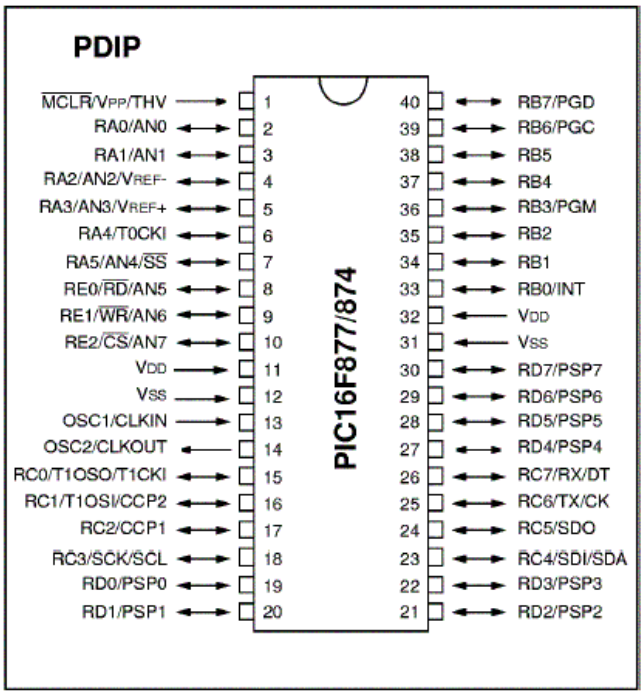
Hình 3.1 Sơ đồ khối của Wall-following mobile robot

3.2 Vi điều khiển PIC 16F877[13]

Vi điều khiển được chọn dùng trong đề tài là loại vi điều khiển PIC 16F877 của công ty Microchip. Sau đây là vài nét chính của vi điều khiển này:

Bộ xử lý chính:

- Loại bộ xử lý: RISC CPU
- Chỉ có tất cả 35 lệnh
- Hầu hết các lệnh là 1 chu kỳ máy, chỉ các lênh rẽ nhánh là 2 chu kỳ
- Tần số tối đa: với thạch anh 20 MHz – chu kỳ máy là 200ns
- Lưu được 8K lệnh trong bộ nhớ chương trình, mỗi lệnh 14 bits.
- Có 368 x 8 bytes bộ nhớ dữ liệu (RAM)
- Có 256 x 8 bytes bộ nhớ EEPROM
- Nhiều loại ngắt (14 loại)
- Power-on Reset (POR)
- Power-up Timer (PWRT) và
- Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) với bộ dao động tích hợp bên trong
- Bảo mật chương trình
- Có chế độ SLEEP để tiết kiệm năng lượng
- In-Circuit Serial Programming (ICSP - chuẩn ghi bộ nhớ chương trình khi vi xử lý vẫn ở trong mạch)
- Single 5V In-Circuit Serial Programming
- In-Circuit Debugging
- Vi xử lý truy cập được vào bộ nhớ chương trình
- Điện áp hoạt động rộng: 2.0V đến 5.5V
- Chịu được nhiệt độ trong môi trường công nghiệp
- Năng lượng tiêu thụ ít:
  < 0.6 mA ở 3V, 4 MHz
  20 mμA ở 3V, 32 kHz
  < 1 mμA ở chế độ standby



Hình 3.2 Sơ đồ chân PIC 16F877

Thiết bị ngoại vi:

- Timer0: 8-bit timer/counter với bộ chia trước 8-bit
- Timer1: 16-bit timer/counter với bộ chia trước, có thể hoạt động trong chế độ SLEEP
- Timer2: 8-bit timer/counter với bộ chia trước và chia sau 8-bit
- 2 bộ tích hợp Capture, Compare, PWM
  - Capture 16-bit, độ phân giải tối đa 12.5 ns
  - Compare is 16-bit, độ phân giải tối đa 200 ns
  - PWM có độ phân giải tối đa 10-bit
- 8 kênh biến đổi Analog-to-Digital 10-bit.
- Synchronous Serial Port (SSP) với 2 chuẩn: SPI (Master mode) và I2C (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI)
- Parallel Slave Port (PSP) 8-bits, có các chân điều khiển RD, WR và CS
- Brown-out Reset (BOR)

3.3 Thiết kế khung giao tiếp I2C

3.3.1 Lý do sử dụng giao tiếp I2C

Vi điều khiển PIC 16F877 hỗ trợ nhiều chuẩn giao tiếp, trong đó có chuẩn giao tiếp I2C được sử dụng làm cho giao tiếp giữa các module trong robot này. Chúng tôi chọn sử dụng chuẩn I2C vì một số lý do sau:

- I2C hỗ trợ một mạng nhiều thiết bị kết nối với nhau. Số thiết bị tối đa có thể có trong mạng I2C gồm 1 module master và 127 module slave (hiện tại mới dùng 2 module slave). Như vậy robot còn nhiều khả năng mở rộng về sau.

- Chuẩn I2C là chuẩn thông dụng, được sử dụng nhiều trong các linh kiện dùng chế tạo robot di động (như cảm biến siêu âm SRF08, cảm biến la bàn CMPS03). Nếu sau này chế tạo một robot di động dựa trên cơ sở robot bám theo tường, ta cũng dễ dàng tích hợp các linh kiện khác vào robot.

- Môi trường lập trình đang dùng hỗ trợ tốt cho việc giao tiếp bằng I2C. Chúng tôi đã thử nghiệm và thấy chương trình giao tiếp hoạt động ổn định.

Để hiểu thêm về chuẩn giao tiếp I2C, xin xem phụ lục A.

3.3.2 Khung giao tiếp I2C trong robot

Cách thức truyền tín hiệu qua I2C trong robot được quy định như sau:

- Module master chỉ ghi giá trị vào các module slave, không có chế độ đọc.

- Địa chỉ của module slave điều khiển di chuyển bánh trái là 0xA0, của module điều khiển di chuyển bánh phải là 0xC0. Mỗi khi muốn gửi dữ liệu cho module slave nào, module master gửi 1 byte địa chỉ của slave đó, sau đó gửi tiếp 2 byte dữ liệu.

- Dữ liệu gửi từ module master đến module slave là một biến số nguyên 16 bit. Trước khi gửi nó phải được cắt thành 2 byte, sau khi module slave nhận thì ghép 2 byte đó trở lại thành số nguyên 16 bit ban đầu.

- Ở mỗi chu kỳ hoạt động, module master gửi cho mỗi module slave một dữ liệu, là vận tốc yêu cầu module slave đạt được. 2 dữ liệu được gửi liên tiếp để giảm độ trễ đáp ứng giữa 2 module slave.

3.4 Thiết kế đế di chuyển và bộ điều khiển động cơ

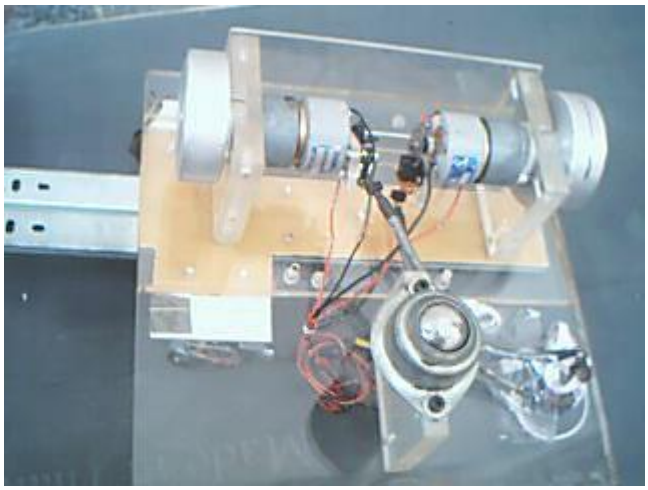
3.4.1 Thiết kế đế di chuyển

Các thành phần của đế di chuyển:

- 2 bánh dẫn động, gắn vào 2 động cơ có hộp giảm tốc, có hồi tiếp bằng encoder quang. Mỗi động cơ được điều khiển bởi 1 mạch điện riêng. Mạch điều khiển, động cơ và bánh xe cấu thành một module di chuyển, robot có 2 module di chuyển trái và phải tách rời nhau.

- Bánh tuỳ động là một bánh cầu, đặt ở sau xe.

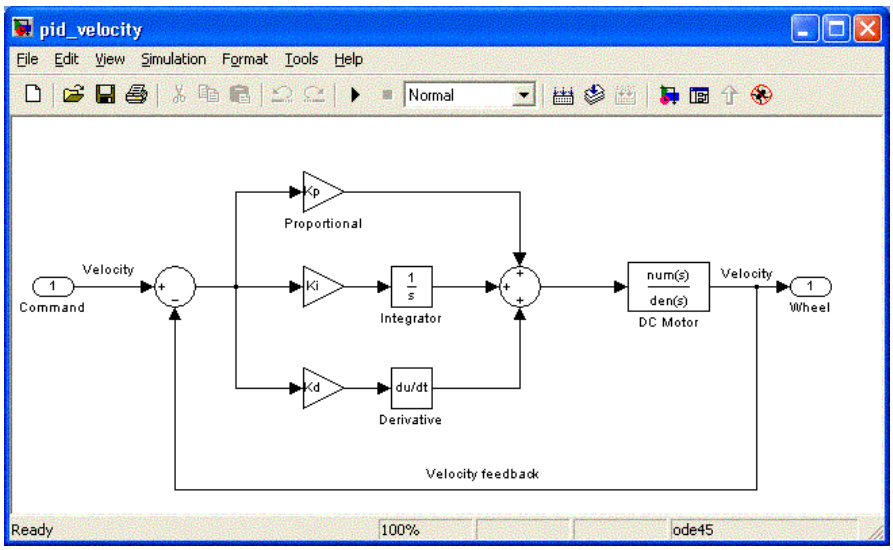
Bản vẽ thiết kế: xem bản vẽ mô hình robot (bản vẽ đính kèm).



Hình 3.3 Mô hình đế di chuyển lật ngược

3.4.2 Bộ điều khiển PID [15]

Mô hình điều khiển PID được sử dụng ở đây là:



Hình 3.4 Bộ điều khiển PID vận tốc theo mô hình song song

Bộ PID được sử dụng để đảm bảo vận tốc quay mà module điều khiển trung tâm ra lệnh cho module di chuyển thực hiện.

- Đầu vào của bộ PID: vận tốc yêu cầu, đơn vị là vòng/phút.

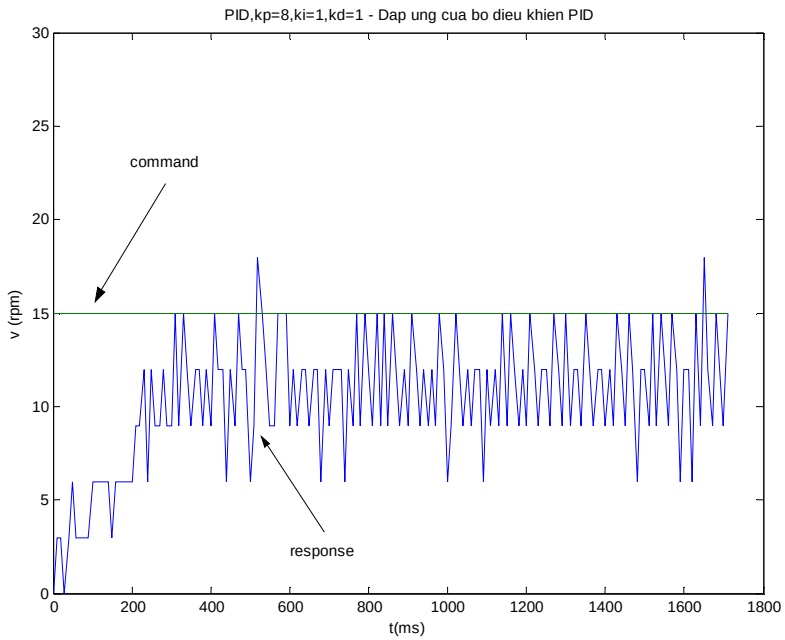
- Tín hiệu hồi tiếp: vận tốc hiện thời, từ số xung encoder đọc được trong 1 chu kỳ, đổi ra vòng/phút.

- Đối tượng điều khiển: vận tốc động cơ.

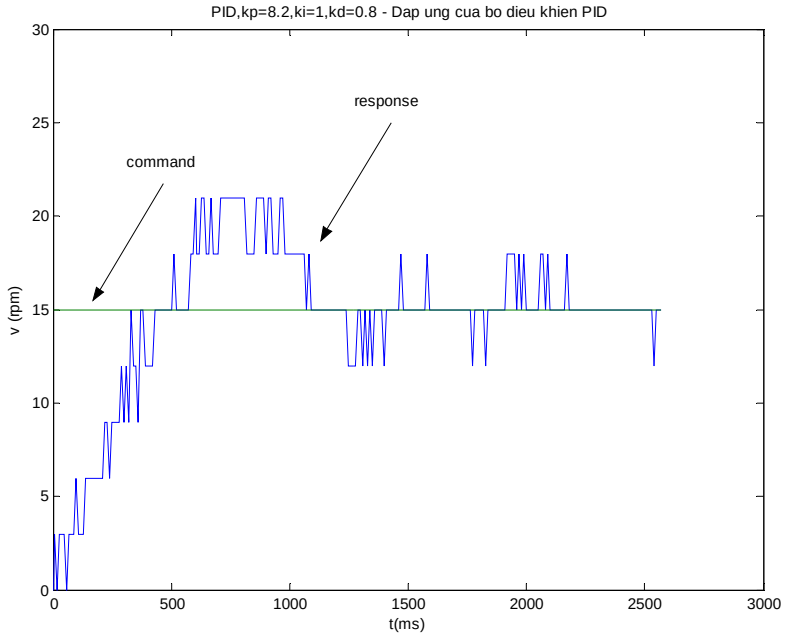
- Đầu ra của bộ PID: giá trị chu kỳ độ rộng xung (PWM duty) của điện áp hai đầu động cơ.

Phương pháp thực hiện bộ điều khiển PID vận tốc: do không có nhiều thời gian để tìm các thông số của động cơ nhằm mô hình hoá động cơ, chúng tôi lập trình bộ điều khiển động cơ trên vi điều khiển PIC với các giá trị kp, ki và kd thay đổi được. Khi hiệu chỉnh từ từ các tham số kp, ki và kd và xem đáp ứng của bộ điều khiển, chúng tôi lựa chọn được bộ tham số thích hợp cho bộ điều khiển.

Ta có thể thấy đáp ứng của bộ PID thay đổi theo sự thay đổi nhỏ của các hệ số điều khiển thông qua các đồ thị ở hình 3.5 và hình 3.6. Bộ PID được trình bày ở hình 4.6 được xem là tốt hơn bộ PID ở hình 3.5 (đáp ứng đạt mức yêu cầu, ít dao động), đó là bộ PID tốt nhất mà chúng tôi tìm được cho các module di chuyển của robot.



Hình 3.5 Đáp ứng của bộ điều khiển PID với kp=8, ki=1, kd=1



Hình 3.6 Đáp ứng của bộ điều khiển PID với kp=8.2, ki=1, kd=0.8

Các chỉ tiêu của bộ điều khiển PID:

- Thời gian đạt mức (rise time): 400ms

- Độ vọt lố (overshoot): 40% (tương ứng với 2 xung encoder/10ms)

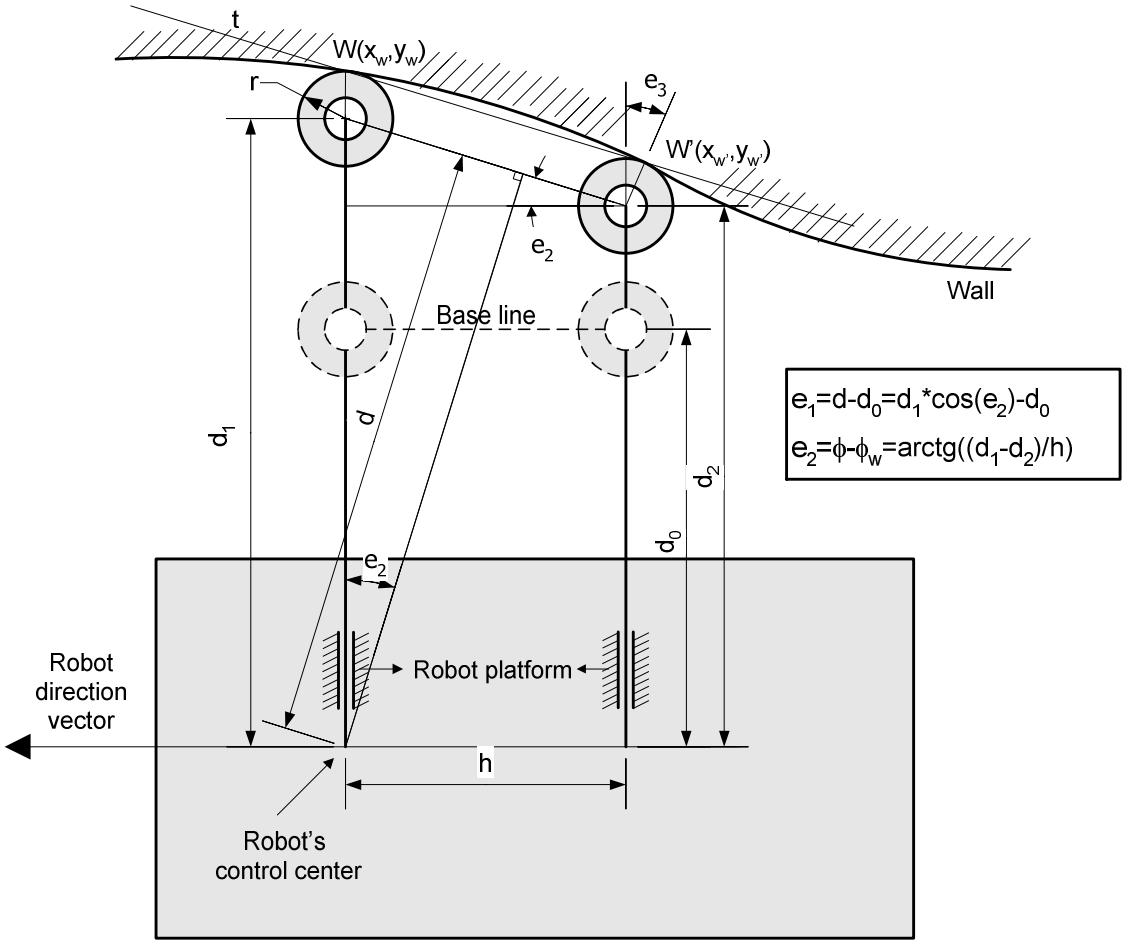
- Thời gian xác lập (settling time): 1000ms

Nhận xét: Do động cơ được sử dụng là một động cơ không tốt (công suất 4W, tốc độ tối đa khoảng 1000 vòng/phút, tỉ số hộp giảm tốc là 10), cộng với encoder có độ phân giải không cao (giá trị vọt lố 40% tương ứng với 2 xung encoder trong 1 chu kỳ lấy mẫu là 10ms), hơn nữa ta lại sử dụng động cơ ở tốc độ quay thấp, nên kết quả của bộ điều khiển PID không được tốt. Tuy nhiên, nếu không có các thành phần I và D thì bộ điều khiển P thông thường sẽ không thể đáp ứng được vận tốc mong muốn trong thời gian ngắn, đó là lý do phải sử dụng bộ điều khiển PID.

3.5 Thiết kế cảm biến

3.5.1 Mô hình toán học của cảm biến

Việc thiết kế một cảm biến tốt có ý nghĩa rất quan trọng trong robot di động này, để làm được điều đó, ta cần một mô hình toán học hợp lý. Một số mô hình toán học của cảm biến đã được nghiên cứu, cuối cùng chúng tôi đưa ra mô hình toán học ở hình 3.7 để thực hiện cảm biến cho robot.



Hình 3.7 Mô hình toán học của cảm biến

Cảm biến được set vị trí ban đầu ứng với e_2=0 và d=d_0. Khi robot chuyển động, 2 thanh trượt tiếp xúc với tường qua 2 con lăn tại các tiếp điểm W và W'. Do bán kính cong của tường lớn, ta xem tường trong đoạn WW' là thẳng. Hơn nữa, nhờ r nhỏ nên e_3 không đáng kể, ta xem e_3=0. Công thức tính e_1 và e_2 là:

$$e_1 = d - d_0 = d_1 * \cos(e_2) - d_0$$
$$e_2 = f - f_w = arctg(\frac{d_1 - d_2}{h}) \quad \textbf{[3-1]}$$

với d_1, d_2: độ dịch chuyển của thanh trượt 1 và 2

h: khoảng cách giữa 2 thanh trượt.

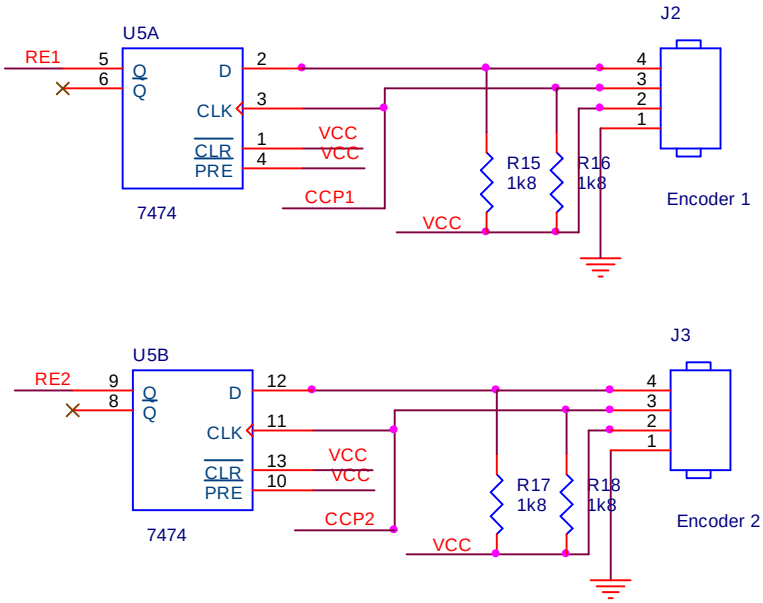
3.5.2 Thực hiện cảm biến

Module cảm biến bao gồm 3 bộ phận chính: 2 thanh trượt có con lăn ở đầu để tiếp xúc với tường, 2 encoder để dò độ dịch chuyển của các thanh trượt và 1 vi điều khiển để đọc độ dịch chuyển.

Thanh trượt được gắn lò xo (ở đây dùng dây thun) đẩy ra để luôn tiếp xúc với tường. Trên mỗi thanh trượt có gắn dây kéo dọc theo thanh, khi thanh trượt chuyển động, dây kéo sẽ kéo con lăn chuyển động. Do con lăn được nối chặt với trục

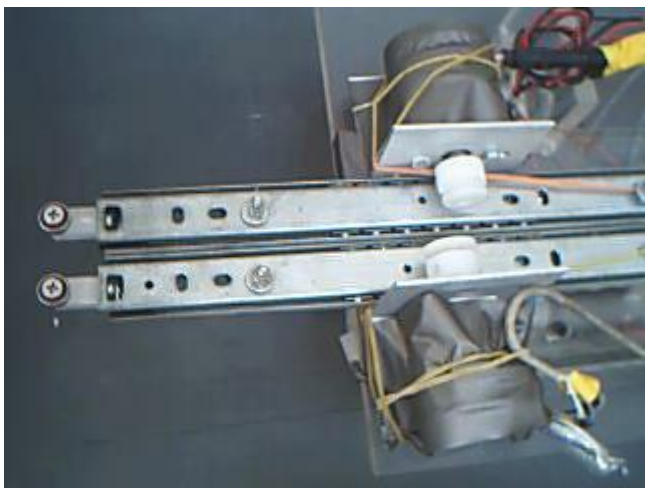
encoder nên chuyển động của con lăn sẽ làm quay đĩa của encoder, với mỗi dịch chuyển nhỏ của đĩa, encoder sẽ gửi xung về cho vi điều khiển để xử lý. Mỗi encoder truyền tín hiệu về cho vi điều khiển qua 2 đường A và B, chúng được cho vào một IC flip-flop để xác định chiều quay của encoder. Vi điều khiển nhận tín hiệu từ các encoder qua 2 cổng CCP (capture) và dùng 2 chân digital input (RE1 và RE2) để nhận biết chiều quay gửi từ IC flip-flop, các xung gửi từ encoder vào cổng CCP sẽ tạo ra ngắt (interrupt) để tiện việc tính toán trên vi điều khiển, tín hiệu ở các chân digital input sẽ cho vi điều khiển biết được xung đó ứng với chuyển động vào hay ra của thanh trượt.

Sơ đồ mạch thu tín hiệu từ encoder:



Hình 3.8 Phần đệm tín hiệu từ encoder vào vi điều khiển ở module master

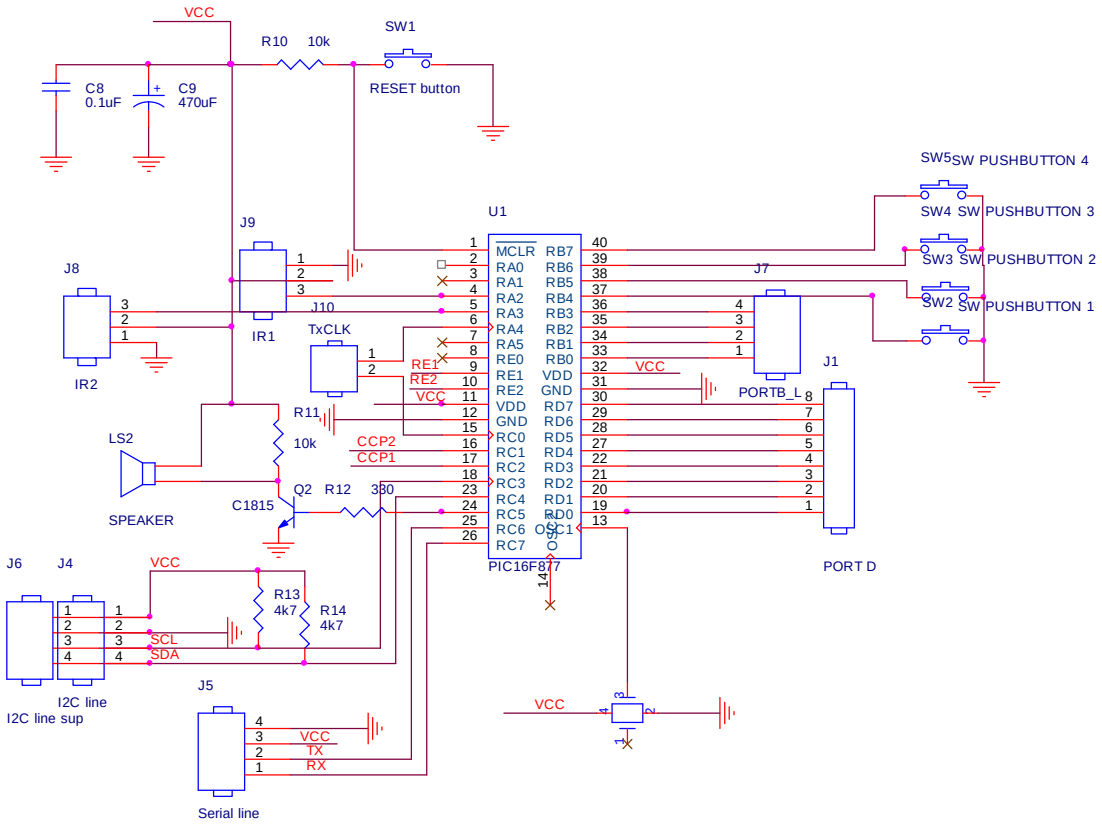
Kết quả thực hiện:



Hình 3.9 Hình chụp module cảm biến

3.6 Thiết kế các mạch điện tử

3.6.1 Mạch module master



Hình 3.10 Sơ đồ nguyên lý của mạch module master

Các thành phần và chức năng của chúng trong mạch master:

- Vi điều khiển PIC 16F877: bộ xử lý của cả mạch.

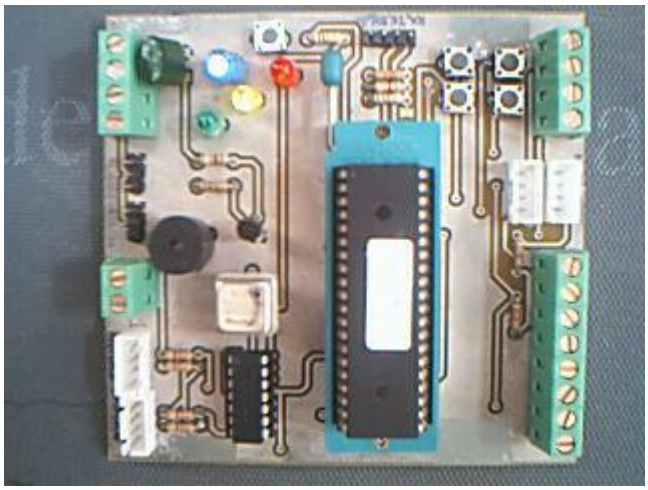
- Ngõ giao tiếp nối tiếp: để giao tiếp với mạch nạp và chương trình kiểm tra trên máy tính.

- Ngõ giao tiếp I2C: để giao tiếp với 2 vi điều khiển của các module slave.

- 2 ngõ nối với encoder để đọc tín hiệu từ cảm biến, có một IC flip-flop để đệm cho tín hiệu từ encoder.

- 4 nút bấm phục vụ việc nhận lệnh từ người sử dụng.

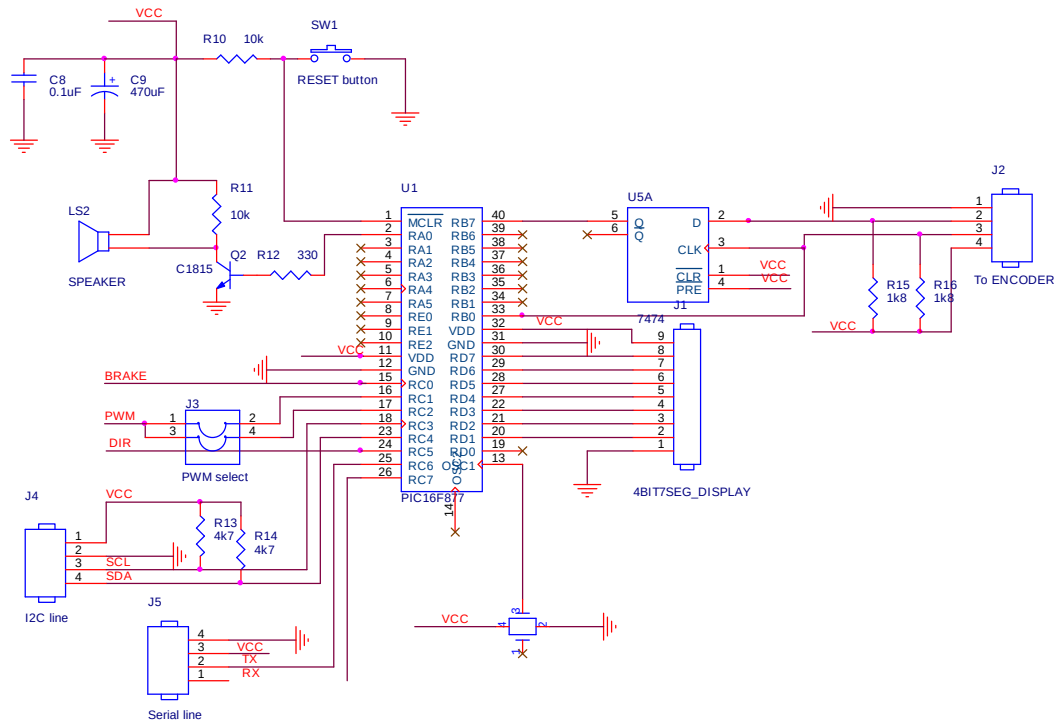
- 1 loa dành để báo hiệu các giai đoạn trong chương trình.



Hình 3.11 Hình chụp module master

3.6.2 Mạch module slave

Gồm 2 khối: khối xử lý chính và khối khuếch đại công suất.



Hình 3.12 Sơ đồ nguyên lý khối xử lý chính của module slave

Các thành phần trong khối xử lý chính:

- Vi điều khiển PIC 16F877: bộ xử lý của module.

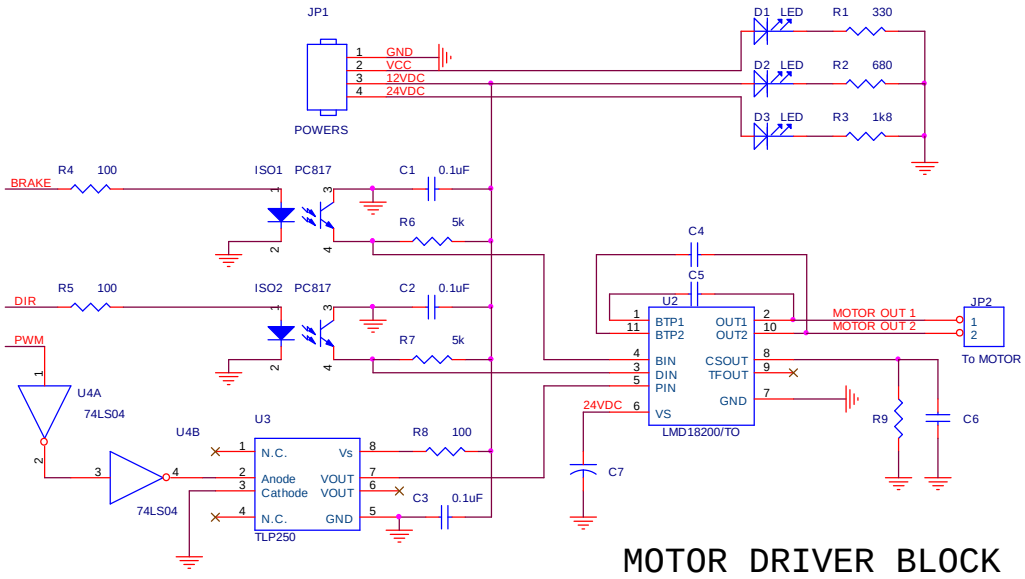
- Ngõ giao tiếp nối tiếp: để giao tiếp với mạch nạp và chương trình kiểm tra trên máy tính.

- Ngõ giao tiếp I2C: để giao tiếp với vi điều khiển của module master.

- Ngõ nối với encoder để đọc tín hiệu hồi tiếp, có một IC flip-flop để đệm cho tín hiệu từ encoder.

- 1 loa dành để báo hiệu.

Khối khuếch đại công suất:



Hình 3.13 Sơ đồ nguyên lý khối khuếch đại công suất của module slave

Các thành phần trong khối khuếch đại công suất:

- Chip điều khiển động cơ LMD18200.

- 3 opto để cách ly các ngõ vào của chip LMD18200, bảo vệ phần mạch phía trước.

- Các cổng logic để nắn tín hiệu cho ngõ vào PWM.

Hình 3.14 Hình chụp module slave

4
THỰC HIỆN BỘ ĐIỀU KHIỂN VÀ KIỂM CHỨNG GIẢI THUẬT

4.1 Sơ đồ giải thuật chương trình

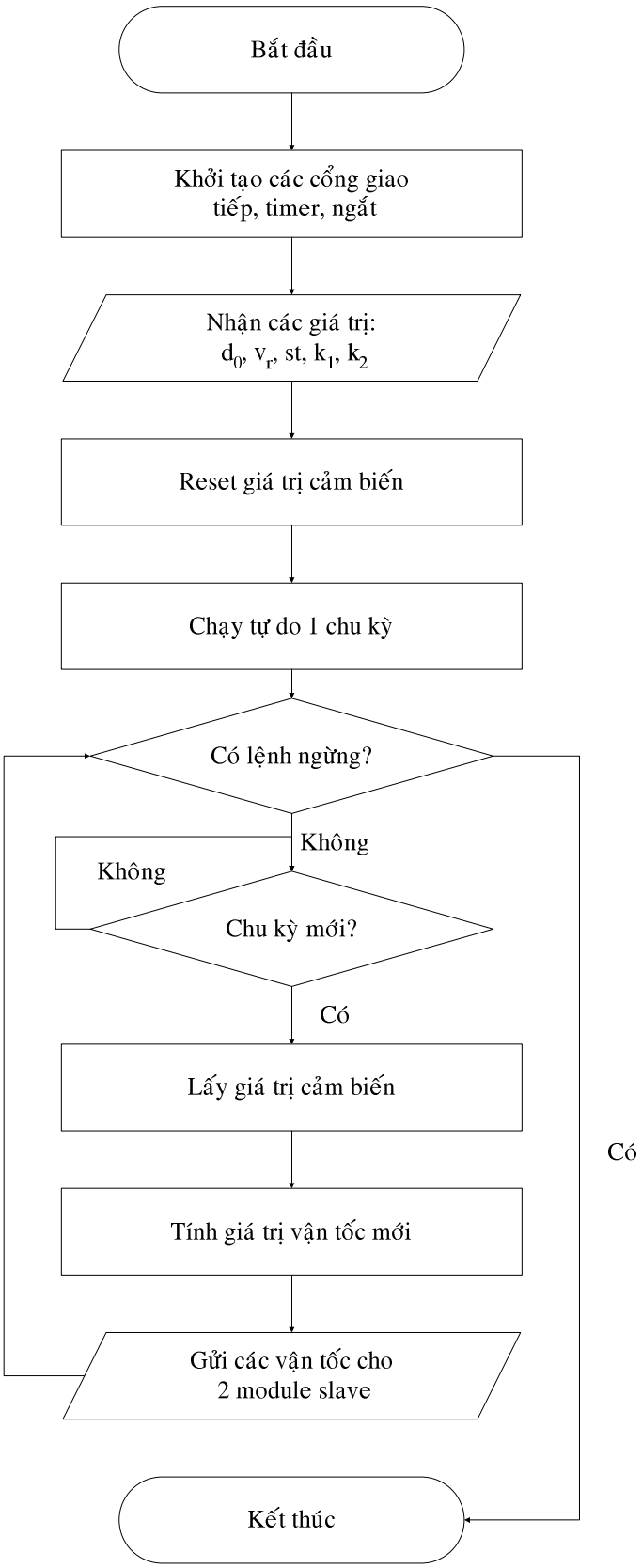
Giải thuật cho robot di động theo tường được thực hiện nhờ 2 chương trình, một chương trình dành cho master module (chương trình chính) và chương trình kia dành cho slave module (chương trình phụ).

Chương trình chính có nhiệm vụ là bộ điều khiển full-state feedback của toàn hệ thống, chương trình phụ có nhiệm vụ là bộ điều khiển PID vận tốc cho mỗi bánh xe. Chức năng cụ thể của từng chương trình được đề cập ở phần 3.1.

Các chương trình được lập trình bằng ngôn ngữ C, biên dịch cho các vi điều khiển PIC bằng trình biên dịch PIC-C, sau đó nạp vào các vi điều khiển (nạp chương trình chính vào vi điều khiển ở master module, nạp chương trình phụ vào 2 vi điều khiển ở 2 slave module).

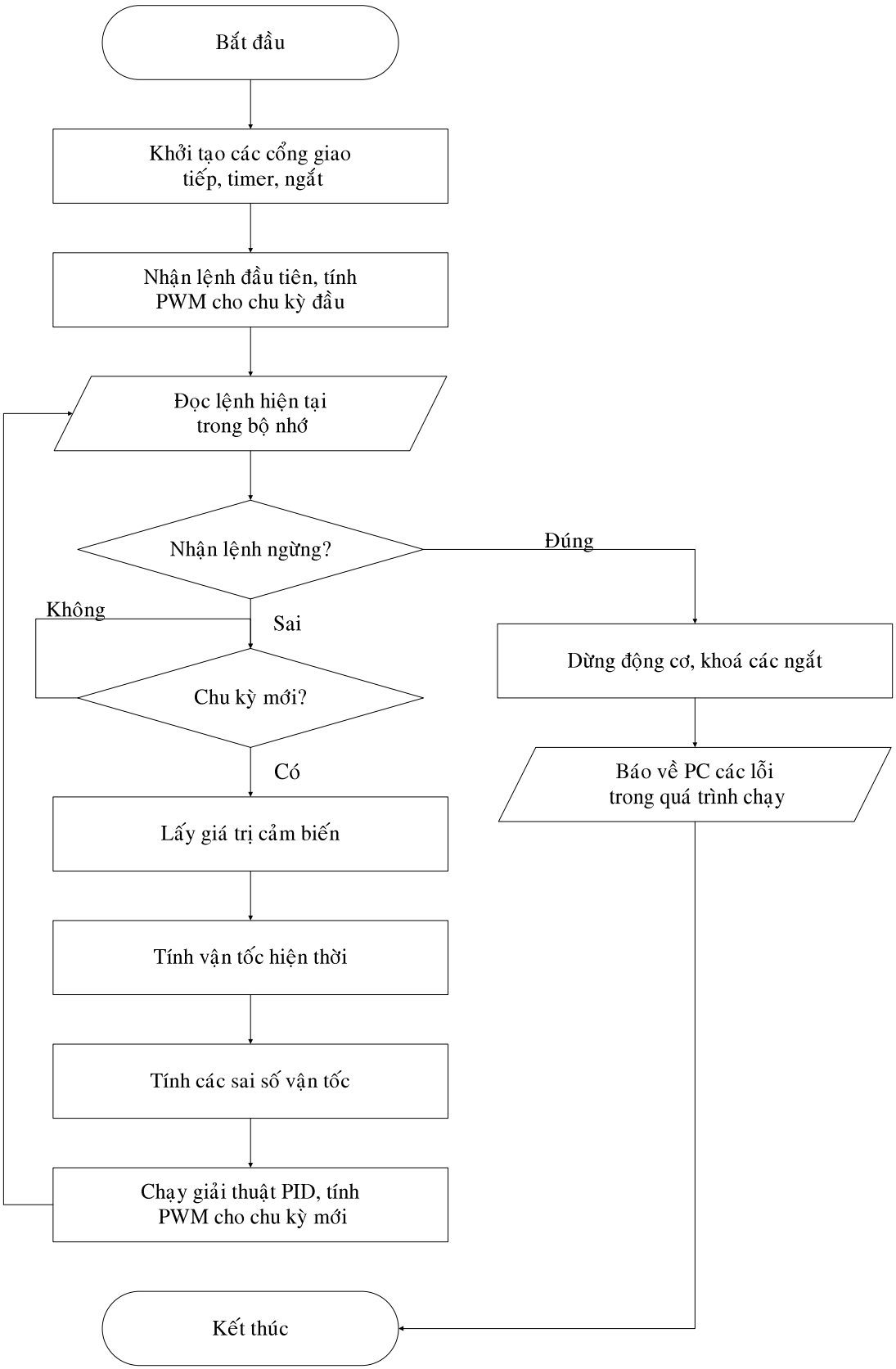
Mã nguồn của các chương trình: xem phụ lục B.

4.1.1 Giải thuật cho master module



Hình 4.1 Lưu đồ giải thuật của master module

4.1.2 Giải thuật cho slave module



Hình 4.2 Lưu đồ giải thuật của slave module

4.2 Tiến hành thí nghiệm

Các thí nghiệm được thực hiện như sau:

* Môi trường hoạt động: gồm tường và sàn nhà. Tường cần được lót một lớp đệm để có bề mặt êm và trơn. Bề mặt trơn để giảm lực ma sát tác động lên robot thông qua cảm biến tiếp xúc, bề mặt cũng cần êm để giảm rung động cho cảm biến, vì theo quan sát, sự rung động ở cảm biến sẽ gây ra sai số đọc encoder. Sàn nhà cũng được lót đệm để tăng hệ số ma sát giữa bánh xe với sàn, và cũng nhằm giảm rung động cho robot.

* Robot di động: robot được nuôi bằng nguồn điện ở ngoài. Thông qua các thiết bị biến thế, ổn áp, từ nguồn điện xoay chiều 220V/50Hz ta có được các nguồn một chiều (5V, 12V và 24V) để nuôi robot. Trên module master của robot có cắm dây nối với máy tính qua cổng giao tiếp nối tiếp, để robot truyền các kết quả đo được về máy tính.

* Máy tính: chúng tôi dùng một máy tính laptop có cổng giao tiếp nối tiếp (chuẩn RS-232) để vừa ra lệnh cho robot, vừa quan sát các trạng thái của robot. Máy tính có một chương trình dạng "terminal" để gửi lệnh và nhận dữ liệu qua cổng nối tiếp, dữ liệu nhận được sẽ được lưu thành file kết quả, file này sẽ được xử lý bằng phần mềm Matlab để thể hiện kết quả thí nghiệm qua các đồ thị.

Tiến trình thực hiện một thí nghiệm gồm các bước:

- Bật nguồn điện, reset các vi điều khiển trên robot, cài đặt thông số cho chương trình terminal trên máy tính. Cài đặt vị trí chuẩn của cảm biến, sau đó dịch chuyển robot để tạo độ lệch ban đầu cho cảm biến.

- Truyền các tham số cho module master của robot, các thông số được truyền theo thứ tự gồm: khoảng cách mong muốn d0, vận tốc mong muốn vr, thời gian lấy mẫu st, các tham số của bộ điều khiển k1 và k2.

- Đợi robot di chuyển đến hết đoạn đường cần thí nghiệm, ngừng robot và lưu kết quả thí nghiệm.

- Chạy phần mềm Matlab để xử lý kết quả thí nghiệm và nhận xét.

Do bị giới hạn về thời gian nghiên cứu, các thí nghiệm trên tường cong chưa được thực hiện. Chúng tôi chỉ xin trình bày ở đây các thí nghiệm robot di chuyển theo tường thẳng.

Hình 4.3 Mô hình thí nghiệm

4.3 So sánh các kết quả mô phỏng và thí nghiệm

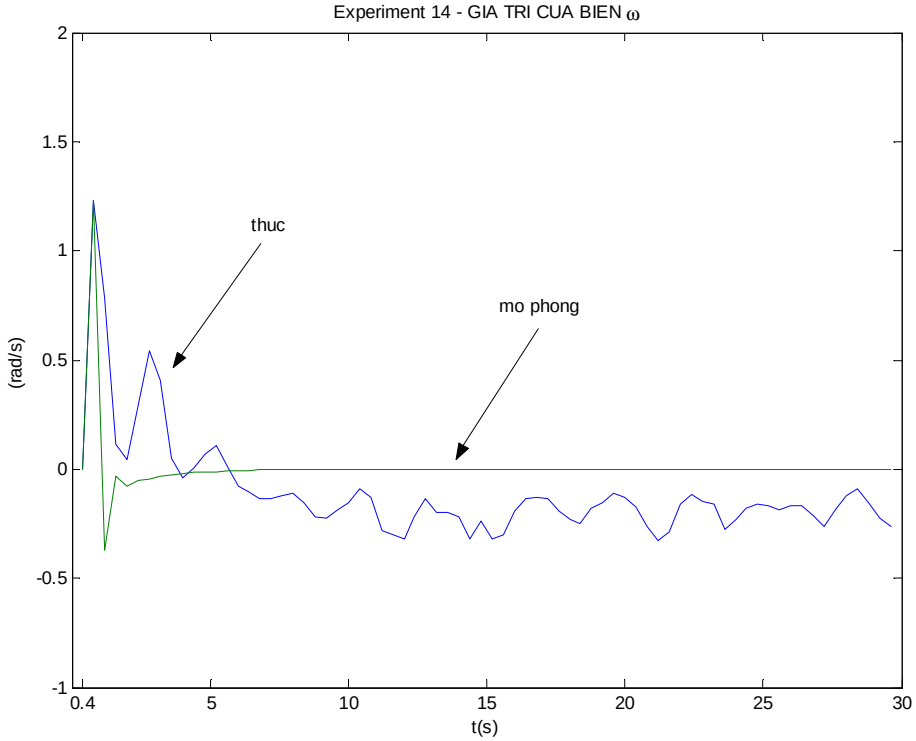
4.3.1 So sánh kết quả mô phỏng bằng Matlab với kết quả thí nghiệm

Chúng tôi tiến hành rất nhiều thí nghiệm với tường thẳng, sau đây là một số thí nghiệm cho kết quả điển hình và các nhận xét.

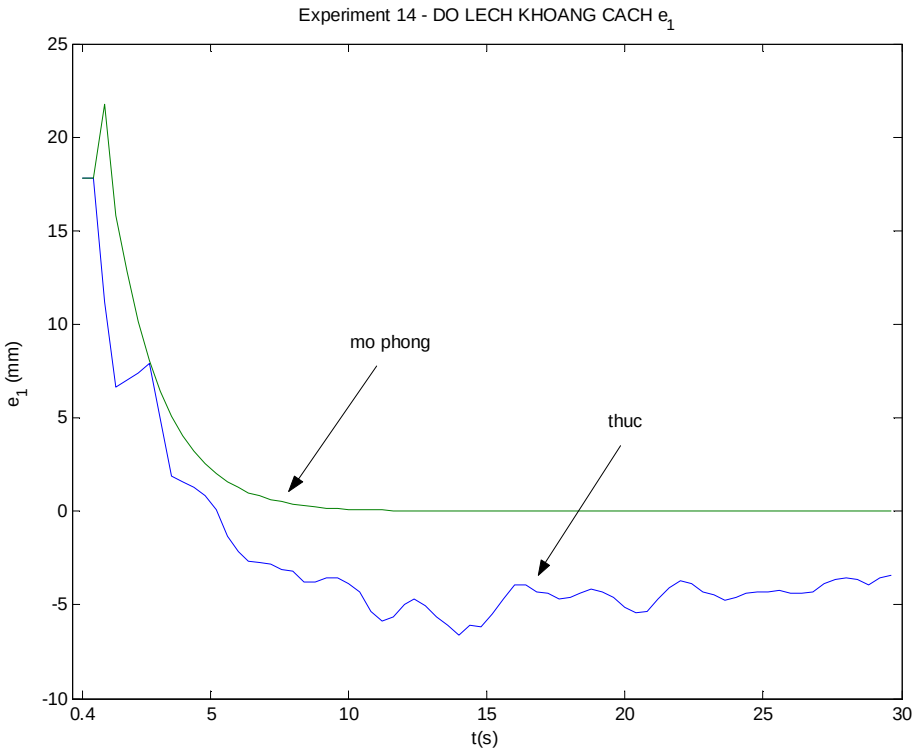
Bảng 4.1 Thông số thí nghiệm

Tham số	*Giá trị*	*Đơn vị*	*Ghi chú*
d_0	0.25	m	d_0 là giống nhau trong tất cả các thí nghiệm
v_r	0.05	m/s	
st	0.4	s	st khá lớn, do bộ điều khiển động cơ đáp ứng chậm
k_1	3.5		
k_2	615		
e_1 ban đầu	0.0178	m	Tính ngược từ giá trị hai encoder ở cảm biến
e_2 ban đầu	-0.1955	rad	Tính ngược từ giá trị hai encoder ở cảm biến

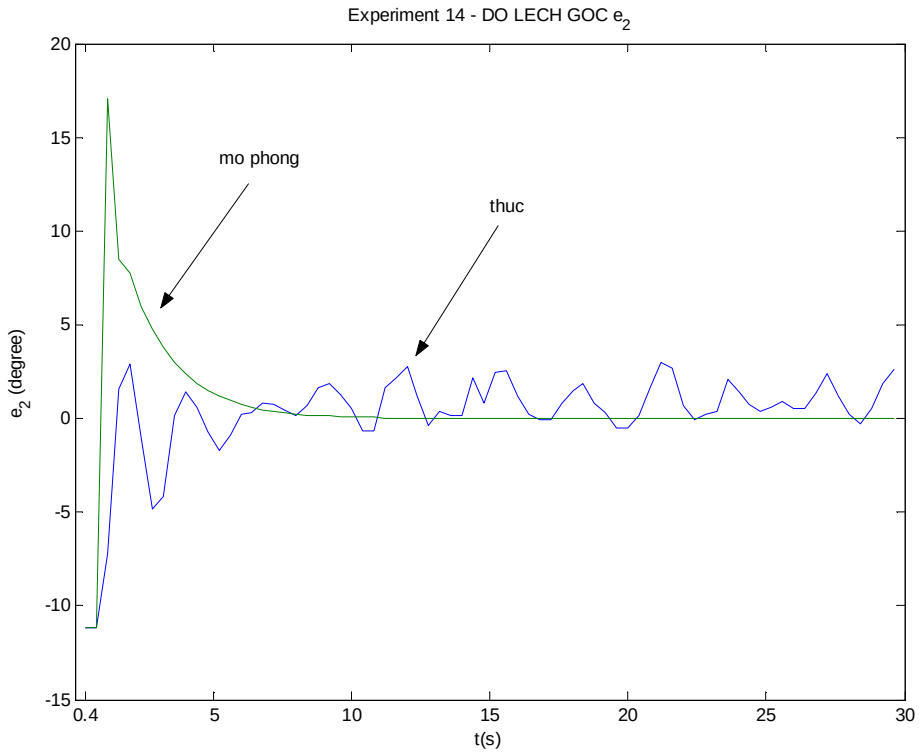
Đồ thị so sánh giữa kết quả mô phỏng và kết quả thí nghiệm:



Hình 4.4 So sánh đồ thị của vận tốc robot



Hình 4.5 So sánh đồ thị của sai số khoảng cách



Hình 4.6 So sánh đồ thị của sai số góc

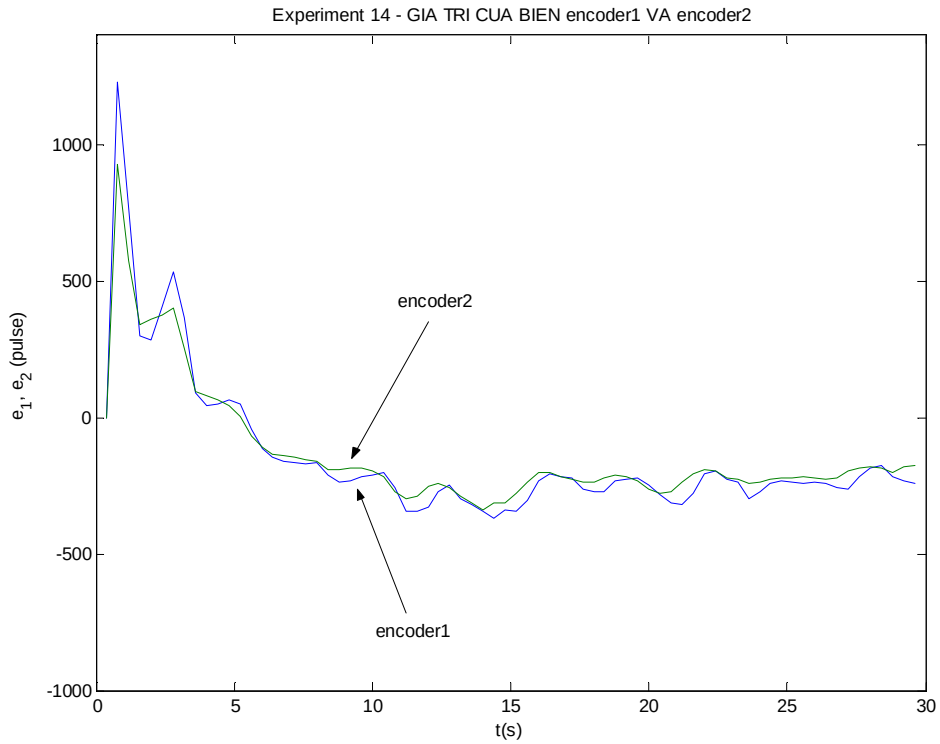
Nhận xét 1: Kết quả thực có đồ thị khá giống dạng đồ thị của kết quả mô phỏng, điều này cho thấy việc mô phỏng đã chỉ đường đúng đắn cho thực nghiệm.

Nhận xét 2: Kết quả thực khá xấu so với kết quả mô phỏng. Điều này cũng dễ hiểu vì robot thực của chúng ta chịu nhiều tác động của các sai số khi chế tạo, các sai số do linh kiện phần cứng và các sai số nhiễu, trong khi việc mô phỏng được thực hiện trong môi trường lý tưởng.

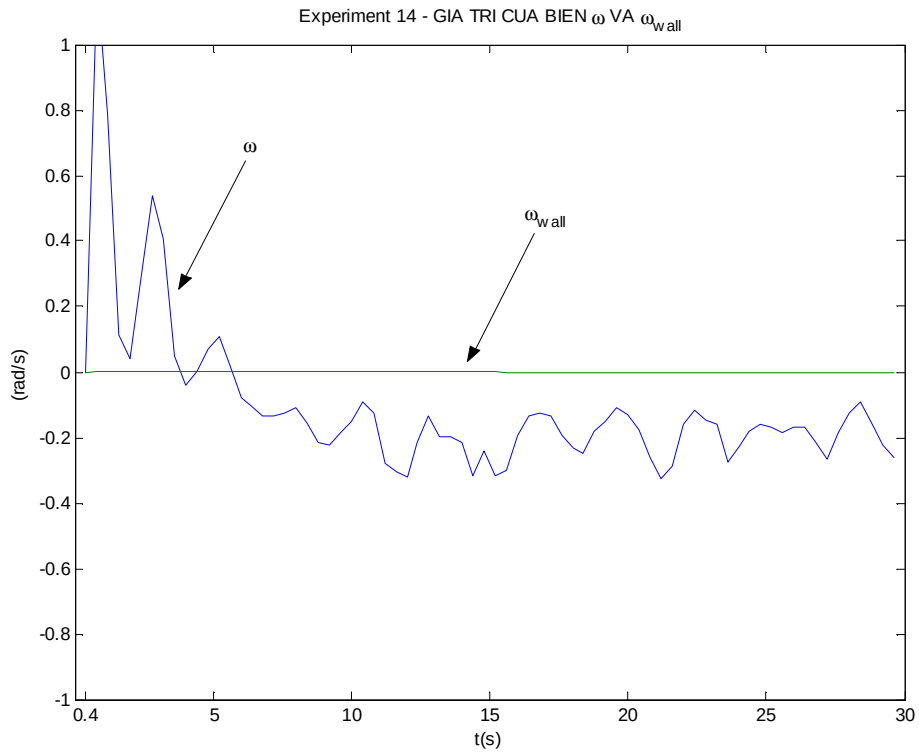
Nhận xét 3: Bộ điều khiển là hội tụ. Thời gian hội tụ là khoảng 10s theo mô phỏng và khoảng 15s theo kết quả thí nghiệm (ta đánh giá giá trị thời gian này là dựa vào nhận xét rằng từ thời điểm 15s, các giá trị e1 và e2 không thay đổi nhiều, chúng dao động ở gần điểm cân bằng).

4.3.2 Các nhận xét bổ sung

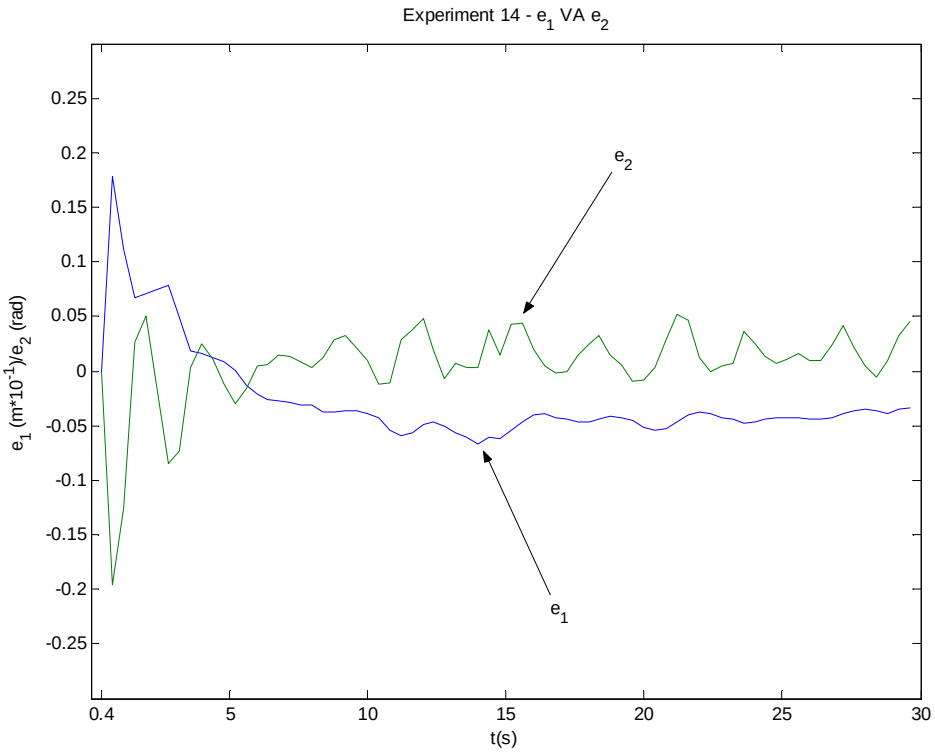
Ta xét thêm các đồ thị kết quả của thí nghiệm đã nêu ở trên:



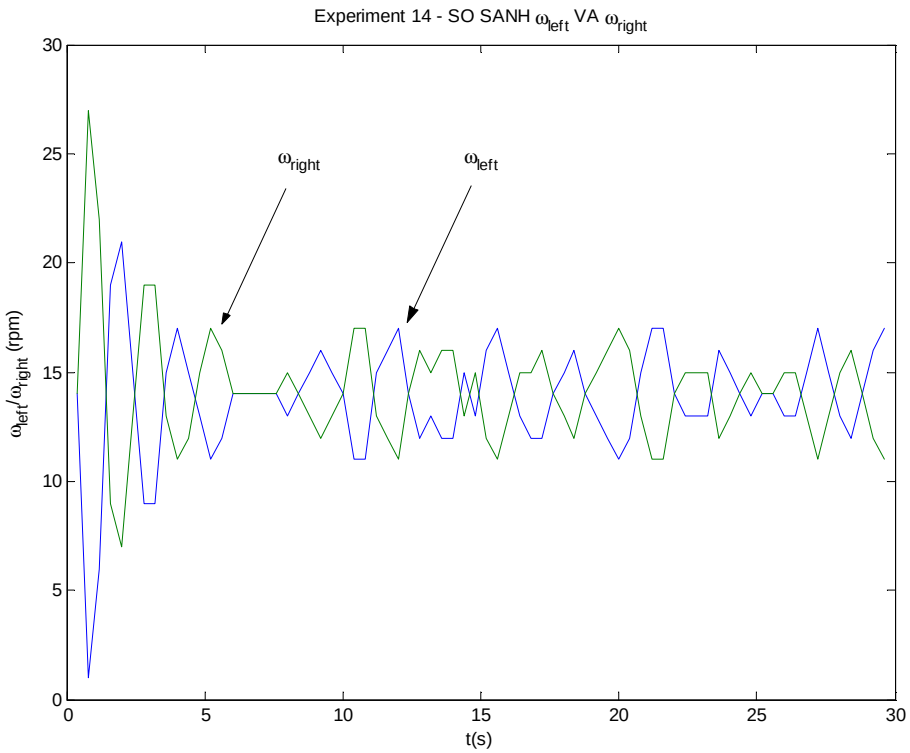
Hình 4.7 Giá trị của cảm biến



Hình 4.8 Giá trị vận tốc góc của robot và vận tốc góc (ước lượng) của tường



Hình 4.9 Biến đổi của các sai lệch trong quá trình hoạt động



Hình 4.10 Giá trị vận tốc ra lệnh cho 2 bánh xe

Nhận xét 4: Bộ điều khiển là ổn định với độ lệch ban đầu nằm trong vùng lân cận điểm cân bằng. (thật ra thí nghiệm được nêu trên đây có độ lệch ban đầu nằm gần biên của vùng lân cận đó).

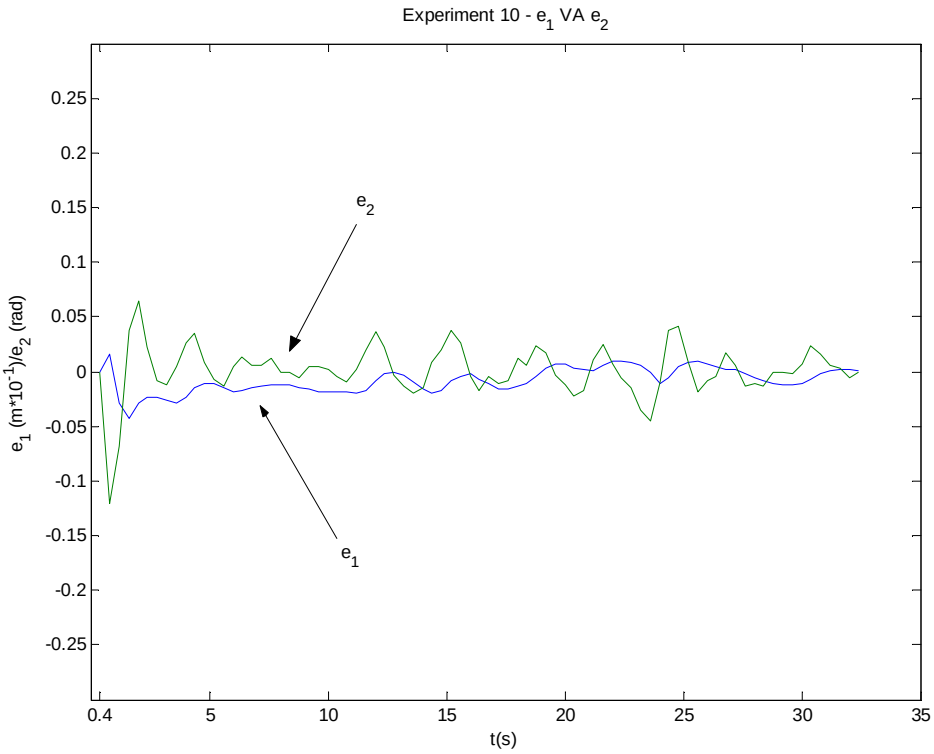
Hai thí nghiệm sau đây sẽ cho thấy tốc độ hội tụ của 2 sai số khoảng cách và góc:

Bảng 4.2 Thông số của 2 thí nghiệm (TN) dùng để so sánh

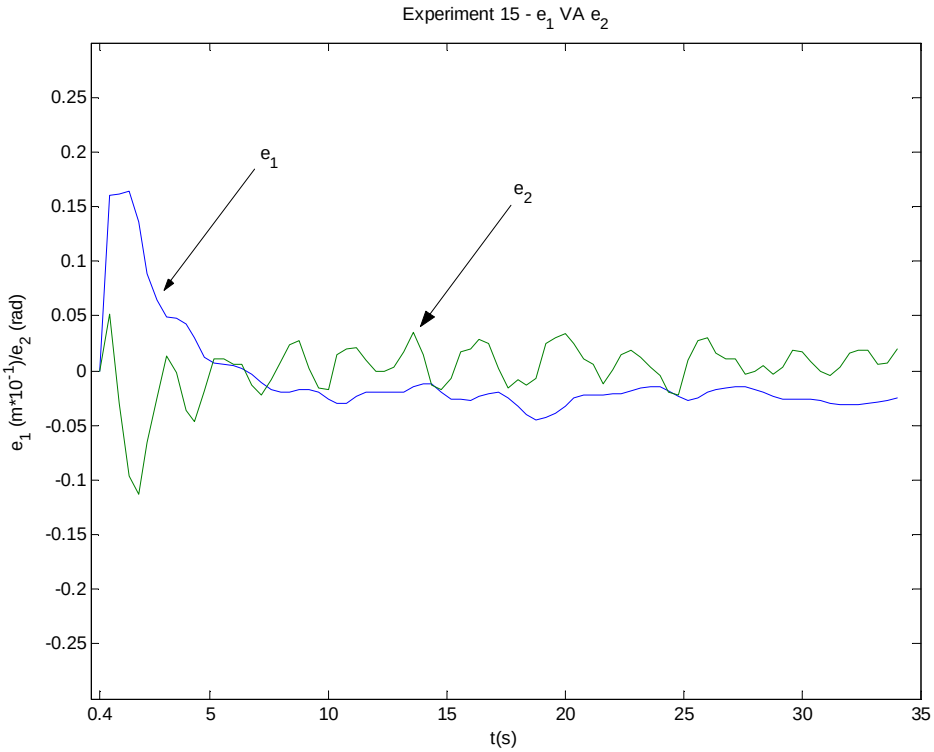
Tham số	*Giá trị*		*Đơn vị*	*Ghi chú*
	TN1	TN2		
d_0	0.25	0.25	m	
v_r	0.05	0.05	m/s	
st	0.4	0.4	s	
k_1	3.5	3.5		
k_2	615	615		
e_1 ban đầu	0.0017	0.0160	m	TH1: e_1 ban đầu rất nhỏ; TH2: e_1 ban đầu lớn
e_2 ban đầu	-0.1203	0.0516	m	TH1: e_2 ban đầu lớn; TH2: e_2 ban đầu rất nhỏ

Các đồ thị kết quả được cho ở hình 4.11.

Nhận xét 5: Sai số về góc (e_2) hội tụ nhanh hơn sai số về khoảng cách (e_1). Khi robot nhận e_1 ban đầu lớn, thời gian hội tụ chung của cả 2 sai số sẽ bị kéo dài hơn so với trường hợp e_2 ban đầu lớn.



a. Đồ thị e_1 và e_2 của TN1



b. Đồ thị e_1 và e_2 của TN2

Hình 4.11 So sánh các đồ thị e_1 và e_2 của hai thí nghiệm

5
KẾT LUẬN

5.1 Độ thích hợp của giải thuật

Luận văn này đã kiểm nghiệm được bộ điều khiển cho robot di động bám tường. Bộ điều khiển được dùng là bộ điều khiển hồi tiếp tất cả trạng thái (full-state feedback controller). Kết quả thí nghiệm đã kiểm chứng tính hội tụ và ổn định Lyapunov của bộ điều khiển.

5.2 Những hạn chế của đề tài

5.2.1 Về việc chế tạo phần cứng

Do chưa có nhiều kinh nghiệm trong việc gia công cơ khí, chúng tôi đã phải mất rất nhiều thời gian cho việc chế tạo phần cơ của robot. Tuy việc chế tạo còn nhiều sai số, robot cũng đạt điều kiện vừa đủ để phục vụ cho việc thí nghiệm. Khuyết điểm lớn nhất ở robot này là sử dụng các động cơ cũ (2 động cơ mua ở chợ động cơ cũ trên đường Vĩnh Viễn, Q.10, Tp.HCM), với các thông số không đảm bảo và mômen tải nhỏ, động cơ có đáp ứng chậm kéo theo thời gian lấy mẫu của hệ thống phải lớn, làm cho kết quả thí nghiệm không tốt như mong muốn.

5.2.2 Những hiện tượng ảnh hưởng đến kết quả và cách khắc phục

Trong quá trình thực hiện đề tài, chúng tôi quan sát thấy một số hiện tượng có thể gây sai lệch ở kết quả. Nhằm giúp các bạn sinh viên đi sau tiết kiệm thời gian, chúng tôi xin trình bày các hiện tượng đó và đề xuất cách khắc phục.

Bảng 5.1 Các hiện tượng ảnh hưởng đến kết quả và cách khắc phục

Hiện tượng	*Cách khắc phục*
Độ rơ của các khớp nối giữa trục động cơ (qua hộp giảm tốc) và bánh xe, bán kính 2 bánh dẫn động không bằng nhau. Hậu quả: bộ điều khiển PID hoạt động không tốt.	Gia công bánh xe bằng kim loại (nên dùng nhôm), cố gắng giảm sai số khi khoan lỗ trục bánh xe.

Rung động ở các thanh trượt và encoder trên cảm biến sẽ gây ra sai số của cảm biến, sai số tích luỹ trong thời gian hoạt động của robot sẽ đủ lớn để dẫn đến kết quả xấu ở bộ điều khiển.	Gá chặt encoder vào cảm biến; sử dụng các thanh trượt chính xác; giảm rung động cho robot bằng việc tạo ra tường và sàn êm.
Chương trình điều khiển lớn, khi biên dịch bằng PICC có thể không hoạt động đúng (do lỗi của trình biên dịch)	Tiết kiệm bộ nhớ chương trình bằng cách hạn chế dùng biến số thực và các lệnh xuất/nhập qua cổng nối tiếp, nên giới hạn chương trình trong khoảng <80% ROM.
Kết quả mô phỏng có thể tốt với nhiều bộ (k1,k2), nhưng khi đưa vào robot thực thì chỉ có một số bộ (k1,k2) thích hợp.	Các tham số k1 và k2 của bộ điều khiển cần phải dò lại trong khi thí nghiệm bằng cách điều chỉnh sau mỗi thí nghiệm.

5.3 Hướng nghiên cứu tiếp

Để tiếp tục đề tài này cho đến hết nhiệm vụ kiểm nghiệm giải thuật, chúng tôi xin đề nghị hướng nghiên cứu tiếp như sau: cải tiến phần cơ của robot để giảm các sai số chế tạo, thay các động cơ hiện có bằng 2 động cơ mới với bộ truyền động có tỉ số truyền lớn, đồng thời gắn encoder có độ phân giải lớn hơn vào động cơ để tăng khả năng điều khiển vận tốc; cố gắng giảm thời gian lấy mẫu của chương trình chính; lập trình bộ điều khiển dùng bộ quan sát để so sánh tốc độ hội tụ giữa việc dùng bộ điều khiển này với bộ điều khiển hồi tiếp tất cả trạng thái đã thực hiện trong đề tài.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

[1] Trần Hữu Quế, Vẽ Kĩ Thuật Cơ Khí, Tập 1&2, Nhà xuất bản Giáo Dục, **2000**

[2] Nguyễn Doãn Phước, Phan Xuân Minh & Hán ThànhTrung, Lý Thuyết Điều *Khiển Phi Tuyến, Nhà xuất bản Khoa Học và Kỹ Thuật, 2003,* trang 53-75.

[3] Nguyễn Viết Hiệp & Phạm Đình Anh Vũ, Mô hình hoá – mô phỏng và điều *khiển Robot theo dấu tường, Luận văn tốt nghiệp Đại học,* Đại học Bách Khoa Tp.HCM, 2004.

Tiếng nước ngoài:

[4] Detriche, Jean-Marie, *Systemes robotiques et Mecatroniques*, Cours d'Ecole Centrale Paris, **1999-2000**.

[5] Lagoudakis, Michail G., *Mobile Robot Local Navigation with a Polar Neural Map*, MSc Thesis, University of Southwestern Louisiana, **1998**.

[6] P. van Turennout, G. Honderd, L.J. van Schelven,*Wall following control of a Mobile Robot,* International Conference on Robotics and Automation, Nice, France, **1992**, p. 280-285.

[7] Medromi, J.Y. Tigli, and M.C. Thomas, *Posture Estimation of a Mobile Robot: Observers-Sensors*, Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, p. 661-666, **1994**.

[8] Urzelai J., Uribe J.P., and Ezkerra M., *Fuzzy Controller for Wall-Following With a non-holonomous Mobile Robot*, Proceedings of the 6th IEEE International Conference on Fuzzy System, Vol. 3, p.1361-1368, **1997**.

[9] A. Bemporad, M.D. Marco, and A. Tesi, *Wall-Following Controllers for a Sonar Mobile Robot*, Proceedings of the 36th Conference on Decision & Control, California USA, p. 3063-3068, **1997**.

[10] T. Yata, L. Kleeman, and S. Yuta, *Wall Following Using Angle Information Measured by a Single Ultrasonic Transducer*, Proceedings of the 1998 IEEE International Conference on Robotics and Automation, p1590-1596, **1998**.

[11] Chung Tan Lam, *A nonlinear Feedback Control of Wall-Following Mobile Robot*, Ms.C Thesis, Pukyong National University, Korea, **2004**.

[12] Rodney A. Brooks, A robust layered control system for a mobile robot, *Research report of Massachusette Institute of Technology*, **1985**.

[13] http://www.microchip.com

[14] **http://www.semiconductors.philips.com/buses/i2c/**

[15] John A. Shaw, *The PID Control Algorithm - How it works, how to tune it, and how to use it, 2nd edition*, (ebook), **2003**.

PHỤ LỤC A
SƠ LƯỢC VỀ CHUẨN GIAO TIẾP I2C [14]

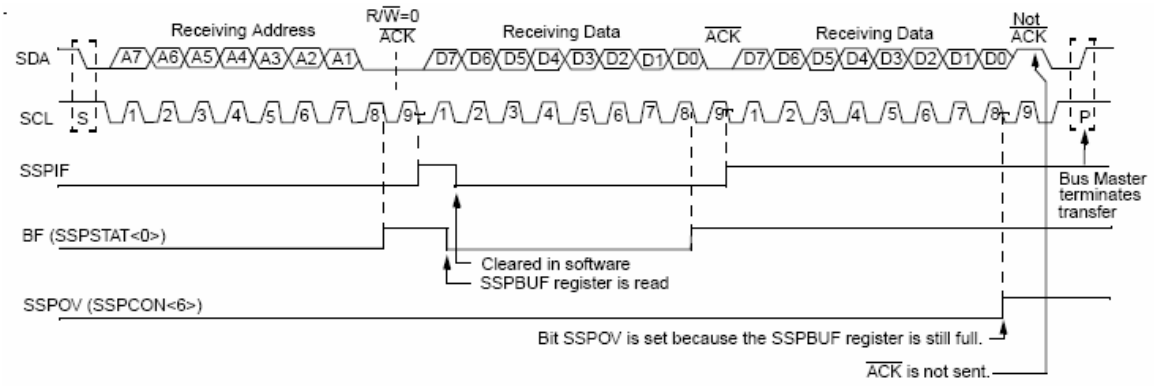
I2C là chuẩn giao tiếp ban đầu được hãng Philips phát triển để phục vụ cho các giao tiếp với Tivi, sau đó, do những lợi thế của chuẩn giao tiếp này nên nó đã trở nên rất phổ biến trong công nghiệp. Các tài liệu về chuẩn I2C được cung cấp rất đầy đủ và chi tiết tại trang web của hãng Philips.

Microchip đã tích hợp chuẩn giao tiếp này vào phần cứng cho một số vi điều khiển, trong đó có PIC16F877. Chuẩn này được biết dưới dạng chuẩn giao tiếp đồng bộ SSP. Trong chuẩn giao tiếp SSP có hai chuẩn con là chuẩn SPI và I2C. SPI là chuẩn giao tiếp nối tiếp dùng 1 dây nối, và I2C là chuẩn giao tiếp nối tiếp 2 dây. Dây SDA là dây để truyền dữ liệu và dây SCL là dây để giữ nhịp cho dữ liệu truyền.

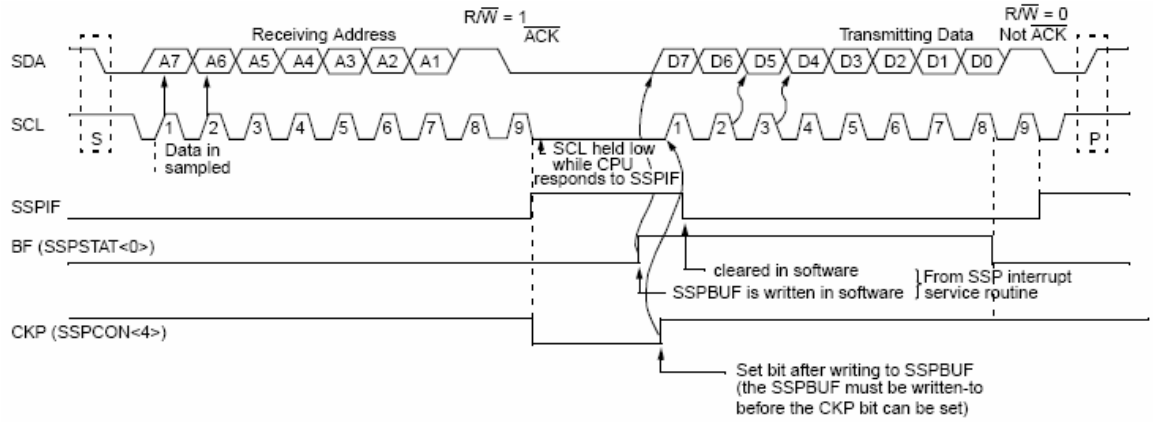
Chuẩn I2C là chuẩn giao tiếp trong đó dùng một thiết bị làm Master, và các thiết bị khác trong mạng là Slave trong một thời điểm nhất định. Tại một thời điểm, Master có quyền đọc và xuất dữ liệu qua tất cả các Slave thông qua địa chỉ của Slave đó.

Master bắt đầu quá trình đọc hoặc ghi dữ liệu vào một Slave bằng cách đặt tín hiệu Start (S) vào đường truyền (hình A.1 và A.2). Byte tiếp theo được gửi đi là byte địa chỉ của Slave cần giao tiếp. Quá trình truyền nhận được kết thúc bằng bit Stop (P).

Sau đây là biểu đồ thời gian truyền và nhận trong chuẩn giao tiếp I2C



Hình A.1: Biểu đồ nhận dữ liệu, địa chỉ 7 bit



Hình A.2: Biểu đồ truyền dữ liệu, địa chỉ 7 bit

I2C phần cứng của PIC 16F877 hỗ trợ hai chế độ định địa chỉ 10 bit và 7 bit. Tuy nhiên, chúng ta vẫn chỉ thường sử dụng địa chỉ 7 bit, hỗ trợ giao tiếp 128 thiết bị. Tuy vậy, chuẩn I2C cũng giống như chuẩn RS232, khi thao tác với các vi điều khiển, có thể được thiết lập bằng phần mềm.

Một lần truyền địa chỉ, sẽ có 8 bit, bit thấp nhất là bit xác định chế độ đọc hoặc ghi (R/W) (hình A.2).

Chuẩn I2C rất dễ sử dụng, có các tốc độ truyền nhận là 100Kbps, 400Kbps và 1Mbps, vi điều khiển PIC 16F877 hỗ trợ tốc độ 100Kbps và 400Kbps, nhanh hơn nhiều lần so với chuẩn RS232. Ngoài ra, không cần dùng bất kỳ thiết bị chuyển đổi nào để chuyển đổi điện áp tín hiệu, do vậy, chuẩn I2C thích hợp nhất cho các giao tiếp trong phạm vi ngắn (dưới 1m) giữa các vi điều khiển.

PHỤ LỤC B
MÃ NGUỒN CÁC CHƯƠNG TRÌNH

Chương trình cho master module:

```
/************************
Description: This sourcecode is for the master module of the thesis
"Study on Control of Wall-Following Mobile Robot"
by Doan Minh Dang - P9900012
Start date: 2004.05.27
End date: 2004.07.03
************************/
#include <16f877.h>
#device PIC16F877 *=16 ADC=10
#include <string.h>
#include <stdlib.h>
#include <math.h>
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6, rcv=PIN_C7)
#use I2C(master, sda=PIN_C4, scl=PIN_C3)
#use fast_IO(E)
#priority ccp1,ccp2,rb,timer1
//address
#define address_left 0xa0
#define address_right 0xc0
//constant
float vr=0.05,d0=0,k1=2.5,k2=250;//unit: [vr]=m/s,[d0]=m
```

```
//variables
int1 vr_ok=0;
char s[5];
int1 stop1=0;
int1 stop2=0;
int1 online=0;//operating flag
//int16 temp=0; for testing
signed int16 encoder_1=0;//supplemental encoder in primitive unit, [encoder_1]=pulse
signed int16 encoder_2=0;//main encoder in primitive unit, [encoder_2]=pulse
//signed int16 encoder_1_old,encoder_2_old;for filtering
signed int16 temp_int16=0;
//float encoder_1_m=0;//d2, [encoder_1_m]=m
//float encoder_2_m=0;//d1, [encoder_2_m]=m
float d1,d2;//[d1]=m,[d2]=m
float e1,e2,omega,delta_omega_mu,part1,omega_left,omega_right;
float omega_mu=0;
//float omega_mu_temp=0;
//[e1]=mm
//[e2]=rad
//[omega]=rad
//[omega_mu]=rad
//[delta_omega_mu]=rad/(T*ms) (T=150)
//[omega_left]=[omega_right]=round/minute (rpm)
signed int16 omega_left_new,omega_right_new;
//[omega_left_new]=[omega_right_new]=round/minute
byte last_b;
int1 count=0;
```

```
int cycle_count=0;
int1 new_cycle=0;
int number_cycle=2;
signed int16 omega_left_last=0;
//[omega_left_last]=round/minute
signed int16 omega_right_last=0;
//[omega_right_last]=round/minute

void tilt_c5(int16 speak_time)
{//master board will speak in specified time(ms)
  output_high(PIN_C5);
  delay_ms(speak_time);
  output_low(PIN_C5);
}
#SEPARATE
void send_commands_via_i2c(int first_add,signed int16 first_data,int second_add,signed int16 second_data)
{
int hi,lo;
//send command to first slave
  hi=make8(first_data,1);
  lo=make8(first_data,0);
      i2c_start();
      i2c_write(first_add);
      delay_ms(1);
      i2c_write(lo);//send low byte of velocity
      delay_ms(1);
      i2c_write(hi);//then send high byte
      i2c_stop();
```

```
//send command to second slave
  hi=make8(second_data,1);
  lo=make8(second_data,0);
      delay_ms(1);
      i2c_start();
      i2c_write(second_add);
      delay_ms(1);
      i2c_write(lo);
      delay_ms(1);
      i2c_write(hi);
      i2c_stop();
}

void calculate_slave_velocities()
{
//input: omega_left, omega_right - float, global variables
//output: omega_left_new,omega_right_new - signed int16, global variables
//This part is moved to a function to reduce memory cost
      //calculate new left and right velocities
      omega_left=(vr-omega*0.095)/0.033;
      omega_right=(vr+omega*0.095)/0.033;
      //send new omega to left and right modules
      omega_left_new=(signed int16)(omega_left*9.55);
//convert from rad/s to rpm: (rad/s)*60/2pi=rpm
      omega_right_new=(signed int16)(omega_right*9.55);
}

#INT_RB
```

```
void rb_isr()
{
  byte changes,new_b;
   new_b=input_b();
  changes = last_b ^ new_b;
  last_b = new_b;
    if (bit_test(changes,5))//RB5: start button
    {
      //b5 went low
      stop1=!stop1;//if it is set, it will be clear, and vice versa
    }
    if (bit_test(changes,4))//RB4: stop button
    {
      //b4 went low
      if (stop1) stop2=1;
    }
  delay_ms(200);  //debounce
}
#INT_CCP1
void vantoc1()
{
   if (input(PIN_E1)) encoder_1++;
   else encoder_1--;
}
#INT_CCP2
void vantoc2()
{
    if (input(PIN_E2)) encoder_2--;
    else encoder_2++;
```

```
}
#INT_TIMER1
void timer1_isr()
{
  set_timer1(34286);
  cycle_count++;
   if (cycle_count==number_cycle)
   {
    cycle_count=0;
    new_cycle=1;
   }
}
void main()
{
//INIT
  tilt_c5(500);
  set_tris_e(0x07);
  set_tris_b(0b00110000);
  last_b=input_b();
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  setup_CCP1(CCP_CAPTURE_FE);
  setup_CCP2(CCP_CAPTURE_FE);
  enable_interrupts(INT_CCP1);
  enable_interrupts(INT_CCP2);
  enable_interrupts(INT_RB);
  enable_interrupts(GLOBAL);
//RECEIVE PARAMETERS
  //vr
  gets(s);
```

```
        vr=atof(s);
        printf("Receive velocity desired: %f \n",vr);
        gets(s);
        number_cycle=atoi(s);
        printf("Sampling time=%u*0.05s \n",number_cycle);
        encoder_1=0;
        encoder_2=0;
        tilt_c5(1000);
        gets(s);
        k1=atof(s);
        printf("k1=%f",k1);
        gets(s);
        k2=atof(s);
        printf("k2=%f",k2);
        //first move: move straight forward with at speed=vr
        omega=0;
        calculate_slave_velocities();
        printf("Go!\n");
        send_commands_via_i2c
(address_left,omega_left_new,address_right,omega_right_new);
        omega_left_last=omega_left_new;
        omega_right_last=omega_right_new;
        printf("l:%ld\nr:%ld\n",omega_left_new,omega_right_new);
        new_cycle=0;
        //up to here, it cost 5146 timer1 counts=8233us
        //it will ring 500-8=492ms
        tilt_c5(492);
        set_timer1(34286);//overflow after 3250 machine
cycles=50ms
```

```
  enable_interrupts(INT_TIMER1);


//MAIN LOOP
   while ((!stop1)&&(!stop2))
   {
      if (new_cycle)
      {
   //start new cycle
//GET d1 AND d2
         d2=(float)encoder_1/51000.0;//scale: 51000 pulse/m
      d1=(float)encoder_2/51000.0;
//SEND ENCODERS' VALUES TO PC
         printf("1:%ld\n2:%ld\n",encoder_1,encoder_2);
//CALCULATE e1 & e2
         e2=atan((d1-d2)/0.030);
      e1=-d1*cos(e2);
         //calculate omega
      part1=vr-(e1+d0)*omega_mu/cos(e2);
      omega=k2*part1*e1-k1*sin(e2)+omega_mu;
         //calculate new omega_mu
      delta_omega_mu=e1*(e1+d0)*tan(e2)-sin(e2)/k2;
      delta_omega_mu=delta_omega_mu*0.05*number_cycle;
      omega_mu +=delta_omega_mu;
         //calculate data to send to slave
     calculate_slave_velocities();
//CHECK FOR EXCEEDED VELOCITIES
         if (omega_left_new>90)//maximum limit
         {
         printf("ERR1");//err1: Left velocity exceed the range
```

```
            omega_left_new=90;
           }
            if (omega_left_new<-90)//maximum limit
            {
            printf("ERR2");//err1: Left velocity exceed the range
            omega_left_new=-90;
            }
            if (omega_right_new>90)//maximum limit
            {
            printf("ERR3");//err2: Right velocity exceed the range
            omega_right_new=90;
            }
            if (omega_right_new<-90)//maximum limit
          {
            printf("ERR4");//err2: Right velocity exceed the range
            omega_right_new=-90;
            }
    //SEND NEW VELOCITIES
        send_commands_via_i2c
(address_left,omega_left_new,address_right,omega_right_new);

printf("l:%ld\nr:%ld\n",omega_left_new,omega_right_new);
        omega_left_last=omega_left_new;
        omega_right_last=omega_right_new;
        //wait until next cycle
        new_cycle=0;
          }
       }
    //SEND STOP COMMAND
```

```
send_commands_via_i2c(address_left,0xffff,address_right,0xffff
);
      tilt_c5(200);//ringing for stop declaring
      delay_ms(200);
      tilt_c5(200);
   }
```

Chương trình cho slave module:

```
/**************
```

Description: This program is used for controlling motor DC velocity with PID method.

There are slight differences between the programs for left and right modules.

Doan Minh Dang - P9900012

Start date: 2004.04.27

End date: 2004.06.20

```
**************/
#include <16f877.h>
#device PIC16F877 *=16 ADC=10
#fuses hs, nowdt, noprotect, nolvp, put, brownout
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6, rcv=PIN_C7)
#use
I2C(slave,sda=PIN_C4,scl=PIN_C3,address=0xa0,FORCE_HW)
#include <stdlib.h>
#include <string.h>
#include <control_motor.c>
//the function run_motor(value,condition,direct,frequency) is
used to control motor
```

```
//constant
float kp=8.2;
float ki=1;
float kd=0.8;
//variables
int1 i2c_flag=0;
int1 start_cycle=0;
int1 stop=0;
int i2c_error=0;
int velocity_error=0;
int16 times=0;
int error_command;
signed int count=0;
signed int count_last=0;
signed int delta_count=0;
char s[7];
byte command_listen[3];
int i=0;
signed int16 command_velocity=0; //[rpm]
int1 direction;
float real_velocity=0;
signed int16 real_velocity_print;
signed int16 duty=0;
int16 PWMduty;
float error_rpm_now=0;
float error_rpm_last=0;
float error_rpm_last_last=0;
float p_value;
float i_value;
```

```
    float d_value;
    float control_value;
    ////////////////////////////////////////////////////////////////
////////////////////////////////
    void tilt_a0(int16 speak_time)
    {//slave board will speak in specified time(ms)
      output_high(PIN_A0);
      delay_ms(speak_time);
      output_low(PIN_A0);
    }

    void convert_duty()
    {
        if (duty<0)
        {
          PWMduty=-duty;
          direction=0;//negative speed: rotate backward
          }
        else
          {
          PWMduty=duty;
          direction=1;//positive speed: rotate forward
          }
          //control_motor(dir=0): backward
          //control_motor(dir=1): forward
      if (PWMduty>1023)
      {
            PWMduty=1023;
            velocity_error++;
```

```
      }
      if (64>PWMduty)
      {
            PWMduty=64;
      }
    }

    void PID_calculation()
    {
        p_value=kp*(error_rpm_now-error_rpm_last);
        i_value=ki*error_rpm_now;
        d_value=kd*(error_rpm_now-2*error_rpm_last
+error_rpm_last_last);
        control_value=p_value+i_value+d_value;
        duty+=(signed int16)control_value;
    }

    ////////////////////////////////////////////////////////////
//////////
    #INT_SSP
    void ssp_isr()
    {
      if (i2c_poll()==FALSE)
       {
        i=0;
       }
      else
       {
            command_listen[i++]=i2c_read();
```

```
//        i=1;
            if (i==3)
          {
                  times++;
          if
((command_listen[1]!=0xa0)&&(command_listen[2]!=0xa0))
    //condition: data must be received correctly
              {
                  i2c_flag=1;
                  i=0;
command_velocity=make16(command_listen[2],command_listen[1]);
                  if (command_velocity==0xffff) //stop command
                  {
                        stop=1;
                  }
              }
              else i2c_error++;
        }
      }
    }
    #INT_EXT
    void count_encoder()
    {
      if (input(PIN_B7)) count--;//backward rotate - left
      else count++;//forward - left
      //reverse for right module
    }
    ////////////////////////////////////////////////////////
/////
```

```
#INT_TIMER1
void update_time()
{
  start_cycle=1;
  set_timer1(15536); //update time = 10ms
  delta_count=count-count_last;
  count_last=count;
//multiplier: 4ms - 7.5; 5ms - 6; 10ms - 3
  real_velocity=(float)delta_count*3.0;
  error_rpm_last_last=error_rpm_last;
  error_rpm_last=error_rpm_now;
  error_rpm_now=(float)command_velocity-real_velocity;
}
//////////////////////////////////////////////////////////////////
//////
void main()
{
  tilt_a0(400);
  set_tris_b(0x81);
  ext_int_edge(0,L_TO_H);
  control_motor(0,0,0,2);
  command_listen[0]=0;
  command_listen[1]=0;
  command_listen[2]=0;
  setup_timer_1(T1_INTERNAL);
  enable_interrupts(GLOBAL);
  enable_interrupts(INT_SSP);
  while (!i2c_flag) {};
  delay_us(3385);
```

```
tilt_a0(500);
set_timer1(15535);
enable_interrupts(INT_EXT);
enable_interrupts(INT_TIMER1);
start_cycle=0;
while (!stop)
{
  if (start_cycle)
  {
    if (command_velocity!=0)
    {
      PID_calculation();
            convert_duty();
      control_motor(PWMduty,1,direction,2);
       }
    else {control_motor(0,1,0,2);}//stop
    start_cycle=0;
  }
}
control_motor(0,0,0,2);
disable_interrupts(GLOBAL);
disable_interrupts(INT_SSP);
disable_interrupts(INT_EXT);
disable_interrupts(INT_TIMER1);
tilt_a0(200);
delay_ms(200);
tilt_a0(200);
delay_ms(200);
if ((i2c_error>0)||(velocity_error>0))
```

```
    {
     tilt_a0(1000);
    }
   while (1)
   {
    gets(s);
     error_command=atoi(s);
    switch (error_command)
    {
            case 1: printf("i2c_error:%u ",i2c_error);
            break;
            case 2: printf("velocity_error:%u",velocity_error);
            break;
            case 3: printf("times:%lu",times);
            break;
    }
   }
}
```