BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN THÙY LINH

ĐIỀU KHIỂN ROBOT HAI BÁNH TỰ CÂN BẰNG SỬ DỤNG THUẬT TOÁN ĐIỀU KHIỂN TRƯỢT

LUẬN VĂN THẠC SĨ

Chuyên ngành : Kỹ thuật Cơ điện tử

Mã số ngành: 60520114

TP. HỒ CHÍ MINH, tháng … năm 2016

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN THÙY LINH

ĐIỀU KHIỂN ROBOT HAI BÁNH TỰ CÂN BẰNG SỬ DỤNG THUẬT TOÁN ĐIỀU KHIỂN TRƯỢT

LUẬN VĂN THẠC SĨ

Chuyên ngành : Kỹ thuật Cơ điện tử

Mã số ngành: 60520114

CÁN BỘ HƯỚNG DẪN KHOA HỌC:

PGS.TS NGUYỄN THANH PHƯƠNG

TP. HỒ CHÍ MINH, tháng … năm 2016

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

Cán bộ hướng dẫn khoa học : PGS.TS NGUYỄN THANH PHƯƠNG

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Luận văn Thạc sĩ được bảo vệ tại Trường Đại học Công nghệ TP. HCM ngày … tháng … năm …

Thành phần Hội đồng đánh giá Luận văn Thạc sĩ gồm:

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ Luận văn Thạc sĩ)

TT	**Họ và tên**	**Chức danh Hội đồng**
1		Chủ tịch
2		Phản biện 1
3		Phản biện 2
4		Ủy viên
5		Ủy viên, Thư ký

Xác nhận của Chủ tịch Hội đồng đánh giá Luận sau khi Luận văn đã được sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM
PHÒNG QLKH – ĐTSĐH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

TP. HCM, ngày..... tháng….. năm 20.....

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: Nguyễn Thùy Linh ..... .Giới tính: Nữ
Ngày, tháng, năm sinh: 03/12/1989 ..... Nơi sinh: Long an
Chuyên ngành: Kỹ thuật Cơ Điện tử .....MSHV: 1341840024

I- Tên đề tài:

Điều khiển Robot 2 bánh tự cân bằng sử dụng điều khiển trượt (Sliding mode) .....
.....
.....

II- Nhiệm vụ và nội dung:

- Nghiên cứu thiết kế và thi công mô hình robot 2 bánh tự cân bằng.
- Sử dụng phương pháp điều khiển phi tuyến là điều khiển trượt.
- Thiết kế bộ điều khiển dựa vào phương pháp điều khiển trên mô phỏng dựa trên Matlab Simulink.

III- Ngày giao nhiệm vụ: *27/05/2015*

IV- Ngày hoàn thành nhiệm vụ: 30/08/2016

V- Cán bộ hướng dẫn: *PGS.TS Nguyễn Thanh Phương*

CÁN BỘ HƯỚNG DẪN
(Họ tên và chữ ký)

KHOA QUẢN LÝ CHUYÊN NGÀNH
(Họ tên và chữ ký)

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này đã được cảm ơn và các thông tin trích dẫn trong Luận văn đã được chỉ rõ nguồn gốc.

Học viên thực hiện Luận văn

(Ký và ghi rõ họ tên)

LỜI CÁM ƠN

Để hoàn thành luận văn này, em xin tỏ lòng biết ơn sâu sắc đến Thầy PGS.TS. Nguyễn Thanh Phương đã tận tình hướng dẫn trong suốt quá trình viết Luận văn tốt nghiệp.

Em chân thành cảm ơn quý Thầy, Cô trong khoa Sau Đại học và khoa Cơ – Điện – Điện tử, Trường Đại học Công nghệ Thành phố Hồ Chí Minh đã tận tình truyền đạt kiến thức trong những năm em học tập. Với kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu khóa luận mà còn là hành trang quí báu để em ứng dụng trong công việc một cách vững chắc và tự tin.

Cuối cùng em kính chúc quý Thầy, Cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý. Đồng kính chúc các học viên lớp cao học 13SCĐ21 luôn dồi dào sức khỏe, đạt được nhiều thành công tốt đẹp trong công việc.

NGUYỄN THÙY LINH

TÓM TẮT

Nhiệm vụ của đề tài là thực thi và ứng dụng kỹ thuật điều khiển phi tuyến để thiết kế bộ điều khiển phù hợp cho mô hình xe hai bánh tự cân bằng.

Mô hình toán học của xe hai bánh tự cân bằng được xây dựng để làm cơ sở thiết kế bộ điều khiển.

Phương pháp điều khiển phi tuyến được tìm hiểu và sử dụng trong luận văn là điều khiển trượt cho vòng điều khiển góc nghiêng kết hợp với bộ điều hiển PD cho vòng điều khiển vị trí xe. Thực hiện mô phỏng bộ điều khiển trượt kết hợp với bộ điều khiển PD cho đối tượng xe hai bánh tự cân bằng với phần mềm Matlab/Simulink. Mô hình thực nghiệm được xây dựng để kiểm chứng bộ điều khiển.

ABSTRACT

The main point of this thesis is to design the non-linear control to manipulate the two-wheeled self-balancing robot.

The controller is designed based on the mathematic model of the two-wheeled self-balancing robot which is studied in this thesis.

The method used in this thesis is sliding mode controller for angular of robot and PD controller for the robot position in Matlab-Simulink presentation.

Simulation results show that the designed controller have good performances in terms of quick response, good balance and stability. The experimental model is established to check the effectiveness of the designed controller.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

DSP	Digital Signal Processor
Segway PT	Segway Personal Transporter

DANH MỤC CÁC HÌNH

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI

1.1. Đặt vấn đề:

Trong ngành tự động hóa – điều khiển tự động nói chung và điều khiển học nói riêng, mô hình con lắc ngược là một trong những đối tượng nghiên cứu điển hình và đặc thù bởi đặc tính động không ổn định của mô hình nên việc điều khiển được đối tượng này trên thực tế đặt ra như một thử thách. Công nghệ robot đã đạt được nhiều thành tựu lớn và xuất hiện trong nhiều lĩnh vực như: robot dân dụng giúp việc gia đình, robot thực hiện việc giải trí - quảng cáo, robot công nghiệp, robot tự hành thám hiểm trong lòng đất, robot thăm dò các hành tinh trong khoa học vũ trụ.

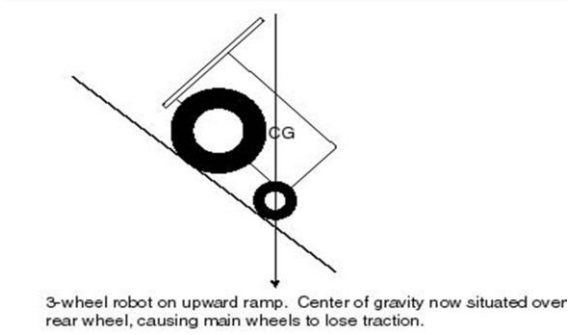
Kết quả nghiên cứu mô hình con lắc ngược cơ bản, ví dụ như mô hình xe-con lắc, con lắc ngược quay… có thể ứng dụng và kế thừa sang các mô hình tương tự khác nhưng có tính ứng dụng thực tiễn hơn, chẳng hạn như mô hình tên lửa, mô hình xe hai bánh tự cân bằng…

Xuất phát từ ý tưởng chiếc xe hai bánh tự cân bằng Segway, một phát minh nổi tiếng của Dean Kamen – một kỹ sư người Mỹ vào năm 2001, đã mở ra một hướng phát triển chế tạo robot mới, đó là các robot hai bánh tự cân bằng. Đây là loại robot có hai bánh, có thể tự giữ thăng bằng, di chuyển và hoạt động dễ dàng trong các khoảng không gian nơi mà sự linh hoạt, cơ động, hiệu quả được đặt lên hàng đầu. Với những ưu điểm đó, robot hai bánh tự cân bằng đã nhận được nhiều sự quan tâm từ các nhà nghiên cứu và các hãng sản xuất robot trên thế giới trong những năm gần đây.

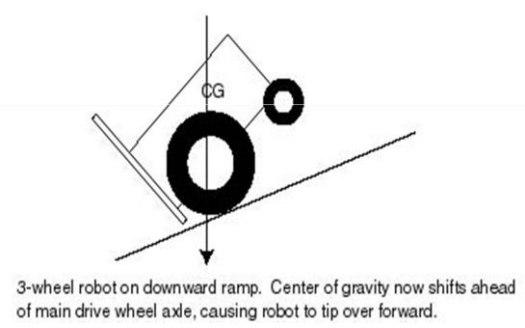
Robot hai bánh tự cân bằng được xem như cầu nối kinh nghiệm giữa mô hình con lắc ngược với robot hai chân và robot giống người. Đây là dạng robot có hai bánh đồng trục, do đó khắc phục được những nhược điểm vốn có của các robot hai hoặc ba bánh kinh điển. Các robot hai hoặc ba bánh kinh điển, theo đó có cấu tạo gồm bánh dẫn động và bánh tự do để đỡ trọng lượng robot. Nếu trọng lượng được đặt nhiều vào bánh lái thì robot sẽ không ổn định và dễ bị ngã, còn nếu đặt vào nhiều bánh đuôi thì hai bánh chính sẽ mất khả năng bám. Nhiều thiết kế robot có thể di chuyển tốt trên địa hình phẳng nhưng không thể di chuyển lên xuống trên địa hình lồi lõm hoặc mặt phẳng nghiêng. Khi di chuyển lên đồi, trọng lượng robot dồn vào đuôi xe làm mất khả năng bám và trượt ngã.

Tuy nhiên, loại robot này hoạt động dựa trên mô hình hệ con lắc ngược gắn lên trên trục có hai bánh xe, nên khuyết điểm chính của nó là cần phải có một bộ điều

khiển để điều khiển cho robot luôn giữ được thăng bằng, di chuyển và hoạt động. Do vậy, bài toán được đặt ra ở đây là phải nghiên cứu, thiết kế được bộ điều khiển phù hợp với một mô hình phi tuyến và có các thông số hệ thống là bất định như là mô hình xe hai bánh tự cân bằng.



Hình 1.1 - Robot dạng 3 bánh xe khi lên dốc, trọng lượng dồn vào bánh trước khiến lực ma sát giúp xe bám trên mặt đường không được đảm bảo.



Hình 1.2 - Robot dạng 3 bánh xe khi xuống dốc, trọng lực dồn vào bánh sau khiến xe có thể bị lật úp.

Ngược lại, các robot dạng hai bánh đồng trục lại thăng bằng rất linh động khi di chuyển trên địa hình phức tạp, mặc dù bản thân robot là một hệ thống không ổn định. Khi robot di chuyển trên địa hình dốc, nó tự động nghiêng ra trước và giữ cho trọng lượng dồn về hai bánh chính. Tương tự, khi di chuyển xuống dốc, nó

nghiêng ra sau và giữ trọng tâm rơi vào bánh chính. Vì vậy, không bao giờ có hiện tượng trọng tâm xe rơi ngoài vùng đỡ bánh xe để có thể gây ra lật úp.

1.2. Các công trình liên quan

1.2.1. Một số mô hình robot hai bánh tự cân bằng

Robot hai bánh tự cân bằng, tự di chuyển và hoạt động, không chở người như : Robot-JOE (mục 1.2.1.1), N-Bot (mục 1.2.1.2), Robot dọn rác (mục 1.2.1.3).

Xe hai bánh tự cân bằng, có chở người, là phương tiện di chuyển: xe Segway (mục 1.2.1.4), robot chở người của hãng Toyota (mục 1.2.1.5).

1.2.1.1. Robot JOE - [2]

Đây là sản phẩm của phòng thí nghiệm Điện tử công nghiệp của Viện công nghệ Federal, Lausanne, Thụy Sĩ. Robot JOE cao 65cm, nặng khoảng 12kg, tốc độ tối đa 1,5 m/s, có thể di chuyển trên dốc nghiêng 30 độ.

Hình 1.3: Robot JOE

Nguồn điện cấp là nguồn pin 32V dung lượng 1.8Ah. Hình dạng của nó gồm hai bánh xe đồng trục, mỗi bánh gắn với một động cơ DC, robot này có thể chuyển động xoay theo hình chữ U. Hệ thống điều khiển gồm hai bộ điều khiển "không gian trạng thái" (state space) tách rời nhau, kiểm soát động cơ để giữ cân bằng cho hệ thống. Thông tin trạng thái được cung cấp bởi hai encoder quang và hai cảm biến là gia tốc góc và con quay hồi chuyển (gyro). JOE được điều khiển bởi một bộ điều khiển từ xa RC. Bộ điều khiển trung tâm và xử lý tín hiệu là một board xử lý tín hiệu số (DSP) phát triển bởi chính nhóm và của viện Federal, kết hợp với FPGA của XILINC.

1.2.1.2. N-Bot, [19]

Hình 1.4: Mô hình N-Bot

N-Bot là sản phẩm của David P.Anderson và là một trong những robot hai bánh cân bằng thành công nhất khi được công nhận là "Robot độc đáo trong tuần" do NASA phong tặng và được một số tổ chức, tạp chí khoa học đánh giá cao.

Nguyên tắc cơ bản của N-Bot là điều khiển hai bánh xe của robot chạy theo chiều mà phần thân phía trên của robot có khả năng ngã đổ. Nếu các bánh có thể được lái theo cách đứng vững theo trọng tâm của robot thì robot sẽ giữ được cân bằng. Quá trình điều khiển sử dụng 2 tín hiệu cảm biến phản hồi là cảm biến góc nghiêng để đo góc nghiêng của robot với phương của trọng lực, và encoder gắn ở bánh để đo vị trí của robot. Các biến sau đây thể hiện sự chuyển động và vị trí của 'con lắc ngược' này giúp nó giữ cân bằng: Góc nghiêng của thân robot (theta), đạo hàm góc nghiêng hay chính là vận tốc góc nghiêng (theta-dot), vị trí robot (position), đạo hàm của vị trí hay vận tốc di chuyển của robot (position-dot). Bốn thông số này sẽ được đưa vào bộ điều khiển để tính ra điện áp điều khiển U cho 2 động cơ lái hai bánh xe.

1.2.1.3. Xe hai bánh cân bằng gom rác

Với dáng tròn trịa, robot có tên gọi DustCart di chuyển trên hai bánh xe, tự định vị và có thể đến đúng ngôi nhà gọi nó tới thu gom rác. Giáo sư Paolo Dario - thuộc trường Sant'Anna ở Pisa và là điều phối viên của dự án DustBot do EU tài trợ - cho biết: "Chúng tôi đã tập hợp những thành phần chế tạo robot tiên tiến nhất để tạo ra DustCart, người máy giúp việc cho các đơn vị thu gom rác trên khắp châu Âu. Nó không chỉ là một thùng rác di động có ngăn kéo để bạn bỏ bao rác vào, mà còn có nhiều tính năng khác".

Hình 1.5: Xe hai bánh "DustCart" gom rác tự động

DustCart có thể di chuyển qua con đường hẹp, được trang bị camera và các thiết bị cảm biến khác. Xe có thể quan sát nơi nó đang di chuyển, chụp ảnh đường đi và phân tích thông tin để tránh va chạm vào các chướng ngại vật cố định. Nó cũng có thể nhận ra những đối tượng đang di chuyển, ví dụ như khách đi bộ, xe máy và nhanh chóng tính toán đường đi rồi đổi hướng để tránh va chạm. Những hình ảnh hiển thị cũng được chuyển về trung tâm kiểm soát để nhân viên phụ trách có thể giám sát hoạt động của DustCart và can thiệp nếu cần thiết. Xe này sử dụng một hệ thống tam giác thông minh để di chuyển đến nhà một hộ dân bằng cách tương tác với mạng không dây. Mạng không dây có thể xác định chính xác vị trí của xe, tính toán tuyến đường tối ưu giữa những lần gom rác và chuyển thông tin này đến xe.

1.2.1.4. Xe Segway PT , [20]

Segway PT (viết tắt của Segway Personal Transporter - xe cá nhân Segway), đặc điểm nổi bật của Segway là cơ chế tự cân bằng nhờ hệ thống máy tính, động cơ và con quay hồi chuyển đặt bên trong xe, nó giúp cho xe dù chỉ có một trục chuyển động với hai bánh nhưng luôn ở trạng thái cân bằng, người sử dụng chỉ việc ngả về đằng trước hoặc đằng sau để điều khiển xe đi tiến hoặc đi lùi.

Với các điều khiển sang phải hoặc sang trái, Segway có một cần lái- muốn điều khiển sang phải hoặc sang trái chỉ cần nghiêng cần lái về phía đó. Động cơ của xe Segway có thể đạt tốc độ 5,6 m/s (khoảng 20 km/h). Do có giá thành khá cao và mới chỉ thích hợp ở các địa điểm bằng phẳng nên Segway hiện chủ yếu được sử dụng ở các sở cảnh sát, căn cứ quân sự, cơ sở sản xuất hoặc khu công nghiệp.

Hình 1.6: Các kiểu dáng của xe Segway

Cơ chế tự cân bằng của Segway dựa trên hoạt động của hệ thống máy tính, hai sensor độ nghiêng và năm con quay hồi chuyển đặt trong xe. Dựa trên các số liệu của sensor, máy tính sẽ tính toán để truyền lệnh cho các động cơ phụ di chuyển bánh xe về phía trước hoặc phía sau để tái lập cân bằng cho xe.

Với các mẫu Segway PT mới, quá trình này lặp đi lặp lại khoảng 100 lần trên giây, đủ để cân bằng xe cho dù người lái ở trạng thái nào. Khi xe đạt tới vận tốc tối đa, các phần mềm trong Segway sẽ tự động điều khiển xe hơi nghiêng về sau giúp xe di chuyển chậm lại, cơ chế này giúp hạn chế khả năng người điều khiển tiếp tục nghiêng về trước ngay cả khi Segway đã ở vận tốc tối đa. Các Segway cũng sẽ tự động giảm tốc và dừng lại khi gặp chướng ngại vật.

Về sự an toàn, Segway có tốc độ tối đa 20 km/giờ và không chạy quá 20 km/h, kể cả khi xuống dốc. Tất cả những thiết bị an toàn (ắc quy, động cơ, máy tính) đều được gắn 2 bộ vào xe. Trong trường hợp 1 bộ phận bị hư hỏng bất ngờ, Segway vẫn có thể ổn định và ngừng một cách an toàn. Những năng lượng có thể tạo ra được khi thắng hoặc trượt dốc đều được nạp lại vào bình ắc quy.

1.2.1.5. Xe di chuyển người của hãng Toyota

Toyota đã trình bày một xe di động hai bánh thông minh trong năm 2010. Xe này được sử dụng cho người già hoặc người tàn tật di chuyển trên địa hình gồ ghề và nhiều chướng ngại vật xung quanh.

Điểm nổi bật của xe này là không gây ra phiền toái nào cho người ngồi trên

ghế vì khả năng tự điều chỉnh của robot khi có những sự thay đổi về địa hình di chuyển.

Hình 1.7: Robot chở người của hãng Toyota

1.2.2. Các báo cáo nghiên cứu khoa học liên quan

Xe hai bánh tự cân bằng đã, đang được sự quan tâm từ nhiều nhà nghiên cứu trên thế giới nên có khá nhiều sách, bài báo khoa học viết về vấn đề này. Các báo cáo khoa học liên quan đến đề tài này thường tập trung vào những nội dung sau:

- Mô hình hóa hệ thống động của xe hai bánh tự cân bằng, tài liệu tham khảo số [1] , [2] , [3] , [5].

- Sử dụng các phương pháp điều khiển phi tuyến để thiết kế bộ điều khiển cho mô hình xe hai bánh tự cân bằng, hệ con lắc ngược.

+ Sử dụng phương pháp điều khiển cuốn chiếu (Backstepping Control), tài liệu tham khảo số [4] , [5] , [6], [7].

+ Sử dụng phương pháp điều khiển trượt, tài liệu tham khảo số [8] , [9].

+ Sử dụng giải thuật điều khiển thông minh để thiết kế bộ điều khiển cho mô hình robot hai bánh tự cân bằng, tài liệu tham khảo số [5], [15], [16].

1.3. Phạm vi nghiên cứu

Trong đề tài này, tác giả tìm hiểu và ứng dụng kỹ thuật điều khiển Trượt.

Các mục tiêu chính của đề tài bao gồm:

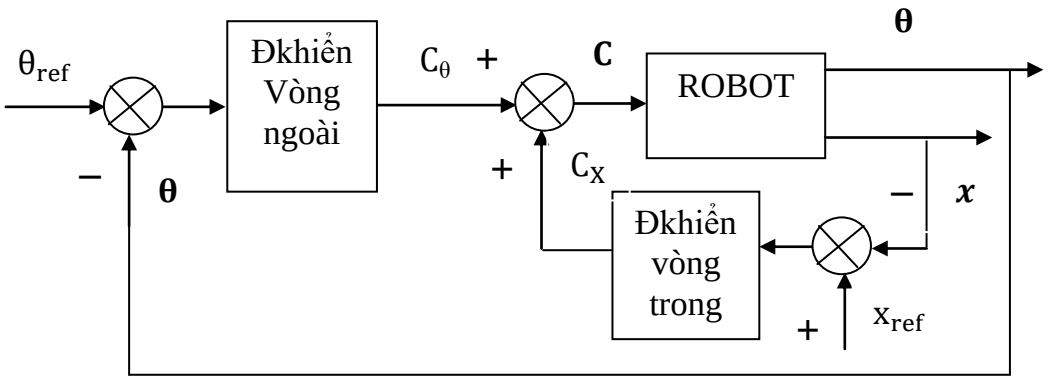
+ Mô hình hóa đối tượng xe hai bánh tự cân bằng.

+ Thực thi và ứng dụng phương pháp điều khiển trượt cho hệ thống.

+ Thiết kế bộ điều khiển dựa vào hai phương pháp điều khiển trên cho xe hai

bánh tự cân bằng trên Matlab Simulink.

+ Sơ đồ điều khiển tổng quát cần thực hiện trong luận văn như sau:

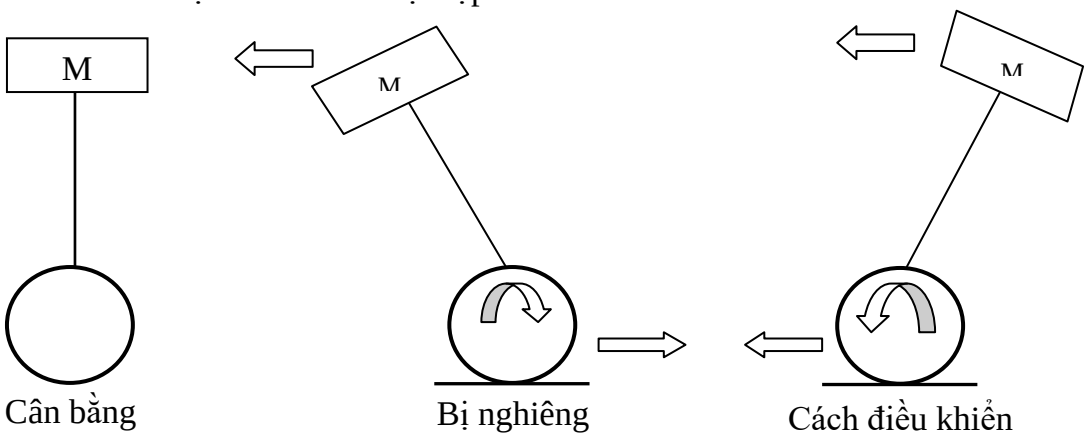


Hình 1.8: Sơ đồ điều khiển

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT LIÊN QUAN

2.1. Nguyên lý hoạt động của xe hai bánh cân bằng:

Xe hai bánh tự cân bằng hoạt động kết hợp giữa mô hình con lắc ngược với hệ hai bánh xe được điều khiển độc lập nhau.



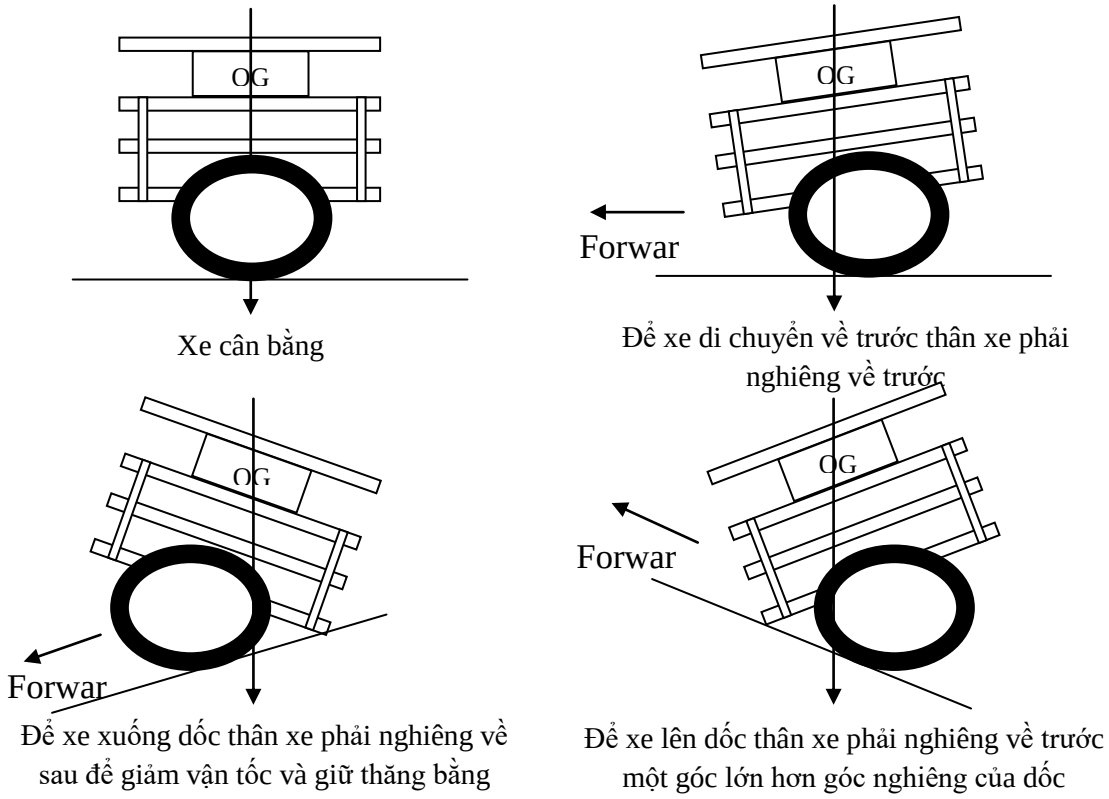
Hình 2.1 - Nguyên lý hoạt động của xe hai bánh tự cân bằng

- Khi cân bằng, góc nghiêng của thân xe với phương trọng lực bằng 0.

- Khi xe nghiêng về phía trước, nếu không điều khiển thì xe sẽ bị ngã. Trường hợp này, cần phải điều khiển xe chạy về phía trước sao cho góc nghiêng bằng 0, để xe thăng bằng trở lại.

Trường hợp xe nghiêng về phía sau cũng phải điều khiển tương tự như vậy, nghĩa là điều khiển xe chạy về hướng đang nghiêng, để robot thăng bằng trở lại.

Hình ảnh dưới đây là các trường hợp di chuyển của xe 2 bánh tự cân bằng.



Hình 2.2: Cách di chuyển của xe 2 bánh cân bằng

2.2. Lý thuyết về phương pháp điều khiển trượt.

2.2.1. Điều khiển bám (Tracking).

Đối tượng: Xét hệ thống phi tuyến biểu diễn bởi phương trình vi phân

$$y^n = f(y, \dot{y}, \ddot{y}, y^{(n-1)})u \quad (2.1)$$

Đặt $x_1 = y, x_2 = \dot{y}, x_3 = \ddot{y}, \dots, x_n = y^{(n-1)}$ (2.2)

khi đó, ta được biểu diễn trạng thái

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \vdots \\ \cdot \\ \dot{x}_{n-1} = x_n \\ \dot{x}_n = f(x) + g(x)u \end{cases} \quad (2.3)$$

$y = x_1$

Yêu cầu: Xác định tín hiệu điều khiển u sao cho tín hiệu ra y bám theo tín hiệu đặt r

- Mặt trượt: Định nghĩa tín hiệu sai lệch

$$e = y - r \quad (2.4)$$

và mặt trượt S như sau: $S = e^{(n-1)} + a_{n-2}e^{(n-2)} + \cdots + a_1\dot{e} + a_0 e$ (2.5)

Trong đó, $a_0, a_1, \ldots, a_{n-3}, a_{n-2}$ là các hệ số được chọn trước sau cho đa thức đặc trưng của phương trình vi phân sau Hurwitz có tất cả các nghiệm với phần thực âm

$$e^{(n-1)} + a_{n-2}e^{(n-2)} + \cdots + a_1\dot{e} + a_0 e = 0 \qquad (2.6)$$

Khi đó, nếu $S = 0$ thì sai lệch $e \to 0$ thì $t \to \infty$

Thay (2.4) và (2.2) vào (2.5) ta được

$$S = x_n + a_{n-2}x_{n-1} + \cdots + a_1 x_2 + a_0 x_1 - \left(r^{(n-1)} + a_{n-2}r^{(n-2)} + \cdots + a_1\dot{r} + a_0 r\right) \qquad (2.7)$$

Phương trình $S = 0$ xác định một mặt cong trong không gian n chiều gọi là mặt trượt. Vấn đề là xác định luật điều khiển u để đưa các quỹ đạo pha của hệ thống về mặt trượt và duy trì trên mặt trượt một cách bền vững đối với các biến động của f(x) và g(x).

Lấy đạo hàm (2.7) và áp dụng vào (2.3) ta có

$$\dot{S} = f(x) + g(x)u + a_{n-2}\left(x_n - r^{(n-1)}\right) + \cdots + a_1(x_3 - \ddot{r}) + a_0(x_2 - \dot{r}) \qquad (2.8)$$

Có thể chọn u sao cho $\dot{S} = -\alpha sign(S)$ (2.9)

trong đó α là hằng số dương chọn trước.

Khi đó luật điều khiển u được xác định bởi:

$$u = -\frac{1}{g(x)}\begin{bmatrix} f(x) + a_{n-2}\left(x_n - r^{(n-1)}\right) + \cdots \\ +a_1(x_3 - \ddot{r}) + a_0(x_2 - \dot{r}) + \alpha sign(S) \end{bmatrix} \qquad (2.10)$$

- Tính bền vững của mặt trượt

Trong điều kiện có sai số mô hình, luật điều khiển (2.10) luôn đưa được quỹ đạo pha của hệ thống về mặt trượt $S = 0$ nếu điều kiện sau đây được thỏa mãn:

Nếu $S > 0$ thì $\dot{S} < 0$

Nếu $S < 0$ thì $\dot{S} > 0$ Nếu $S = 0$ thì $\dot{S} = 0$ (2.11)

- Phương pháp chọn mặt trượt: Hàm S trong (2.5) thỏa mãn điều kiện sau:

+ S không phụ thuộc tường minh vào u nhưng $\dot{S}$ phụ thuộc tường minh vào u.

+ Phương trình vi phân (2.6) Hurwitz để nghiệm $e \to 0$ khi $t \to \infty$

2.2.2. Ổn định hóa (regulation)

- Đối tượng điều khiển:

Xét hệ thống $\begin{cases} \dot{x}_1 = f_1(x_1, x_2) \\ \dot{x}_2 = f_2(x_1, x_2) + g(x_1, x_2)u \end{cases}$ (2.12)

Yêu cầu: đưa vec tơ trạng thái x về 0

- Mặt trượt: Định nghĩa $S = x_2 - \varphi(x_1)$ (2.13)

Trong đó, $\varphi(x_1)$ được chọn thỏa các điều kiện sau

+ $\varphi(0) = 0$

+ Hệ thống con $\dot{x}_1 = f_1(x_1, \varphi(x_1))$ có điểm cân bằng ổn định tiệm cận tại gốc tọa độ.

+ S có bậc tương đối bằng 1

- Luật điều khiển:

$$\dot{S} = \dot{x}_2 - \frac{\partial \varphi}{\partial x_1} f_1(x_1, x_2)$$

$$= f_2(x_1, x_2) + g(x_1, x_2)u - \frac{\partial \varphi}{\partial x_1} f_1(x_1, x_2) \quad (2.14a)$$

Ta có thể chọn u sao cho $\dot{S} = -\alpha sign(S)$ (2.14b)

trong đó, α là hằng số dương chọn trước, khi đó luật điều khiển u được xác định bởi

$$u = -\frac{1}{g(x)}\left[f_2(x) - \frac{\partial \varphi}{\partial x_1} f_1(x_1) + \alpha sign(S)\right] \quad (2.15)$$

2.3. Lý thuyết về lọc Kalman.

Năm 1960 R.E Kalman đã xuất bản một bài báo với tiêu đề "A New Approach to Linear Filtering and Predication Problems". Nghiên cứu của Kalman đã khắc phục hạn chế của bộ lọc Wiener-Hopf trong việc giải quyết bài toán thống kê tự nhiên. Kể từ đó, danh từ bộ lọc Kalman đã ra đời. Bộ lọc này ước lượng trạng thái $x \in R^n$ của quá trình thời gian rời rạc theo phương trình sai phân tuyến tính.

$$x_k = Ax_{k-1} + Bu_{k-1} + W_{k-1} \quad (2.16)$$

Với việc đo $z \in R^n$

$$z_k = Hx_k + v_k \quad (2.17)$$

Biến ngẫu nhiên w_k và v_k biểu diễn nhiễu đo và nhiễu quá trình. Trong thuật toán lọc Kalman, đặc tính thống kê của hai biến này phải được biết trước. Chúng ta giả sử các biến này độc lập có phổ trắng và phân bố Gauss.

$$P(W) \sim N(0, Q) v\grave{a}\ P(R) \sim N(0, R) \quad (2.18)$$

Trong thực tế, ma trận hiệp phương sai **nhiễu quá trình** Q và ma trận hiệp phương sai **nhiễu đo** R phải thay đổi theo từng thời điểm, tuy nhiên chúng ta có thể giả sử là hằng số.

Ma trận A(nxn) trong phương trình sai phân (2.16) là ma trận chuyển trạng thái từ thời điểm trước (k-1) sang thời điểm hiện tại (k). Chú ý rằng, trong thực tế A có thể

thay đổi theo từng thời điểm. Nhưng chúng ta cũng có thể giả sử nó là hằng số. Ma trận B (nx1) là ma trận điều khiển có lối vào $\in R^n$. Ma trận H(mxn) trong phương trình (2.17) là ma trận đo lường (ma trận quan sát). Trong thực tế H có thể thay đổi theo từng thời điểm, ở đây chúng ta giả sử là hằng số.

2.3.1. Bản chất toán học của bộ lọc kalman.

Chúng ta định nghĩa $\bar{\hat{x}}_k \in R^n$ là trạng thái tiền ước lượng ở thời điểm thứ k, $\hat{x}_k \in R^n$ là trạng thái hậu ước lượng tại thời điểm thứ k và cho ra giá trị đo z_k. Chúng ta có thể định nghĩa các lỗi tiền ước lượng và lỗi hậu ước lượng như sau

$$e_k^- = x_k - \hat{\mathrm{x}}_\mathrm{k}^- \quad (2.19)$$

$$e_k = x_k - \hat{x}_k \quad (2.20)$$

Ma trận hiệp phương sai lỗi tiền ước lượng:

$$P_k^- = E[e_k^-(e_k^-)^T] \quad (2.21)$$

Ma trận hiệp phương sai lỗi hậu ước lượng

$$P_k = E[e_k\, e_k^T] \quad (2.22)$$

Xuất phát từ phương trình cho bộ lọc Kalman, chúng ta tìm ra một phương trình tính toán trạng thái **hậu ước lượng $\hat{x}_k$** như là một tổ hợp tuyến tính của trạng thái **tiền ước lượng $\hat{\mathbf{x}}_\mathbf{k}^-$** và sự khác nhau giữa giá trị đo **thực tế** z_k và giá trị **tiên đoán** $H\hat{x}_k^-$ được xác định trong phương trình sau.

$$\hat{x}_k = x_k^- + K(z_k - H\hat{x}_k^-) \quad (2.23)$$

Giá trị $z_k - H\hat{x}_k^-$ trong công thức (2.23) được gọi là **giá trị sai khác** giữa giá trị tiên đoán $H\hat{x}_k^-$ và giá trị thực tế z_k đo được. Giá trị này bằng 0 nghĩa là hai giá trị hoàn toàn đồng nhất với nhau.

Ma trận K(mxn) trong phương trình (2.23) gọi là hệ số khuếch đại Kalman nhằm mục đích tối thiểu hoá hiệp phương sai lỗi hậu ước lượng trong (2.22). Độ khuếch đại Kalman có thể được xác định bởi phương trình sau:

$$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{HP_k^- H^T + R} \quad (2.24)$$

Quan sát phương trình (2.24), chúng ta thấy rằng, khi ma trận hiệp phương sai lỗi đo lượng R tiến tới 0 thì hệ số khuếch đại K được xác định như sau:

$$\lim_{P_k^- \to 0} K_k = H^{-1} \quad (2.25)$$

Trường hợp khác, khi hiệp phương sai lỗi tiền ước lượng P_k^- tiến tới 0 thì

$$\lim_{P_k^- \to 0} K_k = 0 \quad (2.26)$$

Khi hiệp phương sai lỗi đo lường R tiến đến 0 thì giá trị z_k là chính xác hơn, trong khi giá trị tiên đoán $H\hat{x}_k^-$ lại kém chính xác. Trường hợp khi giá trị hiệp phương sai lỗi ước lượng trước P_k^- tiến tới 0, giá trị z_k là kém chính xác trong khi đó giá trị tiên đoán $H\hat{x}_k^-$ lại đạt độ chính xác hơn.

2.3.2. Bản chất thống kê của lọc Kalman.

Công thức (2.23) thể hiện bản chất thống kê của tiền ước lượng $\hat{x}_k^-$ qui định trên tất cả các giá trị đo trước z_k.

$$E[x_k] = \hat{x}_k \quad (2.27)$$

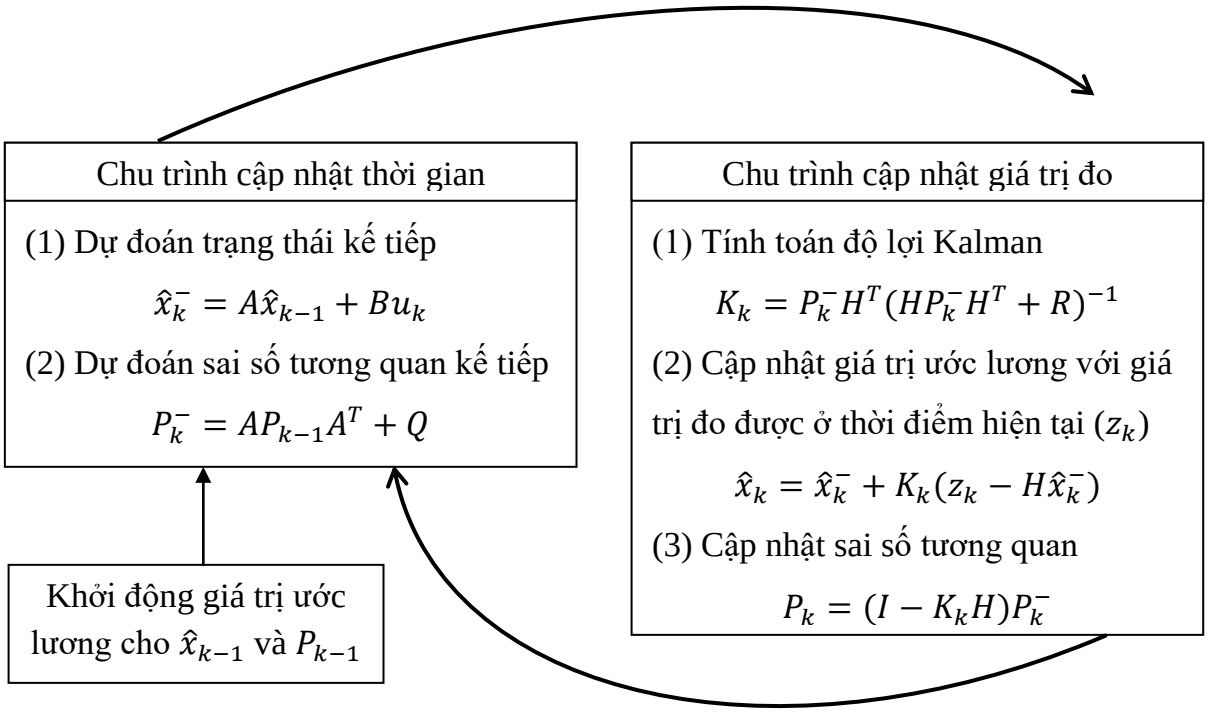
$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k \quad (2.28)$$

Trạng thái hậu ước lượng trong phương trình (2.23) phản ánh giá trị trung bình của phân bố trạng thái nếu các điều kiện (2.24) được thoả mãn. Hiệp phương sai lỗi trạng thái hậu ước lượng trong công thức (2.24) phản ánh sự thay đổi của phân bố trạng thái.

$$P(x_k|z_k) \sim N(E[x_k]), E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]) = N(\hat{x}_k, P_k) \quad (2.29)$$

Tóm lại: Bộ lọc Kalman ước lượng một quá trình bằng việc sử dụng một dạng của điều khiển phản hồi, bộ lọc ước lượng trạng thái quá trình tại một vài thời điểm và sau đó quan sát phản hồi trong dạng của nhiễu đo.

Giải thuật bộ lọc Kalman rời rạc sẽ bao gồm 2 chu trình chính: **chu trình cập nhật thời gian và chu trình cập nhật giá trị đo**. Chu trình cập nhật thời gian dựa vào giá trị trạng thái hiện tại để dự đoán trước giá trị ước lượng của thời điểm kế tiếp. Chu trình cập nhật giá trị đo dựa vào kết quả thật đo được tại ngay thời điểm đó để điều chỉnh giá trị ước lượng đã được dự báo trước ở chu trình cập nhật thời gian. Giá trị của chu trình cập nhật giá trị đo là giá trị ngõ ra của bộ lọc Kalman.



Hình 2.3: Thuật toán bộ lọc Kalman rời rạc

2.3.3. Giải thuật lập trình bộ lọc Kalman rời rạc.

Khai báo biến:

$$P_{init} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.30)$$

R : tượng trưng cho **giá trị nhiễu** covariance. Trong trường hợp này, nó là ma trận 1x1, là giá trị đo được mong đợi. Giá trị của R phụ thuộc vào độ nhạy [V/g] của cảm biến gia tốc góc, nên ta có thể chọn R = 0.08 [V].

Q: tượng trưng cho **tiến trình nhiễu** covariance. Ở đây, nó chỉ mức độ tin cậy của cảm biến gia tốc góc quan hệ với cảm biến vận tốc góc.

$$Q = \begin{bmatrix} Q\_angle & 0 \\ 0 & Q\_gyro \end{bmatrix} = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.003 \end{bmatrix} \quad (2.31)$$

Chương trình bộ lọc Kalman cho vi điều khiển gồm 2 hàm:

- Hàm state_update()

Hàm này được thực hiện trong khoảng thời gian dt trên giá trị cơ sở của cảm biến vận tốc góc. Nó có chức năng cập nhật giá trị góc hiện thời và vận tốc ước lượng.

Giá trị gyro_m được chia thành đúng đơn vị thật, nhưng không cần bỏ gyro_bias độ nghiêng. Bộ lọc quan sát giá trị góc nghiêng.

Vector giá trị được khai báo như sau: $X = \begin{bmatrix} angle \\ gyro\_bias \end{bmatrix}$ (2.32)

Suy ra $\dot{X} = \begin{bmatrix} \dot{angle} \\ \dot{gyro\_bias} \end{bmatrix} = \begin{bmatrix} gyro - gyro\_bias \\ 0 \end{bmatrix}$ (2.33)

Tiếp theo, ta cập nhật ma trận covariance qua công thức thứ hai của chu trình cập nhật ước lượng $P_k^- = AP_{k-1}A^T + Q$ (2.34)

Với A là ma trận Jacobian của $\dot{X}$ và có giá trị như sau:

$$A = \begin{bmatrix} \frac{d(\dot{angle})}{d(angle)} & \frac{d(\dot{angle})}{d(angle\_bias)} \\ \frac{d(\dot{gyro\_bias})}{d(angle)} & \frac{d(\dot{gyro\_bias})}{gyro\_bias} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \quad (2.35)$$

Chi tiết chương trình của hàm state_update() được trình bày trong phần phụ lục C.

- Hàm Kalman_update()

Hàm này được gọi để thực thi khi đã có sẵn giá trị của cảm biến gia tốc góc (biến angel_m). Giá trị của angle_m phải được chuẩn mức 0.

Hàm này không cần phải thực hiện ở mỗi bước thời gian dt (2.5ms), nên ở trong luận văn này, hàm này được thực hiện theo chu kỳ 7.5ms.

Ta tính ma trận C, là ma trận Gradient của giá trị đo lường với giá trị mong đợi (hay còn gọi là vector hiệu chỉnh sai số đo) được xác định như sau:

$$C = \begin{bmatrix} \frac{d(angle\_m)}{d(angle)} & \frac{d(angle\_m)}{d(gyro\_bias)} \end{bmatrix} = [1 \quad 0] \quad (2.36)$$

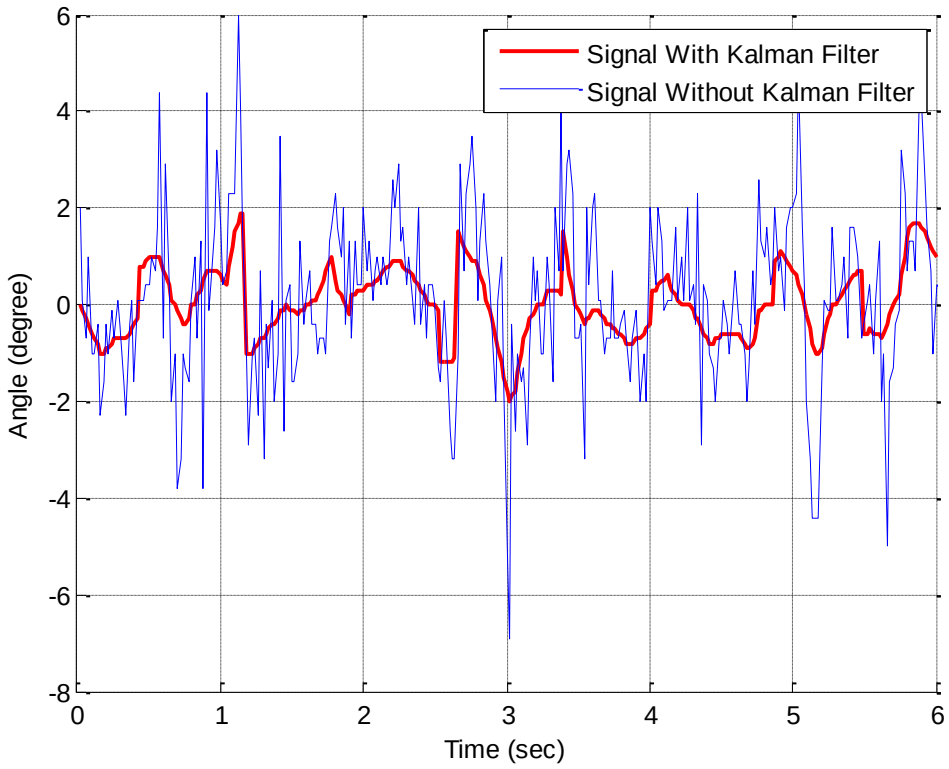
Sau đó, ta thực hiện lần lượt như sau:

+ Tính sai số giữa giá trị góc đo được bởi cảm biến gia tốc và giá trị góc ước lượng.

+ Thực hiện lần lượt 3 phép toán của chu trình cập nhật giá trị đo trong hình 2.7.

Chi tiết chương trình của hàm Kalman_update() được trình bày trong phần phụ lục.

Hình 2.8 trình bày sự so sánh giá trị góc nghiêng thân xe (trong mô hình xe thực nghiệm) trước và sau khi được lọc bởi bộ lọc Kalman.



Hình 2.4: Góc nghiêng thân xe khi có và không có lọc Kalman

Nhận xét: Trong hình 2.8, ta thấy tín hiệu góc nghiêng thân xe trước khi đưa vào bộ lọc Kalman có độ dao động rất lớn và nhiều tín hiệu nhiễu. Còn tín hiệu góc nghiêng sau khi đã được lọc bởi bộ lọc Kalman có độ dao động nhỏ và gần như không có nhiễu. Điều này cho thấy việc sử dụng bộ lọc Kalman để lọc các tín hiệu từ cảm biến góc để đo được góc nghiêng chính xác của mô hình xe hai bánh cân bằng thực nghiệm là rất cần thiết.

2.4. So sánh các bộ lọc với lọc Kalman.

Đối với bộ lọc thông thấp, thông cao hoặc thông dải (lọc thụ động) xấp xỉ Butterworth, Bassel và Chebychev hay elliptic, thường được sử dụng cho một tín hiệu vào và một tín hiệu ra, với tần số làm việc xác định. Ngoài dải tần này, tín hiệu sẽ bị lệch pha, hoặc độ lợi không còn là hằng số mà bị tối thiểu hóa. Do vậy trong trường hợp này, ta dùng hai cảm biến để đo một giá trị là góc (cũng như vận tốc góc), nên việc chỉ dùng một bộ lọc thụ động tỏ ra không phù hợp.

Ta có thể sử dụng bộ lọc bổ phụ (complementary filter) để kết nối hai tín hiệu từ Accelerometer và gyro thành một tín hiệu duy nhất. Accelerometer được đưa qua một bộ lọc thông thấp, còn gyro được đưa qua một bộ lọc thông cao, sau đó hai tín hiệu đã được lọc sẽ nối với nhau thành một tín hiệu duy nhất. Ưu điểm của bộ lọc bổ

phụ là tín toán nhanh, dễ thiết kế. Nhược điểm của bộ lọc này là bản chất vẫn của bộ lọc thông cao và thông thấp, có nghĩa độ lợi tín hiệu không bằng nhau trong toàn dải đo, bị lệch pha rõ rệt tại vùng nối tần số. Hơn nữa giá trị $gyro\_bias$ không được cập nhật thường xuyên dễ làm cho bộ lọc mất tác dụng khi làm việc ở những môi trường rung động hay có nhiệt độ khác nhau. Ngoài ra, cũng phải kể đến việc chuẩn trực bộ lọc này khá khó khăn nếu không có thiết bị quan sát.

Nói tóm lại, các bộ lọc thông thường là một kỹ thuật dùng phần cứng (các mạch điện tử R,L,C) hoặc phần mềm (lọc FIR, lọc IIR, của sổ Hamming… trong xử lý tín hiệu số) là nhằm giữ lại các tín hiệu trong một khoảng thông dải tần số nào đó và loại bỏ tín hiệu ở các dải tần số còn lại. Đối với việc xây dựng bộ lọc bằng phần cứng ra đời trước khi dùng các bộ lọc phần mềm, nhưng việc hiệu chỉnh đặc tính, thay đổi các tham số của bộ lọc phức tạp hơn rất nhiều so với sử dụng giải thuật xử lý tín hiệu số.

Trong các bộ lọc này, nếu tồn tại các tín hiệu nhiễu trong dải thông tần thì kết quả tín hiệu trở nên kém đi rất nhiều để có thể xử lý và điều khiển hệ thống một cách ổn định. Điều này càng tỏ ra rất thực tế đối với các bộ lọc phần cứng, vốn rất dễ bị nhiễu bởi các tín hiệu điện trong lúc hoạt động do sự kém chính xác của các linh kiện và sự bất thường của dòng điện ngõ vào.

Đối với bộ lọc Kalman, thuật ngữ “lọc” không có nghĩa như các bộ lọc trên. Đây là một giải thuật tính toán và ước lượng thống kê tối ưu tất cả các thông tin ngõ vào được cung cấp tới nó để có được một giá trị ra đáng tin cậy nhất cho việc xử lý tiếp theo. Do vậy lọc Kalman có thể sử dụng để loại bỏ các tín hiệu nhiễu mà được mô hình là những tín hiệu nhiễu trắng trên tất cả dải thông mà nó nhận được từ ngõ vào, dựa trên các thống kê trước đo và chuẩn trực lại giá trị ước lượng bằng các giá trị đo hiện tại với độ lệch pha gần như không tồn tại và có độ lợi tối thiểu xấp xỉ 0 đối với những tín hiệu ngõ vào không đáng tin cậy. Mặc dù phải tốn khá nhiều thời gian xử lý lệnh, nhưng với tốc độ hiện tại của các vi điều khiển làm việc tính toán ước lượng tối ưu của bộ lọc này trở nên đơn giản và đáng tin cậy rất nhiều. Nhờ có cơ chế tự cập nhật các giá trị cơ sở (bias) tại mỗi thời điểm tính toán cũng như xác định sai lệch của kết quả đo trước với kết quả đo sau nên giá trị đo luôn được ổn định, chính xác, gần như không bị sai số về độ lợi và độ lệch pha của các tín hiệu. Hơn thế, do được xây dựng bởi hàm trạng thái, vì vậy bộ lọc Kalman có thể kết hợp không chỉ hai tín hiệu từ hai cảm biến, mà có thể kết hợp được nhiều cảm biến đo ở những dải tần khác nhau

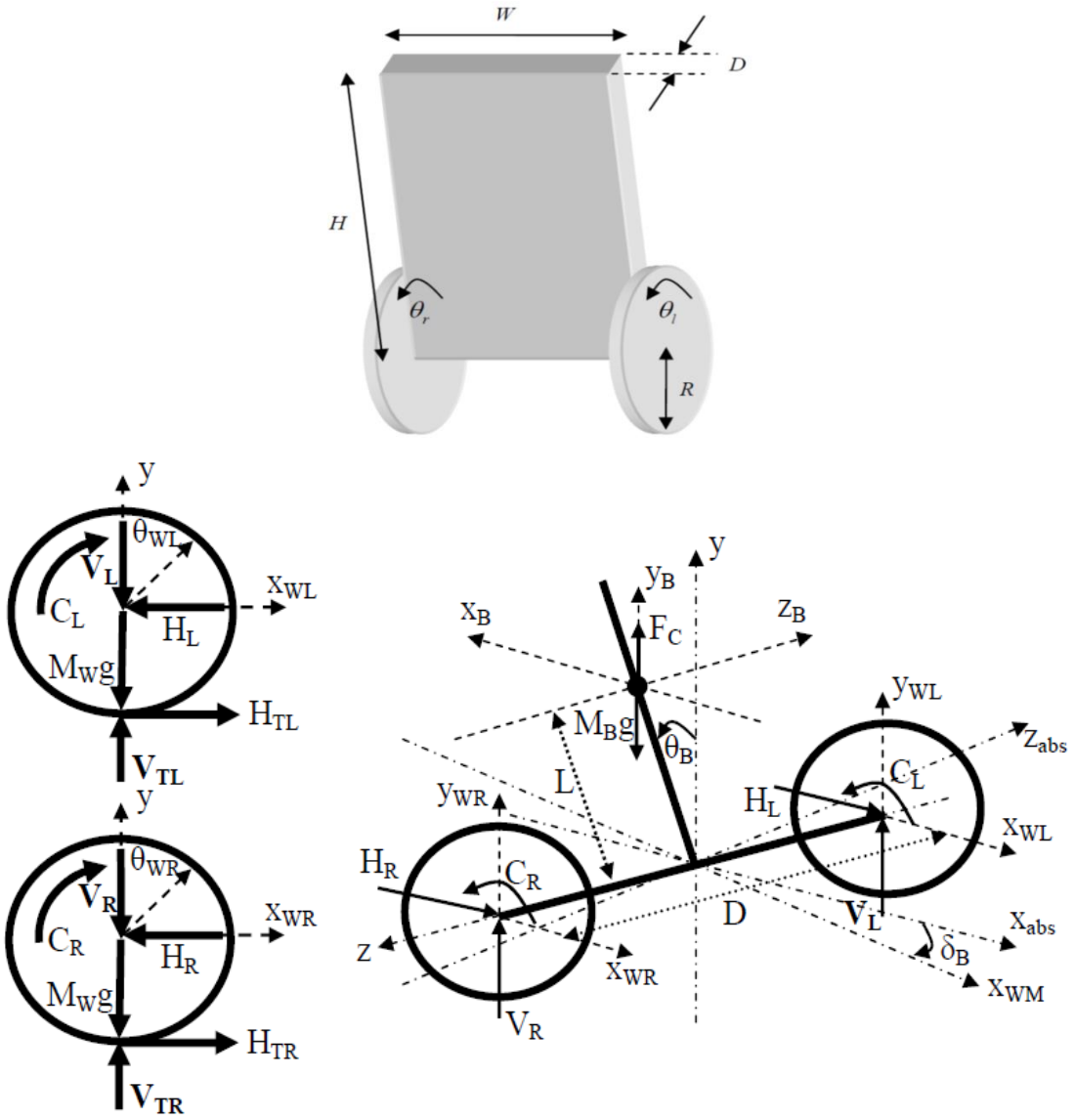
của cùng một giá trị đại lượng vật lý. Chính vì điều này làm bộ lọc Kalman trở nên phổ dụng hơn tất cả những bộ lọc khác trong việc xử lý tín hiệu chính xác của các cảm biến tọa độ, cảm biến la bàn, GPS, gyro,…

Hiện nay, với sự phát triển của trí tuệ nhân tạo (AI), các ứng dụng AI được sử dụng trong nhiều lĩnh vực, đặc biệt là định hướng trong hàng không vũ trụ, nhưng không vì thế mà bộ lọc Kalman giảm vai trò, mà ngược lại đó chính là một ngõ ra đáng tin cậy để cho mạng noron xử lý và ra các quyết định trong các tàu thám hiểm tự hành.

CHƯƠNG 3: THIẾT KẾ BỘ ĐIỀU KHIỂN CHO XE HAI BÁNH CÂN BẰNG

3.1. Mô hình hóa xe hai bánh tự cân bằng

Xây dựng hệ phương trình trạng thái mô tả hệ thống xe hai bánh tự cân bằng.



Hình 3.1: Biểu diễn lực và moment trong mô hình

- Bảng ký hiệu và giá trị các thông số của xe hai bánh cân bằng

Ký hiệu	Thông số	Giá trị [đơn vị]
$M_{WL} = M_{WL} = M_W$	Khối lượng bánh xe trái và phải	0.5[kg]
M_B	Khối lượng qui đổi tại trọng tâm của thân xe	7[kg]

x_W, y_W	Vị trí của bánh xe theo trục x, trục y	[m]
x_B, y_B	Vị trí của trọng tâm thân xe theo trục x và trục y	[m]
L	Khoảng cách từ trọng tâm thân xe đến trục z của hai bánh xe	0.36[m]
R_W	Bán kính bánh xe	0.075
$\theta = \theta_B$	Góc nghiêng thân xe	[rad]
δ	Góc quay của xe	[rad]
D	Khoảng cách giữa hai bánh xe	0.35[m]
H_{TL}, H_{TR}	Lực ma sát giữa bánh xe trái, bánh xe phải với mặt đường	[Nm]
V_{TL}, V_{TR}	Phản lực của mặt đất tương tác lên hai bánh xe trái, bánh xe phải	[Nm]
H_L, H_R	Lực tương tác giữa thân xe và hai bánh xe trái, bánh xe phải	[Nm]
V_L, V_R	Phản lực tương tác giữa thân xe và hai bánh xe trái, phải	[Nm]
g	Gia tốc trọng trường	9.8[m/s^2]
C_L, C_R $C = C_L + C_R$	Moment của động cơ nối với bánh xe trái, bánh xe phải	[Nm]

Hai bánh xe lắc ngược mặc dù phức tạp hơn trong hệ thống động lực nhưng nó tương đồng với hệ con lắc ngược trên xe. Hai bánh con lắc ngược được phân tích tách biệt giai đoạn đầu nhưng cuối cùng hai phương trình của chuyển động đó hoàn toàn mô tả các hành vi của xe cân bằng. Xe có thể bị ảnh hưởng bởi nhiễu cũng như moment xoắn từ động cơ, do đó để sát thực tế mô hình toán phải bao gồm cả các đại lượng này.

- Xét bánh xe trái (bánh phải tương tự)

Áp dụng định luật II Newton lần lượt theo các trục x, trục y và trục quay của bánh xe.

$$M_w \ddot{x} = H_{TL} - H_L \quad (3.1)$$

$$M_w \ddot{y} = V_{TL} - V_L - M_W g \quad (3.2)$$

$$J_{WL}\ddot{H}_{TL} = C_L - H_{TL}R \quad (3.3)$$

- Vị trí của bánh và thân xe

$$x_{wL} = \theta_{WL}R \quad (3.4)$$

$$\longrightarrow \dot{x}_{wL} = \dot{\theta}_{WL}R \quad (3.5)$$

$$\longrightarrow \ddot{x}_{wL} = \ddot{\theta}_{WL}R \quad (3.6)$$

$$x_B = L(sin\theta_B) + \left(\frac{x_{WL}+x_{WR}}{2}\right) \quad (3.7)$$

$$\longrightarrow \dot{x}_B = L\dot{\theta}_B(cos\theta_B) + \left(\frac{\dot{x}_{WL}+\dot{x}_{WR}}{2}\right) = L\dot{\theta}_B(cos\theta_B) + \dot{x}_{WM} \quad (3.8)$$

$$\longrightarrow \ddot{x}_B = L\left(\ddot{\theta}_B\right)(cos\theta_B) - L(sin\theta_B)\left(\dot{\theta}_B\right)^2 + \ddot{x}_{WM} \quad (3.9)$$

$$y_B = -L(1 - cos\theta_B) \quad (3.10)$$

$$\longrightarrow \dot{y}_B = -L\left(\dot{\theta}_B\right)(sin\theta_B) \quad (3.11)$$

$$\longrightarrow \ddot{y}_B = -L\left(\ddot{\theta}_B\right)(sin\theta_B) - L(cos\theta_B)\left(\dot{\theta}_B\right)^2 \quad (3.12)$$

- Xét trên thân xe

Áp dụng định luật II Newton lần lượt theo các trục x, trục y và trục quay tại điểm trọng tâm của thân xe.

$$M_B\ddot{x}_B = H_L + H_R \quad (3.13)$$

$$M_B\ddot{y}_B = V_L + V_R - M_Bg + F_C = V_L + V_R - M_Bg + \left(\frac{C_L+C_R}{L}\right)(sin\theta_B) \quad (3.14)$$

$$J_B\ddot{\theta}_B = L(V_L + V_R)(sin\theta_B) - L(H_L + H_R)(cos\theta_B) - (C_L + C_R) \quad (3.15)$$

$$J_\delta\ddot{\delta} = \frac{D}{2}(H_L - H_R) \quad (3.16)$$

Thay (3.14) và (3.15) vào (3.13) ta được (3.17) như sau:

$$J_B\ddot{\theta}_B = \left(M_B\ddot{y}_B + M_Bg - \left(\frac{C_L + C_R}{L}\right)(sin\theta_B)\right)L(sin\theta_B) - M_BL(cos\theta_B)\ddot{x}_B - (C_L + C_R) \quad (3.17)$$

Rút gọn (3.17) ta được

$$J_B\ddot{\theta}_B = M_BL\left(\ddot{y}_B(sin\theta_B) - \ddot{x}_B(cos\theta_B)\right) + M_BgL(sin\theta_B) - (1 + (sin\theta_B)^2)(C_L + C_R) \quad (3.18)$$

Kết hợp (3.9) với (3.12) ta có

$$\ddot{y}_B(sin\theta_B) - \ddot{x}_B(cos\theta_B) = -L\ddot{\theta}_B - (cos\theta_B)\left(\frac{\ddot{x}_{WL} + \ddot{x}_{WR}}{2}\right)$$
$$= -L\ddot{\theta}_B - \ddot{x}_{WM}(cos\theta_B) \quad (3.19)$$

Thay (3.19) vào (3.18)

$$J_B\ddot{\theta}_B = -M_BL(cos\theta_B)\ddot{x}_B + M_BL^2\left(\ddot{\theta}_B\right) + M_BgL(sin\theta_B)$$
$$-(1 + (sin\theta_B)^2)(C_L + C_R) \quad (3.20)$$

Từ (3.1) ta có

$$M_w\ddot{x}_{WL} + M_w\ddot{x}_{WR} = H_{TL} - H_L + H_{TR} - H_R =$$
$$-(H_L + H_R) + (H_{TL} + H_{TR}) \quad (3.21)$$

Từ (3.3) suy ra $H_{TL} = \frac{C_L - J_{WL}\ddot{\theta}_{WL}}{R}$; $H_{TR} = \frac{C_R - J_{WR}\ddot{\theta}_{WR}}{R}$ (3.22)

Thế (3.13) và (3.22) vào (3.21)

$$M_w(\ddot{x}_{WL} + \ddot{x}_{WR}) = -M_w\ddot{x}_B + \frac{(C_L + C_R) - (J_{WL}\ddot{\theta}_B + J_{WR}\ddot{\theta}_B)}{R} \quad (3.23)$$

Trong đó $\ddot{x}_{WL} + \ddot{x}_{WR} = 2\ddot{x}_{WM}$; $J_{WL}\ddot{\theta}_{WL} = J_{WR}\ddot{\theta}_{WR} = J_W\ddot{\theta}_W$;

Chọn $C = C_L + C_R$. Khi đó biểu thức (3.23) sẽ trở thành:

$$2M_w\ddot{x}_{WM} = -M_B\ddot{x}_B - \frac{C - J_W(\ddot{\theta}_{WL} + \ddot{\theta}_{WR})}{R}$$

Với $\frac{1}{2}(\theta_{WL} + \theta_{WR}) = \theta_B$

Suy ra $2M_w\ddot{x}_{WM} = -M_B\ddot{x}_B - \frac{C - 2J_W\theta_B}{R}$ (3.24)

Thế (3.9) vào (3.24) ta có

$$2M_w\ddot{x}_{WM} = -M_BL\ddot{\theta}_B(cos\theta_B) + M_BL\left(\dot{\theta}_B\right)^2(sin\theta_B) - \frac{2J_W\theta_B}{R} - M_B\ddot{x}_B + \frac{C}{R} \quad (3.25)$$

Từ phương trình (3.20) và (3.25), ta thay $\theta_B = \theta; v\grave{a}\, x_{WM} = x$ ta được hệ phương trình mô tả hệ thống như sau:

$$\begin{cases} J_B\ddot{\theta} = -M_BL(cos\theta)\ddot{x} + M_BL^2\left(\ddot{\theta}\right) + M_BgL(sin\theta) - (1 + (sin\theta)^2)C \\ 2M_w\ddot{x} = -M_BL\left(\ddot{\theta}\right)(cos\theta) + M_BL\left(\dot{\theta}\right)^2(sin\theta) - \frac{2J_W\theta}{R} - M_B\ddot{x} + \frac{C}{R} \end{cases} \quad (3.26)$$

- Xem moment quán tính của thân xe là một thanh có chiều dài L, khối lượng M_B, quay quanh trục z, là trục nối giữa hai bánh xe

$$J_B = \frac{1}{2}\, M_BL^2 \quad (3.26a)$$

- Xem moment quán tính của bánh xe là đĩa tròn xoay có bán kính R, khối lượng M_W , quay quanh trục z, là trục nối giữa hai bánh xe

$$J_W = \frac{1}{2} M_W R^2 \tag{3.26b}$$

Thay (3.26a) và (3.26b) vào (3.26) ta được hệ phương trình như sau

$$\begin{cases} \frac{4}{3} M_B L^2 \left(\ddot{\theta}\right) = -M_B L(cos\theta)\ddot{x} + M_B gL(sin\theta) - (1 + (sin\theta)^2)C \\ (2M_w + M_B)\ddot{x} = M_B L\left(\dot{\theta}\right)^2 (sin\theta) - \left(M_W R + M_B L(cos\theta)\right)\left(\ddot{\theta}\right) + \frac{C}{R} \end{cases} \tag{3.27}$$

Giải hệ phương trình (3.27) ta được

$$\begin{cases} \left(\frac{3}{4}\frac{\left(M_w R + M_B L(cos\theta)\right)(cos\theta)}{(2M_w + M_B)L} - 1\right)\ddot{\theta} = \\ \frac{3}{4}\frac{M_B L(sin\theta)(cos\theta)}{(2M_w + M_B)L}\left(\dot{\theta}\right)^2 - \frac{3}{4}\frac{g(sin\theta)}{L} + \left(\frac{3}{4}\frac{\left(1+(sin\theta)^2\right)}{M_B L^2} + \frac{3}{4}\frac{(cos\theta)}{(2M_w + M_B)RL}\right)C \\ \left(2M_w + M_B - \frac{3}{4}\frac{\left(M_w R + M_B L(cos\theta)\right)(cos\theta)}{L}\right)\ddot{x} = \\ M_B L(sin\theta)\left(\dot{\theta}\right)^2 - \frac{3}{4}\frac{g\left(M_w R + M_B L(cos\theta)\right)(sin\theta)}{L} + \left(\frac{3}{4}\frac{\left(M_w R + M_B L(cos\theta)\right)\left(1+(sin\theta)^2\right)}{M_B L^2} + \frac{1}{R}\right)C \end{cases} \tag{3.28}$$

Chọn biến trạng thái như sau: $x_1 = \theta$; $x_2 = \dot{\theta}$; $x_3 = x$; $x_4 = \dot{x}$. Khi đó hệ phương trình trạng thái mô tả xe ở (3.28) được viết lại như sau:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = A(x_1) + B(x_1, x_2) + g_1(x_1)C \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = A_1(x_1) + B_1(x_1, x_2) + g_2(x_1)C \end{cases} \text{; Với } C = C_L + C_R \tag{3.29}$$

Trong đó:

$$\mathrm{A(x_1)} = \frac{\frac{3}{4}\frac{\mathrm{g(sinx_1)}}{\mathrm{L}}}{\frac{3}{4}\frac{\left(\mathrm{M_w R + M_B L(cosx_1)}\right)(\mathrm{cosx_1})}{(\mathrm{2M_w + M_B})\mathrm{L}} - 1} = \mathrm{f_1(x_1)} \tag{3.30}$$

$$\mathrm{B(x_1, x_2)} = \frac{\frac{3}{4}\frac{\mathrm{M_B L(sinx_1)(cosx_1)}}{(\mathrm{2M_w + M_B})\mathrm{L}}(\mathrm{x_2})^2}{\frac{3}{4}\frac{\left(\mathrm{M_w R + M_B L(cosx_1)}\right)(\mathrm{cosx_1})}{(\mathrm{2M_w + M_B})\mathrm{L}} - 1} = \mathrm{f_2(x_1, x_2)} \tag{3.31}$$

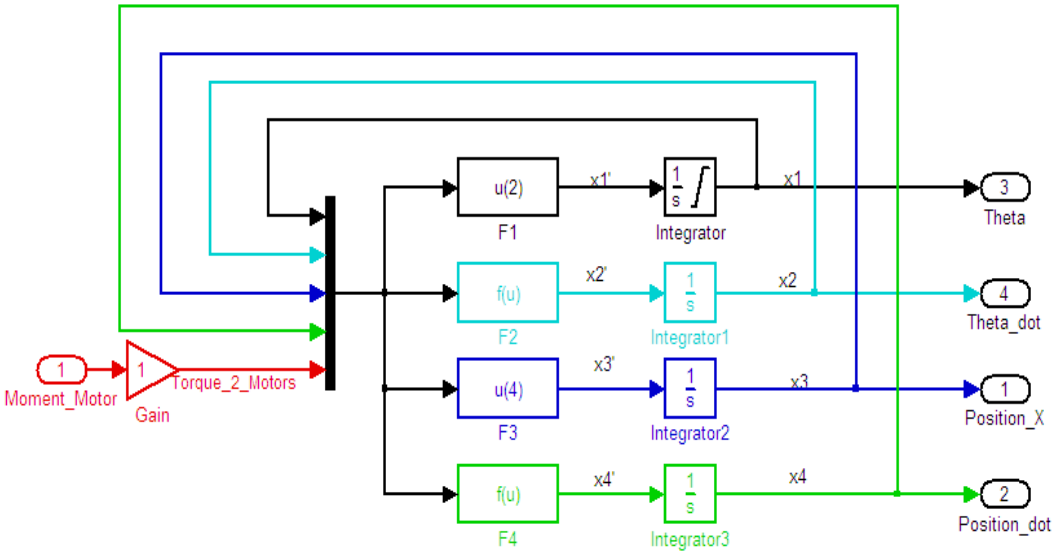
$$A_1(x_1) = \frac{\frac{3}{4}\frac{g\left(M_w R + M_B L(cosx_1)\right)(sinx_1)}{L}}{\left(2M_w + M_B - \frac{3}{4}\frac{\left(M_w R + M_B L(cosx_1)\right)(cosx_1)}{L}\right)} = f_3(x_3) \tag{3.32}$$

$$B_1(x_1, x_2) = \frac{M_B L(sinx_1)(x_2)^2}{\left(2M_w + M_B - \frac{3}{4}\frac{\left(M_w R + M_B L(cos\theta)\right)(cos\theta)}{L}\right)} = f_4(x_1, x_2) \tag{3.33}$$

$$g_1(x_1) = \frac{\frac{3}{4}\frac{(1+(sin\theta)^2)}{M_B L^2} + \frac{3}{4}\frac{(cos\theta)}{(2M_w + M_B)RL}}{\frac{3}{4}\frac{\left(M_w R + M_B L(cosx_1)\right)(cosx_1)}{(2M_w + M_B)L} - 1} \quad (3.34)$$

$$g_2(x_1) = \frac{\frac{3}{4}\frac{\left(M_w R + M_B L(cos\theta)\right)(1+(sin\theta)^2)}{M_B L^2} + \frac{1}{R}}{\left(2M_w + M_B - \frac{3}{4}\frac{\left(M_w R + M_B L(cosx_1)\right)(cosx_1)}{L}\right)} \quad (3.35)$$

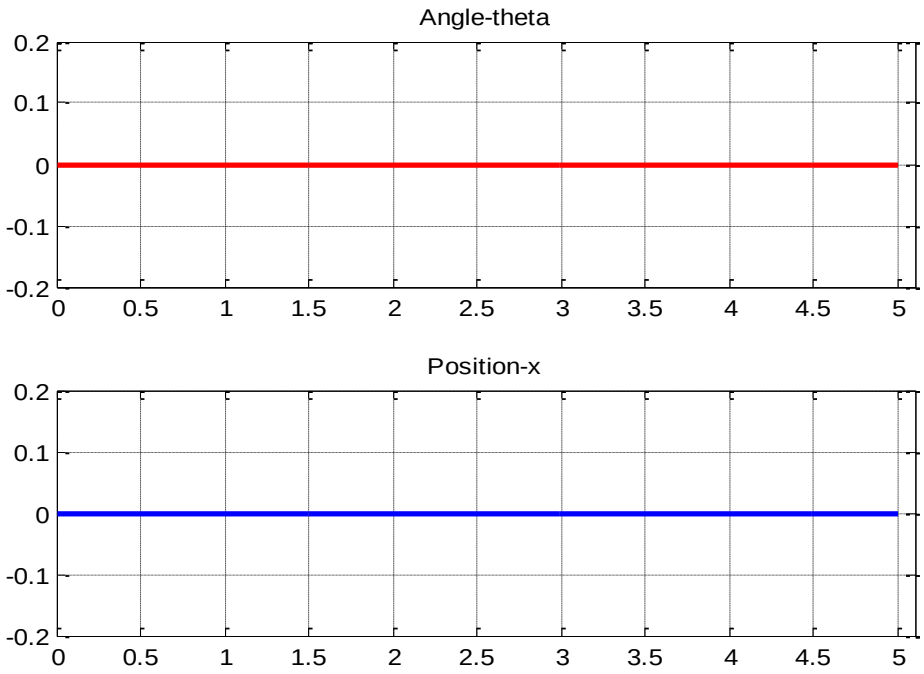
- Từ hệ phương trình (3.29) ta thực hiện mô phỏng mô hình xe hai bánh tự cân bằng trong Matlab như sau:



Hình 3.2: Sơ đồ mô hình xe hai bánh trong mô phỏng MatLab

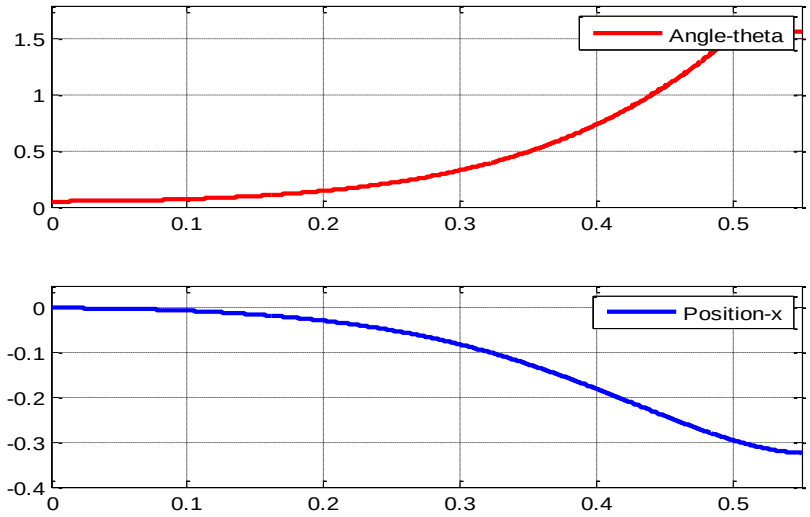
- Các trường hợp mô phỏng sau kiểm chứng kết quả mô hình toán của hệ thống:

+ Trường hợp xe thẳng đứng, không có moment của động cơ tác động vào, xét trường hợp lý tưởng (không nhiễu, không ngoại lực) thì xe sẽ luôn đứng thẳng với góc nghiêng thân xe là 0[độ] so với chiều của trọng trường, và xe không di chuyển thì vị trí bánh xe cũng là 0[m].



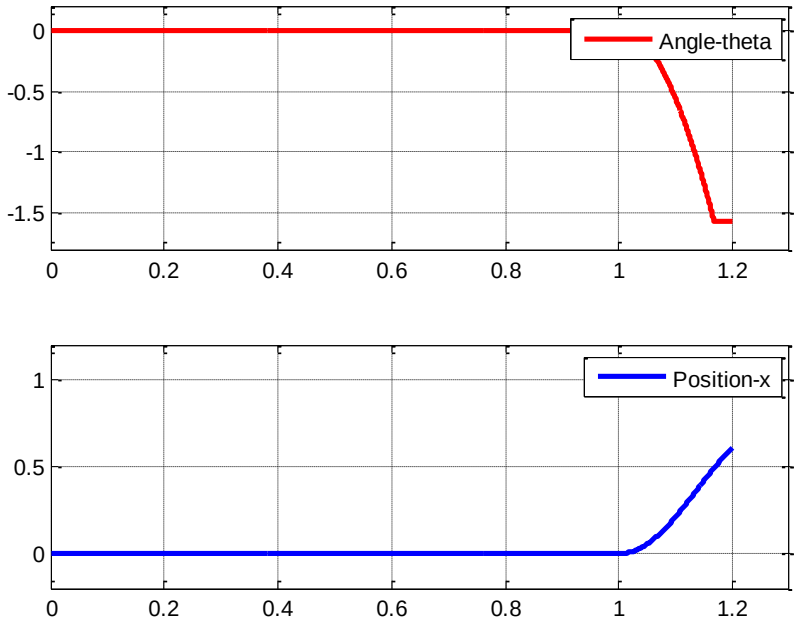
Hình 3.3: Khi xe thẳng đứng, không có moment tác động

+ Trường hợp ban đầu thân xe đã nghiêng một góc nhỏ, không có moment của động cơ tác dụng vào thì theo quán tính thân xe sẽ ngã hẳn về phía trước và phần hai bánh của xe sẽ chạy ngược chút ít về sau.



Hình 3.4: Xe nghiêng góc nhỏ, không có moment tác động

+ Ban đầu xe đang thẳng đứng (góc nghiêng thân robot là 0[độ]), thì đến lúc thời điểm 1[giây] thì có moment 1[Nm] của động cơ tác động vào xe, hai bánh xe sẽ chạy về phía trước (vị trí bánh xe robot có giá trị dương) đồng thời sẽ gây cho robot ngã ngược về phía sau (góc nghiêng thân xe có giá trị âm).



Hình 3.5: Khi có moment tác động vào tại thời điểm 1 giây

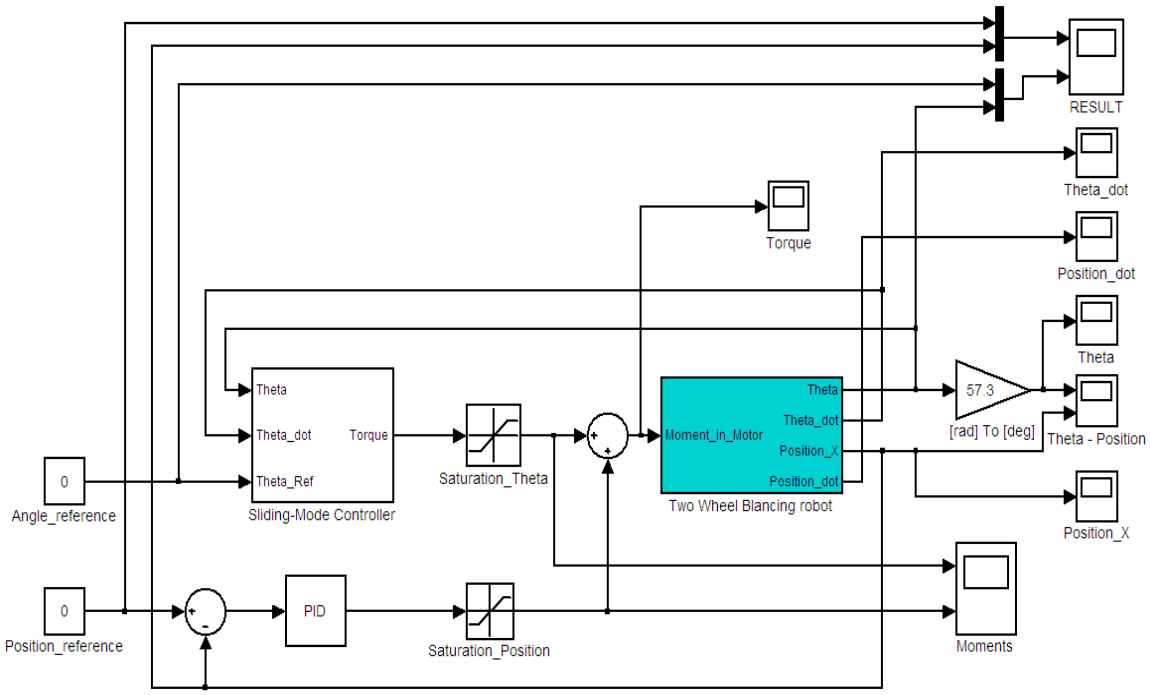
- Nhận xét: Từ các kết quả mô phỏng trên ta thấy mô hình toán xe hai bánh tự cân bằng được xây dựng trong luận văn này tương đối chính xác và phù hợp với nguyên lý hoạt động của xe hai bánh cân bằng trong thực tế.

3.2. Thiết kế bộ điều khiển trượt (Sliding mode) cho xe hai bánh cân bằng

Bộ điều khiển PD-trượt có cấu trúc gồm 2 vòng, vòng trong là bộ điều khiển PD cho vị trí bánh xe, vòng ngoài là bộ điều khiển trượt ổn định góc nghiêng thân xe.

Tương tự như trên, để đảm bảo tính ổn định hệ thống, tín hiệu ngõ ra của bộ điều khiển PD vị trí xe được giới hạn bằng 20% của tín hiệu điều khiển cuối cùng, còn 80% của tín hiệu điều khiển cuối cùng (C) là giá trị ngõ ra của bộ điều khiển trượt ổn định góc nghiêng thân xe.

Sơ đồ mô phỏng trong MatLab của hệ thống điều khiển dùng bộ điều khiển PD-trượt được trình bày ở hình sau.



Hình 3.6: Sơ đồ bộ điều khiển trượt –PD trong mô phỏng

Lưu ý: Chọn giá trị khâu bảo hòa của vị trí xe và góc nghiêng thân xe phụ thuộc vào giá trị moment cực đại của động cơ DC được sử dụng trong mô hình.

Trình tự thiết kế bộ điều khiển góc nghiêng thân robot dựa vào phương pháp điều khiển trượt như sau:

Định nghĩa sai số góc nghiêng

$$e = x_{1ref} - x_1 \tag{3.29}$$

Trong đó, $x_{1ref} = \theta_{ref}$ là tín hiệu đặt của góc nghiêng thân xe

Đạo hàm của hàm e

$$\dot{e} = \dot{x}_{1ref} - \dot{x}_1 = \dot{x}_{1ref} - x_2 \tag{3.30}$$

Đạo hàm của hàm $\dot{e}$

$$\ddot{e} = \ddot{x}_{1ref} - \ddot{x}_1 = \ddot{x}_{1ref} - \dot{x}_2 \tag{3.31}$$

Trong phương trình (3.31), tín hiệu điều khiển C (trong $\dot{x}_2$) đã xuất hiện, vì vậy ta có thể chọn mặt trượt S như sau

$$S = \tau\dot{e} + e \tag{3.32}$$

Với τ là thời gian đáp ứng mong muốn, $\tau > 0$

Đạo hàm của hàm S là

$$\dot{S} = \tau\ddot{e} + \dot{e} \tag{3.33}$$

Thế (3.30), (3.31) vào (3.33) ta được

$$\dot{S} = \tau\left(\ddot{x}_{1ref} - \dot{x}_2\right) + \left(\dot{x}_{1ref} - x_2\right)$$

$$= \tau\left(\ddot{x}_{1ref} - f_1(x_1) - f_2(x_1, x_2) - g_1(x_1)C\right) + \dot{x}_{1ref} - x_2 \quad (3.34)$$

Trong đó, $\boldsymbol{C = C_\theta + C_x}$

Nguyên lý của điều khiển trượt là làm sao cho $S.\dot{S} < 0$ và để tránh hiện tượng chattering nên ta chọn

$$\dot{S} = -Ksat(S) \quad (3.35)$$

+ K là hằng số dương. Khi K tăng, thì sự bền vững của hệ thống càng tăng, nhưng cũng làm tăng hiện tượng dao động (chattering).

+ sat(.) là hàm bão hòa, thay thế cho hàm dấu (signum) nhằm giảm hiện tượng dao động (chattering).

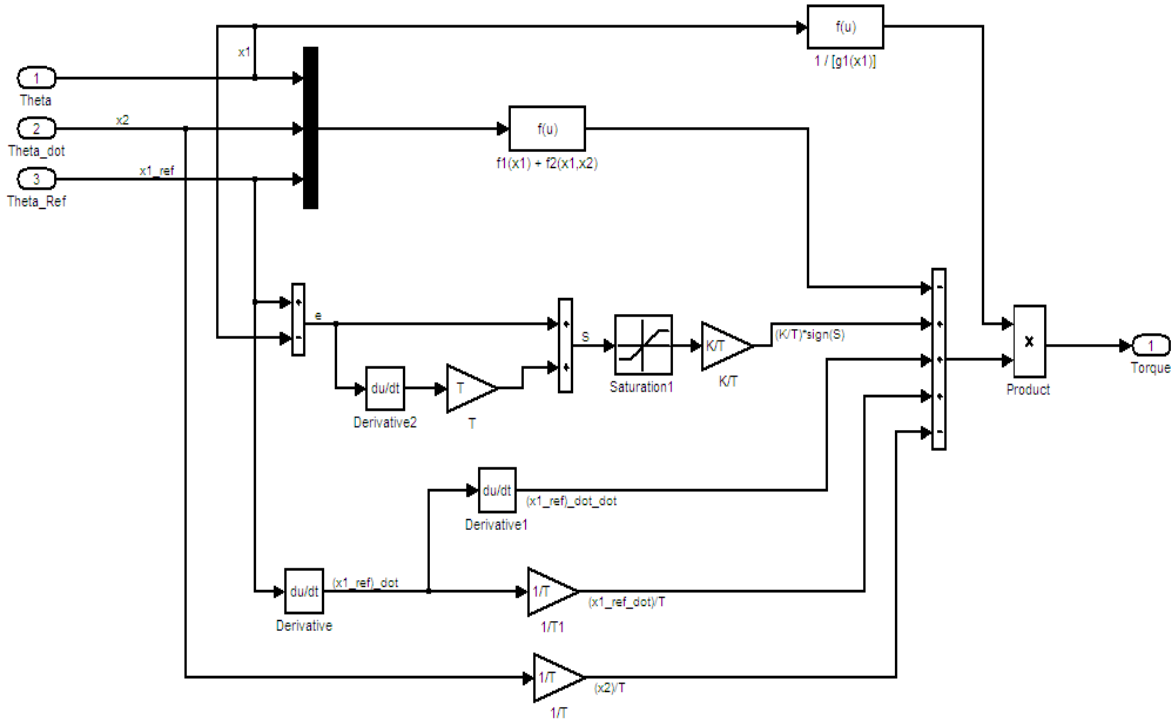
$$\mathrm{S} = \begin{cases} -0.4 & ,\text{khi S} < -0.4 \\ \mathrm{S} & ,\text{khi} -0.4 < S < 0.4 \\ 0.4 & ,\text{khi S} > 0.4 \end{cases} \quad (3.36)$$

Giá trị giới hạn của hàm sat(S) được chọn bằng ± 0.4, là dựa vào kết quả mô phỏng và đáp ứng của mô hình xe thực nghiệm.

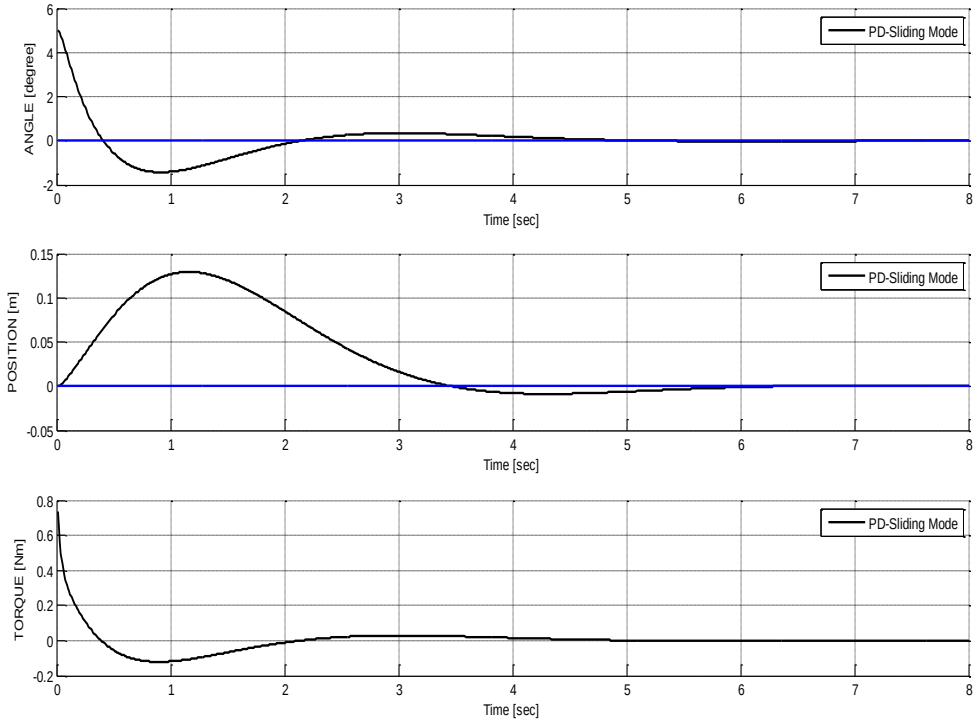
Từ (3.34) và (3.35), ta được tín hiệu điều khiển sau

$$C = \frac{\ddot{x}_{1ref} - f_1(x_1) - f_2(x_1,x_2) + \frac{K}{\tau}[sat(S)] - \frac{x_2}{\tau} + \frac{\dot{x}_{1ref}}{\tau}}{g_1(x_1)} \quad (3.37)$$

Phương trình (3.37) được mô phỏng như hình sau



Hình 3.7: Khối điều khiển trượt trong hệ thống mô phỏng



Hình 3.8: Kết quả mô phỏng bộ điều khiển PD - trượt

3.3. Đánh giá kết quả mô phỏng của hệ thống

+ Việc sử dụng bộ điều khiển trượt để ổn định góc nghiêng thân xe cho kết quả đáp ứng góc nghiêng thân xe với thời gian xác lập nhanh, độ vọt lố thấp, sai số xác lập gần như bằng 0.

+ Bộ điều khiển trượt cho góc nghiêng thân xe được thiết kế từ mô hình phi tuyến của xe nên có sự phù hợp cao đối với mô hình, điều đó giúp cho xe vẫn giữ thăng bằng được khi góc nghiêng ban đầu của thân xe lớn hơn 10 [độ].

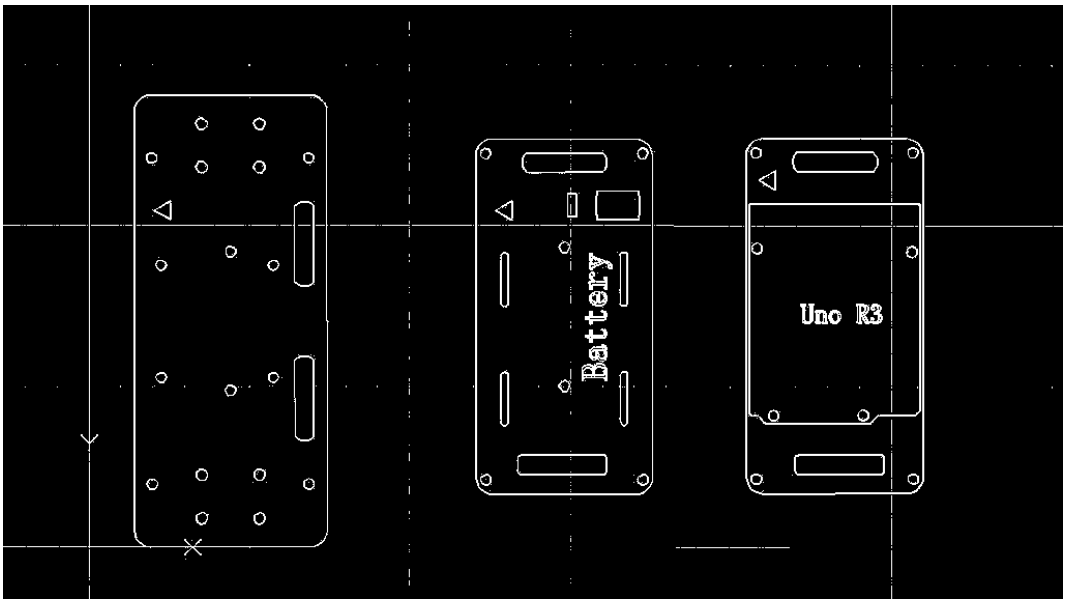
+ Khi muốn xe có thời gian đáp ứng nhanh thì ta có thể chọn hệ số K có giá trị lớn, nhưng điều đó cũng dễ gây ra hiện tượng dao động lớn ở trên mặt trượt S, đồng thời cũng sẽ tạo ra sự dao động của mô hình xe hai bánh tự cân bằng thực nghiệm.

+ Phương trình tính toán luật điều khiển của bộ điều khiển trượt là khá phức tạp nên sẽ mất nhiều thời gian tính toán của vi điều khiển trung tâm trong mô hình xe hai bánh thực nghiệm. Vấn đề này có thể được cải tiến bằng cách sử dụng bộ vi điều khiển có tốc độ cao và hỗ trợ thực hiện phép toán với số thực.

CHƯƠNG 4: THỰC NGHIỆM HỆ THỐNG

4.1 Xây dựng mô hình cơ khí

4.1.1 Thiết kế thân Robot: dùng mica 3mm



Hình 4.1 Bảng vẽ thiết kế thân robot

4.1.2 Động cơ – gá động cơ:

4.1.2.1 Động cơ

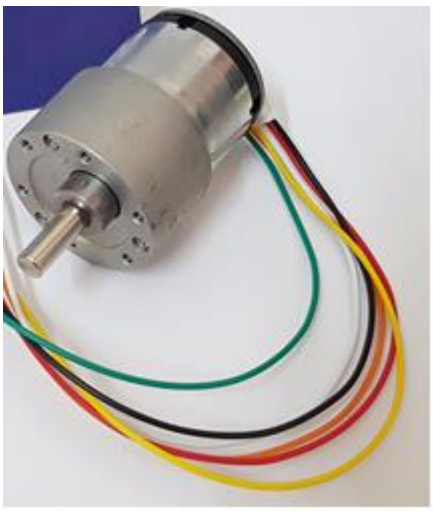
Dùng động cơ GA37V1 là loại động cơ có hộp số và Encoder . Encoder sử dụng từ trường với lực từ mạnh giúp phản hồi xung chính xác.

Thông số động cơ:

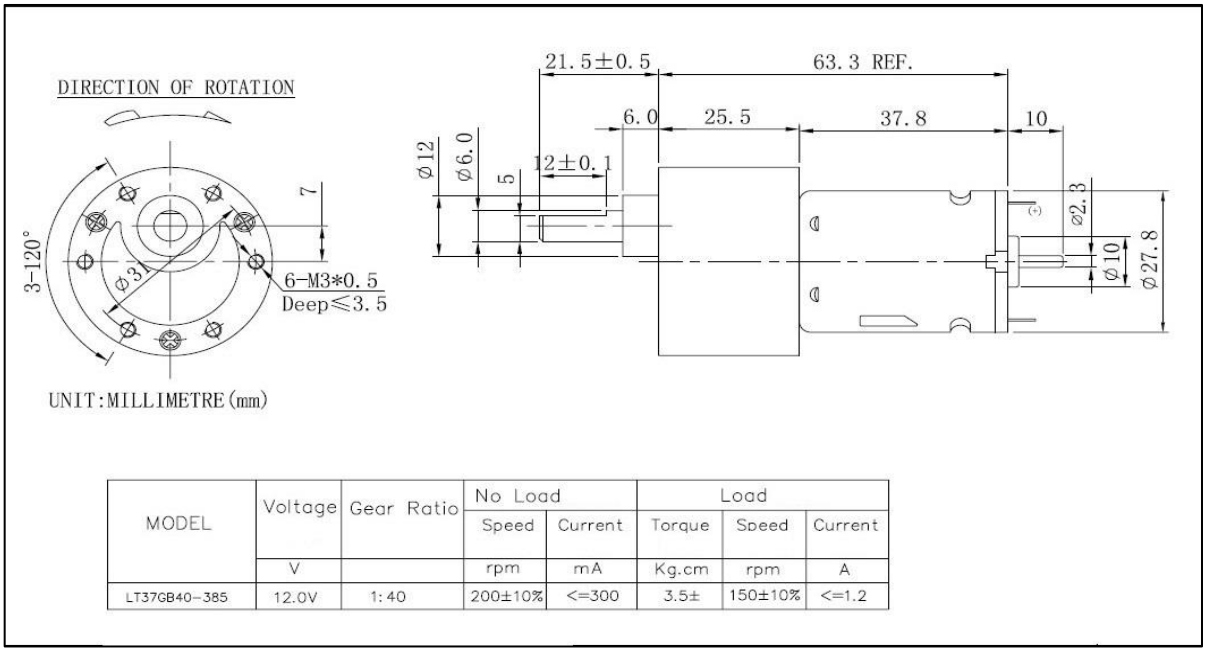
- Điện áp định mức: 12Vdc
- Dòng tối đa: 3A
- Tốc độ định mức trước hộp số giảm tốc: 10.000RPM
- Tỉ số truyền: 1:33
- Đường kính trục: 6mm

Thông số Encoder:

- Điện áp: 3.3Vdc
- Encoder: Đĩa từ 11 xung, 2 kênh AB
- Số xung sau giảm tốc: 363xung/vòng



Hình 4.1 Động cơ GA37V1



MODEL	Voltage	Gear Ratio	No Load		Load		
			Speed	Current	Torque	Speed	Current
	V		rpm	mA	Kg.cm	rpm	A
LT37GB40-385	12.0V	1:40	200±10%	<=300	3.5±	150±10%	<=1.2

Hình 4.2 Bảng thông số động cơ GA37V1

4.1.2.1 Gá động cơ

Hình 4.3 Gá gắn động cơ GA37V1

4.1.3 Bánh xe

Hình 4.4 Bánh xe V2 65mm

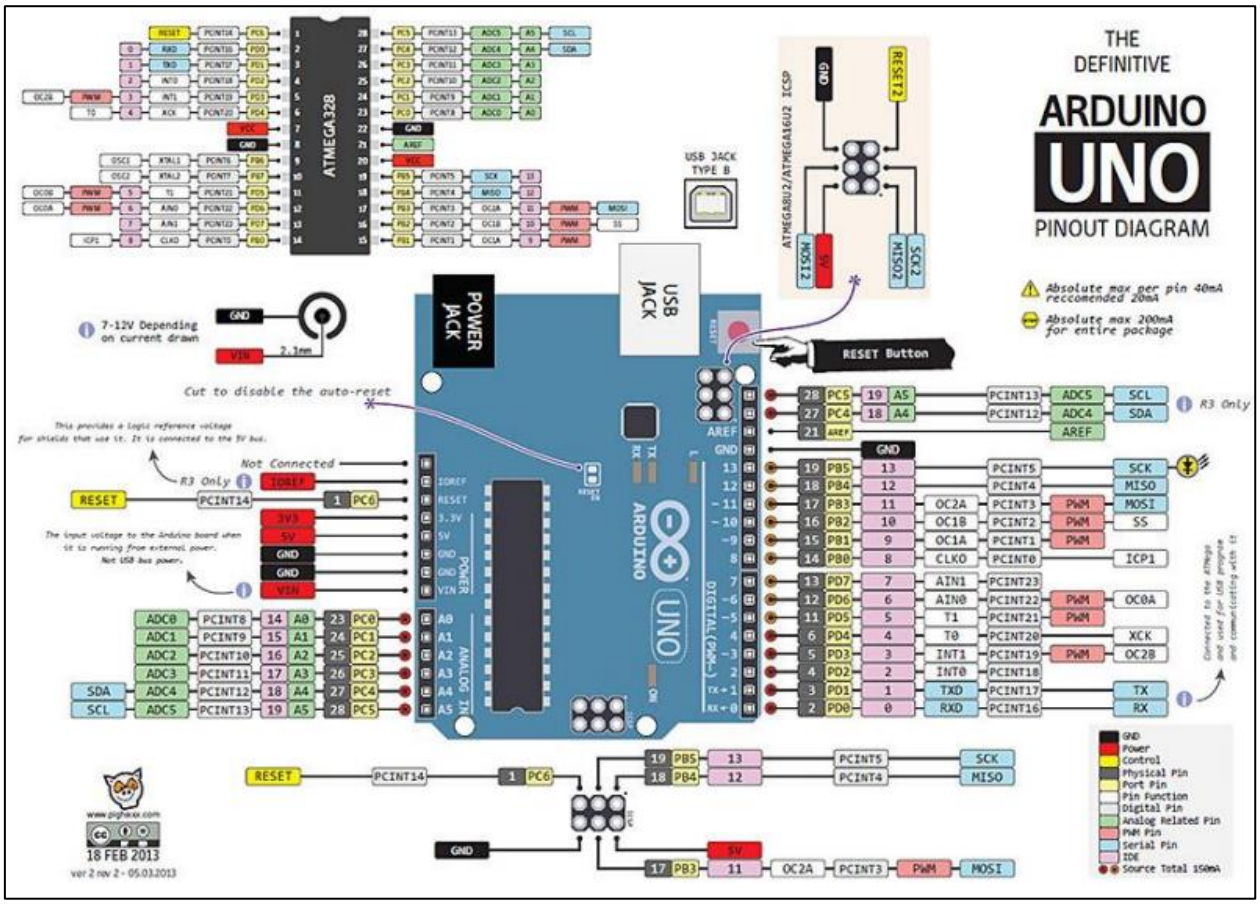
Thông số bánh xe:

- Đường kính bánh xe: 65mm
- Độ dày lốp xe: 6.5mm
- Chiều rông bánh xe: 27mm
- Trục xe: phi 5mm.

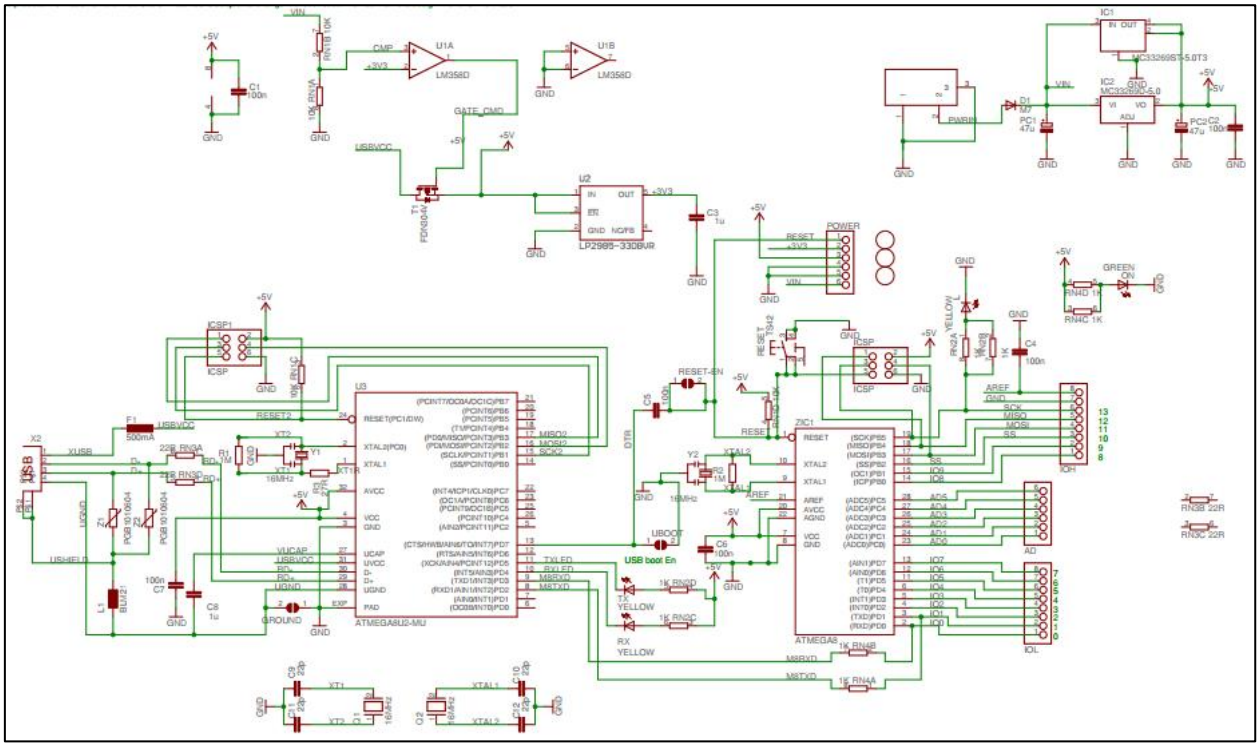
4.2 Nguyên lý phần cứng

4.2.1 Board Arduino uno

Hình 4.5 Board Arduino Uno



Hình 4.6 Cấu trúc board Arduino Uno



Hình 4.7 Sơ đồ nguyên lí board Arduino Uno

Thông số kỹ thuật:

Vi điều khiển: ATmega328 họ 8bit

Điện áp hoạt động: 5VDC

Tần số hoạt động: 16 MHz

Dòng tiêu thụ: khoảng 30mA

Điện áp vào giới hạn: 6-20V DC

Số chân Digital I/O: 14 (6 chân hardware PWM)

Số chân Analog: 6 (độ phân giải 10bit)

Dòng tối đa trên mỗi chân I/O: 30 mA

Dòng ra tối đa (5V): 500 mA

Dòng ra tối đa (3.3V): 50 mA

Bộ nhớ flash: 32 KB (ATmega328) với 0.5KB dùng bởi bootloader

SRAM: 2 KB (ATmega328)

EEPROM: 1 KB (Atmega328)

4.2.2 Cảm biến MPU6050

MPU-6050 là cảm biến của hãng InvenSense. MPU-6050 là một trong những giải pháp cảm biến chuyển động đầu tiên trên thế giới có tới 6 (mở rộng tới 9) trục cảm biến tích hợp trong 1 chip duy nhất.

MPU-6050 tích hợp 6 trục cảm biến bao gồm:

- Con quay hồi chuyển 3 trục (3-axis MEMS gyroscope)
- Cảm biến gia tốc 3 chiều (3-axis MEMS accelerometer)

Ngoài ra, MPU-6050 còn có 1 đơn vị tăng tốc phần cứng chuyên xử lý tín hiệu (Digital Motion Processor – DSP) do cảm biến thu thập và thực hiện các tính toán cần thiết. Điều này giúp giảm bớt đáng kể phần xử lý tính toán của vi điều khiển, cải thiện tốc độ xử lý và cho ra phản hồi nhanh hơn. Đây chính là 1 điểm khác biệt đáng kể của MPU-6050 so với các cảm biến gia tốc và gyro khác.

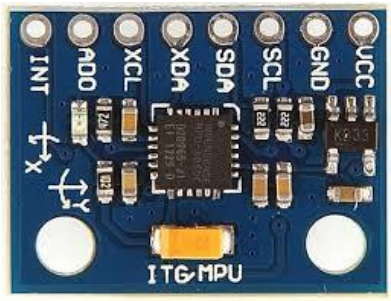
MPU-6050 có thể kết hợp với cảm biến từ trường (bên ngoài) để tạo thành bộ cảm biến 9 góc đầy đủ thông qua giao tiếp I2C.

Các cảm biến bên trong MPU-6050 sử dụng bộ chuyển đổi tương tự - số (Anolog to Digital Converter – ADC) 16-bit cho ra kết quả chi tiết về góc quay, tọa độ… Với 16-bit bạn sẽ có 2^16 = 65536 giá trị cho 1 cảm biến.

Tùy thuộc vào yêu cầu của bạn, cảm biến MPU-6050 có thể hoạt động ở chế độ tốc độ xử lý cao hoặc chế độ đo góc quay chính xác (chậm hơn). MPU-6050 có khả năng đo ở phạm vi:

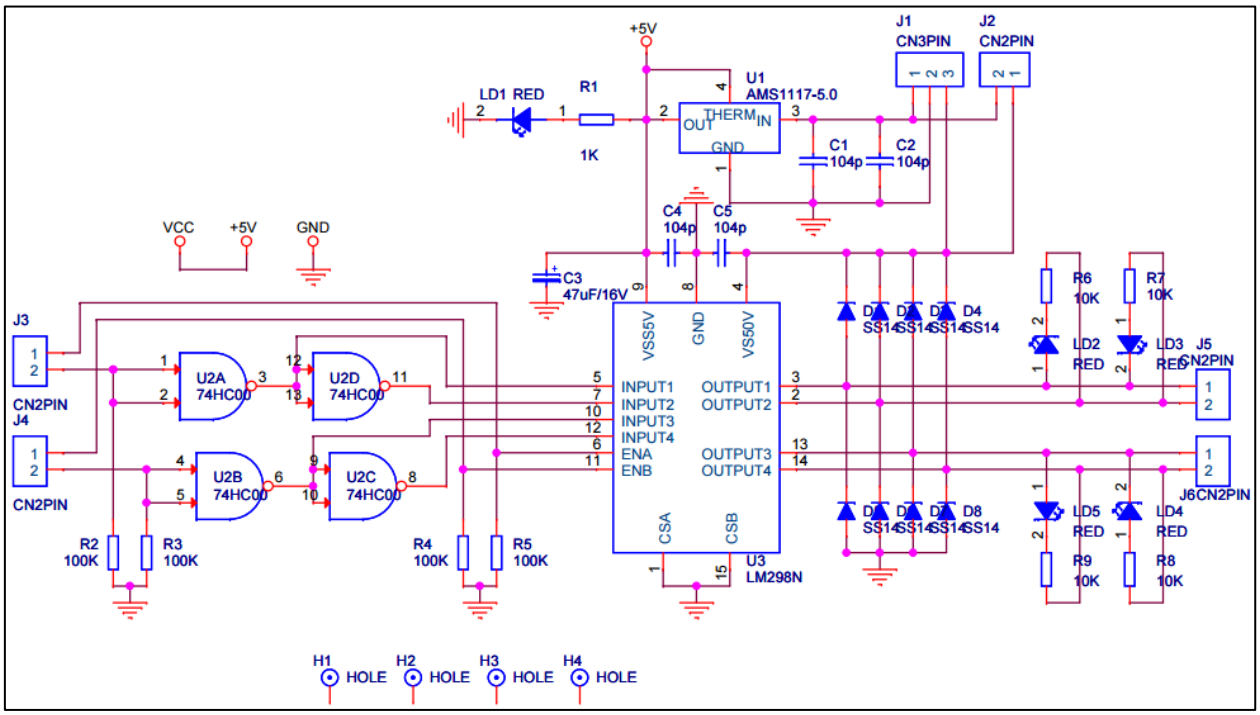
- con quay hồi chuyển: ± 250 500 1000 2000 dps
- gia tốc: ± 2 ± 4 ± 8 ± 16g

Hơn nữa, MPU-6050 có sẵn bộ đệm dữ liệu 1024 byte cho phép vi điều khiển phát lệnh cho cảm biến, và nhận về dữ liệu sau khi MPU-6050 tính toán xong.

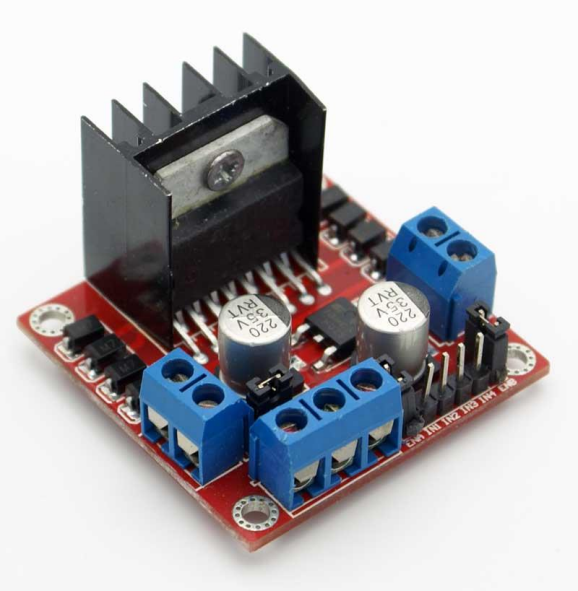


Hình 4.8 Board cảm biến MPU6050

4.2.3 Board công suất L298



Hình 4.9 Sơ đồ nguyên lí Board công suất L298



Hình 4.10 Board công suất L298

Các thông số Kỹ Thuật

Driver: L298

Driver power supply: +5V~+46V

Driver Io: 2A

Logic power output Vss: +5~+7V (internal supply +5V)

Logic current: 0~36mA

Controlling level: Low -0.3V~1.5V, high: 2.3V~Vss

Enable signal level: Low -0.3V~1.5V, high: 2.3V~Vss

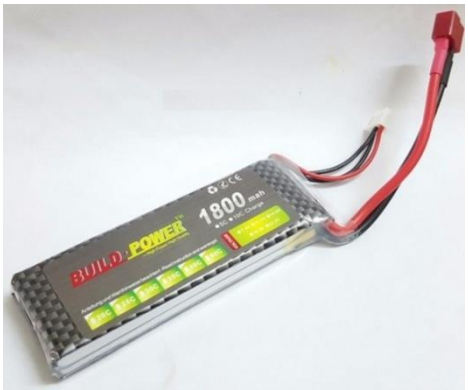
Max power: 25W (Temperature 75 cesus)

Working temperature: -25C~+130C

Dimension: 42mm * 50 mm

Driver weight: ~48g

4.2.4 Pin Lipo



Pin Li-Po (viết tắt của Lithium Polymer):

Các thông số Kỹ Thuật

Chất liệu pin: Lithium Polymer

Số Cell pin: 3S

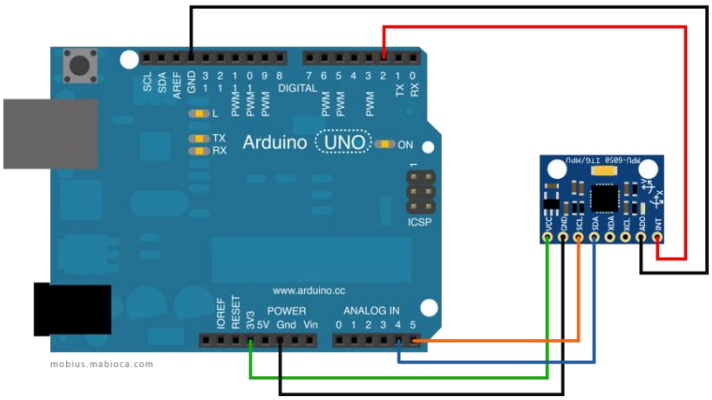
Điện áp trung bình: 11.1VDC

Dung lượng: 1800mAh

Dòng xả tối đa: 25C x 1800mAh/1000= 45A

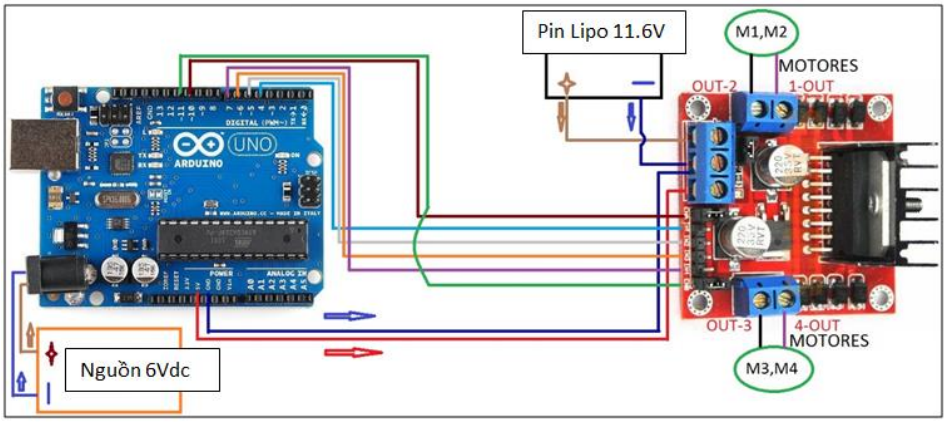
Kích thước: W 33 x L 105 x H 12mm

4.2.5 Kết nối Board UNO – Cảm biến MPU 6050



Hình 4.11 Kết nối Board UNO – Cảm biến MPU 6050

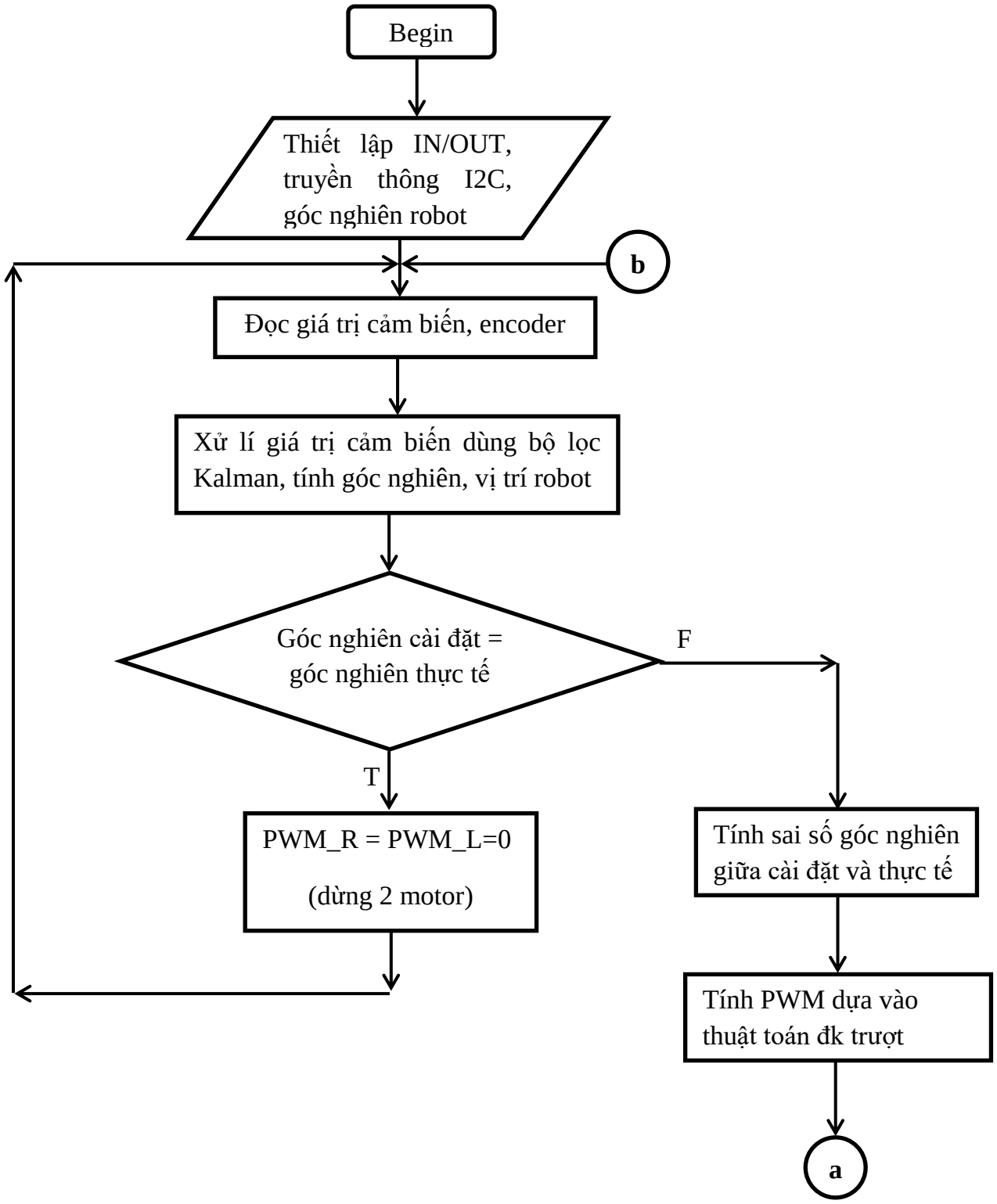
4.2.6 Kết nối Board UNO – Board công suất L298

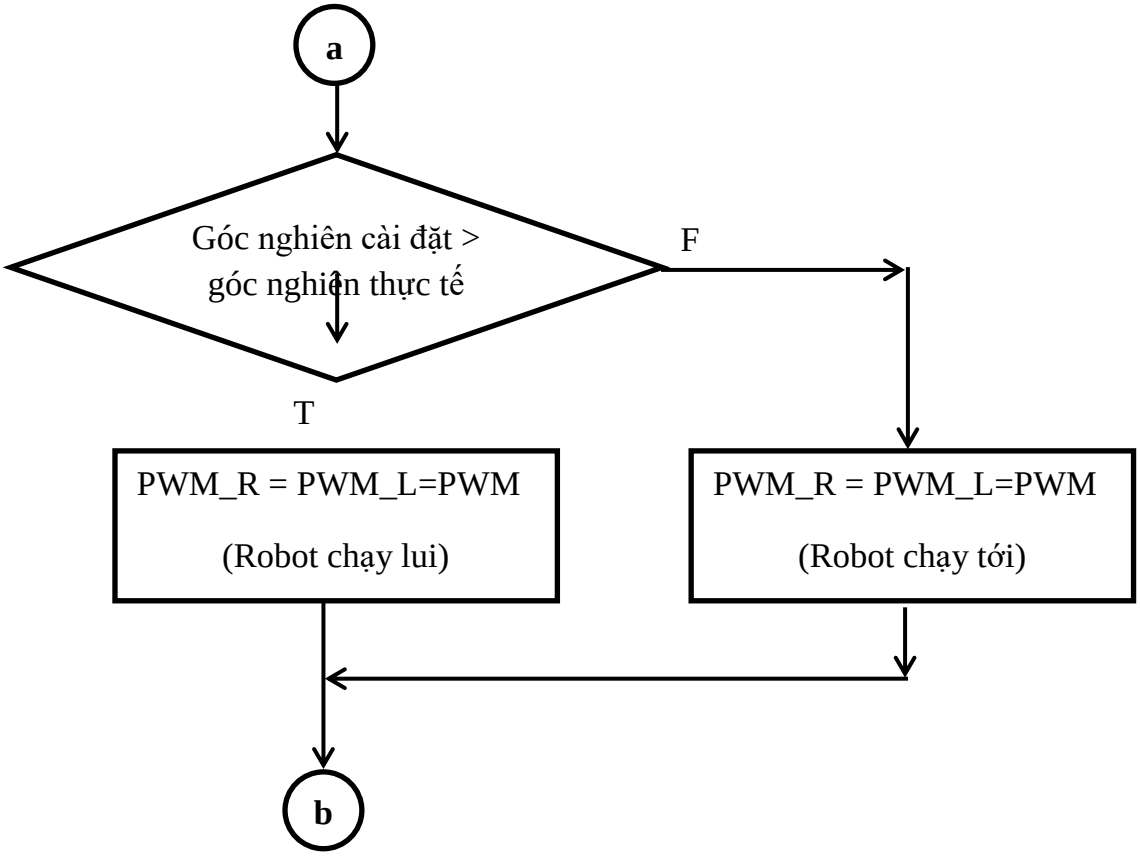


Hình 4.12 Kết nối Board UNO – Board công suất L298

4.3 Xây dựng giải thuật điều khiển dựa vào lý thuyết

Lưu đồ giải thuật

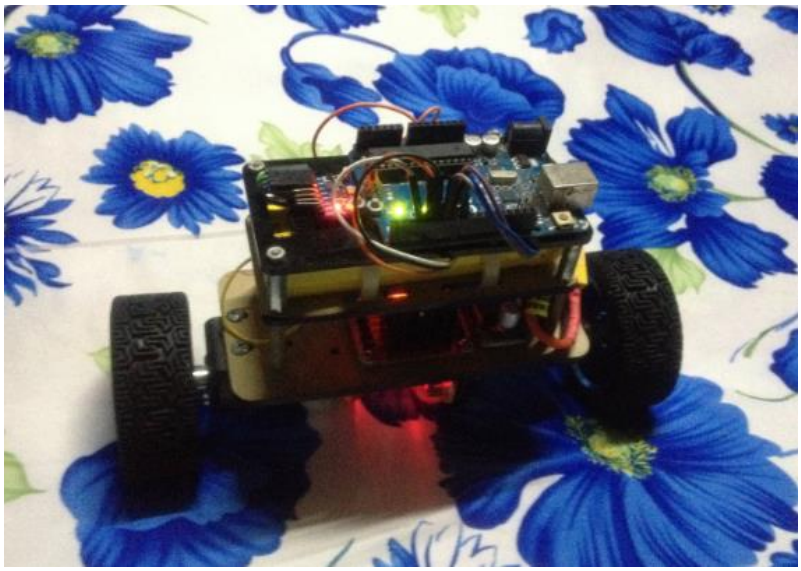




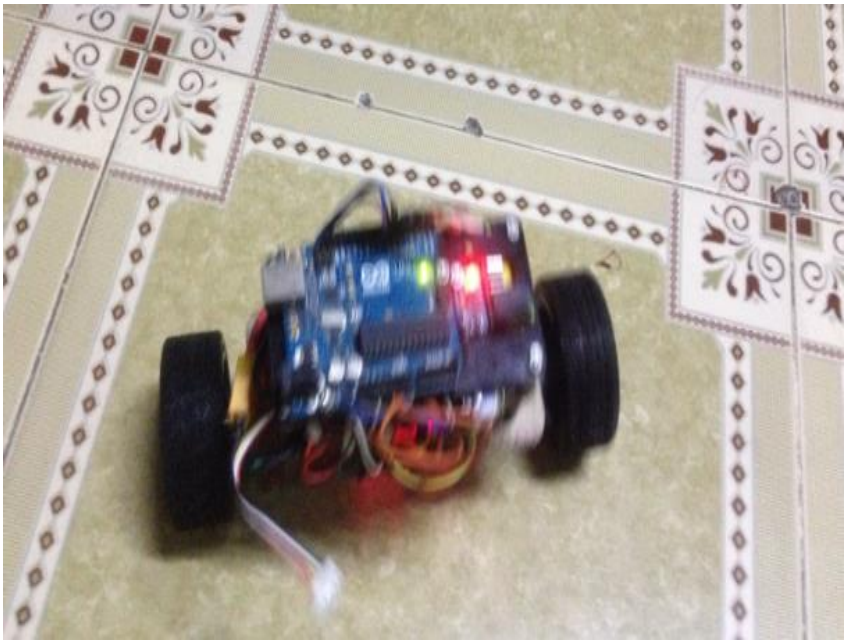
Hình 4.13 Lưu Đồ giải thuật

4.4 Một số hình ảnh thực nghiệm

Hình 4.14 Cân khối lượng Robot



Hình 4.15 Robot hoạt động trên mặt phẳng mềm



Hình 4.16 Robot hoạt động trên mặt phẳng cứng

Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận.

Với mục tiêu đặt ra trong quá trình thực hiện đề tài và qua quá trình làm việc đã đạt được một số kết quả nghiên cứu như sau:

- Nghiên cứu lý thuyết:

Đã có sự nghiên cứu và nắm bắt một cách tổng thể về nguyên lý và lý thuyết tổng quan nhất về robot cân bằng.

- Phần thiết kế và thi công mô hình:

Thiết kế, chế tạo cơ khí hoàn chỉnh mô hình đảm bảo độ bền. Thiết kế mô hình thực chính xác về kích thước và khá tối ưu về khối lượng, tận dụng được nguồn thiết bị và vật liệu có sẵn trên thị trường.

Robot cân bằng có khả năng chịu tải trọng tĩnh và tải trọng động tốt. Mô hình đảm bảo được yêu cầu kỹ thuật và thẩm mỹ.Tuy vậy, việc đưa ra các thông số về kết cấu cơ khí và công suất động cơ còn dựa nhiều vào kinh nghiệm thực tế.

Thiết kế, chế tạo mạch điều khiển: Mạch được thiết kế trên phần mềm chuyên dụng Orcad, mạch trung tâm thiết kế nhỏ gọn có tính thẩm mĩ cao, kích thước mạch nhỏ gọn.

- Về điều khiển:

Xử lý cảm biến, lấy được dữ liệu và xử lý được dữ liệu phục vụ cho việc điều khiển cân bằng qua bộ lọc số Kalman.

Chương trình điều khiển đã được viết hoàn chỉnh và linh hoạt, Robot cân bằng quay trái, phải di chuyển sang trái, phải, trước, sau.

5.2 Hướng phát triển

Tích hợp thêm module Bluetooth, wifi, RF, ... Ngoài ra, việc tích hợp thêm một số thiết bị phụ trợ phục vụ cho từng loại nhiệm vụ cũng cần được quan tâm như việc lắp thêm camera cho Robot cân bằng nhằm ghi lại các đặc điểm địa hình hay các cảm biến siêu âm để dò tìm mục tiêu…

Phát triển đề tài với các ứng dụng trong quân sự, có thể mang các thiết bị quân sự.

Hiện nay ở Việt Nam đã bắt đầu có nhu cầu về loại đồ chơi công nghệ cao dạng Robot cân bằng, vì thế việc nghiên cứu để hoàn thiện và thương mại hóa loại sản phẩm này là một hướng phát triển tốt.

TÀI LIỆU THAM KHẢO

[1] S.W.Nawawi , M.N.Ahmad, J.H.S. Osman , "Development of Two-Wheeled Inverted Pendulum Mobile Robot", SCOReD07,Malaysia, Dec 2007, pp153-158.

[2] Felix Grasser, Aldo D'Arrigo, Silvio Colombi, Alfred Ruffer, "JOE: A Mobile Inverted Pendulum", IEEE Trans. Electronics, vol. 49, Feb 2002, pp. 107-114.

[3] Greg Welch, Gary Bishop; "An Introduction to the Kalman Filter"; Siggraph 2001 Conference, pp. 8-15.

[4] Tatsuya Nomura, Yuta Kitsuka, Hauro Suemistu, Takami Matsuo, "Adaptive Backstepping Control for a Two-Wheeled Autonomous Robot", Proceedings of ICROS-SICE International Joint Conference 2009, Aug. 2009, pp 4687 - 4692.

[5] Xiaogang Ruan, Jianxian Cai, "Fuzzy Backstepping Controller for Two-Wheeled Self-Balancing Robot", Proceedings of 2009 International Asia Conference on Informatics in Control, Automation and Robotics.

[6] Shui-Chun Lin, Ching-Chih Tsai, Hsu-Chih Huang, "Nonlinear adaptive sliding mode control design for two-wheeled human transportation vehicle", Proceedings of Systems Man and Cybernetics 2009, IEEE International Conference on, pp 1965 – 1970.

[7] Arbin Ebrahim, Gregory V.Murphy, "Adaptive Backstepping Controller Design of an Inverted Pendulum", Proceedings of the Thirty-Seventh Southeastern Symposium on, 2005, pp. 172-174.

[8] Jian Huang, Hongwei Wang,T. Matsuno, T. Fukuda, K. Sekiyama, "Robust velocity sliding mode control of mobile wheeled inverted pendulum systems", Proceedings of Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pp 2983 – 2988.

[9] Dianwei Qian, Jianqiang Yi, Dongbin Zhao, "Hierarchical Sliding Mode Control to Swing up a Pendubot", Proceedings of American Control Conference, 2007, ACC '07, pp 5254 – 5259.

[10] D. Y. Lee, Y. H. Kim, B. S. Kim, Y. K. Kwak, "Dynamics and Control of Non-holonomic Two Wheeled Inverted Pendulum Robot", Proceedings of the eighth International Symposium on Artificial Life and Robotics, 2003.

[11] Jingtao Li, Xueshan Gao, Qiang Huang, Matsumoto, "Controller design of a two-wheeled inverted pendulum mobile robot", Proceedings of Mechatronics and Automation 2008, ICMA 2008, IEEE International Conference on.

[12] S.W.Nawawi, M.N.Ahmad, J.H.S. Osman, "Real-Time Control of Two-Wheeled Inverted Pendulum Mobile Robot", www.waset.org/journals/waset.

[13] Lv Qiang, Wang Ke-ke, Wang Guo-sheng; "Research of LQR controller based on Two-wheeled self-balancing robot"; Proceedings of Control and Decision Conference 2009, CCDC '09, pp. 2343 – 2348.

[14] Ruan Xiaogang, Liu Jiang, Di Haijiang, Li Xinyuan; " Design and LQ Control of a two-wheeled self-balancing robot"; Proceedings of Control Conference, 2008. CCC 2008, pp. 275 – 279.

[15] Ching-Chih Tsai, Hsu-Chih Huang, Shui-Chun Lin; " Adaptive Neural Network Control of a Self-Balancing Two-Wheeled Scooter"; Industrial Electronics Journal, IEEE Transactions on, April 2010, pp. 1420 – 1428.

[16] Wei-Song Lin Tien, G. Chia-Hsiang Tu; "Adaptive Critic Neuro-Fuzzy Control of Two-wheel Vehicle"; Proceedings of 2006 IEEE International Conference on, pp. 445 – 450.

[17] Dương Hoài Nghĩa; "Điều khiển hệ thống đa biến"; Nhà xuất bản Đại học Quốc Gia TP.HCM, 2007.

[18] Nguyễn Thi Phương Hà; " Lý Thuyết Điều Khiển Hiện Đại"; Nhà xuất bản Đại học Quốc Gia TP.HCM, 2009.

[19] www.bkinnovation.com/bkbot/

[20] http://www.geology.smu.edu/~dpa-www/robo/nbot/

[21] www.segway.com

[22] www.freescale.com

[23] www.st.com

[24] Huỳnh Thái Hoàng; " Mô Hình Hóa Và Nhận Dạng Hệ Thống"; Đại học bách khoa Tp. HCM.

[25] Nguyễn Phùng Quang; " Matlab và Simulink"; NXB khoa học và kỹ thuật.

[26] Nguyễn Gia Minh Thảo; "A PID Backstepping Controller For Two-Wheeled Self-Balancing Robot"; Đại học bách khoa Tp. HCM.

PHỤ LỤC

Code chương trình

```
#include <Wire.h>
#include "Kalman.h"
#include "ComPacket.h"
#include "I2C.h"
#include <EEPROM.h>
//#include "music.h"
#include "MyEEprom.h"
#include "PinChangeInt.h"
#define UP_DOWN_IN_PIN   16
#define  LEFT_RIGHT_IN_PIN  17
#define  AUX_IN_PIN 14
bool RCWork = false;
volatile uint8_t bUpdateFlagsRC = 0;
#define UP_DOWN_FLAG   0b10
#define LEFT_RIGHT_FLAG 0b100
uint32_t UpDownStart;
uint32_t LeftRightStart;
volatile uint16_t UpDownEnd = 0;
volatile uint16_t LeftRightEnd = 0;
int UpDownIn;
int LeftRightIn;
Kalman kalmanX;
Kalman kalmanY;
ComPacket SerialPacket;
double accX, accY, accZ;
double gyroX, gyroY, gyroZ;
int16_t tempRaw;
double gyroXangle, gyroYangle; // Angle calculate using the gyro only
double compAngleX, compAngleY; // Calculated angle using a complementary filter
double kalAngleX, kalAngleY; // Calculated angle using a Kalman filter
```

```
uint32_t timer;
uint8_t i2cData[14]; // Buffer for I2C data
#define BUZZER 13
#define LED 13
//motor  define
#define PWM_L 6  //M1
#define PWM_R 5   //M2
#define DIR_L1 8
#define DIR_L2 7
#define DIR_R1 3
#define DIR_R2 4
//encoder define
#define SPD_INT_R 11   //interrupt
#define SPD_PUL_R 12
#define SPD_INT_L 10   //interrupt
#define SPD_PUL_L 9
int Speed_L,Speed_R;
int Position_AVG;
int Position_AVG_Filter;
int Position_Add;
int pwm,pwm_l,pwm_r;
double Angle_Car;
double Gyro_Car;
double KA_P,KA_D;
double KP_P,KP_I;
double K_Base;
struct EEpromData SavingData;
struct EEpromData ReadingData;
int Speed_Need,Turn_Need;
int Speed_Diff,Speed_Diff_ALL;
uint32_t LEDTimer;
bool blinkState = false;
```

```
uint32_t calTimer;
uint32_t delayTimer;
uint32_t  lampTimer;
byte speakMode;
void setup() {
  Serial.begin(9600);
  Init();
  Wire.begin();
  TWBR = ((F_CPU / 400000L) - 16) / 2; // Set I2C frequency to 400kHz
  i2cData[0] = 7;
  i2cData[1] = 0x00;
  i2cData[2] = 0x00;
  i2cData[3] = 0x00;
  while (i2cWrite(0x19, i2cData, 4, false));
  while (i2cWrite(0x6B, 0x01, true));
  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) {
    Serial.print(F("Error reading sensor"));
    while (1);
  }
  delay(200); // Wait for sensor to stabilize
  while (i2cRead(0x3B, i2cData, 6));
  accX = (i2cData[0] << 8) | i2cData[1];
  accY = (i2cData[2] << 8) | i2cData[3];
  accZ = (i2cData[4] << 8) | i2cData[5];
#ifdef RESTRICT_PITCH // Eq. 25 and 26
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif
```

```
  kalmanX.setAngle(roll);
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;
   Serial.print("RCWork: "); Serial.println(RCWork);
  if(RCWork == true)
  {
    PCintPort::attachInterrupt(UP_DOWN_IN_PIN, calcUpDown,CHANGE);
    PCintPort::attachInterrupt(LEFT_RIGHT_IN_PIN, calcLeftRight,CHANGE);
  }
  PCintPort::attachInterrupt(SPD_INT_L, Encoder_L,FALLING);
  PCintPort::attachInterrupt(SPD_INT_R, Encoder_R,FALLING);
  timer = micros();
  LEDTimer = millis();
  lampTimer = millis();
}
void loop() {
  double DataAvg[3];
  double AngleAvg = 0;
  DataAvg[0]=0; DataAvg[1]=0; DataAvg[2]=0;
  while(1)
    {
      if(UpdateAttitude())
      {
        DataAvg[2] = DataAvg[1];
        DataAvg[1] = DataAvg[0];
        DataAvg[0] = Angle_Car;
        AngleAvg = (DataAvg[0]+DataAvg[1]+DataAvg[2])/3;
        if(AngleAvg < 40 || AngleAvg > -40){
          PWM_Calculate();
```

```
        Car_Control();
        Serial.print(pwm_l);
        Serial.print("       ");
        Serial.print(pwm_r);
        Serial.print("       ");
        Serial.println(          Angle_Car);
       }
     }
     UserComunication();
      ProcessRC();
  }
}
bool StopFlag = true;
void PWM_Calculate()
{
  float ftmp = 0;
  ftmp = (Speed_L + Speed_R) * 0.5;
  if( ftmp > 0)
    Position_AVG = ftmp +0.5;
   else
Position_AVG = ftmp -0.5;
Speed_Diff = Speed_L - Speed_R;
Speed_Diff_ALL += Speed_Diff;
Position_AVG_Filter = Position_AVG;
Position_Add += Position_AVG_Filter;
Position_Add += Speed_Need;
Position_Add = constrain(Position_Add, -800, 800);
pwm =   (Angle_Car-8 + K_Base)* KA_P    + Gyro_Car * KA_D +   Position_Add *
KP_I +  Position_AVG_Filter * KP_P;
  if(pwm>255) pwm=255;
  if(pwm<-255) pwm=255;
 if((Speed_Need != 0) && (Turn_Need == 0))
```

```
  {
    if(StopFlag == true)
    {
      Speed_Diff_ALL = 0;
      StopFlag = false;
    }
    pwm_r = int(pwm + Speed_Diff_ALL);
    pwm_l = int(pwm - Speed_Diff_ALL);
    }
  else
  {
    StopFlag = true;
     pwm_r = pwm + Turn_Need;
     pwm_l = pwm - Turn_Need;
  }
 Speed_L = 0;
 Speed_R = 0;
}
void Car_Control()
{
 if (pwm_l<0)
  {
    digitalWrite(DIR_L1, HIGH);
    digitalWrite(DIR_L2, LOW);
    pwm_l =- pwm_l;  //cchange to positive
  }
  else
  {
    digitalWrite(DIR_L1, LOW);
    digitalWrite(DIR_L2, HIGH);
  }
  if (pwm_r<0)
```

```
  {
    digitalWrite(DIR_R1, LOW);
    digitalWrite(DIR_R2, HIGH);
    pwm_r = -pwm_r;
  }
  else
  {
    digitalWrite(DIR_R1, HIGH);
    digitalWrite(DIR_R2, LOW);
  }
  if( Angle_Car > 45 || Angle_Car < -45 )
  {
    pwm_l = 0;
    pwm_r = 0;
  }
  analogWrite(PWM_L, pwm_l>255? 255:pwm_l);
  analogWrite(PWM_R, pwm_r>255? 255:pwm_r);
}
void UserComunication()
{
  MySerialEvent();
  if(SerialPacket.m_PackageOK == true)
  {
    SerialPacket.m_PackageOK = false;
    switch(SerialPacket.m_Buffer[4])
    {
      case 0x01:break;
      case 0x02: UpdatePID(); break;
      case 0x03: CarDirection();break;
      case 0x04: SendPID();break;
      case 0x05:
                SavingData.KA_P = KA_P;
```

```
                SavingData.KA_D = KA_D;
                SavingData.KP_P = KP_P;
                SavingData.KP_I = KP_I;
                SavingData.K_Base = K_Base;
                WritePIDintoEEPROM(&SavingData);
                break;
      case 0x06:  break;
      case 0x07:break;
      default:break;
    }
  }
}
void UpdatePID()
{
  unsigned int Upper,Lower;
  double NewPara;
  Upper = SerialPacket.m_Buffer[2];
  Lower = SerialPacket.m_Buffer[1];
  NewPara = (float)(Upper<<8 | Lower)/100.0;
  switch(SerialPacket.m_Buffer[3])
  {
          case 0x01:KA_P = NewPara;
          Serial.print("Get KA_P: \n");
          Serial.println(KA_P);break;
          case 0x02:KA_D = NewPara;
          Serial.print("Get KA_D: \n");
          Serial.println(KA_D);break;
          case 0x03:
          KP_P = NewPara;
          Serial.print("Get KP_P: \n");
          Serial.println(KP_P);break;
case 0x04:
```

```
KP_I = NewPara;
Serial.print("Get KP_I: \n");
Serial.println(KP_I);break;
case 0x05:
K_Base = NewPara;
Serial.print("Get K_Base: \n");
Serial.println(K_Base);break;
default:break;
  }
}

void CarDirection()
{
  unsigned char Speed = SerialPacket.m_Buffer[1];
  switch(SerialPacket.m_Buffer[3])
  {
    case 0x00: Speed_Need = 0;Turn_Need = 0;break;
    case 0x01: Speed_Need = -Speed; break;
    case 0x02: Speed_Need = Speed; break;
    case 0x03: Turn_Need = Speed; break;
    case 0x04: Turn_Need = -Speed; break;
    default:break;
  }
}
void SendPID()
{
  static unsigned char cnt = 0;
  unsigned char data[3];
  switch(cnt)
  {
    case 0:
    data[0] = 0x01;
```

```
   data[1] = int(KA_P*100)/256;
   data[2] = int(KA_P*100)%256;
   AssemblyAndSend(0x04,data);
   cnt++;break;
   case 1:
    data[0] = 0x02;
   data[1] = int(KA_D*100)/256;
   data[2] = int(KA_D*100)%256;
   AssemblyAndSend(0x04,data);
   cnt++;break;
   case 2:
   data[0] = 0x03;
   data[1] = int(KP_P*100)/256;
   data[2] = int(KP_P*100)%256;
   AssemblyAndSend(0x04,data);
   cnt++;break;
   case 3:
   data[0] = 0x04;
   data[1] = int(KP_I*100)/256;
   data[2] = int(KP_I*100)%256;
   AssemblyAndSend(0x04,data);
   cnt++;break;
   case 4:
   data[0] = 0x05;
   data[1] = int(K_Base*100)/256;
   data[2] = int(K_Base*100)%256;
   AssemblyAndSend(0x04,data);
   cnt = 0;break;
  default:break;
  }
}
void AssemblyAndSend(char type ,unsigned char *data)
```

```
{
  unsigned char sendbuff[6];
  sendbuff[0] = 0xAA;
  sendbuff[1] = type;
  sendbuff[2] = data[0];
  sendbuff[3] = data[1];
  sendbuff[4] = data[2];
  sendbuff[5] = data[0]^data[1]^data[2];
  for(int i = 0; i < 6; i++)
  {
    Serial.write(sendbuff[i]);
  }
}
void Init()
{
  pinMode(BUZZER, OUTPUT);
  pinMode(SPD_PUL_L, INPUT);
  pinMode(SPD_PUL_R, INPUT);
  pinMode(PWM_L, OUTPUT);
  pinMode(PWM_R, OUTPUT);
  pinMode(DIR_L1, OUTPUT);
  pinMode(DIR_L2, OUTPUT);
  pinMode(DIR_R1, OUTPUT);
  pinMode(DIR_R2, OUTPUT);
  pinMode(AUX_IN_PIN,INPUT);
  pinMode(UP_DOWN_IN_PIN, INPUT);
  pinMode(LEFT_RIGHT_IN_PIN, INPUT);
  Speed_L = 0;
  Speed_R = 0;
  Position_AVG = 0;
  Position_AVG_Filter = 0;
  Position_Add = 0;
```

```
  pwm = 0;pwm_l = 0;pwm_r = 0;
  Speed_Diff = 0;Speed_Diff_ALL = 0;
  KA_P = 35.0;
  KA_D = 3.5;
  KP_P = 30.0;
  KP_I = 0.34;
  K_Base = 6.7;
 ReadingData.KA_P = KA_P;
 ReadingData.KA_D = KA_D;
 ReadingData.KP_P = KP_P;
 ReadingData.KP_I =KP_I;
 ReadingData.K_Base = K_Base;
  Speed_Need = 0;
  Turn_Need = 0;
 ReadFromEEprom(&ReadingData);
  KA_P = ReadingData.KA_P;
  KA_D = ReadingData.KA_D;
  KP_P = ReadingData.KP_P;
  KP_I = ReadingData.KP_I;
  K_Base = ReadingData.K_Base;
  Serial.print("PID Data is");Serial.print("\t");
  Serial.print(KA_P);Serial.print("\t");
  Serial.print(KA_D);Serial.print("\t");
  Serial.print(KP_P);Serial.print("\t");
  Serial.print(KP_I);Serial.print("\t");
  Serial.print(K_Base);Serial.println("\t");
}
void Encoder_L()   //car up is positive car down  is negative
{
  if (digitalRead(SPD_PUL_L))
    Speed_L += 1;
  else
```

```
   Speed_L -= 1;
}
void Encoder_R()    //car up is positive car down  is negative
{
 if (!digitalRead(SPD_PUL_R))
   Speed_R += 1;
 else
   Speed_R -= 1;
}



void MySerialEvent()
{
          uchar c = '0';
       uchar tmp = 0;
       if(Serial.available()) {
           c = (uchar)Serial.read();
         for(int i = 5; i > 0; i--)
         {
           SerialPacket.m_Buffer[i] = SerialPacket.m_Buffer[i-1];
         }
          SerialPacket.m_Buffer[0] = c;
          if(SerialPacket.m_Buffer[5] == 0xAA)
         {

Tmp=SerialPacket.m_Buffer[1]^SerialPacket.m_Buffer[2]^SerialPacket.m_Buffer[3];
           if(tmp == SerialPacket.m_Buffer[0])
           {
             SerialPacket.m_PackageOK = true;
           }
         }
       }
```

```
}

int UpdateAttitude()
{
 if((micros() - timer) >= 10000)
 {    //10ms
  /* Update all the values */
  while (i2cRead(0x3B, i2cData, 14));
  accX = ((i2cData[0] << 8) | i2cData[1]);
  accY = ((i2cData[2] << 8) | i2cData[3]);
  accZ = ((i2cData[4] << 8) | i2cData[5]);
  tempRaw = (i2cData[6] << 8) | i2cData[7];
  gyroX = (i2cData[8] << 8) | i2cData[9];
  gyroY = (i2cData[10] << 8) | i2cData[11];
  gyroZ = (i2cData[12] << 8) | i2cData[13];
  double dt = (double)(micros() - timer) / 1000000;
  timer = micros();
#ifdef RESTRICT_PITCH // Eq. 25 and 26
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ * accZ)) * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ)) * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif
  double gyroXrate = gyroX / 131.0; // Convert to deg/s
  double gyroYrate = gyroY / 131.0; // Convert to deg/s
#ifdef RESTRICT_PITCH
  if ((roll < -90 && kalAngleX > 90) || (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    compAngleX = roll;
    kalAngleX = roll;
    gyroXangle = roll;
```

```
  } else
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
  if (abs(kalAngleX) > 90)
    gyroYrate = -gyroYrate;
  kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
#else
  if ((pitch < -90 && kalAngleY > 90) || (pitch > 90 && kalAngleY < -90)) {
    kalmanY.setAngle(pitch);
    compAngleY = pitch;
    kalAngleY = pitch;
    gyroYangle = pitch;
  } else
    kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
  if (abs(kalAngleY) > 90)
    gyroXrate = -gyroXrate;
  kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
#endif
   if (gyroXangle < -180 || gyroXangle > 180)
    gyroXangle = kalAngleX;
  if (gyroYangle < -180 || gyroYangle > 180)
    gyroYangle = kalAngleY;
  /* Print Data */
#if 0 // Set to 1 to activate
  Serial.print(accX); Serial.print("\t");
  Serial.print(accY); Serial.print("\t");
  Serial.print(accZ); Serial.print("\t");
  Serial.print(gyroX); Serial.print("\t");
  Serial.print(gyroY); Serial.print("\t");
  Serial.print(gyroZ); Serial.print("\t");
  Serial.print("\t");
#endif
#if 0
```

```
  Serial.print(roll); Serial.print("\t");
 // Serial.print(gyroXangle); Serial.print("\t");
 // Serial.print(compAngleX); Serial.print("\t");
  Serial.print(kalAngleX); Serial.print("\t");
  Serial.print("\t");
  Serial.print(pitch); Serial.print("\t");
 // Serial.print(gyroYangle); Serial.print("\t");
 // Serial.print(compAngleY); Serial.print("\t");
  Serial.print(kalAngleY); Serial.println("\t");
#endif
#if 0 // Set to 1 to print the temperature
  Serial.print("\t");
  double temperature = (double)tempRaw / 340.0 + 36.53;
  Serial.print(temperature); Serial.print("\t");
#endif
  Angle_Car = kalAngleX;
  Gyro_Car = gyroXrate;
   return 1;
 }
  return 0;
}
void LEDBlink()
{
  if((millis() - LEDTimer) > 500)
  {
    LEDTimer = millis();
    blinkState = !blinkState;
  }
}
void calcUpDown()
{
  if(digitalRead(UP_DOWN_IN_PIN) == HIGH)
```

```
  {
    UpDownStart = micros();
  }
  else
  {
    UpDownEnd = (uint16_t)(micros() - UpDownStart);
    bUpdateFlagsRC |= UP_DOWN_FLAG;
  }
}
void calcLeftRight()
{
  if(digitalRead(LEFT_RIGHT_IN_PIN) == HIGH)
  {
    LeftRightStart = micros();
  }
  else
  {
    LeftRightEnd = (uint16_t)(micros() - LeftRightStart);
    bUpdateFlagsRC |= LEFT_RIGHT_FLAG;
  }
}
#define  RC_ERROR 100
#define RC_CAR_SPEED  160
int RCTurnSpeed = 80;
void ProcessRC()
{
  if(bUpdateFlagsRC)
  {
     noInterrupts();
    if(bUpdateFlagsRC & UP_DOWN_FLAG)
    {
      UpDownIn = UpDownEnd;
```

```
    if( abs(UpDownIn - 1500) > RC_ERROR )
    {
     if(UpDownIn > 1500)
      Speed_Need = RC_CAR_SPEED;
      else
      Speed_Need = -RC_CAR_SPEED;
    }
    else
    {
     Speed_Need = 0;
    }
   }

   if(bUpdateFlagsRC & LEFT_RIGHT_FLAG)
   {
    LeftRightIn = LeftRightEnd;
    if( abs(LeftRightIn - 1500) > RC_ERROR)
    {
     if(LeftRightIn > 1500)
     Turn_Need = -RCTurnSpeed;
     else
     Turn_Need = RCTurnSpeed;
    }
    else
    {
     Turn_Need = 0;
    }
   }
   bUpdateFlagsRC = 0;
   interrupts();
  }
}
```