ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẶNG VĂN MƯỜI

THIẾT KẾ, CHẾ TẠO ROBOT 04 BẬC TỰ DO MÔ PHỎNG CHUYỂN ĐỘNG TRÊN TÀU THỦY

LUẬN VĂN THẠC SĨ

CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ, TRUYỀN THÔNG

HÀ NỘI - 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

ĐẶNG VĂN MƯỜI

THIẾT KẾ, CHẾ TẠO ROBOT 04 BẬC TỰ DO MÔ PHỎNG CHUYỂN ĐỘNG TRÊN TÀU THỦY

Ngành: Công nghệ kỹ thuật điện tử, truyền thông

Chuyên ngành: Kỹ thuật điện tử

Mã số: 60520203

LUẬN VĂN THẠC SĨ

CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ, TRUYỀN THÔNG

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN THĂNG LONG

HÀ NỘI - 2017

LỜI CẢM ƠN

Luận văn này được hoàn thành với sự hỗ trợ của đề tài độc lập cấp nhà nước mã số ĐTĐL.CN-02/2017 và đề tài cấp Đại học Quốc gia Hà Nội mã số QG.16.28.

Để hoàn thành luận văn này, ngoài sự nỗ lực của bản thân, tôi còn nhận được sự giúp đỡ nhiệt tình từ phía nhà trường, cán bộ hướng dẫn, gia đình, công ty và bạn bè. Tôi xin được gửi lời cảm ơn chân thành và sâu sắc đến:

- TS. Nguyễn Thăng Long, Bộ môn vi cơ điện tử và vi hệ thống, Khoa Điện Tử Viễn Thông, Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã tận tình hướng dẫn tôi trong suốt quá trình làm luận văn.

- Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội đã tạo điều kiện cho tôi học tập, nghiên cứu tạo tiền đề vững chắc cho tôi hoàn thành khóa luận.

Cuối cùng tôi xin gửi lời cảm ơn đến gia đình và tất cả bạn bè đã luôn ở bên, ủng hộ tôi để hoàn thành khóa luận.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày 29 tháng 12 năm 2017

Đặng Văn Mười

LỜI CAM ĐOAN

Tôi xin cam đoan, luận văn là công trình nghiên cứu của tôi, có hỗ trợ từ cán bộ hướng dẫn là TS. Nguyễn Thăng Long cùng các thành viên trong nhóm nghiên cứu. Nội dung nghiên cứu trong luận văn không sao chép bất kỳ công trình nghiên cứu của người khác. Ngoài ra, luận văn còn sử dụng thông tin, hình vẽ, số liệu được thu thập từ nhiều nguồn khác nhau được chỉ rõ ở phần tài liệu tham khảo.

Nếu có bất kỳ sự gian lận nào, tôi xin chịu hoàn toàn trách nhiệm trước hội đồng nhà trường cũng như kết quả của luận văn này.

Hà Nội, ngày 29 tháng 12 năm 2017

Học viên

Đặng Văn Mười

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

MỞ ĐẦU

Tính cấp thiết của đề tài:

Robot 4 bậc tự do là một trong những mô hình robot song song được thiết kế xây dựng để đáp ứng yêu cầu mô hình hóa, mô phỏng lại các chuyển động thực tế trên tàu thuyền, máy bay, các phương tiện giao thông ... Với hiệu quả rất lớn trong thực tiễn để phục vụ mục đích nghiên cứu, diễn tập, giải trí thì hệ thống robot 4 bậc tự do ngày càng được ứng dụng nhiều hơn nữa. Cụ thể trong đề tài này, hệ thống robot 4 bậc tự do được sử dụng để mô phỏng lại chuyển động trên tàu thủy với mục đích kiểm tra, hoàn thiện các tính năng hoạt động; chạy thử kiểm định các thiết bị trong điều kiện chuyển động với các thông số khác nhau tại phòng thí nghiệm. Việc chế tạo hệ thống robot này là rất cần thiết để phục vụ các nghiên cứu chạy thử nghiệm hệ thống trong phòng thí nghiệm trước khi cho vận hành trong điều kiện thực tế để kiểm soát và tối ưu được các thông số của thiết bị.

Chế tạo robot 4 bậc tự do là một trong các nhiệm vụ được đặt ra của đề tài "*Nghiên cứu phát triển sản phẩm thương mại hóa trạm thu di động tín hiệu truyền hình vệ tinh ứng dụng trên tàu biển"* (QG.16.89) do nhóm nghiên cứu tại trường Đại học Công nghệ, Đại học Quốc gia Hà Nội thực hiện dưới sự chủ trì của GS.TS. Nguyễn Hữu Đức. Đề tài này được phát triển từ đề tài nghiên cứu trong chương trình Khoa học và Công nghệ Vũ trụ và cho đến nay tiếp tục được đầu tư để phát triển thành sản phẩm thương mại. Để hoàn thiện sản phẩm thương mại thì việc đo đạc kiểm định hoạt động của hệ thống trong các điều kiện rung lắc, chuyển động với vận tốc, gia tốc khác nhau theo yêu cầu đặt ra là rất khó thực hiện trong điều kiện thực tế trên tàu biển do phụ thuộc vào thời tiết. Thêm vào đó, việc chạy thử nghiệm trong điều kiện dã ngoại đòi hỏi chi phí cao và xác suất rủi ro thất bại là cao nếu không được vận hành thử nghiệm tốt trong phòng thí nghiệm với các điều kiện tương tự.

Chính vì lý do này, luận văn đặt đề tài Thiết kế, chế tạo robot 4 bậc tự do mô phỏng chuyển động trên tàu thủy với mục tiêu tạo các chuyển động như chuyển động thực của tàu thủy trong không gian 3 chiều với các thông số chuyển động khác nhau phục vụ chạy thử thiết bị Trạm thu di động thông tin vệ tinh. Sản phẩm thiết kế chế tạo được trong luận văn này có thể tiếp tục được nghiên cứu phát triển hướng tới các ứng dụng trong nhiều lĩnh vực khác cả trong nghiên cứu và thực tiễn.

Ý nghĩa khoa học và thực tiễn:

Việc chạy thử nghiệm hệ thống trạm thu di động thông tin vệ tin trong phòng thí nghiệm được mô phỏng như chạy thật của tàu biển với các chế độ rung, lắc, nghiêng,

quay trái phải, chòng chành,… khác nhau ứng với các điều kiện thời tiết khác nhau chỉ có thể thực hiện được khi sử dụng một hệ thống robot được lập trình tự động điều khiển với các hệ thống cảm biến kèm theo. Việc chế tạo thành công hệ robot này giúp tiết kiêm chi phí, thời gian, cho nội dung chạy thử hệ thống thiết bị trong điều kiện phòng thí nghiệm giống như điều kiện dã ngoại thực tế, kiểm tra được các thông số đáp ứng, làm việc, độ bền, độ linh hoạt của hệ thống, có thể thử nghiệm trong thời gian dài 24/7 để đánh giá mà trong điều kiện dã ngoại thực tế rất khó thực hiện.

Với kiến thức được trang bị cũng như tìm tòi tham khảo được, việc thiết kế, chế tạo hoàn thiện hệ thống giúp học viên có cơ hội được vận dụng vào thực tế kiên thức đã được trang bị và tích lũy, đánh giá khả năng làm việc của bản thân cũng như của nhóm nghiên cứu. So sánh sản phẩm tự thiết kế với các sản phẩm nhập ngoại, qua đó có hướng phát triển cho sản phẩm trong tương lai đa dạng hóa thị trường ứng dụng của hệ thống, nhằm hạ giá thành và thay thế các sản phầm nhập ngoại đắt tiền, đưa tự động hóa đến gần với người sử dụng hơn.

Đối tượng nghiên cứu:

Đối tượng nghiên cứu của đề tài là các mô hình robot song song, cách thiết kế chế tạo ra hệ thống robot 4 bậc tự do. Cùng với đó là các thuật toán điều khiển sử dụng PID, các phương pháp lọc số và các mô-đun, bộ KIT điều khiển mới.

Phương pháp nghiên cứu:

Tham khảo các tài liệu giới thiệu về robot, các cơ cấu robot và đặc biệt là các mô hình robot song song. Dựa trên đó để xây dựng được hệ thống robot 4 bậc tự do đáp ứng được yêu cầu đặt ra của đề tài.

Nghiên cứu và áp dụng các giải thuật điều khiển, các phương pháp xử lý tín hiệu, các bộ lọc số. Tìm kiếm các ví dụ tham khảo qua đó tối ưu và áp dụng vào bài toán thực tế.

Nội dung nghiên cứu:

Các nội dung nghiên cứu được thực hiện trong luận văn bao gồm:

- Nghiên cứu, xây dựng mô hình lý thuyết cơ cấu chuyển động cho hệ thống robot 4 bậc tự do.
- Thiết kế cơ khí, lựa chọn linh kiện, vật tư, cơ cấu truyền động chấp hành, motor, cảm biến….
- Lắp đặt, vận hành, chạy thử nghiệm hệ thống không tải và có tải hệ thống trạm thu di động.
- Đo kiểm và đánh giá hoạt động của hệ thống robot.

Chương 1

TỔNG QUAN

1.1. TỔNG QUAN VỀ XÂY DỰNG MÔ HÌNH

1.1.1. *Yêu cầu xây dựng mô hình*

Hình 1.1. Mô phỏng chuyển động trên tàu thủy bằng phương pháp thủ công

Với mục đích thay thế việc mô phỏng các chuyển động, dao động trên tàu thủy khi chịu tác động của sóng bằng phương pháp thủ công như Hình 1.1 bằng một phương pháp nhanh chóng, đơn giản, thuận tiện, đáp ứng được các thông số đặt ra và tiết kiệm chi phí hơn. Do đó yêu cầu thiết kế của đề tài đó là xây dựng được một mô hình robot có khả năng mô phỏng lại được các chuyển động trên tàu thủy. Cụ thể hơn, kết quả của đề tài sẽ mô phỏng lại sự thay đổi về các góc phương hướng của hệ thống antena thu phát sóng đặt trên thuyền.

Vị trí và phương hướng trong không gian của một vật thể, trong trường hợp này là hệ thống antena thu phát sóng, được xác định như một vị trí và phương hướng của một khung tham chiếu chính đặt trong một khung tham chiếu khác cố định với vật thể. Để xác định được phương hướng của vật thể, ta cần ít nhất 03 giá trị độc lập mặc dù một vật thể có thể di chuyển tự do trong không gian (vật thể có 06 bậc tự do) sẽ có 06 giá trị để xác định vị trí và phương hướng.

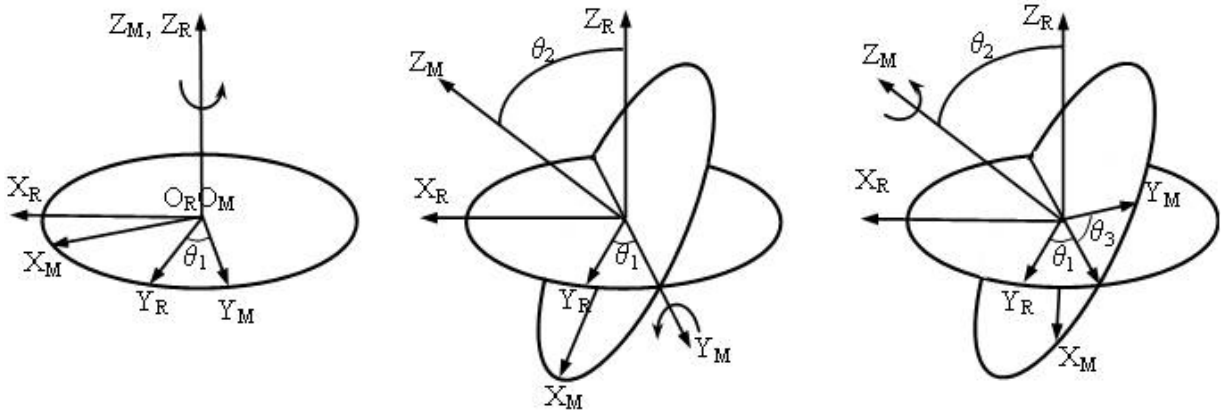
Do hệ thống antena thu phát sóng được đặt trên tàu thủy, nên tâm quay của hệ tọa độ tham chiếu (là antena) sẽ trùng với hệ tọa độ gốc (là tàu thủy). Nếu như hệ tọa độ gốc

đặt cố định tại đất liền (gắn với trái đất) thì ta cần thêm 03 giá trị thể hiện sự tịnh tiến của hệ tọa độ tham chiếu theo hệ tọa độ gốc theo 03 phương vuông góc với nhau.

Kết luận: mô hình robot của đề tài sẽ mô phỏng 03 giá trị góc phương hướng.

1.1.2. *Biểu diễn phương hướng của vật thể*

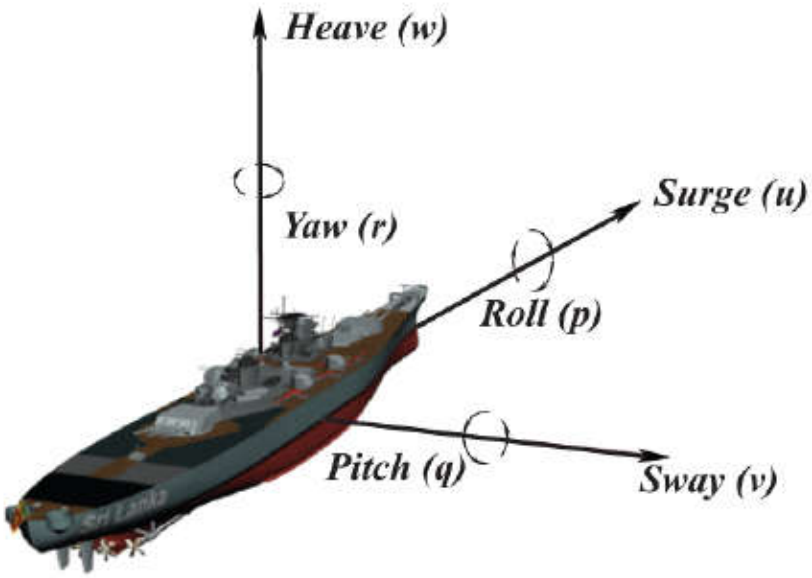
Để biểu diễn phương hướng có rất nhiều phương pháp, trong đó Leonhard Euler là người tiên phong trong việc này, ông đã tưởng tượng ra 03 khung tham chiếu có thể quay lần lượt vòng quanh nhau và nhận ra rằng bằng cách sử dụng một khung tham chiếu cố định và biểu diễn ba vòng quay, ông có thể dùng biểu diễn bất kỳ khung tham chiếu nào khác trong không gian. Giá trị thu được của biểu diễn ba vòng quay được gọi là các góc Euler [5].



Hình 1.2. Phép biểu diễn Euler [6]

Việc biểu diễn các góc Euler này cũng có rất nhiều lựa chọn (12 lựa chọn) trong đó có 06 kiểu gọi là vòng quay nội tại (hay còn được gọi là góc Euler cổ điển) và 06 kiểu còn lại được gọi là vòng quay bên ngoài (hay còn được gọi là góc Tait-Bryan). Khác nhau cơ bản của vòng quay bên ngoài và vòng quay nội tại là các phép quay nguyên tố sẽ xảy ra ở các trục của hệ tọa độ cố định (gắn liền với trái đất); còn vòng quay nội tại thì các phép quay nguyên tố sẽ thay đổi theo trục của hệ tọa độ mới sinh ra sau mỗi phép quay [7].

Chính vì việc phép quay nguyên tố xảy ra ở trục tọa độ cố định (bên ngoài) nên vòng quay bên ngoài hay các góc Tait-Bryan được ứng dụng trong việc xác định phương hướng của các vật thể, phương tiện đi lại và hàng không vũ trụ như: tàu thuyền, máy bay, tên lửa… Các góc Tait-Bryan còn được biết đến dưới dạng các định nghĩa thông dụng hơn như: góc nghiêng (roll angle), góc ngẩng (pitch angle) và góc cuộn (yaw angle).



Hình 1.3. Biểu diễn góc nghiêng, góc ngẩng và góc cuộn

1.2. TỔNG QUAN VỀ ROBOT

1.2.1. *Giới thiệu và phân loại robot*

Khái niệm robot đã được các nhà khoa học đưa ra nhiều cách định nghĩa khác nhau nhưng tổng kết lại thì điểm thống nhất của các khái niệm này là đặc điểm "điều khiển theo chương trình". Các robot xuất hiện ban đầu dưới dạng đơn giản như các tay máy công nghiệp, và sau đó phát triển bùng nổ ra nhiều dạng khác nhau để phục vụ nhiều mục đích của người sử dụng. Robot được phát triển không chỉ về mặt đa dạng về cấu trúc mà còn được phát triển cả về mặt điều khiển, hiện nay các nhà khoa học sáng tạo ra được cả những loại robot có khả năng "suy nghĩ" và có cảm xúc như con người [3].

Để phân loại robot ta có thể chia theo các yếu tố như: theo dạng hình học của không gian hoạt động, theo thế hệ robot, theo bộ điều khiển, theo nguồn dẫn động hoặc theo kết cấu...

1.2.1.a. Phân loại theo dạng hình học của không gian hoạt động

Người ta phân loại robot theo sự phối hợp giữa ba trục chuyển động cơ bản rồi có thể bổ sung thêm các bậc chuyển động nhằm tăng thêm độ linh hoạt. Vùng giới hạn tầm hoạt động của robot được gọi là không gian làm việc.

Robot tọa độ vuông góc: robot loại này có ba bậc chuyển động cơ bản gồm ba chuyển động tịnh tiến dọc theo ba trục vuông góc.

Robot tọa độ trụ: ba bậc chuyển động cơ bản gồm hai trục chuyển động tịnh tiến và một trục quay.

Robot tọa độ cầu: ba bậc chuyển động cơ bản gồm một trục tịnh tiến và hai trục quay.

Robot khớp bản lề: ba bậc chuyển động cơ bản bao gồm ba trục quay.

1.2.1.b. Phân loại theo thế hệ

Theo quá trình phát triển của robot, ta có thể chia ra theo các mức độ sau đây:

Robot thế hệ thứ nhất: bao gồm các dạng robot hoạt động lặp lại theo một chu trình không thay đổi, theo chương trình định trước.

Robot thế hệ thứ hai: bao gồm các robot được trang bị các cảm biến cho phép cung cấp tín hiệu phản hồi trở lại hệ thống điều khiển về trạng thái, vị trí không gian của robot cũng như thông tin về môi trường bên ngoài... giúp cho robot có thể lựa chọn những thuật toán thích hợp để điều khiển robot thực hiện những thao tác xử lý phù hợp. Robot loại này còn được gọi là robot điều khiển thích nghi cấp thấp.

Robot thế hệ thứ ba: bao gồm các robot được trang bị những thuật toán xử lý các phản xạ logic thích nghi theo những thông tin và tác động của môi trường lên chúng, nhờ đó robot tự biết phải làm gì để hoàn thành công việc đã được đặt ra. Đây là dạng phát triển cao nhất của robot tự cảm nhận. Robot loại này bao gồm các robot được trang bị hê thống thu nhận hình ảnh trong điều khiển.

Robot thế hệ thứ tư: bao gồm các robot sử dụng các thuật toán và cơ chế điều khiển thích nghi được trang bị bước đầu khả năng lựa chọn các đáp ứng tuân theo một mô hình tính toán xác định trước nhằm tạo ra những ứng xử phù hợp với điều kiện của môi trường thao tác.

Robot thế hệ thứ năm: bao gồm những robot được trang bị các kỹ thuật của trí tuệ nhân tạo như nhận dạng tiếng nói, hình ảnh, xác định khoảng cách, cảm nhận đối tượng qua tiếp xúc... để đưa ra quyết định và giải quyết các vấn đề hoặc nhiệm vụ đặt ra cho nó.

1.2.1.c. Phân loại theo nguồn dẫn động

Phụ thuộc vào nguồn dẫn động có thể phân loại robot theo một số dạng như sau:

Robot dùng nguồn cấp điện: nguồn cấp điện cho robot có thể là DC, hệ thống có thể dùng nguồn AC sau đó chuyển đổi sang DC để điều khiển động cơ DC. Các động cơ thường dùng là động cơ bước, động cơ DC servo, động cơ AC servo...

Robot dùng nguồn khí nén: hệ thống robot này được trang bị máy nén, bình chứa khí và động cơ máy nén. Robot loại này sử dụng các xy-lanh khí nén để thực hiện chuyển động thẳng và chuyển động quay.

Robot dùng nguồn thủy lực: hệ thống robot này sử dụng các bơm để tạo áp lực dầu, các cơ cấu chấp hành là các xy-lanh thủy lực và thường sử dụng cho các ứng dụng có trọng tải lớn.

Hiện nay các hệ thống robot sử dụng hỗn hợp giữa các nguồn dẫn động để tạo ra cơ cấu hoạt động linh hoạt và đa dạng.

1.2.1.d. Phân loại theo kết cấu động học

Theo kết cấu động học của robot, ta có thể phân loại theo 02 loại: robot nối tiếp và robot song song.

Robot nối tiếp: trong kết cấu động học nối tiếp thông thường thì tất cả các trục chuyển động được bố trí nối tiếp với nhau. Mỗi khâu động được liên kết hoặc nối động với các khâu khác nhờ các khớp động. Mỗi khớp thường chỉ cho phép thực hiện một chuyển động tương đối, mỗi trục tiếp theo sẽ làm cho kết cấu có thêm một bậc tự do. Do tính nối tiếp nên khâu trước phải chịu tải trọng của khâu sau dẫn đến công suất sử dụng ngày càng tăng nếu số lượng khâu tăng lên.

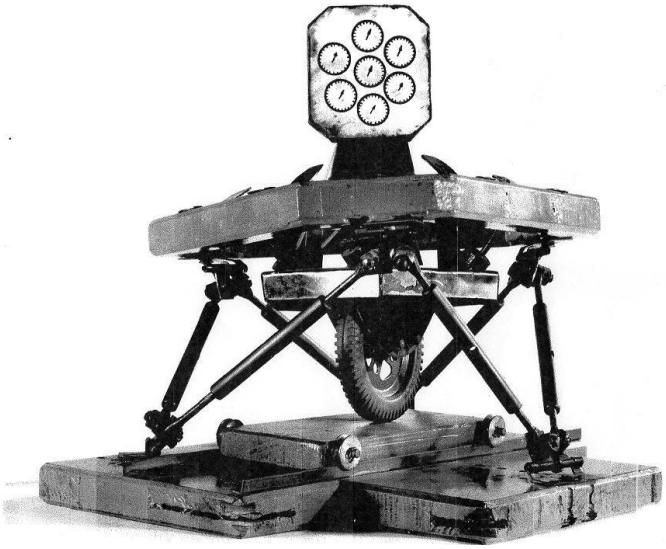
Robot song song: robot loại này có thể xem như một chuỗi động học kín, ở đó mỗi khâu luôn luôn được liên kết với ít nhất hai khâu khác. Cấu trúc động học song song không hẳn đã tồn tại các cấu kiện song song theo ý nghĩa hình học. Mà trong cấu trúc này tất cả các trục khi chuyển động sẽ tác động trực tiếp hoặc gián tiếp lên bàn công tác (cơ cấu chuyển động cần thiết cuối cùng). Để thực hiện một chuyển động thì tất cả các cơ cấu đề phải hoạt động, như vậy sẽ xuất hiện chuỗi động học gọi là động học kín. Số lượng của chuỗi khớp nối sẽ bằng đúng số bậc tự do của cấu trúc.

1.2.2. ***Robot song song và ứng dụng***

Những công trình lý thuyết về cơ cấu động học song song đã có từ hàng trăm năm trước đây nhưng được ứng dụng thực tế thì chỉ được biết đến trong thế kỷ XX.

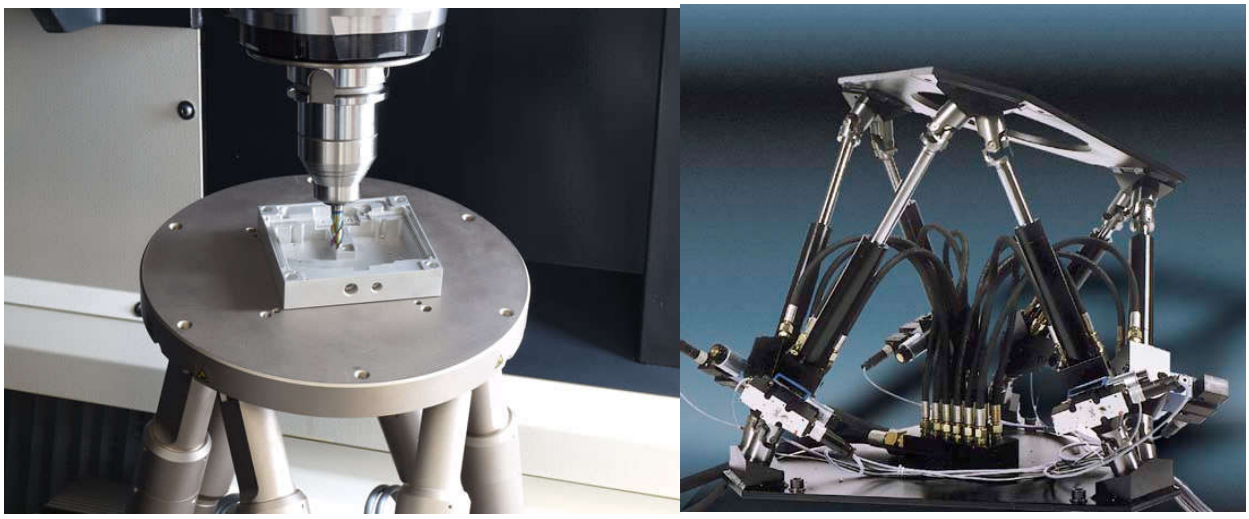
Kết cấu động học song song không gian đầu tiên cho ứng dụng công nghiệp là robot sơn 5 bậc tự do, được thiết kế bởi L.W.Willard nhưng đáng tiếc là thiết kế này không được áp dụng vào thực tế.

Sau đó, một cơ cấu động học song song ngày càng được phát triển, trở nên nổi tiếng và được chế tạo hàng nghìn phiên bản: đó là thiết bị kiểm tra lốp dựa trên nguyên lý hexapod của Eric Gough.



Hình 1.4. Robot song song hexapod của Eric Gough

Ngày nay, các ứng dụng của robot song song đã được phát triển rất đa dạng như trong lĩnh vực mô phỏng chuyển động, các thiết bị vận hành, áp dụng cho máy công cụ với đủ các loại kết cấu, nguồn dẫn động...



Hình 1.5. Các ứng dụng của robot song song

Chương 2

THIẾT KẾ CƠ KHÍ

2.1. TÍNH TOÁN, LỰA CHỌN MÔ HÌNH

2.1.1. *Đánh giá các mô hình robot có sẵn trên thị trường*

Hiện nay, trên thì trường cũng có sẵn rất nhiều mô hình robot song song (3 hoặc 4 bậc tự do) để ứng dụng trong việc mô phỏng các chuyển động phục vụ mục đích thí nghiệm và giải trí. Riêng ở Việt Nam mới chỉ có đề tài nghiên cứu về robot song song 6 bậc tự do (hexapod) ứng dụng trong gia công cơ khí do Viện Cơ học nghiên cứu và đưa ra sản phẩm thực tế; sản phẩm này chưa áp dụng rộng rãi trên thị trường. Các nghiên cứu khác về robot song song mới chỉ dừng lại ở việc nghiên cứu lý thuyết và mô phỏng, chưa có sản phẩm thực tế.

Hình 2.1. Các mô hình có sẵn trên thị trường

Các mô hình	Số bậc tự do	Chi phí
RacingCube	4	4000 Euro
Teslame	3	6000 USD
MotionSystem	3	8500 Euro
Viện Cơ học	6	Không rõ chi phí
Sản phẩm đề tài này	4	25.000.000 VND

Bảng 2.1. So sánh chi phí các mô hình robot song song

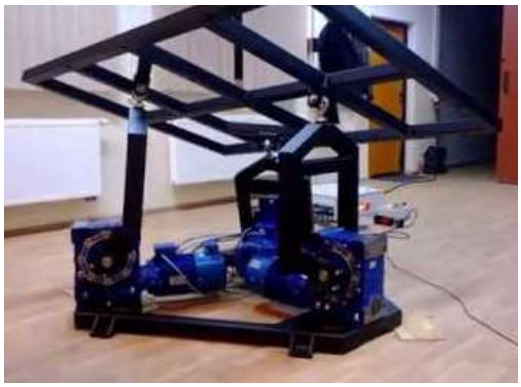
Các mô hình có sẵn trên thị trường thì chúng đều có nhưng ưu nhược điểm, nhưng nhìn chung về mặt chi phí quá cao so với việc tự nghiên cứu, chế tạo trong nước. Các mô hình này thường được thiết kế với tải trọng lớn (>100kg) nên sử dụng các động cơ công suất lớn, động cơ hộp số với tỉ lệ truyền cao. Đa số các mô hình này chỉ mô phỏng lại được chuyển động ngẩng-cúi và nghiêng trái-nghiêng phải mà không mô phỏng lại được chuyển động quay trái-quay phải (trừ sản phẩm racingcube). Mà chuyển động quay trái –

quay phải là một trong những yếu tố quan trọng cần phải có của đề tài này. Đây cũng chính là điểm mới trong nghiên cứu ứng dụng và phát triển hơn những mô hình sẵn có.

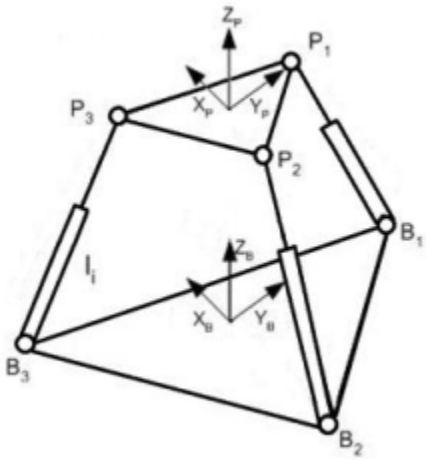
2.1.2. *Lựa chọn mô hình robot song song*

Như đã giới thiệu trong phần trên, các robot song song được ứng dụng rất nhiều trong việc mô phỏng chuyển động (tàu thuyền, máy bay...). Một hệ robot song song 06 bậc tự do (hexapod) có thể cung cấp đầy đủ khả năng mô phỏng phương hướng và vị trí của vật thể trong không gian (đương nhiên có một số giới hạn không gian làm việc do giới hạn về kết cấu chuyển động), nhưng trong phạm vi đề tài này chúng ta sẽ không sử dụng đến mô hình robot đó. Mô hình robot hexapod có 06 chuỗi khớp nối dẫn đến việc thiết kế cơ khí cũng như tính toán động học và điều khiển cho robot quá phức tạp. Đây hoàn toàn là một ý tưởng, hướng đi tốt sau khi hoàn thành được mô hình robot 04 bậc tự do.

Với 03 giá trị góc phương hướng cần thay đổi, mô hình robot xây dậy dựng cho việc mô phỏng phải đạt được ít nhất là 03 bậc tự do. 03 bậc tự do đó là: quay quanh trục X, quay quanh trục Y và quay quanh trục Z.



Hình 2.2. Hình ảnh thực tế của robot 3 bậc tự do



Hình 2.3. Mô hình toán học của robot 3 bậc tự do

Nội dung so sánh	3 bậc tự do	4 bậc tự do	6 bậc tự do
Khả năng tịnh tiến theo trục X	Không	Không	Có
Khả năng tịnh tiến theo trục Y	Không	Không	Có
Khả năng tịnh theo trục Z	Có	Có	Có (nhỏ hơn)
Khả năng quay theo trục X	Có	Có	Có (nhỏ hơn)
Khả năng quay theo trục Y	Có	Có	Có (nhỏ hơn)
Khả năng quay theo trục Z	Không	Có	Có (nhỏ hơn)
Số biến cần tính toán động học	3	4	6
Tốc độ xử lý tính toán	Nhanh	Chậm hơn	Rất chậm
Chi phí thực hiện dự án	Trung bình	Lớn	Rất lớn

Bảng 2.2. So sánh thông số các mô hình robot song song

Bảng trên cho thấy khả năng đáp ứng, cũng như ưu nhược điểm của các mô hình robot trực quan. Dựa vào các thông số trên để đưa ra ý tưởng về việc lựa chọn mô hình phù hợp cho đề tài. Hệ thống robot 3 bậc tự do mới chỉ đáp ứng được việc mô phỏng chuyển động quay quanh trục X, trục Y (mô phỏng góc nghiêng, góc ngẩng) và chuyển động lên xuống. Để đáp ứng được chuyển động quay quanh trục Z (mô phỏng góc cuộn) thì cần phải tích hợp thêm một cơ cấu quay. Khi tích hợp thêm cơ cấu quay này thì hệ thống sẽ trở thành một hệ thống robot 4 bậc tự do. Trong khi đó, hệ thống robot 6 bậc tự do lại không đáp ứng được yêu cầu quay được góc 360^o hoặc lớn hơn. Đây chính là lý do đề tài lựa chọn mô hình robot song song 4 bậc tự do và đặt tên là ***"Thiết kế, chế tạo robot 04 bậc tự do mô phỏng chuyển động trên tàu thủy".***

2.1.3. *Mô hình robot song song 4 bậc tự do*

Về cơ bản, hệ thống robot 4 bậc tự do được xây dựng dựa trên mô hình robot song song 3 bậc tự do. Nếu sử dụng mô hình robot song song 3 bậc tự do thì hệ thống chỉ có thể mô phỏng lại các chuyển động theo chiều lên xuống thẳng đứng, nghiêng trái – nghiêng phải và ngẩng lên – cúi xuống mà không thể mô phỏng lại được chuyển động quay trái – quay phải. Chuyển động quay trái – quay phải là một trong những chuyển động cơ bản nhất của tàu thủy khi thay đổi hướng đi hay chịu tác động của sóng biển. Với yêu cầu của đề tài là mô phỏng được đầy đủ các phương hướng của tàu thủy, sau quá trình nghiên cứu các mô hình robot, nhận thấy việc tích hợp thêm một cơ cấu quay tròn 360^o là cần thiết để đảm bảo yêu cầu đặt ra.

Với một hệ thống robot, để điều khiển một cách tốt nhất thì cần phải xây dựng được bài toán động học thuận và động học ngược. Khi xây dựng hoàn thiện bài toán, hệ thống robot có thể điều khiển trực tiếp bằng các giá trị góc (góc nghiêng, góc ngẩng, góc cuộn). Các phép tính toán sẽ chuyển đổi giá trị các góc này sang độ dài của các chân robot cần phải dịch chuyển để đạt được giá trị đặt trước. Tuy nhiên do thời gian đề tài có hạn nên luận văn sử dụng phương pháp điều khiển trực tiếp, tức là nhập thẳng các giá trị dịch chuyển cần phải đạt được dưới dạng các bước dịch chuyển. Về cơ bản, cách làm này không ảnh hưởng đến việc tạo ra các mô phỏng chuyển động trên tàu biển hay một số phương tiện khác. Những giá trị đầu vào dưới dạng góc cũng đều phải chuyển thành dạng dước dịch chuyển trước khi đưa vào bộ tính toán điều khiển của hệ thống. Do vậy, việc đánh giá tính chính xác, độ ổn định cũng như các yếu tố khác của hệ thống robot hoàn toàn có thể thực hiện được bình thường. Nhược điểm của cách làm này là chưa áp dụng được giá trị góc trực tiếp và đây cũng là phương hướng phát triển sắp tới của đề tài.

Hình 2.4. Robot 4 bậc tự do và trạm thu phát sóng di động

Hình ảnh trên là hệ thống robot 4 bậc tự do sau khi đã hoàn thành và đang tiến hành thử nghiệm cùng với hệ thống trạm thu di động tín hiệu truyền hình vệ tinh ứng dụng trên tàu biển. Cấu hình hình cơ bản của sản phẩm:

- Tải trọng: max 50kg
- Kích thước: 1000 x 1000 x 850 mm
- Khả năng nâng hạ của 1 chân robot: max 300 mm
- Khả năng quay trái – phải: không giới hạn về góc và chiều
- Nguồn điện sử dụng: 24V DC – 10A
- Chuẩn giao tiếp: uART, USB

2.2. THIẾT KẾ VÀ CHẾ TẠO

2.2.1. *Giới thiệu phần mềm Solidworks*

Solidworks 2017 là phiên bản mới nhất của phần mềm Solidworks được phát triển bởi hãng Dassault Systèmes. Một phần mềm dùng để thiết kế 3D mạnh mẽ không chỉ có modul về thiết kế mà phần mềm Solidowrks 2017 còn có thêm các modul khác như lắp ráp, xuất bản vẽ, mô phỏng động học, mô phỏng động lực học, thiết kế khuôn, gia công sản phẩm.

Hình 2.5. Solidworks 2017

Ở phiên bản Solidworks 2017 này hãng đã cải thiện thêm các tính năng mới vào phần mềm.Cụ thể là có hơn 250 những tính năng được cải thiện trong môi trường CAD, những phản hồi này rất có ích và rất thực tế vì nó được cải thiện theo những yêu cầu và phản hồi của người sử dụng. ở phiên bản này người sử dụng có thể upload những mô hình CAD lên môi trường Internet, Solidworks 2017 cũng đã xây dựng những App sử dụng cho IOS và Android giúp người sử dụng tiện lợi hơn.

Những cải tiến mới rất cụ thể trên phần mềm Solidworks 2017 như sau:

- Khả năng sáng tạo mới nhằm giải quyết thiết kế PCB, cơ điện, IOT và hơn nữa.
- Xem thêm sức mạnh cốt lõi và hiệu suất để mở ra những khả năng mới.
- Khả năng mô phỏng mạnh mẽ cho việc tạo ra các thiết kế mang tính đột phá.
- Thiết kế quy trình công việc để loại bỏ các rào cản khi làm việc với dữ liệu của bên thứ ba
- Tích hợp dữ liệu từ các khái niệm thông qua sản xuất.

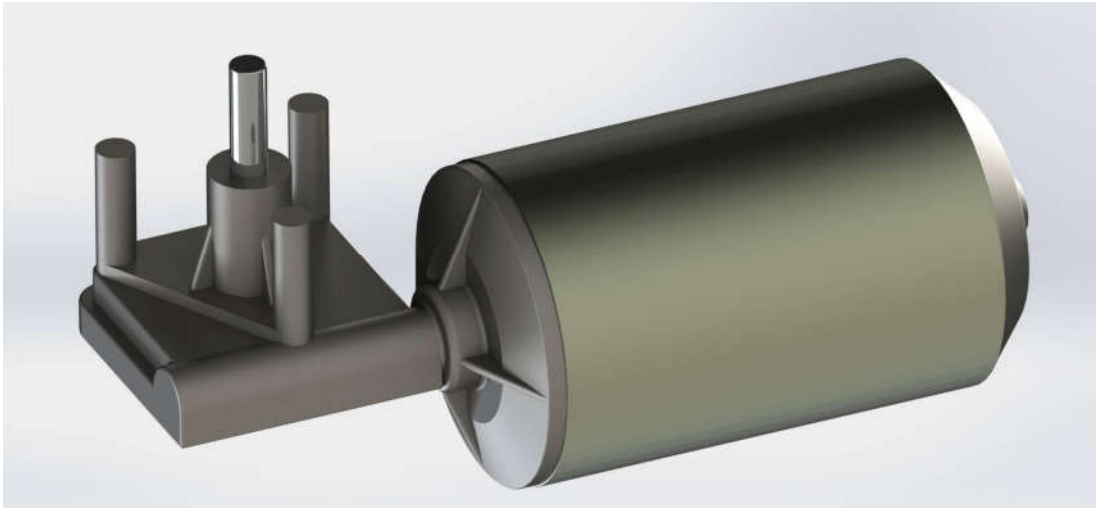
2.2.2. *Thiết kế, mô phỏng và chế tạo*

Với các tính toán trước đó, các bộ phận được đo đạc và thiết kế lại trong phần mềm 3D Solidworks trước khi bắt tay chế tạo thực tế.

Một số hình ảnh về quá trình thiết kế, mô phỏng và chế tạo được thể hiện dưới đây:

Hình 2.6. Động cơ tuyến tính (linear motor)

Hình 2.6 là hình ảnh của động cơ tuyến tính (linear motor). Đây là một trong 3 động cơ chính để mô phỏng lại chuyển động nghiêng trái – nghiêng phải, ngẩng lên – hạ xuống và tịnh tiến lên – xuống của mô hình robot. Như trong hình vẽ ta cũng thấy được đầu trục của cơ cấu vít-me đã được gắn một khớp nối dạng các-đăng cho phép chuyển động với 3 bậc tự do.

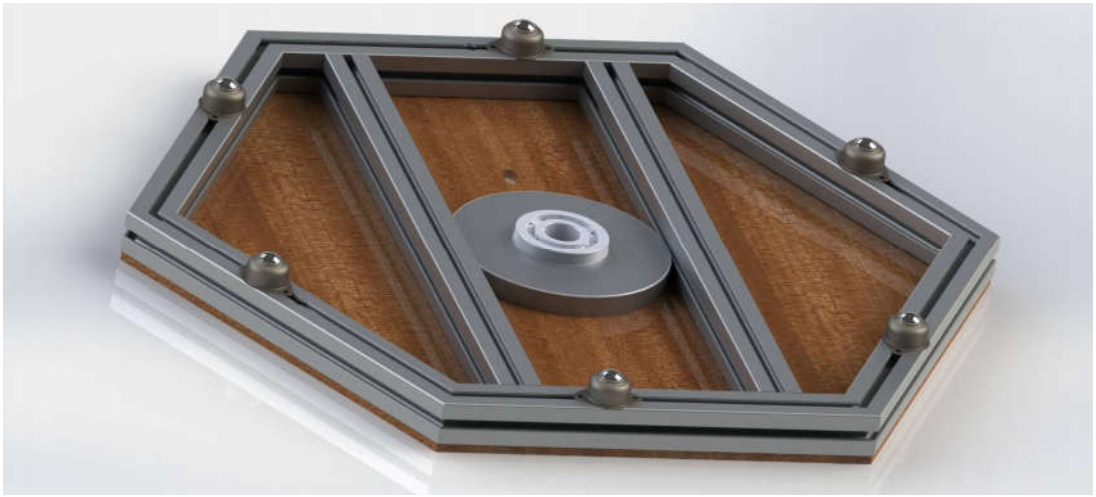


Hình 2.7. Động cơ trục quay

Động cơ trục quay trong Hình 2.7 có nhiệm vụ chính là kéo hệ thống mâm xoay chuyển động quay tròn quanh trục thông qua hệ thống bánh răng và dây đai. Kết quả của chuyển động quay quanh trục của mâm xoay sẽ mô phỏng lại sự thay đổi phương hướng di chuyển của tàu thủy khi đang hoạt động.

Hình 2.8. Đế cố định và các động cơ tuyến tính

Hình 2.8 là kết quả mô phỏng sau khi gắn các động cơ tuyến tính lên đế cố định. Đế cố định có dạng một tam giác đều với 3 góc nhọn được thiết kế lại để tạo thành một đa giác 6 cạnh. Toàn bộ khung đế được thiết kế và chế tạo bằng khung nhôm định hình đem lại một sản phẩm chắc chắn, dễ dàng ghép nối chế tạo.



Hình 2.9. Bàn động

Bàn động như trong Hình 2.9 được thiết kế dựa trên một lục giác đều lắp ghép từ các thanh nhôm định hình. Ở giữa bàn động được gắn một vòng bi côn. Vòng bi côn sẽ hỗ trợ cho chuyển động quay của mâm xoay.

Trên các cạnh của bàn động có gắn thêm 6 bi mắt trâu để đỡ mâm xoay trong trường hợp đặt các vật có tải trọng lớn lên mặt trên của mâm xoay.

Hình 2.10. Mâm xoay gắn trên bàn động

Hình 2.10 là kết quả sau khi gắn mâm xoay lên trên bàn động. Mâm xoay và bàn động được liên kết với nhau qua hệ thống bi côn. Trục của mâm xoay được thiết kế xuyên qua và không tiếp xúc với bàn động. Trục mâm xoay sẽ được gắn thêm bánh răng và kết nối đến động cơ quay.

Hình 2.11. Mô hình hoàn thiện của robot 4 bậc tự do

Hình 2.11 là mô hình hoàn thiện của robot 4 bậc tự do sau khi đã lắp ghép các bộ phận với nhau. Thông qua việc mô phỏng trên phần mềm, các cơ cấu được thể hiện một cách trực quan rõ ràng. Các hành trình, giới hạn cũng như liên động của các bộ phần hoàn toàn có thể mô phỏng bằng phần mềm. Dựa vào kết quả mô phỏng có thể điều chỉnh lại thiết kế để phù hợp với yêu cầu đã đặt ra của đề tài.

Chương 3

THIẾT KẾ ĐIỆN TỬ

3.1. THIẾT KẾ, LỰA CHỌN THIẾT BỊ

3.1.1. *Tính toán, lựa chọn động cơ*

Với yêu cầu thiết kế của robot 04 bậc tự do phải chịu được tải trọng 30 kg cũng như đáp ứng được các điều kiện:

- Phạm vi hoạt động góc nghiêng ±15^o
- Phạm vi hoạt động góc ngẩng ±15^o
- Phạm vi hoạt động góc phương vị (góc cuộn): 360^o
- Tốc độ quay góc nghiêng: 12^o/s
- Tốc độ quay góc ngẩng: 12^o/s
- Tốc độ quay góc phương vị (góc cuộn): 12^o/s

Qua quá trình tìm hiểu các loại động cơ có trên thị trường cũng như tính phù hợp với đề tài luận văn, loại động cơ DC 24V đã được lựa chọn cho hệ robot 04 bậc tự do.

Động cơ DC 24V có các ưu điểm:

- Có nhiều chủng loại động cơ sẵn có trên thị trường, dễ dàng lựa chọn thông số theo thiết kế, giá thành rẻ.
- Đơn giản về mặt điều khiển (mạch điều khiển công suất, các tín hiệu điều khiển chỉ có chiều quay – 02 chân tín hiệu I/O và tốc độ quay – 01 chân tín hiệu PWM).

Trong khi đó động cơ servo và động cơ bước có ưu điểm về độ chính xác điều khiển, momen lực lớn hơn ... nhưng khó tìm loại thiết kế theo kiểu tuyến tính (linear motor, putter) và có giá thành cao cũng như mạch điều khiển công suất phức tạp.

Hình 3.1. Động cơ tuyến tính

03 động cơ nâng hạ đã lựa chọn là loại được thiết kế cho việc đóng mở cửa tự động, cửa gara, nâng hạ vật nặng... Thông số của 03 động cơ nâng hạ như sau:

- Mã hiệu:
- Điện áp sử dụng: 24V DC
- Công suất:
- Tải trọng: 300N
- Tốc độ: 100 mm/s
- Hành trình: 300 mm
- Giới hạn hành trình: tích hợp sẵn công tắc giới hạn
- Cấp bảo vệ: IP43

Hình 3.2. Động cơ quay

Động cơ quay đã lựa chọn là loại động cơ DC 24V sử dụng trong cửa cuốn. Thông số cụ thể như sau:

- Mã hiệu:
- Điện áp sử dụng: 24V DC
- Công suất: 50W
- Tải trọng:
- Tốc độ: 50 vòng/phút

3.1.2. *Tính toán, lựa chọn encoder*

Để giám sát chuyển động của bàn máy, robot cần sử dụng các encoder để xác định số vòng quay của động cơ, từ đó tính toán ra vị trí hiện tại của thanh trượt. Việc tính toán để lựa chọn encoder dựa trên các thông số:

- Tốc độ nâng hạ: 100 mm/s
- Tốc độ động cơ nâng hạ: 3600 vòng/phút
- Độ phân giải cần thiết (để tính toán PID): 200 xung/mm
- Độ phân giải encoder (đơn vị: xung/vòng) được tính theo công thức:

$$\text{Độ phân giải encoder} = \frac{\text{Độ phân giải cần thiết*Tốc độ nâng hạ*60}}{\text{Tốc độ động cơ nâng hạ}} = 333.33$$

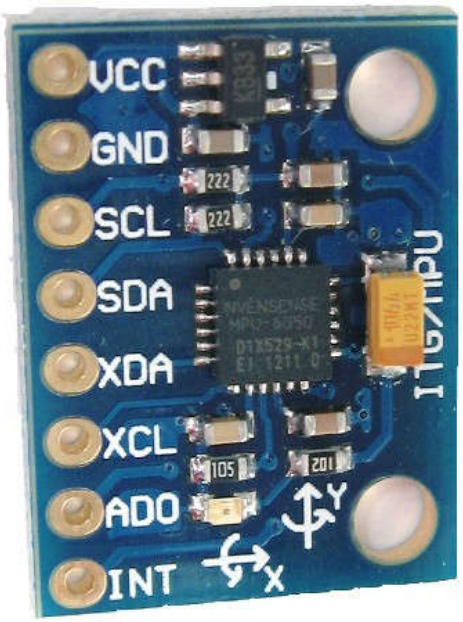
Thông số encoder đã lựa chọn như sau:

- Điện áp sử dụng: 5V DC
- Loại encoder: tương đối
- Số kênh: 2 kênh A, B
- Độ phân giải 334 xung/vòng

Hình 3.3. Encoder 334 xung/vòng

3.1.3. *Cảm biến chuyển động MPU 6050*

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.



Hình 3.4. Cảm biến chuyển động MPU 6050

- MPU-6050 sử dụng công nghệ độc quyền MotionFusion của InvenSense có thể chạy trên các thiết bị di động, tay điều khiển...
- 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 - DMP) 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 tức là 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:

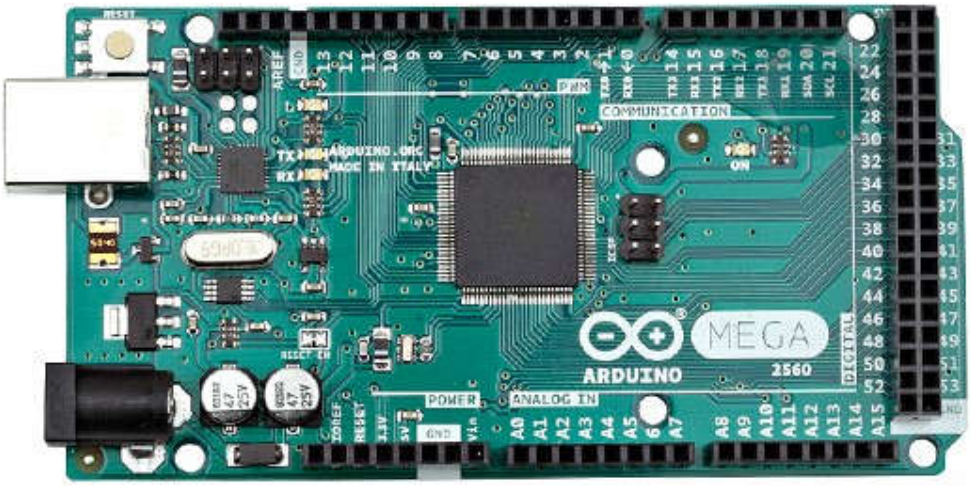
- Con quay hồi chuyển: ± 250 500 1000 2000 dps
- Gia tốc: ± 2 ± 4 ± 8 ± 16g
- Hơn nữa, MPU-6050 có sẵn bộ đệm dữ liệu 1024 byte cho phép vi điều khiển phát lệnh cho cảm biến, và nhận về dữ liệu sau khi MPU-6050 tính toán xong.
- Các thông số kĩ thuật khác của module MPU-6050
- Nguồn: 3-5V, trên module MPU-6050 đã có sẵn LDO chuyển nguồn 5V sang 3V
- Giao tiếp I2C ở mức 3V
- Khoảng cách chân cắm: 2.54mm
- Địa chỉ: 0x68, có thể cấp mức cao vào chân AD0 để chuyển địa chỉ thành 0x69

3.1.4. *Bộ KIT điều khiển Arduino MEGA 2560*

Arduino Mega 2560 là phiên bản nâng cấp của Arduino Mega hay còn gọi là Arduino Mega 1280. Sự khác biệt lớn nhất với Arduino Mega 1280 chính là chip nhân.

Ở Arduino Mega 1280 sử dụng chip ATmega1280 với flash memory 128KB, SRAM 8KB và EEPROM 4 KB.

Còn Arduino Mega 2560 là phiên bản hiện đang được sử dụng rộng rãi và ứng dụng nhiều hơn. Với chip ATmega2560 có bộ nhớ flash memory 256 KB, 8KB cho bộ nhớ SRAM, 4 KB cho bộ nhớ EEPROM. Giúp cho người dùng thêm khả năng viết những chương trình phức tạp và điều khiển các thiết bị lớn hơn như máy in 3D, điều khiển robot.



Hình 3.5. KIT Arduino Mega 2560

Arduino Mega 2560 là một vi điều khiển hoạt động dựa trên chip ATmega2560. Bao gồm:

- 54 chân digital (trong đó có 15 chân có thể được sủ dụng như những chân PWM là từ chân số 2 → 13 và chân 44 45 46).
- 6 ngắt ngoài: chân 2 (interrupt 0), chân 3 (interrupt 1), chân 18 (interrupt 5), chân 19 (interrupt 4), chân 20 (interrupt 3), and chân 21 (interrupt 2).
- 16 chân vào analog (từ A0 đến A15).
- 4 cổng Serial giao tiếp với phần cứng:
- 1 thạch anh với tần số dao động 16 MHz.
- 1 cổng kết nối USB.
- 1 jack cắm điện.
- 1 đầu ICSP.
- 1 nút reset.

Đặc biệt với các ứng dụng liên quan đến Matlab thì Arduino Mega 2560 cũng là một sự chọn lựa tuyệt vời. Nó còn được tích hợp sẵn thư viện dành cho Matlab.

Arduino Mega 2560 có thể sử dụng hầu hết các shiled dành cho các mạch Arduino Uno hay hoặc các mạch trước đây như Duemilanove hay Diecimila với cách cài đặt và nối chân tương tự như Arduino Uno.

Thông số kĩ thuật:

- Chip xử lý: ATmega2560
- Điện áp hoạt động: 5V

- Điện áp vào (đề nghị): 7V-15V
- Điện áp vào (giới hạn): 6V-20V
- Cường độ dòng điện trên mỗi chân 3.3V: 50 mA
- Cường độ dòng điện trên mỗi chân I/O: 20 mA
- Bộ nhớ Flash: 256 KB
- SRAM: 8 KB
- EEPROM: 4 KB
- Tốc độ xung nhịp: 16 MHz

3.1.5. *Mạch điều khiển động cơ DC*

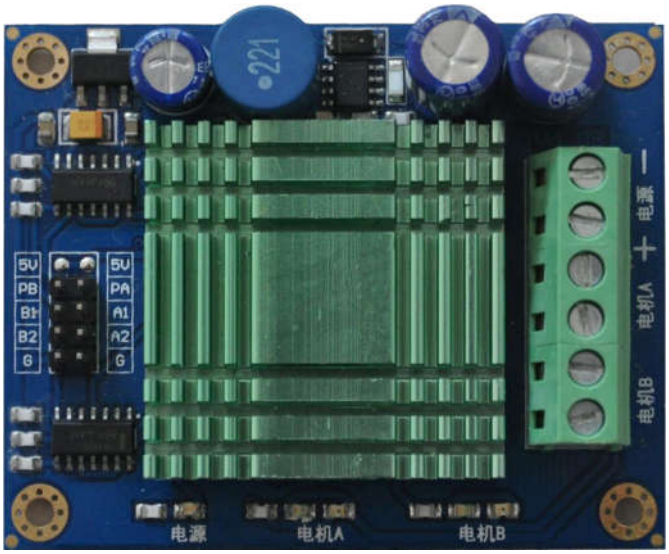
Để điều khiển các động cơ, tín hiệu điều khiển sẽ được gửi từ bộ KIT điều khiển Arduino MEGA 2560 thông qua mạch điều khiển động cơ, động cơ sẽ quay thuận hay ngược chiều kim đồng hồ với tốc độ được thay đổi theo độ rộng xung PWM.

Mạch điều khiển động cơ DC là bo mạch được mua sẵn trên thị trường có thông số như sau:

- Dải điện áp sử dụng: 12 – 30V DC
- Dòng điện tối đa: 60A
- Dòng điện liên tục: 25A
- Nguyên lý: 02 mạch cầu H tích hợp cách ly quang

Tín hiệu điều khiển:

- 5V – nguồn 5V
- PA/PB – tín hiệu PWM
- A1/B1 và A2/B2 sẽ kết hợp để xác định chiều quay của động cơ.
    - A1/B1 = 1, A2/B2 = 0 động cơ A/B quay thuận;
    - A1/B1 = 0, A2/B2 = 1 động cơ A/B quay ngược.
    - A1/B1 = 0, A2/B2 = 0 hoặc A1/B1 = 1, A2/B2 = 1 động cơ sẽ khóa.
- GND – nối đất



Hình 3.6. Mạch điều khiển động cơ

3.1.6. ***Nguồn điện***

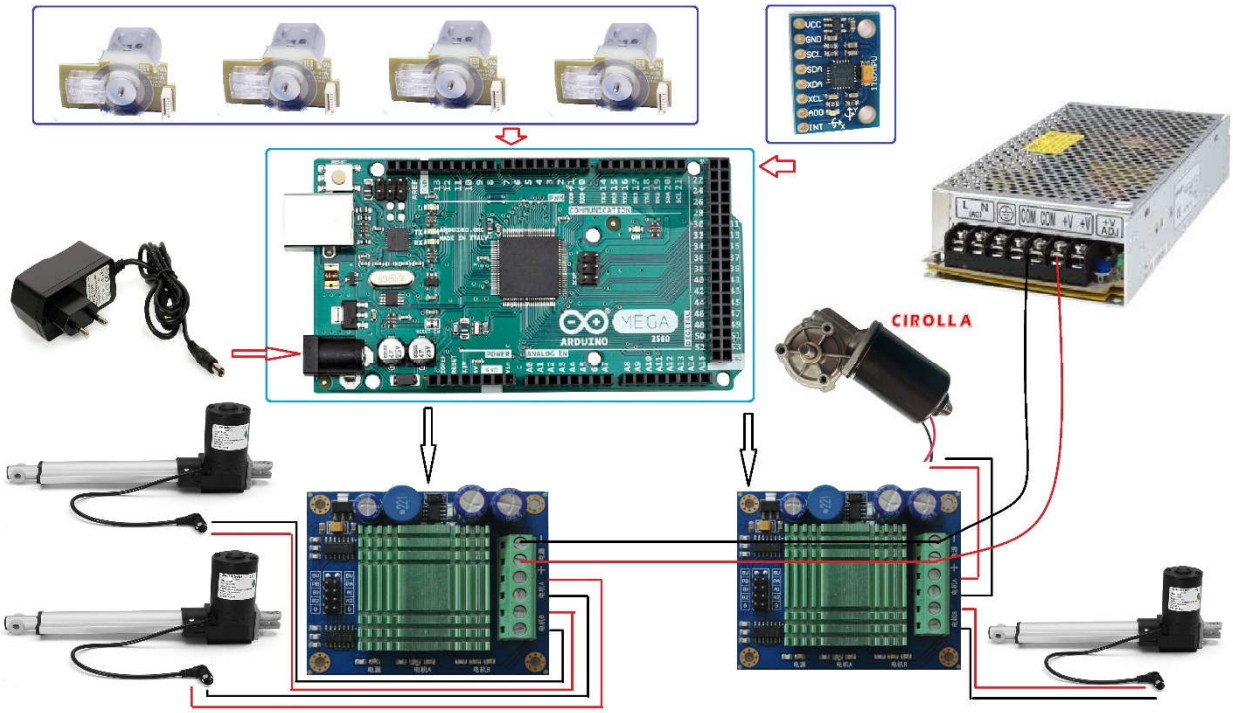
Nguồn điện sử dụng để cung cấp cho các động cơ là loại nguồn xung 24V DC – 10A.

Hình 3.7. Nguồn xung 24V DC – 10A

Nguồn điện cung cấp cho bộ KIT Arduino MEGA 2560 và các cảm biến, encoder... là nguồn xung 12V DC – 1A. Đề tài sử dụng 02 nguồn khác nhau để tránh xung nhiễu từ động cơ tác động lên chip vi xử lý cũng như các cảm biến.

Hình 3.8. Nguồn xung 12V DC – 1A

3.2. MẠCH ĐIỆN VÀ CÁCH GHÉP NỐI



Hình 3.9. Sơ đồ ghép nối hệ thống

Hình 3.9 là sơ đồ ghép nối của hệ thống robot 4 bậc tự do. Hệ thống gồm 5 phần chính:

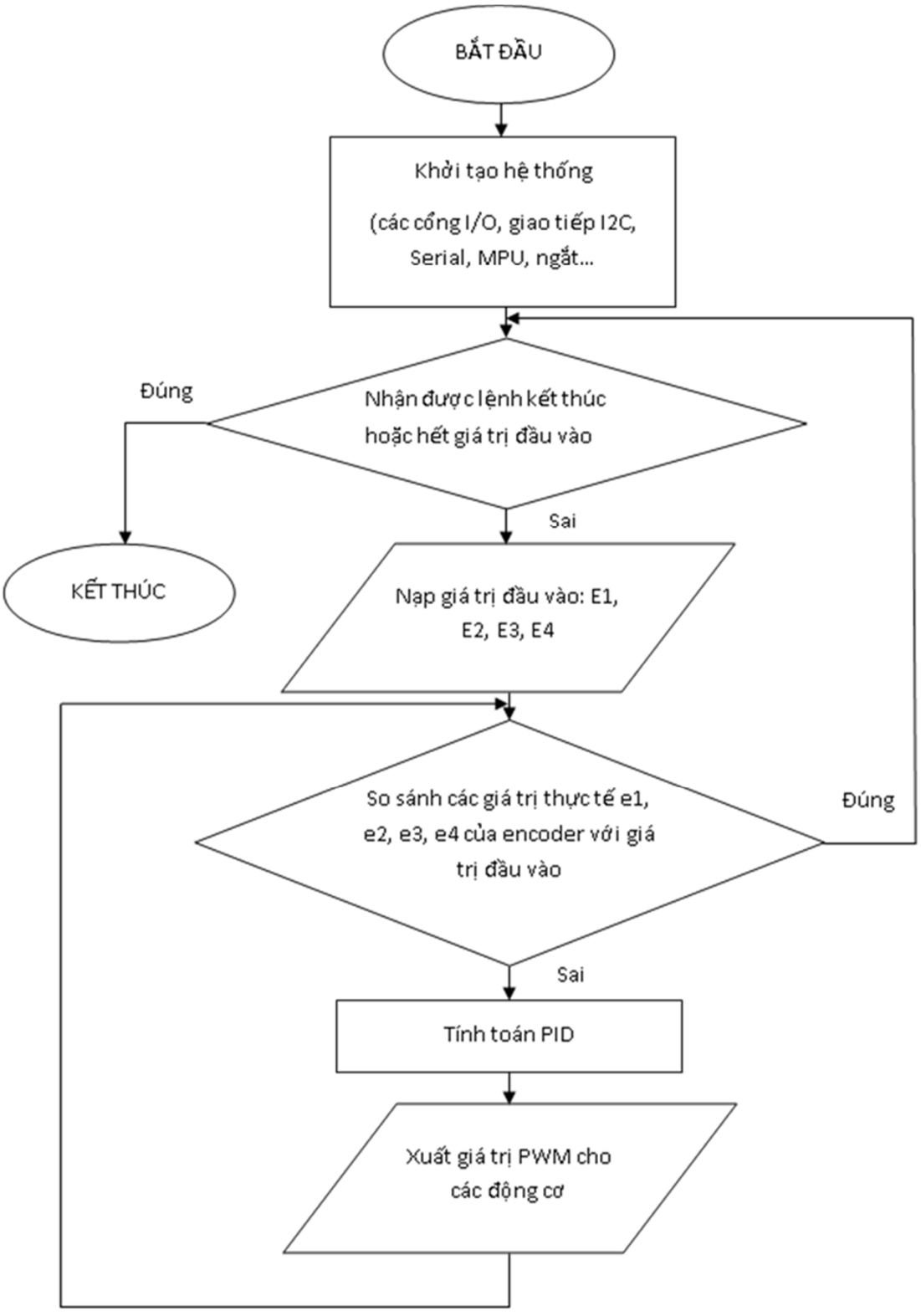
- Khối cảm biến: bao gồm 4 encoder và 1 cảm biến chuyển động. 4 encoder kết nối với mạch điều khiển trung tâm là bộ KIT Arduino MEGA 2560 qua các chân ngắt ngoài. Cảm biến chuyển động được thiết lập kết nối I2C để truyền dữ liệu cho bộ điều khiển xử lý.
- Khối mạch điều khiển trung tâm: bộ KIT Arduino MEGA 2560 là nơi thu thập toàn bộ dữ liệu từ cảm biến, tính toán xử lý các dữ liệu đó và đưa ra các tín hiệu điều khiển.

- Khối mạch điều khiển động cơ: bao gồm 2 mạch điều khiển động cơ, mỗi mạch gồm 2 kênh. Nhận tín hiệu điều khiển từ mạch điều khiển trung tâm và điều khiển hoạt động của các động cơ.
- Khối cơ cấu chấp hành – động cơ: bao gồm 3 động cơ tuyến tính để nâng hạ, thay đổi phương hướng của bàn động và 1 động cơ quay để tạo chuyển động quay cho hệ thống.
- Khối nguồn: bao gồm 2 nguồn xung riêng biệt, 1 nguồn cấp cho mạch điều khiển trung tâm và khối cảm biến, nguồn còn lại cấp cho mạch điều khiển động cơ.

Chương 4

THIẾT KẾ CHƯƠNG TRÌNH ĐIỀU KHIỂN

4.1. SƠ ĐỒ THUẬT TOÁN



Hình 4.1. Sơ đồ thuật toán

4.2. THUẬT TOÁN PID VÀ BỘ LỌC SỐ

4.2.1. *Thuật toán PID*

4.2.1.a. Giới thiệu về thuật toán PID

Một bộ điều khiển PID (PID controller) là một bộ phận điều khiển phản hồi kín được sử dụng rộng rãi trong các hệ thống điều khiển trong công nghiệp. Bộ điều khiển PID sẽ cố gắng sửa sai số giữa biến hệ thống (process variable) đo được với điểm đặt trước (set point) bằng cách tính toán và đưa ra lệnh điều khiển tác động vào tiến trình một cách nhanh chóng và chuẩn xác, nhằm giữ cho sai số luôn ở mức thấp nhất [9].

Tính toán điều khiển PID (hay còn gọi là thuật toán PID) liên quan đến 3 tham số riêng biệt: tham số tỉ lệ, tham số tích phân và tham số vi phân. Tham số tỉ lệ ảnh hưởng tới tác động bù trừ cho sai số hiện tại. Tham số tích phân quyết định tác động dựa trên tổng các sai số, và tham số vi phân điều chỉnh tác động dựa vào mức thay đổi của sai số. Kết quả tổng cộng của cả 3 tác động này sẽ được sử dụng để điều chỉnh tiến trình qua các thành phần điều khiển như vị trí của van hay dòng áp của nguồn điện cấp cho hệ thống đốt nóng.

Bằng cách điều chỉnh 3 tham số trong thuật toán PID, bộ điều khiển có thể đưa ra các tác động phù hợp với yêu cầu cụ thể của tiến trình cần điều khiển. Đáp ứng của hệ thống có thể được mô tả bằng khả năng điều chỉnh khi có sai số, mức độ tăng vọt khỏi điểm thiết đặt và mức độ dao động của hệ thống.

Một số ứng dụng chỉ cần sử dụng một hay hai tham số để điều khiển. Lúc đó bộ điều khiển PID thường được gọi là bộ điều khiển PI, PD hay P, I tương ứng với các tham số dùng để điều khiển. Bộ điều khiển PI thường hay được sử dụng trong thực tế, do tác động vi phân thường hay nhạy cảm với nhiễu của các phép đo, và nếu không có tác động tích phân thì hệ thống thường không đạt được trạng thái cần đặt do giới hạn thực tế của các thành phần điều khiển.

Đầu ra của bộ điều khiển PID thường là một biến điều khiển (manipulated variable - MV). Ta có thể viết:

$$\mathrm{MV(t)} = P_{\mathrm{out}} + I_{\mathrm{out}} + D_{\mathrm{out}}$$ [10]

Trong đó Pout, Iout và Dout lần lượt là tác động của các thành phần tỉ lệ, tích phân và vi phân.

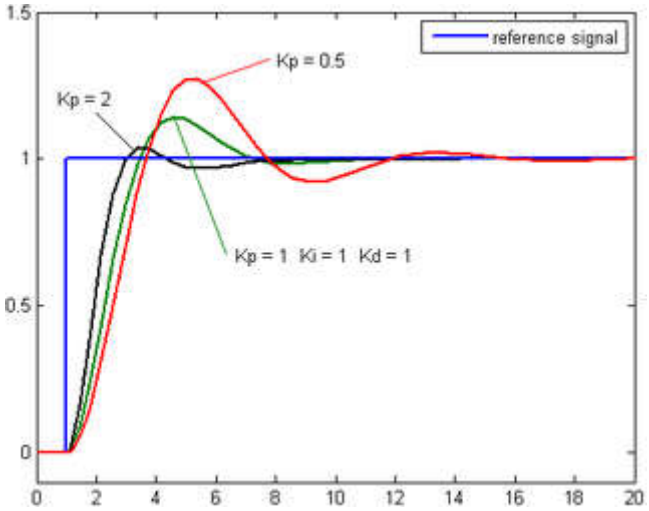
Thành phần tỉ lệ (hay đôi khi còn được gọi là thành phần khuếch đại - gain) làm cho đầu ra thay đổi tỉ lệ với sai số tức thời. Đáp ứng tỉ lệ có thể được thay đổi bằng cách điều chỉnh hệ số tỉ lệ Kp:

$$P_{\mathrm{out}} = K_p\, e(t)$$

Trong đó e là sai số, e = SP – PV với SP là điểm đặt trước, PV là biến trạng thái.

Hệ số tỉ lệ càng lớn sẽ làm cho biến điều khiển thay đổi càng lớn khi có thay đổi sai số. Nếu hệ số tỉ lệ quá lớn sẽ làm cho hệ thống mất ổn định hay dao động. Ngược lại, hệ số tỉ lệ nhỏ sẽ làm cho biến điều khiển thay đổi quá ít khi sai số lớn, dẫn đến 1 hệ thống đáp ứng chậm. Nếu hệ số tỉ lệ quá bé sẽ dẫn đến việc biến điều khiển quá nhỏ để có thể phản ứng lại các thăng giáng của hệ thống.

Khi không có thay đổi, một bộ điều khiển hoàn toàn tỉ lệ sẽ không đưa hệ thống về được trạng thái thiết đặt trước, mà sẽ giữ ở một trạng thái cân bằng với sai số phụ thuộc vào hệ số tỉ lệ và độ tăng ích của tiến trình. Mặc dù vậy, cả về lý thuyết điều chỉnh và thực tế trong công nghiệp đều cho thấy thành phần tỉ lệ thường nên đóng vai trò chính trong việc làm thay đổi đầu ra của hệ thống.

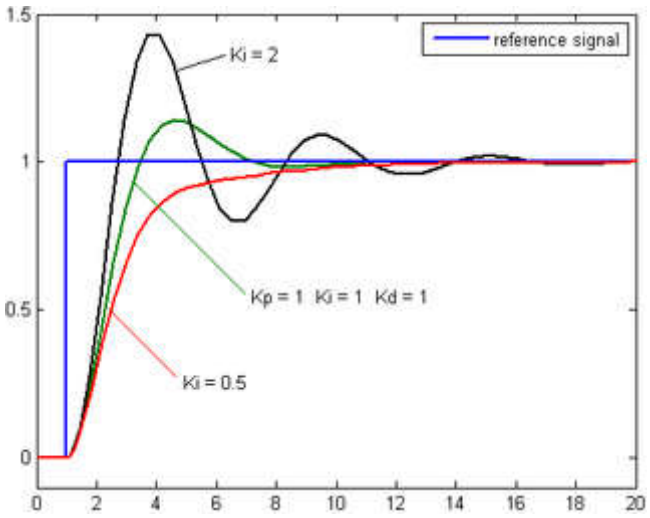


Hình 4.2. Tác động của hệ số tỉ lệ tới đầu ra của hệ thống

Thành phần tích phân tỉ lệ với cả độ lớn của sai số lẫn thời gian kéo dài của sai số. Các sai số trước kia sẽ được tích luỹ và thêm vào đầu ra biến điều khiển sau khi nhân với hệ số tích phân Ki:

$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

Thành phần tích phân giúp tăng tốc quá trình tiến trình đạt được trạng thái thiết đặt và loại bỏ lỗi ở trạng thái cân bằng ở bộ điều khiển hoàn toàn tỉ lệ. Tuy nhiên do thành phần tích phân là tích luỹ của sai số từ trước nên nó có thể khiến cho trạng thái hiện tại bị vượt quá trạng thái thiết đặt (overshot) và dẫn đến mất ổn định của hệ thống.

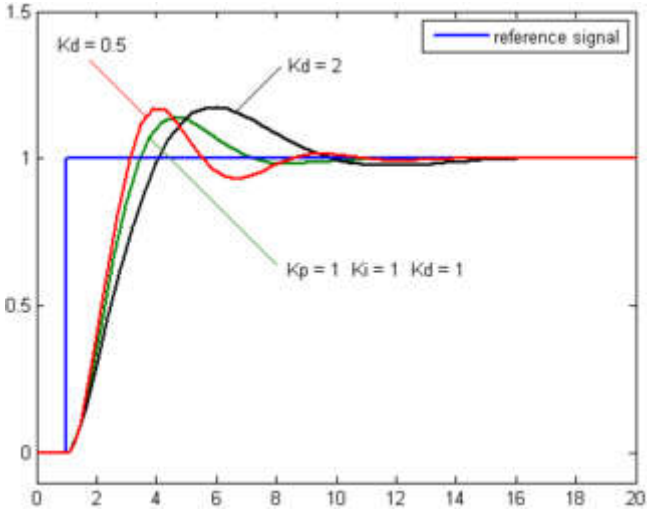


Hình 4.3. Tác động của hệ số tích phân tới đầu ra của hệ thống

Tốc độ thay đổi sai số của tiến trình được tính bằng cách xác định độ dốc của sai số theo thời gian (đạo hàm bậc nhất của sai số theo thời gian), và nhân với hệ số vi phân Kd.

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

Thành phần vi phân làm giảm tốc độ thay đổi của biến điều khiển khi hệ thống gần đạt được trạng thái thiết đặt. Vì vậy thành phần tỉ lệ được dùng để giảm độ vượt quá gây nên bởi thành phần tích phân và cải thiện độ ổn định của hệ thống. Tuy nhiên thành phần vi phân sẽ khuếch đại nhiễu, do vậy nó rất nhạy cảm với nhiễu của đầu vào bộ điều khiển, và có thể khiến cho hệ thống trở nên mất ổn định khi nhiễu và hệ số vi phân đủ lớn.

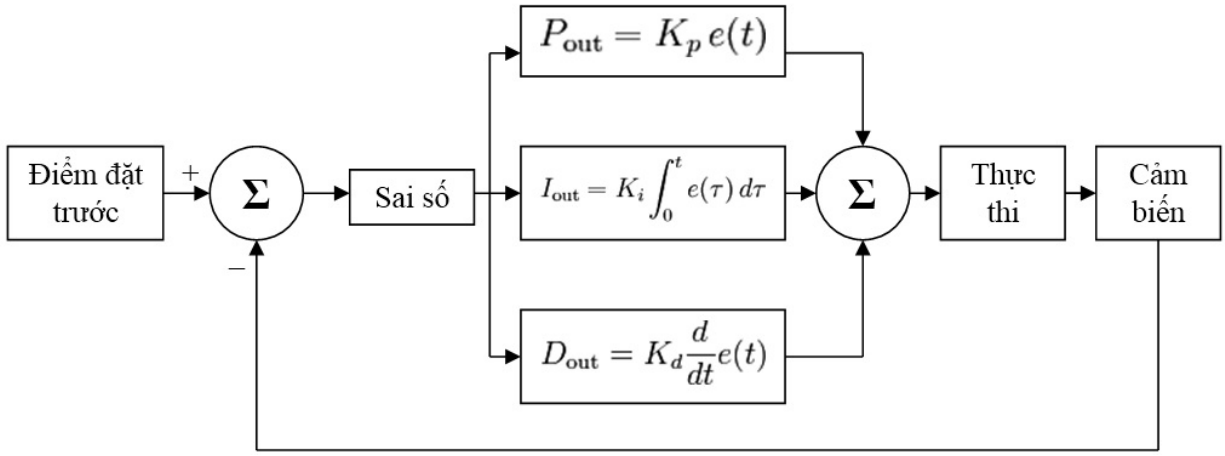


Hình 4.4. Tác động của hệ số vi phân tới đầu ra của hệ thống

Như vậy, nếu gọi u(t) là đầu ra của bộ điều khiển PID thì thuật toán PID có thể được biểu diễn dưới dạng [11]:

$$\mathrm{u(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{d}{dt} e(t)$$

Bài toán của việc thiết kế bộ điều khiển PID trở thành bài toán xác định các hệ số K_p, K_i và K_d (hay còn gọi là điều chỉnh bộ điều khiển - tunning).



Hình 4.5. Sơ đồ khối của bộ điều khiển PID

4.2.1.b. Ứng dụng điều khiển PID cho robot 4 bậc tự do

Trong hệ thống robot 4 bậc tự do mô phỏng chuyển động trên tàu thủy, bộ điều khiển PID được áp dụng trong để luôn đảm bảo các thanh trượt, mâm xoay di chuyển đến đúng vị trí. Tổ hợp vị trí của các cơ cấu cơ khí sẽ cho ra vị trí của khâu cuối cùng đúng theo tính toán và yêu cầu đặt ra. Quá trình mô phỏng chuyển động trên tàu thủy sẽ diễn ra liên tục nên các vị trí đặt ra cho hệ thống robot sẽ liên tục thay đổi, hay nói cách khác, bộ điều khiển PID sẽ có điểm đặt trước (set point) thay đổi được.

Với các hệ thống đơn giản (ít cơ cấu, ít yếu tố tác động…) có thể dễ dàng xây dựng mô hình toán học, tính toán hàm truyền và dựa trên đó sẽ đưa ra được bộ điều khiển tối ưu nhất. Nhưng với hệ thống robot 4 bậc tự do có cơ hệ phức tạp, nhiều yếu tố tác động nên việc xây dựng mô hình toán học là khó khăn, phải dựa vào thực nghiệm nhiều để đưa ra việc tối ưu hệ thống. Cũng vì lý do trên, hệ thống phải sử dụng bộ điều khiển với cả 3 tác động tuyến tính, tích phân và vi phân. Điều này khiến cho việc căn chỉnh hệ thống phức tạp, và cần phải sử dụng một số phụ trợ để hệ thống có thể hoạt động được ổn định.

Ở đây nhất thiết phải dùng thành phần tích phân vì 2 lý do. Thứ nhất là do điểm đặt trước có thể thay đổi được, do vậy khi đặt điểm đặt trước cách xa với vị trí thực tế, đòi hỏi phải có hệ số tích phân lớn để hệ có thể nhanh chóng đạt được vị trí đã đặt. Thứ hai là cơ hệ sử dụng động cơ với kiểu truyền động bánh vít – trục vít có tỉ số truyền thấp, khoảng chết lớn, nếu chỉ dùng hoàn toàn tác động tuyến tính sẽ dẫn đến sai số ở trạng thái cân bằng lớn, nhất là khi điểm đặt trước ở vị trí hiện tại.

Do có hệ số tích phân lớn, đồng thời cơ cấu bánh vít – trục vít bị trượt do tải trọng lớn khi cơ cấu đã đạt vị trí đặt trước, dẫn đến việc cần thiết phải có thành phần vi phân để bù trừ tránh cho hệ thống bị không đạt được điểm đặt trước và giúp ổn định hệ thống. Tuy nhiên việc xác định hệ số vi phân tối ưu cho ứng dụng là rất khó khăn vì thông số này rất nhạy cảm với nhiễu và sự thay đổi điểm đặt, và nếu không kiểm tra cẩn thận rất có thể một hệ thống dường như hoàn hảo sẽ dao động khi bị một nhiễu loạn lớn tác động vào.

4.2.1.c. Lựa chọn bộ thông số PID

Nếu các tham số PID (tỉ lệ, tích phân và vi phân) không được lựa chọn thích hợp, tiến trình cần điều khiển có thể bị mất ổn định, đầu ra bị phân kì và có thể kèm theo dao động, và chỉ bị giới hạn bởi sự bão hoà hoặc nứt gãy cơ khí. Điều chỉnh hệ điều khiển là công việc đặt các thông số tỉ lệ, tích phân và vi phân về các giá trị tối ưu để có đáp ứng đầu ra theo mong muốn [12].

Đáp ứng tối ưu của tiến trình thay đổi theo từng ứng dụng. Một số tiến trình không cho phép đầu ra bị vượt quá điểm đặt, ví dụ như vì lý do an toàn. Một số tiến trình lại cần giảm thiểu năng lượng cần thiết để đầu ra đạt được điểm đặt mới. Thông thường, sự ổn định của đáp ứng đầu ra là cần thiết và tiến trình không được phép dao động trong bất kỳ điều kiện nào và với bất kỳ điểm đặt nào. Một số tiến trình có độ phi tuyến nhất định, và có thể các tham số làm việc tốt ở điều kiện đầy tải sẽ không làm việc được khi tiến trình bắt đầu ở tình trạng không tải.

Có nhiều phương pháp để điều chỉnh hệ PID. Phương pháp hiệu quả nhất thường yêu cầu tìm ra một mô hình toán học cho tiến trình, sau đó chọn P, I và D dựa trên các thông số động trong mô hình đó. Tuy nhiên không phải lúc nào cũng có thể xây dựng được mô hình phù hợp cho tiến trình một cách nhanh chóng hoặc không tốn kém nên các phương pháp điều chỉnh bằng tay vẫn được sử dụng phổ biến trong thực tế. Các phương pháp thường hay được dùng là: điều chỉnh bằng tay, phương pháp Ziegler – Nichols, sử dụng công cụ phần mềm và phương pháp Cohen – Coon.

Phương pháp điều chỉnh bằng tay thường bắt đầu bằng việc đặt các tham số Ki và Kd bằng 0. Sau đó tăng Kp đến khi hệ thống bắt đầu dao động. Tại đó, Kp sẽ được đặt lại bằng khoảng ½ giá trị này, tiếp tục tăng Ki đến khi bù hết được độ sai số khi ở trạng thái cân bằng và hệ thống đạt được điểm đặt với tốc độ hợp lý. Tiếp đó là điều chỉnh Kd nếu cần đến khi hệ thống đạt được điểm đặt với thời gian đủ ngắn khi tải bị thay đổi hoặc có nhiễu loạn trong hệ thống. Tuy nhiên nếu đặt Kd quá cao có thể dẫn đến phản ứng quá lớn và bị vượt mức đặt hoặc dẫn đến hệ thống mất ổn định. Một hệ PID có tốc độ đáp ứng nhanh thường được điều chỉnh để hơi vượt mức một chút để có thể đạt được mức đặt nhanh hơn. Tuy nhiên một số tiến trình không cho phép bị vượt mức, yêu cầu phải có hệ

điều khiển "bù quá mức", với hệ số Kp được đặt thấp hơn nhiều giá trị làm hệ thống bắt đầu dao động. Lợi thế của phương pháp là có thể đạt được một hệ thống có đáp ứng đầu ra như ý muốn, và không nhất thiết phải có mô hình toán học chi tiết của hệ thống. Tuy nhiên nhược điểm của phương pháp này là mất nhiều thời gian và cần có chuyên gia có nhiều kinh nghiệm.

Một phương pháp điều chỉnh khác thường được biết đến với tên phương pháp Ziegler – Nichols. Trong phương pháp này, đầu tiên Ki và Kd được đặt bằng 0. Kp sẽ được tăng đến một giá trị tới hạn Kc, ở đó đầu ra của hệ thống bắt đầu dao động. K_C và chu kỳ dao động P_C sẽ được sử dụng để đặt các tham số còn lại như sau:

$$K_P = 0.6\ K_C$$

$$K_i = 2K_p / P_C$$

$$K_d = K_p P_c / 8$$

Phương pháp này cho kết quả tương đối tốt cho một hệ PID đa năng, và không yêu cầu mô hình toán học tốt cũng như nhân lực có kinh nghiệm, tuy nhiên với các yêu cầu cụ thể thì thường nó không đạt được kết quả tối ưu.

Nhiều xí nghiệp hiện đại giờ đây chuyển sang sử dụng các phần mềm điều chỉnh PID và tối ưu hoá điều khiển để đảm bảo có các kết quả đảm bảo. Các chương trình này sẽ thu thập số liệu, thiết kế mô hình hệ thống, và gợi ý các điều chỉnh tối ưu. Một số chương trình còn có thể tự điều chỉnh hệ thống bằng cách thu thập các số liệu khi thay đổi điểm đặt.

Các phương pháp điều chỉnh PID toán học thường thay đổi điểm đặt hoặc tạo nên một thay đổi xung trong hệ thống, sau đó dựa vào phân tích tần số của đáp ứng xung để thiết kế hệ PID. Trong các tiến trình có thời gian đáp ứng lớn (như các ứng dụng liên quan đến nhiệt), thường sử dụng các phương pháp điều chỉnh PID toán học vì các phương pháp thử và sai có thể mất vài ngày mới đạt được một bộ giá trị tham số để hệ thống ổn định. Các giá trị tối ưu thường khó có thể xác định bằng các phương pháp toán học. Một số các bộ điều khiển số tích hợp khả năng tự điều chỉnh để hệ thống dần dần tự tìm lấy các giá trị tham số tối ưu.

Yêu cầu của hệ thống khi hoạt động là phải đảm bảo đạt được vị trí đặt trước. Trong quá trình thử nghiệm, khi đạt được vị trí động cơ dừng chuyển động và tải trọng của vật thử nghiệm sẽ làm các cơ cấu trượt xuống dẫn đến không đạt được yêu cầu. Đồng thời khi đạt được vị trí, hệ không được phép dao động nên phương pháp Ziegler – Nichols không đáp ứng được nhu cầu, đồng thời mô hình toán học của hệ thống không đủ chi tiết để có thể tính toán các hệ số theo mô hình toán, do vậy phương pháp được sử dụng là phương pháp điều chỉnh bằng tay. Để thuận tiện cho việc quan sát và thu thập số liệu đáp

ứng xung của hệ thống, ứng dụng có sẵn của Arduino IDE là Serial Monitor và Serial Plotter giúp cho việc điều chỉnh được đơn giản và chính xác hơn.

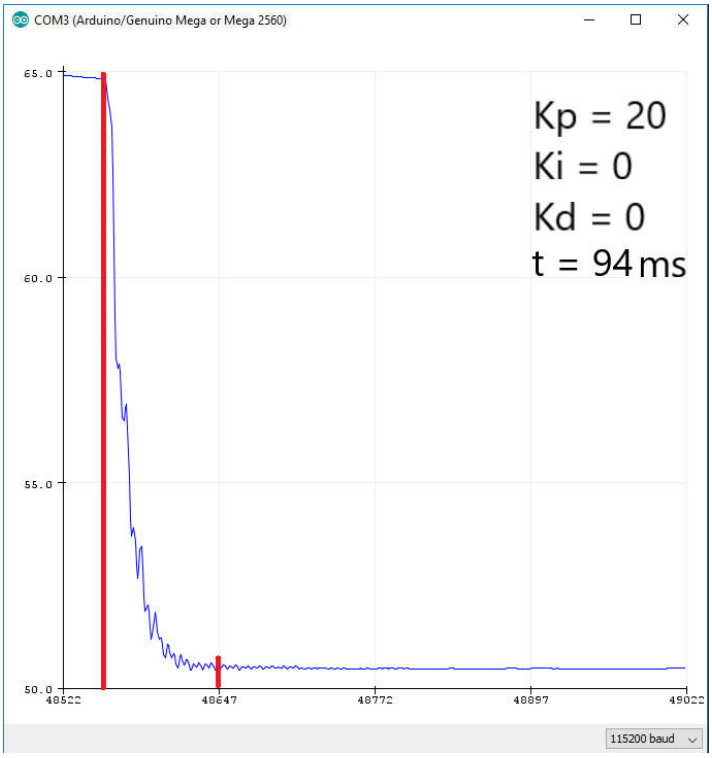
4.2.1.d. Thử nghiệm thực tế bộ thông số PID

Để tiến hành việc lựa chọn bộ thông số PID sử dụng cũng như kiểm chứng các tác động thực tế đến hệ thống khi thay đổi từng tham số, luận văn đã đưa ra một số thử nghiệm. Trong quá trình thực hiện luận văn, việc thử nghiệm được tiến hành rất nhiều lần với nhiều giá trị khác nhau, do đó những kết quả trình bày dưới đây là những chọn tiêu biểu, cung cấp một cái nhìn khái quát về những bước hoàn thành đề tài.

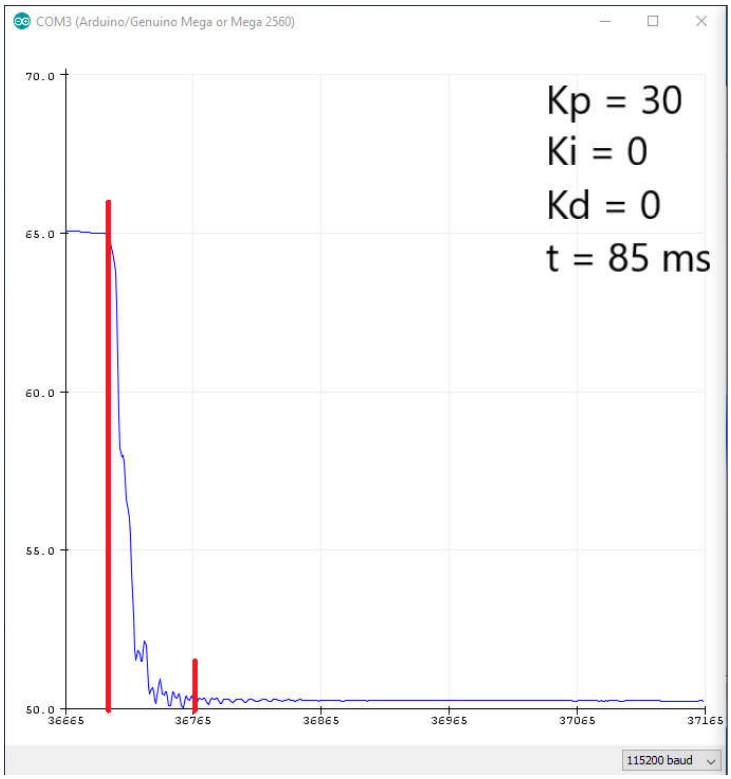
Các thử nghiệm này được áp dụng với động cơ quay trái – quay phải, giá trị đặt trước và giá trị phản hồi được biểu diễn bằng góc quay (độ) thông qua cảm biến chuyển động. Trong thử nghiệm, giá trị góc quay ban đầu của hệ là 65^o và giá trị góc đặt trước (hay giá trị góc cần đạt được) là 50^o. Giá trị góc thực tế và đáp ứng của hệ thống được ghi lại qua giao diện Serial Ploter tích hợp sẵn trên môi trường lập trình của bo mạch Arduino. Các trục tọa độ được biểu diễn tương ứng với giá trị thời gian (trục hoành – đơn vị là ms) và giá trị góc hiện tại (trục tung – đơn vị là độ).

- Thử nghiệm bộ thông số PID với giá trị Kp thay đổi.

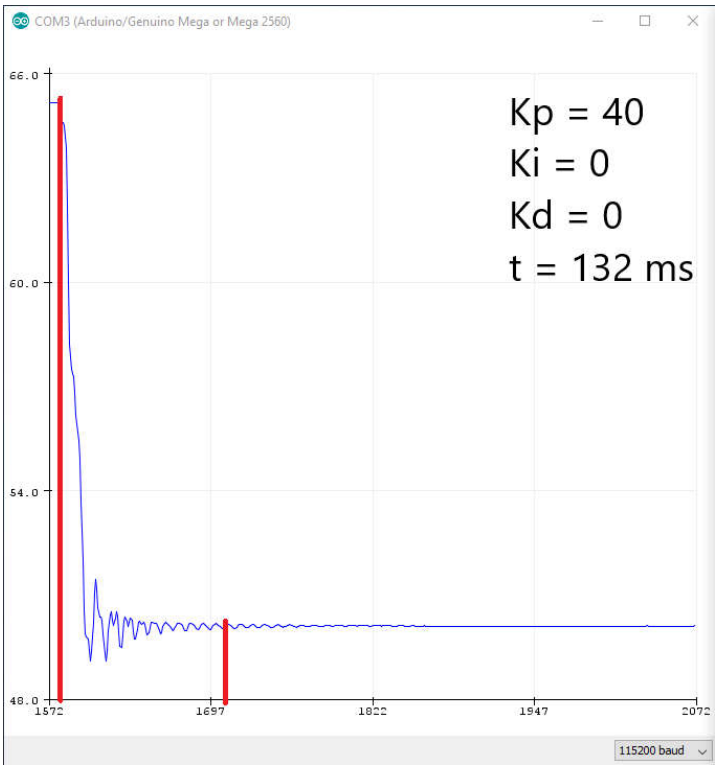
Trước tiên, cũng giống như khi sử dụng phương pháp Ziegler – Nichols hay phương pháp Cohen – Coon, giá trị của tham số tỷ lệ Kp được thay đổi lần lượt Kp = 20, 30, 40, 50, 60 trong khi các giá trị của tham số tích phân Ki và vi phân Kd được đặt bằng 0.



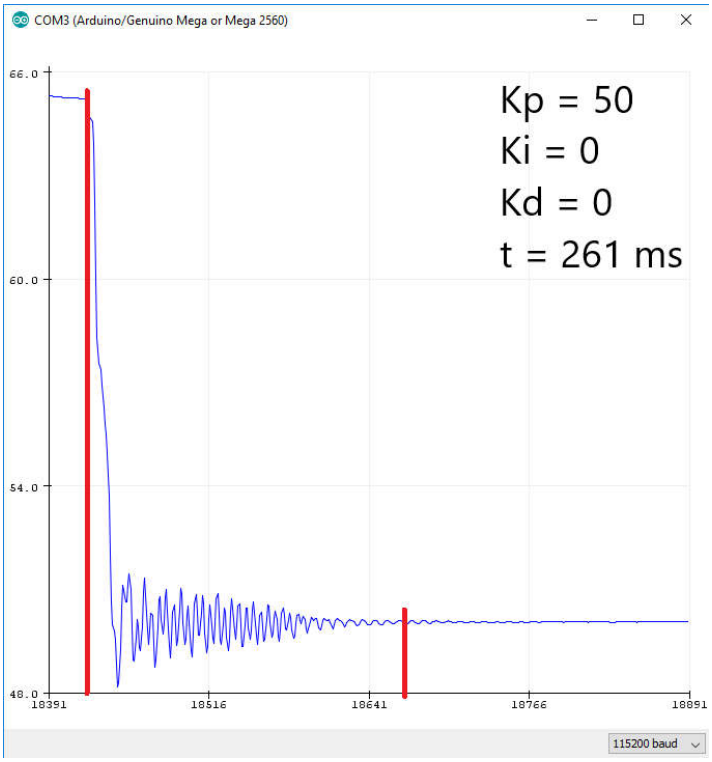
Hình 4.6. Thử nghiệm với giá trị Kp = 20, Ki = 0, Kd = 0



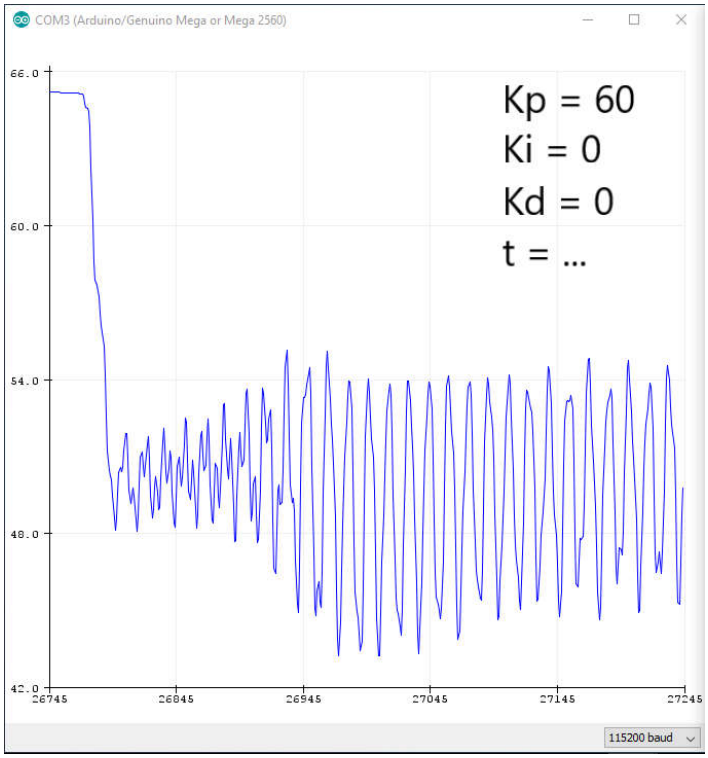
Hình 4.7. Thử nghiệm với giá trị Kp = 30, Ki = 0, Kd = 0



Hình 4.8. Thử nghiệm với giá trị Kp = 40, Ki = 0, Kd = 0



Hình 4.9. Thử nghiệm với giá trị Kp = 50, Ki = 0, Kd = 0



Hình 4.10. Thử nghiệm với giá trị Kp = 60, Ki = 0, Kd = 0

Từ các kết quả thử nghiệm trên có thể rút ra được một số nhận xét như sau:

Trong một khoảng giá trị nhất định (trong thử nghiệm là từ Kp = 0 đến Kp = 30) thì khi giá trị Kp tăng lên, hệ thống sẽ có khoảng thời gian đáp ứng (đạt được 95% giá trị đặt

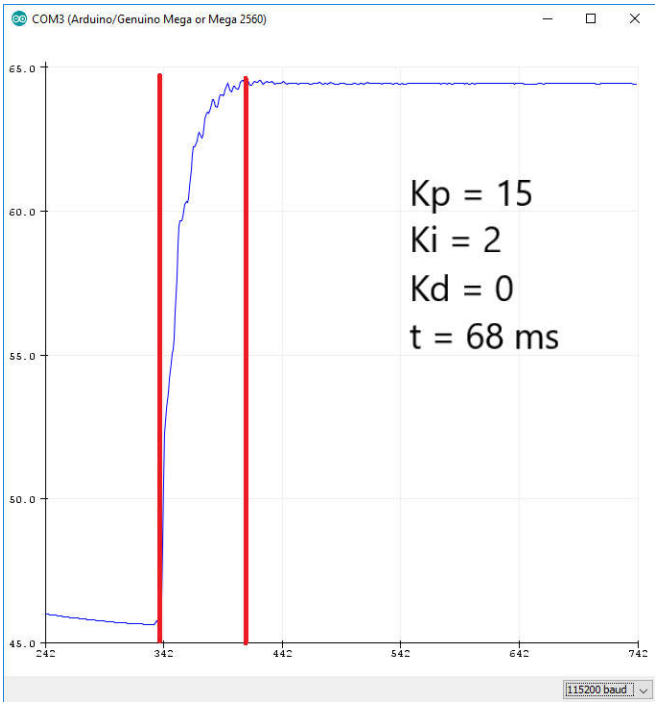
trước) và khoảng thời gian để hệ thống đạt ổn định (dao động nhỏ hơn 0,5 độ) là nhanh hơn. Cụ thể, khi Kp = 20 là t = 94 ms đã giảm xuống còn t = 85 ms khi Kp = 30.

Khi Kp lớn hơn một giá trị nhất định (trong thử nghiệm là Kp > 35) thì khi giá trị Kp tăng lên, hệ thống không thể tăng thời gian đáp ứng hơn được nữa. Điều này có vẻ như trái với lý thuyết về bộ điều khiển PID được trình bày trước đó. Nhưng khi kiểm chứng lại bằng cách hiển thị thêm giá trị của xung PWM mà vi điều khiển tính toán được thì khoảng thời gian đạt đến giá trị tối đa (255) là quá nhỏ; lúc đó động cơ đã hoạt động với công suất tối đa nên việc tăng Kp không còn nhiều ý nghĩa. Đương nhiên, việc thay đổi giá trị Kp vẫn có tác dụng như trong trường hợp giá trị ban đầu và giá trị cần đạt được chênh lệch đáng kể hoặc tốc độ tối đa của động cơ bị thay đổi hoặc khi kết hợp thêm các giá trị Ki và Kd.

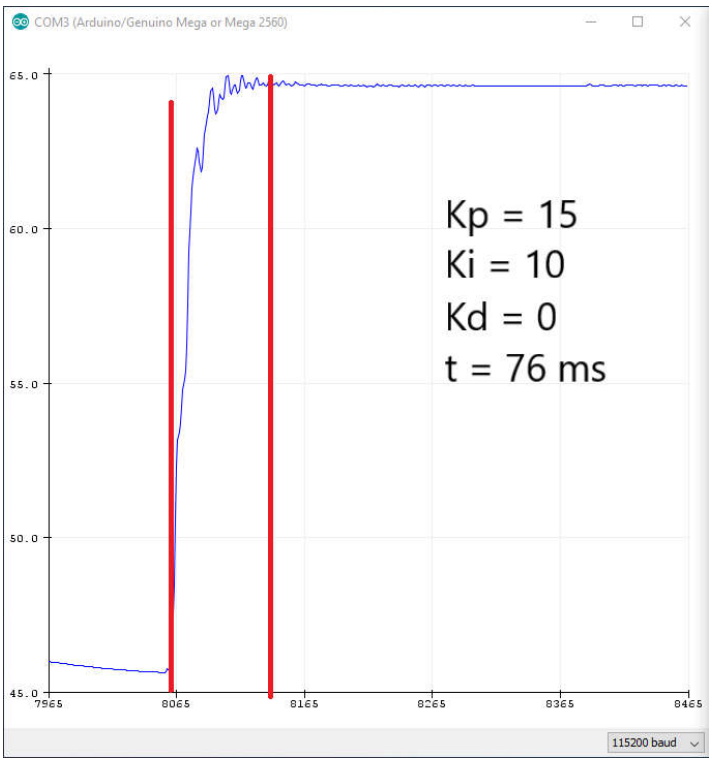
Thời gian để hệ thống đạt được ổn định cũng tăng lên đáng kể khi Kp tăng lên. Với Kp = 40, khi hệ thống đã gần đạt được giá trị đặt trước nhưng vẫn tiếp tục dao động quanh điểm đó một khoảng thời gian trước khi ổn định hẳn (t = 132 ms). Với Kp = 50, khoảng thời gian để hệ thống ổn định tăng lên đến t = 261 ms, gấp 2 lần so với Kp = 40; đồ thị cũng cho thấy sự vọt lố khi hệ thống gần chạm mức 48^o (giá trị đặt trước là 50^o). Các dấu hiệu này tăng lên rõ rệt khi Kp = 60, hệ thống không thể đạt được giá trị đặt trước nữa mà dao động quanh khoảng 50±4^o.

- Thử nghiệm bộ thông số PID với giá trị Ki thay đổi.

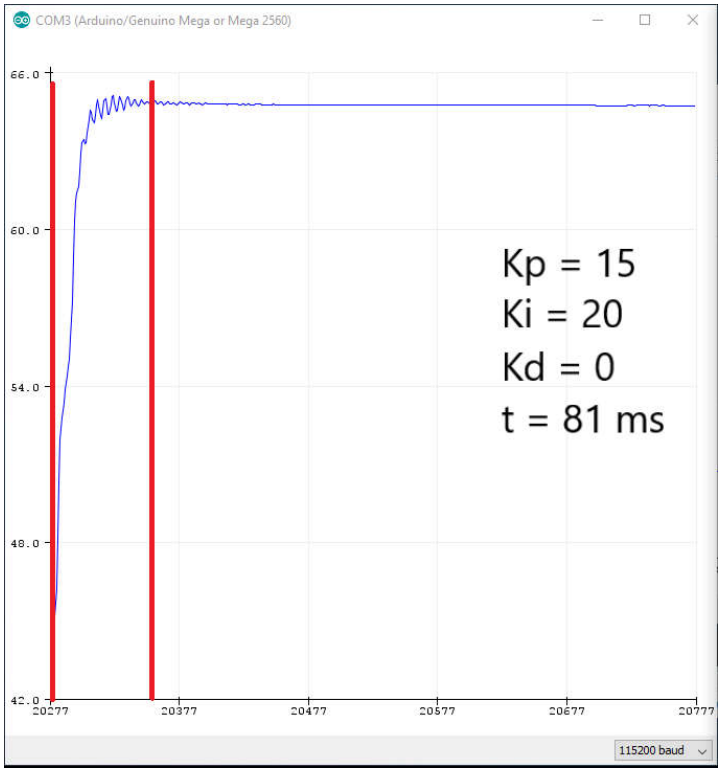
Tương tự như thử nghiệm trước đó, thử nghiệm này được tiến hành bằng cách thay đổi giá trị của tham số Ki để quan sát đáp ứng của hệ thống.



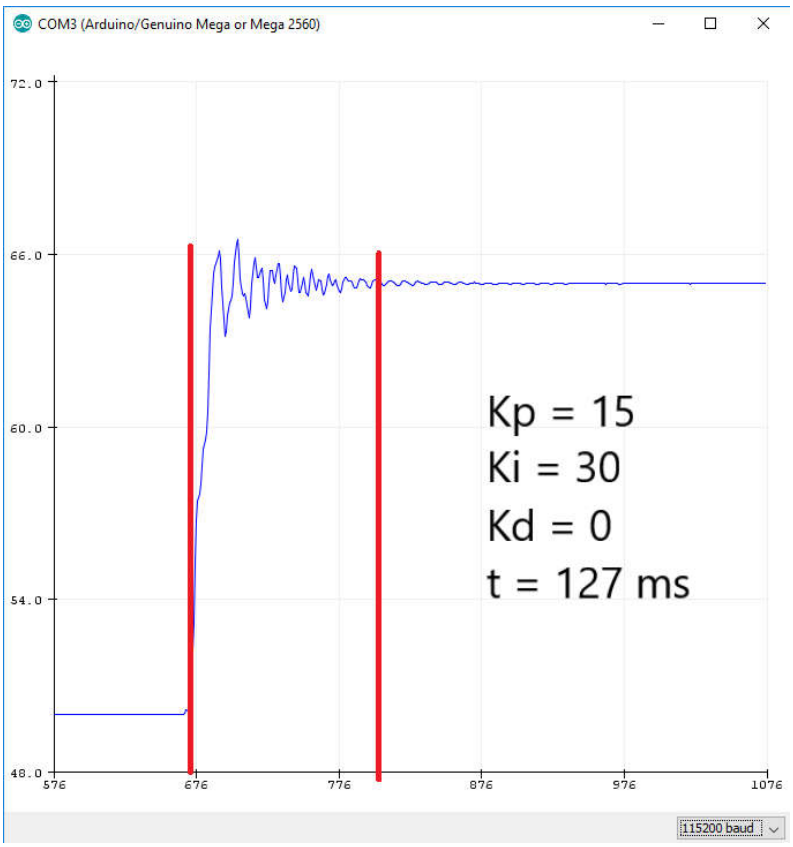
Hình 4.11. Thử nghiệm với giá trị Kp = 15, Ki = 2, Kd = 0



Hình 4.12. Thử nghiệm với giá trị Kp = 15, Ki = 10, Kd = 0



Hình 4.13. Thử nghiệm với giá trị Kp = 15, Ki = 20, Kd = 0



Hình 4.14. Thử nghiệm với giá trị Kp = 15, Ki = 30, Kd = 0

Khi thay đổi giá trị Ki lần lượt từ Ki = 2, 10, 20, 30; sự thay đổi rõ ràng nhất đó là thời gian ổn định của hệ thống. Tương ứng với Ki = 2, thời gian ổn định của hệ thống là 68 ms và tăng dần lên t = 76, 81, 127 khi Ki = 10, 20 , 30.

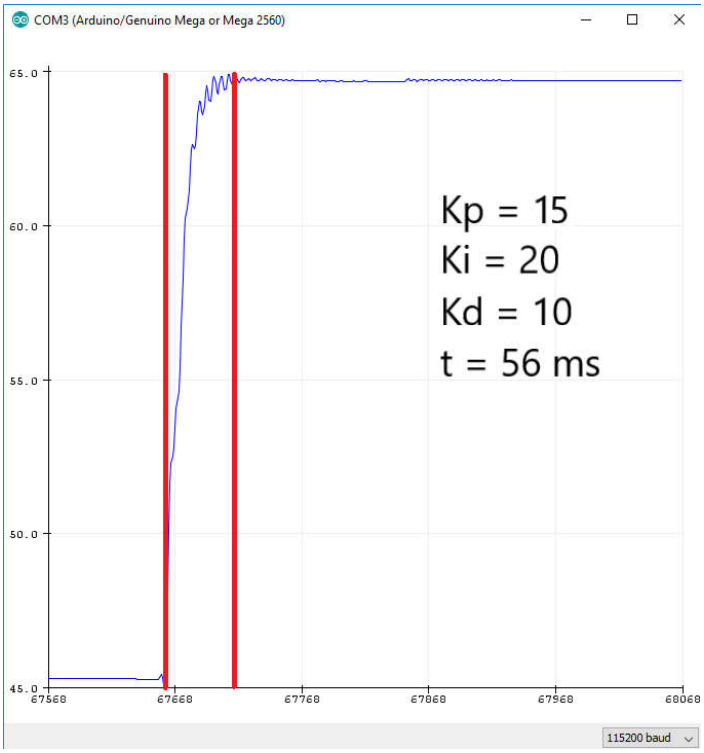
Điểm nhận thấy thứ hai khi tăng giá trị Ki là thời gian đáp ứng của hệ thống (đạt được 95% giá trị đặt trước). Trong trường hợp Ki = 2, thời gian đáp ứng và thời gian ổn định của hệ thống gần như bằng nhau. Khi tăng Ki = 10 thời gian đáp ứng giảm xuống chỉ còn khoảng 50 ms và với Ki = 30 thì thời gian đáp ứng đạt giá trị là 22 ms.

Tăng đồng thời cùng với Ki là độ vọt lố của hệ thống. Trong trường hợp Ki = 2, hệ thống gần như không xảy ra vọt lố và số lần dao động quanh điểm giá trị đặt trước cũng rất ít. Độ vọt lố với Ki = 10 và Ki = 20 là nhỏ, chỉ khoảng 0,5 độ nhưng với Ki = 30 thì lên đến khoảng 1,5 độ.

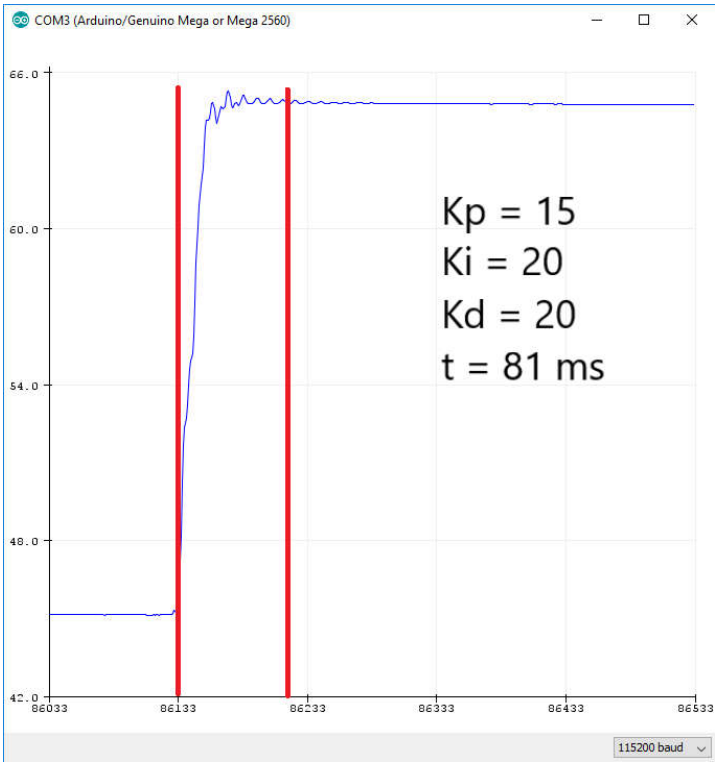
Để đảm bảo hệ thống hoạt động tốt nhất, thông thường sẽ phải thỏa hiệp giữa các yếu tố thời gian đáp ứng, thời gian ổn định và độ vọt lố. Với mỗi hệ thống khác nhau thì việc ưu tiên yếu tố nào đó là phụ thuộc vào nhu cầu của bài toán đặt ra. Trong đề tài này, kết quả của thử nghiệm với Kp = 15, Ki = 20 và Kd = 0 là có thể chấp nhận được, khi đảm bảo cần bằng được các yếu tố nêu trên. Tham số Kd có thể thay đổi để đạt được kết quả đáp ứng của hệ thống tốt hơn với độ vọt lố giảm đi và giảm số lần dao động (hay nói cách khác là thời gian đáp ứng giảm đi).

- Thử nghiệm bộ thông số PID với giá trị Kd thay đổi.

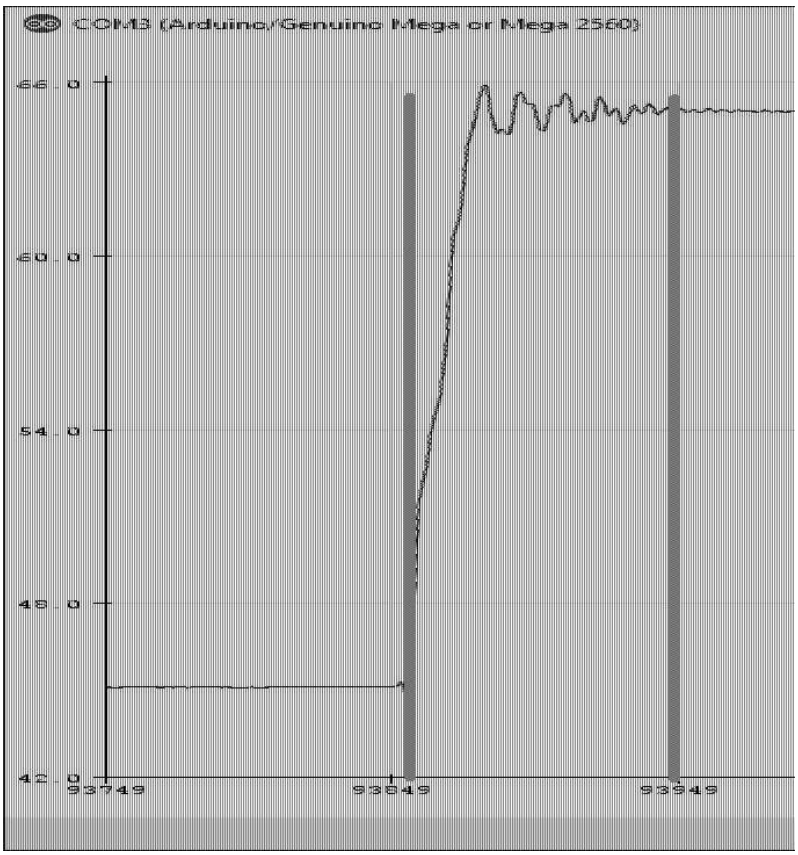
Trong ba tham số của bộ thông số PID thì yếu tố vi phân Kd là yếu tố khó nắm bắt nhất khi thay đổi. Các kết quả dưới đây sẽ thể hiện sự đáp ứng của hệ thông khi tăng dần giá trị Kd lần lượt Kd = 10, 20, 30.



Hình 4.15. Thử nghiệm với giá trị Kp = 15, Ki = 20, Kd = 10



Hình 4.16. Thử nghiệm với giá trị Kp = 15, Ki = 20, Kd = 10



Hình 4.17. Thử nghiệm với giá trị Kp = 15, Ki = 20, Kd = 30

Với kết quả thu được khi thay đổi Kd lần lượt là Kd = 10, 20, 30; thời gian ổn định của hệ thống cũng thay đổi tăng lên từ t = 56 ms khi Kd = 10 đến t = 95 ms khi Kd = 30. Đồng thời với sự thay đổi của thời gian ổn định là thời gian đáp ứng cũng giảm nhưng với lượng không lớn chỉ khoảng 5 – 10 ms. Độ vọt lố và dao động quanh điểm đặt trước cũng tăng lên rõ ràng khi Kd = 30.

Như đã trình bày ở các phần trên, việc thử nghiệm bằng tay rất mất thời gian và yêu cầu kinh nghiệm đối với người sử dụng. Đây cũng là một vấn đề thách thức trong quá trình thực hiện luận văn, qua đó cũng tạo được những phương hướng nghiên cứu trong tương lai, sau khi kết thúc đề tài của tác giả về việc chuyển đổi qua sử dụng động cơ servo với các bộ điều khiển tích hợp sẵn PID hoặc nghiên cứu cách tạo ra các chương trình dò và thử bộ tham số PID (PID autotuning).

4.2.2. *Bộ lọc số*

Tín hiệu thu được từ cảm biến chuyển động MPU 6050 chịu nhiều ảnh hưởng từ bên ngoài như trường điện từ (khi động cơ DC làm việc), nhiễu do rung động của cơ hệ, nhiễu tác động lên đường dây tín hiệu I2C… Do đó việc áp dụng các phương pháp chống nhiễu là cần thiết để đảm bảo kết quả thu được đáng tin cậy trước khi xử lý tính toán.

Các phương pháp chống nhiễu được áp dụng trong đề tài: sử dụng hộp Faraday cho cảm biến chuyển động, sử dụng các đường dây tín hiệu có bọc kim và được nối đất đầy

đủ, sử dụng chống rung cơ học bằng các vòng cao su cho cảm biến và sử dụng các bộ lọc số (Kalman và Complementary) để xử lý tín hiệu.

Mục này sẽ tập trung giới thiệu và mô tả các bộ lọc số được áp dụng trong đề tài.

4.2.2.a. Bộ lọc số Kalman

Bộ lọc Kalman, do Rudolf (Rudy) E. Kálmán công bố năm 1960. Đây là thuật toán dùng chuỗi các giá trị đo đạc, và các giá trị này có thể bị ảnh hưởng bởi nhiễu và sai số, để tiên đoán biến số, qua đó giúp tăng độ chính xác so với việc chỉ sử dụng trực tiếp kết quả đo được. Bộ lọc Kalman cung cấp một phương pháp tính toán đệ quy đối với chuỗi các giá trị đầu vào bị nhiễu, để tối ưu hóa ước lượng trạng thái của một quá trình.

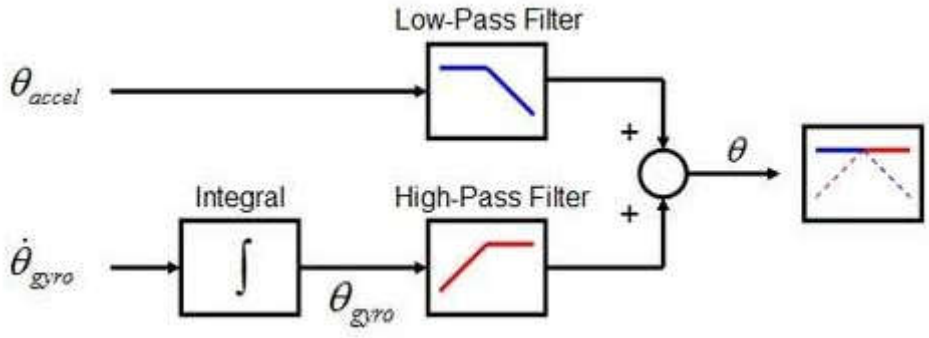
Ngày nay, bộ lọc Kalman được ứng dụng rộng rãi trong nhiều lĩnh vực, các ứng dụng chủ yếu là định hướng, định vị và điều khiển các phương tiện di chuyển. Ngoài ra, để phân tích các dữ liệu để xử lý tín hiệu và phân tích kinh tế, người ta cũng có thể sử dụng bộ lọc Kalman như một lựa chọn hữu ích.

4.2.2.b. Bộ lọc số Complementary

Bộ lọc số Complementary hay còn được biết đến với tên gọi là bộ lọc bù. Đây là một bộ lọc được áp dụng chủ yếu trong việc đọc và xử lý tín hiệu từ cảm biến chuyển động. Bộ lọc bù sử dụng trong hệ thống robot 4 bậc tự do mô phỏng chuyển động trên tàu thủy là sự kết hợp của hai bộ lọc: bộ lọc thông thấp và bộ lọc thông cao.

Nguyên nhân dẫn đến sự kết hợp của hai bộ lọc thông thấp và thông cao bởi vì bản chất của cảm biến chuyển động MPU 6050 sẽ trả lại kết quả của cảm biến gia tốc và cảm biến con quay hồi chuyển. Giá trị của cảm biến gia tốc rất dễ bị ảnh hưởng, đặc biệt là khi hệ cơ học bị rung động hoặc tần số thay đổi vị trí cao, các giá trị trả về sẽ bị nhiễu rất lớn. Nhưng ưu điểm của cảm biến gia tốc đó là góc tính toán từ giá trị cảm biến không bị thay đổi nếu giữ nguyên trạng thái của hệ thống, hiểu theo cách đơn giản là giá trị không bị "trôi". Cảm biến con quay hồi chuyển thì ngược lại, hoạt động ổn định hơn, ít bị tác động bởi rung động hoặc thay đổi liên tục nhưng lại có nhược điểm là khi giữ nguyên trạng thái của hệ thống, kết quả từ cảm biến bị "trôi".

Bộ lọc bù sẽ kết hợp hai tính chất của hai bộ cảm biến để đưa ra giá trị tối ưu, gần với kết quả thực tế nhất và loại bỏ được phần lớn nhiễu hệ thống.



Hình 4.18. Sơ đồ bộ lọc bù [8]

Dữ liệu từ con quay hồi chuyển được tích phân liên tục để tính toán ra giá trị góc, sau đó chúng được kết hợp với giá trị sau khi lọc thông thấp của cảm biến gia tốc để tạo ra một giá trị góc ổn định và chính xác theo thời gian.

Công thức sử dụng để tính toán giá trị góc bằng cách sử dụng bộ lọc bù như sau:

Góc đã lọc bù = α×**(Góc con quay hồi chuyển)** + (1 − α)×**(Góc cảm biến gia tốc)**

Góc con quay hồi chuyển = Góc đã lọc bù trước đó + ω×Δt

Trong đó:

α = τ/(τ + Δt) với Δt = thời gian lấy mẫu

τ = hằng số thời gian, phải lớn hơn nhiễu thông thường của cảm biến gia tốc

Qua thử nghiệm thực tế thì giá trị α = 0.96 ~ 0.98 là hợp lý.

Chương 5

KẾT QUẢ THỰC TẾ VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

5.1. KẾT QUẢ THỬ NGHIỆM THỰC TẾ

Sau khi đã tiến hành thử nghiệm để lựa chọn ra bộ thông số PID phù hợp cho từng động cơ (hay từng cơ cấu chuyển động) như trình bày trước đó; các thử nghiệm tiếp theo được thực hiện để kiểm tra sự phối hợp đồng bộ giữa các cơ cấu trong hệ thống. Qua đó giúp đánh giá được tính chính xác, sự ổn định của mô hình robot trong khi hoạt động cũng như giúp tìm ra được nguyên nhân và cách khắc phục các sai sót xảy ra trong quá trình tính toán xây dựng hệ thống.

5.1.1. *Thử nghiệm tốc độ xử lý của vi điều khiển*

Việc đồng thời phải tính toán cho bộ điều khiển PID của bốn động cơ và tính toán chuyển đổi giữa các giá trị góc (góc nghiêng, góc ngẩng và góc cuộn) sang giá trị bước dịch chuyển (được lấy từ encoder) làm tốc độ xử lý, khả năng đồng bộ của hệ thống giảm đi đáng kể. Để kiểm tra tốc độ xử lý của vi điều khiển, một thử nghiệm nhỏ được tiến hành bằng cách ghi lại số lần thực hiện hoàn tất một vòng điều khiển (bao gồm thu thập dữ liệu, xử lý và đưa ra tín hiệu điều khiển) trong 1 giây. Bảng sau đây sẽ thể hiện kết quả của thử nghiệm.

Nội dung thử nghiệm	Tốc độ xử lý
Đọc và hiển thị kết quả của các cảm biến	141 lần/giây
Đọc, hiển thị kết quả của các cảm biến và tính toán điều khiển PID cho bốn động cơ	35 lần/giây
Đọc, hiển thị kết quả của các cảm biến, chuyển đổi giá trị góc sang giá trị bước dịch chuyển và tính toán điều khiển PID cho bốn động cơ	23 lần/giây

Bảng 5.1. Thử nghiệm tốc độ xử lý của vi điều khiển

5.1.2. *Thử nghiệm giá trị bước dịch chuyển*

Do giới hạn xử lý của vi điều khiển, các giá trị đầu vào cung cấp cho bộ điều khiển là giá trị bước dịch chuyển. Giá trị bước dịch chuyển là giá trị được đọc từ encoder gắn với bốn động cơ; tương ứng với ba động cơ tuyến tính để tạo ra chuyển động nghiêng, ngẩng (mô phỏng lại góc nghiêng và góc ngẩng) và một động cơ tạo ra chuyển động quay (mô phỏng lại góc quay). Encoder gắn với các động cơ đều sử dụng chung một loại và có độ phân giải 334 xung/vòng; nhưng thông qua cơ cấu bánh răng, đai răng với tỉ lệ truyền

khác nhau dẫn đến giá trị của mỗi bước dịch chuyển mà encoder ghi lại được cũng khác nhau. Bảng dưới đây sẽ thể hiện giá trị dịch chuyển thực tế của mỗi động cơ.

Động cơ	Số bước dịch chuyển	Độ dịch chuyển thực tế
Động cơ tuyến tính	1	0,002 mm
Động cơ quay	1	0,09^{o} (độ)

Bảng 5.2. Giá trị bước dịch chuyển của các động cơ

5.1.3. *Thử nghiệm bám vị trí của động cơ*

Mục đích của thử nghiệm là kiểm tra tính chính xác và độ ổn định của bộ điều khiển PID với mỗi động cơ. Kiểm tra xem sự khai khác giữa việc xây dựng mô hình và kết quả thực tế. Phương thức tiến hành thử nghiệm là nhập vào vị trí mà mỗi động cơ phải dịch chuyển đến, ghi lại quá trình động cơ dịch chuyển với tần số lấy mẫu 100Hz và so sánh vị trí kết thúc (khi động cơ hoàn tất dịch chuyển) với vị trí đặt trước. Việc so sánh và kiểm tra sẽ được đánh giá bằng hai tiêu chí đó là: độ sai lệch và tỉ lệ sai lệch. Độ sai lệch được tính bằng giá trị tuyệt đối của hiệu giữa vị trí đặt trước và vị trí kết thúc. Tỉ lệ sai lệch được tính bằng độ sai lệch chia cho tổng quãng đường phải dịch chuyển.

Động cơ	Vị trí ban đầu	Vị trí đặt trước	Vị trí kết thúc	Độ sai lệch (bước)	Tỉ lệ sai lệch (%)
Động cơ tuyến tính 1	22500	40637	40527	110	0,606
Động cơ tuyến tính 2	21406	5040	4844	196	1,198
Động cơ tuyến tính 3	2457	44795	44552	243	0,574
Động cơ quay	524	-583	-593	10	0,903

Bảng 5.3. Thử nghiệm bám vị trí của động cơ – Lần 1

Động cơ	Vị trí ban đầu	Vị trí đặt trước	Vị trí kết thúc	Độ sai lệch (bước)	Tỉ lệ sai lệch (%)
Động cơ tuyến tính 1	11088	8456	8249	207	7,291
Động cơ tuyến tính 2	2415	20435	20424	11	0,061
Động cơ tuyến tính 3	30434	38262	38078	184	2,407
Động cơ quay	-32	417	423	6	1,319

Bảng 5.4. Thử nghiệm bám vị trí của động cơ – Lần 2

Động cơ	Vị trí ban đầu	Vị trí đặt trước	Vị trí kết thúc	Độ sai lệch (bước)	Tỉ lệ sai lệch (%)
Động cơ tuyến tính 1	49144	27989	28139	150	0,714
Động cơ tuyến tính 2	45173	29108	28953	155	0,956
Động cơ tuyến tính 3	44160	47050	46989	61	2,156
Động cơ quay	-312	269	259	10	1,751

Bảng 5.5. Thử nghiệm bám vị trí của động cơ – Lần 3

Động cơ	Vị trí ban đầu	Vị trí đặt trước	Vị trí kết thúc	Độ sai lệch (bước)	Tỉ lệ sai lệch (%)
Động cơ tuyến tính 1	18406	38023	37798	225	1,160
Động cơ tuyến tính 2	46008	28462	28339	123	0,696
Động cơ tuyến tính 3	47719	37764	37726	38	0,380
Động cơ quay	553	539	542	3	27,273

Bảng 5.6. Thử nghiệm bám vị trí của động cơ – Lần 4

Động cơ	Vị trí ban đầu	Vị trí đặt trước	Vị trí kết thúc	Độ sai lệch (bước)	Tỉ lệ sai lệch (%)
Động cơ tuyến tính 1	44495	11455	11639	184	0,560
Động cơ tuyến tính 2	25561	39826	39889	63	0,440
Động cơ tuyến tính 3	31409	2319	2187	132	0,452
Động cơ quay	15	337	345	8	2,424

Bảng 5.7. Thử nghiệm bám vị trí của động cơ – Lần 5

Qua 5 lần đánh giá thử nghiệm, một số kết quả về hệ thống được tổng kết như sau:

- Xét chung đối với ba động cơ tuyến tính thì giá trị sai lệch lớn nhất khi dịch chuyển là $\Delta max = 225$ và nhỏ nhất là $\Delta min = 11$. Nếu như xét theo giá trị thực tế thì độ sai lệch lớn nhất là $\Delta max = 225$ tương đương với $\Delta d = 0,45$ mm (tổng độ dài hành trình của một động cơ tuyến tính sử dụng trong đề tài luận văn này lên đến D = 300 mm). Tỉ lệ sai lệch lớn nhất tính trên quãng đường đã dịch chuyển là 2,407% và nhỏ nhất là 0,061%.
- Đối với động cơ quay, giá trị sai lệch lớn nhất khi dịch chuyển là $\Delta max = 10$ và nhỏ nhất là $\Delta min = 3$. Tỉ lệ sai lệch lớn nhất tính trên quãng đường đã

dịch chuyển chuyển là 27,273% và nhỏ nhất là 0,903%. Con số 27,273% tuy lớn nhưng nếu xét theo giá trị thực tế thì độ sai lệch lớn nhất $\Delta max = 10$ tương đương với độ sai lệch góc tối đa là $0,9^o$ (độ) (cơ cấu quay trong đề tài luận văn này có thể quay được một vòng tròn 360^o).

Nhưng kết quả thu được từ quá trình thử nghiệm được đánh giá là tương đối khả quan, đáp ứng được mục đích mô phỏng những chuyển động trên tàu biển (chuyển động ngẫu nhiên do tác động của sóng biển hay chủ đích khi thay đổi hướng tàu) tuy nhiên nó cũng cho thấy một số vấn đề đang tồn tại trong hệ thống; điển hình như những điểm sau đây:

- Vấn đề trong thiết kế cơ khí: các kết nối, liên động giữa các cơ cấu cơ khí còn chưa tốt. Các rung lắc ngoài mong muốn khi vận hành cũng ảnh hưởng đến kết quả đo đạc. Với những mô phỏng chuyển động không đòi hỏi độ chính xác cao thì hiện tại hệ thống có thể đáp ứng được nhưng cần phải sửa đổi về kết cấu để áp dụng được với những ứng dụng có tính chính xác cao (y tế, gia công vật liệu...). Việc tính toán động học thuận và động học ngược chưa hoàn thiện dẫn đến không thể nhập trực tiếp được các giá trị góc mong muốn để điều khiển hệ thống mà phải thông qua các giá trị bước dịch chuyển trực tiếp của từng động cơ.
- Vấn đề trong thiết kế điện tử: nổi bật là vấn đề lựa chọn động cơ tuyến tính sử dụng, do không có phanh điện nên khi chạy có tải (lắp đặt hệ thống antena lên trên) thì vấn đề quán tính cũng làm vị trí của các động cơ bị trượt khỏi giá trị đặt trước. Các encoder được truyền động thông qua dây dai, bánh răng nên không tránh khỏi việc bị căng hoặc chùng làm ảnh hưởng đến giá trị thu được. Vi điều khiển có tốc độ còn thấp khi phải tính toán xử lý nhiều phép toán dấu phẩy động hay chuyển đổi hệ quy chiếu.
- Vấn đề trong thiết kế chương trình điều khiển: do bộ thông số PID được tìm bằng phương pháp thủ công nên vẫn còn hạn chế. Một số giới hạn mềm được đặt trong chương trình điều khiển để tránh việc động cơ hoạt động liên tục khi cố gắng đạt được vị trí chính xác trong dải sai số nhỏ và rất nhỏ. Điều này cũng có thể thấy rõ ở kết quả khi giá trị cuối cùng không bao giờ bằng được giá trị đặt trước. Các bộ lọc số mới chỉ được áp dụng, chưa nghiên cứu và tìm hiểu sâu để tối ưu các kết quả do cảm biến thu được.

5.2. PHƯƠNG HƯỚNG PHÁT TRIỂN

Với những vấn đề đang tồn tại với hệ thống, phương hướng phát triển hoàn thiện đề tài cũng được phân chia theo ba phần: cơ khí, điện tử và điều khiển.

5.2.1. *Đối với thiết kế cơ khí*

Hướng phát triển đầu tiên là đầu tư nghiên cứu hoàn thiện các phương trình động học thuận và động học ngược. Từ đó có thể nhập trực tiếp giá trị góc mong muốn để điều khiển hệ thống, thông qua đó mở ra các ứng dụng như: đặt một hệ cảm biến góc trên các phương tiện cần mô phỏng để thu thập dữ liệu sau đó truyền lại cho hệ thống robot mô phỏng lại trong phòng thí nghiệm hoặc sử dụng cảm biến góc, điện thoại thông minh để tương tác trực tiếp với hệ thống robot.

Nâng cấp các kết cấu cơ khí, đảm bảo chịu tải và hoạt động ổn định khi tiến hành mô phỏng. Có thể phát triển những phiên bản lớn hơn để mô phỏng chuyển động của buồng lái ô-tô, máy bay hay các phương tiện khác. Phát triển những phiên bản sử dụng thủy lực hoặc khí nén để tăng tải trọng, có thể áp dụng làm bộ gá gia công vật liệu hoặc làm bộ chống rung chủ động cho các máy móc, cầu đường, thiết bị tải trọng lớn.

Phát triển theo hướng robot song song 6 bậc tự do (hexapod) để có thể ứng dụng trong nhiều lĩnh vực hơn như y tế, sơn phủ, dùng trong các phòng thí nghiệm công nghệ cao...

5.2.2. *Đối với thiết kế điện tử*

Động cơ sử dụng hiện tại đang là động cơ DC, hoàn toàn có thể nâng cấp sử dụng sang động cơ DC servo tích hợp sẵn bộ điều khiển PID giúp tăng độ chính xác để sử dụng trong các ứng dụng đòi hỏi độ chính xác cao.

Vi điều khiển và các bo mạch điều khiển có thể nâng cấp lên để có tốc độ xử lý nhanh hơn, làm việc tốt hơn khi tính toán với dấu phẩy động.

Tích hợp thêm màn hình hoặc các hệ thống điện tử khác để mô phỏng các phương tiện một cách chân thực hơn.

5.2.3. *Đối với thiết kế chương trình điều khiển*

Tối ưu hóa bộ thông số PID, có thể phát triển các bộ tự dò và thử tham số Kp, Ki, Kd (PID autotuning). Hoàn thiện phần mềm điều khiển để xử lý các giá trị góc, thực hiện nhiều tác vụ, thêm các chức năng bổ sung cũng như tính toán các trường hợp có thể xảy ra với hệ thống để có biện pháp xử lý.

Chương trình điều khiển có thể mở rộng để kết nối thêm với nhiều thiết bị ngoại vi, tạo nên một hệ mô phỏng chân thực và nhiều tính năng hơn như kết nối với điện thoại thông minh, tương tác với các ứng dụng khoa học và giải trí trên máy tính, điện thoại.

Nghiên cứu thêm về các bộ lọc số, tối ưu các bộ lọc số để có thể ứng dụng tốt nhất cho hệ thống robot cũng như các ứng dụng khác.

KẾT LUẬN

Qua quá trình nghiên cứu, thiết kế chế tạo robot 04 bậc tự do mô phỏng chuyển động trên tàu thủy, luận văn đã thực hiện được một số công việc như sau:

- Nghiên cứu được tổng quan lý thuyết về robot, một số loại robot có trên thị trường, các khái niệm, các cơ cấu, nguyên lý hoạt động và cụ thể hơn là loại robot song song 3 bậc tự do. Từ cơ sở trên để thiết kế ra hệ thống robot 4 bậc tự do bằng cách kết hợp thêm một mâm xoay để đảm bảo yêu cầu đặt ra của đề tài.
- Tìm hiểu phần mềm thiết kế và mô phỏng 3D solidworks 2017 và hoàn thành việc thiết kế, mô phỏng và thử nghiệm trên phần mềm. Từ đó đưa ra các thiết kế chính xác, tối ưu hơn khi bắt tay vào thiết kế thực tế.
- Tính toán, lựa chọn được các thiết bị, cơ cấu, mạch điện điều khiển sử dụng trong hệ thống. Tìm hiểu nguyên lý hoạt động của các mô-đun cảm biến, encoder, các sử dụng bộ KIT Arduino Mega 2560…
- Làm quen, sử dụng Arduino IDE để lập trình điều khiển cho hệ thống. Tìm hiểu các thuật toán điều khiển và đặc biệt là thuật toán PID để tính toán vị trí của các cơ cấu. Tìm hiểu các bộ lọc số, áp dụng bộ lọc Kalman và bộ lọc Complementary (bộ lọc bù) để đọc tín hiệu từ cảm biến chuyển động.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]. Nguyễn Quốc Anh (2013), *Phân tích động học và mô phỏng đồ họa robot song song không gian 3RPS*, Đồ án tốt nghiệp Khoa cơ khí, Trường Đại học Bách khoa Hà Nội.

[2]. Lê Quang Đức, Phạm Tấn Tín (2012), *Robot 4 bậc tự do*, Đồ án Đo lường và điều khiển bằng máy tính, Trường Đại học Công nghệ TP. Hồ Chí Minh.

[3]. Trần Thị Thanh Hải (2007), *Mô hình hóa và mô phỏng robot song song loại hexapod*, Luận văn Thạc sĩ Tự động hóa, Trường Đại học Kỹ thuật Công nghiệp Thái Nguyên.

[4]. Lưu Văn Quyền (2013), *Sử dụng bộ lọc kalman trong bài toán bám mục tiêu*, Luận văn Thạc sĩ Kỹ thuật Viễn thông, Trường Học viện Công nghệ Bưu chính Viễn thông.

Tiếng Anh

[5]. Robert J. Twiss, Eldridge M. Moores (1992). *Structural geology*, W. H. Freeman, pp 11.

[6]. Gregory G. Slabaugh (1999), "Computing Euler angles from a rotation matrix", http://thomasbeatty.com.

[7]. James Diebel (2006), *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*, Stanford University, California.

[8]. Vladimir Kubelka, Michal Reinstein (2012), "Complementary filtering approach to orientation estimation using inertial sensors only", *Robotics and Automation (ICRA), 2012 IEEE International Conference.*

[9]. Antonio Visioli (2006), *Practical PID Control*, Springer, Lodon.

[10]. Arthur Author, Joe Author (1942), "Optimum settings for automatic controllers", Trans. ASME, vol. 64, pp. 759-768.

[11]. A. Datta, M. Ho, S.P. Bhattacharyya (2000), *Structure and synthesis of PID Controllers*, Springer-Verlag, Berlin, Germany.

[12]. J.I. Rego L.H. Keel, S.P. Bhattacharyya (2003), "A new approach to digital PID controller design", *IEEE Transactions on Automatic Control*, vol. 64, pp. 687-692.

[13]. Kalman R.E (1960), "A New Approach to Linear Filtering and Prediction Problems", Journal of Basic Engineering, vol 82.

[14]. Greg Welch, Gary Bishop (2001), *An Introduction to the Kalman Filter*, University of North Carolina, Chapel Hill.

[15]. Mohinder S. Grewal (2014), *Kalman filtering*, Springer, Lodon.

PHỤ LỤC

Dưới đây là chương trình điều khiển của hệ thống robot 4 bậc tự do mô phỏng chuyển động trên tàu thủy. Chương trình được viết bằng ngôn ngữ lập trình C, trên phần mềm Arduino IDE và được sử dụng trên bộ KIT Arduino MEGA 2560.

```
// ================================================== //
// Author: DANG VAN MUOI
// Project: 4 DoF robot for simulating motion on boat
// Ver: 1.1.12
// Date: 24.09.2017
// ================================================== //
#include <Encoder.h>
#include "Wire.h"

//=========================================
//define motor output pin
#define MOT_1_FWD     46
#define MOT_1_BCK     47
#define MOT_2_FWD     48
#define MOT_2_BCK     49
#define MOT_3_FWD     50
#define MOT_3_BCK     51
#define MOT_4_FWD     52
#define MOT_4_BCK     53
//=========================================

//===========PARAMETERS DECLARE==========
// Change these two numbers to the pins connected to your encoder.
//   Best Performance: both pins have interrupt capability
//   Good Performance: only the first pin has interrupt capability
```

```
//   Low Performance:  neither pin has interrupt capabilitys
Encoder myEnc_1(19, 30);
Encoder myEnc_2(18, 32);
Encoder myEnc_3(3, 34);
Encoder myEnc_4(2, 36);
// 1 mm = 500 pulses
// 1 pulse = 0.002 mm

long oldPosition_1  = -999;
long oldPosition_2  = -999;
long oldPosition_3  = -999;
long oldPosition_4  = -999;
long newPosition_1;
long newPosition_2;
long newPosition_3;
long newPosition_4;
bool checkPrint = false;



//define PID parameters
#define iKp_1         0.19
#define iKi_1         0.00000035
#define iKd_1         0.04
#define iKp_2         0.19
#define iKi_2         0.00000035
#define iKd_2         0.04
#define iKp_3         0.19
#define iKi_3         0.00000035
#define iKd_3         0.04
```

```
#define iKp_4         4.4
#define iKi_4         0.0000001
#define iKd_4         0.0005
//============FUNCTION DECLARE============

bool motorStatus_1 = false;
bool motorStatus_2 = false;
bool motorStatus_3 = false;
bool motorStatus_4 = false;
bool moveStatus = false;
bool conStatus = false;
unsigned long rowIndex = 0;
long row;

//Moving to desired position
bool motorMove(int selectMotor, long refPos, char motorSpeed);
//selectMotor: 1, 2, 3 , 4
//refPos: destination position
//motorSpeed: max speed for motor - from 0~254

bool motorsMove(long refPos_1, long refPos_2, long refPos_3, long refPos_4, int maxSpeed);
//moving motors at the same time

void continuousMove(long conPos[][5]);
//countinuous moving, when motors reach desired position then go to the next ones

//Encoder update function
void encoderUpdate();
```

```
bool motorMove(int selectMotor, long refPos, int motorSpeed)
{
  bool motorStatus;
  long posErr = 0;
  long curPos;
  long oldPos;
  int mypwm;
  float QTTmpCalc = 0.0;
  float QTCaltDPos = 0.0;
  float QTCaltIPos = 0.0;
  float iKp = 0.0;
  float iKi = 0.0;
  float iKd = 0.0;
  int pwmPin;
  int outputPinA;
  int outputPinB;
  int cutOff;

  switch (selectMotor)
  {
    case 1:
      motorStatus = motorStatus_1;
      curPos = newPosition_1;
      oldPos = oldPosition_1;
      outputPinA = MOT_1_FWD;
      outputPinB = MOT_1_BCK;
      iKp = iKp_1;
      iKi = iKi_1;
```

```
  iKd = iKd_1;
  pwmPin = 4;
  cutOff = 40;
  break;
case 2:
  motorStatus = motorStatus_2;
  curPos = newPosition_2;
  oldPos = oldPosition_2;
  outputPinA = MOT_2_FWD;
  outputPinB = MOT_2_BCK;
  iKp = iKp_2;
  iKi = iKi_2;
  iKd = iKd_2;
  pwmPin = 5;
  cutOff = 40;
  break;
case 3:
  motorStatus = motorStatus_3;
  curPos = newPosition_3;
  oldPos = oldPosition_3;
  outputPinA = MOT_3_FWD;
  outputPinB = MOT_3_BCK;
  iKp = iKp_3;
  iKi = iKi_3;
  iKd = iKd_3;
  pwmPin = 6;
  cutOff = 40;
  break;
case 4:
```

```
    motorStatus = motorStatus_4;
    curPos = newPosition_4;
    oldPos = oldPosition_4;
    outputPinA = MOT_4_FWD;
    outputPinB = MOT_4_BCK;
    iKp = iKp_4;
    iKi = iKi_4;
    iKd = iKd_4;
    pwmPin = 7;
    cutOff = 85;
    break;
  default:
    break;
}
if (!motorStatus)
{
  QTTmpCalc = refPos - curPos;
  if ((QTTmpCalc <= 50) && (QTTmpCalc >= -50))
    QTTmpCalc = 0;
  QTCaltDPos = QTTmpCalc - posErr;
  posErr = QTTmpCalc;
  QTCaltIPos += posErr * iKi;

  QTTmpCalc = (posErr * iKp + QTCaltIPos + QTCaltDPos * iKd);
  if (QTTmpCalc < - motorSpeed)
    QTTmpCalc = - motorSpeed;
  if (QTTmpCalc > motorSpeed)
    QTTmpCalc = motorSpeed;
  if ((QTTmpCalc > -cutOff) && (QTTmpCalc < cutOff))
```

```
  QTTmpCalc = 0;
if (QTTmpCalc >= 0)
{
  digitalWrite(outputPinA, true);
  digitalWrite(outputPinB, false);
  mypwm = (int)QTTmpCalc;
}
else
{
  digitalWrite(outputPinA, false);
  digitalWrite(outputPinB, true);
  mypwm = (int)(-(QTTmpCalc));
}
analogWrite(pwmPin, mypwm);
if (!mypwm)
{
  switch (selectMotor)
  {
    case 1:
      motorStatus_1 = true;
      break;
    case 2:
      motorStatus_2 = true;
      break;
    case 3:
      motorStatus_3 = true;
      break;
    case 4:
      motorStatus_4 = true;
```

```
        break;
      default:
        break;
    }
   }
  }
}

bool motorsMove(long refPos_1, long refPos_2, long refPos_3, long refPos_4, int maxSpeed)
{
 if (!moveStatus)
 {
  motorMove(1, refPos_1, maxSpeed);
  motorMove(2, refPos_2, maxSpeed);
  motorMove(3, refPos_3, maxSpeed);
  motorMove(4, refPos_4, 180);
  //motorMove(4, refPos_4, maxSpeed);
  if (motorStatus_1 && motorStatus_2 && motorStatus_3 && motorStatus_4)
  {
   moveStatus = true;
  }
 }
}

void continuousMove(long conPos[][5])
{
```

```
motorsMove(conPos[rowIndex][0],conPos[rowIndex][1],conPos[rowIndex][2],conPos[ro
wIndex][3],conPos[rowIndex][4]);
  if ((moveStatus)&&(rowIndex < row))
  {
    rowIndex++;
    moveStatus = false;
    motorStatus_1 = false;
    motorStatus_2 = false;
    motorStatus_3 = false;
    motorStatus_4 = false;
  }
}

//update encoder position
void encoderUpdate()
{
  newPosition_1 = myEnc_1.read();
  if (newPosition_1 != oldPosition_1)
  {
    oldPosition_1 = newPosition_1;
    checkPrint = true;
  }

  newPosition_2 = myEnc_2.read();
  if (newPosition_2 != oldPosition_2)
  {
    oldPosition_2 = newPosition_2;
    checkPrint = true;
```

```
  }

  newPosition_3 = myEnc_3.read();
  if (newPosition_3 != oldPosition_3)
  {
    oldPosition_3 = newPosition_3;
    checkPrint = true;
  }

  newPosition_4 = myEnc_4.read();
  if (newPosition_4 != oldPosition_4)
  {
    oldPosition_4 = newPosition_4;
    checkPrint = true;
  }

  if (checkPrint)
  {
    Serial.print(newPosition_1);
    Serial.print('\t');
    Serial.print(newPosition_2);
    Serial.print('\t');
    Serial.print(newPosition_3);
    Serial.print('\t');
    Serial.println(newPosition_4);
    checkPrint = false;
  }
}
```

```
void setup() {
  pinMode(MOT_1_FWD, OUTPUT);
  pinMode(MOT_1_BCK, OUTPUT);
  pinMode(MOT_2_FWD, OUTPUT);
  pinMode(MOT_2_BCK, OUTPUT);
  pinMode(MOT_3_FWD, OUTPUT);
  pinMode(MOT_3_BCK, OUTPUT);
  pinMode(MOT_4_FWD, OUTPUT);
  pinMode(MOT_4_BCK, OUTPUT);

  //home position
  digitalWrite(MOT_1_FWD, false);
  digitalWrite(MOT_1_BCK, true);
  analogWrite(4, 150);

  digitalWrite(MOT_2_FWD, false);
  digitalWrite(MOT_2_BCK, true);
  analogWrite(5, 150);

  digitalWrite(MOT_3_FWD, false);
  digitalWrite(MOT_3_BCK, true);
  analogWrite(6, 150);

  delay(5000);

  Serial.begin(115200);
  Serial.println("Basic Encoder Test:");
  myEnc_1.write(0);
  myEnc_2.write(0);
```

```
  myEnc_3.write(0);
  myEnc_4.write(0);
}

//long temp = 00000;
//long tempPos[3] = {20000, 5000, 35000};
//int tempIndex = 0;
void loop()
{

  long testPos[50][5] = {
22690,21450,2496,512,173,
40637,5040,44795,-583,154,
35778,17456,36279,-416,151,
10940,2594,30552,-37,164,
11278,5410,6501,369,172,
8456,20435,38262,417,204,
49141,27436,41429,232,195,
15835,24941,38264,142,148,
29133,29794,9420,-226,176,
4035,46500,20057,-419,164,
29769,13956,37328,505,197,
1419,29991,6718,-228,120,
46508,5103,35824,409,197,
23477,32641,33404,-256,170,
9738,11785,24731,-596,211,
30476,43820,20372,-170,169,
40025,9417,45864,-407,156,
22394,37033,41832,170,182,
```

41975,24458,22883,-86,153,

26637,25764,31063,26,146,

34875,44393,8840,397,186,

1,22626,11489,556,189,

46626,1956,38446,509,219,

42234,39660,15462,-590,155,

37029,25789,2686,145,217,

49093,45219,44343,-307,215,

27989,29108,47050,269,141,

18547,46218,47818,544,171,

38023,28462,37764,539,126,

44505,25899,31507,8,205,

11455,39826,2319,337,218,

20218,4902,9149,-85,154,

19688,20370,31951,-430,123,

23990,8837,35496,-474,198,

7651,2148,34145,289,171,

27901,41968,37002,-359,138,

33661,20071,36086,508,173,

5061,33500,20248,-324,141,

28086,33765,21020,-370,140,

41471,37413,14124,225,126,

455,25717,36154,435,141,

32326,7239,37339,69,145,

28295,10310,1350,-279,132,

17911,7305,15748,129,202,

7090,17313,13243,-144,123,

47649,17450,37780,51,195,

30292,42295,8238,-247,122,

```
49625,19452,34872,356,146,
17526,37458,36632,175,127,
0,0,0,0,120
};

 row = sizeof(testPos)/5;
 encoderUpdate();
 continuousMove(testPos);
 /*
 if ((moveStatus)&&(tempIndex < 3))
 {
  tempIndex++;
  moveStatus = false;
  motorStatus_1 = false;
  motorStatus_2 = false;
  motorStatus_3 = false;
  motorStatus_4 = false;
 }
 */
 /*
 motorMove(1, tempPos[tempIndex], 80);
 if ((motorStatus_1) && (tempIndex < 2))
 {
  tempIndex++;
  motorStatus_1 = false;
 }
 */
 //motorMove(2, 25000, 130);
 //motorMove(4, -4000, 200);
```

```
  //digitalWrite(MOT_4_FWD, true );
  //digitalWrite(MOT_4_BCK, false);
  //analogWrite(7, 200);
}
```