

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CƠ KHÍ CHẾ TẠO MÁY
BỘ MÔN CƠ ĐIỆN TỬ



ĐỒ ÁN CƠ ĐIỆN TỬ

THIẾT KẾ VÀ ĐIỀU KHIỂN ROBOT DI ĐỘNG
ỨNG DỤNG TRONG HÀN TỰ ĐỘNG

GVHD: Nguyễn Minh Triết

SVTH: MSSV:

Đặng Kỳ Anh 20146475

Phan Duy Nhất 20146165

Nguyễn Quang Phúc 20146142

Tp.Hồ Chí Minh, tháng 01 năm 2024

PHIẾU NHẬN XÉT ĐỒ ÁN CƠ ĐIỆN TỬ

(Dành cho giảng viên hướng dẫn)

Sinh viên thực hiện:

- | | |
|----------------------|----------------|
| 1. Đặng Kỳ Anh | MSSV: 20146475 |
| 2. Phan Duy Nhất | MSSV: 20146165 |
| 3. Nguyễn Quang Phúc | MSSV: 20146142 |

Ngành đào tạo: Công nghệ kỹ thuật Cơ Điện Tử

Tên đề tài: Thiết kế và điều khiển robot di động ứng dụng trong hàn tự động

Giảng viên hướng dẫn: Ths. Nguyễn Minh Triết

Ý KIẾN NHẬN XÉT

1. Nhận xét về tinh thần làm việc, thái độ sinh viên

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

2. Nhận xét kết quả thực hiện của đồ án Cơ Điện Tử

2.1. Kết cấu, cách thức trình bày đồ án Cơ Điện Tử.

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

2.2. Nội dung đồ án

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

2.3. Kết quả đạt được

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

2.4. Những tồn tại (nếu có)

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

3. Đánh giá

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

4. Kết luận

- Được phép bảo vệ
- Không được phép bảo vệ

TP Hồ Chí Minh, ngày tháng năm 2024

Giảng viên hướng dẫn

(Ký & ghi rõ họ tên)

PHIẾU NHẬN XÉT ĐỒ ÁN CƠ ĐIỆN TỬ

(Dành cho giảng viên phản biện)

Sinh viên thực hiện:

- | | |
|----------------------|----------------|
| 4. Đặng Kỳ Anh | MSSV: 20146475 |
| 5. Phan Duy Nhất | MSSV: 20146165 |
| 6. Nguyễn Quang Phúc | MSSV: 20146142 |

Ngành đào tạo: Công nghệ kỹ thuật Cơ Điện Tử

Tên đề tài: Thiết kế và điều khiển robot di động ứng dụng trong hàn tự động

Giảng viên phản biện:

Ý KIẾN NHẬN XÉT

1. Kết cấu, cách trình bày đồ án Cơ Điện Tử

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

2. Nội dung đồ án

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

3. Kết quả đạt được

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

4. Những tồn tại (nếu có)

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

5. Câu hỏi

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

6. Đánh giá

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

7. Kết luận

- Được phép bảo vệ
- Không được phép bảo vệ

TP Hồ Chí Minh, ngày tháng năm 2024

Giảng viên phản biện

(Ký & ghi rõ họ tên)

LỜI CÁM ƠN

Trước hết, nhóm chúng em xin được bày tỏ lòng biết ơn sâu sắc tới thầy Nguyễn Minh Triết đã tận tình hướng dẫn, chỉ bảo nhóm chúng em cũng như luôn khích lệ, động viên nhóm để chúng em có thể hoàn thành được đề tài này. Đồng thời, thầy cũng là người đã tạo điều kiện thuận lợi về không gian cho nhóm chúng em trong quá trình thực hiện đồ án này.

Tiếp đến, chúng em xin được cảm ơn quý thầy cô đã và đang công tác tại trường Đại học Sư Phạm Kỹ Thuật thành phố Hồ Chí Minh nói chung và đặc biệt là quý thầy cô trong bộ môn Cơ Điện Tử nói riêng đã giảng dạy chúng em với tất cả tâm huyết nhà giáo và tận tình hướng dẫn, hỗ trợ chúng em về mọi mặt cũng như trong quá trình thực hiện đồ án.

Đồng thời, chúng em xin được gửi lời cảm ơn sâu sắc đến gia đình, bạn bè và người thân đã luôn ở bên cạnh, động viên, quan tâm và nâng đỡ chúng em trong suốt quá trình học tập và hoàn thiện đồ án này.

Cuối cùng, tuy nhóm đã nỗ lực để hoàn thiện đồ án cách tốt nhất nhưng với kiến thức, kinh nghiệm cũng như thời gian thực hiện vẫn còn hạn chế, nhóm chúng em chắc chắn không thể tránh khỏi những thiếu sót. Vậy nên, chúng em rất mong nhận được sự góp ý và những lời nhận xét quý báu từ các thầy cô để nhóm có thể hoàn thiện và nâng cao kiến thức. Nhóm chúng em xin tiếp nhận với tất cả lòng biết ơn.

TÓM TẮT

Trong một thế giới đang ngày càng hiện đại hóa với sự phát triển vượt bậc trong lĩnh vực nghiên cứu khoa học kỹ thuật, cuộc sống con người ngày càng tiện nghi hơn, đi kèm với điều đó chính là việc áp dụng rộng rãi những kỹ thuật máy móc hiện đại vào nhiều lĩnh vực trong đời sống: Công nghiệp, nông nghiệp, sản xuất, dịch vụ, y tế, giải trí, học tập, ... Điều đó không thể không nói đến công nghệ chế tạo robot. Ngày nay, robot đã không còn phải là thiết bị gì đó xa lạ với người tiêu dùng nữa. Nhất là trong lĩnh vực sản xuất công nghiệp, dịch vụ và y tế, Robot đã được áp dụng rộng rãi và triệt để để nâng cao hiệu suất công việc, góp phần không nhỏ đến sự phát triển trong từng lĩnh vực.

Vì Robot thực chất có rất nhiều loại dựa theo các đặc tính kỹ thuật, yêu cầu trong từng lĩnh vực nên khi nghiên cứu thiết kế robot ta cần phải hiểu được chúng ta đang ở trong hoàn cảnh, điều kiện như thế nào và mục tiêu chúng ta là gì. Trong đề tài, nhóm chúng em thực hiện nghiên cứu và thiết kế và điều khiển Mobile Robot 2 bánh xe ứng dụng trong việc hàn tự động trên mặt phẳng. Nhóm chúng em thực hiện tính toán, thiết kế, lựa chọn động cơ, xây dựng phương trình động học cho robot, mô phỏng đồ thị chuyển động của robot, sử dụng cơ cấu cảm biến hồng ngoại analog đến từ Sharp mã GP2Y0A02YK0F để tính toán vị trí robot cần đến và lấy dữ liệu từ thực nghiệm. Bên cạnh đó, để điều khiển robot chuyển động, nhóm chúng em sử dụng bộ điều khiển PID (Proportional Integral Derivative – bộ điều khiển vi tích phân tỉ lệ).

Nội dung bài báo cáo gồm 5 chương chính:

- CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI
- CHƯƠNG 2: THIẾT KẾ PHẦN CỨNG HỆ THỐNG
- CHƯƠNG 3: MÔ HÌNH HÓA VÀ XÂY DỰNG HỆ THỐNG ĐIỀU KHIỂN
- CHƯƠNG 4: KẾT QUẢ MÔ PHỎNG VÀ THỰC NGHIỆM
- CHƯƠNG 5: KẾT LUẬN

MỤC LỤC

PHIẾU NHẬN XÉT ĐỒ ÁN CƠ ĐIỆN TỬ	i
PHIẾU NHẬN XÉT ĐỒ ÁN CƠ ĐIỆN TỬ	iii
LỜI CÁM ƠN	v
TÓM TẮT	vi
MỤC LỤC	vii
DANH MỤC CÁC TỪ VIẾT TẮT	ix
DANH MỤC BẢNG	x
DANH MỤC HÌNH ẢNH	xi
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI	1
1.1. Giới thiệu	1
1.2. Tổng quan về robot hàn	1
1.3. Lý do chọn đề tài	2
1.4. Tính cấp thiết của đề tài	2
1.5. Mục tiêu nghiên cứu của đề tài	2
1.6. Phương pháp nghiên cứu	3
CHƯƠNG 2. THIẾT KẾ PHẦN CỨNG HỆ THỐNG	4
2.1. Sơ đồ tổng quan hệ thống	4
2.2. Tính toán, thiết kế phần thân robot hàn	4
2.2.1. Lựa chọn cơ cấu di chuyển	4
2.2.2. Sơ đồ động thân robot hàn	6
2.2.3 Tính toán thiết kế chọn động cơ và tỷ số truyền	6
2.2.4. Tính toán bộ truyền đai	9
2.2.5. Tính toán thiết kế trực	10
2.2.6. Tính toán chọn ô lăn	17
2.2.7. Tính toán trực vít	19
2.3. Thiết kế cơ cấu tay hàn	22
2.4. Mạch điện điều khiển	23
2.4.1. Sơ đồ khói mạch điện điều khiển	23
2.4.2. Mạch nguyên lý mạch điện điều khiển	23
2.5. Các thiết bị điện tử được sử dụng trong mạch điện	24
CHƯƠNG 3. MÔ HÌNH HÓA VÀ XÂY DỰNG HỆ THỐNG ĐIỀU KHIỂN	28
3.1. Động học Robot hàn di động hai bánh xe	28

3.1.1. Mô hình hóa thân robot	29
3.1.2. Phương trình động học thân robot.....	30
3.1.2.1. Động học thuận.....	30
3.1.2.2. Động học nghịch.....	32
3.2. Giải thuật điều khiển.....	33
3.2.1. Tổng quan sơ đồ điều khiển.....	33
3.2.2. Bộ điều khiển động cơ.....	35
3.2.2.1. Giải thuật đọc encoder	35
3.2.2.2. Bộ điều khiển PID cho động cơ.....	37
3.2.2.3. Phương pháp xác định thông số bộ điều khiển.....	40
3.2.2.4. Phương pháp giảm sai lệch điều khiển giữa hai bánh xe.	44
3.2.3. Phương pháp lấy giá trị cảm biến.....	45
3.2.4. Bộ điều khiển hệ thống.....	50
3.2.4.1. Phương án lựa chọn bộ điều khiển	50
3.2.4.2. Bộ điều khiển PID cho hệ thống.....	51
CHƯƠNG 4. KẾT QUẢ MÔ PHỎNG VÀ THỰC NGHIỆM.....	58
4.1. Kết quả mô phỏng:	58
4.1.1. Kết quả mô phỏng khi chạy trên đường thăng khi không có nhiều cảm biến:	58
4.1.2. Kết quả mô phỏng khi chạy trên đường thăng khi có nhiều cảm biến:.....	60
4.1.3. Kết quả mô phỏng khi chạy trên đường tròn khi không có nhiều cảm biến:	63
4.1.4. Kết quả mô phỏng khi chạy trên đường tròn khi có nhiều cảm biến:	66
4.2. Kết quả chạy thực nghiệm:	69
4.2.1. Kết quả chạy thực nghiệm khi chạy trên đường đường thăng:.....	69
4.2.2. Kết quả chạy thực nghiệm khi chạy trên đường đường cong:	71
CHƯƠNG 5: KẾT LUẬN, HƯỚNG PHÁT TRIỂN	73
5.1. Kết luận.....	73
5.2. Hạn chế của đề tài.....	73
5.3. Phương pháp khắc phục và hướng phát triển.....	74
TÀI LIỆU THAM KHẢO.....	75
PHỤ LỤC	76

DANH MỤC CÁC TỪ VIẾT TẮT

PID	Proportional Integral Derivative Controller	Bộ điều khiển vi tích phân tỉ lệ
PWM	Pulse Width Modulation	Điều chế độ rộng xung
CPU	Central Processing Unit	Bộ xử lý trung tâm
ADC	Analog Digital Convert	Bộ chuyển đổi tín hiệu số

DANH MỤC BẢNG

DANH MỤC HÌNH ẢNH

Hình 2.1.Sơ đồ tổng quan hệ thống	4
Hình 2.6.Mô hình tính lực tác dụng lên bánh xe.....	6
Hình 2.5.Mô hình động thân robot	6
Hình 2.7. Biểu đồ nội lực trên trục	11
Hình 2.13	24
Hình 2.11	24
Hình 2.12	24
Hình 2.16.....	25
Hình 2.15	25
Hình 2.14.....	25
Hình 2.17.....	25
Hình 2.18.....	25
Hình 2.19.....	25
Hình 2.22.....	25
Hình 2.21	25
Hình 2.20.....	25
Hình 3.1. Mô hình toán động học của robot hàn di động	28
Hình 3.2. Mô hình thân xe và bánh xe Robot hàn di động.....	29
Hình 3.3. Mô hình thể hiện tâm vận tốc tức thời khi xe chuyển động cong	30
Hình 3.4. Lưu đồ giải thuật thuật toán điều khiển chính cho hệ thống	34
Hình 3.5. Lưu đồ giải thuật thuật toán điều khiển động cơ	34
Hình 3.6. Biểu đồ xung ché độ 1X	35

Hình 3.7. Biểu đồ xung chế độ 2X	36
Hình 3.8. Biểu đồ xung chế độ 4X	36
Hình 3.9. Biểu đồ xung chế độ 4X có thể hiện tương quan giữa trạng thái xung và chiều quay động cơ	36
Hình 3.10.Thiết lập chế đọc encoder trong thanh ghi Timer 2	37
Hình 3.11.Sơ đồ thuật toán điều khiển PID có sử dụng Anti-windup	37
Hình 3.12. Hình minh họa đồ thị quá trình đáp ứng vận tốc của động cơ	41
Hình 3.13. Đồ thị quá trình đáp ứng vận tốc của động cơ	41
Hình 3.14. Đồ thị quá trình đáp ứng vận tốc của động cơ	43
Hình 3.15. Sơ đồ khói hoạt động của bộ điều khiển động cơ khi thỏa điều kiện	45
Hình 3.16 Lưu đồ giải thuật phương pháp đọc giá trị cảm biến	46
Hình 3.17 Đồ thi và biểu thức liên hệ giữa khoảng cách theo giá trị ADC	49
Hình 3.18 Sơ đồ khói cấu trúc điều khiển hệ thống	51
Hình 3.19. Khối xây dựng quỹ đạo mong muốn cho mô phỏng	51
Hình 3.20.Hàm xây dựng quỹ đạo mô phỏng	52
Hình 3.21. Khối tính toán góc theta và khoảng cách mong muốn Hình 3.22.Hàm tính toán góc thea và khoảng cách mong muốn	52
Hình 3.23.Khối điều khiển theo sai số khoảng cách (trái) và khối điều khiển theo sai số góc (phải)	52
Hình 3.24. Cấu trúc bên trong khối điều khiển theo sai số khoảng cách	53
Hình 3.25. Cấu trúc bên trong khối điều khiển theo sai số góc	53
Hình 3.26.Khối điều chỉnh vận tốc góc thiết lập bánh phải (trái) và bánh trái (phải) ..	54
Hình 3.27.Cấu trúc bên trong khối điều chỉnh vận tốc góc thiết lập bánh trái	55
Hình 3.28.Cấu trúc bên trong khối điều chỉnh vận tốc góc thiết lập bánh phải	55
Hình 3.29.Khối điều khiển PID động cơ bánh trái (trái) và động cơ bánh phải (phải).56	56

Hình 3.30. Cấu trúc khói điều khiển PID động cơ	56
Hình 3.31. Mô hình tính toán động học thuận của hệ thống	56
Hình 3.32. Hàm tính toán động học thuận của hệ thống	57
Hình 3.33. Sơ đồ khói hệ thống robot	57
Hình 4.1. Đồ thị sai số x và y	58
Hình 4.2. Đồ thị sai số góc theta	58
Hình 4.3. Đồ thị bám của vận tốc	59
Hình 4.4. Đồ thị bám của WR	59
Hình 4.5. Đồ thị bám của WL	60
Hình 4.6. Đồ thị sai số x và y	60
Hình 4.7. Đồ thị sai số góc theta	61
Hình 4.8. Đồ thị bám của vận tốc	61
Hình 4.9. Đồ thị bám của WR	62
Hình 4.10. Đồ thị bám của WL	62
Hình 4.11. Đồ thị sai số x và y	63
Hình 4.12. Đồ thị sai số góc theta	63
Hình 4.13. Đồ thị bám của vận tốc	64
Hình 4.14. Đồ thị bám của WR	64
Hình 4.15. Đồ thị bám của WL	65
Hình 4.16. Đồ thị sai số x và y	66
Hình 4.17. Đồ thị sai số góc theta	66
Hình 4.18. Đồ thị bám của vận tốc	67
Hình 4.19. Đồ thị bám của WR	67
Hình 4.20. Đồ thị bám của WL	68

Hình 4.21.Đồ thị sai số x và y	69
Hình 4.22.Đồ thị sai số khoảng cách	69
Hình 4.23.Đồ thị sai số góc theta	70
Hình 4.24.Đồ thị vận tốc.....	70
Hình 4.25.Đồ thị sai số x và y	71
Hình 4.26.Đồ thị sai số khoảng cách.	71
Hình 4.27.Đồ thị sai số theta.....	72
Hình 4.28.Đồ thị vận tốc:.....	72

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu.

1.2. Tổng quan về robot hàn.

Robot hàn là một loại robot đã được lập trình sẵn giúp người chủ hoàn toàn tự động hóa quá trình hàn cơ khí.

Tùy vào từng mục đích sử dụng mà thị trường đã cho ra đời nhiều loại robot hàn gia công cơ khí khác nhau về đầu hàn như robot hàn tig (hình 1.2), hàn dây, hàn điểm hay hàn laze (hình 1.1).



Hình 1.1. Robot hàn laser

<https://robovina.vn/robot-han-laser.html>



Hình 1.2. Robot hàn tig

<https://weldcut.vn/robot-han-tig-1>

Trong các loại robot hàn, robot hàn di động với khả năng đảm bảo vận tốc hàn ổn định, góc hàn cố định và đường hàn dài, nên đáp ứng rất tốt cho các yêu cầu hàn theo đường. Robot hàn di động là phiên bản hàn tự động tiên tiến cao, đã và vẫn đang được nghiên cứu phát triển. Một số robot hàn được nghiên cứu như robot hàn di động hai bánh xe được phát triển bởi Sang-Bong Kim và đồng nghiệp hay robot hàn rower do P.Gonzalez De Santos đề xuất,...

Lợi thế của robot hàn di động:

- Tiếp cận các vị trí khó tiếp cận.
- Thực hiện các mối hàn phức tạp và chính xác nhanh hơn so với hàn thủ công.
- Tăng năng suất, giảm chi phí thi công.
- Giúp con người hàn trong những vùng không gian hẹp hoặc những điều kiện khắc nghiệt.

Tất cả những điều trên giúp giải phóng thời gian cho việc sản xuất và cho phép thực hiện các dự án linh hoạt và hiệu quả hơn.

1.3. Lý do chọn đề tài

Với sự phát triển không ngừng của khoa học kỹ thuật kèm với mức độ truy cầu điều kiện sống của con người ngày càng được nâng lên, và khi nói đến khoa học kỹ thuật thì không thể không nói đến robot di động, một sản phẩm nghiên cứu đang gây ra sức ảnh hưởng lớn đến cuộc sống của con người. Đặc biệt, trong lĩnh vực hàn nói riêng và cáclĩnh vực mang tính tiếp xúc với yếu tố độc hại nói chung thì việc có thể thiết kế, phát triển một robot có thể thay thế con người làm việc trong những môi trường như thế là hết sức tuyệt vời. Vậy nên, nhóm chúng em quyết định chọn đề tài “**Thiết kế và điều khiển robot di động ứng dụng trong hàn tự động**”

1.4. Tính cấp thiết của đề tài

Ở phía trên chúng ta đã đề cập đến điều tuyệt vời mà robot có thể đem lại cho cuộc sống con người, nhưng trong quá trình thực hiện được điều đó thì lại không ít gian nan. Đối với robot hàn đang được nghiên cứu gần đây thì thường chú trọng đến dành cho sản xuất công nghiệp, tính di động vẫn chưa có thể đáp ứng tốt được như ý tưởng. Vậy nên việc nghiên cứu một robot hàn có tính di động cao dễ dàng vươn tới các điểm khó là một vấn đề cần thiết.

1.5. Mục tiêu nghiên cứu của đề tài

Để tạo đường hàn chất lượng, robot hàn cần di chuyển chính xác và ổn định. Điều này đòi hỏi cơ cấu robot đưa đũa hàn dọc theo đường hàn với vận tốc và góc hàn không đổi. Cơ cấu di động cần di chuyển sao cho đầu hàn có thể đạt đến vị trí hàn thuận lợi. Để tăng chính xác, cần ước lượng mô-men quán tính và đối phó với sự thay đổi đột ngột

của đường hàn. Điều này giúp robot hàn duy trì độ ổn định và chính xác khi các tham số thay đổi.

Từ yêu cầu trên các mục tiêu cần nghiên cứu về đề tài này chúng em đưa ra như sau:

+Nghiên cứu thuật toán quỹ đạo và xây dựng bộ điều khiển cho robot

+Thiết kế cơ khí (Mô hình robot, PCB)

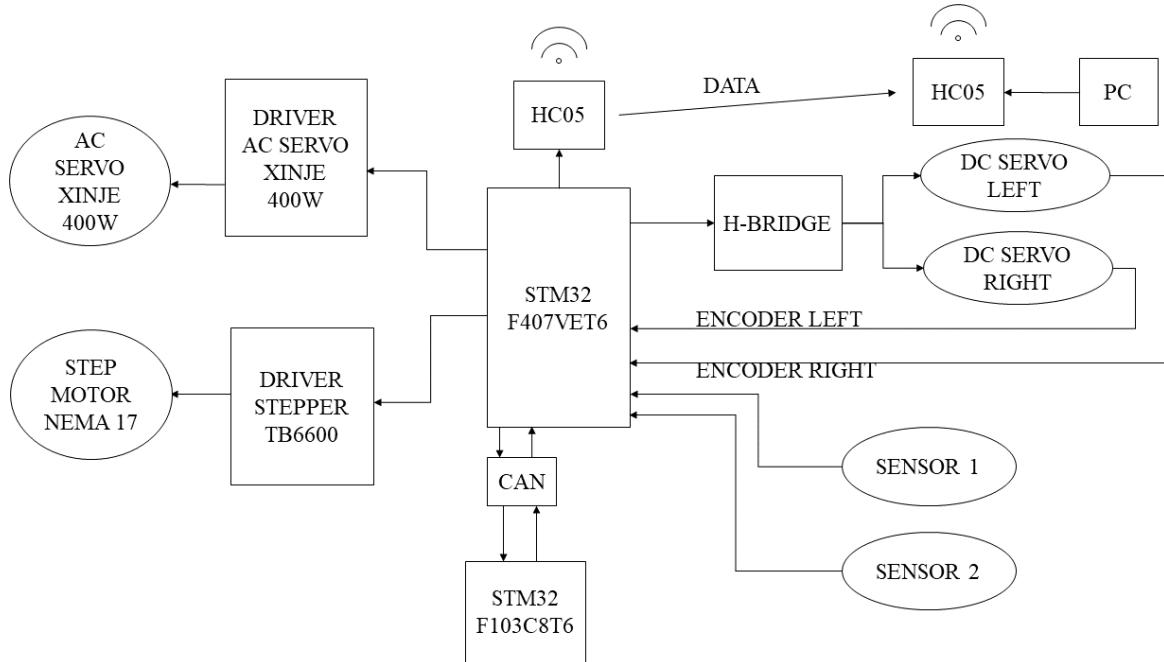
+Xử lý tín hiệu cảm biến

1.6. Phương pháp nghiên cứu

Khi thực hiện đề tài này, nhóm chủ yếu sử dụng phương pháp nghiên cứu kết hợp giữa lý thuyết và thực nghiệm. Về nghiên cứu lý thuyết, nhóm đã tham khảo các tài liệu thiết kế cơ khí, điện điện-tử, hệ thống điều khiển và các bài báo khoa học, về thực nghiệm, nhóm đã tiến hành thiết kế, lắp ráp mô hình và cho robot hoạt động lấy dữ liệu thực tế.

CHƯƠNG 2. THIẾT KẾ PHẦN CỨNG HỆ THỐNG

2.1. Sơ đồ tổng quan hệ thống



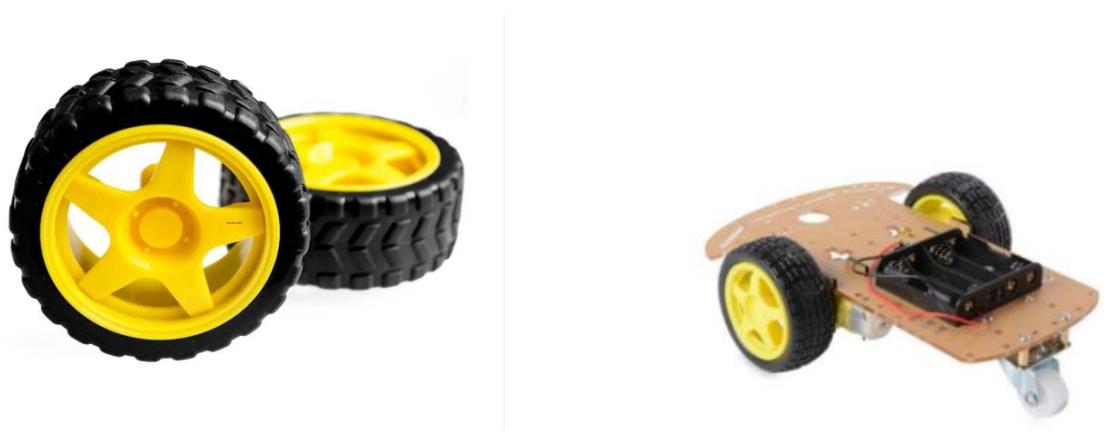
Hình 2.1. Sơ đồ tổng quan hệ thống

2.2. Tính toán, thiết kế phần thân robot hàn.

2.2.1. Lựa chọn cơ cấu di chuyển.

Hiện nay, các robot di động có đa dạng các cơ cấu di chuyển khác nhau và mỗi cơ cấu di chuyển lại có ưu nhược riêng, tùy thuộc vào nhu cầu cũng như ứng dụng để lựa chọn cơ cấu di chuyển. Trong đó:

- Cơ cấu di chuyển bằng bánh xe truyền thống: Là loại bánh xe xuất hiện lâu đời nhất và vẫn được sử dụng phổ biến trong nhiều lĩnh vực robot di động. Ưu điểm của bánh xe truyền thống là giá thành rẻ, cấu tạo đơn giản, lực kéo của động cơ được tối ưu so với các cơ cấu di chuyển còn lại. Nhược điểm là không di chuyển linh hoạt bằng các bánh xe còn lại và dễ bị trượt bánh.



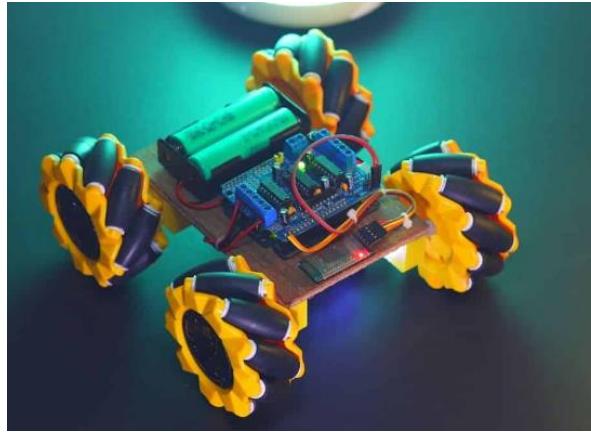
Hình 2.2. bánh xe truyền thống và cơ cấu thông thường

- Cơ cấu di chuyển bằng bánh xe Omni: là một loại bánh xe đa hướng, với cấu trúc đặc biệt gồm các bánh trượt tự do được bố trí đối xứng nhau theo hình của bánh xe truyền thống nhằm kết hợp chuyển động của bánh xe truyền thống và chuyển động tịnh tiến của các bánh trượt một cách độc lập với nhau. Đối với robot di động, hệ sẽ gồm 3 bánh omni đặt cách nhau góc 120 độ. Ưu điểm là khả năng di chuyển linh hoạt và ổn định, độ chính xác cao, cấu trúc tổng thể nhỏ gọn. Mô men kéo thấp, tuổi thọ thấp, khả năng sửa chữa, thay thế khó khăn.



Hình 2.3. Bánh xe Omni và cơ cấu

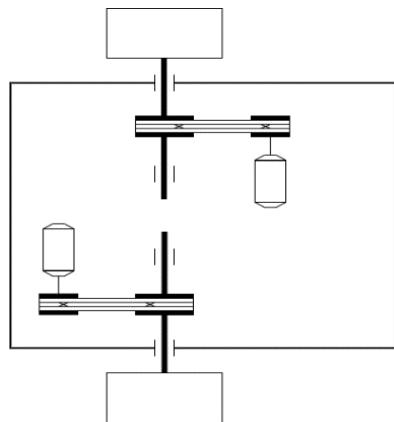
- Cơ cấu di chuyển sử dụng bánh mecanum: Là bánh xe đa hướng có cấu tạo gần giống bánh omni. Tuy nhiên các bánh trượt tự do được bố trí một góc 45 độ so với momen vecto dao động tạo ra trên bánh xe. Nhờ đó, có thể có thể di chuyển như bánh xe truyền thống và khả năng định tuyến một góc 45 độ so với phương di chuyển thẳng. Cơ cấu này thường sử dụng 4 bánh được bố trí 4 góc hình chữ nhật. Ưu và nhược điểm thì tương tự như bánh omni.



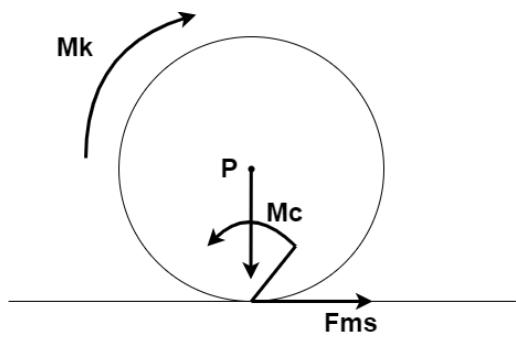
Hình 2.4. Bánh mecanum và cơ cấu

- Kết luận: Nhóm chúng em quyết định sử dụng bánh xe truyền thống vì đè tài yêu cầu lực kéo lớn.

2.2.2. Sơ đồ động thân robot hàn.



Hình 2.5.Mô hình động thân robot



Hình 2.6.Mô hình tính lực tác dụng lên
bánh xe

2.2.3 Tính toán thiết kế chọn động cơ và tỷ số truyền

- Chọn vật liệu:

Thép	Nhiệt luyện	Độ cứng	GHB	GHC
CT 45	Thường hoá	230	600	340

Bảng 2.1.Vật liệu cho trục động cơ

- Chon động cơ:

Bán kính bánh xe :

$$R = 0,065 \text{ m}$$

Gia tốc trọng trường :

$$g = 9,81 \text{ m}^2/\text{s}$$

Khối lượng xe :

$$m = 20 \text{ kg}$$

Trọng lực tác dụng lên xe :

$$P = m \cdot g = 196,2 \text{ N}$$

Hệ số cản lăn :

$$\mu_{msl} = 0,015$$

Lực cản lăn:

$$F_{ms} = \mu_{msl} \cdot m \cdot g = 2,943 \text{ N}$$

Momen kéo của xe :

$$M_k = (P + F_{ms}) \cdot R = (196,2 + 2,943) \cdot 6,5 \cdot 10^{-3} = 12,94 \text{ Nm}$$

Momen kéo đối với mỗi bánh:

$$M_{k1} = \frac{M_k}{2} = 6,472 \text{ Nm}$$

Thiết kế robot hàn với tốc độ tối đa là :

$$V_c = 125 \text{ mm/s}$$

Tốc độ góc tối đa của robot hàn :

$$\omega_{max} = \frac{V}{R} = \frac{125}{65} = 1,923 \text{ rad/s}$$

Công suất cần thiết cho tải :

$$P_t = W_{max} \cdot M_k = 6,472 \cdot 1,923 = 12,446 \text{ W}$$

Tổng hiệu suất bộ truyền:

$$\mu = \sum n_d \cdot n_{ol}^2 \cdot n_{hs} = 0,95 \cdot 0,99^2 \cdot 0,95 = 0,8845$$

Công suất cần thiết cho động cơ :

$$P_{dc} = \frac{P_t}{\mu} = \frac{12,446}{0,8845} = 14,071 \text{ W}$$

Vận tốc dài của robot hàn :

$$V_{dmax} = \frac{V_c \cdot 60}{1000} = \frac{125.60}{1000} = 7,5 \text{ RPM}$$

Số vòng quay trực công tác :

$$n_t = \frac{V_{dmax} \cdot 1000}{\pi \cdot 2 \cdot R} = \frac{7,5 \cdot 1000}{\pi \cdot 2 \cdot 65} = 18,36 \text{ RPM}$$

Cho bộ truyền đai với :

$$u_d = 2$$

Số vòng quay trực động cơ :

$$n_{dc} = 2 \cdot 18,36 = 36,72 \text{ RPM}$$

Chọn động cơ với

$$\begin{cases} n_{dc} = 37 \text{ RPM} \\ P_t = 60W \end{cases}$$

- Tính toán thông số trên trực

Momen kéo đôi với mỗi bánh :

$$Mk1 = \frac{Mk}{2} = 6,472 \text{ Nm}$$

Công suất cần thiết cho tải (trục 2):

$$P2 = Pt = W_{max} \cdot Mk = 6,472 \cdot 1,923 = 12,446 \text{ W}$$

Công suất cần thiết cho động cơ (trục 1):

$$P_1 = P_{dc} = \frac{Pt}{nd \cdot nol^2} = \frac{12,446}{0,95 \cdot 0,99^2} = 13,367 \text{ W}$$

Số vòng quay trục công tác:

$$nt = \frac{Vd\max \cdot 1000}{\pi \cdot 2 \cdot R} = \frac{7,5 \cdot 1000}{\pi \cdot 2,65} = 18,36 \text{ RPM}$$

Số vòng quay trục động cơ : $ndc = 18,36 \cdot 2 = 36,72 \text{ RPM}$

Momen xoắn trên từng trục :

$$Ti = \frac{9,55 \cdot 10^6 \cdot Pi}{ni}$$

Thông số	Trục	I (trục động cơ)	II (trục công tác)
Tỉ số truyền u		2	
Số vòng quay n (RPM)		36,72	18,36
Công suất P (KW)		$13,367 \cdot 10^{-3}$	$12,446 \cdot 10^{-3}$
Momen T (Nmm)		3476,439	6473,818

2.2.4. Tính toán bộ truyền đai

Đường kính bánh đai dẫn:

$$d_1 = 0,0185 \text{ m} = 18.5 \text{ mm}$$

Đường kính bánh đai bị dẫn dãn:

$$d_2 = 0,0375 \text{ m} = 37.5 \text{ mm}$$

Tra tài liệu : $a/d_2 = 1,2$ suy ra chọn sơ bộ khoảng cách trục $a = 45 \text{ mm}$

Chiều dài đai:

$$L = 2a + \frac{\pi(d_1 + d_2)}{2} + \frac{(d_1 - d_2)^2}{4a} = 179,97 \text{ mm}$$

Tính lại :

$$a = \frac{1}{4} \left\{ L - \frac{\pi(d_2 + d_1)}{2} + \sqrt{\left[L - \frac{\pi(d_2 + d_1)}{2} \right]^2 - 2(d_2 - d_1)^2} \right\} = 45,98 \text{ mm}$$

Góc ôm alpha:

$$a_1 = 180 - \frac{57(d_2 - d_1)}{a} = 156,446 > 120 \text{ (thoả)}$$

Lực tác dụng lên trục 2 :

$$F_{t2} = \frac{2T_2}{d_2} = \frac{2.6473,818}{37,5} = 345,2702933 \text{ N}$$

$$F_{r2} = 2 \cdot F_{t2} \cdot \sin\left(\frac{a_1}{2}\right) = 676,0046471 \text{ N}$$

$$\sum M_{bx} = 0 \leftrightarrow \frac{P \cdot 60}{2} - F_{rc} \cdot 20 + R_{dy} \cdot 40 = 0$$

$$\rightarrow R_{dy} = \frac{676,004 \cdot 20 - 196,230}{40} = 190,86 \text{ N}$$

Oy:

$$-\frac{P}{2} - R_{by} - F_{rc} + R_{dy} = 0$$

$$\rightarrow R_{dy} = -\frac{196,2}{2} - 676,004 + 190,85 = -583,264 \text{ N}$$

$$\sum M_{by} = 0 \leftrightarrow M_k - F_t \cdot 20 + R_{dx} \cdot 40 = 0$$

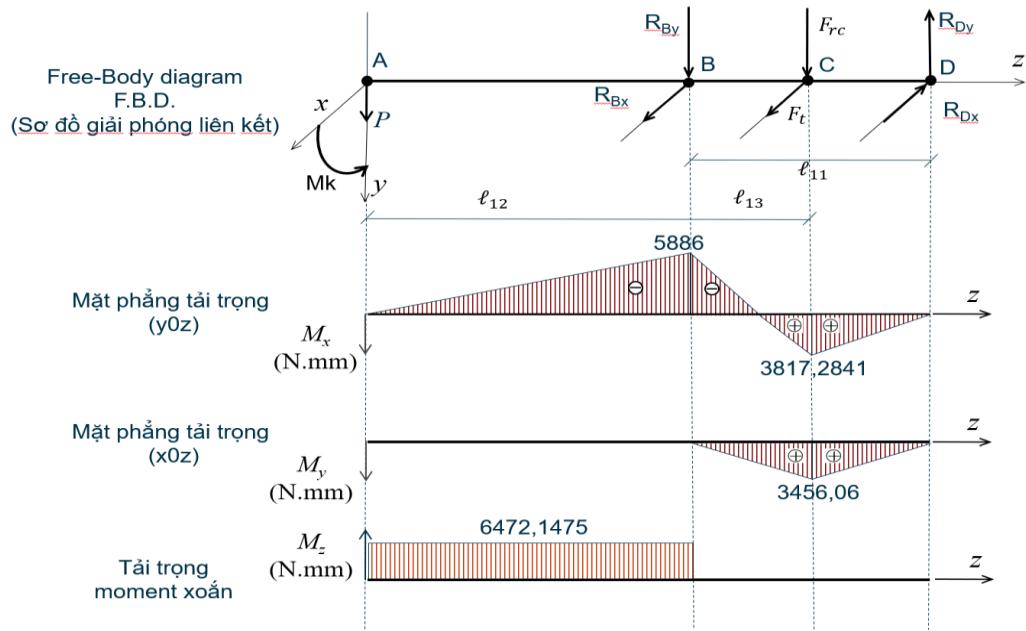
$$\rightarrow R_{dx} = \frac{345,27 \cdot 20 - 6,472}{40} = 172,4794 \text{ N}$$

Ox:

$$-R_{Bx} - F_t + R_{Dx} = 0 \rightarrow R_{Bx} = -345,27 + 172,47 = -172,803$$

2.2.5. Tính toán thiết kế trực

Biểu đồ nội lực – Trục #02



Hình 2.7. Biểu đồ nội lực trên trục

- Momen tại các điểm nguy hiểm:

$$M_A = \sqrt{M_{Ax}^2 + M_{Ay}^2 + 0,75 \cdot T^2} = \sqrt{0^2 + 0^2 + 0,75 \cdot 6472,1475^2}$$

$$= 5605,04415 \text{ Nmm}$$

$$M_B = \sqrt{M_{Bx}^2 + M_{By}^2 + 0,75 \cdot T^2} = \sqrt{5886^2 + 0^2 + 0,75 \cdot 6472,1475^2}$$

$$= 8127,8236 \text{ Nmm}$$

$$M_C = \sqrt{M_{Cx}^2 + M_{Cy}^2 + 0,75 \cdot T^2} = \sqrt{3817,2841^2 + 3456,06^2 + 0,75 \cdot 6472,1475^2}$$

$$= 7611,3421 \text{ Nmm}$$

$$M_D = \sqrt{M_{Dx}^2 + M_{Dy}^2 + 0,75 \cdot T^2} = \sqrt{0^2 + 0^2 + 0,75 \cdot 0^2} = 0 \text{ Nmm}$$

- Đường kính trục tại các tiết diện nguy hiểm:

$$d_A \geq \sqrt[3]{\frac{M_{Atd}}{0.1 \times [\sigma_F]}} = \sqrt[3]{\frac{5605,04415}{0.1 \times 380}} = 9,6 \text{ m}$$

$$d_B \geq \sqrt[3]{\frac{M_{Btd}}{0.1 \times [\sigma_F]}} = \sqrt[3]{\frac{8127,8236}{0.1 \times 380}} = 10,88 \text{ m}$$

$$d_C \geq \sqrt[3]{\frac{M_{Ctd}}{0.1 \times [\sigma_F]}} = \sqrt[3]{\frac{7611,3421}{0.1 \times 380}} = 10,65 \text{ m}$$

$$d_D \geq \sqrt[3]{\frac{M_{Dtd}}{0.1 \times [\sigma_F]}} = \sqrt[3]{\frac{0}{0.1 \times 380}} = 0 \text{ m}$$

- Kiểm nghiệm trực – bended mối

- Tại tiết diện A

$$S_A = \frac{S_{\sigma A} \cdot S_{\tau A}}{\sqrt{S_{\sigma A}^2 + S_{\tau A}^2}} = \frac{3,85 \cdot 8,49}{\sqrt{3,85^2 + 8,49^2}} = 3,5 > [s] = 3 \quad (1)$$

Trong đó :

$$S_{\sigma A} = \frac{\sigma_{-1}}{K_{\sigma dA} \cdot \sigma_{aA} + \varphi_{\sigma} \cdot \sigma_{mA}} = \frac{261,6}{1,19 \cdot 57,1 + 0,05 \cdot 0} = 3,85$$

Với:

$$\sigma_{-1} = 0,436 \cdot \sigma_A = 0,436 \cdot 600 = 261,6 \text{ MPa}$$

$$\sigma_{aA} = \frac{M_A}{W_A} = \frac{5605,04415}{98,17} = 57,1 \text{ MPa}$$

Với:

$$\begin{cases} M_A = \sqrt{M_{xA}^2 + M_{yA}^2} = 5605,04415 \text{ N.mm} \\ W_A = \frac{\pi \cdot d^3}{32} = \frac{\pi \cdot 10^3}{32} = 98,17 \text{ mm}^3 \end{cases}$$

$$\varphi_{\sigma} = 0,05 (\text{bảng 10.7 tài liệu [1]})$$

$\sigma_{mA} = 0$ (công thức 10.22 tài liệu [1])

$$K_{\tau dA} = \frac{\frac{K_\sigma}{\varepsilon_\sigma} + K_x - 1}{K_y} = \frac{\frac{1,76}{0,95} + 1,06 - 1}{1,6} = 1,19$$

Với :

$$\varepsilon_\sigma = 0,95 \text{ (bảng 10.10 tài liệu [1])}$$

$$K_x = 1,06 \text{ (bảng 10.8 tài liệu [1])}$$

$$K_y = 1,6 \text{ (bảng 10.9 tài liệu [1])}$$

$$K_\sigma = 1,76 \text{ (bảng 10.10 tài liệu [1])}$$

Trong đó :

$$S_{\tau A} = \frac{\tau_{-1}}{K_{\tau dA} \cdot \tau_{aA} + \varphi_\sigma \cdot \tau_{mA}} = \frac{151,7}{1,084 \cdot 16,48 + 0} = 8,49$$

Với:

$$\tau_{-1} = 0,58 \cdot \sigma_{-1} = 0,58 \cdot 261,6 = 151,7 \text{ MPa}$$

$$\tau_{aA} = \tau_{mA} = \frac{\tau_A}{2 \cdot W_{oA}} = \frac{6472,15}{2 \cdot 196,35} = 16,48 \text{ MPa}$$

Với:

$$\begin{cases} T_A = 6472,15 \text{ N.mm} \\ W_{oA} = \frac{\pi \cdot d^3}{16} = \frac{\pi \cdot 10^3}{16} = 196,35 \text{ mm}^3 \end{cases}$$

$$\varphi_\tau = 0 \text{ (bảng 10.7 tài liệu [1])}$$

$$K_{\tau dA} = \frac{\frac{K_\tau}{\varepsilon_\tau} + K_x - 1}{K_y} = \frac{\frac{1,54}{0,92} + 1,06 - 1}{1,6} = 1,084$$

Với

$$\varepsilon_\tau = 0,92 \text{ (bảng 10.10 tài liệu [1])}$$

$$K_x = 1,06 \text{ (bảng 10.8 tài liệu [1])}$$

$$K_y = 1,6 \text{ (bảng 10.9 tài liệu [1])}$$

$$K_\tau = 1,54 \text{ (bảng 10.10 tài liệu [1])}$$

- Tại tiết diện B

$$S_B = \frac{S_{\sigma B} \cdot S_{\tau B}}{\sqrt{S_{\sigma B}^2 + S_{\tau B}^2}} = \frac{3,53 \cdot 11,3}{\sqrt{3,53^2 + 11,3^2}} = 3,369 > [s] = 3 \quad (2)$$

Trong đó :

$$S_{\sigma B} = \frac{\sigma_{-1}}{K_{\sigma dB} \cdot \sigma_{aB} + \varphi_\sigma \cdot \sigma_{mB}} = \frac{261,6}{1,19 \cdot 62,2 + 0,05 \cdot 0} = 3,53$$

Với:

$$\sigma_{-1} = 0,436 \cdot \sigma_B = 0,436 \cdot 600 = 261,6 \text{ MPa}$$

$$\sigma_{aB} = \frac{M_B}{W_B} = \frac{8127,8236}{130,67} = 62,2 \text{ MP}$$

Với:

$$\begin{cases} M_B = \sqrt{M_{xB}^2 + M_{yB}^2} = 8127,8236 \text{ N.mm} \\ W_B = \frac{\pi \cdot d^3}{32} = \frac{\pi \cdot 11^3}{32} = 130,67 \text{ mm}^3 \end{cases}$$

$$\varphi_\sigma = 0,05 \text{ (bảng 10.7 tài liệu [1])}$$

$$\sigma_{mB} = 0 \text{ (công thức 10.22 tài liệu [1])}$$

$$K_{\sigma dB} = \frac{\frac{K_\sigma}{\varepsilon_\sigma} + K_x - 1}{K_y} = \frac{\frac{1,76}{0,95} + 1,06 - 1}{1,6} = 1,19$$

Với:

$$\varepsilon_\sigma = 0,95 \text{ (bảng 10.10 tài liệu [1])}$$

$$K_x = 1,06 \text{ (bảng 10.8 tài liệu [1])}$$

$$K_y = 1,6 \text{ (bảng 10.9 tài liệu [1])}$$

$$K_\sigma = 1,76 \text{ (bảng 10.10 tài liệu [1])}$$

Trong đó:

$$S_{\tau B} = \frac{\tau_{-1}}{K_{\tau dB} \cdot \tau_{aB} + \varphi_\sigma \cdot \tau_{mB}} = \frac{151,7}{1,084 \cdot 12,38 + 0} = 11,3$$

Với:

$$\tau_{-1} = 0,58 \cdot \sigma_{-1} = 0,58 \cdot 261,6 = 151,7 \text{ MPa}$$

$$\tau_{aB} = \tau_{mB} = \frac{\tau_B}{2 \cdot W_{oB}} = \frac{6472,15}{2 \cdot 261,34} = 12,38 \text{ MPa}$$

Với:

$$\begin{cases} T_B = 6472,15 \text{ N.mm} \\ W_{oB} = \frac{\pi \cdot d^3}{16} = \frac{\pi \cdot 11^3}{16} = 261,34 \text{ mm}^3 \end{cases}$$

$$\varphi_\tau = 0 \text{ (bảng 10.7 tài liệu [1])}$$

$$K_{\tau dB} = \frac{\frac{K_\tau}{\varepsilon_\tau} + K_x - 1}{K_y} = \frac{\frac{1,54}{0,92} + 1,06 - 1}{1,6} = 1,084$$

Với:

$$\varepsilon_\tau = 0,92 \text{ (bảng 10.10 tài liệu [1])}$$

$$K_x = 1,06 \text{ (bảng 10.8 tài liệu [1])}$$

$$K_y = 1,6 \text{ (bảng 10.9 tài liệu [1])}$$

$$K_\tau = 1,54 \text{ (bảng 10.10 tài liệu [1])}$$

- Tại tiết diện C

$$S_C = \frac{S_{\sigma C} \cdot S_{\tau C}}{\sqrt{S_{\sigma C}^2 + S_{\tau C}^2}} = \frac{3,77 \cdot 11,3}{\sqrt{3,77^2 + 11,3^2}} = 3,57 > [s] = 3 \quad (3)$$

Trong đó :

$$S_{\sigma C} = \frac{\sigma_{-1}}{K_{\sigma dC} \cdot \sigma_{aC} + \varphi_{\sigma} \cdot \sigma_{mC}} = \frac{261,6}{1,19 \cdot 58,25 + 0,05 \cdot 0} = 3,77$$

Với:

$$\sigma_{-1} = 0,436 \cdot \sigma_C = 0,436 \cdot 600 = 261,6 \text{ MPa}$$

$$\sigma_{aC} = \frac{M_C}{W_C} = \frac{7611,3421}{130,67} = 58,25 \text{ MPa}$$

Với:

$$\begin{cases} M_C = \sqrt{M_{xC}^2 + M_{yC}^2} = 7611,3421 \text{ N.mm} \\ W_C = \frac{\pi \cdot d^3}{32} = \frac{\pi \cdot 11^3}{32} = 130,67 \text{ mm}^3 \end{cases}$$

$$\varphi_{\sigma} = 0,05 (\text{ bảng 10.7 tài liệu [1]})$$

$$\sigma_{mC} = 0 (\text{ công thức 10.22 tài liệu [1]})$$

$$K_{\sigma dC} = \frac{\frac{K_{\sigma}}{\varepsilon_{\sigma}} + K_x - 1}{K_y} = \frac{\frac{1,76}{0,95} + 1,06 - 1}{1,6} = 1,19$$

Với:

$$\varepsilon_{\sigma} = 0,95 (\text{ bảng 10.10 tài liệu [1]})$$

$$K_x = 1,06 (\text{ bảng 10.8 tài liệu [1]})$$

$$K_y = 1,6 (\text{ bảng 10.9 tài liệu [1]})$$

$$K_{\sigma} = 1,76 (\text{ bảng 10.10 tài liệu [1]})$$

Trong đó :

$$S_{\tau C} = \frac{\tau_{-1}}{K_{\tau dC} \cdot \tau_{aC} + \varphi_{\sigma} \cdot \tau_{mC}} = \frac{151,7}{1,084 \cdot 12,38 + 0} = 11,3$$

Với:

$$\tau_{-1} = 0,58 \cdot \sigma_{-1} = 0,58 \cdot 261,6 = 151,7 \text{ MPa}$$

$$\tau_{ac} = \tau_{mc} = \frac{\tau_c}{2 \cdot W_{oc}} = \frac{6472,15}{2.261,34} = 12,38 \text{ MPa}$$

Với:

$$\begin{cases} T_c = 6472,15 \text{ N.mm} \\ W_{oc} = \frac{\pi \cdot d^3}{16} = \frac{\pi \cdot 11^3}{16} = 261,34 \text{ mm}^3 \end{cases}$$

$\varphi_\tau = 0$ (bảng 10.7 tài liệu [1])

$$K_{tdc} = \frac{\frac{K_\tau}{\varepsilon_\tau} + K_x - 1}{K_y} = \frac{\frac{1,54}{0,92} + 1,06 - 1}{1,6} = 1,084$$

Với:

$\varepsilon_\tau = 0,92$ (bảng 10.10 tài liệu [1])

$K_x = 1,06$ (bảng 10.8 tài liệu [1])

$K_y = 1,6$ (bảng 10.9 tài liệu [1])

$K_\tau = 1,54$ (bảng 10.10 tài liệu [1])

Từ (1) (2) (3) Thoả điều kiện về độ bền mõi

2.2.6. Tính toán chọn ố lăn

Tải trọng hướng tâm tác dụng lên ố B

$$F_{rB} = \sqrt{R_{Bx}^2 + R_{By}^2} = \sqrt{172,803^2 + 583,264^2} = 608,323 \text{ N}$$

Tải trọng hướng tâm tác dụng lên ố D

$$F_{rD} = \sqrt{R_{Dx}^2 + R_{Dy}^2} = \sqrt{172,4794^2 + 190,86^2} = 257,25 \text{ N}$$

Lực dọc trục $F_a = 0 \text{ N}$

Đường kính trục $d_D = d_B = 11 \text{ mm}$

Ta có $\frac{F_a}{F_r} = 0 < 0,3$ Dựa điều kiện a) trang 212 tài liệu [1] suy ra chọn ố bi đỡ 1 dây

Chọn sơ bộ dùng ố bi cỡ nhẹ 1000901 có các thông số sau:

$$\begin{cases} d = 12 \text{ mm} \\ D = 24 \text{ mm} \\ B = 6 \text{ mm} \\ r = 0,5 \text{ mm} \\ C = 2,66 \text{ KN} \\ C_o = 1,38 \text{ KN} \end{cases}$$

Kiểm tra tải động

Dựa vào bảng 11.4 tài liệu [1] :

$$\frac{F_a}{C_o} = \frac{0}{1,38 \cdot 10^3} = 0 < 0,014$$

Suy ra chọn $e = 0,19$

Ta có :

$$\frac{F_a}{V \cdot F_{rB}} = \frac{0}{1 \cdot F_{rB}} = 0 < e (\text{Vòng trong quay nên } V = 1)$$

Do đó ta được các giá trị $\begin{cases} X = 1 \\ Y = 0 \end{cases}$ (bảng 11.4 tài liệu [1])

Tải trọng động (dựa vào công thức 11.3 tài liệu [1])

$$Q_B = (X \cdot V \cdot F_{rB} + Y \cdot F_a) \cdot k_\tau \cdot k_d = (1 \cdot 1.608,323 + 0) \cdot 1 \cdot 1 = 608,323 \text{ N}$$

$$Q_D = (X \cdot V \cdot F_{rD} + Y \cdot F_a) \cdot k_\tau \cdot k_d = (1 \cdot 1.257,25 + 0) \cdot 1 \cdot 1 = 257,25 \text{ N}$$

$$Q_B > Q_D \text{ suy ra tính theo ổ B}$$

Dựa theo công thức $L_h = 10000 \text{ h}$ dựa vào điều kiện bảng 11.2 tài liệu [1]

Tuổi thọ của ổ tính theo đơn vị triệu vòng quay :

$$L = \frac{L_h \cdot 60 \cdot n_2}{10^6} = \frac{10000 \cdot 60 \cdot 18,36}{10^6} = 11,06 \text{ (triệu vòng quay)}$$

Khả năng tải động tính toán

$$C_u = Q_B \cdot \sqrt[3]{L} = 608,323 \sqrt[3]{11,016} = 1353,5535 \text{ N} = 1,353 \text{ KN} < C = 2,66 \text{ KN}$$

Suy ra thoả tải trọng động

Tuổi thọ của ô tính theo đơn vị giờ

$$L = \frac{L \cdot 10^6}{60 \cdot 18,36} = 52832 \text{ giờ}$$

Kiểm tra tải tĩnh

$$\begin{cases} X = 1 \\ Y = 0 \end{cases} \text{ theo bảng 11.6 tài liệu [1]}$$

$$\begin{cases} Q_o = X_o \cdot F_{rB} + Y_o \cdot F_a = 0,6 \cdot 686,323 = 411,79 N \\ Q_o = 686,323 N \end{cases}$$

Vậy

$$Q_o = F_{rB} = 686,323 N = 0,686 KN < C_o = 1,38 KN$$

Suy ra thoả tải tĩnh

2.2.7. Tính toán trực vít

Thép	Nhiệt luyện	Độ cứng	GHB	GHC
CT 45	Thường hoá	230	600	340

Ta có bộ truyền đai như ở phần thiết kế trực ở trên nên ta chọn

$$u_d = 2$$

Khối lượng tải

$$m = 4 \text{ kg}$$

Đường kính trung bình của ren

$$d_2 \geq \sqrt{\frac{Fa}{\pi \cdot \varphi_H \cdot \varphi_h \cdot [q]}} = \sqrt{\frac{39,24}{\pi \cdot 2,0,5 \cdot 10}} = 1,117 mm$$

Với $\varphi_H = 2$

$$\varphi_h = 0,5$$

$$[q] = 10 MPA$$

Chọn đường kính trung bình $d_2 = 11 \text{ mm}$

Chọn đường kính ngoài $d = 12 \text{ mm}$

Chọn đường kính trong $d_1 = 10 \text{ mm}$

Chiều dài $L = 30 \text{ mm}$

Chọn bước vít $p = 2 \text{ mm}$ Chọn $Z_h = 4$. Ta có bước vít:

$$p_h = Z_h \cdot p = 4 \cdot 2 = 8 \text{ mm}$$

Và góc vít

$$\gamma = \arctg \left[\frac{p_h}{(\pi \cdot d_2)} \right] = \arctg \left[\frac{8}{(\pi \cdot 7)} \right] = 20^\circ$$

Sau khi xác định góc vít ta kiểm tra điều kiện tự hâm:

$$p = \arctg \left[\frac{f}{\cos(\delta)} \right]$$

Trong đó $\delta = 15^\circ$ góc nghiêng cạnh ren làm việc ; ren hình thang

f : hệ số ma sát, thép - đồng thanh không thiếc $f = 0,4$

$$p = \arctg \left[\frac{0,4}{\cos(15^\circ)} \right] = 22,5^\circ$$

$\gamma < p$, thoả điều kiện tự hâm

Hiệu suất bô truyền

$$\mu = \frac{S_v}{S_t} = \frac{\pi \cdot dv2}{Z_H \cdot p} = \frac{\pi \cdot 36}{4.2} = 14,137$$

Ta có công thức liên hệ :

$$F_a = \mu \cdot n \cdot Ft$$

$$F_{t2} = \frac{Fa}{\mu \cdot n} = \frac{39,24}{14,137} = 6,939 \text{ N}$$

$$V_2 = \frac{n2 \cdot Z \cdot P}{60 \cdot 10^3} = \frac{15 \cdot 4 \cdot 2}{60 \cdot 000} = 2 \cdot 10^{-3} (\text{m/s})$$

Với

$$u_d = \frac{n1}{n2} = \frac{ndc}{n2}$$

$$n_2 = \frac{ndc}{nd} = \frac{30}{2} = 15 \text{ rpm}$$

$$P_2 = \frac{F_{t2} \cdot V2}{1000} = \frac{6,939 \cdot 2 \cdot 10^{-3}}{1000} = 0,014 \text{ KW} = 14 \text{ W}$$

$$P_1 = \frac{P_2}{0,95} = 0,029 \text{ KW} = 29 \text{ W}$$

$$T_2 = \frac{9,55 \cdot 10^6 \cdot P_2}{n_2} = \frac{9,55 \cdot 10^6 \cdot 0,014}{15} = 8913 \text{ Nmm}$$

- Kiểm nghiệm trực vít về độ bền :

$$\begin{aligned} \sigma_{td} &= \sqrt{\sigma^2 + 3 \cdot \tau^2} = \sqrt{\left(\frac{4 \cdot Fa}{\pi \cdot d_1^2}\right)^2 + 3 \cdot \left(\frac{T}{0,2 \cdot d_1^3}\right)^2} = \sqrt{\left(\frac{4,39,24}{\pi \cdot 10^2}\right)^2 + 3 \cdot \left(\frac{8913}{0,2 \cdot 10^3}\right)^2} \\ &= 77,19 \text{ MPa} < [\sigma] = \frac{340}{3} = 113,33 \text{ MPa} \end{aligned}$$

- Kiểm nghiệm trực vít về độ ổn định

Công thức kiểm nghiệm có dạng $S = \frac{Fth}{Fa} \geq [S_o]$

Trong đó S_o là hệ số an toàn về tính ổn định

Fth tải trọng giới hạn

Fa lực dọc trực

$[S_o] = 2,5 \dots 4$ – hệ số an toàn ổn định cho phép

Để xác định tải trọng giới hạn cần dựa vào độ mềm của vít

$$\delta = \mu \cdot \frac{l}{i}$$

Với $\mu = 1$ trực vít được cố định hai đầu

$l = 300 \text{ mm}$ chiều dài trực vít

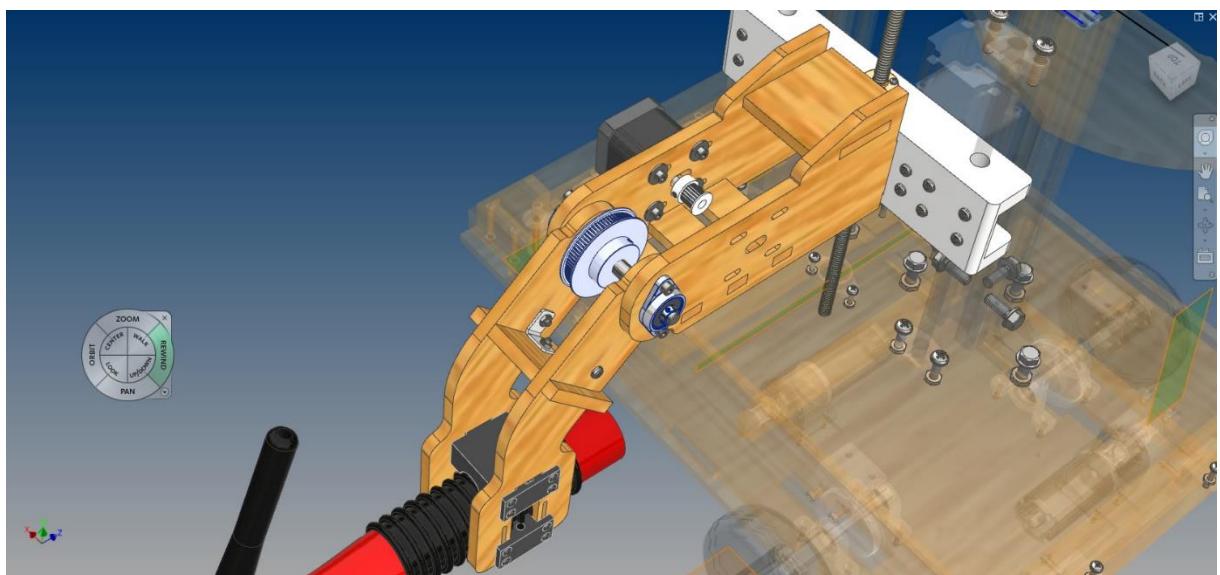
$$i = \sqrt{\frac{4J}{\pi \cdot d_1^2}} = 0,317$$

$$J = \frac{\pi \cdot d^2}{64} \left(0,4 + 0,6 \frac{d}{d_1}\right) = \frac{\pi \cdot 12^2}{64} \left(0,4 + 0,6 \cdot \frac{12}{10}\right) = 7,92 \text{ mm}^4$$

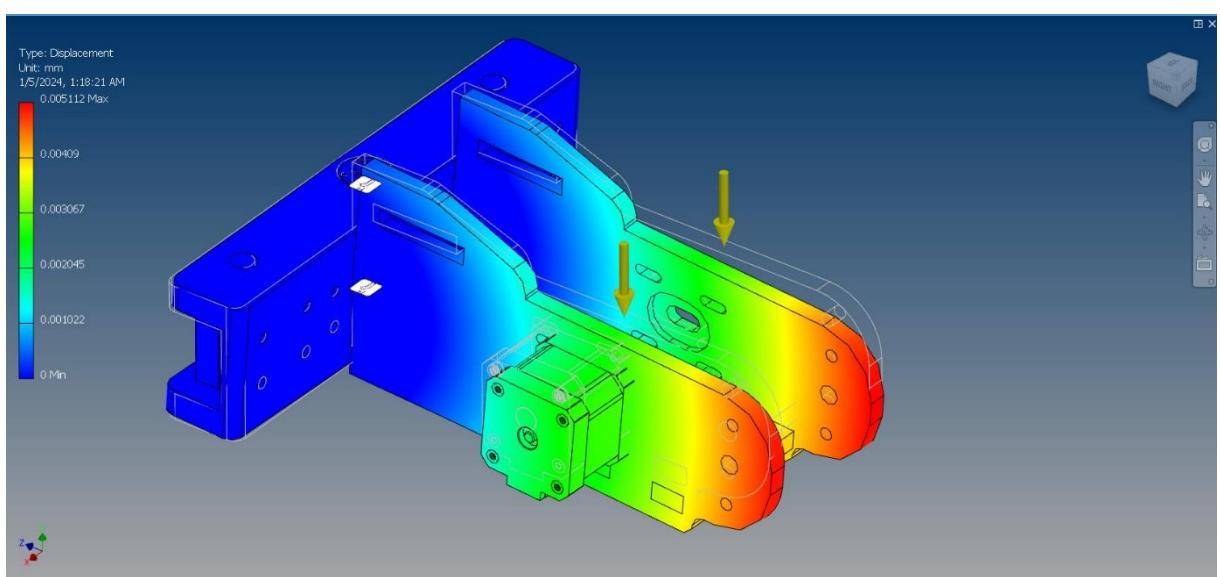
$$\delta = \mu \cdot \frac{l}{i} = 1 \cdot \frac{3}{0,317} = 9,46 < 60$$

2.3. Thiết kế cơ cấu tay hàn

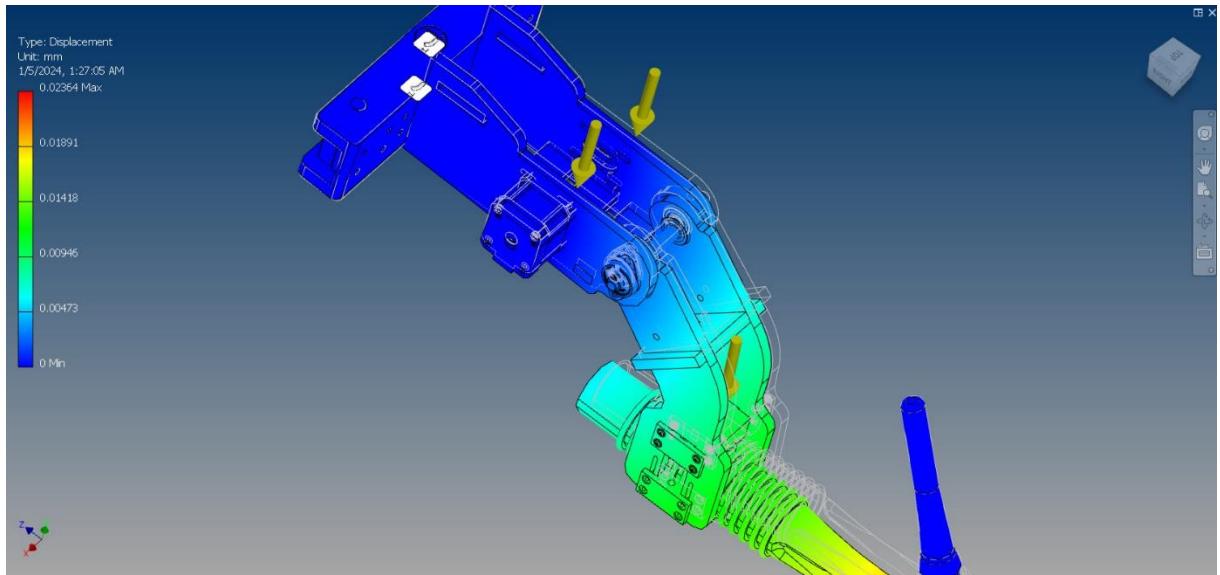
Tay hàn được thiết kế đặt trên một trục vít gắn vào nắp thân xe, gồm hai khâu: một khâu cố định gắn vuông góc với mặt phẳng trục vít để thực hiện chuyển động tịnh tiến lên xuống, một khâu có thể xoay được tạo chuyển động hàn.



Hình 2.8. Hình ảnh 3D cơ cấu tay hàn



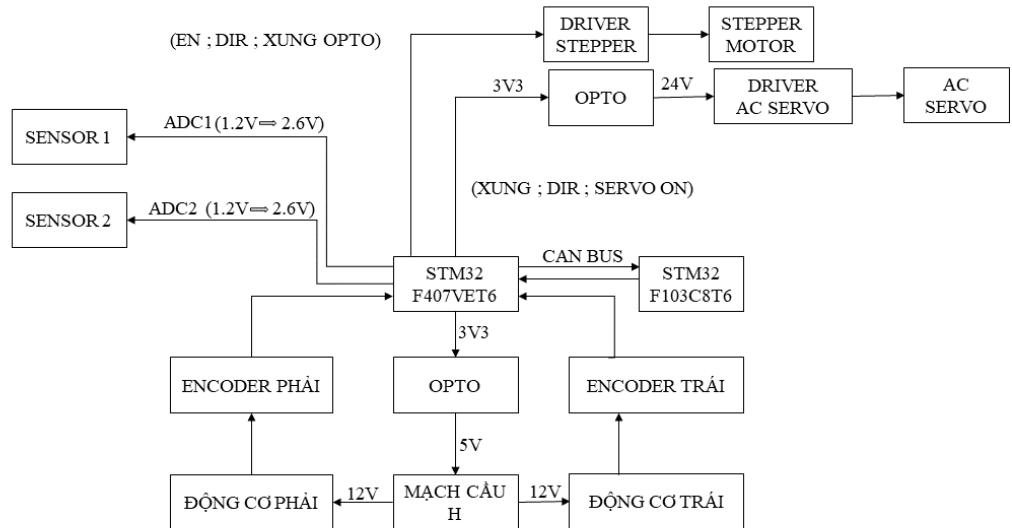
Hình 2.9. Úng suất và độ biến dạng cơ cấu tay hàn khâu tịnh tiến



Hình 2.10. Ứng suất và độ biến dạng cơ cấu tay hàn khâu xoay

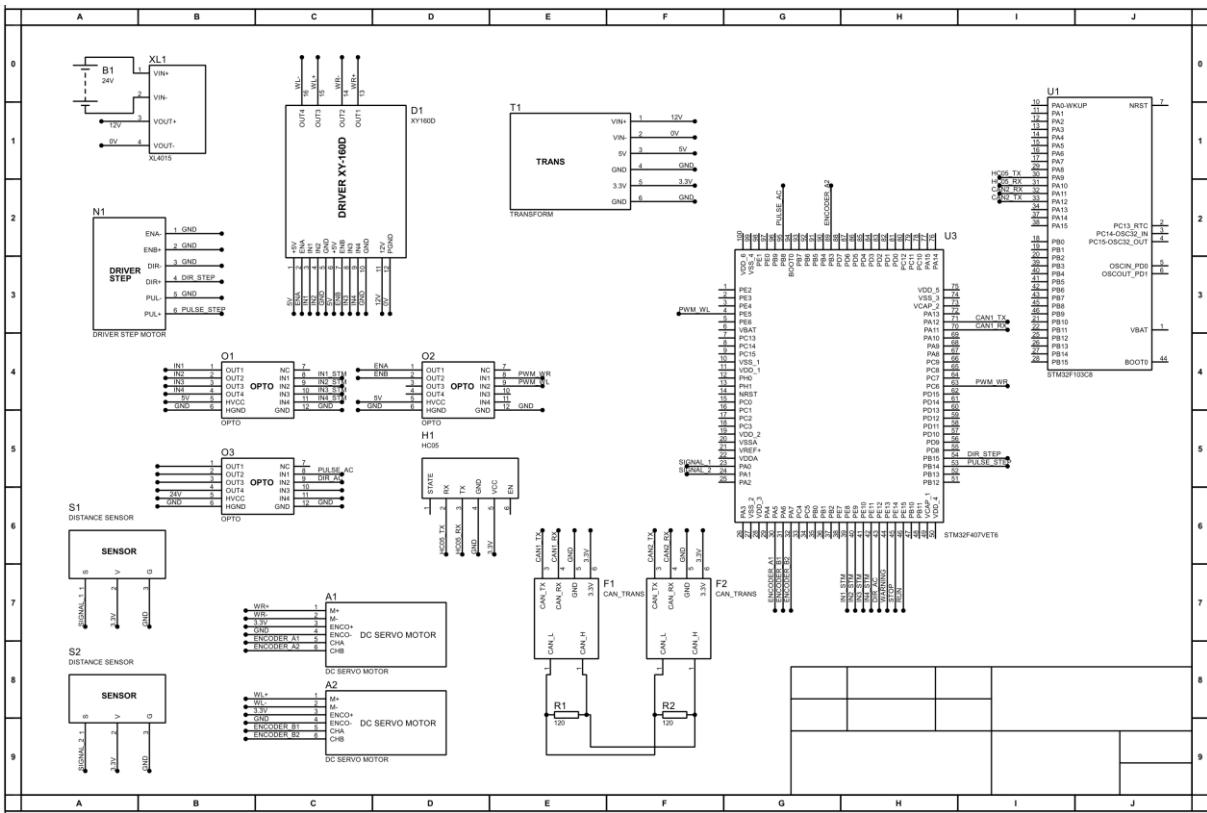
2.4. Mạch điện điều khiển

2.4.1. Sơ đồ khối mạch điện điều khiển



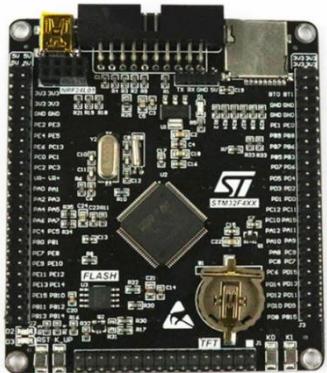
Hình 2.11. Sơ đồ khối mạch điện điều khiển

2.4.2. Mạch nguyên lý mạch điện điều khiển

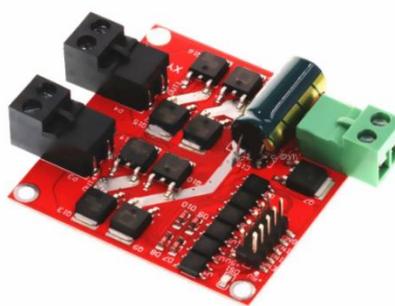


Hình 2.12. Mạch nguyên lý mạch điện điều khiển

2.5. Các thiết bị điện tử được sử dụng trong mạch điện



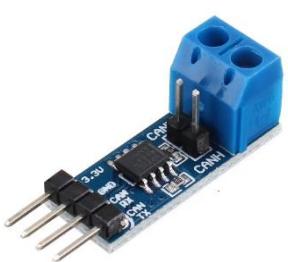
Hình 2.13



Hình 2.14



Hình 2.15



Hình 2.16



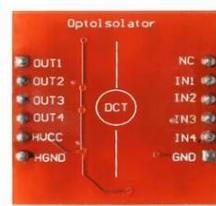
Hình 2.17



Hình 2.18



Hình 2.19



Hình 2.20



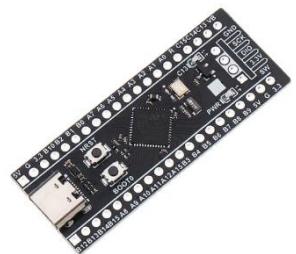
Hình 2.21



Hình 2.22



Hình 2.23



Hình 2.24

STM32 F407VET6 (hình 2.13):

- **Mô tả:** Là bộ vi xử lý chính của hệ thống, thu thập dữ liệu từ cảm biến và encoder.
- **Nhiệm vụ:** Tạo và điều khiển xung PWM để kích động cơ một cách chính xác, cũng như xử lý dữ liệu từ các cảm biến để đưa ra quyết định điều khiển phù hợp.

Driver XY 160D (hình 2.14):

- **Mô tả:** Điều khiển động cơ và nhận điện áp đầu vào 5V.
- **Nhiệm vụ:** Chuyển đổi tín hiệu và điều khiển động cơ để đảm bảo chuyển động chính xác và ổn định.

DC Encoder (hình 2.15):

- **Mô tả:** Cung cấp thông tin về vòng quay của bánh động cơ.

- **Nhiệm vụ:** Giúp xác định vị trí chính xác của động cơ, cung cấp thông tin phản hồi quan trọng cho quá trình điều khiển.

CAN (hình 2.16):

- **Mô tả:** Giao tiếp giữa hai vi điều khiển STM32F103 và STM32F407.
- **Nhiệm vụ:** Truyền thông dữ liệu giữa các vi điều khiển, đảm bảo sự đồng bộ và chia sẻ thông tin chính xác giữa các thành phần.

HC05 (hình 2.17):

- **Mô tả:** Module Bluetooth HC05 để truyền dữ liệu lên máy tính.
- **Nhiệm vụ:** Kết nối không dây và chuyển đổi dữ liệu để hiển thị trạng thái và thông tin quan trọng trên máy tính.

Mạch giảm áp 24V → 12V (hình 2.18):

- **Mô tả:** Chuyển đổi điện áp từ nguồn tổ ong 24V sang 12V.
- **Nhiệm vụ:** Đảm bảo rằng các thành phần trong hệ thống hoạt động với điện áp phù hợp và ổn định.

Mạch nguồn AMS (hình 2.19):

- **Mô tả:** Cung cấp nguồn cho cảm biến và vi điều khiển.
- **Nhiệm vụ:** Đảm bảo nguồn điện ổn định cho các thành phần quan trọng của hệ thống.

OPTO TLP281 (hình 2.20):

- **Mô tả:** Chuyển đổi logic từ 3V3 sang 5V và từ 3V3 sang 24V.
- **Nhiệm vụ:** Tạo điều kiện để các thiết bị hoạt động một cách đồng bộ và hiệu quả.

Cảm biến quang (hình 2.21):

- **Mô tả:** Cảm biến ADC để xác định vị trí lân cận.
- **Nhiệm vụ:** Cung cấp thông tin về môi trường xung quanh để đảm bảo sự an toàn và hiệu suất của hệ thống.

Driver TB6600 (hình 2.22):

- **Mô tả:** Driver kích xung cho step ở chế độ vi bước.
- **Nhiệm vụ:** Điều khiển động cơ bước với độ chính xác cao, phục vụ cho các ứng dụng yêu cầu chuyển động đặc biệt.

Stepper Motor (hình 2.23):

- **Mô tả:** Sử dụng động cơ bước cho cánh tay hoặc các chuyển động khác.
- **Nhiệm vụ:** Tạo chuyển động bước chính xác và kiểm soát được cho các ứng dụng cụ thể.

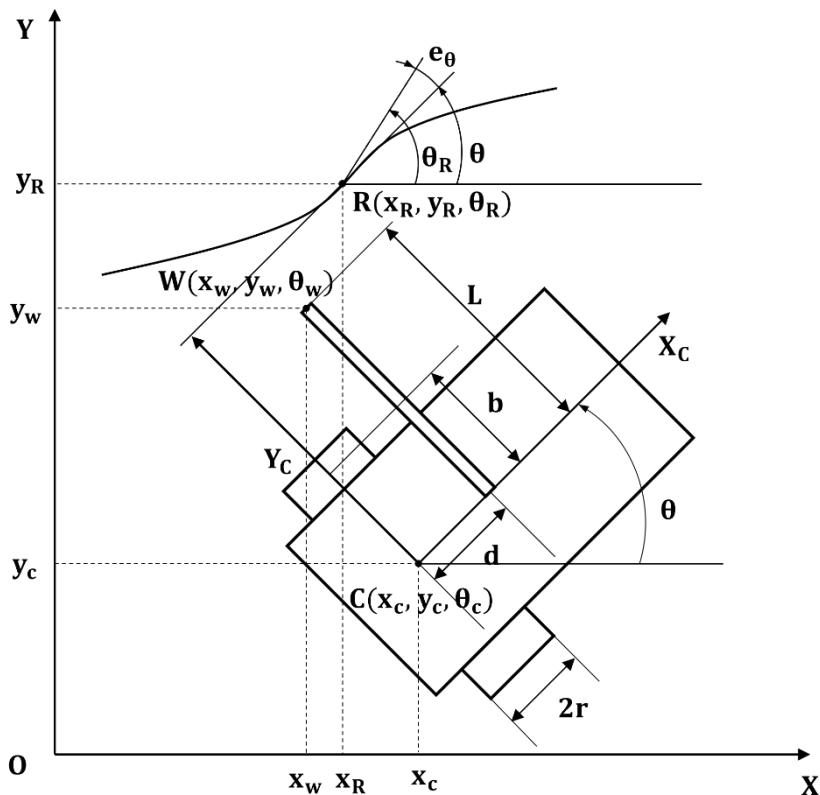
STM32F103 (hình 2.24):

- **Mô tả:** Ví xử lý trung tâm, tính toán động học và tạo PWM để truyền dữ liệu sang STM32F407.
- **Nhiệm vụ:** Đảm bảo tính toán chính xác và tạo điều kiện cho việc điều khiển chính xác của DC Servo

CHƯƠNG 3. MÔ HÌNH HÓA VÀ XÂY DỰNG HỆ THÔNG ĐIỀU KHIỂN

3.1. Động học Robot hàn di động hai bánh xe

Động học là bài toán chuyển động của robot trong không gian không tính đến sự tác động của lực đối với hệ. Động học thể hiện sự liên kết giữa các thông số điều khiển của robot với trạng thái hoạt động của robot trong không gian. Đối với robot hàn của nhóm, bài toán động học được mô hình hóa như hình 3.1 dưới đây:



Hình 3.1. Mô hình toán động học của robot hàn di động.

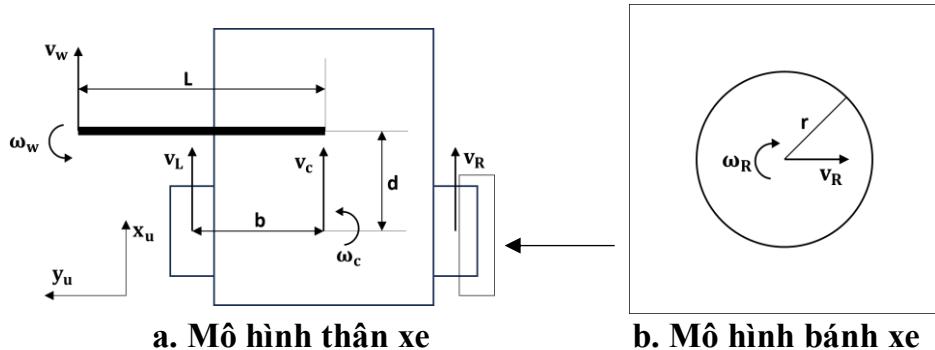
Trong đó, các thông số được giải thích trong bảng dưới đây:

Thông số	Ý nghĩa
OXY	Hệ tọa độ toàn cục trong hệ thống
CX _c Y _c	Hệ tọa độ cục bộ trong hệ thống
C(x _c , y _c)	Tọa độ tâm của robot trong hệ tọa độ toàn cục
θ	Góc quay của hệ tọa độ cục bộ so với hệ tọa độ gốc

$W(x_w, y_w)$	Tọa độ điểm hàn trong hệ tọa độ toàn cục
$\theta_w = \theta$	Góc quay của hệ tọa độ cục bộ so với hệ tọa độ gốc tại điểm W
$R(x_R, y_R)$	Tọa độ điểm hàn mong muốn trong hệ tọa độ toàn cục
θ_R	Góc nghiêng của đường hàn (góc giữa tiếp tuyến của đường hàn tại điểm hàn với trục X hệ tọa độ toàn cục)
b	Khoảng cách từ tâm robot đến bánh xe
d	Khoảng cách từ tâm robot đến điểm hàn theo trục X_C
L	Chiều dài cần hàn (có thể coi là khoảng cách từ tâm robot đến điểm hàn theo trục Y_C)
r	Bán kính bánh xe
e_θ	Sai lệch giữa góc θ và θ_R

Bảng 3.1. Bảng ý nghĩa các thông số của robot hàn di động

3.1.1. Mô hình hóa thân robot



Hình 3.2. Mô hình thân xe và bánh xe Robot hàn di động

Cấu trúc mô hình tính toán cho robot hàn di động 2 bánh xe gồm có các chi tiết sau và được thể hiện ở hình 3.2

- Mô hình thân xe (hình 3.2a):

- + Hệ trục tọa độ cục bộ của robot (Ox_Uy_U).
- + 2 bánh xe được đặt cách tâm một đoạn b (mm) với vận tốc của bánh phải và bánh trái lần lượt là v_R và v_L .

+ 1 cần hàn có chiều dài L được bố trí cách tâm theo trục x_U một đoạn có chiều dài d (mm) và vuông góc với mặt phẳng thân trước của robot, có vận tốc dài theo robot là v_w và vận tốc góc ω_w .

+ Thân robot di chuyển với vận tốc dài v_c và vận tốc góc ω_c .

- Mô hình bánh xe (hình 3.2b): được dùng để thể hiện riêng cho từng bánh (hình bên trái là bánh phải, bánh trái có thể được thể hiện tương tự), gồm có các thông số được quan tâm lần lượt là: vận tốc góc (ω_R và ω_L), bán kính bánh xe ($r_R = r_L = r$), vận tốc dài (v_R và v_L).

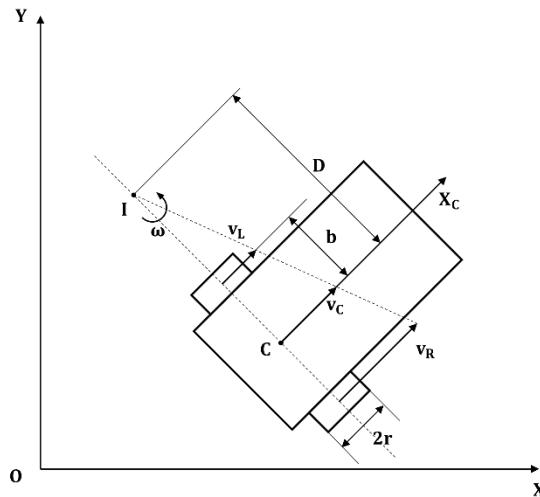
3.1.2. Phương trình động học thân robot.

3.1.2.1. Động học thuận

- Có thể thấy được rằng, với thiết kế robot hàn di động của nhóm thì robot chỉ có thể di chuyển theo trục X_C với vận tốc v_c và xoay quanh trục z (trục vuông góc với mặt phẳng hàn) với vận tốc góc ω_c . Từ đó, ta có nhận xét sau:

+ Nếu xe chạy thẳng ($\omega_c = 0$): ta có $v_c = v_R = v_L = \frac{v_R + v_L}{2}$

+ Nếu xe chạy cong ($\omega_c \neq 0$): thì xe sẽ chạy xoay một tâm I (tâm vận tốc tức thời) được xác định như hình dưới đây:



Hình 3.3. Mô hình thể hiện tâm vận tốc tức thời khi xe chuyển động cong

- Từ hình 3.3, ta có thể tính được các vận tốc tại tâm xe, bánh trái, bánh phải theo vận tốc góc ω , ở đây có thể giả sử khung xe cứng tuyết đối (vị trí các điểm trên xe không

thay đổi trong quá trình chuyển động) cho nên 3 điểm trên xe kể trên đều có cùng vận tốc góc ω . Khi đó ta có:

$$v_L = \omega \cdot (D - b) \quad (3.1)$$

$$v_C = \omega \cdot D \quad (3.2)$$

$$v_R = \omega \cdot (D + b) \quad (3.3)$$

- Có thể thấy khi ta lấy tổng và hiệu hai phương trình (3.3) và (3.1), ta được các kết quả sau:

$$v_R + v_L = 2 \cdot \omega \cdot D = 2v_C \quad (3.4)$$

$$v_R - v_L = 2 \cdot \omega \cdot b \quad (3.5)$$

- Từ 2 phương trình (3.4) và (3.5) kết hợp với sự tương đồng các tham số mô hình tính thân xe hình 3.2a, ta có được mối quan hệ liên quan giữa vận tốc góc và vận tốc dài của xe đối với vận tốc dài tại tâm bánh xe:

$$\begin{cases} \omega_c = \frac{v_R - v_L}{2b} \\ v_c = \frac{v_R + v_L}{2} \end{cases} \quad (3.6)$$

- Với hai bánh xe có cùng bán kính, ta có công thức tính vận tốc dài tại tâm bánh xe theo mô hình bánh xe (hình 3.2b):

$$\begin{cases} v_R = r \cdot \omega_R \\ v_L = r \cdot \omega_L \end{cases} \quad (3.7)$$

- Từ các hệ phương trình (3.6) và (3.7), ta có được mối liên hệ giữa vận tốc góc và vận tốc dài tại tâm xoay của xe với vận tốc góc của từng bánh xe:

$$\begin{cases} \omega_c = \frac{r \cdot (\omega_R - \omega_L)}{2b} \\ v_c = \frac{r \cdot (\omega_R + \omega_L)}{2} \end{cases} \quad (3.8)$$

- Từ hệ phương trình (3.8) kết hợp với mô hình tính động học robot hàn di động (hình 3.1), ta có thể xác định sự thay đổi của điểm C trong tọa độ toàn cục như sau:

$$\begin{cases} \dot{x}_c = v_c \cdot \cos(\theta) \\ \dot{y}_c = v_c \cdot \sin(\theta) \\ \dot{\theta}_c = \omega_c \end{cases} \quad (3.9)$$

- Sau đó, ta tích phân hệ phương trình (3.9) để có thể thu được vị trí của điểm C trên hệ tọa độ toàn cục:

$$\begin{cases} x_c = \int_t^{t+dt} v_c \cdot \cos(\theta(t)) \cdot dt \\ y_c = \int_t^{t+dt} v_c \cdot \sin(\theta(t)) \cdot dt \\ \theta_c = \int_t^{t+dt} \omega_c(t) \cdot dt \end{cases} \quad (3.10)$$

- Từ hình 3.1, ta xác định mối quan hệ giữa tọa độ điểm hàn và tọa độ tâm C trên hệ trực toàn cục như sau:

$$\begin{cases} x_w = x_c + d \cdot \cos(\theta) - L \cdot \sin(\theta) \\ y_w = y_c + d \cdot \sin(\theta) + L \cdot \cos(\theta) \\ \theta_w = \theta_c \end{cases} \quad (3.11)$$

Vậy từ các hệ phương trình (3.8), (3.10) và (3.11), ta đã tính toán bài toán động học thuận. Bài toán động học thuận cung cấp cho ta mối liên hệ từ vận tốc góc của bánh xe đến trạng thái của robot trên hệ tọa độ toàn cục.

3.1.2.2. Động học nghịch

- Từ hệ phương trình (3.8) ở bài toán động học thuận, ta biến đổi để đưa về công thức tính vận tốc góc theo vận tốc dài và vận tốc góc tại tâm xoay của xe:

$$\begin{cases} \omega_R = \frac{v_c + b \cdot \omega_c}{r} \\ \omega_L = \frac{v_c - b \cdot \omega_c}{r} \end{cases} \quad (3.12)$$

- Bên cạnh đó, từ mô hình tính toán động học (hình 3.1), giả sử chúng ta đã biết tọa độ điểm hàn mong muốn $R(x_R, y_R)$ và góc nghiêng đường hàn θ_R (trên thực tế tọa độ này được tính toán dựa trên dữ liệu cảm biến trả về), khi đó gọi e_d là sai số khoảng cách

giữa điểm hàn mong muốn và điểm hàn hiện tại trên hệ tọa độ toàn cục, ta có thể xác định các sai số cần thiết cho việc điều khiển.

$$e_d = \sqrt{(y_R - y_w)^2 + (x_R - x_w)^2} \quad (3.13)$$

$$e_\theta = \theta_R - \theta \quad (3.14)$$

Sau khi đã có được cái sai số cần thiết – sai số khoảng cách (e_d) và sai số góc (e_θ), thông qua luật điều khiển được xây dựng ta có thể xác định được vận tốc dài (v_c) và vận tốc góc (ω_c) của xe. Sau đó, áp dụng hệ phương trình (3.12) ta có thể xác định vận tốc góc của từng bánh xe (ω_R và ω_L).

3.2. Giải thuật điều khiển

3.2.1. Tổng quan sơ đồ điều khiển.

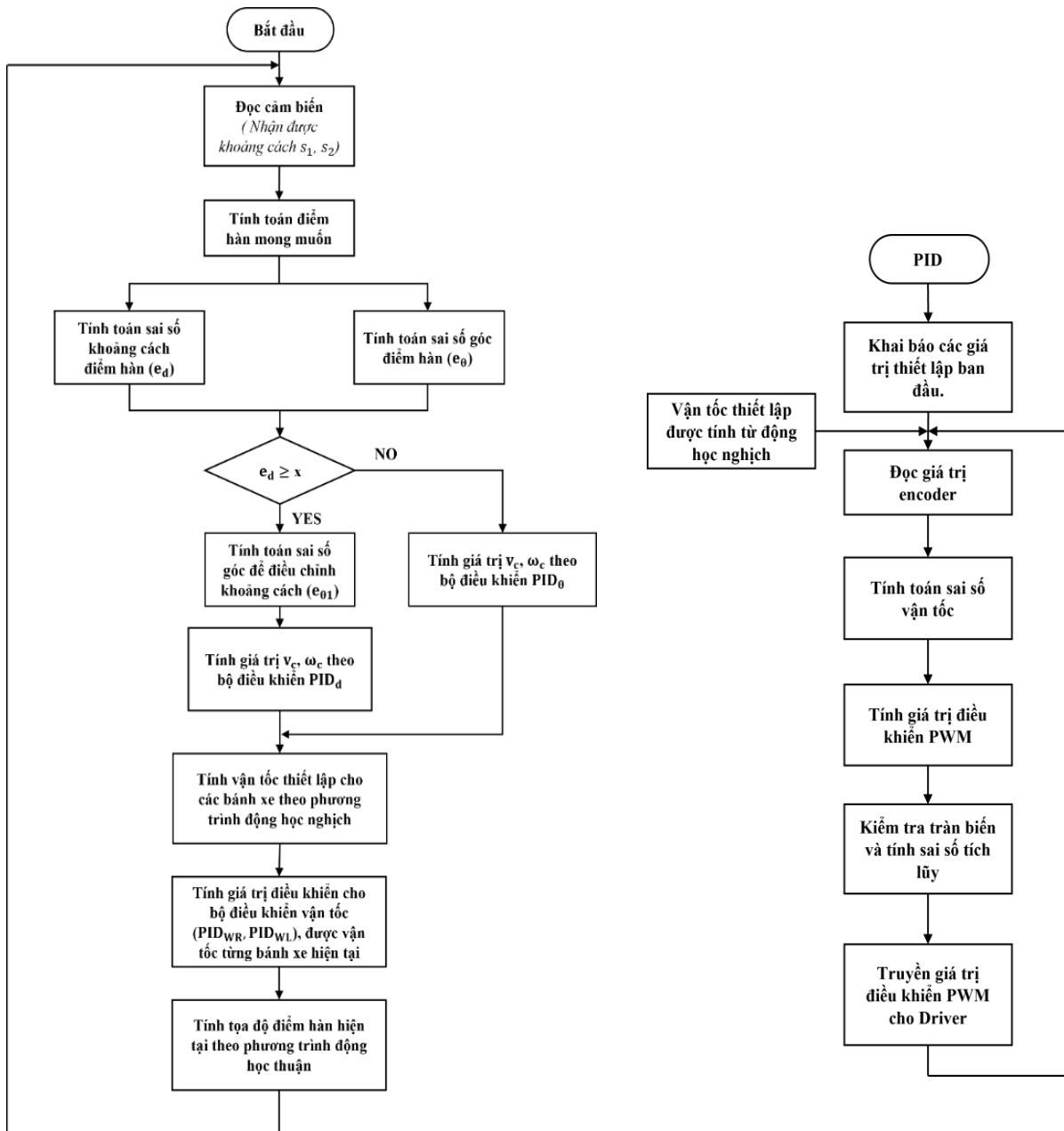
Để có thể điều khiển được robot di chuyển bám theo điểm hàn thì hệ thống điều khiển cần thực hiện hai việc:

- Thứ nhất: Đọc giá trị cảm biến trả về từ đó xác định các thông số trạng thái của robot (tọa độ vị trí của xe, tọa độ điểm hàn mong muốn, góc hiện tại xe, góc của điểm hàn, sai số cần thiết, ...), sau đó tính toán ra các tham số điều khiển trạng thái robot (vận tốc dài v_c và vận tốc góc ω_c) thông qua bộ điều khiển (trong đề tài này nhóm chọn sử dụng bộ điều khiển PID). Khi đã có được các tham số, tính toán ra vận tốc của từng bánh xe theo phương trình động học nghịch. Dùng vận tốc tính toán ra làm vận tốc thiết lập cho bộ điều khiển vận tốc để tìm được vận tốc hiện tại của từng bánh, Cuối cùng ta xác định giá trị tọa độ điểm hàn và liên tục lặp lại quá trình thực hiện này sau một khoảng thời gian t.

- Thứ hai: Điều khiển vận tốc của từng bánh xe thông qua bộ điều khiển động cơ (Trong đề tài này, nhóm chọn sử dụng bộ điều khiển PID). Quá trình thực hiện như sau: Đầu tiên, ta thiết lập các thông số ban đầu, đồng thời có được vận tốc thiết lập cho bộ điều khiển vận tốc. Sau đó, ta thực hiện đọc giá trị encoder trả về và tính được vận tốc hiện tại của bánh xe. Tiếp đến, ta tính toán sai số vận tốc và thông qua bộ điều khiển PI

ta có được giá trị điều khiển PWM cho động cơ. Và quá trình này cũng cần được thực hiện lại sau một khoảng thời gian t_1 .

8 Vậy nên trong chương trình điều khiển, ta cần phải sử dụng hai vòng lặp: Một vòng lặp thực hiện tính toán điều khiển cho hệ thống (lưu đồ hình 3.4) và một vòng lặp con thực hiện điều khiển tốc độ của động cơ (lưu đồ hình 3.5)



Hình 3.4. Lưu đồ giải thuật thuật toán điều khiển chính cho hệ thống

Hình 3.5. Lưu đồ giải thuật thuật toán điều khiển động cơ

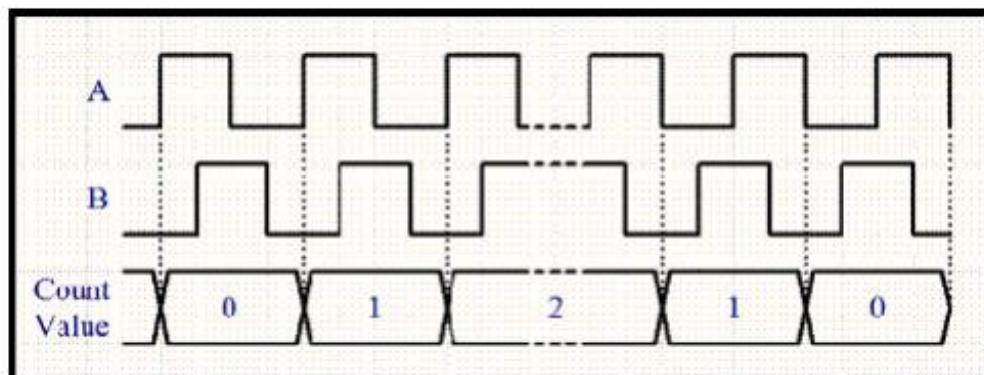
3.2.2. Bộ điều khiển động cơ

3.2.2.1. Giải thuật đọc encoder

Encoder là một bộ cảm biến được gắn vào một vật thể quay (chẳng hạn như trực hoặc động cơ) dùng để đo chuyển động quay. Với việc đo được chuyển động quay, chúng ta có thể xác định được bất kỳ độ dịch chuyển, vận tốc, gia tốc và góc của vật thể quay.

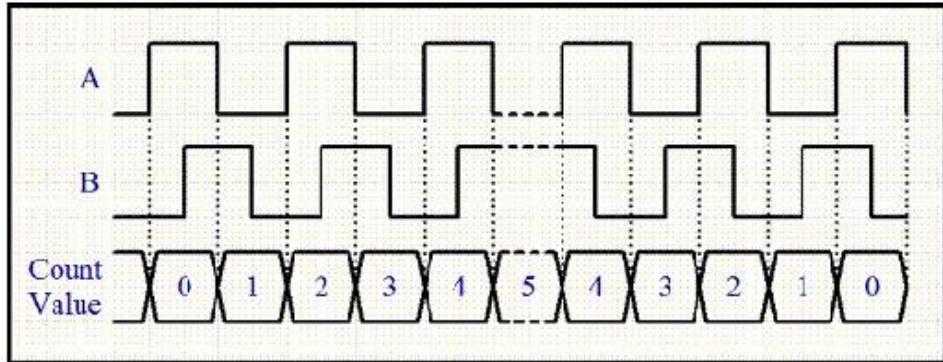
Encoder gồm 2 kênh đặt lệch pha nhau (thông thường lệch nhau 90°) để có xác định chiều quay và số xung hiện tại của động cơ. Giả sử động cơ quay ngược chiều kim đồng hồ thì kênh A sớm pha hơn kênh B, khi đó ta có các chế độ đọc xung encoder như sau:

Chế độ 1X: Chỉ đọc xung trên một cạnh của kênh A. Chiều của động cơ được xác định theo các trường hợp sau: khi kênh A sớm pha hơn kênh B (quay ngược chiều kim đồng hồ) sự tăng xung diễn ra ở cạnh lên của kênh A, khi kênh B sớm pha hơn kênh A (quay cùng chiều kim đồng hồ) sự giảm xung diễn ra ở cạnh xuống của kênh A.



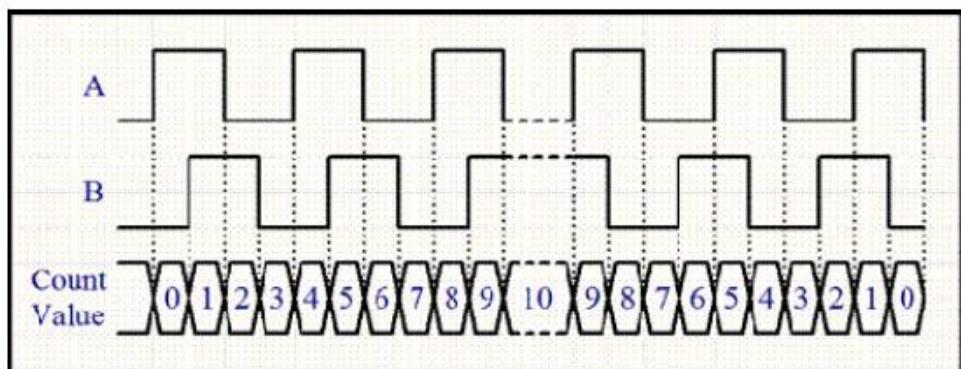
Hình 3.6. Biểu đồ xung chế độ 1X

Chế độ 2X: Đọc xung trên cả hai cạnh của kênh A. Chiều quay của động cơ được xác định dựa vào trạng thái của kênh B tương ứng với mỗi xung của kênh A.

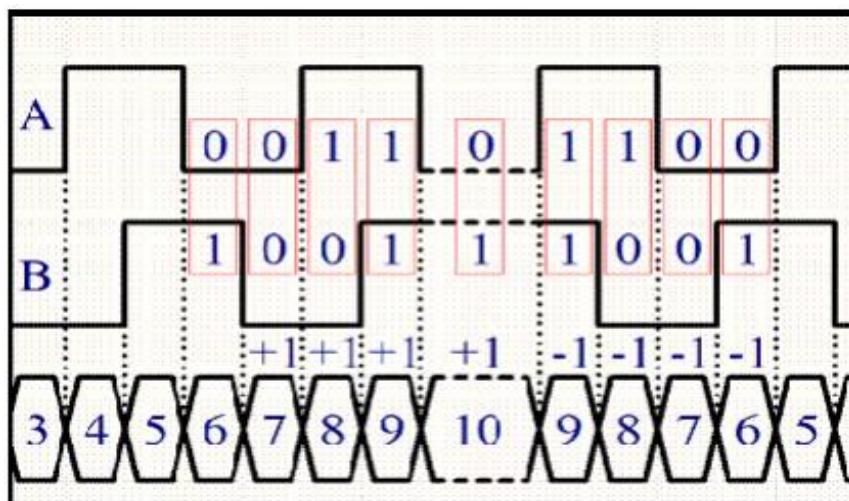


Hình 3.7. Biểu đồ xung chế độ 2X

Chế độ 4X: Đọc cạnh lên và cạnh xuống của cả 2 kênh A và B. Vậy nên, đây là chế độ cho độ phân giải encoder cao nhất. Chiều quay của động cơ phụ thuộc vào trạng thái của xung encoder hiện tại so với xung encoder trước đó.

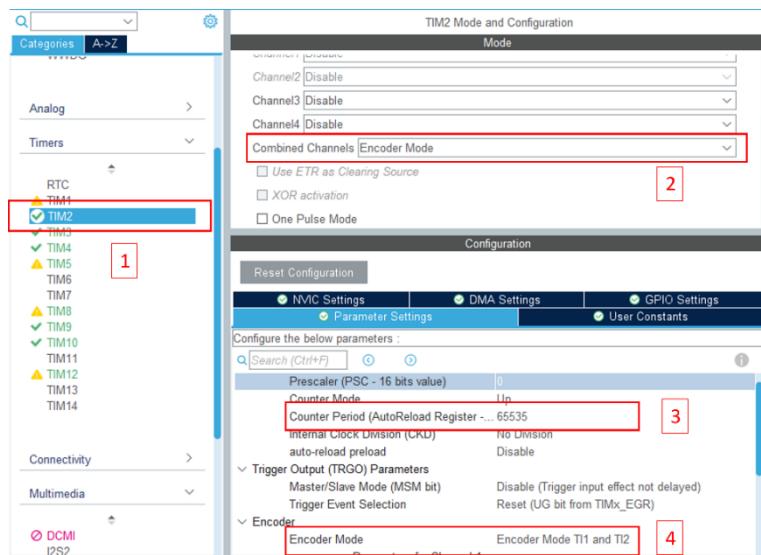


Hình 3.8. Biểu đồ xung chế độ 4X



Hình 3.9. Biểu đồ xung chế độ 4X có thể hiện tương quan giữa trạng thái xung và chiều quay động cơ

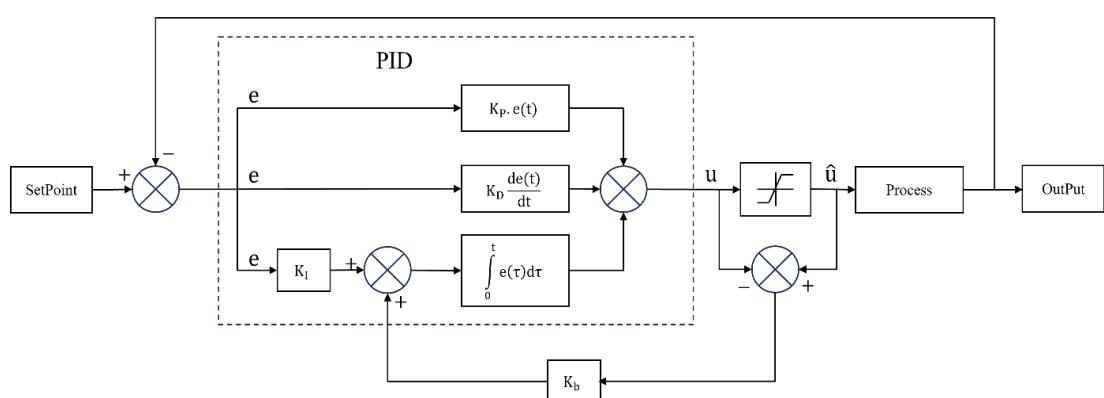
Vậy theo lý thuyết, khi đọc xung encoder bằng vi điều khiển ta sẽ sử dụng tới 2 ngắt ngoài để bắt cạnh của 2 kênh A và kênh B, điều này sẽ gây khó khăn tới thời gian xử lý và tài nguyên bộ nhớ của chúng ta khi thực hiện với những dạng Encoder có PPR lớn. Vậy nên nhóm quyết định sử dụng chế độ đọc encoder có sẵn trên Timer 2 và Timer 3 của STM32, khi đọc bằng chế độ này với thiết lập đọc trên cả TI1 và TI2, ta sẽ mặc định đọc chế độ 4X. Ngoài ra khi sử dụng chế độ này, chúng ta sẽ được tự động xử lý thông tin về tăng giảm số xung theo chiều của động một cách độc lập với các quá trình xử lý tác vụ khác trong CPU.



Hình 3.10.Thiết lập chế độ đọc encoder trong thanh ghi Timer 2

3.2.2.2. Bộ điều khiển PID cho động cơ

* Giới thiệu về bộ điều khiển pid có khâu anti windup



Hình 3.11.Sơ đồ thuật toán điều khiển PID có sử dụng Anti-windup

PID, viết tắt của Proportional Integral Derivative hay còn gọi là bộ điều khiển tích phân tỉ lệ, là một cơ chế phản hồi vòng điều khiển tổng quát được sử dụng rộng rãi trong các hệ thống điều khiển phản hồi công nghiệp hiện nay. Vì là hệ điều khiển có phản hồi nên bộ điều khiển PID có khả năng tính toán giá trị sai số giữa giá trị ngõ ra hiện tại với giá trị thiết lập, sau đó thực hiện giảm sai số bằng cách điều chỉnh giá trị điều khiển đầu vào.

Bộ điều khiển PID gồm 3 khâu: Khâu tỉ lệ P (Proportional), khâu tích phân I (Integral), khâu vi phân D (Derivative).

Thông số đầu vào của bộ điều khiển là sai số của mục tiêu, kết quả ngõ ra của bộ điều khiển là giá trị điều khiển mục tiêu (đối với hình 3.11, e là thông số đầu vào, u là kết quả ngõ ra. Đối với động cơ, sai số vận tốc bánh xe là thông số đầu vào, thông số PWM là kết quả ngõ ra). Kết quả ngõ ra của bộ điều khiển là tổng kết quả của các khâu thành phần.

$$u = P_{out} + I_{out} + D_{out} \quad (3.15)$$

Khâu tỉ lệ làm thay đổi giá trị đầu ra, tỉ lệ với giá trị sai số hiện tại. Đáp ứng tỉ lệ có thể được điều chỉnh bằng cách nhân sai số đó với một hằng số K_P , được gọi là hệ số tỉ lệ.

$$P_{out} = K_P e(t) \quad (3.16)$$

Trong đó:

- P_{out} : Thừa số tỉ lệ của đầu ra
- K_P : Hệ số tỉ lệ
- $e(t)$: sai số tức thời

Khâu tích phân là tích giữa tổng sai số tức thời theo thời gian với một hệ số độ lợi tích phân K_I . Khâu tích phân (khi cộng thêm khâu tỉ lệ) sẽ tăng tốc chuyển động của quá trình tới điểm đặt và khử số dư sai số ổn định với một tỉ lệ chỉ phụ thuộc vào bộ điều khiển. Tuy nhiên, vì khâu tích phân là đáp ứng của sai số tích lũy trong quá khứ, nó có thể khiến giá trị hiện tại vọt lố qua giá trị đặt.

$$I_{out} = K_I \times \int_0^t e(\tau) d\tau \quad (3.17)$$

Trong đó:

- I_{out} : Thừa số tích phân của đầu ra
- K_I : Hệ số độ lợi tích phân
- e : sai số
- τ : biến tích phân trung gian

Khâu vi phân phản là tích giữa độ dốc sai số theo thời gian t và độ vi lợi K_D , điều khiển vi phân được sử dụng để làm giảm biên độ vọt lố được tạo ra bởi thành phần tích phản và tăng cường độ ổn định của bộ điều khiển hỗn hợp. Tuy nhiên, phép vi phân của một tín hiệu sẽ khuếch đại nhiều và do đó khâu này sẽ nhạy hơn đối với nhiều trong sai số, và có thể khiến quá trình trở nên không ổn định nếu nhiều và độ lợi vi phân đủ lớn.

$$D_{out} = K_D \times \frac{de(t)}{dt} \quad (3.18)$$

Trong đó:

- D_{out} : Thừa số vi phân của đầu ra
- K_D : Hệ số độ lợi tích phản
- e : sai số

Trên thực tế khi chúng ta sử dụng bộ điều khiển PID để điều khiển, ta cần thêm khâu bão hòa để giới hạn giá trị đầu ra vì tín hiệu đầu ra để điều khiển thiết bị thông thường luôn có một giới hạn nhất định, xét mô hình 3.11, ta nhận thấy kết quả ngõ ra u hoàn toàn có thể vô cùng lớn nên ta phải giới hạn lại kết quả thành \hat{u} trong một khoảng giá trị phù hợp (HLIM, LOLIM). Tuy nhiên, khi sử dụng khâu bão hòa chúng ta sẽ bị vướng vào một vấn đề đó chính là ở khâu I sẽ tích lũy sai số lên rất lớn do khi u tính ra quá lớn và nhờ khâu bão hòa ta có \hat{u} phù hợp với thiết bị nhưng lại tạo ra sai số khi ta hồi tiếp về nên các sai số đó sẽ được khâu I tích lũy khiến u tính ra sẽ ngày càng lớn, do đó khi chúng ta đột ngột thay đổi giá trị thiết lập thì kết quả u sẽ mất một khoảng thời gian dài

để có thể quay trở lại đúng giá trị đáp ứng. Để giải quyết vấn đề này ta sử dụng khâu Anti-windup nhằm ngăn chặn sự tích lũy của giá trị tích phân trong khâu I khi sử dụng khâu bão hòa.

Từ mô hình 3.11, ta có thể xác định lại được công thức cho khâu I là:

$$I_{out} = \int_0^t [K_I e(\tau) + K_B e_{reset}(\tau)] d\tau \quad (3.19)$$

Trong đó:

- I_{out} : Thừa số tích phân của đầu ra
- K_I : Hệ số độ lợi tích phân
- e : sai số giá trị thiết lập với giá trị thực
- e_{reset} : sai số giá trị ù và u
- τ : biến tích phân trung gian

Từ công thức (3.15), (3.16), (3.18), (3.19), ta có kết quả ngõ ra trong miền thời gian t là:

$$u(t) = K_p e(t) + \int_0^t [K_I e(\tau) + K_B e_{reset}(\tau)] d\tau + K_D \frac{de(t)}{dt} \quad (3.20)$$

Trong thực tế ta cần đưa bộ điều khiển PID về hệ rời rạc để thực tế hệ thống số hóa. Với chu kỳ thời gian lấy mẫu T. Từ đó ta có ngõ ra tác động $u(kT)$:

$$u(kT) = K_p e(kT) + \sum_0^k [K_I e(k) + K_B e_{reset}(k)] T + K_D \frac{e(kT) - e[(k-1)T]}{T} \quad (3.21)$$

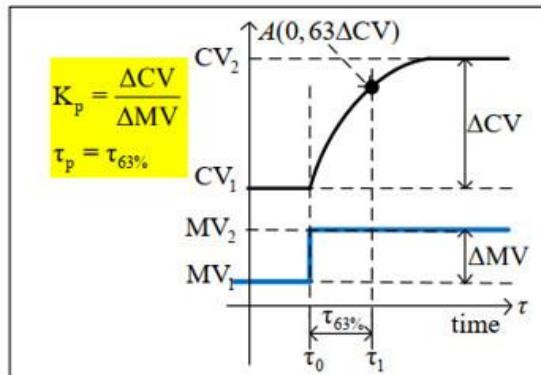
3.2.2.3. Phương pháp xác định thông số bộ điều khiển

Trong đồ án này, nhóm chúng em thực hiện xác định thông số bộ điều khiển bằng phương pháp IMC, gồm có các bước sau:

* Động cơ 1

Bước 1: Xác định hàm truyền của động cơ.

Bằng cách thiết lập cho động cơ chạy vận tốc tối đa khi động cơ đang đứng yên ta thu được đồ thị biểu diễn quá trình tăng trưởng của vận tốc



Hình 3.12. Hình minh họa đồ thị quá trình đáp ứng vận tốc của động cơ



Hình 3.13. Đồ thị quá trình đáp ứng vận tốc của động cơ

Từ đồ thị, ta có các thông số sau:

- Thời điểm bắt đầu đáp ứng: $\tau_1 = 18,294$
- Thời điểm vận tốc đạt 63,2% vận tốc tối đa: $\tau_2 = 18,61398988$
- Vận tốc tối đa động cơ đạt được: $v_{\max} = 1,74$ (rad/s)
- Thời gian trễ: $\theta = 0,085$

Khi đó, ta có thể tính được hệ số K của động cơ là:

$$K = \frac{v_{\max} - v_{\min}}{999} = \frac{1,74 - 0}{999} = 1,741742 \cdot 10^{-3} \quad (3.22)$$

Thời gian để động cơ vận tốc bằng 63,2% vận tốc tối đa là:

$$\tau = \tau_2 - \tau_1 = 18,61398988 - 18,294 = 0,31998988 \text{ (s)} \quad (3.23)$$

Ta có hàm truyền của động cơ 1:

$$G(s) = \frac{K \cdot e^{-\theta}}{\tau s + 1} = \frac{1,741742 \cdot 10^{-3} \cdot e^{-0,085}}{0,31998988 s + 1} \quad (3.24)$$

Bước 2: Tìm thông số bộ điều khiển PI

Từ đó, ta chọn được:

$$\tau_C = 0,95 \cdot \tau = 0,303990386 \text{ (s)} \quad (3.25)$$

Dựa theo luật định chỉnh IMC-PID của CHIEN và FRUEHAUF, ta xác định thông số của bộ PID như sau:

$$\tau_I = \tau + 0,5\theta = 0,31998988 + 0,5 \cdot 0,085 = 0,36248988 \text{ (s)} \quad (3.26)$$

$$K_P = K_C = \frac{\tau_I}{K \cdot (\tau_C + 0,5\theta)} \quad (3.27)$$

$$= \frac{0,36248988}{1,741742 \cdot 10^{-3} \cdot (0,303990386 + 0,5 \cdot 0,085)} = 600,6567457$$

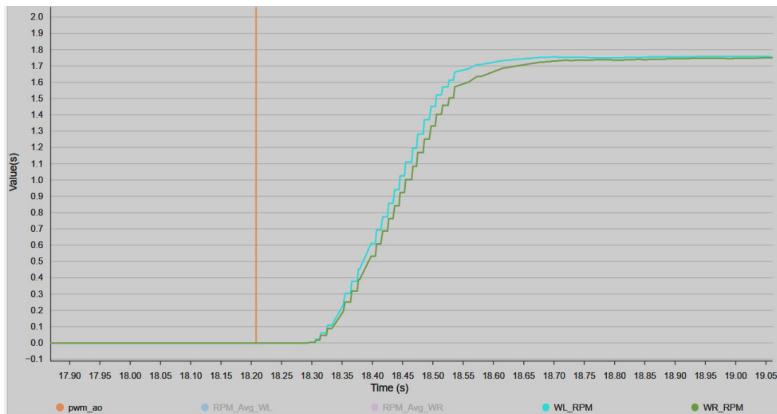
$$K_I = \frac{K_P}{\tau_I} = \frac{600,6567457}{0,36248988} = 1657,030386 \quad (3.28)$$

$$K_b = \frac{1}{\tau_I} = \frac{1}{0,36248988} = 2,758697705 \quad (3.29)$$

Bước 3: Sử dụng thông số PI thu được quan sát thay đổi tốc độ của động cơ xem có ổn định không, nếu không ta tiến hành tăng giảm giá trị K_P và K_I sao cho đáp ứng chính xác nhất có thể.

* Động cơ 2

Bước 1: Xác định hàm truyền của động cơ.



Hình 3.14. Đồ thị quá trình đáp ứng vận tốc của động cơ

Từ đồ thị, ta có các thông số sau:

- Thời điểm bắt đầu đáp ứng: $\tau_1 = 18,294$
- Thời điểm vận tốc đạt 63,2% vận tốc tối đa: $\tau_2 = 18,621848$
- Vận tốc tối đa động cơ đạt được: $v_{\max} = 1,775$ (rad/s)
- Thời gian trễ: $\theta = 0,085$

Khi đó, ta có thể tính được hệ số K của động cơ là:

$$K = \frac{v_{\max} - v_{\min}}{999} = \frac{1,775 - 0}{999} = 1,776776777 \cdot 10^{-3} \quad (3.30)$$

Thời gian để động cơ vận tốc bằng 63,2% vận tốc tối đa là:

$$\tau = \tau_2 - \tau_1 = 18,621848 - 18,294 = 0,327848 \text{ (s)} \quad (3.31)$$

Ta có hàm truyền của động cơ 1:

$$G(s) = \frac{K \cdot e^{-\theta}}{\tau s + 1} = \frac{1,776776777 \cdot 10^{-3} \cdot e^{-0,085}}{0,327848 s + 1} \quad (3.32)$$

Bước 2: Tìm thông số bộ điều khiển PI

Từ đó, ta chọn được:

$$\tau_C = 0,95 \cdot \tau = 0,3114556 \text{ (s)} \quad (3.33)$$

Dựa theo luật định chỉnh IMC-PID của CHIEN và FRUEHAUF, ta xác định thông số của bộ PID như sau:

$$\tau_I = \tau + 0,5\theta = 0,327848 + 0,5 \cdot 0,085 = 0,370348 \text{ (s)} \quad (3,34)$$

$$K_P = K_C = \frac{\tau_I}{K \cdot (\tau_C + 0,5\theta)}$$

$$= \frac{0,370348}{1,776776777 \cdot 10^{-3} \cdot (0,3114556 + 0,5 \cdot 0,085)} = 588,8820908$$

$$K_I = \frac{K_P}{\tau_I} = \frac{588,88209087}{0,370348} = 1590,077686$$

$$K_b = \frac{1}{\tau_I} = \frac{1}{0,36248988} = 2,758697705$$

Bước 3: Sử dụng thông số PI thu được quan sát thay đổi tốc độ của động cơ xem có ổn định không, nếu không ta tiến hành tăng giảm giá trị K_p và K_i sao cho đáp ứng chính xác nhất có thể.

Tóm lại, chúng ta đã có

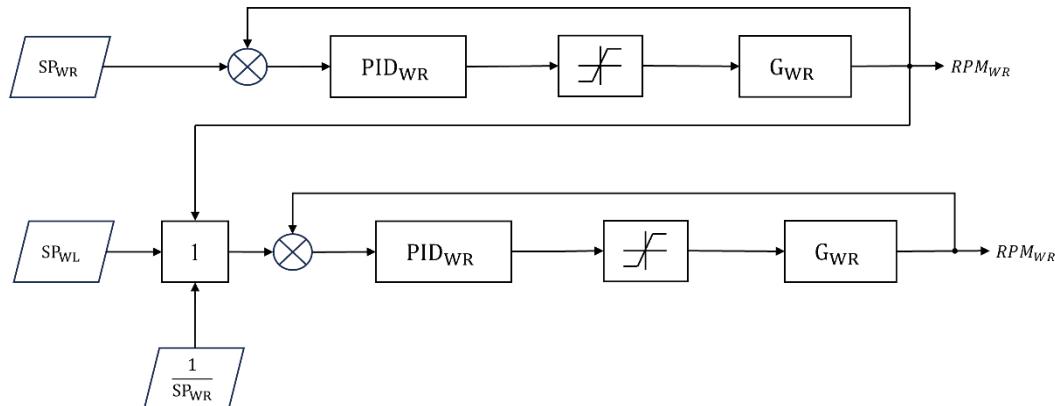
3.2.2.4. Phương pháp giảm sai lệch điều khiển giữa hai bánh xe.

Trên thực tế khi điều khiển hai bánh xe độc lập với nhau, việc để xe có thể hoàn toàn chạy thẳng là điều khá khó khăn bởi vì nhiều lý do khác nhau: Đối với tác nhân bên ngoài, có thể do lực từ tải tác dụng lên mỗi bánh xe không đều dẫn đến khả năng đáp ứng của mỗi bánh xe sẽ khác nhau tạo ra sự sai lệch hay do tín hiệu từ cảm biến trả về không ổn định, còn đối với bên trong hệ thống, thông thường đến từ việc điều khiển hai bánh xe độc lập tức là sử dụng hai động cơ riêng biệt cho từng bánh, mà mỗi động cơ có thông số kỹ thuật là khác nhau nên sẽ dẫn đến sự sai lệch trong điều khiển, điều này dễ dàng xảy ra hơn khi động cơ chạy ở vận tốc rất lớn (gần đạt vận tốc cực đại) vì khi đó giá trị điều khiển (PWM) sẽ xuất ra rất lớn và có thể đạt trạng thái bão hòa nhưng vì trạng thái bão hòa đáp ứng vận tốc của mỗi động cơ là khác nhau nên dẫn đến sai lệch vận tốc khi điều khiển chạy xiên qua một phía, bên cạnh đó, với việc điều khiển vận tốc bằng bộ điều khiển PID thì thông thường sẽ vẫn có sai lệch trong một khoảng cho phép, tuy nhiên điều này vẫn sẽ gây sai lệch vận tốc khi động cơ sai lệch cạnh trên và động cơ sai lệch cạnh dưới.

Đối với mỗi tác nhân gây sai lệch, thì phương án giải quyết sẽ khác nhau, đối với tác nhân bên ngoài có thể sử dụng thực nghiệm kiểm tra được sự sai lệch trong giữa hai bánh xe trong quá trình di chuyển để đưa ra hệ số tỉ lệ giữa hai bánh nhằm cân bằng lại sai lệch. Ở phần này, nhóm chúng em hướng đến giảm sai lệch đối với tác nhân bên trong bằng cách cho một bánh xe chạy bám theo bánh xe còn lại.

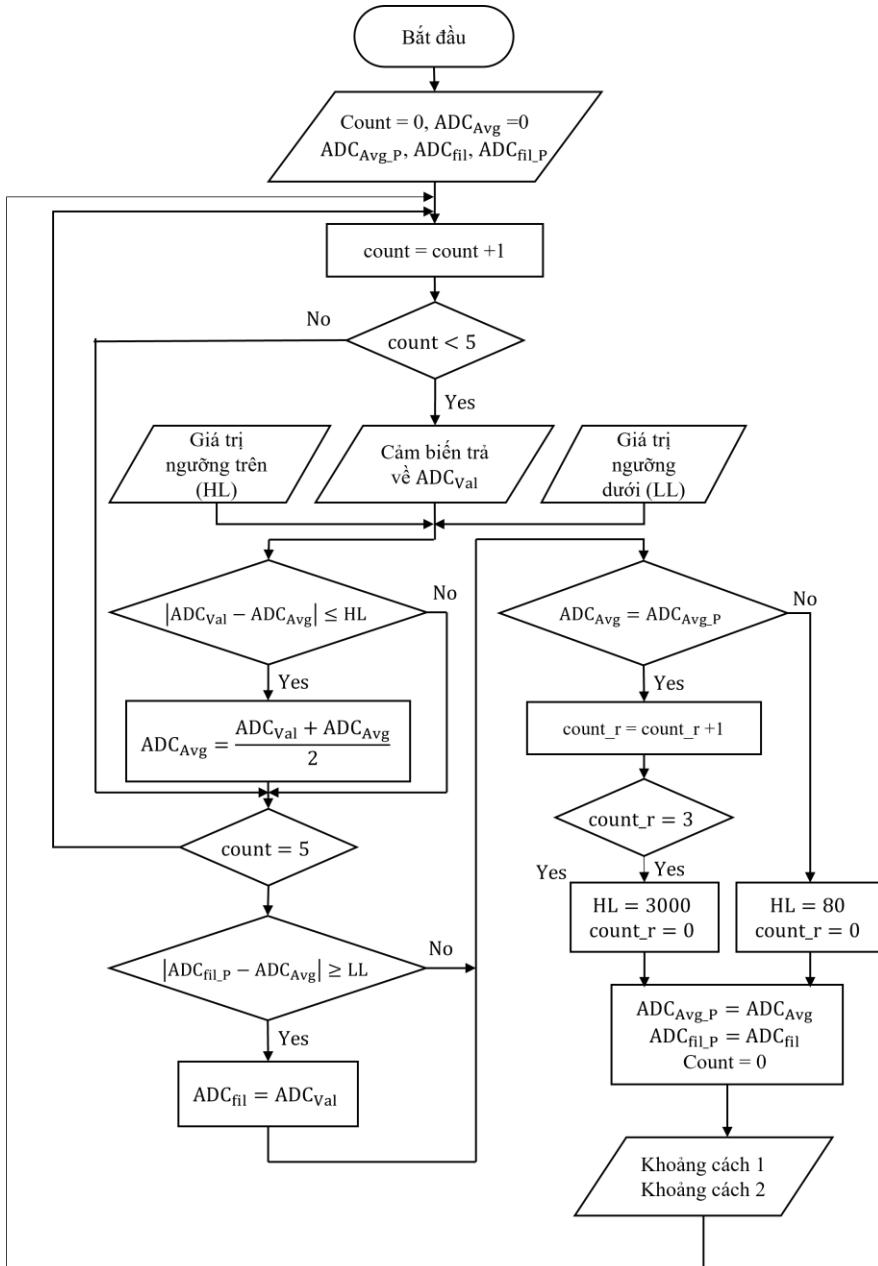
Phương pháp thực hiện như sau: Giả sử gọi lần lượt các biến vận tốc lập được tính ra cho bánh trái và bánh phải là SP_{WL} và SP_{WR} , gọi vận tốc đạt được của từng bánh sau khi qua bộ điều khiển là RPM_{WL} và RPM_{WR} , vận tốc đạt được tối đa của bánh trái cao hơn bánh phải. Nếu SP_{WR} và RPM_{WR} đều khác 0, ta sẽ có tỉ lệ $k = \frac{SP_{WL}}{SP_{WR}}$, khi đó lấy giá trị RPM_{WR} nhân với tỉ lệ k ta được một biến được gọi là vận tốc thiết lập mới cho bánh trái và dùng tham số mới để điều khiển bánh xe trái. Nếu không thỏa điều kiện trên, ta sẽ sử dụng trực tiếp giá trị vận tốc thiết lập ban đầu của bánh trái để điều khiển trực tiếp.

Ta có sơ đồ khái hoạt động của động cơ khi thỏa mãn điều kiện trên:



Hình 3.15. Sơ đồ khái hoạt động của bộ điều khiển động cơ khi thỏa điều kiện

3.2.3. Phương pháp lấy giá trị cảm biến



Hình 3.16 Lưu đồ giải thuật phương pháp đọc giá trị cảm biến

Nhóm chúng em sử dụng cảm biến khoảng cách trả về kết quả analog, để đọc cảm biến này nhóm chúng em sử dụng chế độ chuyển đổi ADC cũng có sẵn trên STM32 có thể trả về giá trị số trong khoảng 12 bit ([0,4095]). Tuy nhiên, vì là cảm biến trả về giá trị analog cho nên sinh ra rất nhiều nhiễu nên cần phải được lọc qua mới có thể dùng làm giá trị tính toán. Hiện nay có rất nhiều phương pháp lọc nhiễu nổi bật như lọc thông thấp hay sử dụng các bộ lọc kalman, tuy nhiên các bộ lọc này khi thiết kế để sử dụng trong thời gian thực có sự phức tạp không nhỏ nên vẫn chưa thể đáp ứng được điều kiện.

Vậy nên nhóm chúng em sử dụng phương pháp trung bình giá trị kết hợp với giới hạn sai số cạnh trên và cạnh dưới và được thể hiện ở lưu đồ giải thuật trên (Hình 3.16)

Để triển khai phương pháp trên, chúng em thực hiện trong ngắt timer 5 có chu kỳ 15ms, với mỗi lần ngắt timer chúng em sẽ thực hiện việc đo cảm biến một lần để có thể lấy được giá trị trung bình của giá trị ADC. Với việc đặt điều kiện $|ADC_{Val} - ADC_{Avg}| \leq HL$, chúng em đã hạn chế được việc phải tính những biến nhiễu có sai số biên độ cao vào giá trị trung bình sẽ dẫn đến giá trị trung bình cũng bị đẩy lên cao làm mất đi sự ổn định. Đồng thời với việc dùng biến count đếm lên, chúng em sẽ thiết lập được khoảng thời gian cập nhật biến ADC sau khi qua bộ lọc (ADC_{fil}) và được dùng để tính toán vào khoảng 75ms. Điều kiện để có thể cập nhật lại giá trị ADC_{fil} là $|ADC_{fil_P} - ADC_{Avg}| \geq LL$, cũng tức là chỉ cập nhật giá trị ADC_{fil} khi nào sai lệch giữa giá trị ADC_{Avg} hiện tại với giá trị ADC_{fil_P} lớn hơn một ngưỡng LL (LOW LIMIT), điều này sẽ giúp làm giảm đi những nhiễu tàn số cao với sai lệch biên độ thấp. Tuy nhiên, phương pháp này cũng có những hạn chế cản giải quyết như không thể thay đổi giá trị thật của cảm biến một cách đột ngột bởi do hạn chế về sai số cạnh trên. Cho nên, để khắc phục hạn chế đó, chúng em đưa thêm chúng em đưa thêm các điều kiện vào chương trình nhằm xoá bỏ hạn chế sai số cạnh trên đó là nếu giá trị trung bình không thay đổi gì sau khoảng 3 lần đọc giá trị thì sẽ thiết lập sai số cạnh trên thành rất lớn (trong chương trình là 3000) như thế sẽ có thể bắt được sự thay đổi giá trị lớn. và sau sự thay đổi đó, ta lại đưa sai số cạnh trên về mình thiết lập ban đầu.

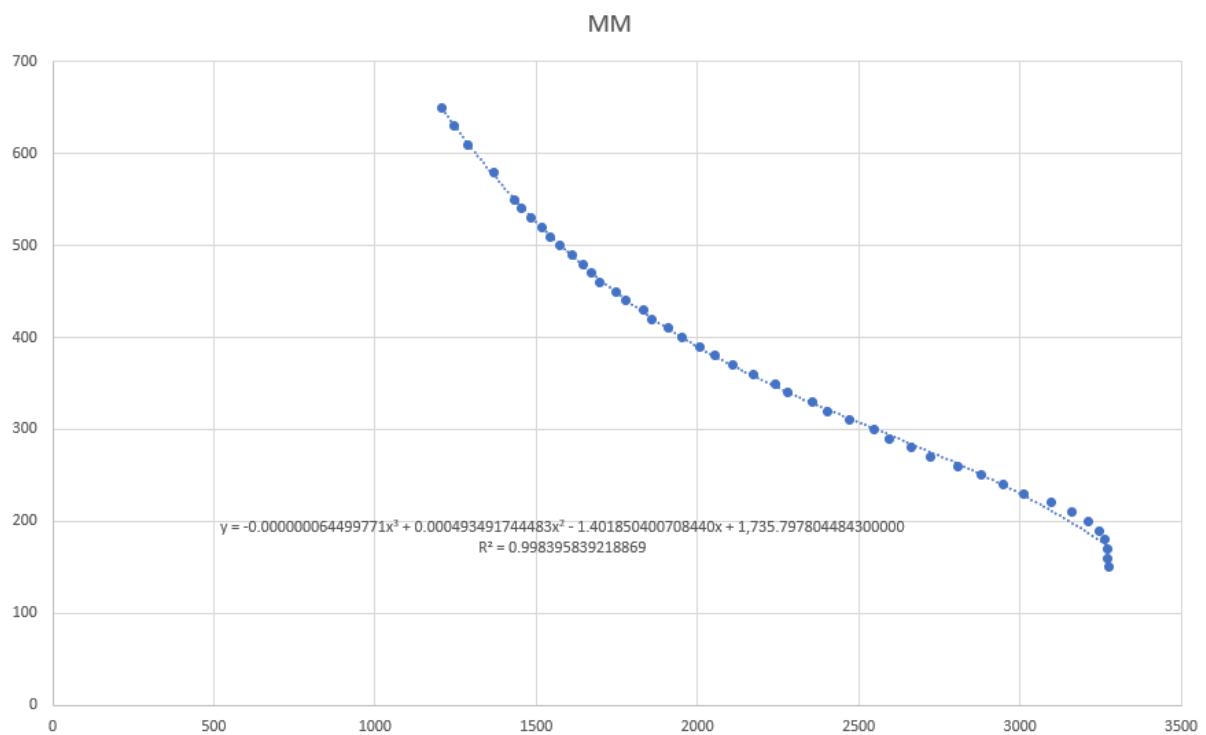
Sau khi đã có được giá trị ADC_{fil} rồi, chúng em tiến hành tính toán giá trị khoảng cách cảm biến trả về qua biến ADC_{fil} , quá trình được tiến hành bằng cách hiệu chỉnh từ thực nghiệm. Thông qua dữ liệu từ nhà sản xuất, chúng em xác định được khoảng đo khoảng từ 20 đến 150cm và khi tiến hành đo thực tế chúng em nhận thấy cảm biến có thể trả về dữ liệu có nghĩa trong khoảng 15 đến khoảng 65 cm, xác định khoảng đo chúng em tiến hành lấy mẫu. Phương pháp đo tương đối đơn giản như sau: chia khoảng đo thành từng khoảng nhỏ cố định (cách nhau 10 mm), với mỗi khoảng đo nhỏ như vậy chúng em lấy khoảng 5 giá trị ADC để tính ra giá trị trung bình ADC tại khoảng cách đó và ghi lại dữ liệu vào một tệp excel. Sau khi đã có đủ dữ liệu, chúng em tiến hành

vẽ đồ thị từ dữ liệu đã có và sử dụng khả năng nội suy phương trình săn có trong excel để tìm được phương trình liên hệ giữa giá trị ADC và khoảng cách cảm biến.

ADC1	ADC2	ADC3	ADC4	ADC5	ADC_AVG	MM
3276	3274	3274	3277	3275	3275.2	150
3255	3266	3278	3273	3280	3270.4	160
3278	3272	3275	3271	3262	3271.6	170
3268	3257	3257	3264	3264	3262	180
3247	3249	3247	3241	3244	3245.6	190
3208	3214	3214	3214	3207	3211.4	200
3153	3153	3151	3172	3175	3160.8	210
3099	3099	3095	3095	3095	3096.6	220
3011	3011	3007	3013	3016	3011.6	230
2953	2960	2960	2934	2932	2947.8	240
2866	2869	2864	2900	2900	2879.8	250
2799	2805	2808	2808	2816	2807.2	260
2722	2722	2727	2727	2718	2723.2	270
2662	2662	2674	2674	2650	2664.4	280
2590	2593	2598	2591	2600	2594.4	290
2555	2555	2558	2530	2543	2548.2	300
2472	2472	2473	2474	2471	2472.4	310
2395	2401	2402	2405	2402	2401	320
2349	2349	2352	2360	2366	2355.2	330
2276	2288	2288	2277	2279	2281.6	340
2257	2257	2222	2254	2222	2242.4	350
2160	2160	2184	2186	2175	2173	360
2102	2102	2118	2116	2116	2110.8	370
2053	2054	2054	2046	2057	2052.8	380
2005	2003	2026	2001	2001	2007.2	390
1943	1940	1976	1949	1945	1950.6	400
1899	1902	1894	1929	1920	1908.8	410
1870	1863	1863	1850	1844	1858	420
1847	1843	1826	1827	1829	1834.4	430

1770	1775	1782	1782	1780	1777.8	440
1748	1752	1742	1748	1743	1746.6	450
1704	1700	1693	1698	1695	1698	460
1682	1683	1683	1654	1665	1669	470
1639	1648	1640	1652	1652	1646.2	480
1607	1604	1607	1613	1618	1609.8	490
1572	1569	1569	1593	1572	1575	500
1539	1540	1544	1549	1537	1541.8	510
1514	1507	1528	1532	1515	1519.2	520
1493	1491	1484	1477	1483	1485.6	530
1441	1453	1453	1456	1468	1454.2	540
1436	1422	1436	1425	1433	1430.4	550
1370	1382	1386	1348	1356	1368.4	580
1298	1289	1279	1286	1286	1287.6	610
1256	1240	1245	1247	1236	1243	630
1211	1215	1203	1222	1193	1208.8	650

Bảng 3.3. Bảng số liệu đo cảm biến



Hình 3.17 Đồ thi và biểu thức liên hệ giữa khoảng cách theo giá trị ADC

3.2.4. Bộ điều khiển hệ thống

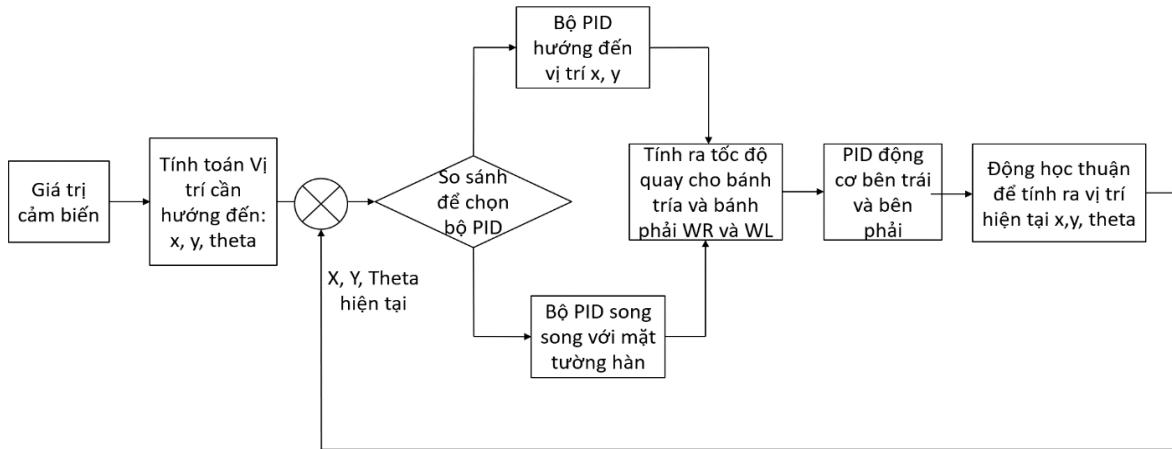
3.2.4.1. Phương án lựa chọn bộ điều khiển

Trong những năm gần đây, đã có nhiều nghiên cứu về ứng dụng robot di động trong lĩnh vực hàn tự động, trong đó có một số nghiên cứu sau đây:

- Năm 2001, Trong nghiên cứu của Yang-Bae Jeon và các đồng nghiệp về điều khiển chuyển động của robot hàn di động đã sử dụng thuật toán PID để điều khiển càn hàn bám đường hàn.
- Năm 2003, trong nghiên cứu của Bùi Trọng Hiếu và các đồng nghiệp về áp dụng điều khiển phi tuyến đơn giản trong robot hàn di động đã sử dụng luật điều khiển theo tiêu chuẩn ổn định Lyapunov.
- Năm 2007, trong nghiên cứu của Ngô Mạnh Dũng và các đồng nghiệp về điều khiển bám đường hàn cong của robot hàn di động hai bánh xe sử dụng phương pháp mặt trượt thích nghi.

Các phương pháp kể trên đều là các phương án nổi tiếng dùng để điều khiển chuyển động của robot di động. Trong đó, với phương pháp sử dụng luật điều khiển theo tiêu chuẩn ổn định Lyapunov và phương pháp điều khiển trượt đều là những phương pháp có khả năng kiểm soát độ ổn định rất mạnh mẽ, nhất là đối với hệ thống động phi tuyến, nơi mà các phương pháp kiểm soát truyền thống có thể gặp khó khăn. Tuy nhiên, hai phương pháp trên tồn tại một số nhược điểm như: đối với phương pháp Lyapunov thì yêu cầu gắt gao về việc xây dựng hàm Lyapunov, đối với bộ điều khiển trượt thì dễ dàng xảy ra hiện tượng chattering – hiện tượng quỹ đạo pha dao động quanh mặt trượt. Vậy nên, nhóm chúng em quyết định sử dụng bộ điều khiển PID để điều khiển điểm hàn bám đường hàn nhờ việc ứng dụng linh hoạt và khả năng tiếp cận đối với phương pháp.

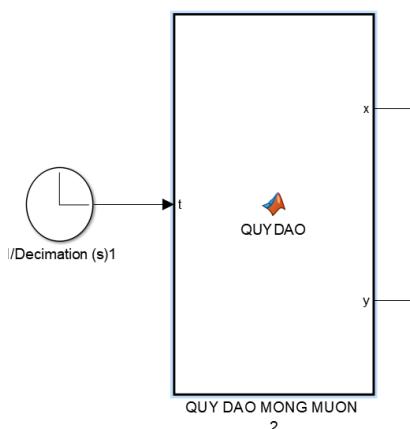
3.2.4.2. Bộ điều khiển PID cho hệ thống



Hình 3.18 Sơ đồ khái niệm của hệ thống

Đầu tiên muốn robot cần hàn hướng đến giá trị đặt. Ta phải thu thập giá trị cảm biến ADC để sau đó chuyển giá trị ADC sang khoảng cách và vị trí cần hàn hướng đến. Nếu sai số khoảng cách lớn hơn 10mm thì sẽ cho robot bật bộ PID bám vị trí hàn và ngược lại nếu đã ổn định sai số khoảng cách thì sẽ bật bộ PID song song để robot bám song song với tường. Sau đó chọn ra WR và WL và qua bộ PID cho động cơ trái và phải thì ta được WR và WL của cả hai bánh xe để nhờ công thức tính động học thuận suy ra được vị trí cần hàn và góc hiện tại.

Trên thực tế, từ giá trị ADC được cảm biến đọc về ta tính ra được khoảng cách hàn. Đối với mô phỏng trong matlab em giả lập tín hiệu cảm biến bằng hàm tham số theo thời gian t.



Hình 3.19. Khối xây dựng quỹ đạo mong muốn cho mô phỏng

Trong phần mô phỏng này, em sẽ thực hiện mô phỏng hai đường quỹ đạo là đường thẳng và đường tròn.

```

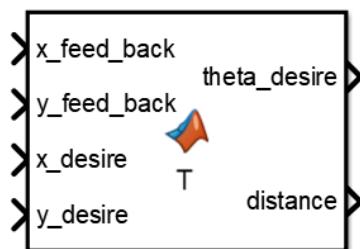
function [x,y] = QUYDAO(t)
x = 2*cos(0.04*t); %% Quỹ đạo đường tròn
y = 2*sin(0.04*t);

x = 0.0625*t;    %% Quỹ đạo đường thẳng
y = 0.05*t;

```

Hình 3.20.Hàm xây dựng quỹ đạo mô phỏng

Sau khi có quỹ đạo mô phỏng em tiến hành tính góc theta và khoảng cách để cần hàn hướng đến.



Tính góc theta

Hình 3.21. Khối tính toán góc theta và khoảng cách mong muốn

```

function [theta_desire, distance] = T(x_feed_back, y_feed_back, x_desire, y_desire)

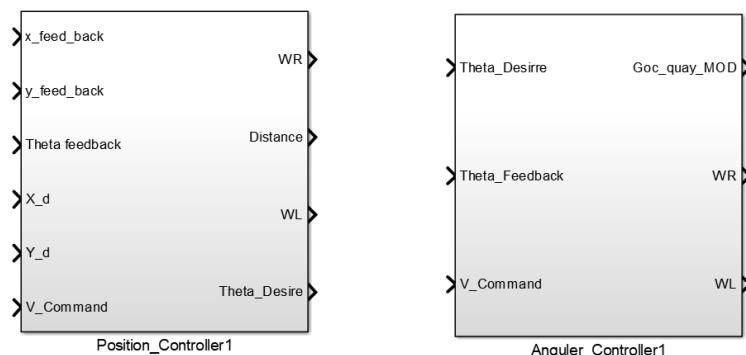
theta_desire = atan2(y_desire - y_feed_back, x_desire - x_feed_back);

distance = sqrt((y_desire - y_feed_back)^2 + (x_desire - x_feed_back)^2);

```

Hình 3.22.Hàm tính toán góc theta và khoảng cách mong muốn

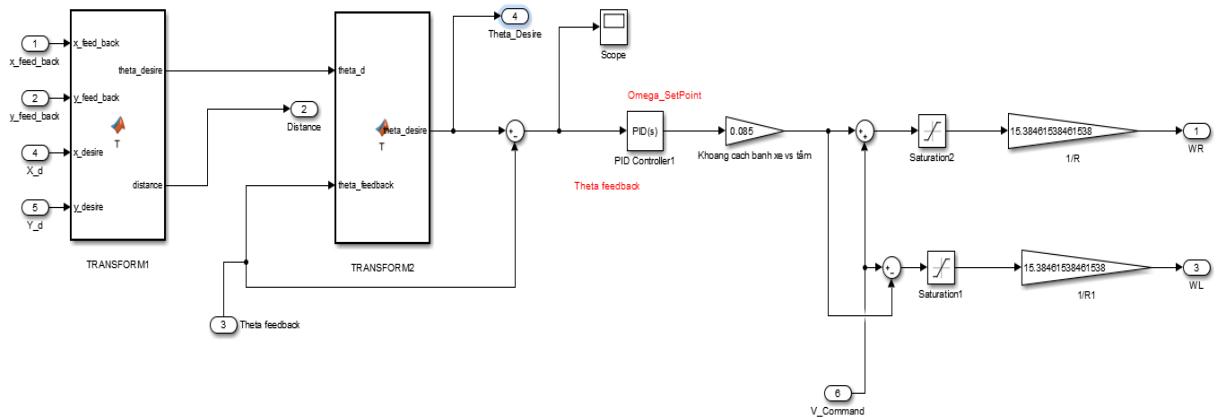
Sau khi tính được góc theta hướng đến và so sánh với giá trị góc theta hiện tại của cần hàn. Ta sẽ so sánh khoảng cách cần hướng đến và góc quay cần hướng đến của cần hàn để chọn bộ PID hợp lý.



Hình 3.23.Khối điều khiển theo sai số khoảng cách (trái) và khối điều khiển theo sai số góc (phải)

Bộ PID để bám vị trí hướng đến x,y. Khi có sai số góc quay theta thì ta đưa vào bộ PID để tính ra được tốc độ quay của hệ xe. Sau đó dựa trên công thức động học nghịch của của xe ta tính ra được WR và WL mong muốn để xe bám vào điểm cần hàn.

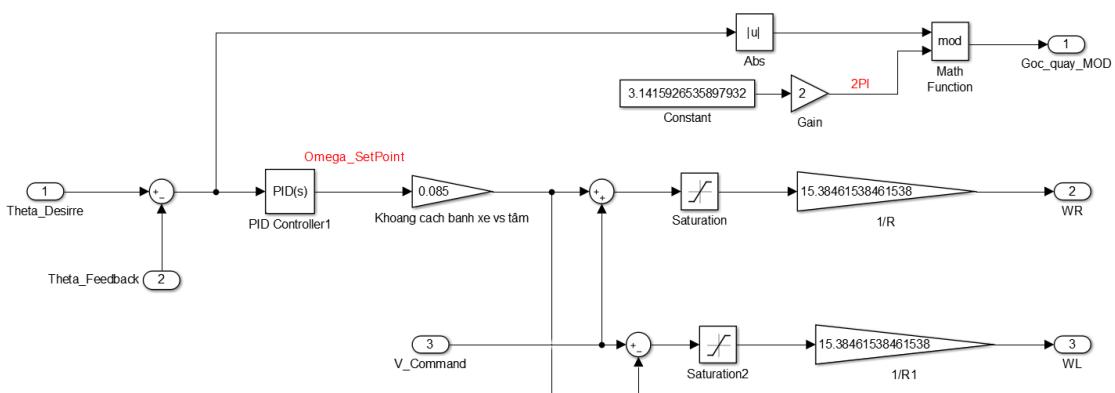
$$\begin{cases} \omega_R = \frac{v_c + b \cdot \omega_c}{r} \\ \omega_L = \frac{v_c - b \cdot \omega_c}{r} \end{cases} \quad (3.38)$$



Hình 3.24. Cấu trúc bên trong khối điều khiển theo sai số khoảng cách

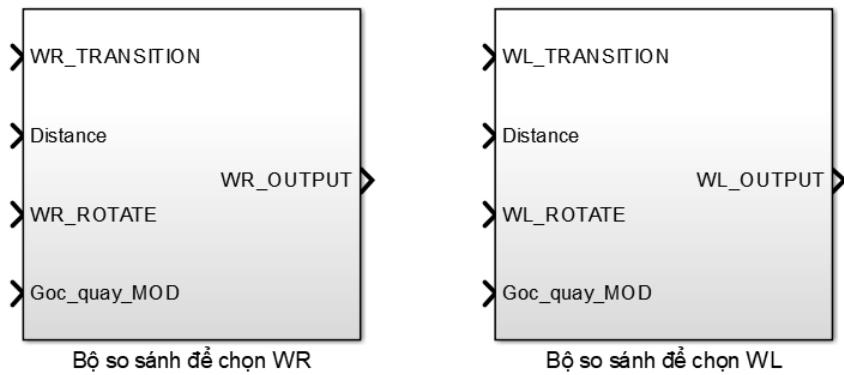
Tương tự như tính được WR và WL để xe bám vào vị trí hàn thì ta có cảm biến thứ hai để đo khoảng cách và tính toán để xe bám song song với tường. Chỉ khác ở công thức tính góc theta hướng đến để robot song song với tường là:

$$\theta = \tan^{-1} \left(\frac{x_1 - x_2}{y_1 - y_2} \right) \quad (3.39)$$

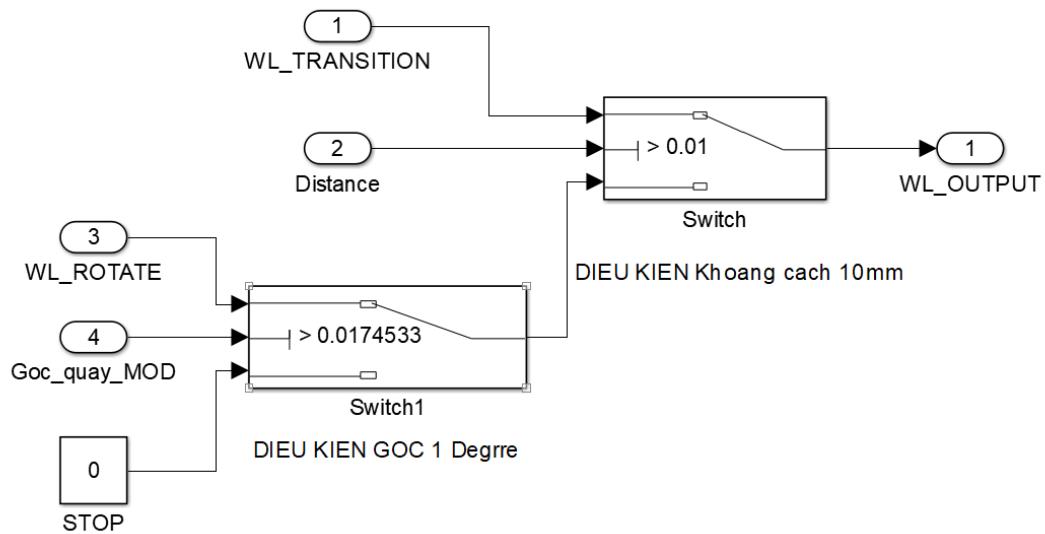


Hình 3.25. Cấu trúc bên trong khối điều khiển theo sai số góc

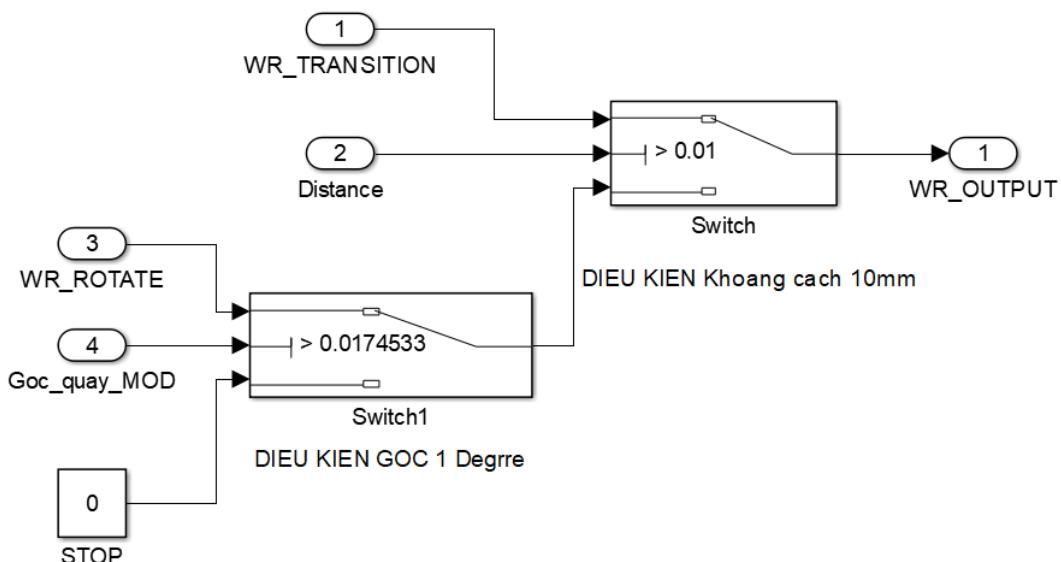
Cả hai bộ điều chỉnh góc quay theo PID trên đều cho ra được cả hai giá trị WR và WL nên ta cần bộ so sánh để chọn ra giá trị WR với WL phù hợp để robot có thể hàn được vị trí mong muốn. Cụ thể hơn là khi sai số khoảng cách e_d lớn hơn 10mm thì ta chọn thông số WR và WL của bộ điều khiển bám vị trí x và y. Ngược lại khi sai số e_d nhỏ hơn 10mm thì điều đó có nghĩa là cần hàn của robot đang gần với với điểm hàn cần hướng đến nên ta bật bộ PID điều chỉnh để robot song song với tường hàn và gần như khi vào quỹ đạo hàn thì robot chỉ dùng bộ PID cho robot song song với tường. Khi bộ song song với tường được bật lên và nếu góc quay nhỏ hơn 1 độ thì cho động cơ dừng để giảm độ sai lệch.



Hình 3.26.Khối điều chỉnh vận tốc thiết lập bánh phải (trái) và bánh trái (phải)

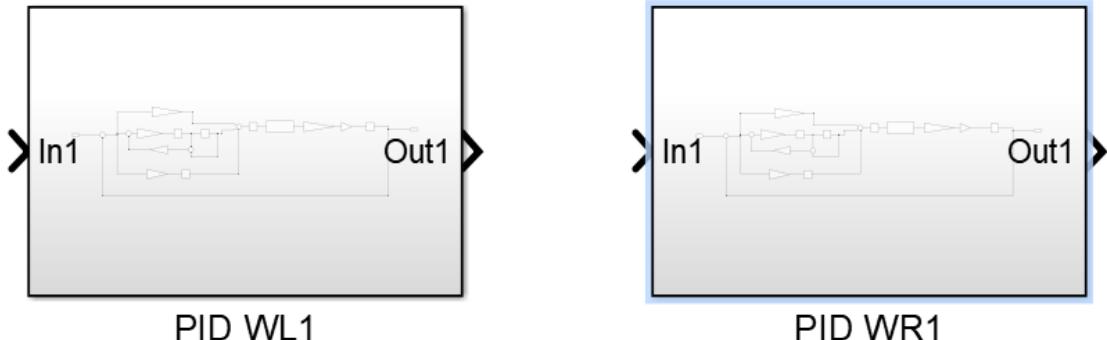


Hình 3.27.Cấu trúc bên trong khối điều chỉnh vận tốc góc thiết lập bánh trái

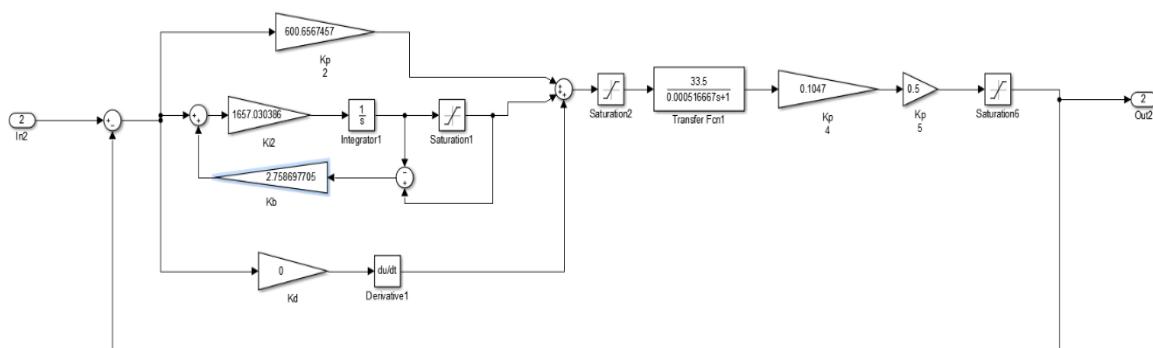


Hình 3.28.Cấu trúc bên trong khối điều chỉnh vận tốc góc thiết lập bánh phải

Khi tính được WR và WL thiết lập để cho cần hàn bám vào vị trí cần hàn thì ta đưa nó vào input của bộ PID cho động cơ như em đã giải thích ở phần trên.

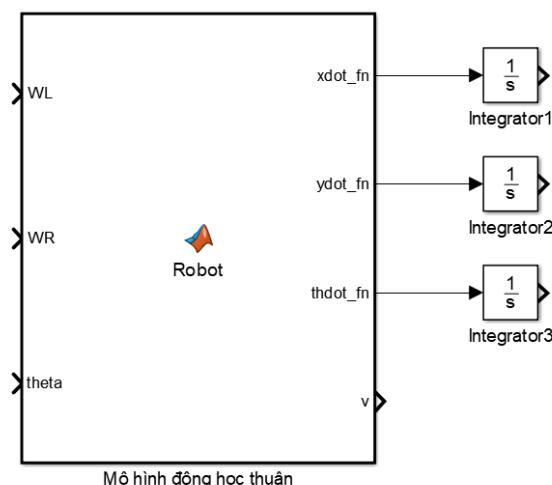


Hình 3.29. Khối điều khiển PID động cơ bánh trái (trái) và bánh phải (phải)



Hình 3.30. Cấu trúc khói điều khiển PID động cơ

Sau khi ra được WR và WL thực tế em đưa vào mô hình tính động học thuận để tính ra được x, y, góc theta và giá trị vận tốc hiện tại của xe. Và đưa tín hiệu hồi tiếp về bộ PID vị trí và góc quay để lắp lại chu kì tiếp theo một cách tuần tự.



Hình 3.31. Mô hình tính toán động học thuận của hệ thống

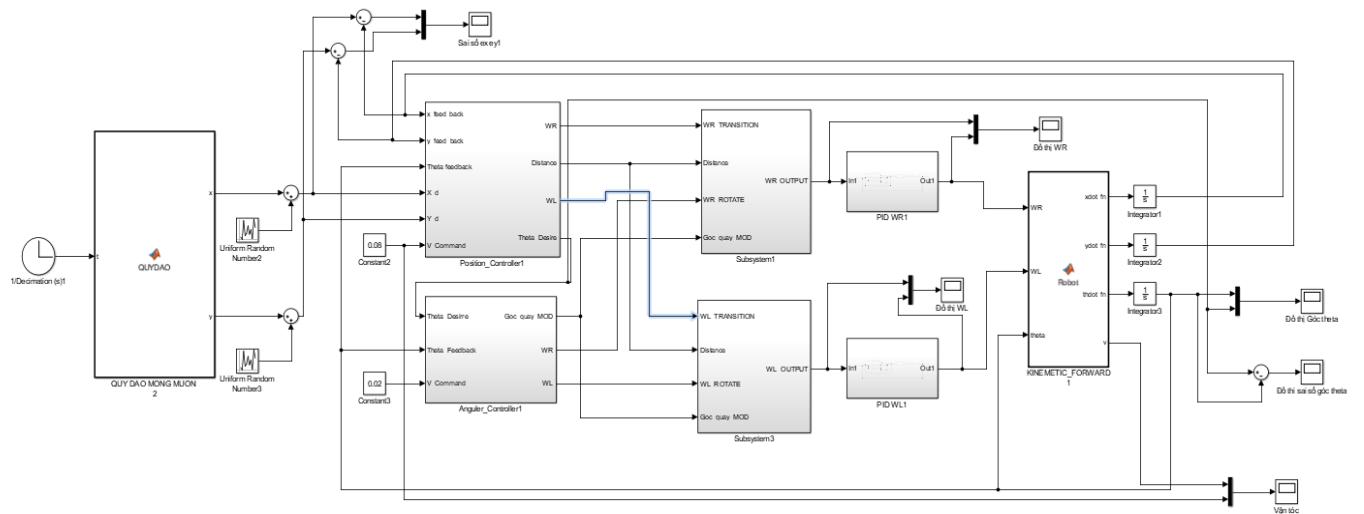
```

function [xdot_fn,ydot_fn,thdot_fn, v] = Robot(WL,WR,theta)
v= (WR+WL)*0.065/2;
w= ((WR-WL)*0.065) / (2*0.17);
thdot_fn=w;
xdot_fn=v*cos(theta);
ydot_fn=v*sin(theta);

```

Hình 3.32. Hàm tính toán động học thuận của hệ thống

Sau khi tổng hợp các khối ở trên, chúng ta được một sơ đồ hệ thống robot của nhóm

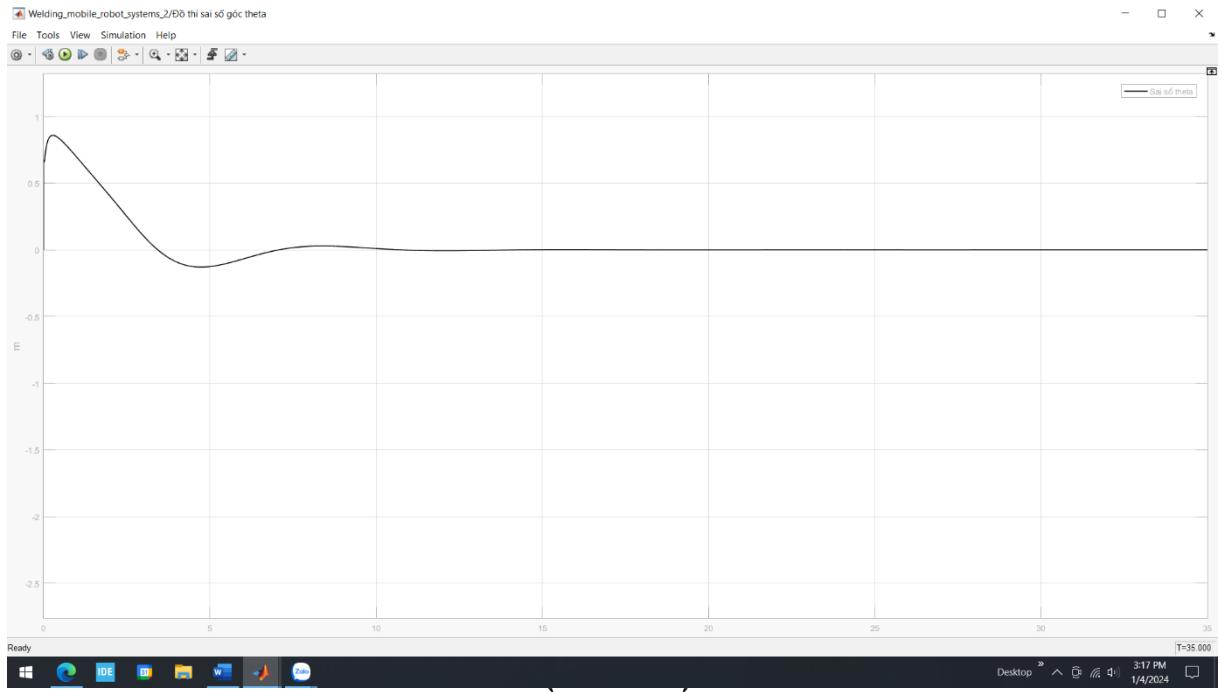


Hình 3.33.Sơ đồ khối hệ thống robot

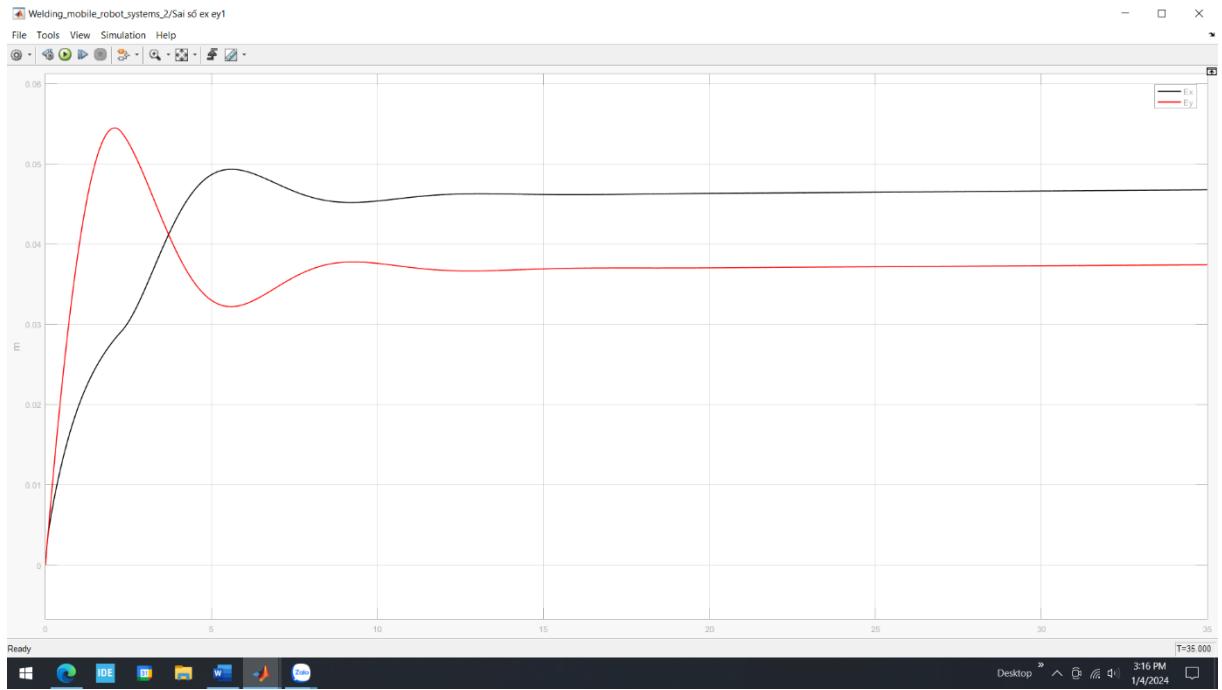
CHƯƠNG 4. KẾT QUẢ MÔ PHỎNG VÀ THỰC NGHIỆM

4.1. Kết quả mô phỏng:

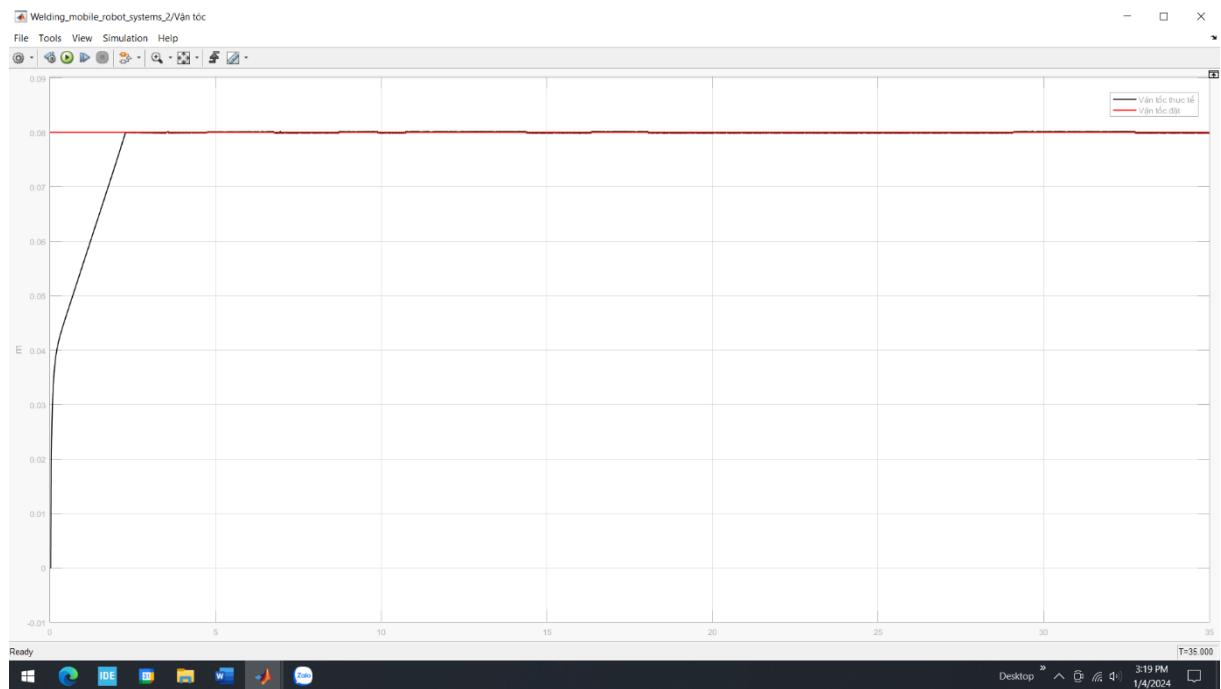
4.1.1. Kết quả mô phỏng khi chạy trên đường thẳng khi không có nhiễu cảm biến:



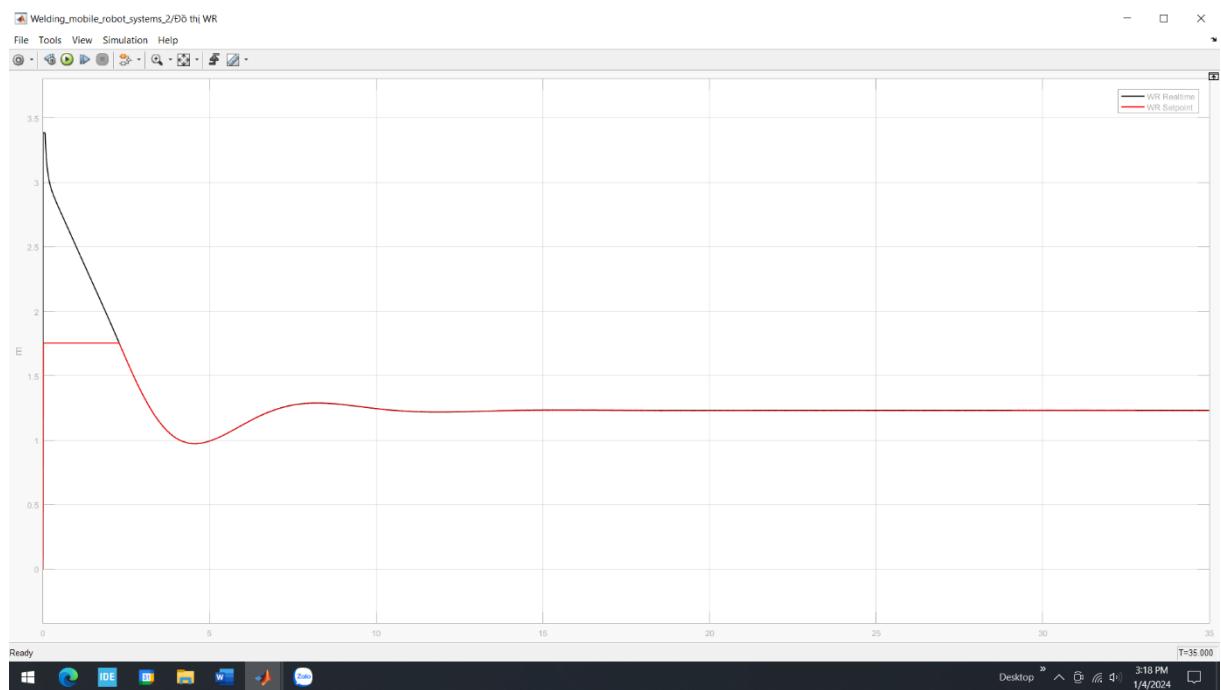
Hình 4.1.Đồ thị sai số x và y.



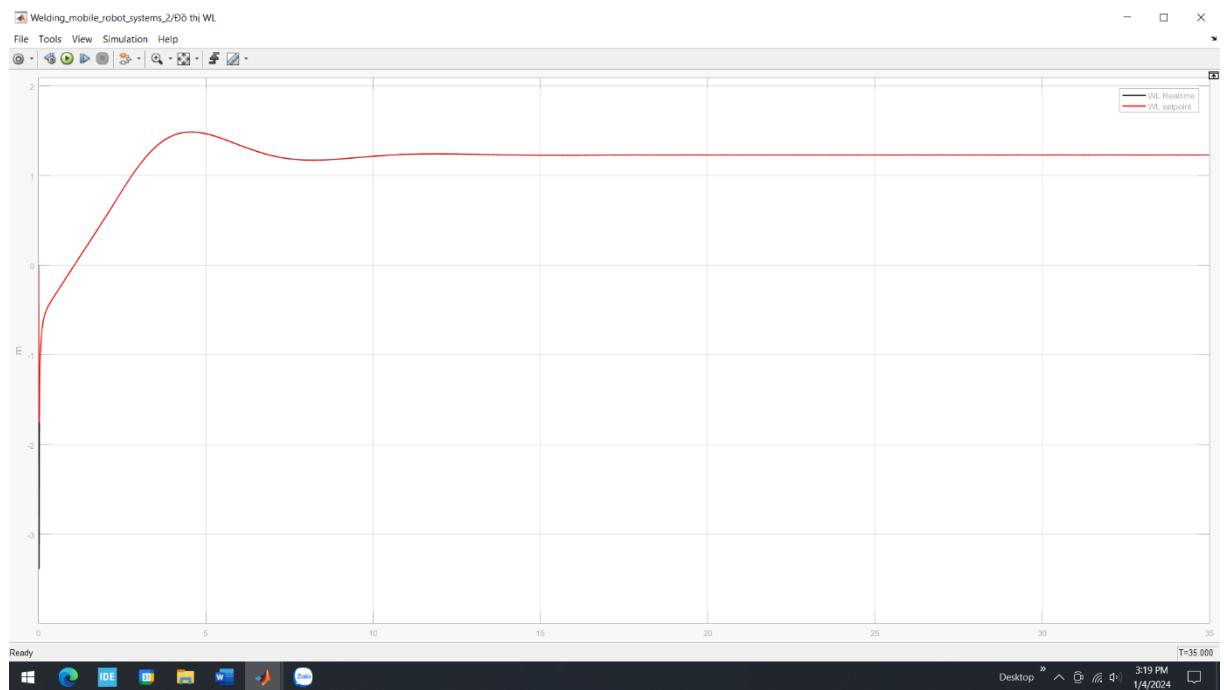
Hình 4.2.Đồ thị sai số góc theta.



Hình 4.3.Đồ thị bám của vận tốc

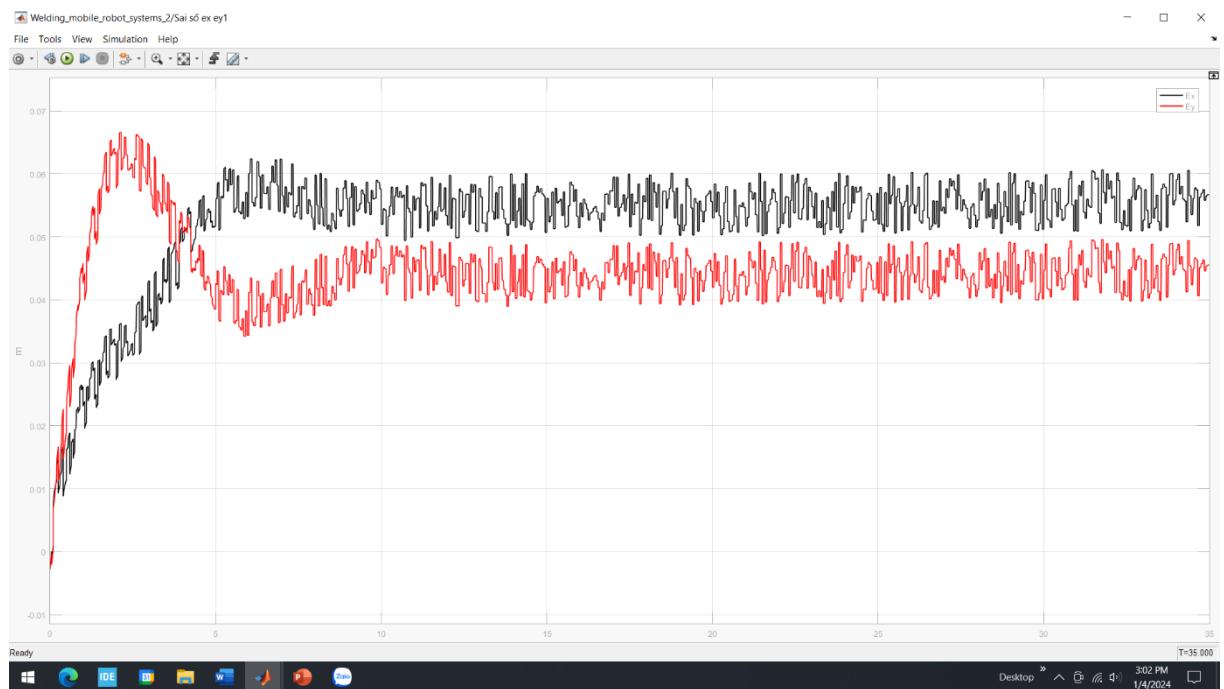


Hình 4.4.Đồ thị bám của WR

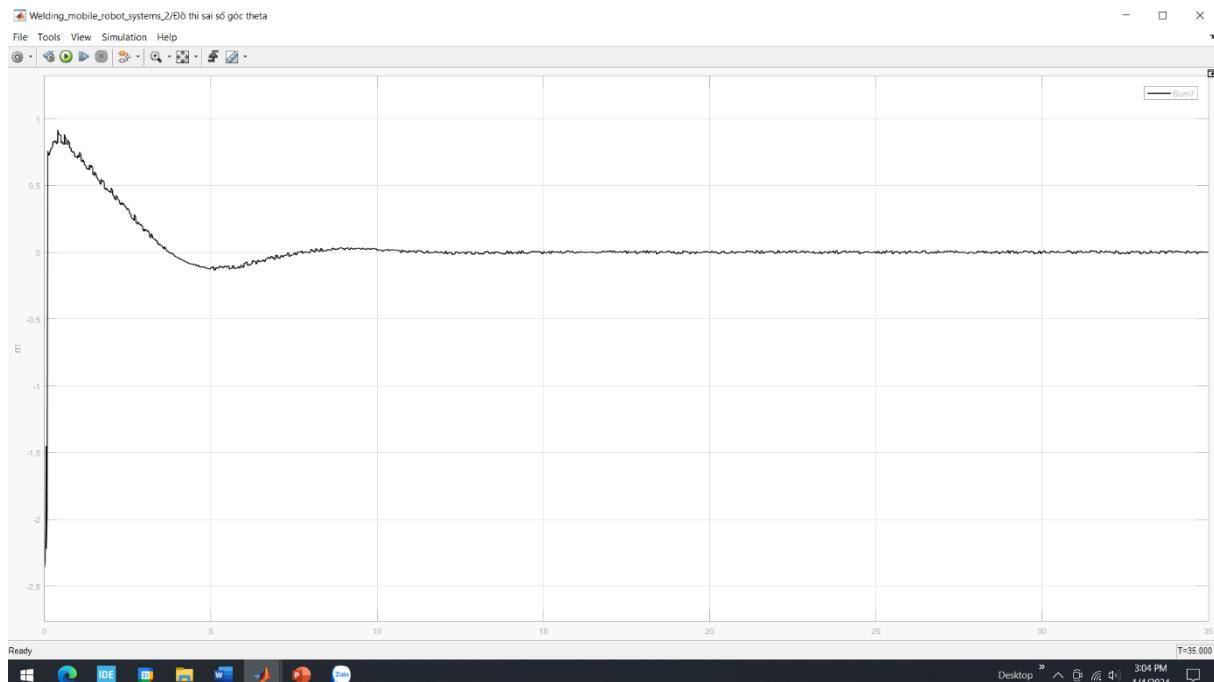


Hình 4.5.Đồ thị bám của WL

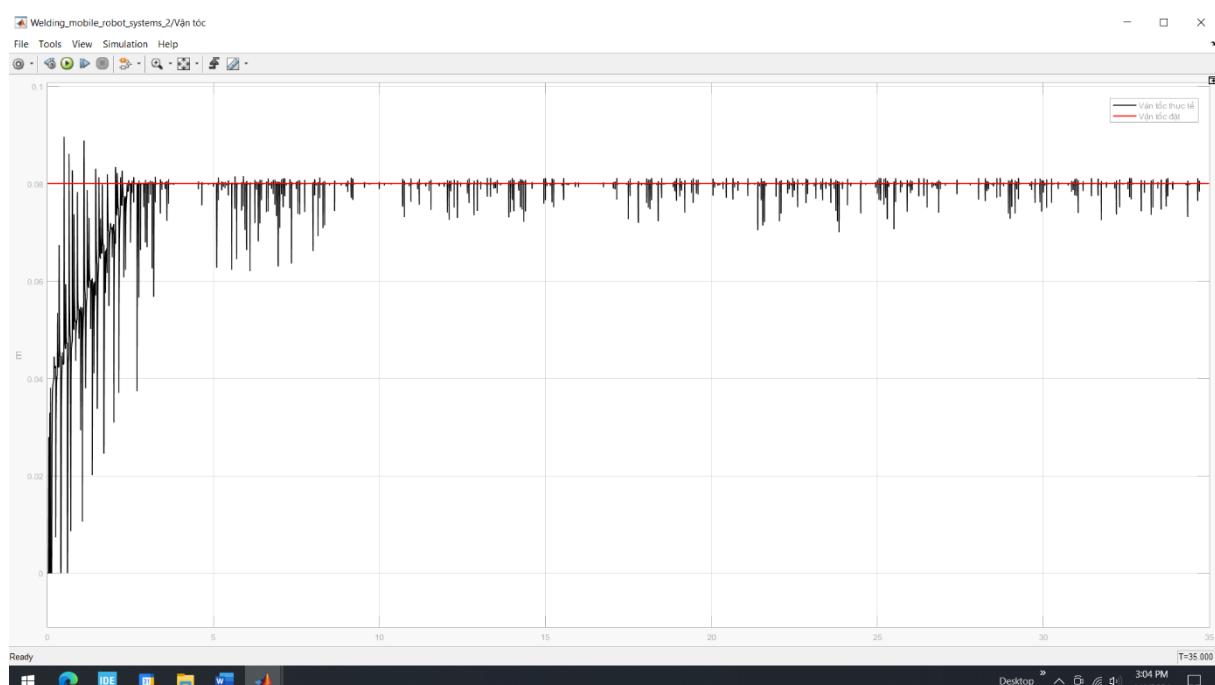
4.1.2. Kết quả mô phỏng khi chạy trên đường thẳng khi có nhiễu cảm biến:



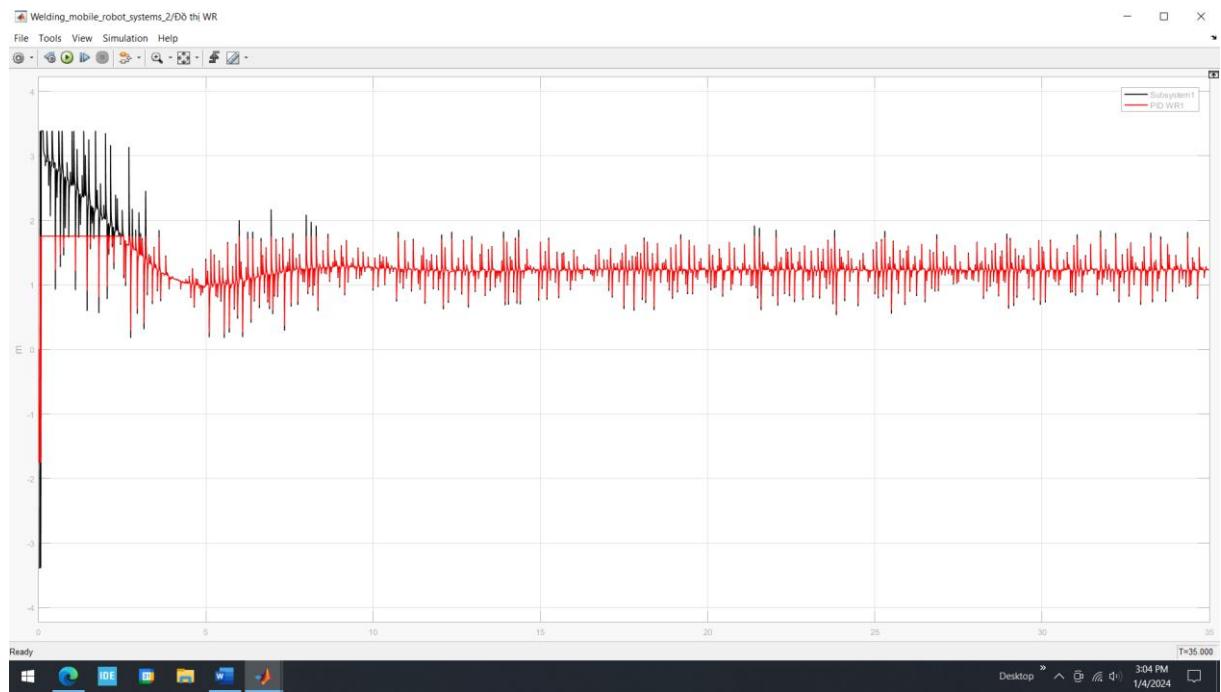
Hình 4.6.Đồ thị sai số x và y.



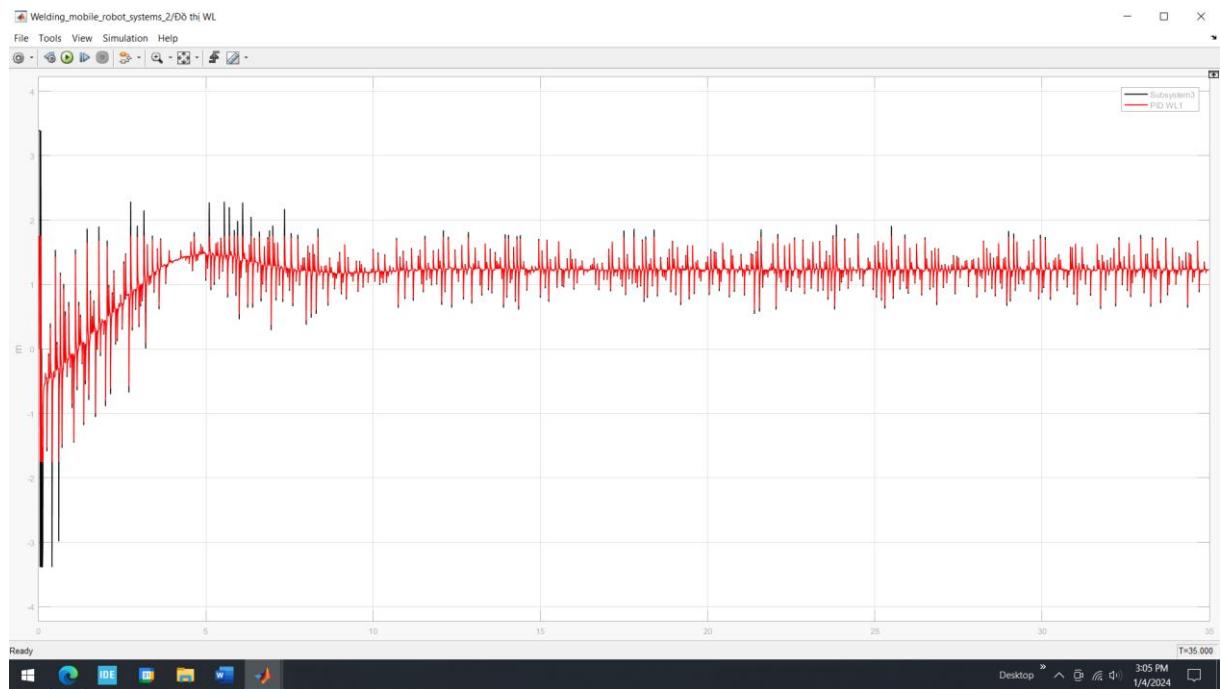
Hình 4.7.Đồ thị sai số góc theta.



Hình 4.8.Đồ thị bám của vận tốc

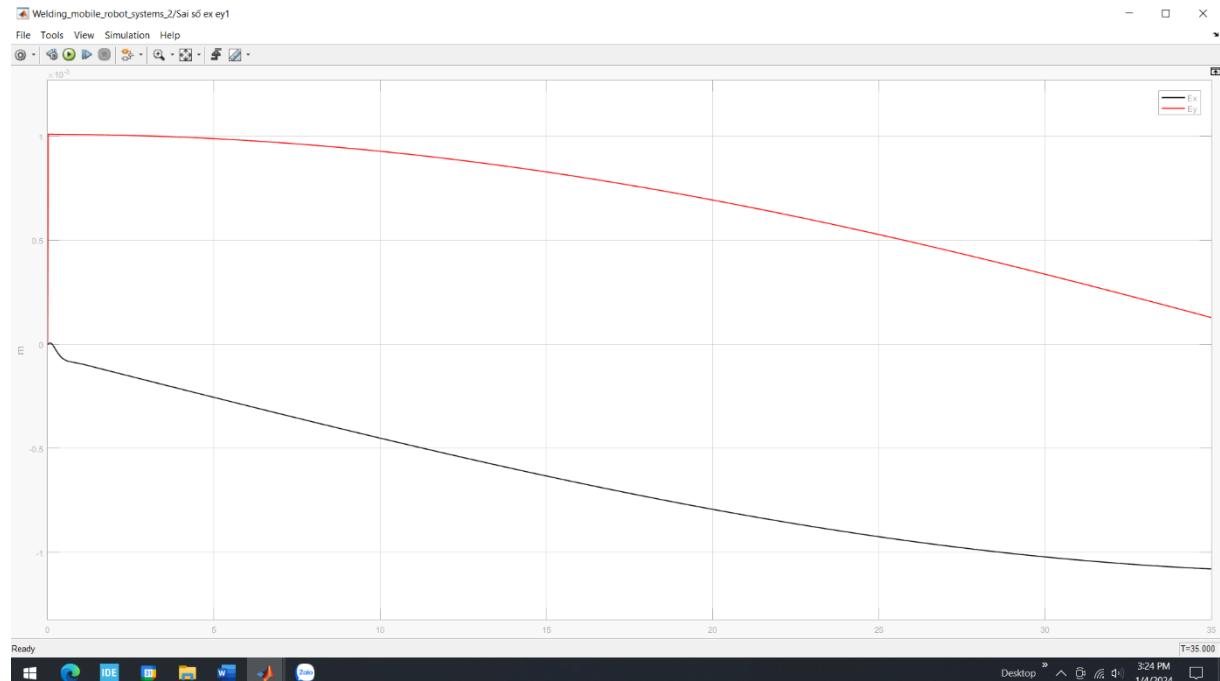


Hình 4.9.Đồ thị bám của WR

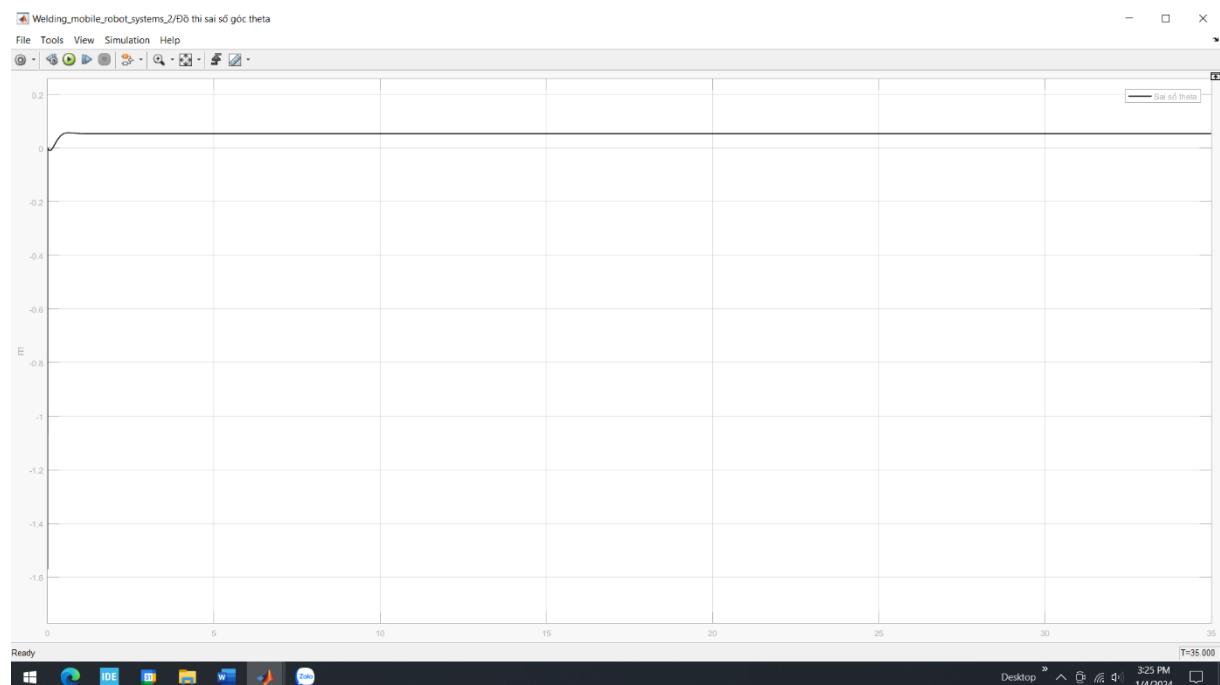


Hình 4.10.Đồ thị bám của WL

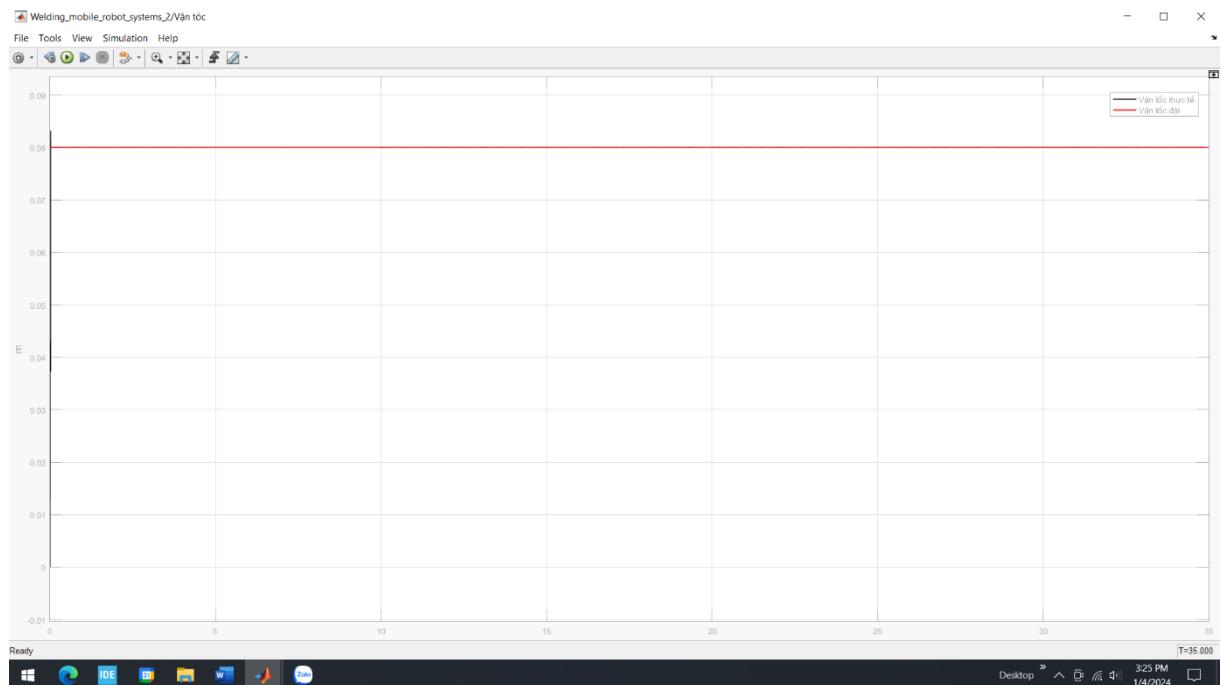
4.1.3. Kết quả mô phỏng khi chạy trên đường tròn khi không có nhiễu cảm biến:



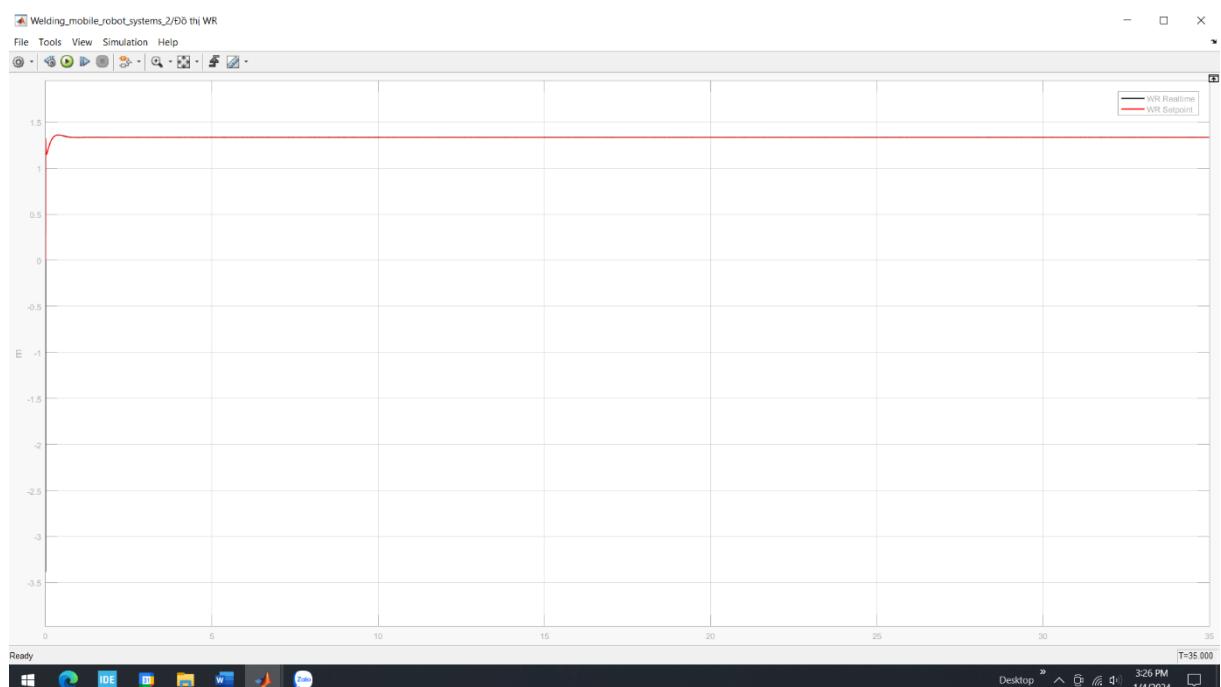
Hình 4.11.Đồ thị sai số x và y.



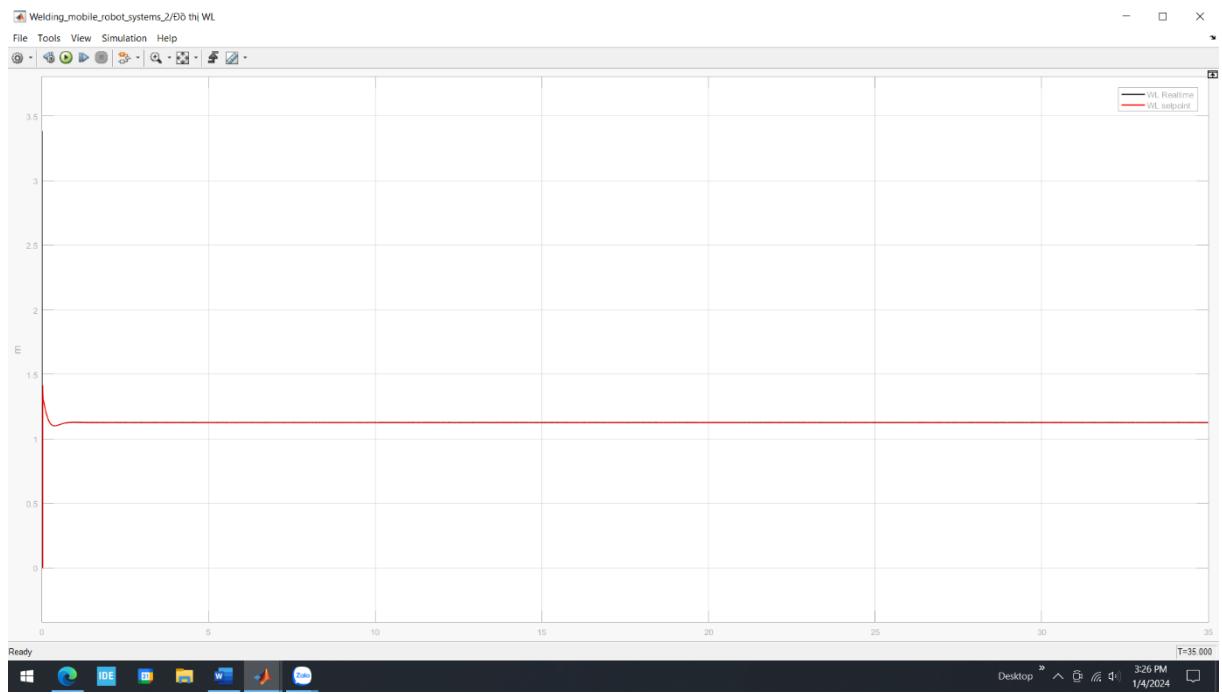
Hình 4.12.Đồ thị sai số góc theta.



Hình 4.13.Đồ thị bám của vận tốc

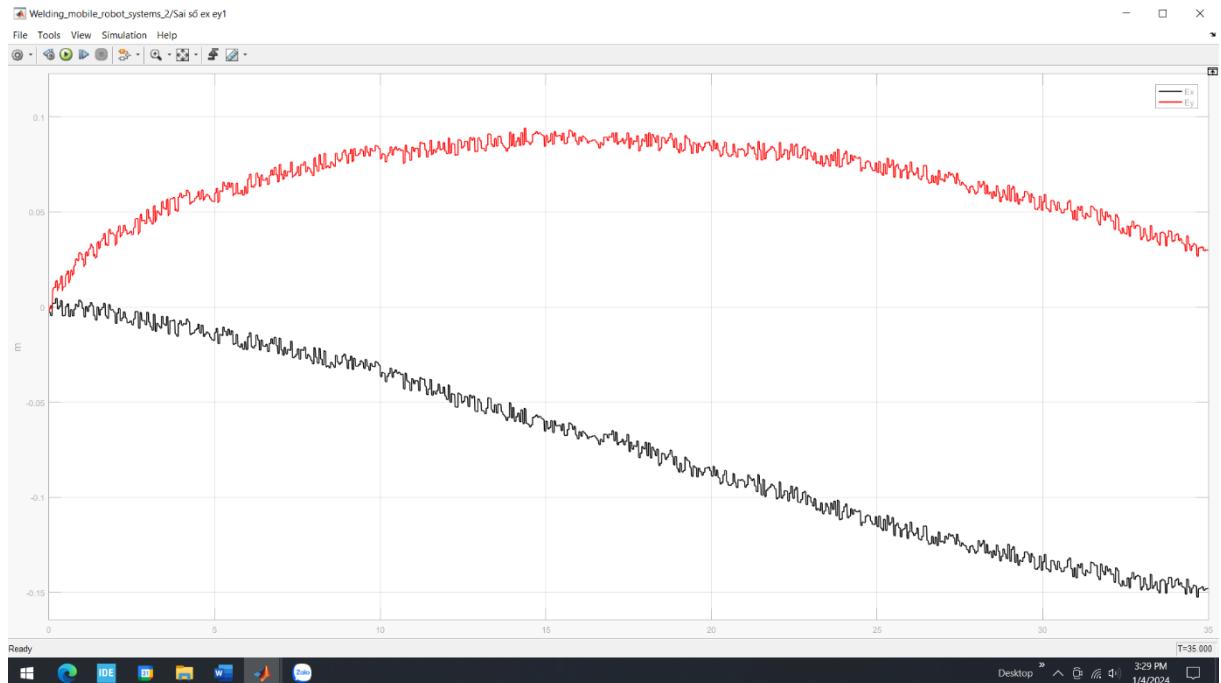


Hình 4.14.Đồ thị bám của WR

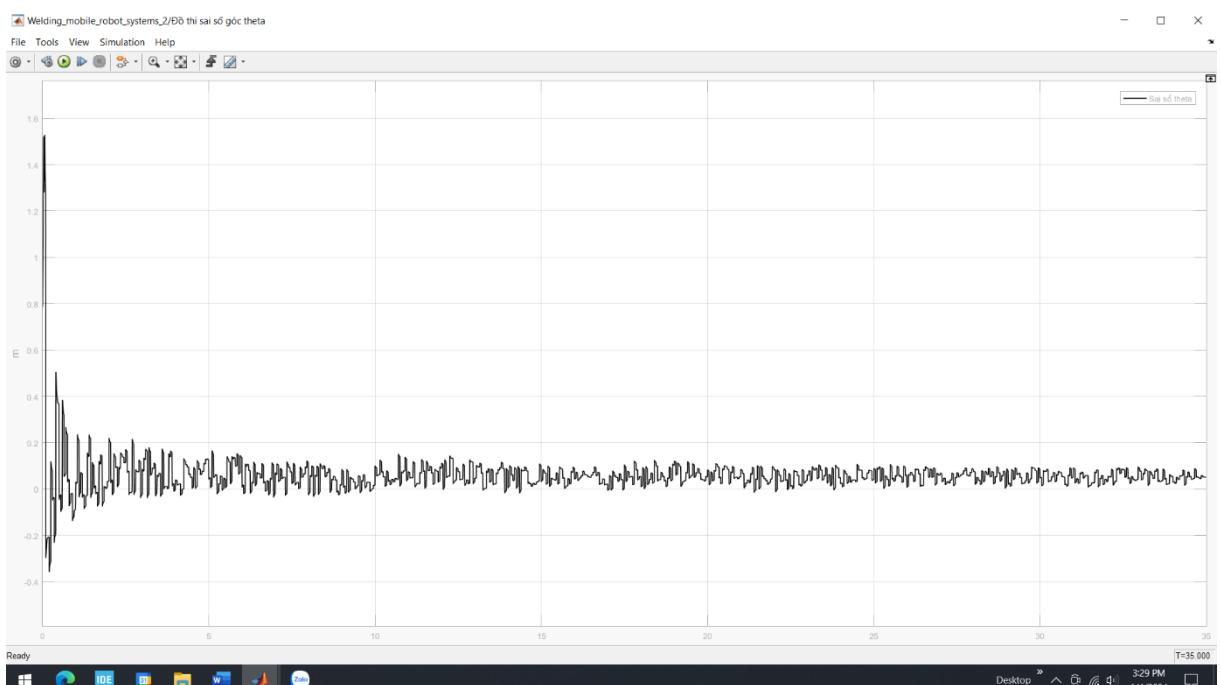


Hình 4.15.Đồ thị bám của WL

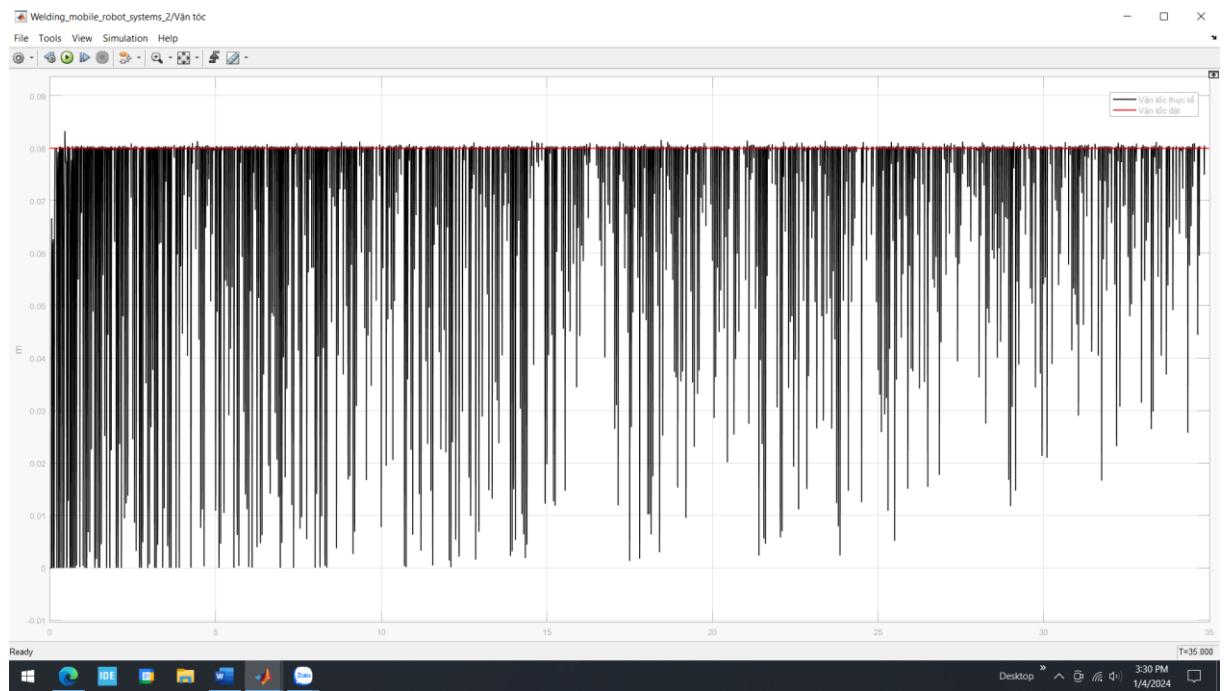
4.1.4. Kết quả mô phỏng khi chạy trên đường tròn khi có nhiễu cảm biến:



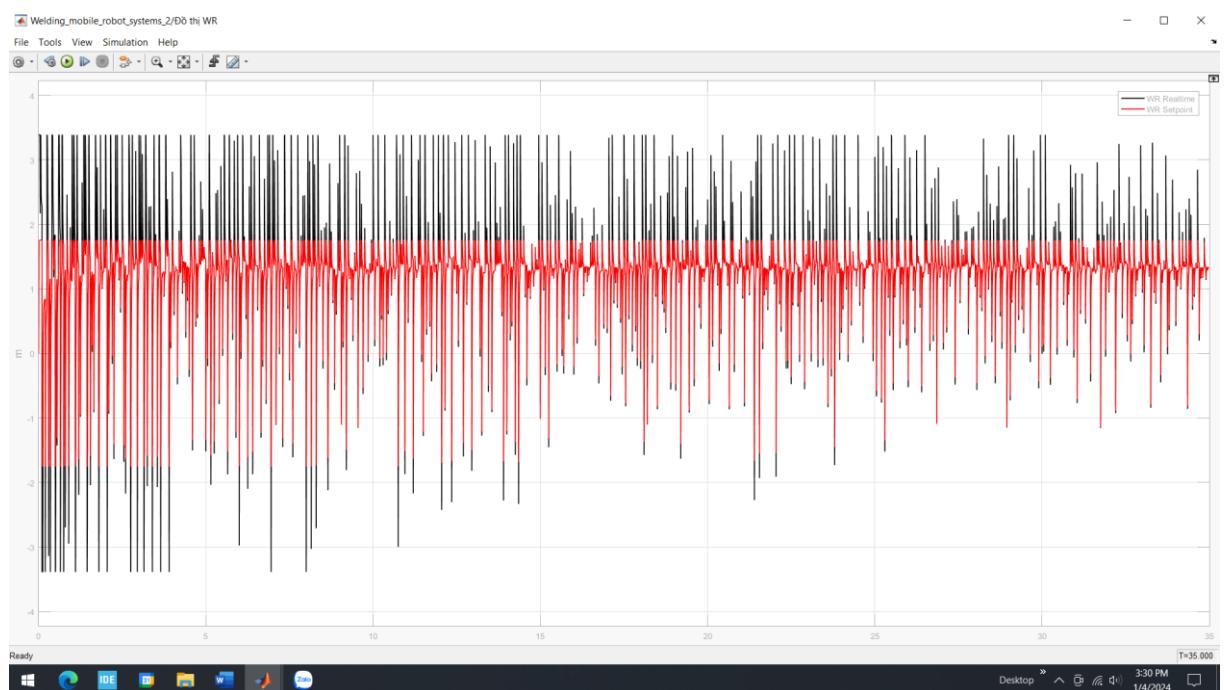
Hình 4.16.Đồ thị sai số x và y.



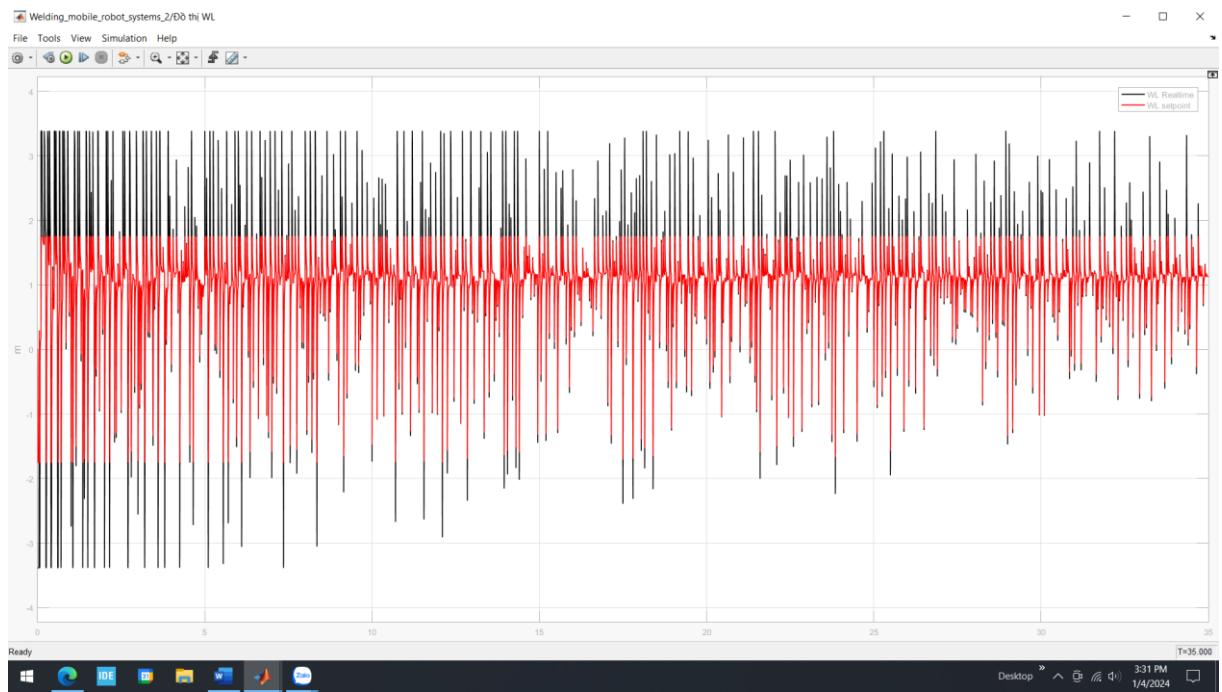
Hình 4.17.Đồ thị sai số góc theta.



Hình 4.18.Đồ thị bám của vận tốc



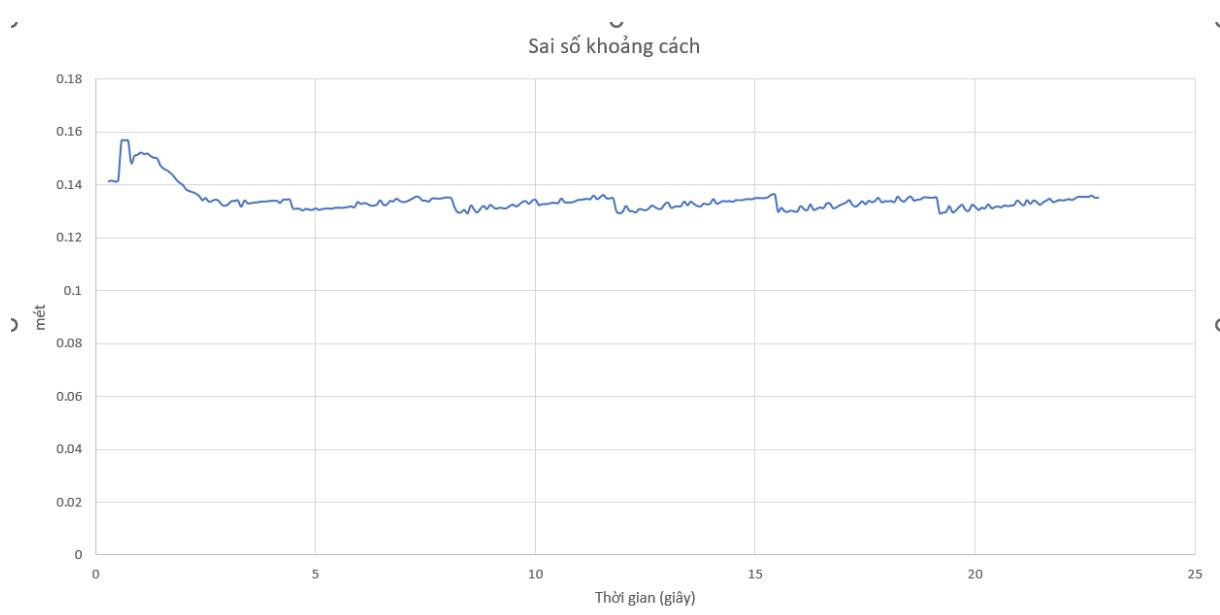
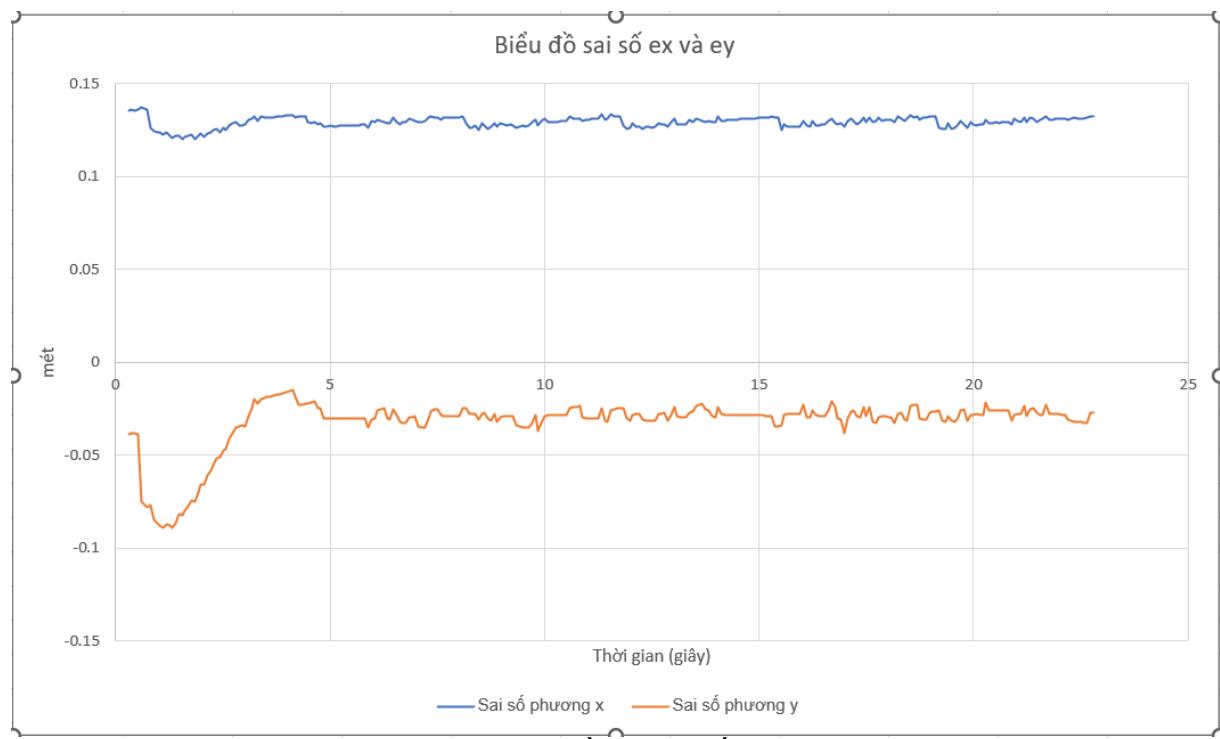
Hình 4.19.Đồ thị bám của WR

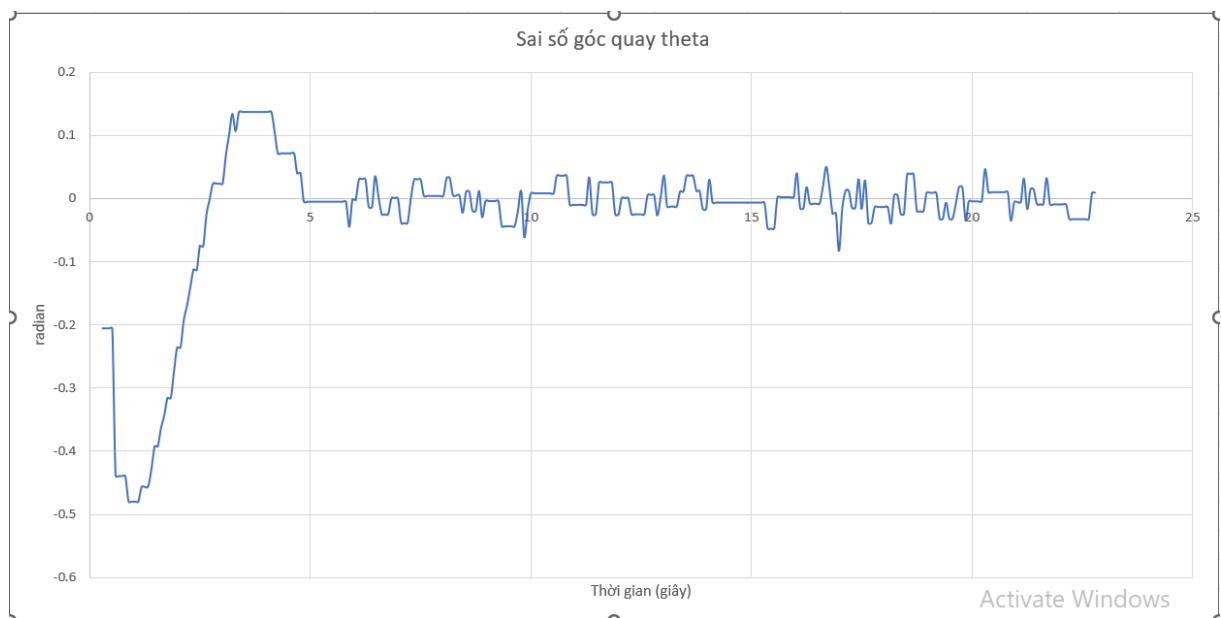


Hình 4.20.Đồ thị bám của WL

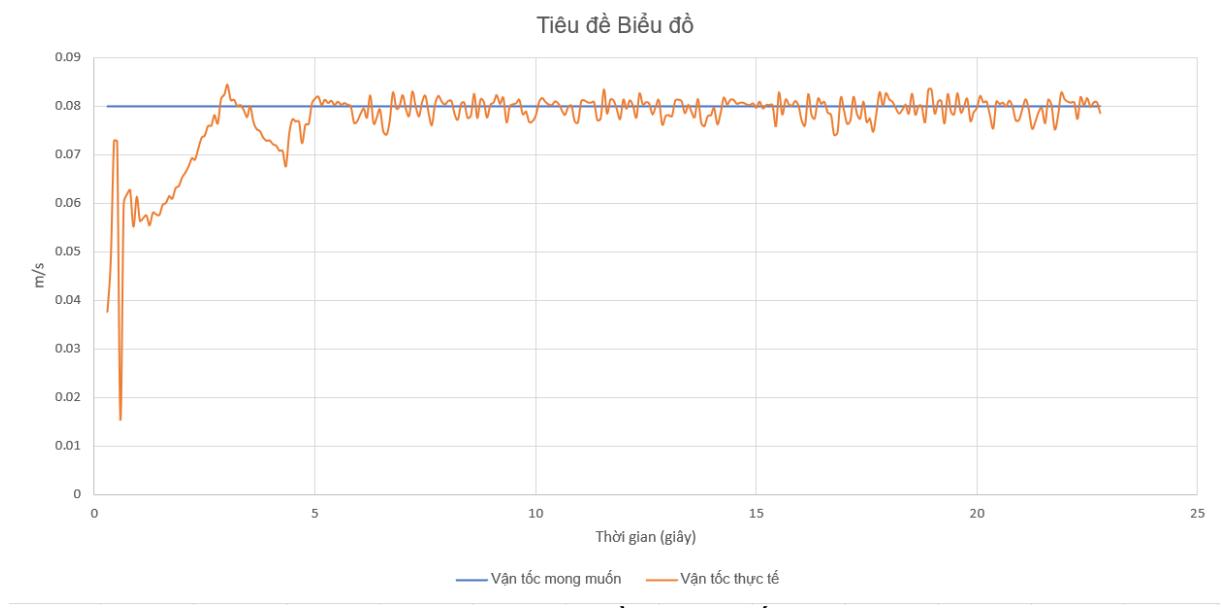
4.2. Kết quả chạy thực nghiệm:

4.2.1. Kết quả chạy thực nghiệm khi chạy trên đường đường thẳng:



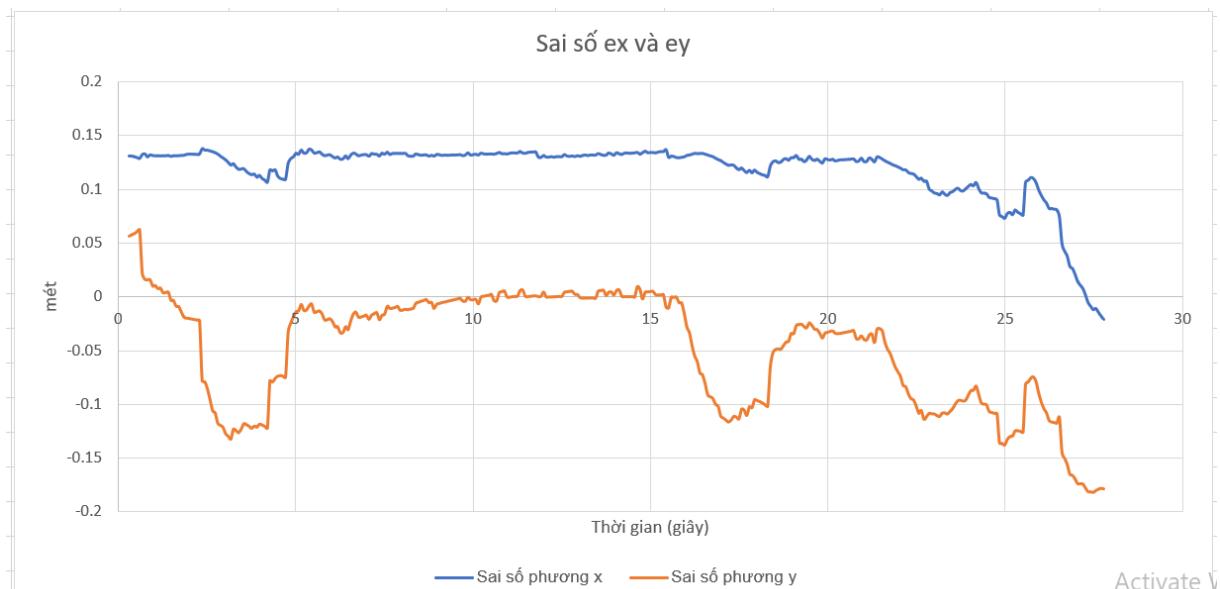


Hình 4.23.Đồ thị sai số góc theta



Hình 4.24.Đồ thị vận tốc

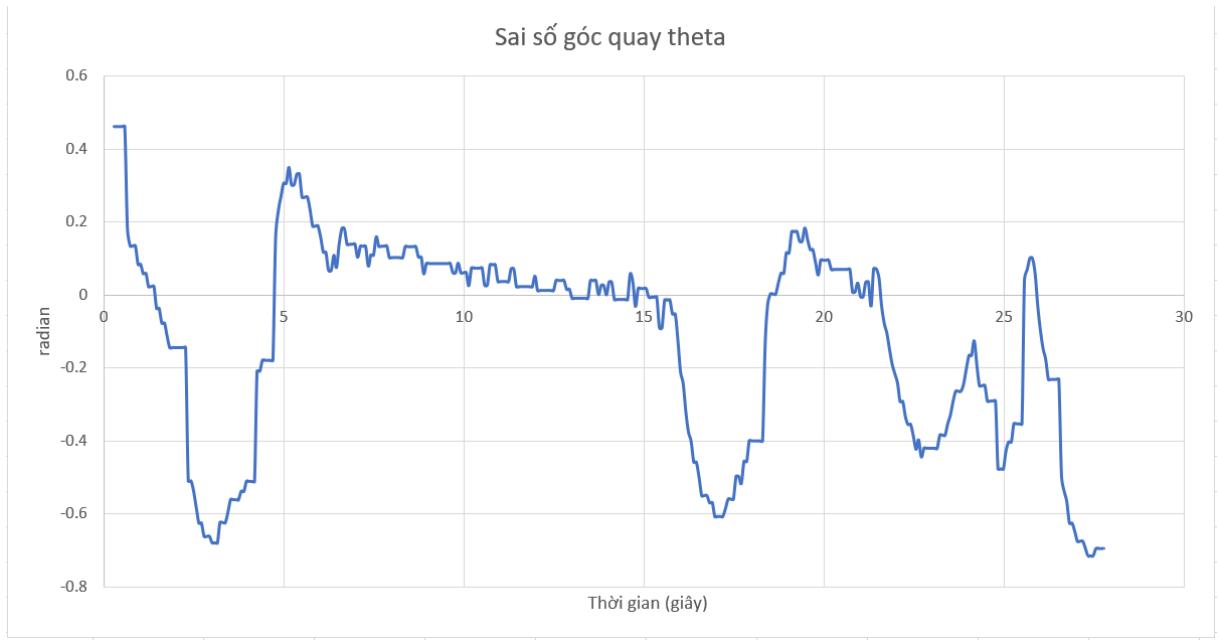
4.2.2. Kết quả chạy thực nghiệm khi chạy trên đường cong:



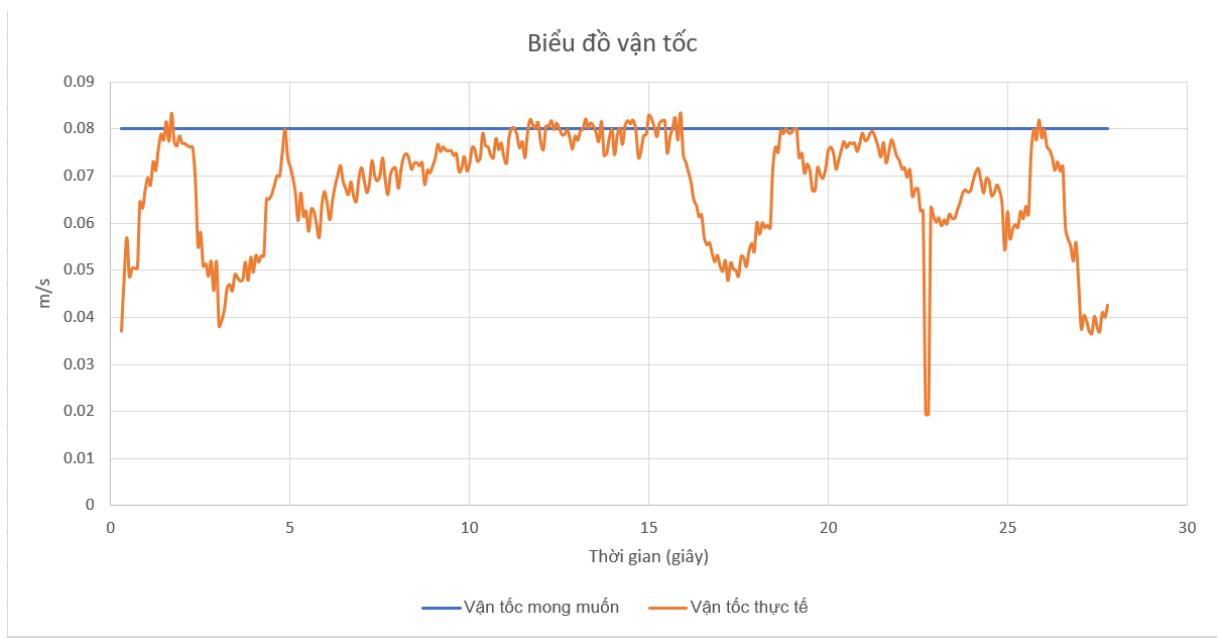
Hình 4.25.Đồ thị sai số x và y



Hình 4.26.Đồ thị sai số khoảng cách.



Hình 4.27.Đồ thị sai số theta

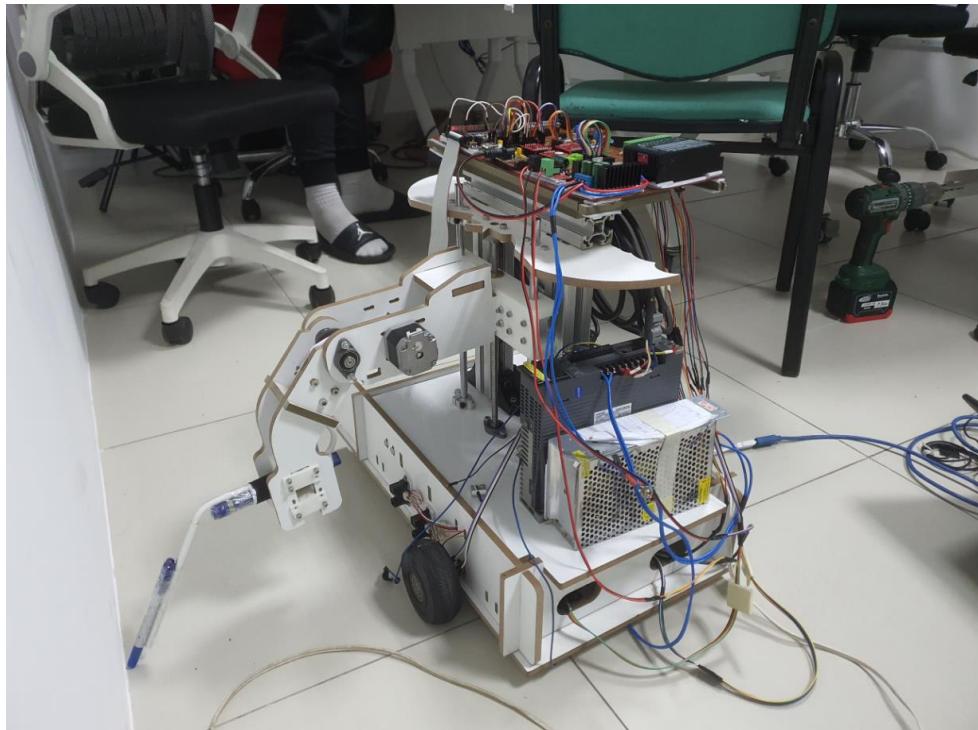


Hình 4.28.Đồ thị vận tốc:

CHƯƠNG 5: KẾT LUẬN, HƯỚNG PHÁT TRIỂN

5.1. Kết luận

- Sau khi trải qua thời gian nghiên cứu tìm hiểu và thực hiện đồ án. Nhóm đã thiết kế được robot hàn di động như hình:



Hình 5.1 Hình ảnh thực tế mô hình robot

- Kết quả đã đạt được:

+ Nhóm đã có thể thiết kế và thi công mô hình cho sản phẩm với vật liệu là gỗ và được lắp ráp hoàn thiện, các chi tiết kết nối chắc chắn, đảm bảo yêu cầu đặt ra.

+ Lựa chọn được đa phần những linh kiện điện tử phù hợp cho robot.

+ Lập trình cho robot chạy được trên mô phỏng matlab và thực tế với 2 chế độ chạy: Theo một quỹ đạo cho trước nhất định và bám theo tường hàn cả đường thẳng và đường cong.

5.2. Hạn chế của đề tài.

Với sự cố gắng của nhóm, đã thực hiện được những mục tiêu ban đầu đề ra nhưng vì lượng thời gian và kiến thức còn hạn chế nên khả năng hoàn thiện còn nhiều sai sót (khi

về xác lập hệ dao động, vận tốc chưa đạt ổn định, mạch điện chưa hoàn chỉnh, còn nhiều vị trí khó hàn xe chưa đáp ứng được...)

5.3. Phương pháp khắc phục và hướng phát triển

Sau khi nghiên cứu, nhóm có thể đề xuất một vài hướng nghiên cứu sau đây:

- Dùng xử lý ảnh để xác định đường hàn.
- Sử dụng bộ điều khiển mới (Lyapunov hay fuzzy) để tăng tính ổn định khi hàn.
- Kết nối với máy hàn bằng rs232 để điều chỉnh dòng điện hàn
- Thiết kế mạch PCB
- Nâng bậc tự do cánh tay để hàn những vị trí khó

TÀI LIỆU THAM KHẢO

- 1.Trịnh Chát và Lê Văn Uyễn, “Tính toán hệ thống dẫn động cơ khí”, NXB Giáo dục
- 2.Byoung-Oh Kam, Motion control of two-wheeled welding mobile robot with seam tracking sensor. Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on 2001
- 3.Ngô Mạnh Dũng, Robust Control of Welding Robot for Tracking a Rectangular Welding Line. International Journal of Advanced Robotic Systems, 2006.
- 4.Yang Bae Jeon, Modeling and Motion Control of Mobile Robot for Lattice Type Welding.
KSME International Journal, 2002.
- 5.Gonzalez de Santos, Ship building with ROWER. Robotics & Automation Magazine, IEEE, 2000. 7(4).
- 6.Bùi Trọng Hiếu, A Simple Nonlinear Control of a Two Wheeled Welding Mobile Robot.
International Journal of Control, Automation and Systems, 2003.
- 7.Phan Tân Tùng, Decentralized Control Design for Welding Mobile Manipulator. Journal of Mechanical Science and Technology, 2005.
- 8.Bùi Trọng Hiếu, Adaptive Tracking Control of Two-Wheeled Welding Mobile Robot with Smooth Curved Welding Path. KSME International Journal, 2003.
- 9.Ngô Mạnh Dũng, Two-Wheeled Welding Mobile Robot for Tracking a Smooth Curved Welding Path Using Adaptive Sliding-Mode Control Technique. International Journal of Control, Automation, and Systems, 2007.
- 10.Nguyễn Hữu Lộc, Cơ sở thiết kế máy, NXB Đại học quốc gia TPHCM, 2004.

PHỤ LỤC

Chương trình thực hiện

```
/* USER CODE BEGIN Header */
/*
 ****
 * @file          : main.c
 * @brief         : Main program body
 ****
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ****
 */
/* USER CODE END Header */
/* Includes ----- */
#include "main.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "stdint.h"
/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

CAN_HandleTypeDef hcan1;

TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;
TIM_HandleTypeDef htim4;
```

```

TIM_HandleTypeDef htim5;
TIM_HandleTypeDef htim8;
TIM_HandleTypeDef htim9;
TIM_HandleTypeDef htim10;
TIM_HandleTypeDef htim12;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
// PID banh xe
uint32_t AA;
uint32_t Encoder_A, Encoder_A_Pre, Encoder_B, Encoder_B_Pre;
float Kb = 900; //1935.5; // Antiwindup
float SP_WL, SP_WR; // Setpoint WR vs WL
float sum_A[20], sum_B[20];
float RPM_Avg_WL, RPM_Avg_WR;
int32_t delta_A, delta_B;
float SP_WL_Fake, SP_WR_Fake;
float RPM_WL_Math, RPM_WR_Math;
float SP_WL_Fake_2;
float WL_RPM, WR_RPM;
float WRL_Max = 1.75; // rad/s

// PID vi tri va bam goc
float V_counst = 0.08;
float V_max = 0.1125;
float EXY_2;
float Toa_do_X;
float Toa_do_Y;
float Var_SS_Theta;
float EX_Test, EY_Test;

// Cam bien
float Nguong_tren_set = 70;

// Bien thoi gian tong cua he welding mobile Robot
float Time_Total = 0;
float Time_Interrupt = 0.005;

// Thong so phan cung
uint16_t Pulse_Per_Round = 2688;
float R_Wheles = 0.0475;
float Khoang_tam_den_banh = 0.1825;// 0.365;
uint16_t HIGH_Limit_PWM = 999;
uint16_t LOW_Limit_PWM = 0;

// Cac bien phu
//1->
float GT_Convert_radian = 0.1047197551;
float PI = 3.1415926535897932;
//2->
int8_t bit_dao_chieu;
//3->
float M1, M2, M3, M4;
float K1, K2;
float Fake_Error_Y;
uint16_t pwm_ao;

```

```

// **** Cac struct ****
// PID
struct PID_Ver_1
{
    float HIGH, LOW;
    float Kp, Ki, Kd;
    float Up, Up_1;
    double Ui, Ui_1;
    float Ud, Ud_1;
    double Ui_Antiwindup;
    uint16_t Out_PWM;
    double Temp_PWM;
    float Out_fPWM;
    float Ti_le_WR_WL;
    float Kb

} PID_wl, PID_wr, PID_x, PID_y;

// Dong hoc Matlab
struct Kinematic_Matlab
{
    float vel, x_dot, y_dot, x, y, theta, theta_dot, b;
    float x1, y1, x1_dot, y1_dot, w1;
    float ax1, ay1;
    float w, Ratio_WR_WL;
    /////////////////////Quy hoach quy dao/////////////////
    float theta_Degree;
    float theta_1;
    float x_QH_welding, y_QH_welding;
    //////////////////Real Time/////////////////
    float vel_Real, x_dot_Real, y_dot_Real, w_Real;
    float x_Real, y_Real, x_welding_Real, y_welding_Real;
    float Theta_Fake;
    // He mm
    float x_mm_Real, y_mm_Real, x1_mm, y1_mm;

}Kinematic_Matlab_a;

// Sensor
struct Sensor
{
    // Cam bien
    float xA, yA;
    float xB, yB;
    float Theta_Sensor, Theta_Sensor_Degrees;
    // Khoang cach cam bien so voi diem tam xe
    float SS_Xa_1, SS_Xa_2;          // He mm
    float SS_Xa_1_Met, SS_Xa_2_Met; // He met

    // Gia tri cua ADC sau khi tinh ra mm
    float Khoang_cach_A1, Khoang_cach_A2; // He mm
    float A1_MET, A2_MET; // He met

    // Gia tri ADC
    int16_t ADC_A1, ADC_A2;
    int16_t ADC_Value[2];

    float ADC_AVG[2], ADC_AVG_Pre[2], ADC_Filter[2], ADC_Filter_Pre[2];
}

```

```

    int16_t Nguong_tren[2], Nguong_duo[2];
    int16_t Counter_nguong[2], Counter_Timer;

    // Gia tri trung binh
    float Avg_ADC_A1, Avg_ADC_A2;      // Gia tri trung binh
    float Sum_A1[20], Sum_A2[20];      // Noi chua mang ga tri

    float theta_XY, theta_XY_Pre;

}Sensor_1;

// Error
struct error_PID
{
    double ek, ek_1;
} Error_wl, Error_wr, Error_x, Error_y;

// Thong so PID he so theta
struct PID_Theta
{
    float Kp_D, Ki_D, Kd_D;      // Chon PID theo khoang cach
    float Kp_Parallel, Ki_Parallel, Kd_Parallel; // Chon PID bam song song
    voi tuong

    // Ti so van toc
    float Cv;
}PID_theta;

// AC Servo
struct AC_Servo
{
    int32_t Counter_AC, Diem_dung;
    int8_t DIR;
    float RPM;
    float Goc;
    float Khoang_cach_mm;
}AC_Servo_1, Stepper_1;

// Can bus
struct CANBUS
{
    CAN_TxHeaderTypeDef TX_Header;
    CAN_RxHeaderTypeDef RX_Header;
    CAN_FilterTypeDef Filetr_Can;
    uint32_t Mail_box[12];
    uint8_t Data_RX[10][8], Data_TX[10][8];
    float Data_RX_F[20], Data_TX_F[20];
}Can_bus;

// POINTER
struct PID_Ver_1* PID_WL = &PID_wl;
struct PID_Ver_1* PID_WR = &PID_wr;
struct error_PID* Error_WL = &Error_wl;
struct error_PID* Error_WR = &Error_wr;
struct error_PID* Error_X = &Error_x;
struct error_PID* Error_Y = &Error_y;
struct PID_Ver_1* PID_X = &PID_x;
struct PID_Ver_1* PID_Y = &PID_y;

```

```

struct Kinematic_Matlab* Kinematic_Matlab_A = &Kinematic_Matlab_a;
struct Sensor* SS = &Sensor_1;
struct PID_Theta* PID_t = &PID_theta;
struct AC_Servo* Ser = &AC_Servo_1;
struct AC_Servo* Step = &Stepper_1;
struct CANBUS* Can = &Can_bus;

/* USER CODE END PV */

/* Private function prototypes -----
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_CAN1_Init(void);
static void MX_TIM3_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM4_Init(void);
static void MX_TIM5_Init(void);
static void MX_TIM8_Init(void);
static void MX_TIM9_Init(void);
static void MX_TIM10_Init(void);
static void MX_TIM12_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----
/* USER CODE BEGIN 0 */
// printf
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif

PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, HAL_MAX_DELAY);
    return ch;
}

/*
                                         CAC FUNCTION
/**Cac ham khoi tao*/
void Init_PID()
{
    // PID khoang cach
    PID_X -> Kp = 0.23;
    PID_X -> Ki = 0.05;
    PID_X -> Kd = 0.12;
    // PID goc
    PID_t->Kp_D = 0.0305;
    PID_t->Ki_D = 0;
    PID_t->Kd_D = 0.015;

    PID_t->Kp_Parallel = 0.032;
    PID_t->Ki_Parallel = 0;
    PID_t->Kd_Parallel = 0.0025;
}

```

```

// PID 1 XY_Theta 0.018 0 0.0012    Theta_Desired 0.0095 0 0.0012
// Goc lon 0.001 0 0.0012
// Goc theta teo khoang cach
PID_Y -> Kp = PID_t->Kp_D;
PID_Y -> Ki = PID_t->Ki_D;
PID_Y -> Kd = PID_t->Kd_D;

// Goc theta theo bam song song voi tuong
PID_Y -> Kp = PID_t->Kp_Parallel; //Kp= 0.03, Kd = 0.18 ||

Kp = 0.0339, Ki = 0.189
PID_Y -> Ki = PID_t->Ki_Parallel;
PID_Y -> Kd = PID_t->Kd_Parallel;

// v = 0.2, k1 0.05 0 0.012
/*
 * So ghi chep thong so PID
 * Kp   Ki   Kd           ||      Kp   Ki   Kd
 * PID_t->Kp_D = 0.06325;
 * PID_t->Ki_D = 0;
 * PID_t->Kd_D = 0.015;
 *
 *
 */
}

void Initial_Cam_bien()
{
    SS->SS_Xa_1 = 245;      // Tam xe dem CB dau
    SS->SS_Xa_2 = 75;       // Tam xe dem CB cuoi
    SS->Nguong_tren[0] = 80;
    SS->Nguong_duois[0] = 20;
    SS->Nguong_tren[1] = 80;
    SS->Nguong_duois[1] = 20;
}

/*
 *                                     SENSOR
 */
void Vi_tri_Cam_bien(struct Sensor* S, struct Kinematic_Matlab* Math)
{
    float Temp[3];
    // Khoang cach 2 Cam bien theo phuong X cua xe
    Temp[0] = S->SS_Xa_1 - S->SS_Xa_2;
    //Khoang cach 2 cam bien theo phuong Y cua xe
    Temp[1] = S->Khoang_cach_A1 - S->Khoang_cach_A2;
    // Goc quay theta mong muon, Goc lech giua line va truc x cua xe
    S->Theta_Sensor = atan2(Temp[1],Temp[0]);
    S->Theta_Sensor_Degrees = S->Theta_Sensor * 180 / PI;
    // Chuyen mm sang met
    S->A1_MET = S->Khoang_cach_A1/1000.0;
    S->A2_MET = S->Khoang_cach_A2/1000.0;
    S->SS_Xa_1_Met = S->SS_Xa_1/1000.0;
    S->SS_Xa_2_Met = S->SS_Xa_2/1000.0;
    // Tinh toan Vi tri cua CB so vi truc toa do toan cuc
    S->x_A = Math->x_Real + S->SS_Xa_1_Met * cos(Math->theta) - (S->A1_MET +
    Khoang_tam_den_banh) * sin(Math->theta);
    S->y_A = Math->y_Real + (S->A1_MET + Khoang_tam_den_banh) * cos(Math-
    >theta) + (S->SS_Xa_1_Met * sin(Math->theta));
}

```

```

/*
 *          MATLAB KINEMATIC
 */
void Forward_Kinematic(float RPM_R, float RPM_L, struct Kinematic_Matlab*
Kinematic_Matlab_A)
{
    // Vận tốc thực của xe      v = ((Wr+Wl)*R)/(2)
    Kinematic_Matlab_A->vel_Real = (RPM_R+RPM_L) * R_Wheles/2;
    // Tốc độ quay của xe      w = ((Wr-Wl)*R)/(2*b)
    Kinematic_Matlab_A->w_Real = ((RPM_R-
RPM_L)*R_Wheles)/(2*Khoang_tam_den_banh);
    // Góc theta thực tế theo radian
    Kinematic_Matlab_A->theta = Kinematic_Matlab_A->theta + Kinematic_Matlab_A-
>w_Real *Time_Interrupt;
    // Góc theta thực tế theo độ
    Kinematic_Matlab_A->theta_Degree = Kinematic_Matlab_A->theta * 180 / PI;
    // X_dot vs Y_dot thực tế
    Kinematic_Matlab_A->x_dot_Real = Kinematic_Matlab_A->vel_Real *
cos(Kinematic_Matlab_A->theta);
    Kinematic_Matlab_A->y_dot_Real = Kinematic_Matlab_A->vel_Real *
sin(Kinematic_Matlab_A->theta);
}

float* Inverse_Kinematic(struct Kinematic_Matlab* Kinematic_Matlab_A, struct
PID_Ver_1* PID_X, struct PID_Ver_1* PID_Y, struct Quy_hoach* QHA)
{
    float* SP = (float*)malloc(2 * sizeof(float));
    SP[0] = (Kinematic_Matlab_A -> vel / R_Wheles) + (Kinematic_Matlab_A ->
w)/(R_Wheles*Khoang_tam_den_banh);
    SP[1] = (Kinematic_Matlab_A -> vel / R_Wheles) - (Kinematic_Matlab_A ->
w)/(R_Wheles*Khoang_tam_den_banh);
    return SP;
}

/*
 *          PID_FUNCTION_CONTROLLER
 */
/** Bô antiwindup để tắt khâu tích luỹ khi nó vượt ngưỡng **/
float Anti_Windup(float Out_PWM, uint16_t HIGH_Limit, uint16_t LOW_Limit, float
Kb)
{
    float e_reset = 0;
    float Ui_anti;

    if (Out_PWM > HIGH_Limit)
    {
        e_reset = (HIGH_Limit - Out_PWM );
    }
    else if (Out_PWM < LOW_Limit)
    {
        e_reset = (LOW_Limit - Out_PWM );
    }
    else
    {
        e_reset = 0;
    }
    Ui_anti = Time_Interrupt * e_reset * Kb;

    return Ui_anti;
}

```

```

/** PID Function **/
void PID_Function(float SP, float Real_Value, struct PID_Ver_1* PID, struct
error_PID* er, int8_t Anti)
{
    /* Sai so */
    // PID X,Y,Theta controller
    er -> ek = (SP - Real_Value);
    // PID
    // Khau P
    PID -> Up = PID -> Kp * er -> ek;
    /* Khau I ANTI WINDUP YES or NO */
    if(Anti == 0)
    {
        PID -> Ui = PID -> Ui_1 + PID_X -> Ki * er -> ek_1 *
Time_Interrupt;
    }
    else if (Anti == 1)
    {
        PID -> Ui_Antiwindup = Anti_Windup(PID -> Temp_PWM, PID->HIGH,
PID->LOW, Kb);
        PID -> Ui = PID -> Ui_1 + PID -> Ki * Error_X -> ek_1 *
Time_Interrupt + PID -> Ui_Antiwindup;
    }
    /* Khau D */
}
PID -> Ud = PID -> Ud_1 * (er -> ek - er -> ek_1);
/* Output PID */
PID -> Temp_PWM = PID -> Up + PID -> Ui + PID -> Ud; // output
FLOAT
PID -> Out_PWM = round(PID -> Temp_PWM); // output
output UINT16_t
/* LIMIT OUTPUT */
if(PID -> Out_PWM >= PID->HIGH)
{
    PID -> Out_PWM = PID->HIGH;
}
else if (PID -> Out_PWM <= PID->LOW )
{
    PID -> Out_PWM = PID->LOW;
}
}

/*
*/
void Linear_Path(float Time, struct Kinematic_Matlab* Matlab, float He_so_A, float
He_so_B)
{
    Matlab->x1 = He_so_A * Time;
    Matlab->y1 = He_so_B * Time;
}

/*
*/
/** Ham tinh trung binh ***/
float Average_5_times(float Var, float Temp[20])
{
    float sum = 0, Out_Average_Var; // Initialize sum to 0
    for (int i = 0; i < 19; i++)

```

```

    {
        Temp[i] = Temp[i + 1]; // gán giá trị hiện tại vào giá trị trước
        sum += Temp[i]; // Cộng dồn các giá trị vừa lưu
    }
    // Gán giá trị mới nhất
    Temp[19] = Var;
    sum += Temp[19];
    // Tính trung bình
    Out_Average_Var = sum / 20;
    return Out_Average_Var;
}
/** Tiến lui banh phải**/
void Tien_lui_DC_RIGHT(int8_t BIT.DAO)
{
    if(BIT.DAO == 1)
    {
        // IN1|IN2 => PE8|PE9
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, SET);
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, RESET);
    }
    else if(BIT.DAO == -1)
    {
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_8, RESET);
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_9, SET);
    }
}
/** Tiến lui banh trái **/
void Tien_lui_DC_LEFT(int8_t BIT.DAO)
{
    if(BIT.DAO == 1)
    {
        // IN3|IN4 => PE10|PE11
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_10, SET);
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_11, RESET);
    }
    else if(BIT.DAO == -1)
    {
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_10, RESET);
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_11, SET);
    }
}
/** Tính giá trị trung bình omega **/
void Average_PID_Omega( uint32_t Counter_Max, float Real_RPM, float AVG_RPM,
int32_t Delta, uint32_t Encoder, uint32_t Encoder_Pre, float Sum[20])
{
    int32_t DELTA_TEMP = 0;
    // Lấy chênh lệch 2 khoảng Encoder
    DELTA_TEMP = abs(Encoder - Encoder_Pre);
    // Nếu giá trị chênh lệch lớn hơn nữa số xung nữa vòng là vô lý nên lấy giá
    tri tràn trừ cho chênh lệch
    if (DELTA_TEMP >= 32000) //Counter_Max / 2
    {
        DELTA_TEMP = Counter_Max - DELTA_TEMP;
    }
    //Tính giá trị RPM hiện tại
    AVG_RPM = ((DELTA_TEMP) / (DELTA_TEMP * 4 * Time_Interrupt) * 60) *
GT_Convert_radian;
    // Tính trung bình
    Real_RPM = Average_5_times(AVG_RPM, Sum);
}

```

```

        Delta = DELTA_TEMP;
    }

/********************* AC Servo *****/
/** Chieu quay cua servo */
void DIR_Servo(GPIO_TypeDef * Port, uint16_t GPIO_PIN, int8_t DIR)
{
    if(DIR == 1)
    {
        HAL_GPIO_WritePin(Port, GPIO_PIN, SET);
    }
    else if(DIR == -1)
    {
        HAL_GPIO_WritePin(Port, GPIO_PIN, RESET);
    }
}

/** Xung cua Servo */
void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim)
{
    /**
     * AC Servo
     */
    if(htim->Instance == TIM10 && htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1 )
    {
        if(Ser->Counter_AC == Ser->Diem_dung)
        {
            __HAL_TIM_SET_COMPARE(&htim10, TIM_CHANNEL_1, 0);
        }
        else if(Ser->Counter_AC < Ser->Diem_dung)
        {
            Ser->DIR = 1;
            DIR_Servo(PULSE_AC_GPIO_Port, PULSE_AC_Pin, Ser->DIR);
            __HAL_TIM_SET_COMPARE(&htim10, TIM_CHANNEL_1, 49);
            Ser->Counter_AC+=1;
        }
        else if(Ser->Counter_AC > Ser->Diem_dung)
        {
            Ser->DIR = -1;
            __HAL_TIM_SET_COMPARE(&htim10, TIM_CHANNEL_1, 49);
            DIR_Servo(PULSE_AC_GPIO_Port, PULSE_AC_Pin, Ser->DIR);
            Ser->Counter_AC-=1;
        }
    }
    /**
     * Stepper
     */
    else if(htim->Instance == TIM12 && htim->Channel ==
HAL_TIM_ACTIVE_CHANNEL_1 )
    {
        if(Step->Counter_AC == Step->Diem_dung)
        {
            __HAL_TIM_SET_COMPARE(&htim12, TIM_CHANNEL_1, 0);
        }
        else if(Step->Counter_AC < Step->Diem_dung)
        {
            Step->DIR = 1;
            DIR_Servo(DIR_STEPPER_GPIO_Port, DIR_STEPPER_Pin, Step->DIR);
            __HAL_TIM_SET_COMPARE(&htim12, TIM_CHANNEL_1, 49);
        }
    }
}

```

```

                Step->Counter_AC+=1;
            }
        else if(Step->Counter_AC > Step->Diem_dung)
        {
            Step->DIR = -1;
            __HAL_TIM_SET_COMPARE(&htim12, TIM_CHANNEL_1, 49);
            DIR_Servo(DIR_STEPPER_GPIO_Port, DIR_STEPPER_Pin, Step->DIR);
            Step->Counter_AC-=1;
        }
    }
}

/* Ham chay Servo theo vi tri */
void AC_Servo(float Khoang_cach_mm)
{
    Ser->Diem_dung = (int32_t)(Khoang_cach_mm*1000)/(2*8);
}

/* Ham chay Stepper */
//Vi_buoc = x2, x4, x8
void Stepper_Motor(float Goc, float Vi_buoc)
{
    Step->Diem_dung = (int32_t)(Goc*Vi_buoc/1.8);
}

***** CAN BUS *****
***** Can configue ****/
void Init_Can_bus()
{
    Can->TX_Header.DLC = 8;
    Can->TX_Header.IDE = CAN_ID_STD; // 0x00 = CAN_ID_STD
    Can->TX_Header.RTR = CAN_RTR_DATA ; //0x00 = CAN_RTR_DATA
    Can->TX_Header.StdId = 0xA2;
}

/* Ham nhan du lieu */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &(Can->TX_Header), (uint8_t *)
Can->Data_TX);
    if(Can->RX_Header.StdId == 0x103)
    {
        // Processing Data
    }
}

/* Ham chuyen 8 byte thanh so thuc */
float Combine_Data(uint8_t Data[8])
{
    float RS_nguyen, RS_thap_phan, RS;
    uint32_t A[4];
    uint32_t Temp[8] = {0};
    // Phan nguyen co 24 bit
    Temp[0] = (uint32_t)((Data[1]<<16)&0xFF0000);
    Temp[1] = (uint32_t)(Data[2]<<8)&0xFF00;
    Temp[2] = (uint32_t)(Data[3])&0xFF;
    A[0] = Temp[0] | Temp[1] | Temp[2];
    RS_nguyen = (float)A[0] ;
    // Phan thap phan 32 bit
}

```

```

        Temp[3] = (uint32_t)((Data[4]<<24)&0xFF000000);
        Temp[4] = (uint32_t)(Data[5]<<16)&0xFF0000;
        Temp[5] = (uint32_t)(Data[6]<<8)&0xFF00;
        Temp[6] = (uint32_t)(Data[7])&0xFF;
        A[1] = Temp[3] | Temp[4] | Temp[5] | Temp[6];
        RS_thap_phan = A[1];
        // Gop 2 phan thap phan va nguyen lai voi nhau
        RS = RS_nguyen + RS_thap_phan / pow(10,9); // Assuming 32-bit precision for
fractional part

        if(Data[0] == 1)
        {
            RS = RS * (-1);
        }
        return RS;
    }

/** Ham chuyen float thanh 8 byte de Can gui du lieu */
void Float_to_int(float A, uint8_t Data_ALL[8])
{
    uint32_t Temp_32[4];
    int32_t Result[10];

    // Kiem tra gia tri am
    // Data daaus
    if (A < 0)
    {
        Result[0] = -1;
        Data_ALL[0] = 1; // Dau am
        A = -A;
    }
    else
    {
        Result[0] = 1;
        Data_ALL[0] = 0; // Dau duong
    }

    Temp_32[0] = floor(A); // Phan nguyen
    Temp_32[1] = (A - (float)Temp_32[0]) * pow(10, 9); // Phan thap phan

    // Xu ly phan nguyen
    Data_ALL[1] = (Temp_32[0] >> 16)&0xFF;
    Data_ALL[2] = (Temp_32[0] >> 8)&0xFF;
    Data_ALL[3] = (Temp_32[0])&0xFF;
    // Xu ly phan thap phan
    Data_ALL[4] = (Temp_32[1] >> 24)&0xFF;
    Data_ALL[5] = (Temp_32[1] >> 16)&0xFF;
    Data_ALL[6] = (Temp_32[1] >> 8)&0xFF;
    Data_ALL[7] = Temp_32[1] &0xFF;
}

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

```

```

/*
 * Gán giá trị PID
 // PID_WL -> Kp = 0.0671641791;
 // PID_WL -> Ki = 129.9951014957;
 // PID_WL -> Kd = 0;
 // PID_WR -> Kp = 0.0671641791;
 // PID_WR -> Ki = 129.9951014957;
 // PID_WR -> Kd = 0;
 PID_WL -> Kp = 459.2872592;
 PID_WL -> Ki = 1494.79277;
 PID_WL -> Kd = 0;
 PID_WL -> Kb = 3.254592285;
 PID_WR -> Kp = 472.7680942;
 PID_WR -> Ki = 1477.44652;
 PID_WR -> Kd = 0;
 PID_WR -> Kb = 3.125098831;
 // Gan gia tri dau
 PID_WL->Ui_1 = 0;
 PID_WR->Ui_1 = 0;
 PID_WL -> Ud_1 = 0;
 PID_WR -> Ud_1 = 0;
 // HIGH LOW PWM
 PID_WL -> HIGH = 999;
 PID_WL -> LOW = 0;
 PID_WR -> HIGH = 999;
 PID_WR -> LOW = 0;
 // Cac gia tri goc vs vi tri ban dau
 Kinematic_Matlab_A->theta = 0;
 Kinematic_Matlab_A->b = 0.0001;
 Kinematic_Matlab_A->Ratio_WR_WL = 1;
 Kinematic_Matlab_A->x_Real = 0.0;
 Kinematic_Matlab_A->y_Real = 0.0;

/* USER CODE END 1 */

/* MCU Configuration-----*/
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_ADC1_Init();
MX_CAN1_Init();

```

```

MX_TIM3_Init();
MX_USART1_UART_Init();
MX_TIM2_Init();
MX_TIM4_Init();
MX_TIM5_Init();
MX_TIM8_Init();
MX_TIM9_Init();
MX_TIM10_Init();
MX_TIM12_Init();
/* USER CODE BEGIN 2 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);

// **** Timer Interrupt ****
HAL_TIM_Base_Start_IT(&htim4); // Timer chinh PID
HAL_TIM_Base_Start_IT(&htim5); // Timer cap nhat cam bien va vi tri

// **** Encoder Mode ****
HAL_TIM_Encoder_Start_IT(&htim2, TIM_CHANNEL_1 | TIM_CHANNEL_2);
HAL_TIM_Encoder_Start_IT(&htim3, TIM_CHANNEL_1 | TIM_CHANNEL_2);

// **** Stepper & AC Servo ****
HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1); // WR
HAL_TIM_PWM_Start(&htim9, TIM_CHANNEL_1); // WL

// **** PID dong co ****
HAL_TIM_PWM_Start(&htim12, TIM_CHANNEL_1); // WR
HAL_TIM_PWM_Start(&htim10, TIM_CHANNEL_1); // WL

// **** CAN BUS ****
HAL_CAN_Start(&hcan1); // Init Can
HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING); // Active
Interrupt Mode

// **** Khoi tao cac ham ****
Initial_Cam_bien(); // Khoi tao cac gia tri cam bien
Init_PID(); // Khoi tao cac gia tri Kp, Ki, Kd
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
        printf("%10f, %10f, %10f, %10f, %10f, %10f \n", Time_Total,
EX_Test, EY_Test, Error_X->ek, Error_Y->ek, Kinematic_Matlab_A->vel,
Kinematic_Matlab_A->vel_Real);
        // SS->theta_XY, SP_WL - WL_RPM, RPM_WR_Math, SP_WL_Fake
        // printf("%10f, %10f, %10f, %10f \n", Time_Total, SS-
>theta_XY, RPM_WL_Math, RPM_WL_Math);
        // Tien_lui_DC_LEFT(1);
        // Tien_lui_DC_RIGHT(1);
        // __HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1, pwm_ao);
        // __HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1, pwm_ao);
        HAL_Delay(50);
    }
    /* USER CODE END 3 */
}

```

```

}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 4;
    RCC_OscInitStruct.PLL.PLLN = 144;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /** Initializes the CPU, AHB and APB buses clocks
    */
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
    {
        Error_Handler();
    }

    /**
     * @brief ADC1 Initialization Function
     * @param None
     * @retval None
     */
    static void MX_ADC1_Init(void)
    {

        /* USER CODE BEGIN ADC1_Init_0 */

        /* USER CODE END ADC1_Init_0 */
    }
}

```

```

ADC_ChannelConfTypeDef sConfig = {0};

/* USER CODE BEGIN ADC1_Init_1 */

/* USER CODE END ADC1_Init_1 */

/** Configure the global features of the ADC (Clock, Resolution, Data Alignment
and number of conversion)
*/
hadc1.Instance = ADC1;
hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
hadc1.Init.Resolution = ADC_RESOLUTION_12B;
hadc1.Init.ScanConvMode = ENABLE;
hadc1.Init.ContinuousConvMode = ENABLE;
hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc1.Init.NbrOfConversion = 2;
hadc1.Init.DMAContinuousRequests = DISABLE;
hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
if (HAL_ADC_Init(&hadc1) != HAL_OK)
{
    Error_Handler();
}

/** Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
*/
sConfig.Channel = ADC_CHANNEL_0;
sConfig.Rank = 1;
sConfig.SamplingTime = ADC_SAMPLETIME_112CYCLES;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
    Error_Handler();
}

/** Configure for the selected ADC regular channel its corresponding rank in the
sequencer and its sample time.
*/
sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = 2;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN ADC1_Init_2 */

/* USER CODE END ADC1_Init_2 */

}

/**
 * @brief CAN1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_CAN1_Init(void)
{

```

```

/* USER CODE BEGIN CAN1_Init_0 */

/* USER CODE END CAN1_Init_0 */

/* USER CODE BEGIN CAN1_Init_1 */

/* USER CODE END CAN1_Init_1 */
hcan1.Instance = CAN1;
hcan1.Init.Prescaler = 10;
hcan1.Init.Mode = CAN_MODE_NORMAL;
hcan1.Init.SyncJumpWidth = CAN SJW_1TQ;
hcan1.Init.TimeSeg1 = CAN_BS1_2TQ;
hcan1.Init.TimeSeg2 = CAN_BS2_3TQ;
hcan1.Init.TimeTriggeredMode = DISABLE;
hcan1.Init.AutoBusOff = DISABLE;
hcan1.Init.AutoWakeUp = DISABLE;
hcan1.Init.AutoRetransmission = DISABLE;
hcan1.Init.ReceiveFifoLocked = DISABLE;
hcan1.Init.TransmitFifoPriority = DISABLE;
if (HAL_CAN_Init(&hcan1) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN CAN1_Init_2 */
Can->Filetr_Can.FilterBank = 0; // Filter number
Can->Filetr_Can.FilterMode = CAN_FILTERMODE_IDMASK; // Identifier mask mode
Can->Filetr_Can.FilterScale = CAN_FILTERSCALE_32BIT; // 32-bit scale for
identifiers
Can->Filetr_Can.FilterMaskIdHigh = 0x0000; // High 16 bits of the filter mask
Can->Filetr_Can.FilterMaskIdLow = 0x0000; // Low 16 bits of the filter mask
Can->Filetr_Can.FilterIdHigh = 0x0000; // High 16 bits of the filter ID
Can->Filetr_Can.FilterIdLow = 0x0000; // Low 16 bits of the filter ID
Can->Filetr_Can.FilterFIFOAssignment = CAN_FILTER_FIFO0; // Assign to FIFO 0
Can->Filetr_Can.FilterActivation = ENABLE;
Can->Filetr_Can.SlaveStartFilterBank = 0;

HAL_CAN_ConfigFilter(&hcan1, &Can->Filetr_Can);
/* USER CODE END CAN1_Init_2 */

}

/**
 * @brief TIM2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM2_Init(void)
{
    /* USER CODE BEGIN TIM2_Init_0 */

    /* USER CODE END TIM2_Init_0 */

    TIM_Encoder_InitTypeDef sConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};

    /* USER CODE BEGIN TIM2_Init_1 */

```

```

/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 0;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 65535;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
sConfig.EncoderMode = TIM_ENCODERMODE_TI1;
sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
sConfig.IC1Filter = 0;
sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
sConfig.IC2Filter = 0;
if (HAL_TIM_Encoder_Init(&htim2, &sConfig) != HAL_OK)
{
    Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN TIM2_Init 2 */

/* USER CODE END TIM2_Init 2 */

}

/**
 * @brief TIM3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM3_Init(void)
{
    /* USER CODE BEGIN TIM3_Init 0 */

    /* USER CODE END TIM3_Init 0 */

    TIM_Encoder_InitTypeDef sConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};

    /* USER CODE BEGIN TIM3_Init 1 */

    /* USER CODE END TIM3_Init 1 */
    htim3.Instance = TIM3;
    htim3.Init.Prescaler = 0;
    htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim3.Init.Period = 65535;
    htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    sConfig.EncoderMode = TIM_ENCODERMODE_TI1;
    sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
    sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;

```

```

sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
sConfig.IC1Filter = 0;
sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
sConfig.IC2Filter = 0;
if (HAL_TIM_Encoder_Init(&htim3, &sConfig) != HAL_OK)
{
    Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN TIM3_Init_2 */

/* USER CODE END TIM3_Init_2 */

}

/**
 * @brief TIM4 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM4_Init(void)
{
    /* USER CODE BEGIN TIM4_Init_0 */

    /* USER CODE END TIM4_Init_0 */

    TIM_ClockConfigTypeDef sClockSourceConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};

    /* USER CODE BEGIN TIM4_Init_1 */

    /* USER CODE END TIM4_Init_1 */
    htim4.Instance = TIM4;
    htim4.Init.Prescaler = 1439;
    htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim4.Init.Period = 499;
    htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim4.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    if (HAL_TIM_Base_Init(&htim4) != HAL_OK)
    {
        Error_Handler();
    }
    sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
    if (HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig) != HAL_OK)
    {
        Error_Handler();
    }
    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig) != HAL_OK)
    {

```

```

        Error_Handler();
    }
/* USER CODE BEGIN TIM4_Init_2 */

/* USER CODE END TIM4_Init_2 */

}

/** 
 * @brief TIM5 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM5_Init(void)
{
    /* USER CODE BEGIN TIM5_Init_0 */

    /* USER CODE END TIM5_Init_0 */

    TIM_ClockConfigTypeDef sClockSourceConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};

    /* USER CODE BEGIN TIM5_Init_1 */

    /* USER CODE END TIM5_Init_1 */
    htim5.Instance = TIM5;
    htim5.Init.Prescaler = 1439;
    htim5.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim5.Init.Period = 1499;
    htim5.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim5.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    if (HAL_TIM_Base_Init(&htim5) != HAL_OK)
    {
        Error_Handler();
    }
    sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
    if (HAL_TIM_ConfigClockSource(&htim5, &sClockSourceConfig) != HAL_OK)
    {
        Error_Handler();
    }
    sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
    sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
    if (HAL_TIMEx_MasterConfigSynchronization(&htim5, &sMasterConfig) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN TIM5_Init_2 */

    /* USER CODE END TIM5_Init_2 */

}

/** 
 * @brief TIM8 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM8_Init(void)

```

```

{

/* USER CODE BEGIN TIM8_Init_0 */

/* USER CODE END TIM8_Init_0 */

TIM_MasterConfigTypeDef sMasterConfig = {0};
TIM_OC_InitTypeDef sConfigOC = {0};
TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};

/* USER CODE BEGIN TIM8_Init_1 */

/* USER CODE END TIM8_Init_1 */
htim8.Instance = TIM8;
htim8.Init.Prescaler = 6;
htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
htim8.Init.Period = 999;
htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim8.Init.RepetitionCounter = 0;
htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_PWM_Init(&htim8) != HAL_OK)
{
    Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL_OK)
{
    Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
if (HAL_TIM_PWM_ConfigChannel(&htim8, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
    Error_Handler();
}
sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
sBreakDeadTimeConfig.DeadTime = 0;
sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
if (HAL_TIMEx_ConfigBreakDeadTime(&htim8, &sBreakDeadTimeConfig) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN TIM8_Init_2 */

/* USER CODE END TIM8_Init_2 */
HAL_TIM_MspPostInit(&htim8);

}

```

```

/**
 * @brief TIM9 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM9_Init(void)
{
    /* USER CODE BEGIN TIM9_Init_0 */

    /* USER CODE END TIM9_Init_0 */

    TIM_OC_InitTypeDef sConfigOC = {0};

    /* USER CODE BEGIN TIM9_Init_1 */

    /* USER CODE END TIM9_Init_1 */
    htim9.Instance = TIM9;
    htim9.Init.Prescaler = 6;
    htim9.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim9.Init.Period = 999;
    htim9.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim9.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    if (HAL_TIM_PWM_Init(&htim9) != HAL_OK)
    {
        Error_Handler();
    }
    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 0;
    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
    sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
    if (HAL_TIM_PWM_ConfigChannel(&htim9, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN TIM9_Init_2 */

    /* USER CODE END TIM9_Init_2 */
    HAL_TIM_MspPostInit(&htim9);
}

/**
 * @brief TIM10 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM10_Init(void)
{
    /* USER CODE BEGIN TIM10_Init_0 */

    /* USER CODE END TIM10_Init_0 */

    TIM_OC_InitTypeDef sConfigOC = {0};

    /* USER CODE BEGIN TIM10_Init_1 */

    /* USER CODE END TIM10_Init_1 */
}

```

```

htim10.Instance = TIM10;
htim10.Init.Prescaler = 23;
htim10.Init.CounterMode = TIM_COUNTERMODE_UP;
htim10.Init.Period = 99;
htim10.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim10.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim10) != HAL_OK)
{
    Error_Handler();
}
if (HAL_TIM_PWM_Init(&htim10) != HAL_OK)
{
    Error_Handler();
}
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim10, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN TIM10_Init_2 */

/* USER CODE END TIM10_Init_2 */
HAL_TIM_MspPostInit(&htim10);

}

/**
 * @brief TIM12 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM12_Init(void)
{
    /* USER CODE BEGIN TIM12_Init_0 */

    /* USER CODE END TIM12_Init_0 */

    TIM_OC_InitTypeDef sConfigOC = {0};

    /* USER CODE BEGIN TIM12_Init_1 */

    /* USER CODE END TIM12_Init_1 */
    htim12.Instance = TIM12;
    htim12.Init.Prescaler = 1439;
    htim12.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim12.Init.Period = 99;
    htim12.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    htim12.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
    if (HAL_TIM_PWM_Init(&htim12) != HAL_OK)
    {
        Error_Handler();
    }
    sConfigOC.OCMode = TIM_OCMODE_PWM1;
    sConfigOC.Pulse = 0;
    sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
}

```

```

sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
if (HAL_TIM_PWM_ConfigChannel(&htim12, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN TIM12_Init_2 */

/* USER CODE END TIM12_Init_2 */
HAL_TIM_MspPostInit(&htim12);

}

/**
 * @brief USART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART1_UART_Init(void)
{

/* USER CODE BEGIN USART1_Init_0 */

/* USER CODE END USART1_Init_0 */

/* USER CODE BEGIN USART1_Init_1 */

/* USER CODE END USART1_Init_1 */
huart1.Instance = USART1;
huart1.Init.BaudRate = 9600;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart1) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN USART1_Init_2 */

/* USER CODE END USART1_Init_2 */

}

/**
 * Enable DMA controller clock
 */
static void MX_DMA_Init(void)
{
    /* DMA controller clock enable */
    __HAL_RCC_DMA2_CLK_ENABLE();

    /* DMA interrupt init */
    /* DMA2_Stream0_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);
}

```

```

}

/***
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOE_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOE, IN1_Pin|IN2_Pin|IN3_Pin|IN4_Pin
|DIR_SERVO_Pin|LIGHT_WARNING_Pin|LIGHT_STOP_Pin|LIGHT_RUN_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(DIR_STEPPER_GPIO_Port, DIR_STEPPER_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pins : IN1_Pin IN2_Pin IN3_Pin IN4_Pin
                           DIR_SERVO_Pin LIGHT_WARNING_Pin LIGHT_STOP_Pin
                           LIGHT_RUN_Pin */
    GPIO_InitStruct.Pin = IN1_Pin|IN2_Pin|IN3_Pin|IN4_Pin
|DIR_SERVO_Pin|LIGHT_WARNING_Pin|LIGHT_STOP_Pin|LIGHT_RUN_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

    /*Configure GPIO pin : DIR_STEPPER_Pin */
    GPIO_InitStruct.Pin = DIR_STEPPER_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(DIR_STEPPER_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
// Ngat Timer PID
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim -> Instance == TIM4)
    {

        // Sai số điểm hàn theo phương X vs Y
        EX_Test = ( SS->xA - Kinematic_Matlab_A->x_welding_Real);
    }
}

```

```

EY_Test = ( SS->yA - Kinematic_Matlab_A->y_welding_Real);
EXY_2 = pow(EX_Test,2) + pow(EY_Test,2);
Error_X -> ek = sqrt(Exy_2);

// PID Góc theta
// ◊?i◊?u kiện để đổi bộ PID
// PID theta bám vi trí XY
if(Error_X -> ek >= 0.035)
{
    SS->theta_XY = atan2((SS->A1_MET - 0.35), 0.135);
    Fake_Error_Y = (SS->theta_XY);
    Error_Y -> ek = Fake_Error_Y;
    //Error_Y -> ek = (SS->theta_XY);
    PID_Y->Kp = PID_t->Kp_D * PID_t->Cv;
    PID_Y->Ki = PID_t->Ki_D * PID_t->Cv;
    PID_Y->Kd = PID_t->Kd_D * PID_t->Cv;
    // ◊?i◊?u kiện khi góc theta thay đổi âm dương liên tục
    if(SS->Khoang_cach_A1 > 580.0)
    {
        PID_Y->Kp = 0.015;
    }

    // Điều kiện quay goc duong am
    if(SS->A1_MET < SS->A2_MET)
    {
        Error_Y -> ek = -Fake_Error_Y;
    }
    else if(SS->A1_MET >= SS->A2_MET)
    {
        Error_Y -> ek = Fake_Error_Y;
    }
}

// PID theta bám song song với tு?ng
else if(Error_X -> ek < 0.035)
{
    Error_Y -> ek = (SS->Theta_Sensor);
    PID_Y->Kp = PID_t->Kp_Parallel;
    PID_Y->Ki = PID_t->Ki_Parallel;
    PID_Y->Kd = PID_t->Kd_Parallel;
}

// Thực thi bộ PID theta
PID_Y -> Up = PID_Y -> Kp * Error_Y -> ek;
PID_Y -> Ui = PID_Y -> Ui_1 + PID_Y -> Ki * Error_Y -> ek_1 *
Time_Interrupt;
//PID_Y -> Ui_Antiwindup;
PID_Y -> Ud = PID_Y -> Kd * (Error_Y -> ek - Error_Y ->
ek_1)/Time_Interrupt;
// Antiwindup
//PID_Y -> Ui_Antiwindup = Anti_Windup(PID_Y -> Temp_PWM, HIGH_Limit_PWM,
LOW_Limit_PWM, Kb);
PID_Y -> Out_fPWM = PID_Y -> Up + PID_Y -> Ui + PID_Y -> Ud;
// Gan lai gia tri
Error_Y -> ek_1 = Error_Y -> ek;
PID_Y -> Ui_1 = PID_Y -> Ui;
PID_Y -> Ud_1 = PID_Y -> Ud;

```

```

// Vận tốc và Omega W
// Cập nhật vận tốc
Kinematic_Matlab_A->vel = V_counst;
// Hệ số Cv thay đổi theo vận tốc
PID_t->Cv = (Kinematic_Matlab_A->vel/(V_max));
// Cập nhật Omega
Kinematic_Matlab_A->w = PID_Y -> Out_PWM;

// Tốc bộ bánh xe
// Cập nhật WR vs WL theo PID
SP_WR_Fake = (Kinematic_Matlab_A->vel/R_Wheles) + (Kinematic_Matlab_A->w)/(R_Wheles*Khoang_tam_den_banh);
SP_WL_Fake = (Kinematic_Matlab_A->vel/R_Wheles) - (Kinematic_Matlab_A->w)/(R_Wheles*Khoang_tam_den_banh);

/****************************************/
// Banh trai chạy theo banh phải
if(SP_WR_Fake == 0)
{
    PID_WL->Ti_le_WR_WL = 0;

}
else if(SP_WR_Fake != 0)
{
    PID_WL->Ti_le_WR_WL = SP_WL_Fake / SP_WR_Fake;
}

if(WR_RPM == 0)
{
    SP_WL_Fake_2 = SP_WL_Fake;
}
else if(WR_RPM != 0)
{
    SP_WL_Fake_2 = PID_WL->Ti_le_WR_WL * WR_RPM;
}
/****************************************/

// Limit output PID vị trí
if(SP_WR_Fake >= WRL_Max)
{
    SP_WR_Fake = WRL_Max;
}
else if(SP_WR_Fake <= -WRL_Max)
{
    SP_WR_Fake = -WRL_Max;
}
if(SP_WL_Fake >= WRL_Max)
{
    SP_WL_Fake = WRL_Max;
}
else if(SP_WL_Fake <= -WRL_Max)
{
    SP_WL_Fake = -WRL_Max;
}

// ♦?ối chỉ♦?u chân dir khi âm
if(SP_WL_Fake < 0)
{
    Tien_lui_DC_RIGHT(-1); // Tien la 1 Lui la -1
}

```

```

        SP_WL = -SP_WL_Fake;
        RPM_WL_Math = -WL_RPM;
    }
    else if(SP_WL_Fake >= 0)
    {
        Tien_lui_DC_RIGHT(1);
        SP_WL = SP_WL_Fake;
        RPM_WL_Math = WL_RPM;
    }
    if(SP_WR_Fake < 0)
    {
        Tien_lui_DC_LEFT(-1);
        SP_WR = -SP_WR_Fake;
        RPM_WR_Math = -WR_RPM;
    }
    else if(SP_WR_Fake >= 0)
    {
        Tien_lui_DC_LEFT(1);
        SP_WR = SP_WR_Fake;
        RPM_WR_Math = WR_RPM;
    }

// PID bánh xe
// Cập nhật Encoder
Encoder_B = __HAL_TIM_GET_COUNTER(&htim2);
// Lấy chênh lệch 2 khoảng Encoder
delta_B = abs(Encoder_B - Encoder_B_Pre);
// Nếu giá trị chênh lệch lớn hơn nữa số xung nữa vòng là vô lý nên lấy giá
tri tràn trừ cho chênh lệch
if (delta_B >= __HAL_TIM_GET_AUTORELOAD(&htim2) / 2)
{
    delta_B = __HAL_TIM_GET_AUTORELOAD(&htim2) - delta_B;
}
//Tính giá trị RPM hiện tại
RPM_Avg_WR = ((delta_B) / (Pulse_Per_Round * 4 * Time_Interrupt) * 60)*
GT_Convert_radian * 0.5; // Nhan voi TST
// Tính trung bình
WR_RPM = Average_5_times(RPM_Avg_WR, sum_B);
// ♦♦?c Encoder A
Encoder_A = __HAL_TIM_GET_COUNTER(&htim3);
// Lấy chênh lệch 2 khoảng Encoder
delta_A = abs(Encoder_A - Encoder_A_Pre);
// Nếu giá trị chênh lệch lớn hơn nữa số xung nữa vòng là vô lý nên lấy giá
tri tràn trừ cho chênh lệch
if (delta_A >= __HAL_TIM_GET_AUTORELOAD(&htim3) / 2)
{
    delta_A = __HAL_TIM_GET_AUTORELOAD(&htim3) - delta_A;
}
//Tính giá trị RPM hiện tại
RPM_Avg_WL = ((delta_A) / (Pulse_Per_Round * 4 * Time_Interrupt) * 60) *
GT_Convert_radian * 0.5; // Nhan voi TST
// Tính trung bình
WL_RPM = Average_5_times(RPM_Avg_WL, sum_A);

// PID controller
//Error_WL -> ek = (SP_WL - WL_RPM);
Error_WL -> ek = (SP_WL_Fake_2 - WL_RPM);
Error_WR -> ek = (SP_WR - WR_RPM);

```

```

// Kênh A
PID_WL -> Up = PID_WL -> Kp * Error_WL -> ek;
PID_WL -> Ui = PID_WL -> Ui_1 + PID_WL -> Ki * Error_WL -> ek_1 *
Time Interrupt + PID_WL -> Ui_Antiwindup;
PID_WL -> Ud = PID_WL -> Ud_1 * (Error_WL -> ek - Error_WL -> ek_1);
// Antiwindup
PID_WL -> Ui_Antiwindup = Anti_Windup(PID_WL -> Temp_PWM, HIGH_Limit_PWM,
LOW_Limit_PWM, PID_WL->Kb);
PID_WL -> Temp_PWM = PID_WL -> Up + PID_WL -> Ui + PID_WL -> Ud;
PID_WL -> Out_PWM = round(PID_WL -> Temp_PWM);

if(PID_WL -> Out_PWM >= 999)
{
    PID_WL -> Out_PWM = 999;
}
else if (PID_WL -> Out_PWM <=0 )
{
    PID_WL -> Out_PWM = 0;
}

// Kênh B
PID_WR -> Up = PID_WR -> Kp * Error_WR -> ek;
PID_WR -> Ui = PID_WR -> Ui_1 + PID_WR -> Ki * Error_WR -> ek_1 *
Time Interrupt + PID_WR -> Ui_Antiwindup;
PID_WR -> Ud = PID_WR -> Ud_1 * (Error_WR -> ek - Error_WR -> ek_1);
PID_WR -> Temp_PWM = PID_WR -> Up + PID_WR -> Ui + PID_WR -> Ud;
// Antiwindup
PID_WR -> Ui_Antiwindup = Anti_Windup(PID_WR -> Temp_PWM,
HIGH_Limit_PWM, LOW_Limit_PWM, PID_WR->Kb);
PID_WR -> Out_PWM = round(PID_WR -> Temp_PWM);
if(PID_WR -> Out_PWM >= 999)
{
    PID_WR -> Out_PWM = 999;
}
else if (PID_WR -> Out_PWM <=0 )
{
    PID_WR -> Out_PWM = 0;
}

// Gán lại giá trị
Error_WL -> ek_1 = Error_WL -> ek;
Error_WR -> ek_1 = Error_WR -> ek;
PID_WL -> Ui_1 = PID_WL -> Ui;
PID_WR -> Ui_1 = PID_WR -> Ui;
PID_WL -> Ud_1 = PID_WL -> Ud;
PID_WR -> Ud_1 = PID_WR -> Ud;
//
Encoder_A_Pre = Encoder_A;
Encoder_B_Pre = Encoder_B;
//
// Xuất xung PWM
__HAL_TIM_SET_COMPARE(&htim9, TIM_CHANNEL_1,PID_WR -> Out_PWM);
__HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1,PID_WL -> Out_PWM);

```

```

// Tính động hòn c thuận và cập nhật vị trí
// Ông hòn c thuận
Forward_Kinematic(RPM_WR_Math, RPM_WL_Math, Kinematic_Matlab_A);

// Cập nhật giá trị theo tích phân
Kinematic_Matlab_A->x_Real = Kinematic_Matlab_A->x_Real +
Kinematic_Matlab_A->x_dot_Real * Time_Interrupt;
Kinematic_Matlab_A->y_Real = Kinematic_Matlab_A->y_Real +
Kinematic_Matlab_A->y_dot_Real * Time_Interrupt;

// Điểm han
Kinematic_Matlab_A->x_welding_Real = Kinematic_Matlab_A->x_Real +
0.111*cos(Kinematic_Matlab_A->theta) - 0.55*sin(Kinematic_Matlab_A->theta);
Kinematic_Matlab_A->y_welding_Real = Kinematic_Matlab_A->y_Real +
0.55*cos(Kinematic_Matlab_A->theta) + 0.111*sin(Kinematic_Matlab_A->theta);

// Quy ve he MM
Kinematic_Matlab_A->x_mm_Real = Kinematic_Matlab_A->x_Real * 1000;
Kinematic_Matlab_A->y_mm_Real = Kinematic_Matlab_A->y_Real * 1000;

// Cap nhat thoi gian dem
Time_Total = Time_Total + Time_Interrupt;

}

else if(htim -> Instance == TIM5 )
{
    // Counter dem nhu dong for de tinh gia tri trung binh cua ADC sau 5
    lan doc ADC
    SS->Counter_Timer = SS->Counter_Timer + 1;

    // Neu nho hon 5 lan doc thi cu doc va tinh trung binh ra, Bo di
    nhung nhieu cao bien do cao (> Nguong_tren)
    if(SS->Counter_Timer < 5)
    {
        // Doc ADC
        HAL_ADC_Start_DMA(&hadc1,(uint32_t*)SS->ADC_Value, 2);

        /*
         *      Neu gia tri ADC doc ve co gia tri nho hon nguong tren
         * thi cho phep cong trung binh. Neu khong thi bo qua
         */
        // Sensor_tren  SENSOR 1
        if(abs((int16_t)(SS->ADC_Value[0] - SS->ADC_AVG[0])) <= SS-
>Nguong_tren[0]) // Bo di nhung so nhieu lon
        {
            SS->ADC_AVG[0] = (SS->ADC_AVG[0]+SS->ADC_Value[0])/2;
        }

        // Sensor_duoii  SENSOR 2
        if(abs(SS->ADC_Value[1] - (int16_t)SS->ADC_AVG[1]) <= SS-
>Nguong_tren[1]) // Bo di nhung so nhieu lon
        {
            SS->ADC_AVG[1] = (SS->ADC_AVG[1]+SS->ADC_Value[1])/2;
        }
    }
    else if(SS->Counter_Timer == 5 )
    {
}

```

```

/*
 * Neu gia tri trung binh tinh ra dao dong trong khoang nguong
duoi thi gia tri sau khi filter ra khong doi.
*/
// SENSOR 1
if( abs((int16_t)(SS->ADC_AVG[0] - SS->ADC_Filter_Pre[0]))>=
SS->Nguong_duo[0])
{
    SS->ADC_Filter[0] = SS->ADC_AVG[0];
}

/*
 * De tranh truong hop no khong bat kip dao dong nguong tren
thi gia tri ADC sau khi filetr ra khong doi,
 * nen phai tang nguong tren len 3000 de ADC filter co the
thay doi
*/
if(SS->ADC_AVG[0] == SS->ADC_AVG_Pre[0])
{
    SS->Counter_nguong[0] = SS->Counter_nguong[0] + 1; // Moi lan trung la +1
    if(SS->Counter_nguong[0] == 3) // 0 day chon la neu 3
lan ma gia tri truoc van bang gia tri sau khi tinh trung binh thi tang nuong tren
len 3000
    {
        SS->Nguong_tren[0] = 3000;
        SS->Counter_nguong[0] = 0; // Reset counter
dem khi AVG = AVG_Pre
    }
}
else if(SS->ADC_AVG[0] != SS->ADC_AVG_Pre[0]) // Khi khong
bi trung thi gia tri nguong tren bang 80
{
    SS->Nguong_tren[0] = Nguong_tren_set;
    SS->Counter_nguong[0] = 0;
}

// SENSOR 2
if( abs((int16_t)(SS->ADC_AVG[1] - SS->ADC_Filter_Pre[1])) >=
SS->Nguong_duo[1])
{
    SS->ADC_Filter[1] = SS->ADC_AVG[1];
}

if(SS->ADC_AVG[1] == SS->ADC_AVG_Pre[1])
{
    SS->Counter_nguong[1] = SS->Counter_nguong[1] + 1;
    if(SS->Counter_nguong[1] == 3)
    {
        SS->Nguong_tren[1] = 3000;
        SS->Counter_nguong[1] = 0;
    }
}
else if(SS->ADC_AVG[1] != SS->ADC_AVG_Pre[1])
{
    SS->Nguong_tren[1] = Nguong_tren_set;
    SS->Counter_nguong[1] = 0;
}

```

```

        // Khoang cach that su
        // Noi suy ADC ra khoang cach
        SS->Khoang_cach_A1 = -0.00000065252408*pow(SS-
>ADC_Filter[0],3) + 0.000491491543157*pow(SS->ADC_Filter[0],2) -
1.375926465289920*SS->ADC_Filter[0] + 1696.27306147618000;
        SS->Khoang_cach_A2 = -0.00000064497840*pow(SS-
>ADC_Filter[1],3) + 0.000493461373136*pow(SS->ADC_Filter[1],2) -
1.401743095774150*SS->ADC_Filter[1] + 1735.70082713128000;

        //SS->Khoang_cach_A2 = -0.000000116009648*pow(SS-
>ADC_Filter[1],3) + 0.000787103643233*pow(SS->ADC_Filter[1],2) -
1.922570875398180*SS->ADC_Filter[1] + 1955.36496703496000;

        // Gan lai gia tri
        SS->ADC_AVG_Pre[1] = SS->ADC_AVG[1]; // Gia tri trung
binh

        SS->ADC_AVG_Pre[0] = SS->ADC_AVG[0];
        SS->ADC_Filter_Pre[0] = SS->ADC_Filter[0]; // Gia tri
Filter

        SS->ADC_Filter_Pre[1] = SS->ADC_Filter[1];
        // Reset Counter
        SS->Counter_Timer = 0;

        // Cap nhat vi tri diem den theo cam bien 30ms x 5 lan = 150ms
        Vi_tri_Cam_bien(SS, Kinematic_Matlab_A);

    }

}

/* USER CODE END 4 */

/***
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/***
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,

```

```
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```