

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CƠ KHÍ ĐỘNG LỰC

ĐỒ ÁN MÔN HỌC

INVERTED PENDULUM

GVHD: ThS. NGUYỄN TRUNG HIẾU

SVTH: MAI ANH TUẤN

MSSV: 21145309

SVTH: MAI GIA BẢO

MSSV: 21145077

SVTH: TRẦN HƯNG THỊNH

MSSV: 21145652

SVTH: NGUYỄN TRUNG HIẾU

MSSV: 21145129

Tp. Hồ Chí Minh, tháng 5 năm 2024

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CƠ KHÍ ĐỘNG LỰC

ĐỒ ÁN MÔN HỌC

Ngành: Công nghệ kỹ thuật ô tô

INVERTED PENDULUM

GVHD: ThS. NGUYỄN TRUNG HIẾU

SVTH: MAI ANH TUẤN

MSSV: 21145309

SVTH: MAI GIA BẢO

MSSV: 21145077

SVTH: TRẦN HƯNG THỊNH

MSSV: 21145652

SVTH: NGUYỄN TRUNG HIẾU

MSSV: 21145129

Tp. Hồ Chí Minh, tháng 5 năm 2024

TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT
TP. HỒ CHÍ MINH

KHOA CƠ KHÍ ĐỘNG LỰC

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập – Tự Do – Hạnh Phúc

TP. Hồ Chí Minh, ngày tháng năm

NHIỆM VỤ ĐÒ ÁN MÔN HỌC

Họ tên sinh viên: 1..... MSSV:
(E-mail: - Điện thoại:)
2..... MSSV:
(E-mail: - Điện thoại:)
3..... MSSV:
(E-mail: - Điện thoại:)

Ngành:

Khóa: Lớp:

1. Tên đề tài

.....
.....

2. Nhiệm vụ đề tài

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

3. Sản phẩm của đề tài

.....
.....

4. Ngày giao nhiệm vụ đề tài:

5. Ngày hoàn thành nhiệm vụ:

TRƯỞNG BỘ MÔN

CÁN BỘ HƯỚNG DẪN

KHOA CƠ KHÍ ĐỘNG LỰC

Bộ môn

PHIẾU NHẬN XÉT ĐỒ ÁN MÔN HỌC

(Dành cho giảng viên hướng dẫn)

Họ và tên sinh viên: MSSV:

Họ và tên sinh viên: MSSV:

Họ và tên sinh viên: MSSV:

Tên đề tài:

Ngành đào tạo:

Họ và tên GV hướng dẫn:

Ý KIẾN NHẬN XÉT

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

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

2. Nhận xét về kết quả thực hiện của ĐATN

2.1. Kết cấu, cách thức trình bày ĐATN:

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

2.2. Nội dung đồ án:

(Cơ sở lý luận, tính thực tiễn và khả năng ứng dụng của đồ án, các hướng nghiên cứu có thể tiếp tục phát triển)

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

2.3. Kết quả đạt được:

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

2.4. Những tồn tại (nếu có)

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

3. Đánh giá:

TT	Mục đánh giá	Điểm tối đa	Điểm đạt được
1.	Hình thức và kết cấu DATN	30	
	Đúng format với đầy đủ cả hình thức và nội dung các mục	10	
	Mục tiêu, nhiệm vụ, tổng quan của đề tài	10	
	Tính cấp thiết của đề tài	10	
2.	Nội dung DATN	50	
	Khả năng ứng dụng kiến thức toán học, khoa học và kỹ thuật, khoa học xã hội...	5	
	Khả năng thực hiện/phân tích/tổng hợp/đánh giá	10	
	Khả năng thiết kế chế tạo một hệ thống, thành phần hoặc quy trình đáp ứng yêu cầu đưa ra với những ràng buộc thực tế	15	
	Khả năng cải tiến và phát triển	15	
	Khả năng sử dụng công cụ kỹ thuật, phần mềm chuyên ngành	5	
3.	Đánh giá về khả năng ứng dụng của đề tài	10	
4.	Sản phẩm cụ thể của DATN	10	
	Tổng điểm	100	

4. Kết luận:

- Được phép bảo vệ
- Không được phép bảo vệ

TP.HCM, ngày tháng năm 2024

Giảng viên hướng dẫn

(Ký, ghi rõ họ tên)

TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT
TP. HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập – Tự Do – Hạnh Phúc

KHOA CƠ KHÍ ĐỘNG LỰC

Bộ môn

PHIẾU NHẬN XÉT ĐỒ ÁN MÔN HỌC

(*Dành cho giảng viên phản biện*)

Họ và tên sinh viên: MSSV:

Họ và tên sinh viên: MSSV:

Họ và tên sinh viên: MSSV:

Tên đề tài:

Ngành đào tạo:

Họ và tên GV hướng dẫn:

Ý KIẾN NHẬN XÉT

1. Kết cấu, cách thức trình bày ĐATN:

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

2. Nội dung đồ án:

(*Cơ sở lý luận, tính thực tiễn và khả năng ứng dụng của đồ án, các hướng nghiên cứu có thể tiếp tục phát triển*)

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

3. Kết quả đạt được:

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

4. Những thiếu sót và tồn tại của ĐATN:

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

5. Câu hỏi:

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

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

6. Đánh giá:

TT	Mục đánh giá	Điểm tối đa	Điểm đạt được
1.	Hình thức và kết cấu ĐATN	30	
	Đúng format với đầy đủ cả hình thức và nội dung các mục	10	
	Mục tiêu, nhiệm vụ, tổng quan của đề tài	10	
	Tính cấp thiết của đề tài	10	
2.	Nội dung ĐATN	50	
	Khả năng ứng dụng kiến thức toán học, khoa học và kỹ thuật, khoa học xã hội...	5	
	Khả năng thực hiện/phân tích/tổng hợp/đánh giá	10	
	Khả năng thiết kế chế tạo một hệ thống, thành phần hoặc quy trình đáp ứng yêu cầu đưa ra với những ràng buộc thực tế	15	
	Khả năng cải tiến và phát triển	15	
3.	Đánh giá về khả năng ứng dụng của đề tài	10	
	Sản phẩm cụ thể của ĐATN	10	
	Tổng điểm	100	

7. Kết luận:

- Được phép bảo vệ
- Không được phép bảo vệ

TP.HCM, ngày tháng năm 2024

Giảng viên phản biện

(Ký, ghi rõ họ tên)

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH
KHOA CƠ KHÍ ĐỘNG LỰC

XÁC NHẬN HOÀN THÀNH ĐỒ ÁN

Tên đề tài:

.....

Họ và tên Sinh viên: MSSV:

..... MSSV:

..... MSSV:

Ngành: Công nghệ kỹ thuật ô tô

Sau khi tiếp thu và điều chỉnh theo góp ý của Giảng viên hướng dẫn, Giảng viên phản biện và các thành viên trong Hội đồng bảo vệ. Đồ án tốt nghiệp đã được hoàn chỉnh đúng theo yêu cầu về nội dung và hình thức.

Chủ tịch Hội đồng: _____

Giảng viên hướng dẫn: _____

Giảng viên phản biện: _____

Tp. Hồ Chí Minh, ngày tháng năm 2024

vì

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc đối với thầy Nguyễn Trung Hiếu đã nhiệt tình hướng dẫn và hỗ trợ chúng em trong quá trình thực hiện đồ án cuối kỳ về đề tài “Inverted Pendulum”. Đây là một đề tài rất thú vị và bổ ích cho chúng em, giúp chúng em nâng cao kỹ năng lập trình và thiết kế hệ thống.

Trong quá trình làm đồ án, thầy đã cung cấp cho chúng em nhiều tài liệu tham khảo quý báu, góp ý cho bài thuyết trình của chúng em, giải đáp những câu hỏi phức tạp của chúng em một cách rõ ràng và chi tiết.

Trong quá trình học tập, cũng như là trong quá trình làm bài báo cáo, khó tránh khỏi sai sót, rất mong thầy bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn của nhóm còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp từ thầy để nhóm chúng em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt hơn bài báo cáo sắp tới.

Chúng em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Hệ thống con lắc ngược, đặc trưng bởi khói tâm nằm phía trên điểm trục của nó, là hình ảnh thu nhỏ của một thách thức điều khiển cỗ điển do tính không ổn định vốn có và động lực học phi tuyến tính của nó. Dự án này đi sâu vào ứng dụng bộ điều khiển tỉ lệ, tích phân và vi phân (PID) để duy trì sự cân bằng của con lắc ngược gắn trên xe đẩy. Bộ điều khiển PID, nổi tiếng về tính linh hoạt trong các hệ thống điều khiển khác nhau, được điều chỉnh chính xác để quản lý vị trí của con lắc bằng cách điều khiển chuyên động ngang của xe đẩy, từ đó tạo ra mô-men xoắn đối trọng cần thiết.

Nghiên cứu này bắt đầu bằng việc phân tích toàn diện về hành vi của con lắc ngược, nêu bật tính chất không thể đoán trước và độ nhạy cảm của nó trước nhiều biến số. Sau đó, dự án khám phá việc thiết kế và triển khai bộ điều khiển PID, bộ điều khiển này điều chỉnh vị trí của xe đẩy trong thời gian thực để ổn định con lắc. Thông qua thử nghiệm và tối ưu hóa nghiêm ngặt, các tham số của bộ điều khiển PID được tinh chỉnh để đạt được trạng thái cân bằng tinh tế, thể hiện tính hiệu quả của bộ điều khiển trong một hệ thống tiêu biểu cho các vấn đề điều khiển phức tạp trong thế giới thực.

Kết quả của dự án này không chỉ nhấn mạnh tính hiệu quả của bộ điều khiển PID trong việc quản lý động lực học phức tạp của con lắc ngược mà còn đóng góp những hiểu biết có giá trị vào lĩnh vực kỹ thuật hệ thống điều khiển rộng hơn. Những phát hiện này có ý nghĩa đối với việc phát triển các chiến lược điều khiển tiên tiến, mở đường cho những đổi mới trong tương lai về tự động hóa và robot.

MỤC LỤC

LỜI CẢM ƠN	vii
LỜI NÓI ĐẦU	viii
MỤC LỤC	ix
DANH MỤC HÌNH ẢNH	xiii
DANH MỤC BẢNG BIỂU	xvi
CHƯƠNG 1: GIỚI THIỆU.....	1
1.1 Lý do chọn đề tài.....	1
1.2 Mục tiêu đề tài.....	2
1.3 Nhiệm vụ đề tài	3
1.4 Đối tượng và phạm vi nghiên cứu.....	3
1.5 Tổng quan lý thuyết	3
1.5.1 Nghiên cứu trong nước	4
1.5.2 Nghiên cứu ngoài nước	4
1.6 Kết quả mong muốn.....	6
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	7
2.1 Giới thiệu con lắc ngược	7
2.2 Mô hình hóa toán học con lắc ngược trên xe	9
2.3 Mô hình hóa con lắc ngược trên xe bằng hàm truyền.....	13
2.4 Một số bộ điều khiển thông dụng cho con lắc ngược	15
2.4.1 Bộ điều khiển LQR	15
2.4.2 Bộ điều khiển Fuzzy Logic	15
2.4.3 Điều khiển Mạng nơ-ron.....	16
2.4.4 Điều khiển trượt (Sliding mode control)	17

2.4.5 Điều khiển PID	17
2.5 Phương pháp điều khiển cân bằng PID cho hệ con lắc ngược trên xe	20
2.5.1 Bộ điều khiển PID 2 bậc tự do.....	20
2.5.2 Bộ điều khiển Cascade PID	21
2.6 Các phương pháp điều chỉnh thông số PID	22
2.6.1 Điều chỉnh thủ công	22
2.6.2 Phương pháp Ziegler-Nichols.....	22
2.6.3 Phương pháp chuyển tiếp.....	22
2.6.4 Phương pháp cohen-coon	23
2.6.5 Phương pháp điều chỉnh bằng phần mềm.....	23
2.6.6 Dùng giải thuật di truyền (GA) để xác định thông số PID	23
CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG.....	25
3.1 Các thông số cơ bản	25
3.2 Xây dựng chương trình mô phỏng bộ điều khiển dựa trên mô hình toán học trên Simulink	25
3.2.1 Phương pháp điều khiển Cascade	25
3.2.2 Phương pháp điều khiển Cascade kết hợp swing up	31
3.3 Xây dựng chương trình mô phỏng bộ điều khiển dựa trên hàm truyền	37
CHƯƠNG 4: THIẾT KẾ MÔ HÌNH CON LẮC NGƯỢC.....	40
4.1 Sơ đồ khói của một hệ con lắc ngược	40
4.2 Yêu cầu thiết kế con lắc ngược	41
4.3 Thiết kế cơ khí	41
4.4 Thiết kế phần điện.....	46
4.4.1 Vi điều khiển.....	46
4.4.2 Driver cầu H.....	48

4.4.3 Mạch giảm áp.....	50
4.4.4 Module Relay.....	51
4.4.5 Encoder	52
4.4.6 Công tắc	53
4.4.7 Nguồn điện.....	54
4.5 Sơ đồ mạch điện.....	55
4.5.1 Thiết kế mạch mô phỏng trên proteus	55
4.5.2 Mạch thiết kế thực tế.....	56
CHƯƠNG 5: XÂY DỰNG BỘ ĐIỀU KHIỂN TRÊN MATLAB/SIMULINK	58
5.1 Xây dựng bộ điều khiển cân bằng trên Simulink.....	58
5.1.1 Giao tiếp giữa vi điều khiển và Matlab/Simulink	58
5.1.2 Xây dựng bộ điều khiển cân bằng con lắc bằng thuật toán PID.....	58
5.1.3 Xây dựng bộ điều khiển cân bằng hệ Cascade PID	60
5.2 Xây dựng bộ điều khiển Swing-up cho con lắc ngược	62
5.4 Bộ điều khiển Swing-up và Cascade cân bằng con lắc ngược	65
5.5 Kết quả	68
CHƯƠNG 6: XÂY DỰNG BỘ ĐIỀU KHIỂN BẰNG NGÔN NGỮ LẬP TRÌNH C.....	76
6.1 Khai báo các giá trị ban đầu của phần cứng	76
6.2 Xây dựng cài đặt các thông số đầu vào cho bộ điều khiển PID	77
6.3 Xây dựng các hàm điều khiển.....	78
6.4 Chương trình điều khiển chính	82
6.5 Sử dụng phần mềm SerialPlot để vẽ đồ thị.....	84
6.6 Hiển thị đồ thị hệ thống qua phần mềm LabView	85
6.7 Kết luận	88
CHƯƠNG 7: KẾT LUẬN	90

7.1	Những kết quả đạt được	90
7.2	Hạn chế của đề tài	90
7.3	Hướng phát triển của đề tài.....	90
BẢNG PHÂN CÔNG VÀ TIẾN ĐỘ CÔNG VIỆC		95

DANH MỤC HÌNH ẢNH

Hình 2.1 Mô hình động lực học con lắc ngược quay.....	7
Hình 2.2 Mô hình động lực học con lắc ngược dạng bánh xe	8
Hình 2.3 Mô hình động lực học con lắc ngược trên xe	8
Hình 2.4 Mô hình toán học con lắc ngược trên xe.....	9
Hình 2.5 Sơ đồ bộ điều khiển LQR	15
Hình 2.6 Nguyên lý hoạt động của bộ điều khiển logic mờ	16
Hình 2.7 Mạng nơ ron nhân tạo	16
Hình 2.8 Bộ điều khiển trượt	17
Hình 2.9 Bộ điều khiển PID.....	18
Hình 2.10 Sơ đồ bộ điều khiển PID cho hệ SIMO	20
Hình 2.11 Sơ đồ bộ điều khiển PID cho hệ SIMO [14].....	22
Hình 2.12 Lưu đồ giải thuật di truyền GA.....	24
Hình 3.1 Sơ đồ khối mô hình điều khiển Cascade trên simulink	25
Hình 3.2 Mô phỏng con lắc ngược trên simulink bằng Cascade	26
Hình 3.3 Mô hình toán hệ con lắc ngược phi tuyến.....	27
Hình 3.4 Khối chuyển đổi từ điện áp sang lực tác động của động cơ	27
Hình 3.5 Khối xác định vị trí mong muốn cho con lắc.....	28
Hình 3.6 Đồ thị vị trí xe phương pháp điều khiển Cascade.....	28
Hình 3.7 Đồ thị vận tốc xe phương pháp điều khiển Cascade.....	29
Hình 3.8 Đồ thị vị trí con lắc phương pháp điều khiển Cascade	29
Hình 3.9 Đồ thị vận tốc góc con lắc phương pháp điều khiển Cascade	30
Hình 3.10 Mô phỏng con lắc ngược trên simulink bằng Cascade sử dụng swing up.....	31
Hình 3.11 Mô phỏng con lắc ngược trên simulink bằng Cascade	31
Hình 3.12 Sơ đồ khối thuật toán điều khiển swing up sử dụng bộ điều khiển PD	32
Hình 3.13 Mô phỏng bộ điều khiển swing up trên Simulink.....	32
Hình 3.14 Khối mô phỏng ngoại lực tác dụng vào con lắc	33
Hình 3.15 Đồ thị vị trí xe phương pháp điều khiển Cascade sử dụng swing-up	34
Hình 3.16 Đồ thị vận tốc xe phương pháp điều khiển Cascade sử dụng swing-up.....	34
Hình 3.17 Đồ thị vị trí con lắc phương pháp điều khiển Cascade sử dụng swing-up	35

Hình 3.18 Đồ thị vận tốc con lắc phương pháp điều khiển Cascade sử dụng swing-up ...	35
Hình 3.19 Sơ đồ mô phỏng hàm truyền con lắc	37
Hình 3.20 Đồ thị vị trí con lắc (đơn vị: rad)	37
Hình 3.21 Đồ thị vị trí xe (đơn vị: m).....	38
Hình 4.1 Sơ đồ khối con lắc ngược.....	40
Hình 4.2 Thông số kỹ thuật động cơ Minertia RO2SA	44
Hình 4.3 Thiết kế mô hình trên Solidworks.....	45
Hình 4.4 Thông số kích thước của mô hình (đơn vị: mm)	45
Hình 4.5 Mô hình thực tế.....	46
Hình 4.6 Arduino Mega 2560 [32]	47
Hình 4.7 Sơ đồ chân kết nối trên Arduino Mega 2560 [15]	47
Hình 4.8 Sơ đồ nguyên lý mạch cầu H [16]	49
Hình 4.9 Sơ đồ chân của mạch cầu H BTS7960 [34]	49
Hình 4.10 Mạch giảm áp LM2596 [17]	50
Hình 4.11 Module giảm áp chịu dòng cao [18]	51
Hình 4.12 Module relay 5V [19]	52
Hình 4.13 Encoder omron [20]	53
Hình 4.14 Công tắc hành trình Omron [21].....	53
Hình 4.15 Công tắc dừng khẩn cấp [22]	54
Hình 4.16 Nguồn xung tần số [23].....	54
Hình 4.17 Thiết kế mô hình mạch điện trên Proteus	55
Hình 4.18 Thi công mạch điện thực tế.....	56
Hình 4.19 Bản thiết kế hộp điều khiển (đơn vị: mm)	57
Hình 4.20 Hộp điều khiển mô hình sau khi hoàn thiện	57
Hình 5.1 Giao tiếp giữa Simulink và vi điều khiển [26].....	58
Hình 5.2 Chương trình điều khiển 1 hệ PID cho con lắc trên Simulink.....	59
Hình 5.3 Bộ điều khiển PID cho hệ con lắc.....	60
Hình 5.4 Lưu đồ giải thuật cho bộ điều khiển Cascade PID	60
Hình 5.5 Chương trình điều khiển cân bằng con lắc ngược bằng cascade control.....	61
Hình 5.6 Chia vùng điều khiển swing up.....	62

Hình 5.7 Lưu đồ điều khiển swing up bằng phương pháp kiểm soát vận tốc	63
Hình 5.8 Bộ điều khiển Swing up bằng phương pháp vận tốc	63
Hình 5.9 Lưu đồ thuật toán điều khiển Swing-up sử dụng thuật toán PD [13]	64
Hình 5.10 Chương trình điều khiển Swing-up sử dụng PD.....	64
Hình 5.11 Lưu đồ thuật toán điều khiển kết hợp swing-up và cân bằng	66
Hình 5.12 Chương trình điều khiển trên simulink.....	66
Hình 5.13 Khối bộ điều khiển cascade	67
Hình 5.14 Khối bộ điều khiển swing up	67
Hình 5.15 Giảm độ phân giải encoder bằng S-Function Builder	68
Hình 5.16 Đồ thị vị trí xe (đơn vị cm)	68
Hình 5.17 Đồ thị vận tốc xe (đơn vị: cm/s)	69
Hình 5.18 Đồ thị vị trí con lắc (đơn vị: rad)	70
Hình 5.19 Đồ thị vận tốc con lắc (đơn vị: rad/s)	70
Hình 5.20 Đồ thị tín hiệu PWM (0-255).....	71
Hình 5.21 Đồ thị vị trí xe khi có ngoại lực tác dụng (đơn vị: cm)	72
Hình 5.22 Đồ thị vị trí con lắc khi có ngoại lực tác dụng (đơn vị: rad)	72
Hình 5.23 Đồ thị vận tốc xe khi có ngoại lực tác dụng (đơn vị cm/s).....	73
Hình 5.24 Đồ thị vận tốc con lắc khi có ngoại lực tác dụng (đơn vị: rad/s).....	73
Hình 5.25 Đồ thị tín hiệu PWM khi có ngoại lực tác dụng	74
Hình 6.1 Hình ứng dụng Serial Plot	84
Hình 6.2 Hình giao diện cài đặt cho Serial Plot.....	85
Hình 6.3 Hình đồ thị được vẽ ra bằng Serial Plot.....	85
Hình 6.4 Giao diện hiển thị trên phần mềm Labview.....	86
Hình 6.5 Khối xử lý dữ liệu gửi đến từ Arduino	87
Hình 6.6 Kết quả hiển thị dữ liệu lên Labview trong quá trình điều khiển	88

DANH MỤC BẢNG BIỂU

Bảng 4.1 Các thông số cơ bản của mô hình.....	42
Bảng 4.2 Ảnh hưởng của các thông số vật lý đến con lắc	42
Bảng 4.3 Thông số kỹ thuật cơ bản của Arduino Mega 2560	48
Bảng 4.4 Bảng thông số kỹ thuật cơ bản của BTS7960	50
Bảng 4.5 Thông số cơ bản của module giảm áp 10A	51
Bảng 4.6 Thông số kỹ thuật cơ bản của encoder 600 xung và 1000 xung	52
Bảng 4.7 Thông số kỹ thuật của bộ nguồn	55

CHƯƠNG 1: GIỚI THIỆU

1.1 Lý do chọn đề tài

Hệ thống con lắc ngược là một ví dụ cổ điển về hệ thống không ổn định động học và có tính phi tuyến cao, đòi hỏi sự điều khiển chính xác để duy trì trạng thái cân bằng. Tính không ổn định vốn có của nó khiến nó trở thành một chủ đề lý tưởng để khám phá và chứng minh tính hiệu quả của các chiến lược kiểm soát khác nhau, đặc biệt là trong môi trường giáo dục và nghiên cứu. Thách thức của việc cân bằng một con lắc ngược phản ánh các vấn đề điều khiển trong thế giới thực được tìm thấy trong nhiều ứng dụng, từ robot đến điều khiển tư thế con người.

Bộ điều khiển Tỷ lệ-Tích phân-Vi phân (PID), với cấu trúc đơn giản và nền tảng lý thuyết vững chắc, trình bày một giải pháp mạnh mẽ nhưng dễ tiếp cận cho bài toán con lắc ngược. Việc sử dụng rộng rãi nó trong các ngành công nghiệp và khả năng cung cấp sự hiểu biết rõ ràng về các nguyên tắc hệ thống điều khiển khiến nó trở thành sự lựa chọn tuyệt vời cho dự án này. Bằng cách triển khai bộ điều khiển PID, chúng em mong muốn không chỉ đạt được sự cân bằng ổn định của con lắc ngược mà còn nâng cao hiểu biết của chúng em về thiết kế và điều chỉnh hệ thống điều khiển.

Hơn nữa, dự án đóng vai trò là nền tảng để thu hẹp khoảng cách giữa kiến thức lý thuyết và ứng dụng thực tế. Nó cho phép trải nghiệm thực tế về mô hình hóa hệ thống, thiết kế bộ điều khiển và phân tích phản hồi của hệ thống theo thời gian thực. Những hiểu biết sâu sắc thu được từ dự án này là vô giá đối với sinh viên cũng như các chuyên gia, thúc đẩy sự đánh giá sâu sắc hơn về sự phức tạp và sắc thái của kỹ thuật điều khiển.

Tóm lại, lý do căn bản đằng sau việc sử dụng bộ điều khiển PID để cân bằng con lắc ngược trên xe đẩy bắt nguồn từ sự liên quan của hệ thống như một thách thức điều khiển, giá trị giáo dục của bộ điều khiển PID và cơ hội áp dụng các khái niệm lý thuyết một cách hữu hình và hấp dẫn. .

1.2 Mục tiêu đề tài

Mục tiêu bao quát của dự án này là thiết lập sự hiểu biết toàn diện và ứng dụng thực tế về hệ thống điều khiển con lắc ngược. Các mục tiêu cụ thể như sau:

- **Nâng vững lý thuyết:** Để phát triển nền tảng vững chắc về lý thuyết động lực học của con lắc ngược, hiểu các nguyên tắc toán học và vật lý chi phối hành vi của chúng.
- **Lý thuyết bộ điều khiển PID:** Để hiểu rõ hơn về lý thuyết bộ điều khiển PID, bao gồm cơ sở toán học, phương pháp điều chỉnh và chiến lược thực hiện.
- **Hoàn thiện chương trình mô phỏng:** Để xây dựng mô hình mô phỏng con lắc ngược và đạt được sự điều khiển của nó bằng bộ điều khiển PID trong môi trường MATLAB và Simulink, từ đó thể hiện khả năng chuyển các khái niệm lý thuyết thành hiện thực mô phỏng.
- **Thiết kế mô hình con lắc ngược trong thực tế:** Để thiết kế được mô hình thực tế, nhóm cần nghiên cứu các thiết kế phù hợp có sẵn cho hệ con lắc ngược, tiêu chí cần đặt ra: mô hình có khối lượng nhẹ, dễ mang vác, tháo lắp, bộ điều khiển chịu được dòng điện cao để kéo động cơ.
- **Thiết kế thuật toán swing up và cân bằng:** thiết kế thuật toán sao cho, con lắc tự di chuyển đến vị trí cân bằng (swing up) và có khả năng tự cân bằng tại một điểm. Hệ thống có khả năng dập nhiễu tốt.
- **Triển khai Arduino:** Thiết kế và triển khai bộ điều khiển PID cho con lắc ngược sử dụng nền tảng Arduino, cụ thể là triển khai thuật toán điều khiển trên bo mạch Arduino Mega 2560, thể hiện kỹ năng thiết kế hệ thống nhúng và lập trình vi điều khiển.
- **Triển khai Simulink đến phần cứng:** Để thiết kế bộ điều khiển PID cho con lắc ngược trong Simulink và nhúng trực tiếp thuật toán điều khiển vào bo mạch Arduino Mega 2560 từ phần mềm Simulink, minh họa khả năng kết nối mô phỏng và ứng dụng trong thế giới thực một cách liền mạch.
- **Hiển thị các thông số hệ thống trên máy tính bằng LabVIEW:** sử dụng phần mềm chuyên về xử lý dữ liệu để hiển thị thông số hệ thống sao cho trực quan và dễ so sánh, có thể ON/OFF hệ thống từ máy tính.

Những mục tiêu này nhằm mục đích thu hẹp khoảng cách giữa kiến thức lý thuyết và thực hành thực tế, cung cấp trải nghiệm thực tế giúp củng cố sự hiểu biết về hệ thống điều khiển một cách hữu hình và có tác động.

1.3 Nhiệm vụ đề tài

Hiểu biết sâu sắc: Các thành viên trong nhóm sẽ hiểu sâu hơn về khái niệm về điều khiển cân bằng qua cách thực hiện và hoàn thành dự án con lắc ngược.

Ứng dụng thực tiễn: Những kiến thức về vi điều khiển và hệ thống điện-điện tử đã học có thể được áp dụng hiệu quả để lập trình Arduino và lắp ráp các mạch điều khiển cho con lắc.

Nền tảng cho các dự án tương lai: Dự án này sẽ đóng vai trò là nền tảng cho các dự án trong tương lai, cho phép nhóm thực hiện chúng một cách có hệ thống và hiệu quả hơn. Hơn nữa, nó có thể truyền cảm hứng cho việc tạo ra các phương tiện có khả năng vận chuyển người hoặc hàng hóa trên nhiều địa hình khác nhau.

Mở rộng kiến thức: Thông qua dự án này, các thành viên trong nhóm sẽ có thêm kiến thức về hệ thống điều khiển tự động trên ô tô. Điều này sẽ đạt được bằng cách nghiên cứu các mạch điện và ứng dụng mà robot cân bằng có thể mang lại cho ô tô.

1.4 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu

- Mô hình con lắc ngược hai bậc tự do
- Bộ điều khiển PID trên Matlab-Simulink và ngôn ngữ lập trình C

Phạm vi nghiên cứu

- Xây dựng mô hình toán học con lắc ngược hai bậc tự do
- Thiết kế phần cứng mô hình con lắc ngược hệ pole and cart
- Điều khiển swing-up và cân bằng hệ thống bằng bộ điều khiển PID
- Mô phỏng hệ thống bằng phần mềm Matlab-Simulink và đánh giá kết quả
- Xây dựng chương trình điều khiển hệ thống con lắc ngược trên miền thời gian thực

1.5 Tổng quan lý thuyết

Đối với nghiên cứu lý thuyết, đối tượng nghiên cứu chính sẽ là động lực học của con lắc ngược và lý thuyết bộ điều khiển PID. Ngoài ra, phần mềm MATLAB và Simulink sẽ được sử dụng để mô phỏng con lắc ngược, giúp hiểu sâu hơn và trực quan hơn.

Đối với nghiên cứu thực nghiệm, đối tượng nghiên cứu sẽ là con lắc ngược trên mô hình xe đây được điều khiển bằng động cơ một chiều thông qua dây đai. Dự án sẽ sử dụng bộ mã hóa để thu thập tín hiệu vị trí của con lắc và xe đây, sau đó sẽ gửi đến bộ vi điều khiển Arduino Mega 2560. Bộ vi điều khiển này sẽ điều khiển động cơ DC thông qua bộ điều khiển động cơ cầu H LN298N.

1.5.1 Nghiên cứu trong nước

Đề tài “Sử dụng thuật toán mờ noron điều khiển cân bằng con lắc ngược” của tác giả Nguyễn Hữu Mỹ, đại học Đà Nẵng (2011). Trong đề tài này, tác giả đã sử dụng lý thuyết mờ kết hợp với mạng noron để thiết kế hệ thống điều khiển cân bằng cho hệ con lắc ngược. Ngoài ra, tác giả cũng đã so sánh kết quả giữa thuật toán PID và bộ điều khiển mờ noron điều khiển cân bằng hệ con lắc ngược. Trong đó, bộ điều khiển PID được thiết kế đơn giản hơn và khi điều khiển đồng thời vị trí xe và góc lệch ban đầu của con lắc thì bộ điều khiển PID cho kết quả điều khiển tốt hơn. Còn bộ điều khiển sử dụng thuật toán mờ noron cho kết quả tốt hơn khi điều khiển theo góc lệch ban đầu, theo vị trí xe hoặc theo lực tác động ngẫu nhiên vào xe [1].

Đề tài “Xây dựng bộ điều khiển nhúng tuyến tính hóa vào ra cho hệ xe con lắc ngược” của tác giả Nguyễn Văn Đông Hải, đại học Quốc gia TP.HCM (2011). Trong đề tài này, tác giả đã sử dụng thuật toán điều khiển LQR rời rạc để thiết kế hệ thống điều khiển cân bằng cho hệ con lắc ngược. Tác giả đã xây dựng chương trình mô phỏng hệ con lắc ngược trên Matlab/Simulink và xây dựng mô hình thực tế của hệ. Sau khi cho chạy thực nghiệm, hệ con lắc ngược đã được điều khiển thành công với bộ điều khiển LQR. Ngoài ra, tác giả còn xây dựng bộ điều khiển swing-up cho hệ con lắc ngược [2].

Đề tài “Nghiên cứu xây dựng hệ thống điều khiển con lắc ngược” của tác giả ThS.Nguyễn Tiến Dũng, đại học hàng hải Việt Nam (2016). Trong đề tài này, tác giả đã sử dụng thuật toán điều khiển mờ để thiết kế hệ thống điều khiển cân bằng cho hệ con lắc ngược. [3]

Và sau khi tác giả thực hiện các nội dung:

- Nghiên cứu và xây dựng mô hình toán học của hệ con lắc ngược.
- Xây dựng được bộ điều khiển mờ cho hệ con lắc ngược.
- Xây dựng được mô hình thực nghiệm điều khiển hệ con lắc ngược trên miền thời gian thực.

Hệ con lắc ngược đã được điều khiển thành công và bộ điều khiển tác giả đã nghiên cứu trong đề tài hoàn toàn đáp ứng được các yêu cầu về chất lượng điều khiển cho hệ con lắc ngược. Như vậy trong quá trình thực hiện đề tài này, tác giả đã giải quyết được vấn đề đã đặt ra.

Đề tài “Thiết kế mô hình cân bằng con lắc ngược” của tác giả ThS.Nguyễn Thanh Tân, đại học Trà Vinh (2017). Trong đề tài này, tác giả đã sử dụng các phương pháp điều khiển khác nhau để thiết kế hệ thống điều khiển cho hệ con lắc ngược như PID, LQR hay LQR kết hợp với giải thuật di truyền GA. Thông qua nghiên cứu và thực nghiệm, tác giả đã xây dựng thành công mô hình hệ con lắc ngược hoạt động ổn định, thẩm mỹ. Dựa ra được một số phương pháp điều khiển khác nhau (PID, LQR, GA kết hợp với LQR) áp dụng trên mô hình hệ con lắc ngược và cho ra kết quả mô phỏng rất tốt trên Matlab/Simulink. Ngoài ra, tác giả còn áp dụng giải thuật di truyền GA vào đề tài nhằm tối ưu các thông số của bộ điều khiển PID và LQR, giúp tiết kiệm thời gian và hạn chế quá trình thử sai nhiều lần. Việc tác giả sử dụng nhiều phương pháp điều khiển khác nhau trong đề tài nhằm mục đích kiểm chứng và tìm ra giải thuật tối ưu cho từng ứng dụng cụ thể của bài toán [4].

Đề tài “Điều khiển hệ con lắc ngược – xe sử dụng đại số gia tử” của các tác giả Bùi Hải Lê, Phạm Minh Nam, Bùi Thanh Lâm, tạp chí khoa học công nghệ (2019). Trong đề tài, tác giả đã sử dụng phương pháp điều khiển đại số gia tử để thiết kế hệ thống điều khiển cho hệ con lắc ngược. Ngoài ra, tác giả cũng đã so sánh kết quả giữa bộ điều khiển LQR và bộ điều khiển đại số gia tử trong việc điều khiển cân bằng hệ con lắc ngược. Trong đó, với bộ điều khiển đại số gia tử, góc lệch và vận tốc góc của con lắc cũng như vị trí và vận tốc của xe được điều khiển đồng thời. Và kết quả mô phỏng cho thấy bộ điều khiển đại số gia tử cho ra kết quả tối ưu hơn, điều khiển chính xác hơn và thời gian đáp ứng nhanh hơn so với bộ điều khiển LQR [5].

Đề tài “Điều khiển cân bằng hệ con lắc ngược” của tác giả Nguyễn Anh Vũ, đại học sư phạm kỹ thuật TP.HCM (2021). Trong đề tài này, tác giả đã thiết kế nhiều bộ điều khiển khác nhau để điều khiển cho hệ con lắc ngược như LQR, LQG, LMI, Fuzzy, FLQR và FLQG. Bên cạnh đó, tác giả cũng đã xây dựng và phân tích được phương trình động học cho hệ thống. Và tác giả đã điều khiển thành công mô hình hệ con lắc ngược cả trên hai phương diện là mô phỏng và thực nghiệm. Dựa vào các kết quả thực nghiệm và mô phỏng, các bộ điều khiển vừa nêu đã điều khiển ổn định thành công hệ xe con lắc ngược tại vị trí

lân cận của điểm làm việc tĩnh và chỉ ra được sự tối ưu hơn và kém hơn của từng giải thuật dựa trên kết quả mô phỏng [6].

1.5.2 Nghiên cứu ngoài nước

Nghiên cứu “Đứng dậy và ổn định con lắc ngược” của tác giả Andrew K. Stimac (1999) sử dụng thuật toán LQR. Tác giả Johnny Lam cũng sử dụng chủ đề “Điều khiển con lắc ngược”. Sử dụng thuật toán LQR (2008) với thời gian điều khiển cân bằng hệ thống lớn hơn 10s.

Nghiên cứu “Điều khiển con lắc ngược dựa trên thị giác bằng bộ lọc hạt xếp tầng” Đại học Công nghệ Graz, Áo (2008) do nhóm Stu less và Markus Brandner của Manuel sử dụng công nghệ xử lý ảnh để điều khiển con lắc ngược cân bằng.

1.6 Kết quả mong muôn

Sau khi hoàn thành dự án này, kết quả dự kiến như sau:

- Mô hình cơ khí thực và mạch điện: Mô hình vật lý của một con lắc ngược trên xe đẩy, hoàn chỉnh với tất cả các bộ phận cơ khí cần thiết. Điều này bao gồm con lắc, xe đẩy, động cơ DC và dây đai. Ngoài ra, một mạch điện để điều khiển con lắc ngược sẽ được thiết kế và triển khai. Mạch này sẽ bao gồm các thành phần như bộ vi điều khiển Arduino Mega 2560, bộ điều khiển động cơ cầu H L298N và bộ mã hóa để phản hồi vị trí.
- Mô hình Simulink của con lắc ngược: Một mô hình Simulink toàn diện thể hiện chính xác động lực học của hệ thống con lắc ngược. Mô hình này sẽ được sử dụng để mô phỏng hành vi của hệ thống và xác nhận chiến lược điều khiển trước khi triển khai trên hệ thống thực.
- Bộ điều khiển PID cho con lắc ngược: Thuật toán bộ điều khiển PID được viết cho Arduino Mega 2560 bằng Arduino IDE. Bộ điều khiển này sẽ sử dụng phản hồi vị trí từ bộ mã hóa để điều khiển chuyển động của xe đẩy và giữ cho con lắc cân bằng.
- Chương trình từ Simulink đến Flash trên Arduino Mega 2560: Một chương trình được tạo từ mô hình Simulink có thể flash trực tiếp lên Arduino Mega 2560. Điều này sẽ cho phép chiến lược điều khiển lý thuyết được phát triển trong Simulink có thể được áp dụng vào hệ thống thực.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

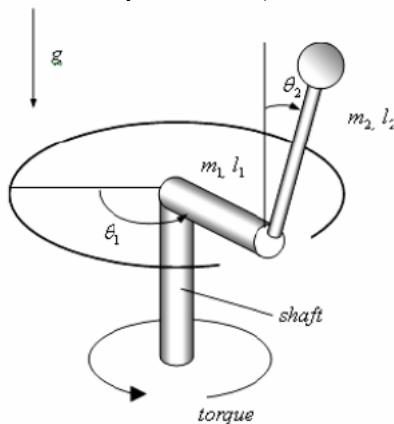
2.1 Giới thiệu con lắc ngược

Hệ thống con lắc ngược là một ví dụ cổ điển về hệ thống không ổn định động học và có tính phi tuyến cao, đòi hỏi sự điều khiển chính xác để duy trì trạng thái cân bằng. Tính không ổn định vốn có của nó khiến nó trở thành một chủ đề lý tưởng để khám phá và chứng minh tính hiệu quả của các chiến lược kiểm soát khác nhau, đặc biệt là trong môi trường giáo dục và nghiên cứu. Thách thức của việc cân bằng một con lắc ngược phản ánh các vấn đề điều khiển trong thế giới thực được tìm thấy trong nhiều ứng dụng, từ robot đến điều khiển tư thế con người.

Trong hầu hết các ứng dụng, con lắc ngược được giới hạn trong một độ tự do bằng cách gắn cán vào một trục quay. Trái với con lắc thông thường, con lắc ngược không ổn định theo bản chất và phải được cân bằng hoạt động để duy trì đứng. Điều này có thể thực hiện bằng cách áp dụng một mô-men xoắn tại điểm treo, di chuyển điểm treo ngang như một phần của hệ thống phản hồi, thay đổi tốc độ quay của khối lượng được gắn trên cán theo trực song song với trục treo và tạo ra một mô-men xoắn net trên con lắc, hoặc dao động điểm treo theo chiều dọc.

Có các loại con lắc ngược:

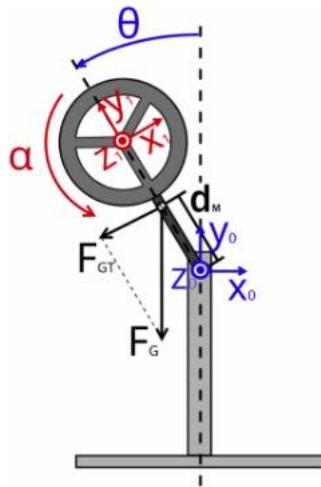
- Con lắc ngược quay (rotary inverted pendulum)



Hình 2.1 Mô hình động lực học con lắc ngược quay

Con lắc ngược dạng quay như hình bên dưới ta thấy đầu trên chuyển động tự do, đầu dưới được kết nối với cánh tay đòn. Con lắc được cân bằng nhờ chuyển động xoay của cánh tay tạo ra một momen giúp cân bằng con lắc về vị trí ta mong muốn.

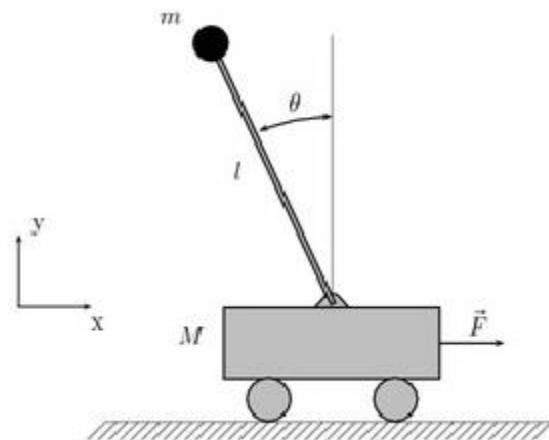
- Con lắc ngược dạng bánh xe (wheel inverted pendulum)



Hình 2.2 Mô hình động lực học con lắc ngược dạng bánh xe

Khác với hai loại ở trên, loại con lắc ngược dạng bánh xe đầu trên được kết nối với một bánh xe có khả năng xoay, đầu dưới kết nối với giá đỡ. Khi con lắc nghiêng sang phải do ngoại lực tác động từ bên trái đến bánh xe sẽ xoay theo chiều kim đồng hồ để tạo ra gia tốc quán tính của bánh xe từ đó sinh ra lực ổn định theo chiều ngược lại giúp con lắc cân bằng.

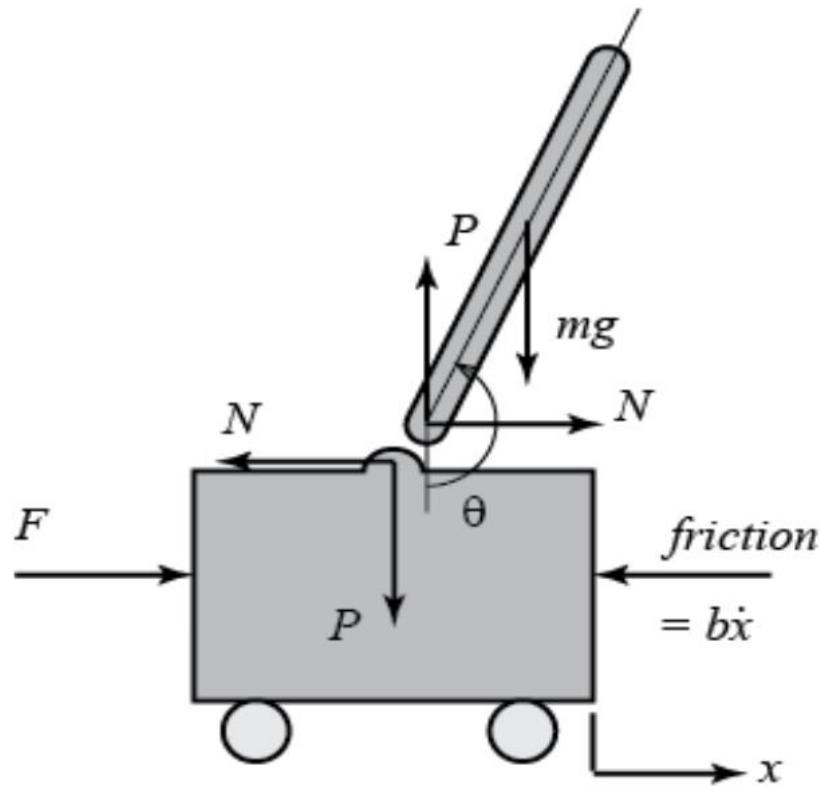
- Con lắc ngược trên xe (pole and cart)



Hình 2.3 Mô hình động lực học con lắc ngược trên xe

Con lắc ngược dạng xe đẩy với đầu trên chuyển động tự do, đầu dưới được kết nối với xe đẩy. Để cân bằng con lắc ta tác động mỗi lực theo phương ngang vào xe đẩy sao cho con lắc duy trì ở vị trí mà ta mong muốn.

2.2 Mô hình hóa toán học con lắc ngược trên xe [7]



Hình 2.4 Mô hình toán học con lắc ngược trên xe

Tiến hành tổng hợp các lực tác động vào xe theo phương ngang ta được phương trình chuyển động:

$$M\ddot{x} + b\dot{x} + N = F \quad (1)$$

Tổng hợp các lực của thanh con lắc theo phương ngang ta được:

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \quad (2)$$

Thay phương trình (2) vào phương trình (1) được:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \quad (3)$$

Tổng hợp các lực vuông góc với thanh con lắc:

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \quad (4)$$

Tổng hợp moment tại trọng tâm thanh con lắc:

$$-Pl\sin\theta - Nl\cos\theta = J\ddot{\theta} \quad (5)$$

Thay phương trình (4) vào phương trình (5) ta được:

$$(J + ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}cos\theta \quad (6)$$

Từ phương trình (3) và (6) ta có hệ phương trình mô tả đặc tính động học phi tuyến của hệ thống con lắc ngược:

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2sin\theta = F \quad (7)$$

$$(J + ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}cos\theta \quad (8)$$

Ta biến đổi (7) và (8) như sau:

$$\ddot{x} = \frac{F - b\dot{x} - ml\ddot{\theta}cos\theta + ml\dot{\theta}^2sin\theta}{M+m} \quad (9)$$

$$\ddot{\theta} = \frac{-mglsin\theta - ml\ddot{x}cos\theta}{J+ml^2} \quad (10)$$

Thay các phương trình (9) và (10) vào các phương trình (7) và (8) ta được phương trình toán của hệ con lắc ngược phi tuyến:

$$\ddot{x} = \frac{(J+ml^2)(F - b\dot{x} - ml\dot{\theta}^2sin\thetacos\theta) + m^2l^2gsin\thetacos\theta}{(J+ml^2)(M+m) - m^2l^2cos\theta^2} \quad (11)$$

$$\ddot{\theta} = \frac{ml(b\dot{x}cos\theta - Fcos\theta - ml\dot{\theta}^2sin\thetacos\theta + (M+m)gsin\theta)}{(J+ml^2)(M+m) - m^2l^2cos\theta^2} \quad (12)$$

Để đơn giản hóa hệ thống ta bỏ qua khối lượng quán tính con lắc, mô hình toán phi tuyến của hệ con lắc ngược được xác định như sau:

$$\ddot{x} = \frac{F + ml\dot{\theta}^2sin\theta - mgsin\thetacos\theta}{M+m - mcos\theta^2} \quad (13)$$

$$\ddot{\theta} = \frac{Fcos\theta - (M+m)gsin\theta + ml\dot{\theta}^2sin\thetacos\theta}{mlcos\theta^2 - (M+m)l} \quad (14)$$

Trong đó:

M: Khối lượng xe

m: Khối lượng con lắc

l: Chiều dài từ tâm thanh con lắc đến điểm quay

g: Gia tốc trọng trường

J: moment quán tính của con lắc

b: hệ số ma sát của xe

N: Phản lực theo phương ngang

F: Lực tác động lên xe

x : Tọa độ vị trí xe

\dot{x} : Vận tốc của xe

\ddot{x} : Gia tốc của xe

θ : góc con lắc so với phương thẳng đứng

$\dot{\theta}$: Vận tốc góc của con lắc

$\ddot{\theta}$: Gia tốc góc của con lắc

Mối quan hệ giữa lực tác dụng f và điện áp điều khiển động cơ u :

$$u = L_m \cdot \frac{di}{dt} + R_m \cdot i + u_b \quad (15)$$

$$u_b = K_b \cdot \dot{\theta} \quad (16)$$

Vì $L_m \cdot \frac{di}{dt}$ rất nhỏ nên bỏ qua, (15) trở thành:

$$u = R_m \cdot i + u_b \quad (17)$$

Suy ra:

$$i = \frac{u - K_b \cdot \dot{\theta}}{R_m} \quad (18)$$

Moment xoắn nội của động cơ:

$$\tau_m = K_t \cdot i = K_t \cdot \frac{u - K_b \cdot \dot{\theta}}{R_m} = \frac{K_t \cdot u}{R_m} - \frac{K_t \cdot K_b \cdot \dot{\theta}}{R_m} \quad (19)$$

Mối quan hệ giữa vận tốc dài và vận tốc góc:

$$x = R \cdot \theta \quad (20)$$

$$\dot{x} = R \cdot \dot{\theta} \quad (21)$$

Thay (21) và (19) ta được:

$$\tau_m = \frac{K_t \cdot u}{R_m} - \frac{K_t \cdot K_b \cdot \dot{x}}{R_m \cdot R} \quad (22)$$

Lực do động cơ tạo ra:

$$f = \frac{\tau_m}{R} = \frac{K_t \cdot u}{R_m \cdot R} - \frac{K_t \cdot K_b \cdot \dot{x}}{R_m \cdot R^2} \quad (23)$$

Trong đó:

u : điện áp cấp vào cho động cơ

u_b : điện áp sinh ra bởi hằng số phản điện

i : dòng điện chạy trong động cơ

K_t : Hằng số moment

K_b : Hằng số phản điện

R : Bán kính puly

u : Điện áp cấp vào cho động cơ

τ_m : Moment xoắn nội của động cơ

R_m : Điện trở động cơ

L_m : Hệ số điện kháng

Mối quan hệ giữa moment lực và gia tốc góc:

Theo phương trình định luật II Newton:

$$F = m \cdot a \quad (24)$$

Mối quan hệ giữa gia tốc tiếp tuyến và gia tốc góc:

$$a = L \cdot \alpha \quad (25)$$

Thay (25) vào (24) ta được:

$$F = m \cdot L \cdot \alpha \quad (26)$$

Nhân hai vế cho l:

$$F \cdot L = m \cdot L^2 \cdot \alpha \quad (27)$$

Moment lực:

$$M = F \cdot L \quad (28)$$

Moment quán tính:

$$I = m \cdot L^2 \quad (29)$$

Thay (28) và (29) vào (27) ta được:

$$M = I \cdot \alpha \quad (30)$$

Suy ra:

$$\alpha = \frac{M}{I} \quad (31)$$

Đối với thanh thẳng đồng chất, có khối lượng m , chiều dài L , trực quay ở một đầu:

$$I = \frac{m \cdot L^2}{3} \quad (32)$$

Thay (32) và (28) vào (31) ta được:

$$\alpha = \frac{M}{I} = \frac{F \cdot L}{\frac{m \cdot L^2}{3}} = \frac{3F}{m \cdot L} \quad (33)$$

F: Ngoài lực tác dụng

a: gia tốc tiếp tuyến

L: chiều dài thanh con lắc

α : Gia tốc góc

M: Moment lực

I: Moment quán tính

2.3 Mô hình hóa con lắc ngược trên xe bằng hàm truyề [8]

Tổng hợp các lực của xe đẩy theo phương ngang ta có phương trình

$$M\ddot{x} = u - H \quad (1)$$

Tổng hợp các lực của con lắc theo phương ngang ta có phương trình

$$m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = H \quad (2)$$

Thay (2) vào (1) ta có phương trình

$$(M + m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = u \quad (3)$$

Tổng hợp lực theo phương dọc của con lắc và xe đẩy ta có phương trình

$$mg - ml\ddot{\theta}\sin\theta - ml\dot{\theta}\cos\theta = V \quad (4)$$

Ta tính tổng các momen xung quanh trọng tâm con lắc để loại bỏ V và H để có phương trình sau

$$-Vl\sin\theta - Hl\cos\theta = J\ddot{\theta} \quad (5)$$

Ta thay phương trình (2),(4) vào (5) sẽ được phương trình sau:

$$(J + ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}\cos\theta \quad (6)$$

Ta sẽ tuyến tính hóa các phương trình về vị trí cân bằng hướng lên trên theo phương thẳng đứng ($\theta=\pi$), và giả sử độ lệch của hệ thống này nằm trong khoảng rất nhỏ so với vị trí cân bằng, gọi ϕ là độ lệch so với vị trí cân bằng ($\theta=\pi + \phi$). Giả sử ϕ nhỏ đến mức ta có thể sử dụng các phép tính gần đúng cho các góc trong các hàm phi tuyến tính của hệ thống.

$$\sin\theta = \sin(\pi + \phi) \approx -\phi$$

$$\cos\theta = \cos(\pi + \phi) \approx -1$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0$$

Ta có:

$$(J + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \quad (7)$$

$$(M + m)\ddot{x} - ml\ddot{\phi} = u \quad (8)$$

Hàm truyền

Để có được hàm truyền của các phương trình tuyến tính, ta biến đổi Laplace của các phương trình hệ thống. Từ đó ta có các phương trình sau:

$$(J + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \quad (9)$$

$$(M + m)X(s)s^2 - ml\Phi(s)s^2 = U(s) \quad (10)$$

Hàm truyền là mối quan hệ giữa một đầu ra và một đầu vào tại một điểm. Để có hàm truyền với đầu vào là $U(s)$ và đầu ra là $\Phi(s)$ thì ta cần loại bỏ $X(s)$ khỏi các phương trình trên.

$$X(s) = \left[\frac{j+ml^2}{ml} - \frac{g}{s^2} \right] \Phi(s) \quad (11)$$

Sau đó thay thế phương trình (11) vào (10) ta có phương trình sau:

$$(M + m) \left[\frac{j+ml^2}{ml} - \frac{g}{s^2} \right] \Phi(s) s^2 - ml\Phi(s)s^2 = U(s) \quad (12)$$

$$\Rightarrow \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 - \frac{(M+m)mgl}{q}s} \quad (13)$$

Với :

$$q = [(M + m)(J + ml^2) - (ml)^2] \quad (14)$$

Hàm truyền con lắc

$$\text{Conlac}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 - \frac{(M+m)mgl}{q}s} \quad \left[\frac{\text{rad}}{\text{N}} \right]$$

Hàm truyền xe đẩy

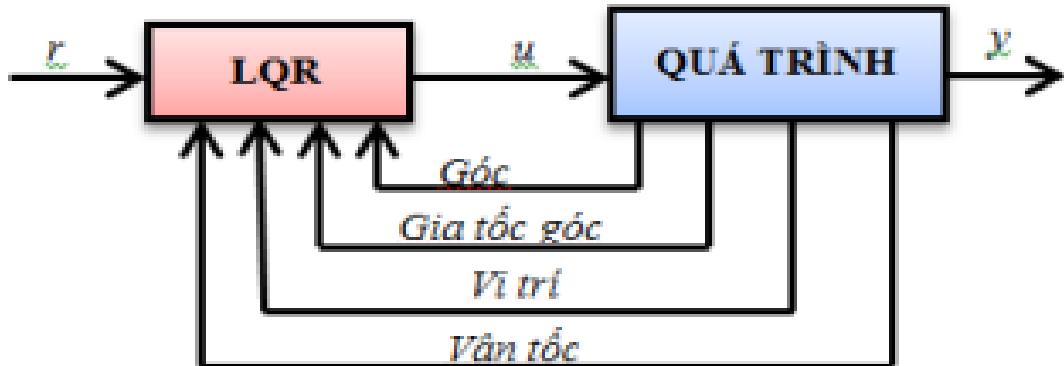
$$\text{Xeday}(s) = \frac{X(s)}{U(s)} = \frac{\frac{(j+ml^2)s^2 - gml}{q}}{s^4 - \frac{(M+m)mgl}{q}s^2} \quad \left[\frac{\text{m}}{\text{N}} \right]$$

2.4 Một số bộ điều khiển thông dụng cho con lắc ngược

2.4.1 Bộ điều khiển LQR

LQR (Linear Quadratic Regulator) là thuật toán điều khiển đợc xây dựng dựa trên cơ sở nguyên lý phản hồi trạng thái, còn gọi là phương pháp tuyến tính hóa dạng toàn phuong. Bộ điều khiển LQR thường được áp dụng trên các hệ phi tuyến với nhiều ngõ vào ra. Bộ điều khiển nhận tín hiệu vào là trạng thái của hệ thống và tín hiệu mẫu sau đó tính toán và chuyển thành tín hiệu điều khiển cho hệ thống.

Một hệ điều khiển đợc thiết kế ở chế độ làm việc tốt nhất là hệ luôn ở trạng thái tối ưu theo một tiêu chuẩn chất lượng nào đó (đạt giá trị cực trị). Trạng thái tối ưu có đạt đợc hay không tùy thuộc vào yêu cầu chất lượng đặt ra, sự hiểu biết về đối tượng và các tác động lên đối tượng, điều kiện làm việc của hệ,....

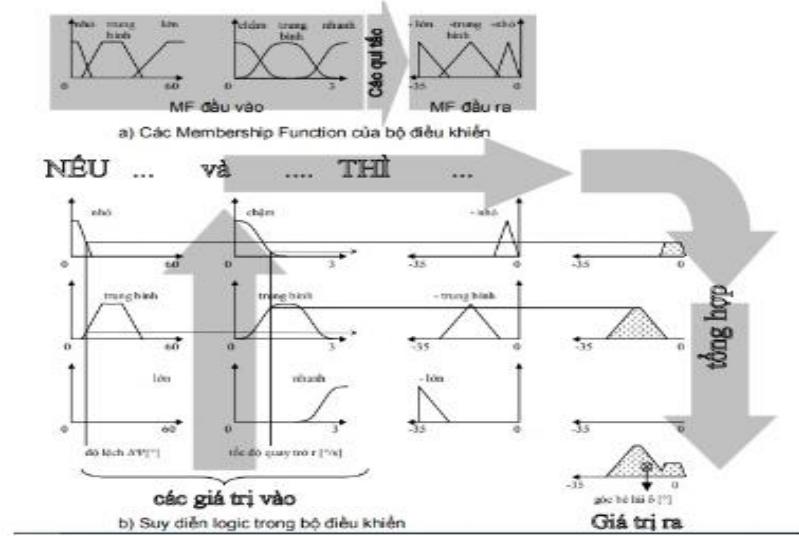


Hình 2.5 Sơ đồ bộ điều khiển LQR

2.4.2 Bộ điều khiển Fuzzy Logic

Biểu thức Fuzzy Logic có thể nhận một trong vô số giá trị nằm trong khoảng số thực (0 đến 1). Khác với kiểu Logic truyền thống chỉ có thể nhận “đúng” hoặc “sai” thì Fuzzy Logic thì mức độ đánh giá nằm trong khoảng 0 đến 1 theo mức độ “đúng” nhiều hay “Ít”. Ví dụ như “chậm”, “trung bình”, “nhanh”, “nóng”, “vừa”, “lạnh”,....

Từ một giá trị cụ thể của đầu vào (MF), bộ điều khiển tiến hành “mờ hóa” tại các đầu vào, xác định xem tín hiệu đó thuộc mức độ nào trong các mức đã định nghĩa trước đó. Sau đó các khái niệm này được xử lý theo quy tắc bộ điều khiển và đưa ra kết quả là một miền giá trị. Từ miền giá trị này khói “giải mờ” biến đổi ngược và cụ thể cho đầu ra.



Hình 2.6 Nguyên lý hoạt động của bộ điều khiển logic mở

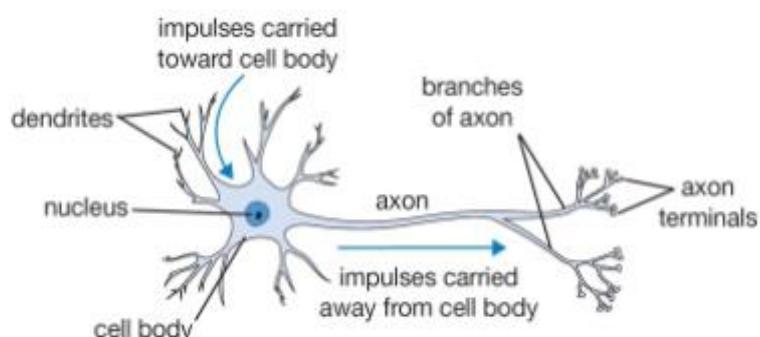
2.4.3 Điều khiển Mạng nơ-ron

Mạng nơ-ron là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người gồm vô số các nơ-ron liên kết với nhau. Khi kết hợp cùng các kỹ thuật học sâu(DEEP LEARNING) trở thành công cụ có khả năng xử lý các bài toán khó liên quan đến hình ảnh, giọng nói,... [10].

Đặc tính của mạng nơ-ron là quá trình tính toán được tiến hành song song và phân tán trên nhiều nơ-ron gần như đồng thời. Thực chất không phải là sơ đồ được định sẵn trước là tính toán là quá trình học.

Có nhiều cấu trúc điều khiển sử dụng mạng nơ-ron như vòng lặp kín-mở, mô hình tham chiếu, overtime,...

Mạng nơ-ron được ứng dụng trong nhiều lĩnh vực như: nhận diện giọng nói, nhận diện khuôn mặt,....



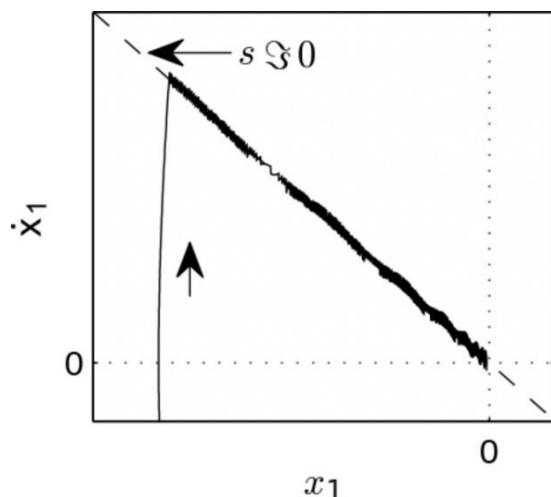
Hình 2.7 Mạng nơ-ron nhân tạo

2.4.4 Điều khiển trượt (Sliding mode control)

Phương pháp điều khiển trượt là phương pháp điều khiển có cấu trúc thay đổi. Các cấu trúc được thiết kế sao cho quỹ đạo luôn di chuyển về phía vùng lân cận có cấu trúc điều khiển khác, quỹ đạo cuối cùng sẽ không tồn tại trong một cấu trúc mà sẽ “trượt” theo dọc ranh giới giữa các cấu trúc điều khiển.

Trong điều khiển trượt một bề mặt được xác định và hệ thống sẽ điều khiển sao cho sai số giữa trạng thái thực tế và bề mặt trượt tiên dàn về không mặc dù là tín hiệu có thể là hàm không liên tục.

Điều khiển trượt được ứng dụng trong chuyên động robot, xe đi dưới nước và nhiều hệ thống phi tuyến khác,....



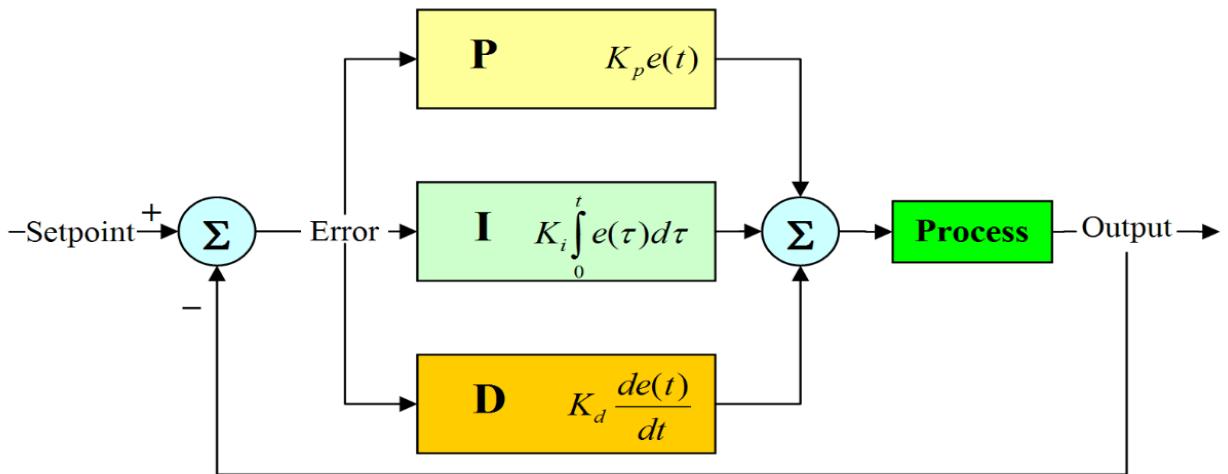
Hình 2.8 Bộ điều khiển trượt

2.4.5 Điều khiển PID [8]

Một bộ điều khiển vi tích phân tỉ lệ (bộ điều khiển PID- Proportional Integral Derivative) là một cơ chế phản hồi vòng điều khiển (bộ điều khiển) tổng quát được sử dụng rộng rãi trong các hệ thống điều khiển công nghiệp – bộ điều khiển PID là bộ điều khiển được sử dụng nhiều nhất trong các bộ điều khiển phản hồi. Bộ điều khiển PID sẽ tính toán giá trị "sai số" là hiệu số giữa giá trị đo thông số biến đổi và giá trị đặt mong muốn. Bộ điều khiển sẽ thực hiện giảm tối đa sai số bằng cách điều chỉnh giá trị điều khiển đầu vào.

Giải thuật tính toán bộ điều khiển PID bao gồm 3 thông số riêng biệt, do đó đôi khi nó còn được gọi là điều khiển ba khâu: các giá trị tỉ lệ, tích phân và đạo hàm, viết tắt là P, I, và D. Giá trị tỉ lệ xác định tác động của sai số hiện tại, giá trị tích phân xác định tác động

của tổng các sai số quá khứ và giá trị vi phân xác định tác động của tốc độ biến đổi sai số. Tổng chập của ba tác động này dùng để điều chỉnh quá trình thông qua một phần tử điều khiển. Nhờ vậy, những giá trị này có thể làm sáng tỏ về quan hệ thời gian: P phụ thuộc vào sai số hiện tại, I phụ thuộc vào tích lũy các sai số quá khứ và D dự đoán các sai số tương lai, dựa vào tốc độ thay đổi hiện tại [12].



Hình 2.9 Bộ điều khiển PID

Bộ điều khiển PID chỉ có thể điều khiển một thông số của hệ thống, để điều khiển được góc con lắc và vị trí của xe con lắc tại cùng một thời điểm thì chúng ta cần hai bộ điều khiển PID. Trong đó một thông số được xem như là thông số chính và được điều khiển trực tiếp moment của động cơ, trong khi đó thông số còn lại đợt lọc đợt lọc áp vào tác động của điểm tham chiếu của thông số chính. Hai tín hiệu đầu vào được đưa vào bộ điều khiển PID và đầu ra là tín hiệu lực tác động vào xe. Để cho con lắc ổn định cần sử dụng một bộ điều khiển hồi tiếp. Việc sử dụng bộ điều khiển hồi tiếp, dữ liệu ngõ ra sẽ có thêm nhiều thông tin để mô tả hệ thống.

Hệ số

Gain là thuật ngữ được sử dụng cho “hệ số nhân”. Bằng cách điều chỉnh cài đặt khuếch đại (hoặc hệ số nhân) của tỷ lệ, tích phân và đạo hàm, người dùng có thể kiểm soát mức độ ảnh hưởng của bộ điều khiển PID đến đầu ra và cách bộ điều khiển sẽ phản ứng với những thay đổi khác nhau trong giá trị quy định.

Khâu tỉ lệ (P)

Tỷ lệ được tính bằng cách nhân P-Gain với sai số. Mục đích của tỷ lệ là tạo ra phản ứng tức thời ở đầu ra để đưa giá trị xử lý gần đến với điểm đặt. Khi sai số giảm đi thì ảnh hưởng của giá trị tỷ lệ đầu ra sẽ ít hơn.

Khâu tích phân (I)

Tích phân được tính bằng cách nhân I-Gain với lỗi, sau đó nhân giá trị này với thời gian chu kỳ của bộ điều khiển (tần suất bộ điều khiển thực hiện phép tính PID) và liên tục tích lũy giá trị này dưới dạng “tổng tích phân”.

Giải thích thêm một chút, mỗi khi bộ điều khiển thực hiện phép tính PID (ví dụ về thời gian chu kỳ là 100 ms), giá trị tích phân mới được tính sẽ được thêm vào tổng tích phân. Tích phân thường sẽ không có nhiều ảnh hưởng ngay lập tức đến đầu ra như tỷ lệ, nhưng do tích phân liên tục tích lũy theo thời gian, nên giá trị xử lý càng mất nhiều thời gian để đạt đến điểm đặt thì tích phân sẽ càng ảnh hưởng nhiều đến đầu ra .

Khâu đạo hàm (D)

Đạo hàm được tính bằng cách nhân D-Gain với tốc độ tăng dần của giá trị quy trình. Mục đích của đạo hàm là để “dự đoán” nơi giá trị xử lý sẽ diễn ra và làm lệch đầu ra theo hướng ngược lại của tỷ lệ và tích phân, nhằm hy vọng ngăn bộ điều khiển vượt quá điểm đặt nếu tốc độ tăng quá nhanh.

Giải thích đơn giản hơn một chút, nếu giá trị quá trình tiếp cận điểm đặt quá nhanh, đạo hàm sẽ giới hạn đầu ra để ngăn giá trị quá trình vượt quá điểm đặt.

Đầu ra (Output)

Đầu ra của bộ điều khiển PID được tính toán bằng cách chỉ cần thêm Tỷ lệ, Tích phân và Đạo hàm. Tùy thuộc vào cài đặt khuếch đại của ba giá trị này, sẽ xác định mức độ ảnh hưởng của chúng đối với đầu ra.

$$Output = P + I + D$$

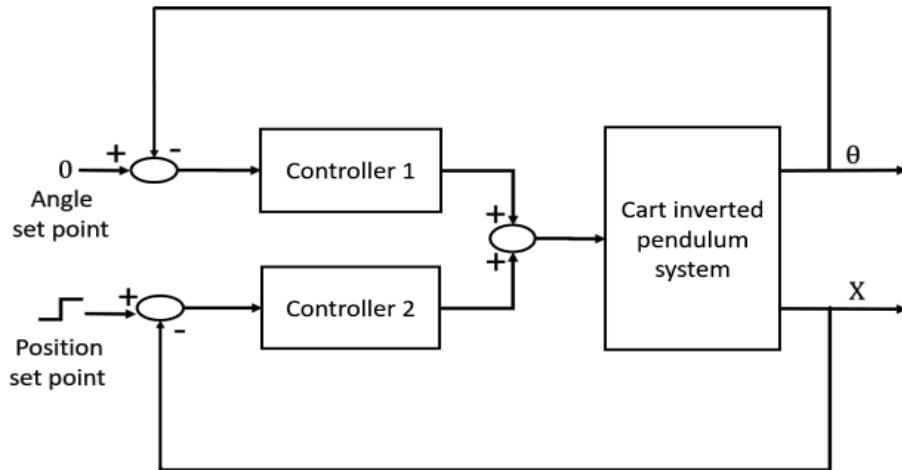
Trong phần PID, quỹ đạo nghiệm số và đáp ứng tần số của bài toán này, chúng ta sẽ chỉ quan tâm đến việc điều khiển vị trí của con lắc. Điều này là do các kỹ thuật được sử dụng trong các phần này phù hợp nhất với các hệ thống một đầu vào, một đầu ra (SISO). Do đó, không có tiêu chí thiết kế nào đề cập đến vị trí của xe đẩy. Tuy nhiên, chúng tôi sẽ điều tra tác động của bộ điều khiển lên vị trí của xe đẩy sau khi bộ điều khiển được thiết kế. Đối với những phần này, chúng ta sẽ thiết kế một bộ điều khiển để khôi phục con lắc

về vị trí thẳng đứng hướng lên sau khi nó chịu một lực "va chạm" xung vào xe đẩy. Cụ thể, tiêu chí thiết kế là con lắc trở về vị trí thẳng đứng.

2.5 Phương pháp điều khiển cân bằng PID cho hệ con lắc ngược trên xe

2.5.1 Bộ điều khiển PID 2 bậc tự do

Trong phương pháp điều khiển này, được chia làm 2 bộ điều khiển PID riêng biệt, một bộ điều khiển vị trí con lắc, bộ còn lại điều khiển vị trí xe. Tuy nhiên, hai giá trị đầu ra của hai bộ sẽ là tham chiếu với nhau để đưa ra tín hiệu điện áp điều khiển. Bộ điều khiển PID chỉ có thể điều khiển một thông số của hệ thống, để điều khiển góc con lắc và vị trí của xe con lắc tại cùng một thời điểm thì chúng ta cần hai bộ điều khiển PID. Trong đó một thông số xem như là thông số chính và điều khiển trực tiếp momen của động cơ, trong khi đó thông số còn lại áp vào tác động của điểm tham chiếu của thông số chính.



Hình 2.10 Sơ đồ bộ điều khiển PID cho hệ SIMO

Hai tín hiệu đầu vào bộ điều khiển PID và đầu ra là tín hiệu lực tác động vào xe. Để cho con lắc ổn định cần sử dụng một bộ điều khiển hồi tiếp. Việc sử dụng bộ điều khiển hồi tiếp, dữ liệu ngõ ra sẽ có thêm nhiều thông tin để mô tả hệ thống. Giá trị hồi tiếp của góc lệch con lắc so với phương thẳng đứng được so sánh với giá trị đặt. Bộ điều khiển PID1 sẽ tính toán giá trị ngõ ra dựa trên giá trị sai lệch này và quyết định giá trị điện áp đặt lên động cơ kéo con lắc. Ban đầu, ta tiến hành thay đổi thông số bộ điều khiển PID1 để xác định đáp ứng góc lệch θ con lắc. Sau đó, ta thiết lập sơ đồ khởi điều khiển con lắc ổn định với bộ điều khiển hai bộ điều khiển PID có hai biến hồi tiếp là góc θ và vị trí xe x . Sơ đồ này được gọi là bộ điều khiển PID thỏa hiệp từ hai bộ điều khiển PID một biến để quyết định giá trị điện áp đặt lên động cơ kéo xe con lắc [13].

Dấu tham chiếu sẽ phụ thuộc vào chiều đọc encoder, chiều quay động cơ nhưng phải thỏa trường hợp, khi con lắc nghiêng về bên phải thì xe phải di chuyển về bên đó để đáp ứng khả năng cân bằng.

2.5.2 Bộ điều khiển Cascade PID

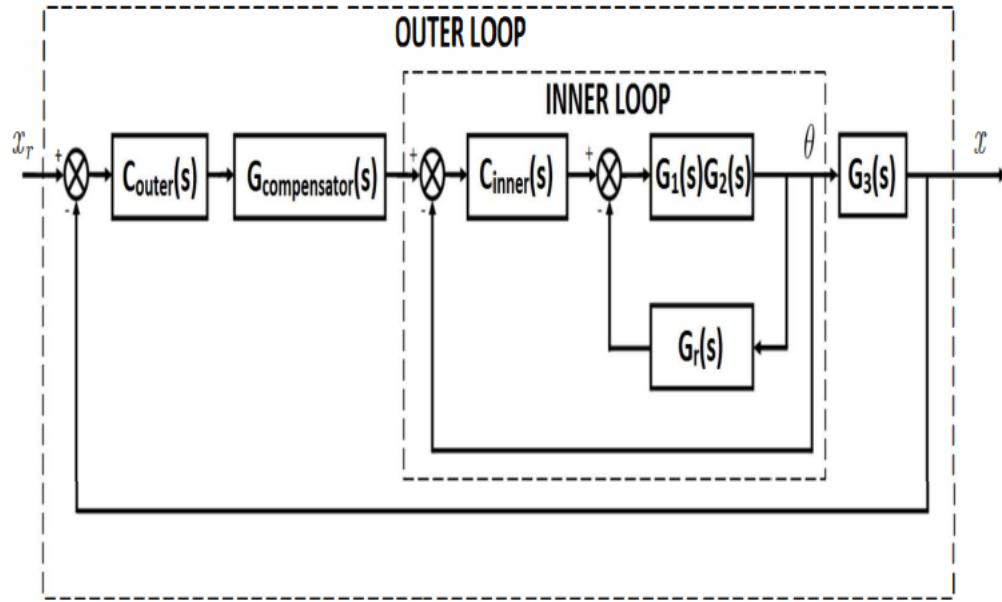
Cascade controller hay còn gọi là bộ điều khiển nối tầng, thể hiện một ưu điểm của bộ điều khiển PID là sử dụng cùng một lúc 2 bộ PID với nhau để đạt được kết quả động học tốt nhất. Trong điều khiển ghép tầng, có hai bộ PID được sắp xếp với một bộ PID điều khiển điểm đặt của bộ còn lại hay nói cách khác đầu ra của bộ PID1 và đầu vào của bộ PID2. Một bộ điều khiển sẽ đóng vai trò như bộ điều khiển vòng ngoài, nó điều khiển các thông số vật lý chính. Bộ điều khiển kia làm việc như vòng trong, sẽ đọc đọc đầu ra của bộ điều khiển vòng ngoài và tính toán cho ra tín hiệu đầu ra đưa vào điều khiển hệ thống, thường sẽ là các thông số điều khiển có tốc độ thay đổi nhanh [14].

Một số yêu cầu của bộ điều khiển cascade PID:

- Bộ điều khiển vòng trong phải có tác động đáng kể đến bộ điều khiển vòng ngoài và mang tính liên tục bởi vì theo cấu trúc cascade, bộ điều khiển vòng trong không thể bị ảnh hưởng bởi chính nó.
- Bộ điều khiển vòng ngoài phải làm việc nhanh hơn bộ điều khiển vòng trong. Bộ điều khiển vòng ngoài phải đáp ứng vòng lặp trong nhanh hơn vòng trong từ 3 – 4 lần so với bộ điều khiển vòng trong đáp ứng đến vòng ngoài. Điều này đảm bảo bộ điều khiển vòng ngoài có đủ thời gian để giảm sự nhiễu loạn của vòng lặp bên trong. Ngoài ra, nó đảm bảo sự nhiễu loạn từ bên trong không ảnh hưởng đến vòng lặp chính.
- Bộ điều khiển vòng ngoài phải ít nhiễu hơn bộ điều khiển vòng trong. Nếu hiện tượng nhiễu loạn hay sự bất ổn định của bộ PID vòng ngoài xảy ra nhiều hơn thì đầu vào của bộ điều khiển vòng trong sẽ bị ảnh hưởng và làm cho hệ thống lại càng mất ổn định. Vì mục đích của cascade là dùng bộ điều khiển vòng ngoài để ổn định bộ điều khiển vòng trong.

Một số lợi ích của bộ điều khiển Cascade PID:

- Giảm độ dao động chung của hệ thống
- Đáp ứng nhanh với nhiễu loạn
- Dễ điều chỉnh bộ thông số nhờ vào vòng ngoài



Hình 2.11 Sơ đồ bộ điều khiển PID cho hệ SIMO [14]

2.6 Các phương pháp điều chỉnh thông số PID [15]

2.6.1 Điều chỉnh thủ công

Cho K_i và K_d bằng 0, tăng K_p cho đến khi thời gian đáp ứng của hệ thống đủ nhanh và overshoot nhỏ. Điều chỉnh K_i để thay đổi ess, tăng K_i từ từ bé đến lớn để đảm bảo giảm ess và không cho xuất hiện overshoot

K_d để loại bỏ overshoot nhưng sẽ xuất hiện ess

2.6.2 Phương pháp Ziegler-Nichols

Đặt K_i và K_d bằng 0, tăng tỷ lệ lên cho đến khi K_u tại đó mà đầu ra của vòng lặp bắt đầu dao động liên tục và chu kỳ dao động T_u được sử dụng để thiết lập các thông số sau:

Loại P: $P=0.5K_u$

Loại PI: $P=0.45K_u$ và $K_i=0.54K_u/T_u$

Loại PID: $P=0.6K_u$, $K_i=1.2K_u/T_u$ và $K_d=3K_uT_u/40$

2.6.3 Phương pháp chuyển tiếp

Sử dụng điều khiển bang-bang và đo các dao động tổng hợp. Các giá trị được chọn sao cho quy trình sẽ vượt qua điểm đặt (không nhất thiết là 0% và 100%) bằng cách chọn được bộ thông số phù hợp có thể tránh được những giao động nguy hiểm

2.6.4 Phương pháp cohen-coon

Dựa trên hàm bậc nhất và độ trễ thời gian. Giống như phương pháp Ziegler-Nichols, một tập các tham số điều chỉnh được phát triển để mang lại phản hồi vòng kín với tỷ lệ phân rã là 1/4.

2.6.5 Phương pháp điều chỉnh bằng phần mềm

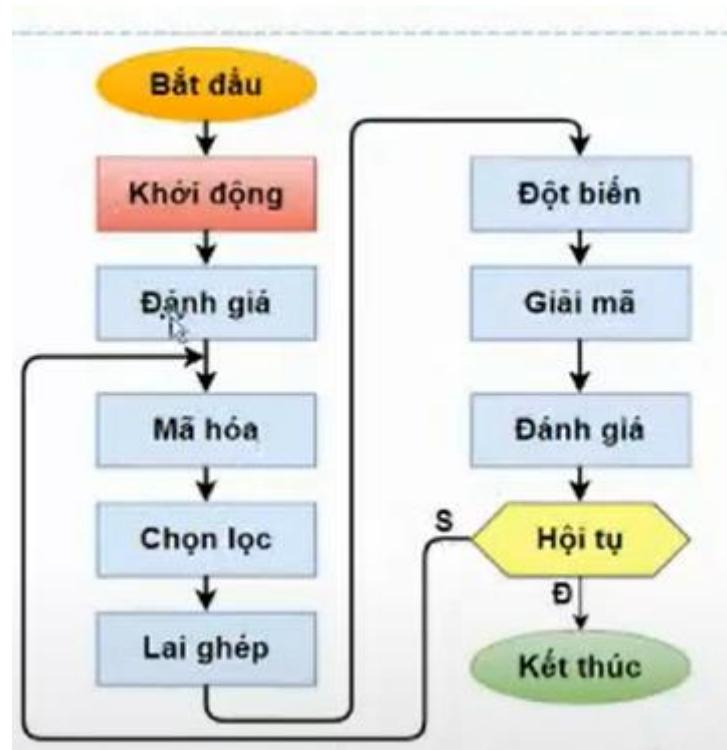
Các Phần mềm thu thập dữ liệu, phát triển mô hình quy trình và đề xuất điều chỉnh tối ưu. Điều chỉnh vòng lặp PID tạo ra một xung trong hệ thống và sau đó sử dụng đáp ứng tần số của hệ thống được điều khiển để thiết kế các giá trị vòng lặp PID.

2.6.6 Dùng giải thuật di truyền (GA) để xác định thông số PID

GA (Generic Algorithm) được lấy cảm hứng từ quá trình tiến hóa sinh học để tồn tại và phát triển của sinh vật trong tự nhiên. Bản chất toán học của GA là giải thuật tìm kiếm theo xác suất. Nó giúp tìm ra giải pháp tối ưu nhất trong điều kiện thời gian và không gian cho phép. Là một kỹ thuật của khoa học máy tính nhằm tìm kiếm giải pháp thích hợp cho các bài toán tối ưu tổ hợp.

Giải thuật GA được mô phỏng thông qua ba quá trình cơ bản của tiến hóa tự nhiên: chọn lọc tự nhiên, lai ghép và đột biến. Các cá thể mới sinh ra trong quá trình tiến hóa nhờ sự lai ghép của thế hệ cha mẹ, mang những tính trạng di truyền của cha mẹ. Tuy nhiên cũng có một số cá thể mang những tính trạng hoàn toàn mới do đột biến. Nhờ vào sự chọn lọc tự nhiên của môi trường (như biến đổi khí hậu, bị kẻ thù săn bắt,...) mà chỉ những cá thể nào thích nghi tốt hơn với môi trường mới có nhiều khả năng tồn tại và phát triển. Cá thể nào không thích nghi được sẽ bị môi trường tự nhiên đào thải.

Tương tự vậy, GA xét đến toàn bộ các giải pháp, bằng cách xét trước nhất một số giải pháp sau đó loại bỏ những thành phần không thích hợp và chọn những thành phần thích hợp hơn để lai tạo và biến hóa nhằm mục đích tạo ra nhiều giải pháp mới có hệ số thích nghi ngày càng cao.



Hình 2.12 Lưu đồ giải thuật di truyền GA

Các bước thực hiện trong một thuật toán di truyền:

- Khởi tạo quần thể ban đầu.
- Lựa chọn các cặp cha mẹ để lai tạo.
- Thực hiện toán tử di truyền: lai ghép, đột biến.
- Đánh giá thế hệ con lai.
- Hợp nhất con lai với quần thể chính và tiếp tục lai tạo cho đến khi tìm được thế hệ con lai thỏa yêu cầu.

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG

3.1 Các thông số cơ bản

$M = 0,3 \text{ kg}$ (Khối lượng của xe)

$m = 0,1 \text{ kg}$ (Khối lượng của con lắc)

$l = 0,1 \text{ m}$ (Chiều dài từ trọng tâm thanh con lắc đến điểm quay)

$g = 9,81 \text{ m/s}^2$ (Gia tốc trọng trường)

$J = 1.33 \times 10^{-3} \text{ kg.m}^2$ (Moment quán tính của con lắc)

$R = 0,012 \text{ m}$ (Bán kính pulley)

Dựa vào động cơ Minitertia, nhóm thu được các thông số cần thiết của động cơ [16]

$K_t = 0.0573398 \text{ N.m/A}$ (Hàng số moment)

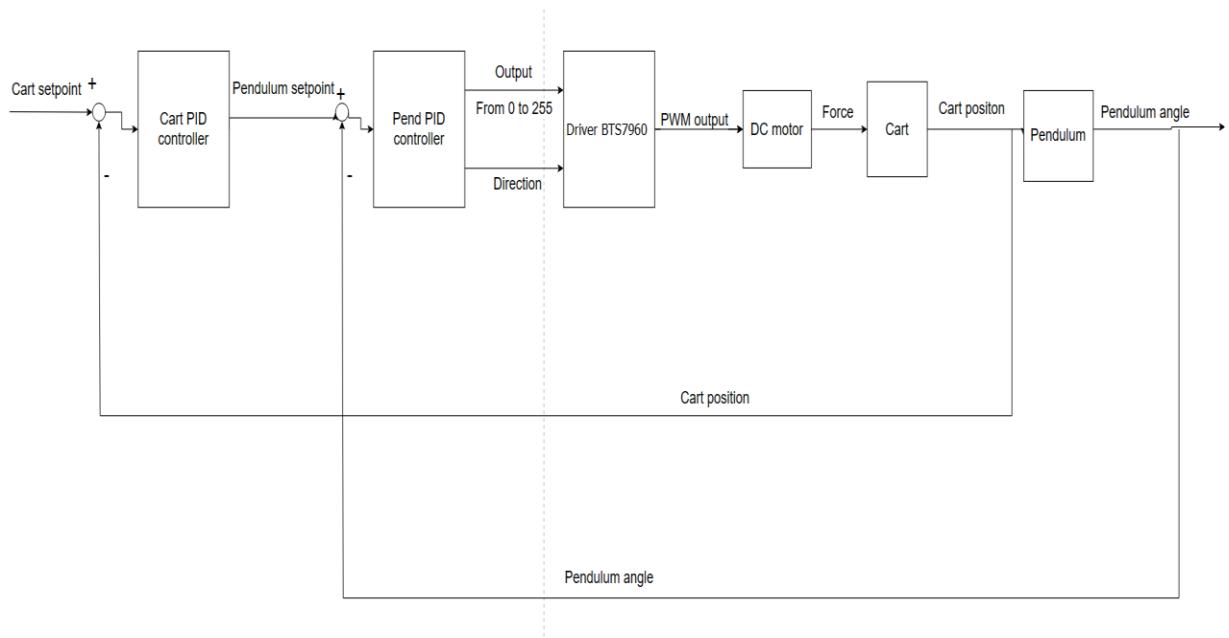
$K_b = 0.05729578 \text{ V.s/rad}$ (Hàng số phản điện)

$R_m = 1.12 \text{ ohm}$ (Điện trở động cơ)

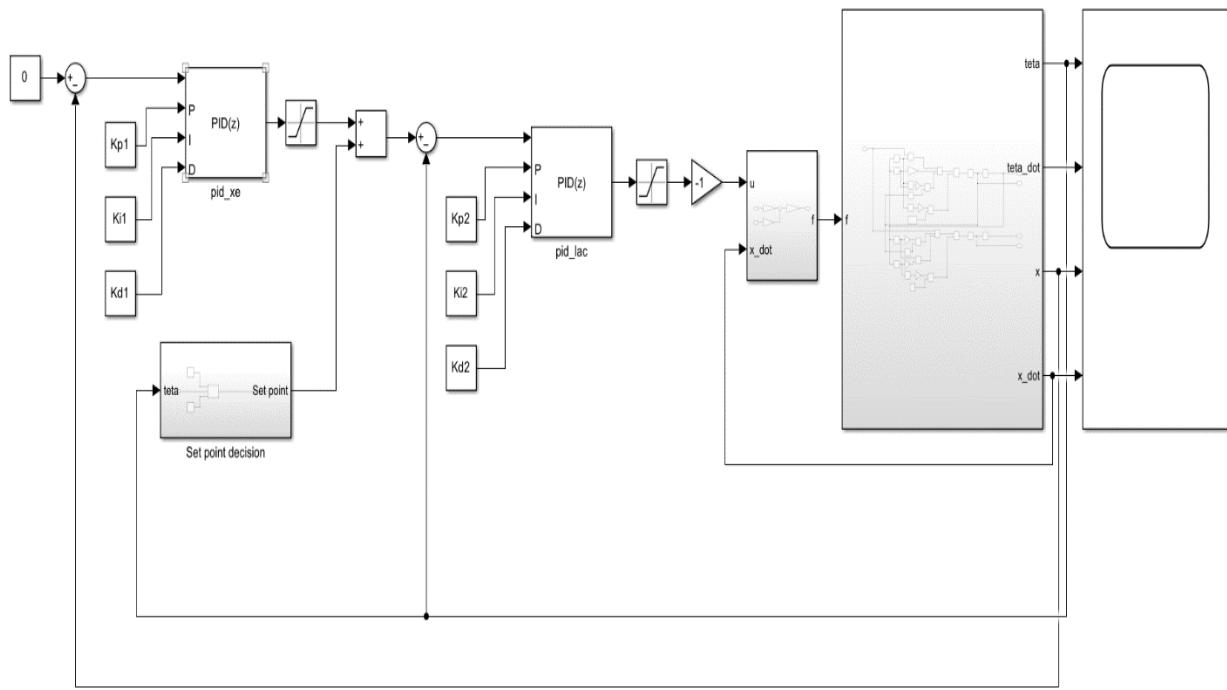
3.2 Xây dựng chương trình mô phỏng bộ điều khiển khiển dựa trên mô hình toán học trên Simulink

3.2.1 Phương pháp điều khiển Cascade

a) Xây dựng mô hình



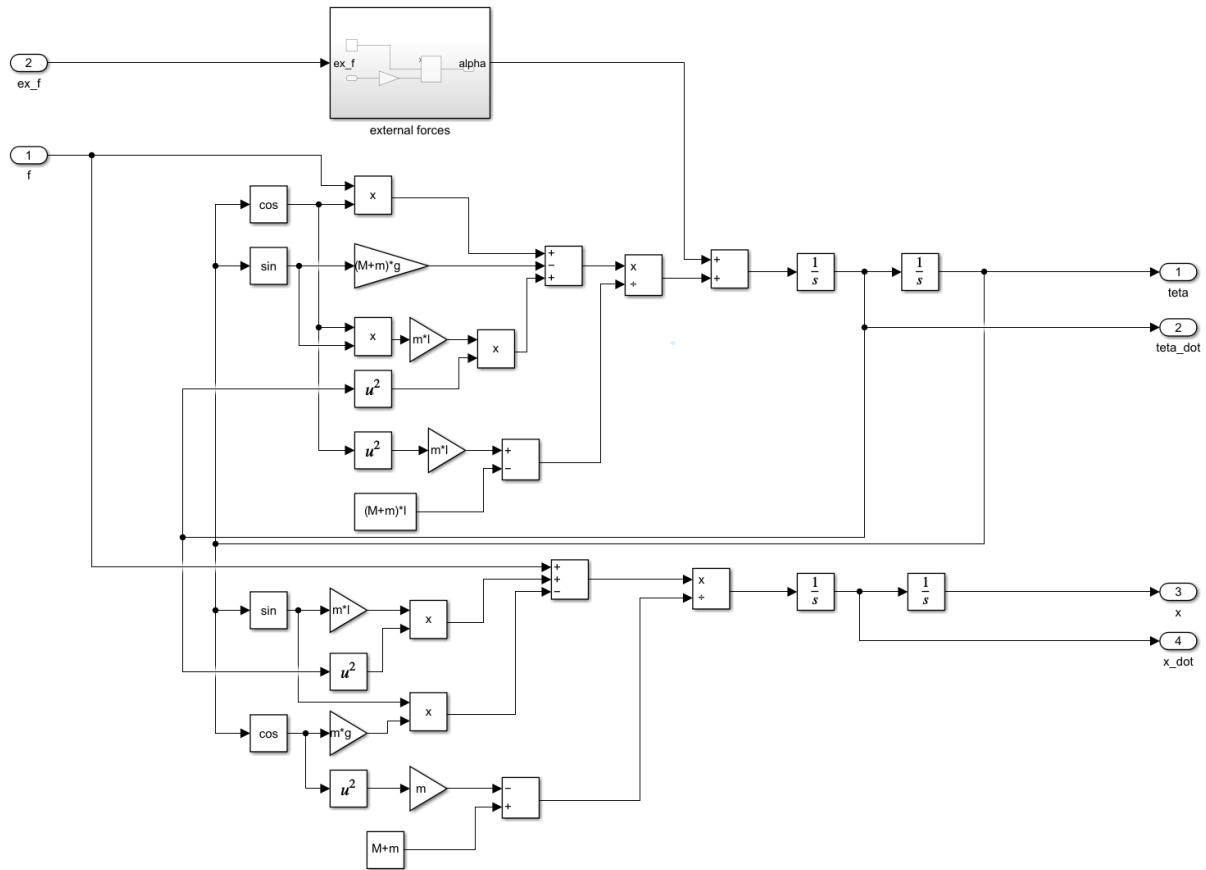
Hình 3.1 Sơ đồ khái mô hình điều khiển Cascade trên simulink



Hình 3.2 Mô phỏng con lắc ngược trên simulink bằng Cascade

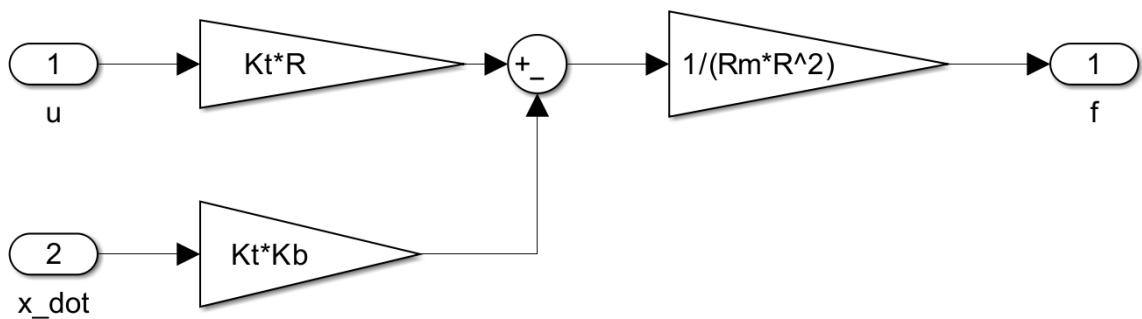
Với phương pháp điều khiển Cascade dùng hai bộ PID để điều khiển vị trí xe trượt và vị trí góc của con lắc. Vòng ngoài là bộ PID điều khiển cho vị trí xe trượt, vòng trong là bộ PID điều khiển cho vị trí con lắc.

Trong chương trình mô phỏng này, ta đặt vị trí mong muốn cho xe trượt là 0. Sau khi thông qua lần lượt hai bộ điều khiển PID, hệ thống tạo ra tín hiệu điện áp. Khối bão hòa có tác dụng giới hạn giá trị điện áp trong khoảng -24V đến 24V. Điện áp thông qua khối chuyển đổi từ điện áp sang lực tác động của động cơ, là đầu vào đưa vào khối mô hình toán hệ con lắc ngược phi tuyến. Và đầu ra của khối mô hình toán hệ con lắc ngược phi tuyến là bốn thành phần: vị trí xe, vận tốc xe, vị trí con lắc và vận tốc góc của con lắc.

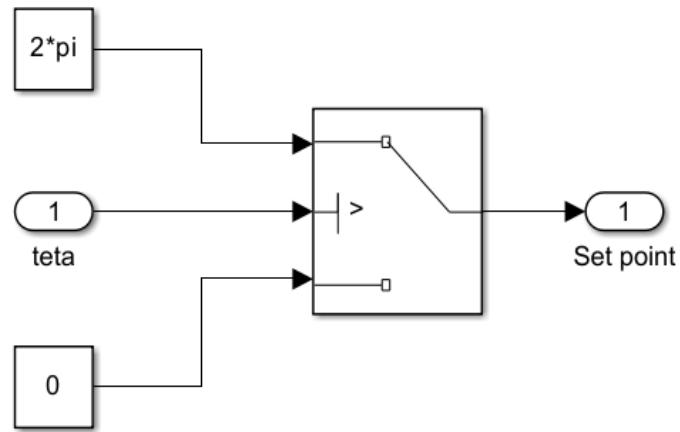


Hình 3.3 Mô hình toán học hệ con lắc ngược phi tuyến

Khối này có tác dụng là mô hình hóa hệ con lắc ngược trong thực tế bằng phương pháp biến đổi toán học. Hệ con lắc ngược của khói có giá trị vị trí con lắc khi chưa điều khiển là π , tức là chiều hướng xuống. Và vị trí cân bằng của con lắc khi điều khiển sẽ là 0.



Hình 3.4 Khối chuyển đổi từ điện áp sang lực tác động của động cơ



Hình 3.5 Khối xác định vị trí mong muốn cho con lắc

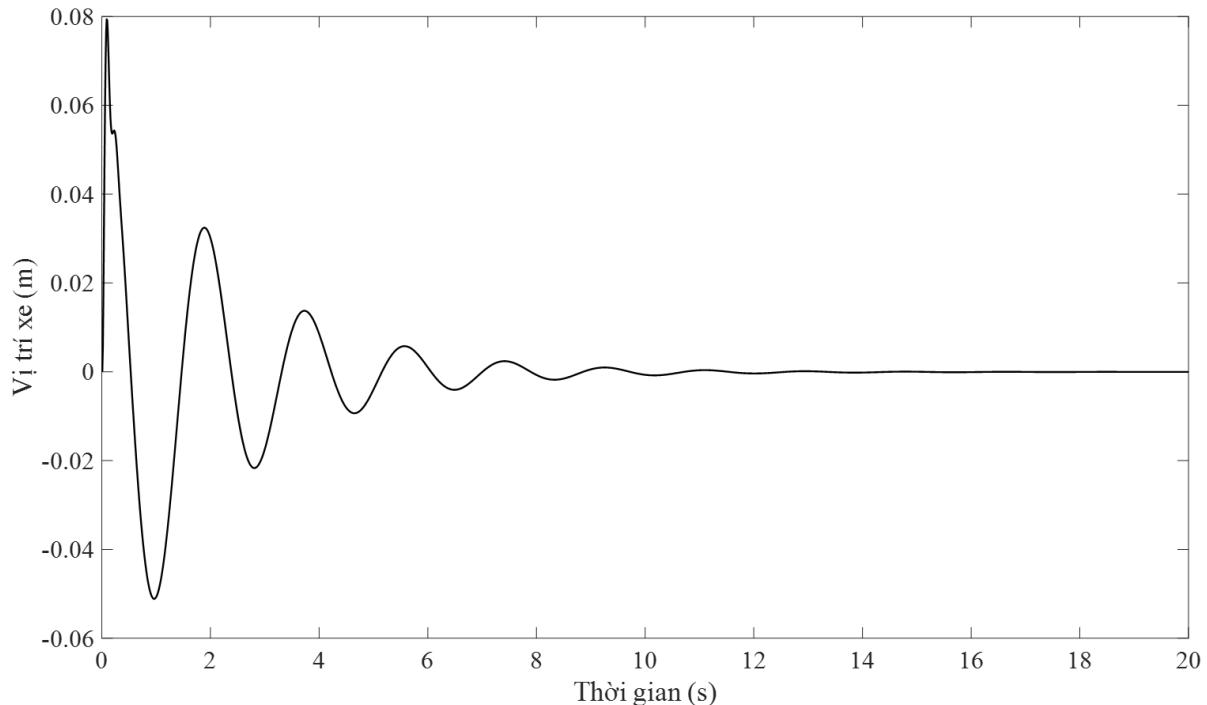
Khối này có tác dụng xác định giá trị của điểm đặt mong muốn cho con lắc khi cân bằng. Khi vị trí con lắc lớn hơn π thì khối này cho ra vị trí mong muốn là 2π . Còn khi vị trí con lắc nhỏ hơn hoặc bằng π thì khối này cho ra vị trí mong muốn là 0.

b) Kết quả

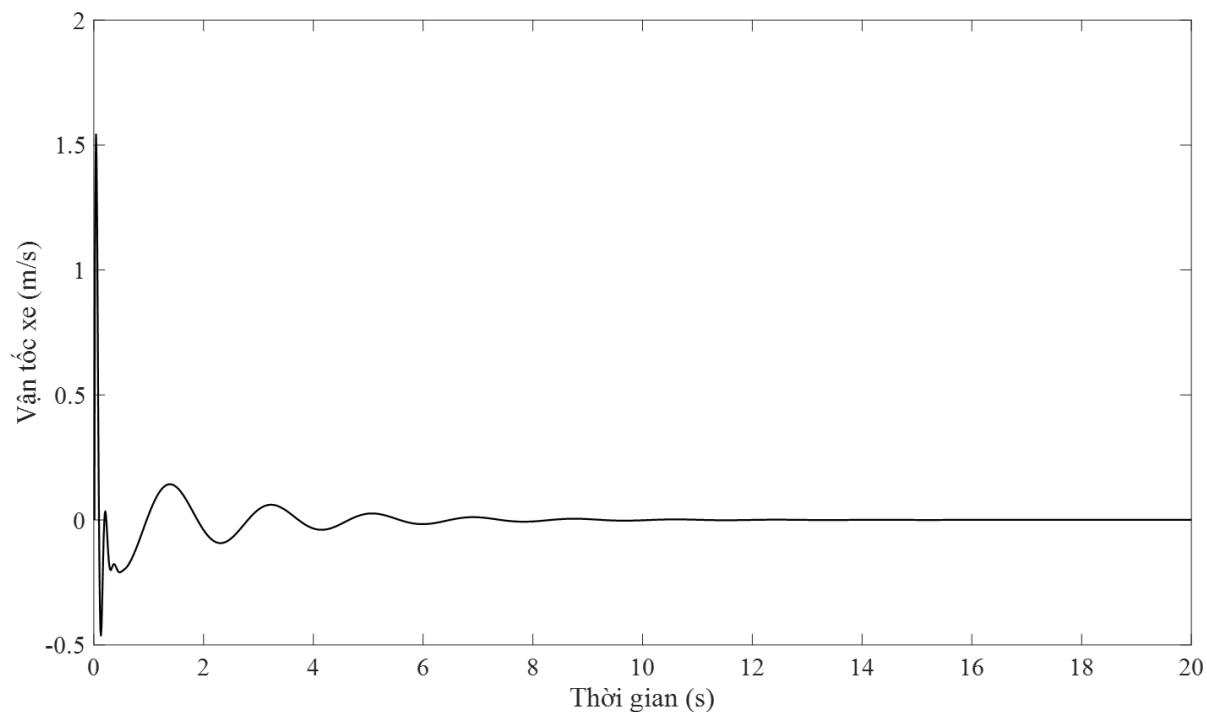
Bộ hệ số PID của hệ tìm được:

$$Kp1 = 0.6, Ki1 = 0.08, Kd1 = 0.12$$

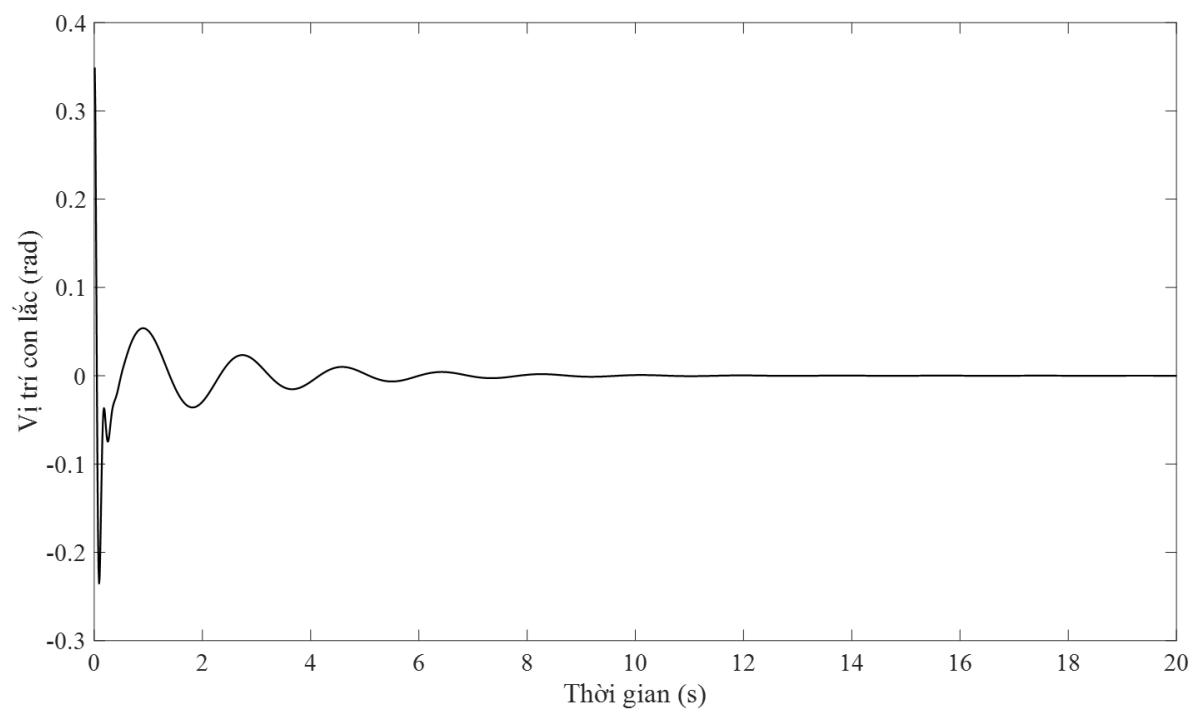
$$Kp2 = 20, Ki2 = 100, Kd2 = 0.1$$



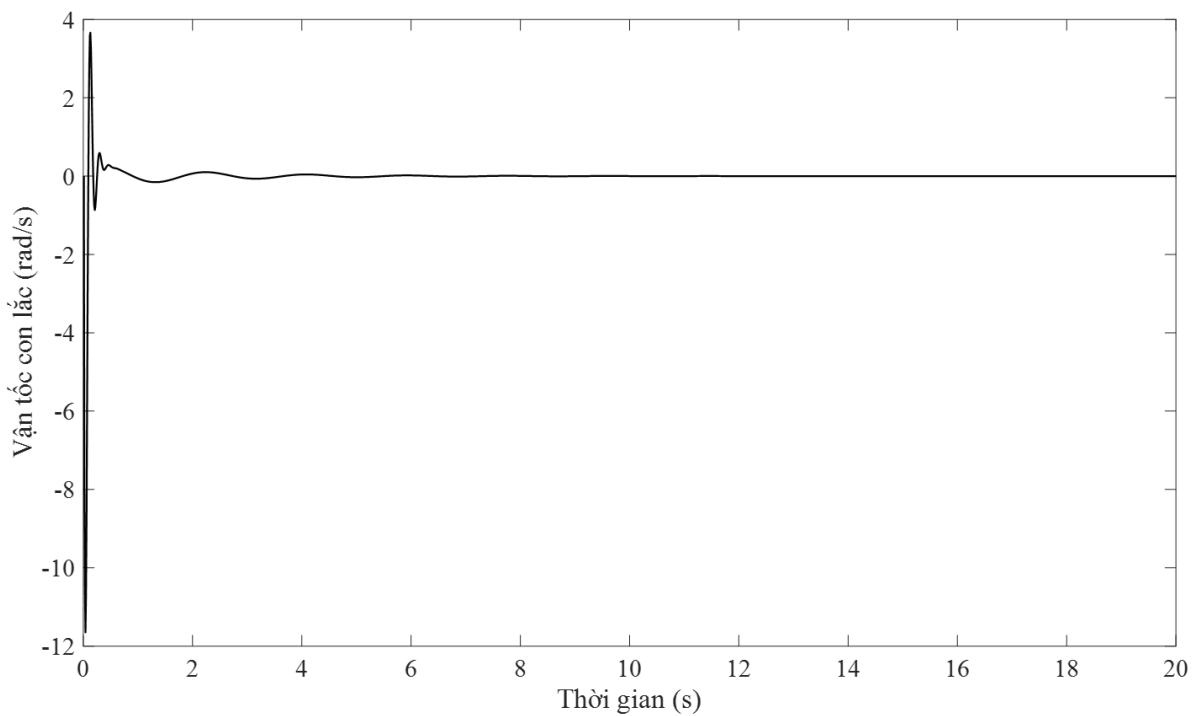
Hình 3.6 Đồ thị vị trí xe phương pháp điều khiển Cascade



Hình 3.7 Đồ thị vận tốc xe phương pháp điều khiển Cascade



Hình 3.8 Đồ thị vị trí con lắc phương pháp điều khiển Cascade



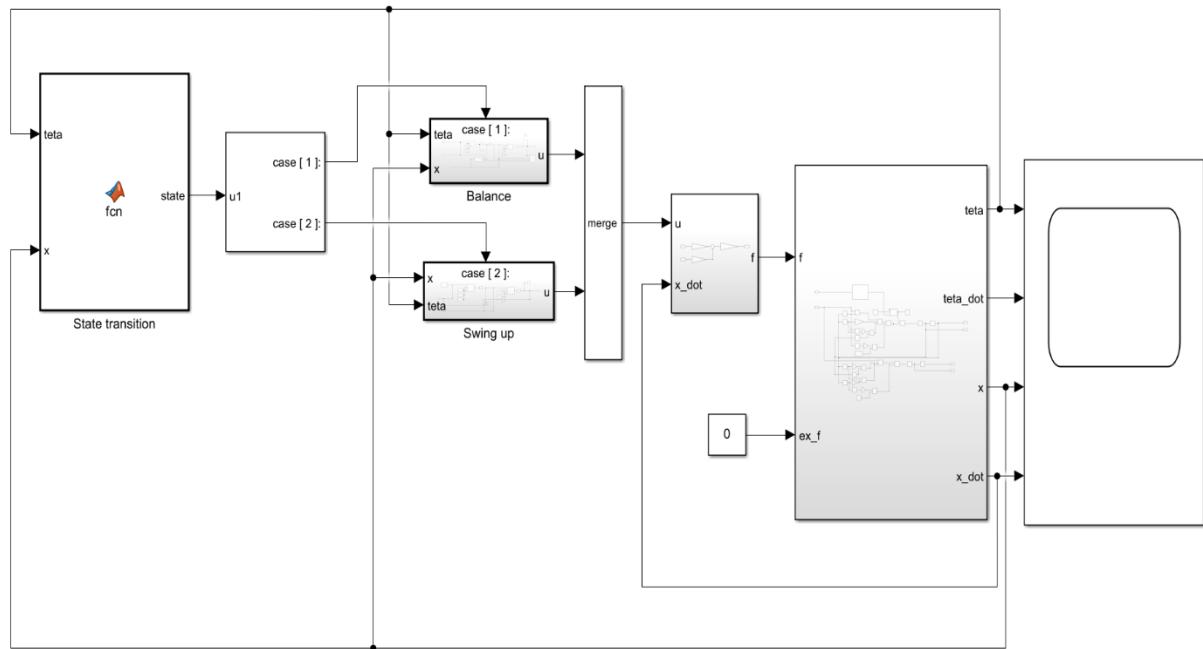
Hình 3.9 Đồ thị vận tốc góc con lắc phương pháp điều khiển Cascade

Với phương pháp điều khiển Cascade, hệ con lắc ngược được cân bằng thành công:

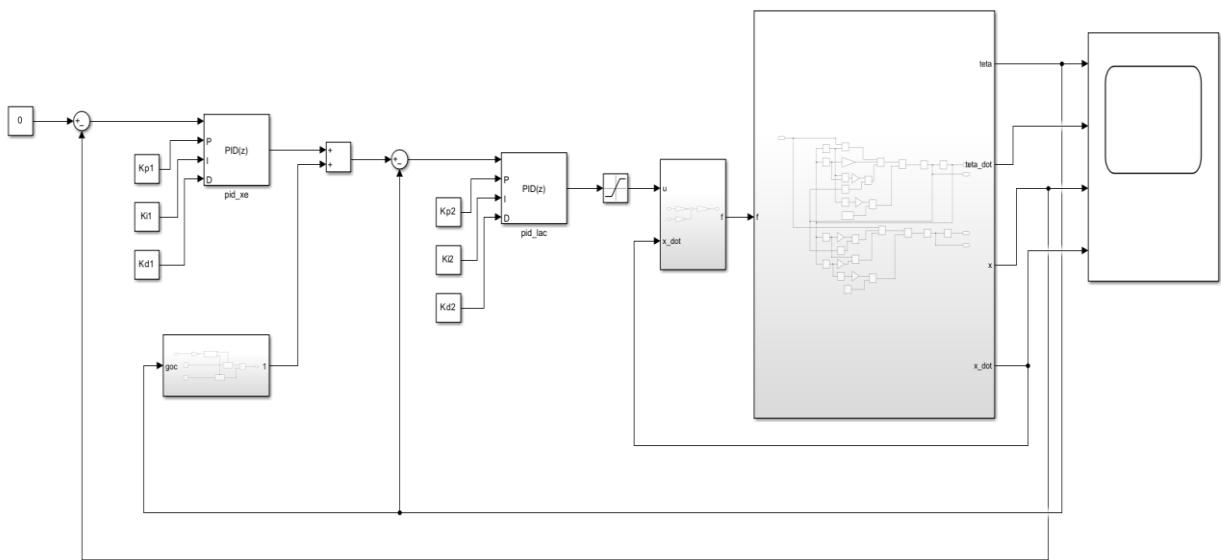
- Đồ thị vị trí xe phương pháp điều khiển Cascade bắt đầu từ vị trí 0 và sau khoảng 12 giây vị trí xe dần đạt được trạng thái cân bằng ở vị trí 0.
- Đồ thị vận tốc xe phương pháp điều khiển Cascade bắt đầu từ vị trí 0 và sau khoảng 12 giây vận tốc xe dần đạt được trạng thái cân bằng ở vị trí 0.
- Đồ thị vị trí con lắc phương pháp điều khiển Cascade bắt đầu từ vị trí $\frac{\pi}{9} \approx 0,35$ và sau khoảng 10 giây vị trí con lắc dần đạt được trạng thái cân bằng ở vị trí 0.
- Đồ thị vận tốc con lắc phương pháp điều khiển Cascade bắt đầu từ vị trí 0 và sau khoảng 10 giây vận tốc con lắc dần đạt được trạng thái cân bằng ở vị trí 0.
- Sau khoảng 12 giây, cả xe trượt và con lắc đều dần đạt được trạng thái cân bằng. Và vị trí xe, vận tốc xe, vị trí con lắc và vận tốc con lắc cân bằng ở vị trí 0.
- Biên độ dao động của vị trí xe trượt và vị trí của con lắc nằm trong giá trị cho phép, hệ thống đạt được trạng thái ổn định.

3.2.2 Phương pháp điều khiển Cascade kết hợp swing up

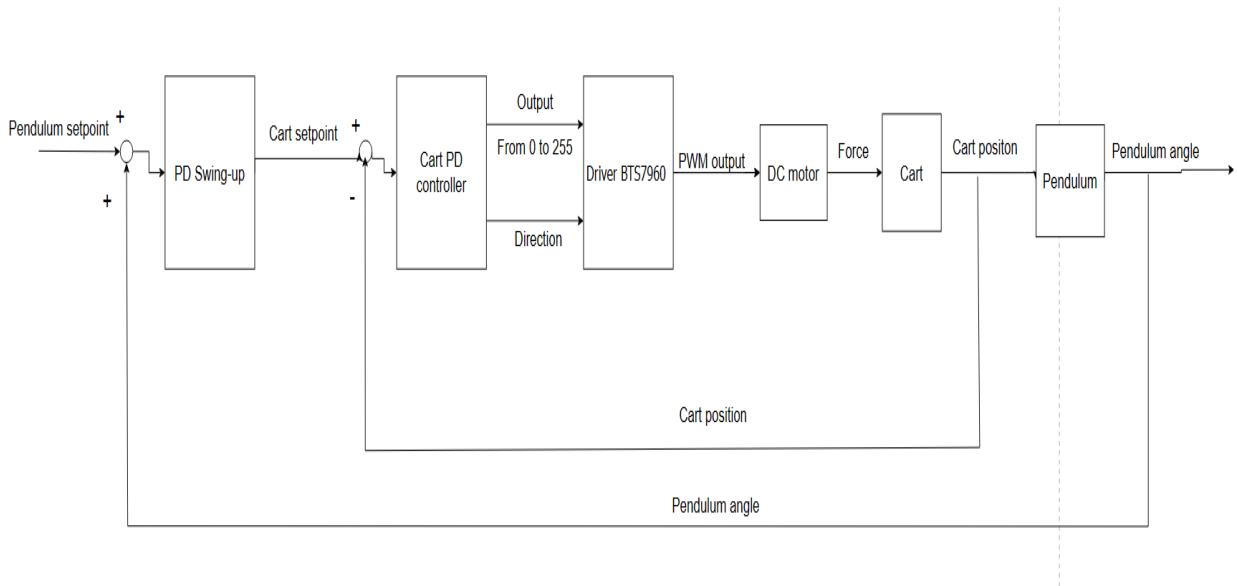
a) Xây dựng mô hình



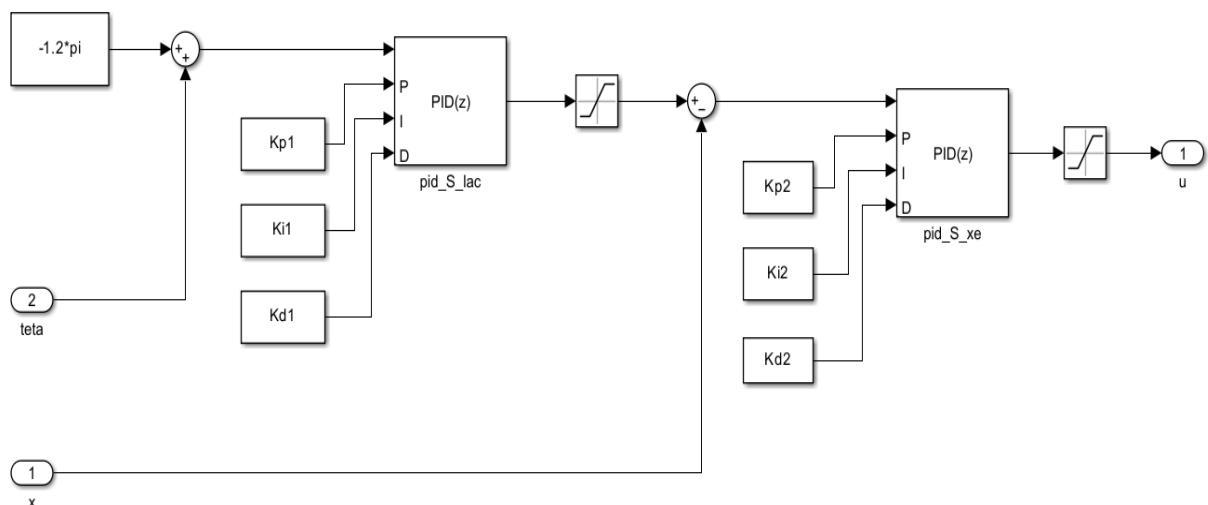
Hình 3.10 Mô phỏng con lắc ngược trên simulink bằng Cascade sử dụng swing up



Hình 3.11 Mô phỏng con lắc ngược trên simulink bằng Cascade



Hình 3.12 Sơ đồ khái thuật toán điều khiển swing up sử dụng bộ điều khiển PD

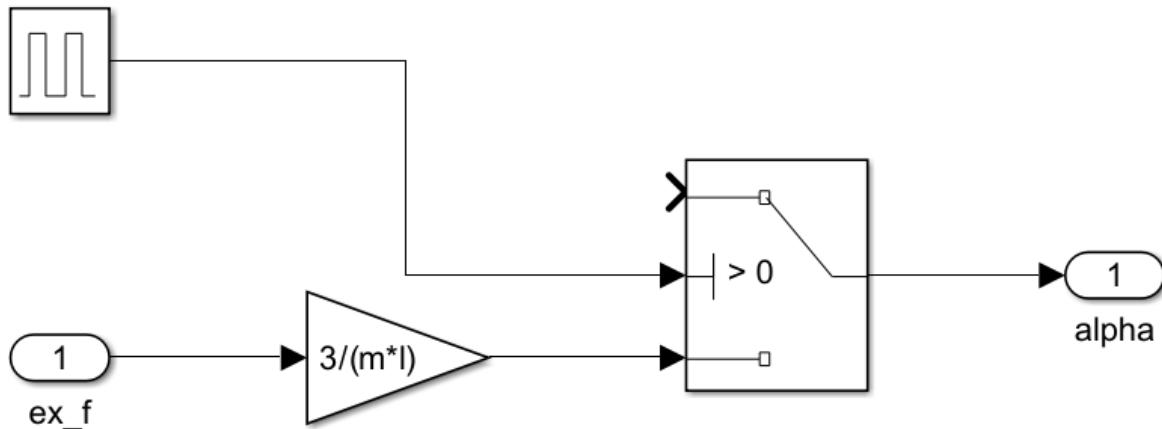


Hình 3.13 Mô phỏng bộ điều khiển swing up trên Simulink

Bộ điều khiển swing up dùng hai bộ PID để điều khiển vị trí xe trượt và vị trí góc của con lắc. Vòng ngoài là bộ PID điều khiển cho vị trí con lắc, vòng trong là bộ PID điều khiển cho vị trí xe trượt.

Trong chương trình mô phỏng này, bộ điều khiển vị trí con lắc ta dùng hồi tiếp dương và đặt vị trí mong muốn cho con lắc là -1.2π . Sau khi thông qua lần lượt hai bộ điều khiển PID, hệ thống tạo ra tín hiệu điện áp. Khối bão hòa có tác dụng giới hạn giá trị điện áp trong khoảng -24V đến 24V.

Đầu vào của khối điều kiện chuyển đổi là điện áp, thông qua khối chuyển đổi, chuyển từ điện áp sang lực tác động của động cơ, là đầu vào đưa vào khối mô hình toán hệ con lắc ngược phi tuyến. Và đầu ra của khối mô hình toán hệ con lắc ngược phi tuyến là bốn thành phần: vị trí xe, vận tốc xe, vị trí con lắc và vận tốc góc của con lắc.



Hình 3.14 Khối mô phỏng ngoại lực tác dụng vào con lắc

Khối này cộng thêm vào giá trị góc của hệ con lắc ngược một giá trị xác định trong thời gian ngắn để mô phỏng lại hành động tác dụng lực vào thanh con lắc khi ở trạng thái cân bằng.

Đầu vào của khối là một giá trị ngoại lực, giá trị này có thể thay đổi được thông qua việc điều chỉnh. Đầu ra của khối là một giá trị giá tốc góc sinh ra do ngoại lực tác động.

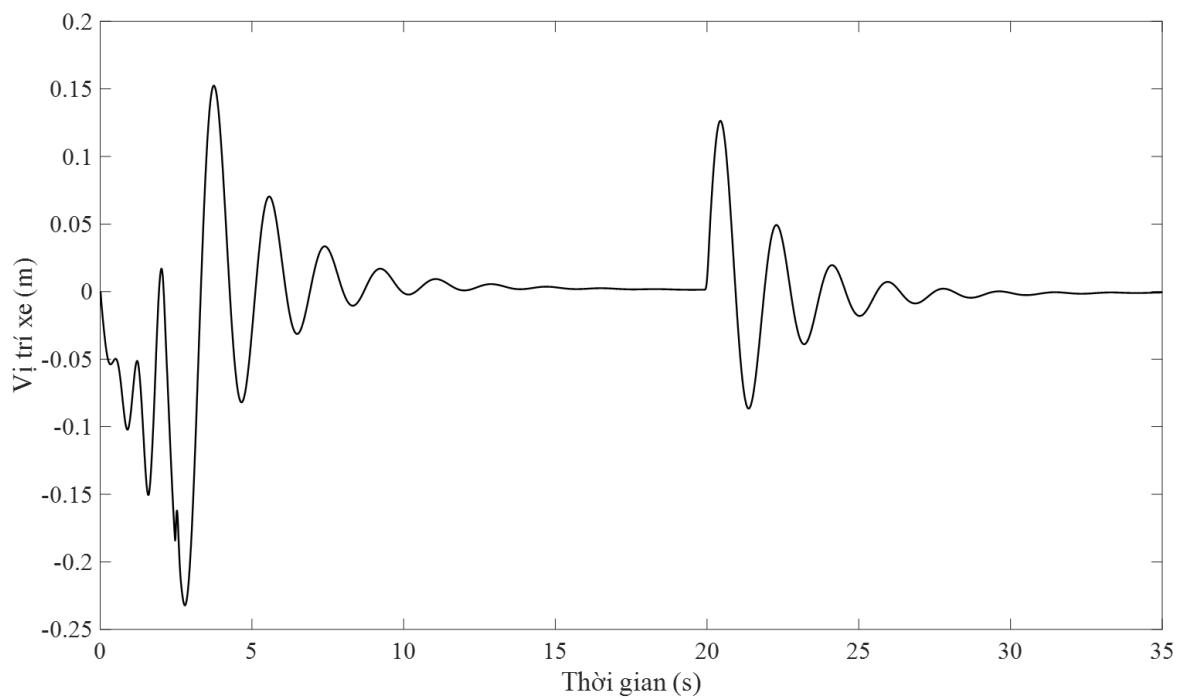
Khối tín hiệu xung cùng với khối công tắc tạo giả lập trạng thái thời gian tác dụng lực vào hệ con lắc ngược khi đã ở trạng thái cân bằng. Ở trong mô phỏng này, thời gian tác dụng ngoại lực xấp xỉ 20 giây một lần.

b) Kết quả

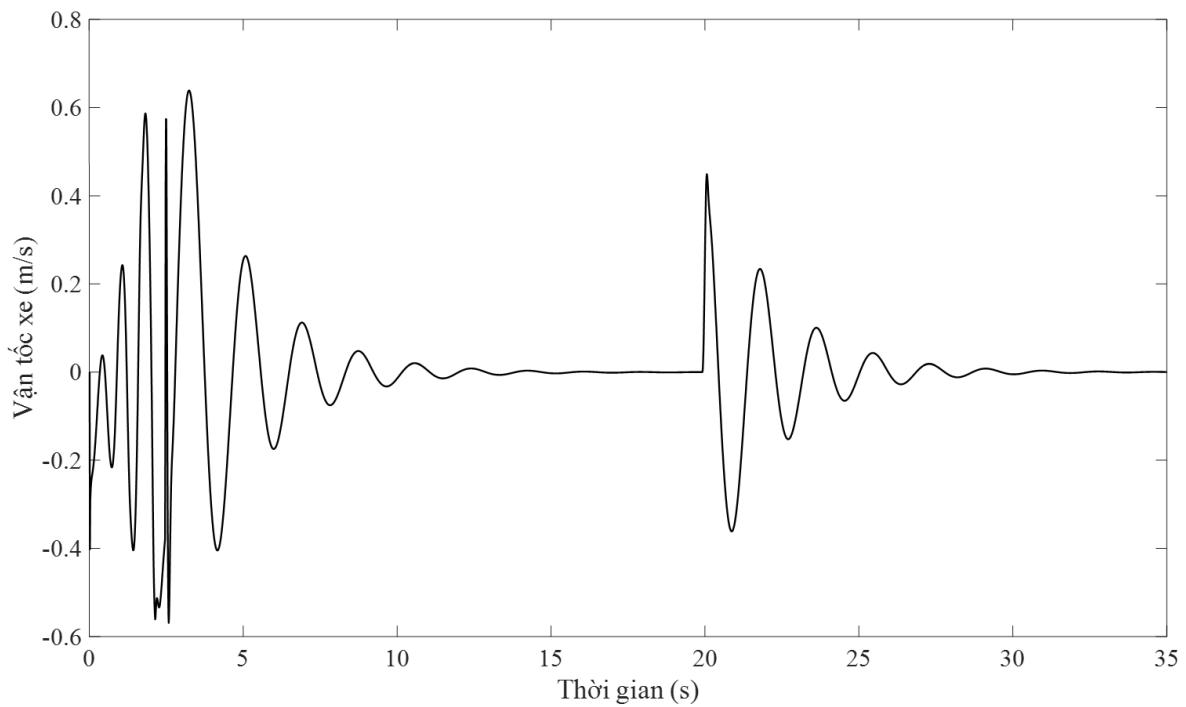
Bộ hệ số PID của hệ tìm được:

$$Kp1 = 0.15, Ki1 = 0, Kd1 = 0.01$$

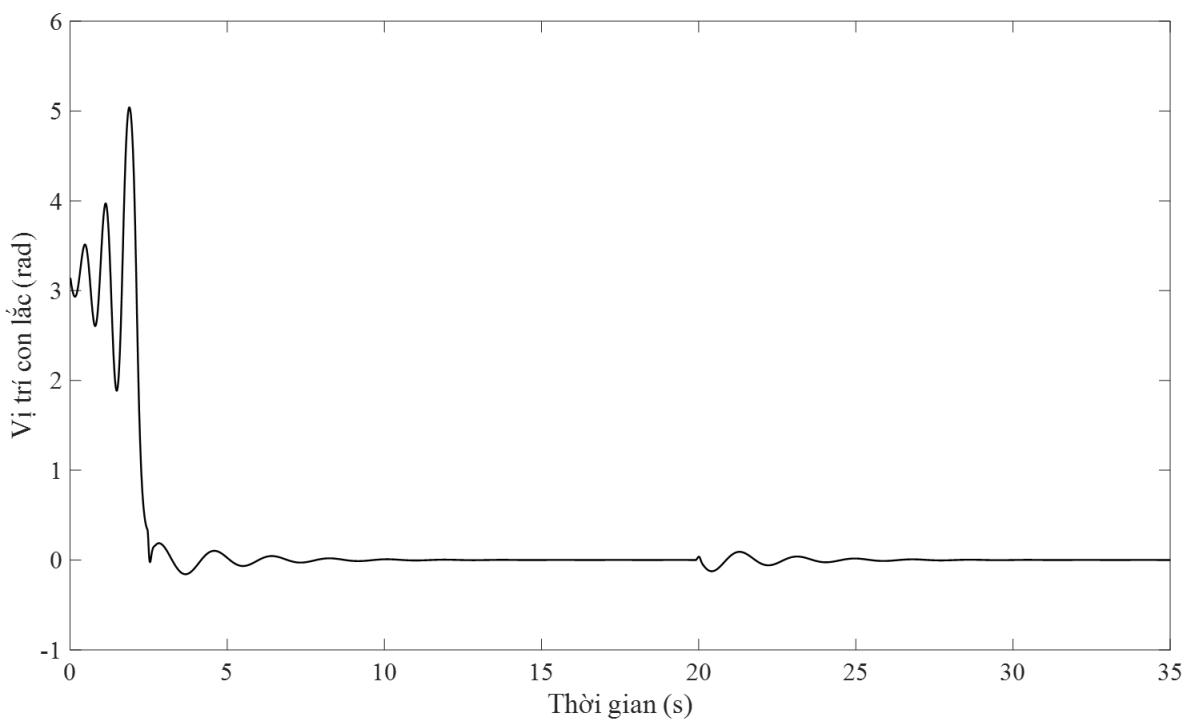
$$Kp2 = 10, Ki2 = 0, Kd2 = 0.01$$



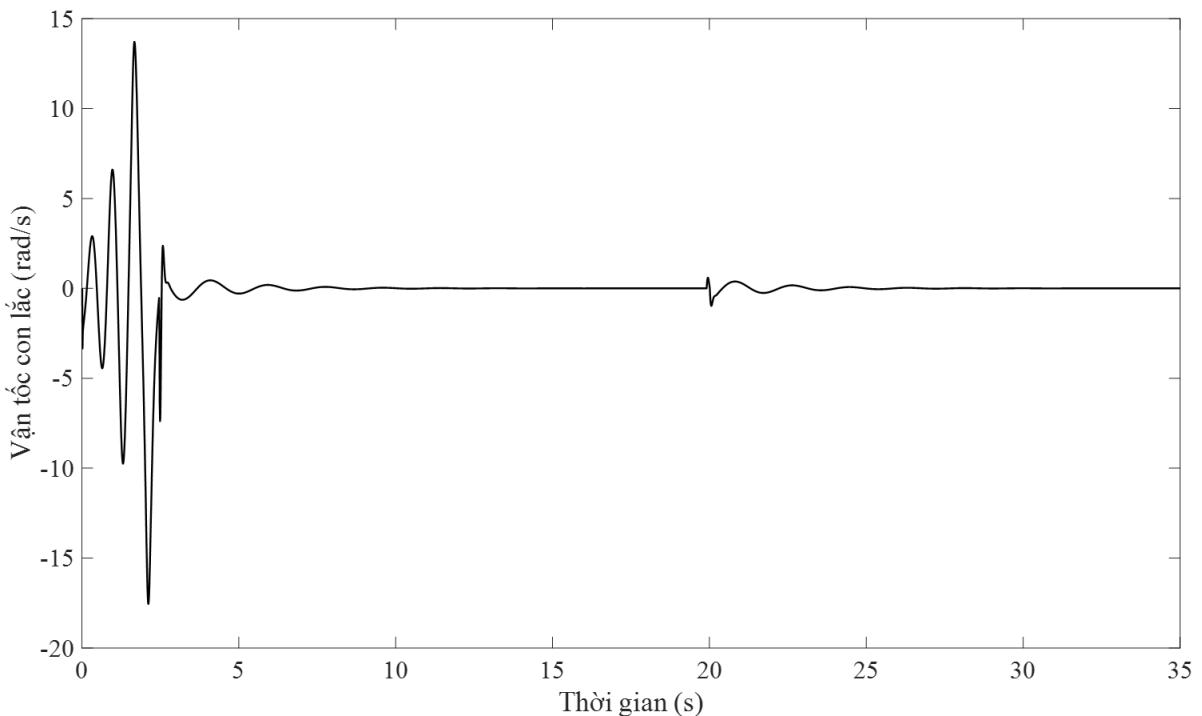
Hình 3.15 Đồ thị vị trí xe phương pháp điều khiển Cascade sử dụng swing-up



Hình 3.16 Đồ thị vận tốc xe phương pháp điều khiển Cascade sử dụng swing-up



Hình 3.17 Đồ thị vị trí con lắc phương pháp điều khiển Cascade sử dụng swing-up

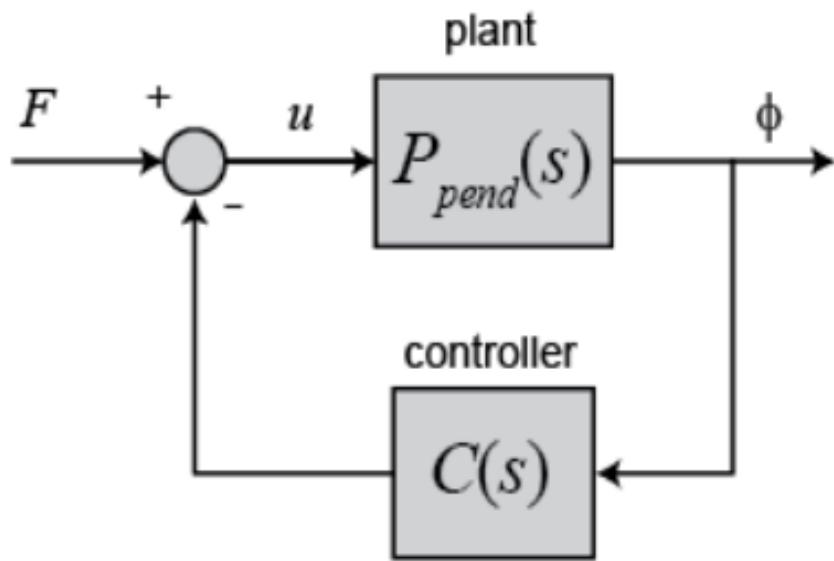


Hình 3.18 Đồ thị vận tốc con lắc phương pháp điều khiển Cascade sử dụng swing-up

Với phương pháp điều khiển Cascade sử dụng swing up, hệ con lắc ngược được cân bằng thành công:

- Đồ thị vị trí xe phương pháp điều khiển Cascade sử dụng swing up bắt đầu từ vị trí 0 và sau khoảng 15 giây vị trí xe dần đạt được trạng thái cân bằng ở vị trí 0. Và khi có ngoại lực tác động vào ở giây thứ 20, vị trí xe lại trở về trạng thái cân bằng sau 10 giây.
- Đồ thị vận tốc xe phương pháp điều khiển Cascade sử dụng swing up bắt đầu từ vị trí 0 và sau khoảng 15 giây vận tốc xe dần đạt được trạng thái cân bằng ở vị trí 0. Và khi có ngoại lực tác động vào ở giây thứ 20, vận tốc xe lại trở về trạng thái cân bằng sau 10 giây.
- Đồ thị vị trí con lắc phương pháp điều khiển Cascade sử dụng swing up bắt đầu từ vị trí $\pi \approx 3,14$ và sau khoảng 12 giây vị trí con lắc dần đạt được trạng thái cân bằng ở vị trí 0. Và khi có ngoại lực tác động vào ở giây thứ 20, vị trí con lắc lại trở về trạng thái cân bằng sau 7 giây.
- Đồ thị vận tốc góc con lắc phương pháp điều khiển Cascade sử dụng swing up bắt đầu từ vị trí 0 và sau khoảng 12 giây vận tốc con lắc dần đạt được trạng thái cân bằng ở vị trí 0. Và khi có ngoại lực tác động vào ở giây thứ 20, vận tốc con lắc lại trở về trạng thái cân bằng sau 7 giây.
- Sau khoảng 15 giây, cả xe trượt và con lắc đều dần đạt được trạng thái cân bằng. Và vị trí xe, vận tốc xe, vị trí con lắc và vận tốc góc con lắc cân bằng ở vị trí 0. Và khi có ngoại lực tác động vào hệ con lắc ngược ở giây thứ 20, sau 10 giây thì hệ lại trở về trạng thái cân bằng.
- Biên độ dao động của vị trí xe trượt và vị trí của con lắc nằm trong giá trị cho phép, hệ thống đạt được trạng thái ổn định.

3.3 Xây dựng chương trình mô phỏng bộ điều khiển dựa trên hàm truyền

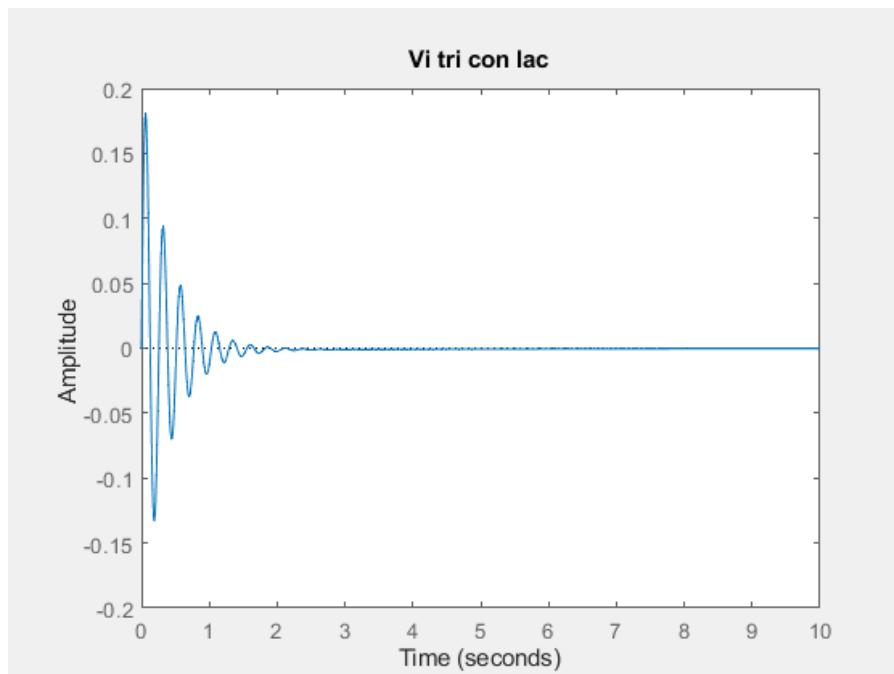


Hình 3.19 Sơ đồ mô phỏng hàm truyền con lắc

Với sơ đồ hệ thống trên ta thiết kế hàm truyền với đầu vào là lực F và đầu ra là góc của con lắc :

$$T(s) = \frac{\Phi(s)}{F(s)} = \frac{Conlac(s)}{1 + C(s)Conlac(s)}$$

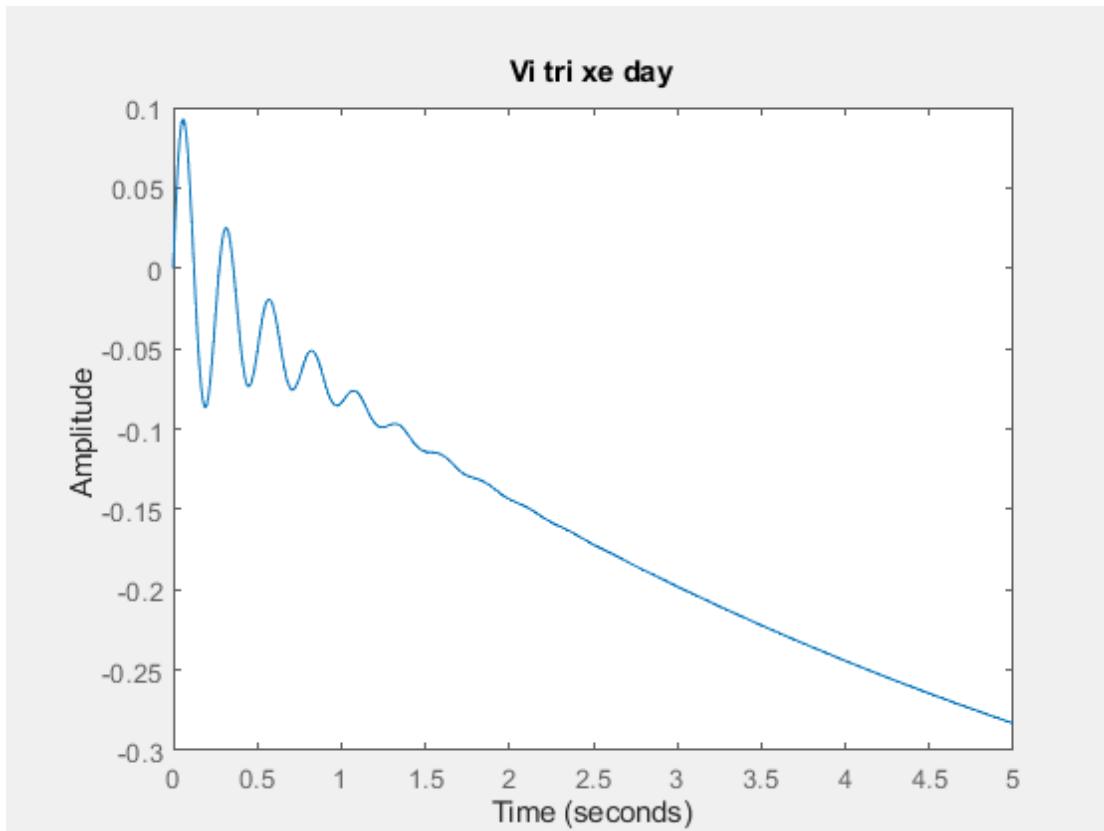
Kết quả



Hình 3.20 Đồ thị vị trí con lắc (đơn vị: rad)

Nhận xét:

- Về thời gian đáp ứng: đạt được thời gian phản hồi như mong muốn
- Về độ vượt lỗ : Không quá 0.2 Radian
- Con lắc ổn định sau khi cân bằng



Hình 3.21 Đồ thị vị trí xe (đơn vị: m)

Nhận xét:

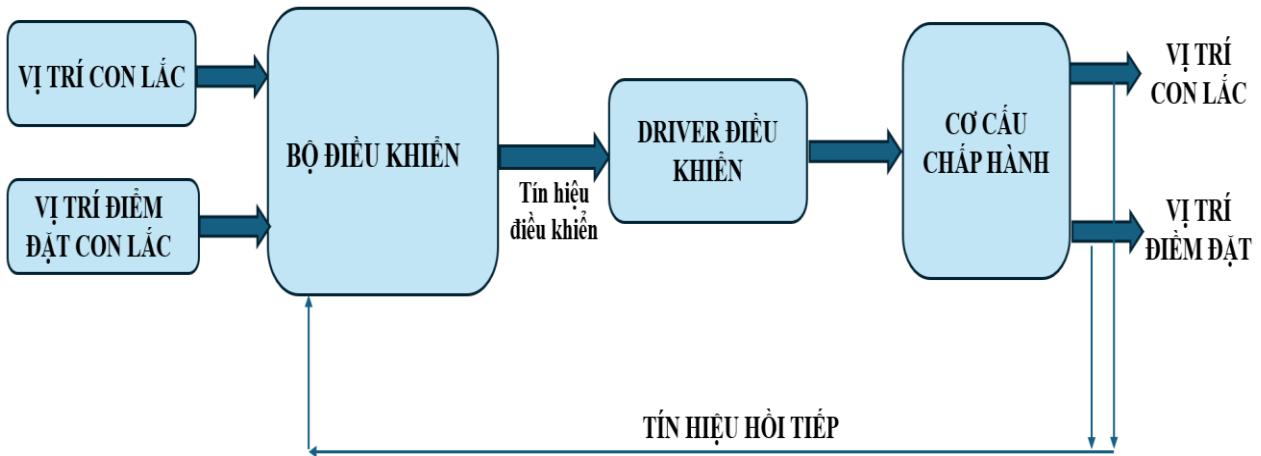
Xe đẩy chuyển động với vận tốc hầu như không đổi và vị trí có chiều hướng về phía âm. Mặc dù con lắc đã ổn định tốt với bộ điều khiển PID nhưng xe đẩy thì vẫn chưa hợp lý khi mang vào thực tế.

Source code

```
M = 0.3;
m = 0.1;
j = 0.006;
g = 9.8;
l = 0.17;
q = (M+m)*(j+m*l^2)-(m*l)^2;
s = tf('s');
Conlac = (m*l*s/q)/(s^3 - ((M + m)*m*g*l)*s/q);
Kp =120;
Ki = 20;
Kd = 1;
C = pid(Kp,Ki,Kd);
T = feedback(Conlac,C);
t=0:0.01:10;
impulse(T,t)
axis([0, 10, -0.2, 0.2]);
title({'Vi tri con lac'});
Xeday = (((j+m*l^2)/q)*s^2 - (m*g*l/q))/(s^4 - ((M +
m)*m*g*l)*s^2/q);
T2 = feedback(1,Conlac*C)*Xeday;
t = 0:0.01:5;
impulse(T2, t);
title({'Vi tri xe day'});
```

CHƯƠNG 4: THIẾT KẾ MÔ HÌNH CON LẮC NGƯỢC

4.1 Sơ đồ khái của một hệ con lắc ngược



Hình 4.1 Sơ đồ khái con lắc ngược

Đối với hệ con lắc ngược là hệ có một vật được cân bằng tại điểm có điểm cân bằng thấp hơn khỏi tâm. Con lắc sẽ được giữ thẳng đứng trên một chiếc xe (hệ pole and cart) hay giữ đứng trên một đĩa quay (hệ rotary). Tất cả các hệ con lắc ngược đều có 2 nhiệm vụ chính đó là cân bằng cân lắc và đưa vị trí cân bằng về điểm setpoint đặt trước.

Để làm được điều này, cần phải thiết kế cho nó một bộ điều khiển kín, có tín hiệu hồi tiếp trả về. Điều này giúp cho vi điều khiển biết được vị trí hiện tại của con lắc sau khi truyền tín hiệu điều khiển. Thiết bị thường dùng để nhận biết vị trí con lắc là encoder có độ phân giải phù hợp, nếu độ phân giải quá cao sẽ tạo gánh nặng cho vi điều khiển trong quá trình xử lý, ngược lại nếu độ phân giải quá thấp, hệ sẽ cân bằng không tốt. Đối với vị trí điểm cân bằng theo phương ngang, có thể dùng cảm biến siêu âm đặt ở một đâu để đo khoảng cách của vật (đối với hệ pole and cart), đối với hệ rotary thì là một bộ encoder để điều khiển bài toán vị trí quay của motor. Tuy nhiên trên thực tế, vẫn có đề tài sử dụng encoder đặt đồng trục với động cơ hoặc qua bộ truyền để xác định được vị trí của điểm cân bằng. Nếu không có bộ xác định vị trí, con lắc ngược sẽ xoay quanh trục liên tục (hệ rotary), hoặc có một thanh trượt dài vô tận để cân bằng con lắc (hệ pole and cart).

Tín hiệu sau khi xử lý qua bộ điều khiển sẽ được khuếch đại thông qua driver điều khiển. Driver điều khiển sẽ được lựa chọn phù hợp với cơ cấu chấp hành. Trong trường hợp này, cơ cấu chấp hành là động cơ DC, nên một số driver điều khiển cầu H như L298N

hoặc BTS7960 được sử dụng để khuếch đại tín hiệu điều khiển. Yêu cầu đặt ra cho vi điều khiển phải có tốc độ xử lý nhanh, tối thiểu 4 chân ngắn, và có bộ nhớ Flash đủ lớn để chứa chương trình điều khiển. Trên thực tế, hệ sẽ điều khiển trong miền thời gian rì rạc, nên việc lựa chọn sample time (T_s) điều khiển, phụ thuộc vào vi điều khiển rất nhiều. Bộ điều khiển cần có 2 thuật toán điều khiển cho hai trạng thái của con lắc là swing up và cân bằng.

4.2 Yêu cầu thiết kế con lắc ngược

- Khối lượng mô hình nhỏ hơn 5kg.
- Dễ tháo lắp, vận chuyển.
- Khối lượng con lắc nhẹ, giảm lực quán tính.
- Bộ điều khiển có khả năng chịu tải dòng điện tối thiểu 3A.
- Động cơ có momen lớn, quán tính trực quay thấp.
- Chiều dài thanh trượt vừa đủ, không quá ngắn, không quá dài
- Có cơ cấu dừng khẩn cấp hệ thống, và bảo vệ an toàn.

4.3 Thiết kế cơ khí

Hệ thống con lắc ngược dựa trên các thành phần như:

- Thanh ngang dài 1,0m có dạng ray khớp với ray trên bánh xe có rãnh chữ V. Điều này giúp xe tránh bị trượt khi di chuyển trên ray.
- Đường ray được khóa chặt vào đế bằng các giá đỡ góc đúc gắn vào hệ thống nhôm định hình giống như Máy in 3D.
- Con lắc là một thanh nhôm đồng nhất có chiều dài 20cm được gắn vuông góc với trục encoder để đo góc của con lắc.
- Encoder sử dụng cho con lắc là encoder có tốc độ 1000xung/vòng của thương hiệu Omron.
- Động cơ dùng để điều khiển hệ thống là DC Motor servo của Minitertia (70W). Khi làm có thể tham khảo các thông số qua datasheet của nhà sản xuất.
- Động cơ Minitertia có sẵn encoder gắn vào trục sau của động cơ. Tuy nhiên, encoder này có độ phân giải 2000 xung/vòng, quá lớn để xử lý cho vi điều khiển. Vì vậy, nhóm đã sử dụng một encoder khác có tốc độ 600 xung/vòng của thương hiệu Omron phù hợp với hệ thống.

- Ngoài ra còn thiết kế bộ cảng đai cho hệ thống. Điều này sẽ tránh được hiện tượng dây đai bị dãn và trượt theo thời gian.
- Hệ thống truyền động được thiết kế nhằm giảm trọng tâm của động cơ để tránh rung động.

Bảng 4.1 Các thông số cơ bản của mô hình

Thông số	Giá trị	Đơn vị
Trọng lượng của con lắc	0,1	kg
Trọng lượng của xe đẩy	0,3	kg
Chiều dài của con lắc	0,2	m
Chiều dài của thanh trượt	1	m
Trọng lượng của toàn bộ hệ thống	3kg	kg

Bảng 4.2 Ảnh hưởng của các thông số vật lý đến con lắc

Thông số thiết kế	Tác động đến điều khiển góc con lắc
Tỉ số truyền	Giảm tỷ số truyền sẽ làm tăng độ lợi của hàm truyền. Điều này có nghĩa là cần có điện áp điều khiển nhỏ hơn để hiệu chỉnh cho cùng một độ lệch góc.
Chiều dài con lắc	Việc tăng chiều dài của con lắc ngược sẽ kéo cực không ổn định lại gần gốc tọa độ. Vì vậy, một cây gậy dài hơn sẽ dễ giữ thẳng bằng hơn. Vì mức tăng giảm khi tăng chiều dài nên cần có điện áp điều khiển cao hơn để điều chỉnh cho cùng độ lệch góc.
Khối lượng xe	Nếu khối lượng của con lắc ngược tăng thì cực không ổn định sẽ di chuyển ra xa gốc tọa độ và độ lợi giảm. Điều này có nghĩa là con

	lắc ngược khó cân bằng hơn và cần có điện áp điều khiển cao hơn để điều chỉnh độ lệch góc.
Khối lượng con lắc	Việc giảm bán kính của bánh xe có tác dụng tương tự như việc giảm G. Do đó, nếu sử dụng bánh xe nhỏ hơn thì cần có điện áp điều khiển nhỏ hơn để hiệu chỉnh cho cùng một góc lệch.
Bán kính bánh xe	Việc giảm bán kính của bánh xe có tác dụng tương tự như việc giảm G. Do đó, nếu sử dụng bánh xe nhỏ hơn thì cần có điện áp điều khiển nhỏ hơn để hiệu chỉnh cho cùng một góc lệch.

Các yếu tố cần quan tâm khi chọn động cơ DC cho hệ thống con lắc ngược

- Để có thể cân bằng con lắc ngược, động cơ DC phải đáp ứng các tiêu chí nhất định về mô men xoắn và tốc độ đủ lớn, quán tính trực quay thấp. Hơn nữa, nó phải phù hợp với hệ thống có tính đến điện áp và dòng điện yêu cầu.
- Để có thể cân bằng thanh, động cơ DC phải tạo ra mô-men xoắn đủ lớn. Theo mô phỏng do Van Hawarden thực hiện, lực F tối đa cần thiết là khoảng 1 N.cm. Tỷ số truyền của bánh răng G là 1:19 và bán kính bánh xe rw khoảng hai cm. Vậy lúc này mô men xoắn cực đại cần thiết T là: $T = G \cdot r_w \cdot F$
- Theo mô phỏng tương tự do Van Hawarden thực hiện, vận tốc yêu cầu v là khoảng 1 m/s. Điều này có nghĩa là tốc độ góc tối đa cần thiết của trục động cơ là 950 rad/s hoặc 9000 vòng/phút với bộ mã hóa 1000 xung/vòng quay.

$$\omega = \frac{v}{G \cdot r_w}$$

- Tần số càng cao động cơ hoạt động càng êm, tuy nhiên momen tạo ra sẽ giảm

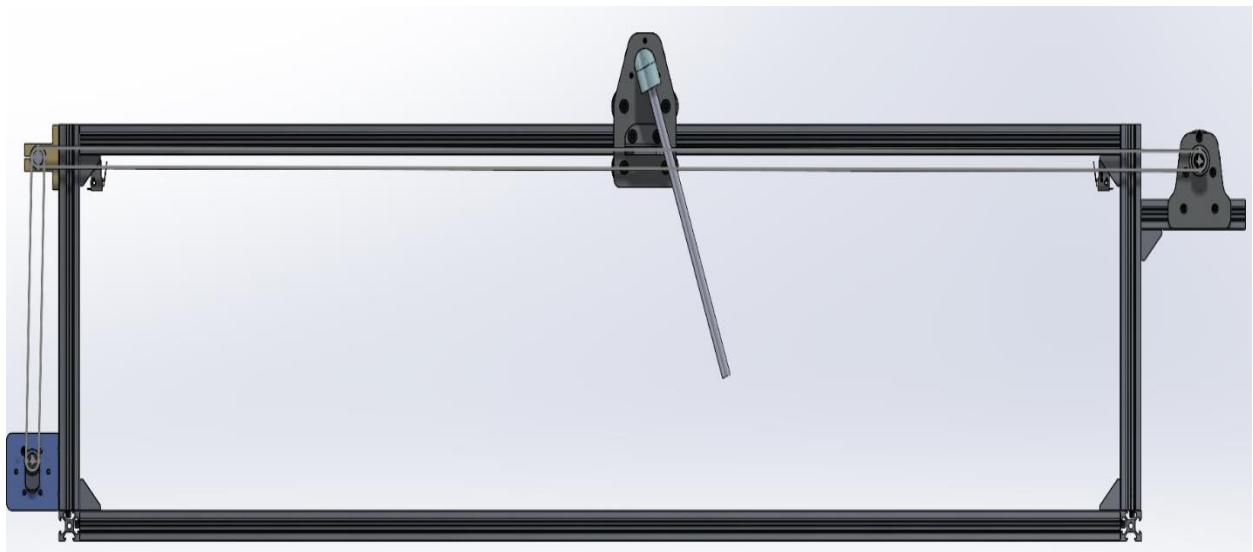
Động cơ DC

Trong project này nhóm sử dụng động cơ Minertia RO2SA với datasheet được cung cấp sẵn từ nhà sản xuất

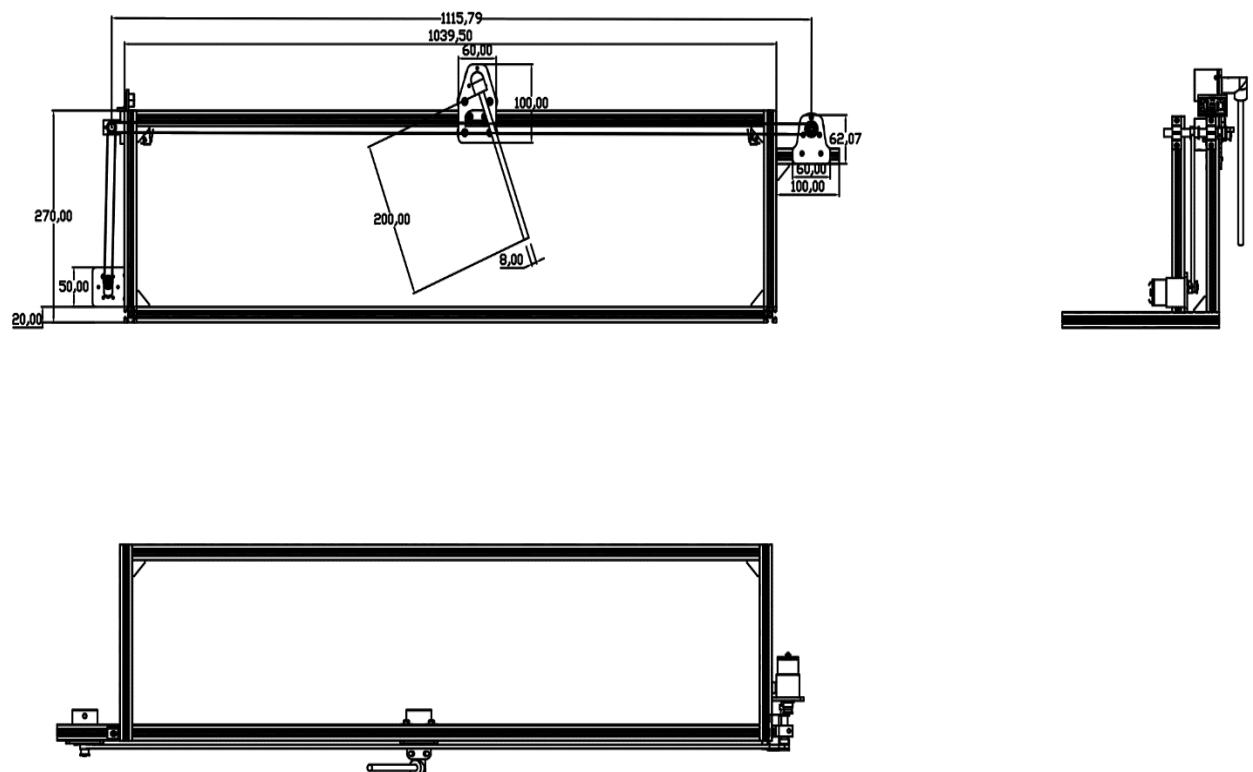
Motor Type	R01SA	R02SA	R02MA	R04SA	R04MA	R08SA	R08MB	R40SA	R40MA
Peak rated torque oz·in	75	150	250	300	500	750	1250	1190	2140
Rated torque oz·in	15	30	50	60	100	150	250	450	850
Torque constant oz·in/A	7.58	8.12	11.5	11.9	16.8	20.2	32.7	51.6	69.0
Armature winding resistance Ohms (at 25°C)	2.75	1.12	0.94	0.59	0.41	0.41	0.49	0.57	0.34
Armature inductance mH	1.2	0.9	0.9	0.7	0.6	1.2	2.0	4.0	3.1
Peak current amps	10	18.8	22.1	25.5	30.1	37.5	38.5	27.6	36.3
Voltage constant volts/1000 rpm	5.6	6.0	8.5	8.8	12.4	14.9	24.2	38.2	38.2
Viscous damping coefficient oz·in/1000 rpm	0.42	0.82	1.5	2.2	3.7	3.5	2.5	4.03	7.23
Friction torque oz·in	0.9	2.4	3.5	3.6	5.1	6.4	8.3	15.5	19.4
Inertia oz·in·sec ² × 10 ⁻³	0.652	2.22	3.96	13.6	23.7	72.2	118	366	625
Mechanical time constant millisecond	4.4	5.4	4	8	4.9	10	7.7	11	6.1
Electrical time constant millisecond	0.44	0.8	0.96	1.2	1.5	2.9	4.1	7.1	9.4
Power rate kW/sec	2.43	2.86	4.45	1.87	2.97	2.2	3.73	3.94	8.16
Torque inertia ratio rad/sec ²	23000	13500	12600	4410	4210	2080	2120	1230	1360
Thermal resistance deg C/watt	3.5	2.33	1.86	1.86	1.55	1.25	1.0	0.8	0.6
Max temperature rise deg	100	100	100	100	100	100	100	155	155
Rated speed rpm	3000	3000	3000	3000	3000	3000	3000	2500	2200
Max safe operating speed rpm	4500	4000	4000	4000	4000	4000	4000	4000	3500
Max no load speed rpm	5000	5000	5000	5000	5000	5000	5000	4000	3500
Cooling required cfm, in H ₂ O								Totally-enclosed self-cooled	
Weight lb	0.89	1.77	2.43	3.09	4.86	8.36	11.5	18.1	24.3

Hình 4.2 Thông số kỹ thuật động cơ Minertia RO2SA

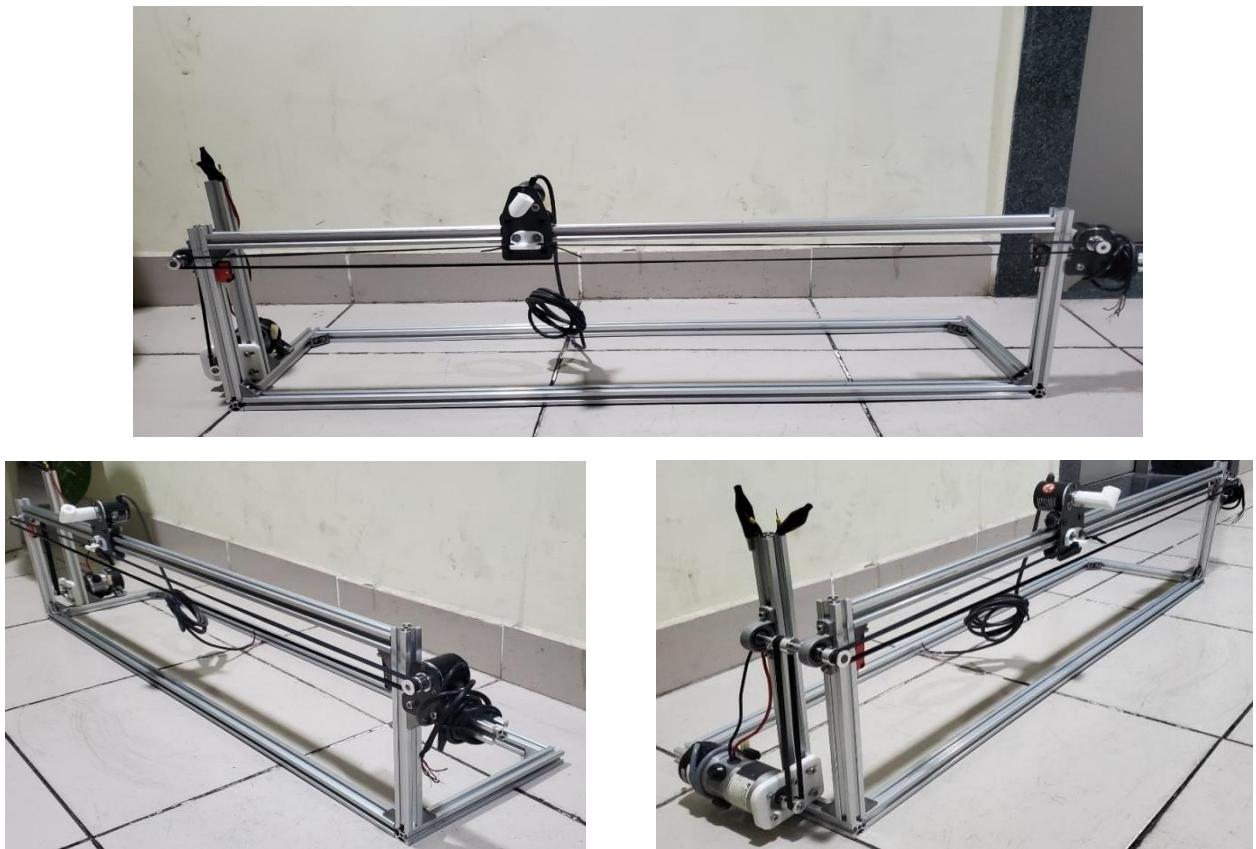
Mô hình 3D trên SolidWorks



Hình 4.3 Thiết kế mô hình trên Solidworks



Hình 4.4 Thông số kích thước của mô hình (đơn vị: mm)



Hình 4.5 Mô hình thực tế

4.4 Thiết kế phần điện

4.4.1 Vi điều khiển

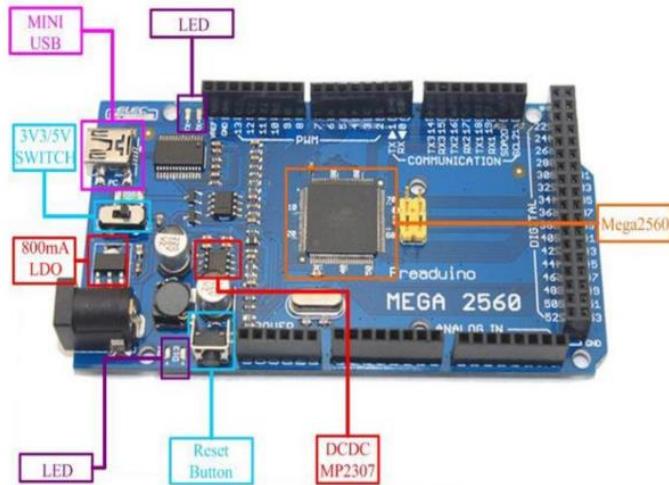
Yêu cầu về vi điều khiển:

- Tốc độ xử lý đủ đáp ứng (tối thiểu 10ms)
- Giá thành thấp
- Có ít nhất 4 chân ngắt ngoài để đọc 2 bộ encoder
- Có khả năng kết nối với phần mềm Simulink
- Có môi trường lập trình sẵn như Arduino IDE
- Tiêu thụ năng lượng thấp

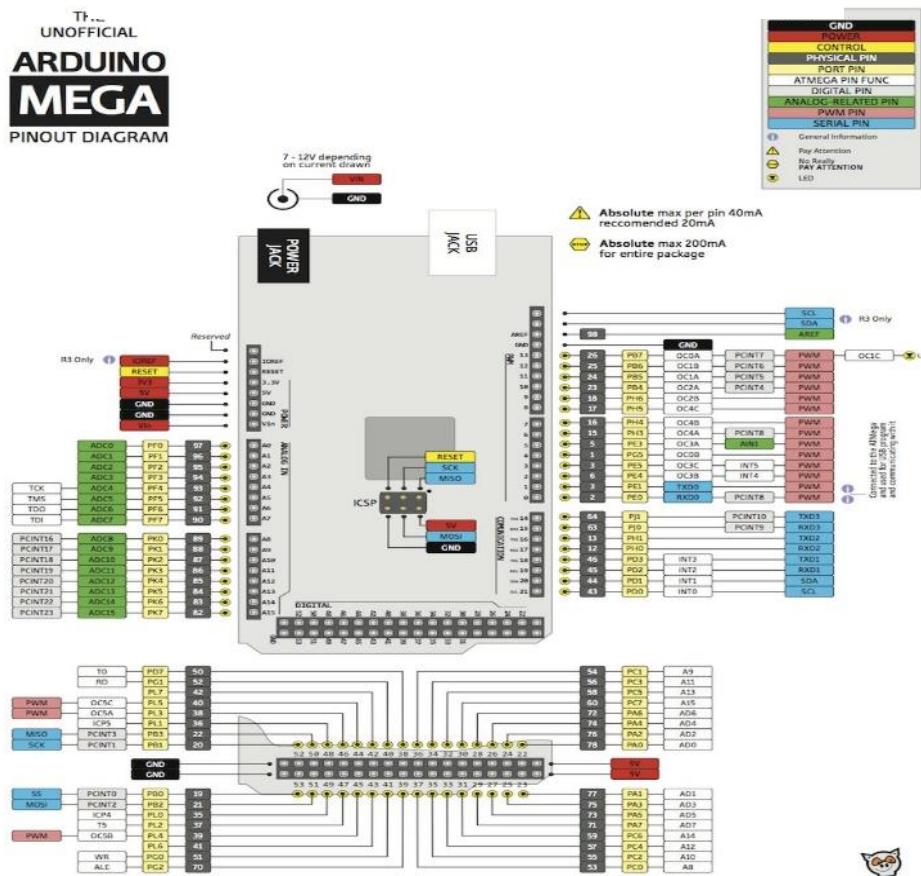
Trong dự án này, bo mạch Arduino Mega được sử dụng vì nó có cổng kết nối với máy tính thông qua cổng USB. Ngoài ra, nó còn có Arduino Mega có 6 kênh ngắt ngoài với chân Digital pin 2, Digital pin 3, Digital pin 21, Digital pin 20, Digital pin 19 và Digital pin 18, sẽ phù hợp cho 2 encoder từ vị trí xe đẩy và góc con lắc. Nó sẽ được cấp nguồn bằng điện áp ngoài 5V thay vì sử dụng điện áp trực tiếp từ máy tính.

Arduino là một nền tảng nguồn mở. Phần cứng có chức năng xây dựng các ứng dụng tương tác với nhau hoặc với môi trường thuận tiện hơn thông qua bảng mạch mã nguồn mở được thiết kế trên nền tảng vi xử lý AVR Atmel 8bit hoặc ARM Atmel 32-bit.

Arduino Mega 2560 là phiên bản nâng cấp của Arduino Mega. Điểm khác biệt và cải tiến lớn nhất của dòng Arduino này chính là chip nhân.



Hình 4.6 Arduino Mega 2560 [32]



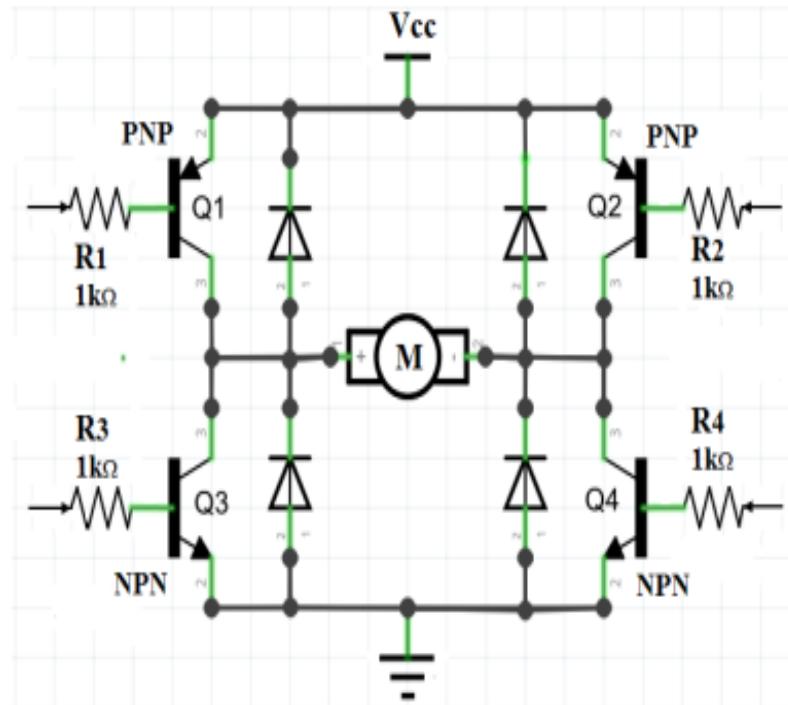
Hình 4.7 Sơ đồ chân kết nối trên Arduino Mega 2560 [15]

Bảng 4.3 Thông số kỹ thuật cơ bản của Arduino Mega 2560

Chip xử lí	Arduino Mega 2560
Điện áp hoạt động	5V
Điện áp đầu vào (Khuyến nghị)	7V-15V
Điện áp đầu vào (Giới hạn)	6V-20V
Cường độ dòng điện trên mỗi pin 3.3V	50mA
Cường độ dòng điện trên mỗi chân I/O	20mA
Tốc độ xử lí	16MHz
Số chân digital	54 (15 chân PWM)
Số chân analog	16
Thạch anh	16MHz
SRAM	8KB
EPPROM	4KB
Bộ nhớ Flash	256KB
Số kêtUART	4

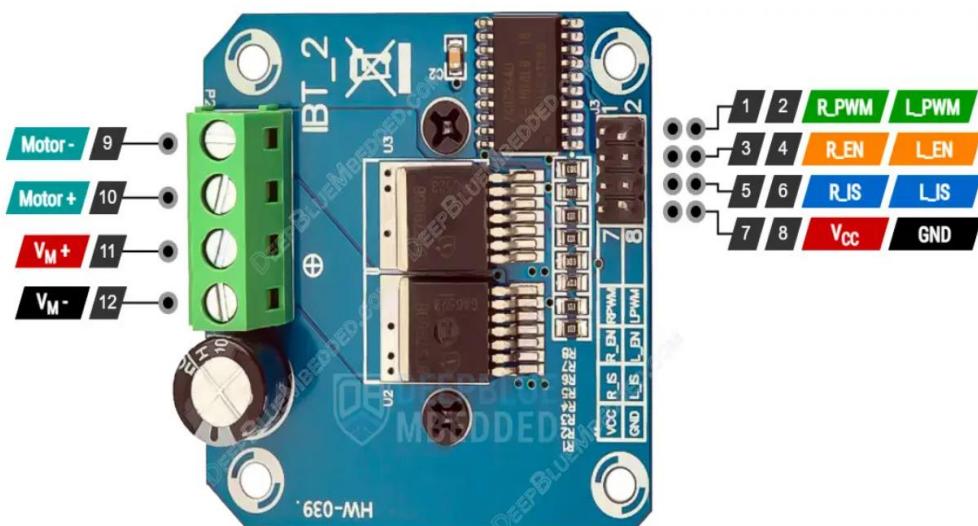
4.4.2 Driver cầu H

Việc kiểm soát điện áp cung cấp cho động cơ được thực hiện bằng cách tạo ra các xung PWM từ MCU. Bằng cách thay đổi độ rộng xung PWM, giá trị điện áp trung bình sẽ được áp dụng; đối tượng điều khiển sẽ thay đổi tương ứng. Tín hiệu xung PWM qua mạch điện điều khiển động cơ là mạch cầu H. Nguyên lý của mạch cầu H được mô tả trên hình 2.6, Giả sử nếu Q3 và Q2 đóng (Q1 và Q4 tắt) thì động cơ quay theo chiều thuận còn nếu Q1 và Q4 đóng (Q2 và Q3 tắt), động cơ sẽ quay theo hướng ngược lại.



Hình 4.8 Sơ đồ nguyên lý mạch cầu H [16]

Driver BTS7960 sẽ được sử dụng cho dự án vì đáp ứng đầy đủ các tiêu chí như giá cả hợp lý, phù hợp với hệ thống hiện tại, hiệu suất cao,... Driver mạch cầu H có thể chịu được dòng điện lên tới 43A và có khả năng điều chỉnh xung. Có 2 đầu vào R_PWM và L_PWM dùng để điều khiển tốc độ động cơ, có thể lựa chọn được hướng quay của động cơ.



Hình 4.9 Sơ đồ chân của mạch cầu H BTS7960 [34]

Bảng 4.4 Bảng thông số kỹ thuật cơ bản của BTS7960

Điện áp cung cấp động cơ (V_M)	6V – 27V
Điện áp cung cấp logic (V_{CC})	5V
Điện áp đầu vào logic	3,3V/ 5V
Dòng điện đầu ra (I_O)	43A
Dòng điện cực đại đầu ra trên mỗi kênh (I_{OMax})	60A

4.4.3 Mạch giảm áp

LM2596

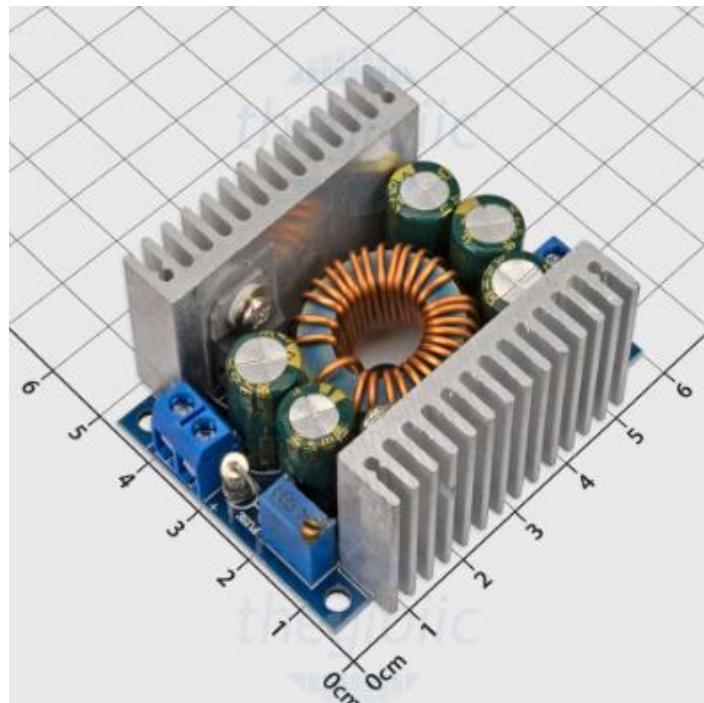
Mạch giảm áp DC LM2596 3A nhỏ gọn có khả năng giảm áp từ 30V xuống 1.5V mà vẫn đạt hiệu suất cao (92%). Thích hợp cho các ứng dụng chia nguồn, hạ áp, cấp cho các thiết bị như camera, motor, robot, Trong project này LM2596 đóng vai trò cấp nguồn tín hiệu cho hệ thống như module relay và Arduino Mega để nhận biết công tắc hành trình có đóng ngắt hay không.



Hình 4.10 Mạch giảm áp LM2596

Module giảm áp 10A

Module nguồn không sử dụng cách ly có kích thước nhỏ gọn, có khả năng giảm áp từ 30V xuống 1,25V, điều chỉnh được dòng ra cực đại là 10A với hiệu suất 92%. Dùng để cấp nguồn với điện áp đầu ra là 20V cho cầu H và động cơ hoạt động trong dãy điện áp từ 0V đến 20V



Hình 4.11 Module giảm áp chịu dòng cao [18]

Bảng 4.5 Thông số cơ bản của module giảm áp 10A

Nguồn đầu vào	4,5V-30V
Nguồn đầu ra	1,25V-30V
Công suất	100W
Dòng ra cực đại	10A

4.4.4 Module Relay

Module Relay 5V cấu tạo đơn giản, không có opto cách ly, kích thước khá nhỏ gọn so với các loại khác, gồm 1 rơ le hoạt động tại điện áp 5VDC, kích trạng thái đóng mở ở mức cao. Được sử dụng để đóng ngắt tải với công suất phù hợp như động cơ DC khi hệ con lắc trở nên hỗn loạn và bật công tắc hành trình trong quá trình điều khiển để đảm bảo hệ thống an toàn.



Hình 4.12 Module relay 5V [19]

4.4.5 Encoder

Encoder là một thiết bị cơ điện chuyển đổi chuyển động của trục hoặc vị trí góc thành kỹ thuật số hoặc tín hiệu đầu ra analog. Encoder được dùng để phát hiện hướng di chuyển, vị trí, tốc độ... của động cơ bằng cách đếm số vòng quay được của trục. Cấu tạo của encoder gồm các phần chính: đĩa quang tròn có rãnh nhỏ quay quanh trục, bộ cảm biến thu, nguồn sáng.

Bảng 4.6 Thông số kỹ thuật cơ bản của encoder 600 xung và 1000 xung

Encoder (hãng OMRON)	600 xung	1000 xung
Điện áp sử dụng	5V-24V	5V-24V
Dòng tiêu thụ (cực đại)	80mA	80mA
Số kênh xung	3 kênh xung riêng biệt A, B	3 kênh xung riêng biệt A, B, Z
Tần số đáp ứng tối đa	100KHz	100KHz
Đường kính trục	6mm	6mm
Đường kính thân	40mm	40mm



Hình 4.13 Encoder omron [20]

4.4.6 Công tắc

- **Công tắc hành trình**

Công tắc hành trình là dạng công tắc dùng để giới hạn hành trình của các bộ phận chuyển động nào đó trong một cơ cấu hay một hệ thống. Nó có cấu tạo như công tắc điện bình thường, vẫn có chức năng đóng và mở nhưng có thêm cần tác động để cho các bộ phận chuyển động tác động vào làm thay đổi trạng thái của tiếp điểm bên trong nó.



Hình 4.14 Công tắc hành trình Omron [21]

- **Công tắc shutdown**

Nút dừng khẩn cấp là công tắc được thiết kế để dừng ngay lập tức các thiết bị hoặc hệ thống trong trường hợp xảy ra những tình huống nguy hiểm. Khi nút dừng khẩn cấp được nhấn, tín hiệu được gửi đi và ngay lập tức các quy trình làm việc sẽ được dừng lại. Nút dừng khẩn cấp thường được sử dụng trong các hệ thống điều khiển với mục đích an toàn.



Hình 4.15 Công tắc dừng khẩn cấp [22]

4.4.7 Nguồn điện

Nguồn xung là bộ nguồn có tác dụng biến đổi từ nguồn điện xoay chiều sang nguồn điện một chiều bằng chế độ dao động xung tạo bằng mạch điện tử kết hợp với một biến áp xung. Dùng để cấp nguồn cho toàn bộ hệ thống



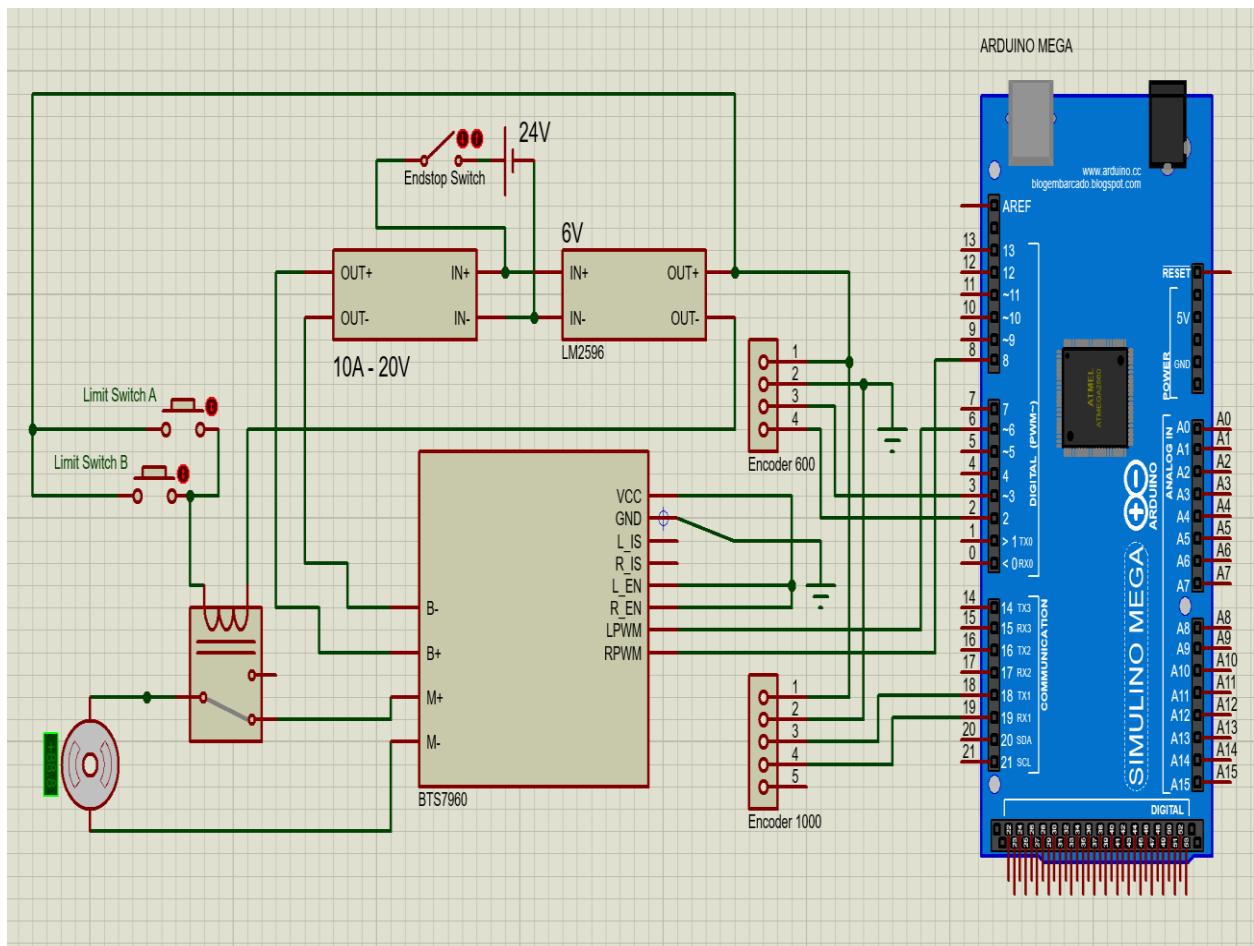
Hình 4.16 Nguồn xung tổng [23]

Bảng 4.7 Thông số kỹ thuật của bộ nguồn

Công suất	250W
Điện áp đầu vào	110-240VAC
Điện áp đầu ra	24VDC
Dòng điện đầu ra cực đại	10A

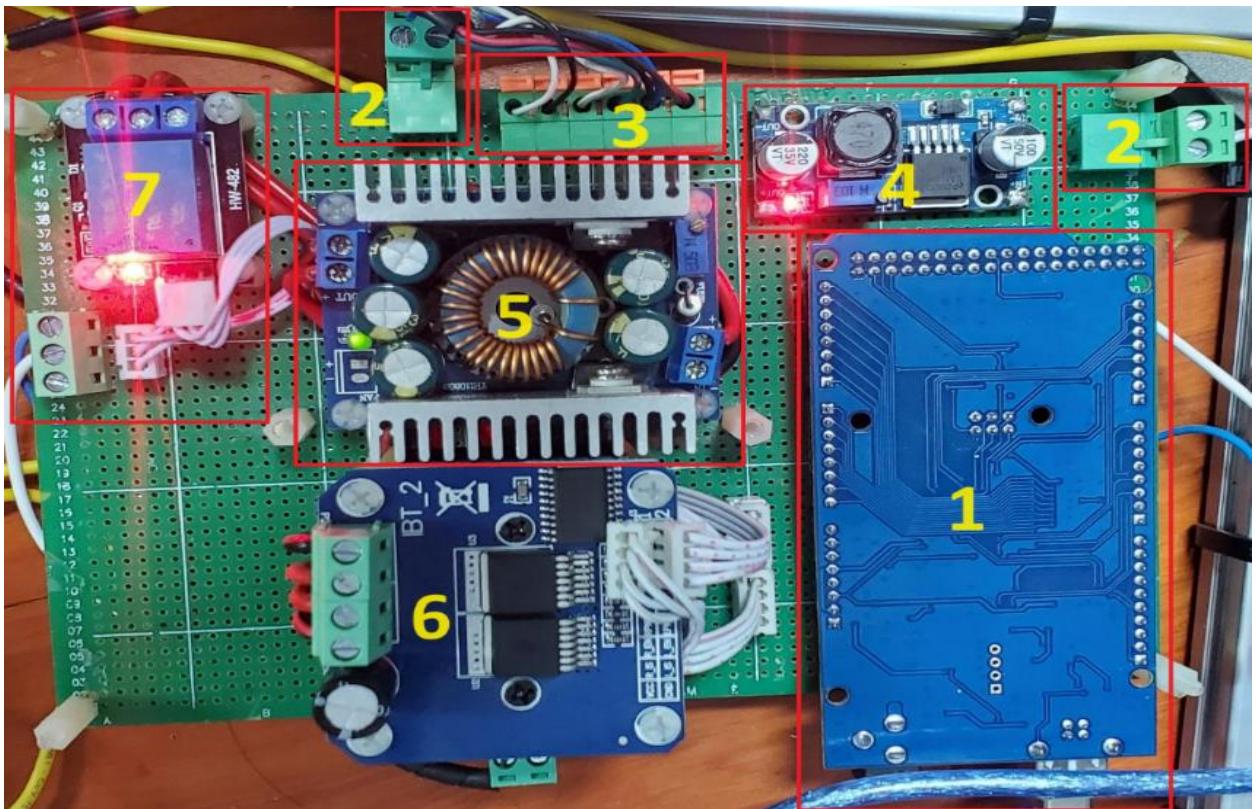
4.5 Sơ đồ mạch điện

4.5.1 Thiết kế mạch mô phỏng trên proteus



Hình 4.17 Thiết kế mô hình mạch điện trên Proteus

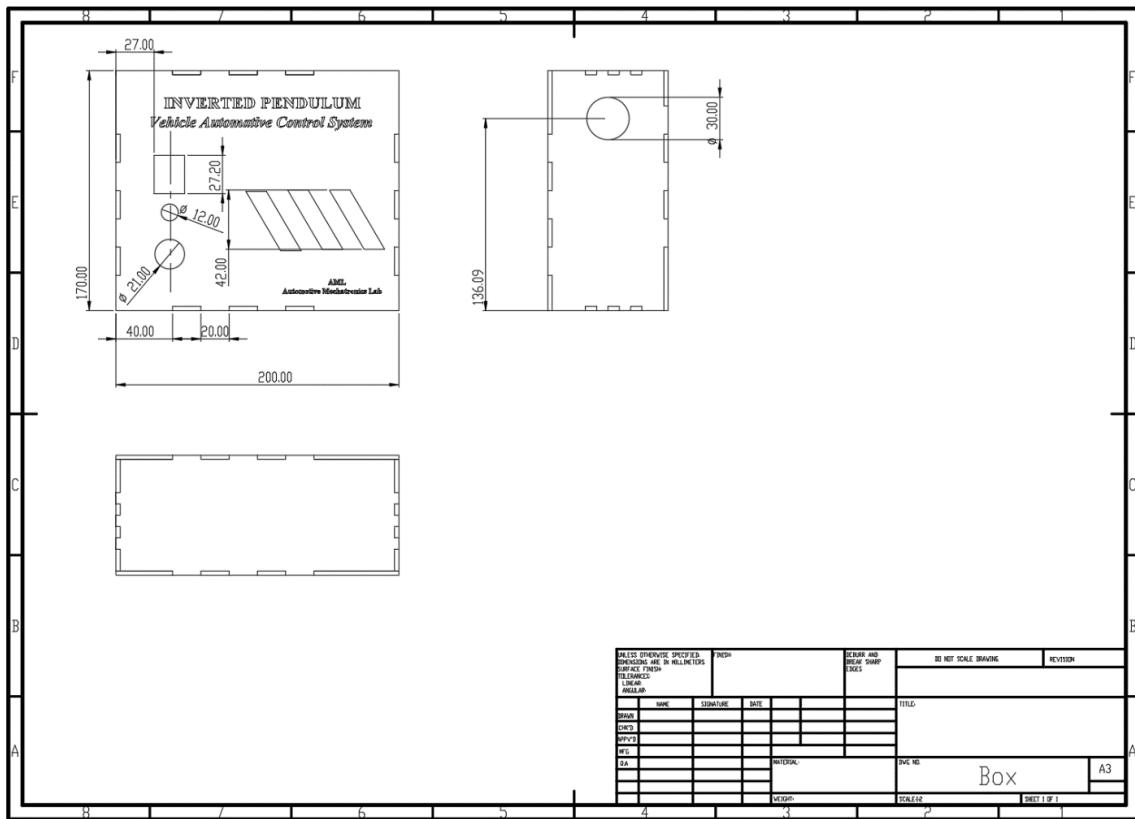
4.5.2 Mạch thiết kế thực tế



Hình 4.18 Thi công mạch điện thực tế

1. Arduino Mega.
2. Ngõ vào từ nguồn 24V.
3. Ngõ cấp nguồn và lấy tín hiệu 2 encoder.
4. Mạch giảm áp LM2596 với điện áp ra 6V.
5. Mạch giảm áp 10A công suất cấp cầu H để chạy động cơ.
6. Cầu H BTS7960.
7. Khối relay ngắt áp cấp cho động cơ khi hệ thống mất ổn định.

Hộp điều khiển sau khi hoàn thành có độ hoàn thiện cao, nhỏ gọn, đạt tính thẩm mỹ, các nút tương tác cần thiết cho hệ thống như dừng khẩn cấp, nút nguồn, nút reset đều được trang bị đầy đủ thuận tiện cho việc điều khiển, và dễ dàng ngắt nguồn hệ thống khi có sự cố xảy ra.



Hình 4.19 Bản thiết kế hộp điều khiển (đơn vị: mm)



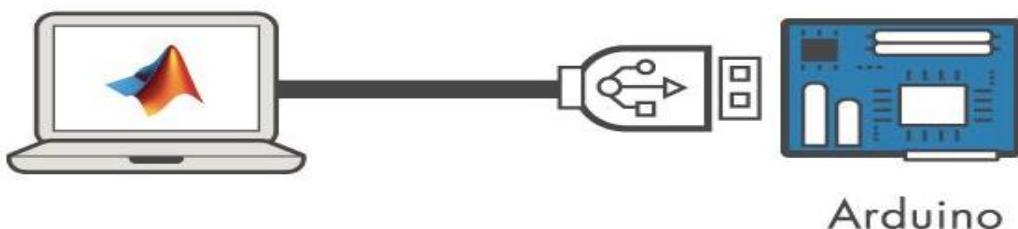
Hình 4.20 Hộp điều khiển mô hình sau khi hoàn thiện

CHƯƠNG 5: XÂY DỰNG BỘ ĐIỀU KHIỂN TRÊN MATLAB/SIMULINK

5.1 Xây dựng bộ điều khiển cân bằng trên Simulink

5.1.1 Giao tiếp giữa vi điều khiển và Matlab/Simulink

Chương trình điều khiển được viết trên phần mềm MATLAB/Simulink, tận dụng chức năng hỗ trợ lập trình Arduino có sẵn trong MATLAB/Simulink. Lập trình trên MATLAB có ưu điểm là đơn giản, thân thiện với người dùng, dễ hiểu và người dùng có thể dễ dàng kiểm tra, điều chỉnh.

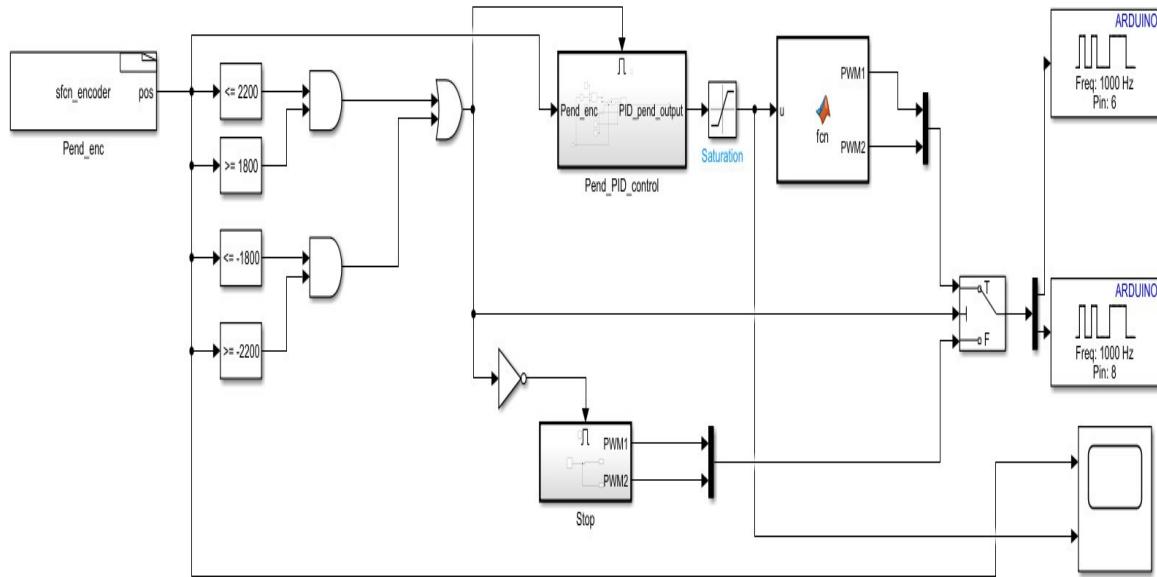


Hình 5.1 Giao tiếp giữa Simulink và vi điều khiển [33]

5.1.2 Xây dựng bộ điều khiển cân bằng con lắc bằng thuật toán PID

Mô hình sử dụng thuật toán PID đơn giản, với đầu vào là error của điểm setpoint đã được thiết lập từ trước và giá trị thực tế của encoder con lắc, đầu ra sẽ là tín hiệu có giá trị từ -255 đến 255 đại diện cho mức duty sẽ xuất ra cho 2 chân PWM và chiều của chúng.

Ngoài ra thuật toán còn giới hạn điều kiện cân bằng của bộ điều khiển, bộ điều khiển chỉ thực hiện cân bằng trong khoảng xấp xỉ từ +15 độ đến -15 độ tính từ điểm cân bằng hướng lên theo phương thẳng đứng, ngoài khoảng đó hệ thống sẽ xuất ra giá trị PWM với duty bằng 0 để dừng xe đẩy. Nguyên nhân cho việc điều khiển cân bằng trong một số góc cố định là bởi vì nếu cân bằng ở những góc lớn hơn thì sẽ gặp rất nhiều khó khăn liên quan đến tốc độ, gia tốc của motor điện một chiều và độ dài cố định của thanh trượt, lúc này hệ thống cần một khoảng dịch chuyển rất lớn để cân bằng ổn định con lắc.



Hình 5.2 Chương trình điều khiển 1 hệ PID cho con lắc trên Simulink

Giá trị điểm setpoint của con lắc sẽ phụ thuộc vào chiều quay của nó. Trong trường hợp, nếu con lắc được swing up và quay theo chiều kim đồng hồ sẽ đặt giá trị setpoint bằng 2000 và ngược lại, nếu quay ngược chiều kim đồng hồ sẽ là -2000.

Các thông số cài đặt cho bộ điều khiển:

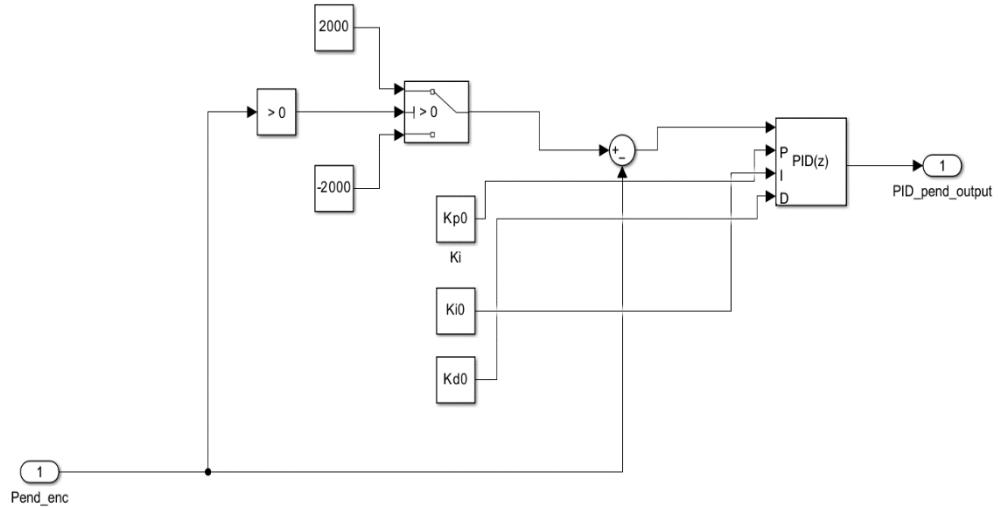
- Bộ thông số PID cho thuật toán:

$$K_p = 1.08$$

$$K_i = 10.8$$

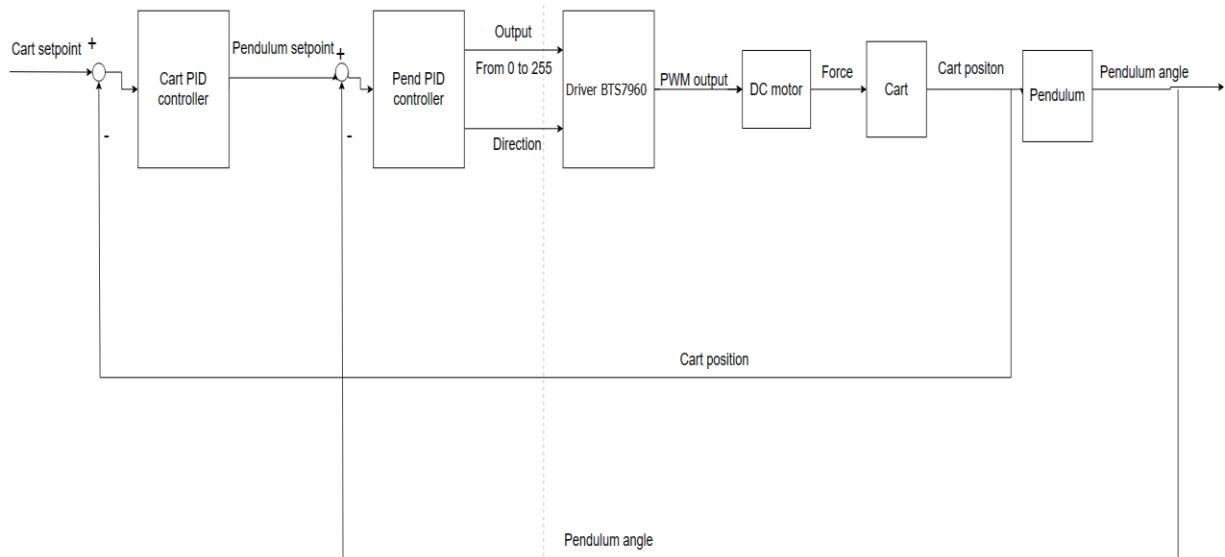
$$K_d = 0.27$$

- Thời gian lấy mẫu cho Encoder, tính toán các bộ PID: 5ms
- Tần số điều khiển động cơ: 1000Hz



Hình 5.3 Bộ điều khiển PID cho hệ con lắc

5.1.3 Xây dựng bộ điều khiển cân bằng hệ Cascade PID



Hình 5.4 Lưu đồ giải thuật cho bộ điều khiển Cascade PID

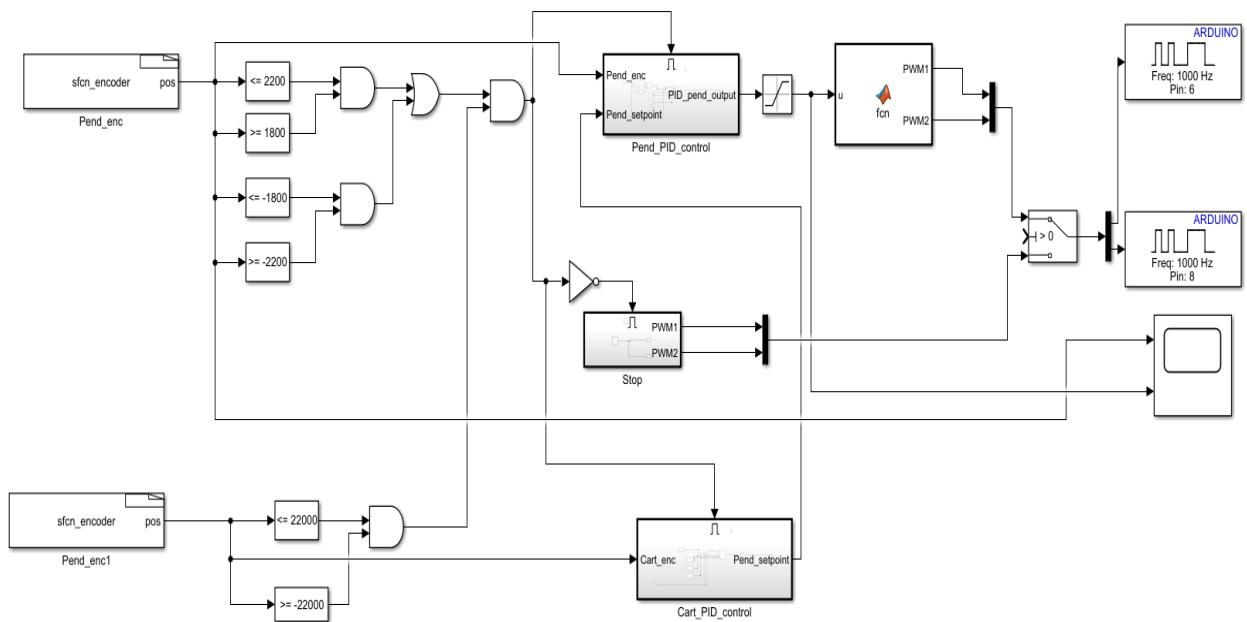
Trong đề tài lần này có thể xem hệ điều khiển là hệ MISO (Multi input single output), trong đó các tín hiệu hồi tiếp về làm giá trị đầu vào là vị trí xe và vị trí con lắc, và chúng ta cần điều khiển một giá trị đầu ra là điện áp động cơ. Nên nhóm đã sử dụng phương pháp Cascade control PID với out của bộ PID1 sẽ là input của bộ PID2.

Ưu điểm của bộ điều khiển cascade đối với hệ con lắc ngược trên xe sẽ giúp giảm dao động khi đưa xe về vị trí setpoint. Trong khi đó đối với bộ PID 2DOF sẽ làm cho hệ dao động mạnh về có sự mâu thuẫn trong quá trình điều khiển, bởi vị đầu ra của hai hệ này là tham chiếu với nhau. Nếu một trong hai hệ gặp nhiễu loạn, hệ thống chung sẽ bị ảnh

hướng ngay lập tức. Bộ điều khiển không thể nào vừa cân bằng con lắc vừa đặt xe về vị trí setpoint được và hệ sẽ dao động quanh điểm setpoint liên tục. Đối với cascade, bộ một bộ điều khiển có ảnh hưởng ít hơn khi gặp nhiễu loạn sẽ đóng vai trò dập nhiễu cho bộ còn lại. Hai bộ này sẽ cùng ảnh hưởng với nhau.

Bộ điều khiển PID vòng ngoài (hệ xe) có nhiệm vụ tính toán một góc mà con lắc cần đạt tới để xe trượt có thể hướng về điểm giữa thanh trượt trong khi cân bằng, góc này sẽ là một góc nhỏ trong khoảng 0-10 độ hướng vào trong.

Bộ điều khiển PID vòng trong (hệ con lắc) tương tự bộ điều khiển PID điều khiển cân bằng, ở trên trong lấy Setpoint từ bộ điều khiển PID vòng ngoài và tính toán ra tín hiệu điều khiển motor cần thiết để có thể đạt được Setpoint của điểm cân bằng.



Hình 5.5 Chương trình điều khiển cân bằng con lắc ngược bằng cascade control

Các thông số cài đặt cho bộ điều khiển:

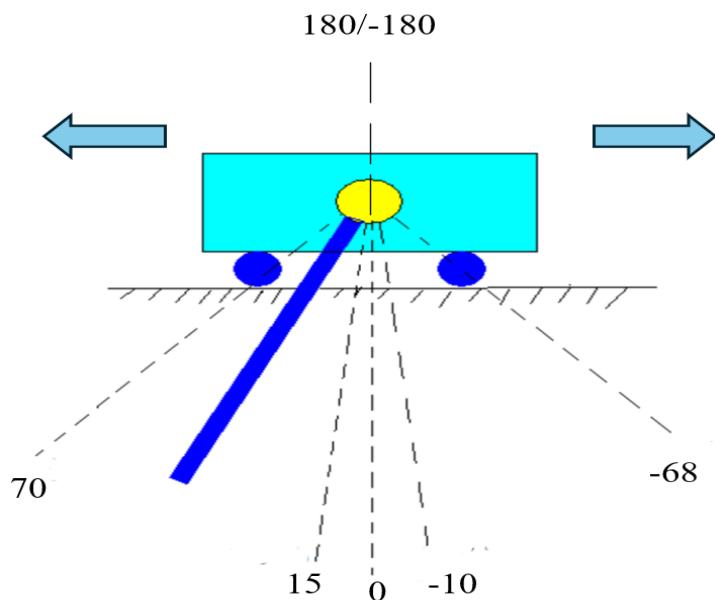
- Bộ thông số PID bộ điều khiển vòng ngoài (điều khiển cho xe trượt về giữa):
 - $K_p = 1.08$
 - $K_i = 10.8$
 - $K_d = 0.27$
- Bộ thông số PID cho bộ điều khiển trong (điều khiển cân bằng con lắc):
 - $K_p = 0.0025$

$$Ki = 0.002$$

$$Kp = 0.003$$

- Thời gian lấy mẫu cho Encoder, tính toán các bộ PID: 5ms
- Tần số điều khiển động cơ là 1000Hz

5.2 Xây dựng bộ điều khiển Swing-up cho con lắc ngược



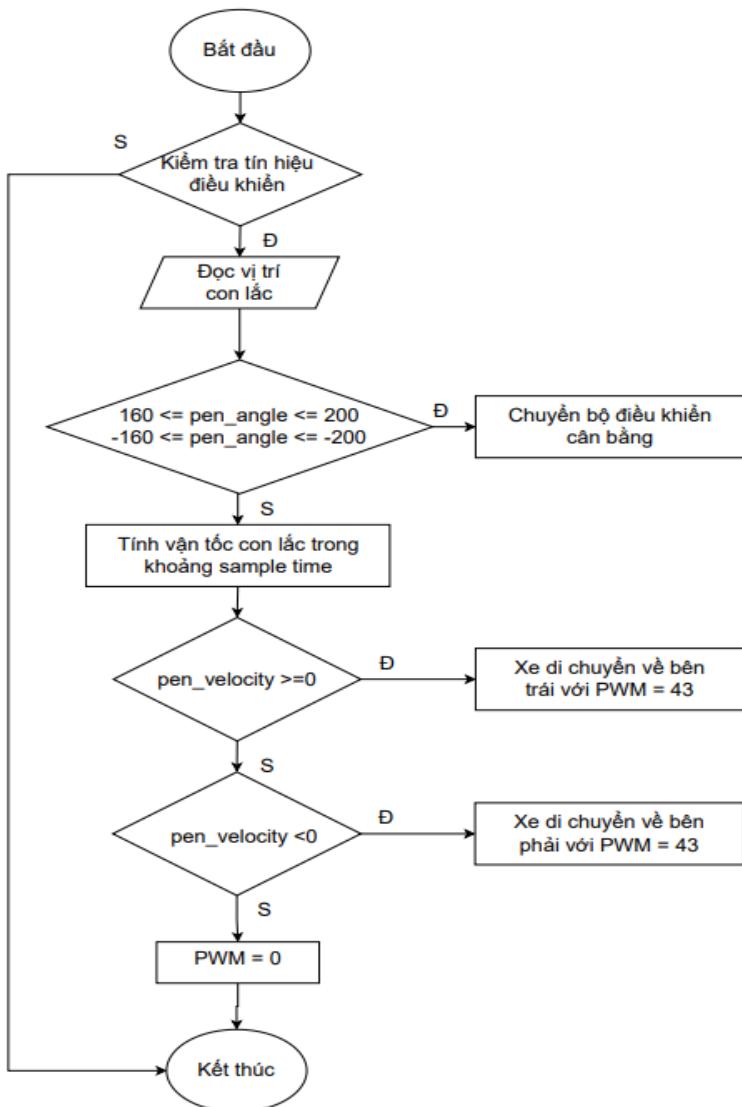
Hình 5.6 Chia vùng điều khiển swing up

Trong đề tài này, nhóm chọn gốc tọa độ nằm tại vị trí cân bằng bền là 0^0 , vị trí cân bằng con lắc là 180^0 . Để con lắc di chuyển sang góc 15^0 thì xe phải di chuyển sang phải, lực làm cho con lắc di chuyển là lực quán tính tác động lên thanh con lắc. Để con lắc quay ngược lại thì xe phải di chuyển sang trái. Không thể đưa con lắc lên vị trí cân bằng trong một đường di chuyển, do sự giới hạn về momen của động cơ và chiều dài thanh trượt, nên cần phải tạo ra biên độ dao động tăng dần để đưa con lắc lên vị trí cân bằng. Tại cùng một lực tác dụng tại một thời điểm phù hợp có thể tăng dần biên độ dao động của con lắc. Dựa vào nguyên lý trên, nhóm đã thiết kế và so sánh 2 phương pháp điều khiển swing up khác nhau để khảo sát sự hiệu quả.

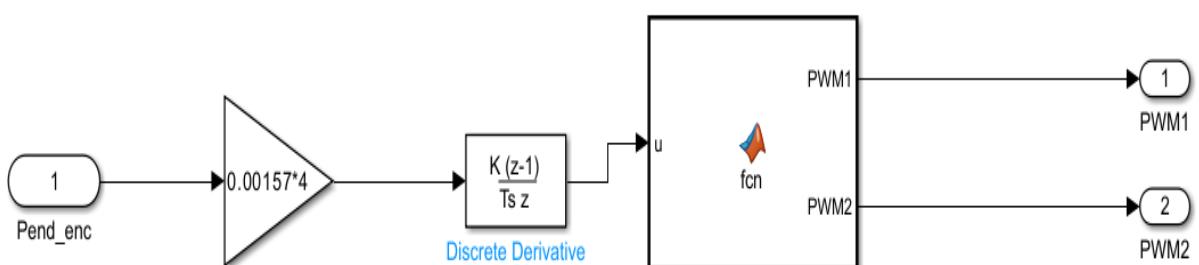
Phương pháp 1: Swing up bằng điều khiển vận tốc

Cho xe di chuyển một quãng đường với tốc độ quay của động cơ phù hợp và đảo chiều quay của động cơ khi vận tốc của con lắc bằng không. Tại thời điểm vận tốc của con lắc bằng không động cơ đảo chiều tiếp thêm năng lượng giúp con lắc quay được góc lớn

hơn so với trước đó. Cùng 1 khoảng thời gian, biên độ dao động của con lắc sẽ tăng dần nếu đặt thời điểm đảo chiều động cơ phù hợp.



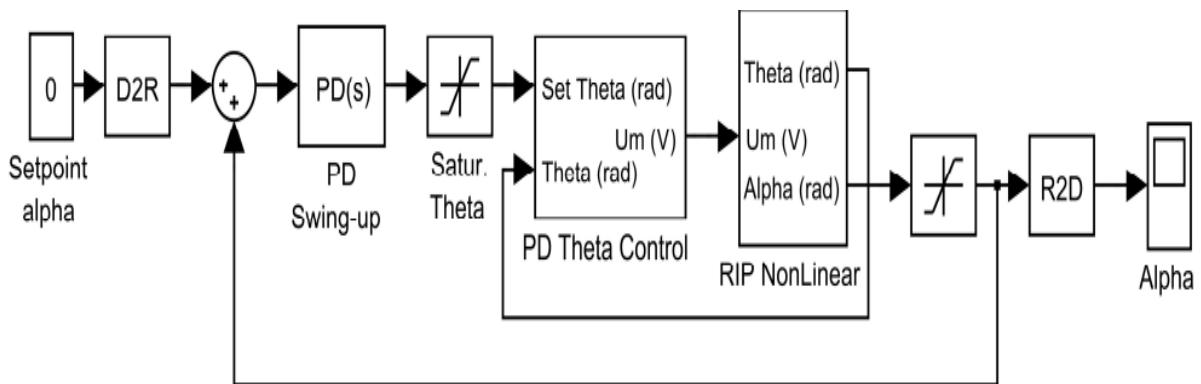
Hình 5.7 Lưu đồ điều khiển swing up bằng phương pháp kiểm soát vận tốc



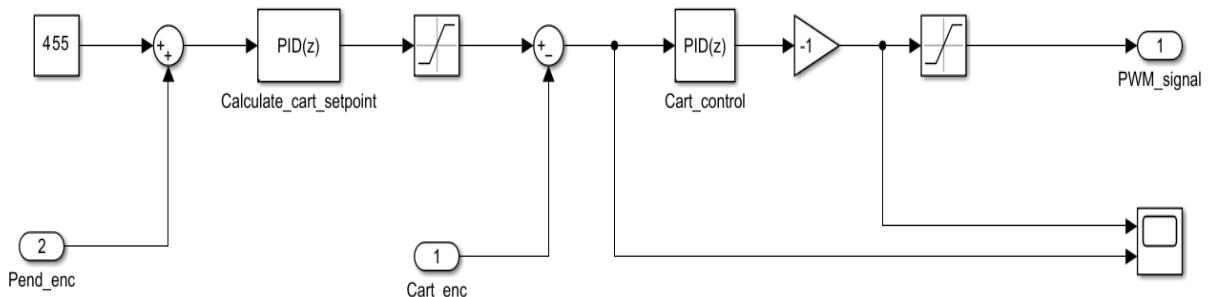
Hình 5.8 Bộ điều khiển Swing up bằng phương pháp vận tốc

Phương pháp 2: Thiết kế bộ điều khiển PD hồi tiếp dương

Phương pháp tích lũy năng lượng bằng phương pháp dùng bộ điều khiển PD hồi tiếp dương. Phương pháp này có ưu điểm là cấu trúc đơn giản, hiệu quả, dễ tinh chỉnh. Bộ điều khiển PD vòng ngoài có tác dụng tính toán quỹ đạo của xe trượt và bộ điều khiển vòng trong điều khiển cho xe trượt đến vị trí đó. Bộ điều khiển PD vòng trong sẽ điều khiển xe trượt đến vị trí mong muốn. Vòng hồi tiếp dương có tác dụng làm tăng thêm năng lượng cho hệ khi con lắc lên cao gần setpoint.



Hình 5.9 Lưu đồ thuật toán điều khiển Swing-up sử dụng thuật toán PD [9]



Hình 5.10 Chương trình điều khiển Swing-up sử dụng PD

Các thông số cài đặt cho bộ điều khiển:

- Bộ thông số PID bộ điều khiển vòng ngoài (điều khiển cho xe trượt về giữa):
 - $K_p = 2$
 - $K_i = 0$
 - $K_d = 0.75$

- Bộ thông số PID cho bộ điều khiển trong (điều khiển cân bằng con lắc):

$$K_p = 0.05$$

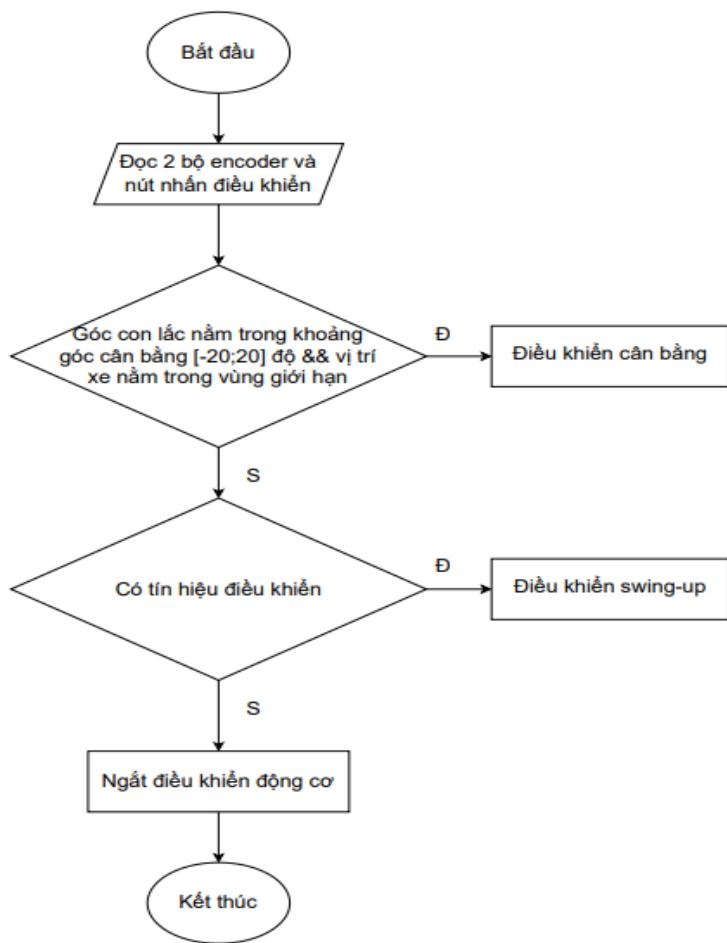
$$K_i = 0$$

$$K_d = 0.02$$

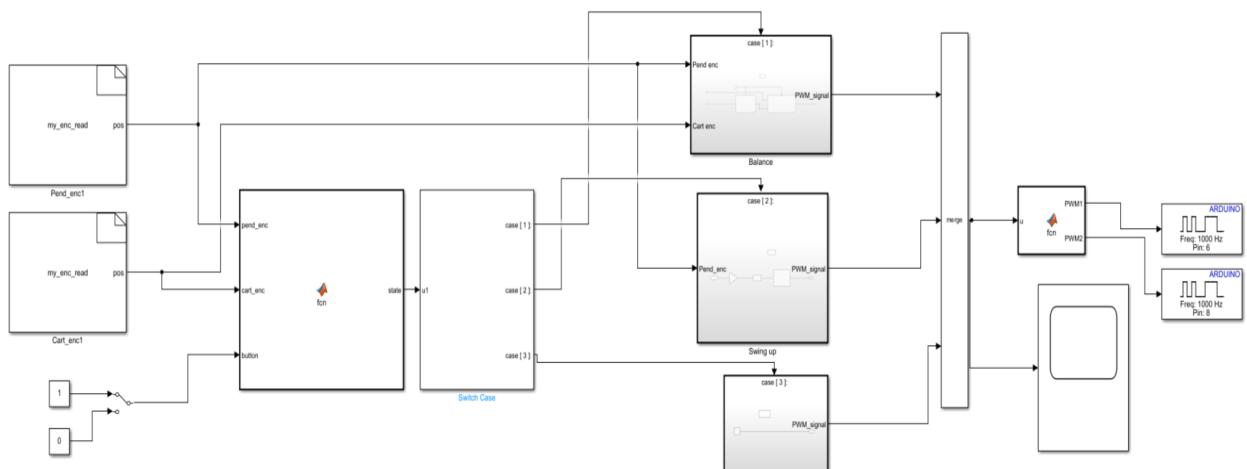
- Thời gian lấy mẫu cho Encoder, tính toán các bộ PID: 5ms

5.4 Bộ điều khiển Swing-up và Cascade cân bằng con lắc ngược

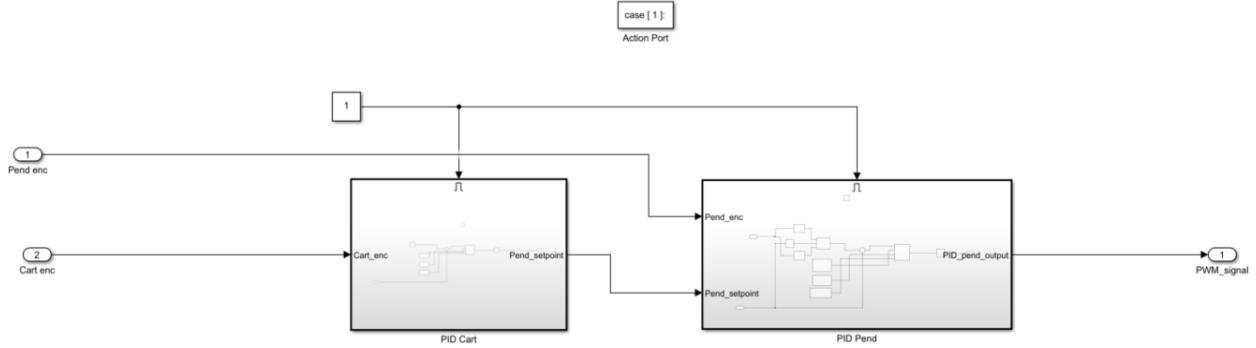
Trong quá trình thực nghiệm, nhóm đã nhận thấy rằng, Arduino Mega sẽ bị nghẽn tại điểm chuyển pha điều khiển, do vừa phải đọc cả 2 bộ encoder với độ phân giải cao. Thư viện có sẵn sử dụng phương pháp đọc sự thay đổi bất kì trạng thái trên phase A và phase B nên số xung đọc được sẽ gấp 4 lần so với thực tế. Trong một sample time (T_s), vi điều khiển “bị ép” phải vừa đọc 2 bộ encoder có độ phân giải cao cùng một lúc, bộ encoder con lắc 4000 xung/vòng, bộ encoder xe 2400 xung/vòng, vừa phải xử lý tín hiệu điều khiển sao cho hệ thống đáp ứng kịp thời. Một số hiệu tượng khi vi điều khiển bị nghẽn: làm cho hệ không cân bằng được; tại điểm chuyển pha cân bằng hệ sẽ bị giật; điểm setpoint của xe sẽ bị trôi; hệ dao động mạnh qua lại tại điểm setpoint của xe. Để giải quyết vấn đề trên, nhóm đã tăng sample time ($T_s = 10\text{ms}$) thay vì 5ms như trước đó, đồng thời giảm độ phân giải của bộ đọc encoder xuống bằng cách viết lại thư viện mới, đọc xung từ encoder khi có cạnh lên, số xung đọc được sẽ đúng với giá trị thực tế. Lúc này các điểm setpoint, điều kiện sẽ thay đổi và hệ số PID sẽ tăng lên gấp 4 lần.



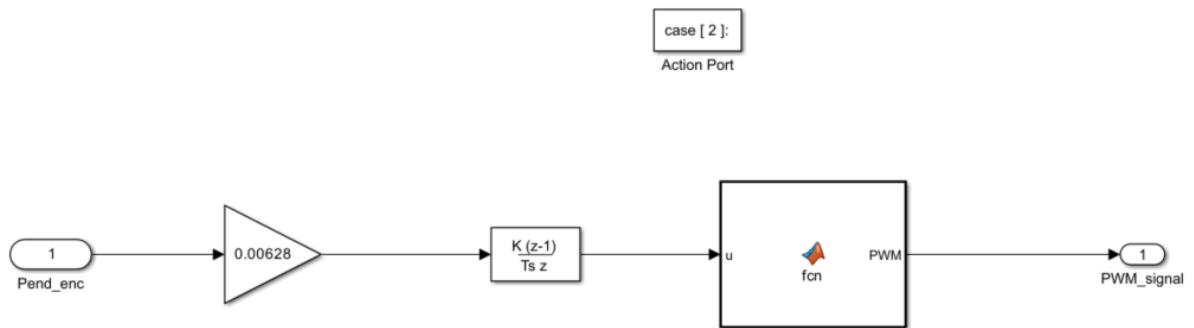
Hình 5.11 Lưu đồ thuật toán điều khiển kết hợp swing-up và cân bằng



Hình 5.12 Chương trình điều khiển trên simulink



Hình 5.13 Khối bộ điều khiển cascade



Hình 5.14 Khối bộ điều khiển swing up

```

/* Interrupt Service Routine: change on pin A for Encoder 0 */      /* attach interrupts */
void irsPinAEn0(){                                                 switch(enc[0]) {
    /* read pin B right away                                         */
    int drB = digitalRead(Enc[0].pinB);                                case 0:
    /* possibly wait before reading pin A, then read it               */
    debounce(Enc[0].del);                                              attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, CHANGE);
    int drA = digitalRead(Enc[0].pinA);                                              attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);
    /* this updates the counter                                         */
    if (drA == HIGH) { /* low->high on A? */                                break;
        /* check pin B */                                                 case 1:
        Enc[0].pos++; /* going clockwise: increment */                      attachInterrupt(getIntNum(Enc[1].pinA), irsPinAEn1, CHANGE);
        } else {                                                       attachInterrupt(getIntNum(Enc[1].pinB), isrPinBEn1, CHANGE);
        Enc[0].pos--; /* going counterclockwise: decrement */           break;
    }                                                               case 2:
} else {                                                       /* must be high to low on A */          attachInterrupt(getIntNum(Enc[2].pinA), irsPinAEn2, CHANGE);
    /* check pin B */                                                 attachInterrupt(getIntNum(Enc[2].pinB), isrPinBEn2, CHANGE);
    Enc[0].pos++; /* going clockwise: increment */           break;
} else {                                                       /* end counter update */          }
} /* end ISR pin A Encoder 0 */                                     */
}

```

a) Trước khi giảm

```

/* Interrupt Service Routine: change on pin A for Encoder 0 */
void irsPinAEn0(){

    /* read pin B right away */
    int drB = digitalRead(Enc[0].pinB);

    /* possibly wait before reading pin A, then read it */
    if (drB == LOW) { /* check pin B */
        Enc[0].pos++; /* going clockwise: increment */
    } else {
        Enc[0].pos--; /* going counterclockwise: decrement */
    }
} /* end ISR pin A Encoder 0 */

/* attach interrupts */
switch(enc[0]) {
    case 0:
        attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, RISING);
        break;
    case 1:
        attachInterrupt(getIntNum(Enc[1].pinA), irsPinAEn1, RISING);
        break;
    case 2:
        attachInterrupt(getIntNum(Enc[2].pinA), irsPinAEn2, RISING);
        break;
}

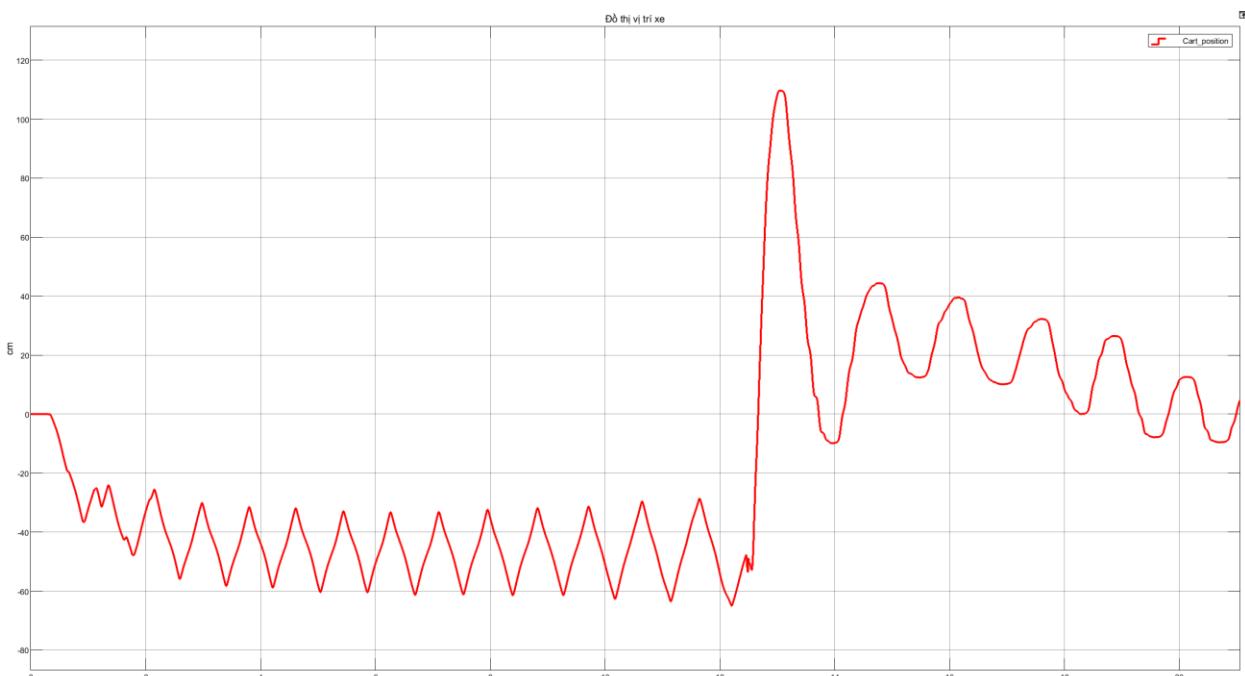
```

b) Sau khi giảm

Hình 5.15 Giảm độ phân giải encoder bằng S-Function Builder

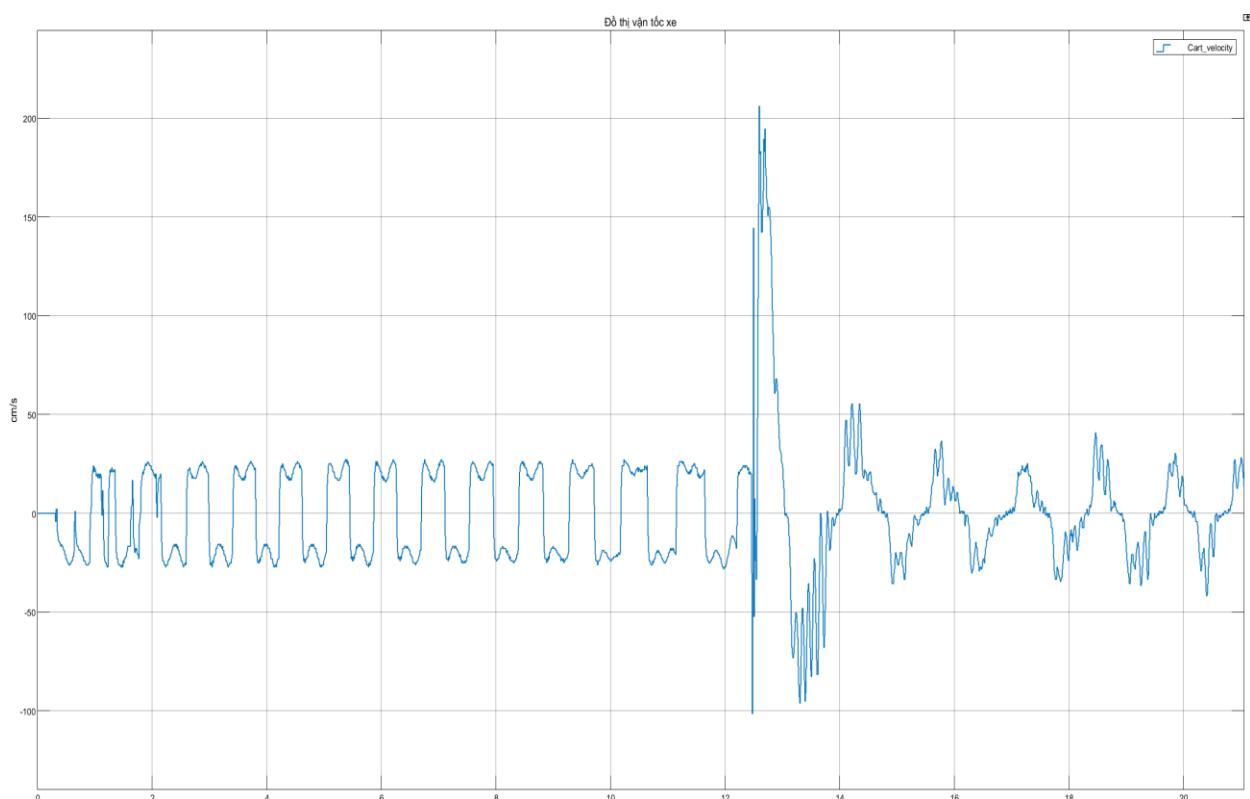
5.5 Kết quả

Trong phương pháp dùng PD để swing-up con lắc ngược, hệ thống đáp ứng nhanh, vị trí con lắc đạt điểm cần bằng dưới 5s, tuy nhiên việc kiểm soát năng lượng tại điểm cân bằng chưa tốt, nên hệ thống không thể cân bằng do quán tính lớn và việc phải đọc thêm bộ encoder từ chiếc xe đã làm cho hệ thống bị nghẽn khi bắt đầu chuyển đổi bộ điều khiển từ swing-up sang cân bằng. Điều này dẫn đến hệ con lắc không thể cân bằng được và trở nên hỗn loạn. Vấn đề đặt ra ở đây là cần kiểm soát năng lượng con lắc tại điểm cân bằng sao cho quán tính và vận tốc đạt nhỏ nhất. Nên nhóm đã sử dụng phương pháp kiểm soát vận tốc con lắc, nên nhóm đã áp phương pháp 2 điều khiển swing-up bằng kiểm soát vận tốc.

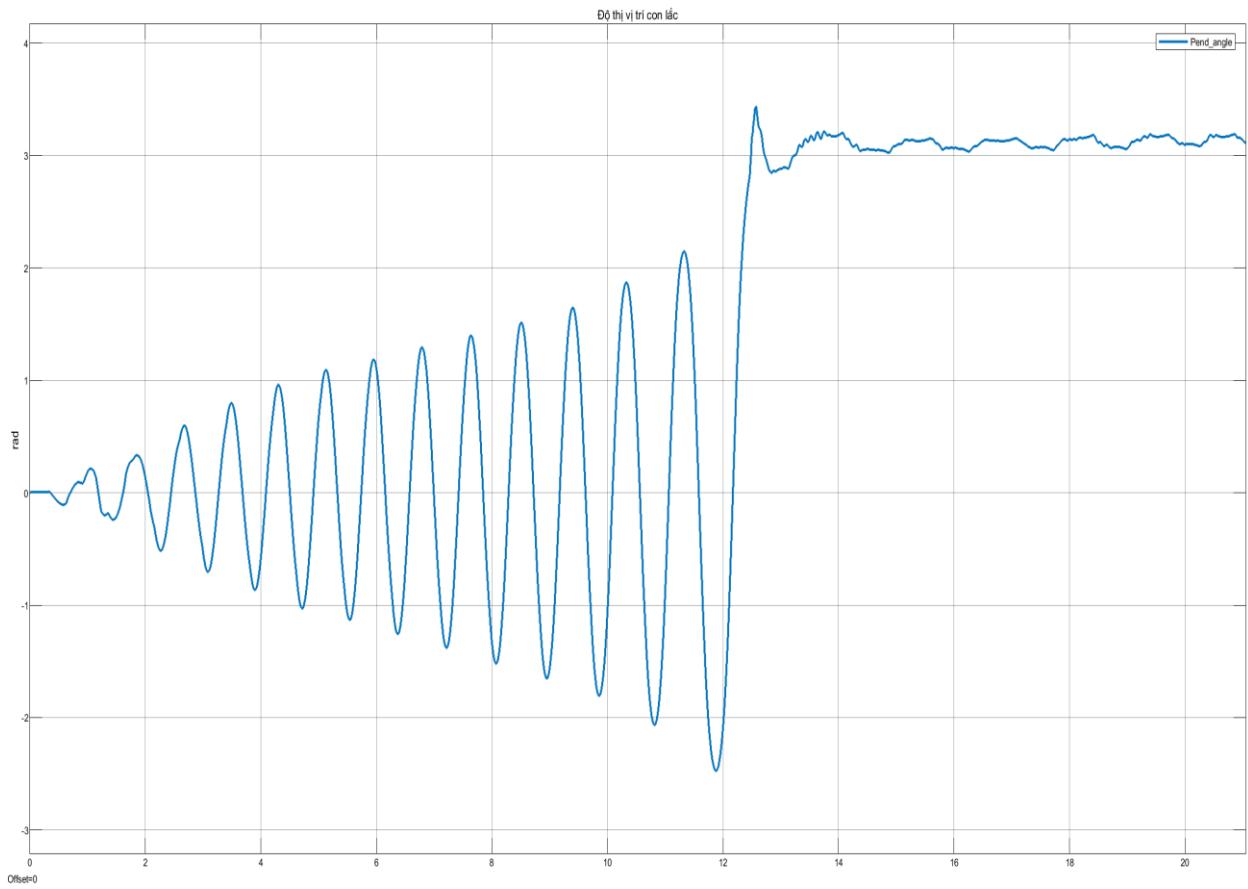


Hình 5.16 Đồ thị vị trí xe (đơn vị cm)

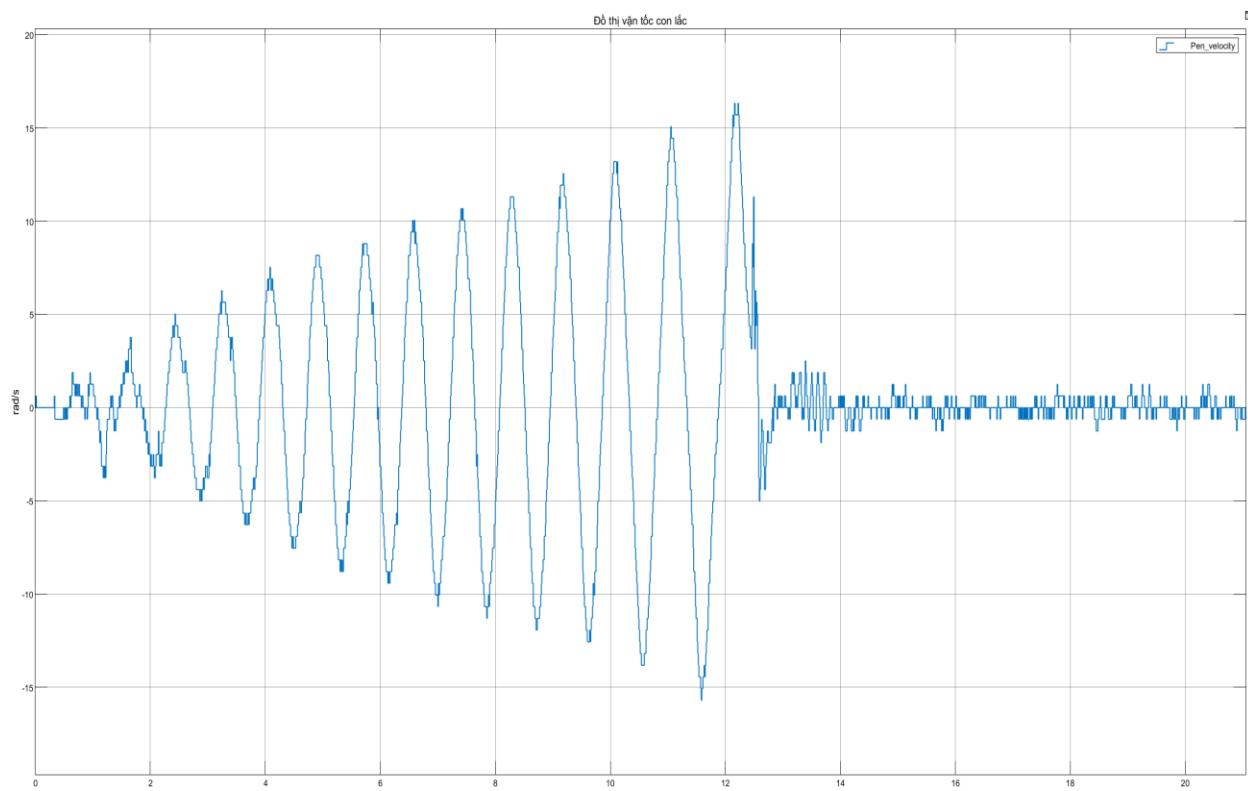
Sau khoảng 13s, con lắc đạt đến điểm chuyển bộ điều khiển, thời gian đáp ứng khá lâu. Tuy nhiên hệ thống được ổn định do chỉ sử dụng tín hiệu encoder đọc được để phân tích ra vận tốc. Thời điểm bắt đầu cân bằng chuyển đổi khá ổn định, không có bất kì sự nghẽn chương trình nào. Tuy nhiên độ hiệu quả cao, nhưng phương pháp này còn nhiều hạn chế do thời gian swing up lớn hơn 10s, nhóm đã thử nghiệm nhiều lần để tìm ra thời gian delay với mục đích giảm năng lượng khi đảo chiều và tìm ra tín hiệu PWM phù hợp, vì yêu cầu đặt ra tại thời điểm đảo chiều động cơ, quán tính của con lắc phải được triệt tiêu hoàn toàn, năng lượng tại điểm chuyển pha cân bằng phải nhỏ nhất.



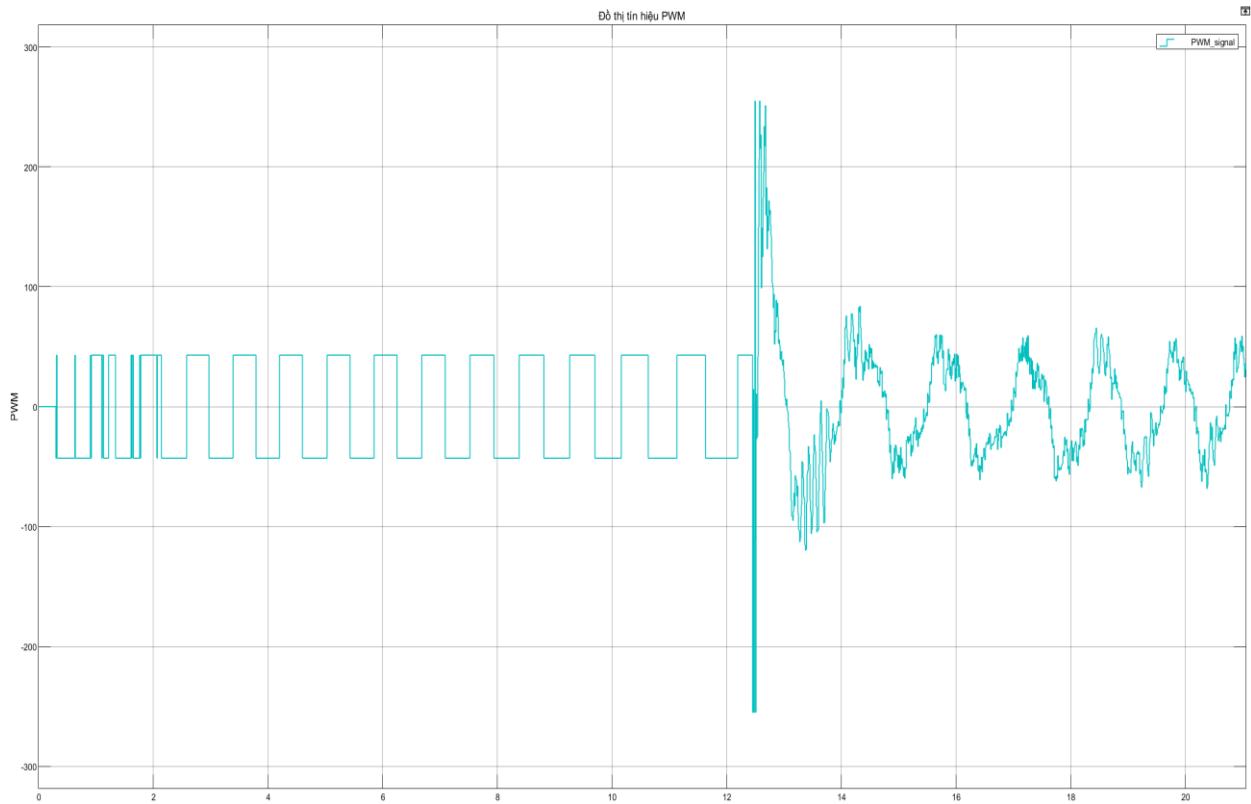
Hình 5.17 Đồ thị vận tốc xe (đơn vị: cm/s)



Hình 5.18 Đồ thị vị trí con lắc (đơn vị: rad)



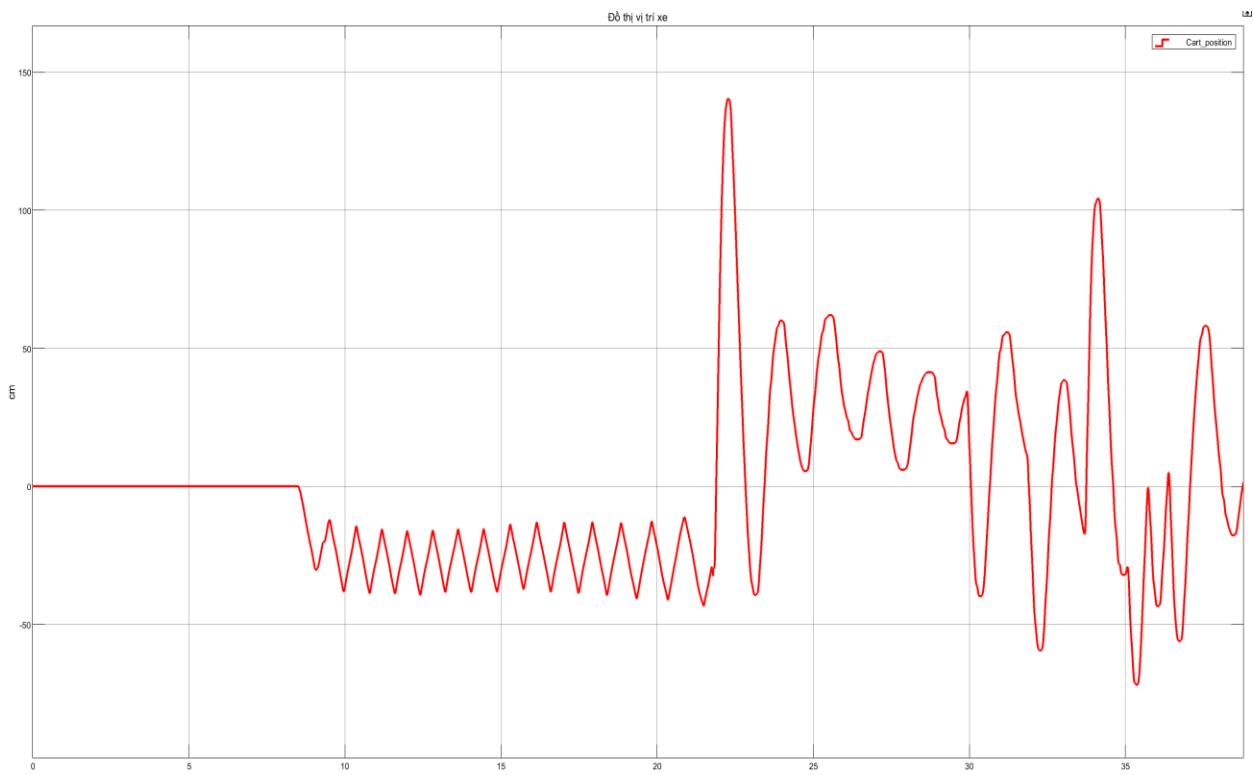
Hình 5.19 Đồ thị vận tốc con lắc (đơn vị: rad/s)



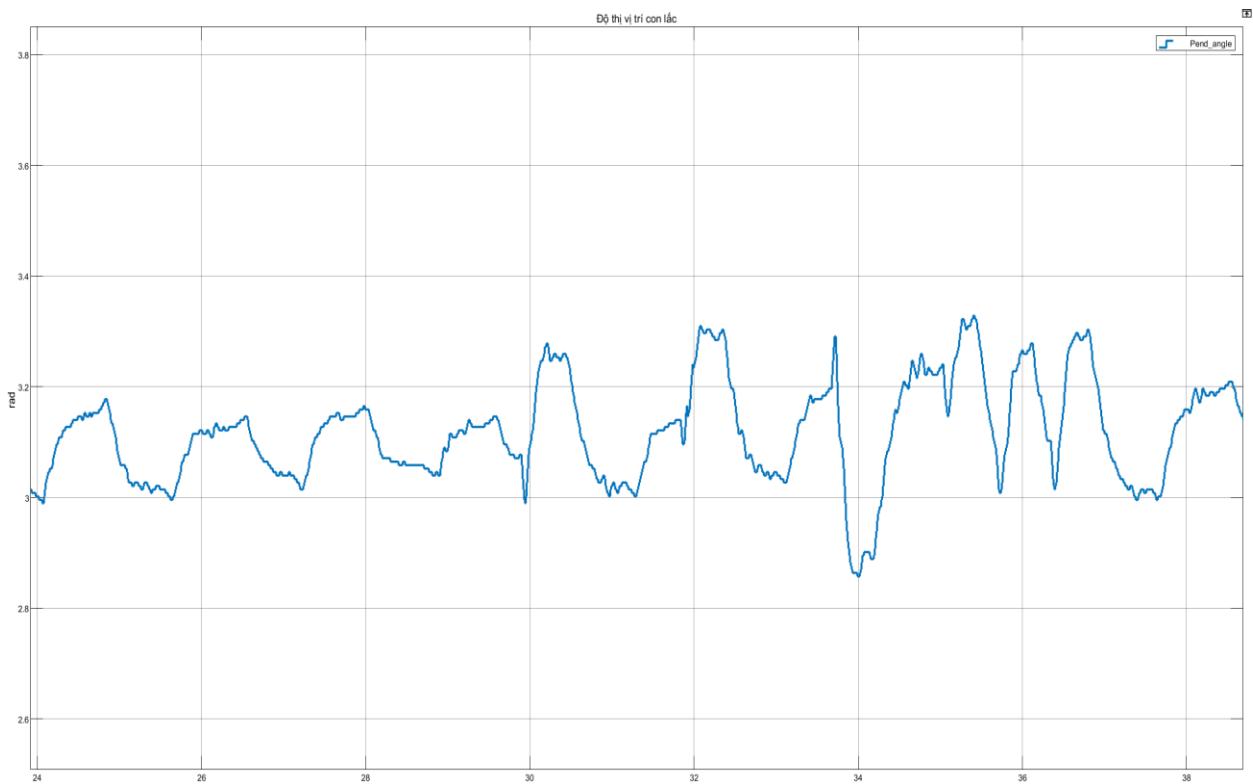
Hình 5.20 Đồ thị tín hiệu PWM (0-255)

Tín hiệu PWM của 1 kênh sẽ ON/OFF theo chu kỳ với 30ms. Thời điểm ban đầu, bộ điều khiển phải đảo chiều động cơ liên tục do vận tốc nhỏ và tín hiệu đầu ra bị nghẽn, tín hiệu PWM bị chương trình Simulink xé nhỏ như (trên hình ...) nên làm cho vị trí xe bị lệch đi so với setpoint ban đầu, tuy nhiên giá trị này rất nhỏ. Đến thời điểm, vận tốc con lắc đủ lớn thì hệ thống chạy ổn định hơn và lực quán tính của con lắc tại điểm cân bằng đạt giá trị nhỏ nhất. Sau khi đặt vị trí cân bằng, tín hiệu PWM sẽ đáp ứng theo đầu ra của bộ PID để giúp con lắc cân bằng

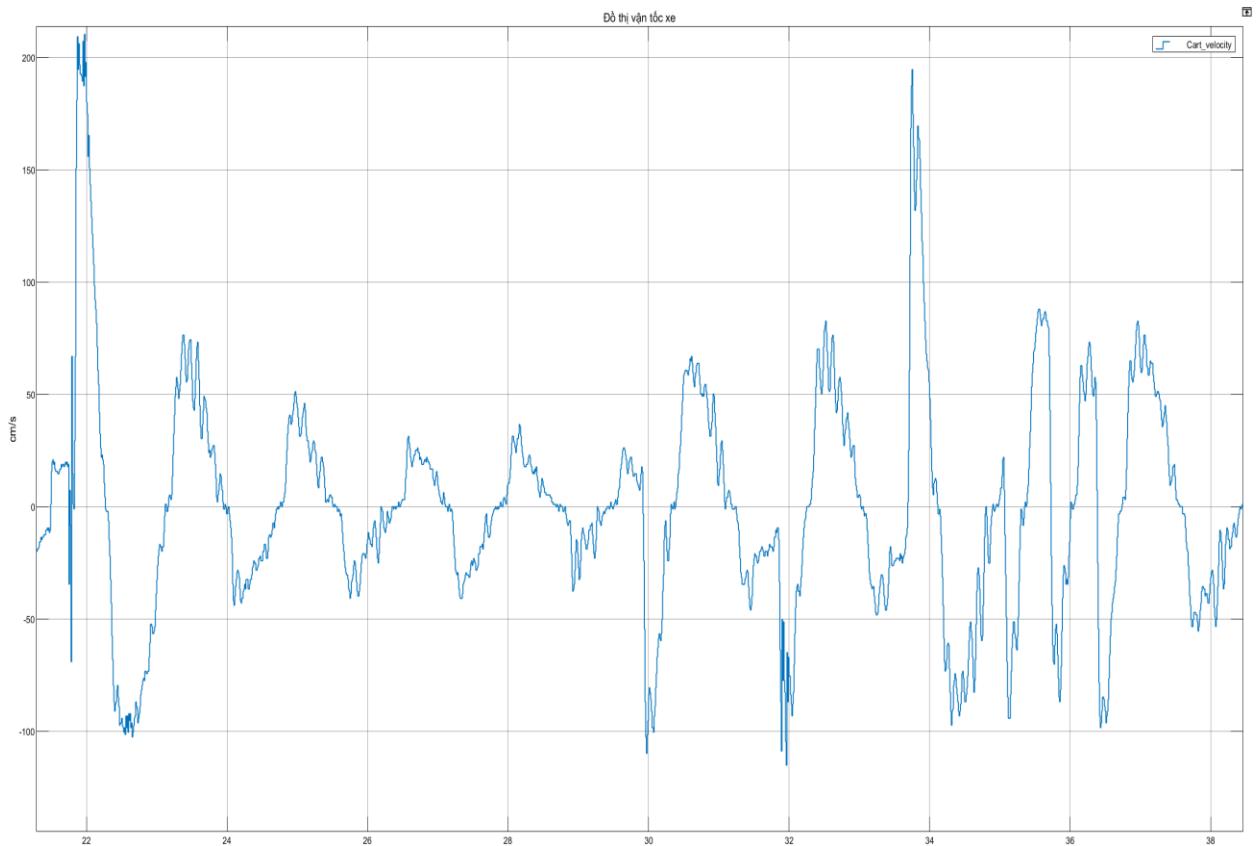
Sau khi con lắc được cân bằng tại vị trí setpoint của hệ xe, nhóm đã tác động ngoại lực lên con lắc, để kiểm tra khả năng đáp ứng khi có nhiễu xãy ra. Kết quả thông qua đồ thị cho thấy, hệ thống đáp ứng với nhiễu khá tốt, có khả năng dập dao động nhanh, phản ứng gần như lập tức.



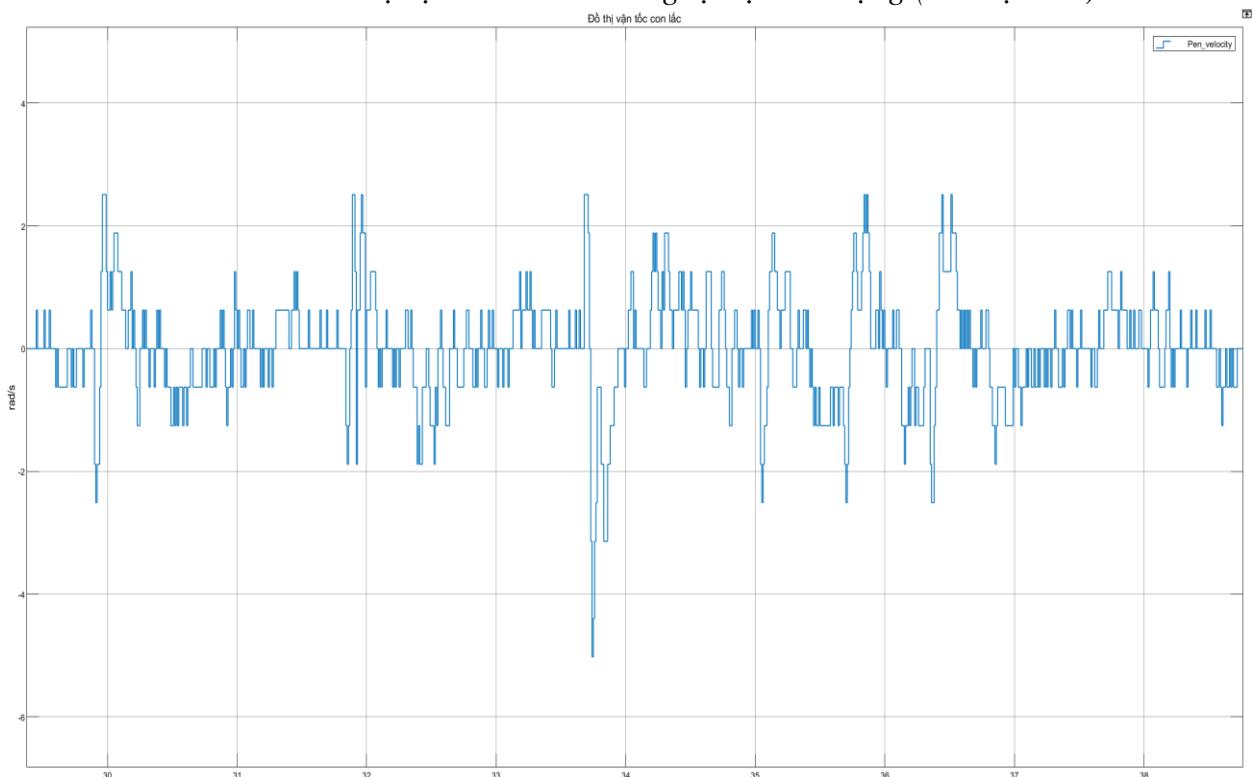
Hình 5.21 Đồ thị vị trí xe khi có ngoại lực tác dụng (đơn vị: cm)



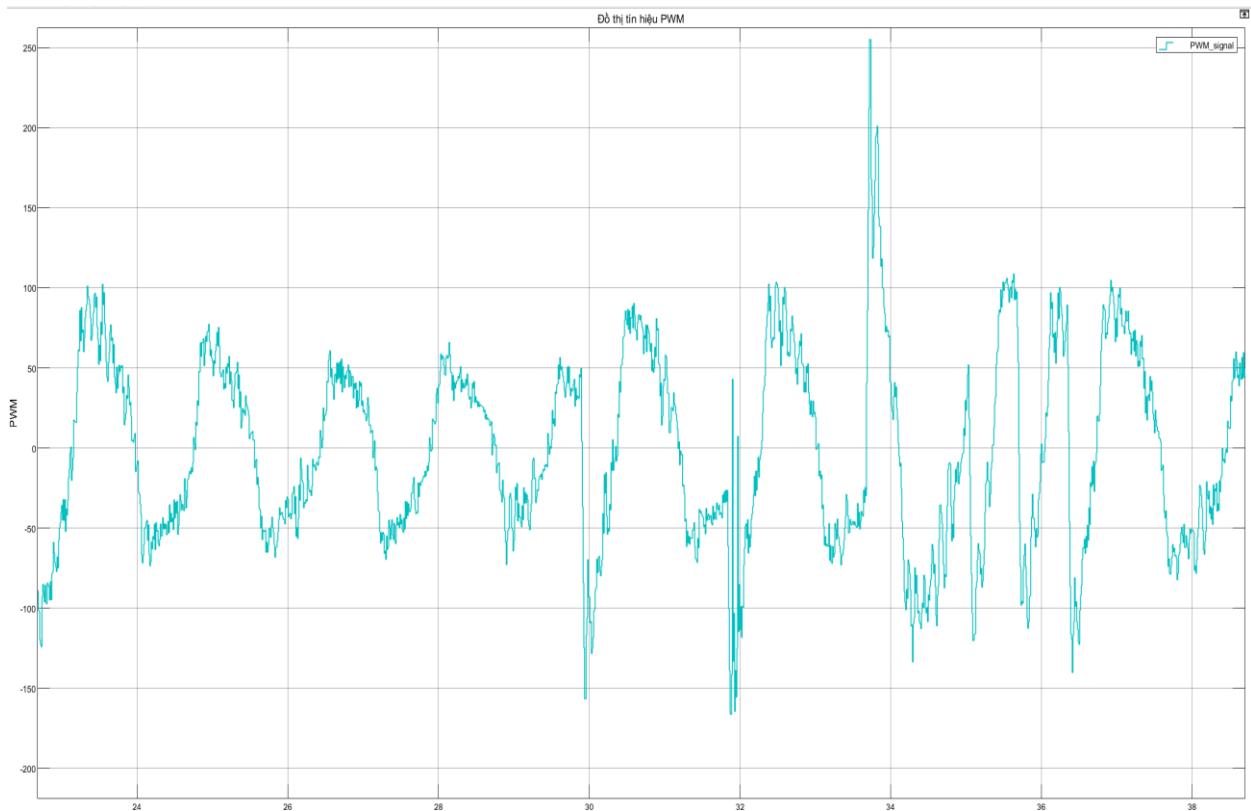
Hình 5.22 Đồ thị vị trí con lắc khi có ngoại lực tác dụng (đơn vị: rad)



Hình 5.23 Đồ thị vận tốc xe khi có ngoại lực tác dụng (đơn vị cm/s)



Hình 5.24 Đồ thị vận tốc con lắc khi có ngoại lực tác dụng (đơn vị: rad/s)



Hình 5.25 Đồ thị tín hiệu PWM khi có ngoại lực tác dụng

Sau khi Flash thành công chương trình từ Simulink xuống Arduino Mega 2560, điều khiển cân bằng con lắc ngược và giữ cho xe trượt về vị trí giữa thành công:

- Quá trình swing-up diễn ra trong 13s
- Con lắc cân bằng tốt, ổn định với các tác động bên ngoài.
- Xe trượt dao động qua lại điểm giữa thanh trượt với một biên độ khoảng 10cm, không ổn định được tại một vị trí.
- Xuất được đồ thị để quan sát theo thời gian thực.

Hạn chế:

- Xe trượt dao động qua lại điểm giữa thanh trượt với một biên độ khoảng 10cm, không ổn định được tại một vị trí.
- Nghẽn chương trình dẫn đến điều khiển phản ứng trễ, không chính xác do phải đọc thêm một Encoder vị trí của xe trượt, dẫn đến khi xe trượt di chuyển nhanh thì chương trình sẽ bị nghẽn do ngắt quá nhiều, gây ra điều khiển sai.
- Xe trượt bị đụng vào khung trượt do thanh trượt có giới hạn, con lắc dễ bị trôi khi chịu tác động khi cân bằng.

Hệ thống không tính toán đúng và đáp ứng kịp nhu cầu của hệ thống, nguyên nhân là do vi điều khiển Arduino Mega không đáp ứng được tốc độ xử lý và tốc độ giao tiếp với phần mềm Simulink. Giải pháp tăng thời gian lấy mẫu của hệ thống lên 15ms hoặc 20ms không hiệu quả do hệ thống không thể điều khiển chính xác, êm diệu hệ thống với thời gian lấy mẫu lớn, gây rung lắc và mất ổn định hệ thống.

Giải pháp tối ưu nhất là lập trình điều khiển hệ thống bằng ngôn ngữ lập trình C không thông qua Simulink để giảm bớt gánh nặng phải giao tiếp với Simulink và tăng được tốc độ xử lý thuật toán bằng cách lập trình thuật toán tối ưu hơn bằng ngôn ngữ lập trình C.

CHƯƠNG 6: XÂY DỰNG BỘ ĐIỀU KHIỂN BẰNG NGÔN NGỮ LẬP TRÌNH C

6.1 Khai báo các giá trị ban đầu của phần cứng

```
#include "PID_v1.h"
#include <Arduino.h>
#include "pwm_control.h"
#include "PWM.h"

//#define USE_ENCODER_h_LIB
#define USE_MY_ENCODER_h_LIB

#ifndef USE_MY_ENCODER_h_LIB
    #include "my_encoder.h"
#else
    #include "Encoder.h"
#endif

#define ENCODER_OPTIMIZE_INTERRUPTS
#define SAMPLE_TIME 3

//UART pins
//UART0 0(RX), 1(TX)
//UART1 19(RX1), 18(TX1)
//UART2 17(RX2), 16(TX2)
//UART3 15(RX3), 14(TX3)

//Encdoer pins (interrupt pins)
#define pin_cart_enc1            3
#define pin_cart_enc2            2
#define pin_pend_enc1           18
#define pin_pend_enc2           19

#define pend_enc_slots          1000
#define cart_enc_slots          600
#define travel_slot              13000
#define cart_middle_pos          0
#define cart_limit_max          6000
#define cart_limit_min          -6000
#define pend_max_positive_angle 550
#define pend_min_positive_angle 450
#define pend_max_negative_angle -550
#define pend_min_negative_angle -450

// DC motor PIN
#define pin_motor_right          6
#define pin_motor_left           8
```

```

//button pin
#define pin_button 7

//MODE
#define balance_mode 1
#define swingup_mode 2
#define stop_mode 3
// Encoder initialize
#ifndef USE_ENCODER_h_LIB
Encoder Enc_cart (pin_cart_enc1,pin_cart_enc2);
Encoder Enc_pend (pin_pend_enc1,pin_pend_enc2);
#endif

#ifndef USE_MY_ENCODER_h_LIB
Encoder * Encoder::instances[3] = {NULL, NULL, NULL};
Encoder Enc_cart(0, pin_cart_enc1,pin_cart_enc2, INTERRUPT_MODE);
Encoder Enc_pend(1, pin_pend_enc1,pin_pend_enc2, INTERRUPT_MODE);
#endif

```

6.2 Xây dựng cài đặt các thông số đầu vào cho bộ điều khiển PID

```

// parameters for PID balancing controllers
double cart_kp = 0.004;
double cart_ki = 0.0005;
double cart_kd = 0.0025;
double cart_setpoint = 0;
double cart_output = 0; //output of cart is an offset for the setpoint of
pendulum angle

double pend_kp = 4.32;
double pend_ki = 43.2;
double pend_kd = 0.108;
double pend_setpoint = 0;
double pend_output = 0;
//pwm signal for the motor
double pwm_output = 0;
// parameters for PD swing-up controllers
double pend_kp_swingup = 6;
double pend_kd_swingup = 0.75;
double cart_kp_swingup = 0.05;
double cart_kd_swingup = 0.02;
double cart_setpoint_swingup = 0;
double pend_setpoint_swingup = 100 ;
double pend_output_swingup = 0;

// velocity and position of the pendulum
double pend_angle = 0;
double opposite_pend_angle = 0;
double pend_angle_in_degree = 0;

```

```

double pend_vel = 0;
double pend_vel_in_degree_per_second = 0;
//velocity and position of the cart
double cart_position = 0;
double cart_vel = 0;
double cart_position_in_centimeter = 0;
double cart_vel_in_centimeter_per_second = 0;

//control mode:balance, swingup, stop
uint8_t control_mode = 0;
//variables for the calculation of the velocity
double timer;
const int array_length = 50;
double cart_position_array[array_length];
double pend_angle_array[array_length];
double timer_array[array_length];

//PID initialize for balancing control
PID PID_cart_balance(&cart_position, &cart_output, &cart_setpoint, cart_kp,
cart_ki, cart_kd, P_ON_E, DIRECT);
PID PID_pend_balance(&pend_angle, &pend_output, &pend_setpoint, pend_kp, pend_ki,
pend_kd, P_ON_E, DIRECT);

//PID initialize for PD swing-up control
PID PID_pend_swingup(&opposite_pend_angle, &cart_setpoint_swingup,
&pend_setpoint_swingup, pend_kp_swingup, 0, pend_kd_swingup, P_ON_E, DIRECT);
PID PID_cart_swingup(&cart_position, &pend_output_swingup,
&cart_setpoint_swingup, cart_kp_swingup, 0, cart_kd_swingup, P_ON_E, REVERSE);

```

6.3 Xây dựng các hàm điều khiển

- Hàm khởi tạo và cập nhật các giá trị của các Encoder:

```

void enc_Setup() {
    //Initialize begin position for cart and pendulum
    Enc_cart.write(0);
    Enc_pend.write(0);
}

void enc_read() {
    pend_angle = Enc_pend.read();
    cart_position = Enc_cart.read();
}

void encoder_update() {
    // Calculate the velocity of the pendulum
    //Pushing the linear position values one step down in the array to make place at
    //element [0] for the present linear position
}

```

```

    for (int i=array_length; i>1; i--){
        cart_position_array[i-1]=cart_position_array[i-2];
    }
    //Pushing the angle values one step down in the array to make place at element
    [0] for the present angle
    for (int i=array_length; i>1; i--){
        pend_angle_array[i-1]=pend_angle_array[i-2];
    }

    //Pushing the timer values one step down in the array to make place at element
    [0] for the present time in milliseconds
    for (int i=array_length; i>1; i--){
        timer_array[i-1]=timer_array[i-2];
    }

    //reading the present values for timer, angle and linear position and save them
    to element 0 in their arrays
    timer=millis();
    timer_array[0]=timer;
    pend_angle=Enc_pend.read();
    opposite_pend_angle = - pend_angle;
    pend_angle_array[0]=pend_angle;
    pend_angle_in_degree = pend_angle_array[0]/pend_enc_slots*360;
    cart_position=Enc_cart.read();
    cart_position_array[0]=cart_position;
    cart_position_in_centimeter = cart_position_array[0]/travel_slot*100;
}

```

- Hàm kiểm tra điều kiện xem con lắc có còn nằm trong khoảng điều khiển và xe vẫn ở trong giới hạn không:

```

bool is_controlable() {
    if((pend_angle < pend_max_positive_angle && pend_angle >
    pend_min_positive_angle)
        || (pend_angle > pend_max_negative_angle && pend_angle <
    pend_min_negative_angle))
    &&
    (cart_position < cart_limit_max && cart_position > cart_limit_min))
    return true;
    else return false;
}

```

- Hàm tính toán vận tốc góc của con lắc và vận tốc con trượt:

```

void calculate_cart_velocity() {
    // Calculate the velocity of the cart

```

```

    cart_vel = (cart_position_array[0]-cart_position_array[array_length-1])/(timer_array[0]-timer_array[array_length-1])*1000;
    cart_vel_in_centimeter_per_second = cart_vel/travel_slot*100;
}

```

```

void calculate_pend_velocity() {
    // Calculate the velocity of the pendulum
    pend_vel = (pend_angle_array[0]-pend_angle_array[array_length-1])/(timer_array[0]-timer_array[array_length-1])*1000;
    pend_vel_in_degree_per_second = pend_vel/pend_enc_slots*360;
}

```

- Hàm điều khiển cân bằng con lắc

```

void calculate_setpoint_for_pend_balance() {
    if (pend_angle >= 0) pend_setpoint = cart_output + pend_enc_slots/2;
    else pend_setpoint = cart_output - pend_enc_slots/2;
}
void balance_control(){
    //PID_cart_balance.SetTunings(cart_kp,cart_ki,cart_kd);
    PID_cart_balance.Compute();
    calculate_setpoint_for_pend_balance();
    //PID_cart_balance.SetTunings(pend_kp,pend_ki,pend_kd);
    PID_pend_balance.Compute();
}

```

- Hàm điều khiển swing-up con lắc:

```

void swingup_control_PD() {
    PID_pend_swingup.Compute();
    PID_cart_swingup.Compute();
}
void swingup_control_basic(){
    calculate_pend_velocity();
    pend_output_swingup = (pend_vel_in_degree_per_second >0)? 63:-63;
    for (int i = 0; i < 300; i++){
        if (is_controlable()) break;
        delay(1);
    }
};

```

- Hàm điều khiển động cơ:

```

void go_right(double pwm_value ) {
    analogWrite(pin_motor_left,0);
    analogWrite(pin_motor_right,pwm_value);
}

```

```

void go_left(double pwm_value) {
    analogWrite(pin_motor_left,pwm_value);
    analogWrite(pin_motor_right,0);
}

void go_stop() {
    analogWrite(pin_motor_left,0);
    analogWrite(pin_motor_right,0);
}

void motor_control(int16_t pwm_value) {
    if(pwm_value > 0) {go_left(pwm_value);}
    else if (pwm_value < 0) {go_right(abs(pwm_value));}
    else {go_stop();}
}

```

- Hàm chọn chế độ điều khiển:

```

void mode_selection(){
    if (is_controlable()) {
        control_mode = balance_mode;
    }
    else if (digitalRead(pin_button) == LOW) {
        control_mode = swingup_mode;
    }
    else {
        control_mode = stop_mode;
    }
}

void mode_selection(){
    if (is_controlable()) {
        control_mode = balance_mode;
    }
    else if (digitalRead(pin_button) == LOW) {
        control_mode = swingup_mode;
    }
    else {
        control_mode = stop_mode;
    }
}

```

- Hàm in các giá trị cần thiết:

```

void print_values(){
    // Serial.print(pend_angle);
    // Serial.print(",");
    // Serial.print(cart_position);
}

```

```

// Serial.print(",");
//Serial.print(opposite_pend_angle);
//Serial.print(",");
Serial.print(pend_setpoint/pend_enc_slots*360); // pendsetpoint in degree
Serial.print(",");
Serial.print(pend_angle_in_degree);
Serial.print(",");
//Serial.print(control_mode);
//Serial.print(",");
Serial.print(pend_vel_in_degree_per_second);
Serial.print(",");
Serial.print(cart_position_in_centimeter);
Serial.print(",");
Serial.print(cart_vel_in_centimeter_per_second);
Serial.print(",");
Serial.println(pwm_output);
}

```

6.4 Chương trình điều khiển chính

```

#include "inverted_pendulum.h"
#include "TimerOne.h"

void setup() {
    Serial.begin(115200);
    Timer1.initialize (5000); //200Hz
    Timer1.attachInterrupt(print_values);

    //DC motor setup
    pinMode(pin_motor_right,OUTPUT);
    pinMode(pin_motor_left, OUTPUT);

    // PID setup for balance control
    PID_pend_balance.SetOutputLimits(-255,255);
    PID_pend_balance.SetSampleTime(SAMPLE_TIME);
    PID_pend_balance.SetMode(AUTOMATIC);

    PID_cart_balance.SetOutputLimits(-30,30);
    PID_cart_balance.SetSampleTime(SAMPLE_TIME);
    PID_cart_balance.SetMode(AUTOMATIC);

    //PID setup for swingup control
    PID_pend.swingup.SetOutputLimits(-3000,3000);
    PID_pend.swingup.SetSampleTime(10);
    PID_pend.swingup.SetMode(AUTOMATIC);

    PID_cart.swingup.SetOutputLimits(-100,100);
}

```

```

PID_cart_swingup.SetSampleTime(10);
PID_cart_swingup.SetMode(AUTOMATIC);

//Encoder setup
enc_Setup();
// set PWM frequency for DC motor approximately 4000Hz by using
divisor
setPwmFrequencyMEGA2560(6,2);
setPwmFrequencyMEGA2560(8,2);
control_mode = balance_mode;
}

void loop() {
//reading values from timer and encoders
encoder_update();
//mode selection
mode_selection();
//control mode selection
if(control_mode == balance_mode) {
    balance_control();
    pwm_output = pend_output;
}
else if(control_mode == swingup_mode) {
    swingup_control_PD();
    //pend_output = 0;
    //swingup_control_basic();
    pwm_output = pend_output_swingup;

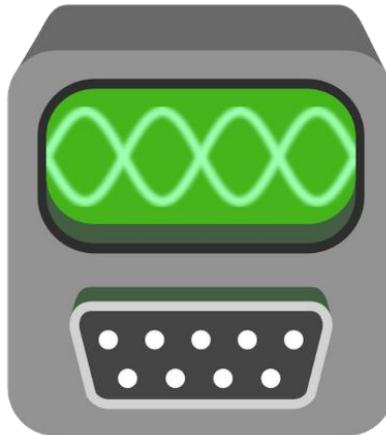
}
else if(control_mode == stop_mode) {
    //go_stop();
    pwm_output = 0;
}
else {
    pwm_output = 0;
}
//motor control
motor_control(pwm_output);
enc_read();
calculate_cart_velocity();
calculate_pend_velocity();
}

```

6.5 Sử dụng phần mềm SerialPlot để vẽ đồ thị

SerialPlot là ứng dụng phổ biến dùng để vẽ đồ thị thời gian thực. SerialPlot chấp nhận 3 loại dữ liệu đầu vào khác nhau:

- Dòng nhị phân đơn giản, hỗ trợ các định dạng số khác nhau (không dấu/có dấu - 8/16/32 bit và số thực)
- Dữ liệu ASCII theo định dạng CSV
- Định dạng khung do người dùng xác định (byte bắt đầu khung, kích thước khung, checksum, v.v...)



Hình 6.1 Hình ứng dụng Serial Plot

Cài đặt ban đầu cho ứng

- Bước 1: Chon cổng Arduino
- Bước 2: Chọn Baud
- Bước 3: Chọn số kêt
- Bước 4: Khởi động l

Hình 6.2 Hình giao diện cài đặt cho Serial Plot

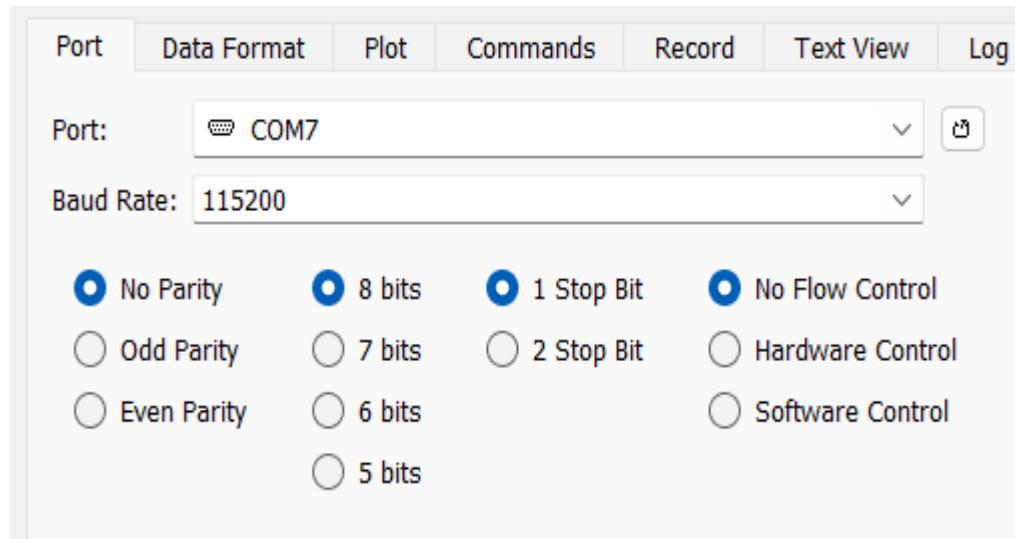
)

ting với đã thiết lập trên

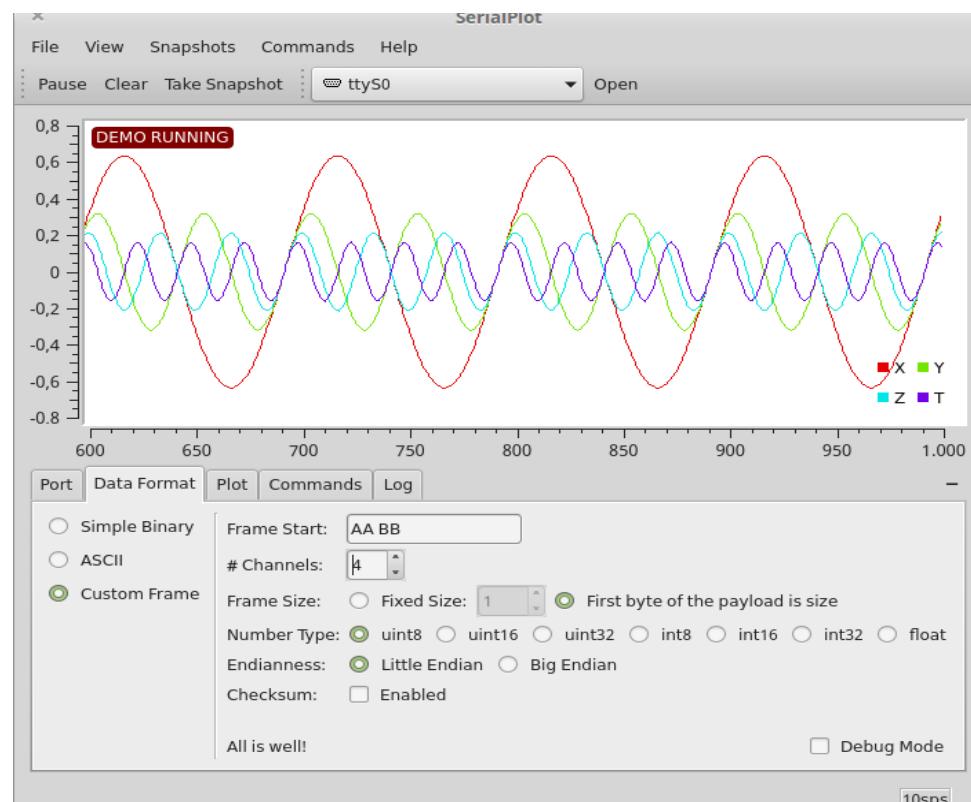
Hình 6.3 Hình đồ thị được vẽ ra bằng Serial Plot

)

Hình 6.4 Giao diện hiển thị trên phần mềm Labviewnh 6.5 Hình ứng dụng Serial Plot



Hình 6.8 **Error! Bookmark not defined.** Hình giao diện cài đặt cho Serial Plot



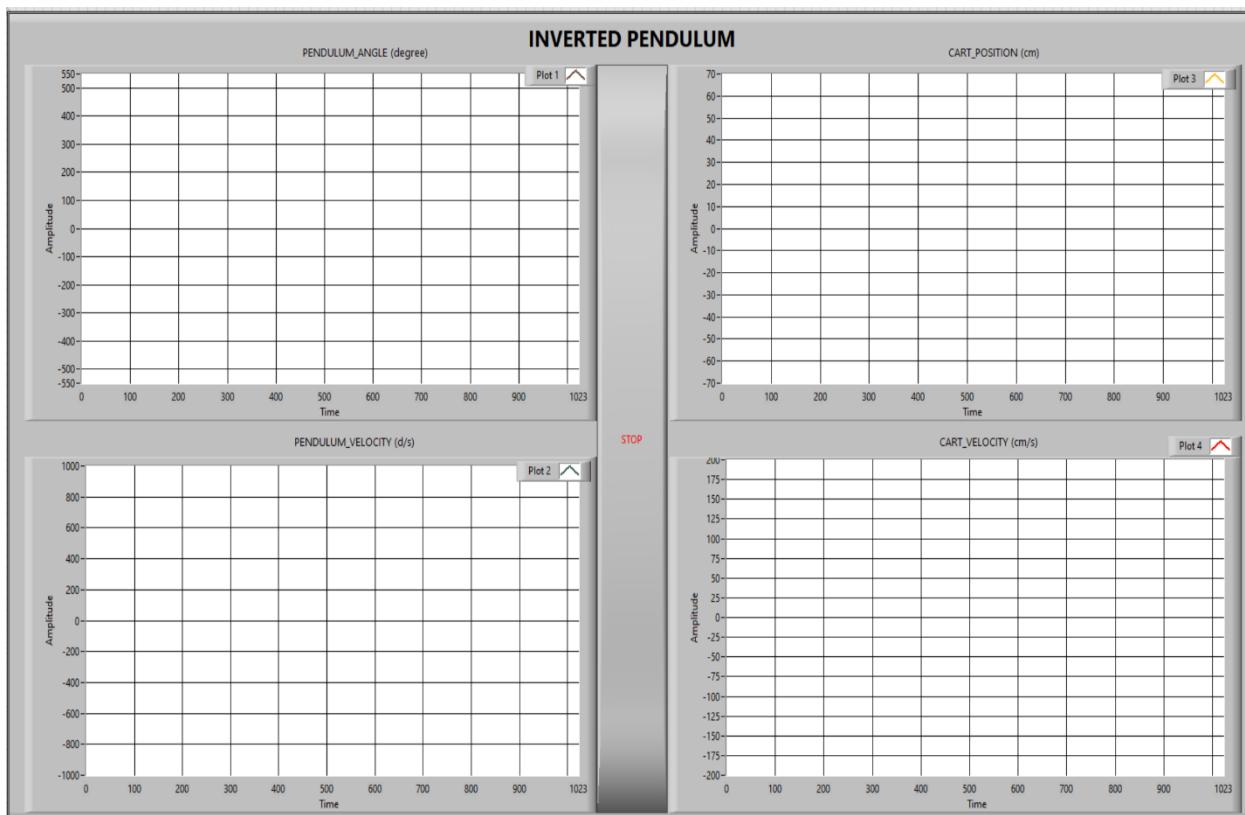
Hình 6.9 Hình đồ thị được vẽ ra bằng Serial Plot

6.6 Hiển

Đ **Hình 6.10 Giao diện hiển thị trên phần mềm Labview** **Hình 6.11** Hình **giao diện** **đồ thị** **được** **vẽ** **ra** **bằng** **Serial Plot** **trường** **mà** **nhóm** **đang** **sử** **dụng** **để** **viết** **code** **C** **cho** **hệ** **thống** **là** **Visual** **Code** **Studio**, **không** **có** **chức** **năng** **hiện** **thì** **đồ** **thị** **trong** **lúc** **điều** **khiển**. **Vì** **vậy**, **nhóm** **đã** **tìm** **kiếm** **một** **phần** **mềm** **có** **thể**

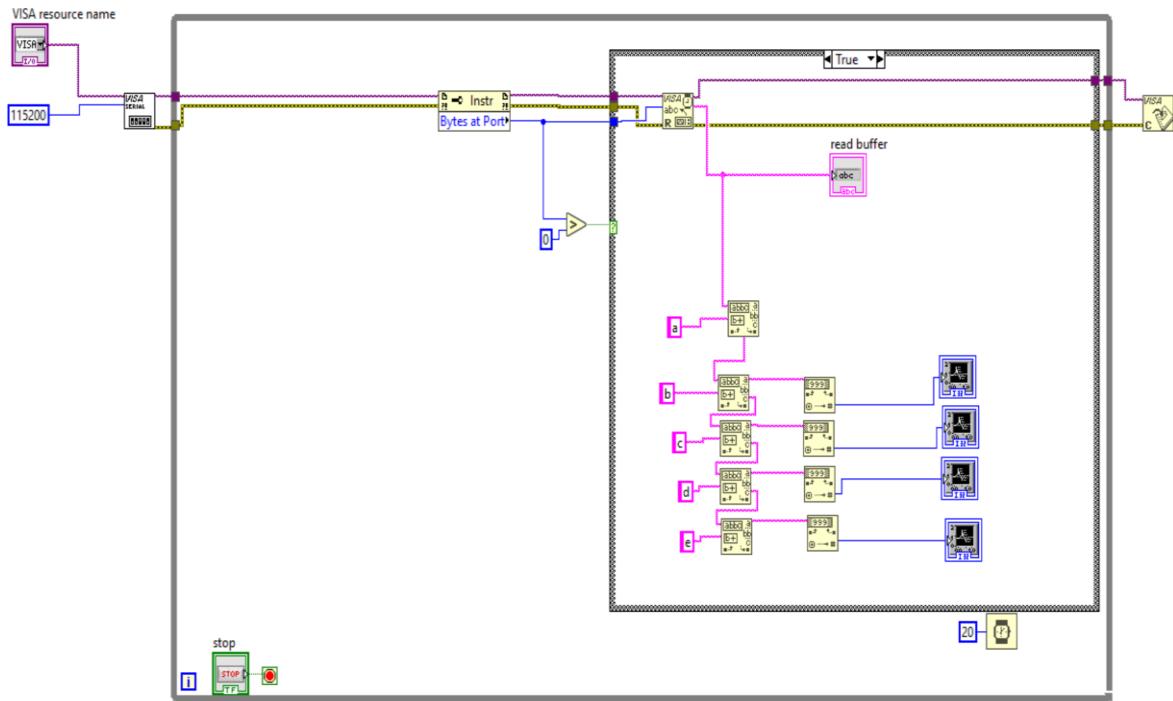
xuất đồ thị thông qua giao tiếp UART, đồ thị hiện thị trực quan, dễ tinh chỉnh, mỗi một đồ thị là một channel khác nhau. Xem xét những nhu cầu trên thì nhóm đã sử dụng LabView để hiện thị các thông số trong quá trình điều khiển. Ngoài ra, với Labview, nhóm có thể gửi tín hiệu từ máy tính không mà không cần phải tương tác vật lý đến bộ điều khiển trong thực tế.

Để điều khiển hệ thống trên máy tính, cần phải gửi một byte giá trị từ máy tính xuống vi điều khiển, sau khi nhận được byte điều khiển, bộ điều khiển sẽ bắt đầu hoạt động. Từ đó, các thông tin của hệ thống như vị trí con lắc, tốc độ quay con lắc, vị trí xe, tốc độ xe sẽ được gửi lên máy tính thông qua công giao tiếp UART. Tuy nhiên giá trị gửi lên sẽ được vi điều khiển tự động mã hóa thành dạng chuỗi, để hiển thị các giá trị này cần mã hóa ngược lại dạng số. Nhóm đã sử dụng kỹ thuật tách chuỗi qua phương pháp nhận dạng ký tự, chuỗi sau khi tách sẽ được mã hóa thành dạng số nhờ vào các khối chức năng có sẵn trên LabView.



Hình 6.12 Giao diện hiển thị trên phần mềm Labview

Hình 6.13 Khối xử lý dữ liệu gửi đến từ ArduinoHình 6.14 Giao diện hiển thị trên phần mềm Labview



Hình 6.15 Khối xử lý dữ liệu gửi đến từ Arduino

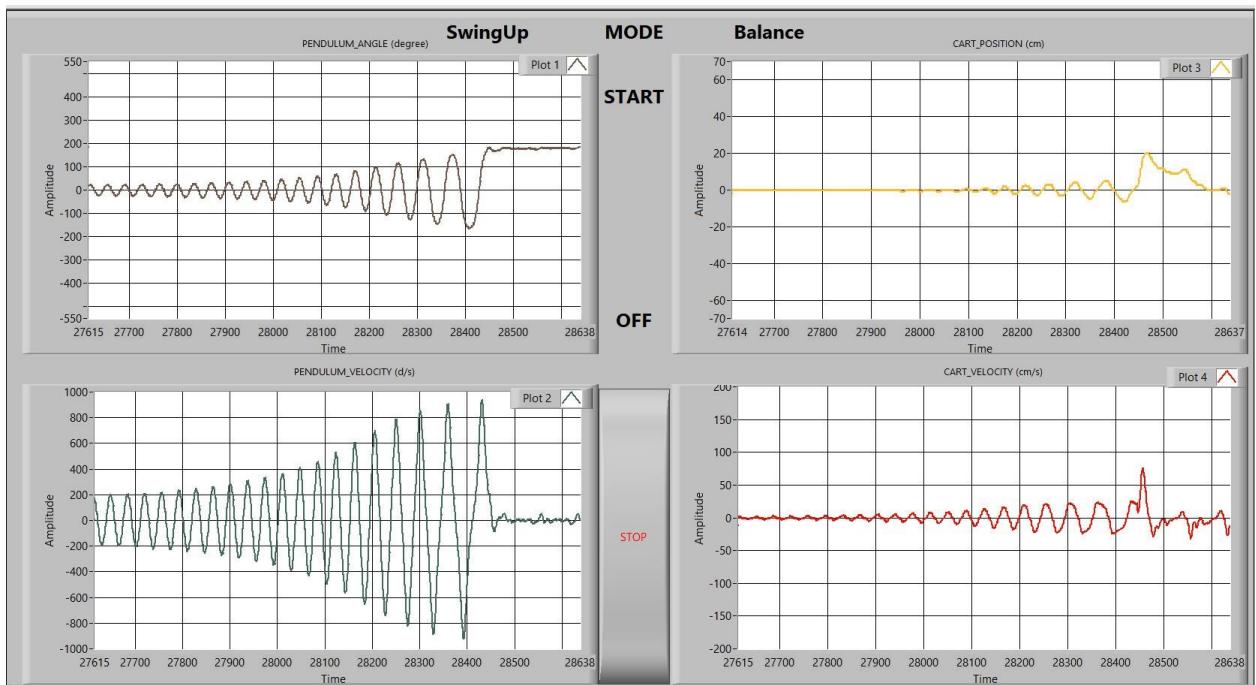
Hình 6.16 Kết quả hiển thị dữ liệu lên Labview trong quá trình điều khiển
Hình 6.17 Khối xử lý dữ liệu gửi đến từ Arduino

```

Serial.print(pend_angle_in_degree);
Serial.print("b");
Serial.print(pend_vel_in_degree_per_second);
Serial.print("c");
Serial.print(cart_position_in_centimeter);
Serial.print("d");
Serial.print(cart_vel_in_centimeter_per_second);
Serial.print("e");
}

```

Thông tin hiệu thị cần được giới hạn, mỗi lần gửi sẽ tốn thời gian xử lý của vi điều khiển và ảnh hưởng đến điều khiển của hệ thống, nên nhóm chỉ hiển thị 4 thông tin cần thiết nhất. Tuy nhiên, trong hệ thống này, do sử dụng phương pháp swing up bằng điều khiển vận tốc, nên khoảng thời gian delay (300ms) sẽ làm ảnh hưởng đến quá trình gửi dữ liệu lên máy tính, nên làm cho các độ thi trong quá trình swing up không hiển thị được. Quá trình gửi dữ liệu thực sự hiểu quả trong lúc cân bằng.



Hình 6.18 Kết quả hiển thị dữ liệu lên Labview trong quá trình điều khiển

6.7 Kết luận

Chương trình điều khiển đã đáp ứng được hết những yêu cầu đặt ra đối với bộ điều khiển như tốc độ đáp ứng, độ ổn định, khả năng xuất đồ thị trên thời gian thực.

Những ưu điểm:

- Tốc độ nạp chương trình nhanh, dễ dàng tinh chỉnh, giúp giảm thời gian tinh chỉnh bộ điều khiển.
- Tốc độ xử lý nhanh, đáp ứng được thời gian đọc Encoder và các phần tính toán cần thiết.
- Các hàm, thư viện được viết phù hợp cho nhiều chân cảm hoặc vi điều khiển với nhiều macros, class, nên dễ dàng ứng dụng chương trình trên các phần cứng và vi điều khiển khác.
- Động cơ êm ái, không gây tiếng ồn khó chịu.
- Dễ dàng đọc được trạng thái của hệ thống thông qua đồ thị trên SerialPlot.

Những nhược điểm:

- Vì giới hạn phần cứng nên đôi khi lúc Swing-up con lắc lên cao, lúc chuyển sang trạng thái cân bằng thì thanh trượt không đủ dài để xe đầy có thể cân bằng được trước khi xe trượt chạm vào cuối hành trình di chuyển.

- Thuật toán không khống chế được tốc độ con lắc khi nó gần đến vị trí cân bằng phía trên, dẫn đến đôi khi Swing-up con lắc, tốc độ của con lắc khi đi đến vị trí cao nhất vẫn còn cao, dẫn đến hệ thống không đáp ứng cân bằng kịp.

CHƯƠNG 7: KẾT LUẬN

7.1 Những kết quả đạt được

Qua quá trình nghiên cứu và thực hiện đề tài đã đạt được một số kết quả sau.

- Nhóm đã xây dựng thành công mô hình con lắc ngược đơn hoạt động ổn định, đạt thẩm mỹ.
- Hiểu rõ được thuật toán PID và ứng dụng vào điều khiển trên một hệ thống.
- Kỹ năng tư duy và giải quyết vấn đề trong lúc thực hiện đề tài.
- Nắm rõ mô hình động học con lắc để đưa vào mô phỏng trên Simulink và cho kết quả mô phỏng.
- Thiết kế được bộ điều khiển chạy thực tế bằng 2 cách: sử dụng Simulink nhúng xuống Arduino Mega để điều khiển hệ thống và lập trình C trên Visual Code Studio.
- Sử dụng nhiều phần mềm để hỗ trợ trong quá trình thiết phần cứng như mô phỏng mạch điện trên Proteus, thiết kế phần cứng trên Solidworks.
- Có khả năng sử dụng Labview giao tiếp với máy tính để hiển thị các thông tin cần thiết trong lúc hệ thống vận hành.

Các kết quả đạt được của đề tài sẽ là nền tảng tốt để nhóm có thể đi sâu vào các lĩnh vực điều khiển và hệ thống nhúng. Hi vọng mô hình sẽ được áp dụng cho việc giảng dạy các môn học liên quan đến điều khiển và ứng dụng trong các thí nghiệm áp dụng các thuật toán điều khiển khác cho con lắc ngược.

7.2 Hạn chế của đề tài

- Giải thuật Swing up chưa tối ưu, thời gian để con lắc lên vị trí cân bằng còn chậm.
- Trong quá trình cân bằng con lắc không thể đứng yên tại setpoint của hệ xe, mà dao động.
- Điểm setpoint bị trôi trong quá trình swing-up.
- Phương pháp tìm kiếm bộ PID chưa tối ưu, chủ yếu thử và sai.

7.3 Hướng phát triển của đề tài

- Phần cứng: xây dựng con lắc nữa (con lắc đã cho sẽ được gắn thêm 1 con lắc tự do nữa – hệ double inverted pendulum).

- Phần điều khiển: Thực hiện việc điều khiển con lắc giữ thăng bằng khi thanh ray nằm nghiêng, nghiên cứu giải thuật điều khiển swing-up cho phù hợp hơn. Áp dụng, kết hợp các phương pháp điều khiển khác như PD mờ.
- Có thể áp dụng công nghệ xử lý ảnh kết hợp với các giải thuật hiện đại khác áp dụng cho hệ thống con lắc ngược.

TÀI LIỆU THAM KHẢO

- [1] N. H. Mỹ, “Sử dụng thuật toán mờ noron điều khiển cân bằng con lắc ngược,” *Trường đại học Đà Nẵng*, 2011.
- [2] N. V. Đ. Hải, “Xây dựng bộ điều khiển nhúng tuyến tính hóa vào ra cho hệ xe con lắc ngược,” *Trường đại học Quốc gia TP.HCM*, 2011.
- [3] T. N. T. Dũng, “Nghiên cứu xây dựng hệ thống điều khiển con lắc ngược,” *Trường đại học hàng hải Việt Nam*, 2016.
- [4] T. N. T. Tân, “Thiết kế mô hình cân bằng con lắc ngược,” *Trường Đại học Trà Vinh*, 2017.
- [5] B. H. Lê, P. M. Nam và B. T. Lâm, “Điều khiển hệ con lắc ngược – xe sử dụng đại số gia tử,” *Tạp chí khoa học công nghệ*, 2019.
- [6] N. A. Vũ, “Điều khiển cân bằng hệ con lắc ngược,” *Trường đại học sư phạm kỹ thuật TP.HCM*, 2021.
- [7] “Inverted Pendulum: System Modeling,” [Trực tuyến]. Available: <https://s.net.vn/KREm>.
- [8] “PID là gì? Bộ điều khiển PID – Proportional Integral Derivative,” MESIDAS, [Trực tuyến]. Available: <https://mesidas.com/pid/>.
- [9] S. E. Oltean, “Swing-up and stabilization of the rotational inverted pendulum using PD and Fuzzy-PD controllers,” *The 7th International Conference Interdisciplinarity in Engineering*, pp. 57-64, 2014.
- [10] N. T. Tân, “Thiết kế mô hình cân bằng con lắc ngược,” Trường Đại học Trà Vinh, Trà Vinh, 2017.
- [11] Gopikrishnan, A. A. Kesarkar và N. Selvaganesan, “Design of Fractional Controller for Cart-Pendulum SIMO System,” *IEEE International Conference on Advanced Communication Control and Computing Technologies*, p. 170, 2012.
- [12] I. K. E. C. Fuat Peker và S. Atic, “Cascade Control Approach for a Cart Inverted Pendulum System Using Controller Synthesis Method,” *Mediterranean Conference on Control and Automation (MED)*, pp. 96-101, 2018.

- [13] “Sliding mode control,” Wikipedia, [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Sliding_mode_control.
- [14] T. Đ. K. Quốc, “Điều khiển mô hình con lắc ngược sử dụng bộ điều khiển LQR với hai vòng phản hồi,” *Trường Cao đẳng Công nghệ Tây Nguyên*, 2018.
- [15] “Arduino MEGA 2560 R3 Atmega16u2,” Nshop, [Trực tuyến]. Available: <https://nshopvn.com/product/arduino-mega2560-r3-atmega16u2/>.
- [16] “Mạch Cầu H là gì? Cấu tạo và nguyên lý hoạt động,” Thé giới điện curo, [Trực tuyến]. Available: <https://thegioidienco.vn/mach-cau-h.html>.
- [17] “LM2596HVS Mạch Giảm Áp 3A,” Thegioiic, [Trực tuyến]. Available: <https://www.thegioiic.com/lm2596hvs-mach-giam-ap-3a>.
- [18] “Mạch Giảm Áp 12A 100W,” Thegioiic, [Trực tuyến]. Available: <https://www.thegioiic.com/mach-giam-ap-12a-100w>.
- [19] “Module Relay 5V 10A,” Nshop, [Trực tuyến]. Available: <https://nshopvn.com/product/module-relay-5v-10a/>.
- [20] “E6B2-CWZ5B Rotary Encoder 1000 Xung PNP,” Thegioiic, [Trực tuyến]. Available: <https://www.thegioiic.com/e6b2-cwz5b-rotary-encoder-1000-xung-pnp>.
- [21] “Công tắc hành trình Omron V-156-1C25,” Icdayroi, [Trực tuyến]. Available: <https://icdayroi.com/cong-tac-hanh-trinh-omron-v-156-1c25>.
- [22] “Nút dừng khẩn cấp 22MM LA38-11ZS (ST0P),” Icdayroi, [Trực tuyến]. Available: <https://icdayroi.com/nut-dung-khan-cap-22mm-la38-11zs-st0p>.
- [23] “Nguồn тюнинг 24V 10A,” Nshop, [Trực tuyến]. Available: <https://nshopvn.com/product/nguon-to-ong-24v-10a/>.
- [24] “Minertia motor R series bulletin,” Yaskawa.
- [25] D. H. Đấu, “Vật Rắn,” [Trực tuyến]. Available: <https://s.net.vn/1kOT>.
- [26] “The Benefits of Cascade Control,” WATLOW, 22 September 2020. [Trực tuyến]. Available: <https://s.net.vn/s6BJ>.

- [27] “Điều khiển Logic mờ (Fuzzy Logic),” Bao Anh Automation, 2020. [Trực tuyến]. Available: <https://s.net.vn/KWEo>.
- [28] “Điều khiển mạng nơ-ron và các ứng dụng của điều khiển mạng nơ-ron,” Bao An Automation, 2020. [Trực tuyến]. Available: <https://s.net.vn/oRhL>.
- [29] “Giải thuật di truyền,” Wikipedia, [Trực tuyến]. Available: <https://s.net.vn/ruCl>.
- [30] “Điều khiển tối ưu,” Wikipedia, [Trực tuyến]. Available: <https://s.net.vn/WJSU>.
- [31] “Proportional–integral–derivative controller,” Wikipedia, [Trực tuyến]. Available: <https://s.net.vn/ruCl>.
- [32] “Giới thiệu về Arduino Mega 2560 sử dụng trong chế tạo máy in 3D,” Đại học Điện Lực, [Trực tuyến]. Available: <https://s.net.vn/ay00>.
- [33] “What Is Arduino Programming?,” MathWorks, [Trực tuyến]. Available: <https://s.net.vn/rLBt>.
- [34] “BTS7960 Module Điều Khiển Động Cơ DC 43A,” Thegioiic, [Trực tuyến]. Available: <https://s.net.vn/ud1T>.
- [35] L. Q. Vũ, N. M. Tâm và D. H. Nghĩa, “ĐIỀU KHIỂN TRƯỢT HỆ CON LẮC NGƯỢC QUAY SLIDING MODE CONTROL FOR ROTARY INVERTED PENDULUM,” *Tạp Chí Khoa Học Giáo Dục Kỹ Thuật (34/2015) Trường Đại Học Sư Phạm Kỹ Thuật TP. Hồ Chí Minh*, pp. 24-29, 2015.
- [36] “Wheeled inverted pendulum model,” 2014. [Trực tuyến]. Available: <https://scaron.info/robotics/wheeled-inverted-pendulum-model.html>.
- [37] N. H. Mỹ, “Sử dụng thuật toán mờ nơron điều khiển cân bằng con lắc ngược,” *Trường đại học Đà Nẵng*, 2011.

BẢNG PHÂN CÔNG VÀ TIẾN ĐỘ CÔNG VIỆC

Nội dung công việc	Chi tiết công việc	Người thực hiện	Tuần 5	Tuần 6	Tuần 7	Tuần 8	Tuần 9	Tuần 10	Tuần 11	Tuần 12
Tổng quan đề tài	Tên đề tài	Hiếu, Thịnh								
	Lý do chọn đề tài	Hiếu, Thịnh								
	Giới hạn và phạm vi đề tài	Hiếu, Thịnh								
Xây dựng cơ sở lý thuyết	Khảo sát đề tài và đề ra phương hướng thực hiện	Tuân, Bảo								
	Nguyên lí hoạt động của hệ thống	Tuân, Thịnh								
	Tìm hiểu các thông số kỹ thuật	Tuân, Hiếu								
Xây dựng phần cứng	Thiết kế 3D, Vẽ Proteus Gia công thiết bị	Tuân								
Thiết kế mạch điện	Thi công sơ đồ mạch điện	Tuân								
Xây dựng hàm truyền	Mô hình hóa hệ thống và tìm hàm truyền	Thịnh, Hiếu								

Mô phỏng hệ thống	Viết chương trình trên matlab-simulink	Thịnh, Hiếu							
Giao tiếp Arduino với Matlab	Tìm hiểu cách đồ chương trình Matlab xuống Arduino	Tuấn							
Xây dựng thuật toán	Điều khiển cân bằng trên Arduino IDE	Bảo							
Xây dựng mô hình-thực nghiệm	Chế tạo sản phẩm bằng các nguyên vật liệu phù hợp	Tuấn, Bảo, Thịnh, Hiếu							
Thực nghiệm	Cho hoạt động thử mô hình	Tuấn, Bảo							
Kiểm tra và chỉnh sửa	Kiểm tra	Tuấn, Bảo, Thịnh, Hiếu							
Chuẩn bị báo cáo	Viết báo cáo, thuyết trình	Tuấn, Bảo, Thịnh, Hiếu							