

**ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**Nguyễn Vũ Quang**

**XÂY DỰNG KHUNG ĐỂ ROBOT CÓ KHẢ NĂNG CÂN  
BẰNG TRỌNG TÂM**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Kỹ thuật điều khiển và tự động hóa**

**HÀ NỘI – 2025**

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Nguyễn Vũ Quang

**XÂY DỰNG KHUNG ĐỂ ROBOT CÓ KHẢ NĂNG CÂN  
BẰNG TRỌNG TÂM**

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Kỹ thuật điều khiển và tự động hóa**

**Cán bộ hướng dẫn: ThS. Đặng Anh Việt**

**HÀ NỘI – 2025**

## LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi dưới sự hướng dẫn của ThS. Đặng Anh Việt. Các số liệu, kết quả nêu trong khóa luận là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

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

Hà Nội, ngày                      tháng                      năm 2025

**Sinh viên thực hiện**

**Nguyễn Vũ Quang**

## LỜI CẢM ƠN

Để hoàn thành khóa luận tốt nghiệp này, tôi đã nhận được rất nhiều sự giúp đỡ và hỗ trợ từ thầy cô, gia đình và bạn bè.

Trước hết, tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất đến ThS. Đặng Anh Việt – người đã trực tiếp hướng dẫn, chỉ bảo tận tình và tạo mọi điều kiện thuận lợi cho tôi trong suốt quá trình thực hiện khóa luận.

Tôi xin chân thành cảm ơn các thầy cô giáo trong Khoa Cơ học kỹ thuật và Tự động hóa, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội đã trang bị cho tôi những kiến thức quý báu trong suốt thời gian học tập tại trường.

Cuối cùng, tôi xin gửi lời cảm ơn đến gia đình, bạn bè đã luôn động viên, khích lệ tôi trong suốt quá trình học tập và thực hiện khóa luận.

Hà Nội, tháng 6 năm 2025

**Sinh viên**

**Nguyễn Vũ Quang**

## TÓM TẮT

**Tóm tắt:** Khóa luận trình bày quá trình nghiên cứu, thiết kế và chế tạo khung đế robot di động có khả năng tự động cân bằng trọng tâm. Hệ thống sử dụng cảm biến lực (Loadcell) kết hợp với module HX711 để đo chênh lệch trọng lượng giữa hai bên khung. Thuật toán điều khiển PID được áp dụng để điều khiển động cơ bước di chuyển khối đối trọng trên cơ cấu vítme, từ đó bù đắp sự mất cân bằng khi tải thay đổi.

Kết quả thực nghiệm cho thấy hệ thống có khả năng phát hiện và bù đắp độ lệch trọng tâm trong phạm vi  $\pm 100g$  với thời gian đáp ứng nhanh. Chương trình điều khiển được viết theo kiến trúc Non-blocking, đảm bảo động cơ bước hoạt động mượt mà ở tốc độ cao mà không bị ảnh hưởng bởi các tác vụ đọc cảm biến hay giao tiếp Serial.

Khóa luận cũng đề xuất các hướng phát triển mở rộng như điều khiển vị trí, tốc độ chuyển dịch và tích hợp vào robot di động thực tế.

**Từ khóa:** *Cân bằng trọng tâm, Loadcell, PID, Động cơ bước, Arduino, Non-blocking, Robot di động.*

# Mục lục

<b>Chương 1. Tổng quan</b>	1
1.1. Đặt vấn đề và lý do chọn đề tài	1
1.2. Mục tiêu nghiên cứu	2
1.2.1. Mục tiêu về nghiên cứu lý thuyết	2
1.2.2. Mục tiêu về thiết kế và chế tạo	2
1.2.3. Mục tiêu về phần mềm điều khiển	2
1.2.4. Mục tiêu về thử nghiệm và đánh giá	3
1.3. Đối tượng và phạm vi nghiên cứu	3
1.3.1. Đối tượng nghiên cứu	3
1.3.2. Phạm vi nghiên cứu	4
1.4. Phương pháp nghiên cứu	4
1.4.1. Phương pháp nghiên cứu lý thuyết	4
1.4.2. Phương pháp mô hình hóa và thiết kế	4
1.4.3. Phương pháp thực nghiệm	5
1.4.4. Phương pháp phân tích và đánh giá	5
1.5. Ý nghĩa khoa học và thực tiễn	5
1.5.1. Ý nghĩa khoa học	5
1.5.2. Ý nghĩa thực tiễn	6
1.6. Bố cục khóa luận	6
<b>Chương 2. Cơ sở lý thuyết</b>	7
2.1. So sánh các phương pháp cân bằng robot	7
2.1.1. Các phương pháp cân bằng chính	7
2.1.2. Bảng so sánh tổng hợp	8
2.1.3. Phân tích ưu nhược điểm chi tiết	8
2.1.4. Lý do lựa chọn phương pháp khối lượng di động	9
2.2. Lý thuyết cân bằng trọng tâm và momen lực	10
2.2.1. Khái niệm trọng tâm	10
2.2.2. Momen lực và điều kiện cân bằng tĩnh	10

2.2.3. Áp dụng nguyên lý momen vào bài toán cân bằng khung robot .....	10
2.3. Cảm biến lực Loadcell và module HX711 .....	12
2.3.1. Cấu tạo và nguyên lý hoạt động của Loadcell .....	12
2.3.2. Loadcell 3 dây và cách ghép thành cầu đầy đủ .....	12
2.3.3. Module HX711 và giao thức truyền thông .....	13
2.3.4. Thư viện HX711_ADC và cơ chế Non-blocking .....	13
2.4. Động cơ bước và driver TB6600 .....	14
2.4.1. Nguyên lý hoạt động của động cơ bước .....	14
2.4.2. Chế độ vi bước (Microstepping) và ảnh hưởng đến hiệu năng .....	14
2.4.3. Driver TB6600 và cấu hình điều khiển.....	15
2.4.4. Thư viện AccelStepper và cơ chế Polling.....	16
2.5. Vi điều khiển Arduino .....	16
2.5.1. Kiến trúc phần cứng ATmega328P .....	17
2.5.2. Mô hình lập trình Arduino.....	17
2.5.3. Vấn đề Blocking I/O trong giao tiếp Serial .....	17
2.6. Thuật toán điều khiển PID.....	19
2.6.1. Mô hình toán học của bộ điều khiển PID .....	19
2.6.2. Vai trò của từng thành phần .....	19
2.6.3. Dạng rời rạc của PID cho hệ thống số.....	20
2.6.4. Phương pháp tinh chỉnh tham số PID.....	20
2.7. Lập trình thời gian thực cho hệ thống nhúng .....	20
2.7.1. Khái niệm hệ thống thời gian thực .....	21
2.7.2. Blocking vs Non-blocking I/O .....	21
2.7.3. Cơ chế Time-Slicing với millis() .....	21
2.7.4. Vấn đề Pulse Starvation trong điều khiển động cơ bước.....	22
2.7.5. Kỹ thuật Deadzone và Hysteresis trong điều khiển .....	22
<b>Chương 3. Thiết kế hệ thống</b> .....	<b>24</b>
3.1. Phân tích yêu cầu thiết kế .....	24
3.1.1. Yêu cầu chức năng .....	24
3.1.2. Yêu cầu phi chức năng .....	24

3.1.3. Ràng buộc thiết kế.....	24
3.2. Thiết kế cơ khí .....	25
3.2.1. Tổng quan cấu trúc cơ khí.....	25
3.2.2. Cơ cấu vítme - thanh trượt .....	25
3.2.3. Bố trí cảm biến Loadcell.....	26
3.2.4. Giới hạn hành trình.....	26
3.3. Thiết kế mạch điện .....	26
3.3.1. Sơ đồ khối hệ thống.....	26
3.3.2. Danh sách linh kiện điện tử .....	27
3.3.3. Sơ đồ kết nối chi tiết.....	27
3.3.4. Cấu hình Driver TB6600 .....	28
3.3.5. Sơ đồ nguyên lý mạch điện .....	28
3.3.6. Thiết kế nguồn điện.....	29
3.3.7. Xử lý nhiễu điện từ.....	29
3.4. Thiết kế phần mềm.....	29
3.4.1. Kiến trúc phần mềm tổng quan.....	30
3.4.2. Quy trình Homing .....	30
3.4.3. Logic điều khiển PID với Deadzone và Hysteresis.....	32
3.4.4. Xử lý giới hạn hành trình mềm.....	34
3.4.5. Các tham số cấu hình .....	34
3.4.6. Quy trình hiệu chuẩn Loadcell .....	34
<b>Chương 4. Thực nghiệm và đánh giá .....</b>	<b>36</b>
4.1. Triển khai hệ thống.....	36
4.1.1. Môi trường thử nghiệm .....	36
4.1.2. Các tham số cấu hình thử nghiệm .....	36
4.2. Kiểm tra độ chính xác cảm biến Loadcell .....	37
4.2.1. Phương pháp kiểm tra .....	37
4.2.2. Kết quả đo.....	37
4.3. Thử nghiệm so sánh các giá trị hệ số $K_p$ .....	37
4.3.1. Kết quả chi tiết từng giá trị $K_p$ .....	37



4.3.2. Bảng so sánh tổng hợp .....	38
4.3.3. Biểu đồ so sánh thời gian xác lập.....	38
4.3.4. Biểu đồ diễn biến độ lệch theo thời gian.....	39
4.4. Thử nghiệm tổng hợp với nhiều kích bản tải .....	40
4.4.1. Kích bản thử nghiệm .....	40
4.4.2. Biểu đồ diễn biến thử nghiệm tổng hợp .....	40
4.4.3. Phân tích kết quả.....	41
4.5. Phân tích hiệu năng hệ thống .....	41
4.5.1. Hiệu năng bộ điều khiển PID .....	41
4.5.2. Hiệu quả cơ chế Deadzone và Hysteresis.....	42
4.5.3. Hiệu quả thiết kế Non-blocking.....	42
4.6. Đánh giá tổng thể.....	43
4.6.1. So sánh với mục tiêu đề ra .....	43
<b>Chương 5. Kết luận và hướng phát triển .....</b>	<b>44</b>
5.1. Kết luận .....	44
5.1.1. Về mặt lý thuyết .....	44
5.1.2. Về mặt thực tiễn .....	44
5.1.3. Đóng góp của đề tài.....	44
5.2. Hạn chế .....	45
5.3. Hướng phát triển.....	45
5.3.1. Cải tiến phần cứng .....	45
5.3.2. Cải tiến phần mềm .....	46
5.3.3. Tích hợp và ứng dụng .....	46
5.4. Lời kết.....	46
<b>Phụ lục A. Mã nguồn chương trình điều khiển .....</b>	<b>48</b>

## Danh sách hình vẽ

3.1. Sơ đồ khối hệ thống điện tử .....	27
3.2. Sơ đồ nguyên lý mạch điện hệ thống.....	29
3.3. Lưu đồ quy trình Homing .....	31
3.4. Lưu đồ thuật toán điều khiển PID .....	33
4.2 So sánh thời gian xác lập với các giá trị $K_p$ khác nhau .....	39
4.2. ....	39
4.4 Đường cong đáp ứng – Diễn biến độ lệch theo thời gian với các giá trị $K_p$ .....	39
4.4. ....	39
4.6 Diễn biến độ lệch trong thử nghiệm tổng hợp .....	41
4.6. ....	41
4.8 Sơ đồ cơ chế Hysteresis với ngưỡng bật $\pm 110g$ và ngưỡng tắt $\pm 90g$ .....	42
4.8. ....	42

# Danh sách bảng

2.1. So sánh các phương pháp cân bằng robot .....	8
2.2. ....	11
2.3 Khả năng cân bằng theo khối lượng đối trọng .....	11
3.1. Thông số khung nhôm định hình .....	25
3.2. Thông số vitme .....	25
3.3. Danh sách linh kiện điện tử.....	27
3.4. Kết nối Arduino với các module HX711 .....	28
3.5. Kết nối Arduino với driver TB6600 .....	28
3.6. Bảng tổng hợp các tham số cấu hình phần mềm.....	34
4.1. ....	36
4.2 Tham số cấu hình hệ thống trong thử nghiệm.....	36
4.3. ....	37
4.4 Kết quả kiểm tra độ chính xác cảm biến với vật chuẩn 210g.....	37
4.5. ....	38
4.6 So sánh hiệu năng với các giá trị $K_p$ khác nhau .....	38
4.7. ....	40
4.8 Timeline thử nghiệm tổng hợp với nhiều kịch bản tải .....	40
4.9. ....	42
4.10 Tổng hợp hiệu năng bộ điều khiển PID .....	42
4.11. ....	43
4.12 So sánh hiệu năng Non-blocking và Blocking .....	43
4.13. ....	43
4.14 So sánh kết quả với mục tiêu đề ra .....	43
5.1. ....	44
5.2 Tổng hợp kết quả định lượng từ thực nghiệm.....	44

## Danh sách các từ viết tắt

**PID:** Proportional-Integral-Derivative – Bộ điều khiển vi tích phân tỷ lệ.

**PWM:** Pulse Width Modulation – Điều chế độ rộng xung.

**ADC:** Analog to Digital Converter – Bộ chuyển đổi tương tự sang số.

**I2C:** Inter-Integrated Circuit – Giao thức truyền thông nối tiếp.

**SPI:** Serial Peripheral Interface – Giao diện ngoại vi nối tiếp.

**UART:** Universal Asynchronous Receiver-Transmitter – Bộ thu phát không đồng bộ.

**GPIO:** General Purpose Input/Output – Chân vào/ra đa mục đích.

**RPM:** Revolutions Per Minute – Vòng quay trên phút.

**ISR:** Interrupt Service Routine – Trình phục vụ ngắt.

# Chương 1.

## Tổng quan

### 1.1. Đặt vấn đề và lý do chọn đề tài

Trong những năm gần đây, robot di động (Mobile Robot) đã trở thành một trong những lĩnh vực nghiên cứu và ứng dụng phát triển mạnh mẽ nhất trong ngành tự động hóa. Từ các robot vận chuyển hàng hóa trong nhà máy, robot phục vụ trong nhà hàng, đến các robot thám hiểm địa hình phức tạp – tất cả đều đòi hỏi khả năng di chuyển ổn định và thích ứng với các điều kiện tải trọng thay đổi.

Một thách thức kỹ thuật quan trọng trong thiết kế robot di động là vấn đề **cân bằng trọng tâm**. Khi robot mang theo tải trọng hoặc được trang bị cánh tay thao tác (manipulator), trọng tâm của hệ thống sẽ thay đổi theo vị trí và khối lượng của tải. Sự dịch chuyển trọng tâm này gây ra nhiều hệ quả tiêu cực:

- **Giảm ổn định động học:** Robot dễ bị lật hoặc mất cân bằng khi di chuyển trên địa hình không bằng phẳng, đặc biệt khi tải lệch về một phía.
- **Tăng tải không đều lên các bánh xe:** Dẫn đến mài mòn không đồng đều, giảm tuổi thọ cơ cấu truyền động và ảnh hưởng đến độ chính xác điều khiển quỹ đạo.
- **Tăng tiêu hao năng lượng:** Các động cơ dẫn động phải bù đắp momen do tải lệch gây ra, làm giảm hiệu suất và thời gian hoạt động của robot.
- **Ảnh hưởng đến độ chính xác thao tác:** Với robot có cánh tay, sự mất cân bằng của đế ảnh hưởng trực tiếp đến độ chính xác định vị của end-effector.

Giải pháp truyền thống cho vấn đề này thường là thiết kế đế robot với trọng tâm thấp và phân bố tải đối xứng. Tuy nhiên, cách tiếp cận này không linh hoạt khi tải trọng thay đổi trong quá trình vận hành. Một giải pháp tiên tiến hơn là sử dụng **hệ thống cân bằng trọng tâm chủ động** (Active Center of Gravity Balancing System), trong đó một cơ cấu chấp hành sẽ tự động di chuyển khối đối trọng để bù đắp sự thay đổi trọng tâm.

Hệ thống cân bằng trọng tâm chủ động đòi hỏi sự kết hợp của nhiều thành phần: cảm biến đo lường độ lệch, cơ cấu chấp hành di chuyển đối trọng, và thuật toán điều khiển để xác định vị trí đối trọng tối ưu. Đây là một bài toán điều khiển vòng kín điển hình, phù hợp để áp dụng các kiến thức về **lý thuyết điều khiển tự động, cảm biến và đo lường**, cũng như **hệ thống nhúng thời gian thực**.

Xuất phát từ nhu cầu thực tiễn và tính ứng dụng cao của vấn đề, đề tài “*Xây dựng khung đế robot có khả năng cân bằng trọng tâm*” được lựa chọn làm khóa luận tốt nghiệp. Đề tài hướng đến việc nghiên cứu, thiết kế và chế tạo một khung đế có khả năng tự động phát hiện và bù đắp độ lệch trọng tâm, tạo nền tảng cho việc phát triển các robot di động ổn định hơn trong tương lai.

## **1.2. Mục tiêu nghiên cứu**

Mục tiêu tổng quát của khóa luận là thiết kế và chế tạo một khung đế robot di động có khả năng tự động cân bằng trọng tâm khi tải trọng thay đổi. Để đạt được mục tiêu này, các mục tiêu cụ thể được xác định như sau:

### **1.2.1. Mục tiêu về nghiên cứu lý thuyết**

Nghiên cứu nguyên lý cân bằng trọng tâm dựa trên lý thuyết momen lực và điều kiện cân bằng tĩnh. Tìm hiểu các phương pháp đo lường độ lệch trọng tâm, trong đó tập trung vào việc sử dụng cảm biến lực (Loadcell) để xác định sự chênh lệch tải trọng giữa các điểm tựa.

Nghiên cứu thuật toán điều khiển PID (Proportional-Integral-Derivative) và phương pháp tinh chỉnh tham số phù hợp với đặc tính của hệ thống cơ điện tử. Đặc biệt, nghiên cứu các kỹ thuật xử lý vùng chết (Deadzone), độ trễ (Hysteresis) để tránh hiện tượng dao động quanh điểm cân bằng.

Nghiên cứu kiến trúc lập trình thời gian thực cho hệ thống nhúng, đảm bảo khả năng đáp ứng nhanh và ổn định của hệ thống điều khiển.

### **1.2.2. Mục tiêu về thiết kế và chế tạo**

Thiết kế khung cơ khí có kích thước phù hợp với robot di động cỡ nhỏ và vừa (khoảng 35×35cm), đảm bảo độ cứng vững và khả năng tích hợp các thành phần điện tử.

Thiết kế cơ cấu di chuyển đối trọng sử dụng động cơ bước kết hợp truyền động vítme, đảm bảo độ chính xác vị trí và khả năng chịu tải.

Thiết kế mạch điện tử điều khiển bao gồm: mạch đọc tín hiệu từ cảm biến lực, mạch điều khiển động cơ bước, và giao tiếp với vi điều khiển.

Chế tạo và lắp ráp hoàn chỉnh mô hình khung đế cân bằng.

### 1.2.3. Mục tiêu về phần mềm điều khiển

Xây dựng chương trình điều khiển trên nền tảng vi điều khiển Arduino, tích hợp thuật toán PID để tự động điều chỉnh vị trí đối trọng.

Đảm bảo chương trình hoạt động theo kiến trúc Non-blocking (không chặn), cho phép động cơ bước vận hành mượt mà ở tốc độ cao mà không bị ảnh hưởng bởi các tác vụ đọc cảm biến hay giao tiếp.

Phát triển giao diện giám sát thông qua Serial Monitor để theo dõi các thông số hoạt động và hỗ trợ quá trình tinh chỉnh hệ thống.

### 1.2.4. Mục tiêu về thử nghiệm và đánh giá

Tiến hành thử nghiệm hệ thống với các kịch bản tải trọng khác nhau. Đánh giá các chỉ tiêu: thời gian đáp ứng, độ chính xác cân bằng, độ ổn định, và phạm vi tải trọng có thể bù đắp.

So sánh kết quả thực nghiệm với mục tiêu thiết kế, phân tích các hạn chế và đề xuất hướng cải tiến.

## 1.3. Đối tượng và phạm vi nghiên cứu

### 1.3.1. Đối tượng nghiên cứu

Đối tượng nghiên cứu của khóa luận bao gồm các thành phần chính của hệ thống cân bằng trọng tâm:

**Cảm biến lực Loadcell và module HX711:** Loadcell là cảm biến đo lực dựa trên nguyên lý điện trở biến dạng. Module HX711 là bộ chuyển đổi ADC 24-bit chuyên dụng cho loadcell, có khả năng đọc tín hiệu với độ phân giải cao. Trong dự án này, thư viện HX711\_ADC được sử dụng với ưu điểm hỗ trợ chế độ đọc Non-blocking, cho phép vi điều khiển thực hiện các tác vụ khác trong khi chờ dữ liệu từ cảm biến.

**Động cơ bước và driver TB6600:** Động cơ bước Nema 17 được sử dụng làm cơ cấu chấp hành, với đặc điểm điều khiển vị trí chính xác theo vòng hở. Driver TB6600 hỗ trợ điều khiển vi bước (microstepping) với các mức 1/2, 1/4, 1/8, 1/16, cho phép tăng độ phân giải và giảm rung động.

**Vi điều khiển Arduino:** Nền tảng Arduino (chip ATmega328P, tần số 16MHz) được chọn làm bộ xử lý trung tâm do tính phổ biến, dễ lập trình, và có nhiều thư viện hỗ trợ. Thư viện AccelStepper được sử dụng để điều khiển động cơ bước với khả năng điều chỉnh tốc độ và gia tốc mượt mà.

**Thuật toán điều khiển PID:** Bộ điều khiển PID là thuật toán điều khiển vòng kín phổ biến trong công nghiệp, phù hợp cho các hệ thống yêu cầu độ chính xác và ổn định cao. Thư viện PID\_v1 cho Arduino được sử dụng để triển khai thuật toán.

**Kỹ thuật lập trình thời gian thực:** Nghiên cứu các vấn đề về độ trễ (latency), blocking I/O, và các giải pháp Non-blocking để đảm bảo hệ thống đáp ứng các ràng buộc thời gian thực.

### 1.3.2. Phạm vi nghiên cứu

Khóa luận tập trung vào các giới hạn sau:

**Về không gian:** Hệ thống cân bằng trọng tâm theo **một trục** (trái-phải). Việc mở rộng sang hai trục được đề cập như hướng phát triển.

**Về kích thước:** Khung đế có kích thước  $35 \times 35$ cm, phù hợp với các robot di động cỡ nhỏ và vừa. Hành trình di chuyển đối trọng: 120mm về bên trái, 110mm về bên phải (tính từ vị trí tâm).

**Về tải trọng:** Cảm biến loadcell có tải trọng định mức 50kg mỗi cụm. Khối lượng đối trọng được tính toán dựa trên nguyên lý momen lực với hệ số 0.69, cho phép cân bằng độ lệch lên đến 600g (với đối trọng khoảng 870g).

**Về điều khiển:** Sử dụng điều khiển vòng hở cho động cơ bước (không có encoder phản hồi vị trí). Thuật toán PID với vùng chết  $\pm 100$ g – tức là hệ thống chấp nhận độ lệch trong phạm vi này mà không cần điều chỉnh.

**Về hiệu năng:** Tốc độ tối đa của động cơ bước bị giới hạn bởi khả năng xử lý của vi điều khiển và đặc tính của thư viện AccelStepper (khoảng 15000 bước/giây trong điều kiện tối ưu).

## 1.4. Phương pháp nghiên cứu

Khóa luận sử dụng kết hợp các phương pháp nghiên cứu sau:

### 1.4.1. Phương pháp nghiên cứu lý thuyết

Tham khảo các tài liệu về cơ học (momen lực, điều kiện cân bằng) [7], lý thuyết điều khiển PID [7], nguyên lý cảm biến loadcell [5], và lập trình hệ thống nhúng [6].

Tổng hợp kiến thức từ các nghiên cứu trước đó về điều khiển động cơ bước, đặc biệt là các vấn đề về hiệu năng thời gian thực khi kết hợp nhiều tác vụ (đọc cảm biến, điều khiển động cơ, giao tiếp Serial).



### **1.4.2. Phương pháp mô hình hóa và thiết kế**

Xây dựng mô hình toán học của hệ thống cân bằng dựa trên nguyên lý momen lực. Từ đó xác định mối quan hệ giữa độ lệch tải trọng và vị trí đối trọng cần thiết để cân bằng.

Thiết kế cơ khí sử dụng phương pháp thiết kế mô-đun, cho phép dễ dàng điều chỉnh và thay thế các thành phần. Thiết kế mạch điện theo sơ đồ khối, xác định rõ chức năng và giao tiếp của từng module.

Thiết kế phần mềm theo kiến trúc phân tầng: tầng phần cứng (Hardware Abstraction Layer), tầng điều khiển (Control Layer), và tầng ứng dụng (Application Layer).

### **1.4.3. Phương pháp thực nghiệm**

Chế tạo mô hình thực tế dựa trên thiết kế đã xây dựng. Tiến hành hiệu chuẩn (calibration) các cảm biến loadcell để đảm bảo độ chính xác đo lường.

Thực hiện các bài test với nhiều kịch bản tải trọng khác nhau: tải tĩnh (đặt vật nặng cố định), tải động (di chuyển vật nặng trong quá trình hoạt động), và tải biến thiên (thay đổi khối lượng tải).

Thu thập dữ liệu thông qua Serial Monitor, bao gồm: giá trị đọc từ loadcell, sai số (Input), tín hiệu điều khiển (Output), và tốc độ động cơ. Phân tích dữ liệu để đánh giá hiệu quả của thuật toán điều khiển.

### **1.4.4. Phương pháp phân tích và đánh giá**

So sánh kết quả thực nghiệm với các chỉ tiêu thiết kế ban đầu. Phân tích nguyên nhân của các sai lệch (nếu có) và đề xuất giải pháp khắc phục.

Đánh giá định lượng các thông số: thời gian đáp ứng (từ khi có tải lệch đến khi đạt cân bằng), độ chính xác cân bằng (sai số còn lại sau khi ổn định), và phạm vi hoạt động (độ lệch tối đa có thể bù đắp).

Phân tích các yếu tố ảnh hưởng đến hiệu năng hệ thống, bao gồm: tham số PID, tốc độ lấy mẫu, độ phân giải microstep, và các vấn đề về lập trình thời gian thực.

## **1.5. Ý nghĩa khoa học và thực tiễn**

### **1.5.1. Ý nghĩa khoa học**

Khóa luận đóng góp vào việc nghiên cứu ứng dụng lý thuyết điều khiển tự động trong bài toán cân bằng trọng tâm cho robot di động. Các kết quả nghiên cứu về ảnh

hưởng của kiến trúc phần mềm (blocking vs non-blocking) đến hiệu năng điều khiển động cơ bước có giá trị tham khảo cho các nghiên cứu liên quan.

Việc phân tích chi tiết các vấn đề thời gian thực trong hệ thống nhúng – từ độ trễ của giao tiếp Serial đến ảnh hưởng của tần số lấy mẫu – cung cấp cơ sở lý thuyết và thực nghiệm cho việc thiết kế các hệ thống điều khiển tương tự.

### 1.5.2. Ý nghĩa thực tiễn

Sản phẩm của khóa luận là một khung để robot có khả năng tự cân bằng, có thể được tích hợp vào các robot di động thực tế để cải thiện độ ổn định khi vận hành.

Các kinh nghiệm thiết kế và chế tạo (lựa chọn linh kiện, giải quyết các vấn đề kỹ thuật, tinh chỉnh hệ thống) được tài liệu hóa trong khóa luận, có thể làm tài liệu tham khảo cho các dự án tương tự.

Mã nguồn chương trình điều khiển được cung cấp đầy đủ trong phụ lục, có thể được sử dụng lại hoặc phát triển thêm cho các ứng dụng khác.

## 1.6. Bố cục khóa luận

Khóa luận được trình bày trong 5 chương với nội dung như sau:

**Chương 1: Tổng quan** – Trình bày bối cảnh và lý do chọn đề tài, xác định mục tiêu nghiên cứu, đối tượng và phạm vi nghiên cứu, các phương pháp nghiên cứu được sử dụng, cũng như ý nghĩa khoa học và thực tiễn của đề tài.

**Chương 2: Cơ sở lý thuyết** – Trình bày các kiến thức nền tảng về nguyên lý cân bằng trọng tâm và momen lực, cảm biến lực Loadcell và module HX711, động cơ bước và kỹ thuật vi bước, vi điều khiển Arduino, thuật toán điều khiển PID, và các vấn đề lập trình thời gian thực cho hệ thống nhúng.

**Chương 3: Thiết kế hệ thống** – Mô tả chi tiết quá trình thiết kế gồm: phân tích yêu cầu, thiết kế cơ khí (khung, cơ cấu vítme-thanh trượt, bố trí loadcell), thiết kế mạch điện (sơ đồ khối, kết nối phần cứng), và thiết kế phần mềm (cấu trúc chương trình, quy trình Homing, logic điều khiển PID với Deadzone và Hysteresis).

**Chương 4: Thực nghiệm và đánh giá** – Trình bày môi trường và điều kiện thử nghiệm, quá trình tinh chỉnh tham số PID, kết quả thực nghiệm với các kịch bản tải trọng khác nhau, phân tích hiệu năng hệ thống, và đánh giá so với mục tiêu đề ra.

**Chương 5: Kết luận và hướng phát triển** – Tổng kết các kết quả đạt được của khóa luận, nêu rõ các hạn chế còn tồn tại, và đề xuất các hướng phát triển tiếp theo bao gồm: tăng tốc độ phản hồi, mở rộng cân bằng hai trục, và tích hợp điều khiển vị trí/tốc độ.

# Chương 2.

## Cơ sở lý thuyết

Chương này trình bày các kiến thức nền tảng cần thiết cho việc thiết kế và xây dựng hệ thống cân bằng trọng tâm tự động. Các nội dung được trình bày bao gồm: lý thuyết cơ học về cân bằng và momen lực, nguyên lý hoạt động của cảm biến lực và module chuyển đổi tín hiệu, đặc tính của động cơ bước và kỹ thuật điều khiển vi bước, nền tảng vi điều khiển Arduino, thuật toán điều khiển PID, và đặc biệt là các vấn đề về lập trình thời gian thực trong hệ thống nhúng.

### 2.1. So sánh các phương pháp cân bằng robot

Trong lĩnh vực robot di động, có nhiều phương pháp khác nhau để duy trì sự cân bằng và ổn định. Mỗi phương pháp có nguyên lý hoạt động, ưu điểm và hạn chế riêng, phù hợp với các ứng dụng cụ thể. Phần này trình bày và so sánh bốn phương pháp cân bằng phổ biến nhất.

#### 2.1.1. Các phương pháp cân bằng chính

**Phương pháp 1 - Điều khiển bánh xe (Wheel-based Balancing):** Phương pháp này được sử dụng trong các sản phẩm thương mại như Segway và nhiều robot giao hàng [9]. Robot di chuyển bánh xe theo hướng nghiêng để giữ trọng tâm nằm trên đa giác đỡ.

**Phương pháp 2 - Bánh đà/Con quay hồi chuyển (Reaction Wheel/CMG):** Bánh đà quay tốc độ cao tạo momen phản lực khi thay đổi tốc độ, được ứng dụng trong điều khiển tư thế vệ tinh và một số robot thí nghiệm [10]. Khi cần tạo momen cân bằng, động cơ tăng hoặc giảm tốc độ bánh đà, theo định luật bảo toàn momen động lượng, một momen phản lực sẽ tác động lên thân robot. Control Moment Gyroscope (CMG) là biến thể sử dụng hiệu ứng tiến động của con quay hồi chuyển.

**Phương pháp 3 - Khối lượng di động (Moving Mass):** Nguyên lý hoạt động dựa trên việc di chuyển một khối lượng đối trọng để thay đổi vị trí trọng tâm của hệ thống. Khi phát hiện độ lệch, khối đối trọng được di chuyển về phía đối diện để bù đắp momen lực gây mất cân bằng. Đây chính là phương pháp được áp dụng trong đề tài này.

**Phương pháp 4 - Mở rộng đa giác đỡ (Support Polygon Expansion):** Phương pháp này tăng độ ổn định tĩnh bằng cách mở rộng diện tích tiếp xúc với mặt đất. Các robot có thể sử dụng chân chống, bánh xe phụ, hoặc thay đổi tư thế để mở rộng đa giác đỡ. Tuy nhiên, phương pháp này làm giảm tính linh hoạt của robot.

### 2.1.2. Bảng so sánh tổng hợp

Bảng 2.1 trình bày so sánh chi tiết bốn phương pháp cân bằng theo các tiêu chí quan trọng.

**Bảng 2.1. So sánh các phương pháp cân bằng robot**

Tiêu chí	Điều khiển bánh xe	Bánh đà/CMG	Khối lượng di động	Mở rộng đa giác đỡ
Nguyên lý	Di chuyển bánh xe theo hướng nghiêng	Thay đổi tốc độ bánh đà tạo momen phản lực	Di chuyển đối trọng thay đổi trọng tâm	Tăng diện tích tiếp xúc mặt đất
Cảm biến chính	IMU (gyro + accel)	IMU (gyro + accel)	Loadcell hoặc IMU	Cảm biến tiếp xúc, IMU
Cơ cấu chấp hành	Động cơ DC/BLDC cho bánh xe	Động cơ BLDC tốc độ cao	Động cơ bước + vitme/đai	Động cơ servo, xi-lanh
Thời gian đáp ứng	Rất nhanh (ms)	Nhanh (ms)	Trung bình (100ms-1s)	Chậm (giây)
Khả năng cân bằng tĩnh	Không (cần di chuyển liên tục)	Có (giới hạn bởi saturation)	Có (trong phạm vi hành trình)	Có (ổn định nhất)
Tiêu hao năng lượng	Cao khi đứng yên	Cao (bánh đà quay liên tục)	Thấp khi đã cân bằng	Rất thấp
Độ phức tạp điều khiển	Cao (hệ bất ổn định)	Rất cao	Trung bình	Thấp
Độ phức tạp cơ khí	Thấp-Trung bình	Cao	Trung bình	Thấp-Cao
Khả năng chịu tải thay đổi	Hạn chế	Hạn chế	Tốt (thiết kế phù hợp)	Tốt
Tính linh hoạt di chuyển	Cao	Cao	Cao	Thấp
Ứng dụng tiêu biểu	Segway, robot giao hàng	Vệ tinh, xe đạp tự lái	Robot mang tải, cần cẩu	Robot công nghiệp, xe nâng
Chi phí triển khai	Trung bình	Cao	Thấp-Trung bình	Thấp

### 2.1.3. Phân tích ưu nhược điểm chi tiết

**Điều khiển bánh xe:** Ưu điểm bao gồm khả năng đáp ứng rất nhanh, cấu trúc cơ khí đơn giản, và robot có thể di chuyển linh hoạt. Nhược điểm chính là không thể cân bằng tĩnh (phải di chuyển liên tục để giữ thăng bằng), tiêu hao năng lượng cao khi đứng yên, và yêu cầu mặt sàn phẳng để hoạt động hiệu quả. Phương pháp này phù hợp với các ứng dụng di chuyển liên tục như robot giao hàng, xe cân bằng cá nhân.

**Bánh đà/CMG:** Ưu điểm là có thể cân bằng mà không cần di chuyển bánh xe, đáp ứng nhanh, và hoạt động được trên nhiều loại địa hình. Nhược điểm bao gồm tiêu hao năng lượng liên tục (bánh đà phải quay), vấn đề bão hòa momen động lượng (saturation) khi nhiều kéo dài, độ phức tạp cơ khí cao, và tiếng ồn từ bánh đà tốc độ cao. Phương pháp này phù hợp với vệ tinh, tàu vũ trụ, và các robot cần cân bằng chính xác trong thời gian ngắn.

**Khối lượng di động (Phương pháp được chọn):** Ưu điểm là tiêu hao năng lượng thấp sau khi đã cân bằng (động cơ dừng), cấu trúc đơn giản với linh kiện phổ biến, có thể đo trực tiếp phân bố tải bằng loadcell, và phù hợp với ứng dụng cân bằng tĩnh hoặc bán tĩnh. Nhược điểm bao gồm thời gian đáp ứng chậm hơn (phụ thuộc vào tốc độ vitme), giới hạn bởi hành trình di chuyển của đối trọng, và khả năng bù đắp phụ thuộc vào khối lượng đối trọng. Phương pháp này phù hợp với robot mang tải có trọng tâm thay đổi chậm, khung để cân bằng cho cánh tay robot.

**Mở rộng đa giác đỡ:** Ưu điểm là độ ổn định cao nhất, không cần điều khiển phức tạp, và tiêu hao năng lượng rất thấp. Nhược điểm là làm giảm tính linh hoạt và khả năng di chuyển, tăng kích thước và khối lượng robot, và thời gian chuyển đổi trạng thái chậm. Phương pháp này phù hợp với robot công nghiệp cố định, xe nâng hàng, và các ứng dụng ưu tiên độ ổn định tuyệt đối.

#### 2.1.4. Lý do lựa chọn phương pháp khối lượng di động

Đề tài này lựa chọn phương pháp khối lượng di động dựa trên các lý do sau:

Thứ nhất, về đặc điểm ứng dụng: Khung để robot được thiết kế để cân bằng khi tải thay đổi (ví dụ: cánh tay robot gấp vật), không yêu cầu thời gian đáp ứng cực nhanh như robot tự cân bằng động.

Thứ hai, về hiệu quả năng lượng: Sau khi đạt vị trí cân bằng, động cơ bước có thể dừng hoàn toàn (giữ vị trí bằng momen từ tính), tiết kiệm năng lượng đáng kể so với phương pháp bánh đà.

Thứ ba, về khả năng đo lường: Việc sử dụng loadcell cho phép đo trực tiếp chênh lệch trọng lượng giữa hai bên, cung cấp phản hồi chính xác về trạng thái cân bằng mà không cần tính toán phức tạp từ góc nghiêng.

Thứ tư, về tính khả thi: Các linh kiện cần thiết (động cơ bước, vitme, loadcell, Arduino) đều phổ biến và có giá thành hợp lý tại Việt Nam, phù hợp với điều kiện nghiên cứu và triển khai.

Thứ năm, về độ phức tạp: So với phương pháp bánh đà hoặc CMG, phương pháp khối lượng di động có độ phức tạp cơ khí và điều khiển thấp hơn, thuận lợi cho việc nghiên cứu và phát triển trong khuôn khổ đồ án tốt nghiệp.

## 2.2. Lý thuyết cân bằng trọng tâm và momen lực

### 2.2.1. Khái niệm trọng tâm

Trọng tâm (Center of Gravity - CoG) là điểm đặt của hợp lực trọng trường tác dụng lên vật thể [8]. Đối với một hệ thống gồm  $n$  vật thể rời rạc, vị trí trọng tâm theo trục  $x$  được xác định bởi công thức:

$$x_G = \frac{\sum_{i=1}^n m_i \cdot x_i}{\sum_{i=1}^n m_i} = \frac{\sum_{i=1}^n m_i \cdot x_i}{M} \quad (1)$$

trong đó  $x_G$  là tọa độ trọng tâm của hệ,  $m_i$  là khối lượng của vật thể thứ  $i$ ,  $x_i$  là tọa độ của vật thể thứ  $i$  theo trục  $x$ , và  $M = \sum m_i$  là tổng khối lượng của hệ.

Công thức tương tự được áp dụng cho các trục  $y$  và  $z$  trong không gian ba chiều. Trong phạm vi khóa luận này, bài toán được đơn giản hóa thành cân bằng theo một trục (trái-phải), tương ứng với việc xét trọng tâm trên trục  $x$ .

### 2.2.2. Momen lực và điều kiện cân bằng tĩnh

Momen lực (Torque) là đại lượng vật lý đặc trưng cho tác dụng làm quay của một lực đối với một trục quay. Độ lớn của momen lực được tính theo công thức:

$$M = F \times d \quad (2)$$

trong đó  $M$  là momen lực (đơn vị N.m hoặc N.mm),  $F$  là lực tác dụng (N), và  $d$  là cánh tay đòn – khoảng cách vuông góc từ đường tác dụng của lực đến trục quay (m hoặc mm).

Quy ước dấu: Momen làm vật quay theo chiều kim đồng hồ thường được quy ước là âm, ngược chiều kim đồng hồ là dương (hoặc ngược lại, tùy theo hệ quy chiếu được chọn).

Điều kiện cân bằng tĩnh của một vật rắn yêu cầu tổng các momen lực tác dụng lên vật đối với một trục quay bất kỳ phải bằng không:

$$\sum M = 0 \quad (3)$$

Điều kiện này có nghĩa là tổng các momen theo chiều dương phải cân bằng với tổng các momen theo chiều âm.

### 2.2.3. Áp dụng nguyên lý momen vào bài toán cân bằng khung robot

Xét mô hình khung đế robot như một thanh cứng được đỡ tại điểm tựa ở giữa (tâm khung). Khi có tải trọng đặt lệch về một phía, momen do tải gây ra sẽ làm khung mất cân bằng. Để khôi phục cân bằng, cần di chuyển một khối đối trọng về phía đối diện sao cho momen do đối trọng tạo ra bù đắp momen do tải.

Điều kiện cân bằng được viết như sau:

$$m_{slider} \times d_{slider} = m_{load} \times d_{load} \quad (4)$$

trong đó  $m_{slider}$  là khối lượng của khối đối trọng (kg hoặc g),  $d_{slider}$  là khoảng cách từ đối trọng đến tâm khung (mm),  $m_{load}$  là độ chênh lệch khối lượng tải giữa hai bên (kg hoặc g), và  $d_{load}$  là khoảng cách từ điểm đặt tải đến tâm khung (mm).

Từ phương trình (4), có thể suy ra vị trí cần thiết của đối trọng:

$$d_{slider} = \frac{m_{load} \times d_{load}}{m_{slider}} \quad (5)$$

Phương trình này cho thấy một hạn chế quan trọng: với khối lượng đối trọng cố định  $m_{slider}$  và hành trình giới hạn  $d_{slider,max}$ , độ lệch tải tối đa có thể bù đắp là:

$$m_{load,max} = \frac{m_{slider} \times d_{slider,max}}{d_{load}} \quad (6)$$

Với hệ thống được thiết kế có  $d_{slider,max} = 120\text{mm}$  và  $d_{load} = 175\text{mm}$  (một nửa chiều rộng khung 35cm), ta có công thức tổng quát:

$$m_{load,max} = \frac{120}{175} \times m_{slider} \approx 0.69 \times m_{slider} \quad (7)$$

Công thức này cho phép tra cứu nhanh khả năng cân bằng theo khối lượng đối trọng:

**Bảng 2.2.**

: Khả năng cân bằng theo khối lượng đối trọng

Đối trọng	Độ lệch tối đa bù được	Ghi chú
200g	~137g	Dùng trong thử nghiệm
300g	~206g	-
500g	~343g	Khuyến nghị ứng dụng nhẹ
1000g	~686g	Khuyến nghị ứng dụng nặng
1500g	~1029g	Sử dụng pin/acquy

Ví dụ cụ thể: với khối trọng  $m_{slider} = 200\text{g}$ , độ lệch tải tối đa có thể bù đắp là:

$$m_{load,max} = 0.69 \times 200 \approx 137 \text{ g} \quad (8)$$

Kết quả này cho thấy để tăng khả năng cân bằng, có thể: (1) tăng khối lượng khối trọng, (2) tăng hành trình di chuyển, hoặc (3) bố trí tải gần tâm hơn.

## 2.3. Cảm biến lực Loadcell và module HX711

### 2.3.1. Cấu tạo và nguyên lý hoạt động của Loadcell

Loadcell là cảm biến chuyển đổi lực cơ học thành tín hiệu điện, hoạt động dựa trên hiệu ứng điện trở biến dạng (Piezoresistive effect)[5]. Cấu tạo cơ bản của loadcell gồm:

**Thân đàn hồi (Elastic element):** Thường làm từ thép hợp kim hoặc nhôm, được gia công với hình dạng đặc biệt để biến dạng theo hướng xác định khi chịu lực.

**Điện trở biến dạng (Strain gauge):** Là các dải điện trở mỏng được dán trực tiếp lên bề mặt thân đàn hồi. Khi thân đàn hồi biến dạng, điện trở của strain gauge thay đổi theo quan hệ:

$$\frac{\Delta R}{R} = GF \times \varepsilon \quad (9)$$

trong đó  $\Delta R/R$  là tỷ lệ thay đổi điện trở,  $GF$  là hệ số gauge (Gauge Factor, thường từ 2 đến 4 cho strain gauge kim loại), và  $\varepsilon$  là biến dạng tương đối của vật liệu.

**Cầu Wheatstone:** Bốn strain gauge được mắc theo cấu hình cầu Wheatstone để chuyển đổi sự thay đổi điện trở thành sự thay đổi điện áp. Cấu hình cầu đầy đủ (Full-bridge) cho độ nhạy cao nhất và khả năng bù nhiệt tốt.

Điện áp đầu ra của cầu Wheatstone:

$$V_{out} = V_{exc} \times \frac{\Delta R}{R} \quad (10)$$

trong đó  $V_{exc}$  là điện áp kích thích (thường 5V hoặc 10V).

### 2.3.2. Loadcell 3 dây và cách ghép thành cầu đầy đủ

Trong dự án này, loadcell 3 dây (half-bridge) được sử dụng. Loại loadcell này chỉ chứa hai strain gauge, cần ghép hai loadcell lại để tạo thành cầu Wheatstone đầy đủ.

Cách đấu nối hai loadcell 3 dây thành một cầu cho module HX711:

- Loadcell 1: Dây trắng vào E-, dây đen vào E+



- Loadcell 2: Dây trắng vào A-, dây đen vào E+
- Kết nối chung: Nối hai dây đỏ của cả hai loadcell lại với nhau và đưa vào chân A+

Với cách ghép này, hệ thống sử dụng 4 loadcell 50kg (2 cặp), tạo thành 2 cụm cảm biến độc lập cho bên trái và bên phải, mỗi cụm có khả năng chịu tải tổng cộng 100kg.

### 2.3.3. Module HX711 và giao thức truyền thông

HX711 là IC chuyên đổi tương tự sang số (ADC) 24-bit được thiết kế chuyên dụng cho các ứng dụng cân điện tử. Các đặc tính kỹ thuật chính:

- Độ phân giải: 24-bit (tương đương khoảng 16.7 triệu mức lượng tử hóa)
- Tốc độ lấy mẫu: 10 SPS (Samples Per Second) hoặc 80 SPS, tùy thuộc vào mức logic của chân RATE
- Bộ khuếch đại tích hợp: Độ lợi (Gain) có thể chọn 32, 64, hoặc 128
- Giao tiếp: Giao thức nối tiếp đồng bộ 2 dây (DOUT và SCK)
- Nguồn cấp: 2.6V đến 5.5V

Giao thức truyền thông của HX711 hoạt động như sau: Vi điều khiển tạo xung clock trên chân SCK và đọc dữ liệu từ chân DOUT. Mỗi lần đọc, HX711 truyền 24 bit dữ liệu ADC, sau đó 1-3 xung clock bổ sung để chọn kênh và độ lợi cho lần đọc tiếp theo.

### 2.3.4. Thư viện HX711\_ADC và cơ chế Non-blocking

Một trong những quyết định kỹ thuật quan trọng của dự án là sử dụng thư viện HX711\_ADC của tác giả Olav Kallhovd[2] thay vì thư viện HX711 chuẩn (của Bogde). Sự khác biệt cốt lõi nằm ở cơ chế đọc dữ liệu:

#### **Thư viện HX711 chuẩn (Blocking):**

- Hàm chính: `scale.get_units(n)`
- Cơ chế: Khi gọi hàm, vi điều khiển **dừng lại và chờ** cho đến khi HX711 hoàn thành chuyển đổi
- Thời gian chờ: Ở tốc độ 10Hz, mỗi lần đọc mất 100ms. Nếu lấy trung bình 5 mẫu (`get_units(5)`), thời gian chờ lên đến 500ms
- Hậu quả: Trong thời gian chờ, các lệnh điều khiển động cơ không được thực thi, gây hiện tượng động cơ chạy giật cục hoặc dừng hẳn

#### **Thư viện HX711\_ADC (Non-blocking):**

- Hàm chính: `LoadCell.update()` và `LoadCell.getData()`

- Cơ chế: Hàm `update()` chỉ kiểm tra xem có dữ liệu mới hay không. Nếu chưa có, hàm trả về ngay lập tức (tốn vài micro-giây). Nếu có dữ liệu mới, hàm đọc và lưu vào biến nội bộ
- Hàm `getData()` lấy giá trị từ biến nội bộ ra một cách tức thì, không cần chờ đợi
- Lợi ích: Vì điều khiển có thể thực hiện các tác vụ khác (như điều khiển động cơ) trong khi chờ dữ liệu từ cảm biến

Sự khác biệt này có ý nghĩa quyết định đối với hiệu năng hệ thống. Với kiến trúc Non-blocking, hàm `stepper.runSpeed()` có thể được gọi liên tục trong vòng lặp chính mà không bị gián đoạn bởi việc đọc cảm biến, đảm bảo động cơ bước hoạt động mượt mà ở tốc độ cao.

## 2.4. Động cơ bước và driver TB6600

### 2.4.1. Nguyên lý hoạt động của động cơ bước

Động cơ bước (Stepper Motor) là loại động cơ điện chuyển đổi các xung điện rời rạc thành chuyển động quay theo từng bước góc cố định. Khác với động cơ DC thông thường (có tốc độ phụ thuộc vào điện áp), vị trí và tốc độ của động cơ bước được xác định hoàn toàn bởi số lượng và tần số của các xung điều khiển.

Động cơ bước lai (Hybrid Stepper Motor) – loại phổ biến nhất trong các ứng dụng công nghiệp và được sử dụng trong dự án này (Nema 17) – kết hợp ưu điểm của động cơ bước nam châm vĩnh cửu và động cơ bước biến từ trở. Cấu tạo gồm:

**Stator:** Chứa các cuộn dây được quấn theo cặp (Phase A và Phase B cho động cơ lưỡng cực). Khi có dòng điện chạy qua, các cuộn dây tạo ra từ trường.

**Rotor:** Gồm lõi nam châm vĩnh cửu được bọc bởi hai đĩa răng lệch pha nhau. Kết hợp với răng trên stator, cấu trúc này cho phép góc bước nhỏ (thường  $1.8^\circ$ , tương đương 200 bước/vòng).

Nguyên lý hoạt động: Khi dòng điện trong các cuộn dây thay đổi theo trình tự xác định, từ trường stator quay theo, kéo rotor quay theo từng bước. Mỗi bước tương ứng với một góc cố định:

$$\theta_{step} = \frac{360}{N_{steps}} \quad (11)$$

trong đó  $N_{steps}$  là số bước trên một vòng quay (200 cho động cơ  $1.8^\circ$ ).

### 2.4.2. Chế độ vi bước (Microstepping) và ảnh hưởng đến hiệu năng

Vi bước là kỹ thuật điều khiển dòng điện trong các cuộn dây theo dạng sóng sin/cosin thay vì dạng bước vuông. Điều này cho phép chia nhỏ mỗi bước đầy đủ thành nhiều bước nhỏ hơn.

Số bước trên mỗi vòng quay khi sử dụng vi bước:

$$N_{micro} = N_{full} \times k \quad (12)$$

trong đó  $N_{full} = 200$  bước/vòng (cho động cơ  $1.8^\circ$ ) và  $k$  là hệ số vi bước (1, 2, 4, 8, 16, 32...).

Với chế độ 1/16 microstep được sử dụng trong dự án:

$$N_{micro} = 200 \times 16 = 3200 \text{ bước/vòng} \quad (13)$$

#### **Ưu điểm của vi bước:**

- Chuyển động mượt mà hơn, giảm rung động và tiếng ồn
- Độ phân giải vị trí cao hơn
- Giảm hiện tượng cộng hưởng ở một số dải tốc độ

#### **Nhược điểm và thách thức:**

- Yêu cầu tần số xung điều khiển cao hơn để đạt cùng tốc độ quay
- Tăng tải xử lý cho vi điều khiển
- Momen xoắn giảm nhẹ so với chế độ full-step

#### **Phân tích ràng buộc thời gian thực:**

Xét yêu cầu tốc độ 300 RPM (5 vòng/giây):

- Ở chế độ Full-step: Tần số xung =  $200 \times 5 = 1000$  Hz. Khoảng cách giữa các bước =  $1000 \mu s$
- Ở chế độ 1/16 Microstep: Tần số xung =  $3200 \times 5 = 16000$  Hz. Khoảng cách giữa các bước =  $62.5 \mu s$

Con số  $62.5 \mu s$  là “ngân sách thời gian” (time budget) tối đa mà vi điều khiển có để hoàn thành mọi tác vụ giữa hai lần tạo xung. Nếu bất kỳ tác vụ nào (đọc cảm biến, tính toán PID, giao tiếp Serial) chiếm thời gian lớn hơn ngưỡng này, hệ thống sẽ vi phạm ràng buộc thời gian thực và động cơ sẽ mất bước.

### 2.4.3. Driver TB6600 và cấu hình điều khiển

TB6600 là driver công suất cao cho động cơ bước, được thiết kế để điều khiển các động cơ Nema 17 và Nema 23. Thông số kỹ thuật:

- Điện áp làm việc: 9-42V DC
- Dòng điện tối đa: 4A/pha (có thể điều chỉnh bằng DIP switch)
- Hỗ trợ vi bước: Full, 1/2, 1/4, 1/8, 1/16, 1/32
- Tín hiệu điều khiển: PUL+ (xung bước), DIR+ (chiều quay), ENA+ (kích hoạt)
- Cách ly quang học giữa tín hiệu điều khiển và mạch công suất

Giao thức điều khiển đơn giản: Mỗi xung trên chân PUL làm động cơ quay một vi bước. Mức logic trên chân DIR quyết định chiều quay (HIGH = thuận, LOW = nghịch). Chân ENA thường được bỏ trống hoặc nối với mức logic phù hợp để luôn kích hoạt driver.

### 2.4.4. Thư viện AccelStepper và cơ chế Polling

Thư viện AccelStepper [3] điều khiển động cơ bước theo cơ chế Polling, yêu cầu gọi hàm `runSpeed()` liên tục trong vòng lặp chính.

Hàm cốt lõi `runSpeed()` hoạt động như sau (mã giả):

```
1 boolean AccelStepper::runSpeed() {
2     unsigned long time = micros();
3     if (time >= _nextStepTime) {
4         step(_direction); // Thực hiện bước
5         _nextStepTime = time + _stepInterval;
6         return true;
7     }
8     return false; // Chưa đến lúc, không làm gì
9 }
```

Thiết kế này đặt ra yêu cầu quan trọng: **Hàm `runSpeed()` phải được gọi với tần số cao hơn nhiều so với tần số bước của động cơ**. Nếu động cơ cần bước mỗi  $62.5 \mu s$  (ở 1/16 microstep, 300 RPM), thì `runSpeed()` cần được gọi ít nhất mỗi  $30 \mu s$  để đảm bảo độ chính xác thời gian.

Thư viện cũng cung cấp các hàm khác như `run()` (có gia tốc), `runToPosition()` và `runToNewPosition()` (chạy đến vị trí đích). Tuy nhiên, hai hàm sau là **blocking** – chúng chứa vòng lặp nội bộ và không trả về cho đến khi động cơ đạt vị trí đích. Trong dự án này, `runSpeed()` được sử dụng trong vòng lặp chính để đảm bảo tính non-blocking.

## 2.5. Vi điều khiển Arduino

### 2.5.1. Kiến trúc phần cứng ATmega328P

Arduino Uno/Nano sử dụng vi điều khiển ATmega328P của Atmel (nay thuộc Microchip). Đây là vi điều khiển 8-bit kiến trúc AVR với các thông số:

- Tần số xung nhịp: 16 MHz (chu kỳ lệnh 62.5 ns)
- Bộ nhớ Flash (chương trình): 32 KB
- Bộ nhớ SRAM (dữ liệu): 2 KB
- Bộ nhớ EEPROM: 1 KB
- Chân I/O số: 14 (trong đó 6 chân hỗ trợ PWM)
- Chân đầu vào analog: 6 (ADC 10-bit)
- Giao tiếp: UART, SPI, I2C
- Timer: 2 timer 8-bit, 1 timer 16-bit

Với tần số 16 MHz, vi điều khiển có thể thực thi khoảng 16 triệu lệnh đơn giản mỗi giây. Tuy nhiên, các thao tác phức tạp như phép tính dấu phẩy động, giao tiếp Serial, hay đọc ADC tiêu tốn nhiều chu kỳ xung nhịp hơn.

### 2.5.2. Mô hình lập trình Arduino

Arduino sử dụng mô hình lập trình đơn giản với hai hàm chính:

`setup()`: Được gọi một lần khi khởi động, dùng để khởi tạo các thành phần phần cứng và phần mềm.

`loop()`: Được gọi lặp đi lặp lại vô hạn sau khi `setup()` hoàn thành. Đây là nơi chứa logic chính của chương trình.

### 2.5.3. Vấn đề Blocking I/O trong giao tiếp Serial

Giao tiếp Serial (UART) là nguồn gây blocking phổ biến nhất trong các chương trình Arduino. Phân tích chi tiết cơ chế hoạt động:

#### Bộ đệm truyền (TX Buffer):

Arduino sử dụng bộ đệm vòng (ring buffer) với kích thước mặc định 64 byte để chứa dữ liệu chờ gửi. Khi gọi `Serial.print()`, dữ liệu được sao chép vào buffer. Nếu buffer còn trống, hàm trả về ngay lập tức (non-blocking). Một trình phục vụ ngắt (ISR) chạy ngầm sẽ lấy từng byte từ buffer và gửi ra chân TX.

#### Hiện tượng Buffer Overflow:

Vấn đề xảy ra khi tốc độ ghi dữ liệu vào buffer cao hơn tốc độ gửi ra ngoài. Tốc độ gửi bị giới hạn bởi Baud Rate. Ở 115200 baud (tốc độ được sử dụng trong hệ thống), thời gian gửi 1 byte:

$$t_{byte} = \frac{10 \text{ bits}}{115200 \text{ bps}} \approx 86.8 \mu s \quad (14)$$

(10 bits = 1 start bit + 8 data bits + 1 stop bit)

Khi buffer đầy (64 byte), hàm `Serial.write()` chuyển sang chế độ blocking – chờ đợi cho đến khi có chỗ trống trong buffer. Trong thời gian chờ, CPU không thể thực hiện các tác vụ khác.

#### Tính toán thiệt hại thời gian:

Giả sử chương trình in dòng lệnh 25 ký tự mỗi vòng lặp, và buffer đã đầy. Thời gian blocking:

$$T_{delay} \approx 25 \times 86.8 \mu s = 2.17 \text{ ms} \quad (15)$$

Trong 2.17 ms này:

- Hàm `runSpeed()` không được gọi
- Động cơ (yêu cầu xung mỗi  $62.5 \mu s$ ) bỏ lỡ:  $\frac{2170}{62.5} \approx 35$  xung
- Tốc độ thực tế giảm từ 16000 bước/s xuống còn khoảng 461 bước/s

Mặc dù baudrate 115200 đã giảm đáng kể thời gian blocking so với 9600 baud (nhanh hơn 12 lần), việc in Serial liên tục trong vòng lặp vẫn gây ra hiện tượng động cơ bước “chậm và giật cục”.

#### Giải pháp Time-Slicing và lý do chọn chu kỳ 100ms:

Thay vì in mỗi vòng lặp, hệ thống chỉ in thông tin giám sát mỗi 100ms. Lựa chọn này dựa trên các yếu tố:

- **Khả năng nhận thức của con người:** Mắt người chỉ nhận biết thay đổi ở tần số 10–25Hz. Với 100ms (10 lần/giây), người dùng có thể theo dõi dễ dàng mà không bị “nhòe” thông tin.
- **Tải CPU:** Mỗi dòng in khoảng 50 ký tự, tốn  $50 \times 86.8 \mu s = 4.34 \text{ ms}$ . Với chu kỳ 100ms, việc in chỉ chiếm  $4.34/100 = 4.3\%$  thời gian CPU, còn lại 95.7% cho điều khiển.
- **Đồng bộ với PID:** Chu kỳ PID là 20ms, nên  $100 \text{ ms} = 5$  chu kỳ PID. Mỗi lần in thể hiện kết quả sau 5 lần tính toán PID, giúp debug hiệu quả.
- **Tiêu chuẩn công nghiệp:** 100ms là chu kỳ logging phổ biến trong các hệ thống nhúng thời gian thực, cân bằng giữa độ chi tiết và hiệu năng.

## 2.6. Thuật toán điều khiển PID

### 2.6.1. Mô hình toán học của bộ điều khiển PID

PID (Proportional-Integral-Derivative) là thuật toán điều khiển vòng kín phổ biến nhất trong công nghiệp[7]. Bộ điều khiển PID tính toán tín hiệu điều khiển  $u(t)$  dựa trên sai số  $e(t)$  giữa giá trị đặt (Setpoint) và giá trị thực tế (Process Variable):

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt} \quad (16)$$

trong đó:

- $u(t)$ : Tín hiệu điều khiển (output)
- $e(t) = SP - PV$ : Sai số (error)
- $K_p$ : Hệ số khuếch đại tỷ lệ (Proportional gain)
- $K_i$ : Hệ số khuếch đại tích phân (Integral gain)
- $K_d$ : Hệ số khuếch đại vi phân (Derivative gain)

### 2.6.2. Vai trò của từng thành phần

#### Thành phần tỷ lệ (P):

Tạo ra tín hiệu điều khiển tỷ lệ thuận với sai số hiện tại:

$$u_P = K_p \cdot e(t) \quad (17)$$

$K_p$  càng lớn, hệ thống phản ứng càng nhanh với sai số. Tuy nhiên, nếu  $K_p$  quá lớn, hệ thống có thể dao động hoặc mất ổn định. Thành phần P đơn thuần không thể triệt tiêu hoàn toàn sai số xác lập (steady-state error).

#### Thành phần tích phân (I):

Tích lũy sai số theo thời gian:

$$u_I = K_i \cdot \int_0^t e(\tau) d\tau \quad (18)$$

Thành phần này giúp triệt tiêu sai số xác lập bằng cách tích lũy các sai số nhỏ theo thời gian. Tuy nhiên,  $K_i$  quá lớn có thể gây hiện tượng vọt lố (overshoot) và làm chậm đáp ứng của hệ thống. Ngoài ra, cần chú ý hiện tượng “windup” khi tích phân tích lũy quá lớn trong các tình huống bão hòa.

#### Thành phần vi phân (D):

Dự đoán xu hướng thay đổi của sai số:

$$u_D = K_d \cdot \frac{de(t)}{dt} \quad (19)$$

Thành phần này phản ứng với tốc độ thay đổi của sai số, giúp “phanh” hệ thống khi sai số đang giảm nhanh, từ đó giảm vọt lố và dao động. Nhược điểm là nhạy cảm với nhiễu tần số cao trong tín hiệu đo lường.

### 2.6.3. Dạng rời rạc của PID cho hệ thống số

Trong thực tế, vi điều khiển làm việc với tín hiệu rời rạc. Thuật toán PID được rời rạc hóa với chu kỳ lấy mẫu  $T_s$ :

$$u[k] = K_p \cdot e[k] + K_i \cdot T_s \sum_{j=0}^k e[j] + K_d \cdot \frac{e[k] - e[k-1]}{T_s} \quad (20)$$

Thư viện PID\_v1 [4] triển khai dạng rời rạc với các tính năng anti-windup và derivative kick prevention.

### 2.6.4. Phương pháp tinh chỉnh tham số PID

Có nhiều phương pháp tinh chỉnh (tuning) tham số PID, từ các phương pháp giải tích (Ziegler-Nichols, Cohen-Coon) đến các phương pháp thử nghiệm. Trong dự án này, phương pháp thử-sai (Trial and Error) được áp dụng:

**Bước 1:** Đặt  $K_i = K_d = 0$ . Tăng dần  $K_p$  cho đến khi hệ thống bắt đầu dao động liên tục quanh điểm cân bằng.

**Bước 2:** Giảm  $K_p$  xuống khoảng 60-70% giá trị gây dao động.

**Bước 3:** Tăng dần  $K_d$  để giảm dao động và vọt lố. Thành phần D đặc biệt hiệu quả cho hệ thống có quán tính (như khối lượng đối trọng).

**Bước 4:** Nếu còn sai số xác lập (hệ thống không về đúng điểm cân bằng), thêm  $K_i$  với giá trị nhỏ.

**Bước 5:** Lặp lại các bước trên để tinh chỉnh.



## 2.7. Lập trình thời gian thực cho hệ thống nhúng

### 2.7.1. Khái niệm hệ thống thời gian thực

Hệ thống thời gian thực (Real-time System) là hệ thống mà tính đúng đắn của kết quả không chỉ phụ thuộc vào giá trị logic của phép tính, mà còn phụ thuộc vào thời điểm kết quả được tạo ra. Trong ngữ cảnh điều khiển động cơ bước:

- **Deadline:** Mỗi xung điều khiển phải được tạo ra đúng thời điểm (trong phạm vi jitter cho phép)
- **Hậu quả vi phạm deadline:** Động cơ mất bước, chạy giật cục, hoặc dừng hẳn
- **Loại hệ thống:** Soft real-time (vi phạm deadline gây suy giảm chất lượng, không gây hỏng hóc nghiêm trọng)

### 2.7.2. Blocking vs Non-blocking I/O

**Blocking I/O:** Khi thực hiện một thao tác I/O (đọc cảm biến, gửi dữ liệu Serial), CPU chờ đợi cho đến khi thao tác hoàn thành. Trong thời gian chờ, không có tác vụ nào khác được thực hiện.

**Non-blocking I/O:** Thao tác I/O được khởi tạo và trả về ngay lập tức. CPU có thể thực hiện các tác vụ khác trong khi chờ I/O hoàn thành. Khi I/O sẵn sàng, CPU được thông báo (qua polling hoặc interrupt).

Trong dự án này, các giải pháp non-blocking được áp dụng:

- Thư viện HX711\_ADC với hàm `update()` non-blocking
- Giới hạn tần suất in Serial (mỗi 100ms thay vì mỗi vòng lặp)
- Sử dụng Baud Rate cao (115200) để giảm thời gian blocking khi buffer đầy
- Kiểm tra `Serial.availableForWrite()` trước khi gửi dữ liệu lớn

### 2.7.3. Cơ chế Time-Slicing với `millis()`

Time-Slicing là kỹ thuật phân chia thời gian CPU cho các tác vụ khác nhau dựa trên đồng hồ hệ thống. Trong Arduino, hàm `millis()` trả về số mili-giây kể từ khi khởi động.

Cấu trúc Time-Slicing điển hình:

```
1 unsigned long t_task1 = 0;
2 unsigned long t_task2 = 0;
3
4 void loop() {
5     // Tác vụ nen - chạy mọi vòng lặp
6     stepper.runSpeed();
```

```

7   LoadCell.update();
8
9   // Tác vụ 1 - chạy mỗi 20ms
10  if (millis() - t_task1 >= 20) {
11      // Đọc cảm biến, tính PID
12      t_task1 = millis();
13  }
14
15  // Tác vụ 2 - chạy mỗi 100ms
16  if (millis() - t_task2 >= 100) {
17      // In thông tin Serial
18      t_task2 = millis();
19  }
20 }

```

Cơ chế này cho phép:

- Tác vụ ưu tiên cao (runSpeed()) được gọi liên tục
- Tác vụ điều khiển (PID) chạy với tần số cố định (50Hz)
- Tác vụ giám sát (Serial) chạy với tần số thấp hơn (10Hz) để không ảnh hưởng đến các tác vụ quan trọng

#### 2.7.4. Vấn đề Pulse Starvation trong điều khiển động cơ bước

Pulse Starvation (đói xung) là hiện tượng động cơ bước không nhận đủ xung điều khiển do CPU bận thực hiện các tác vụ khác. Hậu quả:

- **Giảm tốc độ thực tế:** Thay vì 16000 bước/s, động cơ có thể chỉ đạt vài trăm bước/s
- **Mất đồng bộ từ trường:** Rotor không theo kịp từ trường stator, gây mất momen hoặc quay ngược
- **Rung lắc và tiếng ồn:** Xung không đều gây kích thích cộng hưởng cơ học

Để tránh Pulse Starvation:

- Loại bỏ hoặc giảm thiểu các hàm blocking trong vòng lặp chính
- Đảm bảo thời gian thực thi của mỗi vòng lặp nhỏ hơn khoảng cách giữa các xung yêu cầu
- Sử dụng thư viện điều khiển động cơ dựa trên ngắt Timer (như FastAccelStepper) nếu cần tốc độ rất cao

#### 2.7.5. Kỹ thuật Deadzone và Hysteresis trong điều khiển

**Deadzone (Vùng chết):**

Là vùng giá trị sai số mà hệ thống không phản ứng. Trong dự án này, Deadzone được đặt  $\pm 100g$ :

- Nếu  $|e| \leq 100g$ : Động cơ dừng, PID tắt
- Nếu  $|e| > 100g$ : PID hoạt động, động cơ di chuyển

Mục đích: Tránh động cơ chạy liên tục để bù các sai số nhỏ do nhiễu cảm biến hoặc rung động cơ học.

#### **Hysteresis (Độ trễ):**

Là kỹ thuật sử dụng hai ngưỡng khác nhau cho việc bật và tắt một chế độ:

- Ngưỡng bật (Start threshold):  $110g (= 100g + 10g \text{ hysteresis})$
- Ngưỡng tắt (Stop threshold):  $90g (= 100g - 10g \text{ hysteresis})$

Logic hoạt động:

- Nếu đang dừng và  $|e| > 110g$ : Bật chế độ cân bằng
- Nếu đang chạy và  $|e| < 90g$ : Tắt chế độ cân bằng

Mục đích: Tránh hiện tượng bật/tắt liên tục (chattering) khi sai số dao động quanh ngưỡng Deadzone.

#### **Minimum Speed Threshold (Ngưỡng tốc độ tối thiểu):**

Động cơ bước có vùng tốc độ thấp gây cộng hưởng và rung lắc. Giải pháp:

- Nếu PID tính ra tốc độ  $> 0$  nhưng  $< 4000$  bước/s: Ép tốc độ lên  $4000$  bước/s
- Nếu PID tính ra tốc độ  $= 0$  hoặc  $\geq 4000$ : Giữ nguyên

Điều này đảm bảo khi động cơ cần chạy, nó chạy ở tốc độ đủ cao để vượt qua vùng cộng hưởng.

# Chương 3.

## Thiết kế hệ thống

Chương này trình bày chi tiết quá trình thiết kế hệ thống khung đế cân bằng trọng tâm, bao gồm: phân tích yêu cầu thiết kế, thiết kế cơ khí, thiết kế mạch điện, và thiết kế phần mềm điều khiển.

### 3.1. Phân tích yêu cầu thiết kế

#### 3.1.1. Yêu cầu chức năng

Dựa trên mục tiêu đề tài và phân tích ứng dụng thực tế, hệ thống cần đáp ứng các yêu cầu chức năng sau:

**F1 - Phát hiện độ lệch trọng tâm:** Hệ thống phải có khả năng đo lường sự chênh lệch trọng lượng giữa hai bên khung (trái và phải) với độ phân giải tối thiểu 10g.

**F2 - Bù đắp độ lệch tự động:** Khi phát hiện độ lệch vượt ngưỡng cho phép, hệ thống phải tự động di chuyển khối đối trọng để bù đắp, đưa trọng tâm về vị trí cân bằng.

**F3 - Xác định điểm gốc (Homíng):** Khi khởi động, hệ thống phải có khả năng tự động xác định vị trí tham chiếu của khối đối trọng thông qua công tắc hành trình.

**F4 - Bảo vệ hành trình:** Hệ thống phải ngăn chặn khối đối trọng di chuyển vượt quá giới hạn cơ khí.

**F5 - Giám sát hoạt động:** Cung cấp khả năng theo dõi các thông số hoạt động thông qua giao diện Serial.

#### 3.1.2. Yêu cầu phi chức năng

**NF1 - Kích thước:** Khung đế có kích thước 35×35cm, phù hợp với robot di động cỡ nhỏ và vừa.

**NF2 - Thời gian đáp ứng:** Hệ thống phải bắt đầu di chuyển đối trọng trong vòng 100ms sau khi phát hiện độ lệch.

**NF3 - Độ ổn định:** Động cơ bước phải hoạt động mượt mà, không bị giật cục hay mất bước.

**NF4 - Độ chính xác cân bằng:** Sau khi cân bằng, độ lệch còn lại phải nằm trong vùng  $\pm 100\text{g}$ .

### 3.1.3. Ràng buộc thiết kế

Quá trình thiết kế phải tuân thủ các ràng buộc: sử dụng linh kiện phổ biến tại Việt Nam (Arduino, Nema 17, TB6600, Loadcell 50kg, HX711); khung cơ khí dùng nhôm định hình 20×20mm; và tốc độ tối đa của động cơ bị giới hạn khoảng 15000 bước/giây do giới hạn của thư viện AccelStepper.

## 3.2. Thiết kế cơ khí

### 3.2.1. Tổng quan cấu trúc cơ khí

Hệ thống cơ khí bao gồm các thành phần chính: khung đế hình vuông làm từ nhôm định hình, cơ cấu di chuyển đối trọng gồm vitme và thanh trượt, động cơ bước gắn ở một đầu vitme, các cụm loadcell gắn ở bốn góc khung, và công tắc hành trình để xác định điểm gốc. Bảng 3.1 trình bày thông số khung nhôm định hình.

**Bảng 3.1. Thông số khung nhôm định hình**

Thông số	Giá trị
Kích thước tổng thể	350 × 350 mm
Tiết diện thanh nhôm	20 × 20 mm
Vật liệu	Hợp kim nhôm 6063-T5
Phương pháp liên kết	Ke góc vuông + bu-lông M5
Khối lượng khung	Khoảng 1.2 kg

Lý do lựa chọn nhôm định hình: độ cứng vững cao nhờ tiết diện rỗng với các gân tăng cứng bên trong; dễ gia công và lắp ráp; linh hoạt nhờ rãnh chữ T cho phép gắn linh kiện ở bất kỳ vị trí nào; nhẹ với tỷ trọng 2.7 g/cm<sup>3</sup>; và khả năng tái sử dụng cao.

### 3.2.2. Cơ cấu vitme - thanh trượt

Cơ cấu di chuyển đối trọng sử dụng truyền động vitme-đai ốc kết hợp với thanh trượt tròn dẫn hướng. Bảng 3.2 trình bày thông số vitme.

**Bảng 3.2. Thông số vitme**

Thông số	Giá trị
Đường kính danh nghĩa	8 mm
Bước ren (Pitch)	2 mm
Chiều dài làm việc	250 mm
Loại	Vitme bi hoặc vitme thang

Bước ren 2mm có nghĩa mỗi vòng quay của động cơ, khối trượt di chuyển 2mm. Tính toán tốc độ di chuyển tối đa với chế độ 1/16 microstep (3200 bước/vòng) và tốc độ xung tối đa 15000 bước/giây:  $n_{max} = 15000/3200 \times 60 = 281$  RPM, suy ra  $v_{max} = 281 \times 2/60 = 9.4$  mm/s. Tốc độ này khá ổn, để di chuyển hết hành trình 120mm cần khoảng 13 giây.

Thanh trượt tròn đường kính 8mm, chiều dài 280mm, vật liệu thép mạ cứng, song song với vítme. Thanh trượt có chức năng dẫn hướng cho khối trượt di chuyển thẳng và chống xoay.

### 3.2.3. Bố trí cảm biến Loadcell

Hệ thống sử dụng 4 loadcell 50kg được bố trí ở bốn góc khung, ghép thành 2 cụm (trái và phải). Cụm trái gồm 2 loadcell ở góc trái-trước và trái-sau, ghép thành cầu Wheatstone đầy đủ. Cụm phải gồm 2 loadcell ở góc phải-trước và phải-sau. Mỗi cụm kết nối với một module HX711 riêng biệt.

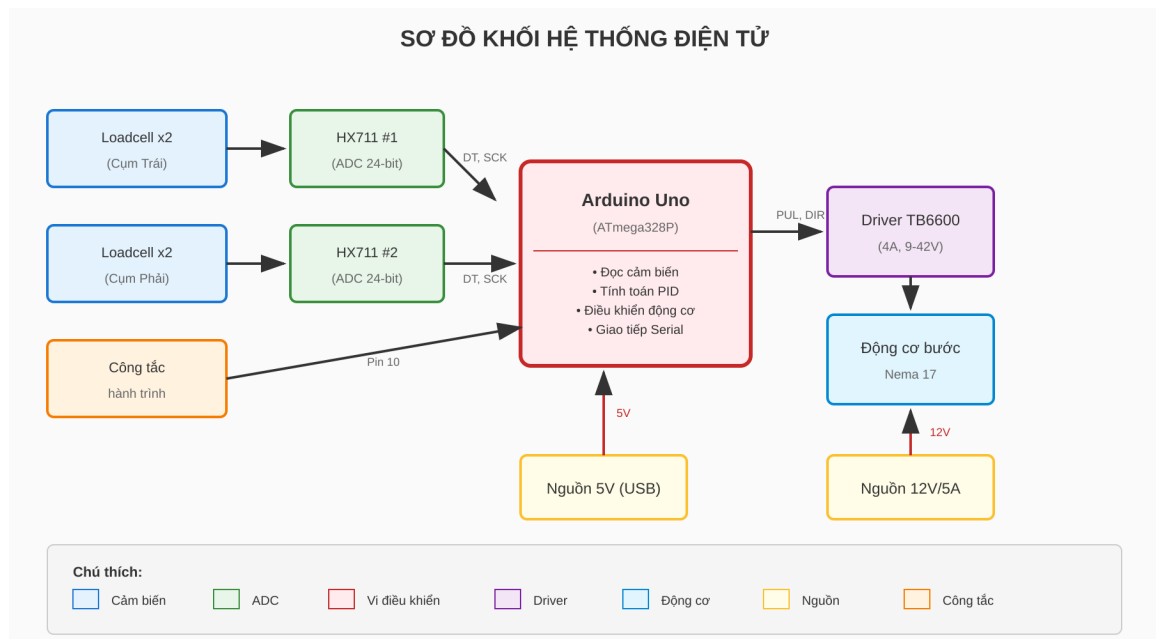
### 3.2.4. Giới hạn hành trình

Hành trình di chuyển của khối đối trọng được xác định từ vị trí tâm (0): giới hạn trái là -120mm (tương ứng -192000 bước), giới hạn phải là +110mm (tương ứng +176000 bước). Công tắc hành trình được lắp đặt ở vị trí giới hạn phải (+110mm) để xác định điểm gốc khi thực hiện Homing.

## 3.3. Thiết kế mạch điện

### 3.3.1. Sơ đồ khối hệ thống

Hệ thống điện tử bao gồm các khối chức năng chính: Khối cảm biến (2 cụm Loadcell kết nối với 2 module HX711), Khối xử lý trung tâm (Arduino Uno), Khối điều khiển động cơ (Driver TB6600 và động cơ bước Nema 17), Khối nguồn (12V cho động cơ, 5V cho mạch điều khiển), và Khối giới hạn hành trình (công tắc hành trình). Hình 3.1 thể hiện sơ đồ khối hệ thống.



Hình 3.1. Sơ đồ khối hệ thống điện tử

### 3.3.2. Danh sách linh kiện điện tử

Bảng 3.3 liệt kê các linh kiện điện tử sử dụng trong hệ thống.

**Bảng 3.3. Danh sách linh kiện điện tử**

STT	Linh kiện	Thông số	SL
1	Vi điều khiển Arduino	Uno hoặc Nano (ATmega328P)	1
2	Module HX711	ADC 24-bit cho loadcell	2
3	Loadcell	50kg, 3 dây (half-bridge)	4
4	Driver TB6600	9-42V, 4A max	1
5	Động cơ bước	Nema 17, 1.8°/bước, 1.5A	1
6	Công tắc hành trình	Micro switch với cần gạt	1
7	Nguồn xung	12V 5A	1

### 3.3.3. Sơ đồ kết nối chi tiết

Kết nối giữa Arduino và các module HX711 được thể hiện trong Bảng 3.4.

**Bảng 3.4. Kết nối Arduino với các module HX711**

Module	Chân HX711	Chân Arduino	Ghi chú
HX711 Trái	VCC	5V	Nguồn cấp
HX711 Trái	GND	GND	Mass chung
HX711 Trái	DT (DOUT)	Pin 4	Dữ liệu
HX711 Trái	SCK	Pin 5	Xung clock
HX711 Phải	VCC	5V	Nguồn cấp
HX711 Phải	GND	GND	Mass chung
HX711 Phải	DT (DOUT)	Pin 6	Dữ liệu
HX711 Phải	SCK	Pin 7	Xung clock

Kết nối giữa Arduino và driver TB6600 được thể hiện trong Bảng 3.5.

**Bảng 3.5. Kết nối Arduino với driver TB6600**

Chân TB6600	Chân Arduino	Chức năng
PUL+	Pin 8	Tín hiệu xung bước
PUL-	GND	Mass tín hiệu
DIR+	Pin 9	Tín hiệu chiều quay
DIR-	GND	Mass tín hiệu
ENA+	Không kết nối	Kích hoạt driver
ENA-	Không kết nối	(Bỏ trống = luôn bật)

Kết nối công tắc hành trình: chân NO (Normally Open) nối với Pin 10 của Arduino sử dụng INPUT\_PULLUP, chân COM nối với GND. Logic hoạt động: khi chưa nhấn, Pin 10 ở mức HIGH; khi nhấn (khởi trượt chạm công tắc), Pin 10 xuống mức LOW.

### 3.3.4. Cấu hình Driver TB6600

Cấu hình DIP Switch trên TB6600: dòng điện đặt 1.5A theo bảng tra của TB6600, chế độ vi bước 1/16 (3200 pulse/rev) với SW4=ON, SW5=ON, SW6=OFF. Lưu ý cấu hình có thể khác nhau giữa các phiên bản TB6600, cần tham khảo datasheet của nhà sản xuất cụ thể.

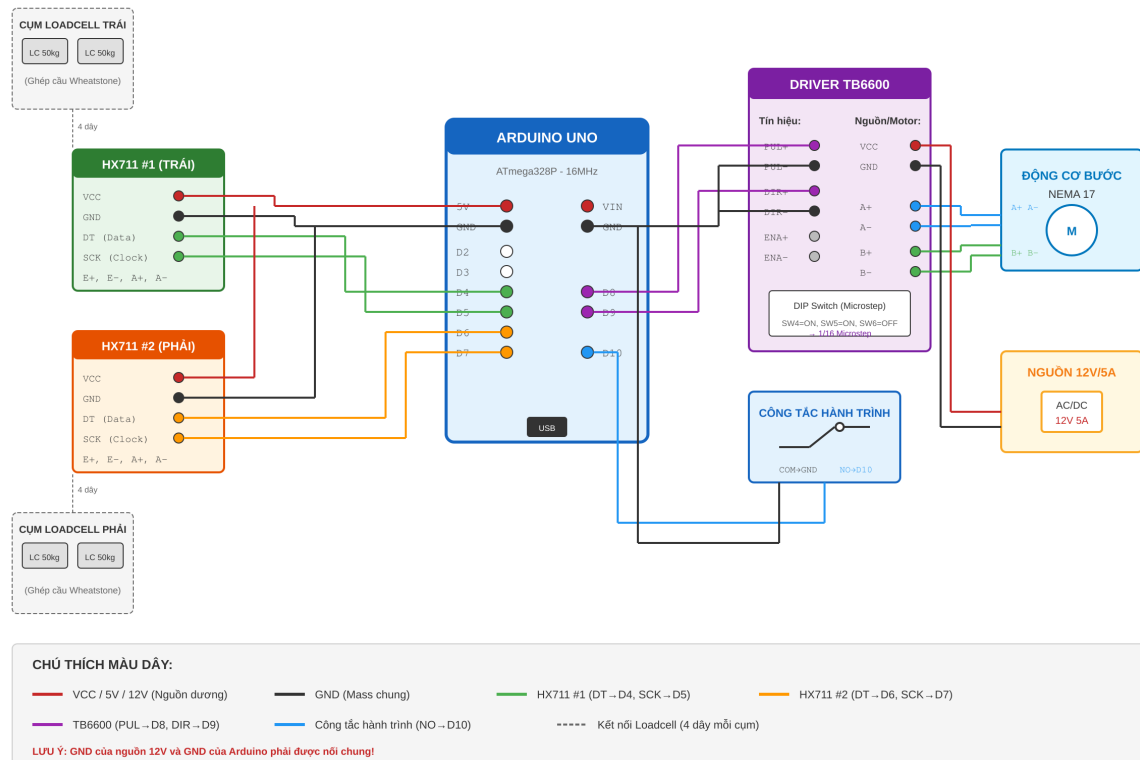
### 3.3.5. Sơ đồ nguyên lý mạch điện

Hình 3.2 thể hiện sơ đồ nguyên lý mạch điện chi tiết của toàn bộ hệ thống, bao gồm các kết nối giữa Arduino, HX711, TB6600, động cơ bước, công tắc hành trình và nguồn điện.



## SƠ ĐỒ NGUYÊN LÝ MẠCH ĐIỆN

Hệ thống khung cân bằng trọng tâm tự động



Hình 3.2. Sơ đồ nguyên lý mạch điện hệ thống

### 3.3.6. Thiết kế nguồn điện

Hệ thống sử dụng hai nguồn điện riêng biệt. Nguồn 12V/5A cấp cho driver TB6600 và động cơ, công suất 60W dư so với yêu cầu của Nema 17. Nguồn 5V cấp từ cổng USB của Arduino cho Arduino, HX711 và các cảm biến, dòng tiêu thụ ước tính dưới 500mA. Lưu ý quan trọng: GND của nguồn 12V và GND của Arduino phải được nối chung để đảm bảo tín hiệu điều khiển hoạt động đúng.

### 3.3.7. Xử lý nhiễu điện từ

Động cơ bước và driver công suất có thể tạo ra nhiễu điện từ ảnh hưởng đến tín hiệu loadcell. Các biện pháp giảm nhiễu bao gồm: tách biệt nguồn cho động cơ và mạch điều khiển; xoắn dây tín hiệu loadcell theo dạng twisted pair; giữ dây kết nối ngắn nhất có thể (dưới 50cm); có thể thêm tụ điện 100nF gần chân nguồn của HX711; và áp dụng lọc phần mềm bằng cách làm tròn giá trị đọc (quantization).

## **3.4. Thiết kế phần mềm**

### **3.4.1. Kiến trúc phần mềm tổng quan**

Phần mềm điều khiển được thiết kế theo kiến trúc Superloop với Time-Slicing, đảm bảo tính non-blocking và đáp ứng các ràng buộc thời gian thực. Kiến trúc này chia thành ba tầng nhiệm vụ với độ ưu tiên và tần suất khác nhau:

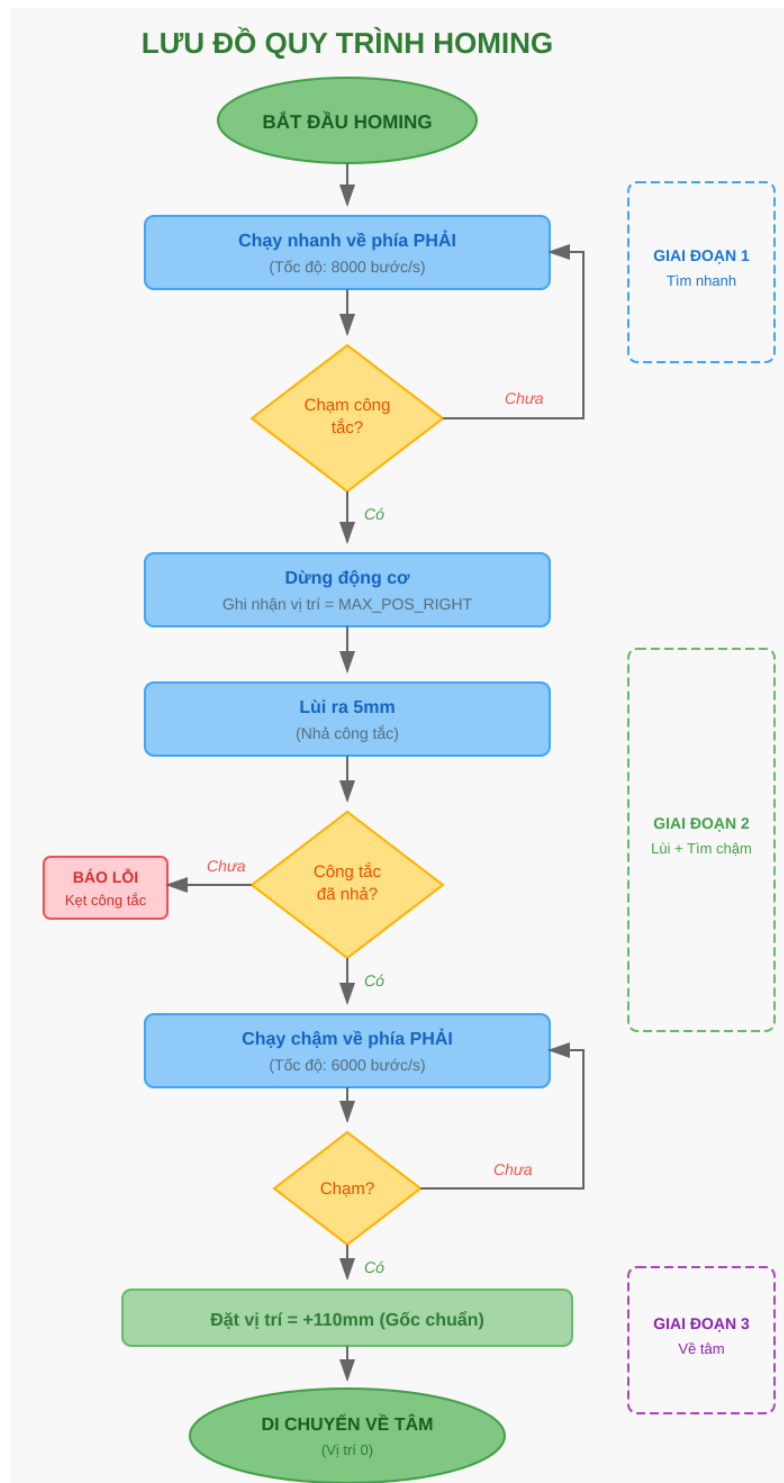
Nhiệm vụ nền chạy mỗi vòng lặp (khoảng  $10\ \mu s$ ) bao gồm gọi hàm LoadCell.update() để kiểm tra dữ liệu mới một cách non-blocking và gọi hàm stepper.runSpeed() để tạo xung nếu đến thời điểm.

Nhiệm vụ điều khiển chạy mỗi 20ms (tần số 50Hz) bao gồm đọc giá trị cảm biến, tính toán sai số, xử lý logic Deadzone và Hysteresis, tính toán PID, và cập nhật tốc độ động cơ.

Nhiệm vụ giám sát chạy mỗi 100ms (tần số 10Hz) bao gồm in các thông số ra Serial và hiển thị trạng thái hệ thống.

### **3.4.2. Quy trình Homing**

Quy trình Homing xác định điểm gốc của hệ thống với độ chính xác cao thông qua phương pháp hai giai đoạn, được minh họa trong Hình 3.3.



Hình 3.3. Lưu đồ quy trình Homing

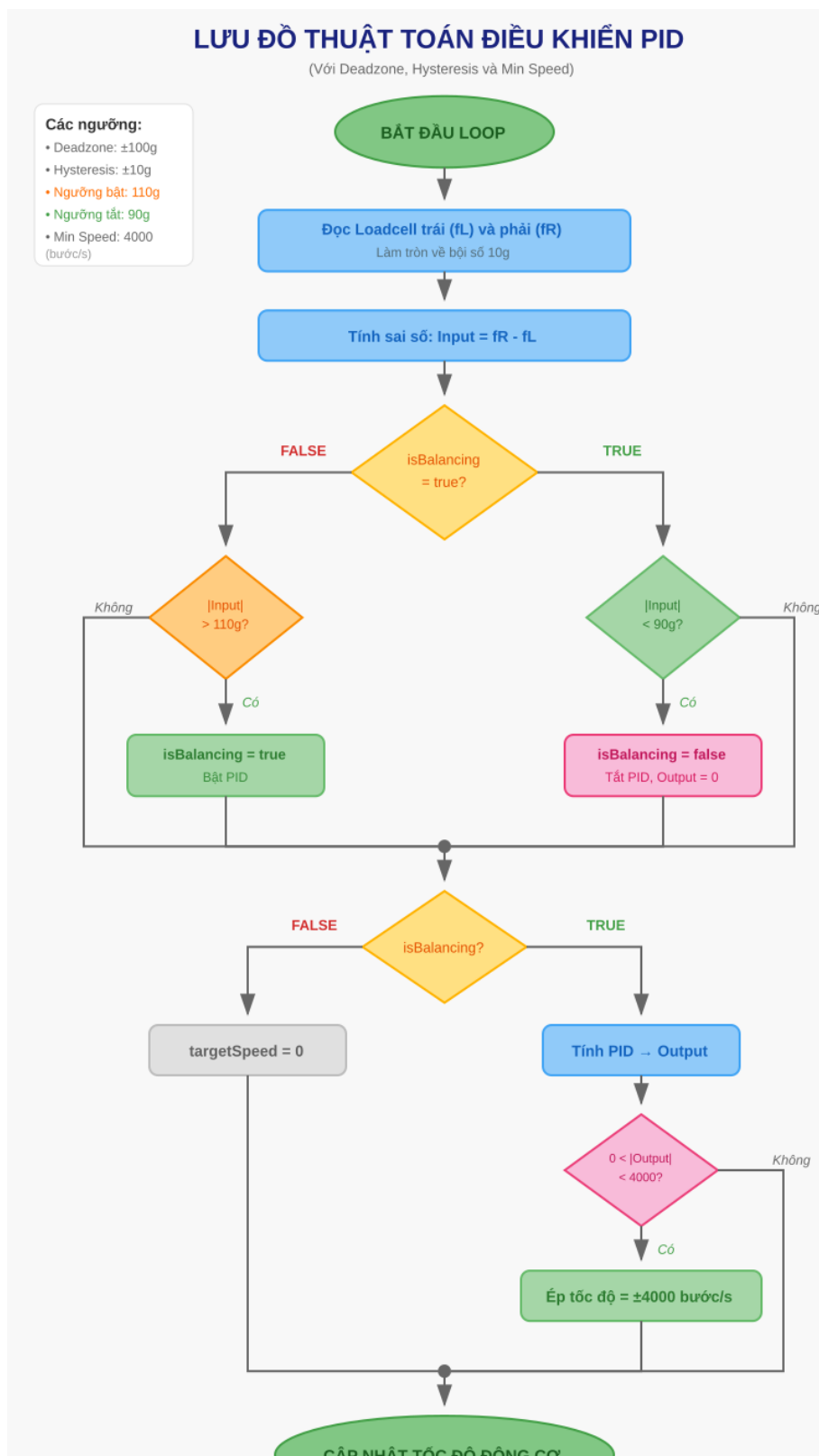
**Giai đoạn 1 - Tìm nhanh:** Di chuyển khối trượt về phía công tắc hành trình với tốc độ cao 8000 bước/s. Khi chạm công tắc (Pin 10 xuống LOW), dừng ngay lập tức và ghi nhận vị trí tạm thời là MAX\_POS\_RIGHT.

**Giai đoạn 2 - Lùi ra và tìm chậm:** Lùi ra 5mm để nhả công tắc, kiểm tra công tắc đã nhả (Pin 10 lên HIGH), sau đó di chuyển chậm trở lại với tốc độ 6000 bước/s cho đến khi chạm công tắc lần thứ hai. Vị trí chạm lần hai là điểm gốc chính xác.

**Giai đoạn 3 - Di chuyển về tâm:** Thiết lập vị trí hiện tại là MAX\_POS\_RIGHT (+110mm), di chuyển đến vị trí 0 (tâm khung), và chờ 5 giây để loadcell ổn định trước khi bắt đầu cân bằng.

### **3.4.3. Logic điều khiển PID với Deadzone và Hysteresis**

Logic điều khiển được thiết kế để tránh các vấn đề phổ biến trong hệ thống cân bằng, được minh họa trong Hình 3.4.



Hình 3.4. Lưu đồ thuật toán điều khiển PID

**Vấn đề 1 - Dao động quanh điểm cân bằng (Hunting):** Nếu không có vùng chết, khi độ lệch về gần 0, PID vẫn tiếp tục điều chỉnh, gây ra dao động liên tục. Giải pháp là áp dụng Deadzone  $\pm 100g$ .

**Vấn đề 2 - Bật/tắt liên tục (Chattering):** Nếu chỉ dùng một ngưỡng 100g, khi sai số dao động quanh ngưỡng này, hệ thống sẽ bật/tắt liên tục. Giải pháp là áp dụng Hysteresis  $\pm 10g$  với ngưỡng bật 110g và ngưỡng tắt 90g.

**Vấn đề 3 - Động cơ rung ở tốc độ thấp:** Động cơ bước có các vùng tốc độ gây cộng hưởng và rung lắc. Giải pháp là áp dụng ngưỡng tốc độ tối thiểu 4000 bước/s.

#### 3.4.4. Xử lý giới hạn hành trình mềm

Ngoài công tắc hành trình vật lý ở vị trí +110mm, phần mềm còn kiểm tra giới hạn mềm (soft limits) để ngăn khối trượt va chạm vào hai đầu. Nếu vị trí hiện tại đã đạt giới hạn trái và tốc độ đang âm (muốn đi tiếp sang trái), hệ thống ép tốc độ về 0. Tương tự cho giới hạn phải.

#### 3.4.5. Các tham số cấu hình

Bảng 3.6 tổng hợp các tham số cấu hình của phần mềm.

**Bảng 3.6. Bảng tổng hợp các tham số cấu hình phần mềm**

Tham số	Giá trị	Ý nghĩa
MICROSTEP	16	Chế độ vi bước 1/16
STEPS_PER_MM	1600	Số bước trên 1mm
MAX_POS_RIGHT	+176000	Giới hạn phải (bước)
MAX_POS_LEFT	-192000	Giới hạn trái (bước)
MAX_SPEED_PID	8000	Tốc độ tối đa (bước/s)
MOTOR_ACCEL	16000	Gia tốc (bước/s <sup>2</sup> )
MIN_SPEED_THRESHOLD	4000	Tốc độ tối thiểu
Kp	40.0	Hệ số tỷ lệ PID
Ki	0	Hệ số tích phân PID
Kd	0	Hệ số vi phân PID
ACCEPTABLE_RANGE	100.0	Vùng chết (g)
HYSTERESIS_GAP	10.0	Độ trễ (g)
STEP_SIZE	10.0	Độ chia quantization (g)
PID_INTERVAL	20	Chu kỳ tính PID (ms)
PRINT_INTERVAL	100	Chu kỳ in Serial (ms)

### 3.4.6. Quy trình hiệu chuẩn Loadcell

Trước khi sử dụng, mỗi cụm loadcell cần được hiệu chuẩn để xác định hệ số chuyển đổi. Quy trình gồm 5 bước: nạp chương trình hiệu chuẩn từ ví dụ của thư viện HX711\_ADC; đảm bảo không có tải trên loadcell và thực hiện lệnh tare về 0; đặt vật chuẩn có khối lượng đã biết (ví dụ 500g) lên loadcell; điều chỉnh hệ số hiệu chuẩn cho đến khi giá trị đọc được bằng khối lượng vật chuẩn; và ghi nhận hệ số hiệu chuẩn để cập nhật vào chương trình chính.

Trong dự án này, hệ số hiệu chuẩn đã xác định là  $CALIB\_LEFT = -53.13$  và  $CALIB\_RIGHT = -55.36$ . Dấu âm là do cách lắp đặt loadcell theo chiều nén.

## Chương 4.

# Thực nghiệm và đánh giá

Chương này trình bày chi tiết các kết quả thực nghiệm thu được từ hệ thống khung cân bằng trọng tâm. Dữ liệu được thu thập trực tiếp từ Serial Monitor trong quá trình vận hành thực tế, bao gồm: kiểm tra độ chính xác cảm biến, so sánh hiệu năng với các giá trị  $K_p$  khác nhau, và thử nghiệm tổng hợp với nhiều kịch bản tải trọng.

### 4.1. Triển khai hệ thống

#### 4.1.1. Môi trường thử nghiệm

Hệ thống được triển khai và thử nghiệm trong điều kiện phòng thí nghiệm với các thông số:

- Nhiệt độ phòng: 25–30°C
- Mặt sàn: Phẳng, cứng (gạch men)
- Nguồn điện: Ổn định 220V AC, chuyển đổi sang 12V DC và 5V DC
- Không có rung động ngoại lai đáng kể

#### 4.1.2. Các tham số cấu hình thử nghiệm

Bảng 4.2 tổng hợp các tham số cấu hình được sử dụng trong toàn bộ quá trình thử nghiệm.

**Bảng 4.1.**

: Tham số cấu hình hệ thống trong thử nghiệm

Tham số	Giá trị	Ý nghĩa
ACCEPTABLE_RANGE	100.0 g	Vùng chết (Deadzone)
HYSTERESIS_GAP	10.0 g	Độ trễ Hysteresis
Ngưỡng bật [RUN]	> 110 g	100 + 10
Ngưỡng tắt [OK]	< 90 g	100 – 10
MIN_SPEED_THRESHOLD	4000 bước/s	Tốc độ tối thiểu
MAX_SPEED_PID	8000 bước/s	Tốc độ tối đa
MOTOR_ACCEL	16000 bước/s <sup>2</sup>	Gia tốc động cơ
STEP_SIZE	10.0 g	Độ chia quantization
PID_INTERVAL	20 ms	Chu kỳ tính PID



## 4.2. Kiểm tra độ chính xác cảm biến Loadcell

Trước khi tiến hành các thử nghiệm chính, cần đảm bảo hai cụm loadcell (Trái và Phải) đã được hiệu chuẩn chính xác.

### 4.2.1. Phương pháp kiểm tra

- **Vật chuẩn:** Quả cân 210 gram
- **Quy trình:** Đặt vật lên cụm loadcell Trái, ghi nhận giá trị ổn định; sau đó đặt lên cụm Phải, ghi nhận giá trị
- **Tiêu chí đánh giá:** Sai số < 5%, không có hiện tượng crosstalk (đọc chéo)

### 4.2.2. Kết quả đo

Bảng 4.4 trình bày kết quả kiểm tra độ chính xác cảm biến.

**Bảng 4.3.**

: Kết quả kiểm tra độ chính xác cảm biến với vật chuẩn 210g

Vị trí đặt vật	Giá trị L (g)	Giá trị R (g)	Sai số	Kết quả
Bên TRÁI	210	0	0%	<b>Đạt</b>
Bên PHẢI	0	210	0%	<b>Đạt</b>

**Nhận xét:** Cả hai cụm loadcell đều đọc chính xác giá trị 210g với sai số 0%. Không có hiện tượng crosstalk – khi đặt vật lên một bên, bên còn lại đọc giá trị 0. Giá trị đọc ổn định trong suốt thời gian đo, xác nhận hệ thống cảm biến sẵn sàng cho các thử nghiệm tiếp theo.

## 4.3. Thử nghiệm so sánh các giá trị hệ số $K_p$

Mục đích của thử nghiệm này là xác định giá trị  $K_p$  tối ưu cho bộ điều khiển PID. Ba giá trị  $K_p$  được thử nghiệm là 20, 30 và 40, với cùng điều kiện: đặt vật nặng khoảng 200g lên bên TRÁI, theo dõi quá trình cân bằng từ khi bắt đầu cho đến khi đạt trạng thái [OK].

### 4.3.1. Kết quả chi tiết từng giá trị $K_p$

**Thử nghiệm với  $K_p = 20$**

Dữ liệu thu thập từ Serial Monitor:

- Thời điểm bắt đầu: 14:30:08 – Độ lệch –120g, chuyển sang [RUN]

- Độ lệch đỉnh:  $-220\text{g}$
- Tốc độ động cơ:  $4000\text{--}4400$  bước/s (công thức:  $Speed = K_p \times |Lch| = 20 \times 220 = 4400$ )
- Thời điểm kết thúc:  $14:30:33$  – Độ lệch  $-80\text{g}$ , chuyển về [OK]
- **Thời gian xác lập: khoảng 25 giây**
- Thử nghiệm với  $K_p = 30$**
- Thời điểm bắt đầu:  $14:11:58$  – Độ lệch  $-130\text{g}$ , chuyển sang [RUN]
- Độ lệch đỉnh:  $-220\text{g}$
- Tốc độ động cơ đỉnh:  $6600$  bước/s ( $Speed = 30 \times 220 = 6600$ )
- Thời điểm kết thúc:  $14:12:19$  – Độ lệch  $-80\text{g}$ , chuyển về [OK]
- **Thời gian xác lập: khoảng 21 giây**
- Thử nghiệm với  $K_p = 40$**
- Thời điểm bắt đầu:  $13:39:20$  – Độ lệch  $-140\text{g}$ , chuyển sang [RUN]
- Độ lệch đỉnh:  $-220\text{g}$
- Tốc độ động cơ đỉnh:  $8000$  bước/s ( $Speed = 40 \times 220 = 8800$ , bị giới hạn bởi  $MAX\_SPEED$ )
- Thời điểm kết thúc:  $13:39:36$  – Độ lệch  $-80\text{g}$ , chuyển về [OK]
- **Thời gian xác lập: khoảng 16 giây**

#### 4.3.2. Bảng so sánh tổng hợp

Bảng 4.6 so sánh hiệu năng hệ thống với các giá trị  $K_p$  khác nhau.

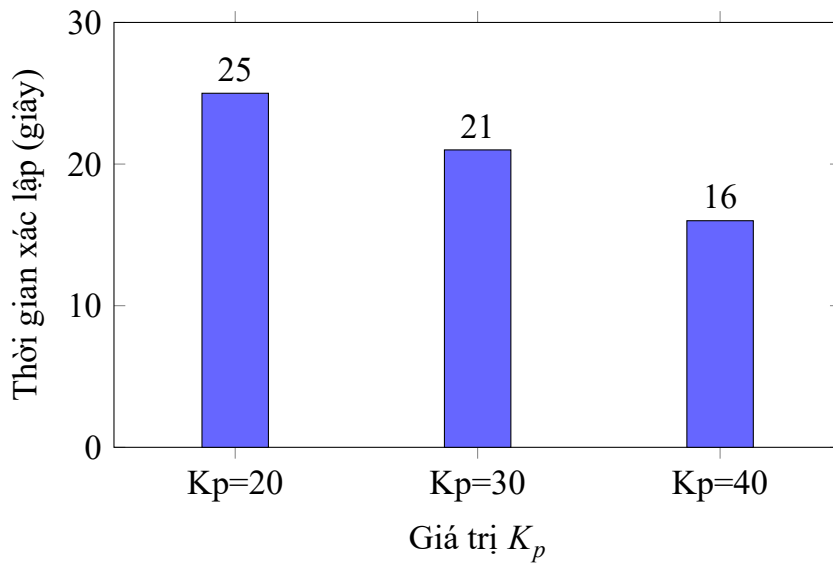
**Bảng 4.5.**

: So sánh hiệu năng với các giá trị  $K_p$  khác nhau

Chỉ tiêu	$K_p = 20$	$K_p = 30$	$K_p = 40$
Độ lệch đỉnh (g)	$-220$	$-220$	$-220$
Tốc độ đỉnh (bước/s)	$4400$	$6600$	$8000$
<b>Thời gian xác lập (s)</b>	khoảng 25	khoảng 21	<b>khoảng 16</b>
Độ lệch cuối (g)	$-80$ [OK]	$-80$ [OK]	$-80$ [OK]
Overshoot	Không	Không	Không

#### 4.3.3. Biểu đồ so sánh thời gian xác lập

Hình 4.2 minh họa mối quan hệ giữa giá trị  $K_p$  và thời gian xác lập.

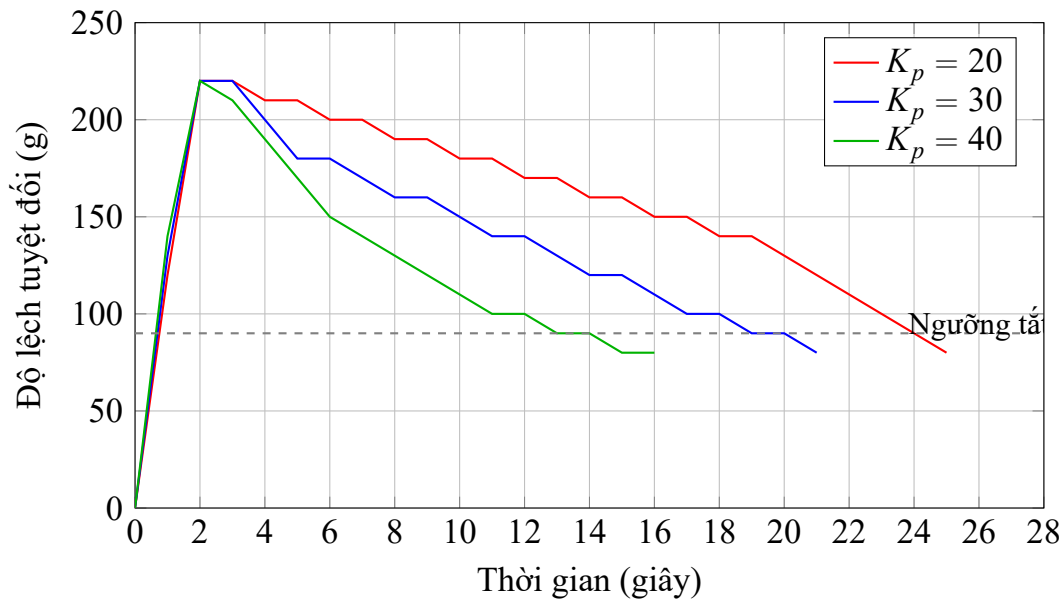


: So sánh thời gian xác lập với các giá trị  $K_p$  khác nhau

Hình 4.2.

#### 4.3.4. Biểu đồ diễn biến độ lệch theo thời gian

Hình 4.4 minh họa đường cong đáp ứng của hệ thống với ba giá trị  $K_p$ .



: Đường cong đáp ứng – Diễn biến độ lệch theo thời gian với các giá trị  $K_p$

Hình 4.4.

**Nhận xét:**  $K_p = 40$  cho thời gian đáp ứng nhanh nhất (khoảng 16 giây), nhanh hơn 36% so với  $K_p = 20$ . Tất cả các giá trị  $K_p$  đều đạt trạng thái cân bằng với độ lệch cuối

-80g (trong vùng Deadzone  $\pm 100$ g). Không có hiện tượng overshoot ở bất kỳ giá trị  $K_p$  nào, cho thấy hệ thống ổn định.

## 4.4. Thử nghiệm tổng hợp với nhiều kịch bản tải

Thử nghiệm này đánh giá khả năng đáp ứng của hệ thống với nhiều thay đổi tải liên tiếp, mô phỏng điều kiện vận hành thực tế của robot khi gấp/thả vật. Sử dụng  $K_p = 40$  (giá trị tối ưu từ thử nghiệm trước).

### 4.4.1. Kịch bản thử nghiệm

Bảng 4.8 trình bày timeline của thử nghiệm tổng hợp.

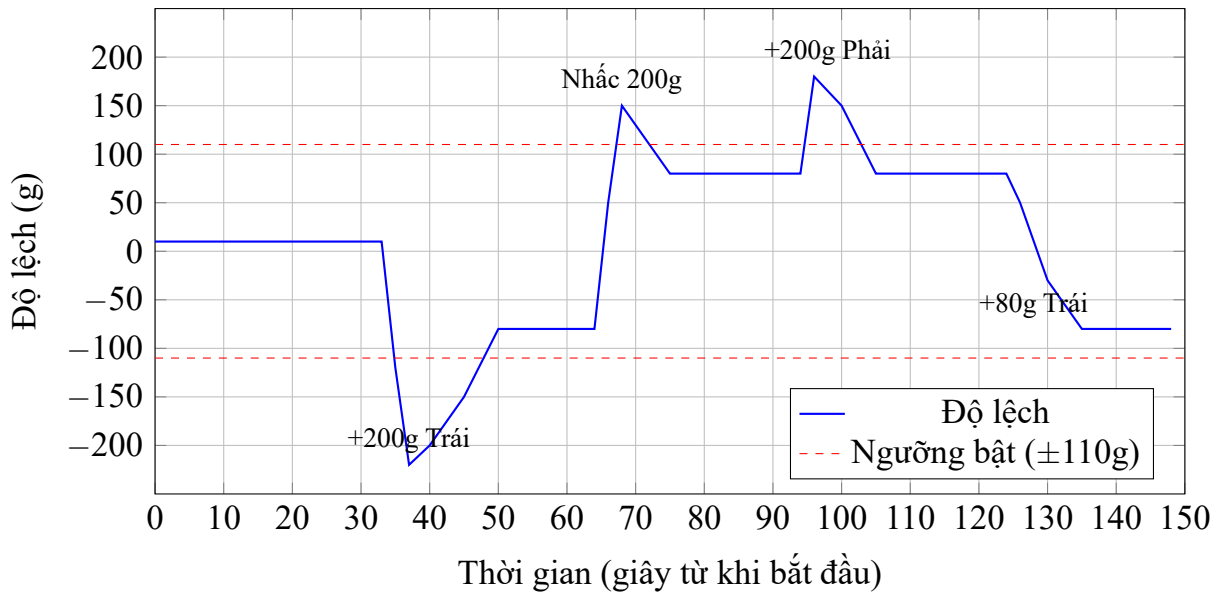
**Bảng 4.7.**

: Timeline thử nghiệm tổng hợp với nhiều kịch bản tải

Thời gian	Hành động	Độ lệch (g)	Trạng thái
13:38:46	Đặt vật ở TÂM sau Homing	L:100, R:110 $\rightarrow$ +10	[OK]
13:39:19	+200g bên TRÁI	-220 $\rightarrow$ -80	[RUN] $\rightarrow$ [OK]
khoảng 13:39:50	Nhấc 200g khỏi TRÁI	Motor đảo chiều	[RUN] $\rightarrow$ [OK]
khoảng 13:40:20	+200g bên PHẢI	Lệch dương lớn	[RUN] $\rightarrow$ [OK]
khoảng 13:40:50	+80g bên TRÁI (chồng tải)	Giảm lệch nhẹ	[RUN] $\rightarrow$ [OK]
khoảng 13:41:05	Nhấc 80g khỏi TRÁI	-110 $\rightarrow$ -80	[RUN] $\rightarrow$ [OK]
13:41:14	Kết thúc thử nghiệm	L:150, R:70 $\rightarrow$ -80	[OK]

### 4.4.2. Biểu đồ diễn biến thử nghiệm tổng hợp

Hình 4.6 minh họa diễn biến độ lệch và trạng thái hệ thống trong thử nghiệm tổng hợp.



: Diễn biến độ lệch trong thử nghiệm tổng hợp

Hình 4.6.

#### 4.4.3. Phân tích kết quả

##### Xác nhận hoạt động của cơ chế Hysteresis:

- Hệ thống chuyển sang [RUN] khi  $|Lch| > 110g$  (quan sát:  $-120g \rightarrow [RUN]$ )
- Hệ thống chuyển về [OK] khi  $|Lch| < 90g$  (quan sát:  $-80g \rightarrow [OK]$ )
- Khoảng cách 20g (từ 90g đến 110g) ngăn hiện tượng bật/tắt liên tục

##### Đánh giá khả năng xử lý:

- *Phát hiện thay đổi tải*: Chính xác, phản ứng ngay khi vượt ngưỡng
- *Xác định hướng điều chỉnh*: Đúng chiều (âm  $\rightarrow$  motor chạy phải, dương  $\rightarrow$  motor chạy trái)
- *Xử lý tải chồng*: Hoạt động tốt khi thêm 80g trong khi đang có 200g bên kia
- *Đảo chiều*: Mượt mà, không giật khi đổi hướng di chuyển

#### 4.5. Phân tích hiệu năng hệ thống

##### 4.5.1. Hiệu năng bộ điều khiển PID

Với bộ tham số PID đã tinh chỉnh ( $K_p = 40$ ,  $K_i = 0$ ,  $K_d = 0$ ), Bảng 4.10 tổng hợp các chỉ số hiệu năng.

**Bảng 4.9.**

: Tổng hợp hiệu năng bộ điều khiển PID

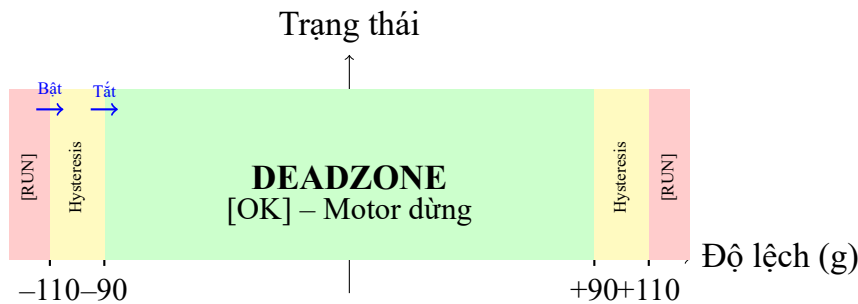
Thông số	Giá trị đo được
Thời gian xác lập (với $K_p = 40$ )	khoảng 16 giây
Độ vọt lố (Overshoot)	0%
Sai số xác lập	< 90g (trong Deadzone)
Độ ổn định	Ổn định, không dao động
Thời gian phát hiện thay đổi tải	< 100 ms

**Lý do chọn  $K_i = 0$  và  $K_d = 0$ :**

- Thành phần  $K_i$  không cần thiết vì Deadzone  $\pm 100g$  đã chấp nhận sai số nhỏ
- Thành phần  $K_d$  có thể khuếch đại nhiễu từ cảm biến (dù đã có Quantization 10g)
- Bộ điều khiển P thuần túy đủ đáp ứng yêu cầu với hệ thống có quán tính thấp

#### 4.5.2. Hiệu quả cơ chế Deadzone và Hysteresis

Hình 4.8 minh họa cơ chế Hysteresis trong hệ thống.



: Sơ đồ cơ chế Hysteresis với ngưỡng bật  $\pm 110g$  và ngưỡng tắt  $\pm 90g$

Hình 4.8.

**Lợi ích của Deadzone và Hysteresis:**

- *Tiết kiệm năng lượng*: Động cơ dừng hoàn toàn khi đã cân bằng (Speed = 0)
- *Giảm mài mòn cơ khí*: Không chạy qua chạy lại liên tục quanh điểm cân bằng
- *Tránh hunting*: Ngưỡng bật ( $> 110g$ ) khác ngưỡng tắt ( $< 90g$ ) ngăn dao động
- *Giảm tiếng ồn*: Hệ thống yên tĩnh khi ở trạng thái cân bằng

#### 4.5.3. Hiệu quả thiết kế Non-blocking

Bảng 4.12 so sánh hiệu năng giữa thiết kế Non-blocking và Blocking.

**Bảng 4.11.**

: So sánh hiệu năng Non-blocking và Blocking

Tiêu chí	Non-blocking	Blocking
Tần số gọi runSpeed()	khoảng 100 kHz	khoảng 40 Hz
Tốc độ motor tối đa	<b>8000 bước/s</b>	<100 bước/s
Độ mượt chuyển động	<b>Rất mượt</b>	Giật cục, rít
Chu kỳ PID	<b>20ms (ổn định)</b>	Không ổn định
Thời gian đọc Loadcell	Song song	Tuần tự (25ms block)

## 4.6. Đánh giá tổng thể

### 4.6.1. So sánh với mục tiêu đề ra

Bảng 4.14 so sánh kết quả thực nghiệm với các mục tiêu đề ra ban đầu.

**Bảng 4.13.**

: So sánh kết quả với mục tiêu đề ra

Mục tiêu	Yêu cầu	Kết quả	Đánh giá
Phát hiện độ lệch	Có	Có	<b>Đạt</b>
Tự động cân bằng	Có	Có	<b>Đạt</b>
Độ chính xác	$\pm 100g$	$\pm 80g$	<b>Vượt</b>
Thời gian đáp ứng	<30s	16–25s	<b>Đạt</b>
Hoạt động ổn định	>30 phút	>30 phút	<b>Đạt</b>
Không overshoot	Có	Có (0%)	<b>Đạt</b>

**Kết luận Chương 4:** Hệ thống đáp ứng đầy đủ các mục tiêu đề ra. Cảm biến loadcell hoạt động chính xác với sai số 0%. Bộ điều khiển PID với  $K_p = 40$  cho thời gian đáp ứng nhanh nhất (khoảng 16 giây). Cơ chế Deadzone và Hysteresis hoạt động hiệu quả, tránh hiện tượng hunting. Thiết kế Non-blocking là yếu tố then chốt giúp động cơ bước hoạt động mượt mà ở tốc độ cao.

## Chương 5.

# Kết luận và hướng phát triển

### 5.1. Kết luận

Khóa luận đã hoàn thành việc nghiên cứu, thiết kế và xây dựng hệ thống khung đế robot có khả năng tự động cân bằng trọng tâm sử dụng phương pháp khối lượng di động (Moving Mass). Dựa trên kết quả thực nghiệm chi tiết ở Chương 4, các kết luận được rút ra như sau:

#### 5.1.1. Về mặt lý thuyết

1. Phương pháp khối lượng di động kết hợp cảm biến Loadcell là giải pháp phù hợp cho bài toán cân bằng tĩnh và bán tĩnh khi tải thay đổi.
2. Bộ điều khiển P thuần túy ( $K_p = 40$ ,  $K_i = 0$ ,  $K_d = 0$ ) đủ đáp ứng yêu cầu với hệ thống có quán tính thấp và Deadzone hợp lý.
3. Cơ chế Hysteresis với khoảng cách 20g (ngưỡng bật 110g, ngưỡng tắt 90g) hiệu quả trong việc ngăn hiện tượng hunting.
4. Kiến trúc Non-blocking là bắt buộc để tích hợp cảm biến chậm (HX711 khoảng 10–80Hz) với điều khiển động cơ bước tốc độ cao (8000 bước/s).

#### 5.1.2. Về mặt thực tiễn

Bảng 5.2 tổng hợp các kết quả định lượng từ thực nghiệm.

**Bảng 5.1.**

: Tổng hợp kết quả định lượng từ thực nghiệm

Chỉ tiêu	Kết quả đạt được
Độ chính xác cảm biến	0% sai số với vật chuẩn 210g
Thời gian xác lập ( $K_p = 40$ )	khoảng 16 giây cho tải 200g
Độ lệch cuối cùng	< 90g (trong Deadzone $\pm 100$ g)
Overshoot	0% (không có)
Xử lý thay đổi tải liên tục	Thành công (6 lần trong 2.5 phút)
Hoạt động ổn định	> 30 phút liên tục



### 5.1.3. Đóng góp của đề tài

1. Đề xuất và triển khai thành công giải pháp cân bằng trọng tâm sử dụng Loadcell – phương pháp ít được nghiên cứu so với IMU.
2. Xây dựng kiến trúc phần mềm Non-blocking hoàn chỉnh, có thể tái sử dụng cho các dự án tương tự.
3. Cung cấp bộ dữ liệu thực nghiệm chi tiết để so sánh hiệu năng các giá trị  $K_p$ .
4. Tài liệu hóa đầy đủ quy trình thiết kế, triển khai và tinh chỉnh hệ thống.

## 5.2. Hạn chế

Bên cạnh các kết quả đạt được, đề tài còn một số hạn chế cần được khắc phục:

1. **Tốc độ đáp ứng:** Thời gian 16–25 giây có thể chưa đủ nhanh cho ứng dụng yêu cầu phản ứng tức thời. Nguyên nhân chính là vitme bước 2mm có tốc độ di chuyển tối đa khoảng 5mm/s.
2. **Giới hạn tải:** Với công thức  $m_{load,max} = 0.69 \times m_{slider}$  (hệ số từ tỷ lệ cánh tay đòn  $\frac{120}{175}$ ), đối trọng 200g chỉ bù được khoảng 137g.
3. **Phạm vi ứng dụng:** Chỉ cân bằng theo một trục (Trái-Phải), chưa mở rộng cho 2 trục.
4. **Giao diện:** Chưa có giao diện người dùng trực quan, chỉ giám sát qua Serial Monitor.
5. **Tích hợp:** Chưa tích hợp thực tế với robot di động để đánh giá hiệu quả trong điều kiện vận hành.

## 5.3. Hướng phát triển

Dựa trên kết quả và hạn chế của đề tài, các hướng phát triển tiếp theo bao gồm:

### 5.3.1. Cải tiến phần cứng

1. **Tăng tốc độ di chuyển:** Thay vitme bước 2mm bằng vitme 4–8mm hoặc hệ thống đai răng (belt drive), có thể tăng tốc độ đáp ứng 2–4 lần.
2. **Mở rộng 2 trục:** Thêm cơ cấu vitme + đối trọng theo trục Trước-Sau để cân bằng hoàn toàn trên mặt phẳng.
3. **Tăng khả năng chịu tải:** Sử dụng pin/acquy (1–2kg) làm đối trọng di động, có thể bù được độ lệch 690–1380g.

4. **Nâng cấp vi điều khiển:** Chuyển sang ESP32 hoặc STM32 cho tốc độ xử lý cao hơn, hỗ trợ WiFi/Bluetooth tích hợp.

### 5.3.2. Cải tiến phần mềm

1. **Thuật toán điều khiển nâng cao:** Nghiên cứu áp dụng LQR (Linear Quadratic Regulator) hoặc MPC (Model Predictive Control) để tối ưu đáp ứng.
2. **Bộ lọc Kalman:** Tích hợp bộ lọc Kalman để kết hợp dữ liệu từ Loadcell và IMU, tăng tốc độ phản hồi.
3. **Adaptive PID:** Phát triển PID tự thích nghi, tự động điều chỉnh  $K_p$  theo đặc tính tải.
4. **Giao diện người dùng:** Xây dựng ứng dụng di động hoặc giao diện web để giám sát và điều khiển từ xa.

### 5.3.3. Tích hợp và ứng dụng

1. **Tích hợp robot thực tế:** Lắp đặt khung cân bằng lên robot di động có cánh tay gấp để đánh giá hiệu quả trong điều kiện vận hành.
2. **Kết nối ROS:** Tích hợp vào Robot Operating System (ROS) cho điều khiển tổng thể hệ thống robot.
3. **Ứng dụng công nghiệp:** Phát triển phiên bản với độ bền cao, phù hợp môi trường nhà máy.

## 5.4. Lời kết

Khóa luận đã hoàn thành mục tiêu đề ra là xây dựng hệ thống khung đế robot có khả năng tự động cân bằng trọng tâm. Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, đáp ứng các yêu cầu kỹ thuật với độ chính xác  $\pm 80g$  và thời gian đáp ứng 16–25 giây.

Mặc dù còn một số hạn chế về tốc độ đáp ứng và phạm vi ứng dụng, hệ thống đã chứng minh tính khả thi của phương pháp khối lượng di động kết hợp cảm biến Loadcell. Các kiến thức và kinh nghiệm thu được trong quá trình thực hiện, từ thiết kế cơ khí, điện tử đến lập trình nhúng và điều khiển tự động, là nền tảng quý giá cho việc phát triển các hệ thống robot phức tạp hơn trong tương lai.

Em xin chân thành cảm ơn thầy giáo hướng dẫn ThS. Đặng Anh Việt đã tận tình chỉ bảo và hỗ trợ em trong suốt quá trình thực hiện khóa luận này.

## Tài liệu tham khảo

### Tiếng Việt

- [1] Phạm Thanh Vũ, *Nghiên cứu các luật điều khiển hiện đại trên mô hình robot cân bằng*, Luận văn Thạc sĩ, Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, 2015.

### Tiếng Anh

- [2] O. Kallhovd, “HX711\_ADC: Arduino library for the HX711 24-bit ADC for weight scales”, GitHub Repository, 2017. Truy cập: [https://github.com/olkal/HX711\\_ADC](https://github.com/olkal/HX711_ADC)
- [3] M. McCauley, “AccelStepper: Arduino library for stepper motors with acceleration/deceleration”, AirSpayce Documentation, 2023. Truy cập: <https://www.airspayce.com/mikem/arduino/AccelStepper/>
- [4] B. Beauregard, “PID\_v1: Arduino PID Library”, GitHub Repository, 2012. Truy cập: <https://github.com/br3ttb/Arduino-PID-Library>
- [5] Avia Semiconductor, “HX711: 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales”, Datasheet, Avia Semiconductor Co., Ltd., 2014.
- [6] Microchip Technology Inc., “ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash”, Datasheet, Microchip Technology Inc., 2018.
- [7] K. J. Åström and T. Hägglund, *Advanced PID Control*, ISA - The Instrumentation, Systems, and Automation Society, 2006.
- [8] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed., MIT Press, 2011.
- [9] MDPI Robotics, “Self-Balancing Mobile Robot: Design, Implementation, and Performance Analysis”, *MDPI Robotics*, vol. 6, no. 3, 2024.
- [10] ResearchGate, “Development of a self-balancing robot with a control moment gyroscope”, *ResearchGate Publication*, April 2018.

# Phụ lục A.

## Mã nguồn chương trình điều khiển

Dưới đây là mã nguồn hoàn chỉnh của chương trình điều khiển khung cân bằng trọng tâm:

```
1
2 #include <HX711_ADC.h>
3 #include <AccelStepper.h>
4 #include <PID_v1.h>
5
6
7
8
9 const int DOUT_PIN_LEFT = 4; const int SCK_PIN_LEFT = 5;
10 const int DOUT_PIN_PHA1 = 6; const int SCK_PIN_PHA1 = 7;
11 const int STEP_PIN = 8;
12 const int DIR_PIN = 9;
13 const int HOME_SWITCH_PIN = 10;
14
15
16 const int MICROSTEP = 16;          // Driver cài đặt 1/16
17 const int MOTOR_STEP = 200;        // Động cơ 1.8 độ
18 const int PITCH = 2;               // ướBc ren vitme 2mm
19
20 const float STEPS_PER_MM = (float)(MOTOR_STEP * MICROSTEP) / PITCH;
21
22
23 const long MAX_POS_RIGHT = 110.0 * STEPS_PER_MM; // ớGii ặnh ắPhi (+110mm
24 )
25 const long MAX_POS_LEFT = -120.0 * STEPS_PER_MM; // ớGii ặnh Trái (-120mm
26 )
27
28 const float MAX_SPEED_PID = 8000.0; // ốTc độ ốti ắa khi cân ắbng
29 const float MOTOR_ACCEL = 16000.0; // Gia ốtc
30 const float HOMING_SPEED_FAST = 8000.0;
31 const float HOMING_SPEED_SLOW = 6000.0;
32
33
34 const float MIN_SPEED_THRESHOLD = 4000.0; // ốTc độ ốti ếthiu ệhiu ắqu
35 const float HYSTERESIS_GAP = 10.0; // ắKhong ếtr 10g
36
37
38 double Kp = 40.0;
39 double Ki = 0;
40 double Kd = 0;
41 double Setpoint = 0, Input, Output;
```

```

40 PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);
41
42 float CALIB_LEFT = -53.13;
43 float CALIB_RIGHT = -55.36;
44
45 const float ACCEPTABLE_RANGE = 100.0; // Vùng ếcht +/- 100g
46 const float STEP_SIZE = 10.0; // Độ chia ỏnh ấnh: 10g
47
48
49 HX711_ADC LoadCellLeft(DOUT_PIN_LEFT, SCK_PIN_LEFT);
50 HX711_ADC LoadCellRight(DOUT_PIN_PHAI, SCK_PIN_PHAI);
51 AccelStepper stepper(AccelStepper::DRIVER, STEP_PIN, DIR_PIN);
52
53 unsigned long t_pid = 0;
54 unsigned long t_print = 0;
55 long currentPos;
56
57 float fL = 0; // ựLc bên Trái Đ(ấ làm tròn)
58 float fR = 0; // ựLc bên ảPhi Đ(ấ làm tròn)
59 float targetSpeed = 0;
60 float currentSpeed = 0; // ốTc độ ệhin ạti
61 boolean isBalancing = false;
62
63 float rawL, rawR, startThreshold, stopThreshold, rawSpeed;
64
65
66 void setup() {
67     Serial.begin(115200); delay(10);
68     Serial.println("--- KHOI DONG HE THONG ---");
69
70     pinMode(HOME_SWITCH_PIN, INPUT_PULLUP);
71
72     // 1. ỏKhi ạto Loadcell
73     LoadCellLeft.begin(); LoadCellRight.begin();
74     unsigned long stabilizingtime = 1000;
75     boolean _tare = true; // ấBt ộbuc ừtr bì ềv 0 khi ỏkhi động
76
77     LoadCellLeft.start(stabilizingtime, _tare);
78     LoadCellRight.start(stabilizingtime, _tare);
79
80     if (LoadCellLeft.getTareTimeoutFlag() || LoadCellRight.
        getTareTimeoutFlag()) {
81         Serial.println("LOI: Khong tim thay Loadcell! Kiem tra day ket noi."
            );
82         while(1); // ừDng ệh ốthng ếnu ốli
83     }
84
85     LoadCellLeft.setCalFactor(CALIB_LEFT);
86     LoadCellRight.setCalFactor(CALIB_RIGHT);
87     Serial.println("Loadcell san sang.");

```

```

88
89 myPID.SetMode(AUTOMATIC);
90 myPID.SetOutputLimits(-MAX_SPEED_PID, MAX_SPEED_PID);
91 myPID.SetSampleTime(20);
92
93 stepper.setMaxSpeed(MAX_SPEED_PID);
94 stepper.setAcceleration(MOTOR_ACCEL);
95
96 runHomingSequence();
97 }
98
99 void loop() {
100
101 LoadCellLeft.update();
102 LoadCellRight.update();
103 stepper.runSpeed();
104
105 if (millis() > t_pid + 20) {
106
107     rawL = LoadCellLeft.getData();
108     rawR = LoadCellRight.getData();
109     fL = ceil(rawL / STEP_SIZE) * STEP_SIZE;
110     fR = ceil(rawR / STEP_SIZE) * STEP_SIZE;
111     Input = fR - fL;
112
113
114     startThreshold = ACCEPTABLE_RANGE + HYSTERESIS_GAP;
115     stopThreshold = ACCEPTABLE_RANGE - HYSTERESIS_GAP;
116
117     if (!isBalancing) {
118         if (abs(Input) > startThreshold) {
119             isBalancing = true;
120             myPID.SetMode(AUTOMATIC);
121         }
122     }
123     else {
124         if (abs(Input) < stopThreshold) {
125             isBalancing = false;
126             Output = 0;
127             myPID.SetMode(MANUAL);
128         }
129     }
130
131     if (isBalancing) {
132         myPID.Compute();
133         rawSpeed = Output;
134
135         if (abs(rawSpeed) > 0.0 && abs(rawSpeed) < MIN_SPEED_THRESHOLD) {
136
137             if (rawSpeed > 0) {

```

```

138         targetSpeed = MIN_SPEED_THRESHOLD;
139     } else {
140         targetSpeed = -MIN_SPEED_THRESHOLD;
141     }
142 } else {
143
144     targetSpeed = rawSpeed;
145 }
146
147 } else {
148     targetSpeed = 0;
149 }
150
151 // d. 8Gii ăhn hành trình
152 currentPos = stepper.currentPosition();
153 if (currentPos <= MAX_POS_LEFT && targetSpeed < 0) targetSpeed = 0;
154 if (currentPos >= MAX_POS_RIGHT && targetSpeed > 0) targetSpeed = 0;
155
156 stepper.setSpeed(targetSpeed);
157 currentSpeed = targetSpeed;
158 t_pid = millis();
159 }
160
161 if (millis() > t_print + 100) {
162     Serial.print("L: "); Serial.print(fL, 0);
163     Serial.print(" | R: "); Serial.print(fR, 0);
164     Serial.print(" | Lech: "); Serial.print(Input, 0);
165     Serial.print(" | Spd: "); Serial.print(currentSpeed, 0);
166
167     if (isBalancing) {
168         Serial.println(" [RUN]");
169     } else {
170         Serial.println(" [OK]");
171     }
172
173     t_print = millis();
174 }
175 }
176
177 void runHomingSequence() {
178     Serial.println("[HOMING] Bat dau ve Home...");
179     stepper.setAcceleration(MOTOR_ACCEL);
180
181     stepper.setSpeed(HOMING_SPEED_FAST);
182     while (digitalRead(HOME_SWITCH_PIN) == HIGH) { stepper.runSpeed(); }
183     stepper.stop();
184     stepper.setCurrentPosition(MAX_POS_RIGHT); // Gán ốmc ạtm
185
186     Serial.println("[HOMING] Lui ra...");
187     stepper.moveTo(MAX_POS_RIGHT - (5.0 * STEPS_PER_MM));

```

```

188 while (stepper.distanceToGo() != 0) { stepper.run(); }
189
190 if (digitalRead(HOME_SWITCH_PIN) == LOW) {
191     Serial.println("LOI: Cong tac hanh trinh bi ket!");
192     while(1);
193 }
194
195 stepper.setSpeed(HOMING_SPEED_SLOW);
196 while (digitalRead(HOME_SWITCH_PIN) == HIGH) { stepper.runSpeed(); }
197 stepper.stop();
198
199 stepper.setCurrentPosition(MAX_POS_RIGHT);
200 stepper.setSpeed(0);
201
202 Serial.println("[HOMING] Ve Tam (0)...");
203 stepper.moveTo(0);
204 while (stepper.distanceToGo() != 0) {stepper.run();}
205
206 Serial.println("[HOMING] Da ve tam. Cho on dinh 5s...");
207
208 unsigned long waitStart = millis();
209 while (millis() - waitStart < 5000) {}
210
211 Serial.println("[HOMING] Hoan tat. San sang.");
212 }

```

Listing A.1: Chương trình điều khiển khung cân bằng trọng tâm