**TRƯỜNG ĐẠI HỌC BÌNH DƯƠNG**

**KHOA CÔNG NGHỆ THÔNG TIN, ROBOT**

**VÀ TRÍ TUỆ NHÂN TẠO**

---oOo---

**TIỂU LUẬN MÔN HỌC**

**LẬP TRÌNH HỆ THỐNG**

**Tên đề tài:**

**ROBOT TỰ CÂN BẰNG 2 BÁNH**

**Giáo viên hướng dẫn:** Lê Duy Hùng

**Sinh viên thực hiện:** Nguyễn Trung Kiên – 22050061

Lê Hữu Sơn - 22050058

**Thành phố Hồ Chí Minh, Tháng 8/2025**

**TRƯỜNG ĐẠI HỌC BÌNH DƯƠNG**

**KHOA CÔNG NGHỆ THÔNG TIN, ROBOT**

**VÀ TRÍ TUỆ NHÂN TẠO**

---oOo---

**TIỂU LUẬN MÔN HỌC**

**LẬP TRÌNH HỆ THỐNG**

**Tên đề tài:**

**ROBOT TỰ CÂN BẰNG 2 BÁNH**

**Giáo viên hướng dẫn:** Lê Duy Hùng

**Sinh viên thực hiện:** Nguyễn Trung Kiên - 22050061

Lê Hữu Sơn – 22050058

**Thành phố Hồ Chí Minh, Tháng 8/2025**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**GIẢNG VIÊN HƯỚNG DẪN**

(*Ký và ghi rõ họ tên*)

# **LỜI CẢM ƠN**



Với tất cả tình cảm chân thành và lòng biết ơn chúng em kính gửi đến:

Ban giám hiệu nhà trường cùng các thầy cô đã và đang công tác giảng dạy tại trường Đại học Bình Dương lời chúc sức khỏe dồi dào và thành công trong công việc. Nhà trường đã tạo cho em có điều kiện thâm nhập thực tế để so sánh đối chiếu với thực tiễn từ những gì mình đã học từ sách vở, đặc biệt là các thầy cô khoa Công nghệ thông tin đã dạy dỗ và truyền đạt cho em những kiến thức bổ ích, đã tận tình giúp đỡ em trong thời gian qua. Đồng thời em xin gửi tới các thầy cô giáo hướng dẫn lời cảm ơn chân thành vì thầy cô đã bỏ thời gian quý báu của mình để hướng dẫn bài tiểu luận của chúng em được hoàn thành tốt.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Lê Duy Hùng, giảng viên môn Lập trình hệ thống. Trong suốt quá trình học tập và thực hiện đề tài Robot cân bằng 2 bánh, chúng em đã có cơ hội tiếp cận sâu hơn với các kiến thức về lập trình hệ thống, điều khiển đa luồng, xử lý cảm biến và thuật toán điều khiển trong thực tế.

Sự tận tình, nghiêm khắc nhưng đầy cảm hứng từ thầy đã giúp chúng em nâng cao tư duy lập trình, rèn luyện khả năng phân tích, giải quyết vấn đề và vận dụng kiến thức lý thuyết vào việc xây dựng một hệ thống robot thực tiễn.

Những kiến thức tiếp thu ở trường và trong quá trình thực hiện đề tài sẽ là tiền đề cho bản thân em bước vào thực tế trong công tác sau này. Chúng em xin hứa sẽ vận dụng và phát huy những kiến thức này để phục vụ đất nước trong giai đoạn nước ta bước vào công nghiệp hóa, hiện đại hóa.

# **MỤC LỤC**

[**LỜI CẢM ƠN** i](#_Toc207160629)

[**MỤC LỤC** ii](#_Toc207160630)

[**DANH MỤC HÌNH ẢNH** iv](#_Toc207160631)

[**DANH MỤC BẢNG BIỂU** v](#_Toc207160632)

[**CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN** 1](#_Toc207160633)

[**1.1.** **Lý do thực hiện đề tài** 1](#_Toc207160634)

[**1.2.** **Các hệ thống tương tự** 1](#_Toc207160635)

[**1.3.** **Phát biểu bài toán** 2](#_Toc207160636)

[**1.4.** **Kết quả cần đạt** 2](#_Toc207160637)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 4](#_Toc207160638)

[**2.1. Cơ sở lý thuyết** 4](#_Toc207160639)

[***2.1.1. Nguyên lý con lắc ngược*** 4](#_Toc207160640)

[***2.1.2. Cơ chế giữ thăng bằng*** 4](#_Toc207160641)

[***2.1.3. Điều khiển phản hồi (Feedback Control)*** 4](#_Toc207160642)

[**2.2. Công nghệ sử dụng** 4](#_Toc207160643)

[***2.2.1. Phần cứng*** 4](#_Toc207160644)

[***2.2.2. Phần mềm*** 6](#_Toc207160645)

[**2.3. Cách tiếp cận** 7](#_Toc207160646)

[***2.3.1. Phương pháp nghiên cứu*** 7](#_Toc207160647)

[***2.3.2. Quy trình thực hiện*** 7](#_Toc207160648)

[**CHƯƠNG 3: MÔ HÌNH HỆ THỐNG** 8](#_Toc207160649)

[**3.1. Các yêu cầu chức năng** 8](#_Toc207160650)

[***3.1.1. Danh sách Actor*** 8](#_Toc207160651)

[***3.1.2. Sơ đồ ngữ cảnh*** 8](#_Toc207160652)

[***3.1.3. Danh sách usecase*** 8](#_Toc207160653)

[***3.1.4. Sơ đồ Usecase chính*** 9](#_Toc207160654)

[**3.2 Các yêu cầu phi chức năng** 10](#_Toc207160655)

[**3.3 Mô hình hệ thống** 10](#_Toc207160656)

[***3.3.1. Mô hình tổng quát*** 10](#_Toc207160657)

[***3.3.2. Kiến trúc phần mềm*** 11](#_Toc207160658)

[**3.4. Mô hình xử lý** 12](#_Toc207160659)

[***3.4.1. Sơ đồ tuần tự*** 12](#_Toc207160660)

[***3.4.2. Sơ đồ hoạt động*** 13](#_Toc207160661)

[**3.5. Thiết kế giao diện người dùng** 13](#_Toc207160662)

[**3.6. Thiết kế chi tiết** 13](#_Toc207160663)

[***3.6.1. Task FreeRTOS*** 13](#_Toc207160664)

[***3.6.2. Thiết kế dữ liệu*** 14](#_Toc207160665)

[**CHƯƠNG 4: KẾT QUẢ VÀ THỰC NGHIỆM** 15](#_Toc207160666)

[**4.1. Kịch bản thử nghiệm** 15](#_Toc207160667)

[**4.2. Quá trình thực nghiệm** 15](#_Toc207160668)

[**4.3.1. Thử nghiệm khởi động** 15](#_Toc207160669)

[**4.3.2. Thử nghiệm cân bằng tĩnh** 16](#_Toc207160670)

[**4.3.3. Thử nghiệm cân bằng động** 16](#_Toc207160671)

[**4.3.4. Thử nghiệm di chuyển** 16](#_Toc207160672)

[**4.3. Xử lý ngoại lệ** 16](#_Toc207160673)

[**CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN** 17](#_Toc207160674)

[**5.1 Kết quả đạt được** 17](#_Toc207160675)

[**5.2 Hạn chế** 17](#_Toc207160676)

[**5.3 Hướng phát triển** 18](#_Toc207160677)

[**Tài liệu tham khảo** 19](#_Toc207160678)

# **DANH MỤC HÌNH ẢNH**

**Hình 3.1:** Sơ đồ ngữ cảnh 8

**Hình 3.2:** Sơ đồ usecase chính 9

**Hình 3.3:** Mô hình tổng quát 10

**Hình 3.4:** Kiến trúc phần mềm 11

**Hình 3.5:** Sơ đồ tuần tự 12

**Hình 3.6:** Sơ đồ hoạt động 13

# **DANH MỤC BẢNG BIỂU**

**Bảng 1.1.** Bảng kết quả cần đạt 3

**Bảng 3.1.** Bảng thiết kế dữ liệu 15

**Bảng 5.1.** Bảng kết quả đạt được 17

# **CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN**

* 1. **Lý do thực hiện đề tài**

Trong những năm gần đây, sự phát triển của công nghệ robot đã và đang trở thành một trong những lĩnh vực nghiên cứu quan trọng, góp phần vào cuộc cách mạng công nghiệp 4.0. Trong đó, **robot tự cân bằng 2 bánh** là một mô hình tiêu biểu thể hiện sự kết hợp giữa cơ khí, điện tử và điều khiển tự động.

Robot 2 bánh có khả năng **giữ thăng bằng trên mặt phẳng hẹp** nhờ việc sử dụng cảm biến và thuật toán điều khiển phản hồi. Nguyên lý này tương tự như việc con người giữ thăng bằng khi đi xe đạp: luôn điều chỉnh góc nghiêng để không bị ngã.

Việc nghiên cứu, thiết kế và chế tạo robot tự cân bằng 2 bánh có ý nghĩa:

* Giúp sinh viên tiếp cận các kiến thức liên ngành (nhúng, điều khiển, tự động hóa, lập trình thời gian thực).
* Có thể mở rộng thành nền tảng cho xe tự hành, robot vận chuyển, hoặc các hệ thống robot cá nhân thông minh.
* Tăng tính ứng dụng thực tế trong giáo dục, nghiên cứu, và công nghiệp.
  1. **Các hệ thống tương tự**

Trên thế giới, nhiều nghiên cứu và sản phẩm thương mại đã được phát triển dựa trên nguyên lý robot tự cân bằng:

* **Segway PT**: phương tiện cá nhân hai bánh nổi tiếng, ứng dụng nguyên lý tự cân bằng để di chuyển ổn định.
* **Hoverboard**: xe điện hai bánh cho cá nhân, điều khiển bằng cảm biến nghiêng.
* Các dự án **DIY Robot Balancing** trên nền Arduino/ESP32 với MPU6050 đã được cộng đồng mã nguồn mở triển khai rộng rãi.

Tuy nhiên, các hệ thống này vẫn tồn tại một số hạn chế:

* Giá thành cao (đối với sản phẩm thương mại).
* Chưa phổ biến trong giáo dục tại Việt Nam.
* Một số nghiên cứu tập trung nhiều vào cơ khí mà chưa khai thác sâu về **lập trình hệ điều hành thời gian thực (FreeRTOS)** để quản lý đa nhiệm hiệu quả.

Đề tài này mang tính mới ở chỗ:

* Ứng dụng **FreeRTOS** trong điều khiển robot 2 bánh.
* Phân chia rõ ràng các tác vụ (task đọc cảm biến, task PID, task motor, task debug).
* Giúp sinh viên làm quen với lập trình nhúng chuyên nghiệp hơn so với cách lập trình tuần tự thông thường.
  1. **Phát biểu bài toán**

**Mục tiêu:**

* Xây dựng một robot 2 bánh có khả năng tự cân bằng theo phương thẳng đứng.
* Ứng dụng FreeRTOS để quản lý và xử lý đa nhiệm.
* Triển khai điều khiển PID để giữ robot ổn định.

**Phạm vi:**

* Phần cứng sử dụng: ESP32, MPU6050, driver L298N, động cơ DC, pin, khung cơ khí.
* Phần mềm: FreeRTOS trên nền Arduino IDE hoặc PlatformIO.
* Robot hoạt động trong môi trường phẳng, không yêu cầu nhận dạng vật cản.

**Giả định và ràng buộc:**

* Robot chỉ hoạt động ổn định khi pin còn đủ điện áp.
* Chưa tối ưu cho bề mặt gồ ghề hoặc môi trường có nhiều nhiễu.
* Thời gian phát triển hạn chế nên tập trung vào chức năng giữ thăng bằng.
  1. **Kết quả cần đạt**

Dự kiến sản phẩm và tiêu chí đánh giá như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Kết quả cần đạt** | **Tiêu chí đánh giá** | **Ứng dụng** |
| 1 | Robot 2 bánh hoàn chỉnh (cơ khí + điện tử) | Robot có thể đứng thẳng trên 2 bánh ít nhất 30 giây | Mô hình demo giáo dục |
| 2 | Tích hợp cảm biến MPU6050 và đọc dữ liệu ổn định | Sai số đo góc nghiêng < 2° | Cơ sở cho điều khiển PID |
| 3 | Thuật toán PID cân bằng | Robot không ngã trong phạm vi ±15° | Ứng dụng điều khiển phản hồi |
| 4 | FreeRTOS với 4 task chính (Sensor, PID, Motor, Debug) | Task hoạt động song song, không xung đột | Học tập lập trình RTOS |
| 5 | Báo cáo và mã nguồn hoàn chỉnh | Có thể tái sử dụng, mở rộng nghiên cứu | Hỗ trợ sinh viên khác |

**Bảng 1.1. Bảng kết quả cần đạt**

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1. Cơ sở lý thuyết**

### ***2.1.1. Nguyên lý con lắc ngược***

Con lắc ngược là một trong những mô hình kinh điển trong điều khiển tự động. Khác với con lắc thường có trọng tâm phía dưới, con lắc ngược có trọng tâm phía trên điểm tựa nên luôn ở trạng thái không ổn định.

Trong bài toán robot 2 bánh, thân robot chính là **thanh con lắc**, còn hai bánh xe đóng vai trò **điểm tựa động**. Để robot không ngã, động cơ bánh xe phải liên tục quay để bù lại hướng nghiêng của thân.

Ví dụ: nếu robot ngả về phía trước, bánh xe cần quay về phía trước nhanh hơn để đưa trọng tâm trở lại. Ngược lại, nếu robot ngả về phía sau, bánh xe phải quay ngược lại.

### ***2.1.2. Cơ chế giữ thăng bằng***

* **Bước 1:** Cảm biến MPU6050 liên tục đo góc nghiêng (pitch, roll).
* **Bước 2:** Bộ điều khiển PID nhận giá trị góc, so sánh với giá trị mong muốn (90° thẳng đứng).
* **Bước 3:** Sai số được tính toán thành tín hiệu điều khiển động cơ.
* **Bước 4:** Động cơ quay theo hướng ngược lại với hướng nghiêng.
* **Bước 5:** Chu trình lặp lại với tần số cao (100–200 lần/giây).

### ***2.1.3. Điều khiển phản hồi (Feedback Control)***

Điều khiển phản hồi giúp hệ thống tự điều chỉnh dựa trên sự chênh lệch giữa giá trị đầu ra thực tế và giá trị mong muốn. Đây là nguyên tắc cốt lõi trong robot cân bằng.

Ví dụ: nếu robot cần giữ ở 90° nhưng thực tế đo được 85°, sai số = 5°. Bộ điều khiển sẽ ra lệnh cho động cơ quay để giảm sai số này về 0.

## **2.2. Công nghệ sử dụng**

***2.2.1. Phần cứng***

**a) Vi điều khiển ESP32 Devkit V1**

ESP32 là vi điều khiển thế hệ mới do **Espressif Systems** phát triển, tích hợp WiFi và Bluetooth, có hiệu năng mạnh mẽ, tài nguyên bộ nhớ lớn, và đặc biệt là hỗ trợ **FreeRTOS** ngay trong SDK.

**Đặc điểm nổi bật của ESP32:**

* CPU **dual-core Xtensa LX6**, tốc độ 240 MHz.
* RAM: 520 KB, Flash: 4 MB (tùy module).
* Hỗ trợ giao tiếp phong phú: **I2C, SPI, UART, PWM, ADC, DAC**.
* WiFi 802.11 b/g/n và Bluetooth 4.2 BLE.
* Hỗ trợ lập trình bằng Arduino IDE, PlatformIO hoặc ESP-IDF.

**Ứng dụng trong đề tài:**

* Chạy hệ điều hành **FreeRTOS** để quản lý đa nhiệm.
* Giao tiếp với cảm biến MPU6050 qua I2C.
* Xuất tín hiệu PWM để điều khiển module L298N → động cơ DC.
* Có thể mở rộng điều khiển qua smartphone nhờ WiFi/Bluetooth.

**b) Cảm biến MPU6050**

MPU6050 là module **IMU (Inertial Measurement Unit)** tích hợp **3 trục gia tốc kế + 3 trục con quay hồi chuyển**, cho phép đo:

* Gia tốc tuyến tính (m/s²).
* Vận tốc góc (°/s).
* Góc nghiêng (pitch, roll, yaw) sau khi xử lý dữ liệu.

**Thông số kỹ thuật:**

* Nguồn hoạt động: 3.3V – 5V.
* Giao tiếp: I2C (SDA, SCL).
* Dải đo gia tốc: ±2g, ±4g, ±8g, ±16g.
* Dải đo gyro: ±250, ±500, ±1000, ±2000 °/s.

**Ứng dụng trong đề tài:**

* Đo góc nghiêng của robot theo thời gian thực.
* Cung cấp dữ liệu cho bộ điều khiển PID.
* Kết hợp **bộ lọc Kalman** hoặc **Complementary filter** để giảm nhiễu.

**c) Mạch điều khiển động cơ L298N**

L298N là IC cầu H kép, cho phép điều khiển hai động cơ DC độc lập.

**Đặc điểm:**

* Điện áp hoạt động: 5–35V.
* Dòng tải: 2A mỗi kênh.
* Hỗ trợ điều khiển tốc độ bằng **PWM**.
* Có khả năng đảo chiều động cơ.

**Ứng dụng trong đề tài:**

* Nhận tín hiệu PWM từ ESP32.
* Xuất dòng điều khiển 2 động cơ DC.
* Cho phép robot di chuyển tiến, lùi, hoặc đứng yên để cân bằng.

**d) Động cơ DC giảm tốc + bánh xe**

* Động cơ DC giảm tốc được sử dụng, tích hợp hộp số để tạo **mô-men xoắn lớn** và **tốc độ quay phù hợp** (~100–200 vòng/phút).
* Hai động cơ gắn hai bánh xe, đảm nhận vai trò giữ thăng bằng và di chuyển.

**e) Nguồn cung cấp**

* Pin Li-ion 18650 (2 cell, 3.7V).
* Có mạch sạc và mạch bảo vệ (BMS) để đảm bảo an toàn.
* Pin cung cấp cho cả vi điều khiển và động cơ qua L298N.

***2.2.2. Phần mềm***

**a) FreeRTOS**

FreeRTOS là hệ điều hành thời gian thực nhẹ, hỗ trợ:

* **Task:** chia chương trình thành nhiều nhiệm vụ nhỏ.
* **Scheduler:** lập lịch thực thi theo mức ưu tiên.
* **Queue, Mutex, Semaphore:** đồng bộ dữ liệu giữa các task.

Ứng dụng:

* Task đọc dữ liệu từ MPU6050.
* Task tính toán PID.
* Task điều khiển động cơ.
* Task hiển thị/debug dữ liệu.

**b) Arduino IDE**

* Miễn phí, dễ sử dụng, cộng đồng hỗ trợ rộng lớn.
* Nhiều thư viện sẵn có: MPU6050, PID, FreeRTOS.
* Hỗ trợ lập trình trực tiếp ESP32 qua USB.

**c) Thư viện phần mềm hỗ trợ**

* **MPU6050 Library (I2Cdevlib, Adafruit MPU6050):** đọc dữ liệu cảm biến.
* **Kalman Filter / Complementary Filter Library:** lọc dữ liệu cảm biến.
* **PID Library (Arduino PID):** triển khai thuật toán điều khiển PID.
* **FreeRTOS Library (ESP32 có sẵn, Arduino phải cài thêm):** xử lý đa nhiệm.

## **2.3. Cách tiếp cận**

***2.3.1. Phương pháp nghiên cứu***

**Lý thuyết:** tìm hiểu con lắc ngược, PID, FreeRTOS.

**Mô phỏng:** sử dụng Matlab/Proteus (nếu có) để mô phỏng mô hình điều khiển.

**Thực nghiệm:** xây dựng robot mẫu, thử nghiệm thực tế, điều chỉnh tham số PID.

***2.3.2. Quy trình thực hiện***

1. Khảo sát đề tài và chọn phần cứng.
2. Thiết kế sơ đồ nguyên lý, sơ đồ khối hệ thống.
3. Lập trình trên Arduino IDE + FreeRTOS.
4. Tích hợp các task: sensor → PID → motor.
5. Thử nghiệm, điều chỉnh thông số PID đến khi robot giữ cân bằng ổn định.
6. Đánh giá kết quả, ghi nhận hạn chế.

# **CHƯƠNG 3: MÔ HÌNH HỆ THỐNG**

**3.1. Các yêu cầu chức năng**

***3.1.1. Danh sách Actor***

* **Người dùng (User):** sinh viên hoặc kỹ sư vận hành robot.
* **Robot:** thực hiện giữ thăng bằng.
* **Cảm biến MPU6050:** cung cấp dữ liệu góc nghiêng.
* **Driver L298N + Động cơ DC:** thực hiện chuyển động.
* **Hệ thống Debug (PC/Serial Monitor):** hiển thị dữ liệu phục vụ giám sát.

***3.1.2. Sơ đồ ngữ cảnh***

**A diagram of a car

AI-generated content may be incorrect.**

**Hình 3.1. Sơ đồ ngữ cảnh**

### ***3.1.3. Danh sách usecase***

1. Bật/tắt robot.
2. Robot tự giữ thăng bằng.
3. Đọc dữ liệu cảm biến MPU6050.
4. Tính toán PID.
5. Điều khiển động cơ thông qua L298N.
6. Hiển thị dữ liệu PID, cảm biến lên Serial Monitor.

### ***3.1.4. Sơ đồ Usecase chính***

***A diagram of a flowchart

AI-generated content may be incorrect.***

**Hình 3.2. Sơ đồ usecase chính**

**3.2 Các yêu cầu phi chức năng**

* Hiệu năng: robot phải xử lý dữ liệu cảm biến với chu kỳ ≤ 10ms.
* Ổn định: robot giữ cân bằng trong góc nghiêng ±15°.
* Khả năng chịu lỗi: robot không bị treo khi một task bị quá tải.
* Khả năng mở rộng: có thể tích hợp thêm Bluetooth/WiFi mà không ảnh hưởng cân bằng.
* Dễ sử dụng: bật nguồn là robot tự động hoạt động.

## **3.3 Mô hình hệ thống**

### ***3.3.1. Mô hình tổng quát***

A diagram of a task

AI-generated content may be incorrect.

**Hình 3.3. Mô hình tổng quát**

### ***3.3.2. Kiến trúc phần mềm***

A diagram of a system

AI-generated content may be incorrect.

**Hình 3.4. Kiến trúc phần mềm**

## **3.4. Mô hình xử lý**

### ***3.4.1. Sơ đồ tuần tự***

**A diagram of a task

AI-generated content may be incorrect.**

**Hình 3.5. Sơ đồ tuần tự**

### ***3.4.2. Sơ đồ hoạt động***

**A diagram of a process

AI-generated content may be incorrect.**

**Hình 3.6. Sơ đồ hoạt động**

## **3.5. Thiết kế giao diện người dùng**

Trong đề tài này, giao diện chính là Serial Monitor: hiển thị giá trị góc, giá trị PID, và trạng thái động cơ.

## **3.6. Thiết kế chi tiết**

### ***3.6.1. Task FreeRTOS***

* Task 1 (Sensor): đọc dữ liệu MPU6050 mỗi 5–10ms.
* Task 2 (PID): xử lý sai số, tính toán giá trị PID.
* Task 3 (Motor): xuất PWM điều khiển động cơ.
* Task 4 (Debug): hiển thị dữ liệu trên Serial.

### ***3.6.2. Thiết kế dữ liệu***

|  |  |  |  |
| --- | --- | --- | --- |
| **Task sinh ra** | **Dữ liệu tạo ra** | **Mô tả** | **Task sử dụng** |
| **Task Sensor** | Dữ liệu cảm biến (góc nghiêng, vận tốc góc) | Đọc từ MPU6050 qua I2C, tính toán góc nghiêng (pitch, roll) | Task PID  Task Debug |
| **Task PID** | Sai số (Error) | Hiệu giữa setpoint (90°) và góc thực tế | Nội bộ Task PID (dùng tính PID) |
| **Task PID** | PID Output | Kết quả tính toán điều khiển (giá trị PWM) | Task Motor |
| **Task Motor** | Tín hiệu PWM | Chuỗi xung PWM điều khiển tốc độ và chiều quay của động cơ | Driver L298N (phần cứng) |
| **Task Debug** | Dữ liệu Debug (Sensor, Error, PID Output, PWM) | Thông tin tổng hợp để hiển thị, theo dõi, giám sát | PC/Serial Monitor (người dùng) |

**Bảng 3.1. Bảng thiết kế dữ liệu**

# **CHƯƠNG 4: KẾT QUẢ VÀ THỰC NGHIỆM**

## **4.1. Kịch bản thử nghiệm**

Để đánh giá khả năng giữ thăng bằng của robot, nhóm xây dựng các kịch bản thử nghiệm sau:

1. **Thử nghiệm khởi động:** Robot được bật nguồn, kiểm tra xem có thể tự đứng thẳng mà không cần tác động ngoài.
2. **Thử nghiệm cân bằng tĩnh:** Robot đứng yên trên mặt phẳng phẳng, kiểm tra độ ổn định trong 60 giây.
3. **Thử nghiệm cân bằng động:** Robot bị đẩy nhẹ về phía trước/sau, kiểm tra khả năng tự cân bằng lại.
4. **Thử nghiệm di chuyển:** Robot di chuyển tiến/lùi ngắn để kiểm tra tính ổn định trong quá trình vận hành.

## **4.2. Quá trình thực nghiệm**

Bước 1: Lắp ráp hệ thống phần cứng (ESP32, MPU6050, L298N, động cơ, pin).

Bước 2: Nạp chương trình điều khiển FreeRTOS vào ESP32.

Bước 3: Sử dụng Serial Monitor để theo dõi dữ liệu cảm biến (góc nghiêng, sai số, PID output).

Bước 4: Điều chỉnh thông số PID (Kp, Ki, Kd) cho đến khi robot giữ thăng bằng ổn định.

Bước 5: Thực hiện các kịch bản thử nghiệm, ghi nhận dữ liệu.

**4.3. Kết quả thử nghiệm**

### **4.3.1. Thử nghiệm khởi động**

* Khi bật nguồn, robot cần khoảng **2–3 giây** để hệ thống MPU6050 khởi tạo và hiệu chỉnh cảm biến.
* Sau khi khởi tạo, robot có thể tự đứng thẳng mà không cần tác động ngoài.
* Trong quá trình này, dao động ban đầu khá lớn (±10°) nhưng sau đó giảm xuống ổn định còn **±3°**.

### **4.3.2. Thử nghiệm cân bằng tĩnh**

* Robot được đặt trên mặt phẳng phẳng, đứng yên trong 60 giây.
* Góc dao động duy trì ở mức **±2.5° đến ±3°** quanh vị trí cân bằng.
* Hệ thống PID phản hồi nhanh, không có hiện tượng trễ rõ rệt.
* Kết quả cho thấy robot giữ thăng bằng ổn định trong thời gian dài.

### **4.3.3. Thử nghiệm cân bằng động**

* Khi robot bị đẩy nhẹ về phía trước (góc lệch 5°), robot hồi phục về vị trí cân bằng trong khoảng **0.4–0.6 giây**.
* Khi bị đẩy về phía sau với góc lệch 10°, thời gian hồi phục lâu hơn, khoảng **1.0–1.2 giây**.
* Nếu robot bị nghiêng quá 15°, robot không thể tự cân bằng và sẽ ngã.
* Điều này chứng tỏ bộ điều khiển hoạt động tốt trong phạm vi nhiễu nhỏ và trung bình.

### **4.3.4. Thử nghiệm di chuyển**

* Robot có thể di chuyển tiến hoặc lùi với quãng đường **20–30 cm** mà vẫn giữ được trạng thái cân bằng.
* Khi thay đổi vận tốc đột ngột, robot có hiện tượng dao động mạnh hơn trong 1–2 giây rồi mới ổn định lại.
* Trong các thử nghiệm di chuyển ngắn, robot duy trì được sự cân bằng, tuy nhiên ở tốc độ cao thì dao động tăng lên, cho thấy giới hạn của thuật toán PID hiện tại.

## **4.3. Xử lý ngoại lệ**

* **Trường hợp robot ngã:** Hệ thống tự động dừng động cơ để tránh hư hỏng.
* **Trường hợp cảm biến mất tín hiệu:** FreeRTOS sẽ đưa giá trị mặc định (0°) và cảnh báo qua Serial Monitor.
* **Trường hợp pin yếu:** Động cơ không đủ dòng → robot mất cân bằng. Giải pháp: thêm mạch giám sát điện áp pin.

# **CHƯƠNG 5: KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN**

**5.1 Kết quả đạt được**

|  |  |
| --- | --- |
| **Mục tiêu đề ra** | **Kết quả đạt được** |
| Xây dựng mô hình robot hai bánh có khả năng giữ thăng bằng | Đã thiết kế và chế tạo robot tự cân bằng 2 bánh sử dụng ESP32, MPU6050, L298N, động cơ DC |
| Thu thập dữ liệu cảm biến góc nghiêng, vận tốc góc | Đã đọc thành công dữ liệu từ MPU6050 qua giao tiếp I2C, xử lý trong Task Sensor |
| Áp dụng thuật toán PID để điều khiển cân bằng | PID hoạt động tốt, robot giữ cân bằng trong phạm vi ±2–3° |
| Robot có khả năng tự đứng thẳng sau khi khởi động | Robot khởi động và tự đứng sau 2–3 giây, ổn định sau đó |
| Kiểm tra khả năng hồi phục khi bị tác động bên ngoài | Robot hồi phục tốt với góc lệch ≤10°, thời gian hồi phục <1.2 giây |
| Robot có thể di chuyển tiến/lùi mà vẫn giữ được trạng thái cân bằng | Đã di chuyển ngắn (20–30 cm), robot duy trì được cân bằng |
| Thực hiện thử nghiệm và đánh giá hiệu năng hệ thống | Đã tiến hành thử nghiệm khởi động, cân bằng tĩnh, cân bằng động, di chuyển, đạt kết quả ổn định |

**Bảng 5.1. Bảng kết quả đạt được**

**5.2 Hạn chế**

Mặc dù robot đã hoạt động ổn định trong các thử nghiệm cơ bản, đề tài vẫn còn một số hạn chế:

* Hệ thống **chỉ hoạt động ổn định trong góc nghiêng nhỏ (≤ 15°)**, nếu vượt quá ngưỡng này robot sẽ ngã.
* **Thuật toán PID chưa tối ưu hoàn toàn** cho các tình huống di chuyển nhanh, dẫn đến dao động mạnh khi tăng tốc.
* **Pin hạn chế**, robot chỉ hoạt động liên tục trong khoảng 10–15 phút, sau đó động cơ giảm hiệu suất.
* **Khung robot còn đơn giản**, chưa chắc chắn, ảnh hưởng đến khả năng cân bằng trong điều kiện môi trường phức tạp.
* Chưa có tích hợp giao diện trực quan (app di động / dashboard IoT) để điều khiển và giám sát từ xa.

**5.3 Hướng phát triển**

Để nâng cao chất lượng và tính ứng dụng thực tế, nhóm dự kiến mở rộng nghiên cứu theo các hướng sau:

1. **Nâng cấp thuật toán điều khiển:**
   * Kết hợp **PID + Fuzzy logic** hoặc **LQR (Linear Quadratic Regulator)** để tăng khả năng cân bằng và ổn định khi chịu tải hoặc di chuyển nhanh.
   * Ứng dụng **bộ lọc Kalman** để tăng độ chính xác dữ liệu cảm biến MPU6050.
2. **Cải tiến phần cứng:**
   * Sử dụng động cơ DC công suất cao hơn, kèm encoder để tăng độ chính xác khi điều khiển tốc độ.
   * Nâng cấp khung robot bằng nhôm/inox/in 3D để chắc chắn và chịu tải tốt hơn.
   * Trang bị pin dung lượng cao, kèm mạch quản lý pin (BMS).
3. **Mở rộng chức năng:**
   * Tích hợp **Bluetooth/WiFi** để điều khiển và giám sát robot qua smartphone hoặc web app.
   * Phát triển **giao diện trực quan** cho phép tinh chỉnh tham số PID, hiển thị dữ liệu cảm biến theo thời gian thực.
   * Mở rộng robot thành **nền tảng nghiên cứu** cho các ứng dụng như robot vận chuyển nhỏ, robot tự hành.
4. **Ứng dụng trong thực tiễn:**
   * Robot có thể được phát triển thành **xe tự cân bằng mini** (giống Segway).
   * Kết hợp thêm camera/AI để robot di chuyển và cân bằng thông minh hơn.

# **Tài liệu tham khảo**

1. Richard C. Dorf, Robert H. Bishop (2017), Modern Control Systems, 13th Edition, Pearson.
2. Ogata, K. (2010), Modern Control Engineering, 5th Edition, Prentice Hall.
3. FreeRTOS (2024), FreeRTOS Reference Manual – API Functions and Technical Documentation, truy cập từ: https://www.freertos.org