TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐIỆN - ĐIỆN TỬ

ĐỒ ÁN MÔN HỌC 2

MÔN HỌC: HỆ THỐNG ĐIỀU KHIỂN TỰ ĐỘNG

ĐỀ TÀI: ROBOT XE HAI BÁNH TỰ CÂN BẰNG

GVHD: PGS.TS Ngô Văn Thuyên

SVTH: Phùng Minh Đức 20151462

TP.Hồ Chí Minh, Tháng 06 năm 2023

LỜI CẢM ƠN

Em xin gửi lời cảm ơn đến thầy PGS.TS Ngô Văn Thuyên đã đồng hành cùng em trong suốt quá trình học tập và thực hiện đề tài. Trong suốt quá trình thực hiện đề tài của em luôn nhận được sự quan tâm, hướng dẫn và giúp đỡ tận tình từ phía thầy để chúng em hoàn thành được đồ án môn học này.

Do trình độ nghiên cứu còn hạn chế và trong quá trình làm còn nhiều thiếu sót rất mong thầy bỏ qua. Bên cạnh đó em rất mong nhận được những nhận xét, góp ý từ phía thầy để nhóm chúng em hoàn thiện hơn bài báo cáo.

Cuối cùng nhóm em xin chúc thầy có nhiều sức khỏe và thành công trong công việc!

Em xin chân thành cảm ơn!

TP Hồ Chí Minh, tháng 06 năm 2023

Sinh viên thực hiện

Phùng Minh Đức

DANH MỤC HÌNH ẢNH

DANH SÁCH BẢNG

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

- Với sự phát triển mạnh mẽ của khoa học kĩ thuật, quá trình công nghiệp hóa hiện đại hóa ngày càng mạnh mẽ và cấp thiết. Sự phát hiện và phát triển của các robot đã hỗ trợ một phần không nhỏ vào quá trình ấy. Ngày nay các robot đã và đang thay đổi để phát huy tốt hơn những vai trò của mình trong nhiều lĩnh lực khác nhau. Nắm bắt được nhu cầu đó của xã hội việc nghiên cứu và phát triển những cơ cấu, ứng dụng của robot luôn được thúc đẩy và đầu tư rất nhiều.
- Đề tài: "Rotbot xe hai bánh tự cân bằng", cũng là đề tài được nhiều nhà nghiên cứu quan tâm vì mô hình này có khả năng di chuyển linh hoạt nhưng lại chiếm ít không gian. Phát triển đề tài cũng là phát triển cơ cấu di chuyển mới giúp robot trở nên nhỏ gọn hơn ứng dụng được nhiều hơn, nâng cao khả năng thay thế con người trong đa dạng các môi trường làm việc.

1.2. Mục tiêu đề tài

- Mục tiêu của đề tài là xây dựng một mô hình đơn giản robot xe hai bánh thực tế có khả năng tự cân bằng, xây dựng được mô hình toán học và thiết kế bộ điều khiển cho mô hình
- Áp dụng nguyên tắc con lắc ngược trong việc thiết kế và điều khiển mô hình xe hai bánh tự cân bằng. Nguyên tắc con lắc ngược là một ví dụ điển hình của một hệ thống không ổn định, bất kỳ một chút lệch lạc nào cũng có thể khiến cân bằng bị phá vỡ. Để điều khiển động cơ bánh xe cho robot tự cân bằng qua mạch cầu L298N, chúng ta cần một số thông tin về trạng thái của robot như: điểm thăng bằng cần cài đặt cho robot, hướng mà robot đang nghiêng, góc nghiêng và tốc độ nghiêng. Tất cả các thông tin này được thu thập từ MPU6050 và đưa vào bộ điều khiển PID để tạo ra một tín hiệu điều khiển động cơ, giữ cho robot ở điểm thăng bằng.
- PID là viết tắt của Proportional, Integral và Derivative. Mỗi thuật ngữ này cung cấp một hành động cho robot tự cân bằng:
    - Proportional tạo ra một phản ứng là tỷ lệ thuận với lỗi. Đối với hệ thống của chúng ta, lỗi là góc nghiêng của robot.

- Integral tạo ra một phản ứng dựa trên các lỗi tích lũy. Về cơ bản, đây là tổng của tất cả các lỗi nhân với khoảng thời gian lấy mẫu. Đây là một phản ứng dựa trên hành vi của hệ thống trong quá khứ.
- Derivative tỷ lệ với đạo hàm của lỗi. Đây là sự khác biệt giữa lỗi hiện tại và lỗi trước đó chia cho khoảng thời gian lấy mẫu. Điều này đóng vai trò như một thuật ngữ dự đoán giúp cách robot có thể hoạt động trong vòng lặp lấy mẫu tiếp theo.

- Những gì chúng ta đang cố gắng làm ở đây là giữ cho trọng tâm của robot vuông góc với mặt đất.

1.3. Phương pháp nghiên cứu

- Phương pháp phân tích và tổng hợp lý thuyết
- Phương pháp mô hình hóa
- Phương pháp quan sát khoa học
- Phương pháp phân tích và tổng kết kinh nghiệm
- Phương pháp toán học

1.4. Tình hình nghiên cứu

- *Robot xe hai bánh tự cân bằng nBot:*

Hình 1: Mô hình nBot

Mô hình "nBot" là một trong những mô hình robot xe hai bánh cân bằng tiêu biểu đầu tiên, ra mắt lần đầu tiên vào ngày 19 tháng 10 năm 2003 và được giới thiệu là " *NASA's Cool Robot of the Week"* của tác giả *David P. Anderson.* Ý tưởng cơ bản cho một robot cân bằng động hai bánh là khá đơn giản: lái các bánh xe theo hướng mà phần trên của robot đang rơi. Nếu các

bánh xe có thể được điều khiển theo cách để: Ở dưới trọng tâm của robot, robot vẫn giữ thăng bằng. Trong thực tế Điều này đòi hỏi hai cảm biến phản hồi: cảm biến độ nghiêng hoặc góc để đo độ nghiêng của robot đối với trọng lực và bộ mã hóa bánh xe để đo vị trí của cơ sở của robot.

- *Xe Điện Cân Bằng Robot Thông Minh Segway Loomo*

Hình 2: Xe điện Segway Loomo

+ Xe Cân Bằng Robot Thông Minh Segway Loomo gây sốt trên thị trường với sự kết hợp giữa xe điện Niniebot và robot thông minh có khả năng tự di chuyển. Sử dụng công nghệ RealSense của Intel, giúp nó có thể phân tích các môi trường xung quanh, nhận diện khuôn mặt, theo dõi cử chỉ và nhận diện giọng nói của chủ nhân.
+ Hệ thống điều khiển chuyển động với khả năng loại bỏ nhiễu và các kỹ thuật phản ứng nhanh cho phép Loomo chạy mượt mà trên nhiều loại bề mặt khác nhau bao gồm: bùn, cỏ, va đập, dốc và các bề mặt không bằng phẳng khác.
+ Lập kế hoạch thông minh & tránh chướng ngại vật
+ Bằng cách xây dựng các bản đồ độ phân giải cao về môi trường vật lý của nó. Tạo ra các mô hình dự đoán dự đoán hành vi của người qua

đường và tiến hành phân tích cơ học liên tục. LOOMO chủ động lên kế hoạch di chuyển và phản ứng tức thì với môi trường xung quanh. Do đó, LOOMO có thể di chuyển một cách đáng tin cậy đến một điểm đến được cung cấp. Theo dõi và theo dõi mục tiêu một cách trơn tru và tránh các chướng ngại vật trên đường đi.

- *"DJ Robot" của Toyota*

Hình 3: DJ Robot

+ "DJ Robot" của Toyota, một android hai bánh thuộc về một nhóm nhạc sĩ robot đã giải trí cho khách tham quan tại Triển lãm Thế giới Aichi 2005, Robot DJ cao 1 mét, lăn trên một cặp bánh xe giống như Segway, đã làm việc để cải thiện khả năng tương tác và giao tiếp với con người. Là một nhân viên lễ tân, máy sẽ sử dụng các kỹ năng này để cung cấp thông tin, trả lời các câu hỏi và hiển thị cho khách tham quan xung quanh các văn phòng và triển lãm.

1.5. Nội dung và kế hoạch thực hiện

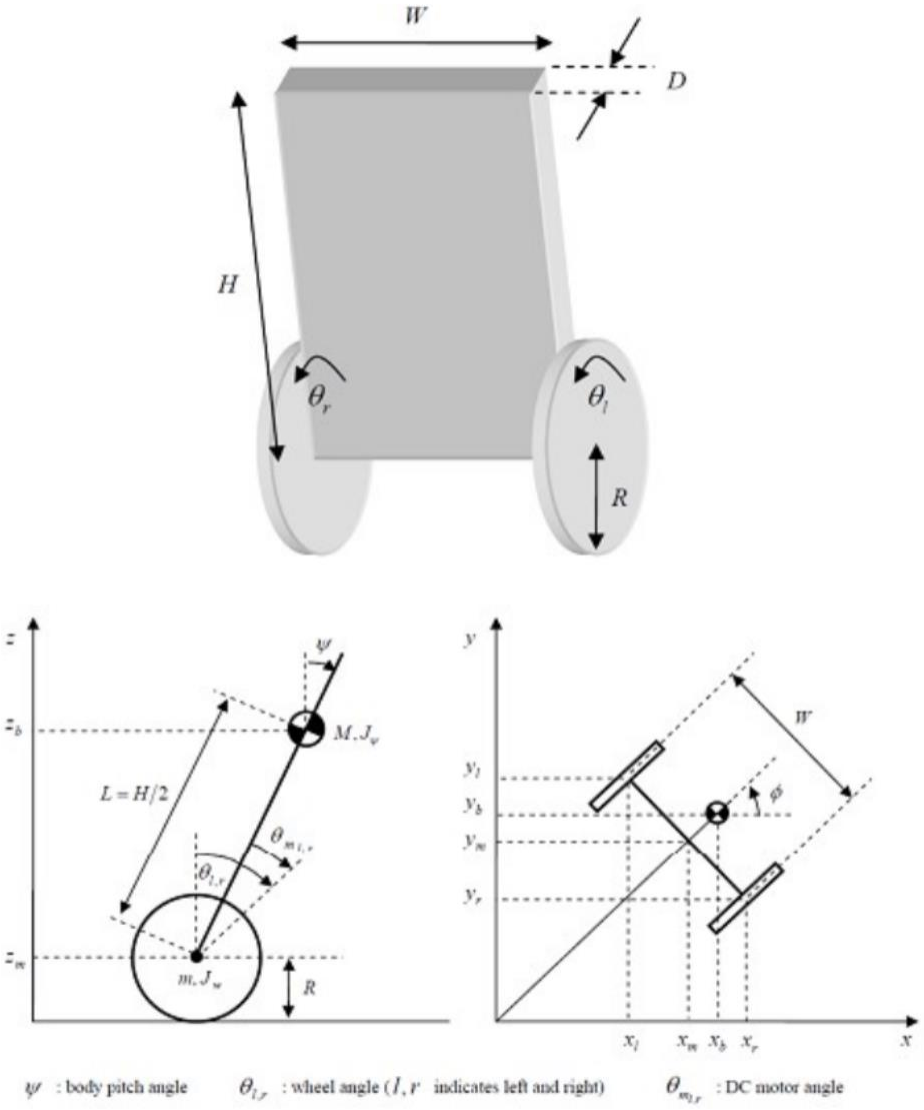
Bảng 1: Bảng kế hoạch

STT	Nội dung công việc	Thời gian	Người thực hiện	Sản phẩm dự kiến
1	Thực hiện đề xuất thiết kế của đề tài	1 tuần	Phùng Minh Đức	Các bước thực hiện đề tài
2	Nghiên cứu lý thuyết	1 tuần	Phùng Minh Đức	Các hàm truyền, PTTT, thông số bộ điều khiển của hệ thống
3	Mô phỏng hệ thống	1 tuần	Phùng Minh Đức	Xét sự ổn định của hệ thống bằng Matlab
4	Thiết kế phần cứng	2 tuần	Phùng Minh Đức	Các linh kiện cần thiết và bản vẽ hệ thống
5	Lắp ráp thiết bị	1 tuần	Phùng Minh Đức	Mô hình xe hai bánh
6	Viết chương trình lấy dữ liệu từ cảm biến và thiết kế bộ điều khiển PID	1 tuần	Phùng Minh Đức	Thuật toán điều khiển hệ thống
7	Chạy thiết bị trên thực tế và sửa các lỗi phát sinh	2 tuần	Phùng Minh Đức	Mô hình xe hai bánh tự cân bằng hoàn chỉnh
8	Kiểm tra và nhận xét các kết quả từ lý thuyết và thực tế	2 tuần	Phùng Minh Đức	Báo cáo đề tài và hoàn thiện sản phẩm

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1. Đặc tính động lực học

- Xây dựng hệ phương trình trạng thái mô tả hệ thống robot hai bánh tự cân bằng



Hình 4: Mô hình robot hai bánh tự cân bằng trên mặt phẳng

Bảng 2: Ký hiệu đơn vị và ý nghĩa các đại lượng

Ký hiệu	Đơn vị	Ý nghĩa
M	Kg	Khối lượng robot
m	Kg	Khối lượng bánh xe
R	m	Bán kính bánh xe
W	m	Chiều rộng robot
D	m	Chiều ngang robot
H	m	Chiều cao robot
L	m	Khoảng cách từ trọng tâm robot đến trục bánh xe
f_w		Hệ số ma sát giữa bánh xe và mặt phẳng di chuyển
f_m		Hệ số ma sát giữa robot và động cơ DC
J_m	$Kg.m^2$	Moment quán tính của động cơ DC
R_m	Ohm	Điện trở động cơ DC
Kb	V sec/rad	Hệ số EMF của động cơ DC
Kt	Nm/A	Moment xoắn của động cơ DC
N		Tỉ số giảm tốc
G	m/s^2	Gia tốc trọng trường
θ	Rad	Góc trung bình của bánh trái và phải
$\theta_{l,r}$	Rad	Góc của bánh trái và phải
ψ	Rad	Góc nghiêng của phần thân robot
ϕ	Rad	Góc xoay của robot
x_l, y_l, z_l	m	Tọa độ bánh trái
x_r, y_r, z_r	m	Tọa độ bánh phải
x_m, y_m, z_m	m	Tọa độ trung bình
F_θ, F_ψ, F_ϕ	Nm	Moment phát động theo các phương khác nhau
$F_{l,r}$	Nm	Moment phát động của động cơ bánh trái, phải
i_l, i_r	A	Dòng điện động cơ bánh trái, phải
v_l, v_r	V	Điện áp động cơ bánh trái, phải

- Sử dụng phương pháp Euler – Lagrange để xây dựng mô hình động học. Giả sử tại thời điểm t = 0, robot di chuyển theo chiều dương trục x, ta có các phương trình sau:
  - Góc tịnh tiến trung bình của hai bánh xe và góc xoay của robot được xác định như sau:

$$\begin{bmatrix}\theta\\ \phi\end{bmatrix}=\begin{bmatrix}\frac{1}{2}(\theta_l+\theta_r)\\ \frac{R}{W}(\theta_l-\theta_r)\end{bmatrix} \tag{2.1}$$

- Trong đó tọa độ trung bình của robot trong hệ qui chiếu:

$$\begin{bmatrix}x_m\\ y_m\\ z_m\end{bmatrix}=\begin{bmatrix}\int x'_m\\ \int y'_m\\ R\end{bmatrix} \tag{2.2}$$

Và $$\begin{bmatrix}x'_m\\ y'_m\end{bmatrix}=\begin{bmatrix}R\theta'\cos\phi\\ R\theta'\sin\phi\end{bmatrix} \tag{2.3}$$

- Tọa độ bánh trái trong hệ qui chiếu:

$$\begin{bmatrix}x_l\\ y_l\\ z_l\end{bmatrix}=\begin{bmatrix}x_m-\frac{W}{2}\sin\phi\\ y_m+\frac{W}{2}\cos\phi\\ z_m\\ \end{bmatrix} \tag{2.4}$$

- Tọa độ bánh phải trong hệ qui chiếu:

$$\begin{bmatrix}x_r\\ y_r\\ z_r\end{bmatrix}=\begin{bmatrix}x_m+\frac{W}{2}\sin\phi\\ y_m-\frac{W}{2}\cos\phi\\ z_m\\ \end{bmatrix} \tag{2.5}$$

- Tọa độ tâm đối xứng giữa hai động cơ trong hệ qui chiếu:

$$\begin{bmatrix}x_b\\ y_b\\ z_b\end{bmatrix}=\begin{bmatrix}x_m+L\sin\psi\cos\phi\\ y_mL\sin\psi\sin\phi\\ z_m+L\cos\psi\end{bmatrix} \tag{2.6}$$

- Phương trình động năng của chuyển động tịnh tiến:

$$T_1 = \frac{1}{2}m\left(x_l'^2 + y_l'^2 + z_l'^2\right) + \frac{1}{2}m\left(x_r'^2 + y_r'^2 + z_r'^2\right) + \frac{1}{2}m\left(x_b'^2 + y_b'^2 + z_b'^2\right) \quad (2.7)$$

- Phương trình động năng của chuyển động quay:

$$T_2 = \frac{1}{2}J_w\theta_l'^2 + \frac{1}{2}J_w\theta_r'^2 + \frac{1}{2}J_\psi\psi'^2 + \frac{1}{2}J_\phi\phi^2 + \frac{1}{2}n^2J_m\left(\theta_l' - \psi'\right)^2 + \frac{1}{2}n^2J_m\left(\theta_r' - \psi'\right)^2 \quad (2.8)$$

Với $\frac{1}{2}n^2J_m\left(\theta_l' - \psi'\right)^2 + \frac{1}{2}n^2J_m\left(\theta_r' - \psi'\right)^2$ là động quay của phần ứng động cơ trái và phải

- Phương trình thế năng:

$$U = mgz_l + mgz_r + mgz_b \quad (2.9)$$

- Phương trình Lagrange:

$$L = T_1 + T_2 - U \quad (2.10)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \theta'}\right) - \frac{\partial L}{\partial \theta} = F_\theta \quad (2.11)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \psi'}\right) - \frac{\partial L}{\partial \psi} = F_\psi \quad (2.12)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \phi'}\right) - \frac{\partial L}{\partial \phi} = F_\phi \quad (2.13)$$

- Lấy đạo hàm L theo các biến ta được:

$$\left[\left(2m + M\right)R^2 + 2J_w + 2n^2J_m\right]\theta' + \left(MLR\cos\psi - 2n^2J_m\right)\psi' - MLR\psi'^2\sin\psi = F_\theta \quad (2.14)$$

$$\left(MLR\cos\psi - 2n^2J_m\right)\theta' + \left(ML^2 + J_\psi + 2n^2J_m\right)\psi' - MgL\sin\psi - ML^2\phi'^2\sin\psi\cos\psi = F_\psi \quad (2.15)$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2J_m\right) + ML^2\sin^2\psi\right]\phi'^2 + 2ML^2\psi'\phi'\sin\psi\cos\psi = F_\phi \quad (2.16)$$

- Moment động lực do động cơ DC sinh ra:

$$\begin{bmatrix} F_\theta \\ F_\psi \\ F_\phi \end{bmatrix} = \begin{bmatrix} F_l + F_r \\ F_\psi \\ \dfrac{W}{2R(F_l - F_r)} \end{bmatrix} \tag{2.17}$$

Và:

$$F_l = nK_t i_l + f_m(\psi' - \theta_l') - f_w\theta_l' \tag{2.18}$$

$$F_r = nK_t i_r + f_m(\psi' - \theta_r') - f_w\theta_r' \tag{2.19}$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m(\psi' - \theta_l') - f_m(\psi' - \theta_r') \tag{2.20}$$

- Sử dụng phương pháp PWM để điều khiển động cơ nên chuyển từ dòng điện sang điện áp động cơ:

$$L_m i_{l,r}'' = v_{l,r} + K_b(\psi' - \theta_{l,r}') - R_m i_{l,r} \tag{2.21}$$

- Xem điện cảm bằng phần ứng tương đối nhỏ (gần bằng 0), có thể bỏ qua, suy ra:

$$i_{l,r} = \frac{v_{r,l} + K_b(\psi' - \theta_{l,r}')}{R_m} \tag{2.22}$$

- Từ đó các moment lực sinh ra:

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + f_w)\theta' + 2\beta\psi' \tag{2.23}$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\theta' - 2\beta\psi' \tag{2.24}$$

Với $\alpha = \dfrac{nK_t}{R_m}$ và $\beta = \dfrac{nK_tK_b}{R_m} + f_m$

$$F_\phi = \frac{W}{2R}\alpha(v_l + v_r) - \frac{W^2}{2R^2}(\beta + f_w)\phi' \tag{2.25}$$

- Thu được phương trình động học mô tả chuyển động của robot như sau:

$$\left[(2m+M)R^2 + 2J_w + 2n^2J_m\right]\theta' + \left(MLR\cos\psi - 2n^2J_m\right)\psi' - MLR\psi'^2\sin\psi = \alpha(v_l + v_r) - 2(\beta + f_w)\theta' + 2\beta\psi' \tag{2.26}$$

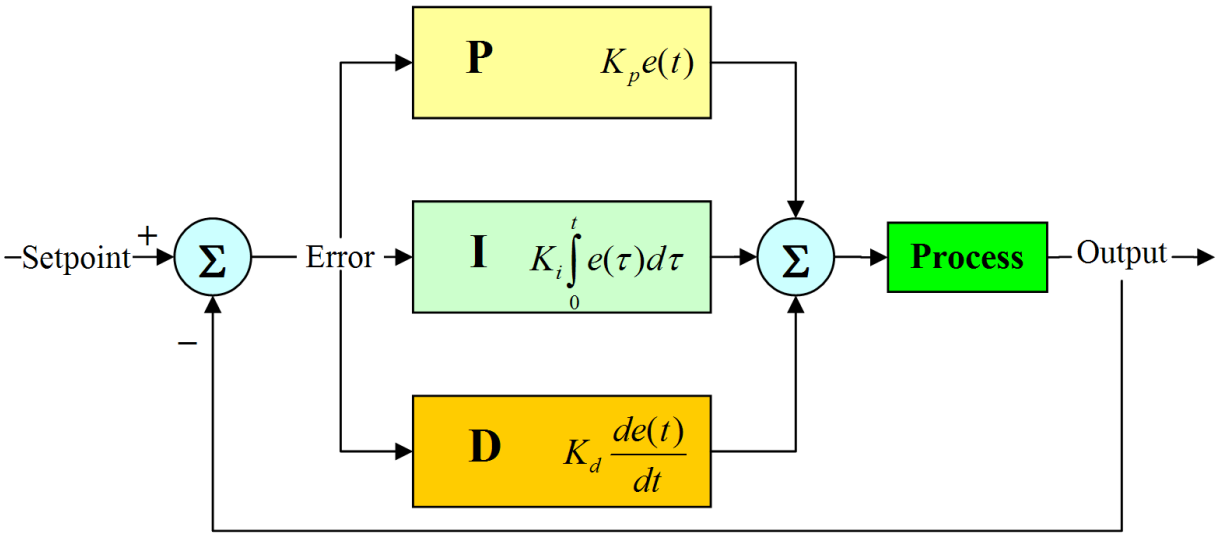
$$\left(MLR\cos\psi - 2n^2J_m\right)\theta' + \left(ML^2 + J_\psi + 2n^2J_m\right)\psi' - MgL\sin\psi - ML^2\phi'^2\sin\psi\cos\psi = -\alpha(v_l + v_r) + 2\beta\theta' - 2\beta\psi' \tag{2.27}$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2 J_m\right) + ML^2\sin^2\psi\right]\phi'^2 + 2ML^2\psi'\phi'\sin\psi\cos\psi = \frac{W}{2R}\alpha\left(v_l + v_r\right) - \frac{W^2}{2R^2}\left(\beta + f_w\right)\phi' \quad (2.28)$$

2.2. Giải thuật điều khiển

- Giới thiệu bộ điều khiển PID
  + PID đầy đủ là Proportional Integral Derivative. Đây là một cơ chế phản hồi vòng điều khiển trong các hệ thống điều khiển công nghiệp được sử dụng rộng rãi. Bộ điều khiển này được đưa vào sử dụng nhiều nhất trong các hệ thống điều khiển vòng kín hay các hệ thống có tín hiệu phản hồi.

  + Bộ điều khiển PID giúp các máy công nghệ tính toán ra các giá trị sai số là hiệu số giữa các giá trị đo thông số về biển đối và giá trị đặt theo mong muốn của người thiết kế, người dùng. Thông qua các điều chỉnh giá trị điều khiển đầu mà mà nhờ đó hệ thống điều khiển khi thực hiện sẽ giảm được tối đa những sai số, cho hoạt động chính xác hơn và đạt hiệu quả cao hơn
  + PID là viết tắt của Proportional, Integral và Derivative. Mỗi thuật ngữ này cung cấp một hành động cho robot tự cân bằng.
    - **Proportional** tạo ra một phản ứng là tỷ lệ thuận với lỗi. Đối với hệ thống của chúng ta, lỗi là góc nghiêng của robot.
    - **Integral** tạo ra một phản ứng dựa trên các lỗi tích lũy. Về cơ bản, đây là tổng của tất cả các lỗi nhân với khoảng thời gian lấy mẫu. Đây là một phản ứng dựa trên hành vi của hệ thống trong quá khứ.
    - **Derivative** tỷ lệ với đạo hàm của lỗi. Đây là sự khác biệt giữa lỗi hiện tại và lỗi trước đó chia cho khoảng thời gian lấy mẫu. Điều này đóng vai trò như một thuật ngữ dự đoán giúp cách robot có thể hoạt động trong vòng lặp lấy mẫu tiếp theo.
  + Nhân mỗi thuật ngữ với các hằng số tương ứng của chúng (ví dụ, Kp, Ki và Kd) và lấy tổng kết quả, chúng ta tạo ra đầu ra được gửi như lệnh để điều khiển động cơ.



Hình 5: Sơ đồ khối bộ điều khiển PID

- Tín hiệu điều khiển của bài:

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

Với u(t) là điện áp cấp cho động cơ, e(t) là sai số của tín hiệu đặt là ngõ ra

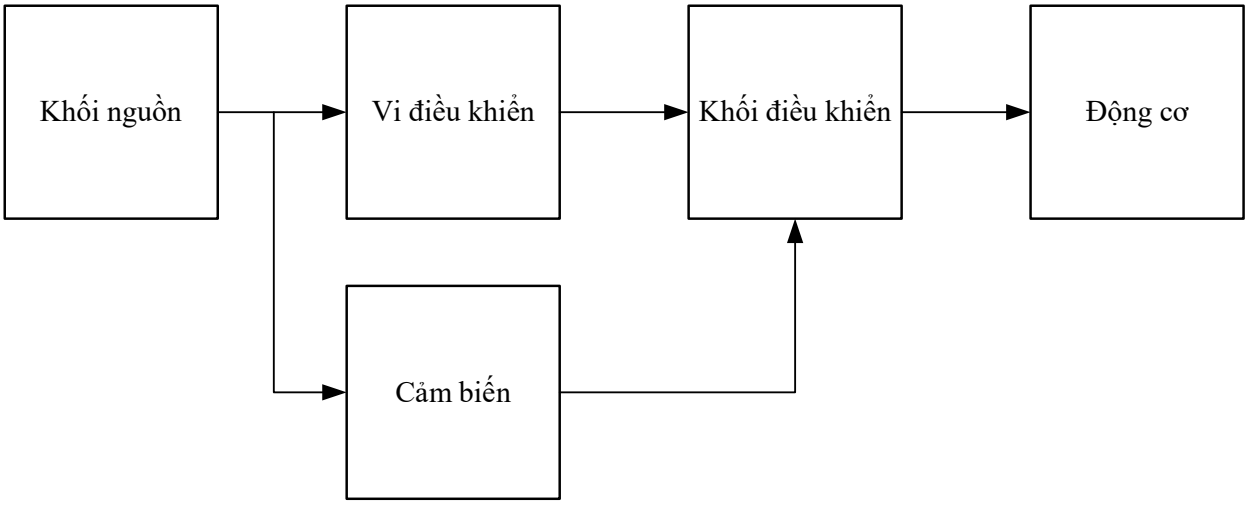
- Điều chỉnh hằng số PID
  + Đặt Ki và Kd về 0 và tăng dần Kp sao cho robot bắt đầu dao động về vị trí 0.
  + Tăng Ki để phản xạ của robot nhanh hơn khi nó mất cân bằng. Ki phải đủ lớn sao cho góc nghiêng không tăng. Robot sẽ quay trở lại vị trí 0 nếu nó nghiêng.
  + Tăng Kd để giảm dao động. Các overshoots cũng nên được giảm.
  + Lặp lại các bước trên bằng cách tinh chỉnh từng thông số để đạt được kết quả tốt nhất.

CHƯƠNG 3: THIẾT KẾ PHẦN CỨNG

3.1. Yêu cầu thiết kế

- Yêu cầu của hệ thống robot xe hai bánh cân bằng trước hết là khi hệ thống ở vị trí xác lập thì xe cần được cân bằng. Bằng việc sử dụng mô-men xoắn từ động cơ và các quy tắc quán tính thì hệ thống cần trở lại vị trí xác lập cân bằng khi có tác động ngoại lực cụ thể là tác động bên ngoài khiến xe bị nghiêng. Ngoài ra để phù hợp với hoạt động xe thì yêu cầu ngoại hình xe cần nhỏ gọn, tối ưu về trọng lượng và trọng tâm của xe.

3.2. Sơ đồ khối của hệ thống



Hình 6: Sơ đồ khối hệ thống

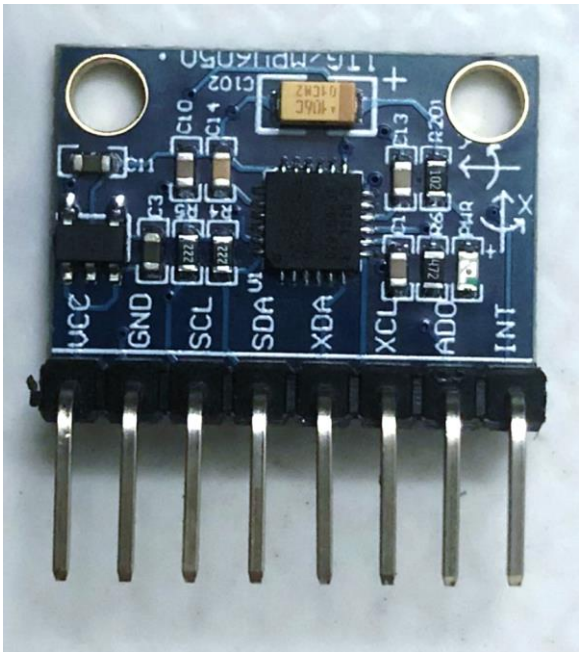
- Hệ thống gồm:
  + Khối nguồn: gồm 3 pin cell 18650 3.7V/4.2V 2000mAh nối tiếp.
  + Khối vi điều khiển: sử dụng Arduino Nano V3.0 ATmega328P
  + Khối cảm biến: Cảm Biến gia tốc GY-521 6DOF IMU MPU6050
  + Khối điều khiển: sử dụng IC Driver A4988 để điều khiển và giao tiếp với vi điều khiển thông qua Module CNC Shield V4.
  + Khối động cơ: sử dụng Động cơ bước size 42: KV 4234-F2B009

3.3. Lựa chọn thiết bị

3.3.1. Lựa chọn cảm biến

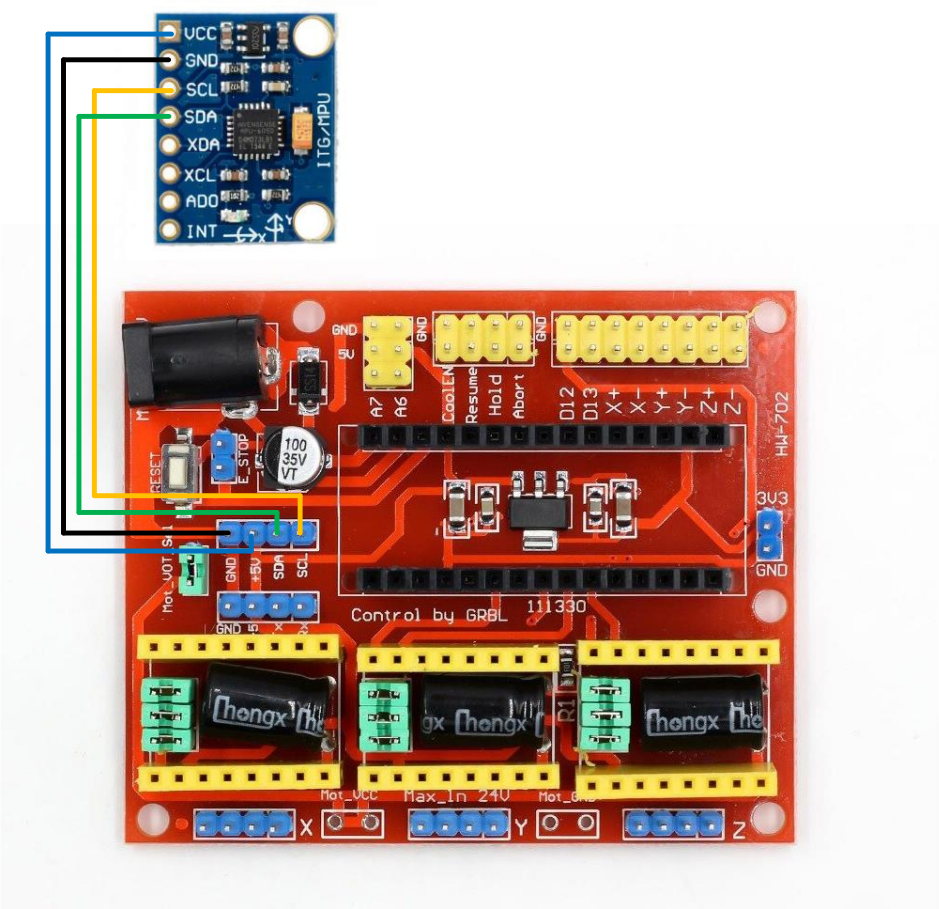
- Từ yêu cầu giữ Robot xe hai bánh ở vị trí cân bằng chúng ta cần lựa chọn một cảm biến có thể đọc và trả về tín hiệu góc lệch, vậy nên ở mô hình này

sử dụng 1 cảm cảm biến nhận biết góc nghiêng Cảm Biến gia tốc GY-521 6DOF IMU MPU6050



Hình 7: Cảm biến gia tốc MPU6050

- Thông số kĩ thuật:
  - Điện áp sử dụng: 3~5VDC
  - Điện áp giao tiếp: 3~5VDC
  - Chuẩn giao tiếp: I2C
  - Giá trị Gyroscopes trong khoảng: +/- 250 500 1000 2000 degree/sec
  - Giá trị Acceleration trong khoảng: +/- 2g, +/- 4g, +/- 8g, +/- 16g
- Ở hệ thống Robot xe hai bánh cân bằng chỉ sử dụng Cảm Biến gia tốc GY-521 6DOF IMU MPU6050 để đo góc lệch của xe.



Hình 8: Sơ đồ nối dây cảm biến

3.3.2. Lựa chọn cơ cấu chấp hành

- Để phù hợp với yêu cầu đề ra mô hình cần 2 động cơ giúp xe di chuyển qua lại với tốc độ phụ thuộc vào góc lệch của xe. Cơ cấu chấp hành của hệ thống bao gồm 2 động cơ được kết nối với IC Driver A4988 để điều khiển và giao tiếp với vi điều khiển thông qua Module CNC Shield V4.
- Động cơ được sử dụng trong mô hình là Động cơ bước size 42: KV 4234-F2B009, là loại động cơ có giá thành thấp có thể điều chỉnh chính xác được góc quay, cung cấp được mô-men xoắn lớn ở dải vận tốc trung bình và thấp, dễ dàng lắp đặt và điều khiển. Phù hợp với yêu cầu của mô hình

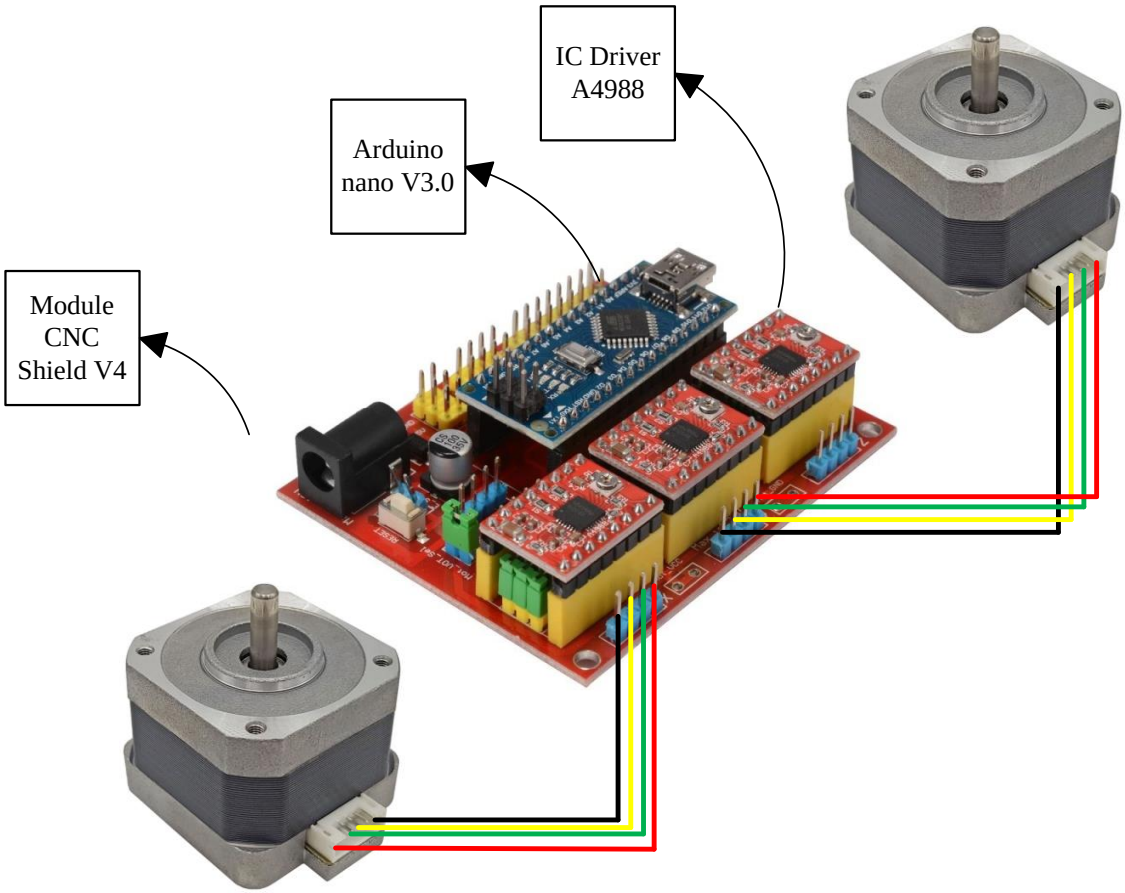
Hình 9: Động cơ Step

- Thông số cơ bản của động cơ

Bảng 3: Thông số động cơ

Bước(°)	1.8
Điện áp(V)	4.0
Dòng(A)	0.6
Điện trở của cuộn dây(Ω/pha)	5.0
Cảm kháng(mH/pha)	7.5
Momen xoắn	260

- Kết nối :



Hình 10: Sơ đồ nối dây động cơ

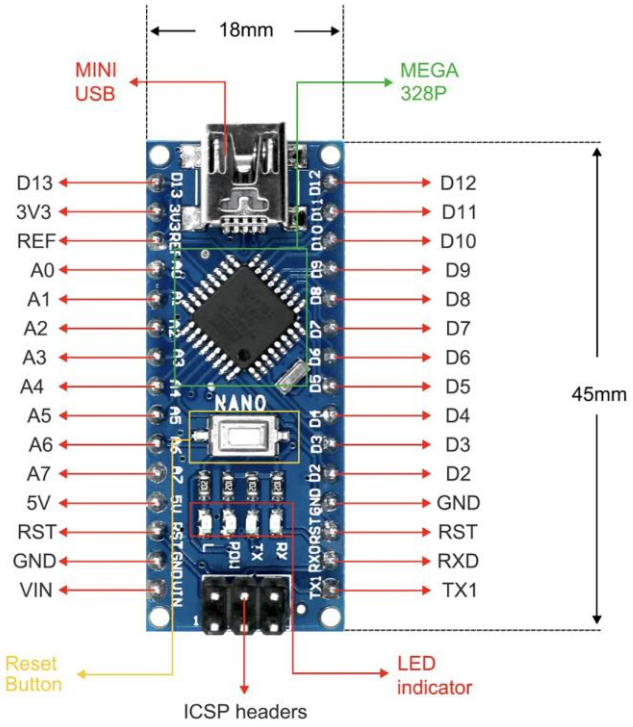
3.3.3. Lựa chọn vi điều khiển

- Để phù hợp với yêu cầu điều khiển và mô hình nhỏ gọn, chọn vi điều khiển Arduino Nano V3.0 ATmega328P. Bởi vì Arduino Nano V3.0 ATmega328P là bản thu nhỏ của các dòng như Arduino Uno, nhưng nó có thiết kế nhỏ gọn, linh hoạt, tiện sử dụng cho các Breadboard nhỏ, cũng như cho các Project đòi hỏi tính nhỏ gọn. Một tính năng đáng chú ý nữa trên Arduino Nano là sử dụng chip CH340 để giao tiếp, việc này giúp tiết kiệm và hạ giá thành sản phẩm.

- Thông số kĩ thuật:

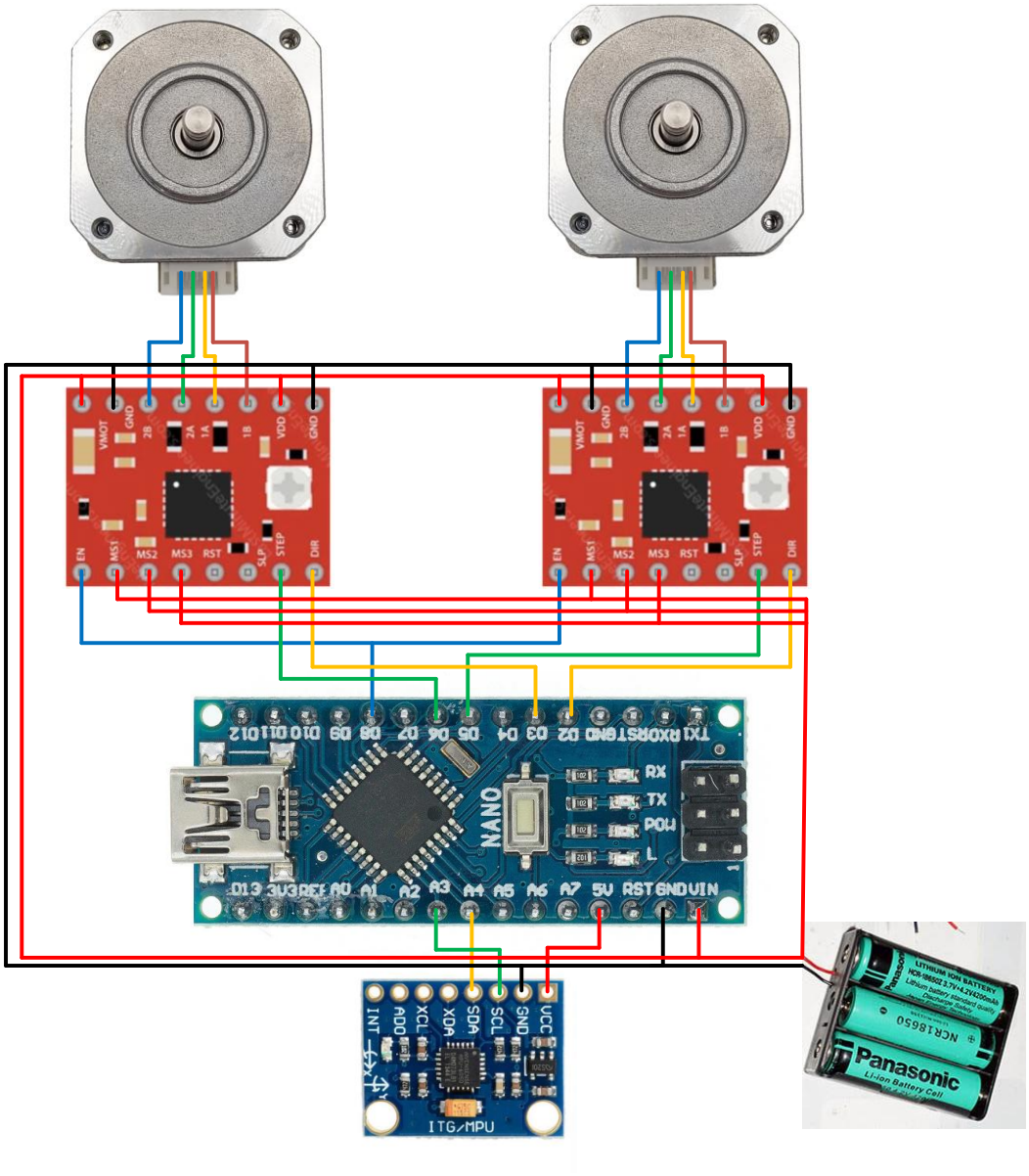
Bảng 4: Thông số Arduino Nano V3.0

Vi điều khiển	Arduino Nano V3.0 ATmega328P
Điện áp hoạt động	5V
Điện áp đầu vào(giới hạn)	6-20V
Điện áp đầu vào (khuyên dùng)	7-12V
Chân Digital I/O	14
Chân PWM Digital I/O	6
Chân đầu vào Analog	8
Dòng sử dụng I/O Pin	20mA (tối đa 40mA)
Bộ nhớ Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Chiều dài	45 mm
Chiều rộng	18 mm
Trọng lượng	5g



Hình 11: Sơ đồ chân Arduino Nano V3.0

3.4. Sơ đồ nối dây



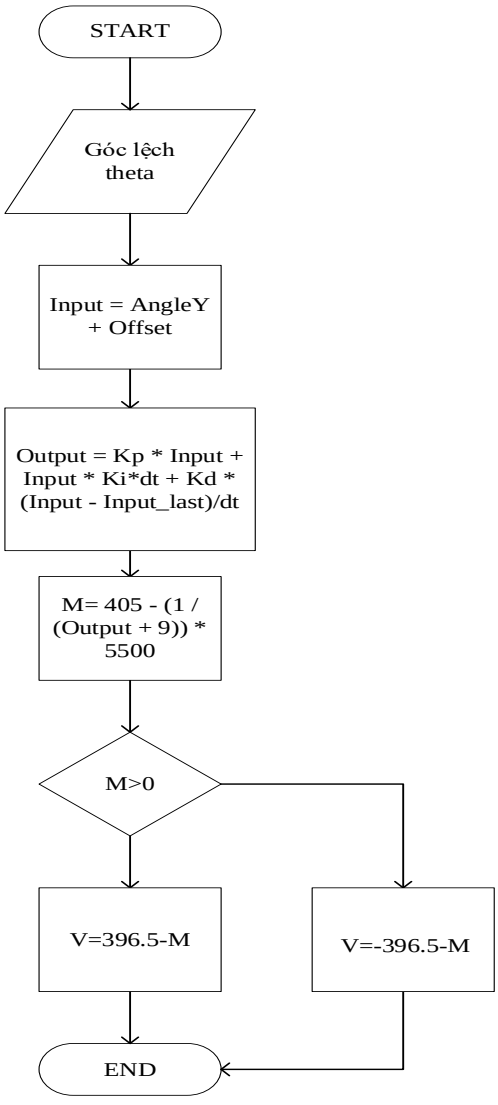
Hình 12: Sơ đồ nối dây hệ thống

CHƯƠNG 4: THIẾT KẾ PHẦN MỀM

4.1. Yêu cầu thiết kế

- Giao tiếp bằng vi điều khiển Arduino Nano V3.0 với yêu cầu trước hết là đưa xe lên được vị trí cân bằng và duy trì nó. Áp dụng những kiến thức được học về bộ điều khiển PID thay đổi những thông số K_P, K_I, K_D để thay đổi thời gian xác lập giảm độ vọt lố và sự giao động của tín hiệu điều khiển một cách tối ưu nhất cho xe hoạt động

4.2. Giải thuật, chương trình



Hình 13: Lưu đồ chương trình

- Bộ điều khiển của mô hình cơ bản là hệ thống có 1 ngõ vào và 1 ngõ ra. Hệ thống thu thập dữ liệu góc lệch theta từ cảm biến MPU6050 so với tín hiệu đặt là 0, tín hiệu ngõ vào sẽ được cộng thêm với tham số offset ở đây là giá trị trả về của cảm biến có thể khác 0 khi xe cân bằng do việc thi công phần cứng có thể làm sai lệch. Ngõ ra bộ điều khiển sẽ sử dụng công thức của bộ điều khiển PID tính toán Output để điều khiển động cơ ta thông qua hàm M là hàm truyền tuyến tính của động cơ được thành lập qua việc khảo sát vùng hoạt động của động cơ và thay đổi theo ngõ ra output của bộ điều khiển.
- Chọn chế độ hoạt động của động cơ ở chế độ 1/16 vận tốc của động cơ sẽ phụ thuộc vào V. Sử dụng timer2 trong vi điều khiển để tạo xung step với |V| là chu kì của 1 xung step, do đó ứng với giá trị |V| càng nhỏ thì vận tốc của động cơ sẽ càng lớn và chiều quay của động cơ sẽ phụ thuộc vào dấu của V.
- Dùng phần mềm Arduino IDE để viết chương trình cho vi điều khiển. Chương trình cụ thể ở Phụ lục [1], thư viện dùng cho Cảm Biến gia tốc GY-521 6DOF IMU MPU6050 ở Phụ lục [2].

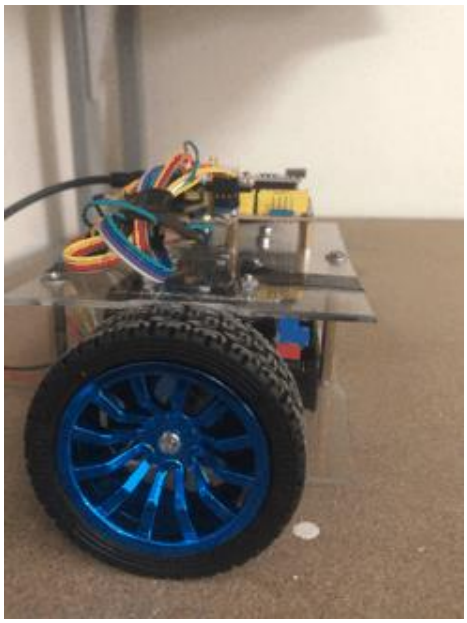
CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM

5.1. Kết quả xây dựng mô hình

- Mô hình xe hai bánh cân bằng đáp ứng được yêu cầu đề ra về phần cứng cần nhỏ gọn mang lại tính linh hoạt cao cho robot nhưng trong quá trình xây dựng lắp đặt mô hình cần có những lưu ý cơ bản sau:
  + Động cơ hoạt động ở chế độ 1/16 thế nên 3 chân MS1,MS2,MS3 của module CNC Shield V4 đã được hàn trực tiếp vào Vcc
  + Khi lắp cảm biến cần độ chính xác cao điều chỉnh sao cho vị trí cảm biến gần nhất với trọng tâm của xe
  + Mô hình xe hai bánh cân bằng cần thi công lắp ráp sao cho mang tính đối xứng cao để thuận lợi hơn cho việc điều khiển và hoạt động trong thời gian dài.

5.2. Kết quả thử nghiệm

- Qua nhiều lần thử khác nhau đã đưa ra bộ thông số PID tối ưu như sau:
  + $K_{p=}25$
  + $K_i=0.01$
  + $K_d=0.5$
  + Offset=-1.8
- Kết quả của mô hình xe hai bánh cân bằng dưới đây theo dạng gif



Hình 14: Kết quả thực nghiệm

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- Qua việc nghiên cứu thực hành và thi công đề tài "Robot xe hai bánh cân bằng" em đã đúc kết được rất nhiều kiến thức và kinh nghiệm trong việc tìm kiếm, tham khảo, và tự mình xây dựng một sản phẩm. Bài báo cáo giúp em hiểu thêm nhiều hơn về điều khiển một hệ thống phi tuyến, tiếp cận sâu hơn về bộ điều khiển PID, tiếp xúc với các cảm biến, vi điều khiển và các cơ cấu chấp hành. Mô hình xe hai bánh cân bằng đã hoạt động tương đối thành công với yêu cầu mà chính bản thân em đề ra, tuy nhiên vẫn còn có sai sót qua đó việc nâng cấp và bổ sung cho mô hình là rất cần thiết cho mô hình này.
- Hướng phát triển sắp tới của mô hình là em muốn cải tiến thêm cho robot có thể di chuyển theo sự điều khiển mà vẫn giữ được trạng thái cân bằng. Hơn thế nữa là sẽ xây dựng một mô hình có tải trọng lớn hơn có thể áp dụng trong việc vận chuyển những hàng hóa có trọng lượng nhẹ đến trung bình. Những mô hình robot quản gia hay phục vụ nhà hàng cũng là mục tiêu phát triển của mô hình "Robot xe hai bánh tự cân bằng"

TÀI LIỆU THAM KHẢO

1. “Tìm hiểu chung về bộ điều khiển PID: Khái niệm, phân loại và ứng dụng”, bkaii.com.vn .
2. “Xe hai bánh tự cân bằng di chuyển trên địa hình phẳng”, Mai Tuấn Đạt.
3. “Self balancing robot using stepper motor,dành cho người mới lập trình Arduino”, Youtube Kỹ thuật điện tử.
4. “Mô hình robot hai bánhtự cân bằng”, Nguyễn Khắc Mẫn

PHỤ LỤC

[1]. Chương trình của vi điều khiển

```
/*
  Robot xe hai bánh tự cân bằng
  Programmed by Phùng Minh Đức
  Date:20/6/2023

  Dùng Arduino nano  kết hợp CNC Shield V4
  Điều khiển động cơ dùng 2 x A4988
  2 Đông cơ bước : size 42 1.8 step
  Cảm biến góc nghiêng : MPU6050

  thoi gian 1 xung = 20*x us = 0.00002*x s

    nếu chọn vi bước là 1/16
                1 vong =3200 xung----> thoi gian 1 vong ---> 3200*0.00002*x s
                                                V---------> 60 s
                                                V=60/(32*0.002*x)

                x=5 ----> v= 187.5 vong/ phut
                x=50----> v= 18.75 vong/phut
*/
#include "stmpu6050.h"
SMPU6050 mpu6050;
// ĐỊNH NGHĨA CHÂN CNC SHIELD V4
//                chân ARDUINO   ký hiệu trên          chân PORT AVR
//                               board Arduino nano      Atmega 328P
# define Enable       8           //D8                 //PORTB 0

# define Step_2       6           //D6                 //PORTD 6

# define Step_1       5           //D5                 //PORTD 5

# define Dir_2        3           //D3                 //PORTD 3

# define Dir_1        2           //D2                 //PORTD 2

//     HÀM KHAI BÁO CÁC CHÂN ARDUINO NANO
//.................................
void  pin_INI() {
  pinMode(Enable, OUTPUT);
  pinMode(Step_1, OUTPUT);
  pinMode(Step_2, OUTPUT);
```

```
  pinMode(Dir_1, OUTPUT);
  pinMode(Dir_2, OUTPUT);
  digitalWrite(Enable, LOW);
}
//     HÀM KHAI BÁO TIMER2
//.....................................
void timer_INI() {

  TCCR2A = 0;                                                    //Gán
giá trị thanh ghi TCCR2A là 0
  TCCR2B = 0;                                                    //Gán
giá trị thanh ghi TCCR2B là 0
  TCCR2B |= (1 << CS21);                                         //Set
bit CS21 ở thanh ghi TCCR2B lên 1
  OCR2A = 39;                                                    //The
compare register is set to 39 => 20us / (1s / (16.000.000Hz / 8)) - 1
  TCCR2A |= (1 << WGM21);                                        //Set
counter 2 to CTC (clear timer on compare) mode Chế độ CTC bộ đếm được xóa về 0
khi giá trị bộ đếm (TCNT0) khớp với OCR0A
  TIMSK2 |= (1 << OCIE2A);                                       //Set
the interupt enable bit OCIE2A in the TIMSK2 register
}

int8_t Dir_M1, Dir_M2;                                           //Biến xác
định hoạt động của động cơ và chiều quay Dir_Mx >0 quay thuận , Dir_Mx <0 quay
ngược Dir_Mx =0 motor ngừng quay
volatile int Count_timer1, Count_timer2;                        //đếm các lần
TIMER xuất hiện trong chu kỳ xung STEP
volatile int32_t Step1, Step2;
int16_t Count_TOP1, Count_BOT1, Count_TOP2, Count_BOT2;  //vị trí cuối của phần
cao và cuối phần thấp trong 1 chu kỳ xung STEP
float Input_L, Input_R, Offset, Output, I_L, I_R, Input_lastL, Input_lastR,
Output_L, Output_R, M_L, M_R, Motor_L, Motor_R, Vgo;
float Kp = 25;
float Ki = 0.01;
float Kd = 0.5;
unsigned long loop_timer;

//     CHƯƠNG TRÌNH NGẮT CỦA TIMER2
//.....................................
ISR(TIMER2_COMPA_vect) {
  //tạo xung STEP cho MOTOR1
  if (Dir_M1 != 0) {
 //nếu MOTOR cho phép quay
```

```
    Count_timer1++;
    if (Count_timer1 <= Count_TOP1)PORTD |= 0b00100000;
 //nếu là nhịp nằm trong phần cao trong xung STEP
    else PORTD &= 0b11011111;
//nếu là nhịp nằm trong phần thấp của xung STEP
    if (Count_timer1 > Count_BOT1) {
      Count_timer1 = 0;                              //nếu là nhịp cuối của 1 xung
STEP
      if (Dir_M1 > 0)Step1++;
      else if (Dir_M1 < 0)Step1--;
    }
  }

  //tạo xung STEP cho MOTOR2
  if (Dir_M2 != 0) {
    Count_timer2++;
    if (Count_timer2 <= Count_TOP2)PORTD |= 0b01000000;
    else PORTD &= 0b10111111;
    if (Count_timer2 > Count_BOT2) {
      Count_timer2 = 0;
      if (Dir_M2 > 0)Step2++;
      else if (Dir_M2 < 0)Step2--;
    }
  }

}

//     HÀM TỐC ĐỘ DI CHUYỂN MOTOR_RIGHT
//.....................................
void Speed_R(int16_t x) {
  if (x < 0) {
    Dir_M1 = -1;
    PORTD &= 0b11111011;
  }
  else if (x > 0) {
    Dir_M1 = 1;
    PORTD |= 0b00000100;
  }
  else Dir_M1 = 0;

  Count_BOT1 = abs(x);
  Count_TOP1 = Count_BOT1 / 2;
}
```

```
//     HÀM TỐC ĐỘ DI CHUYỂN MOTOR_LEFT
//.................................
void Speed_L(int16_t x) {
  if (x < 0) {
    Dir_M2 = -1;
    PORTD &= 0b11110111;
  }
  else if (x > 0) {
    Dir_M2 = 1;
    PORTD |= 0b00001000;
  }
  else Dir_M2 = 0;

  Count_BOT2 = abs(x);
  Count_TOP2 = Count_BOT2 / 2;
}

////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////
void setup() {
  mpu6050.init(0x68);
  Serial.begin(9600);               //Khai báo Serial
  pin_INI();                        //Khai báo PIN Arduino đấu nối 3 DRIVER A4988
  timer_INI();                      //Khai báo TIMER2
  delay(500);
    loop_timer = micros() + 4000;
}
////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////
void loop() {
  float AngleY = mpu6050.getYAngle();
  Serial.println(AngleY);
  Offset = -1.8;
  Vgo = 0.05;
  //Dùng PID cho MOTOR_L
  Input_L = AngleY + Offset - Vgo ;                              //Vgo<0  chạy
lui,Vgo >0 chạy tới
  I_L += Input_L * Ki;
  I_L = constrain(I_L, -400, 400);
  Output_L = Kp * Input_L + I_L + Kd * (Input_L - Input_lastL);
  Input_lastL = Input_L;                                          //Lưu làm độ
lệch trước cho vòng loop sau
  //Khống chế OUTPUT theo sự phi tuyến MOTOR_L
  if (Output_L > -5 && Output_L < 5)Output_L = 0;
```

```
  Output_L = constrain(Output_L, -400, 400);


  //Dùng PID cho MOTOR_R
  Input_R = AngleY + Offset - Vgo; //Vgo<0  chạy lui,Vgo >0 chạy tới
  I_R += Input_R * Ki;
  I_R = constrain(I_R, -400, 400);
  Output_R = Kp * Input_R + I_R + Kd * (Input_R - Input_lastR);
  Input_lastR = Input_R;
  //Khống chế OUTPUT theo sự phi tuyến MOTOR_R
  if (Output_R > -5 && Output_R < 5)Output_R = 0;
  Output_R = constrain(Output_R, -400, 400);

  //Khắc phục sự phi tuyến của MOTOR_L
  if (Output_L > 0)M_L = 405 - (1 / (Output_L + 9)) * 5500;
  else if (Output_L < 0)  M_L = -405 - (1 / (Output_L - 9)) * 5500;
  else M_L = 0;

  //Khắc phục sự phi tuyến của MOTOR_R
  if (Output_R > 0)M_R = 405 - (1 / (Output_R + 9)) * 5500;
  else if (Output_R < 0)M_R = -405 - (1 / (Output_R - 9)) * 5500;
  else M_R = 0;

  //Làm ngược giá trị truyền vào hàm Speed_L()
  if (M_L > 0)Motor_L = 396.55 - M_L;
  else if (M_L < 0)Motor_L = -396.55 - M_L;
  else Motor_L = 0;

  //Làm ngược giá trị truyền vào hàm Speed_R()
  if (M_R > 0)Motor_R = 396.55 - M_R;
  else if (M_R < 0)Motor_R = -396.55 - M_R;
  else Motor_R = 0;

  //cho 2 MOTOR chạy
  Speed_L(Motor_L);
  Speed_R(Motor_R);

  while(loop_timer > micros());
  loop_timer += 4000;
}
```

[2]. Thư viện stmpu6050

```
#ifndef _SAIGONTECH_MPU6050_H_
#define _SAIGONTECH_MPU6050_H_
```

```
#include "Wire.h"
#define   RAD2DEG   57.295779

class SMPU6050 {
  public:
    SMPU6050() {
    };

    void init(int address) {
      this->i2cAddress = address;

      this->gyroXOffset = 0;
      this->gyroYOffset = 0;
      this->gyroZOffset = 0;

      this->xAngle = 0;
      this->yAngle = 0;
      this->zAngle = 0;


      this->accX = 0;
      this->accY = 0;

      this->prevMillis = millis();

// reset và cấu hình MPU6050
      Wire.begin();
      Wire.beginTransmission(this->i2cAddress);
      Wire.write(0x6B);// đi đến thanh ghi PWR_MGMT_1
      Wire.write(0);   // reset MPU6050
      Wire.endTransmission(true);

      Wire.beginTransmission(this->i2cAddress);
      //đi tới thanh ghi SMPLRT_DIV thanh ghi chia tỉ lệ lấy mẫu
      Wire.write(0x19); //Sample Rate = Gyrocope Output Rate/(1+SMPLRT_DIV)
      Wire.write(0);  //Giá trị không dấu 8 bit. Tốc độ mẫu được xác định bằng
cách chia tốc độ đầu ra của con quay hồi chuyển cho giá trị này
      Wire.endTransmission(true);

      Wire.beginTransmission(this->i2cAddress);
      Wire.write(0x1B); //thanh ghi cấu hình GYRO_CONFIG Thanh ghi này được sử
dụng để kích hoạt tự kiểm tra con quay hồi chuyển và cấu hình phạm vi quy mô đầy
đủ của con quay hồi chuyển.
```

```
      Wire.write(0); //=0 -->phạm vi quy mô đầy đủ của đầu ra con quay hồi chuyển
= +- 250 dec/s không tự kiểm tra
      Wire.endTransmission(true);

      Wire.beginTransmission(this->i2cAddress);
      Wire.write(0x1C); //Cấu hình gia tốc ACCEL_CONFIG Thanh ghi này được sử
dụng để kích hoạt tự kiểm tra gia tốc kế và định cấu hình phạm vi toàn thang đo
gia tốc. Thanh ghi này cũng cấu hình Bộ lọc thông cao kỹ thuật số (DHPF).
      Wire.write(0);// =0 -->chọn phạm vi toàn thang đo của các đầu ra gia tốc =
+- 2g .không tự kiểm tra
      Wire.endTransmission(true);
    }
//hàm hiệu chỉnh
    void calibrate(int times) {
      long gyroXTotal = 0, gyroYTotal = 0, gyroZTotal = 0;
      int count = 0;
      int gyroRawX, gyroRawY, gyroRawZ;
      for (int i = 0; i < times; i++) {
        Wire.beginTransmission(this->i2cAddress);
        Wire.write(0x43); //3thanh ghi 16 bit từ (0x43-0x48) GYRO_XOUT  GYRO_YOUT
 GYRO_ZOUT Những thanh ghi này lưu trữ các phép đo con quay gần đây nhất
        Wire.endTransmission(false);
        Wire.requestFrom(this->i2cAddress, 6, true);

        gyroRawX = Wire.read() << 8 | Wire.read();
        gyroRawY = Wire.read() << 8 | Wire.read();
        gyroRawZ = Wire.read() << 8 | Wire.read();

        gyroXTotal += gyroRawX;
        gyroYTotal += gyroRawY;
        gyroZTotal += gyroRawZ;
        count += 1;
      }
      gyroXOffset = -gyroXTotal * 1.0 / count;
      gyroYOffset = -gyroYTotal * 1.0 / count;
      gyroZOffset = -gyroZTotal * 1.0 / count;
    }

    double getXAngle() {
      this->readAngles();
      return this->xAngle;
    };

    double getYAngle() {
```

```
    this->readAngles();
    return this->yAngle;
  };

  double getZAngle() {
    this->readAngles();
    return this->zAngle;
  };

  double getXAcc() {
    this->readAngles();
    return this->accX;
  };

  double getYAcc() {
    this->readAngles();
    return this->accY;
  };



  void getXYZAngles(double &x, double &y, double &z) {
    this->readAngles();
    x = xAngle;
    y = yAngle;
    z = zAngle;
  }

private:
  int i2cAddress;
  double accX, accY, gyroX, gyroY, gyroZ;
  double gyroXOffset, gyroYOffset, gyroZOffset;
  double xAngle, yAngle, zAngle;
  unsigned long prevMillis;

  void readAngles() {
    if (millis() - this->prevMillis < 3)
      return;

    int accRawX, accRawY, accRawZ, gyroRawX, gyroRawY, gyroRawZ;


    Wire.beginTransmission(this->i2cAddress);
    Wire.write(0x3B);
```

```
      Wire.endTransmission(false);
      Wire.requestFrom(this->i2cAddress, 14, true);

      accRawX = Wire.read() << 8 | Wire.read();
      accRawY = Wire.read() << 8 | Wire.read();
      accRawZ = Wire.read() << 8 | Wire.read();
      Wire.read(); Wire.read();
      gyroRawX = Wire.read() << 8 | Wire.read();
      gyroRawY = Wire.read() << 8 | Wire.read();
      gyroRawZ = Wire.read() << 8 | Wire.read();

      accX = atan((accRawY / 16384.0) / sqrt(pow((accRawX / 16384.0), 2) +
pow((accRawZ / 16384.0), 2))) * RAD2DEG;
      accY = atan(-1 * (accRawX / 16384.0) / sqrt(pow((accRawY / 16384.0), 2) +
pow((accRawZ / 16384.0), 2))) * RAD2DEG;

      gyroX = (gyroRawX + gyroXOffset) / 131.0;
      gyroY = (gyroRawY + gyroYOffset) / 131.0;
      gyroZ = (gyroRawZ + gyroZOffset) / 131.0;

      unsigned long curMillis = millis();
      double duration = (curMillis - prevMillis) * 1e-3;
      prevMillis = curMillis;

      xAngle = 0.98 * (xAngle + gyroX * duration) + 0.02 * accX;
      yAngle = 0.98 * (yAngle + gyroY * duration) + 0.02 * accY;
      zAngle = zAngle + gyroZ * duration;
    }
};

void mpu6050Init(SMPU6050 &smpu, int address) {
  smpu.init(address);
}

void mpu6050Calibrate(SMPU6050 &smpu, int times) {
  smpu.calibrate(times);
}

double mpu6050GetXAngle(SMPU6050 &smpu) {
  return smpu.getXAngle();
}

double mpu6050GetYAngle(SMPU6050 &smpu) {
  return smpu.getYAngle();
```

```
}

double mpu6050GetZAngle(SMPU6050 &smpu) {
  return smpu.getZAngle();
}

void mpu6050GetXYZAngles(SMPU6050 &smpu, double &xAngle, double &yAngle, double
&zAngle) {
  smpu.getXYZAngles(xAngle, yAngle, zAngle);
}

#endif  /*_SAIGONTECH_MPU6050_H_*/
```