TRƯỜNG ĐH CÔNG NGHIỆP THỰC PHẨM TP. HCM

KHOA CÔNG NGHỆ ĐIỆN – ĐIỆN TỬ

BỘ MÔN TỰ ĐỘNG HÓA

----ꕥ📖ꕥ----

ĐỒ ÁN CHUYÊN NGÀNH TỰ ĐỘNG HÓA

XE HAI BÁNH TỰ CÂN BẰNG

GVHD: Trần Hoàn

SVTH: Trần Bá Vạn

MSSV:2032176604

TP. HỒ CHÍ MINH, tháng 5 năm 2021

TRƯỜNG ĐH CÔNG NGHIỆP THỰC PHẨM TP. HCM

KHOA CÔNG NGHỆ ĐIỆN – ĐIỆN TỬ

BỘ MÔN TỰ ĐỘNG HÓA

----ꕥ📖ꕥ----

ĐỒ ÁN CHUYÊN NGÀNH TỰ ĐỘNG HÓA

XE HAI BÁNH TỰ CÂN BẰNG

GVHD: Trần Hoàn

SVTH: Trần Bá Vạn

MSSV:2032176604

TP. HỒ CHÍ MINH, tháng 5 năm 2021

TRƯỜNG ĐH CÔNG NGHIỆP THỰC PHẨM TP. HCM

KHOA CÔNG NGHỆ ĐIỆN – ĐIỆN TỬ

BỘ MÔN: TỰ ĐỘNG HÓA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

TP. HCM, ngày….tháng…..năm……..

NHẬN XÉT ĐỒ ÁN CHUYÊN NGÀNH

CỦA GIẢNG VIÊN HƯỚNG DẪN

Tên đồ án:

XE HAI BÁNH TỰ CÂN BẰNG

Sinh viên thực hiện:		**Giảng viên hướng dẫn:**
Trần Bá Vạn	2032176604	Trần Hoàn

Đánh giá Luận văn

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo: .....
.....
.....
.....

2. Về nội dung đồ án: .....
.....
.....
.....

3. Về tính ứng dụng: .....
.....

.....
.....

4. Về thái độ làm việc của sinh viên:

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

Đánh giá chung: .....
.....
.....
.....

Điểm từng sinh viên:

Trần Bá Vạn: ..…../10

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CÁM ƠN

Trong suốt quá trình thực hiện đề tài, mặc dù gặp phải nhiều khó khăn nhưng được sự giúp đỡ, hỗ trợ kịp thời từ quý Thầy Cô và các bạn nên đồ án đã hoàn thành đúng tiến độ. Em xin chân thành cảm ơn thầy Trần Hoàn đã tận tình hướng dẫn, chỉ bảo kinh nghiệm quý báu cũng như hỗ trợ phương tiện thí nghiệm trong suốt quá trình tìm hiểu, nghiên cứu đề tài.

Em cũng xin cảm ơn các thành viên trong lớp 08DHTDH1 đã có những ý kiến đóng góp, bổ sung, cũng như động viên khích lệ giúp em hoàn thành tốt đề tài.

Mặc dù nhóm thực hiện đề tài đã cố gắng hoàn thiện được đồ án, nhưng trong quá trình soạn thảo cũng như kiến thức còn hạn chế nên có thể còn nhiều thiếu sót. Nhóm thực hiện đề tài mong nhận được sự đóng góp ý kiến của quý thầy cô cùng các bạn sinh viên.

Sau cùng nhóm thực hiện xin chúc Thầy cô sức khoẻ, thành công và tiếp tục đào tạo những sinh viên giỏi đóng góp cho đất nước. Chúc các anh (chị), các bạn sức khỏe, học tập thật tốt để không phụ công lao các Thầy Cô đã giảng dạy. Nhóm thực hiện xin chân thành cảm ơn.

MỤC LỤC

DANH MỤC KÝ HIỆU, CỤM TỪ VIẾT TẮT

KÝ HIỆU	ĐƠN VỊ	THUẬT NGỮ
M	kg	Khối lượng của bánh xe
M	kg	Khối lượng của robot
R	m	Bán kính bánh xe
W	m	Chiều rộng của robot
D	m	Chiều ngang của robot
H	m	Chiều cao của robot
L	m	Khoảng cách từ trọng tâm của 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		Moment quán tính của động cơ DC
R	ohm	Điện trở động cơ DC
Kb	V sec/rad	Hệ số EMF của động cơ
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
θ_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 và phải
v_l, v_r	V	Điện áp động cơ bánh trái, phải

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH ẢNH

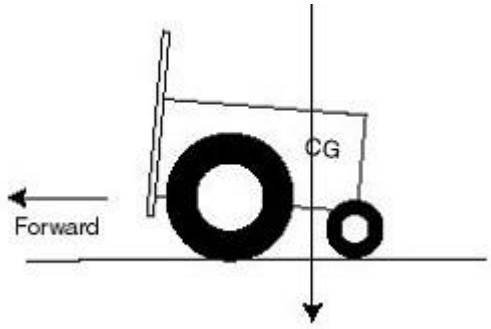
CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Đặt vấn đề và nguyên lý hoạt động

Trong ngành tự động hóa - điều khiển tự động nói chung và điều khiển học nói riêng, mô hình con lắc ngược là một trong những đối tượng nghiên cứu điển hình và đặc thù bởi đặc tính động không ổn định của mô hình nên việc điều khiển được đối tượng này trên thực tế đặt ra như một thử thách.

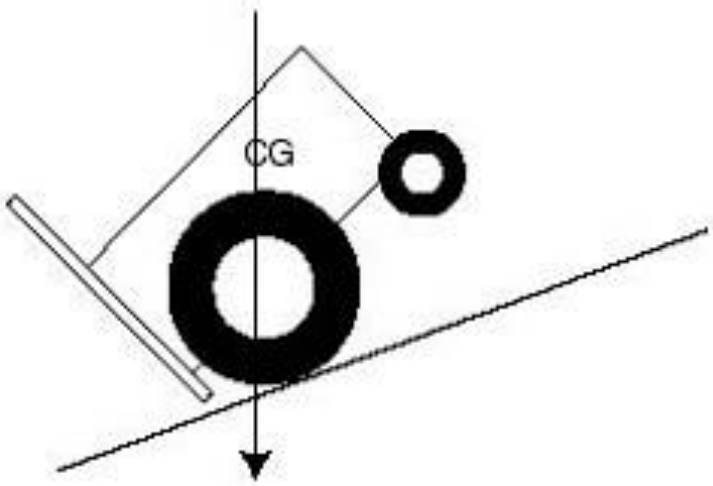
Kết quả nghiên cứu mô hình con lắc ngược cơ bản, ví dụ như mô hình xe-con lắc, con lắc ngược quay… có thể ứng dụng và kế thừa sang các mô hình tương tự khác nhưng có tính ứng dụng thực tiễn hơn, chẳng hạn như mô hình tên lửa, mô hình xe hai bánh tự cân bằng… do đó khắc phục được những nhược điểm vốn có của các robot hai hoặc ba bánh kinh điển. Các robot hai hoặc ba bánh kinh điển, theo đó có cấu tạo gồm bánh dẫn động và bánh tự do (hay bất kì cái gì khác) để đỡ trọng lượng robot. Nếu trọnglượng được đặt nhiều vào bánh lái thì robot sẽ không ổn định và dễ bị ngã, còn nếu đặt vào nhiều bánh đuôi thì hai bánh chính sẽ mất khả năng bám. Nhiều thiết kế robot có thể di chuyển tốt trên địa hình phẳng nhưng không thể di chuyển lên xuống trên địa hình lồi lõm hoặc mặt phẳng nghiêng. Khi di chuyển lên đồi, trọng lượng robot dồn vào đuôi xe làm mất khả năng bám và trượt ngã.

Robot dạng 3 bánh xe di chuyển trên địa hình bằng phẳng trọng lượng được chia đều cho bánh lái và bánh dẫn nhỏ.



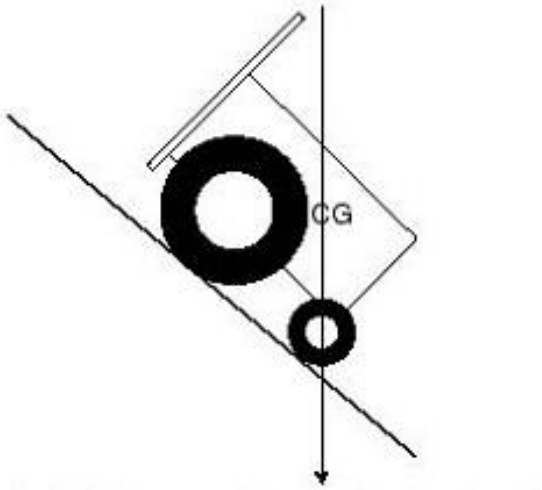
Hình 1. 1 Robot ba bánh di chuyển trên địa hình phẳng.

Robot dạng 3 bánh xe khi xuống dốc, trọng lực dồn vào bánh sau khiến xe có thể bị lật úp.



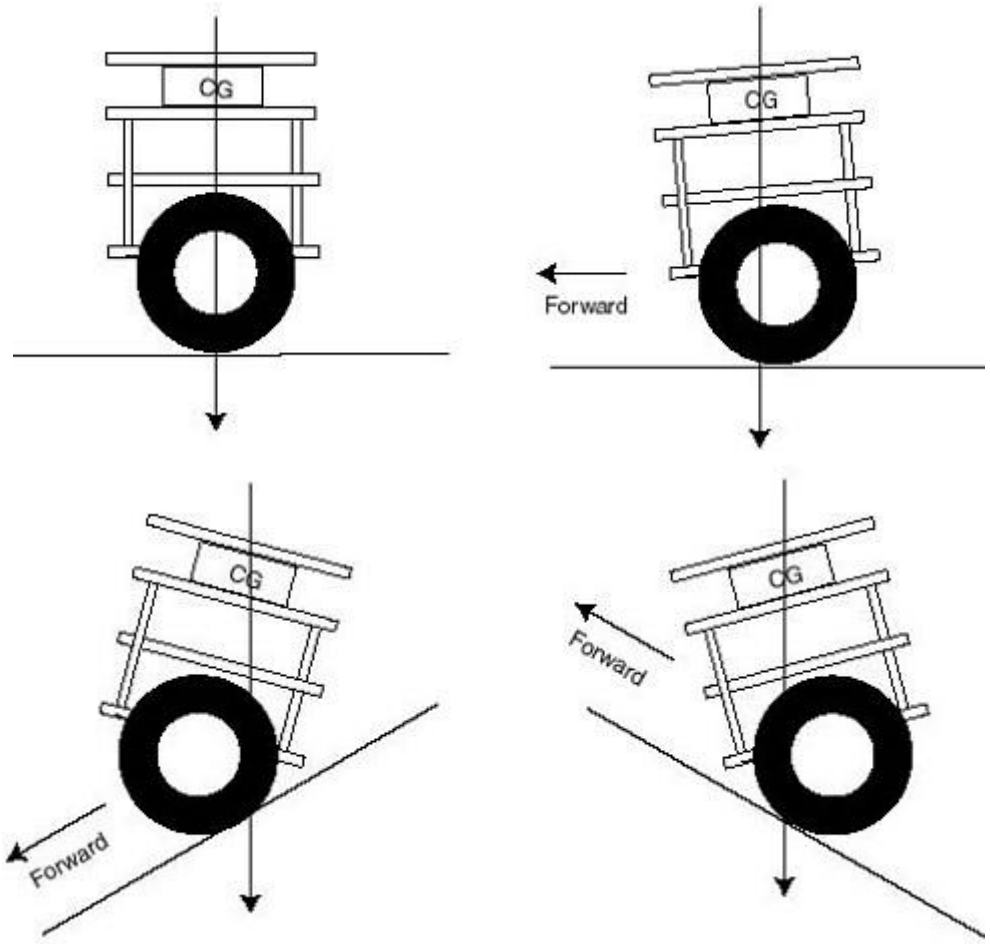
Hình 1. 2 Robot ba bánh di chuyển xuống dốc.

Robot dạng 3 bánh xe khi lên dốc, trọng lượng dồn vào bánh trước khiến lực ma sát giúp xe bám trên mặt đường không được đảm bảo.

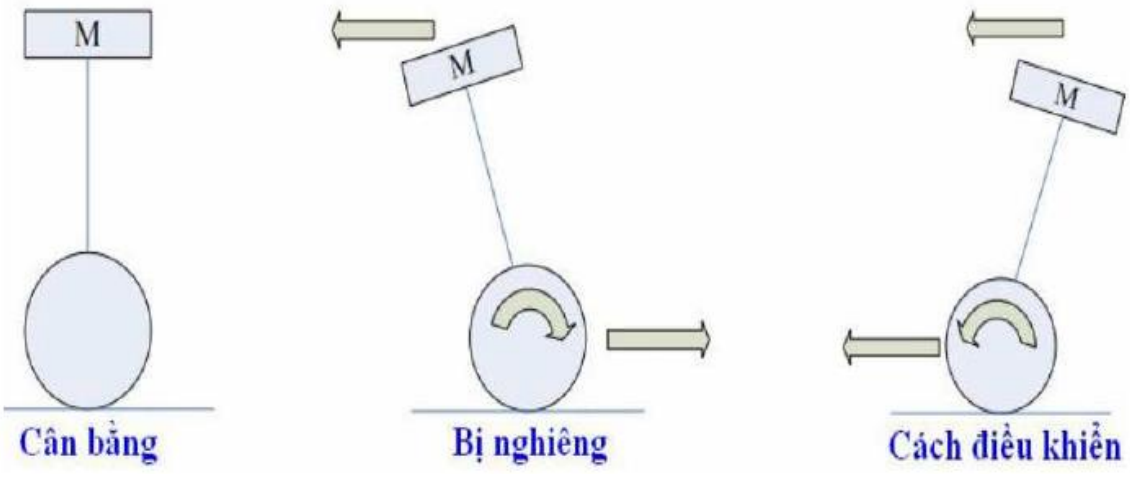


Hình 1. 3 Robot ba bánh di chuyển xuống dốc.

Ngược lại, các robot dạng hai bánh đồng trục lại thăng bằng rất linh động khi di chuyển trên địa hình phức tạp, mặc dù bản thân robot là một hệ thống không ổn định. Khi robot di chuyển trên địa hình dốc, nó tự động nghiêng ra trước và giữ cho trọng lượng dồn về hai bánh chính. Tương tự, khi di chuyển xuống dốc, nó nghiêng ra sau và giữ trọng tâm rơi vào bánh chính. Vì vậy, không bao giờ có hiện tượng trọng tâm xe rơi ngoài vùng đỡ bánh xe để có thể gây ra lật úp.



Hình 1. 4 Robot hai bánh di chuyển trên địa hình khác nhau theo hướng tự cân bằng.



Hình 1.5 Nguyên lý hoạt động của Rô Bốt tự cân bằng

Khi rô bốt đứng cân bằng thì góc nghiêng θ của thân rô bốt và trục thẳng đứng (trục của lực trọng trường) bằng 0.

Khi xe nghiêng về phía trước $\theta > 0$, nếu không có điều khiển thì theo quán tính, hai bánh rô bốt sẽ chạy về phía sau (phía ngược lại của thân rô bốt đang

nghiêng), dẫn đến rô bốt sẽ bị đổ. Nên trong trường hợp này, chúng ta sẽ điều khiển cho 2 bánh của rô bốt chạy về phía trước (phía mà rô bốt đang bị nghiêng) làm cho góc lệch $\theta = 0$, rô bốt sẽ thăng bằng trở lại.

Khi rô bốt nghiêng về phía sau $\theta < 0$, nếu không có điều khiển thì theo quán tính hai bánh rô bốt sẽ chạy về phía trước (phía ngược lại của thân rô bốt đang bị nghiêng) dẫn đến rô bốt bị đổ. Nên trong trường hợp này, chúng ta sẽ điều khiển cho hai bánh xe chạy về phía sau (phía mà rô bốt đang bị nghiêng) làm cho góc lệch $\theta = 0$, rô bốt sẽ thăng bằng trở lại.

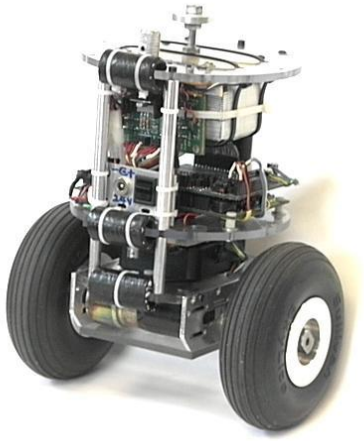
1.2 Các công trình nghiên cứu liên quan

❖ nBot

nBot do David P.Anderson sáng chế. nBot được lấy ý tưởng để cân bằng như sau: các bánh xe sẽ phải chạy theo hướng mà phân trên robot sắp ngã. Nếu bánh xe có thể được lái lái theo cách đứng vững theo trọng tâm robot, robot sẽ vẫn được giữ cân bằng. Trong thực tế, điều này đòi hỏi hai cảm biến thông tin phản hồi: cảm biến góc nghiêng để đo góc nghiêng của robot với trọng lực, và encoder trên bánh xe để đo vị trí của robot. Bốn thông số ngõ vào để xác định hoạt động và vị trí của xe con lắc ngược cân bằng là:

1) Góc nghiêng.
2) Đạo hàm của góc nghiêng, vận tốc góc
3) Vị trí bánh xe.
4) Đạo hàm vị trí bánh xe, vận tốc bánh xe.

Bốn giá trị đo lường được cộng lại và phản hồi tới điện áp động cơ, tương ứng với momen quay, cân bằng, và bộ phận lái robot.



Hình 1. 6 nBot

❖ Balance Bot I

Balance-bot I (do Sanghuyk, Hàn Quốc thực hiện) là một robot hai bánh tự cân bằng bằng cách kiểm soát thông tin phản hồi. Hệ thống cao 50cm. Khung chính được làm bằng nhôm. Nó có hai trục bánh xe nối với hộp giảm tốc và động cơ DC cho sự phát động. Tổng cộng có ba bộ vi xử lý Atmel đucợ sử dụng. Vi điều khiển chính (master) thi hành những nguyên lý kiểm soát và thuật toán ước lượng. Một vi điều khiển khác kiểm soát tất cả cảm biến analog. Vi điều khiên thứ ba điều khiên động DC.

Linear quadratic regulator (LQR) được thiết kế và thực thi mạch điều khiển. Nó có bốn giá trị khác nhau – góc nghiêng, vận tốc góc nghiêng, góc quay bánh xe, và vận tốc góc quay, sau đó nó tạo lệnh cho động cơ DC để điều chỉnh tốc độ bánh xe.

Hình 1. 7Balance Bot I

- JOE

Phòng thí nghiệm điện tử công nghiệp của Viện Công nghệ Federal, Lausanne, Thụy Sĩ, đã tạo ra cuộc các mạng đầu tiên khi xây dựng mô hình xe hai bánh. Robot JOE cao 65cm, nặng 12kg, tốc độ tối đa khoảng 1,5m/s, có khả năng leo dốc nghiêng đến 30◦. Nguồn điện cấp là pin 32V.

Hình dạng của nó gồm hai bánh xe trụ, mỗi bánh gắn với một động cơ DC, chiếc xe này có thể chuyển động xoay theo hình U. Hệ thống điều khiển được lắp từ hai bộ điều khiển state-space tách rời nhau, kiểm soát động cơ để giữ cân bằng cho hệ thống. Những thông tin về trạng thái của JOE được cung cấp bởi hai encoder quang và vận tốc con quay hồi chuyển.

JOE được điều khiển bởi một bộ điều khiển từ xa R/C thường được sử dụng để điều khiển các máy bay mô hình. Bộ điều khiển trung tâm và xử lý tín hiệu là một board sử lý tín hiệu số (DSP) được phát triển bởi chính nhóm và của viện Federal, có khả năng sử lý dấu chấm động (SHARC floating point), FPGA XILINC, 12 bộ biến đổi A/D 12 bit và 4 bộ biến đổi D/A 10 bit.

Hình 1. 8 Robot JOE

1.3 Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng mô hình robot 2 bánh tự cân bằng dựa trên nền tảng lý thuyết mô hình con lắc ngược. Trong thời gian làm đề tài, những mục tiêu của đề tài được đặt ra như sau:

- Tìm hiểu các mô hình xe, robot 2 bánh tự cân bằng và các nguyên lý cơ bản về cân bằng.

- Tìm hiểu và áp dụng Bộ lọc Kalman để lọc nhiễu cho cảm biến, xây dựng các thuật toán bù trừ để có giá trị góc chính xác.

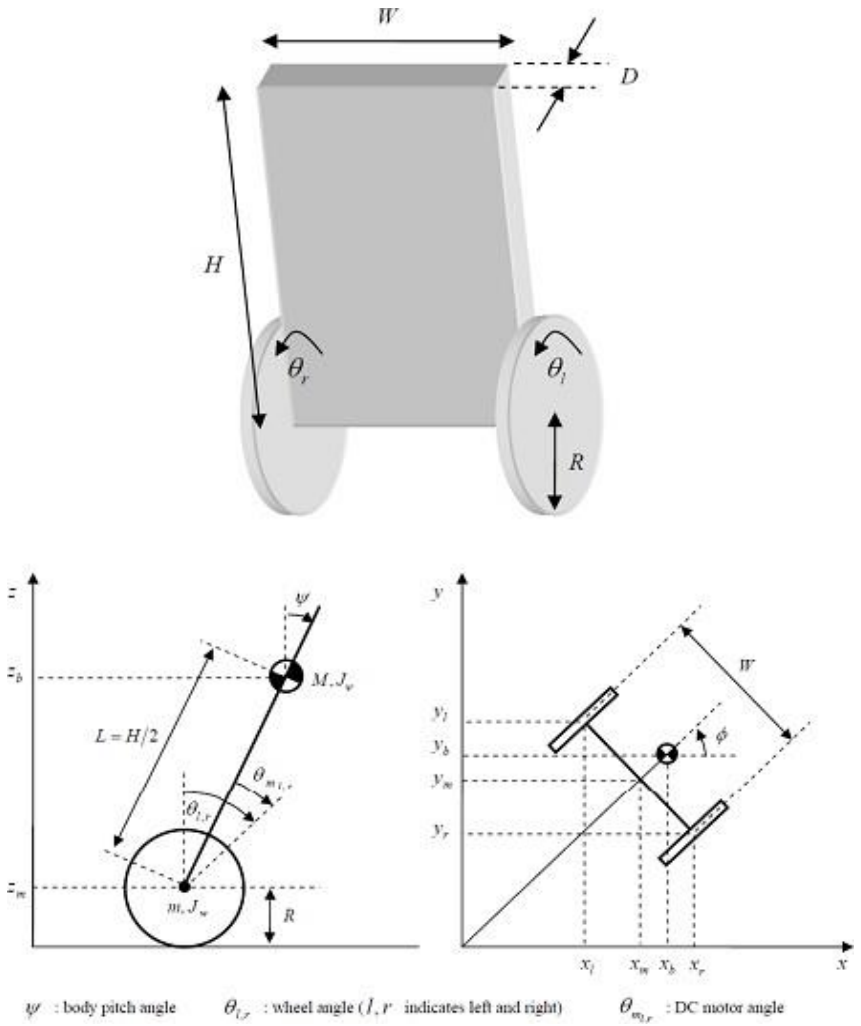
- Xây dựng thuật toán điều khiển động cơ, giữ thăng bằng cho robot.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Đặc tính động lực học

2.1.1 Mô hình robot 2 bánh tự cân bằng trên địa hình phẳng

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



Hình 2. 1 Mô hình robot 2 bánh tự cân bằng

Trong đề tài này sẽ sử dụng các kí hiệu, đơn vị trong danh mục kí hiệu và cụm từ viết tắt.

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:

$$\begin{pmatrix} \boldsymbol{\theta} \\ \phi \end{pmatrix} = \begin{pmatrix} \frac{1}{2}(\theta_l + \theta_r) \\ \frac{R}{W}(\theta_l - \theta_r) \end{pmatrix} \quad [2.1]$$

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

$$\begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = \begin{bmatrix} \int \dot{x}_m \\ \int \dot{y}_m \\ R \end{bmatrix} \quad [2.2]$$

Và $$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} R\dot{\theta}\cos\phi \\ R\dot{\theta}\sin\phi \end{bmatrix} \quad [2.3]$$

Tọa độ bánh trái trong hệ qui chiếu :

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{pmatrix} x_m - \frac{W}{2}\sin\phi \\ y_m + \frac{W}{2}\cos\phi \\ Z_M \end{pmatrix} \quad [2.4]$$

Tọa độ bánh phải trong hệ qui chiếu :

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{pmatrix} x_m - \frac{W}{2}\sin\phi \\ y_m + \frac{W}{2}\cos\phi \\ Z_M \end{pmatrix} \quad [2.5]$$

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

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} x_m + L\sin\psi\cos\phi \\ y_m L\, sin\,\psi\sin\phi \\ z_m + L\cos\phi \end{bmatrix} \quad [2.6]$$

Phương trình động năng của chuyển động tịnh tiến:

$$T_1 = \frac{1}{2}m(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2) + \frac{1}{2}m(\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2) + \frac{1}{2}m(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2) \quad [2.7]$$

Phương trình động năng của chuyển động quay :

$$T_2 = \frac{1}{2}J_W\dot{\theta}_l^2 + \frac{1}{2}J_W\dot{\theta}_r^2 + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_l - \dot{\psi}\right)^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_r - \dot{\psi}\right)^2 \quad [2.8]$$

Với

$\frac{1}{2}n^2J_m(\dot{\theta}_r - \dot{\psi})^2$; $\frac{1}{2}n^2J_m(\dot{\theta}_r - \dot{\psi})^2$ là động năng quay của phần ứng động cơ trái và phải.

Phương trình thế năng:

$$U = mgz_l + mgz_r + mgz_b \quad [2.10]$$

Phương trình lagrange:

$$L = T_1 + T_2 - U \quad [2.11]$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = F_\theta \quad [2.12]$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = F_\psi \quad [2.13]$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} = F_\phi \quad [2.14]$$

Lấy đạo hàm L theo các biến ta được:

$$[(2m + M)R^2 + 2J_W + 2n^2J_m]\ddot{\theta} + (MLR\cos\psi - 2n^2J_m)\ddot{\psi} - MLR\dot{\psi}^2\sin\psi = F_\theta \quad [2.15]$$

$$(MLR\cos\psi - 2n^2J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\sin\psi - ML^2\dot{\phi}^2\sin\psi\cos\psi = F_\psi \quad [2.16]$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}(J_W + n^2J_m) + ML^2sin^2\psi\right]\ddot{\phi}^2 + 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi = F_\phi \quad [2.17]$$

Momen động lực do động cơ DC sinh ra:

$$\begin{bmatrix} F_\theta \\ F_\psi \\ F_\phi \end{bmatrix} = \begin{bmatrix} F_l + F_r \\ F_\psi \\ \frac{W}{2R}(F_l - F_r) \end{bmatrix} \quad [2.18]$$

Và:

$$F_l = nK_t i_l + f_m\left(\dot{\psi} - \dot{\theta}_l\right) - f_W\dot{\theta}_l \quad [2.19]$$

$$F_r = nK_t i_r + f_m\left(\dot{\psi} - \dot{\theta}_r\right) - f_W\dot{\theta}_r \quad [2.20]$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m\left(\dot{\psi} - \dot{\theta}_l\right) - f_m\left(\dot{\psi} - \dot{\theta}_r\right) \quad [2.21]$$

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

$$L_m i_{l,r} = v_{l,r} + K_b\left(\dot{\psi} - \dot{\theta}_{l,r}\right) - R_m i_{l,r} \quad [2.22]$$

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

$$i_{l,r} = \frac{v_{l,r} + k_b(\dot{\psi} - \dot{\theta}_{l,r})}{R_m} \quad [2.23]$$

Từ đó, các moment lực sinh ra:

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \quad [2.24]$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \quad [2.25]$$

Với $\alpha = \frac{nk_t}{R_m}$ và $\beta = \frac{nK_tK_b}{R_m} + f_m$ [2.26]

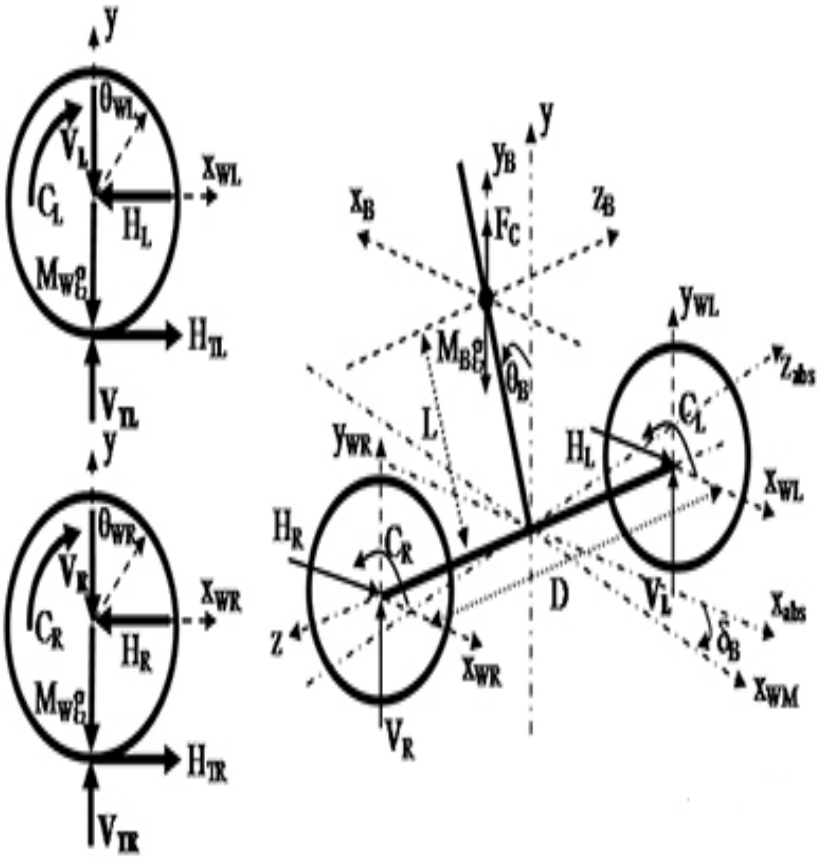
$$F_\phi = \frac{W}{2R}\alpha(v_r - v_l) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \quad [2.27]$$

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

$$[(2m + M)R^2 + 2J_w + 2n^2J_m]\ddot{\theta} + (MLR\cos\psi - |2n^2J_m)\ddot{\psi} - MLR\dot{\psi}^2\sin\psi = \alpha(v_l + v_r) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \quad [2.28]$$

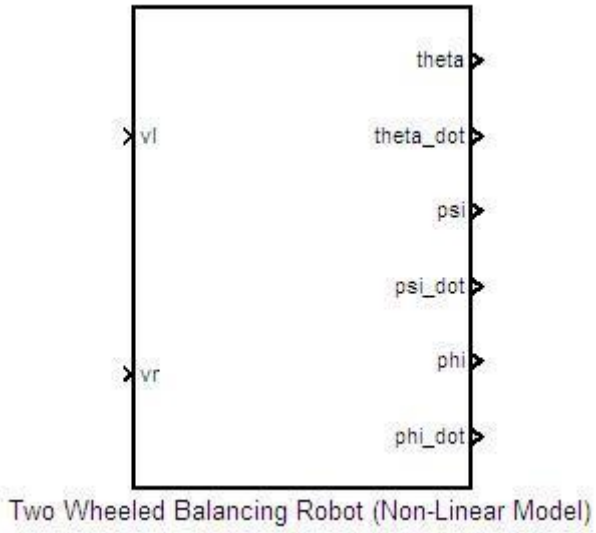
$$(MLR\cos\psi - 2n^2J_m)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2J_m\right)\ddot{\psi} - MgL\sin\psi - ML^2\dot{\phi}^2\sin\psi\cos\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \quad [2.29]$$

$$\left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}(J_w + n^2J_m) + ML^2\sin^2\psi\right]\ddot{\phi}^2 + 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi = \frac{W}{2R}\alpha(v_r - v_l) - \frac{W^2}{2R^2}(\beta + f_w)\dot{\phi} \quad [2.30]$$

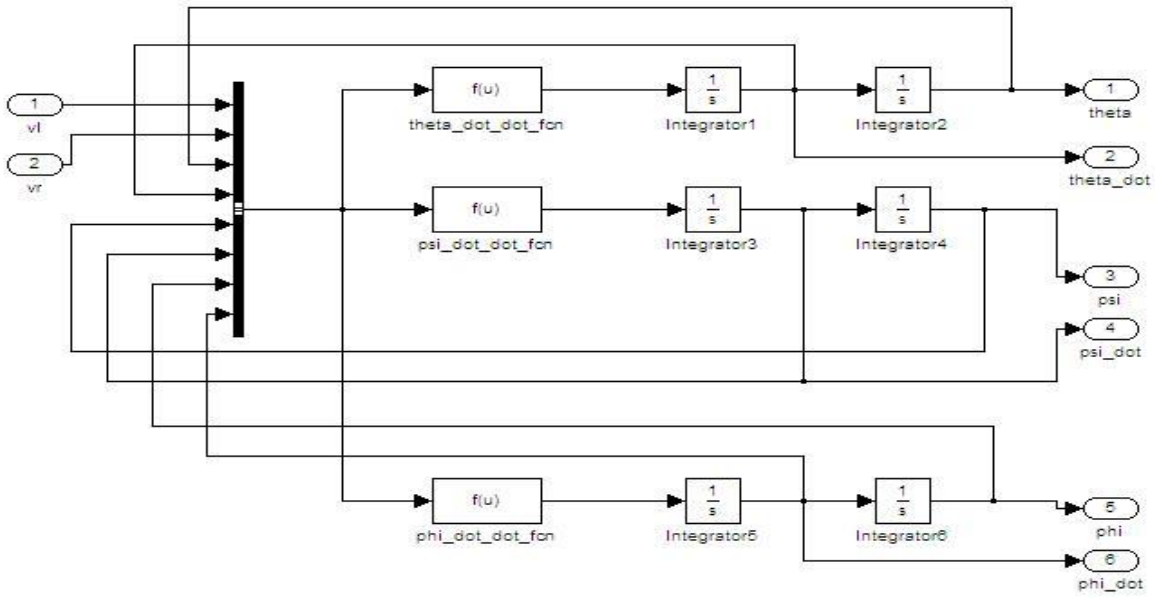


Hình 2. 2 Biểu diễn lực và momen của mô hình

2.1.2 Mô hình hóa xe hai bánh tự cân bằng trên địa hình phẳng trong Matlab Simulink



Hình 2. 3 Mô hình phi thuyền của xe trong Matlab Simulink



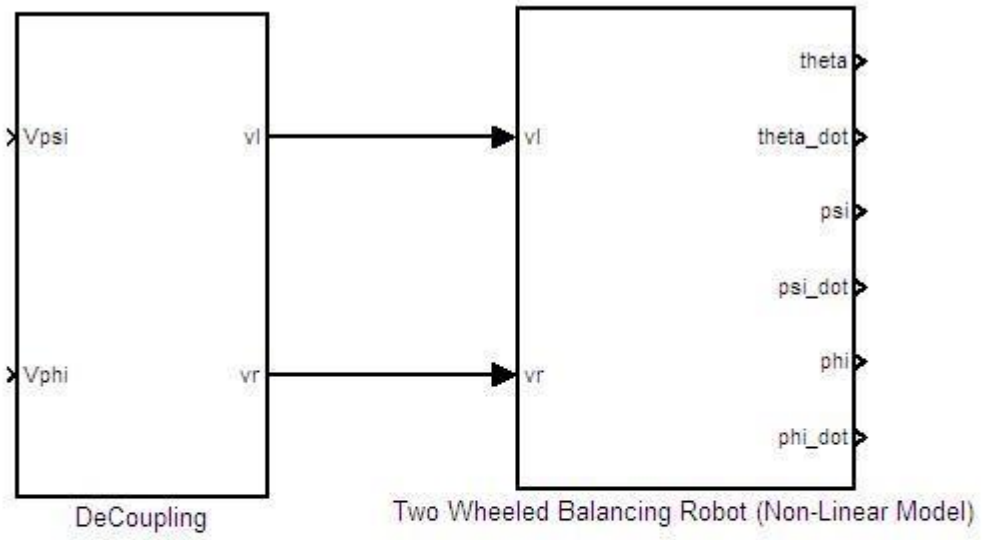
Hình 2. 4 Bên trong khối "Two Wheeled Balancing Robot (Non-Linear Model)"

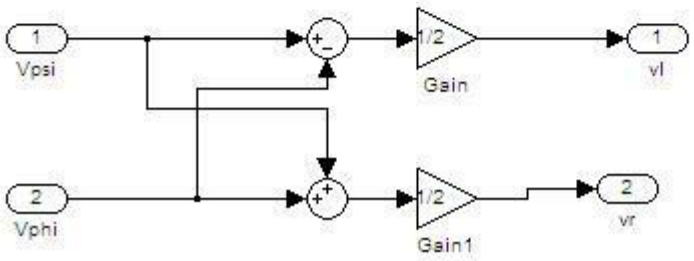
Phương trình động lực học của robot như trên thể hiện mối quan hệ giữa giá trị điện áp điều khiển hai động cơ, với độ nghiêng, vị trí, vận tốc của hệ robot, giá trị điện áp hai động cơ v_l, v_r tác động lên các thông số đó dưới dạng tổng $v_l + v_r$ còn với góc xoay, giá trị điện áp hai động cơ v_l, v_r tác động

lên thông số này dưới dạng hiệu v_r-v_l. Khi đó, tách bài toán hệ robot thành hai bài toán nhỏ hơn với hai tín hiệu điều khiển V_ψ, V_ϕ

$$\begin{cases} v_\psi = v_l + v_r \\ v_\phi = v_r - v_l \end{cases} \Rightarrow \begin{cases} v_l = \frac{1}{2}(V_\psi - V_\phi) \\ v_r = \frac{1}{2}(V_\psi + V_\phi) \end{cases} \quad [2.31]$$

Khối thực hiện chức năng này gọi là khối phân tách (decoupling)





Hình 2. 5 Bên trong khối "DeCoupling"

2.2 Bộ lọc Kalman

2.2.1 Giới thiệu bộ lọc Kalman

Bộ lọc Kalman, được Rudolf (Rudy) E. Kálmán công bố năm 1960, là thuật toán sử dụng chuỗi các giá trị đo lường, bị ảnh hưởng bởi nhiễu hoặc sai số, để ước đoán biến số nhằm tăng độ chính xác so với việc sử dụng duy nhất một giá trị đo

lường. Bộ lọc Kalman thực hiện phương pháp truy hồi đối với chuỗi các giá trị đầu vào bị nhiễu, nhằm tối ưu hóa giá trị ước đoán trạng thái của hệ thống.

Bộ lọc Kalman được ứng dụng rộng rãi trong kỹ thuật, phổ biến trong các ứng dụng định hướng, định vị và điều khiển các phương tiện di chuyển. Ngoài ra, bộ lọc Kalman còn được ứng dụng để phân tích dữ liệu trong các lĩnh vực xử lý tín hiệu và kinh tế.

2.2.2 Quá trình ước lượng

Vấn đề chung của bộ lọc Kalman nhằm ước lượng biến trạng thái$x \in Rn$ của quá trình điều khiển rời rạc được điều chỉnh bởi các phương trình tuyến tính ngẫu nhiên khác nhau. Phương trình không gian trạng thái của bộ lọc:

$$xk = Axk-1 + Buk-1 + wk-1 \quad [2.32]$$

Với giá trị $z \in Rm$là:

$$zk = Hxk + vk \quad [2.33]$$

Biến ngẫu nhiên wk, vk đặc trưng cho nhiễu quá trình và nhiễu đo của hệ. Chúng độc lập với nhau, tần suất phân bố thông thường:

$$p(w)\sim N(0, Q) \quad [2.34]$$

$$p(v)\sim N(0, R) \quad [2.35]$$

Trên thực tế, ma trận tương quan nhiễu quá trình Q và ma trận tương quan nhiễu đo R có thể thay đổi sau mỗi bước thời gian hay giá trị, tuy nhiên để đơn giản, trong hầu hết các trường hợp Q và R được xem là hằng số.

Ma trận vuông A trong phương trình [2.32] thể hiện mối quan hệ của các biến trạng thái ở thời điểm k-1 với thời điểm hiện tại k. Thực ra trên thực tế ma trận A thay đổi sau mỗi bước thời gian, nhưng ở đây ma trận A xem như hằng số. Ma trận B thể hiện mối liên hệ tín hiệu điều khiển $u \in RL$đối với biến trạng thái x. Ma trận H trong phương trình [2.33] thể hiện mới liên hệ giữa biến trạng thái với tín hiệu ra z, H cũng được xem là hằng số.

Những tính toán căn bản của bộ lọc:

Định nghĩa:

$\hat{x}k^{-} = E\{xk|y1, y2 \dots yk-1\}$là giá trị ước lượng của x_k trước khi ta xử lý giá trị đo tại thời điểm k.

$\hat{x}k \in Rn$ là giá trị ước lượng trạng thái sau tại bước k có được sau khi so sánh với giá trị đo z_k.Và chúng ta có sai số ước lượng trạng thái trước và sau:

$$\begin{cases} e_{\bar{k}} \equiv x_k - \hat{x}_{\bar{k}} \\ e_k \equiv x_k - \hat{x}_k \end{cases} \quad [2.36]$$

Tương quan sai số ước lượng trước "priori":

$$P_{\bar{k}} = E\{e_{\bar{k}} e_{\bar{k}}\} \quad [2.37]$$

Tương quan sai số ước lượng sau "posteriori":

$$P_k = E\{e_k e_k^T \quad [2.38]$$

Khi lấy đạo hàm phương trình bộ lọc Kalman, với mục đích tìm một phương trình để tính toán trạng thái ước lượng posteriori $\hat{x}k$ thể hiện sự tương quan giữa giá trị ước lượng priori $\hat{x}k^-$ và độ sai lệch giữa giá trị đo thực zk và giá trị đo ước lượng $H\hat{x}k^-$:

$$\hat{x}_k = \hat{x}_{\bar{k}} + K(Z_k - H\hat{x}_{\bar{k}}) \quad [2.39]$$

Ma trận K trong [3.8] là ma trận độ lợi hay hệ số trộn để tối thiểu hóa phương trình tương quan sai số posteriori. Biểu thức tính K để tối thiểu hóa phương trình [3.8] như sau:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R} \quad [2.40]$$

Từ đó thấy rằng tương quan sai số giá trị đo lường R tiến tới 0, khi đó:

$$\lim_{R_K \to 0} K_{K=H^{-1}} \quad [2.41]$$

Mặt khác, tương quan sai số ước lượng priori của P_k tiến đến 0, khi đó:

$$\lim_{P_{\bar{k}} \to 0} K_k = 0 \quad [2.42]$$

Một cách nghĩ khác về giá trị hiệu chỉnh bù bởi K là nếu ma trận tương quan sai số giá trị đo lường R tiến tới 0 thì giá trị đo được z_k sẽ có độ tin cậy càng cao, trong khi giá trị ước lượng $Hx_{\bar{k}}$ sẽ có độ tin cậy càng thấp. Mặt khác, nếu tương quan sai số ước lượng priori P_k^- tiến tối 0 thì z_k sẽ không đáng tin mà giá trị ước lượng $Hx_{\bar{k}}$ sẽ càng đáng tin.

2.2.3 Bản chất xác xuất của bộ lọc

Sự điều chỉnh cho x_k trong [3.8] đã xác định bản chất ước lượng priori $\hat{x}_k$ với điều kiện tất cả các giá trị đo z_k đều có nghĩa (Luật phân bố Bayer). Điều đó cho thấy bộ lọc Kalman duy trì hai thời điểm đầu tiên của sự phân bố trạng thái:

$$E[x_k] = \hat{x}_k \quad [2.43]$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k \quad [2.44]$$

Phương trình ước lượng trạng thái posteriori phản ánh giá trị trung bình của phân bố trạng thái. Tương quan sai số ước lượng trạng thái posteriori phản ánh sự thay đổi của phân bố trạng thái. Ngoài ra ta còn có:

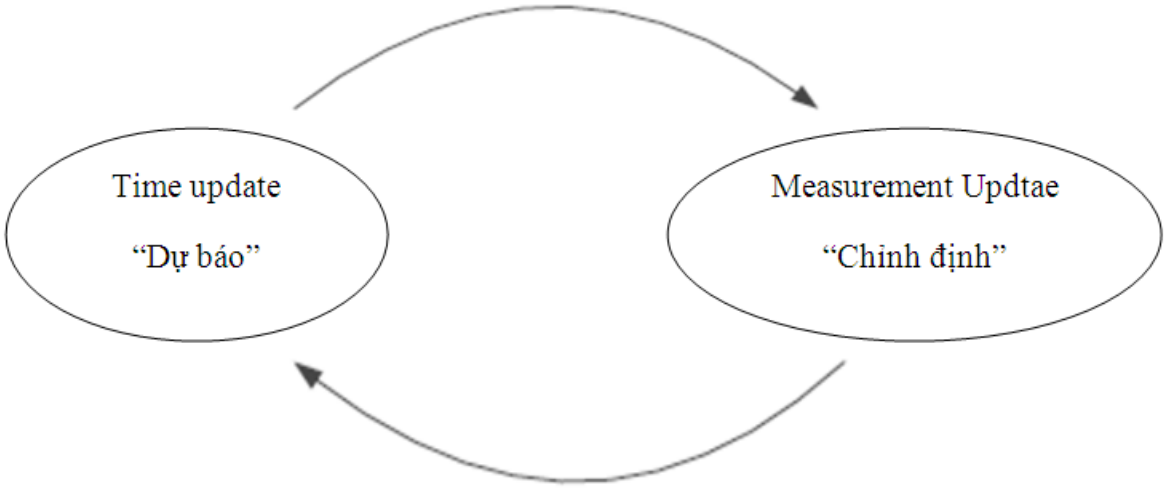
$$p(x_k|z_k) \sim N(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]) = N(\hat{x}_k, P_k) \quad [2.45]$$

2.2.4 Thuật toán Kalman rời rạc

Bộ lọc Kalman ước lượng tiến trình bằng cách sử dụng dạng điều khiển hồi tiếp: bộ lọc ước lượng các trạng thái của quá trình tại một vài thời điểm và sau đó chứa tín hiệu hồi tiếp trong các dạng của giá trị đolường. Do đó, phương trình bộ lọc Kalman chia làm hai nhóm: phương trình cập nhật thời gian và phương trình cập nhật giá trị đo lường. Phương trình cập nhật thời gian chịu trách nhiệm cho việc dự báo trước (về mặt thời gian) của trạng thái hiện tại và ước lượng sai số tương quan để chứa vào bộ ước lượng trước

priori cho bước thời gian tiếp theo. Phương trình cập nhật giá trị đo lường chịu trách nhiệm cập nhật cho tín hiệu hồi tiếp, nghĩa là cập nhật giá trị mới vào giá trị ước lượng tước priori để tạo tín hiệu ước lượng sau posteriori tốt hơn.

Phương trình cập nhật thời gian cũng có thể được coi là phương trình dự đoán. Trong khi đó phương trình cập nhật giá trị đo lường thì được xem như là phương trình hiệu chỉnh.Vì vậy, thuật toán ước lượng cuốicùng đều giống nhau ở thuật toán dự đoán và hiệu chỉnh để giải quyết vấn đề số học như hình vẽ dưới đây:



Hình 2. 6 Quy trình thực hiện của bộ lọc Kalman

Phương trình cập nhật thời gian cho bộ lọc Kalman rời rạc:

$$\hat{x}_{\bar{k}} = A\hat{x}_{\bar{k}-1} + Bu_{k-1} \quad [2.46]$$

$$P_k^- = AP_{k-1}A^T + Q \quad [2.47]$$

Phương trình cập nhật giá trị đo lường cho bộ lọc Kalman rời rạc:

$$K_k = P_k^- H^T (HP^- H^T + R)^{-1} \quad [2.48]$$

$$\hat{x}_k = \hat{x}_{\bar{k}} K(z_k - H\hat{x}_{\bar{k}}) \quad [2.49]$$

$$P_k = (1 - K_k H)P_k^- \quad [2.50]$$

Nhiệm vụ đầu tiên trong suốt quá trình cập nhật giá trị đo lường là tính toán độ lợi KalmanK_k. Bước tiếp theo là xử lí giá trị đo thực được chứa trong z_k. Sau đó, tính trạng thái ước lượng sau posteriori bằng cách kết hợp giá trị đo được theo công thức $\hat{x}_k$ ở trên. Bước cuối cùng là tính giá trị sai số ước lượng tương quan posteriori vào P_k. Sau mỗi chu trình tính toán của bộ lọc Kalman, các giá trị được cập nhật theo cặp, tiến trình được lặp lại với ước lượng posteri ori của trạng thái trước dùng để dự đoán ước lượng priori mới. Trạng thái đệ quy tự nhiên là một trong những điểm đặc trưng của bộ lọc Kalman, nó thay thế điều kiện đệ quy ước lượng hiện tại cho giá trị đã qua.

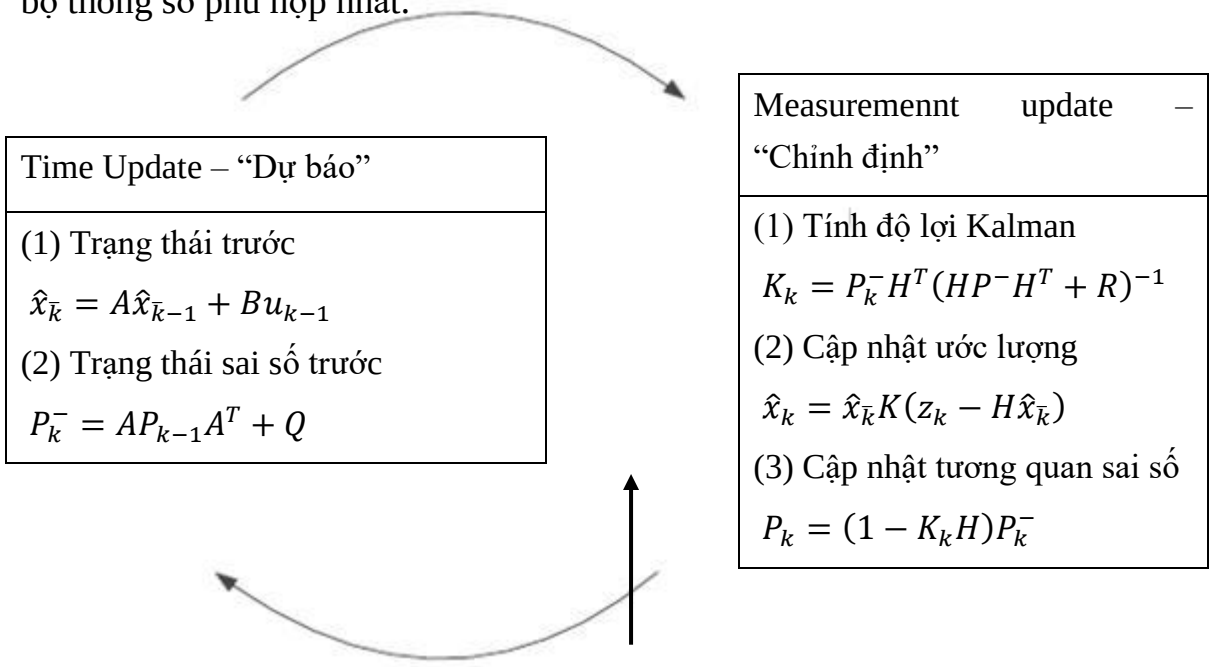
Trong điều kiện thực hiện thực tế của bộ lọc, giá trị nhiễu tương quan R thường được dùng làm giá trị ưu tiên để tính toán cho bộ lọc. Trên thực tế, việc đo các giá trị ma trận R là rất phổ biến bởi vì chúng ta có thể đo quy trình theo nhiều cách vì vậy mà thường lấy mẫu giá trị để đưa ra khuynh hướng thay đổi của giá trị nhiễu.

Sự xác định rõ tương quan nhiễu quá trình Q thường rất khó bởi vì điều điển hình là chúng ta không có khả năng quan sát trực tiếp tiến trình mà chúng ta đang ước lượng. Đôi khi sự liên hệ tới những quy trình mẫu đơn giản có thể đưa ra những giá trị chấp nhận được nếu một mẫu xen vào không chắc chắn đủ với tiến trình thông qua sự lựa chọn Q . Chắc chắn trong trường hợp này,mẫu đó sẽ hi vọng rằng giá trị tiến trình là đáng tin cậy.

Trong những trường hợp khác, dù muốn hay không chúng ta đều có cái chuẩn hợp lí cho việc lựa các thôngsố, thường thì chất lượng bộ lọc sẽ tốt hơn nhiều lần khi có chứa sự hiệu chỉnh các tham số Q và R.Sự hiệu chỉnh thường được thực hiện gián tiếp, thường thì với sự giúp đỡ của một bộ lọc Kalman khác trong quy trình chung, liên hệ như một hệ thống đồng nhất.

Với điều kiện Q và R là các hằng số thực,cả hai cho phép ước lượng sai số tương quan P_k và độ lợi Kalman K_k sẽ ổn định nhanh chóng và sau đó trở thành hằng số.

Trong điều kiện luận văn, thông số Q và R được hiệu chỉnh dựa vào quá trình thử sai để dự đoán khuynh hướng hiệu chỉnh của hệ thống và tìm ra bộ thông số phù hợp nhất.



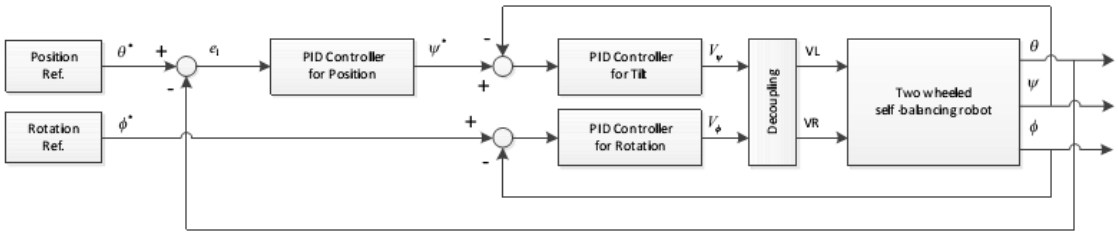
Hình 2. 7 Tổng quan chu trình thực hiện bộ lọc Kalman hoàn chỉnh

2.3 Giải thuật điều khiển

2.3.1 Cấu trúc bộ điều khiển PID cho xe robot hai bánh tự cân bằng

Ba bộ PID được sử dụng để điều khiển robot hai bánh tự cân bằng, bao gồm:

- Bộ PID điều khiển góc nghiêng (ψ)
- Bộ PID điều khiển vị trí (θ)
- Bộ PID điều khiển góc xoay (ϕ)



Hình 2. 8 Cấu trúc bộ điều khiển PID cho hệ xe hai bánh tự cân bằng

Hàm truyền đạt bộ điều khiển PID liên tục:

$$G_{PID}(S) = \frac{U(S)}{E(S)} = K_P + \frac{K_I}{s} + K_D\left(\frac{s}{1+\tau S}\right) \quad [2.51]$$

Rời rạc hóa đạo hàm theo thời gian:

$$y(k) \approx \frac{f(k)-f(k-1)}{T_S} \quad [2.52]$$

Rời rạc hóa tích phân theo thời gian:

$$\int_0^{k.T_S} y(t)dt = y(k) \approx y(k-1) + \frac{f(k)-f(k-1)}{2}.T_S \quad [2.53]$$

Phép biến đổi rời rạc (z-Tranform)

$$X[z] = \mathcal{Z}\{\mathrm{x[k]}\} = \sum_{k=0}^{\infty} x[k]z^{-k} \quad [2.54]$$

$$z = Ae^{j\theta} = A(\cos\theta + J\sin\theta) \quad [2.55]$$

Ta có $\mathcal{Z}\{x[k-n]\} = z^{-n}X[z]$ và $\mathcal{Z}\{x[k]\} = X[z]$ [2.56]

Do đó

$$y(k) = \frac{f(k)-f(k-1)}{T_S} \rightarrow y[z] = \frac{F[z]-z^{-1}F[z]}{T_S} \Longrightarrow \frac{y[z]}{F[z]} = \frac{z-1}{zT_S} \quad [2.57]$$

Và

$$y(k) = y(k-1) + \frac{f(k)-f(k-1)}{2}.T_S \rightarrow Y[z](1-z^{-1}) = \frac{T_S}{2}F[z](1+z^{-1}) \Rightarrow \frac{y[z]}{F[z]} = \frac{T_S}{2}\frac{z+1}{z-1} \quad [2.58]$$

$$\frac{U[z]}{E[z]} = K_p + K_i\frac{T_S}{2}\frac{z+1}{z-1} + K_d\frac{z-1}{zT_S} \quad [2.59]$$

$$\Leftrightarrow \frac{U[z]}{E[z]} = \frac{K_p(z^2-z)+K_i\frac{T_s}{2}(z^2+z)+\frac{K_d}{T_s}(z^2-2z+1)}{z^2-z}$$

$$\Leftrightarrow \frac{U[z]}{E[z]} = \frac{\left(K_p+K_i\frac{T_s}{2}+\frac{K_d}{T_s}\right)z^2+\left(-K_p+K_i\frac{T_s}{2}+\frac{K_d}{T_s}\right)Z+\frac{K_d}{T_s}}{z^2-z}$$

$$\Leftrightarrow \frac{U[z]}{E[z]} = \frac{\left(K_p+K_i\frac{T_s}{2}+\frac{K_d}{T_s}\right)+\left(-K_p+K_i\frac{T_s}{2}+\frac{K_d}{T_s}\right)z^{-1}+\frac{k_d}{T_s}z^{-2}}{1-z^{-1}}$$

$$\Leftrightarrow U[z] = z^{-1}U[z] + aE[z] + bz^{-1}E[z] + cz^{-2}E[z] \quad [2.60]$$

$$\Leftrightarrow u[k] = u[k-1] + ae[k] + be[k-1] + ce[k-2] \quad [2.61]$$

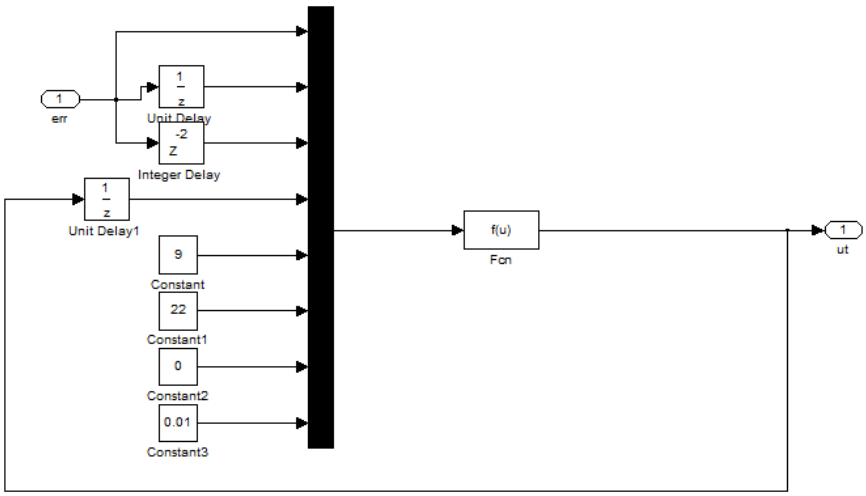
Trong đó: $a = K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s}$; $b = -K_p + K_i\frac{T_s}{2} + \frac{K_d}{T_s}$; $c = \frac{K_d}{T_s}$ [2.62]

2.3.2 Bộ điều khiển PID với thông số cố định

Bộ điều khiển PID với thông số K_P, K_I, K_D cố định như trên, hệ thống chỉ làm

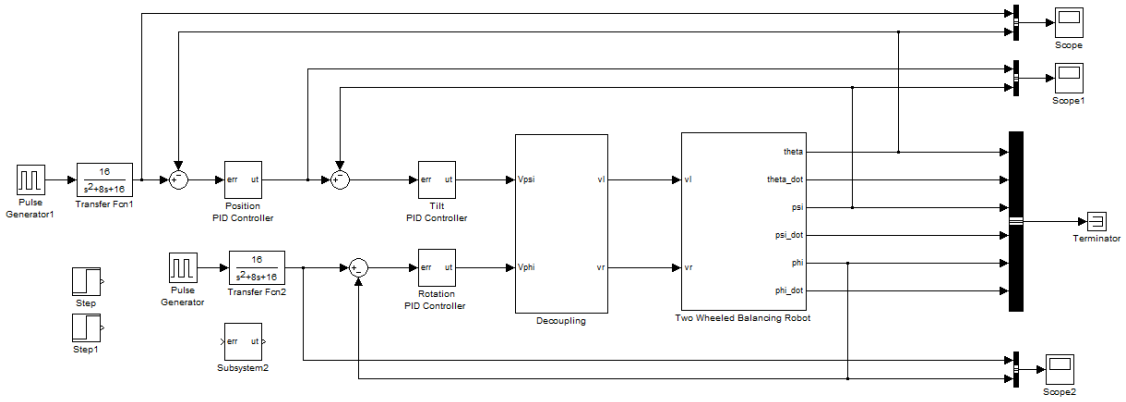
việc tốt trong điều kiện hệ số K_P, K_I, K_D đã được chỉnh định tối ưu và trong quá trình làm việc, các thông số trong mô hình không đổi.

Cấu trúc bộ điều khiển PID rời rạc với hệ số cố định sử dụng để điều khiển góc nghiêng, vị trí và góc xoay là như nhau và được hiện thực như sau trong Matlab Simulink.



Hình 2. 9 Cấu trúc bên trong bộ điều khiển PID rời rạc với thông số cố định

Hệ thống điều khiển robot hai bánh tự cân bằng sử dụng bộ PID rời rạc với thông số cố định hiện thực như sau:



Hình 2. 10 Xe hai bánh tự cân bằng sử dụng 3 bộ điều khiển PID cố định

Trong đó, mô hình Robot 2 bánh tự cân bằng trong trường hợp này sẽ được thay đổi các thông số như khối lượng thân Robot(M) và hệ số ma sát giữa bánh xe với bề mặt di chuyển (f_w) thông qua file "Two_wheelRobot.m"

Có thể thấy, đáp ứng ngõ ra vị trí của hệ thống bị ảnh hưởng một cách rõ rệt, ngõ ra vị trí của hệ thống không thể tiếptục đáp ứng theo tín hiệu đặt như ban đầu khi hệ số f_w thay đổi. Còn các bộ PID điều khiển góc nghiêng thì vẫn đảm bảo ngõ ra hệ thống bám tốt theo tín hiệu đặt.

Đặc biệt khi có sự thay đổi về khối lượng Robot (M), ngõ ra vi trí của hệ thống bị vọt lố khá lớn. Tuy nhiên khả năng điều khiển góc nghiêng của bộ PID vẫn duy trì và bám khá tốt với tín hiệu đặt.

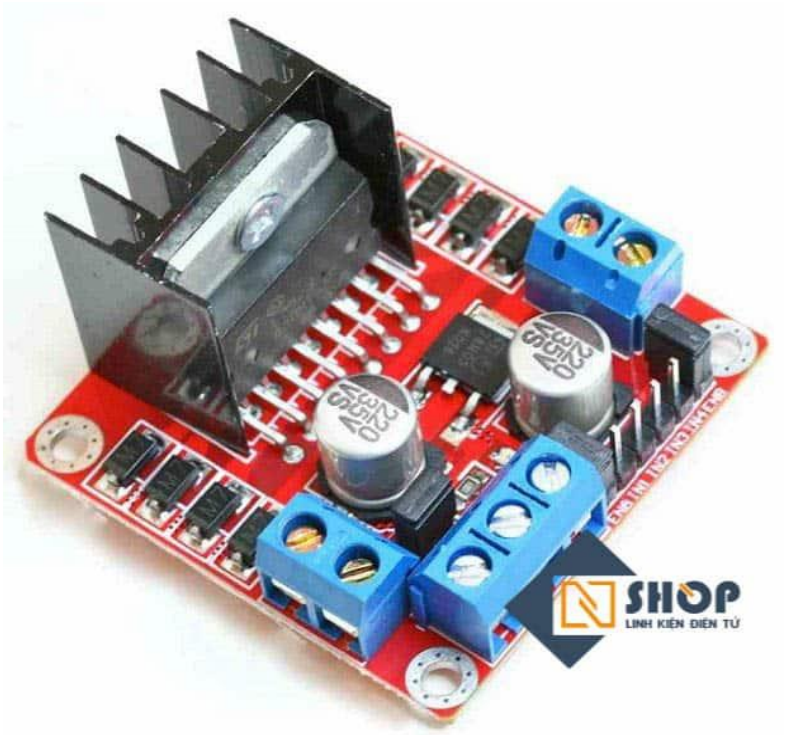
Như vậy trong trường hợp có sự thay đổi hệ số ma sát fw và khối lượng Robot M trong mô hình thì bộ PID cố định điều khiển vị trí không thể đảm bảo được chất lượng điều khiển. Tuy nhiên khâu góc nghiêng và góc xoay đáp ứng khá tốt.

2.3.3 Các thành phần chính của mô hình

2.3.3.1 Mạch điều khiển động cơ DC L298N

Mạch điều khiển động cơ DC L298N có khả năng điều khiển 2 động cơ DC, dòng tối đa 2A mỗi động cơ, mạch tích hợp diod bảo vệ và IC nguồn 7805 giúp cấp nguồn 5VDC cho các module khác (chỉ sử dụng 5V này nếu nguồn cấp <12VDC).

Mạch điều khiển động cơ DC L298N dễ sử dụng, chi phí thấp, dễ lắp đặt, là sự lựa chọn tối ưu trong tầm giá.

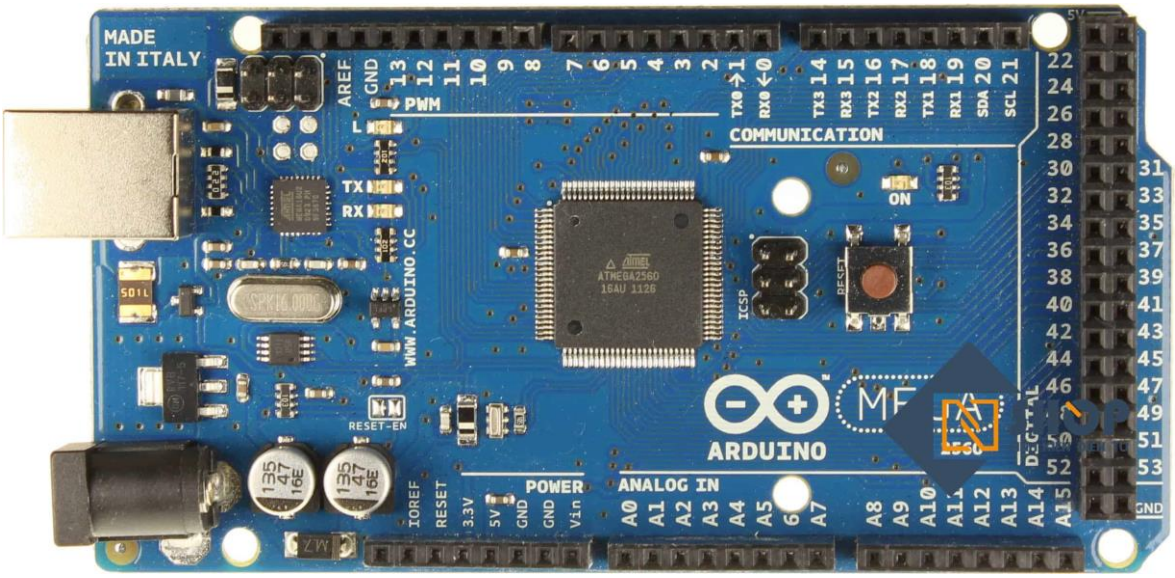


Hình 2. 11 Mạch điều khiển động cơ L298N

THÔNG SỐ KỸ THUẬT

- IC chính: L298 – Dual Full Bridge Driver
- Điện áp đầu vào: 5~30VDC
- Công suất tối đa: 25W 1 cầu (lưu ý công suất = dòng điện x điện áp nên áp cấp vào càng cao, dòng càng nhỏ, công suất có định 25W).
- Dòng tối đa cho mỗi cầu H là: 2A
- Mức điện áp logic: Low -0.3V~1.5V, High: 2.3V~Vss
- Kích thước: 43x43x27mm

2.3.3.2 Arduino MEGA 2560 R3



Hình 2. 12 Arduino MEGA 2560 R3

Mạch điện Arduino Mega 2560 R3 Atmega16u2 là phiên bản nâng cấp của Arduino Uno R3 với số chân giao tiếp, ngoại vi và bộ nhớ nhiều hơn, mạch được thiết kế và sử dụng các linh kiện tương đương với phiên bản chính hãng trên Arduino.cc, phù hợp cho các ứng dụng cần nhiều bộ nhớ hoặc nhiều chân, cổng giao tiếp hơn so với Arduino Uno, giá của mạch đã bao gồm cáp USB.

Mạch Arduino Mega 2560 Atmega là một board vi điều khiển dựa trên ATmega2560. Board này có 54 chân I/O (14 chân PWM), 16 analog đầu hàng vào, 4 UARTs (phần cứng cổng tuần tự), sử dụng thạch anh 16 MHz, kết nối cổng USB, một Jack cắm điện, chân ICSP, và một nút reset. Board có tất cả mọi thứ cần thiết để hỗ trợ vi điều khiển.

Thông Số Kỹ Thuật:

- IC nạp và giao tiếp UART: ATmega16U2.
- Nguồn nuôi mạch: 5VDC từ cổng USB hoặc nguồn ngoài cắm từ giắc tròn DC (khuyên dùng 7-9VDC để đảm bảo mạch hoạt động tốt. Nếu bạn cắm 12V thì IC ổn áp rất dễ chết và gây hư hỏng mạch).
- Số chân Digital: 54 (15 chân PWM)
- Số chân Analog: 16

- Giao tiếp UART : 4 bộ UART
- Giao tiếp SPI : 1 bộ (chân 50 -> 53) dùng với thư viện SPI của Arduino
- Giao tiếp I2C : 1 bộ
- Ngắt ngoài : 6 chân
- Bộ nhớ Flash: 256 KB, 8KB sử dụng cho Bootloader
- SRAM: 8 KB
- EEPROM: 4 KB
- Xung clock: 16 MHz

Tất cả các Shield của Arduino Uno đều chạy được với Arduino Mega.

2.3.3.3 Động cơ DC giảm tốc GA25 370 130rpm

Hình 2. 13 Động cơ DC giảm tốc GA25 130rpm

Động cơ DC giảm tốc GA25 thường được sử dụng trong các ứng dụng cần xác định tốc độ, vị trí, chiều quay của động cơ DC: Robot mê cung, robot xe hai bánh tự cân bằng,…

Động cơ DC giảm tốc GA25 thực tế là động cơ DC GA25 thường có gắn thêm phần Encoder để có thể trả xung về vi điều khiển giúp xác định vị trí, vận tốc,… Về cách điều khiển thì động cơ DC giảm tốc GA25 sử dụng Driver để điều khiển công suất động cơ, tốc độ và đảo chiều: L298, L293,….

Thông số kỹ thuật:

- Điện áp cung cấp : 6~18VDC

- Tốc độ sau hộp 130rpm
- Moment xoắn : 5 kg.cm = 0,49 N.m
- Dòng điện không tải : 50mA

2.3.3.4 Bánh xe 65mm khớp lục giác

Hình 2. 14 Bánh xe 65mm khớp lục giác

Thông số kỹ thuật:

- Chất liệu: Nhựa cứng, lớp đệm mút, cao su tốt
- Đường kính: 65mm
- Độ rộng bánh: 27mm

2.3.3.5 Pin cell 18650 2000mAh

Hình 2. 15 Pin cell 18650 200mAh

Thông số kỹ thuật:

- Điện áp: 3.7v
- Dung lượng: 2000mah
- Điện áp sạc đầy: 4.2v

2.3.3.6 Trục đồng đực cái 55mm

Hình 2. 16 Trục đồng đực cái

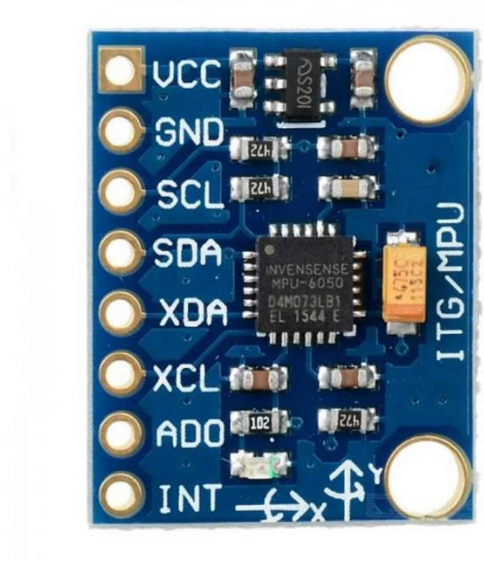
2.3.3.7 Hộp đế pin 18650 3 cell

Hình 2. 17 Hộp đế pin 18650 3 cell

Thông số kỹ thuật:

- Các Cell pin được nối tiếp với nhau với điện áp tối đa: 4.2 x 3
- Dây màu đỏ là +
- Dây màu đen là –

2.3.3.8 Cảm biến gia tốc GY-521 6DOF IMU MPU6050



Hình 2. 18 Cảm biến gia tốc GY-521 6DOF IMU MPU6050

Cảm biến gia tốc GY-521 6DOF IMU MPU6050 được sử dụng để đo 6 thông số: 3 trục Góc quay (Gyro), 3 trục gia tốc hướng (Accelerometer), là loại cảm biến gia tốc phổ biến nhất trên thị trường hiện nay, ví dụ và code dành cho nó rất nhiều và hầu như có trên mọi loại vi điều khiển.

Thông số kỹ 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
- Board mạch mạ vàng, linh kiện hàn tự động bằng máy chất lượng tốt nhất.

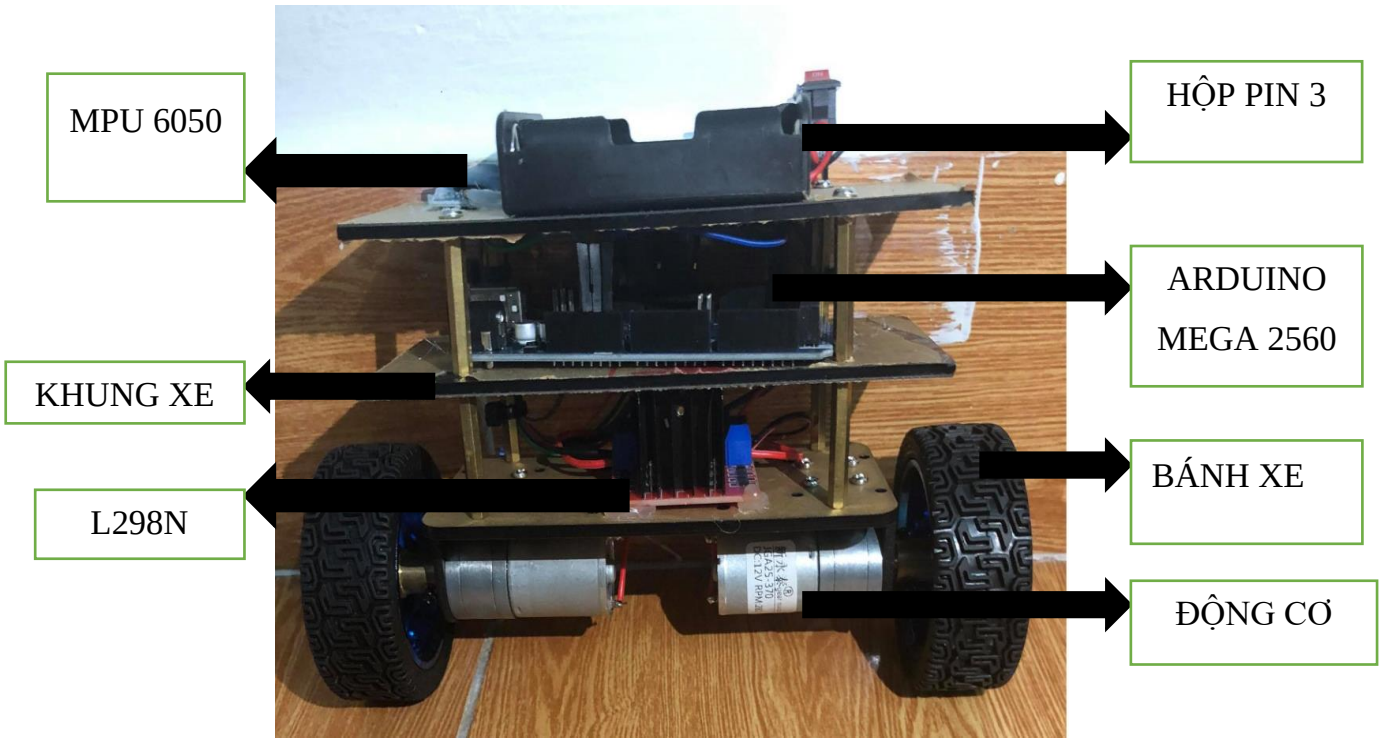
CHƯƠNG 3: CƠ SỞ THỰC HIỆN

3.1 Thiết kế phần cứng mô hình xe hai bánh tự cân bằng

3.1.1 Thiết kế cơ khí

Khung sườn sử dụng chất liệu mica màu 4mm, được cố định theo cơ cấu ghép rãnh . Cơ cấu chuyền động bố trí mạch theo chiều dựng đứng.

Mô hình cơ khí của hệ xe hai bánh tự cân bằng được thiết kế thể hiện ở hình 5.1 hệ gồm 3 phần chính : Khung xe, bánh xe và 2 động cơ, có thể điều chỉnh tốc độ quay của 2 động cơ thông qua mạch công suất L298N.



Hình 3. 1 Mô Hình robot thực tế

Một số thông số mô hình như sau:

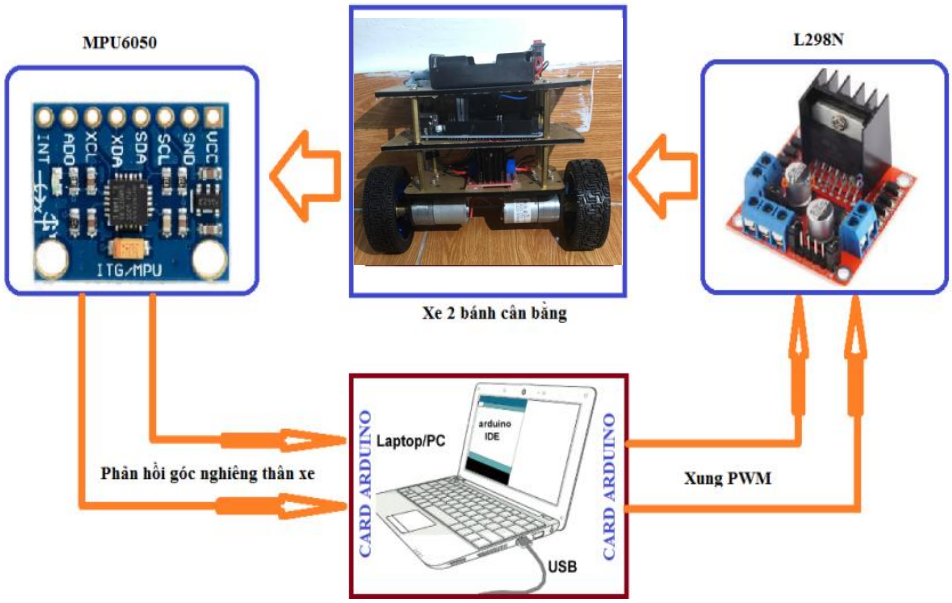
- Khối lượng bánh xe: m = 0.09 (kg).
- Khối lượng robot: M = 1 (kg).
- Bán kính bánh xe: R = (m).
- Chiều rộng robot: W = 0.01 (m).
- Chiều sâu robot: D = 0.017 (m).

- Chiều cao robot: H = 0.015 (m).
- Khoảng cách từ trọng tâm robot đến trục bánh xe: L = H / 2 (m).
- Hệ số ma sát giữa bánh xe và mặt phẳng di chuyển: f_w
- Hệ số ma sát giữa robot và động cơ DC: f_m
- Moment quán tính của động cơ DC: J_m = 0.002 [kg.m^2]
- Gia tốc trọng trường: g = 9.8 m/s^2
- Góc trung bình của bánh trái và phải : θ [rad]
- Góc nghiêng của phần thân robot: ψ [rad]

Góc xoay của robot: ϕ– [rad]

3.1.2 Hệ thống điều khiển và kết nối phần cứng

3.1.2.1 Hệ thống điều khiển



Hình 3.2 Sơ đồ kết nối hệ thống điều khiển hệ xe 2 bánh cân bằng

Trong đó khối máy tính có nhiệm vụ thu thập dữ liệu của cảm biến góc nghiêng thân xe từ card arduino mega 2560 để xuất dữ liệu ra màn hình.

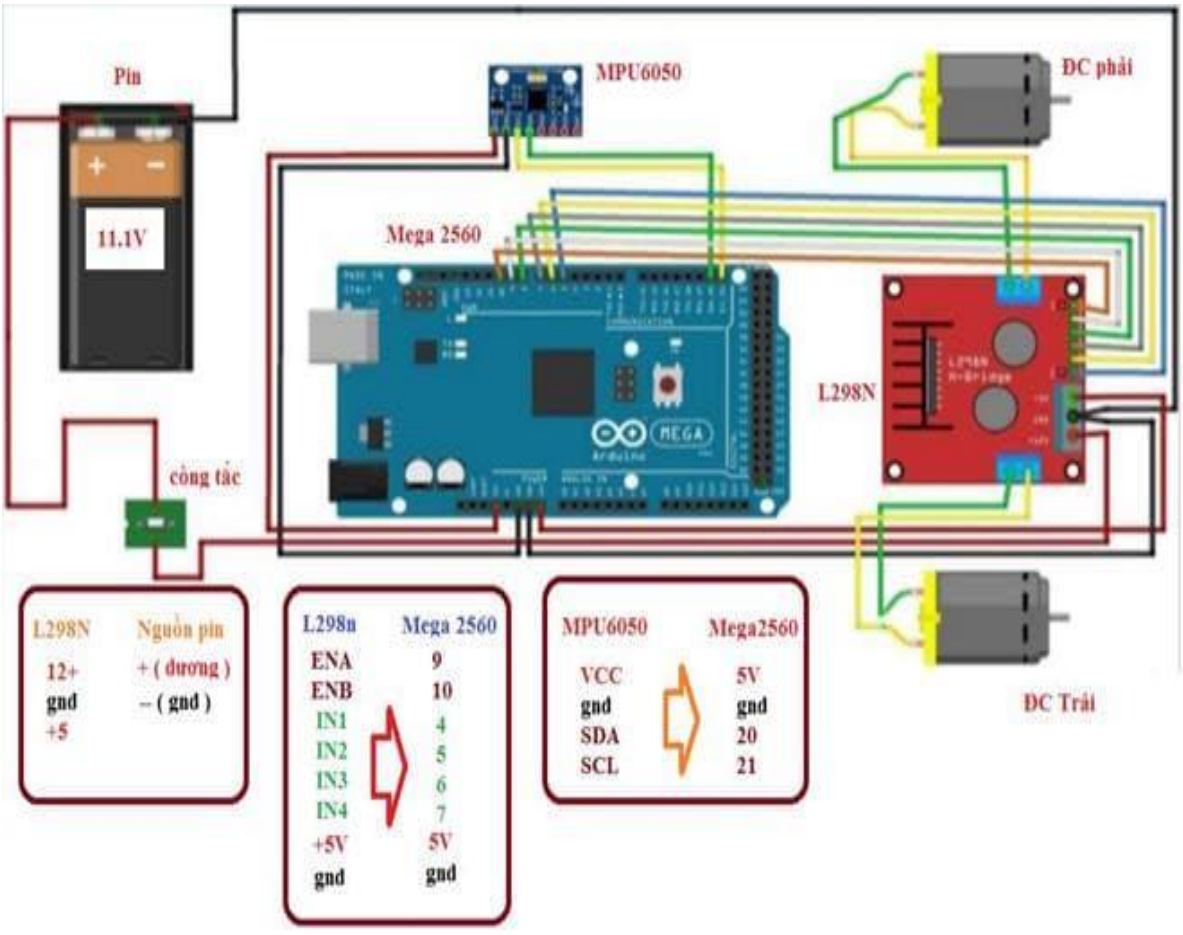
Card arduino mega 2560 có nhiệm vụ đọc dữ liệu từ cảm biến góc nghiêng trên mô hình xe 2 bánh rồi so sánh với giá trị đặt từ đó xuất tín hiệu điều khiển đến các động cơ thông qua mạch công suất (L298).

Khối công suất có nhiệm vụ điều khiển chiều và tốc độ hai động cơ một chiều có gắng 2 bánh xe ở hai đầu.

3.1.2.2. Kết nối phần cứng và sơ đồ kết nối

Board Arduino Mega 2560	**Chức năng**	**Kết nối**	**Ghi chú**
Chân 2	Input	Chân ngắt	INT của MPU 6050
Chân 4	Output	Chân input L298N – IN1	
Chân 5	Output	Chân input L298N – IN2	
Chân 6	Output	Chân input L298N – IN3	Output L298N
Chân 7	Output	Chân input L298N – IN4	nối motor
Chân 9	Output	Chân EA - L298N	
Chân 10	Output	Chân EB - L298N	
Chân 20	Input	Chân SDA cảm biến Gyro	Giao tiếp chuẩn
Chân 21	Input	Chân SCL cảm biến Gyro	I2C

Bảng 3. 3 Kết nối phần cứng



Hình 3.4 : Sơ đồ kết nối

3.2 Thiết kế phần mềm

3.2.1 Bộ lọc Kalman

Robot sử dụng hai cảm biến: cảm biến gia tốc (accelecrometer) và cảm biến con quay hồi chuyển(gyroscope) để đo góc nghiêng và vận tốc góc nghiêng.Tuy nhiên, vấn đề đặt ra là cần phải kết hợp thông tin từ hai cảm biến để xác định chính xác góc nghiêng thực của hệ robot loại bỏ được ảnh hưởng của nhiễu đo và nhiễu quá trình. Để giải quyết vấn đề này, giải thuật lọc Kalman được sử dụng, với mục đích ước lượng giá trị góc nghiêng của hệ robot từ hai loại cảm biến trên và loại bỏ

được ảnh hưởng của nhiễu. Bộ lọc Kalman được khảo sát với mô hình 3 biến trạng thái như sau:

Hình 3. 5 Mô hình Kalman với 3 biến trạng thái.

Với mô hình này, bộ lọc sử dụng 2 biến ngõ vào là vận tốc góc nghiêng từ cảm biến con quay hồi chuyển và góc nghiêng từ cảm biến gia tốc; 1 biến ngõ ra là góc nghiêng ước lượng

Ma trận $P_{khoitao}$ đặc trưng cho tương quan sai số:

Với $P_{khoitao} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ [3.1]

Angle, q_bias lần lượt là góc, vận động sử dụng trong tính toán của bộ lọc.

R tượng trưng cho giá trị nhiễu coveriance. Trong trường hợp này, nó là ma trận 1x1 được mong đợi có giá trị 0.08 rad ≈ 4,5 độ từ cảm biến gia tốc:

R_angle = 0.08 [3.2]

Q là ma trận 2x2 tượng trưng cho giá trị nhiễu coveriance. Trong trường hợp này, nó chỉ mức độ tin cậy của cảm biến gia tốc quan hệ với cảm biến gyro:

$$Q = \begin{bmatrix} q\_angle & 0 \\ 0 & q\_gyro \end{bmatrix} = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.003 \end{bmatrix} \quad [3.3]$$

- "**Giai đoạn tự báo"**

Trong mỗi khoảng chu kỳ quét dt với giá trị cơ sở gyro được canh chỉnh tùy theo nhu cầu sử dụng và cách lắp cơ khí bởi người dùng module IMU. Giai đoạn sẽ cập nhật góc hiện thời và vận tốc ước lượng.

Vecto giá trị X = $\begin{bmatrix} angle \\ gyro\_bias \end{bmatrix}$

[3.4]

Nó chạy trên sự ước lượng giá trị hàm giá trị:

$$\dot{X} = \begin{bmatrix} angle \\ gyro\_bias \end{bmatrix} = \begin{bmatrix} gyro - gyro\_bias \\ 0 \end{bmatrix}$$

[3.5]

Và cập nhật ma trận covariance qua hàm:

$$\dot{X} = AP + PA' + Q$$

[3.6]

$$\dot{X} = A.P + P.A' + Q \; lý \; thuyết \; là \; (P = A.P.A^T + Q)$$

[3.7]

A là Jocobian của $\dot{X}$ với giá trị mong đợi:

$$A = \begin{bmatrix} \frac{d(angle)}{d(angle)} & \frac{d(angle)}{d(gyro\_bias)} \\ \frac{d(gyro\_bias)}{d(angle)} & \frac{d(gyro\_bias)}{d(gyro\_bias)} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$$

[3.8]

Để dễ dàng trong việc lập trình, ma trận P sẽ được khai triển đến mức tối thiểu:

$$\dot{P} = \begin{bmatrix} Q\_angle - P[0][1] - P[1][0] & -P[1][1] \\ -P[1][1] & Q\_gyro \end{bmatrix}$$

[3.9]

$$= \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P[0][0] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix} + \begin{bmatrix} P[0][0] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix} \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} Q\_angle & 0 \\ 0 & Qgyrro \end{bmatrix}$$

$$= \begin{bmatrix} -P[1][0] & -P[1][1] \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -P[0][1] & 0 \\ -P[1][1] & 0 \end{bmatrix} + \begin{bmatrix} Q\_angle & 0 \\ 0 & Q\_gyro \end{bmatrix}$$

[3.10]

Lưu trữ giá trị ước lượng chưa bias của gyro:

$$rate = q = q\_m - q\_bias$$

[3.11]

(Với rate là sai số góc tiên đoán)

Cập nhật ước lượng góc:

Cập nhật ma trận covariance:

$$P = P + \dot{P}dt$$

$$= \begin{bmatrix} P[0][1] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix} + \begin{bmatrix} Q\_angle - P[0][1] - P[0][1] & -P[1][1] \\ -P[1][1] & Q\_gyro \end{bmatrix} dt$$

[3.13]

- **"Giai đoạn cập nhật giá trị bộ lọc Kalman"**

Ma trận C là ma trận 1x2 (giá trị x trạng thái), đó là ma trận Jacobian của giá trị đo lường với giá trị mong đợi. Trường hợp này C là:

$$C = \begin{bmatrix} d(angle\_m) & d(angle\_m) \\ d(angle) & d(gyro\_bias) \end{bmatrix} = [1 \quad 0]$$ (C chính là H)

[3.14]

Vì giá trị góc đáp ứng trực tiếp với góc ước lượng và giá trị góc không quan hệ với giá trị gyro bias nên C_0 cho thấy giá trị trạng thái quan hệ trực tiếp với trạng thái ước lượng như thế nào, C_1 cho thấy giá trị trạng thái không quan hệ với giá trị cơ sở gyro ước lượng.

Error là giá trị khác nhau trong giá trị đo lường và giá trị ước lượng. Trong trường hợp này, nó khác nhau giữa hai gia tốc kế đo góc và góc ước lượng.

$$angle_{error} = angle_m - angle$$

[3.15]

Tính sai số ước lượng. Từ bộ lọc Kalman :

$$E = CPC^T + R = [1 \quad 0]\begin{bmatrix} P[0][0] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + R = P[0][0] + R$$

[3.16]

$$PCt\_0 = C[0]\ x\ P[0][0]$$

[3.17]

$$PCt\_1 = C[0]\ x\ P[1][0]$$

[3.18]

Ước tính bộ lọc Kalman đạt được. từ lý thuyết bộ lọc Kalman:

$$\begin{bmatrix} K0 \\ K1 \end{bmatrix} = \begin{bmatrix} PCt\_0/E \\ PCt\_1/E \end{bmatrix}$$

[3.19]

$$P = (1 - KH)P = P - KHP = P - KCP$$

[3.20]

Ta có phép nhân điểm trôi (floating point)

$$CP = [1 \quad 0]\begin{bmatrix} P[0][0] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix} = [P[0][0] \quad P[0][1]] = [t\_0 \ \ t\_1]$$

[3.21]

$$P = \begin{bmatrix} P[0][0] & P[0][1] \\ P[1][0] & P[1][1] \end{bmatrix} - \begin{bmatrix} K0 \\ K1 \end{bmatrix}[t_0 \quad t_1] = P - \begin{bmatrix} K0.t\_0 & K0.t\_1 \\ K1.t\_0 & K1.t\_1 \end{bmatrix}$$

[3.22]

Cập nhật giá trị ước lượng. Lần nữa, từ Kalman:

$$X = \begin{bmatrix} angle \\ gyro\_bias \end{bmatrix} = X + K(Z_{measure} - HX_{estimate}) = X + K\begin{bmatrix} angle\_err \\ angle\_er \end{bmatrix}$$

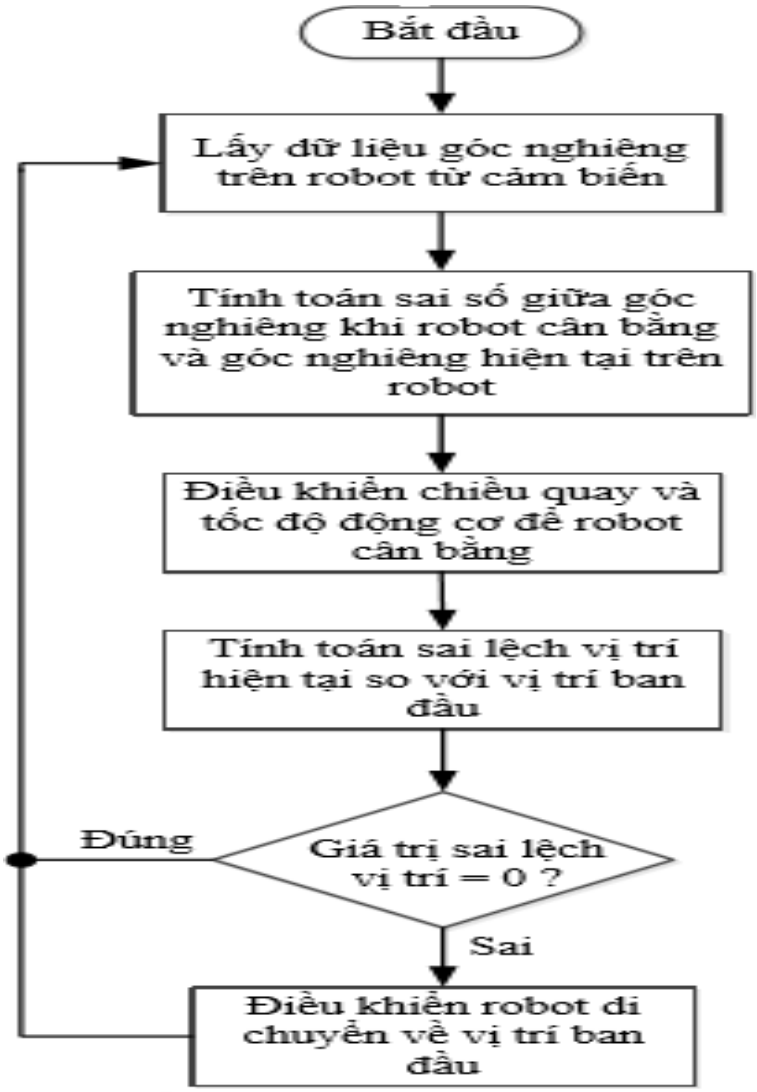
[3.23]

$$angle\_err = q\_bias\_err$$

[3.24]

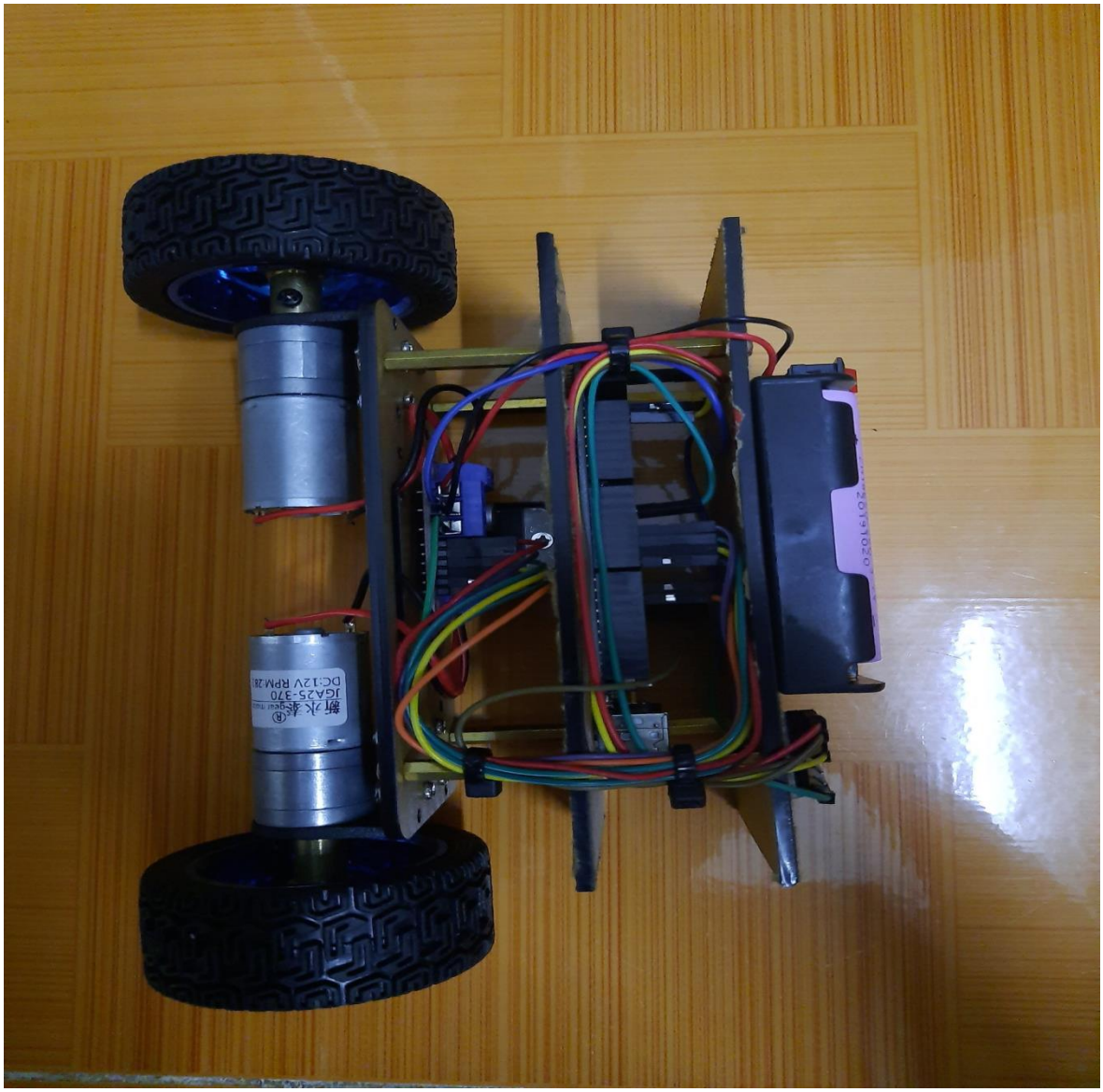
3.2.2 Lưu đồ giải thuật điều khiển

Dựa trên nền tảng của thuật toán điều khiển PID, giải thuật cân bằng và điều khiển ta có lưu đồ giải thuật như sau:



CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM

4.1 Ảnh chụp thực tế sản phẩm mô hình xe hai bánh tự cân bằng



Hình 4. 1 Ảnh chụp sản phẩm

CHƯƠNG 5: KẾT LUẬN VÀ ĐỊNH HƯỚNG ĐỀ TÀI

5.1 Kết quả đạt được

- Thiết kế hoàn chỉnh, hoàn thiện mô hình robot 2 bánh tự cân bằng
- Thiết lập mô hình toán học, hàm trạng thái của mô hình và mô phỏng thành công mô hình trên Matlab Simulink.
- Các thành phần mạch điện và vi điều khiển hoạt động tốt:
  - Giá trị cảm biến đọc về chính xác, không bị trễ , sai sót, mất frame truyền.
  - Mạch cầu H hoạt động ổn định không bị quá tải, quá nhiệt.
  - Vi điều khiển hoạt động ổn định, không tự reset hoặc ngắt kết nối
- Mô hình có thể giữ thăng bằng tại chỗ.

5.2 Hạn chế và cần khắc phục

Giữ cân bằng chưa tốt, dễ bị vọt lố. Chịu tác động của ngoại lực còn yếu. Hệ thống chưa đáp ứng được khi hệ số ma sát và khối lượng M của robot thay đổi. Chỉ có thể giữ thăng bằng tại chỗ mà chưa di chuyển được, Thời gian đáp ứng góc nghiêng thân rô bốt của hệ thống còn chậm, Robot vẫn còn rung lắc, Tín hiệu điều khiển động cơ chưa được tốt.

5.3 Hướng phát triển của đề tài

Trong tương lai mô hình robot 2 bánh tự cân bằng sẽ được phát triển như sau:

- Mô hình sẽ được điều khiển thông qua máy tính nhờ kết nối bluetooth và giaodiện trên máy tính sẽ thân thiện trực quan hơn.
- Tối ưu hóa khối lượng, thuật toán để mô hình có thể hoạt động trơn tru, linh hoạt, dễ điều khiển, tiết kiệm năng lượng.
- Có thể gắn thêm camera và gps để robot có thể định vị, ghi hình, xử lý ảnh và tự hoạt động trong không gian lớn.

PHỤ LỤC

Code chương trình

```
#include "PID_v1.h"
#include "LMotorController.h"
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
 #include "Wire.h"
#endif
#define MIN_ABS_SPEED 30
//int estado = 'h';
//int led = 4;
MPU6050 mpu;
// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector
```

```
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity vector
//PID
double originalSetpoint = 172.50;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
//adjust these values to fit your own design
double Kp = 50;
double Kd = 2.2;
double Ki = 290;
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
double motorSpeedFactorLeft = 0.6;
double motorSpeedFactorRight = 0.5;
//MOTOR CONTROLLER
int ENA = 9;
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4, motorSpeedFactorLeft, motorSpeedFactorRight);
volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin has gone high
```

```
void dmpDataReady()
{ mpuInterrupt = true;}
void setup()
{ pinMode(ENA, OUTPUT);/// out
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENB, OUTPUT);
//  pinMode(led, OUTPUT);
  //digitalWrite(led,0);
  Serial.begin(9600);
 // join I2C bus (I2Cdev library doesn't do this automatically)
 #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
 Wire.begin();
 TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
 #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
 Fastwire::setup(400, true);
 #endif
 mpu.initialize();
 devStatus = mpu.dmpInitialize();
 // supply your own gyro offsets here, scaled for min sensitivity
 mpu.setXGyroOffset(220);
 mpu.setYGyroOffset(76);
```

```
mpu.setZGyroOffset(-85);
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
// make sure it worked (returns 0 if so)
if (devStatus == 0)
{ // turn on the DMP, now that it's ready
mpu.setDMPEnabled(true);
// enable Arduino interrupt detection
attachInterrupt(0, dmpDataReady, RISING);
mpuIntStatus = mpu.getIntStatus();
// set our DMP Ready flag so the main loop() function knows it's okay to use it
dmpReady = true;
// get expected DMP packet size for later comparison
packetSize = mpu.dmpGetFIFOPacketSize();
//setup PID
pid.SetMode(AUTOMATIC);
pid.SetSampleTime(10);
pid.SetOutputLimits(-255, 255); }
else
{ // ERROR!
// 1 = initial memory load failed
// 2 = DMP configuration updates failed
// (if it's going to break, usually the code will be 1)
Serial.print(F("DMP Initialization failed (code "));
Serial.print(devStatus);
```

```
 Serial.println(F(")")); }}
void loop()
{ // if programming failed, don't try to do anything
 if (!dmpReady) return;
// wait for MPU interrupt or extra packet(s) available
 while (!mpuInterrupt && fifoCount < packetSize)
 { //no mpu data - performing PID calculations and output to motors
 pid.Compute();
 motorController.move(output, MIN_ABS_SPEED); }
 // reset interrupt flag and get INT_STATUS byte
 mpuInterrupt = false;
 mpuIntStatus = mpu.getIntStatus();
 // get current FIFO count
 fifoCount = mpu.getFIFOCount();
 // check for overflow (this should never happen unless our code is too inefficient)
 if ((mpuIntStatus & 0x10) || fifoCount == 1024)
 { // reset so we can continue cleanly
 mpu.resetFIFO();
 Serial.println(F("FIFO overflow!"));
 // otherwise, check for DMP data ready interrupt (this should happen frequently) }
 else if (mpuIntStatus & 0x02)
 {// wait for correct available data length, should be a VERY short wait
 while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
 // read a packet from FIFO
```

```
mpu.getFIFOBytes(fifoBuffer, packetSize);
 // track FIFO count here in case there is > 1 packet available
// (this lets us immediately read more without waiting for an interrupt)
fifoCount -= packetSize;
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
input = ypr[1] * 180/M_PI + 180;}}
```

Giới thiệu phần mềm sử dụng: Matlab 2018a, Arduino IDE 1.8.9.

TÀI LIỆU THAM KHẢO

[1] Tài liệu trong nước:

- Huỳnh Thái Hoàng, Lý thuyết điều khiển tự động, Nhà xuất bản Đại học Quốc Gia,2005
- Huỳnh Thái Hoàng; "Mô Hình Hóa Và Nhận Dạng Hệ Thống"; Đại học bách khoa Tp. HCM.
- Nguyễn Phùng Quang; "Matlab và Simulink"; NXB khoa học và kỹ thuật.
- Dương Hoài Nghĩa; "Điều khiển hệ thống đa biến"; Nhà xuất bản Đại học Quốc Gia TP.HCM, 2007.
- Huỳnh Thái Hoàng; "Mô Hình Hóa Và Nhận Dạng Hệ Thống"; Đại học bách khoa Tp. HCM.

[2] Tài liệu nước ngoài:

- Greg WELCH and Gary BISHOP , *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, 2004

[3] Tài liệu website:

- http://www.segway.com
- http://sourceforge.net
- http://www.ti.com
- http://www.mathworks.com/matlabcentral/answers/