ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN- ĐIỆN TỬ
BỘ MÔN TỰ ĐỘNG

BÁO CÁO ĐỒ ÁN 1

THIẾT KẾ BỘ ĐIIỀU KHIỂN PID SỐ VÀ GUI ĐIỀU KHIỂN CHO ĐỘNG CƠ SERVO

Giảng viên hướng dẫn: BÙI THANH HUYỀN

Sinh viên thực hiện: Nguyễn Hùng Phong

Mã số sinh viên: 2014123 – HK 231

Thành phố Hồ Chí Minh - 2023

LỜI CẢM ƠN

Kính gửi đến cô Bùi Thanh Huyền lời cảm ơn chân thành sâu sắc, cảm ơn cô đã tận tình giúp đỡ, định hướng và chỉ dạy em trong suốt quá trình thực hiện Đồ án này. Nhờ sự giảng dạy của cô, em đã hoàn thành Đồ án cũng như nắm bắt cách thực hiện đề tài. Và đây sẽ là hành trang quý giá cho con đường phát triển sau này của em.

Em cũng xin chân thành cảm ơn các quý thầy cô của trường Trường Đại học Bách khoa - Đại học Quốc gia Thành phố Hồ Chí Minh nói chung, của khoa Điện và bộ môn Điều Khiển Tự Động nói riêng và còn có cả các anh chị sinh viên khóa trước đã tận tình chỉ bảo, trang bị cho em những kiến thức bổ ích, quý báu.

Em đã cố gắng trình bày rõ ràng các kiến thức tìm hiểu, nghiên cứu trong đồ án. Song không thể tránh khỏi những sai sót và hạn chế. Em rất mong nhận được những ý kiến đóng góp, chỉ bảo từ quý thầy cô và các bạn.

Tp. Hồ Chí Minh, ngày 4 tháng 1 năm 2024

Sinh viên

Nguyễn Hùng Phong

TÓM TẮT ĐỒ ÁN

Nhiệm vụ của đề tài là sử dụng giải thuật PID và ứng dụng vi điều khiển để điều chỉnh, ổn định tốc độ và vị trí động cơ DC để đạt được giá trị mong muốn một cách nhanh và tối ưu nhất.

Đề tài được thực hiện như sau:

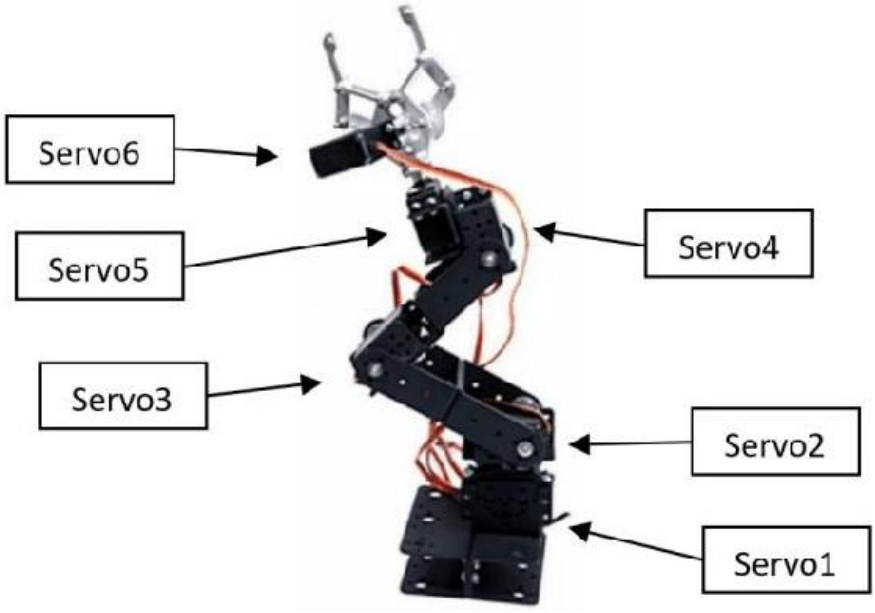
- Xây dựng bộ điều khiển PID số, dùng vi điều khiển STM32F103C8T6 làm nhân điều khiển trung tâm. Phần công suất sử dụng mạch Driver tích hợp BTS7960 để điều chỉnh động cơ đạt giá trị đặt, phần giao tiếp dùng mạch chuyển tín hiệu UART - USB CP2102 để kết nối máy tính và vi điều khiển.
- Sử dụng phần mềm Qt Creator viết bằng ngôn ngữ C++ để thiết kế giao diện giao tiếp với người dùng.
- Cụ thể hơn, Encoder quang sẽ đưa xung phản ánh tốc độ động cơ về vi điều khiển, sau đó vi điều khiển sẽ tính toán tốc độ hiện tại của động cơ và thực hiện giải thuật điều khiển PID để điều chế độ rộng xung (PWM – Pulse Width Modulation) điều khiển động cơ thông qua BTS7960.

MỤC LỤC

I. Tổng quan đề tài
1. Khái quát đề tài

Trong đời sống hiện nay, với sự phát triển khoa học càng ngày tiến bộ vượt bậc; đa số phần lớn các công việc hiện nay đều được thay thế bằng các loại máy móc tự động và hiện đại. Chiếm đa số hầu hết trong các loại máy móc như máy CNC, máy giặt, thiết bị dân dụng, máy kéo đều là các động cơ điện DC và AC. Những thiết bị như vậy đều yêu cầu về sự hoạt động chính xác cao, tiết kiệm năng lượng hết mức có thể, tuổi thọ cao. Một trong những yêu cầu cần được đáp ứng đối với những động cơ nói trên là tiêu chí về điều khiển về tốc độ và vị trí của động cơ một cách chính xác, ổn định, đáp ứng nhanh, vận hành mượt mà khi xác lập và thay đổi trạng thái.

Từ bài toán đã được đặt ra ở trên, ngay sau đó các nhà kỹ sư đã đưa ra những thuật toán kinh điển để giải quyết bài toán trên một cách nhanh nhất và đạt hiệu quả nhất như là Fuzzy Logic Control, Adaptive Control, Modle Predictive Control, Sliding Mode Control, thuật toán điều khiển PI, thuật toán điều khiển PD, thuật toán điều khiển PID. Nhưng dựa vào khảo sát đa số cho thấy, thuật toán PID là một thuật toán quen thuộc nhất, mặc dù nó đã được ra đời từ cách đây khá lâu. Hiện nay đã có ra đời nhiều giải thuật điều khiển khác được phát triển mạnh mẽ, có độ chính xác cao hơn rất nhiều với PID, mặc dù vậy thuật toán PID này vẫn được ứng dụng khá rộng rãi và nghiên cứu phát triển vì tính đơn giản, dễ áp dụng và cũng là cơ sở để phát triển nhiều giải thuật khác.



Hình 1. Ứng dụng của điều khiển động cơ trong lĩnh vực cánh tay robot

2. Nhiệm vụ đề tài

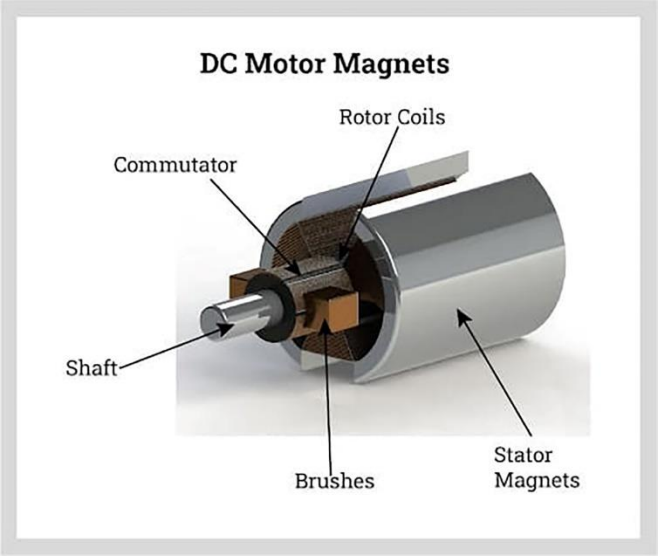
Từ sự tổng quan ở trên, từ đó em rút ra được đề tài này đặt ra những yêu cầu cơ bản như lý thuyết về mô hình hóa đối tượng tuyến tính, giải thuật điều khiển vòng kín PID, kiến thức về các driver lái động cơ, mạch nguồn, tiếp theo là đến kỹ năng sử dụng vi điều khiển để tính toán các thông số và điều khiển động cơ theo mong muốn của người dùng. Cuối cùng là một phần không thể thiếu đó là khi chúng ta muốn động cơ chạy với tốc độ và vị trí khác thì mình và người dùng không thể cứ mỗi lần cài đặt setpoint khác hoặc là thay đổi thông số của bộ điều khiển qua code được; từ đó để giải quyết được vấn đề này chúng ta sẽ thiết kế GUI (Graphical User Interface) giúp người dùng dễ dàng làm được các điều trên và dễ dàng sử dụng.

Do đó trong phạm vi đồ án này của em, em xin trình bày về thiết kế bộ điều khiển PID, xây dựng mô hình phần cứng để điều khiển động cơ qua vi điều khiển STMF103C8T6, sau đó là xây dựng GUI để điều khiển động cơ qua Qt C++, cuối cùng là ghi nhận các kết quả thu được và hướng phát triển đề tài.

II. Cơ sở lý thuyết

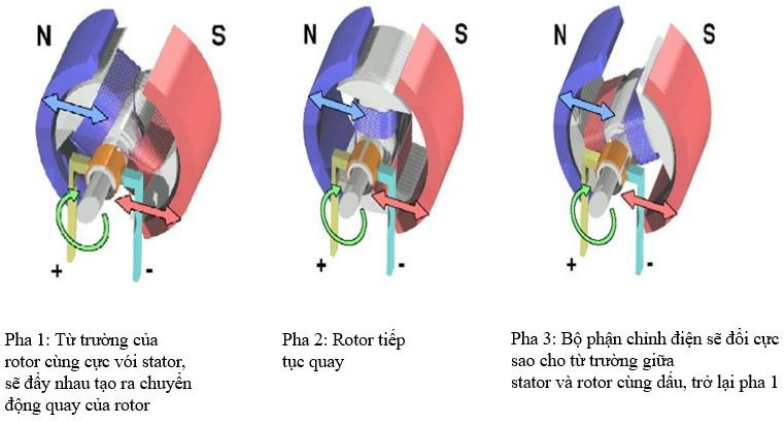
1. Đối tượng động cơ DC

1.1. Cấu tạo và nguyên lý hoạt động



Hình 2: Cấu tạo động cơ DC

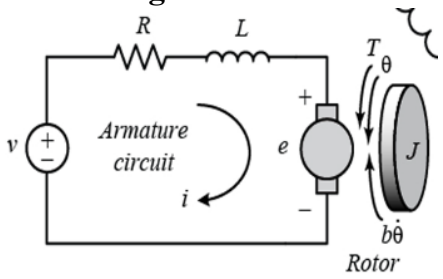
Dựa vào hình trên, cấu tạo chính của động cơ DC gồm stator, rotor và bộ chổi than cổ góp. Trong đó stator là phần cảm, cấu tạo bằng nam châm vĩnh cửu hoặc cuộn dây kích từ. Rotor là phần cứng, gồm cuộn dây quấn quanh mạch từ thép kĩ thuật điện. Bộ chổi than cổ góp gồm hai xấp lá các bon graphite thay phiên nhau quét trên hai vành bán khuyên nối với nguồn điện, tạo ra từ trường thay đổi giúp động cơ quay.



Hình 3. Hoạt động của động cơ DC

Động cơ DC lấy năng lượng điện thông qua dòng điện trực tiếp và chuyển đổi năng lượng điện này thành năng lượng Moment làm quay trục động cơ. Khi động cơ DC lấy năng lượng điện sẽ tạo ra một từ trường trong Stator. Từ trường này sẽ hút và đẩy cặp nam châm trên Rotor, làm cho Roto quay.

1.2. Mô hình hóa động cơ DC từ trường vĩnh cửu



Hình 4: Mô hình tương đương của động cơ DC

Các thông số chính của mạch trên bao gồm:

+ R: điện trở phần ứng

+ L: điện cảm phần ứng

+ J: moment quán tính trên trục động cơ

+ b: hệ số ma sát nhớt

+ K_t: hằng số moment xoắn

+ K_e: hằng số sức phản điện

+ v: điện áp nguồn

+ θ: vị trí của động cơ

+ $\dot{\theta}$: tốc độ của động cơ

+ T_m là moment xoắn trên trục của động cơ: $T_m = K_t$.i (với i là dòng điện nguồn cấp)

+ e là sức điện động trên động cơ: e = $K_e i$

Theo định luật II Newton ta có: J$\dot{\theta}$ + b$\dot{\theta}$ = K_t.i (1)

Theo định luật Kirchhoff: L$\frac{di}{dt} + Ri = U - K_e \dot{\theta}$ (2)

Biến đổi Laplace phương trình (1) ta có:

$$s(Js + b)\theta(s) = K_t.I(s) \quad (3)$$

Biến đổi Laplace phương trình (2) ta có:

$$(Ls + R)I(s) = U(s) - K_e.s.\theta(s) \quad (4)$$

Từ phương trình (3) và (4) ta suy ra được hàm truyền của vị trí và tốc độ của động cơ:

$$\frac{\theta(s)}{U(s)} = \left(\frac{1}{S}.\frac{K_t}{RJs+(bR+K_eK_t)}\right) = \frac{1}{S}.\frac{K}{\tau s+1} \quad \text{(vị trí)}$$

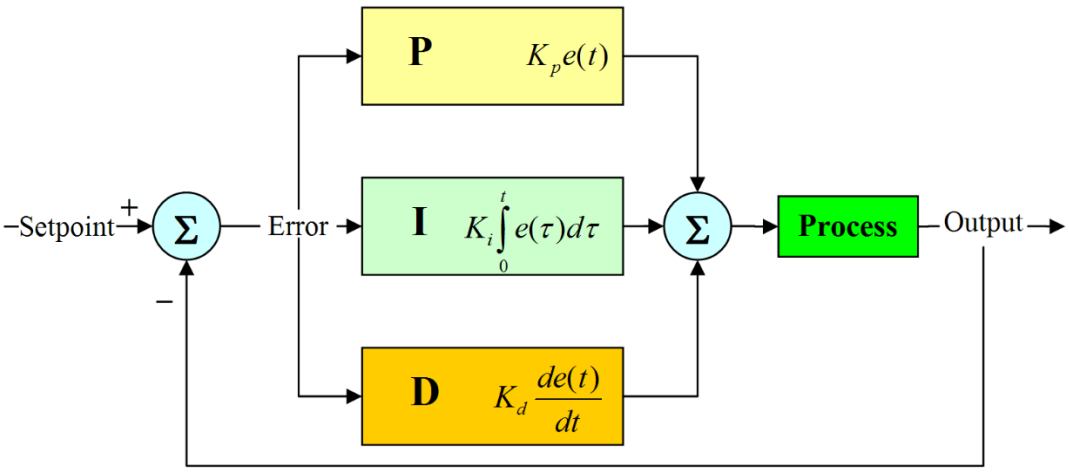
$$\frac{\dot{\theta}(s)}{U(s)} = \left(\frac{K_t}{RJs+(bR+K_eK_t)}\right) = \frac{K}{\tau s+1} \quad \text{(tốc độ)}$$

Với K là độ lợi: K = $\frac{K_t}{bR+K_eK_t}$

Với τ là thời hằng: $\tau = \frac{JR}{bR+K_eK_t}$

2. Giải thuật PID

Một bộ điều khiển vi tích phân tỉ lệ (PID - Proportional Integral Derivative) là một cơ chế phản hồi vòng điều khiển tổng quát được sử dụng rộng rãi trong các hệ thống điều khiển công nghiệp – bộ điều khiển PID là bộ điều khiển được sử dụng nhiều nhất trong các bộ điều khiển phản hồi. Bộ điều khiển PID sẽ tính toán giá trị "sai số" là hiệu số giữa giá trị đo thông số biến đổi và giá trị đặt mong muốn. Bộ điều khiển sẽ thực hiện giảm tối đa sai số bằng cách điều chỉnh giá trị điều khiển đầu vào. Trong trường hợp không có hoặc rất khó để tìm mô hình toán học về hệ thống điều khiển thì bộ điều khiển PID là sẽ bộ điều khiển tốt nhất. Tuy nhiên, để đạt được kết quả tốt nhất, các thông số PID sử dụng trong tính toán phải điều chỉnh theo tính chất của hệ thống trong khi kiểu điều khiển là giống nhau, các thông số phải phụ thuộc vào đặc thù của hệ thống.



Hình 5: Sơ đồ khối bộ điều khiển PID

Công thức cho bộ điều khiển PID là:

$$u(t) = K_P.e(t) + K_I \int_0^t e(\tau)d\tau + K_D.\frac{de(t)}{dt}$$

Trong đó;

+ u là tín hiệu điều khiển

+ e là sai lệch giữa giá trị đặt và giá trị thực tế thu được

Dựa vào sơ đồ ở hình 5: bộ PID sẽ gồm 3 thành phần: khâu tỉ lệ, khâu tích phân và khâu vi phân.

Hàm truyền của bộ điều khiển PID:

$$G_{PID}(s) = K_P + \frac{K_I}{s} + K_D s$$

Với các tham số của bộ điều khiển là K_P, K_I, K_D.

Để điều chỉnh đáp ứng ngõ ra đạt chất lượng mong muốn, ta điều chỉnh các hệ số K_P, K_I, K_D . Mỗi hệ số sẽ tác động đến chất lượng hệ thống theo một chiều hướng nhất định.

2.1. Khâu tỉ lệ (P)

Công thức: $u(t) = K_p e(t)$

Tác động của thành phần tích đơn giản là tín hiệu điều khiển tỉ lệ tuyến tính với sai lệch điều khiển. Ban đầu, khi sai lệch lớn thì tín hiệu điều khiển sẽ lớn. Sai lệch giảm dần thì tín hiệu điều khiển cũng giảm dần. Khi sai lệch e(t) = 0 thì u(t) = 0. Một vấn đề là khi sai lệch đổi dấu thì tín hiệu cũng sẽ đổi dấu.

Thành phần P có ưu điểm là tác động nhanh và đơn giản. Hệ số tỉ lệ Kp càng lớn thì thời gian lên và thời gian xác lập càng giảm, do đó thành phần P có vai trò lớn trong giai đoạn đầu của quá trình quá độ.

Tuy nhiên, khi hệ số tỉ lệ Kp càng lớn thì sự thay đổi tín hiệu điều khiển sẽ càng mạnh dẫn đến dao động lớn, đồng thời làm hệ nhạy cảm hơn với nhiễu đo vì nhiễu cũng sẽ bị khuếch đại theo hệ số Kp.

2.2. Khâu tích phân (I)

Công thức: $u(t) = K_I \int_0^t e(\tau) d\tau$

Là tích phân của sai lệch theo thời gian lấy mẫu. Điều khiển tích phân là phương pháp điều chỉnh để tạo ra các tín hiệu điều chỉnh sao cho độ sai lệch giảm về 0. Từ đó cho ta biết tổng sai số tức thời theo thời gian hay sai số tích lũy trong quá khứ. Khi thời gian càng nhỏ thể hiện tác động điều chỉnh tích phân càng mạnh, tương ứng với độ lệch càng nhỏ.

Nhược điểm của thành phần tích phân là do phải mất một khoảng thời gian đợi e(t) về 0 nên đặc tính tác động của bộ điều khiển sẽ chậm hơn xác lập hơn. Ngoài ra, thành phần tích phân đôi khi còn làm xấu đi đặc tính động học của hệ thống, thậm chí có thể làm mất ổn định nếu hệ số Ki quá lớn, hệ sẽ liên tục dao động đảo chiều liên tục trong khoảng thời gian ngắn.

Người ta thường sử dụng bộ PI hoặc PID thay vì chỉ bộ I đơn thuần vừa để cải thiện tốc độ đáp ứng, vừa đảm bảo yêu cầu động học.

2.3. Khâu vi phân (D)

Công thức: $u(t) = \frac{de(t)}{dt}$

Là vi phân của sai lệch. Điều khiển vi phân tạo ra tín hiệu điều chỉnh sao cho tỉ lệ với tốc độ thay đổi sai lệch đầu vào. Thời gian càng lớn thì phạm vi điều chỉnh vi phân càng mạnh, tương ứng với bộ điều chỉnh đáp ứng với thay đổi đầu vào càng nhanh. Khâu vi phân đóng vai trò dư đoán đầu ra của quá trình và đưa ra phản ứng thích hợp dựa trên chiều hướng và tốc độ thay đổi của sai lệch e(t), làm tăng tốc độ đáp ứng của hệ.

Nhược điểm của khâu vi phân là rất nhạy cảm với nhiễu đo hay của giá trị đặt do tính

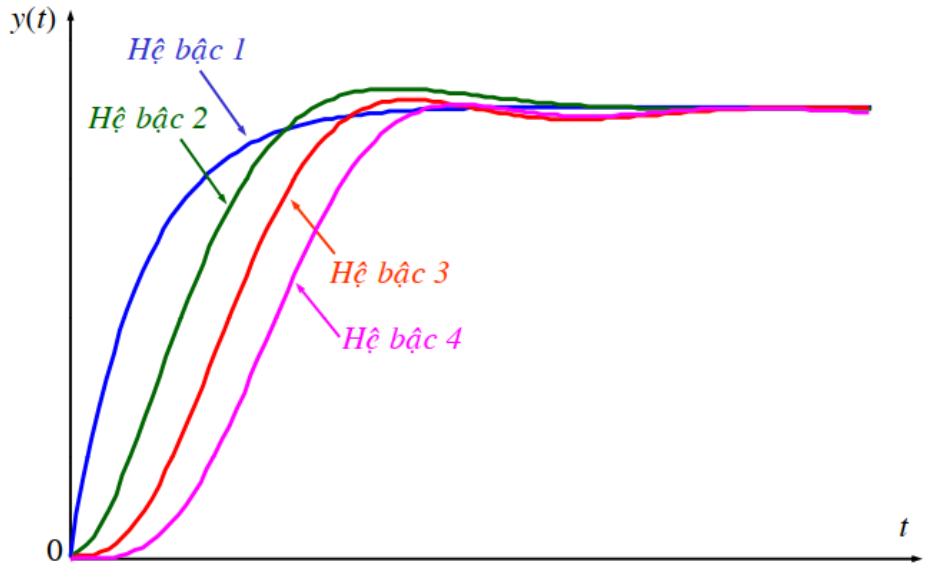
đáp ứng nhanh của nó.

2.4. Tác động của từng khâu lên chất lượng của hệ thống

	Thời gian lên	Vọt lố	Thời gian quá độ	Sai số xác lập
K_P	Giảm	Tăng	Thay đổi nhỏ	Giảm
K_I	Giảm	Tăng khi Ki quá lớn	Tăng	Loại bỏ
K_D	Thay đổi nhỏ	Giảm	Giảm	Không tác động

Dựa vào bảng trên thì để có thể đưa ra một hệ thống điều khiển tốt thì lúc này chúng ta cần phải chọn các thông số K_P, K_I, K_D sao cho có độ vọt lố $< 10\%$, thời gian quá độ và thời gian lên ngắn, sai số xác lập phải gần bằng 0 càng tốt.

Dưới đây là tiêu chuẩn ITAE về chất lượng quá độ chúng ta có thể tham khảo để chọn các thông số K_P, K_I, K_D cho phù hợp.



Hình 6: Đồ thị đáp ứng theo tiêu chuẩn ITAE

3. Phương pháp điều chế độ rộng xung PWM (Pulse Width Modulation)

Chúng ta điều khiển động cơ theo mong muốn bằng cách chúng ta sẽ thay đổi điện áp đặt vào hai đầu của động cơ. Ở trường hợp này em sẽ lựa chọn phương pháp PWM để thay đổi điện áp đặt vào hai đầu động cơ.

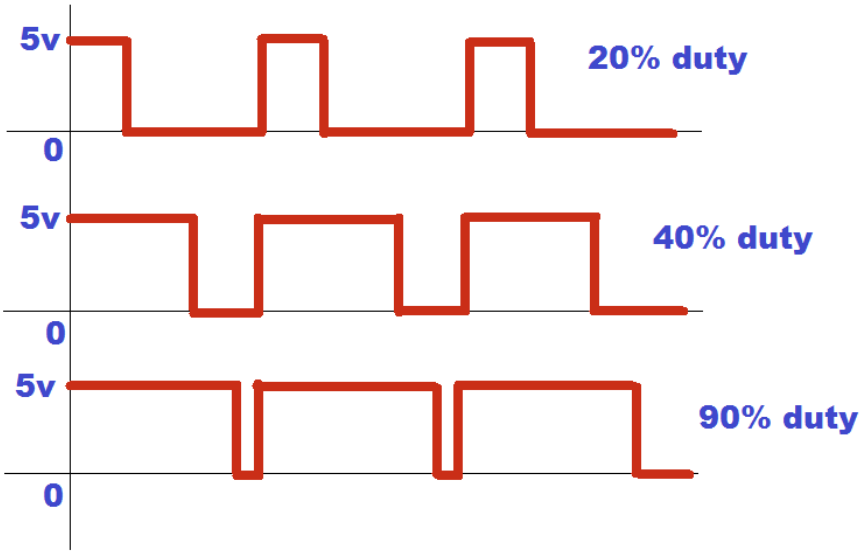
PWM (Pulse Width Modulation) là phương pháp điều chế độ rộng xung được hiểu là

cách điều khiển điện áp tải ra, dựa trên sự thay đổi độ rộng xung vuông, dẫn đến sự thay đổi điện áp.

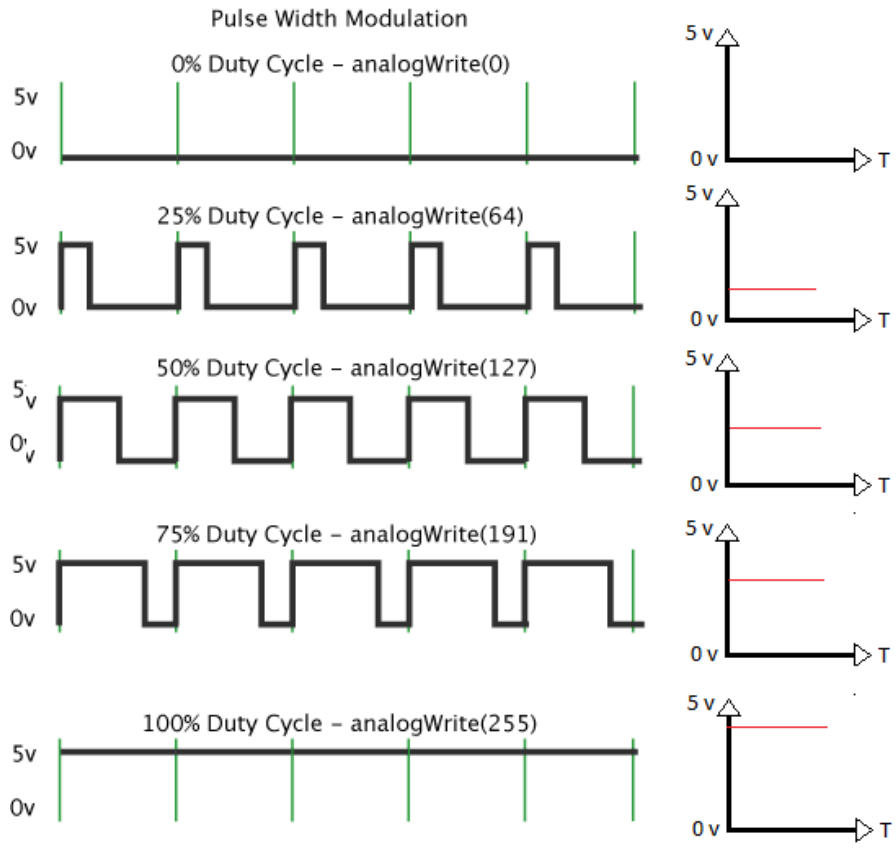
Hiểu theo cách khác, PWM là tín hiệu được tạo ra từ vi mạch kỹ thuật số, như vi mạch điều khiển. Tín hiệu được tạo ra sẽ là một nhóm xung ở dạng xung sóng vuông, có nghĩa là ở bất cứ thời điểm nào sóng sẽ cao hoặc thấp.

Điều chế độ rộng xung được xem là một kỹ thuật điều khiển dòng tiện lợi, cho phép người vận hành kiểm soát tốc độ của động cơ, hiệu suất hoạt động, tiết kiệm năng lượng…

Từ hàm truyền, ta thấy muốn thay đổi tốc độ hay vị trí động cơ, ta phải thay đổi điện áp cấp. Tuy nhiên việc thay đổi trực tiếp điện áp một chiều với tần số cao là khá khó khăn và dễ hỏng nguồn điện. Do đó ta thường dùng phương pháp điều chế độ rộng xung PWM này.



Hình 7. Xung PWM với độ rộng xung khác nhau

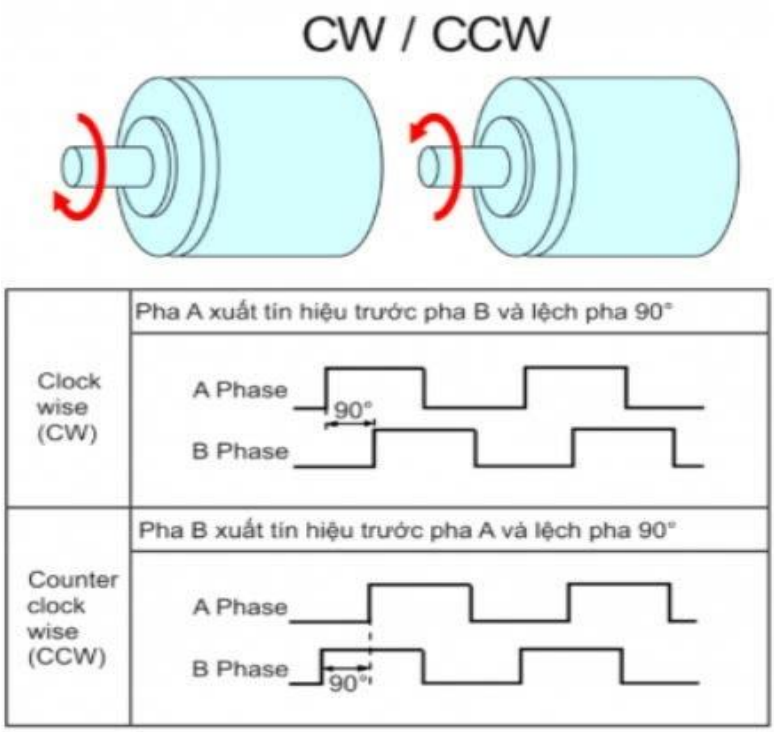


Hình 8: Ảnh hưởng thông số duty cycle của xung PWM tới điện áp đầu ra

4. Xác định vận tốc và chiều quay của động cơ

4.1. Xác định chiều quay

Dựa vào encoder ta có thể xác định được chiều quay của động cơ. Qui định xung A xuất tín hiệu trước xung B thì encoder đang quay cùng chiều kim đồng hồ CW(Clockwise), lúc này xung A đi trước và lệch pha 90 độ so với xung B. Ngược lại trục encoder quay ngược chiều kim đồng hồ CCW(Couter Clockwise), thì xung B xuất tín hiệu trước và lệch pha 90 độ xung A.



Hình 1. Xác định chiều quay động cơ

4.2. Xác định vận tốc của động cơ

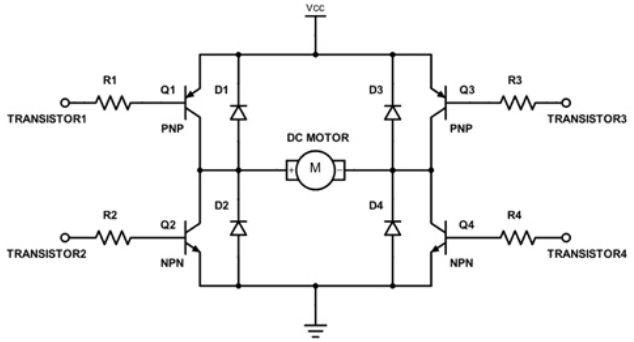
Dựa vào số xung trả về từ encoder ta có thể đếm được số xung mà động cơ DC trả về được sau thời gian T (s) là bao nhiêu. Từ đó ta có thể tính được tốc độ của động cơ như sau:

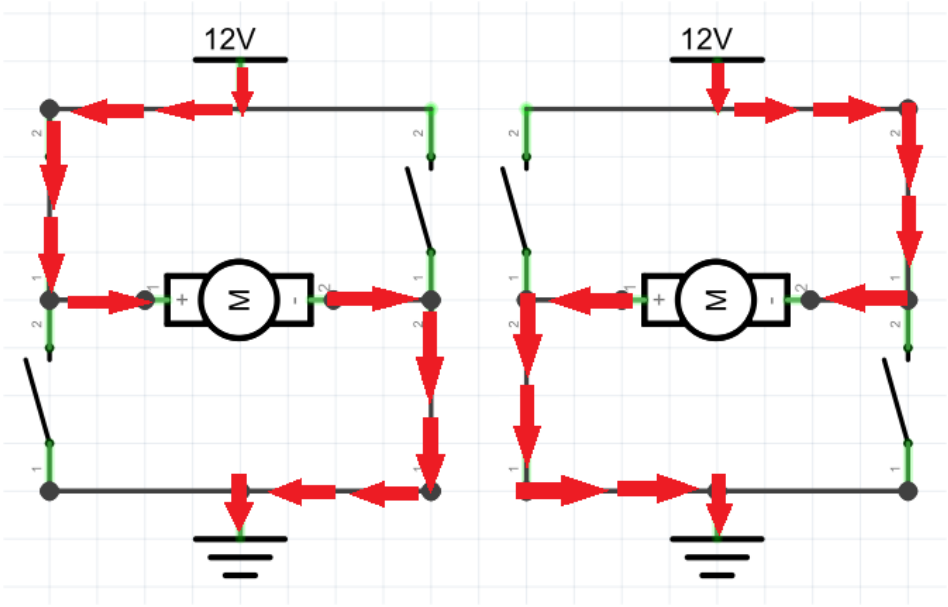
$$\text{Tốc độ động cơ} = \frac{Pluse\ quay\ \text{được} .60}{T} \text{(rpm)}$$

Với Pluse quay được là số xung mà encoder trả về sau khi động cơ DC quay được trong T (s).

5. Điều khiển động cơ theo nguyên lý mạch cầu H

Xét một cách tổng quát, mạch cầu H là một mạch gồm 4 "công tắc" được mắc theo hình chữ H. Bằng cách điều khiển 4 "công tắc" này đóng mở, ta có thể điều khiển được dòng điện qua động cơ cũng như các thiết bị điện tương tự. Bốn "công tắc" này thường là Transistor BJT, MOSFET hay relay. Tùy vào yêu cầu điều khiển khác nhau mà người ta lựa chọn các loại "công tắc" khác nhau.





Hình 11. Nguyên lý mạch cầu H

Từ đó dựa vào tần số PWM băm cho các MOSFET hoặc BJT trên mạch cầu H chúng ta có thể quyết định được điện áp đặt trên hai đầu động cơ. Ngoài ra bằng cách thay đổi thứ tự đóng mở của các MOSFET hoặc BJT chúng ta có thể đổi chiều quay của động cơ DC.

6. Thuật toán EMA Filter

Thuật toán EMA Filter còn có một tên gọi khác là thuật toán bộ lọc thông thấp. Đây là một phần kiến thức nhỏ trong phần xử lý số tín hiệu. Thuật toán này có nhiệm vụ chính là tính toán dữ liệu hiện tại phụ thuộc chủ yếu vào dữ liệu trước đó. Do bởi vì khi điều khiển động cơ hiện tượng nhiễu sẽ xảy ra bất kỳ lúc nào, nên việc tin tưởng dữ liệu vận tốc và vị trí tại thời điểm hiện tại có thể gây ra lỗi lớn. Hoặc là trong trường hợp encoder của chúng ta sử dụng có độ phân giải quá nhỏ dẫn tới không thể biểu diễn được chi tiết các

mức vận tốc vì các khoảng nhảy mỗi lần của encoder là khá lớn. Trong cả hai trường hợp vừa được nêu ở trên thì sử dụng EMA filter sẽ giảm thiểu được hậu quả của những vấn đề này.

Công thức của thuật toán này là:

The *difference equation* for an exponential moving average filter is:

$$y[i] = \alpha \cdot x[i] + (1 - \alpha) \cdot y[i - 1]$$

where:

y is the output ($[i]$ denotes the sample number)

x is the input

α is a constant which sets the cutoff frequency (a value between 0 and 1)

Với α là *hệ số cho sự tin cậy vào dữ liệu trước đó hay không*.

III. Thiết kế hệ thống

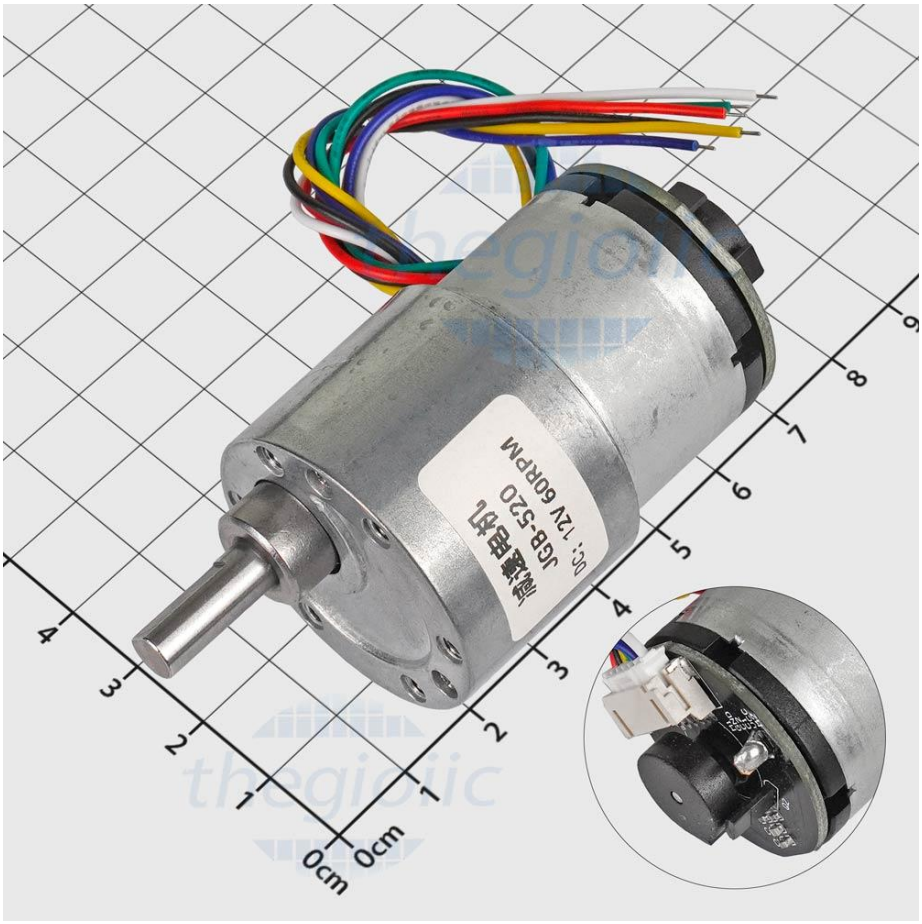
1. Thiết kế phần cứng

1.1. Chọn thiết bị cho hệ thống

1.1.1. Động cơ Servo

Trong phạm vi của đồ án, phần tử chấp hành được sử dụng là động cơ DC. Do để khảo sát được đặc tính phi tuyến của động cơ DC nên em sẽ phải chọn động cơ có công suất trong khoảng từ 30 W – 36 W.

Dựa vào những yêu cầu trên, em đã chọn động cơ này cho đồ án của mình:



Hình 12: Động cơ DC JBG37-520

Động cơ này có nhãn hiệu là JBG37-520, động cơ có tích hợp hộp số ở đầu cho chức năng giảm tốc, có gắn encoder ở sau thuận lợi cho việc thu xung về để tính toán.

Thông số chi tiết của động cơ là:

Thông số kĩ thuật	
Động cơ	
Điện áp định mức	6 -24V/ 12VDC
Dòng tối đa	3A
Công suất	36W
Series	GB37
Tốc độ không tải (sau giảm tốc)	60 rpm
Tỉ lệ Hộp số	1:168

Encoder	
Điện áp nguồn	3.3 - 5VDC
Số xung/vòng (trước giảm tốc)	11
Số xung/vòng (sau giảm tốc)	1848

1.1.2. Driver lái động cơ

Bởi vì động cơ em chọn ở mục trước có công suất là 36 W, nên là ở bước này em cần phải chọn driver có thể đáp ứng được tối thiểu công suất là 36 W cho động cơ. Ở đồ án này em quyết định chọn driver lái động cơ là module BTS7960.

Hình 13: Mạch điều khiển động cơ DC BTS7960 43A High-Power Motor Driver

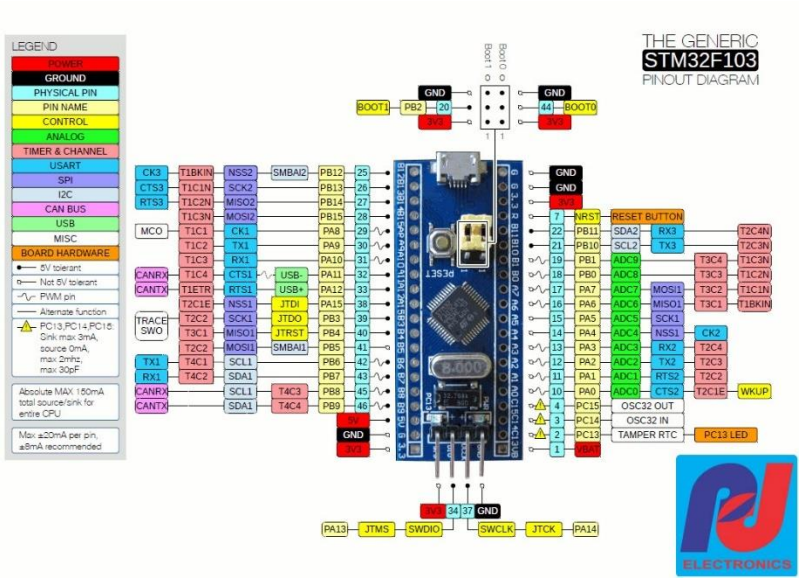
Thông số kĩ thuật của driver này:

+ Nguồn cấp: 6 ~ 27VDC
+ Dòng điện tải mach: 43A (Tải trở) hoặc 15A (Tải cảm)
+ Tín hiệu logic điều khiển: 3.3 ~ 5VDC.
+ Tần số điều khiển tối đa: 25KHz.

+ Tự động shutdown khi điện áp thấp: để tránh điều khiển động cơ ở mức điện áp thấp thiết bị sẽ tự shutdown. Nếu điện áp < 5.5VDC, driver sẽ tự ngắt điện và sẽ mở lại sau khi điện áp > 5.5VDC.
+ Bảo vệ quá nhiệt: BTS7960 bảo vệ chống quá nhiệt bằng cảm biến nhiệt tích hợp bên trong. Đầu ra sẽ bị ngắt khi có hiện tượng quá nhiệt.
+ Kích thước: 40 x 50 x12mm.

1.1.3. Vi điều khiển STM32F103C8T6

Kit phát triển STM32F103C8T6 Blue Pill ARM Cortex-M3 là loại được sử dụng để nghiên cứu về ARM nhiều nhất hiện nay do có mức giá rẻ đồng thời có nhiều chức năng áp dụng tiện lợi cho đồ án 1 như là DMA, Encoder mode của Timer. Bo mạch này được thiết kế nhỏ gọn, hoạt động vô cùng ổn định, các chân ngoại vi được đưa ra ngoài giúp dễ dàng kết nối, giao tiếp. Do đó em quyết định sử dụng vi điều khiển này cho hệ thống của mình.



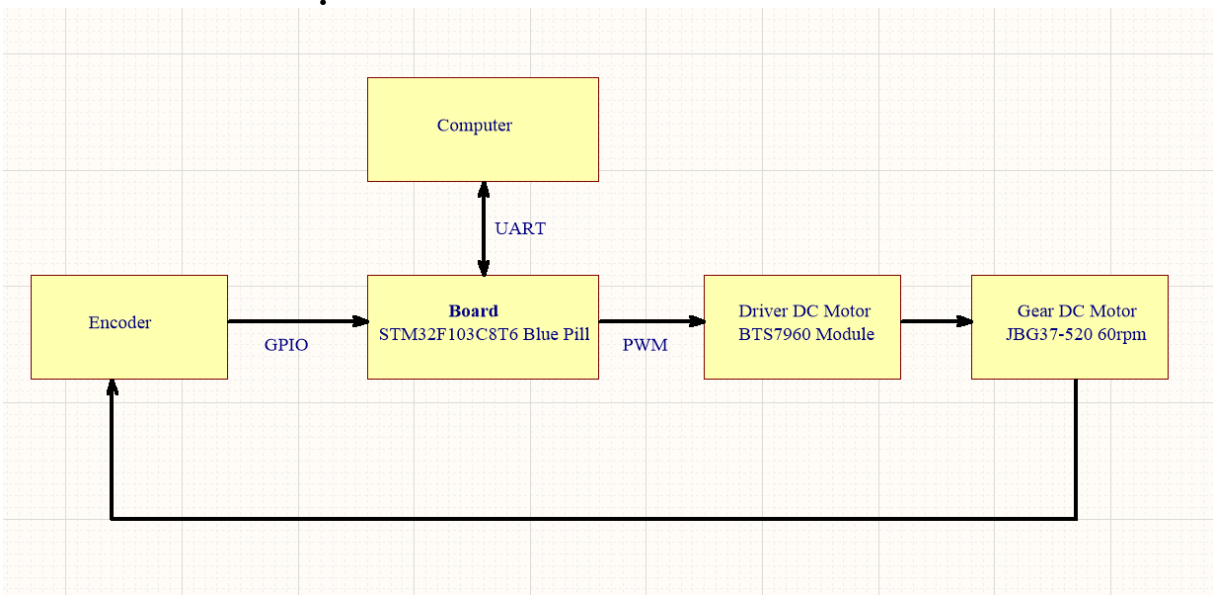
Hình 14. Kit STM32F103C8T6

Thông số kĩ thuật	
RAM	20 KB
Flash memory	64 KB
Nguồn cấp qua cổng USB	5VDC
Core	ARM 32 Cortex-M3 CPU
Thạch anh	4 -16 MHz

Tần số	72MHz
Điện áp hoạt động	2~3.6VDC
ADC	2 kênh - 12bits
DMA	7 kênh
Timer	7 bộ timer
I2C interface	2 kênh
USART interface	3 kênh
SPI interface	2 kênh
CAN interface	1 kênh
USB interface	USB 2.0 full speed

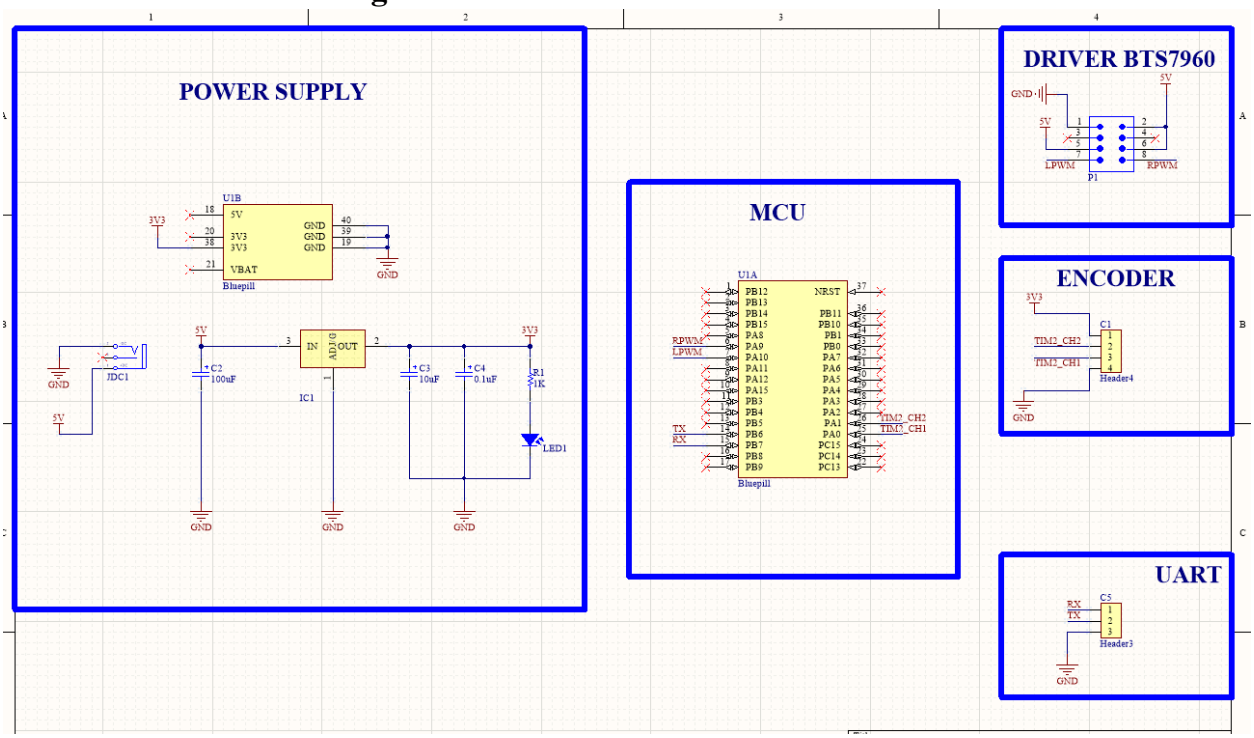
1.2. Thiết kế mạch

1.2.1. Sơ đồ khối của mạch



Hình 15: Sơ đồ khối của hệ thống

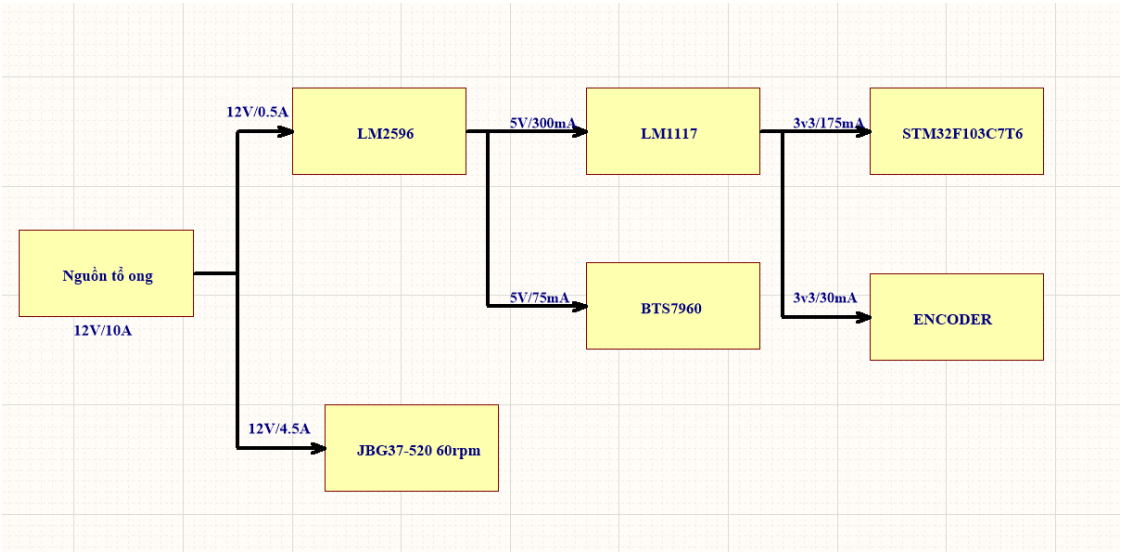
1.2.2. Chi tiết kết nối từng khối



Hình 16: Sơ đồ chi tiết đi dây

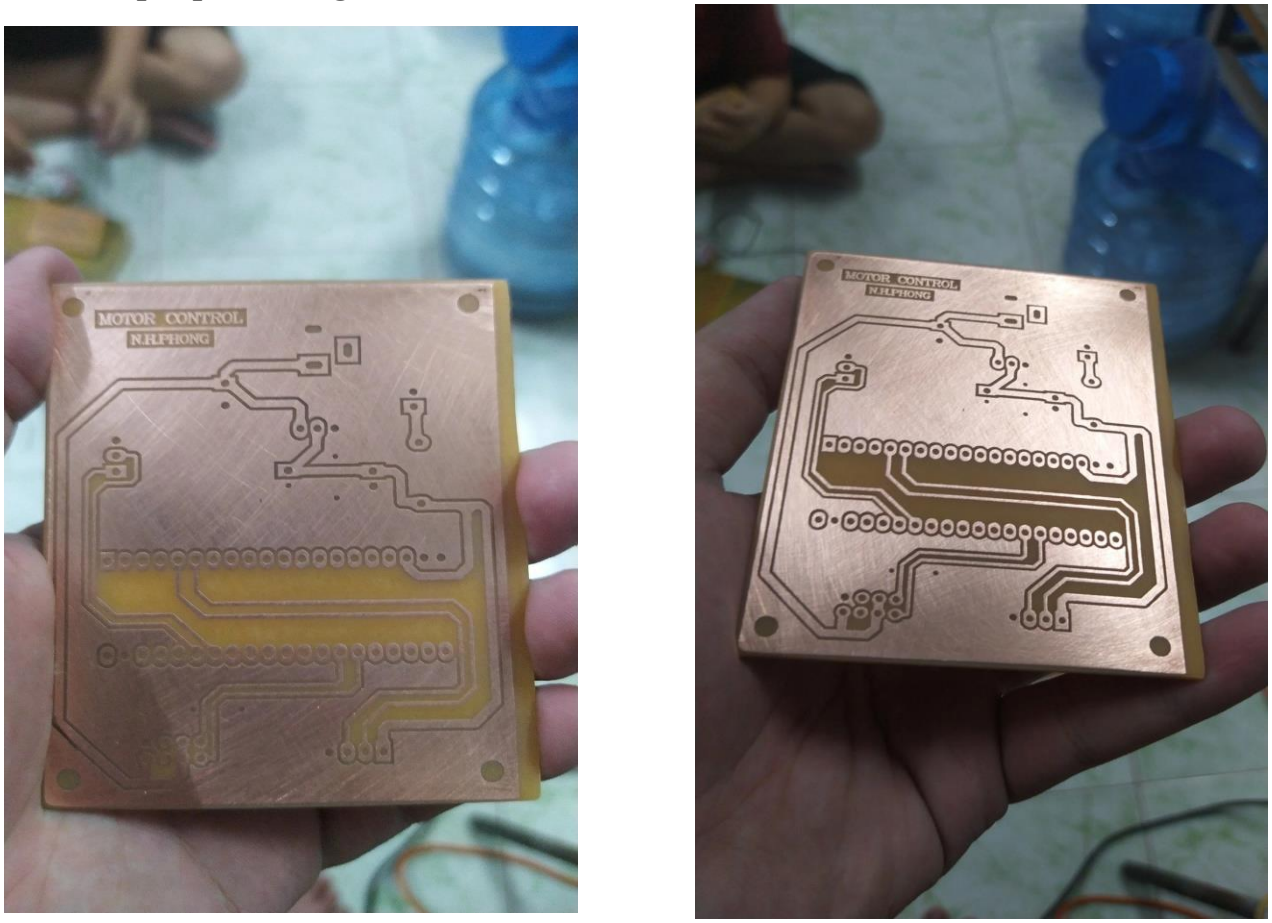
1.2.3. Sơ đồ nguồn cho hệ thống

Để có thể đáp ứng được công suất cho tất cả các thiết bị trong hệ thống hoạt động bình thường em tính toán sơ đồ công suất như sau:

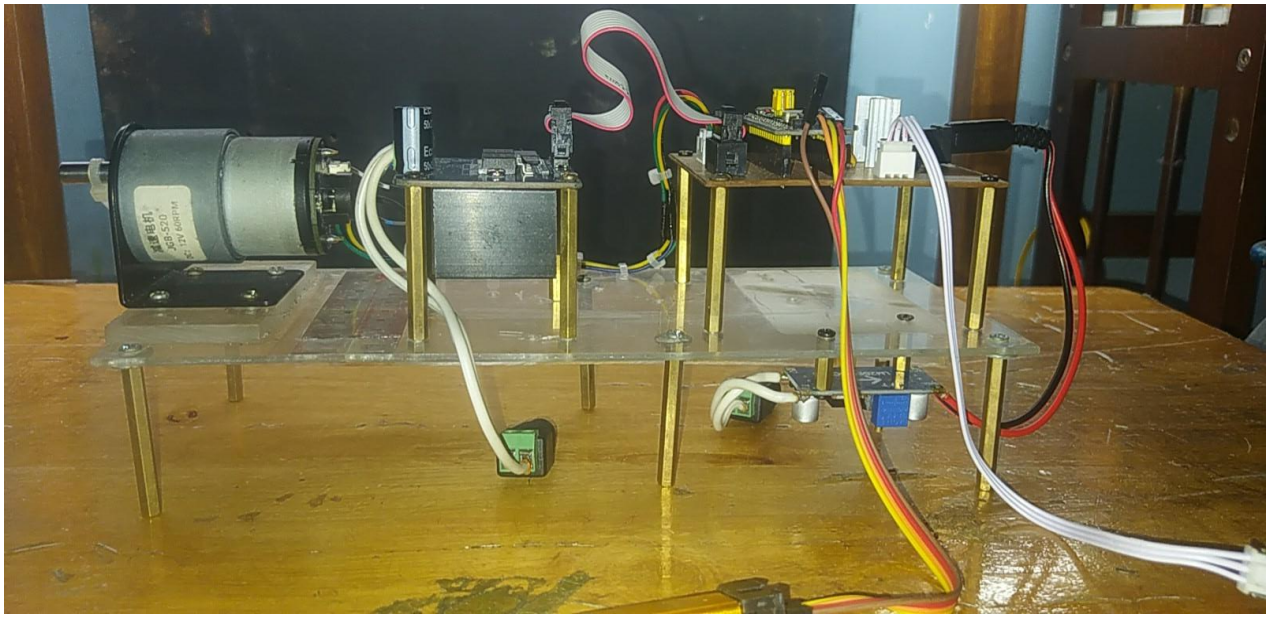


Hình 17: Sơ đồ nguồn của hệ thống

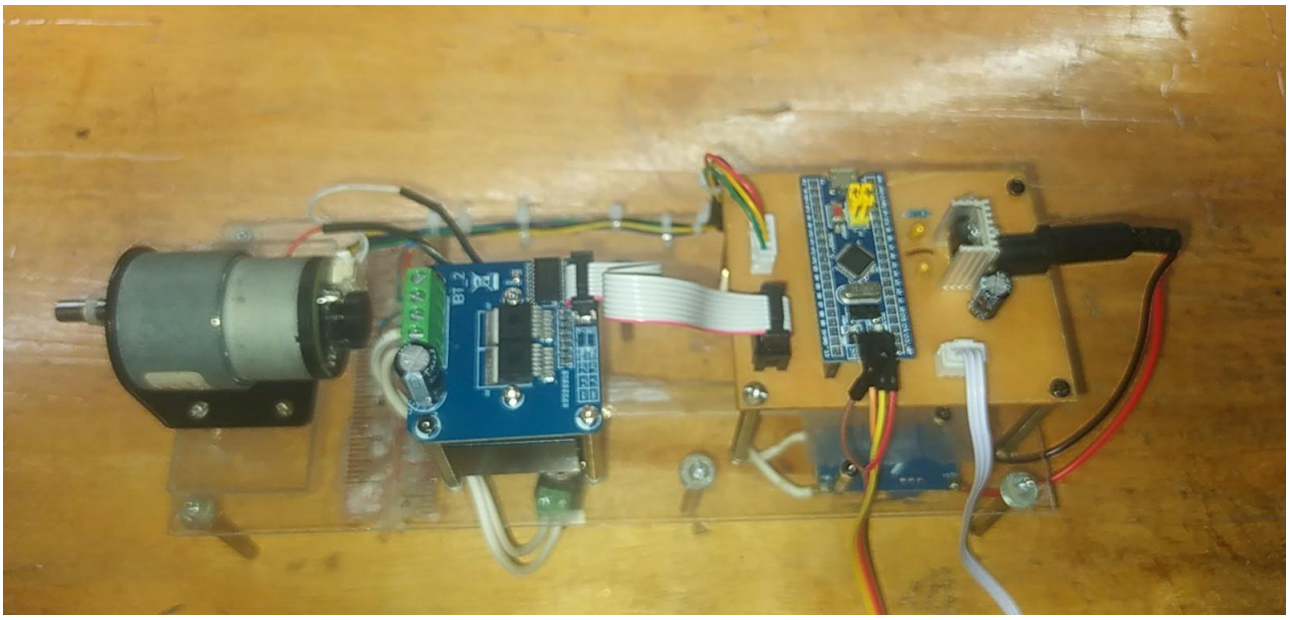
1.3. Kết quả phần cứng thu được



Hình 18: Mạch in của mạch điều khiển hệ thống



Hình 19: Sơ đồ hoàn chỉnh của hệ thống



Hình 20: Sơ đồ hệ thống hoàn chỉnh chụp từ phía trên

2. Thiết kế phần mềm

2.1. Thiết kế firmware trên vi điều khiển

Vi điều khiển đóng vai trò là bộ điều khiển trung tâm của hệ thống, tính thuật toán bộ điều khiển PID và sau đó tính toán xung PWM phù hợp để điều khiển động cơ. Để có thể lập trình cho vi điều khiển ta sử dụng phần mềm hỗ trợ STM32 CubeIDE v1.4 viết bằng thư viện HAL (Hardware Abstraction Layer). Chương trình của vi điều khiển gồm các khối chức năng sau: Read Encoder, PID Algorithm, Khối tính toán thông số Motor. Ngoài ra do vi điều khiển của chúng ta còn phải giao tiếp với GUI ở trên máy vi tính, nên chúng ta sẽ có thêm một khối UART Communication chịu trách nhiệm trao đổi data giữa GUI và vi điều khiển.

2.1.1. Khối read enconder

Khối này có chức năng đọc tín hiệu Encoder bằng mode đọc encoder được hỗ trợ sẵn trong chip sau đó lưu dữ liệu xung đọc về vào các biến của khối này.

2.1.2. Khối tính toán thông số motor

Khối này có chức năng sẽ dựa trên các dữ liệu xung đọc về ở khối read encoder chuyển đổi dữ liệu này sang vận tốc đơn vị rpm và vị trí của động cơ.

Khối này còn có chức năng xuất xung PWM để điều khiển động cơ duawh vào tham số được lấy từ đâu ra của khối PID.

2.1.3. Khối PID

Khối PID nhận tín hiệu đặt (Setpoint), các thông số bộ điều khiển từ giao diện trên máy tính thông qua khối UART Communication, đồng thời nhận tín hiệu hồi tiếp vị trí, tốc độ của khối Read Encoder, sau đó sẽ áp dụng giải thuật PID tính toán để tìm được tín hiệu điều khiển.

2.1.4. Khối UART Comunication

Khối này em sử dụng kết hợp cả hai phương thức UART và DMA lại với nhau. Bởi vì về mặt bản chất DMA có thể được coi như là một CPU thứ hai đứng sau CPU chính của vi điều khiển, nên là khi cần gửi hoặc nhận gì DMA sẽ đảm nhận công việc này, lúc đó CPU của chúng ta có thể làm được việc khác mà không phải bị treo hoặc bị ngắt quãng do CPU cứ phải đợi chờ để truyền dữ liệu hoặc nhận dữ liệu.

Khối này nhận tín hiệu đặt và thông số số bộ điều khiển từ máy tính, rồi gửi cho khối PID để tính toán tín hiệu điều khiển. Bên cạnh đó, khối UART sẽ lấy dữ kiệu từ khối đọc encoder rồi gửi lên máy tính để giao diện thực hiện vẽ đồ thị đáp ứng của động cơ theo thời gian thực, giúp ta có thể dễ dàng theo dõi, giám sát quá trình hoạt động của động cơ cũng như đánh giá chất lượng của bộ điều khiển.

Trong thực tế, khi truyền tải mỗi dữ liệu gồm một chuỗi các bit, các lỗi có thể phát sinh, bit 1 có thể biến thành bit 0 và ngược lại, có thể lỗi một bit hoặc nhiều bit. Nguyên nhân của việc này có thể là do nhiễu, chuyển đổi nguồn điện, nguồn cung cấp điện kém, … Việc nhận sai dữ liệu truyền nhận có thể làm cho hệ thống hoạt động sai, gây ra những hậu quả khó lường. Để tránh tình huống đó, ở đây em đã thêm các thông tin phụ vào bản tin chỉ nhằm mục đích giúp kiểm tra lỗi. Mã thừa sẽ được loại bỏ sau khi xác định xong độ chính xác của quá trình truyền. Phương pháp kiểm tra lỗi em dùng ở đây là phương pháp kiểm tra lỗi bằng thuật toán CRC (Cyclic Redundancy Check)

Thuật toán CRC:

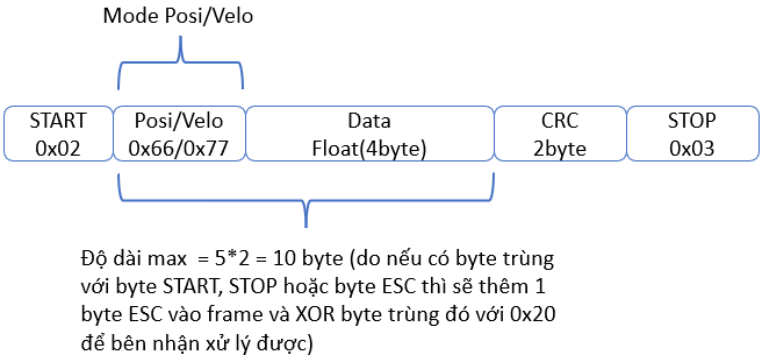
Ý tưởng cơ bản của thuật toán CRC là coi dữ liệu được truyền là một số nhị phân rất dài các chữ số. Chia số này cho một số khác, phần số dư của phép chia này được nối với dữ liệu ban đầu dưới dạng dữ liệu kiểm tra.

Thuật toán CRC sẽ tương tự với quy trình này: phép nhân và chia được sử dụng như là phép nhân và chia thông thường, trong khi đó phép cộng và phép trừ được thay bằng phép XOR

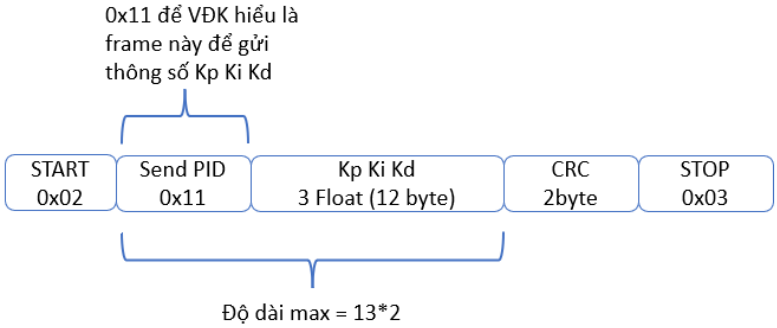
bit. Do đó phép chia sẽ được thực hiện dễ dàng hơn vì không phải quan tâm đến cộng trừ có nhớ. Tại đây em không tự tính CRC mà dùng thư viện có sẵn để tính, công việc thực hiện chỉ là code tạo frame truyền – nhận và thêm 2 byte CRC tính từ thư viện đó vào.

Các frame truyền tự tạo như sau:

+ Frame truyền giá trị của vận tốc hiện tại hoặc vị trí hiện tại từ vi điều khiển lên GUI để vẽ đồ thị:

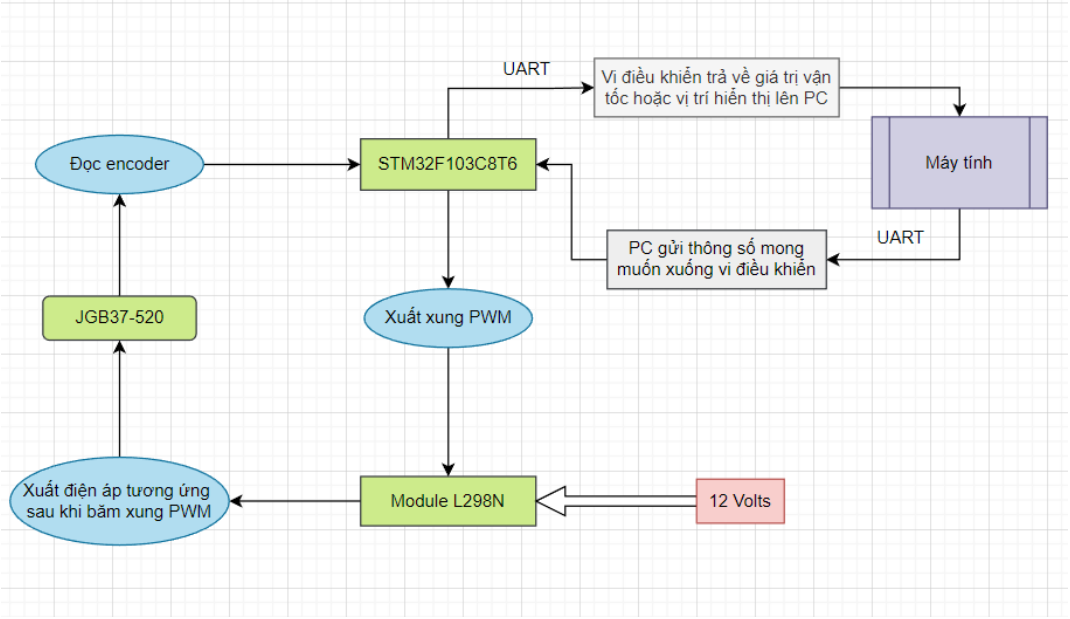


+ Frame truyền các thông số Kp Ki Kd xuống vi điều khiển:



+ Và một vài frame khác cũng tương tự với 2 frame trên các chức năng gửi tín hiệu điều khiển thay đổi setpoint, điều khiển động cơ run, stop, reset.

2.1.5. Flowchart chính của chương trình



Hình 21: Flowchart của chương trình trên vi điều khiển

2.2. Thiết kế software cho GUI trên máy tính

2.2.1. Giới thiệu về phần mềm Qt Creator:

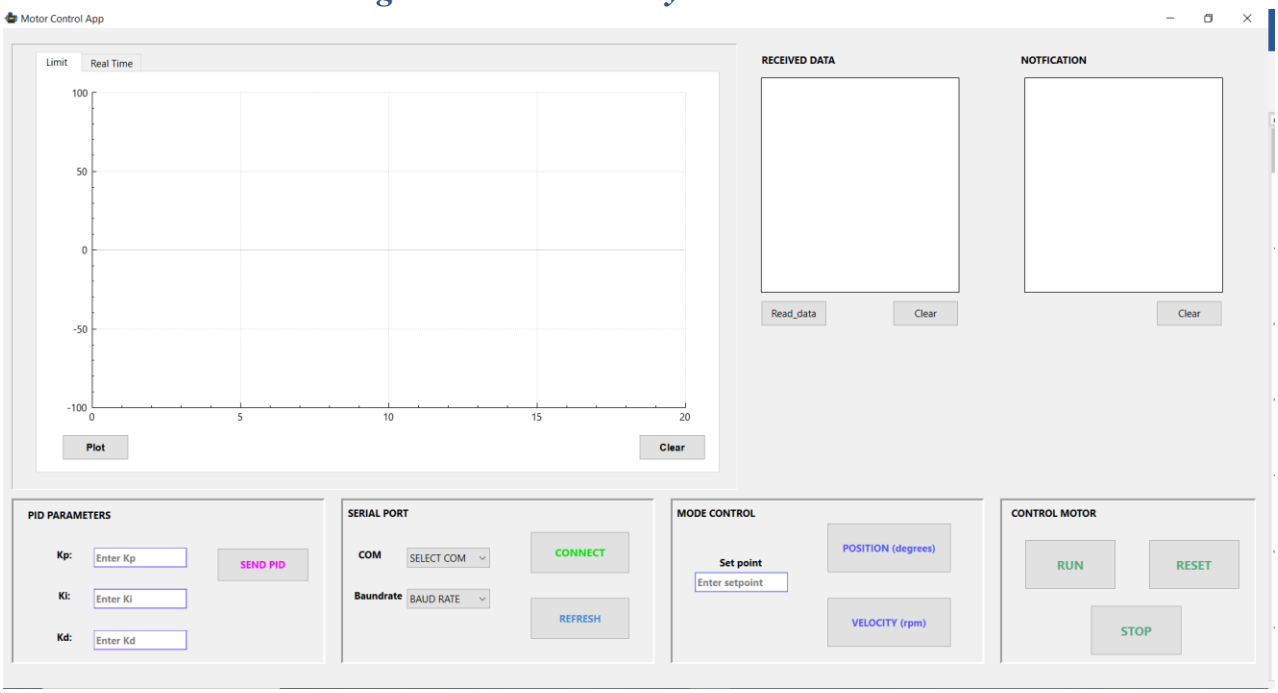
Qt là một framework đa nền tảng. Một số ứng dụng phổ biến được viết từ Qt có thể kể đến như KDE, Opera, Google Earth, và Skype. Qt lần đầu tiên được giới thiệu vào tháng 5 năm 1995. Qt có thể được dùng để phát triển ứng dụng mã nguồn mở lẫn các ứng dụng dành cho doanh nghiệp. Bộ công cụ phát triển Qt rất mạnh mẽ vì nó được cả một cộng đồng mã nguồn mở hỗ trợ. Có đến hàng ngàn các nhà phát triển mã nguồn mở sử dụng Qt trên toàn thế giới.

Qt được viết bằng C++ và được thiết kế để sử dụng trong C++. Tuy nhiên, hiện nay chúng ta đã có thể dùng thư viện này với nhiều ngôn ngữ khác như Java hay Python, …

Trên thực tế, Qt không phải một thư viện mà là 1 tập hợp các thư viện. Chúng rất rộng và thường thì người ta sử dụng thuật ngữ framework, nghĩa là 1 khối kiến trúc tập hợp cung cấp nhiều công cụ để việc lập trình của chúng ta trở nên hữu hiệu hơn.

Hình 22. Logo của phần mềm Qt Creator

2.2.2. Cấu trúc của chương trình GUI trên máy tính



Hình 23. GUI điều khiển động cơ trên máy tính

Giao diện có các khối tương tác như khối kết nối (Connect), hủy kết nối (Disconnect), chọn gửi thông số Kp Ki Kd xuống vi điều khiển (Send PID), khối để chọn chế độ điều khiển động cơ (Position hoặc Velocity), khối hiển thị dữ liệu nhận về cũng như hiển thị các thông báo của hệ thống (Data read và Command), khối hiển thị đồ thị đáp ứng của hệ thống (Graph1

và Graph2) và khối System Control gồm Run, Reset, Stop.

2.2.2.1. Khối Connect và Disconnect:

Có chức năng chọn cổng COM kết nối với USB UART đang kết nối giữa máy tính và vi điều khiển, có nút nhấn để cho phép kết nối cũng như ngắt kết nổi giữa 2 thiết bị. Khi chưa kết nối nút nhấn sẽ hiển thị chữ "CONNECT" màu chữ xanh dương và khi đã kết nối thành công thì sẽ hiển thị chữ "DISCONNECT" màu chữ đỏ để báo cho người sử dụng biết. Khi đang không kết nối, các nút nhấn trong System Control cũng như nút gửi thông số bộ điều khiển khi nhấn sẽ hiện thông báo cảnh báo.

2.2.2.2. Khối System Control

Khối này cho phép hệ thống chạy hoặc reset lại các thông số khi nhấn nút.

Khi nhấn nút "RUN", máy tính sẽ gửi xuống vi điều khiển một khung tin. Sau khi xác nhận được khung tin, vi điều khiển sẽ gửi trả một khung tin cho máy tính xác nhận động cơ đang chạy và sẽ bắt đầu khởi động bộ điều khiển và xuất xung PWM. Để có thể chạy được hệ thống phải có đầy đủ thông số PID và chọn mode điều khiển.

Khi nhấn nút "RESET", máy tính sẽ gửi xuống vi điều khiển một khung tin. Sau khi xác nhận được khung tin, vi điều khiển sẽ gửi trả một khung tin cho máy tính xác nhận đã ngừng lại động cơ, và sẽ ngắt bộ điều khiển PID và xuất xung PWM = 0 để dừng động cơ, đồng thời đưa biến đọc encoder về giá trị 0.

Khi nhấn "STOP", máy tính sẽ gửi một frame xuống vi điều khiển. Sau khi vi điều khiển giải mã frame và biết được đây là tín hiệu STOP thì vi điều khiển sẽ lập tức ngừng cấp xung PWM đang cấp cho động cơ, khiến động cơ ngừng chạy.

2.2.2.3. Khối PID Parmeters và khối đặt setpoint

Chứa các ô để nhập giá trị Kp Ki Kd gửi xuống vi điều khiển,

Các ô để nhập giá trị Kp Ki Kd, setpoint chỉ có thể nhập được số thực, đã lập trình các ô đấy không thể điền các ký tự hoặc chữ cái vào.

Sau khi kiểm tra thông số, nhấn nút "Send PID" để gửi thông số xuống vi điều khiển. Khối đặt setpoint mục đích để đặt setpoint vị trí (position, tính theo độ) hoặc setpoint vận tốc (velocity, tính theo vòng/phút)

2.2.2.4. Khối Graph

Gồm Graph1 và Graph2, Graph1 (limit_time) dùng để quan sát đồ thị đáp ứng, Graph2 (real_time) dùng để tune thông số liên tục theo thời gian mà không bị lag phần mềm Qt khi lượng dữ liệu thu được ngày càng lớn.

Khi có giá trị vị trí hoặc vận tốc được nhận trong lúc động cơ hoạt động, khối này sẽ được liên tục cập nhật đồ thị đáp ứng mức.

2.2.2.5. Khối data read và command

Khối command có nhiệm vụ hiển thị các lệnh gửi từ GUI xuống vi điều khiển, có phản hồi nếu frame nhận được bị sai, khối data read dùng hiển thị giá trị vận tốc/vị trí hiện tại của động cơ (phía sau là thời gian tại vận tốc/vị trí đó).

IV. Đánh giá chất lượng và kết quả chạy thực tế

1. Tuning vận tốc

Từ quá trình chạy thực nghiệm:

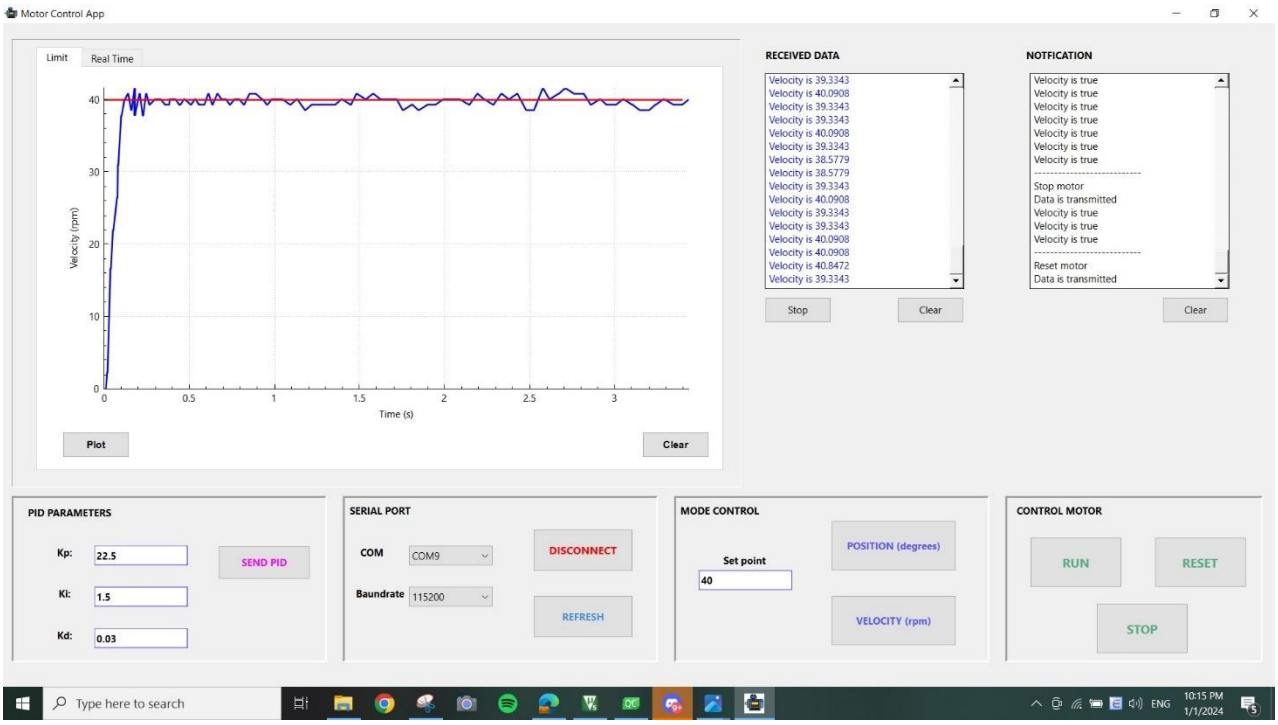
+ Đầu tiên em chỉ tăng Kp thôi và kết quả em quan sát được là: thời gian quá độ, thời gian lên giảm, độ vọt lố tăng

+ Tiếp theo đó em tăng dần dần Ki lên thì kết quả thu được là: sai số xác lập triệt tiêu, giảm thời gian lên, vọt lố tăng cao khi Ki tăng quá lớn.

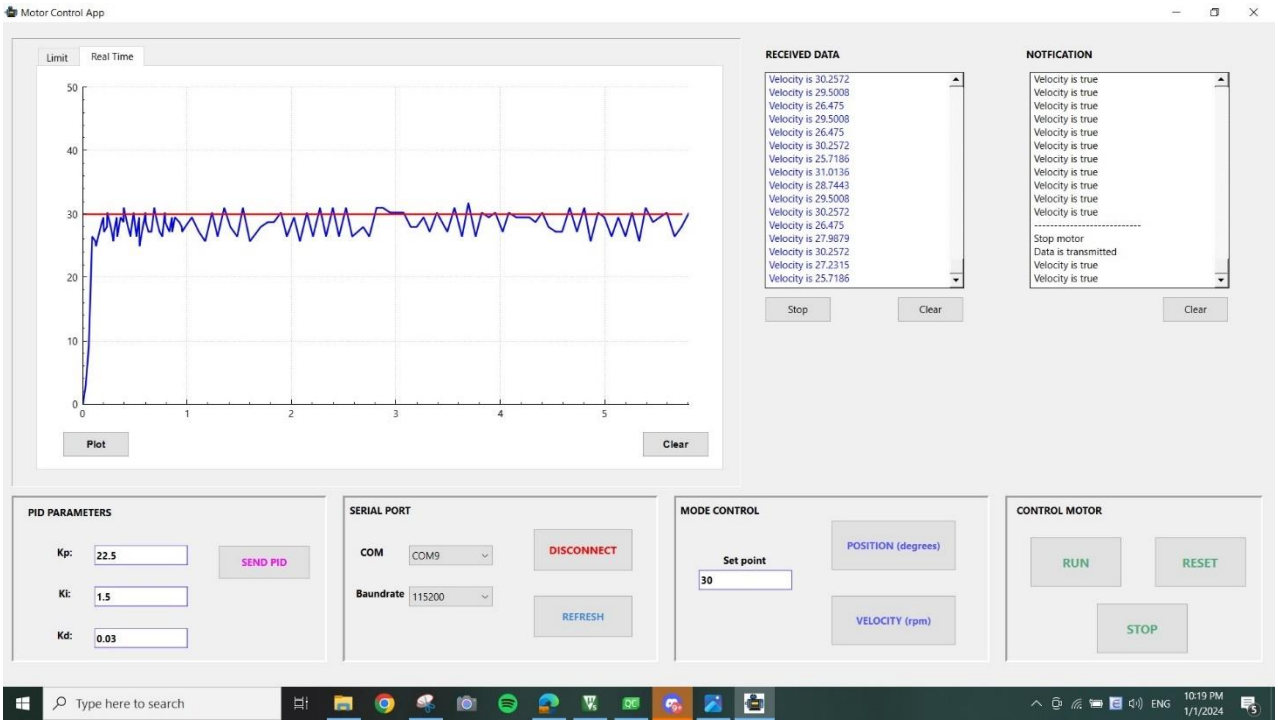
+ Cuối cùng là em tăng dần Kd lên thì kết quả thu được là làm giảm vọt lố nhưng lại làm tăng thời gian và sai số xác lập, hệ mất ổn định khi Kd tăng quá lớn.

=> Từ quá trình chạy thực nghiệm để xét sự ảnh hưởng của từng hệ số lên hệ thống như thế nào thì em đã chọn ra được bộ thông số PID để có thể đáp ứng tốt nhất có thể: Kp = 22.5, Ki = 1.5, Kd = 0.03.

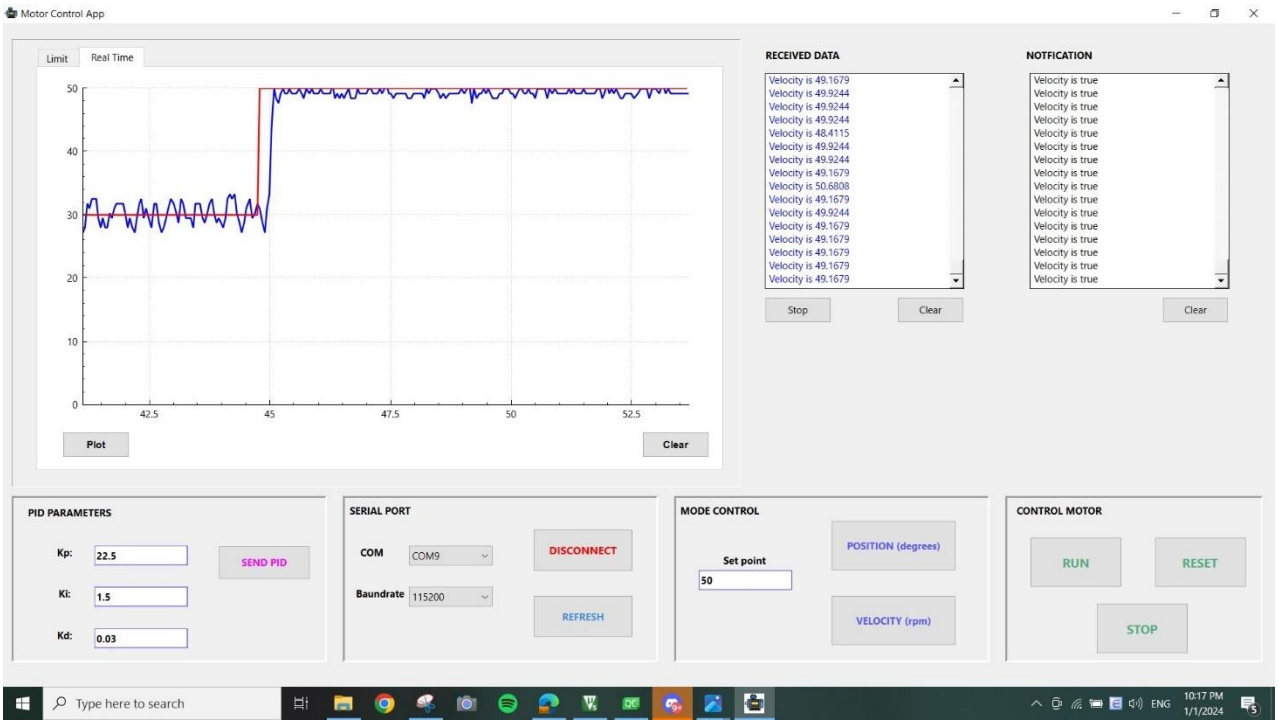
+ Đáp ứng thu được với bộ thông số trên khi setpoint bằng 40 rpm :



+ Đáp ứng thu được với bộ thông số trên khi setpoint bằng 30 rpm :



+ Đáp ứng thu được khi đang chạy thay đổi setpoint từ 30 rpm lên 50 rpm



2. Tuning vị trí

Từ các nhận xét ở trên, ta tiến hành tune tìm thông số PID như sau:

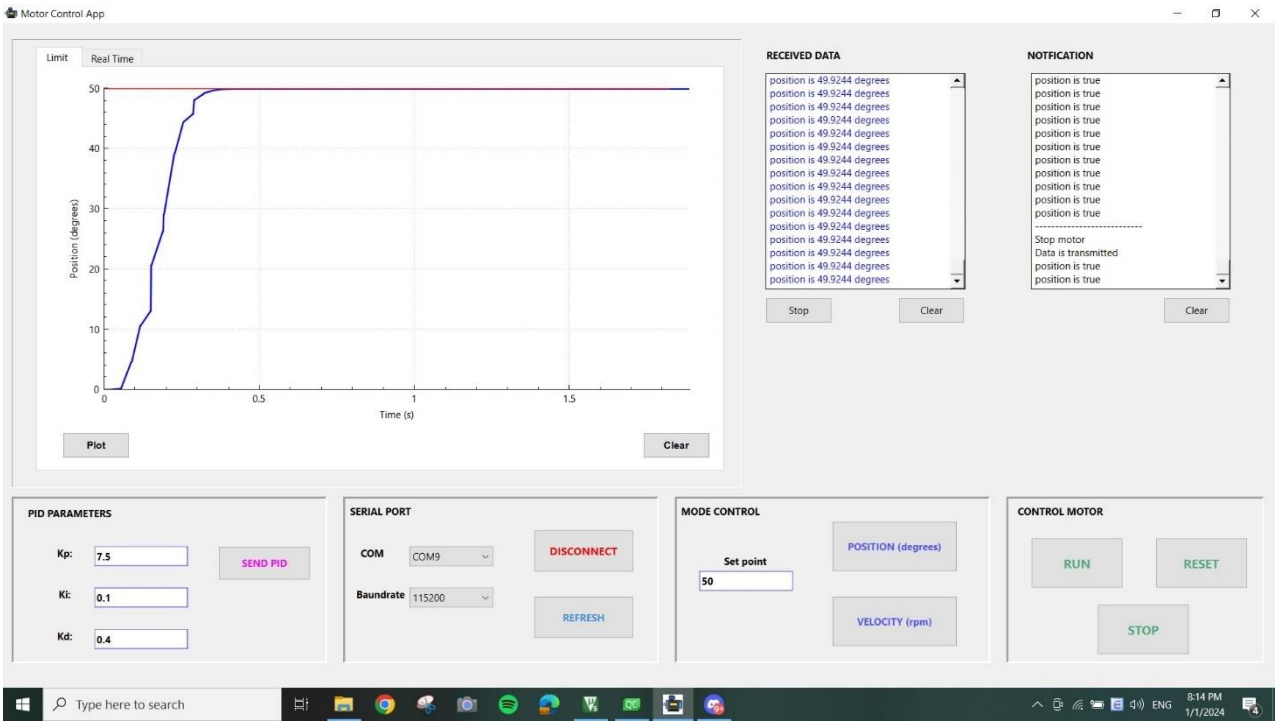
+ Tune Kp đến khi có overshoot và undershoot để đủ năng lượng cung cấp cho hệ thống

+ Tune Kd để giảm đi vọt lố do Kp tạo ra, giảm còn khoảng 5 - 10%

+ Tune Ki để triệt tiêu sai số xác lập còn sót và giảm đi thời gian lên, không tăng quá lớn Ki để tránh độ vọt lố lại tiếp tục tăng

=> Từ kết quả chạy thực nghiệm em thu được bộ thông số PID có đáp ứng tốt nhất: Kp = 7.5, Ki = 0.1, Kd = 0.4.

+ Đáp ứng thu được khi set point 50 degrees:



3. Kết luận thu được sau quá trình chạy thực tế

Tùy theo mục đích điều khiển và những yêu cầu về tối ưu để ta có thể xác định được việc đặt những thông số PID thích hợp nhất. Cách tìm được các thông số này có thể sử dụng các phương pháp tính toán từ thực nghiệm như:

+ Điều chỉnh thủ công bằng thực nghiệm. Phương pháp này đơn giản nhất nhưng đòi hỏi người kĩ sư phải có chuyên môn cao, ước tính gần đúng nhất các thông số thích hợp thì quá trình tìm ra thông số tốt nhất sẽ nhanh chóng.

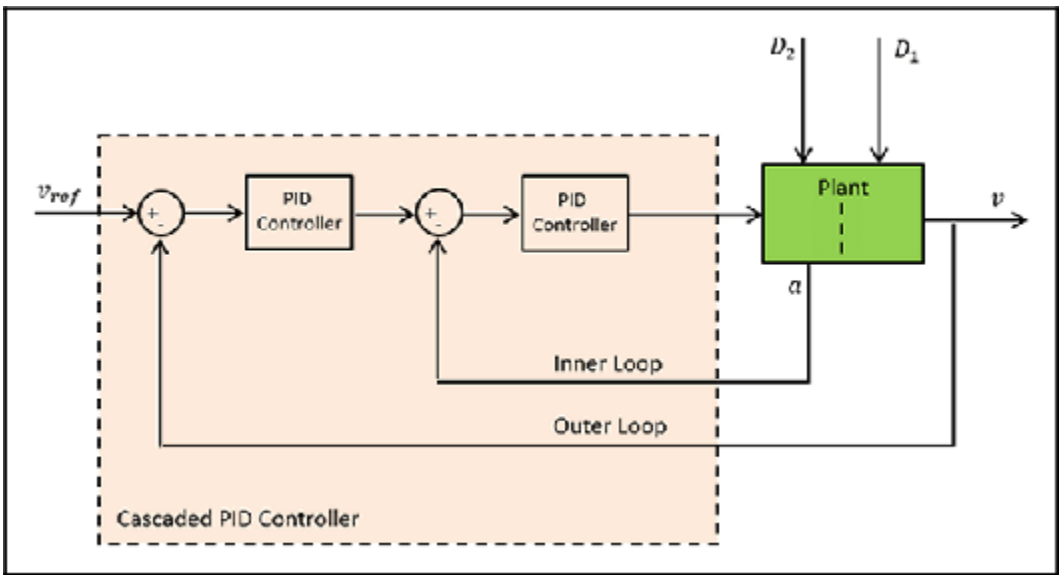
+ Phương pháp cổ điển nhất được áp dụng khá nhiều đó là phương pháp Ziegler– Nichols thường được ứng dụng nhiều và đưa vào chương trình giảng dạy. Phương pháp này sử dụng khá đơn giản vì sẽ chạy thực nghiệm, không cần tính toán nhiều, tuy nhiên sẽ rất mất thời gian thử nhiều lần, nếu thử những thông số quá xấu có thể khiến hiện thống dao động rất mạnh.

+ Khi quá trình phát triển của công nghệ kĩ thuật ngày càng mạnh mẽ, và ứng dụng AI ngày càng được ứng dụng rộng rãi thì các phần mềm tự điều chỉnh thông số được ra đời. Các phần mềm hỗ trợ rất nhiều trong việc tìm chính xác và nhanh chóng các hệ số PID phù hợp, tuy

nhiên giá cả khá cao và cần phải được huấn luyện nhiều mới có thể áp dụng tốt vào thực tiễn.

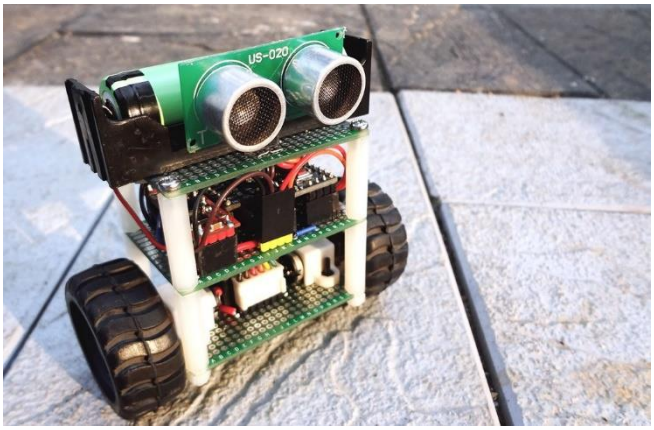
V. Hướng phát triển

Hướng 1: Từ nắm vững nền tảng của bộ điều khiển PID đơn, từ đó em sẽ phát triển hệ thống của mình lên hệ thống có bộ điều khiển PID cascade, đây là những bộ điều khiển được áp dụng rất thường xuyên và rộng rãi trong môi trường công nghiệp. Sau đó là tiếp tục vươn lên các bộ điều khiển STR..



Hình 24: PID cascades

Hướng 2: Sử dụng bộ điều khiển PID điều khiển động cơ vào trong việc làm xe cân bằng. Để cân bằng một chiếc xe 2 bánh mang phía trên một tải có trọng lượng lớn sẽ cần đến bộ điều khiển PID giúp xe xác định được góc nghiêng hiện tại của xe so với góc thẳng đứng để thay đổi chiều quay của động cơ giúp xe luôn được giữ vững



Hình 25: Ứng dụng PID trong làm xe cân bằng

Hướng 3: Làm drone

Drone, có nhiều loại như quadcopter(4 cánh) , bicopter(2 cánh),… Được thiết kế nhẹ nhàng và dễ dàng cầm nắm, drone mini trở thành một công cụ phổ biến cho giải trí, chụp ảnh và quay phim từ không gian cao. Một trong những ưu điểm nổi bật của drone mini là tính di động. Kích thước nhỏ giúp nó dễ dàng mang đi bất cứ nơi Nhiều drone cũng hỗ trợ chế độ bay tự động và các chức năng thông minh như bay theo đường đi được lập trình trước, bay quanh một vật thể cố định hoặc bay theo chế độ theo dõi đối tượng.

Hình 26: Ứng dụng PID để điều khiển drone

VI. Tài liệu tham khảo

[1] Đ. Đ. Công, “Điều khiển tốc độ động cơ DC dùng bộ điều khiển PID”, Hanoi University of Science and Technology, 2013.

[2] H. T. Hoàng, "Cơ sở điều khiển tự động”, Ho Chi Minh University of Technology, 2011.

[3] Bài báo cáo đồ án điều khiển động cơ của các anh chị Bách Khoa đi trước.