

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG



BÁO CÁO ĐỒ ÁN 1

ĐỀ TÀI:

THIẾT KẾ BỘ ĐIỀU KHIỂN PID SỐ ĐIỀU KHIỂN

ĐỘNG CƠ DC SERVO

Giảng viên hướng dẫn: cô Bùi Thanh Huyền

Sinh viên thực hiện

Nguyễn Ngọc Khanh

Mã số sinh viên

2111474

Nguyễn Thị Vân Hà

2110148

Thành phố Hồ Chí Minh, tháng 6, năm 2024

MỤC LỤC

1.	Khái quát đề tài	4
1.1.	Đặt vấn đề.....	4
1.2.	Hướng giải quyết	5
2.	Động cơ DC	7
2.1.	Cấu tạo và nguyên lý hoạt động	7
2.2.	Mô hình hóa động cơ DC từ trường vĩnh cửu	9
2.3.	Phương pháp điều khiển tốc độ động cơ	11
3.	Giải thuật điều khiển PID	13
4.	Phần cứng.....	16
4.1.	Động cơ Encoder giảm tốc JGB37 - 545	16
4.2.	Mạch lái	18
4.3.	USB - UART CP2102	20
5.	Lập trình nhúng.....	22
5.1.	Giới thiệu vi điều khiển ESP32	22
5.2.	Lưu đồ giải thuật của chương trình nhúng.....	24
6.	Lập trình giao diện	25
6.1.	Giới thiệu công cụ Visual Studio	25
6.2.	Cấu trúc chương trình giao diện	25
6.2.1.	Khối Communication	26
6.2.2.	Khối Data	26
6.2.3.	Khối Operation	26
6.2.4.	Đồ thị.....	27
7.	Phương thức trao đổi dữ liệu giữa ESP32 và GUI	28
8.	Đánh giá kết quả thực hiện, chất lượng hệ thống.....	29
8.1.	Điều khiển vị trí	29
8.1.1.	Thay đổi Kp (Bộ điều khiển P)	29
8.1.2.	Thay đổi Ki (Bộ điều khiển PI)	32

8.1.3. Thay đổi Kd (Bộ điều khiển PID)	34
8.1.4. Nhận xét.....	36
8.2. Điều khiển vận tốc	37
8.2.1. Thay đổi Kp (bộ điều khiển P)	37
8.2.2. Thay đổi Ki (bộ điều khiển PI).....	40
8.2.3. Thay đổi Kd (bộ điều khiển PID).....	42
8.2.4. Nhận xét.....	44
9. Kết luận chung.....	45
Danh mục tài liệu tham khảo	46

1. Khái quát đề tài

1.1. Đặt vấn đề

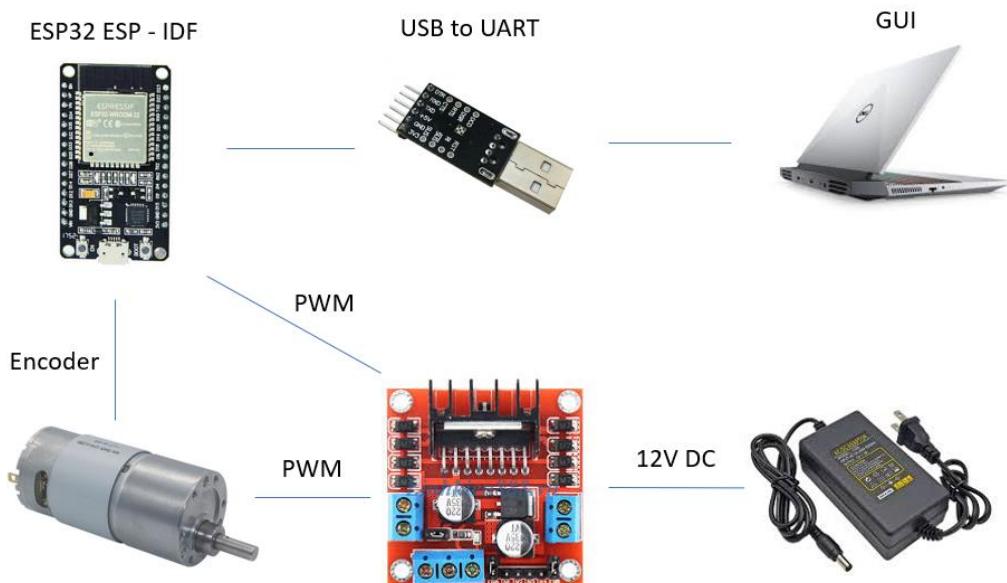
Trong thời đại hiện nay, động cơ DC đóng vai trò quan trọng trong nhiều lĩnh vực của khoa học và đời sống, từ sản xuất công nghiệp đến các ứng dụng trong ô tô điện, tàu thủy, máy bay, và nhiều lĩnh vực khác. Việc thiết kế mạch để điều khiển động cơ DC đã trở thành một đề tài phổ biến trong ngành điều khiển tự động. Để đạt được hiệu suất tốt, cần thiết phải thiết kế một bộ điều khiển linh hoạt để điều khiển vị trí và tốc độ động cơ một cách ổn định và đáp ứng nhanh.

Mặc dù có nhiều phương pháp điều khiển khác nhau, nhưng trong khuôn khổ đề tài, nhóm em đã chọn bộ điều khiển PID (Proportional – Integral - Derivative) với sự đơn giản và độ tin cậy cao.

Thực tế, hơn 90% các bộ điều khiển sử dụng trong các ứng dụng thực tế đều là PID. Sự linh hoạt của PID cho phép người sử dụng dễ dàng tích hợp và tinh chỉnh các luật điều khiển như tỉ lệ (P), tích phân (I), tỉ lệ tích phân (PI), tỉ lệ vi phân (PD) và tỉ lệ tích phân vi phân (PID), các luật điều khiển được áp dụng tùy thuộc vào đặc điểm cụ thể của đối tượng điều khiển. PID cũng là nền tảng cơ bản để phát triển và áp dụng các phương pháp và thuật toán điều khiển tiên tiến hơn trong tương lai.

Đề tài này đặt ra mục tiêu cung cấp kiến thức lý thuyết và kỹ năng thực hành cần thiết cho việc thiết kế và lập trình bộ điều khiển PID cho động cơ DC. Ngoài ra, thông qua các công đoạn tìm hiểu, thiết kế, và cải thiện chất lượng hệ thống điều khiển, đề tài cũng giúp sinh viên làm quen với quy trình làm việc thực tế của một kỹ sư điều khiển tự động.

Sơ đồ mạch của hệ thống điều khiển động cơ DC như sau:



1.2. Hướng giải quyết

Đầu tiên, nhóm sẽ nắm vững về nguyên lý hoạt động của động cơ DC và cách thức hoạt động của Encoder. Điều này bao gồm việc hiểu cách động cơ hoạt động, cách Encoder đo lường vị trí và tốc độ của động cơ. Sau đó nhóm chọn động cơ Encoder giảm tốc JGB37 - 545 (DC 12V - 1/30- 168 RPM) làm đối tượng nghiên cứu. Đồng thời, sử dụng mạch lái L298N để hỗ trợ PWM lái động cơ và mạch chuyển tín hiệu USB to UART (CP2102) để giao tiếp giữa vi xử lý và giao diện WinForm GUI trên máy tính.

Thứ hai, nhóm tìm hiểu cách hoạt động của vi điều khiển ESP32 và sử dụng framework ESP – IDF có thể can thiệp sâu vào phần cứng để lập trình cho vi điều khiển này. Điều này đảm bảo việc điều khiển động cơ DC được thực hiện một cách chính xác và hiệu quả.

Thứ ba, nhóm sử dụng phần mềm Visual Studio trên hệ điều hành Windows và ngôn ngữ lập trình C#, nhóm sẽ thiết kế giao diện người dùng WinForm GUI trên máy tính. Giao diện này sẽ kết nối với vi xử lý đã kết nối với động cơ để thay đổi thông số PID và hiển thị đáp ứng của động cơ.

Cuối cùng, nhóm sẽ đánh giá sơ bộ về kết quả thu được đối với đối tượng động cơ DC. Điều này bao gồm việc đánh giá tính ổn định và thời gian đáp ứng của hệ thống điều khiển.

Thông qua các bước trên, nhóm sẽ có thể hiểu rõ hơn về cách thiết kế và điều khiển động cơ DC sử dụng bộ điều khiển PID, đồng thời áp dụng kiến thức này vào thực tế thông qua việc thực hiện dự án.

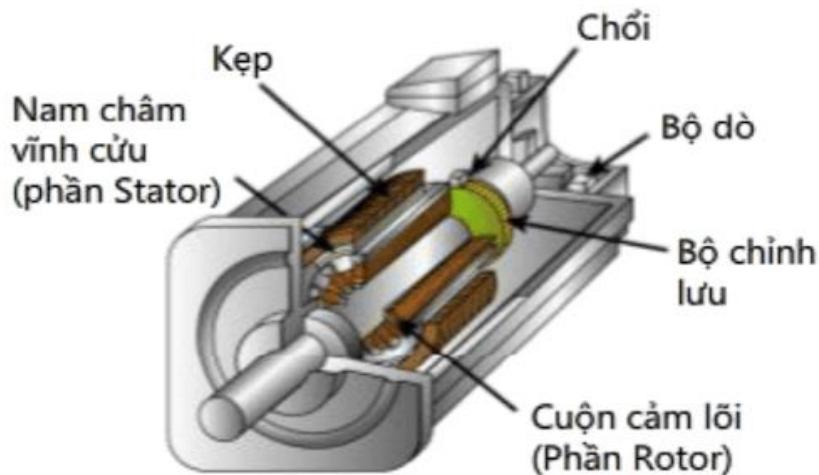
2. Động cơ DC

2.1. Cấu tạo và nguyên lý hoạt động

Động cơ DC (Direct Current Motor) là động cơ một chiều được sử dụng và ứng dụng trong dòng điện một chiều. Động cơ DC còn được coi là máy điện chuyển đổi năng lượng điện thành năng lượng cơ học.

⊕ Cấu tạo của động cơ DC:

Gồm có 3 phần chính đó là rotor (phần ứng), stator (phần cảm) và phần cố gòp – chỉnh lưu.



Cấu tạo chung động cơ điện một chiều DC

- Stator là phần không quay của động cơ, bao gồm các cực từ tạo ra từ trường. Stator của động cơ điện 1 chiều thường sẽ là 1 hoặc nhiều những cặp nam châm vĩnh cửu, hoặc là nam châm điện.
- Rotor là phần quay của động cơ, được gắn trên trục quay. Rotor có những cuộn dây quấn và được nối với những nguồn điện một chiều.

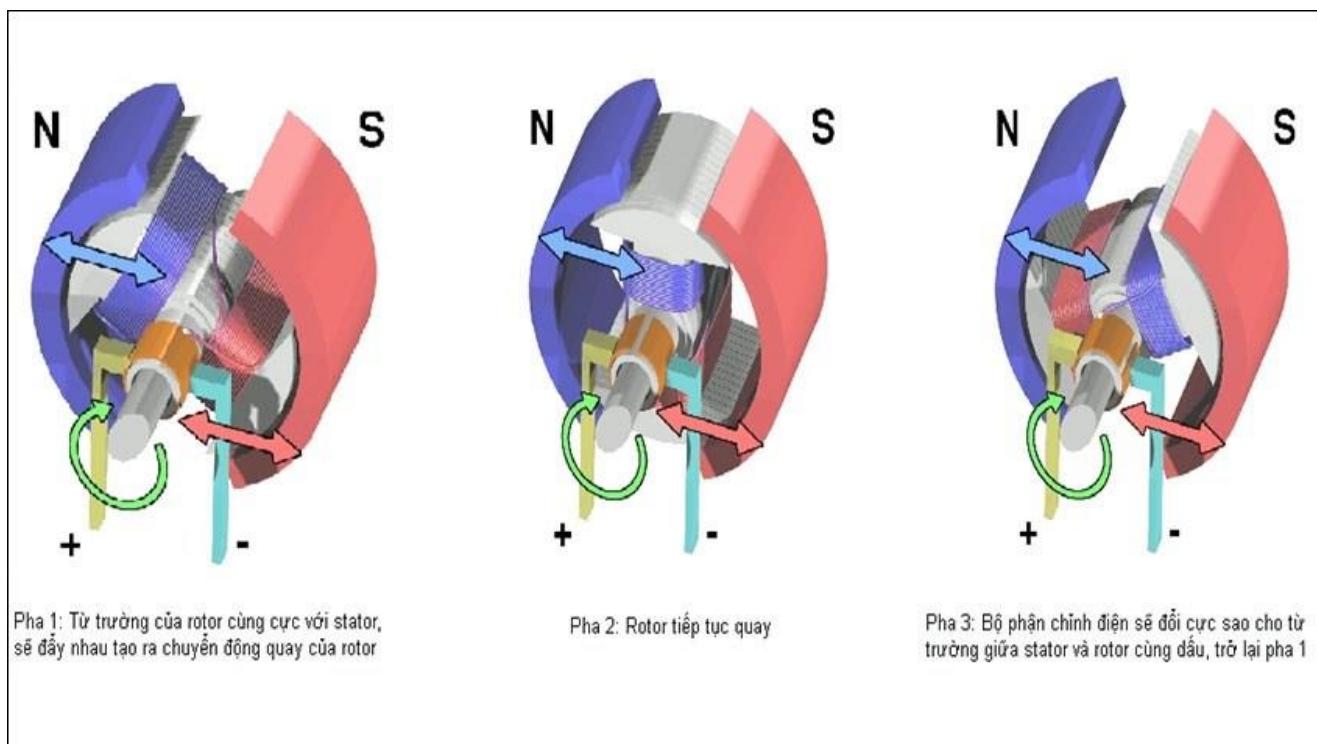
- Bộ phận chỉnh lưu có nhiệm vụ là đổi chiều của dòng điện trong khi chuyển động quay của Rotor là liên tục. Thông thường thì bộ phận này gồm có một bộ cỗ gộp và một bộ chổi than để tiếp xúc với cỗ gộp.

Nguyên lý hoạt động của động cơ DC:

Động cơ DC là thiết bị chuyển đổi năng lượng điện trực tiếp thành chuyển động cơ học. Khi năng lượng điện được cung cấp, động cơ DC tạo ra một từ trường trong phần Stator. Từ trường này tác động lên nam châm trên Rotor, tạo ra lực đẩy và thu hút, thúc đẩy Rotor quay. Nguyên lý này dựa trên "Quy tắc bàn tay trái" trong vật lý.

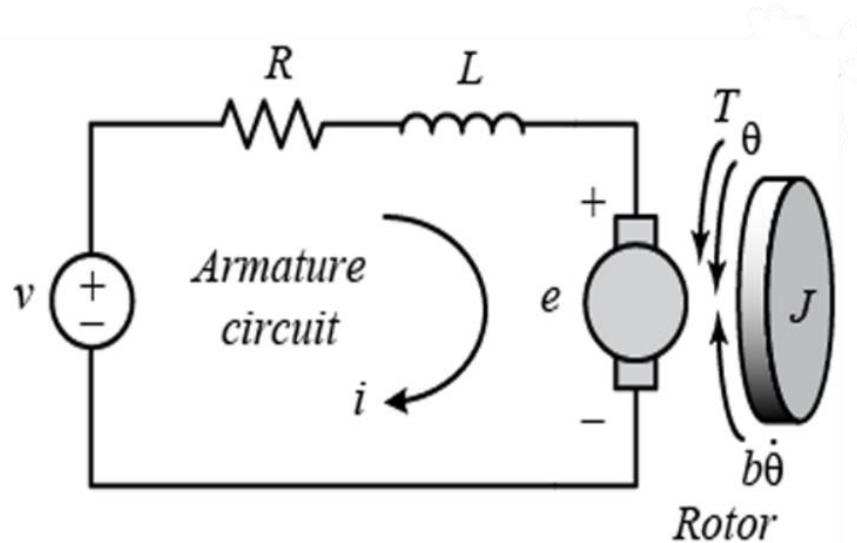
Để đảm bảo Rotor quay liên tục, bộ chuyển đổi được gắn vào bàn chải kết nối với dòng điện để cung cấp nguồn điện cho cuộn dây động cơ.

Tốc độ quay của động cơ DC thay đổi theo chu kỳ thời gian và phụ thuộc vào ứng dụng cụ thể. Đo lường tốc độ này thường được thực hiện dưới dạng vòng/phút hoặc nghìn vòng/phút. Chúng em sẽ sử dụng đơn vị vòng/phút (RPM) cho tốc độ động cơ.



Nguyên lý hoạt động của động cơ điện một chiều DC

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



Mô hình động cơ DC

Trong đó:

- R điện trở phản ứng
- L điện cảm phản ứng
- J momen quán tính trên trục động cơ
- B hệ số ma sát nhót
- K_t hằng số momen xoắn
- K_e hằng số sức phản điện

Ngô vào:

- U điện áp nguồn

Ngõ ra:

- θ vị trí động cơ

- $\dot{\theta}$ tốc độ động cơ

Moment xoắn trên trục động cơ

$$T = K_t \cdot i$$

Sức điện động trên động cơ

$$e = K_e i$$

Theo định luật 2 Newton

$$J\ddot{\theta} + b\dot{\theta} = K_t \cdot i \quad (*)$$

Theo định luật Kirchoff

$$L \frac{d_i}{dt} + R_i = U - K_e \dot{\theta}$$

Biến đổi Laplace 2 phương trình

$$s(Js + b) \cdot \theta(s) = K_t \cdot I(s)$$

$$(Ls + R) \cdot I(s) = U(s) - K_e s \theta(s)$$

Suy ra hàm truyền vị trí và tốc độ động cơ

$$\frac{\theta(s)}{U(s)} = \frac{1}{s} \cdot \frac{K_t}{(Js + b)(Ls + R) + K_e \cdot K_t}$$

$$\frac{\dot{\theta}(s)}{U(s)} = \frac{K_t}{(Js + b)(Ls + R) + K_e \cdot K_t}$$

Ở tần số thấp, điện cảm động cơ rất nhỏ nên có thể bỏ qua, khi đó

$$\frac{\theta(s)}{U(s)} = \frac{1}{s} \cdot \frac{K_t}{RJs + (bR + K_e \cdot K_t)} = \frac{1}{s} \cdot \frac{K}{\tau s + 1}$$

$$\frac{\dot{\theta}(s)}{U(s)} = \frac{K_t}{RJs + (bR + K_e \cdot K_t)} = \frac{K}{\tau s + 1}$$

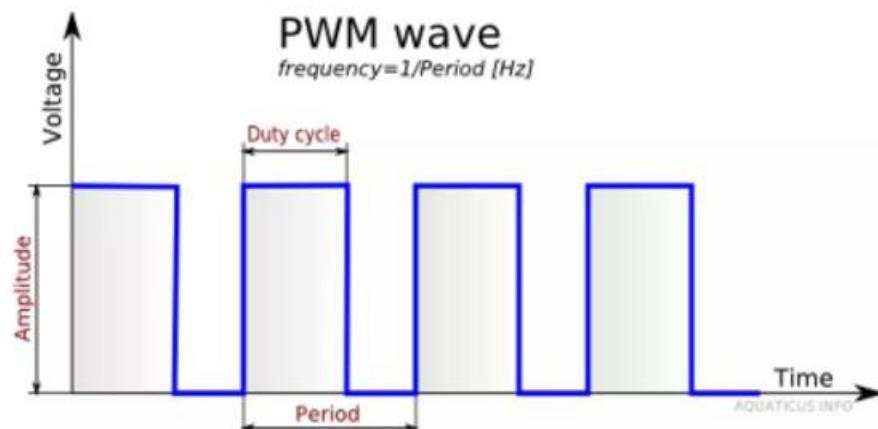
Với độ lợi và thời hằng:

$$k = \frac{K_t}{bR + K_t K_e}, \tau = \frac{JR}{bR + K_t K_e}$$

2.3. Phương pháp điều khiển tốc độ động cơ

Dựa vào phương trình (*) như trên, có thể thấy quan hệ giữa tốc độ động cơ với điện áp đặt vào hai đầu cuộn dây phản ứng. Vì tác động cơ học của động cơ tương đối nhanh nên muốn thay đổi tốc độ hay vị trí động cơ thì phải thông qua thay đổi điện áp cấp tuy nhiên điều này khá khó khăn.

Do đó nhóm sử dụng phương pháp điều chế độ rộng xung (PWM), là cách điều chỉnh điện áp đầu ra bằng cách thay đổi độ rộng của các xung vuông. Thay vì sử dụng điện áp liên tục, chúng ta sử dụng các xung có tần số cố định nhưng có độ rộng xung thay đổi. Khi độ rộng xung thay đổi, giá trị trung bình của điện áp cũng thay đổi, từ đó điều khiển được tốc độ của động cơ một chiều. Điều này giúp ổn định tốc độ và điều chỉnh được tốc độ của động cơ một chiều một cách hiệu quả.



Nguyên lý điều chế độ rộng xung PWM

Giá trị trung bình điện áp trên tải:

$$U_t = U_{max} \frac{T_{on}}{T_{off}} = Amplitude \cdot \frac{Duty\ cycle}{Period}$$

Ta thấy giá trị điện áp trên tải phụ thuộc vào tỉ số $\gamma = \frac{Duty\ cycle}{Period}$, do đó ứng với mỗi tần số xung, ta có thể điều chỉnh Duty cycle để điều chỉnh điện áp. Tuy nhiên, để tạo xung

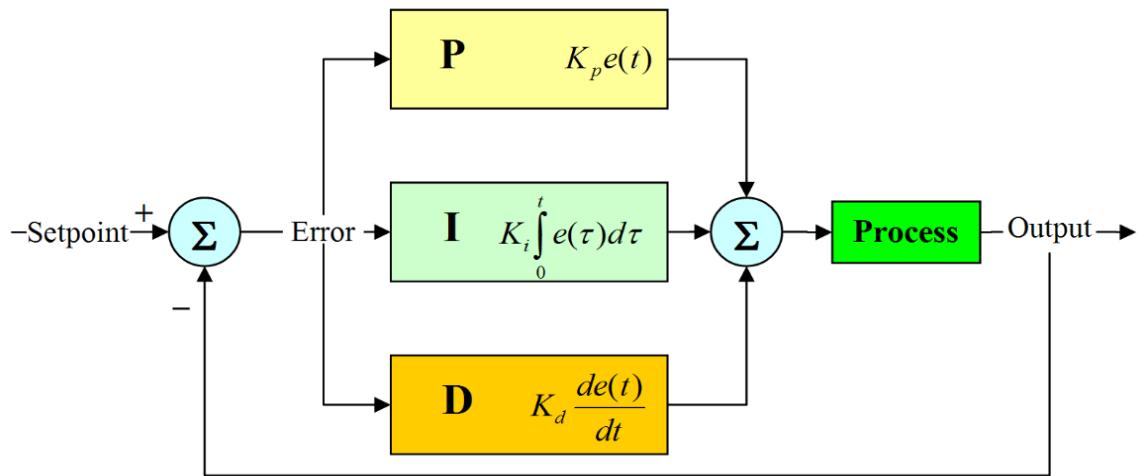
PWM, ta cần điều khiển thông qua giá trị duty cycle nên dải tần số sẽ thu hẹp lại, phụ thuộc vào dải điều chỉnh của duty cycle.

Ví dụ: Nếu ta lấy duty cycle có giá trị trong khoảng từ 0 – 255 thì dải tần số của xung PWM xuất ra sẽ là khoảng 15Hz - 4kHz.

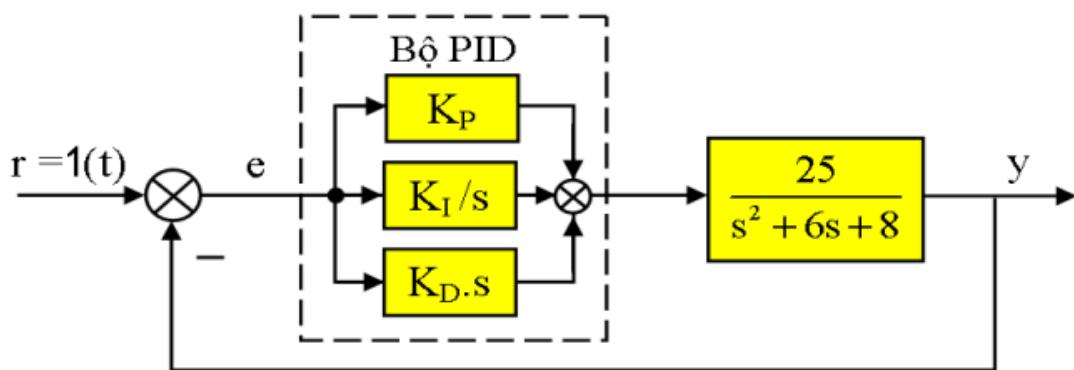
Khi lựa chọn tần số của xung PWM, ta cần lựa chọn sao cho đáp ứng cơ học của động cơ đủ mịn để không có cảm giác bị vấp do điện áp thay đổi.

3. Giải thuật điều khiển PID

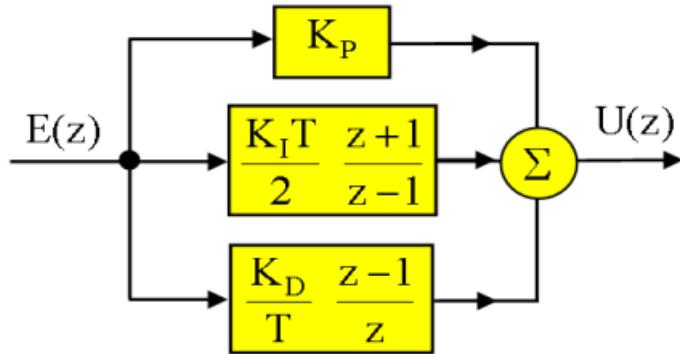
Điều khiển PID là một kiểu điều khiển có hồi tiếp vòng kín được sử dụng nhiều trong công nghiệp, ngõ ra thay đổi tương ứng với sự thay đổi của giá trị đo. PID là sự kết hợp của 3 bộ điều khiển: tỉ lệ, tích phân và vi phân, có khả năng làm triệt tiêu sai số xác lập, tăng tốc độ đáp ứng, giảm độ vọt lồ nếu thông số của bộ điều khiển được lựa chọn thích hợp.



Sơ đồ khái của một hệ kín có bộ PID



Bộ điều khiển PID số (liên tục)



Bộ điều khiển PID số (rời rạc)

Ngõ ra miền thời gian của bộ điều khiển PID

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

Hàm truyền bộ điều khiển PID miền liên tục

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

Hàm truyền bộ điều khiển PID miền rời rạc

$$G_{PID}(z) = K_P + \frac{K_I T}{2} \cdot \frac{z+1}{z-1} + \frac{K_D}{T} \cdot \frac{z-1}{z}$$

Để đáp ứng ngõ ra đạt chất lượng mong muốn, ta điều chỉnh hệ số K_P, K_I, K_D : điều khiển tỉ lệ (K_P) có ảnh hưởng làm giảm thời gian lên và sẽ làm giảm nhưng không loại bỏ sai số xác lập. Điều khiển tích phân (K_I) sẽ loại bỏ sai số xác lập nhưng có thể làm đáp ứng quá độ xấu đi. Điều khiển vi phân (K_D) có tác dụng làm tăng sự ổn định của hệ thống, giảm vọt lồ và cải thiện đáp ứng quá độ. Ảnh hưởng của mỗi bộ điều khiển K_P, K_I, K_D lên hệ thống vòng kín được tóm tắt ở bảng bên dưới (Bảng 1).

Bảng 1: *Ảnh hưởng của mỗi bộ điều khiển K_P , K_I , K_D*

Đáp ứng vòng kín	Thời gian lên	Vọt lố	Thời gian xác lập	Sai số xác lập
K_p	Giảm	Tăng	Thay đổi nhỏ	Giảm
K_I	Giảm	Tăng	Tăng	Loại bỏ
K_d	Thay đổi nhỏ	Giảm	Giảm	Thay đổi nhỏ

Chú ý rằng các mối liên hệ này không chính xác hoàn toàn bởi vì K_p , K_I , K_D phụ thuộc vào nhau. Vì vậy, bảng này chỉ dùng tham khảo khi xác định các tham số K_p , K_I , K_D

4. Phần cứng

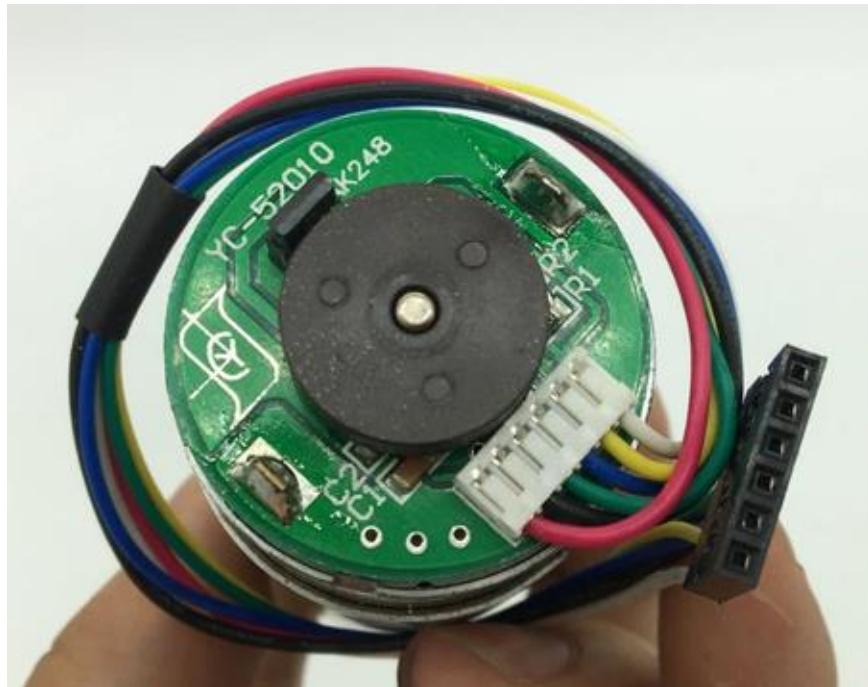
4.1. Động cơ Encoder giảm tốc JGB37 - 545

Động cơ DC Encoder không chỉ đơn thuần cung cấp động lực, mà còn cung cấp số xung từ Encoder, từ đó ta có thể tính toán chính xác vị trí và vận tốc để phục vụ cho việc điều khiển.

Thông qua việc phản hồi (feedback) từ Encoder, vi điều khiển có thể thực hiện điều chỉnh động cơ thông qua mạch công suất, sử dụng các thuật toán điều khiển như PID (Proportional – Integral - Derivative). Điều này giúp đạt được việc điều khiển tốc độ và vị trí của động cơ một cách chính xác và hiệu quả. Đồng thời, thông tin từ Encoder cung cấp dữ liệu quan trọng cho quá trình giám sát và điều khiển toàn bộ hệ thống, tăng cường tính đáng tin cậy và hiệu suất của các ứng dụng Robotics.

Thông số động cơ Encoder giảm tốc JGB37 - 545 (12V DC - 1/30- 168 RPM)

Thông số kỹ thuật	
Động cơ	
Điện áp làm việc	6 – 12 VDC
Tốc độ không tải	168 RPM
Tỉ số hộp số	1:30
Encoder	
Điện áp nguồn	5 VDC
Số xung/vòng	480



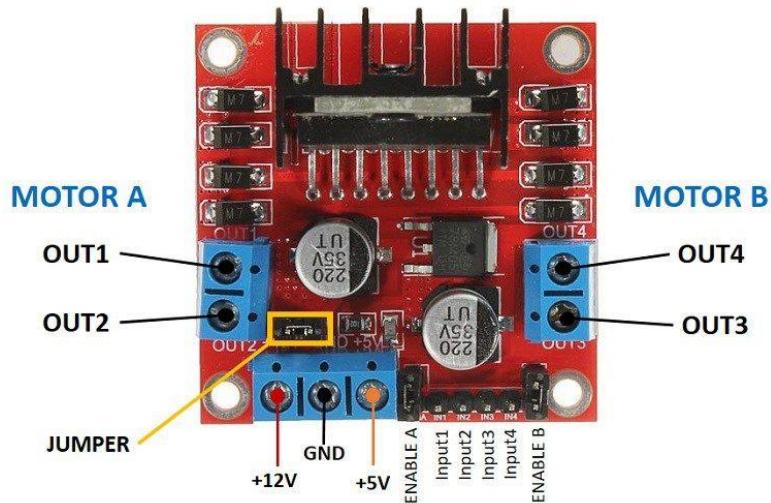
Encoder của động cơ JGB37 – 545

Encoder, hay còn gọi là bộ mã hóa vòng quay, được sử dụng chủ yếu để phát hiện vị trí, hướng di chuyển, và tốc độ của động cơ bằng cách đếm số vòng mà trực quay được.

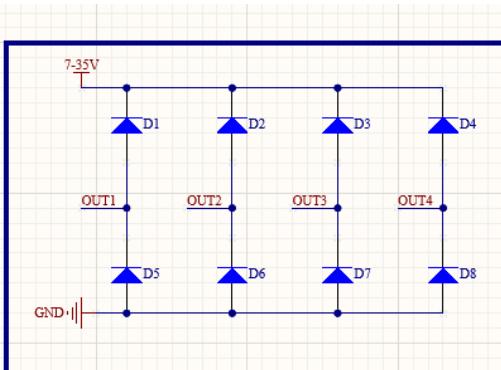
Cấu tạo của Encoder bao gồm thân và trực, nguồn phát sáng (light source), đĩa mã hóa (code disk) chia vòng tròn thành các góc bằng nhau, bộ cảm biến ánh sáng thu tín hiệu (photosensor), và bo mạch điện tử (electronic board) giúp khuếch đại tín hiệu.

Encoder hoạt động dựa trên nguyên lý đĩa quay quanh trực. Các rãnh trên đĩa mã hóa khi quay và chiếu đèn LED sẽ tạo ra sự ngắt quãng, từ đó ghi nhận được số xung và tốc độ xung. Cảm biến thu ánh sáng sẽ bật tắt liên tục để tạo ra các xung vuông, và các bộ mã hóa ghi nhận lại số xung và tốc độ xung này. Tín hiệu dạng xung được truyền về bộ xử lý trung tâm (vi xử lý, PLC, ...), từ đó kỹ sư cơ khí có thể biết được vị trí và tốc độ của động cơ.

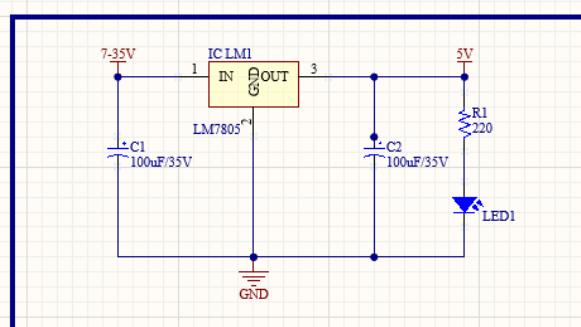
4.2. Mạch lái



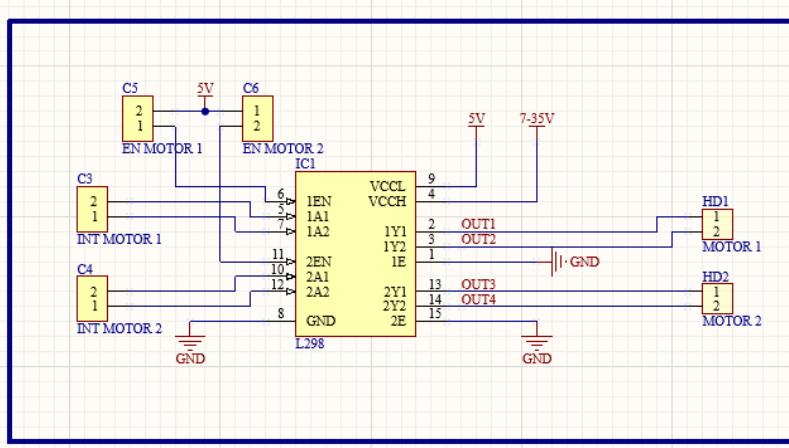
Module điều khiển động cơ L298N



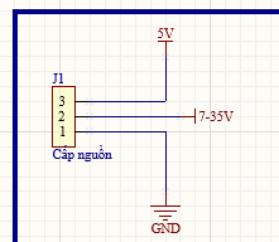
Khối chống ngược dòng



Khối 5V



Khối L298N



Khối nguồn

Sơ đồ nguyên lý module L298N

Thông số kỹ thuật	Giá trị
Điện áp điều khiển	+5V ~ +12V
Dòng tối đa của mỗi cầu H	2A
Điện áp tín hiệu điều khiển	+5V ~ +7V
Dòng tín hiệu điều khiển	0 ~ 36mA
Công suất hao phí	20W (khi nhiệt độ T = 75°C)



Chức Năng của L298N:

Module điều khiển động cơ L298N là một bộ điều khiển sử dụng mạch cầu H để dễ dàng điều khiển chiều quay và tốc độ của tối đa 2 động cơ DC. Mạch cầu H có các Diode để dẫn dòng điện ngược dòng sinh ra từ cuộn cảm của Motor về nguồn, tránh làm hỏng vi xử lý.



Các Chân Trong Module L298N và Chức Năng:

Bốn chân INPUT: IN1, IN2, IN3, IN4 là các chân nhận tín hiệu điều khiển.

Bốn chân OUTPUT: OUT1, OUT2, OUT3, OUT4 (tương ứng với các chân INPUT). Các chân này sẽ được nối với động cơ DC hoặc động cơ bước.

Hai chân cho phép ENA và ENB dùng để điều khiển mạch cầu H trong IC L298N. Nếu ở mức logic “1” (nối với nguồn 5V) cho phép mạch cầu H hoạt động, nếu ở mức logic “0” thì mạch cầu H không hoạt động. Các chân này cũng được dùng để điều khiển tốc độ của động cơ.

Jumper: Tháo ra khi sử dụng nguồn trên 12V.



Cách Sử Dụng L298N:

Nên sử dụng L298N với động cơ có điện áp điều khiển từ 5V – 35V.

Sử dụng các chân OUT1, OUT2 cho động cơ 1 và OUT3, OUT4 cho động cơ 2.

Các chân IN1, IN2, IN3, IN4, ENA, ENB nối với 6 chân điều khiển của vi điều khiển.

Dùng Module L298N Để Điều Khiển Động Cơ;

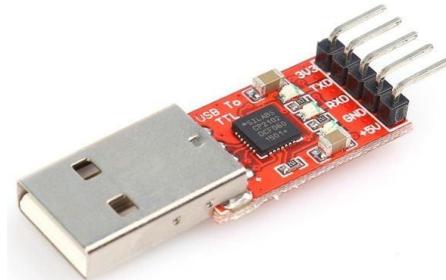
IN1 = 1, IN2 = 0: Động cơ quay thuận.

IN1 = 0, IN2 = 1: Động cơ quay nghịch.

IN1 = IN2: Động cơ dừng ngay lập tức.

ENA ∈ (0,255): Để điều khiển tốc độ động cơ. Với ENB cũng tương tự với IN3, IN4.

4.3. USB - UART CP2102



Mạch Chuyển USB UART CP2102 sử dụng chip chuyển đổi CP2102 USB to TTL của Silabs. Sản phẩm này được thiết kế để giao tiếp nối tiếp giữa các mạch điện tử với máy tính, làm mạch nạp cho vi điều khiển có hỗ trợ nạp thông qua giao tiếp UART.

Thông số	Chi tiết
Chip UART	CP2102
Nguồn cung cấp	5 VDC qua cổng USB
Điện áp ngõ ra	5 VDC và 3.3 VDC
Dòng điện ngõ ra	Tối đa 100 mA
Tốc độ truyền/nhận	300 bps - 1 Mbps

Sơ đồ ra chân	
3.3V (Output)	Chân điện áp dương 3.3V
TXD (Output)	Cổng truyền tín hiệu UART
RXD (Input)	Cổng nhận tín hiệu UART
GND (Output)	Chân điện áp 0V
+5V (Output)	Chân điện áp dương 5V

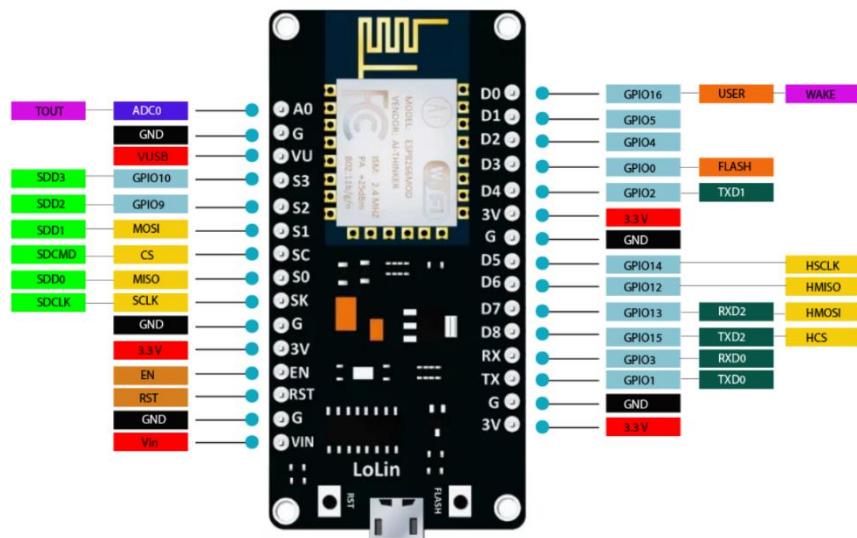
5. Lập trình nhúng

5.1. Giới thiệu vi điều khiển ESP32

Với yêu cầu của đề tài đặt ra là điều khiển được các thiết bị điện, qua quá trình nghiên cứu, tìm hiểu tài liệu chúng em đã chọn giải pháp tối ưu cho khối xử lý trung tâm là ESP32 được phát triển bởi Espressif Systems. Là một vi điều khiển 32bit, đáp ứng đầy đủ các yêu cầu ban đầu đặt ra, ngoài ra còn có thể mở rộng thêm một số chức năng khác khi cài tiến hệ thống.

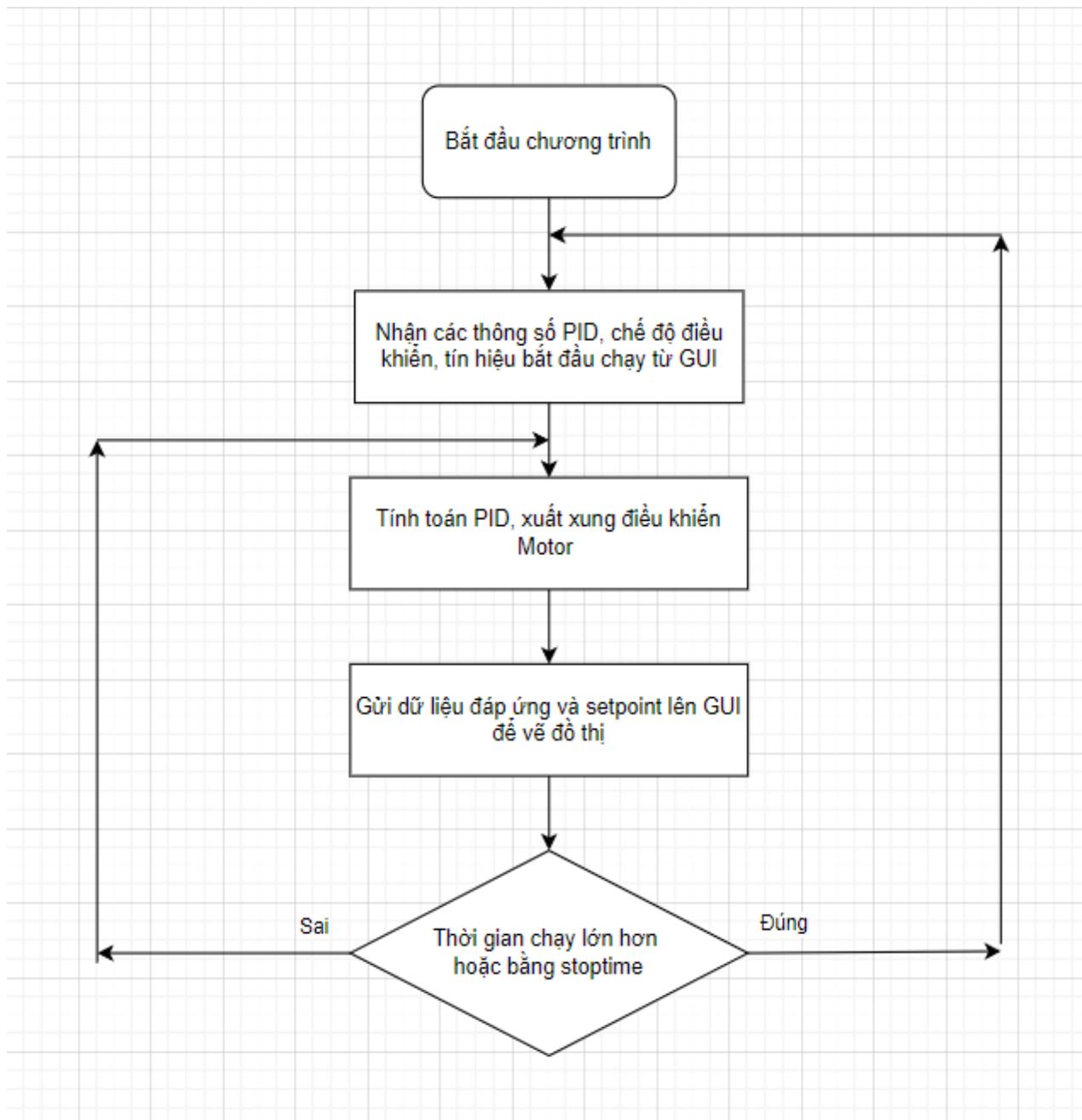
ESP32 là một chip được tích hợp công nghệ WiFi và Bluetooth với công nghệ tiêu thụ năng lượng cực thấp. Nó cung cấp một nền tảng tích hợp mạnh mẽ, đáp ứng nhu cầu hiệu năng tốt nhất, tính linh hoạt, thiết kế nhỏ gọn, hiệu suất cao và độ tin cậy trong nhiều ứng dụng. Các dòng chip ESP32 bao gồm ESP32-D0WDQ6, ESP32-D0WD, ESP32-D2WD và ESP32-S0WD.

Espressif đã thiết kế và sản xuất ra một số module để người dùng dễ dàng tiếp cận hơn với dòng chip ESP32. Các thành phần chính trên những module này bao gồm chip ESP32, bộ tạo dao động thạch anh, mạch ăngten, chỉ khác nhau về một số chức năng tùy từng phiên bản như số lượng chân GPIO, các thiết bị ngoại vi được thêm vào như: màn hình LCD, bảng cảm ứng, khe cắm thẻ SD, module máy ảnh, ...



Thông số	Chi tiết
Điện áp sử dụng	2.2V ~ 3.6V DC
Dòng điện sử dụng	~90mA
Nhân xử lý trung tâm	ESP32 - D0WDQ6 Dual-core low power Xtensa® 32-bit LX6
ROM	448 KBytes
SRAM	520 KBytes
SRAM trong RTC SLOW	8 KBytes
SRAM trong RTC FAST	8 KBytes
EFUSE	1 Kbit
MAC	256 bits
Wi-Fi	802.11 b/g/n/d/e/i/k/r (802.11n up to 150 Mbps)
Bluetooth	v4.2 BR/EDR and BLE specification
Chế độ Wi-Fi	Station/softAP/SoftAP+station/P2P
Bảo mật	WPA/WPA2/WPA2-Enterprise/WPS
Mã hóa	AES/RSA/ECC/SHA
Giao tiếp	SD-card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, GPIO, capacitive touch sensor, ADC, DAC, Hall sensor, temperature sensor
Kích thước	18 x 25.5 x 3.1mm

5.2. Lưu đồ giải thuật của chương trình nhúng.



6. Lập trình giao diện

6.1. Giới thiệu công cụ Visual Studio

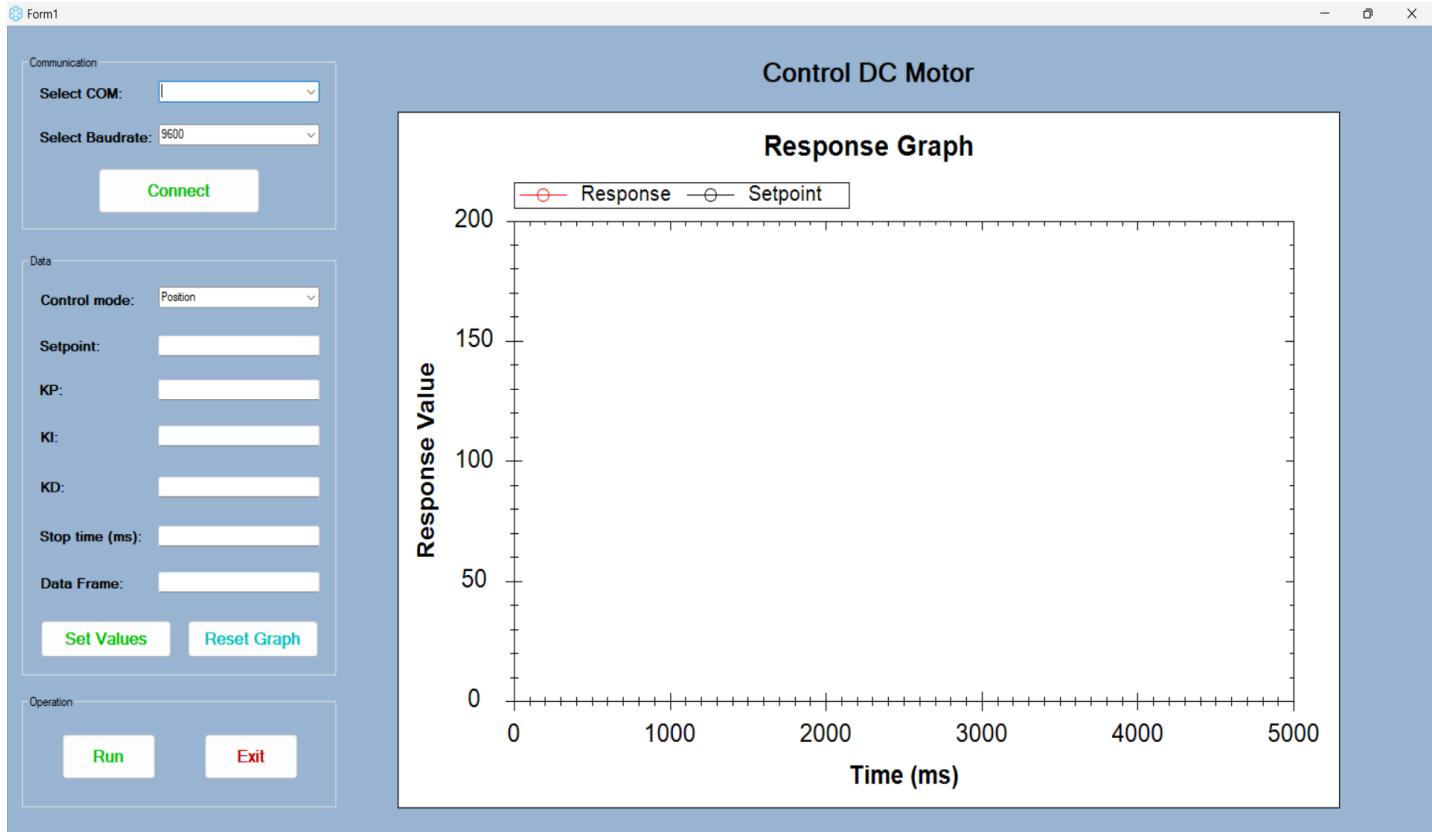
Visual Studio là một trình soạn thảo mã nguồn mở và miễn phí được phát triển bởi Microsoft. Mặc dù được thiết kế chủ yếu cho việc phát triển ứng dụng web, nhưng Visual Studio cũng hỗ trợ nhiều ngôn ngữ lập trình và nền tảng khác nhau.

WinForms là một bộ công cụ lập trình giao diện người dùng (GUI) cho các ứng dụng Windows trong .NET Framework. Nó cung cấp các thành phần và điều khiển để tạo giao diện người dùng trực quan và dễ sử dụng cho ứng dụng Windows.

Ở đồ án này, nhóm sử dụng ngôn ngữ C# để tạo giao diện trực quan và hiệu quả.

6.2. Cấu trúc chương trình giao diện

Giao diện có các khôi: khôi Communication, khôi Data, khôi Operation và đồ thị.



6.2.1. Khối Communication

Chức năng của khối này là cho phép người dùng chọn chế độ kết nối hoặc ngắt kết nối với thiết bị, đồng thời chọn cổng COM của thiết bị và baudrate giao tiếp.

Khi người dùng chọn kết nối, máy tính mở cổng và gửi một lệnh kết nối tới thiết bị. Khi kết nối đã được thiết lập thành công, dòng chữ “Connected” sẽ xuất hiện. Ngược lại, khi người dùng chọn ngắt kết nối, dòng chữ “Disconnected” sẽ xuất hiện.

6.2.2. Khối Data

Khối này cho phép người dùng chọn chế độ điều khiển (vị trí hoặc vận tốc), setpoint, các thông số của bộ điều khiển PID (Kp, Ki, Kd), và thời gian kết thúc của một lần Motor chạy (Stop time).

Khi người dùng nhấn nút "Set Values" trên máy tính, máy tính gửi một chuỗi dữ liệu đã được quy định sẵn tới ESP32. Chuỗi dữ liệu này sẽ được hiển thị lên ô “Data Frame”, ta đọc ô này để kiểm tra và xác nhận lại tính đúng đắn của dữ liệu.

Khi người dùng nhấn nút “Reset Graph”, các điểm và đường đã được vẽ trên đồ thị sẽ bị xóa để chuẩn bị cho lần vẽ đồ thị tiếp theo.

6.2.3. Khối Operation

Khối này cho phép người dùng ra lệnh bắt đầu chạy Motor (nút “Run”) hoặc thoát khỏi giao diện (nút “Exit”).

Khi người dùng nhấn nút "Run" trên máy tính, máy tính gửi một chuỗi dữ liệu đến ESP32. ESP32 khi nhận được chuỗi dữ liệu này sẽ bật cờ lệnh tương ứng và bắt đầu cho động cơ chạy, đồng thời khởi động các khối PID và PWM để điều khiển động cơ. Động cơ sẽ được chạy cho đến khi bằng Stop time đã set.

Khi người dùng nhấn nút “Exit” trên máy tính, máy tính sẽ tắt giao diện.

6.2.4. Đồ thị

Khối này cho phép người dùng thu thập dữ liệu và vẽ đồ thị setpoint, vị trí/tốc độ của động cơ khi đang hoạt động. Khi người dùng nhấn nút "Run" ở khía Operation, đồ thị sẽ bắt đầu vẽ.

Khi ESP32 gửi dữ liệu, GUI sẽ tách chuỗi, lấy dữ liệu và vẽ lên đồ thị. Trục "Time" của đồ thị được tự động điều chỉnh khi người dùng nhập "Stop time", nên dữ liệu sẽ được vẽ đúng và đầy đủ. Trục "Response Value" cũng sẽ được tự động điều chỉnh bằng cách cộng lên 200 so với setpoint để dễ dàng quan sát các đáp ứng vọt lố. Khi người dùng nhấn nút "Reset Graph" ở khía Data, các điểm và đường đã được vẽ trên đồ thị sẽ bị xóa để chuẩn bị cho lần vẽ tiếp theo.

7. Phương thức trao đổi dữ liệu giữa ESP32 và GUI

1. Dữ liệu nhận được từ GUI có 2 dạng: "M...S...P...I...D...E...T" và "RUN"

a) Chuỗi "M...S...P...I...D...E...T" được sử dụng để cung cấp cho ESP32 chế độ điều khiển, tham số PID, thời gian dừng, và giá trị setpoint.

Dữ liệu ... giữa 'M' và 'S' đại diện cho chế độ điều khiển: '1' cho điều khiển vị trí, '2' cho điều khiển tốc độ.

Dữ liệu ... giữa 'S' và 'P' đại diện cho giá trị setpoint.

Dữ liệu ... giữa 'P' và 'I' đại diện cho giá trị KP.

Dữ liệu ... giữa 'I' và 'D' đại diện cho giá trị KI.

Dữ liệu ... giữa 'D' và 'E' đại diện cho giá trị KD.

Dữ liệu ... giữa 'E' và 'T' đại diện cho giá trị thời gian dừng.

Ví dụ: "M2S100P3I5D0E10T" chứa:

Chế độ điều khiển '2' cho điều khiển tốc độ

Giá trị setpoint = 100 (vòng/phút vì đây là điều khiển tốc độ)

Giá trị KP = 3

Giá trị KI = 5

Giá trị KD = 0

Giá trị thời gian dừng = 10 giây.

b) Chuỗi "RUN" được sử dụng để chỉ dẫn ESP32 khởi động động cơ.

2. Dữ liệu truyền từ ESP32 đến GUI có định dạng: "@...&...#"

Giá trị giữa '@' và '&' đại diện cho giá trị của setpoint.

Giá trị giữa '&' và '#' đại diện cho giá trị của đáp ứng ở thời điểm hiện tại.

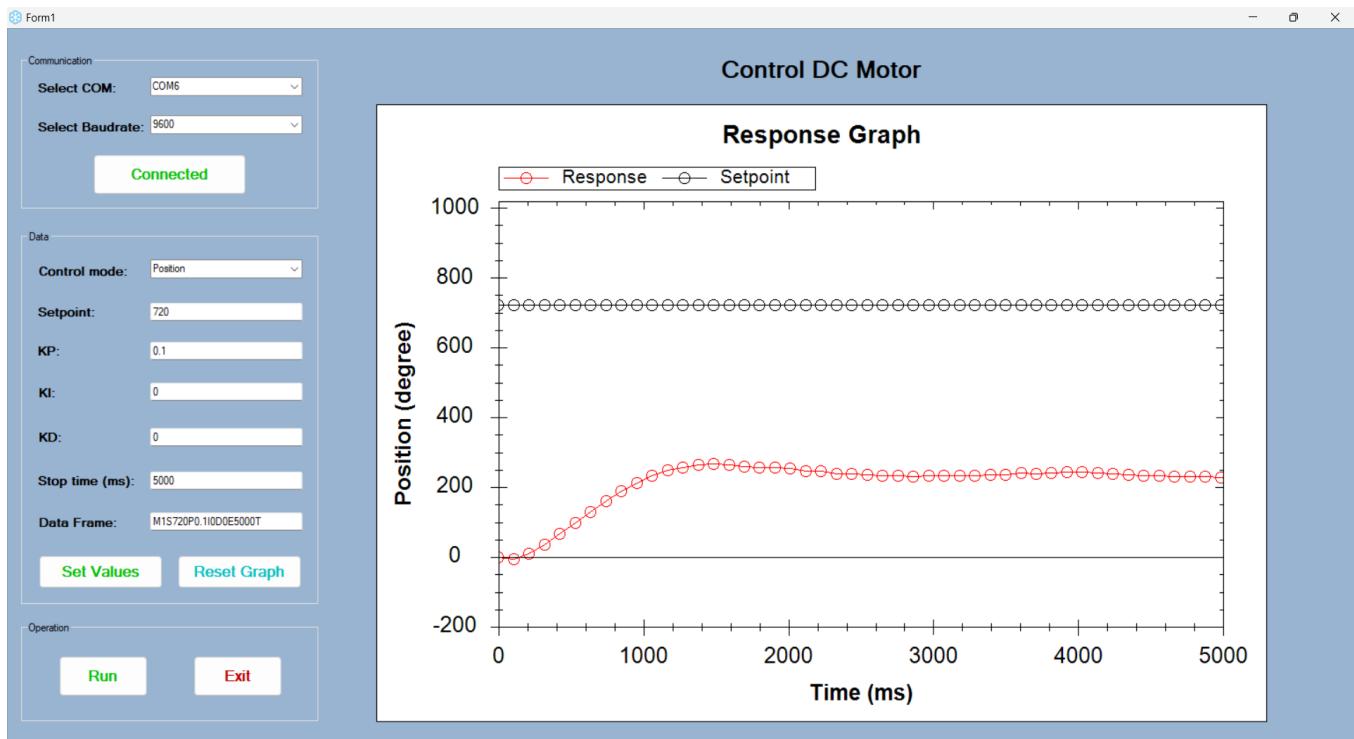
GUI sẽ phân tích khung này, trích xuất dữ liệu, vẽ biểu đồ đáp ứng và setpoint.

8. Đánh giá kết quả thực hiện, chất lượng hệ thống

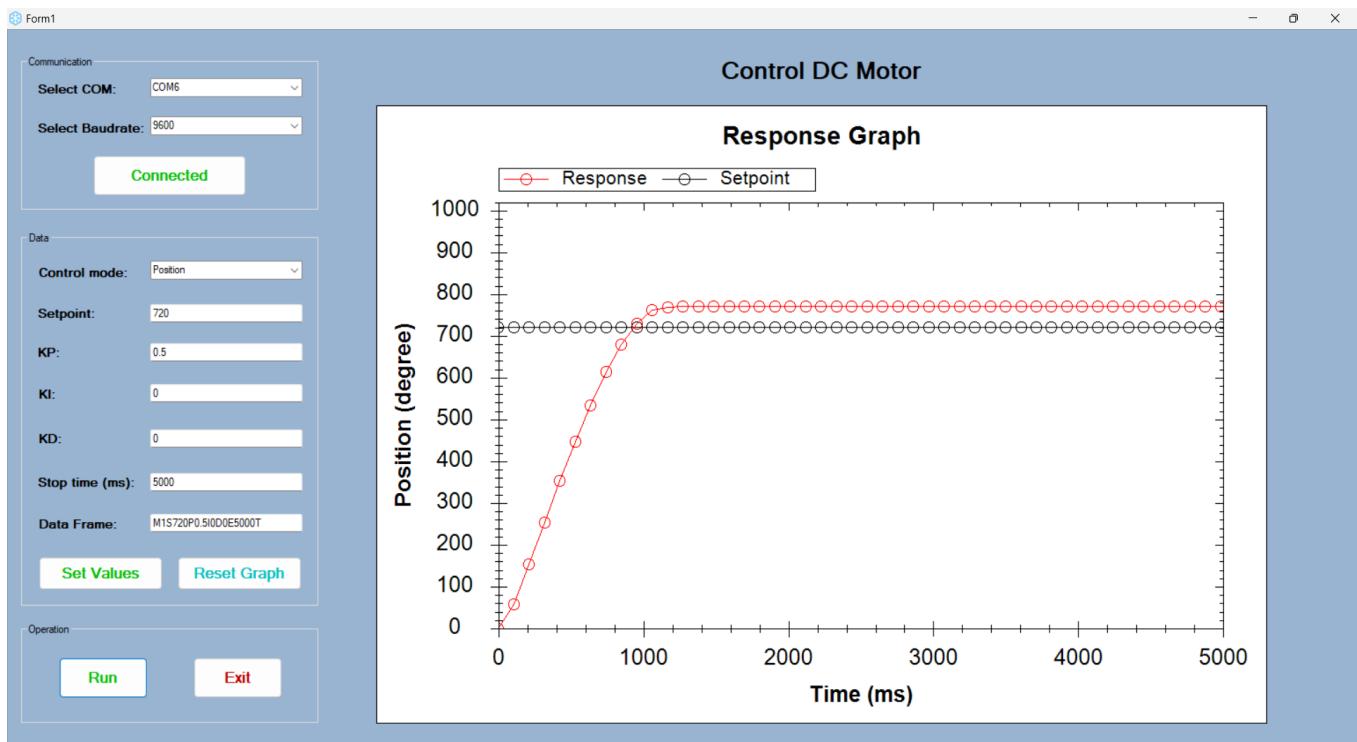
8.1. Điều khiển vị trí

Vị trí đặt là 720 độ, tương ứng với 2 vòng quay của Motor.

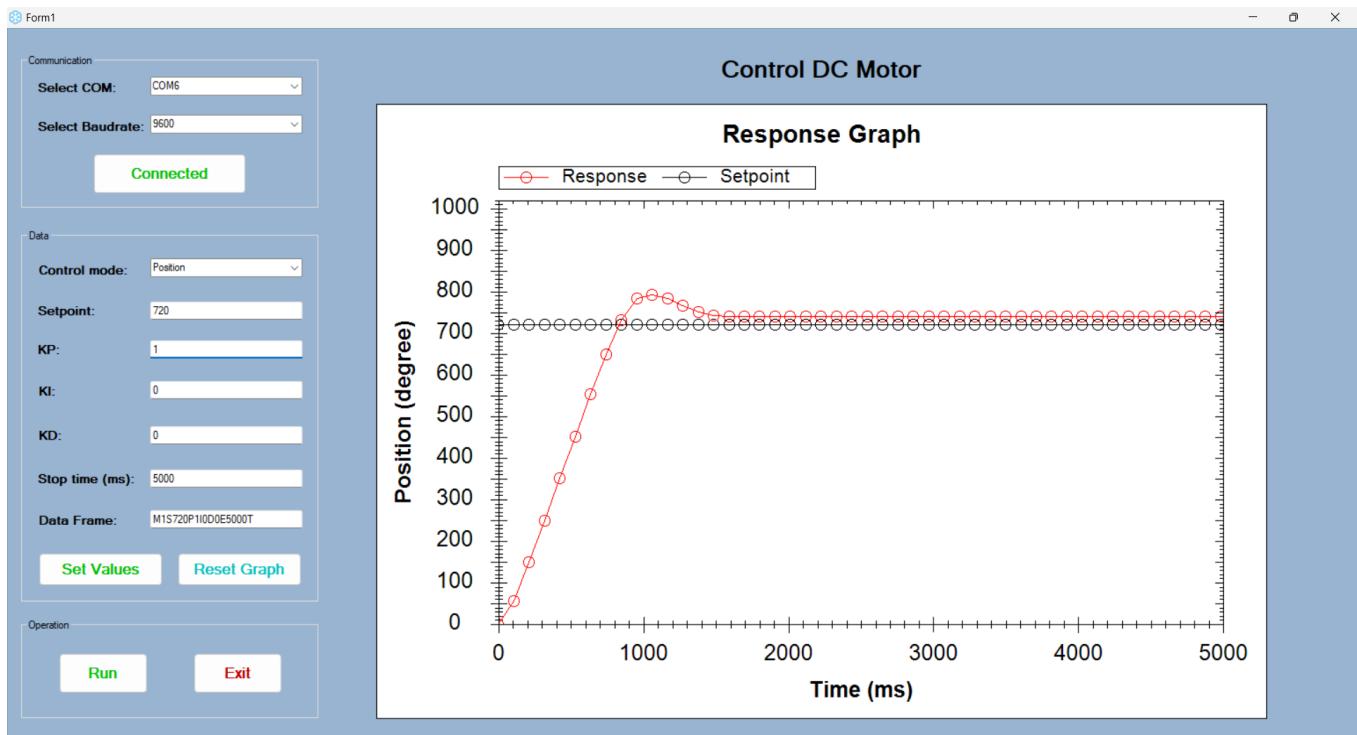
8.1.1. Thay đổi Kp (Bộ điều khiển P)



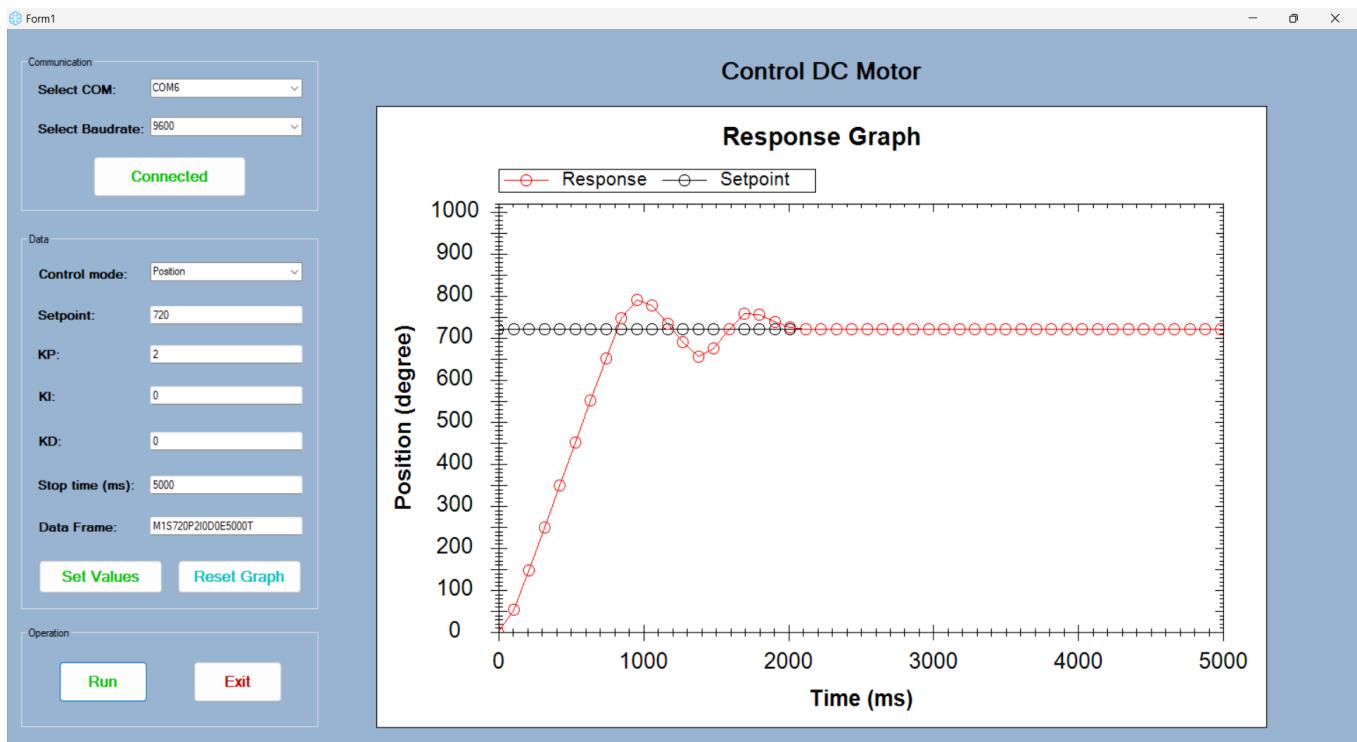
Đáp ứng vị trí khi $K_p = 0.1, K_i = 0, K_d = 0$



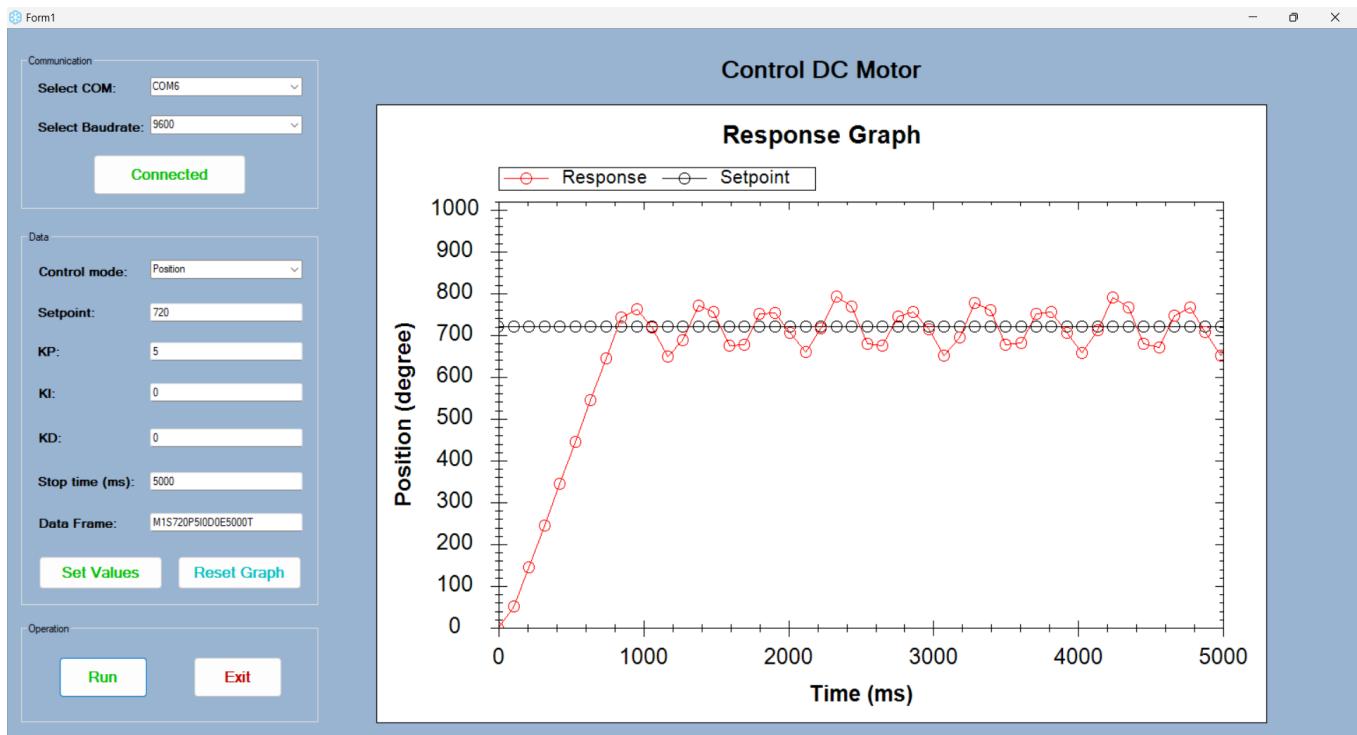
Đáp ứng với tri Kp = 0.5, Ki = 0, Kd = 0



Đáp ứng với tri Kp = 1, Ki = 0, Kd = 0

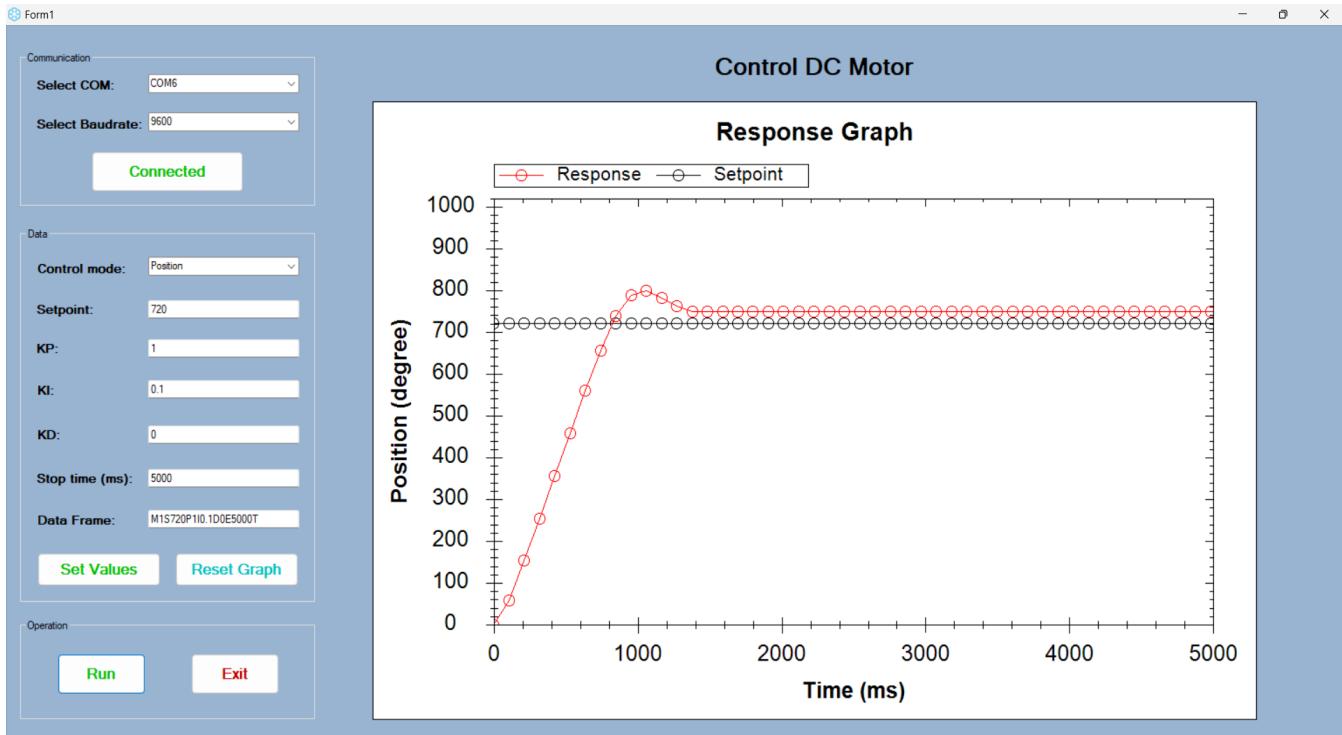


Đáp ứng vị trí khi $K_p = 2, K_i = 0, K_d = 0$

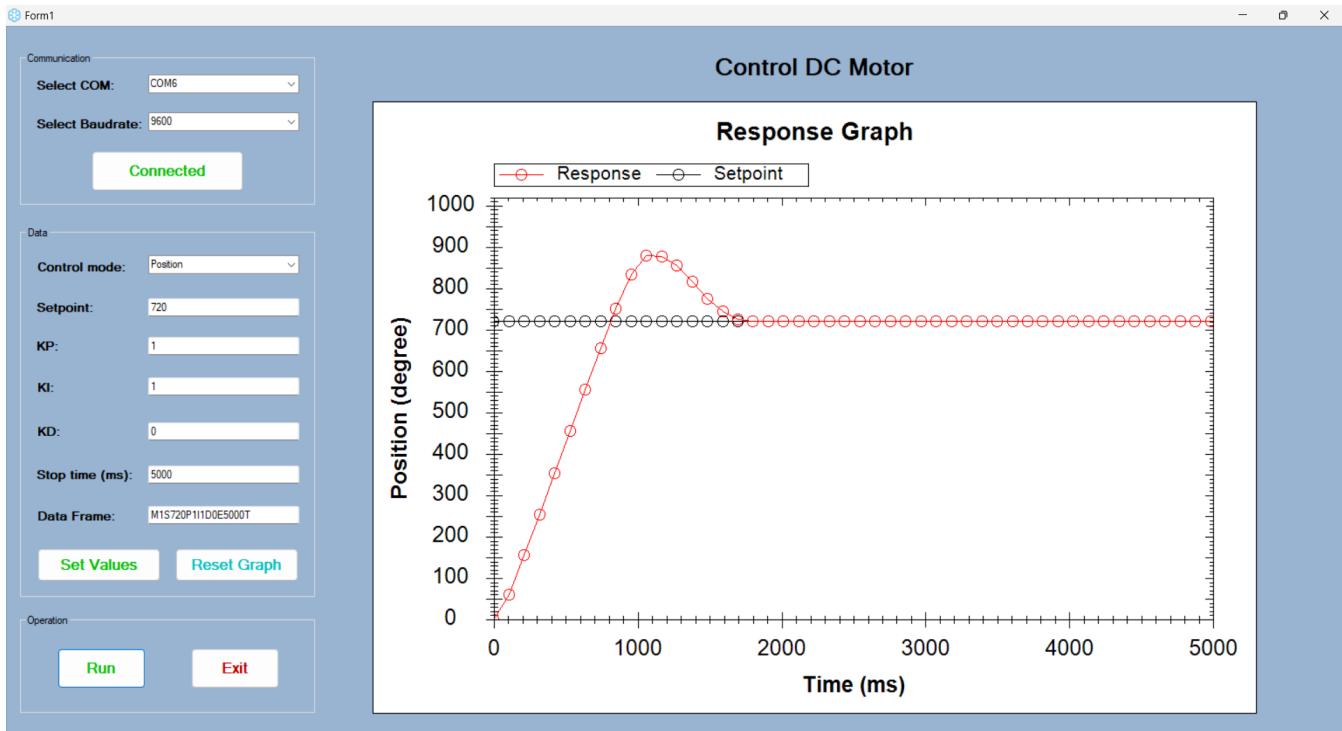


Đáp ứng vị trí khi $K_p = 5, K_i = 0, K_d = 0$

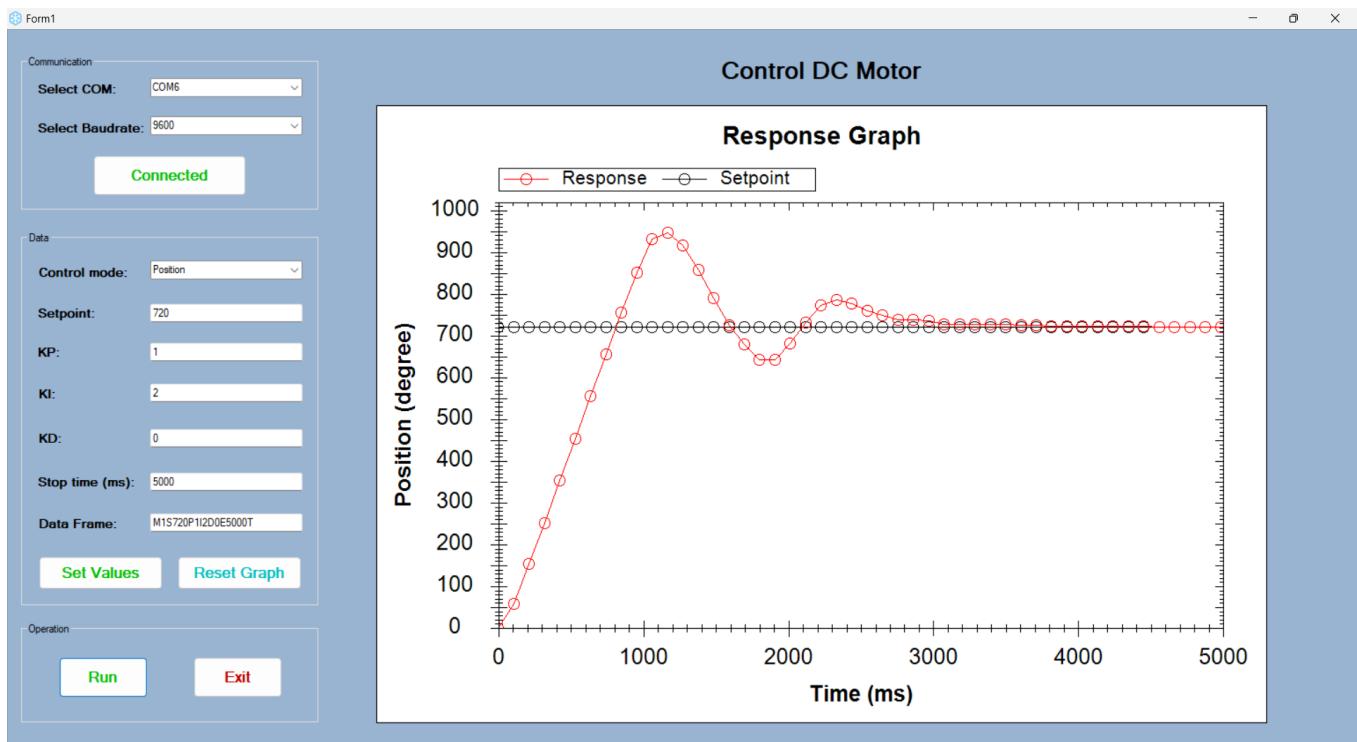
8.1.2. Thay đổi K_i (Bộ điều khiển PI)



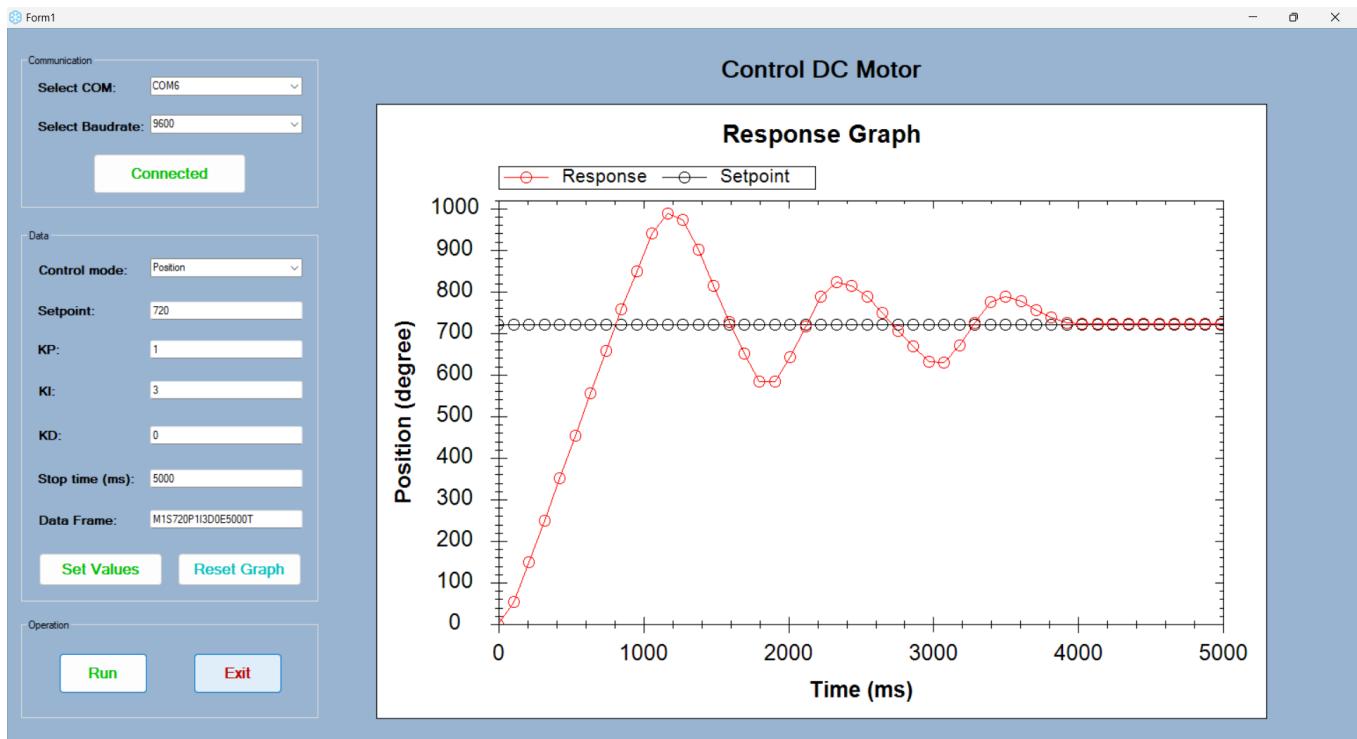
Đáp ứng vị trí khi $K_p = 1, K_i = 0.1, K_d = 0$



Đáp ứng vị trí khi $K_p = 1, K_i = 1, K_d = 0$

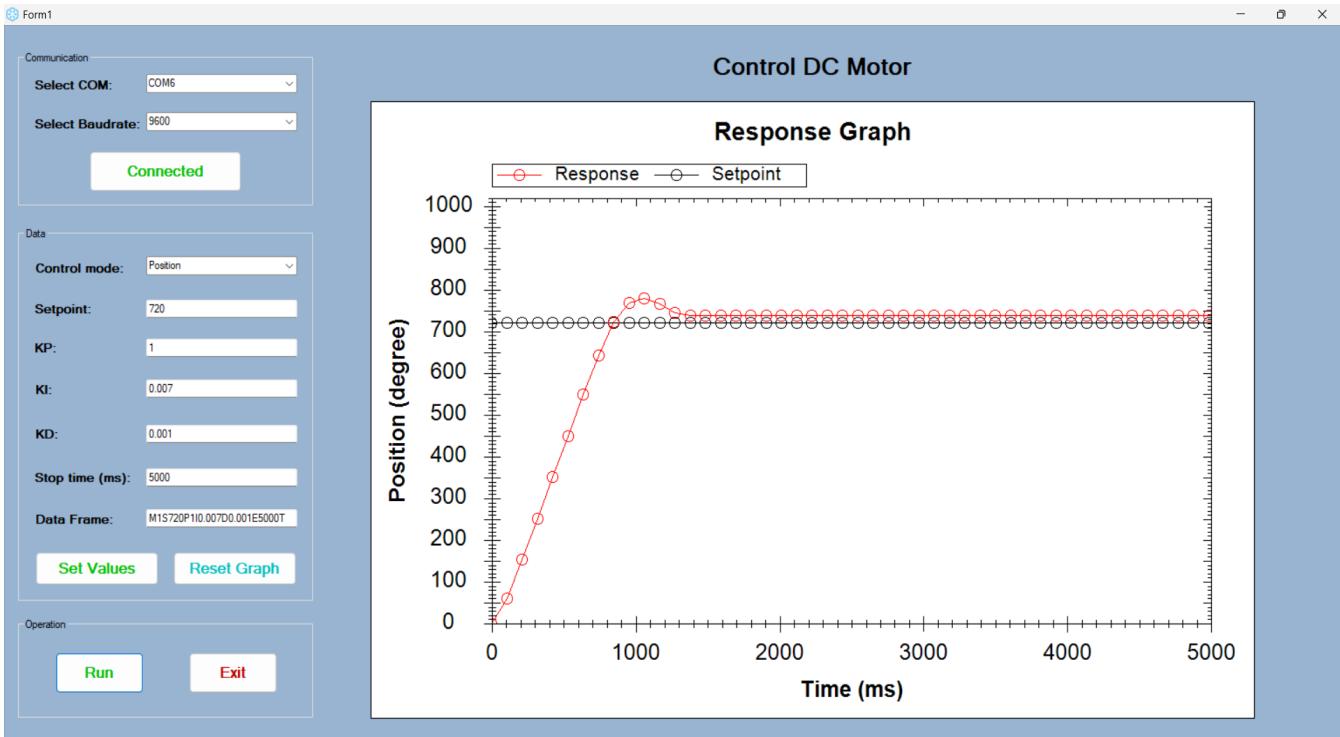


Đáp ứng với tri khi $K_p = 1, K_i = 2, K_d = 0$

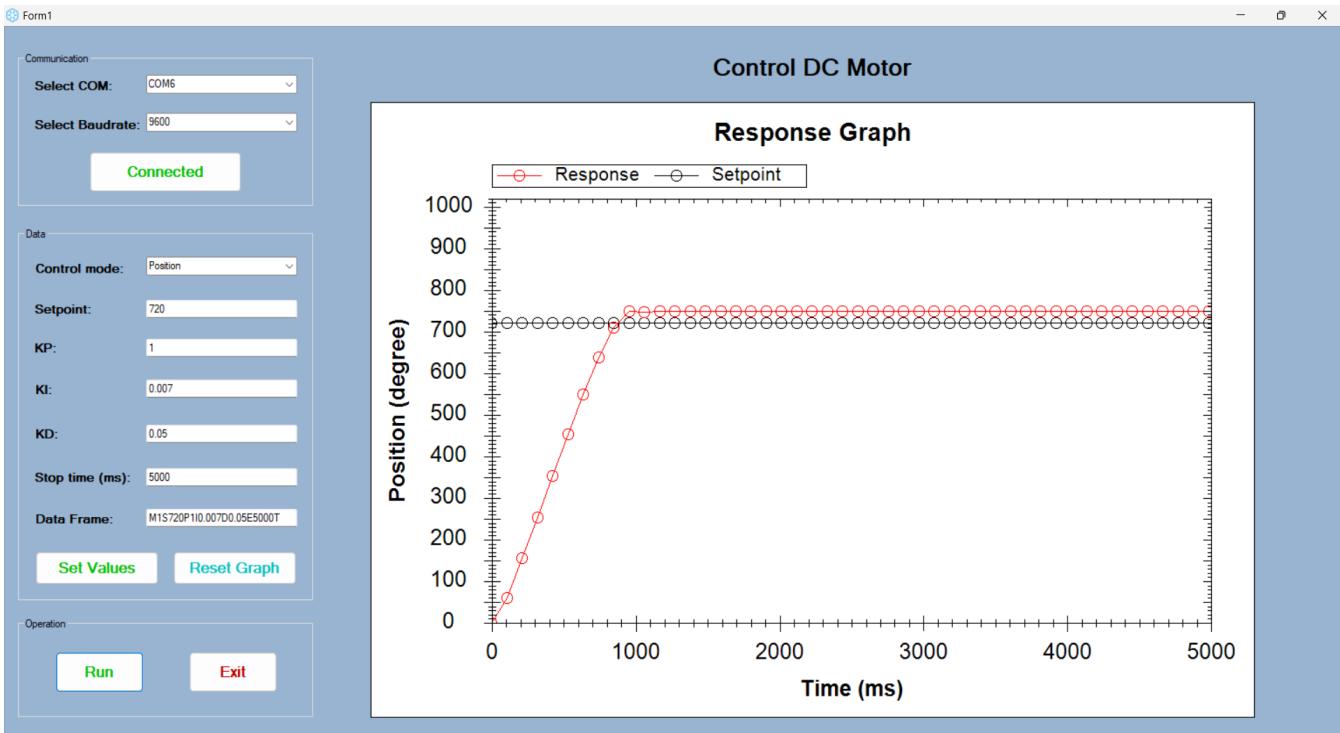


Đáp ứng với tri khi $K_p = 1, K_i = 3, K_d = 0$

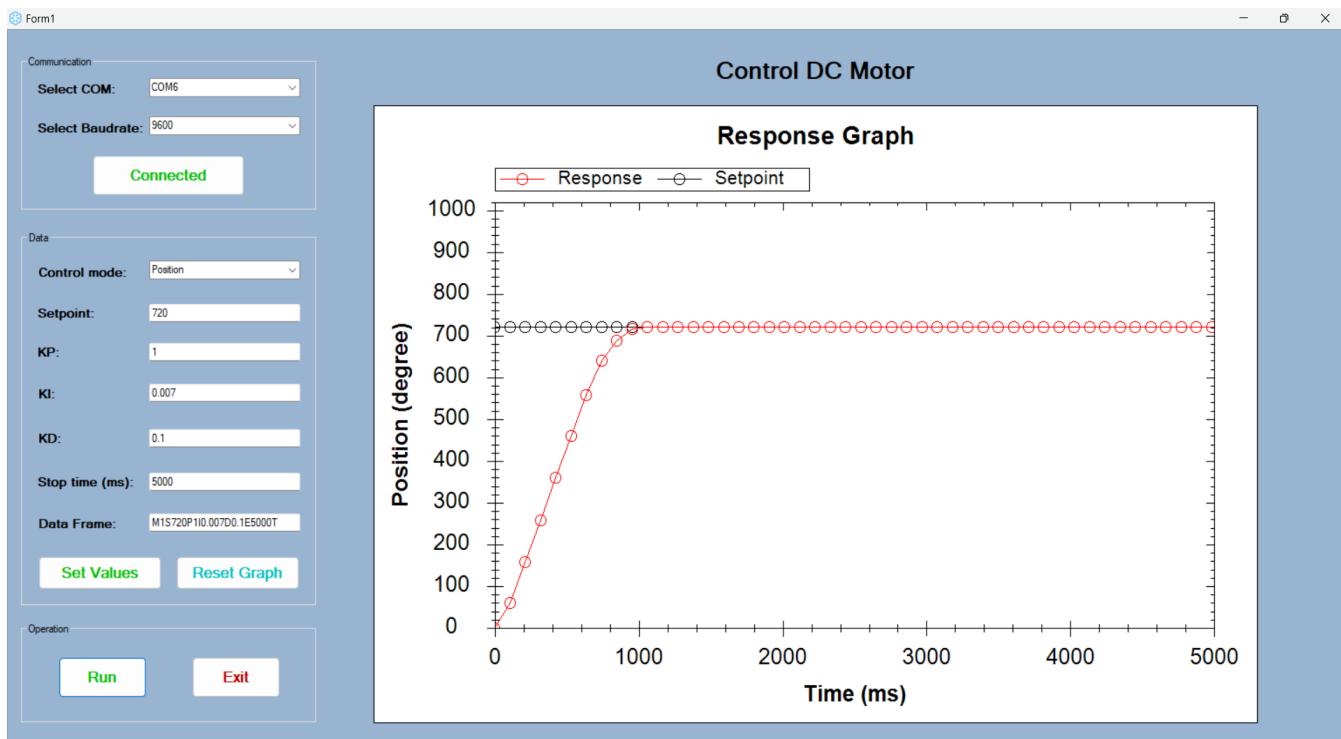
8.1.3. Thay đổi Kd (Bộ điều khiển PID)



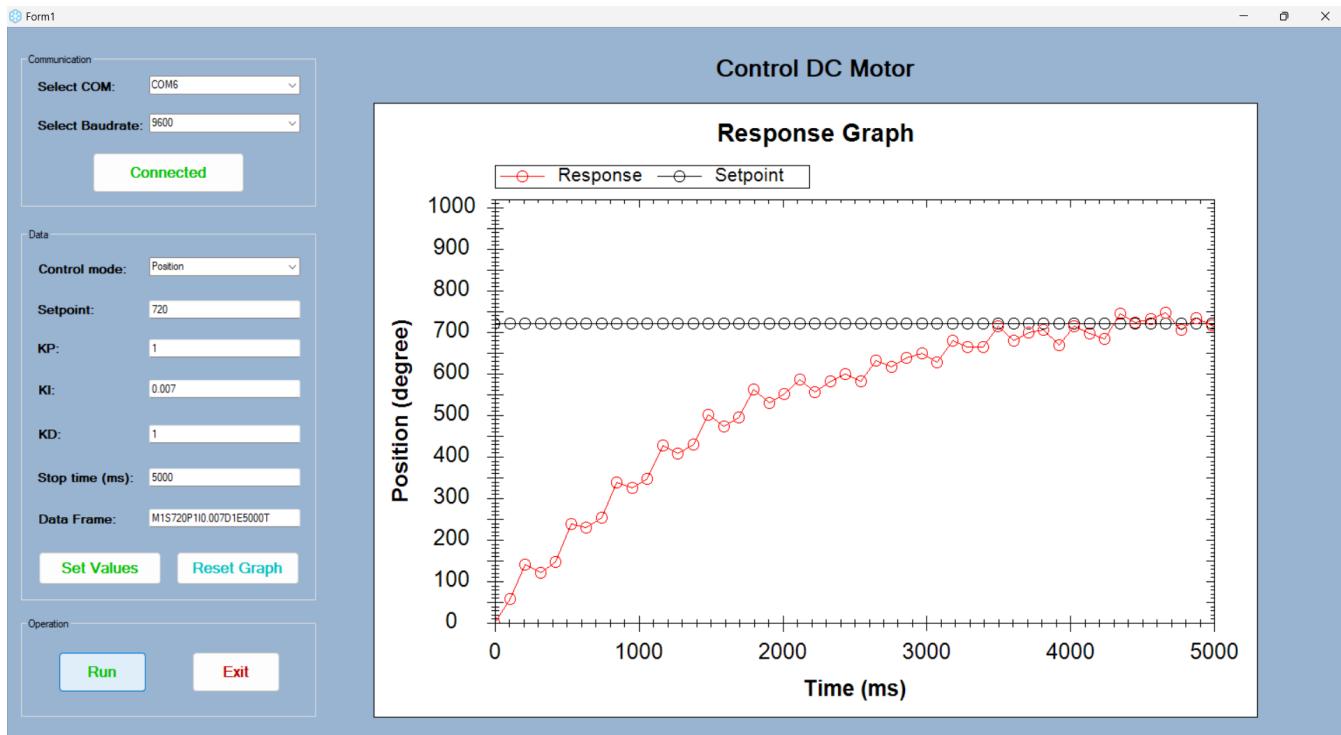
Dáp ứng vị trí khi $K_p = 1, K_i = 0.007, K_d = 0.001$



Dáp ứng vị trí khi $K_p = 1, K_i = 0.007, K_d = 0.05$



Đáp ứng với tri Kp = 1, Ki = 0.007, Kd = 0.1



Đáp ứng với tri Kp = 1, Ki = 0.007, Kd = 1

8.1.4. Nhận xét

Kp	Ki	Kd	Độ vọt lồ (%)	T/g quá độ (s)	S/s xác lập (độ)
0.1	0	0	0	2.226	495
0.5	0	0	0	1	42.75
1	0	0	7.72	1.1	18.75
2	0	0	9.06	1.5	0
5	0	0	Không xác định	Không xác định	Không xác định
1	0.1	0	9.58	1.2	15.75
1	1	0	22.08	1.55	0
1	2	0	31.46	2.5	0
1	3	0	37.81	3.65	0
1	0.007	0.001	5.48	1.2	18.75
1	0.007	0.05	0	0.85	18
1	0.007	0.1	0	0.8	0
1	0.007	1	Không xác định	Không xác định	Không xác định

Ảnh hưởng của các thông số trong điều khiển vị trí

Từ bảng số liệu, ta nhận thấy:

Khi Kp tăng: thời gian quá độ và sai số xác lập giảm, độ vọt lồ tăng. Hệ sẽ mất ổn định khi Kp quá lớn.

Khi Ki tăng: sai số xác lập bị triệt tiêu, tuy nhiên độ vọt lồ và thời gian quá độ tăng.

Khi Kd tăng: cả độ vọt lồ, thời gian quá độ và sai số xác lập đều giảm. Hệ sẽ mất ổn định khi Kd quá lớn.

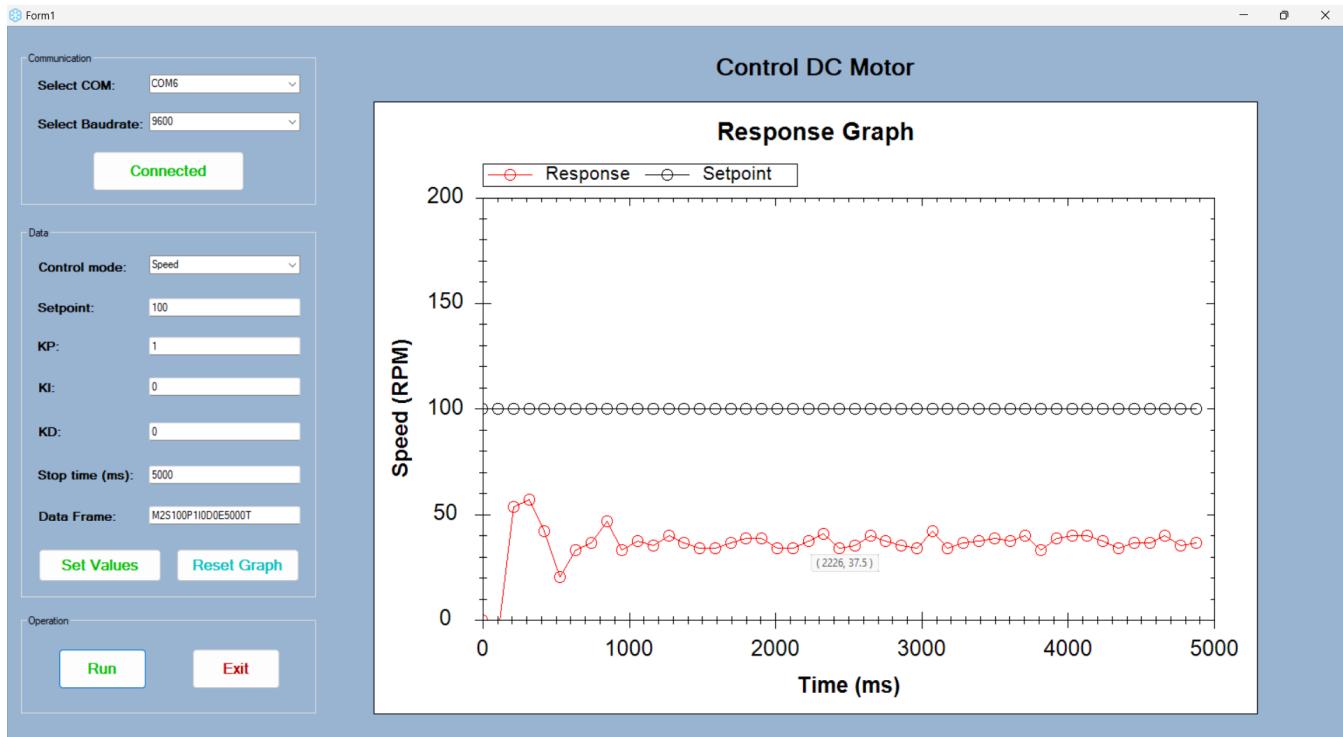
Thông số cho ra đáp ứng tốt nhất trong chế độ điều khiển vị trí là trường hợp:

$K_p = 1, K_i = 0.007, K_d = 0.1$.

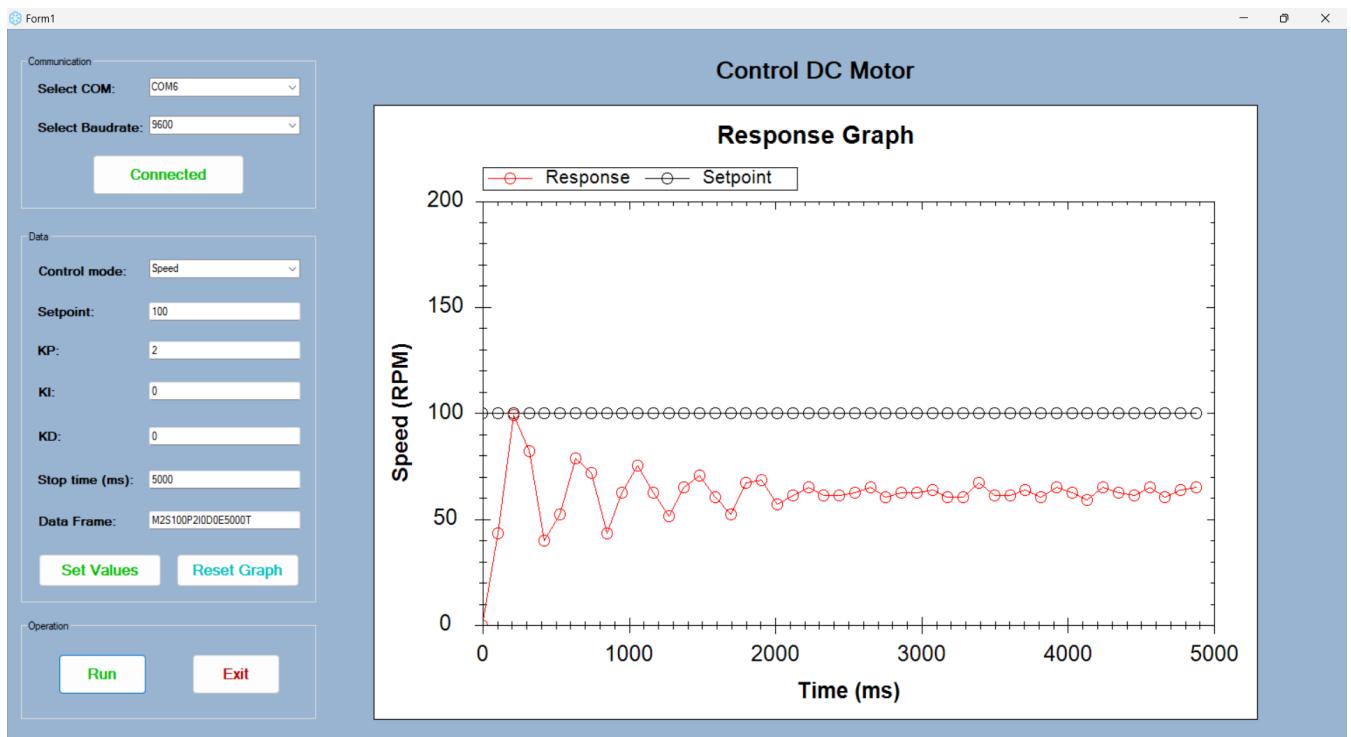
8.2. Điều khiển vận tốc

Vận tốc đặt là 100 RPM

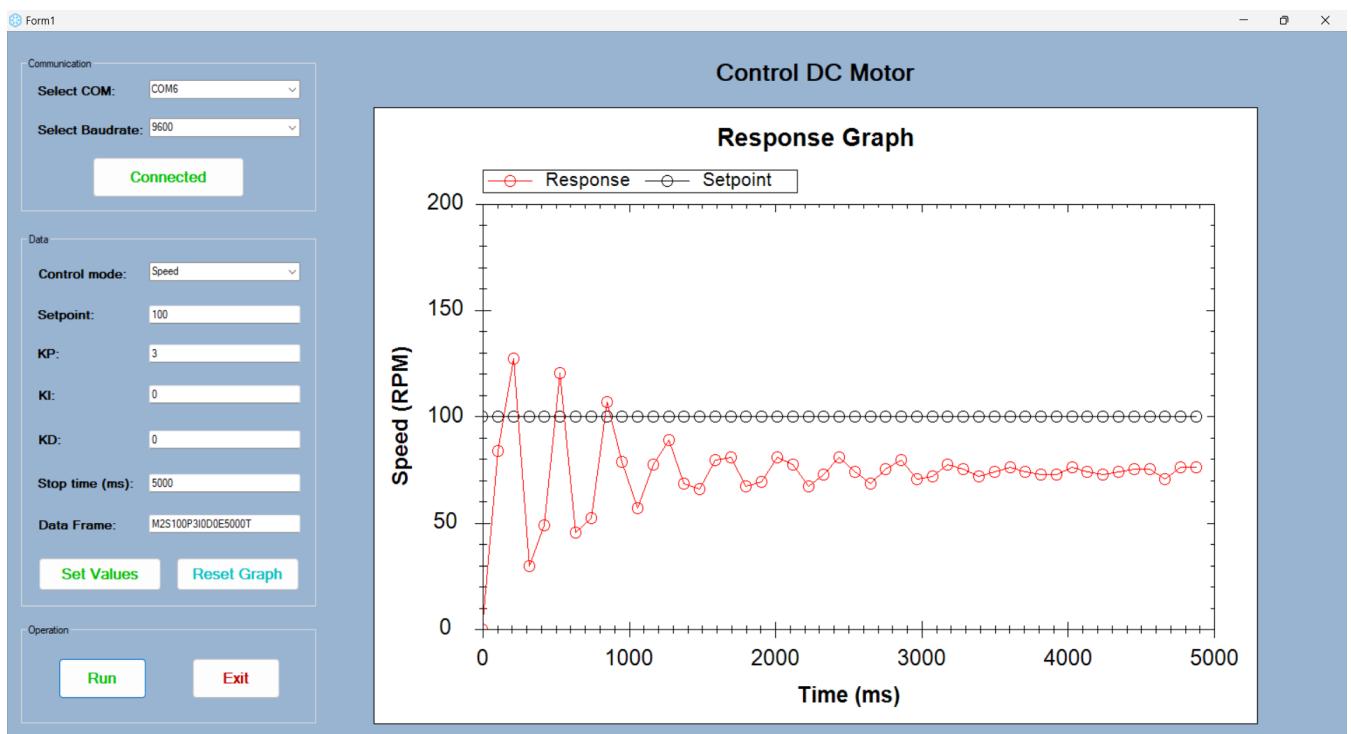
8.2.1. Thay đổi Kp (bộ điều khiển P)



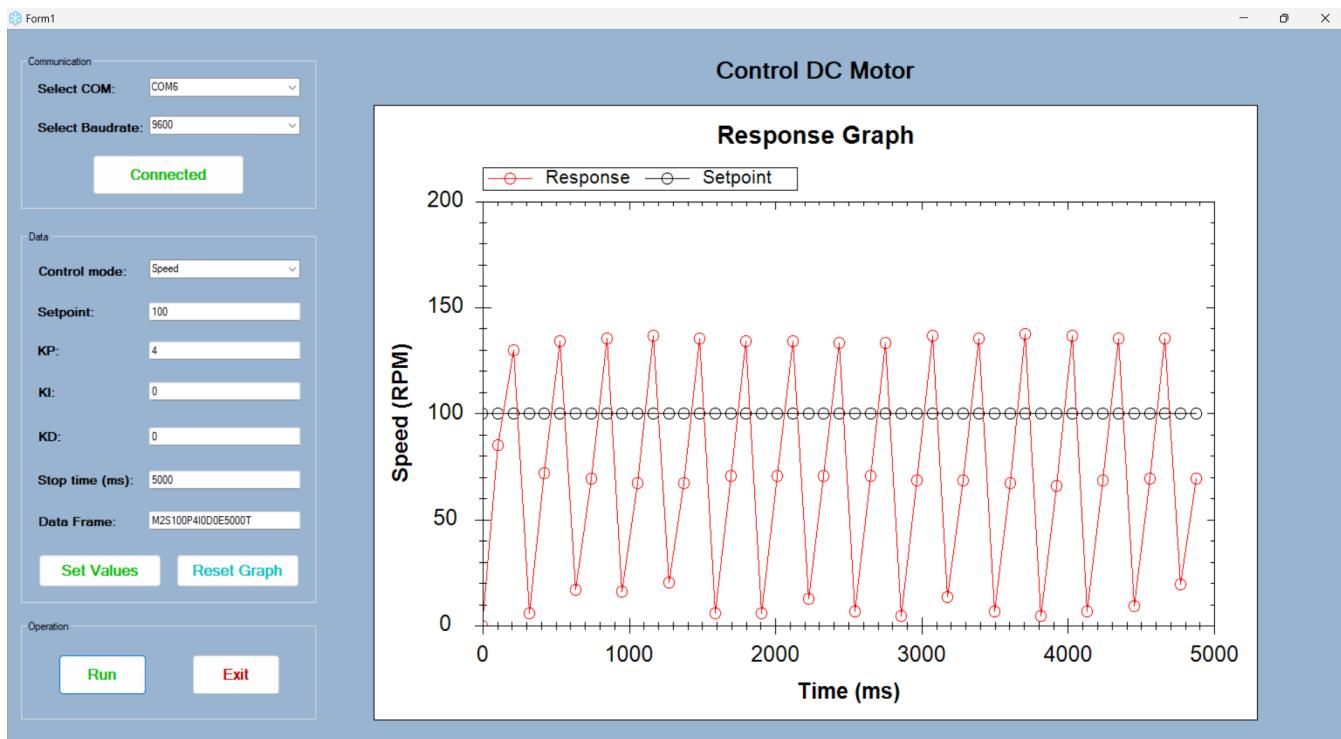
Đáp ứng vận tốc khi $K_p = 1, K_i = 0, K_d = 0$



Đáp ứng vận tốc khi $K_p = 2, K_i = 0, K_d = 0$

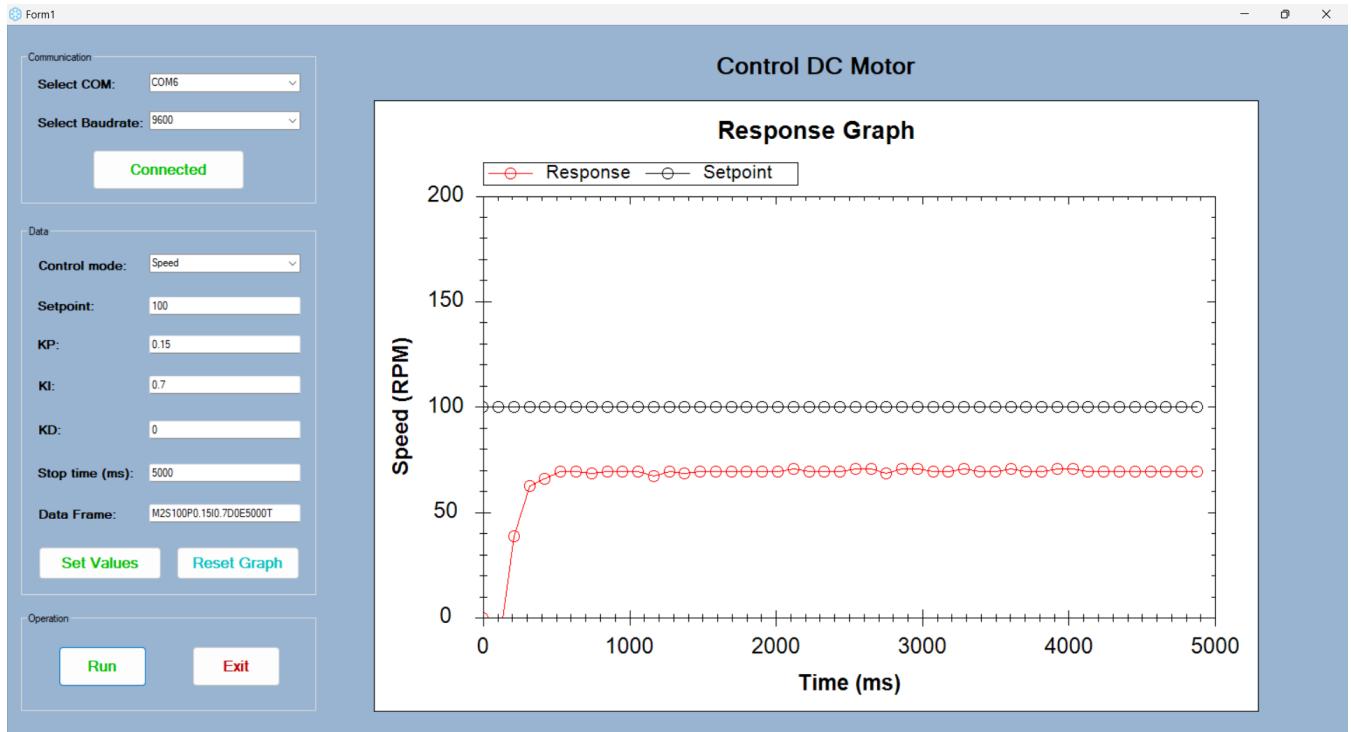


Đáp ứng vận tốc khi $K_p = 3, K_i = 0, K_d = 0$

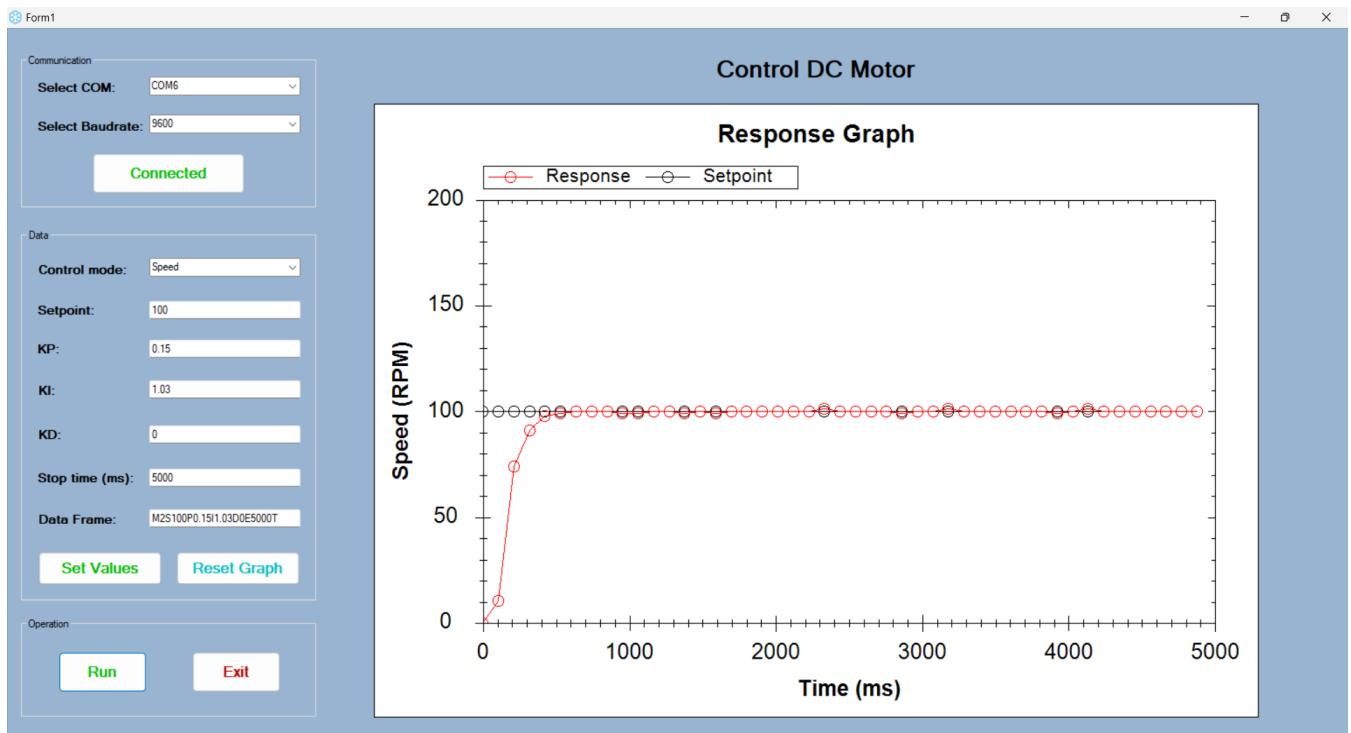


Đáp ứng vận tốc khi $K_p = 4, K_i = 0, K_d = 0$

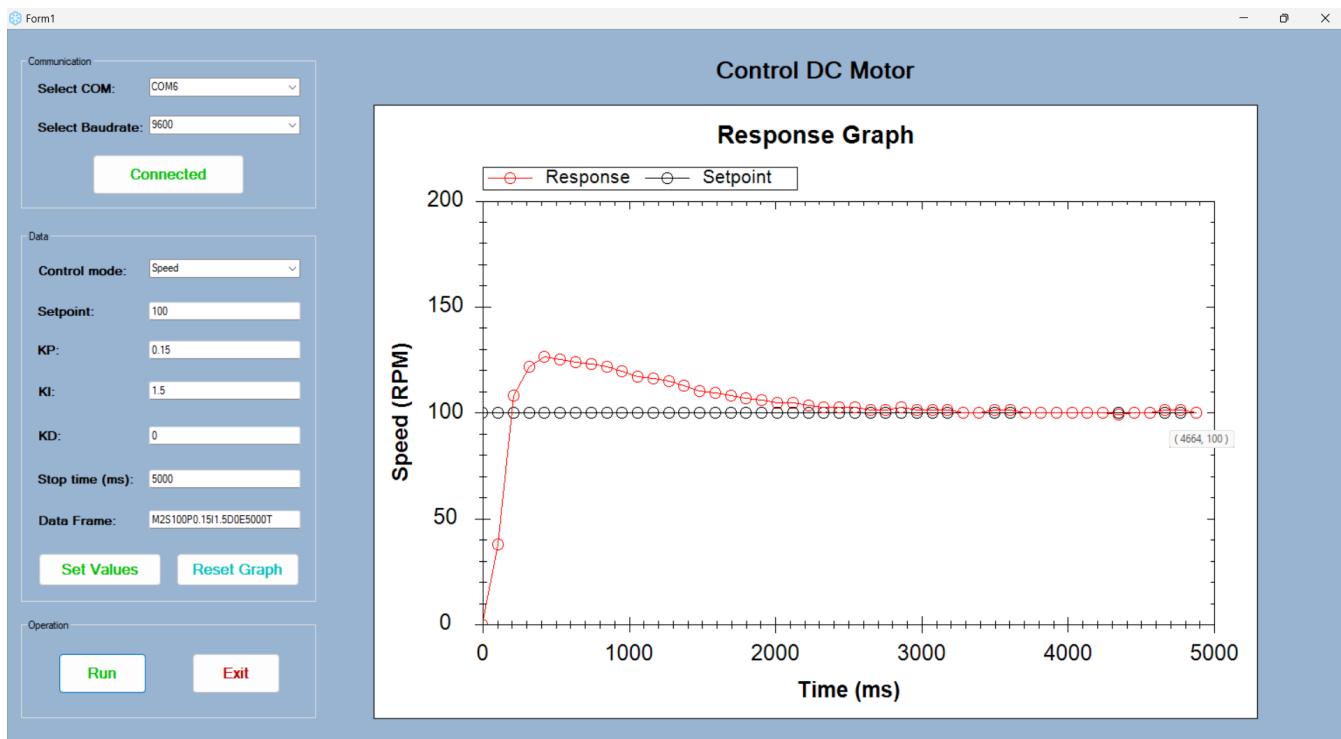
8.2.2. Thay đổi Ki (bộ điều khiển PI)



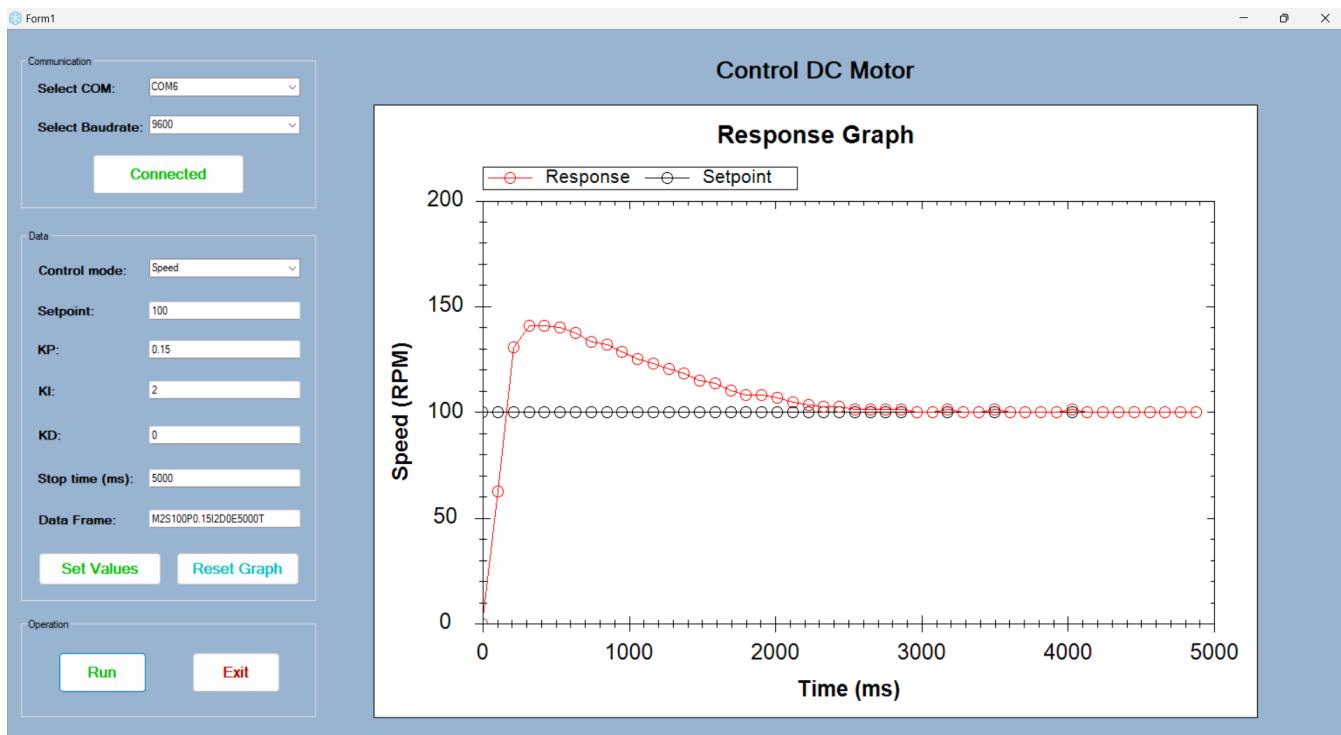
Đáp ứng vận tốc khi $K_p = 0.15$, $K_i = 0.7$, $K_d = 0$



Đáp ứng vận tốc khi $K_p = 0.15$, $K_i = 1.03$, $Kd = 0$

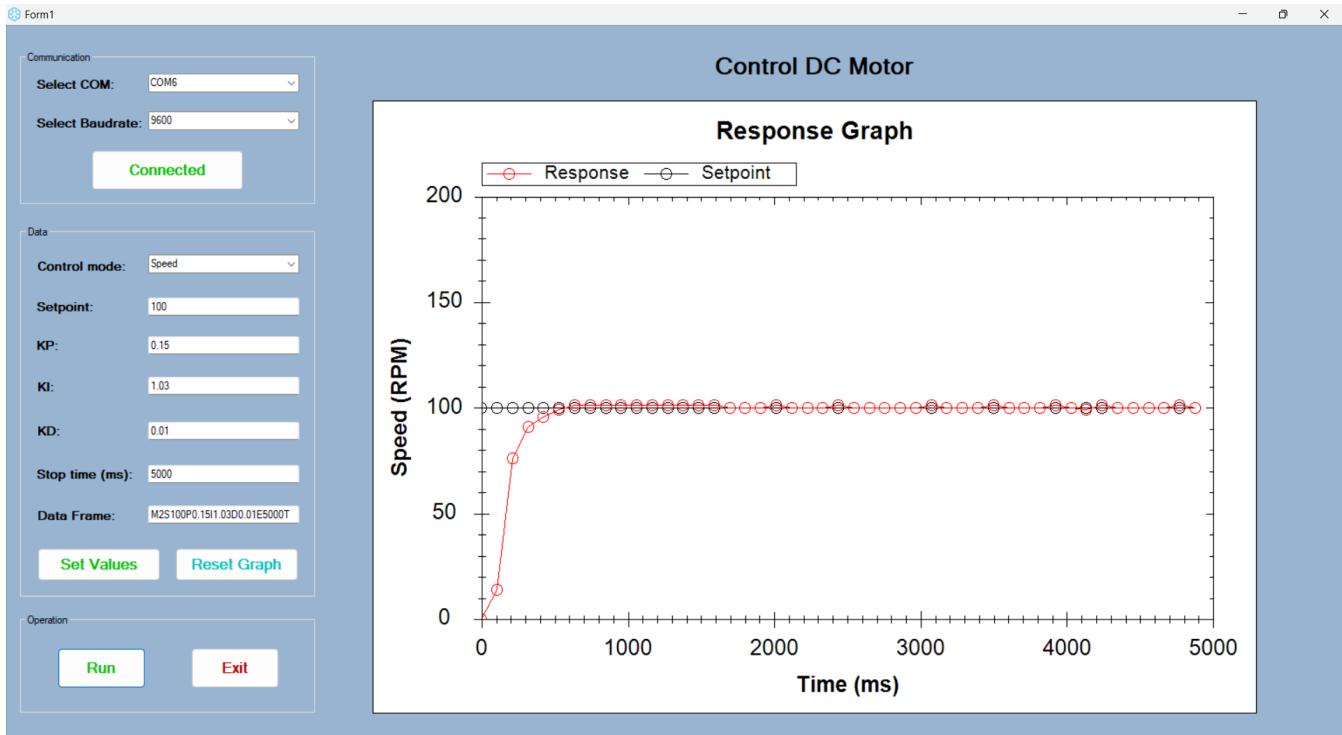


Đáp ứng vận tốc khi $K_p = 0.15, K_i = 1.5, K_d = 0$

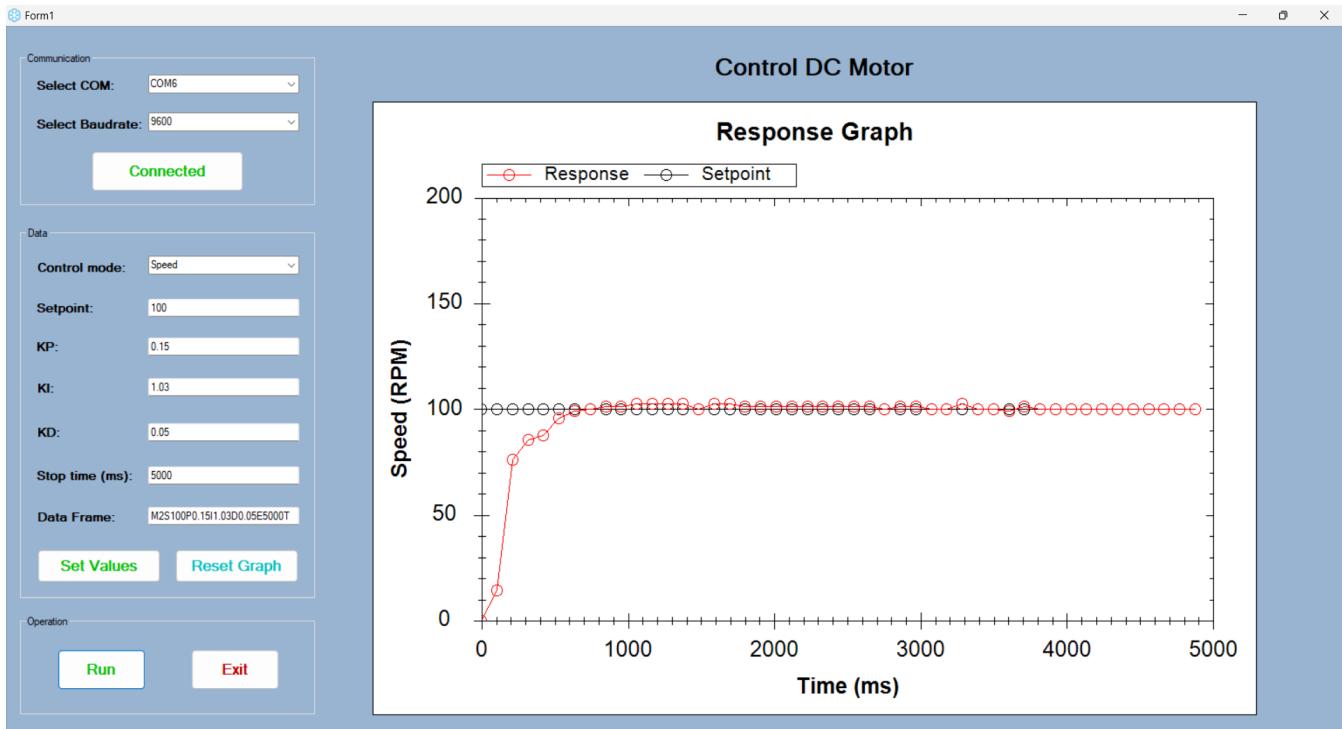


Đáp ứng vận tốc khi $K_p = 0.15, K_i = 2, K_d = 0$

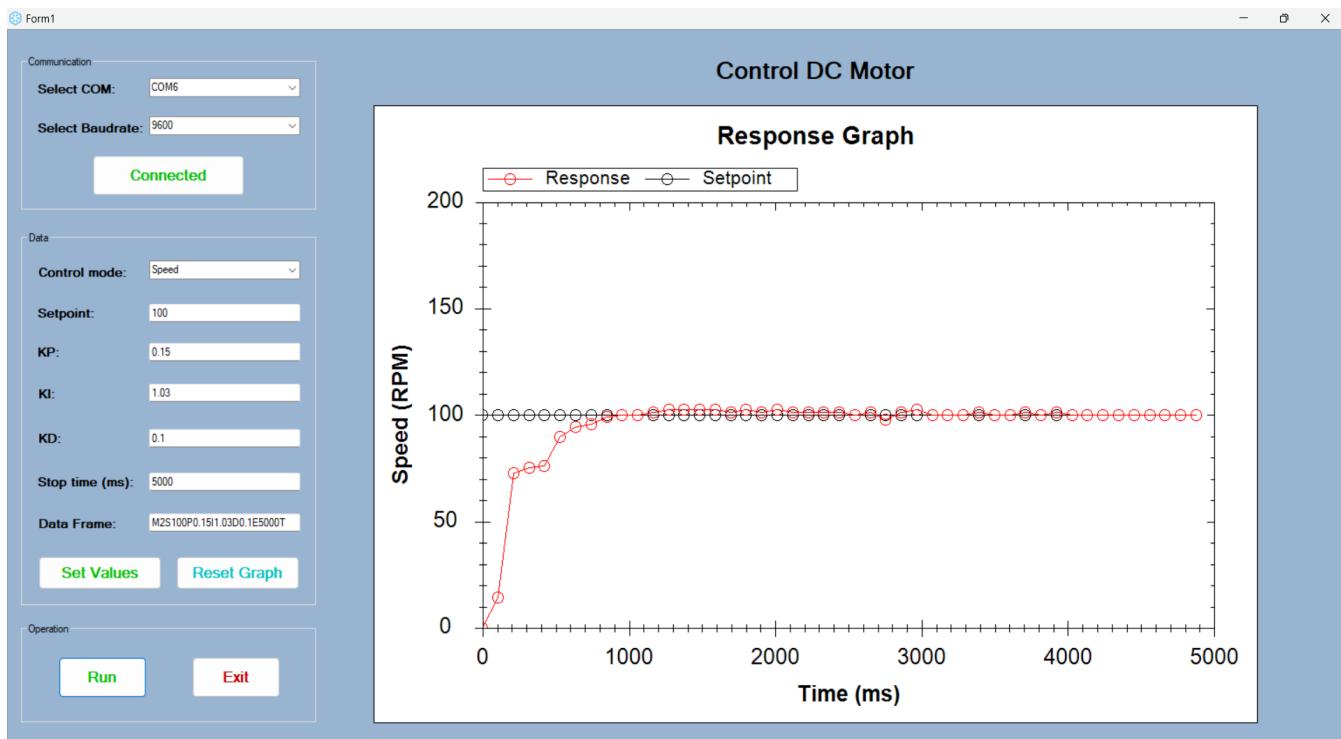
8.2.3. Thay đổi Kd (bộ điều khiển PID)



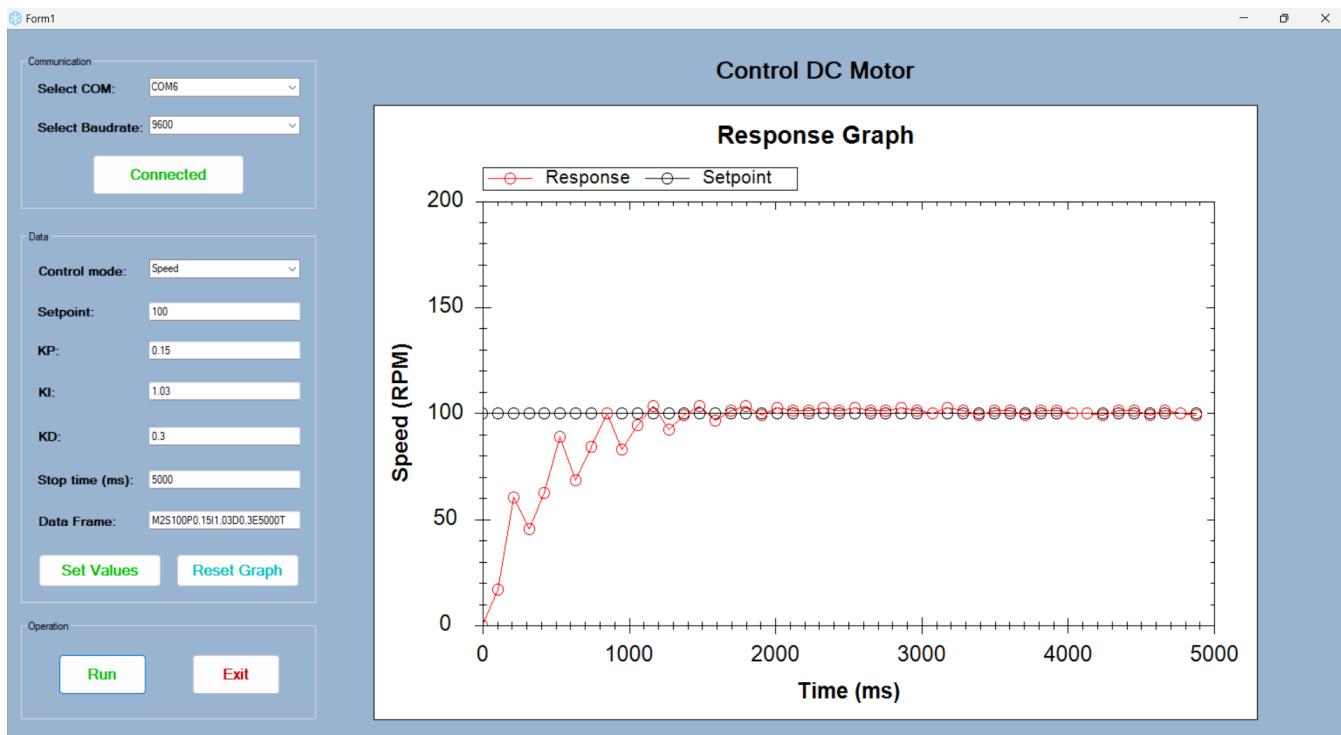
Dáp ứng vận tốc khi $K_p = 0.15, K_i = 1.03, K_d = 0.01$



Dáp ứng vận tốc khi $K_p = 0.15, K_i = 1.03, K_d = 0.05$



Đáp ứng vận tốc khi $K_p = 0.15, K_i = 1.03, K_d = 0.1$



Đáp ứng vận tốc khi $K_p = 0.15, K_i = 1.03, K_d = 0.3$

8.2.4. Nhận xét

Kp	Ki	Kd	Độ vọt lố (%)	T/g quá độ (s)	S/s xác lập (RPM)
1	0	0	51.52	1.1	62.5
2	0	0	61.11	2.1	38.64
3	0	0	69.69	3.1	25
4	0	0	Không xác định	Không xác định	Không xác định
0.15	0.7	0	0	0.4	30.68
0.15	1.03	0	0	0.38	0
0.15	1.5	0	26.14	2	0
0.15	2	0	40.91	2.12	0
0.15	1.03	0.01	0	0.41	0
0.15	1.03	0.05	0	0.5	0
0.15	1.03	0.1	0	0.72	0
0.15	1.03	0.3	0	1.3	0

Ảnh hưởng của các thông số trong điều khiển vận tốc

Từ bảng số liệu, ta nhận thấy:

Khi Kp tăng: sai số xác lập giảm nhưng không triệt tiêu hoàn toàn, độ vọt lố tăng.

Hệ sẽ mất ổn định khi Kp quá lớn.

Khi Ki tăng: sai số xác lập bị triệt tiêu, tuy nhiên độ vọt lố và thời gian quá độ tăng.

Khi Kd tăng: độ vọt lố và sai số xác lập bị triệt tiêu, thời gian quá độ tăng nhưng không đáng kể.

Thông số cho ra đáp ứng tốt nhất trong chế độ điều khiển vận tốc là trường hợp:

$$K_p = 1, K_i = 0.007, K_d = 0.1.$$

9. Kết luận chung

Qua đề tài này, nhóm em rút ra kết luận sau:

Về bộ điều khiển: bộ điều khiển PID phù hợp với đối tượng động cơ DC không tải với các thông số không thay đổi. Với việc tinh chỉnh các thông số từ thực nghiệm, nhóm em có thể điều khiển động cơ DC bằng bộ điều khiển PID cho ra đáp ứng với chất lượng tốt: thời gian đáp ứng nhanh, không xảy ra vọt lố và triệt tiêu được hoàn toàn sai số xác lập. Tuy nhiên, khi động cơ mang tải, khi xảy ra sự hao mòn cơ khí hoặc khi môi trường làm việc của động cơ khác với môi trường thực hiện đề tài sẽ dẫn đến các thông số của động cơ thay đổi, lúc này ta cần phải tinh chỉnh lại các thông số của bộ điều khiển PID, việc này sẽ không phù hợp với môi trường ứng dụng thực tế. Thay vì bộ điều khiển PID, ta có thể dùng bộ điều khiển STR, MRAS... để hệ vẫn đảm bảo chất lượng khi thông số của động cơ thay đổi.

Về phần nhúng: vi điều khiển ESP32 với framework ESP - IDF cho tốc độ xử lý rất nhanh với RTOS, đảm bảo được tính realtime của hệ thống khi phải xử lý nhiều tác vụ phức tạp cùng lúc. Vi điều khiển ESP32 có giá thành rẻ, chất lượng gia công tốt và có độ phỏ biến, phù hợp với quy mô và ứng dụng của đề tài.

Về phần cứng: mạch lái L298N thực hiện rất tốt nhiệm vụ lái động cơ. Bên cạnh đó, mạch cầu H cũng hạn chế tối đa việc dòng điện ngược chiều sinh ra từ cuộn cảm của động cơ làm hư hỏng vi xử lý.

Về phần mềm: giao diện WinForm GUI trên Visual Studio cung cấp tất cả các chức năng cần có để người dùng dễ dàng giao tiếp qua cổng COM giữa vi điều khiển và máy tính. Visual Studio cung cấp các hàm và tính năng có sẵn giúp người dùng thuận tiện trong việc làm quen và lập trình giao diện.

Danh mục tài liệu tham khảo

- [1] Diễm Quỳnh. (27/08/2022). *Động cơ DC là gì? Cấu tạo của động cơ DC trong máy chạy bộ.* Truy cập từ: <https://www.dienmayxanh.com/kinh-nghiem-hay/dong-co-dc-la-gi-cau-tao-cua-dong-co-dc-1460196>
- [2] *Giới thiệu về bo mạch phát triển ESP32.* Truy cập từ: <https://mecsu.vn/ho-tro-ky-thuat/gioi-thieu-ve-bo-mach-phat-trien-esp32.KyJ>
- [3] *GB37-545 động cơ giảm tốc có Encoder 12VDC, 168 rpm, trục 6mm.* Truy cập từ: <https://www.thegioiic.com/gb37-545-dong-co-giam-toc-co-encoder-12vdc-168-rpm-truc-6mm>
- [4] *Tìm hiểu chung về bộ điều khiển PID: Khái niệm, phân loại và ứng dụng.* Truy cập từ: <https://bkaii.com.vn/tin-tuc/1030-tim-hieu-chung-ve-bo-dieu-khien-pid-khai-niem-phan-loai-va-ung-dung>
- [5] Trần Hoàng Khôi Nguyên. (05/2017). *Thiết kế bộ điều khiển PID số động cơ Servo.*