

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ



NHẬP MÔN ĐIỀU KHIỂN THÔNG MINH
ĐỀ TÀI: TÌM HIỂU, THIẾT KẾ VÀ THỰC NGHIỆM
BỘ ĐIỀU KHIỂN MODEL-FREE ADAPTIVE (MFA)
ĐIỀU KHIỂN TỐC ĐỘ ĐỘNG CƠ DC

GVHD: HUỖNH THÁI HOÀNG

NHÓM LỚP: L01

Nhóm 1

Nguyễn Gia Khiêm	1810236
Phạm Ngọc Trân	1811291
Giang Văn Tín	1811279
Trương Duy Khang	1810985

TP Hồ Chí Minh, tháng 06 năm 2021

MỤC LỤC

1. Đặt vấn đề	3
2. Cơ sở lý thuyết	4
2.1. Mạng Noron	4
2.1.1. Khái niệm mạng Noron nhân tạo	4
2.1.2. Cấu trúc mạng Noron	4
2.1.3. Mạng MLP (Multilayer Perceptron)	5
2.2. Bộ điều khiển MFA (Model-free adaptive)	8
2.2.1. Hệ thống vòng lặp đơn sử dụng bộ điều khiển MFA	8
2.2.3. Thuật toán điều khiển của SISO MFA	10
3. Nội dung thực hiện	12
3.1. Lập trình giải thuật bằng MATLAB và C	12
3.1.1. Lập trình giải thuật Matlab	12
3.1.2. Lập trình giải thuật C	13
3.2. Tổng quan kết nối	15
3.3. Thiết kế và gia công mạch ra chân	16
3.4. Lập trình vi điều khiển	17
3.4.1. Sơ đồ tổ chức thư mục	17
3.4.2. Lưu đồ lập trình	18
3.4.3. Lập trình truyền nhận dữ liệu giữa vi điều khiển và máy tính	19
3.5. Lập trình giao diện	20
4. Kết quả thực hiện	22
4.1. Kết quả mô phỏng Matlab	22
4.2. Phần cứng	30
4.3. Giao diện máy tính	31
4.4. Đáp ứng động cơ	32
5. Đánh giá kết quả và hướng phát triển	36
5.1. Đánh giá kết quả	36
5.2. Hướng phát triển	36
6. Tài liệu tham khảo	37

1. Đặt vấn đề

Điều khiển thích ứng không mô hình (MFA) là một công nghệ đã có tác động lớn đến lĩnh vực điều khiển tự động. Vì bộ điều khiển MFA có thể thay thế trực tiếp các bộ điều khiển PID cũ mà không cần thiết kế lại "hệ thống". Chúng có thể dễ dàng nhúng vào các công cụ và thiết bị điều khiển thông minh khác nhau. Điều này làm giảm bớt những lo ngại liên quan đến chi phí thay đổi và cũng làm cho MFA trở thành một công cụ hấp dẫn cho các dự án với quy mô lớn.

Điều khiển thích ứng không cần mô hình (MFA) là một phương pháp điều khiển thích ứng không yêu cầu các mô hình quy trình. Các biến thể của công nghệ điều khiển MFA cốt lõi được ứng dụng giải quyết các vấn đề điều khiển cụ thể như: SISO MFA để thay thế bộ điều khiển PID, loại bỏ việc điều chỉnh bộ điều khiển thủ công; MFA phi tuyến để điều khiển các quá trình phi tuyến; bộ điều khiển pH MFA để điều khiển các quá trình pH; bộ MIMO MFA để kiểm soát các quy trình đa biến;... Bộ điều khiển MFA dành cho từng ứng dụng cụ thể có thể được nhúng dễ dàng và ngày càng trở nên khả dụng trên nhiều nền tảng. Ở đây, chúng ta sử dụng bộ điều khiển SISO MFA (Single Input Single Output) để điều khiển đáp ứng về tốc độ động cơ DC.

2. Cơ sở lý thuyết

2.1. Mạng Nơron

2.1.1. Khái niệm mạng Nơron nhân tạo

Mạng nơron nhân tạo, gọi tắt là mạng nơron là một mô hình toán học được xây dựng dựa trên cơ sở các mạng nơron sinh học, là sự tái tạo bằng kỹ thuật những chức năng của hệ thần kinh con người bao gồm một số lượng lớn các phần tử (gọi là nơron) kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) làm việc như một thể thống nhất để giải quyết các vấn đề cụ thể thông qua một quá trình học từ tập các mẫu huấn luyện.

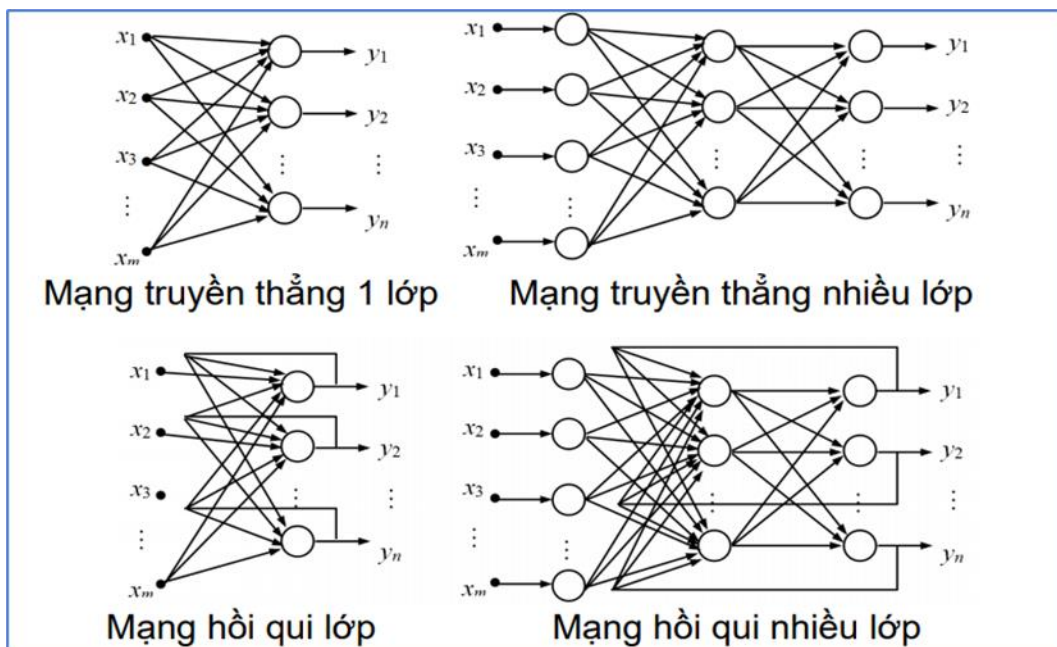
Hai đặc tính cơ bản của mạng nơron là:

- Quá trình tính toán được tiến hành song song và phân tán trên nhiều nơron gần như đồng thời.
- Tính toán thực chất là quá trình học, chứ không phải thực hiện theo sơ đồ định sẵn từ trước

2.1.2. Cấu trúc mạng Nơron

Nguyên lý cấu tạo của một mạng nơron là bao gồm nhiều lớp, mỗi lớp bao gồm nhiều nơron có cùng một chức năng. Có 2 dạng liên kết mạng cơ bản:

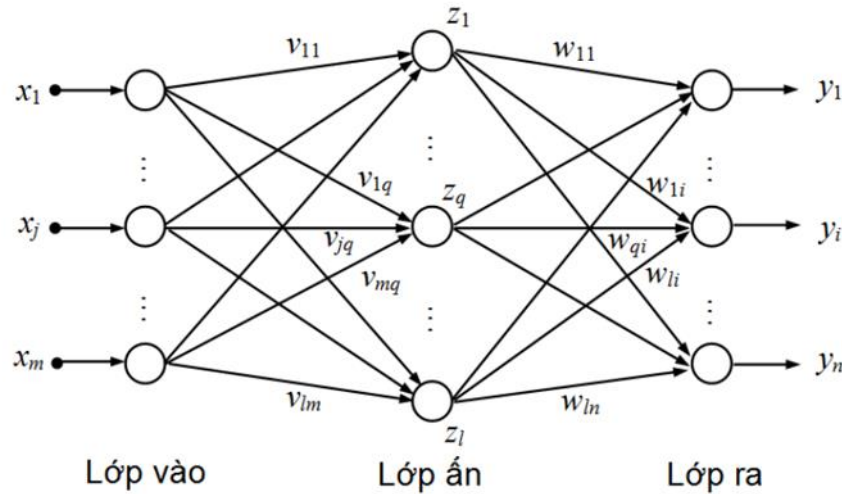
- Mạng truyền thẳng (Feedforward Neural Networks)
- Mạng có hồi tiếp



Hình 2.1.2 Cấu tạo mạng nơ-ron

2.1.3. Mạng MLP (Multilayer Perceptron)

Có rất nhiều nghiên cứu về mạng MLP và đã cho thấy nhiều ưu điểm của mạng này. Mạng MLP là cơ sở cho thuật toán lan truyền ngược và khả năng xấp xỉ liên tục. Một mạng MLP tổng quát là mạng có n lớp ($n \geq 2$) trong đó bao gồm một lớp vào, một lớp ra và một hoặc nhiều lớp ẩn.



Hình 2.1.3 Tổng quát mạng MLP

Các nơron đầu vào thực chất không phải các nơron theo đúng nghĩa, bởi lẽ chúng không thực hiện bất kỳ một tính toán nào trên dữ liệu vào, đơn giản nó chỉ tiếp nhận các dữ liệu vào và chuyển cho các lớp kế tiếp. Các nơron ở lớp ẩn và lớp ra mới thực sự thực hiện các tính toán, kết quả được định dạng bởi hàm đầu ra (hàm tác động).

- Hàm tuyến tính (purelin)

$$a(f) = f$$

- Hàm sigmoid (Sigmoid function (logsig))

$$a(f) = \frac{1}{1 + e^{-f}}$$

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện (trained) bởi thuật toán lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng $[0 \ 1]$.

- Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig))

$$a(f) = \frac{1 - e^{-2f}}{1 + e^{-2f}}$$

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng $[-1 \ 1]$.

Thiết kế cấu trúc mạng

Số lớp ẩn:

Người ta đã xác định rằng đối với phần lớn các bài toán cụ thể, chỉ cần sử dụng một lớp ẩn cho mạng là đủ. Các bài toán sử dụng hai lớp ẩn hiếm khi xảy ra trong thực tế. Thậm chí đối với các bài toán cần sử dụng nhiều hơn một lớp ẩn thì trong phần lớn các trường hợp trong thực tế, sử dụng chỉ một lớp ẩn cho ta hiệu năng tốt hơn là sử dụng nhiều hơn một lớp. Việc huấn luyện mạng thường rất chậm khi mà số lớp ẩn sử dụng càng nhiều.

Số đơn vị trong lớp ẩn:

Một vấn đề quan trọng trong việc thiết kế một mạng là cần có bao nhiêu đơn vị trong mỗi lớp. Sử dụng quá ít đơn vị có thể dẫn đến việc không thể nhận dạng được các tín hiệu đầy đủ trong một tập dữ liệu phức tạp. Sử dụng quá nhiều đơn vị sẽ tăng thời gian luyện mạng. Trong trường hợp này mạng có quá nhiều thông tin, hoặc lượng thông tin trong tập dữ liệu mẫu không đủ các dữ liệu đặc trưng để huấn luyện mạng.

Số lượng tốt nhất của các đơn vị ẩn phụ thuộc vào rất nhiều yếu tố - số đầu vào, đầu ra của mạng, số trường hợp trong tập mẫu, độ nhiễu của dữ liệu đích, độ phức tạp của hàm lỗi, kiến trúc mạng và thuật toán luyện mạng. Cách tốt nhất là sử dụng phương pháp thử-sai (trial-and-error).

Thuật toán lan truyền ngược (Back-Propagation)

Bước 1: Chọn η ; gán $k = 1$, $E = 0$; Khởi động ngẫu nhiên trọng số lớp ẩn và lớp ra $v_{jq}(k)$, $w_i(k)$ ($1 \leq q \leq l$, $1 \leq i \leq n$)

Bước 2: (truyền thuận dữ liệu) Tính ngõ ra của mạng với tín hiệu vào $x(k)$

- Ngõ ra tế bào thần kinh thứ q ở lớp ẩn:

$$z_q = a_h(net_{hq}) = a_h\left(\sum_{j=1}^m v_{jq}x_j\right)$$

- Ngõ ra tế bào thần kinh thứ i ở lớp ra:

$$y_i = a_o(net_{oi}) = a_o\left(\sum_{q=1}^l w_{qi}z_q\right)$$

Bước 3: (Lan truyền ngược sai số) Cập nhật trọng số:

- Biểu thức cập nhật trọng số lớp ra:

$$\delta_{oi}(k) = [(d_i(k) - y_i(k))][a'_{o.}(net_{oi}(k))]$$

$$w_i(k+1) = w_i(k) + \eta \delta_{oi}(k) z(k)$$

- Biểu thức cập nhật trọng số lớp ẩn:

$$\delta_{hq}(k) = \left[\sum_{i=1}^n \delta_{oi}(k) w_{iq}(k) \right] [a'_{o.}(net_{hq}(k))]$$

$$v_q(k+1) = v_q(k) + \eta \delta_{hq}(k) x(k)$$

Bước 4: Tính sai số tích lũy: Quá trình học là việc thực hiện cực tiểu hoá sai số có biểu thức sau:

$$E = E + \frac{1}{2} \sum_{i=1}^n [(k) - y_i(k)]^2$$

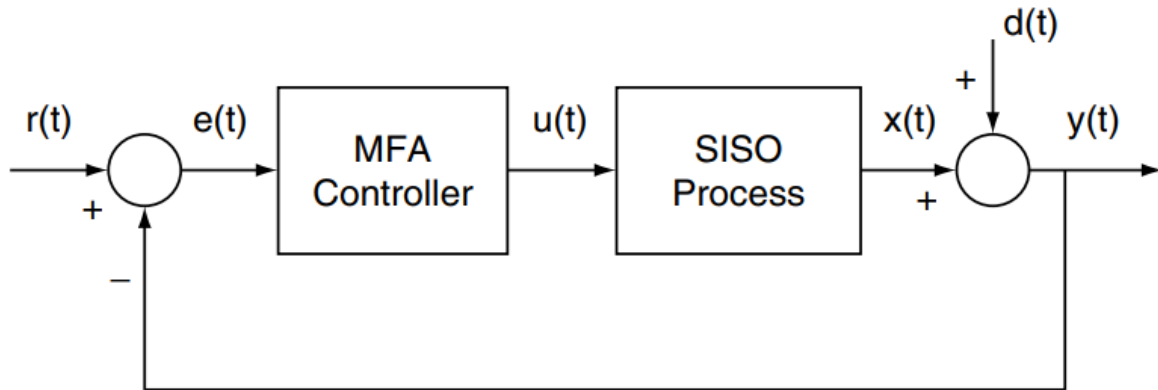
Bước 5: Nếu $k < K$ thì gán $k = k+1$ và trở lại bước 2. Nếu $k=K$ thì tiếp tục bước 6

Bước 6: Kết thúc chu kỳ huấn luyện. Nếu $E \leq \varepsilon$ thì kết thúc. Nếu $E \geq \varepsilon$ thì gán gán $k = 1$, $E = 0$ và trở lại bước 2

Mạng MLP là một giải pháp hữu hiệu cho việc mô hình hoá, đặc biệt với quá trình phức tạp hoặc cơ chế chưa rõ ràng và không đòi hỏi phải biết trước dạng hoặc tham số.

2.2. Bộ điều khiển MFA (Model-free adaptive)

2.2.1. Hệ thống vòng lặp đơn sử dụng bộ điều khiển MFA



Hình 2.2.1: Hệ thống vòng lặp đơn sử dụng bộ điều khiển MFA

$r(t)$: Setpoint (SP)

$y(t)$: Process variable (PV)

$x(t)$: Process output

$u(t)$: Controller output (OP)

$d(t)$: Noise

$e(t)$: Error between SP and PV, $e(t) = r(t) - y(t)$

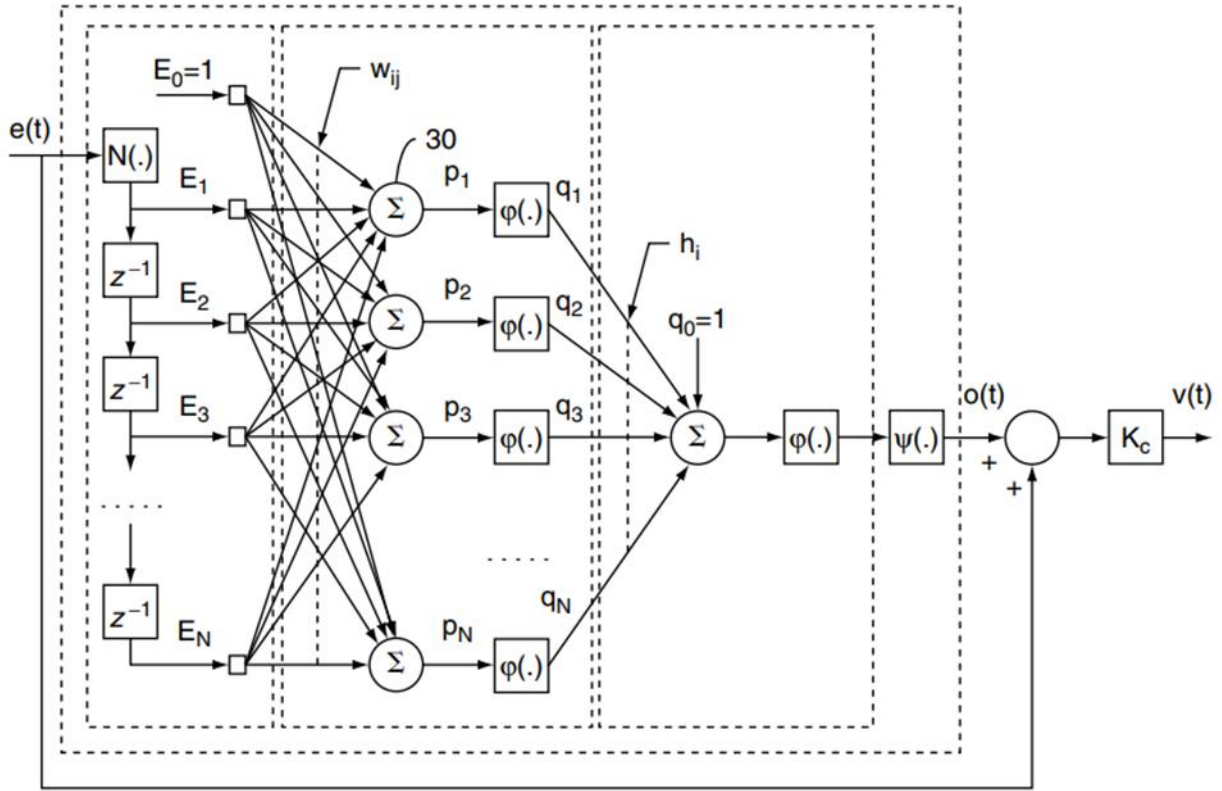
Bộ điều khiển thích nghi không có mô hình là bộ điều khiển hoạch định thời gian thực trực tuyến. Mục đích điều khiển của nó là làm cho biến quá trình $y(t)$ bám theo quỹ đạo đã cho của điểm đặt $r(t)$. Nói cách khác, nhiệm vụ của bộ điều khiển MFA là giảm thiểu sai số $e(t)$ trong dạng trực tuyến.

Hàm mục tiêu cho hệ thống điều khiển MFA là:

$$E_s(t) = \frac{1}{2} e(t)^2 = \frac{1}{2} [r(t) - y(t)]^2$$

Các thuật toán được sử dụng trong bộ điều khiển MFA để cập nhật các trọng số dựa trên một mục tiêu duy nhất, đó là giảm thiểu sai số giữa điểm đặt và biến quá trình. Điều đó có nghĩa là, khi quá trình diễn ra trạng thái ổn định khi đó sai số sẽ gần bằng 0, không cần cập nhật hệ số trọng số bộ điều khiển.

2.2.2. Kiến trúc bộ điều khiển SISO MFA



Hình 2.2.2 Kiến trúc bộ điều khiển SISO MFA

Cấu trúc của mạng gồm một lớp đầu vào, một lớp ẩn với N nơ-ron và một lớp đầu ra với một nơ-ron.

Tín hiệu đầu vào $e(t)$ của lớp đầu vào được chuyển đổi thành tín hiệu sai số chuẩn hóa E_1 với phạm vi -1 đến 1 bằng cách sử dụng đơn vị chuẩn hóa, trong đó $N(\cdot)$ là một hàm tiêu chuẩn. Tùy vào tín hiệu sai số mà ta chọn hàm $N(\cdot)$ cho phù hợp.

Tín hiệu E_1 sau đó đi qua một loạt các đơn vị trễ lặp lại, trong đó z^{-1} biểu thị toán tử độ trễ đơn vị. Một tập hợp các tín hiệu sai số chuẩn hóa E_2 đến E_N sau đó được tạo ra. Bằng cách này, một tín hiệu liên tục $e(t)$ được chuyển đổi để thành một loạt các tín hiệu rời rạc, được sử dụng làm đầu vào cho mạng nơ-ron. Những tín hiệu sai số trễ $E_i, i = 1, \dots, N$, sau đó được chuyển đến lớp ẩn thông qua các kết nối mạng nơ-ron. Nó tương đương với việc thêm một cấu trúc phản hồi của mạng nơ-ron.

Mỗi tín hiệu đầu vào được truyền tải riêng biệt đến từng tế bào thần kinh trong lớp ẩn thông qua một đường dẫn được đánh trọng số bởi một hệ số trọng số riêng w_{ij} , trong đó $i = 1, 2, \dots, N$ và $j = 1, 2, \dots, N$. Các đầu vào cho mỗi tế bào thần kinh trong lớp ẩn được tổng hợp bởi bộ cộng với $E_0 = 1$, tín hiệu ngưỡng cho lớp ẩn, thông qua các trọng số không đổi

$W_{0j} = 1$ để tạo ra tín hiệu p_j . Khi đó tín hiệu p_j được lọc bởi một hàm kích hoạt để tạo ra q_j , trong đó j biểu thị nơ-ron thứ j trong lớp ẩn.

Hàm kích hoạt $\varphi(.)$ là hàm sigmoidal ánh xạ các số thực để chúng nằm trong $(0,1)$:

$$\varphi(x) = \frac{1}{1 + e^{-x}}$$

Mỗi tín hiệu đầu ra từ lớp ẩn được chuyển đến một nơ-ron duy nhất trong lớp đầu ra thông qua một đường dẫn có trọng số là h_i , trong đó $i=1, 2, \dots, N$. Các tín hiệu này được tổng hợp trong bộ cộng với $h_0 = 1$, ngưỡng tín hiệu cho lớp đầu ra, và sau đó được lọc bởi một hàm kích hoạt $\psi(.)$:

$$\psi(y) = \ln \frac{y}{1-y}$$

hàm kích hoạt này có tác dụng ánh xạ dãy đầu ra từ $(0, 1)$ phía trên về lại không gian thực để tạo đầu ra $o(t)$ cho mạng nơ-ron.

2.2.3. Thuật toán điều khiển của SISO MFA

Thuật toán điều chỉnh đầu vào/đầu ra của bộ điều khiển bao gồm những phương trình sau:

$$p_j(n) = \sum_{i=1}^N w_{ij}(n)E_i(n) + 1$$

$$q_j(n) = \varphi(p_j(n))$$

$$o(n) = \psi \left[\varphi \left(\sum_{j=1}^N h_j(n)q_j(n) + 1 \right) \right] = \sum_{j=1}^N h_j(n)q_j(n) + 1$$

$$v(t) = K_c[o(t) + e(t)]$$

trong đó n biểu thị lần lặp thứ n , $o(t)$ là hàm liên tục của $o(n)$, $v(t)$ là đầu ra của bộ điều khiển thích ứng không có mô hình và $K_c > 0$, được gọi là độ lợi bộ điều khiển, là hằng số dùng để điều chỉnh khả năng ảnh hưởng của bộ điều khiển. Hằng số này rất hữu ích để điều chỉnh hiệu suất của bộ điều khiển hoặc giữ cho hệ thống trong phạm vi ổn định.

Thuật toán học tập trực tuyến được phát triển để liên tục cập nhật các giá trị hệ số trọng số của bộ điều khiển MFA như sau:

$$\Delta w_{ij}(n) = \eta K_c \frac{\partial y(n)}{\partial u(n)} e(n) q_j(n) (1 - q_j(n)) E_i(n) \sum_{k=1}^N h_k(n)$$

$$\Delta h_j(n) = \eta K_c \frac{\partial y(n)}{\partial u(n)} e(n) q_j(n)$$

trong đó $\eta > 0$ là tốc độ học, và đạo hàm riêng $\frac{\partial y(n)}{\partial u(n)}$ là *gradient* của $y(t)$ đối với $u(t)$, đại diện cho độ nhạy của đầu ra $y(t)$ đối với các thay đổi của đầu vào $u(t)$. Ta đặt:

$$S_f(n) = \frac{\partial y(n)}{\partial u(n)}$$

là hàm độ nhạy của quá trình.

Vì quá trình này chưa được biết, nên hàm độ nhạy cũng không được biết. Thông qua phân tích độ ổn định của bộ điều khiển thích ứng không có mô hình, người ta thấy rằng nếu quá trình được điều khiển là vòng lặp hở ổn định, có thể kiểm soát được, và kiểu hoạt động của nó không thay đổi trong toàn bộ thời gian điều khiển, giới hạn $S_f(n)$ với một tập tùy ý các hằng số khác 0 có thể đảm bảo hệ thống được giới hạn đầu vào và giới hạn đầu ra (BIBO) ổn định. Tức là, hàm độ nhạy của quá trình $S_f(n)$ có thể được thay thế một cách đơn giản bởi một hằng số. Bằng cách chọn $S_f(n) = 1$, thuật toán học được kết quả như sau:


$$\Delta w_{ij}(n) = \eta K_c e(n) q_j(n) (1 - q_j(n)) E_i(n) \sum_{k=1}^N h_k(n)$$

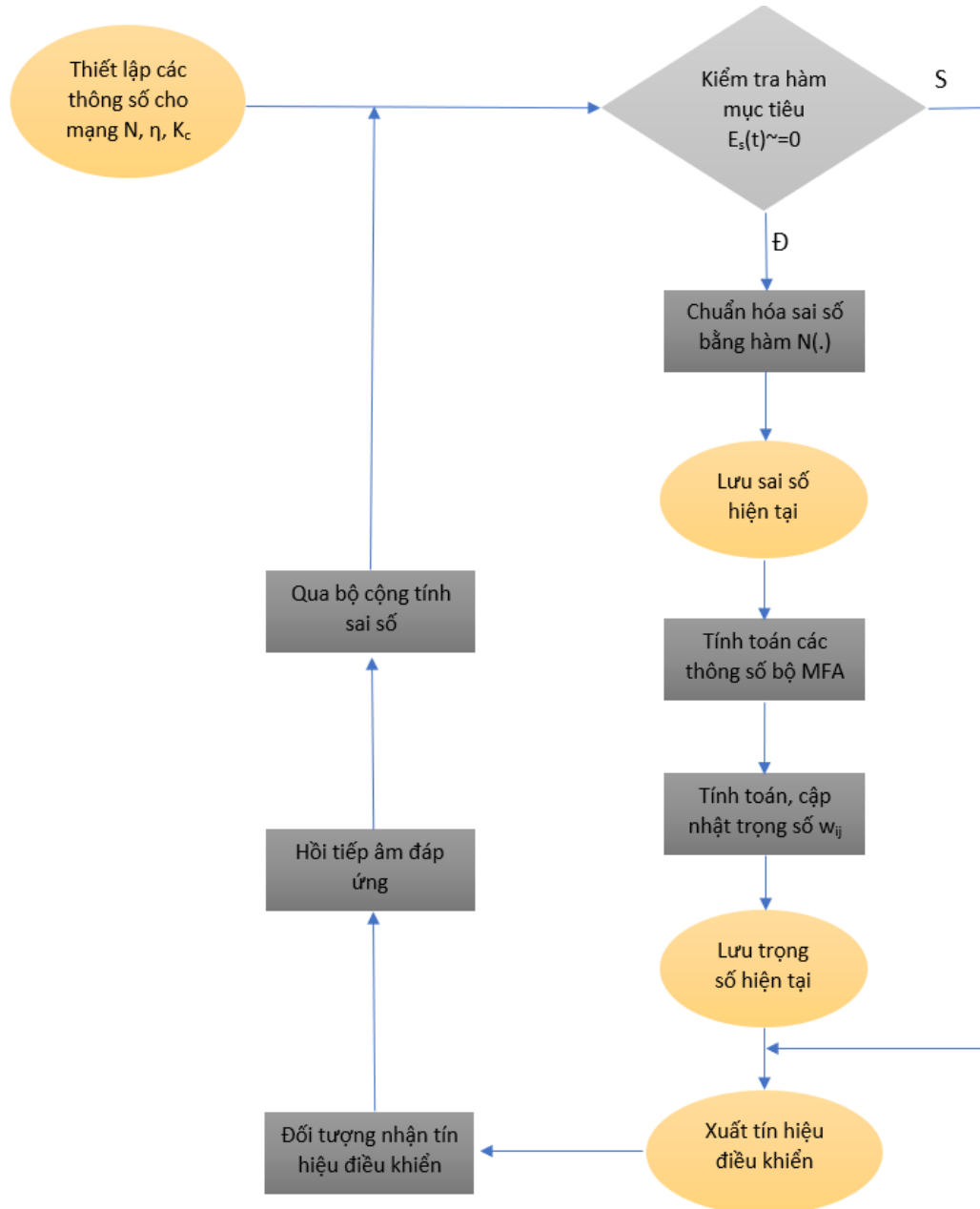
$$\Delta h_j(n) = \eta K_c e(n) q_j(n)$$

3. Nội dung thực hiện

3.1. Lập trình giải thuật bằng MATLAB và C

3.1.1. Lập trình giải thuật Matlab

Matlab sử dụng khối *Data Store Memory*  để tạo một mảng lưu các thông số trước đó của mạng nơ-ron dưới dạng *global* (các thông số đầu tiên sẽ có giá trị bằng 0).



Hình 3.1.1: Lưu đồ giải thuật Matlab

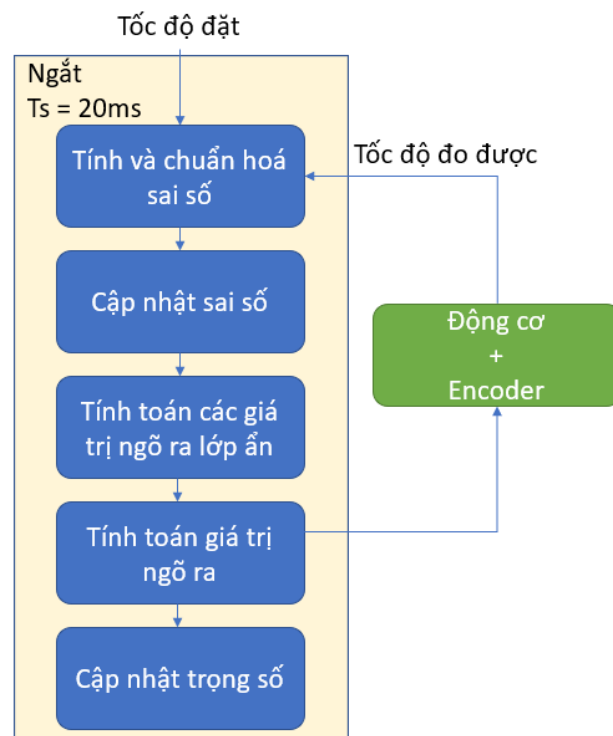
Các thông số của bộ điều khiển sẽ được lưu vào mảng theo thứ tự sau:

	1	2	3	4	5	N
1	E_1	w_{11}	w_{12}	w_{13}	w_{14}	...	w_{1N}	p_1	q_1	h_1	Sum_h k	o
2	E_2	w_{21}	w_{22}	w_{23}	w_{24}	...	w_{2N}	p_2	q_2	h_2		
3	E_3	w_{31}	w_{32}	w_{33}	w_{34}	...	w_{3N}	p_3	q_3	h_3		
4	E_4	w_{41}	w_{42}	w_{43}	w_{44}	...	w_{4N}	p_4	q_4	h_4		
...		
...		
N	E_N	w_{N1}	w_{NN}	p_N	q_N	h_N		

Bảng 3.1 Mảng lưu các thông số của bộ điều khiển

Các thông số của bộ điều khiển sẽ được tính toán và cập nhật vào mảng trên. Bằng cách này giúp ta dễ dàng thay đổi được số nơ-ron trong quá trình mô phỏng so với việc sử dụng các khối *Block* để tạo tính hiệu *Delay* thì việc thiết kế bộ điều khiển sẽ khó thực hiện và khó thay đổi được số lượng nơ-ron.

3.1.2. Lập trình giải thuật C



Hình 3.1.2: Lưu đồ giải thuật C

Trong lập trình C, ta tạo 2 struct: struct Motor và struct MFA_Controller.

Struct Motor sẽ chứa những tham số quan trọng của Motor như:

- Mảng chứa tốc độ đáp ứng (y)
- Mảng chứa tốc độ đặt (yr)
- Sai số tốc độ (e)

Struct MFA_Controller sẽ chứa những tham số quan trọng của bộ điều khiển như:

- Số lượng neuron (N), tốc độ cập nhật (eta), độ lợi (Kc), deadband (alpha)
- Mảng chứa các sai số đã chuẩn hoá (E[N])
- Mảng chứa các trọng số ($w[N*N]$ và $h[N]$)
- Mảng chứa giá trị ngõ ra của lớp ẩn ($p[N]$ và $q[N]$)
- Ngõ ra của bộ điều khiển (o và v)

Để thay đổi số lượng neuron của bộ điều khiển, nhóm đã có 3 hướng tiếp cận như sau:

1. Khởi tạo các mảng tham số với số neuron mong muốn, chỉ có thể chỉnh sửa khi nạp lại firmware.
2. Khởi tạo các mảng tham số với số neuron lớn, khi sử dụng bao nhiêu thì chỉ lấy các phần tử đầu của mảng đã tạo.
3. Khởi tạo các mảng tham số bằng cấp phát động.

Hướng (3) không khả thi do khi cấp phát động trên các phần tử nhưng không đơn giản như làm việc với hệ điều hành và phải cần một lượng kiến thức đủ vững. Hướng (2) sẽ làm lãng phí bộ nhớ của vi điều khiển. Khi thực hiện, nhóm nhận thấy, khi số lượng neuron đạt đến một mức đủ nhiều (khoảng 5 neuron trong đề tài này) thì chất lượng của bộ điều khiển không thay đổi nhiều. Vì vậy, nhóm quyết định lựa chọn hướng tiếp cận đầu tiên là đặt trước số lượng neuron cần dùng trong firmware.

Trong vi điều khiển STM32F4 có bộ tính toán số float giúp tăng tốc độ xử lý dữ liệu. Và để có thể sử dụng tốt bộ tính toán này, khi lập trình ta sử dụng những hàm toán trong số float như $\logf()$, $\tanhf()$,...

Ở những phần còn lại ta lập trình theo cấu trúc như khi lập trình bằng MATLAB.

3.2. Tổng quan kết nối

Sơ đồ khối tổng quan kết nối các khối phần cứng như sau:



Hình 3.2.1 Tổng quan kết nối phần cứng

Đề tài sử dụng encoder tương đối có thông số là 11 xung/vòng quay gắn vào trục động cơ giảm tốc GA25 – 400 rpm. Mạch cầu H sử dụng ở đây dùng để điều khiển động cơ nhằm mục đích đánh giá việc đọc xung từ encoder.

Vi điều khiển STM32F407VG là một trong những vi điều khiển mạnh mẽ ở thời điểm hiện tại, có nhiều chức năng, hỗ trợ từ cộng đồng rất lớn. Vi điều khiển được sử dụng với các chức năng: đọc xung từ encoder sử dụng chức năng Input Capture với Encoder Interface (để x4 lần xung đọc được) được hỗ trợ sẵn, giao tiếp với GUI trên máy tính bằng cổng COM với chức năng USART. Bảng dưới đây thể hiện chức năng và các chân sử dụng trên vi điều khiển STM32F407VG.

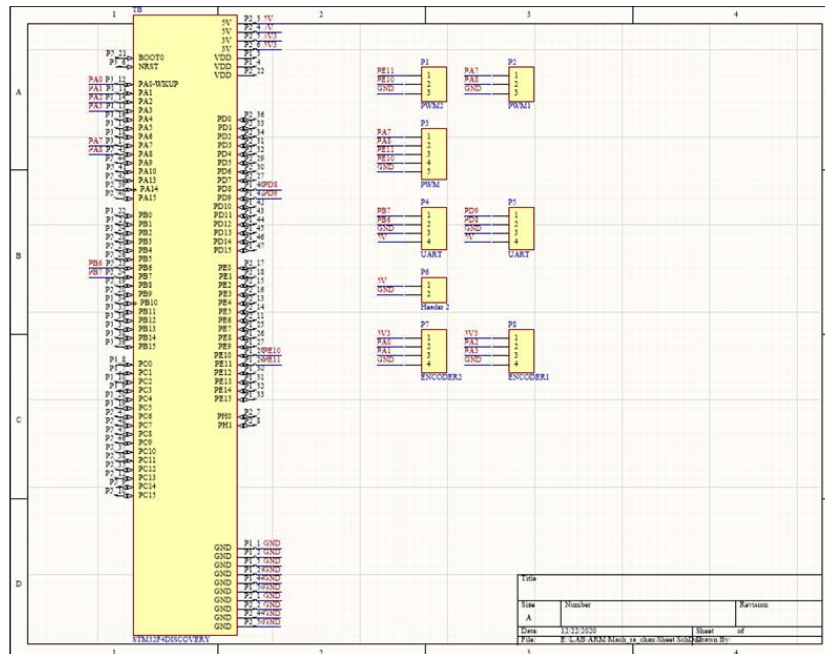
Chức năng	Chân	Ứng dụng
USART3, DMA1	PD8, PD9	Giao tiếp với GUI trên máy tính
PWM1, TIMER1	PA7, PA8	Xuất PWM cho động cơ
TIM6, TIM3		Timer thời gian dùng để tạo ngắt và delay
TIM5	PA0, PA1	Đọc xung từ Encoder

Bảng 3.2 Bảng chức năng và các chân sử dụng trên vi điều khiển STM32F407VG

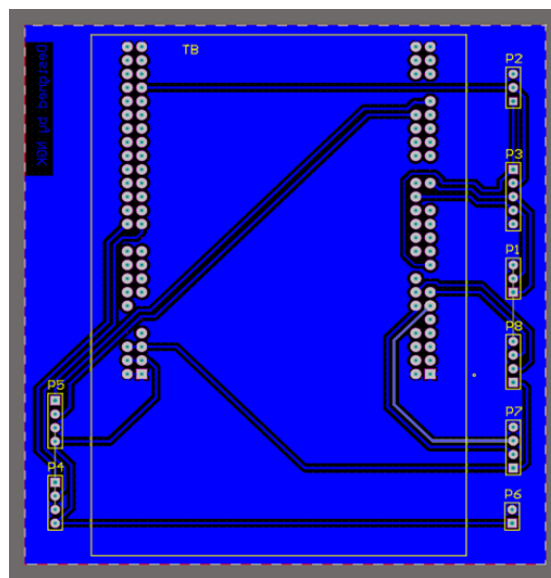
3.3. Thiết kế và gia công mạch ra chân

Thiết kế mạch ra chân nhằm mục đích dễ dàng kết nối các khối phần cứng khác đến vi điều khiển, đồng thời cố định phần cứng, tạo sự thuận tiện và chính xác trong quá trình chạy thử kết quả.

Sử dụng phần mềm Altium 19 để vẽ sơ đồ nguyên lý và PCB của mạch ra chân.



Hình 3.3.1 Sơ đồ nguyên lý của mạch ra chân



Hình 3.3.2 PCB của mạch ra chân

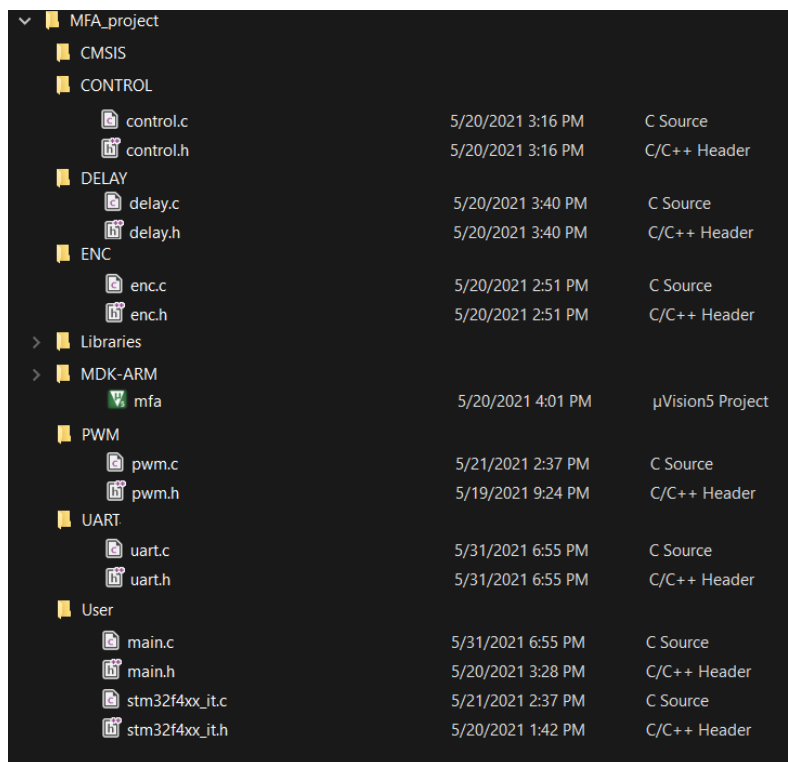
3.4. Lập trình vi điều khiển

Chọn KeilC là IDE để lập trình vi điều khiển STM32F407VG, sử dụng thư viện ST Standard Peripheral Library.

3.4.1. Sơ đồ tổ chức thư mục

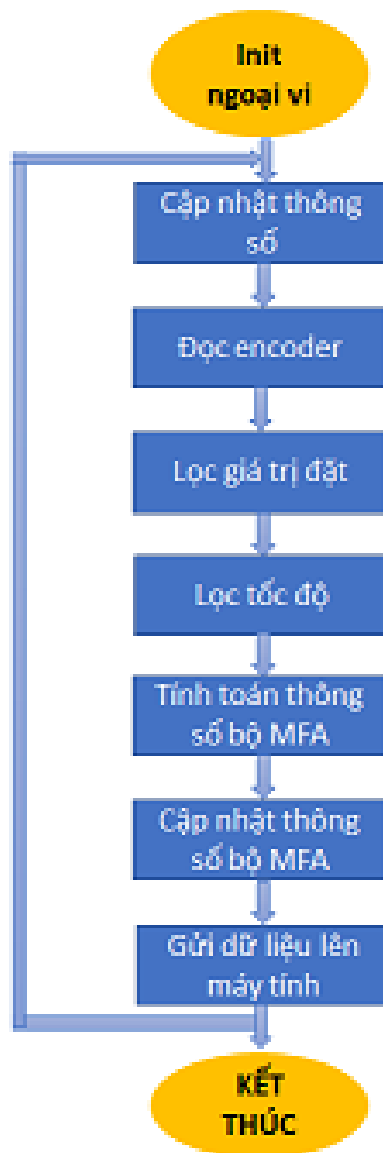
Việc tổ chức thư mục giúp trực quan, dễ quản lý code. Mỗi một thư mục con chứa một khối chức năng hoặc các tệp hệ thống.

- CMSIS: Thư mục chứa các tệp hệ thống.
- Libraries: Thư mục thư viện gốc, chứa tệp hệ thống và các driver.
- MDK-ARM: Thư mục chứa tệp project và các thư mục, tệp khác hỗ trợ compile, debug.
- User: Thư mục người dùng, chứa các tệp chương trình chính và ngắt.
- CONTROL, DELAY, UART, PWM, ENC: Thư mục chứa các tệp source và header dùng để lập trình các khối chức năng.

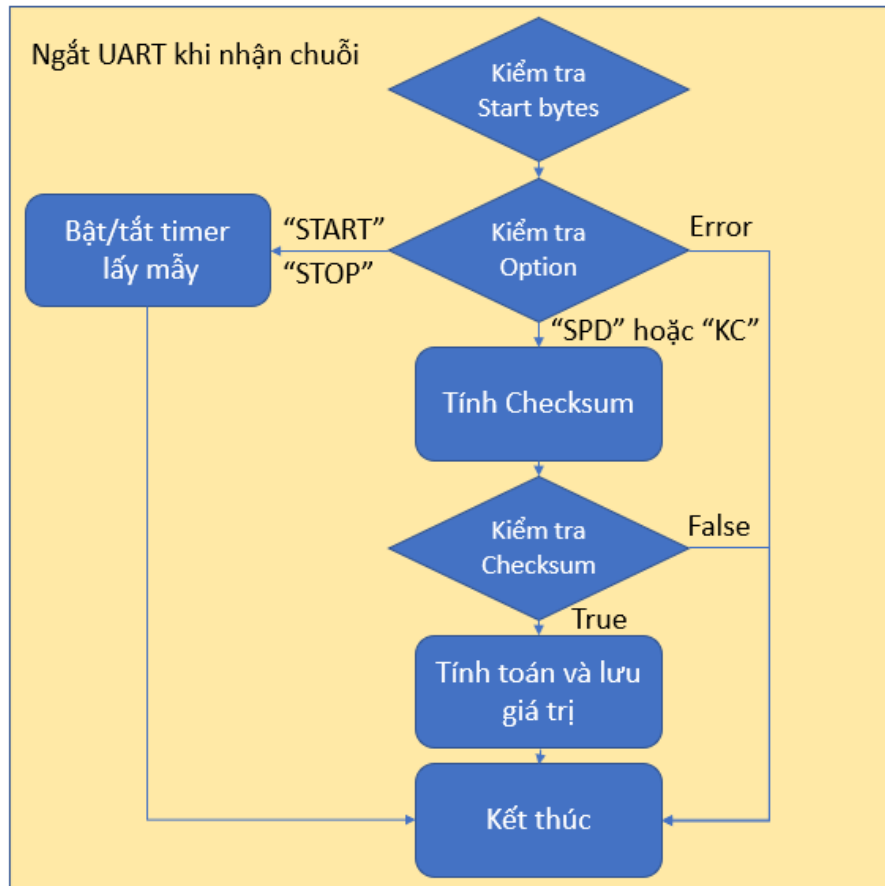


Hình 3.4.1 Sơ đồ tổ chức thư mục

3.4.2. Lưu đồ lập trình



Hình 3.4.2.1 Lưu đồ lập trình các khối chức năng



Hình 3.4.2.2 Lưu đồ lập trình ngắt nhận dữ liệu

3.4.3. Lập trình truyền nhận dữ liệu giữa vi điều khiển và máy tính

Ta quy định các frame truyền đi từ vi điều khiển như sau:



Frame truyền có chiều dài 9 byte, cụ thể là:

- 3 byte Start: "MFA," dùng để nhận diện frame
- 4 byte Speed: theo kiểu float
- 1 byte CheckSum: dùng để kiểm tra lỗi
- 1 byte Stop: "\$"

Ta quy định các frame nhận vào vi điều khiển như sau:



Frame nhận có chiều dài 17 byte, cụ thể là:

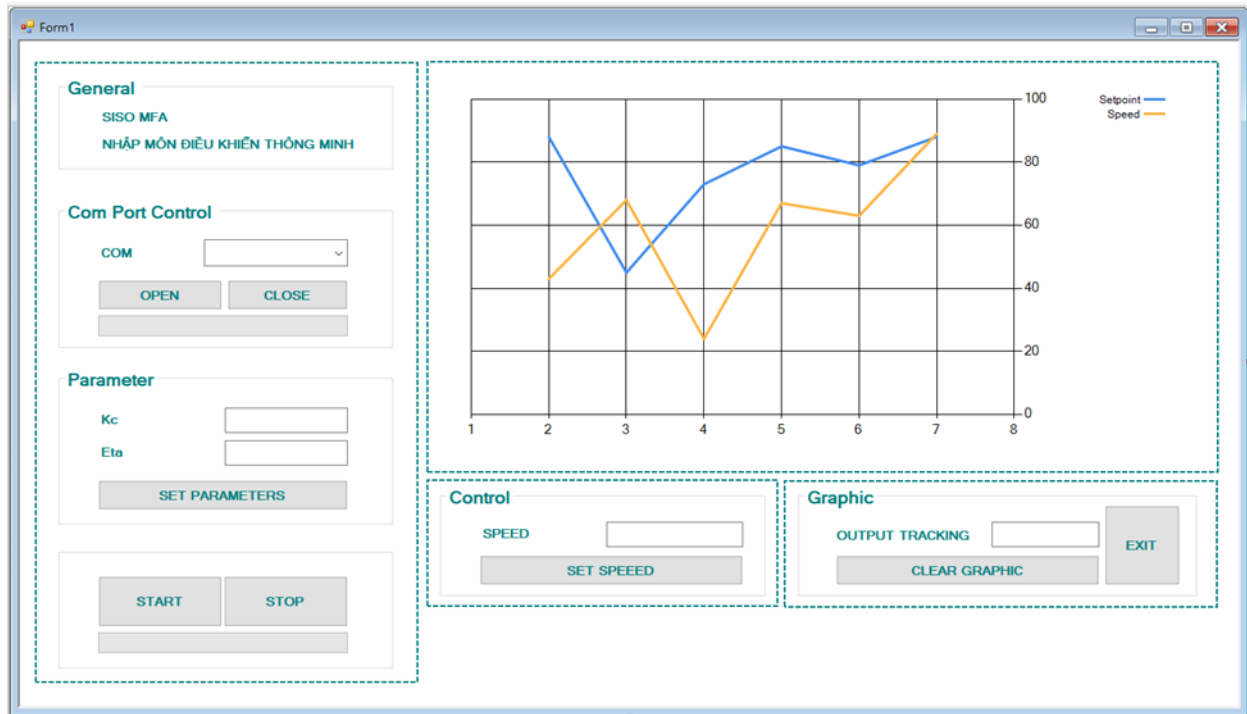
- 3 byte Start: “MFA” dùng để nhận diện frame
- byte Option/Value: bật tắt timer/gửi speed hoặc chứa dữ liệu cần gửi (setpoint hoặc các thông số bộ MFA)
- byte Value: dữ liệu cần gửi (setpoint hoặc các thông số bộ MFA).
- 1 byte CheckSum: dùng để kiểm tra lỗi
- 1 byte Stop: “\$”

3.5. Lập trình giao diện

GUI được viết bằng ngôn ngữ C# sử dụng .NET framework – một framework mạnh mẽ, môi trường thực thi cung cấp nhiều tính năng cao cấp (như quản lý bộ lý, xử lý ngoại lệ, v.v.), do đó việc lập trình ứng dụng đơn giản hơn. Sơ đồ khối của giao diện được minh họa bằng hình bên dưới:



Hình 3.5.1: Sơ đồ khối GUI



Hình 3.5.2: Giao diện GUI hoàn chỉnh

Giao diện được xây dựng để thực hiện chức năng truyền nhận dữ liệu giữa vi xử lý và máy tính:

- Kết nối với máy tính thông qua cổng COMPORT (serial port – USB Uart)
- Thực hiện gửi dữ liệu xuống vi điều khiển khi thao tác các nút nhấn:
 - **SET PARAMETERS** – gửi frame truyền chứa các thông số cấu hình ban đầu: Kc, Eta.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12
Tx	M	F	A	Kc			Eta			checksum		S	

- **START / STOP** – thông báo cho vi xử lý bắt đầu thực hiện/dừng chương trình.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12
Tx	M	F	A	S	T	A	R	T	-	-	-	checksum	S

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12
Tx	M	F	A	S	T	O	P	-	-	-	-	checksum	S

- **SET SPEED** – gửi giá trị setpoint điều khiển mong muốn

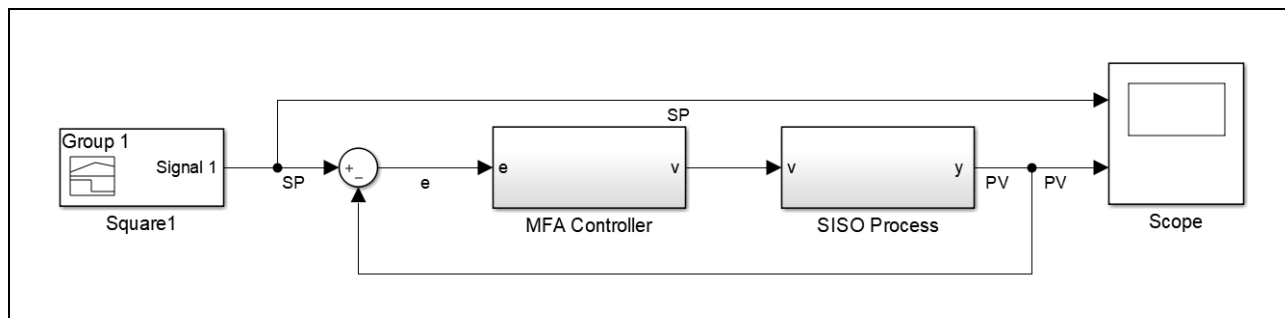
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12
Tx	M	F	A	S	S	P	D	SPEED				checksum	S

- Vì xử lý gửi lên PC giá trị tốc độ đọc từ Encoder để vẽ đồ thị theo chu kỳ 50ms

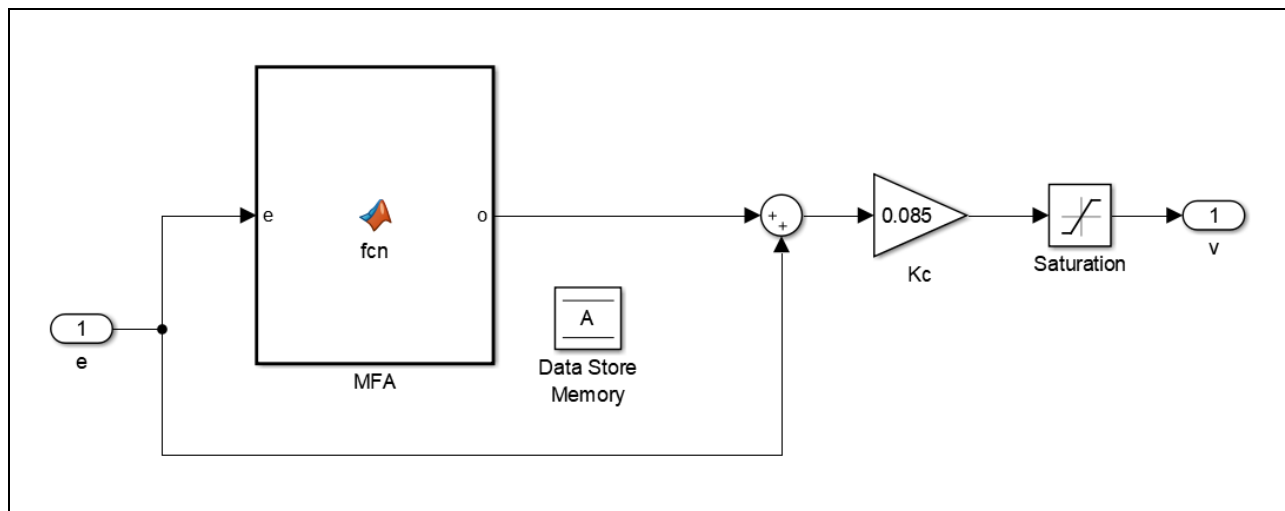
Byte	0	1	2	3	4	5	6	7	8
Rx	M	F	A	SPEED				checksum	S

4. Kết quả thực hiện

4.1. Kết quả mô phỏng Matlab

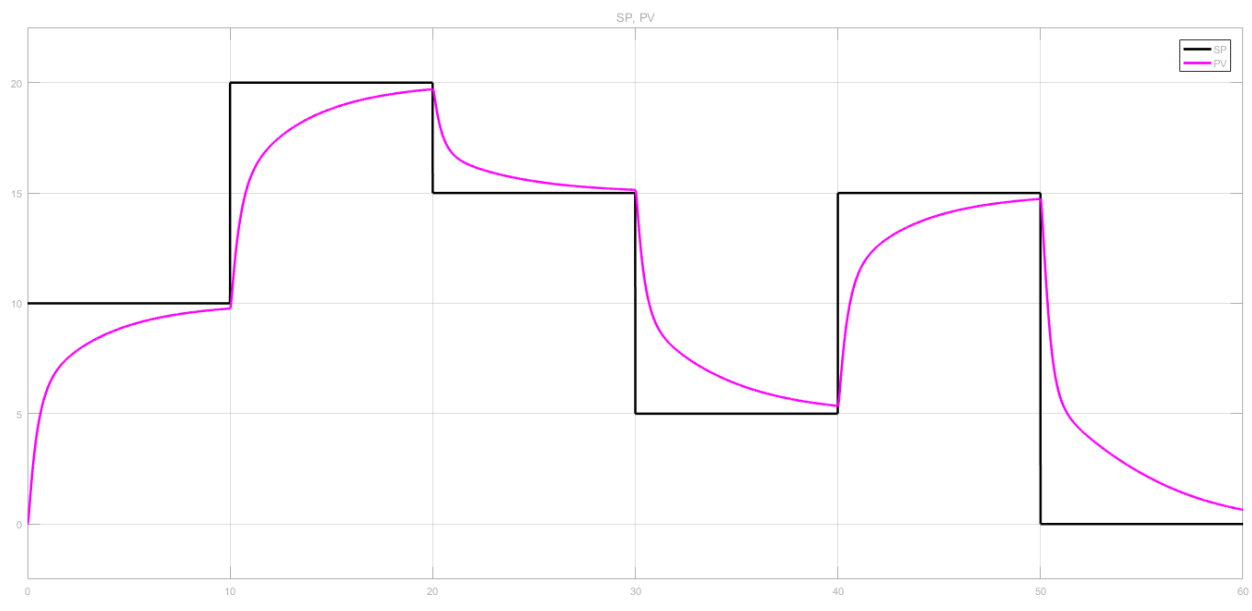


Hình 4.1.1 Sơ đồ mô phỏng vòng lặp đơn sử dụng bộ MFA

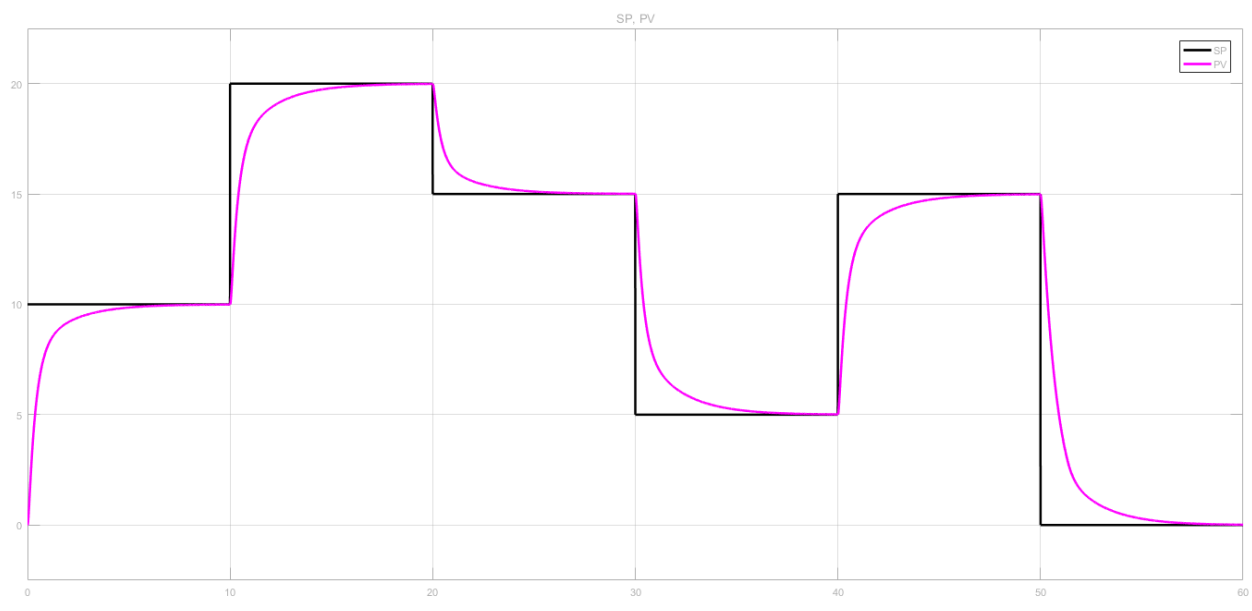


Hình 4.1.2 Sơ đồ mô phỏng bộ MFA
(có sử dụng khâu bão hòa để quá trình điều khiển sát với thực tế)

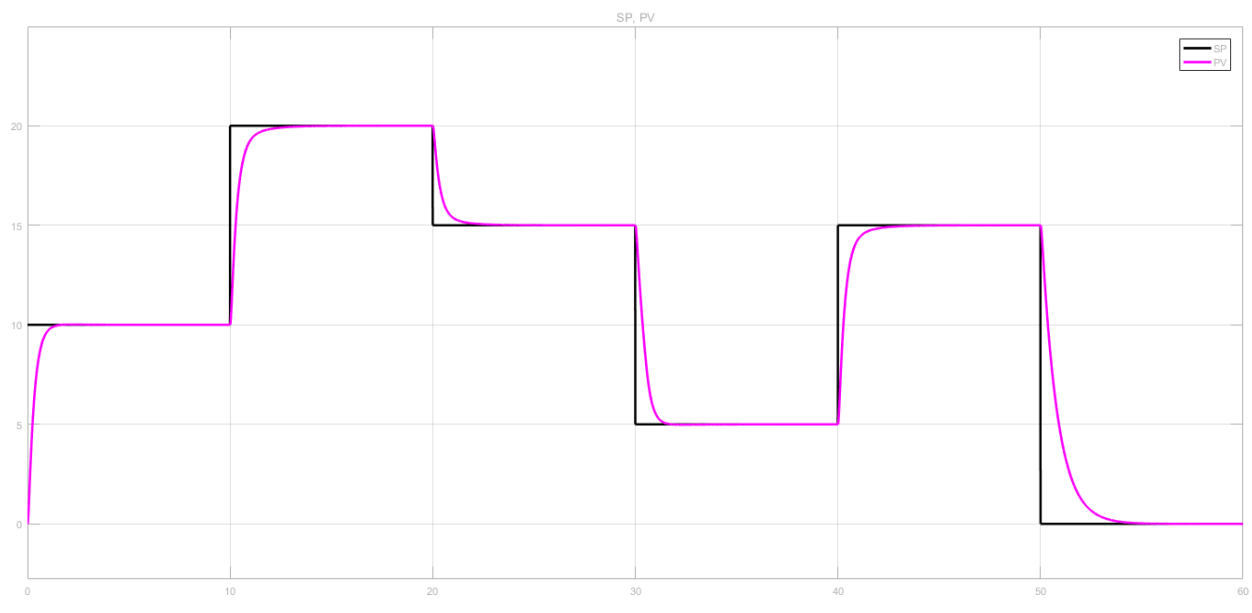
- Thực hiện với cùng các giá trị đặt và chỉ thay đổi độ lợi K_c



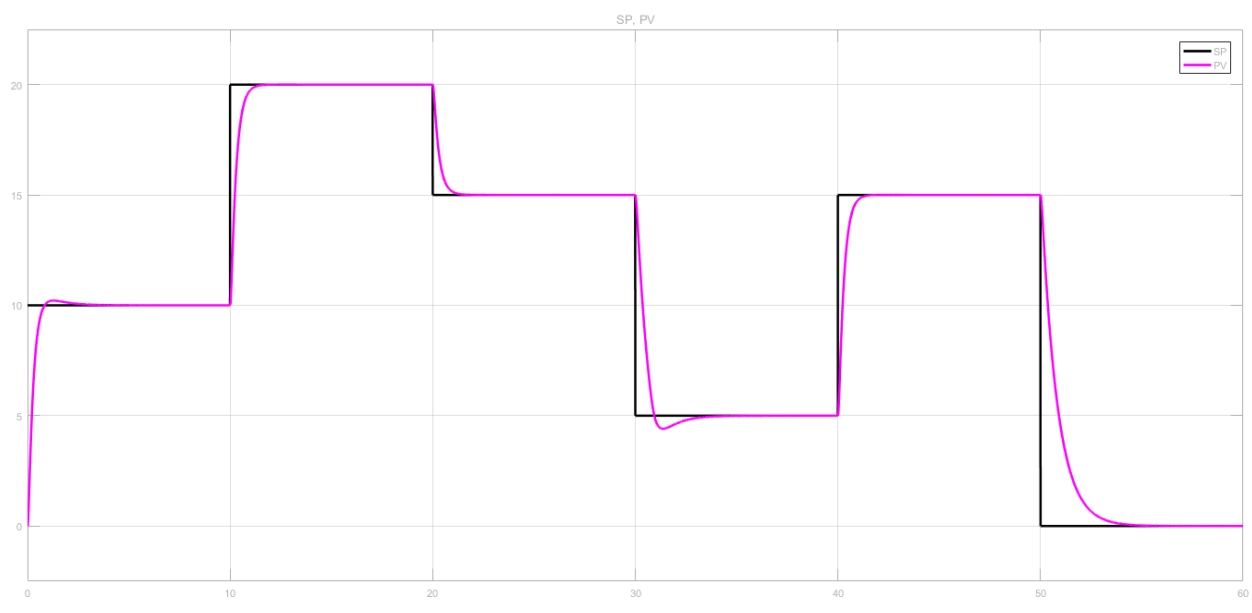
Hình 4.1.1 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.3$)



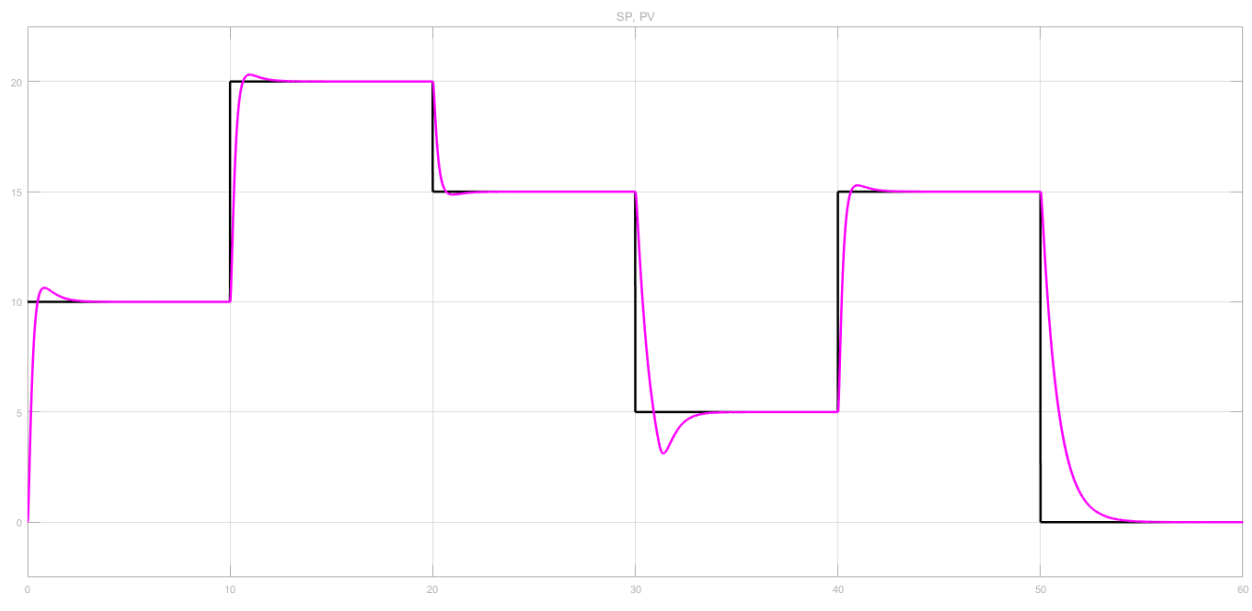
Hình 4.1.2 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.45$)



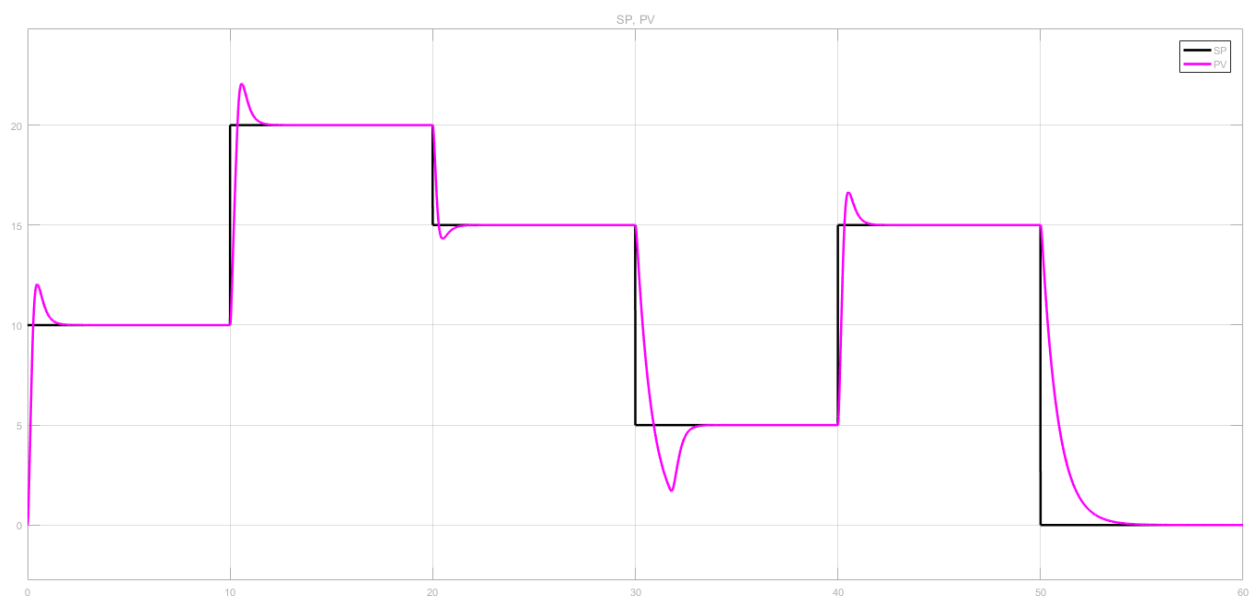
Hình 4.1.3 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.65$)



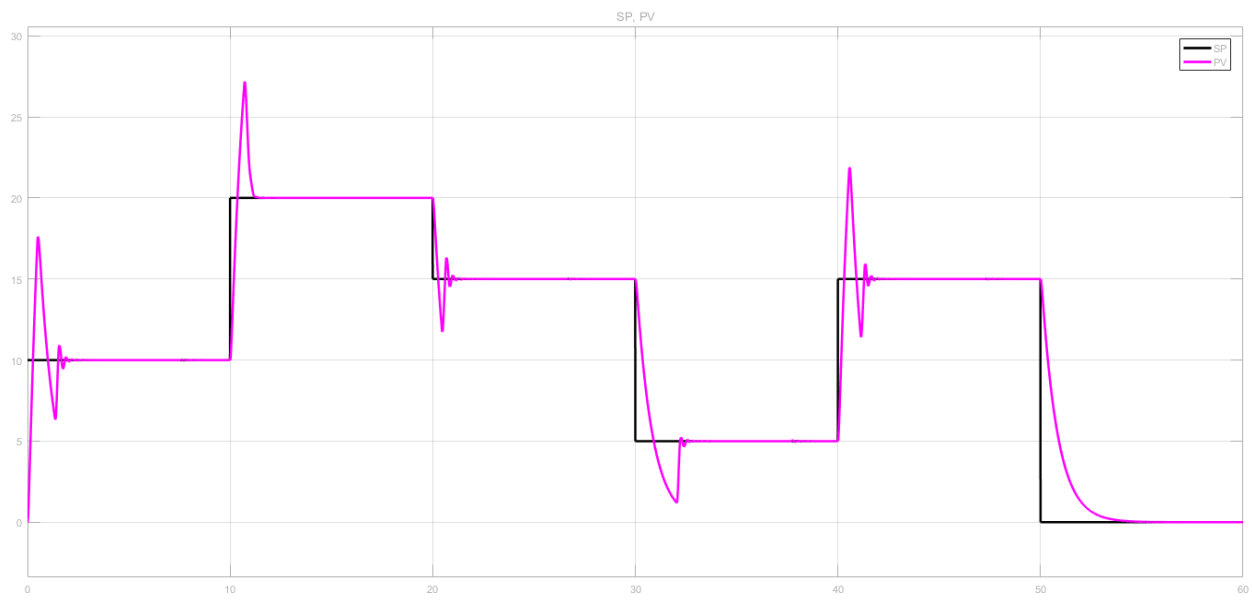
Hình 4.1.4 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.75$)



Hình 4.1.5 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.95$)

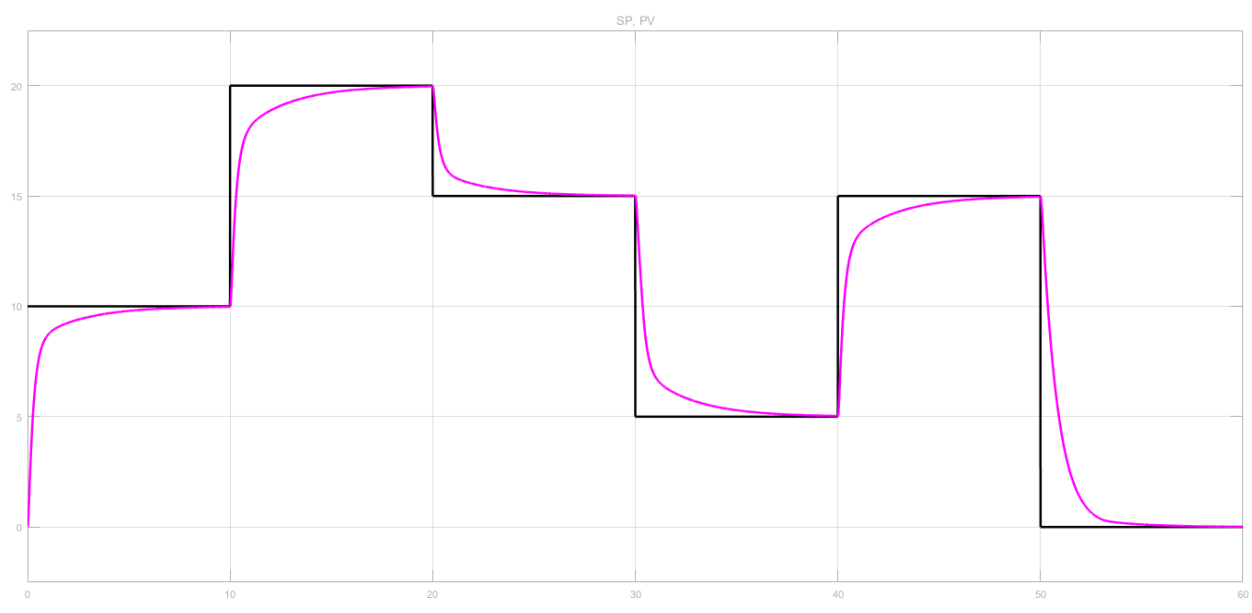


Hình 4.1.6 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 1.5$)

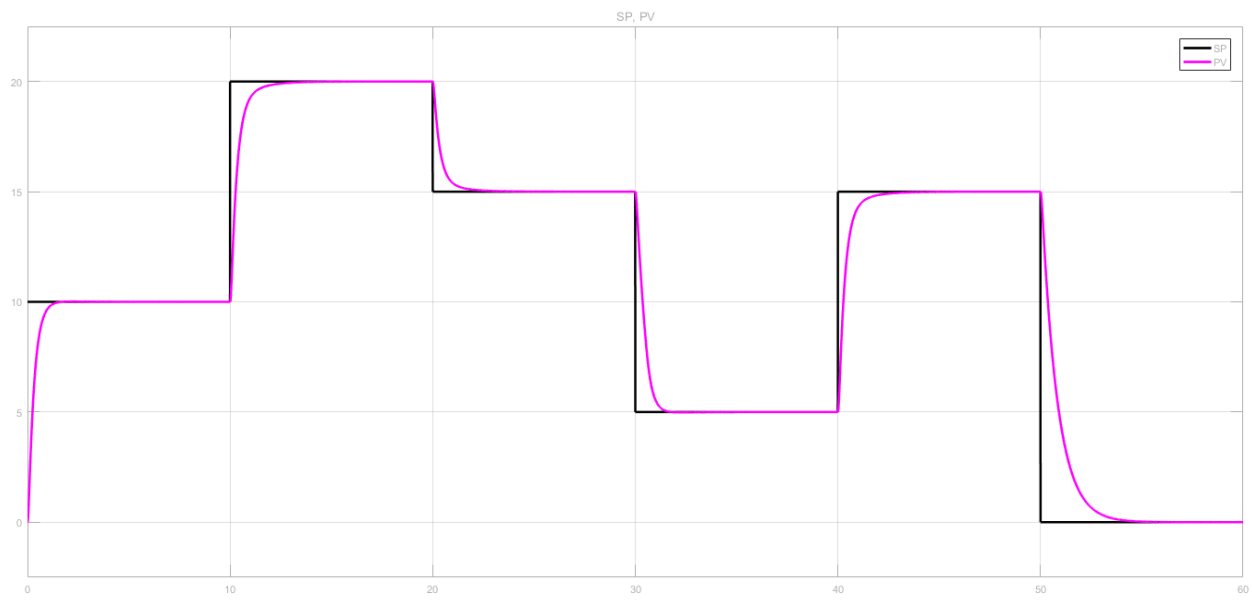


Hình 4.1.7 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 6$)

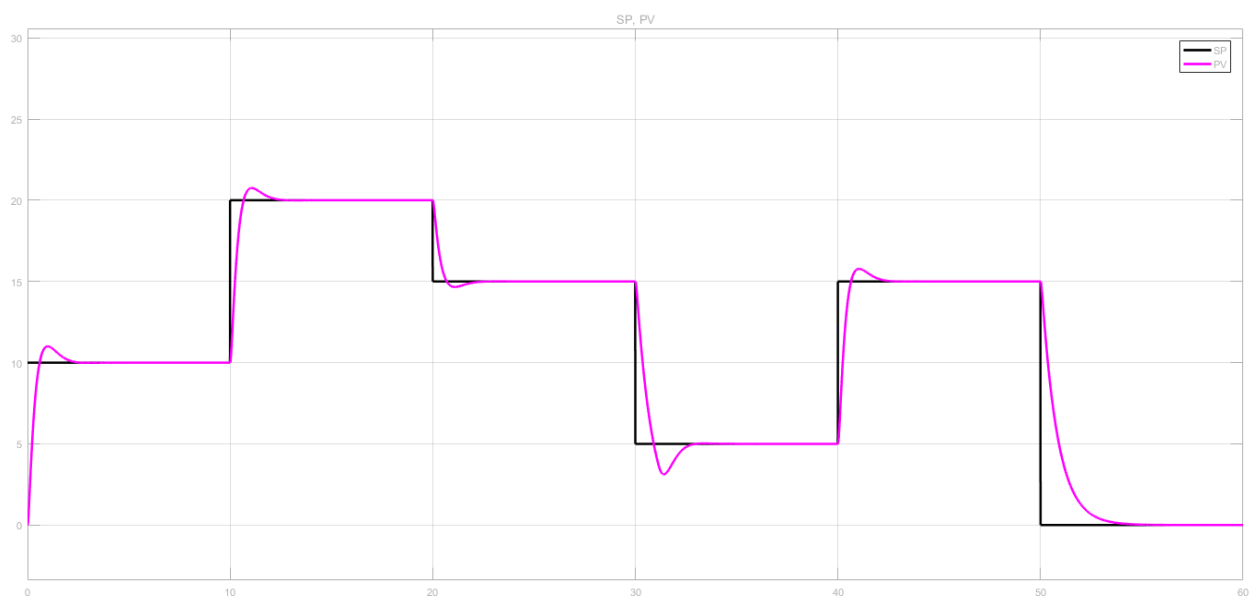
- Thực hiện với cùng các giá trị đặt và chỉ thay đổi hệ số học η



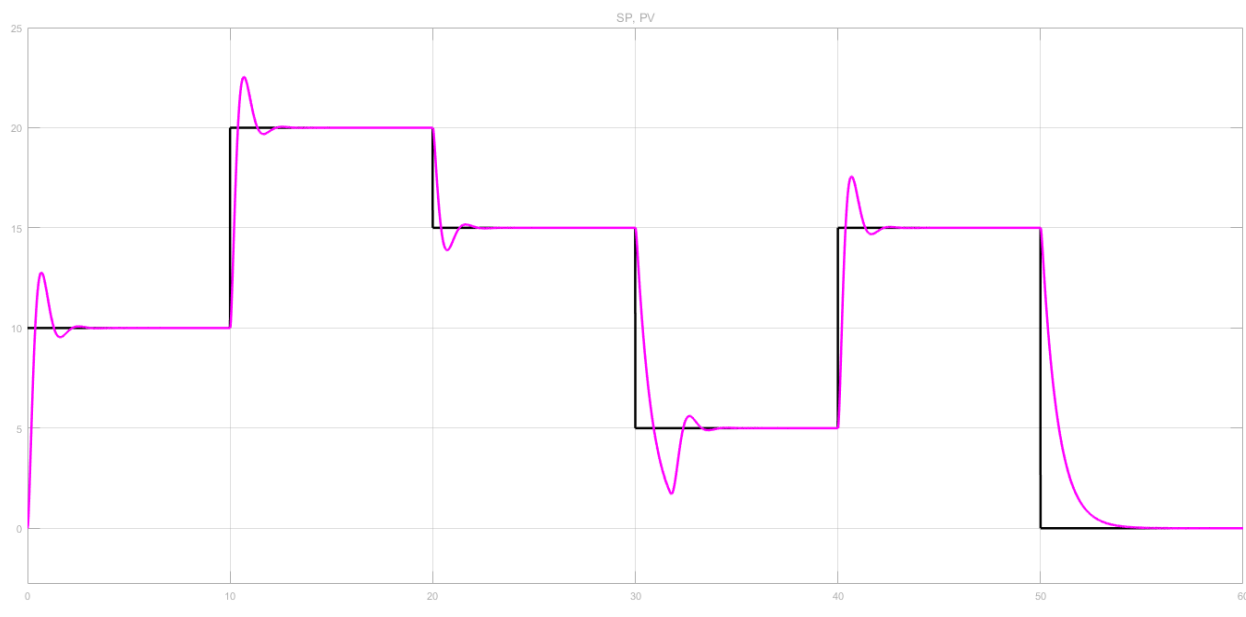
Hình 4.1.8 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.1, K_c = 0.65$)



Hình 4.1.9 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.65$)

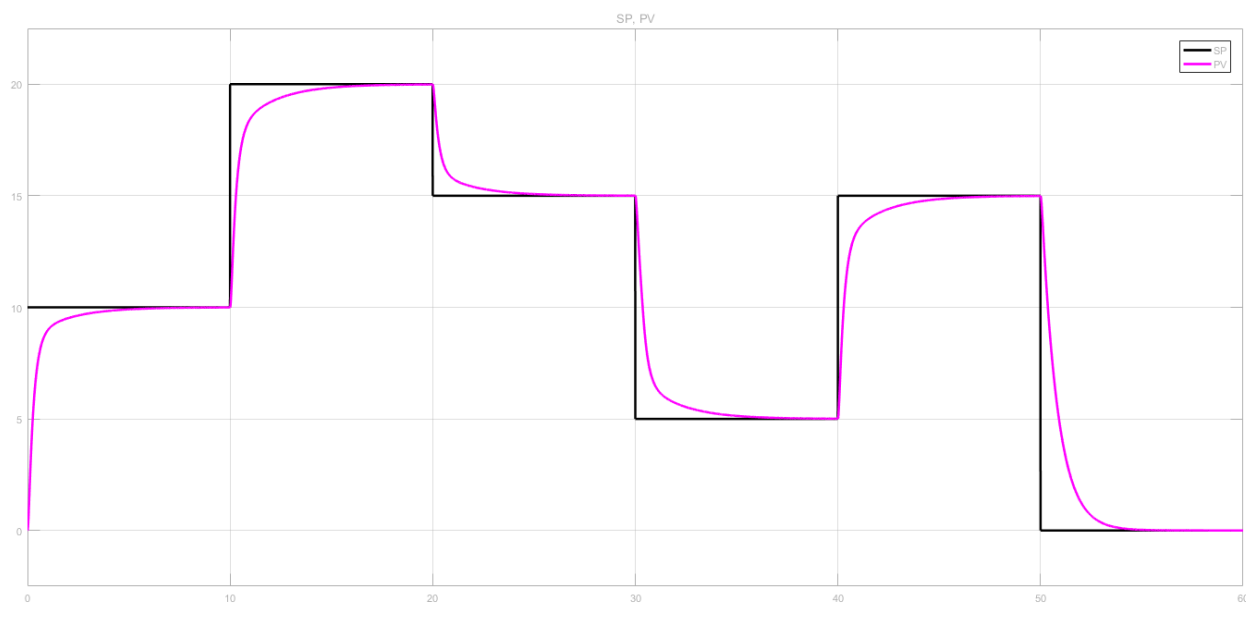


Hình 4.1.10 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.4, K_c = 0.65$)

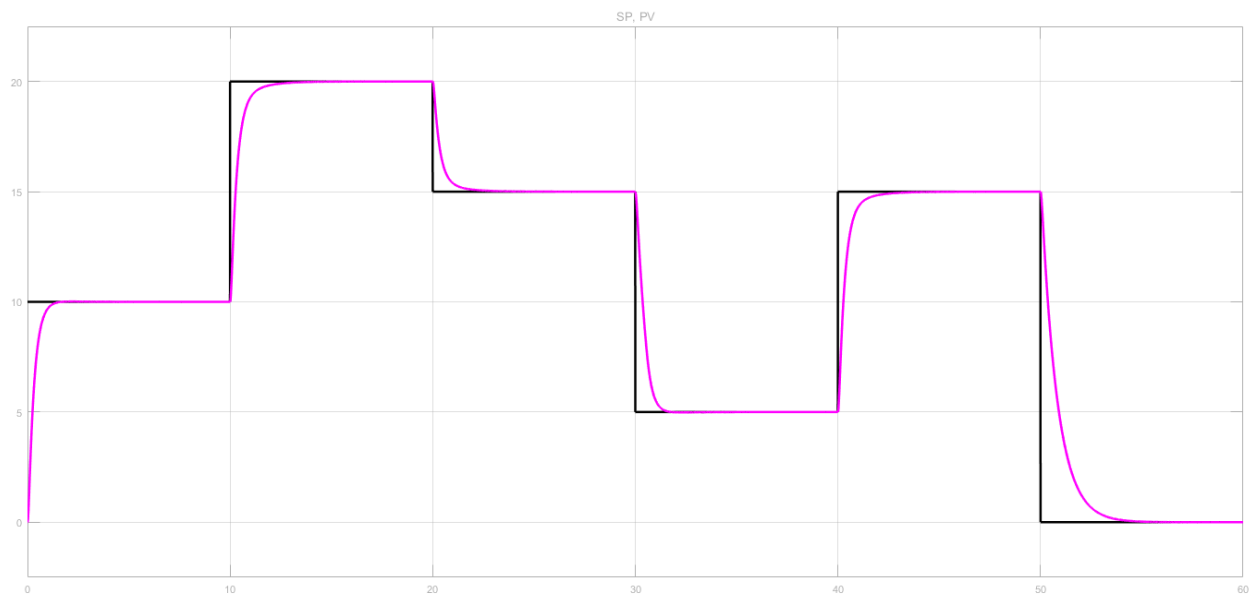


Hình 4.1.11 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.9, K_c = 0.65$)

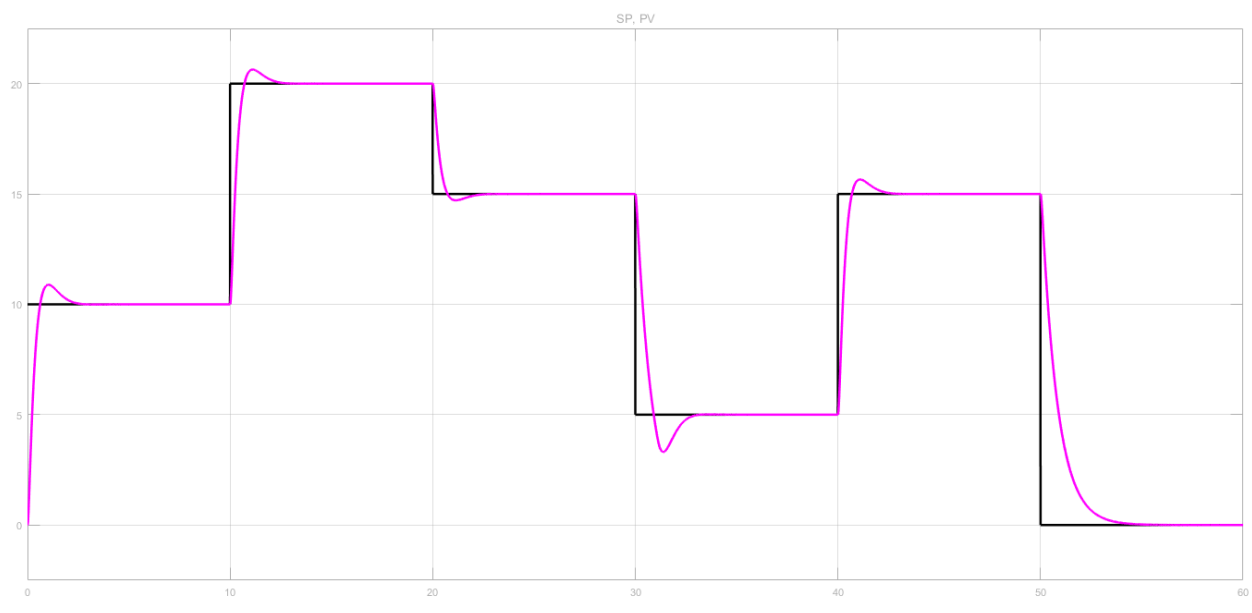
- Thực hiện với cùng các giá trị đặt và chỉ thay đổi số nơ-ron



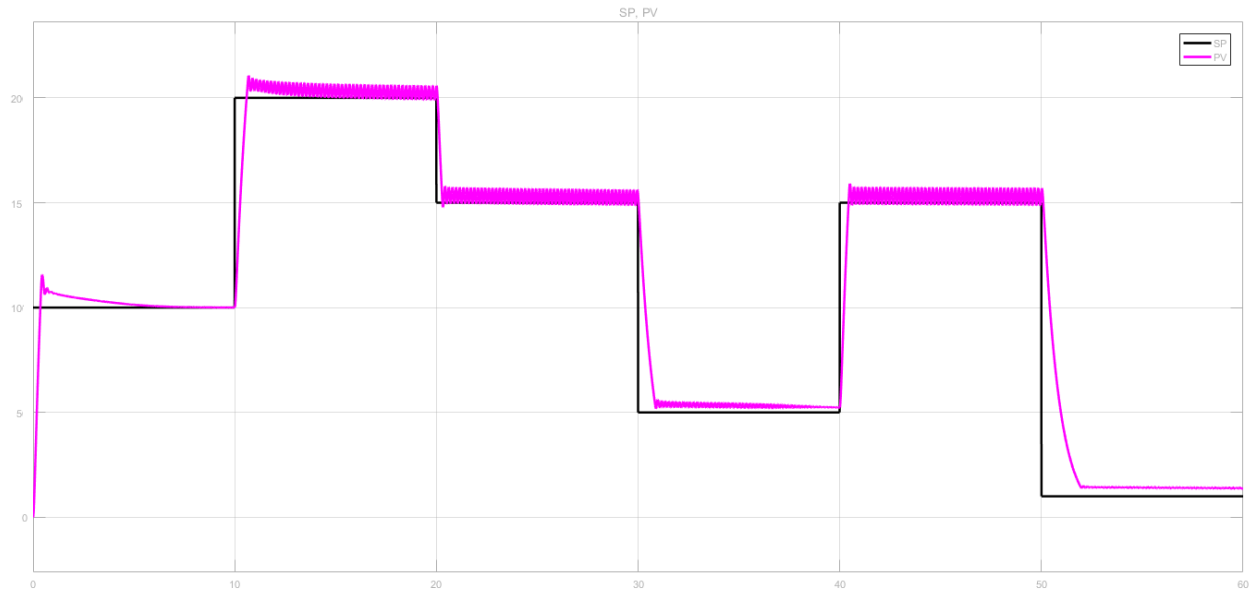
Hình 4.1.12 Kết quả đáp ứng mô phỏng ($N = 5, \eta = 0.2, K_c = 0.65$)



Hình 4.1.13 Kết quả đáp ứng mô phỏng ($N = 8, \eta = 0.2, K_c = 0.65$)



Hình 4.1.14 Kết quả đáp ứng mô phỏng ($N = 15, \eta = 0.2, K_c = 0.65$)



Hình 4.1.15 Kết quả đáp ứng mô phỏng ($N = 25, \eta = 0.2, K_c = 0.65$)

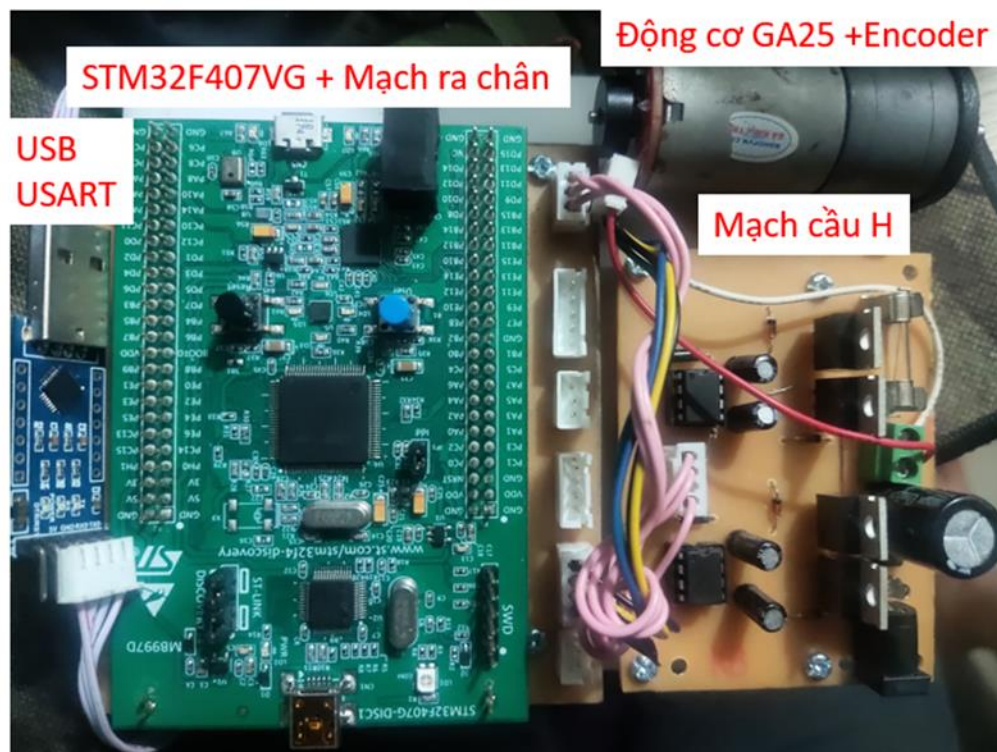
Nhận xét:

- Khi ta thực hiện tăng K_c , hay η thì hệ thống dần trở nên ổn định và đến một giá trị nhất định sẽ xuất hiện vọt lố và sẽ không còn xác lập nữa
- Số nơ-ron trong mạng giúp bộ điều khiển linh động hơn. Tuy nhiên khi số lượng nơ-ron lên quá nhiều sẽ ảnh hưởng đến hệ thống khiến quá trình không thể xác lập được.

4.2. Phần cứng

Dưới đây là bảng mạch phần cứng hoàn thiện, bao gồm:

- Vi điều khiển STM32F407VG.
- Mạch driver 2 kênh.
- Động cơ 350RPM và encoder 11 xung/vòng.
- USB UART CP2102 kết nối vi điều khiển và máy tính.
- Mạch ra chân kết nối cái phần tử mạch.



Hình 4.2.1 Mạch phần cứng hoàn thiện

4.3. Giao diện máy tính

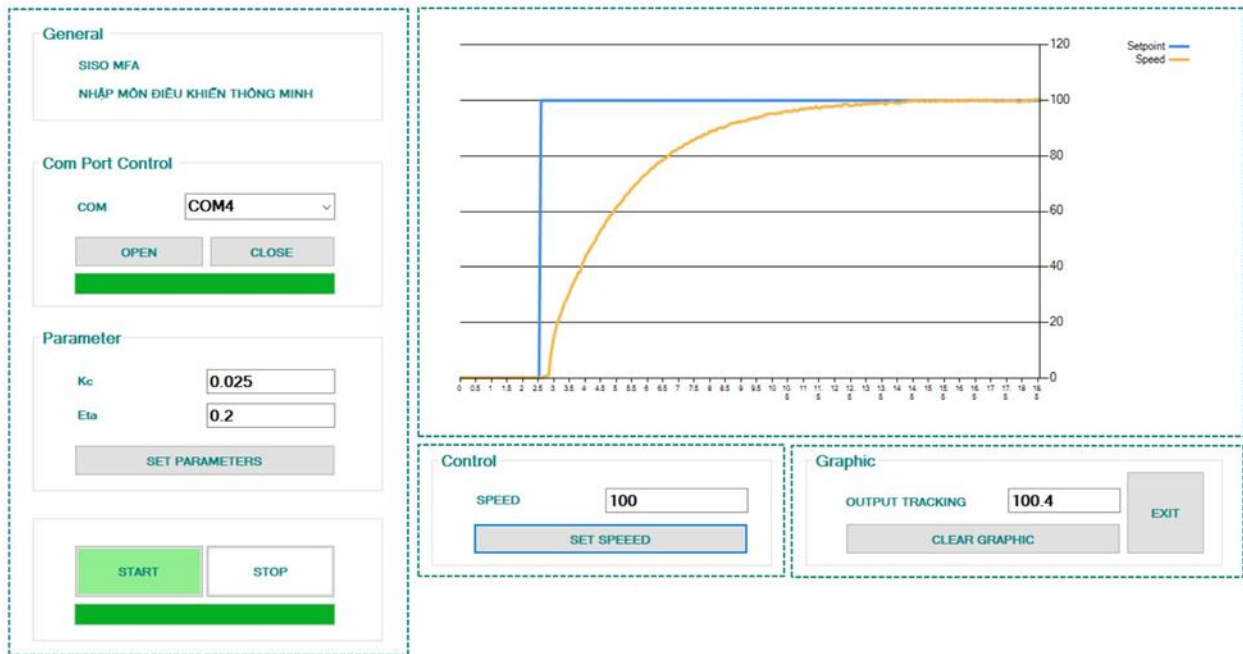
Giao diện máy tính hoàn thiện có đầy đủ các chức năng: Kết nối cổng COM, điều chỉnh thông số của bộ điều khiển, đặt tốc độ mong muốn, các chức năng với đồ thị.

<p>General Thông tin chung</p> <p>SISO MFA</p> <p>NHẬP MÔN ĐIỀU KHIỂN THÔNG MINH</p>	<p>Đồ thị tốc độ đặt và đáp ứng</p> <p>Setpoint — Speed —</p>	
<p>Com Port Control Kết nối cổng COM</p> <p>COM: <input type="text"/></p> <p>OPEN CLOSE</p>		
<p>Parameter Thông số bộ điều khiển</p> <p>Kc: <input type="text"/></p> <p>Eta: <input type="text"/></p> <p>SET PARAMETERS</p>	<p>Control Cài đặt tốc độ</p> <p>SPEED: <input type="text"/></p> <p>SET SPEED</p>	<p>Graphic Chức năng với đồ thị</p> <p>OUTPUT TRACKING: <input type="text"/></p> <p>CLEAR GRAPHIC EXIT</p>
<p>Bắt đầu và dừng điều khiển</p> <p>START STOP</p>		

Hình 4.3.1 Giao diện máy tính hoàn thiện

4.4. Đáp ứng động cơ

Với cùng giá trị đặt và K_c khác nhau ta có được kết quả như sau:



Hình 4.4.1 Đáp ứng động cơ với $K_c = 0.025$, $\eta = 0.2$ và $y_r = 100$



Hình 4.4.2 Đáp ứng động cơ với $K_c = 0.035$, $\eta = 0.2$ và $y_r = 100$



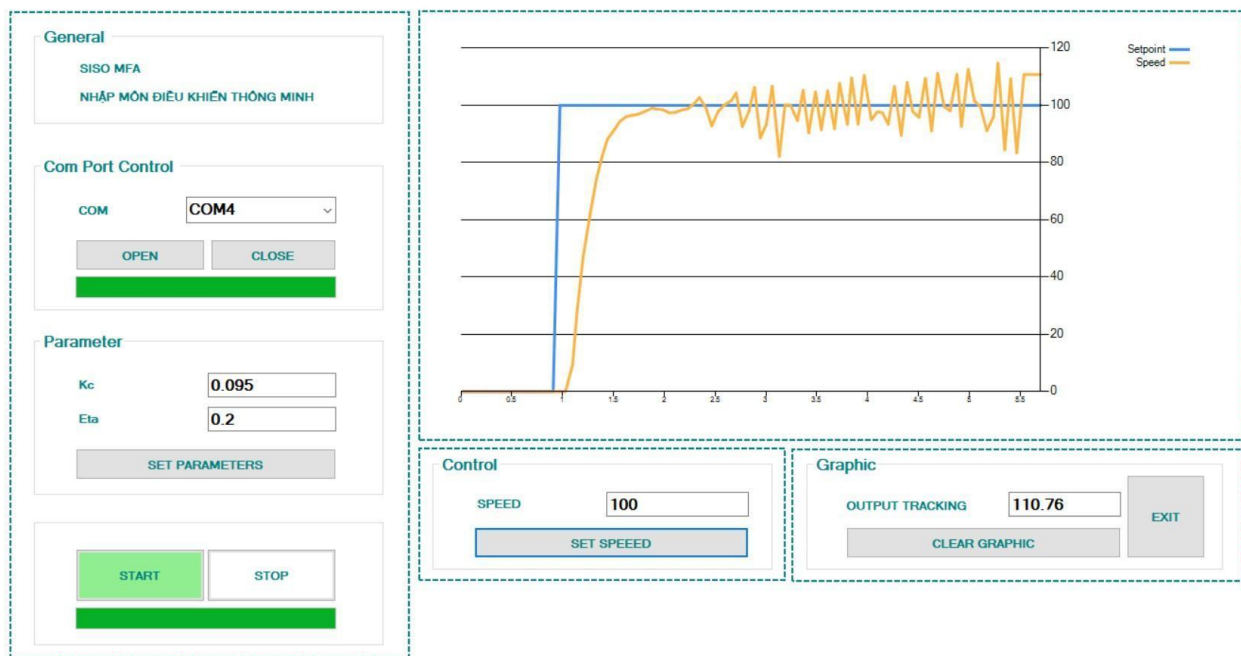
Hình 4.4.3 Đáp ứng động cơ với $K_c = 0.05$, $\eta = 0.2$ và $y_r = 100$



Hình 4.4.4 Đáp ứng động cơ với $K_c = 0.065$, $\eta = 0.2$ và $y_r = 100$



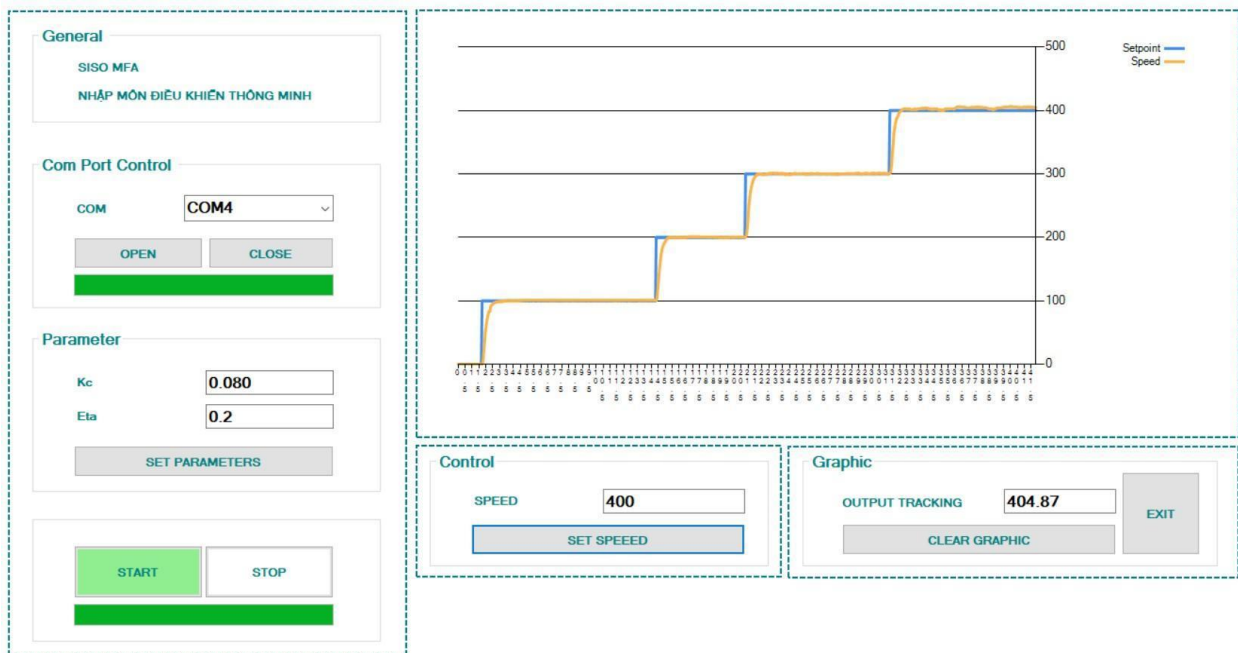
Hình 4.4.5 Đáp ứng động cơ với $K_c = 0.085$, $\eta = 0.2$ và $y_r = 100$



Hình 4.4.6 Đáp ứng động cơ với $K_c = 0.095$, $\eta = 0.2$ và $y_r = 100$



Hình 4.4.7 Đáp ứng động cơ với $K_c = 0.15$, $\eta = 0.2$ và $y_r = 100$



Hình 4.4.8 Đáp ứng động cơ với $K_c = 0.080$, $\eta = 0.2$ và y_r thay đổi

Dưới đây là tổng hợp các lần thử và giá trị đánh giá

Lần thử	Kc	Vọt lố (%)	Thời gian xác lập (s)
1	0.025	0	7.5
2	0.035	0	4.5
3	0.05	0	2.25
4	0.065	0	1.7
5	0.085	0	1.3
6	0.095	Không xác lập	
7	0.15	Không xác lập	

Bảng 4.4.1 Bảng tổng hợp đáp ứng động cơ khi thay đổi Kc

Nhận xét:

- Thời gian xác lập của bộ điều khiển nhanh hơn khi tăng Kc.
- Do đặc tính động cơ và bộ điều khiển nên không có sai số xác lập.
- Khi tăng Kc đến một giá trị nhất định thì đáp ứng không xác lập.

5. Đánh giá kết quả và hướng phát triển

5.1. Đánh giá kết quả

- Xây dựng được bộ điều khiển MFA theo lý thuyết nêu trên.
- Thực hiện mô phỏng và áp dụng được bộ điều khiển vào mô hình thực tế.
- Thực hiện được giao diện điều khiển với các chức năng cần thiết.
- Tốc độ đáp ứng thực tế của động cơ được điều khiển bằng bộ MFA có kết quả bám tốt giá trị đặt mong muốn.
- Hạn chế: chưa thực hiện được đổi chiều quay cho động cơ DC
- Khi giá trị đặt có thay đổi lớn thì đáp ứng bị rung.

5.2. Hướng phát triển

- Hoàn thiện bộ điều khiển tốc độ cho động cơ
- Điều chỉnh bộ điều khiển để có thể điều khiển động cơ đảo chiều
- Thêm các chức năng trong giao diện để thiết lập các chi tiết của bộ điều khiển như thời gian lấy mẫu, số neuron lớp ẩn, vùng chết deadband,...

6. Tài liệu tham khảo

- [1] Huỳnh Thái Hoàng, Giáo trình "Hệ thống điều khiển thông minh", NXB Đại học Quốc Gia TP HCM.
- [2] Vance J. VanDoren, Techniques for Adaptive Control, Control Engineering Magazine
- [3] Al Smadi, Prof-Takialddin & Al-Agha, Osman & Alsmadi, Khalid. (2018). Overview of Model Free Adaptive (MFA) Control Technology. IAES International Journal of Artificial Intelligence. 7. 165-169. 10.11591/ijai.v7.i4.pp165-169.