ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN – ĐIỆN TỬ

BÁO CÁO HỌC PHẦN MỞ RỘNG

LÝ THUYẾT ĐIỀU KHIỂN NÂNG CAO

ĐỀ TÀI:

THIẾT KẾ VÀ THỰC HIỆN BỘ ĐIỀU KHIỂN THÍCH NGHI THEO MÔ HÌNH CHUẨN (MRAS – PID) CHO ĐỐI TƯỢNG ĐỘNG CƠ DC

NGUYỄN GIA KHIÊM – 1810236

HỒ NGHĨA GIA BẢO - 1810034

GIẢNG VIÊN HƯỚNG DẪN:

TS. NGUYỄN VĨNH HẢO

ThS. TRẦN QUỐC TIẾN DŨNG

TP Hồ Chí Minh, tháng 1 năm 2021

MỤC LỤC

LỜI NÓI ĐẦU

Như chúng ta đã biết, một trong những hệ thống điều khiển được sử dụng nhiều nhất là hệ thống điều khiển tốc độ và vị trí động cơ DC. Nhiều cải tiến được đưa vào hệ thống này bao gồm thiết kế của hệ thống, điều khiển đầu ra của nó và cải thiện hiệu suất của nó.

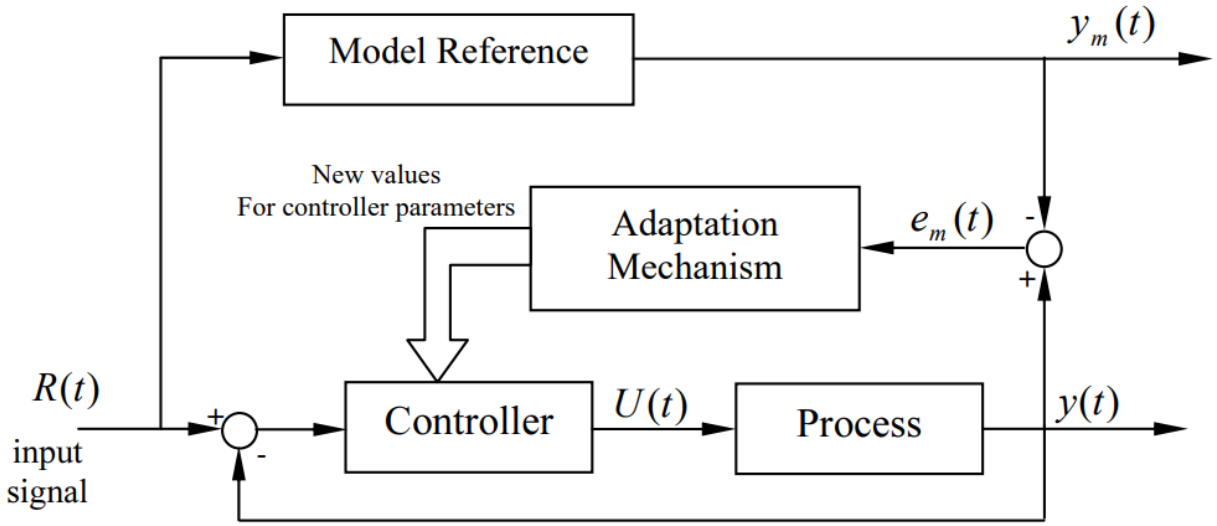
Một trong những phương pháp kinh điển để cải thiện hiệu suất của hệ thống điều khiển tốc độ động cơ DC là sử dụng bộ điều khiển PID. Khi đó, các tham số của bộ điều khiển PID được tính toán để đạt được đầu ra mong muốn của hệ thống.

Có nhiều phương pháp khác nhau để điều chỉnh bộ điều khiển PID, ví dụ, phương pháp Nicolas - zigglar. Khi các thông số của hệ thống điều khiển bị thay đổi, các thông số của bộ điều khiển PID phải được điều chỉnh lại để đạt được đáp ứng mong muốn. Thay vì phải lặp đi lặp lại tính toán PID khi đặc tính đối tượng thay đổi, hệ thống điều khiển thích nghi tham chiếu mô hình chuẩn (MRAC) là một cách tiếp cận phổ biến hơn trong thực tế.

1. CƠ SỞ LÝ THUYẾT

1.1. Hệ thống điều khiển thích nghi theo mô hình chuẩn (MRAC)

Hệ thống điều khiển thích nghi mô hình (MRAC) là một bộ điều khiển thích nghi quan trọng. Sơ đồ khối của hệ thống MRAC được thể hiện trong hình ở dưới. Trường hợp hệ thống có một vòng hồi tiếp như thông thường, bao gồm khối Process và Controller, và một vòng hồi tiếp khác thay đổi các tham số của bộ điều khiển. Các tham số được thay đổi trên cơ sở tín hiệu hồi tính từ sai số $e_m(t)$, là sai số giữa đầu ra của hệ thống $y(t)$ và đầu ra của mô hình tham chiếu $y_m(t)$. Vòng hồi tiếp thông thường được gọi là vòng trong, và vòng điều chỉnh tham số được gọi là vòng ngoài.



Hình 1. Sơ đồ khối bộ điều khiển thích nghi theo mô hình chuẩn (MRAC)

1.2. Hệ thống MRAC sử dụng Phương pháp Gradient

Giả sử rằng bộ điều khiển có các tham số điều chỉnh θ_i. Sự thích nghi (tự điều chỉnh) của các tham số bộ điều khiển được thực hiện theo cách để giảm tối đa chỉ tiêu sai số theo biểu thức:

$$J = \frac{1}{2}\int_0^t e_m^2(\tau).d(\tau)$$

Trong đó: $e_m(t) = y(t) - y_m(t)$

Để làm cho J nhỏ nhất, cách phù hợp nhất là điều chỉnh các thông số bộ điều khiển θ_i ngược hướng gradient của J:

$$\Delta\theta_i = -\gamma_i . \frac{\partial J}{\partial \theta_i}$$

Việc điều chỉnh các tham số bộ điều khiển được thực hiện theo phương trình sau, được gọi là quy tắc MIT :

$$\frac{d\theta_i}{dt} = -\gamma_i . \frac{d}{dt}\left(\frac{\partial J}{\partial \theta_i}\right)$$

1.3. Động cơ DC

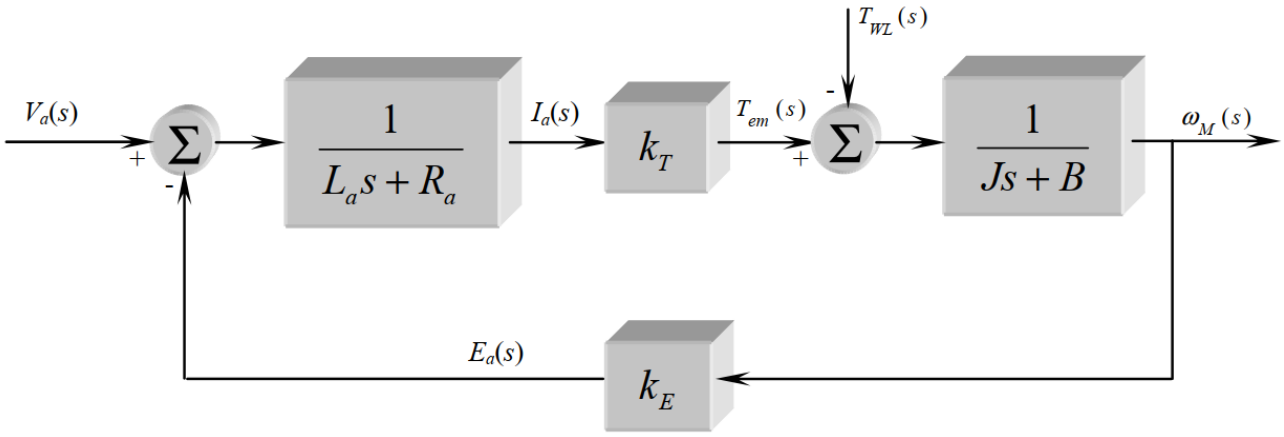
Hàm truyền đối tượng có dạng bậc 2 như sau:

$$G(s) = \frac{K}{(T_1 s + 1)(T_2 s + 1)} \quad (1)$$

Trong đó:

T_1: thời hằng điện (s)

T_2: thời hằng cơ (s)



Hình 2. Sơ đồ khối hàm truyền động cơ

Hoàn toàn tương tự ta có thể viết lại hàm truyền dưới dạng như sau:

$$G(s) = \frac{b_0}{s^2 + a_1 s + a_2} \quad (2)$$

Với: b_0, a_1, a_2 là các hằng số có được từ động cơ thực tế.

1.4. Bộ điều khiển PID thích nghi cho động cơ DC sử dụng phương pháp Gradient

Hàm truyền bộ điều khiển PID:

$$PID(s) = k_P + \frac{k_I}{s} + k_D s \ (3)$$

Bộ điều khiển có 2 thông số cần cập nhật theo thời gian đó là k_P và k_I.

Kết hợp (1) và (3), ta có hàm truyền tổng của hệ như sau:

$$G_P(s) = \frac{Y(s)}{R(s)} = \frac{b_0(k_I + k_P s + k_D s^2)}{s^3 + (a_1 + b_0 k_D)s^2 + (a_2 + b_0 k_P)s + b_0 k_P}$$

Mô hình chuẩn có hàm truyền:

$$G_m(s) = \frac{Y_m(s)}{R(s)} = \frac{b_{0m}}{s^2 + a_{1m}s + a_{2m}}$$

Ta có luật MIT gần đúng cập nhật thông số theo phương pháp gradient:

$$\dot{k_P} = -\gamma_1 e_m(s) \frac{\partial y}{\partial k_P}$$

$$\dot{k_I} = -\gamma_2 e_m(s) \frac{\partial y}{\partial k_I}$$

$$\dot{k_D} = -\gamma_3 e_m(s) \frac{\partial y}{\partial k_D}$$

Ta có:

$$\frac{\partial y}{\partial k_P} = \frac{b_{0m}}{b_0} . s . G_m(s)[R(s) - Y(s)]$$

$$\frac{\partial y}{\partial k_I} = \frac{b_{0m}}{b_0} G_m(s)[R(s) - Y(s)]$$

$$\frac{\partial y}{\partial k_D} = \frac{b_{0m}}{b_0} . s^2 . G_m(s)[R(s) - Y(s)]$$

Thay vào luật cập nhật ta được:

$$\dot{k_P} = -\gamma_1'.s.e_m(s)G_m(s)[R(s) - Y(s)]$$

$$\dot{k_I} = -\gamma_2' e_m(s)G_m(s)[R(s) - Y(s)]$$

$$\dot{k_D} = -\gamma_3'.s^2.e_m(s)G_m(s)[R(s) - Y(s)]$$

Trong đó:

$$e_m(t) = y(t) - y_m(t)$$

$$\gamma_1' = \gamma_1\left(\frac{b_{0m}}{b_0}\right); \gamma_2' = \gamma_2\left(\frac{b_{0m}}{b_0}\right); \gamma_3' = \gamma_3\left(\frac{b_{0m}}{b_0}\right)$$

2. NỘI DUNG THỰC HIỆN

2.1. Tính toán thông số

2.1.1. Mô hình chuẩn

Các bước tính toán thông số mô hình chuẩn chuyển sang miền thời gian, sử dụng công cụ hỗ trợ tính toán của MATLAB:

- *Bước 1:* Nhập hàm truyền, ta được:

$$G_m = \frac{8.2}{s^2 + 4.8s + 8.2}$$

- *Bước 2:* Biến đổi hàm truyền từ miền s về miền z, sử dụng lệnh >>c2d, thời gian lấy mẫu là T = 0.01s.

- *Bước 3:* Viết phương trình ngõ ra ở miền thời gian.

Từ đây ta được phương trình sai phân:

Phương trình sai phân vừa tìm được sử dụng cho việc lập trình vi điều khiển bằng ngôn ngữ C.

2.1.2. Luật gradient cập nhật thông số của bộ điều khiển

Tương tự mục 3.1.1., với các luật điều khiển đã cho, ta thực hiện các bước sau:

- *Bước 1:* Nhập hàm truyền ở miền s.
- *Bước 2:* Biến đổi hàm truyền từ miền s về miền z, sử dụng lệnh >>c2d, thời gian lấy mẫu là T = 0.01s.
- *Bước 3:* Viết phương trình ngõ ra ở miền thời gian.
- *Bước 4:* Áp dụng để viết code vi điều khiển.

2.1.3. Bộ lọc thông thấp

Bộ lọc setpoint

Lựa chọn bộ lọc thông thấp bậc 1 có hàm truyền:

$$G_{LF1}(s) = \frac{1}{0.1s + 1}$$

Tác dụng của bộ lọc này là loại bỏ bớt nhiễu của tín hiệu đặt vào để giảm bớt sai số trong quá trình xử lý kế tiếp.

Đặc trưng của bộ lọc này là không nhạy với nhiễu nên đáp ứng khi qua bộ lọc ít bị trễ.

Bộ lọc vận tốc

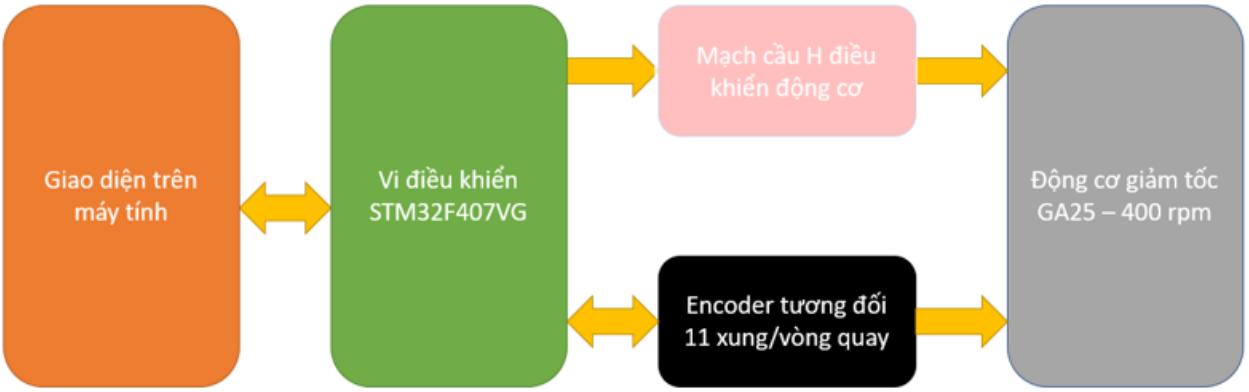
Lựa chọn bộ lọc thông thấp bậc 1 có hàm truyền:

$$G_{LF}(s) = \frac{1}{0.5s + 1}$$

Do phương pháp đọc encoder không chính xác, encoder thường xuyên bị lệch 1 xung so với số xung thực tế dẫn đến vận tốc đọc được từ động cơ sai lệch, do đó mà quá trình xử lý tiếp theo cũng không tốt. Thêm vào bộ lọc thông thấp để loại bỏ sai số này.

Đặc trưng của bộ lọc này là nhạy với nhiễu, làm giảm sai số, tuy nhiên, đáp ứng khi qua bộ lọc bị trễ đi. Ta chọn bộ lọc vừa lọc được nhiễu đến mức tín hiệu có đủ tốt và đáp ứng không quá trễ khi qua bộ lọc.

2.2. Tổng quan kết nối



Hình 3. Tổng quan kết nối phần cứng

Đề tài sử dụng encoder tưởng 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.

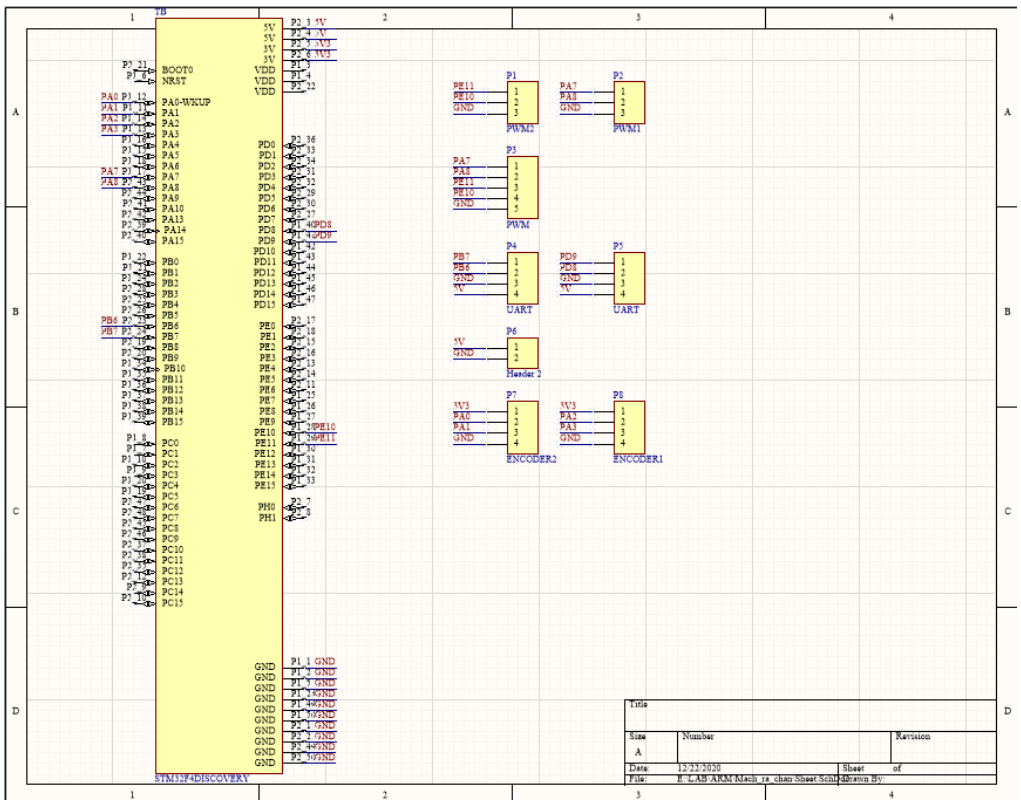
Vi điều khiển STM32F407VG là một trong những vi điều khiển mạnh mẽ ở thời điểm hiện tại, có nhiều chức năng, hỗ trợ từ cộng đồng rất lớn. Vi điều khiển được sử dụng với các chức năng: đọc xung từ encoder sử dụng chức năng Input Capture với Encoder Interface (để x4 lần xung đọc được) được hỗ trợ sẵn,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

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 ngắt
TIM5	PA0, PA1	Đọc xung từ Encoder

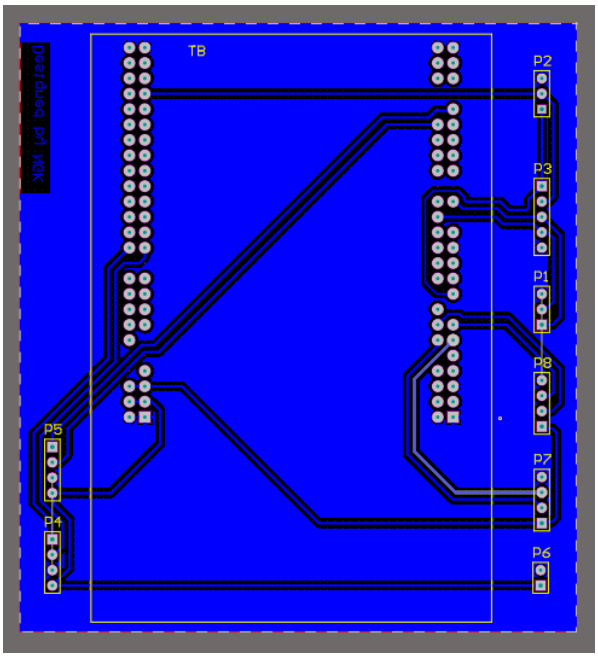
Bảng 2.1. Chức năng và các chân sử dụng trên vi điều khiển STM32F407VG

2.3. Thiết kế và thi công mạch ra chân

Sử dụng phần mềm Altium ta vẽ được sơ đồ nguyên lý mạch và PCB ra chân:



Hình 4. Sơ đồ nguyên lý mạch ra chân



Hình 5. PCB của mạch ra chân

2.4. Lập trình vi điều khiển

2.4.1. Lập trình giao tiếp GUI máy tính

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

Hình 6. Frame truyền đi lên máy tính

Frame truyền có chiều dài 18 byte, cụ thể là:

- 7 byte Start: "$$DLCN," dùng để nhận diện frame
- 4 byte Speed: theo kiểu float
- 4 byte Position: theo kiểu float
- 1 byte CheckSum: dùng để kiểm tra lỗi
- 2 byte Stop: "\r\n"

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

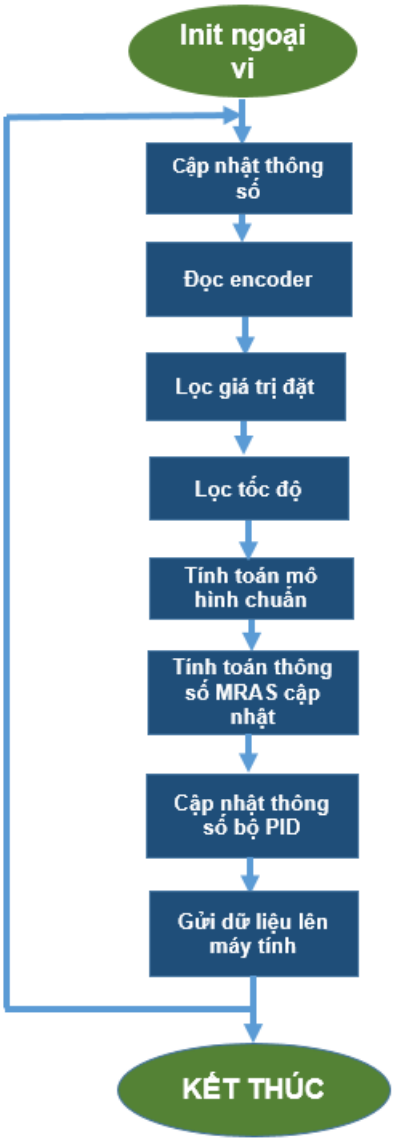
Hình 7. Frame nhận vào vi điều khiển

Frame nhận có chiều dài 14 byte, cụ thể là:

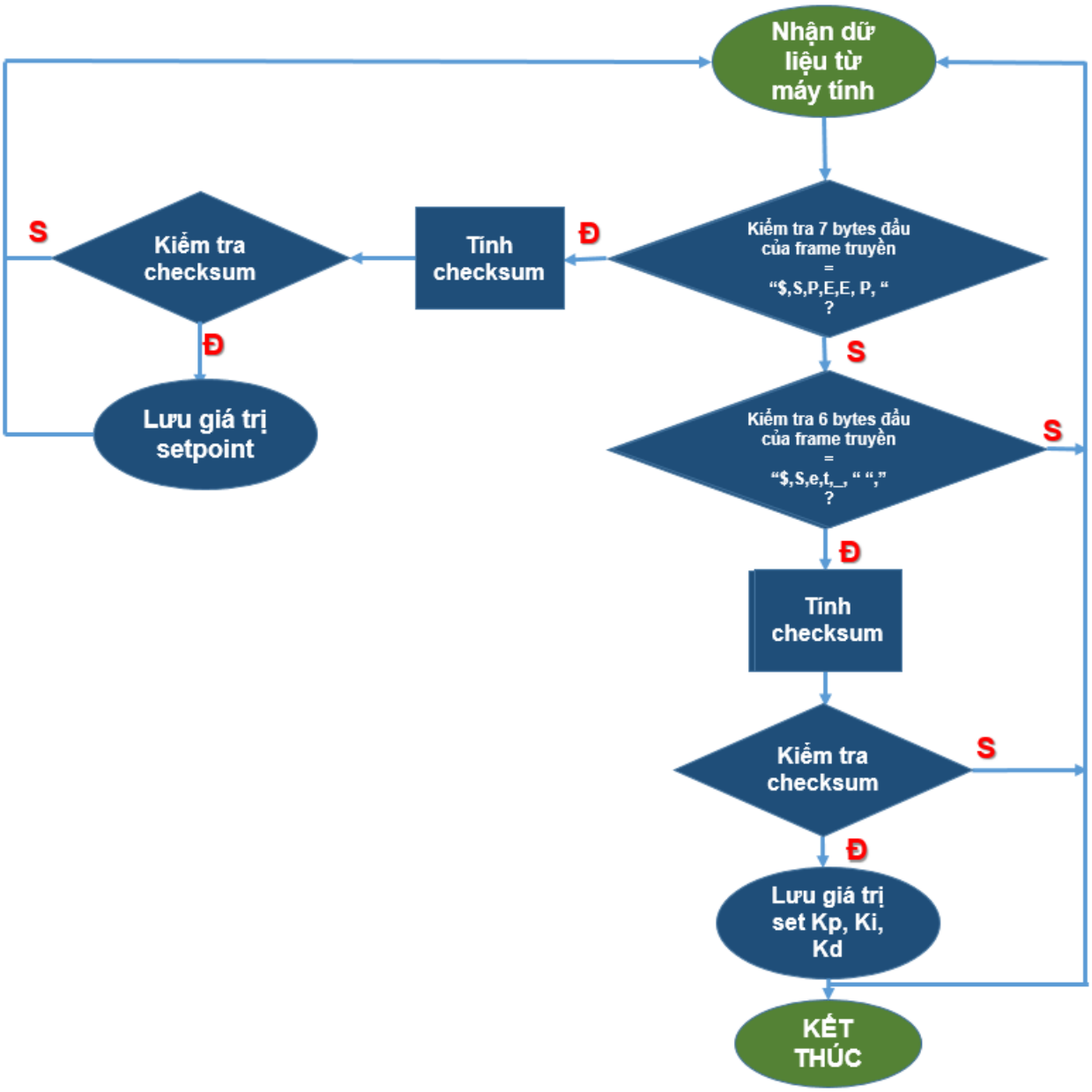
- 6 byte Start: "$Set_," dùng để nhận diện frame
- 1 byte Option: Để vi điều khiển biết là đặt Set point hay các thông số
- 4 byte Set parameter: theo kiểu float
- 1 byte CheckSum: dùng để kiểm tra lỗi
- 2 byte Stop: "\r\n"

2.4.2. Lưu đồ lập trình

Ta lập trình vi điều khiển theo lưu đồ giải thuật sau



Hình 8. Lưu đồ giải thuật bộ điều khiển



Hình 9. Lưu đồ giải thuật lập trình ngắt nhận dữ liệu

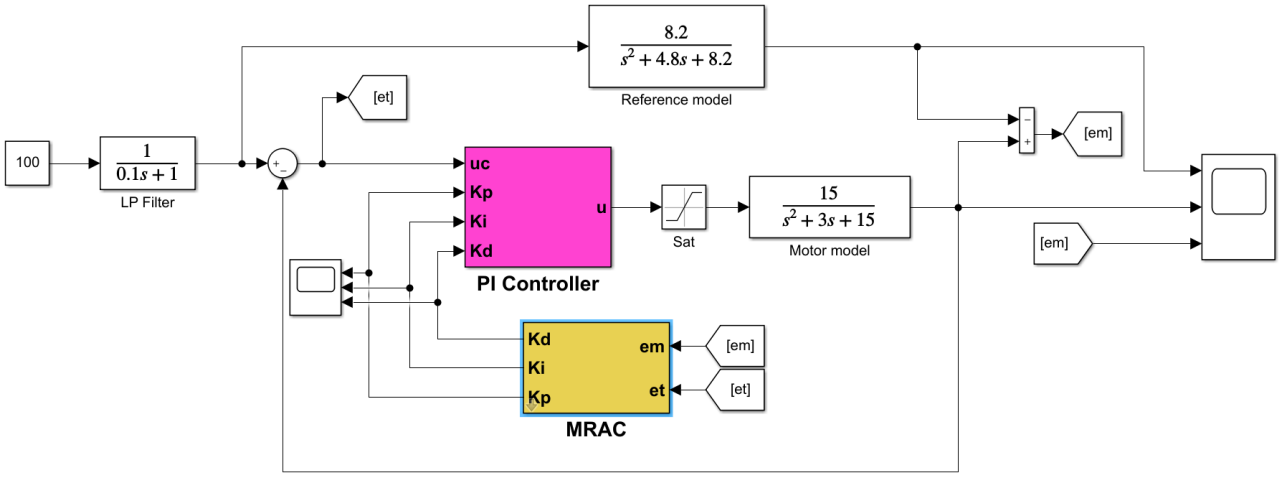
2.5. Lập trình giao diện

GUI được viết bằng ngôn ngữ C++ sử dụng framwork Qt – một framework mạnh mẽ, hỗ trợ nhiều chức năng khác nhau và có một cộng đồng lớn. Sơ đồ khối của giao diện được minh họa bằng hình bên dưới:

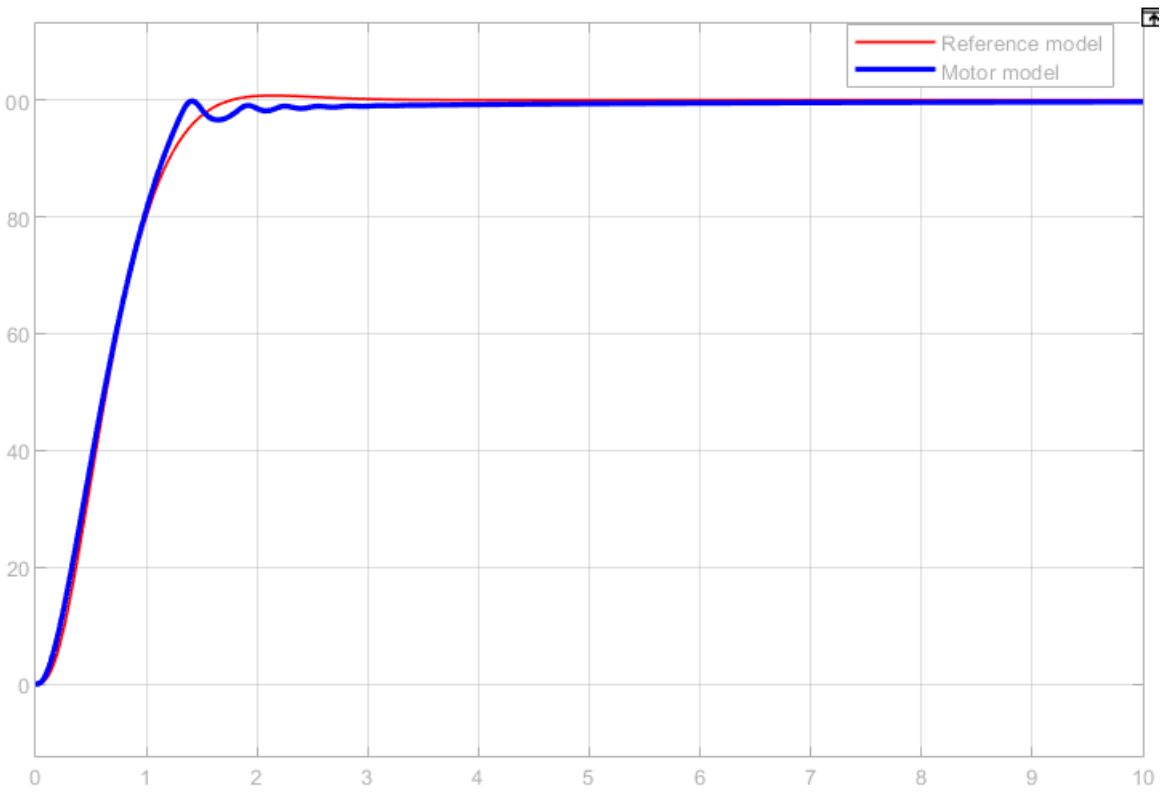
Hình 10. Sơ đồ khối giao diện

3. KẾT QUẢ

3.1. Mô phỏng Matlab Simulink



Hình 11. Mô phỏng MatLab Simulink

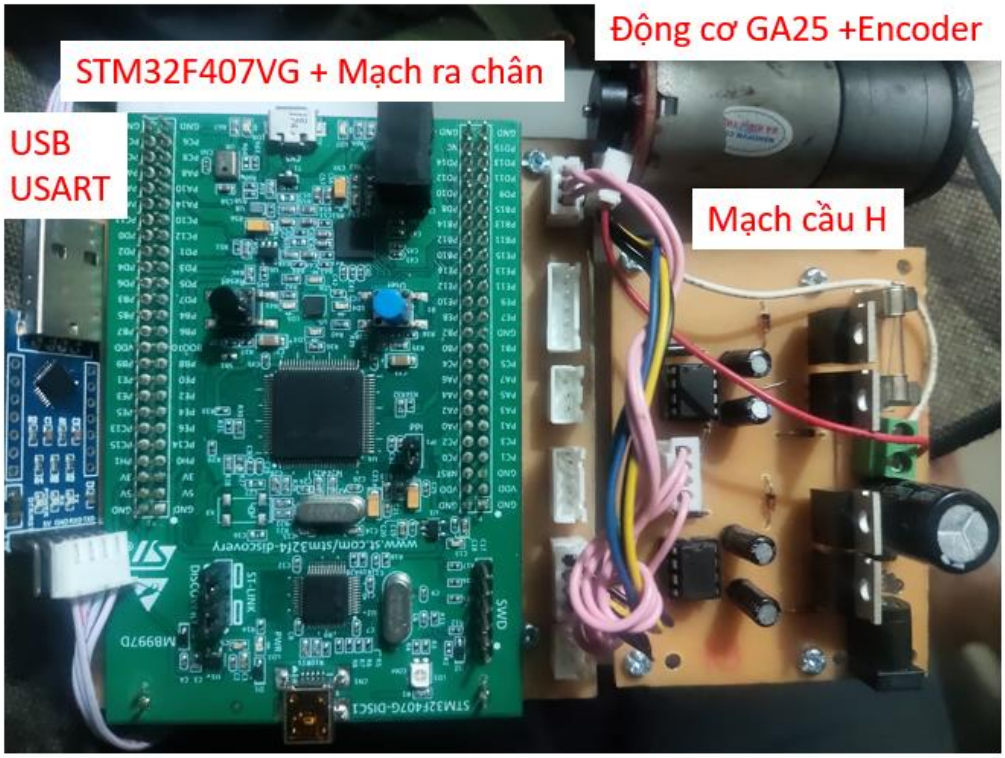


Hình 12. Đáp ứng của mô phỏng có sử dụng
bộ bão hòa tín hiệu điều khiển để sát với thực tế

3.2. Phần cứng

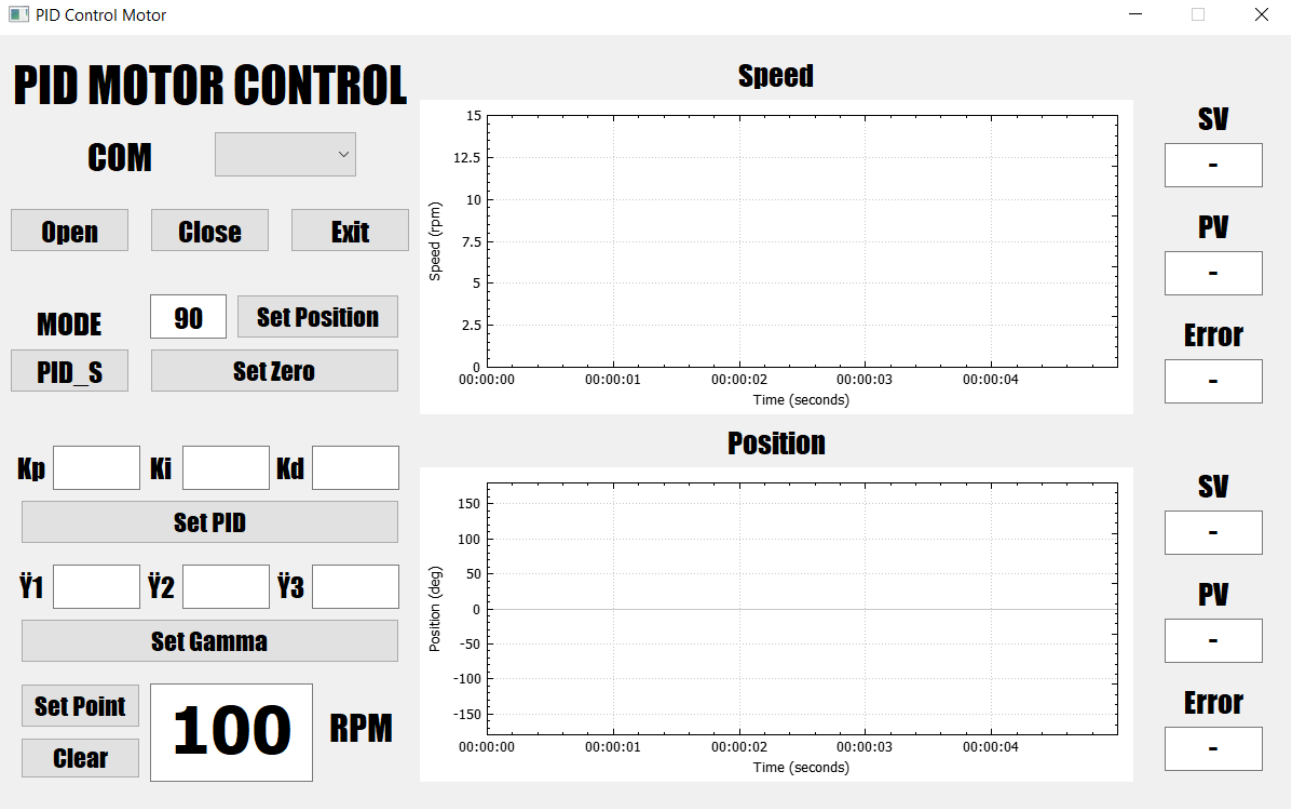
Dưới đây là bảng mạch phần cứng hoàn thiện, bao gồm**:**

- 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.



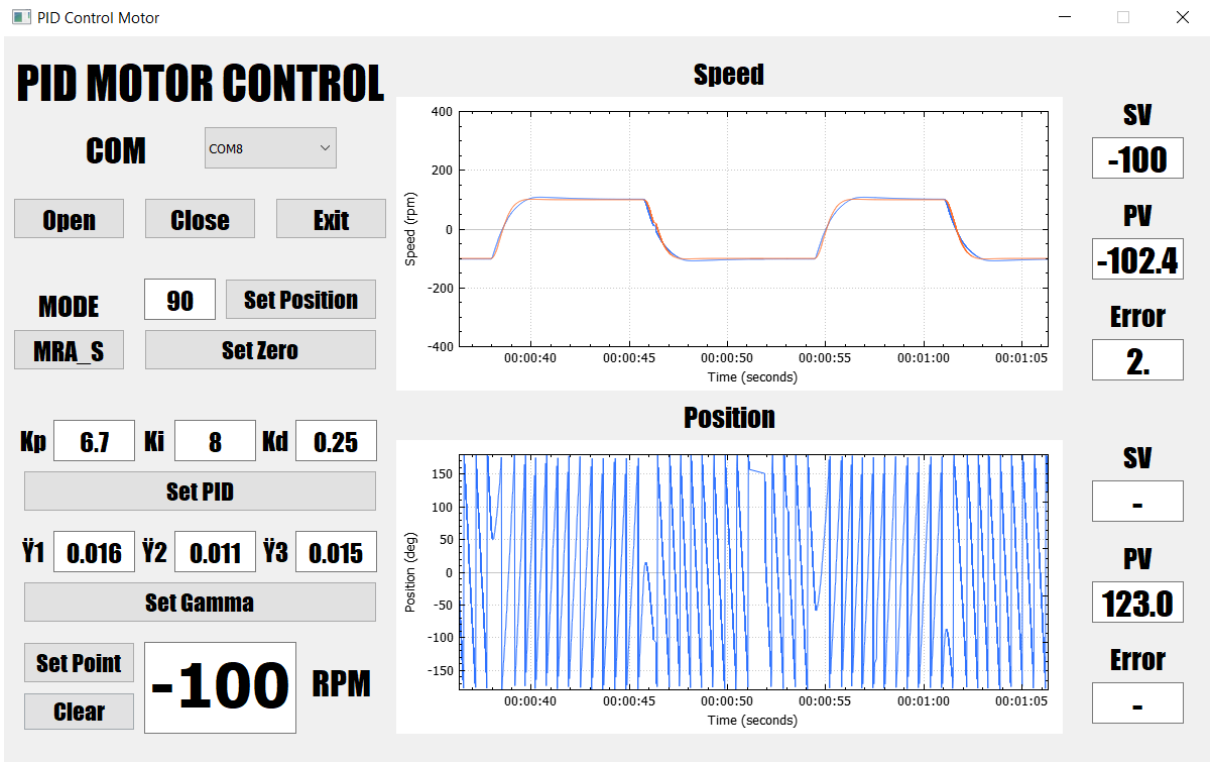
Hình 13. Phần cứng hoàn thiện

3.3. Giao diện máy tính

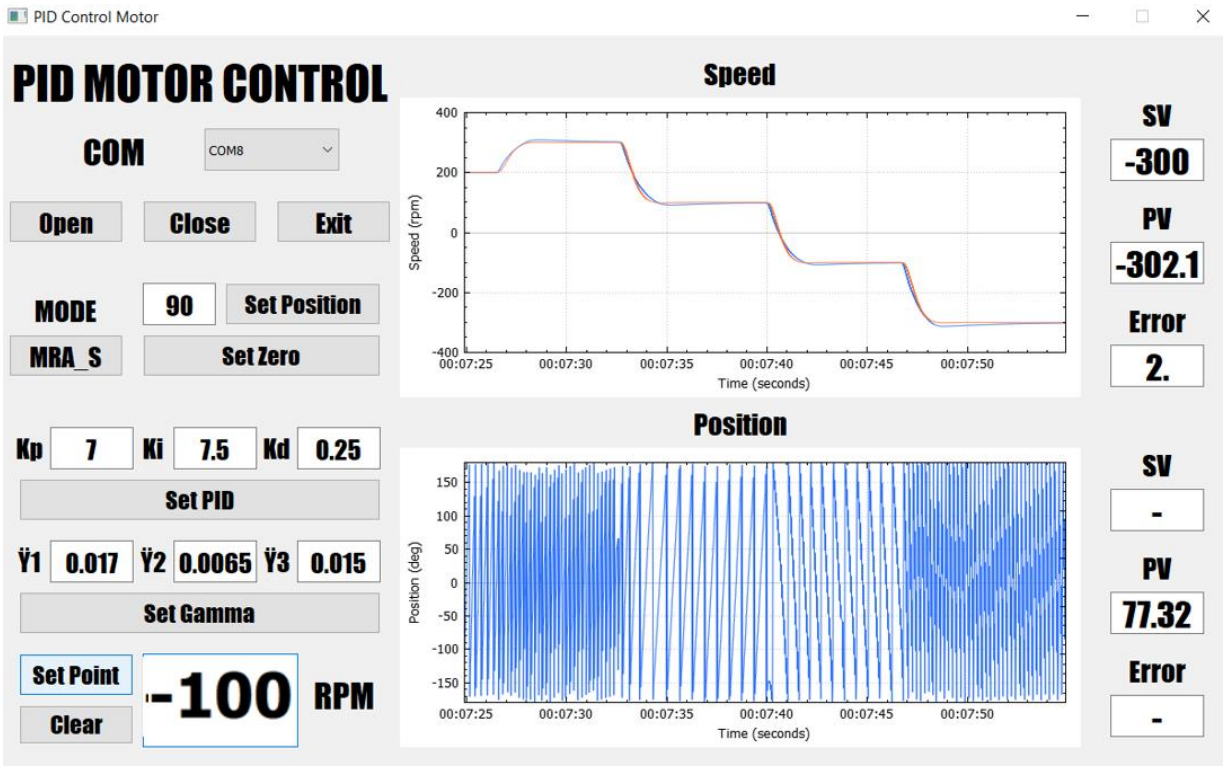


Hình 14. Giao diện hoàn thiện

3.4. Đáp ứng động cơ thực tế



Hình 15. Đáp ứng ngõ ra động cơ khi ngõ vào thay đổi giữa 2 giá trị



Hình 16. Đáp ứng ngõ ra động cơ khi ngõ vào thay đổi nhiều giá trị

4. ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

4.1. Đánh giá kết quả:

- Mô phỏng được bộ điều khiển bám tốt theo mô hình chuẩn đặt ra.
- Đáp ứng tốc độ thực tế của động cơ được điều khiển theo hệ thống MRAS PID bám khá tốt mô hình chuẩn khi điểm đặt thay đổi giá trị không quá lớn.
- Khi điểm đặt thay đổi lớn, hệ thống chưa đáp ứng tốt.
- Chưa hoàn thành bộ điều khiển vị trí cho động cơ.

4.2. Hướng phát triển

- Tối ưu hóa việc viết code vi điều khiển và code giao diện
- Hoàn thiện bộ điều khiển tốc độ cho động cơ.
- Hoàn thành bộ điều khiển vị trí cho động cơ.

TÀI LIỆU THAM KHẢO

[1] Muna, Munadi & Akbar, M Amirullah & Naniwa, Tomohide & Taniai, Yoshiaki. (2016). Model Reference Adaptive Control for DC motor based on Simulink.

[2] Sar, S.K. & Dewan, L.. (2014). MRAC based PI controller for speed control of D.C. motor using lab view.

[3] "STM32 step-by-step," [Online]. Available: https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-step-by-step.html

[4] "Qt Examples And Tutorials [Online]. Available: https://doc.qt.io/qt-5/qtexamplesandtutorials.html

[5] Huỳnh Thái Hoàng (2019) Chapter 4_Advanced Control Theory_Adaptive Control

[6] Huỳnh Thái Hoàng (2018) Chapter 6_Fundamental of Control System_Discrete time control System

[7] Huỳnh Thái Hoàng (2018) Chapter 5_Fundamental of Control System_Continous time control System