**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH**

**KHOA ĐIỆN – ĐIỆN TỬ**

**BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG**


# BÁO CÁO ĐỒ ÁN 1

# ĐIỀU KHIỂN ĐỘNG CƠ DC BẰNG BỘ ĐIỀU KHIỂN PID SỬ DỤNG VI XỬ LÝ

**Giảng Viên Hướng Dẫn: Nguyễn Chí Nghĩa**

**Sinh viên thực hiện: Huỳnh Khương**

**MSSV: 1812689**

TP. HỒ CHÍ MINH 2021

# Mục Lục

# Danh mục bảng, sơ đồ, hình vẽ

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Hiện nay, với sự phát triển của công nghệ số hóa, nền công nghiệp sản xuất phát triển vô cùng mạnh mẽ. Những ứng dụng công nghệ tiên tiến được áp dụng vào quá trình sản xuất công nghiệp ngày càng rộng rãi nhằm chuyển một phần hoặc toàn bộ công đoạn từ thủ công bằng sức người, sang tự động hóa máy móc bằng hệ thống điều khiển tiên tiến. Dựa trên tinh thần đó, lĩnh vực nghiên cứu về kỹ thuật điều khiển tự động được thúc đẩy phát triển mạnh một cách vượt bậc từ những lý thuyết điều khiển đơn giản (như phương pháp điều khiển PID) đến các thuật toán phức tạp (phương pháp fuzzy, neural network, …), phục vụ nhiều nhu cầu khác nhau, điều khiển những hệ thống đơn giản như động cơ trong công nghiệp hay những hệ phi tuyến phức tạp, thiếu ổn định như hệ thống tàu ngầm, robot, …

Nhắc đến điều khiển trong công nghiệp không thể nhắc đến bộ điều khiển PID. Trong công nghiệp, hơn 90% những ứng dụng công nghiệp được điều khiển bằng bộ điêu khiển PID vì sự đơn giản nhưng đem lại hiệu quả lớn nhờ vào môi trường ít nhiễu, và cố định.

Bộ điều khiển PID được ra đời từ những năm 1890 trong các thiết kế bộ điều tốc, và sau đó được phát triển trong hệ thống lái tàu (thủy) tự động. PID được nghiên cứu lý thuyết đầu tiên vào năm 1922 với tác giả là kỹ sư người Mỹ Minorsky. Sau đó nghiên cứu tương tự được tiến hành và xuất bản bởi nhiều người khác vào thập niên 1930.

Mục tiêu của đồ án là khảo sát bộ điều khiển PID truyền thống, giúp cho sinh viên có một cái nhìn trực quan và hiểu rõ hơn về bộ điều khiển PID rời rạc thông qua mô hình điều khiển động cơ DC thực tế, đơn giản, ít sai sót. Đồng thời, cũng giải thích rõ ý nghĩa của các công thức PID trong thuật toán, giúp sinh viên hiểu rõ về quá trình tìm các hệ số của bộ điều khiển, qua đó làm rõ những điểm ưu, nhược của bộ điều khiển PID, làm tiền đề để hướng sinh viên đến với những bộ điều khiển tốt hơn, đáp ứng tốt hơn, hứa hẹn đem đến hiệu suất cao hơn so với bộ điều khiển PID truyền thống.

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ PHẦN CỨNG

## 2.1 Vi điều khiển:

Chúng ta nên dùng những dòng vi xử lý lõi ARM Cortex M3/4 điển hình là các họ vi điều khiển STM32 vì khả năng tiêu thụ năng lượng thấp, được sử dụng trong những ứng dụng nhúng cần đáp ứng ngắt nhanh. Ngoài ra, vi xử lý còn được kết hợp bộ tính toán số thực FPU, cùng những xung nhịp tương đối từ 80MHz – 500MHz giúp cho vi điều khiển có đáp ứng nhanh, có khả năng điều khiển mượt động cơ DC. Cần nhắc rõ, vì PID là thuật toán đơn giản, nên dòng vi điều khiển STM32 không phải là sự lựa chọn duy nhất, ngoài ra còn có những dòng chip có khả năng đáp ứng tốt như PIC16, Adruino…, Ta nên chọn dòng vi xử lý STM32 vì những dự án sau này tìm hiểu và nâng cấp bộ điều khiển PID, yêu cầu tính toán phức tạp, nhanh trong chu kì nhỏ trung bình 5ms (thời gian trung bình để điều khiển mượt động cơ) như bộ điều khiển PID fuzzy, hay bộ điều khiển tự chỉnh, điều mà các vi điều khiển PIC hay Arduino không làm được. Như thế sẽ có cái nhìn rõ hơn thông qua sự so sánh trực quan giữa các bộ điều khiển này.

Ở đây ta chọn vi điều khiển STM32F103C8T6 vì giá cả rẻ, phù hợp với yêu cầu đồ án:

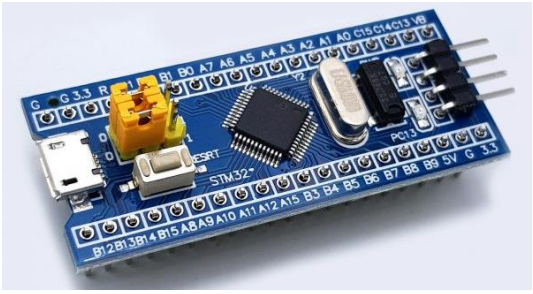

*Figure 1: Kit STM32F103C8T6 Blue Pill*

Các ngoại vi yêu cầu của vi điều khiển:

- 1 timer xuất xung PWM để điều khiển tốc độ động cơ.
- 1 timer tạo ngắt để tính toán thuật toán điều khiển chu kì khoảng 10ms.
- chân General Purpose Ouput để điều khiển hướng động cơ.
- kênh General Purpose Input để nhận tín hiệu encoder của động cơ.
- 1 kênh UART để truyền nhận thông tin từ máy tính.

## 2.2 Mạch cầu H

Là mạch công suất để cấp nguồn và điều khiển chiều động cơ. Bản thân điện áp ngõ ra của vi điều khiển không đủ để cấp nguồn cho động cơ chạy. Vì vậy cần có một mạch cầu H để thực hiện yêu cầu trên. Ta chọn mạch cầu H điều khiển động cơ L298.

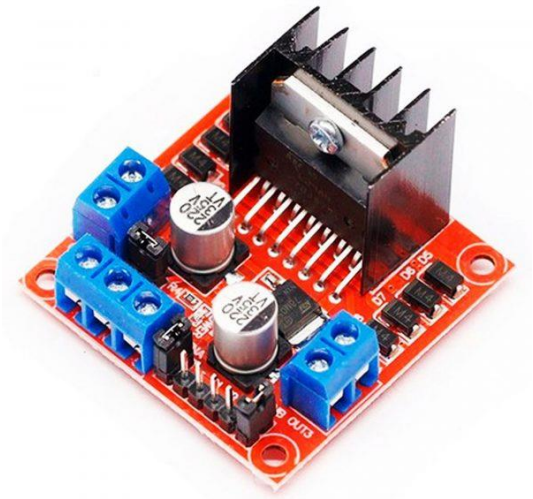

*Figure 2. Mạch điều khiển động cơ L298v2*

Các thông số kĩ thuật của mạch:

- IC chính: L298 - Dual Full Bridge Driver
- Điện áp cấp nguồn: 5~30VDC; Dòng tối đa cho mỗi cầu H: 2A
- Công suất tối đa: 25W 1 cầu.

## 2.3 Động cơ DC:

Ta chọn động cơ DC có tích hợp encoder để có thể dễ dàng xác định vị trí, tốc độ của động cơ: DC Servo GM25-370.

*Figure 3. Động cơ DC Servo GM25-370.*

Các thông số kĩ thuật của động cơ:

- Điện áp định mức: 12VDC
- Encoder: Cảm biến từ trường Hall, có 2 kênh A, B. độ phân giải: 11 xung/1 kênh/ 1 vòng.
- Tỉ số truyền 34:1 (động cơ quay 34 vòng trục chính hộp giảm tốc quay 1 vòng).
- Độ phân giải thực: 11 x 34 = 374 xung/kênh/vòng.
- Điện áp cấp cho Encoder: 3.3~5VDC
- Dòng không tải: 150mA
- Dòng chịu đựng tối đa khi có tải: 750mA
- Tốc độ không tải: 250 RPM (250 vòng 1 phút)
- Tốc độ chịu đựng tối đa khi có tải: 140 RPM (140 vòng 1 phút)
- Lực kéo Moment định mức: 4.3 Kg.cm
- Lực léo Moment tối đa: 5.2 Kg.cm

## CHƯƠNG 3. PHÂN TÍCH GIẢI THUẬT:

Đây là sơ đồ giải thuật điều khiển PID động cơ:

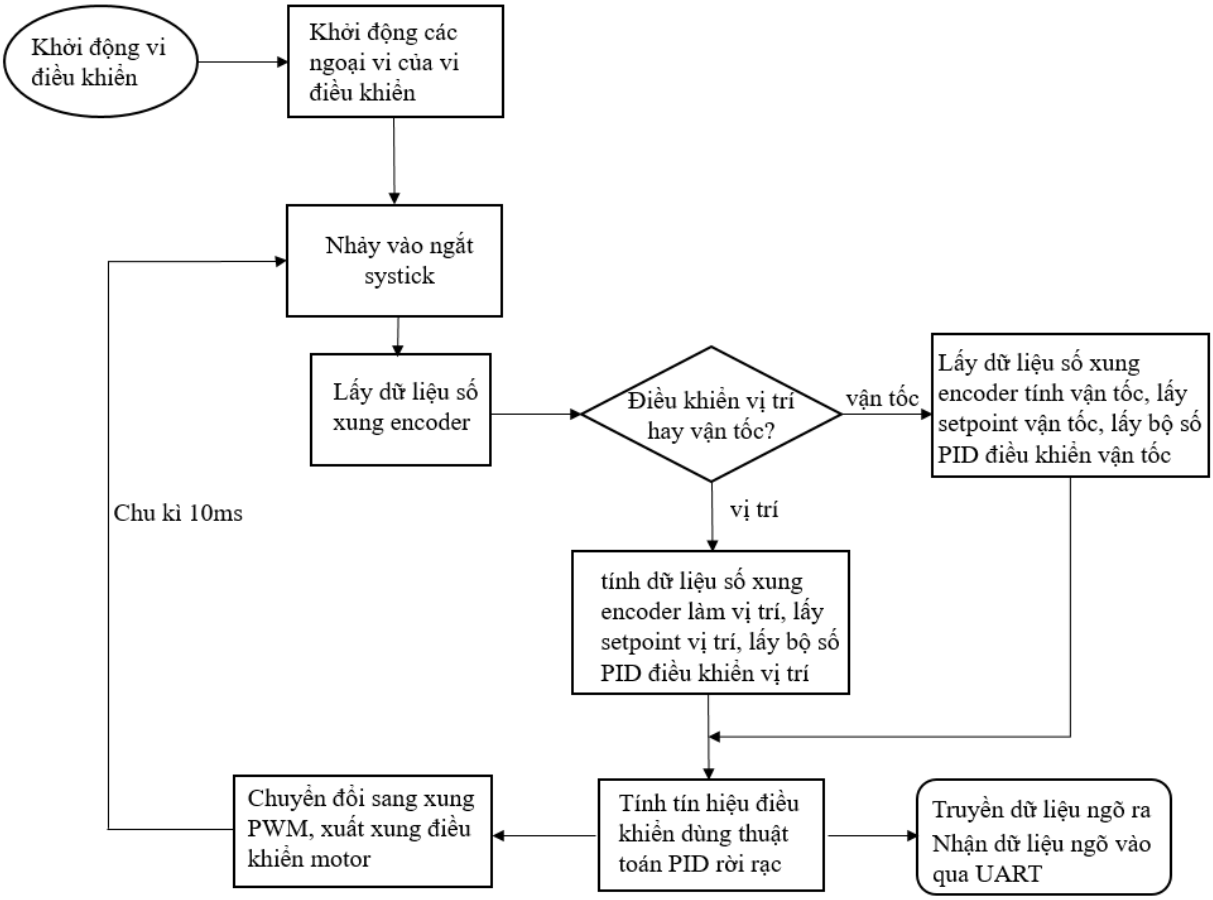


*Figure 4. Sơ đồ giải thuật điều khiển động cơ DC*

Trong đó Firmware có nhiệm vụ tính toán giải thuật, xuất xung PWM điều khiển động cơ, và truyền UART các thông tin vị trí / tốc độ lên Software (PC) để hiển thị

Software (PC) có nhiệm vụ hiển thị các thông tin vị trí / tốc độ và điều chỉnh các mode (vị trí, vận tốc) điều khiển động cơ, đồng thời thiết lập các điểm đặt bất kì cho động cơ.

## 3.1 Lý thuyết:

### *3.1.1 Mô hình toán học điều khiển vận tốc của động cơ DC:*

Ta xây dựng mô hình toán học điều khiển vận tốc của động cơ DC với ngõ vào là điện áp cấp vào V (Volt), ngõ ra là vận tốc của độc cơ DC θ̇ (RPM):

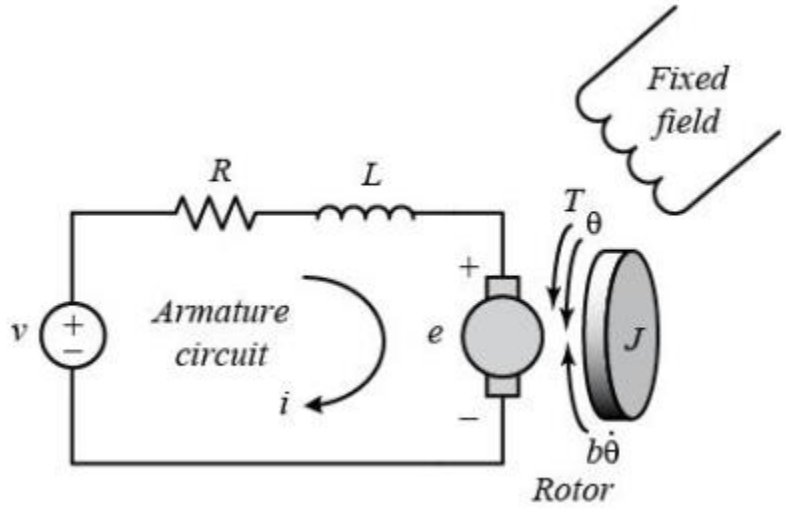


*Figure 5. Mô hình vật lý của động cơ DC*

Theo định lý Kirchoff, ta có:

$$V = Ri + L\frac{di}{dt} + e \qquad (2)$$

Điện áp **e** tỉ lệ thuận với vận tốc góc của động cơ với hằng số không đổi:

$$e = K_e\dot{\theta}$$

Giả định rằng từ trường là không đổi, ta có momen điện từ tỉ lệ thuận với dòng điện phần ứng của động cơ qua hệ số Kd:

$$T_d = K_t\phi i$$

Phương trình của động cơ:

$$T_d = J\ddot{\theta} + B\dot{\theta}$$

Suy ra:

$$K_t\phi i = J\ddot{\theta} + B\dot{\theta} \qquad (1)$$

Từ phương trình (1)(2) ta có hệ phương trình vi phân mô tả hệ thống:

$$\begin{cases} Kt\phi i = J\ddot{\theta} + B\dot{\theta} \\ V = Ri + L\dfrac{di}{dt} + e = Ri + L\dfrac{di}{dt} + Ke\dot{\theta} \end{cases}$$

Hay:

$$\begin{cases} i = \dfrac{J\ddot{\theta} + B\dot{\theta}}{Kt\phi} \\ V = R\dfrac{J\ddot{\theta} + B\dot{\theta}}{Kt\phi} + L\dfrac{B\ddot{\theta} + J\dddot{\theta}}{Kt\phi} + Ke\dot{\theta} \quad (*) \end{cases}$$

Áp dụng biến đổi Laplace: phương trình (*) được viết lại theo biến s

$$V = R\frac{J\dot{\theta}\, s + B\dot{\theta} + TL}{Kt\phi} + L\frac{B\dot{\theta}\, s + J\dot{\theta}\, s^2}{Kt\phi} + Ke\dot{\theta} \quad (*)$$

$$\Leftrightarrow \frac{\dot{\theta}(s)}{V(s)} = \frac{Kt\phi}{(Js+B)(Ls+R)+KeKt\phi}\left[\frac{rad/sec}{V}\right]$$

Từ hàm truyền liên tục, dựa vào lý thuyết rời rạc hóa, ta thu được mô hình rời rạc vận tốc động cơ như sau:

$$\frac{\dot{\theta}(z)}{V(z)} = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

*3.1.2 Mô hình toán học vị trí động cơ:*

Bằng cách thêm 1 khâu tích phân vào hàm truyền vận tốc động cơ, ta được hàm truyền vị trí của động cơ:

$$\frac{\theta(s)}{V(s)} = \frac{Kt\phi}{s[(Js+B)(Ls+R)+KeKt\phi]} \left[\frac{rad}{V}\right]$$

Từ hàm truyền liên tục, dựa vào lý thuyết rời rạc hóa, ta thu được mô hình rời rạc vị trí động cơ như sau:

$$\frac{\theta(z)}{V(z)} = \frac{b_1z^2+b_2z+b_3}{a_1z^3+a_2z^2+a_2z+a_3} \left[\frac{rad}{V}\right]$$

*3.1.3 Bộ điều khiển PID:*

Bộ điều khiển PID là sự kết hợp giữa 3 bộ điều khiển: P (tỉ lệ), I (tích phân), D (vi phân). Đây là bộ điều khiển có khả năng triệt tiêu sai số xác lập, tăng tốc độ đáp ứng (giảm thời gian xác lập), giảm độ vọt lố, nếu tìm được bộ thông số thích hợp.

Phương trình hàm truyền liên tục của bộ điều khiển PID:

$$U(s) = (K_p + K_i/s + K_ds)\, E(s)$$

Trong đó, U(s) là tín hiệu điều khiển đối tượng

E(s) là giá trị sai số giữa tín hiệu mong muốn và tín hiệu ngõ ra của đối tượng:

$$E = Y_d - Y$$

Bộ điều khiển PID bao gồm 3 thông số riêng biệt: Kp, Ki, Kd, lần lượt đại diện cho 3 khâu là tỉ lệ, tích phân, đạo hàm.

Tác động của 3 hệ số trên:

- Kp càng lớn thì tốc độ đáp ứng càng nhanh, sai số xác lập càng nhỏ (nhưng không thể triệt tiêu). Kp càng lớn thì các cực của hệ thống có xu hướng di chuyển ra xa trục thực => Hệ thống càng dao động và độ vọt lố càng cao. Nếu Kp tăng quá giá trị giới hạn thì hệ thống sẽ dao động không tắt dần => mất ổn định.

- Ki càng lớn thì đáp ứng quá độ càng chậm, sai số xác lập càng nhỏ. đặc biệt hệ số khuếch đại của khâu tích phân bằng vô cùng khi tần số bằng 0 => triệt tiêu sai số xác lập với hàm nấc. KI càng lớn thì độ vọt lố càng cao.
- KD càng lớn thì đáp ứng quá độ càng nhanh, độ vọt lố càng nhỏ. Hệ số khuếch đại tại tần số cao là vô cùng lớn nên khâu hiệu chỉnh D rất nhạy với nhiễu tần số cao.

Tìm các hệ số Kp, Ki, Kd:

a. *Phương pháp Ziegler – Nichols:*

**Đối với hệ chỉ bao gồm các khâu quán tính bậc 1:**

Tác động hệ bằng tín hiệu hàm nấc, ta được đáp ứng thời gian của hệ như sau:

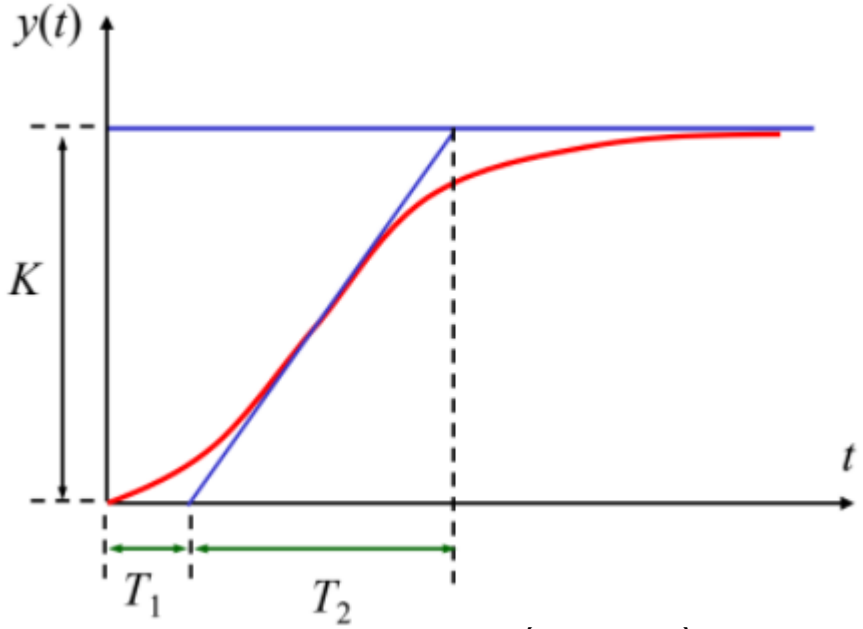


*Figure 6. Đáp ứng thời gian với tín hiệu hàm nấc của hệ gồm khâu quán tính bậc 1*

Vẽ đường tiếp tuyến với đường cong lên của đồ thị đáp ứng. Qua đó ta xác định được các thông số T1, T2.

Sau đó ta tìm các thông số Ki, Kp, Kd theo bảng sau:

| Bộ điều khiển | Kp | Ti | Td |
|---|---|---|---|
| P | $\frac{T_2}{T_1 K}$ | $\infty$ | 0 |
| PI | $0.9\frac{T_2}{T_1 K}$ | $\frac{T_1}{0.3}$ | 0 |
| PID | $1.2\frac{T_2}{T_1 K}$ | $2T_1$ | $0.5T_2$ |

**Đối với những hệ khác:**

Xác định thông số dựa vào đáp ứng hàm nấc của hệ kín:

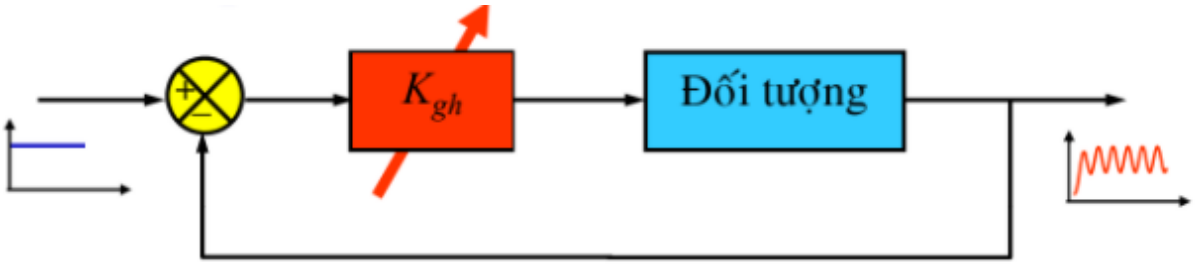


*Figure 7. Sơ đồ khối hệ kín*

Tăng dần $K_{gh}$ cho đến khi đáp ứng của hệ kín là dao động điều hòa:

Xác định $T_{gh}$ là chu kì của dao động điều hòa.

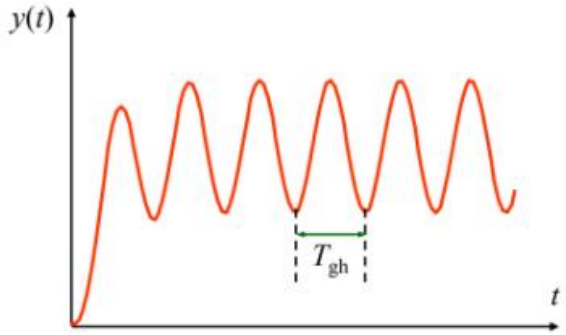


*Figure 8. Đáp ứng của hệ kín với tín hiệu vào hàm nấc, tăng K đến giới hạn ổn định*

Từ đó ta xác định thông số Kp, Ki, Kd theo công thức sau:

| Bộ điều khiển | Kp | Ti | Td |
|---|---|---|---|
| P | $0.5K_{gh}$ | | 0 |
| PI | $0.45K_{gh}$ | $0.83T_{gh}$ | 0 |
| PID | $0.6K_{gh}$ | $0.5T_{gh}$ | $0.125T_{gh}$ |

*b. Phương pháp tìm thủ công bằng tay thông qua thực nghiệm:*

Ta bắt đầu tìm các thông số bằng những bước sau:

Một hệ thống điều khiển PID với các thông số: Kp =1, Ki = 0, Kd=0

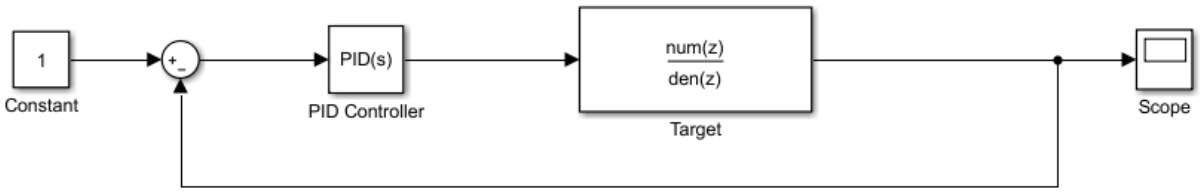


*Figure 9. Sơ đồ khối hệ kín*

- Bước 1: Tăng dần (hoặc giảm dần) hệ số Kp, để tìm giá trị Kp mà tại đó đáp ứng hệ kín dao động tuần hoàn.
- Bước 2: Tăng dần hệ số Kd đến khi cho hệ thống có đáp ứng quá độ tốt nhất.
- Bước 3: Tăng dần hệ số Ki đến khi sai số xác lập của hệ thống được triệt tiêu.

Sau cùng ta được bộ số Kp, Ki, Kd hoàn chỉnh.

Đây là bảng so sánh tác động của 3 hệ số đối với đáp ứng của hệ thống:

| Thông số | Thời gian lên | Vọt lố | Thời gian xác lập | Sai số xác lập | Tính ổn định |
|---|---|---|---|---|---|
| Kp | Giảm | Tăng | Thay đổi nhó | Giảm | Giảm |
| Ki | Giảm | Tăng | Tăng | Loại bỏ | Giảm |

| | | | | | |
|---|---|---|---|---|---|
| Kd | Thay đổi nhó | Giảm | Giảm | Không ảnh hướng | Tăng nếu Kd nhỏ |

## 3.2 Thuật toán điều khiển PID trên vi điều khiển

### *3.2.1 Hàm truyền thuật toán điều khiển PID rời rạc:*

Từ hàm truyền bộ điều khiển PID liên tục, dựa vào lý thuyết rời rạc hóa, ta thu được mô hình rời rạc bộ điều khiển PID như sau:

$$\frac{U(z)}{E(z)} = K_p + \frac{K_i T}{2}\frac{z+1}{z-1} + \frac{K_d}{T}\frac{z-1}{z}$$

Suy ra, ta được luật điều khiển PID rời rạc như sau:

$$u(k) = u(k-1) + Kp\,(e(k)-e(k-1)) + \frac{K_i T}{2}(e(k) + e(k-1)) + \frac{K_d}{T}(e(k) -2e(k-1)+e(k-2))$$

### *3.2.2 Lập trình điều khiển PID vị trí, vận tốc động cơ trên vi điều khiển STM32f103C8T6:*

*a. Cách đọc encoder của bộ điều khiển:*

Tận dụng hỗ trợ phần cứng của STM32, bộ điều khiển sẽ đọc tín hiệu xung A, B của encoder và lưu trữ số xung đó trong thanh ghi counter của timer đếm tương ứng.

Để đọc vị trí, ta đếm số xung của 2 kênh A, B và tính ra giá trị góc θ (°) theo công thức sau:

$$\theta = \frac{\text{Số xung đếm}}{\text{Độ phân giải}} * 360$$

Tuy nhiên, để chính xác hơn, bộ điều khiển sẽ điều khiển dựa trên số xung luôn. Việc tính toán góc chỉ để thể hiện cho người dùng xem.

Để đọc vận tốc, ta tính số xung tăng dần sau một khoảng thời gian nhất định (ở đây là khoảng thời gian ngắt) theo công thức sau:

$$\text{Vận tốc} = \frac{\text{Số xung hiện tại} - \text{Số xung trước đó}}{\text{Chu kì ngắt}} * \frac{60}{\text{Độ phân giải}}$$

*b. Khởi tạo các ngoại vi tương ứng qua phần mềm STM32CubeMX:*

- Ta thiết lập timer 3 xuất xung PWM để điều khiển tốc độ động cơ qua chân: PA6.
- Thiết lập timer hệ thông (Systick) tạo ngắt để tính toán thuật toán điều khiển chu kì khoảng 10ms.
- 2 chân General Purpose Ouput để điều khiển hướng động cơ: PB12, PB13.
- Dùng Timer2 Mode Encoder, để tận dụng hỗ trợ phần cứng của vi điều khiển đọc tín hiệu encoder trả về của động cơ qua 2 chân PA0, PA1.
- Dùng UART1 để truyền nhận thông tin với máy tính: PA10 tương ứng chân RX, PA9 tương ứng chân TX. Trong đó có cấu hình ngắt nhận UART.

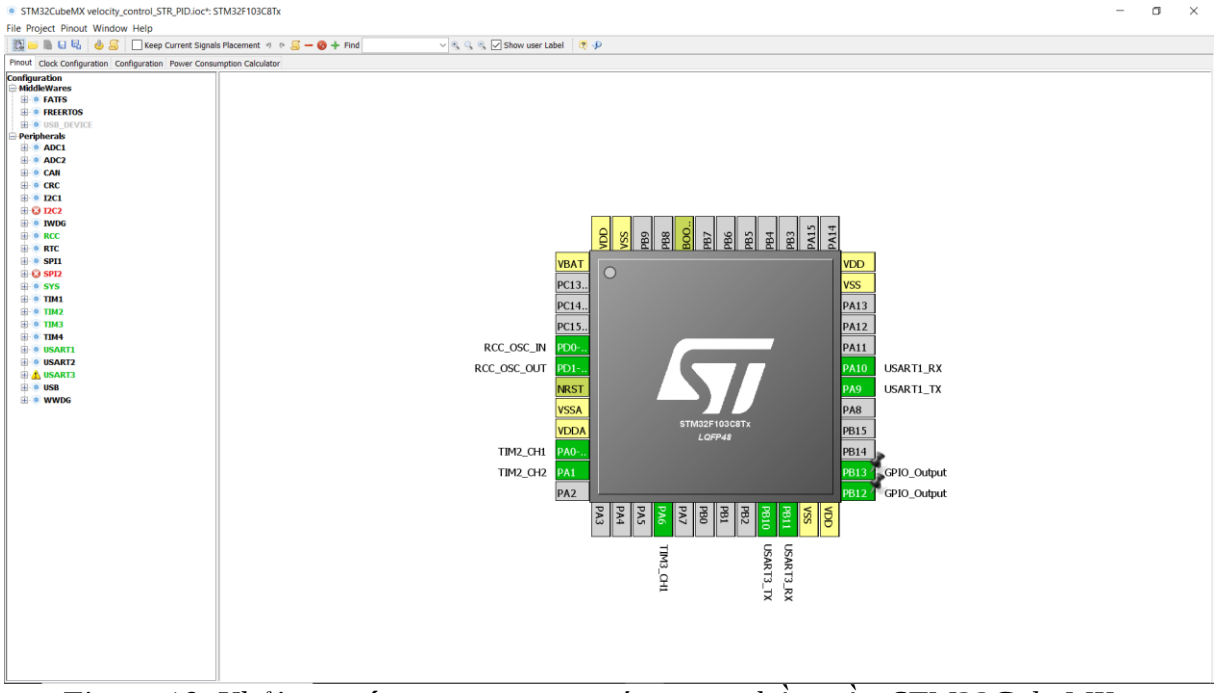


*Figure 10. Khởi tạo các ngoại vi tương ứng qua phần mềm STM32CubeMX*

*c. Thực hiện lập trình trên phần mềm Keil µVision 5:*

Trong hàm main ta khởi động tất cả các thông số:

```
176  int main(void)
177 {
178   /* USER CODE BEGIN 1 */
179
180   /* USER CODE END 1 */
181
182   /* MCU Configuration--------------------------------------------------------*/
183
184   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
185   HAL_Init();
186
187   /* Configure the system clock */
188   SystemClock_Config();
189
190   /* Initialize all configured peripherals */
191   MX_GPIO_Init();
192   MX_TIM2_Init();
193   MX_TIM3_Init();
194   MX_USART1_UART_Init();
195   HAL_UART_Transmit(&huart1,(uint8_t *)"0 0 \r\n",13,1000);
196   HAL_UART_Receive_IT(&huart1,&u8_RxData,1);
197   HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);
198   HAL_TIM_Encoder_Start(&htim2,TIM_CHANNEL_1|TIM_CHANNEL_2);
199   HAL_TIM_Base_Start_IT(&htim2);
200   HAL_GPIO_WritePin(GPIOB,GPIO_PIN_12,GPIO_PIN_RESET);
201   HAL_GPIO_WritePin(GPIOB,GPIO_PIN_13,GPIO_PIN_SET);
202   //__HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,2999);
203   Velocity_Position = Mode_Position;
204   CounterForTransmit = 0;
205   Wait = 0;
206   /* USER CODE BEGIN 2 */
207
208   /* USER CODE END 2 */
209
210   /* Infinite loop */
211   /* USER CODE BEGIN WHILE */
```

*Figure 11. Code hàm main*

Trong hàm ngắt system, ta thực hiện giải thuật điều khiển PID:

```
538  void HAL_SYSTICK_Callback(void)
539 {
540   /* NOTE : This function Should not be modified, when the callback is needed,
541             the HAL_SYSTICK_Callback could be implemented in the user file
542    */
543   Wait++;
544   if (Wait >= 100)
545   {
546   if (Velocity_Position == Mode_Position)
547     {PID_Position();
548      //printf ("%f %f \n",Position,Set_Position);
549   }
550   else
551     {
552       PID_Velocity();
553       //printf ("%f %f \n",motor_speed1,Set_Frequency);
554     }
555
556   __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,out_int);
557   Send2PC();
558   }
559 }
```

*Figure 12. Code trong hàm Callback phục vụ ngắt system*

Các giải thuật điều khiển tốc độ, vị trí được tính trong hàm PID_Position, PID_Velocity:

```
void PID_Position(void)
{
  Position = __HAL_TIM_GET_COUNTER(&htim2) + 65536*count_temp1; //lay gia t
  /*PID Calculation*/
  Error_value = Set_Position - Position; //gia tri xac din ref - gia tri er
  P_part = (Error_value - pre_Error_value) * pKp; //cac he so trong PID //K
  I_part = (Error_value + pre_Error_value) * 0.5 * pKi; //Ki*e/2
  D_part = pKd* (Error_value - (2*pre_Error_value) + pre_pre_Error_value);
  out+=  P_part + I_part + D_part;//(pE*Error_value) + (ppE*pre_Error_valu
  //mot cach tinh khac double const pE = pKp + 0.5*pKi + pKd, ppE = -pKp +
  if (out<0) //neu ngo ra u < 0 thi doi chieu dong co, doi thanh |u|> 0
  {
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,GPIO_PIN_SET);
    if (out < -PWM_MAX) {out = -PWM_MAX;} //gioi han u de thanh do phan gia
    out_temp = out;
    out = -out;
  }
  else
  {
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13,GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,GPIO_PIN_RESET);
    if (out > PWM_MAX) {out = PWM_MAX;}
    out_temp = out;
  }
  out_int = (unsigned)(int)(rint(out)); //lam tron u va chuyen kieu thanh s
  // o day ta luu cac bien
  out = out_temp;
  pre_pre_Error_value = pre_Error_value;
  pre_Error_value = Error_value;
  pre_out = out;
}
```

```
void PID_Velocity(void)
{
  encoder_pulse1 = __HAL_TIM_GET_COUNTER(&htim2);
//day la gi? co the xem trong cac ngat doc
  y = (encoder_pulse1+ 65536*count_temp1);
  motor_speed1 = (y - y_update)*6000/1496;
//xu ly ngo vao la toc do
  Error_value = Set_Frequency - motor_speed1;

//  P_part = vKp * Error_value;
//  I_part += vKi * Error_value;
//  D_part = vKd * (Error_value - pre_Error_value);
//  out+= P_part + I_part + D_part;
  P_part = (Error_value - pre_Error_value) * vKp;
  I_part = (Error_value + pre_Error_value) * 0.5 * vKi*0.01;
  D_part = vKd/0.01* (Error_value - (2*pre_Error_value) + pre_pre_Error_value);
  //u += (vE*Error_value) + (vpE*pre_Error_value) + (vppE*pre_pre_Error_value);
  out+= P_part + I_part + D_part;
  u = out*4000;
// thuat toan PID
  if (u<0) {
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12,GPIO_PIN_SET);
    if (u < -PWM_MAX) {u = -PWM_MAX;}
    out_temp = u;
    u = -u;
  }
  else{
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_12,GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13,GPIO_PIN_SET);
    if (u > PWM_MAX) {u = PWM_MAX;}
    out_temp = u;
  }
  out_int = (unsigned)(int)(rint(u));
  pre_pre_Error_value = pre_Error_value;
  pre_Error_value = Error_value;
  y_update = y;
}
```

*Figure 13. Code các thuật toán PID điều khiển vận tốc, vị trí*

Các thông số sẽ được thiết lập thông qua ngắt nhận UART theo thiết lập sau:

- Nếu nhận kí tự ‘#’ thì bộ điều khiển sẽ đổi tuần tự các mode điều khiển vị trí, điều khiển động cơ.
- Nếu nhận một chuỗi số cùng kí tự ‘*’ thì bộ điều khiển sẽ thay đổi set point tương ứng với mode điều khiển hiện tại.

```
445  void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
446  {
447    /* Prevent unused argument(s) compilation warning */
448    UNUSED(huart);
449    if(huart->Instance == USART1)
450    {
451        HAL_UART_Receive_IT(&huart1,&u8_RxData,1);
452
453      if((u8_RxData != '*') && (u8_RxData  != '#') && (_rxIndex < MAX_STRLEN) )
454      {
455        u8_Rxbuff [_rxIndex] = u8_RxData;
456       _rxIndex++;
457      }
458      else
459      {
460       if(u8_RxData == '*')
461       {
462         if (Velocity_Position == Mode_Position)
463          {
464            Set_Position = atoi(u8_Rxbuff); //Set_Position is int_type
465
466          }
467          else
468          {
469            Set_Frequency = atof(u8_Rxbuff); //Set_Frequency is float_type
470          }
471          _rxIndex = 0; // reset index
472          memset(u8_Rxbuff, '\0', sizeof u8_Rxbuff);
473       }
474      }
475      if(u8_RxData == '#')
476       {
477         if(Velocity_Position == Mode_Position)
478           {
479             __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1,0);
480             HAL_GPIO_WritePin(GPIOB,GPIO_PIN_12,GPIO_PIN_RESET);
481             HAL_GPIO_WritePin(GPIOB,GPIO_PIN_13,GPIO_PIN_SET);
482             Velocity_Position = Mode_Velocity; //switch mode
483             //HAL_TIM_Encoder_DeInit(&htim2);//Restart &htim2
```

*Figure 14. Code truyền nhận dữ liệu qua giao thức UART*

### *3.2.3 Giao tiếp với người dùng bằng PC thông qua Visual Studio:*

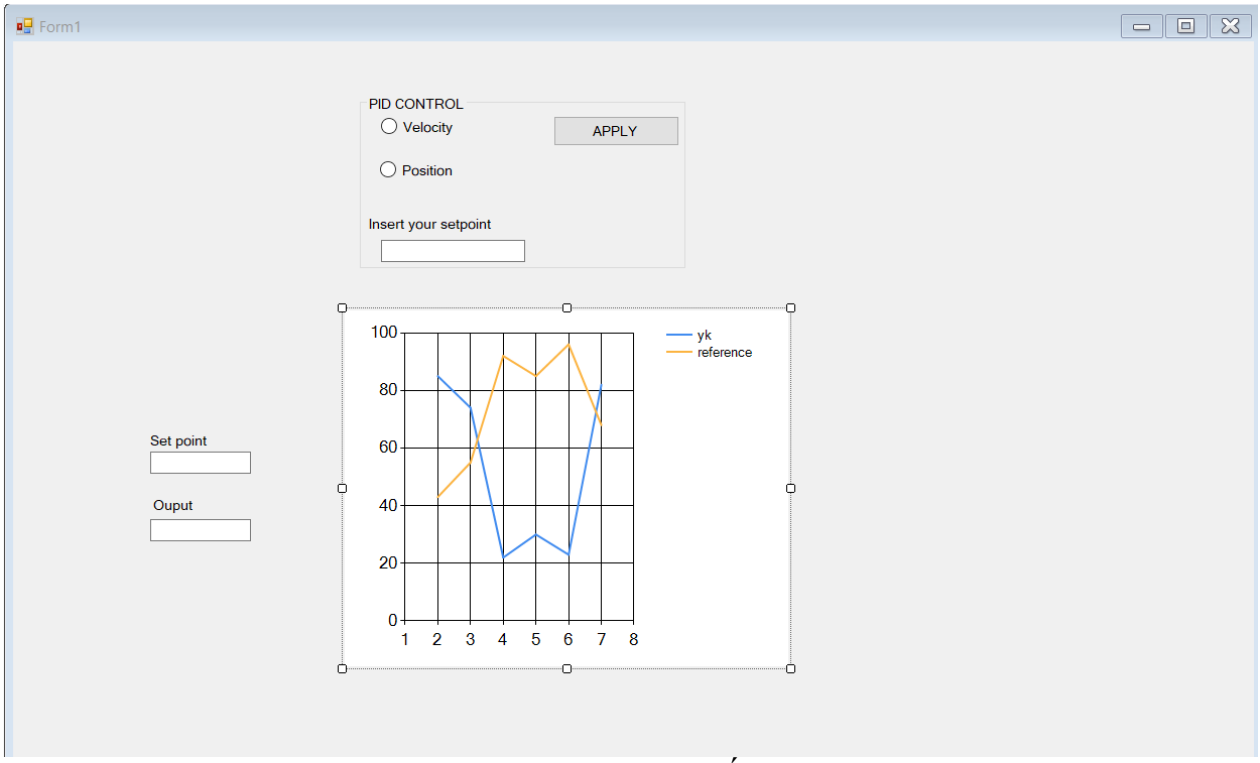


*Figure 15. Giao diện giao tiếp người dùng:*

# CHƯƠNG 4. KẾT QUẢ THỰC NGHIỆM

## 4.1 Điều khiển vận tốc

Chúng ta làm một thí nghiệm đơn giản để so sánh hiệu suất của bộ điều khiển PID đối với 2 phương pháp tìm hệ số, trong trường hợp không tải. Vì bộ điều khiển PID là bộ điều khiển đơn giản không có độ thích nghi tốt nên việc thay đổi tải sẽ làm cho đáp ứng bộ điều khiển PID bị xấu, không thể so sánh 2 bộ điều khiển.

### *4.1.1 Đánh giá chất lượng đáp ứng hệ thống của bộ điều khiển PID với hệ số tìm bằng tay.*

*a. Đánh giá chất lượng đáp ứng*

Đặt setpoint là 100 (RPM), bằng phương pháp tìm hệ số bằng tay, cùng với kinh nghiệm và may mắn, ta tìm được hệ số PID như sau:

$$Kp = 0.004, Kd = 0, Ki = 0.03381$$

Thực hiện khảo sát đáp ứng hệ thống:


*Figure 16. Đáp ứng vận tốc của động cơ với bộ điều khiển PID có hệ số tìm bằng tay.*

**Nhận xét**: Với hệ số tìm được, bộ điều khiển PID cho kết quả đáp ứng rất tốt với ngõ vào thay đổi liên tục, không xảy ra vọt lố, thời gian xác lập nhanh, sai số xác lập được triệt tiêu.

*b. Đánh giá chất lượng tín hiệu điều khiển.*

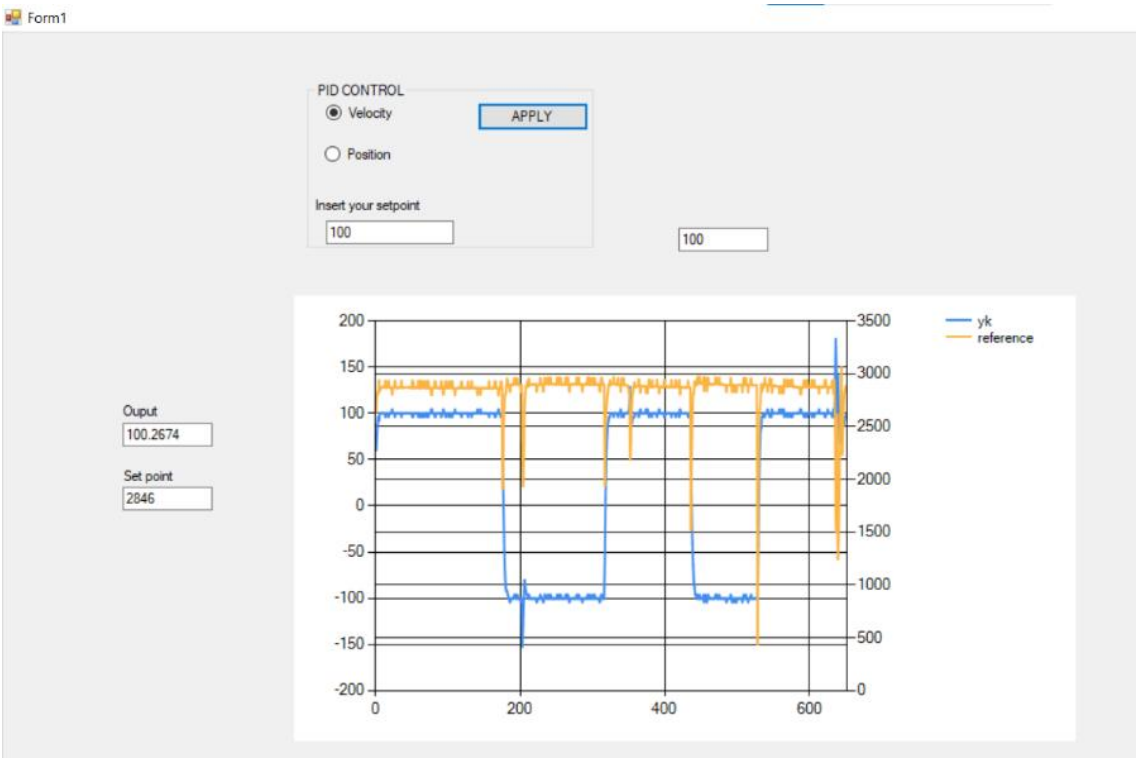


*Figure 17. Tín hiệu điều khiển, điều khiển vận tốc của động cơ*

**Nhận xét**: Tín hiệu điều khiển của bộ điều khiển PID có đáp ứng nhanh, thay đổi tức thì khi thay đổi điểm đặt, giúp cho tín hiệu ngõ ra đáp ứng tốt.

### *4.1.2 Đánh giá chất lượng đáp ứng của bộ điều khiển PID với hệ số Zigler Nicholes*

Áp dụng phương pháp Ziegler Nicholes, ta tìm được các hệ số:

- Kp = 0.0119
- Ti = 5 => Ki = 0.0024
- Td = 2.75 => Kd = 0.0327

Thực hiện khảo sát đáp ứng hệ thống:

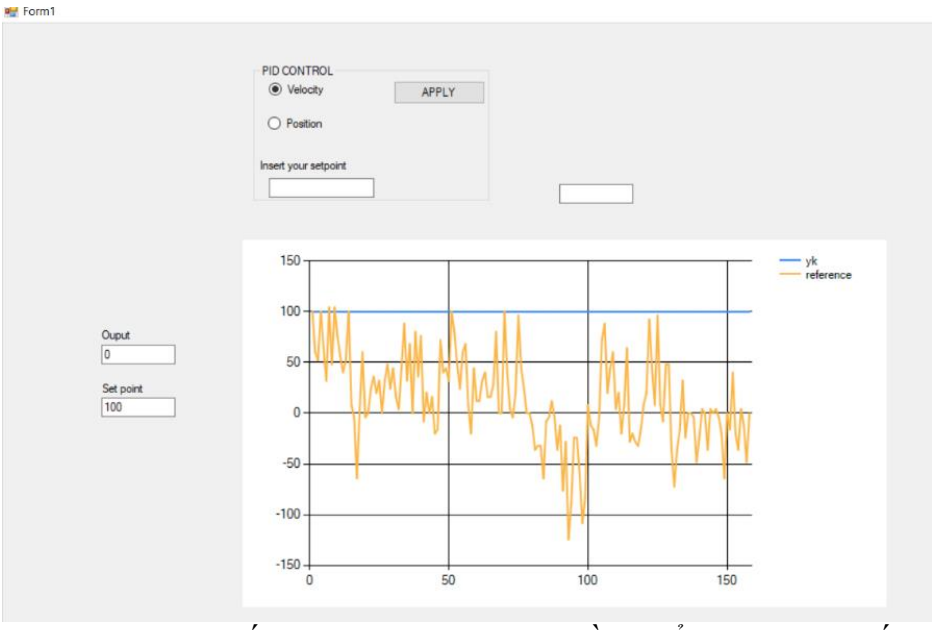


*Figure 18. Đáp ứng vận tốc của động cơ với bộ điều khiển PID có hệ số tìm theo phương pháp Ziegler Nicholes*

**Nhận xét**: Với hệ số trên hệ thống bị mất ổn định, dao động vô hạn. Lý do là vì phương pháp Ziegler Nicholes chỉ cho phép tìm hệ số một cách tương đối, xảy ra sai số lớn, tùy thuộc vào cách người điều khiển vẽ đường tiếp tuyến. Muốn tìm được hệ số tốt, cần có nhiều thời gian để tìm. Khi đó, phương pháp Ziegler Nicholes không có hiệu quả so với phương pháp tìm bằng tay.

## 4.2 Điều khiển vị trí

### *4.2.1 Đánh giá chất lượng đáp ứng*

Vì mô hình toán học của điều khiển vị trí của động cơ là mô hình bậc 3 nên thực hiện mô hình vòng kín sẽ không xảy ra hiện tượng dao động điều hòa. Chính vì vậy, trong bài toán điều khiển vị trí, ta chỉ khảo sát đối với bộ điều khiển PID có hệ số tìm theo phương pháp bằng tay.

Theo phương pháp băng tay, ta tìm được bộ số PID sau:

$$Kp = 8, Kd = 0.00, Ki = 0.05;$$

Thực hiện khảo sát đáp ứng hệ thống:


*Figure 19. Đáp ứng vị trí của động cơ*

**Kết luận**: Như đã phân tích về thực tế động cơ, khi điểm đặt được thay đổi với biên độ lớn, bộ điều khiển cho kết quả đáp ứng tốt, không có vọt lố, thời gian lên nhanh. Tuy nhiên khi điểm đặt thay đổi với biên độ nhỏ, dẫn đến bộ điều khiển tính toán tín hiệu ngõ vào nhỏ, chưa đủ để động cơ quay. Qua một khoảng thời gian, tín hiệu được tích lũy lớn dần mới làm cho hệ thống xác lập về, khiến cho thời gian xác lập lớn.

### *4.2.2 Đánh giá chất lượng tín hiệu điều khiển.*


*Figure 20. Tín hiệu điều khiển, điều khiển vị trí của động cơ*

Khi điểm đặt thay đổi lớn, tín hiệu điều khiển thay đổi lớn, điều khiển động cơ quay đến vị trí của điểm đặt. Khi động cơ quay gần đến điểm đặt, tín hiệu điều khiển nhỏ, không đủ để động cơ quay, dẫn đến hiện tượng tín hiệu điều khiển tích lũy dần (các đường chéo của tín hiệu điều khiển).

Để cải thiện, cần phải điều chỉnh tín hiệu Ki, Kd sao cho hợp lý.

# CHƯƠNG 5. KẾT LUẬN

## 5.1 Tổng kết.

Qua đồ án lần này, ta đã làm rõ được những vấn đề sau:

- Cho sinh viên hiểu rõ thế nào là một hệ thống điều khiển nhúng, cách áp dụng các bộ điều khiển vào các hệ thống thông qua vi điều khiển.
- Khảo sát bộ điều khiển PID truyền thống, xây dựng một bộ điều khiển PID có đáp ứng tốt.
- Quan sát, hiểu rõ hơn về cơ chế hoạt động của bộ điều khiển PID thông qua quan sát tín hiệu điều khiển, nhận thấy sự khác nhau giữa mô phỏng mô hình và mô hình thực tế. Qua đó, tích lũy kinh nghiệm điều chỉnh các hệ số PID sao cho bộ điều khiển có hiệu quả tốt nhất.

## 5.2 Những khuyết điểm cần phải cải thiện.

❖ Cách xác định vị trí, vận tốc qua encoder truyền thống gây ra nhiều bất cập:

- Xác định vận tốc: tăng lượng tính toán phức tạp cho vi điều khiển, làm hiệu suất vi điều khiển giảm, làm tăng nhiễu đầu ra của hệ thống.
- Xác định vị trí: Vì tận dụng phần cứng timer của vi điều khiển STM32 để đếm xung encoder, nên xảy ra các trường hợp làm tín hiệu ngõ vào của bộ điều khiển sai: tràn timer (giới hạn tỉmer là $2^{16}$), timer sẽ đếm về 0; chỉ xác định với các giá trị vị trí (góc) dương vì phần cứng timer chỉ xác định được số không dấu.

Để khắc phục: tận dụng ngắt Compare, ngắt Capture, của timer để xác định thời gian ngắt giữa 2 xung encoder, qua đó xác định được tần số.

Thiết lập ngắt tràn timer, kết hợp một biến counter để tích lũy số xung đếm timer, tránh giá trị đếm được bị reset về 0 khi tràn timer.

❖ Vì hạn chế về phần cứng, kết quả thí nghiệm chưa đầy đủ, chưa thể hiện được khuyết điểm của bộ điều khiển PID, chưa thể hiện được PID là bộ điều khiển mang tính thích nghi thấp, Nếu có thêm thời gian, sẽ thay đổi tải tác động lên động cơ, để thấy được những khuyết điểm của PID, qua đó, hướng đến các bộ điều khiển khác có tính thích nghi tốt hơn.

## PHỤ LỤC. TÀI LIỆU THAM KHẢO


- TS. Huỳnh Thái Hoàng. (2011). *Giáo trình cơ sở điều khiển tự động, chương 6; 9.*
- Nguyễn Tấn Khoa. (2013). *Báo cáo luận văn tốt nghiệp: THIẾT KẾ VÀ THỰC HIỆN BỘ ĐIỀU KHIỂN THÍCH NGHI TỰ CHỈNH CHO ĐỘNG CƠ DC.*