

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





**BÁO CÁO BÀI TẬP LỚN  
THIẾT KẾ HỆ THỐNG NHÚNG**

**TÊN ĐỀ TÀI:**

**XÂY DỰNG MÔ HÌNH PHẦN CỨNG VÀ GIẢI THUẬT  
ĐIỀU KHIỂN CÓ HỒI TIẾP ĐỘNG CƠ, KHẢO SÁT ĐÁP  
ỨNG CỦA HỆ THỐNG BẰNG PHẦN MỀM MÁY TÍNH**

**LỚP L03 – NHÓM 15 – HK241**

**Giảng viên hướng dẫn: Thầy Nguyễn Phan Hải Phú**

| Sinh viên thực hiện | MSSV    | Điểm số | Ký tên  |
|---------------------|---------|---------|---|
| Huỳnh Chí Thành     | 2114774 |         |  |
| Võ Anh Khoa         | 2211659 |         |  |

*TP. Hồ Chí Minh, tháng 11 năm 2024*

## BẢNG PHÂN CÔNG

| Họ và tên       | MSSV    | Lớp | Phân công   | Mức độ hoàn thành |
|-----------------|---------|-----|---|-------------------|
| Huỳnh Chí Thành | 2114774 | L03 | -Lập trình vi điều khiển<br>-Lập trình giao diện<br>-Tổng hợp báo cáo | 100%              |
| Võ Anh Khoa     | 2211659 | L03 | -Hàn mạch<br>-Vẽ PCB Layout<br>-Làm báo cáo                           | 100%              |

# MỤC LỤC

|   |    |
|---|----|
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT .....                               | 1  |
| 1.1.Xây dựng bộ điều khiển PID: .....                         | 1  |
| 1.2.Tìm thông số PID bằng phương pháp Ziegler– Nichols: ..... | 1  |
| CHƯƠNG 2: PHƯƠNG ÁN THỰC HIỆN ĐỀ TÀI.....                     | 3  |
| 2.1. Các phương án có thể chọn: .....                         | 3  |
| 2.2.Giới thiệu linh kiện: .....                               | 4  |
| 2.2.1.Mạch công suất : .....                                  | 4  |
| 2.2.2 Raspberry Pi Pico RP2040: .....                         | 8  |
| 2.2.3. Động cơ JGA25-370-12VDC280RPM: .....                   | 9  |
| 2.2.4.Encoder: .....  | 11 |
| 2.2.5.Nguồn cấp:.....   | 12 |
| 2.2.6. Board mở rộng để kết nối các linh kiện:.....           | 13 |
| CHƯƠNG 3: MẠCH NGUYÊN LÝ.....                                 | 15 |
| 3.1 Sơ đồ khối: .....   | 15 |
| 3.2 Sơ đồ nối dây:.....                                       | 15 |
| CHƯƠNG 4: LƯU ĐỒ GIẢI THUẬT .....                             | 16 |
| 4.1.Ý tưởng: .....  | 16 |
| 4.2.Sơ đồ giải thuật : .....                                  | 17 |
| CHƯƠNG 5: LẬP TRÌNH .....                                     | 18 |
| 5.1.Firmware( Raspberry Pi Pico RP2040):.....                 | 18 |
| 5.1.2.Chương trình code: (Trình bày ở phần Phụ Lục).....      | 18 |
| 5.1.2,Phân tích code:.....                                    | 18 |
| 5.2.Software(Visual Studio 2022): .....                       | 22 |
| 5.2.1.Giao diện: .....  | 22 |
| 5.2.1.Phân tích code:.....                                    | 22 |
| CHƯƠNG 6: KẾT QUẢ THỰC NGHIỆM.....                            | 28 |
| 6.1.Mô hình thực tế: .....                                    | 28 |
| 6.2.Tìm thông số PID: .....                                   | 30 |
| KẾT LUẬN.....   | 35 |
| TÀI LIỆU THAM KHẢO .....                                      | 36 |
| DANH MỤC HÌNH ẢNH .....                                       | 37 |

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Xây dựng bộ điều khiển PID:

Bộ điều khiển PID có đặc tính mềm dẻo phù hợp hầu hết các đối tượng trong công nghiệp.

Phương trình vi phân mô tả quan hệ tín hiệu vào ra của bộ điều khiển:

$$u(t) = K_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(t) dt + T_d \cdot \frac{de(t)}{dt})$$

Trong đó :

- $e(t)$ : sai số giữa giá trị đặt và giá trị thực tế của bộ điều khiển
- $u(t)$ : Là tín hiệu ra của bộ điều khiển.
- $K_p$ : Hệ số khuếch đại
- $T_i$  : Hằng số thời gian tích phân
- $T_d$  : Hằng số thời gian vi phân

Khi chuyển sang miền Laplace:

$$U(s) = K_p \cdot (E(s) + \frac{1}{T_i \cdot s} E(s) + T_d \cdot s \cdot E(s))$$

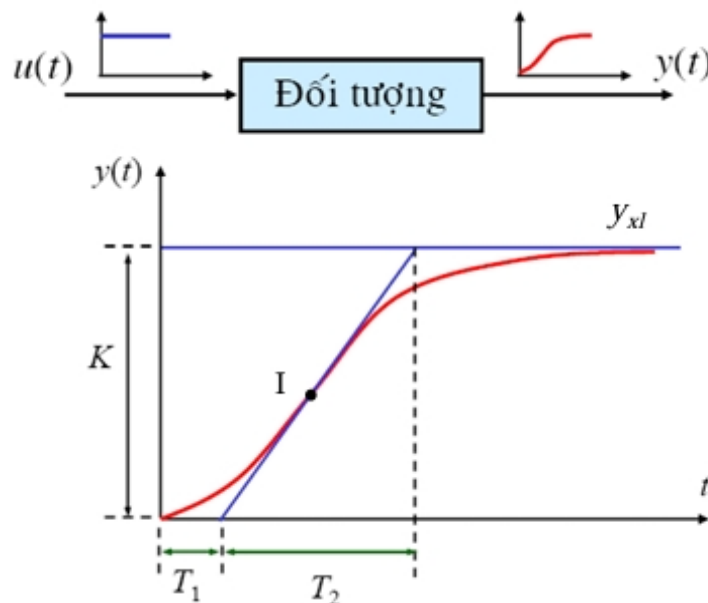
## 1.2. Tìm thông số PID bằng phương pháp Ziegler– Nichols:

Khi cung cấp tín hiệu điều khiển  $u$  cho động cơ thì tốc độ động cơ sẽ tăng lên theo thời gian và đạt giá trị xác lập  $y_{xl}$ .

Tại điểm uốn I của đường cong  $y(t)$ , tiếp tuyến đi qua I sẽ cắt đường  $y(0)$  và đường  $y_{xl}$ . Giá trị  $K$ ,  $T_1$ ,  $T_2$  được xác định như hình và  $K$  được xác định bằng công thức:

$$K = \frac{y_{xl} - y(0)}{u}$$

Trong đó  $y(0)$  là tốc độ ban đầu của lò nhiệt khi không cung cấp tín hiệu điều khiển  $u$ .



Hình 1. Quan hệ ngõ ra theo thời gian của đối tượng

Sau khi có  $K$ ,  $T_1$ ,  $T_2$ , ta tiến hành tính các thông số  $K_p$ ,  $K_i$ ,  $K_d$  của bộ điều khiển PID theo công thức:

| $K_p$               | $T_i$     | $T_d$     |
|---------------------|-----------|-----------|
| $1.2 * T_2 / T_1 K$ | $2 * T_1$ | $T_1 / 2$ |

## CHƯƠNG 2: PHƯƠNG ÁN THỰC HIỆN ĐỀ TÀI

### 2.1. Các phương án có thể chọn:

Phản ứng: Động cơ JGB37-545 là động cơ giảm tốc, khả năng điều khiển PID của động cơ giảm tốc sẽ tốt hơn nhiều so với động cơ bình thường vì động cơ sẽ bám sát giá trị đặt nên sẽ ổn định và đúng hơn. Cùng với số xung đọc được của 1 kênh khi động cơ quay 1 vòng của trục chính là 2688 xung

Động cơ JGA25-370 là động cơ giảm tốc, khả năng điều khiển PID cũng khá tốt. Một vòng quay của trục chính encoder đọc được 234.3 xung. Việc đọc được ít xung như vậy sẽ làm động cơ điều không chính xác bằng động cơ đọc xung nhiều hơn

Driver lái động cơ ở đây có hai đề xuất một là module L298N điều khiển 2 động cơ và Module JGA25-370 điều khiển một động cơ. L298N thì điều khiển động cơ có công suất nhỏ hơn IBT2. Theo thông số tại em tìm hiểu được thì module IBT2 vượt trội hơn so với L298N.

Phương án chọn: Vì kinh phí mua động cơ JGB37-545 lớn nên nhóm tại em sẽ chuyển qua dùng động cơ JGA25-370 để điều khiển động cơ. Do với động cơ này thì chỉ cần đến module L298N để điều khiển động cơ nên tại em cũng sẽ chọn module L298N để điều khiển động cơ

| Các thiết bị linh kiện     | Giá thành(VNĐ) |
|----------------------------|----------------|
| Động cơ giảm tốc JGB37-545 | 396.00         |
| Động cơ JGA25-370          | 215.00         |
| Module L298N               | 35.000         |
| Module BTS 7960            | 76.000         |
| Raspberry Pico RP2040      | 150.00         |
| Nguồn tổ ong 12V 5A        | 169.000        |
| Board đa năng              | 20.000         |
| Dây nối, chì, nhựa thông   | 30.000         |

| Các thiết bị linh kiện     | Giá thành(VNĐ) | Tổng tiền (VNĐ) |
|----------------------------|----------------|-----------------|
| Động cơ giảm tốc JGA25-370 | 215.000        | 609.000         |
| Module L298N               | 35.000         |                 |
| Raspberry Pico RP2040      | 140.000        |                 |
| Nguồn tổ ong 12V 5A        | 169.000        |                 |
| Tấm đồng                   | 20.000         |                 |
| Dây nối, chì, nhựa thông   | 30.000         |                 |

Việc sử dụng phương án trên sẽ giúp tiết kiệm được một số tiền lớn, nhưng vẫn đảm bảo hoàn thành chủ đề bài tập lớn.

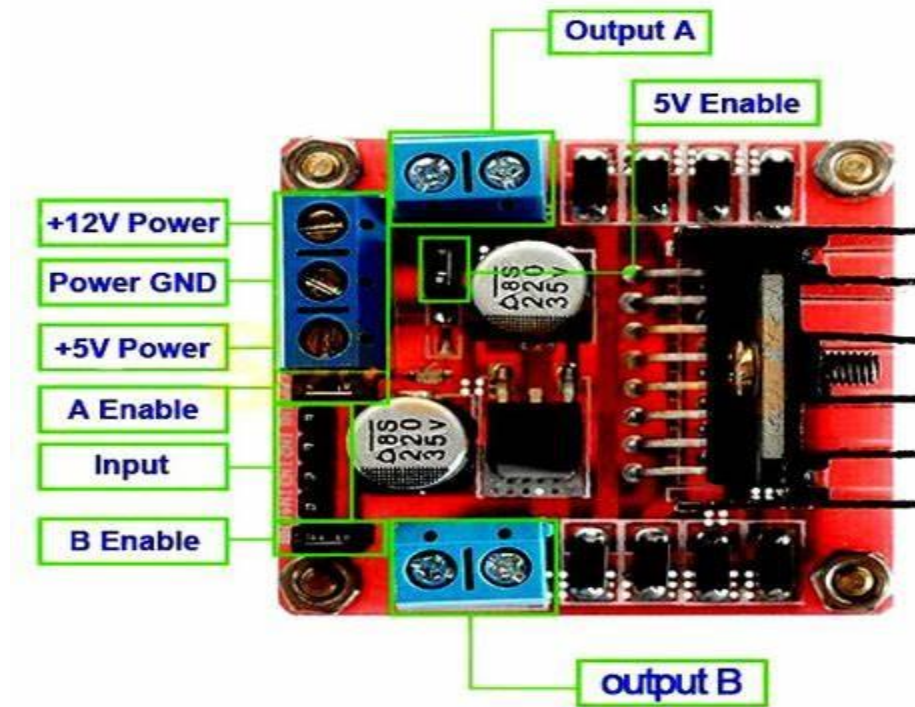
## 2.2.Giới thiệu linh kiện:

### 2.2.1.Mạch công suất :

Ta sử dụng mạch cầu H và dùng IC L298N để tạo một mạch công suất, dựa trên module L298N trên thị trường:

- Mạch cầu H: là một mạch đơn giản dùng để điều khiển động cơ DC quay thuận hoặc quay nghịch. Trong thực tế, có nhiều kiểu mạch cầu H khác nhau tùy vào cách chúng ta lựa chọn linh kiện có dòng điện, áp điều khiển lớn hay nhỏ, tần số xung PWM... Và chúng sẽ quyết định đến khả năng điều khiển của cầu H.
- Module L298N: là một mạch tích hợp nguyên khối với hai mạch cầu H. Nó có thể được sử dụng để đảo chiều quay của động cơ theo cả hai chiều và điều khiển tốc độ của động cơ bằng kỹ thuật điều khiển chế độ rộng xung PWM (Pulse Width Modulation).

Module điều khiển động cơ DC L298N ứng dụng nhiều trong lĩnh vực embedded, đặc biệt là trong robot. Hầu hết các bộ vi điều khiển hoạt động ở điện áp (5V) và dòng điện rất thấp còn các động cơ thì yêu cầu điện áp và dòng điện cao hơn. Vì vậy, các vi điều khiển không thể cấp dòng điện cao hơn cho động cơ nếu không sử dụng các IC điều khiển động cơ DC.



Hình 2. Mạch cầu H L298N thực tế

Thông số kĩ thuật :

- Driver: L298N
- Điện áp điều khiển: +5 V ~ +12 V.
- Dòng tối đa cho mỗi cầu H là: 2A ( $\Rightarrow$  2A cho mỗi motor)
- Điện áp của tín hiệu điều khiển: +5 V ~ +7 V.
- Dòng của tín hiệu điều khiển: 0 ~ 36mA.
- Công suất hao phí: 20W (khi nhiệt độ  $T = 75^{\circ}\text{C}$ ).
- Nhiệt độ bảo quản:  $-25^{\circ}\text{C} \sim +130^{\circ}\text{C}$ .

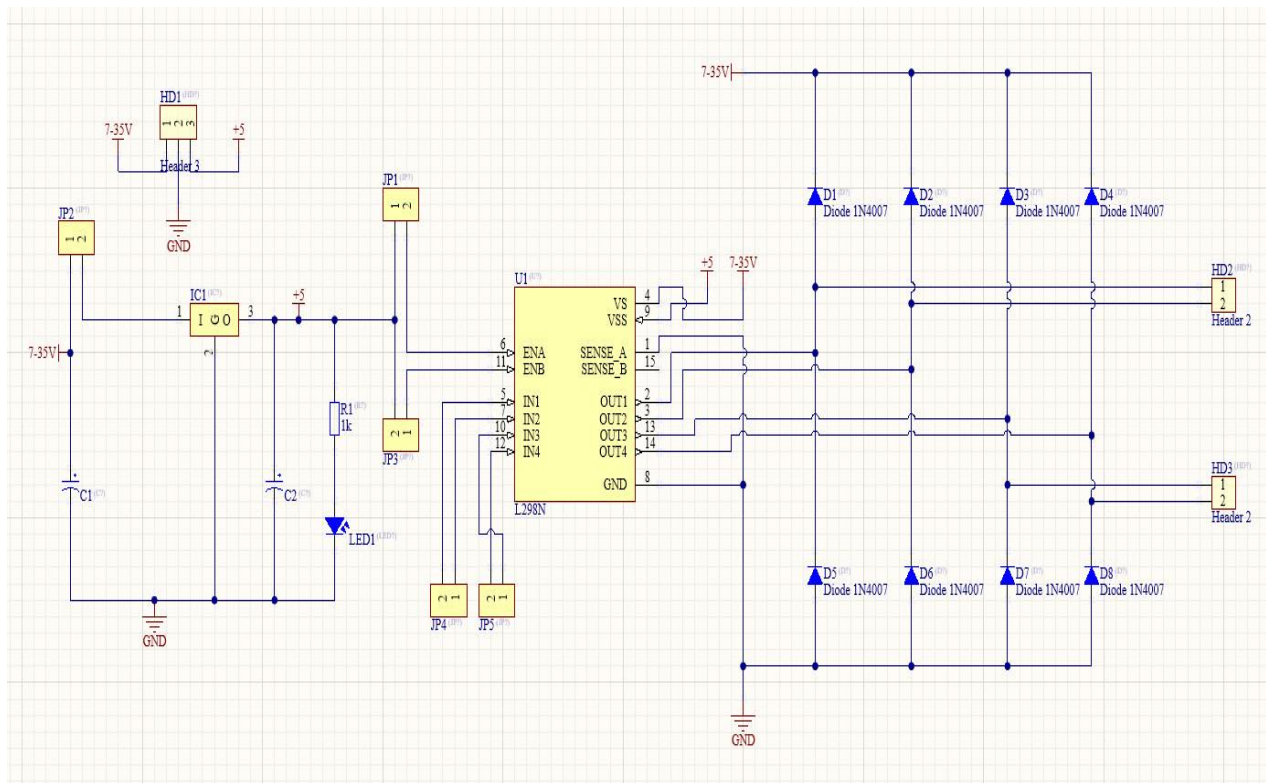
Chức năng các chân L298N:

| Chân      | Chức năng  |
|-----------|--|
| IN1 & IN2 | Các chân đầu vào điều khiển hướng quay động cơ A |
| IN3 & IN4 | Các chân đầu vào điều khiển hướng quay động cơ b |

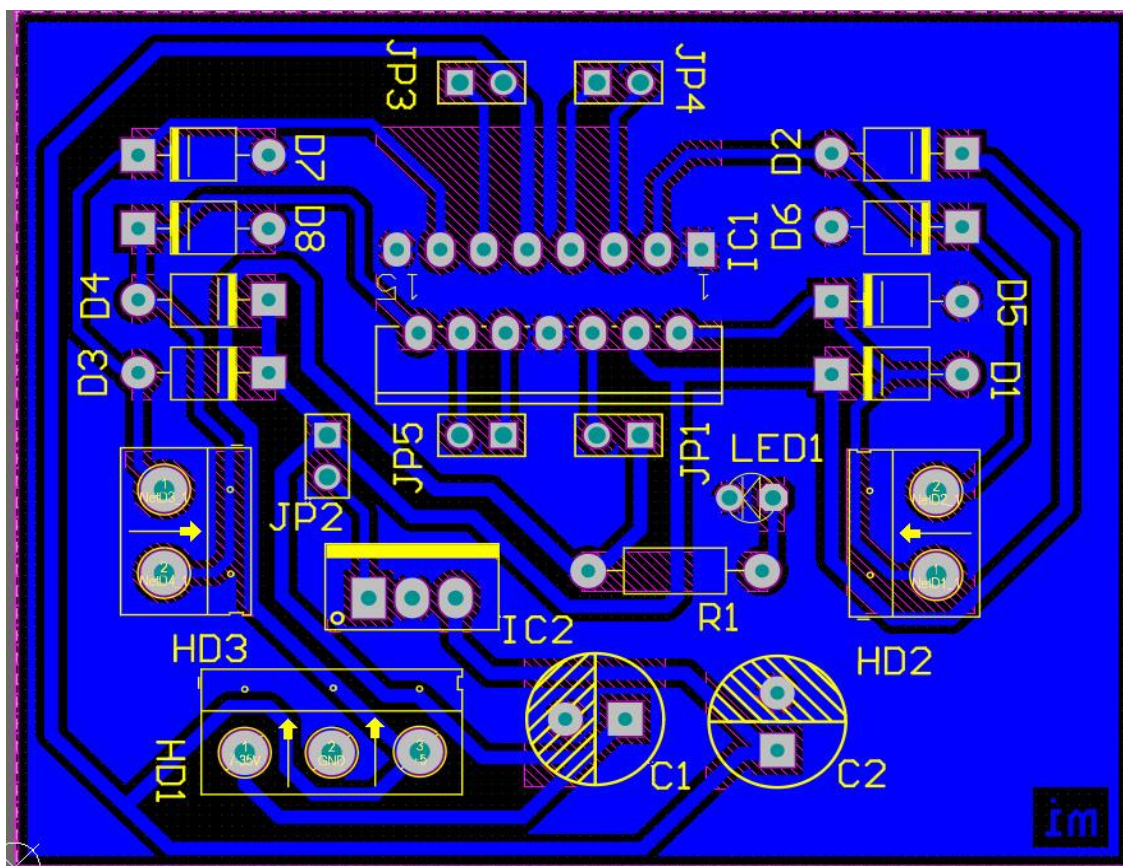


|            |   |
|------------|---|
| ENA        | Kích hoạt tín hiệu PWM cho Động cơ A        |
| ENB        | Kích hoạt tín hiệu PWM cho Động cơ B        |
| OUT1& OUT2 | Chân đầu ra cho động cơ A                   |
| OUT3& OUT4 | Chân đầu ra cho động cơ B                   |
| 12V        | Đầu vào cấp nguồn 12V                       |
| 5V         | Cấp nguồn cho mạch logic bên trong IC L298N |
| GND        | Chân nối đất                                |

Sử dụng Altium, ta thiết kế mạch công suất :



Hình 3. Mạch nguyên lý của mạch công suất dựa trên module L298N



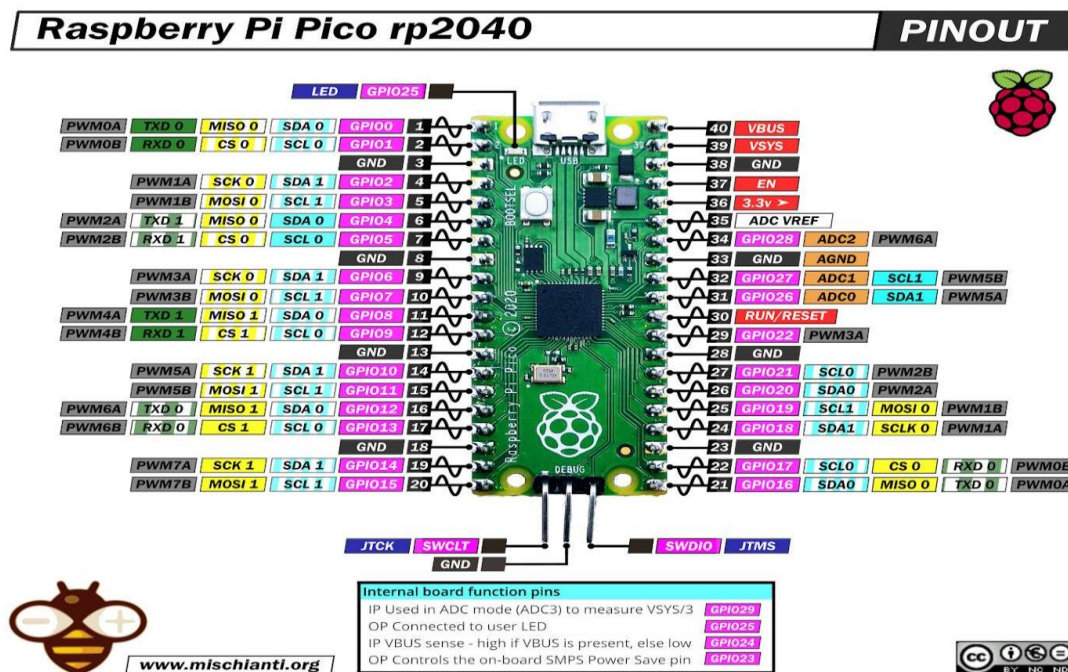
Hình 4. PCB Layout



Hình 5. Mạch thực tế

### 2.2.2 Raspberry Pi Pico RP2040:

[Raspberry Pi Pico](#) (Pi Pico) là board vi điều khiển hiệu năng cao, chi phí thấp được xây dựng dựa trên chip [RP2040](#) – chip vi điều khiển được thiết kế bởi chính Raspberry Pi và được ra mắt vào năm 2020. Đây là bo mạch vi điều khiển đầu tiên của Raspberry Pi, được thiết kế đặc biệt cho các dự án điều khiển, lập trình nhúng. Raspberry Pi Pico có thể được lập trình dễ dàng thông qua ngôn ngữ C/C++ hoặc MicroPython, đây cũng chính là 2 ngôn ngữ lập trình cơ bản hỗ trợ cho loại board vi điều khiển này.



Hình 6. Sơ đồ chân và chức năng các chân của Raspberry pi pico RP2040

Thông số kỹ thuật :

- Điện áp cấp nguồn: 1.8~5.5VDC, cấp nguồn qua cổng USB / VBUS (5VDC) hoặc chân VSYS (1.8~5.5VDC)
- Điện áp giao tiếp GPIO: TTL 1.8~3.3VDC
- Xung nhịp tối đa 133 MHz
- 264KB SRAM
- 26 GPIO pins
- $2 \times$  UART,  $2 \times$  SPI,  $2 \times$  I2C, 16 kênh PWM
- 21 mm  $\times$  51 mm
- Nhiệt độ bảo quản :-20°C to +85°C



### 2.2.3. Động cơ JGA25-370-12VDC280RPM:



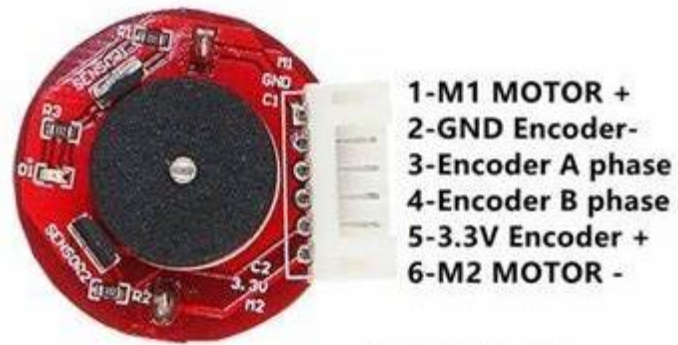
*Hình 7. Động cơ Encoder JGA25-370*

Động Cơ DC Servo JGA25-370 DC Geared Motor được tích hợp thêm Encoder hai kênh AB giúp đọc và điều khiển chính xác vị trí, chiều quay của động cơ trong các ứng dụng cần độ có chính xác cao: điều khiển PID, Robot tự hành,....

Thông số kỹ thuật:

- Điện áp sử dụng: 12VDC
- Đường kính động cơ: 25mm
- Tỉ số truyền 21.3:1 (động cơ quay 21.3 vòng trục chính hộp giảm tốc quay 1 vòng).
- Dòng không tải: 70mA
- Dòng chịu đựng tối đa khi có tải: 1A
- Tốc độ không tải: 280RPM (280 vòng 1 phút)
- Tốc độ chịu đựng tối đa khi có tải: 215RPM (215 vòng 1 phút)
- Lực kéo Moment định mức: 0.4KG.CM
- Lực kéo Moment tối đa: 2KG.CM
- Chiều dài hộp số L: 19mm
- Số xung Encoder mỗi kênh trên 1 vòng quay trục chính:  $11 \times 21.3 = 234.3$  xung.

Sơ đồ chân của động cơ:



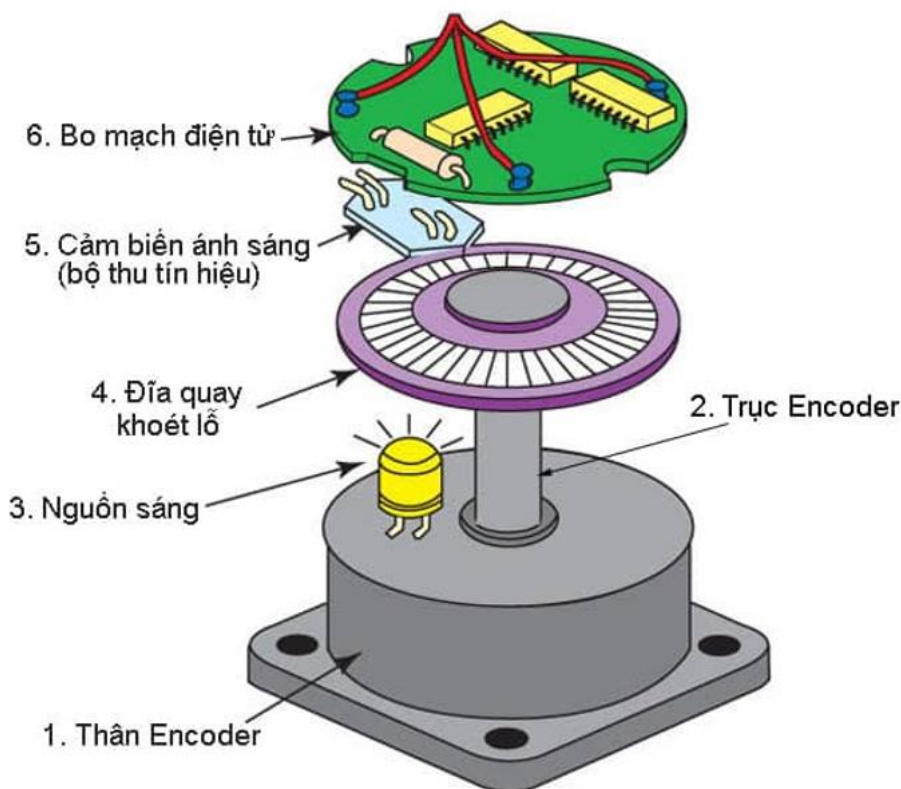
*Hình 8. Tên các chân của động cơ JGA25-370*

1. **M1:** Dây cấp nguồn cho động cơ.
2. **GND:** Dây cấp nguồn cho Encoder, 0VDC.
3. **C1/A:** Kênh trả xung A
4. **C2/B:** Kênh trả xung B
5. **VCC:** Dây cấp nguồn cho Encoder 3.3~5VDC
6. **M2:** Dây cấp nguồn cho động cơ

#### 2.2.4.Encoder:

-Khái niệm: Encoder hay còn gọi là Bộ mã hóa quay hoặc bộ mã hóa trục, là một thiết bị cơ điện chuyển đổi vị trí góc hoặc chuyển động của trục hoặc trục thành tín hiệu đầu ra analog hoặc kỹ thuật số. Encoder được dùng để phát hiện vị trí, hướng di chuyển, tốc độ... của động cơ bằng cách đếm số vòng quay được của trục.

-Cấu tạo :



*Hình 9.Cấu tạo Encoder bên trong động cơ*

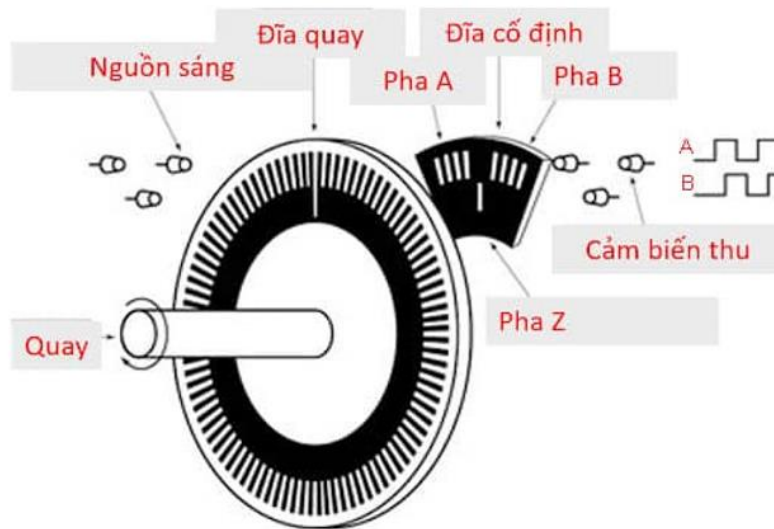
Một bộ mã hóa Encoder gồm có các bộ phận chính:

- **Thân và trục.**
- **Nguồn phát sáng** (lightsource): là 1 đèn LED
- **Đĩa mã hóa** (code disk): có rãnh nhỏ quay quanh trục, khi đĩa này quay và chiếu đèn LED lên trên mặt đĩa thì sẽ có sự ngắt quãng xảy ra. Các rãnh trên đĩa chia vòng tròn 360° thành các góc bằng nhau. Một đĩa có thể có nhiều dãy rãnh tính từ tâm tròn.
- **Bộ cảm biến ánh sáng thu tín hiệu** (photosensor): là một con mắt thu quang điện để nhận tín hiệu từ đĩa quay
- **Bo mạch điện tử** (electronic board): giúp khuếch đại tín hiệu.

-Nguyên lý hoạt động:

+ Khi đĩa quay quanh trục, trên đĩa có các rãnh để tín hiệu quang chiếu qua (Led). Chỗ có rãnh thì ánh sáng xuyên qua được, chỗ không có rãnh ánh sáng không xuyên qua được. Với các tín hiệu có/không người ta ghi nhận đèn Led có chiếu qua hay không.

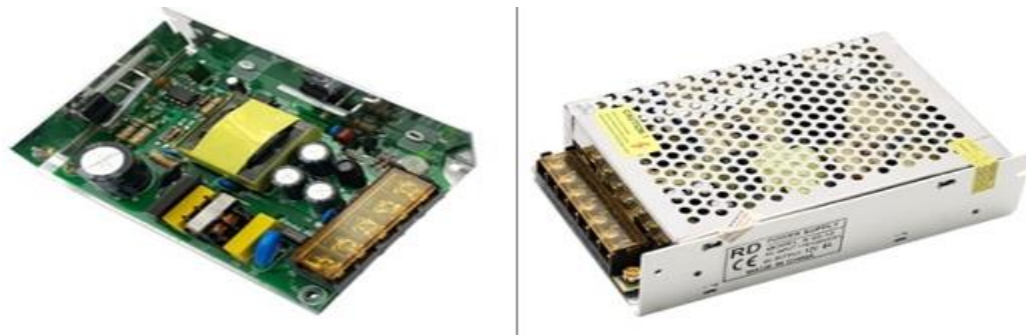
+ Số xung Encoder được quy ước là số lần ánh sáng chiếu qua khe. Ví dụ trên đĩa chỉ có 100 khe thì cứ 1 vòng quay, encoder đếm được 100 tín hiệu. Đây là nguyên lý hoạt động của loại Encoder cơ bản, còn đối với nhiều chủng loại khác thì đương nhiên đĩa quay sẽ có nhiều lỗ hơn và tín hiệu thu nhận cũng sẽ khác hơn.



Hình 10. Nguyên lý hoạt động Encoder

#### 2.2.5.Nguồn cấp:

Nguồn tổ ong 12V 5A được thiết kế để chuyển đổi điện áp từ nguồn xoay chiều 180V-240V thành nguồn một chiều 12VDC để cung cấp cho các thiết bị hoạt động. Khi công tắc điện mở, nguồn điện sẽ được đi qua nguồn xung, Khi đó cuộn sơ cấp của biến áp được đóng cắt điện liên tục bằng sò công suất và sẽ xuất hiện từ trường biến thiên. Dẫn đến cuộn thứ cấp của biến áp cũng xuất hiện một điện áp ra.



Hình 11. Nguồn tổ ong 12V-5A

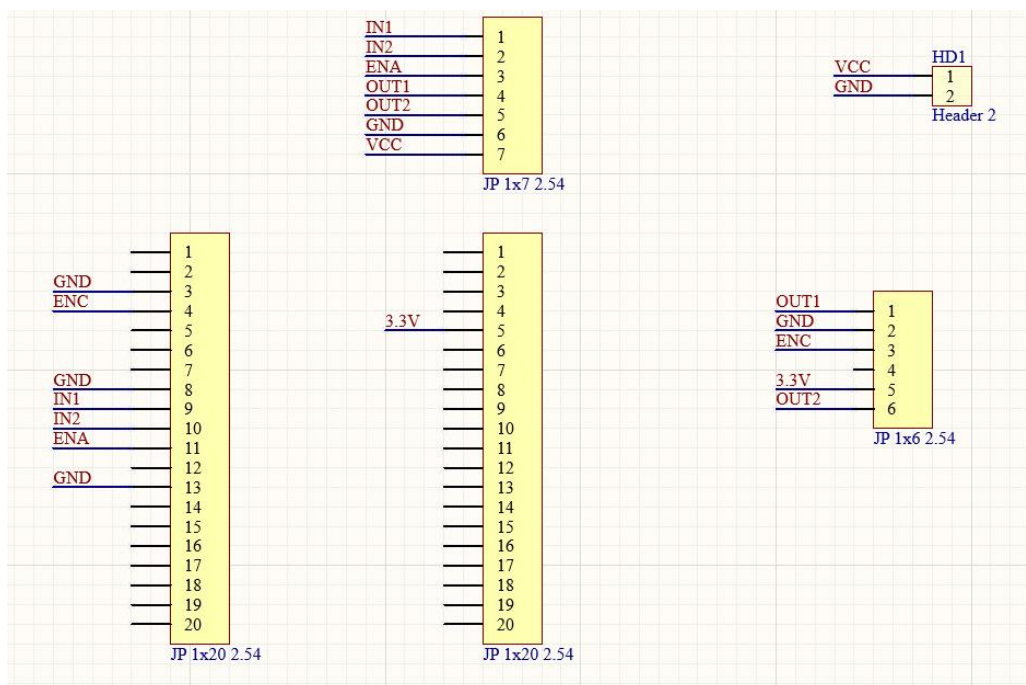
Thông số kỹ thuật:

- Điện áp đầu vào: 180V-240V.
- Tần số hoạt động: 47 ~ 63Hz.
- Công suất: 60W.
- Điện áp đầu ra: 12V.
- Dòng điện tối đa: 5A.
- Hiệu suất:  $\geq 85$
- Bảo vệ quá tải 105% -150%
- Công suất định mức, phục hồi tự động.
- Chức năng bảo vệ ngắn mạch tự động.
- Bảo vệ quá áp 105% -150% điện áp định mức.
- Nhiệt độ làm việc: - 200C ~ 600C.
- Nhiệt độ bảo quản: - 400C ~ 850C.
- Kích thước: 110\*78\*H36 (mm)

#### 2.2.6. Board mở rộng để kết nối các linh kiện:

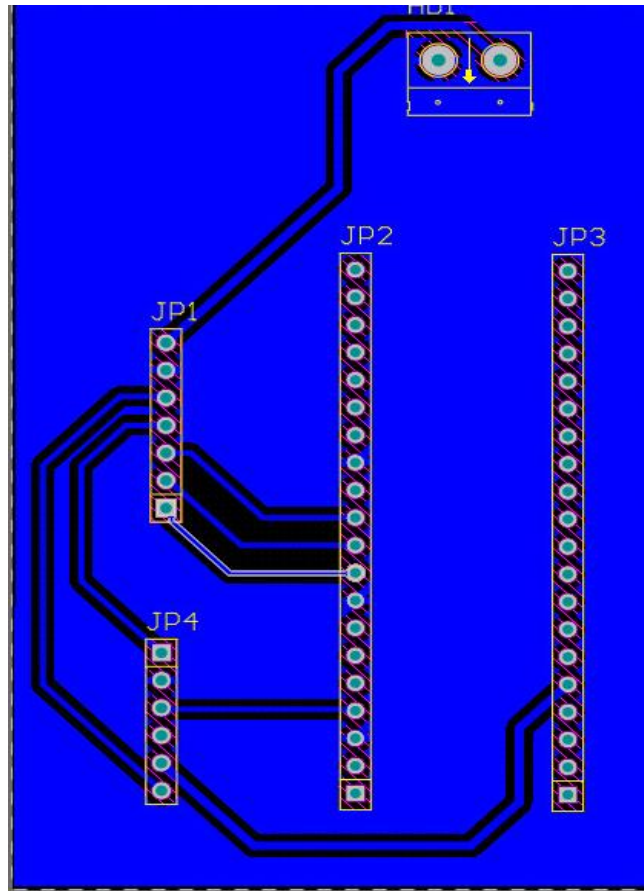
Để đảm bảo tính thẩm mỹ cho sản phẩm, ta sẽ giải quyết việc đi dây bằng cách tạo một Evaluate Board cho Raspberry Pi Pico RP2040.

Thực hiện vẽ PCB .



Hình 12. Mạch nguyên lý





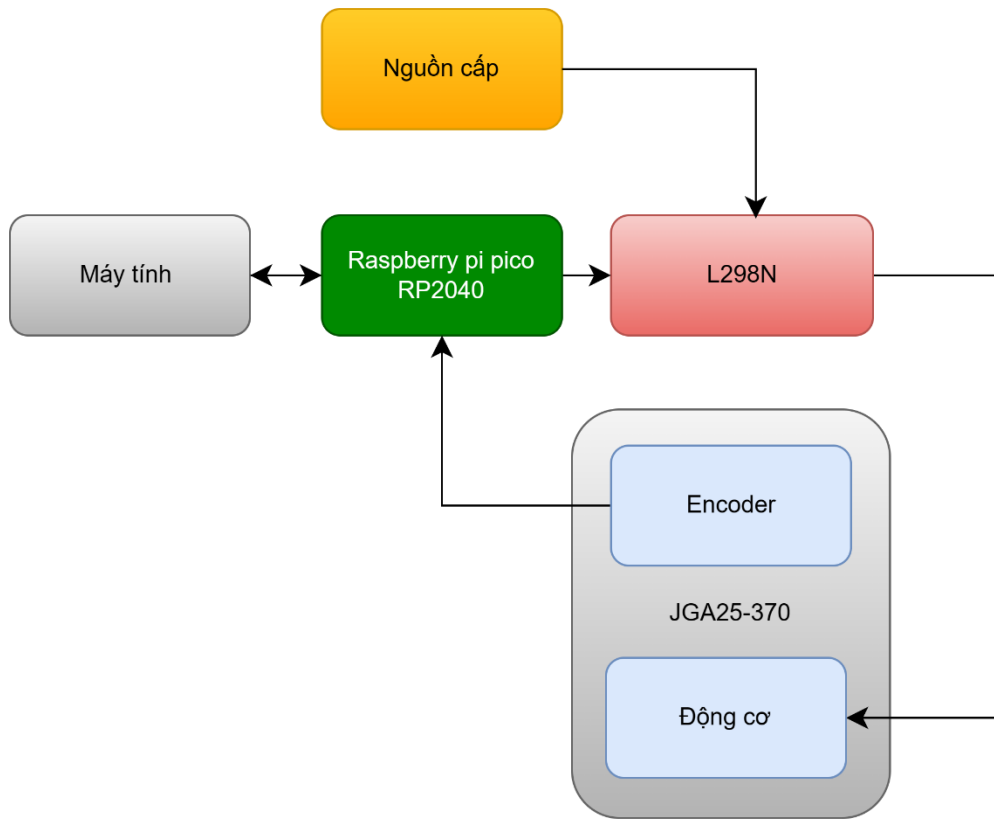
Hình 13. PCB Layout



Hình 14. Mạch thực tế

## CHƯƠNG 3: MẠCH NGUYÊN LÝ

### 3.1 Sơ đồ khối:



### 3.2 Sơ đồ nối dây:

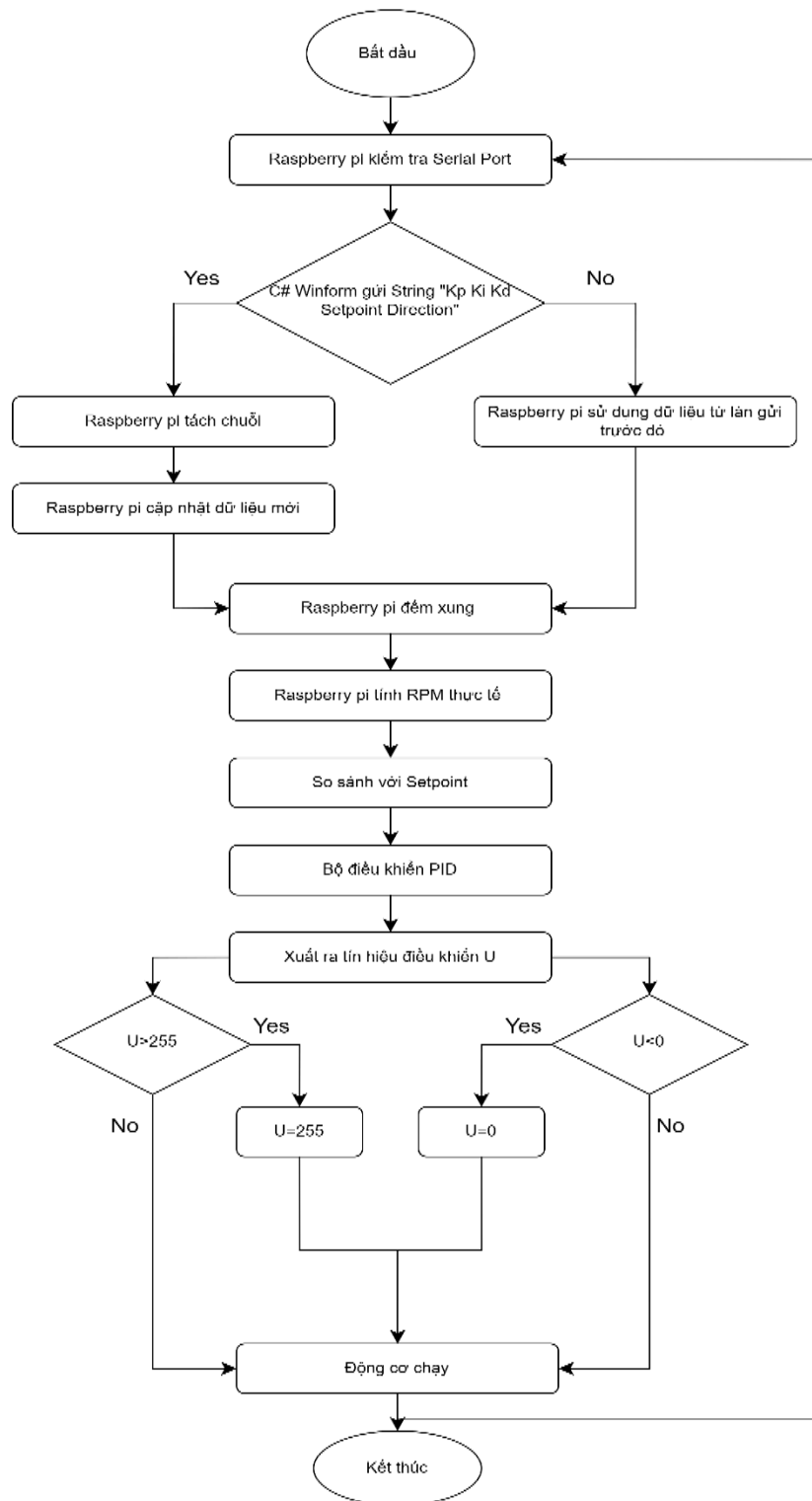
| Raspberry pi pico RP2040 | L298N | Động cơ JGB37-520 | Nguồn tổ ong |
|--------------------------|-------|-------------------|--------------|
|                          | OUT1  | M1                |              |
| GPIO2                    |       | C1/A              |              |
|                          |       | C2/B              |              |
| 3V3                      |       | VCC               |              |
|                          | OUT2  | M2                |              |
|                          | 12V   |                   | V+           |
| GND                      | GND   | GND               | V-           |
| GPIO6                    | IN1   |                   |              |
| GPIO7                    | IN2   |                   |              |
| GPIO8                    | ENA   |                   |              |

## CHƯƠNG 4: LƯU ĐỒ GIẢI THUẬT

### 4.1.Ý tưởng:

1. Ta nối dây theo sơ đồ.
2. Máy tính sẽ gửi String có dạng: "Kp Ki Kd Setpoint Direction" xuống Vi điều khiển qua giao tiếp Serial Port.
3. Raspberry pi pico RP2040 nhận chuỗi và xử lý tách chuỗi, đưa ra 5 thông số để xử lý .
4. Về phần Raspberry pi pico RP2040, ta sử dụng Ngắt ngoài ở GPIO2 để đếm xung mà Encoder trả về trong khoảng thời gian lấy mẫu Ts(đọc cạnh lên)
5. Từ xung đọc được, ta tính được tốc độ thực tế
6. So sánh tốc độ thực tế với tốc độ đặt, đưa qua bộ điều khiển PID, ta được tín hiệu điều khiển U xuất ra L298N
7. Lặp lại bước 4-6 cho tới khi xác lập .
8. Nếu có thay đổi từ C# Winform, lặp lại bước 2-6

#### 4.2. Sơ đồ giải thuật :



## CHƯƠNG 5: LẬP TRÌNH

### 5.1.Firmware( Raspberry Pi Pico RP2040):

#### 5.1.2.Chương trình code: (Trình bày ở phần Phụ Lục)

(Trình bày ở phần I của Phụ Lục)

#### 5.1.2,Phân tích code:

```
#include <SimpleKalmanFilter.h>
#include <stdio.h>
#include<string.h>
```

Đầu tiên, ta gọi thư viện SimpleKalmanFilter để lọc nhiễu môi trường hiệu quả hơn.

```
SimpleKalmanFilter filter(2, 2, 0.001);
double pulse; //đếm xung
double speed, Setpoint = 0; //tốc độ thực tế , tốc độ đặt
double E, E1, E2;
double alpha, gama, beta;
double Output, LastOutput; //PWM
const int IN1_PIN =7; //IN1
const int IN2_PIN =6; //IN2
const int SPEED_PIN = 8; //ENA
int encol = 2; //Chân đọc Encoder A
int enco2 = 3; //Chân đọc Encoder B
int waittime = 100; // Thời gian lấy mẫu
double T = (waittime * 1.0)/1000;
float Kp_temp=0,Ki_temp=0,Kd_temp=0,Setpoint_temp=0;
unsigned long counttime; //đếm thời gian
unsigned long present;
float Kp = 0, Kd = 0, Ki = 0; // Biến chứa thông số Ki Kp Kd từ C# Winform
String Direction;//Biến chứa chiều quay
```

Tiếp đến, ta khai báo các biến cần thiết trong quá trình hoạt động, bao gồm:

-Biến đếm xung: pulse

-Biến tốc độ :

- Tốc độ thực tế : Speed
- Tốc độ đặt :Setpoint

-Biến dùng trong phương trình sai phân:

- E1,E2,E: sai số giữa tốc độ đặt và thực tế
- Output, LastOutput: tín hiệu điều khiển

-Biến dùng trong lấy mẫu dữ liệu:

- Present
- Counttime
- Waittime

-Biến dữ liệu nhận được từ C# Winform:

- Kp,Ki,Kd: Các thông số của bộ điều khiển PID
- Setpoint: Tốc độ đặt

- Direction: Chiều quay
- Khai báo tên các chân:
- Chân điều khiển chiều quay động cơ: IN1,IN2
  - Chân xuất xung PWN điều khiển tốc độ động cơ : SPEED\_PIN
  - Chân đọc Encoder: enco1, enco2

```
void setup()
{
  pinMode(enco1, INPUT);//chân đọc encoder
  pinMode(enco2, INPUT);
  pinMode(SPEED_PIN, OUTPUT);//chân PWM
  pinMode(IN1_PIN, OUTPUT);//chân DIR1
  pinMode(IN2_PIN, OUTPUT);//chân DIR2
  speed=0;
  E = 0 ;
  E1 = 0;
  E2 = 0;
  Output=0;
  LastOutput=0;
  Serial.begin(9600);

  attachInterrupt(digitalPinToInterrupt(enco1), PulseCount, RISING); //khi có cạnh lên ở enco1,
  //ngắt ngoài xảy ra sẽ thực hiện Chương trình PulseCount
}

void PulseCount()
{
  pulse++;
}
```

Sau khi định nghĩa các biến,ta tiếp hành thiết lập tính năng ngõ ra/vào cho các chân và gán các giá trị ban đầu cho biến

Bắt đầu giao tiếp Serial Port với Baud Rate = 9600 bps

Tạo một ngắt ngoài cho chân enco1 và chương trình ngắt PulseCount, mục đích đếm xung trong một chu kỳ lấy mẫu.

Sau khi thiết lập xong, ta tiến hành vào chương trình chính :

```
//nhận chuỗi dữ liệu từ C# Winform
String data ;
while (Serial.available() > 0) {
  char c = Serial.read();
  data += c;
  delay(5);
}
//Loại bỏ kí tự " " ở cuối chuỗi
data.trim();
```

Liên tục đọc cổng Serial để nhận dữ liệu từ C# Winform gửi xuống Raspberry pi pico RP2040.

```
if (data == NULL){
    goto after_get_data;
}
else{
    const int maxTokens = 10; // Số lượng token tối đa
    char* tokens[maxTokens];
    int numTokens = 0;

    char* dataPtr = const_cast<char*>(data.c_str()); // Chuyển đổi kiểu dữ liệu
    char* token = strtok(dataPtr, " ");
    while (token != NULL && numTokens < maxTokens) {
        tokens[numTokens] = token;
        numTokens++;
        token = strtok(NULL, " ");
    }
    Kp=atof(tokens[0]);
    Ki=atof(tokens[1]);
    Kd=atof(tokens[2]);
    Setpoint=atof(tokens[3]);
    Direction= tokens[4];
    Output=0;

    analogWrite(SPEED_PIN, 0);
    digitalWrite(IN1_PIN, HIGH);
    digitalWrite(IN2_PIN, LOW);
}

after_get_data:
```

Nếu có dữ liệu được gửi xuống,ta sẽ thực hiện tách chuỗi để tìm ra 5 thông số Kp,Ki,Kd,Setpoint,Direction.

```
after_get_data:
    counttime = millis();
    if (counttime - present >= waittime)
    {
        present = counttime;
        speed = (pulse/(21.3*11))*(1/T)*60;

        Serial.println(String(filter.updateEstimate(speed)));
        pulse=0;

        // Tính toán giá trị ngõ ra PID rồi rạc
        E = Setpoint - speed;
        alpha = 2*T*Kp + Ki*T*T + 2*Kd;
        beta = T*T*Ki -4*Kd -2*T*Kp;
        gama = 2*Kd;
        Output = (alpha*E + beta*E1 + gama*E2 +2*T*LastOutput)/(2*T);
        if(Output>255){
            Output=255;}
        if(Output<0){
            Output=0;}
        LastOutput = Output;
        E2=E1;
        E1=E;
    }
}
```

Sau khi có các thông số cần thiết,ta tiến hành điều khiển tốc độ:

```
speed = (pulse/(21.3*11))*(1/T)*60;

Serial.println(String(filter.updateEstimate(speed)));
pulse=0;
```

-Tính tốc độ thực tế từ số xung đọc được.

```
// Tính toán giá trị ngõ ra PID rồi rạc
E = Setpoint - speed;
alpha = 2*T*Kp + Ki*T*T + 2*Kd;
beta = T*T*Ki - 4*Kd - 2*T*Kp;
gama = 2*Kd;
Output = (alpha*E + beta*E1 + gama*E2 + 2*T*LastOutput)/(2*T);
if(Output>255){
    Output=255;}
if(Output<0){
    Output=0;}
LastOutput = Output;
E2=E1;
E1=E;
}
```

So sánh tốc độ thực tế với tốc độ đặt, cho sai số đi qua bộ điều khiển PID để tính tín hiệu điều khiển.

```
counttime = millis();
if (counttime - present >= waittime)
{
    present = counttime;
```

Hàm millis() sẽ lấy thời gian thực thi chương trình hiện tại, ta trừ đi lần dùng millis() trước đó, nếu hiệu lớn hơn hoặc bằng 100ms thì sẽ thực hiện tính toán ngõ ra PID rồi rạc, tạo thành chu kỳ lấy mẫu.

```
//điều chỉnh chiều quay
analogWrite(SPEED_PIN, Output);
if(Direction == "Thuan"){
    digitalWrite(IN1_PIN, HIGH);
    digitalWrite(IN2_PIN, LOW);
}
if(Direction == "Nghich"){
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, HIGH);
}
```

Cuối cùng là chiều quay, sau khi có tín hiệu điều khiển, ta xuất ra chân SPEED\_PIN và kiểm tra chiều quay yêu cầu, sau đó tổ hợp 2 chân IN1\_PIN và IN2\_PIN để chỉnh chiều quay.

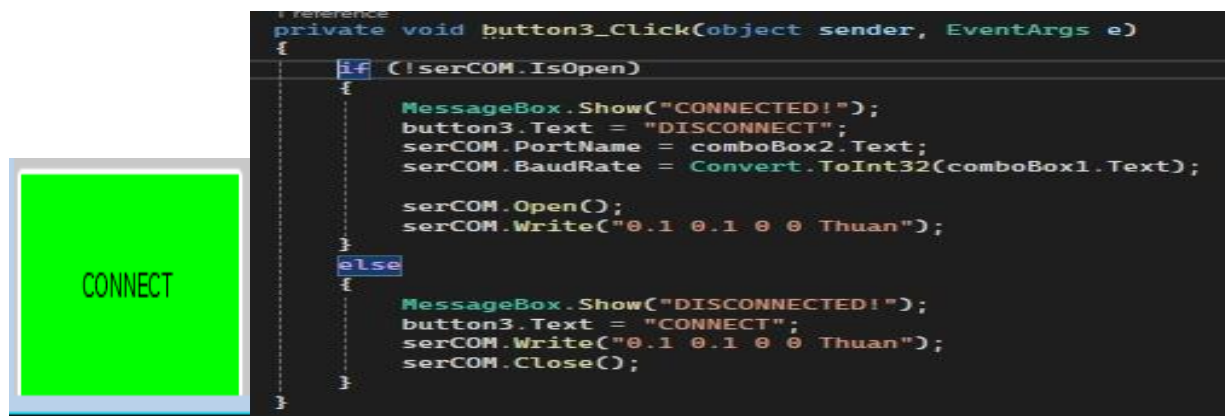


## 5.2.Software(Visual Studio 2022):

### 5.2.1.Giao diện:

#### 5.2.1.Phân tích code:

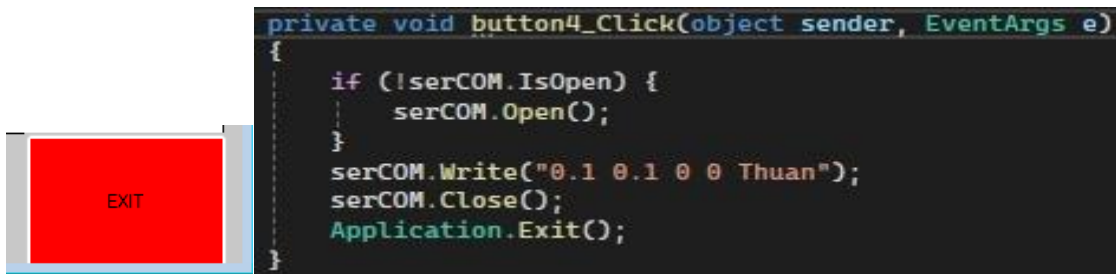
-Kết nối giữa C# Winform và Raspberry pi pico RP2040: sử dụng nút “CONNECT”, khi nhấn nút:



- Nếu serCOM(Serial Port Name) chưa được kết nối:
  - Sẽ có 1 Message Box hiện thông báo “CONNECTED!”.
  - Nút “CONNECT” chuyển thành “DISCONNECT”.
  - PortName và BaudRate sẽ được xác định qua 2 comboBox
  - Sau đó cổng COM được mở, gửi chuỗi “0.1 0.1 0 0 Thuan” để động cơ ngừng quay.

- Nếu serCOM đã được kết nối :
  - Sẽ có 1 Message Box hiện thông báo “DISCONNECTED!”
  - Nút “DISCONNECT” chuyển thành “CONNECT”.
  - Gửi chuỗi “0.1 0.1 0 0 Thuan” để động cơ ngừng quay.
  - Đóng cổng COM lại, ngừng giao tiếp giữa C# Winform và Raspberry pi pico RP2040

-Thoát ứng dụng:



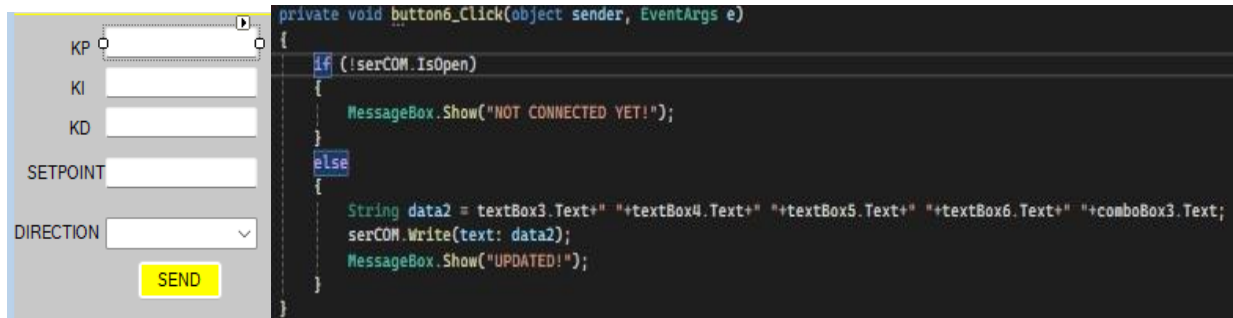
- Nếu cổng COM chưa mở thì mở cổng COM
- Gửi chuỗi “0.1 0.1 0 0 Thuan” xuống Raspberry pi pico RP2040.
- Đóng cổng COM.
- Thoát ứng dụng

-Chọn BaudRate và cổng COM:



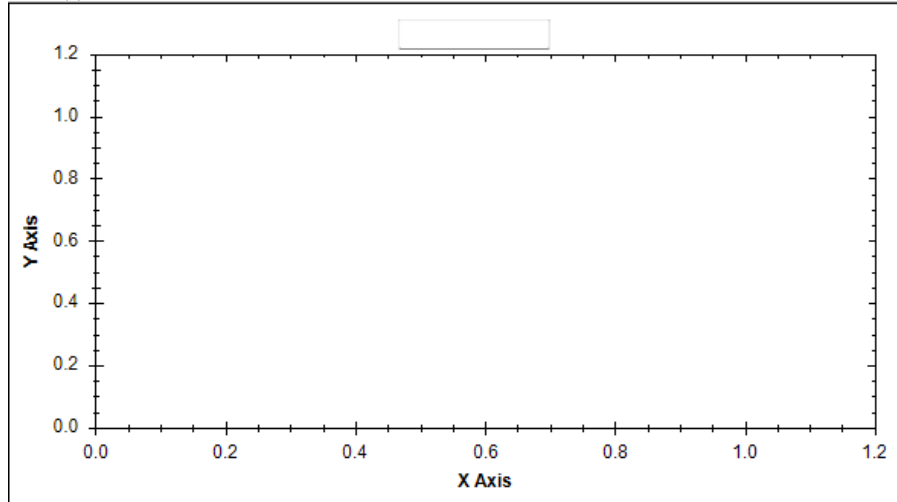
- Tạo comboBox BaudRate với những giá trị chọn được từ String BaudRate được tạo
- Tạo comboBox COM với những giá trị được máy tính lấy bằng lệnh `comboBox2.DataSource=SerialPort.GetPortNames();`

-Gửi dữ liệu từ C# Winform xuống Raspberry pi pico RP2040:



- Kiểm tra kết nối giữa C# Winform và Raspberry pi pico RP2040
- Gửi String data2 có dạng “Kp Ki Kd Setpoint Direction” xuống Raspberry pi pico RP2040

-Vẽ đồ thi RPM(t):



- Sử dụng thư viện Zedgraph `using ZedGraph;`
- Lấy 500.000 mẫu dữ liệu `RollingPointPairList list = new RollingPointPairList(500000);`

- Cài đặt chức năng AutoScale cho đồ thi

```
zedGraphControl1.AxisChange();
graph.XAxis.ResetAutoScale(zedGraphControl1.GraphPane, CreateGraphics());
```

- Cài đặt trục x,y:

```
graph.XAxis.Scale.Max = 10;
graph.XAxis.Scale.Min = 0;
graph.XAxis.Scale.MinorStep = 1;
graph.XAxis.Scale.MajorStep = 1;
graph.YAxis.Scale.Min = 0;
graph.YAxis.Scale.Max = 300;
graph.YAxis.Scale.MinorStep = 1;
graph.YAxis.Scale.MajorStep = 1;
```

- Tạo đường line dữ liệu :

```
LineItem line = graph.AddCurve("data", list, Color.Red, SymbolType.None);
```

- Tạo hàm draw để vẽ line:

```
double tong = 0;
1 reference
public void draw(double line) {

    LineItem duongline = zedGraphControl1.GraphPane.CurveList[0] as LineItem;
    if (duongline == null) {
        return;
    }
    IPointListEdit list = duongline.Points as IPointListEdit;
    if (list == null)
    {
        return;
    }
    list.Add(tong, line);
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();
    tong += 0.1;
}
```

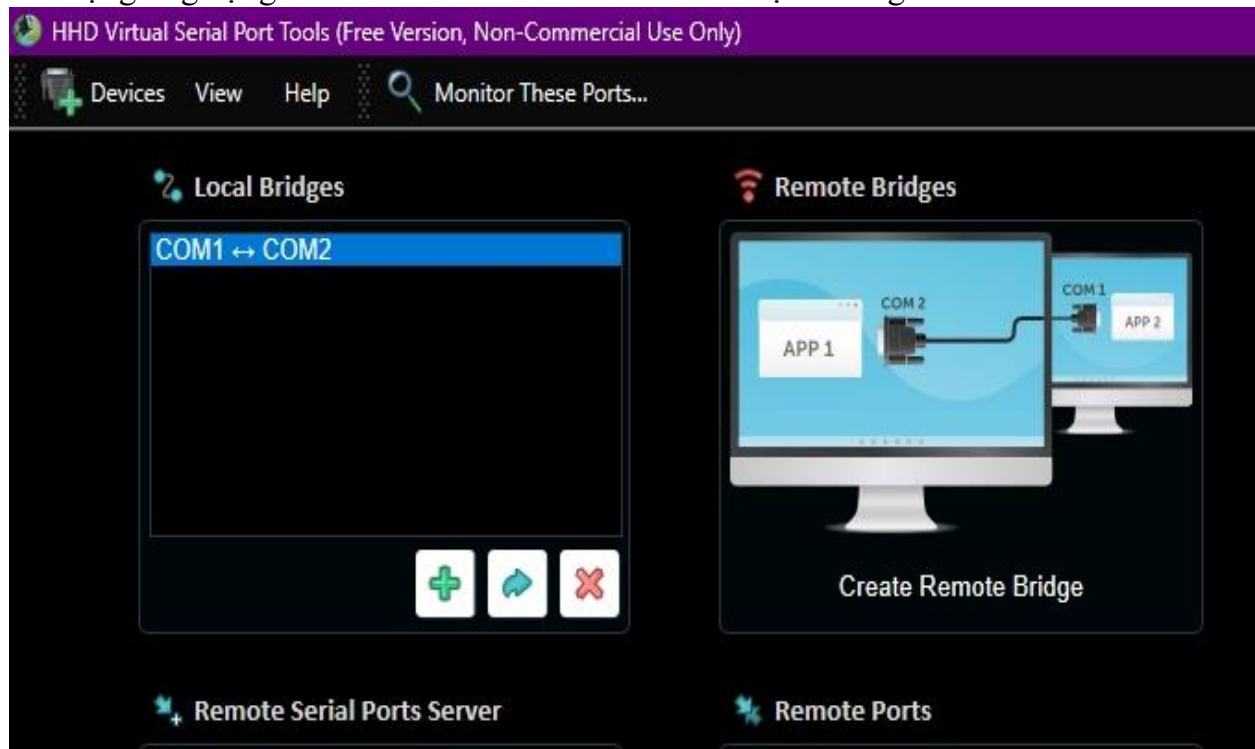
0.1 là giá trị trực thời gian công thêm sau một lần lấy mẫu (100ms)

- Nhận dữ liệu từ Serial Port để vẽ đồ thị :

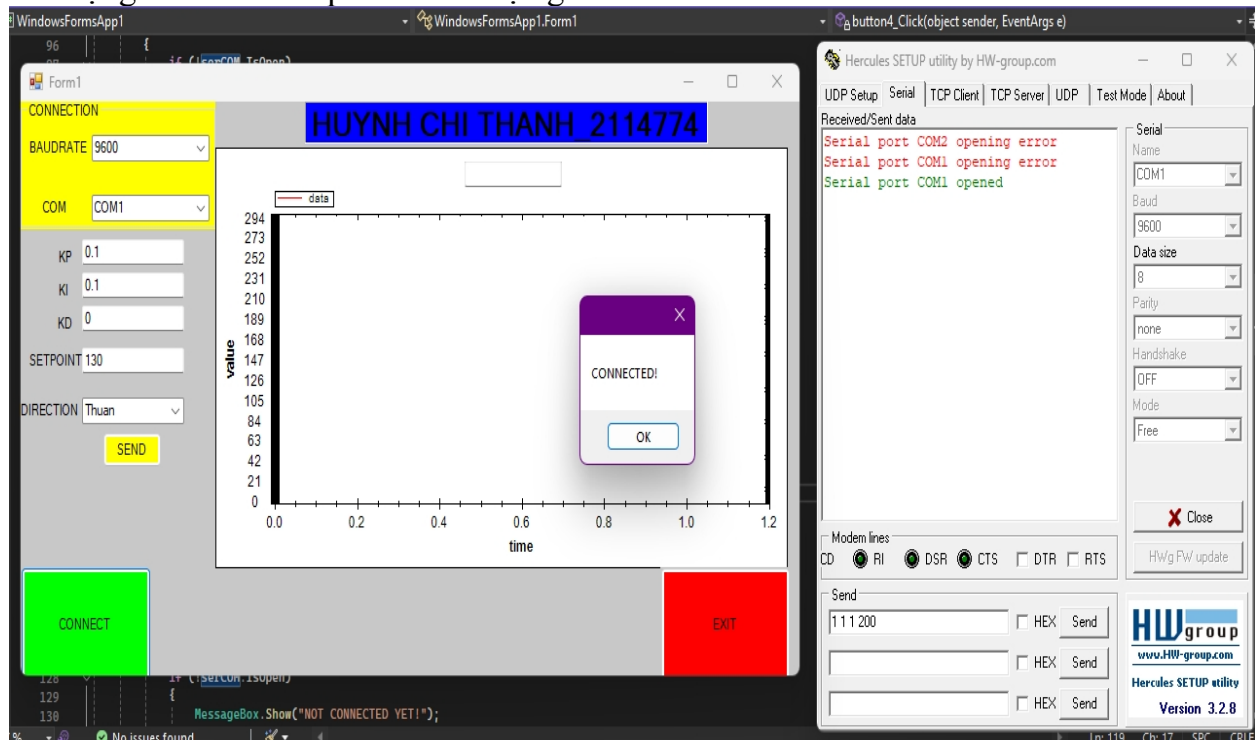
```
private void serCOM_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    String data1 = "";
    data1 = serCOM.ReadLine();
    int len = data1.Length;
    if( len > 0){
        textBox2.Text = data1;
        double data2;
        if (double.TryParse(data1, out data2))
        {
            // Conversion successful, use parsedValue
            Invoke(new MethodInvoker(() => draw(data2)));
        }
        else
        {
            MessageBox.Show("WRONG FORMAT!");
            // Handle invalid input (e.g., show an error message)
        }
        // Invoke(new MethodInvoker(() => draw(Convert.ToDouble(data1))));
    }
}
```

- Kiểm tra các chức năng của C# Winform:

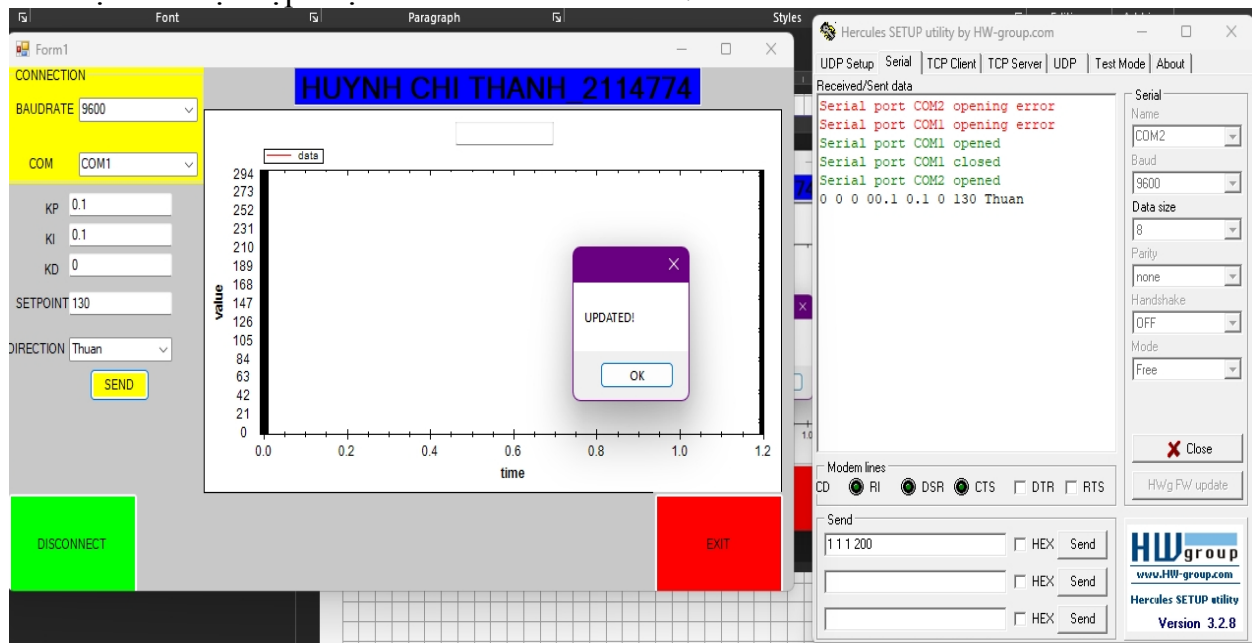
-Sử dụng ứng dụng HHD Virtual Serial Port Tools để tạo 2 cổng COM ảo



-Sử dụng Hercules để quan sát dữ liệu gửi từ C# Winform:



-Dữ liệu đã được cập nhật sau khi nhấn “SEND”:



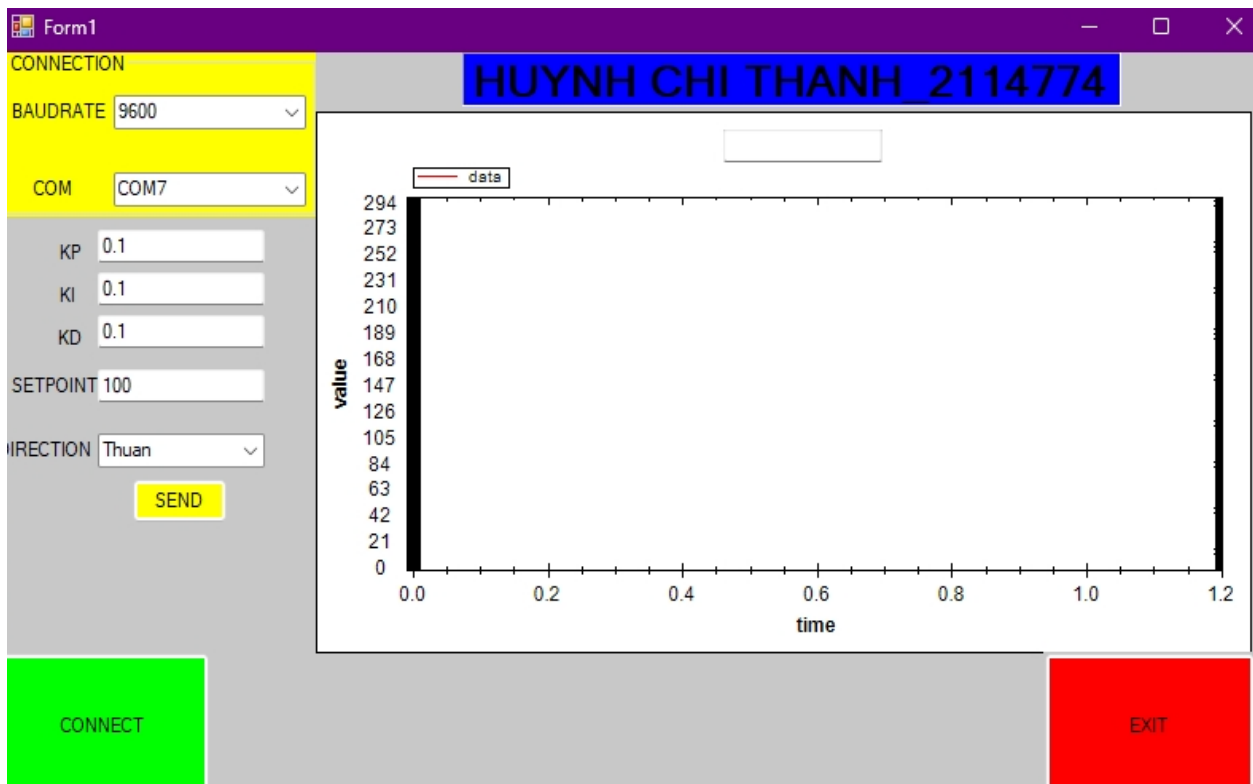
## CHƯƠNG 6: KẾT QUẢ THỰC NGHIỆM

### 6.1. Mô hình thực tế:

Đây là mô hình thực tế của em:

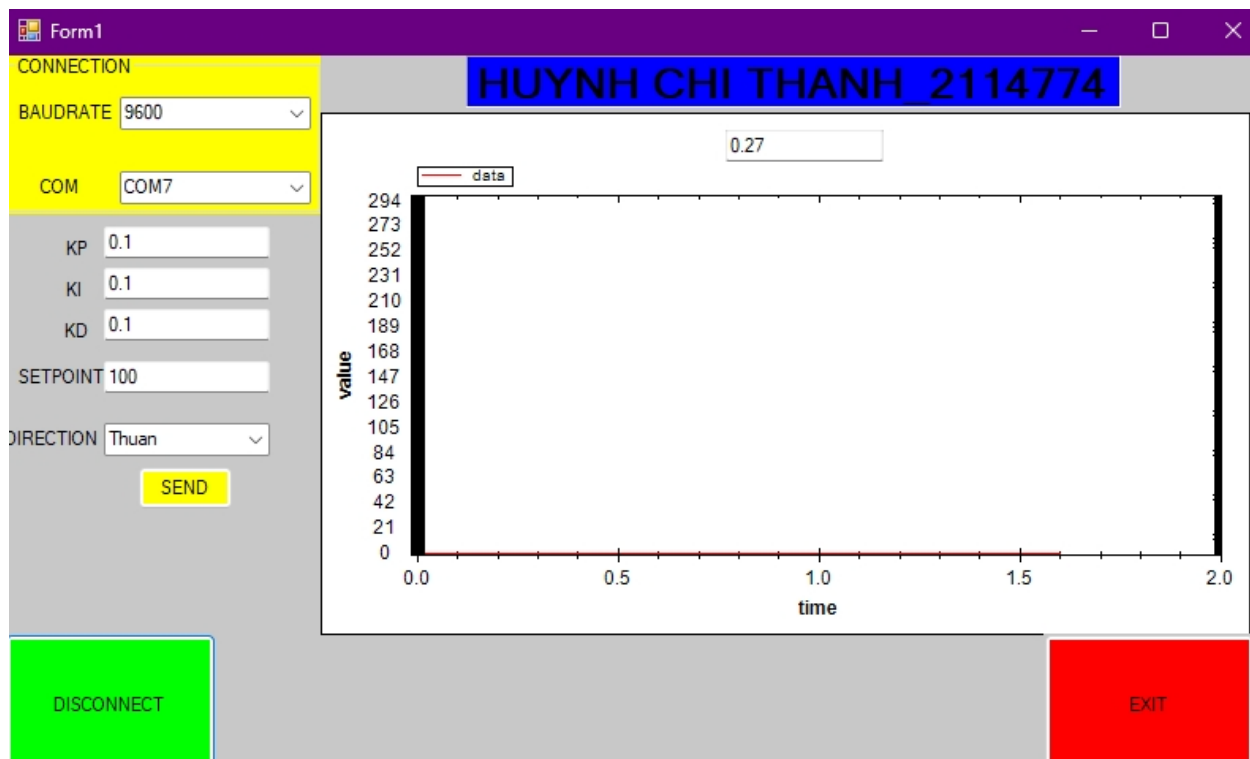


Khi vừa khởi động, giao diện HMI sẽ như hình dưới:

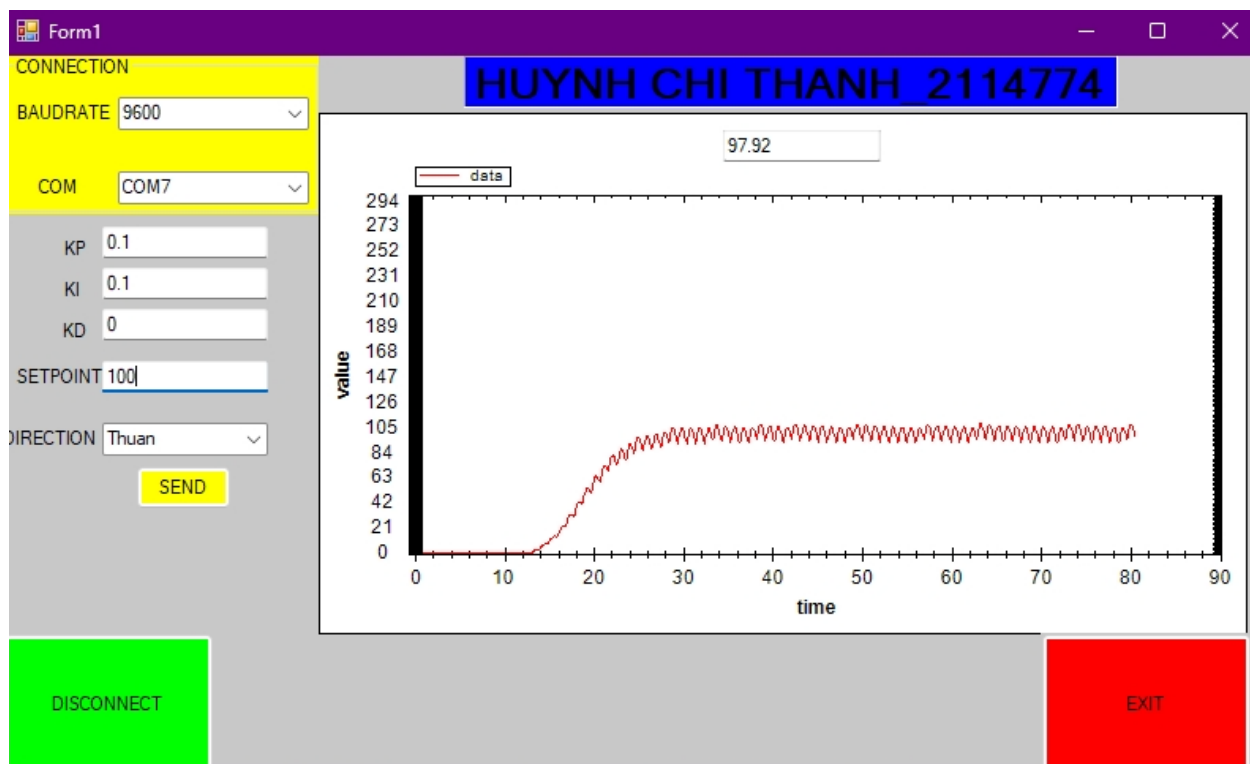




Sau khi kết nối, dữ liệu sẽ bắt đầu được cập nhật trên Zedgraph:

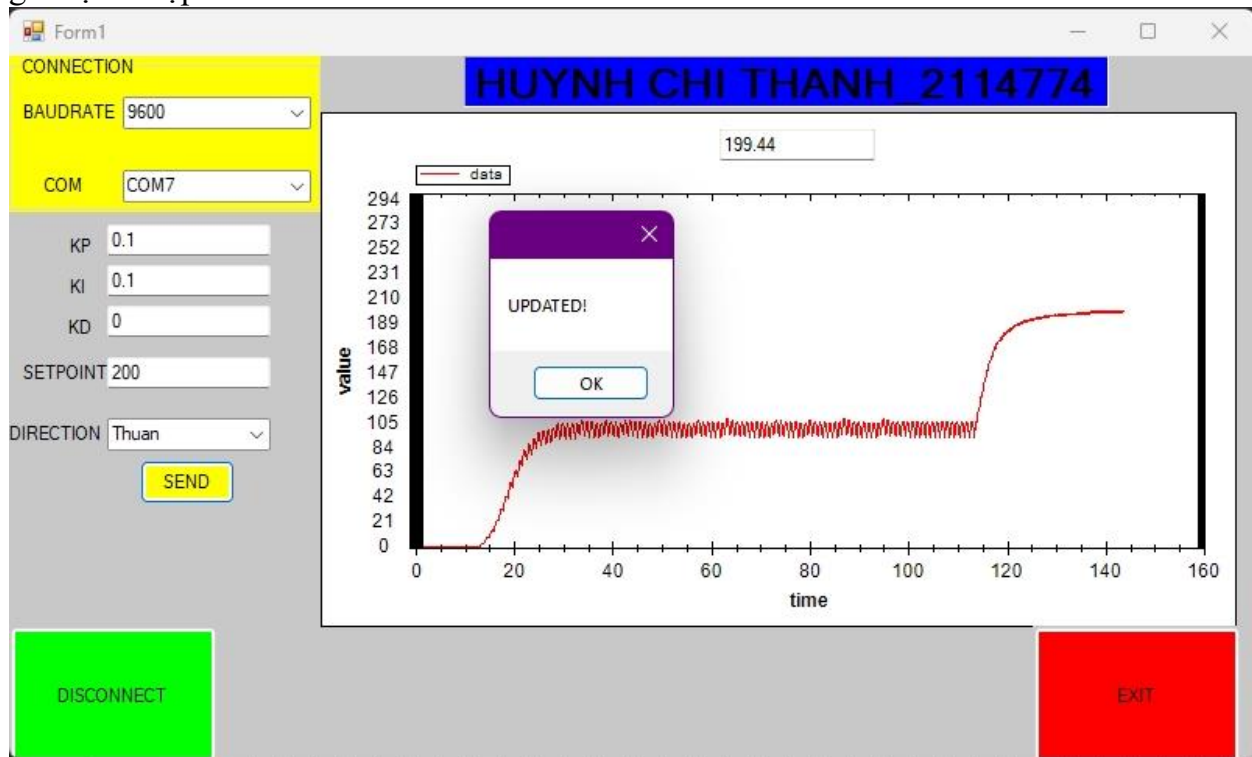


Đáp ứng khi đặt Setpoint bằng 100 RPM





Khi nhấn “SEND” lần nữa, Setpoint mới sẽ được cập nhật và đáp ứng nhanh chóng đi đến giá trị xác lập ở 200 RPM



- ❖ Nhận xét: Có dao động xung quanh giá trị xác lập  
=> Cần chỉnh hệ số PID dùng phương pháp Ziegler-Nichols

## 6.2. Tìm thông số PID:

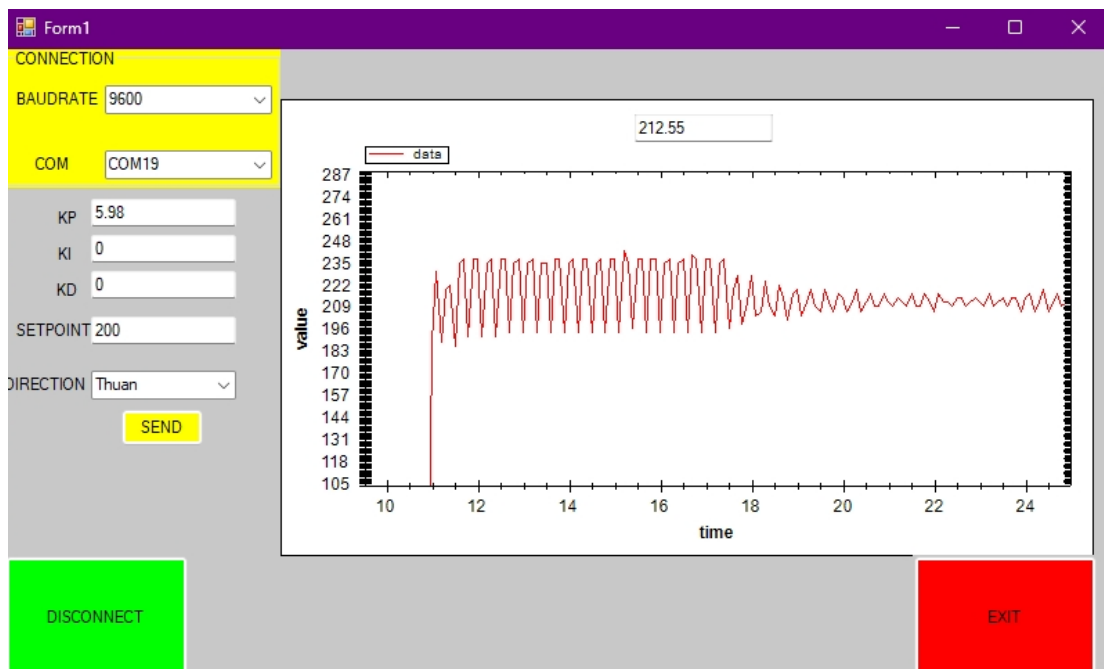
Ta sử dụng phương pháp Ziegler-Nichols.

Cho thông số  $K_i = 0$ ,  $K_d = 0$ , tăng dần  $K_p$  đến khi hệ có dao động, giá trị  $K_p$  chính là  $K_{gh}$ .

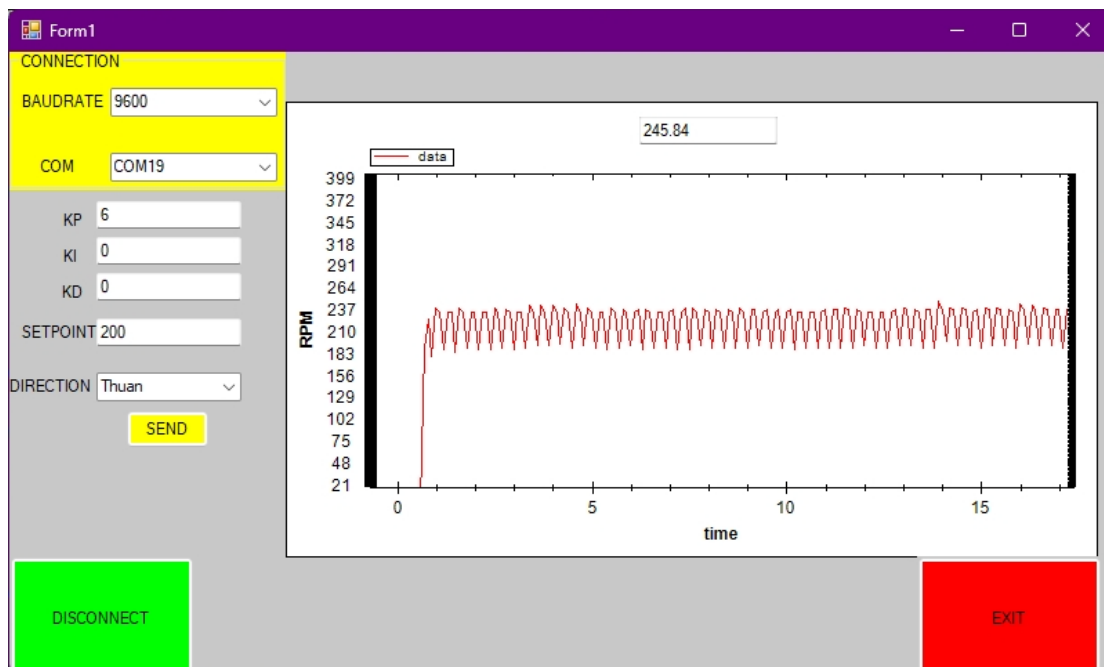
|                       | $K_p$           | $T_i$     | $T_d$   |
|-----------------------|-----------------|-----------|---------|
| <b>P Controller</b>   | $0.5 * K_{gh}$  | $\infty$  | 0       |
| <b>PI Controller</b>  | $0.45 * K_{gh}$ | $T_u/1.2$ | 0       |
| <b>PID Controller</b> | $0.5 * K_{gh}$  | $T_u/2$   | $T_u/8$ |

*Bảng thông số bộ điều khiển PID theo phương pháp Ziegler-Nichols*

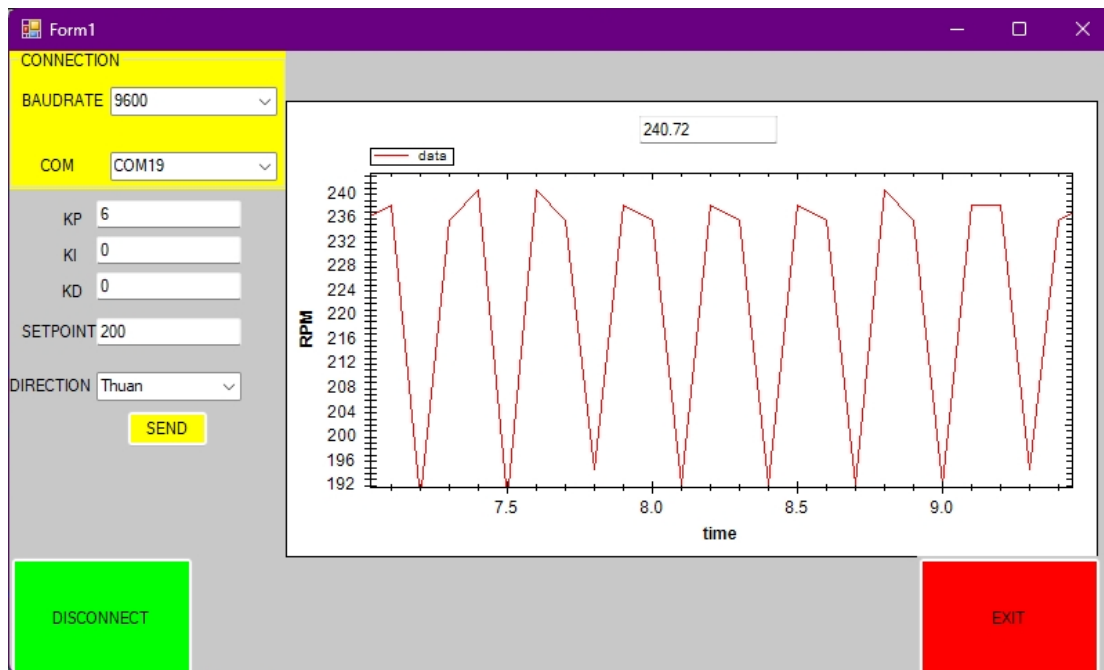
Khảo sát tại  $K_p=5.98$



Khảo sát tại  $K_p= 6.0$



Đáp ứng ngõ ra bắt đầu xuất hiện dao động nên ta chọn  $K_{gh}= 6.0$



Phóng to đồ thị, ta tìm được  $T_u = 0.3$  s

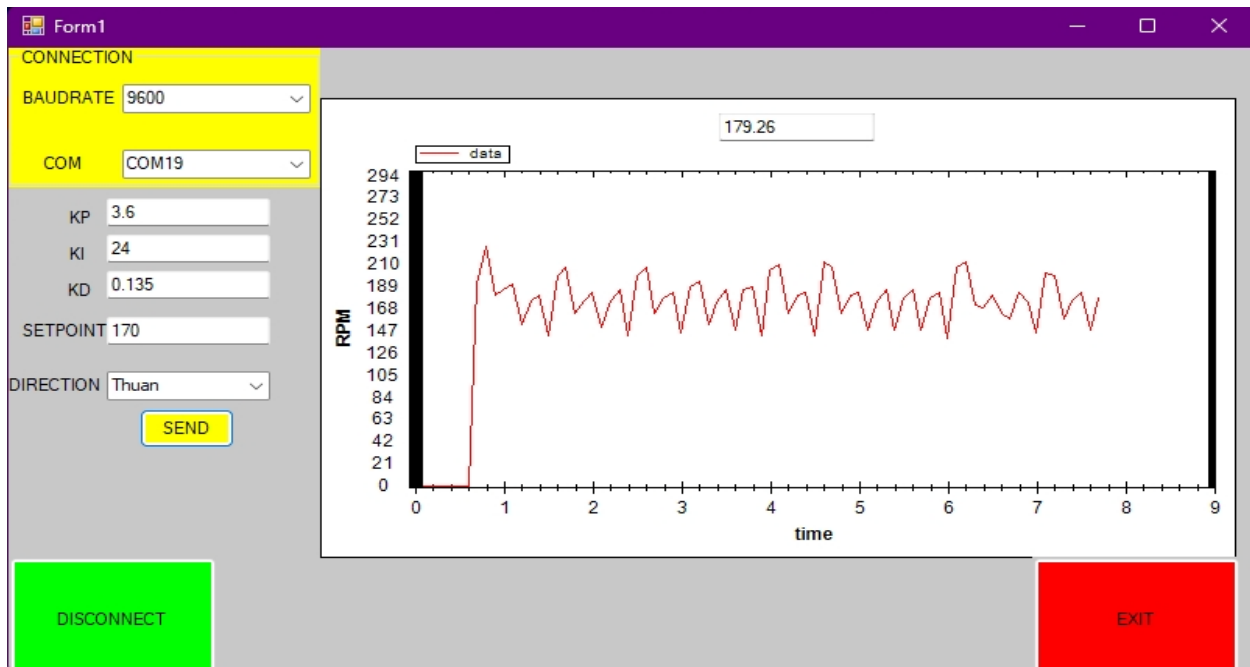
Với  $K_{gh} = 6.0$ ,  $T_u = 0.3$ s, ta tính được các thông số:

$$K_p = 0.6 * K_{gh} = 0.6 * 6 = 3.6$$

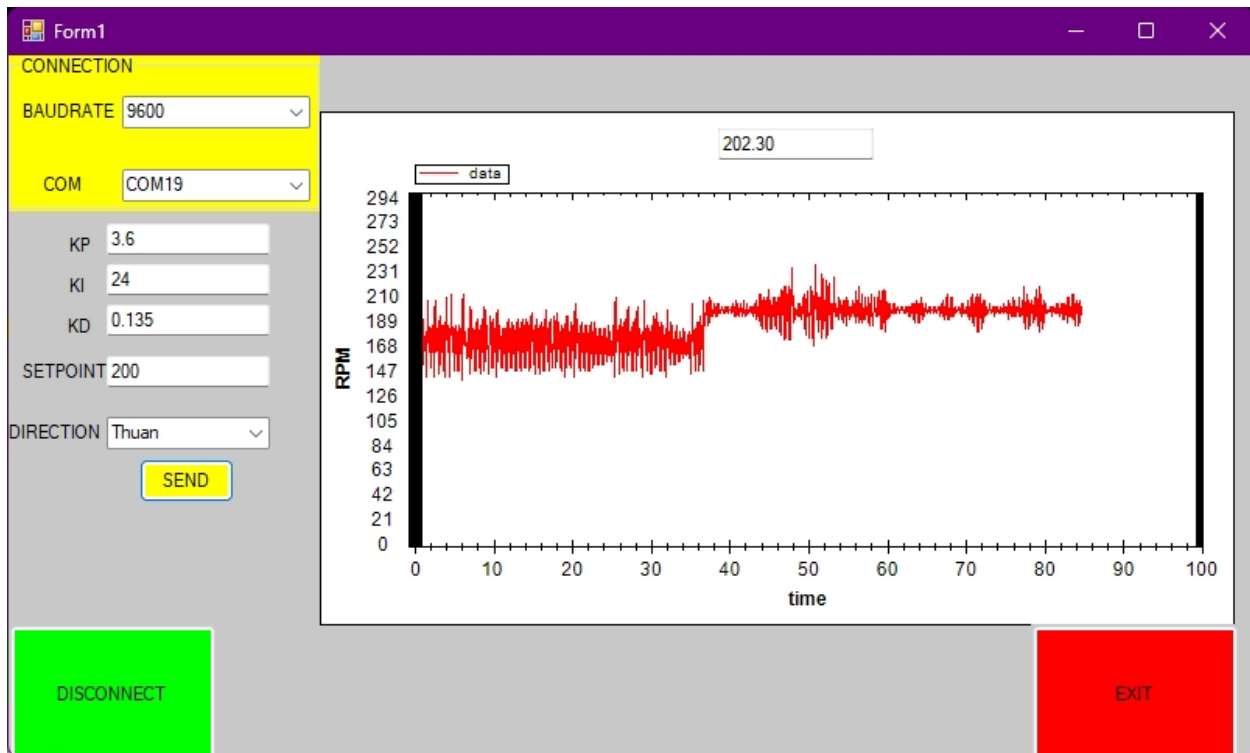
$$T_i = \frac{T_u - 0.3}{2} = 0.15 \rightarrow K_i = \frac{K_p}{T_i} = \frac{3.6}{0.15} = 24$$

$$T_d = \frac{T_u - 0.3}{8} = 0.0375 \rightarrow K_d = K_p * T_d = 3.6 * 0.0375 = 0.135$$

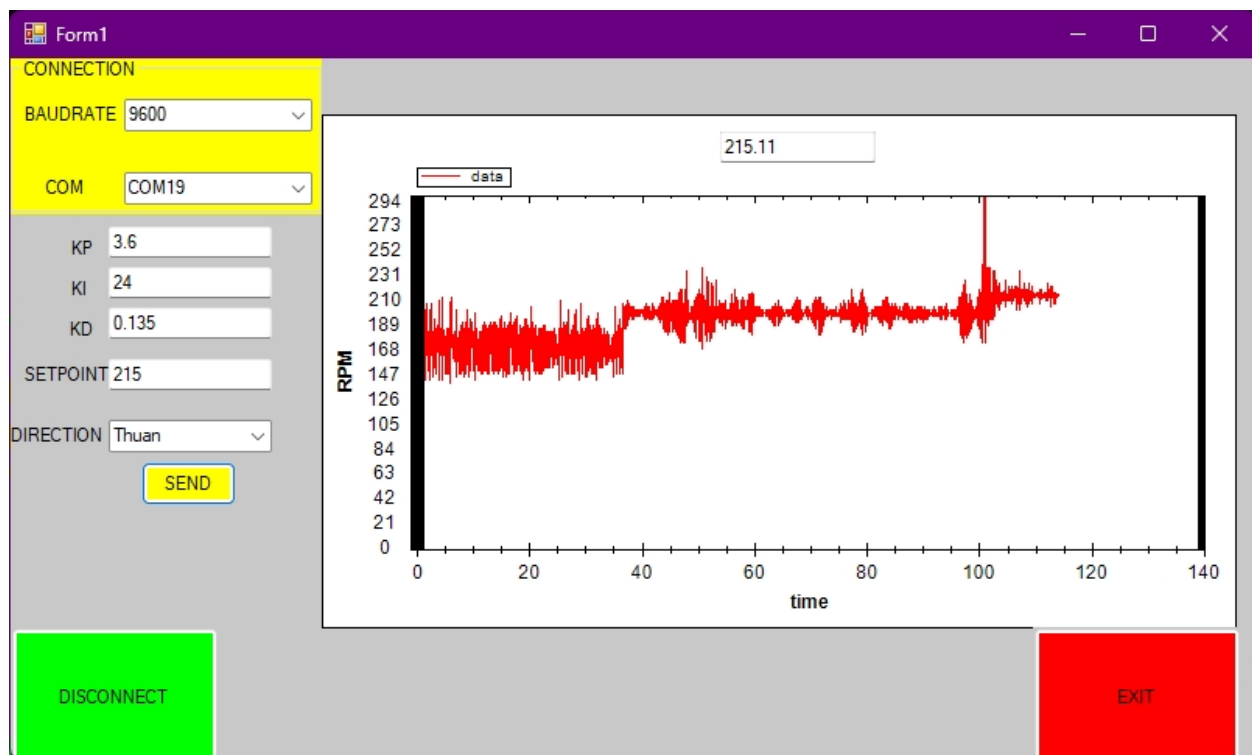
❖ Kết quả khảo sát :



Đồ thị đáp ứng với Kp, Ki, Kd vừa tính được tại Setpoint =170 RPM



Đồ thị đáp ứng với Kp, Ki, Kd vừa tính được tại Setpoint =170 RPM



Đồ thị đáp ứng với  $K_p$ ,  $K_i$ ,  $K_d$  vừa tính được tại Setpoint = 215 RPM

## KẾT LUẬN

Sau thời gian nghiên cứu em đã hoàn thành với các kết quả đạt được như sau:

- Biết được cách áp dụng bộ điều khiển PID vào mô hình thực tế.
  - Tìm hiểu cách sử dụng Encoder.
  - Viết code, dùng ngôn ngữ C, Tìm hiểu thêm về các bo mạch Raspberry pi pico RP2040.
- Tuy nhiên Bài tập lớn vẫn còn một số vấn đề tồn tại, hạn chế cần giải quyết:
- Đáp ứng của động cơ còn chưa tối ưu
  - C# Winform đôi khi bị treo vì giới hạn phần cứng.
  - Phần cứng còn chưa gọn gàng, chống nhiễu cơ khí chưa hiệu quả.

Do đó em rất mong sẽ nhận được những ý kiến đóng góp của thầy và các bạn để đề tài của chúng em ngày một được hoàn thiện hơn.

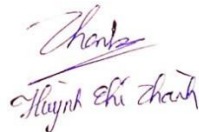
Hướng phát triển đề tài:

- Xây dựng bộ điều khiển PID cho hệ thống điều khiển vị trí.
- Ứng dụng điều khiển xe cân bằng.
- Ứng dụng điều khiển xe dò line.

Xin chân thành cảm ơn thầy Nguyễn Phan Hải Phú đã hướng dẫn em trong quá trình hoàn thành Bài tập lớn môn Thiết kế hệ thống nhúng.

TP. Hồ Chí Minh, ngày 29 tháng 11 năm 2024

Sinh viên thực hiện:



Huỳnh Chí Thành

Sinh viên thực hiện:



Võ Anh Khoa

## TÀI LIỆU THAM KHẢO

1. Bộ môn Điều khiển Tự động – Khoa Điện-Điện tử - ĐH Bách Khoa TPHCM. Bài Thí Nghiệm 2 – Điều khiển hệ quạt và tấm phẳng. Truy cập từ : [Bai 2 - Dieu khien he quat va tam phang.pdf \(hcmut.edu.vn\)](#)
2. Phạm Trung Đức và Nguyễn Mạnh Tùng (07/2023). Điều khiển tốc độ động cơ dùng PID Và Arduino. Truy cập từ: [Điều khiển tốc độ động cơ dùng PID Và Arduino - TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HÙNG YÊN KHOA ĐIỆN – - Studocu](#)
3. Cửa hàng HSHOP ĐIỆN TỬ VÀ ROBOT. Động Cơ DC Servo GA25-370 DC Geared Motor. Truy cập từ: [Động Cơ DC Servo GA25-370 DC Geared Motor – Hshop.vn](#)
4. Raspberry Pi. Raspberry Pi Documentation - Raspberry Pi Pico and Pico W. Truy cập từ : [Raspberry Pi Pico and Pico W - Raspberry Pi Documentation](#)
5. PGS-TS Nguyễn Thị Phương Hà. Lý thuyết điều khiển tự động. Truy cập từ : [tailieuXANH - Bài giảng lý thuyết điều khiển tự động - Mô hình toán học, hệ thống điều khiển liên tục part 1](#)
6. Renzo Mischianti(26/09/2022). Schede Raspberry Pi Pico e rp2040: pinout, specifiche e configurazione IDE Arduino – 1. Truy cập từ: [Schede Raspberry Pi Pico e rp2040: pinout, specifiche e configurazione IDE Arduino – 1 – Renzo Mischianti](#)
7. STMICROELECTRONICS [STMicroelectronics](01/2000). DUAL FULL BRIDGE DRIVER. Truy cập từ: [L298N bảng dữ liệu\(PDF\) - STMicroelectronics \(alldatasheet.vn\)](#)
8. Amazen (16/04/2022). ENCODER LÀ GÌ? CẤU TẠO, NGUYÊN LÝ, ỨNG DỤNG CỦA ENCODER. Truy cập từ : [Encoder là gì? Cấu tạo, nguyên lý, ứng dụng của Encoder|Amazen](#)

## DANH MỤC HÌNH ẢNH

1. **Hình 1:** Bộ môn Điều khiển Tự động-Khoa Điện-Điện tử-ĐH Bách Khoa TPHCM. Bài thí nghiệm 4: Điều khiển đối tượng có trễ. Truy cập từ: [Bai 4 - Dieu khien doi tuong co tre.pdf \(hcmut.edu.vn\)](#)
2. **Hình 2:** instructable\_translator(16/06/2016). Cách dùng Module điều khiển động cơ L298N - cầu H để điều khiển động cơ DC. Truy cập từ: [Cách dùng Module điều khiển động cơ L298N - cầu H để điều khiển động cơ DC | Công đồng Arduino Việt Nam](#)
3. **Hình 6:** Renzo Mischianti(26/09/2022). Schede Raspberry Pi Pico e rp2040: pinout, specifiche e configurazione IDE Arduino – 1. Truy cập từ: [Schede Raspberry Pi Pico e rp2040: pinout, specifiche e configurazione IDE Arduino – 1 – Renzo Mischianti](#)
4. **Hình 7:** Cửa hàng HSHOP ĐIỆN TỬ VÀ ROBOT. Động Cơ DC Servo GA25-370 DC Geared Motor. Truy cập từ: [Động Cơ DC Servo GA25-370 DC Geared Motor – Hshop.vn](#)
5. **Hình 8:** Cửa hàng HSHOP ĐIỆN TỬ VÀ ROBOT. Động Cơ DC Servo GA25-370 DC Geared Motor. Truy cập từ: [Động Cơ DC Servo GA25-370 DC Geared Motor – Hshop.vn](#)
6. **Hình 9:** Amazen(16/04/2022). ENCODER LÀ GÌ? CẤU TẠO, NGUYÊN LÝ, ỨNG DỤNG CỦA ENCODER. Truy cập từ : [Encoder là gì? Cấu tạo, nguyên lý, ứng dụng của Encoder|Amazen](#)
7. **Hình 10:** Amazen(16/04/2022).. ENCODER LÀ GÌ? CẤU TẠO, NGUYÊN LÝ, ỨNG DỤNG CỦA ENCODER. Truy cập từ : [Encoder là gì? Cấu tạo, nguyên lý, ứng dụng của Encoder|Amazen](#)
8. **Hình 11:** Phạm Trung Đức và Nguyễn Mạnh Tùng(07/2023). Điều khiển tốc độ động cơ dùng PID Và Arduino. Truy cập từ: [Điều khiển tốc độ động cơ dùng PID Và Arduino - TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HÙNG YÊN KHOA ĐIỆN – - Studocu](#)