ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ
----------o0o----------

BÀI TẬP LỚN THIẾT KẾ HỆ THỐNG NHÚNG

ĐỀ TÀI: DC MOTOR CONTROLLER USING PWM

GVHD: Nguyễn Phan Hải Phú

Nhóm: 09 - L03

Sinh viên thực hiện:

STT	MSSV	Họ và Tên	Phân công nhiệm vụ	Đánh giá
1	2310346	Hồ Bá Chiến	Lập trình phần mềm.	100%
2	2310321	Nguyễn Thanh Ca	Thiết kế sơ đồ mạch. Tổng quan lý thuyết.	100%
3	2310093	Nguyễn Đoàn Quốc Anh	Thiết kế sơ đồ mạch.	100%
4	2312479	Nguyễn Khắc Minh Nhật	Lập trình phần mềm.	100%

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2025

TÓM TẮT ĐỀ TÀI

Đề tài DC motor controller using pwm trình bày về việc thiết kế và thi công hệ thống điều khiển tốc độ của động cơ điện một chiều (DC Motor) sử dụng kỹ thuật điều chế độ rộng xung (PWM). Hệ thống sử dụng vi điều khiển STM32F103C8 làm bộ xử lý trung tâm để tạo xung PWM và tín hiệu điều khiển, kết hợp với module công suất L298N để lái động cơ. Kết quả thực nghiệm sẽ cho thấy hệ thống hoạt động ổn định, đáp ứng tốt các yêu cầu thay đổi tốc độ và đảo chiều quay.

DANH SÁCH HÌNH MINH HỌA

1. GIỚI THIỆU

1.1. Tổng quan

Động cơ DC được sử dụng rộng rãi trong các ứng dụng công nghiệp và dân dụng như robot, băng tải, và các thiết bị tự động hóa. Vấn đề điều khiển chính xác tốc độ và chiều quay của động cơ là một yêu cầu cơ bản. Phương pháp phổ biến và hiệu quả nhất hiện nay là sử dụng kỹ thuật PWM (Pulse Width Modulation).

Vi điều khiển STM32F103C8 với khả năng xử lý mạnh mẽ và các bộ Timer tích hợp hiện đại là lựa chọn tối ưu để thực hiện giải thuật PWM. Kết hợp với mạch cầu H (Module L298N), hệ thống có thể điều khiển động cơ công suất vừa và nhỏ một cách linh hoạt.

1.2. Nhiệm vụ đề tài

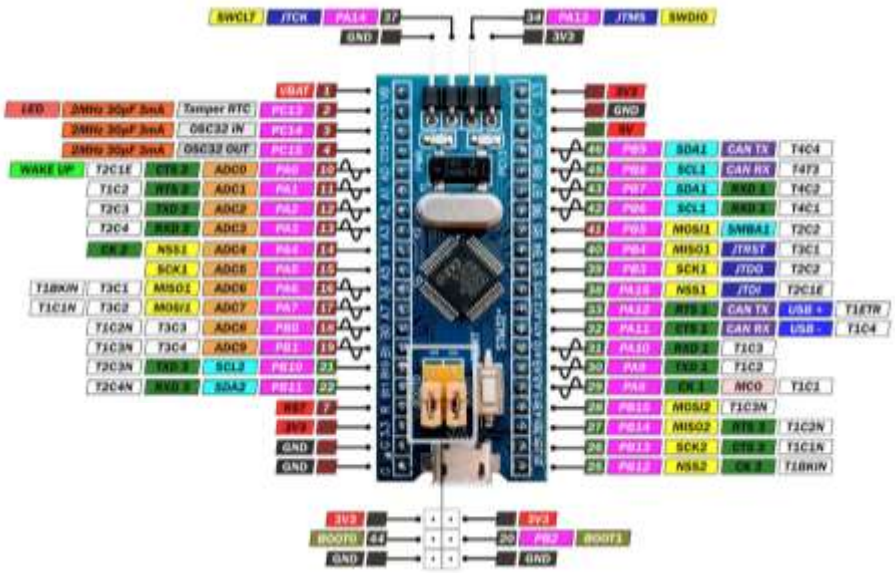
Nhiệm vụ là thiết kế một hệ thống có khả năng:

- Điều chỉnh tốc độ động cơ từ 0% đến 100% thông qua PWM.
- Điều chỉnh đảo chiều động cơ
- Sử dụng vi điều khiển STM32F103C8 và driver L298N.

2. LÝ THUYẾT

2.1. Tổng quan về vi điều kiển STM32F103C8 Blue Pill

Các ngõ vào/ra (I/O pins) của vi điều khiển STM32F103C8 Blue Pill



Hình 1: Các ngõ vào/ra (I/O pins) của vi điều khiển STM32F103C8

Module Blue Pill STM32F103C8T6 là một board phát triển dựa trên vi điều khiển STM32F103C8T6 của STMicroelectronics, có xung nhịp lên đến 72 MHz và bộ nhớ Flash 64KB. Thiết kế của board bao gồm nhiều chân GPIO và các module giao tiếp như UART, SPI, I2C, ADC, PWM, RTC, CAN, USB.

Nguyên tắc hoạt động của module này là nhận và xử lý tín hiệu từ các module giao tiếp. Sau đó, nó chuyển đổi tín hiệu thành dữ liệu số và xử lý theo

thuật toán được lập trình trên vi điều khiển. Kết quả xử lý có thể được điều khiển đến các tín hiệu đầu ra, ví dụ như hiển thị kết quả đo đạc trên màn hình LCD hoặc điều khiển các thiết bị như động cơ, đèn LED, cảm biến, v.v.

Module Blue Pill STM32F103C8T6 hỗ trợ lập trình bằng các công cụ như Keil C, IAR Embedded Workbench, hoặc STM32CubeIDE. Quá trình lập trình và tải chương trình có thể thực hiện thông qua giao tiếp SWD hoặc UART. Kit phát triển này được sử dụng rộng rãi trong nghiên cứu về ARM, có giá rẻ và có thể nạp bootloader Blue Pill để giao tiếp và lập trình một cách thuận lợi. Đặc điểm của kit bao gồm chất lượng gia công tốt và độ bền cao.

Thông số kỹ thuật

- Vi điều khiển chính: STM32F103C8T6.
- Điện áp hoạt động: 3.3VDC.
- Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho Vi điều khiển chính.
- Tích hợp sẵn thạch anh 8Mhz.
- Tích hợp sẵn thạnh anh 32Khz cho các ứng dụng RTC.
- Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,...
- Tích hợp Led trạng thái nguồn, Led PC13, Nút Reset.
- Kích thước: 53.34 x 15.24mm.
- Mạch nạp: có khá nhiều loại mạch nạp như: ULINK, J-LINK, CMSIS- DAP, STLINK… ở đây mình sử dụng Stlink vì giá thành khá rả và debug lỗi cũng tốt.

Nguồn cấp cho STM32F103C8

Điện áp 5VDC được cấp qua cổng USB-A qua mạch nạp ST-Link được chuyển thành 3.3V cấp cho Vi điều khiển chính

Bộ nhớ:

Bộ nhớ Flash nhúng, Có sẵn 64 hoặc 128 Kbyte Flash nhúng để lưu trữ các chương trình và dữ liệu.

Bộ nhớ RAM, 20 Kbyte SRAM nhúng được truy cập (đọc / ghi) ở tốc độ CPU Clock với 0 trạng thái chờ.

Loại chân	Tên các chân	Chi tiết
Nguồn	3.3V, 5V, GND	3.3V: Điện áp đầu ra được điều chỉnh từ bộ điều chỉnh trên bo mạch. 5V: Có thể sử dụng chân 5V từ USB hoặc bộ điều chỉnh để cấp nguồn cho ngoại vi. GND: Chân nối đất.
Analog Pins	PA0 – PA7 PB0 – PB1	Các chân đóng vai trò là ADC với độ phân giải 12
I/O pins	PA0 – PA15 PB0 – PB15 PC13 – PC15	37 chân GPIO
Serial	TX1, RX1 TX2, RX2 TX3, RX3	3 UART với các chân RTS và CTS
Ngắt ngoài	PA0 – PA15 PB0 – PB15 PC13 – PC15	Tất cả các chân kỹ thuật số đều có khả năng ngắt
PWM	PA0 – PA3 PA6 – PA10 PB0 – PB1 PB6 – PB9	Tổng cộng 15 chận PWM
SPI	MISO0, MOSI0, SCK0, CS0 MISO1, MOSI1, SCK1, CS1	2SPI
LED	PC13	LED tích hợp với chân PC13

Sơ lược về các chân của STM32

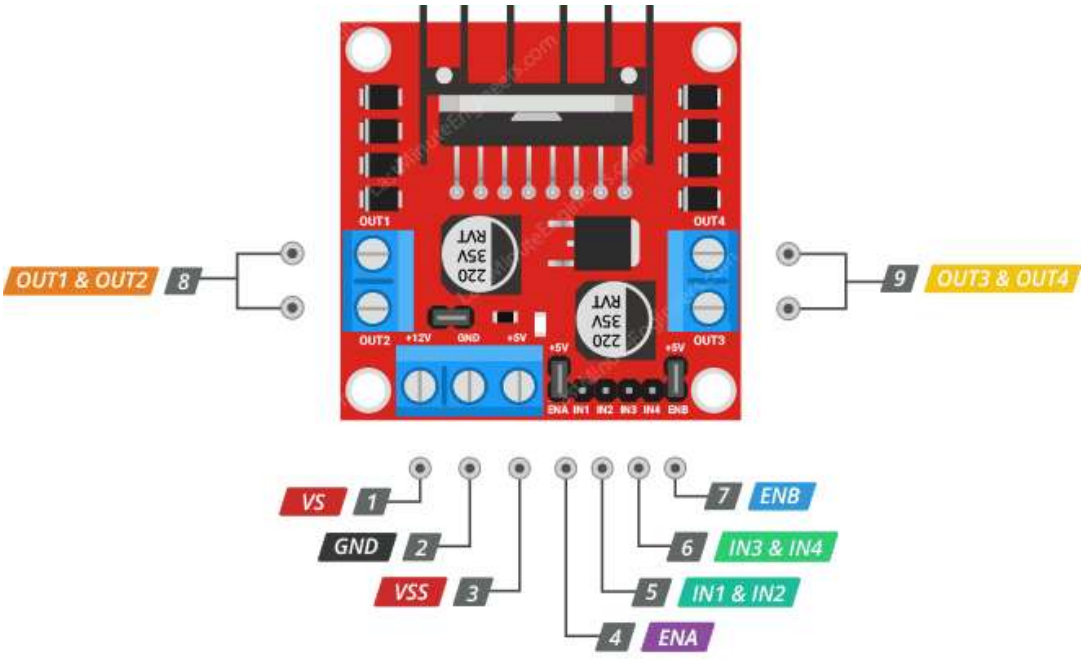
2.2. Tổng quan về chip điều khiển động cơ L298N

L298N là module điều khiển động cơ trong các xe DC và động cơ bước. Module có một IC điều khiển động cơ L298 và một bộ điều chỉnh điện áp 5V 78M05. Module L298N có thể điều khiển tối đa 4 động cơ DC hoặc 2 động cơ DC với khả năng điều khiển hướng và tốc độ.

Thông số kỹ thuật

- Module điều khiển: 2A L298N
- Chip điều khiển: Cặp H-Bridge L298N
- Điện áp cấp cho động cơ (Tối đa): 46V
- Dòng điện cấp động cơ (tối đa): 2A
- Điện áp logic: 5V
- Điện áp hoạt động của IC: 5-35V
- Dòng điện hoạt động IC: 2A
- Dòng logic: 0-36mA
- Công suất tối đa (W): 25W
- Cảm biến dòng điện cho mỗi động cơ
- Có tản nhiệt cho hiệu suất tốt hơn
- Có đèn báo LED bật nguồn

Chức năng



Hình 2: l298N

Pin điện

VS: Đầu vào điện áp cung cấp (lên đến 46V)

GND: Pin mặt đất

Pin điều khiển logic

In1, In2: Được sử dụng để điều khiển hướng quay của động cơ 1

In3, In4: Được sử dụng để điều khiển hướng quay của động cơ 2

Pin điều khiển động cơ:

OUT1, OUT2: Được sử dụng để điều khiển hướng của động cơ 1

OUT3, OUT4: Được sử dụng để điều khiển hướng của động cơ 2

ENA: Kích hoạt pin, được sử dụng để điều khiển tốc độ của động cơ 1

ENB: Kích hoạt pin, được sử dụng để điều khiển tốc độ của động cơ 2

Nguyên lý hoạt động

Cấu trúc: Gồm 4 khóa chuyển mạch điện tử (trong L298N là các cặp Transistor Darlington) được sắp xếp thành hình chữ H, với động cơ nằm ở giữa.

Hoạy động:

Khi khóa S1 và S4 đóng (dẫn điện): Dòng điện chạy từ nguồn dương → S1 →Động cơ → S4 → Mass. Động cơ quay theo chiều thuận.

Khi khóa S2 và S3 đóng: Dòng điện chạy từ nguồn dương →S3 →Động cơ → S2 → Mass. Dòng điện đảo chiều, động cơ quay nghịch.

Khi các khóa cùng phía (S1, S3) hoặc (S2, S4) cùng đóng hoặc cùng mở: Không có dòng điện chênh lệch qua động cơ → Động cơ dừng tự do hoặc hãm động năng.

2.3. Mạch nạp ST-LINK V2

2.3.1. Giới thiệu

Mạch nạp STM8, STM32 ST-Link V2 được sử dụng để nạp chương trình và debug cho Vi điều khiển STM32 và STM8 của ST, mạch nạp có kích thước nhỏ gọn, chi phí thấp, độ bền cao.

Hình 3:Mạch nạp ST-Link V2

Thông số kỹ thuật

- Mạch nạp STM8, STM32 ST-Link V2 Mini

- Chuẩn nạp: JTAG, SWD, SWV.

2.1.3.2. Nguyên lý hoạt động cơ bản của ST-Link V2:

Giao Tiếp và Kết Nối:

- ST-Link V2 kết nối với máy tính hoặc máy phát triển thông qua cổng USB.

- Khi được kết nối, nó trở thành một thiết bị nạp và debugger mà các phần mềm khát triển có thể tương tác.

Debug và Nạp Chương Trình:

- ST-Link V2 cho phép các phần mềm phát triển như STM32CubeIDE, Keil, hoặc các công cụ khác giao tiếp với vi điều khiển và thực hiện các thao tác debug và nạp chương trình.

- Các tính năng bao gồm đọc/ghi bộ nhớ, kiểm tra giá trị biến, theo dõi các

dòng lệnh và nhiều tính năng khác để giúp phát triển và sửa lỗi chương

trình.

Giao Tiếp JTAG/SWD:

- ST-Link V2 thường sử dụng giao thức JTAG (Joint Test Action Group) hoặc SWD (Serial Wire Debug) để giao tiếp với và kiểm soát vi điều khiển.

- JTAG được sử dụng cho việc debug và nạp chương trình, trong khi SWD thường được sử dụng cho các ứng dụng yêu cầu ít dây.

Nạp Chương Trình và Firmware:

- ST-Link V2 có khả năng nạp chương trình và firmware mới vào vi điều khiển.

- Nó cung cấp khả năng nạp chương trình thông qua giao diện người dùng của các công cụ phát triển hoặc thông qua lệnh từ dòng lệnh.

Nguồn Cung Cấp:

- ST-Link V2 cung cấp nguồn cung cấp cho vi điều khiển khi nạp chương trình hoặc khi chương trình đang chạy.

- Nguồn cung cấp có thể được chọn để đáp ứng yêu cầu cụ thể của ứng dụng.

Kích Thước Nhỏ Gọn:

- ST-Link V2 có kích thước nhỏ gọn, điều này giúp nó dễ dàng tích hợp vào các dự án phát triển và thử nghiệm.

Phần Mềm Hỗ Trợ:

- ST-Link V2 thường đi kèm với các phần mềm hỗ trợ từ STMicroelectronics, như ST-Link Utility, giúp thực hiện các thao tác cơ bản như đọc/ghi bộ nhớ và kiểm tra thiết bị.

- ST-Link V2 là một công cụ quan trọng trong quá trình phát triển và debug cho các ứng dụng sử dụng vi điều khiển STM32.

2.4. Động cơ DC

GA25-370 động cơ giảm tốc có Encoder 12VDC, 280 rpm

- Mode: GA25-370
- Tỉ số truyền: 1/21.3 cho động cơ 280RPM
- Điện áp cấp cho động cơ hoạt động: 12VDC
- Điện áp cấp cho Encoder hoạt động: 3.3VDC – 5VDC
- Dòng điện không tải: 100mA
- Công suất: 3W
- Tốc độ động cơ chưa giảm tốc: 6 – 2000RPM
- Tốc độ sau giảm tốc: 280RPM

- Đĩa Encoder 11 xung, hai kênh A-B.
- Loại trục: Chữ D
- Đường kính động cơ: 25mm
- Đường kính trục: 4mm
- Trọng lượng: 93g
- Kích thước: 50mm * 24mm

Hình 4: Động cơ DC 12V

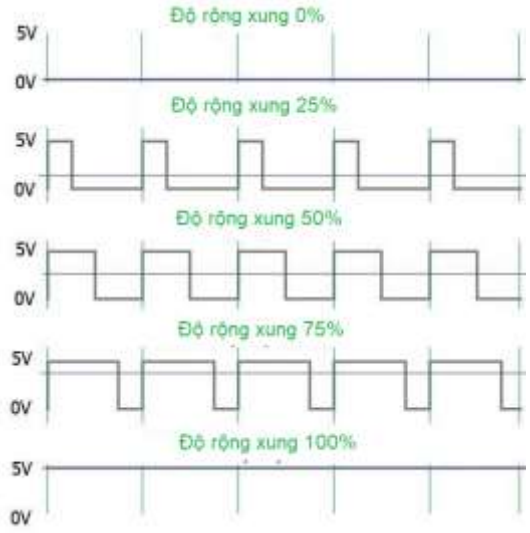
Nguyên lý hoạt động

Encoder hoạt động theo nguyên lý đĩa quay quanh trục. Trên đĩa mã hóa có các rãnh nhỏ để nguồn phát sáng chiếu tín hiệu quang qua đĩa. Chỗ có rãnh thì ánh sáng xuyên qua được, chỗ không có rãnh ánh sáng không xuyên qua được. Với các tín hiệu có, hoặc không có ánh sáng chiếu qua, người ta ghi nhận được đèn led có chiếu qua lỗ hay không. Số xung đếm được và tăng lên được tính bằng số lần ánh sáng bị cắt.

Cảm biến thu ánh sáng sẽ bật tắt liên tục để tạo ra các xung vuông. Việc sử dụng các bộ mã hóa sẽ ghi nhận lại số xung và tốc độ xung. Tín hiệu dạng xung sẽ được truyền về bộ xử lý trung tâm (vi xử lý, PLC,…) và từ đó kỹ sư cơ khí sẽ biết được vị trí và tốc độ của động cơ.

2.5. Nguyên lý điều khiển tốc độ động cơ bằng PWM

Phương pháp PWM là phương pháp dùng để điều chỉnh điện áp ra tải dựa trên sự thay đổi độ rộng và chuỗi xung vuông khiến điện áp thay đổi. Các PWM khi biến đổi sẽ có chung một tần số chỉ khác về độ rộng của sườn dương hoặc sườn âm.



Hình 5: Minh họa độ rộng xung PWM

Phương pháp PWM được sử dụng chính trong các ứng dụng điều khiển. Có thể bắt gặp ở các động cơ, các bộ xung áp và điều áp,… PWM giúp điều chỉnh độ nhanh chậm của động cơ, điều chỉnh sự ổn định của tốc độ động cơ

Phương pháp này hoạt động dựa trên nguyên lý ngắt – đóng nguồn điện cung cấp cho tải theo chu kỳ, trong đó tỉ lệ giữa thời gian bật và tắt sẽ quyết định điện áp trung bình ngõ ra. Trong đề tài này, việc thực hiện PWM được áp dụng thông qua Module L298N.

Hiện nay, có hai cách phổ biến để tạo ra tín hiệu PWM: bằng phần cứng và bằng phần mềm.

Bằng phần cứng: Tín hiệu PWM có thể được tạo ra bằng các mạch so sánh hoặc sử dụng trực tiếp các IC tạo dao động xung vuông như 555 hoặc LM556.

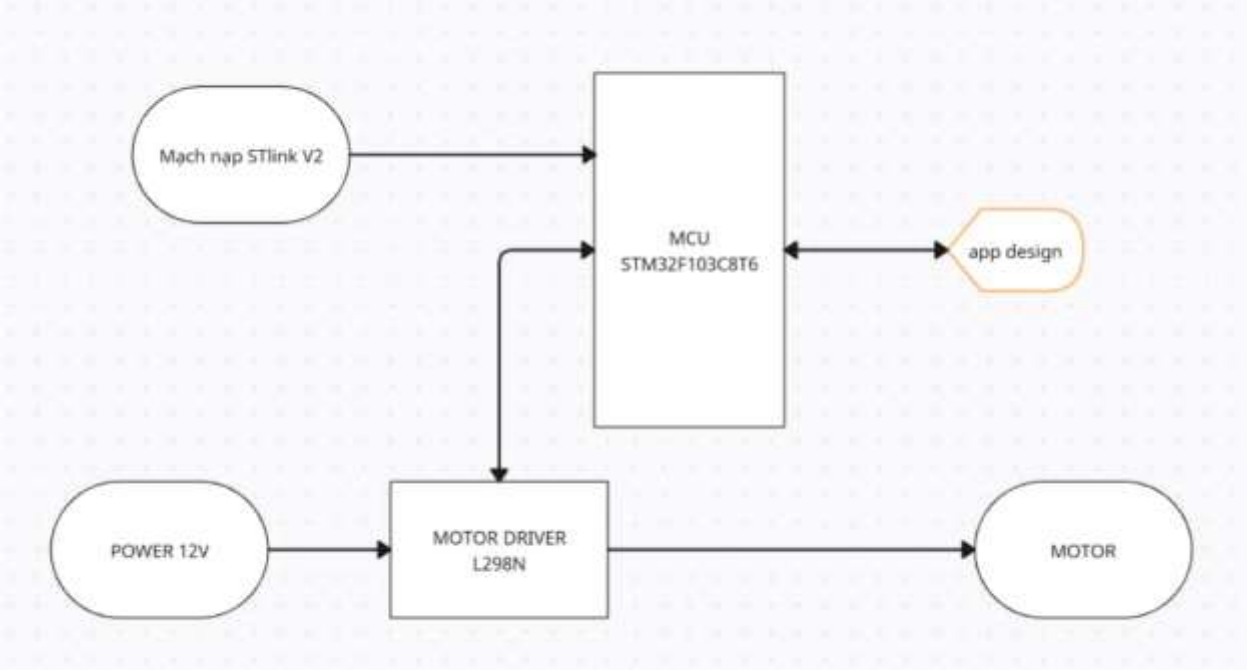
Bằng phần mềm: Sử dụng các vi điều khiển hoặc vi xử lý có thể lập trình được để sinh ra xung PWM theo yêu cầu

3. THIẾT KẾ VÀ THỰC HIỆN PHẦN CỨNG

3.1. Phân tích lựa chọn phương án

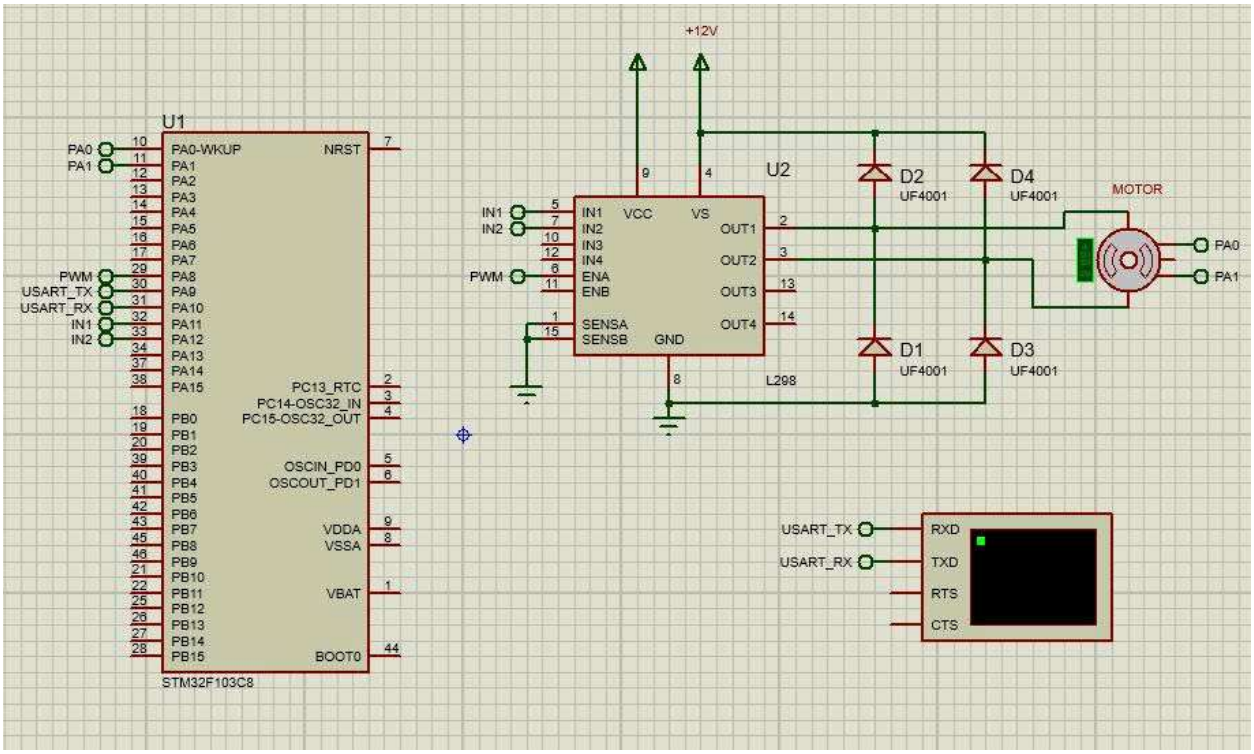
- **Khối động lực:** Nhóm chọn Module L298N vì tính phổ biến, giá thành rẻ và tích hợp sẵn diode bảo vệ, tản nhiệt.
- **Khối điều khiển:** Sử dụng STM32F103C8T6 vì có nhiều bộ Timer phần cứng, dễ dàng tạo PWM tần số cao giúp động cơ chạy mượt mà hơn so với các dòng vi điều khiển 8-bit cũ.

3.2. Sơ đồ khối



Hình 6: Sơ đồ khối

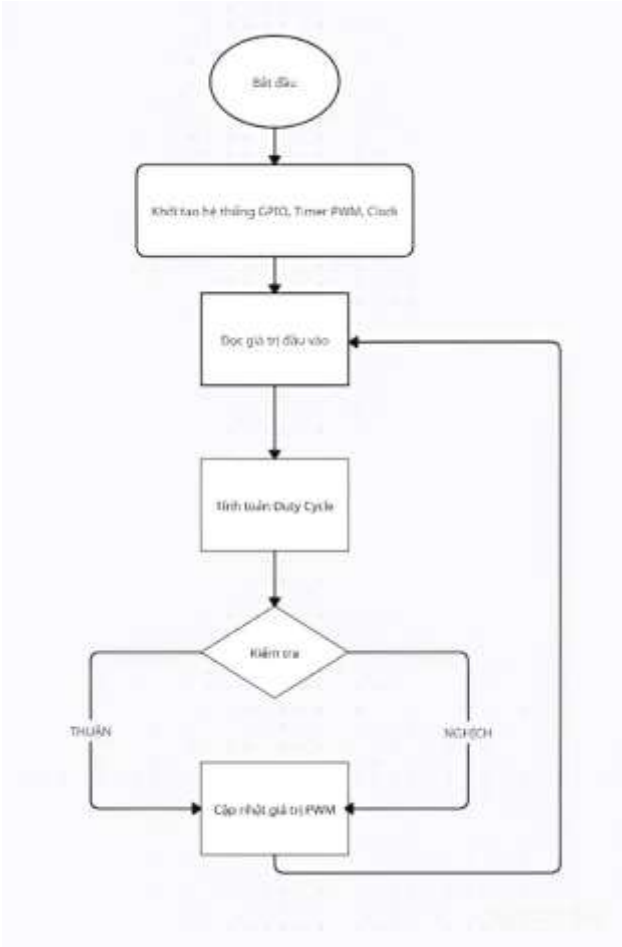
3.3. Sơ đồ nguyên lý



Hình 7: Sơ đồ nguyên lý

4. THIẾT KẾ VÀ THỰC HIỆN PHẦN MỀM

4.1. Lưu đồ giải thuật



Hình 8: Lưu đồ giải thuật

4.2. Tính toán

Tần số PWM (Timer 1)

$$f_{PWM} = \frac{f_{Clock}}{(B\text{ộ } chia \text{ tần} + 1) * (chu\ k\text{ì đếm} + 1)}$$

Với $f_{Clock} = 8MHz$

$B\text{ộ } chia = 8$

$B\text{ộ đếm} = 100$

$$\rightarrow f_{PWM} = \frac{8.10^6}{(8+1)*(100+1)} \approx 8{,}8KHz$$

Công thức tính Số xung trên 1 vòng quay (Encoder)

Thông số: Encoder gốc 11 xung, Mode đếm x4 (TI12), Tỉ số truyền 1/21.3.

Công thức: $\frac{xung}{vòng} = Xung\ gốc * 4 * Ti\ số\ truyền = 11 * 421.3 = 937.2$

$$tốc\ độ\ V = \frac{\Delta Pulse * 1000}{\Delta t * 937.2} (\frac{vòng}{s})$$

4.3. Code chương trình

Khai báo và định nghĩa biến:

```
#include "main.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#define FILTER_SIZE 8
volatile static float speed_buffer[FILTER_SIZE];
volatile static int buffer_index = 0;
volatile static float v_smooth = 0;
volatile uint8_t data_received_flag=0;
volatile uint8_t chieu1=0;
volatile uint8_t chieu2=0;
volatile static uint16_t pulsebefore = 0;
volatile static uint16_t pulsenow = 0;
volatile static float v;
char data_TX[20];
uint16_t pwmValue;
uint8_t data_RX[4]={0};
int a=0,b=1;
```

Vòng lặp bao gồm các công việc truyền giá trị vận tốc tính toán được lên winform, nhận giá trị pwm từ winform và đảo chiều động cơ:

```
while (1)
  {
   if (HAL_GetTick() - last_time_print > 100)
   {
    sprintf(data_TX, "%.2f\n", v_smooth);
    HAL_UART_Transmit_IT(&huart1,(uint8_t*)data_TX,strlen(data_TX));
    last_time_print = HAL_GetTick();
   }
   if (data_received_flag ==1)
   {
    data_received_flag=0;
    data_RX[3] = '\0';
    pwmValue=atoi((char*)data_RX);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,pwmValue );
   }
  if (chieu1 == 1)
   {
    chieu1 = 0;
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, 0);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, 1);
   }
   else if (chieu2 == 1)
   {
    chieu2 = 0;
```

```
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, 1);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, 0);
   }
  }
 }
```

Hàm callback nhận các dữ liệu từ UART:

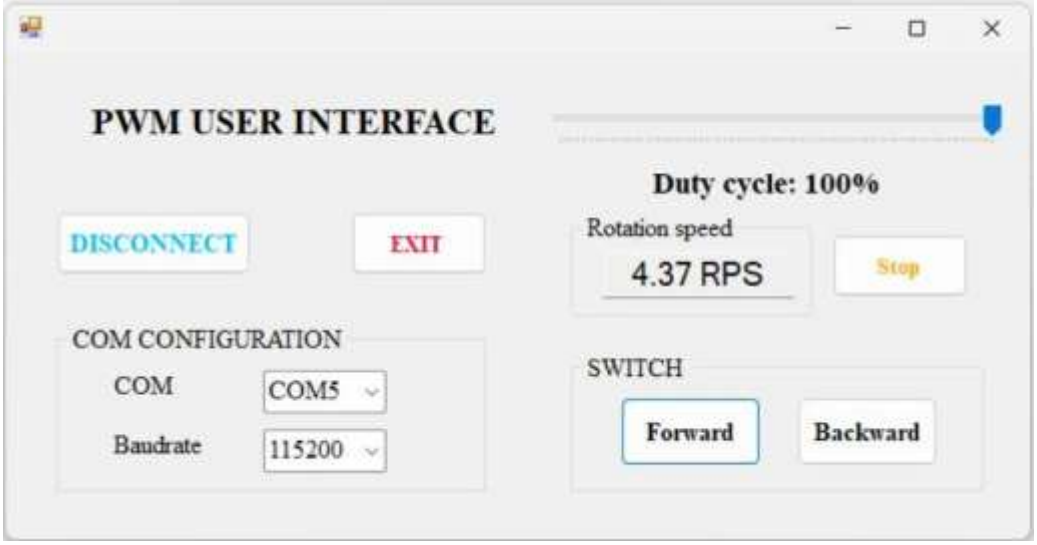
```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
 if(huart->Instance==USART1)
 {
  if(data_RX[0]=='T'&&data_RX[1]=='H'&&data_RX[2]=='N')//thuan=THN
  {
   chieu1=1; //thuan
   chieu2=0;
  }
   else if(data_RX[0] == 'N' && data_RX[1] == 'G' && data_RX[2] == 'H')//nghich=NGH
   {
    chieu2=1; // nghich
    chieu1=0;
   }
     else //data_RX<=100
     {
      data_received_flag=1;
     }
  }
```

```
    HAL_UART_Receive_DMA (&huart1, data_RX ,3);
}
```

Hàm callback tính toán vận tốc lấy từ 2 chân encoder:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
 if (htim == &htim2)
 {
  pulsenow = __HAL_TIM_GET_COUNTER(&htim3);
  int16_t delta_pulse = (int16_t)(pulsenow - pulsebefore);
  v = (float)delta_pulse * 100.0 / 937.2;
  speed_buffer[buffer_index] = v;
  buffer_index = (buffer_index + 1) % FILTER_SIZE;
  float sum = 0;
  for (int i = 0; i < FILTER_SIZE; i++){
  sum += speed_buffer[i];
  }
    v_smooth = sum / FILTER_SIZE;
    pulsebefore = pulsenow;
 }
}
```

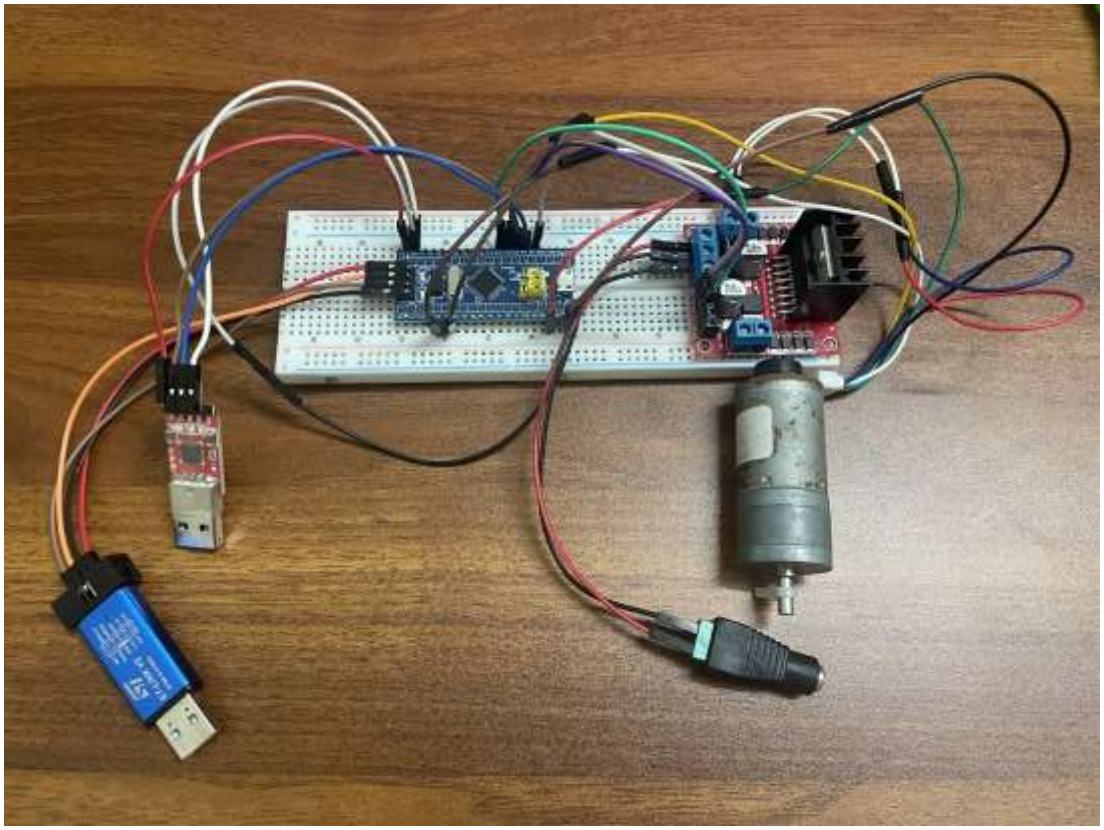
Giao diện điều khiển



Hình 9: Giao diện điều khiển

5. KẾT QUẢ THỰC HIỆN

5.1. Mạch thi công



Hình 7: Mạch thi công

5.2. Kết quả

Sau khi thi công, lắp ráp phần cứng và hoàn thiện lập trình nạp code, hệ thống đã được tiến hành thử nghiệm thực tế. Cho thấy hệ thống hoạt động ổn định, các chức năng điều khiển được đáp ứng. Kết quả khảo sát cho thấy hệ thống hoạt động đúng theo kịch bản thiết kế, có khả năng tùy chỉnh tốc độ chính xác cho động cơ.

6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Sau quá trình nghiên cứu, thiết kế và thi công thực tế, nhóm thực hiện đề tài đã đạt được những kết quả sau:

Về mặt lý thuyết: Đã nắm vững kiến thức về nguyên lý điều khiển động cơ DC sử dụng mạch cầu H, hiểu rõ cơ chế điều chế độ rộng xung (PWM) để thay đổi điện áp trung bình cấp cho tải. Đồng thời, nhóm cũng đã làm chủ được việc cấu hình các ngoại vi cơ bản trên vi điều khiển STM32 như GPIO, Timer và giao tiếp cơ bản.

Về mặt sản phẩm: Đã xây dựng thành công mô hình phần cứng hoàn chỉnh bao gồm khối điều khiển trung tâm (STM32), khối công suất (L298N) và khối chấp hành (Động cơ DC). Hệ thống hoạt động ổn định, đáp ứng đúng các yêu cầu đặt ra trong phần nhiệm vụ đề tài:

Điều chỉnh tốc độ động cơ mượt mà từ 0% đến 100% bằng phương pháp PWM.

Thay đổi được chiều quay động cơ

Giao diện điều khiển hoạt động chính xác.

Về kỹ năm mềm: Tích lũy thêm được kỹ năng làm việc nhóm, cách xử lý các vấn đề xảy ra, tính toán các chi phí một cách hợp lý..

Nhóm chúng em xin cảm ơn thầy. Thầy đã có những định hướng cụ thể kèm theo đó là những bài giảng trên giảng đường. Đã hỗ trợ nhóm chúng em trng suốt thời gian thực hiện đề tài này.

6.2. Hướng phát triển

Để hệ thống có thể hoàn thiện hơn, được ứng dụng một cách rộng rãi, nhóm đề xuất các hướng phát triển như sau:

- Mở rộng khả năng giám sát và điều khiển từ xa:

Giải pháp: Tích hợp thêm các module giao tiếp không dây như bluetooth (HC-05/HC-06) hoặc WiFi (ESP8266/ESP32).

Ứng dụng: Cho phép người dùng điều khiển tốc độ, chiều quay và giám sát các thông số (điện áp pin, dòng điện tiêu thụ, tốc độ thực tế) thông qua ứng dụng trên điện thoại thông minh hoặc giao diện Web, hướng tới mô hình IoT (Internet of Things).

-Tăng cường tính năng an toàn và bảo vệ hệ thống

Giải pháp:

Thêm mạch đo dòng điện (sử dụng điện trở Shunt hoặc cảm biến dòng ACS712) để phát hiện tình trạng kẹt cơ khí hoặc quá tải. Vi điều khiển sẽ tự động ngắt PWM khi dòng điện vượt ngưỡng an toàn.

Lập trình tính năng "Khởi động mềm" (Soft-start): Tăng dần Duty Cycle từ 0 lên giá trị đặt thay vì tăng đột ngột, giúp bảo vệ hộp số động cơ và tránh sụt áp nguồn đột ngột.

7. TÀI LIỆU THAM KHẢO

[1]. STM32F103C8T6 pdf, STM32F103C8T6 Description, STM32F103C8T6 Datasheet, STM32F103C8T6 view ::: ALLDATASHEET :::

[2].Datasheet - L298 - Dual full-bridge driver

[3].(PDF) DC Motor Speed Control Using PWM