ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

MÔN CƠ SỞ ĐO LƯỜNG VÀ ĐIỀU KHIỂN SỐ

Bài tập C3.1

Thích ứng tín hiệu và chuyển đổi tương tự - số

Thành viên thực hiện: Nhóm 3

22022118 Phạm Văn Duy

22022103 Ngô Đức Hiếu

22022175 Nguyễn Quốc Toản

22022145 Tạ Đình Kiên

Hà Nội, tháng 3 năm 2025

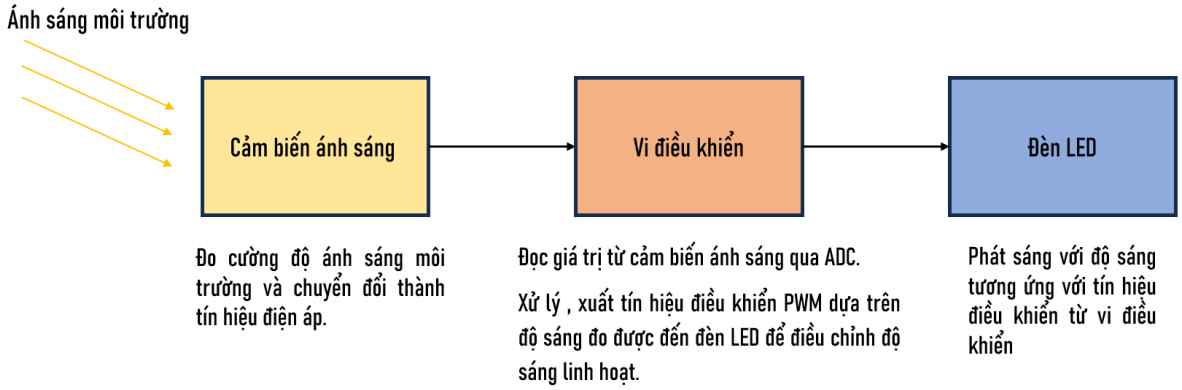
Mục lục

Bài 1: Thiết kế hệ thống điều khiển cường độ sáng của đèn thích nghi với ánh sáng môi trường?

1. Phân tích yêu cầu hệ thống

Hệ thống có chức năng điều chỉnh độ sáng của đèn LED dựa trên cường độ ánh sáng môi trường. Khi trời tối, đèn LED sẽ sáng mạnh hơn, và khi trời sáng, đèn sẽ giảm độ sáng hoặc tắt hẳn.

Sơ đồ mô tả hoạt động của hệ thống:



2. Đối tượng đo và điều khiển

Đối tượng: Ánh sáng môi trường.

Cảm biến: Module cảm biến ánh sáng sử dụng IC LM393 kết hợp với quang trở (LDR - Light Dependent Resistor).

Mô tả:

Quang trở (LDR) là một linh kiện có điện trở thay đổi theo cường độ ánh sáng. Khi ánh sáng mạnh, điện trở LDR giảm; khi ánh sáng yếu, điện trở tăng.

IC LM393 là bộ so sánh điện áp, nhưng trong trường hợp này, nó được cấu hình để xuất tín hiệu analog (thông qua mạch phân áp với LDR) vào chân PA0 của STM32F401RE để đọc bởi ADC.

3. Các tham số đặc trưng của đối tượng đo

Cường độ ánh sáng (Lux):

Đây là đại lượng vật lý đặc trưng cho mức độ sáng của môi trường, được cảm biến LDR gián tiếp đo thông qua sự thay đổi điện trở.

Phạm vi đo: Tùy thuộc vào đặc tính của LDR, thường từ 0 Lux (tối hoàn toàn) đến hàng nghìn Lux (ánh sáng mạnh như ánh nắng mặt trời).

Giá trị điện áp analog:

Điện áp từ mạch phân áp LDR (nối với LM393) thay đổi từ 0V (tối) đến khoảng 3.3V (sáng mạnh, vì STM32F401RE dùng nguồn 3.3V).

ADC của STM32 chuyển đổi điện áp này thành giá trị số từ 0 đến 4095 (12-bit).
Độ chia nhỏ nhất = 3.3V/4096≈0.0008057V (~0.806 mV).

Giá trị PWM điều khiển LED:

Được ánh xạ từ giá trị ADC (0-4095) sang duty cycle PWM (0-1000).

Logic: Ánh sáng yếu (ADC thấp) → PWM cao (LED sáng mạnh); ánh sáng mạnh (ADC cao) → PWM thấp (LED mờ).
Độ chia nhỏ nhất = 100%/1000=0.1% duty cycle.

Với nguồn 3.3V, độ chia nhỏ nhất về điện áp hiệu dụng là 3.3V/1000=0.0033V (3.3 mV)

Tần số PWM:

Được cấu hình là 1 kHz (TIM1->PSC = 83, TIM1->ARR = 1000), phù hợp để điều khiển LED mà không gây nhấp nháy nhìn thấy được.

Ngưỡng tối thiểu:

Code đặt ngưỡng: Nếu pwm_value1 < 100, PWM = 0 (LED tắt), để tránh LED sáng yếu không cần thiết khi ánh sáng môi trường đủ mạnh.

4. Phân tích các yếu tố ảnh hưởng đến hệ thống

a. Yếu tố từ đối tượng đo (ánh sáng môi trường)

Biến đổi cường độ ánh sáng:

Ảnh hưởng: Ánh sáng tự nhiên thay đổi liên tục (do mây, góc chiếu nắng), có thể gây dao động giá trị ADC, dẫn đến PWM thay đổi không ổn định.

Loại ánh sáng:

Ảnh hưởng: LDR nhạy khác nhau với ánh sáng tự nhiên (nắng) và nhân tạo (đèn huỳnh quang, LED). Điều này có thể làm sai lệch kết quả đo.

b. Yếu tố từ phần cứng

Độ nhạy của LDR:

Ảnh hưởng: LDR có phạm vi nhạy sáng hạn chế, không tuyến tính ở mức ánh sáng rất yếu hoặc rất mạnh.

Nguồn điện:

Ảnh hưởng: Nếu nguồn 3.3V không ổn định (do USB hoặc pin yếu), giá trị ADC sẽ dao động, ảnh hưởng đến PWM.

LED:

Ảnh hưởng: LED có đặc tính dòng-điện áp khác nhau, nên độ sáng thực tế không tuyến tính hoàn toàn với PWM.

c. Yếu tố từ phần mềm

Độ trễ (Delay):

Ảnh hưởng: Hàm Delay(100000) tạo độ trễ ~100ms bằng vòng lặp đơn giản, nhưng không chính xác do phụ thuộc tần số CPU và tối ưu hóa compiler.

Tần suất cập nhật:

Ảnh hưởng: Cập nhật PWM mỗi 100ms có thể chậm với thay đổi ánh sáng nhanh (như đèn nhấp nháy).

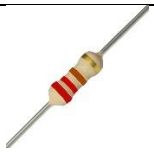
d. Yếu tố môi trường bên ngoài

Ảnh hưởng: Nếu LDR bị che khuất (bụi, vật cản), giá trị đo không phản ánh đúng ánh sáng môi trường.

5. Thiết kế và thực thi phần cứng

a. Yêu cầu phần cứng

	**Cảm biến ánh sáng có bộ thích ứng tín hiệu LM391** Số lượng: 1
	**Vi điều khiển STM32 NUCLEO F401RE** Số lượng: 1

	**Đèn LED** Số lượng: 1
	**Điện trở** Số lượng: 1
	**Dây nối** Số lượng: 5
	**Breadborad** Số lượng: 1

b. Thiết kế, thực thi phần cứng

Kết nối các module phần cứng:

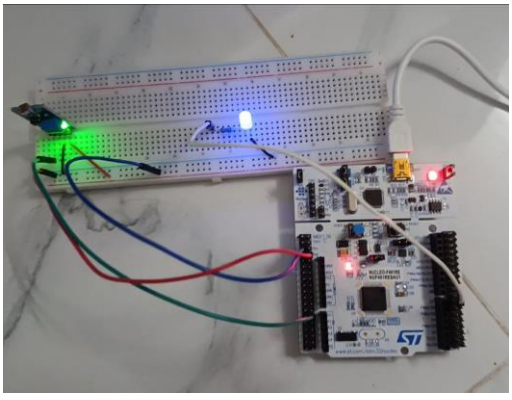
- Kết nối cảm biến

VCC	**5V**
GND	**GND**
A0	**PA0**

- Kết nối đèn LED

Anode	**PA8 + Điện trở**
Cathode	**GND**

Thực thi phần cứng:



6. Thực thi phần mềm

Mã nguồn:

```c
#include "stm32f4xx.h"
#include <stdint.h>
#include <main.h>
// Hàm cấu hình hệ thống clock (84 MHz với HSI)
void SystemClock_Config(void) {
    // Bật HSI (16 MHz)
    RCC->CR |= RCC_CR_HSION;
    while (!(RCC->CR & RCC_CR_HSIRDY));

    // Cấu hình PLL: 84 MHz (HSI * 336 / 8 / 2)
    RCC->PLLCFGR = (336 << 6) | (8 << 24) | RCC_PLLCFGR_PLLSRC_HSI;
    RCC->CR |= RCC_CR_PLLON;
    while (!(RCC->CR & RCC_CR_PLLRDY));

    // Chọn PLL làm nguồn clock hệ thống
    RCC->CFGR |= RCC_CFGR_SW_PLL;
    while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL);

    // Cấu hình bus: AHB = 84 MHz, APB1 = 42 MHz, APB2 = 84 MHz
    RCC->CFGR |= RCC_CFGR_HPRE_DIV1 | RCC_CFGR_PPRE1_DIV2 | RCC_CFGR_PPRE2_DIV1;
}
```

```c
// Hàm cấu hình GPIO
void GPIO_Init(void) {
    // Bật clock cho GPIOA
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;

    // PA0: Chân Analog cho ADC
    GPIOA->MODER |= (3 << 0); // Chế độ Analog (11)

    // PA8: Chân PWM (TIM1_CH1) - Alternate Function
    GPIOA->MODER |= (2 << 16); // Chế độ AF (10)
    GPIOA->AFR[1] |= (1 << 0); // AF1 (TIM1) cho PA8
}

// Hàm cấu hình ADC1 (PA0)
void ADC1_Init(void) {
    // Bật clock cho ADC1
    RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;

    // Cấu hình ADC: 12-bit, single conversion
    ADC1->CR1 = 0;                  // Độ phân giải 12-bit
    ADC1->CR2 = ADC_CR2_ADON;       // Bật ADC
    ADC1->SMPR2 = (3 << 0);         // Sampling time: 56 cycles cho kênh 0
    ADC1->SQR3 = 0;                 // Chọn kênh 0 (PA0)
}
```

```c
// Hàm đọc giá trị ADC
uint16_t ADC1_Read(void) {
    ADC1->CR2 |= ADC_CR2_SWSTART;   // Bắt đầu chuyển đổi
    while (!(ADC1->SR & ADC_SR_EOC)); // Đợi hoàn tất chuyển đổi
    return ADC1->DR;                // Trả về giá trị ADC (0-4095)
}

// Hàm cấu hình PWM (TIM1_CH1 trên PA8)
void TIM1_PWM_Init(void) {
    // Bật clock cho TIM1
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;

    // Cấu hình Timer: f = 1 kHz, PWM mode
    TIM1->PSC = 83;                 // Prescaler: 84MHz / (83+1) = 1MHz
    TIM1->ARR = 1000;               // Auto-reload: 1MHz / 1000 = 1kHz
    TIM1->CCMR1 |= (6 << 4);        // PWM mode 1 (110)
    TIM1->CCER |= TIM_CCER_CC1E;    // Bật kênh 1
    TIM1->BDTR |= TIM_BDTR_MOE;     // Bật đầu ra chính (Main Output Enable)
    TIM1->CR1 |= TIM_CR1_CEN;       // Bật Timer
}

// Hàm cập nhật PWM duty cycle
void Set_PWM_Duty(uint16_t duty) {
    if (duty > 1000) duty = 1000;   // Giới hạn duty cycle
    if (duty < 0) duty = 0;         // Giới hạn dưới
    TIM1->CCR1 = duty;              // Cập nhật giá trị PWM
}
```

```
188
189 // Hàm delay
190 void Delay(uint32_t count) {
191     for (volatile uint32_t i = 0; i < count; i++);
192 }
193
194 int main(void) {
195     uint16_t adc_value = 0;
196     uint16_t pwm_value = 0;
197     uint16_t pwm_value1 = 0;
198
199     // Khởi tạo hệ thống
200     SystemClock_Config();
201     GPIO_Init();
202     ADC1_Init();
203     TIM1_PWM_Init();
204
205     while (1) {
206         // Đọc giá trị từ ADC
207         adc_value = ADC1_Read();
208
209         // Ánh xạ giá trị ADC (0-4095) sang PWM (0-1000)
210         // ADC thấp (tối) -> PWM cao (LED sáng max)
211         // ADC cao (sáng) -> PWM thấp (LED mờ min)
212         pwm_value1 = 1000- (4095 - adc_value) * 1000 / 4095;
213         if(pwm_value1 < 100) pwm_value = 0;
214         else pwm_value = pwm_value1;
215         // Cập nhật PWM
216         Set_PWM_Duty(pwm_value);
217
218         // Delay khoảng 100ms
219         Delay(100000);
220     }
221 }
222
```

Giải thích chi tiết mã nguồn

Mã nguồn bao gồm các đoạn mã chính sau:

Cấu hình hệ thống clock

```
5  void SystemClock_Config(void) {
6      RCC->CR |= RCC_CR_HSION;
7      while (!(RCC->CR & RCC_CR_HSIRDY));
8
9      RCC->PLLCFGR = (336 << 6) | (8 << 24) | RCC_PLLCFGR_PLLSRC_HSI;
10     RCC->CR |= RCC_CR_PLLON;
11     while (!(RCC->CR & RCC_CR_PLLRDY));
12
13     RCC->CFGR |= RCC_CFGR_SW_PLL;
14     while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL);
15
16     RCC->CFGR |= RCC_CFGR_HPRE_DIV1 | RCC_CFGR_PPRE1_DIV2 | RCC_CFGR_PPRE2_DIV1;
17 }
```

- Bật HSI (High-Speed Internal) 16 MHz.
- Cấu hình PLL để đạt tốc độ 84 MHz.
- Chọn PLL làm nguồn clock hệ thống.
- Cấu hình tần số bus:
    - AHB: 84 MHz
    - APB1: 42 MHz
    - APB2: 84 MHz

Cấu hình GPIO

```
20 void GPIO_Init(void) {
21     RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;
22
23     GPIOA->MODER |= (3 << 0);
24
25     GPIOA->MODER |= (2 << 16);
26     GPIOA->AFR[1] |= (1 << 0);
27 }
```

- Bật clock cho GPIOA.
- PA0 được cấu hình ở chế độ Analog để sử dụng với ADC.
- PA8 được cấu hình ở chế độ Alternate Function để sử dụng PWM (TIM1_CH1).

Cấu hình ADC1 (PA0)

```
30 void ADC1_Init(void) {
31     RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;
32
33     ADC1->CR1 = 0;
34     ADC1->CR2 = ADC_CR2_ADON;
35     ADC1->SMPR2 = (3 << 0);
36     ADC1->SQR3 = 0;
37 }
```

- Bật clock cho ADC1.
- Cấu hình ADC với độ phân giải 12-bit.
- Sampling time 56 cycles.
- Chọn kênh ADC là kênh 0 (PA0).

Đọc giá trị ADC

```
40 uint16_t ADC1_Read(void) {
41     ADC1->CR2 |= ADC_CR2_SWSTART;
42     while (!(ADC1->SR & ADC_SR_EOC));
43     return ADC1->DR;
44 }
```

- Bắt đầu quá trình chuyển đổi ADC.
- Đợi đến khi kết quả sẵn sàng.
- Trả về giá trị ADC từ thanh ghi dữ liệu.

Cấu hình PWM trên TIM1_CH1 (PA8)

```c
void TIM1_PWM_Init(void) {
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;

    TIM1->PSC = 83;
    TIM1->ARR = 1000;
    TIM1->CCMR1 |= (6 << 4);
    TIM1->CCER |= TIM_CCER_CC1E;
    TIM1->BDTR |= TIM_BDTR_MOE;
    TIM1->CR1 |= TIM_CR1_CEN;
}
```

- Bật clock cho TIM1.
- Cấu hình Timer:
  - Prescaler: 84 MHz / 84 = 1 MHz.
  - Auto-reload: 1 MHz / 1000 = 1 kHz (Tần số PWM 1 kHz).
  - Chế độ PWM mode 1.
  - Kích hoạt đầu ra PWM trên kênh 1.

Cập nhật chu kỳ PWM

```c
void Set_PWM_Duty(uint16_t duty) {
    if (duty > 1000) duty = 1000;
    if (duty < 0) duty = 0;
    TIM1->CCR1 = duty;
}
```

- Giới hạn giá trị PWM từ 0 đến 1000.
- Cập nhật giá trị thanh ghi CCR1 của TIM1 để điều chỉnh độ rộng xung PWM.

Hàm delay tạo trễ

```c
void Delay(uint32_t count) {
    for (volatile uint32_t i = 0; i < count; i++);
}
```

Main

```c
int main(void) {
    uint16_t adc_value = 0;
    uint16_t pwm_value = 0;
    uint16_t pwm_value1 = 0;

    SystemClock_Config();
    GPIO_Init();
    ADC1_Init();
    TIM1_PWM_Init();

    while (1) {
        adc_value = ADC1_Read();

        pwm_value1 = 1000- (4095 - adc_value) * 1000 / 4095;
        if(pwm_value1 < 100) pwm_value = 0;
        else pwm_value = pwm_value1;

        Set_PWM_Duty(pwm_value);

        Delay(100000);
    }
}
```

- Đọc giá trị ADC từ PA0.

- Chuyển đổi giá trị ADC (0-4095) thành PWM (0-1000).
- Nếu giá trị PWM nhỏ hơn 100, tắt LED bằng cách đặt pwm_value = 0.
- Cập nhật tín hiệu PWM tương ứng với độ sáng LED.
- Thực hiện delay để cập nhật giá trị mới.

Video thử nghiệm hệ thống:

Bài 1_C3.1.mp4

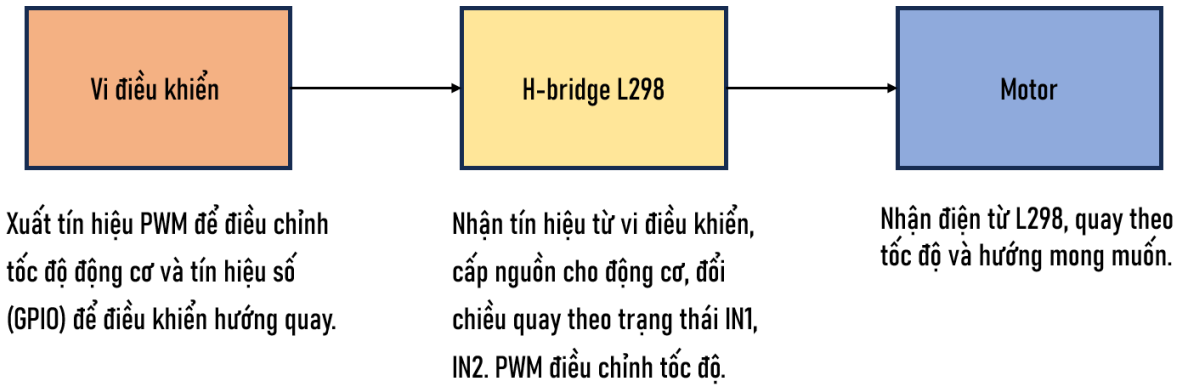
Đánh giá hiệu quả của hệ thống sau khi thực nghiệm:

- Đèn LED phản hồi nhanh khi thay đổi cường độ ánh sáng môi trường.
- Khi ánh sáng đạt 90% ngưỡng tối đa, đèn LED phải tắt hoàn toàn.
- Khi trời tối, đèn LED sáng tối đa mà không bị nhấp nháy.
- Độ sáng LED thay đổi thích hợp, không có hiện tượng thay đổi đột ngột gây khó chịu.

Bài 2: Sử dụng vi điều khiển kết nối với mạch H-bridge L298, động cơ DC và lập trình điều khiển tốc độ động cơ.

1. Phân tích yêu cầu hệ thống

Mạch dùng vi điều khiển để điều chỉnh tốc độ và hướng quay của động cơ thông qua mạch cầu H L298. Vi điều khiển xuất PWM để kiểm soát tốc độ và tín hiệu số để đổi chiều quay. L298 cấp nguồn phù hợp cho động cơ hoạt động.



2. Đối tượng điều khiển

Tốc độ động cơ: Được điều khiển thông qua tín hiệu PWM từ TIM2 trên chân PA5, kết nối với chân EN (Enable) của module L298N.

Trạng thái nút nhấn: Dùng để thay đổi tốc độ (PA0, PA1) và hướng quay (PC13). Nút nhấn trên PA0 (tăng tốc), PA1 (giảm tốc), và PC13 (đổi hướng qua ngắt EXTI).

3. Các tham số đặc trưng của đối tượng điều khiển

Tín hiệu PWM (Tốc độ động cơ):

Tần số: 1 kHz (TIM2->PSC = 83, TIM2->ARR = 1000).

Duty cycle: Giá trị PWM (pwm_value) thay đổi từ 0 (dừng) đến 1000 (tốc độ tối đa), tương ứng 0-100%.

Mặc định: 50% (500/1000), có thể tăng/giảm từng bước 100 đơn vị.

Độ phân giải: ARR=1000−1=999 ARR = 1000-1 = 999 ARR=1000−1=999 → 1000 mức (0-999).

Độ chia nhỏ nhất = 100%/1000=0.1% duty cycle.

Với nguồn 3.3V từ PA5, độ chia nhỏ nhất về điện áp hiệu dụng là 3.3V/1000=0.0033V (3.3 mV).

Tuy nhiên, code điều chỉnh PWM từng bước 100 đơn vị (10%), nên độ chia thực tế là 100/1000=10%(0.33mV với tín hiệu 3.3V).

4. Phân tích các yếu tố ảnh hưởng đến hệ thống

a. Yếu tố từ đối tượng đo (Nút nhấn và động cơ)

Tín hiệu nút nhấn:

Ảnh hưởng: Nút nhấn cơ học dễ bị dội (bouncing), gây ra nhiều lần đọc HIGH trong một lần nhấn, dẫn đến tăng/giảm PWM không chính xác.

Đặc tính động cơ:

Ảnh hưởng: Động cơ DC có quán tính, nên thay đổi tốc độ không tức thời khi PWM thay đổi, đặc biệt ở tốc độ thấp (<100).

b. Yếu tố từ phần cứng

Module L298N:

Ảnh hưởng: L298N có độ sụt áp (~1-2V), làm giảm điện áp thực tế cấp cho động cơ, ảnh hưởng đến tốc độ tối đa.

Nguồn điện:

Ảnh hưởng: Nguồn yếu hoặc không ổn định (ví dụ: pin cạn) làm giảm hiệu suất PWM và khả năng cấp dòng cho động cơ.

Pull-up/Pull-down:

Ảnh hưởng: PA0/PA1 dùng pull-down, PC13 dùng pull-up. Nếu nhiễu môi trường cao, tín hiệu có thể bị sai lệch.

c. Yếu tố từ phần mềm

Độ trễ (Delay):

Ảnh hưởng: Hàm Delay_ms dùng vòng lặp đơn giản, không chính xác tuyệt đối và làm CPU bận 100%, ảnh hưởng đến hiệu suất hệ thống.

Logic điều khiển tốc độ:

Ảnh hưởng: Tăng/giảm PWM từng bước 100 đơn vị có thể quá thô, không mượt mà ở một số ứng dụng.

Giải pháp: Giảm bước xuống (ví dụ: 10 đơn vị) và tăng tần suất kiểm tra nút nhấn.

Xử lý ngắt:

Ảnh hưởng: Ngắt EXTI trên PC13 chỉ đổi state, nhưng nếu nhấn nhanh liên tục, có thể bỏ sót ngắt do thiếu chống dội.

5. Thiết kế và thực thi phần cứng

a. Yêu cầu phần cứng

	**Vi điều khiển STM32 NUCLEO F401RE** Số lượng: 1
	**DC Motor Driver Dual H-Bridge L298N** Số lượng: 1
	**Động Cơ Encoder 334 Xung JGA25-371 + Giảm Tốc RP126 (1:34)** Số lượng: 1
	**Điện trở** Số lượng: 2

	**Button** Số lượng: 2
	**Dây nối** Số lượng: 11
	**Breadborad** Số lượng: 1

b. Thiết kế, thực thi phần cứng

Kết nối các module phần cứng:

- Kết nối STM32 với L298N

PA5	ENA
PA6	IN2
PA7	IN1
GND	GND
5V	5V

- Kết nối L98N với động cơ encoder

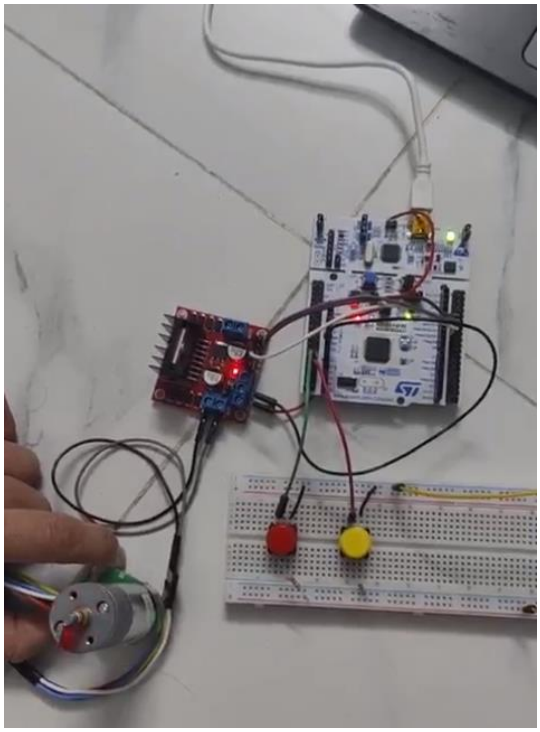
OUT1	M+
OUT2	M-

- Kết nối STM32 với button

PA0	Button + Điện Trở + 5V + GND

PA1	Button + Điện Trở + 5V + GND

Thực thi phần cứng:



6. Thực thi phần mềm

Mã nguồn:

```c
#include "stm32f4xx.h"

void TIM2_PWM_Init(void);
void GPIO_Init(void);
void EXTI_Init(void);
void Delay_ms(uint32_t ms);

volatile uint16_t pwm_value = 500;  // Mặc định 50% tốc độ
volatile int state = 0;  // Trạng thái điều khiển động cơ

int main(void) {
    GPIO_Init();
    TIM2_PWM_Init();
    EXTI_Init();

    while (1) {
        // Kiểm tra nút nhấn tăng tốc (PA0)
        if (GPIOA->IDR & (1 << 0)) {
            if (pwm_value < 1000) pwm_value += 100;  // Giới hạn max 1000
            TIM2->CCR1 = pwm_value;
            Delay_ms(200);  // Chống dội phím
        }

        if (GPIOA->IDR & (1 << 1)) {
                    if (pwm_value > 100) {
                        pwm_value -= 100;
                    } else {
                        pwm_value = 0; // Nếu pwm < 100, giảm tiếp sẽ dừng hẳn
                    }
                    TIM2->CCR1 = pwm_value;
                    Delay_ms(200);  // Chống dội phím
                }

        // Điều khiển hướng động cơ
        if (state == 0) {
            GPIOA->ODR |= (1 << 7);  // IN1 HIGH
            GPIOA->ODR &= ~(1 << 6); // IN2 LOW
        } else {
            GPIOA->ODR |= (1 << 6);  // IN1 HIGH
            GPIOA->ODR &= ~(1 << 7); // IN2 LOW
        }
    }
}

// ------------------------------
// ⚙ Khởi tạo GPIO
// ------------------------------
void GPIO_Init(void) {
    RCC->AHB1ENR |= (1 << 0); // Bật clock GPIOA
    RCC->AHB1ENR |= (1 << 2); // Bật clock GPIOC
```

```
51
52      // PA6, PA7 là output (Điều khiển L298N)
53      GPIOA->MODER |= (1 << (6 * 2)) | (1 << (7 * 2));
54      GPIOA->MODER &= ~((1 << (6 * 2 + 1)) | (1 << (7 * 2 + 1)));
55
56      GPIOA->OTYPER &= ~((1 << 6) | (1 << 7)); // Push-Pull
57      GPIOA->OSPEEDR |= (3 << (6 * 2)) | (3 << (7 * 2)); // High Speed
58
59      // PA0, PA1 là input (Nút nhấn)
60      GPIOA->MODER &= ~((3 << (0 * 2)) | (3 << (1 * 2)));
61
62      // Kéo xuống GND (Pull-down)
63      GPIOA->PUPDR |= (2 << (0 * 2)) | (2 << (1 * 2));
64
65      // PC13 là input với pull-up
66      GPIOC->MODER &= ~(3 << (13 * 2));  // Chế độ input (00)
67      GPIOC->PUPDR &= ~(3 << (13 * 2));  // Xóa cấu hình cũ
68      GPIOC->PUPDR |= (1 << (13 * 2));   // Kéo lên (Pull-up)
69  }
70
71  // -------------------------------
72  // ❁ Cấu hình EXTI (Ngắt ngoài) cho PC13
73  // -------------------------------
74  void EXTI_Init(void) {
75      RCC->APB2ENR |= (1 << 14);  // Bật clock SYSCFG
76
77      SYSCFG->EXTICR[3] |= (2 << 4);  // EXTI13 kết nối với PC13
78
79      EXTI->IMR |= (1 << 13);   // Bật ngắt EXTI13
80      EXTI->FTSR |= (1 << 13);  // Kích hoạt ngắt cạnh xuống (Falling edge)
81
82      NVIC_EnableIRQ(EXTI15_10_IRQn);  // Bật ngắt trong NVIC
83      NVIC_SetPriority(EXTI15_10_IRQn, 2);  // Đặt mức ưu tiên
84  }
85
86  // -------------------------------
87  // ❁ Trình xử lý ngắt EXTI (Ngắt PC13)
88  // -------------------------------
89  void EXTI15_10_IRQHandler(void) {
90      if (EXTI->PR & (1 << 13)) {  // Kiểm tra nếu PC13 gây ngắt
91          state = !state;  // Đổi trạng thái
92          EXTI->PR |= (1 << 13);  // Xóa cờ ngắt
93      }
94  }
95
96  // -------------------------------
97  // ❁ Cấu hình TIM2 PWM trên PA5
98  // -------------------------------
99  void TIM2_PWM_Init(void) {
100     RCC->APB1ENR |= (1 << 0); // Bật clock TIM2
101     RCC->AHB1ENR |= (1 << 0); // Bật clock GPIOA
102
103     // Cấu hình PA5 là AF1 (TIM2_CH1)
104     GPIOA->MODER |= (2 << (5 * 2));
105     GPIOA->AFR[0] |= (1 << (5 * 4)); // AF1 cho PA5
106
107     TIM2->PSC = 84 - 1;    // Chia tần số xuống 1MHz
108     TIM2->ARR = 1000 - 1;  // PWM tần số 1kHz
109     TIM2->CCR1 = pwm_value; // Mặc định duty cycle = 50%
110
111     TIM2->CCMR1 |= (6 << 4); // Chọn PWM mode 1 cho CH1
112     TIM2->CCER |= (1 << 0);  // Bật CH1 output
113     TIM2->CR1 |= (1 << 0);   // Bật TIM2
114 }
115
116 // -------------------------------
117 // ❁ Delay chống dội phím
118 // -------------------------------
119 void Delay_ms(uint32_t ms) {
120     for (uint32_t i = 0; i < ms * 4000; i++) {
121         __NOP();
122     }
123 }
```

Giải thích chi tiết mã nguồn

Mã nguồn bao gồm các đoạn chính sau:

(a) Biến toàn cục

```
volatile uint16_t pwm_value = 500;  // Mặc định 50% tốc độ
volatile int state = 0;  // Trạng thái điều khiển động cơ
```

- pwm_value : Điều chỉnh độ rộng xung PWM (0 - 1000).
- State: Xác định hướng quay của động cơ (0 hoặc 1).

(b) Hàm main

```
int main(void) {
    GPIO_Init();
    TIM2_PWM_Init();
    EXTI_Init();

    while (1) {
        // Kiểm tra nút nhấn tăng tốc (PA0)
        if (GPIOA->IDR & (1 << 0)) {
            if (pwm_value < 1000) pwm_value += 100;  // Giới hạn max 1000
            TIM2->CCR1 = pwm_value;
            Delay_ms(200);  // Chống dội phím
        }

        if (GPIOA->IDR & (1 << 1)) {
                    if (pwm_value > 100) {
                        pwm_value -= 100;
                    } else {
                        pwm_value = 0; // Nếu pwm < 100, giảm tiếp sẽ dừng hẳn
                    }
                    TIM2->CCR1 = pwm_value;
                    Delay_ms(200);  // Chống dội phím
                }

        // Điều khiển hướng động cơ
        if (state == 0) {
            GPIOA->ODR |= (1 << 7);  // IN1 HIGH
            GPIOA->ODR &= ~(1 << 6); // IN2 LOW
        } else {
            GPIOA->ODR |= (1 << 6);  // IN1 HIGH
            GPIOA->ODR &= ~(1 << 7); // IN2 LOW
        }
    }
}
```

Khởi tạo GPIO, PWM và ngắt ngoài (GPIO_Init(), TIM2_PWM_Init(), EXTI_Init()).

Vòng lặp vô hạn (while(1)).

- Kiểm tra nút nhấn:
    - Nếu nhấn PA0: Tăng pwm_value lên 100 (tối đa 1000).
    - Nếu nhấn PA1: Giảm pwm_value xuống 100 (không nhỏ hơn 0).
    - Cập nhật giá trị PWM (TIM2->CCR1 = pwm_value).

Điều khiển hướng quay động cơ dựa vào biến state:

- state == 0: Quay theo hướng 1 (IN1 HIGH, IN2 LOW).
- state != 0: Quay theo hướng ngược lại (IN1 LOW, IN2 HIGH)

(c) Khởi tạo GPIO

```
void GPIO_Init(void) {
    RCC->AHB1ENR |= (1 << 0); // Bật clock GPIOA
    RCC->AHB1ENR |= (1 << 2); // Bật clock GPIOC

    // PA6, PA7 là output (Điều khiển L298N)
    GPIOA->MODER |= (1 << (6 * 2)) | (1 << (7 * 2));
    GPIOA->MODER &= ~((1 << (6 * 2 + 1)) | (1 << (7 * 2 + 1)));

    GPIOA->OTYPER &= ~((1 << 6) | (1 << 7)); // Push-Pull
    GPIOA->OSPEEDR |= (3 << (6 * 2)) | (3 << (7 * 2)); // High Speed

    // PA0, PA1 là input (Nút nhấn)
    GPIOA->MODER &= ~((3 << (0 * 2)) | (3 << (1 * 2)));

    // Kéo xuống GND (Pull-down)
    GPIOA->PUPDR |= (2 << (0 * 2)) | (2 << (1 * 2));

    // PC13 là input với pull-up
    GPIOC->MODER &= ~(3 << (13 * 2));  // Chế độ input (00)
    GPIOC->PUPDR &= ~(3 << (13 * 2));  // Xóa cấu hình cũ
    GPIOC->PUPDR |= (1 << (13 * 2));   // Kéo lên (Pull-up)
}
```

· Bật clock cho GPIOA và GPIOC (cần thiết để sử dụng các chân GPIO).

· Cấu hình chân PA6, PA7 làm đầu ra (điều khiển module L298N):

- Đặt chế độ output.
- Kiểu xuất Push-Pull.
- Tốc độ cao.

· Cấu hình PA0, PA1 làm đầu vào (nút nhấn):

- Đặt chế độ input.
- Kéo xuống GND (Pull-down) để đảm bảo trạng thái ổn định khi không nhấn.

· Cấu hình PC13 làm đầu vào với pull-up (nút nhấn hoặc cảm biến):

- Đặt chế độ input.
- Xóa cấu hình cũ.
- Kéo lên mức cao (Pull-up) để giữ trạng thái mặc định là HIGH.

(d) Cấu hình EXTI (Ngắt ngoài)

```
void EXTI_Init(void) {
    RCC->APB2ENR |= (1 << 14);  // Bật clock SYSCFG

    SYSCFG->EXTICR[3] |= (2 << 4);  // EXTI13 kết nối với PC13

    EXTI->IMR |= (1 << 13);   // Bật ngắt EXTI13
    EXTI->FTSR |= (1 << 13);  // Kích hoạt ngắt cạnh xuống (Falling edge)

    NVIC_EnableIRQ(EXTI15_10_IRQn);  // Bật ngắt trong NVIC
    NVIC_SetPriority(EXTI15_10_IRQn, 2);  // Đặt mức ưu tiên
}
```

Cấu hình ngắt ngoài (External Interrupt - EXTI) trên chân PC13 của vi điều khiển STM32. Cụ thể:

- Kích hoạt clock cho SYSCFG để có thể cấu hình ngắt ngoài.
- Kết nối EXTI13 với chân PC13, cho phép chân này kích hoạt ngắt.
- Cấu hình EXTI13:
    - Cho phép ngắt từ EXTI13.
    - Kích hoạt ngắt khi có tín hiệu cạnh xuống (Falling edge), tức là khi mức điện áp chuyển từ cao xuống thấp (thường dùng cho nút nhấn).
- Cấu hình bộ điều khiển ngắt (NVIC):
    - Bật ngắt EXTI13 trong NVIC.
    - Đặt mức ưu tiên ngắt.

(e) Trình xử lý ngắt EXTI

```
void EXTI15_10_IRQHandler(void) {
    if (EXTI->PR & (1 << 13)) {  // Kiểm tra nếu PC13 gây ngắt
        state = !state;  // Đổi trạng thái
        EXTI->PR |= (1 << 13);  // Xóa cờ ngắt
    }
}

```

Khi có tín hiệu giảm từ mức cao xuống mức thấp (Falling edge) trên chân PC13, chương trình sẽ thực hiện các bước sau:

- Kiểm tra xem ngắt có thực sự được kích hoạt từ chân PC13 hay không.
- Thay đổi trạng thái của biến state (ví dụ: nếu state đang là 0 thì đổi thành 1, và ngược lại). Điều này có thể dùng để bật/tắt đèn LED hoặc thực hiện một hành động nào đó.
- Xóa cờ ngắt để hệ thống sẵn sàng nhận lần ngắt tiếp theo.

(f) Khởi tạo TIM2 PWM

```
void TIM2_PWM_Init(void) {
    RCC->APB1ENR |= (1 << 0); // Bật clock TIM2
    RCC->AHB1ENR |= (1 << 0); // Bật clock GPIOA

    // Cấu hình PA5 là AF1 (TIM2_CH1)
    GPIOA->MODER |= (2 << (5 * 2));
    GPIOA->AFR[0] |= (1 << (5 * 4)); // AF1 cho PA5

    TIM2->PSC = 84 - 1;     // Chia tần số xuống 1MHz
    TIM2->ARR = 1000 - 1;   // PWM tần số 1kHz
    TIM2->CCR1 = pwm_value; // Mặc định duty cycle = 50%

    TIM2->CCMR1 |= (6 << 4); // Chọn PWM mode 1 cho CH1
    TIM2->CCER |= (1 << 0);  // Bật CH1 output
    TIM2->CR1 |= (1 << 0);   // Bật TIM2
}
```

Khởi tạo PWM (Pulse Width Modulation) trên kênh 1 của Timer 2 (TIM2_CH1) tại chân PA5 trên vi điều khiển STM32.

- Bật clock cho TIM2 và GPIOA.
- Cấu hình chân PA5 làm đầu ra PWM (Alternate Function AF1).
- Thiết lập thông số PWM:
  - Tần số PWM là 1 kHz.
  - Độ rộng xung (duty cycle) mặc định là 50%.
- Kích hoạt chế độ PWM mode 1.
- Bật kênh đầu ra và khởi động Timer 2 để tạo xung PWM.

(g) Hàm delay chống dội phím

```
void Delay_ms(uint32_t ms) {
    for (uint32_t i = 0; i < ms * 4000; i++) {
        __NOP();
    }
}
```

- Delay đơn giản bằng vòng lặp.

Video thử nghiệm hệ thống

Bài 2_C3.1.mp4

Đánh giá hiệu quả của hệ thống sau khi thực nghiệm:

- Động cơ phản hồi nhanh khi nhấn nút điều khiển.
- Khi PWM = 0, động cơ dừng hoàn toàn.
- Khi thay đổi hướng quay, động cơ không bị giật hoặc ngừng đột ngột.
- Tốc độ động cơ thay đổi tuyến tính theo tín hiệu PWM để đảm bảo hoạt động ổn định.