

3. 제어 시스템의 모델링

3.9 실험 실습: Lab 3

3.9.1 모터 제어를 위한 엔코더와 파워 앰프

이 실험에서는 DC 모터 제어에 필요한 엔코더를 마이크로컨트롤러를 이용해서 읽는 실험을 실시한다. 제어 실험에 사용할 DC 모터는 비교적 저렴한 가격에 쉽게 구입할 수 있다. 이 책의 실험에서 사용할 DC 모터는 다음 그림과 같으며 아래의 링크에서 구입할 수 있다.



그림 3-67 실험에 사용할 DC 모터

https://www.motorbank.kr/goods/goods_view.php?goodsNo=1000008065

이 모터에는 광학식 엔코더가 장착되어 있으며 한 회전 당 96 펄스가 출력된다. 쿼드러처 방식의 엔코더 이므로 한 회전 당 읽을 수 있는 정밀도(resolution)는 $4 \times 96 = 384$ 이다.

이 책에서 사용하는 STM32F429 마이크로컨트롤러에 내장된 타이머/카운터에는 엔코더 신호를 읽을 수 있는 기능이 있다. 만약 사용하는 마이크로컨트롤러에 엔코더 신호를 읽는 기능이 없는 경우에는 인터럽트를 사용한 프로그램으로 엔코더 신호를 읽는 기능의 구현이 가능하다. 이 실습에서는 실습용 보드의 타이머/카운터의 엔코더 신호 읽는 기능을 이용해서 모터에 부착된 쿼드러처 엔코더를 읽는 프로그램을 작성한다.

엔코더를 사용할 때는 마이크로컨트롤러의 전압과 호환이 되는지 확인할 필요가 있다. 엔코더와 마이크로컨트롤러의 연결 방법은 비교적 간단하다. 접지(GND)와 전원 전압(통상 5 볼트이지만, 3 볼트에서 동작하는 경우도 많음)을 연결하고 쿼드러처 엔코더의 2 개 채널 신호가 출력되는 전선을 아래에서 설명하는 타이머/카운터의 엔코더 신호 입력 핀에 연결한다. 전원 연결시에 주의할 점은 모터에 연결하는 전원선과 엔코더에 공급하는 전원선은 서로 관련이 없으므로, 모터에 공급하는 전압과는 별도로 엔코더에 전압이 공급되어야 한다. 일반적으로는 마이크로컨트롤러에 공급하는 전압을 그대로 연결한다. 이 책에서 사용하는 실습용 보드에서는 마이크로컨트롤러에 공급하는 전압이 3 볼트이므로 이 전압을 엔코더 공급 전압으로 사용할 수

있다. 일반적으로 엔코더도 3 볼트에서 동작하는 경우가 많지만, 만약 엔코더에 5 볼트의 전압을 공급할 경우에는 반드시 엔코더에서 출력되는 신호의 전압을 3 볼트로 낮추어서 마이크로컨트롤러의 입력에 연결해야 한다. 3 볼트에서 동작하는 마이크로컨트롤러에 5 볼트에서 동작하는 장치를 연결할 경우 마이크로컨트롤러가 손상될 수 있으므로 주의가 필요하다.

먼저 STM32CubeIDE 에서 새로운 프로젝트를 만들어서 프로젝트 이름을 Encoder 로 정한다. 이 프로젝트에서는 쿼드러처 엔코더 값을 읽어서 시리얼 터미널에 프린트하는 프로그램을 작성한다. 시리얼 터미널을 사용하므로 Lab1 과 같이 USART1 활성화를 확인한다. 이 프로젝트에서는 Timer 2 를 사용해서 엔코더 신호를 읽는다. 그림 3-68 과 같이 Pinout & Configuration 에서 TIM2 를 선택한 후, Encoder Mode 를 선택한다. TIM2 는 32 비트 타이머/카운터이다.

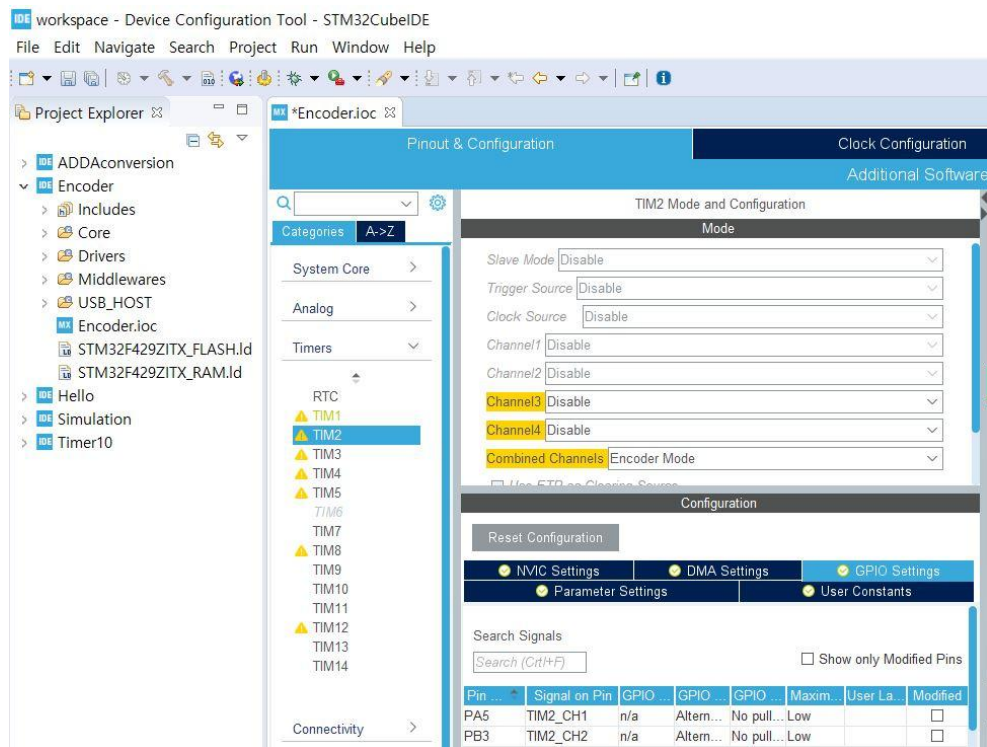


그림 3-68 Timer 2 설정

쿼드러처 엔코더 신호를 읽기 위해서는 2 개의 입력 핀이 필요하며, 그림 3-68 에서 볼 수 있듯이 PA5 와 PB3 가 입력으로 배정되어 있다. 그러나 PA5 는 D/A 컨버터 출력으로 사용해야 하므로 사용할 수 없다. 따라서 PA5 대신 다른 핀을 사용해야 하므로 다른 핀으로 변경할 필요가 있다. 그림 3-69 와 같이 Pinout view 화면에서 PA15 핀을 찾아서 TIM2_CH1 으로 변경한다.

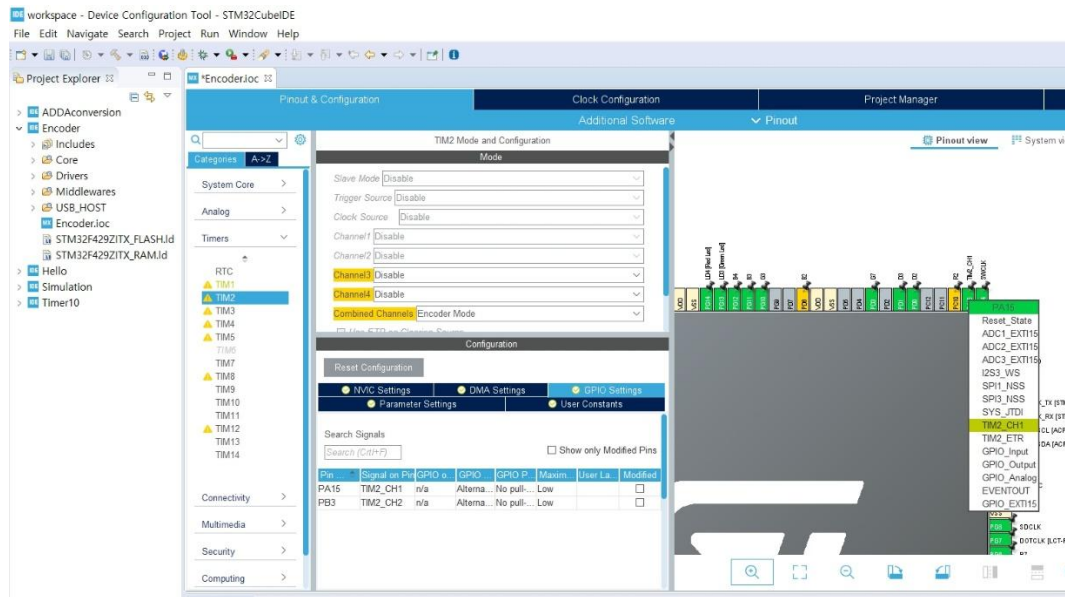


그림 3-69 PA15 로 변경

그림 3-70 과 같이 TIM2_CH1 이 PA15 로 변경된 것을 확인한다.

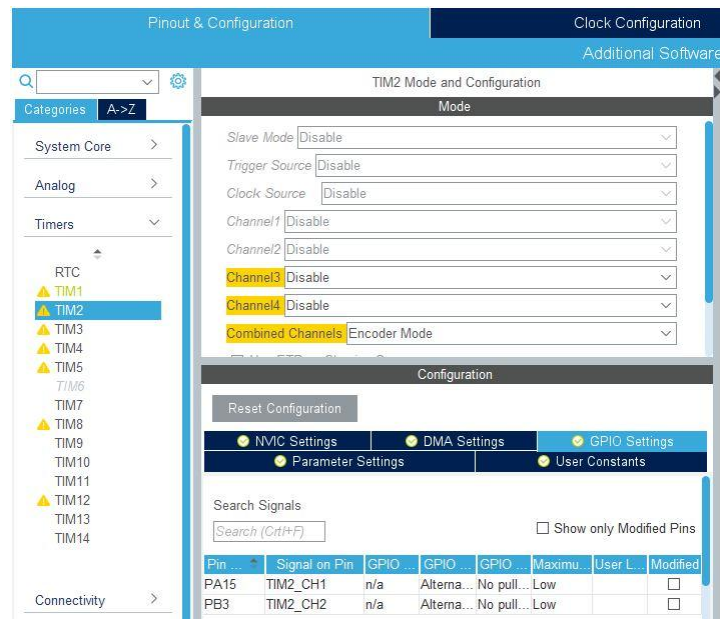


그림 3-70 TIM2 GPIO 설정

다음으로 그림 3-71 과 같이 Counter Period 를 32 비트 레지스터의 최대 값인 0xFFFFFFFF 으로 변경한다. (2 의 보수 숫자를 사용하기 위해서는 32 비트의 최대 값으로 설정할 필요가 있음. 32 비트의 최대 값을 사용하지 않을 경우 가장 높은 자리의 숫자가 항상 0 이므로 음수의 숫자를 나타낼 수 없음.) 또한 그림 3-71 과 같이 쿼드러쳐 엔코더 입력 기능을 활성화 하기 위해서 Encoder Mode 에서 Encoder Mode TI1 and TI2 으로 설정한다.

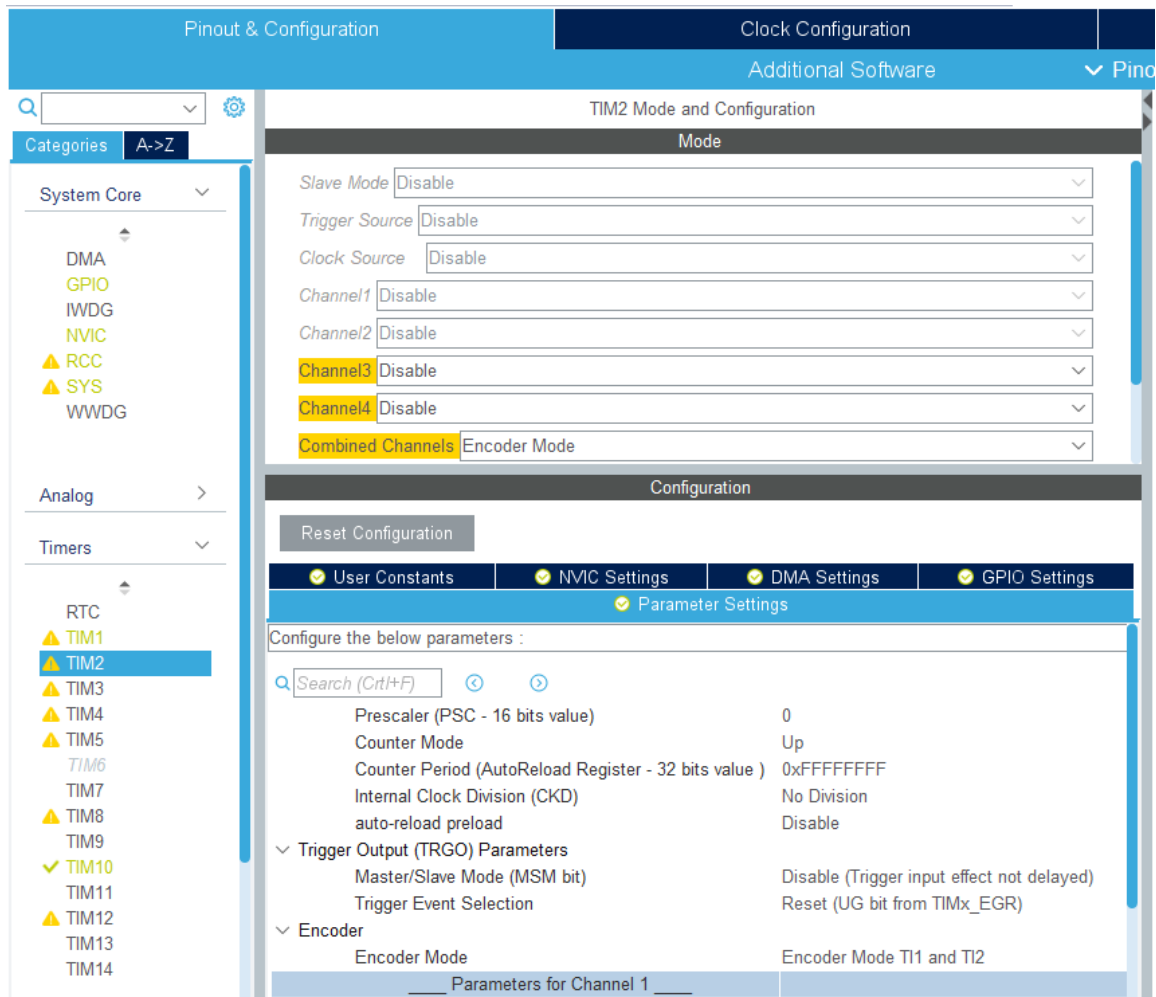


그림 3-71 Counter Period 설정과 Encoder Mode 설정

코드 생성을 완료한 후 다음과 같은 코드를 입력한다.

코드 3.1

```
/* USER CODE BEGIN Includes */
#include "stdio.h"
/* USER CODE END Includes */
```

코드 3.2

```
/* USER CODE BEGIN 0 */
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
PUTCHAR_PROTOTYPE
{
    HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
/* USER CODE END 0 */
```

코드 3.3

```
/* USER CODE BEGIN 2 */
  HAL_TIM_Encoder_Start(&htim2,TIM_CHANNEL_1);
/* USER CODE END 2 */
```

코드 3.4

```
/* USER CODE BEGIN WHILE */
while (1)
{
    printf("%ld\r\n",TIM2->CNT);
    HAL_Delay(100);
/* USER CODE END WHILE */
```

코드 3.4 에서 HAL_Delay(100)는 100msec 의 딜레이 함수이다. 프로그램을 실행하기 전에 2 갭-엔코더 출력 신호를 연결하고 전원과 접지를 연결한다. 또한 시리얼 터미널 프로그램을 열어서 프린트 출력을 볼 수 있도록 준비한다. 모든 준비가 끝나고 프로그램을 실행한 후 모터의 축을 손으로 돌리면서 엔코더 값의 변화를 관찰한다. 특히 회전 방향을 바꾸면서 음의 숫자 출력도 확인한다. 또한 모터 축의 한바퀴 회전에 해당하는 엔코더 출력 값을 확인한다.

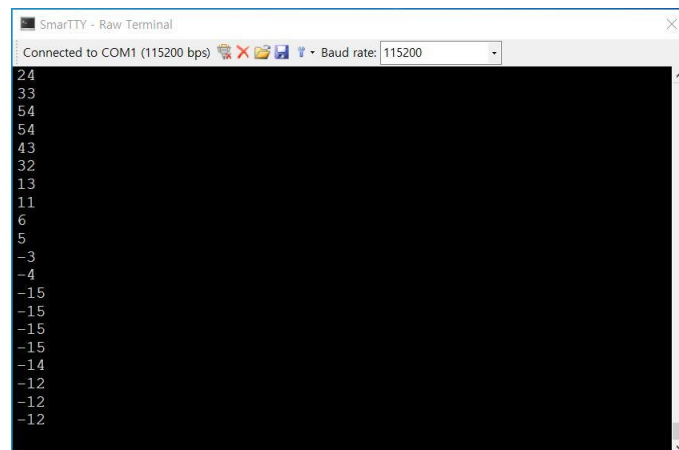
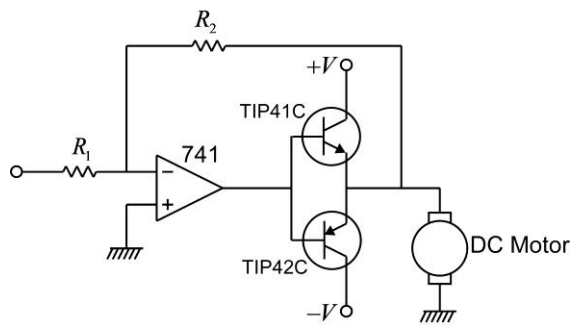


그림 3-72 시리얼 터미널에 출력되는 엔코더 출력 값

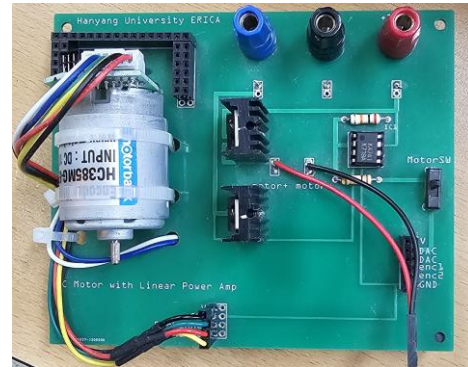
■ 모터 구동을 위한 파워 앰프

DC 모터를 구동하기 위해서는 파워 앰프 회로가 필요하다. D/A 컨버터에서 출력되는 신호는 모터를 구동하기에 출력이 약하므로 증폭할 수 있는 앰프 회로가 필요하다. 통상적으로 모터 구동에 사용되는 앰프 회로는 PWM 앰프와 선형 앰프이며 각각의 장단점이 있다. 선형 앰프를 이용해서 모터를 구동하기 위해서는 D/A 컨버터가 필요하다. 실습에 사용하는 마이크로컨트롤러에는 D/A 컨버터가 있으므로 선형 앰프를 사용해서 모터를 구동하기로 한다. D/A 컨버터가 없는 마이크로컨트롤러를 이용해서 모터를 제어할 경우에는 일반적으로 PWM 앰프를 이용한다. 작은 DC 모터를 구동하기 위한 선형 파워 앰프는 그림 3-73 과 같이 OP 앰프와

2 개의 파워 트랜지스터를 이용해서 간단하게 구성할 수 있다. 이 회로는 간단한 회로이므로 브레드보드를 이용해서 구성할 수 있지만, 안정된 동작을 위해서는 그림 3-73(b)와 같이 PCB 를 제작해서 구성하는 것을 권장한다. 파워 트랜지스터에서는 열이 발생할 수 있으므로 그림 3-73(b)과 같이 방열판과 함께 사용하는 것을 권장한다.



(a)



(b)

그림 3-73 선형 파워 앰프

실습 연습문제 3.1

그림 3-73 과 같은 선형 앰프 회로를 구성하고 모터를 연결한 후, DAC2 를 사용할 수 있도록 설정해서 D/A 출력을 선형 앰프 회로의 입력에 연결한다. 마이크로컨트롤러에서 수식을 계산할 때 2 의 보수로 나타낸 수치로 계산을 한다. 이 수치가 D/A 컨버터를 통해서 모터까지 전달 되는 과정은 그림 3-74 와 같다. 2 의 보수를 이용해서 수행한 수식 계산 결과의 값에 2048 을 더한 후, 전압 크기 변환 회로를 거쳐서 파워 앰프에 입력이 된다.

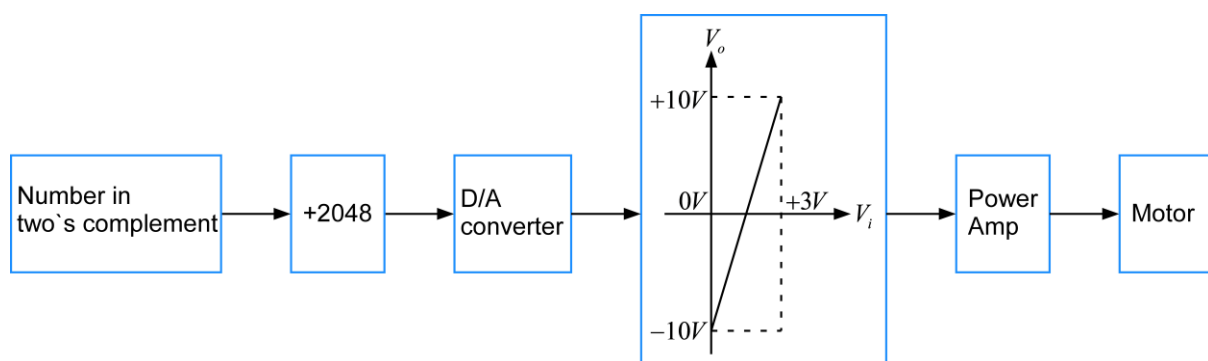


그림 3-74 모터 제어 신호 출력 과정

그림 3-74 에서 2 의 보수로 나타낸 숫자가 0 이라면 D/A 변환기에는 2048 이 입력이 되고 D/A 출력 핀에서 출력되는 전압은 1.5 V 이다. 전압 크기 변환 회로를 거치면 파워 앰프에 입력되는 전압은 0 V 이다. 또한 만약 2 의 보수로 나타낸 숫자가 1024 라면 D/A 변환기에는

2048+1024=3072 가 입력되고 D/A 출력 핀 전압은 $1.5+0.75=2.25$ V 이며 파워 앰프에는 5 V의 전압이 입력된다. 2의 보수로 나타낸 숫자가 -1024 라면 D/A 변환기에는 $2048-1024=1024$ 가 입력되고 D/A 출력 핀 전압은 $1.5-0.75=0.75$ V 이며 파워 앰프에는 -5 V의 전압이 입력된다. 2의 보수로 나타낸 숫자가 양(+)의 값인 경우와 음(-)의 값인 경우에 모터의 회전 방향이 서로 반대 방향인지 관찰한다. 또한 전압의 크기에 따라서 모터의 속도가 변화하는지 관찰한다. 특히 2의 보수로 나타낸 숫자가 양(+)의 숫자인 경우에 엔코더의 값이 증가하는지 관찰한다. 만약 그렇지 않다면 엔코더 입력의 2개 채널 연결을 서로 바꾸어서 양의 숫자에 대해서 엔코더 값이 증가하도록 구성 해야한다. 이는 이후 제어기를 구성했을 때 제어 신호의 값이 양(+)이면 출력 신호는 증가하도록 구성한다는 의미이다.

다음 절의 실험을 실시하기 전에 **실습 연습문제 3.1**을 실행해서 모터가 원만하게 구동이 되는지 확인할 필요가 있다.

3.9.2 DC 모터 모델의 계수 측정

모터 모델의 모든 계수를 실험에 의해서 측정하는 것은 쉽지 않은 작업이다. 그러나 비교적 덜 중요한 계수를 생략한다면 모터 모델의 계수 측정이 쉬워질 수 있다. 특히 아마추어 코일의 인덕턴스를 무시한다면 모터 모델의 미분 방정식의 차수가 줄게 되므로 모터의 모델을 구하는 것이 쉬워진다. 이 실험에서는 실험을 쉽게 하기 위해서 인덕턴스를 무시한 모델을 사용한다. 또한 선형 마찰도 무시한다. 모터의 모델에 존재하는 마찰의 종류에는 선형 마찰과 비선형 마찰이 있으며, 선형 마찰은 비선형 마찰에 비해서 크기가 작은 편이라 무시해도 큰 영향이 없다. 비선형 마찰은 무시할 수 있을 만큼 작다고 말하기 어려울 정도로 영향을 미치기는 하지만, 비선형 마찰에 관한 이론은 이 책의 범위를 벗어나므로 이 실험의 모델에는 포함하지 않는다. 이후의 실험에서 이론과 실험이 차이가 생기는 원인은 주로 비선형 마찰로 인한 부분이라고 말할 수 있다.

아래의 식은 간략하게 만든 DC 모터의 미분 방정식이다. 계수와 변수에 대한 정의는 3.1의 정의와 동일하다.

$$e_a = R_a i_a + K_b \frac{d\theta}{dt} \quad (3.250)$$

$$\tau = K_t i_a = J_a \frac{d^2\theta}{dt^2} \quad (3.251)$$

위 미분 방정식의 라플라스 변환을 취하면 다음의 식을 얻을 수 있다.

$$E_a(s) = R_a I_a(s) + K_b s\Theta(s) \quad (3.252)$$

$$K_t I_a(s) = J_a s^2 \Theta(s) \quad (3.253)$$

위의 식에서 다음과 같이 전달 함수를 얻을 수 있다.

$$\frac{\Theta(s)}{E_a(s)} = \frac{1/K_b}{s \left(\frac{R_a J_a}{K_t K_b} s + 1 \right)} = \frac{1/K_b}{s(T_m s + 1)}, \quad T_m = \frac{R_a J_a}{K_t K_b} \quad (3.254)$$

위의 식에서 T_m 은 모터의 시정수(time constant)이다. 위의 전달 함수에서 모든 계수의 개별적인 값을 구할 필요는 없으며 T_m 과 K_b 를 구하는 것으로 충분하다. 전달 함수는 출력 변수의 정의에 따라서 다를 수 있다. 위의 전달 함수는 출력이 모터의 각도인 경우의 전달 함수이다. 만약 출력을 모터의 각속도로 가정하면 전달 함수는 다음과 같다.

$$\frac{\Omega(s)}{E_a(s)} = \frac{s\Theta(s)}{E_a(s)} = \frac{1/K_b}{T_m s + 1} \quad (3.255)$$

위의 식에서 $\Omega(s)$ 는 각속도 ω 의 라플라스 변환이며, ω 는 각도 θ 의 미분이다. 위의 전달 함수의 식에서 T_m 과 K_b 의 값을 구하면 전달 함수를 결정할 수 있다.

이 실험에서는 DC 모터 속도의 계단 응답을 기록해서 전달 함수를 결정하기 위한 계수를 구하는 실험을 수행한다. 즉, 전달 함수가 식 (3.255)인 모터에 대해서 계단 입력 신호를 입력 했을 때 출력 신호인 모터의 각속도를 기록해서 그래프로 그리는 실험을 수행한다. 앞의 다른 실험과 마찬가지로 STM32CubeIDE에서 새로운 프로젝트를 만들고 프로젝트의 이름을 MotorSpeed로 정한다. 프로젝트의 설정은 1장 Lab1의 **1.4.3 타이머 인터럽트와 D/A 변환** 프로젝트와 동일하게 설정한 후, 엔코더 신호의 입력을 위해서 3.9.1절과 같은 방법으로 Timer2에 대한 설정을 한다. 또한 이 프로젝트에서는 사용자 버튼 스위치를 이용해서 버튼을 누르는 순간 계단 입력 신호가 발생되어 모터에 인가되도록 한다. STM32F429 디스커버리 보드에서는 사용자가 사용할 수 있는 파란색의 버튼 스위치가 장착되어 있다. 이 버튼 스위치를 누르는 순간 인터럽트가 발생하도록 설정한다. 사용자 버튼은 PA0 핀에 연결되어 있으며 그림 3-75와 같이 외부 인터럽트 입력으로 사용될 수 있도록 설정한다. 그림 3-75는 System Core 항목에서 GPIO 항목을 선택하면 볼 수 있는 화면이며, 이 화면에서 PA0/WKUP 항목을 클릭하면 GPIO mode 선택 창이 아래에 나온다. 이 창에서 External Interrupt Mode with Rising edge trigger detection 항목을 선택한다.

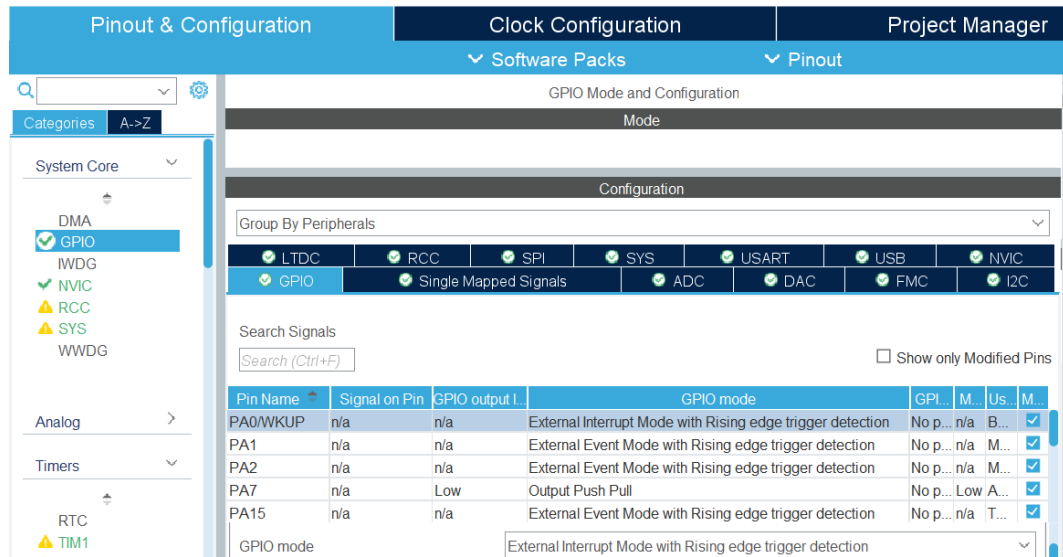


그림 3-75 PA0을 외부 인터럽트 모드로 설정

다음으로 그림 3-76과 같이 NVIC 탭을 선택해서 EXTI line0 interrupt 항목의 Enabled를 체크해서 외부 입력 인터럽트를 활성화 한다.

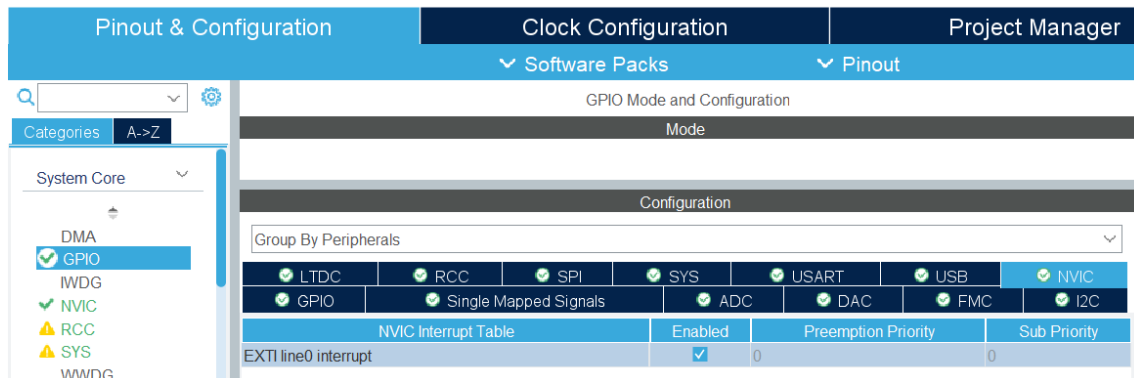


그림 3-76 Enable EXTI line0 interrupt

코드 생성을 실행한 후 아래의 코드를 main.c에 입력한다. 아래의 코드 이외에 printf 함수를 사용하기 위해서 코드 3.1과 코드 3.2의 입력이 필요하다.

코드 3.5

```
/* USER CODE BEGIN PV */
#define CAPTURE_START      1
#define CAPTURE_FINISHED  2
int data_flag=0;
int data_counter=0,sf=1000;
int data[4000];
/* USER CODE END PV */
```

코드 3.6

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim10);
```

```

    HAL_TIM_Encoder_Start(&htim2,TIM_CHANNEL_1);
    HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
/* USER CODE END 2 */

```

코드 3.7

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if (data_flag == CAPTURE_FINISHED) {
        printf("%d %d\r\n",0,0);
        for (int i=1; i < 2*sf ;i++){
            printf("%d %d\r\n",i,data[i]-data[i-1]);
        }
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
        data_flag = 0;
    }
/* USER CODE END WHILE */

```

코드 3.8

```

/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
    data_flag = CAPTURE_START;
}
/* USER CODE END 4 */

```

코드 3.9

```

/* USER CODE BEGIN Callback 0 */
int x_encoder;
if (htim->Instance == TIM10) {
    if (data_flag == CAPTURE_START) {
        HAL_DAC_SetValue(&hdac,DAC_CHANNEL_2,DAC_ALIGN_12B_R,(uint32_t)(2048+1024));
        x_encoder = TIM2->CNT;
        data[data_counter++] = x_encoder;
        if (data_counter >= 2*sf) {
            data_flag = CAPTURE_FINISHED;
            data_counter = 0;
            HAL_DAC_SetValue(&hdac,DAC_CHANNEL_2,DAC_ALIGN_12B_R,(uint32_t)(2048));
        }
    }
}
/* USER CODE END Callback 0 */

```

모든 연결을 완료한 후 SmarTTY 시리얼 터미널 프로그램을 실행한다. SmarTTY 프로그램은 터미널 화면에 프린트되는 내용을 파일에 저장할 수 있는 기능이 있다. 실행 프로그램을 실습용 보드에 다운로드 해서 실행한 후, 파란색 사용자 버튼 스위치를 누르면 5V 의 직류 전압이 2 초 동안 파워 앰프에 인가된다. 모터는 2초 동안 회전 후 멈추고 터미널 화면에는 그림 3-77 과 같이 모터의 속도 값이 프린트 된다. 터미널 화면에는 한 개의 줄에 2개의 숫자가 프린트 되며, 두번째 숫자는 1msec 동안 증가하는 엔코더의 카운트 값이다. 이 값을 이용하면 모터의 속도를 계산할 수 있다. 각 줄의 첫번째 숫자는 1msec 마다 증가하는 샘플링 카운트 값으로서 그래프를

그리는데 사용된다. 샘플링 주파수가 1KHz 이므로 2 초 동안의 데이터는 모두 2000 개 이다. SmartTTY 메뉴 바의 디스켓 모양의 버튼을 누르면 텍스트 파일 형식의 파일로 저장할 수 있다. 실험을 시작하기 전에 SmartTTY 메뉴 바의 빨간색 X 표 모양의 버튼(screen clear button)을 클릭해서 이전에 프린트된 내용을 모두 삭제하는 것을 잊지 않는다.

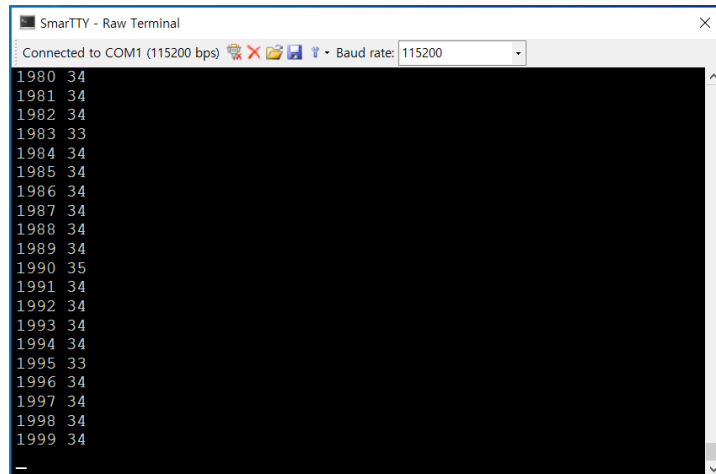


그림 3-77 시리얼 터미널 스크린

아래의 MATLAB 프로그램은 저장된 데이터 파일을 이용해서 그림 3-78 과 같은 계단 응답 그래프를 그린다. 아래의 프로그램에서 데이터 파일의 이름은 확장자 없이 data 라고 가정한다.

코드 3.10

```
clear
clf
sf=1000;
load -ascii data
for i=1:2*sf
    x(i)=data(i,1)/sf;
    y(i)=data(i,2);
end
figure(1)
plot(x,y)
axis([0 0.2 0 50])
xlabel('Time(sec)');
ylabel('Encoder count/(1msec)');
grid on
```

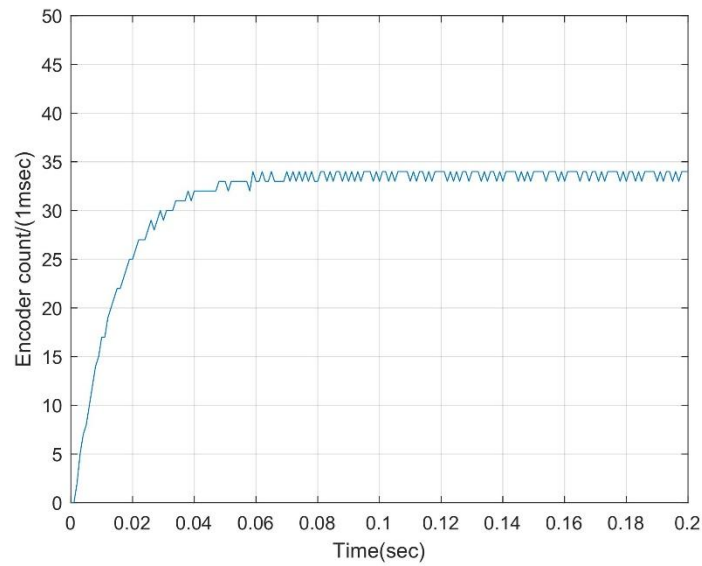


그림 3-78 모터 속도의 계단 응답

이 실험에서 사용된 모터의 엔코더는 한 회전 당 96 개의 펄스가 출력되고 쿼드러처 방식의 엔코더 이므로 모터의 한 회전 당 엔코더 카운트의 수는 $96 \times 4 = 384$ 이다. 그림 3-77 에서 정상 상태의 값은 대략 34 이므로 모터의 정상 상태 각속도는 다음과 같이 계산할 수 있다. 샘플링 주기는 1msec 라는 사실을 기억한다.

$$\frac{34 \times 1000 \times 2\pi}{96 \times 4} = 556 \text{ (rad/sec)} \quad (3.256)$$

이 실험에서 사용된 파워 앰프의 게인은 2 이므로 모터에 인가된 전압은 10V 이다. 따라서 전달 함수의 DC 이득(DC gain)은 55.6 이다. 다음으로 그림 3-77 의 계단 응답에서 정상 상태 출력의 63%에 도달하는 시간을 읽으면 대략 0.015 이다. 따라서 모터의 시정수 T_m 은 대략 0.015 라고 볼 수 있다. 위에서 얻은 수치를 이용하면 다음과 같이 전달 함수를 결정할 수 있다.

$$\frac{\Omega(s)}{E_a(s)} = \frac{s\Theta(s)}{E_a(s)} = \frac{1/K_b}{T_ms+1} = \frac{55.6}{0.015s+1} \quad (3.257)$$

위의 실험은 D/A 컨버터에서 출력되는 전압을 변경해서 반복할 수 있다. 이상적인 선형 시스템이라면 입력 전압이 바뀌어도 동일한 결과를 얻을 수 있어야 하지만 실제 DC 모터는 완벽한 선형 시스템이 아니므로 D/A 컨버터 값에 따라서 차이가 나는 결과를 얻을 수 있다. 좀더 정확한 실험 결과를 얻기 위해서 여러 가지 다른 D/A 컨버터 값에 대한 실험을 반복한 후 계수들의 평균 값을 취하는 등의 방법을 취할 수도 있다. 그러나 제어 시스템의 설계를 위해서 반드시 완벽한 모델이 필요한 것은 아니며 근사적인 모델을 이용해서 제어 시스템을 설계해도 원하는 결과를 얻는 것이 가능하다.