4주차 실험 결과 보고서

9조

201524414 기호영

201524437 김요한

201624410 권선근

201824461 남지원

1. 실험 목표

1) 스캐터 파일의 이해 및 플래시 프로그래밍

2) 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작

3) 센싱에서 폴링 방식의 이해

2. 배경 지식

그림 1. 릴레이 모듈

릴레이 모듈

① 릴레이를 제어하는 모듈

② 전자기 유도원리를 이용해 스위치 역할을 한다.

③ 릴레이에 신호를 가하면 출력 상태(ON/OFF)가 변경된다.

스캐터 파일

: 실행시킬 바이너리 이미지가 메모리에 로드될 때, 바이너리 이미지의 어떤 영역이 어느 주소에 어느 크기만큼 배치돼야 할지 작성한 파일

센싱 방식

Interrupt	Polling
Hardware 의 변화를 감지해 외부로부터 전기신호 입력을 CPU 가 알아챈다.	Hardware 의 변화를 지속적으로 읽어 들여 변화를 알아챈다.
CPU 마다 다른 방식으로 동작한다.	신호를 판단하기 위해 지속적으로 확인해야 한다.
진행 중인 일을 잠시 멈추고 interrupt 처리 루틴을 실행해서 신호를 처리한다.	다른 일을 하는 중에는 신호를 읽을 수 없다.

3. 실험 방법

1) 보드 연결

보드와 J-LINK를 연결하고 순서대로 J-LINK와 보드의 전원을 켜서 작동을 확인한다.

2) 프로젝트 생성 및 옵션 설정

3주차의 과정과 동일하다.

3) 스캐터 파일 작성 및 업로드

제공된 스캐터 파일에서 작성해야 하는 레지스터 값은 ROM과 RAM에서 할당할 메모리 주소다.

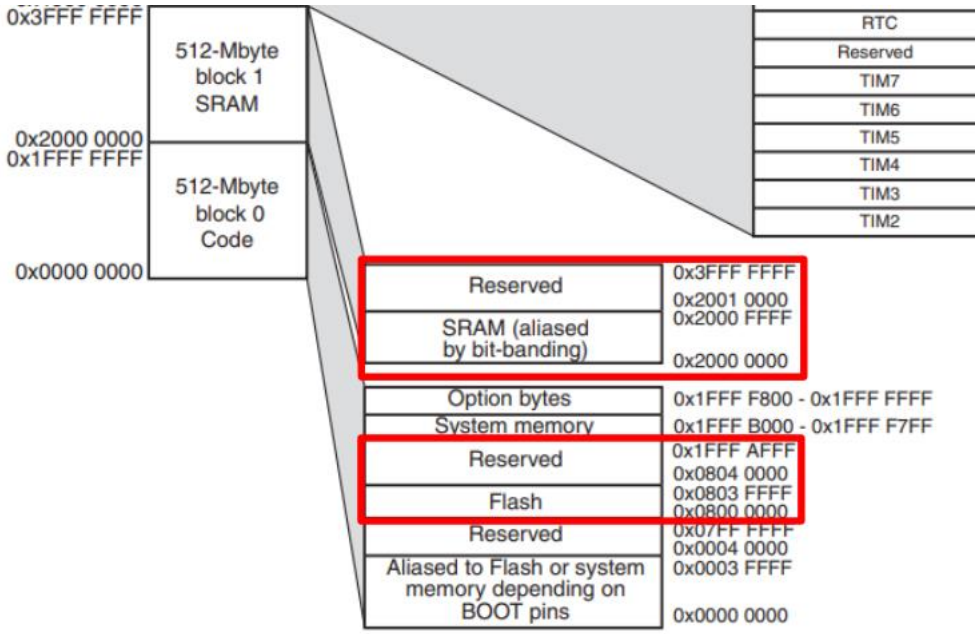


그림 2. ROM/RAM 메모리 주소

그림 1에서 RAM과 ROM의 메모리 시작 주소가 각각 0x20000000, 0x08000000 임을 알 수 있다. 별도 미션지에서 제시한 RAM의 크기는 0x00008000, ROM의 크기는 0x00080000 이므로 할당할 메모리 주소는 그림 3과 같이 작성했다.

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$₩config₩ide₩IcfEditor₩cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x08080000;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__   = 0x20008000;
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x1000;
define symbol __ICFEDIT_size_heap__   = 0x1000;
/**** End of ICF editor section. ###ICF###*/

define memory mem with size = 4G;
define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];
define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__];

define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };
```

그림 3. 수정한 스캐터 파일

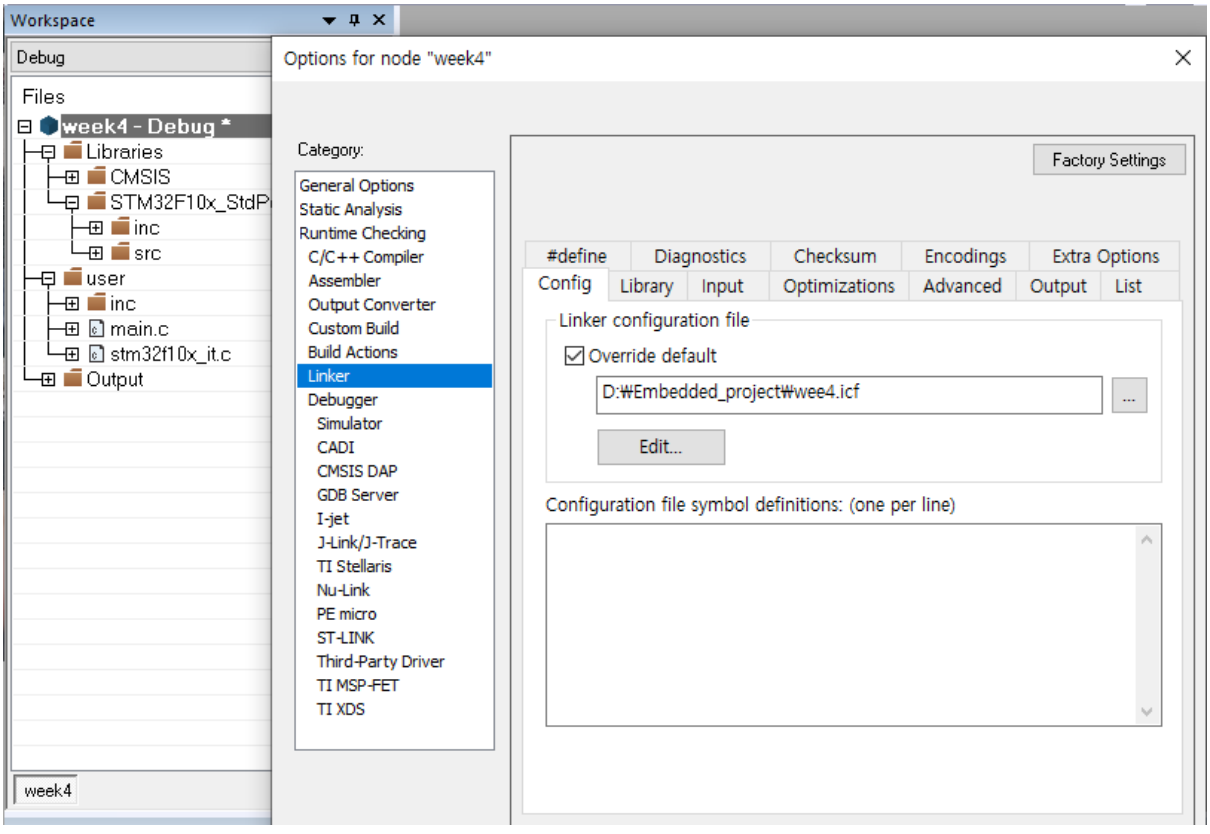


그림 4. 스캐터 파일 업로드

작성한 스캐터 파일은 그림 4와 같이 프로젝트에 업로드한다.

4) 소스코드 작성

```
#include "stm32f10x.h"
#define RCC_APB2_ENR *(volatile unsigned int *)0x40021018

#define PORTCL *(volatile unsigned int *)0x40011000
#define PORTCH *(volatile unsigned int *)0x40011004
#define PORTC_set *(volatile unsigned int *)0x40011010
#define PORTC_reset *(volatile unsigned int *)0x40011014
#define PORTC_data *(volatile unsigned int *)0x40011008

#define PORTDL *(volatile unsigned int *)0x40011400
#define PORTDH *(volatile unsigned int *)0x40011404
#define PORTDH_data *(volatile unsigned int *)0x40011408
#define SET_LED *(volatile unsigned int *)0x40011410
#define RESET_LED *(volatile unsigned int *)0x40011414

static void delay(volatile unsigned int nTime){
   for(;nTime>0;nTime--);
}
int main(void){
   RCC_APB2_ENR=0x38;

   PORTCL &= 0x00000000;
   PORTCL |= 0x00444400;
   PORTCH &= 0x00000000;
   PORTCH |= 0x00000033;
   PORTC_set = 0x3C;
   PORTC_reset |= 0x300;

   PORTDL &= 0x00000000;
   PORTDL |= 0x30033300;
   PORTDH &= 0x00000000;
   PORTDH |= 0x00004000;

   SET_LED = 0x800;
   SET_LED &= 0x00000000;

   while(1){
      if (~PORTC_data & 0x20) {
```

```
            PORTC_set |= 0x300;
            delay(2000000);
            PORTC_reset |= 0x300;
            delay(2000000);
        }
        else if(~PORTC_data & 0x4){
            PORTC_set |= 0x300;
            delay(5000000);
            PORTC_reset |= 0x300;
            delay(2000000);
        }
        else if(~PORTC_data & 0x10) {
            PORTC_set |= 0x100;
            delay(2000000);
            PORTC_reset |= 0x100;
            delay(2000000);
        }
        else if (~PORTC_data & 0x8) {
            PORTC_set |= 0x200;
            delay(2000000);
            PORTC_reset |= 0x200;
            delay(2000000);
        }
        else if (~PORTDH_data & 0x800) {
            while(1){
                SET_LED |= 0x9C;
                delay(800000);
                RESET_LED |= 0x9C;
                delay(800000);
                if(~PORTDH_data & 0x800){
                    delay(1000000);
                    RESET_LED |= 0x9C;
                    break;
                }
            }
        }
    }
    return 0;
}
```

i. Memory Address

```
#define RCC_APB2_ENR *(volatile unsigned int *)0x40021018

#define PORTCL *(volatile unsigned int *)0x40011000
#define PORTCH *(volatile unsigned int *)0x40011004
#define PORTC_set *(volatile unsigned int *)0x40011010
#define PORTC_reset *(volatile unsigned int *)0x40011014
#define PORTC_data *(volatile unsigned int *)0x40011008

#define PORTDL *(volatile unsigned int *)0x40011400
#define PORTDH *(volatile unsigned int *)0x40011404
#define PORTDH_data *(volatile unsigned int *)0x40011408
#define SET_LED *(volatile unsigned int *)0x40011410
#define RESET_LED *(volatile unsigned int *)0x40011414
```

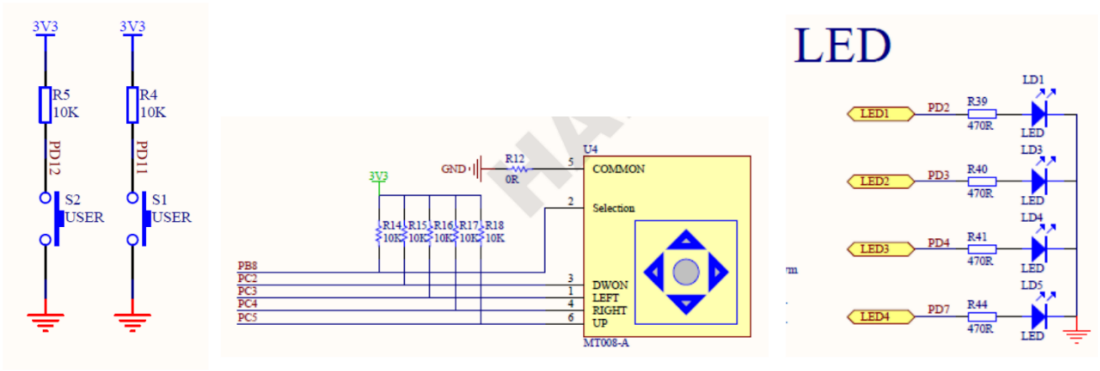


그림 5. Schematic GPIO Port

이번 실험에서 사용할 보드의 기능은 User Button, 조이스틱, LED이고 이들은 Port C와 Port D에 연결되어 있다. 또한 릴레이 모듈을 사용하기 위해서는 Port C에 연결해야 한다. 따라서 이번 실험에서는 Port C와 Port D를 사용할 것이다.

Port D	0x4001 1400 - 0x4001 17FF
Port C	0x4001 1000 - 0x4001 13FF

그림 6. GPIO Port Address

그림 5에서 각 Port의 Base address를 알 수 있다. Base address에 원하는 레지스터의 Address offset을 더한 값들을 define한 것이 상단의 코드이다. 아래에서 각 레지스터 별 Address offset을 확인할 수 있다.

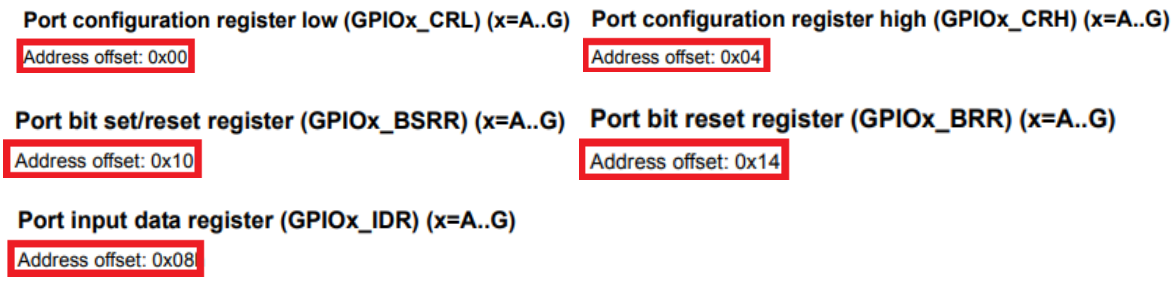


그림 7. Register Address Offset

ii. delay 함수

```
static void delay(volatile unsigned int nTime){
for(;nTime>0;nTime--);
}
```

delay 함수는 입력값 nTime만큼 반복문을 실행해 시간을 소요하게 하는 함수이다.

iii. Main 함수

a. Init

```
int main(void){
  RCC_APB2_ENR=0x38;

  PORTCL &= 0x00000000;
  PORTCL |= 0x00444400;
  PORTCH &= 0x00000000;
  PORTCH |= 0x00000033;
  PORTC_set = 0x3C;
  PORTC_reset |= 0x300;

  PORTDL &= 0x00000000;
  PORTDL |= 0x30033300;
  PORTDH &= 0x00000000;
  PORTDH |= 0x00004000;

  SET_LED = 0x800;
  SET_LED &= 0x00000000;
...
```

Register를 사용하기 위해 각 Port들의 clock을 enable 해야 한다. 이번 실험에서 사용할 Port는 LED의 PD 2, 3, 4, 7, User Button의 PD 11, 조이스틱의 PC 2, 3, 4, 5, 릴레이 모듈의 PC 8, 9 이다. 따라서 PC와 PD 모두 CRL(0~7)과 CRH(8~15)를 사용하게 된다.

코드에서는 먼저 사용할 레지스터의 값을 &=0x0 연산을 통해 초기화 시킨다. 그 다음 각 레지스터에 맞는 Input mode와 Output mode를 계산해서 입력해야 한다.

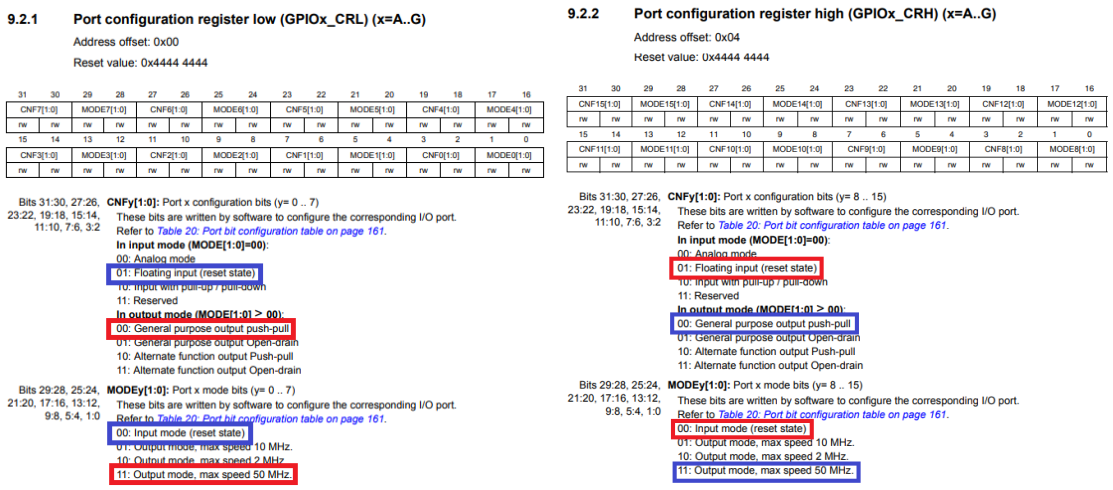


그림 8. CRL/CRH input/output mode

Port C의 Low Register를 사용하는 조이스틱은 Input mode를 사용하게 되므로 0x0100 = 0x4를 각 Port 마다 입력해야 한다. 그리고 Port C의 High Register를 사용하는 릴레이 모듈은 Output mode를 사용하기 때문에 0x0011 = 0x3을 입력해야 한다. 따라서

PORTCL |=0x00444400; **PORTCH |= 0x00000033;**

다음으로 Port D는 LED가 Low Register를 사용하기 때문에 Output mode인 0x0011 = 0x3을 입력해야 한다. 그리고 High Register를 사용하는 User Button은 Input mode인 0x0100 = 0x4를 입력한다. 따라서

PORTDL |= 0x30033300; **PORTDH |= 0x00004000;**

그리고 각 Port의 reset을 위해 Port C와 Port D의 BSRR register와 BRR register address를 초기화해야 한다. 따라서

0b00111100 = 0x3C 0b001100000000 = 0x300 0b100000000000=0x800

PORTC_set = 0x3C **PORTC_reset = 0x300** **LED_SET = 0x800**

b. Loop

```
while(1){
    if (~PORTC_data & 0x20) {
      PORTC_set |= 0x300;
      delay(2000000);
      PORTC_reset |= 0x300;
      delay(2000000);
    }
```

PC5와 연결된 조이스틱의 UP을 확인하기 위해 if문으로 (~PORTC_data & 0x20)를 확인한다. 여기서 PORTC_data는 Port C의 IDR 레지스터이다. UP이 입력된 것을 확인하면 릴레이 모듈이 연결된 PC 8, 9를 일정한 간격으로 Set, Reset한다. 간격은 delay 함수를 이용해 설정했다.

```
else if(~PORTC_data & 0x4){
      PORTC_set |= 0x300;
      delay(5000000);
      PORTC_reset |= 0x300;
      delay(2000000);
    }
```

PC2와 연결된 조이스틱의 Down을 if (~PORTC_data & 0x4)로 확인한다. Down이 입력된 것을 확인하면 PC 8, 9를 일정한 간격으로 Set, Reset한다. 단 UP일 때보다 간격은 길게 설정한다.

```
    else if(~PORTC_data & 0x10) {
      PORTC_set |= 0x100;
      delay(2000000);
      PORTC_reset |= 0x100;
      delay(2000000);
    }
```

PC4와 연결된 조이스틱의 Right를 if (~PORTC_data & 0x10)로 확인한다. Right이 입력된 것을 확인하면 PC 8을 일정한 간격으로 Set, Reset한다.

```
    else if (~PORTC_data & 0x8) {
      PORTC_set |= 0x200;
      delay(2000000);
      PORTC_reset |= 0x200;
      delay(2000000);
    }
```

PC3과 연결된 조이스틱의 Left를 if (~PORTC_data & 0x8)로 확인한다. Left가 입력된 것을 확인하면 PC 9를 일정한 간격으로 Set, Reset 한다.

```
else if (~PORTDH_data & 0x800) {
        while(1){
          SET_LED |= 0x9C;
          delay(800000);
          RESET_LED |= 0x9C;
          delay(800000);
          if(~PORTDH_data & 0x800){
            delay(1000000);
            RESET_LED |= 0x9C;
            break;
          }
        }
    }
```

PD 11과 연결된 User Button을 if (~PORTDH_data & 0x800)로 확인한다. 여기서 PORTDH_data는 Port D의 IDR 레지스터이다. User Button이 눌린 것을 확인하면 LED 4개가 일정한 간격으로 점멸하도록 한다. LED는 PD2, 3, 4, 7과 연결되어 있으므로 0x10011100 = 0x9C를 Set, Reset해 점멸하게 할 수 있다.

5) 릴레이 모듈 연결

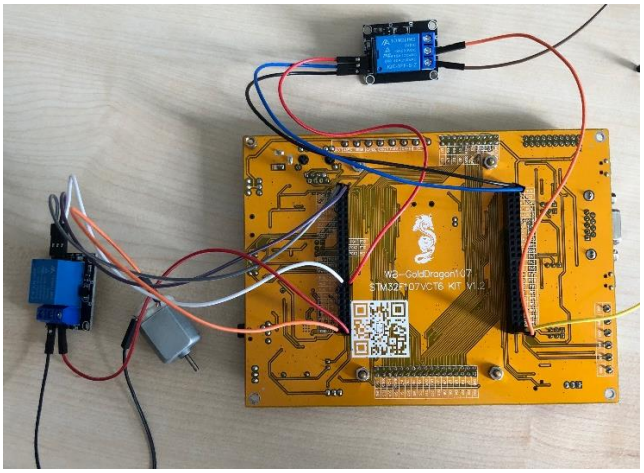


그림 9. 릴레이 모듈과 보드 연결

그림 10. 릴레이 모듈

i. 릴레이 모듈의 입력전압(VCC)과 보드의 3.3V 전원 연결

ii. 릴레이 모듈의 접지(GND)와 보드의 GND 연결

iii. 릴레이 모듈의 제어 신호(IN)와 보드의 PC 8, 9 연결

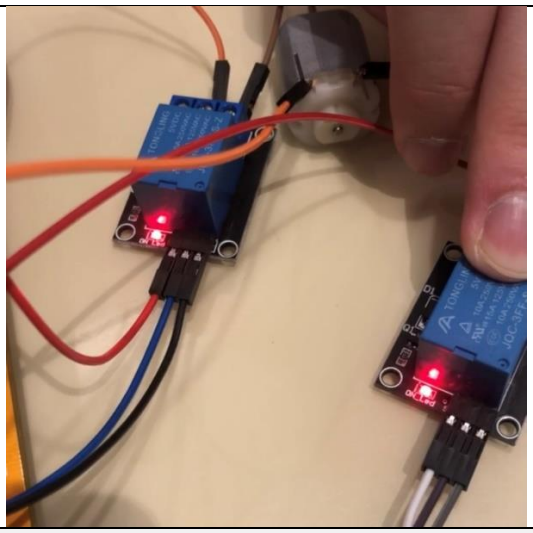
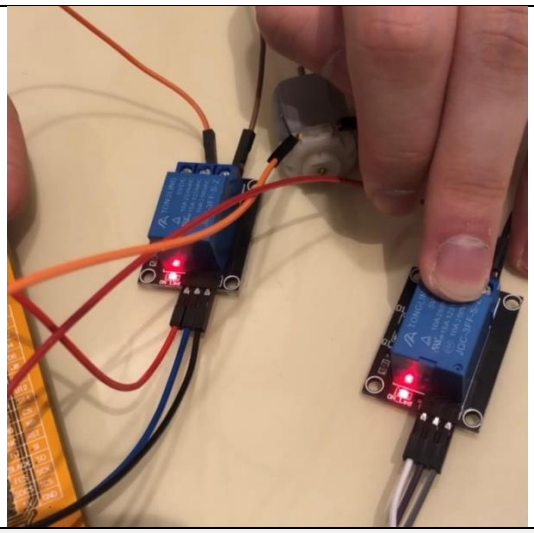
iv. 릴레이 모듈의 공통 단자(COM)에서 보드로 5V 전원 인가

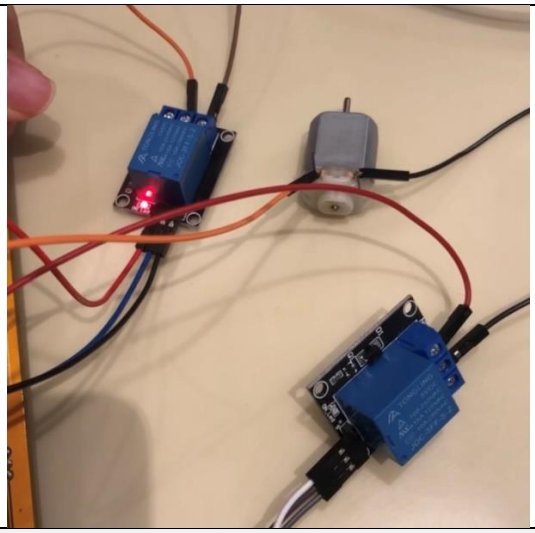
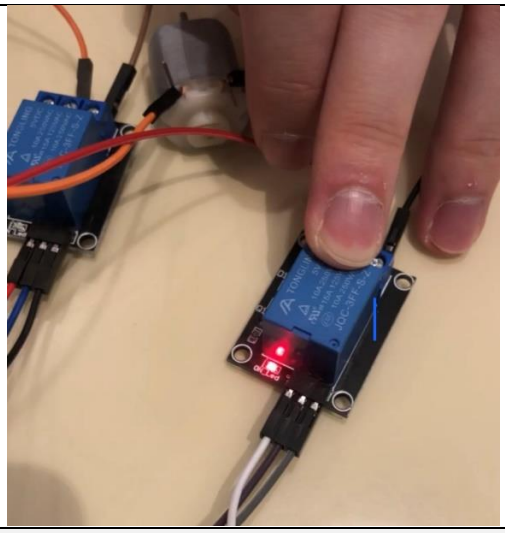
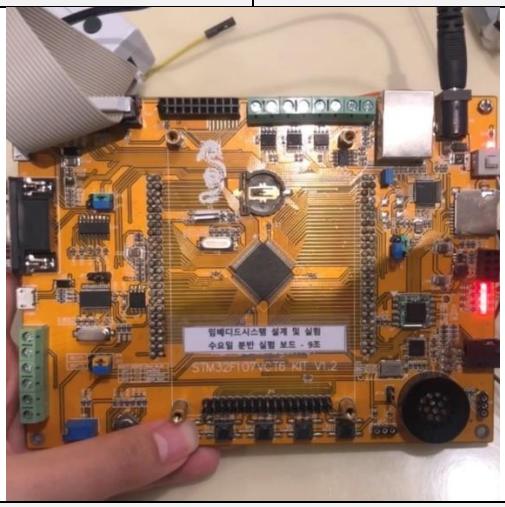
v. 릴레이 모듈의 NO와 모터 연결

릴레이 모듈과 모터를 연결할 때 NO와 NC 중 무엇을 연결하는지에 따라 동작에 차이가 있다. NO는 평소에 open 상태로 high 신호가 들어오면 close 상태가 된다. 반대로 NC는 평소에 close 상태이고 high 신호가 들어오면 open 상태로 변한다. 즉 NO는 전류가 흐를 때 모터가 동작하고 NC는 전류가 흐를 때 모터가 동작을 중지한다. 이번 실험에서는 조이스틱으로 신호를 주면 모터가 돌아가도록 만들 것이기 때문에 릴레이 모듈의 NO에 모터를 연결했다.

4. 결론

1) 보드 작동

	
Up	Down

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>Left</td><td>Right</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">User Button</td></tr>
</table>

2) 토의 및 감상

i. 스캐터 파일의 기능을 이해하고 업로드 할 수 있다.

ii. 릴레이 모듈을 보드에 연결하고 작동시킬 수 있다.

iii. Polling 방식을 이해할 수 있다.