[임베디드 설계 및 실험] 2조 4주차 결과 보고서
- Scatter File의 이해 및 릴레이 모듈의 활용

▣ 과 목 명	임베디드 설계 및 실험
▣ 담당교수	김 원 석
▣ 제 출 일	2022.09.25
▣ 학 과	정보컴퓨터공학부
▣ 조 원	202055600 정홍빈
	201924617 끼앗띠퐁 유옌
	201824636 이강우
	201824534 윤상호

1. 실험 목적

스케터 파일과 릴레이 모듈을 이해하고 임베디드 펌웨어를 통한 동작을 구현할 수 있다. 폴링 방식을 이해하고, 인터럽트와의 차이를 설명할 수 있다.

2. 실험 원리 및 이론

1). Scatter file

Scatter file이란 실행시킬 바이너리 이미지가 메모리에 로드될 때, 바이너리 이미지의 어떤 영역이 어느 주소에 어느 크기만큼 배치되어야 하는지에 대한 내용이 작성된 파일을 의미한다. 바이너리의 여러 부분을 각각 별개의 메모리 영역에 로드해야 하는 상황에 사용되며, 자주 사용되거나 빠른 실행을 요구하는 코드 영역을 우선 배치하여 성능 향상을 기대할 수 있다.

2). Floating과 Pull Up, Pull Down

Floating이란 전압이 High인지 Low인지 보기 힘든 상태를 의미한다. 작은 진동이나 잡음으로도 high와 low 사이를 빠르게 이동하여 오동작을 유발할 수 있어 이를 방지하기 위해 Pull Up, Pull Down 방식이 있다. Pull Up은 항상 Vcc에 연결 되어 있어 스위치가 On이 될 때 input에는 Low 신호가, Pull Down은 반대로 항상 GND에 연결 되어 있어 스위치가 On이 될 때 input에는 High 신호가 발생한다.

3). Polling 과 Interrupt의 차이점

Polling은 hardware의 변화를 지속적으로 읽어들여 변화를 알아채는 방법이다. Interrupt 방식보다 구현은 쉬우나 다른 일을 하는 중에 신호를 읽을 수 없기 때문에 시스템 성능저하의 원인이 되는 경우가 있다. 처리에 정확한 타이밍이 필요한 작업과는 어울리지 않는 방식이다.

Interrupt는 Polling보다 구현이 복잡하지만 처리속도가 빠르며 정확한 타이밍이 요구될 때 적합하다. 인터럽트 발생 시 진행 중인 일을 잠시 멈추고 인터럽트 처리 루틴을 실행하여 신호를 처리한다.

4). Relay 모듈

Relay 모듈이란 relay를 제어하는 모듈로, 전자기 유도 원리를 이용하여 스위치 역할을 할 수 있다. 릴레이에 신호를 가하면 출력 상태 (On/Off) 가 변경된다. 제어 신호(IN)를 받아 공통 단자(COM)의 이동으로 NO 또는 NC에 붙는 형식이다.

NO: 평소에 open, high 신호가 들어오면 close
NC: 평소에 close, high 신호가 들어오면 open

그림 1 Relay 모듈

3. 실습 진행 과정

1). Scatter File 수정 및 업로드

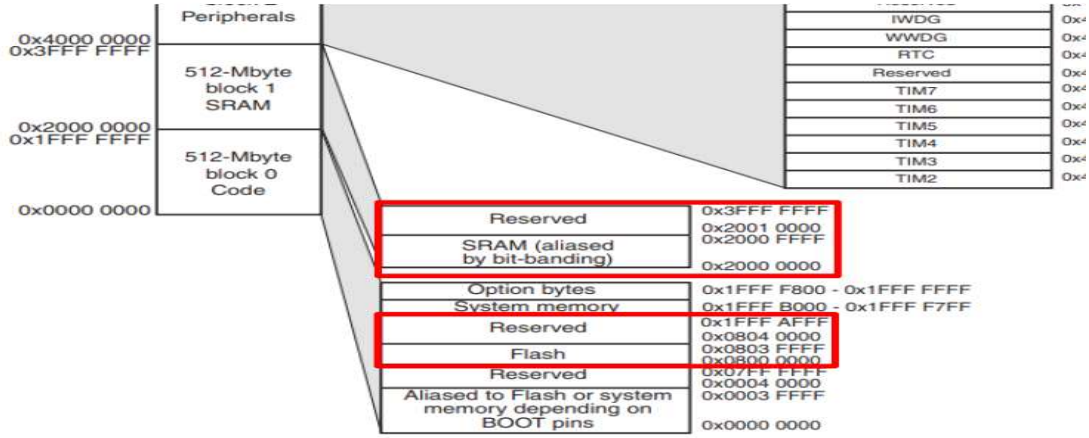

그림 2 ROM과 RAM의 주소

ROM은 0x08000000, RAM은 0x20000000이 시작 주소이다.

그림 3 수정된 myicf.icf 파일

ROM 크기(0x80000)와 RAM 크기(0x8000)를 할당하기 위해 수정한 파일이다.
크기는 미션지에서 제시된 내용으로 할당하였다.

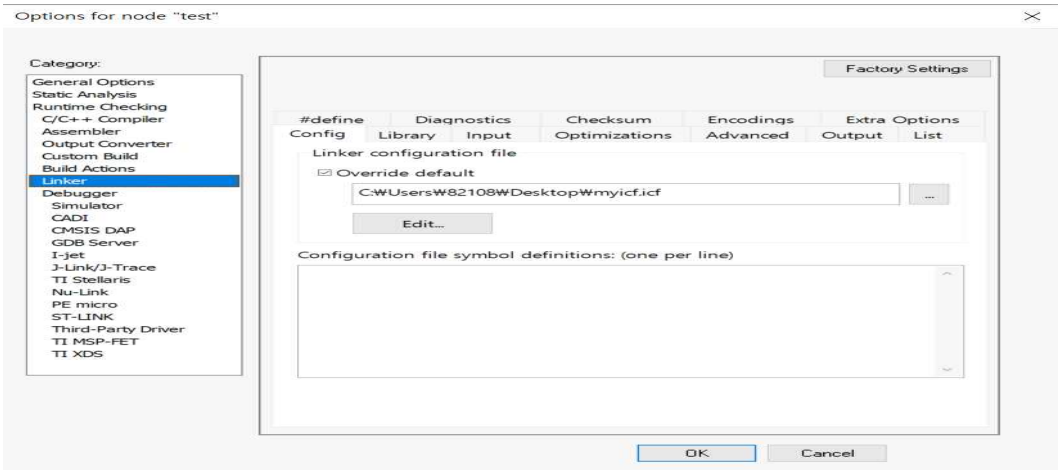

그림 4 myicf.icf 적용

작성한 myicf.icf 파일을 project -> Options -> Linker -> Config 로 이동한 뒤, Override default를 클릭하여 myicf.icf가 있는 위치로 이동하여 파일을 선택한다.

그 후, OK를 누르면 파일이 업로드된다.

2). 소스코드와 레퍼런스

```
#include <stdio.h>
#include "stm32f10x.h"

#define R_CC 0x40021000
#define PORT_B 0x40010C00          // pb 8 selection
#define PORT_C 0x40011000          // pc 5 stick, pc 8 relay
#define PORT_D 0x40011400          // pd 2,3,4 led, pd 11,12 user

typedef volatile unsigned int* VOL;
```

그림 5 사전 약속

코드의 간결성을 위하여 그림 5와 같이 선언하였다. 접근 시 항상 메모리 참조가 되어야 하기에 자주 쓰이게 될 자료형인 volatile unsigned int*를 VOL로 선언하였다.

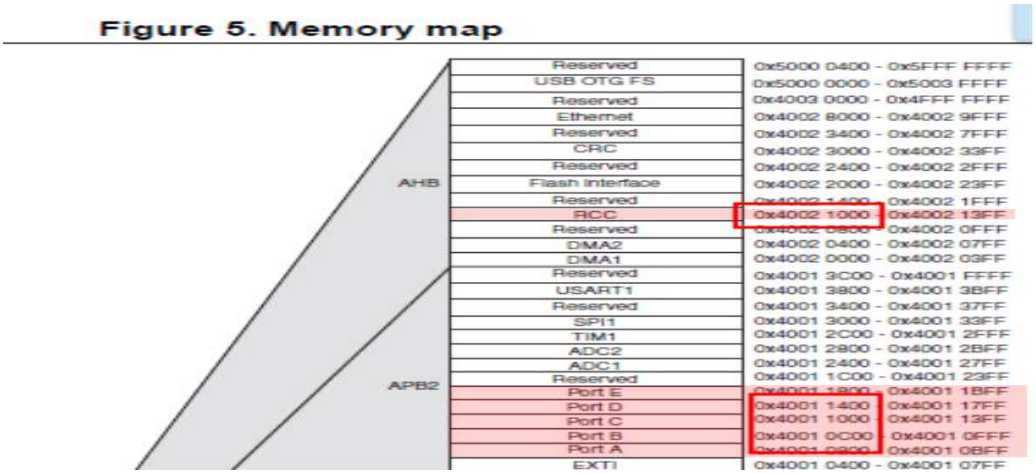

그림 6 (datasheet p.33) 레지스터 메모리 매핑 주소

3주차에 진행했던 방식대로 datasheet에서 레지스터 메모리 매핑 주소를 찾아 RCC와 사용할 포트들의 주소를 찾아내었다.

```
void delay()
{
  int i;
  for(i = 0; i < 10000000; i++) {}
}
```

그림 7 delay 함수

제공된 delay 함수이다. 이 함수로 delay를 가능하게 한다.

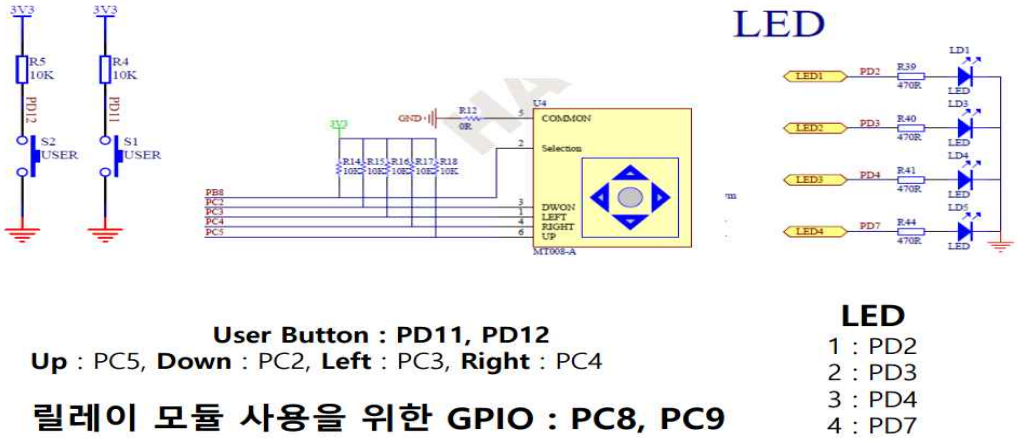

그림 8 전체 회로도 및 사용할 GPIO

릴레이 모듈을 위한 추가적인 GPIO가 필요하다. 릴레이 모듈의 제어 신호핀은 다른 입출력과 겹치면 안 되기 때문에 사용하지 않는 GPIO 중 PC8을 선택하기로 하였다. 그래서 이번에는 port B, C, D를 사용한다.

```
int main(void)
{
  *(VOL) (R_CC + 0x18) |= 0x38;          // clock enable portB, port C, port D
  *(VOL) (PORT_B + 0x04) = 0x8;          // configuration high, selection
  *(VOL) (PORT_C + 0x00) = 0x800000;     // configuration low, stick 5
  *(VOL) (PORT_C + 0x04) = 0x3;          // configuration high, relay 8
  *(VOL) (PORT_D + 0x00) = 0x00033300;   // configuration low, led 2,3,4
  *(VOL) (PORT_D + 0x04) = 0x88000;      // configuration high, user 11, 12
```

그림 9 enable과 configuration

이번 미션 수행에 필요한 port들을 enable하고, 사용할 GPIO들을 모두 configuration하였다.

7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

그림 10 (Reference 7.3.7) RCC_APB2ENR

레퍼런스를 참조하여 port B, C, D가 3,4,5번째 비트이므로 11100(2) -> 0x38로 enable 시켜준다.

9.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 0 .. 7)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

그림 11 (Reference 9.2.1) CRL

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2 **CNFy[1:0]:** Port x configuration bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
In input mode (MODE[1:0]=00):
00: Analog mode
01: Floating input (reset state)
10: Input with pull-up / pull-down
11: Reserved
In output mode (MODE[1:0] > 00):
00: General purpose output push-pull
01: General purpose output Open-drain
10: Alternate function output Push-pull
11: Alternate function output Open-drain

Bits 29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1:0 **MODEy[1:0]:** Port x mode bits (y= 8 .. 15)
These bits are written by software to configure the corresponding I/O port.
Refer to *Table 20: Port bit configuration table on page 161.*
00: Input mode (reset state)
01: Output mode, max speed 10 MHz.
10: Output mode, max speed 2 MHz.
11: Output mode, max speed 50 MHz.

그림 12 (Reference 9.2.2) CRH

그림 11, 12의 레퍼런스를 참고하여 input이면 pullup, pulldown으로, output이면 max speed 50MHz로 설정해두었다.

여기서 리셋값은 0x00000000이 아니라 0x44444444이므로, 이를 유의해서 바꿔야 할 비트는 or 연산자가 아니라 = 연산자를 사용하는 것이 바람직하다.

```
while(1){
  if(!(*(VOL)(PORT_C + 0x08) & 0x20)){                    //up 7
    *(VOL) (PORT_C + 0x10) = 0x100;          // relay start
    delay();
    *(VOL) (PORT_C + 0x14) = 0x100;          // relay stop
  }
  else if((*(VOL)(PORT_D + 0x08) & 0x800) != 0x800){      // 11
    *(VOL) (PORT_D + 0x10) = 0x4;            // led on
    delay();
    *(VOL) (PORT_D + 0x14) = 0x4;            // led off
  }
  else if(!(*(VOL)(PORT_D + 0x08) & 0x1000)){             // 12
    *(VOL) (PORT_D + 0x10) = 0x8;            // led on
    delay();
    *(VOL) (PORT_D + 0x14) = 0x8;            // led off
  }
  else if(!(*(VOL)(PORT_B + 0x08) & 0x100)){              // 8
    *(VOL) (PORT_D + 0x10) = 0x10;           // led on
    delay();
    *(VOL) (PORT_D + 0x14) = 0x10;           // led off
  }
}
return 0;
}
```

그림 13 전체 logic

지난 3주차 실험과 거의 동일한 logic이다. 다만 이번에는 delay 함수로 육안으로 풀링을 확인할 수 있다는 점이 차이이다. IDR 값을 받아서 비교하고, True이면 if문 내부로 들어가 BSRR값을 변경하여 set하고, delay 함수를 실행시킨 후, delay 함수가 리턴되면 BRR 값을 변경하여 reset하고, 이를 무한 반복하는 형식이다. 참조한 레퍼런스는 다음과 같다. 3주차 실험과 같은 내용이다.

9.2.3 Port input data register (GPIOx_IDR) (x=A..G)

Address offset: 0x08h

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDRy:** Port input data (y= 0 .. 15)
These bits are read only and can be accessed in Word mode only. They contain the input value of the corresponding I/O port.

그림 14 (Reference 9.2.3) IDR

9.2.5 Port bit set/reset register (GPIOx_BSRR) (x=A..G)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x Reset *bit y (y= 0 .. 15)*
These bits are write-only and can be accessed in Word mode only.
0: No action on the corresponding ODRx bit
1: Reset the corresponding ODRx bit
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x *Set bit y (y= 0 .. 15)*
These bits are write-only and can be accessed in Word mode only.
0: No action on the corresponding ODRx bit
1: Set the corresponding ODRx bit

그림 15 (Reference 9.2.5) BSRR

9.2.6 Port bit reset register (GPIOx_BRR) (x=A..G)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved

Bits 15:0 **BRy:** Port x Reset bit y (y= 0 .. 15)

These bits are write-only and can be accessed in Word mode only.

0: No action on the corresponding ODRx bit

1: Reset the corresponding ODRx bit

그림 16 (Reference 9.2.6) BRR

3). Relay 모듈 연결

이제 릴레이 모듈을 살펴보자. 코드 구현은 완료했기 때문에 적절한 위치에 핀들을 꽂아주면 작동하게 될 것이다.

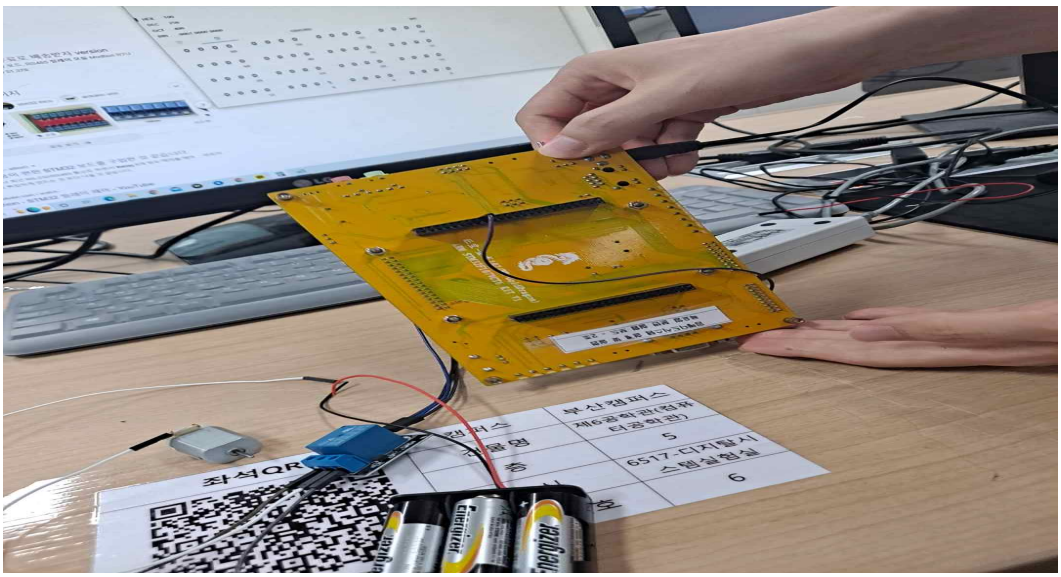

그림 17 PC8에 연결된 릴레이 모듈의 제어 신호 핀

먼저 코드에 적힌 대로 PC8에 제어 신호 핀을 연결하였다.

그림 18 Relay 모듈(재)

그림 1을 다시 보도록 하자. 제어 신호는 연결했으니, GND와 VCC도 연결해 주어야 한다. ARM에 있는 핀 중 GND와 5V에 각각 연결한다. 5V 핀 대신에 3V3 핀에 연결해서 진행하여도 무관하다.

실험에 사용되는 릴레이 모듈은 IN에 high를 넣으면 low로 동작하고, low를 넣으면 high로 동작한다. 조이스틱 up이 눌리면 값을 set 한다는 것과 IN을 high인 PC8 핀에 연결했다는 점을 고려하여 평소에 open, high 신호가 들어오면 close 상태인 NO와 COM을 모터와 연결해 주었다.

모터가 돌아가지 않았다. 보드에 인가되는 전원만으론 전류가 부족해 모터 구동이 잘 안될 수 있다고 한다. 우리 팀은 외부 전원을 연결하여 이 문제를 해결하였다.

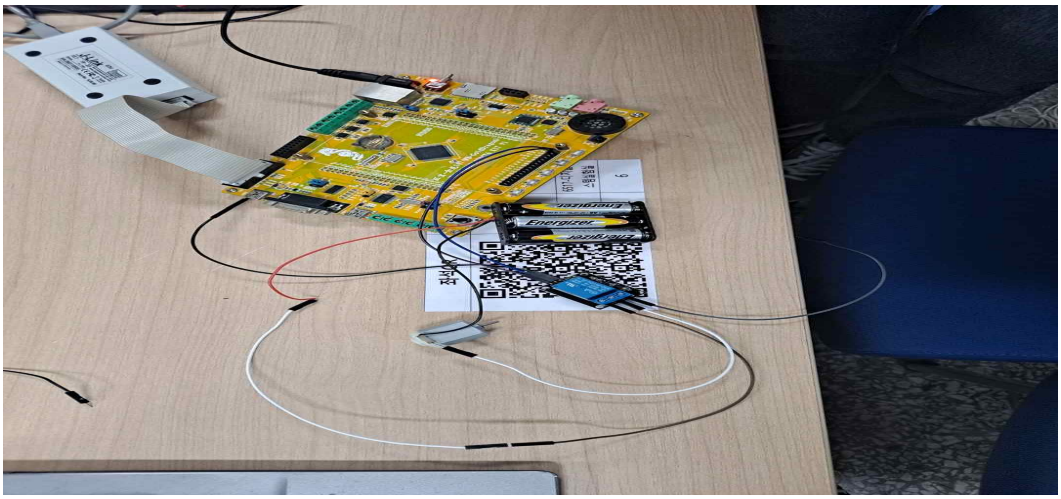

그림 19 외부 전원이 연결된 모습.

4. 실험 결과

1). 조이스틱 up

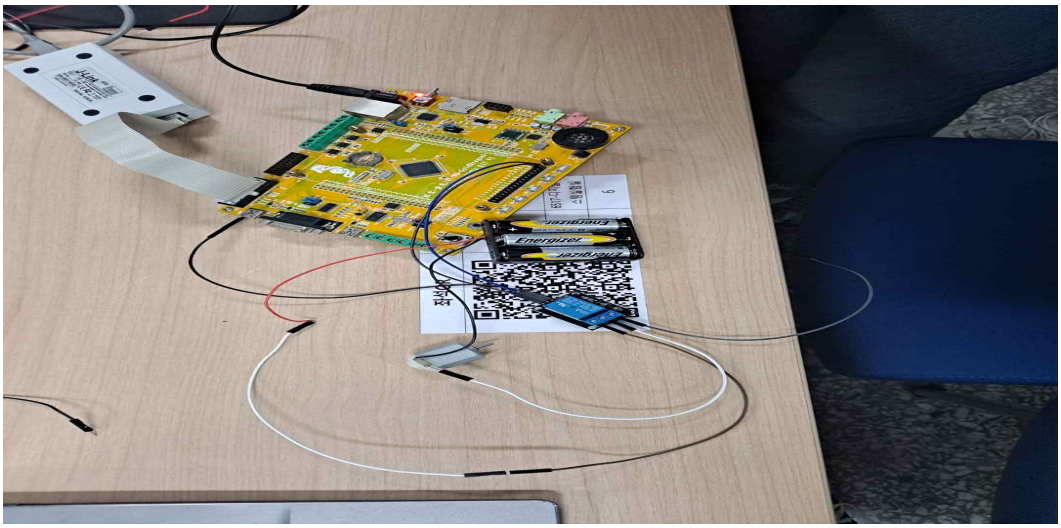

그림 20 전체 모습 (up 버튼을 누른 후의 상태. 올바르게 작동한다.)

딜레이만큼 릴레이가 구동된다.

2). Button 1 click

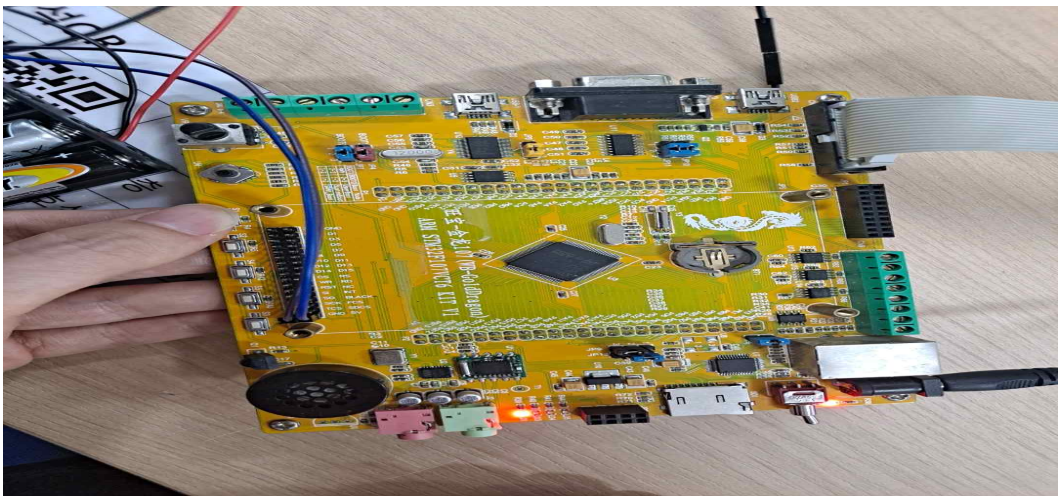

그림 21 user button 1 (PD11) click

LED1이 딜레이만큼 켜졌다가 꺼진다.

3). Button 2 click

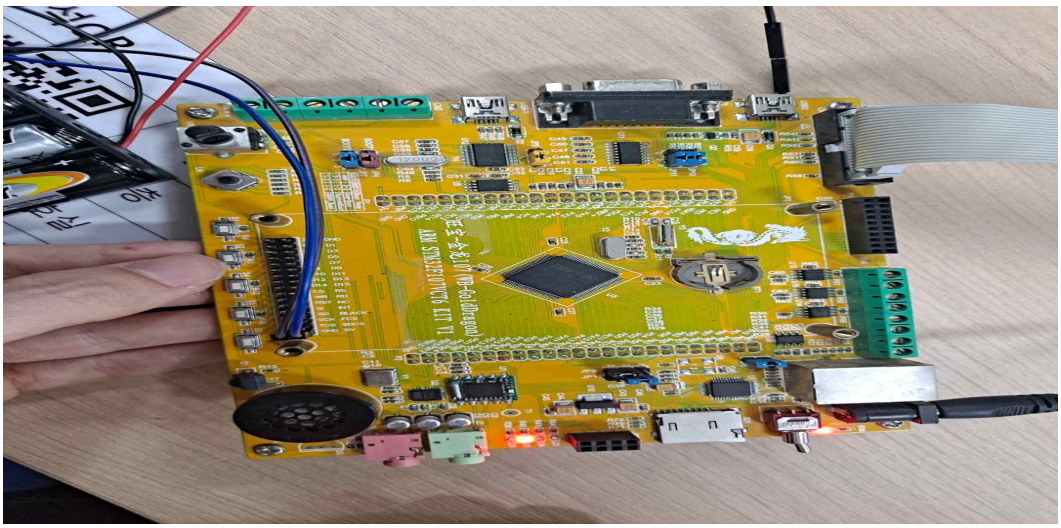

그림 22 user button 2 (PD12) click

LED2가 딜레이만큼 켜졌다가 꺼진다.

4). Selection click

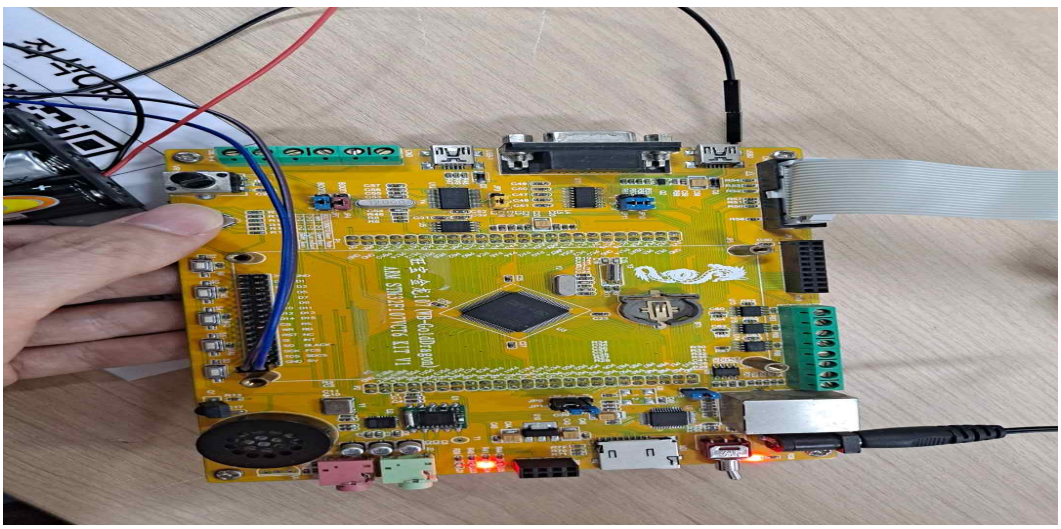

그림 23 selection click

LED3가 딜레이만큼 켜졌다가 꺼진다.

5. 결론

3주차 실험에서 조이스틱과 led제어를 진행하였기 때문에 이번 실험은 지난 내용을 바탕으로 새로 배운 내용들을 적용시키는 것이 핵심이었다.

스캐터 파일을 이용하여 원하는 메모리 위치에 원하는 크기만큼 적재할 수 있게 되었다.
릴레이 모듈의 구성 요소를 알고, 이를 연결하여 사용할 수 있게 되었다.

또한, 풀링 방식은 실행 도중에 다른 일을 할 수 없기 때문에 구현은 복잡할 수 있지만 인터럽트 방식이 필요하다는 것을 알게 되었다.

configuration 과정에서 초기화를 하지 않고 진행하였다가 버튼을 누르지 않았는데도 led가 켜지는 문제가 있어 1시간 동안 헤매다 찾아내어 수정하였다. 이 같은 상황을 막기 위해 값을 입력하기 전에 초기화를 해주는 방법을 채택하는 것이 좋을 것 같다.