REPORT

5주차 결과 보고서

수강과목 : 임베디드 시스템 설계 및 실험

담당교수 : 백윤주 교수님

학 과 : 전기컴퓨터공학부 정보컴퓨터공학전공

조 원 : 201724648 김경호 201724651 김장환

201624481 박윤형 201524588 조창흠

제출일자 : 2021 / 10 / 04

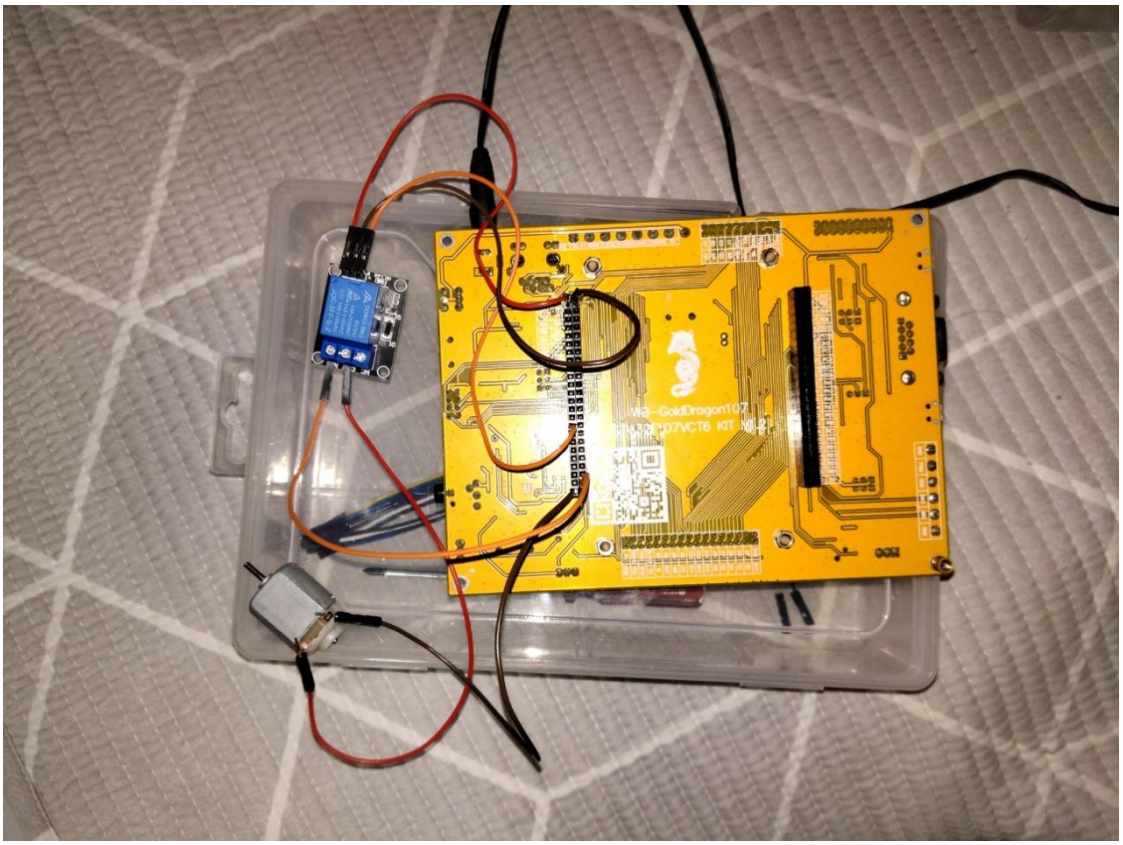
3주차 결과보고서

1. 실험 목표

1.1. 스캐터 파일의 이해 및 플래시 프로그래밍
1.2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작
1.3. 폴링 방식의 이해

2. 실험 내용

2.1. 보드와 릴레이 모듈의 연결



<그림1 릴레이 모듈 연결>

모터는 릴레이 모듈의 NO와 COM에 연결했다. 릴레이 모듈의 입력전압 핀과 접지 핀을 각각 보드의 3V3과 GND에 연결하고 릴레이 모듈의 제어신호 핀을 보드의 PC8과 연결해줬다.

2.2. 레지스터 및 주소 설정

-사용하는 핀 파악

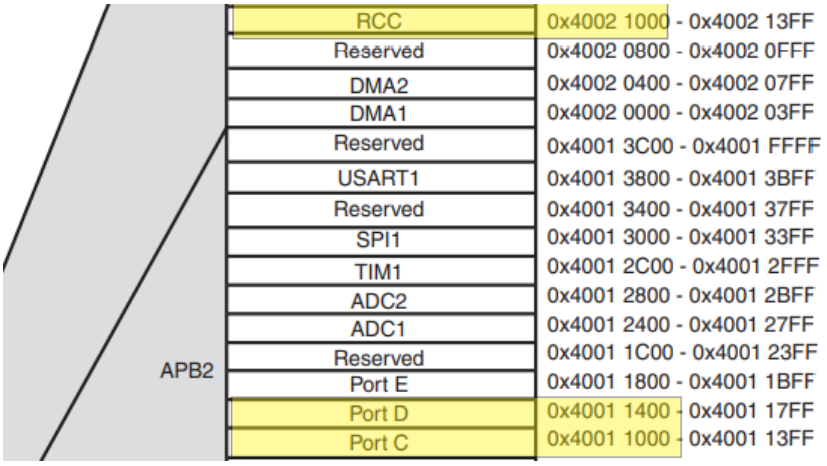
Input : S1 USER (PD11), S2 USER(PD12)

Output : LED (PD7), 릴레이 모듈 제어 신호(PC8)

-main.c 코드 설명

```
const int S1 = 0x800;    //S1 Button
const int S2 = 0x1000;   //S2 Button

const int LED = 0x80;    // PD7 LED
const int PC8 = 0x100;   //Give signal to relay module
```

핀 번호에 따라 S1(PD11), S2(PD12), LED(PD7), 릴레이 모듈 제어 신호(PC8)의 주소 값을 할당해준다.



<그림2. Memory mapping>

메모리 맵핑 정보를 참고해 RCC와 Port C, Port D의 base주소값을 알아낸다.

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.

<그림3. RCC_APB2ENR>

```
/*
 RCC BASE --> 0x4002 1000
 RCC_APB2ENR -> APB2 peripheral clock enable register address --> 0x18
 RCC_APB2ENR FINAL ADDRESS --> 0x4002 1000 + 0x18 = 0x40021018
*/
#define RCC_apb2 (*(volatile unsigned *)0x40021018)
```

실험에서 사용하는 Port C, Port D가 APB2 버스에 연결되어 있기 때문에 APB2에 clock을 인가해줬다.

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 4. Port configuration register high>

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

P

,

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 5. Port configuration register low >

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 6. Port input data register>

9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

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
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only

Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).

<그림 7. Port output data register>

```
#define PORT_C_CRH_config (*(volatile unsigned *)0x40011004)
#define PORT_C_CRH_set 0x44444445;

#define PORT_C_ODR (*(volatile unsigned *)0x4001100c)
```

그림 4에 따르면 PC8를 제어하기 위해서 포트의 offset을 0x04로 주어야 한다.

Port C 의 base 가 0x40011000 이므로 Port C의 CRH 주소인 PORT_C_CRH_config는 0x40011004가 된다. PC8을 output모드로사용하므로 포트PORT_C_CRH의 셋팅 값으로 0x44444445를 준다.

그림 7에 따라 output data register의 offset이 0x0c이므로 Port C의 output data register의 주소 값은 0x4001100c가 된다.

```
#define PORT_D_CRL_config (*(volatile unsigned *)0x40011400)
#define PORT_D_CRL_set 0x50000000

#define PORT_D_CRH_config (*(volatile unsigned *)0x40011404)
#define PORT_D_CRH_set 0x444cc444

#define PORT_D_ODR (*(volatile unsigned *)0x4001140c)

#define PORT_D_IDR (*(volatile unsigned *)0x40011408)    //button input
```

S1(PD11), S2(PD12)를 제어하기 위해 Port D 의 CRH를 세팅하고, LED(PD7)를 제어하기 위해 Port D의 CRL을 세팅해줬다. CRL의 세팅 값은 PD7을 input으로 설정한 0x50000000이고 CRH의 세팅 값은 PD11, PD12를 output으로 설정한 0x444cc444이다.

PD11, PD12가 input이고 PD7이 output이므로 Port D의 ODR과 IDR의 레지스터 주소를 구해놨다.

```
int main(void)
{
  // Apply the clock to the GPIO you want to use using RCC (reset and clock control).
  RCC_apb2 |= 0x30;

  //pin setting
  PORT_C_CRH_config ^= PORT_C_CRH_set;

  PORT_D_CRH_config ^= PORT_D_CRH_set;

  PORT_D_CRL_config ^= PORT_D_CRL_set;

  while(1){
    if(~PORT_D_IDR&S1){
      PORT_C_ODR = PC8;
      delay();
    }
    else if(~PORT_D_IDR&S2){
      PORT_D_ODR = LED;
      delay();
    }
    else{
      PORT_D_ODR = !LED;
      PORT_C_ODR = !PC8;
    }
  }
  return 0;
}
```

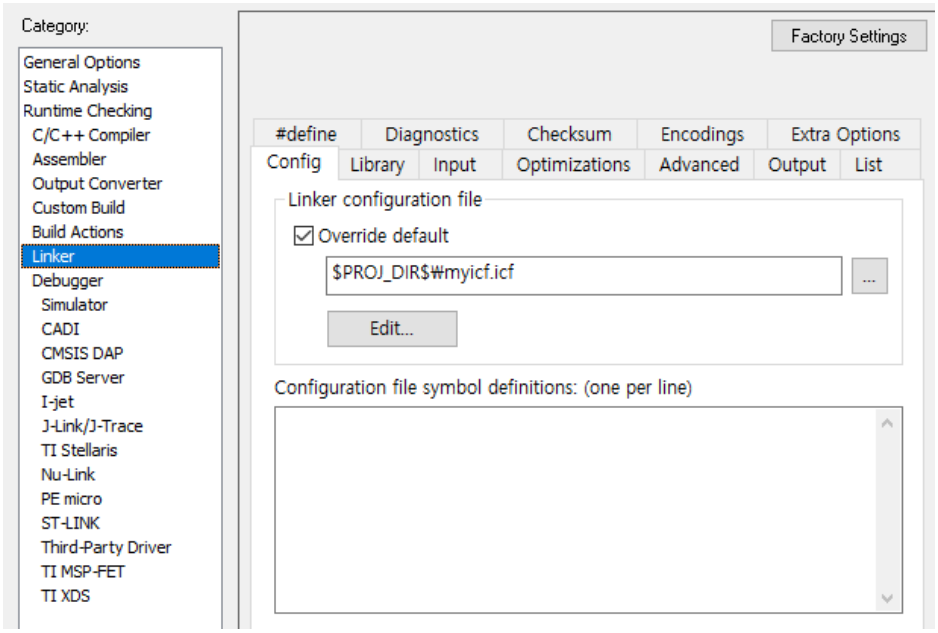
Main 함수에서 port configuration설정을 해주고 while문을 통해 입력값과 출력값을 제어한다. Port D에서 S1 입력이 들어오면 PC8에 신호를 주고, S2입력이 들어오면 LED에 신호를 준다. 입력값이 없으면 출력값을 주지 않는다. delay()함수는 수업자료의 것을 그대로 사용했다.

```
/*###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;// TODO
define symbol __ICFEDIT_region_ROM_end__   = 0x08080000;// TODO
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;// TODO
define symbol __ICFEDIT_region_RAM_end__   = 0x20008000;// TODO
/*-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__     { };

initialize by copy { readwrite };
do not initialize  { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region   { readonly };
place in RAM_region   { readwrite,
                        block CSTACK, block HEAP };
```

<그림8. Scatter File>

2.3. 스캐터 파일을 통해 원하는 메모리 위치에 프로그램 다운로드 확인.

ROM을 Flash의 시작주소인 0x08000000부터 0x80000만큼의 크기로 할당했고, RAM을 SRAM의 시작주소인 0x20000000부터 0x8000의 크기만큼 할당했다.



<그림 9. Scatter file link>

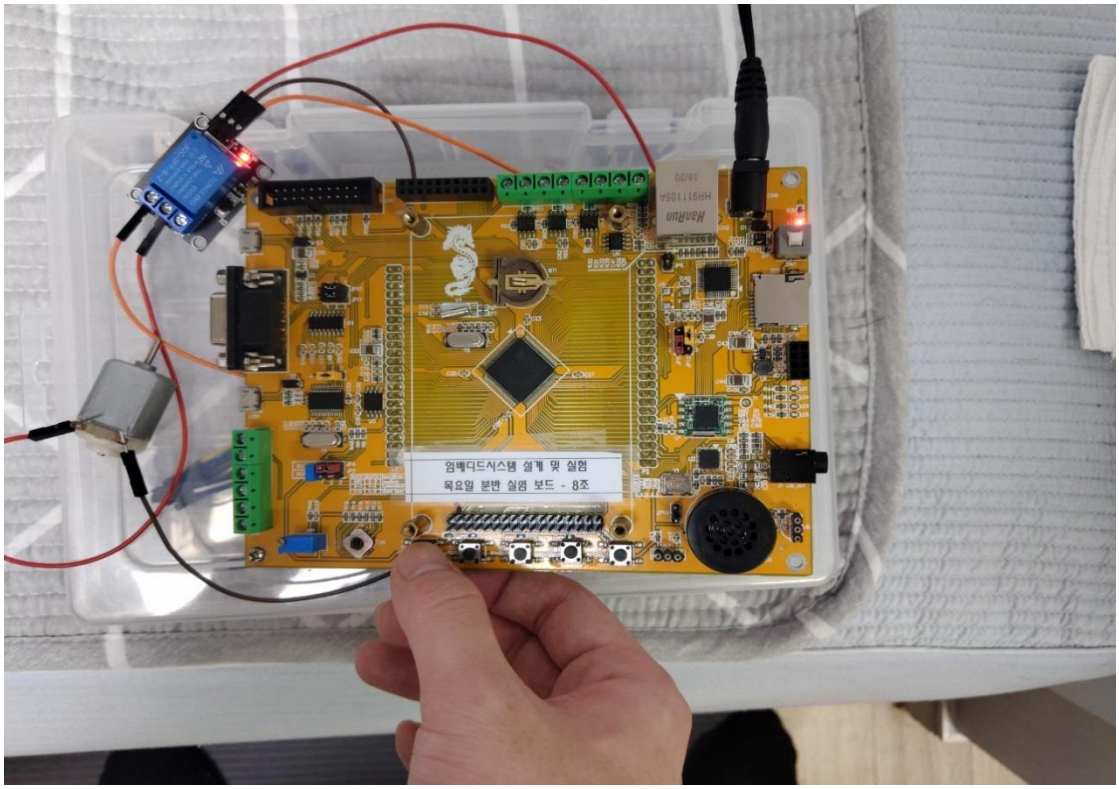
프로젝트 옵션에서 프로젝트 위치내의 myicf.icf 파일을 링크해줬다.

```
*******************************************************************************
*** PLACEMENT SUMMARY
***

"A0":  place at address 0x800'0000 { ro section .intvec };
"P1":  place in [from 0x800'0000 to 0x808'0000] { ro };
define block CSTACK with size = 4K, alignment = 8 { };
define block HEAP with size = 4K, alignment = 8 { };
"P2":  place in [from 0x2000'0000 to 0x2000'8000] {
          rw, block CSTACK, block HEAP };

No sections matched the following patterns:

  rw  in "P2"
```

<그림 10. 메모리 맵에서 할당된 메모리 확인>

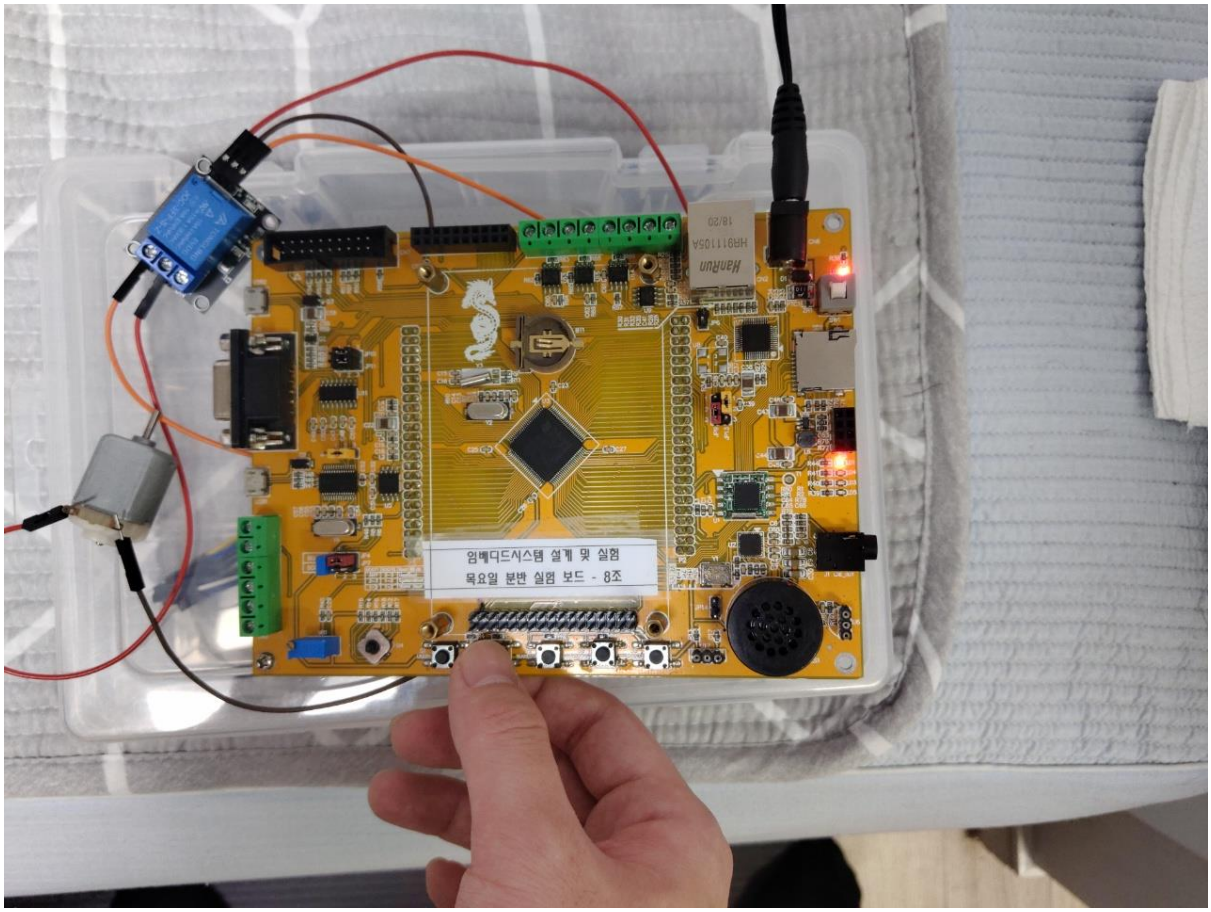
프로젝트의 memory map 파일을 봤을 때 스캐터 파일에서 할당한 위치에 프로그램이 다운로드 되어 있음을 확인할 수 있다. 또한 연결을 끊고 보드를 다시 실행해도 프로그램이 정상적으로 동작함을 확인할 수 있었다.

3. 실험 결과



<그림 11. S1을 눌렀을때>

S1 버튼을 누르면 PD8에 연결된 릴레이보드에 신호가 들어가 릴레이보드의 LED가 켜진 것을 확인할 수 있다.



<그림 12. S2를 눌렀을때>

S2를 누르면 PD7의 LED가 켜진 것을 확인할 수 있다.

4. 결론 및 느낀점

- Scatter file로 flash memory를 이용해 보드를 다시 기동시켰을 때 보드에 올라간 펌웨어가 유지될 수 있음을 알았다.

- 릴레이 모듈의 구조를 이해하고 보드와 릴레이 모듈을 연결해 신호를 주는 방법을 알았다.

- main함수가 polling 방식으로 입력을 받기 때문에 딜레이 함수의 시간동안 새롭게 신호를 읽을 수 없음을 알았다.