임베디드 시스템 설계 및 실험
4주차 실험 결과 보고서

조 : 8조

조원 : 서진욱, 이승민, 하연지, 하태훈

● 실험목표

1. 스캐터 파일의 이해 및 플래시 프로그래밍

- 스캐터 파일 코드 분석
- 스캐터 파일 업로드

2. 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작

- 전자기유도원리 이해하고 릴레이 모듈 사용
- 릴레이모듈 보드와 연결 및 실험진행

3. 폴링 방식의 이해.

- interrupt와 polling 방식 비교분석

● 개념설명

1. 플로팅, pull up, pull down

- 플로팅(Floating)

→ 스위치가 연결되지 않은 상태에서 전류가 흐르는지 아닌지 알 수 없는 상태

→ 어떤 전자 부품이나 회로 요소가 어떤 전압이나 전류와 연결되지 않고 떠 있는 상태

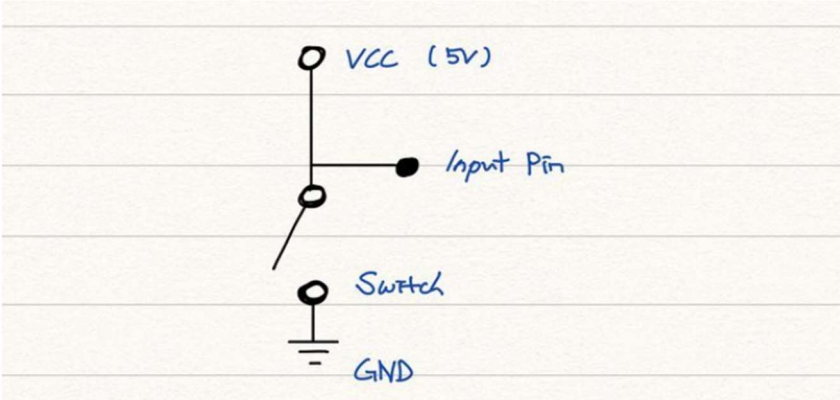

그림 1. 플로팅(Floating) 현상

- Pull up

→ 저항을 앞에 붙여줘 플로팅 현상을 해결하는 방법

→ 스위치 = open : Input pin으로 전류가 흐르고 전원 전압과 같은 전압(HIGH) 걸림

→ 스위치 = close : 모든 전류는 GND로 흐르고 Input pin에는 0V(LOW)전압이 걸림

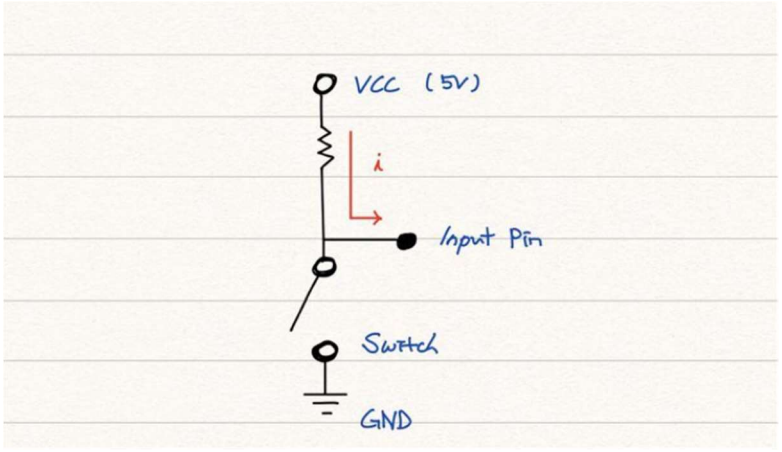

그림 2. Pull up 저항

- Pull down

→ 아래에 저항을 연결하는 방식(Pull up과 반대)

→ 스위치 = open : 전류가 흐르지 않고, Input pin엔 0V전압(LOW) 걸림

→ 스위치 = close : 아래의 저항에 의해 전류가 모두 Input pin쪽으로 흐르고, Input pin에는 전원 전압과 같은 전압(HIGH)이 걸리게 됨

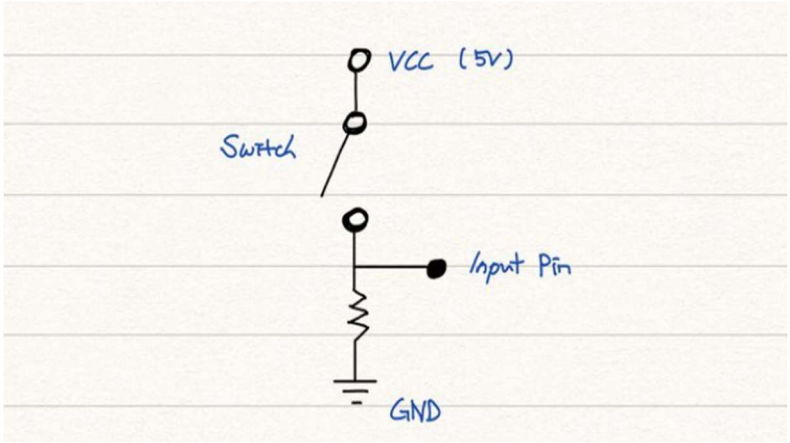

그림 3, Pull down 저항

2. 스캐터 파일(scatter file)

- 텍스트 파일 내에 기술된 설명을 사용해 이미지의 메모리 맵을 링커에 지정할 수 있게 해주는 도구
- 펌웨어 빌드, 링킹, 프로그램 실행 등에서의 메모리 매핑을 정의하는 파일
- 프로그램 메모리 매핑과 여러 시스템 데이터의 메모리 매핑 관리를 위해 사용

3. 릴레이 모듈

- 전기 신호에 의해 제어되는 전기스위치
  - 주로 저전압 마이크로컨트롤러나 컨트롤시스템이 고전압 혹은 고전류의 회로를 컨트롤하기 위해 사용됨
- 전자 자석과 일련의 기계적인 접점으로 구성됨

  → 전자 자석이 극성이 될 때, 접점은 한 위치(예: 열림)에서 다른 위치(예: 닫힘)로 전환됨

그림 4. 릴레이 모듈

● 실험 진행 및 각 과정별 이론

1. 스캐터 파일을 통해 플래시 메모리에 프로그램 다운로드

i) 스캐터 파일(stm32f107xc.icf) 수정을 통해 ROM, RAM 크기 수정

- ROM 크기 : 0x80000
- RAM 크기 : 0x8000
- 스캐터 파일 내 "Memory Regions" 부분에서 memory 크기에 맞게 수정
  - ROM 크기를 수정할 때 __ICFEDIT_region_ROM_end__ 값을 start인 0x08000000에서 0x80000을 더한 값인 0x08080000으로 설정
  - RAM 크기를 수정할 때 __ICFEDIT_region_RAM_end__ 값을 start인 0x20000000에서 0x8000을 더한 값인 0x20008000으로 설정

```
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0803FFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__   = 0x2000FFFF;
```

코드 1. 수정 전 메모리 주소

```
/*-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;
```

코드 2. 수정 후 메모리 주소

2. 릴레이 모듈 및 LED, Button에 대한 레지스터 및 주소 정의(Code 구현)

i) RCC 초기화 및 릴레이 모듈을 활성화 하기 위해 사용할 핀 활성화

- 버튼 핀(PB10, PC4, PC13) 정의 및 초기화
- PD2, PC3 핀을 릴레이 모듈의 트리거로 사용할 예정이므로 해당하는 포트 초기화

```
RCC_APB2_ENR = 0x3C;
```

코드 3. RCC 초기화

```
GPIOD_CRL = 0x44444444;
GPIOC_CRH = 0x44444444;
GPIOC_CRL = 0x44444444;
GPIOB_CRH = 0x44444444;


GPIOD_CRL = 0x00003300; // LED(PD2, PD3)

GPIOD_BSRR = 0x00000000; // all led reset

GPIOD_BRR |= 0x9C;
```

코드 4. 버튼 핀 input 설정 및 릴레이 모듈의 트리거 핀(PD2, PD3)을 50MHz output으로 설정 및 초기화

ii) 작동구현 코드 작성

※ 무한 반복 되어야 하므로 while(1)문 안에 코드 작성

※ 각 버튼의 IDR값 변화를 인식 하고 필요한 릴레이 모듈의 트리거 핀의 BSRR값 변경

- <코드 6> : 버튼1(PC4)를 누르면 두 릴레이 모듈이 모두 1초간 작동 후 꺼짐
- <코드 7> : 버튼2(PB10)을 누르면 PD2에 연결된 릴레이 모듈이 1초간 작동 후 꺼짐
- <코드 8> : 버튼3(PB13)을 누르면 PD3에 연결된 릴레이 모듈이 1초간 작동 후 꺼짐

- <코드 5> : 버튼이 눌렸을 때 핀을 1초간 활성화 시킨 후 다시 비활성화 해야 하므로 delay 함수 선언 필요
- 각 코드의 구조는 버튼이 눌리면 핀을 활성화 후 1초 딜레이가 지나면 핀을 비활성화 하는 방식으로 구현 되어있음

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

코드 5. 릴레이 모듈을 약 1초간 작동 시키기 위한 delay 함수 선언

```
while(1) {
  // press button1

  if (~GPIOC_IDR & 0x10){
    GPIOD_BSRR |= 0x240000; //enable Relay 1
    GPIOD_BSRR |= 0x80000; //enable Relay 2
    delay();
    GPIOD_BSRR |= 0x04; //disable Relay 1
    GPIOD_BSRR |= 0x08; //disable Relay 2
  }
```

코드 6. 버튼 1이 눌리면 PD2, PD3를 1초간 활성화

```
  // press button2
  if (~GPIOB_IDR & 0x400) {
    GPIOD_BSRR |= 0x240000;
    delay();
    GPIOD_BSRR |= 0x04;
  }
```

코드 7. 버튼 2가 눌리면 PD2를 1초간 활성화

```
  // press button3
  if (~GPIOC_IDR & 0x2000) {
    GPIOD_BSRR |= 0x80000;
    delay();
    GPIOD_BSRR |= 0x08;
  }


}
```

코드 8. 버튼 3이 눌리면 PD3를 1초간 활성화

3. 회로 구성

- 전원 인가 : 릴레이 모듈에 3.3V의 전원을 공급하기 위하여 STM32의 3V3을 브레드보드의 (+)에, GND를 브레드보드의 (-)에 연결
- 릴레이 모듈의 INPUT 연결 : 릴레이 모듈1, 2는 각각 PD2, 3에 대응되어 동작 시키고자 하므로 릴레이 모듈1은 PD2와, 릴레이 모듈2는 PD3과 연결

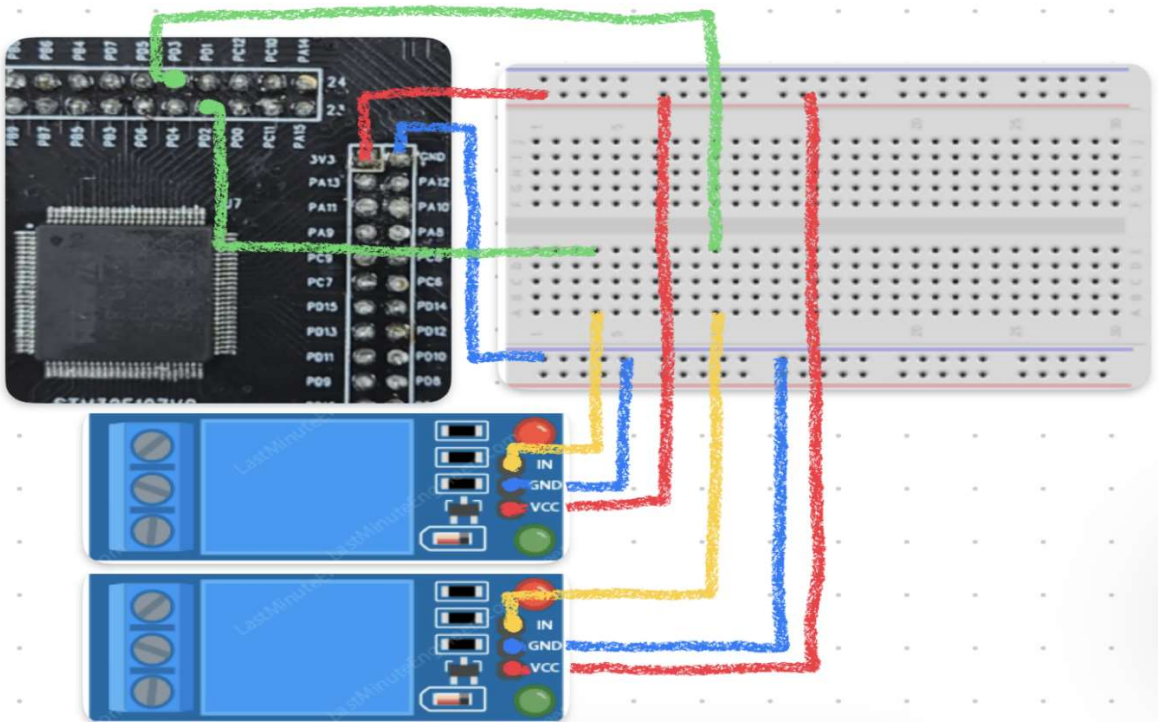

그림 5. 회로 구성 상황

4. LED, Button 및 릴레이 모듈 동작 확인

i) 초기

- 모든 LED off 상태

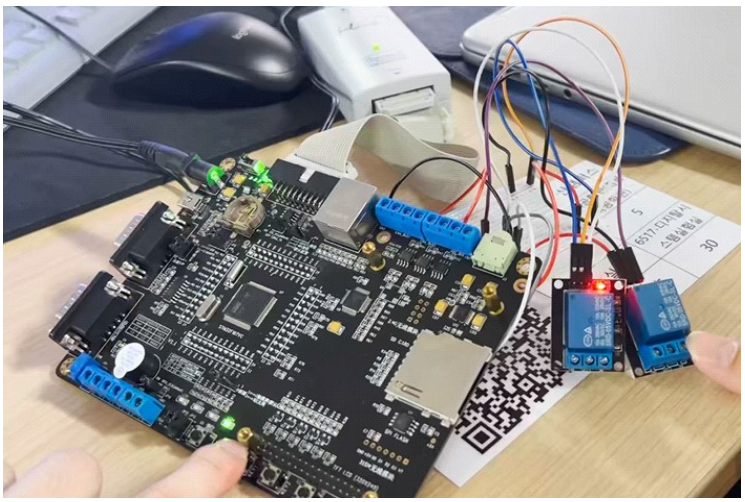

그림 6. 초기 상태(모든 LED off)

ii) 동작1

- button1 누름

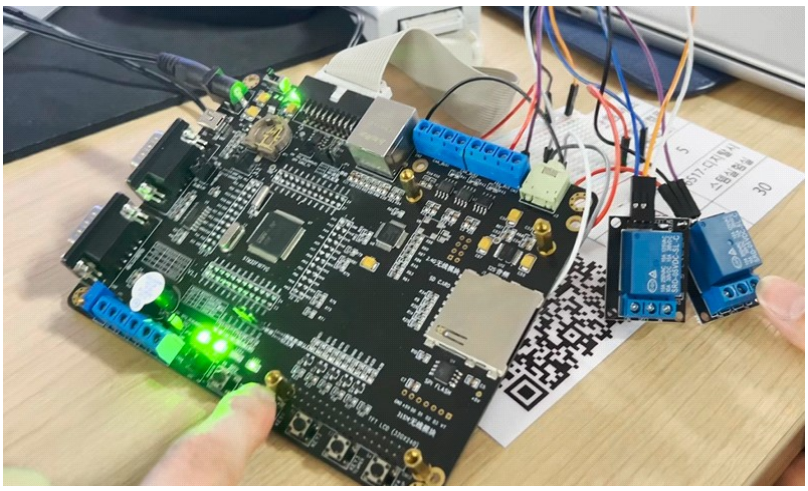

그림 7. button1 동작(LED1, LED2 켜졌다 꺼짐)

iii) 동작2

- button2

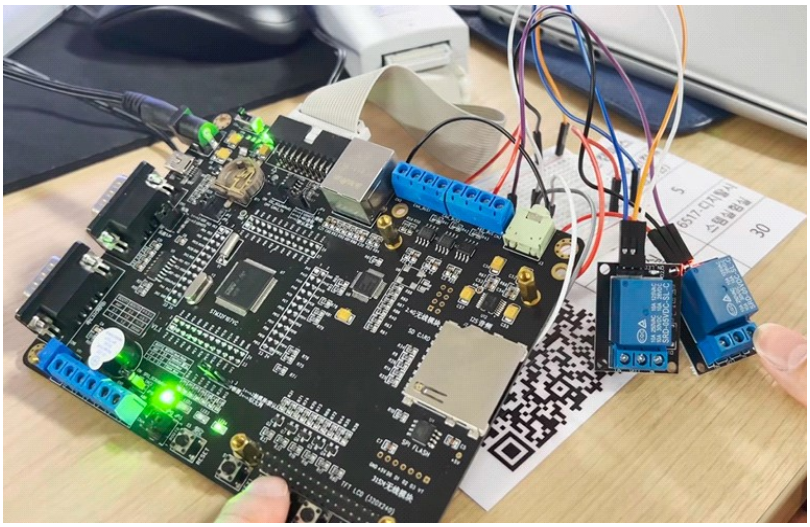

그림 8. button2 동작(LED1 켜졌다 꺼짐)

ii) 동작3

- button3

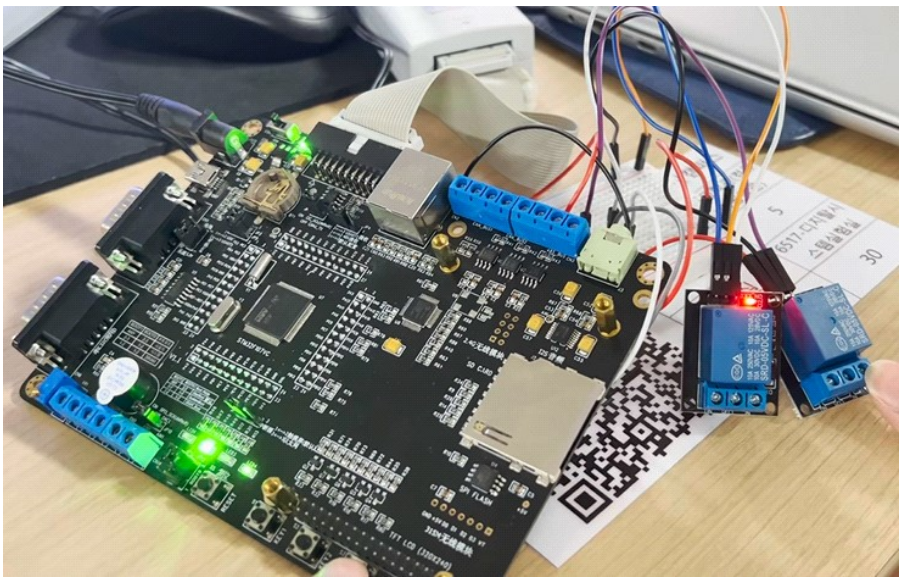

그림 9. button3 동작(LED2 켜졌다 꺼짐)

● 결론

- 이번 4주차 실습을 통하여, 스캐터 파일을 활용하여 플래시 메모리에 프로그램을 다운로드하여 메모리 매핑을 관리하고 프로그램의 구조를 최적화할 수 있음을 확인할 수 있었다.
  - 스캐터 파일을 사용하여 프로그램의 매핑을 정의함으로써, 프로그램의 각 부분이 어떤 메모리 주소 범위에 위치하는지를 확인할 수 있었다.
  - 스캐터 파일을 통해 ROM, RAM 크기를 의도하는 바에 맞게 수정하여 효율적으로 메모리 할당을 조정할 수 있음을 확인할 수 있었다.
- 플로팅 현상과 Pull up, Pull down의 개념을 이해하고 이를 회로 구성하는 것에 있어 적절하게 활용할 수 있었다.
- 릴레이 모듈을 활용하여 LED와 버튼 동작을 제어하는 과정을 수행함으로써, 버튼 조작을 이용하여 릴레이 모듈에 대한 활성화를 조작하여 LED를 켜고 끄는 것을 확인할 수 있었다.
- 추가적으로 주소 초기화 시, 0x44444444로 초기화하는 것이 아닌 단순한 값으로 초기화하는 방법도 고려할 수 있다는 것을 확인할 수 있었다.