임베디드 시스템 설계 및 실험
003분반 4주차 예비 발표

6조 김선규 이동현 이주승 이지현 최세희

Contents

Scatter file 이란?

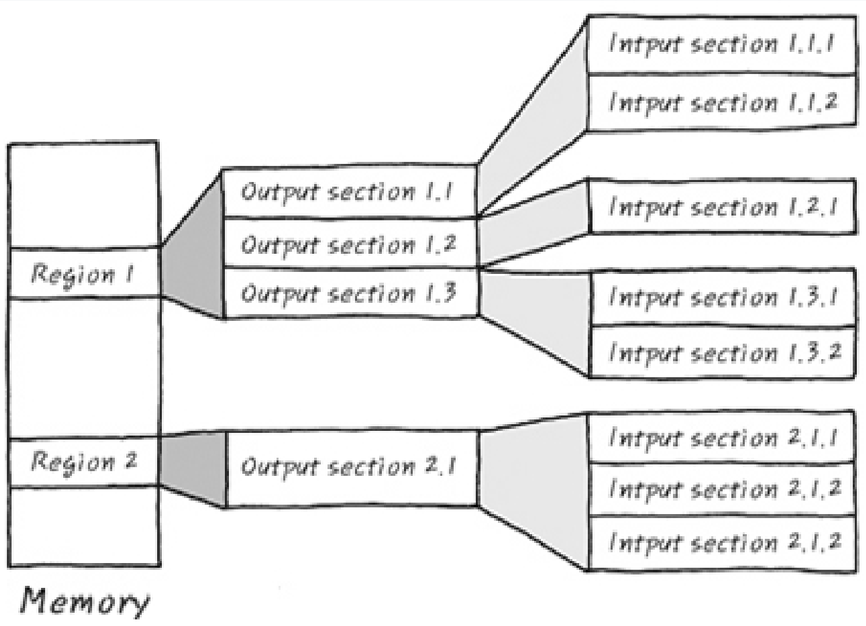
- 코드와 데이터를 Memory의 필요한 위치에 배치하는 script (확장자 .scf)
ARM linker는 이 Scatter file을 통해 (scatter-loading mechanism)
ROM, RAM의 위치 정보, 즉 image의 메모리 맵을 지정할 수 있다.

- C startup file을 사용할 때 linking을 하기 위해 요구된다.

- Scatter file에는 아래의 region들이 포함된다.

Code (read-only data, execute-only data), RAM (read/write data, zero-initialized data),
Stack, Heap, Stack seal (Armv8-M/v8.1-M), CMSE veneer (Armv8-M/v8.1-M)

Scatter file 이 필요한 이유



1. ARM linker가 image file을 생성할 때
RO, RW, ZI 및 기타 데이터의 저장 주소를 할당하는 방법을 지정할 수 있다.

2. 일반적인 상황에서는 Scatter file이 필요하지 않지만,
특정 주소에 데이터를 넣고 싶을 때 Scatter file은 중요한 역할을 한다.

ex) ARM microcontroller chip LPC2378은 multiple discrete SRAM을 가진다.
일반적인 RAM의 크기는 32KB인데, 32KB는 많은 데이터를 저장하기에 부족하다.
이때 Scatter file을 작성해 USB SRAM의 특정 .C에 RW 데이터를 넣음으로써 해결할 수 있다.

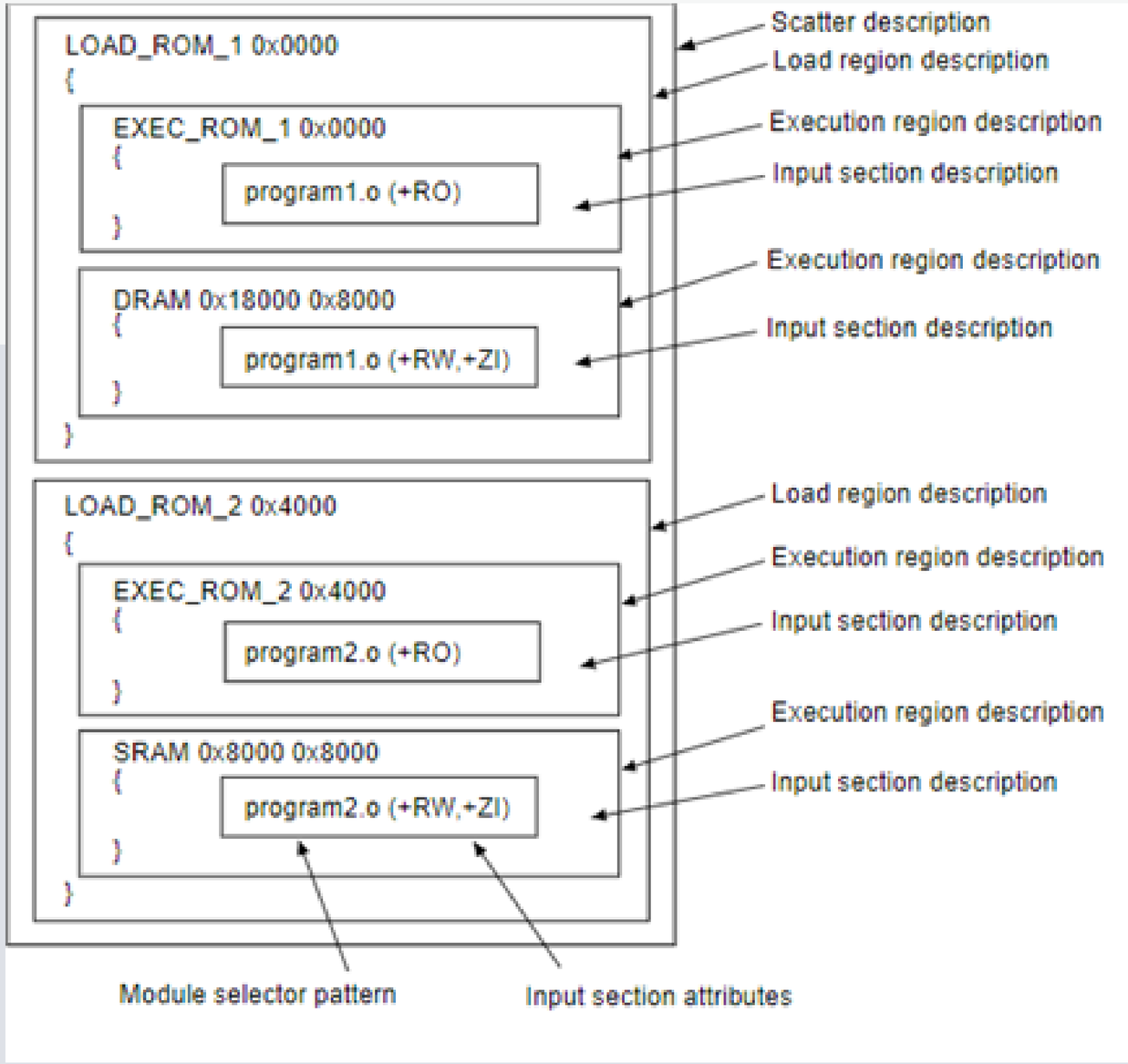
3. image 구성 요소의 grouping 및 배치를 완벽하게 제어할 수 있다.

- 여러 개의 input section이 output section 및 region으로 grouping 된다.
- linker를 통해 최종 실행 image를 만들 때는 XIP가 가능한 메모리에는 code를,
RW가 가능한 영역에는 data를 매핑시켜 배치해야 한다. 이 역할을 Scatter loading이 하게 된다.

*XIP : 메모리 상에서 직접 software를 실행할 수 있는 기술, byte나 word 등의 단위로 random access가 가능해야 함

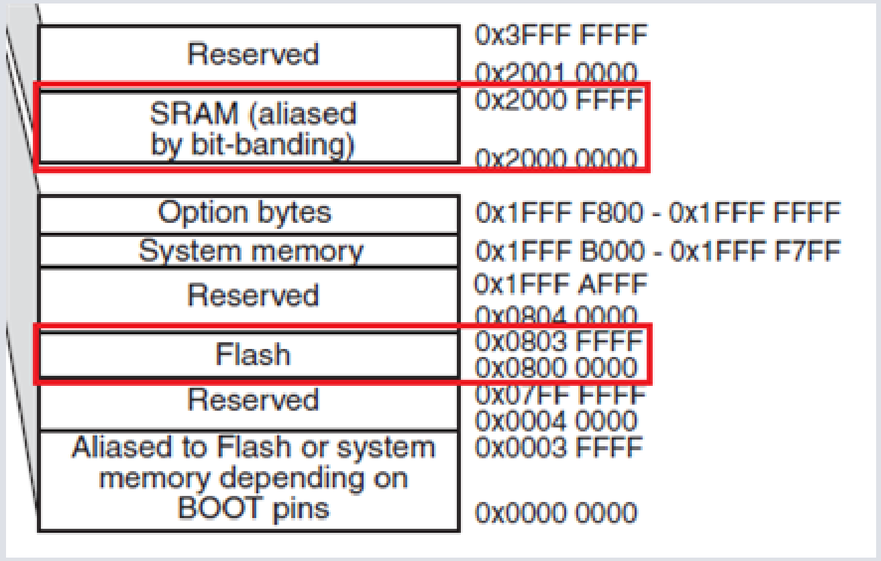
Scatter file 코드 분석

Scatter file은 하나 이상의 load region을 포함하고,
각각의 load region은 하나 이상의 execution region을 포함한다.



[Scatter File의 구성 요소]

Scatter file 코드 분석

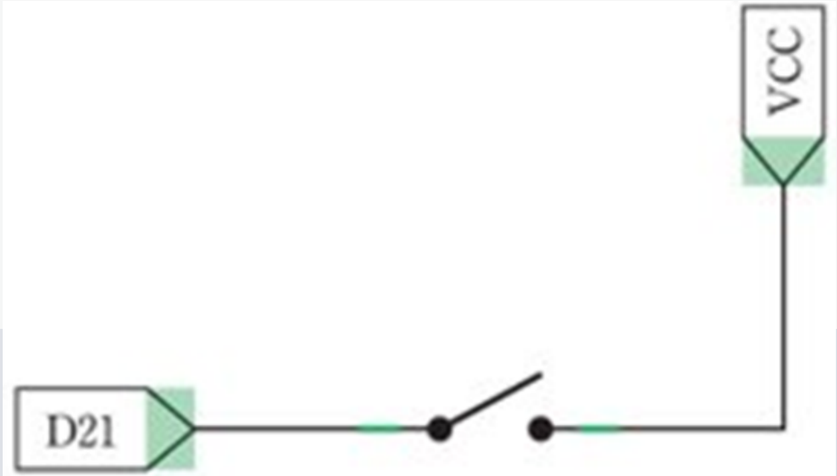


[Flash Memory(ROM)와 SRAM의 주소]

<stm32 scatter file example>

```
1   // Datasheet 에서 ROM과 RAM 주소를 참고한다.
2
3   // (1) Load region description - region 크기만큼 region을 load함
4   // LR_IROM1 0x08000000 0x00008000 에서
5   // 첫 번째는 load section의 시작주소, 두 번째는 load section의 크기이다.
6   LR_IROM1 0x08000000 0x00008000
7   {
8       // (2) Execution region description - load 주소와 execution 주소는 동일함
9       ER_IROM1 0x08000000 0x00008000
10      {
11          // (3) Input section description
12          // - 모든 .o 코드를 RESET Section의 처음(First)에 위치시킨다
13          *.o(RESET, +First)
14          // ARM Library section 을 root section 에 자동 배치시킨다
15          *(InRoot$$Sections)
16          // 나머지 code와 RO data를 ROM에 위치시킨다
17          .ANY (+RO)
18      }
19
20      // RAM Section에서는 SRAM의 시작주소 0x20000000를 사용
21      RW_IRAM1 0x20000000 0x00008000
22      {
23          // RW 와 ZI data section을 RAM 에 위치시킨다
24          .ANY (+RW +ZI)
25      }
26  }
27
```

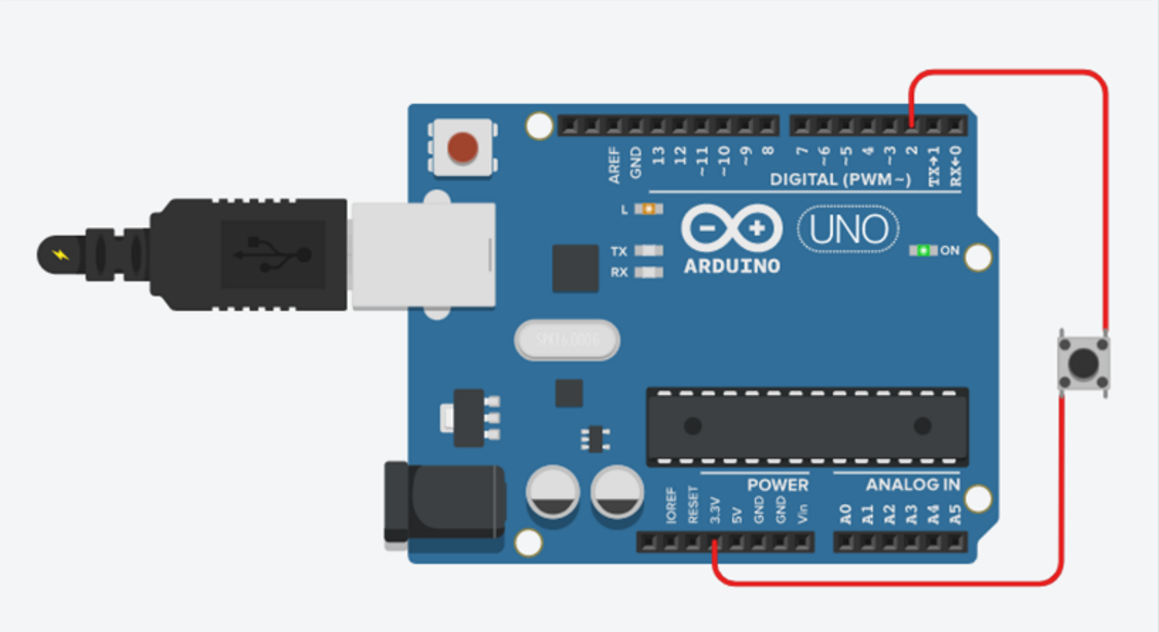
Floating



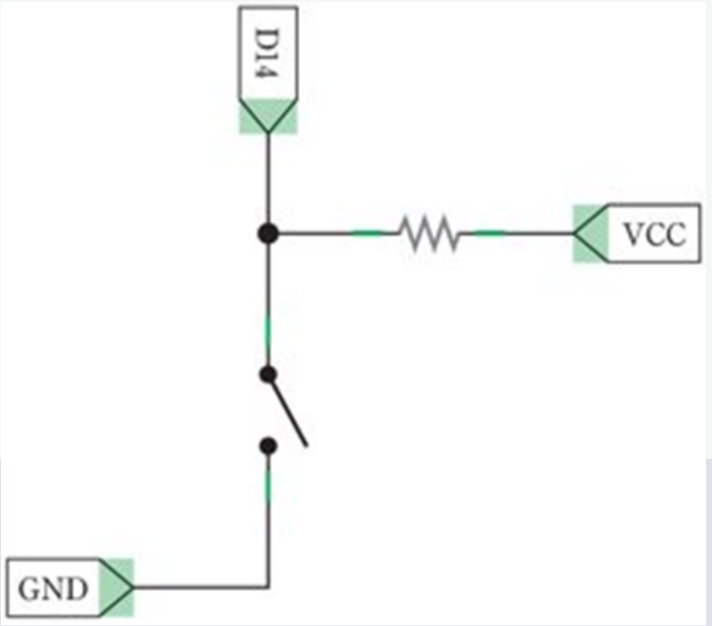
- 버튼을 누르면 전류가 흐르나, 버튼을 누르지 않은 상태에서는 주변의 노이즈로 인해 신호가 불안해져 오작동이 생기는 현상

- 1도 아니고 0도 아닌 떠있는(floating) 상태

- 풀업 저항, 풀다운 저항을 통해 해결 가능

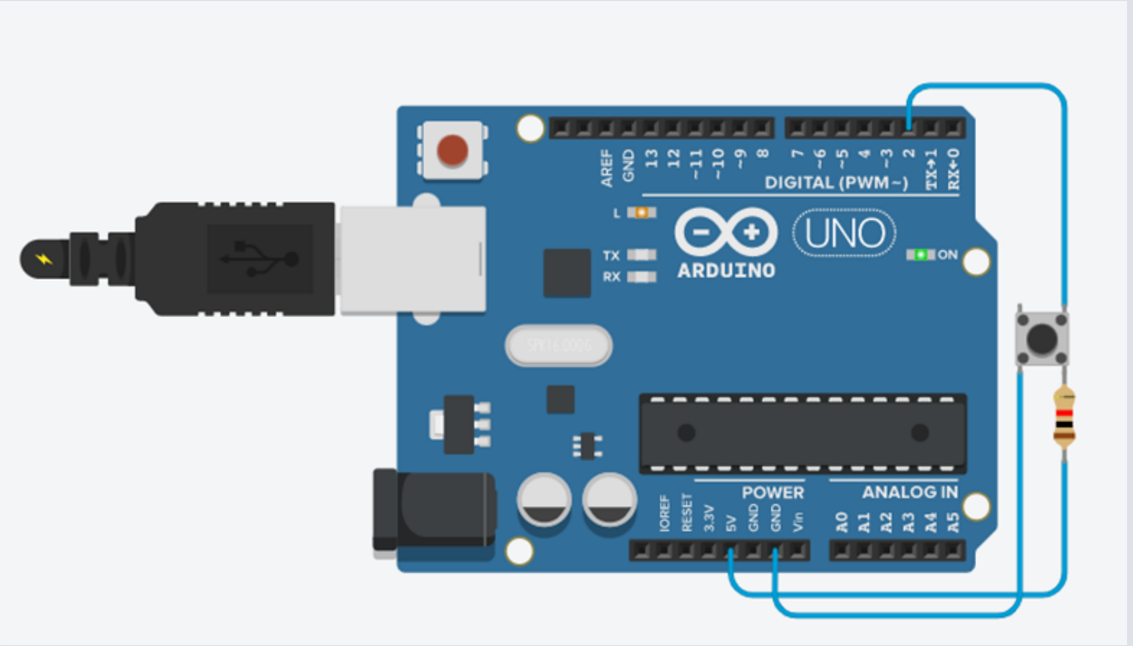


풀업(pull-up) 저항

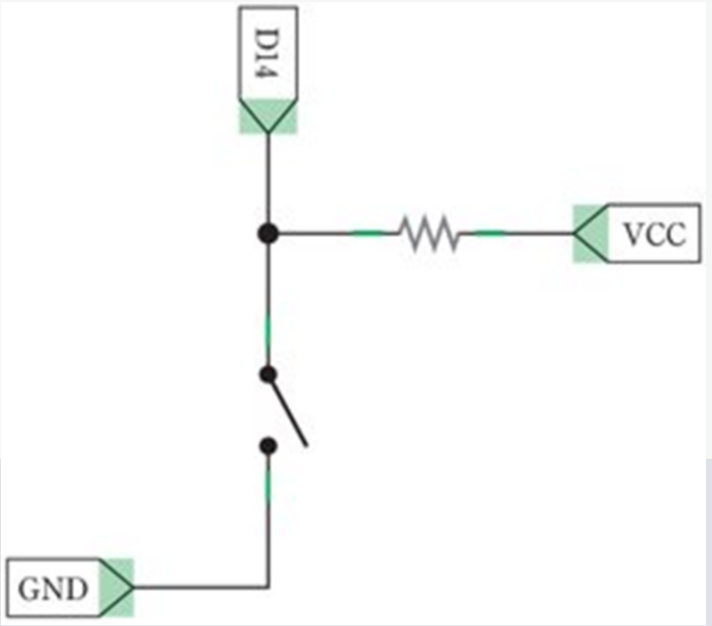


- 입력 핀과 VCC사이에 저항을 연결하는 방법

- 버튼을 누르면 0(LOW), 버튼을 누르면 1(HIGH)

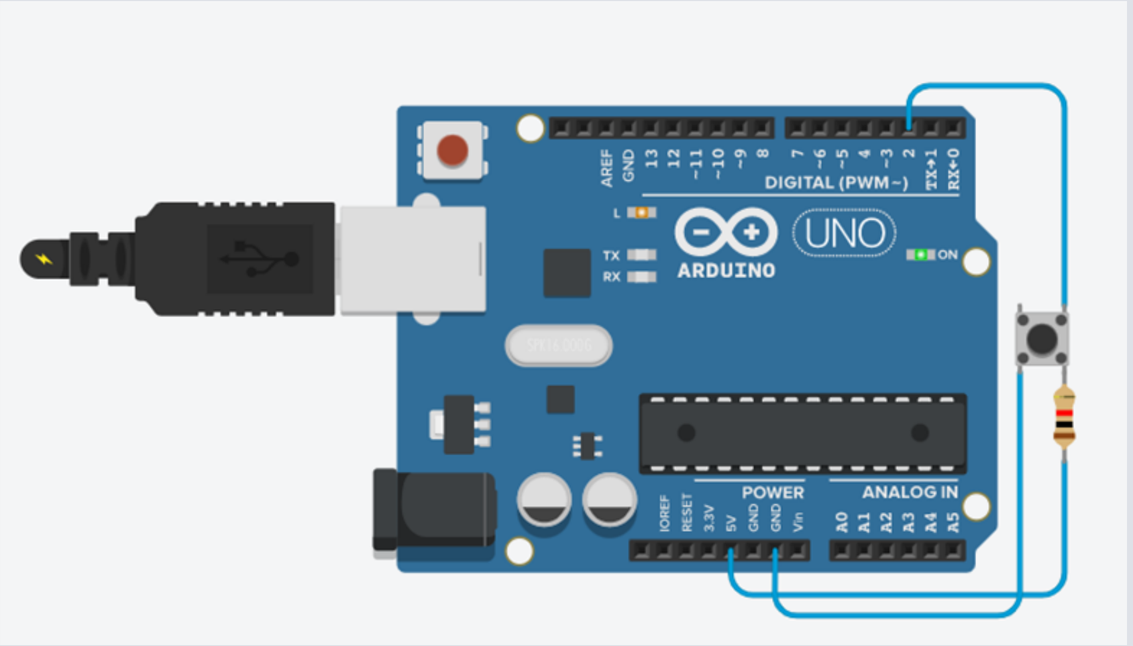


풀다운(pull-down) 저항



- 입력 핀과 GND사이에 저항을 연결하는 방법

- 버튼을 누르면 1(HIGH), 버튼을 누르면 0(LOW)



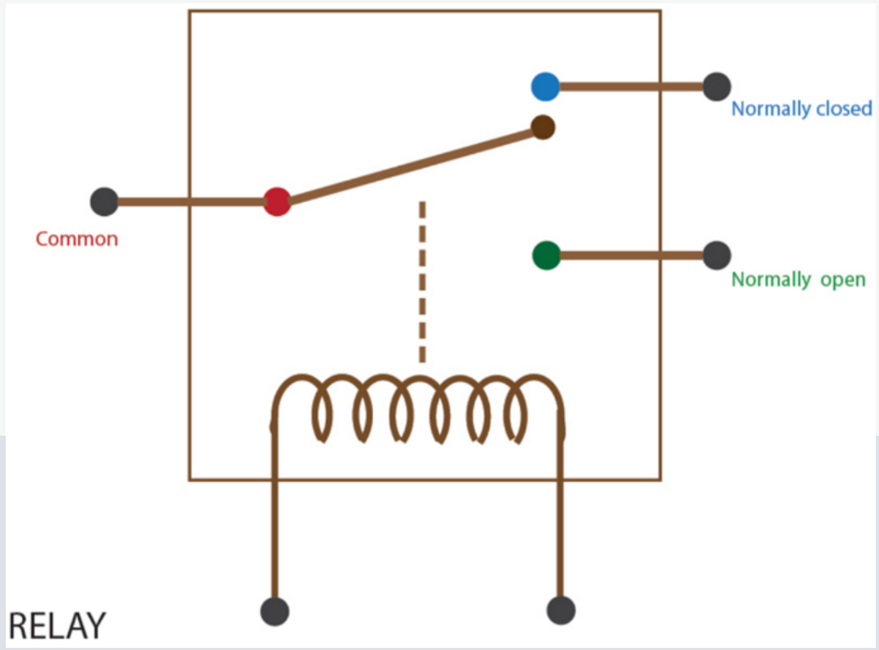
floating, pull up, pull down 비교

저항연결	버튼 누르지 않음	버튼 누름
Floating	플로팅 상태(미결정 상태)	1
Pull up	0	1
Pull down	1	0

인터럽트와 폴링 방식 비교

	Polling	Interrupt
의미	일정 주기마다 스레드를 돌면서 시그널이 들어왔는지 확인	인터럽트가 들어오면 실행하고 있던 것을 멈추고 인터럽트에 대한 임무 수행 후 기존 동작을 재진행
장점	구현이 쉽다 우선순위 변경이 용이하다	반응 시간이 빠르다 시스템 부하가 적다
단점	정확한 타이밍에 시그널 발생을 확인하기 어렵다 시스템 부하가 많다	구현이 복잡하다

릴레이 모듈



- 스위치와 같은 역할. 수동인 스위치와 달리 전기 신호에 따라 동작.

- 낮은 전압 전류를 이용해 높은 전압 전류 기기를 제어할 때 쓰임.

- 'NO'는 normal open을 의미. 스위차가 열려있는 상태가 기본상태로, 릴레이에 전류가 흐르면 스위치가 닫힘.

- 'NC'는 반대의 경우로, 평소에는 닫혀있고 릴레이에 전류가 흐르면 스위치가 열림.

참고 자료

- https://developer.arm.com/documentation/101754/0618/armlink-Reference/Scatter-loading-Features?lang=en
- http://recipes.egloos.com/5013253
- https://arm-software.github.io/CMSIS_5/Core/html/linker_sct_pg.html#linker_sct_preproc_sec

감사합니다.