

# 2021년 2학기 임베디드 시스템 설계 및 실험

- 4주차 보고서 -

004분반 9조

## 개요

이번 실험에서는 스캐터 파일을 이용한 주소할당과 릴레이 모듈을 통한 모터조작에 대하여 알아보았다.

## 목표

- 스캐터 파일의 이해 및 플래시 프로그래밍
- 릴레이 모듈의 이해 및 임베디드 펌웨어를 통한 동작
- 폴링 방식의 이해

## 세부실험내용

- 참고자료를 통해 당 레지스터 및 주소에 대한 설정 이해
- 스캐터 파일을 통해 플래시 메모리에 프로그램 다운로드
- 플래시 메모리에 올려진 프로그램 정상적인 동작 확인

## 개념

- Floating : 전압이 0 또는 1로 판별할 수 없는 상태
- Pull Up : VCC에 저항을 연결
- Pull Down : GND에 저항을 연결
- Polling : 프로그램의 변화를 지속적으로 읽어들여 변화를 알아채는 방법
- Relay Module : 전자기유도원리를 이용한 스위치 역할을 하고 릴레이를 제어하는 모듈
- Scatter file : 메모리 계층 구조에 따른 속도차이에 의한 성능저하를 완화시키기 위해서 상위 계층 메모리에 공간을 할당하여 우선 배치하도록 설정하는 파일

# 실험과정

## - 세팅

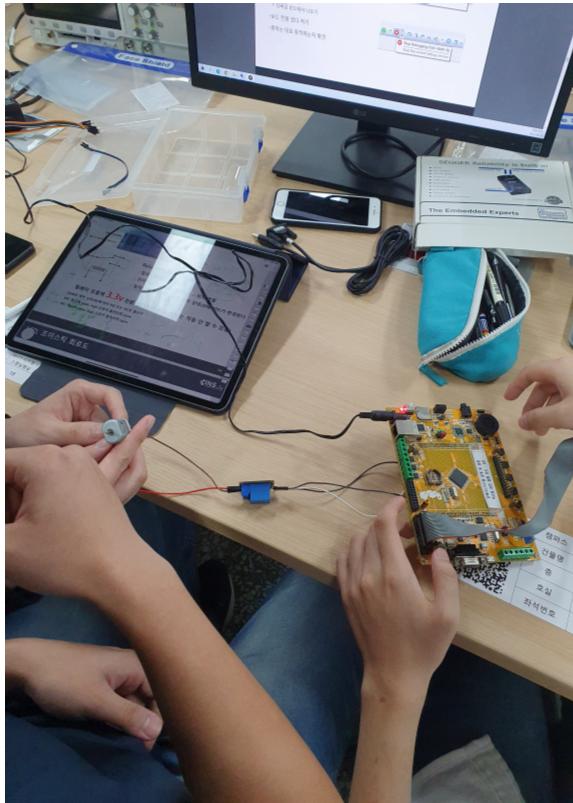


그림 1) '보드 - J-Link - PC' 순으로 연결

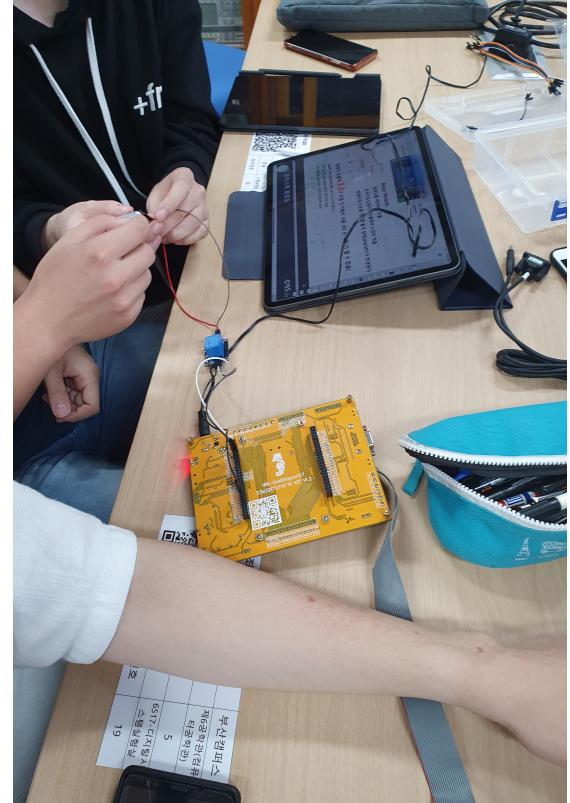


그림 2) '보드 - 릴레이 모듈' 연결

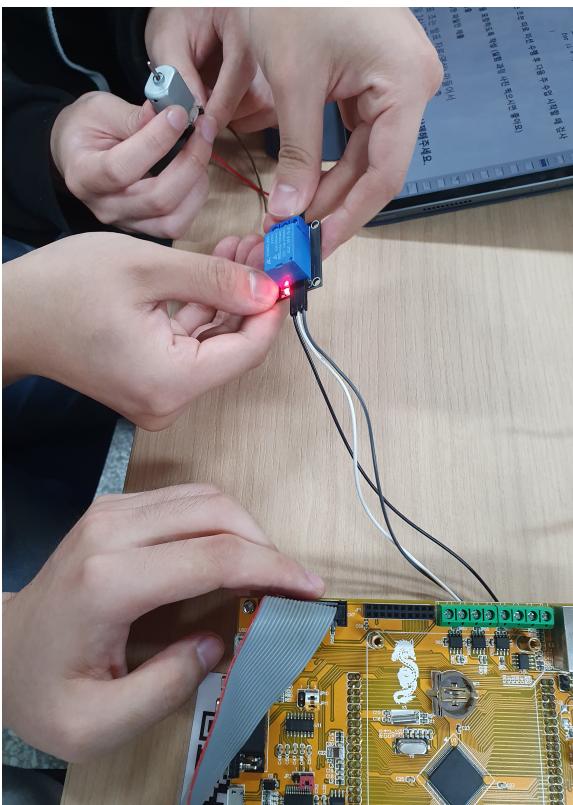


그림 3) 릴레이 모듈 동작 확인

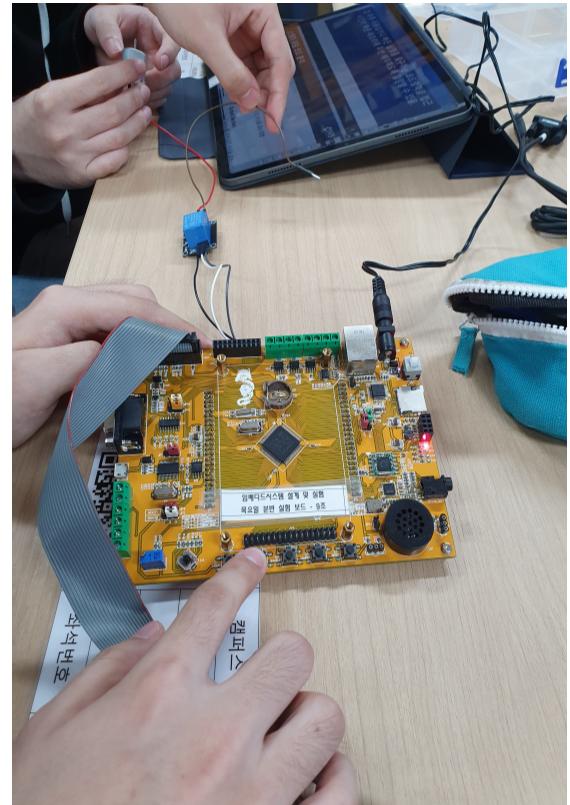


그림 4) LED 동작 확인

## - 코드

```
1 #include "stm32f10x.h"
2
3 void delay() {
4     int i;
5     for(i=0; i<10000000; i++) {}
6 }
7
8 int main(void) {
9     // GPIO 설정을 위한 InitTypeDef 변수 선언
10    GPIO_InitTypeDef GPIO_InitInput;
11    GPIO_InitTypeDef GPIO_InitLEDOutput;
12    GPIO_InitTypeDef GPIO_InitRelayOutput;
13
14    // Input(버튼) 설정
15    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
16    // Port D의 11,12번 핀에 대하여 설정
17    GPIO_InitInput.GPIO_Pin = GPIO_Pin_11|GPIO_Pin_12;
18    // Pull-Up 모드로 설정(버튼을 누르면 Low, 가만히 두면 High)
19    GPIO_InitInput.GPIO_Mode = GPIO_Mode_IPU;
20
21    // Output(릴레이모듈) 설정
22    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
23    // Port B의 8번 핀에 대하여 설정
24    GPIO_InitRelayOutput.GPIO_Pin = GPIO_Pin_8;
25    // Push-Pull 모드로 설정
26    GPIO_InitRelayOutput.GPIO_Mode = GPIO_Mode_Out_PP;
27    // 저전력 모드인 2MHz로 설정
28    GPIO_InitRelayOutput.GPIO_Speed = GPIO_Speed_2MHz;
29
30    // OutPut(LED) 설정
31    // Port D의 7번 핀에 대하여 설정
32    GPIO_InitLEDOutput.GPIO_Pin = GPIO_Pin_7;
33    // Push-Pull 모드로 설정
34    GPIO_InitLEDOutput.GPIO_Mode = GPIO_Mode_Out_PP;
35    // 저전력 모드인 2MHz로 설정
36    GPIO_InitLEDOutput.GPIO_Speed = GPIO_Speed_2MHz;
37
38    // 위에서 지정한 설정대로 각 포트를 초기화
39    GPIO_Init(GPIOC, &GPIO_InitInput);
40    GPIO_Init(GPIOB, &GPIO_InitRelayOutput);
41    GPIO_Init(GPIOB, &GPIO_InitLEDOutput);
42
43    // 초기화 이후 아래 동작을 계속 반복
44    while (1) {
45        uint8_t relayInputPin = GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_11);
46        uint8_t ledInputPin = GPIO_ReadInputDataBit(GPIOB,GPIO_Pin_12);
47
48        // S1 버튼을 누를 경우(Pull-up으로 0이 인가될 경우)
49        // 릴레이 모듈(PB8)에 delay동안 신호를 줌
50        if(relayInputPin == Bit_RESET) {
51            GPIO_SetBits(GPIOB,GPIO_Pin_8);
52            delay();
53            GPIO_ResetBits(GPIOB,GPIO_Pin_8);
54        }
55
56        // S2 버튼을 누를 경우
57        // LED(PD7)에 delay동안 신호를 줌
58        if(ledInputPin == Bit_RESET) {
59            GPIO_SetBits(GPIOB,GPIO_Pin_7);
60            delay();
61            GPIO_ResetBits(GPIOB,GPIO_Pin_7);
62        }
63    }
64
65    return 0;
66 }
```

main.c

```
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__ = 0x0807FFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x20007FFF;
```

scatter file

## 결론

스캐터 파일을 이용한 RAM과 ROM에서의 주소 할당과 릴레이 모듈을 통한 모터 제어를 실습해보았다.  
스캐터 파일을 단순히 메모리 계층구조에 따른 속도저하 때문에 사용한다고 생각하였는데  
그 뿐만이 아니라 한 번에 많은 처리를 수행하는 FPGA 보드 특성상 여러 포트를 사용하게 되면  
그 포트들의 핀들도 동시에 사용되게 되며 이는 다양한 에러를 발생시킬 수 있는데  
스캐터 파일로 주소를 할당함으로써 각 포트들을 구분하여 에러를 감소시킬 수 있다는 것을 알게 되었다.