
임베디드시스템설계 실습 (7)

Embedded System Design

**Real-Time Computing and Communications Lab.
Hanyang University**

VPOS 커널을 포팅하기 위한 준비

1. 커널 컴파일 + 커널 이미지를 RAM에 적재
2. Startup code 작성
3. UART 설정
4. TIMER 설정
5. Hardware Interrupt Handler 구현
(1) UART Interrupt
6. Software Interrupt Entering/Leaving Routine 구현
7. Timer Interrupt

목차

1. **Timer Interrupt (C Code)**
2. **Timer Interrupt (Assembly)**
3. **GPIO (LED)**

TIMER INTERRUPT (C CODE)

Timer Interrupt

□ 목적

- 일정 주기마다 특정 이벤트를 실행시키기 위해 사용
- 실습에서는 스케줄러를 호출하여 **Context Switching**을 함

□ Timer4 사용

- ES_Practice_5.pdf 참고

S5PC100에서의 Timer

□ Vectored Interrupt Controller에서 Timer4

Timer4는 VIC0의 25번

	26	System Timer	System Timer Interrupt
	25	TIMER4	Timer 4 Interrupt
	24	TIMER3	Timer 3 Interrupt
	23	TIMER2	Timer 2 Interrupt
	22	TIMER1	Timer 1 Interrupt
	21	TIMER0	Timer 0 Interrupt
	20	PDMA1	Peri DMA Interrupt
	19	PDMA0	Peri DMA Interrupt
	18	MDMA	M2M DMA Interrupt
	17	BATF	Battery Fault Interrupt
	16	EINT 16_31	EXT_INT[16] ~ [31]
	15	EINT15	EXT_INT[15]
	14	EINT14	EXT_INT[14]
	13	EINT13	EXT_INT[13]
	12	EINT12	EXT_INT[12]
	11	EINT11	EXT_INT[11]
	10	EINT10	EXT_INT[10]
	9	EINT9	EXT_INT[9]

VIC0
System, DMA,
Timer

레지스터 주소 정의

□ 위치

- `vpos/hal/include/vh_io_hal.h`

□ VIC0 관련 레지스터 주소 정의

- `VIC0INTSELECT`
- `VIC0INTENABLE`
- `VIC0INTENCLEAR`
- `VIC0SWIPRIORITYMASK`
- `VIC0VECTADDR25`

VPOS_kernel_main()

□ 소개

- VPOS 커널 데이터 구조체를 초기화
- 시리얼 장치와 타이머 등 하드웨어를 초기화
- 인터럽트 **enable**
- 부팅 메시지 출력
- 쉘 스레드 생성
- 스케줄러 호출하는 **VPOS_start** 루틴으로 진입

□ 소스 코드 위치

- vpos/kernel/kernel_start.c

```
void VPOS_kernel_main( void )
{
    pthread_t p_thread, p_thread_0, p_thread_1, p_thread_2;

    /* static and global variable initialization */
    vk_scheduler_unlock();
    init_thread_id();
    init_thread_pointer();
    vh_user_mode = USER_MODE;
    vk_init_kdata_struct();

    vk_machine_init();
    set_interrupt();

    printk("%s\n%s\n%s\n", top_line, version, bottom_line);

    /* initialization for thread */
    race_var = 0;
    pthread_create(&p_thread, NULL, VPOS_SHELL, (void *)NULL);
    pthread_create(&p_thread_0, NULL, race_ex_1, (void *)NULL);
    pthread_create(&p_thread_1, NULL, race_ex_0, (void *)NULL);
    pthread_create(&p_thread_2, NULL, race_ex_2, (void *)NULL);

    VPOS_start();

    /* cannot reach here */
    printk("OS ERROR: VPOS_kernel_main( void )\n");
    while(1){}
}
```


set_interrupt()

□ 위치

- vpos/kernel/kernel_start.c

□ 코드 추가

- `vh_timer_irq_enable(vh_TIMER4)` 함수 호출
 - `vh_TIMER4`는 '4'를 의미. 5개의 타이머 중 Timer4를 가리킴

```
void set_interrupt(void)
{
    // interrupt setting
    vh_serial_irq_enable();
    vh_timer_irq_enable(vh_TIMER4);
}
```

vh_timer_irq_enable()

□ 역할

- **VIC0**의 레지스터들을 설정하여 **Timer4 Interrupt**를 활성화 (**enable**)시키는 함수

□ 위치

- **vpos/hal/io/timer.c**

vh_timer_irq_enable()

□ 실행 순서

1. **VIC0VECTADDR25** 레지스터에 **ISR** 주소 저장
 - vh_timer_interrupt_handler 함수 주소 저장
2. **VIC0INTENABLE** 레지스터에서 **Timer4** 인터럽트를 활성화
 - 25번 비트를 1로 set
 - vh_io_hal.h에서 선언된 vh_VIC_TIMER4_bit 사용
3. **VIC0INTSELECT** 레지스터에서 **Timer4** 인터럽트를 **IRQ**로 설정
 - 25번 비트를 0으로 clear
 - vh_io_hal.h에서 선언된 vh_VIC_TIMER4_bit 사용
4. **VIC0SWPRIORITYMASK** 레지스터를 모두 **mask**
 - 모든 비트를 1로 set

vh_timer_irq_enable()

□ 코드 추가

```
void vh_timer_irq_enable(int timer)
{
```

```
    switch(timer){
    case 0:
        break;
    case 1:
        break;
    case 2:
        break;
    case 3:
        break;
```

```
    case 4:
        break;
    default: break;
}
```

Timer4 Interrupt를 활성화하는 부분
레지스터에 값 입력

vh_timer_interrupt_handler()

□ 설명

- Timer4 Interrupt Service Routine(ISR)
- 스케줄러 호출 (주기 : 1초)

□ 위치

- vpos/hal/io/timer.c

vh_timer_interrupt_handler()

□ 실행 순서

1. Timer4 Interrupt를 비활성화

- vk_timer_irq_disable() 함수 호출

2. Timer4 인터럽트를 pending clear

- VIC0INTENCLEAR 레지스터에서 Timer4 인터럽트를 비활성화
✓ 25번 비트를 1로 set
- VIC0INTENABLE 레지스터에서 Timer4 인터럽트를 활성화
✓ 25번 비트를 1로 set

3. Software Interrupt를 발생시켜 Supervisor 모드로 진입하고 스케줄러 함수를 호출

- vk_swi_scheduler() 함수 호출

vh_timer_interrupt_handler()

□ 코드 추가

```
void vh_timer_interrupt_handler(void)
{
    vk_timer_irq_disable(); <-----Timer4 Interrupt를 비활성화
    vh_save_thread_ctx(vk_timer_save_stk);

    // timer interrupt clear & enable <----- Timer4 Interrupt를 Pending Clear
    vh_VIC0INTENCLEAR |=
    vh_VIC0INTENABLE |=

    vk_sched_save_tcb_ptr = (unsigned int)vk_timer_save_stk;
    vk_timer_flag = 1;

    ++(vk_current_thread->cpu_tick);
    if(vk_sched_lock==0) {
        vk_swi_scheduler(); <-----
    }
}
```

1. 소프트웨어 인터럽트를 발생시켜
Supervisor 모드로 전환
2. Scheduler 호출

vk_timer_irq_enable()

□ 설명

- Timer4 시작 & Timer4 Interrupt를 활성화

□ 위치

- vpos/hal/io/timer.c


vk_timer_irq_enable()

□ 실행 순서 (Timer4 Start & Timer4 Interrupt Enable)

1. TCON의 Timer4 관련 비트를 모두 0으로 clear
2. TCON에서 Auto Reload On & Manual Update
3. TCON의 Timer4 관련 비트를 모두 0으로 clear
4. TCON에서 Auto Reload On & Timer4 Start
5. TINT_CSTAT 에서 Timer4 Interrupt Enable & Timer 4 Interrupt Status Bit Clear
 - Timer 4 Interrupt Status 비트에 1을 set
→ Timer 4 Interrupt Status Bit Clear

vk_timer_irq_enable()

□ 코드 추가

```
void vk_timer_irq_enable(void)
{
    
}
```

레지스터에 값 입력

vk_timer_irq_disable()

□ 설명

- **Timer4 정지 & Timer4 Interrupt**를 비활성화
- **Timer**를 통해 발생하는 특정 이벤트가 실행되길 원하지 않을 때 사용

□ 위치

- **vpos/hal/io/timer.c**


vk_timer_irq_disable()

□ 실행 순서 (Timer4 Stop & Timer4 Interrupt Disable)

1. TCON의 Timer4 관련 비트를 모두 0으로 clear
2. TCON에서 Auto Reload On & Manual Update
3. TINT_CSTAT에서 Timer4 Interrupt Disable & Timer 4 Interrupt Status Bit Clear
 - Timer 4 Interrupt Status 비트에 1을 set
→ Timer 4 Interrupt Status Bit Clear

vk_timer_irq_disable()

□ 코드 추가

```
void vk_timer_irq_disable(void)
{
    
}
```

← 레지스터에 값 입력

TIMER INTERRUPT (ASSEMBLY)

vh_irq_VIC0

□ 설명

- **VIC0**에 연결된 인터럽트를 처리
- 인터럽트의 **ISR**로 점프하고 복귀 루틴 구현

□ 위치

- `vpos/hal/cpu/HAL_arch_startup.S`

vh_irq_VIC0

□ 루틴 흐름

1. **VICADDRESS**의 값을 **pc**에 저장
 - 기존 pc값은 r14(lr)에 미리 저장
2. **CPSR**을 수정하여 **IRQ** 모드로 바꾸고 **IRQ Mask bit**를 1로 set
3. **IRQ** 모드의 **SPSR**, **lr**을 스택에서 복원
4. 이전 모드의 레지스터들을 스택에서 복원
 - 범용 레지스터 포함
5. **movs pc, lr** 명령어를 사용하여 원래의 루틴으로 복귀

VIC0 진입 루틴 & 복귀 루틴

□ 코드 추가

▪ vpos/hal/cpu/HAL_arch_startup.S

```
vh_irq:
    sub    lr,lr,#4
    str    sp, vk_save_irq_mode_stack_ptr
    stmfd  sp,{r14}^
    sub    sp, sp, #4
    stmfd  sp,{r13}^
    sub    sp, sp, #4
    stmfd  sp!,{r0-r12}
    mrs    r0, spsr_all
    stmfd  sp!,{r0, lr}
    str    sp, vk_save_irq_current_tcb_bottom
    ldr    r0, =0xe4000000
    ldr    r1, [r0]
    cmp    r1, #0x0
    bne    vh_irq_VIC0
    ldr    r0, =0xe4100000
    ldr    r1, [r0]
    cmp    r1, #0x0
    bne    vh_irq_VIC1
```

vh_irq_VIC0:

vh_irq_VIC1:

ldr r0, =0xe4100f00

실습

□ Coding

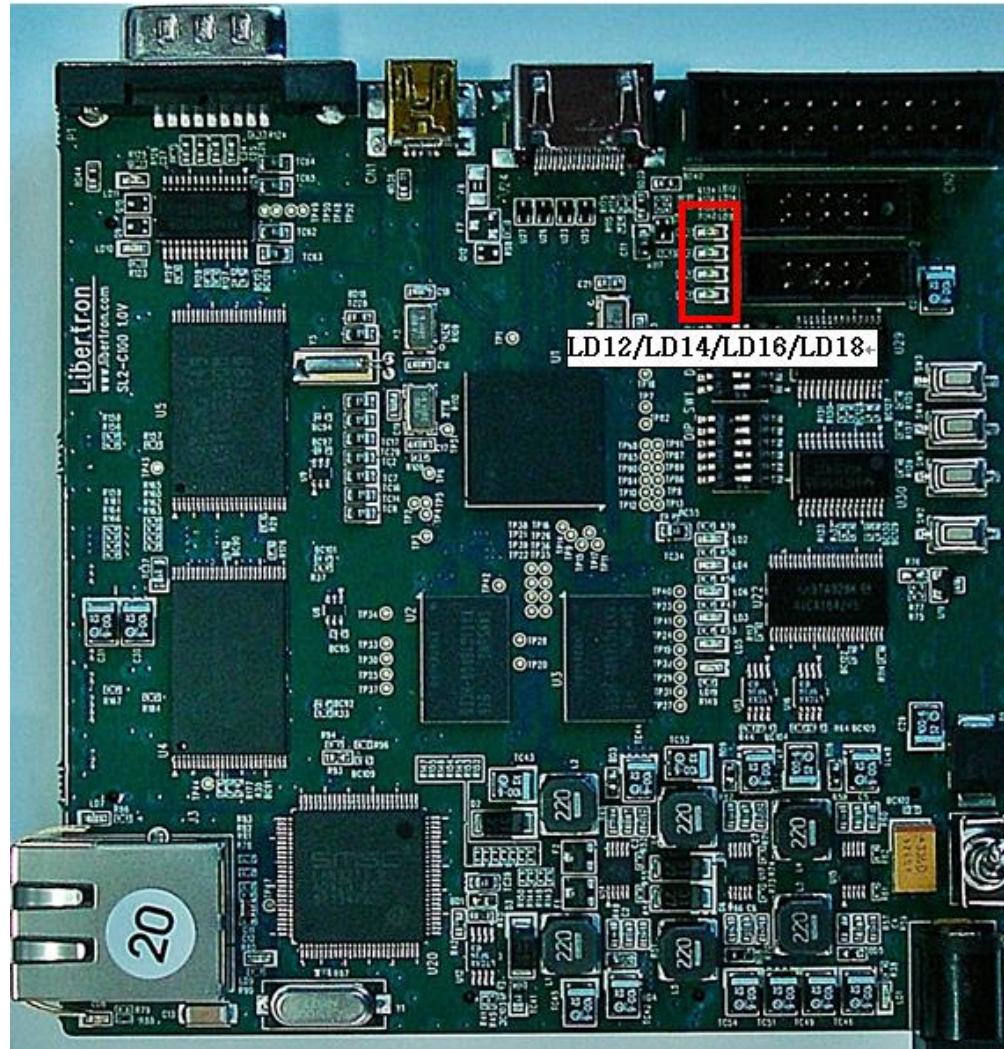
- **vh_io_hal.h**
 - 레지스터 주소 정의
- **HAL_arch_startup.S**
 - vh_irq_VIC0
- **timer.c**
 - vh_timer_irq_enable()
 - vh_timer_interrupt_handler()
 - vk_timer_irq_enable()
 - vk_timer_irq_disable()

실습

```
Shell>temp
thread1 cpsr : 60000110
thread1 sp   : 20107628
thread2 cpsr : 60000110
thread2 sp   : 20106be0
thread1 cpsr : 60000110
thread1 sp   : 20107628
thread2 cpsr : 60000110
thread2 sp   : 20106be0
thread1 cpsr : 60000110
thread1 sp   : 20107628
thread2 cpsr : 60000110
thread2 sp   : 20106be0
thread1 cpsr : 60000110
thread1 sp   : 20107628
thread2 cpsr : 60000110
thread2 sp   : 20106be0
thread1 cpsr : 60000110
thread1 sp   : 20107628
thread2 cpsr : 60000110
thread2 sp   : 20106be0
Shell>
```

GPIO (LED)

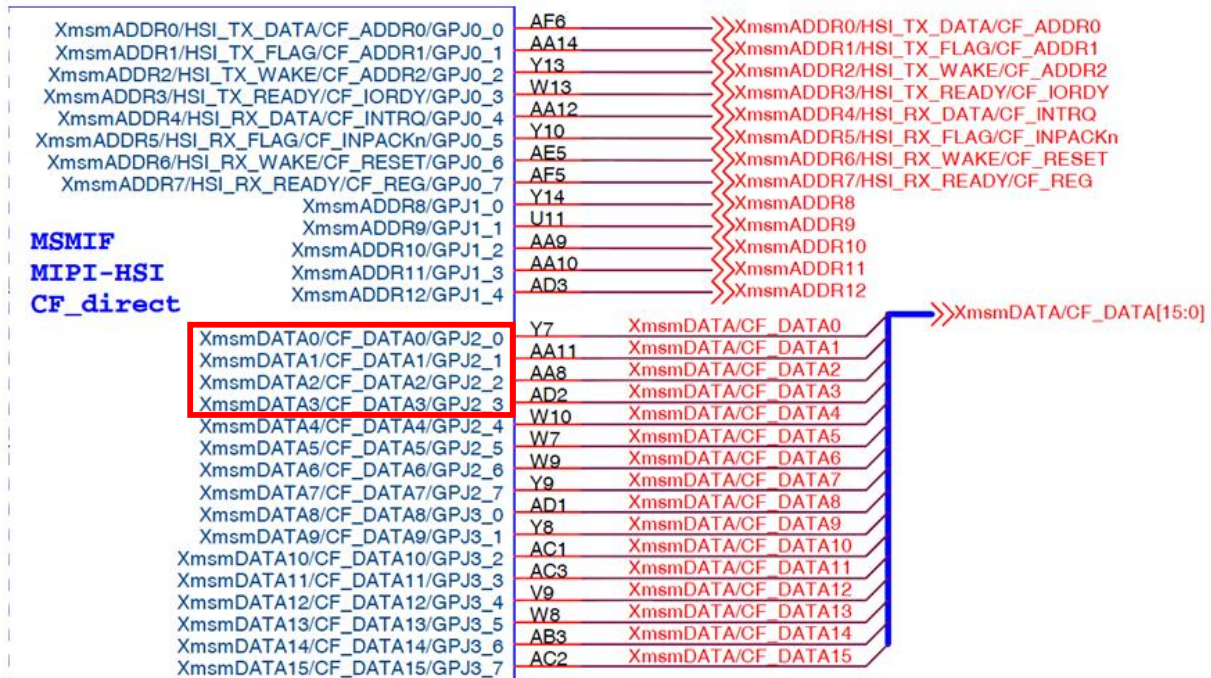
LED in SP5PC100 Board



GPIO & LED

□ LED는 GPJ2의 0 ~ 3번 핀에 연결되어 있음

SP5PC100 Chip



No.	Reference		Description
12	LD12	XmsmDATA/CF_DATA0	User LED (GPIO)
13	LD14	XmsmDATA/CF_DATA1	
14	LD16	XmsmDATA/CF_DATA2	
15	LD18	XmsmDATA/CF_DATA3	

레지스터 주소 정의

□ GPIO 관련 레지스터

▪ GPJ2CON

- GPJ2 Port의 제어 레지스터
- 각 포트의 핀을 입력 핀 or 출력 핀으로 결정

▪ GPJ2DAT

- GPJ2 Port의 데이터 레지스터
- 출력 핀일 경우, 해당 레지스터에 값을 **write**하면 핀을 통해 데이터를 외부로 전송
- 입력 핀일 경우, 현재 핀의 상태를 나타냄

GPJ2CON

□ Port Group GPJ2 Configuration Register (0xe0300240)

- 해당 포트의 각 핀을 입력 핀 or 출력 핀으로 설정
- 물리적으로 특정 하드웨어가 연결된 경우 해당 하드웨어와 연결
ex) UART

Field	Bit	Description	Reset Value
GPJ2CON[0]	[3:0]	0000 = Input, 0001 = Output, 0010 = MSM_D[0], 0011 = Reserved, 0100 = CF_D[0] 1111 = NWU_INT18[0]	0000
GPJ2CON[1]	[7:4]	0000 = Input, 0001 = Output, 0010 = MSM_D[1] , 0011 = Reserved, 0100 = CF_D[1] , 1111 = NWU_INT18[1]	0000
GPJ2CON[2]	[11:8]	0000 = Input, 0001 = Output, 0010 = MSM_D[2] , 0011 = Reserved, 0100 = CF_D[2] , 1111 = NWU_INT18[2]	0000
GPJ2CON[3]	[15:12]	0000 = Input, 0001 = Output, 0010 = MSM_D[3] , 0011 = Reserved, 0100 = CF_D[3] , 1111 = NWU_INT18[3]	0000

GPJ2DAT

□ Port Data Register (0xe0300244)

- 8비트로 구성. 비트는 각 핀과 대응
- 핀을 입력 핀으로 사용할 경우 현재 핀의 상태를 나타냄
- 핀을 출력 핀으로 사용할 경우 출력할 데이터를 저장

Field	Bit	Description	Reset Value
DAT[n] (n=0~7)	[n]	If the bit is configured as input, it represents the pin state. If the bit is configured as output, the pin state is the same as the value of the bit. If the port is configured as functional pin, an undefined value is read.	-

LED 초기화 방법

1. **GPJ2CON에서 0 ~ 3번 핀을 출력 핀으로 설정**
 - 각 핀에 해당하는 필드에 **0001**을 write
2. **GPJ2DAT에 0x0을 write**
 - GPJ2DAT 레지스터를 초기화

LED On/Off 방법

- GPJ2DAT 레지스터의 원하는 비트(핀)에 1을 저장
→ LED On
- GPJ2DAT 레지스터의 원하는 비트(핀)에 0을 저장
→ LED Off

VPOS_kernel_main()

□ 소개

- VPOS 커널 데이터 구조체를 초기화
- 시리얼 장치와 타이머 등 하드웨어를 초기화
- 인터럽트 **enable**
- 부팅 메시지 출력
- 쉘 스레드 생성
- 스케줄러 호출하는 **VPOS_start** 루틴으로 진입

□ 소스 코드 위치

- vpos/kernel/kernel_start.c

```
void VPOS_kernel_main( void )
{
    pthread_t p_thread, p_thread_0, p_thread_1, p_thread_2;

    /* static and global variable initialization */
    vk_scheduler_unlock();
    init_thread_id();
    init_thread_pointer();
    vh_user_mode = USER_MODE;
    vk_init_kdata_struct();

    vk_machine_init();
    set_interrupt();

    printk("%s\n%s\n%s\n", top_line, version, bottom_line);

    /* initialization for thread */
    race_var = 0;
    pthread_create(&p_thread, NULL, VPOS_SHELL, (void *)NULL);
    pthread_create(&p_thread_0, NULL, race_ex_1, (void *)NULL);
    pthread_create(&p_thread_1, NULL, race_ex_0, (void *)NULL);
    pthread_create(&p_thread_2, NULL, race_ex_2, (void *)NULL);

    VPOS_start();

    /* cannot reach here */
    printk("OS ERROR: VPOS_kernel_main( void )\n");
    while(1){}
}
```

vk_machine_init()

□ Code

- 하드웨어 장치 초기화
- `vh_serial_init()`: UART 초기화
- `vh_timer_init()`: Timer 초기화
- `vh_LedInit()`: LED 초기화 및 On

```
void vk_machine_init(void)
{
    vh_LedInit();
    vh_serial_init();
    vh_timer_init();
}
```

□ 소스 코드 위치

- `vpos/kernel/machine_init.c`

vh_LedInit()

□ 설명

- **GPIO** 설정 및 **LED** 초기화
- **4개의 LED** 모두 제대로 동작하는지 테스트
 - 0 ~ 3번 **LED**를 순서대로 켜기. 이를 5번 반복

□ 위치

- `vpos/hal/io/led.c`

vh_LedInit()

□ 코드

```
#define DELAY          0x10000

void vh_LedInit(void)
{
    int i,j, dly;
    vh_GPJ2CON = 0x1111; ← 0 ~ 3번 핀을 출력 핀으로 설정
    vh_GPJ2DAT = 0x0;
    for (i=0; i<5; i++) {
        for (j=0; j<4; j++) {
            vh_LedSet(j); ← j번 LED를 On
            for(dly=0; dly<DELAY; dly++);
        }
    }

    vh_GPJ2DAT = 0;
}
```

vh_LedSet()

□ 설명

- 매개변수에 해당하는 LED를 On

□ 위치

- `vpos/hal/io/led.c`

vh_LedSet()

□ 코드

```
void vh_LedSet(unsigned char data)
{
    switch(data) {
        case 0: vh_GPJ2DAT = 0x1; break;
        case 1: vh_GPJ2DAT = 0x2; break;
        case 2: vh_GPJ2DAT = 0x4; break;
        case 3: vh_GPJ2DAT = 0x8; break;
    }
}
```

0번 LED를 On

보고서 제출

□ 보고서

- 학과, 학번, 이름
- 코드를 캡처해서 보고서에 첨부
 - Timer interrupt
 - ✓ 실습 및 과제 (30 p)
 - GPIO (LED)
 - ✓ `vh_io_hal.h`: 레지스터 주소 정의
 - ✓ `led.c`: `vh_LedInit()`, `vh_LedSet()`
- **LED 동작 사진 촬영하여 첨부**
 - 이상 있을 경우 보고서에 명시

최종 소스 제출

□ VPOS 소스코드

- vpos_(학번).zip으로 압축
- 과제6 보고서와 함께 메일로 제출

제출 방법

□ 제출 방법

- 워드나 한글로 작성하여 메일에 첨부
- 문서 제목에 학번과 이름을 적을 것

□ E-mail (반드시 아래 2개의 메일 계정으로 모두 전송)

- jypark@rtcc.hanyang.ac.kr

□ 메일 제목

- [임베디드 시스템 실습 과제6]학번_이름

□ 마감일

- 다음 실습 수업시간 전까지

수고하셨습니다.