**Embedded System Software 과제 4**

**(과제 수행 결과 보고서)**

과목명: [CSE4116] 임베디드시스템소프트웨어

담당교수: 서강대학교 컴퓨터공학과 박 성 용

학번 및 이름: 20181593, 계인혜

개발기간: 2021. 06. 10. - 2021. 06. 21.

**최 종 보 고 서**

**I. 개발 목표**

Module programming, System call, Android programming, JNI 등 배운 내용을 활용하여 보드 상에서 자유 주제로 임베디드 소프트웨어를 개발하고 발표한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

1. Bulls and Cows device driver : 디바이스 드라이버와 커널 타이머 기능을 포함한 하나의 모듈을 구현한다.
2. 테스트 응용 프로그램 : 작성한 디바이스 드라이버와 interrupt handling이 올바르게 작동하는지 확인하기 위하여 JAVA를 사용하기 전에 테스트 응용 프로그램을 돌려보았다.
3. 타이머 기능 구현 : 일정한 시간 간격으로 스위치 입력을 읽기 위해 커널 타이머를 구현한다. (my\_timer\_func) 또한 게임 종료 조건을 확인하기 위한 타이머를 추가로 구현한다.
4. Fpga 모듈 출력 기능 구현 : 숫자 야구 게임에 맞도록 Device들의 출력을 수행한다.(FND, LCD, LED)
5. Interrupt Handling : 구현 목적에 따라 HOME, VOL+, VOL- 버튼에 대한 initialize, interrupt handler 구현, remove를 수행한다.
6. Android 응용 프로그램 : 해당 프로젝트에서 구현한 모듈을 커널에 넣고 C가 아닌 JAVA로 응용 프로그램을 제작한다. 이 과정에서 c로 짠 native 함수를 JNI로 수행한다.

**나. 개발 내용**

1. *Bulls and Cows device driver*

이전 프로젝트와 같이 커널에 모듈 형태로 올릴 수 있도록 Bulls and Cows 모듈을 제작하였다. 디바이스 드라이버를 짜기 위해서는 device interface table의 함수 포인터에 연결할 함수들을 작성해야 한다. 해당 프로젝트에서는 open, release, read, ioctl에 해당하는 함수들을 구현한 후 연결하였다. 또한, 해당 모듈이 insert 될 때 module\_init을 통해 호출되는 bullsAndCows\_device\_init 함수와 모듈이 remove 될 때 module\_exit을 통해 호출되는 bullsAndCows\_device\_exit 함수도 구현하였다. 마지막으로 MAJOR\_NUM으로는 242를, DEVICE\_NAME으로는 /dev/BullsAndCows를 사용하여 디바이스를 register 하였다.

1. *테스트 응용 프로그램*

해당 테스트 응용 프로그램에는 parameter가 없으며, 게임이 종료되기 전까지 테스트 응용 프로그램은 커널로부터 게임에 대한 정보를 주기적으로 받는다. 숫자 야구 게임이 끝나면 테스트 응용 프로그램은 종료된다.

1. *타이머 기능 구현*

유저의 스위치 입력을 확인하기 위한 것으로 0.1초 마다 입력 값에 따라 FND의 출력을 갱신하고, 종료 조건을 만족하면 출력에 사용한 디바이스(FND, LED, LCD)를 초기 단계로 재설정한다. 또한 스위치 입력을 위한 타이머는 게임이 종료되기 전까지 꾸준히 돌아야 하기 때문에 expire될 때마다 계속 타이머를 넣어준다.

또한 VOL- 키를 3초 이상 눌렀다가 떼었을 때 게임이 종료되는 조건을 확인하기 위해 새로운 타이머를 추가하였다. 이는 VOL-키를 누른 시점부터 VOL- 키에서 손을 뗀 시점까지의 시간을 계산하여 3초 이상이면 게임을 정상적으로 종료할 수 있도록 한다.

1. *Fpga 모듈 출력 기능 구현*

프로젝트 초기 단계에 구상한대로 FND, LED, LCD 디바이스에 올바른 값을 출력한다. 이전의 프로젝트에서 사용한 방법을 그대로 빌려와 physical address를 ioremap하여 bullsAndCows 디바이스에서 활용할 수 있었다.

1. *Interrupt Handling*

FPGA 보드에 있는 HOME, VOL+, VOL- 버튼을 이용하여 bullsAndCows 디바이스를 조작할 수 있다. 먼저, interrupt handler를 install 하기 위해 request\_irq 함수를 사용하였고, free 하기 위해서는 free\_irq 함수를 사용하였다. 해당 함수는 linux/interrupt.h에 정의되어 있다. 또한 interrupt handler를 implement 하기 위해 interrupt handler를 각 버튼마다 하나씩 구현하고 이를 request\_irq의 handler로 넘겨주었다.

1. *Android 응용 프로그램*

JAVA를 이용하여 미리 만들어 둔 커널 모듈과 데이터를 주고받을 수 있는 응용 프로그램을 제작한다. 이때 C로 짠 native 함수를 JAVA에서 돌릴 수 있도록 JNI를 사용하였다.

1. *새로운 System call 제작*

디바이스를 오픈하는 과정에서 에러가 발생하는지 확인할 수 있는 시스템 콜을 제작한다.

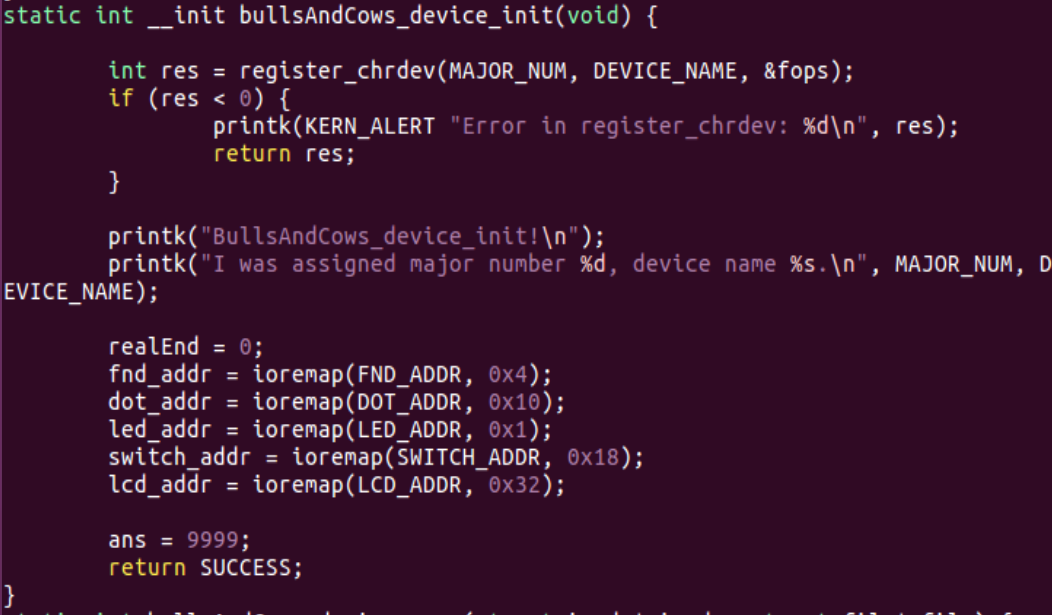
**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 21.06.10 ~ 21.06.12 | 프로그램 구상 |
| 21.06.14 | 모듈 기능 구현 |
| 21.06.18 ~ 21.06.19 | 모듈 기능 구현, 숫자야구 관련 함수 제작,  FPGA 출력 기능 구현 |
| 21.06.19 ~ 21.06.20 | Android app 구현, JNI 구현 |
| 21.06.20 ~ 21.06.21 | 보고서 작성, 발표 자료 제작 |

**나. 개발 방법**

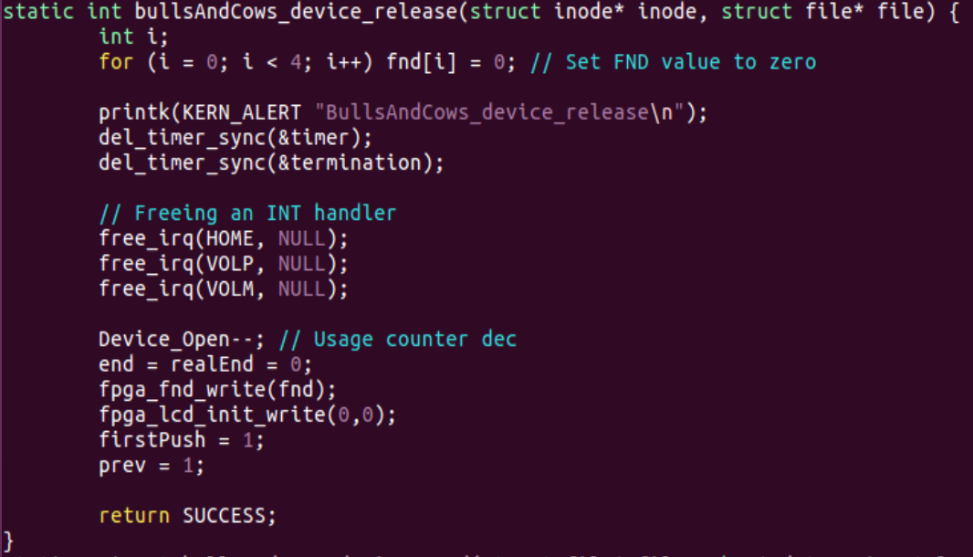
1. *BullsAndCows device driver*
2. Module\_init(bullsAndCows\_device\_init)





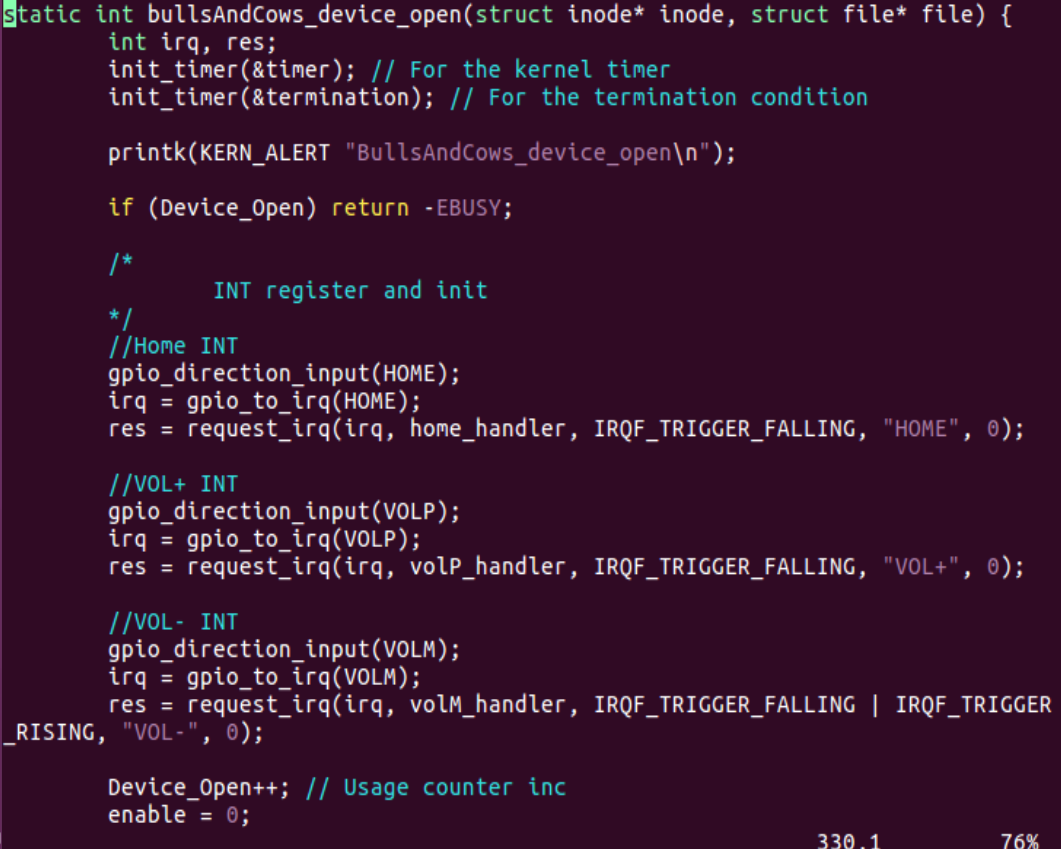
bullsAndCows 모듈을 insert 하면 호출되는 함수로 file\_operations 구조체의 멤버에 직접 구현한 함수들을 연결하여 드라이버를 register 한다. 성공적으로 작업이 수행되면 자신의 MAJOR\_NUM, DEVICE\_NAME을 출력한다. 또한, 보드의 다른 장치로의 출력을 위해 디바이스의 물리 주소로 ioremap한다.

1. Module\_exit(bullsAndCows\_device\_exit)



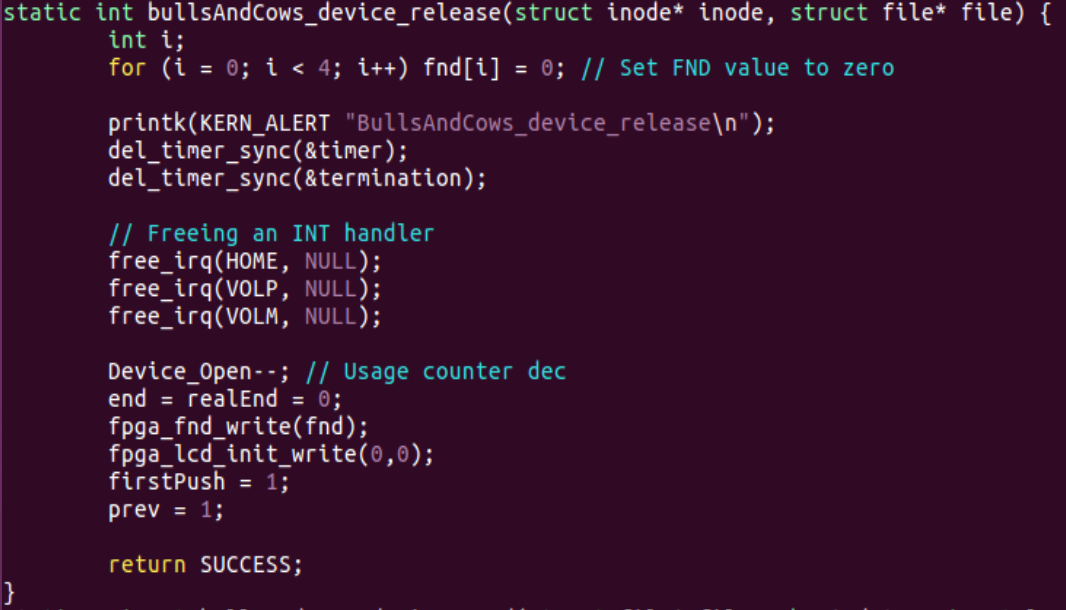
bullsAndCows 모듈을 delete 하면 수행되는 함수로 보드의 다른 장치로의 출력을 위한 장치를 iounmap한다. 시간 측정을 위한 타이머와 종료 조건을 위한 타이머를 모두 삭제하고, 해당 드라이버를 unregister한다.

1. bullsAndCows\_device\_open



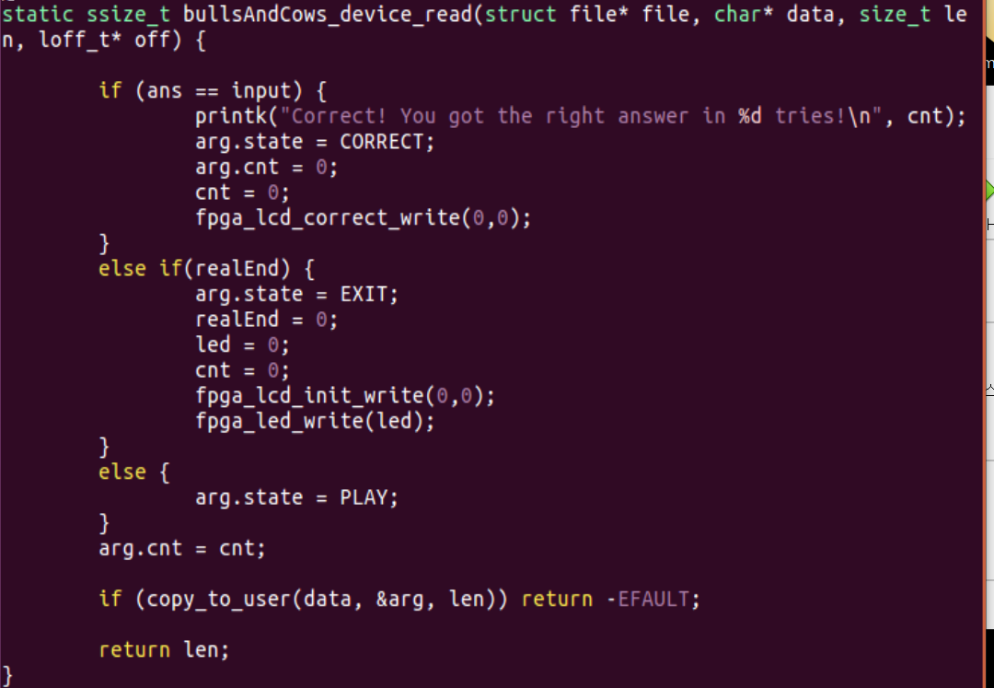
해당 함수는 먼저 스톱워치의 기능을 위한 타이머(timer), 종료 조건을 위한 타이머(termination)를 초기화한다. 이어서 usage counter(Device\_Open)을 통해 사용할 드라이버의 사용 여부를 확인하고, interrupt를 등록한다.

1. bullsAndCows\_device\_release



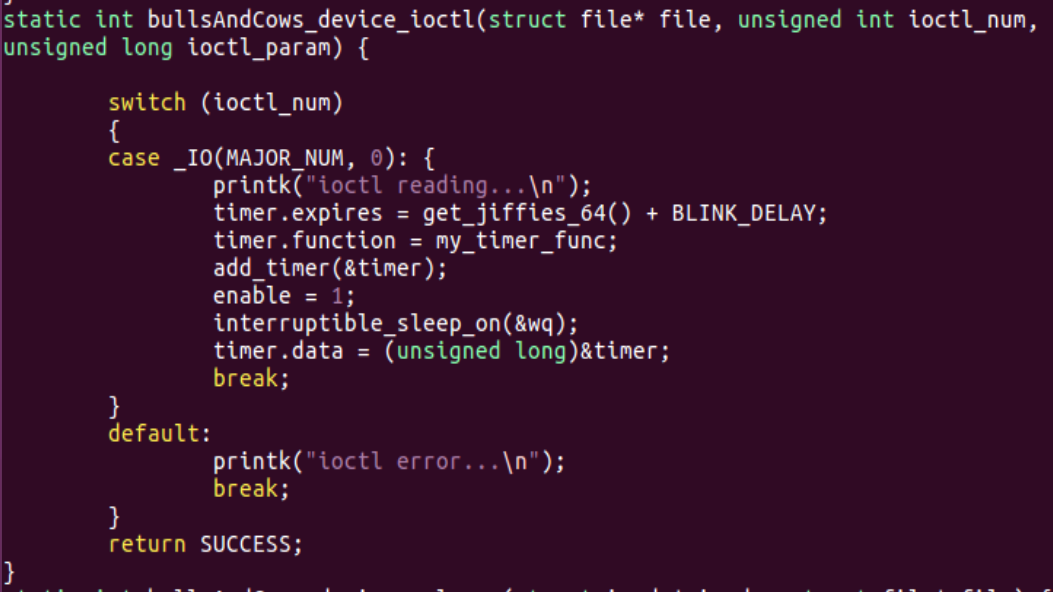
해당 함수는 등록한 커널 타이머들을 삭제하고, 등록된 interrupt들을 free한다. 또한, usage counter(Device\_Open) 값을 줄여 사용하지 않음을 표시한다. 종료 시 보드의 출력을 처음 상태로 초기화 할 수 있도록 설정한다.

1. bullsAndCows\_device\_read



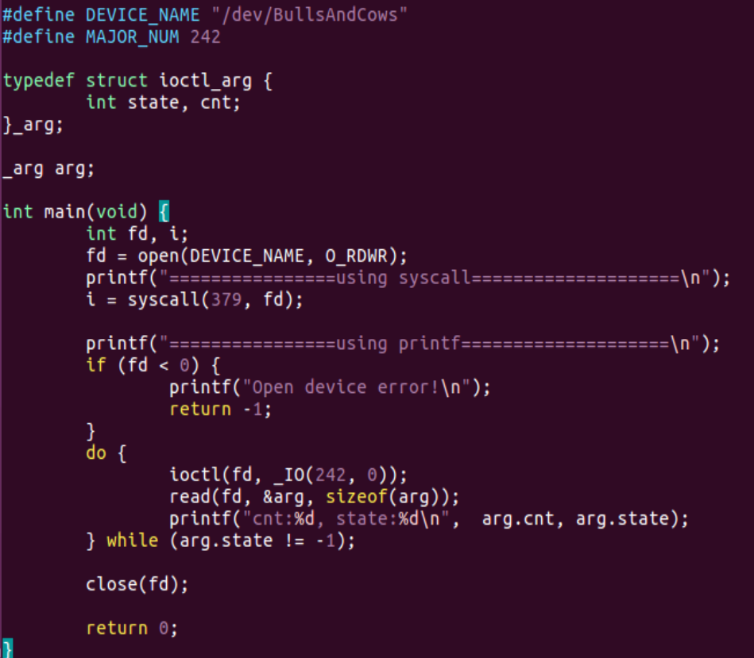
해당 함수는 유저가 주는 스위치 입력을 받기 위한 것으로 ioctl 함수와 연관이 깊다. 먼저, ioctl을 통해 응용 프로그램을 sleep 시키고 주기적으로 유저로부터 스위치 입력이 들어오는지 확인한다. 유저의 입력을 통해 정답 여부를 확인하고 그 과정에서 게임 종료 조건을 만족하는지 함께 확인한다. 만약 유저의 입력이 정답이라면 lcd 화면에는 정답 문구가 뜨고 다음 숫자를 맞추기 위한 게임을 계속 진행한다. 유저의 입력이 틀렸다면 응용 프로그램에게 게임이 계속 진행중이라는 데이터를 전달한다. 한편 그 사이에 VOL- 버튼을 3초 이상 눌렀다가 떼어서 게임이 종료된다면 응용 프로그램에게 게임이 종료되었다는 데이터를 전달하고 종료 시 디바이스에 초기화 상태를 표시할 수 있도록 한다.

1. bullsAndCows\_device\_ioctl



해당 함수는 커널과 응용 프로그램이 데이터를 주고받을 수 있도록 하며, ioctl 함수에서 응용 프로그램을 sleep 시키고 유저가 현재 FND에 표시된 숫자로 게임을 진행하기 위해 HOME 버튼을 눌렀을 때 응용 프로그램을 wake 한다. 다시 깨어난 응용 프로그램은 저장된 입력을 읽는다. 이때 응용 프로그램은 게임이 종료 조건을 만족했을 때에도 wake 하도록 설계하였다.

1. *테스트 응용 프로그램*

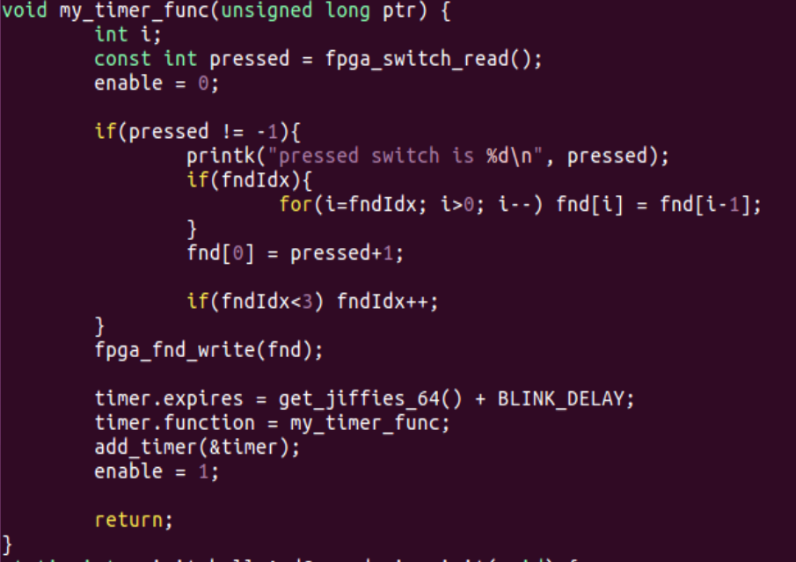


해당 프로젝트에서 테스트 응용 프로그램은 모듈을 open 하고, 모듈이 종료되면 다시 wake하여 디바이스 파일을 close하고 프로그램은 종료된다. 또한 앞서 설명한 것처럼 ioctl과 read를 이용해 주기적으로 유저로부터 스위치 입력을 받아 게임의 진행 상태를 파악한다. 만약 게임의 진행 상태가 EXIT(-1)이라면, 디바이스 파일을 close하고 프로그램을 종료한다.

1. *타이머 기능 구현*

타이머가 expire 되어도 게임이 종료되기 전까지는 타이머가 주기적으로 돌면서 사용자의 입력을 읽어와야 하기 때문에 handler 내부에서 계속 타이머를 넣어주었다. 이때 스위치 입력을 읽어오는 주기는 0.1초로 설정하였다.

타이머가 expire 될 때마다 스위치의 입력을 확인하지만, 사용자가 그 사이에 입력을 주지 않은 경우를 구분하기 위하여 pressed라는 값을 두어 입력이 없는 경우에는 별다른 일을 하지 않도록 하였다. 만약 유저가 스위치 입력을 주었다면, 스위치의 입력을 처리하기 쉽도록 변형하여 FND에 보기 좋게 출력한다.



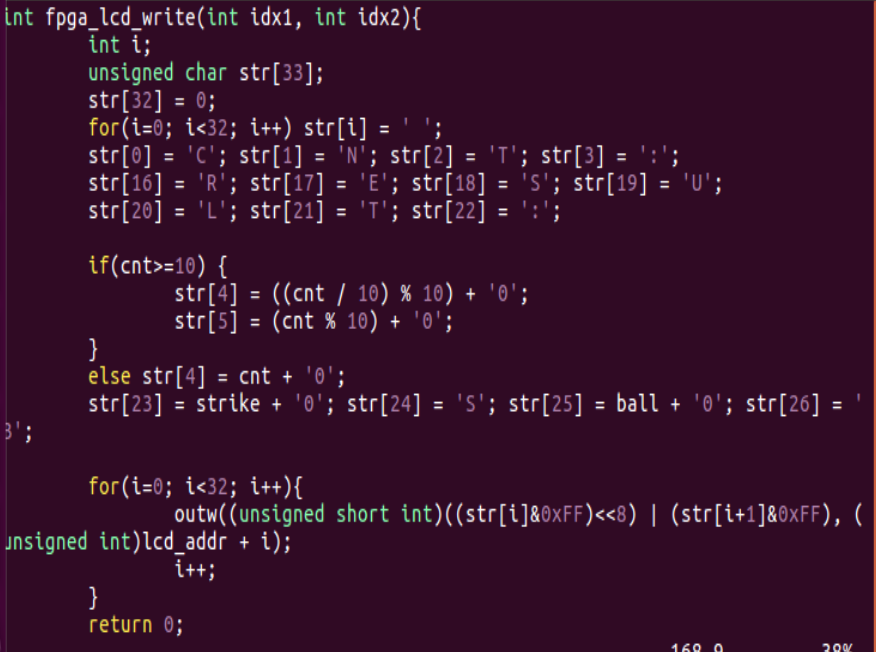
1. *Fpga 모듈 출력 기능 구현*
2. *Fpga\_fnd\_write*

텍스트이(가) 표시된 사진

자동 생성된 설명

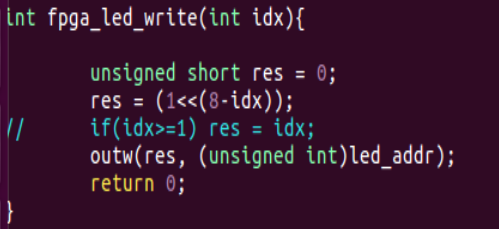
현재 FND에 출력하려는 값을 배열로 받아와 적절한 연산을 거쳐 하나의 정수 값으로 변환한다. 변환된 정수 값을 FND의 physical address로 보내 FND에 출력한다.

1. Fpga\_lcd\_write



현재 LCD에 출력하고자 하는 위치를 줄 별로 받아와 적절한 연산을 거쳐 구상 단계에서 적절한 형식으로 출력한다. 표시되는 정보는 사용자의 입력과 정답을 비교했을 때 strike와 ball의 개수, 그리고 현재까지 정답을 맞추기 위해 사용자가 도전한 횟수를 표시한다.

1. Fpga\_led\_write



현재 LED에 출력하려는 값을 받아 적절한 연산을 거쳐 하나의 정수 값으로 변환한다. 변환된 정수 값을 LED의 physical address로 보내 LED에 출력한다.

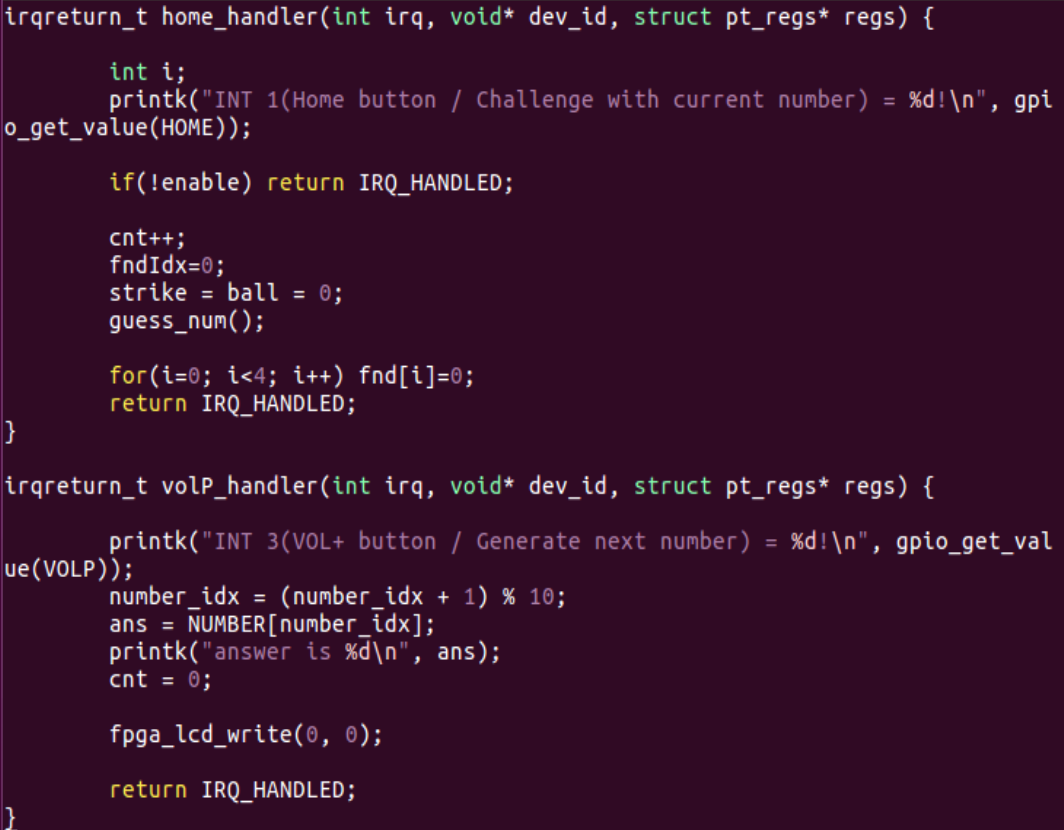
1. *Interrupt Handling*

bullsAndCows\_device\_open 함수에서 request\_irq 함수를 사용해 interrupt를 등록한다. 이때 VOL- 버튼은 누른 시간을 측정해야 하므로 버튼을 눌렀을 때와 버튼에서 손을 뗄 때 모두 interrupt가 발생해야 한다. 이를 구현하기 위해 flag에 IRQF\_TRIGGER\_FALLING, IRQF\_TRIGGER \_RISING을 OR로 연결하여 넣어주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

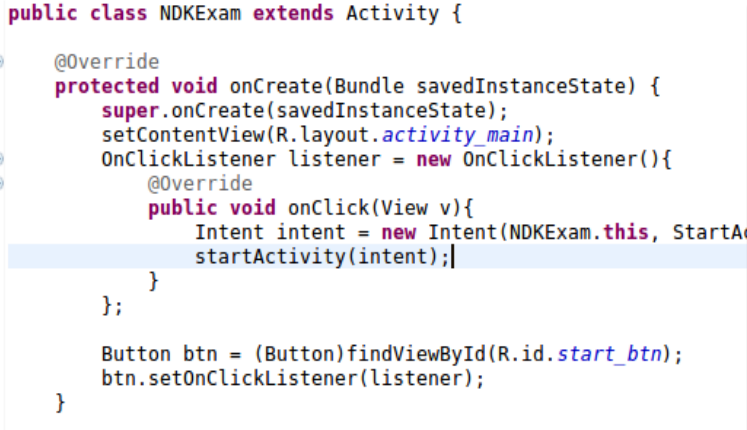
아래 사진은 HOME, VOL+ 버튼을 눌렀을 때 발생하는 인터럽트를 처리하기 위한 핸들러이다. HOME 버튼을 누르면 현재 FND에 표시된 숫자로 게임을 진행하며, VOL+ 버튼을 누르면 다음 숫자로 패스하여 게임을 이어서 진행하게 된다.



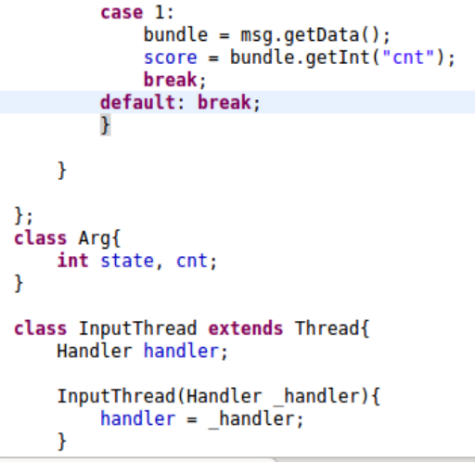
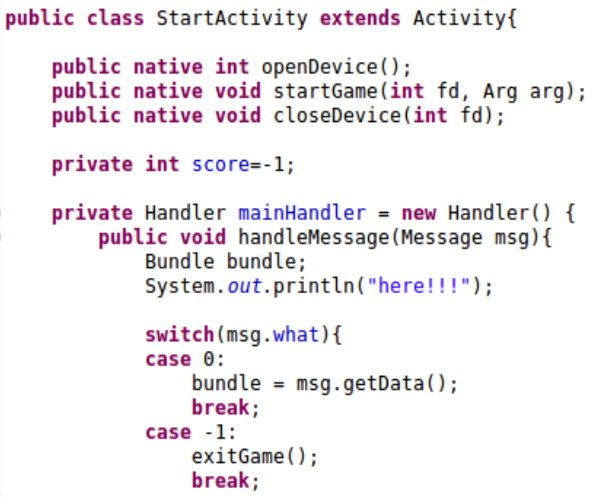
*VOL- 버튼은 3초 이상 누르고 있다가 떼었을 때 게임을 종료하는 기능을 수행하며 3초를 재기 위하여 버튼을 눌렀을 때의 시간을* prev에 저장하고, 이어서 버튼에서 손을 떼었을 때의 시간을 curr에 저장한다. 즉, curr-prev(diff)는 버튼을 누르고 있던 시간을 의미한다. 이를 통하여 사용자가 해당 버튼을 3초 이상 누르고 있었는지 확인할 수 있다. 만약 3초 이상 버튼을 눌러 게임이 종료된다면 FPGA 디바이스의 출력을 초기값으로 설정해준다.



1. *Android 응용 프로그램*
2. *Activity*

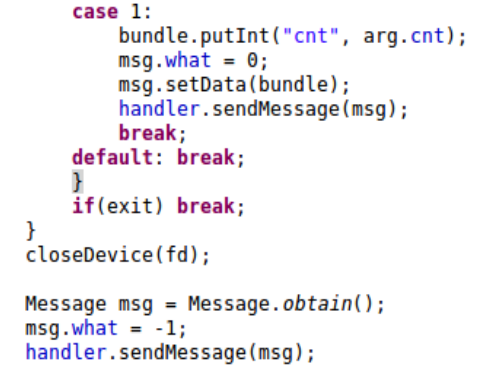
**

위 사진은 어플의 초기화면을 구현한 것으로 매우 단순한 형태를 띤다. 실행 시 ‘Start’라고 적혀있는 버튼을 누르면 listener를 이용해 다음 activity로 화면을 전환한다.

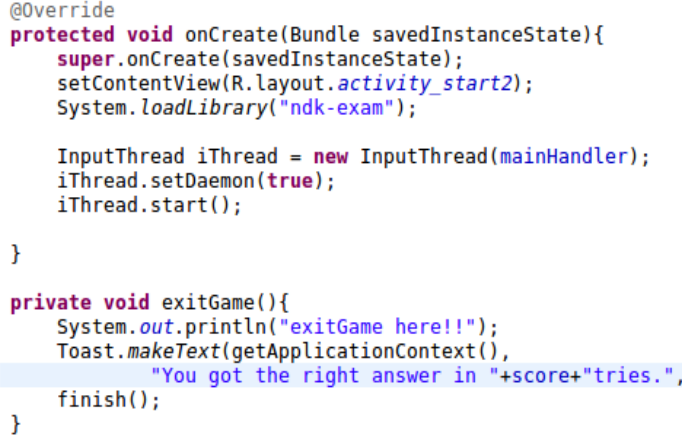


위 사진은 초기 화면에서 Start 버튼을 누른 후 전환된 activity로, 커널 모듈로부터 데이터를 받아 게임의 진행 상태에 따라 switch 문으로 구분하여 게임의 진행 여부를 결정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 사진은 모듈로부터 데이터를 읽기 위한 thread로 Android app의 main thread에 이를 전달한다.



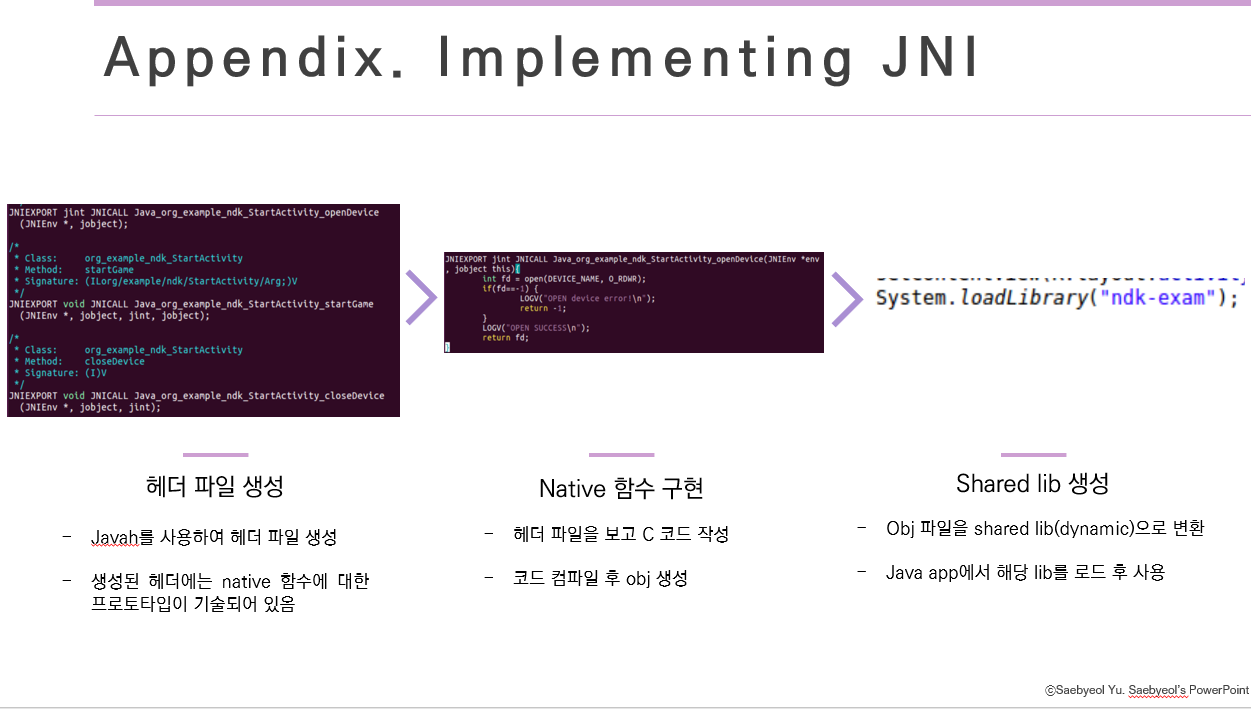
다음은 두 번째 activity의 초기 화면으로 위에서 언급한 input thread를 daemon thread로 변경하여 main thread가 종료되었을 때 같이 종료될 수 있도록 한다.

1. JNI

텍스트이(가) 표시된 사진

자동 생성된 설명

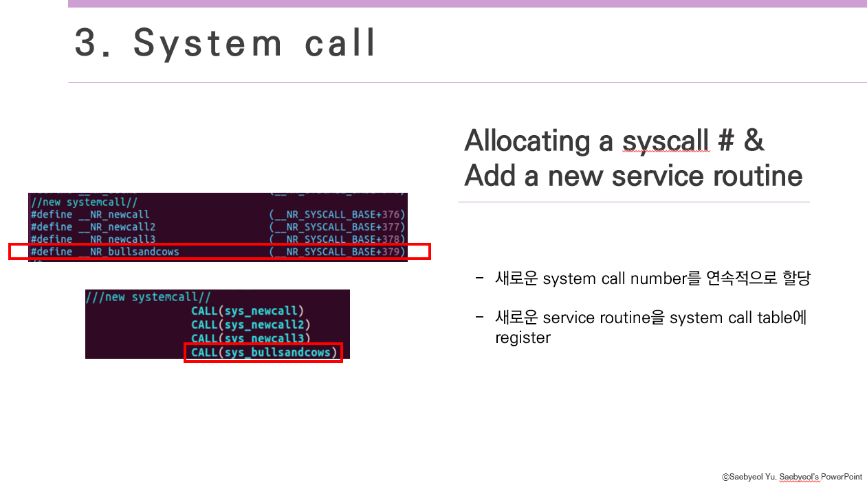
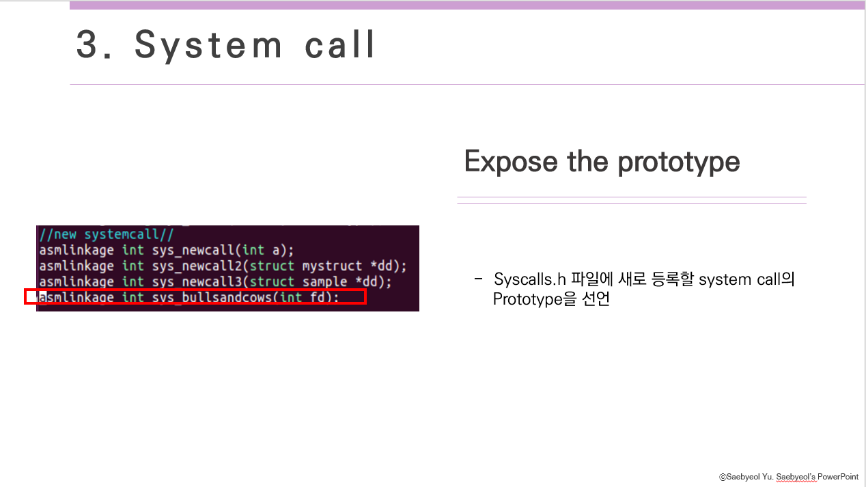
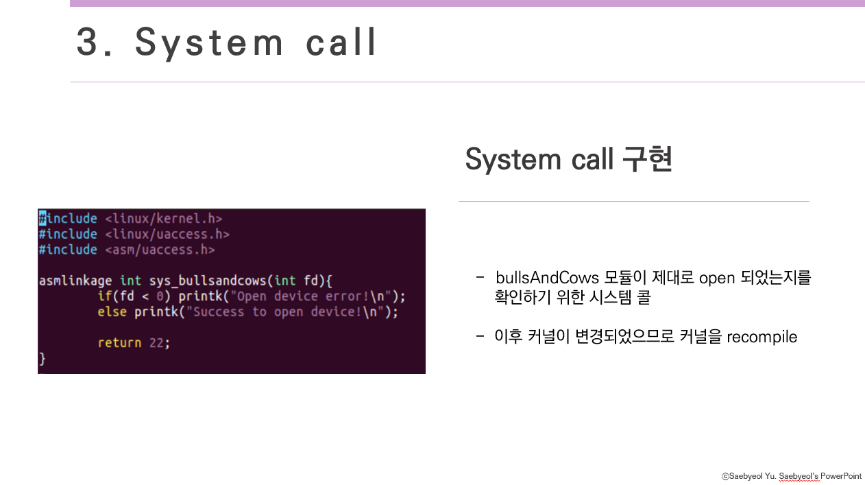
안드로이드 응용 프로그램의 두 번째 activity에 native method를 세 개 선언하였다. OpenDevice, closeDevice method 각각은 숫자 야구 모듈을 열고 닫는 역할을 하며, startGame은 JAVA와 native program 사이에 argument를 전달하는 method이다.



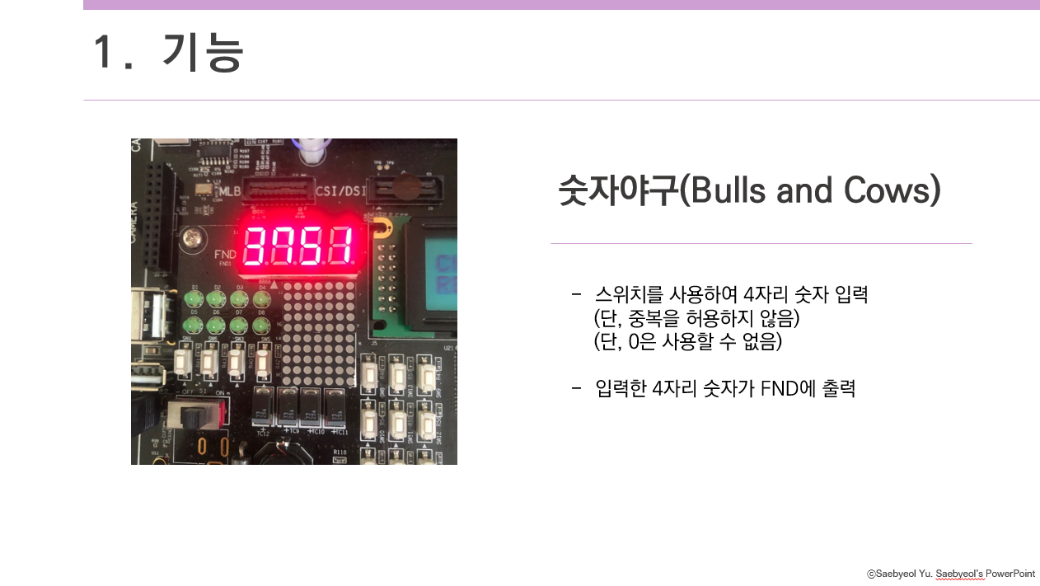
다음은 NDK와 JNI를 사용하여 JAVA와 C를 연결하는 과정을 한 눈에 보기 쉽도록 나타낸 것이다. 먼저, javah를 사용하여 헤더 파일을 생성한다. 이때 헤더 파일에는 native 함수에 대한 프로토 타입이 선언되어 있다. 이후 헤더 파일을 참고하여 native 함수를 실제로 작성한 다음 컴파일 하여 obj 파일을 생성한다. 마지막으로 이렇게 생성된 obj 파일을 안드로이드에서 사용하는 형태인 dynamic library(shared library)로 만든다. 이렇게 생성된 라이브러리를 java app에서 로드하면 마치 자바에 있는 함수처럼 native 함수를 사용할 수 있다. 즉, 이 과정을 통해 java app은 application framework를 거치지 않고 직접 C코드를 이용할 수 있다.

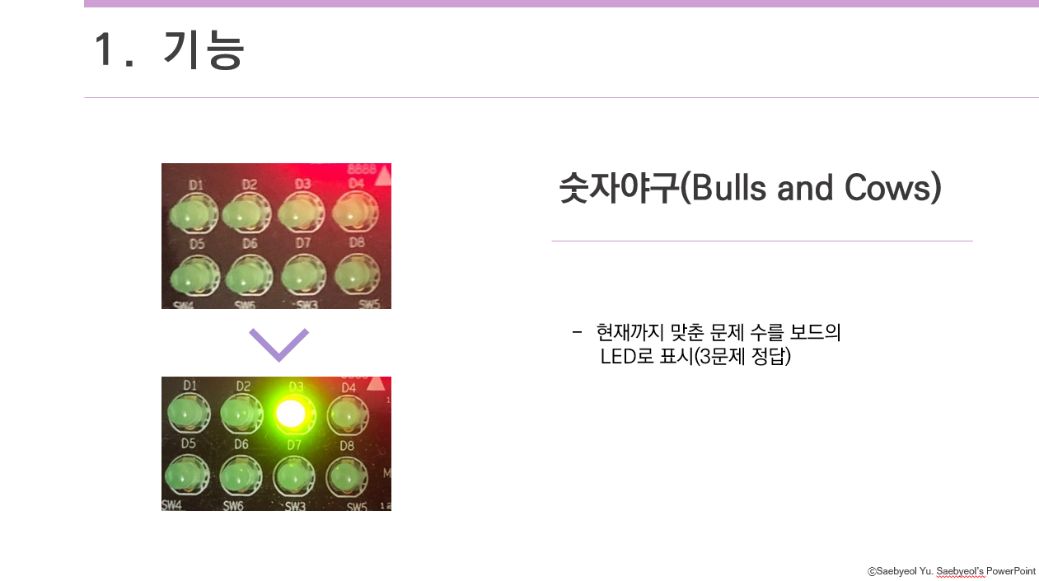
1. *새로운 System call 제작*

디바이스를 오픈하는 과정에서 에러가 발생하는지 확인할 수 있는 시스템 콜을 제작한다. 다음은 이번 프로젝트에서 새로운 시스템 콜을 만들어 커널에 추가하는 과정을 나타낸 슬라이드이다. 먼저 system call number를 연속적으로 할당하여 *arch/arm/include/asm/unistd.h*에 저장하고, system call table에 이를 register 한다. 그 다음, 새로운 system call의 prototype을 *work/achroimx\_kernel/ include/linux/syscalls.h*에 추가한다. 마지막으로 새로 등록할 system call을 실제로 작성한다. 이때 커널이 변경되므로 커널을 recompile한 후, fastboot 하고 나니 새로 추가한 시스템 콜을 성공적으로 사용할 수 있었다.

**

**IV. 연구 결과**



**V. 기타**

Fpga\_lcd\_write 함수는 이전 과제에서 직접 구현했던 함수로 이번 프로젝트에 맞게 변형을 하지 않고 그대로 가져와서 사용하였다. 그 결과 당연하게도 과제에서 사용했던 기능과 이번 프로젝트에서의 LCD 표기법이 달라 LCD 출력을 위한 코드가 상당히 비효율적이라고 생각한다. 아쉽게도 시간이 부족한 관계로 최적화를 하지 못하였다.

또한, 정말 안타까운 점은 모듈을 minicom으로 테스트해보았을 때는 무척 잘 실행이 되는데, JAVA로 테스트 어플리케이션을 만들면 왠지 모르게 제대로 작동이 안되었다. 강의 자료를 몇 번이고 정독하고, 구글링도 매우 열심히 했지만, 끝까지 원하는 만큼의 성과를 내지 못한 것 같아 아쉬움이 크다. 시간이 조금만 더 있었더라면 디버깅에 시간을 더 투자할 수 있었을 텐데, 매번 느끼는 거지만 프로그래밍 과제는 예상보다 더 일찍 시작해야 하는 것 같다.