Project #2: Doke'mon

Internet Mobile Programming

Due: June 15, 2018

Team Kaqiqi

Hyunik Kim (32132582)
Jihye Mun (32151569)
Saeyoung Wi (32152775)

Department of Mobile Systems Engineering
College of International Studies
Dankook University

1. Project Introduction

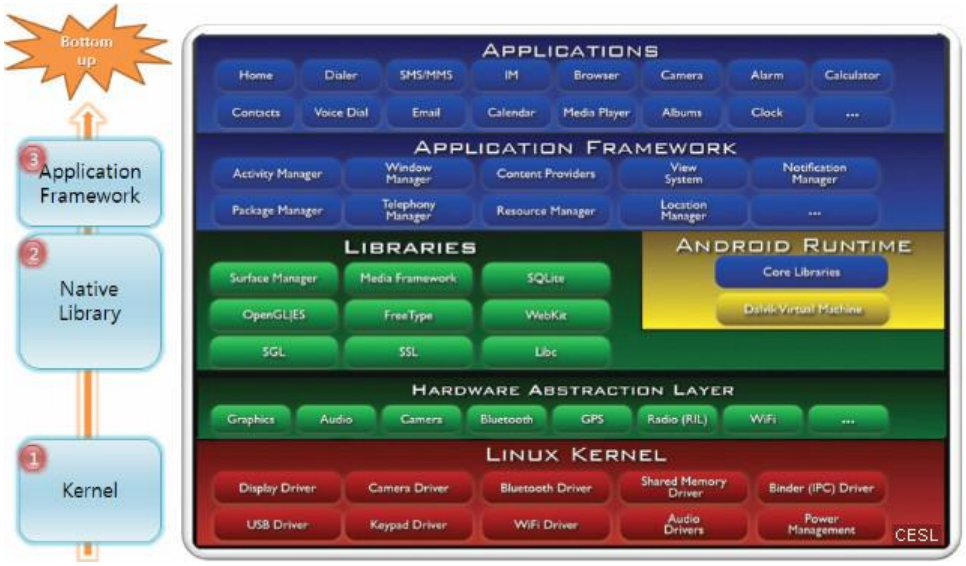
이번 프로젝트는 안드로이드 기반의 HBE-SM5-S4210의 코어 하드웨어와 기타 주변 디바이스를 활용하는 어플리케이션을 만드는 것이다. 안드로이드는 리눅스 커널을 기반으로 구글에서 제작한 프레임워크이다. 해당 프로젝트를 통해 Java기반의 어플리케이션 레벨에서 low-level인 하드웨어를 제어하는 원리와 과정을 배우는 것이 목표이다.

Doke'Mon은 화면 속에 나타난 포켓몬의 등장 순서를 기억한 후, 그 순서대로 버튼을 입력하는 게임이다. 입력은 화면 터치와 보드의 키패드를 통해 이루어진다. 게임에서 사용되는 디바이스는 OLED, TextLCD, FullLED, SimpleLED 이다. TextLED와 SimpleLED 현재 스테이지 표시, FullLED는 게임의 결과 그리고 OLED는 각 스테이지 이미지를 띄운다.

2. Core concepts

A. 안드로이드

안드로이드(Android)는 구글에서 제작하는 리눅스 커널을 기반으로 하는 운영체제 프레임워크이다. 전 세계에서 가장 대표적인 오픈소스 플랫폼이라고 할 수 있으며 GNU 라이선스를 기반으로 한다. 모바일 운영체제의 점유율의 대부분을 차지한다.



안드로이드는 상당히 오랜 기간 개발된 커널인 리눅스커널을 기반으로 동작하며 메모리와 프로세스 관리 모델이 표준화 되어있어 매우 안정적이라고 평가를 받는다. 안드로이드 플랫폼은 커널, 하드웨어 추상 레이어, 라이브러리, 어플리케이션 프레임워크 등의 4개 부분으로 나뉜다. 어플리케이션 프레임워크는 자바를 기반으로 동작하며 하위 레이어는 C와 C++로 구현되어 있다. 따라서 어플리케이션 프레임워크와 하위계층이 상호동작 하려면 JNI를 사용해야 한다. JNI에 관해서는 C항에 상세히 설명되어 있다.

B. Device

HBE-SM5-S4210는 다양한 입/출력 장치를 가지고 있다. 본 하드웨어에서 동작하는 Linux 내에는 이에 대한 장치드라이버가 존재한다. 따라서, 리눅스 커널에서 파일로 쉽게 접근이 가능하며 하드웨어 제조사가 정의한 스펙에 따라 write()이나 ioctl() 함수를 통해 장치의 상태를 변화시킬 수 있다. 본 프로젝트 에서는 TextLCD, SimpleLED, FullLED, OLED 등 4가지의 출력 장치를 사용하며 Keypad와 Touchscreen의 2가지 입력 장치를 사용한다.

C. JNI

Java Native Interface로 안드로이드 프레임워크에서 C/C++과 자바 레이어가 유기적으로 동작하게 만들려면 자바 레이어(상위)와 C/C++ 레이어(하위)를 상호 연결해 주는 매개체가 필요하다. 자바와 C/C++ 모듈 간의 인터페이스를 가능하게 해주는 것이 바로 JNI(Java Native Interface)이다.

안드로이드 SDK를 토대로 만든 안드로이드 애플리케이션은 달빅 가상 머신(Dalvik Virtual Machine) 위에서 동작하는 자바 기반의 프로그램이다. 때문에 C/C++로 생성한 애플리케이션에 비해 느린 실행 속도 등 자바가 지닌 여러 한계를 그대로 가지고 있다. 가령, 그래픽 처리나 시그널 프로세싱처럼 CPU의 처리 속도가 중요한 부분에서는 자바보다 C/C++ 같은 네이티브 코드로 작성한 모듈이 훨씬 더 나은 성능을 낸다.

- 빠른 처리 속도를 요구하는 루틴 작성

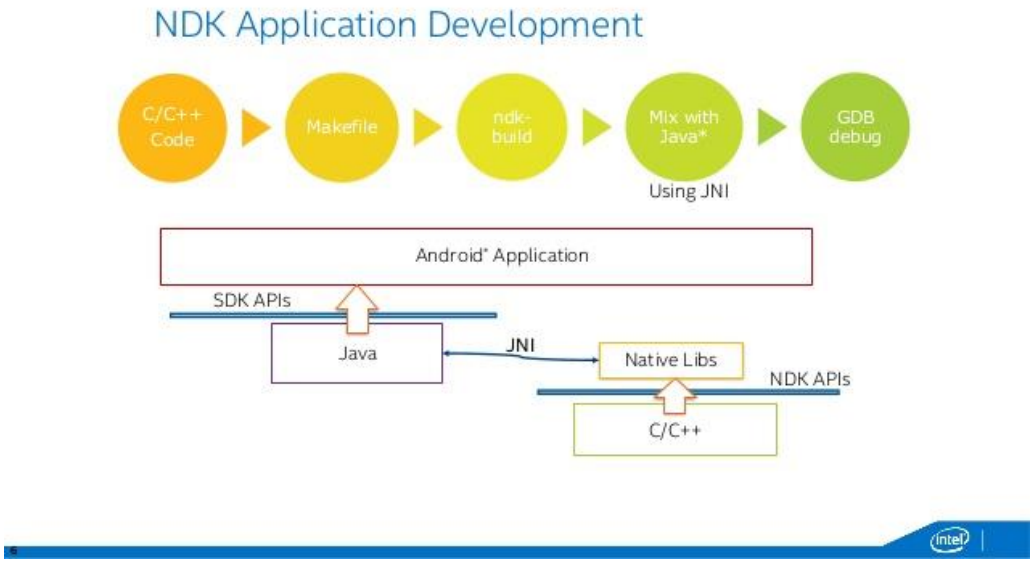
: 자바는 네이티브 코드(C/C++등)에 비해 처리 속도가 느리다. 따라서 빠른 처리 속도를 요하는 부분은 C/C++로 작성하고 이를 JNI를 통해 자바에서 호출하는 방법을 사용하여 처리 속도를 향상시킬 수 있다.

- 하드웨어 제어

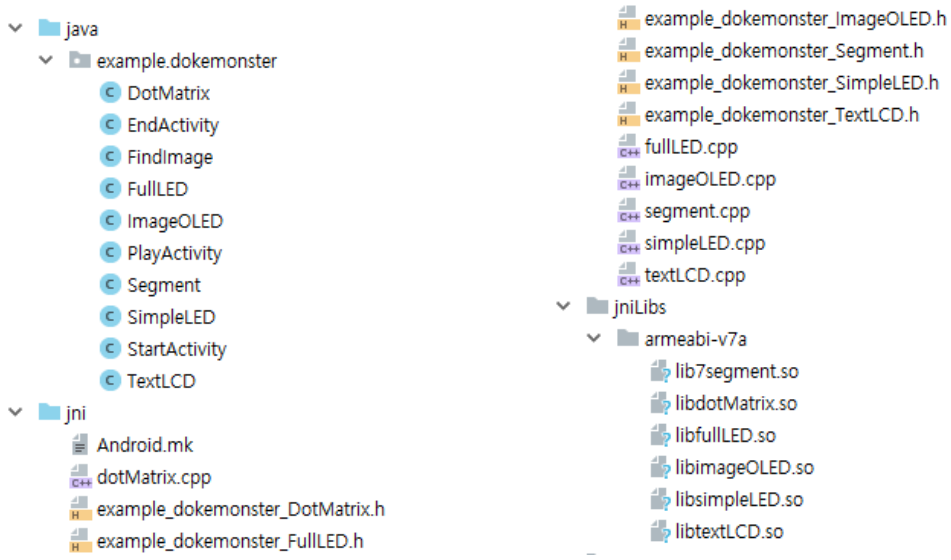
: 하드웨어 제어 코드를 C로 작성한 후 JNI를 통해 자바 레이어와 연결하면 자바에서도 하드웨어 제어가 가능하다.

- 기존 C/C++ 프로그램의 재사용

: 기존 코드가 C/C++로 작성돼 있다면 굳이 자바로 동일한 코드를 다시 작성하기보다는 JNI를 통해 기존 코드를 활용할 수 있다.



3. Program Structure



DotMatrix 클래스 : 입력 받은 string을 디바이스에 출력

- Stage 시작시 : Go! 출력
- Stage 성공시 : Correct! 출력
- Stage 실패시 : Fail! 출력
-

EndActivity : 모든 stage가 끝난 후 게임의 종료를 알리는 화면

FindImage : 조건에 맞은 이미지 반환

- selectStageImg(int stage_cnt) : stage별 이미지 반환
- selectBoard_init_img(int num) : 보드 번호 별 초기화 이미지 반환
- selectBoard_img(int num) : 보드 번호 별 게임 이미지 반환
-

FullLED : 게임의 승패와 스테이지 변환 시 각기 다른 불빛 출력

ImageOLED : stage 변환, 게임 종료 시 사진 출력

StartActivity : 어플리케이션의 첫번째 화면을 담당하는 클래스

- 디바이스 화면의 크기에 맞는 이미지뷰를 띄움
- 해당 이미지뷰를 클릭하면 다음 화면으로 넘어감
-

SimpleLED : 현재 플레이하고 있는 스테이지의 상태를 알려줌

- Display(int count) : 파라미터로 받은 stage에 따라 출력시킬 LED를 결정
-

PlayAcitivity : 게임 실행과 여러 이미지뷰의 클릭리스너를 처리

- error() : 입력한 값이 틀릴때 실행되는 함수로 4 * 4 보드에 정답을 띄움

- correct() : 입력한 값이 맞을때 실행되는 함수로 stage_cnt를 증가키고 해당 스테이지에 맞는 효과음과 FullLED를 출력
- playGame(int count) : 게임을 실행시키는 함수로 count에 따라 4 * 4 보드에 띄우는 이미지 수를 결정
- effectSound(int id) : 효과음을 내는 함수
- setInputImg(int cur, in img_id) : 입력창의 이미지뷰의 이미지를 결정하는 함수
- setBoard_image(int board_num, int image) : 4 * 4 보드의 이미지뷰의 이미지를 결정하는 함수

fullLED.cpp: 4개의 RGB 출력이 가능한 LED를 제어하는 C 코드

imageOLED.cpp: OLED 디바이스로 jni를 통해 이미지를 write()함수를 사용하여 출력하면 overhead가 상당하기 때문에 스테이지 이미지(사이즈 128*128 pixels)의 RGB 값을 추출하여 별도의 파일로 저장한 후 OLED에 접근하여 이미지 교체해주는 바이너리와 함께 임베디드 보드의 리눅스 영역에 저장하였다. 따라서 imageOLED는 다른 네이티브 라이브러리들과는 다르게 데이터를 전송하지 않고 시스템 함수로 기 내장된 바이너리를 호출하여 데이터 전송과정 없이 이미지를 교체한다.

simpleLED.cpp: 8개의 LED를 제어하기 위한 C코드

textLCD.cpp: textLCD를 제어하는 C코드

4. Source code

> https://github.com/0xc0ffeecafe/dockemon

5. Video demonstration

> https://youtu.be/MNNdUCwnxZc

6. Personal Feelings and Feedback

모바일 프로그래밍 시간에 안드로이드 어플리케이션 개발 경험이 있어서 그런지 이번에 어플리케이션은 큰 어려움 없이 만들 수 있었다. 하지만, jni를 연결하는데 예상치 못한 오류가 발생하여 이를 해결하는데 많은 시간이 걸렸다..ㅠㅠㅠ 힘들었지만 그래도 완성본을 보니 기분도 좋고 뿌듯했다. 마지막 프로젝트라는게 조금 아쉽다..(이것은 지극히 2명의 팀원의 의견입니다.)