**Embedded 5th HW.**

**(설계 프로젝트 수행 결과)**

**과목 명: [CSE4116] Embedded Systems**

**담당 교수: 서강대학교 컴퓨터 공학과 박 성 용**

**학번 및 이름: 20091648, 이 준 호**

**개발 기간: 2014. 06. 24. - 2014. 06. 30.**

**최종 보고서**

**I. 개발 목표**

본 프로젝트에서는 achro 4210 Exynos 보듸의 Android OS 위에서 작동하는 java application을 구현한다. 또한 Java 환경에서 JNI 함수로 device driver를 제어하여 HW1, HW2, HW3, HW4에서 구현한 내용을 구현한다.

**II. 개발 범위 및 내용**

**가. 개발 범위**

Huins 보드 위에서 실행시킬 수 있는 application을 구현한다. Application에서 board를 제어할 수 있도록 JNI를 통해 이루어진 Mode들을 구현한다.

**나. 개발 내용**

1) App 실행 시 초기화면

좌측 상단에 학번을 기재한다.

Text Editor 버튼 : Text Editor 페이지로 넘어간다.

Figure Switch 버튼 : Figure Switch 페이지로 넘어간다.

Watch 버튼 : Watch 페이지로 넘어간다

Puzzle 버튼 : Puzzle 페이지로 넘어간다

My Mode 버튼 : Custom Mode 페이지로 넘어간다

Quit 버튼 : Application을 종료 시킨다.

2) Text Editor Mode

HW1에서 구현하였던 Text Editor와 같은 기능을 하는 mode로, 사용자가 application이나 board 중 하나에서 입력을 실행할 경우 다른 곳에서도 동일한 값이 출력되어야 한다.

Modify 버튼을 눌렀을 경우 사용자가 application에 입력한 문자열이 board의 text LCD에 해당 문자열이 출력되고, fpga fnd에는 입력 받은 문자열의 길이를 출력해준다. 또, fpga led에는 문자열 길이를 이진수로 출력되도록 구현하고, fpga dot matrix는 현재 fpga fnd에 출력되고 있는 숫자의 일의 자리 숫자를 출력해주도록 한다. 그리고 application의 text length에는 현재 입력된 문자열의 길이가 출력된다.

Clear 버튼을 눌렀을 경우 application과 board의 모든 값들을 초기화 해주며, main menu 버튼을 누를 경우, application과 board의 모든 값을 초기화하고 main 화면으로 돌아간다. Get CPU usage 버튼을 누르면 현재 CPU 의 사용률을 출력해준다.

사용자가 board에서 입력하는 버튼을 9개로 이루어진 push button으로 구성한다. 새로운 버튼의 입력이 한 번 들어올 때마다 한 글자씩 출력한다. 한번 눌렀던 버튼을 다시 누를 때마다 해당 알파벳을 입력 수에 맞게 바꾸어준다.두 개의 버튼이 동시에 입력될 경우 그에 맞는 기능을 수행하도록 한다. 사용자가 스위치를 누를 때 마다, 보드의 fpga led, fpga fnd, fpga dot matrix의 값은 application의 text화면과 board상이 동일하게 최신화 되어야 하며, 문자열의 개수가 32를 넘어갈 경우 text lcd에 에러 메시지를 출력하도록 한다.

3) Figure Switch Mode

HW2에서 구현하였던, 사용자에게 입력 받은 일정한 시간 간격으로, 문양을 출력하는 기능을 가지는 mode를 구현한다. Go 버튼을 누를 경우, [시간] [횟수] [옵션]을 입력 값으로 받아 application에는 현재 출력되고 있는 문양의 번호를 출력해주고, board에는 HW2와 동일한 방법으로 출력을 시작한다.

gpio fnd : FND 4자리를 출력으로 컨트롤하고, 일정한 순서의 문양을 timer에 따라 반복해서 출력한다. 출력하는 위치는 한번의 로테이션이 끝날 때 마다 우측으로 이동한다. 지정 횟수 만큼의 출력이 끝나면 FND의 불을 꺼준다.

gpio led : gpio fnd가 현재 문양을 출력하는 위치의 바로 아래 LED를 점등한 상태로 꺼지지 않고, FND의 문양 출력 위치기 바뀔 경우 LED의 위치도 함께 바뀐다. 모든 문양의 출력이 끝나면 LED의 불을 꺼준다.

fpga led : 현재 gpio fnd에서 출력 중인 문양의 번호를 나타낸다. 모든 문양의 출력이 끝나면 fpga led의 불을 꺼준다.

fpga fnd : 초기 값은 문양이 출력되는 횟수이고, 한 문양이 출력될 때 마다 1씩 감소한다. 모든 문양의 출력이 끝나면 0으로 초기화 된다.

fpga dot : 현재 gpio fnd에서 출력 중인 문양과 같은 모양의 문양을 출력한다. gpio fnd의 문양이 바뀐다면, fpga dot도 gpio fnd와 같은 문양으로 함께 바뀐다. 지정 횟수 만큼의 출력이 끝나면 dot의 불을 꺼준다.

fpga text lcd : 첫 번째 줄에는 자신의 학번을 입력하고, 두 번째 줄에는 자신의 이름을 영문으로 입력한다. 두 줄 모두 timer에 따라 오른쪽으로 한 칸씩 shift 이동을 하고, 오른쪽 칸에 더 이상의 공백이 없을 경우, 왼쪽으로 shift 이동을 시작한다. 이 과정은 모든 문양의 출력이 끝날 때까지 계속되고, 종료 시 text lcd를 초기화 시킨다.

Clear 버튼을 누르면 application과 board의 모든 값들을 초기화하고, main menu 버튼을 누를 경우, application과 board의 모든 값들을 초기화 하고 main 화면으로 돌아간다. 또한 Text Editor mode와 다르게 CPU 사용률을 매 1초마다 주기적으로 갱신한다.

4) Watch Mode

Month 버튼을 누를 경우 현재 화면의 달을 증가 시켜 주고, day 버튼을 누를 경우 현재 화면의 날짜를 증가 시켜준다. Hour 버튼을 누르면, 현재 화면의 시를 증가 시키고, minute 버튼을 누르면 현재 화면의 분을 증가 시킨다. 마지막으로 setting new time 버튼을 누르면 board의 text lcd에 바뀐 날짜와 시간을 출력 해준다.

Stop watch의 경우 start 버튼을 누르면 application의 stop watch가 시작되고, board의 fpga fnd 도 실행되어, application과 같은 값으로 최신화된다. Pause 버튼을 누르면 application과 board의 stop watch를 모두 일시 정지 시키고, stop 버튼을 누를 경우 application과 board의 stop watch 를 모두 0으로 초기화 시킨다. main menu 버튼의 경우 application과 board 모두를 초기화 시키고 main으로 돌아간다.

또한 board의 fpga push switch의 1~4번까지의 버튼을 이용해 stop watch의 start, pause, stop, main을 각각 구현한다.

5) Puzzle Mode

HW4에서 구현한 puzzle mode에 점수 시스템을 도입한다. Start 버튼을 누르면 사용자에게 입력 받은 puzzle의 size를 통해, puzzle을 생성하고, board의 fpga fnd에는 처음 점수인 0점을 출력한다. 시간이 10초 미만이 남았을 경우, fpga dot에 남은 시간을 출력하고, 시간이 다 되었을 때, 순서대로 정렬되어 있는 puzzle의 개수만큼 점수를 올려 fpga fnd에 출력한다.

6) Custom Mode

fpga push switch를 이용하여 application과 board가 상호 연동이 되도록 구현한다. board의 switch를 누를 경우 application에 있는 9개의 버튼 중 해당하는 버튼이 클릭되며, 이는 application에서 선택해도 마찬가지이다. 버튼의 선택을 마친 후 modify 버튼을 누르면, 현재 눌려져 있는 버튼의 개수가 application에 나타나게 된다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

|  |  |
| --- | --- |
| 2014. 06. 24. | Application의 기본 layout 구현 |
|  | JNI 환경 구축 및 기본 기능 구현 |
| 2014. 06. 26. | Text editor 및 Figure switch 구현 |
| 2014. 06. 27. | Watch 및 Puzzle 구현 |
| 2014. 06. 28. | Custom mode 구현 및 readme.txt 작성 |
| 2014. 06. 29. | Code 최적화 및 정리 |
| 2014. 06. 30. | Document 작성 및 제출 |

**나. 개발 방법**

App 실행 시 초기 화면 : xml에서 각 버튼의 width를 MATCH\_PARENT로 해주어 가로 화면에 하나의 버튼이 가득 차도록 해준다. 또한 TextView를 이용하여 app의 최상단에 학번을 출력해 줄 수 있도록 하고, 각 버튼에 onClickListener를 더해주어 각 버튼에 맞는 기능을 수행 할 수 있도록 한다.

Text Editor Activity : 기본적은 layout을 구성하고, application 상에서 우선적으로 요구 사항에 대한 기능들을 구현한다. xml에서 각 object에 대해 width를 match\_parent를 사용하여 화면에 가득차도록 하고, onClickListener를 이용해 각각의 task를 수행할 수 있도록 한다. board상에서 application을 제어하기 위해, 별도의 thread를 생성하여 주기적으로 호출하여 text editor로써 제 기능을 할 수 있도록 사용자로부터 input을 받아온다.

FigureSwitch Activity : 다수의 linear layout을 이용해 화면을 구상하고 사용자로부터 input을 입력 받으면 그에 맞는 결과를 출력하도록 한다. 이때, input의 적합성을 판별하기 위해 pattern을 이용하여 검사하고, JNI를 이용하여 board의 device를 사용할 수 있도록 한다. 여기서 thread를 하나 생성하여 입력받은 시간 간격으로 작동하도록 하고 여기에 JNI함수를 호출하도록 하면, application과 board상의 데이터가 정확히 동일한 값을 나타내게 된다. 또한 board상에서 9번의 버튼은 0의 입력을 할 수 있도록 하였다.

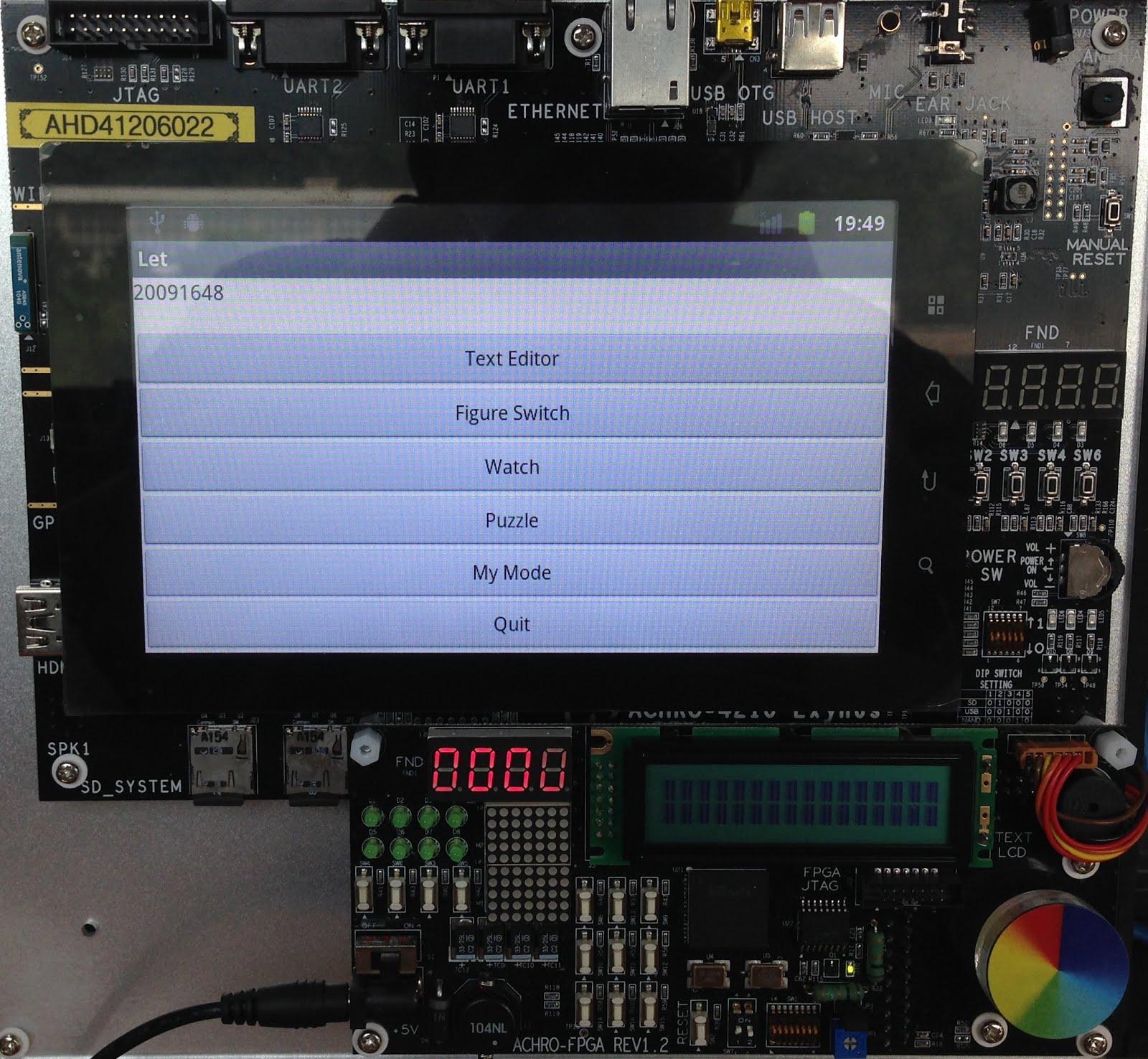
Watch Activity : 2개의 linear layout을 이용하여 화면을 구상하고, 왼쪽에는 사용자가 시간을 설정할 수 있도록 watch 기능을, 오른쪽에는 stop watch 기능을 구현한다. setting time버튼을 누르면 JNI 함수를 호출하여 fpga text lcd를 갱신하고, stop watch의 경우 thread를 이용해 버튼을 입력 받을 수 있도록 한다. 또한, 매 초마다 갱신 하기 위해 별도의 thread를 생성한 후 JNI함수를 계속해서 호출할 수 있도록 한다.

Puzzle Activity : HW4에서 구현하였던 puzzle mode에 thead를 생성하여 시간을 구현한다. 30초의 시간을 주어주고 남은 시간이 10초 미만이 되었을 때 thread안에 있는 JNI 함수를 호출하여 fpga dot을 갱신하여 주고, thread가 종료되면, fpga fnd를 갱신하여 계산한 점수를 출력해준다. 이때, 점수는 제자리에 위치해 있는 버튼의 개수 X 10으로 계산한다.

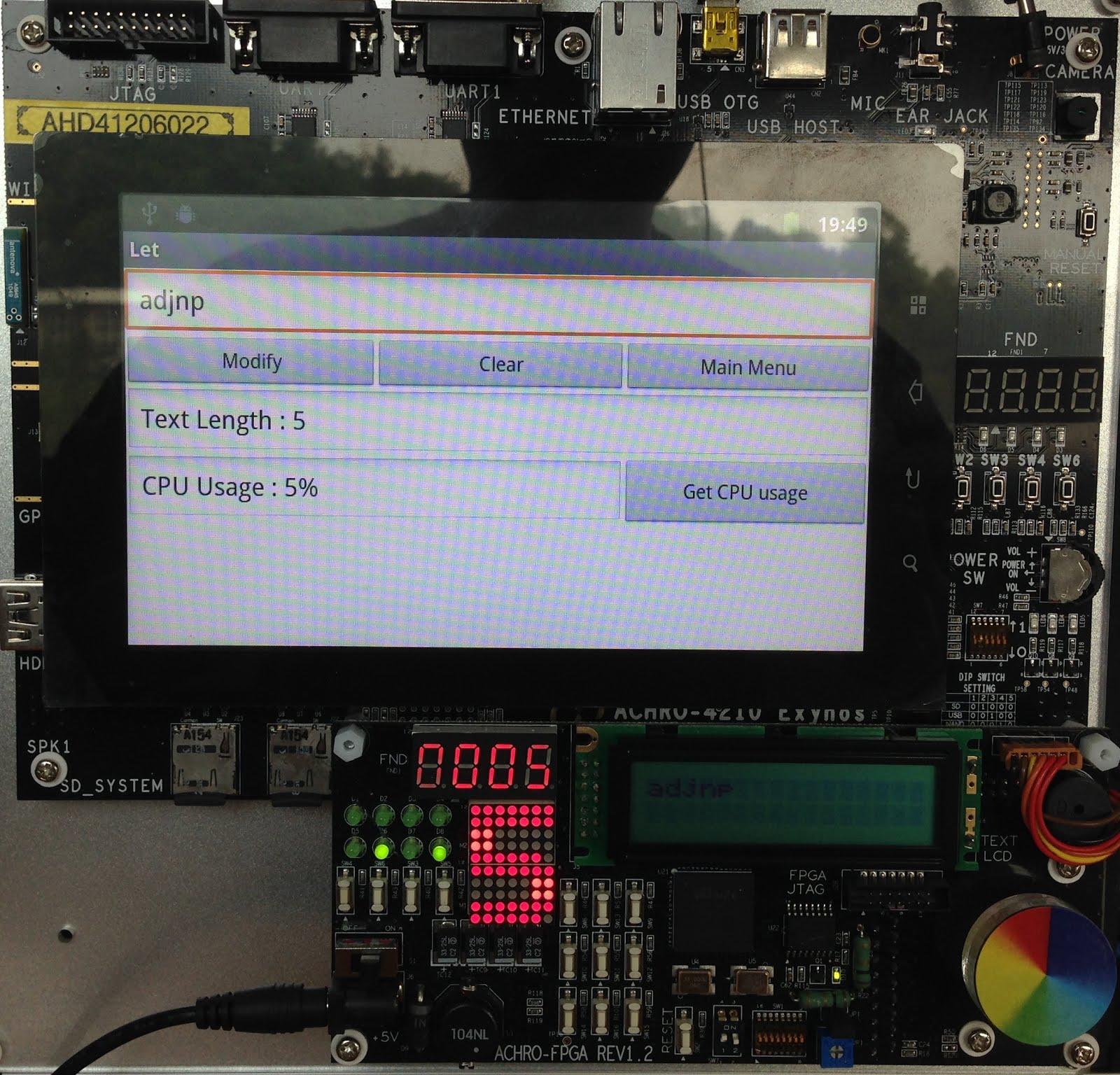
Custom Activity : Custom mode에서는 thread를 생성하여 application과 board의 push switch가 상호 연동할 수 있도록 구현하였다. application에서 버튼을 선택하면, 해당 버튼이 검은색으로 처리가 되고 다시 누르면 원래대로 복구 된다. thread를 생성하여 fpga push switch의 input을 받아 application상에서 해당하는 버튼을 application에서와 동일하게 작동되도록 한다. 또한 modify버튼을 누를 경우, 현재 눌려져 있는 버튼의 개수를 application과 board의 fpga dot에 출력해준다.

**IV. 연구 결과**

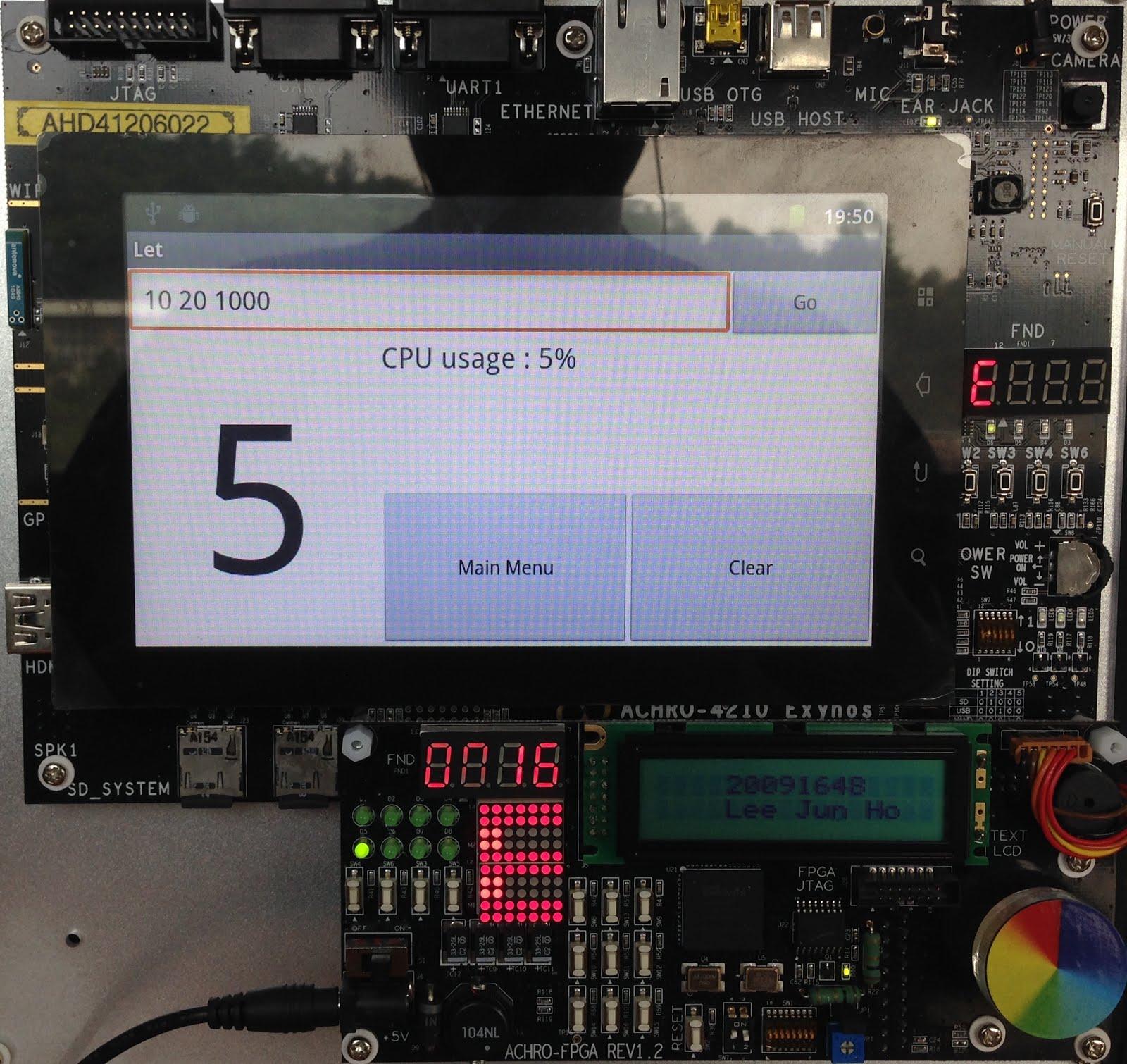
**1. 합성 내용:**



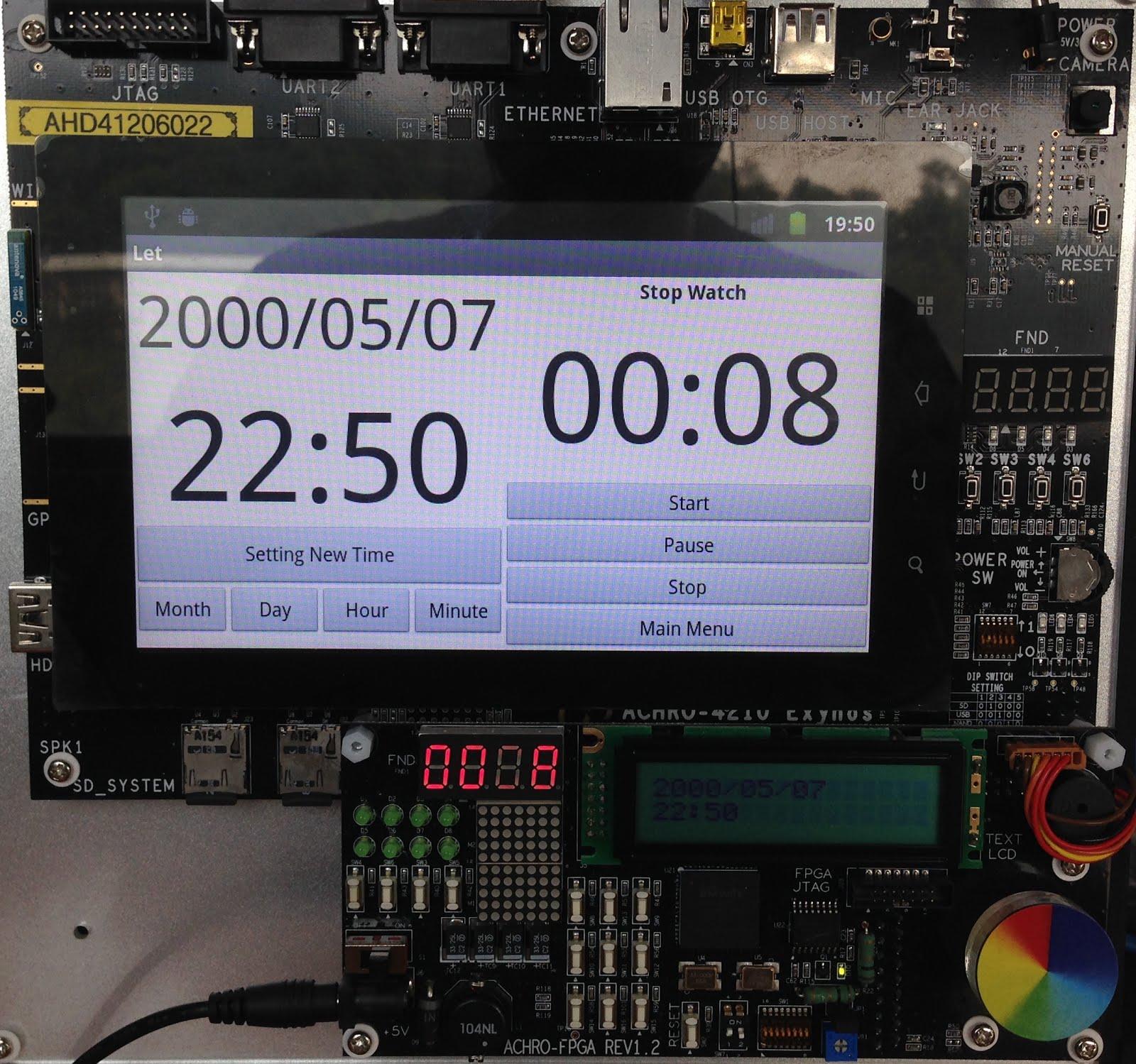
<Main Activity>



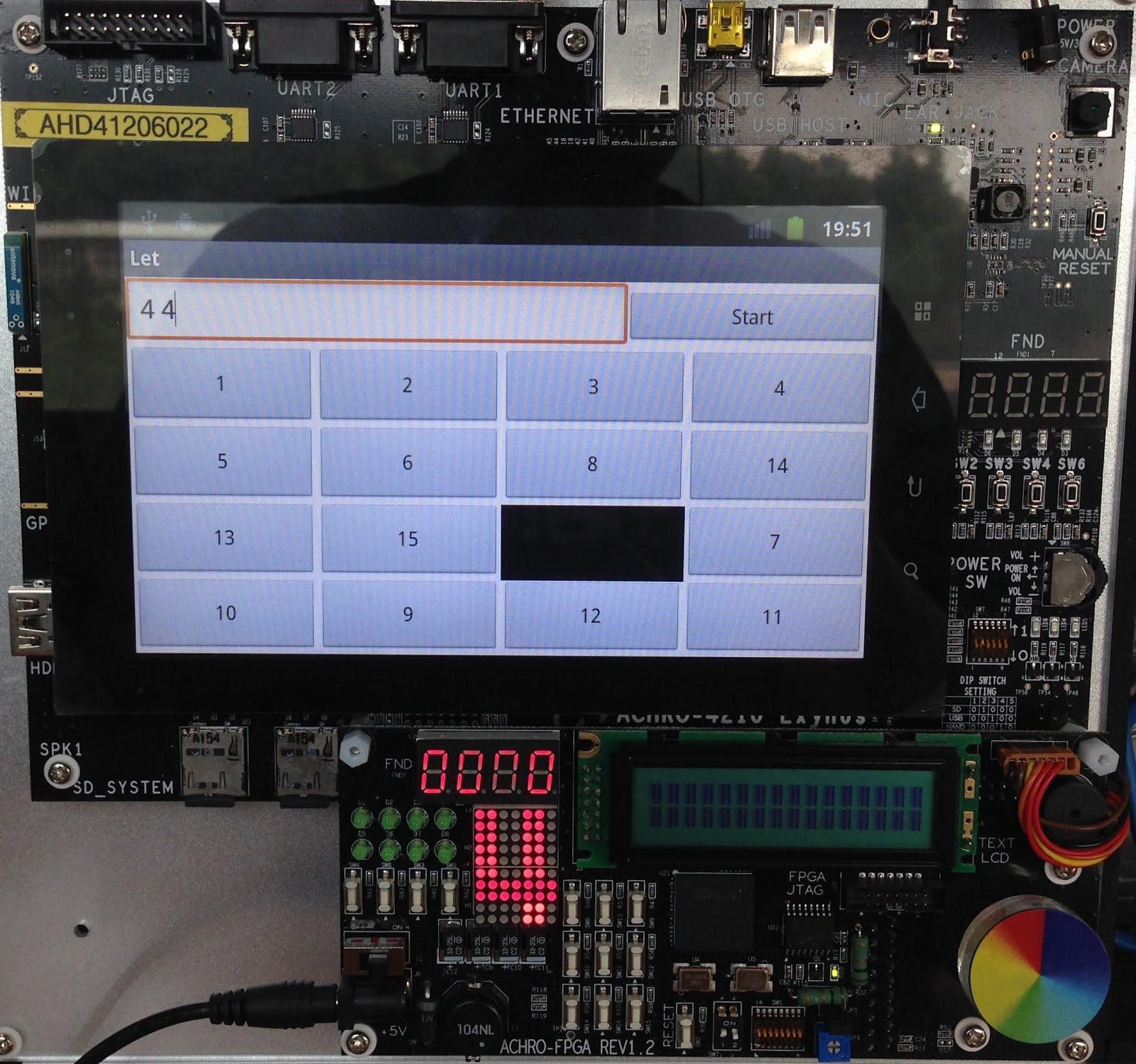
<Text Editor Activity>



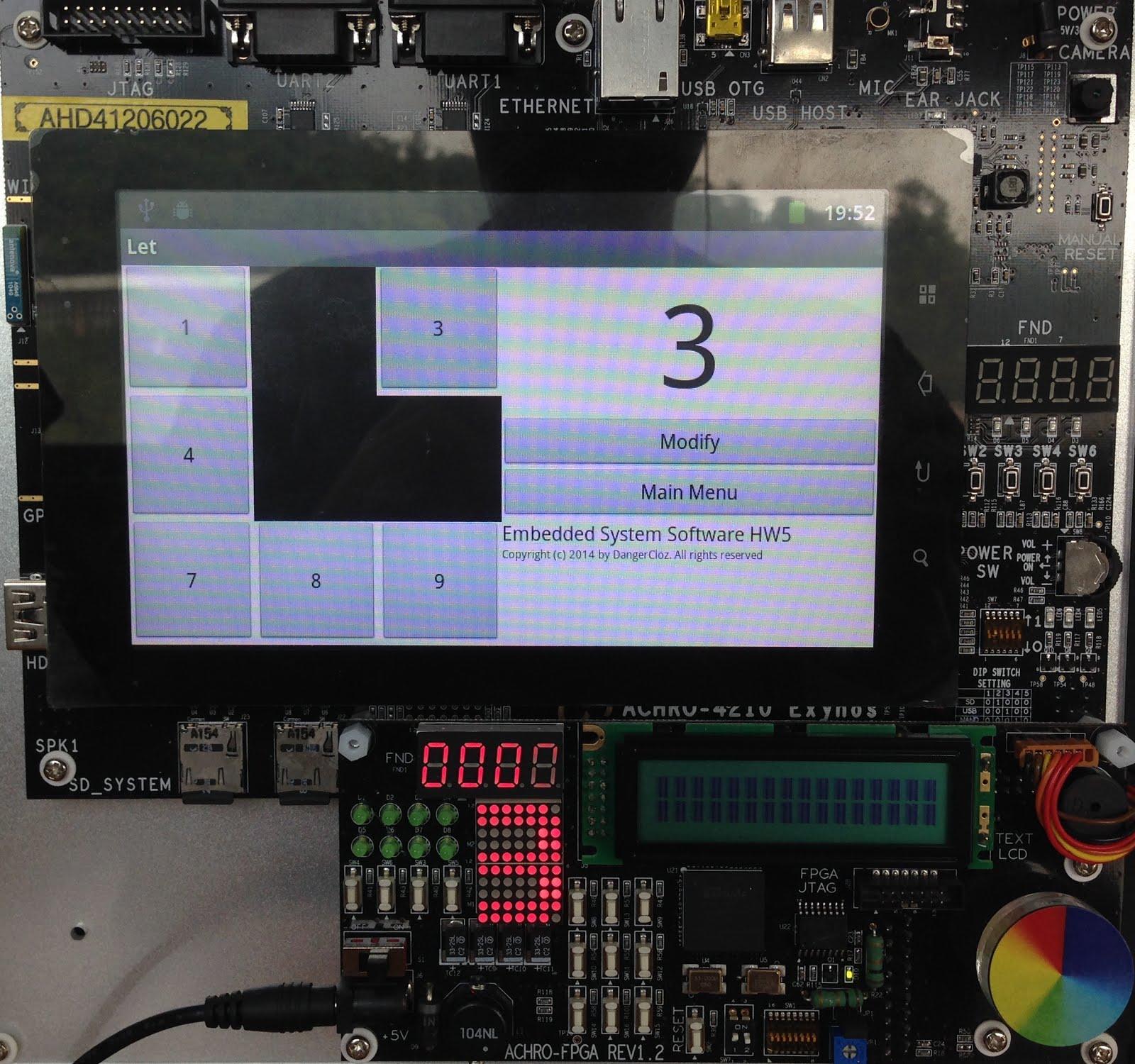
<Figure Switch Activity>



<Watch Activity>



<Puzzle Activity>



<Custom Activity>

**2. 제작 내용: 개발 결과**

**MainActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity에 버튼과 TextView를 생성하고 각 버튼에 다른 activity로 넘어가는 기능을 수행한다.

구현 방법 : onClickListener를 이용해 각 버튼마다 listener를 할당해주어 다른 activity로 넘어갈 수 있도록 도와준다.

**TextActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity의 기본 layout을 구성하고 각 버튼에 기능을 설정한다.

구현 방법 : 각 버튼에 onClickListener를 이용해 각 기능에 맞는 함수를 추가한다.

**class getSwitch extends Thread**

역할 : JNI를 이용하여 fpga push switch의 input을 입력받고, 그에 맞는 작업을 수행한다.

구현 방법 : JNI를 이용하여 fpga push switch로부터 input string을 받고, application에서 edit text의 값을 변경한다. 이 과정에서 다시 JNI 함수를 이용해 board상의 device들의 값을 변경하여 준다.

**public native void TextEditor(String string)**

역할 : board상의 device들의 값을 변경한다.

구현 방법 : 각 device들의 device driver를 open한 후, parameter로 넘겨 받은 데이터를 변환하여 각 device에 맞게 출력하여 준다.

**public native String PushSwitch(void)**

역할 : 사용자로부터 input을 받는다.

구현 방법 : fpga push switch의 device driver를 이용하여 0.01초마다 호출되어 input을 받도록 한다. 입력 받은 데이터는 다시 jstring형태로 변환하여 리턴하여 준다.

**public boolean onKeyDown(int keyCode, android.view.KeyEvent event)**

역할 : 사용자가 물리 back 버튼을 누를 시 모든 동작을 중단하고 main activity로 간다.

구현 방법 : Android의 물리 back 버튼을 누를 때 thread의 flag를 false로 바꾸어 모든 동작을 중단 시키고 main activity로 돌아간다.

**FigureActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity의 기본 layout을 구성하고 각 버튼에 기능을 설정한다.

구현 방법 : 각 버튼에 onClickListener를 이용해 각 기능에 맞는 함수를 추가한다.

**public native void FigureSwitch(String option, String left)**

역할 : board상의 device들의 값을 변경한다.

구현 방법 : 각 device들의 device driver를 open한 후, parameter로 넘겨 받은 데이터를 변환하여 각 device에 맞게 출력하여 준다.

**public native String PushSwitch(void)**

역할 : 사용자로부터 input을 받는다.

구현 방법 : fpga push switch의 device driver를 이용하여 0.01초마다 호출되어 input을 받도록 한다. 입력 받은 데이터는 다시 jstring형태로 변환하여 리턴하여 준다.

**class hwThread extends Thread**

역할 : 사용자로부터 입력받은 데이터에 대한 출력 결과 (무늬 출력) 작업을 수행한다.

구현 방법 : Thread를 이용하여 사용자로부터 입력받은 시간 간격으로 계산을 수행하고 JNI 함수를 이용해 board상의 device에 출력을 한다.

**class usageThread extends Thread**

역할 : CPU 사용률을 받아 출력한다.

구현 방법 : Thread를 이용하여 지속적으로 CPU의 jiffies를 읽어와 계산하여 application 상에 출력하여 준다.

**public boolean onKeyDown(int keyCode, android.view.KeyEvent event)**

역할 : 사용자가 물리 back 버튼을 누를 시 모든 동작을 중단하고 main activity로 간다.

구현 방법 : Android의 물리 back 버튼을 누를 때 thread의 flag를 false로 바꾸어 모든 동작을 중단 시키고 main activity로 돌아간다.

**WatchActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity의 기본 layout을 구성하고 각 버튼에 기능을 설정한다.

구현 방법 : 각 버튼에 onClickListener를 이용해 각 기능에 맞는 함수를 추가한다.

**public native void Watch(String date, String time)**

역할 : fpga text lcd에 application에 나와있는 현재 시간을 출력한다.

구현 방법 : JNI 함수를 이용해 jstring을 c string으로 변환하여 출력한다.

**public native void WatchFND(String stop)**

역할 : fpga fnd에 stop watch의 시간을 출력한다.

구현 방법 : JNI 함수를 이용해 jstring을 c string으로 변환한 후 출력한다.

**public native String WatchControl(void)**

역할 : fpga switch를 이용해 stop watch를 제어한다.

구현 방법 : JNI 함수를 이용해 1~4번 까지의 switch가 stopwatch의 버튼이 되도록 한다.

**class stopwatch extends Thread**

역할 : 매 초마다 stop watch의 count가 증가되도록 한다

구현 방법 : Thread의 sleep time을 1초로 하여, 매 초마다 count가 증가되도록 한다.

**class stopwatchSwitch extends Thread**

역할 : 사용자로부터 fpga switch를 받도록 한다

구현 방법 : Thread를 이용하여 매 0.01초마다 JNI 함수를 호출하여 fpga push switch로 부터 입력값을 받도록 한다.

**public boolean onKeyDown(int keyCode, android.view.KeyEvent event)**

역할 : 사용자가 물리 back 버튼을 누를 시 모든 동작을 중단하고 main activity로 간다.

구현 방법 : Android의 물리 back 버튼을 누를 때 thread의 flag를 false로 바꾸어 모든 동작을 중단 시키고 main activity로 돌아간다.

**PuzzleActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity의 기본 layout을 구성하고 각 버튼에 기능을 설정한다.

구현 방법 : 각 버튼에 onClickListener를 이용해 각 기능에 맞는 함수를 추가한다.

**public native void PuzzleCount(String time\_left)**

역할 : 남은 시간이 10초 미만일 경우 fpga dot에 남은 시간을 출력한다.

구현 방법 : thread에서 남은 시간이 10초 미만일 경우 JNI 함수를 호출하여 fpga dot driver에 write 한다.

**public native void PuzzleScoring(String score)**

역할 : fpga fnd에 최종 점수를 출력한다.

구현 방법 : puzzle 게임이 끝나면 합산된 점수를 JNI호출을 통해 board상의 fpga fnd에 출력하도록 한다.

**public void check\_finish(void)**

역할 : 게임이 끝났는지 여부를 판별한다.

구현 방법 : for문을 이용해 dynamic button 을 저장하고 있는 array를 확인한다.

**public void mix\_puzzle(int r, int c)**

역할 : 순서대로 정렬되어 있는 puzzle을 섞어 준다.

구현 방법 : random 변수를 만들어 switch 문으로 검은색으로 되어 있는 버튼이 상하좌우 중 움직일 수 있는 곳으로 이동한다. 이 작업을 1000번 반복하여 순서대로 정렬되어 있던 퍼즐을 사용자가 풀어야 하도록 섞어준다.

**public void create\_button(final int r, final int )**

역할 : 입력 받은 input에 맞는 버튼을 생성하고, 마지막 버튼은 검은색으로 칠한다.

구현 방법 : 2중 for문을 사용하여 버튼을 생성하고 생성한 버튼은 dynamic array에 저장하여 준다. 또한 각 버튼을 생성할 때 미리 준비해 둔 movement 알고리즘을 onClickListener를 이용해 등록하여 준다.

**public boolean onKeyDown(int keyCode, android.view.KeyEvent event)**

역할 : 사용자가 물리 back 버튼을 누를 시 모든 동작을 중단하고 main activity로 간다.

구현 방법 : Android의 물리 back 버튼을 누를 때 thread의 flag를 false로 바꾸어 모든 동작을 중단 시키고 main activity로 돌아간다.

**ModeActivity.java**

**public void onCreate(Bundle savedInstanceState)**

역할 : activity의 기본 layout을 구성하고 각 버튼에 기능을 설정한다.

구현 방법 : 각 버튼에 onClickListener를 이용해 각 기능에 맞는 함수를 추가한다.

public native String btnSwitch(void)

역할 : 사용자로부터 fpga switch input을 입력 받는다.

구현 방법 : Thread를 이용하여 매 0.01초 마다 JNI 함수를 호출하여 input데이터를 가져온다.

public native void printNumber(int count)

역할 : Modify 버튼을 눌렀을 때 fpga dot 에 해당 숫자를 출력해준다.

구현 방법 : Modify 버튼을 눌렀을 시에 JNI 함수를 호출하여 fpga dot device에 write 한다.

class switchThread extends Thread

역할 : 사용자로부터 fpga switch input을 받는다.

구현 방법 : Thread를 이용하여 매 0.01초마다 JNI 함수를 호출하여 사용자로부터 fpga switch input을 받는다. 이 데이터를 이용하여 application상의 버튼을 제어한다.

**public boolean onKeyDown(int keyCode, android.view.KeyEvent event)**

역할 : 사용자가 물리 back 버튼을 누를 시 모든 동작을 중단하고 main activity로 간다.

구현 방법 : Android의 물리 back 버튼을 누를 때 thread의 flag를 false로 바꾸어 모든 동작을 중단 시키고 main activity로 돌아간다.

**3. 시험 및 평가 내용:**

주어진 요구 사항에 대해 만족스러운 output을 만들어 내었다. JNI함수를 좀 더 세분화 시키지 못하여 코드가 난잡해졌지만, 성능이나 기능에 있어서는 별 다른 문제를 발견할 수 없었다. 또한 thread가 activity를 벗어나서도 돌고 있는 것을 방지 하기 위해 flag를 걸어 SW적/HW적 back버튼을 눌렀을 시에도 정상적으로 종료가 될 수 있도록 하여 추가적인 메모리 누수가 발생하지 않도록 하였다. Android application을 제작한 것이기 때문에 지속적인 업데이트를 통해 발견되는 문제 사항을 수정해 나아갈 수 있으며, 추가적인 새로운 기능 역시 언제든지 추가할 수 있다.

**V. 기타**

**1. 연구조원 기여도:**

20091648 이준호 : 100%

**2. 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.**

Android application을 개발한지 얼마 지나지 않아 바로 JNI를 이용해 Hardware를 직접 제어한다는 것이 너무나도 어렵게 다가왔다. 그렇기 때문에 프로젝트 초반에 큰 시간을 소모하였는데, 이론 적으로는 어떻게 작동하는지 알지만 실질적으로 구현을 하려 함에 있어서 많은 어려움이 있었다. 하지만 기존의 application틀에서 벗어나 좀 더 다양한 device를 제어할 수 있는 기회를 가진것은 큰 성과였던 것 같다. 이를 바탕으로 추후 새로운 device나 application을 개발함에 있어 많은 도움이 될 것 같다.