**Embedded 4st HW**

**(설계 프로젝트 수행 결과)**

**과목명: [CSE4116] 임베디드시스템소프트웨어**

**담당교수: 서강대학교 컴퓨터공학과 박 성 용**

**학번 및 이름: 20151543, 민지우**

**개발기간: 2019. 06. 20. -2019. 06. 21.**

**최 종 보 고 서**

**I. 개발 목표**

- 실습 시간 때 배운 내용을 활용하여 간단한 java application 프로그램을 작성한다..

**II. 개발 범위 및 내용**

**가. 개발 범위**

보드에서 실행시킬 수 있는 android application을 구현한다.

모든 버튼을 생성할 때 linear layout로 구현하도록 한다.

**나. 개발 내용**

퍼즐 프로그램 : java application로 입력한 수에 맞는 퍼즐을 생성한다.

make button을 누르기 시작한 이후의 퍼즐게임 플레이 시간을 나타내는 타이머를 Service로 구현한다.

**III. 추진 일정 및 개발 방법**

**가. 추진 일정**

1일 : java application을 만든다.

2일 : timer service을 만들어 연동한다.

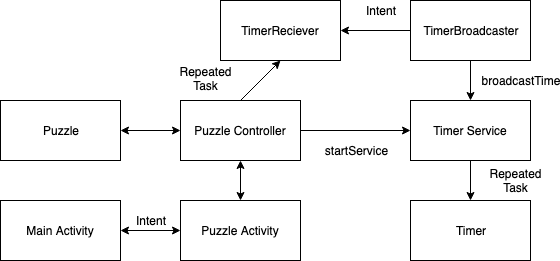
**나. 개발 방법**

Java application은 android studio을 사용하여 구현, 컴파일 결과는 device에서 확인하였다.

**IV. 연구 결과**

**1. 합성 내용:**

- 설계 목표에 필요한 내용을 조사 분석한 후 그들을 바탕으로 구성한 **전체 프로그램 구성도**.

****

앱 구성도

**2. 제작 내용:** 개발 결과

- 구현한 컴포넌트들에 대한 역할 및 구현 방법에 관한 내용을 기술할 것. (예: 자료구조, 알고리즘).

(1)Android application

- 화면 2개가 필요하기 때문에 activity 두개를 사용하였다. 타이머 서비스를 위해 android **service**와 custom timer를 구현하였다. 서비스와 Activity가 경과 시간이라는 메세지를 통신해야 하기 때문에 android의 **broadcaster**을 사용하였다.

-초기 화면(MainActivity)

\* linear layout을 이용해서 Button, TextView를 추가하였다.

\*java 파일에서는 onCreate함수에서 onClickListener과 intent를 사용해 서 버튼을 누를 시 Puzzle Activity로 화면을 전환하도록 하였다.

-퍼즐 화면(PuzzleActivity)

\*(activity\_puzzle.xml) : row, col을 입력 받을 EditText와 입력 button, 경과 시간을 표현할 textView를 추가하였다. 그리고 퍼즐을 들고 있을 linearLayout도 추가하였다

액티비티에서는 view단 즉 puzzle 게임의 로직보다는 ui와 관련된 method들이 정의되어 있다.

\*(PuzzleActivity.java) :

onCreate: puzzleController를 등록하고 start button에 puzzleController를 시작하는 listener을 추가한다.

MakeButtons: click listener와 puzzle model을 받아서 puzzle의 상태를

반영해 button을 만들고 listener을 등록한 뒤 PuzzleContainer에 추가한다.

clearPuzzle: puzzleContainer를 비운다

- 퍼즐 게임 (Puzzle Controller, Puzzle)

\* Puzzle: Puzzle의 상태를 저장하고 있음

현재 Puzzle의 크기, 값들을 java의 ArrayList를 이용해 배열로 저장하고 있음. Puzzle 게임을 구현한 class

checkFinished: 게임이 끝났는지 확인

isAvailableMove: 해당 index가 움직일 수 있는지 확인

\* Puzzle Controller: puzzle model과 puzzle activity (이하 view)을 연결하고 control한다. 퍼즐을 시작하거나 퍼즐 조각을 움직일 때 일어나야 하는 view의 작업들과 data 변경들을 관리하고 실행한다. TimerService또한 controller에서 제어한다. 그리고 TimeReceiver를 실행해 service로 부터 시간을 받아온다.

startPuzzle: 퍼즐 게임을 시작한다. View 에서는 버튼을 만들고, Puzzle모델을 이때 생성한다. 그리고 TimerService을 시작시키고, Receiver을 실행해 sec을 받을 때마다 view update하도록 한다.

endPuzzle: 퍼즐 게임을 종료한다. View에서는 puzzle을 clear하게 하고, 서비스를 종료하고, Receiver또한 종료시킨다.

- 타이머 서비스 (TimerService, Timer)

\* Timer: message handler을 사용해 custom handler을 제작하였다.

Java의 timer의 경우 android에서는 사용하는 것을 권장하지 않아서 제작하게 되었다. Second을 member 변수로 가지고 있어 timer가 tick 될 시 마다 증가된다. Timer 사용자에게 Repeated Task interface을 받아 tick 될 시 second을 증가시킨다.

\* TimerService: TimerService는 started service로 구현하였다. Start 될 시에 Timer을 실행한다. 그리고 종료할 때 해당 timer을 종료한다. 이 때 timer의 task 로서 TimeBroadcast의 broadcastTime을 실행하는 RepeatedTask 의 구현체를 등록한다. (아래항목에서 자세히 언급)

초반에 bindService로 구현해 tick될 때 마다 activity을 변경하게 하였으나 ui변경은 main thread에서만 일어나야 하기 때문에 tick 될 때 마다 main thread에게 변경하도록 메시지를 보내는 방식으로 변경하게 되었다.

- 타이머 브로드캐스터 (TimeBroadcaster, TimeReceiver)

\* TimeBroadcaster: 안드로이드의 broadcaster을 이용해 broadcast Receiver들에게 Intent을 보낸다. 이때 해당 Intent에는 Timer 서비스에서 넘겨주는 time이 들어있다. TCP통신에서의 서버 소켓과 같은 비슷함

\* TimeReceiver: broadcaster의 receiver. Broadcaster에서 Intent을 보낼 시에 이를 받아서 등록된 Repeated Task을 수행한다. 이때 callback 구조를 사용하므로 불필요하게 client단에서 thread을 만들 필요가 없었다. 그리고 activity의 main thread에서 ui을 변경할 수 있게 되었다. TCP 통신의 client 소켓과 비슷하다.

**3. 시험 및 평가 내용:**

- 평가 방법에 대한 설명을 기술하라.(수행 sequence 등)

- 자신들의 결과물이 갖는 **보건 및 안정**, **생산성과 내구성**에 대하여 반드시 기술할 것.

평가방법

\*row, col을 4이상 입력한다.

\*row, col을 입력 한 후 퍼즐이 생성되는지 확인한다.

\*퍼즐이 다 종료되지 않을 시에 makeButton 버튼을 눌러도 정상작동하는지 확인한다.

생산성

숫자로만 row,col을 제대로 입력하면 버튼이 만들어 지므로 생산성이 있다.

내구성

Button을 여러 번 눌러도 계속할 수 있으므로 내구성이 있다.

**V. 기타**

- 기타 관련 내용을 기술할 것.

**1.** **연구 조원 기여도**: 민지우 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.

서비스는 ui작업이 필요 없는 행위를 해야하는 데 해당 프로젝트에서는 서비스의 timer진행상황을 ui로 보여줘야 해서 service을 구현하는 데에 많은 어려움이 있었다. 그래서 수업 때 진행하지는 않았지만 안드로이드의 component끼리 메세지를 주고받을 때 사용하는 broadcast을 사용할 수 밖에 없었다. 해당 방법을 알아내는 데까지 정말 많은 시간이 필요했다.