마이크로프로세서응용실습

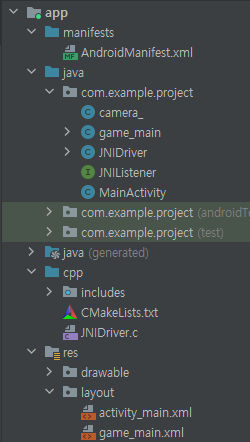
최종 과제 보고서

학과 : 전자전기컴퓨터공학부

학번 : 2017440104

이름 : 이승훈

## 프로젝트에 사용되거나 수정된 파일



위에서부터 camera\_.java는 카메라를, game\_main.java는 2번째 액티비티, JNIDriver와 JNIListener는 GPIO button을,

MainActivity는 메인화면, 그리고 아래는 드라이버를 위한 파일과, activity\_main.xml은 첫번 째 사진을 찍는 액티비티를

위한 xml파일, 두번 째 game\_main.xml은 두번 째 퍼즐을 하기 위한 액티비티의 xml파일이다.

프로젝트에 사용된 디바이스

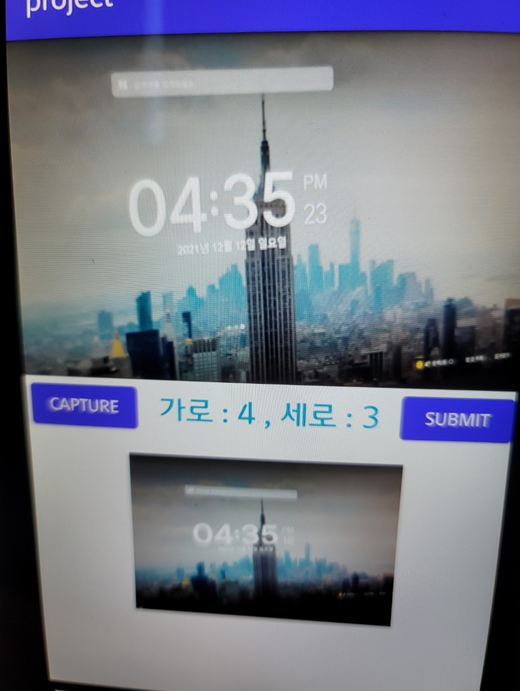
GPIO 버튼과, LED, 카메라, 7-segment가 사용되었다.

GPIO버튼은 첫번 째 화면에서 퍼즐의 잘리는 수를 결정한다.

LED는 정답인 자리에 있지 않은 퍼즐의 수를 의미한다.

카메라는 사진을 찍는데 사용되며, 7-segment는 100부터 시작해서 한번 움직일 때 마다 1씩 감소된다.(남은기회)

## 프로젝트 실행 화면



첫번 째 화면

첫번 째 액티비티의 화면입니다. 처음에 default로 가로 : 2, 세로 : 2로 정해져 있으며, GPIO버튼의 위쪽 버튼을 누르면

세로값이 1 증가, 아래 버튼을 누르면 세로값이 1 감소, 오른쪽 버튼을 누르면 가로가 1증가, 왼쪽을 누르면 가로가 1 감

소한다. 그 이후 SUBMIT버튼을 누르면 다음 액비티비로 값이 전달되며 두번 째 화면으로 바뀐다.



두번 째 화면

사진에서 잘 찍히지 않았지만, 7-segment는22를 가리킨다. 사진을 찍기 전에 100에서 78번 움직여서 22로 감소된 것이다.

원본 사진과 비교해보면, 검은색 빈칸을 제외한 원본 사진과 맞지 않는 자리에 있는 퍼즐의 수가 5개로, LED에 불이 5개가

들어온다. 퍼즐을 움직이는 방법은 검은색 바탕화면을 가진 버튼이 빈칸이며, 원본 사진을 가로, 세로로 자른 후 마지막 칸을

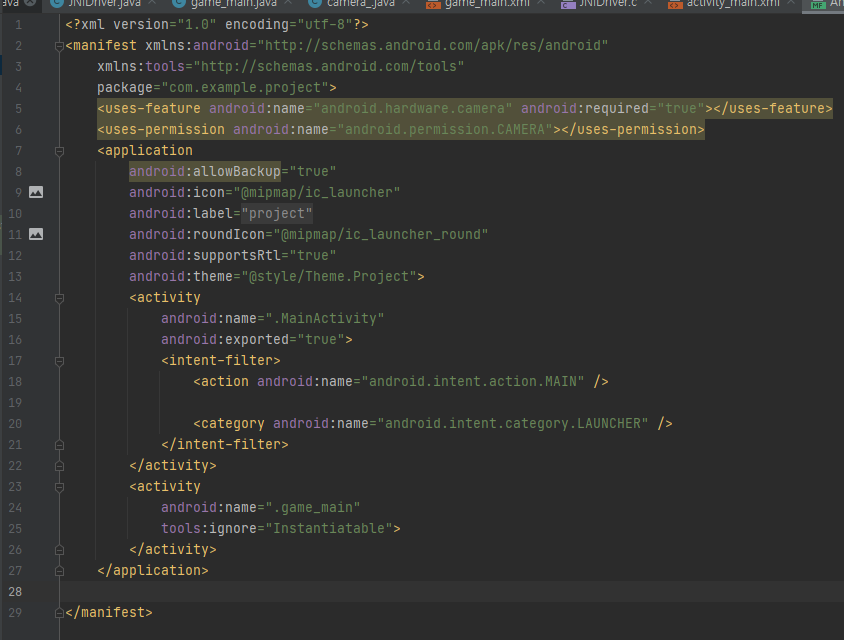
빈칸으로 정한다. 이 빈칸의 상,하,좌,우 에 해당하는 퍼즐을 클릭하면 서로의 자리가 바뀌게되며, segment의 숫자

가 1 줄어들어 기회를 1번 쓰게된다. 맨 위 TextView에 쓰인 4 x 3 퍼즐은 그때그때 가로, 세로에 잘린 퍼즐의 수에 따라 결

정된다.

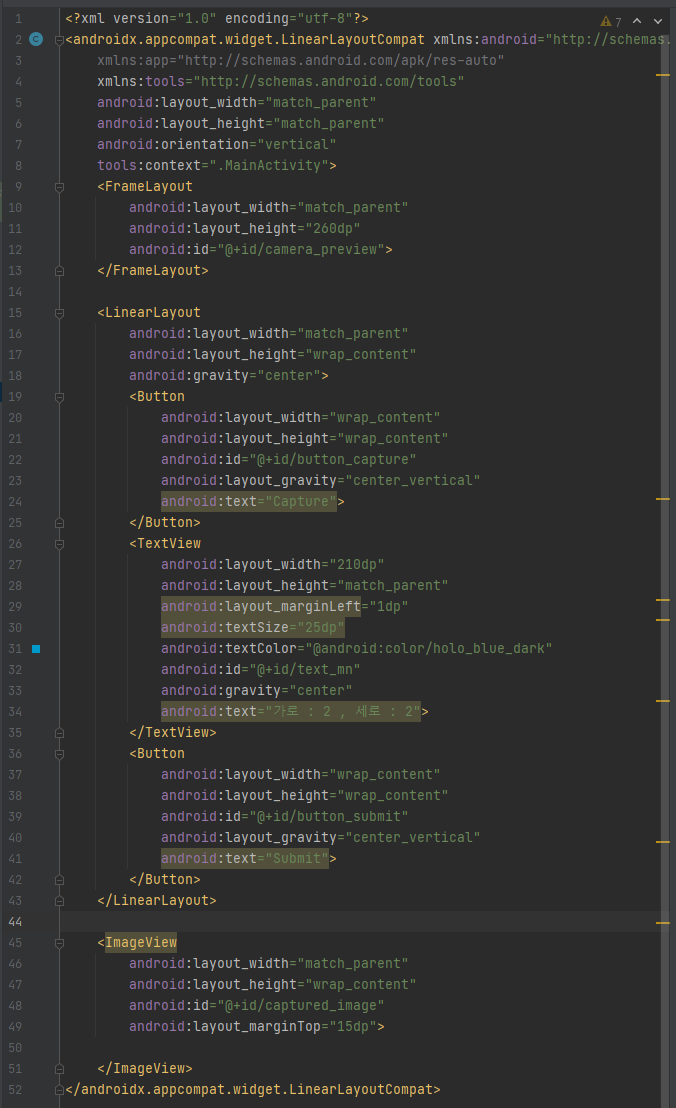
## 코드 설명

camera\_. java, JNIDriver, JNIListener, CMakeLists.txt는 여태까지 했던 실험들과 비슷합니다.



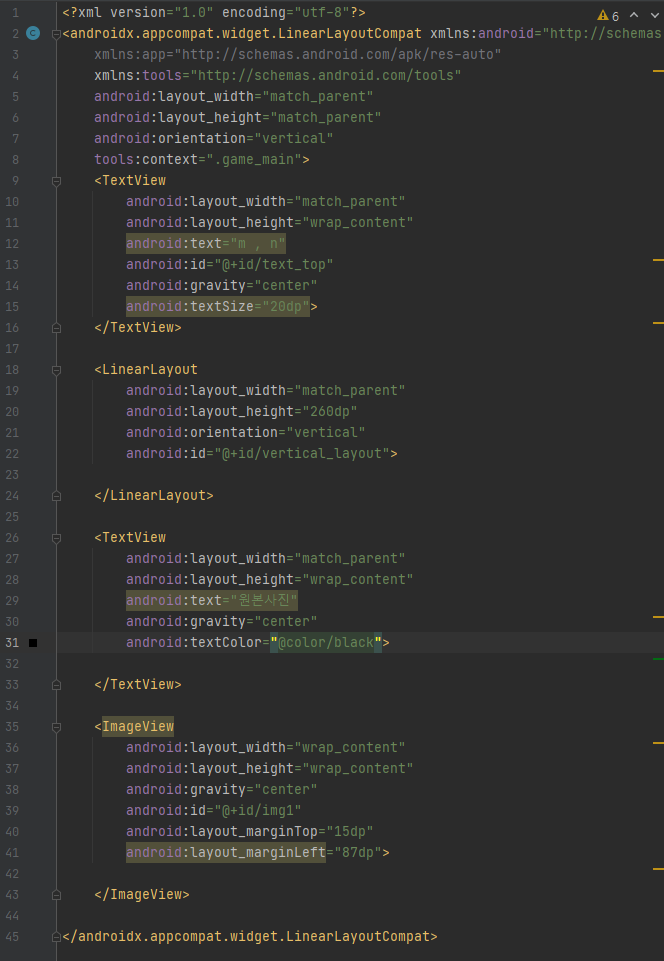
AndroidManifest.xml

카메라를 위한 권한, 액티비티가 1개만 사용하는게 아니라 2개를 사용하므로 액티비티를 추가해주었다.



activity\_main.xml 파일이다. 버튼은 Caputure버튼과 Submit버튼 2개이며, 그 사이에 TextView를 넣었다. 위와 아래는 현재

카메라에 담긴 모습과, 찍힌 사진이다.



game\_main.xml이다. 다른 부분에 해당하는 것들은 두번 째 화면이 찍힌 사진에서 볼 수 있으며, 가운데 있는 LinearLayout이

중요하다. 이는 game\_main.java파일에서 확인할 수 있다.



MainActivity의 Submit버튼 클릭 리스너

Submit 버튼을 누르면 Intent객체를 만들어 다음 액티비티로 값을 넘겨줄 준비를 한다. m, n 값과 사진을 찍은 image를 넘겨

준다. 만약, 사진이 찍혀져 있지 않았다면(한번도 capture버튼이 눌리지 않았다면) “사진을 찍어주세요”라는 텍스트가 뜨게

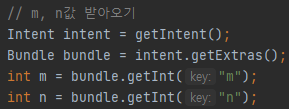
된다.



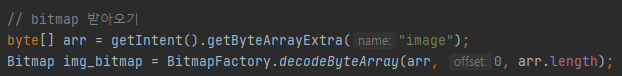
MainActivity에서 GPIO의 Handler부분

GPIO 버튼을 눌렀을 때 m, n값을 증가, 감소 시키는 코드이다. 중앙 버튼은 “Retry”라는 텍스트가 뜨게된다.

**Game\_main 액티비티**



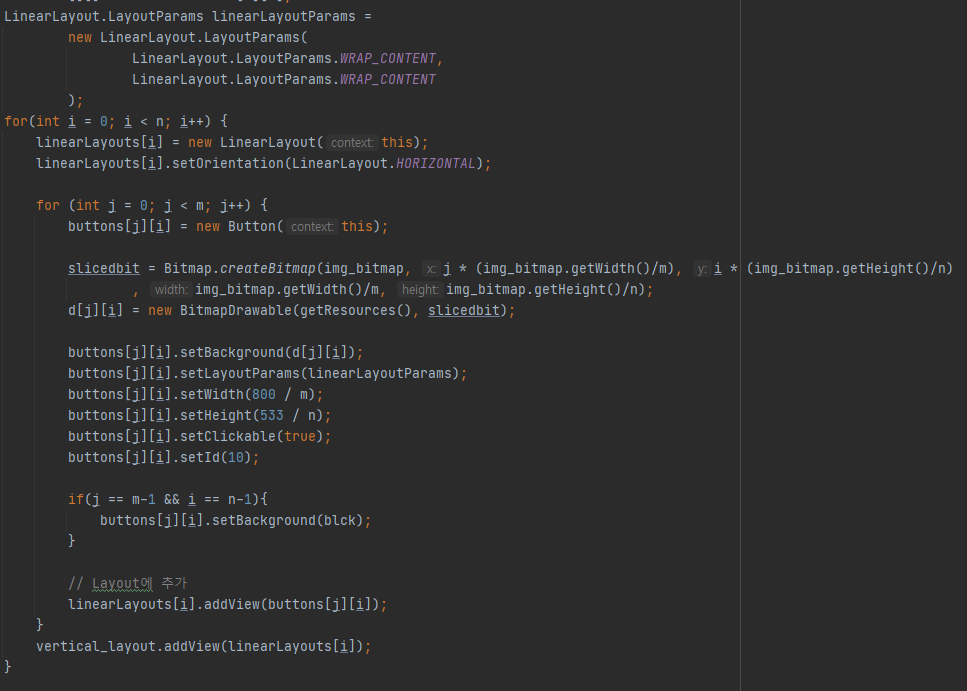
game\_main에서 m, n값을 받아오는 과정



game\_main에서 bitmap을 받아오는 과정

Bitmap의 경우 그냥 bitmap을 바로 전달하고 받아올 수 없으므로, ByteArray로 변경 후 받아왔다.

game\_main

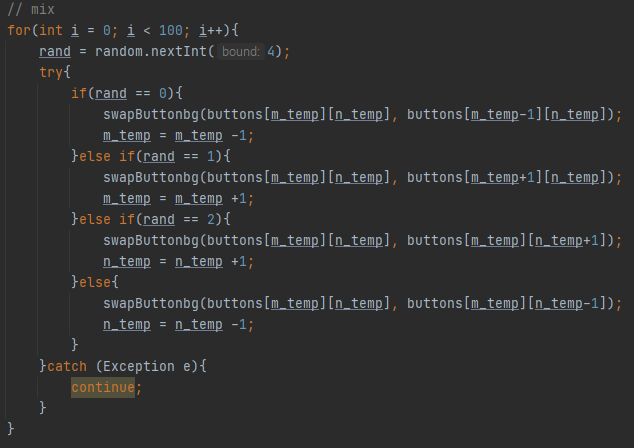


game\_main에서 xml파일의 중간 linearlayout

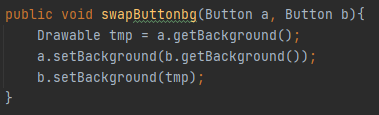
xml파일에서 중간에 있던 linear layout을 사용하는 과정이다. Orientation이 Vertical형인 Linear layout에 n개의 linear layout

을 추가해준다. 이는, 세로로 n칸을 잘랐다는 뜻이며, 각 n개의 linear layout에다가 m개만큼의 버튼을 추가해주는 과정이다.

또한, 마지막 버튼은 빈칸(배경이 검은색)으로 정한다.

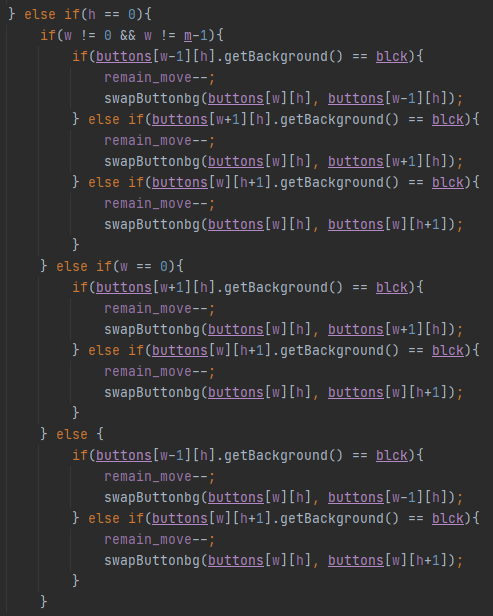
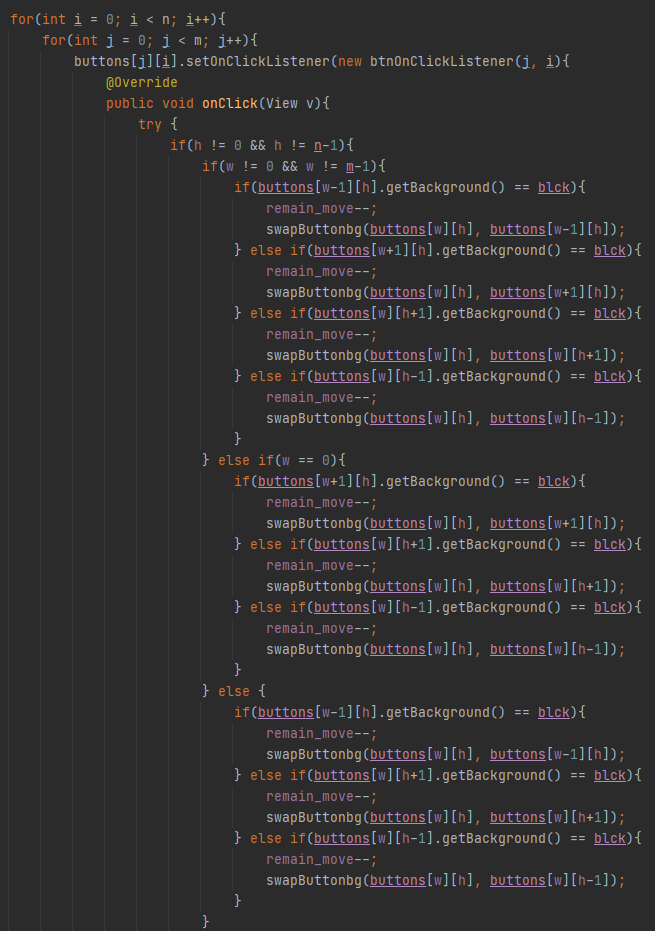


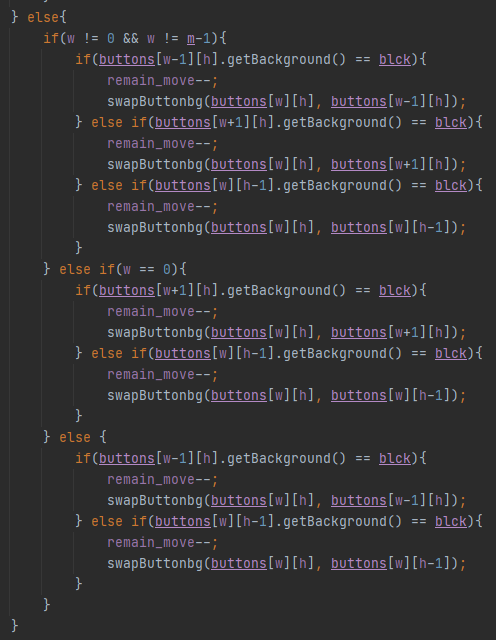
자른 사진들을 섞는 과정이다. swapButtonbg함수는 아래와 같이 구현하였다.



100번 섞으며, Index에 관한 오류가 생길 경우, 다시 실행하기 위해 try, catch문을 이용하였다.

다음은 각 버튼에 리스너를 다는 과정이다.





위 세 그림은 검은색 버튼의 상,하,좌,우 를 클릭하였을 때 서로의 위치를 바꾸는 과정이다. 처음에는 index관련 오류를 막기

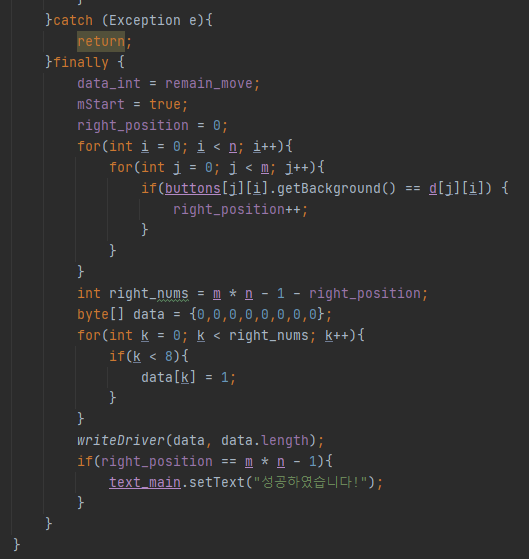
위해 try, catch문을 사용하였지만, 저렇게 상황을 하나하나 나눈 순간부터 따로 필요가 없긴하다. 처음에 try, catch문을 사용

한 이유로는 알아서 잡아주고 안될 줄 알았지만, 생각해보니 절차지향적으로 위에서 아래로 코드가 실행되고, 위에 검은색

바탕의 오른쪽에 해당하여 swap하는 경우가 있고, 그 아래 왼쪽에 해당하는 버튼과 swap하는 경우가 있다면, 만약 왼쪽에

해당하는 버튼과 swap을 해야하지만, 그 위에서 오류가 발생하여 catch문으로 이동되면 아래서 왼쪽 버튼과 swap이 되지

않는 경우가 발생한다.



다음으로 finally문이다. data\_int는 segment에 값을 표현하기 위한 변수이며, right\_position은 LED를 표시하기 위한 변수

이다. right\_position변수는 정답인 자리에 위치하는 퍼즐의 숫자이며, right\_nums변수를 정하는데 사용되며, right\_nums

변수값이 켜지는 LED의 숫자이다. remain\_move는 버튼이 클릭되어 서로 바뀔 떄마다 1씩 감소하며, segment에는 한번

씩 누를 떄마다 1씩 감소되는 것을 확인할 수 있다. 또한, 만약 모든 퍼즐을 다 맞출경우(빈칸을 제외한 모든 퍼즐이 정답)

맨 위의 텍스트가 “성공하였습니다!”가 출력된다.

참고문헌

LinearLayout 생성 관련 코드 : https://m.blog.naver.com/diceworld/220896628939

비트맵 전달과정 : https://kworks.tistory.com/377