**임베디드 시스템 LAB 4 실험보고서**



**팀 4**

**201501509 김주용**

**201601647 노윤표**

**201601659 이인호**

**201601663 장현빈**

목차

1. 과제

2. 과제 수행

2.1 개발 환경

2.2 기본적인 구현

2.2.1 하드웨어 설계 및 구현

2.2.2 코드 설계

2.2.2.1 User Interface

2.2.2.2 Header

2.2.2.3 Main

2.2.3 예외처리

3. 결론

1. 과제



저희 팀은 Touch – screen을 활용한 그림판 구현 과제를 선택하였습니다.

**2. 과제 수행**

**2.1 개발환경**

**1. Github**



팀원들과 서로 코드를 공유하고, 서로 피드백을 해 주기 위해서 Github를 활용하여 과제를 진행함

Github주소 : <https://github.com/soulsystem00/embedded-project>

위의 주소로 들어가면 전체 코드를 다운받을 수 있음

**2. Linux**

Odroid 연결 및 프로그램 실행을 위해 리눅스에 연결하여 과제를 진행

사용된 리눅스 버전 : 16.04.6 32bit 및 18.04.4 64bit

**3. GCC 버전**

gcc-4.9.real (Ubuntu/Linaro 4.9.2-0ubuntu1~14.04) 4.9.2

**4. 오드로이드 커널 버전**

Linux odroid 3.10.107 #2

SMP PREEMPT Tue Aug 20 06:08:00 PDT 2019 armv7l armv7lx

**2.2 기본적인 설계 및 구현**

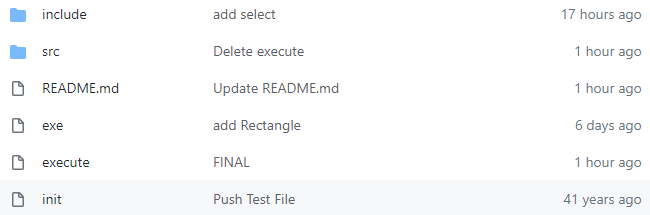
**하드웨어 설계**



추가적인 버튼과 같은 input 없이, 터치스크린만을 사용하기 때문에, Odroid 보드에 직접 TFT LCD를 연결한 형태로 하드웨어를 구성

**코드 구성 및 구조**

- 전체적인 파일 구성



**include** : 코딩에 필요한 헤더파일이 들어있는 디렉토리

**src** : 소스파일이 들어있는 디렉토리

**exe** : 코드의 컴파일 및 프로그램 실행에 관한 명령이 들어있는 파일

**init** : TFT LCD의 초기화에 관한 명령이 들어있는 파일

위의 기본적인 파일 구성을 마친 후 각자 파트를 분배하여 개발을 진행함

**2.3 파트 분배**

**201501509 김주용** : 본인이 한 일 작성

**201601647 노윤표** : 본인이 한 일 작성

**201601659 이인호** : 본인이 한 일 작성

**201601663 장현빈** : UI 구성, 전체적인 동작 및 FreeDraw, Select, Erase, Clear, Fill 함수 구현

**2.4 User Interface**

**2.4.1 Screen.c**

Direction에서 제공된 해당 그림과 유사한 UI를 구현하기 위해 Screen.c 파일에 320 \* 240 크기의 1차원 배열(76,800개의 index)로 구현함





- Screen.c

이러한 형태로 UI를 표현하기 위한 데이터가 저장되어 있으며, 이를 사용하기 위해 소스파일에서 출력을 위한 코드 작성을 진행

**2.4.2 색상 정보 초기화 및 출력**



main.c 에서 screen 배열의 데이터를 사용하기 위한 선언 및 우측 상단의 color를 표현하기 위한 픽셀 정보 초기화

위의 색상 정보는 전역변수로 선언 되어있어 그림을 그릴때도 이용됨

**배열 출력**



헤더 파일에 구현된 Screen 배열을 이용해 터치 스크린에 픽셀을 표현하는 PrintScreen 함수.

**색상 정보 출력**



우측 상단의 color 값을 표현하기 위해 직접 좌표를 이용해 픽셀 값을 수정하는 FillinitColor 함수

(8개 색상에 대한 좌표에 대해 모두 구현이 되어 있음, 코드는 동일하기 때문에 생략)



UI 디자인 구현을 완료 한 모습

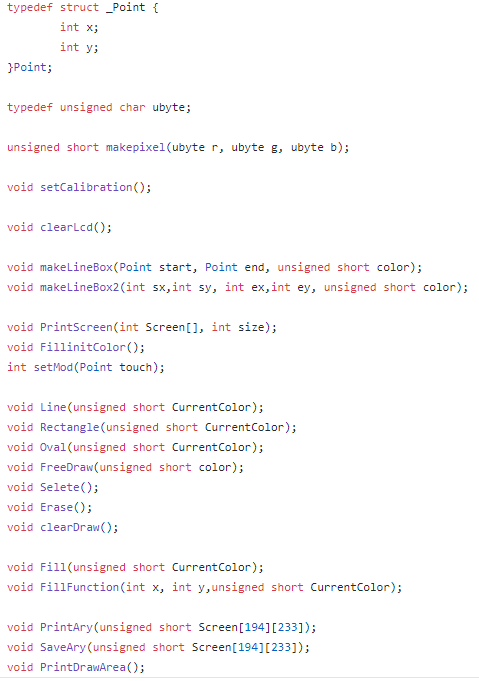
**2.5 기본적인 파일 설명**

필요한 모든 변수들과 함수들을 따로 정의하여 파일을 만듬.

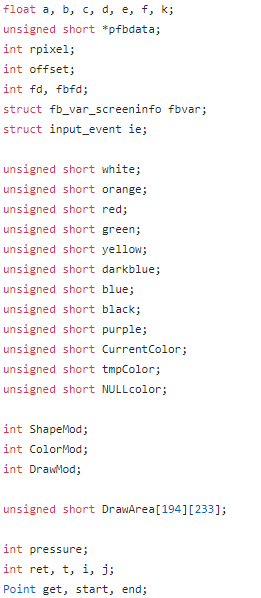
**Header.h** - 프로젝트에서 사용할 헤더들



**Function.h** – 프로젝트에서 사용할 함수들을 선언한 파일



**Var.h** – 프로젝트에서 사용할 전역변수들을 사용한 파일



**CommonFun.c** – 그리기 위해 사용되는 7개의 함수(Line, Rectangle, Oval, FreeDraw, Select, Erase, Clear)이외의 것들을 구현해 놓은 파일

ex)



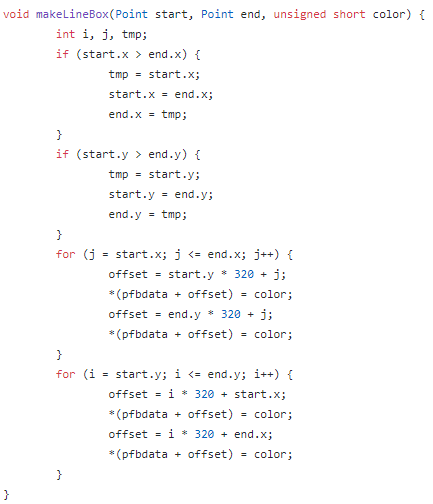
CommonFun.c에서 구현된 makepixel 함수와 setCalibration 함수.



CommonFun.c에서 구현된 clearLcd 함수

**DrawFunction.c -** 그리기 위해 사용되는 7개 함수들(Line, Rectangle, Oval, FreeDraw, Select, Erase, Clear)을 구현해 놓은 파일

ex)



DrawFunction.c에서 구현된 makeLineBox 함수

3 UI동작 및 코드

UI 출력이 아닌 UI 상의 동작들(그리기 모드 선택, 색상 선택 등)에 대한 설명 및 코드

UI는 총 3단계로 구성됨

1. 그리기 모드 선택

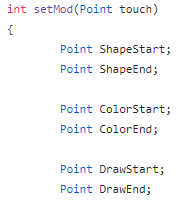
2. 색상 선택

3. 그리기

3.1 그리기 모드 선택

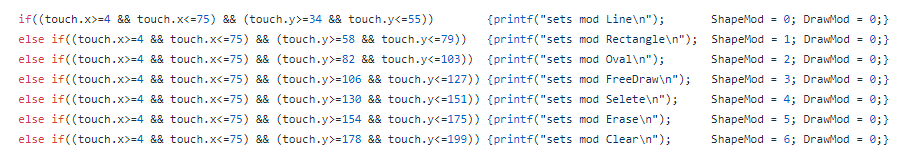
그리기 모드는 setMod 함수에 의해 동작함

setMod 함수는, 사용자가 터치 펜으로 특정한 기능을 선택할 때,(Line, Rectangle과 같은 버튼들을 터치 할 때) 해당 버튼의 위치에 맞는 기능들을 불러오는 함수입니다.



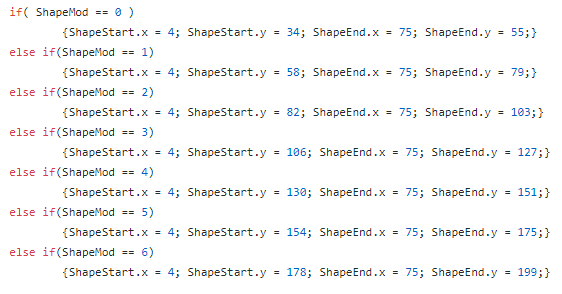
ShapeStart, ShapeEnd 구조체는 선택된 버튼에 대한 좌표 값을 저장

ColorStart, ColorEnd 구조체는 선택된 색상의 버튼에 대한 좌표 값을 저장

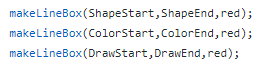
DrawStart, DrawEnd 구조체는 Pen과 Fill중 선택된 방법에 대한 버튼의 좌표 값을 저장

버튼들의 좌표와 비교해서 해당 위치에 맞는 기능을 동작하도록 변수를 설정

색상을 고르는 부분과, Pen, Fill에 대해서도 같은 방식으로 동작



선택된 버튼에 대해서 해당 버튼의 좌표가 선택되었다는 표시로, 따로 좌표 값을 대입해줍니다.



그리고 해당 좌표 값을 이용해 빨간색으로 색칠하여 사용자에게 어떤 버튼이 선택되었는지 시각적으로 보여주게 됩니다.(색상 버튼과 나머지 버튼들도 같은 방식으로 동작하게 됩니다.)

\* 기능 구현에 사용 된 알고리즘

저희의 LCD는 320 \* 240의 좌표로 이루어져 있습니다. 여기서, 직선을 긋는다면 시작점과 끝 점 사이에서 기울기를 구해서 그 기울기에 맞는 좌표들을 모두 찍어야 합니다. 일반적으로 기울기를 구하는 공식은,

(시작점 = (x1, y1) 끝점 = (x2, y2)



이러한 직선의 방정식을 떠올릴 수 있습니다. 하지만, 실수 연산이 모두 정수형으로 형 변환이 되기 때문에 이러한 식으로 직선을 구현한다면, 기울기가 정수인 경우에만 그려질 것입니다. 이를 해결하기 위해

‘ 브레슨 햄 알고리즘 ‘ 을 착안하였습니다.



저희는 LCD에 시작점과 끝 점 사이의 최소의 거리들을 모두 점으로 표현하고 싶은 것이 목적입니다. 즉 시각적으로 직선에 가장 가깝게 보이도록 표현해야 합니다. 이를 표현하기 위해서 흔한 표현으로, 위의 사진처럼 직선 렌더링 기법을 사용하기로 결정했습니다.

이를 표현하기 위해서는, 기울기가 1보다 작은 경우와, 기울기가 1보다 큰 두 가지 경우로 나뉩니다.



기울기가 1보다 클 때는, x좌표가 증가하는 기분 보단, y좌표가 증가하는 기준으로 점이 찍히게 됩니다. 기울기가 1보다 작은 경우는, 반대입니다.