임베디드 시스템 설계 및 실험

10주차

7조
201424470 서민영
201424421 김시은
201424533 정종진
201424532 정재광

목차

1. 실험목표 ····· 3
2. 배경지식 ····· 3 - 5
3. 실험과정 ····· 5 - 12
4. 소스코드 및 실험결과 ····· 12 - 26
5. 결론 및 느낀점 ····· 26

1. 실험목표

1). TFT LCD 제어

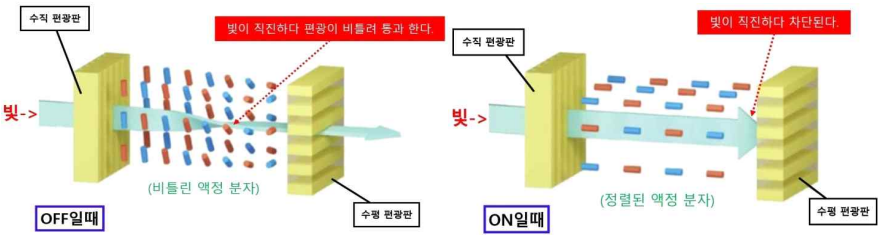
2. 배경지식

1). TFT LCD

1.1. LCD란

- Liquid Crystal Display : 액정 디스플레이
- Liquid Crystal : 평소에는 액체의 성질을 띄나 전압을 걸어주면 고체의 성질을 띄는 유기분자
- 고체와 액체의 중간적인 특성을 가지는 액정의 전기적 특성을 이용한 디스플레이
- 백라이트, 편광판, 액정, 컬러필터, 유리기판으로 구성
- 빛의 편광현상을 이용하여 원하는 빛을 표현
- LCD의 종류 : STN와 TFT
- STN : 가격은 싸지만 화질이 떨어져 보급형 제품에 많이 쓰인다.
- TFT : 속도가 빠르고 화질이 정밀해 고가의 노트북, 컴퓨터 등에 사용되나 가격이 비싸다.

1.2. LCD 동작원리

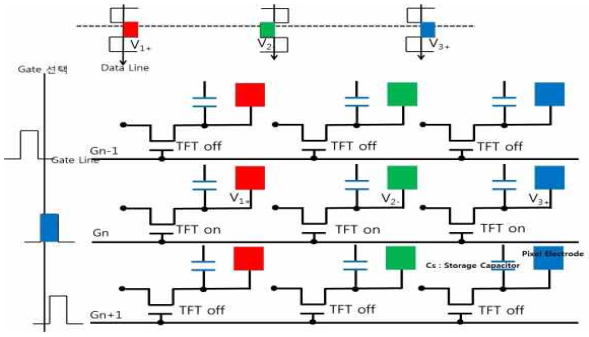


- 편광판을 수직이 되도록 배치
- 전압을 인가하지 않았을 때는 백라이트에서 나온 빛이 두 편광판을 통과하여 빛이 통과된다.
- 액정에 전압을 인가하면 특정방향(수직)으로 액정이 배열되고, 그 액정배열에 일치하는 특정 방향의 빛은 수평 편광판을 통과하지 못하므로 빛이 차단된다.
- 이렇게 액정분자의 배열을 제어하여 패널을 통과하는 빛의 양을 조절해서 원하는 빛을 표현한다.

1.3. LCD의 한계

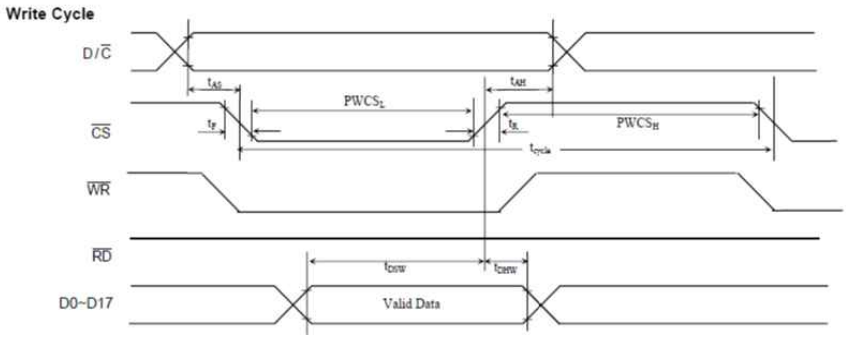
- 화질이 낮고 응답속도가 느리기 때문에 빠르게 움직이는 영상을 표시할 때 잔상이 발생한다.
- 각각의 픽셀을 제어하는데 어려움이 있다.

1.4. TFT LCD



- Transistor를 통해 각각의 픽셀을 개별적으로 제어 가능하여 매트릭스 형태로 픽셀을 제어하는 LCD보다 응답속도가 빨라 잔상이 적다.
- 속도가 빠르고 화질이 정밀해 고가의 노트북, 컴퓨터 등에 사용되나 가격이 비싸다.

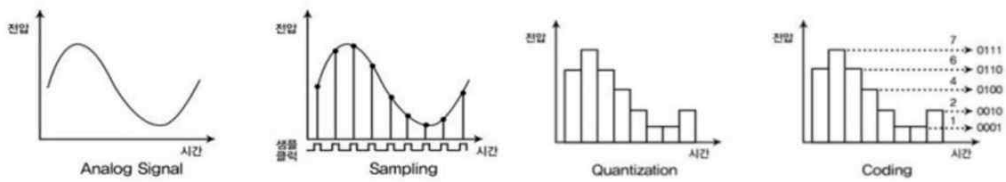
1.5. Timing Diagram



- LCD에 Data를 Write하기 위해서는 RD(Read Enable)라인에 HIGH, CS(Chip Select)라인에 LOW, WR(Write Enable)라인에 LOW를 주고 Data Line(D0~D17)에 데이터를 실으면 해당 데이터가 Write

2). ADC 와 DAC

- ADC(Analog to Digital Converter) : 아날로그 데이터를 디지털 데이터로 변환
- DAC(Digital to Analog Converter) : 디지털 데이터를 아날로그 데이터로 변환



2.1. ADC 1단계 (필터링)

- 본래의 신호를 정확히 Sampling하기 위해 잡음 등의 신호를 차단하는 과정
- Ex) 음성신호의 경우 원하는 대역폭에 대한 신호만 필터링한다.
- 나이퀴스트 정리(Nyquist theorem) • 최대 주파수 성분이 $fmax$인 아날로그 신호는 적어도 $2fmax$이상 의 sampling rate로 샘플링할 경우 원 신호를 완전히 복원할 수 있다.

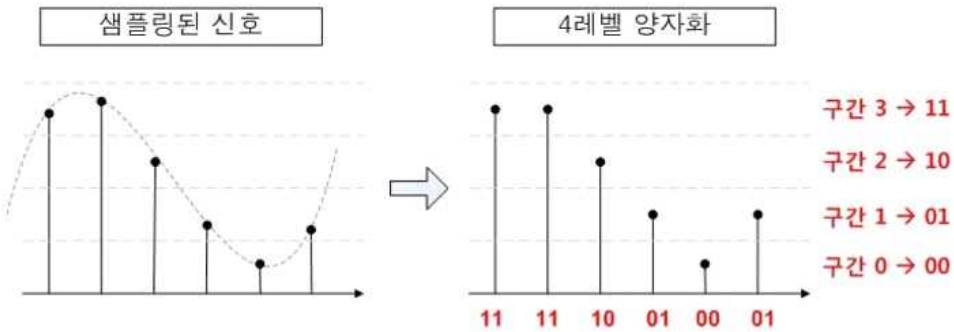
2.2. ADC 2단계 (표본화)

- 아날로그 파형을 디지털 형태로 변환하기 위해 표본을 취하는 것을 의미

- Sampling rate : 1초동안 취한 표본 수를 말하며, 단위는 주파수와 같은 Hz를 사용
- Sampling rate가 높아지면 본래 신호 특성을 더 잘 유지할 수 있지만, 데이터양이 많아진다.
- 나이퀴스트 주파수 (Nyquist Rate) = 최소 샘플링 주파수 • 표본화 정리에 따라 원래의 정보를 재생할 수 있도록 신호가 갖 는 최고 주파수($fmax$)의 두 배가 되는 표본화 주파수(fs)를 말함 • $fs = 2fmax$

2.3. ADC 3단계 (양자화)

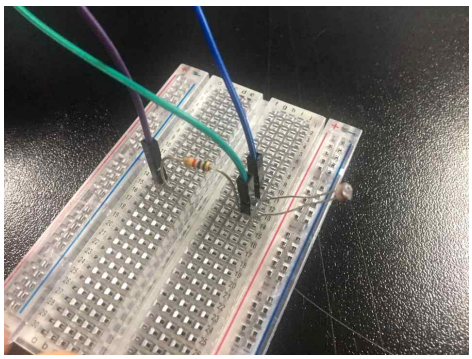
- 표본화에서 얻어진 수치를 대표 값으로 n개의 레벨로 분해하고, 샘플 값을 근사시키는 과정
- 디지털 형태로 표현할 때 어느 정도의 정밀도를 가지고 표현할 것인지를 의미
- 표본화 된 각 점에서 값을 표현하기 위해 사용되는 비트 수



2.4. ADC 4단계 (부호화)

- 양자화된 값을 비트로 변환

3). 조도센서



- 빛의 양에 따라 저항 값 변화
- 저항 값(아날로그 값)을 읽어와 ADC를 통해 디지털 값으로 변환

3. 실험과정

1) 보드 연결

<그림 1 : Coretex M3/JTAG/DSTREAM을 연결한 모습>

1-1) Coretex M3/JTAG/DSTREAM 연결

다음과 같은 순서로 보드를 연결한다. 이 때 연결 및 분리 순서를 제대로 지키지 않으면 장비가 망가질 수 있으니 주의해야한다.

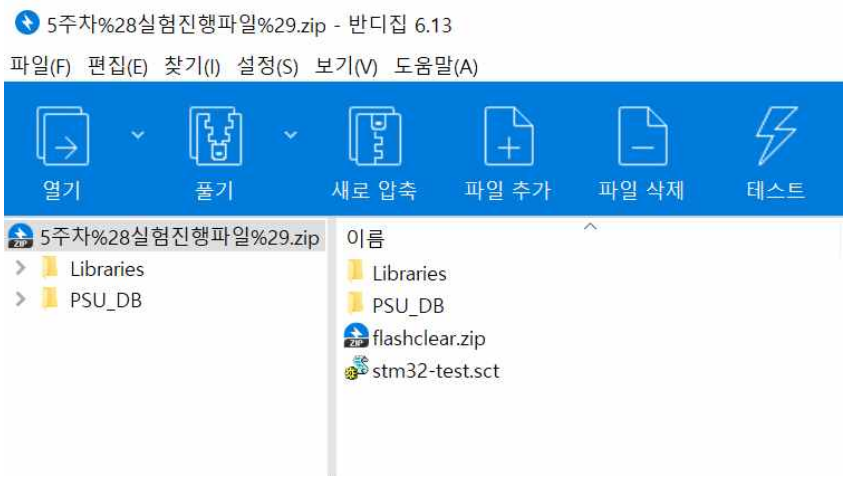
① 보드와 DSTREAM JTAG 연결
② 보드 전원선만 연결
(보드의 전원은 OFF 상태)
③ DSTREAM 전원 연결 및 ON
④ DSTREAM Status LED 점등 확인
⑤ 보드 전원 ON
⑥ DSTREAM Target LED 점등 확인
⑦ DS-5에서 'connect target'

<표 1 : 보드 연결 순서>

2) DS-5 디바이스 데이터베이스 추가

2-1) 수업게시판에서 실습파일로 제공되는 PSU_DB, LIBRARIES, SCATTER FILE을 다운

2-2) 이번 실험에서 include하는 코드의 양이 커서 기존에 scatter file에 설정된 값을 사용하면 load되지 않기 때문에 Data Sheet를 참고하여 scatter file의 메모리 범위를 수정해야 한다.



<그림 7 : 실습파일로 제공된 파일들>

```
LR_IROM1 0x08000000 0x00008000 {   : load region size_region
  ER_IROM1 0x08000000 0x00008000 {  : load address = execution address
   *.o(RESET, +First)
   *(InRoot$$Sections)  Data Sheet를 참고하여 0x00080000 으로 수정함
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x00008000 {  : RW data
   .ANY (+RW +ZI)
  }
}
```

2-3) Eclipse에 데이터베이스 추가

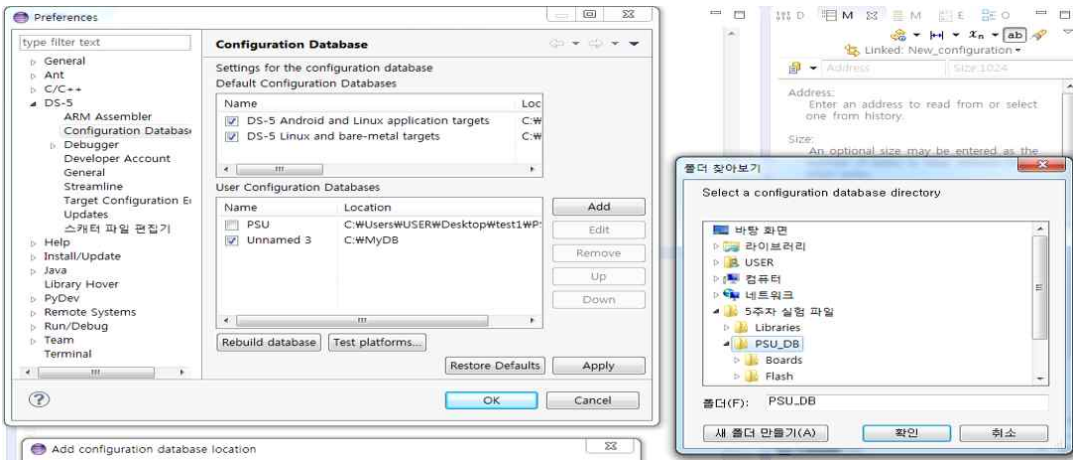
① 시작 → 모든 프로그램 → ARM DS-5 → Eclipse for DS-5 메뉴를 선택

② Windows → Preferences

③ DS-5 → Configuration Database 항목을 선택

④ Add 버튼을 클릭하여 사용자 데이터베이스의 디렉터리를 지정

⑤ Rebuild database 버튼을 클릭하여 데이터베이스 추가를 완료



<그림 8 : Eclipse에 데이터베이스 추가>

3) C Project 생성 및 환경설정

3-1) C Project 생성

① New Project → C project → Excutable → Empty Project를 선택해주고 Toolchains는 ARM Cmpiler 5(DS-5 built in)로 선택 후 프로젝트 생성

3-2) C Project Properties 설정

① C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → ARM Linker 5 → Image Layout → Scatter file 설정

(Scatter file에서 RO/RW base address를 지정해주므로 설정해줄 필요가 없음)

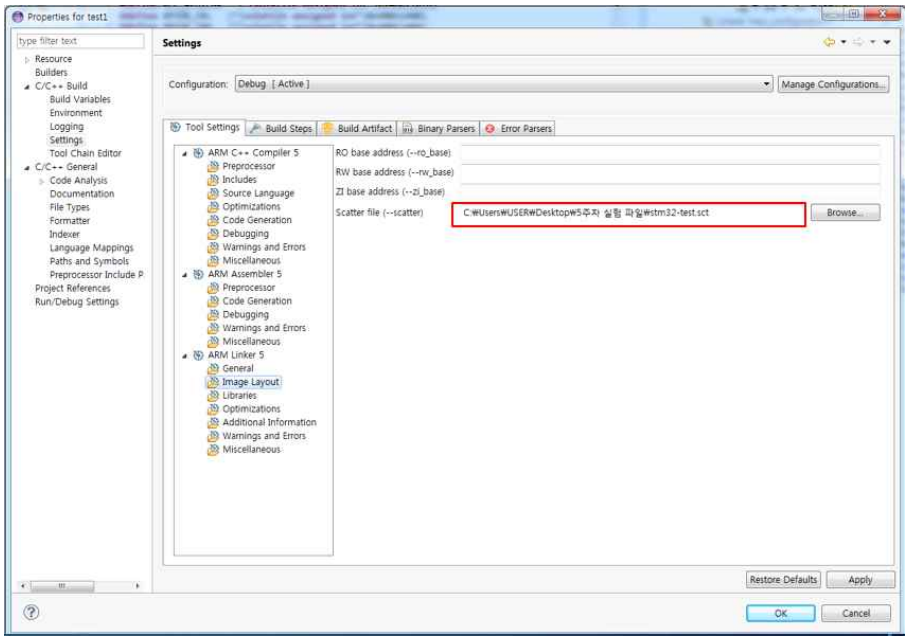
② C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → Code Generation과 General의 Target CPU를 Cortex M3로 설정

(entry point를 main으로 지정해주지 않아도 됨)

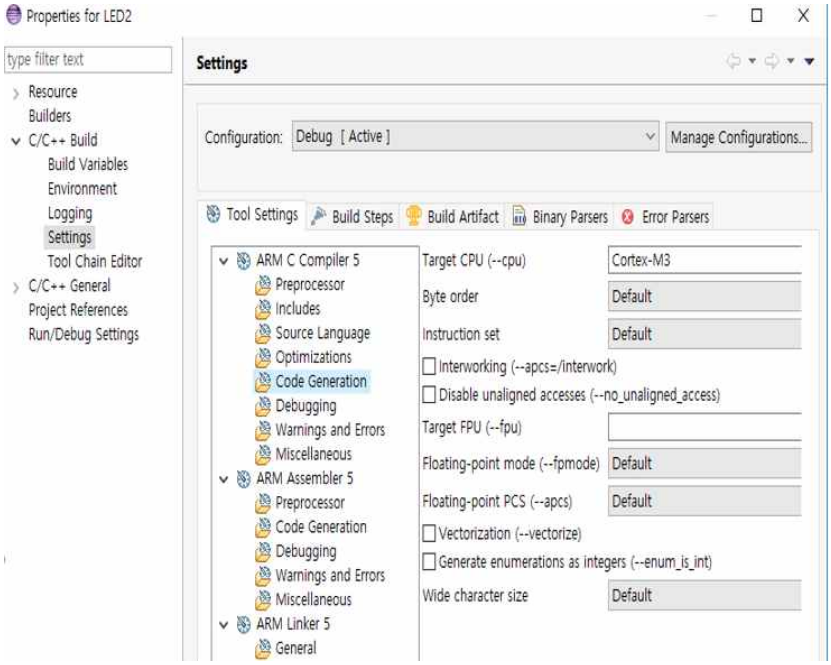
③ C project 우클릭 후 Properties 선택 → C/C++ Build → Settings -> Optimization에서 Optimization level을 High로 설정 (High로 설정 시 컴파일 할 때 코드의 크기가 작아짐)

3-3) LIBRARIES 추가

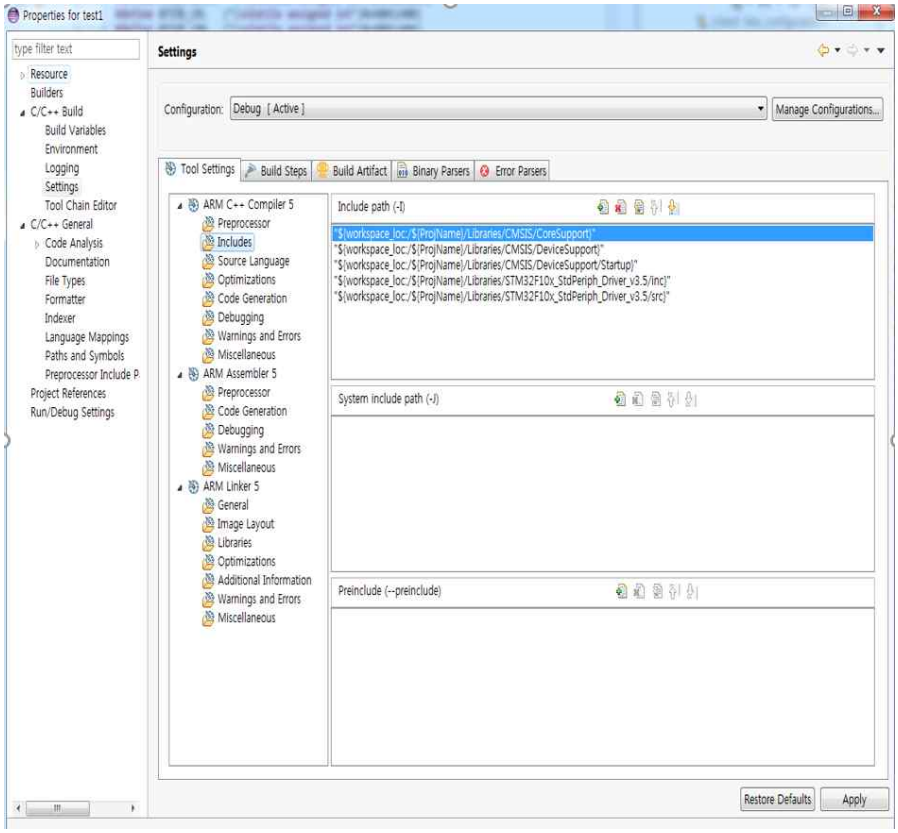
① C project 우클릭 후 Properties 선택 → C/C++ Build → Settings → Includes에서 제공되는 LIBRARIES 추가



<그림 9 : Scatter file 설정>



<그림 10 : Target CPU 설정>



<그림 11 : 제공되는 LIBRARIES 추가>

4) DS-5 Debugger 연결

4-1) Debug Configuration 설정

① Run → Debug Configuration 메뉴를 선택

② DS-5 Debugger 더블 클릭하여 새로운 하위 오브젝트 생성

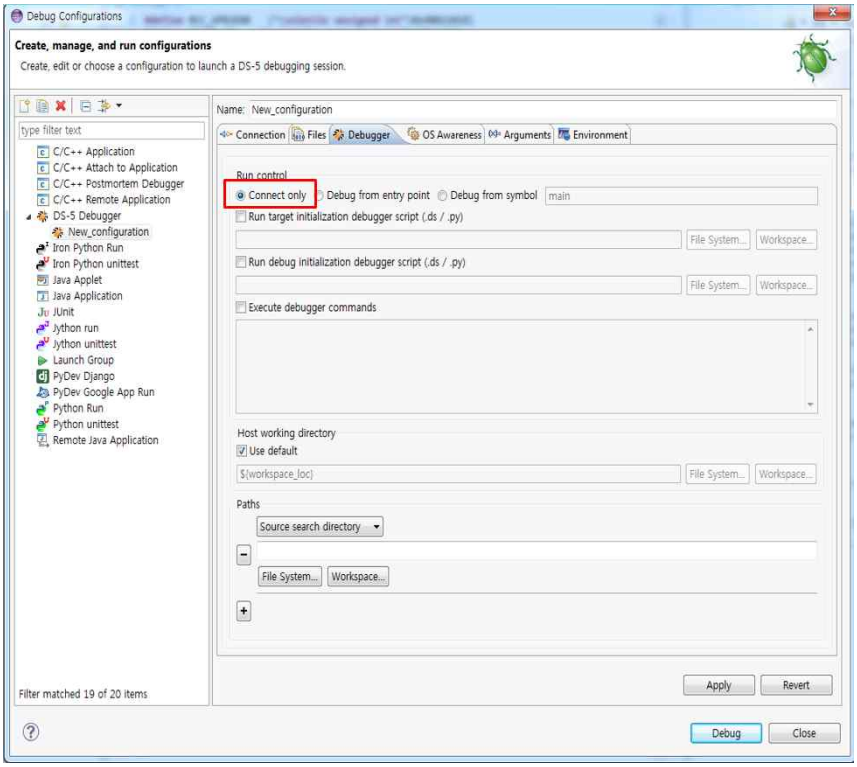
③ Name, Platform 등 Debug 환경설정을 변경

④ Browse 버튼을 클릭하여 DSTREAM 장비를 detection

4-2) Debug

① Debugger 탭에서 Connect only 체크

② Apply 클릭



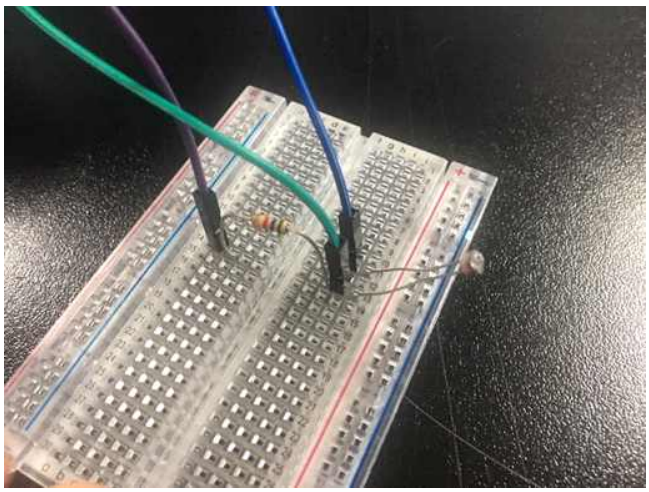
<그림 12 : Debug Configurations에서 Debugger탭의 설정화면>

5) C source code 작성

① LCD timing diagram을 참고하여 LCD.c에 TODO부분 소스코드를 작성

② main문에 RCC_Configure(), GPIO_Configure(), ADC_Configure() 및 while문 작성

6) 저항과 조도센서를 보드와 연결



7) C project 빌드 및 .axf 업로드

7-1) C project 빌드 및 Debug As

① C Project 빌드

(.axf 파일이 생성된 것을 확인할 수 있음)

② C Project 우클릭 → Debug → Debug As

7-2) flashclear.axf 및 생성된 .axf 업로드

① command 라인에 다음과 같은 명령어를 입력해서 flashclear.axf를 업로드
flash load “flashclear.axf 파일경로”

② disconnect한 다음 보드를 껐다가 켬

③ command 라인에 다음과 같은 명령어를 입력해서 생성된 .axf를 업로드
flash load “생성된 .axf 파일경로”

④ disconnect한 다음 보드를 껐다가 켬
(flash load 후에는 반드시 diconnect를 하고 보드를 껐다가 켜야 함)

7-3) 오실로스코프에 나타나는 파형 확인

① 조도센서에 빛을 비추면서 저항값의 변화를 관찰.

② 제대로 동작하지 않으면 5) C source code 작성으로 돌아감

8) 보드 연결 해체

앞서 보드 연결과 마찬가지로, 보드 연결 해체 시에도 순서를 제대로 지키지 않으면 보드가 망가질 수 있으므로 유의해야한다. 보드 연결 해체 순서는 다음과 같다.

① DS-5에서 ‘disconnect target’ ② 보드 전원 OFF ③ DSTREAM 전원 해제 및 OFF ④ 보드 전원선 분리 ⑤ DSTREAM과 보드 JTAG 분리

<표 2 : 보드 연결 해체 순서>

4. 작성한 소스코드 및 실험결과

1) 작성한 소스코드

team07.c

```
// flash load "C:\Users\Team07\Desktop\week10\team07\flashclear\flashclear.axf"
// flash load "C:\Users\Team07\Desktop\week10\team07\Debug\team07.axf"

#include "stm32f10x.h"
#include "core_cm3.h"
#include "misc.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"
#include "stm32f10x_adc.h"
#include "lcd.h"
#include "touch.h"

int                                                color[12]                                                =
{WHITE,CYAN,BLUE,RED,MAGENTA,LGRAY,GREEN,YELLOW,BROWN,BRRED,GRAY};

void RCC_Configure(void) {
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
```

```
}

void GPIO_Configure(void) {
    GPIO_InitTypeDef GPIO_InitStructure;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    GPIOD->CRL  = (GPIO_CRL_MODE2_0 | GPIO_CRL_MODE3_0 | GPIO_CRL_MODE4_0 |
GPIO_CRL_MODE7_0);

}

void ADC_Configure(void) {
    ADC_InitTypeDef ADC_InitStructure;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;

    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);

    ADC_Init(ADC1,&ADC_InitStructure);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1, ADC_SampleTime_239Cycles5);
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1)!=RESET); // return val?
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1)!=RESET); // return val?
}

void delay(int i){
    int j;
    for(j=0; j<=i * 100000; j++);
}

int button_pos[4][2];
int button[] = {0, 0, 0, 0};

int main() {

    int i, rSize = 30 , cSize = 15;

    unsigned int ledOn[] = { GPIO_BSRR_BS2, GPIO_BSRR_BS3, GPIO_BSRR_BS4,
            GPIO_BSRR_BS7 };
    unsigned int ledOff[] = { GPIO_BRR_BR2, GPIO_BRR_BR3, GPIO_BRR_BR4,
            GPIO_BRR_BR7 };
    char str[10];
    uint16_t pos_x,pos_y;
    uint16_t pix_x,pix_y;

    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    LCD_Clear(WHITE);

    RCC_Configure();
    GPIO_Configure();
```

```
    ADC_Configure();

    while(1){
        uint16_t temp;
        ADC_SoftwareStartConvCmd(ADC1, ENABLE); // enable?
        while(ADC_GetFlagStatus(ADC1, SET)!=RESET); // ADC_flag?? return???
        temp = ADC_GetConversionValue(ADC1);
        LCD_ShowString(100, 100, "TUE_TEAM07", BLACK, WHITE);
         LCD_ShowNum(100, 150, temp, 8, BLACK, WHITE);
         delay(1);
        ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);

        Touch_GetXY(&pos_x, &pos_y, 0);

        button_pos[0][0] = 120 * 0 + 30;
        button_pos[1][0] = 120 * 1 + 30;
        button_pos[2][0] = 120 * 0 + 30;
        button_pos[3][0] = 120 * 1 + 30;

        button_pos[0][1] = 200;
        button_pos[1][1] = 200;
        button_pos[2][1] = 280;
        button_pos[3][1] = 280;

        LCD_ShowString(120 * 0 + 30, 200, "LED1", BLACK, WHITE);
        LCD_ShowString(120 * 1 + 30, 200, "LED2", BLACK, WHITE);
        LCD_ShowString(120 * 0 + 30, 280, "LED3", BLACK, WHITE);
        LCD_ShowString(120 * 1 + 30, 280, "LED4", BLACK, WHITE);

        LCD_DrawRectangle(120 * 0 + 30 - rSize, 200 - cSize,
                          120 * 0 + 30 + rSize, 200 + cSize);
        LCD_DrawRectangle(120 * 1 + 30 - rSize, 200 - cSize,
                          120 * 1 + 30 + rSize, 200 + cSize);
        LCD_DrawRectangle(120 * 0 + 30 - rSize, 280 - cSize,
                          120 * 0 + 30 + rSize, 280 + cSize);
        LCD_DrawRectangle(120 * 1 + 30 - rSize, 280 - cSize,
                          120 * 1 + 30 + rSize, 280 + cSize);

        Convert_Pos(pos_x, pos_y, &pix_x, &pix_y);

        for (i = 0; i <= 3; i++) {
            if (button_pos[i][0] - rSize <= pix_x
                    && pix_x <= button_pos[i][0] + rSize
                    && button_pos[i][1] - rSize <= pix_y
                    && pix_y <= button_pos[i][1] + rSize) {
                button[i] = (button[i] + 1) % 2;
                if (button[i])
                    GPIOD->BSRR = ledOn[i]; // led 켜기
                else {
                    GPIOD->BRR = ledOff[i]; // led  끄기
                }
            }
        }

        Draw_Circle(pix_x, pix_y, 7);
        sprintf(str, "%d, %d", pix_x, pix_y);
        LCD_ShowString(100, 2, str, BLACK, WHITE);
        delay(1);
    }
```

```
}
```

LCD.c

```
#include "lcd.h"
#include "stdio.h"
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_gpio.h"
#include "font.h"
#include "stdlib.h"

/* Private variable ---------------------------------------------------------*/
uint16_t DeviceCode;

/* Private typedef ----------------------------------------------------------*/

/* private function---------------------------------------------------------- */

void LCD_RD(int n){
     if(n == 1) GPIO_SetBits(GPIOD, GPIO_Pin_15);
    else GPIO_ResetBits(GPIOD, GPIO_Pin_15);
}

void LCD_WR(int n){
    if(n == 1) GPIO_SetBits(GPIOD, GPIO_Pin_14);
    else GPIO_ResetBits(GPIOD, GPIO_Pin_14);
}

void LCD_CS(int n){
    if(n == 1) GPIO_SetBits(GPIOC, GPIO_Pin_6);
    else GPIO_ResetBits(GPIOC, GPIO_Pin_6);
}

void LCD_RS(int n){
    if(n == 1) GPIO_SetBits(GPIOD, GPIO_Pin_13);
    else GPIO_ResetBits(GPIOD, GPIO_Pin_13);
}

void LCD_WR_REG(uint16_t LCD_Reg)
{
    LCD_RD(1);
    LCD_RS(0);
    LCD_CS(0);
    LCD_WR(0);
    DataToWrite(LCD_Reg);
    LCD_CS(1);
    LCD_WR(1);
}

void LCD_WR_DATA(uint16_t LCD_Data)
{
     LCD_RD(1);
    LCD_RS(1);
    LCD_CS(0);
    LCD_WR(0);
    DataToWrite(LCD_Data);
    LCD_CS(1);
    LCD_WR(1);
```

```
}


void LCD_WriteReg(uint16_t LCD_Reg ,uint16_t LCD_RegValue)
{
   LCD_WR_REG(LCD_Reg);
   LCD_WR_DATA(LCD_RegValue);
}

void LCD_WriteRAM_Prepare(void)
{
   LCD_WR_REG(0x22);
}

void LCD_WriteRAM(u16 RGB_Code)
{
   LCD_WR_DATA(RGB_Code);
}

/**
   LCD_SetCursor(u16 Xpos, u16 Ypos)

 **/
void LCD_SetCursor(u16 Xpos, u16 Ypos)
{
   LCD_WriteReg(0x004E, Xpos);
   LCD_WriteReg(0X004F, Ypos);
}

/**
   LCD_Clear(uint16_t Colour)
 **/
void LCD_Clear(uint16_t Color)
{
   uint32_t index=0;
   LCD_SetCursor(0x00,0x0000);
   LCD_WriteRAM_Prepare();
   for(index=0;index<76800;index++)
   {
      LCD_WR_DATA(Color);
   }
}

/*****************************************************************************
 * LCD_GetPoint(u16 x,u16 y)

 * CurrentColor = LCD_GetPoint(10,10);

 *****************************************************************************/
//u16 LCD_GetPoint(u16 x,u16 y)
//{
//   LCD_SetCursor(x,y);
//   if(DeviceCode=0X8999)
//      return (LCD_ReadRAM());
//   else
//      return (LCD_BGRtoRGB(LCD_ReadRAM()));
//}

/**
 * LCD_DrawPoint(void)
```

```
 **/
void LCD_DrawPoint(uint16_t xsta, uint16_t ysta)
{
    LCD_SetCursor(xsta,ysta);
    LCD_WR_REG(0x22);
    LCD_WR_DATA(POINT_COLOR);
}

/**
    LCD_WindowMax()

 **/
void LCD_WindowMax (unsigned int x,unsigned int y,unsigned int x_end,unsigned int
y_end)
{
    LCD_WriteReg(0x44,x|((x_end-1)<<8));
    LCD_WriteReg(0x45,y);
    LCD_WriteReg(0x46,y_end-1);
}

/**
 * LCD_Fill(uint8_t xsta, uint16_t ysta, uint8_t xend, uint16_t yend, uint16_t colour)

 **/
void LCD_Fill(uint8_t xsta, uint16_t ysta, uint8_t xend, uint16_t yend, uint16_t colour)
{
    u32 n;

    LCD_WindowMax (xsta, ysta, xend, yend);
    LCD_SetCursor(xsta,ysta);
    LCD_WriteRAM_Prepare();
    n=(u32)(yend-ysta+1)*(xend-xsta+1);
    while(n--){LCD_WR_DATA(colour);}


    LCD_WindowMax (0, 0, 240, 320);
}

/**
 * LCD_DrawLine(uint16_t xsta, uint16_t ysta, uint16_t xend, uint16_t yend)

 **/
void LCD_DrawLine(uint16_t xsta, uint16_t ysta, uint16_t xend, uint16_t yend)
{
    u16 x, y, t;
    if((xsta==xend)&&(ysta==yend))LCD_DrawPoint(xsta, ysta);
    else if(abs(yend-ysta)>abs(xend-xsta))
    {
        if(ysta>yend)
        {
            t=ysta;
            ysta=yend;
            yend=t;
            t=xsta;
            xsta=xend;
            xend=t;
        }
        for(y=ysta;y<yend;y++)
```

```
        {
            x=(u32)(y-ysta)*(xend-xsta)/(yend-ysta)+xsta;
            LCD_DrawPoint(x, y);
        }
    }
    else
    {
        if(xsta>xend)
        {
            t=ysta;
            ysta=yend;
            yend=t;
            t=xsta;
            xsta=xend;
            xend=t;
        }
        for(x=xsta;x<=xend;x++)
        {
            y =(u32)(x-xsta)*(yend-ysta)/(xend-xsta)+ysta;
            LCD_DrawPoint(x,y);
        }
    }
}

/**
 * Draw_Circle(uint16_t x0, uint16_t y0, uint8_t r)

 **/
void Draw_Circle(uint16_t x0, uint16_t y0, uint8_t r)
{
    int a,b;
    int di;
    a=0;b=r;
    di=3-(r<<1);
    while(a<=b)
    {
        LCD_DrawPoint(x0-b,y0-a);             //3
        LCD_DrawPoint(x0+b,y0-a);             //0
        LCD_DrawPoint(x0-a,y0+b);             //1
        LCD_DrawPoint(x0-b,y0-a);             //7
        LCD_DrawPoint(x0-a,y0-b);             //2
        LCD_DrawPoint(x0+b,y0+a);             //4
        LCD_DrawPoint(x0+a,y0-b);             //5
        LCD_DrawPoint(x0+a,y0+b);             //6
        LCD_DrawPoint(x0-b,y0+a);
        a++;

        if(di<0)di +=4*a+6;
        else
        {
            di+=10+4*(a-b);
            b--;
        }
        LCD_DrawPoint(x0+a,y0+b);
    }
}

/**
 * LCD_DrawRectangle(uint16_t xsta, uint16_t ysta, uint16_t xend, uint16_t yend)
```

```
 **/
void LCD_DrawRectangle(uint16_t xsta, uint16_t ysta, uint16_t xend, uint16_t yend)
{
    LCD_DrawLine(xsta,ysta,xend,ysta);
    LCD_DrawLine(xsta,ysta,xsta,yend);
    LCD_DrawLine(xsta,yend,xend,yend);
    LCD_DrawLine(xend,ysta,xend,yend);
}

/**
 * LCD_ShowChar(u8 x, u16 y, u8 num, u8 size, u16 PenColor, u16 BackColor)

 **/
void LCD_ShowChar(u8 x, u16 y, u8 num, u8 size, u16 PenColor, u16 BackColor)
{
#define MAX_CHAR_POSX 232
#define MAX_CHAR_POSY 304
    u8 temp;
    u8 pos,t;
    if(x>MAX_CHAR_POSX||y>MAX_CHAR_POSY)return;
    LCD_WindowMax(x,y,x+size/2,y+size);
    LCD_SetCursor(x, y);

    LCD_WriteRAM_Prepare();
    num=num-' ';
    for(pos=0;pos<size;pos++)
    {
        if(size==12)
            temp=asc2_1206[num][pos];
        else
            temp=asc2_1608[num][pos];
        for(t=0;t<size/2;t++)
        {
            if(temp&0x01)
            {
                LCD_WR_DATA(PenColor);
            }
            else
                LCD_WR_DATA(BackColor);
            temp>>=1;
        }
    }
    LCD_WindowMax(0x0000,0x0000,240,320);
}


u32 mypow(u8 m,u8 n)
{
    u32 result=1;
    while(n--)result*=m;
    return result;
}

void LCD_ShowNum(u8 x,u8 y,u32 num,u8 len, u16 PenColor, u16 BackColor)
{
    u8 size = 16;
    u8 t,temp;
    u8 enshow=0;
    for(t=0;t<len;t++)
```

```
    {
        temp=(num/mypow(10,len-t-1))%10;
        if(enshow==0&&t<(len-1))
        {
            if(temp==0)
            {
                LCD_ShowChar(x+(size/2)*t,y,' ',size, PenColor, BackColor);
                continue;
            }else enshow=1;

        }
        LCD_ShowChar(x+(size/2)*t,y,temp+'0',size, PenColor, BackColor);
    }
}

/**
    LCD_ShowCharString(uint16_t x, uint16_t y, const uint8_t *p, uint16_t PenColor,
uint16_t BackColor)

 **/
void LCD_ShowCharString(uint16_t x, uint16_t y, const uint8_t *p, uint16_t PenColor,
uint16_t BackColor)
{
    uint8_t size = 16;

    if(x>MAX_CHAR_POSX){x=0;y+=size;}
    if(y>MAX_CHAR_POSY){y=x=0;LCD_Clear(WHITE);}
    LCD_ShowChar(x, y, *p, size, PenColor, BackColor);
}

/**
 * findHzIndex(u8 *hz)

 **/
u16 findHzIndex(u8 *hz)
{
    u16 i=0;
    FNT_GB16 *ptGb16 = (FNT_GB16 *)GBHZ_16;
    while(ptGb16[i].Index[0] > 0x80)
    {
        if ((*hz == ptGb16[i].Index[0]) && (*(hz+1) == ptGb16[i].Index[1]))
        {
            return i;
        }
        i++;
        if(i > (sizeof((FNT_GB16 *)GBHZ_16) / sizeof(FNT_GB16) - 1))
        {
            break;
        }
    }
    return 0;
}

/**
 * WriteOneHz(uint16_t x0, uint16_t y0, uint8_t *pucMsk, uint16_t PenColor, uint16_t
BackColor)

 **/
void WriteOneHz(u16 x0, u16 y0, u8 *pucMsk, u16 PenColor, u16 BackColor)
{
```

```
    u16 i,j;
    u16 mod[16];
    u16 *pusMsk;
    u16 y;

    u16 size = 16;

    pusMsk = (u16 *)pucMsk;


    for(i=0; i<16; i++)
    {
        mod[i] = *pusMsk;
        mod[i] = ((mod[i] & 0xff00) >> 8) | ((mod[i] & 0x00ff) << 8);
        pusMsk = pusMsk+1;
    }
    y = y0;
    LCD_WindowMax(x0,y0,x0+size,y0+size);
    LCD_SetCursor(x0,y0);
    LCD_WriteRAM_Prepare();
    for(i=0; i<16; i++)
    {
        for(j=0; j<16; j++)
        {
            if((mod[i] << j) & 0x8000)
            {
                LCD_WriteRAM(PenColor);
            }
            else
            {
                LCD_WriteRAM(BackColor);
            }
        }
        y++;
    }
    LCD_WindowMax(0x0000,0x0000,240,320);
}

/**
 * LCD_ShowHzString(u16 x0, u16 y0, u8 *pcStr, u16 PenColor, u16 BackColor)

 **/
void LCD_ShowHzString(u16 x0, u16 y0, u8 *pcStr, u16 PenColor, u16 BackColor)
{
#define MAX_HZ_POSX 224
#define MAX_HZ_POSY 304
    u16 usIndex;
    u8 size = 16;
    FNT_GB16 *ptGb16 = 0;
    ptGb16 = (FNT_GB16 *)GBHZ_16;

    if(x0>MAX_HZ_POSX){x0=0;y0+=size;}
    if(y0>MAX_HZ_POSY){y0=x0=0;LCD_Clear(WHITE);}

    usIndex = findHzIndex(pcStr);
    WriteOneHz(x0, y0, (u8 *)&(ptGb16[usIndex].Msk[0]),  PenColor, BackColor);
}

/**
 * LCD_ShowString(u16 x0, u16 y0, u8 *pcstr, u16 PenColor, u16 BackColor)
```

```
 **/
void LCD_ShowString(u16 x0, u16 y0, u8 *pcStr, u16 PenColor, u16 BackColor)
{
    while(*pcStr!='\0')
    {
        if(*pcStr>0x80)
        {
            LCD_ShowHzString(x0, y0, pcStr, PenColor, BackColor);
            pcStr += 2;
            x0 += 16;
        }
        else
        {
            LCD_ShowCharString(x0, y0, pcStr, PenColor, BackColor);
            pcStr +=1;
            x0+= 8;
        }

    }

}

u16 LCD_RGBtoBGR(u16 Color)
{
    u16  r, g, b, bgr;

    b = (Color>>0)  & 0x1f;
    g = (Color>>5)  & 0x3f;
    r = (Color>>11) & 0x1f;

    bgr =  (b<<11) + (g<<5) + (r<<0);

    return( bgr );
}


void LCD_DrawPicture(u16 StartX,u16 StartY,u16 Xend,u16 Yend,u8 *pic)
{
    static    u16 i=0,j=0;
    u16 *bitmap = (u16 *)pic;

    LCD_WindowMax(StartX, StartY, Xend, Yend);
    LCD_SetCursor(StartX,StartY);
    LCD_WriteRAM_Prepare();
    for(j=0; j<Yend-StartY; j++)
    {
        for(i=0; i<Xend-StartX; i++) LCD_WriteRAM(*bitmap++);
    }

    LCD_WindowMax(0, 0, 240, 320);
}

void LCD_Configuration(void)
{
    /* LCD 占쏙옙占쏙옙 GPIO 占쏙옙占쏙옙 */
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD
            |RCC_APB2Periph_GPIOE, ENABLE);
```

```
    /*DB00~DB16*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 |GPIO_Pin_3
          | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7
          | GPIO_Pin_8 | GPIO_Pin_9 | GPIO_Pin_10 |GPIO_Pin_11
          | GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOE, &GPIO_InitStructure);


    /* LCD_RS  LCD_WR  LCD_RD*/
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOD, &GPIO_InitStructure);


    /* LCD_CS */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}


/**
 * LCD_Init()

 **/

void Delay_10ms(int time) {
    int i,j;
    for(i = 0;i < time; i++) {
        for(j=0;j<1140;j++){}}
}

void LCD_Init(void)
{
    LCD_Configuration();

    /* Delay_10ms() 占쌉쇽옙 占쏙옙占쏙옙*/
    Delay_10ms(10); /* delay 100 ms */
    Delay_10ms(10); /* delay 100 ms */
//    DeviceCode = LCD_ReadReg(0x0000);
    Delay_10ms(10); /* delay 100 ms */

    {
        /*-----    Start Initial Sequence ------*/
        LCD_WriteReg(0x00, 0x0001); /*Start internal OSC */
        LCD_WriteReg(0x01, 0x3B3F); /*Driver output control */
        LCD_WriteReg(0x02, 0x0600); /* set 1 line inversion    */
        /*-------- Power control setup --------*/
        LCD_WriteReg(0x0C, 0x0007); /* Adjust VCIX2 output voltage */
        LCD_WriteReg(0x0D, 0x0006); /* Set amplitude magnification of VLCD63 */
        LCD_WriteReg(0x0E, 0x3200); /* Set alternating amplitude of VCOM */
        LCD_WriteReg(0x1E, 0x00BB); /* Set VcomH voltage */
        LCD_WriteReg(0x03, 0x6A64); /* Step-up factor/cycle setting  */
        /*-------- RAM position control --------*/
        LCD_WriteReg(0x0F, 0x0000); /* Gate scan position start at G0 */
        LCD_WriteReg(0x44, 0xEF00); /* Horizontal RAM address position */
```

```
        LCD_WriteReg(0x45, 0x0000); /* Vertical RAM address start position*/
        LCD_WriteReg(0x46, 0x013F); /* Vertical RAM address end position */
        /* ------ Adjust the Gamma Curve -------*/
        LCD_WriteReg(0x30, 0x0000);
        LCD_WriteReg(0x31, 0x0706);
        LCD_WriteReg(0x32, 0x0206);
        LCD_WriteReg(0x33, 0x0300);
        LCD_WriteReg(0x34, 0x0002);
        LCD_WriteReg(0x35, 0x0000);
        LCD_WriteReg(0x36, 0x0707);
        LCD_WriteReg(0x37, 0x0200);
        LCD_WriteReg(0x3A, 0x0908);
        LCD_WriteReg(0x3B, 0x0F0D);
        /*--------- Special command -----------*/
        LCD_WriteReg(0x28, 0x0006); /* Enable test command    */
        LCD_WriteReg(0x2F, 0x12EB); /* RAM speed tuning     */
        LCD_WriteReg(0x26, 0x7000); /* Internal Bandgap strength */
        LCD_WriteReg(0x20, 0xB0E3); /* Internal Vcom strength */
        LCD_WriteReg(0x27, 0x0044); /* Internal Vcomh/VcomL timing */
        LCD_WriteReg(0x2E, 0x7E45); /* VCOM charge sharing time */
        /*--------- Turn On display ------------*/
        LCD_WriteReg(0x10, 0x0000); /* Sleep mode off */
        Delay_10ms(3);                  /* Wait 30mS  */
        LCD_WriteReg(0x11, 0x6870); /* Entry mode setup. 262K type B, take care on the
data bus with 16it only */
        LCD_WriteReg(0x07, 0x0033); /* Display ON    */
    }
    Delay_10ms(5);
    LCD_Clear(BLACK);
}
```

※touch.c는 주어진 파일을 수정하지 않고 그대로 활용하였으므로, 따로 소스코드를 첨부안함

2) 실험결과

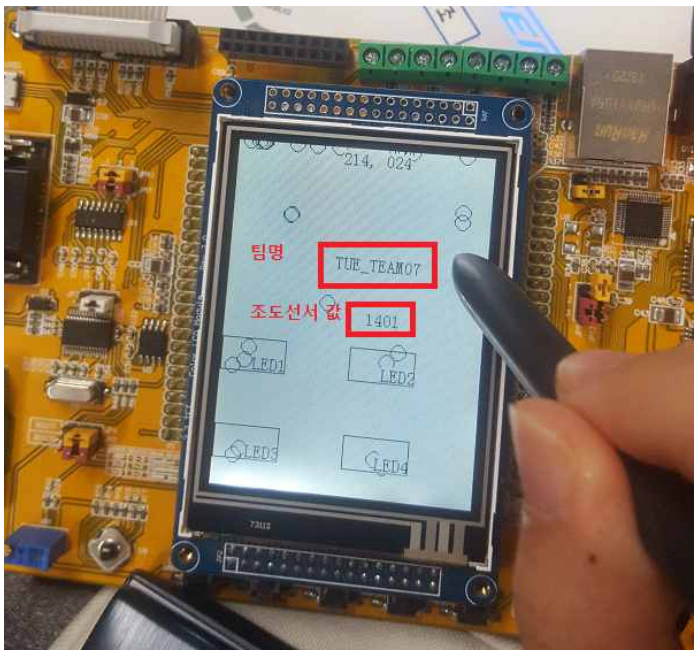
2-1) TFT-LCD에 TUE_Team07이 출력되는 것을 확인

2-2) LCD 터치 시 해당 위치에 작은 원을 그리고 좌표를 출력하는 것을 확인

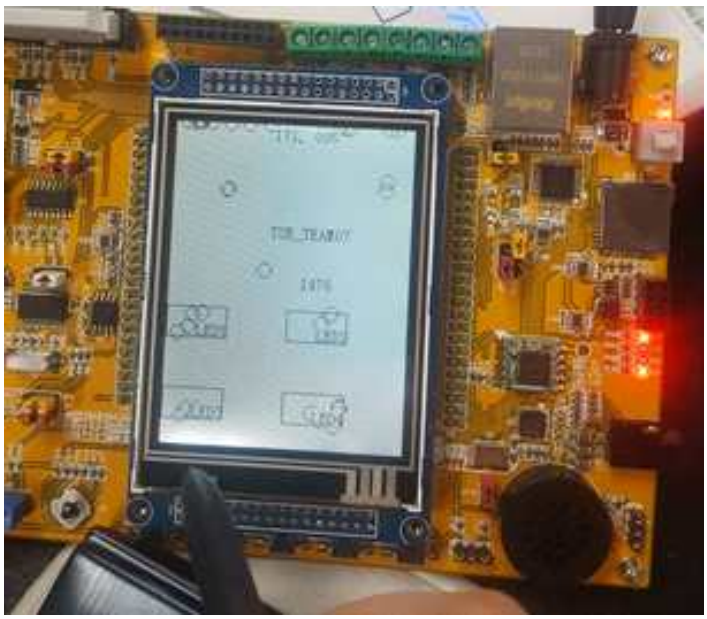
2-3) LCD에 조도센서에서 받아온 값이 출력되는 것을 확인

2-4) LED1, LED2, LED3, LED4에 대한 버튼을 클릭 시 해당 LED가 켜지는 것을 확인

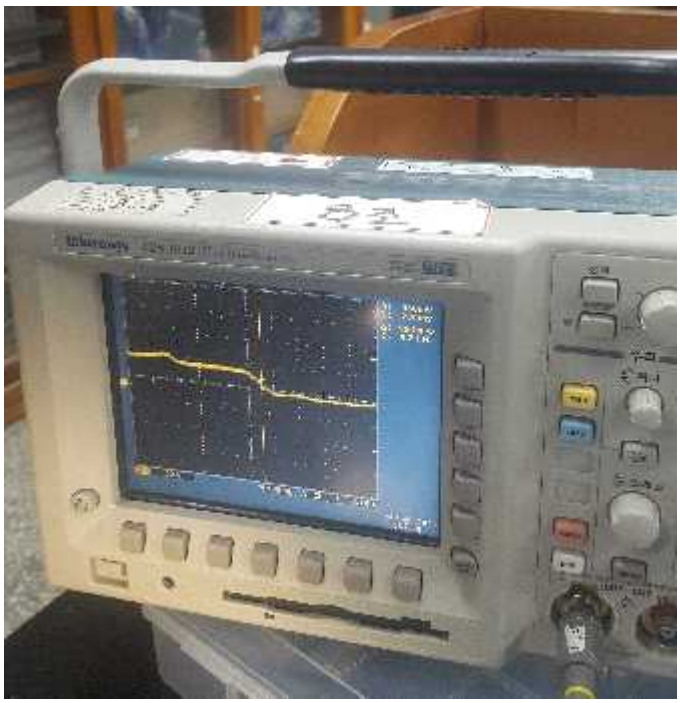
2-5) 오실로스코프를 통해 빛의 양에 따른 저항값 변화를 확인



<그림 4 : TFT-LCD에 팀명과 조도센서 값 출력>



<그림 5 : LED1~4 버튼 터치 시 해당 LED on>



<그림 6 : 빛의 양에 따른 저항값 변화>

5. 결론 및 느낀점

이번 실험에서는 TFT-LCD를 통해 코드로 작성한 텍스트, 도형, 센서를 통해 측정한 데이터 등을 LCD에 시각적으로 표현해 보았다. 조도센서를 통해 측정된 아날로그 신호들은 ADC 회로를 이용하여 디지털 신호로 LCD에 출력되었다. 또한 오실로스코프를 사용하여 조도센서의 측정값에 따라 오실로스코프에 출력되는 값이 달라지는 것을 확인하였다. 출력뿐 아니라 LCD를 터치하는 입력으로 LED를 제어할 수도 있었다.

ADC회로를 사용하면 아날로그 신호를 측정하여 디지털 값으로 변환할 수 있기 때문에 하드웨어를 유동적으로 제어할 수 있다는 것을 이해할 수 있었고, 오실로스코프를 사용하여 센서를 통한 측정값에 따라 파형이 달라짐을 확인하였다. 그리고 ADC핀이 공통으로 사용되기도 해서 회로도에 표시된 핀만큼 많은 센서들을 사용할 수는 없다는 점을 확인하였는데 이런 부분은 텀프로젝트등을 진행할 때에도 주의하여 사용해야 할 것 같다.