임베디드 시스템 설계 및 실험
화요일 분반 9주차 실험 결과 보고서

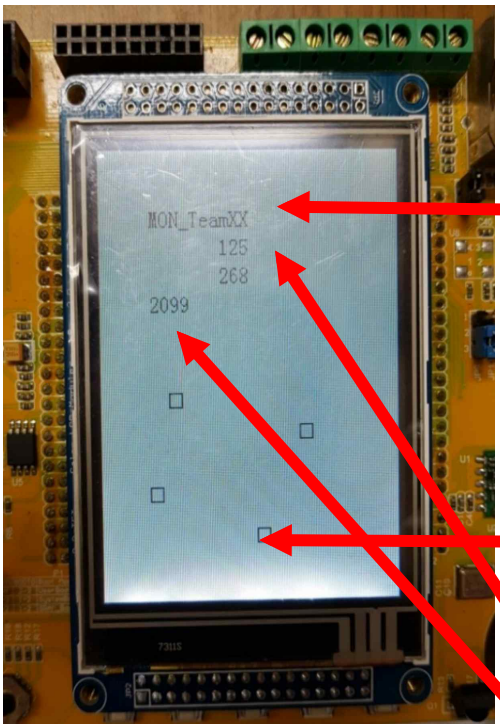
조 : 6조

조원 : 이주승 김선규 이동현 이지현 최세희

실험목표

1. TFT LCD 제어 및 ADC 사용

세부 목표



1. TFT-LCD를 보드에 올바르게 결착한다.

2. lcd.c에서 write 관련 코드 작성한다.

3. TFT-LCD에 Text(팀명)을 출력한다.

4. ADC channel과 인터럽트를 사용하여 조도 센서 값을 전역변수에 저장한다.

5. LCD 터치 시(main에서 폴링 방식) 해당 위치에 작은 원을 그리고,

좌표(X, Y)와

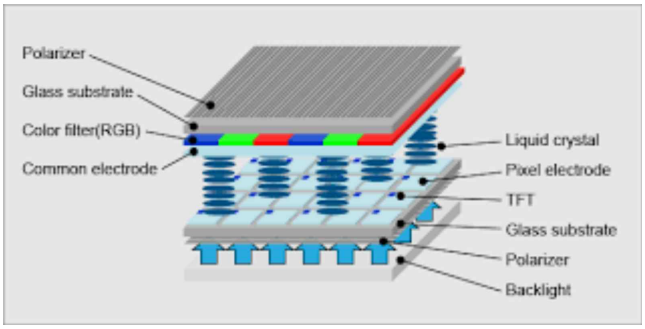
전역변수에 저장했던 조도 센서 값을 출력한다.

이론적 배경

1. TFT-LCD

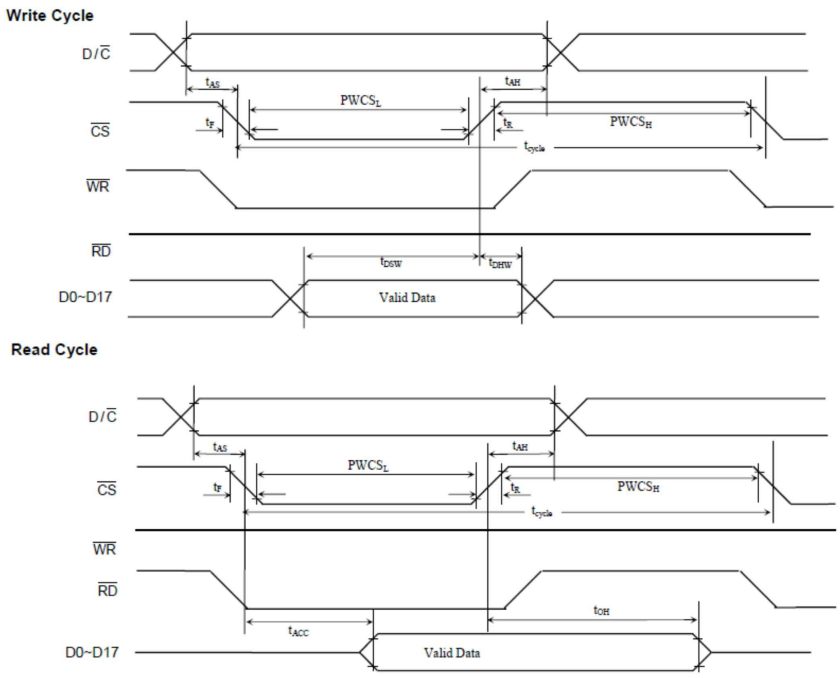
TFT-LCD(Thin Film Transistor liquid Crystal Display)는 초 박막 액정 표시 장치 이다. 액체와 고체의 중간 특성을 가진 액정의 상태 변화와 편광판의 편광 성질을 이용하여 통과하는 빛의 양을 조

절함으로써 정보를 표시한다. 색을 구성하는 Color Filter, 액정에 신호 전압을 인가하는 TFT 기판, 광원인 Back light로 구성된다.



2. Timing Diagram

각 신호들이 시간 별로 처리되는 과정을 그림으로 나타낸 것으로 다른 객체의 상태 변화 사이의 시간적 제약을 보여주는 데 유용하다. 실제 파형의 움직임은 전기 신호가 물리적으로 변동되는 데 시간이 걸리기 때문에 직각으로 떨어지지 않는다. Low에서 High로 올라가는 구간을 Rising Edge, High에서 Low로 떨어지는 구간을 Falling Edge라고 한다. 교차 형태의 경우에는 High/Low 둘 중 하나의 값을 가질 수 있다는 것을 의미한다.

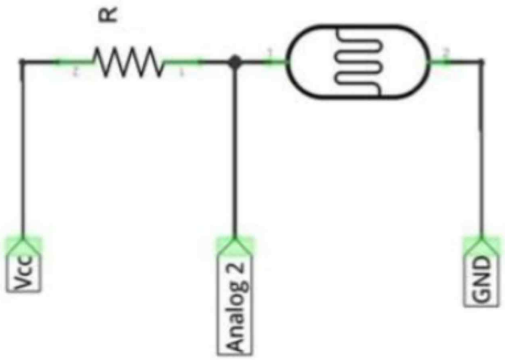


3. ADC와 DAC

ADC(Analog to Digital Converter)는 아날로그 신호를 디지털 신호로 변환하는 것이다.
DAC(Digital to Analog Converter)는 디지털 신호를 아날로그 신호로 변환하는 것이다. 아날로그 신호에서 디지털 신호 변환 과정은 표본화, 양자화, 부호화 단계를 거친다. 표본화는 디지털 신호로 바꾸기 위해 x축을 기준으로 일정한 간격으로 나누는 것이다. 양자화는 표본화된 값을 디지털화 하는 작업이다. 부호화 과정에서는 양자화된 값을 이진수로 변환한다.

4. 조도센서

조도 센서는 주변의 밝기를 측정하는 센서이다. 빛의 양이 많을수록 전도율이 높아져 내부 저항이 낮아진다. 풀업 저항으로 연결 시 밝을 때는 측정 값이 작아지고 어두울 때는 커진다. 풀다운 저항으로 연결 시 밝을 때는 측정 값이 커지고 어두울 때는 값이 작아진다.



구현 내용

1. Data / Command Write

TFT-LCD Pin맵과 Timing Diagram을 참고해서 lcd.c 에서 Data / Command 부분을 완성한다. TFT-LCD Pin맵에서 RS는 PD13, CS는 PC6, RD는 PD15, WR는 PD14이다. 그림1은 데이터시트의 Timing Diagram에서 Write Cycle이다.

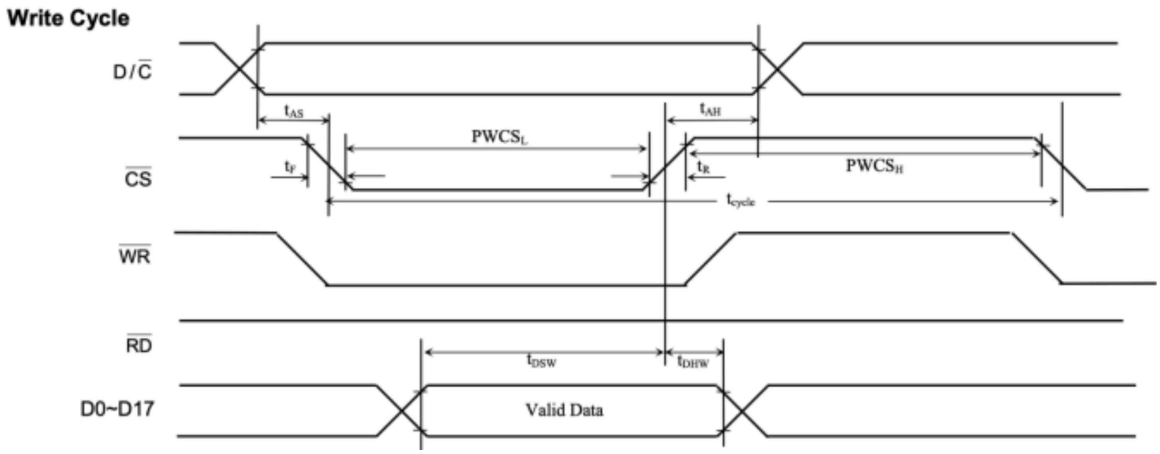


그림 . Write Cycle

Command는 DC와 CS, WR를 모두 Low로 두고 커맨드를 전송하므로 LCD_WR_REG 함수에서 GPIO_ResetBits로 각각의 핀을 Low로 만든다. 그리고 다시 GPIO_SetBits로 CS와 WR를 High로 만든다. 구현은 그림2와 같다.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOD, GPIO_Pin_13);
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14);

        GPIO_Write(GPIOE, LCD_Reg);

        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOC, GPIO_Pin_6);
        GPIO_SetBits(GPIOD, GPIO_Pin_14);

}
```

그림 . LCD_WR_REG 코드

Data는 DC를 High, CS와 WR는 Low로 두고 Data를 display에 전송하므로 LCD_WR_DATA 함수에서 GPIO_SetBits와 GPIO_ResetBits로 각각의 핀을 High, Low로 설정한다. 그리고 다시 GPIO_SetBits로 CS와 WR를 High로 만든다. 구현은 그림3 과 같다.

```
static void LCD_WR_DATA(uint16_t LCD_Data)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOD, GPIO_Pin_13);
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14);

        GPIO_Write(GPIOE, LCD_Data);
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOC, GPIO_Pin_6);
        GPIO_SetBits(GPIOD, GPIO_Pin_14);

}
```

그림 . LCD_WR_DATA 코드

2. ADC Configuration

ADC 설정은 다음과 같이 구조체를 선언한다. ADC를 1개 사용하므로 독립적인 타이밍으로 동작해도 되는 indepedent mode를 사용하고, Scan Conversion Mode를 disable 한다. 연속적으로 Conversion을 수행하기 위해 Continuous Conversion Mode는 enable 한다. 내부 타이머를 설정하지 않으므로 External trigger conversion은 none으로 하고, 데이터는 right alignment로 오른쪽에서부터 비트를 할당한다. ADC_RegularChannelConfig() 함수로 initialize한 ADC1의 channel 우선순위를 결정한다. sampling cycle은 가장 높게 설정한다. ADC_ITConfig로 ADC의 interrupt를 enable한다. ADC_Cmd()로 ADC1을 enable한다.

리셋 후 1 회 ADC Self Calibration 을 해줘야 제대로 된 ADC 값을 읽을 수 있다. 사용하는 ADC1의 reset calibration 레지스터를 enable하고, while문에서 ADC_GetResetCalibrationStatus로 reset 상태를 체크한다. ADC1의 calibration을 시작하고, 그 결과를 while문에서 체크한다. 이후 SoftwareStartConvCmd()로 소프트웨어 conversion을 시작한다.

```c
void ADC_Configure()
{
    ADC_InitTypeDef ADC_InitStructure;

    // ADC Configure
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent; // Independent Mode
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE; // Continuous conversion
    ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None // 내부 트
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);

    ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 1, ADC_SampleTime_239Cycles5)
    // Enable interrupt
    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);
    // Enable ADC1
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibration(ADC1);

    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1); // ADC Calibration start
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

그림 . ADC_Configure() 코드

3. ADC Interrupt Handler

ADC Interrupt에 대한 Request Handler를 구현한다. interrupt를 enable하여 ADC1이 ADC_IT_EOC 상태가 되면 ADC1의 conversion 값을 value 변수에 받아온다. interrupt 동작을 마무리할 때 pending bit를 clear시켜주어 인터럽트가 다시 발생하지 않도록 한다. (GPIO, NVIC, RCC 등의 설정은 이전 주차와 동일한 방식으로 구현한다.)

```c
void ADC1_2_IRQHandler()
{
    if (ADC_GetITStatus(ADC1, ADC_IT_EOC) != RESET) {
        value = ADC_GetConversionValue(ADC1);
        ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
    }
}
```

그림 5. ADC1_2_IRQHandler 코드

4. Main

메인 함수는 TPT-LCD에 입력된 정보를 기록하고 출력을 담당한다. 폴링 방식을 통해서 LCD터치시

에 터치 된 자리에 작은 원을 그리고, 터치 된 곳의 X,Y값을 기록하며, 조도센서의 값을 기록한다. 터치 입력된 X,Y은 출력해야 하는 X,Y값과 수치가 다르므로 이를 LCD 240X320 기준으로 변환시켜 출력시킨다.

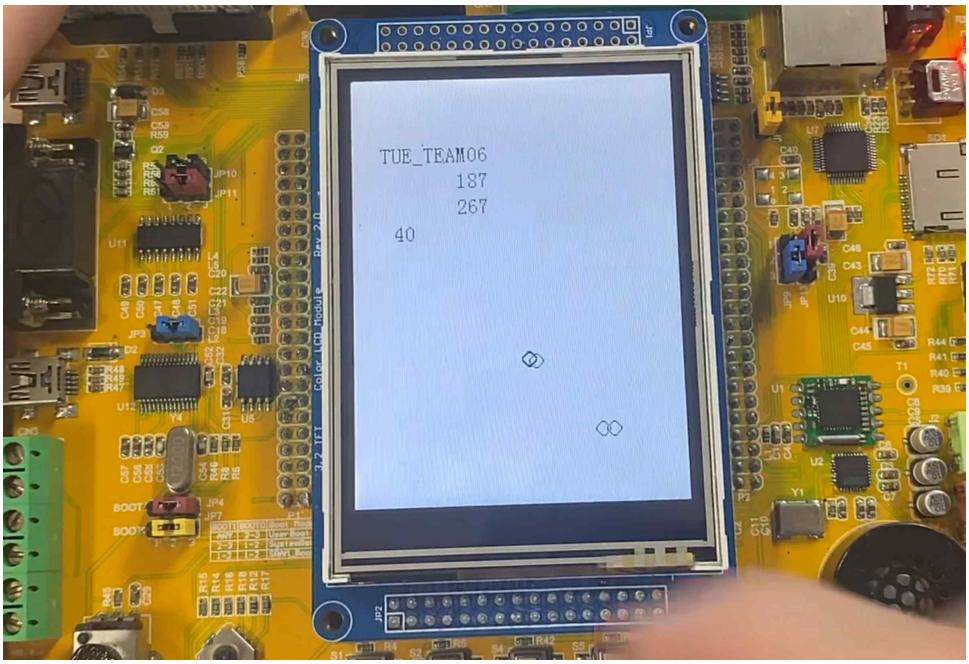
```
int main()
{
    SystemInit();
    RCC_Configure();
    GPIO_Configure();
    ADC_Configure();
    NVIC_Configure();
    LCD_Init();
    Touch_Configuration();
    Touch_Adjust();
    LCD_Clear(WHITE);
    uint16_t r = 5;
    while(1) {
        LCD_ShowString(LCD_NAME_POS_X, LCD_NAME_POS_Y, "TUE_TEAM06", BLACK, WHITE)

        if (!T_INT) {
            Touch_GetXY(&pos_x, &pos_y, 0);
            Convert_Pos(pos_x, pos_y, &cvt_pos_x, &cvt_pos_y);
            LCD_DrawCircle(cvt_pos_x, cvt_pos_y, r);
            LCD_ShowNum(LCD_LUXVAL_X, LCD_LUXVAL_Y, value, 5, BLACK, WHITE);
            LCD_ShowNum(LCD_X_POS_X, LCD_X_POS_Y, pix_x, 4, BLACK, WHITE);
            LCD_ShowNum(LCD_Y_POS_X, LCD_Y_POS_Y, pix_y, 4, BLACK, WHITE);
            delay();
        }
    }
}
```

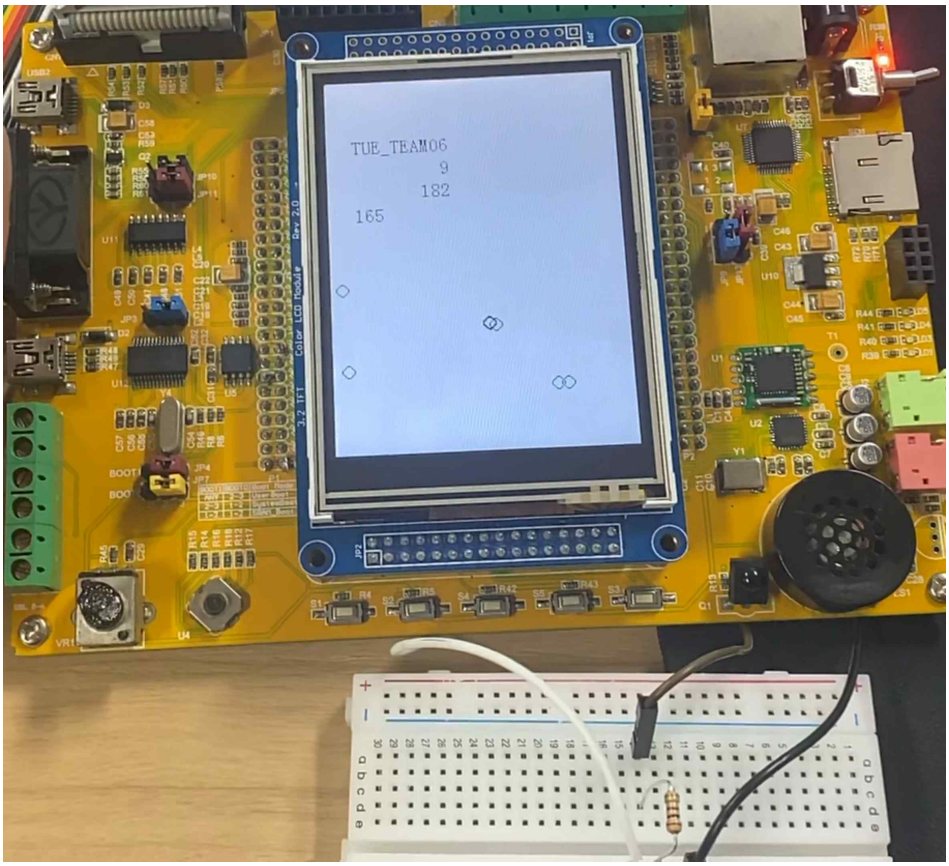
그림 6. main() 코드

실험 결과

동작영상과 같이 lcd를 터치한 부분에 원을 그릴 수 있고, 원 좌표값과 조도 센서값이 아래 사진처럼 표시된다.



손으로 조도센서를 가렸을 때 40이 나오고,



손을 뗐을 때 165로 값이 올라가는 것을 확인할 수 있다.

손으로 터치할 때 그려지는 원의 좌표 값도 팀명 아래에 나타난다.

어려웠던 점

1. 텍스트를 출력하는 과정에서 미션지에 있던 것과 같이 텍스트를 깔끔하게 정렬하는 과정에서 어려움을 겪었다. 이는 결국 좌표 값을 계속 수정하는 과정을 걸쳐서 깔끔하게 정렬을 맞추었다.

2. 실험을 진행할 때 순서를 올바르지 하지 않았기 때문에 어려움을 겪었다. 이번 실험에서는 LCD의 작동 여부를 먼저 확인을 한 후 조도센서의 작업을 들어가야 했지만. 이를 생각 안 하고 조도센서부터 작업을 하려는 과정에서 작성한 조도센서 소스를 확인하는 방법이 없었다. 따라서 LCD부터 작업을 하게 되었고 이후 조도센서의 잘못된 부분을 수정함으로써 시간을 허비하게 되었다.