REPORT

10주차 결과 보고서

수강과목: 임베디드 시스템 설계 및 설계

담당교수: 백윤주 교수님

학 과: 전기컴퓨터공학부 정보컴퓨터공학전공

조 원: 201724651 김장환 201724648 김경호

201624481 박윤형 201524588 조창흠

제출일자: 2021. 11. 09

실험 목표

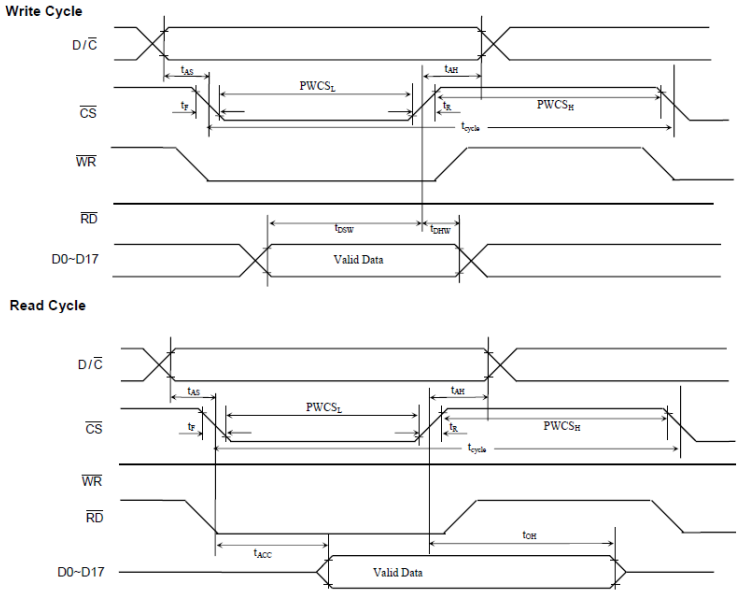
1. TFT-LCD의 원리와 동작 방법에 대한 이해
2. TFT-LCD 라이브러리 작성과 이해
3. TFT-LCD Touch 동작 제어
4. ADC 개념 이해
5. 조도 센서 사용 방법 학습

배경 지식

(1) TFT-LCD

- TFT 기술을 이용하여 화질을 향상시킨 LCD
- 다른 유형의 LCD 에 비해 가볍고 얇으며 에너지 효율이 좋다.
- RGB 픽셀이 유리판에 코딩 되어 컬러 영상을 구현하는 Color Filter
- 각 픽셀마다 트랜지스터 스위치를 배치해 각 픽셀을 독립적으로 제어

(2) Timing Diagram



(3) ADC

- Analog to digital converter 의 약자로 연속적인 Analog 신호를 digital 신호로 변환하는 장치
- 신호의 변환을 위해 Sampling(표 본화), Quantization(양자화), Coding(부호화)를 거침

- 온도, 습도, 조도 등의 analog 물리량들이 ADC 의 변환 과정을 거쳐 컴퓨터 시스템에 전달

(4) DAC

- Digital to Analog converter 의 약자로 부호화된 digital 신호를 analog 신호로 변환하는 장치
- Digital 신호를 현실에서 사용하기 위해 Analog 신호를 변환할 필요가 있음
- 비트수가 많을수록 분해능이 높아져 신호의 정밀성이 증가

(5) 조도센서

- 주변 환경의 밝기를 측정할 수 있는 센서
- 저항의 값은 빛의 양에 따라 가변적으로 달라지며, 전압의 차이로 조도를 구분할 수 있다.

실험 내용

1. TFT-LCD

- 보드에 TFT-LCD 연결

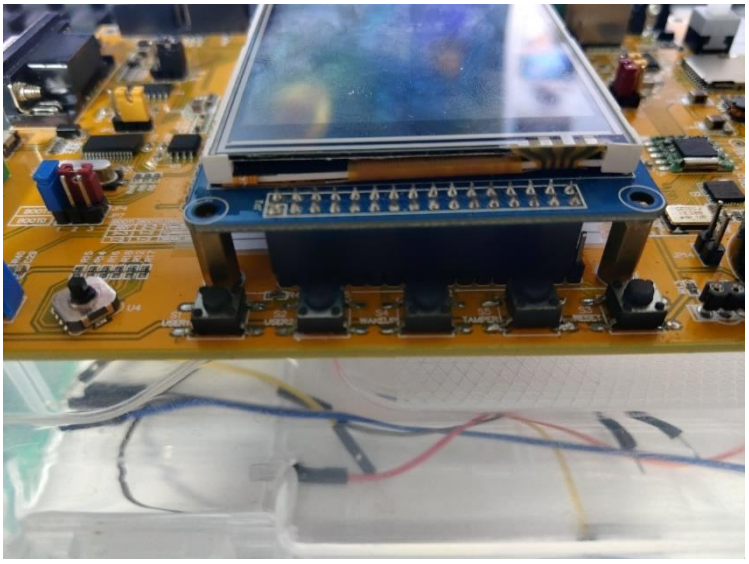


그림1. TFT-LCD 연결

사진1처럼 보드의 JP5에 TFT-LCD를 연결한다.

- **lcd.c 작성**

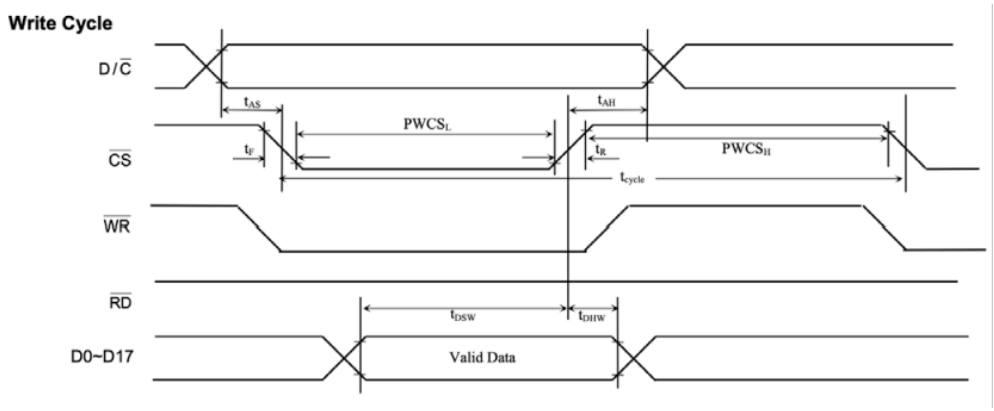


그림2. Timing Diagram – Write Cycle

그림2에 있는 write cycle시의 timing diagram을 참고해서 lcd.c의 LCD_WR_REG()와 LCD_WR_DATA()를 작성한다.

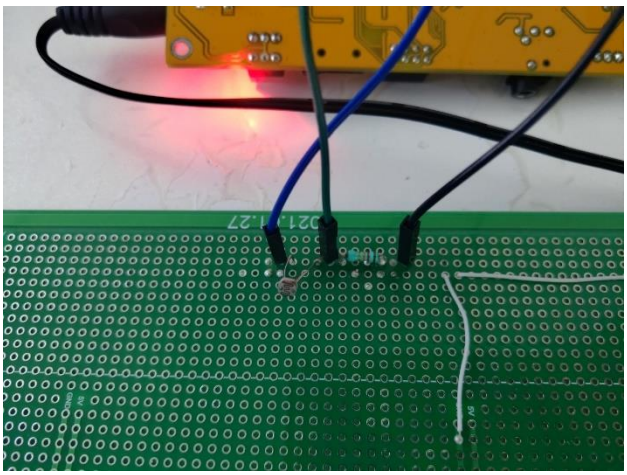
```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
        GPIO_ResetBits(GPIOD, GPIO_Pin_13); // LCD_RS(0);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);

        GPIO_Write(GPIOE, LCD_Reg);
        // TODO implement using GPIO_ResetBits/GPIO_SetBits

        GPIO_SetBits(GPIOD, GPIO_Pin_14); // LCD_WR(1);
        GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
}

static void LCD_WR_DATA(uint16_t LCD_Data)
{
        // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
        GPIO_SetBits(GPIOD, GPIO_Pin_13); // LCD_RS(1);
        GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);

        GPIO_Write(GPIOE, LCD_Data);
        // TODO implement using GPIO_ResetBits/GPIO_SetBits

        GPIO_SetBits(GPIOD, GPIO_Pin_14); // LCD_WR(1);
        GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
}
```

그림3. Lcd.c 코드 작성

LCD_WR_REG()에서 Command를 전송하기 전에 GPIO_ResetBits()로 RS에 Low, CS에 Low, WR에 Low의 값을 준다. GPIO_Write()로 Command를 전송한 후에 GPIO_SetBits()로 CS와 WR에 High의 값을 준다.

LCD_WR_DATA()에서 Data를 전송하기 전에 GPIO_ResetBits()로 CS에 Low, WR에 Low의 값을 주고, GPIO_SetBits()로 RS에 High의 값을 준다. GPIO_Write()로 Data를 전송한 후에 GPIO_SetBits()로 CS와 WR에 High의 값을 준다.

2. 조도센서 회로구성



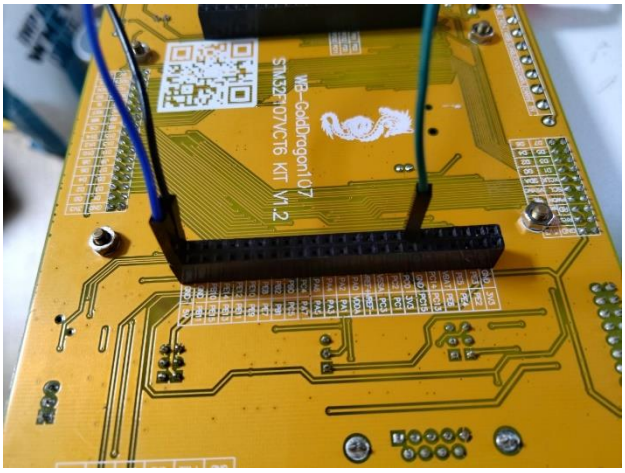


그림4. 조도센서 회로 구성

조도센서와 저항에 5V의 전압을 인가하고 조도센서의 값을 측정하기 위해 조도센서와 저항 사이의 부분을 보드와 연결한다. 이때, 조도센서의 값을 ADC를 이용해서 읽어 내기 위해 ADC가 연결된 핀 중 하나인 PC0에 연결한다.

3. main.c

- GPIO_Confiture()

```
void RCC_Configure(void)
{
    //Enable the APB2 peripheral clock using the function 'RCC_APB2PeriphClockCmd'
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

}
```

ADC를 사용하기위해 ADC1을 활성화시키고 PC0을 사용하기 위해 GPIOC를 활성화시킨다. ADC1와 GPIOC가 APB2로 연결되기 때문에 APB2를 활성화시킨다.

- GPIO_InitTypefigure()

```
void GPIO_Configure(void)
{
    GPIO_InitTypeDef GPIO_ADC;

    //Initialize the GPIO pins using the structure 'GPIO_InitTypeDef' and the function 'GPIO_Init'
    GPIO_ADC.GPIO_Pin = GPIO_Pin_0;
    GPIO_ADC.GPIO_Mode = GPIO_Mode_AIN;
    GPIO_ADC.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_ADC);
}
```

PC0을 입력으로 사용하기 위해 GPIOC의 0번 pin을 analog input모드로 설정한다.

- ADC_Configure()

```
void ADC_Configure(void) {
    ADC_InitTypeDef ADC;

    ADC.ADC_Mode = ADC_Mode_Independent ;
    ADC.ADC_ContinuousConvMode = ENABLE;
    ADC.ADC_DataAlign = ADC_DataAlign_Right;
    ADC.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADC.ADC_NbrOfChannel = 1;
    ADC.ADC_ScanConvMode = DISABLE;

    ADC_Init(ADC1, &ADC);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_239Cycles5);
    ADC_ITConfig(ADC1,  ADC_IT_EOC, ENABLE );
    ADC_Cmd(ADC1, ENABLE);

    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1)) ;

    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1)) ;

    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
```

ADC의 Init structure를 선언하고 설정 값을 준다.

1. ADC_Mode는 Independent로 설정한다.
2. 값을 계속해서 일기 위해 ContinuousConvMode를 활성화.
3. DataAlign을 Right로 설정.
4. 외부 트리거를 사용하지 않기 때문에 ExternalTrigConv는 None으로 설정.
5. 채널의 수는 1로 설정.
6. ScanConvMode는 비활성화.

ADC_Init 함수에 ADC의 Init_Structure 전달해준다.

ADC_RegularChannelConfig로 사용할 ADC채널의 설정해준다. PC0이 ADC1의 10번채널로 연결되므로 ADC_Channel1_10을 랭크1, 239.5 cycles로 설정한다.

ADC가 값을 변환하고 Inturrupt가 발생하게 하기위해 ADC_ITConfig()함수로 ADC1의 ADC_IT을 EOC로 설정한다.

- NVIC_Configure(), ADC1_2_IRQHandler()

```
void NVIC_Configure(void) {
    NVIC_InitTypeDef NVIC_ADC;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    //Initialize the NVIC using the structure 'NVIC_InitTypeDef' and the function 'NVIC_Init'

    NVIC_EnableIRQ(ADC1_2_IRQn);
    NVIC_ADC.NVIC_IRQChannel = ADC1_2_IRQn;
    NVIC_ADC.NVIC_IRQChannelPreemptionPriority = 0x0;
    NVIC_ADC.NVIC_IRQChannelSubPriority = 0x0;
    NVIC_ADC.NVIC_IRQChannelCmd = ENABLE;

    NVIC_Init(&NVIC_ADC);
}
void ADC1_2_IRQHandler(void) {
    if(ADC_GetITStatus(ADC1, ADC_IT_EOC)!=RESET){
        value = ADC_GetConversionValue(ADC1);

        ADC_ClearITPendingBit(ADC1,ADC_IT_EOC);
    }
}
```

ADC의 값을 인터럽트로 받기위해 NVIC_Configure()와, ADC1_2_IRQHandler()을 정의한다.

- **main()**

```
int main(void) {

    SystemInit();
    RCC_Configure();
    GPIO_Configure();
    ADC_Configure();
    NVIC_Configure();

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

    LCD_ShowString(LCD_TEAM_NAME_X, LCD_TEAM_NAME_Y, "Tue_Team 08", BLACK, WHITE);
    LCD_ShowString(LCD_COORD_X_X - 17, LCD_COORD_X_Y, "X: ", BLACK, WHITE);
    LCD_ShowString(LCD_COORD_Y_X - 17, LCD_COORD_Y_Y, "Y: ", BLACK, WHITE);

    while (1) {
        Touch_GetXY(&cur_x, &cur_y, 1);
        Convert_Pos(cur_x, cur_y, &pixel_x, &pixel_y);

        LCD_DrawCircle(pixel_x, pixel_y, 3);
        LCD_ShowNum(LCD_COORD_X_X, LCD_COORD_X_Y, pixel_x, 4, BLACK, WHITE);
        LCD_ShowNum(LCD_COORD_Y_X, LCD_COORD_Y_Y, pixel_y, 4, BLACK, WHITE);
        LCD_ShowNum(LCD_LUX_VAL_X, LCD_LUX_VAL_Y, value, 4, BLUE, WHITE);


    }
    return 0;
}
```

LCD_TEAM_NAME_X, LCD_TEAM_NAME_Y : 이름을 표시할 위치

LCD_COORD_X_X, LCD_COORD_X_Y : 터치한 위치의 X좌표를 표시할 위치

LCD_COORD_Y_X, LCD_COORD_Y_Y : 터치한 위치의 Y좌표를 표시할 위치

LCD_LUX_VAL_X, LCD_LUX_VAL_Y : 조도센서로부터 받은 값을 표시할 위치

먼저 LCD_ShowString()으로 "Tue_Team 08"을 출력한다. while문에서 Touch_GetXY()로 터치한 위치의 값을 Convert_Pos로 변환해서 pixel_x, pixel_y에 저장한다. LCD_DrawCircle()로 터치한 위치에 작은 원을 그린다. LCD_ShowNum으로 터치한 위치의 X값, Y값 그리고 ADC에서 인터럽트가 일어나 value에 저장된 값을 LCD에 출력한다.

실험 결과

그림5. 전원을 켰을 때 팀명을 화면에 출력

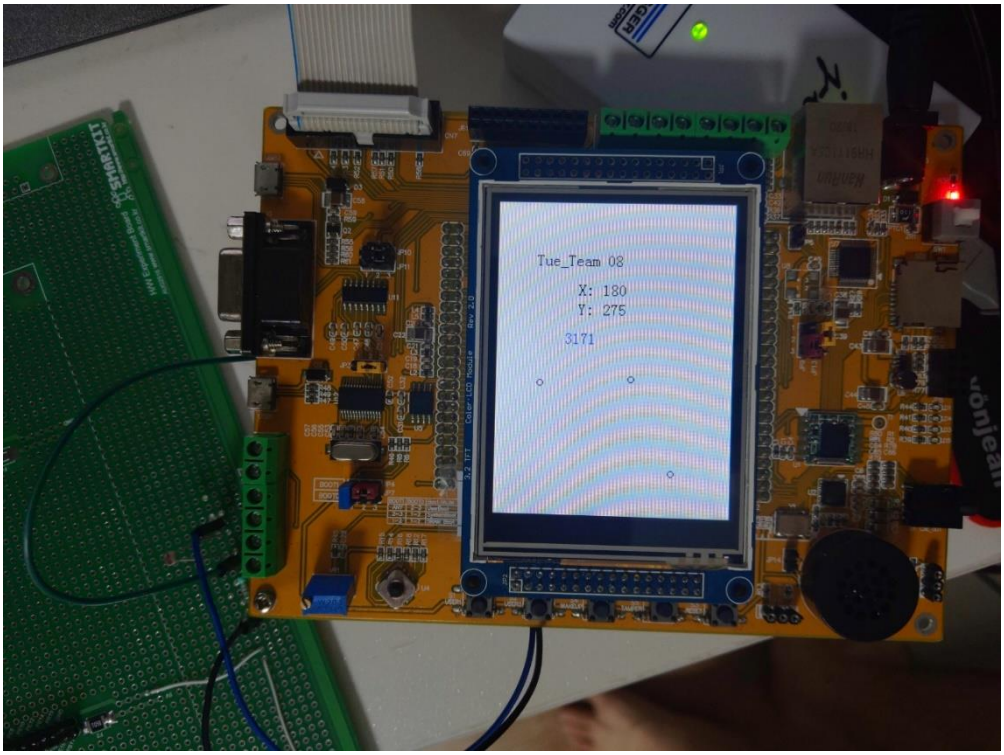


그림6. 터치를 하면 원을 그리고 좌표와 조도센서의 값을 출력

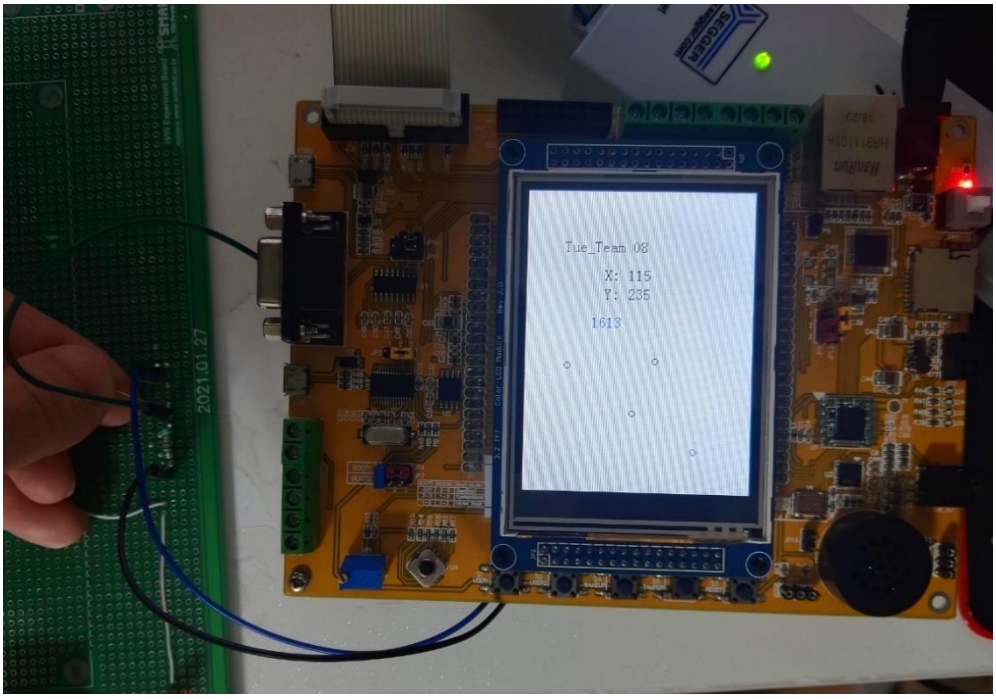


그림7. 주변 밝기에 따라 조도센서의 값의 변화

전원을 키면 화면에 팀명 "Tue_Team 08"을 출력한다. 화면을 터치하면 그 자리에 원을 생성하고 터치한 자리의 위치와 조도센서의 값을 출력한다. 주변환경의 밝기가 줄어들면 조도센서에서 읽어들이는 값이 줄어든다.

결론 및 느낀점

TFT LCD를 이용하여 데이터를 표시하거나 입력을 받는 실험을 진행 하였다. lcd.c에서 LCD_WR_REG와 LCD_WR_DATA의 기능에 대한 이해가 부족하여 시간이 조금 걸리긴 하였지만 예비 발표 및 강의 자료를 참고하여 해결하였다. 코드 자체는 이전에 실습한 실험에 이어지는 과정이었기에 수월하게 진행할 수 있었다.