임베디드시스템 설계 및 실험 보고서

[002 분반 - 2 조 - 10 주차]

조원	202055531 김후겸 202055584 이태경 202155540 김채현 202255535 김진우

실험날짜	2024-11-6

1. 실험 주제

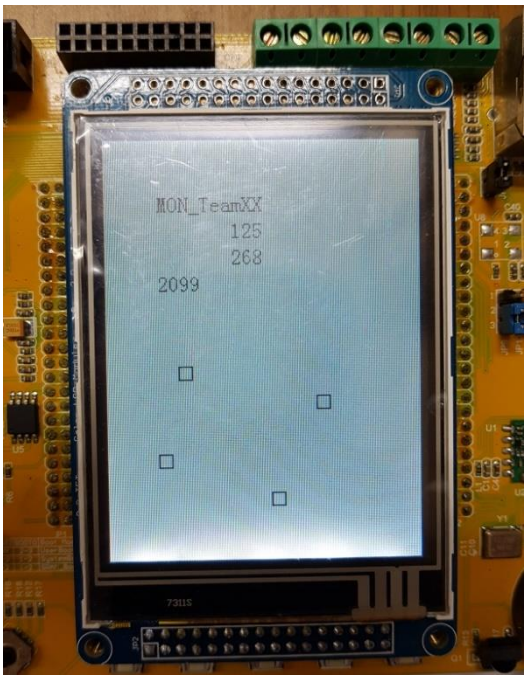
- LCD 및 ADC

2. 실험 목적

- TFT-LCD 의 원리와 동작 방법에대한이해
- TFT-LCD 라이브러리 작성과 이해
- TFT-LCD Touch 동작 제어
- ADC 개념이해
- 조도 센서 사용방법 학습

3. 세부 실험 목적

1. TFT-LCD 보드에 올바르게 결착
2. lcd.c 에서 write 관련 코드 작성
3. TFT-LCD 에 Text(팀명) 출력
4. ADC channel 과 인터럽트를 사용하여 조도 센서값을 전역변수에 저장
5. LCD 터치 시(main 에서 폴링 방식) 해당 위치에 작은 원을 생성하며 좌표(X, Y), 전역변수에 저장했던 조도 센서 값 출력



4. 실험 장비

- STM32F107VCT6
- TFT-LCD
- 조도 센서

5. 실험 과정

1. LCD 라이브러리 등록 및 lcd.c 완성

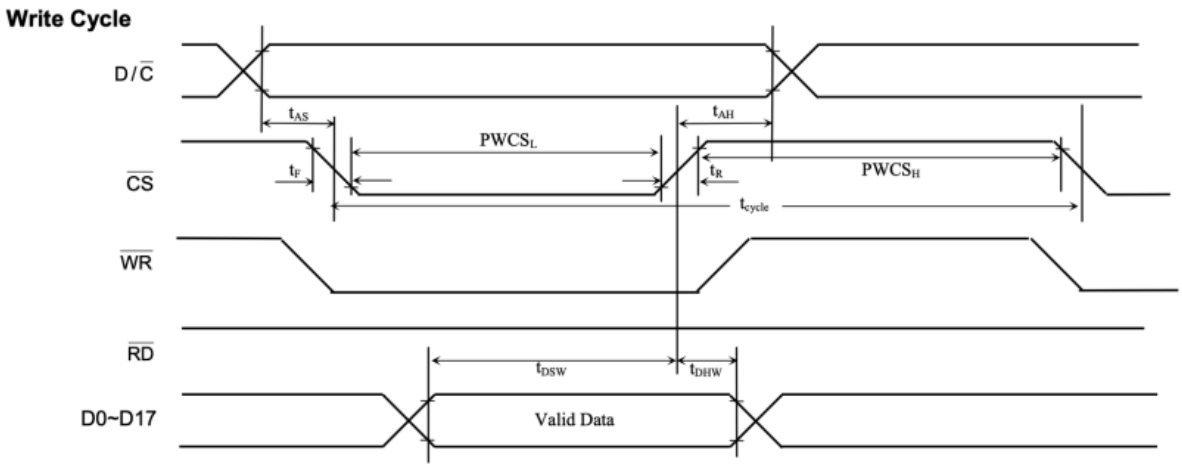
Libraries 폴더 밑에 LCD 폴더 생성 후 font.h, lcd.c, lcd.h, touch.c, touch.h 파일을 추가한다.

lcd.c 에서 아래의 두 함수를 완성해야 한다.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_Write(GPIOE, LCD_Reg);
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOC, GPIO_Pin_8); // CS LOW
    GPIO_ResetBits(GPIOB, GPIO_Pin_14); // WR LOW
    GPIO_ResetBits(GPIOD, GPIO_Pin_13); // D/C LOW
    4
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_SetBits(GPIOC, GPIO_Pin_8); // CS HIGH
    GPIO_SetBits(GPIOB, GPIO_Pin_14); // WR HIGH
}
```

```
static void LCD_WR_DATA(uint16_t LCD_Data)
{
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_ResetBits(GPIOC, GPIO_Pin_8); // CS LOW
        GPIO_ResetBits(GPIOB, GPIO_Pin_14); // WR LOW
                GPIO_SetBits(GPIOD, GPIO_Pin_13); // D/C HIGH
        GPIO_Write(GPIOE, LCD_Data);
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
        GPIO_SetBits(GPIOC, GPIO_Pin_8); // CS HIGH
        GPIO_SetBits(GPIOB, GPIO_Pin_14); // WR HIGH
}
```

TFT-LCD 가 데이터를 쓸 때 쓰기 신호, 데이터 신호 등 신호 간의 순서와 시간 차이를 맞춰야 작동한다. Timing Diagram 은 이러한 신호들이 시스템에서 시간에 따라 어떻게 변화하는지 시각적으로 나타낸 그래프이다. Write Cycle 은 마이크로 컨트롤러가 LCD 로 데이터를 전달할 때 필요한 신호와 타이밍을 정의한 주기이다.



D/C(RS) : LOW 일 때, Command 를 전송, HIGH 일 때, Data 를 전송한다.

CS(Chip Select) : LOW 일 때, Chip 이 동작한다.

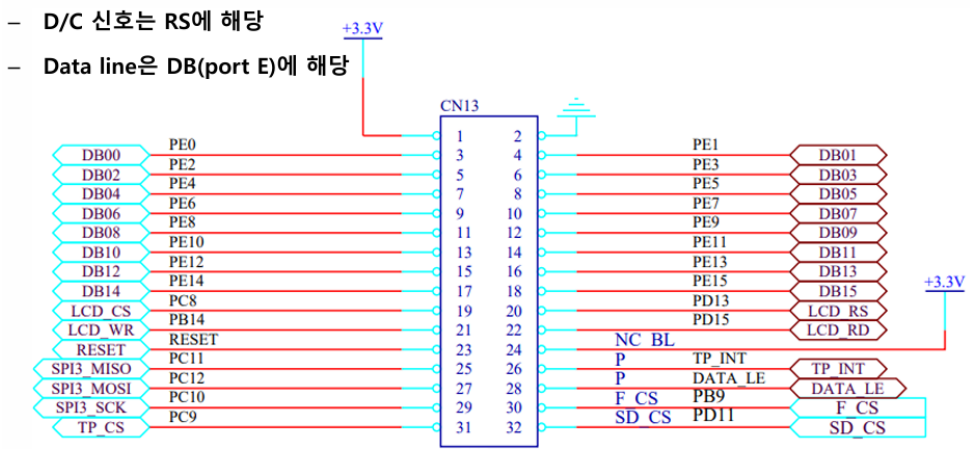
WR(Write) : low 상태가 되면 Write 작업이 시작한다.

LCD 에 text 를 디스플레이 하기

i. COMMAND
   - D/C = LOW, CS = LOW, WR = LOW

ii. DATA
   - D/C = HIGH, CS = LOW, WR = LOW

후 CS 와 WR 을 모두 HIGH 로 돌려놓는다.



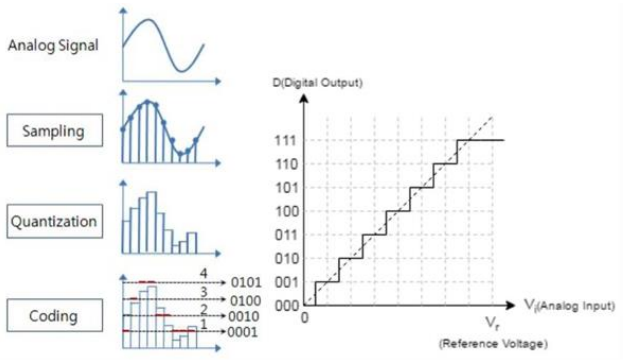
LCD_CS⇒PC8, LCD_WR⇒PB14, LCD_RS⇒PD13 으로 GPIOB, GPIOC, GPIOD 를 사용한다. 따라서 GPIOC_PIN8, GPIOB_PIN14, GPIOD_PIN13 을 각각 설정해준다.

RS 는 HIGH 일 때 데이터를 전송해주므로 LCD_WR_DATA()에서 reset 이 아니라 set 을 해준다.

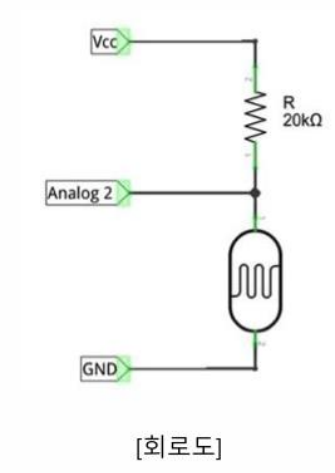
2. 조도 센서와 TFT-LCD 연결

조도 센서는 빛의 양이 많을 때는 저항이 낮아지고, 빛이 적을 때는 저항이 높아진다. 이 저항의 변화는 아날로그 신호로 나타나며, ADC 를 통해 디지털 신호로 변환되어 MCU 에서 처리된다.

ADC(Analog to Digital Converter)는 아날로그 신호를 디지털 값으로 변환한다. 변환 과정은 표본화(Sampling), 양자화(Quantization), 부조화(Coding)로 이루어진다.



조도 센서의 한 쪽은 GND 에 다른 한 쪽은 저항에 연결한다. 저항은 VCC 에 연결되어 있다.



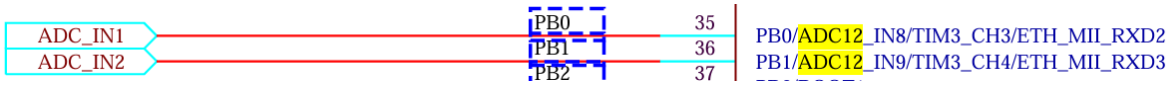
[회로도]

3. Main.c 완성

```
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, SENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
}
```

ADC1 과 ADC2 중 ADC1 을 사용한다. 위에서 봤던 LCD_CS, LCD_WR, LCD_RS 가 Port B,C,D 를 사용하므로 각각의 clock 을 활성화한다. 외부 인터럽트와 ADC 입력 설정을 위해 AFIO clock 도 활성화한다.

```
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(GPIOB, &GPIO_ADC);
}
```

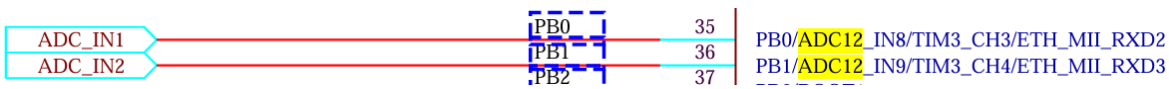


ADC1 이 PB0 이므로 GPIOB 의 Pin0 에 대해 설정을 해준다.

```
53    void NVIC_Configure(void) {
54        NVIC_InitTypeDef NVIC_ADC;
55
56        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
57        //Initialize the NVIC using the structure 'NVIC_InitTypeDef' and the function 'NVIC_Init'
58
59        NVIC_EnableIRQ(ADC1_2_IRQn);
60        NVIC_ADC.NVIC_IRQChannel = ADC1_2_IRQn;
61        NVIC_ADC.NVIC_IRQChannelPreemptionPriority = 0x0;
62        NVIC_ADC.NVIC_IRQChannelSubPriority = 0x0;
63        NVIC_ADC.NVIC_IRQChannelCmd = ENABLE;
64
65        NVIC_Init(&NVIC_ADC);
66    }
```

ADC 변환이 완료되면 인터럽트가 발생하도록 NVIC(중단 벡터 컨트롤러)를 설정하는 코드이다. ADC1_2_IRQn 은 ADC1 과 ADC2 의 인터럽트를 처리하는 중단 벡터이다.

```
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_8, 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);
}
```



ADC 설정을 위한 함수이다. 독립적으로 한 채널을 사용하기 위해 Independent mode 와 NbrOfCannel 을 1 로 설정한다. 또한 ContinuousConvMode 를 통해 연속 변환 모드를 활성화하여 지속적으로 변환할 수 있게 한다. channel 8 을 사용하여 analog

신호를 digital 신호로 변환한다.

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

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

ADC 변환이 완료되었을 때 호출되는 인터럽트 서비스 루틴이다. Value 변수에 변환된 ADC 값을 저장한다.

```
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, "WED_Team02", 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 스크린을 터치하면 해당 좌표의 x,y 값을 얻고, 원을 그린다. 조도 센서에서 얻은 값을 4 자리 숫자로 LCD 에 표시한다.

6. 실험 결과

7. 분석 및 결론

이번 실험에서는 아날로그 값을 디지털 값으로 변환하는 ADC 와, LCD 를 출력하는 방법에 대해 알아보았다. ADC 의 independent mode 는 master 인 ADC1 만 사용하므로 다른 ADC Block 과 sync 를 맞출 필요가 없어 independent 로 설정하는 것이었다. 텀 프로젝트에서 사용할 LCD 의 사용방법에 대해 자세히 습득할 수 있는 실험이었다.