임베디드 시스템 설계 및 실험 8주차 보고서

002분반 3조

201724601 최성렬

201729163 이희근

202055522 김은지

202055574 이다은

202055590 장서윤

실험 목표

1. TFT-LCD 원리와 동작방법을 이해하여 라이브러리를 작성한다.
2. polling 방식으로 입력을 기다리고, TFT-LCD 화면을 터치할 시 해당 좌표에 원을 그린다.
3. 조도센서의 값을 입력 받고 ADC 과정을 거친 후, TFT-LCD에 값을 출력한다.

이론

(1) TFT-LCD

초 박막 액정표시장치(Thin Film Transistor Liquid Crystal Display)의 약자로, 액체와 고체의 중간 특성을 가진 액정의 상태 변화와 편광판의 편광 성질을 이용하여, 통과하는 빛의 양을 조절함으로써 정보를 표시하는 첨단 디지털 디스플레이이다.

Color Filter**,** TFT기판, 액정, Back Light Unit으로 구성되어 있다. Color Filter는 RGB 픽셀이 유리판에 코딩 되어 컬러 영상을 구현하는 역할을 하며, TFT가 형성된 두 장의 유리기판은 액정을 제어한다. Back Light Unit는 광원 역할을 한다.

(2) Timing Diagram

각 신호들이 시간 별로 처리되는 과정을 그림으로 나타낸 것으로, 제어신호 CS, WR, D/C, RD 에 따라 처리 과정이 결정된다. 제어신호는 $\overline{CS}$, $D/\bar{C}$, $\overline{WR}$, $\overline{RD}$ 로 4가지이다.

1. $\overline{CS}$ (Chip Select) : Falling Edge(High 에서 Low)일 때, LCD Chip을 사용한다.
2. $D/\bar{C}$ (Data/Command) : low에서 Command, high에서 Data를 전송한다.
3. $\overline{WR}$, $\overline{RD}$ (Write/Read) : Falling Edge(High 에서 Low)일 때, Data를 Write/Read 한다

[그림1] 은 Write Cycle의 Timing Diagram이다. 살펴보면

1. Command를 Write하기 위해선 $\overline{CS}$: low, $\overline{WR}$: low, $\boldsymbol{D/\overline{C}}$ **: low** 로 두고 Command를 전송한 뒤, $\overline{CS}$: high, $\overline{WR}$: high로 돌려놓는다.

2. Data를 Write하기 위해선 $\overline{CS}$: low, $\overline{WR}$: low, $\boldsymbol{D/\overline{C}}$ **: high** 로 두고 Data를 전송한 뒤, $\overline{CS}$: high, $\overline{WR}$: high로 돌려놓는다.

이는 $\boldsymbol{D/\overline{C}}$에 따라 Command를 전송할지, Data를 전송할지 정해지기 때문이다. 해당 동작방식은 실험 과정 중 lcd.c의 Write 관련 코드를 작성할 때 쓰인다.

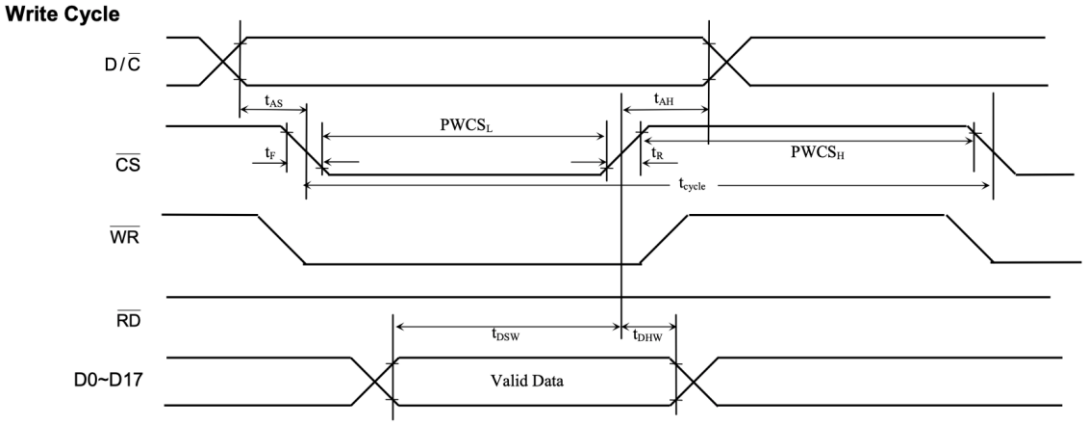

그림1. Write Cycle의 Timing Diagram

(3) ADC

Analog to Digital Converter의 약자로 아날로그 신호(조도, 온도, 습도 등)를 디지털 값으로 변환하는 것이다. 표본화 양자화, 부호화 3단계를 거친다.

x축 간격을 나눠 아날로그를 디지털화 시키고(표본화), y축도 아날로그값을 레벨을 나눠 디지털화를 시킨다(양자화). 레벨에 속한 값을 이진수로 변환한다(부호화).

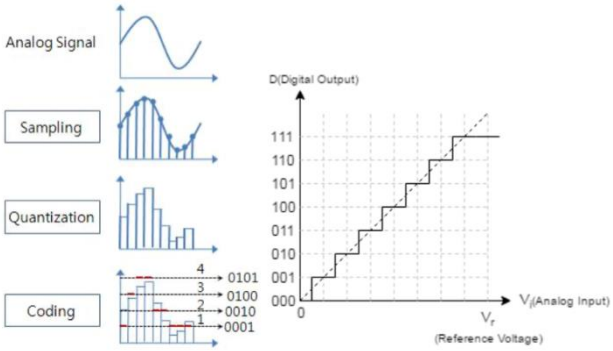

그림2. ADC 3단계

(4) 조도센서

주변의 밝기를 측정하는 센서로, 빛의 양이 많아질수록 전도율이 높아져 저항이 낮아진다. 이번 실험에서는 조도센서의 값을 아날로그로 받은 뒤, ADC 과정을 거쳐줄 것이다.

실험 과정

Lcd.c 코드

WR_REG, WR_DATA 함수를 작성한다. Write Cycle의 Timing Diagram을 참고했다.

Command를 Write하기 위해선 $\overline{CS}$: low, $\overline{WR}$: low, $D/\bar{C}$: low 로 두고 Command를 전송한 뒤, $\overline{CS}$: high, $\overline{WR}$: high로 돌려놓는다. 반대로 Data를 Write하기 위해선, $D/\bar{C}$: high 로 둔다.

```
17  static void LCD_WR_REG(uint16_t LCD_Reg)
18  {
19          // TODO implement using GPIO_ResetBits/GPIO_SetBits
20          GPIO_ResetBits(GPIOD, GPIO_Pin_13);   // LCD_RS(0);
21          GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
22          GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);
23
24          GPIO_Write(GPIOE, LCD_Reg);
25
26          // TODO implement using GPIO_ResetBits/GPIO_SetBits
27          GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
28          GPIO_SetBits(GPIOD, GPIO_Pin_14);  // LCD_WR(1);
29
30  }
31
32  static void LCD_WR_DATA(uint16_t LCD_Data)
33  {
34          // TODO implement using GPIO_ResetBits/GPIO_SetBits
35          GPIO_SetBits(GPIOD, GPIO_Pin_13);   // LCD_RS(1);
36          GPIO_ResetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(0);
37          GPIO_ResetBits(GPIOD, GPIO_Pin_14); // LCD_WR(0);
38
39          GPIO_Write(GPIOE, LCD_Data);
40
41          // TODO implement using GPIO_ResetBits/GPIO_SetBits
42          GPIO_SetBits(GPIOC, GPIO_Pin_6);  // LCD_CS(1);
43          GPIO_SetBits(GPIOD, GPIO_Pin_14);  // LCD_WR(1);
44
45  }
```

Main.c 코드

1. RCC_Configure - clock enable

ADC, LCD, NVIC를 사용하기 위해 각각 clock을 연결해준다.

```
23  void RCC_Configure(void)
24  {
25      // TODO: Enable the APB2 peripheral clock using the function
26
27      /* Alternate Function IO clock enable */
28      RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
29
30      /* ADC */
31      RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
32
33      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
34  }
```

2. ADC_Configure – ADC 구조체 INIT

사용할 채널 개수는 1이다(43 line). ppt에서 나와 있듯이 ADC1(45 line)을 사용하고, 10채널(46 line)을 사용했다.

```
36  void ADC_Configure(void) {
37      ADC_InitTypeDef ADC_InitStructure;
38      ADC_InitStructure.ADC_ScanConvMode = DISABLE;
39      ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
40      ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
41      ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
42      ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
43      ADC_InitStructure.ADC_NbrOfChannel = 1;
44
45      ADC_Init(ADC1, &ADC_InitStructure);
46      ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_239Cycles5);
47      ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);
48      ADC_Cmd(ADC1, ENABLE);
49      ADC_ResetCalibration(ADC1);
50      while(ADC_GetResetCalibrationStatus(ADC1));
51      ADC_StartCalibration(ADC1);
52      while(ADC_GetCalibrationStatus(ADC1));
53      ADC_SoftwareStartConvCmd(ADC1, ENABLE);
54  }
```

3. GPIO Configuration

```
56  void GPIO_Configure(void)
57  {
58      GPIO_InitTypeDef GPIO_InitStructure;
59
60      // TODO: Initialize the GPIO pins using the structure 'GPIC
61
62      // ADC
63      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
64      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
65      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
66      GPIO_Init(GPIOC, &GPIO_InitStructure);
67  }
```

4. 조도센서 값 전역변수 지정

```
21  uint16_t ADC1_CONVERTED_VALUE;
```

5. NVIC_Configure – 우선순위 지정

우선순위는 ADC 인터럽트 하나만 사용한다. 참고자료를 찾아보면 ADC1_2_IRQn이다. 0 순위로 지정해주었다.

```
70  void NVIC_Configure(void) {
71
72      NVIC_InitTypeDef NVIC_InitStructure;
73
74      // TODO: fill the arg you want
75      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
76
77      // TODO: Initialize the NVIC using the structure 'NVIC_InitTypeDef' ar
78
79      // 'NVIC_EnableIRQ' is only required for USART setting
80      // NVIC_EnableIRQ(USART1_IRQn);
81      NVIC_InitStructure.NVIC_IRQChannel = ADC1_2_IRQn;
82      NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; // TODO
83      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0; // TODO
84      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
85      NVIC_Init(&NVIC_InitStructure);
```

6. ADC1_2_IRQHandler - 조도센서 값 읽어오기

조도센서 값 변경때마다 작동하며, 아까 전역변수로 지정해줬던 ADC1_CONVERTED_VALUE에 값을 넣어준다.

```
95  void ADC1_2_IRQHandler(void) {
96      uint16_t diff = ADC_GetConversionValue(ADC1) - ADC1_CONVERTED_VALUE;
97      ADC1_CONVERTED_VALUE = ADC_GetConversionValue(ADC1);
98  }
```

7. main 함수

lcd화면 터치 좌표를 읽고, 해당 좌표에 원을 그려준다. 조도센서 값을 출력해준다. Lcd.h , touch.h 를 참고하였다.

```
116       uint16_t x;
117       uint16_t y;
118       uint16_t convert_x;
119       uint16_t convert_y;
120       while(1) {
121           Touch_GetXY(&x, &y, 0);
122           Convert_Pos(x, y, &convert_x, &convert_y);
123           if (T_INT != 1) {
124             LCD_ShowNum(0x10, 0x60, convert_x, 16, BLACK, WHITE);
125             LCD_ShowNum(0x10, 0x50, convert_y, 16, BLACK, WHITE);
126             //LCD_DrawRectangle(convert_x, convert_y, convert_x + 10, convert_y + 10);
127             LCD_DrawCircle(convert_x,convert_y,4);
128             LCD_ShowNum(0x10, 0x80, ADC1_CONVERTED_VALUE, 16, BLACK, WHITE);
129             Delay();
130           }
131           LCD_ShowString(0x45, 0x30, "WED_TEAM03", BLACK, WHITE );
132
133       }
134       return 0;
```

실험 결과

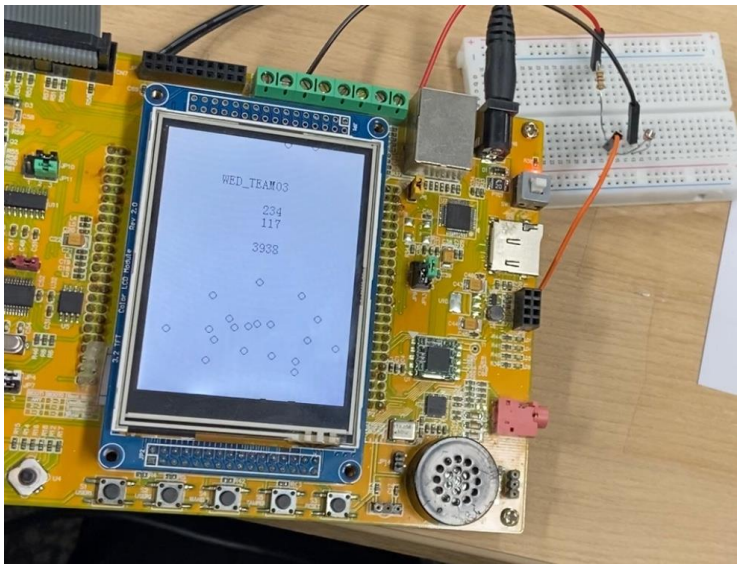

그림3. 실험결과

LCD화면에 팀이름이 출력되며, 화면을 터치할 때 마다, 누른 lcd화면 좌표에 원을 그려주고 조도 센서의 값을 받아와 출력해주는 것을 볼 수 있다.

고찰

처음엔 조도센서 값을 계속 입력 받고 출력하게 했다. 그러나 LCD 출력 업데이트는 LCD화면 터치 시 한번만 일어나게 하라는 것을 보고 main.c를 수정하였다. 그 결과 계속 변화되어 출력되던 조도센서 값이 화면 터치시에만 바뀌게 되었다.