임베디드 설계 및 실험

실험 결과 보고서

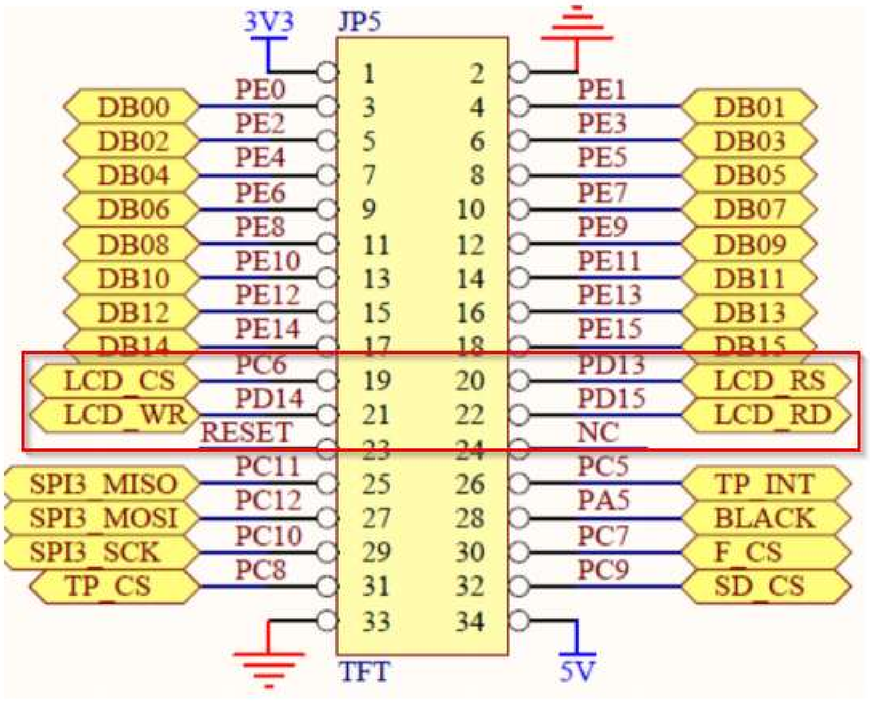
10 주차 3 조

실험 내용

1. Lcd.c 에서 write 관련 코드 작성
2. ADC 설정/interrupt 설정
3. Main.c 작성
   - LCD 에 Text(팀명) 출력
   - LCD 터치 시 좌표(X, Y) 출력
   - 센서 값 LCD 에 출력
   - LCD 에 버튼 4 개 만들고 해당 버튼 클릭 시 LED On
4. 실험 결과

Lcd.c 완성

Lcd 의 schematic 는 다음과 같다.



- LCD_WR_REG

  LCD_CS 는 Low 로 Falling Edge => High 에서 Low 로 변경하면 됨.

  LCD_RS(핀맵에서 RS 를 의미함)는 Command 를 전송해야 됨 => Low

  LCD_WR 는 Low 로 Falling Edge => High 에서 Low 로 변경하면 됨.

    D13 핀은 Low 로,

    C6, D14 핀을 reset 시켜 Failing Edge 를 만들고 그 뒤에 다시 1 로 원위치.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
    // TODO implement using GPIO_ResetBits/GPIO_SetBits
    GPIO_SetBits(GPIOD, GPIO_Pin_15);
    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(GPIOD, GPIO_Pin_14);
    GPIO_SetBits(GPIOC, GPIO_Pin_6);

}
```

- LCD_WR_DATA

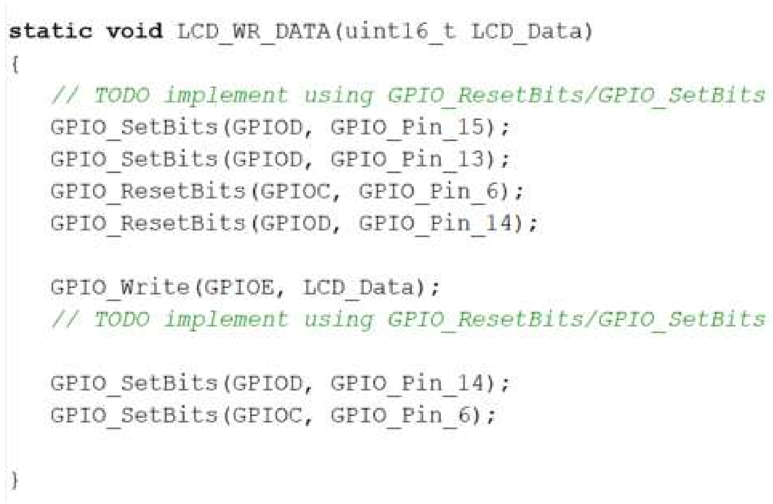
  LCD_CS 는 Low 로 Falling Edge => High 에서 Low 로 변경하면 됨.

  LCD_RS(핀맵에서 RS 를 의미함)는 Command 를 전송해야 됨 => Low

  LCD_RD 는 Low 로 Falling Edge => High 에서 Low 로 변경하면 됨.

    D13 는 High,

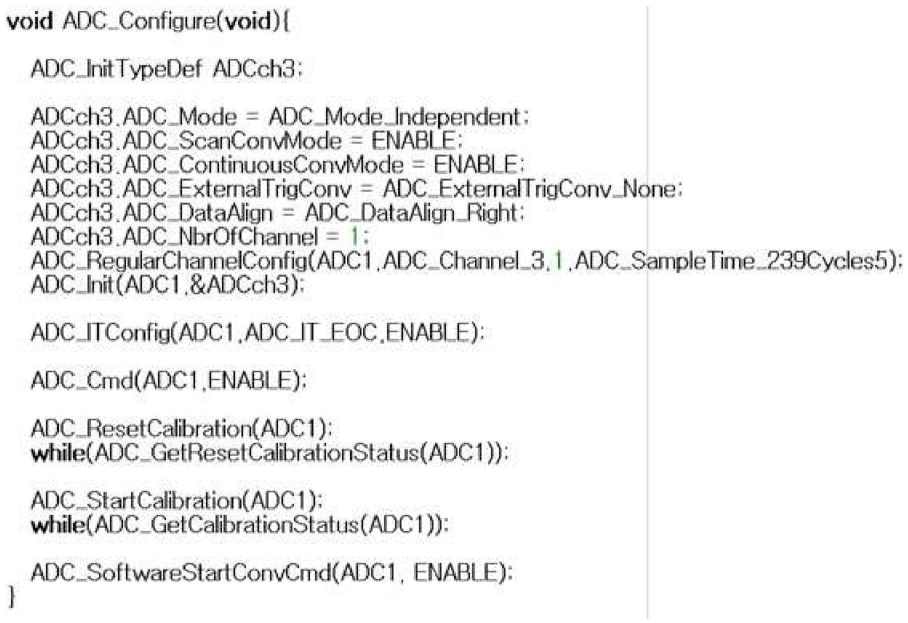
    C6, D15 핀을 reset 시켜 Failing Edge 를 만들고 그 뒤에 다시 1 로 원위치.



```
static void LCD_WR_DATA(uint16_t LCD_Data)
{
   // TODO implement using GPIO_ResetBits/GPIO_SetBits
   GPIO_SetBits(GPIOD, GPIO_Pin_15);
   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(GPIOD, GPIO_Pin_14);
   GPIO_SetBits(GPIOC, GPIO_Pin_6);

}
```

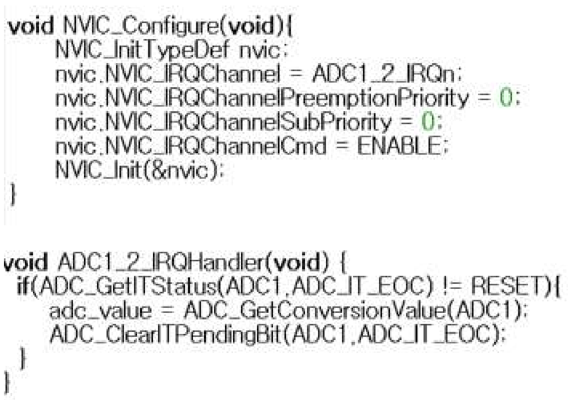
ADC Channel 설정

- ADC 값 읽기는 interrupt 를 사용함.



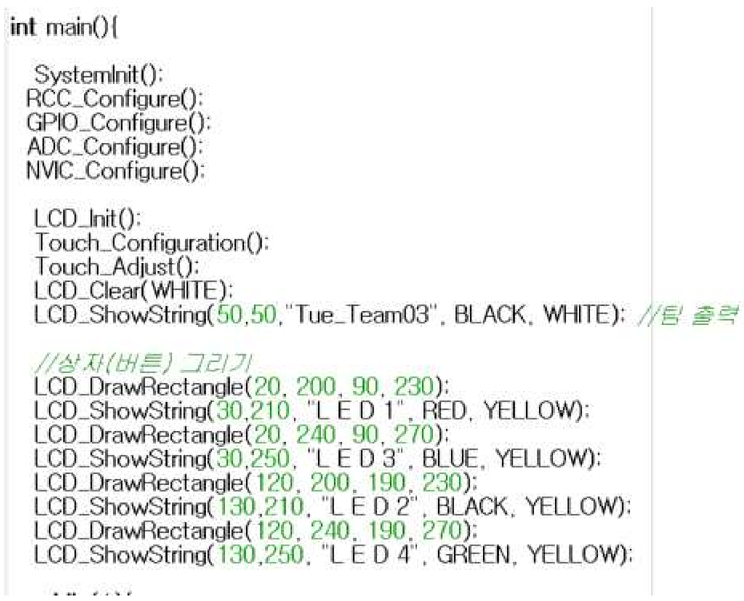
```
void ADC_Configure(void){
  ADC_InitTypeDef ADCch3;

  ADCch3.ADC_Mode = ADC_Mode_Independent;
  ADCch3.ADC_ScanConvMode = ENABLE;
  ADCch3.ADC_ContinuousConvMode = ENABLE;
  ADCch3.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADCch3.ADC_DataAlign = ADC_DataAlign_Right;
  ADCch3.ADC_NbrOfChannel = 1;
  ADC_RegularChannelConfig(ADC1,ADC_Channel_3,1,ADC_SampleTime_239Cycles5);
  ADC_Init(ADC1,&ADCch3);

  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);
}
```

Interrupt 설정



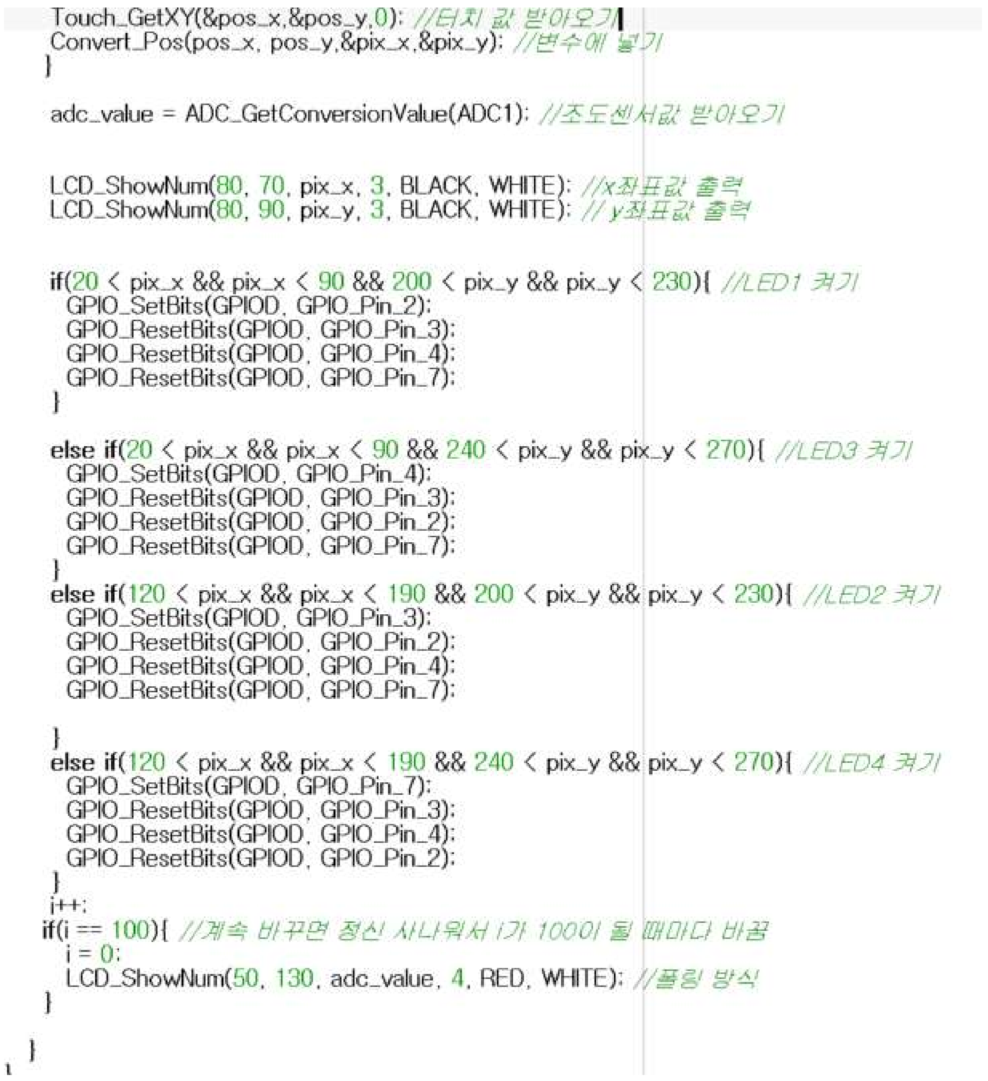
```
void NVIC_Configure(void){
      NVIC_InitTypeDef nvic;
      nvic.NVIC_IRQChannel = ADC1_2_IRQn;
      nvic.NVIC_IRQChannelPreemptionPriority = 0;
      nvic.NVIC_IRQChannelSubPriority = 0;
      nvic.NVIC_IRQChannelCmd = ENABLE;
      NVIC_Init(&nvic);
}

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

LCD 에 화면 띄우기



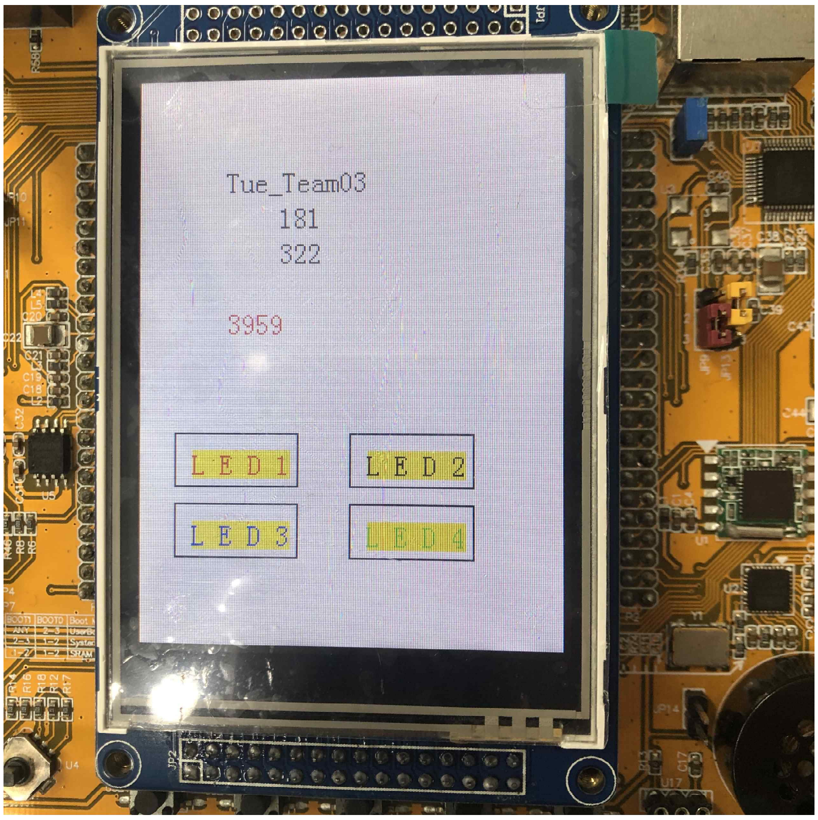
```
int main(){

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

   LCD_Init();
   Touch_Configuration();
   Touch_Adjust();
   LCD_Clear(WHITE);
   LCD_ShowString(50,50,"Tue_Team03", BLACK, WHITE); //팀 출력

   //상자(버튼) 그리기
   LCD_DrawRectangle(20, 200, 90, 230);
   LCD_ShowString(30,210, "L E D 1", RED, YELLOW);
   LCD_DrawRectangle(20, 240, 90, 270);
   LCD_ShowString(30,250, "L E D 3", BLUE, YELLOW);
   LCD_DrawRectangle(120, 200, 190, 230);
   LCD_ShowString(130,210, "L E D 2", BLACK, YELLOW);
   LCD_DrawRectangle(120, 240, 190, 270);
   LCD_ShowString(130,250, "L E D 4", GREEN, YELLOW);
```

화면에 터치 위치 출력/ 터치 위치에 따라 LED on,off / 조도 센서 값 주기적으로 출력



```
      Touch_GetXY(&pos_x,&pos_y,0); //터치 값 받아오기
      Convert_Pos(pos_x, pos_y,&pix_x,&pix_y); //변수에 넣기
    }

      adc_value = ADC_GetConversionValue(ADC1); //조도센서값 받아오기


      LCD_ShowNum(80, 70, pix_x, 3, BLACK, WHITE); //x좌표값 출력
      LCD_ShowNum(80, 90, pix_y, 3, BLACK, WHITE); // y좌표값 출력


      if(20 < pix_x && pix_x < 90 && 200 < pix_y && pix_y < 230){ //LED1 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      }

      else if(20 < pix_x && pix_x < 90 && 240 < pix_y && pix_y < 270){ //LED3 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      }
      else if(120 < pix_x && pix_x < 190 && 200 < pix_y && pix_y < 230){ //LED2 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);

      }
      else if(120 < pix_x && pix_x < 190 && 240 < pix_y && pix_y < 270){ //LED4 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_7);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
      }
      i++;
     if(i == 100){ //계속 바꾸면 정신 사나워서 i가 100이 될 때마다 바꿈
        i = 0;
        LCD_ShowNum(50, 130, adc_value, 4, RED, WHITE); //폴링 방식
    }

  }
}
```

실행 결과



- 나머지는 동영상으로 첨부.

전체 코드

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

int color[12]={WHITE,CYAN,BLUE,RED,MAGENTA,LGRAY,GREEN,YELLOW,BROWN,BRRED,GRAY};
int i = 0;
uint16_t adc_value;
uint16_t pos_x,pos_y;
uint16_t pix_x,pix_y;

void GPIO_Configure() {

   GPIO_InitTypeDef GPIO_InitStructure;
   GPIO_InitTypeDef PA3;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
   GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7); //LED
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_Init(GPIOD, &GPIO_InitStructure);

   PA3.GPIO_Mode = GPIO_Mode_AIN;
   PA3.GPIO_Pin = GPIO_Pin_3; //ADC
   GPIO_Init(GPIOA,&PA3);
}

void RCC_Configure(void){
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
    /*TODO : APB2PeriphClockEnable */
     RCC_APB2PeriphClockCmd( // 사용하고자하는 port들의 clock을 enable 한다.
     RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOD | RCC_APB2Periph_ADC1, ENABLE); //ADC1 레퍼런
스 146p에 있음
}


void ADC_Configure(void){
    ADC_InitTypeDef ADCch3;

    ADCch3.ADC_Mode = ADC_Mode_Independent;
    ADCch3.ADC_ScanConvMode = ENABLE;
    ADCch3.ADC_ContinuousConvMode = ENABLE;
    ADCch3.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
    ADCch3.ADC_DataAlign = ADC_DataAlign_Right;
    ADCch3.ADC_NbrOfChannel = 1;
    ADC_RegularChannelConfig(ADC1,ADC_Channel_3,1,ADC_SampleTime_239Cycles5);
    ADC_Init(ADC1,&ADCch3);

    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);
}

void NVIC_Configure(void){
        NVIC_InitTypeDef nvic;
        nvic.NVIC_IRQChannel = ADC1_2_IRQn;
        nvic.NVIC_IRQChannelPreemptionPriority = 0;
        nvic.NVIC_IRQChannelSubPriority = 0;
        nvic.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&nvic);
}


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

```
int main(){

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

   LCD_Init();
   Touch_Configuration();
   Touch_Adjust();
   LCD_Clear(WHITE);
   LCD_ShowString(50,50,"Tue_Team03", BLACK, WHITE); //팀 출력

   //상자(버튼) 그리기
   LCD_DrawRectangle(20, 200, 90, 230);
   LCD_ShowString(30,210, "L E D 1", RED, YELLOW);
   LCD_DrawRectangle(20, 240, 90, 270);
   LCD_ShowString(30,250, "L E D 3", BLUE, YELLOW);
   LCD_DrawRectangle(120, 200, 190, 230);
   LCD_ShowString(130,210, "L E D 2", BLACK, YELLOW);
   LCD_DrawRectangle(120, 240, 190, 270);
   LCD_ShowString(130,250, "L E D 4", GREEN, YELLOW);

   while(1){
     if(!T_INT){ //LCD를 터치했을 때만 측정하게 함
      Touch_GetXY(&pos_x,&pos_y,0); //터치 값 받아오기
      Convert_Pos(pos_x, pos_y,&pix_x,&pix_y); //변수에 넣기
     }

      adc_value = ADC_GetConversionValue(ADC1); //조도센서값 받아오기


      LCD_ShowNum(80, 70, pix_x, 3, BLACK, WHITE); //x좌표값 출력
      LCD_ShowNum(80, 90, pix_y, 3, BLACK, WHITE); // y좌표값 출력


      if(20 < pix_x && pix_x < 90 && 200 < pix_y && pix_y < 230){ //LED1 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      }

      else if(20 < pix_x && pix_x < 90 && 240 < pix_y && pix_y < 270){ //LED3 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);
      }
      else if(120 < pix_x && pix_x < 190 && 200 < pix_y && pix_y < 230){ //LED2 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_7);

      }
      else if(120 < pix_x && pix_x < 190 && 240 < pix_y && pix_y < 270){ //LED4 켜기
        GPIO_SetBits(GPIOD, GPIO_Pin_7);
        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_4);
        GPIO_ResetBits(GPIOD, GPIO_Pin_2);
      }
      i++;
     if(i == 100){ //계속 바꾸면 정신 사나워서 i가 100이 될 때마다 바꿈
        i = 0;
        LCD_ShowNum(50, 130, adc_value, 4, RED, WHITE); //폴링 방식
     }

   }
}
```