임베디드 시스템 설계 및 실험 보고서
11 주차 실험_Direct Memory Access

분반 : 001 분반
교수님 : 정 상화 교수님
조교님 : 유 동화 조교님
실험일 : 2019-11-11
제출일 : 2019-11-18

00. 목차

7 조
장 수현
박 창조
임 다영
이 힘찬

01. 실험 목적

- TFT LCD 의 이해 및 제어
- ADC(조도센서)의 이해 및 제어
- DMA(Direct Memory Access)의 이해, 제어 및 구현

02. 실험 과제

- TFT LCD 에 ADC(조도 센서)의 변화 값 출력(ADC 인터럽트 사용 금지)
- DMA 를 사용하여 ADC 1 개의 채널을 통해 1 개의 조도센서 값을 받아와서 출력
- 센서 값에 적당한 기준(밝음/어두움)을 둬서 어둡다고 판단되면 LED 점등, 밝다고 판단되면 LED 소등

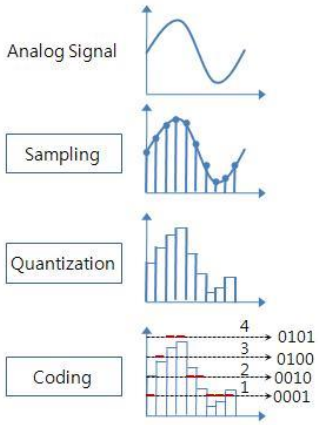
03. 실험 준비

3.1 실험에 필요한 기초지식

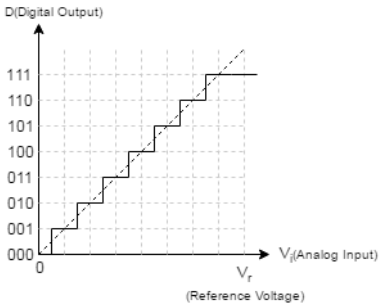
3.1.1 TFT-LCD

- 초박막 액정표시장치(Thin Film Transistor Liquid Crystal Display)의 약자
- 액체와 고체의 중간 특성을 가진 액정의 상태 변화와 편광판의 편광 성질 이용
- 통과하는 빛의 양을 조절함으로써 정보를 표시하는 첨단 디지털 디스플레이
- TFT-LCD 는 Color Filter 와 TFT 가 형성된 두 장의 유리기판과 그 사이에 주입된 액정(Liquid Crystal), 광원(Back Light Unit)으로 구성됨.

3.1.2 ADC



- 아날로그 신호를 디지털 신호로 변환한다.
- 아날로그 신호가 입력되어 들어오면 이를 샘플링하여 양자화를 거쳐 부호화 한다.
- 실험에서는 ADC 의 한 종류인 조도센서를 사용한다.



- 부호화(coding) : 신호처리가 용이한 디지털 코드 형태로 변환
- 샘플링(sampling) : 시간 축 방향으로 연속된 아날로그 신호를 특정 시간 간격으로 나누고, 나눈 시간 간격의 값을 추출해서 추출한 표본으로 샘플 생성
- 양자화(Quantization) : 일반적으로 ADC 의 아날로그 입력(Vi)과 디지털 출력(D)의 상관관계는

  D = Vi * Vr * 2n 으로 표현할 수 있다.(Vr 은 레퍼런스 전압, n 은 디지털 출력 비트 수)

3.1.3 DAC

- 디지털 신호를 아날로그 신호로 변환한다.

3.1.4 조도 센서(Photo Resistor)

- 주변의 밝기를 측정하는 센서
- 빛의 양이 많아질수록 전도율이 높아져 저항이 낮아진다.(반비례)
- 조도 센서를 가리면 저항이 높아져 TFT LCD 에 출력되는 숫자가 높아진다.

3.1.5 DMA(Direct Memory Access)

- CPU 가 관여하지 않고도 RAM 을 업데이트 하기 위하여 사용
- Circular Buffer 사용, 메모리 Offset, DMA Offset 등 다양한 옵션이 존재한다.
- 다른 외부 장치에서 DMA 를 읽어오는 것도 가능하다.

3.2 실험 진행 시 주의사항

- DMA_Configure() 사용
- DMA 를 위해 조도 센서 값이 들어가는 변수는 전역으로 선언
- ADC 선은 저항 가운데에 꽂아야 한다.
- 조도 센서의 +는 5V 에, -는 GND 에, 저항이랑 연결된 곳은 디지털 핀에 연결한다.
- 최적화 level 을 high 로 설정한다.

3.3 환경 설정

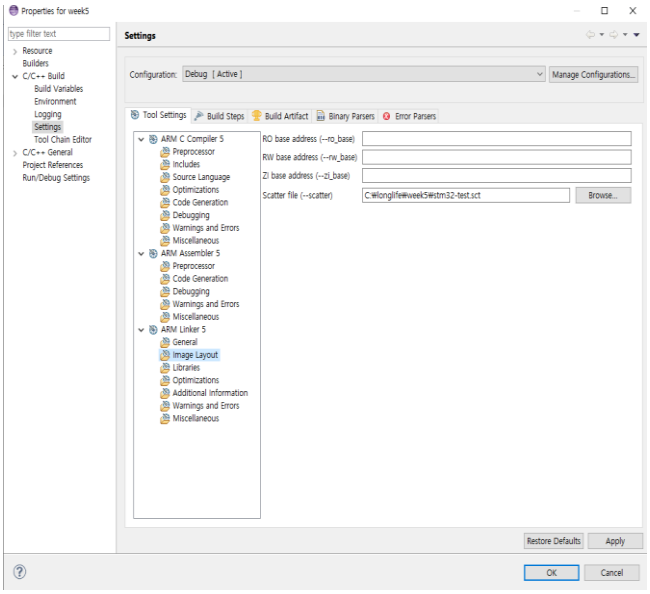
실험을 진행하기 전 아래의 프로젝트 생성 및 기본 환경 설정을 해주어야 한다.

3.3.1 프로젝트 생성

C 언어로 소스코드를 작성하기 때문에 C++프로젝트가 아닌 C 프로젝트를 생성해준다. Project type 은 Bare-metal Executable -〉 Empty Project 로, Toolchains 은 Arm Compiler 5 로 설정한다.

3.3.2 프로젝트 Properties-Settings

4 주차 실험과 동일하게 스캐터 파일을 이용한다.



3.3.3 보드 연결

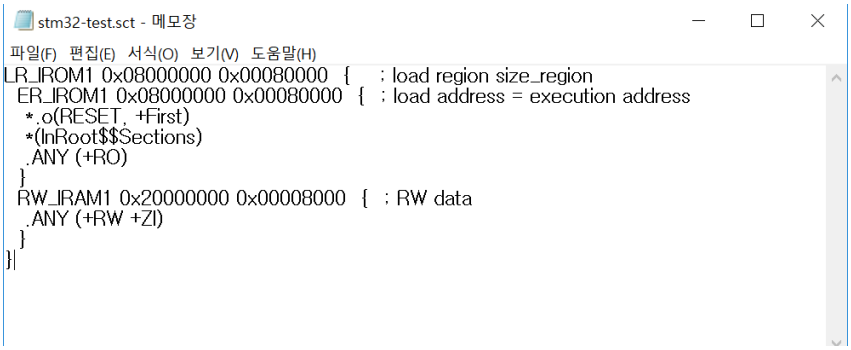
보드 연결 시에는 반드시 연결 순서를 지키고, 규격에 맞는 전원선을 사용해야한다.

연결 순서 : 보드와 Dstream JTAG 연결 -〉 보드전원선 연결(보드 전원은 OFF) -〉 Dstream 전원 연결 및 ON -〉 Dstream Status LED 점등 확인 후 보드 전원 ON -〉 Dstream Target LED 점등 확인 후 DS-5 에서 'connect target'

3.3.4 데이터 베이스 설정

Dstream 를 USB 로 컴퓨터에 연결하고 Debug Hardware config 에서 보드를 connect 한다. 보드 연결 후에는 Auto Configure 를 클릭하여 설정을 진행하고, rvc 파일로 저장한다. rvc 파일 저장 경로에 한글이 들어가지 않도록 주의한다.

3.3.5 Scatter file 수정



스캐터 파일의 메모리 범위를 위와 같이 수정한다.

3.3.6 Optimization level 설정

실험에서 사용하는 라이브러리의 용량이 크기 때문에 최적화 레벨을 high 로 설정한다. (O2 로 설정하되 필요시 O3 설정 가능)

Project -〉 Option -〉 Settings -〉 Optimization level high

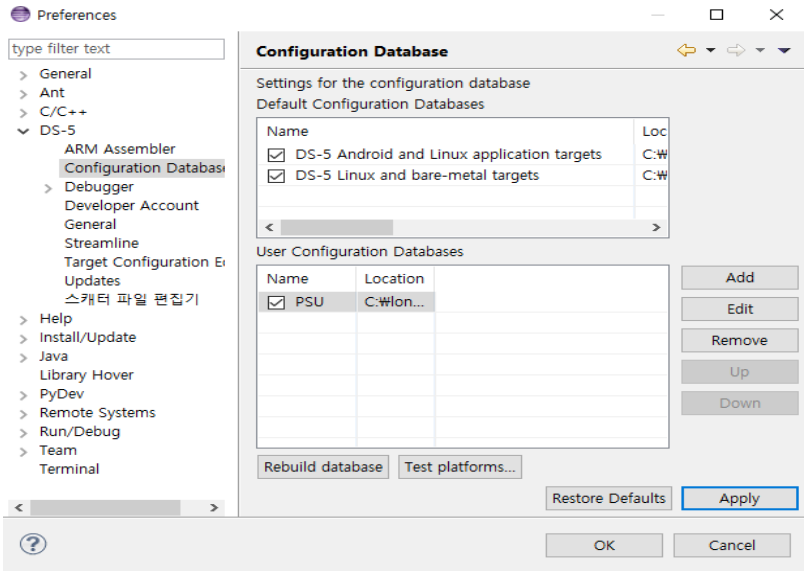
3.3.7 cdbimporter

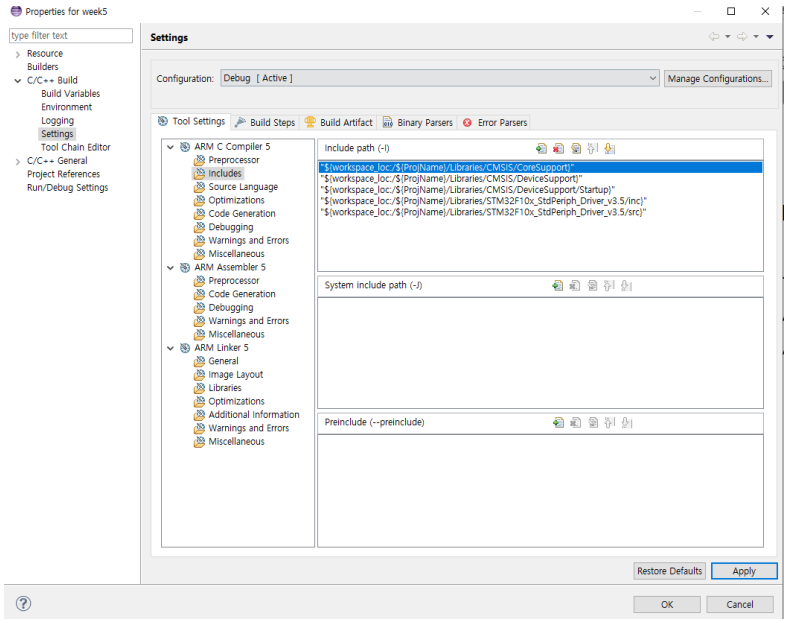
DS-5 Command Prompt 에서 RVC 파일이 있는 폴더로 이동 후 아래와 같이 명령문을 작성한다.

3.3.8 데이터 베이스 등록 및 라이브러리 추가

아래와 같이 데이터 베이스 등록 후 프로젝트 폴더에 라이브러리를 추가한다.

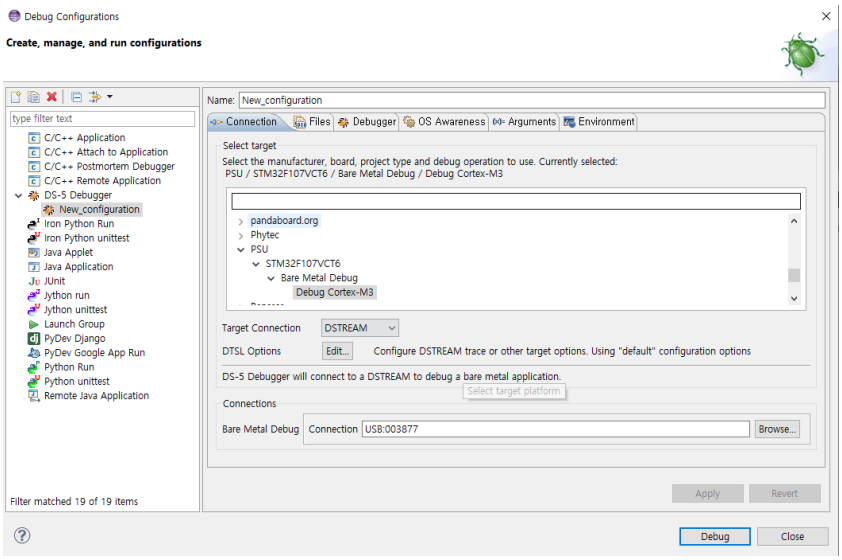
프로젝트 파일이 있는 경로에 font.h, lcd.c, lcd.h, touch.c, touch.h 를 추가한다.

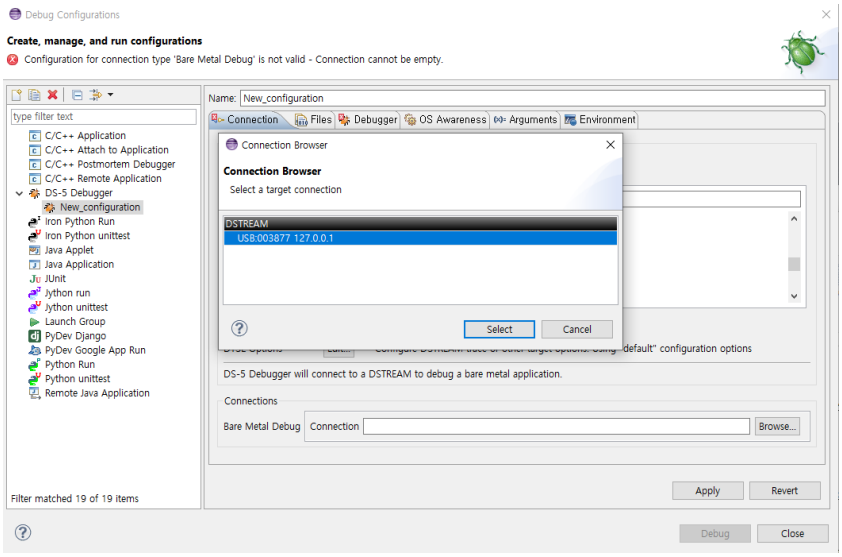


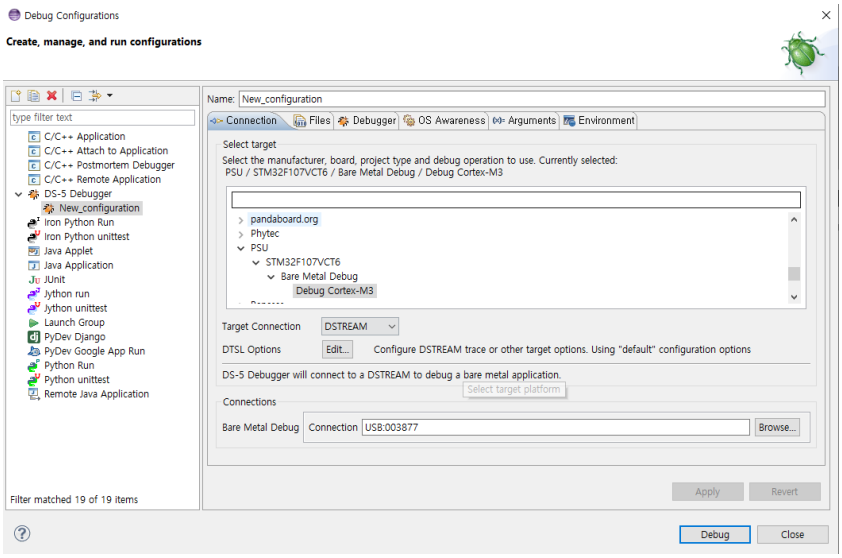


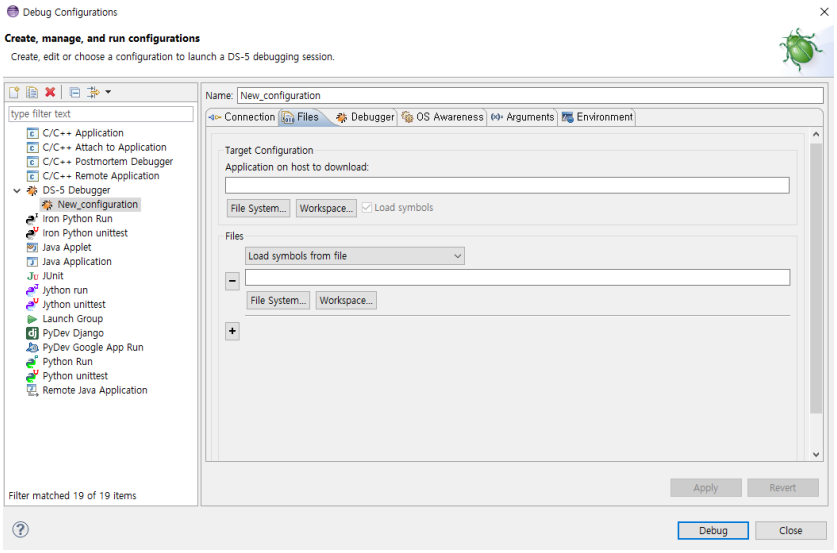
3.3.9 디버그 설정

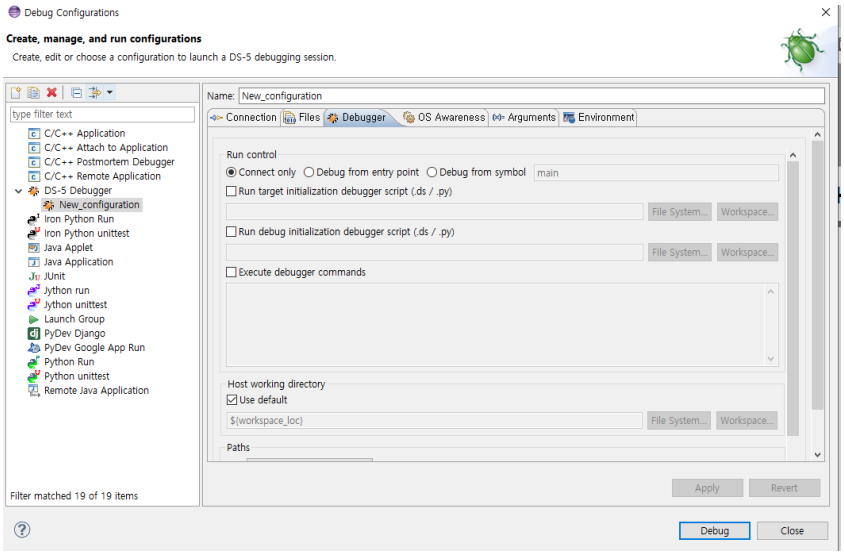
C 언어 소스파일을 만들어 빌드하고 디버그 설정을 한다. 4 주차와 동일한 환경으로 진행한다.






Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Select target
Select the manufacturer, board, project type and debug operation to use. Currently selected:
PSU / STM32F107VCT6 / Bare Metal Debug / Debug Cortex-M3
pandaboard.org
Phytec
PSU
STM32F107VCT6
Bare Metal Debug
Debug Cortex-M3
Target Connection
DSTREAM
DTSL Options
Edit...
Configure DSTREAM trace or other target options. Using "default" configuration options
DS-5 Debugger will connect to a DSTREAM to debug a bare metal application.
Select target platform
Connections
Bare Metal Debug
Connection
USB:003877
Browse...
Apply
Revert
Debug
Close

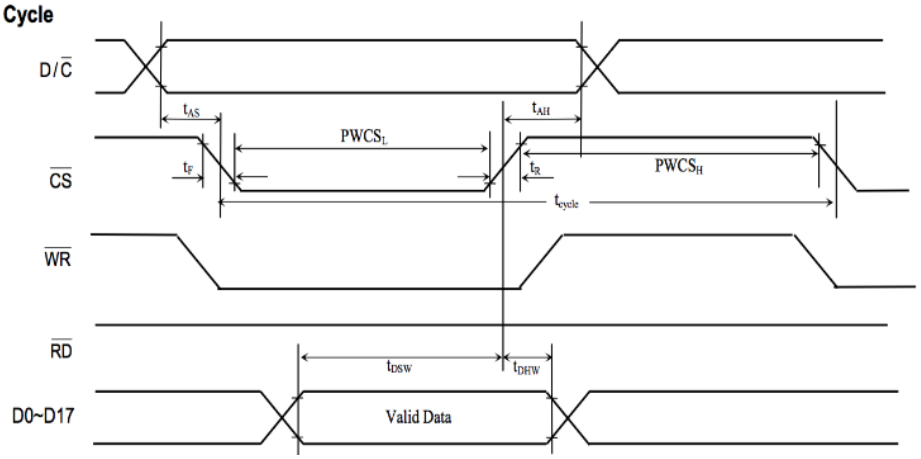

Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Target Configuration
Application on host to download:
File System...
Workspace...
Load symbols
Files
Load symbols from file
File System...
Workspace...
Apply
Revert
Debug
Close


Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Run control
Connect only
Debug from entry point
Debug from symbol
main
Run target initialization debugger script (.ds / .py)
File System...
Workspace...
Run debug initialization debugger script (.ds / .py)
Execute debugger commands
Host working directory
Use default
${workspace_loc}
Paths
Apply
Revert
Debug
Close

04. 실험 및 과제 해결

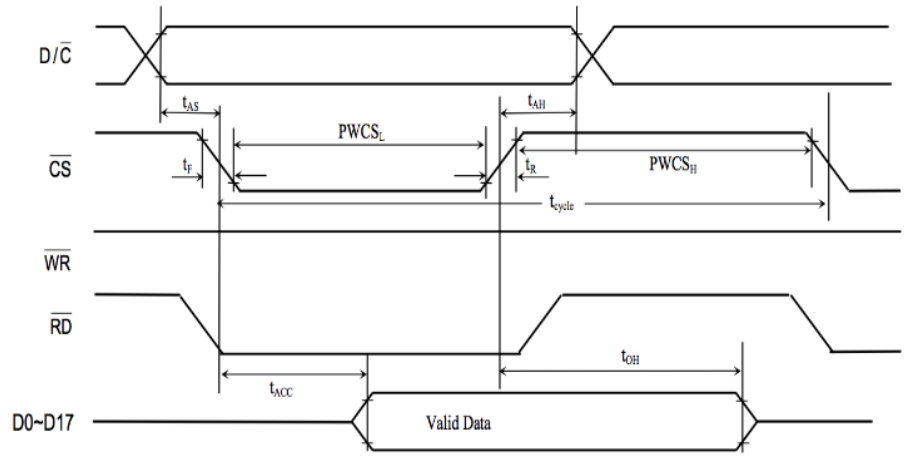
4.1 LCD Library 수정

Write Cycle



1. CS 를 low 로 하여 chip 동작시킨다.
2. RS(=D/C)가 low 라면 command, high 라면 display data 가 register 에 write 된다.
3. Write cycle 이므로 RD 는 항상 high 이다.
4. WR 이 low-〉high 일 때 display RAM 에 data/ register 에 command 를 write 한다.

Read Cycle



1. CS 를 low 로 하여 chip 동작시킨다.
2. RS 를 high 로 하여 display RAM 에 저장된 display data 를 읽어오도록 한다.
3. Read cycle 이므로 WR 는 high 를 유지한다.
4. RD 가 low-〉high 일 때 display data 를 읽는다.

위의 Timing diagram 을 참고하여 lcd.c 파일을 아래와 같이 수정한다.

```
static void LCD_WR_REG(uint16_t LCD_Reg)
{
    LCD_CS(0);
```

```
    LCD_RS(0);
    LCD_WR(0);
    GPIO_Write(GPIOE, LCD_Reg);
    LCD_WR(1);
    LCD_CS(1);
}
static void LCD_WR_DATA(uint16_t LCD_Data)
{
    LCD_CS(0);
    LCD_RS(1);
    LCD_WR(0);
    GPIO_Write(GPIOE, LCD_Data);
    LCD_WR(1);
    LCD_CS(1);
}
static uint16_t LCD_ReadReg(uint16_t LCD_Reg)
{
      uint16_t temp;
      GPIO_InitTypeDef GPIO_InitStructure;
      LCD_WR_REG(LCD_Reg);
      // To Read from Data Line
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
      GPIO_Init(GPIOE, &GPIO_InitStructure);
      LCD_CS(0);
      LCD_RS(1);
      LCD_RD(0);
      temp = GPIO_ReadInputData(GPIOE);
      LCD_RD(1);
      LCD_CS(1);
      // Read Done, Reset
      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_All;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
      GPIO_Init(GPIOE, &GPIO_InitStructure);
      return temp;
}
```

4.2 RCC & GPIO Configure.

- 실험에 필요한 포트에 클럭을 인가하고 아래와 같이 GPIO 를 설정하였다.

```
void RCC_Configure() {
         RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
         RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE | RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1,
ENABLE);
         RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
}

void GPIO_Configure() {
         GPIO_InitTypeDef GPIO_InitStructure;

         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
         GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2);
         GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
         GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

4.3 ADC Configure

- ADC channel 을 사용하여 센서 값을 LCD 에 출력하기 위한 ADC configure 은 아래와 같다.
- 이번 실험에서는 ADC1 을 사용하므로 Cmd, ChannelConfig, Calibration(Reset/Start)에 ADC1 을 사용한다.
- 각각 ResetCalibration, StartCalibration 이후에 while 문을 사용해 상태를 확인해서 제대로 셋팅 되었는지 확인한다.
- ADC_DMACmd 함수를 추가로 실행하여 ADC1 의 DMA Request 를 ENABLE 한다.

```
typedef struct
{
  uint32_t ADC_Mode;                   /*!< Configures the ADC to operate in independent or
                                            dual mode.
                                            This parameter can be a value of @ref ADC_mode */

  FunctionalState ADC_ScanConvMode;    /*!< Specifies whether the conversion is performed in
                                            Scan (multichannels) or Single (one channel) mode.
                                            This parameter can be set to ENABLE or DISABLE */

  FunctionalState ADC_ContinuousConvMode; /*!< Specifies whether the conversion is performed in
                                            Continuous or Single mode.
                                            This parameter can be set to ENABLE or DISABLE. */

  uint32_t ADC_ExternalTrigConv;       /*!< Defines the external trigger used to start the analog
                                            to digital conversion of regular channels. This parameter
                                            can be a value of @ref ADC_external_trigger_sources_for_regular_channels_conversion */

  uint32_t ADC_DataAlign;              /*!< Specifies whether the ADC data alignment is left or right.
                                            This parameter can be a value of @ref ADC_data_align */

  uint8_t ADC_NbrOfChannel;            /*!< Specifies the number of ADC channels that will be converted
                                            using the sequencer for regular channel group.
                                            This parameter must range from 1 to 16. */

}ADC_InitTypeDef;
```

```
void ADC_DMACmd(ADC_TypeDef* ADCx, FunctionalState NewState)
{
  /* Check the parameters */
    /*
  //assert_param(IS_ADC_DMA_PERIPH(ADCx));
  //assert_param(IS_FUNCTIONAL_STATE(NewState));
  */
  if (NewState != DISABLE)
  {
    /* Enable the selected ADC DMA request */
    ADCx->CR2 |= CR2_DMA_Set;
  }
  else
  {
    /* Disable the selected ADC DMA request */
    ADCx->CR2 &= CR2_DMA_Reset;
  }
}
```

```
void ADC_Configure() {
          ADC_InitTypeDef ADC_InitStructure;
          ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
          ADC_InitStructure.ADC_ScanConvMode = ENABLE;
          ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
          ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
          ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
          ADC_InitStructure.ADC_NbrOfChannel = 1;

          ADC_Init(ADC1, &ADC_InitStructure);
          ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_239Cycles5);
          ADC_DMACmd(ADC1, ENABLE);
          ADC_Cmd(ADC1, ENABLE);

          ADC_ResetCalibration(ADC1);
          while(ADC_GetResetCalibrationStatus(ADC1));
          ADC_StartCalibration(ADC1);
          while(ADC_GetCalibrationStatus(ADC1));
}
```

4.4 DMA Configure

- DMA 는 인터럽트를 사용하지 않는다.

- MemoryBaseAddr 에 전역변수로 선언한 ADC_V 의 주소 값을 넣는다. ADC_V 는 조도 센서 값을 받아오는 변수다. PeripheralBaseAddr 에는 ADC1->DR 의 주소 값을 넣는다.

- Circular Buffer 로 모드를 설정, 우선순위 설정, 주변기기 및 메모리 참조 ENABLE 을 해준다.

- 데이터 사이즈는 모두 1 Word Size 로 설정한다.

- 추가적으로 DMA_DeInit, DMA_Init, DMA_Cmd 함수를 통해 DMA 채널, Init 구조체를 Setting 하고 Enable 한다.

```
typedef struct
{
  uint32_t DMA_PeripheralBaseAddr; /*!< Specifies the peripheral base address for DMAy Channelx. */

  uint32_t DMA_MemoryBaseAddr;     /*!< Specifies the memory base address for DMAy Channelx. */

  uint32_t DMA_DIR;                /*!< Specifies if the peripheral is the source or destination.
                                        This parameter can be a value of @ref DMA_data_transfer_direction */

  uint32_t DMA_BufferSize;         /*!< Specifies the buffer size, in data unit, of the specified Channel.
                                        The data unit is equal to the configuration set in DMA_PeripheralDataSize
                                        or DMA_MemoryDataSize members depending in the transfer direction. */

  uint32_t DMA_PeripheralInc;      /*!< Specifies whether the Peripheral address register is incremented or not.
                                        This parameter can be a value of @ref DMA_peripheral_incremented_mode */

  uint32_t DMA_MemoryInc;          /*!< Specifies whether the memory address register is incremented or not.
                                        This parameter can be a value of @ref DMA_memory_incremented_mode */

  uint32_t DMA_PeripheralDataSize; /*!< Specifies the Peripheral data width.
                                        This parameter can be a value of @ref DMA_peripheral_data_size */

  uint32_t DMA_MemoryDataSize;     /*!< Specifies the Memory data width.
                                        This parameter can be a value of @ref DMA_memory_data_size */

  uint32_t DMA_Mode;               /*!< Specifies the operation mode of the DMAy Channelx.
                                        This parameter can be a value of @ref DMA_circular_normal_mode.
                                        @note: The circular buffer mode cannot be used if the memory-to-memory
                                              data transfer is configured on the selected Channel */

  uint32_t DMA_Priority;           /*!< Specifies the software priority for the DMAy Channelx.
                                        This parameter can be a value of @ref DMA_priority_level */

  uint32_t DMA_M2M;                /*!< Specifies if the DMAy Channelx will be used in memory-to-memory transfer.
                                        This parameter can be a value of @ref DMA_memory_to_memory */
}DMA_InitTypeDef;
```

```
void DMA_DeInit(DMA_Channel_TypeDef* DMAy_Channelx)
{
  /* Check the parameters */
  //assert_param(IS_DMA_ALL_PERIPH(DMAy_Channelx));

  /* Disable the selected DMAy Channelx */
  DMAy_Channelx->CCR &= (uint16_t)(~DMA_CCR1_EN);

  /* Reset DMAy Channelx control register */
  DMAy_Channelx->CCR  = 0;

  /* Reset DMAy Channelx remaining bytes register */
  DMAy_Channelx->CNDTR = 0;

  /* Reset DMAy Channelx peripheral address register */
  DMAy_Channelx->CPAR  = 0;

  /* Reset DMAy Channelx memory address register */
  DMAy_Channelx->CMAR = 0;

  if (DMAy_Channelx == DMA1_Channel1)
  {
    /* Reset interrupt pending bits for DMA1 Channel1 */
    DMA1->IFCR |= DMA1_Channel1_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel2)
  {
    /* Reset interrupt pending bits for DMA1 Channel2 */
    DMA1->IFCR |= DMA1_Channel2_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel3)
  {
    /* Reset interrupt pending bits for DMA1 Channel3 */
    DMA1->IFCR |= DMA1_Channel3_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel4)
  {
    /* Reset interrupt pending bits for DMA1 Channel4 */
    DMA1->IFCR |= DMA1_Channel4_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel5)
  {
    /* Reset interrupt pending bits for DMA1 Channel5 */
    DMA1->IFCR |= DMA1_Channel5_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel6)
  {
    /* Reset interrupt pending bits for DMA1 Channel6 */
    DMA1->IFCR |= DMA1_Channel6_IT_Mask;
  }
  else if (DMAy_Channelx == DMA1_Channel7)
  {
    /* Reset interrupt pending bits for DMA1 Channel7 */
    DMA1->IFCR |= DMA1_Channel7_IT_Mask;
  }
  else if (DMAy_Channelx == DMA2_Channel1)
  {
    /* Reset interrupt pending bits for DMA2 Channel1 */
    DMA2->IFCR |= DMA2_Channel1_IT_Mask;
  }
  else if (DMAy_Channelx == DMA2_Channel2)
  {
    /* Reset interrupt pending bits for DMA2 Channel2 */
    DMA2->IFCR |= DMA2_Channel2_IT_Mask;
  }
  else if (DMAy_Channelx == DMA2_Channel3)
  {
    /* Reset interrupt pending bits for DMA2 Channel3 */
    DMA2->IFCR |= DMA2_Channel3_IT_Mask;
  }
  else if (DMAy_Channelx == DMA2_Channel4)
  {
    /* Reset interrupt pending bits for DMA2 Channel4 */
    DMA2->IFCR |= DMA2_Channel4_IT_Mask;
  }
  else
  {
    if (DMAy_Channelx == DMA2_Channel5)
    {
      /* Reset interrupt pending bits for DMA2 Channel5 */
      DMA2->IFCR |= DMA2_Channel5_IT_Mask;
    }
  }
}
```

```
void DMA_Cmd(DMA_Channel_TypeDef* DMAy_Channelx, FunctionalState NewState)
{
  /* Check the parameters */
  //assert_param(IS_DMA_ALL_PERIPH(DMAy_Channelx));
  //assert_param(IS_FUNCTIONAL_STATE(NewState));

  if (NewState != DISABLE)
  {
    /* Enable the selected DMAy Channelx */
    DMAy_Channelx->CCR |= DMA_CCR1_EN;
  }
  else
  {
    /* Disable the selected DMAy Channelx */
    DMAy_Channelx->CCR &= (uint16_t)(~DMA_CCR1_EN);
  }
}

/**
```

```
void DMA_Configure() {
        DMA_InitTypeDef DMA_InitStructure;

        DMA_DeInit(DMA1_Channel1);
        DMA_InitStructure.DMA_BufferSize =2;
        DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
        DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
        DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADC_V;
        DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
        DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
        DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
        DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
        DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Enable;
        DMA_InitStructure.DMA_Priority = DMA_Priority_High;

        DMA_Init(DMA1_Channel1, &DMA_InitStructure);
        DMA_Cmd(DMA1_Channel1, ENABLE);
}
```

4.5 main

- 원래는 ADC_GetConversionValue 함수를 이용해 조도 센서 값을 받아오지만 DMA 방식을 사용해 전역변수로 값을 연결해줬으므로 전역변수를 바로 사용하면 된다.
- 헤더파일에 〈stm32f10x_dma.h〉 가 추가된다.
- 조도 센서 값이 밝음/어두움 일 때 값을 각각 실행해서 알아본 후 특정 값을 정해 LED 점/소 등을 결정한다.

```
int main(void) {
        SystemInit();
        RCC_Configure();
        GPIO_Configure();
        ADC_Configure();
        DMA_Configure();

        LCD_Init();
        Touch_Configuration();
        Touch_Adjust();
        LCD_Clear(WHITE);
        while(1) {
                ADC_SoftwareStartConvCmd(ADC1, ENABLE);
                LCD_ShowNum(40, 130, temp, 5, BLACK, WHITE);

                if(temp > 2000)   GPIO_SetBits(GPIOD, GPIO_Pin_2);
                else              GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        }
}
```

05. 실험 결과

5.1 소스코드

위의 내용을 바탕으로 작성한 전체 소스코드는 아래와 같다.

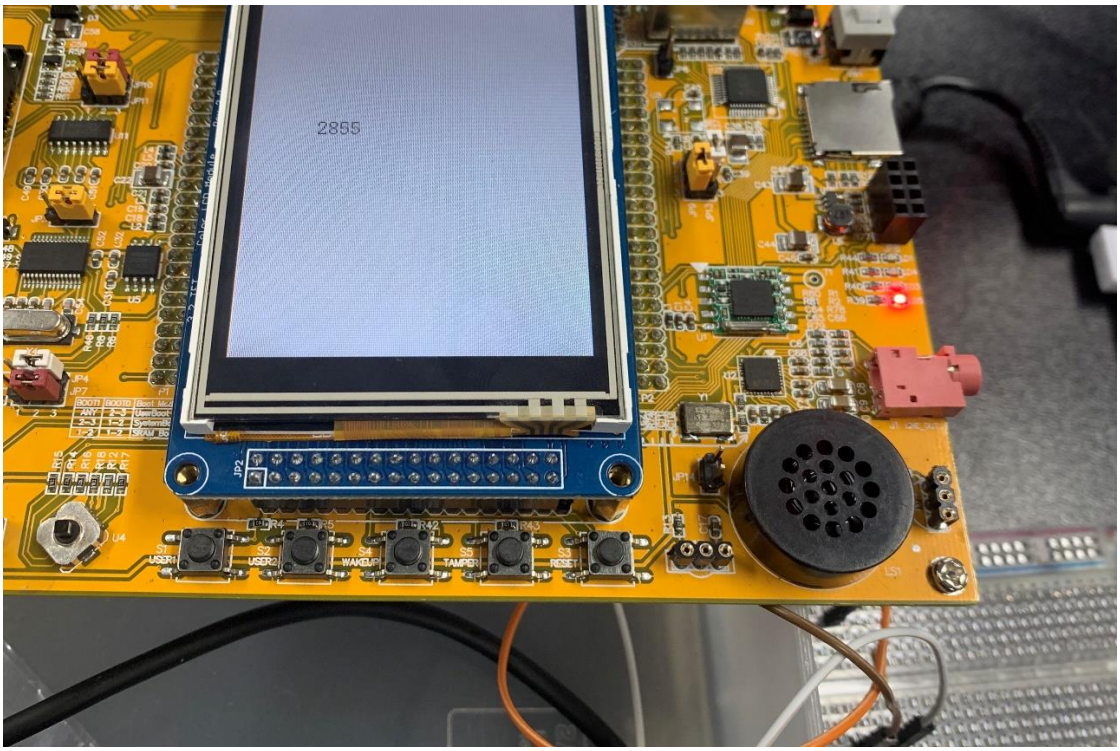
```
#include <misc.h>
#include <stm32f10x.h>
#include <stm32f10x_exti.h>
#include <stm32f10x_gpio.h>
#include <stm32f10x_rcc.h>
#include <stm32f10x_usart.h>
#include <stm32f10x_tim.h>
#include <stm32f10x_adc.h>
#include <stm32f10x_dma.h>
#include "lcd.h"
#include "touch.h"

uint16_t temp;
vu32 ADC_V;

void RCC_Configure() {
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOC |
RCC_APB2Periph_GPIOD | RCC_APB2Periph_GPIOE | RCC_APB2Periph_GPIOB | RCC_APB2Periph_ADC1,
ENABLE);
	RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
}

void GPIO_Configure() {
	GPIO_InitTypeDef GPIO_InitStructure;

	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2);
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init(GPIOD, &GPIO_InitStructure);
}

void ADC_Configure() {
	ADC_InitTypeDef ADC_InitStructure;
	ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
	ADC_InitStructure.ADC_ScanConvMode = ENABLE;
	ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
	ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
	ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
	ADC_InitStructure.ADC_NbrOfChannel = 1;

	ADC_Init(ADC1, &ADC_InitStructure);
	ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_239Cycles5);
	ADC_DMACmd(ADC1, ENABLE);
	ADC_Cmd(ADC1, ENABLE);

	ADC_ResetCalibration(ADC1);
	while(ADC_GetResetCalibrationStatus(ADC1));
	ADC_StartCalibration(ADC1);
	while(ADC_GetCalibrationStatus(ADC1));
}

void DMA_Configure() {
	DMA_InitTypeDef DMA_InitStructure;

	DMA_DeInit(DMA1_Channel1);
	DMA_InitStructure.DMA_BufferSize =2;
	DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
	DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
	DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)&ADC_V;
	DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Word;
	DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
```

```
        DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
        DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&ADC1->DR;
        DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Word;
        DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Enable;
        DMA_InitStructure.DMA_Priority = DMA_Priority_High;

        DMA_Init(DMA1_Channel1, &DMA_InitStructure);
        DMA_Cmd(DMA1_Channel1, ENABLE);
}

 int main(void) {
        SystemInit();
        RCC_Configure();
        GPIO_Configure();
        ADC_Configure();
        DMA_Configure();

        LCD_Init();
        Touch_Configuration();
        Touch_Adjust();
        LCD_Clear(WHITE);
        while(1) {
                ADC_SoftwareStartConvCmd(ADC1, ENABLE);
                LCD_ShowNum(40, 130, temp, 5, BLACK, WHITE);

                if(temp > 2000)   GPIO_SetBits(GPIOD, GPIO_Pin_2);
                else              GPIO_ResetBits(GPIOD, GPIO_Pin_2);
        }
}
```

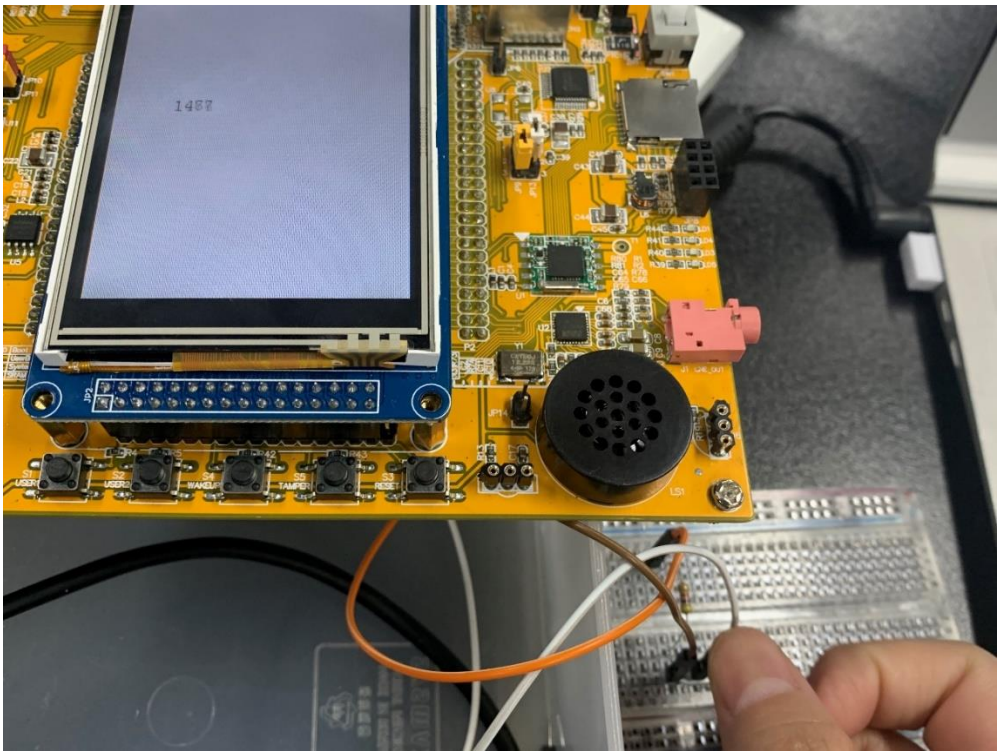
5.2 결과 확인 및 사진

소스 코드의 동작을 다음과 같이 확인했다.

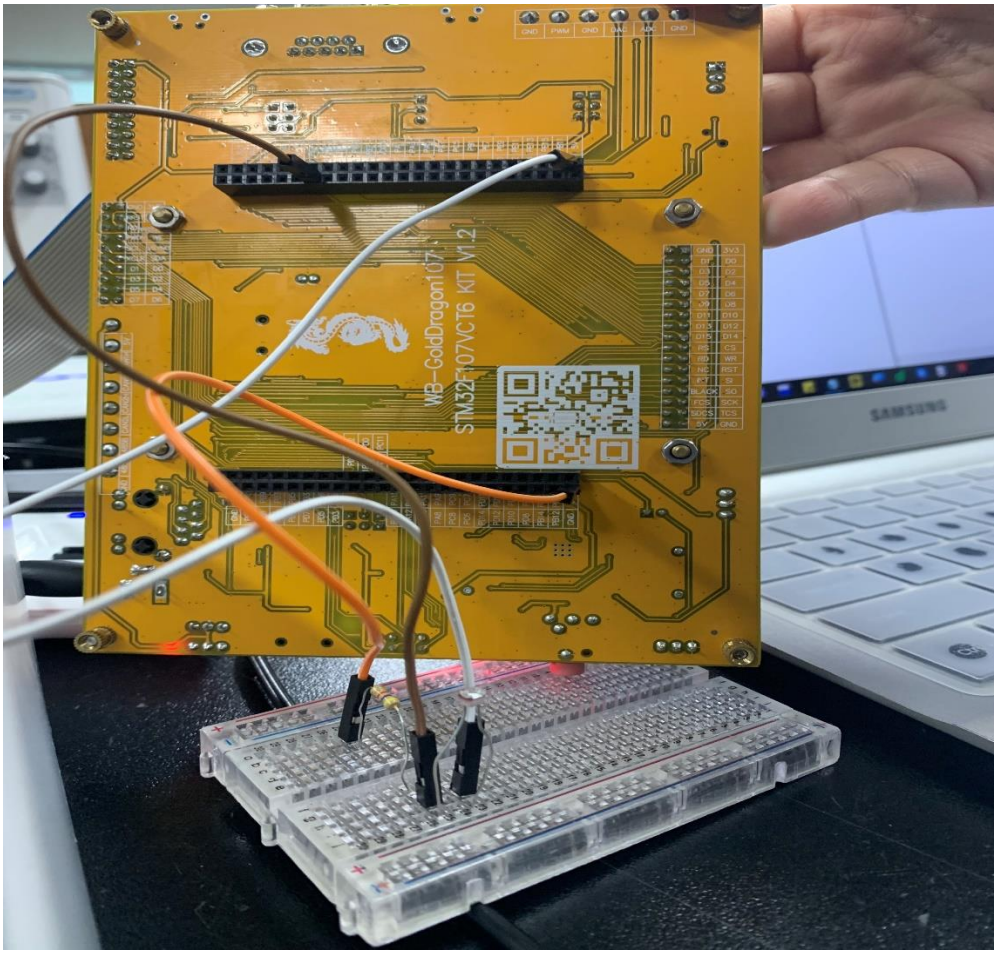
① 밝음 상태의 조도 센서 값 및 LED 점등



② 어두움 상태의 조도 센서 값 및 LED 소등



③ 보드 및 조도 센서 연결 방식



06. 결론

8 주차 실험에서 사용했던 조도 센서 및 TFT LCD 를 그대로 이용하여 조도 센서의 값을 읽고 그에 따라 LED 를 실행시키는 실험을 해보았다. 그러나 8 주차와의 차이점은 Interrupt 를 사용하는 것이 아닌 DMA(Direct Memory Access)를 사용하여 ADC Value 를 가져오는 것이었다. 다른 주차 실험들과 달리 DMA_Configure 함수를 추가로 만들어 DMA_InitStructure 을 이용하여 버퍼, 데이터 사이즈, BaseAddr 등을 설정해 DMA 를 설정해줄 수 있었다. BaseAddr 에 전역변수 주소 값을 대입해 조도 센서로 받는 ADC Value 를 전역 변수로 바로바로 사용할 수 있었다. 원래는 (줄여서)ADCGetValue 함수로 값을 받아와야 하지만 바로 전역 변수를 사용해 LCD 에 값 출력, 또는 If-Else 에 전역 변수를 사용해 LED 점/소등을 결정할 수 있었다. DMA 실험을 통해 특정 상황에서는 인터럽트보다 DMA 를 활용해 훨씬 강력하고 간편하게 소스코드를 짜고 실행시킬 수 있다는 것을 배웠다. 임베디드시스템 과목이나 컴퓨터 구조 과목에서 이론으로만 배웠을 때는 완전히 이해하기 어려웠는데, 직접 실험을 통해 소스코드를 작성하면서 DMA 를 활용해보니 생각보다 쉬웠고 더 잘 이해가 됐다. 이후에 텀 프로젝트에서 사용할 인체 감지 센서나 피에조 부저를 활용할 때 DMA 를 활용하여 더 쉽게 코드를 작성할 수 있을 것 같다.