

복합 예제

- 가변저항 값에 따라 Piezo 옥타브 제어하기
- 스위치 값에 따라 가로등, 선풍기 제어하기
- 온습도 값에 따라 LED 및 Text LCD 제어하기
- 스탑워치 제어하기



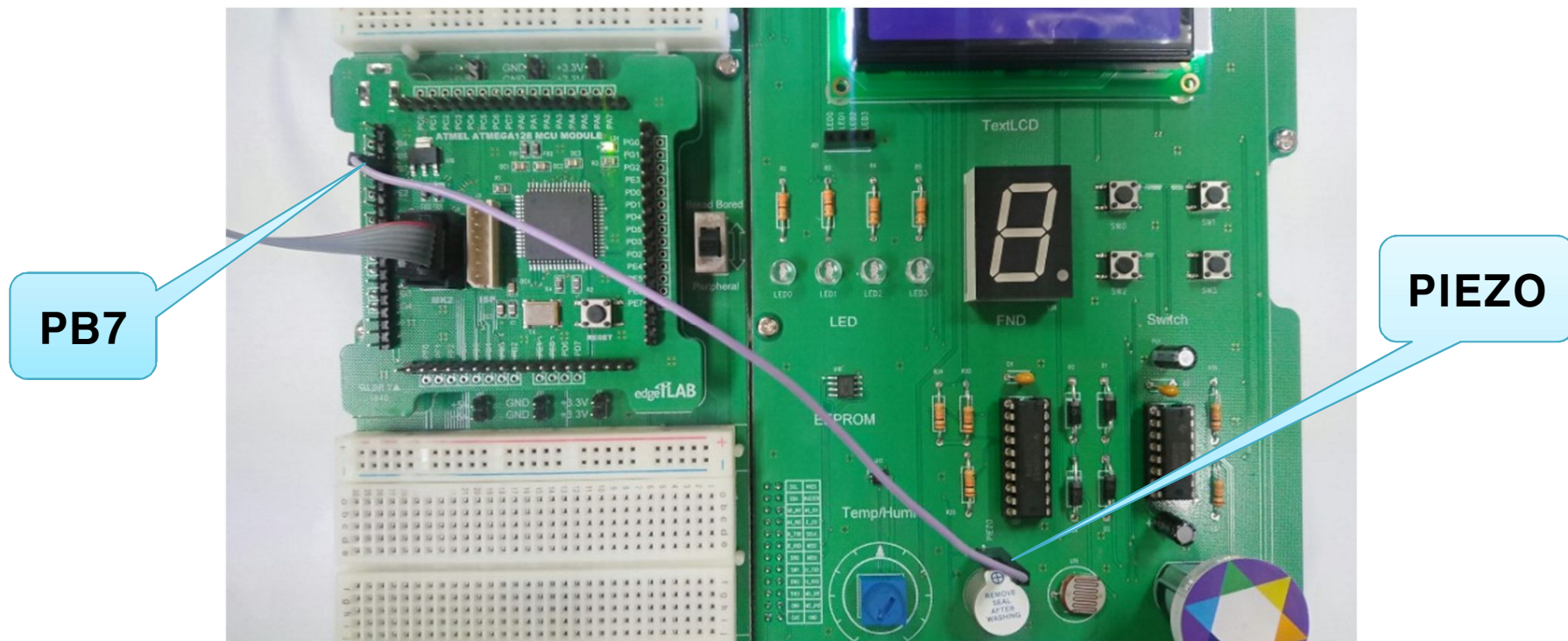
엣지아이랩

복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 실습 개요
 - 가변저항 값을 4단계로 나눠 단계별로 FND에 출력
 - 각 단계별로 PIEZO의 옥타브를 변경하여 소리 출력
- 실습 목표
 - FND LED 동작 원리 이해
 - PWM 동작 원리 이해

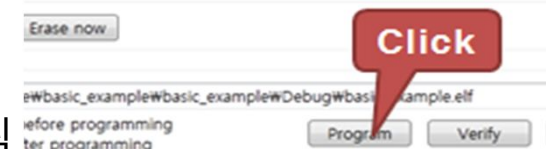
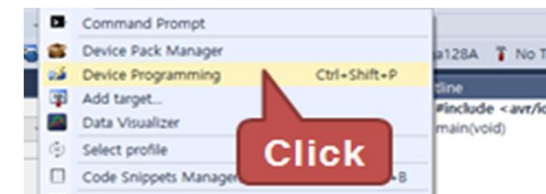
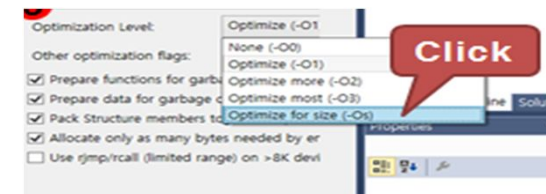
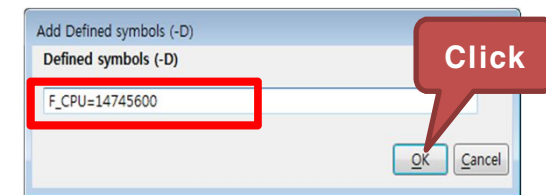
복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 실습 준비
 - 사용보드를 다음 그림과 같이 연결
 - Edge-MCU보드의 PB7 → Edge-Peri 보드의 PIEZO(PIEZO상단우측)



복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 14_VrFndPiezo_Application, Location : D:\#AVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 구동 프로그램
 - main.c

```
#include <avr/io.h>           //AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>        // delay 함수사용을 위한 헤더파일

//피아노 도음(5, 6, 7, 8옥타브) 해당하는 PWM 주파수
unsigned int Do[4] = {523, 1046, 2093, 4186};
// 7-Segment에 표시할 글자의 입력 데이터를 저장
unsigned char FND_DATA_TBL[] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x67,0x77,0x7C,0x39,
0x5E,0x79,0x71,0x08,0x80};

unsigned char FND_Count = 1, FND_Count_pre = 0;
```

복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    unsigned int AdData = 0;      //10bit ADC값 저장 변수

    DDRB = 0x80; // PWM(OCR1C)을 출력 포트로 설정 한다.
    // PB7에 PIEZO 연결
    DDRA = 0xFF; // 포트A 를 출력포트로 설정한다.

    TCCR1A |= 0x0A;      // COM1C(1:0) = "10",   OC1C 핀 사 용 ,
    WGM3(1:0) = "10"
    TCCR1B |= 0x19;      // WGM3(3:2) = "11"   , CS3(2:0) = "001" 1
    분주 사용
    TCCR1C = 0x00; // WGM3(3:0) = "1110", Fast PWM, 모드 14
    TCNT1 = 0x0000; // 타이머1 카운터 초기화
```

복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 구동 프로그램
 - main.c

```
ADMUX = 0x41;          //ADC1선택 //single mode, 1번 채널, 3.3V 외
부 기준전압(AREF)
ADCSRA = 0x87; // 10000111 //ADC 허가, 128분주

while (1)
{
    // ADC0을 통한 ADC값 읽어오기
    ADCSRA |= 0x40; //ADSC AD 개시(Start)
    while((ADCSRA & 0x10) == 0x00); //ADIF AD 다 될 때까지 기다
림.

    AdData = ADC; //전압이 디지털로 변환된 값 읽어오기.
    if(AdData > 799) // 최대값을 800으로 설정
        AdData = 799;
```

복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

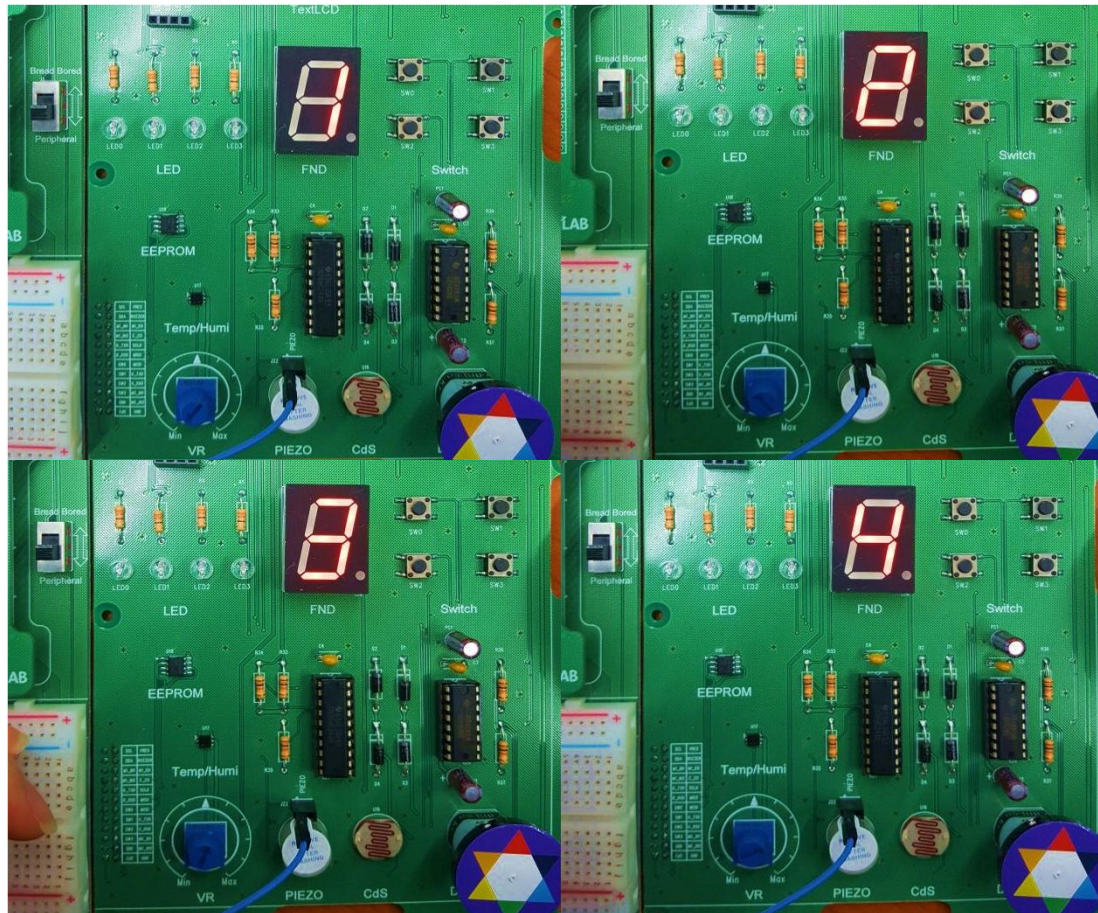
- 구동 프로그램
 - main.c

```
200      FND_Count = AdData / 200 + 1;    // 799를 4등분한 값은 약
      // 799 / 200 = 3.99 는 -> 3임
      // ADC / 200 은 0 ~ 3의 값

      if(FND_Count_pre != FND_Count)    // 이전 값과 다를 경우 실행
      {
          FND_Count_pre = FND_Count;
          ICR1 = 14745600/Do[FND_Count-1];    // 단계에 맞는 음
          향을 연주한다
          OCR1C = ICR1/2;                    // 50% 듀티비
          PORTA = FND_DATA_TBL[FND_Count];    // FND 현재 단계를
          출력한다.
      }
      _delay_ms(200);
  }
}
```


복합 : 가변저항 값에 따라 Piezo 옥타브 제어하기

- 실행 결과
 - Vr의 변화에 따라 음계와 FND의 숫자가 변함

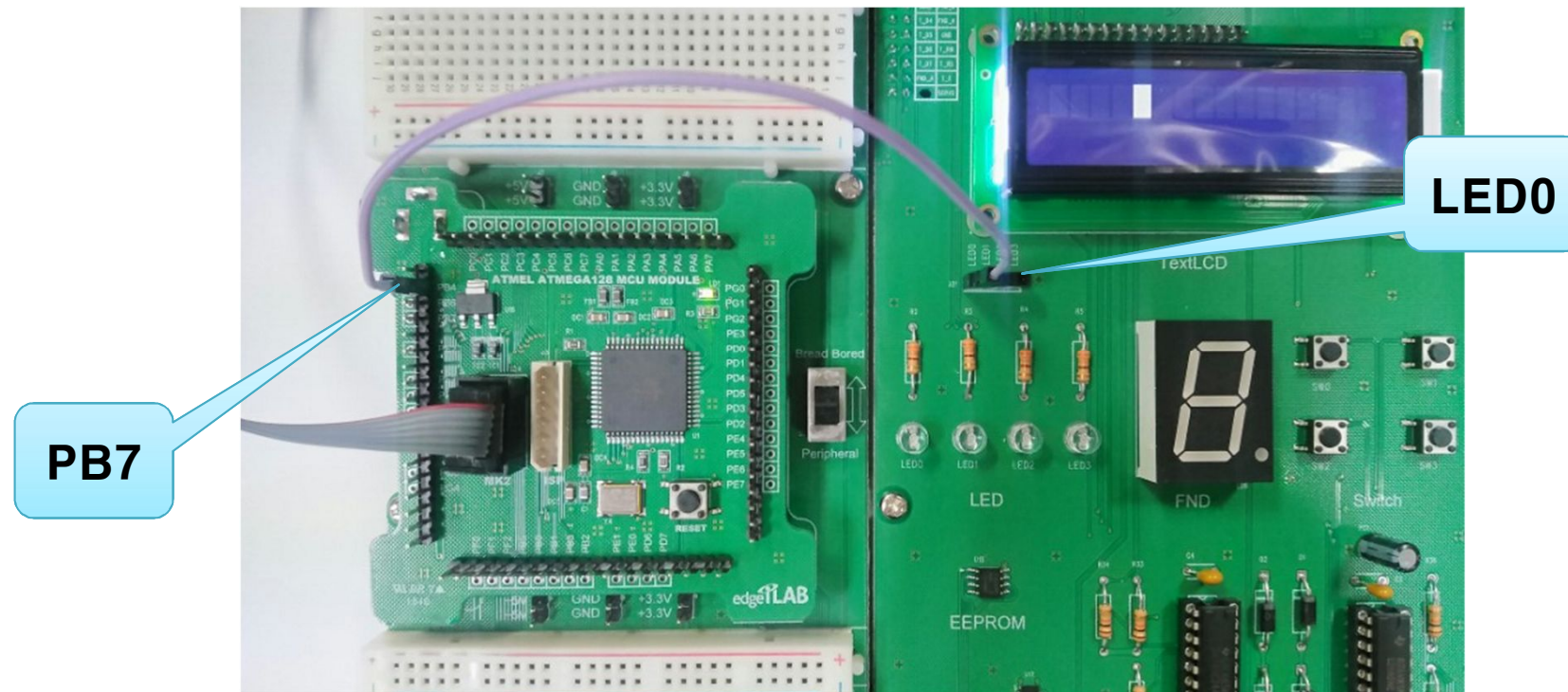


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 실습 개요
 - 스위치로 광센서 값에 따라 타이머 PWM 기능으로 LED 밝기조절 기능 ON/OFF
 - 스위치로 온습도 값에 따라 타이머 PWM 기능으로 DC Motor 속도 조절 기능 ON/OFF
 - 센서 등의 정보를 EEPROM에 저장하고, 해당 스위치를 누르면 저장된 정보를 UART로 전송
- 실습 목표
 - 인터럽트의 이해
 - PWM 출력의 이해
 - EEPROM의 이해
 - UART 통신의 이해

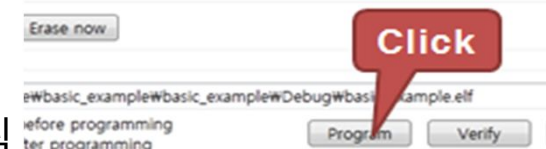
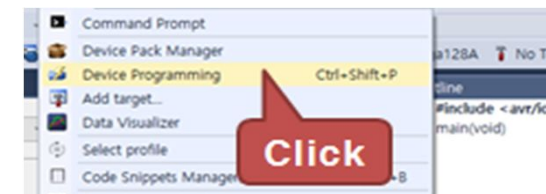
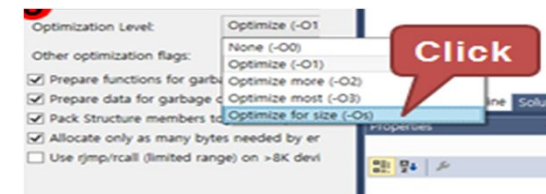
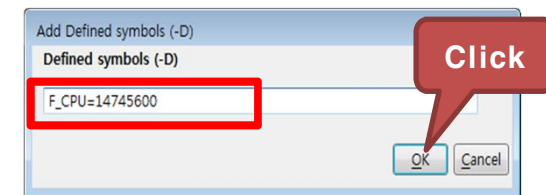
복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 실습 준비
 - 다음 그림과 같이 연결
 - Edge-MCU보드의 **PB7** → Edge-Peri 보드의 **LED0(LED 상단)**



복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 14_LedDcmotor_Application, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



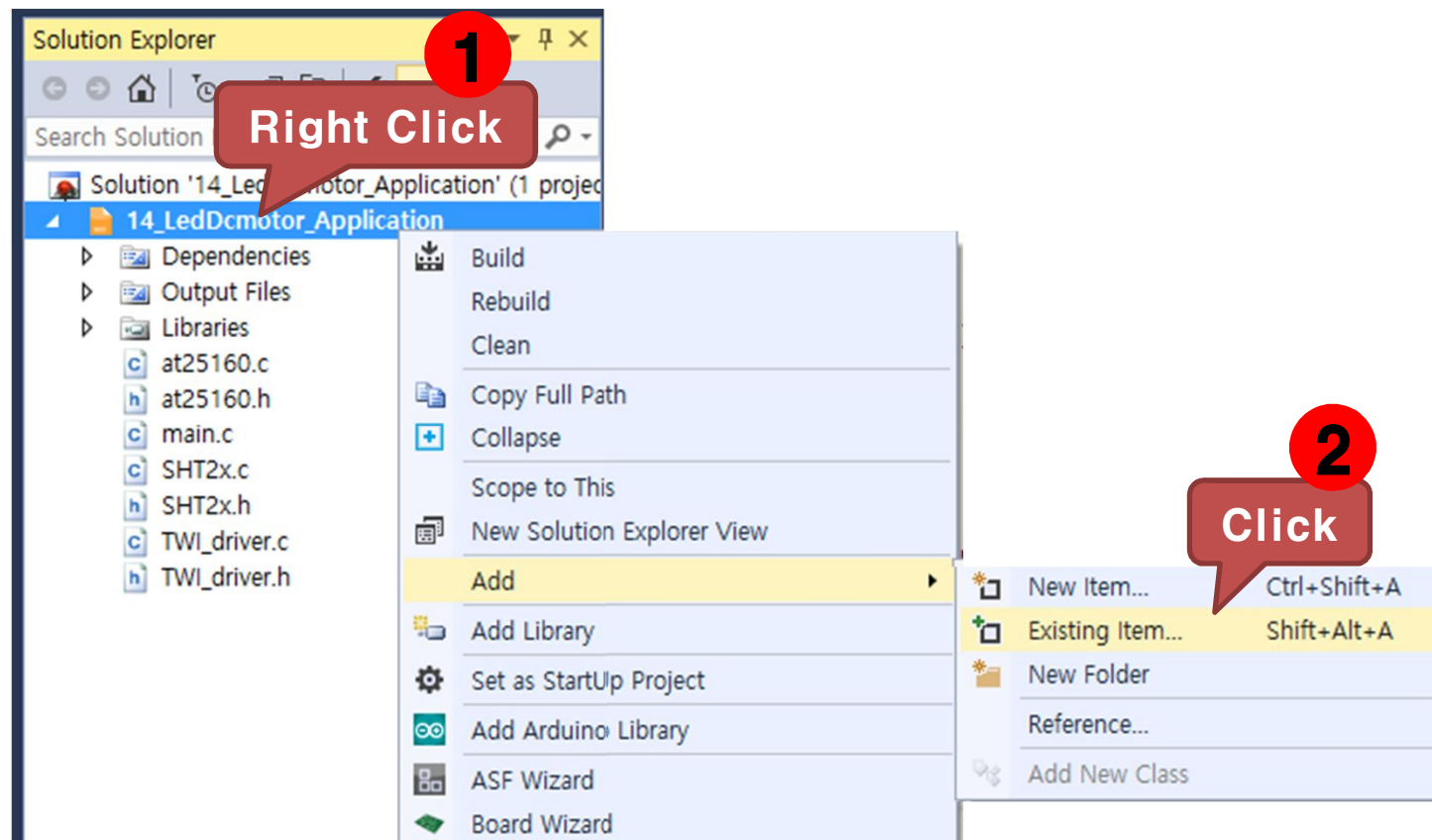
복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - SHT2x.c, SHT2x.h, TWI_driver.c, TWI_driver.h 파일은 "AVR_Example\library\sht20" 폴더에 존재
 - at25160.c, at25160.h 파일은 "AVR_Example\library\at25" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사



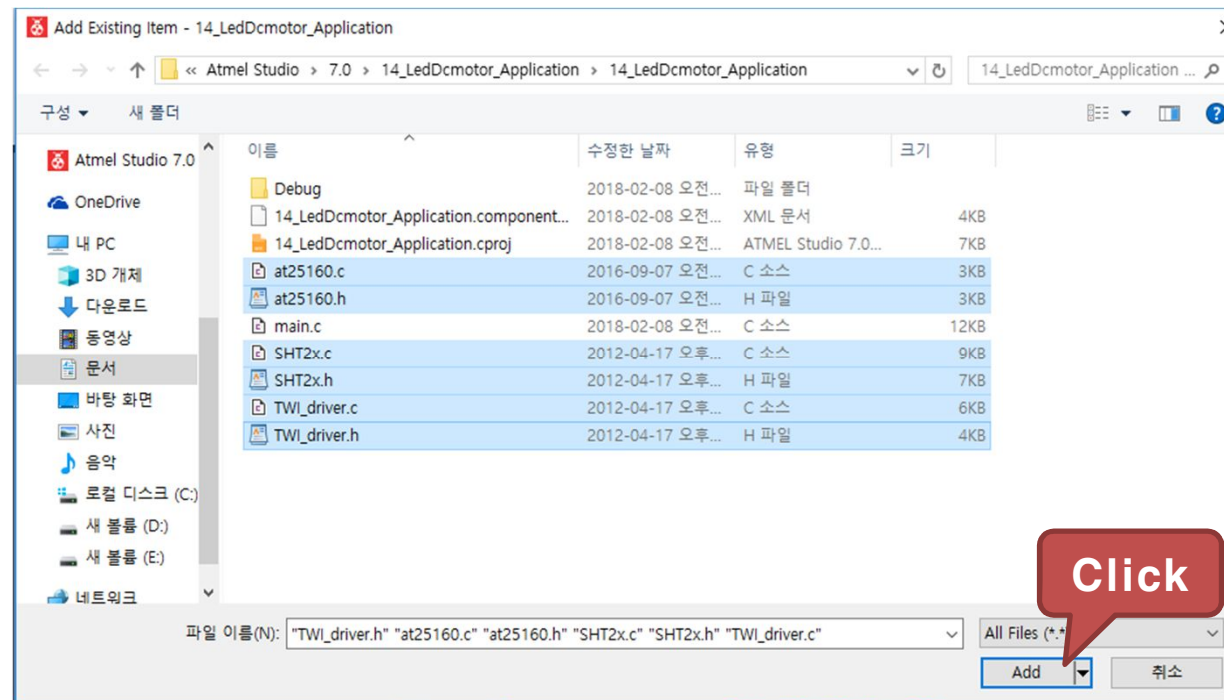
복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 여섯개의 파일을 추가
 - Ctrl 키를 누르고 파일 여섯개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
#include <avr/io.h>           //AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>      // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>         //delay 함수사용을 위한 헤더파일
#include "at25160.h"           //AT25160B 라이브러리 사용을 위한 헤더 파일
#include "TWI_driver.h"        //TWI 사용을 위한 헤더 파일
#include "SHT2x.h"             //SHT2x 라이브러리 사용을 위한 헤더 파일

#define M1_Forword 0x10
#define M1_Reverse 0x20
#define M1_Enable 0x20

unsigned int temperatureC, humidityRH; //온도, 습도 값 측정에 사용
되는 변수
unsigned int DC_PWM_Value = 0, LED_PWM_Value = 0; // PWM 제어 값
volatile unsigned int timer0Cnt=0;
volatile unsigned char Read_flag = 1, TX_flag = 0;
volatile unsigned char Light_Auto_flag = 0, FAN_Auto_flag = 0;
```


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
// EEPROM에 읽고 쓰기 위한 데이터
typedef union {
    unsigned int data16;
    unsigned char data8[2];
}DataDB;

// EEPROM 읽기 버퍼
DataDB Read_Data;
// EEPROM 쓰기 버퍼
DataDB Write_Data;

void putch(unsigned char data)
{
    while((UCSR0A & 0x20) == 0); // 전송준비가 될 때까지 대기
    UDR0 = data;                // 데이터를 UDR0에 쓰면 전송된다
    UCSR0A |= 0x20;
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
// 문자열을 출력하는 함수
void putch_Str(char *str)
{
    unsigned char i=0;
    while(str[i]!='\0')
        putch(str[i++]);    //문자열을 출력
}

int main(void)
{
    unsigned int tmp, i;
    unsigned long ADC_Data;
    unsigned char error = 0;    //에러를 저장하는 변수

    nt16 sRH;    //습도의 raw 데이터를 저장하는 변수
    nt16 sT;    //온도의 raw 데이터를 저장하는 변수
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    SPI_Init();

    Init_TWI();           //TWI를 초기화 한다
    SHT2x_Init();         //SHT 센서를 초기화 한다

    DDRD |= 0x30;         // MOTOR1_IN1, MOTOR1_IN2를 출력 포트로 설정 한다.
    DDRB |= 0xA0;         // MOTOR1_EN 를 출력 포트로 설정 한다.
    // PB7를 LED0에 연결
    DDRE |= 0x02; // Rx(입력 0), Tx(출력, 1), SW0 ~ 3 입력
    DDRF = 0x00;         // PF0을 입력 포트로 설정한다.

    PORTD |= M1_Forword; // DC Motor 정회전
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    UCSR0A = 0x00;
    UCSR0B = 0x18; // Rx, Tx enable
    UCSR0C = 0x06; // 비동기 방식, No Parity bit, 1 Stop bit

    UBRR0H = 0x00;
    UBRR0L = 0x07; // 115200 bps

    EICRB = 0xFF; // 인터럽트 4, 5, 6, 7을 상승엣지에서 동작하
    도록 설정한다.
    EIMSK = 0xF0; // 인터럽트 4, 5, 6, 7을 허용
    EIFR = 0xF0; // 인터럽트 4, 5, 6, 7 플래그를 클리어

    TCCR0 = 0x07;
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    TCNT0 = 112; // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
    TIMSK |= 0x01;
    TIFR |= 1 << TOV0;
    TCCR1A=0x8B;          // COMnA1=1  COMnA0=0  COMnB1=0  COMnB0=0
    COMnC1=1 COMnC0=0 WGMn1=1 WGMn0=1
    TCCR1B=0x01;          // ICNCn=0 ICESn=0 Blink=0 WGMn3=0 WGMn2=0
    CSn2=0 CSn1=0 CSn0=1
    TCCR1C=0x00; // PWM Phase Correct, 10-bit, Mode 3

    TCNT1=0; // 주기 1474560/2/1024 = 720hz
    OCR1A=0; // DC 모터 Off
    OCR1C=0; // LED OFF
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    ADMUX = 0x40;    // ADC0선택, 3.3V 외부 기준전압(AREF)
    ADCSRA = 0x87;   // 10000111 ADC 허가, single mode, 128분주

    sei();           // 전체 인터럽트를 허용
    while(1)
    {
        if(Read_flag)
        {
            // ADC0을 통한 CDS 센서 ADC값 읽어오기
            ADCSRA |= 0x40; //ADSC AD 개시(Start)
            while((ADCSRA & 0x10) == 0x00); //ADIF AD 다 될 때까지
            기다림.

            ADC_Data = ADC; //전압이 디지털로 변환된 값 읽어오기.
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    if(ADC_Data > 800) // 최대값을 800으로 설정
        ADC_Data = 800;
    // 다시 800을 1023이 되도록 변환
    ADC_Data = ADC_Data * 1023 / 800;
    // LED 밝기 제어 값 계산
    LED_PWM_Value = 1023 - ADC_Data;
    // CDS, LED 밝기, 자동 모드 값 EEPROM에 저장
    // Address      값
    // 0x100         CDS
    // 0x110         LED 밝기
    // 0x120         자동모드
    Write_Data.data16 = ADC_Data;
    at25160_Write_Arry(0x0100, Write_Data.data8,2); // EEPROM
    에 CDS 저장
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    Write_Data.data16 = LED_PWM_Value;
    at25160_Write_Arry(0x0110, Write_Data.data8,2); // EEPROM
에 밝기 제어 값 저장
    Write_Data.data16 = Light_Auto_flag;
    at25160_Write_Arry(0x0120, Write_Data.data8,2); // EEPROM
에 자동제어상태 저장
    error |= SHT2x_MeasureHM(HUMIDITY, &sRH); //습도를 측정한다.
    error |= SHT2x_MeasureHM(TEMP, &sT);      //온도를 측정한다.
    //온도 습도를 계산, 소숫점 첫째자리까지 출력하기 위해 10을 곱한
다.
    temperatureC = SHT2x_CalcTemperatureC(sT.u16)*10; // 온 도 를
계산한다.
    humidityRH   = SHT2x_CalcRH(sRH.u16)*10;          // 습 도
를 계산한다.
```


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if(error == SUCCESS)    //에러없이 정상 측정 되었으면
    {
        // DC 모터 PWM 값 계산
        // 온도 26도 이하 0단계, 29도 이하 1단계, 32도 이하 2단계, 35
        // 38도 이하 4단계, 38도보다 크면 5단계
        // 습도 55% 미만 0단계, 65% 미만 1단계, 75% 미만 2단계, 85%
        // 95% 미만 4단계, 95% 이상 5단계
        // 온도 및 습도 단계의 합으로 DC 모터의 PWM 계산 0 ~ 10단계로
        // 설정
        tmp = 0;
        for(i=0; i<5; i++)
        {
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if(temperatureC > (i * 30 + 260))
        tmp++;
    else
        break;
}
for(i=0; i<5; i++)
{
    if(humidityRH >= (i * 100 + 550))
        tmp++;
    else
        break;
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    // DC 모터 속도 제어 값 계산
    if(tmp == 0)
        DC_PWM_Value = 0;
    else
        DC_PWM_Value = tmp * 60 + 423;           // 최소 483으로 모
    터가 회전하도록 함.
    // 온도, 습도, DC 속도, 자동 모드 값 EEPROM에 저장
    // Address      값
    // 0x200        온도
    // 0x210        습도
    // 0x220        DC 속도
    // 0x230        자동모드
    Write_Data.data16 = temperatureC;
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    at25160_Write_Arry(0x0200, Write_Data.data8,2); // EEPROM
에 온도 저장
    Write_Data.data16 = humidityRH;
    at25160_Write_Arry(0x0210, Write_Data.data8,2); // EEPROM
에 습도 저장
    Write_Data.data16 = DC_PWM_Value;
    at25160_Write_Arry(0x0220, Write_Data.data8,2); // EEPROM
에 속도 제어 값 저장
    Write_Data.data16 = FAN_Auto_flag;
    at25160_Write_Arry(0x0230, Write_Data.data8,2); // EEPROM
에 자동제어상태 저장
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if(Light_Auto_flag) // LED 자동 밝기 제어
    {
        OCR1C = LED_PWM_Value;
    }
    if(FAN_Auto_flag)    // DC 모터 자동 속도 제어
    {
        OCR1A = DC_PWM_Value;
    }
    Read_flag = 0;
}
if(TX_flag)
{
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if(TX_flag & 0x01)
    {
        at25160_Read_Arry(0x0100, Read_Data.data8,2); // EEPROM
에서 CDS 읽음
        ADC_Data = Read_Data.data16;
        at25160_Read_Arry(0x0110, Read_Data.data8,2); // EEPROM
에서 밝기 제어 값 읽음
        tmp = Read_Data.data16;
        at25160_Read_Arry(0x0120, Read_Data.data8,2); // EEPROM
에서 자동제어상태 읽음
        putch_Str("\n\r CDS : ");
        // CDS 값 출력
    }
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if(ADC_Data/1000)
    {
        putchar(ADC_Data/1000 + '0');
        putchar((ADC_Data/100)%10 + '0');
        putchar((ADC_Data/10)%10 + '0');
    }
    else
    {
        if((ADC_Data/100)%10)
        {
            putchar((ADC_Data/100)%10 + '0');
        }
    }
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    putchar((ADC_Data/10)%10 + '0');
}
else
{
    if((ADC_Data/10)%10)
    {
        putchar((ADC_Data/10)%10 + '0');
    }
}
```


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
}
putch(ADC_Data%10 + '0');
putch_Str(", Light Value : ");

// 밝기 제어 값 출력
if(tmp/1000)
{
    putch(tmp/1000 + '0');
    putch((tmp/100)%10 + '0');
    putch((tmp/10)%10 + '0');
}
else
{
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if((tmp/100)%10)
    {
        putchar((tmp/100)%10 + '0');
        putchar((tmp/10)%10 + '0');
    }
    else
    {
        if((tmp/10)%10)
        {
            putchar((tmp/10)%10 + '0');
        }
    }
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    putch(tmp%10 + '0');
    // 자동 모드 상태 출력
    putch_Str(", Auto Mode : ");
    if(Read_Data.data16)
        putch_Str("ON");
    else
        putch_Str("OFF");
    TX_flag &= 0x02; // 0번째 비트를 0으로 설정
}
if(TX_flag & 0x02)
{
    at25160_Read_Arry(0x0200, Read_Data.data8,2); // EEPROM
    에서 온도 읽음
    temperatureC = Read_Data.data16;
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    at25160_Read_Arry(0x0210, Read_Data.data8,2); // EEPROM
에서 습도 읽음
    humidityRH = Read_Data.data16;
    at25160_Read_Arry(0x0220, Read_Data.data8,2); // EEPROM
에서 속도 제어 값 읽음
    tmp = Read_Data.data16;
    at25160_Read_Arry(0x0230, Read_Data.data8,2); // EEPROM
에서 자동제어상태 읽음

    putch_Str("\n\r Temp: ");
    // 온도 출력
    // 100의 자리 값 확인
    if(temperatureC/100) // 100의 자리가 0이 아닐 경우
    {
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    // 100의 자리 출력
    putchar(temperatureC/100 + '0');
}
putchar((temperatureC/10)%10 + '0');
putchar('.');
putchar(temperatureC%10 + '0');
putchar_Str("[C], Humi : ");
// 습도 출력
// 100의 자리 값 확인
if(humidityRH/100) // 100의 자리가 0이 아닐 경우
{
    // 100의 자리 출력
    putchar(humidityRH/100 + '0');
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    putchar((humidityRH/10)%10 + '0');
    putchar('.');
    putchar(humidityRH%10 + '0');
    putchar_Str("[%], Speed Value : ");

    // 속도 제어 값 출력
    if(tmp/1000)
    {
        putchar(tmp/1000 + '0');
        putchar((tmp/100)%10 + '0');
        putchar((tmp/10)%10 + '0');
    }
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    else
    {
        if((tmp/100)%10)
        {
            putchar((tmp/100)%10 + '0');
            putchar((tmp/10)%10 + '0');
        }
        else
        {
            if((tmp/10)%10)
            {
                putchar((tmp/10)%10 + '0');
            }
        }
    }
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
}
putch(tmp%10 + '0');
// 자동 모드 상태 출력
putch_Str(", Auto Mode : ");
if(Read_Data.data16)
    putch_Str("ON");
else
    putch_Str("OFF");
TX_flag &= 0x01; // 1번째 비트를 0으로 설정
}
}
}
```


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(TIMER0_OVF_vect)
{
    cli();
    TCNT0 = 112; // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
    timer0Cnt++; // timer0Cnt 변수를 1 증가 시킨다.

    if(timer0Cnt == 50)
    {
        // 10ms * 25 = 250ms // 250ms를 얻기 위한 카운트 횟수
        Read_flag = 1; // 온도와 습도 그리고 CDS 측정하고 그 값을
        EEPROM에 저장하도록 1로 설정
        timer0Cnt = 0;
    }
    sei();
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT4_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지

    if(Light_Auto_flag == 0)
    {
        Light_Auto_flag = 1; // 밝기에 따라 LED 자동제어
    }
    else
    {
        Light_Auto_flag = 0; // LED 자동 제어 정지
        OCR1C = 0;           // LED OFF
    }
    sei();              // 전체 인터럽트를 허용
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT5_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    if(FAN_Auto_flag == 0)
    {
        FAN_Auto_flag = 1; // 온도 및 습도에 따라 DC 모터 자동제어
    }
    else
    {
        FAN_Auto_flag = 0; // DC 모터 자동 제어 정지
        OCR1A = 0;         // DC 모터 OFF
    }
    sei();              // 전체 인터럽트를 허용
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

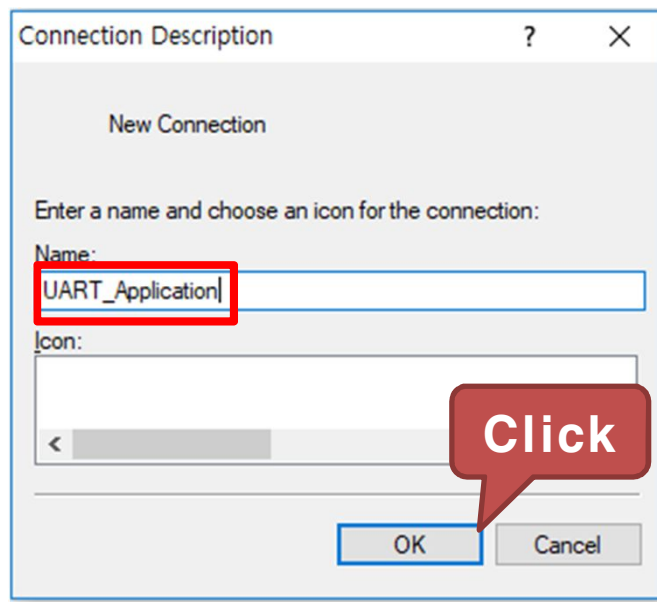
- 구동 프로그램
 - main.c

```
SIGNAL(INT6_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_flag |= 0x01;    // 광센서 및 LED 제어 PWM_Value 및 동작 상태를
                        // PC로 전송하도록 설정
    sei();              // 전체 인터럽트를 허용
}

SIGNAL(INT7_vect)      // 인터럽트 서비스 루틴
{
    cli();              // 전체 인터럽트를 금지
    TX_flag |= 0x02;    // 온도, 습도, PWM_Value 및 동작 상태를 PC로 전
                        // 송하도록 설정
    sei();              // 전체 인터럽트를 허용
}
```

복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 하이퍼터미널 실행
 - 연결 설명 창의 이름 란에 "UART_Application" 라고 확인을 클릭
 - 연결 대상 창의 연결에 사용할 모뎀 번호를 설정하고 확인을 클릭



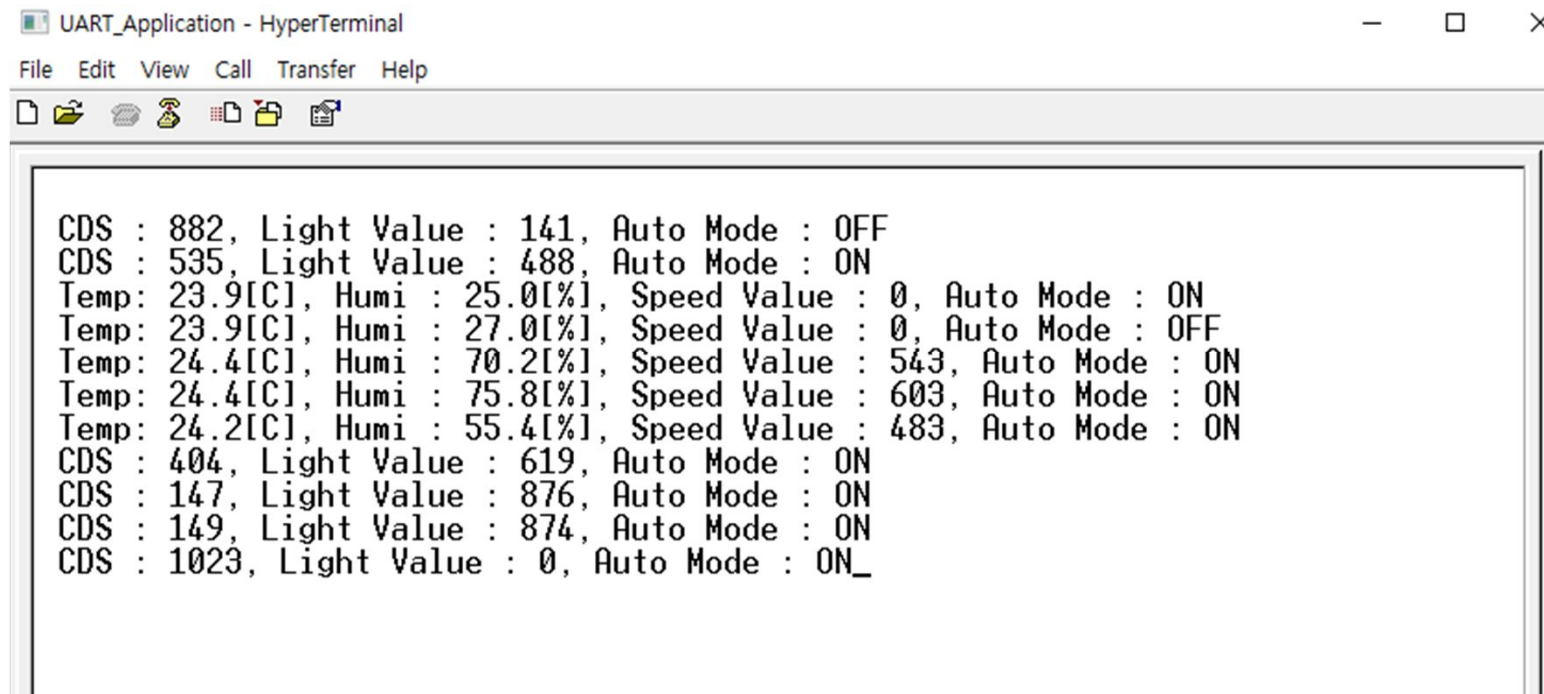
복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 하이퍼터미널 실행
 - 통신 설정을 그림과 같이하고, 확인을 클릭



복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 실행 결과
 - SW0을 누르면 가로등 Auto Mode 설정
 - SW1을 누르면 선풍기 Auto Mode 설정
 - SW2를 누르면 가로등과 관련된 내용이 하이퍼 터미널로 전송
 - SW3을 누르면 선풍기와 관련된 내용이 하이퍼 터미널로 전송

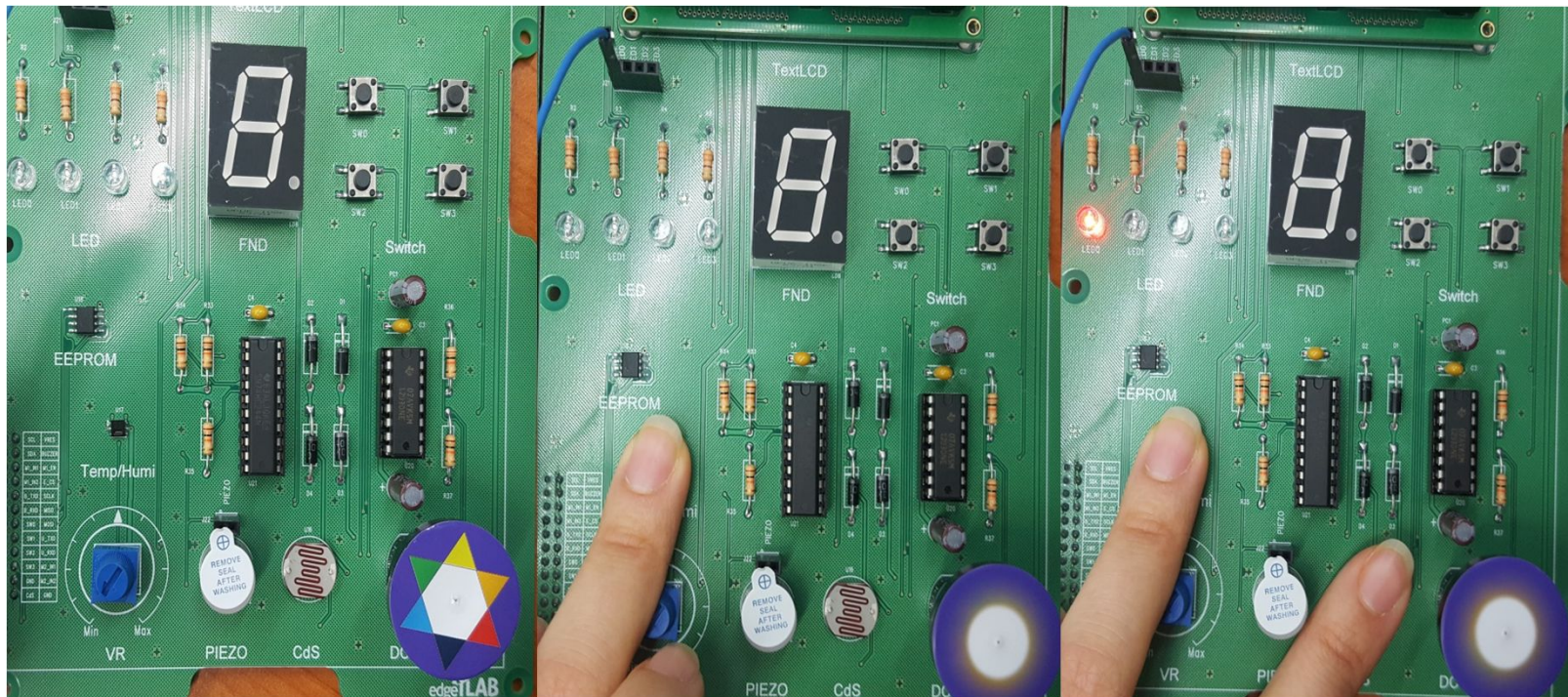


```
UART_Application - HyperTerminal
File Edit View Call Transfer Help

CDS : 882, Light Value : 141, Auto Mode : OFF
CDS : 535, Light Value : 488, Auto Mode : ON
Temp: 23.9[C], Humi : 25.0[%], Speed Value : 0, Auto Mode : ON
Temp: 23.9[C], Humi : 27.0[%], Speed Value : 0, Auto Mode : OFF
Temp: 24.4[C], Humi : 70.2[%], Speed Value : 543, Auto Mode : ON
Temp: 24.4[C], Humi : 75.8[%], Speed Value : 603, Auto Mode : ON
Temp: 24.2[C], Humi : 55.4[%], Speed Value : 483, Auto Mode : ON
CDS : 404, Light Value : 619, Auto Mode : ON
CDS : 147, Light Value : 876, Auto Mode : ON
CDS : 149, Light Value : 874, Auto Mode : ON
CDS : 1023, Light Value : 0, Auto Mode : ON_
```


복합 : 스위치 값에 따라 가로등, 선풍기 제어하기

- 실행 결과
 - 온/습도 및 조도 값에 따라 가로등과 선풍기 제어

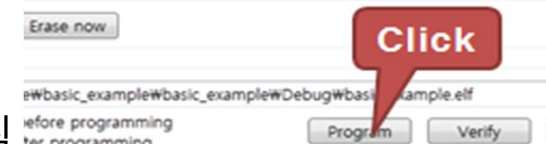
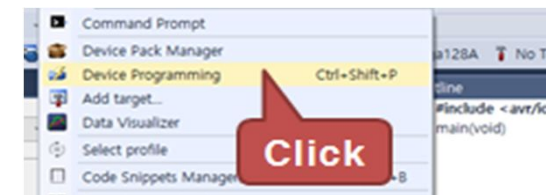
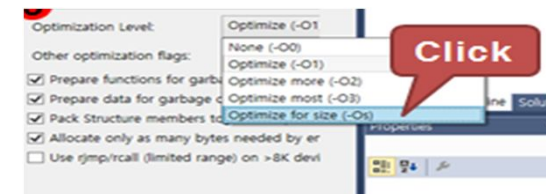


복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 실습 개요
 - 실시간으로 온습도를 측정하고, 값에 따라 LED0~3을 작동
 - 온습도 값을 Text LCD에 표시
- 실습 목표
 - I2C의 동작 원리 이해
 - SHT20 온습도 센서 제어 방법의 이해
 - Text LCD의 동작 원리 이해
 - Text LCD 제어 프로그램 방법의 이해

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 14_Sht20Led_Application, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - SHT2x.c, SHT2x.h, TWI_driver.c, TWI_driver.h 파일은 "AVR_Example\library\sht20" 폴더에 존재
 - lcd.c, lcd.h, lcdconf.h 파일은 "AVR_Example\library\lcd" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사

lcd Library 폴더

이름
lcd
lcd
lcdconf

sht20 Library 폴더

이름
SHT2x
SHT2x
TWI_driver
TWI_driver

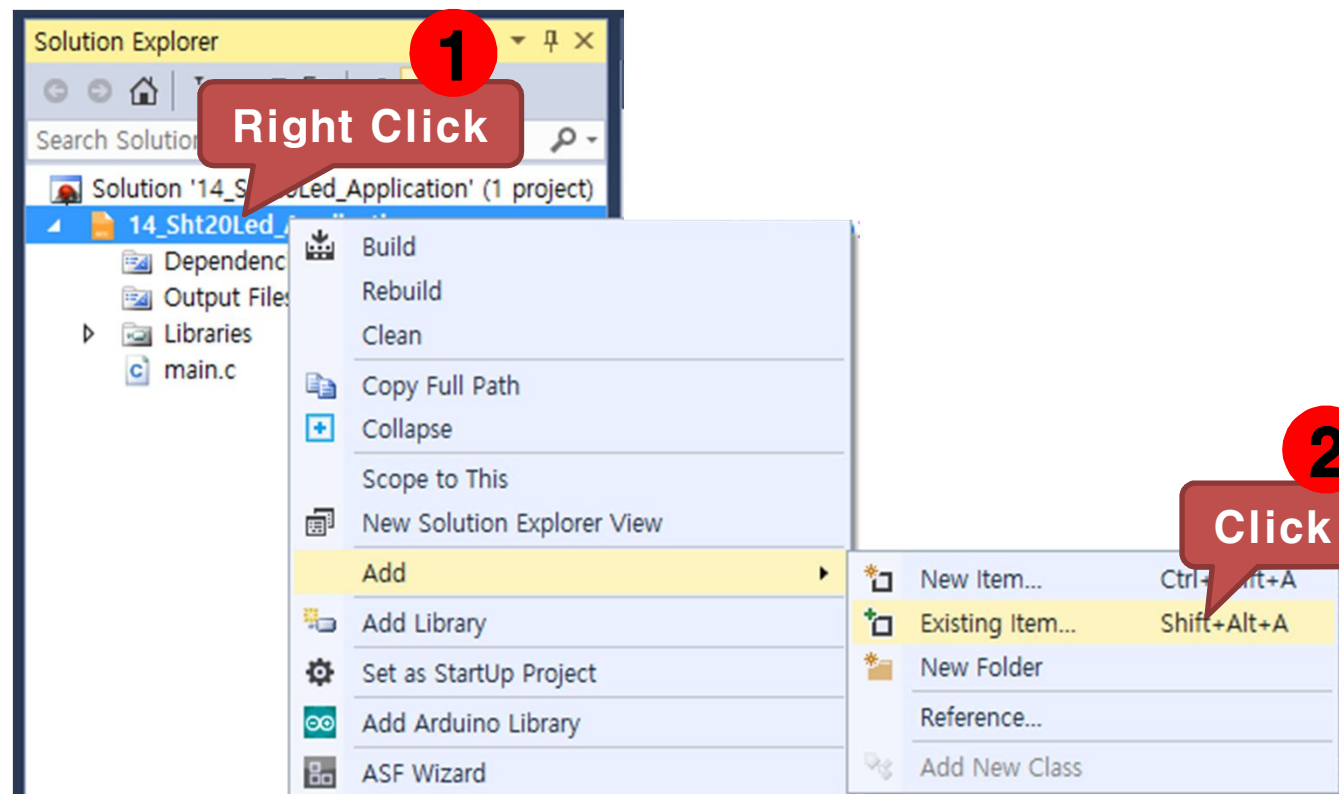
14_Sht20Led_Application
프로젝트 폴더

이름	수정한 날짜	유형
Debug	2018-02-08 오후...	파일 폴
14_Sht20Led_Application.componentinf...	2018-02-08 오후...	XML 문
14_Sht20Led_Application.cproj	2018-02-08 오후...	ATMEL
main.c	2018-02-08 오후...	C 소스

파일 복사

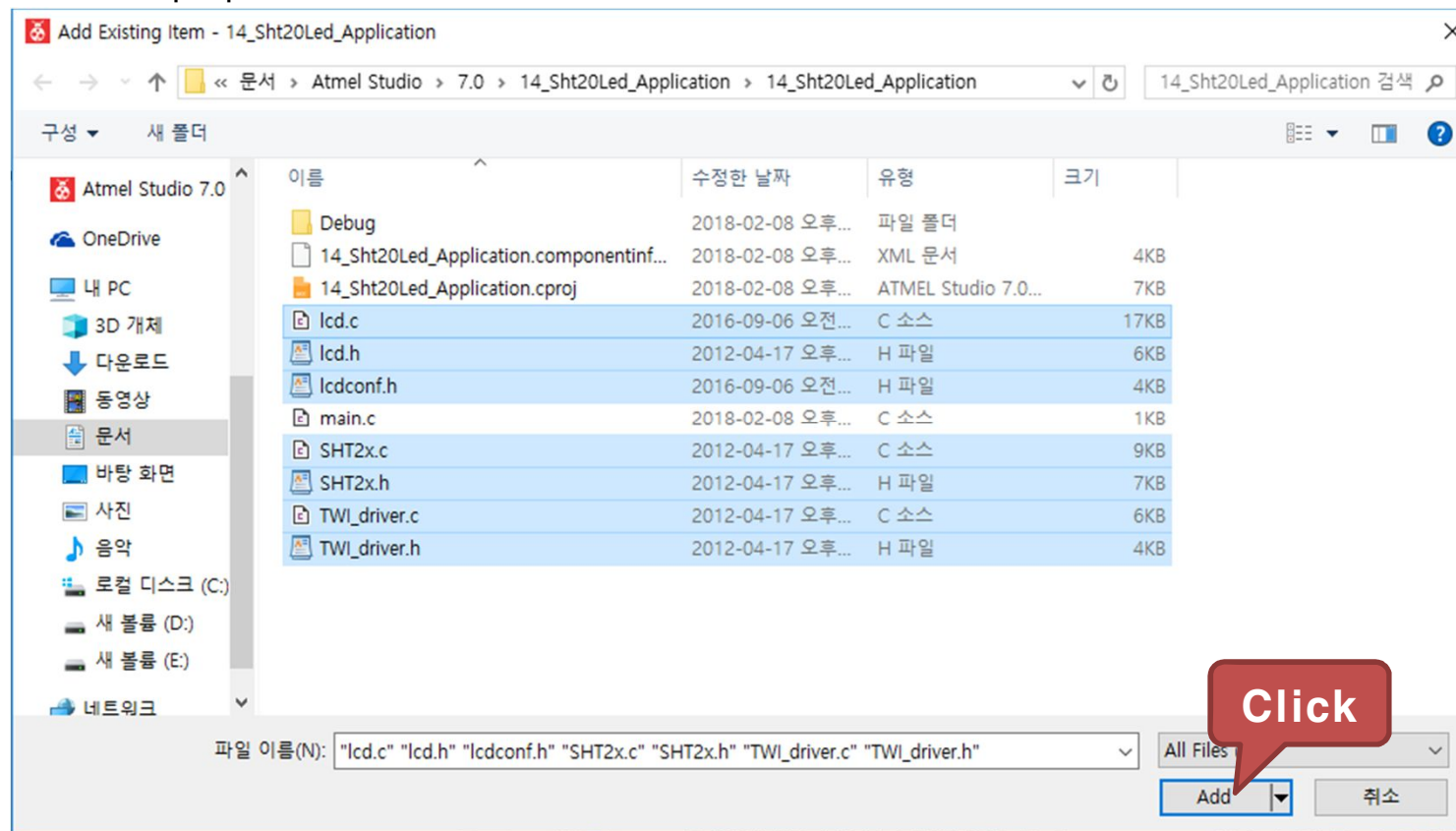
복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 일곱개의 파일을 추가
 - Ctrl 키를 누르고 파일 일곱개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
#include <avr/io.h>      //AVR 입출력에 대한 헤더 파일
#include <util/delay.h>  //delay 함수사용을 위한 헤더파일
#include "TWI_driver.h"  //TWI 사용을 위한 헤더 파일
#include "SHT2x.h"       //SHT2x 라이브러리 사용을 위한 헤더 파일
#include "lcd.h"         //Text LCD를 사용하기 위한 헤더 파일

void printf_2dot1(unsigned char sense, unsigned int sense_temp);
//온도 및 습도를 LCD에 출력하는 함수
unsigned int temperatureC, humidityRH;    //온도, 습도 값 측정에 사용
되는 변수
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    uint8_t error = 0;    //에러를 저장하는 변수

    nt16 sRH;              //습도의 raw 데이터를 저장하는 변수
    nt16 sT;              //온도의 raw 데이터를 저장하는 변수

    DDRC = 0x0F; // 포트C 를 출력포트로 설정한다.

    Init_TWI();            //TWI를 초기화 한다
    lcdInit();            //Text LCD를 초기화
    SHT2x_Init();         //SHT 센서를 초기화 한다
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    while(1)
    {
        error |= SHT2x_MeasureHM(HUMIDITY, &sRH); //습도를 측정한다.
        error |= SHT2x_MeasureHM(TEMP, &sT);      //온도를 측정한다.
        //온도 습도를 계산, 소숫점 첫째자리까지 출력하기 위해 10을 곱한
다.
        temperatureC = SHT2x_CalcTemperatureC(sT.u16)*10; //온도를
계산한다.
        humidityRH    = SHT2x_CalcRH(sRH.u16)*10;          //습도를
계산한다.
```


복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    if(error == SUCCESS) //에러없이 정상 측정 되었으면
    {
        lcdGotoXY(0,0);           //현재 커서위치를 첫번째줄
        첫번째칸으로 이동한다.
        printf_2dot1(TEMP,temperatureC); // 온도를 출력한다.
        lcdGotoXY(0,1);           //현재 커서위치를 두번째줄
        첫번째칸으로 이동한다.
        printf_2dot1(HUMIDITY,humidityRH); // 습도를
        출력한다.

        // 온도에 따른 LED0,1 동작
        if(temperatureC < 250)
            PORTC = (PINC & 0x0C) | 0x01; // 온도가 25도 미
            만이면 LED0를 켜다.(저온)
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    else
        PORTC = PINC & 0x0C; // 온도가 24 ~ 28도 이면
        LED0, 1을 끈다.(정상)
        // 습도에 따른 LED2,3 동작
        if(humidityRH < 400)
            PORTC = (PINC & 0x03) | 0x04; // 습도가 40% 미만
            이면 LED2를 켜다.(습도가 낮다)
            else if(humidityRH > 600)
                PORTC = (PINC & 0x03) | 0x08; // 습도가 60% 보다
                크면 LED3을 켜다.(습도가 높다)
            else
                PORTC = PINC & 0x03; // 습도가 40 ~ 60% 이면
                LED2, 3을 끈다.(정상)
}
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    else //에러가 있을 경우
    {
        lcdGotoXY(0,0);           //현재 커서위치를 첫번째줄
        첫번째칸으로 이동한다.
        lcdPrintData(" Temp: ---C",12); //온도를 ---로 출
        력한다.
        lcdGotoXY(0,1);           //현재 커서위치를 두번째줄
        첫번째칸으로 이동한다.
        lcdPrintData(" Humi: ---%",12); //습도를 ---로 출
        력한다.
    }
    _delay_ms(300);              //다음 측정을 위한 시간 지연(300ms)
}
}
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

```
void printf_2dot1(unsigned char sense, unsigned int sense_temp)
{
    uint8_t s100,s10;

    if(sense == TEMP) lcdPrintData(" Temp: ",7);           //온도
출력시 " Temp: " 출력
    else if(sense == HUMIDITY) lcdPrintData(" Humi: ",7);    //습도 출력
시 " Humi: " 출력

    s100 = sense_temp/100;           //100의 자리 추출

    if(s100> 0) lcdDataWrite(s100+'0'); //100의 자리 값이 있으면 100
의 자리 출력
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 구동 프로그램
 - main.c

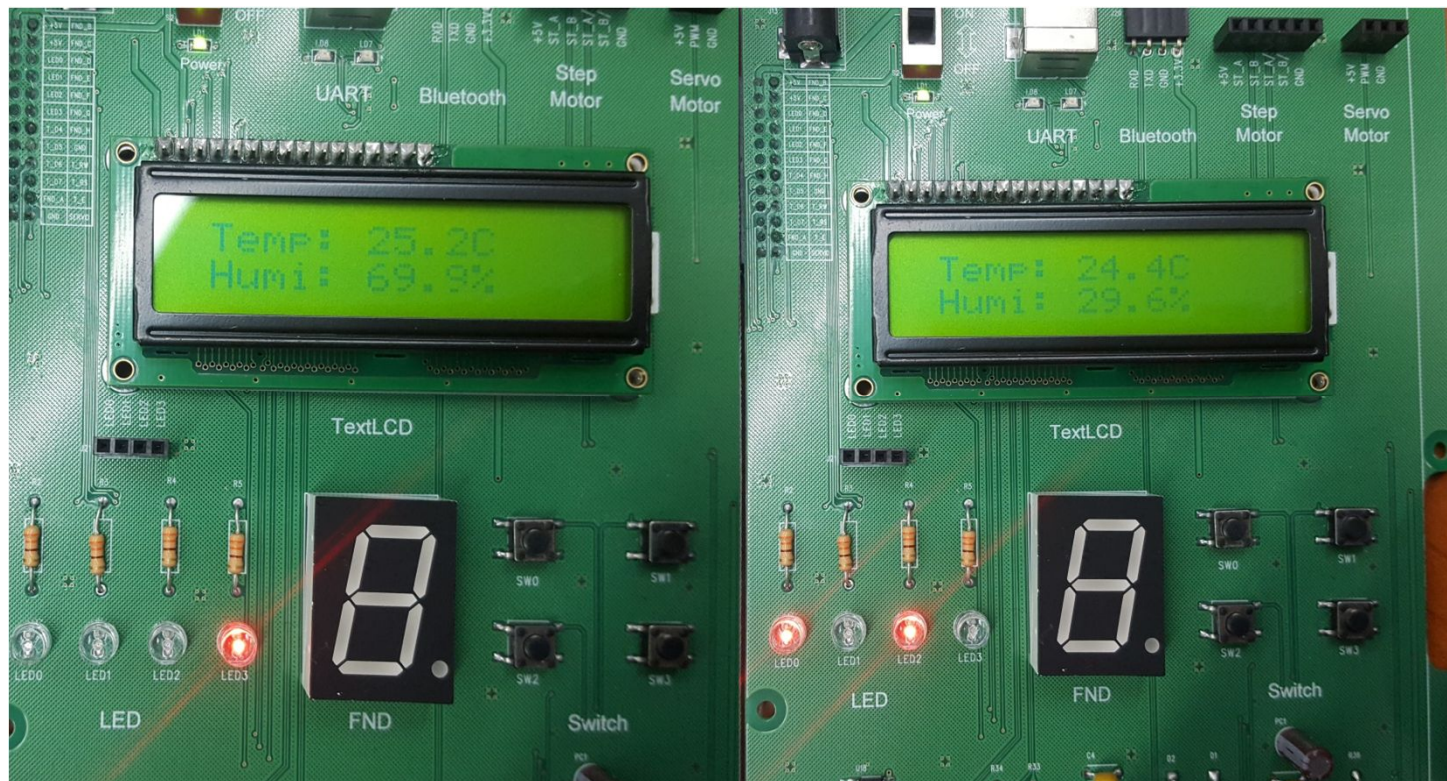
```
void printf_2dot1(unsigned char sense, unsigned int sense_temp)
{
    ...
    else lcdPrintData(" ",1);    //100의 자리 값이 없으면 빈칸 출력

    s10 = sense_temp%100;        //100의 자리를 제외한 나머지 추출
    lcdDataWrite((s10/10)+'0');   //10의 자리 추출하여 출력
    lcdPrintData(".",1);         //소숫점 출력
    lcdDataWrite((s10%10)+'0');   //1의 자리 추출하여 출력

    if(sense == TEMP) lcdDataWrite('C'); //온도 단위 출력
    else if(sense == HUMIDITY) lcdDataWrite('%'); //습도 단위 출력
}
```

복합 : 온습도 값에 따라 LED 및 Text LCD 제어하기

- 실행 결과
 - 현재 온/습도에 따라 LED가 점등

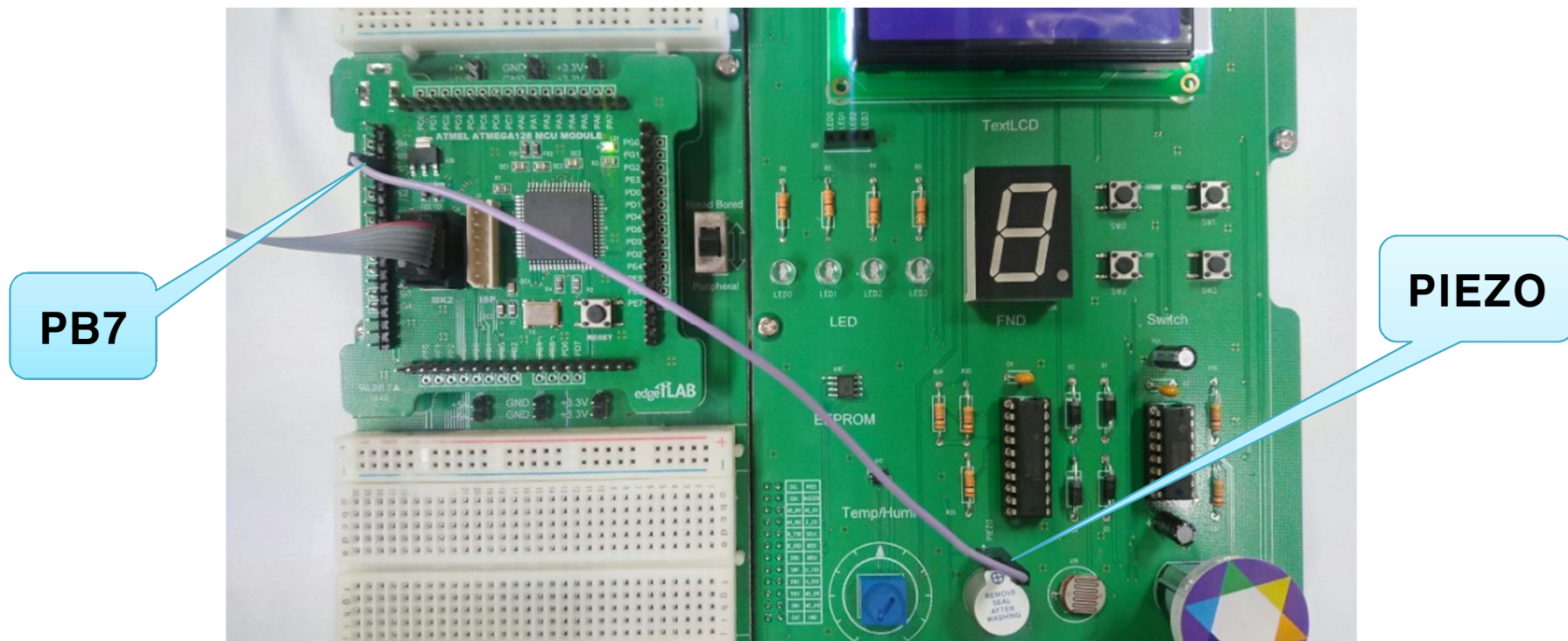


복합 : 스탑워치 제어하기

- 실습 개요
 - SW0으로 스탑 워치 정지 및 시작 기능
 - SW1으로 모드 변경
 - SW 2,3으로 시간 설정
 - Text LCD에 시간을 출력
 - 시간이 다 되면, PIEZO를 울리고 SW를 누르면 해제
- 실습 목표
 - Text LCD의 동작 원리 이해
 - Text LCD 제어 프로그램 방법의 이해
 - 인터럽트의 이해
 - PWM 출력의 이해

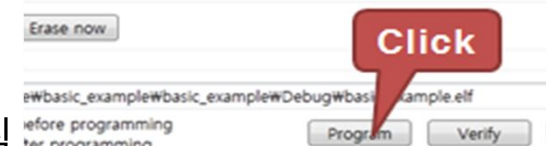
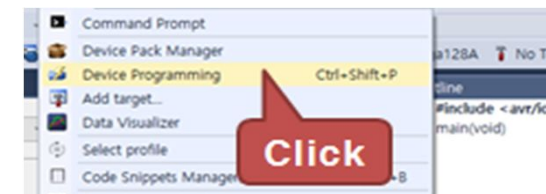
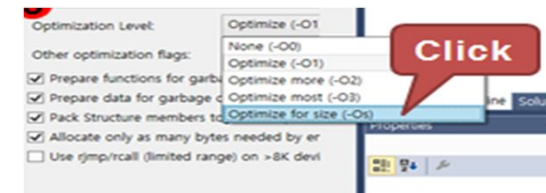
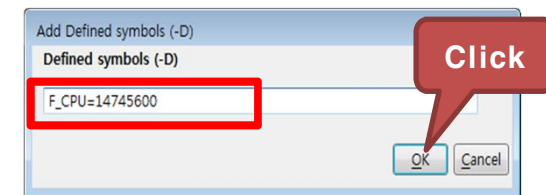
복합 : 스탑워치 제어하기

- 실습 준비
 - 사용보드를 다음 그림과 같이 연결
 - Edge-MCU보드의 PB7 → Edge-Peri 보드의 PIEZO(PIEZO상단우측)



복합 : 스탑워치 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 14_SwitchPiezo_Application, Location : D:\WAVR_Example
 - Device Selection : ATmega128A
 - 프로젝트 설정
 - Project 탭에서 "... Properties..." 선택
 - Toolchain -> AVR/GNU C Compiler에서
 - Symbols -> F_CPU=14745600 추가
 - Optimization -> Optimize for size (-OS) 선택
 - 저장 (Ctrl+S)
 - 소스코드 작성
 - 프로젝트 빌드
 - Build 탭에서 "Build Solution" 클릭
 - 프로그래밍
 - Tool 탭에서 "Device Programming" 클릭
 - AVRISP mkII, ATmega128A 선택 후 "Apply" 클릭
 - 인식 완료되면, Memories 탭 선택, "Program" 클릭



복합 : 스탑워치 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 라이브러리 함수파일을 프로젝트내로 복사
 - lcd.c, lcd.h, lcdconf.h 파일은 "AVR_Example\library\lcd" 폴더에 존재
 - 라이브러리 파일은 Atmel Studio에서 새 프로젝트를 생성한 후 프로젝트가 생성된 폴더안에 있는 프로젝트 폴더에 복사

lcd Library 폴더

이름
lcd
lcd
lcdconf

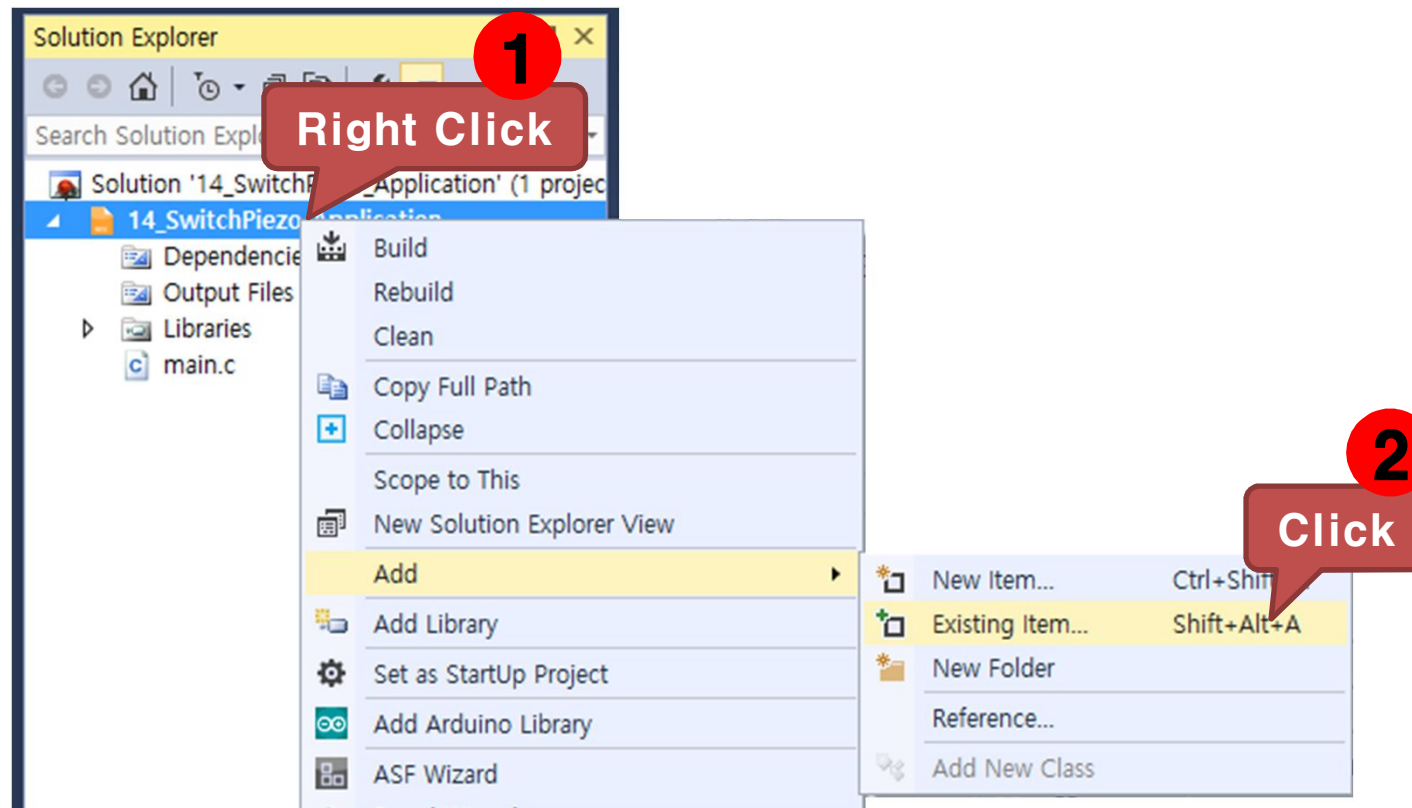
14_SwitchPiezo_Application 프로젝트 폴더

이름	수정한 날짜	유형
Debug	2018-02-08 오후...	파일 폴더
14_SwitchPiezo_Application.component...	2018-02-08 오후...	XML 문서
14_SwitchPiezo_Application.cproj	2018-02-08 오후...	ATMEL Studio
main.c	2018-02-08 오후...	C 소스

파일 복사

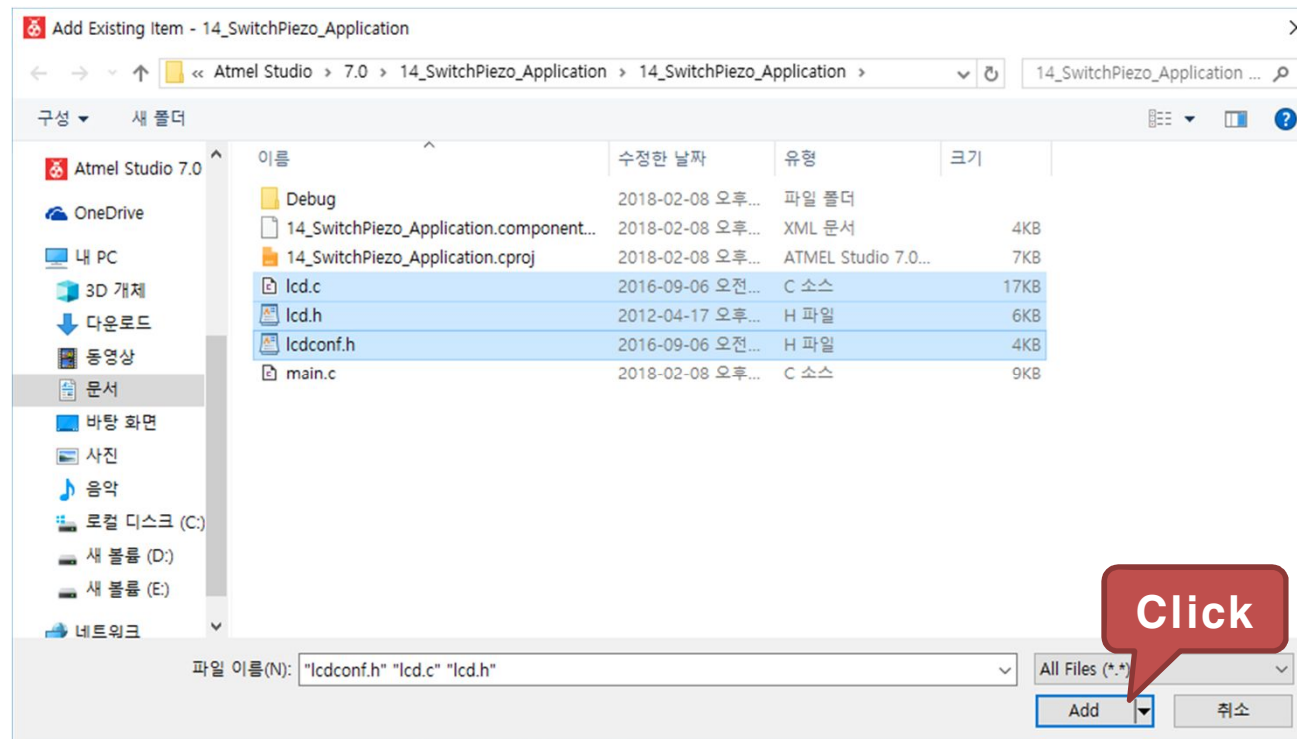
복합 : 스탱워치 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - Atmel Studio 상에서 프로젝트의 솔루션 탐색기에서 라이브러리 파일들을 추가



복합 : 스탑워치 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - 세개의 파일을 추가
 - Ctrl 키를 누르고 파일 세개를 순서대로 클릭하여 동시에 선택하고 한번에 추가



복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
#include <avr/io.h>           // AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>     // AVR 인터럽트에 대한 헤더파일
#include <util/delay.h>        // delay 함수사용을 위한 헤더파일
#include "lcd.h"               //Text LCD를 사용하기 위한 헤더 파일

//피아노 음계에 해당하는 PWM 주파수
unsigned int Piezo = 1046;
volatile signed char Time_buff[4] = {0 ,0 ,0, 0};
volatile unsigned char Count_flag = 0, Setup_flag = 0, Status_flag
= 1;
volatile unsigned char Buzzer_flag = 0, LCD_flag = 1, LCD_flag2 =
0;
volatile unsigned int Time_count = 0, last_Time = 0;
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    unsigned int tmp;

    DDRB = 0x80; // PWM 출력, OCR1C
    // Piezo를 PB7에 연결한다.
    DDRE = 0x00; // 포트E 를 입력포트로 설정한다.

    lcdInit();                //Text LCD를 초기화

    EICRB = 0xFF;             // 인터럽트 4, 5, 6, 7을 상승엣지에서 동작하
    도록 설정한다.
    EIMSK = 0xF0;             // 인터럽트 4, 5, 6, 7을 허용
    EIFR = 0xF0; // 인터럽트 4, 5, 6, 7 플래그를 클리어
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    TCCR1A |= 0x0A;          // COM1C(1:0) = "10", OC1C핀 사용,
    WGM3(1:0) = "10"
    TCCR1B |= 0x19;          // WGM3(3:2) = "11" , CS3(2:0) = "001" 1
    분주 사용
    TCCR1C = 0x00; // WGM3(3:0) = "1110", Fast PWM, 모드 14
    TCNT1 = 0x0000; // 타이머1 카운터 초기화

    ICR1 = 14745600/Piezo; // 부저 주기 설정

    TCCR3A = 0x00;
    TCCR3B = 0x05; // 프리스케일러 1024;
    TCCR3C = 0x00;
    TCNT3 = 51136; // 65536 - 14400 = 51136-> 1초 마다 한번씩 인터럽
    트 발생
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    ETIMSK = 0x04;
    ETIFR |= 1 << TOV3;

    lcdGotoXY(0,0); //현재 커서위치를 첫번째줄 첫번째칸으로 이동한다.
    //첫번째 매개변수는 행을 의미하고, 두번째 매개변수는 열을 의미한다.
    lcdPrintData("Status : ",9); //"Status : " 문자열을 출력한다.

    lcdGotoXY(2,1); //현재 커서위치를 두번째 줄 세번째 칸으로 이동한다.
    lcdPrintData("Time      : ",12);
    //"Time      : " 문자열을 출력한다.

    sei();
}
```


복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    while(1)
    {
        if(Status_flag)
        {
            // 현재 상태를 LCD 1행에 출력
            lcdGotoXY(9,0);
            switch(Status_flag)
            {
                case 1:
                    lcdPrintData("Standby",7);    // 대기(준비) 상태
출력
                break;
                case 2:
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    lcdPrintData(" Count ",7);    // 카운트 상태 출
    력
    break;
    case 3:
        lcdPrintData("Finish ",7);    // 카운트 완료 상
    태 출력(부저 울림)
    break;
    case 4:
        lcdPrintData(" Stop  ",7);    // 카운트 정지 상
    태 출력
    break;
    case 5:
        lcdPrintData(" Setup ",7);    // 시간 설정 상태
    출력
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
        break;
    }
    if((Status_flag == 5) && (Setup_flag == 1))
        LCD_flag2 = 4;           // LCD 커서 및 커서 깜박임 설정
        Status_flag = 0;
    }
    if(LCD_flag)    // 현재 카운트 시간을 LCD 2행에 출력
    {
        LCD_flag = 0;
        tmp = Time_count;
        lcdGotoXY(9,1);
        // 분의 십의 자리 출력
        lcdDataWrite((tmp/60)/10 + '0');
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    // 분의 일의 자리 출력
    lcdDataWrite((tmp/60)%10 + '0');
    lcdGotoXY(12,1);
    // 초의 십의 자리 출력
    lcdDataWrite((tmp%60)/10 + '0');
    // 초의 일의 자리 출력
    lcdDataWrite((tmp%60)%10 + '0');
}
if(Setup_flag && LCD_flag2)
{
    if(Setup_flag < 5)
    {
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    // 설정 시간을 출력
    // LCD_flag2 : 2 => 해당값 증가
    //              3 => 해당값 감소
    // Setup_flag 1 => 분의 십의 자리 선택
    // Setup_flag 2 => 분의 일의 자리 선택
    // Setup_flag 3 => 초의 십의 자리 선택
    // Setup_flag 4 => 초의 일의 자리 선택
    if(LCD_flag2 == 4)
    {
        // LCD 커서 ON, 깜박임 ON 으로 설정
        lcdControlWrite(1<<LCD_ON_CTRL |
            1<<LCD_ON_DISPLAY | 1<<LCD_ON_CURSOR | 1<<LCD_ON_BLINK);
        _delay_ms(100);
    }
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    if((Setup_flag == 1) || (LCD_flag2 == 2) || (LCD_flag2 == 3))
    {
        if(LCD_flag2 == 2)
        {
            // 해당하는 시간 값을 증가
            Time_buff[Setup_flag-1]++;
            if(Time_buff[Setup_flag-1] > 9)
                Time_buff[Setup_flag-1] = 0;
        }
        else if(LCD_flag2 == 3)
        {
            // 해당하는 시간 값을 감소
            Time_buff[Setup_flag-1]--;
        }
    }
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...

    if(Time_buff[Setup_flag-1] < 0)
        Time_buff[Setup_flag-1] = 9;
}
lcdGotoXY(9,1);
// 분의 십의 자리 출력
lcdDataWrite(Time_buff[0] + '0');
// 분의 일의 자리 출력
lcdDataWrite(Time_buff[1] + '0');
lcdGotoXY(12,1);
// 초의 십의 자리 출력
lcdDataWrite(Time_buff[2] + '0');
// 초의 일의 자리 출력
lcdDataWrite(Time_buff[3] + '0');
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    // 설정할 시간으로 커서를 이동하여 깜박이도록 함.
    if(Setup_flag < 3)
        lcdGotoXY((Setup_flag + 8),1);
    else
        lcdGotoXY((Setup_flag + 9),1);
}
else
{
    // 설정 값을 Time_count 변수에 계산하여 저장함.
    Time_count = Time_buff[0]*10 + Time_buff[1];
    Time_count = Time_count * 60 + Time_buff[2]*10 +
    Time_buff[3];
    Status_flag = 1;    // 대기 상태 출력
```


복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    //...
    LCD_flag = 1;           // 시간 출력
    lcdControlWrite(1<<LCD_ON_CTRL | 1<<LCD_ON_DISPLAY);
    Setup_flag = 0;         // 설정 완료
}
LCD_flag2 = 0;             // 설정에 대한 LCD 출력 완료
}
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(TIMER3_OVF_vect)
{
    cli();
    TCNT3 = 51136; // 65536 - 1400 = 51136-> 1초 마다 한번씩 인터럽트
    발생
    if(Count_flag)
    {
        if(Time_count == 0)
        {
            Count_flag = 0;           // 시간 Count 완료
            Buzzer_flag = 1;          // 부저 ON 상태
            OCR1C = ICR1/2;           // 부저 ON
            Status_flag = 3;          // 완료 상태 출력
        }
    }
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(TIMER3_OVF_vect)
{
    ...
    else
    {
        Time_count--;           // 시간 감소
    }
    LCD_flag = 1;              // 현재 남은 시간을 출력하도록 설정
}
sei();
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT4_vect) // 인터럽트 서비스 루틴
{
    cli(); // 전체 인터럽트를 금지

    if(Setup_flag == 0)
    {
        if(Buzzer_flag == 1)
        {
            Buzzer_flag = 0; // 부저 OFF 상태
            OCR1C = 0; // 부저 OFF
            Status_flag = 1; // 대기 상태 출력
            Time_count = last_Time; // 카운트 동작전 시간을 현재 시간
에 저장
            LCD_flag = 1; // 현재 남은 시간을 출력하도록 설정
        }
        else
        {
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT4_vect) // 인터럽트 서비스 루틴
{
    ...
    if(Count_flag == 0)
    {
        Count_flag = 1; //카운트 ON 상태
        Status_flag = 2; // count 상태 출력
        last_Time = Time_count; // 카운트 동작전 시간에 현
재 시간을 저장
    }
    else
    {
        Count_flag = 0; //카운트 ON 상태
        Status_flag = 4; // 정지 상태 출력
    }
}
sei(); // 전체 인터럽트를 허용
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT5_vect) // 인터럽트 서비스 루틴
{
    cli(); // 전체 인터럽트를 금지

    if(Buzzer_flag == 1)
    {
        Buzzer_flag = 0; // 부저 OFF 상태
        OCR1C = 0; // 부저 OFF
        Status_flag = 1; // 대기 상태 출력
        Time_count = last_Time; // 카운트 동작전 시간을 현재 시간에
저장
        LCD_flag = 1; // 현재 남은 시간을 출력하도록 설정
    }
    else if(Count_flag == 0)
    {
        LCD_flag2 = 1;
        Setup_flag++; // 설정 및 변경할 시간의 자릿수 선택
    }
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT5_vect) // 인터럽트 서비스 루틴
{
    ...
    if(Setup_flag == 1)
    {
        Status_flag = 5; // 설정 상태 출력
        // 현재 남은 시간 값 Time_buff에 저장
        Time_buff[0] = (Time_count/60)/10; // 분의 십의 자리
        Time_buff[1] = (Time_count/60)%10; // 분의 일의 자리
        Time_buff[2] = (Time_count%60)/10; // 초의 십의 자리
        Time_buff[3] = (Time_count%60)%10; // 초의 일의 자리
    }
    else
        LCD_flag2 = 1; // 설정모드에서 LCD 출력
    }

    sei(); // 전체 인터럽트를 허용
}
```

복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT6_vect) // 인터럽트 서비스 루틴
{
    cli();          // 전체 인터럽트를 금지

    if(Setup_flag)
        LCD_flag2 = 2; // 시간 증가
    else if(Buzzer_flag == 1)
    {
        Buzzer_flag = 0; // 부저 OFF 상태
        OCR1C = 0;       // 부저 OFF
        Status_flag = 1; // 대기 상태 출력
        Time_count = last_Time; // 카운트 동작전 시간을 현재 시간에 저장
        LCD_flag = 1;    // 현재 남은 시간을 출력하도록 설정
    }

    sei();          // 전체 인터럽트를 허용
}
```


복합 : 스탑워치 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(INT7_vect) // 인터럽트 서비스 루틴
{
    cli();          // 전체 인터럽트를 금지

    if(Setup_flag)
        LCD_flag2 = 3; // 시간 감소
    else if(Buzzer_flag == 1)
    {
        Buzzer_flag = 0; // 부저 OFF 상태
        OCR1C = 0;       // 부저 OFF
        Status_flag = 1; // 대기 상태 출력
        Time_count = last_Time; // 카운트 동작전 시간을 현재 시간에 저장
        LCD_flag = 1;    // 현재 남은 시간을 출력하도록 설정
    }

    sei();          // 전체 인터럽트를 허용
}
```

복합 : 스탑워치 제어하기

- 실행 결과
 - SW1,2,3을 이용하여 스탑워치의 시간을 조정
 - SW0을 이용하여 시작/정지 가능
 - 시간이 0이 되면 PIEZO가 울리며, SW를 누르면 꺼짐

