

SERVO MOTOR

- 서보 모터(Servo Motor)
- 서보 모터 위치 제어 1
- 서보 모터 위치 제어 2
- 서보 모터 위치 제어 3
- 서보 모터 위치 제어 4
- PC에서 정보를 입력받아 서보모터 제어하기



엣지아이랩

서보 모터(Servo Motor)

- 서보의 어원
 - 서보기구라는 용어는 1934년에 H.L.Hazen 교수에 의해 처음 쓰여졌으나, Servo의 어원은 라틴어의 Servus(영어의 Slave : 노예)라고 함
 - 노예의 역할이 주인의 명령을 충실히 따르고 육체노동을 하는 것이므로 그러한 역할을 해내는 장치로 명명
- 서보의 역사
 - 본래의 목적인 위치에 최초로 응용한 것은 어뢰
 - 어뢰의 자동 조정에 이어 배와 항공기의 자동 조정이 실현
 - 자동제어는 속도, 서보의 순으로 발달하여 1920년경부터 프로세스제어에도 실현
 - 레이더로 항공기를 자동적으로 추정하는 연구하던 중 시스템으로서 서보계 하드웨어의 개발에 성공하면서 이러한 제어계를 설계, 조정하는 실용적 제어이론의 개발에 성공 피드백 제어이론이 체계화
 - 오일쇼크로 인해 유압서보에서 전기서보로 바뀜
 - 처음에는 DC서보를 사용했으나 교류서보 전동기를 이용한 AC서보의 성능이 향상되면서 더 효율적인 AC서보를 더 많이 사용하게 됨

서보 모터(Servo Motor)

- 서보 모터의 종류

DC 서보모터	AC 서보 모터
브러시 모터(Brushed Motor)	브러시리스 모터(Brushless Motor)
제어구조가 간단하고 쉽다	제어구조가 복잡하고 어렵다
단상으로 제어한다	3상을 제어한다
회전 전기자형	회전 자계형
회전자가 권선으로 방열 나쁘다	고정자가 권성으로 방열이 쉽다
브러시의 유지 보수가 필요하다	브러시의 유지 보수가 필요없다
기계적 구조로 최대 속도가 낮다	전기적 구조로 최대 속도가 높다
정격 용량을 크게 하기 어렵다	정격 용량을 크게 하기 어렵다

서보 모터(Servo Motor)

● 서보 모터의 종류에 따른 장단점

종류	장점	단점
DC서보모터	<ul style="list-style-type: none"> -기동토크가 크다. -크기에 비해 큰 토크 발생. -효율이 높다. -제어기 용이하다. -속도제어 범위가 넓다. -비교적 가격이 저렴하다. 	<ul style="list-style-type: none"> -브러쉬 마찰로 기계적 손실이 큼 -브러쉬의 보수가 필요 -접촉부의 신뢰성이 떨어진다. -정류속도에 한계가 있다. -사용환경에 제한이 있다. -방열이 나쁘다.
동기기형 AC서보모터	<ul style="list-style-type: none"> -브러쉬가 없어 보수가 용이 -내환경성이 우수 -정류속도에 한계가 없다. -신뢰성이 우수하다. -고속, 고 토크의 사용이 가능 -방열이 좋다. 	<ul style="list-style-type: none"> -시스템이 복잡하고,고가이다. -전기적 시정수가 크다. -회전 검출기가 필요 -영구자석을 고정자로 사용하므로 대용량의 모터 제작이 어렵다.
유도기형 AC서보모터	<ul style="list-style-type: none"> -브러쉬가 없어 보수가 용이 -내환경성이 우수 -자석을 사용하지 않는다. -방열이 좋다. -회전검출기가 불필요 	<ul style="list-style-type: none"> -시스템이 복잡하고 고가이다. -전기적 시정수가 크다.

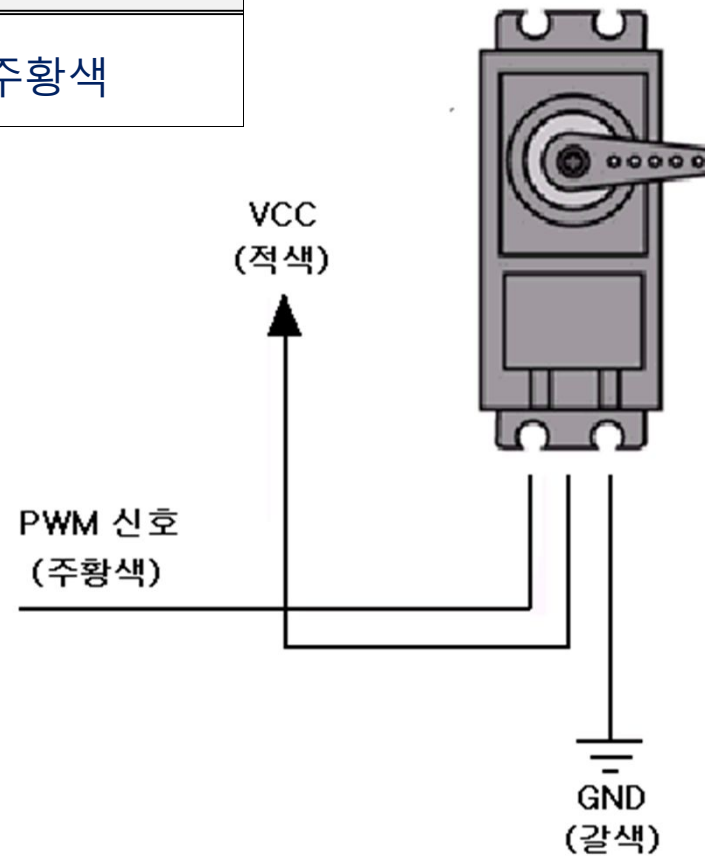
서보 모터(Servo Motor)

- DC 서보 모터의 제어원리
 - DC 서보 모터는 전기자 전류에 대하여 발생 토크의 관계가 직선성으로 우수하고, OA에서부터 FA분야까지 광범위하게 사용
 - 서보 모터는 자동화시스템에서 Actuator로 사용되는 중요한 요소
 - 서보모터는 경량, 소형, 설치의 용이성, 고효율성, 정확한 제어성, 유지, 보수가 용이한 특징
 - 서보모터는 DC, 스텝 모터와는 다르게 각도에 보통 180°의 제한

서보 모터(Servo Motor)

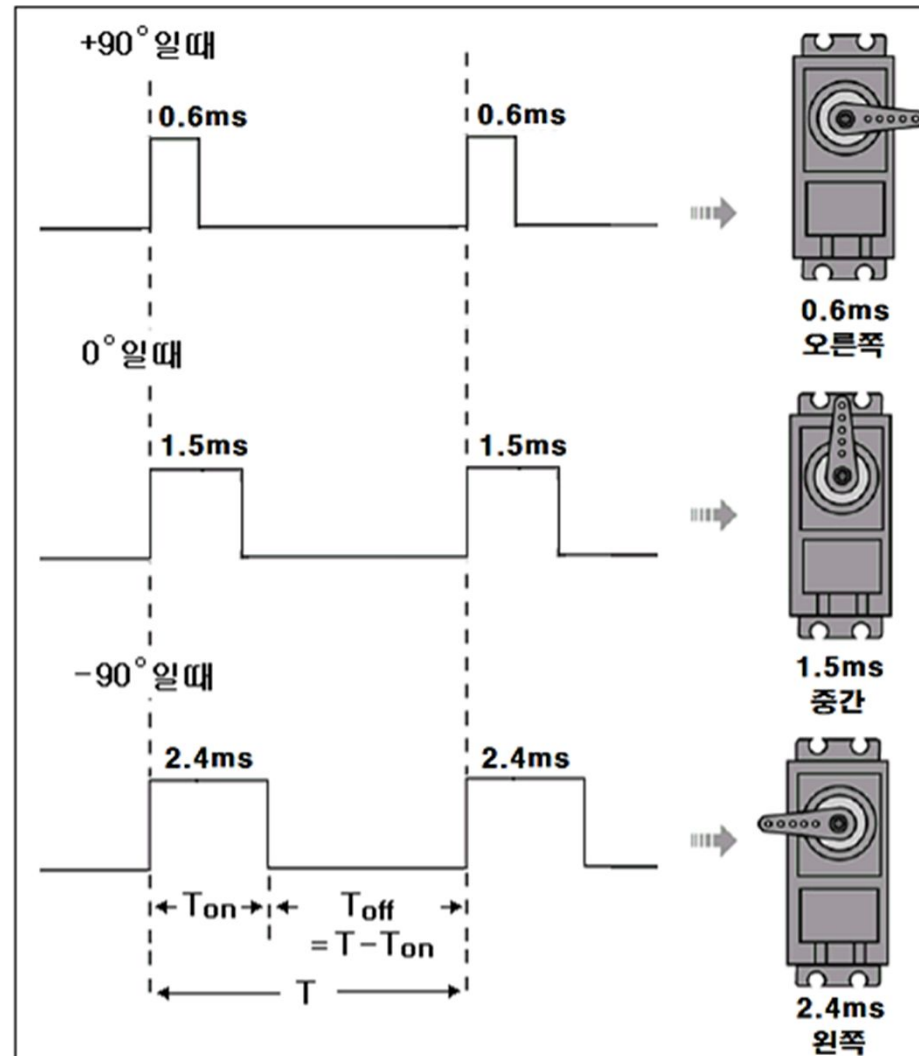
- 서보모터는 펄스 입력선으로 일정주기의 펄스신호를 주면 모터가 0~180도 사이에서 동작

GND	VCC	PWM 신호 입력선
갈색	적색	주황색



서보 모터(Servo Motor)

- 서보모터 PWM의 일반적인 주기는 20[ms]
 - High 부분의 펄스 폭에 따라 결정

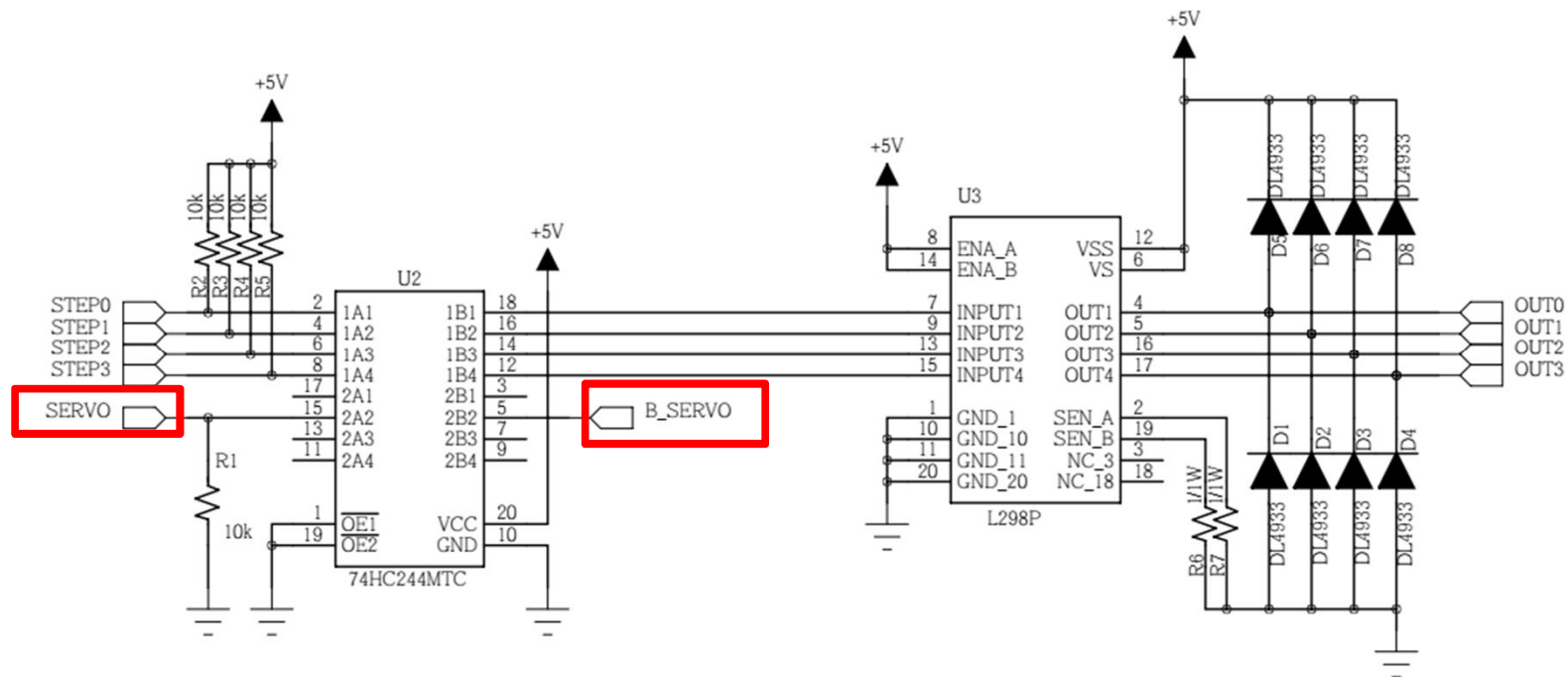


실습 1 : 서보 모터 위치 제어 1

- 실습 개요
 - ATmega128의 GPIO핀에 서보 모터의 제어 신호를 연결하여, 서보 모터의 각도를 제어
 - 시간 지연 함수를 사용하여 서보의 위치를 영점(0° 위치)에 오도록 함
- 실습 목표
 - 서보 모터의 동작원리를 이해
 - ATmega128A의 GPIO를 이용한 서보 모터 구동 방법 습득

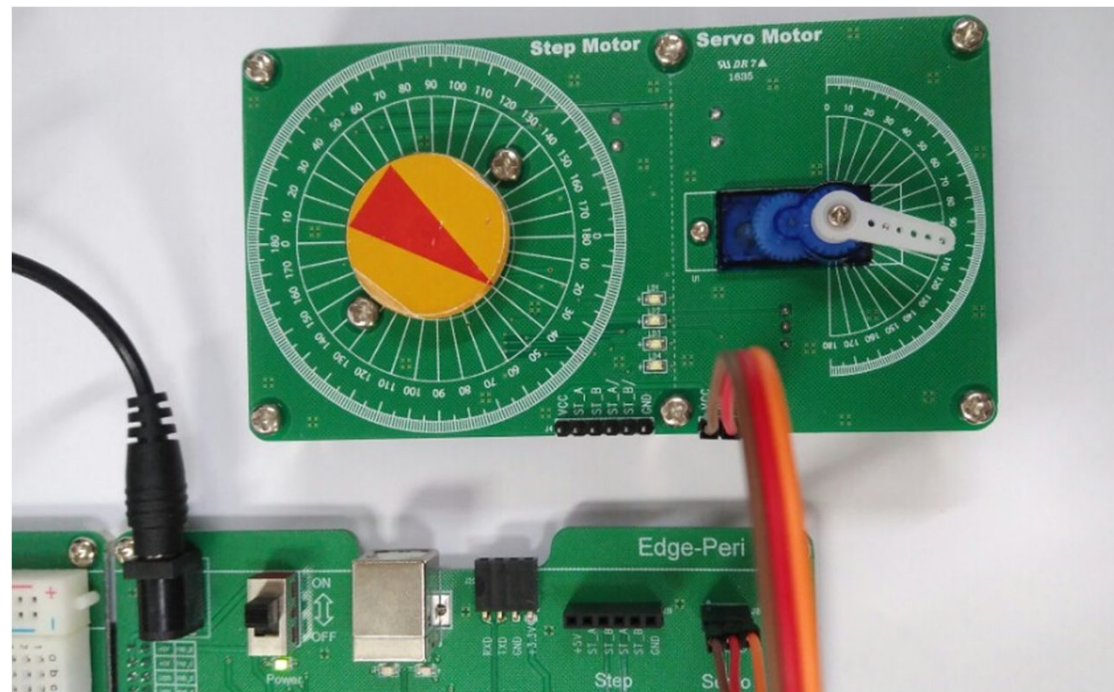
실습 1 : 서보 모터 위치 제어 1

- 사용 모듈
 - 서보 모터 모듈의 회로



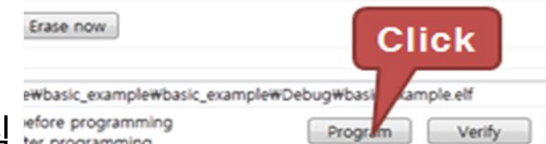
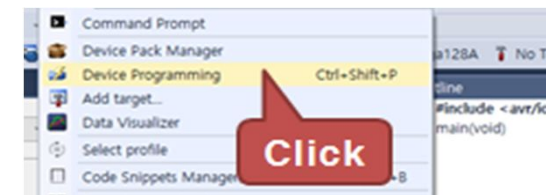
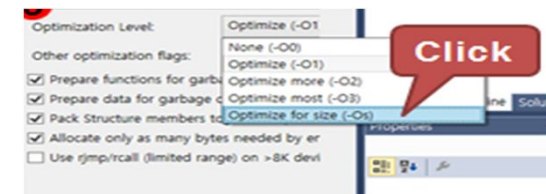
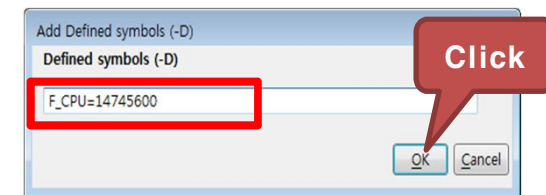
실습 1 : 서보 모터 위치 제어 1

- 실습 준비
 - 사용 보드를 다음 그림과 같이 연결
 - Edge-Peri보드의 +5V → Servo 모터 보드의 +5V
 - Edge-Peri보드의 PWM → Servo 모터 보드의 PWM
 - Edge-Peri보드의 GND → Servo 모터 보드의 GND



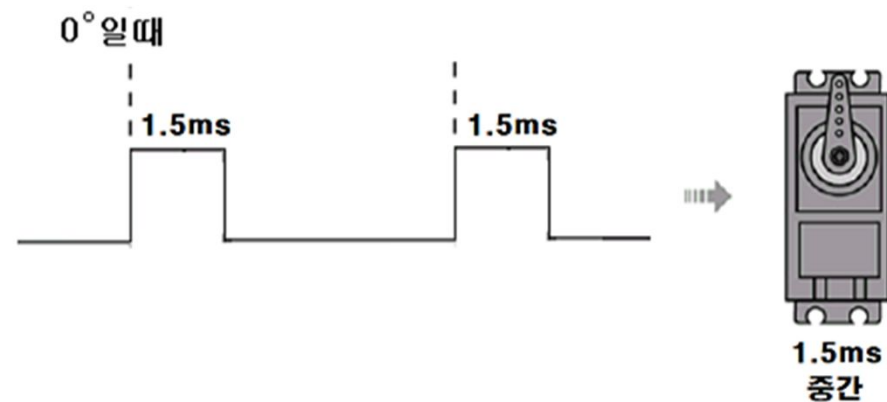
실습 1 : 서보 모터 위치 제어 1

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 13_SERVO_Example_01, 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



실습 1 : 서보 모터 위치 제어 1

- 구동 프로그램 : 사전지식
 - GPIO 사용
 - B_SERVO → PORTE(PE3)
 - 시간 지연 함수 사용
 - 서보 모터의 제어 펄스는 주기가 20ms 이며, High 부분이 1.5ms



실습 1 : 서보 모터 위치 제어 1

- 구동 프로그램
 - main.c

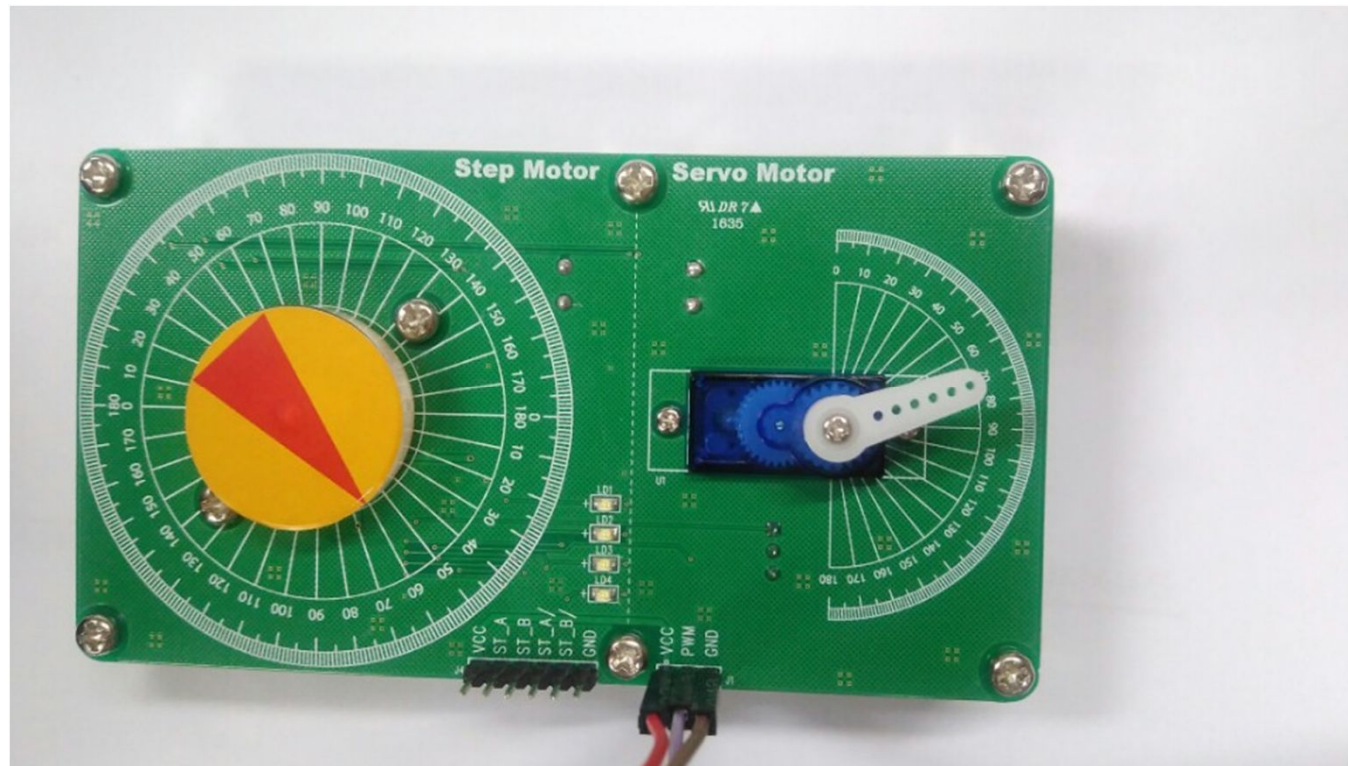
```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h>  // delay 함수사용을 위한 헤더파일

int main(void)
{
    DDRE = 0x08;          // SERVO 핀을 출력 포트로 설정

    while (1)
    {
        // PWM 주기를 20[ms]
        PORTE |= 0x08;
        _delay_us(1500);   // ON Time 1.5[ms]
        PORTE &= ~0x08;
        _delay_us(18500);  // OFF Time 18.5[ms]
    }
}
```

실습 1 : 서보 모터 위치 제어 1

- 실행 결과
 - 서보모터는 초기 상태인 0도로 이동

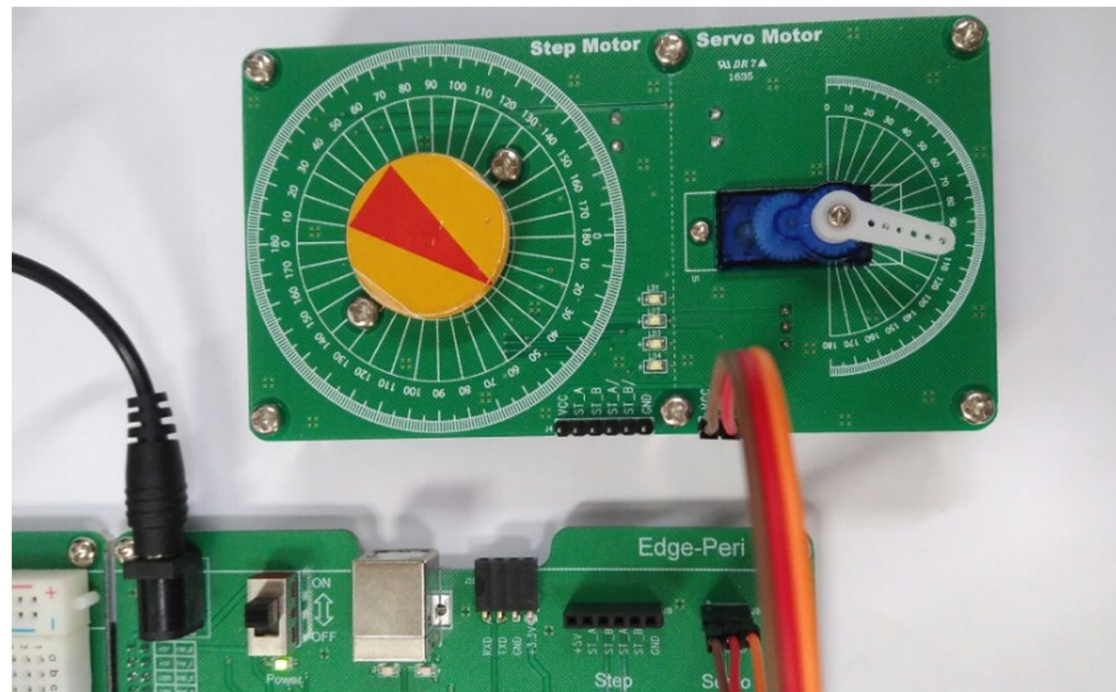


실습 2 : 서보 모터 위치 제어 2

- 실습 개요
 - ATmega128의 GPIO핀에 서보 모터의 제어 신호를 연결하여, 서보 모터의 각도를 제어
 - 시간 지연 함수를 사용하여 서보의 위치를 $+90^{\circ} \rightarrow 0^{\circ} \rightarrow -90^{\circ}$ 를 왕복하는 프로그램을 구현
- 실습 목표
 - 서보 모터의 동작원리를 이해
 - ATmega128A의 GPIO를 이용한 서보 모터 구동 방법 습득

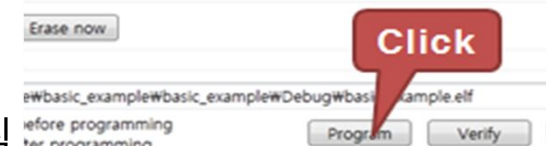
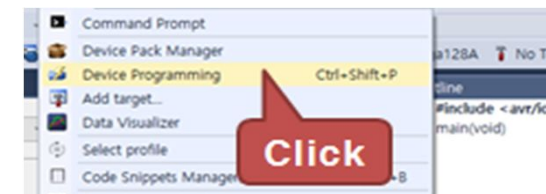
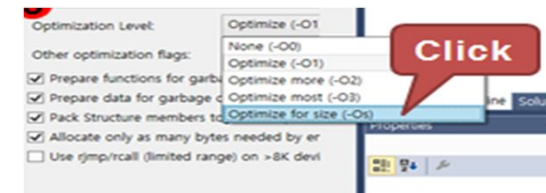
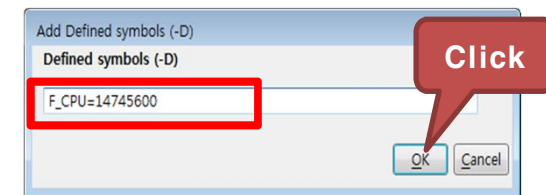
실습 2 : 서보 모터 위치 제어 2

- 실습 준비
 - 사용 보드를 다음 그림과 같이 연결
 - Edge-Peri보드의 +5V → Servo 모터 보드의 +5V
 - Edge-Peri보드의 PWM → Servo 모터 보드의 PWM
 - Edge-Peri보드의 GND → Servo 모터 보드의 GND



실습 2 : 서보 모터 위치 제어 2

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 13_SERVO_Example_02, 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



실습 2 : 서보 모터 위치 제어 2

- 구동 프로그램 : 사전지식
 - GPIO 사용
 - B_SERVO → PORTE(PE3)
 - 시간 지연 함수 사용
 - 서보 모터의 제어 펄스는 주기가 20ms
 - +90° 일때 High 부분이 0.6ms
 - 0° 일때 High 부분이 1.5ms
 - -90° 일때 High 부분이 2.4ms

실습 2 : 서보 모터 위치 제어 2

- 구동 프로그램
 - main.c 코드 작성

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

int main(void)
{
    unsigned char i;
    DDRE = 0x08;          // SERVO 핀을 출력 포트로 설정
    while (1)
    {
        for(i = 0; i < 50 ; i++ )    // +90도 동작
        {
            PORTE |= 0x08;
            _delay_us(600);    // ON Time | 0.6[ms]
            PORTE &= ~0x08;
            _delay_us(19400); // OFF Time | 19.4[ms]
        }
    }
}
```

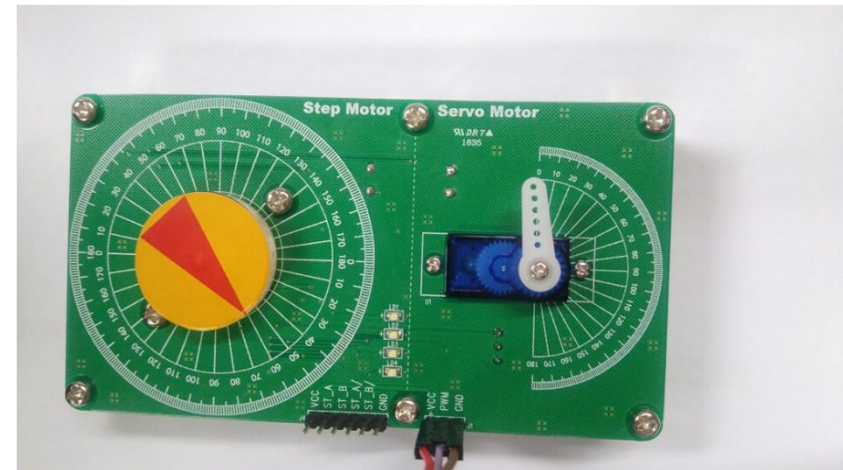
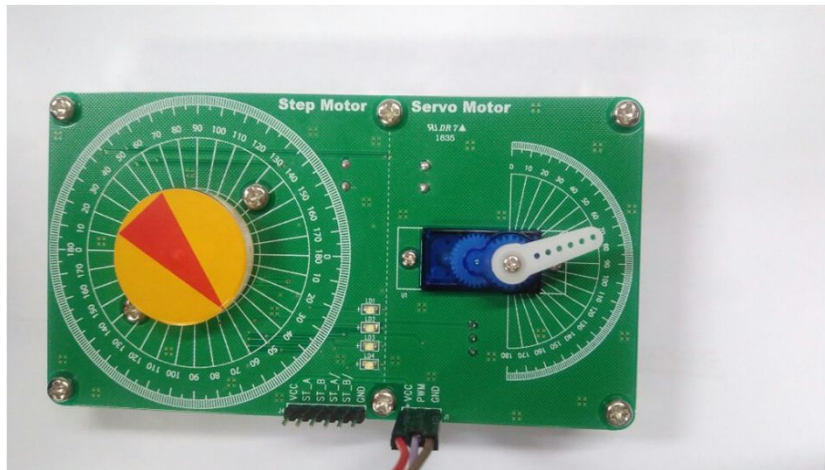
실습 2 : 서보 모터 위치 제어 2

- 구동 프로그램
 - main.c 코드 작성

```
while(1) {  
    //...  
    for(i = 0; i < 50 ; i++ ) {    // 0도 동작  
        PORTE |= 0x08;  
        _delay_us(1500);           // ON Time 0 | 1.5[ms]  
        PORTE &= ~0x08;  
        _delay_us(18500);         // OFF Time 0 | 18.5[ms]  
    }  
    for(i = 0; i < 50 ; i++ ) {    // -90도 동작  
        PORTE |= 0x08;  
        _delay_us(2400);           // ON Time 0 | 2.4[ms]  
        PORTE &= ~0x08;  
        _delay_us(17600);         // OFF Time 0 | 17.6[ms]  
    }  
}  
}
```

실습 2 : 서보 모터 위치 제어 2

- 실행 결과
 - 서보 모터는 $+90^{\circ} \rightarrow 0^{\circ} \rightarrow -90^{\circ}$ 로 반복해서 이동

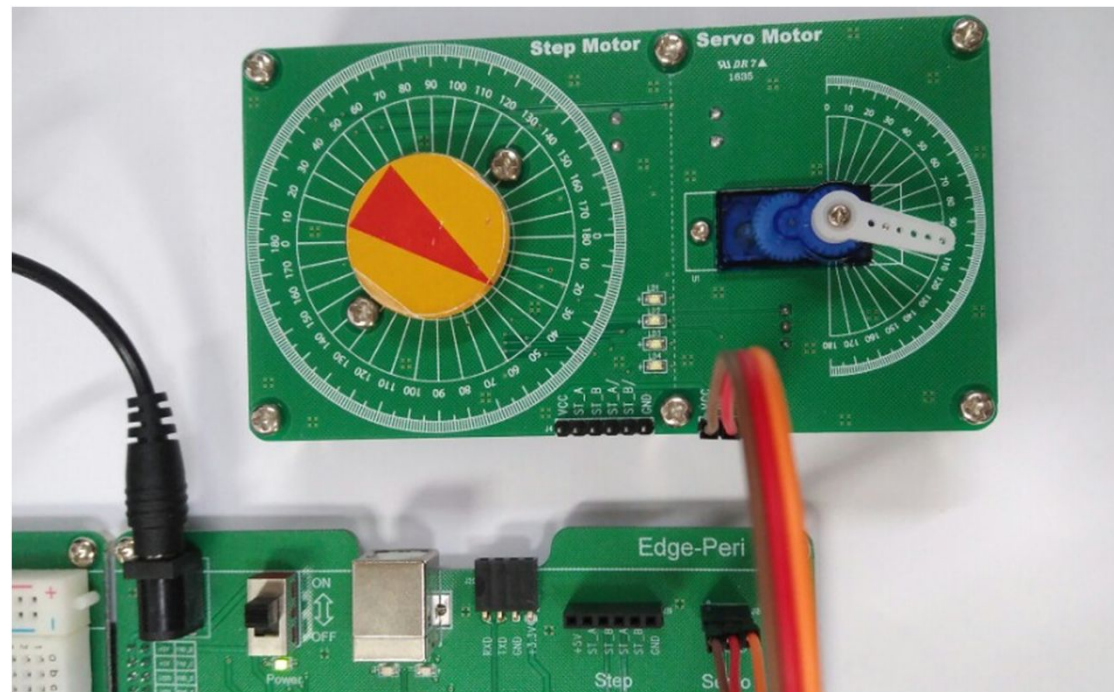


실습 3 : 서보 모터 위치 제어 3

- 실습 개요
 - ATmega128의 GPIO핀에 서보모터의 제어 신호를 연결하여, 서보모터의 각도를 제어
 - 타이머를 사용하여 서보 제어 신호를 만들고, 서보의 위치를 $+90^{\circ} \rightarrow 0^{\circ} \rightarrow -90^{\circ}$ 를 왕복하는 프로그램을 구현
- 실습 목표
 - 서보 모터의 동작원리를 이해
 - ATmega128A의 GPIO를 이용한 서보 모터 구동 방법 습득
 - ATmega128A의 타이머 사용법 습득

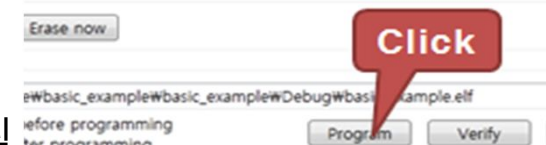
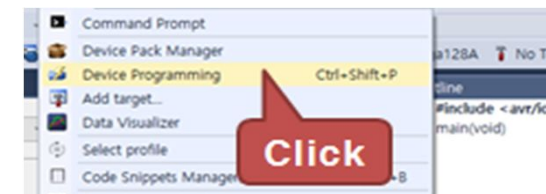
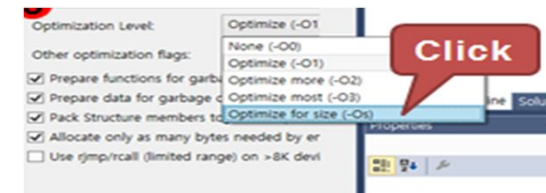
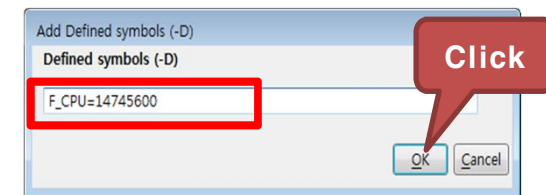
실습 3 : 서보 모터 위치 제어 3

- 실습 준비
 - 사용 보드를 다음 그림과 같이 연결
 - Edge-Peri보드의 +5V → Servo 모터 보드의 +5V
 - Edge-Peri보드의 PWM → Servo 모터 보드의 PWM
 - Edge-Peri보드의 GND → Servo 모터 보드의 GND



실습 3 : 서보 모터 위치 제어 3

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 13_SERVO_Example_03, 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



실습 3 : 서보 모터 위치 제어 3

- 구동 프로그램 : 사전지식
 - GPIO 사용
 - B_SERVO → PORTE(PE3)
 - 타이머 사용
 - 서보 모터의 제어 펄스는 주기가 20ms
 - 타이머/카운터 0이 약 0.1[ms] 마다 오버플로가 발생하기 위해서는 먼저 타이머/카운터 0의 소스클록을 몇 분주로 할것인지 TCCR0 레지스터를 설정
 - CLK/8 분주로 설정

	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
TCCR0	0	0	0	0	0	0	1	0

- 카운트 TCNT0의 초기 값을 설정
 - $TCNT0 = 0xff(\text{카운트 최댓값}) + 1 - \text{클록 카운트 값}(0xB8) = 0x48$
 - $TCNT0 = 0x48$ 로 설정하면 $256(0x100) - 72(0x48) = 184(0xB8)$
즉, 184번 클록을 카운트 한 다음에 인터럽트가 발생

실습 3 : 서보 모터 위치 제어 3

- 구동 프로그램
 - main.c 코드 작성

```
#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

int timer0Cnt = 0;
int AngleCount = 0;

void Servo(int);
SIGNAL(TIMERO_OVF_vect); // Timer0 Overflow0 ISP

int main(void) {
    DDRE = 0x08;          // SERVO 핀을 출력 포트로 설정
    AngleCount = 23;      // 초기 서보 위치를 +90도로 설정
    TCCR0 = 0x02;         // 8분주
    TCNT0 = 0x48;
    // 256(0x100)-72(0x48)=184(0xB8) -> 0.1ms 마다 인터럽트 발생
```

실습 3 : 서보 모터 위치 제어 3

- 구동 프로그램
 - main.c 코드 작성

```
int main(void) {  
    //...  
    TIMSK = 0x01;           // TOIE0 '1'  
    TIFR = 0x01;           // TOV0 '1'  
    sei();  
  
    while (1) {  
        // delay 1초를 주는 이유는 서보가 움직일 충분한 시간을 주기 위함  
        Servo(90);  
        _delay_ms(1000);  
        Servo(0);  
        _delay_ms(1000);  
        Servo(-90);  
        _delay_ms(1000);  
    }  
}
```

실습 3 : 서보 모터 위치 제어 3

- 구동 프로그램
 - main.c 코드 작성

```
void Servo(int Angle)
{
    if (Angle == -90 )
        AngleCount = 23;    // PWM ON Time 2.3[ms]
    else if (Angle == 0)
        AngleCount = 15;    // PWM ON Time 1.5[ms]
    else if (Angle == 90)
        AngleCount = 7;     // PWM ON Time 0.7[ms]
}
```

실습 3 : 서보 모터 위치 제어 3

- 구동 프로그램
 - main.c 코드 작성

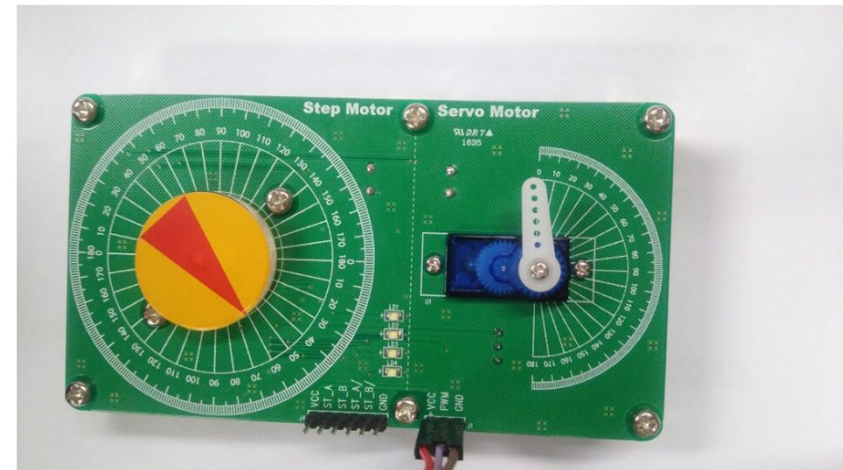
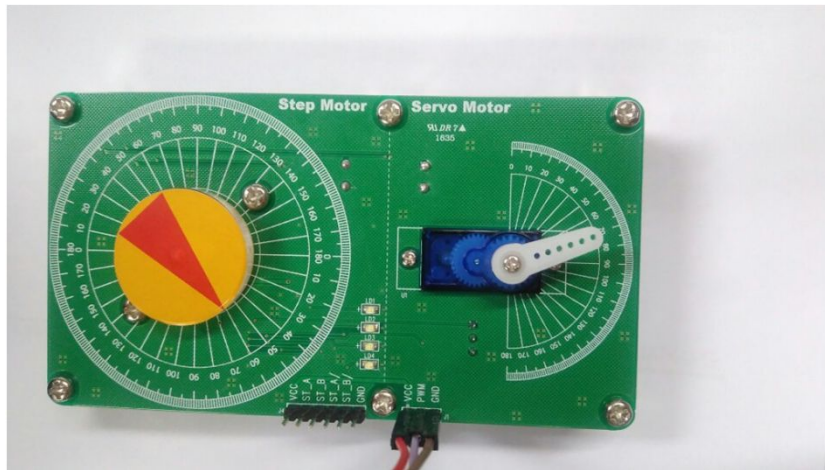
```
SIGNAL(TIMER0_OVF_vect)
{
    cli();
    TCNT0 = 0x48;
    // 256(0x100)-72(0x48)=184(0xB8)->0.1ms마다 한번씩 인터럽트 발생
    timer0Cnt++;          // timer0Cnt 변수를 1 증가

    if(timer0Cnt <= AngleCount) {
        PORTE |= 0x08;    // PWM ON Time
    }
    else PORTE &= ~0x08;  // PWM OFF Time

    if(timer0Cnt >= 200)
        timer0Cnt=0;      // timer0Cnt 카운터 초기화
    sei();
}
```

실습 3 : 서보 모터 위치 제어 3

- 실행 결과
 - 서보 모터는 $+90^\circ \rightarrow 0^\circ \rightarrow -90^\circ$ 로 반복해서 이동

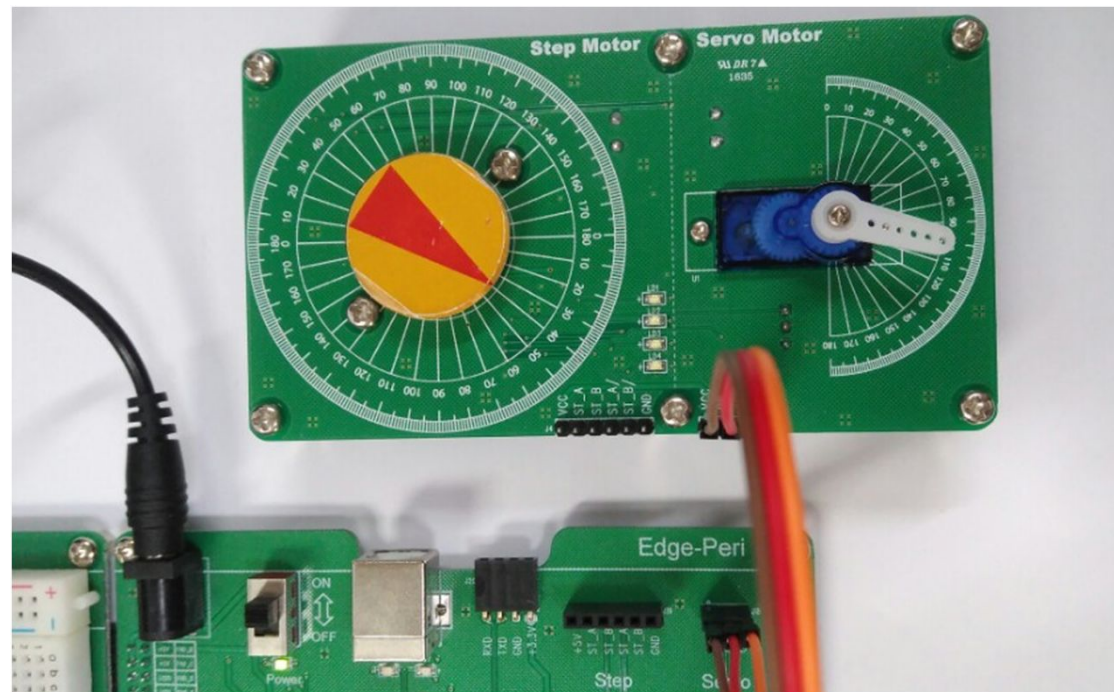


실습 4 : 서보 모터 위치 제어 4

- 실습 개요
 - ATmega128의 GPIO핀에 서보 모터의 제어신호를 연결하여, 서보 모터의 각도를 제어
 - 타이머의 PWM 기능을 사용하여 서보 제어신호를 만들고, 서보의 위치를 $+90^{\circ} \rightarrow 0^{\circ} \rightarrow -90^{\circ}$ 를 왕복하는 프로그램을 구현
- 실습 목표
 - 서보 모터의 동작원리를 이해
 - ATmega128A의 GPIO를 이용한 서보 모터 구동 방법 습득
 - ATmega128A의 타이머의 PWM 기능 사용법 습득

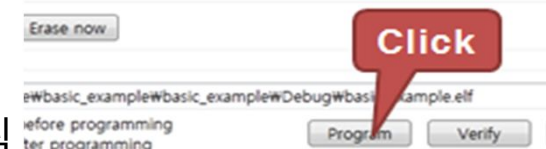
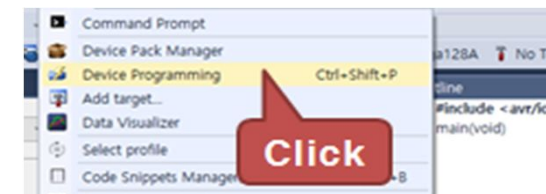
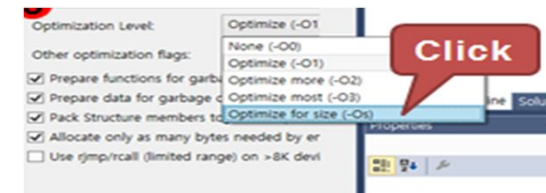
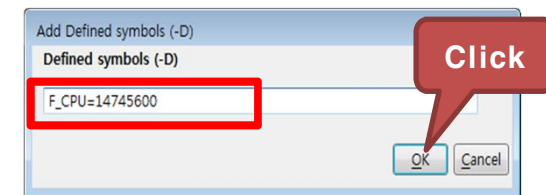
실습 4 : 서보 모터 위치 제어 4

- 실습 준비
 - 사용 보드를 다음 그림과 같이 연결
 - Edge-Peri보드의 +5V → Servo 모터 보드의 +5V
 - Edge-Peri보드의 PWM → Servo 모터 보드의 PWM
 - Edge-Peri보드의 GND → Servo 모터 보드의 GND



실습 4 : 서보 모터 위치 제어 4

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 13_SERVO_Example_04, 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



실습 4 : 서보 모터 위치 제어 4

- 구동 프로그램 : 사전지식
 - PWM 사용
 - B_SERVO → PORTE(PE3) → OCR3A
 - 타이머 사용
 - 서보 모터의 제어 펄스는 주기가 20ms 이다.
 - 타이머/카운터 1을 모드 8 Phase and Frequency PWM 모드로 설정
 - CLK/8 분주로 설정

	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM11	WGM10
TCCR3A	1	0	0	0	0	0	0	0
	ICNC3	ICES3		WGM33	WGM32	CS32	CS31	CS30
TCCR3B	0	0	0	1	0	0	1	0

- ICR1의 초기 값을 설정
 - 18432으로 설정 $14745600/2*8*18432 = 50\text{Hz}$

실습 4 : 서보 모터 위치 제어 4

- 구동 프로그램
 - main.c

```
#include <avr/io.h>      // AVR 입출력에 대한 헤더 파일
#include <util/delay.h>  // delay 함수사용을 위한 헤더파일
void Move_ServoMotor(signed char degree);
int main(void) {
    DDRE = 0x08;          // SERVO 핀을 출력 포트로 설정
    // 8분주, PWM, Phase and Frequency Correct, TOP : ICR3
    // Waveform Generation Mode : 8
    // COMnA1=1,COMnA0=0,COMnB1=0,COMnB0=0
    // COMnC1=0,COMnC0=0,WGMn1=0,WGMn0=0
    TCCR3A = 0x80;
    // ICNCn=0,ICESn=0,Blink=0,WGMn3=1,WGMn2=0,CSn2=0,CSn1=1,CSn0=0
    TCCR3B = 0x12;
    TCCR3C = 0x00;
    ICR3 = 18432; // frequency = 14745600/2*8*18432 = 50Hz(20ms)
    TCNT3 = 0x0000;
    OCR3A = 0x0000;
```

실습 4 : 서보 모터 위치 제어 4

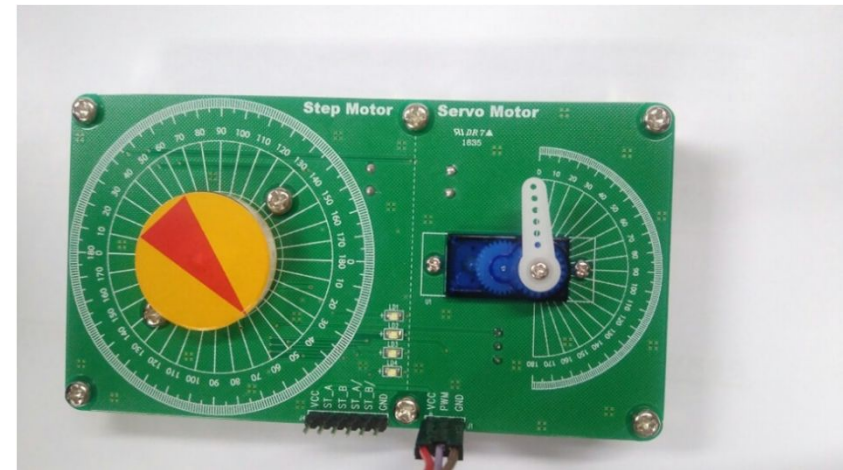
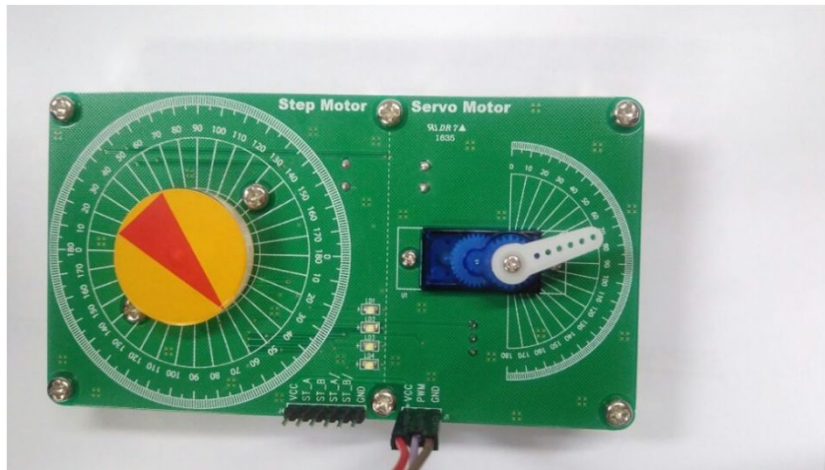
- 구동 프로그램
 - main.c

```
int main(void) {
    //...
    while (1) {
        Move_ServoMotor(90);
        Move_ServoMotor(0);
        Move_ServoMotor(-90);
    }
}

void Move_ServoMotor(signed char degree) {
    unsigned int _duty = 0, _pw = 0;
    // -90 = 2400,  0 = 1500,  +90 =  600
    _pw = 1500+degree*(-10);
    _duty = 0.9216 * _pw; // 0.9216 every 1us
    OCR3A = _duty;
    _delay_ms(1000);
}
```

실습 4 : 서보 모터 위치 제어 4

- 실행 결과
 - 서보 모터는 초기 상태인 0도로 움직이고 -90 -> 0 -> 90 -> 0 -> -90로 1초마다 반복적으로 동작

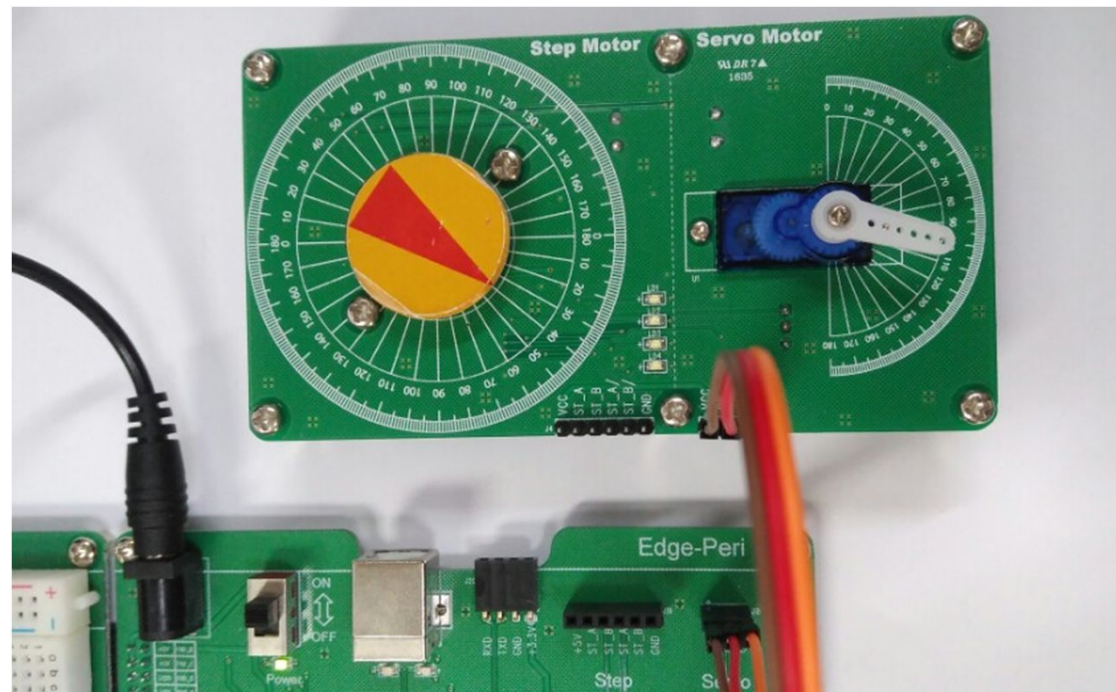


응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 실습 개요
 - ATmega128의 GPIO핀에 서보 모터의 제어신호를 연결하여, 서보 모터의 각도를 제어
 - 타이머의 PWM 기능을 사용하여 서보 제어신호를 만들고, UART통신을 이용하여 받은 값에 따라 서보를 제어하는 프로그램을 구현
- 실습 목표
 - 서보 모터의 동작원리를 이해
 - ATmega128A의 GPIO를 이용한 서보 모터 구동 방법 습득
 - ATmega128A의 타이머의 PWM 기능 사용법 습득

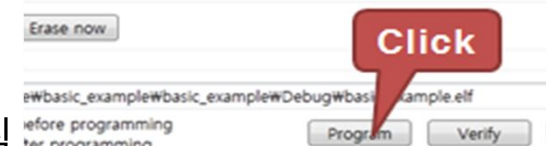
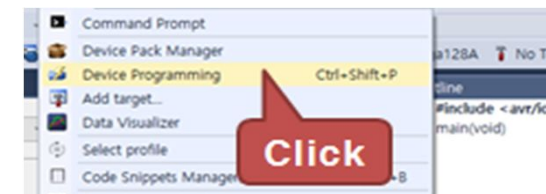
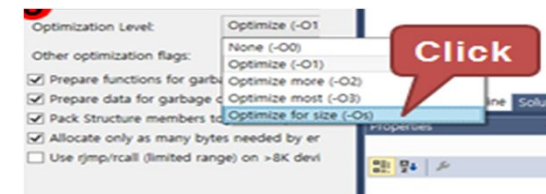
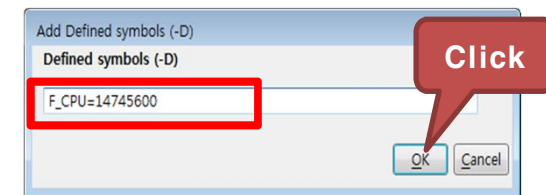
응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 실습 준비
 - 사용 보드를 다음 그림과 같이 연결
 - Edge-Peri보드의 +5V → Servo 모터 보드의 +5V
 - Edge-Peri보드의 PWM → Servo 모터 보드의 PWM
 - Edge-Peri보드의 GND → Servo 모터 보드의 GND



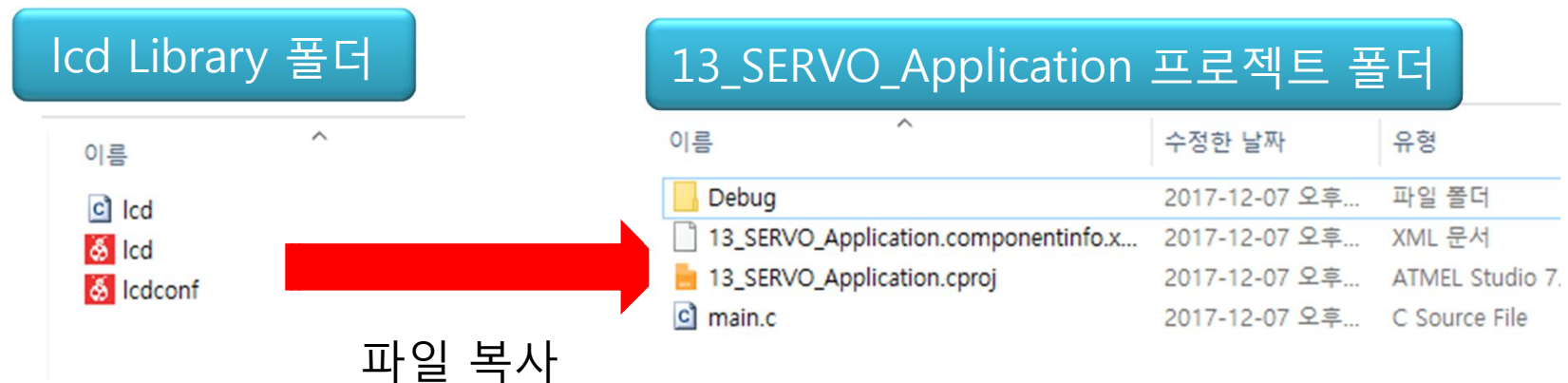
응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 예제 프로그램 작성 및 구동
 - Atmel Studio 실행
 - New Project 생성
 - Name : 13_SERVO_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



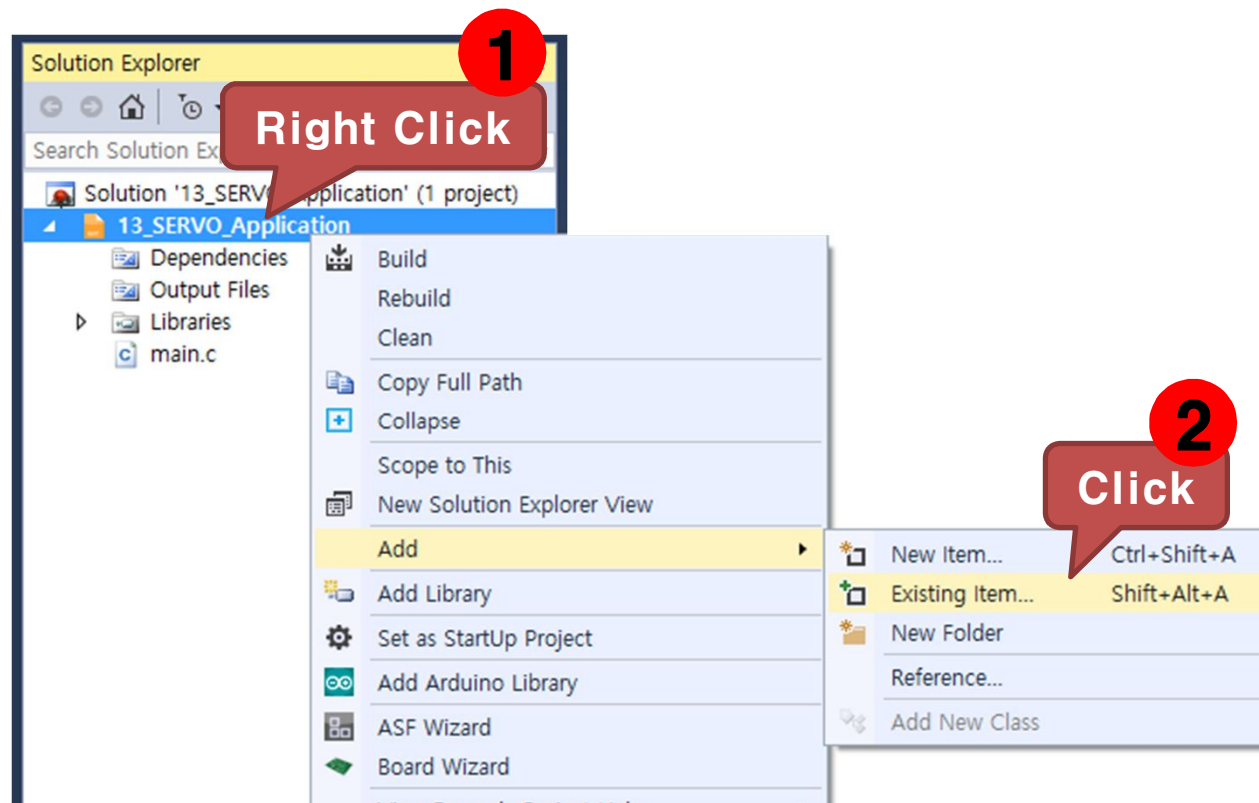
응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 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



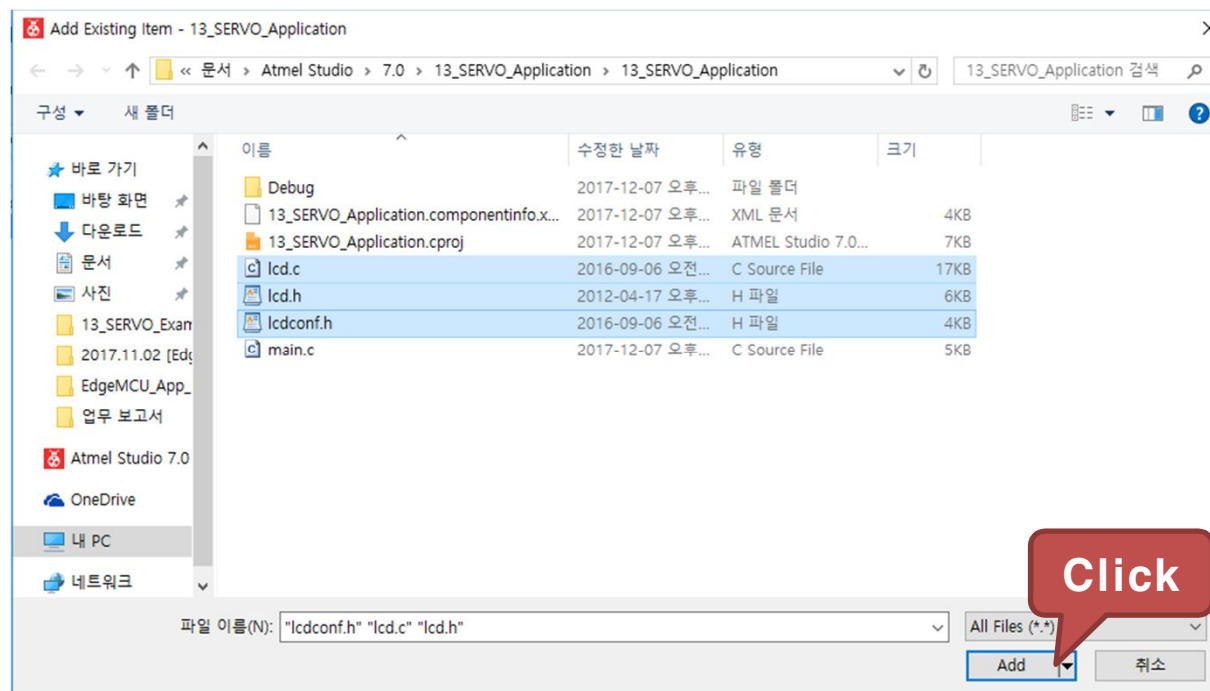
응용 : PC에서 정보를 입력받아 서보모터 제어하기

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



응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램 : Atmel Studio 에서 라이브러리 함수 추가하는 방법
 - lcd.c, lcd.h, lcdconf.h 파일을 각각 추가
 - 또는 한번에 세 파일을 동시에 추가
 - Ctrl 키를 누르고 파일 세개를 순서대로 클릭하여 세파일을 동시에 선택하고 한번에 추가



응용 : PC에서 정보를 입력받아 서보모터 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

volatile unsigned char buf[3], Disp_buf[3];
volatile signed char Degree = 0, Servo_flag = 0;

void Move_ServoMotor(signed char degree);
void putch(unsigned char data);
void putch_Str(char *str);
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{
    unsigned char i;

    DDRE = 0x0A; // Rx(입력 0), Tx(출력, 1), SERVO 핀을 출력 포트로
                // 설정 한다.

    lcdInit(); //Text LCD를 초기화

    UCSRA = 0x00;
    UCSRB = 0x98; // RX Complete Interrupt Enable, Rx 및 Tx enable,
    UCSRC = 0x06; // 비동기 방식, No Parity bit, 1 Stop bit

    UBRRH = 0x00;
    UBRRL = 0x07; // 115200 bps
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{ //...
    //8분주, PWM, Phase and Frequency Correct, TOP : ICR3
    //Waveform Generation Mode : 8
    TCCR3A = 0x80;          //COMnA1=0   COMnA0=0   COMnB1=0   COMnB0=0
    COMnC1=1 COMnC0=0 WGMn1=0 WGMn0=0
    TCCR3B = 0x12;          //ICNCn=0 ICESn=0 Blink=0 WGMn3=1 WGMn2=0
    CSn2=0 CSn1=1 CSn0=0
    TCCR3C = 0x00;

    ICR3 = 18432; //frequency = 14745600/2*8*18432 = 50Hz(20ms)

    TCNT3 = 0x0000;
    OCR3A = 0x0000;

    sei();
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{ //...
    lcdGotoXY(2,0);           //현재 커서위치를 첫번째 줄 세번째
    칸으로 이동한다.
    //첫번째 매개변수는 행을 의미하고, 두번째 매개변수는 열을 의미한다.
    lcdPrintData("Servo Motor",11); // "Servo Motor" 문자열을
    출력한다.

    lcdGotoXY(2,1);           //현재 커서위치를 두번째 줄 세번째
    칸으로 이동한다.
    lcdPrintData("Degree : 0",12); // "Degree : 0" 문자열을
    출력한다.

    Move_ServoMotor(0);

    while (1)
    {
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
int main(void)
{ //...
    if(Servo_flag)
    {
        Servo_flag = 0;

        // LCD에 제어 각도 출력
        lcdGotoXY(11,1);
        switch(Servo_flag)
        {
            case 1:
                lcdDataWrite(' ');
            case 2:
                lcdDataWrite(' ');
        }
    }
}
```


응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

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

    for(i=0; i<Servo_flag; i++)
        lcdDataWrite(Disp_buf[i]);

    // 입력 값에 따른 서보모터 제어
    Move_ServoMotor(Degree);
}
}
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
void Move_ServoMotor(signed char degree)
{
    unsigned int _duty = 0, _pw = 0;

    //-90 = 2400
    //  0 = 1500
    //+90 =  600
    _pw = 1500+degree*(-10);

    //0.9216 every 1us
    _duty = 0.9216 * _pw;

    OCR3A = _duty;

    _delay_ms(1000);
}
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

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

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

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
SIGNAL(USART0_RX_vect)
{
    cli();

    static unsigned char RX_count = 0;
    unsigned char receive_Data, i;

    receive_Data = UDR0;           // 수신 값 변수 receive_Data에 저장

    if(receive_Data != 0x0D)       // Enter가 입력 되지 않으면
    {
        if(RX_count == 0)         // 수신된 값이 처음 이며
        {
            if((receive_Data == '-') || ((receive_Data >= '0') &&
(receive_Data <= '9')))          // 숫자와 '-' 만 저장
            {
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
        buf[RX_count++] = receive_Data; // 변수 buf에 저장
        putchar(receive_Data);          // 수신 값 출력
    }
}
else
{
    if((receive_Data >= '0') && (receive_Data <= '9'))
    // 숫자만 저장
    {
        buf[RX_count++] = receive_Data; // 변수 buf에 저장
        putchar(receive_Data);          // 수신 값 출력
    }
}
}
else // 입력이 완료되면(Enter를 입력하면)
{
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
// 입력 값 출력
putch_Str("\n\r Input value : ");
for(i=0; i<RX_count; i++) {
    putch(buf[i]);
    Disp_buf[i] = buf[i];
}
putch_Str("\n\r ");
// 각도 값 계산
Degree = 0;                // 각도 초기화
if(buf[0] == '-') {        // 음수
    // 각도 계산
    for(i=1; i<RX_count; i++) {
        Degree *= 10;
        Degree += (buf[i] - '0');
    }
}
```

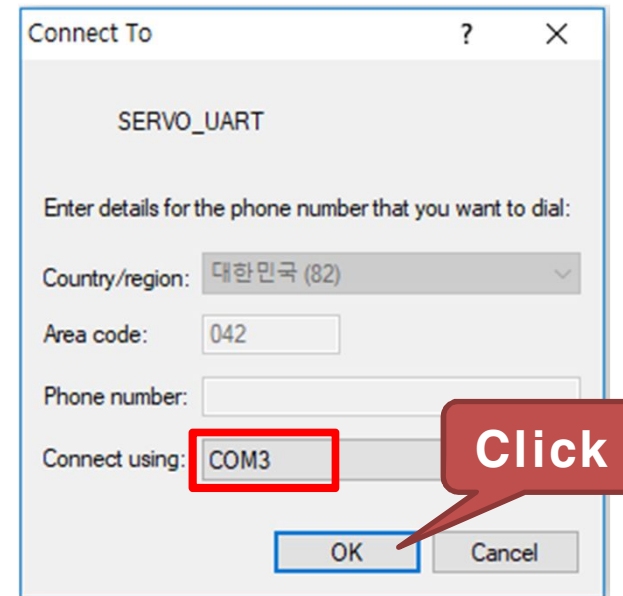
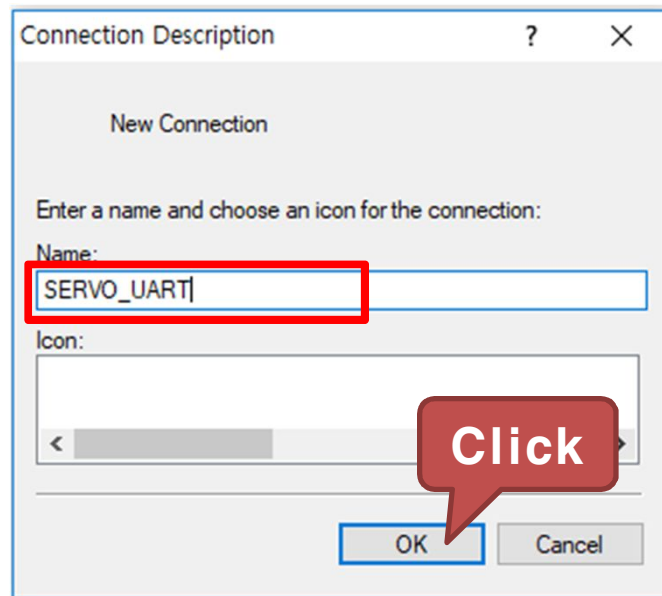
응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 구동 프로그램
 - main.c

```
        Degree = -Degree;
    }
    else
    {
        // 각도 계산
        for(i=0; i<RX_count; i++)
        {
            Degree *= 10;
            Degree += (buf[i] - '0');
        }
    }
    Servo_flag = RX_count;
    RX_count = 0;
}
sei();
}
```

응용 : PC에서 정보를 입력받아 서보모터 제어하기

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



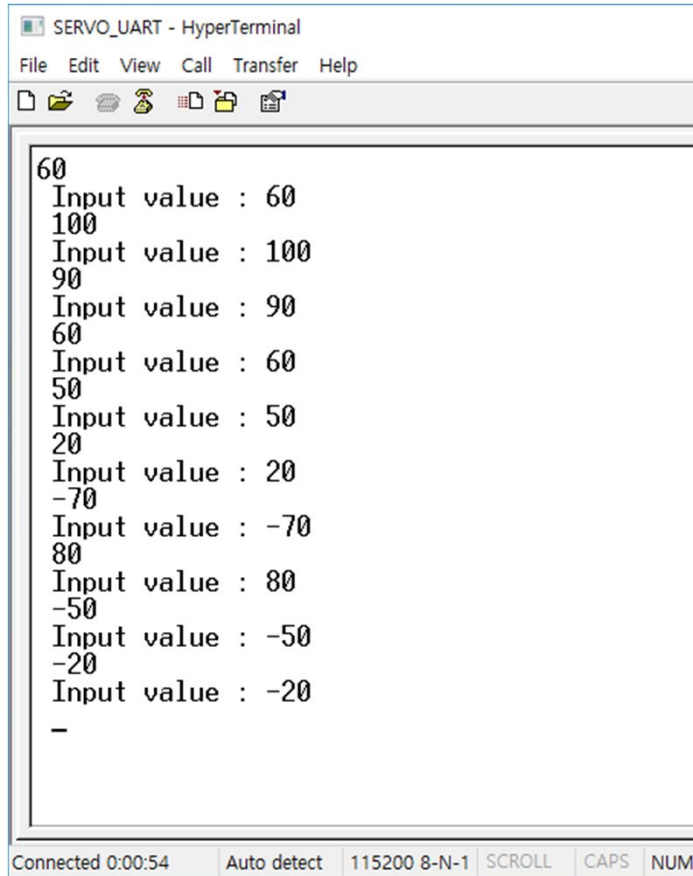
응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 하이퍼터미널 실행
 - 통신 설정을 그림과 같이하고, 확인을 클릭



응용 : PC에서 정보를 입력받아 서보모터 제어하기

- 실행 결과



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

60
Input value : 60
100
Input value : 100
90
Input value : 90
60
Input value : 60
50
Input value : 50
20
Input value : 20
-70
Input value : -70
80
Input value : 80
-50
Input value : -50
-20
Input value : -20
-

Connected 0:00:54 Auto detect 115200 8-N-1 SCROLL CAPS NUM
```

