STEP MOTOR

- 스텝 모터(Step Motor)
- 1상 여자 방식으로 스텝 모터 돌리기
- 2상 여자 방식으로 스텝 모터 돌리기
- 1-2상 여자 방식으로 스텝 모터 돌리기 1
- 1-2상 여자 방식으로 스텝 모터 돌리기 2
- 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

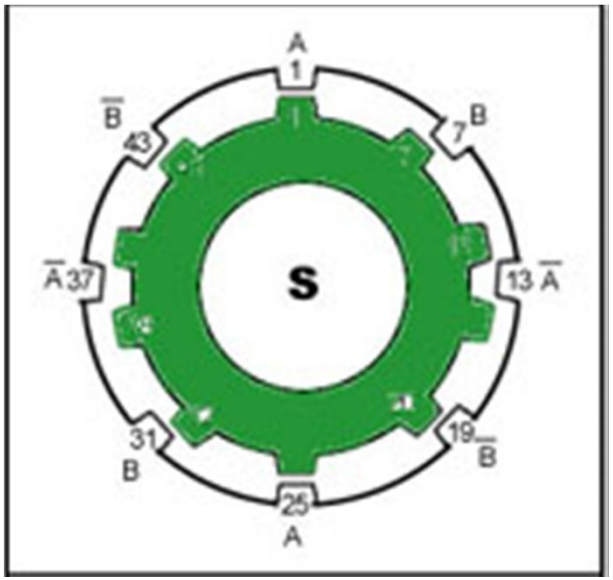
엣지아이랩

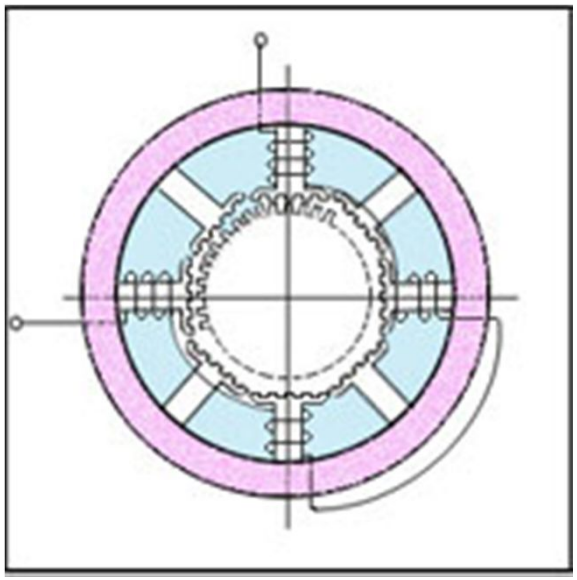
스텝 모터(Step Motor)

- 스테핑 모터(스텝모터, 펄스 모터)
  - 1902년 영국에서 처음 개발되어, 1960년대 일본에서 NC공작기계에 처음 도입
  - AC 서보, DC 서보 모터에 비하여 값이 싸고 정확한 각도제어에 유리
  - 기계적인 이동량을 정밀하게 제어하는 곳에 스테핑 모터가 많이 사용
  - 펄스에 의해 디지털적으로 제어하는 것이 가능하므로 마이크로컨트롤러에서 사용하기에 적합한 모터
  - 샤프트(축)의 위치를 검출하기 위한 별도의 피드백 신호없이 정해진 각도를 회전하고 (Open Loop Control), 상당히 높은 정확도로 정지할 수 있음
  - 다른 모터에 비해 정지시 매우 큰 유지 토그가 있기 때문에 전자 브레이크 등의 위치 유지 기구를 필요로 하지 않음
  - 회전 속도가 펄스 속도에 비례하므로 간편하게 제어 할 수 있음

스텝 모터(Step Motor)

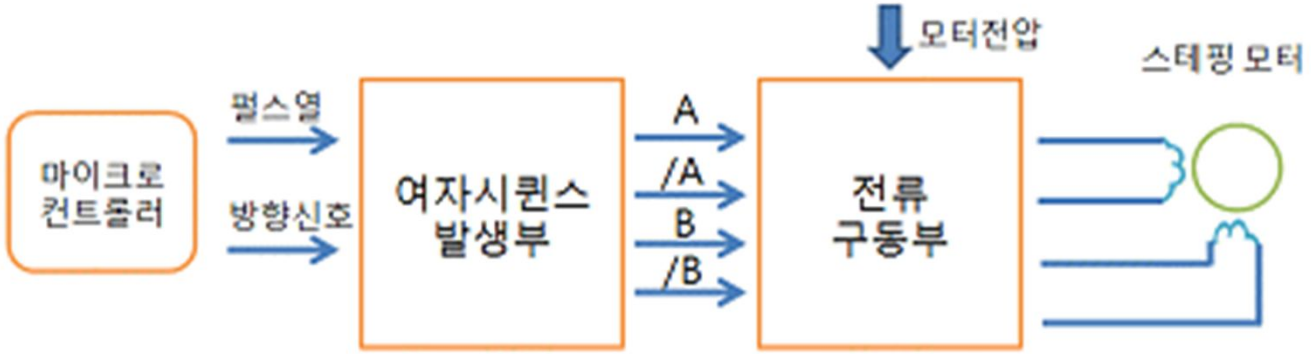
- 스텝 모터의 구조





스텝 모터(Step Motor)

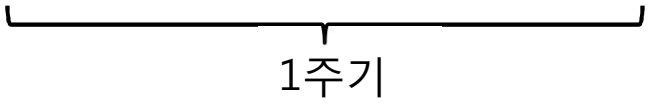
- 스텝 모터 구동방법
  - 스텝 모터 제어 회로
    - 여자 신호 발생부 : 펄스를 인가하여 각상의 여자 신호를 발생함
    - 구동 회로부 : 신호를 받아서 권선에 여자 전류를 흘려줌
  - 마이크로 컨트롤러
    - 방향 신호와 펄스열을 발생



스텝 모터(Step Motor)

- 1상 여자 방식 (Full step)
  - 구동 방법
    - 스텝 모터를 구동하기 위한 최소한의 구동 방법
    - STEP_A → STEP_/A → STEP_B → STEP_/B → STEP_A ⇒ 정회전
  - 특징
    - 1개의 코일 만을 차례로 여자 하는 방식
    - 소비 전력이 낮고 1스텝당 각 정밀도가 높음
    - 감쇠 진동이 크고 탈조 하기 쉬움

구분	1	2	3	4	5	6	7	8	9
A	1	0	0	0	1	0	0	0	1
B	0	1	0	0	0	1	0	0	0
/A	0	0	1	0	0	0	1	0	0
/B	0	0	0	1	0	0	0	1	0

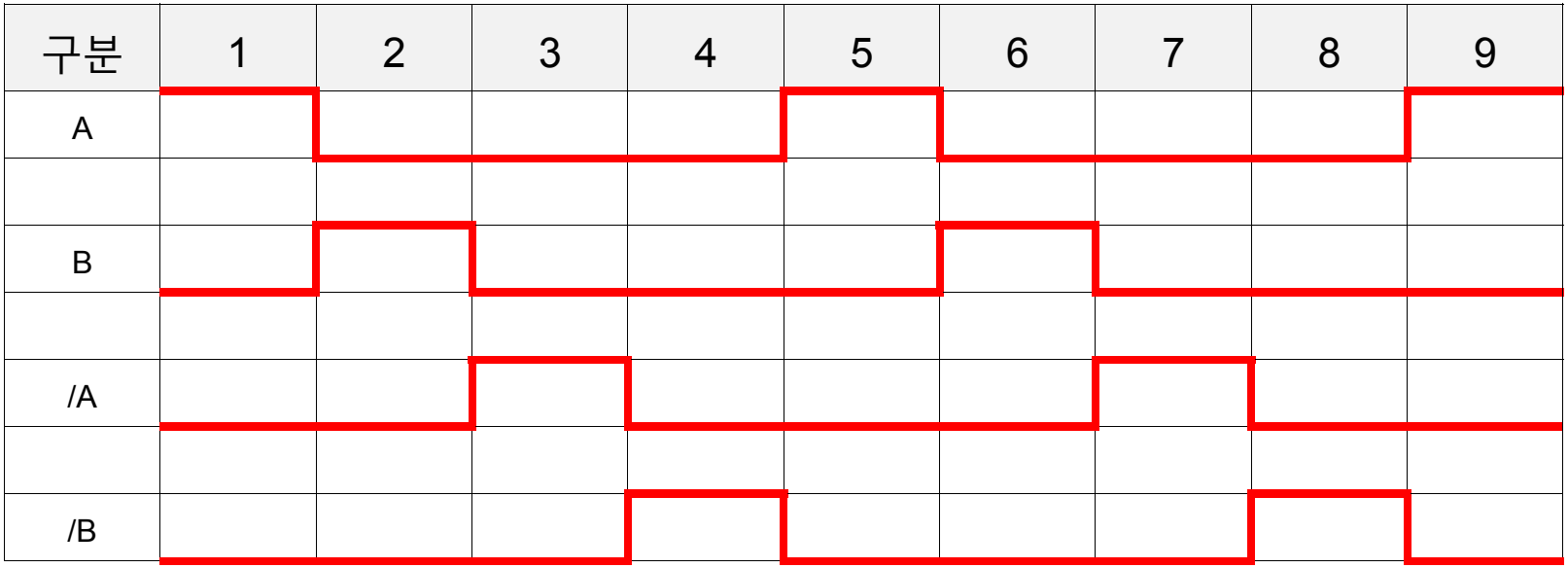


스텝 모터(Step Motor)

- 1상 여자 방식 (Full step)
  - 1상 여자 방식 동작 타이밍

구분	1	2	3	4	5	6	7	8	9
A	1	0	0	0	1	0	0	0	1
B	0	1	0	0	0	1	0	0	0
/A	0	0	1	0	0	0	1	0	0
/B	0	0	0	1	0	0	0	1	0

1주기



스텝 모터(Step Motor)

- 2상 여자 방식
  - 구동 방법
    - 항상 2상이 여자 되므로 기동 토크가 주어져 난조가 일어나기 어렵고 항상 2개의 상에서 전류가 흐르게 하도록 해야 함
    - STEP_A,STEP_/A → STEP_/A, STEP_B → STEP_B, STEP_/B → STEP_/B,STEP_A ⇒ 정회전
  - 특징
    - 2개의 코일을 동시에 여자 하는 방식
    - 1상 여자 구동에 비해 2배의 전류가 필요하지만 토크가 크고 감쇠진동이 적음
    - 주파수(damping)특성이 양호하여 가장 널리 이용되는 방법

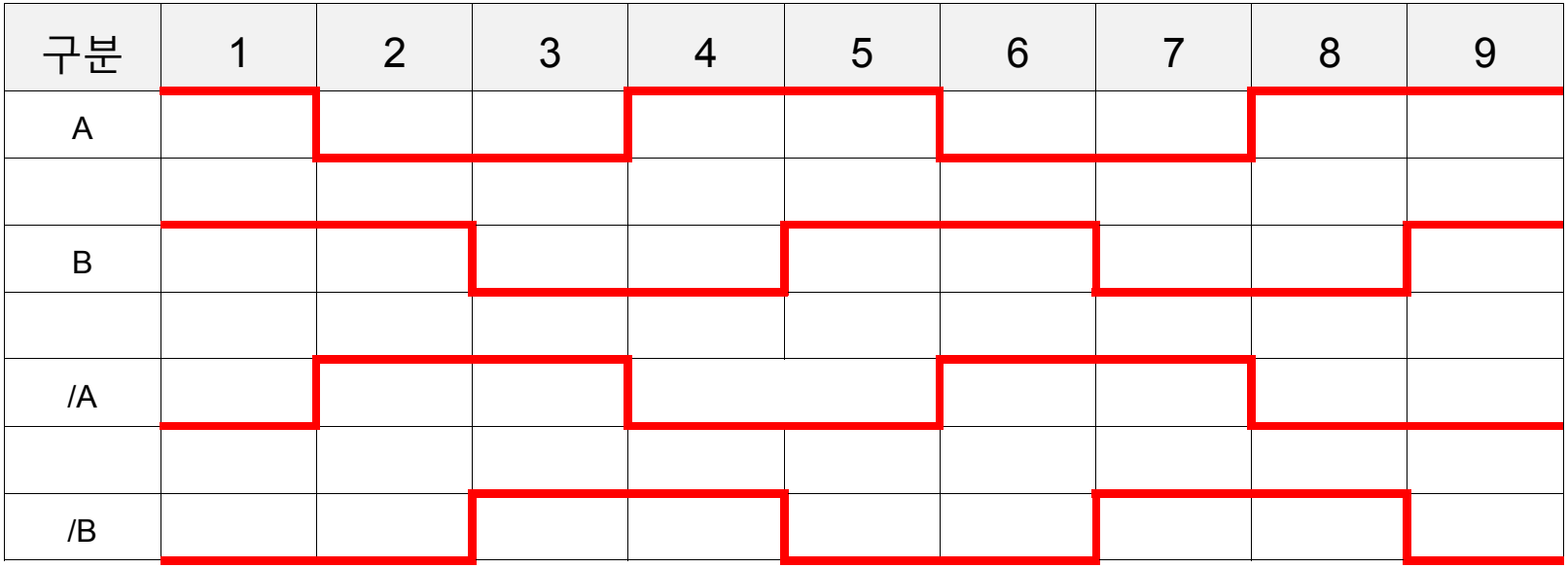
구분	1	2	3	4	5	6	7	8	9
A	1	0	0	1	1	0	0	1	1
B	1	1	0	0	1	1	0	0	1
/A	0	1	1	0	0	1	1	0	0
/B	0	0	1	1	0	0	1	1	0

스텝 모터(Step Motor)

- 2상 여자 방식
  - 2상 여자 방식 동작 타이밍

구분	1	2	3	4	5	6	7	8	9
A	1	0	0	1	1	0	0	1	1
B	1	1	0	0	1	1	0	0	1
/A	0	1	1	0	0	1	1	0	0
/B	0	0	1	1	0	0	1	1	0

1주기



스텝 모터(Step Motor)

- 1-2상 여자 방식
  - 구동 방법
    - 하나의 상과 두개의 상에 교대로 전류를 흐르게 하는 방식
    - 스텝각은 1,2상 여자 방식의 1/2 이며 응답 스텝 비율은 1,2 상의 2배
  - 특징
    - 1상 여자 구동에 비해 1.5배의 전류가 필요
    - 1펄스에 대한 스텝 각은 1상 여자와 2상 여자에 의한 스텝 각의 1/2
    - 각도를 정밀하게 제어하는 경우에 사용

구분	1	2	3	4	5	6	7	8	9
A	1	1	0	0	0	0	0	1	1
B	0	1	1	1	0	0	0	0	0
/A	0	0	0	1	1	1	0	0	0
/B	0	0	0	0	0	1	1	1	0

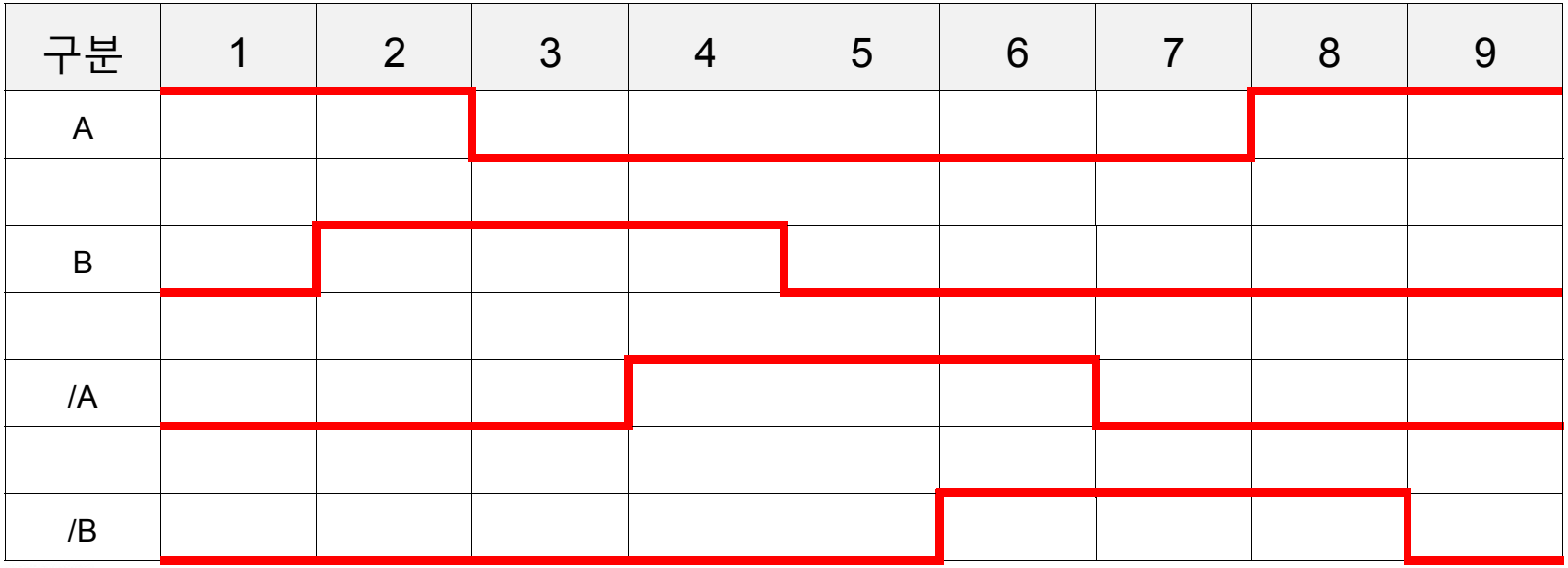
1주기

스텝 모터(Step Motor)

- 1-2상 여자 방식
  - 1-2상 여자 방식 동작 타이밍

구분	1	2	3	4	5	6	7	8	9
A	1	1	0	0	0	0	0	1	1
B	0	1	1	1	0	0	0	0	0
/A	0	0	0	1	1	1	0	0	0
/B	0	0	0	0	0	1	1	1	0

1주기



실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

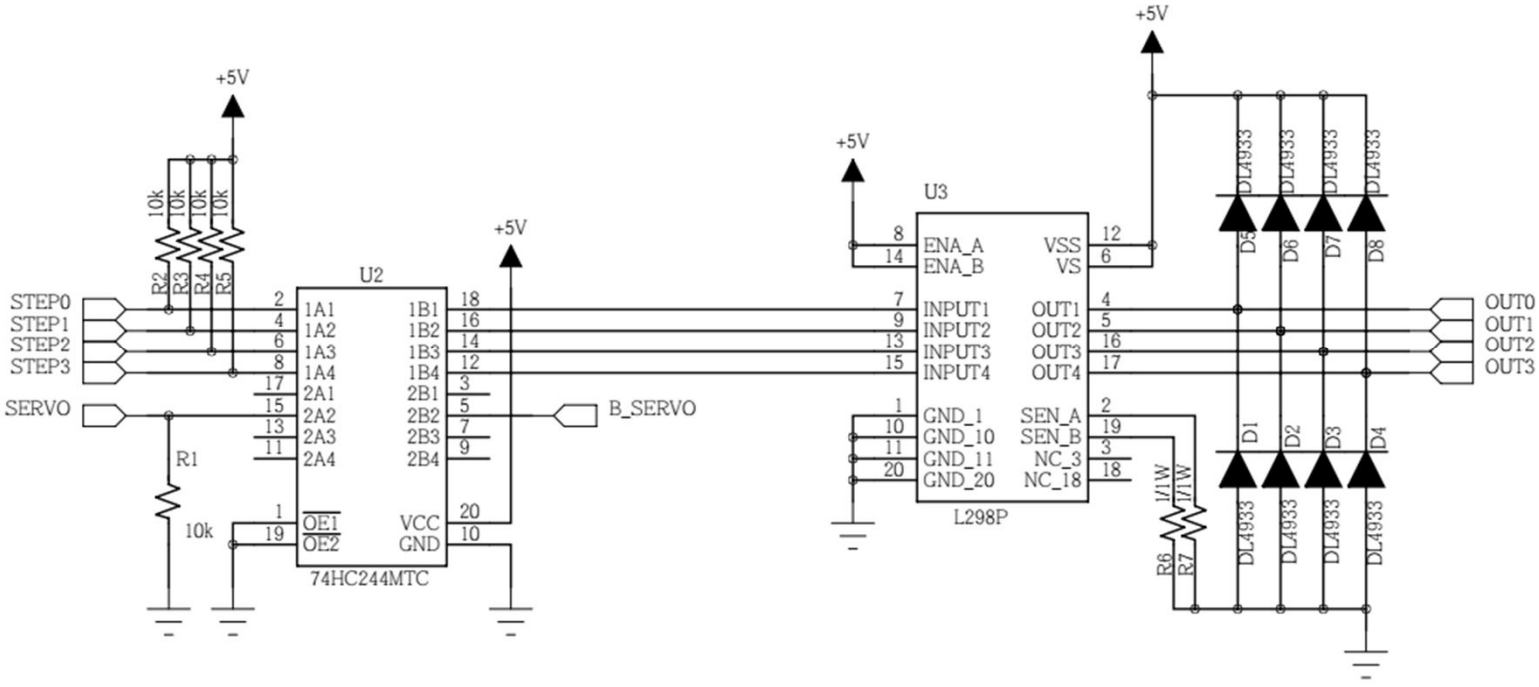
edgeILAB

- 실습 개요
  - ATMega128의 GPIO핀에 스텝 모터의 제어 신호를 연결하여, 스텝 모터를 회전 시키기
  - 1상 여자 방식을 사용하여 반 시계방향으로 무한히 회전
  - 본 실험에서 사용하고 있는 Step 모터는 1상, 2상 여자 방식일 때 펄스당 7.5도, 1-2상 여자 방식일 때 펄스당 3.75도 회전

- 실습 목표
  - 스텝 모터의 동작원리를 이해
  - ATmega128A의 GPIO를 이용한 스텝 모터 구동 방법 습득

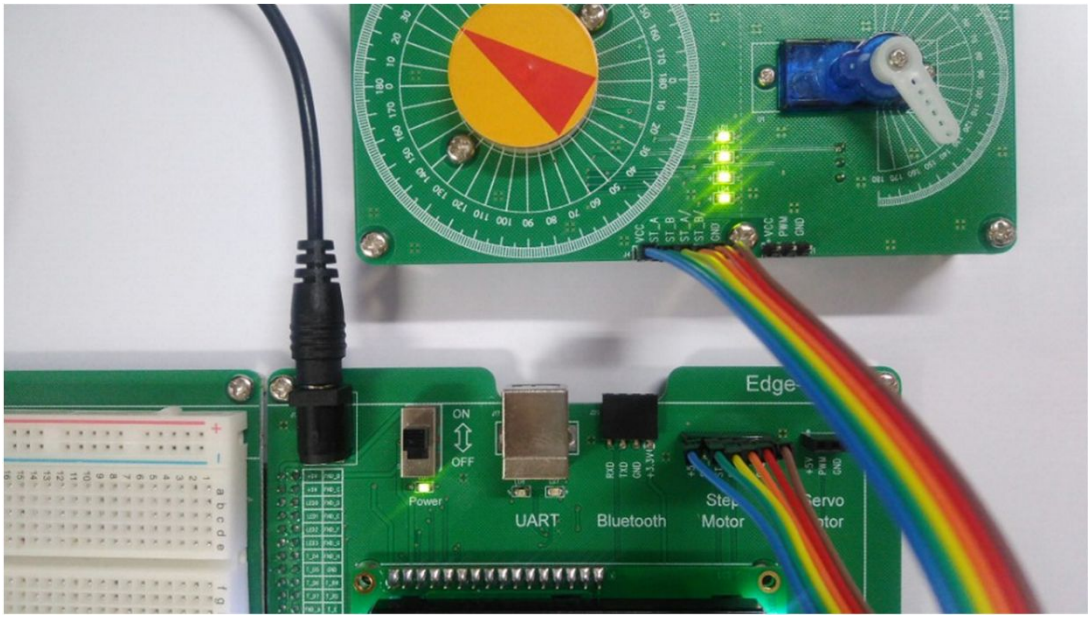
실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

- 사용 모듈
  - 스텝 모터 모듈의 회로
    - 제어 신호
      - OUT_0 → STEP_A, OUT_1 → STEP_B
      - OUT_2 → STEP_/A, OUT_3 → STEP_/B



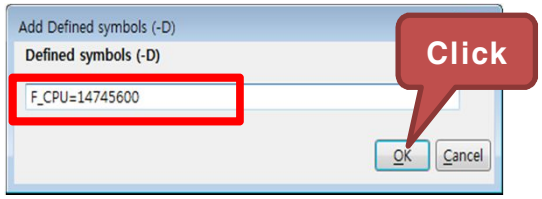
실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

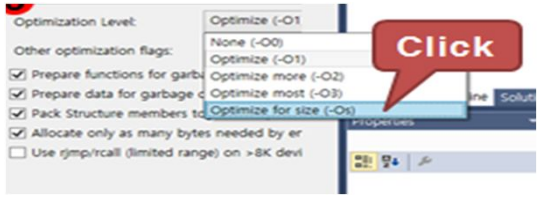
- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-Peri보드의 +5V → Step 모터 보드의 +5V
    - Edge-Peri보드의 ST_A → Step 모터 보드의 ST_A
    - Edge-Peri보드의 ST_B → Step 모터 보드의 ST_B
    - Edge-Peri보드의 ST_/A → Step 모터 보드의 ST_/A
    - Edge-Peri보드의 ST_/B → Step 모터 보드의 ST_/B
    - Edge-Peri보드의 GND → Step 모터 보드의 GND

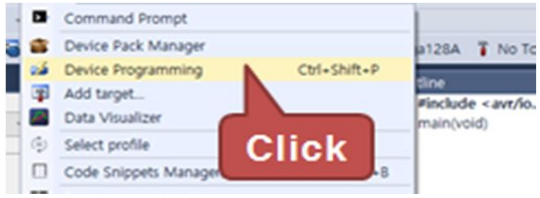


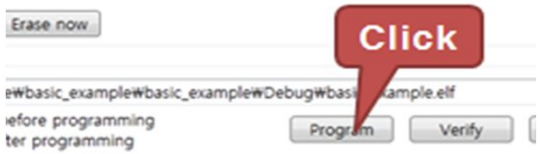
실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 11_Step1Phase_Example, 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









실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

- 구동 프로그램 : 사전지식
  - 스텝 모터의 1상 여자 방식 신호 만들기
    - D(7:4)에 차례대로, 0x10, 0x20, 0x40, 0x80을 한 주기로 하여 계속 전송
    - 반대방향으로 돌리고 싶다면, 0x80, 0x40, 0x20, 0x10순서로 데이터를 전송

1상 여자 방식 제어 신호(반 시계 방향)

신호	1	2	3	4	5	6	7	8	9
PD4(A)	1	0	0	0	1	0	0	0	1
PD5(B)	0	1	0	0	0	1	0	0	0
PD6(/A)	0	0	1	0	0	0	1	0	0
PD7(/B)	0	0	0	1	0	0	0	1	0
	0x10	0x20	0x40	0x80	0x10	0x20	0x40	0x80	0x10

실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

edgeILAB

- 구동 프로그램 : 사전지식
  - GPIO 사용
    - OUT_0(STEP_A) → PORTD(PD4)
    - OUT_1(STEP_B) → PORTD(PD5)
    - OUT_2(STEP_/A) → PORTD(PD6)
    - OUT_3(STEP_/B) → PORTD(PD7)
  - 시간 지연 함수 사용
    - 1상 여자방식 신호를 만들기 위해서 D(7:4)에 데이터를 보내는 시간간격을 일정하게 유지해야 함
    - 시간 지연 함수를 이용하여 10ms 마다 상을 변화

실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

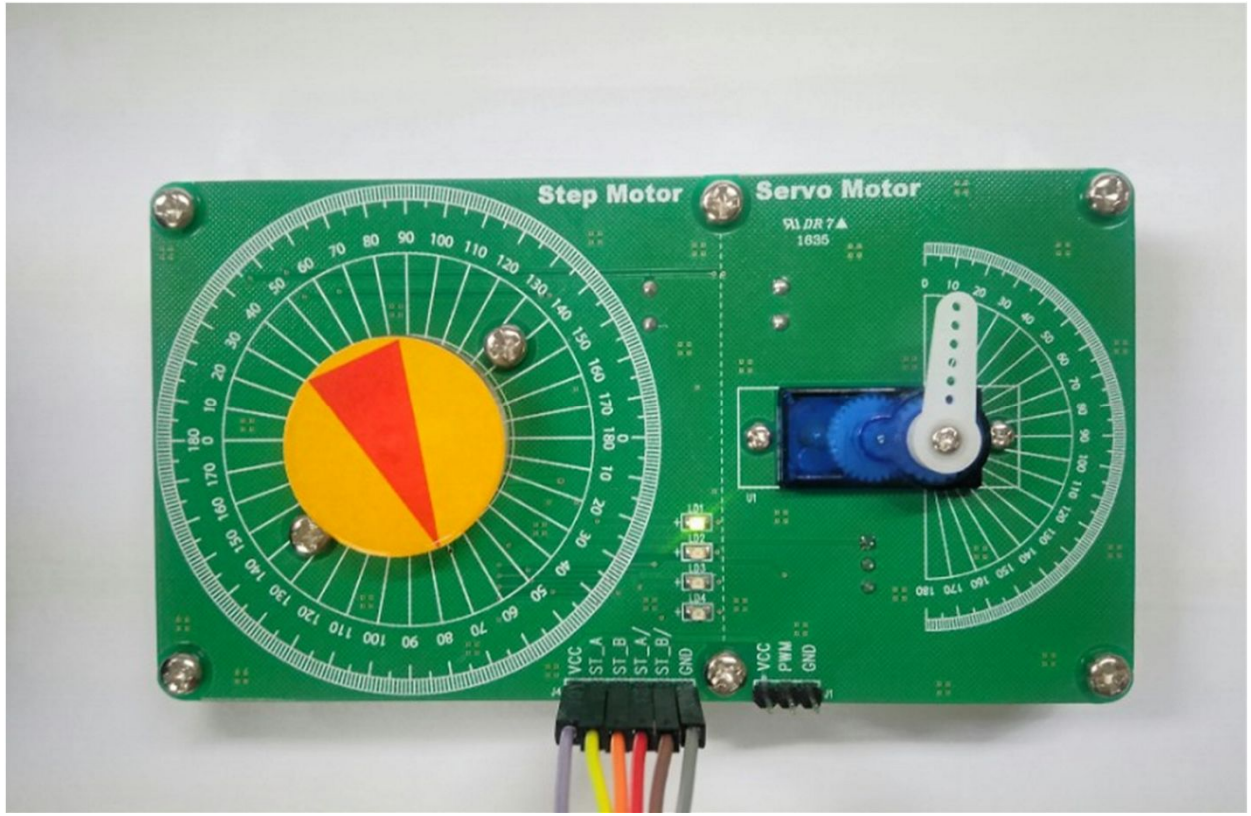
edgeILAB

- 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) {
  DDRB = 0x20;           // MOTOR1_EN를 출력 포트로 설정
  DDRD = 0xF0;           // STEP 0~STEP 3을 출력 포트로 설정
  PORTB &= ~0x20;        // M1 Disable, DC 모터 정지
  while (1) {
    PORTD = 0x10;                // 1 step
    _delay_ms(10);
    PORTD = 0x20;                // 2 step
    _delay_ms(10);
    PORTD = 0x40;                // 3 step
    _delay_ms(10);
    PORTD = 0x80;                // 4 step
    _delay_ms(10);
  }
}
```

실습 1 : 1상 여자 방식으로 스텝 모터 돌리기

- 실행 결과
  - 스텝 모터가 1상 여자 방식으로 반시계방향 회전



실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

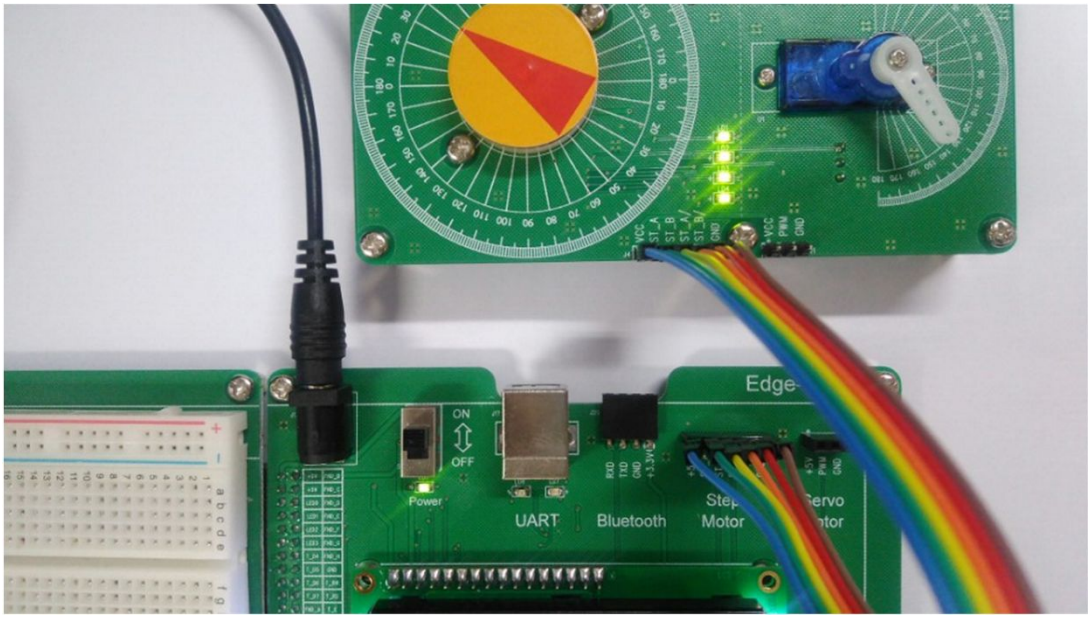
edgeILAB

- 실습 개요
  - ATMega128의 GPIO핀에 스텝 모터의 제어 신호를 연결하여, 스텝 모터를 회전 시키기
  - 2상 여자 방식을 사용하여 시계방향으로 1바퀴 돌고 1초 정지 시키는 것을 무한히 반복
  - 본 실험에서 사용하고 있는 Step 모터는 1상, 2상 여자 방식일 때 펄스당 7.5도, 1-2상 여자 방식일 때 펄스당 3.75도 회전

- 실습 목표
  - 스텝 모터의 동작원리를 이해
  - ATmega128A의 GPIO를 이용한 스텝 모터 구동 방법 습득

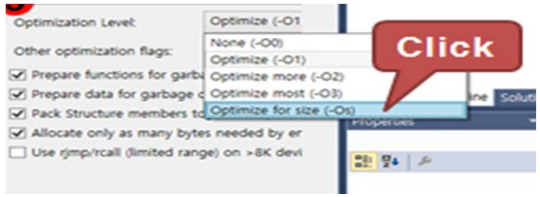
실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

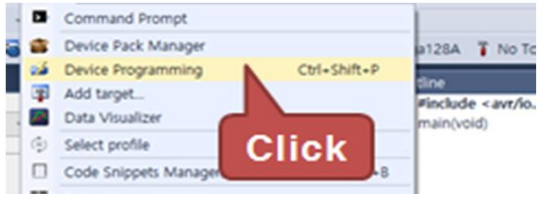
- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-Peri보드의 +5V → Step 모터 보드의 +5V
    - Edge-Peri보드의 ST_A → Step 모터 보드의 ST_A
    - Edge-Peri보드의 ST_B → Step 모터 보드의 ST_B
    - Edge-Peri보드의 ST_/A → Step 모터 보드의 ST_/A
    - Edge-Peri보드의 ST_/B → Step 모터 보드의 ST_/B
    - Edge-Peri보드의 GND → Step 모터 보드의 GND

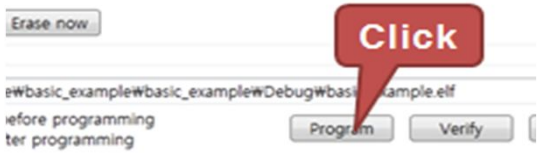


실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : Step2Phase, 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







실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

- 구동 프로그램 : 사전지식
  - 스텝 모터의 2상 여자 방식 신호 만들기
    - 2상 여자 방식이므로 스텝 모터는 펄스당 7.5도가 회전
      - 시계 방향일 경우는 PORTD에 0x30 → 0x90 → 0xC0 → 0x60 → … 이러한 명령이 총 360/7.5 = 48, 즉 48개의 펄스가 입력되어야함
      - 한 주기가 4개의 명령으로 이루어 지므로 (0x30 → 0x90 → 0xC0 → 0x60) 12 주기의 명령이 PORTD로 입력되어야 스텝 모터는 1회전 함

2상 여자 방식 제어 신호(반 시계 방향)

신호	1	2	3	4	5	6	7	8	9
PD4(A)	1	0	0	1	1	0	0	1	1
PD5(B)	1	1	0	0	1	1	0	0	1
PD6(/A)	0	1	1	0	0	1	1	0	0
PD7(/B)	0	0	1	1	0	0	1	1	0
	0x30	0x60	0xC0	0x90	0x30	0x60	0xC0	0x90	0x30

실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

edgeILAB

- 구동 프로그램 : 사전지식
  - GPIO 사용
    - OUT_0(STEP_A) → PORTD(PD4)
    - OUT_1(STEP_B) → PORTD(PD5)
    - OUT_2(STEP_/A) → PORTD(PD6)
    - OUT_3(STEP_/B) → PORTD(PD7)
  - 시간 지연 함수 사용
    - 2상 여자 방식 신호의 4 스텝을 12번 실행 시켜 48 스텝을 진행
    - 2상 여자 방식 신호를 만들기 위해서 D(7:4)에 데이터를 보내는 시간간격을 일정하게 유지해야 함
    - 시간 지연 함수를 이용하여 10ms 마다 상을 변화

실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

- 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;

  DDRB = 0x20;           // MOTOR1_EN 를 출력 포트로 설정
  DDRD = 0xF0;           // STEP0 ~ STEP3을 출력 포트로 설정
  PORTB &= ~0x20;        // M1 Disable, DC 모터 정지
```

실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

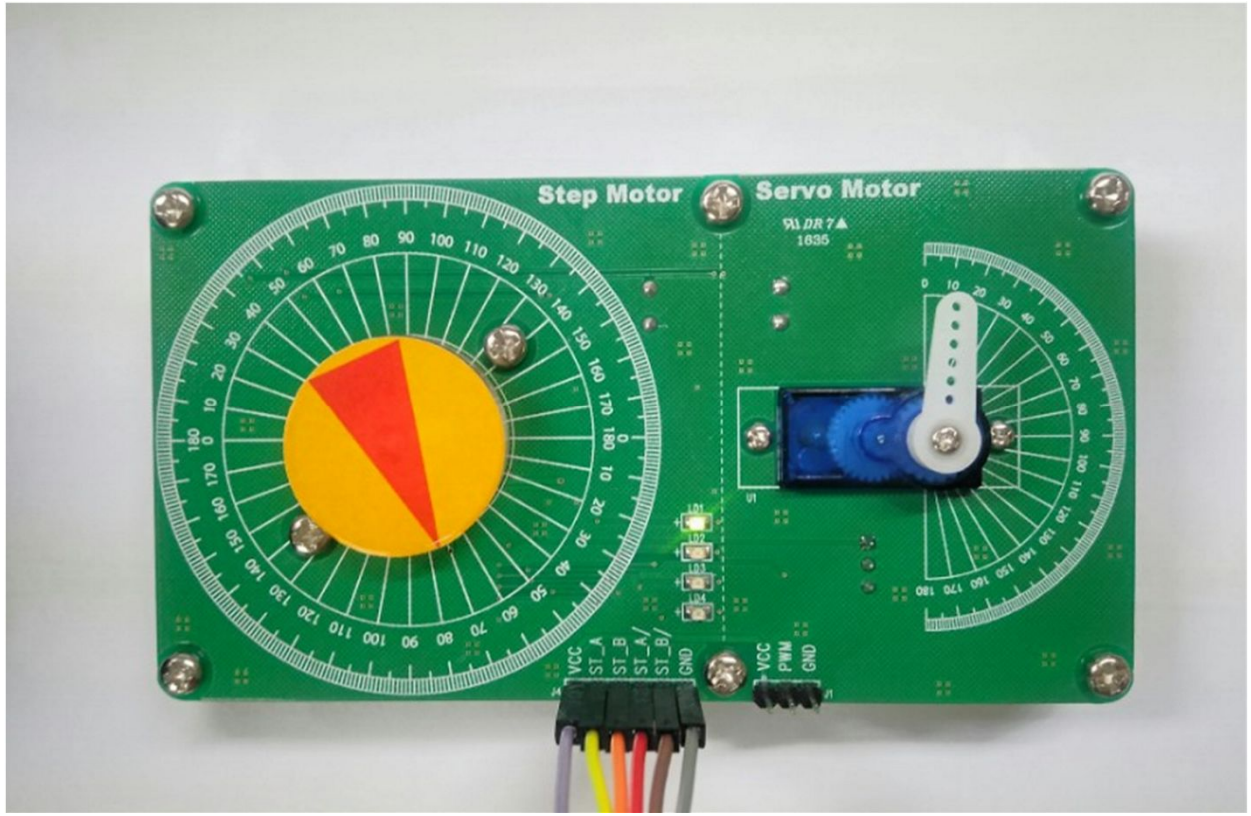
edgeILAB

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  //...
  while (1) {
    for ( i = 0; i < 12 ; i++ ) // 48 스텝 실행
    {
      PORTD = 0x30;      // 1 step
      _delay_ms(10);
      PORTD = 0x90;      // 2 step
      _delay_ms(10);
      PORTD = 0xC0;      // 3 step
      _delay_ms(10);
      PORTD = 0x60;      // 4 step
      _delay_ms(10);
    }
    _delay_ms(1000);
  }
}
```

실습 2 : 2상 여자 방식으로 스텝 모터 돌리기

- 실행 결과
  - 스텝 모터가 2상 여자 방식으로 시계 방향으로 한바퀴 회전 후 1초간 대기



실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

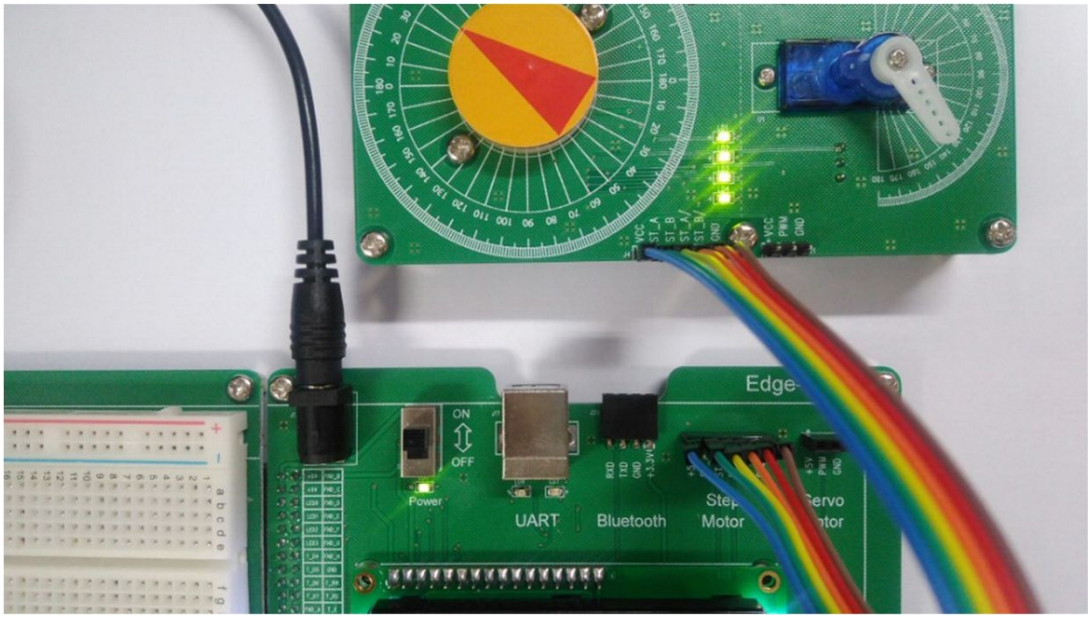
edgeILAB

- 실습 개요
  - ATMega128의 GPIO핀에 스텝 모터의 제어 신호를 연결하여, 스텝 모터를 회전 시키기
  - 1-2상 여자 방식을 사용하여 시계방향으로 90도 회전하고 1초 정지 시키는 것을 무한히 반복
  - 본 실험에서 사용하고 있는 Step 모터는 1상, 2상 여자 방식일 때 펄스당 7.5도, 1-2상 여자 방식일 때 펄스당 3.75도 회전

- 실습 목표
  - 스텝 모터의 동작원리를 이해
  - ATmega128A의 GPIO를 이용한 스텝 모터 구동 방법 습득

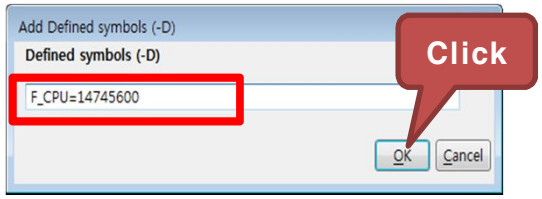
실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

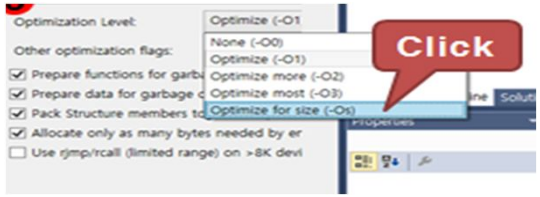
- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-Peri보드의 +5V → Step 모터 보드의 +5V
    - Edge-Peri보드의 ST_A → Step 모터 보드의 ST_A
    - Edge-Peri보드의 ST_B → Step 모터 보드의 ST_B
    - Edge-Peri보드의 ST_/A → Step 모터 보드의 ST_/A
    - Edge-Peri보드의 ST_/B → Step 모터 보드의 ST_/B
    - Edge-Peri보드의 GND → Step 모터 보드의 GND

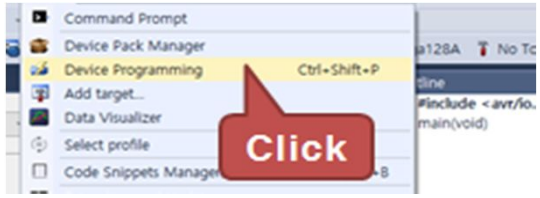


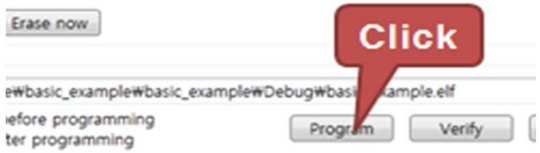
실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 12_Step12Phase_Example_01, 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









실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

- 구동 프로그램 : 사전지식
  - 스텝 모터의 1-2상 여자 방식 신호 만들기
    - 1-2상 여자 방식이므로 스텝 모터는 펄스당 3.75도가 회전
      - 반 시계 방향일 경우는 PORTD에 0x10 → 0x30 → 0x20 → 0x60 → 0x40 → 0xC0 → 0x80 → 0x90 … 이러한 명령이 총 360/3.75 = 96, 즉 96개의 펄스가 입력되어야 함
      - 90도만 회전 하려면 96/4 = 24, 즉 24개의 스텝이 필요

1-2상 여자 방식 제어 신호(반 시계 방향)

신호	1	2	3	4	5	6	7	8	9
PD4(A)	1	1	0	0	0	0	0	1	1
PD5(B)	0	1	1	1	0	0	0	0	0
PD6(/A)	0	0	0	1	1	1	0	0	0
PD7(/B)	0	0	0	0	0	1	1	1	0
	0x10	0x30	0x20	0x60	0x40	0xC0	0x80	0x90	0x10

실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

edgeILAB

- 구동 프로그램 : 사전지식
  - GPIO 사용
    - OUT_0(STEP_A) → PORTD(PD4)
    - OUT_1(STEP_B) → PORTD(PD5)
    - OUT_2(STEP_/A) → PORTD(PD6)
    - OUT_3(STEP_/B) → PORTD(PD7)
  - 시간 지연 함수 사용
    - 1-2상 여자 방식 신호의 스텝을 24 스텝 실행 시켜 킴
    - 1-2상 여자 방식 신호를 만들기 위해서 D(7:4)에 데이터를 보내는 시간간격을 일정하게 유지
    - 시간 지연 함수를 이용하여 10ms 마다 상을 변화

실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

- 구동 프로그램
  - main.c 코드 작성

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

// 1-2상 여자 방식 제어 신호를 저장 하고 있는 배열
unsigned char Step[8] = {0x90,0x80,0xC0,0x40,0x60,0x20,0x30,0x10};

int main(void)
{
  unsigned char i, t = 0;

  DDRB = 0x20;           // MOTOR1_EN 를 출력 포트로 설정
  DDRD = 0xF0;           // STEP0 ~ STEP3을 출력 포트로 설정
  PORTB &= ~0x20;        // M1 Disable, DC 모터 정지
```

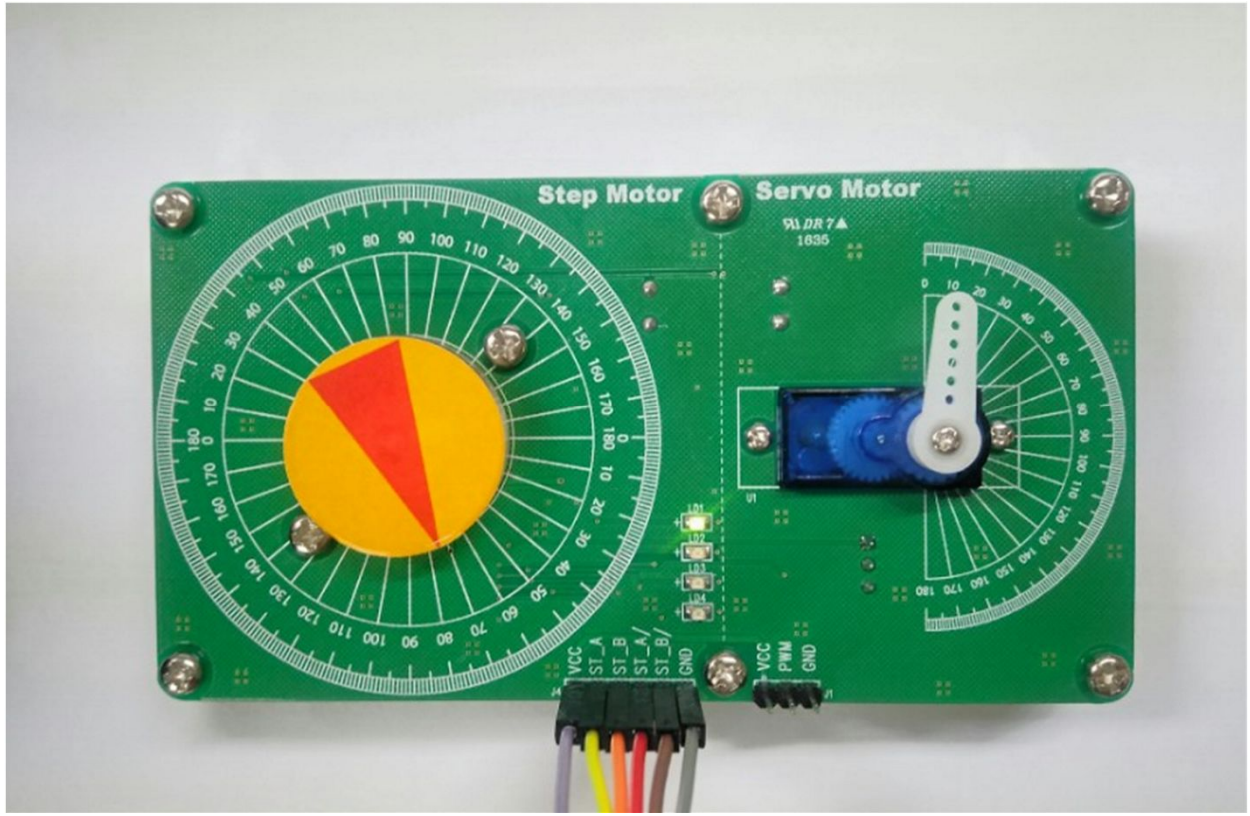
실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  //...
  while (1) {
    // 명령당 3.75도 회전, 90도 회전하려면 총 24개의 명령이 필요
    for ( i = 0; i < 24 ; i++ )
    {
      PORTD = Step[t];  // 한 스텝 실행
      t++;                        // 다음 스텝 실행을 위해 t 증가
      if(t > 7) t = 0;            // 8 스텝을 초과하지 않도록 초기화
      _delay_ms(10);
    }
    _delay_ms(1000);
  }
}
```

실습 3 : 1-2상 여자 방식으로 스텝 모터 돌리기 1

- 실행 결과
  - 스텝 모터가 1-2상 여자 방식으로 시계방향으로 90도 회전한 후 1초간 대기



실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

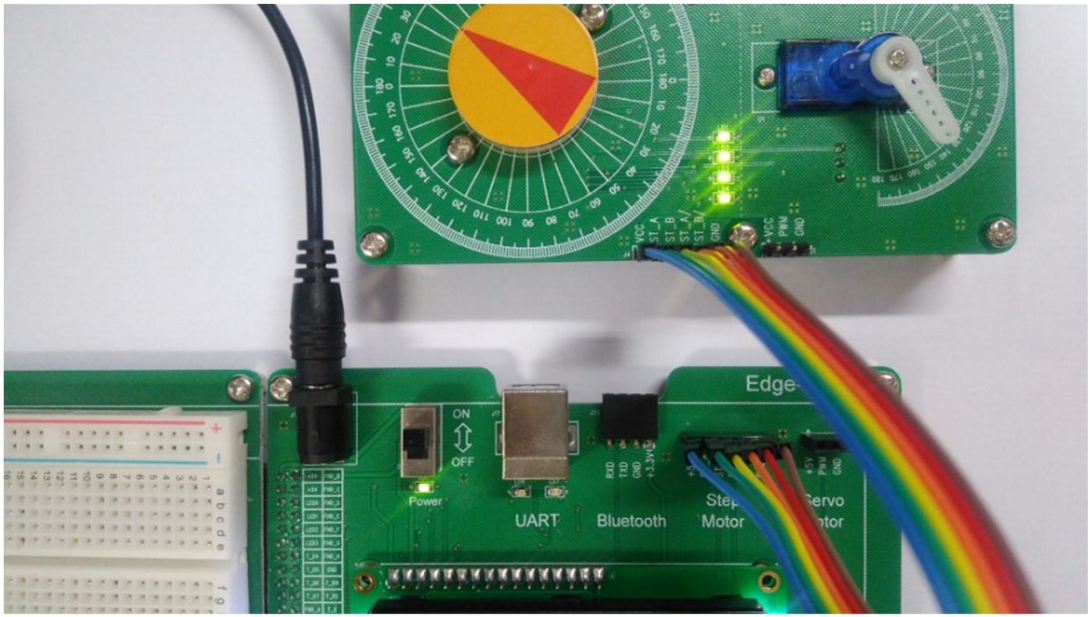
edgeILAB

- 실습 개요
  - ATMega128의 GPIO핀에 스텝 모터의 제어 신호를 연결하여, 스텝 모터를 회전 시키기
  - 타이머를 이용하여 2초 마다 모터가 방향을 전환
  - 스텝 모터의 구동 방식은 1-2상 여자 방식을 쓰도록 함
  - 앞에서 배운 타이머의 기능을 복합적으로 이용

- 실습 목표
  - 스텝 모터의 동작원리를 이해
  - ATmega128A의 GPIO를 이용한 스텝 모터 구동 방법 습득
  - 타이머, 인터럽트, GPIO 제어의 복합적인 프로그램 능력 배양

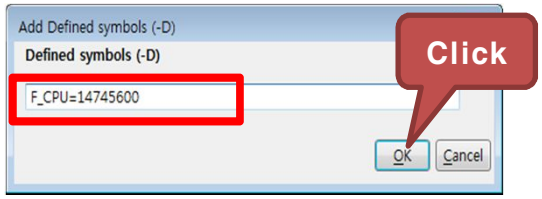
실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

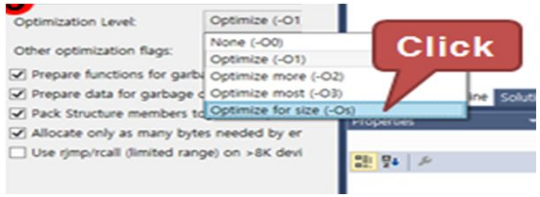
- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-Peri보드의 +5V → Step 모터 보드의 +5V
    - Edge-Peri보드의 ST_A → Step 모터 보드의 ST_A
    - Edge-Peri보드의 ST_B → Step 모터 보드의 ST_B
    - Edge-Peri보드의 ST_/A → Step 모터 보드의 ST_/A
    - Edge-Peri보드의 ST_/B → Step 모터 보드의 ST_/B
    - Edge-Peri보드의 GND → Step 모터 보드의 GND

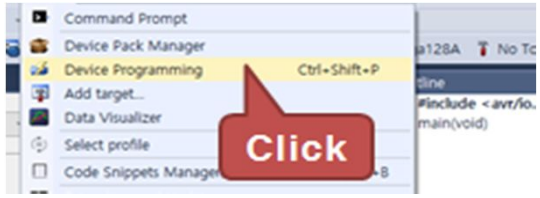


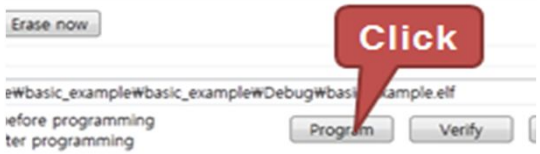
실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 12_Step12Phase_Example_02, 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









실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 구동 프로그램 : 사전지식
  - 스텝 모터의 1-2상 여자 방식 신호 만들기
    - 1-2상 여자 방식이므로 스텝 모터는 펄스당 3.75도가 회전
      - 반 시계 방향일 경우는 PORTD에 0x10 → 0x30 → 0x20 → 0x60 → 0x40 → 0xC0 → 0x80 → 0x90 … 이러한 명령을 한 주기로 반복해서 전송
      - 역 회전하려면 순서를 역순으로 바꾸어 전송

1-2상 여자 방식 제어 신호(반 시계 방향)

신호	1	2	3	4	5	6	7	8	9
PD4(A)	1	1	0	0	0	0	0	1	1
PD5(B)	0	1	1	1	0	0	0	0	0
PD6(/A)	0	0	0	1	1	1	0	0	0
PD7(/B)	0	0	0	0	0	1	1	1	0
	0x10	0x30	0x20	0x60	0x40	0xC0	0x80	0x90	0x10

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

edgeILAB

- 구동 프로그램 : 사전지식
  - GPIO 사용
    - OUT_0(STEP_A) → PORTD(PD4)
    - OUT_1(STEP_B) → PORTD(PD5)
    - OUT_2(STEP_/A) → PORTD(PD6)
    - OUT_3(STEP_/B) → PORTD(PD7)
  - 타이머 사용
    - 1-2상 여자 방식 신호를 만들기 위해서는 D(7:4)에 데이터를 보내는 시간간격을 일정하게 유지해야 함
    - 이를 위한 타이머가 필요, 여기서는 타이머/카운터 0를 사용하며, 타이머의 주기는 45KHz로 설정

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 구동 프로그램
  - main.c 코드 작성

```
#include <avr/io.h>                 // AVR 입출력에 대한 헤더 파일
#include <avr/interrupt.h>          // AVR 인터럽트에 대한 헤더파일
#define DIR_L 0
#define DIR_R 1

unsigned char timer0Cnt=0, mot_cnt=0;
volatile unsigned char dir=DIR_R;          // 처음 방향은 우측부터
                                           // 1-2 상 여자 값을 사용
unsigned char Step[] = {0x90, 0x80, 0xC0, 0x40,
                        0x60, 0x20, 0x30, 0x10};
SIGNAL(TIMER0_OVF_vect);           // Timer0 Overflow0 ISP
SIGNAL(TIMER1_OVF_vect);           // Timer1 Overflow1 ISP
```

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 구동 프로그램
  - main.c 코드 작성

```
int main(void) {
  DDRB = 0x20;            // MOTOR1_EN 를 출력 포트로 설정
  PORTB &= ~0x20;         // M1 Disable, DC 모터 정지
  DDRD = 0xF0;            // STEP0 ~ STEP3을 출력 포트로 설정
  TCCR0 = 0x07;
  TCNT0 = 112;  // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
  TCCR1A = 0;
  TCCR1B = 0x04;
  // (1/(14.7456Mhz/256prescaler))*1267=21.99ms => 22ms -> 45khz
  TCNT1H = 0xFB;          // 65536-1267 = 64269
  TCNT1L = 0x0D;          // 64269 = 0xFB0D
  TIMSK = 0x05;           // TOIE1, TOIE0 모두 '1'
  TIFR = 0x05;            // TOV1, TOV0 모두 '1'
  sei();
  while (1) {}
}
```

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(TIMER0_OVF_vect)
{
  cli();

  TCNT0 = 112;     // 256-144=112 -> 0.01초 마다 한번씩 인터럽트 발생
  timer0Cnt++;           // timer0Cnt 변수를 1 증가
  if(timer0Cnt == 200)   // 0.01s * 200 = 2s
                         // 1초를 얻기 위한 카운트 횟수
  {
    dir^=1;              // 방향 전환
    timer0Cnt=0;         // timer0Cnt 카운터 초기화
  }
  sei();
}
```

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

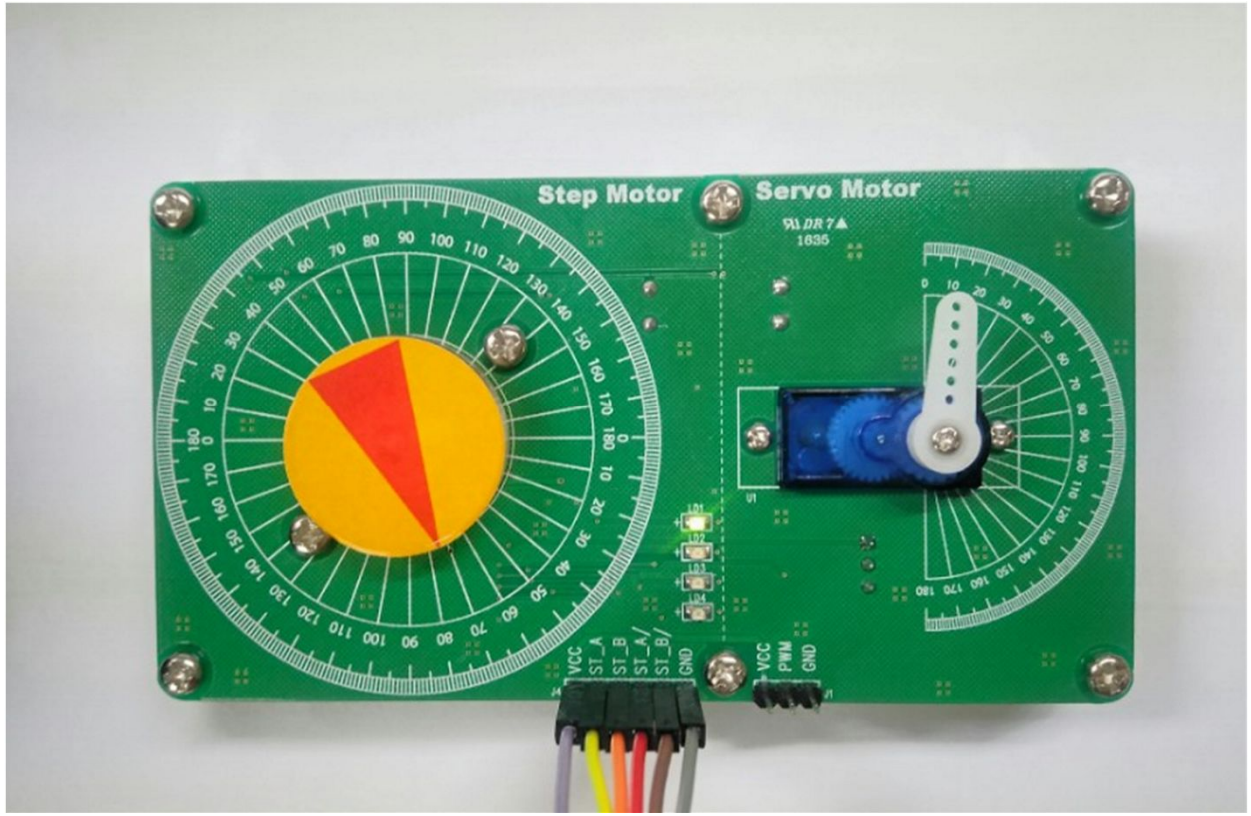
- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(TIMER1_OVF_vect)
{
  cli();
  TCNT1H=0xFB;                    // 22ms
  TCNT1L=0x0D;

  PORTD = Step[mot_cnt];          // 1-2상 여자 방식 한 스텝 진행
  if(dir==DIR_R)                  // 회전 방향이 우측 방향이면
  {
    if(mot_cnt++==7) mot_cnt=0;  // 스텝 카운터 증가
  }
  else if(mot_cnt--==0) mot_cnt=7; // 스텝 카운터 감소
  sei();
}
```

실습 4 : 1-2상 여자 방식으로 스텝 모터 돌리기 2

- 실행 결과
  - 스텝 모터가 1-2상 여자 방식으로 회전하며 2초마다 방향 전환



응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

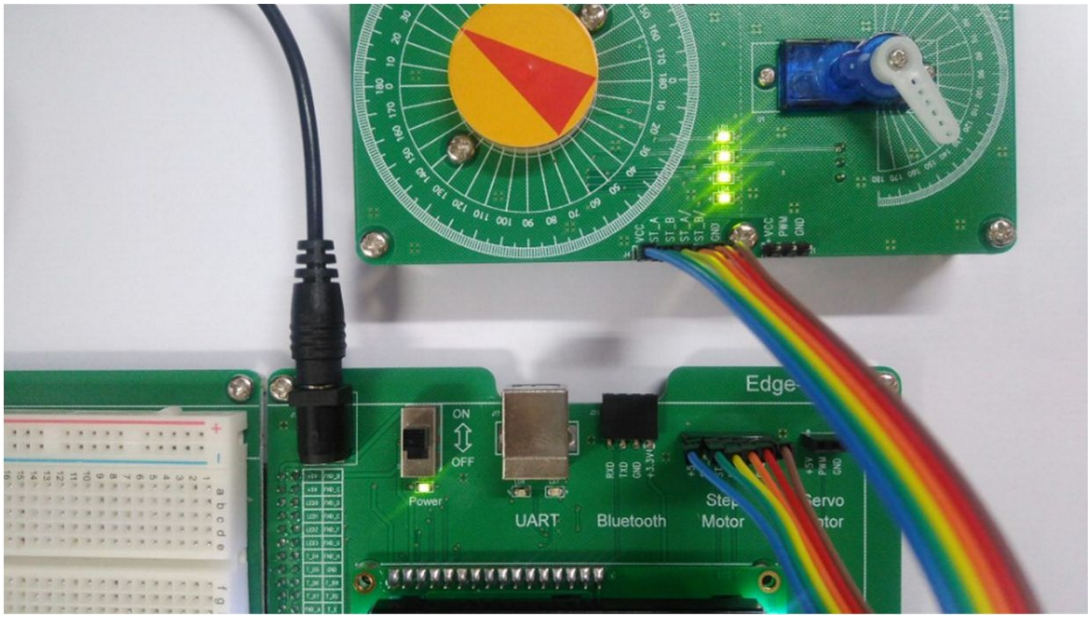
edgeILAB

- 실습 개요
  - ATMega128의 GPIO핀에 스텝 모터의 제어 신호를 연결하여, 스텝 모터를 회전 시키기
  - 각 스위치를 누르면 정지, PIEZO, 시계방향, 반시계방향으로 스텝모터와 PIEZO를 제어
  - 스텝 모터의 구동 방식은 1-2상 여자 방식을 쓰도록 함
  - 각 상태를 TextLCD화면에 표시

- 실습 목표
  - 스텝 모터의 동작원리를 이해
  - ATmega128A의 GPIO를 이용한 스텝 모터 구동 방법 습득
  - 인터럽트, GPIO 제어의 복합적인 프로그램 능력 배양

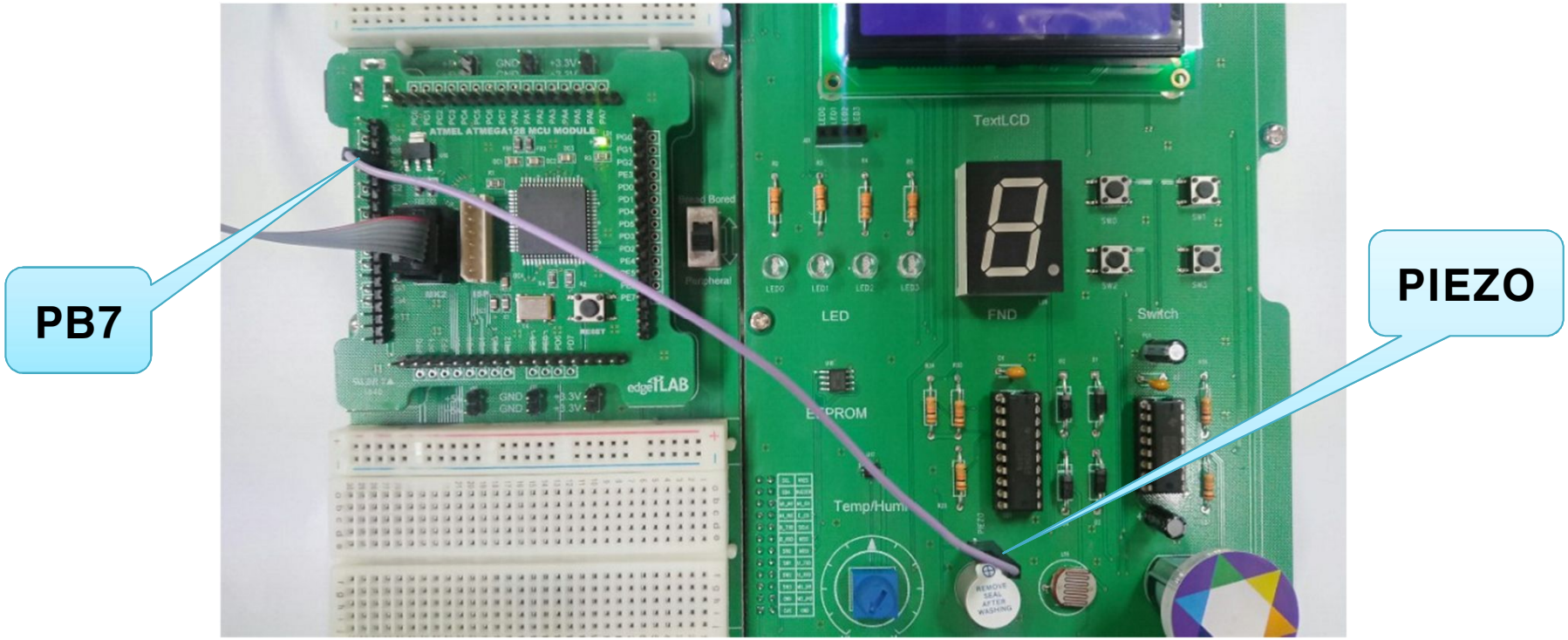
응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-Peri보드의 +5V → Step 모터 보드의 +5V
    - Edge-Peri보드의 ST_A → Step 모터 보드의 ST_A
    - Edge-Peri보드의 ST_B → Step 모터 보드의 ST_B
    - Edge-Peri보드의 ST_/A → Step 모터 보드의 ST_/A
    - Edge-Peri보드의 ST_/B → Step 모터 보드의 ST_/B
    - Edge-Peri보드의 GND → Step 모터 보드의 GND



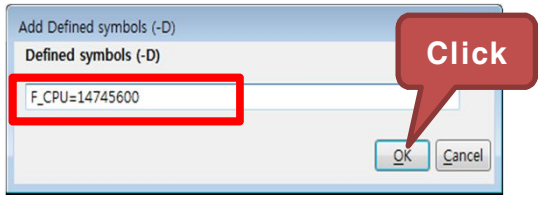
응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

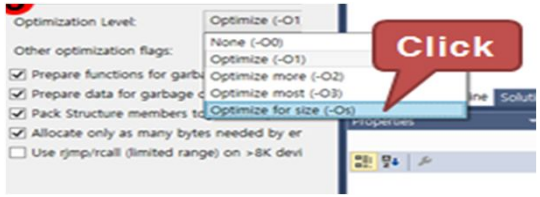
- 실습 준비
  - 사용보드를 다음 그림과 같이 연결
    - Edge-MCU보드의 PB7 → Edge-Peri 보드의 PIEZO(PIEZO상단우측)

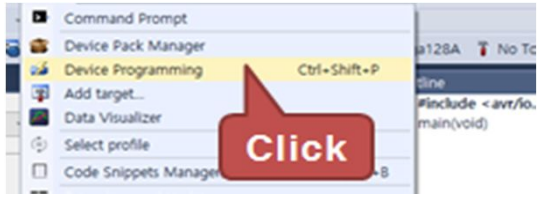


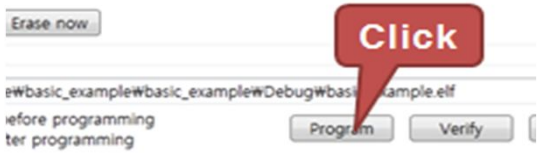
응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 예제 프로그램 작성 및 구동
  - Atmel Studio 실행
  - New Project 생성
    - Name : 12_Step_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



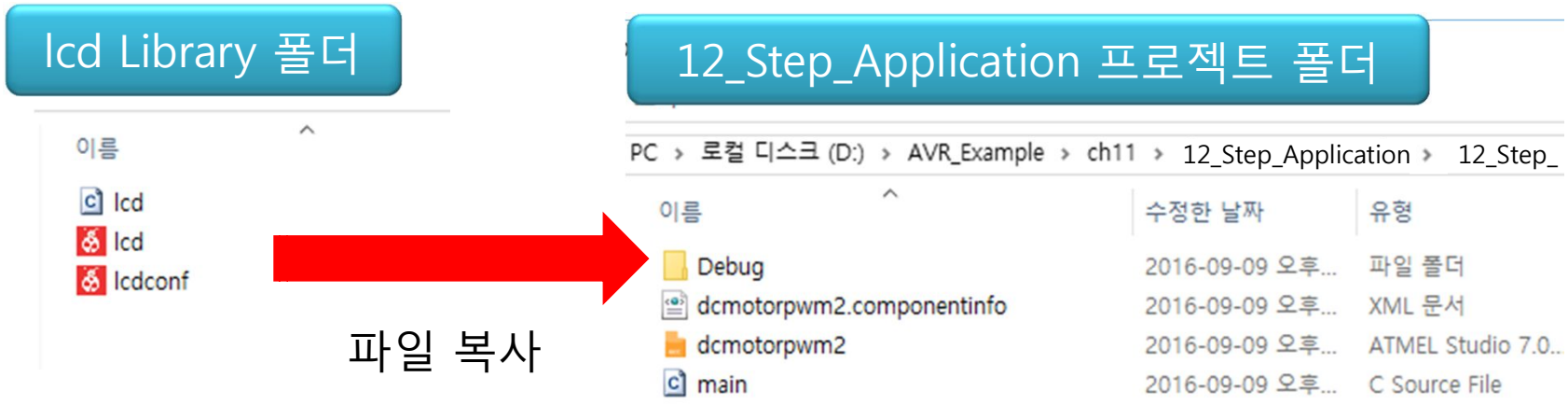






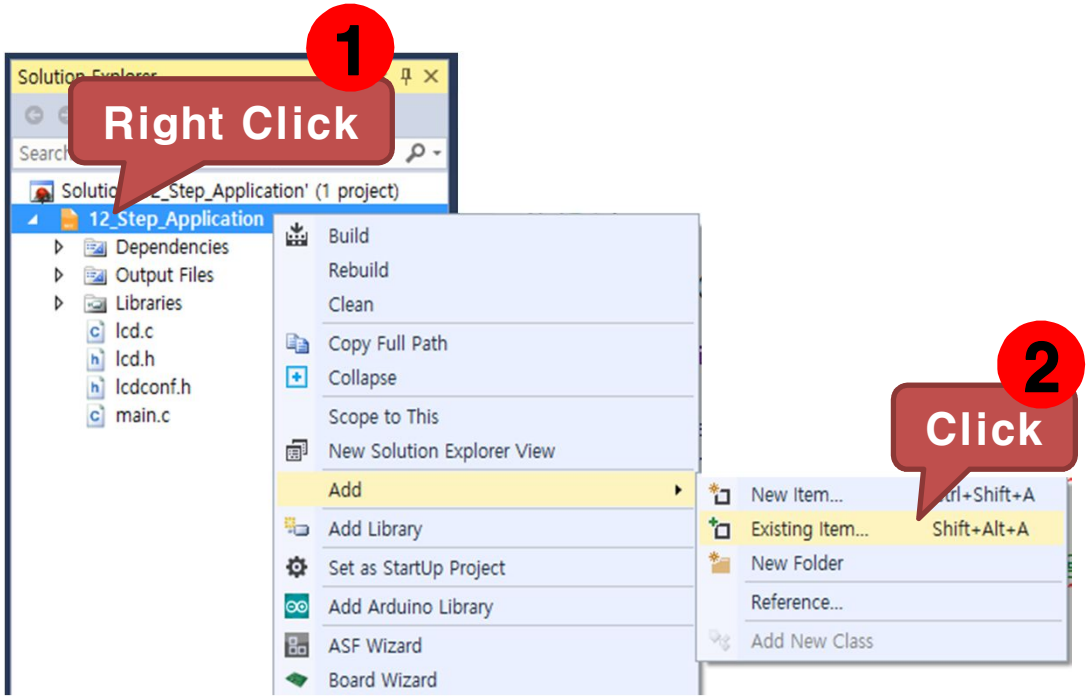
응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 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



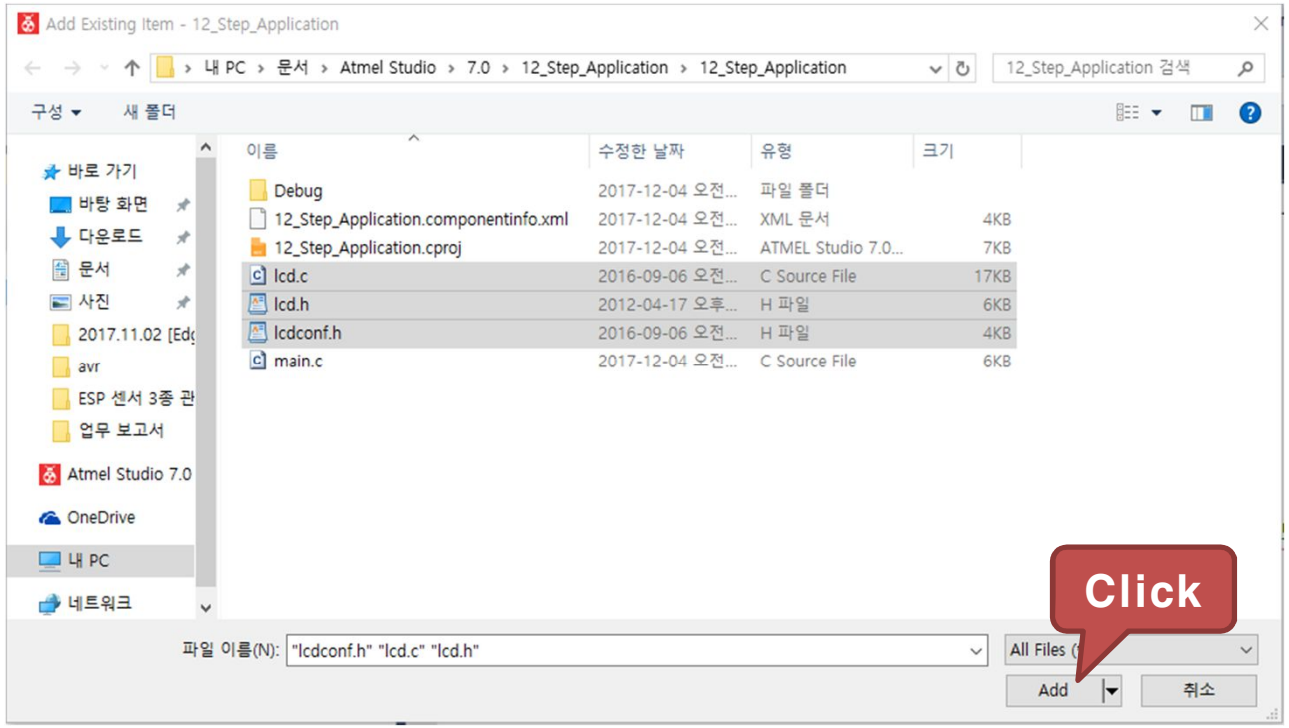
응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

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



응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 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 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

edgeILAB

- 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
#include "lcd.h"                       //Text LCD를 사용하기 위한 헤더 파일

#define DIR_L 0
#define DIR_R 1

unsigned char timer0Cnt=0, mot_cnt=0;
volatile unsigned char dir=DIR_R;   //처음방향은 우측부터
volatile unsigned char Step_flag = 0, buzzer_flag = 0, LCD_flag =
0;
//1-2 상 여자 값을 사용
unsigned char Step[]={0x90, 0x80, 0xC0, 0x40, 0x60, 0x20, 0x30,
0x10};
//피아노 음계에 해당하는 PWM 주파수
unsigned int DoReMi[8] = {523,587, 659, 698, 783,880, 987, 1046};
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void)
{
   unsigned char piano=0;

   DDRB = 0xA0;     // MOTOR1_EN, PWM(OCR1C)을 출력 포트로 설정 한다.
   // PB7에 PIEZO 연결
   DDRD = 0xF0;     // STEP0, STEP1, STEP2, STEP3을 출력 포트로 설정
한다.
   DDRE = 0x00;     // 포트E 를 출력포트로 설정한다.

  PORTB &= ~0x20;  //M1 Disable, DC 모터 정지

  lcdInit();        //Text LCD를 초기화

   TCCR0 = 0x07;
   TCNT0 = 112;     // 256-144=112 -> 10ms 마다 한번씩 인터럽트 발생
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

edgeILAB

- 구동 프로그램
  - main.c 코드 작성

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

   TIMSK = 0x01;         // TOIE0 '1'
   TIFR = 0x01;          // TOV0 '1'

   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
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

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

   lcdGotoXY(0,0); //현재 커서위치를 첫번째줄 첫번째칸으로 이동한다.
   //첫번째 매개변수는 행을 의미하고, 두번째 매개변수는 열을 의미한다.
   lcdPrintData("STEP Motor : OFF",16); //"STEP Motor : OFF" 문자
열을 출력한다.

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

   while (1)
   {
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void)
{
   //...
        if(LCD_flag)
        {
          if(LCD_flag & 0x01)    // 스텝 모터 및 부저 상태 OFF
          {
                lcdGotoXY(13,0);
                lcdPrintData("OFF",3);  // STEP 모터 상태 off 출력
                lcdGotoXY(11,1);
                lcdPrintData("OFF",3);  // 부저 상태 off 출력
                LCD_flag &= 0x0E;
          }
          if(LCD_flag & 0x02)    // 부저 상태 ON
          {
                lcdGotoXY(11,1);
                lcdPrintData("ON ",3);  // 부저 상태 ON 출력
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void)
{
   //...
                buzzer_flag = 1;  // buzzer 동작 허용
                LCD_flag &= 0x0D;
          }
          if(LCD_flag & 0x04) // 스텝 모터 상태 우측방향으로 회전(CW)
          {
                lcdGotoXY(13,0);
                lcdPrintData("CW ",3);  // STEP 모터 상태 CW 출력
                LCD_flag &= 0x0B;
          }
          if(LCD_flag & 0x08)    // 스텝 모터 상태 좌측방향으로 회전
(CCW)
          {
                lcdGotoXY(13,0);
                lcdPrintData("CCW",3);  // STEP 모터 상태 CCW 출력
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
int main(void)
{
   //...
                  LCD_flag &= 0x07;
            }
         }
         if(buzzer_flag)
         {
            ICR1 = 14745600/DoReMi[piano]; // 버튼에 맞는 음향을 연주
한다
            OCR1C = ICR1/2;              // 50% 듀티비
            piano++;                     // piano 변수 1증가
            if(8 < piano) piano = 0;     // piano가 9가 되면 초기화
            _delay_ms(1000);
         }
   }
}
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(TIMER0_OVF_vect)
{
  cli();
  TCNT0 = 112; // 256-144=122 -> 0.01초 마다 한번씩 인터럽트 발생
  if(Step_flag)
  timer0Cnt++; // timer0Cnt 변수를 1 증가 시킨다.
  if(timer0Cnt == 2)      // 10ms * 2 = 20ms  //20ms를 얻기 위한 카운트 횟수
  {     // (1 / (14745600 / 1024)) * 144 * 2 = 20ms -> 50hz
        timer0Cnt = 0;
        PORTD = Step[mot_cnt];   //1-2상 여자 방식 한 스텝 진행
        if(dir==DIR_R)           //회전 방향이 우측 방향이면
        {
          if(mot_cnt++==7) mot_cnt=0;  // 스텝 카운터 증가
        }
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
        else if(mot_cnt--==0) mot_cnt=7; // 스텝 카운터 감소
    }
    sei();
}

SIGNAL(INT4_vect)          // 인터럽트 서비스 루틴
{
    cli();          // 전체 인터럽트를 금지
    Step_flag = 0;          // STEP 모터 동작 정지
    PORTD = 0;              // STEP 모터 정지
    buzzer_flag = 0;        // buzzer 동작 정지
    OCR1C = 0;              // buzzer OFF
    LCD_flag |= 0x01;       // LCD에 부저 및 스텝 모터 상태를 출력하도록
설정
    sei();          // 전체 인터럽트를 허용
}
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

```
SIGNAL(INT5_vect)          // 인터럽트 서비스 루틴
{
   cli();          // 전체 인터럽트를 금지
   LCD_flag |= 0x02;      // LCD에 부저 동작 상태를 출력하도록 설정
   sei();          // 전체 인터럽트를 허용
}

SIGNAL(INT6_vect)          // 인터럽트 서비스 루틴
{
   cli();          // 전체 인터럽트를 금지
   Step_flag = 1;  // STEP 모터 동작 허용
   dir = DIR_R;    // 우측방향으로 회전
   LCD_flag |= 0x04;      // LCD에 스텝모터 동작 상태를 출력하도록 설정
   sei();          // 전체 인터럽트를 허용
}
```

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 구동 프로그램
  - main.c 코드 작성

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

   Step_flag = 1;          // STEP 모터 동작 허용
   dir = DIR_L;            // 좌측방향으로 회전
   LCD_flag |= 0x08;       // LCD에 스텝모터 동작 상태를 출력하도록 설정

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

응용 : 스위치 입력 값에 따라 스텝모터 제어와 PIEZO 울리기

- 실행 결과
  - 버튼에 따라 스텝모터의 방향이 변하고 LCD에 CW,CCW 혹은 OFF가 표시
  - SW1을 누르면 PIEZO가 울리고 SW0를 누르면 스텝모터와 PIEZO가 정지