

Chapter 111. PWM

1. PWM (Pulse Width Modulation, 펄스 폭 변조)

마이크로컨트롤러로 처리할 수 있는 데이터는 0이나 1의 디지털 데이터 뿐이다. 디지털 데이터로 많은 동작을 제어할 수 있지만 단순히 LED를 켜거나 끄는 동작이 아니라 밝기를 조절하고 싶은 경우가 있을 수 있다. 또는 모터의 속도를 조절하고 싶을 수도 있다. 이런 경우 디지털 데이터로는 제어가 어려우며 아날로그 데이터가 필요하다. 마이크로컨트롤러는 디지털 데이터만을 처리할 수 있지만 아날로그 데이터 역시 디지털 데이터로 변환하여 처리할 수 있으며 아날로그 데이터를 처리하기 위한 대표적인 방법으로 AD (Analog to Digital) 변환이 있다. AD 변환의 경우 일반적으로 PAM (Pulse Amplitude Modulation) 방식을 사용한다. 그림 1과 같은 아날로그 신호가 있다고 가정해 보자.

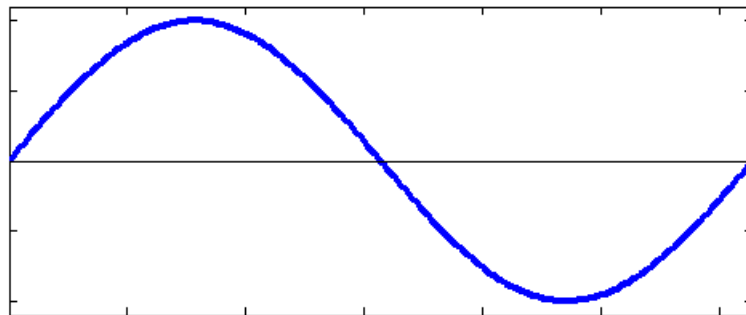


그림 1. 아날로그 신호

PAM 방식으로 아날로그 데이터를 디지털 데이터로 변환하기 위해서는 먼저 샘플링 과정을 거친다. 샘플링 과정은 동일한 시간 간격으로 아날로그 신호의 값을 취하는 과정이다.

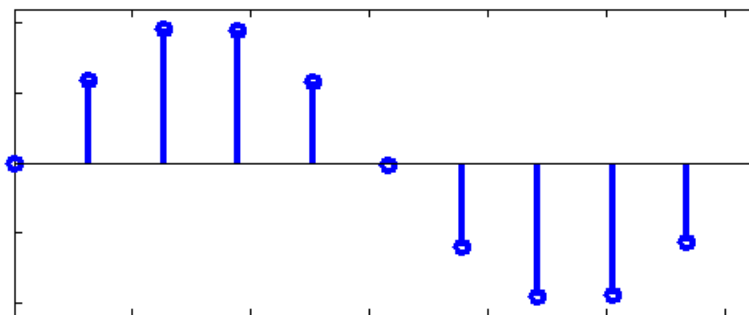


그림 2. 샘플링

샘플링된 신호는 다시 그 높이를 주어진 해상도에 따라 가장 가까운 디지털 값으로 양자화 한다. 8비트 해상도를 가진다고 가정하면 그림 2에서 샘플링 된 값들은 0에서 255 사이의 값으로 표현된다.

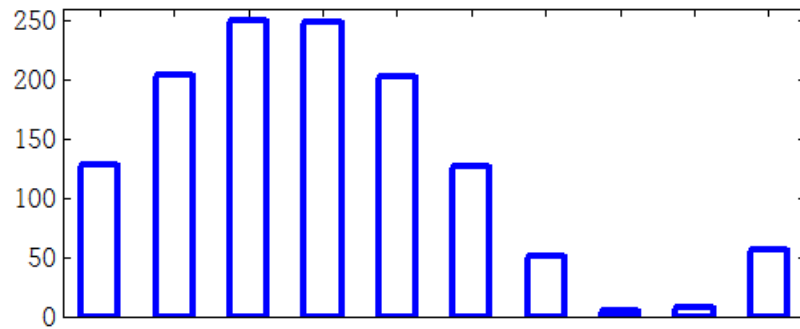


그림 3. 양자화

Atmega328의 경우에는 10비트의 해상도를 가지므로 신호의 높이를 0에서 1023까지 1024 단계의 디지털 값으로 바꾼다. 양자화된 신호는 부호화 과정을 통해 0과 1을 열로 표현되어 마이크로컨트롤러에서 처리된다.

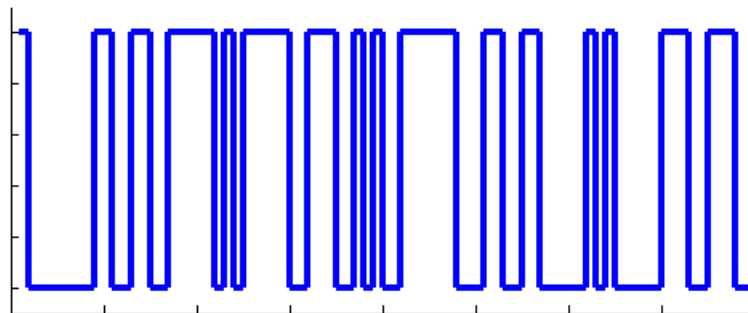


그림 4. PAM 방식의 디지털 신호

PWM 역시 아날로그 데이터를 디지털로 처리하는 방식 중 하나이지만 아날로그 값을 펄스의 높이가 아닌 펄스의 폭으로 표현한다는 점에서 PAM 방식과 차이가 있다.

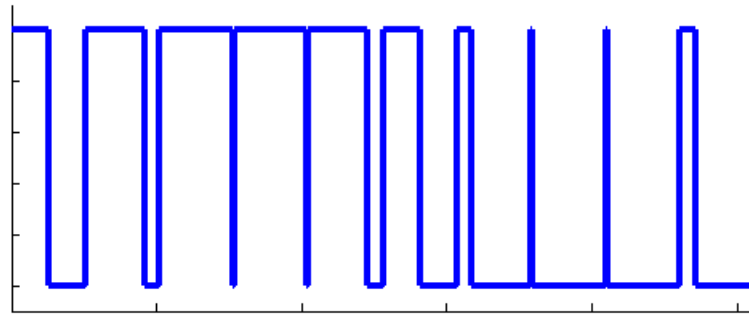


그림 5. PWM 방식의 디지털 신호

그림 4는 하나의 샘플을 동일한 개수(해상도)의 0 또는 1을 이용하여 이진수로 표현한 것이라면 그림 5는 하나의 샘플을 한 주기 내에서 HIGH 값이 가지는 비율로서 표현한 것이다. 이처럼 한 주기 내에서 HIGH 값이 가지는 비율을 듀티 사이클(duty cycle)이라고 하며 그림 6은 듀티 사이클에 따른 PWM 신호의 파형을 보여준다.

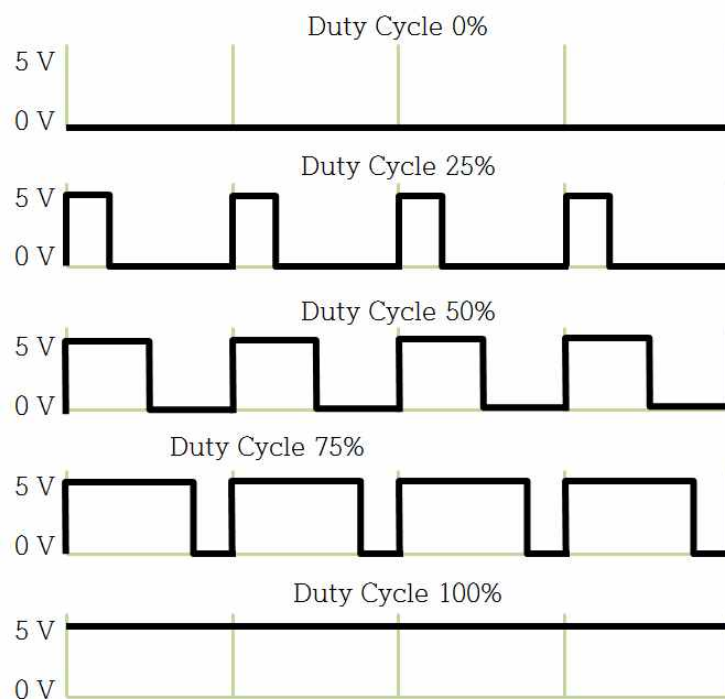


그림 6. PWM 신호의 듀티 사이클

정확한 비교는 아니지만 그림 4의 PAM 방식에서 하나의 샘플을 표현하기 위해 8비트가 필요하다면, 그림 5의 PWM 방식에서는 하나의 샘플을 표현하기 위해 256비트가 필요한 것으로 볼 수 있다. 즉, PWM 방식으로 디지털 데이터를 저장하기 위해서는 많은 저장 공간이 필요

하며 따라서 동작 주파수 역시 PAM과 비교했을 때 훨씬 높다. PAM 방식에 비해 그다지 장점이 없어 보이지 않는가?

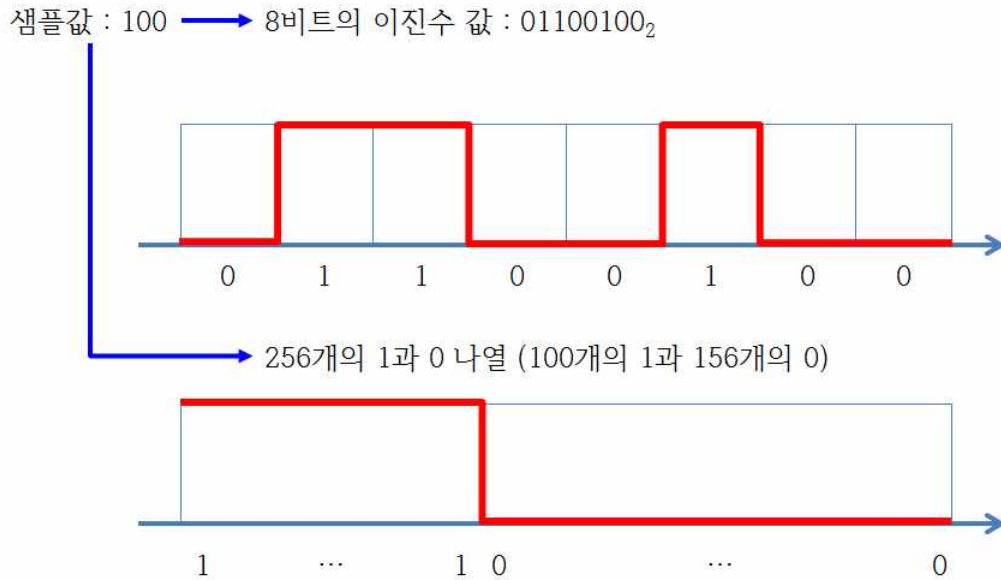


그림 7. PAM 방식 샘플과 PWM 방식 샘플 비교

PWM 신호가 PAM 신호에 비해 신호 대 잡음비가 높은 것도 사실이지만 큰 저장 공간과 높은 동작 주파수로 인해 디지털 데이터의 저장이나 전송을 위한 방식으로는 널리 사용되지는 않는다. 하지만 PWM 신호는 PAM 신호에 비해 간단하게 아날로그 신호로 변환할 수 있어 모터의 속도나 LED의 밝기를 제어하기에 편리하므로 마이크로컨트롤러에서 널리 사용된다.

LED의 예를 들어보자. LED의 밝기를 조절하기 위해서는 LED에 가변저항을 연결하여 LED에 가해지는 전압이나 전류를 조절함으로써 가능하다. 하지만 이는 아날로그 영역으로 마이크로컨트롤러로는 전류나 전압을 직접 제어하기 어렵다. 다른 방법은 주어진 시간, 즉, 한 주기 내에서 LED를 켜는 시간을 조절하는 것이다. 한 주기 동안 계속해서 LED를 켜면, 즉, 듀티 사이클이 100%라면 LED는 항상 켜져 있게 되고 100%의 밝기를 보여줄 것이다. 하지만 한 주기 내에서 절반만 LED를 켜고 나머지 절반은 LED를 끈다면 잔상 효과에 의해 50% 정도의 밝기로 인식될 수 있다. 이처럼 PWM은 빠른 속도로 스위치를 켜고 끄기를 반복하여 평균 전압을 조절함으로써 0이나 1이 아닌 그 사이의 ‘아날로그 값과 유사한’ 값을 가질 수 있도록 해준다. 하지만 한 가지 주의할 점은 LED가 PWM 신호의 한 주기 내에서 켜고 끄는 동작에 개별적으로 반응할 수 없을 정도로 빨리 동작하여야 한다는 점이다. Blink 예제를 기억하는가? 그림 7에서와 같이 256비트로 구성되는 PWM 신호가 있을 때 하나의 비트가 1초에 해당

한다면 100초 동안은 LED가 켜져 있고 나머지 156초 동안은 LED가 꺼져 있을 것이다. LED가 최대 밝기의 $\frac{100}{256} \times 100 \approx 39\%$ 정도로 켜져 있기를 기대한 것과는 차이가 있다. 따라서 PWM 신호에서의 1비트는 LED가 반응할 수 없을 정도로 빨라야 하며 개별 비트가 아닌 한 주기 내에 256개 비트의 평균값에 LED가 반응하도록 하여야 한다. PWM 신호의 한 주기는 제어하고자 하는 장치에 따라 달라지며 이를 'PWM 주파수'로 설명하는 것이 일반적이다. 코드 1을 살펴보자.

코드 1. delay 함수를 이용한 PWM

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

#define LED_TIME 20

void turn_on_LED_in_PWM_manner(int dim)
{
    int i;

    PORTB = 0xFF;                // LED 켜기

    for(i = 0; i < 256; i++){
        if(i > dim) PORTB = 0x00;    // 듀티 사이클을 넘어가면 LED 끄기
        _delay_us(LED_TIME);
    }
}

int main(void)
{
    DDRB = 0xFF;                // PB5 핀(아두이노 13번)을 출력으로 설정

    int dim = 0;                // 현재 LED 밝기
    int direction = 1;          // 밝기 증가(1) 또는 감소(-1)

    while(1)
    {
```

```

    if(dim == 0) direction = 1;
    if(dim == 255) direction = -1;

    dim += direction;                // 밝기 변화

    turn_on_LED_in_PWM_manner(dim);
}
}

```

코드 1은 한 주기 내에서 LED를 켜고 끄는 시간 비율을 조절함으로써 LED의 밝기를 조절하는 예이다. LED는 PB5 핀(아두이노 13번 핀)에 연결한 것으로 설명하였지만 포트 B의 핀 어디에나 LED를 연결하여도 무방한다. 아두이노 우노 보드의 경우 PB5 핀(아두이노 13번 핀)에 연결된 내장 LED가 존재하지만 칩 LED여서 밝기 변화를 관찰하기가 쉽지 않을 수 있으므로 별도로 LED를 연결해서 확인하기를 추천한다. 또 한 가지 유의할 점은 turn_on_LED_in_PWM_manner 함수에서 밀리초 단위의 지연 함수 _delay_ms가 아닌 마이크로초 단위의 지연 함수인 _delay_us 함수가 사용되었다는 점이다. PWM 신호는 높은 주파수로 동작한다.

코드 1에서 알 수 있듯이 PWM을 구현하는 것은 쉽다. 하지만 한 가지 심각한 문제가 있다. PWM 신호를 출력하기 위해 거의 대부분의 CPU 클럭을 사용하여야 한다는 점이다. 코드 1의 main 함수에서는 계속해서 turn_on_LED_in_PWM_manner 함수를 호출하고 있으며 대부분의 시간은 이 함수 내에서 소비된다. PWM 신호 생성을 직접 구현할 수는 있지만 직접 구현하게 되면 다른 작업을 할 수 있는 여지가 그다지 많지 않다. 다행히 PWM 신호 출력을 위한 전용 회로가 ATmega328에 포함되어 있어 CPU 클럭을 소비하지 않고도 PWM 신호를 출력할 수 있도록 해준다. 하지만 모든 범용 입출력 핀으로 PWM 신호를 출력할 수 있는 것은 아니며 PB1, PB2, PB3, PD3, PD5, PD6의 6개 핀(아두이노 3, 5, 6, 9, 10, 11번 핀)으로만 ‘하드웨어적으로 생성된’ PWM 신호 출력이 가능하다. 눈에 익은 핀들이 아닌가? 타이머/카운터의 동작을 설명하면서 비교일치가 발생하면 인터럽트와 더불어 파형 생성기를 통해 신호를 특정 핀으로 출력할 수 있다고 설명했던 것을 기억하는가? PWM을 사용할 수 있는 핀은 바로 파형 생성기를 통해 신호가 출력되는 OCnx (n = 0, 1, 2; x = A, B) 핀에 해당한다.

타이머/카운터	파형 출력 핀		아두이노 핀 번호
0	OC0A	PD6	6
	OC0B	PD5	5
1	OC1A	PB1	9
	OC1B	PB2	10
2	OC2A	PB3	11
	OC2B	PD3	3

표 1. 비교일치 인터럽트 시 파형 출력 핀

타이머/카운터 1번의 경우 TCCR1A 레지스터와 TCCR1B 레지스터를 통해 16개의 동작 모드를 설정할 수 있으며 그 중 정상 모드와 CTC (Clear Timer on Compare Match) 모드만을 설명하였고 나머지 12개는 PWM과 관계된 모드라고 설명한 것을 기억하는가? 타이머/카운터를 이용하여 PWM 신호를 생성해 보기 전에 PWM 신호 생성을 위한 모드를 살펴보자.

2. PWM 모드

ATmega328에서 사용할 수 있는 PWM 신호 생성 모드에는 고속 (fast) PWM 모드, 위상 교정 (phase correct) PWM 모드, 위상 및 주파수 교정 (phase and frequency correct) PWM 모드 등 세 가지가 존재한다. PWM 모드 설명을 위해서는 몇 가지 용어를 명확히 할 필요가 있으며 이들을 표 2에 나타내었다. 표 2는 8비트 타이머/카운트의 경우에 해당하며 16비트 타이머/카운트의 경우도 유사하게 정의된다.

단어	설명
BOTTOM	카운터의 값이 0x00일 때를 가리킨다.
MAX	카운터가 0xFF일 때를 가리킨다.
TOP	카운터가 가질 수 있는 최댓값을 가리킨다. 오버플로 인터럽트의 경우 TOP은 0xFF이지만, 비교일치 인터럽트의 경우 사용자가 설정한 값이 TOP이 된다.

표 2. 타이머/카운터 동작을 위한 용어 정의

(1) 고속 PWM 모드

가장 간단한 파형 생성 방법이며 가장 흔히 사용되는 방법으로 카운터는 BOTTOM에서 TOP까지 카운트하고 다시 BOTTOM으로 초기화된다. 비반전 출력 모드(non-inverting compare output mode)인 경우 카운터가 BOTTOM일 때 해당 OCnx 핀으로 HIGH를 출력하고 카운트 값이 출력 비교 레지스터인 OCRnx 값과 일치하는 경우, 즉, 비교일치 상황에서 LOW를 출력한다. 이는 코드 1에서 PWM 신호를 생성한 방식과 동일하다. 반전 출력 모드인 경우 HIGH와 LOW 출력은 바뀌어 나타난다. 출력 비교 레지스터인 OCRnx 레지스터의 값은 BOTTOM과 TOP 사이의 값을 가지도록 설정하여야 하며 OCRnx 레지스터의 값이 클수록 듀티 사이클은 커진다. 실제 파형이 출력되도록 하기 위해서는 OCnx 핀을 출력으로 설정하여야 한다는 점도 잊지 말아야 한다.

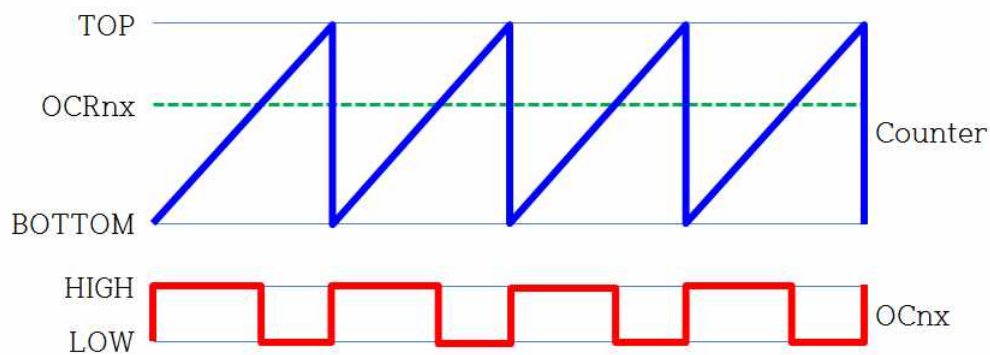


그림 8. 비반전 고속 PWM 모드에서의 PWM 파형 생성

고속 PWM 모드에서 PWM 주파수($f_{fast\ PWM}$)는 타이머/카운터에 공급되는 주파수를 TOP 값으로 나눈 값에 근접한 값이다. 타이머/카운터에 공급되는 주파수는 시스템 클록(f_{clk})을 분주율(N)로 나눈 값이므로 PWM 주파수는 식 (1)과 같이 표현할 수 있다.¹⁾

$$f_{fast\ PWM} \approx \frac{f_{clk}}{N \cdot TOP} \quad (1)$$

(2) 위상 교정 PWM 모드

위상 교정 PWM 모드는 해상도가 높은 PWM 파형을 생성하기 위해 사용된다. 위상 교정

1) 8비트 타이머/카운터인지 16비트 타이머/카운터인지에 따라 고속 PWM 모드에서의 PWM 주파수는 약간의 차이가 있으므로 식 (1)에서는 근사값으로 표현하였다. 보다 상세한 내용은 데이터시트를 참고하기 바란다.

PWM 모드에서 카운터는 BOTTOM에서 TOP까지 상향으로 카운트한 후 TOP에서 BOTTOM으로 하향으로 카운트한다. 즉, 카운터의 값은 이중 경사 (dual slope) 파형을 이루므로 PWM 주파수는 고속 PWM 모드에 비해 절반이지만 해상도는 2배가 된다. 비반전 비교 출력 모드인 경우 상향 카운트에서 카운트 값이 출력 비교 레지스터인 OCRnx 값과 일치하는 경우, 즉, 비교일치 상황에서 해당 OCnx 핀으로 LOW 값이 출력되며 하향 카운트에서 비교일치 상황이 발생하면 HIGH 값이 출력된다. 반전 비교 출력 모드인 경우 HIGH와 LOW 출력은 바뀌어 나타난다.

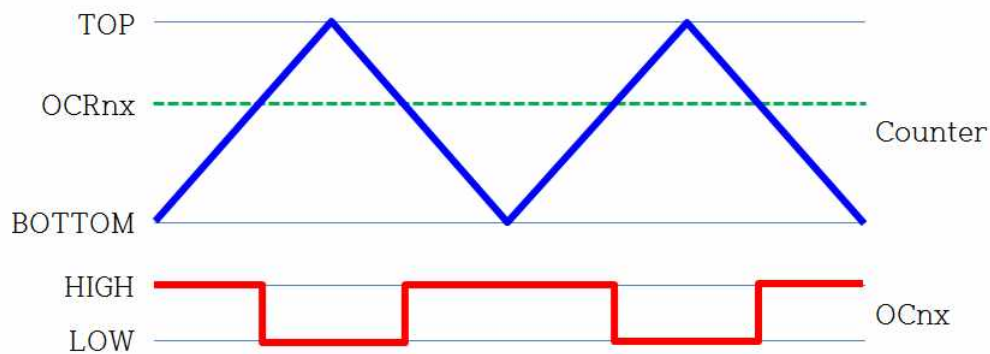


그림 9. 비반전 위상 교정 PWM 모드에서의 PWM 파형 생성

위상 교정 PWM 모드에서 PWM 주파수(f_{PCPWM})는 타이머/카운터에 공급되는 주파수를 TOP 값의 2배로 나눈 값에 근접한 값으로 식 (2)와 같이 표현할 수 있다.²⁾

$$f_{PCPWM} \approx \frac{f_{clk}}{N \cdot 2 \cdot TOP} \quad (1)$$

고속 PWM 모드와 위상 교정 PWM 모드의 차이는 PWM 주파수와 해상도에 있다. 그렇다면 ‘위상이 교정되었다’는 말은 무슨 말일까? 그림 10은 고속 PWM 모드에서 서로 다른 출력 비교 레지스터 값에 따라 출력되는 PWM 파형을 보인 것이다. 유사하게 그림 11은 위상 교정 PWM 모드에서 출력 비교 레지스터 값에 따라 출력되는 PWM 파형을 보인 것이다.

2) 고속 PWM 모드에서와 마찬가지로 위상 교정 PWM 모드에서도 8비트 타이머/카운터인지 16비트 타이머/카운터인지에 따라 PWM 주파수는 약간의 차이가 있다.

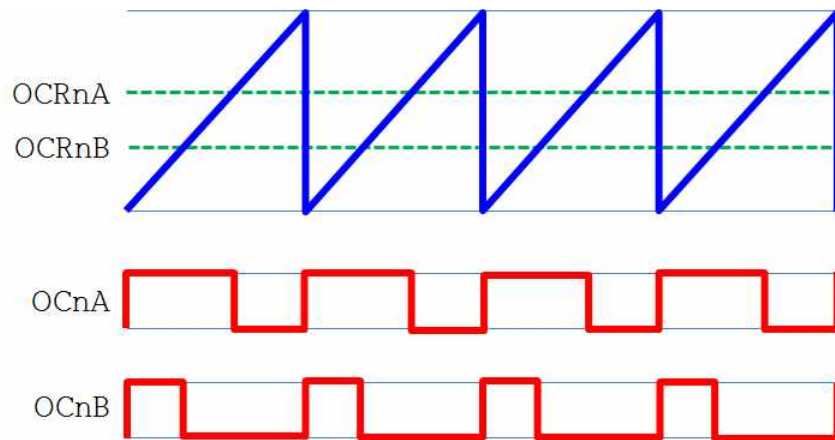


그림 10. 고속 PWM 모드에서 비교일치 값에 따른 파형



그림 11. 위상 교정 PWM 모드에서 비교일치 값에 따른 파형

PWM 주파수가 달라진 점이 눈에 띈다. 위상과 관련된 차이점을 발견했는가? 그림 10과 그림 11에서 위상과 관련된 차이는 펄스의 위치에서 찾아볼 수 있다. 고속 PWM 모드의 경우 펄스가 왼쪽에 맞추어져 있는 반면, 위상 교정 PWM 모드의 경우 펄스가 중앙에 맞추어져 있다. 일반적으로 모터 제어의 경우 위상 교정 PWM이 많이 사용된다. 속도 제어 측면에서 두 모드 사이에 큰 차이는 없지만 고속 PWM 모드의 경우 OCRnx 레지스터의 값이 BOTTOM으로 설정되면 1클럭 넓이의 스파이크가 발생하는 반면, 위상 교정 PWM의 경우 항상 LOW 값이 출력된다. 즉, 고속 PWM 모드의 경우 모터를 완전히 정지시키는 것이 불가능하므로 위상 교정 PWM 모드가 주로 사용된다.

(3) 위상 및 주파수 교정 PWM 모드

위상 및 주파수 교정 PWM 모드는 16비트 타이머/카운터에서만 사용할 수 있는 모드이다. 위상 및 주파수 교정 PWM 모드에서의 PWM 파형 출력 방식은 TOP 값이 고정되어 있는 경우 위상 교정 PWM 모드에서와 동일하므로 PWM 파형 생성 중 TOP 값을 변경하지 않는다면 위상 교정 PWM 모드나 위상 및 주파수 교정 PWM 모두 중 어느 것을 사용해도 동일하다. TOP 값을 변경하면 PWM 주파수가 변하지만 PWM 주파수는 시작 시점에서 결정하고 실행 도중 변경하지 않는 것이 일반적이다. 따라서 여기서는 위상 및 주파수 교정 PWM 모드에 대해서는 자세히 다루지 않을 것이므로 자세한 내용은 데이터시트를 참고하기 바란다.

3. PWM을 이용한 LED 밝기 제어

코드 2를 살펴보자. 코드 2는 2번 타이머/카운터의 고속 PWM 모드를 통해 OC2A 핀에 연결된 LED를 서서히 밝게 하다가 완전히 밝아지면 끄기를 반복하는 예이다.

코드 2. 고속 PWM 모드를 통한 LED 밝기 변화 1

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    // 파형 출력 핀인 PB3 핀(OC2A 핀, 아두이노 11번)을 출력으로 설정
    DDRB |= (1 << PB3);

    // 타이머/카운터 2번을 고속 PWM 모드로 설정
    TCCR2A |= (1 << WGM21) | (1 << WGM20);
    TCCR2A |= (1 << COM2A1);           // 비반전 모드
    TCCR2B |= (1 << CS22);             // 분주비 64

    while(1)
    {
        for(int i = 0; i < 256; i++){
            OCR2A = i;                 // 분주비 설정
            _delay_ms(10);
        }
    }
}
```

```

    }
  }
}

```

코드 2에서는 타이머/카운터 2번의 OCR2A 레지스터를 통해 고속 PWM 방식으로 생성된 신호를 출력하고 있다. 이를 위해서 먼저 해당 신호를 출력할 OC2A 핀(PB3 핀, 아두이노 11번 입출력 핀)을 출력하고 설정한다. 다음은 타이머/카운터 2번이 고속 PWM 방식으로 신호를 생성하도록 설정해 주어야 하며 이는 TCCR2A 및 TCCR2B 레지스터(Timer/Counter Control Register)를 통해 이루어진다. TCCR2A 레지스터와 TCCR2B 레지스터의 구조는 각각 그림 12 및 그림 13과 같다.

비트	7	6	5	4	3	2	1	0
	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20
읽기/쓰기	R/W	R/W	R/W	R/W	R	R	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 12. TCCR2A 레지스터 구조

비트	7	6	5	4	3	2	1	0
	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20
읽기/쓰기	W	W	R	R	R/W	R/W	R/W	R/W
초깃값	0	0	0	0	0	0	0	0

그림 13. TCCR2B 레지스터 구조

이 중 PWM 모드 설정에 사용되는 비트는 WGM (Waveform Generation Mode) 비트들이다. WGM 비트는 TCCR2A 레지스터에 2비트, TCCR2B 레지스터에 1비트, 총 3비트가 존재하며 8가지 모드를 설정할 수 있다. 표 3은 WGM 비트 설정에 따른 파형 생성 모드를 나타낸다.

모드	TCCR2B	TCCR2A		설명	TOP
	WGM22	WGM21	WGM20		
0	0	0	0	정상 모드	0xFF
1	0	0	1	위상 교정 PWM 모드	0xFF
2	0	1	0	CTC 모드	OCR2A
3	0	1	1	고속 PWM 모드	0xFF
4	1	0	0	-	-
5	1	0	1	위상 교정 PWM 모드	OCR2A
6	1	1	0	-	-
7	1	1	1	고속 PWM 모드	OCR2A

표 3. WGM 비트 설정에 따른 2번 타이머/카운터의 파형 생성 모드

정상 모드에서는 0에서 0xFF까지 반복적으로 카운팅을 하는 모드로 디폴트값이다. CTC (Clear Timer on Compare Match) 모드는 OCR2A 레지스터에 설정된 값과 현재 카운트 값이 동일한 경우 TCNT2 레지스터의 값을 초기화하는 점을 제외하면 정상 모드와 동일하다. 이들 모드에 대해서는 타이머/카운트 장에서 설명하였다.

이외에도 2개의 고속 PWM 모드와 2개의 위상 교정 PWM 모드를 2번 타이머/카운터를 통해 사용할 수 있다. 먼저 고속 PWM 모드를 살펴보면 2개의 모드는 TOP 값만이 다르다. 고속 PWM 모드의 경우 OCRnx 레지스터의 값과 현재 카운트 값을 비교하여 PWM 신호를 생성한다. 3번 고속 PWM 모드의 경우 TOP 값이 0xFF로 설정되어 있으므로 OCR2A나 OCR2B 레지스터 값을 통해 OC2A나 OC2B 핀을 통해 출력되는 PWM 신호의 듀티 사이클을 조절하는 것이 가능하다. 하지만 7번 고속 PWM 모드의 경우 TOP 값이 OCR2A 레지스터에 저장되므로 OC2A 핀을 통해 PWM 신호를 출력하는 용도로는 사용할 수 없으며 OC2B 핀을 통해서만 PWM 신호를 출력할 수 있다. 이는 위상 교정 PWM 모드의 경우에도 동일하다. 코드 2의 경우 3번 고속 PWM 모드를 선택하였으므로 OCR2A 핀을 통해 PWM 신호를 출력하고 있다. TCCR2A 레지스터의 COM2An ($n = 0, 1$) 비트와 COM2Bn ($n = 0, 1$) 비트 역시 파형 생성에 영향을 미치는 비트로 PWM 모드가 아닌 정상 모드나 CTC 모드의 경우 세트, 클리어 또는 반전 출력을 선택할 수 있도록 해준다.

COM2A1	COM2A0	설명
0	0	OC2A 핀으로 데이터가 출력되지 않으며 OC2A 핀은 일반적인 범용 입출력 핀으로 동작한다.
0	1	비교일치가 발생하면 OC2A 핀의 출력은 반전된다.
1	0	비교일치가 발생하면 OC2A 핀의 출력은 LOW 값으로 바뀐다.
1	1	비교일치가 발생하면 OC2A 핀의 출력은 HIGH 값으로 바뀐다.

표 4. 비 PWM 모드에서 COM2An (n = 0, 1) 비트 설정에 따른 OC2A 핀의 동작

반면 PWM 모드의 경우에는 반전 또는 비반전 모드를 선택할 수 있도록 해주며 코드 2에서는 비반전 모드를 선택하고 있다.

COM2A1	COM2A0	설명
0	0	OC2A 핀으로 데이터가 출력되지 않으며 OC2A 핀은 일반적인 범용 입출력 핀으로 동작한다.
0	1	WGM22 = 1인 경우 OC2A 핀으로 데이터가 출력되지 않으며 OC2A 핀은 일반적인 범용 입출력 핀으로 동작한다. WGM22 = 0인 경우 비교일치가 발생하면 OC2A 핀의 출력은 반전된다.
1	0	비교일치가 발생하면 OC2A 핀의 출력은 LOW 값으로 바뀌고 BOTTOM 값에서 OC2A 핀의 출력은 HIGH 값으로 바뀐다. (비반전 모드)
1	1	비교일치가 발생하면 OC2A 핀의 출력은 HIGH 값으로 바뀌고 BOTTOM 값에서 OC2A 핀의 출력은 LOW 값으로 바뀐다. (반전 모드)

표 5. 고속 PWM 모드에서 COM2An (n = 0, 1) 비트 설정에 따른 OC2A 핀의 동작

COM2A1	COM2A0	설명
0	0	OC2A 핀으로 데이터가 출력되지 않으며 OC2A 핀은 일반적인 범용 입출력 핀으로 동작한다.
0	1	WGM22 = 1인 경우 OC2A 핀으로 데이터가 출력되지 않으며 OC2A 핀은 일반적인 범용 입출력 핀으로 동작한다. WGM22 = 0인 경우 비교일치가 발생하면 OC2A 핀의 출력은 반전된다.
1	0	상향 카운트에서 비교일치가 발생하면 OC2A 핀의 출력은 LOW 값으로 바뀌고 하향 카운트에서 비교일치가 발생하면 OC2A 핀의 출력은 HIGH 값으로 바뀐다. (비반전 모드)
1	1	상향 카운트에서 비교일치가 발생하면 OC2A 핀의 출력은 HIGH 값으로 바뀌고 하향 카운트에서 비교일치가 발생하면 OC2A 핀의 출력은 LOW 값으로 바뀐다. (반전 모드)

표 6. 위상 교정 PWM 모드에서 COM2An (n = 0, 1) 비트 설정에 따른 OC2A 핀의 동작

CS2n (n = 0, 1, 2) 비트는 분주율을 결정하기 위해 사용되는 비트이다. CS2n 비트 설정에 따른 분주비는 표 7과 같다.

CS02	CS01	CS00	설명
0	0	0	클록 소스 없음 (타이머/카운터 정지)
0	0	1	분주비 1
0	1	0	분주비 8
0	1	1	분주비 32
1	0	0	분주비 64
1	0	1	분주비 128
1	1	0	분주비 256
1	1	1	분주비 1024

표 7. CS2n (n = 0, 1, 2) 비트 설정에 따른 분주비

한 가지 유의할 점은 0번과 1번 타이머/카운트의 경우 선택할 수 있는 분주비가 1, 8, 64, 256, 1024의 5가지 중 하나이지만 2번 타이머/카운트의 경우 1, 8, 32, 64, 128, 256, 1024

등 7가지의 분주비 중 하나를 선택할 수 있다. 대신 0번과 1번 타이머/카운터의 경우에는 T_n ($n = 0, 1$) 핀으로 공급되는 외부 클록을 타이머/카운트의 클록으로 사용할 수 있지만 2번 타이머/카운터의 경우에는 외부 클록을 사용할 수 없다.

코드 2번의 경우는 3번의 고속 PWM 모드를 사용하고 있다. 이를 7번의 고속 PWM 모드를 사용하도록 해보자. 앞에서 설명하였던 것처럼 7번 고속 PWM 모드는 OC2B 핀을 통해서만 PWM 신호를 출력할 수 있다. 코드 3은 7번 고속 PWM 모드를 사용하여 LED 밝기를 조절하는 코드의 예이다.

코드 3. 고속 PWM 모드를 통한 LED 밝기 변화 2

```
#define F_CPU 16000000L
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    // 파형 출력 핀인 PD3 핀(OC2B 핀, 아두이노 3번)을 출력으로 설정
    DDRD |= (1 << PD3);

    // 타이머/카운터 2번을 고속 PWM 모드로 설정
    TCCR2A |= (1 << WGM21) | (1 << WGM20);
    TCCR2B |= (1 << WGM22);
    TCCR2A |= (1 << COM2B1);           // 비반전 모드
    TCCR2B |= (1 << CS22);             // 분주비 64

    OCR2A = 128;                       // TOP 설정

    while(1)
    {
        for(int i = 0; i < 128; i++){
            OCR2B = i;                 // 분주비 설정
            _delay_ms(10);
        }
    }
}
```

코드 3에서는 OCR2A 레지스터에 TOP 값을 128로 설정하였으므로 코드 2에 비해 2배 빠른 속도로 밝기가 변하는 것을 관찰할 수 있다.

4. 아두이노의 PWM

아두이노에서는 각 타이머/카운터의 OCnx (n = 0, 1, 2; x = A, B) 핀을 통해 PWM 신호를 출력할 수 있도록 analogWrite 함수를 제공하고 있다. 위에서 살펴본 것처럼 PWM 신호는 아날로그 신호가 아니라 디지털 신호이다. 하지만 디지털 신호를 통해 아날로그 신호와 유사한 효과를 얻고 있으므로 아두이노에서 PWM 신호 출력을 위해 제공하는 함수에는 ‘analog’라는 단어가 포함되어 있다.

```
void analogWrite(uint8_t pin, int value)
```

매개변수

pin : 핀 번호

value : 듀티 사이클 (duty cycle). 0(0%)에서 255(100%) 사이의 값

반환값 : 없음

아두이노의 analogWrite 함수는 핀 번호와 0에서 255 사이의 듀티 사이클을 매개변수로 가지므로 간단히 사용할 수 있는 장점은 있지만 다양한 제어를 위해 사용할 수 없는 단점이 있다. 특히 PWM 주파수를 조절할 수 있는 방법을 제공하지 않으므로 흔히 사용하는 LED의 밝기 조절이나 모터의 속도 조절에서는 문제가 없겠지만 다른 장치를 제어하고자 하는 경우에는 직접 레지스터를 조작하여야 할 수도 있다.

아두이노 우노 보드에서 PWM 신호 출력을 위해 사용할 수 있는 핀은 표 1에 나타난 것처럼 3, 5, 6, 9, 10, 11번 핀이다. 이들 핀에는 핀 번호 앞에 ‘~’ 기호가 붙어 있으므로 쉽게 구별할 수 있다.



그림 14. 아두이노 우노 보드에서 PWM 신호 출력 가능 핀³⁾

코드 4는 analogWrite 함수를 이용하여 LED의 밝기를 조절하는 예를 보인 것이다. LED는 PWM 출력이 가능한 디지털 11번 핀에 연결된 것으로 가정한다.

코드 4. analogWrite

```
int LED_pin = 11;                // LED 연결 핀

void setup()
{
    pinMode(LED_pin, OUTPUT);    // LED 연결 핀을 출력으로 설정
    analogWrite(LED_pin, 0);     // LED는 끈 상태에서 시작
}

void loop()
{
    for(int i = 0; i < 256; i++){ // 점차 밝아짐
        analogWrite(LED_pin, i);
        delay(10);
    }

    for(int i = 255; i >= 0; i--){ // 점차 어두워짐
        analogWrite(LED_pin, i);
        delay(10);
    }
}
```

3) 이미지 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>