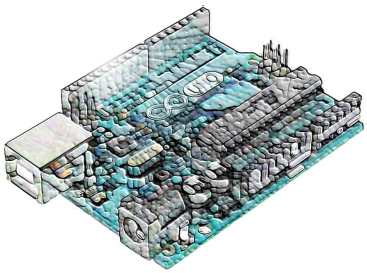

* 본 자료는 서경대학교 아두이노 프로그래밍 수업 수강자를 위해 작성된 강의 교재입니다.

강의교재– 아두이노 프로그래밍

9장 PWM과 DC 모터 제어

서경대학교 김진헌

9

PWM과 DC 모터 제어

내용

9.1 펄스폭 변조(PWM)

디지털 데이터는 0과 1의 논리값으로 표현한다. 논리 0의 실제 값 즉, 물리적인 전압값은 0V를 의미하는 것이고 논리 1은 물리적인 전압 5V를 의미한다. 따라서 디지털 데이터로는 0V 혹은 +5V 두 가지 물리적 전압만을 표현할 수 있다. 0V~5V 중간에 존재하는 아날로그 전압을 디지털 회로로 출력하는 방안이 없을까?

디지털 값만으로 아날로그 전압을 제어한 것과 같은 효과를 내게 하는 것이 PWM(Pulse Width Modulation)이다. PWM을 LED의 밝기 제어에 적용하는 사례를 들어 보자. LED에 디지털 데이터 1과 0을 제공하되 1의 값이 제공되는 시간이 0이 제공되는 시간의 비율을 다르게 제어하는 경우를 생각해 보자. 이 경우 LED에 공급하는 H의 폭을 넓게 하거나 좁게 하여 LED 전구를 밝게 하거나 어둡게 할 수 있다. 전체적으로 보면 전구의 밝기는 H로 공급한 시간의 펄스의 폭에 의해 결정되기 때문이다.

디지털 값 1과 0으로 이루어진 펄스(Pulse) 신호에서 1과 0의 비율을 조절하는 신호 변조 작업을 PWM이라고 한다. PWM은 디지털 신호로 아날로그의 전압 제어 혹은 전류 제어와 유사한 효과를 얻을 수 있는 특징이 있다.

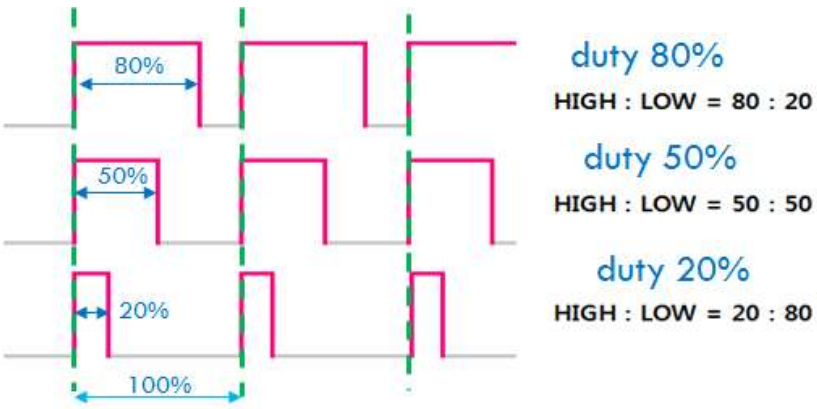

그림 9.1.1 3개의 신호: 주기 같음. 듀티비 다름.

한 주기의 신호에서 전제 주기 중에서 High Pulse가 차지하는 폭의 비율을 듀티비(duty rate)라 한다. 그림 9.1.1에 보인 3개의 디지털 펄스 신호는 같은 주기를 갖고 있으면서 각각 80%, 50%, 20%의 다른 듀티비를 갖고 있다. 다르게 표현해 보면 듀티 비율을 바꾸는 작업을 PWM이라고 말할 수 있다.

듀티비는 전자 장치에 제공하는 전력량을 제어하는 것과 같은 효과를 얻을 수 있다. 듀티비의 제어는 전원을 공급하는 시간을 제어하는 것과 같은 효과를 얻기 때문이다[1]. 직류 모터의 회전 속도는 전압의 높고 낮음으로 제어할 수 있는데, 전압을 제어할 수 없다면 전압 제공시간을 PWM의 듀티 비율을 조절하여 모터의 속도를 제어하는데 활용할 수 있다.

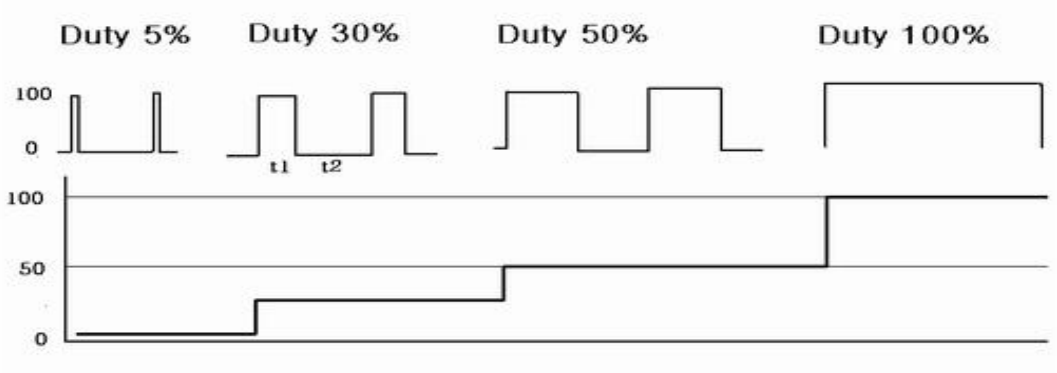

그림 9.1.2 듀티비의 변화에 따른 전력 제어 효과

그림 9.1.2에서 4가지의 듀티비의 사례에 따른 전력량을 보였다. 전체적으로 보면 일정 기간 동안 High로 제공된 면적이 커질수록 모터에 제공되는 전력량이 많아지기 때문에 모터는 고속회전을 한다고 말할 수 있다.

1) 전력량=전압×전류×시간. 전압, 전류는 고정값이고, 전압이 공급되는 시간을 듀티비가 제어하는 셈이므로 전력량을 제어한다고 말할 수 있다.

9.2 아두이노의 PWM 제어 표준함수

아두이노에서는 다음과 같이 특정 단자에 지정하여 원하는 듀티비의 PWM 신호를 출력하는 함수를 제공한다. 이 함수는 타이머[2] 장치를 통해 CPU가 설정한 파형을 자체적(H/W)으로 생성되게 한다. 이 때문에 PWM 신호 출력은 타이머에 연결된 특정 단자에서만 실행된다.

void analogWrite(pinNum, value)		
기 능	타이머 장치로 하여금 지정된 단자에서 지정한 듀티비의 PWM 신호를 송출하게 한다. 출력된 PWM 신호의 주파수는 대다수 490Hz인데 Uno의 5번 6번 단자[3]는 대략 980Hz의 주파수로 출력한다. 이 함수를 호출하기 전에 pinMode() 함수를 호출하여 해당 단자를 출력모드로 설정할 필요는 없다.	
매개변수	uint8_t pinNum	PWM을 출력할 아두이노 단자 번호 PWM이 가능한 단자 번호를 선택해야 한다. **아두이노 메가의 경우 : 2~13, 44~46** **아두이노 우노의 경우 : 5, 3, 6, 9, 10, 11**
	uint8_t value	듀티비 : 0(0%, off) ~ 255(100%, 항상 ON) 0~255 사이의 값으로 듀티 비율을 제어한다. 255이면 펄스 제공없이 계속 H만 출력된다. 이때는 H의 듀티비가 가장 높아서 가장 강한 전력을 공급할 수 있다.
리턴 값	void	없음

2) 타이머는 일반적으로 원하는 주파수의 펄스를 출력하는 장치이다. AVR은 타이머를 통해 주파수뿐만 아니라 듀티비도 제어할 수 있다.
3) 아두이노 공식 사이트에서는 단자 5, 6번의 경우에는 millis()와 delay() 함수에 사용되는 타이머의 채널과 공용으로 사용되기 때문에 analogWrite() 함수의 PWM 출력 듀티비가 기대치보다 좀 높게 나타날 수 있으며, 0으로 설정해도 0으로 off되지는 않는다고 밝히고 있다.

- value = 0
  - duty 0. high 펄스 없이 0만 출력
- value = 64
  - duty rate = 64/255 = 25.1%
- value = 127
  - duty rate = 127/255 = 49.8%
- value = 191
  - duty rate = 191/255 = 74.9%
- value = 255
  - duty rate = 255/255 = 100%

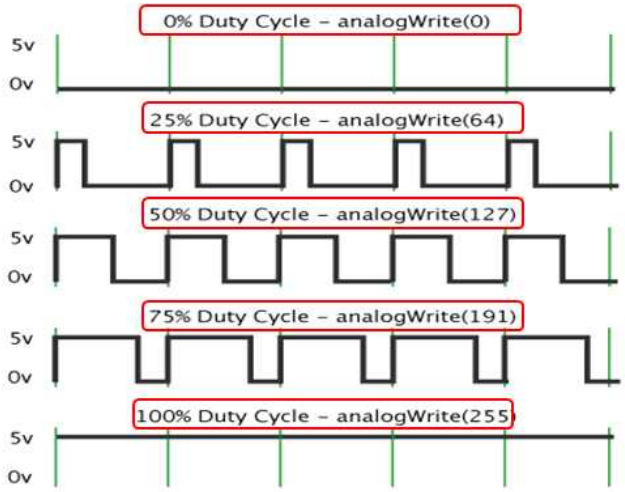

그림 9.2.1 analogWrite() 함수로 생성한 PWM 파형

다양한 value 값에 대한 듀티비와 파형을 그림 9.2.1에 보였다. 이 신호는 모두 같은 주파수(490Hz)와 주기(2.04ms)를 갖고 있다.

9.3 DC 모터의 구조와 동작원리-참고용

=> 참고용으로 기술합니다.

❒	모터(Motor)란?

모터는 전류와 자기력을 이용하여 생긴 힘을 이용해 회전 구동력을 얻는 장치이다. 본 절에서는 브러쉬가 있는 직류 전동기를 중심으로 모터의 구조와 동작에 대해 알아본다.

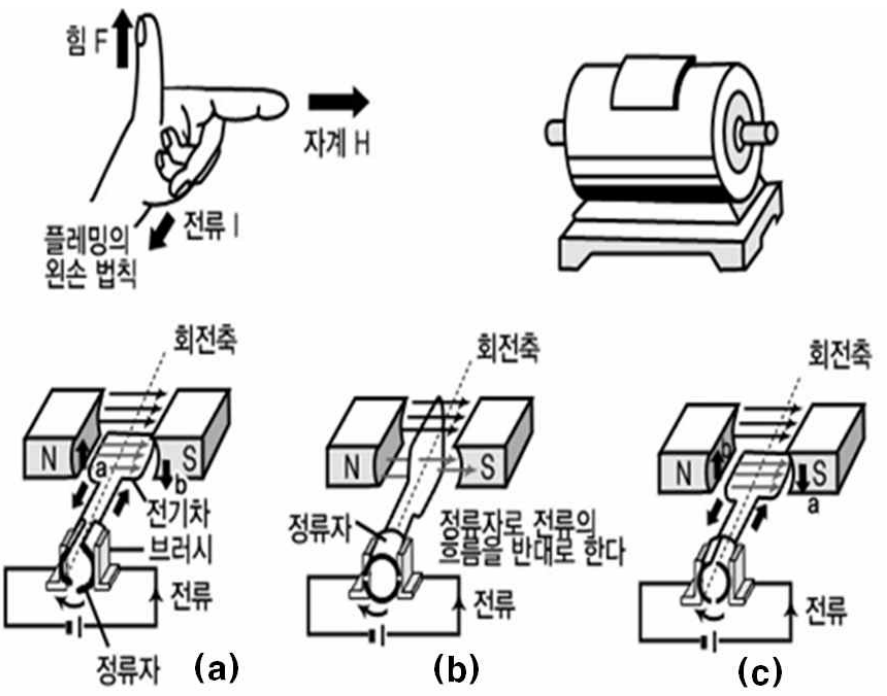

그림 9.3.1 정류자가 있는 직류 모터의 동작설명도

모터의 회전력은 플레밍의 왼손법칙으로 설명할 수 있다. 이는 자기장 속

에 있는 도선에 전류가 흐를 때 자기장의 방향과 도선에 흐르는 전류의 방향으로 도선이 받는 힘의 방향을 결정하는 규칙이다. 왼손의 검지를 자기장의 방향, 중지를 전류의 방향으로 했을 때, 엄지가 가리키는 방향이 도선이 받는 힘의 방향이 된다. 그림 9.3.1을 이용해 이를 설명하면 다음과 같다.

▣ 단계 1 : (a)에서 회로에 전류를 흘려보내면 DC 모터에서 연결된 브러시(brush)에서 정류자를 통해서 모터 내부로 전류를 보낸다.

N극 근처의 코일을 전류는 전방으로 흘러나온다, 그리고 자기력은 N->S로 흐르니 플레밍의 왼손 법칙대로 하면 힘은 위쪽 방향으로 작용한다. 이 내용은 N극과 접해있는 코일 a의 힘의 방향이 위쪽을 향한 것으로 표현되어 있다.

S극 근처의 코일을 보면 보면 전류는 후방으로 흘러들어간다. 그리고 자기력은 N->S로 흐르니 플레밍의 왼손 법칙대로 하면 힘은 아래로 작용한다. 이 내용은 S극과 접해있는 코일 b의 힘의 방향이 아래 쪽을 향한 것으로 표현되어 있다.

N극 쪽의 코일은 힘이 위로 향하고 S극 쪽의 코일은 아래로 작용하니 모터의 축은 시계 방향으로 회전하게 된다.

▣ 단계 2 : (b)의 그림에 보인 바와 같이 모터가 회전하면 코일에 연결된 정류자도 함께 회전한다. 이 때문에 코일에는 반대 방향으로 전류가 흐르게 되어 코일부는 (a)에서와는 반대 방향으로 힘을 받게되어 역시 시계방향으로 회전하게 된다.

▣ 단계 3 : (c)의 상황은 다시 단계 1의 처음 상황과 같게 되어 계속 시계방향으로 회전한다.

만약 초기에 흘려주는 전류의 방향을 반대로하면 역 시계 방향으로 회전한다. 또한 인가하는 전류의 양을 많게 하거나 전압을 높이면 속도가 빨라지고 그 반대이면 속도가 늦어진다. 여기서는 이것과 같은 효과를 얻기 위해

PWM의 듀티비를 제어한다.

참고로 그림 9.3.1은 전류의 흐름을 바꾸어 주는 정류자와 브러시의 세부 구조를 그림으로 보인 것이다. 이러한 구조의 DC 모터는 정류자와 브러시간의 마찰 때문에 정기적인 교환이나 유지 보수가 필요하며 접촉 스파크 노이즈가 발생할 수 있는 단점이 있으나 저렴하고 소형화가 용이한 장점이 있다.

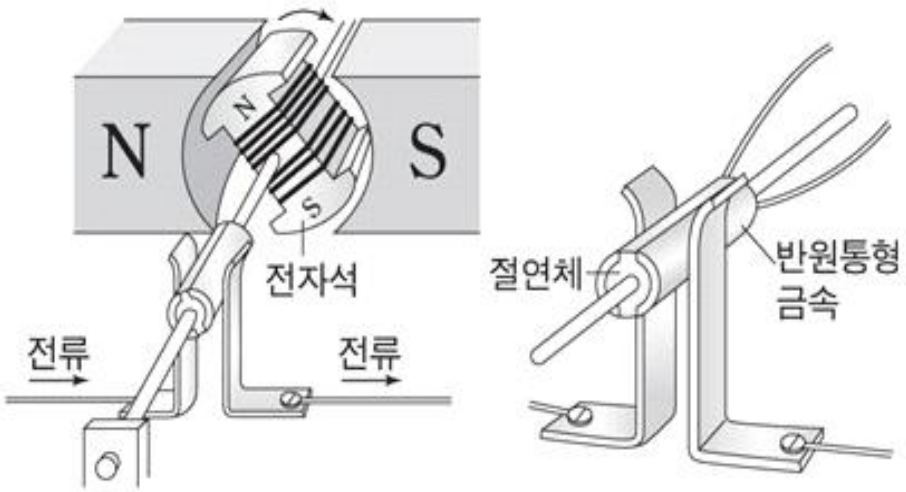

그림 9.3.1 정류자와 브러시의 세부 그림

9.4 DC 모터 드라이버

그림 9.4.1은 실험 키트(Ardu-Ez)에 포함된 DC 모터를 구동하기 위한 DC 모터와 모터 드라이버와 바퀴 구성품이다.

그림 9.4.1 DC 모터 제어를 위한 구성품

인터페이스 모듈에는 모터 드라이버 칩이 내장되어 있다. 이 칩은 약한 전류(약 10mA)의 PWM 신호를 입력받아 모터를 구동시킬 수 있는 큰 전류의 신호로 만들어 내는 역할을 한다. 또한, 전류의 방향을 바꾸어 역방향으로 회전하게 하거나 내부의 코일을 제어하여 회전 중의 모터를 급정지시키는 브레이크 동작을 지원한다.

이 인터페이스 모듈은 그림 9.4.2의 회로도에 보인 바와 같이 아두이노의 단자 7번과 8번에 연결되어 있다.

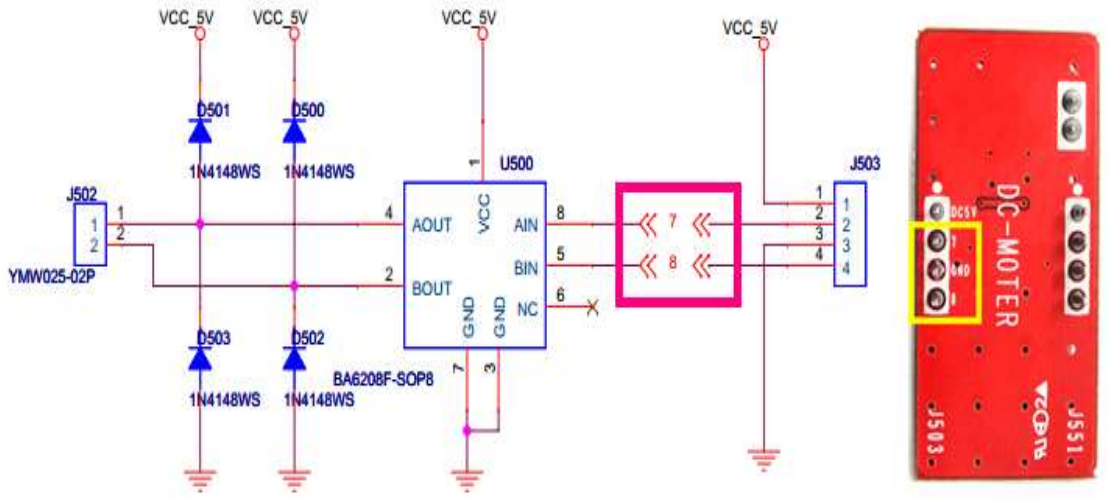

그림 9.4.2 DC 모터 인터페이스 모듈 및 회로도

아두이노에서는 DC 모터 인터페이스 모듈과 다음의 방식으로 교신한다.

<table>
<tr><td>연결 방식</td><td>PWM</td></tr>
<tr><td>아두노의
연결 단자</td><td>7번 단자(PWM) : 정 방향(AIN)
8번 단자(PWM) : 역 방향(BIN)
<table>
<tr><th>AIN</th><th>BIN</th><th colspan="2">동작</th></tr>
<tr><td>PWM</td><td>0</td><td>정방향 회전</td><td rowspan="2">해당 단자를 통해 최대 100% 듀티비의 PWM 신호를 제공한다.</td></tr>
<tr><td>0</td><td>PWM</td><td>역방향 회전</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">브레이크 정지(역 전원 제공. 강제 정지)</td></tr>
<tr><td>0</td><td>0</td><td colspan="2">정지. 전원 차단으로 인한 자연 정지</td></tr>
</table></td></tr>
<tr><td>동작 모드</td><td>- PWM 신호에 따라 모터 드라이버를 통해 전원을 제공
- 듀티비가 높아짐에 따라 회전 속도가 증가한다.</td></tr>
</table>

9.5 DC 모터 제어(드라이버 기반)

예제(1) - 모터의 속도 제어(analogWrite()함수를 이용)

analogWrite() 함수의 value 인자의 값을 증가시키면 PWM의 듀티비가 증가한다. 이 때문에 모터에 제공되는 전원의 공급 시간이 증가하여 모터의 회전 속도가 증가하게 된다.

❒ 예제 1 : DC 모터의 속도를 연속적으로 증가시켜 본다.

DCM_1.ino : analogWrite() 함수를 이용한 모터 속도 제어

```
01  #define DIR_F 7 // forward direction output
02  #define DIR_R 8 // reverse direction output
03  void setup() {
04    pinMode(DIR_R, OUTPUT); digitalWrite(DIR_R, 0);
05  }
06  void loop( ) {
07    int i=0;                          //- DC 모터 속도 값 저장 변수
08    for(i=0;i<=255;i++)               //- 0 ~ 255 범위
09    {
10      analogWrite(DIR_F, i);
11      delay(100);
12    }
13  }
```

예제(2) - 정지 및 브레이크 동작(digitalWrite() 함수 이용)

stop 동작은 모터 회전 중 전원을 차단하여 모터를 자연스럽게 정지하게 한다. brake는 역전압을 인가하여 급격하게 정지시킨다. 이 때문에 모터에

반동이 느껴지는데 실험 중 잘 관찰해 보기 바란다.

❒ 예제 2 : 5초 간격으로 DC 모터를 구동 & 정지를 반복시킨다.

DCM_2.ino : digitalWrite() 함수를 이용한 모터 구동 실험

```
01  #define DIR_F   7      // forward direction output
02  #define DIR_R   8      // reverse direction output
03  void setup() {
04    //pinMode(DIR_F, OUTPUT);   pinMode(DIR_R, OUTPUT);
05  }
06  void loop( ) {
07        analogWrite(DIR_F, 200);   analogWrite(DIR_R, 0);
08        delay(5000);          // forward rotation during this delay
09        digitalWrite(DIR_F, HIGH);      digitalWrite(DIR_R, HIGH);
10        delay(1000);           // brake
11        analogWrite(DIR_F, 0);     analogWrite(DIR_R, 200);
12        delay(3000);      // reverse rotation during this delay
13        digitalWrite(DIR_F, LOW);      digitalWrite(DIR_R, LOW);
14        delay(1000);           // stop
15  }
```

9.6 Tinkercad 기반의 실험

실험(1) - LED의 밝기 제어(analogWrite() 함수 이용)

본 실험에서는 9번 단자를 PWM 출력단자로 설정하였다. 그림 9.6.1에 보인 바와 같이 PWM 단자의 출력에는 ~ 기호가 마킹되어 있다.

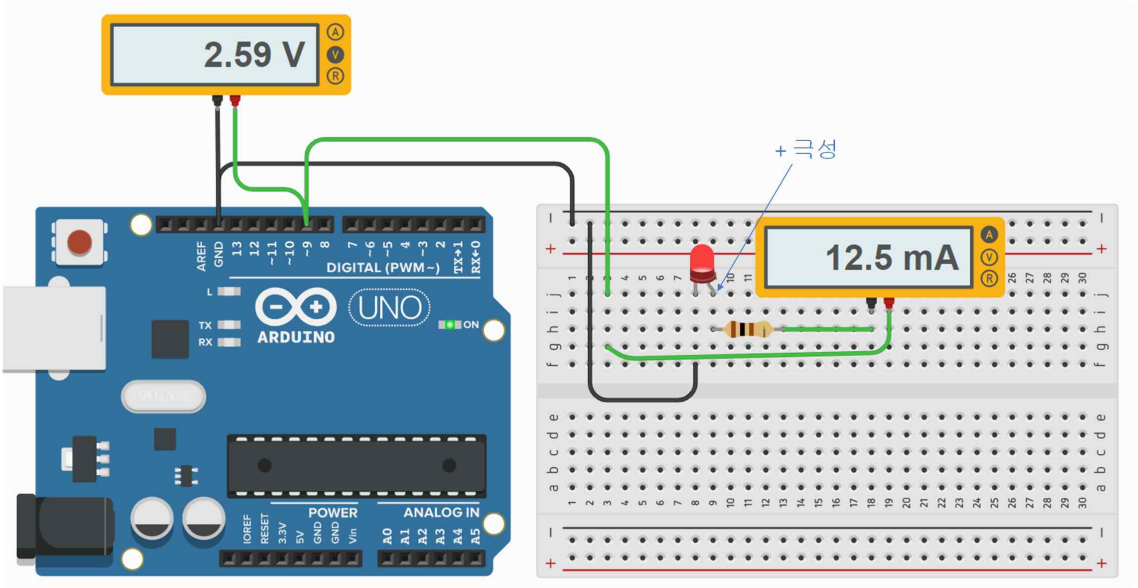

그림 9.6.1 PWM 기반 LED 밝기 제어 결선도(전압, 전류 측정)

그림에서는 PWM 신호의 전력량변화가 아날로그 소자에서는 어떤 효과로 작용하는지 알아보기 위해 전압계와 전류계를 부착하였다. 전류계는 부하(Load, 여기서는 LED)와 직렬로 연결하며, 전압계는 전원과 병렬로 연결하여야 한다.

그림에서는 본 것은 일순간의 전압과 전류를 보인 것이다. 원래 전압은 5V 혹은 0V가 제공되는 것이 원칙이지만, 매우 짧은 시간 이것을 반복하면 H를 제공한 시간만큼 전압이 인가되는 것처럼 느껴진다. 시뮬레이션 실험에 따르면 전압은 0V~4.02V, 전류는 0mA~19.5mA가 관측되었다.

여기서 알 수 있는 중요한 사실은 PWM의 듀티 변화가 전압계에서는 전압 변화로 관측된다는 것이다. PWM 신호는 자체로는 0V, 5V의 교번 전압을 출

력하지만 아날로그 소자에게는 H 펄스의 폭이 전압 변화로 작용한다는 것이다. 여기서 한 가지 주의할 것은 듀티비에만 변화를 준다고 항상 전압 변화로 나타나는 것은 아니라는 것이다. 만약 주파수가 지나치게 낮으면 그냥 깜박거리는 신호로도 인지될 수도 있다. PWM이 성공하려면 아날로그 소자가 느끼기에 연속적인 신호인 것처럼 충분히 주파수가 높아야 한다는 전제가 있다.

마치 영화가 수십 장의 정지된 영상으로 구성되어 있음에도 우리가 그것을 정지된 사진이 나열된 것으로 보지 않고 움직이는 사진으로 느끼는 것과 유사한 원리이다. frame per second(FPS)가 낮으면 동영상으로 느끼기가 어려울 것이다.

❒ 예제 9.6.1 : PWM신호를 기반으로 LED에 공급되는 전력량을 제어하여 LED의 밝기를 제어한다. 더불어 이때 인가되는 아날로그 전압과 전류를 측정한다.

PWM_LED.ino : analogWrite() 함수를 이용한 LED의 밝기 제어 실험

```
01  // 아두이노 우노의 PWM 단자는 -혹은 ~ 마크가 되어 있다.
02  // => PWM 전용 단자는 3, 5, 6, 9, 10, 11
03  #define PWM_PIN 9
04  int brightness = 0;
05  void setup() {
06    // PWM 단자는 출력 모드로 설정 안해도 됨.
07    // pinMode(PWM_PIN, OUTPUT);
08  }
09  void loop() {
10    for (brightness = 0; brightness <= 255; brightness += 5) {
11      analogWrite(PWM_PIN, brightness);
12      delay(50); // Wait for 30 millisecond(s)
13    }
14    delay(3000);
15    for (brightness = 255; brightness >= 0; brightness -= 5) {
16      analogWrite(PWM_PIN, brightness);
17      delay(50); // Wait for 30 millisecond(s)
18    }
19    delay(3000);
20  }
```

실험(2) - 오실로스코프 장비를 이용한 PWM 신호 관찰

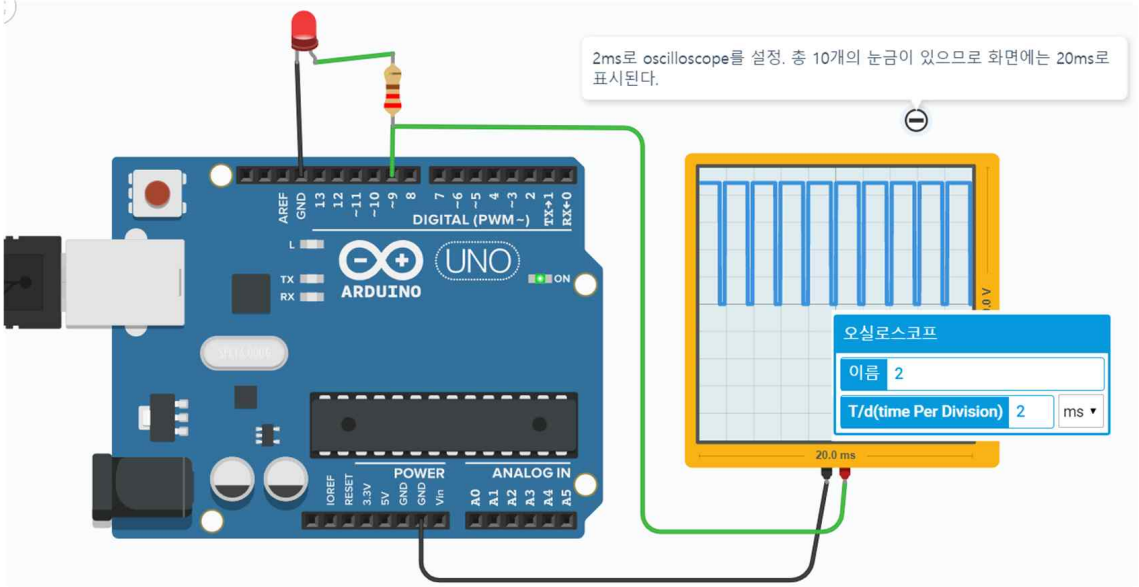

그림 9.6.2 오실로스코프 장비를 이용한 PWM 신호의 측정

❒ 예제 9.6.2 : 오실로스코프를 장착하고 PWM 신호의 파형을 관찰한다.

PWM_.ino : analogWrite()로 일정한 듀티비만 출력하고 파형을 관찰한다.

```
01   #define PWM 9
02   void setup() {
03     //analogWrite(PWM, 255/4);       // 실험 (a)
04     analogWrite(PWM, 3*255/4);       // 실험 (b)
05   }
06
07   void loop() {
08   }
```

analogWrite() 함수로 생성한 PWM 신호의 변화하는 모습을 오실로스코프(oscilloscope) 장비를 이용하여 관찰하였다. 이 계측 장비는 보통 주기적 신호에 대해 관찰이 용이하다. 장비의 특성상 동기 신호에 대해 진폭과 주기를 점검할 수 있는 장비이기 때문이다. 마침 아두이노 우노의 PWM 주파수가 일정(490Hz)하므로 동기(sync)가 맞아 이 장비를 사용하기에 적정하다.

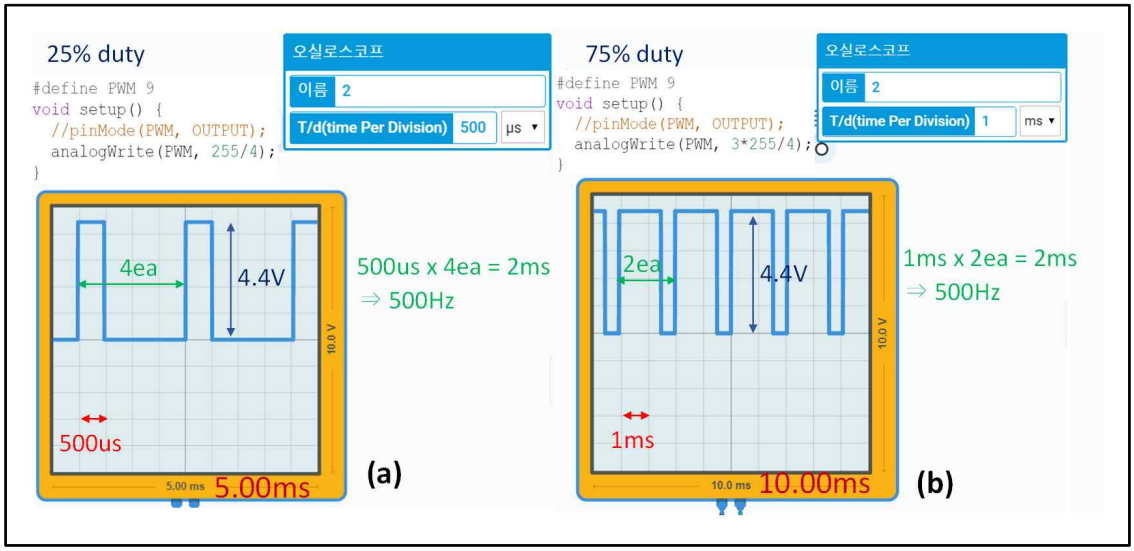

그림 9.6.3 PWM 신호의 측정 사례

그림 9.6.3에는 2개의 계측 사례를 보인 것이다. 두 사례 모두 setup() 함수 내에서 analogWrite() 함수로 특정 듀티비의 PWM 신호만을 출력하고, loop() 함수에서는 아무 동작도 시키지 않았다.

사례 (a)

(1) 25% 듀티비의 PWM 신호신호를 출력하였다. 따라서 H 펄스의 폭이 1 주기에서 25%를 차지하는 것이 관찰된다.

(2) 오실로스코프는 눈금을 500us로 설정하였다. 4개의 눈금으로 한 주기가 이루어지므로 500us x 4 = 2ms, 따라서 주파수는 500Hz이다. 아두이노 우노가 약 490Hz의 주파수로 출력되므로 대략 맞는 값이 관찰된다.

사례 (b)

(1) 75% 듀티비의 PWM 신호신호를 출력하였다. 따라서 H 펄스의 폭이 1 주기에서 75%를 차지하는 것이 관찰된다.

(2) 오실로스코프는 눈금을 1ms로 설정하였다. 2개의 눈금으로 한 주기가 이루어지므로 1ms x 2 = 2ms, 따라서 주파수는 500Hz이다. 아두이노 우노

가 약 490Hz의 주파수로 출력되므로 대략 맞는 값이 관찰된다.

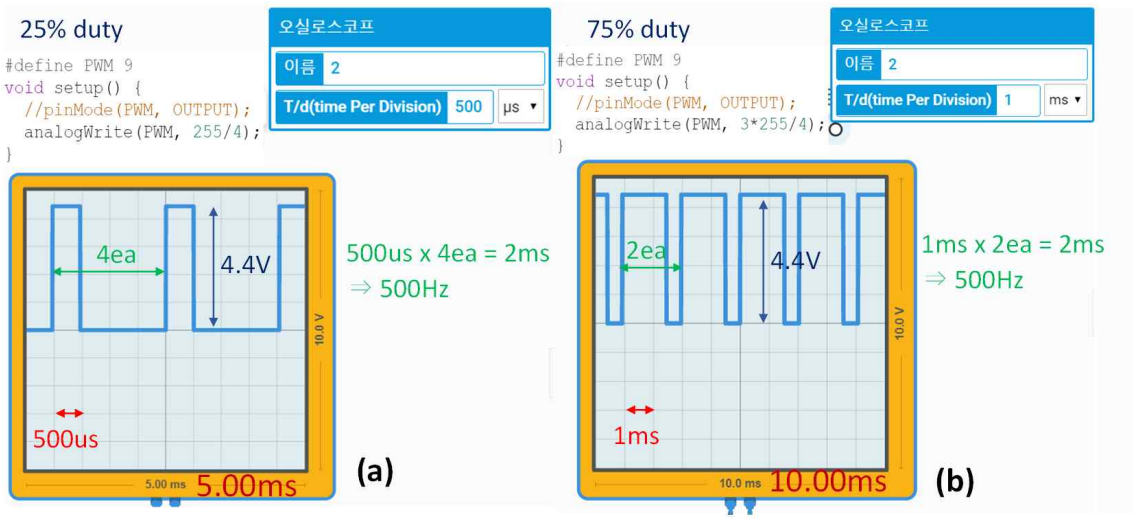

그림 9.6.3 PWM 신호의 측정 사례

실험(3) - RGB LED의 점등 제어 (digitalWrite(), analogWrite() 함수 이용)

RGB LED는 3개의 LED가 한 소자에 집산된 것으로 이해할 수 있다. RGB 단자 외에 공통 GND 단자가 제공된다. 본 실험에서는 그림 9.6.4에 보인 바와 같이 11, 10, 9번 단자를 RGB LED의 Red/Green/Blue 단자에 제공되는 출력 제어 단자로 설정하였다.

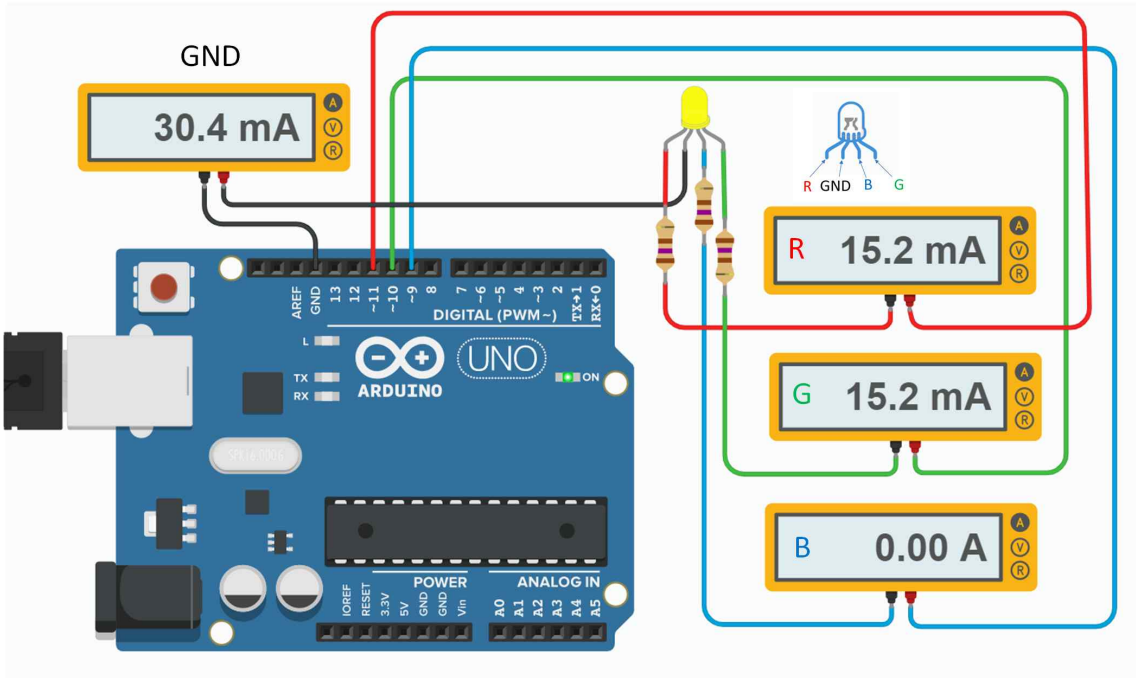

그림 9.6.4 RGB LED 밝기 제어 결선도(RGB 전류 측정)

같은 결선에 대해 실험은 GPIO와 PWM, 2가지로 행하기로 한다. 그림은 회로 결선과 더불어 GPIO로 R과 G 단자에 모두 논리 1의 출력된 순간을 포착하여 보인 것이다. R과 G에 모두 1이 출력되었으므로 "R+G=Yellow"로서 노란색이 출력된다. 신호선의 색상은 RGB LED의 색상과 같은 색으로 표기하였다. 먼저 digtalWite() 함수로 각 GPIO 출력단자에 각 단자에서 출력된 신호는 먼저 직렬로 연결된 전류계를 거쳐 저항을 통해 LED의 해당 신호 공급 단자로 인입된다. LED를 거친 전류는 저항을 GND로 흘러 들어간다. 그림에

서 보듯 R과 G의 전류계는 15.2mA가 각각 흐르는데 GND로 인입되는 전류계는 이들을 합한 30.4mA로 흘러 들어가는 것을 볼 수 있었다. 해당 예제 프로그램을 아래에 보였다.

❒ 예제 9.6.3 : RGB LED의 동작 원리를 보이기 위해 GPIO 기반으로 RGB LED의 색상을 제어한다. 더불어 이때 각 RGB 단자를 통해 흐르는 전류와 GND 단자를 통해 Ground로 흡수되는 전류를 측정한다.

RGB_LED.ino : digitalWrite() 함수를 이용한 RGB LED의 색상 제어 실험

```
01  // RGB LED의 색상 출력 실험
02  // R-> Y -> G -> C -> B -> -> M -> all On -> all off
03  # define LED_R 11
04  # define LED_G 10
05  # define LED_B 9
06  void setup() {
07    pinMode(LED_R, OUTPUT);
08    pinMode(LED_G, OUTPUT);
09    pinMode(LED_B, OUTPUT);
10  }
11  void loop() {
12    digitalWrite(LED_R, HIGH);          // Red
13    delay(1000); // Wait for 1000 millisecond(s)
14    digitalWrite(LED_G, HIGH);          // Yellow = R+G
15    delay(1000); // Wait for 1000 millisecond(s)
16    digitalWrite(LED_R, LOW);                    // Green
17    delay(1000); // Wait for 1000 millisecond(s)
18    digitalWrite(LED_B, HIGH);          // Cyan = G + B
19    delay(1000); // Wait for 1000 millisecond(s)
20    digitalWrite(LED_G, LOW);                    // Blue
21    delay(1000); // Wait for 1000 millisecond(s)
22    digitalWrite(LED_R, HIGH);          // Magenta = B + R
23    delay(1000); // Wait for 1000 millisecond(s)
24    digitalWrite(LED_G, HIGH);          // LED on
25    delay(3000); // Wait for 2000 millisecond(s)
26    digitalWrite(LED_B, LOW);                    // LED off
27    digitalWrite(LED_R, LOW);                    // LED off
28    digitalWrite(LED_G, LOW);                    // LED off
29    delay(3000); // Wait for 2000 millisecond(s)
30  }
```

실험(4) - 직류모터의 회전속도 제어(analogWrite() 함수 이용)

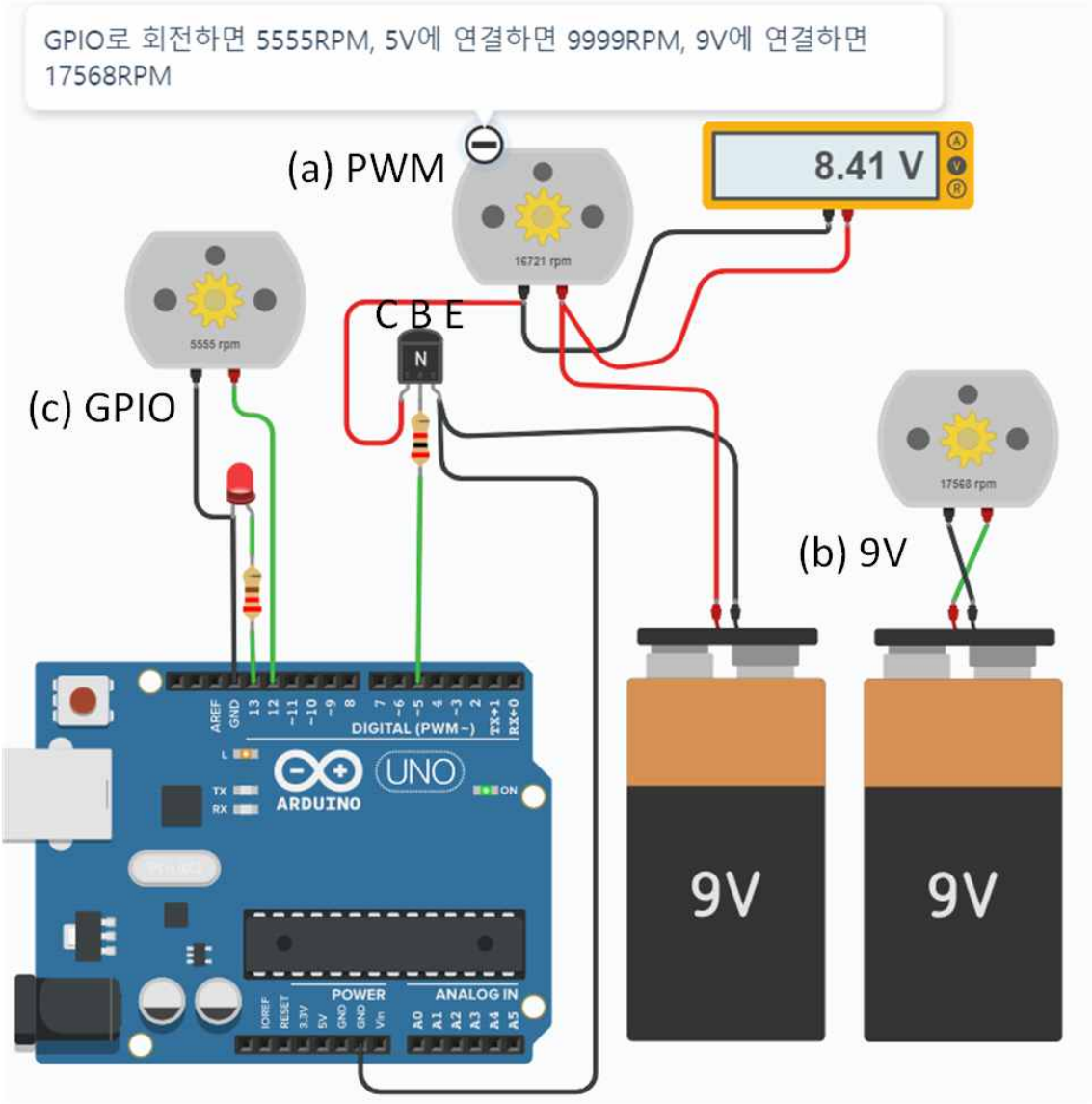

그림 9.6.5 DC 모터제어: (a) PWM 속도 제어, (b) 9V 정속, (C) GPIO-전력부족,

❒ 예제 9.6.4 : NPN 트랜지스터의 베이스 전압을 올리면 콜렉터로 인입되는 전류를 에미터로 빠져 나갈 수 있게 하여 부하에 공급되는 전압을 통제할 수 있다. 여기서는 베이스를 PWM 신호로 제어하여 모터에 인가되는 전압을 통제하여 모터의 회전 속도를 제어한다.

PWM_MOTOR.ino : 트래지스터를 활용한 PWM 신호의 모터 속도 제어 실험

```
01  # define          LED 13
02  # define          GPIO_PIN 12
03  # define          PWM 5
04  int i;
05  void setup() {
06    pinMode(LED, OUTPUT);
07    pinMode(GPIO_PIN, OUTPUT);
08  }
09  void loop() {
10    digitalWrite(LED, HIGH);
11    digitalWrite(GPIO_PIN, HIGH);
12    delay(2000);
13    for (i=0; i<256; i++) {
14      analogWrite(PWM, i);
15      delay(50);
16    }
17    delay(3000);
18    digitalWrite(LED, LOW);
19    digitalWrite(GPIO_PIN, LOW);
20    delay(2000);
21  }
```

그림 9.6.5에서 (a)만이 PWM을 기반의 모터제어로 적절하다.

(b)의 경우:

최고 전압(9V) 고정으로 인가하였으므로 건전지가 소모될 때까지는 회전 변화가 없다. 여기서는 9V가 인가되었을 때의 최고 RPM을 점검해볼 요량으로 장착하였다. 시뮬레이션 결과 17,569 RPM이 관찰되었다. 따라서 트랜지스터를 이용해 9V 전원을 통제하는 (a)의 경우에는 최고 회전속도가 이 정도일 것이다.

(c)의 경우

디지털 출력 장치 GPIO의 출력단으로 모터를 구동하기 때문에 전압은 5V이고, 전류는 20mA 이내일 것이다. 따라서 극히 소형 모터가 아닌 이상 이 정도의 전력으로는 회전하기 어렵다. 일반적으로는 GPIO에 모터를 연결한다면 모터가 GPIO가 내보낼 수 있는 모든 전류를 인출하기 때문에 프로세서를 포함한 전자회로가 작동을 정지할 가능성이 높다. 때로는 회로가 손상을 입을 수도 있다. 따라서 실제 상황에서는 시도하기를 권하지 않고 별로 실험 가치도 없는 일이라 본다. 시뮬레이션에 의하면 GPIO로도 5,555 RPM을 달성하였는데 모터가 그것을 감당할 만큼 극히 소형이라는 가정을 바탕으로 했을 것으로 이해하고자 한다.

(a)의 사례가 학습 가치가 있는 회로 구성이다. 이를 위해 잠시 부록 B.6의 트랜지스터 부분을 학습하고 돌아올 필요가 있다.

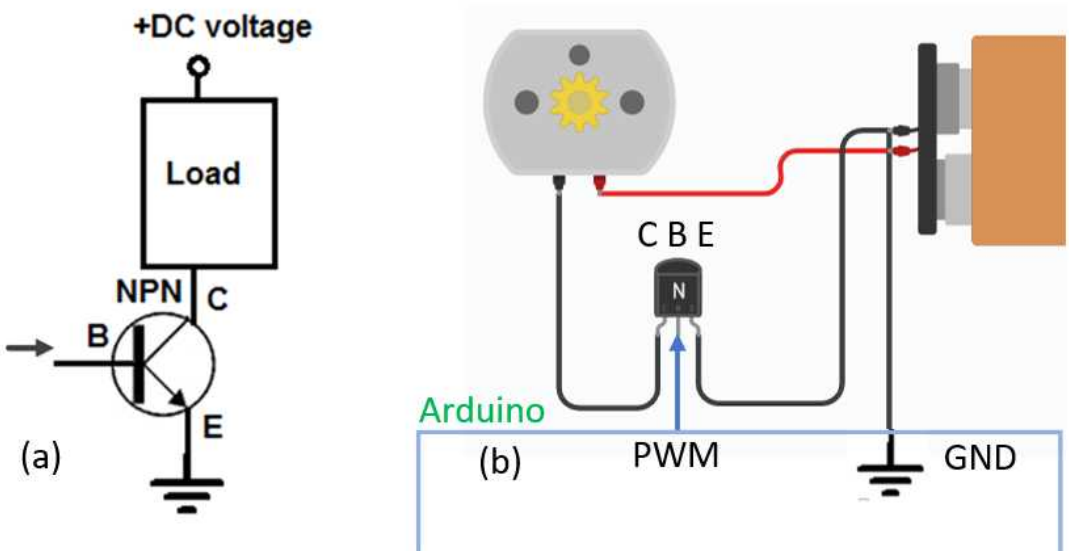

그림 9.6.6 TR을 이용한 증폭회로: (a) NPN TR을 이용한 전력제어 개념도, (b) PWM을 이용한 모터의 전력제어 배선도

그림 9.6.6에는 트랜지스터(TR)를 이용하여 부하(Load, 여기서는 모터)를 베이스(Base) 단자를 이용하여 제어하는 사례를 보였다. 실험(1)의 사례에서 보였듯이 PWM의 듀티 변화가 아날로그 소자에게는 전압 변화로 나타난다. 따라서 PWM 단자를 사용하여 베이스 전압의 높낮이에 해당하는 동작을 PWM 신호의 펄스 폭으로 대신하게 하는 것이 가능한 것이다.

주의할 것은 모터에 제공되는 전압은 외부(여기서는 9V 건전지)에서 공급되는데 외부의 GND 단자가 아두이노의 GND 단자와 서로 연결되어 있어야 원활한 신호의 전달이 이루어진다는 것이다.

실험(5) - Micro Servo Motor

그림 9.6.7 마이크로 서보와 신호선

마이크로 서보는 RC(Remote Control, 무선조종)에서 많이 사용되어 RC 서보라고도 한다. 대략 3~6V 정도의 전압에 기동 순간에 흐르는 순간 전류의 폭이 200mA~700mA로 좀 넓은 것 같다. 사진에 보인 것은 micro type이며, 좀 더 큰 것은 표준형 서보라고 하며 더 높은 전압에 더 많은 전류를 소비한다[4].

마이크로 서보는 PWM 신호를 기반으로 사용자가 지정한 회전 각도를 계속 유지하는 기능을 수행한다. 일반 서보 모터가 지정한 RPM의 속도로 계속 회전하는 것을 목표로 하는 것에 비해 크게 그 기능이 다르다. 마이크로 서보는 내부에 회전각도를 측량하는 potentiometer와 제어회로를 담당하는 chip이 내장되어 있어 나름 독립된 closed loop 제어를 행하고 있는 장치이다. 본 자료에서는 서보모터의 동작원리 보다는 이를 이용하는 측면에서의 내용을 중심으로 다루기로 한다.

4) 이런 마이크로, 스탠다드 서보는 PWM을 제어 신호로 사용한다. 수십 와트 이상의 서보 모터는 그 동작과 제어 원리/방법이 크게 다르다.

서보 제어는 아두이노 표준 함수에서 지원하기 때문에 따로 설치할 필요는 없다. 다음의 함수가 지원된다. 자세한 설명은 생략하기로 한다.

attach()
write()
writeMicroseconds()
read()
attached()
detach()

그림 9.6.8에는 2개의 마이크로 서버를 아두이노에 연결하는 배선도를 보였으며, 예제 9.6.5에는 이들을 제어하는 예제를 보인 것이다.

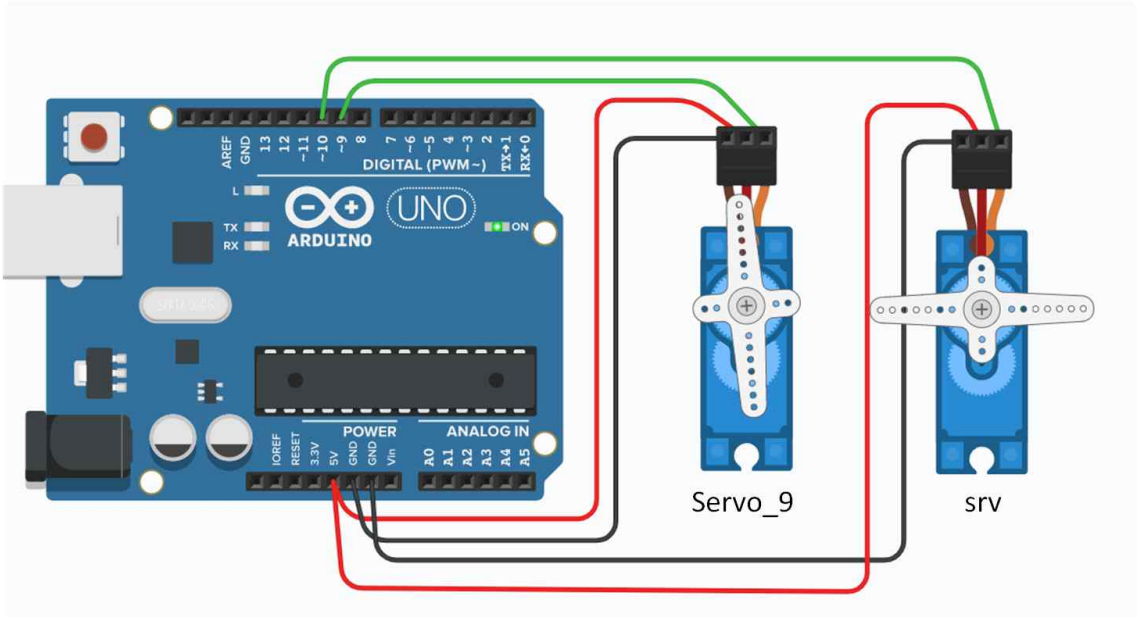

그림 9.6.8 마이크로 서보 연결도

❒ 예제 9.6.5 : 마이크로 서보 모터의 제어

PWM_SERVO.ino : Servo 라이브러리를 이용한 서모 모터의 제어

```
01  #include <Servo.h>          // Servo 라이브러리를 사전에 설치해야 한다.
02  Servo srv;                  // 서보 제어를 담당할 객체를 선언한다.
03  Servo servo_9;   // 편의상 핀번호를 지정할 수도 있다.
04  void setup() {
05    servo_9.attach(9);        // PWM 단자번호를 객체의 메서드(method)
06    srv.attach(10);           // 함수로 지정한다.
07  }
08  void loop() {
09    servo_9.write(0);         // 0도 각도의 위치로 지정
10    delay(3000);
11    srv.write(0);             // 0도 각도의 위치로 지정
12    servo_9.write(180);       // 180도 각도의 위치로 지정
13    delay(3000);
14    srv.write(180);
15  }
```

서보 모터를 제어할 때는 서보 객체를 먼저 선언해야 한다. 이는 Servo라는 클래스에 의해 이루어지며, 이때 선언되는 객체는 본 예제에서는 srv, servo_9로 하였다.

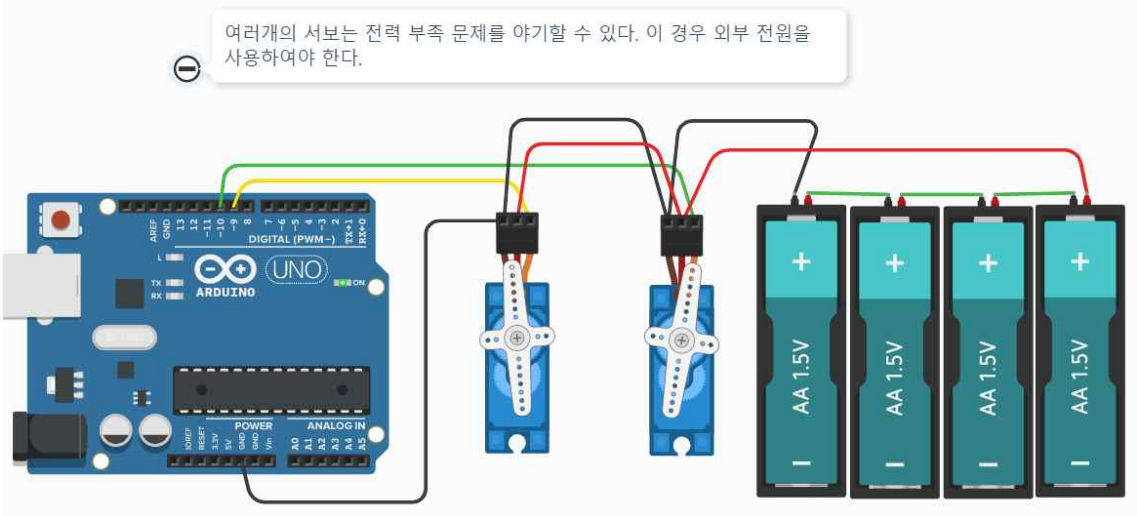

그림 9.6.9 마이크로 서보 연결도

USB 전력 공급이 5V, 0.5A로서 약 2.5W이다. 이것을 넘을 정도의 전력 소비가 일어나면 당연히 문제가 될 것이므로 전류를 많이 소비하는 표준 서보나 혹은 여러 대의 마이크로 서보를 연결할 때는 그림 9.6.9처럼 별도 전원을 제공해야 할 수도 있다. 이때는 외부 전원의 GND와 아두이노의 GND가 연결되어야 한다. 일반적으로 마이크로 서보는 4~6V 정도의 전압을 사용하므로 서보를 연결할 때는 서보의 사양을 잘 살펴야 할 것이다.

서보가 회전하지 않을 때 전류가 소비되는지 알아보기 위해 그림 9.6.10와 같은 회로를 결선하고 회전과 정지를 충분히 간격(4초 정도)을 두어 전류의 변화량을 관찰하였다[5].

실험에 의하면 예상한 대로 초기 기동할 때 많은 전류가 흐르다가 목표치에 가까워 지면 곧 2~3mA가 되는 것을 관찰할 수 있었다. 최고치는 화면에 보인 바와 같이 98mA이고 수십~수mA로 떨어지는 것을 관찰하였다. 이와 같이 정상적이라면 특정 각도에서 더 이상의 신호를 주지 않으면 전력 소모

5) 이 실험이 시뮬레이션에 불과하기 때문에 "사실"을 증명하는 것이냐 하는 질문에는 대답이 궁색한 측면이 있다. 그러나 적어도 이론적인 입장에서 보면 제법 꽤 합리적인 수치를 출력하는 것으로 판단된다.

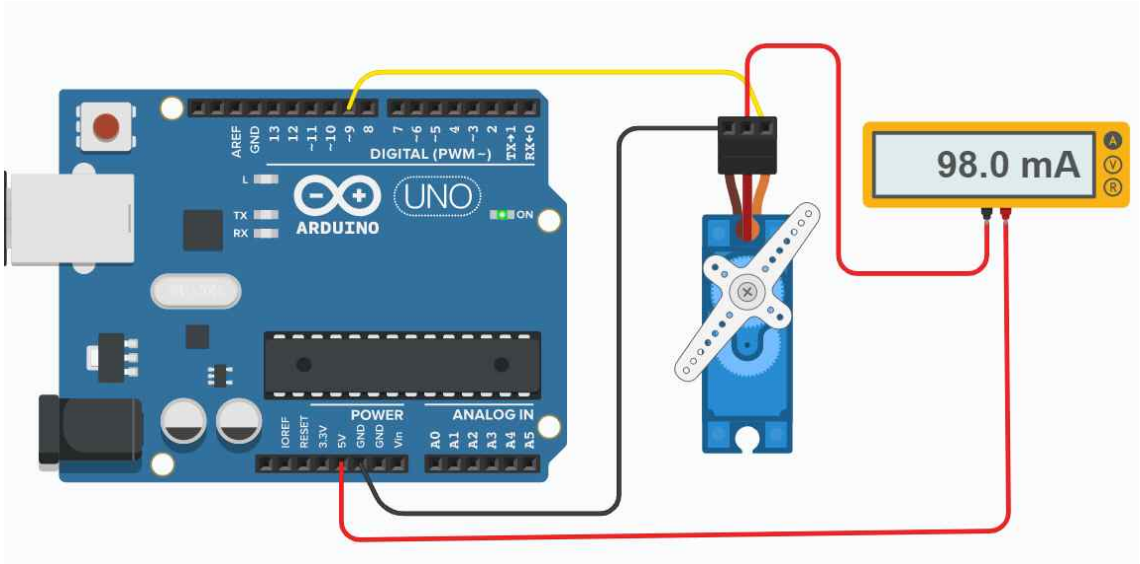

그림 9.6.10 마이크로 서보 연결도

가 거의 일어나지 말아야 한다.

그런데 실제로 실험을 해보면 모터가 특정 각도에서 정지시켰는데도 계속 진동하는 경우를 만날 수 있었다[6]. 이 현상을 jitter, buzz 등의 용어로 일컫는데 이는 전력 소모가 계속 일어나고 있는 것으로 판단되며, 심지어 열도 발생하는 것을 느낄 수 있다. 서보 내부에는 외부로부터 유입된 각도를 찾아가도록 제어기 칩이 내장되어 있다[7]. 이 제어기가 목표 각도에 돌입하지 못했다고 판단하여 스스로 정상 상태를 찾으려고 계속 순방향/역방향 회전하는 것을 멈추지 않는 것이다. 사실 이런 일은 제어회로 설계의 초기 단계에서 매우 흔한 일이다. 이 원인과 처방도 너무 다양해서 일일이 열거하기도 어려운데 아쉬운 대로 몇 가지만 나열하기로 한다.

- 전원이 부족해서 그럴 수 있다. 임시 처방으로 전하를 저장하는 커패시터(capacitor)를 전원에 연결한다. 이러면 최소한 초기 기동 전력 문제는 어느

6) 아닌 것도 있다. 목표 각도에 도달하면 전혀 진동없이 안정된 상태를 유지하는 서보도 많다. 사실 이것이 정상이다.
7) 참조: 서보 내부 모습, 서보 동작원리

정도 보완할 수 있다.

- 서보의 내부 센서가 고장일 수도 있다. 특히 저급한 사양의 서보는 매우 약한 내구성 혹은 출고 당시 튜닝 부실이 의심된다.
- 아쉬운 대로 사양(0~180도)을 100% 실현하지 말고 약간의 마진을 두어 운전하면 현상이 조금 나아질 수도 있다. 예를 들면 10~170도 정도만 운영한다는 것이다.
- 아래 그림과같은 Ferrite core에 신호선을 통과시켜 잡음을 제거하여 jitter를 해결하였다는 사례도 있다. URL 이 소자는 원형 링과 같이 생긴 자성체인데 전선이 이곳을 통과하면 고주파 신호성분을 잃지 않고도 외부의 전자장으로 유기되는 잡음을 제거하는 소자이다.

9.7 고찰

다음 주제에 대하여 답할 수 있는지 검토해 보면서 그동안 습득한 지식을 정리해 보자.

1. analogWrite() 함수에 대해서 : 이 함수는 analog value를 출력하는가?

2. analogWrite() 함수로 DC 모터의 속도를 제어하는 방법을 기술하시오.

3. analogWrite(pinNo, 255); digitalWrite(pinNo, HIGH); 두 개의 함수 중 DC 모터를 더 고속으로 회전시키는 함수는 어떤 것인가? 그 이유는? 어느 정도 빠를 것인지 그 결과를 예측하시오(가정 : 회전 속도는 인가된 전력 양에 비례한다).

4. tone() 함수로 DC 모터의 속도를 제어할 수 있는가? 그 이유 혹은 방법을 설명하시오.

5. digitalWrite() 함수만으로 DC 모터의 속도를 제어하고자 한다. 방법을 제시하시오.

6. analogWrite() 함수에 의한 PWM 신호는 어떤 장치가 생성하는가? GPIO로 활용 가능한 단자를 PWM 단자로 사용 가능한가?

7. 모터 드라이버 모듈 없이 DC 모터를 구동시키면 어떤 문제가 발생할지 기술하시오. 모터 드라이버 없이 DC 모터를 제어할 방안을 제시하시오.

8. analogWrite() 함수를 이용해 LED의 밝기를 제어할 수 있을까? 그 원리는 무엇인가?