

9. 4자리 7세그먼트 표시장치

이전 장에서 1자리 7세그먼트 표시장치를 제어하는 방법을 살펴보았다. 이외에서 2자리 이상의 숫자를 표시하기 위한 보다 7세그먼트 장치를 흔히 볼 수 있다. 1자리의 7세그먼트 표시장치를 제어하기 위해서는 최소 8개의 핀(8개의 세그먼트를 개별적으로 제어하기 위한 핀)이 필요하므로 두 자리를 제어하기 위해서는 16개, 세 자리를 제어하기 위해서는 24개 등 필요로 하는 핀의 개수가 증가한다. 아두이노 우노의 경우 입출력 핀의 개수가 20개이므로 이러한 계산에 의하면 세 자리 이상의 7세그먼트 표시장치를 제어하는 것은 불가능하다. 하지만 아두이노 우노로도 4자리 7세그먼트 표시장치를 제어하는 것이 가능하다. 이 장에서는 12개의 디지털 출력 핀만으로 4자리 7세그먼트 표시장치를 제어하는 방법을 알아본다.

1. 4자리 7세그먼트 표시장치

4자리 7 세그먼트 표시장치에는 4개의 숫자를 표현할 수 있다. 4자리의 숫자를 동시에 나타내기 위해서는 최소 32개의 핀이 필요하며 아두이노 우노로는 불가능하다. 하지만 4자리 숫자를 빠른 속도로 하나씩만 켜다만 잔상 효과에 의해 4자리 숫자가 동시에 켜지는 것과 같은 효과를 볼 수 있다. 다만 4자리의 숫자 중 특정 순간에 켜 자리를 결정하기 위해서는 각 자리에 해당하는 제어 신호가 추가로 필요하므로 4자리 7 세그먼트 표시장치의 경우 8개의 세그먼트 제어 핀과 4개의 자릿수 선택 핀, 총 12개의 핀으로 4자리 숫자를 표현할 수 있다. 2자리 숫자를 표현하는 7세그먼트 표시장치의 경우에는 개별적으로 제어하는 장치와 잔상 효과를 이용하여 공통으로 제어하는 장치가 판매되고 있지만 세 자리 이상의 숫자의 경우 개별적으로 제어하기 위해서는 핀의 수가 많아 개별적으로 제어하는 장치는 찾아보기 힘들다. 개별 제어 형태의 경우 1자리 7세그먼트 표시장치를 두 개 이상 사용하는 것과 동일하다.

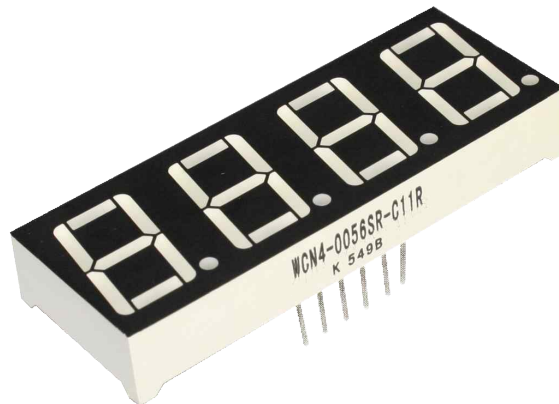


그림 1. 네 자리 7 세그먼트 표시장치

그림 1은 4자리 7 세그먼트 표시장치로 12개의 핀이 존재한다.¹⁾ 핀 번호는 1자리 7세그먼트 표시장치와 동일하게 소수점이 표시된 쪽의 왼쪽부터 반시계 방향으로 1번부터 증가한다. 1자리의 7세그먼트 표시장치를 제어하기 위해서는 최소 8개의 핀이 필요하므로 네 자리를 제어하기 위해서는 최소 32개의 핀이 필요할 것을 보이지만 그림 2의 회로도를 살펴보면 핀의 개수를 줄일 수 있는 비밀이 있다.

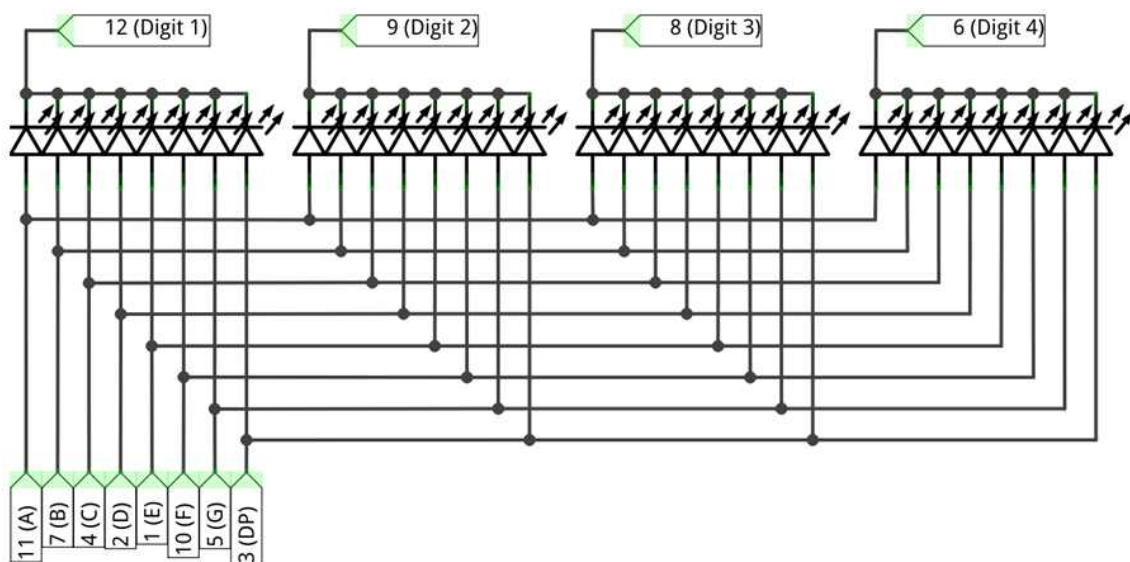


그림 2. 네 자리 7 세그먼트 표시장치 내부 연결 회로도

1) 네 자리 7 세그먼트 표시장치 중에는 중앙에 콜로(:)이 포함된 장치도 흔히 볼 수 있으므로 data sheet를 참조하여 핀의 개수 및 배치를 확인하여야 한다.

그림 2에서 알 수 있듯이 12번, 9번, 8번, 6번 핀은 네 자리 중 원하는 자릿수를 선택하기 위해 사용되고 8개 세그먼트에 해당하는 핀은 공통으로 사용된다. 이 장에서 사용한 7 세그먼트 표시장치는 공통 음극 방식이므로 12번 핀에 LOW를 가하고 8개 세그먼트 제어 핀에 0xFC(A, B, C, D, E, F, G, DP의 순서로 1비트씩 할당된 표현)를 가하면 첫 번째 자리(왼쪽부터 첫 번째 자리로 표현함)에 0이 표시된다. 이 때 9번과 8번 핀에 LOW를 가하면 동일한 숫자가 두 번째와 세 번째 자리에 표시된다. 따라서 특정 자리에 숫자를 표시하기 위해서는 해당 자리에 해당하는 핀은 단자는 LOW를, 나머지 자리에 해당하는 핀은 HIGH를 가해줘야 한다. 이처럼 여러 자리를 표시하기 위한 7 세그먼트 표시장치는 핀을 공유하고 있으므로 핀의 개수를 줄일 수 있는 장점은 있지만 서로 다른 숫자를 두 개 이상의 자리에 표시할 수 없는 한계가 있다. 예를 들어 '3333'은 표현 가능하지만 '1234'는 표현 불가능하다. 이를 해결하는 방법은 빠른 속도로 각 자릿수에 값을 출력하기를 반복함으로써 잔상 효과를 통해 4자리가 동시에 켜지는 것과 같은 효과를 얻는 것이다.

2. 4자리 7세그먼트 모듈

실험에 사용할 모듈은 공통 음극 방식의 4자리 7세그먼트 모듈이다. 그림 3은 4자리 7세그먼트 모듈의 회로도를 나타낸다.

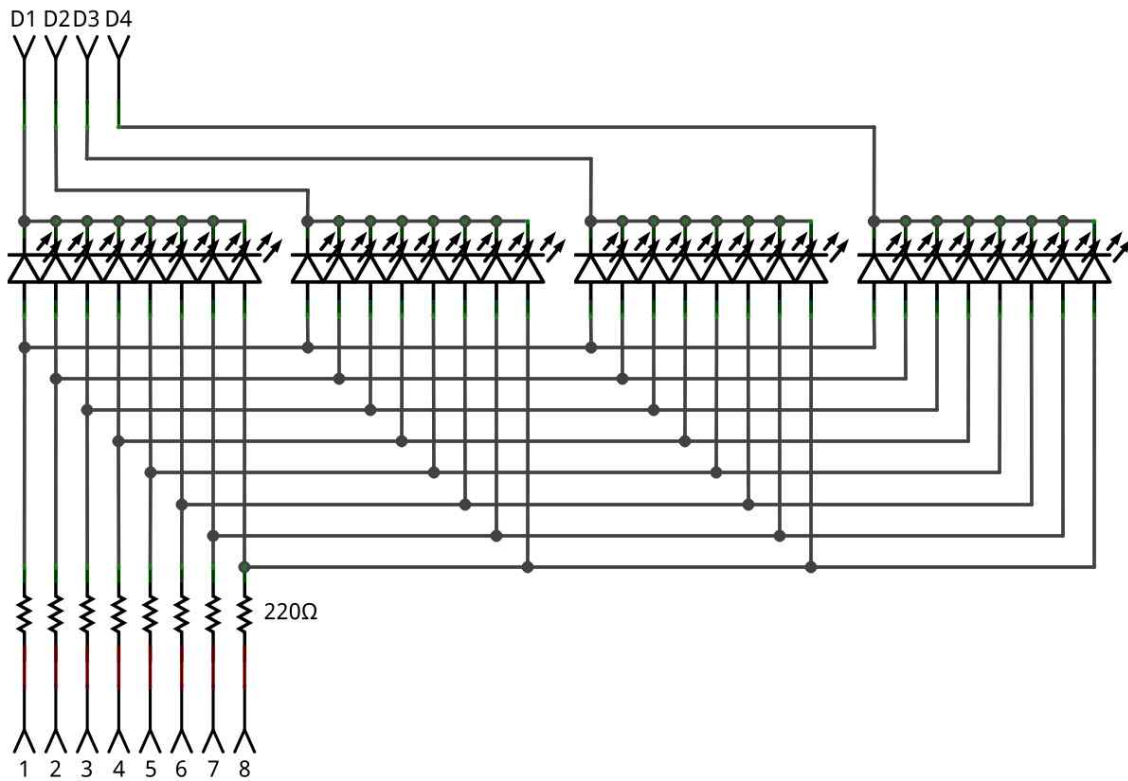


그림 3. 4자리 7세그먼트 모듈 회로도

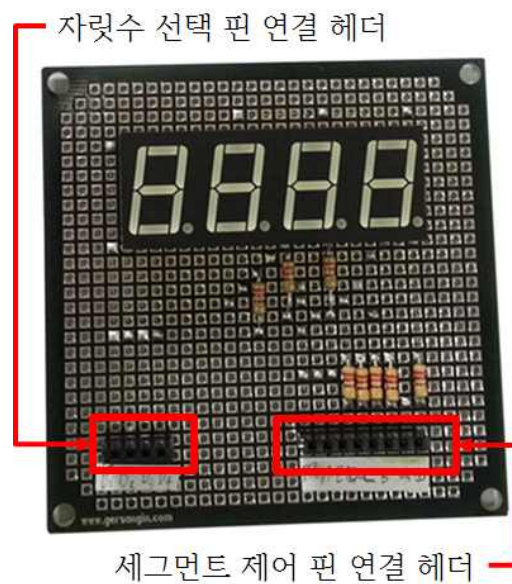


그림 4. 4자리 7세그먼트 모듈

3. 1자리 숫자 표현하기

4자리 7세그먼트 표시장치를 아두이노와 연결해 보자. 7세그먼트 제어 선은 디지털 2, 3, 5, 6, 7, 8, 10, 11번 핀에 연결하고 자릿수 선택을 위한 제어 선은 디지털 12, 13번 및 아날로그 A0, A1에 연결한다.

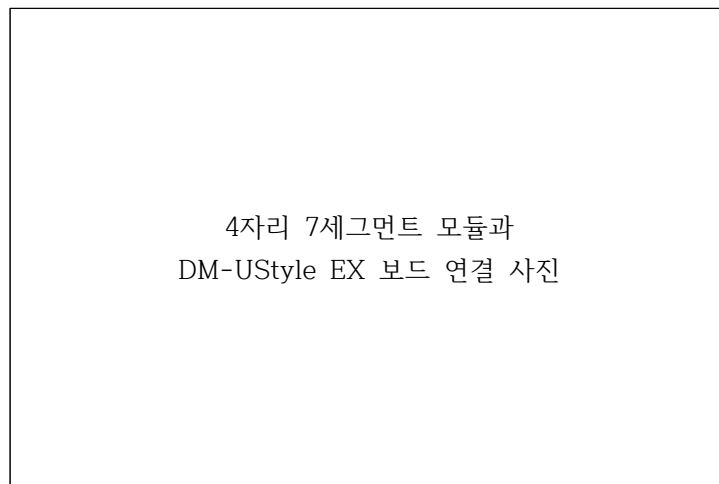


그림 5. 5자리 7세그먼트 모듈 연결

먼저 4자리 중 하나의 자리에 0에서 9 사이의 숫자를 표시해 보자. 자리는 D1에서 D4 중 원하는 자리에 해당하는 핀에만 LOW 값을 출력하고 나머지는 HIGH 값을 출력하면 된다. 숫자 표시를 위해서는 해당 세그먼트에 해당하는 제어 핀에 HIGH 값을 출력하면 켜지고 LOW 값을 출력하면 꺼진다. 1자리 7세그먼트 모듈은 공통 양극 형식인 것과 달리 4자리 7세그먼트 모듈은 공통 음극 형식이므로 제어 방식이 반대인 점에 유의하여야 한다. 스케치 1은 특정 자리에 특정 숫자를 하나만 표시하는 스케치의 예이다.

스케치 1. 4자리 7세그먼트에 숫자 한 자리 나타내기

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int digit_select_pin[] = {12, 13, A0, A1};           // 자릿수 선택 핀
```

```

int segment_pin[] = {2, 3, 5, 6, 7, 8, 10, 11};    // 7세그먼트 모듈 연결 핀

void setup(){
    for(int i = 0; i < 4; i++){                    // 자릿수 선택 핀을 출력으로 설정
        pinMode(digit_select_pin[i], OUTPUT);
    }
    for(int i = 0; i < 8; i++){                    // 세그먼트 제어 핀을 출력으로 설정
        pinMode(segment_pin[i], OUTPUT);
    }
}

// 해당 자리에 숫자 하나를 표시하는 함수
void show_digit(int pos, int number){             // (위치, 출력할 숫자)
    for(int i = 0; i < 4; i++){
        if(i + 1 == pos)                         // 해당 자릿수의 선택 핀만 LOW로 설정
            digitalWrite(digit_select_pin[i], LOW);
        else                                     // 나머지 자리는 HIGH로 설정
            digitalWrite(digit_select_pin[i], HIGH);
    }

    for(int i = 0; i < 8; i++){
        // 공통 음극 방식이므로 반전하지 않고 출력한다.
        // 1자리 7세그먼트 모듈과 반대이다.
        boolean on_off = bitRead(patterns[number], i);
        digitalWrite(segment_pin[i], on_off);
    }
}

void loop(){
    show_digit(1, 3);                             // 첫 번째 자리에 '3'을 출력
}

```

4. 4자리 숫자 표현하기

4자리 7세그먼트 표시장치에 숫자 4개를 표현해 보자. 스케치 1에서 loop 함수를 다음과 같

이 수정하고 실행해 보자.

```
void loop(){
  show_digit(1, 1);    // 첫 번째 자리에 '1'을 출력
  show_digit(2, 2);    // 두 번째 자리에 '2'를 출력
  show_digit(3, 3);    // 세 번째 자리에 '3'을 출력
  show_digit(4, 4);    // 네 번째 자리에 '4'를 출력
}
```

‘1234’가 표시되는 것처럼 보이지만 숫자가 겹쳐 보이거나 번져 보이는 현상이 발생한다. 한 자리를 출력한 후 다음 자리를 출력하는 사이에 시간 지연을 설정해 보자.

```
void loop(){
  show_digit(1, 1);    // 첫 번째 자리에 '1'을 출력
  delay(100);
  show_digit(2, 2);    // 두 번째 자리에 '2'를 출력
  delay(100);
  show_digit(3, 3);    // 세 번째 자리에 '3'을 출력
  delay(100);
  show_digit(4, 4);    // 네 번째 자리에 '4'를 출력
  delay(100);
}
```

0.1초 시간 간격을 주면 ‘1234’가 출력되기는 하지만 한 번에 하나씩만 표시된다. 지연 시간을 조절하면 한 번에 4개의 숫자가 동시에 표시되는 것을 확인할 수 있다. 하지만 실제로는 한 번에 하나의 숫자만 켜지는 것이며 적당히 빠른 속도로 숫자를 번갈아 표시함으로써 한꺼번에 표시되는 것과 같은 효과를 내는 것일 따름이다. 스케치 2는 4자리 이하의 숫자를 표시하는 스케치의 예이다.

스케치 2. 4자리 7세그먼트에 숫자 네 자리 나타내기

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int digit_select_pin[] = {12, 13, A0, A1};           // 자릿수 선택 핀
int segment_pin[] = {2, 3, 5, 6, 7, 8, 10, 11};      // 7세그먼트 모듈 연결 핀

int SEGMENT_DELAY = 5;                             // 숫자 표시 사이의 시간 간격

void setup(){
    for(int i = 0; i < 4; i++){                      // 자릿수 선택 핀을 출력으로 설정
        pinMode(digit_select_pin[i], OUTPUT);
    }
    for(int i = 0; i < 8; i++){                      // 세그먼트 제어 핀을 출력으로 설정
        pinMode(segment_pin[i], OUTPUT);
    }
}

// 해당 자리에 숫자 하나를 표시하는 함수
void show_digit(int pos, int number){               // (위치, 출력할 숫자)
    for(int i = 0; i < 4; i++){
        if(i + 1 == pos)                            // 해당 자릿수의 선택 핀만 LOW로 설정
            digitalWrite(digit_select_pin[i], LOW);
        else                                         // 나머지 자리는 HIGH로 설정
            digitalWrite(digit_select_pin[i], HIGH);
    }

    for(int i = 0; i < 8; i++){
        // 공통 음극 방식이므로 반전하지 않고 출력한다.
        // 1자리 7세그먼트 모듈과 반대이다.
        boolean on_off = bitRead(patterns[number], i);
        digitalWrite(segment_pin[i], on_off);
    }
}
```

```
// 4자리 7세그먼트 표시장치에 4자리 숫자를 표시하는 함수
void show_4_digit(int number){
    number = number % 10000;           // 4자리로 제한
    int thousands = number / 1000;     // 천 자리
    number = number % 1000;
    int hundreads = number / 100;      // 백 자리
    number = number % 100;
    int tens = number / 10;            // 십 자리
    int ones = number % 10;            // 일 자리

    show_digit(1, thousands);
    delay(SEGMENT_DELAY);
    show_digit(2, hundreads);
    delay(SEGMENT_DELAY);
    show_digit(3, tens);
    delay(SEGMENT_DELAY);
    show_digit(4, ones);
    delay(SEGMENT_DELAY);
}

void loop(){
    show_4_digit(1357);
}
```

5. 시간 표현하기

4자리 7세그먼트 표시장치에 시간을 표현해 보자. 시간은 분과 초로 나누어 두 자리씩 표시한다. 아두이노에서 별도의 외부 장치 없이 현재 시간을 알 수 있는 방법은 없으므로 시작 시에 12분34초로 초기화하고 시작하는 것으로 한다. 먼저 분과 초를 저장하기 위해 두 개의 변수를 전역적으로 정의한다.

```
int minutes = 12;           // 분 초기화
int seconds = 34;          // 초 초기화
```

loop 함수에서는 1초 간격으로 시간을 업데이트하여 출력 함수를 호출하도록 다음과 같이 수정하여 스케치를 실행시켜 보자. 예상한 바와는 달리 동작할 것이니 놀라지 말기 바란다.

```
void loop(){
    seconds++;                // 초 증가
    if(seconds == 60){        // 60초가 되면 분 증가
        seconds = 0;
        minutes++;           // 분 증가
        if(minutes == 60){    // 60분이 되면 0으로
            minutes = 0;
        }
    }

    show_4_digit(minutes * 100 + seconds);    // 4자리 정수로 변환

    delay(1000);
}
```

이전에도 delay 함수를 사용하면 스케치 실행 중에 입력을 받을 수 없는 등의 문제가 발생한다고 언급하였다. 4자리 7세그먼트 표시장치의 경우 숫자 4개를 동시에 보여주는 것이 아니라 특정 순간에 하나의 숫자만을 보여주므로 1초에 한 번 출력하는 것으로는 마지막에 출력한 자리의 숫자만을 볼 수 있을 뿐이다. 따라서 4자리 7세그먼트 장치를 사용하는 경우에는 delay 함수를 사용해서는 안된다. 스케치 3은 millis 함수를 사용하여 초 단위로 나타나는 시계를 구현한 예이다.

스케치 3. 4자리 7세그먼트에 시간 표시하기

```
// 0에서 9까지 숫자 표현을 위한
```

```
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int digit_select_pin[] = {12, 13, A0, A1};           // 자릿수 선택 핀
int segment_pin[] = {2, 3, 5, 6, 7, 8, 10, 11};      // 7세그먼트 모듈 연결 핀

int SEGMENT_DELAY = 5;                             // 숫자 표시 사이의 시간 간격
int minutes = 12;                                    // 분 초기화
int seconds = 34;                                    // 초 초기화

unsigned long time_previous, time_current;

void setup(){
    for(int i = 0; i < 4; i++){                      // 자릿수 선택 핀을 출력으로 설정
        pinMode(digit_select_pin[i], OUTPUT);
    }
    for(int i = 0; i < 8; i++){                      // 세그먼트 제어 핀을 출력으로 설정
        pinMode(segment_pin[i], OUTPUT);
    }

    time_previous = millis();
}

// 해당 자리에 숫자 하나를 표시하는 함수
void show_digit(int pos, int number){               // (위치, 출력할 숫자)
    for(int i = 0; i < 4; i++){
        if(i + 1 == pos)                            // 해당 자릿수의 선택 핀만 LOW로 설정
            digitalWrite(digit_select_pin[i], LOW);
        else                                          // 나머지 자리는 HIGH로 설정
            digitalWrite(digit_select_pin[i], HIGH);
    }

    for(int i = 0; i < 8; i++){
        // 공통 음극 방식이므로 반전하지 않고 출력한다.
```

```
// 1자리 7세그먼트 모듈과 반대이다.
boolean on_off = bitRead(patterns[number], i);
digitalWrite(segment_pin[i], on_off);
}
}

// 4자리 7세그먼트 표시장치에 4자리 숫자를 표시하는 함수
void show_4_digit(int number){
    number = number % 10000;           // 4자리로 제한
    int thousands = number / 1000;     // 천 자리
    number = number % 1000;
    int hundreads = number / 100;      // 백 자리
    number = number % 100;
    int tens = number / 10;            // 십 자리
    int ones = number % 10;           // 일 자리

    show_digit(1, thousands);
    delay(SEGMENT_DELAY);
    show_digit(2, hundreads);
    delay(SEGMENT_DELAY);
    show_digit(3, tens);
    delay(SEGMENT_DELAY);
    show_digit(4, ones);
    delay(SEGMENT_DELAY);
}

void loop(){
    time_current = millis();

    if(time_current - time_previous >= 1000){
        time_previous = time_current;

        seconds++;                     // 초 증가
        if(seconds == 60){             // 60초가 되면 분 증가
            seconds = 0;
            minutes++;                 // 분 증가
        }
    }
}
```

```

        if(minutes == 60){           // 60분이 되면 0으로
            minutes = 0;
        }
    }

    show_4_digit(minutes * 100 + seconds);    // 4자리 정수로 변환
}

```

6. 가변저항 값 표시하기

가변저항의 값을 읽어 4자리 7세그먼트 표시장치에 표시 해 보자. 가변저항은 그림 6과 같이 아날로그 핀 3번 A3에 연결한다.

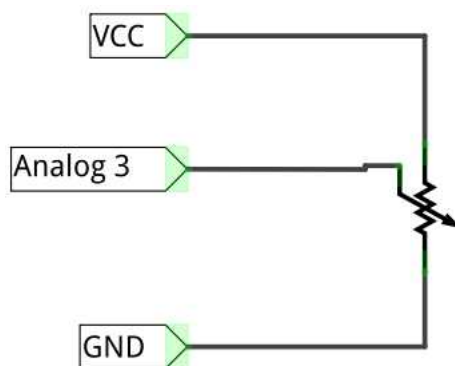


그림 6. 가변 저항 연결 회로도

아날로그 입력은 10비트 아날로그-디지털 변환기를 거쳐 0에서 1023 사이의 값이 입력되므로 4자리 7세그먼트 표시장치에 값을 표시할 수 있다. 다만 스케치 2와 스케치 3의 show_4_digit 함수는 항상 4자리 숫자만을 표시한다. 즉, '123'을 표시할 수 없으며 '0123'과 같이 4자리를 표현하도록 되어 있다. 4자리 미만의 숫자인 경우 앞부분을 공백으로 남기도록 show_4_digit 함수를 수정해 보자. 공백을 표현하기 위해서는 현재 숫자의 자릿수를 알아야 한다. 다음 코드는 숫자의 자릿수를 세는 예이다.

```
int count_digit(int number)
{
    int n = 0;

    while(number > 0){
        number /= 10;
        n++;
    }

    // number == 0인 경우에도 0을 표시하기 위해 최소 1자리로 지정
    n = max(n, 1);

    return n;
}
```

count_digit 함수를 이용하여 첫머리 영을 표시하지 않도록 새로운 함수를 작성해 보자.

```
void show_4_digit_no_leading_zero(int number){
    number = number % 10000;           // 4자리로 제한
    int n = count_digit(number);       // 자릿수 세기
    int digits[4] = {0, };             // 각 자릿값이 저장되는 배열

    for(int i = 0; i < 4; i++){
        digits[i] = number % 10;       // 각 자릿값 계산
        number = number / 10;
    }

    for(int i = 0; i < n; i++){         // 자릿수만큼만 표시
        show_digit(4 - i, digits[i]);
        delay(SEGMENT_DELAY);
    }
}
```

스케치 4는 가변저항의 값을 읽어서 4자리 7세그먼트 표시장치에 표시하는 스케치의 예이다.

스케치 4. 4자리 7세그먼트에 가변저항 값 표시하기

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int digit_select_pin[] = {12, 13, A0, A1};           // 자릿수 선택 핀
int segment_pin[] = {2, 3, 5, 6, 7, 8, 10, 11};       // 7세그먼트 모듈 연결 핀

int SEGMENT_DELAY = 5;                             // 숫자 표시 사이의 시간 간격
int R_pin = A3;

void setup(){
    for(int i = 0; i < 4; i++){                       // 자릿수 선택 핀을 출력으로 설정
        pinMode(digit_select_pin[i], OUTPUT);
    }
    for(int i = 0; i < 8; i++){                       // 세그먼트 제어 핀을 출력으로 설정
        pinMode(segment_pin[i], OUTPUT);
    }
}

// 해당 자리에 숫자 하나를 표시하는 함수
void show_digit(int pos, int number){                // (위치, 출력할 숫자)
    for(int i = 0; i < 4; i++){
        if(i + 1 == pos)                             // 해당 자릿수의 선택 핀만 LOW로 설정
            digitalWrite(digit_select_pin[i], LOW);
        else                                           // 나머지 자리는 HIGH로 설정
            digitalWrite(digit_select_pin[i], HIGH);
    }

    for(int i = 0; i < 8; i++){
        // 공통 음극 방식이므로 반전하지 않고 출력한다.
```

```
// 1자리 7세그먼트 모듈과 반대이다.
boolean on_off = bitRead(patterns[number], i);
digitalWrite(segment_pin[i], on_off);
}
}

int count_digit(int number)
{
    int n = 0;

    while(number > 0){
        number /= 10;
        n++;
    }

    // number == 0인 경우에도 0을 표시하기 위해 최소 1자리로 지정
    n = max(n, 1);

    return n;
}

// 4자리 7세그먼트 표시장치에 4자리 숫자를 표시하는 함수
void show_4_digit_no_leading_zero(int number){
    number = number % 10000;           // 4자리로 제한
    int n = count_digit(number);       // 자릿수 세기
    int digits[4] = {0, };             // 각 자릿값이 저장되는 배열

    for(int i = 0; i < 4; i++){
        digits[i] = number % 10;        // 각 자릿값 계산
        number = number / 10;
    }

    for(int i = 0; i < n; i++){          // 자릿수만큼만 표시
        show_digit(4 - i, digits[i]);
        delay(SEGMENT_DELAY);
    }
}
```

```
}

void loop(){
    show_4_digit_no_leading_zero(analogRead(R_pin));
}
```