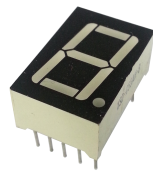


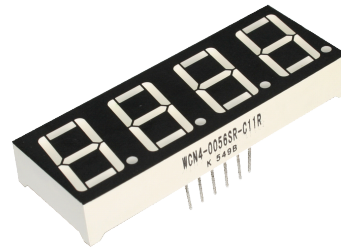
Chapter 11. 4자리 7세그먼트 표시장치

1. 7세그먼트 표시장치

7세그먼트 표시장치(7-segment display)는 7개의 선분으로 숫자나 글자를 표시하기 위해 발광 다이오드(LED)를 사용하여 만든 출력 장치의 일종으로 7개 선분에 소수점(decimal point) 표시를 위한 LED가 추가된 8개의 세그먼트들을 1바이트의 데이터로 제어할 수 있도록 만들어져 있다.



(a) 한 자리



(b) 네 자리

그림 1. 7세그먼트 표시장치

먼저 1자리 7세그먼트 표시장치의 구조를 살펴보자. 1자리 7세그먼트 표시장치는 세그먼트를 제어하는 방식에 따라 공통 양극(common anode) 방식과 공통 음극(common cathode) 방식의 두 가지가 있다. 두 종류의 7세그먼트 장치에서 핀의 개수와 배치는 동일하다.

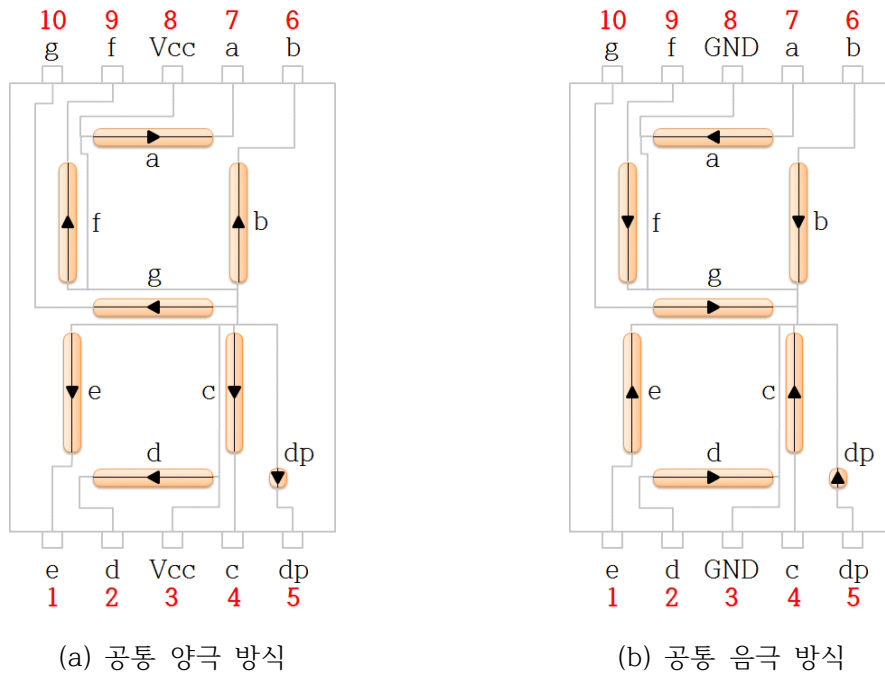


그림 2. 7세그먼트 표시장치의 핀 배열

7세그먼트 표시장치에서 10개의 핀 중 2개는 VCC나 GND를 연결하는 공통 핀이며 나머지 8개 핀은 해당 세그먼트를 제어하는 제어 핀이다. 공통 핀에 VCC를 연결하고 각 제어 핀을 GND에 연결하면 불이 들어오는 방식이 공통 양극 방식이며, 공통 핀을 GND에 연결하고 각 제어 핀을 VCC에 연결하면 불이 들어오는 방식이 공통 음극 방식이다. 7세그먼트 표시장치에 숫자를 표시하기 위해서는 8개 세그먼트 각각에 LOW나 HIGH 값을 출력하여 각각의 세그먼트를 개별적으로 제어하여야 한다. 그림 3은 7세그먼트 표시장치에 숫자를 표시하는 예이다.

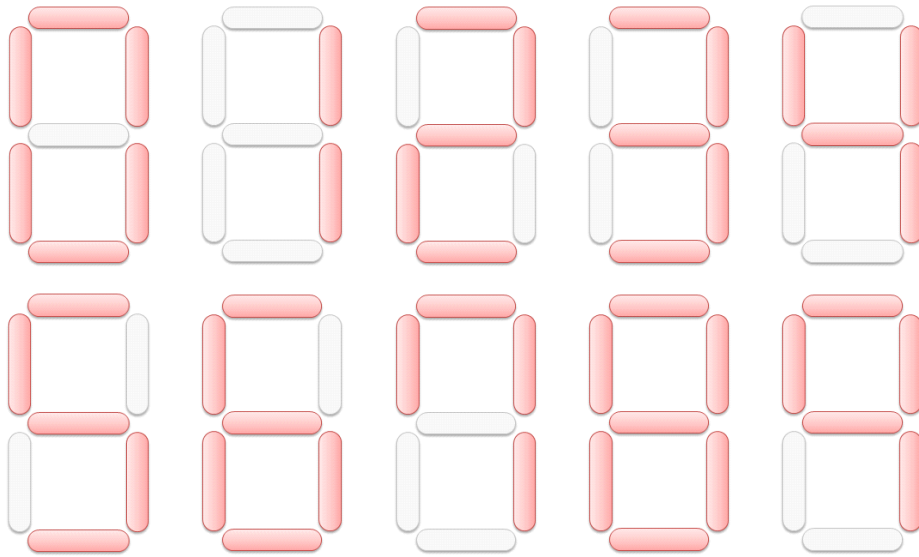


그림 3. 7세그먼트 표시장치의 숫자 표현

그림 2에 표시된 것처럼 8개의 세그먼트는 a, b, c, d, e, f, g, dp로 표시되며, 그림 3과 같이 숫자를 표시하기 위한 1바이트의 데이터는 다음과 같다.

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, // 0b11111100, '0'
    0x60, // 0b01100000, '1'
    0xDA, // 0b11011010, '2'
    0xF2, // 0b11110010, '3'
    0x66, // 0b01100110, '4'
    0xB6, // 0b10110110, '5'
    0xBE, // 0b10111110, '6'
    0xE4, // 0b11100100, '7'
    0xFE, // 0b11111110, '8'
    0xE6, // 0b11100110, '9'
};
```

2. 4자리 7세그먼트 표시장치

1자리 7세그먼트 표시장치를 제어하기 위해서는 각 세그먼트를 위한 8개의 제어선이 필요하다. 4자리 7세그먼트 표시장치를 1자리 7세그먼트 표시장치 4개와 동일하게 생각한다면 32개의 제어선이 필요할 것으로 생각할 수 있다. 물론 이러한 방식으로 제어는 가능하지만 아두이노 우노의 경우 최대 20개의 디지털 입출력 핀만을 제공하므로 이러한 방식으로는 4자리 7세그먼트 표시장치를 사용하는 것은 불가능하다. 아두이노 메가2560의 경우에는 70개의 입출력 핀을 사용할 수 있지만 절반에 가까운 핀이 필요하므로 역시 실용적이지 못하다. 따라서 4자리 7세그먼트 표시장치를 제어하기 위해서는 다른 방법이 사용된다.

사람의 눈은 반응 속도가 그리 빠르지 않으므로 눈앞에서 물건이 사라진 이후에도 잠깐 동안은 물건이 사라진 것을 인식하지 못하며 이를 잔상효과라고 이야기 한다. 잔상효과를 이용하여 한 번에 한 자리의 숫자만을 표시하고 빠른 속도로 자리를 바꾸어 표시하기를 반복한다면 4자리 숫자가 동시에 표시되는 것과 동일한 효과를 얻을 수 있다. 잔상효과를 이용하는 4자리 7세그먼트 표시장치는 32개가 아닌 12개의 핀만을 가지고 있다.

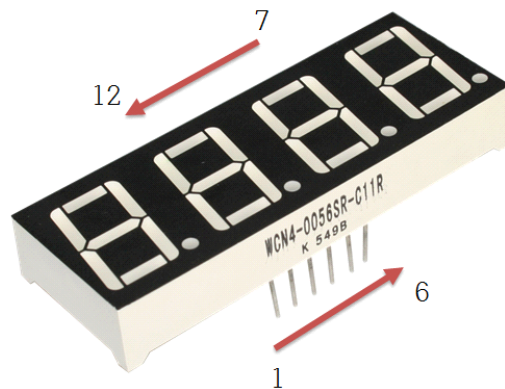


그림 4. 4자리 7 세그먼트 표시장치

4자리 7세그먼트 표시장치의 경우 12개의 핀 중 8개의 핀이 각 세그먼트 제어를 위해 사용되는 점은 1자리 7세그먼트 표시장치와 동일하며, 나머지 4개 핀은 각 자릿수 선택을 위해 사용된다. 즉, 1자리 7세그먼트 표시장치의 경우 항상 한 자리가 선택되어 있는 상태라면 4자리 7세그먼트 표시장치의 경우 자릿수 선택 핀을 통해 특정 자리를 선택할 수 있는 차이가 있다. 4자리 7세그먼트 표시장치는 그림 5와 같이 구성된다.

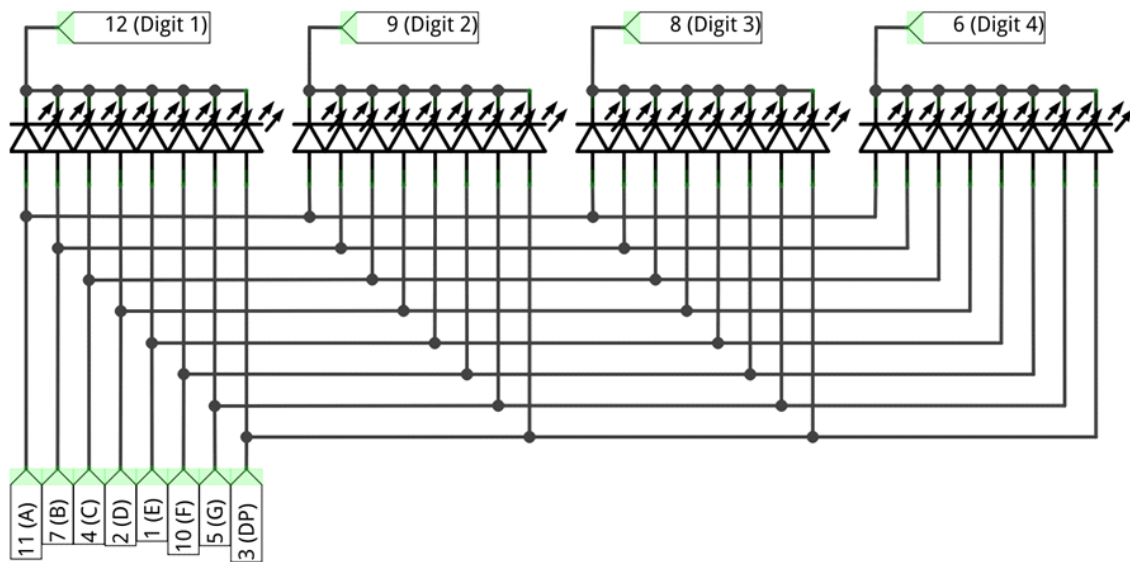


그림 5. 4자리 7세그먼트 표시장치 회로도

그림 5에서 알 수 있듯이 12번, 9번, 8번, 6번 핀은 네 자리 중 원하는 자릿수를 선택하기 위해 사용되고, 8개 세그먼트를 제어하는 핀은 공통으로 사용된다. 학습보드의 7 세그먼트 표시 장치는 공통 음극 방식이므로 자릿수 선택 핀에 LOW를 가하여 특정 자리를 선택한다. 이 때 다른 자릿수 선택 핀에는 HIGH를 가해주어야 한다는 점도 잊지 말아야 한다. 학습보드에는 4 자리 7세그먼트 표시장치가 아날로그 4번 A4(디지털 58번)에서 아날로그 15번 A15(디지털 69번)까지 12개 핀을 사용하여 연결되어 있다.

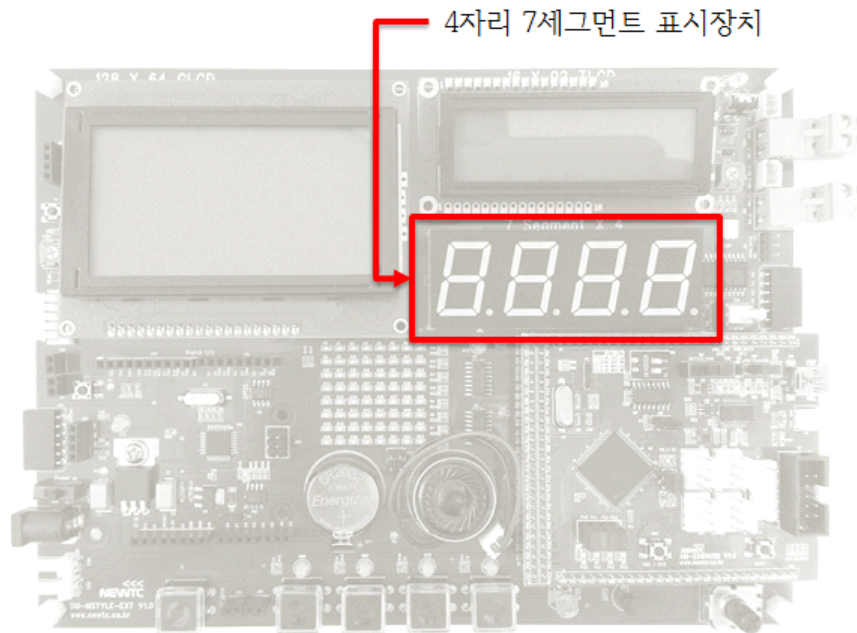


그림 6. 학습보드의 4자리 7세그먼트 표시 장치

스케치 1은 왼쪽부터 오른쪽으로 자리를 옮겨 가면서 숫자를 나타내는 스케치의 예이다. 나타내는 숫자는 0에서 시작하여 4자리에 각각 숫자를 표시한 후 1씩 증가하며, 표시하는 숫자는 0에서 9 사이를 반복한다. 스케치 1에서 show_digit 함수는 지정한 자리에 지정한 숫자를 표시하는 함수로 자릿수는 왼쪽부터 1~4의 값을, 숫자는 0~9의 값을 지정한다.

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[] = {66, 67, 68, 69};           // 자릿수 선택 핀

// 7세그먼트 모듈 연결 핀 'a, b, c, d, e, f, g, dp' 순서
int segment_pin[] = {58, 59, 60, 61, 62, 63, 64, 65};

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++){
        boolean on_off = bitRead(patterns[number], 7 - i);
        digitalWrite(segment_pin[i], on_off);
    }
}

void loop(){
    for(int no = 0; no < 10; no++){
        for(int pos = 1; pos <= 4; pos++){
            show_digit(pos, no);      // 'pos' 자리에 숫자 'no'를 표시
            delay(200);
        }
    }
}

```

스케치 1은 4자리 중 한 자리에만 숫자를 표현할 수 있다. 여러 자리에 동시에 숫자를 표시하기 위해서는 잔상효과를 이용하여야 한다. 스케치 2는 4자리 숫자를 한 자리씩 빠른 속도로 출력하기를 반복함으로써 여러 자리의 숫자가 동시에 표시되는 것처럼 보이는 스케치의 예로 0.1초에 한 번씩 카운터 값이 증가한다.

스케치 2. 4자리 7세그먼트에 카운터 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[] = {66, 67, 68, 69};          // 자릿수 선택 핀

// 7세그먼트 모듈 연결 핀 'a, b, c, d, e, f, g, dp' 순서
int segment_pin[] = {58, 59, 60, 61, 62, 63, 64, 65};

int count = 0;
int SEGMENT_DELAY = 5;                            // 숫자 표시 사이의 시간 간격
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++){
    boolean on_off = bitRead(patterns[number], 7 - 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 hundreds = 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, hundreds);            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 >= 100){
        time_previous = time_current;

        count++;
    }
    show_4_digit(count);
}
```