

Chapter 8. 아날로그 데이터 입출력

아두이노 메가2560에 사용된 ATmega2560 마이크로컨트롤러에는 아날로그 데이터 입력을 위해 16채널의 아날로그-디지털 변환기(Analog-to-Digital Converter, ADC)가 포함되어 있으며 각 채널은 A0에서 A15까지의 아날로그 핀으로 정의되어 있다. 아날로그 핀을 나타내는 상수는 A0가 디지털 54번 핀에 해당하며 A15가 디지털 69번 핀에 해당한다. 따라서 아날로그 0번 핀을 지정하기 위해서는 디지털 핀 번호인 '54'나 이에 해당하는 상수 'A0' 중 어느 것을 사용해도 무방하다.

ATmega2560 마이크로컨트롤러에 ADC가 포함되어 있지만 디지털-아날로그 변환기(DAC)는 포함되어 있지 않으므로 아날로그 데이터를 출력할 수는 없다. 하지만 아날로그 데이터와 유사한 효과를 내는 펄스 폭 변조(PWM) 디지털 신호 출력을 지원한다. PWM 신호 출력을 위한 함수의 이름도 analogWrite로 'analog'라는 단어가 포함되어 있지만 실제로 출력되는 신호는 PWM 형식의 디지털 신호라는 점에 유의하여야 한다.

1. 아날로그 데이터 입력

아두이노 메가2560에는 아날로그 데이터 입력을 위해 16개의 아날로그 입력 핀이 준비되어 있다. 아두이노 메가2560에 사용되는 ATmega2560 마이크로컨트롤러의 ADC는 10비트의 해상도를 가진다. 즉, 입력되는 아날로그 값을 10비트의 디지털 값으로 변환하여 반환하므로 0에서 1023 사이의 값을 반환한다. 아날로그 데이터 입력을 위해 사용하는 함수는 analogRead 함수이다.

```
int analogRead(uint8_t pin)
- 매개변수
  pin : 핀 번호
- 반환값 : 0에서 1023 사이의 정수값
```

아날로그 데이터를 읽을 때 고려하여야 할 점 중 한 가지는 기준 전압의 설정이다. 기준 전압은 analogRead 함수가 반환할 수 있는 최대값인 1023에 해당하는 전압을 말한다. 기본적으로 기준 전압은 아두이노 메가2560 보드의 AVCC(Analog VCC) 값인 5V를 사용한다. 하지만

아두이노 메가2560과 사용할 수 있는 주변장치들이 출력하는 전압의 범위가 0~5V가 아닌 경우도 있으므로 이런 경우에는 analogReference 함수를 통해 기준 전압 변경이 가능하다.

```
void analogReference(uint8_t type)
```

- 매개변수

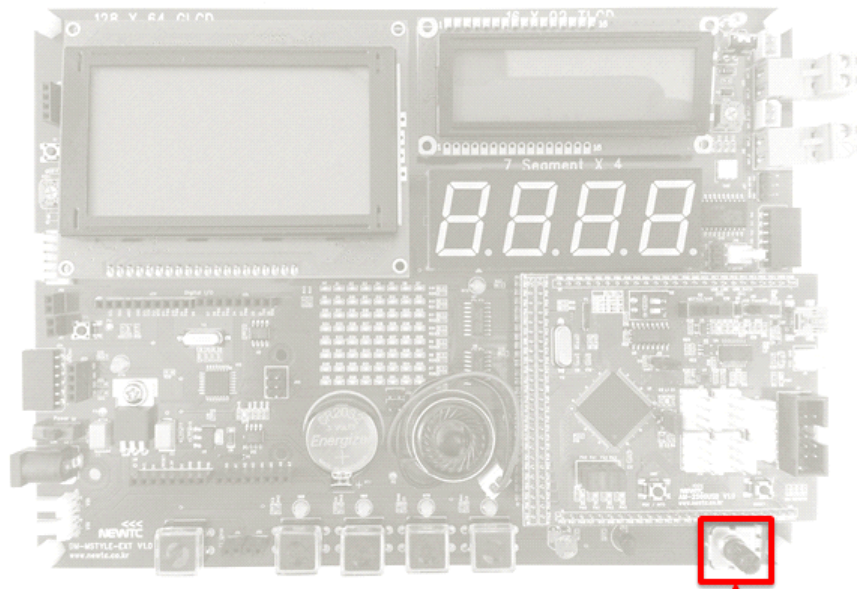
type : DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56, EXTERNAL 중 한 가지

- 반환값 : 없음

analogReference 함수로 설정할 수 있는 기준 전압 옵션에는 다음과 같은 것들이 있다.

- ◆ DEFAULT : 아두이노 보드의 동작 전압인 5V로 설정한다. 3.3V를 사용하는 아두이노 보드의 경우 3.3V로 설정한다.
- ◆ INTERNAL : 내부 기준 전압으로 설정한다. ATmega328의 경우 1.1V로 설정된다. 아두이노 메가2560의 경우에는 사용할 수 없다.
- ◆ INTERNAL1V1 : 내부 1.1V를 기준 전압으로 설정한다. 아두이노 메가2560에서만 사용할 수 있다.
- ◆ INTERNAL2V56 : 내부 2.56V를 기준 전압으로 설정한다. 아두이노 메가2560에서만 사용할 수 있다.
- ◆ EXTERNAL : AREF 핀에 인가된 0V ~ 5V 사이의 전압을 기준 전압으로 설정한다.

아날로그 0번 A0 핀에 가변저항을 연결하여 A0 핀에 가해지는 전압을 디지털로 변환하여 읽어보자. 기준전압은 디폴트인 5V를 사용한다.



A0 (디지털 54번) 핀 가변저항

그림 1. 학습보드에 포함된 가변저항

가변저항을 돌리면 가변저항에는 VCC(5V)와 GND(0V) 사이의 전압이 A0 핀에 가해지고, 가해진 전압은 0에서 1023 사이의 디지털 값으로 변환되어 입력된다. 스케치 1은 가변저항의 값을 읽어 시리얼 모니터로 출력하는 스케치의 예이다.

스케치 1. 가변저항 테스트

```
int vResistor = A0;                                // A0 핀에 가변저항 연결

void setup() {
  Serial.begin(9600);                               // 시리얼 통신 초기화
  pinMode(vResistor, INPUT);
}

void loop() {
  Serial.println(analogRead(vResistor));             // 가변저항 값을 읽어 출력

  delay(1000);                                       // 1초 (1000ms) 대기
}
```

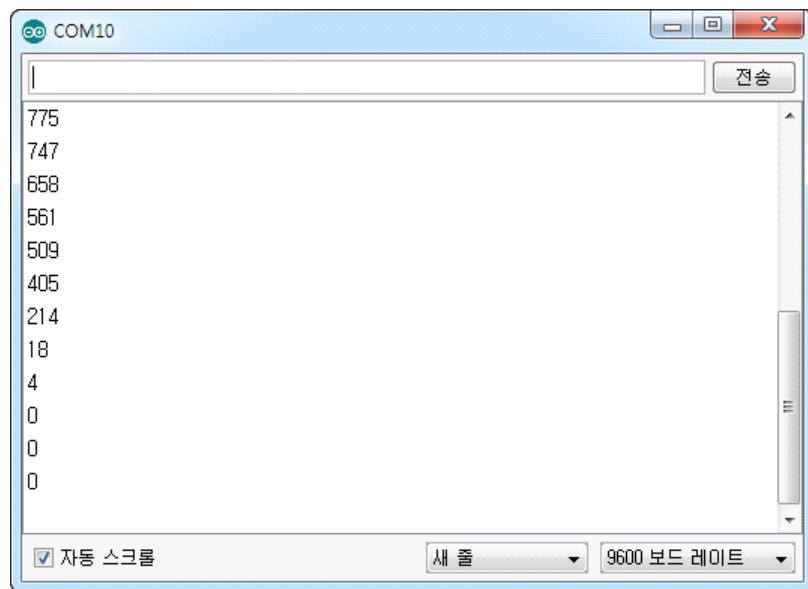


그림 2. 스케치 1 실행 결과

가변저항의 값은 `analogRead` 함수를 통해 0~1023 사이의 디지털 값으로 입력된다. 이 값을 4단계로 나누어서 디지털 2, 3, 4, 5번 핀에 연결된 LED의 개수로 나타내어 보자. ADC에서 반환되는 값은 10비트의 디지털 값으로 표현되므로 이 값을 오른쪽으로 8비트만큼 이동시키면 0~3 사이의 값을 얻을 수 있고 이 값으로 켜야 하는 LED 개수를 결정할 수 있다.

스케치 2. 가변저항으로 LED 개수 조절하기

```
int vResistor = A0;           // A0 핀에 가변저항 연결
int pins_LED[] = {2, 3, 4, 5}; // LED 연결 핀

void setup() {
    Serial.begin(9600);        // 시리얼 통신 초기화
    pinMode(vResistor, INPUT);
    for(int i = 0; i < 4; i++){
        pinMode(pins_LED[i], OUTPUT);
        digitalWrite(pins_LED[i], LOW);
    }
}

void loop() {
```

```
int adc = analogRead(vResistor);    // 가변저항 값 읽기
int count_led = (adc >> 8) + 1;    // LED 개수 결정

for(int i = 0; i < 4; i++){        // LED 점멸
    if(i < count_led)
        digitalWrite(pins_LED[i], HIGH);
    else
        digitalWrite(pins_LED[i], LOW);
}

// ADC 값과 LED 개수 출력
Serial.println(String("ADC : ") + adc + ", LED count : " + count_led);

delay(1000);                      // 1초 (1000ms) 대기
}
```

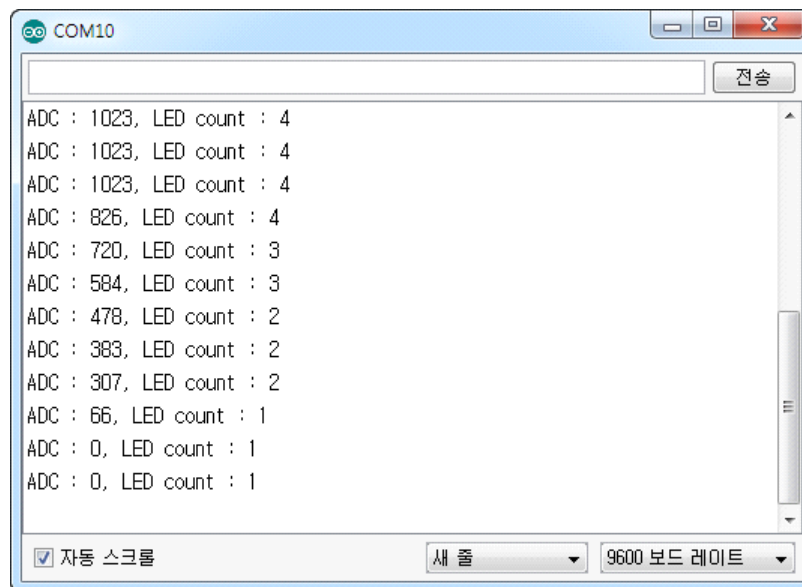


그림 3. 스케치 2 실행 결과

2. 아날로그 데이터 출력

아두이노 메가2560에 사용된 ATmega2560은 ADC를 포함하고 있어 아날로그 데이터 입력은 가능하지만 DAC를 포함하고 있지는 않으므로 아날로그 데이터 출력은 불가능하다. 대신 펄스

폭 변조 (Pulse Width Modulation, PWM) 신호를 이용하여 아날로그 신호와 유사한 효과를 얻을 수 있다. ATmega2560에는 15개의 핀으로 PWM 신호를 출력할 수 있다. PWM 신호는 구형파(square wave)를 기본으로 하고 한 주기 내에서 ON 상태인 시간의 비율로 표시된다. 한 주기 내에서 ON 상태인 시간의 비율을 듀티 사이클(duty cycle)이라 하며, 그림 4는 듀티 사이클에 따라 생성되는 구형파를 보여주고 있다.

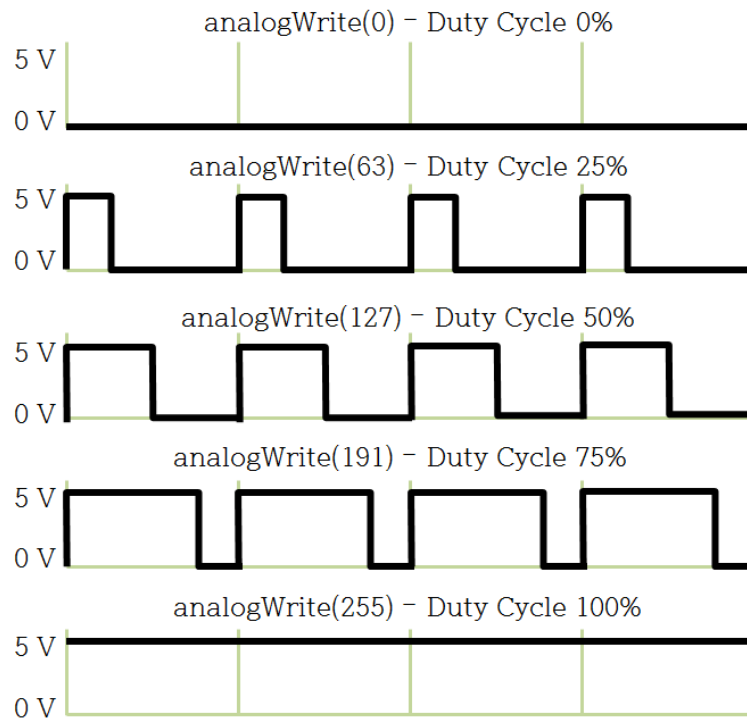


그림 4. 펄스 폭 변조 방식 (Pulse Width Modulation)

PWM 신호를 출력하기 위해서는 디지털 핀에 `analogWrite` 함수를 이용하여 0에서 255 사이의 값을 출력함으로써 서로 다른 듀티 사이클을 가지는 구형파를 출력할 수 있다. 255를 출력하면 듀티 사이클은 100%가 되고 0을 출력하면 듀티 사이클은 0%가 된다.

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

- 매개변수

- pin : 핀 번호

- value : 듀티 사이클 (duty cycle). 0(항상 OFF)에서 255(항상 ON) 사이의 값

- 반환값 : 없음

학습보드의 디지털 6, 7, 8번 핀에는 RGB LED가 연결되어 있고 디지털 2, 3, 4, 5번 핀에는 LED가 연결되어 있다. 이들 핀 모두는 PWM 신호 출력이 가능하다.

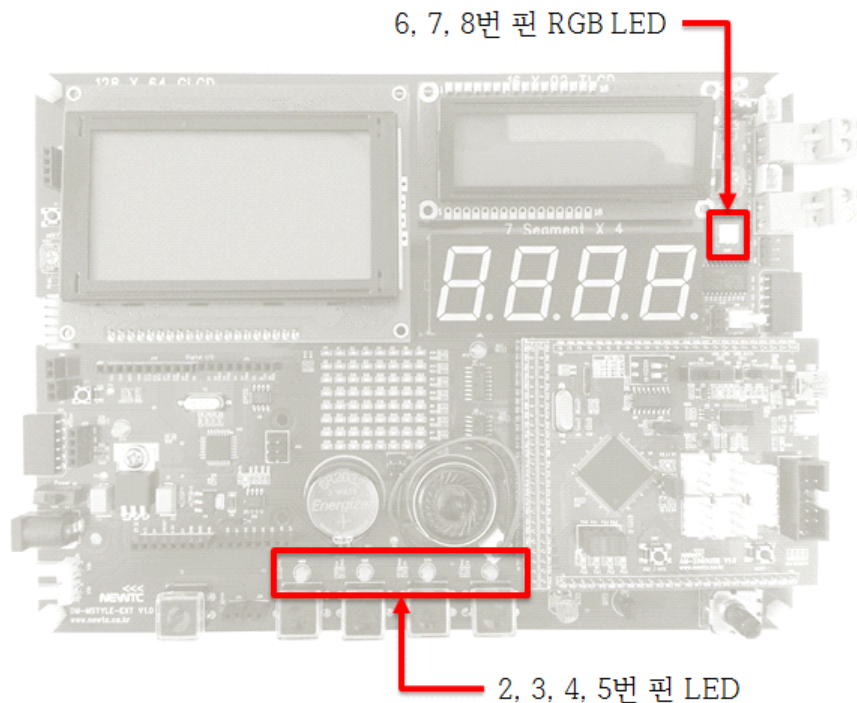


그림 5. 학습보드에 연결된 LED와 RGB LED

RGB LED는 빛의 3원색에 해당하는 Red, Green, Blue LED를 하나의 패키지에 포함시켜 놓은 형태로 각 원색을 제어하기 위한 개별적인 제어 핀 3개와 공통 핀 1개를 가지고 있다. 공통 핀을 GND에 연결하고 각 제어 핀에 VCC를 연결하면 LED가 켜지는 방식을 공통 음극 방식이라 하며, 공통 핀을 VCC에 연결하고 각 제어 핀에 GND를 연결하면 LED가 켜지는 방식을 공통 양극 방식이라 한다. 학습보드에 포함된 RGB LED는 공통 양극 방식이다.

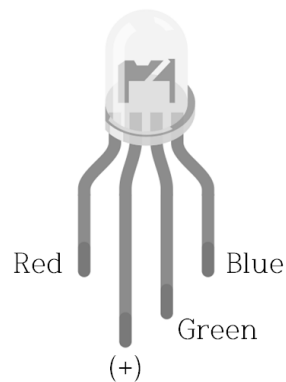


그림 6. RGB LED

스케치 3은 RGB LED의 RGB 중 하나만을 서서히 변화시키는 스케치의 예이다.

스케치 3. RGB LED의 밝기 조절

```
int RGB_LED[] = {6, 7, 8};           // RGB LED 연결 핀

void setup(){
  for(int i = 0; i < 3; i++){
    pinMode(RGB_LED[i], OUTPUT);
  }
}

void loop(){
  // Blue 색상 조절. Green, Red는 끄
  digitalWrite(RGB_LED[1], HIGH);
  digitalWrite(RGB_LED[2], HIGH);
  for(int i = 255; i >= 0; i--){
    analogWrite(RGB_LED[0], i);
    delay(10);
  }

  // Green 색상 조절
  digitalWrite(RGB_LED[0], HIGH);
  digitalWrite(RGB_LED[2], HIGH);
  for(int i = 255; i >= 0; i--){
```

```
    analogWrite(RGB_LED[1], i);
    delay(10);
}

// Red 색상 조절
digitalWrite(RGB_LED[0], HIGH);
digitalWrite(RGB_LED[1], HIGH);
for(int i = 255; i >= 0; i--){
    analogWrite(RGB_LED[2], i);
    delay(10);
}
}
```