

사물인터넷(IoT) 프로그래밍 기초

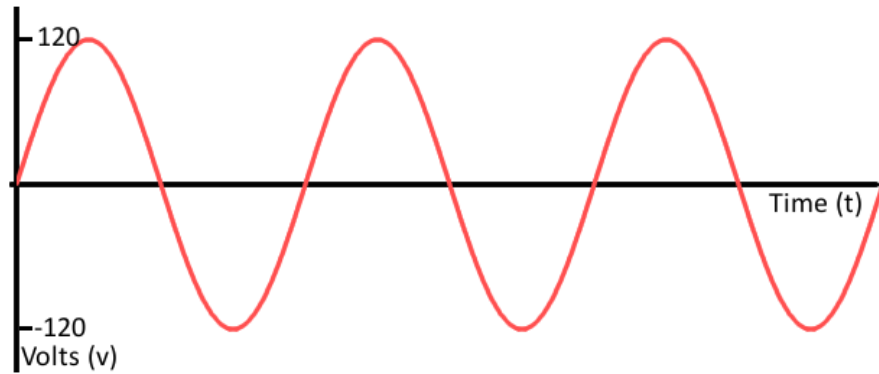
4. 아날로그 입력/아날로그 출력(PWM) /가변저항/조도센서

목차

1. 아날로그(Analog) vs 디지털(digital)
2. 아날로그 입력
 - 가변저항(Potentiometer)의 이해
 - 아날로그 입력 실습
 - 실습 1: 전압 측정하기
3. 아날로그 출력(PWM)
 - 조도센서(CDS: photo resistor)의 이해
 - 아날로그 입출력 실습
 - 실습 2: 아날로그 출력으로 LED 밝기 제어하기
 - 실습 3: 조도센서값 읽어내기

1. 아날로그 vs. 디지털

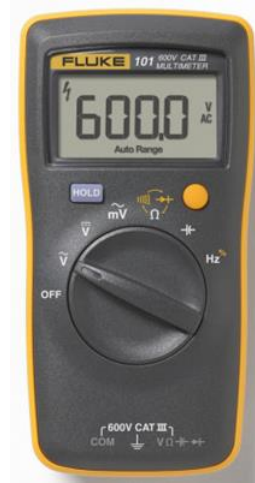
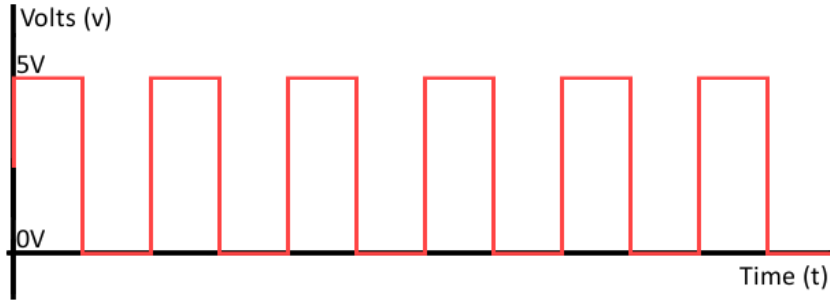
- 아날로그 신호(Analog Signal)



- 시간에 따라 연속적으로 변화하는 크기(전류, 전압)

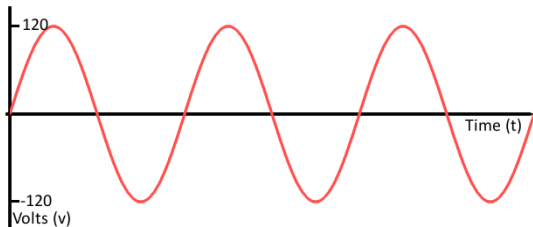
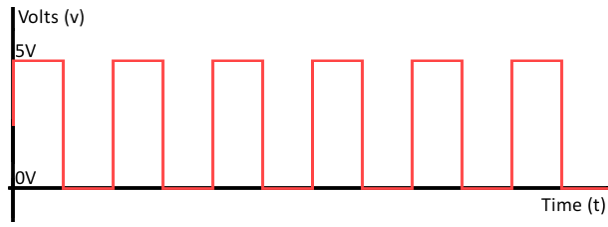
아날로그 vs. 디지털

- 디지털 신호(Digital Signal)



- 0과 1로 이루어진 **이산적** 값으로 데이터 표현

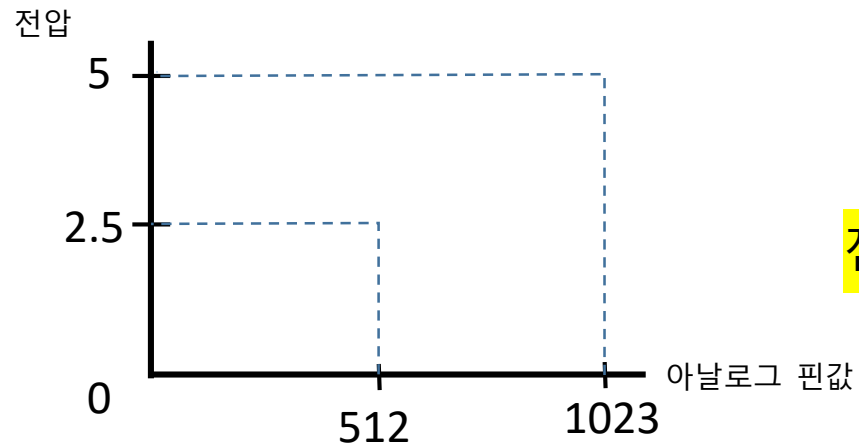
아날로그 vs. 디지털 비교

구분	아날로그 신호	디지털 신호
정의	<ul style="list-style-type: none"> 빛, 소리, 전압 등과 같이 연속적으로 변하는 신호 	<ul style="list-style-type: none"> 특정한 값을 단위로 하여 이산적으로 변하는 신호
예	<ul style="list-style-type: none"> 지진계 바늘의 위치 변화 온도계의 눈금 변화 	<ul style="list-style-type: none"> 컴퓨터, 휴대폰 같은 현대 문명에서 사용되는 대부분의 신호
장점	<ul style="list-style-type: none"> 세밀한 표현 가능 원본과 같은 정보 	<ul style="list-style-type: none"> 정보의 저장과 전달 용이 변형없이 전달 가능
단점	<ul style="list-style-type: none"> 신호 전달 시 변형될 수 있음 잡음(노이즈) 대용량의 정보로 처리하기 어려움 	<ul style="list-style-type: none"> 원본의 정보를 그대로 기록하고 재생할 수 없음
그래프	 <p>The graph shows a continuous sine wave representing an analog signal. The vertical axis is labeled 'Volts (v)' with markings at 120 and -120. The horizontal axis is labeled 'Time (t)'. The wave oscillates smoothly between these two voltage levels.</p>	 <p>The graph shows a square wave representing a digital signal. The vertical axis is labeled 'Volts (v)' with markings at 5V and 0V. The horizontal axis is labeled 'Time (t)'. The signal switches abruptly between 5V and 0V, remaining at each level for a specific duration.</p>

2. 아날로그 입력

• ADC(Analog to Digital Converter)

- 아날로그 신호(**0~5V** 사이의 전압)을 디지털 값으로 변환해 주는 기능
- 10비트(bit) 지원(**0~1023 : $2^{10} = 1024$, 10bit**)
- 아날로그 신호를 **샘플링** → **양자화** → **부호화** 과정을 통해 디지털 신호로 변환



$$\text{전압} = \frac{5}{1023} \times \text{아날로그 핀값}$$

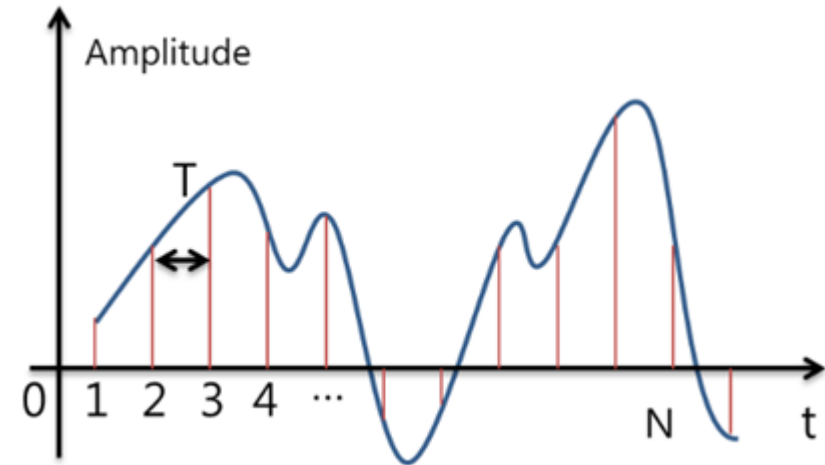
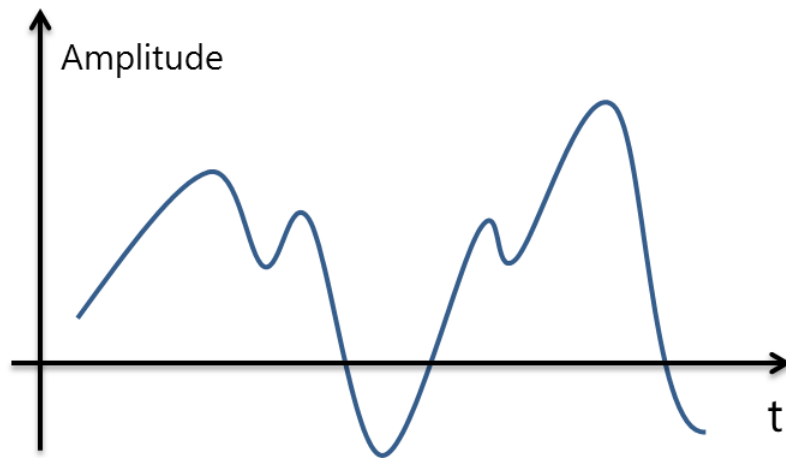
$$\begin{aligned} \text{아날로그 핀값} : \text{전압} &= 1023 : 5 \\ \text{전압} \times 1023 &= 5 \times \text{아날로그 핀값} \end{aligned}$$

$$\text{전압(V)} = \frac{5}{1023} \times \text{아날로그 핀값}$$

샘플링 → 양자화 → 부호화

• 샘플링(Sampling)

- 아날로그 신호를 일정한 시간 간격으로 **신호를 채취**하는 것
- 무한한 데이터 값에서 전체 패턴을 추정하기 위해 유한한 개수의 데이터를 추출하는 과정

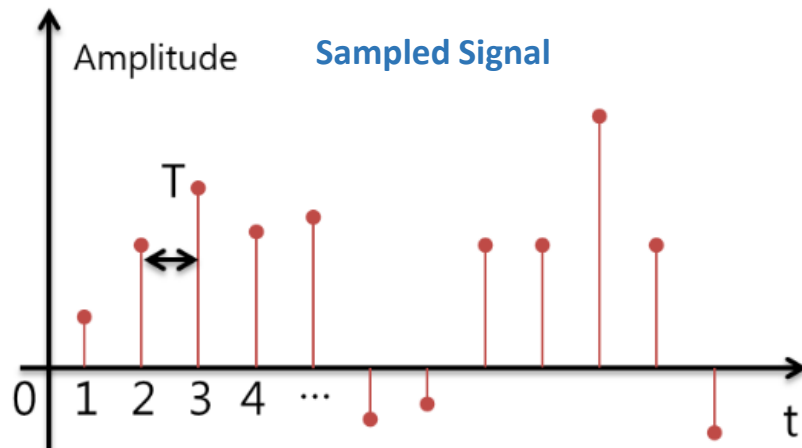


- 샘플링 비율(Sampling rate) : 1초당 추출되는 샘플 개수
 - 샘플 수가 많을 수록 원본과 유사해지나 정보량이 많아짐
 - 그래프에서는 세로선의 개수 의미

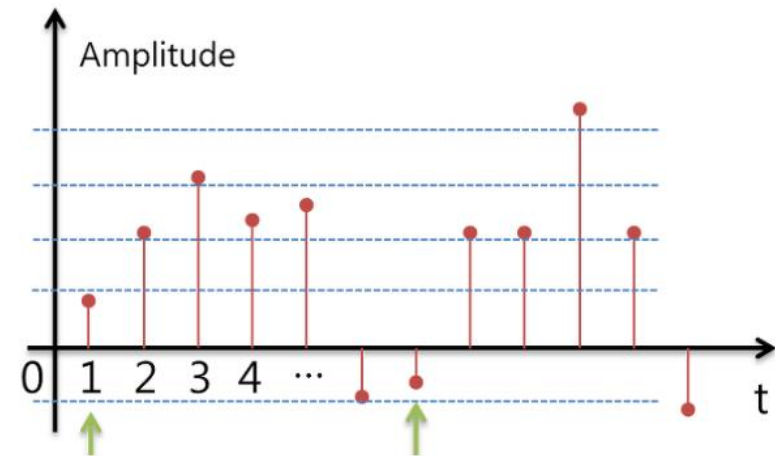
샘플링 → 양자화 → 부호화

• 양자화(Quantization)

- 샘플링 된 신호를 일정한 **전압 레벨의 구간**을 나눈 영역에 강제로 대응 시키는 과정



→
양자화

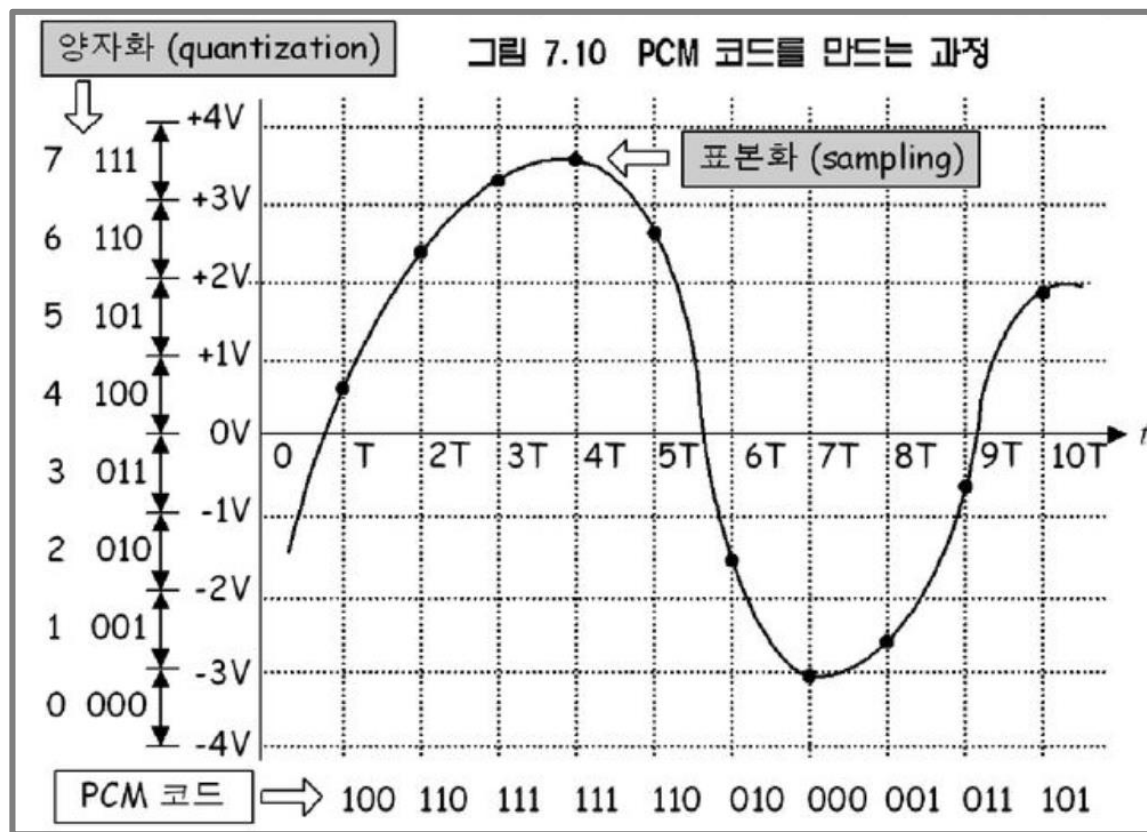


- 양자화 레벨(Quantization level, Resolution) : 대표 값 레벨의 개수
 - 양자화 레벨 수가 많을 수록 더 세밀한 표현 가능
 - 그래프에서는 가로선(전압 레벨) 의미

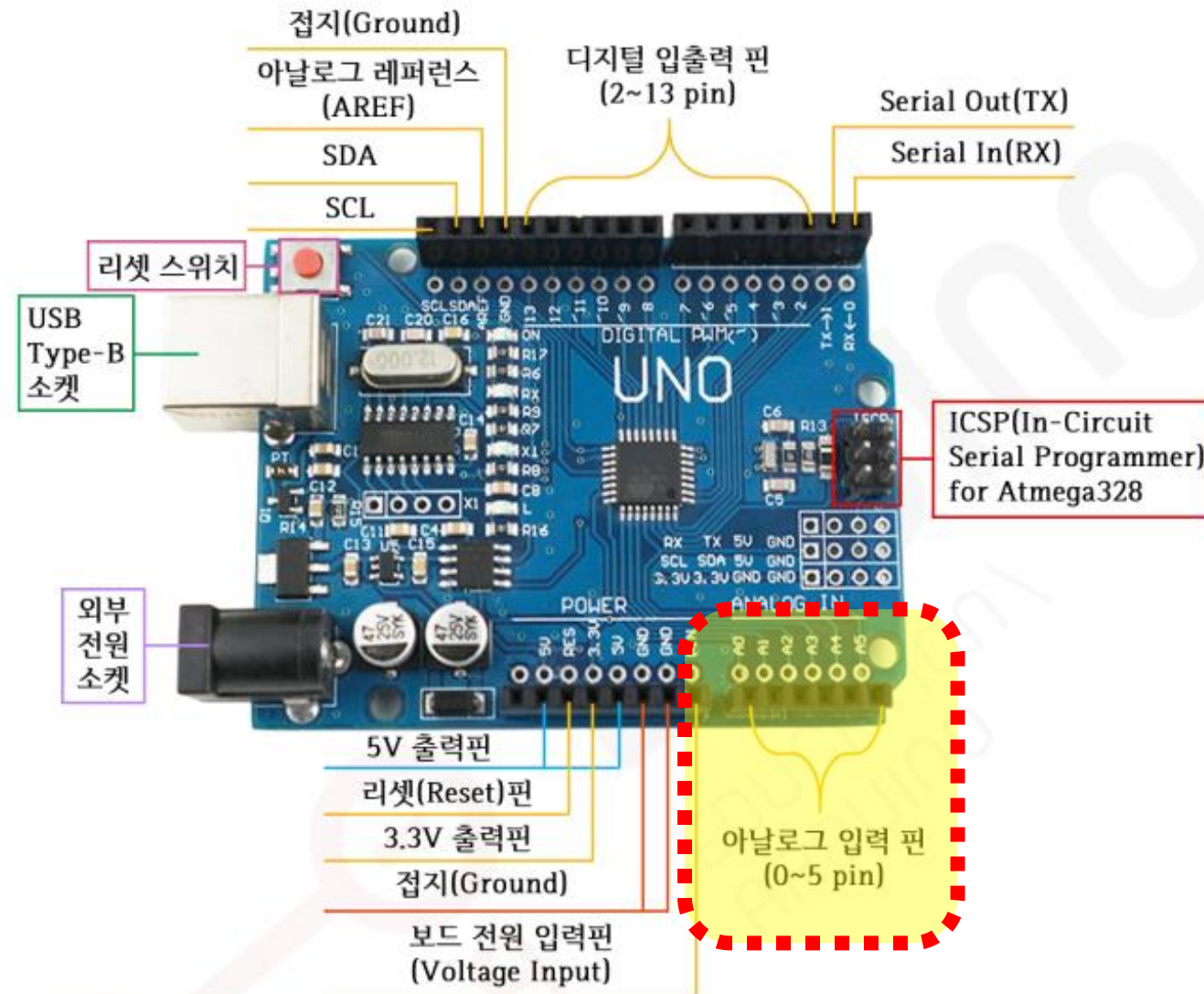
샘플링 → 양자화 → 부호화

• 부호화(Coding)

- 표본화 과정으로 아날로그 신호를 일정한 전압 레벨의 구간을 나눈 영역에 이진 수를 강제적으로 대응 시키는 과정



아두이노 우노 UNO SMD R3 호환보드



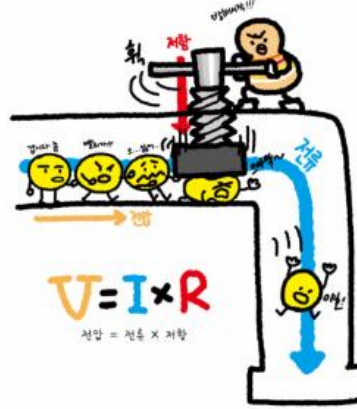
analogRead(*pin*)

- 지정된 아날로그 핀(A0~A5)으로부터 전압(0 ~ 5V)을 0 ~ 1023의 단계로 읽어 냄
- pin : A0 ~ A5까지 아날로그 입력 핀

```
void loop()
{
    int value = analogRead(A0) //아날로그핀 A0로부터 0~1023
                                //까지의 1024단계 값을 읽음
}
```

가변저항(Potentiometer)

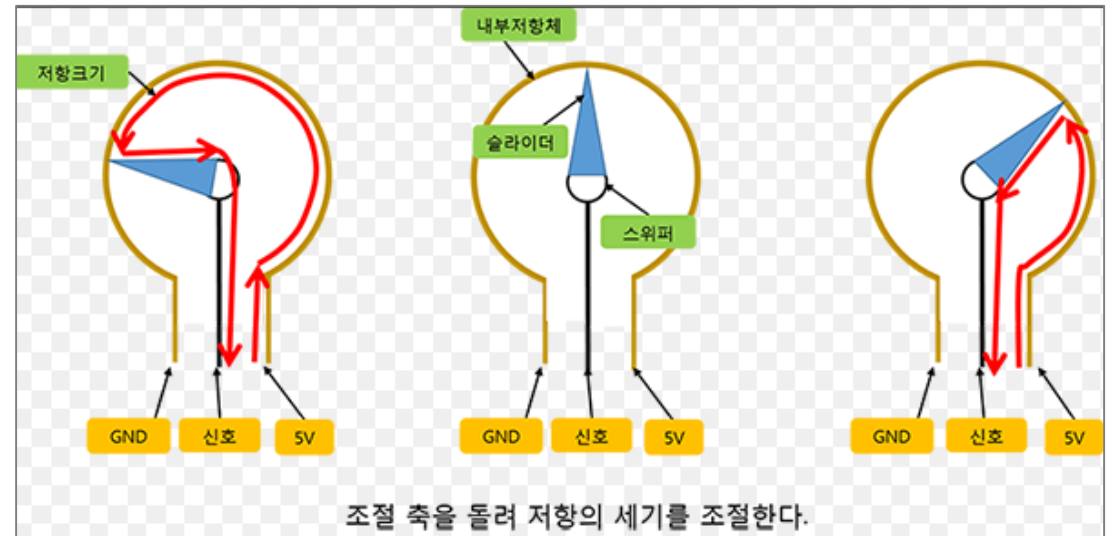
- 임의의 저항 값으로 전류의 조절이 가능한 저항기
 - 전기 저항의 크기를 조절하면 전류의 크기도 조절함



- 최소저항 0Ω , 최대 저항의 종류는 $1K$, $10K$, $100K\Omega$ 등
- 슬라이더의 위치에 따라 저항 값이 변화(아날로그 신호 표현 가능)
- 가변저항 사용 예
 - 음향 장치의 볼륨
 - 조명 밝기 조절 장치 등

가변저항(Potentiometer)

- 공통적으로 세 개의 단자
- 양 끝의 두 단자는 전원(5V)과 그라운드(GND)에 연결
- 가운데 단자는 와이퍼(Wiper)라고 하여 가변저항의 변경된 저항 값을 읽을 수 있는 단자
- 저항은 극성이 없는 전자소자이기 때문에 전원과 그라운드는 서로 바꿔서 연결해도 무방
- 하지만 방향을 변경하면 가변저항의 값의 범위가 뒤바뀐다는 점은 기억해야 함



실습 1: 전압 측정하기

- 아날로그 핀에서 읽어 낸 값(0~1023)을 전압(0~5V)으로 변경하기

1. 회로구성

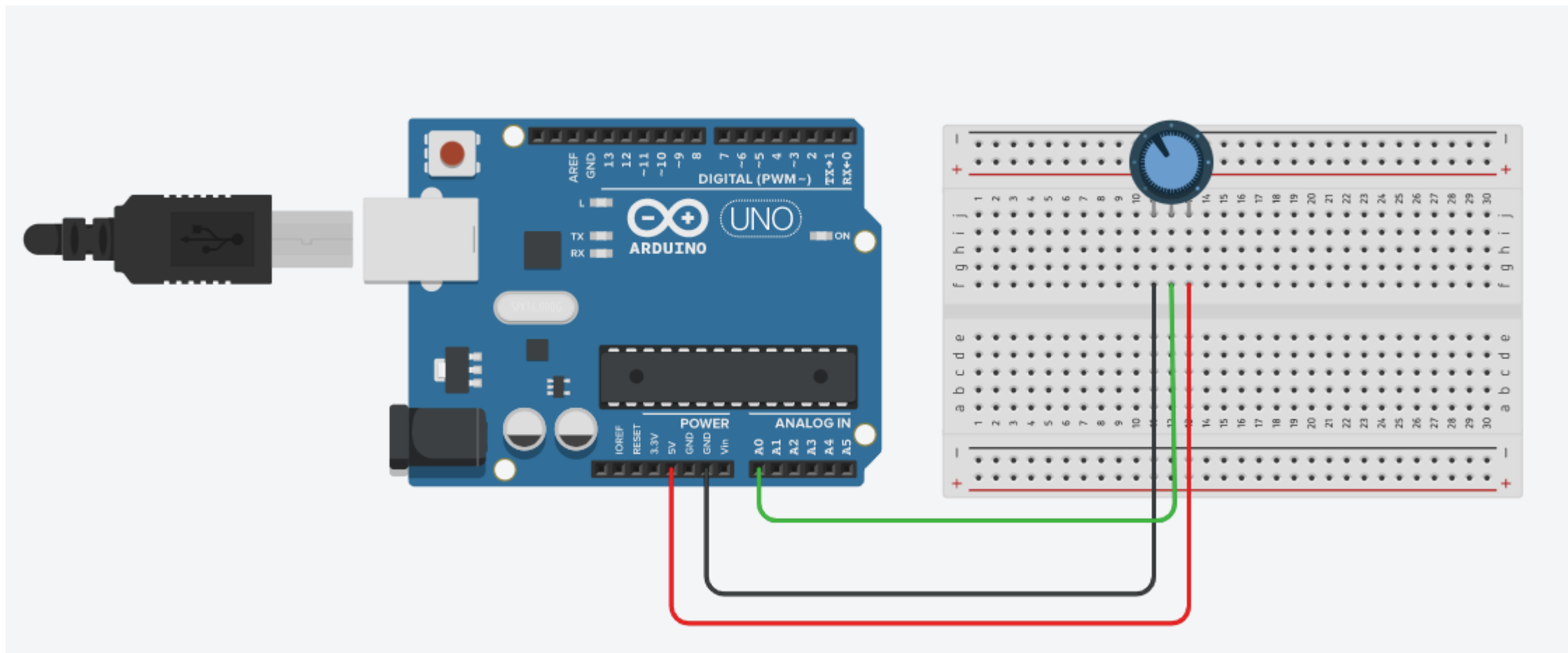
- 준비물 : 가변 저항(분압기), 케이블

2. 스케치 프로그래밍

- 아날로그 입력 값을 전압으로 변환하여 시리얼모니터에 출력

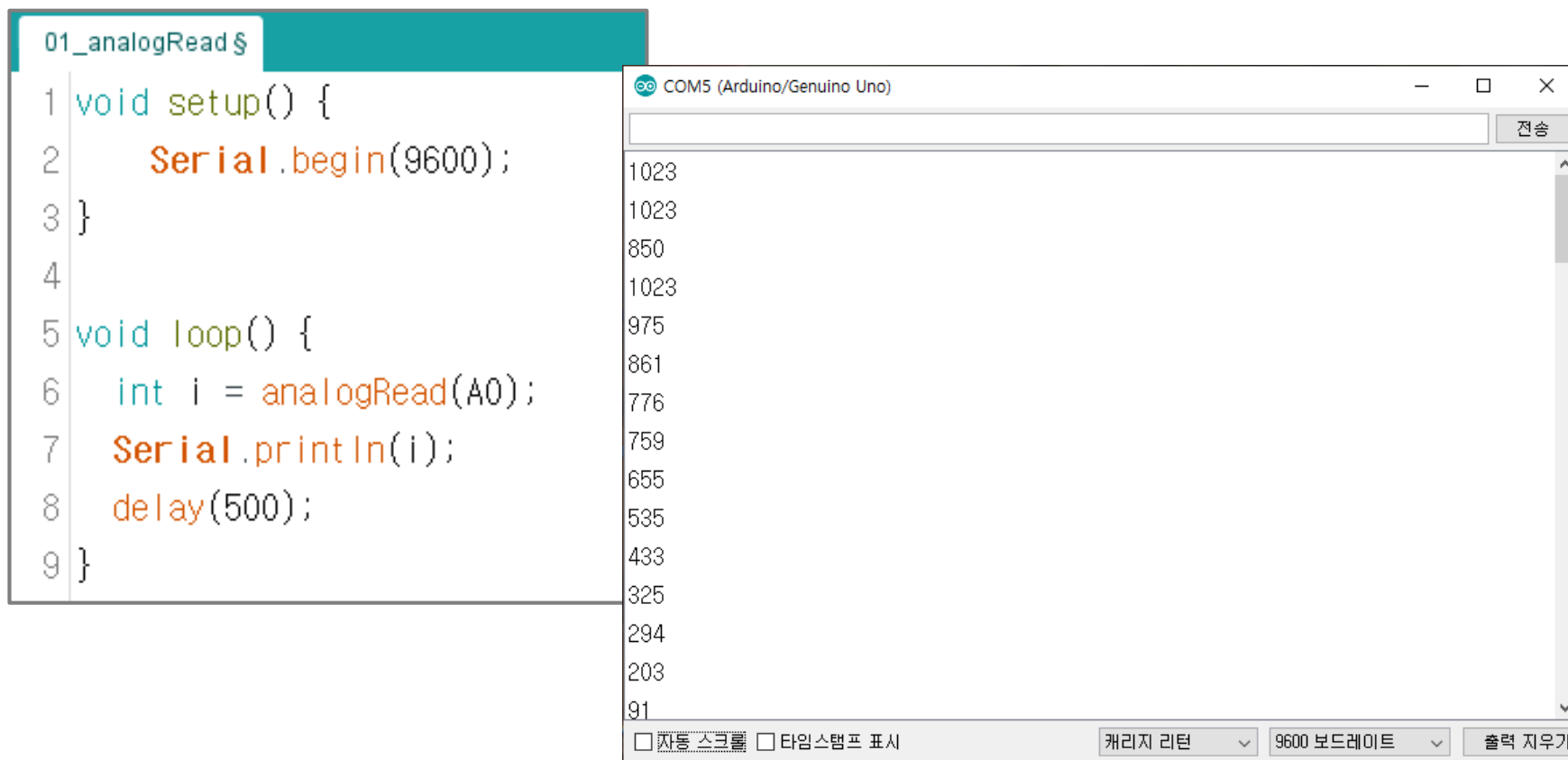
3. 실행

1) 회로구성



2) 스케치 프로그래밍

- 아날로그 핀에서 읽어 값 읽어내기(0~1023)
- 가변저항의 와이퍼를 오른쪽 왼쪽으로 돌려 저항의 크기를 변경해 본다.



The image shows the Arduino IDE interface. On the left, the sketch editor displays a program named '01_analogRead \$'. The code is as follows:

```
1 void setup() {  
2     Serial.begin(9600);  
3 }  
4  
5 void loop() {  
6     int i = analogRead(A0);  
7     Serial.println(i);  
8     delay(500);  
9 }
```

On the right, the serial monitor window is open, titled 'COM5 (Arduino/Genuino Uno)'. It shows a list of values printed by the program: 1023, 1023, 850, 1023, 975, 861, 776, 759, 655, 535, 433, 325, 294, 203, and 91. The values decrease from top to bottom, indicating the potentiometer's wiper is being moved from the right end to the left end. The serial monitor has a '전송' (Send) button and a scroll bar. At the bottom, there are checkboxes for '자동 스크롤' (Auto scroll) and '타임스탬프 표시' (Show timestamps), and dropdown menus for '캐리지 리턴' (Carriage return) and '9600 보드레이트' (9600 baud rate), along with a '출력 지우기' (Clear output) button.

- 전압으로 변환하여 출력

```
01_analogRead$
1 int i;    //아날로그핀 값 저장 변수
2 float v;  //전압 값 저장 변수
3
4 void setup() {
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     i = analogRead(A0);    //아날로그 핀 A0에서 핀값
10    v = (5.0/1023.0)*(float)i; //0~5V 값으로 변환
11    Serial.print("전압(V) : "); //시리얼모니터에 출력
12    Serial.println(v);
13    delay(500);
14 }
```

$$\text{전압} = \frac{5}{1023} \times \text{아날로그 핀값}$$

COM5 (Arduino/Genuino Uno)

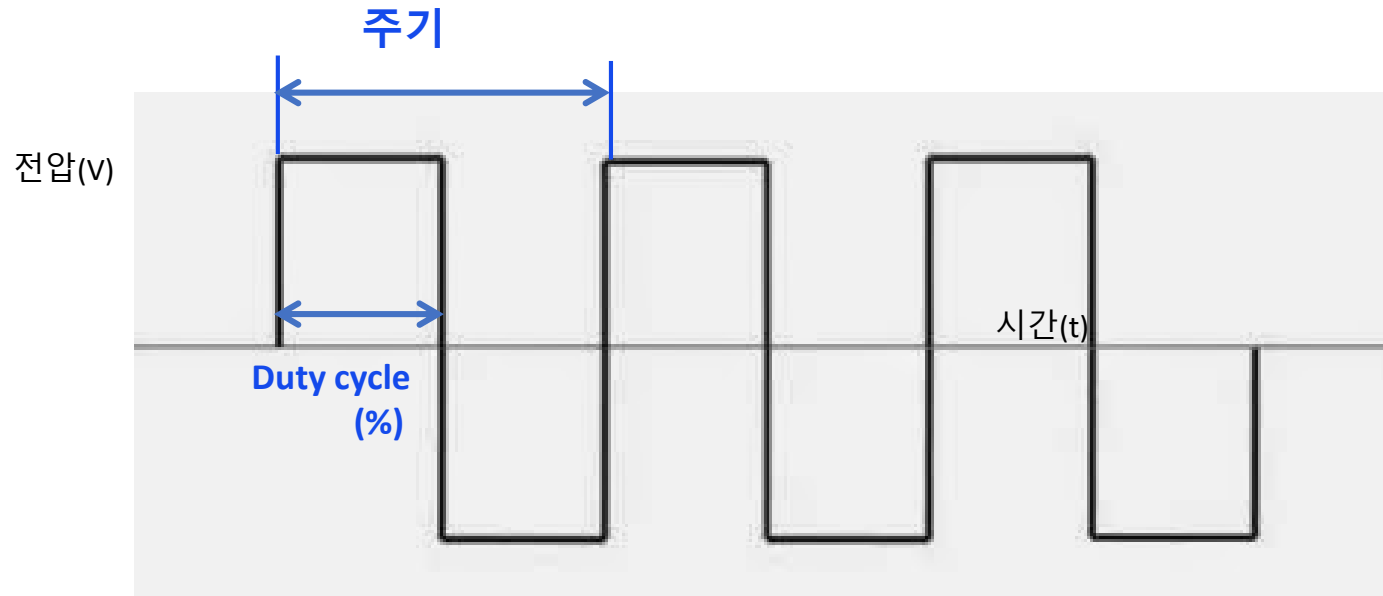
```
전압 : 5.00
전압 : 5.00
전압 : 5.00
전압 : 5.00
전압 : 4.16
전압 : 3.60
전압 : 3.23
전압 : 2.89
전압 : 2.43
전압 : 1.59
전압 : 1.19
전압 : 0.84
전압 : 0.00
```

3. 아날로그 출력

- 아두이노 MCU(Micro Controller Unit)
 - 0V와 5V만 인식 가능
 - 아날로그 값을 만들어 낼 수 없음
 - **PWM**(Pulse Width Modulation) : 디지털값으로 아날로그값과 유사하게 변조
 - **PWM**이 가능한 디지털 핀(" ~ "기호)
 - 3, 5, 6, 9, 10, 11
 - 이 핀들이 값을 256 단계로 구분하여 출력(0~255)
- 아두이노는 아날로그 값을 **ADC**를 통해 디지털신호로 변환하고 **PWM**(Pulse Width Modulation : 펄스 폭 변조)으로 아날로그 출력과 유사하게 출력한다.

아날로그 출력

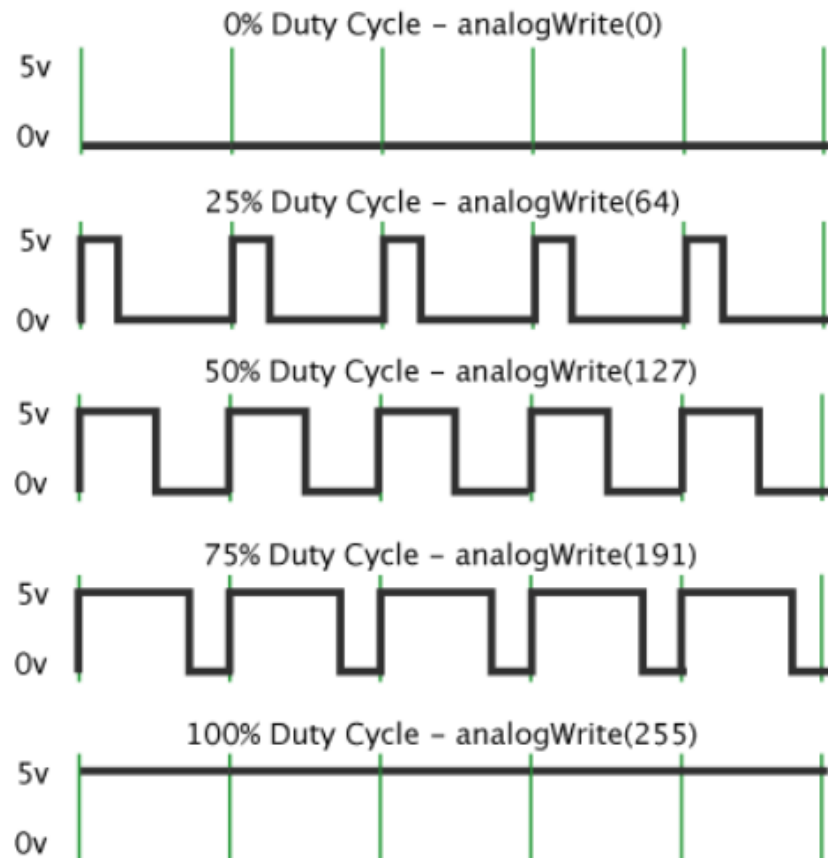
- PWM(Pulse Width Modulation)
 - 펄스 폭을 컨트롤 하는 주기 제어방법
 - 사각파의 출력으로 아날로그 값과 유사하게 0V와 5V사이의 값을 출력



- duty cycle: 한 주기 내에서 HIGH 가 차지하는 비율

아날로그 출력

- **PWM(Pulse Width Modulation) : 펄스 폭 변조**
 - 디지털 신호를 아날로그 회로처럼 제어하여 다양한 값으로 출력하는 방법

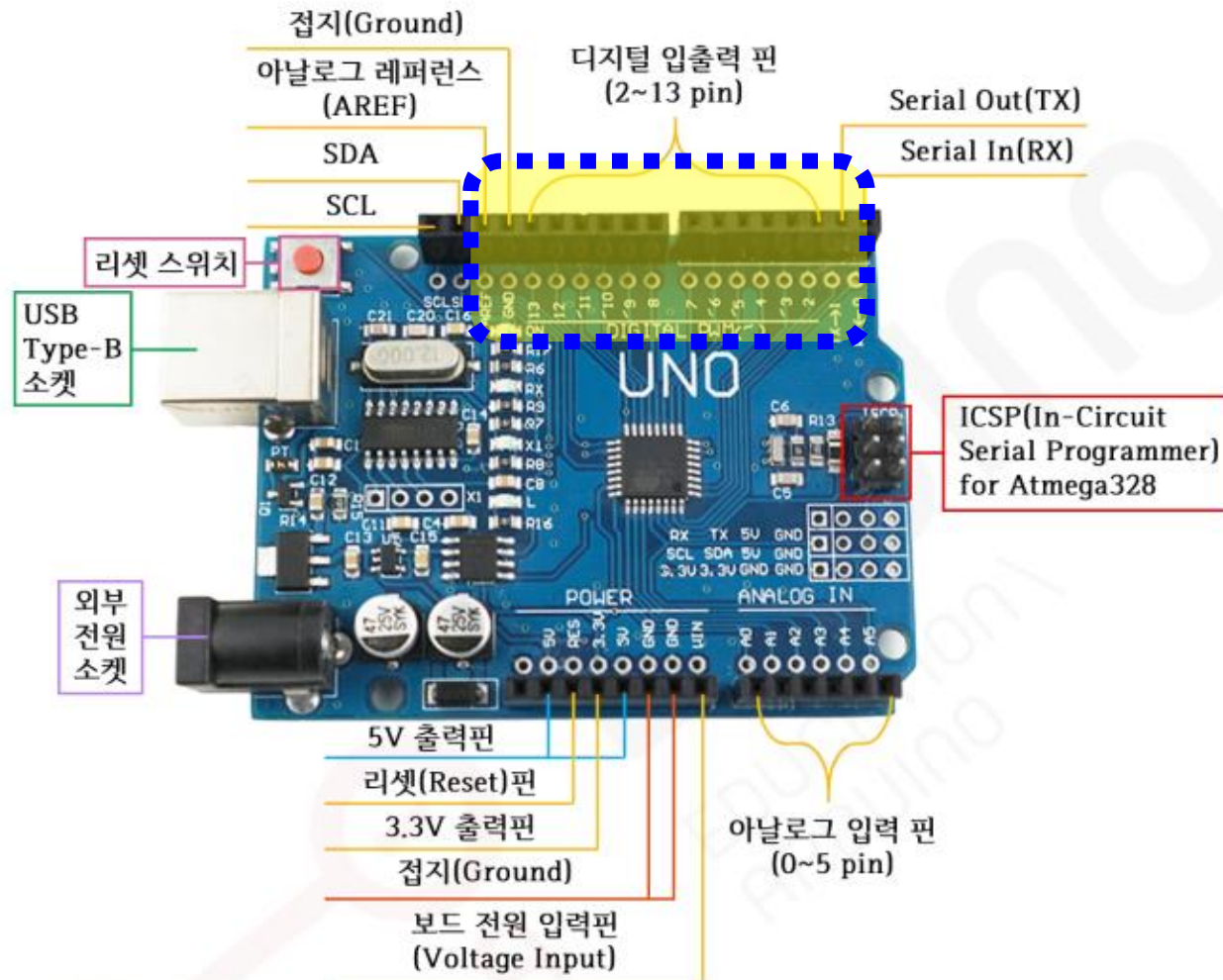


펄스의 평균값으로 다양한 값 표현

PWM기능을 가진 핀 :

- 디지털핀에서 "~" 기호가 있는 핀
- 3, 5, 6, 9, 10, 11(6개)

아두이노 우노 UNO SMD R3 호환보드



analogWrite(*pin*, *value*)

- **PWM** 파형을 만듦
 - pin : PWM 파형이 출력될 핀 번호
 - value : 출력할 파형의 세기
 - duty cycle을 나타내는 0~255사이
 - 0은 0%를 255는 100%를 의미
- 출력 핀으로 설정할 필요 없음(pinMode 설정 필요 없음)

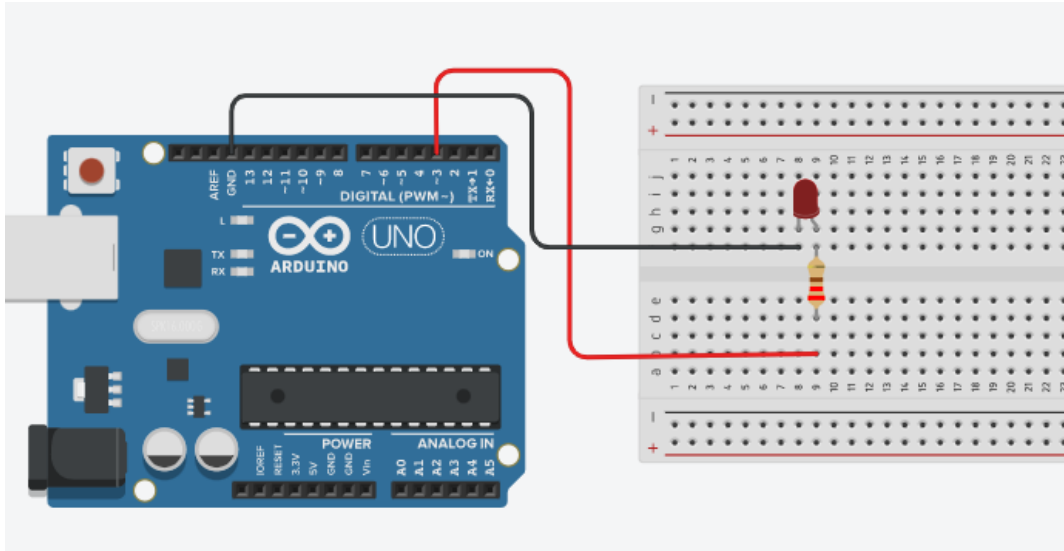
```
void loop()
{
    int value = 100;
    analogWrite(3, value);
}
```

map(value, fromLow, fromHigh, toLow, toHigh)

- 한 범위에서 다른 범위로 값을 매핑
- **정수값**만을 사용함
 - value : 변환시킬 값
 - fromLow : 값의 현재 범위의 하한값
 - fromHigh : 값의 현재 범위의 상한값
 - toLow : 값의 현재 범위의 하한값
 - toHigh : 값의 현재 범위의 상한값

```
void loop()  
{  int val = analogRead(0);  
    val = map(val, 0, 1023, 0, 255);  
    analogWrite(9, val);  
}
```

실습2 : 아날로그 출력으로 led 밝기 변화 시키기



```
int green = 3; // PWM이 가능한 핀
```

```
void setup() {  
  
}
```

```
void loop() {  
  
    analogWrite(green, 0);  
    delay(1000);  
    analogWrite(green, 100);  
    delay(1000);  
    analogWrite(green, 200);  
    delay(1000);  
    analogWrite(green, 255);  
    delay(1000);  
  
}
```


과제 1 : 가변저항으로 밝기 변화하는 스탠드 조명 만들기

- 가변저항(Potentiometer)을 사용하여 LED 밝기를 변화시키기

1. 회로구성

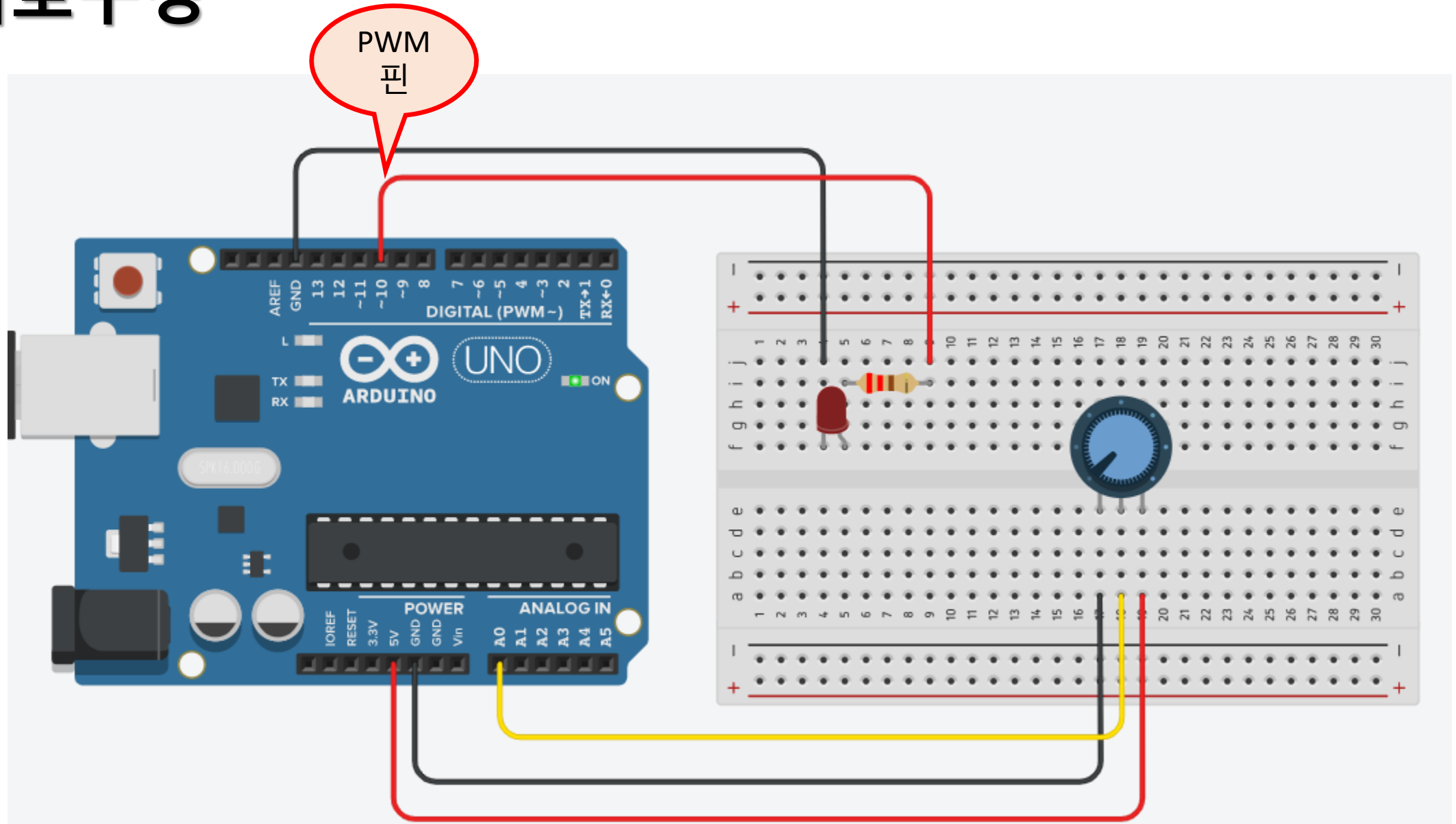
- 준비물 : 가변저항, LED, 220Ω 저항, 케이블

2. 스케치 프로그래밍

- 가변저항에 따른 값(0~1023)을 읽어 디지털값(0~255)값으로 변환하여 LED 밝기 변화를 제어

3. 실행

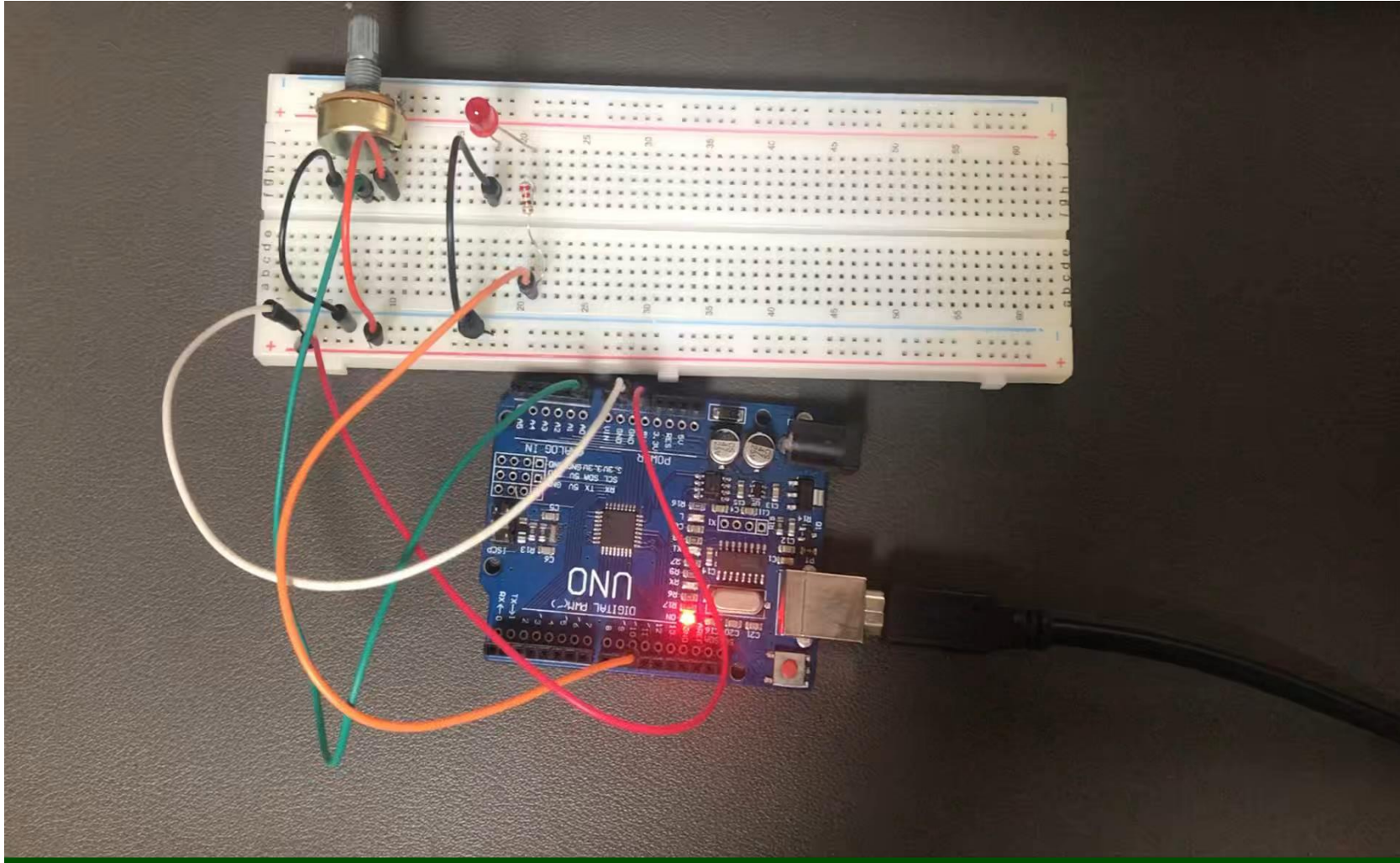
1) 회로구성



2) 스케치 프로그래밍



3) 실행



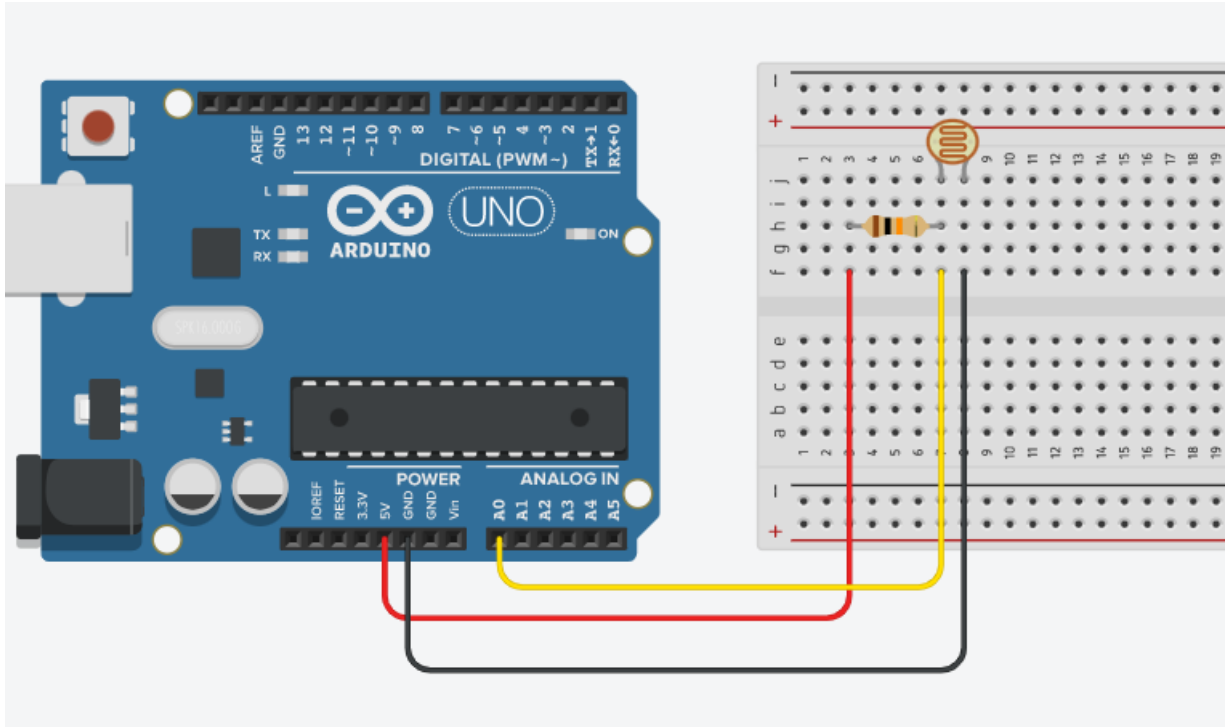
조도센서(Photoregister)

- 받아들이는 빛의 양에 따라 내부 저항값이 변하는 가변저항으로 포토레지스터라고도 불림



- 극성 없음
- 밝을 수록 저항값이 낮아지고($1\text{K}\Omega$), 어두울 수록 저항값($50\text{K}\Omega$)이 높아짐 (빛의 양과 저항값은 반비례)
- $10\text{K}\Omega$ 저항의 용도
 - 빛이 너무 밝아 조도센서의 저항이 0이 되어도 과도한 전류가 흐르지 않도록 함
- 사용 예
 - 가로등 켜고/끄기
 - 빛에 따라 모터 돌리기 등

실습 3 : 조도센서 값 읽어 내기



```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
  
    int value = analogRead(A0);  
  
    Serial.println(value);  
}
```

과제 2 : 조도센서를 사용하여 가로등 불빛 밝기 조절하기

- 주위가 어두워지면 가로등 불빛이 켜지 듯 조도센서(photo resistor) 사용하여 LED 밝기를 제어

1. 회로구성

- 준비물 : 조도센서, LED, $10K\Omega$, 220Ω , 케이블 등

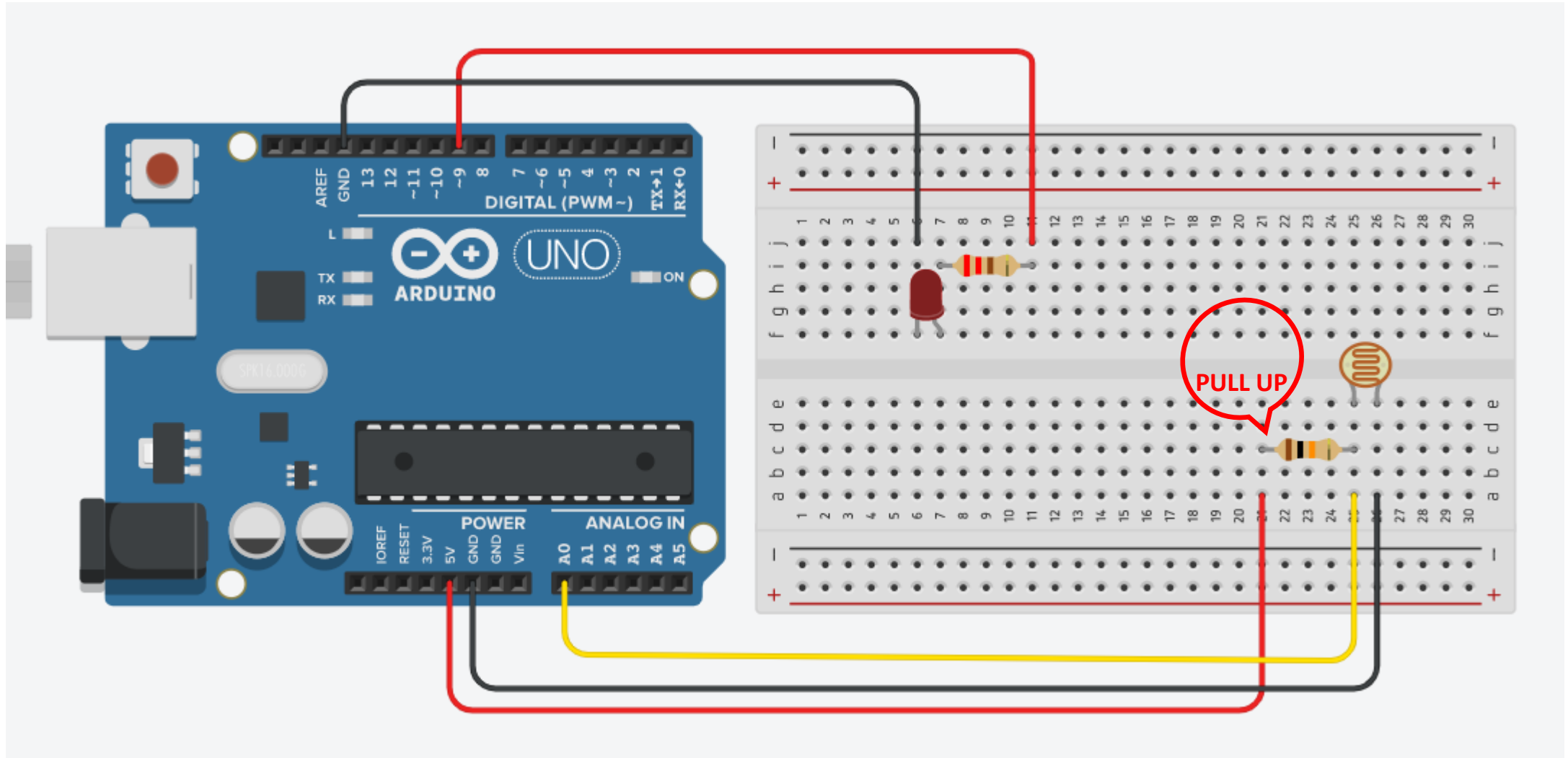
2. 스케치 프로그래밍

- 조도센서 값($0\sim1023$)을 읽어 디지털값($0\sim255$)값으로 변환하여 LED에 출력

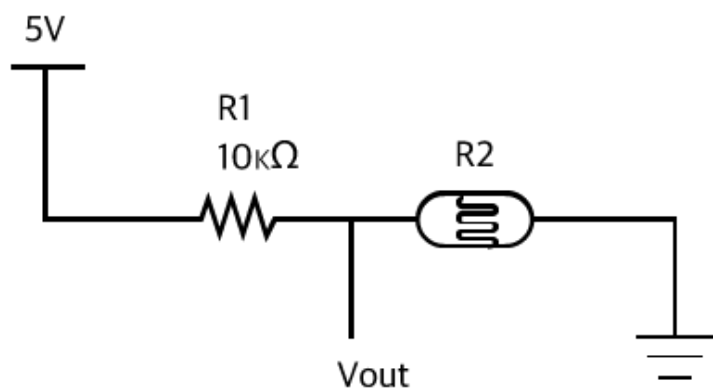
3. 실행

1) 회로구성

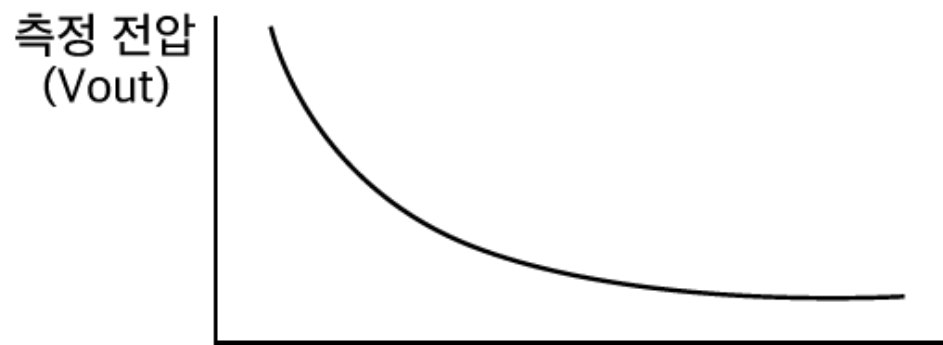
- 조도 센서는 PULL UP 저항 사용



PULL UP 저항과 측정 전압



풀업 저항 사용



풀업 저항 사용시 밝기에 대한 측정 전압

2) 스케치 프로그래밍



3) 실행

