사물인터넷

Chapter 2.
디지털 출력과 입력

1. 디지털 출력: LED 제어
2. 시리얼 모니터로 LED 제어
3. 풀업/풀다운 실험
4. 디지털 입력: 버튼을 이용한 디지털 제어

학습목표

- 디지털신호와 아날로그 신호를 설명할 수 있다.
- 저항을 읽을 수 있다.
- 풀업 회로와 풀다운 회로의 특징을 설명 할 수 있다.
- 시리얼 모니터에서 데이터를 읽을 수 있다.
- 아두이노 스케치에서 핀 모드를 출력으로 설정 할 수 있다.
- 아두이노 스케치를 업로드 할 수 있다.

1. 디지털 신호와 아날로그 신호

1) 디지털 신호

- 자료(data)를 표현하는 최소 단위를 수치화
- 이산적(離散的, 셀 수 있는)인 수치를 사용

- 특징
  - 데이터가 특정 값으로 표현
  - 자료의 변형 없이 저장, 복제, 삭제, 편집 등이 가능
  - 디지털 신호의 형태는 시간에 따라 신호의 크기가 특정한 값을 나타내는 막대그래프 모양으로 표현할 수 있음 (그림 2.1)

1. 디지털 신호와 아날로그 신호

- 디지털 신호 형태

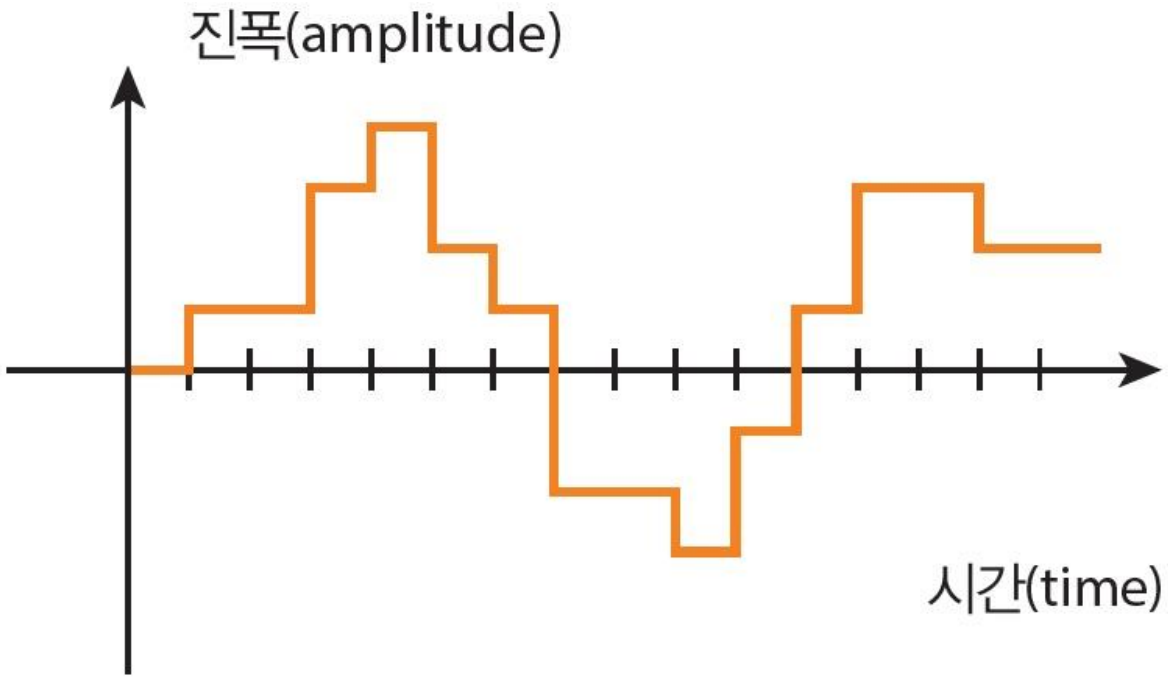

그림 2.1 디지털 신호의 형태

1. 디지털 신호와 아날로그 신호

2) 아날로그 신호

- 자료를 물리량으로 나타냄
- 자연에서 발생되는 신호
  - 예)빛의 정도, 소리의 높낮이, 바람의 세기 등
- 자연에서 발생되는 자료는 신호가 끊어짐이 없는 연속적인 형태
- 아날로그 신호는 연속적인 형태로 표현 (그림 2.2)
- 아날로그 신호를 일정한 주기로 샘플링하여 디지털로 변환

1. 디지털 신호와 아날로그 신호

- 아날로그 신호 형태

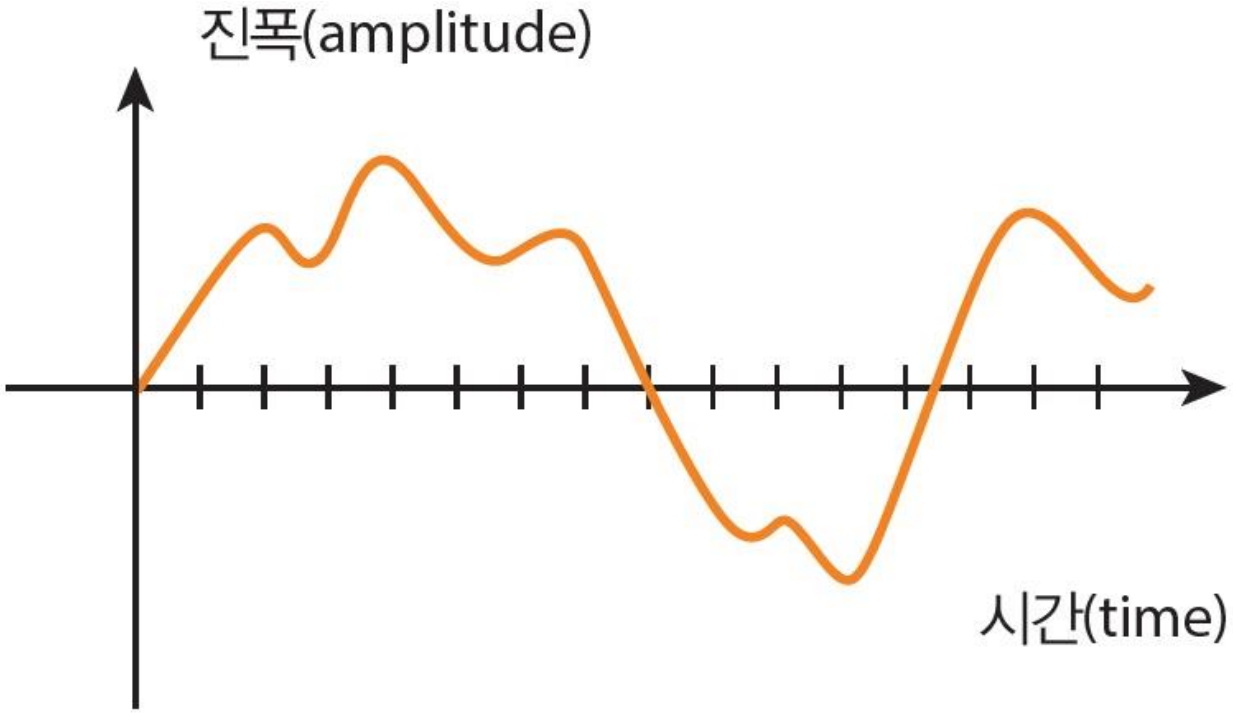

그림 2.2 아날로그 신호의 형태

1. 디지털 신호와 아날로그 신호

- 디지털 샘플링 주기
  - 디지털 데이터는 일정 주기로 추출
  - 샘플링 주기에 따라 데이터 품질에 영향
  - 샘플링 주기가 짧을수록 원래 데이터와 비슷해짐

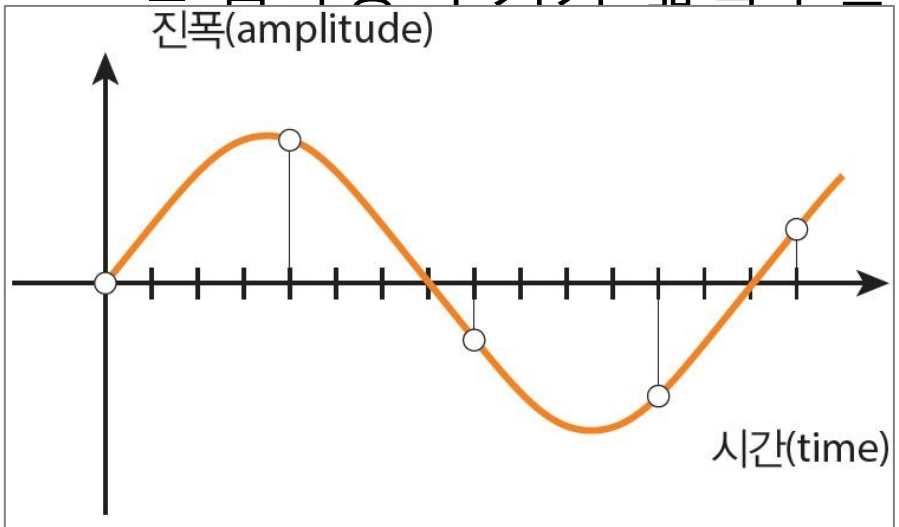

그림 2.3 샘플링 주기를 길게한 예

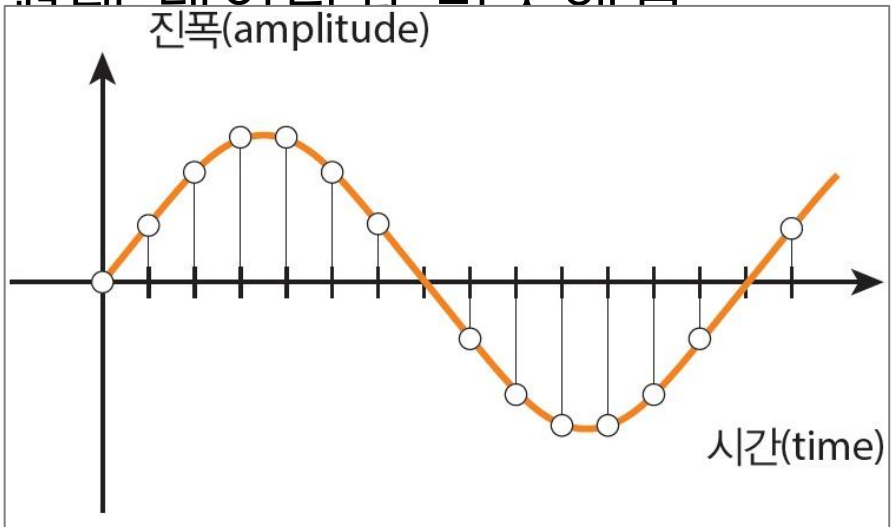

그림 2.4 샘플링 주기를 짧게한 예

1. 디지털 신호와 아날로그 신호

[아두이노 보드]

- 디지털 입출력
  - 핀 0번 ~ 핀 13번
- 아날로그 입력
  - 핀 A0 ~ A5
- 아날로그 출력
  - PWM 방식
  - ~3, ~5, ~6, ~9, ~10, ~11번

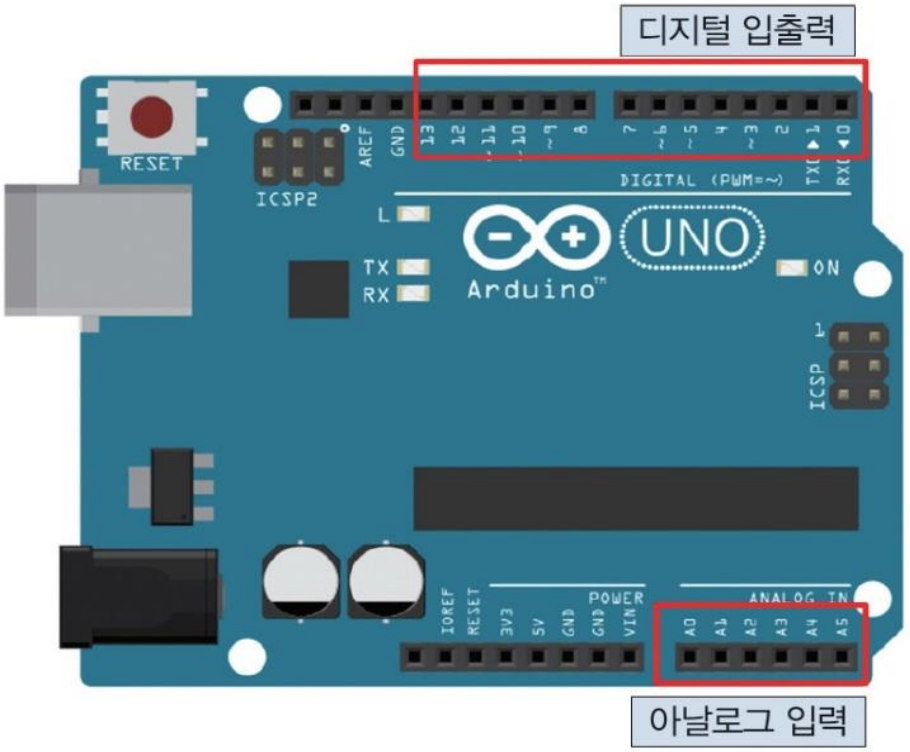

그림 2.5 아두이노의 디지털과 아날로그 입출력

1. 디지털 신호와 아날로그 신호

3) LED (Light Emitting Diode)

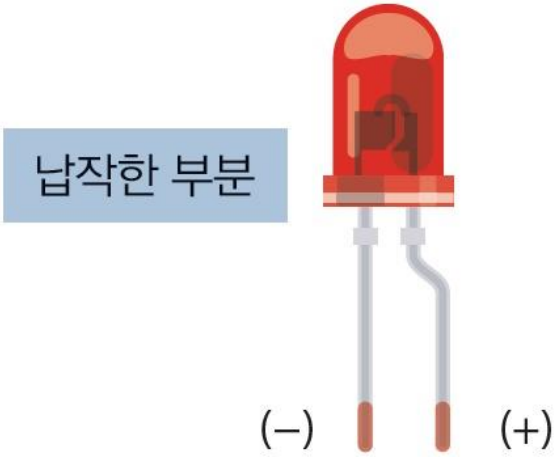

그림 2.6 LED 부품 모양

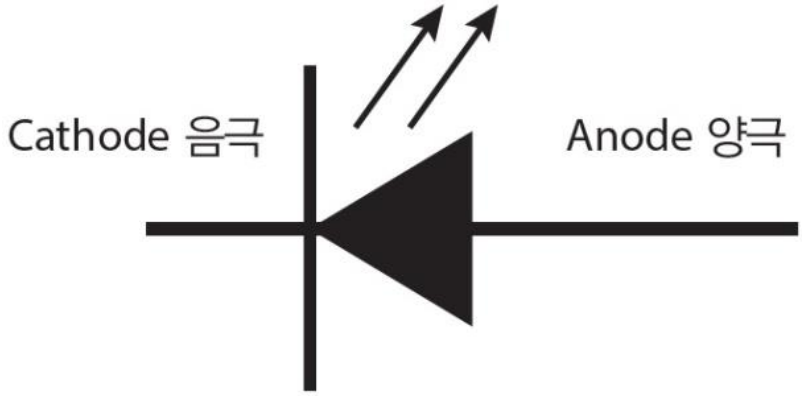

그림 2.7 LED 기호

- LED는 두 개의 다리가 연결되어 있음
- 하나는 상대적으로 조금 길고, 다른 하나는 조금 짧음
- 긴 쪽이 양극(+)
- LED의 빛이 나오는 뚜껑 모양을 손으로 만져보면 납작한 곳이 음극(-)

1. 디지털 신호와 아날로그 신호

3) LED (Light Emitting Diode)

- 순방향으로 전압을 가했을 때 빛을 발생하는 반도체 부품
- 발광다이오드라고도 함
- LED는 방향성이 있어 양극(+)에서 음극 (-)으로 전류가 흐를 때 빛을 나타냄
- (-)극(Cathode:음극) / (+)극(Anode:양극)

1. 디지털 신호와 아날로그 신호

4) 저항 읽기

– 저항은 전류가 흐를 때 흐름을 방해해서 일정한 양만 흐르도록 해주는 역할

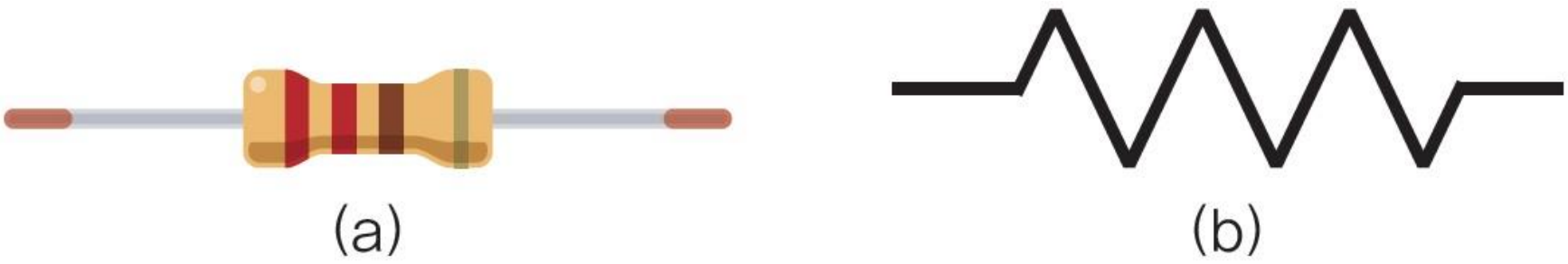

그림 2.8 저항 모양 및 기호

1. 디지털 신호와 아날로그 신호

4) 저항 읽기

- 저항의 크면 전류가 적게 흐름, 저항이 작으면 전류가 많이 흐름
  - 플라스틱은 전류가 흐르지 못해 저항이 아주 큼
  - 구리선은 저항이 거의 없어 전류가 잘 흐름
- 단위 : 옴(Ohm), 기호는 Ω로 표시
  - 4,700옴은 4.7KΩ으로, 10,000 옴은 10kΩ으로 표시
  - 1Ω : 1V(볼트)의 전압으로 1A(암페어)의 전류가 흐를 때의 저항
- 저항은 전류와 전압과 밀접한 관계
  - [계산식] 저항(R) = 전압(V) / 전류(I)

1. 디지털 신호와 아날로그 신호

4) 저항 읽기

- 아두이노에서 회로를 만들 때 저항을 자주 사용
- LED를 깜박이는 회로를 만들 때 220Ω(옴) 저항을 사용
- 저항은 각각 용량이 다르며 용량은 저항의 표면에 색상 띠로 표시되어 있음
- 색상 값은 숫자의 의미
- 색상을 읽으면 저항의 용량을 알 수 있음
- 저항의 첫 번째 색 띠는 첫째 자리 숫자, 두 번째 색 띠는 둘째 자리 숫자, 그리고 세 번째 색 띠는 10의 배수를 나타냄

1. 디지털 신호와 아날로그 신호

4) 저항 읽기

표 2.1 저항 색상 값

저항 색상 띠 값									
검정	갈색	빨강	주황	노랑	초록	파랑	보라	회색	흰색
0	1	2	3	4	5	6	7	8	9

표 2.2 저항 색상 오차 값

네 번째 색 띠 오차 범위		
금	은	없음
5%	10%	20%

1. 디지털 신호와 아날로그 신호

4) 저항 읽기(색상 띠 3개)

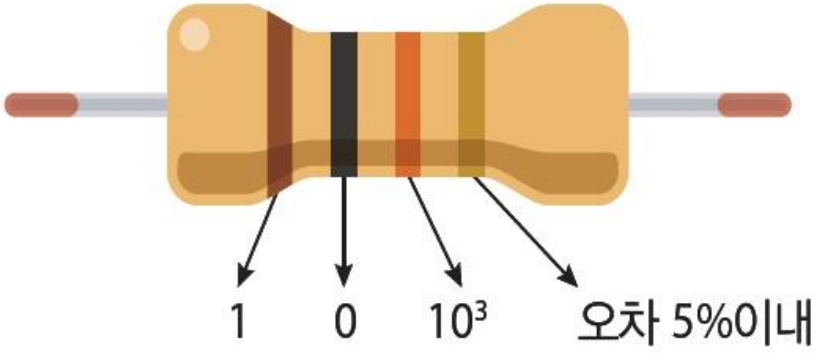

그림 2.9 저항 읽기 예

표 2.3 그림 2.9 저항의 용량

순서	첫 번째	두 번째	세 번째(0의 개수)	오차	저항값
색상	갈색	검정	주황	금색	10kΩ
값	1	0	10^3	5%이내	

1. 디지털 신호와 아날로그 신호

4) 저항 읽기(색상 띠 4개)

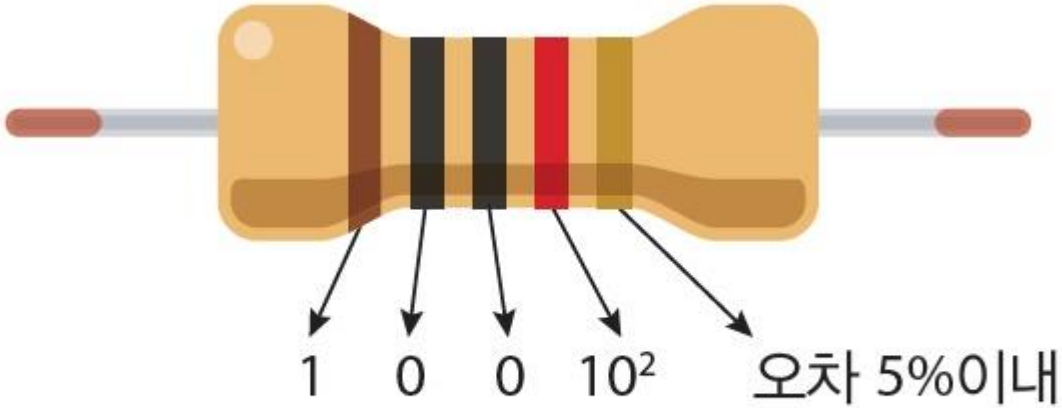

그림 2.10 띠가 네 개인 경우

표 2.4 그림 2.10 저항의 용량

순서	첫 번째	두 번째	세 번째	네 번째	저항값
색상	갈색	검정	검정	빨강	10kΩ
값	1	0	0	10^2	

2. 디지털 출력으로 LED 제어하기

- 디지털 출력은 마이크로컨트롤러에서 전류를 외부로 내보내는 것
- 5V를 내보내는 것 : HIGH
- 0V를 내보내는 것 : LOW
- 디지털 출력을 담당하는 핀은 1번부터 13번

2. 디지털 출력으로 LED 제어하기

- 디지털 입/출력 핀

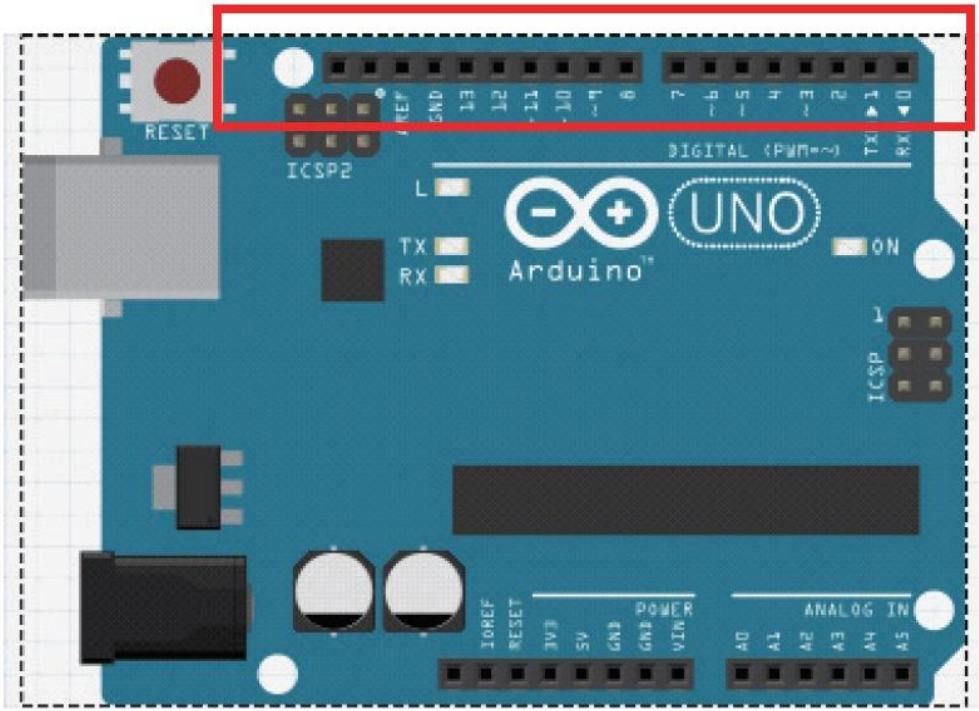

그림 2.11 아두이노 디지털 입력 출력 핀

2. 디지털 출력으로 LED 제어하기

1) 회로 만들기

- 필요한 부품
  - 아두이노, 브레드보드, 저항(220Ω), LED
- LED는 양극과 음극이 있으므로 반드시 확인
- 아두이노의 4번 핀과 LED의 (+)극인 긴 다리가 연결
- 아두이노의 GND가 LED의 (-)극인 짧은 다리에 연결

2. 디지털 출력으로 LED 제어하기

1) 회로 만들기

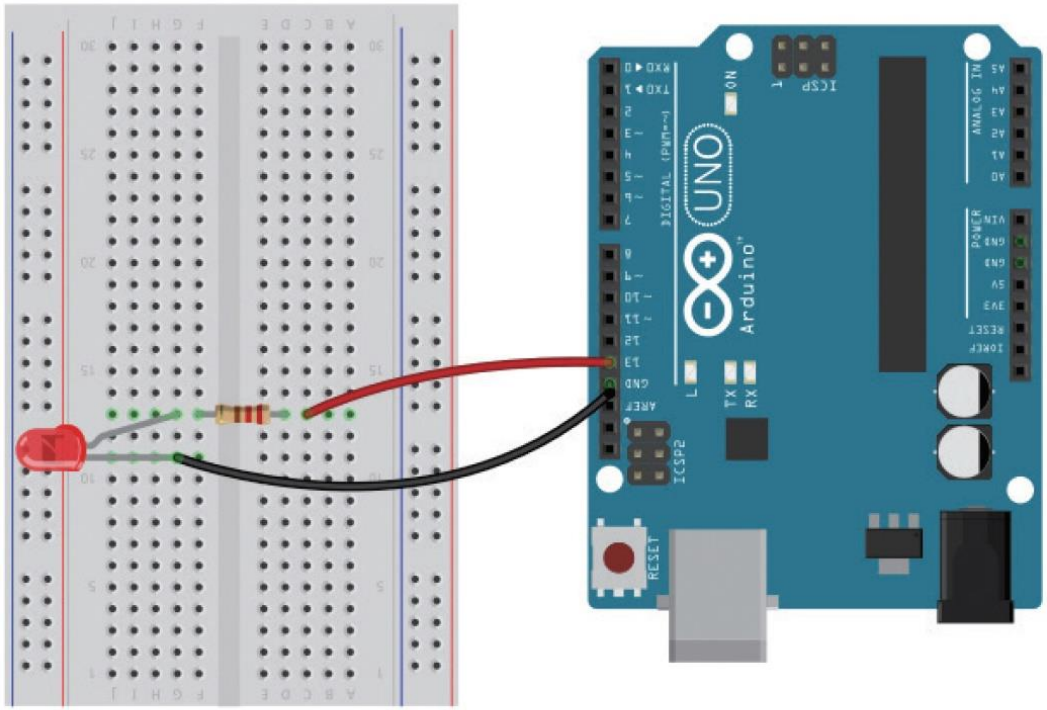

그림 2.12 LED 제어 배선도

2. 디지털 출력으로 LED 제어하기

2) 스케치 작성하기

- LED를 제어하는 스케치를 작성
- 아두이노 IDE를 실행시키고 편집 화면의 메뉴에서 [파일]-[예제]-[01.Basic-Blink]를 선택

```
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // wait for a second
}
```

2. 디지털 출력으로 LED 제어하기

[코드설명]

- '//'
  - 스케치 중에 문장의 제일 앞에 '//'로 표시된 부분은 설명문으로 스케치를 설명하는 역할
  - 이 부분은 코드가 실행되지 않음

- void setup() { }
  - 프로그램 실행 초기에 한번만 실행하는 부분
  - 일반적으로 핀모드 설정이나 시리얼 통신 설정 등을 이곳에 입력

- void loop() { }
  - 프로그램의 메인 코드를 입력하는 부분
  - 계속 반복하는 명령을 입력

2. 디지털 출력으로 LED 제어하기

[코드설명]

- pinMode(LED_BUILTIN, OUTPUT);
  - pinMode(핀번호, 모드)는 아두이노 핀의 모드를 설정
  - 'LED_BUILTIN'는 아두이노 보드에 기본으로 설정된 핀 번호인 13번 핀을 의미
  - 이 부분은 13으로 고쳐 적어도 됨
  - OUTPUT은 핀의 모드를 '출력'으로 설정한다는 의미(즉, 기본설정 핀 13)

2. 디지털 출력으로 LED 제어하기

[코드설명]

- 세미콜론(;)
  - 스케치의 한 문장이 마칠 때 반드시 세미콜론 ';'을 씀
  - 세미콜론을 적지 않으면 오류 발생
- digitalWrite(LED_BUILTIN, HIGH);
  - digitalWrite(핀번호, 값);
  - 핀모드가 설정되는 전류를 공급하여 디지털 데이터를 전달
  - LED_BUILTIN 는 기본설정 핀 번호 13번
  - HIGH 또는 LOW를 설정
  - HIGH는 5V를 공급하여 LED가 빛이 나게 하고
  - LOW는 0V를 전달하여 LED의 빛이 꺼지게 함

2. 디지털 출력으로 LED 제어하기

[코드설명]

- delay(1000);
  - delay(시간);
  - delay()함수는 괄호 안의 숫자 시간만큼 프로그램을 멈춤
  - 매개변수의 숫자는 ms 단위, 1000이면 1초
  - delay(500)이면 0.5초 동안 프로그램을 멈춤
  - delay()함수를 사용하지 않으면 명령어 실행이 너무 빨리 지나가서 정확한 결과를 확인하기 힘듦

2. 디지털 출력으로 LED 제어하기

- 간단한 스케치

```
void setup() {
  pinMode(13, OUTPUT); //핀 모드 출력으로 설정
}
void loop() {
  digitalWrite(13, HIGH); //LED 켜기
  delay(1000); // 1초간 기다리기
  digitalWrite(13, LOW); //LED 끄기
  delay(1000); // 1초간 기다리기
}
```

2. 디지털 출력으로 LED 제어하기

3) 컴파일 하기

- 스케치를 업로드하기 전에 먼저 컴파일을 실행
- 컴파일은 작성한 코드를 컴퓨터가 실행할 수 있는 실행 파일로 변환하는 과정
- 문법적인 오류(Syntax error)가 있으면 아두이노 편집 창 아래의 결과 확인 창에 주황색으로 표시됨
- 스케치의 코드를 정렬하는 단축키는 'Ctrl + t'

그림 2.14 컴파일 버튼

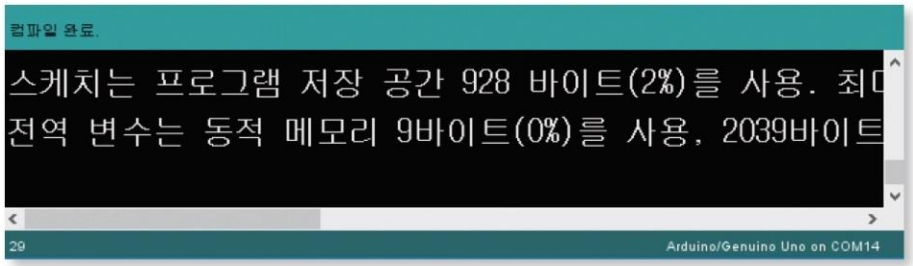

그림 2.15 컴파일 완료된 화면

2. 디지털 출력으로 LED 제어하기

4) 보드와 포트 확인하기

- (보드확인)
  - 컴파일을 한 다음 스케치를 아두이노에 업로드하기 위해 보드의 종류와 컴포트를 반드시 확인
  - '툴-보드-Arduino/Genuino Uno'가 선택되었는지 확인

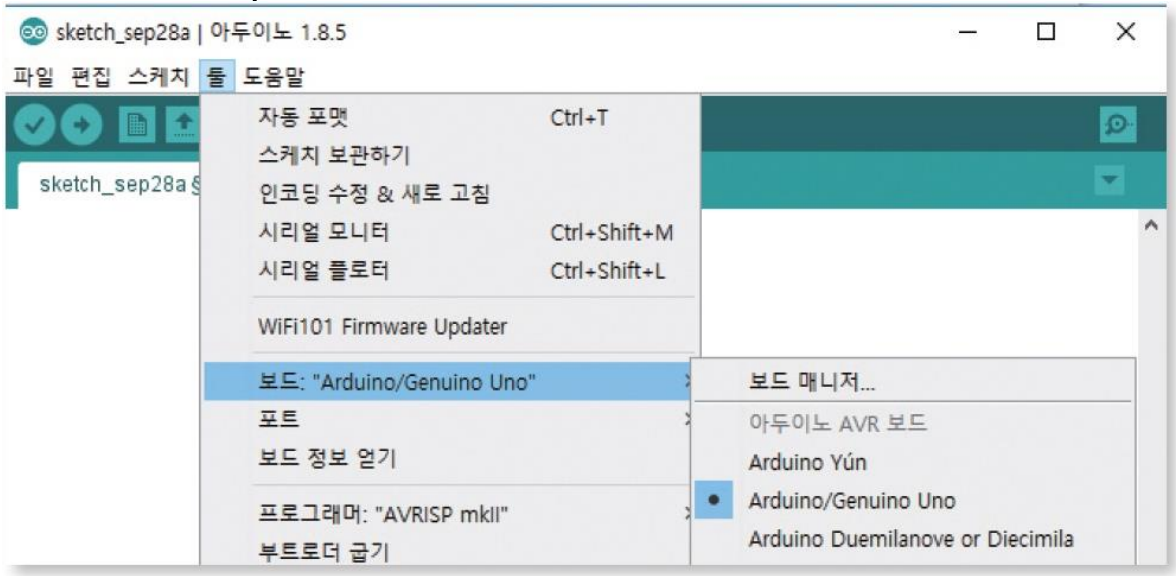

그림 2.16 보드 확인하기

2. 디지털 출력으로 LED 제어하기

4) 보드와 포트 확인하기

- (포트 확인)
  - 메뉴에서 '툴-포트-COM번호'를 선택
  - 컴포트 번호는 컴퓨터마다 숫자가 다름
  - 데스트탑 컴퓨터 : 'COM 1'번과 'COM 2'번을 제외한 높은 숫자 번호
  - 노트북 컴퓨터 : 'COM 1'이 할당
  - 최근 버전에서는 'COM 3(Arduino/Genuino Uno)'와 같이 표시됨

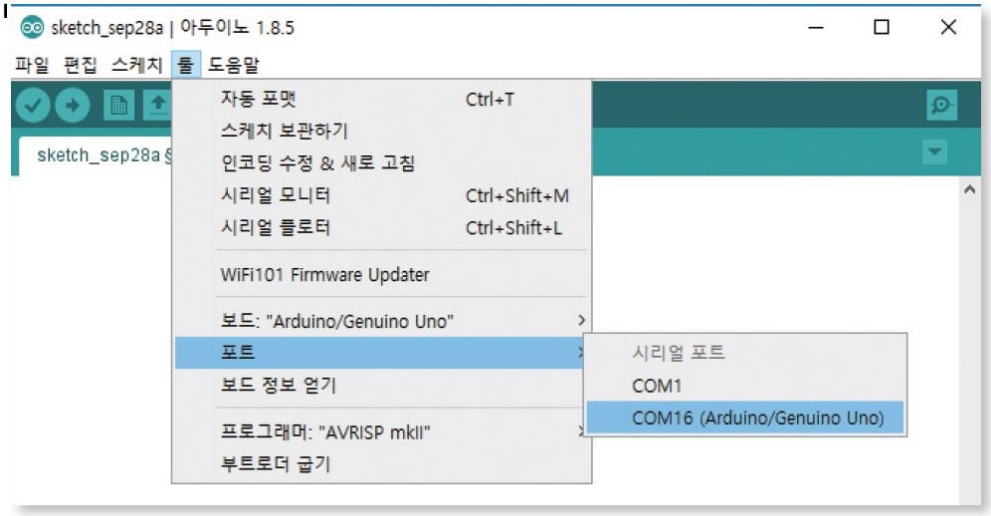

그림 2.17 포트 확인하기

2. 디지털 출력으로 LED 제어하기

5) 업로드하기

- 아두이노의 보드 설정과 컴포트 확인을 마쳤으면 업로드 버튼을 클릭하여 작성한 스케치를 아두이노에 업로드

그림 2.18 업로드 버튼

2. 디지털 출력으로 LED 제어하기

6) 동작 확인하기

- 아두이노에 연결된 LED가 1초에 한번씩 깜박이는지 확인

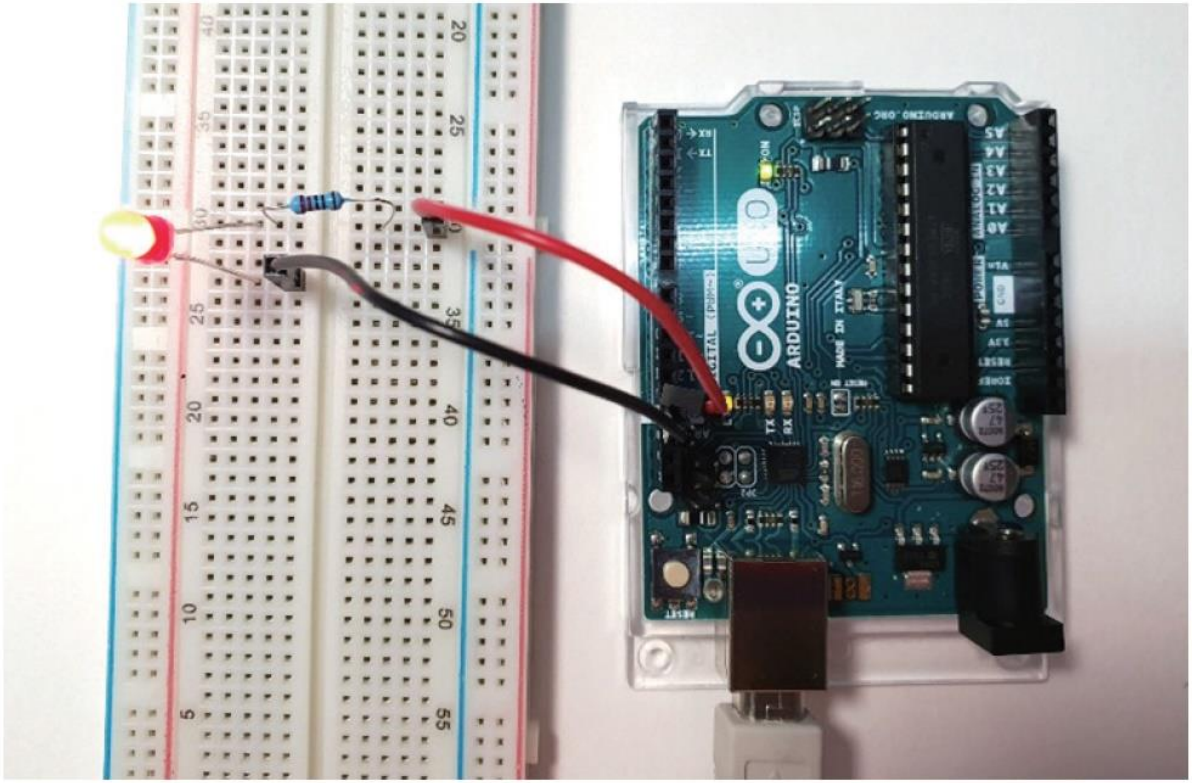

그림 2.19 LED 깜박이는 모습

3. LED 여러 개 깜박이기

- 준비물
  - 아두이노, 브레드보드, LED 여러 개, 저항 여러 개, 점퍼선 여러 개
- 1) 회로 만들기
  - LED는 2,4,6번

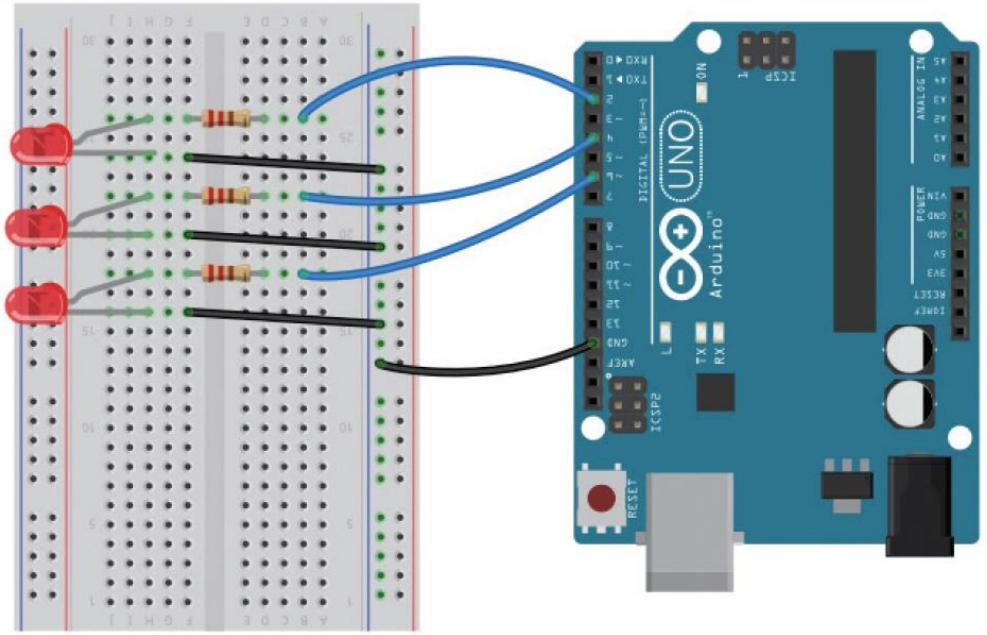

그림 2.20 LED를 세 개 연결한 모습

3. LED 여러 개 깜박이기

2) 스케치 작성하기

- LED가 여러 개 연결되어 있는 경우 LED를 연결한 아두이노의 핀의 모드 설정과 동작을 각각 제어해야 한다.

```
int led1 = 2, led2 = 4, led3 = 6;
void setup() {
  pinMode(led1, OUTPUT); //핀모드 출력으로 설정
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT); }
void loop() {
  digitalWrite(led1, HIGH); //LED에 5V 공급
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  delay(1000);
  digitalWrite(led1, LOW); //LED에 0V 공급
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  delay(1000);
}
```

3. LED 여러 개 깜박이기

[코드설명]

```
int led1 = 2;
```

변수형 변수이름 저장 값

- int led1 = 2;
  - 변수 설정
  - 브레드보드에 여러 개의 LED를 연결할 때 핀번호를 변수(variable)로 지정
  - 이 스케치에서는 아두이노에 연결되는 핀 번호를 변수로 지정함
  - 변수는 특정한 값을 저장하는 메모리 공간
  - 변수에는 값의 형식, 변수 이름, 저장된 값이 지정됨

3. LED 여러 개 깜박이기

[코드설명]

- 'int'
  - 저장된 변수 값이 정수라는 의미
  - 정수는 1, 2, 3, 4와 같은 소수점이 없는 숫자
  - 소수점이 있는 숫자를 사용하려면 float(플로트)를 사용
- 'led1'
  - 변수의 이름
  - 이름을 만들 때는 영문 소문자로 시작하고 띄어쓰기 없음
- '2'
  - 저장된 값
  - 변수형이 정수형이므로 정수만 저장
  - 정수형이므로 소수점이 있거나 문자는 값으로 저장 안 됨

3. LED 여러 개 깜박이기

- [실행결과]

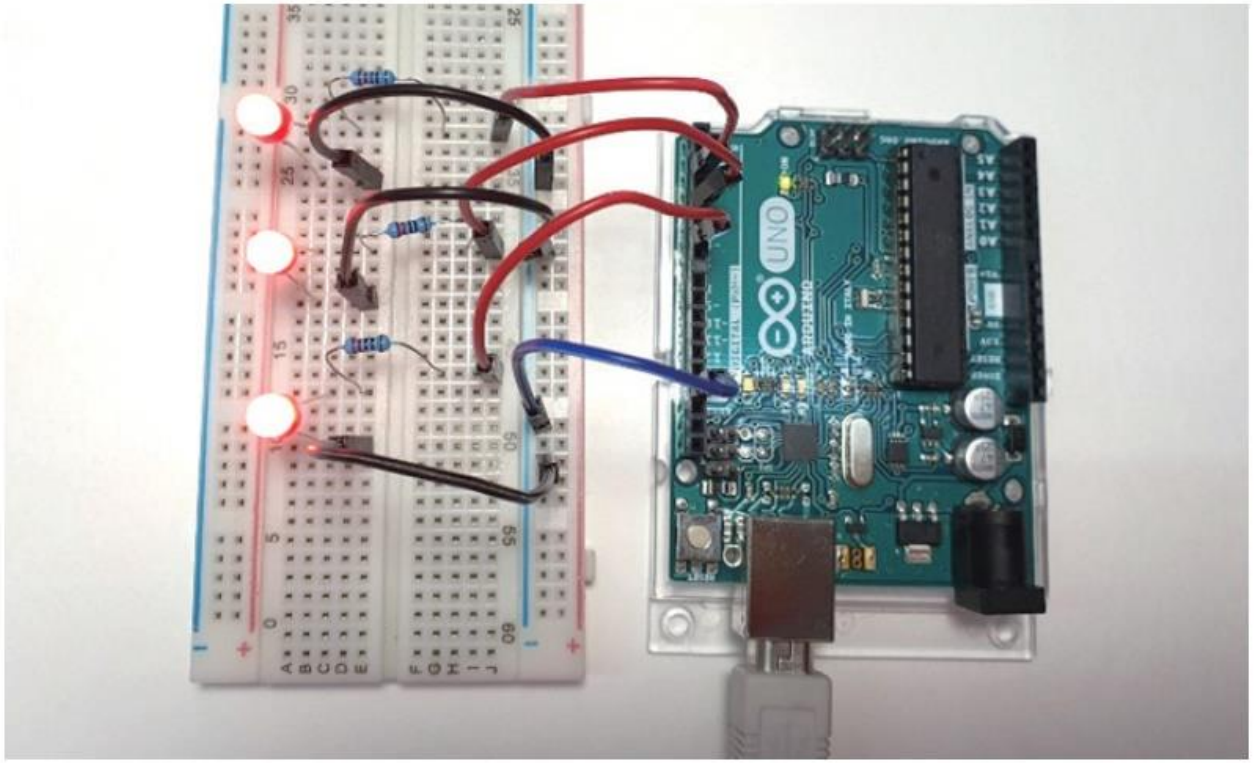

그림 2.21 LED 3개를 연결해서 실습하는 모습

3. LED 여러 개 깜박이기

- 배열과 for문을 이용한 LED 제어하기

```
int myLed[] = {2, 4, 6}; // LED 번호를 배열로 지정
void setup() {
  for (int i = 0; i < 3; i++) {
  pinMode(myLed[i], OUTPUT); //pinMode 설정
 }
}
void loop() {
  for (int i = 0; i < 3; i++) {
    digitalWrite(myLed[i], HIGH);
    delay(500);
    digitalWrite(myLed[i], LOW);
    delay(500);
 }
}
```

3. LED 여러 개 깜박이기

- [배열]
  - 배열은 인덱스(Index)번호로 접근이 되는 변수들의 모임

```
int myLed[] = {led1, led2, led3};
```

배열형 | 배열이름 | 초기화 | 저장 값

핀 번호	2	4	6
배열 번호	myLed[0]	myLed[1]	myLed[2]

3. LED 여러 개 깜박이기

- [배열]
  - int myLed[] = {led1, led2, led3};
  - int myLed[3] = {led1, led2, led3};
    // 두 형식 모두 가능

3. LED 여러 개 깜박이기

- [for 반복문]

```
for(int i = 0; i < 3; i++)
```

초기화 조건 증가

  - 'for' 문은 중괄호로 싸인 블록을 반복할 때 사용
  - 배열에 저장된 데이터를 사용할 때 for 반복문을 이용해서 사용
  - 초기화는 처음에 한번만 실행
  - 조건부에서 조건이 실행되고 , 실행이 True이면 증가 부분이 실행됨
  - 조건이 끝날 때까지 반복 후 종료

3. LED 여러 개 깜박이기

- [스케치 설명]

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

  - 초기값이 i= 0이고 i가 0부터 3보다 작을 때까지 반복하므로
  - i는 0, 1, 2가 되고 myLed[i]는 myLed[0], myLed[1], myLed[2]와 같이 차례로 핀 모드 설정이 실행됨

3. LED 여러 개 깜박이기

- [스케치 설명]

```
for (int i = 0; i < 3; i++) {
  digitalWrite(myLed[i], HIGH);
  delay(500);
  digitalWrite(myLed[i], LOW);
  delay(500);
}
```

  - 배열 순서에 따라 깜박이는 것 반복

3. LED 여러 개 깜박이기

[연습문제 1]

LED의 핀번호와 깜박이는 시간을 배열과 for 문을 이용하여 스케치를 작성하시오.

- int myLed[] = {2, 4, 6}; //LED 핀 번호
- int ledDelay[] = {500, 300, 500}; //delay 시간

3. LED 여러 개 깜박이기

[스케치]

```
int myLed[] = {2, 4, 6}; // LED 번호를 배열로 지정
int ledDelay[] = {500, 300, 500};
void setup() {
  for (int i = 0; i < 3; i++) {
  pinMode(myLed[i], OUTPUT); //pinMode 설정
 }
}
void loop() {
  for (int i = 0; i < 3; i++) {
  digitalWrite(myLed[i], HIGH);
  delay(ledDelay[i]);
  digitalWrite(myLed[i], LOW);
  delay(ledDelay[i]);
 }
}
```

3. LED 여러 개 깜박이기

2. 시리얼 모니터로 LED 제어

- 시리얼 통신은 아두이노와 컴퓨터 사이에 통신할 때 사용하는 방법
- 모든 아두이노 보드는 시리얼 포트(UART)가 최소 한 개 이상 있음
- 시리얼 통신을 하면 아두이노의 디지털 핀 '0'번은 RX, '1'번은 'TX'를 담당
- 시리얼 통신을 위해서 시리얼 설정을 먼저 함
- Seral.begin(9600);
  - 시리얼 통신을 시작한다는 의미
  - 9600은 통신 속도이며 단위는 bps
  - 통신 속도를 다르게 할 수 있으며 이번 예제는 9600bps

3. LED 여러 개 깜박이기

2. 시리얼 모니터로 LED 제어

- 시리얼 모니터에 숫자 '1'을 입력하면 LED가 켜지고 숫자 '0'을 입력하면 LED가 꺼지는 스케치를 작성해보자.

```
void setup() {
Serial.begin(9600);
  pinMode(13, OUTPUT);
}
int i;
void loop() {
  if (Serial.available() > 0) {
  i = Serial.read();
  Serial.println(i);
  if (i == '1') digitalWrite(13, HIGH);
  if (i == '0') digitalWrite(13, LOW);
 }
}
```

3. LED 여러 개 깜박이기

[코드 설명]

- Serial.begin(9600); Serial.begin(속도);
  - 시리얼 통신을 선언하고 속도를 설정
- if (Serial.available() > 0){ }
  - if(조건문)은 조건을 확인하기 위해 사용
  - 조건문을 실행시키는 블록을 중괄호로 감쌈
- Serial.read();
  - 시리얼 케이블로 들어오는 시리얼 데이터를 읽음
- if (i == '1') digitalWrite(13, HIGH);
  - 입력 값이 '1'이면 뒤에 나오는 코드를 실행

3. LED 여러 개 깜박이기

[코드 설명]

- Serial.println(i);
  - 입력 값을 확인하기 위해 시리얼 모니터에 값을 나타내게 하는 명령어
  - 시리얼 모니터를 열고 값을 1 또는 0을 입력
  - 아두이노의 LED가 깜박이는지 확인

3. LED 여러 개 깜박이기

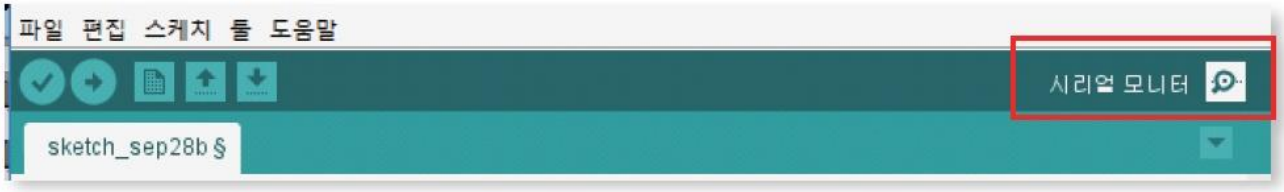

그림 2.22 시리얼 모니터 버튼

그림 2.23 시리얼 모니터에 값 표시

4. 디지털 입력

3. 풀업 풀다운 저항

- 아두이노에서 스위치를 만들면 그림 2.24와 같음
- 버튼을 누르면 전류가 흐르고 입력 전류가 5V(HIGH)가 됨
- 그러나 스위치가 연결되지 않으면 전류가 흐르는지 아닌지 알 수 없는 상태가 되는데, 이때를 플로팅(floating)현상이라고 함
- 즉, 입력 값이 0V인지 5V인지 인식할 수 없음
- 이와 같은 플로팅 현상을 막기 위해 풀업 또는 풀다운 저항을 연결하여 안정된 입력 값을 얻을 수 있음

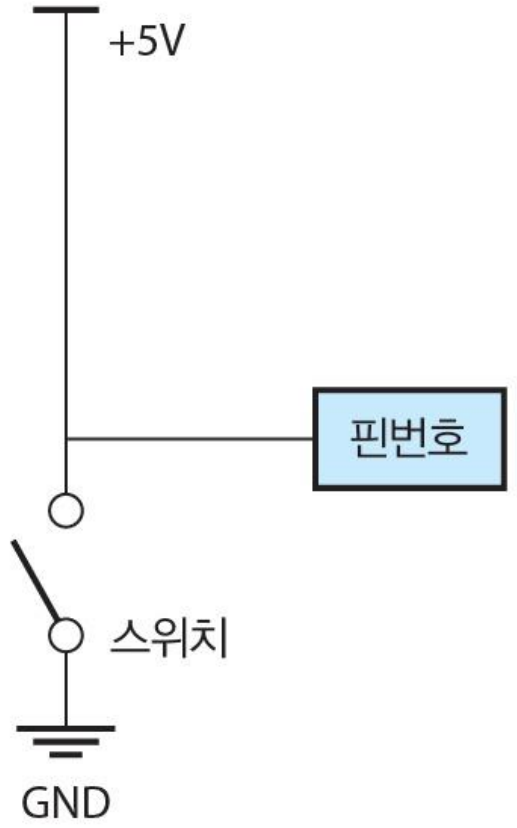

그림 2.24 버튼 연결 회로

4. 디지털 입력

- [풀업저항]
  - 저항이 5V쪽에 연결된 상태
  - 스위치가 열린 상태 : 입력 핀에 5V의 전류가 흐르고 입력 값은 HIGH가 됨
  - 스위치가 닫힌 상태 : 입력 핀에 전류가 흐르지 않는 상태가 되어 0V 전압, 입력 값은 LOW

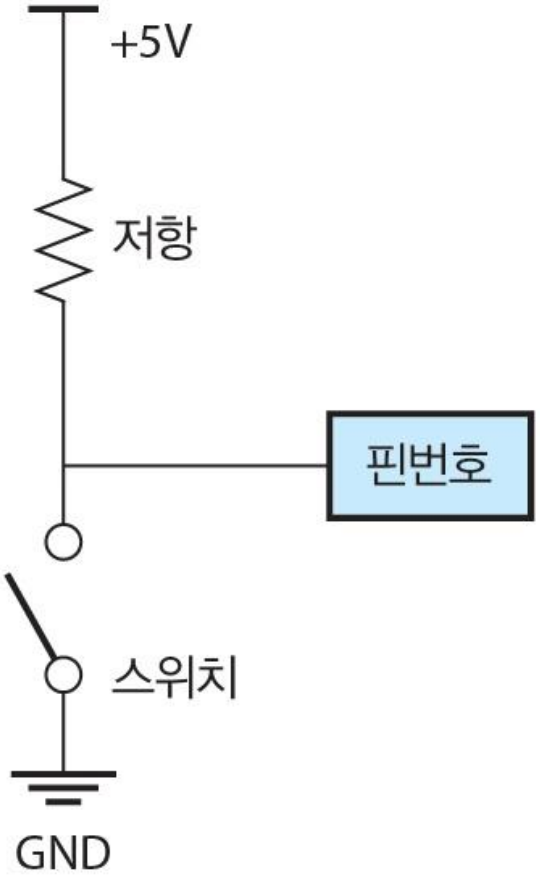

그림 2.25 풀업 스위치 회로

4. 디지털 입력

- [풀다운저항]
  - 풀다운 저항은 저항이 GND 쪽에 연결
  - 스위치가 열린 상태 : 입력핀에 0V 전압이 걸리게 되어 입력 값은 LOW
  - 스위치가 닫힌 상태 : 입력 핀에 5V 전압이 걸리고, 입력 값은 HIGH

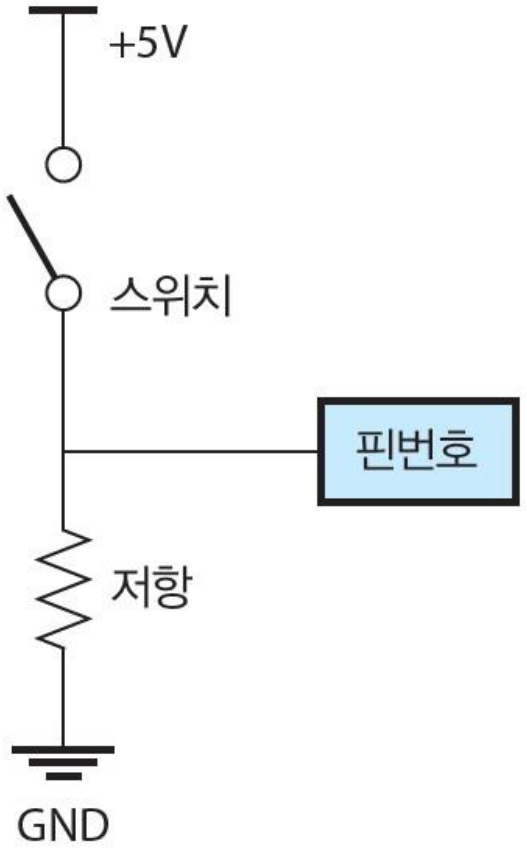

그림 2.26 풀다운 스위치 회로

4. 디지털 입력

- 풀업/풀다운 저항

표 2.5 각 저항의 스위치 입력 값

풀업/풀다운 저항	스위치 닫힘(ON)	스위치 열림(OFF)
사용 안 함	5V	플로팅
풀업(PULL-UP) 저항 값	0V	5V
풀다운(PULL-DOWN) 저항 값	5V	0V

4. 디지털 입력

[풀다운 저항 스위치 만들기]

- 누르면 스위치가 연결되는 푸시버튼 사용
- 준비물 : 푸시버튼, 저항 10KΩ(갈, 검, 주), 아두이노, 브레드보드, 케이블, 점퍼선

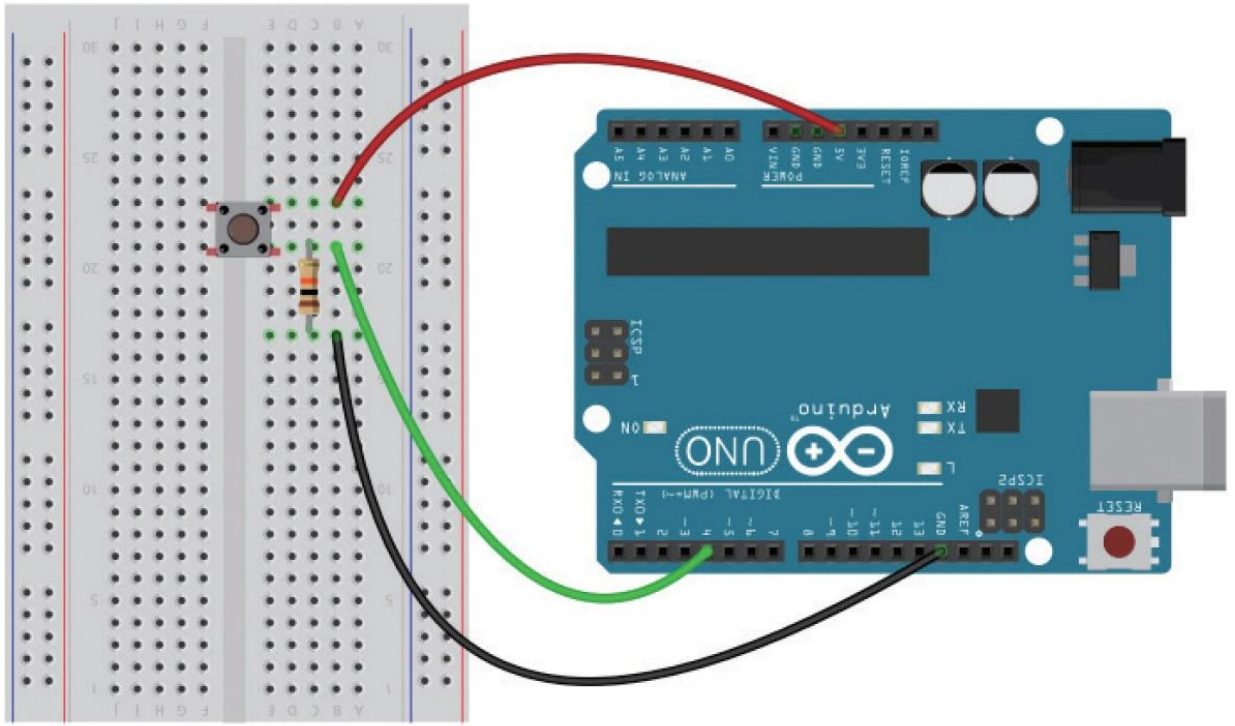

그림 2.27 풀다운 스위치 연결 모습

4. 디지털 입력

[풀업 스위치 회로]

- 스위치가 열린 상태에서 핀에 5V 가 전달되는 회로
- 스위치가 열린 상태에서 '1'이 전달

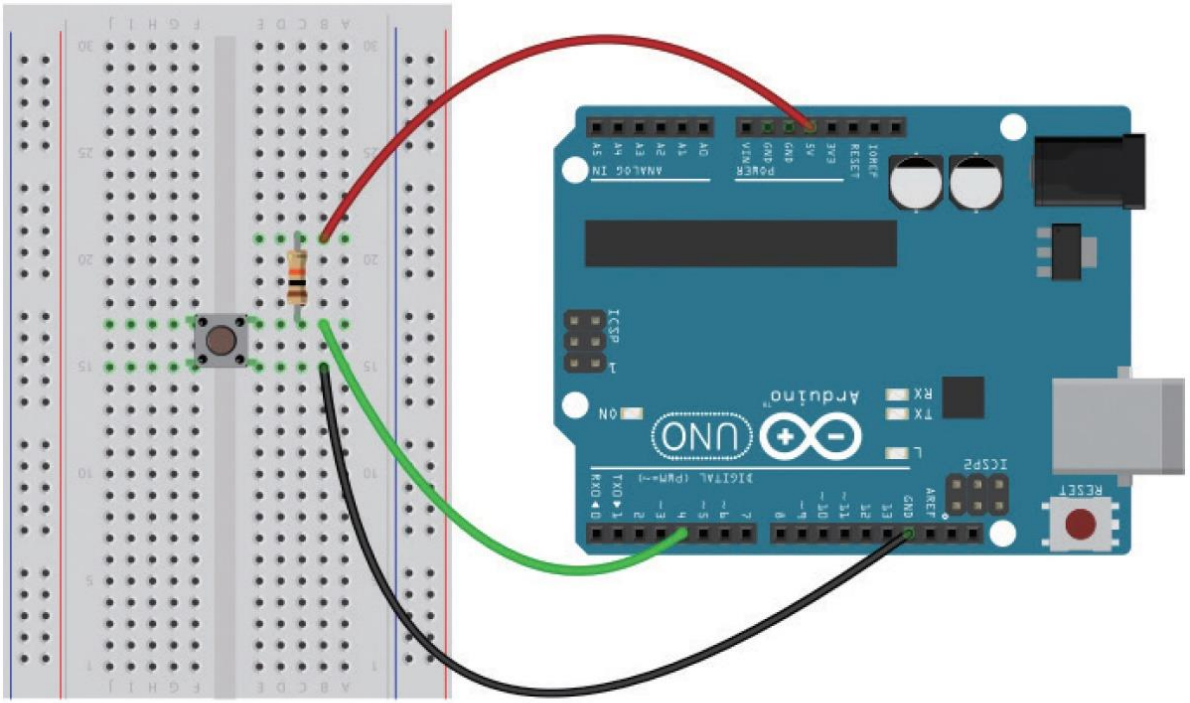

그림 2.28 풀업 스위치 연결 모습

4. 디지털 입력

2) 스케치 작성하기(풀다운 스위치)

- 이번 예제는 풀다운 스위치에 적용
- 버튼을 누를 때 입력 값을 시리얼 모니터에서 확인하는 스케치

```
int inPin = 4; // pushbutton은 디지털 핀 4에 연결됨
int val = 0;   // 읽은 값을 저장할 변수
void setup() {
  Serial.begin(9600);
  pinMode(inPin, INPUT); // 디지털 7을 입력으로 설정
}
void loop(){
val=digitalRead(inPin);
Serial.println(val);
delay(100);
}
```

4. 디지털 입력

- 스위치 테스트 결과

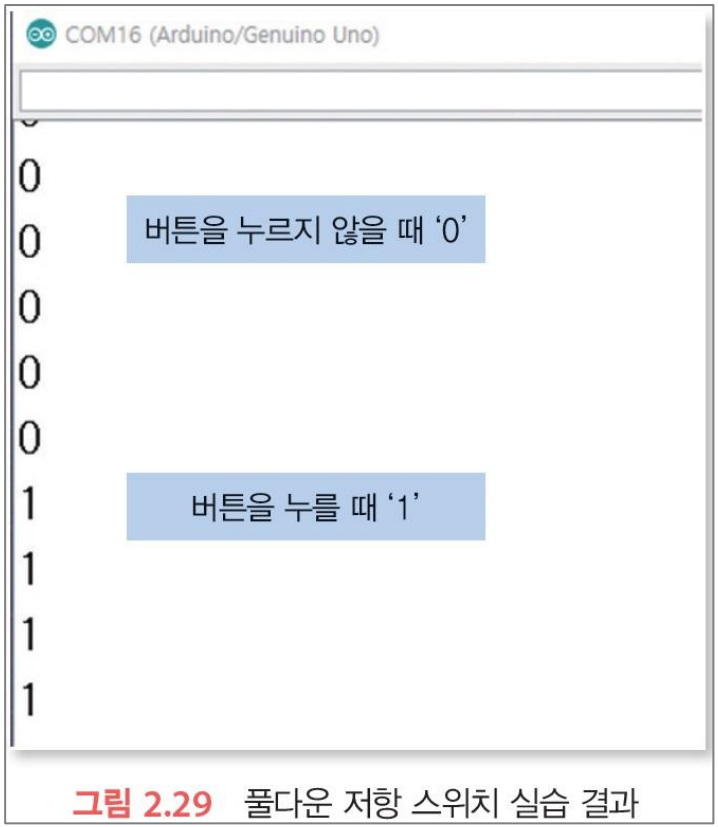

그림 2.29 풀다운 저항 스위치 실습 결과

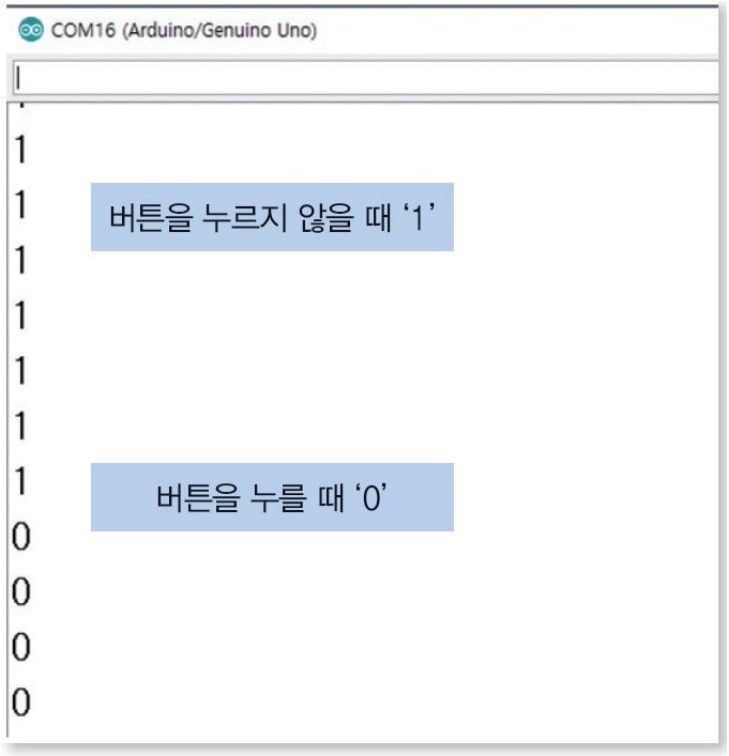

그림 2.30 풀업 저항 스위치 실습 결과

4. 디지털 입력

- [실습예제] 버튼을 이용하여 LED 제어하기
- [회로 만들기]
- 풀다운 회로로 만들고 LED는 13번에 연결

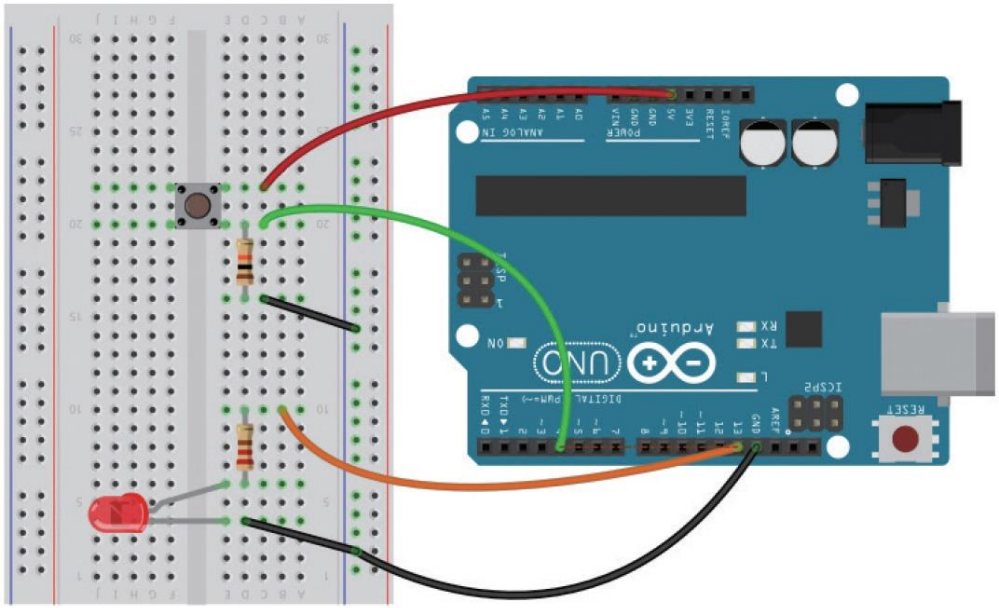

그림 2.31 버튼과 LED를 연결한 회로

4. 디지털 입력

- [실습예제] 버튼을 이용하여 LED 제어하기
- [스케치 작성하기]

```
int buttonPin = 4; // 푸시버튼이 연결된 번호
int ledPin = 13;    // LED가 사용하는 핀 번호
int buttonState = 0; // 입력핀의 상태를 저장
void setup() {
  pinMode(ledPin, OUTPUT);    // LED는 출력으로 설정
  pinMode(buttonPin, INPUT);  // 푸시버튼은 입력으로 설정
}
void loop(){
  buttonState = digitalRead(buttonPin);  //입력 값을 읽고 저장
  // 버튼이 눌렸는지 확인, 버튼이 눌렸으면 입력핀의 상태는 HIGH가 됨
   if (buttonState == HIGH) {
     digitalWrite(ledPin, HIGH); //LED 켬
   }
   else {
     digitalWrite(ledPin, LOW); //LED 끔
   }
}
```

요약

- 아날로그 신호는 소리, 빛, 음파 등 자연의 신호이고 시간에 따라 연속적으로 표현
- 디지털 신호는 컴퓨터와 스마트폰과 같은 디지털 기기에서 사용되는 신호
- 디지털 신호는 아날로그 신호를 시간에 따라 숫자로 표현한 것, 비연속인 데이터
- 저항은 전류의 흐름을 방해하는 부품
- 저항의 크기는 저항 표면의 색상 띠로 알 수 있다.
- 아두이노는 아두이노 프로그램으로 스케치를 작성한다.
- 작성한 스케치는 컴파일을 하고 업로드 한다.
- 변수는 어떤 값을 저장하는 저장 공간이고 이름을 지정한다.
- 변수를 사용하기 위해서는 변수 선언을 해야 한다.
- 배열은 같은 여러 개의 데이터를 하나의 변수에 담아서 저장하는 공간이다.
- 시리얼 통신은 아두이노와 컴퓨터 사이에 통신할 사용하는 방법이다.
- 스위치는 저항의 위치에 따라 풀업 저항 스위치와 풀다운 저항 스위치
- 풀업 저항 스위치는 버튼을 누르지 않을 때 HIGH가 된다.

수고하셨습니다^^