사물인터넷

Chapter 3.
아날로그 데이터 제어

1. 아날로그 데이터 처리
2. 아날로그 데이터 입력/출력
3. 디지털 및 아날로그 데이터 처리
4. 가변저항기를 이용한 블링크 속도 제어
5. 온도센서(CDS)를 이용하여 조도 측정

학습목표

- 아두이노에서 아날로그 데이터를 입력 받을 수 있다.
- 조도센서를 이용하여 방 안의 밝기 값을 읽을 수 있다.
- 조도에 따라 LED를 제어할 수 있다.
- LED 밝기를 아날로그 출력으로 설정할 수 있다.
- 가변저항기를 이용하여 LED의 밝기를 제어할 수 있다.

1. 아날로그 데이터 출력

- 아두이노에서 아날로그 데이터 출력이 가능
- 아날로그 신호의 출력은 자연의 소리나 빛과 같이 데이터가 연속적으로 연결되는 신호
- 아두이노에서는 PWM(Pulse Width Modulation)을 이용하여 값을 제어
- 아두이노에서 출력할 수 있는 값은 0~255 까지 총 256가지의 값을 출력 할 수 있음

1. 아날로그 데이터 출력

- 디지털 신호와 아날로그 신호

그림 3.1 디지털 신호와 아날로그 신호

1. 아날로그 데이터 출력

- 아두이노의 PWM 주기

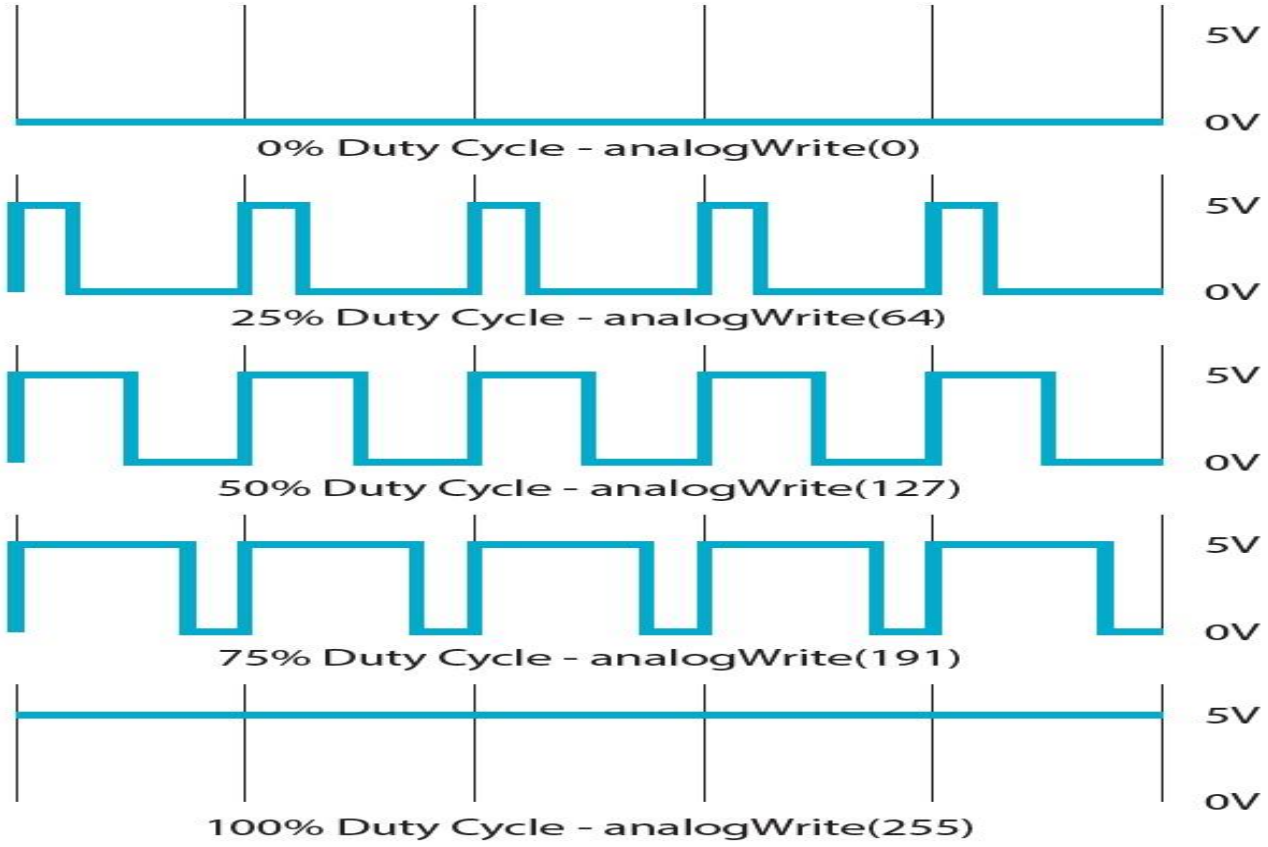


그림 3.2 아두이노의 PWM 주기

1. 아날로그 데이터 출력

- 아두이노에서 PWM을 사용할 수 있는 번호 :
  ~3, ~5, ~6, ~9, ~10, ~11번 핀

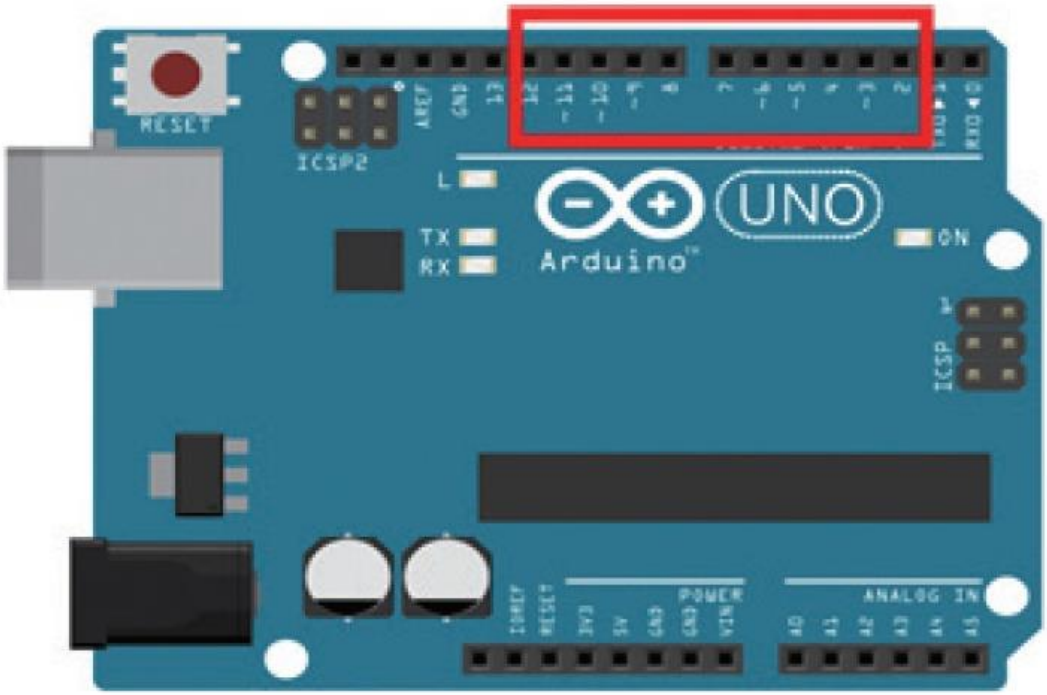


그림 3.3 아두이노 PWM 핀

2. LED 밝기 제어하기 실습

- LED 두 개를 연결하여 밝기를 서로 다르게 제어하는 실습
- 필요한 부품 : LED 2개, 저항 2개, 점퍼선 여러 개, 브레드보드, 아두이노, 케이블

1) 회로 만들기

- LED의 양극(긴 다리)를 PWM 출력이 가능한 5번, 11번에 연결

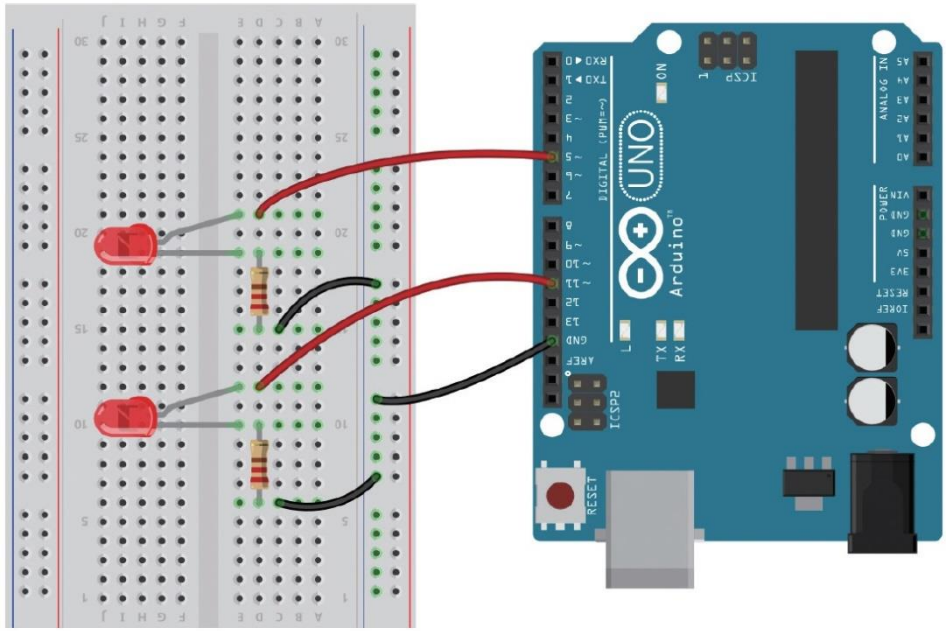


그림 3.4 아두이노와 두 개의 LED의 연결

2. LED 밝기 제어하기 실습

2) 스케치 작성하기

```
int led1 = 5;   // LED를 PWM 5번에 연결
int led2 = 11; // LED를 PWM 11번에 연결
void setup() {
  pinMode(led1, OUTPUT); // 핀을 출력으로 설정
  pinMode(led2, OUTPUT); // 핀을 출력으로 설정
}
void loop() {
  analogWrite(led1, 50);   // analogWrite 값은 0부터 255까지
  analogWrite(led2, 255); // analogWrite 값은 0부터 255까지
}
```

2. LED 밝기 제어하기 실습

[코드 설명]

- int led1 = 5;
  - LED의 핀번호 지정하기
- analogWrite(led1, 10);
  - analogWrite(핀번호, 값);
  - 아날로그 신호 전달
  - PWM 출력 형태, 값은 0~255까지 입력 가능
  - 255인 경우 최대 밝기
  - 아날로그 값은 단계별로 제어가 가능

2. LED 밝기 제어하기 실습

3) 컴파일 하기

- 컴파일 하고 에러를 확인 후 업로드

그림 3.5 컴파일 버튼과 업로드 버튼

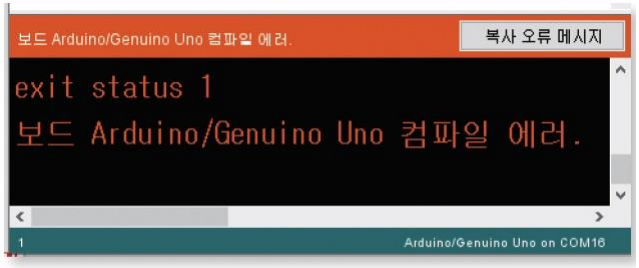


그림 3.6 컴파일 에러 메시지

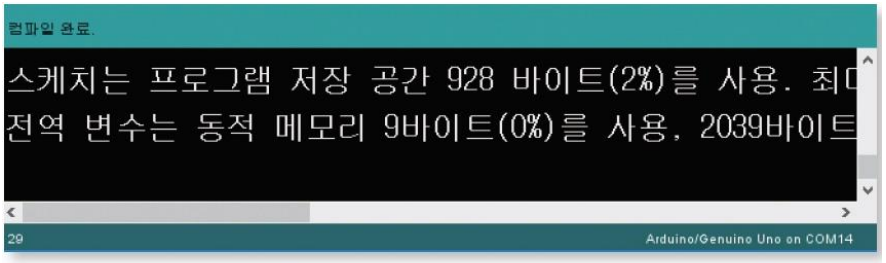


그림 3.7 컴파일 완료된 화면

2. LED 밝기 제어하기 실습

3) 업로드하기

- 포트와 보드 확인 후 업로드

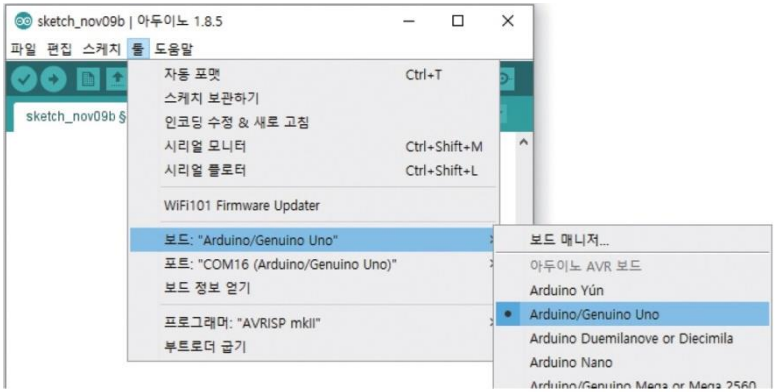


그림 3.8 보드 확인

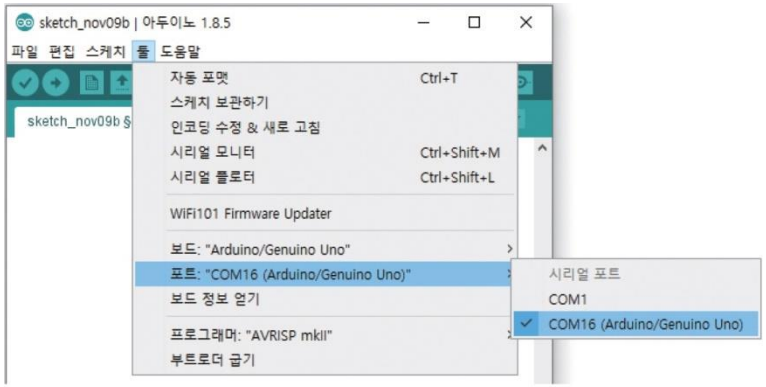


그림 3.9 포트 확인

2. LED 밝기 제어하기 실습

5) 테스트하기

- 스케치 업로드가 완료되면 각 LED의 밝기를 확인
- 5번 핀에 연결된 LED가 11번 핀에 연결된 LED 보다 조금 더 어두움
- analogWrite(핀번호, 값); 함수에서 값을 변경하여 원하는 밝기의 PWM 출력을 실행할 수 있음

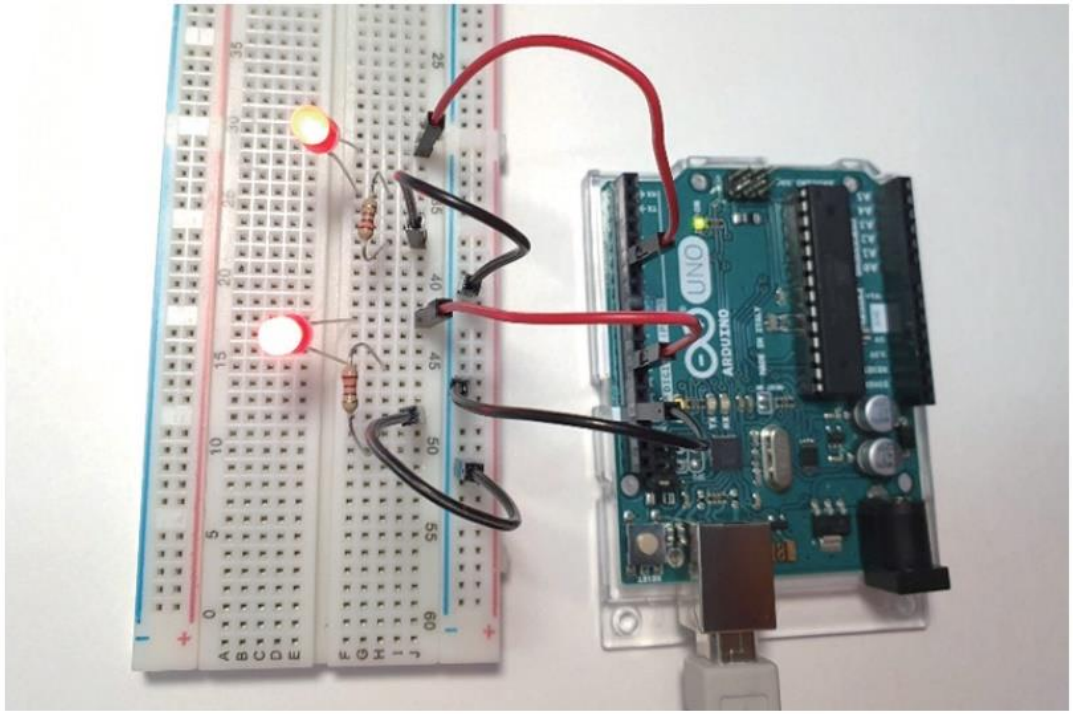


그림 3.11 아날로그 출력 테스트

3. [아날로그 출력] 부저로 음악 연주하기

[부저로 음악 연주하기]

- 준비물 : 아두이노, 브레드보드, 수동 부저,

능동 부저

수동 부저

- 능동부저: 내장된 회로가 있어 '삐'소리만 남
- 수동부저: tone() 함수로 주파수 제어 가능

그림 3.12 부저 종류

3. [아날로그 출력] 부저로 음악 연주하기

- (1) 회로 만들기

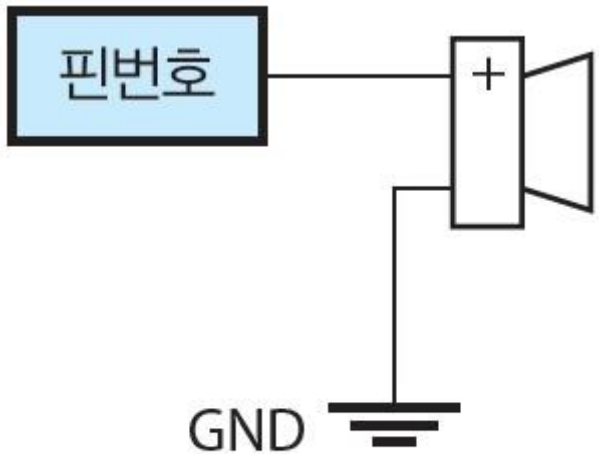


그림 3.13 부저 회로

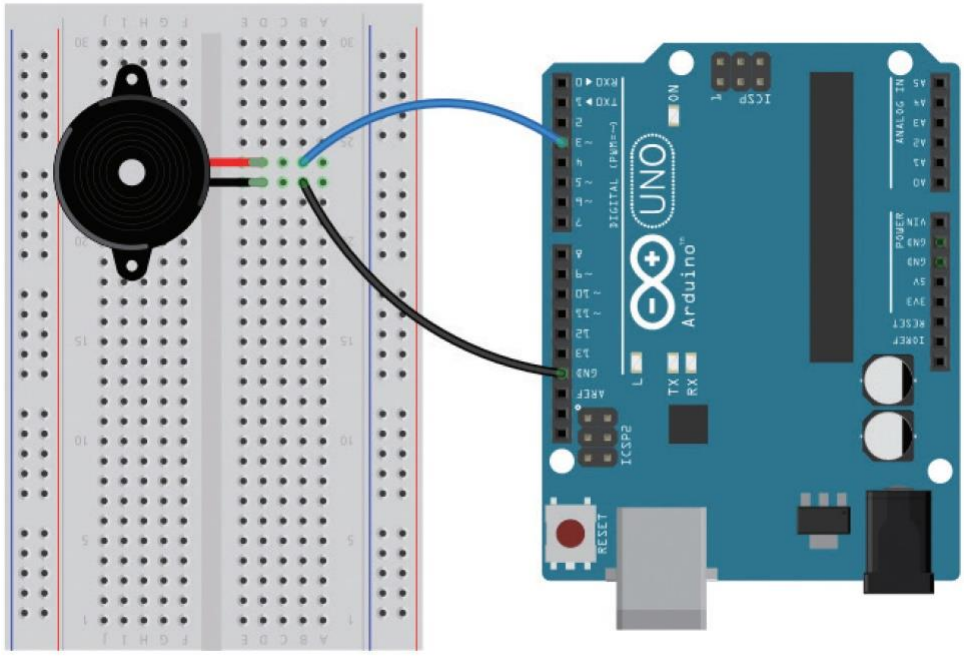


그림 3.14 부저를 아두이노에 연결한 모습

3. [아날로그 출력] 부저로 음악 연주하기

- (2) 스케치 작성하기

```
// 주파수 이용해서 부저 소리내기
void setup(){
  tone(3, 523, 500);     // 523 주파수로 1초간 소리 냄
  delay(500);            // 소리가 끝날 때 까지 기다림
}
void loop(){             // 반복 구간, 빈 부분을 그대로 둔다.
}
```

3. [아날로그 출력] 부저로 음악 연주하기

[코드 설명]

- tone(3, 523, 500);
  - tone(핀번호, 주파수, 지속시간);
  - 핀번호는 아두이노에 연결하는 핀
- 주파수(frequency)
  - 소리의 주파수이면 단위는 헤르츠(Hertz)
  - 데이터 유형은 unsigned int를 사용
  - 지속시간은 밀리초 단위로 소리의 지속시간
  - 데이터는 unsigned long
- void loop(){ }
  - 소리를 계속 내지 않고 한번만 내므로 반복문 구간에는 코드를 넣지 않음
  - 반복문 구간에 코드가 없더라도 void loop() 함수는 적어야 함
  - 이 함수가 없으면 에러가 발생

3. [아날로그 출력] 부저로 음악 연주하기

- 피아노 건반 주파수

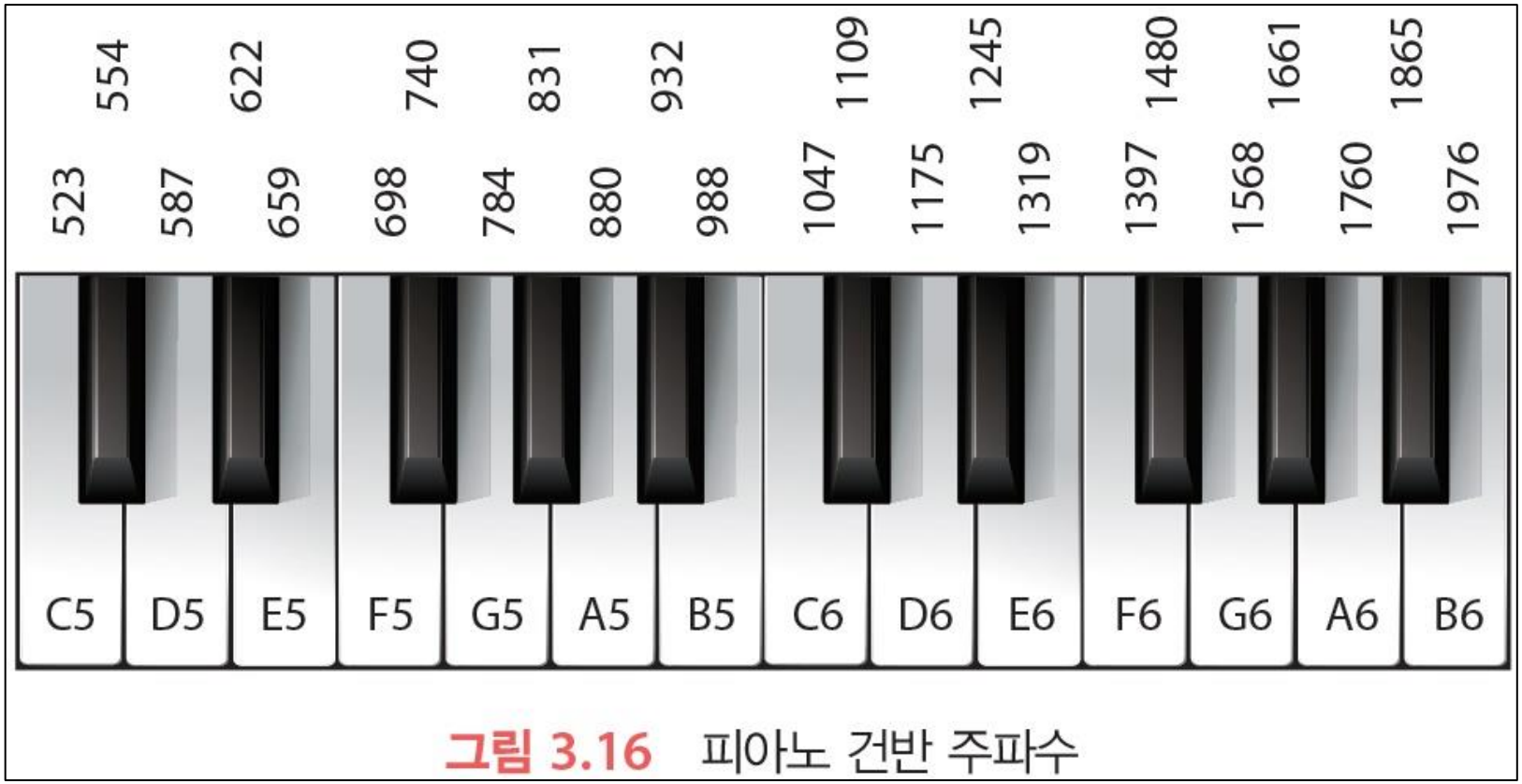


그림 3.16 피아노 건반 주파수

3. [아날로그 출력] 부저로 음악 연주하기

- "도, 레, 미, 파"를 주파수를 이용하여 음을 만들어 보자.
- 다음은 음을 만드는 예제이다.

음 이름	도(C5)	레(D5)	미(E5)	파(F5)
주파수	523	587	659	698

```
// toneDoReMi
void setup(){
  tone(3, 523, 1000); delay(1000);
  tone(3, 587, 1000); delay(1000);
  tone(3, 659, 1000); delay(1000);
  tone(3, 698, 1000); delay(1000);
}
void loop(){
}
```

3. [아날로그 출력] 부저로 음악 연주하기

[코드설명]

- tone(3, 523, 1000);
  - tone(핀번호, 주파수, 지속시간);
- delay(1000);
  - delay(지속시간);
  - 다음 코드가 진행되기 전까지 코드를 지속시간 만큼 지연 시킴
  - 지속시간이 tone() 함수의 지속시간보다 짧으면 소리가 제대로 나지 않는 경우가 있음
  - delay() 지속시간이 1000보다 길면 음과 음 사이에 소리가 잠시 끊어지는 현상이 일어남

3. [아날로그 출력] 부저로 음악 연주하기

- [실습] 주파수 이용하여 노래 만들기
- 권태호 작곡의 "봄 나들이"
- 계명: "솔미솔미솔라솔"

음 이름	솔(G5)	미(E5)	솔(G5)	미(E5)	솔(G5)	라(A5)	솔(G5)
주파수	784	659	784	659	784	880	784
음 길이(박자)	250	250	250	250	250	250	500

3. [아날로그 출력] 부저로 음악 연주하기

- (1) 스케치 작성하기

```
// 봄 나들이 스케치
int myNote[]={784, 659, 784, 659, 784, 880, 659};
int myDu[]={250, 250, 250, 250, 250, 250, 500};
void setup() {
  for (int i = 0; i < 7; i++) {
  tone(3, myNote[i], myDu[i]); //핀 번호 확인
  delay(myDu[i]*1.3);
  noTone(3);
  }
}
void loop() {
}
```

3. [아날로그 출력] 부저로 음악 연주하기

- [코드 설명]
- int myNote[]={784, 659, 784, 659, 784, 880, 659};
  - 소리 음계를 데이터로 가진 배열
  - 음의 데이터는 정수형으로 지정
- int myDu[]={250, 250, 250, 250, 250, 250, 500};
  - 음의 길이를 데이터로 가진 배열
  - 음을 만들 때 4분 음표는 500으로 지정하고 8분 음표의 길이를 250으로 지정함
  - 숫자는 변경 할 수 있음
- tone(3, myNote[i], myDu[i]);
  - tone(핀번호, 주파수, 음길이);
- delay(myDu[i]*1.3);
  - delay(지연시간*1.3);
  - 지연시간을 1.3만큼 곱해 줌
  - 음의 연결 부분에 간격을 주어 음이 또렷하게 들리게 하기 위함
  - 1.3을 곱하는 것을 하지 않고 다른 수를 곱해도 됨

3. [아날로그 출력] 부저로 음악 연주하기

(2) 업로드하기

- 스케치 업로드 후 처음부터 다시 시작하려면 리셋 버튼을 누름

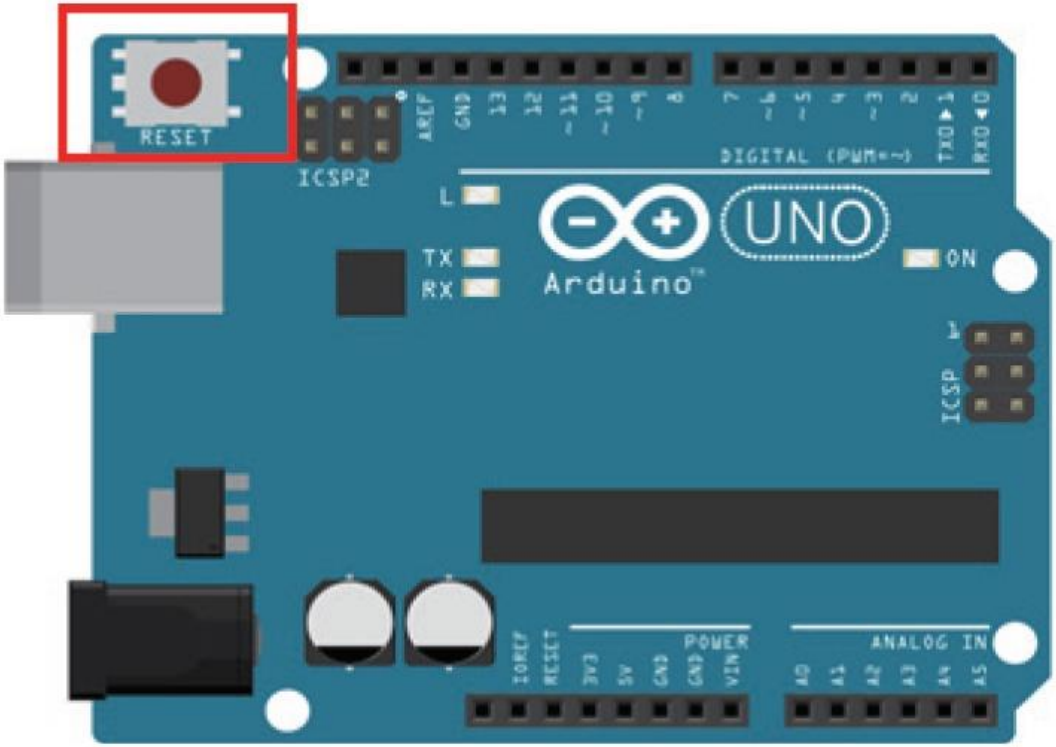
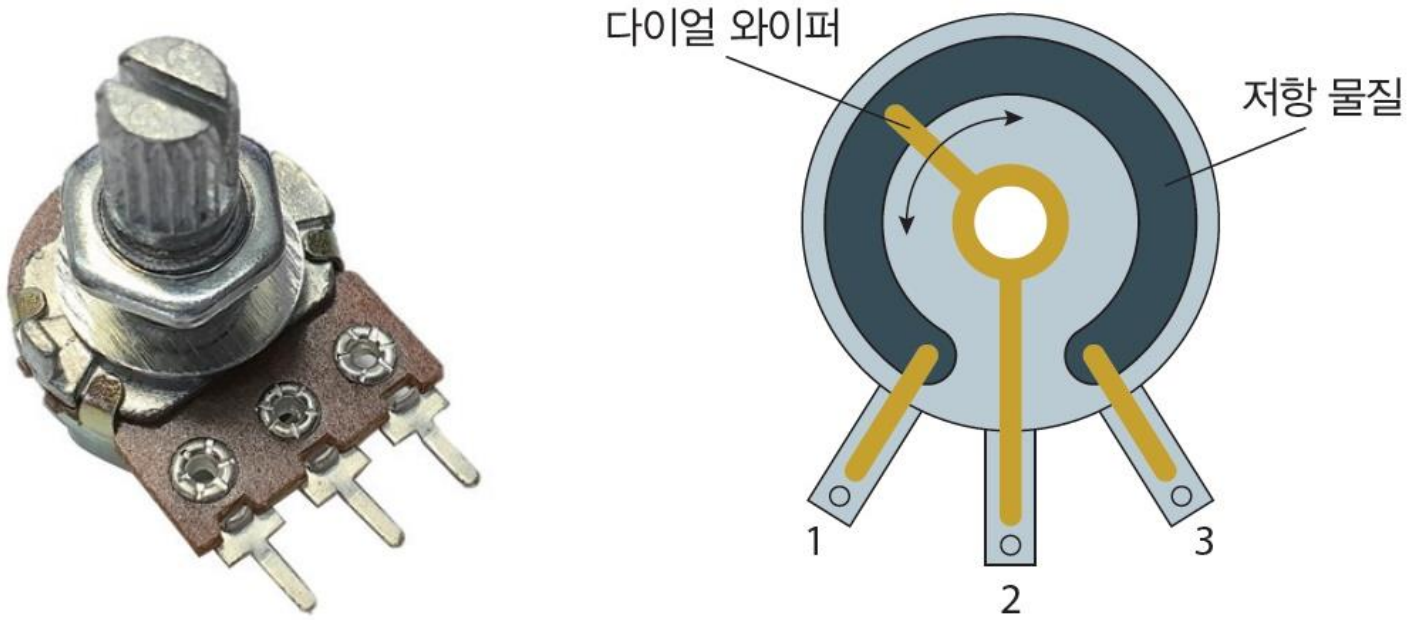


그림 3.17 아두이노 리셋 버튼

4. 아날로그 입력

- 가변 저항기 실습(아날로그 데이터)



(a) 가변저항기 (b) 가변저항기 원리

그림 3.18 가변저항기와 원리

4. 아날로그 입력

- 손잡이 돌리는 위치에 따른 전압의 변화

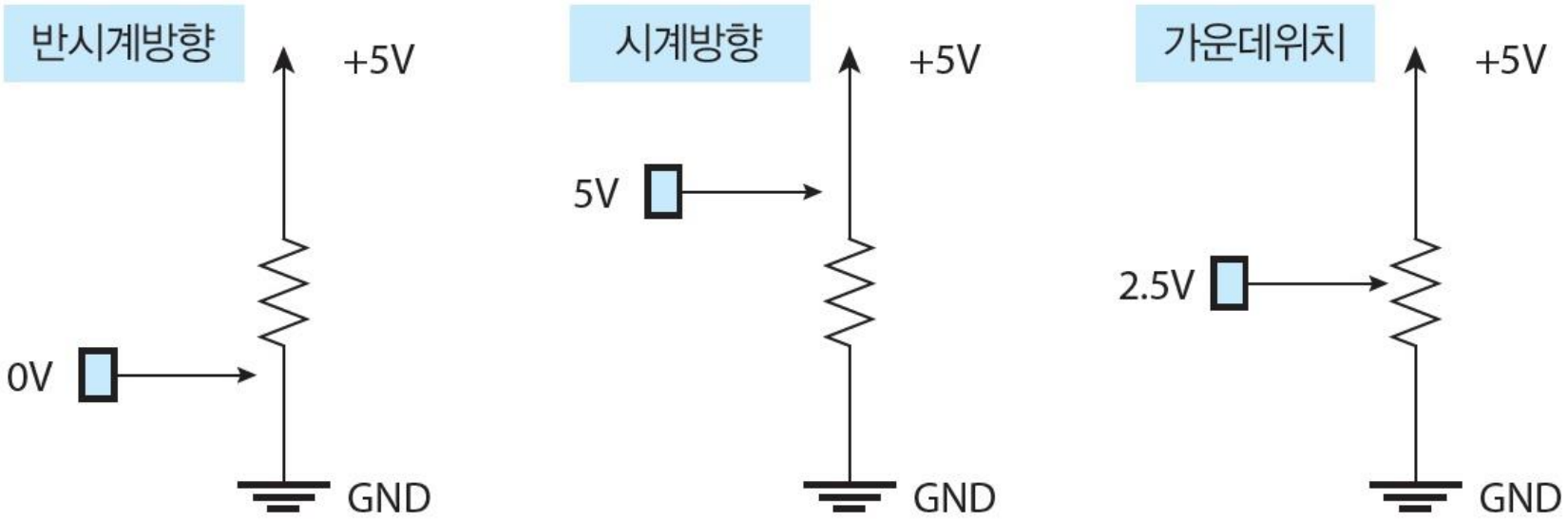


그림 3.19 손잡이의 위치에 따른 전압의 변화

4. 아날로그 입력

- 가변저항기
  - 저항물질이 둥글게 연결되어 있고 저항물질의 길이의 변화에 따라 저항 값이 변함
- 아두이노에서 아날로그 값을 읽는 함수는 analogRead()
- 입력된 전압의 크기를 10비트, 즉 1 ~1023 의 정수 값으로 대응
- 전압의 크기가 5V이면 입력 값은 1023

표 3.1 전압의 크기에 대응되는 정수 값

전압	아두이노 입력 값
0V	0
5V	1023

4. 아날로그 입력

- 아두이노에서 아날로그 입력 값을 읽을 수 있는 핀은 A0 ~ A5
- 아날로그 값을 입력 받기 위해서 ADC기능을 사용
- ADC(Analog to Digital Converter)는 아날로그 신호 값을 디지털로 변환하여 사용할 수 있게 하는 기능

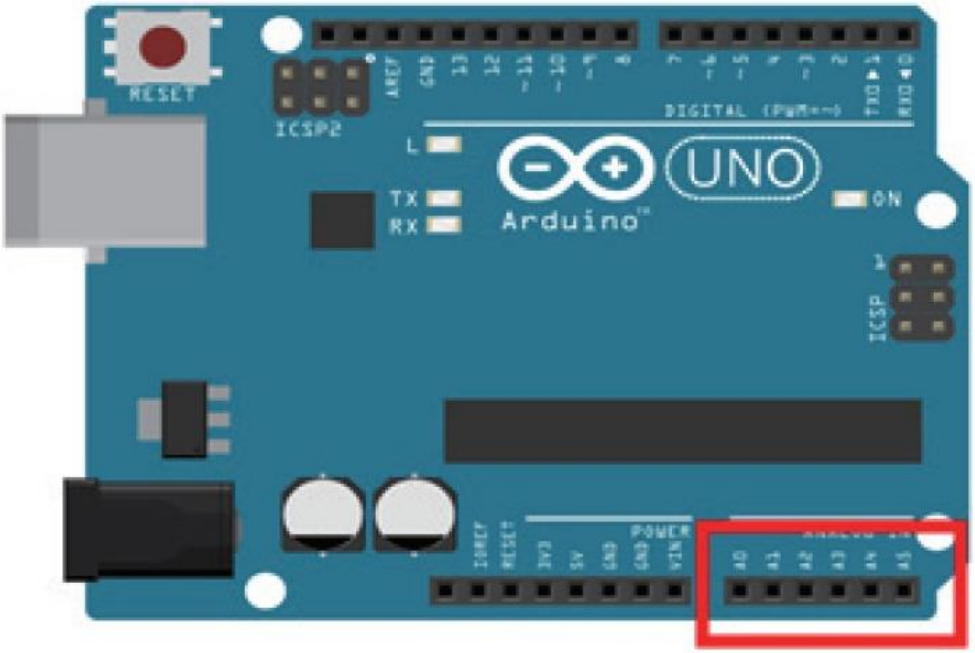


그림 3.20 아날로그 입력 핀

4. 아날로그 입력

[가변저항기 값 읽어오기]

(1) 회로 만들기

- 가변저항기는 세 개의 다리가 있음
- 가변저항기의 (+)와 (-)가 바뀌면 값을 읽을 수 있고 숫자가 커지거나 작아지는 것이 반대로 나타남

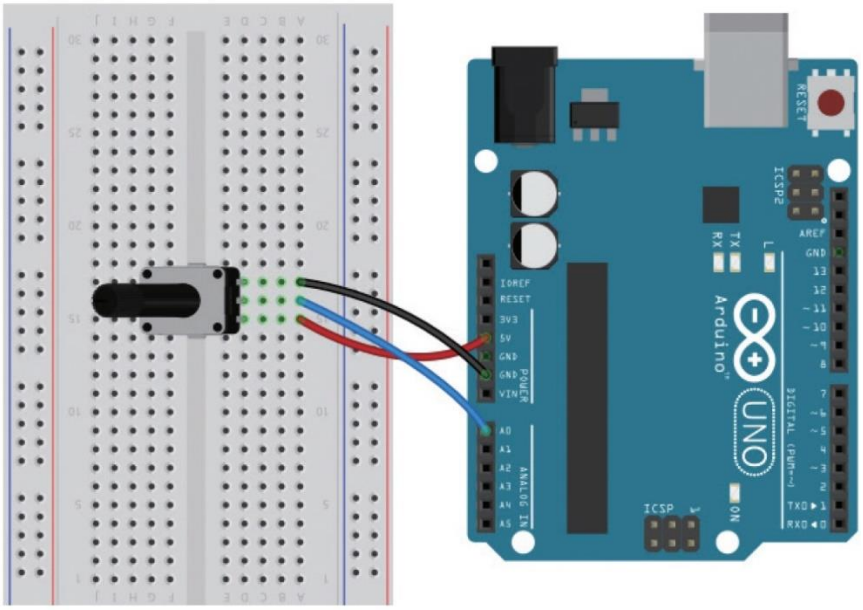


그림 3.21 가변저항기 연결 모습

- 가변저항기의 1번 다리-> 아두이노의 5V에 연결
- 3번 다리 ->아두이노의 GND에 연결
- 나머지 가운데 다리는 A0에 연결

4. 아날로그 입력

(2) 스케치 작성하기

가변저항기로 입력되는 값은 시리얼 모니터로 확인

```
// 가변저항기 테스트
void setup() {
  Serial.begin(9600);
}
void loop() {
  int a = analogRead(A0);
  Serial.println(a);
  delay(100);
}
```

4. 아날로그 입력

[코드 설명]

- Serial.begin(9600);
  - 시리얼 통신을 시작하고 통신 속도는 9600bps로 설정
- analogRead(A0);
  - analogRead(핀번호);
  - 지정한 아날로그 핀에서 입력 값을 읽음
- Serial.println(a);
  - 시리얼 모니터에서는 데이터를 ASCII 텍스트로 표시
    Serial.println()는 (ASCII 13 또는 '₩ r')을 표시하므로 다음 줄에 표시됨

4. 아날로그 입력

(3) 스케치 업로드하고 테스트

- 스케치를 업로드 한 후 가변저항기를 이용하여 시리얼 모니터에 값을 읽어보자.
- 가변저항기를 올리면서 값이 변화하는 것을 살펴보자.

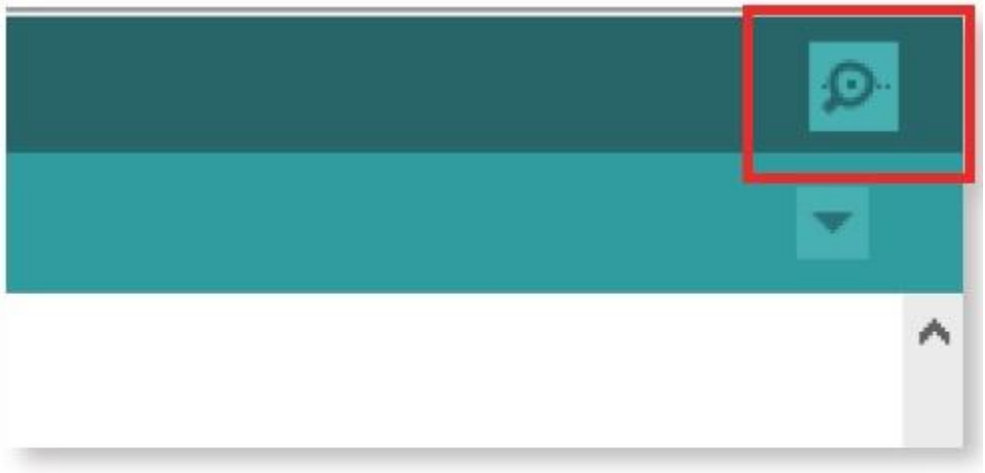


그림 3.30 시리얼 모니터 버튼

4. 아날로그 입력

- (3) 스케치 업로드하고 테스트

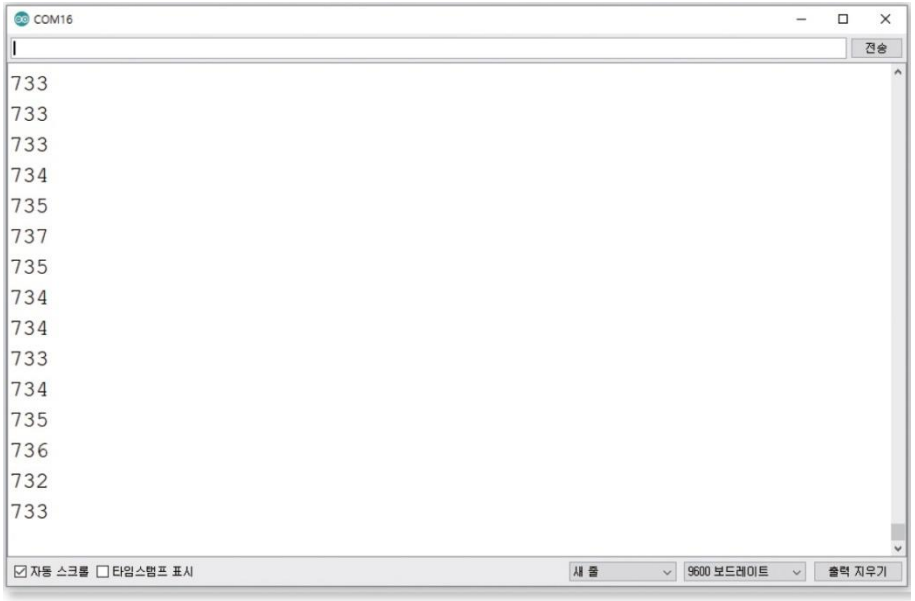


그림 3.31 조도센서 값을 시리얼 모니터에 표시

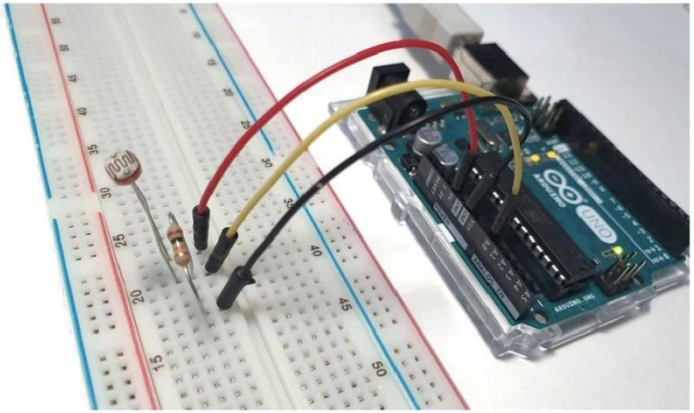


그림 3.32 조도센서 테스트 모습

4. 아날로그 입력

[가변저항기를 이용하여 LED 밝기 제어]

- 1) 회로 만들기

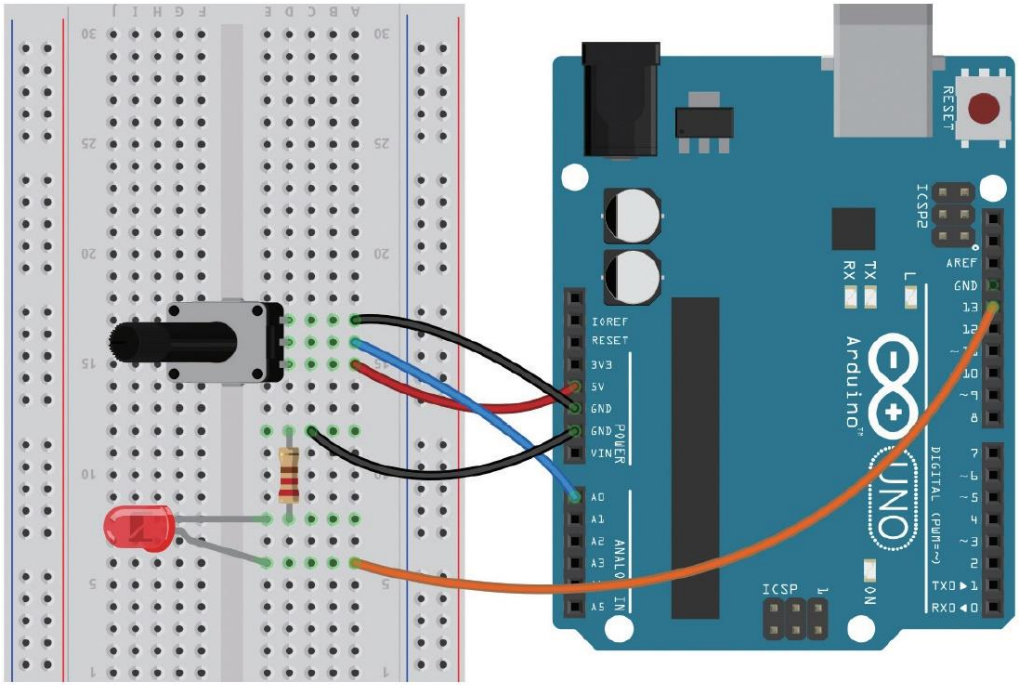


그림 3.25 가변저항기와 LED 연결 모습

4. 아날로그 입력

- (2) 스케치 작성하기

```
// 가변저항기 값을 읽고 LED로 보내기
int ledPin = 13; // LED는 13번 핀
int val = 0; // 읽은 아날로그 값을 저장하는 변수
void setup() {
  pinMode(ledPin, OUTPUT); // ledPin 핀을 OUTPUT 으로
}
void loop() {
  val = analogRead(A0); // A0에서 값을 읽음
  digitalWrite(ledPin, HIGH); // ledPin을 켬
  delay(val); // 잠시 프로그램을 중단
  digitalWrite(ledPin, LOW); // ledPin을 끔
  delay(val); // 잠시 프로그램을 중단
}
```

4. 아날로그 입력

[코드 설명]

- int ledPin = 13;
  - LED를 13번 핀에 연결
- val = analogRead(A0);
  - 변수 val에 analogRead(A0)로 읽은 값을 저장
- digitalWrite(ledPin, HIGH);
  - 디지털 값으로 'HIGH'를 입력하여 LED를 켬
- digitalWrite(ledPin, LOW);
  - 디지털 값으로 'LOW'를 입력하여 LED를 끔
- delay(val);
  - val에 저장된 값을 delay 시간 값으로 적용
  - val값에 따라 LED의 밝기 제어

4. 아날로그 입력

(3) 스케치를 업로드하고 테스트

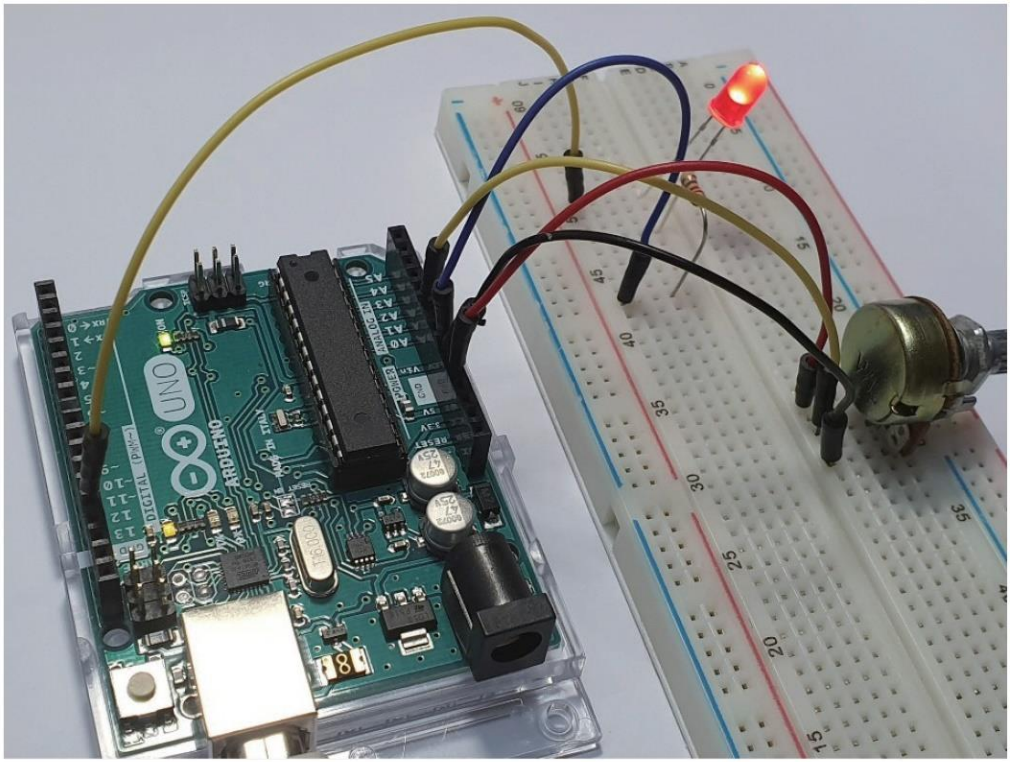
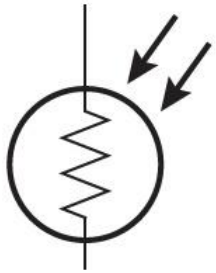


그림 3.26 가변저항기 테스트 모습

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 빛의 밝기를 측정하는 센서
- CDS 조도센서는 황화카드뮴(cadmium sulfide)이라는 의미이고 빛이 많이 들어오면 저항이 작아지고 빛이 적게 들어오면 저항이 커지는 성질
- 이 저항 성질을 이용하여 조도를 측정하는 센서로 활용

(a) CDS 조도센서



(b) 조도센서 기호

그림 3.27 조도센서

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 1) 회로 만들기

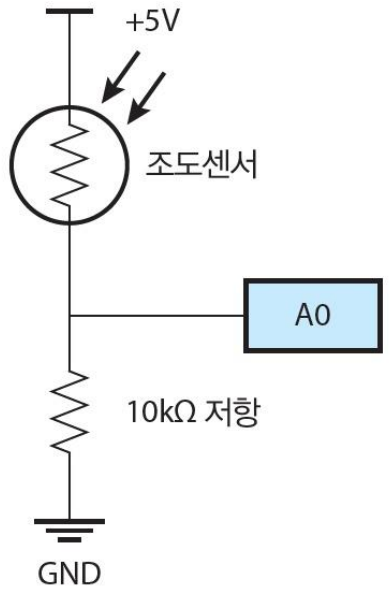


그림 3.28 조도센서 연결 회로도

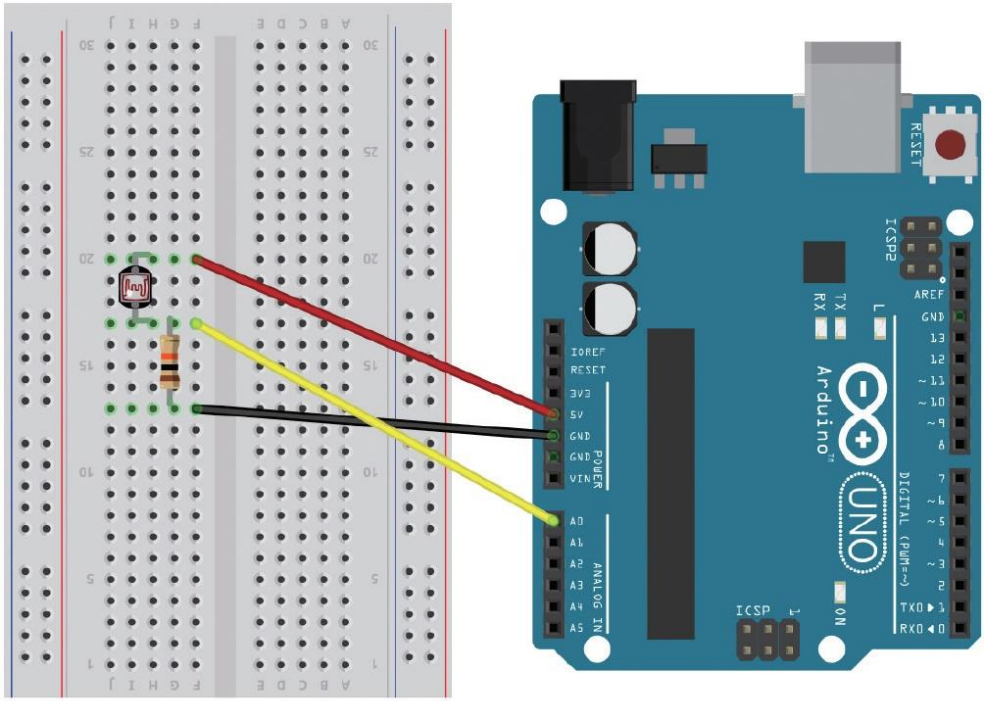


그림 3.29 조도센서와 아두이노의 연결 모습

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 2) 스케치 작성하기
  - 조도센서로 빛의 밝기를 읽을 때는 analogRead() 함수 사용
  - 시리얼 모니터에서 입력 값을 읽기 위해서 Serial.begin()을 setup()함수에 입력

```
void setup() {
  Serial.begin(9600); // 시리얼 모니터 속도를 9600으로
}
void loop() {
  int sensorValue = analogRead(A0); // 아날로그 값 읽기
  Serial.println(sensorValue); // 시리얼 포트로 출력
  delay(200); // 0.2초 동안 기다림
}
```

5. 조도센서(CDS)를 이용하여 조도 측정하기

[스케치 설명]

- Serial.begin(9600);
  - 시리얼 통신으로 데이터를 읽기 위해서 Serial.begin(9600);에서 통신 속도를 설정
- int sensorValue
  - 아날로그 입력 값을 저장하기 위해 정수형 변수를 선언
- int sensorValue = analogRead(A0);
  - A0 핀에서 읽은 값을 'sensorValue'에 저장
- Serial.println(sensorValue);
  - 입력핀에서 읽은 값을 저장한 sensorValue값을 시리얼 모니터에 표시

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 4) 테스트

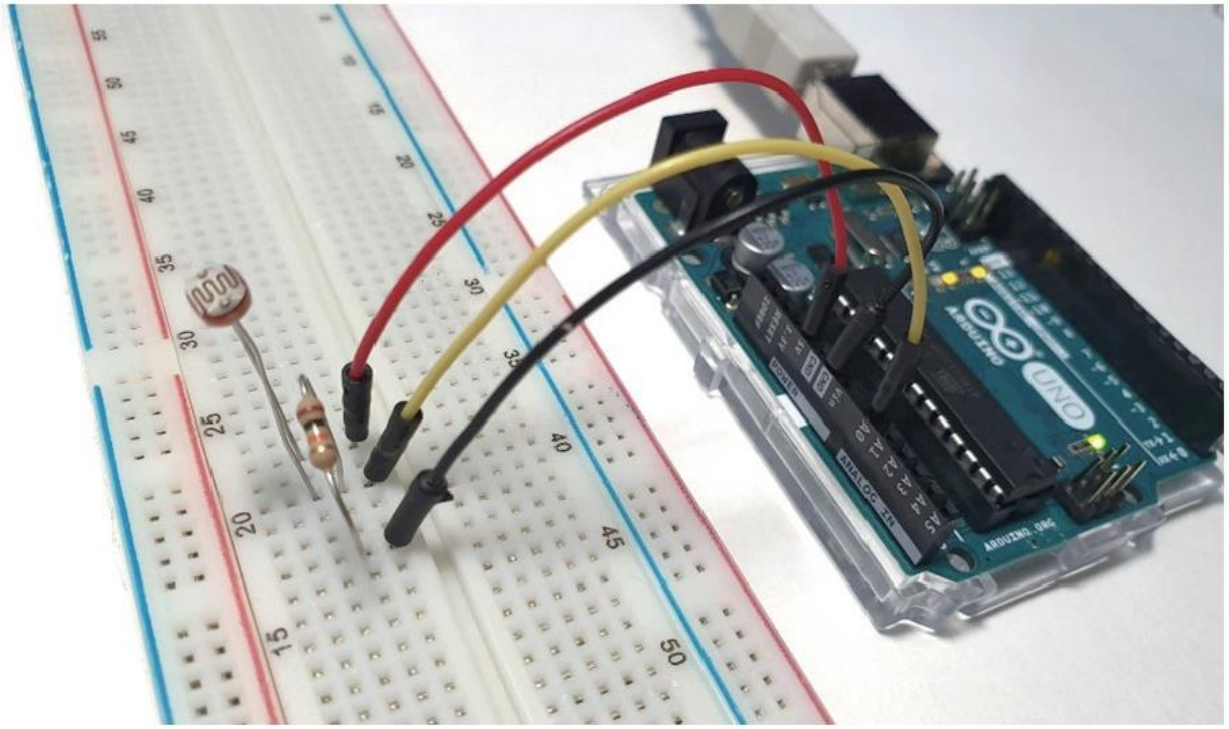


그림 3.32 조도센서 테스트 모습

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 3) 업로드와 테스트
  - 시리얼 모니터를 열고 조도 센서 값을 읽음

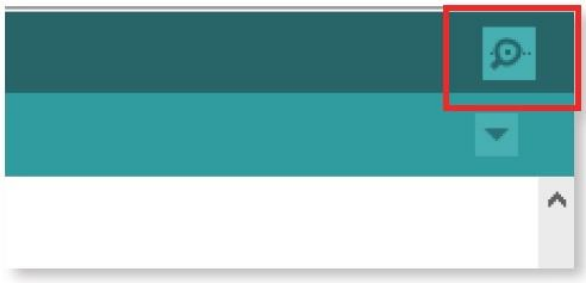


그림 3.30 시리얼 모니터 버튼

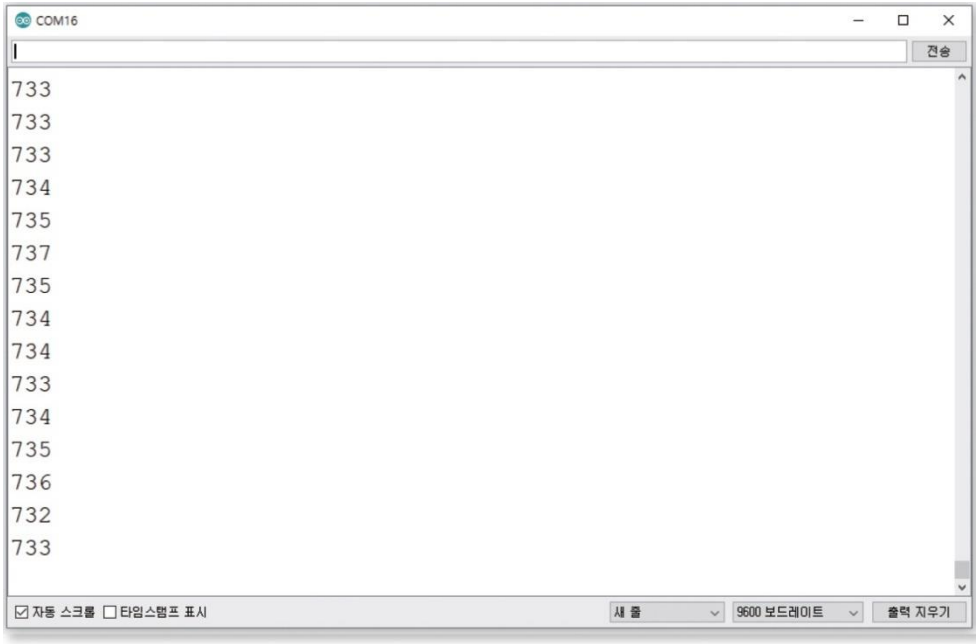


그림 3.31 조도센서 값을 시리얼 모니터에 표시

5. 조도센서(CDS)를 이용하여 조도 측정하기

[조도에 따라 LED 제어하기]
거리에 가로등을 보면 밤에는 가로등이 켜지고 밤에는 가로등이 꺼진다.
조도 센서를 이용하여 가로등을 만들어 보자.

- 1) 회로 만들기
  - 필요 부품 : 조도센서, 저항 10KΩ, LED, 점퍼선, 아두이노

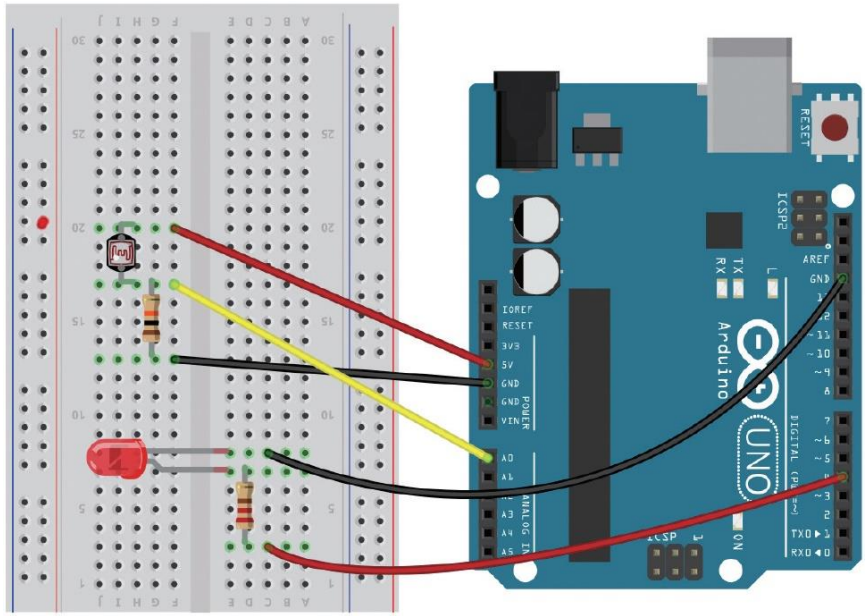


그림 3.33 조도센서와 LED 회로

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 2) 스케치 작성하기

```
void setup() {
  Serial.begin(9600); // 시리얼 모니터 속도를 9600으로
  pinMode(4, OUTPUT);
}
void loop() {
  int sensorValue = analogRead(A0); // 아날로그 값 읽기
  Serial.println(sensorValue); // 시리얼 포트로 출력
  if (sensorValue > 700) digitalWrite(4, LOW);
  else digitalWrite(4, HIGH);
  //delay(200); // 0.2초 동안 기다림
}
```

5. 조도센서(CDS)를 이용하여 조도 측정하기

[스케치 설명]

- Serial.begin(9600);
  - 시리얼 통신으로 데이터를 읽기 위해서 Serial.begin(9600);에서 통신 속도를 설정
- int sensorValue;
  - 아날로그 입력 값을 저장하기 위해 정수형 변수를 선언
- if (sensorValue > 제어값)
  - sensorValue의 값이 제어값보다 클 때 다음 명령을 실행하는 의미
  - 제어값의 숫자는 테스트 과정에서 밝을 때와 어두울 때를 측정하여 평균 값을 계산한 값으로 사용함
  - 이 값은 설계에 따라 변경 될 수 있음
- else
  - if() 값이 아닐 때 실행하는 명령어
- // delay(200);
  - //는 주석 문이기 때문에 실행되지 않음
  - 프로그램 작성 중 실행시키지 않을 부분은 '//'으로 실행을 중단시킬 수 있음

5. 조도센서(CDS)를 이용하여 조도 측정하기

- 3) 업로드하고 테스트하기

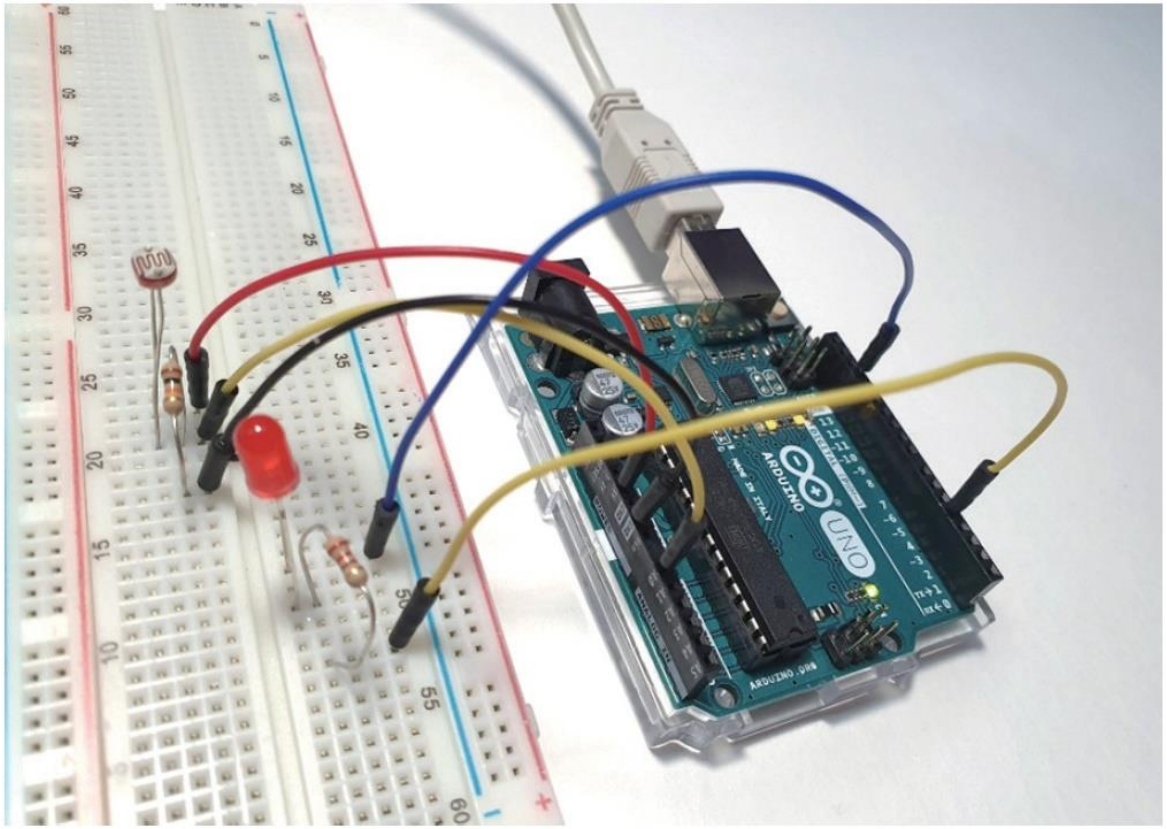


그림 3.34 조도센서와 LED 테스트 모습

요약

- 아두이노에서는 PWM 방식을 이용하여 아날로그 출력 값을 제어할 수 있다.
- LED의 밝기를 PWM 펄스폭을 조절하여 제어할 수 있다.
- LED의 밝기를 256단계로 조절할 수 있다.
- 아두이노에서 PWM을 사용할 수 있는 번호는 ~3, ~5, ~6, ~9, ~10, ~11번이다.
- 수동 부저는 내장된 회로가 없어 주파수 제어가 가능하다.
- 수동 부저를 이용하여 '도', '레', '미'와 같은 음을 만들 수 있다.
- tone() 함수를 이용하여 부저 소리를 낼 수 있다.
- tone() 함수를 void loop() 함수에 입력하면 반복해서 소리가 계속 난다.
- 아두이노의 리셋 버튼을 누르면 프로그램이 처음부터 다시 작동된다.
- 아날로그 데이터는 자연에서 발생하는 데이터이다.
- 조도 센서, 온도 센서는 아날로그 값을 읽는 센서이다.
- 가변저항기는 가운데 손잡이를 돌려 저항 값을 증가시키거나 감소시킨다.
- 아날로그 값을 읽는 함수는 analogRead()이다.
- 아두이노에서 입력된 아날로그 값을 읽으려면 시리얼 모니터를 이용한다.
- 가변저항기의 입력 값을 delay()값으로 저장하여 LED 밝기를 제어할 수 있다.
- CDS 조도센서는 빛이 많이 들어오면 저항이 작아지는 성질을 이용한 센서이다.
- CDS 조도센서와 LED를 이용하여 어두울 때 자동으로 켜지는 가로등을 만들 수 있다.