

# 1. Hello LED

[학습목표]

- 아두이노의 기본적인 사용법을 익힐 수 있다.
- LED를 켜기 위한 회로와 프로그래밍을 할 수 있다.

## 가. LED의 이해

### 1) 개요

우리 주변에는 스마트폰의 충전 상태를 나타내는 작은 LED에서부터 조명기구까지 다양한 분야에서 폭넓게 LED가 사용되고 있다. LED(Light Emitting Diode)는 한쪽 방향으로만 전류가 흐르는 특성을 가지고 있으며, 다양한 색상의 빛을 낼 수 있는 저렴한 소자이다. LED는 극성을 가지고 있는 소자로서 다리가 긴 쪽이 +이며, 브레드보드에 장착할 시 극성에 주의해서 연결을 해야 한다. 또한 LED는 최대 전류가 제한되어 있으므로 LED의 파손을 방지하기 위해서 전류의 양을 제한시켜주어야 한다. LED에 흐르는 전류를 제한시켜 주기 위해서는 옴의 법칙을 이용하여 저항값을 계산 한 후 회로에 적절한 저항을 추가시켜 주어야 한다. 사용하는 LED의 데이터시트를 참고하여 저항값을 계산해야 하며, 이번 작업에서는 2V의 순전압과 20mA의 전류량을 가지는 LED를 기준으로 하여 저항값을 계산한다. 이 값은 사용하는 LED 별로 다를 수 있으므로 데이터 시트를 참고하면 정확하게 저항을 계산할 수 있다. 저항의 계산은 간단한 옴의 법칙을 이용하여 다음과 같이 알 수 있다.

$$R = \frac{V_s - V_f}{I}$$

5V 아두이노를 사용하였을 경우 위 수식은  $\frac{5-2}{0.02} = 150\Omega$ 이 되며, 계산한 저항보다는 큰 저항을 사용하는 것이 안전하다. 다만 너무 높은 저항을 사용하게 되면, LED의 밝기는 매우 약하게 될 것이다. LED는 극성이 있으므로 다리가 긴쪽을 +에 연결하여 준다.

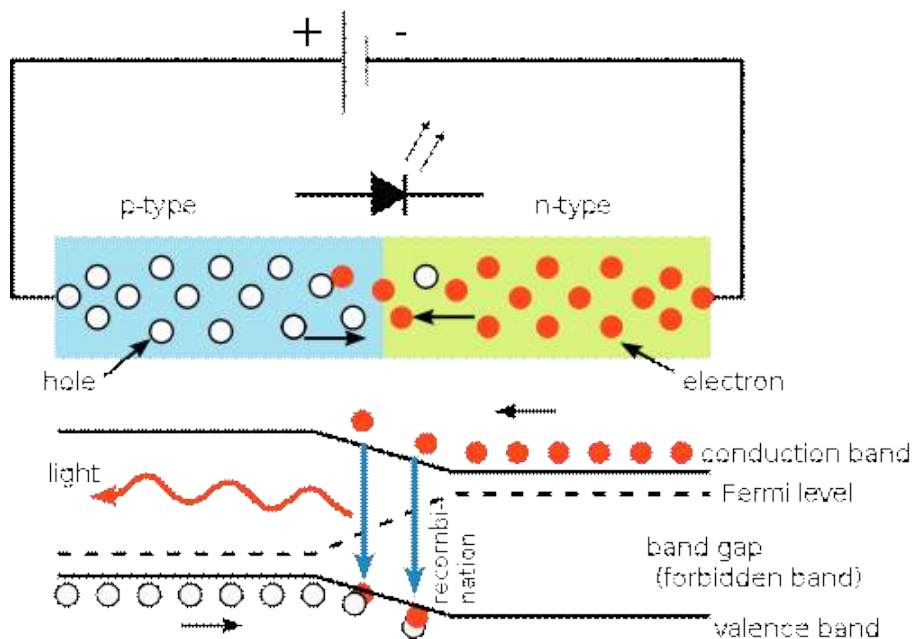
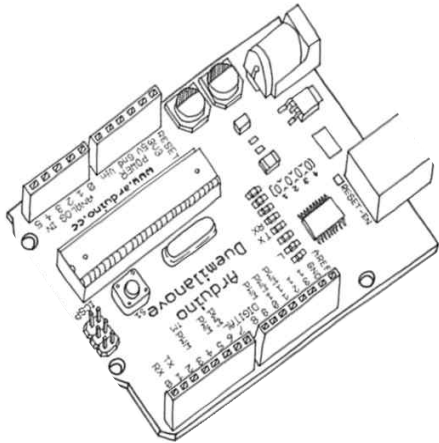



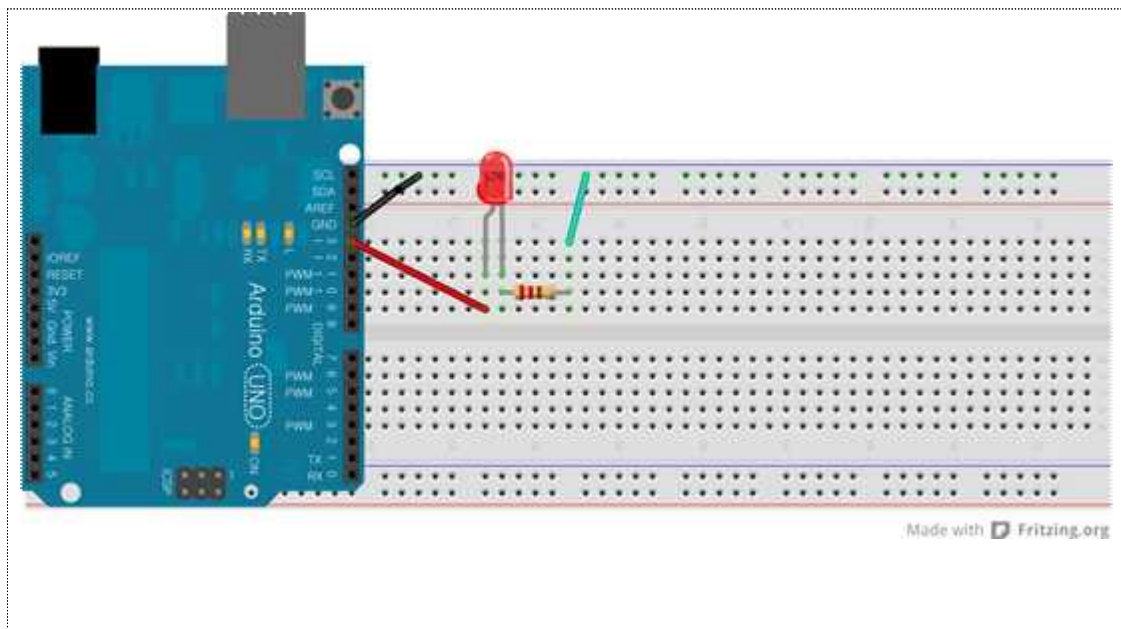
그림 1 LED의 원리

## 2) 준비물

	
아두이노 키트	LED 및 저항

## 3) 브레드보드 레이아웃

LED의 +는 아두이노에서 사용할 디지털 포트에 연결해준다. -는 아두이노의 GND에 연결하며 LED에 너무 많은 전류가 흐르지 않도록 위에서 계산한 적합한 저항을 연결하여 준다.



#### 4) 스케치

스케치를 아두이노 IDE에서 작성하고 Upload 한다.

01	int led = 13;//13번 핀에 led 연결
02	void setup()
03	{
04	pinMode(led, OUTPUT);//13번 핀의 출력 모드 설정
05	}
06	void loop()
07	{
08	digitalWrite(led, HIGH);//HIGH값을 출력한다.
09	delay(1000);            //1초가 dealy
10	digitalWrite(led, LOW); //LOW값을 출력한다.
11	delay(1000);            //1초가 delay
12	}
13	
14	

#### 5) 결과

스케치를 업로드하고 연결된 LED가 1초 간격으로 점멸하는 것을 확인할 수 있다.

#### 나. LED의 이해 도전과제

- 1) 다수의 LED를 연결하여 순서대로 점멸해 보기
- 2) LED의 점멸 주기를 변경하여 보기.

## 2. Hello PID

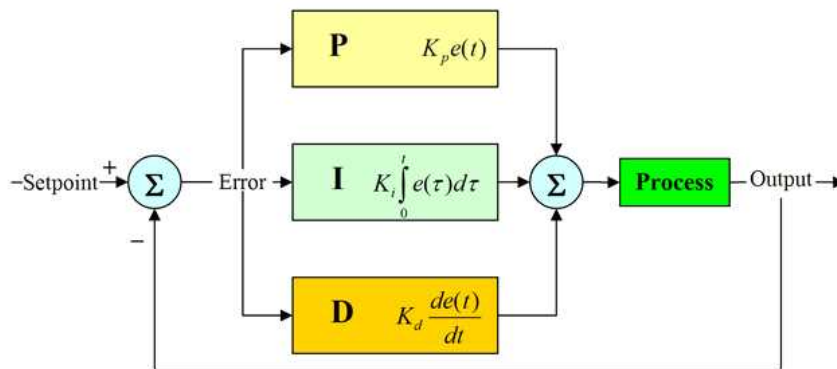
[학습목표]

- PID제어의 기초를 알 수 있다.
- 아두이노와 PID제어를 통해 주변 환경을 제어할 수 있다.

### 가. PID의 이해

#### 1) 개요

PID제어 기법은 비례-적분-미분 제어의 약자로서 실제 응용분야에서 가장 많이 사용되는 대표적인 제어 기법의 하나이다. PID제어기는 기본적으로 피드백 제어기의 형태를 가지고 있으며, 제어하고자 하는 대상의 출력값을 측정하여 이를 원하는 설정값과 비교하여 오차를 계산하고, 이 오차값을 이용하여 제어에 필요한 제어값을 계산하는 구조로 되어 있다.



표준적인 형태의 PID 제어기는 아래의 식과 같이 세 개의 항을 더하여 제어값을 계산하도록 구성이 되어 있다.

$$MV(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de}{dt}$$

이 항들은 각각 오차값, 오차값의 적분(integral), 오차값의 미분(derivative)에 비례하기 때문에 비례-적분-미분 제어기 (Proportional-Integral-Derivative controller)라는 명칭을 가진다. 이 세개의 항들의 직관적인 의미는 다음과 같다.

비례항 : 현재 상태에서의 오차값의 크기에 비례한 제어작용을 한다.

적분항 : 정상상태(steady-state) 오차를 없애는 작용을 한다.

미분항 : 출력값의 급격한 변화에 제동을 걸어 오버슈트(overshoot)을 줄이고 안정성(stability)을 향상시킨다.

PID 제어기는 위와 같은 표준식의 형태로 사용하기도 하지만, 경우에 따라서는 약간 변형된 형태로 사용하는 경우도 많다. 예를 들어, 비례항만을 가지거나, 혹은 비례-적분, 비례-미분항만을 가진 제어기의 형태로 단순화하여 사용하기도 하는데, 이때는 각각 P, PI, PD 제어기라 불린다.

위의 식에서 제어 파라미터  $K_p$ ,  $K_i$ ,  $K_d$ 를 이득값 혹은 게인(gain)이라고 하고, 적절한 이득값을 수학적 혹은 실험적/경험적 방법을 통해 계산하는 과정을 튜닝(tuning)이라고 한다. PID 제어기의 튜닝 중 가장 널리 알려진 것으로는 지글러-니콜스 방법이 있다.

## 2) 준비물

### 하드웨어

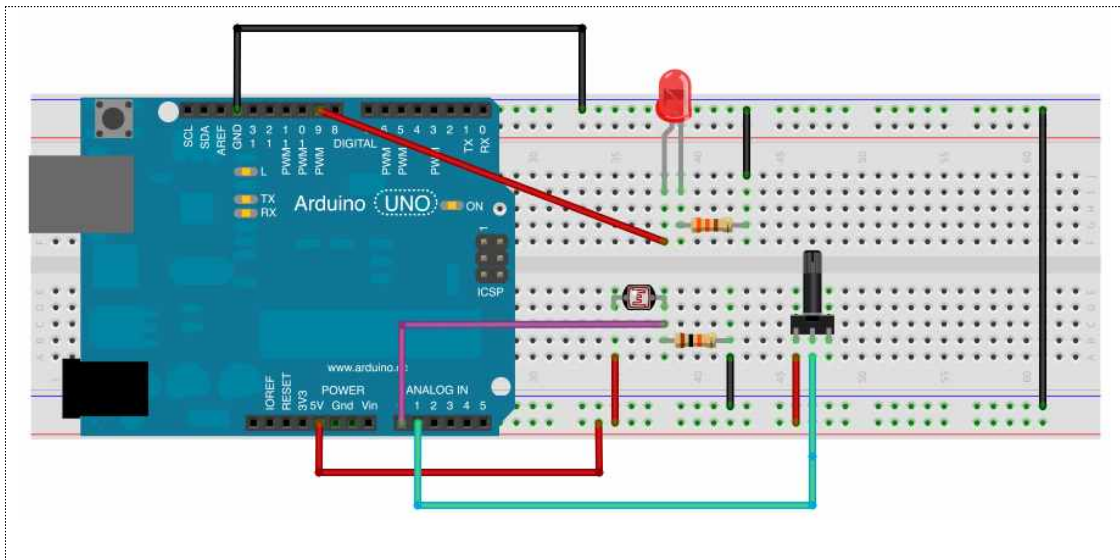
	
빛 센서	LED
	
가변 저항	
	
저항(10킬로, 220옴)	

### 소프트웨어

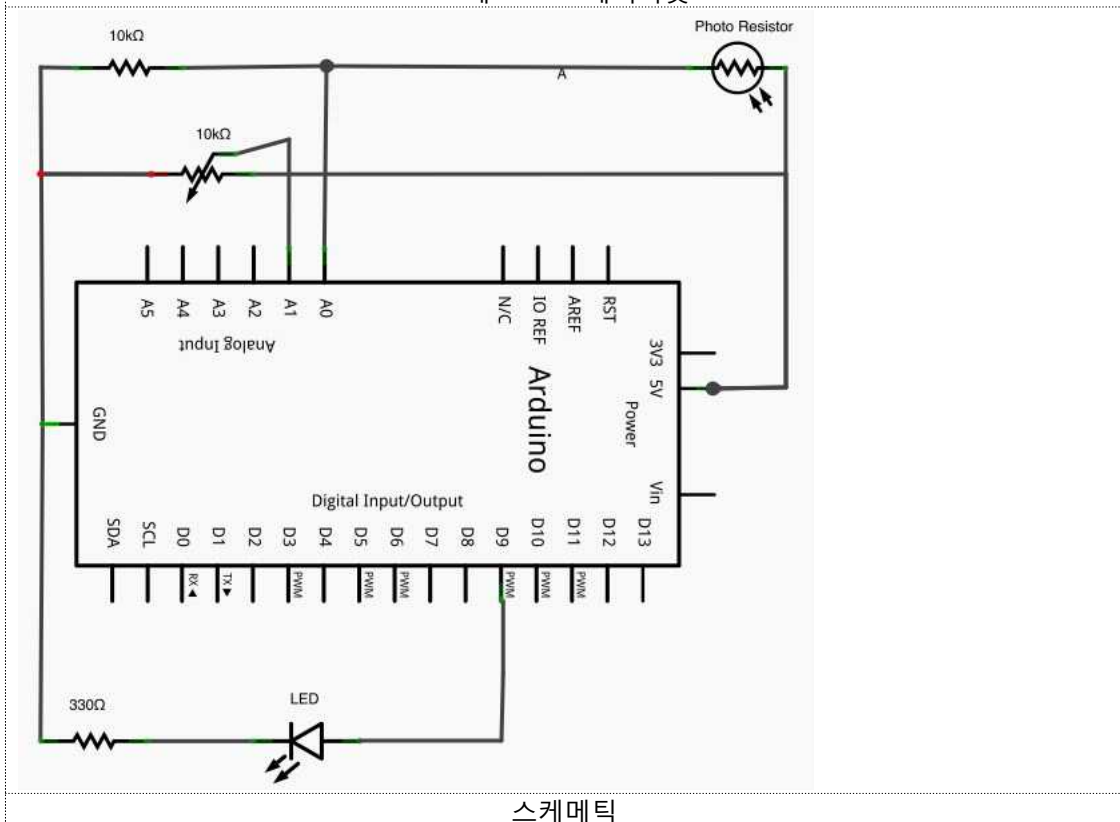
<a href="https://github.com/br3ttb/Arduino-PID-Library/archive/master.zip">https://github.com/br3ttb/Arduino-PID-Library/archive/master.zip</a>	<a href="http://www.sojamo.de/libraries/controlP5/">http://www.sojamo.de/libraries/controlP5/</a>
아두이노 PID 라이브러리	controlP5(프로세싱 라이브러리)

### 3) 브레드보드 레이아웃

빛 센서를 이용하여 외부에서 받아들이는 값을 PID 제어의 입력으로 사용하며, 이를 위해 LED의 밝기를 조절하는 것이 출력값이 된다. 목표값은 가변저항으로 조절하게 되며 이를 통해 아두이노를 활용한 PID 제어의 기본적인 사용법을 알아본다.



브레드보드 레이아웃



스케메틱

#### 4) 스케치

가변저항과 빛 센서의 값을 읽기 위해서는 analogRead 함수를 사용해야 한다. 연속적으로 변하는 형태의 값인 아날로그 값을 읽기 위해서는 아날로그 핀에 연결된 외부 입력으로 부터 값을 읽어 들여야 한다. 아두이노 Uno는 6채널의 아날로그 입력 포트를 가지고 있으며, 10비트의 해상도를 가진다. 아날로그 입력 역시 전압의 변화 값으로 읽혀지게 되며 0~5V 사이의 변화 값은 10비트 해상도에 따라 0~1023의 정수 값으로 변환되어 입력이 된다. 위 회로를 성공적으로 구성한 후 다음의 코드를 통해 빛 센서와 가변 저항의 값을 읽어서 시리얼 모니터로 출력해보자.

아날로그 입력

```
01 int sensorPin = A0;    // select the input pin for the potentiometer
02 int sensorValue = 0;  // variable to store the value coming from the sensor
03
04 void setup() {
05     Serial.begin(9600);
06 }
07
08
09 void loop() {
10     // read the value from the sensor:
11     sensorValue = analogRead(sensorPin);
12     Serial.println(sensorValue);
13 }
```

아날로그 출력

지금까지 LED를 끄고 켜는 것은 단순히 연결된 포트에 HIGH, LOW 값을 주어서 제어한 것이다. 이러한 출력은 디지털 값을 통한 제어였으며, 각각의 LED의 밝기를 조절할 수는 없었다. PWM(Pulse Width Modulation)은 아두이노가 아날로그 출력을 만들어 낼 수 없기에 펄스의 폭을 빠르게 변화시킴으로써 평균 전압의 양을 조절하여 아날로그 출력을 만들어 내는 것이다. 아날로그 출력 함수의 인자로서 0을 넘겨주면 0V의 전압이 나가게 되는 것이며 255까지의 값을 이용하여 그 전압의 크기를 조절한다. 아두이노의 PWM 주파수는 500Hz, 2ms 이므로 아주 빠르며 LED 점등에 사용하면 우리 눈은 그 차이를 거의 구별하지 못할 만큼 서서히 밝기를 조절할 수 가 있다. 아두이노의 PWM 출력은 디지털 핀에 ~ 표시가 된 포트를 통하여 출력할 수 있다.(3, 5, 6, 9, 10, 11핀)

아래 코드는 LED가 연결된 디지털 핀에 analogWrite함수를 이용하여 밝기를 서서히 조절해주는 코드이다. 해당 코드를 실행 시켜 보면 서서히 밝아졌다가 어두워지는 것을 알 수 있을 것이다.

```
int ledPin = 3;    // LED connected to digital pin 9

void setup() {
  // nothing happens in setup
}

void loop() {
  // fade in from min to max in increments of 5 points:
  for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }

  // fade out from max to min in increments of 5 points:
  for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```



이번 프로젝트의 목표는 가변 저항의 값을 목표치로 삼아 LED의 밝기를 조절하는 것이다. 외부 밝기의 측정은 위의 회로에서처럼 빛 센서를 통해서 받아들인 수치로 정한다. 즉 PID 제어를 위한 Setpoint: 가변 저항 값, Input: 빛 센서의 값, Output: LED 출력 값이 된다.

<pre>#include &lt;PID_v1.h&gt; const int photores = A0; const int pot = A1; const int led = 9; double lightLevel; // Tuning parameters const double Kp=0.01; const double Ki=90; const double Kd=0.001; double Setpoint, Input, Output; PID myPID(&amp;Input, &amp;Output, &amp;Setpoint, Kp, Ki, Kd, DIRECT); const int sampleRate = 1; // Communication setup const long serialPing = 500; // Serial pingback interval in milliseconds unsigned long now = 0; // placehodler for current timestamp unsigned long lastMessage = 0; // last message timestamp.</pre>	<pre>void setup(){     lightLevel = analogRead(photores);     Input = map(lightLevel, 0, 1024, 0, 255);     Setpoint = map(analogRead(pot), 0, 1024, 0, 255);     Serial.begin(9600);     myPID.SetMode(AUTOMATIC);     myPID.SetSampleTime(sampleRate);     Serial.println("Begin"); // Hello World!     lastMessage = millis(); // timestamp }  void loop(){     Setpoint = map(analogRead(pot), 0, 1024, 0, 255);     lightLevel = analogRead(photores);     Input = map(lightLevel, 0, 900, 0, 255);     myPID.Compute();     analogWrite(led, Output);     now = millis();     if(now - lastMessage &gt; serialPing) {         Serial.print("Setpoint = ");         Serial.print(Setpoint);         Serial.print(" Input = ");         Serial.print(Input);         Serial.print(" Output = ");         Serial.print(Output);         Serial.print("\n");         lastMessage = now;     } }</pre>
--	---

참고 코드

FirstPID.ino

FirstPID\_Serial.ino

PID\_FrontEnd\_ArduinoSampleCode.ino

PID\_FrontEnd\_v03.pde

