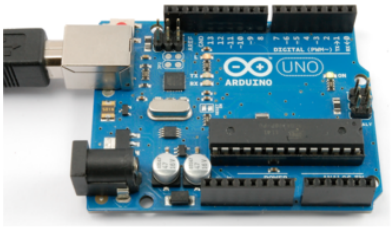
[아두이노 강좌] 1. LED 깜빡이기

아두이노 강좌

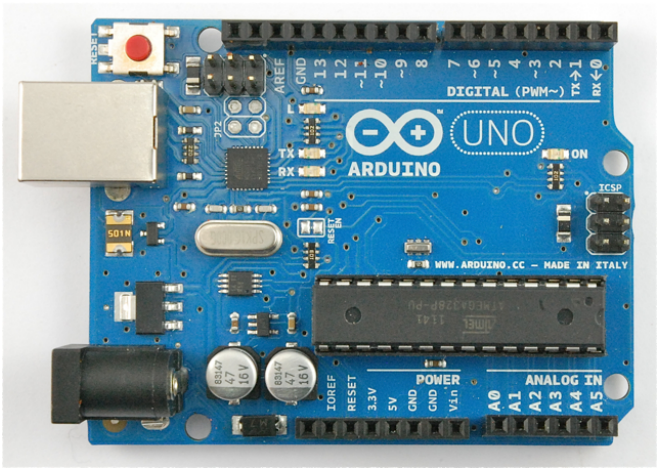
2013/06/12 11:10 http://blog.naver.com/ubicomputing/150169791025

본 게시물에서는 아두이노를 프로그래밍하여 아두이노 상의 LED를 깜빡거리게 하는 내용을 설명합니다.



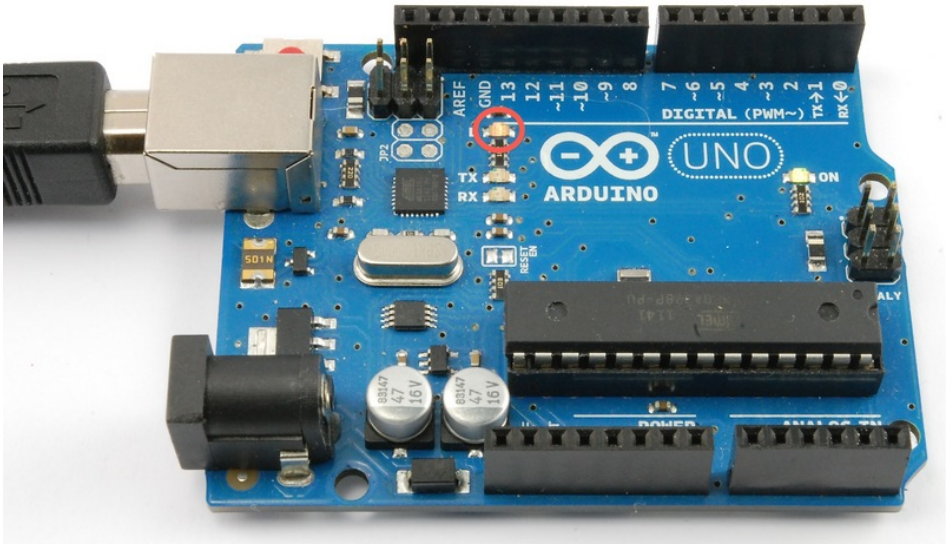
필요한 장비

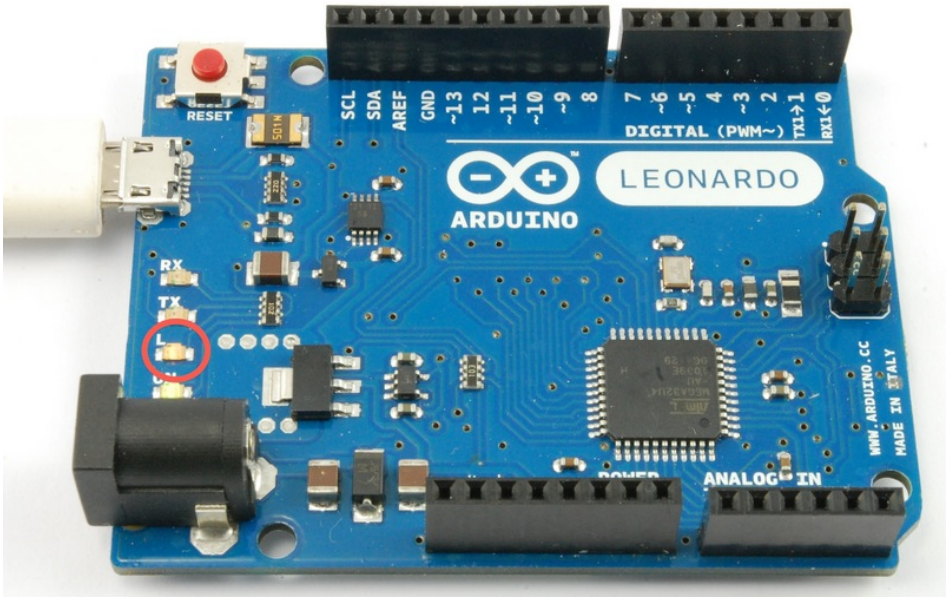
	이름	수량

	Arduino Uno R3	1
	USB케이블 (type A-B)	1

보드상의 L표시 LED

아두이노는 양쪽 사이드에 일렬의 커넥터를 가지고 있어 다른 디바이스를 연결하거나 쉴드등을 연결할 수 있습니다. 하지만 아두이노 보드상에도 LED를 하나 가지고 있어 스케치에서 제어를 할 수 있는데 LED옆에 L이라고 표시가 되어 있어 흔히 L LED라고 부릅니다. (아래에 동그라미로 표기된 부분 참조)

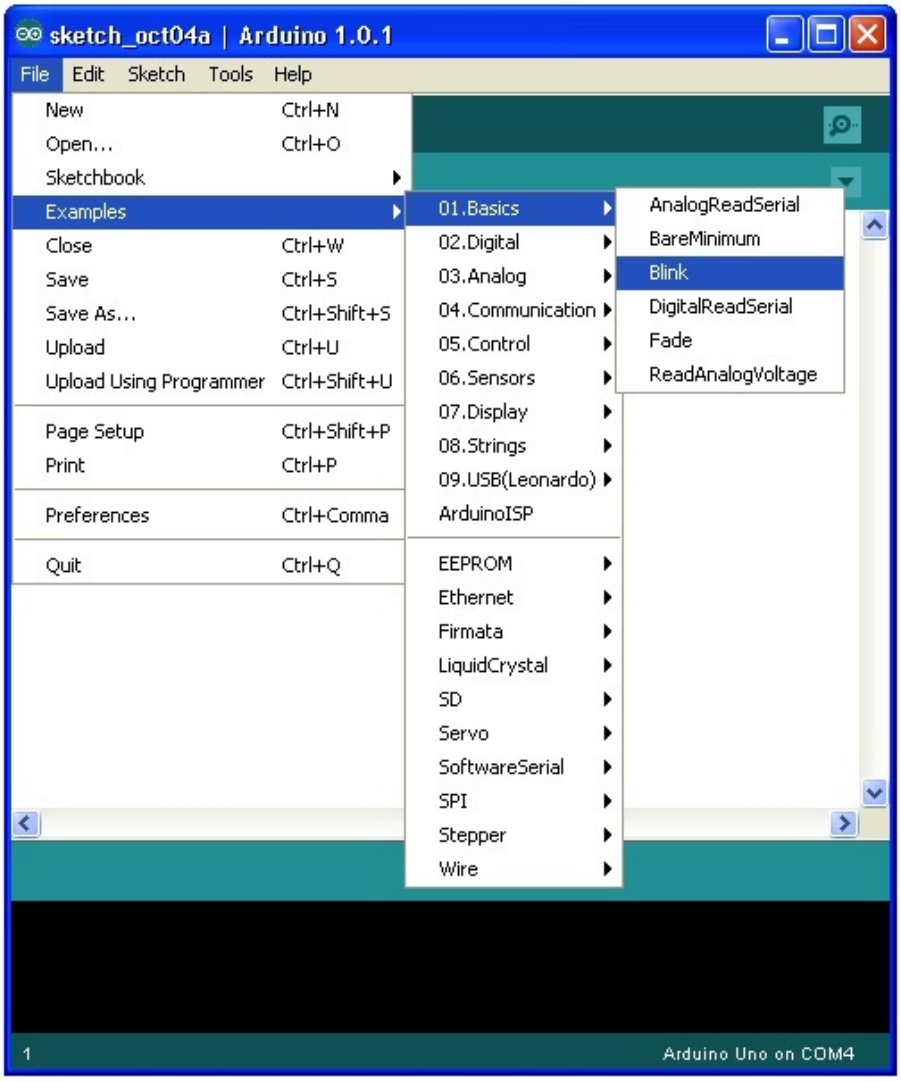




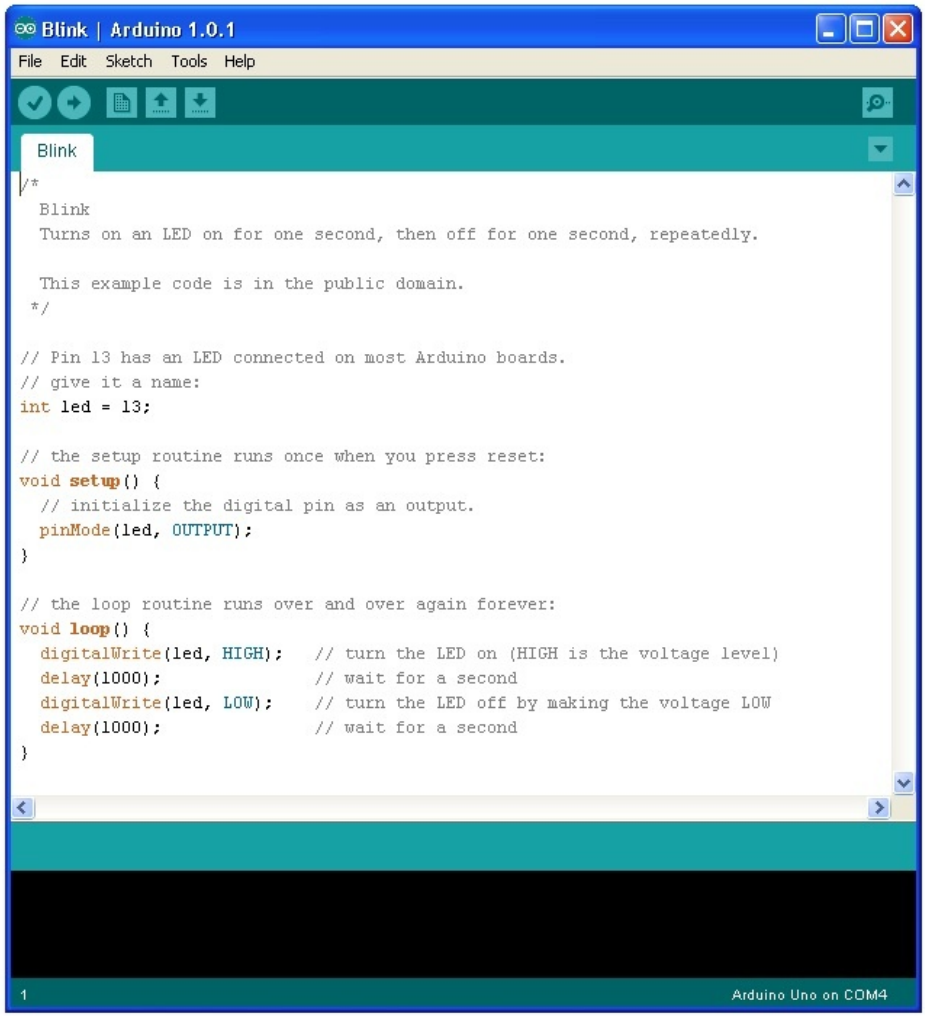
'Blink' 예제 로딩하기

아두이노에 USB케이블을 연결하여 전원을 공급하면 아두이노의 LED가 이미 깜빡이는 것을 볼 수 있을 수 있습니다. 이것은 보통 아두이노를 생산할때 미리 blink 스케치를 인스톨하였기 때문에 그렇습니다.
여기서는 blink 예제에서 LED가 깜빡이는 속도 등을 조절하여 다시 아두이노를 재 프로그래밍 하여보도록 하겠습니다.
아두이노 IDE를 www.arduino.cc에서 다운받아 셋업하고, 올바른 시리얼 포트를 찾아 아두이노를 컴퓨터와 통신가능한 상태로 만듭니다. 이 통신연결을 이용하여 아두이노에 프로그래밍을 하겠습니다.

아두이노 IDE는 사용가능한 많은 예제 스케치들을 포함하고 있습니다. Blink 스케치는 IDE의 메뉴 File --> Examples --> 0.1 Basics에서 찾을 수 있습니다.

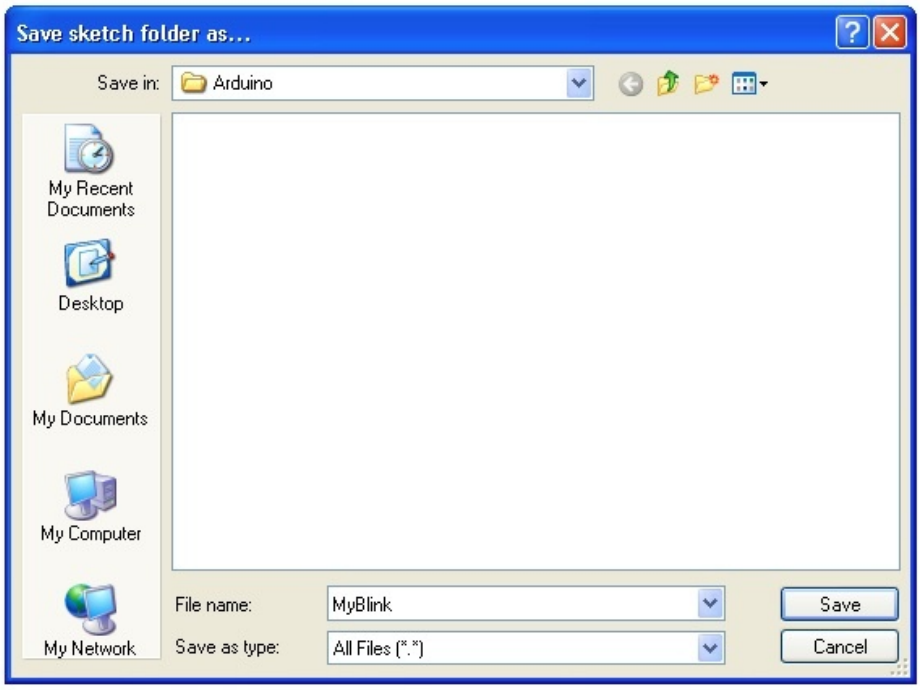


스케치 윈도우가 열리면 적절하게 화면 크기를 조절하여 스케치 전체가 보일 수 있게 하시는 것이 좋습니다.

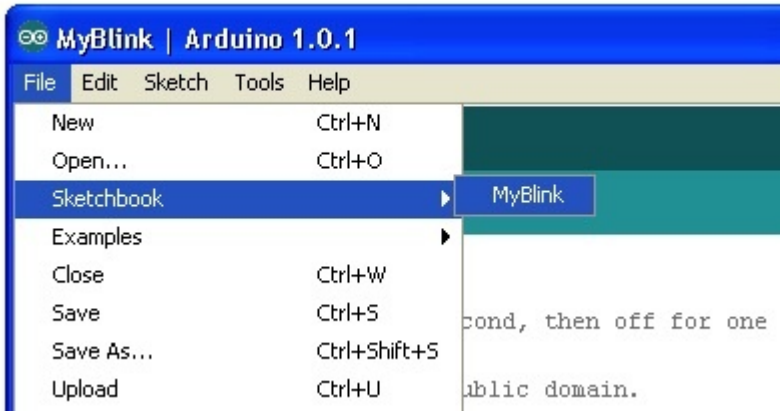


Blink 스케치 복사본 저장하기

아두이노 IDE에 포함된 예제 스케치 파일들은 읽기만 가능합니다. 수정을 하려면 다른 파일 형태로 저장을 하여 주어야 합니다. IDE에서 Save As 옵션을 선택하여 'MyBlink'라는 이름으로 저장합니다.



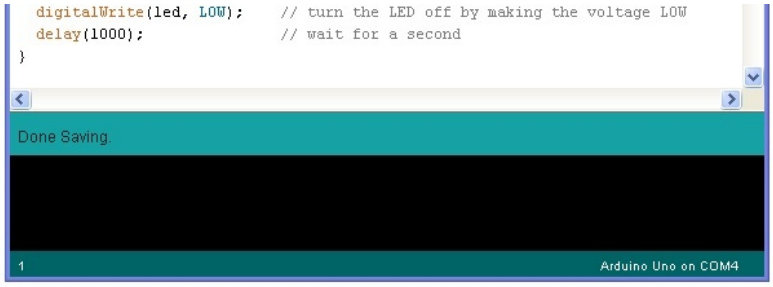
저장을 하게 되면 sketchbook에서 MyBlink를 확인 할 수 있습니다. 차후에 다시 스케치 코드를 살펴보려면 File --> Sketchbook 메뉴에서 스케치를 로드 할 수 있습니다.



보드에 스케치 업로드 하기

아두이노 보드를 USB 케이블을 이용하여 컴퓨터에 연결하고 Board Type과 Serial Port가 올바르게 설정되어 있는지 확인합니다.

아두이노 IDE의 하단을 보면 현재 셋팅되어 있는 환경이 나타납니다.

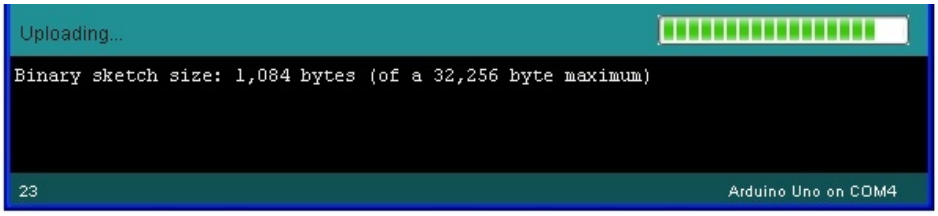


아래의 그림과 같이 생신 Upload버튼을 클릭합니다.

IDE의 하단의 상태영역을 보면, 프로그레스바와 메세지가 나타나는 것을 볼수 있습니다. 스케치를 컴파일한다고 나타나 있네요.

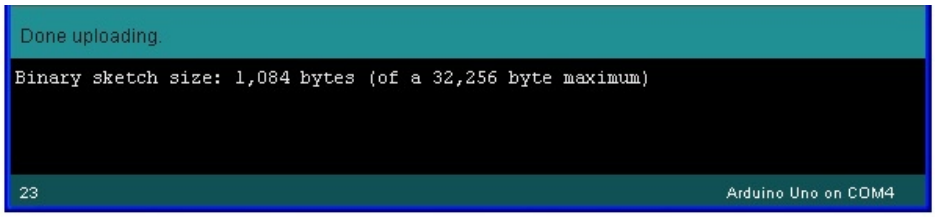
" target=_blank〉

컴파일이 끝나니 상태영역에 Uploading이라는 메세지가 나타났습니다. 컴파일된 스케치가 아두이노로 전송이 되고 있는 상태입니다. 아두이노상의 LED가 깜빡거립니다.

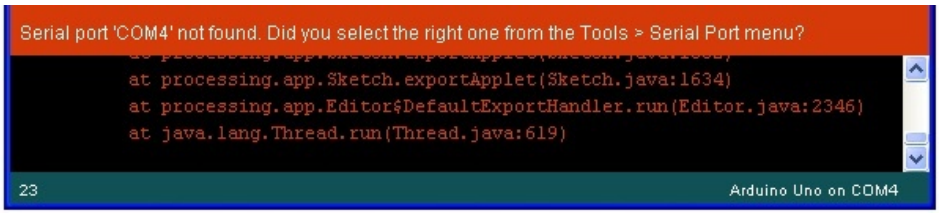


상태가 Done으로 변경되었습니다.

" target=_blank〉



그리고 업로딩한 스케치가 전체 32,256 바이트에서 1,084 바이트를 사용하고 있다고 말해주고 있습니다. 만약 컴파일중에 아래와 같은 에러 메세지가 나타난다면 아마도 보드가 컴퓨터와 연결되지 않아서 나타나거나 드라이버가 설치되지 않거나 잘못된 시리얼 포트가 선택되어 나타나는 문제입니다.



업로드가 끝나면 리셋되고 LED가 깜빡이기 시작합니다.

Blink 스케치 코드는 어떻게 동작하는가?

아래는 blink 스케치 코드입니다.

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
```

```
// the loop routine runs over and over again forever:
void loop() {
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

첫번째 코드는 보는 바와 같이 아래와 같습니다.

```
int led = 13;
```

코드에 코멘트로도 설명이 되어 있지만 led라는 변수를 선언하여 이름을 부여하고 led가 붙어 있는 13핀을 의미하는 숫자 13이 할당되었습니다.
다음으로는 setup 함수가 나와 있는데 이 함수는 아두이노가 리셋되고 한번만 실행이 되는 함수입니다. 그렇게 때문에 대부분 초기화 관련 루틴을 넣어 사용하는 함수 입니다.

```
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
```

모든 아두이노 스케치는 반드시 setup 함수를 가지고 있어야 합니다. 여기에서는 LED핀을 출력모드로 설정하라는 코드가 포함되어 있습니다.
모든 스케치에 setup함수가 포함되어야 하는 것처럼 loop함수도 반드시 포함되어 있어야 합니다. setup함수가 리셋후 한번만 실행되는 것과는 달리 loop함수는 계속 반복 됩니다.

```
void loop() {
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
```

```
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
```

loop 함수안에는 LED핀을 켜는(HIGH) 명령이 있으며, 1초동안 딜레이 후 LED핀을 끄는 명령을 담고 있으며, 끈 후에도 역시 1초동안 딜레이를 주는 명령으로 작성되어 습니다.

깜빡이는 속도변경하기

LED가 깜빡이는 속도를 변경시키려면 무엇을 변경해야 할까요? delay함수의 파라메터값인 1000을 변경하면 LED가 깜빡이는 속도를 변경할 수 있습니다.

만약 1000을 500으로 변경한 후 컴파일하여 업로딩한다면 LED가 이전 보다 두배 빨리 깜빡이는 것을 볼 수 있습니다.

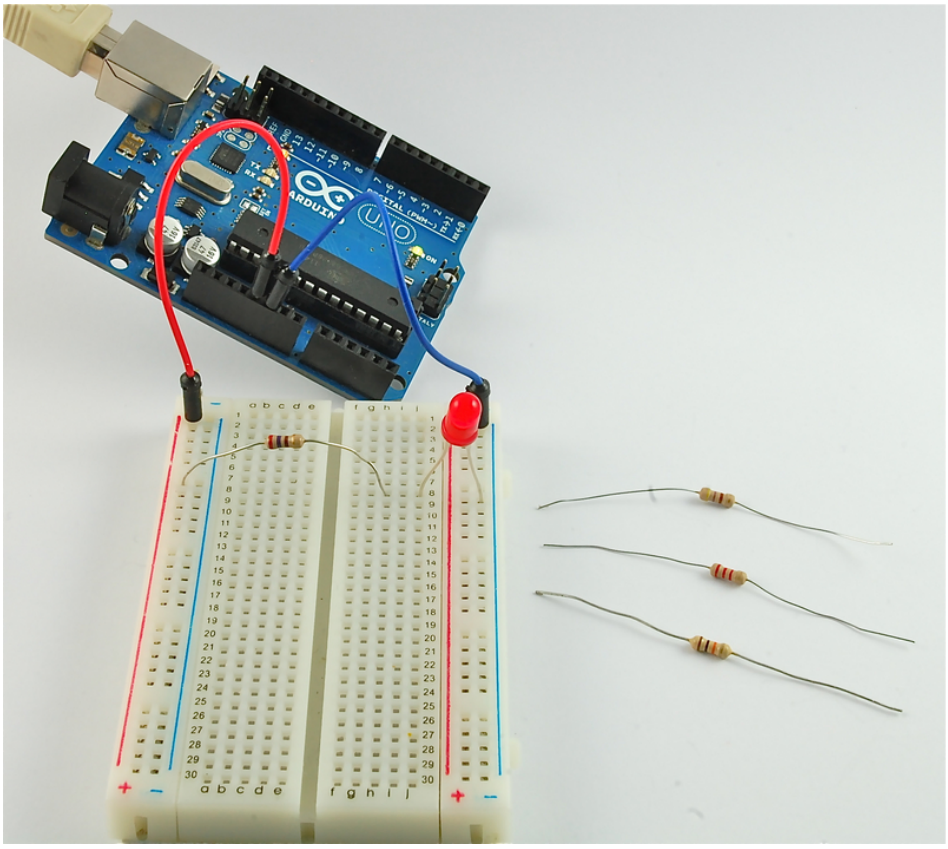
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 2. LED 밝기 변경하기

아두이노 강좌

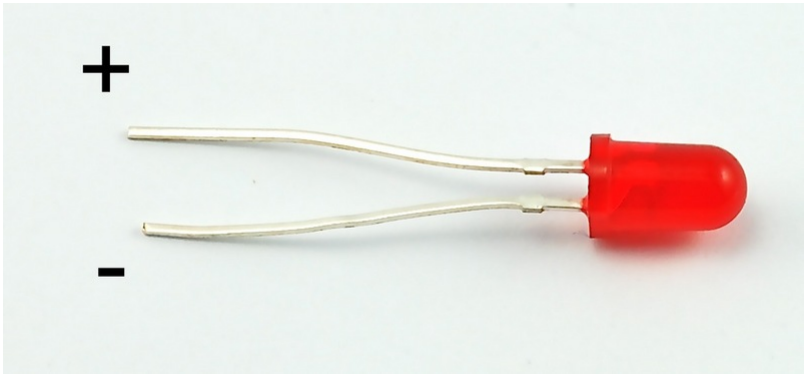
2013/06/12 13:12 http://blog.naver.com/ubicomputing/150169797910

본 게시물에서는 여러가지 저항을 사용하여 LED의 밝기를 어떻게 변경하는지 알아 보겠습니다.



LED

LED는 매우 작은 전류를 사용하며 오래 가는 부품으로 빛을 발산하는 부품입니다.
LED를 배터리라 전원소스에 직접 연결하면 안되는데, 그 이유는 LED로 들어오는 전류의 양을 제한하기 위해 반드시 저항과 같이 사용히야하기 때문입니다. 만약 저항을 이용하여 LED로 들어오는 전류의 양을 제한하지 않는다면 LED는 타버리게 됩니다. 그리고 LED는 음극과 양극을 가지고 있어 극성을 달리하여 연결하면 빛이 나오지 않습니다.



LED를 저항과 같이 사용하지 않는다면 대부분의 LED는 빛을 발산하는 부분 거의 즉시 망가지게 됩니다.

LED의 음극과 양극을 확인하는 방법은 크게 두 가지 입니다.

- 먼저 양극은 좀 더 다리가 깁니다.
- 두번째로 음극 다리가 붙어 있는 LED 케이스에 납짝한 부분이 있습니다.

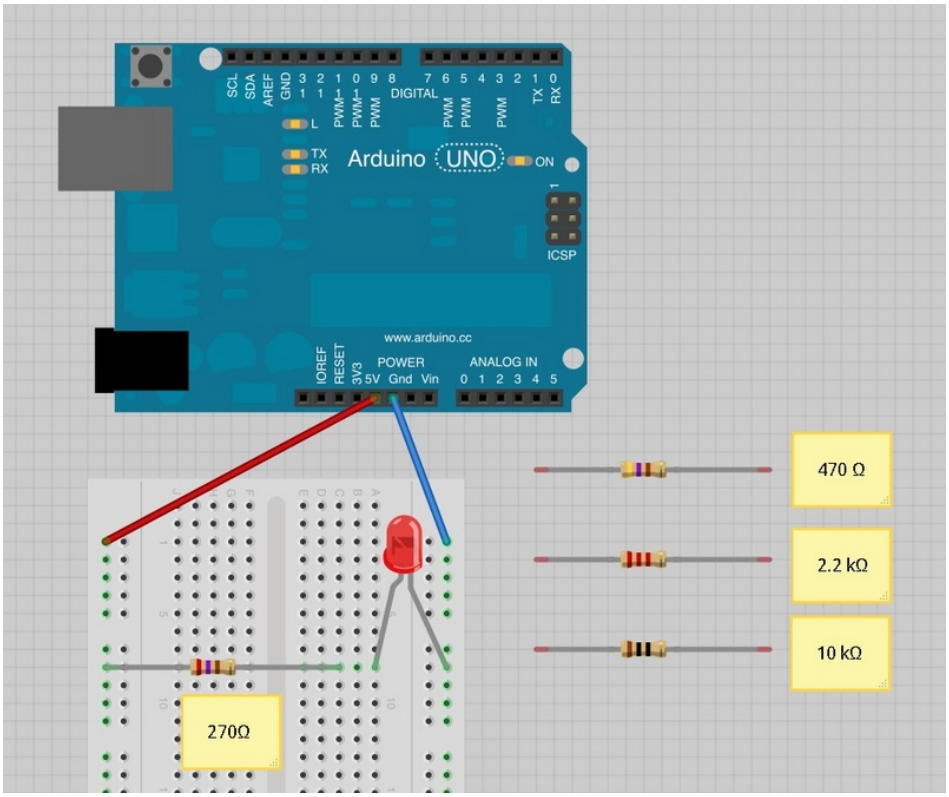
저항

저항은 그 이름이 암시하듯 전기의 흐름을 방해하는 부품입니다. 저항의 값이 높아 질수록 흐르는 전류의 양은 줄어 듭니다. 이러한 저항을 이용하여 LED에 들어가는 전류의 양을 조절하여 결과적으로 LED의 밝기를 조절할 수 있게 됩니다.

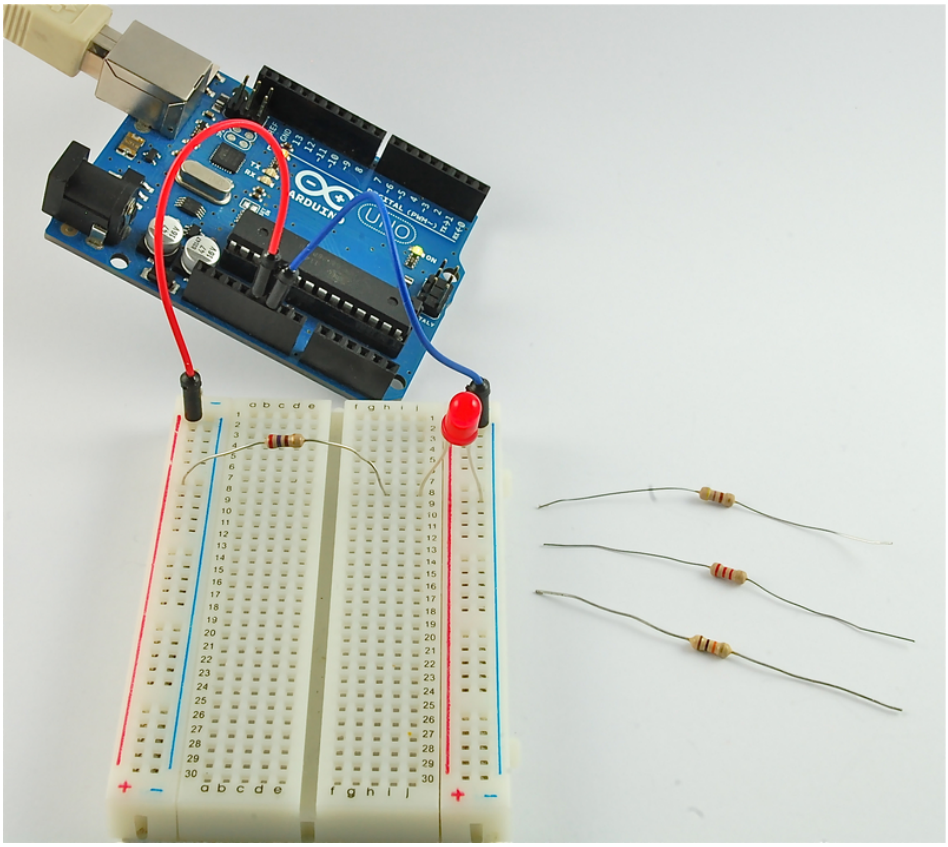
여기에서는 4개의 서로다른 값을 가진 저항(270Ω, 470Ω, 2.2kΩ,10kΩ)을 사용하여 실험을 진행하여 보도록 하겠습니다. 참고로 저항의 값은 저항에 색칠되어 있는 컬러로 확인 할 수 있으며 LED와 다르게 저항에는 극성이 없습니다.

브레드보드 레이아웃

아래와 같이 브레드보드에 부품을 연결 합니다.



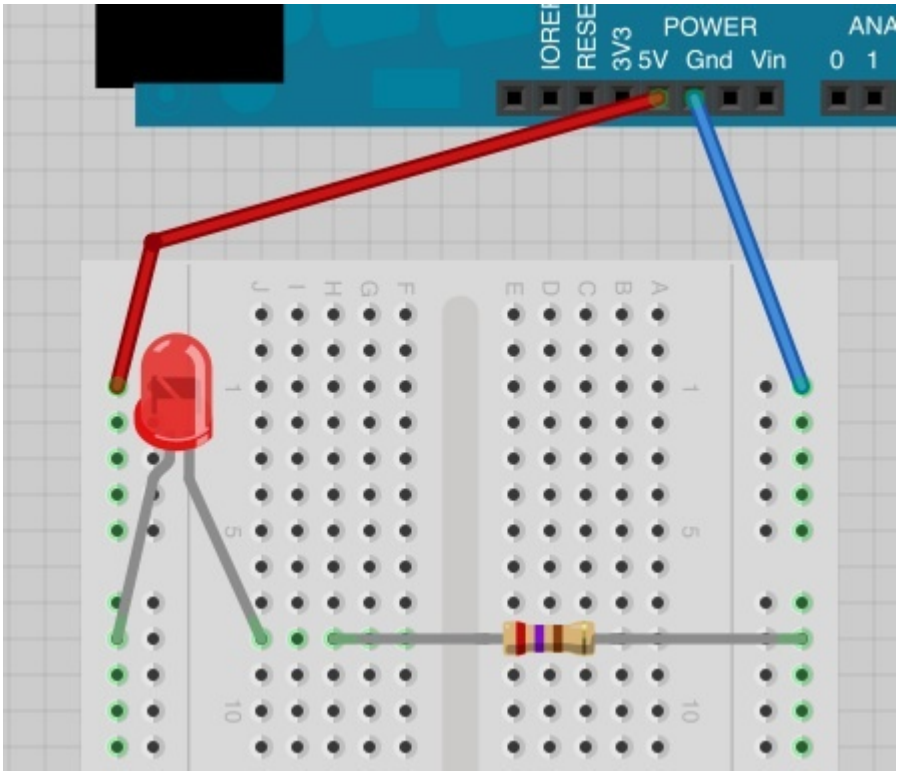
아두이노는 편리한 5V 전원 소스입니다. 여기서는 LED와 저항에 5V전원을 공급하도록 결선되었습니다. 여기서 아두이노는 단지 전원만 공급하므로 USB 케이블 연결이외에는 아무것도 할 필요가 없습니다.



270 Ω 저항을 연결하였을 경우 LED는 꽤 밝습니다만 470오옴 저항을 연결하면 LED가 살짝 어두워 집니다. 만약 2.2K오옴 저항을 연결한다면 LED는 많이 흐릿하여 질것입니다.

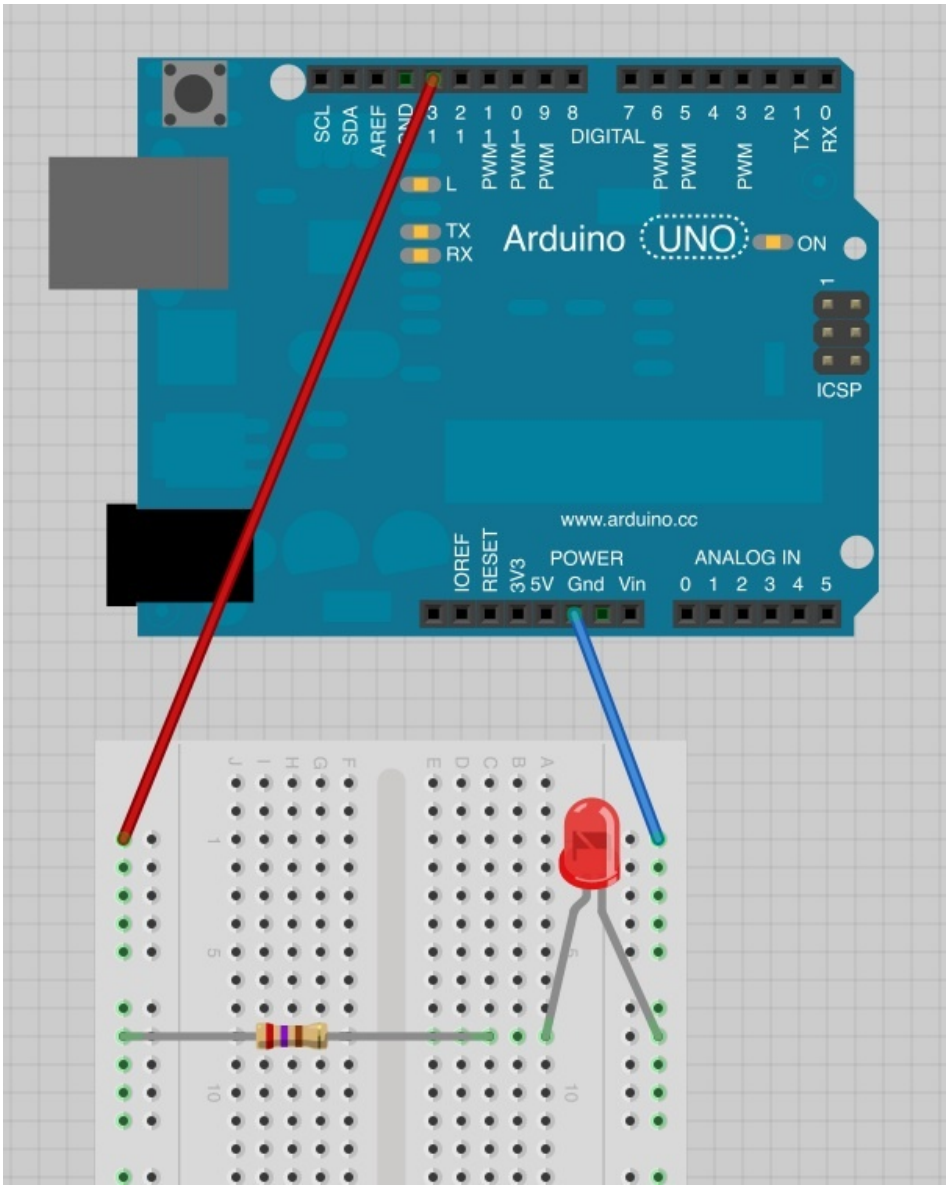
저항 위치 변경하기

먼저 설치 했던 LED의 위치를 아래와 같이 반대로 연결하여도 LED는 동작합니다. 저항은 LED의 음극, 양극 어느쪽이든 위치하기만 하면 LED는 동작합니다.



LED깜박이게 하기

브레드보드의 결선을 아래와 같이 살짝 변경하여 보도록 하겠습니다. 기존에 아두이노 5V에 연결되어 있던 빨강 와이어를 D13으로 연결합니다.



강좌 1에서 작성하였던 Blink예제를 아두이노에 업로드합니다. 그러면 브레드보드에 설치된 LED와 아두이노 보드상의 L LED도 같이 깜빡이는 것을 확인 할 수 있습니다.

빨강 점퍼를 이번에는 D13에서 D7 위치로 옮겨보도록 하겠습니다. 그리고 blink예제의 아래 라인을 찾아 13을 7로 수정합니다.

int led = 13;

이렇게 수정합니다.
int led = 7;

컴파일하여 아두이노에 업로드합니다. LED가 역시 깜박이는 것을 볼수 있지만 이번에는 D7핀을 이용하였습니다.

가치창조기술 | www.vctec.co.kr

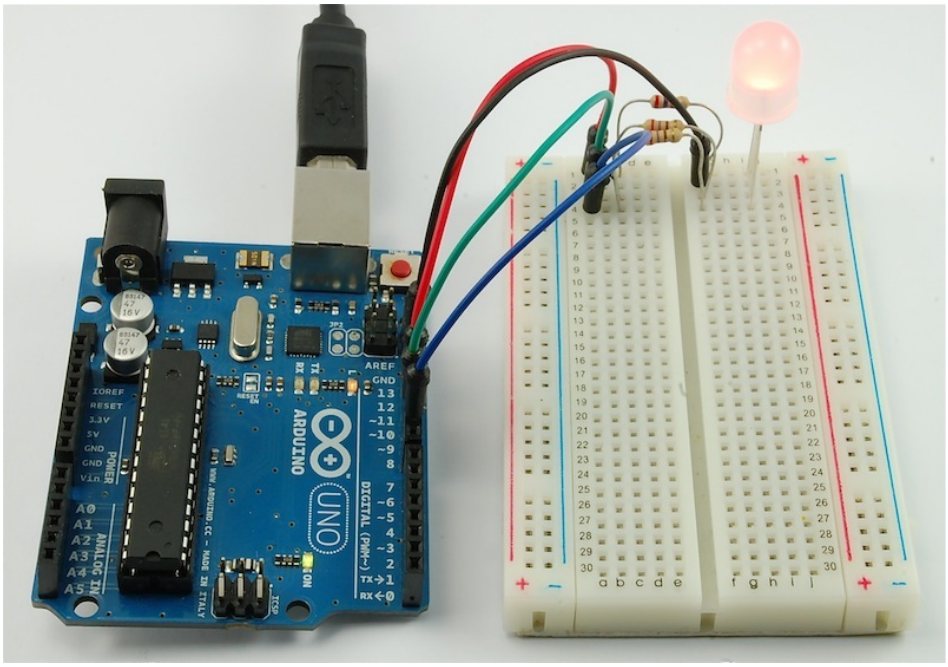
[아두이노 강좌] 3. RGB LED 색깔 변경하기

아두이노 강좌

2013/06/12 19:18 http://blog.naver.com/ubicomputing/150169821822

이번 게시글에서는 아두이노로 RGB LED를 사용하는 방법에 대해 설명하겠습니다.

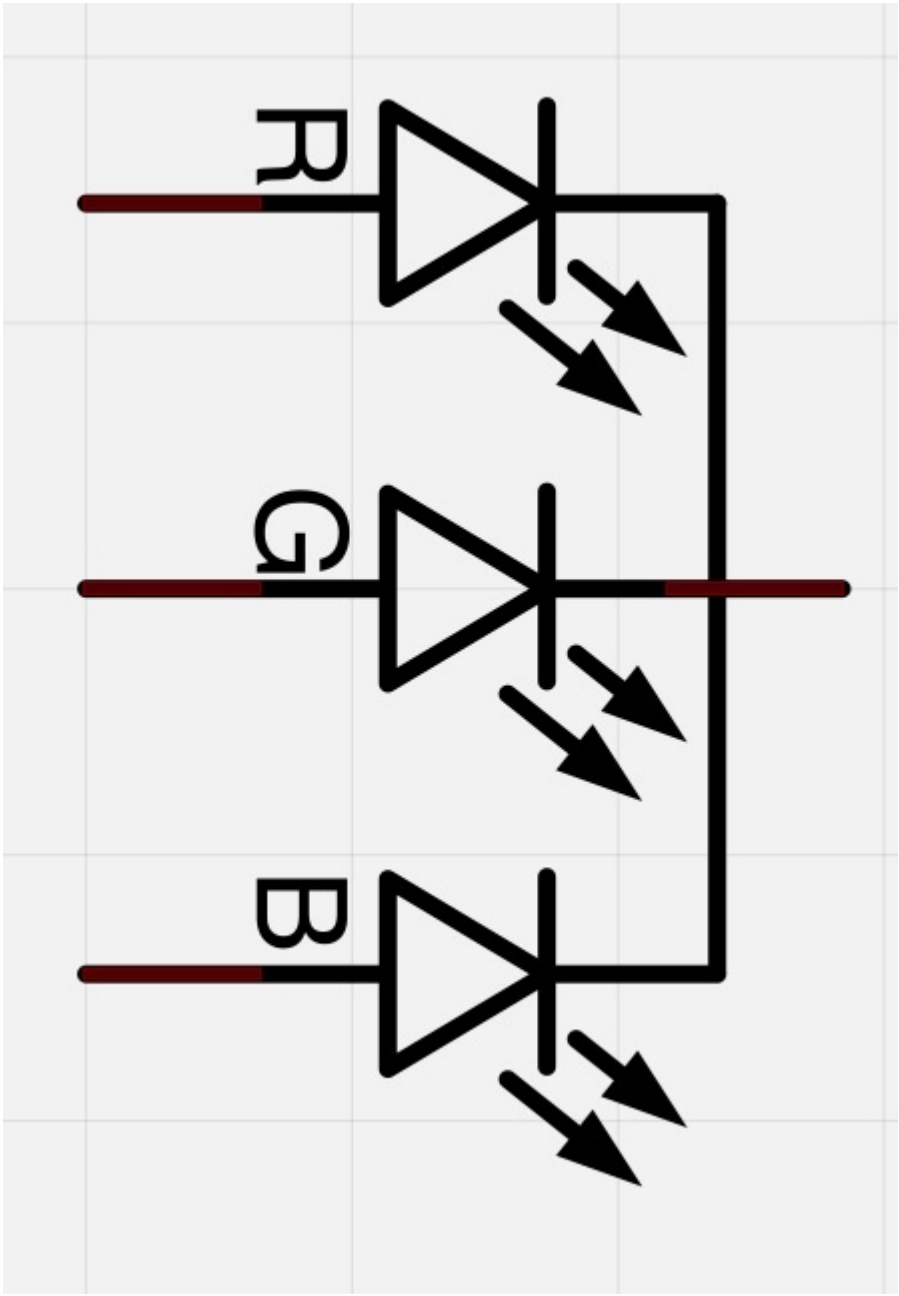
LED의 컬러를 제어하기 위해서는 아두이노의 analogWrite 함수를 사용하게 될 것입니다.



먼저 RGB LED를 보면 일반 LED와 비슷하게 생겼지만 그 내부는 실제적으로 세개의 LED(각각 빨강, 초록, 파랑)가 있습니다. 이 세개의 LED의 밝기를 조절하여 섞으면 원하는 컬러를 만들수가 있습니다.
이것은 마치 물감을 가지고 색을 만들어 내는 것과 비슷한 작업입니다. LED의 밝기는 공급되는 전원의 양에 의해 제어되며, 아두이노의 analogWrite함수를 이용하여 전원의 양을 조절할 수 있습니다.

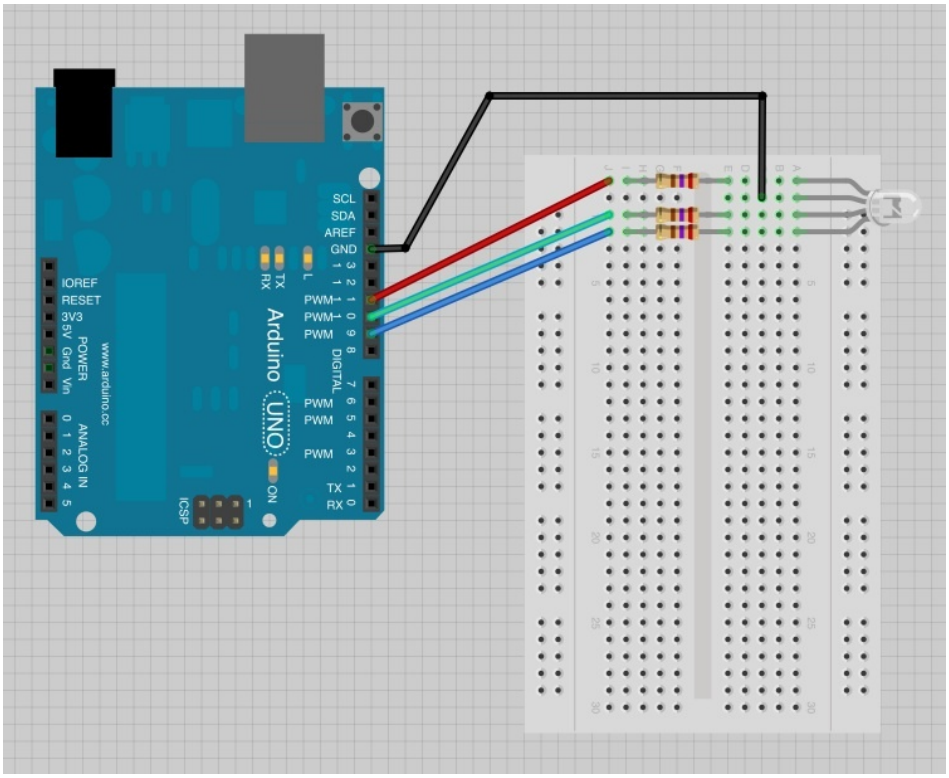
브레드보드 레이아웃

RGB LED는 4개의 다리가 나와 있습니다. 각각의 LED에는 양극 리드가 연결되어 있으며 세개의 LED의 음극리드는 하나로 묶여서 한개의 음극리드로 나와 있습니다.



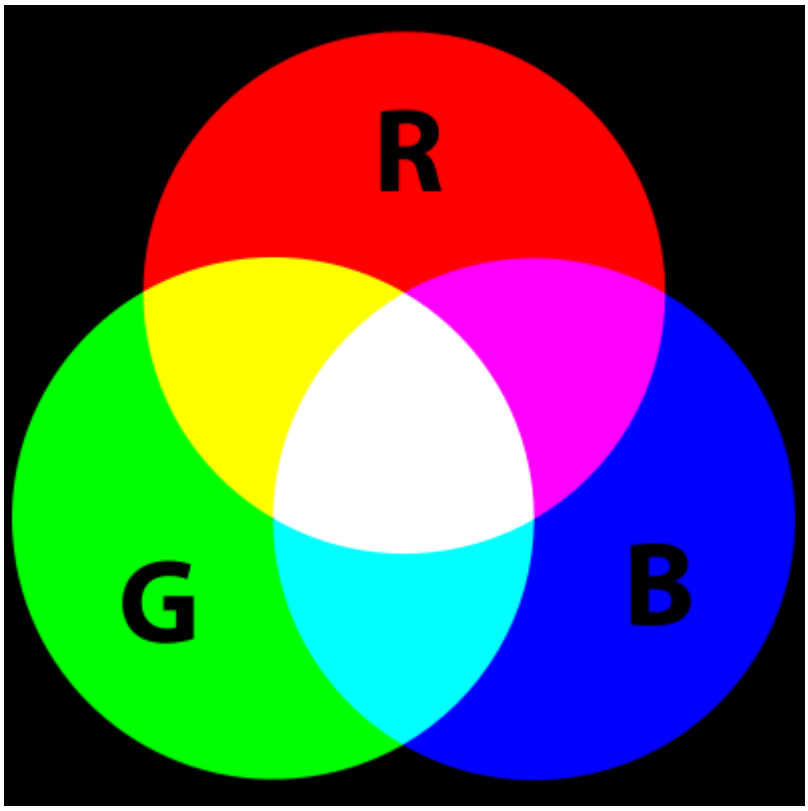
공통으로 사용하는 음극핀은 LED의 평평한 면으로부터 두번째 핀이며, 4개의 리드중 제일 깁니다. 이 리드는 그라운드로 연결되어야 합니다.
내부의 각 LED는 너무 많은 전류가 LED로 흘러와 LED가 망가지는 것을 방지하기 위해 270오옴의 저항이 각각 필요합니다. 세개의 양극 리드는 아두이노의 출력핀에 연결되는데 이 연결 사이에 아래와 같이 저항이 위치하여야 합니다.

만약 사용하는 LED의 양극이 1개이고 음극이 3개라면 제일 긴 다리를 가진 양극을 GND대신 +5에 연결하십시오.



색깔

아마도 중고등하교 미술시간에 여러가지 물감을 빨강, 초록, 파랑 물감을 섞어 검정 물감을 만들어 보신 기억이 있을 겁니다. 빛은 물감과 다르게 빨강, 초록, 파랑 빛을 섞으면 하얀색이 됩니다. 세개의 LED의 밝기를 동일하게 설정하면 전체적인 컬러는 하얀색이되며 파랑LED를 끄고, 빨강과 초록 LED의 밝기를 동일하게 설정하면 노랑색이 됩니다. 이러한 방식으로 원하는 색을 만들어 낼수 있습니다.



아두이노 스케치

아래의 스케치 여러가지 색을 LED가 반복하여 표현할 수 있게 작성하여 놓은 스케치 코드입니다.

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;

void setup()
{
   pinMode(redPin, OUTPUT);
   pinMode(greenPin, OUTPUT);
   pinMode(bluePin, OUTPUT);
}

void loop()
{
```

```
  setColor(255, 0, 0); // red
  delay(1000);
  setColor(0, 255, 0); // green
  delay(1000);
  setColor(0, 0, 255); // blue
  delay(1000);
  setColor(255, 255, 0); // yellow
  delay(1000);
  setColor(80, 0, 80); // purple
  delay(1000);
  setColor(0, 255, 255); // aqua
  delay(1000);
}

void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

코드의 시작부분을 보면 먼저 LED의 각 리드가 아두이노의 어떤 핀에 연결되었는지 알려줍니다.

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
```

다음은 setup함수로 먼저 게시글에서 언급되었듯이 이함수는 아두이노가 리셋되면 딱한번 실행되는 함수입니다. setup함수에서 3개의 핀이 출력으로 사용된다고 설정하였습니다.

```
void setup()
{
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
```

loop함수들 들어가기 전에 제일 마지막에 있는 setColor함수를 살펴보면

```
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

이 함수는 세개의 파라메서를 받는데 이 파라메터는 각각 빨강, 초록, 파랑 LED의 밝기값입니다. 0~255의 값을 받으며, 0은 off 255는 최대값의 밝기를 의미합니다. 그리고 analogWrite 함수를 호출하여 LED의 밝기를 설정합니다.

loop함수를 살펴보면 여러가지색을 만들고 각각1초간 딜레이하였다가 하는 작업을 반복하는 것을 확인 할 수 있습니다. 같은 수준의 밝기를 사용한 것을 확인할 수 있습니다.

```
void loop()
{
  setColor(255, 0, 0); // red
     delay(1000);
  setColor(0, 255, 0); // green
     delay(1000);
  setColor(0, 0, 255); // blue
  delay(1000);
  setColor(255, 255, 0);// yellow
  delay(1000);
  setColor(80, 0, 80); // purple
  delay(1000);
  setColor(0, 255, 255);// aqua
  delay(1000);
}
```

만약 양극이 한개인 RGB LED를 사용한다면 컬러의 밝기값은 다음과 같이 255에서 뺀값을 사용하여야 합니다. : analogWrite(redPin, 255-red); .

16진수 컬러 코드 사용하기

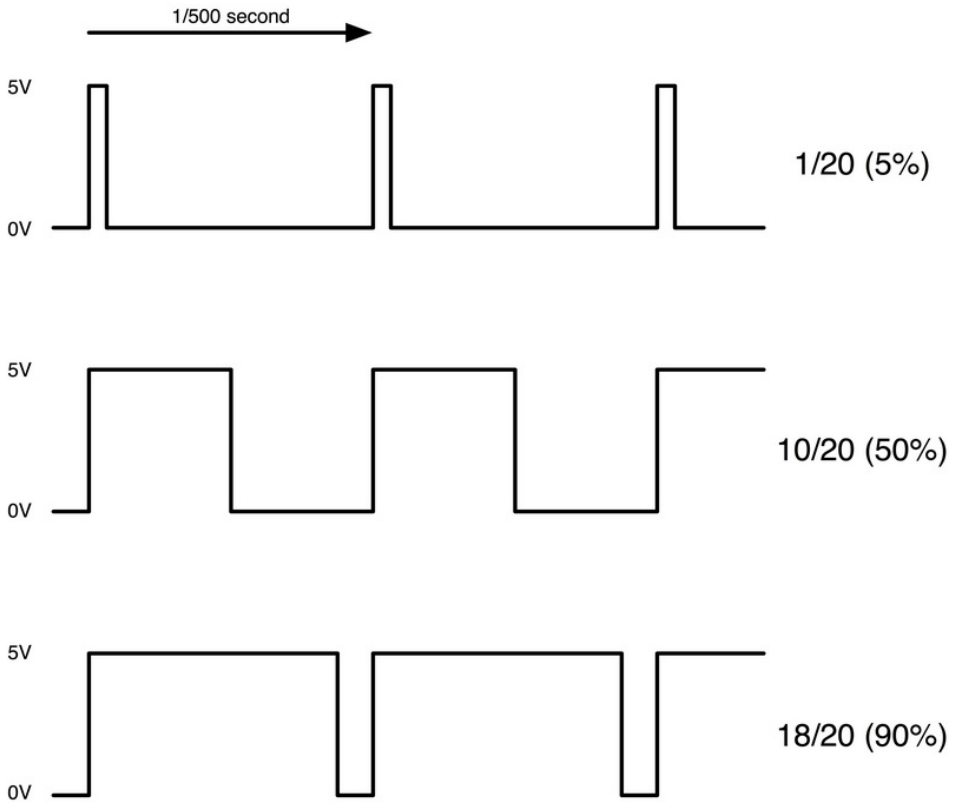
인터넷이나 포토샵과 같은 프로그램을 보면 컬러를 #FF0000과 같은 6개의 16진수로 표기한 것을 본적이 있을 것입니다. 이 6개의 숫자는 처음의 두자리는 빨강, 그다음 두개는 초록, 마지막 두개는 파랑을 의미하는 숫자입니다. 빨강은 #FF0000인데 16진수 FF가 255를 의미하기 때문입니다. 이런 컬러코드방식은 코드별로 색상테이블이 존재하기 때문에 RGB LED에서 원하는 컬러를 디스플레이할 시 사용하기 편리합니다. 아래와 같은 인디고(#4B0082) 색상을 RGB LED에서 만들어 보겠습니다.

인디고의 빨강,초록, 파랑은 16진수로 각각 4B, 00, 82입니다. 이 값들을 아래와 이이 setColor의 파라메터 값으로 넘겨주겠습니다.

setColor(0x4B, 0x0, 0x82); // indigo
파라메터 앞에 붙은 0x는 숫자가 16진수임을 시스템에 알려주는 prefix입니다.

이론 (PWM)

Pulse Width Modulation (PWM)은 전원을 제어하는 기술입니다. LED의 밝기를 제어하기 위해 사용됩니다. 아래의 그림은 아두이노 PWM핀에서 나오는 신호를 보여줍니다.



대략 1/500 초마다 PWM출력은 펄스를 만들어 냅니다. 이 펄스의 길이는 analogWrite 함수에 의해 제어 됩니다. analogWrite(0)은 아무런 펄스를 만들지 않는 것을 의미하고 analogWrite(255)는 펄스를 계속 만들고 있는 것을 의미합니다.

analogWrite에 넘겨주는 값을 지정함으로써 펄스를 만들수 있고, 만약 펄스가 전체 펄스에서 5%만 나온다면 전체 전력에서 5%만 공급되는 것이라고 볼수 있습니다.

실제적으로 LED는 pulse의 주기마다 계속 on/off되고 있지만 사람의 눈은 이런 속도로 깜박이는 것을 인식할수 없고 단지 밝기가 변하고 있는 것만 인식할 수 있습니다.

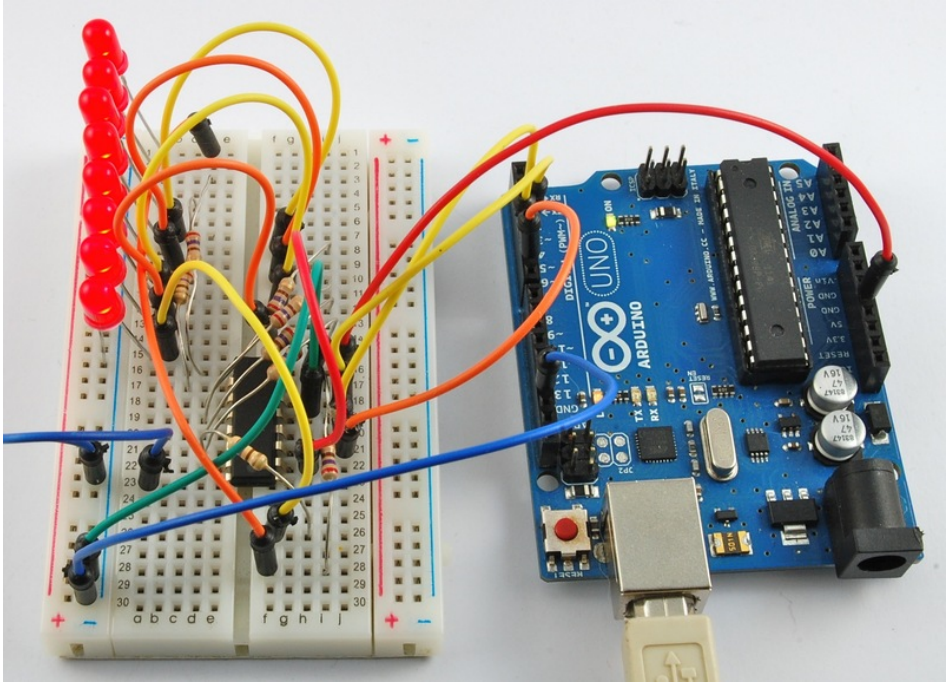
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 4. 쉬프트 레지스터를 이용한 LED제어

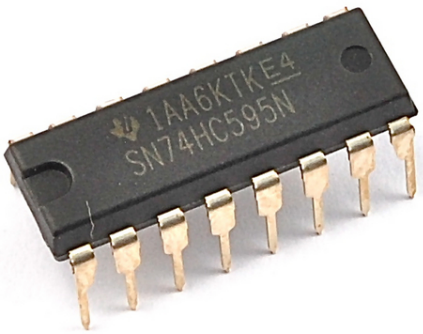
아두이노 강좌

2013/06/13 11:47 http://blog.naver.com/ubicomputing/150169859736

본 게시글에서는 아두이노의 핀 8개를 사용하지 않고도 8개의 LED를 제어하는 방법에 대해서 설명하고자 합니다.



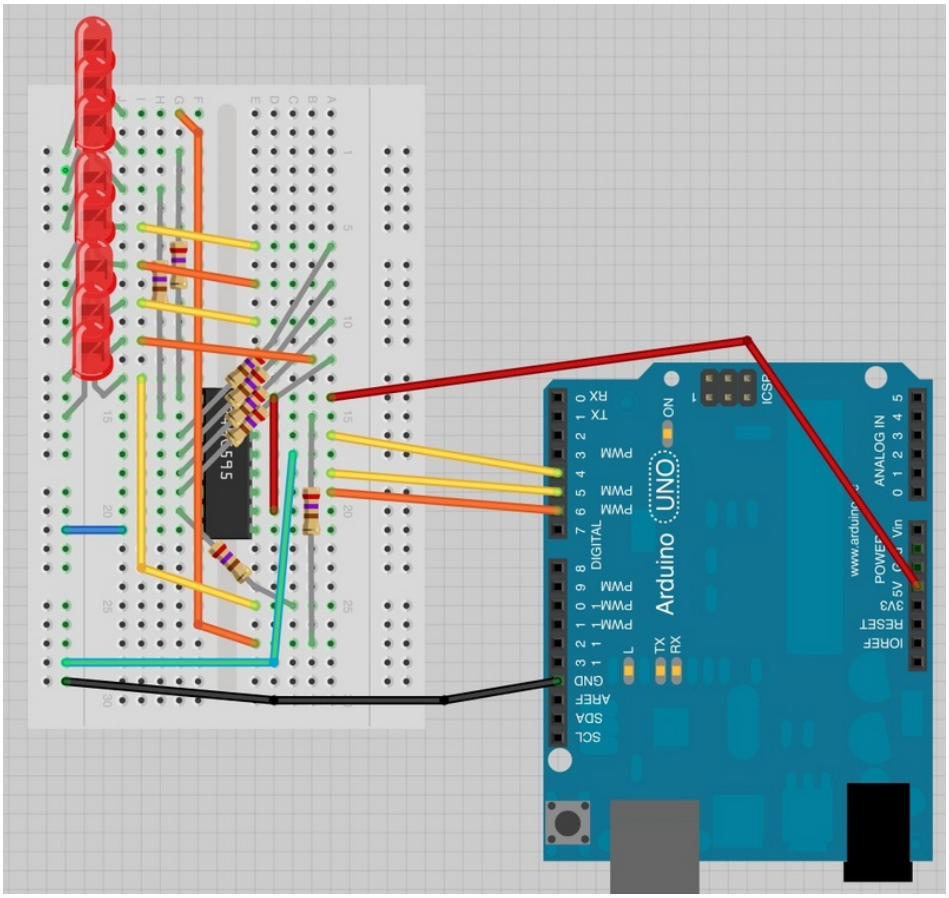
8개의 LED를 아두이노로 제어하려면 8개의 LED를 아두이노의 핀 8개에 직접 연결(저항을 통하여)을 하여도 되겠지만 이는 쓸수 있는 아두이노 핀의 갯수를 많이 제한하게 됩니다. 아두이노에 LED만을 연결하려고 하면 문제가 없겠지만 버튼이나, 센서, 서보 등과 같은 다른 부품을 연결하고자 하면 이는 문제가 될수 있습니다. 그래서 74HC595 parallel to serial 컨버터 칩을 사용하여 이러한 문제를 해결합니다. 이 칩은 8개의 출력을 가지고 있으며 3개의 입력 핀을 가지고 있습니다.



이 칩을 거치면 LED를 동작시키는 것이 약간은 느려지겠지만, 여전히 아주 매우 빠르게 LED를 동작시킬 수 있습니다.

브레드보드 레이아웃

아래의 그림은 8개의 LED와 저항, 74HC595칩과 아두이노를 결선한 모습입니다.



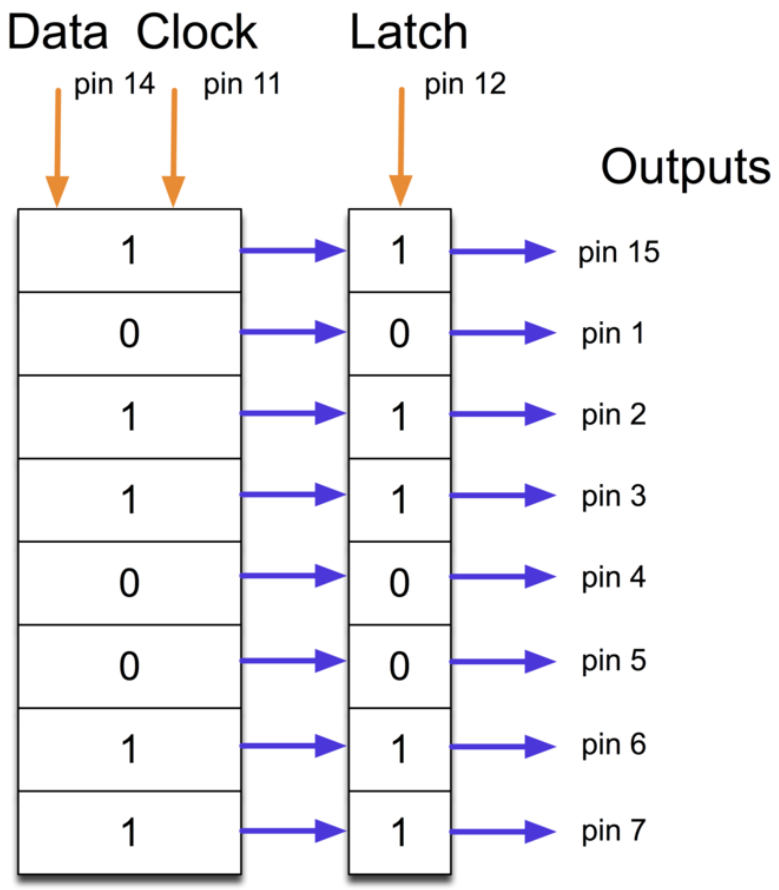
결선을 쉽게 하실려면 74HC595 칩을 먼저 꼽으시는 것이 좋습니다. U짜 형태로 홈이 파져 있는 곳을 브레드보드 위쪽 방향으로 연결하십시오. 칩의 1번 핀이 U홈의 왼쪽에 있습니다.

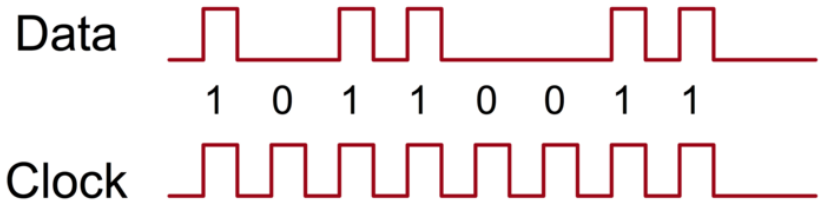
74HC595칩의 좌측에 거의 모든 출력 핀들이 위치하고 있으므로 LED를 쉽게 연결하기 위해서 74HC595 칩 좌측에 LED를 설치하여야 할 것입니다. 저항들을 연결하고 LED를 연결합니다. 저항과 LED가 제대로 연결되었는지 다시 한번 확인합니다.
LED의 긴 다리 양극쪽이 반드시 칩쪽으로 연결되어야 합니다.

점퍼를 위의 그림과 같이 연결합니다. IC칩의 8번핀은 브레드보드의 GND로 연결해야 하는 것을 잊지 마십시오.
올바르게 연결하였다면 아래에 있는 예제 스케치를 아두이노에 올려 테스트를 하여봅니다. LED가 순서대로 점등 되고 모두 점등되면 다시 꺼지게 됩니다.

74HC595 쉬프트 레지스터

코드를 설명하기 전에 74HC595 칩에 대해 간단히 살펴보겠습니다. 이 칩은 쉬프트 레지스터라고 부르는 형태의 제품입니다.





쉬프트 레지스터는 1 혹은 0을 저장할 수 있는 8개의 메모리 공간을 가지고 있습니다. 이 메모리 공간에 1 혹은 0을 넣을려면 칩의 Data와 Clock 핀을 이용하여야 합니다.

클럭핀은 한번에 8개의 펄스를 받습니다. 만약 데이터핀이 high라면 1이 쉬프트 레지스터에 들어가고 그렇지 않다면 0이 들어갑니다. 8개의 모든 펄스가 수신되었으면 Latch핀을 활성화시켜 8개의 값을 Latch register에 복사합니다.
칩에는 OE(output enble) 핀이 있는데 이 핀은 출력을 활성화 하거나 비활성화 할수 있습니다. 아두이노의 PWM 핀을 이 핀에 붙여서 LED의 밝기를 조절할 수 있습니다. 이 핀은 active low이므로 GND에 연결합니다.

아두이노 코드

아두이노는 shiftOut이라는 특별한 함수를 가지고 있습니다. 이 함수는 데이터를 쉬프트 레지스터에 보내기 위해 특별히 디자인되었습니다. 아래는 전체 스케치 코드입니다.

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;

byte leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  leds = 0;
  updateShiftRegister();
  delay(500);
  for (int i = 0; i < 8; i++)
  {
     bitSet(leds, i);
     updateShiftRegister();
     delay(500);
  }
}

void updateShiftRegister()
```

```
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

맨 처음 할 것은 우리가 사용할 세개의 핀을 정의하는 일입니다. 이 핀들은 아두이노 디지털 출력핀으로 쉬프트레지스터의 latch, clock, data핀에 연결됩니다.

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
```

다음으로 leds라는 변수가 정의됩니다. 이 변수는 8개의 LED의 on/off 패턴을 저장하기 위해 사용됩니다. byte 타입의 변수이므로 8비트에 대한 정보를 저장할 수 있어 8개의 LED에 대한 On/Off 값을 저장이 능합니다.

```
byte leds = 0;
```

setup함수는 세개의 이 디지털 출력으로 사용되는 것을 알려줍니다.

```
void setup()
<span style="COLOR: rgb(72,72,76)" class="pln">
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}
</span>
```

loop 함수시작 부분에서 leds변수에 0을 할당함으로써 모든 LED를 off시키는 비트 패턴을 할당하고 updateShiftRegister라는 함수를 호출합니다. 이 함수는 leds변수의 값을 쉬프트 레지스터에 전송하여 모든 LED를 실질적으로 off시킵니다. 이 함수에 대한 용은 나중에 시시 습니다.

loop함수는 0.5초동안 딜레이 하다가 for 문안에서 bitset함수를 이용하여 leds변수의 비트를 셋팅합니다.

그 다음에 updateShiftRegister를 호출하여 leds변수의 비트패턴을 업데이트합니다. 다시 0.5초를 기다린 후 다음 루프를 실행합니다.

```
void loop()
{
  leds = 0;
  updateShiftRegister();
  delay(500);
  for (int i = 0; i < 8; i++)
  {
      bitSet(leds, i);
     updateShiftRegister();
     delay(500);
  }
}
```

updateShiftRegister 함수 내부를 살펴보면 먼저 latch 핀을 low로 만들고 아두이노의 shiftOut함수를 부릅니다. shitOut은 네개의 파라메터를 가지고 있는데 처음 두개는 Data와 Clock으로 사용되는 핀이며, 세번째 파라메터는 데이터 어느쪽부터 시작을 것인지 즉 가장 오른쪽 비트부터 시작할 인지 가장 왼쪽 비트부터 시작할 인지를 정해줍니다. 기서는 가장오른쪽 비트 즉 LSB(Least Significant Bit)부터 시작합니다. .마지막 네번째 파라메터는 실제 데이터로 쉬프트레지스터에 쓰여질 데이터입니다.

```
void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

만약에 LED를 끄고 싶으면 bitSet 함수대신 bitClear함수를 사용하여 비트를 0으로 만들수 있습니다.

밝기 제어

74HC595에는 Output Enable핀이 있습니다. 이 핀은 13번핀이며 브레드 보드상에서 그라운드에 연결되어 있습니다. 이 핀은 스위치와 같이 동작하여 출력을 enable/disable시킬수 있습니다. 이 핀은 active low이기 때문에 enable로 들기 위해서는 그라운드에 연결시켜야 합니다. 만약 5V에 연결된다면 모든 출력은 off가 됩니다.
analogWrite 함수를 이 핀에 사용하면 PWM을 이용하여 LED의 밝기를 조절할수 있습니다.

74HC595의 13번핀을 브레드보드의 그라운드 대신 아두이노의 3번 핀에 연결하여 LED밝기를 조절하여 보십시오.
아래는 밝기를 조절하는 스케치 코드입니다.

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
int outputEnablePin = 3;

byte leds = 0;

void setup()
{
   pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(outputEnablePin, OUTPUT);
}

void loop()
{
  setBrightness(255);
  leds = 0;
  updateShiftRegister();
  delay(500);
  for (int i = 0; i < 8; i++)
  {
     bitSet(leds, i);
```

```
      updateShiftRegister();
      delay(500);
   }
   for (byte b = 255; b > 0; b--)
   {
      setBrightness(b);
      delay(50);
   }
}

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}

void setBrightness(byte brightness) // 0 to 255
{
   analogWrite(outputEnablePin, 255-brightness);
}
```

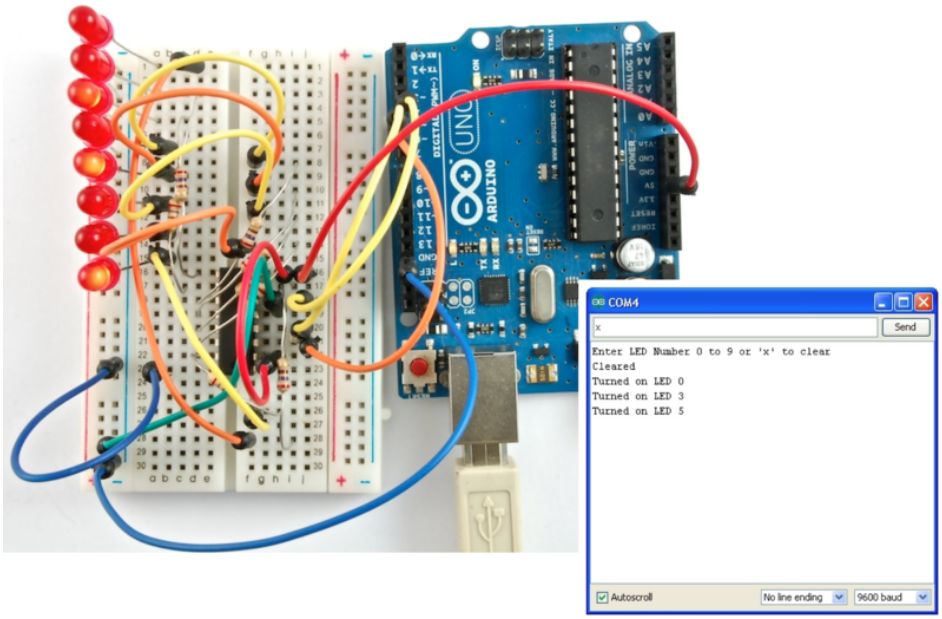
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 5. PC상에서 Serial Monitor로 LED 제어하기

아두이노 강좌

2013/06/13 18:20 http://blog.naver.com/ubicomputing/150169883963

본 게시글에서는 아두이노의 시리얼 모니터 프로그램을 이용하여 컴퓨터에서 LED를 제어하는 방법에 대해 설명하겠습니다. 시리얼 모니터는 아두이노와 컴퓨터사이에서 메세지를 주고 받을 수 있게 하여 줍니다. 디버깅하기 쉽고 아두이노를 컴퓨터에서 제어 할 수 있게 하여 줍니다.



예를 들어 컴퓨터에서 LED를 켜거나 끄라는 명령을 보낼 수 있습니다. 본 게시글은 강좌4에서 사용하였던 아두이노와 브레드보드 레이아웃 셋팅을 다시 사용합니다.

Serial Monitor 프로그램

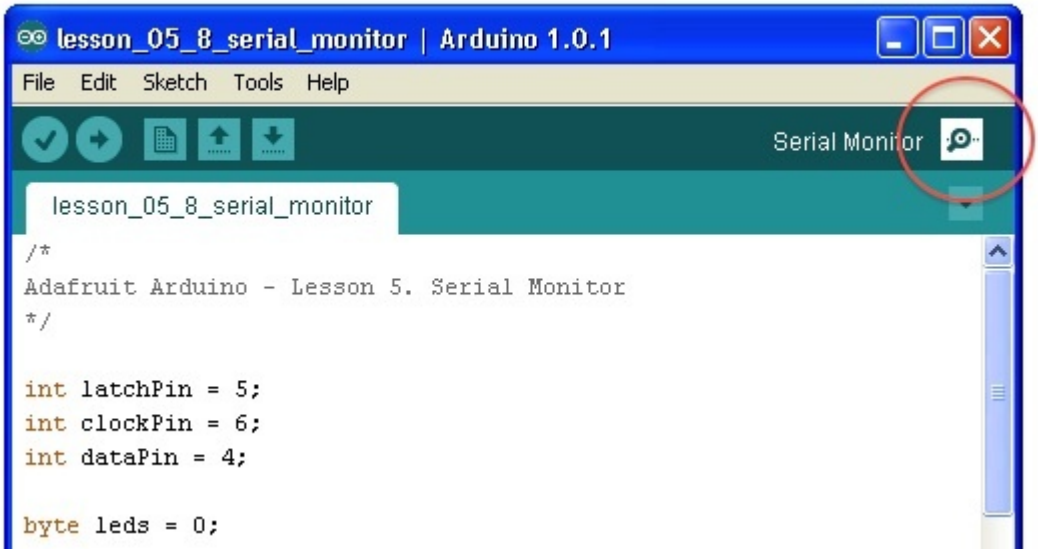
아래는 이번 게시글에 다룰 스케치 코드입니다. 아두이노에 업로드하여 먼저 한번 시험하여 보십시오.

```
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
```

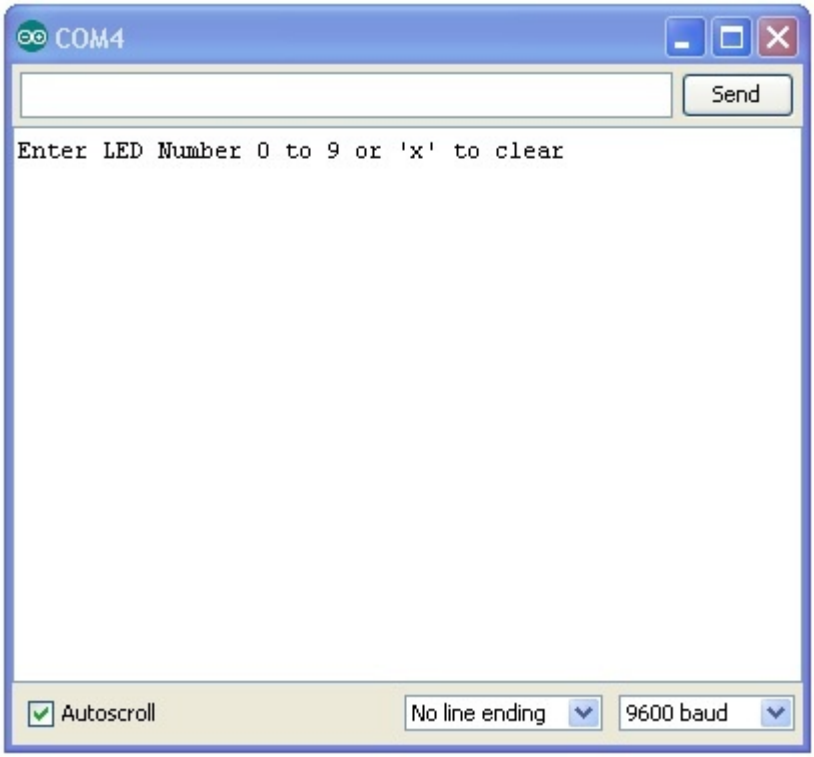
```
byte leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  updateShiftRegister();
  Serial.begin(9600);
  while (! Serial); // Wait untilSerial is ready - Leonardo
  Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}

void loop()
{
  if (Serial.available())
  {
     char ch = Serial.read();
     if (ch >= '0' && ch <= '7')
     {
        int led = ch - '0';
        bitSet(leds, led);
        updateShiftRegister();
        Serial.print("Turned on LED ");
        Serial.println(led);
       }
    if (ch == 'x')
    {
       leds = 0;
       updateShiftRegister();
       Serial.println("Cleared");
    }
  }
}
```

```
void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

스케치를 아두이노에 업로드한 이후에 IDE 오른쪽의 버튼을 눌러 시리얼 모니터를 실행합니다.



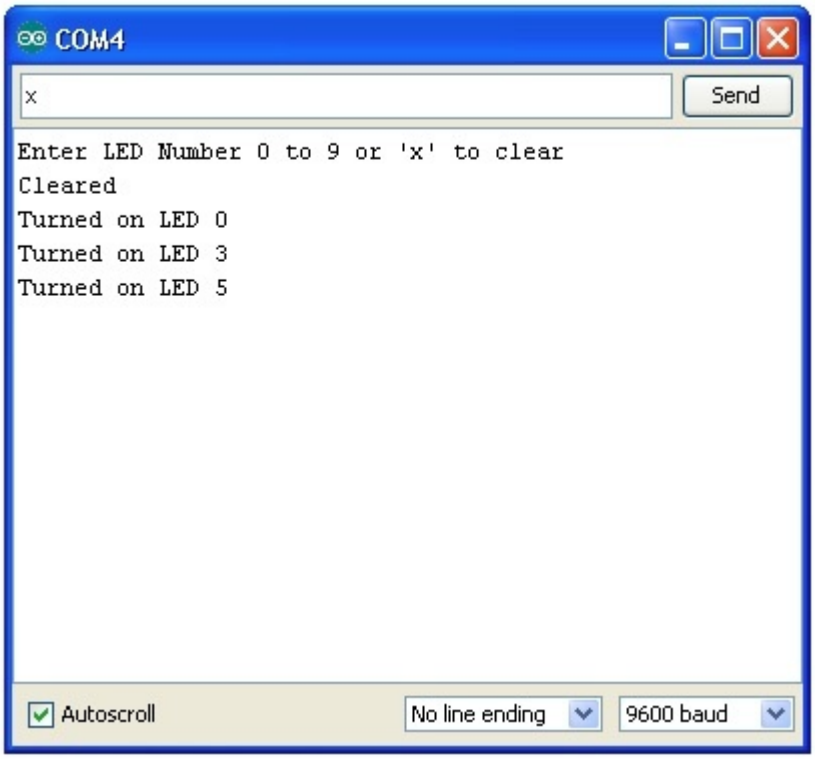
아래와 같은 창이 열립니다.



이 윈도우를 시리얼 모니터라고 부르며 아두이노 IDE의 한 부분입니다. 이 프로그램은 사용자가 컴퓨터에서 아두이노로 메세지를 보낼수 있게 만들어주며, 반대로 아두이노로부터 오는 메세지를 PC로 받을 수 있게 하여 줍니다.

창에 나오는 "Enter LED Number 0 to 9 or 'x' to clear" 메세지는 아두이노가 PC로 보낸 메세지로 x는 모든 LED를 끄며, 숫자를 입력하면 해당 LED를 켜게 됩니다. LED는 0번부터 7번까지 8개 숫자입니다.

예를 들어 x 0 3 5라는 명령을 입력하면 0, 3, 5번 LED가 켜지게 되고 시리얼 모니터창에는 아래와 같이 결과가 표시됩니다.



x를 다시 입력하면 모든 LED가 꺼지게 됩니다.

아두이노 코드

아래의 스케치 코드는 이전 강좌에서 사용되었던 코드이기 때문에 여기서는 이전 강좌에서 설명된 부분을 제외한 부분을 설명하도록 하겠습니다.
먼저 setup함수를 보면 새로운 세줄의 코드가 들어갔습니다.

```
<span style="COLOR: rgb(30,52,123)" class="kwd">
void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  updateShiftRegister();
  Serial.begin(9600);
  while (! Serial); // Wait until Serial is ready - Leonardo
  Serial.println("Enter LED Number 0 to 9 or 'x' to clear");
```

```
}
```

Serial.begin(9600) 명령은 시리얼통신을 시작하게 합니다. 아두이노는 USB연결을 통해 명령어를 보낼수 있게 됩니다. 9600은 시리얼통신 속도로써 숫자가 높을 수록 빠르게 통신합니다. 이 숫자를 변경하려면 시리얼모니터 역시 같은 값을 가지도록 변경하여야 어야 니다.
그 다음의 while문은 시리얼 연결이 준비될때까지 기다립니다. 이 라인은 아두이노 레오나르도를 사용하고 있을 경우에만 필요합니다. 우노는 시리얼 모니터프로그램을 열면 자동으로 아두이노를 리셋합니다.
아래는 핵심이 되는 loop문입니다.

```
void loop()
{
  if (Serial.available())
  {
     char ch = Serial.read();
     if (ch >= '0' && ch <= '7')
     {
        int led = ch - '0';
        bitSet(leds, led);
        updateShiftRegister();
        Serial.print("Turned on LED ");
        Serial.println(led);
     }
     if (ch == 'x')
     {
        leds = 0;
        updateShiftRegister();
        Serial.println("Cleared");
     }
  }
}
```
Serial.available()함수는 아두이노가 시리얼 데이터를 수신하면 true를 리턴합니다. 들어오는 메세지는 버퍼에 보관하고 이 버퍼가 비어있지않으면 Serial.available()은 true를 리턴합니다.

```
char ch = Serial.read();
```

Serial.read()는 버퍼에서 다음 문자를 읽어 ch변수에 할당하고 읽은 문자를 버퍼에서 제거합니다. 시리얼 모니터상에서 알려주는 명령 설명대로 명령을 입력하였으면 읽은 문자는 0~7의 숫자중 나이거나 x 문자일 것입니다.
다음 if 문에서는 읽은 문자가 숫자인지를 확인합니다. 각 문자는 ASCII 코드로 표시되는데 그렇기 때문에 문자를 <=나 >=와 같은 연산자로 비교를 할 수 있습니다.

int led = ch - '0';
이 코드는 문자에 대해 산술연산을 수행하는 라인입니다. 만약 사용자가 시리얼 모니터에 0을 입력했으면 '0' -'0' = 0이 되어 숫자 0을 계산해 낼수 있고 만약 7을 입력했으면 '7' -'0' = 7이되어 7을 계산할 수 있습니다. 실제적으로는 아스키 코드값이 연산되는 것입니다.
이제 켜고자 하는 LED의 번호를 알았으니 아래의 코드와 같이 leds변수에 비트를 셋팅하고 쉬프트 레지스터를 업데이트 합니다.

bitSet(leds, led);
updateShiftRegister();
다음 두개의 라인은 확인 메세지를 시리얼 모니터에 보내는 내용입니다.
Serial.print("Turned on LED ");
Serial.println(led);
첫번째 줄은 Serial.println대신 Serial.print를 이용하였습니다. 이 두함수의 차이는 Serial.print는 문자열을 출력하고 새 라인을 시작하지 않는 다는 점이며 Serial.println은 문자열을 출력하고 다음주에 새 라인을 시작하는다는 점입니다.먼저 Serial.print를 이용하여 Turned on LED를 프린트 한후 LED번호를 찍게 됩니다.
그 다음 if문에서는 들어온 문자가 x인지 확인을 합니다.

if (ch == 'x')
{
  leds = 0;
  updateShiftRegister();
  Serial.println("Cleared");
}

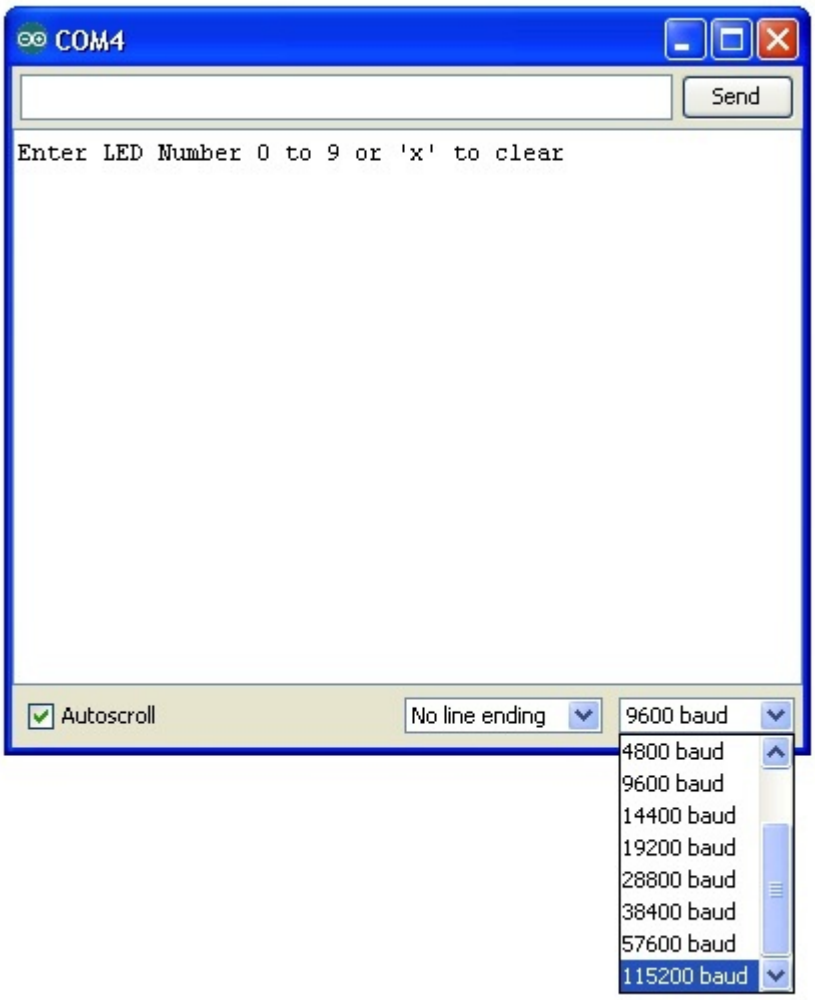
만약 x가 맞다면 모든 LED를 끄고 확인 메세지를 보냅니다.

시도해볼 다른 것들

위의 예제에서 시리얼 모니터상에 문자를 입력할때 문자를 분리하여 입력을 하였지만 아래와 같이 다 붙여서 명령을 주어도 잘 동작합니다. 아래의 명령을 넣고 send버튼을 눌러보십시오.

x0246

다음으로는 스케치에서 통신속도를 9600이 아닌 115200으로 변경하고 업로드 후에 시리얼 모니터 프로그램에서 아래와 같이 속도를 변경하여 테스트를 하여 봅니다. 아두이노와 PC가 더 빠른 속도로 통신을 하게 됩니다.



속도를 높여도 잘 동작하는 것을 확인 할 수 있을 겁니다. 만약 스케치와 시리얼 모니터의 속도가 맞지 않는 다면 화면에는 이상한 문자들이 표시될 것입니다.
시리얼 모니터 프로그램은 스케치 코드를 디버깅하는데도 아주 유용합니다. 스케치 코드가 원하는대로 동작하지 않는다면 Serial.println() 명령을 이용하여 디버깅 메세지를 넣고 프로그램이 어떻게 동작하는지 시리얼 모니터 프로그램으로 살펴볼 수 있을 것입니다.

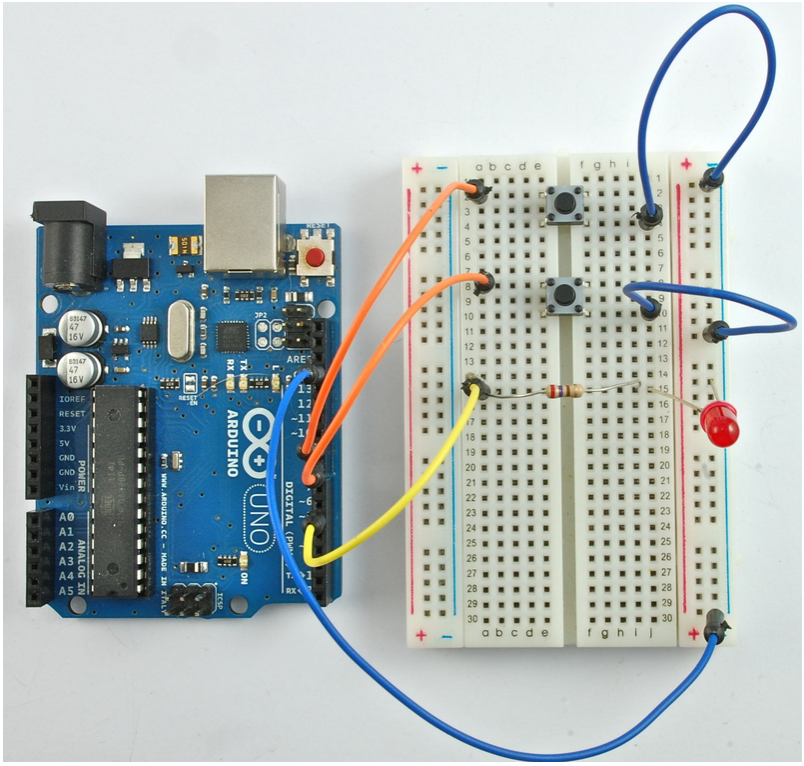
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 6. 푸쉬버튼을 이용한 디지털 입력하기

아두이노 강좌

2013/06/13 21:35 http://blog.naver.com/ubicomputing/150169895029

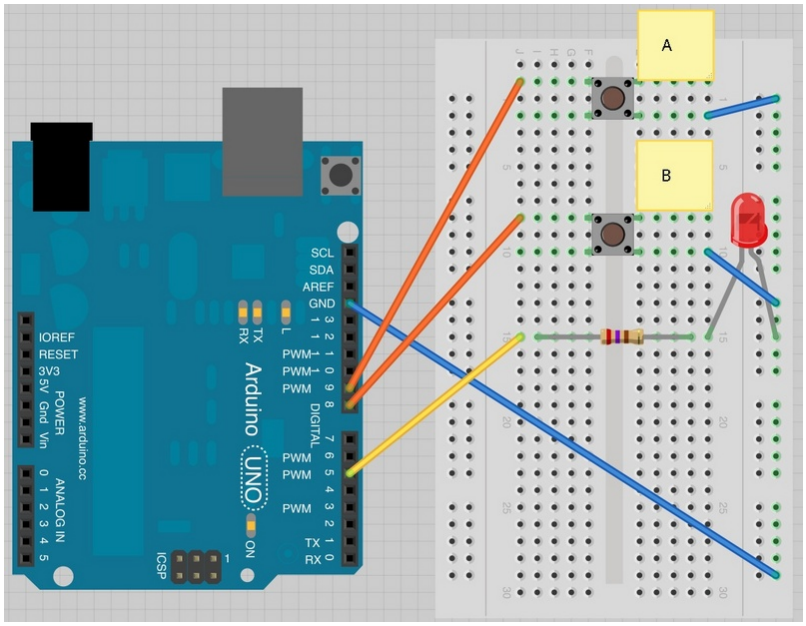
본 게시물에서는 푸시 버튼을 이용하여 LED를 켜고 끄는 방법에 대해 살펴보도록 하겠습니다.



맨위에 있는 푸쉬 버튼을 누르면 LED가 켜지고 다른 버튼을 누르면 LED가 꺼지게 될 것입니다.

브레드보드 레이아웃

스위치 버튼의 핀은 스위치 반대편에 위치하여 있습니다.



LED를 연결할 때는 극성을 주의하여 연결하십시오. 짧은 쪽이 음극이며 오른쪽에 위치 시켜야 합니다.

아두이노 코드

아래의 스케치를 아두이노에 로드시키고 맨 위에 위치한 푸쉬버튼을 눌러보십시오. LED가 켜지는것을 확인한 후 다른 버튼을 눌러 LED를 끄십시오.

```
int ledPin = 5;
int buttonApin = 9;
int buttonBpin = 8;

byte leds = 0;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
}
```

```
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
      digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
      digitalWrite(ledPin, LOW);
  }
}
```

코드의 첫부분에서는 세개의 핀을 위한 변수를 정의하였습니다. ledPin은 출력 핀이며 buttonApin은 제일 위에 있는 버튼, buttonBpin은 다른 스위치를 의미합니다.

setup함수에서 ledPin을 디지털 출력모드로 설정하였습니다. buttonApin, buttonBpin은 입력으로 설정되었는데 INPUT_PULLUP이 파라메터로 쓰였습니다.

```
pinMode(buttonApin, INPUT_PULLUP);
pinMode(buttonBpin, INPUT_PULLUP);
```

INPUT_PULLUP 핀모드는 핀이 입력으로 사용될 것이지만 아무것도 연결되지 않았다면 HIGH 상태로 pullup 시켜두라는 것을 의미합니다. 다시 말하면 입력핀의 기본 상태는 HIGH입니다. 그리고 버튼을 눌르면 상태는 LOW가 되어야 합니다.

그래서 스위치들이 GND에 연결되어 있습니다. 스위치가 눌리면 스위치는 입력핀을 그라운드와 연결시키게 되고 입력핀은 더이상 HIGH상태가 될수 없게 됩니다.

입력은 보통 HIGH이고 버튼이 눌리면 LOW로 되는 로직을 loop 함수에서 처리합니다.

```
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
      digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
      {
```

```
    digitalWrite(ledPin, LOW);
  }
}
```

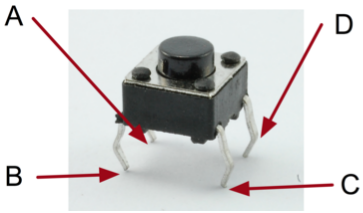
루프 함수 안에는 두개의 if문이 있고 각각 버튼 두개의 상태를 검사합니다.

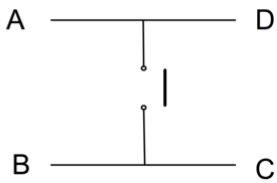
버튼이 눌린다면 눌린쪽에 연결된 입력핀은 LOW가 되므로 버튼A가 눌려 LOW상태가 된다면 digitalWrite 함수가 ledpin을 HIGH로 만들어 LED를 켜게 됩니다.
비슷하게 버튼 B가 눌린다면 LED를 끄게 됩니다.

푸쉬 스위치

스위치는 간단한 부품으로 버튼을 누르면 두개의 접촉부분을 연결시켜 전기를 흐르게 만드는 부품입니다.

여기서 사용되는 택틀 스위치는 4개의 다리를 가지고 있어 무엇이 무엇인지 약간 헷갈립니다.





실제로 버튼에는 두개의 전기적 연결만이 있습니다. 스위치 케이스 안에는 핀B와 핀C가 서로 연결되어 있고 핀A와 핀D가 서로 연결되어 있습니다.

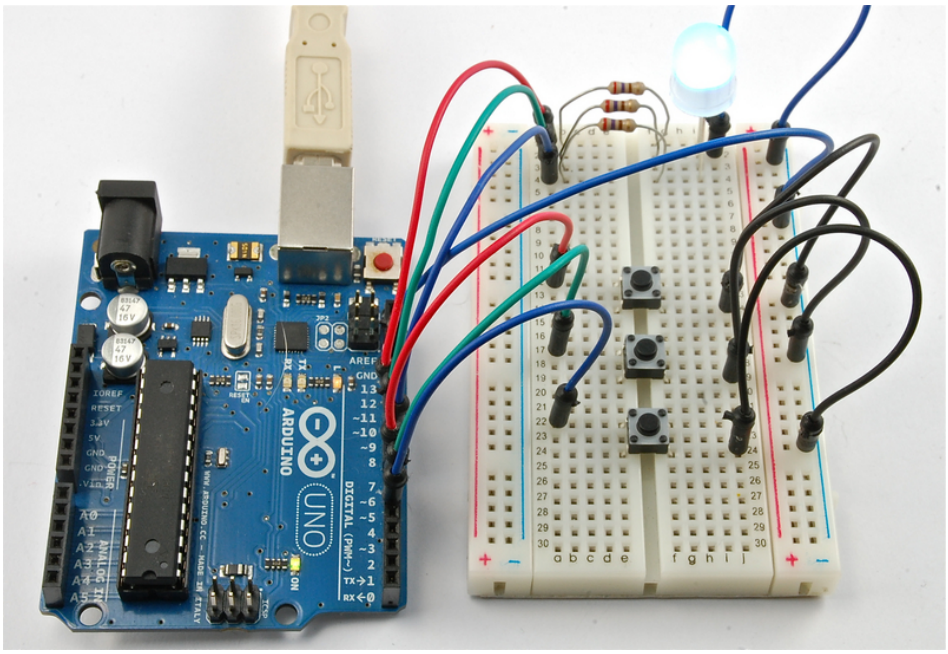
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 7. 버튼을 이용하여 RGB LED 색상 제어하기

아두이노 강좌

2013/06/14 10:29 http://blog.naver.com/ubicomputing/150169919612

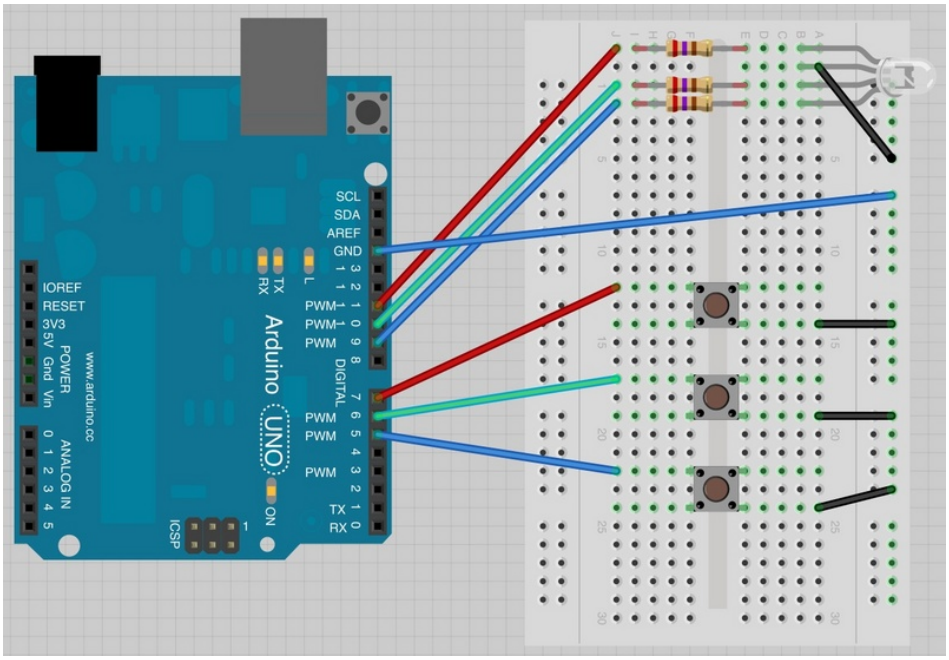
이번 게시글에서는 6번 강좌에서 셋업하였던 세개의 푸쉬버튼을 이용하여 RGB LED의 색깔을 제어하는 실험을 진행하도록 하겠습니다.



브레드보드 레이아웃

브레드보드 레이아웃은 아래와 같습니다. RGB LED의 가장 긴다리는 2번째 줄에 연결하여 GND에 연결되어 있는 것을 주의하십시오

본 레이아웃은 RGB LED가 공통 음극을 가진 타입을 기준으로 작성되었습니다. 만약 공통 양극을 가진 RGB LED를 가지고 있다면 가장 긴 다리를 +5V에 GND대신 연결하십시오.



아두이노 코드

아래의 코드를 아두이노에 로드합니다.

시작할때, 모든 LED는 꺼져 있게 되며 버튼을 누르고 있으면 LED는 점점 밝게 빛나게 됩니다. 브레드보드상의 제일 상단에 있는 버튼은 빨강, 중간은 초록, 제일 밑의 버튼은 파랑색을 조절하는 버튼입니다.

버튼들을 서로 눌러서 색이 어떻게 섞이는 지 확인하여 보십시오. 다시 시작하고 싶으시면 아두이노의 리셋 버튼을 눌러 초기화 시키십시오. USB커넥터 옆에 있는 버튼입니다.

```
int redLEDPin = 11;
int greenLEDPin = 10;
int blueLEDPin = 9;

int redSwitchPin = 7;
int greenSwitchPin = 6;
int blueSwitchPin = 5;

int red = 0;
int blue = 0;
```

```
int green = 0;

void setup()
{
  pinMode(redLEDPin, OUTPUT);
  pinMode(greenLEDPin, OUTPUT);
  pinMode(blueLEDPin, OUTPUT);
  pinMode(redSwitchPin, INPUT_PULLUP);
  pinMode(greenSwitchPin, INPUT_PULLUP);
  pinMode(blueSwitchPin, INPUT_PULLUP);
}

void loop()
{
  if (digitalRead(redSwitchPin) == LOW)
  {
      red ++;
      if (red > 255) red = 0;
  }
  if (digitalRead(greenSwitchPin) == LOW)
  {
      green ++;
      if (green > 255) green = 0;
  }
  if (digitalRead(blueSwitchPin) == LOW)
  {
      blue ++;
      if (blue > 255) blue = 0;
    }
  analogWrite(redLEDPin, red);
  analogWrite(greenLEDPin, green);
  analogWrite(blueLEDPin, blue);
  delay(10);
}
```

스케치를 보면 LED를 제어하기 위한 세개의 출력 핀이 있고, 이 출력 핀은 PWM핀으로 LED의 각 컬러로 들어가는 전력을 제어할 있습니다.

또 다른 세개의 핀이 필요한데 이 핀들은 버튼을 위하여 사용이 됩니다. setup함수에서 입력으로 설정되었으며 HIGH상태로 풀업되어, 만약 버튼이 눌리게 된다면 LOW상태로 변화하게 됩니다.

핀이 정의된 다음에는 red, green, blue라는 변수들이 나오게 됩니다.

```
int red = 0;
int blue = 0;
int green = 0;
```

이 변수들은 LED의 각 RGB 채널의 빛의 세기를 저장하는 값입니다. 만약 red가 0이라면 LED의 빨강색부분은 꺼지게 되며, 255라면 최고치의 빨강색을 밝게 표현합니다.

loop함수는 두 부분으로 나뉘어 있는데 첫번째 부분은 버튼을 체크해서 red, green, blue변수에 버튼의 상태에 따라 필요한 변화를 주는 부분입니다. 예를 들어 빨강색용 버튼을 체크하는 부분은 아래와 같습니다.

```
if (digitalRead(redSwitchPin) == LOW)
{
  red ++;
  if (red > 255) red = 0;
}
```

digitalRead함수를 실행하게 되고 red 핀이 LOW상태인것을 발견하게 되면 이것은 버튼이 눌렸다는 것을 의미합니다. 그래서 red변수에 1을 더하게 됩니다.

PWM용으로 쓸수 있는 최대 값은 255이기 때문에 다음 라인에서 red가 255를 넘는지를 확인 한 후 넘었다면 0으로 다시 설정합니다.

두번째 부분은 analogWrite함수를 수행하는 부분으로 각각의 LED의 색을 설정하게 됩니다.

```
analogWrite(redLEDPin, red);
analogWrite(greenLEDPin, green);
analogWrite(blueLEDPin, blue);
```

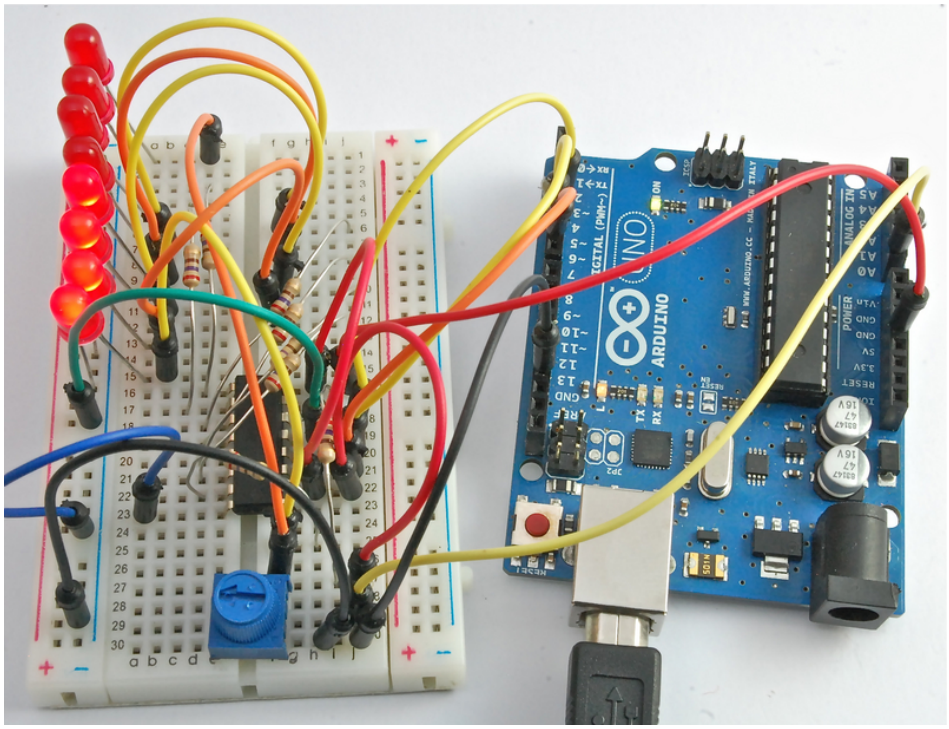
마지막으로 loop의 마지막에는 짧은 딜레이를 주어 색의 변화를 조금 늦추어 줍니다. 이 딜레이가 다다면 푸쉬버튼의 물리적인 노이즈로 색이 랜덤하게 변할 수 있습니다.

[아두이노 강좌] 8. 가변저항을 이용하여 아날로그 입력으로 LED 밝기 조절하기

아두이노 강좌

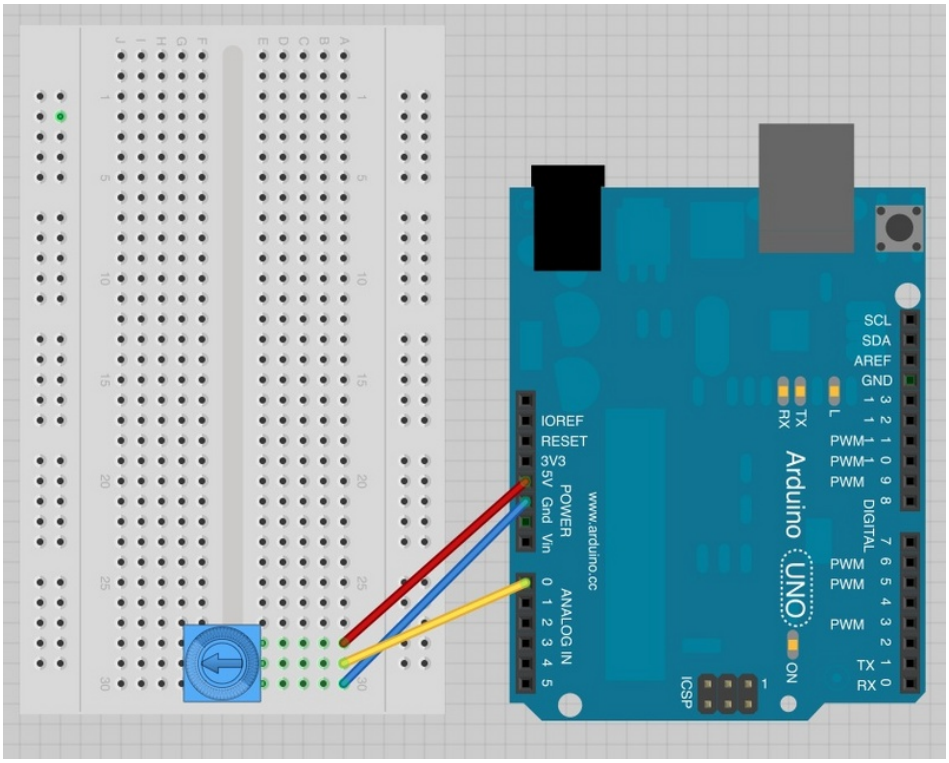
2013/06/14 12:54 http://blog.naver.com/ubicomputing/150169927862

본 게시물에서는 시리얼 모니터 프로그램을 이용하여 아날로그 입력값을 디스플레이하는 것을 설명한 후 가변저항을 사용하여 켜지는 LED 갯수를 제어하는 실험을 할 것입니다.



실험

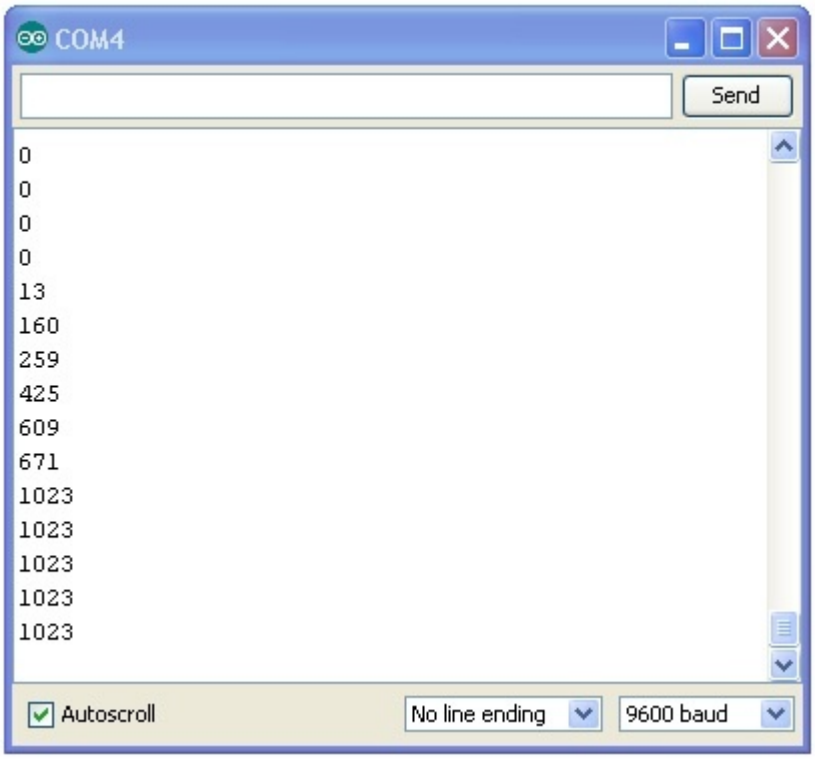
먼저 포텐셔미터라고 알려진 가변저항에 대한 간단한 실험을 하여 보겠습니다. 브레드보드에 아래와 같이 연결을 하십시오.



아래의 스케치 파일을 아두이노에 업로드 하여보십시오.

```
int potPin = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int reading = analogRead(potPin);
  Serial.println(reading);
  delay(500);
}
```

업로드가 끝났으면 시리얼 모니터를 열어 보십시오. 화면 숫자들이 나타나는 것을 볼수 있을 것입니다.



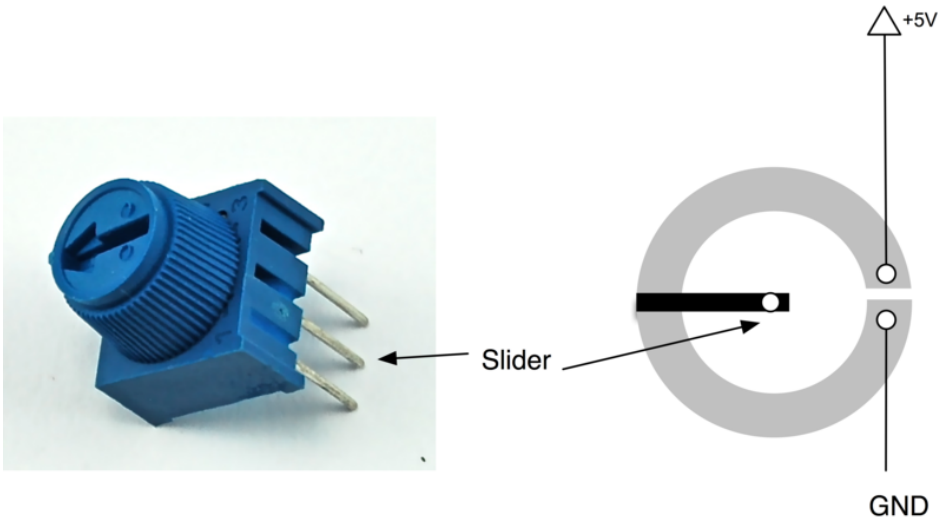
가변저항에 달려 있는 손잡이를 돌려 보면 숫자들이 0~1023사이에서 변하는 것을 확인 할 수 있을 겁니다.

시리얼 모니터는 아두이노가 아래의 스케치코드를 실행하면서 A0로부터 읽어 들인 아날로그 값을 표시하고 있는 것입니다.

int reading = analogRead(potPin);
A0의 전압은 0에서 1023의 숫자로 변환되어 전송이 됩니다.

가변저항 (포텐셔미터)

가변저항은 A0핀의 전압레벨을 변경시킬 수 있습니다. 이 변경된 전압은 스케치 코드에 의해 0~1023의 값으로 변환이 됩니다.

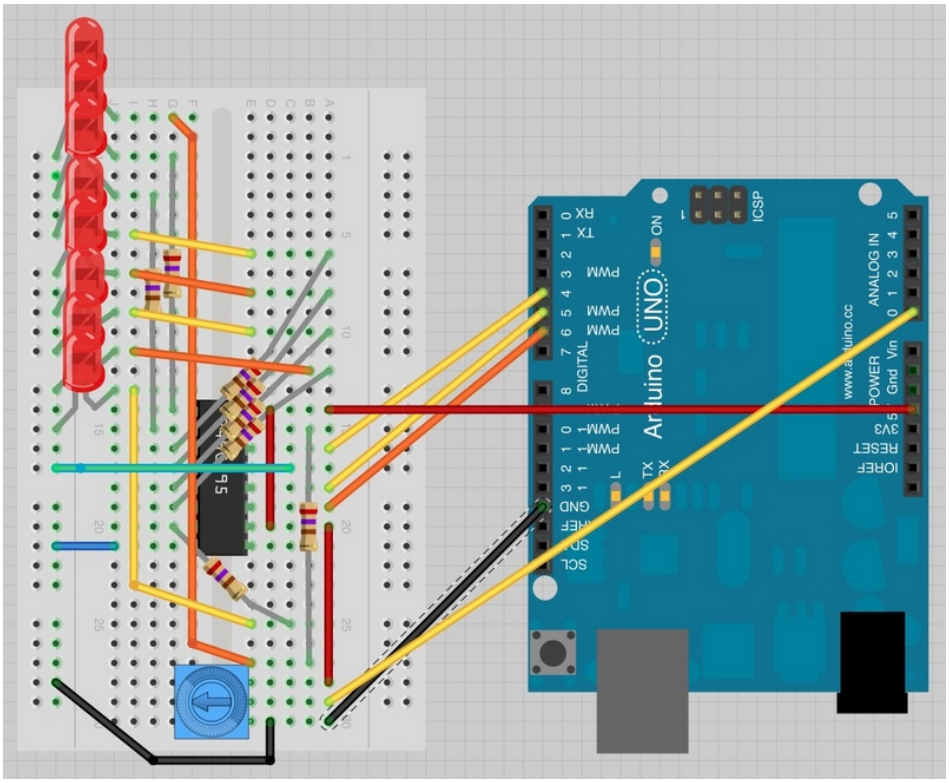


포텐셔미터는 둥글게 생긴 트랙이 있는데 저항과 같은 역활을 합니다. 포텐셔미터가 일반 저항과 다른점은 슬라이더라 불리는 중간 연결입니다. 이 슬라이더는 포텐셔미터를 돌리면 같이 돌아가는데 5V쪽으로 돌리면 점차 5V가 되고, GND쪽으로 돌리면 0V가 됩니다.

브레드보드 레이아웃

그러면 이제 좀 더 재미있는 프로젝트를 진행하여 보겠습니다. 가변저항을 요리조리 돌리면 켜지는 LED의 갯수를 변경시키는 것입니다.

아래와 같이 부품을 셋업합니다.



아두이노 코드

아래의 스케치를 아두이노에 업로드합니다.

```
int potPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;

int leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
 }
```

```
void loop()
{
  int reading = analogRead(potPin);
  int numLEDSLit = reading / 114; //1023 / 9
  leds = 0;
  for (int i = 0; i < numLEDSLit; i++)
  {
      bitSet(leds, i);
  }
  updateShiftRegister();
}

void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

코드가 상당부분 4번 강좌와 비슷합니다. 만약 코드가 잘 이해가 안되시는 부분이 있다면 이전의 강좌를 참고 바랍니다.

이 스케치의 핵심부분은 아날로그 핀을 정의하는 부분인데 이 아날로그 핀은 포센셔미터의 슬라이더와 연결되게 됩니다.

```
int potPin = 0;
```

핀을 아날로그 입력으로 설정하기 위해 setup함수에 아무것도 넣지 않았다는 점을 유의하여 보십시오

메인 루프에서 아래와 같이 아날로그 값을 읽게 됩니다.

```
int reading = analogRead(potPin);
```

읽은 아날로그 값은 0에서 1023 사이의 값입니다. 이 값들을 0에서 8의 9개의 LED에 해당하는 숫자로 변경하여 주어야 합니다. 그래서 읽은 값을 114로 나누어 줍니다.

```
int numLEDSLit = reading / 114;
```

이제 LED를 원하는 숫자 만큼 켜주기 위해서 for loop안에서는 0에서 numLEDSLit까지 숫자를 카운트하면서 비트를 셋팅하게 됩니다.

```
leds = 0;
for (int i = 0; i < numLEDSLit; i++)
{
  bitSet(leds, i);
}
```

updateShiftRegister함수를 호출하여 shift register를 업데이트합니다.

```
updateShiftRegister();
```

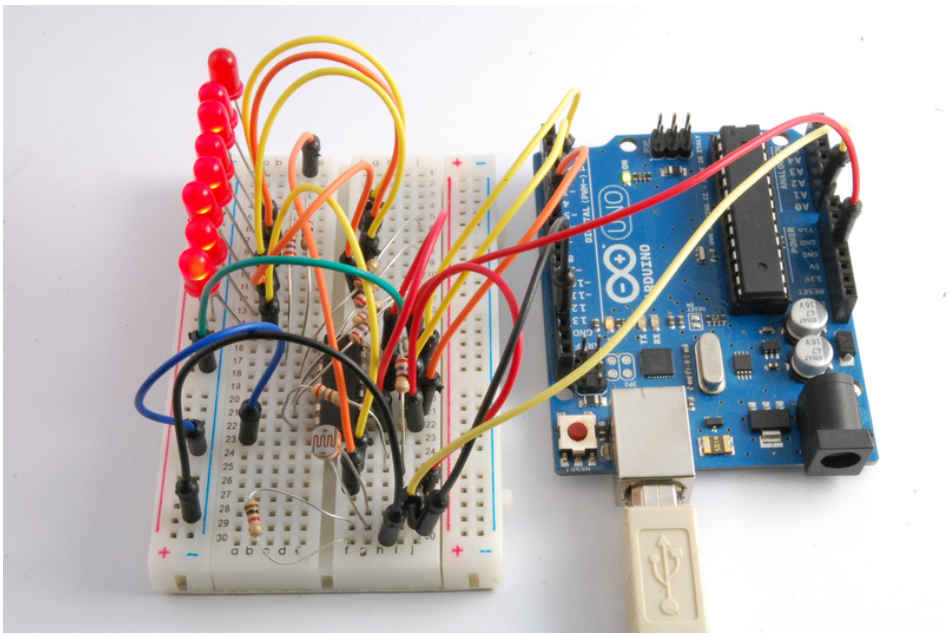
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 9. 광센서로 조도 측정하기

아두이노 강좌

2013/06/14 15:10 http://blog.naver.com/ubicomputing/150169936270

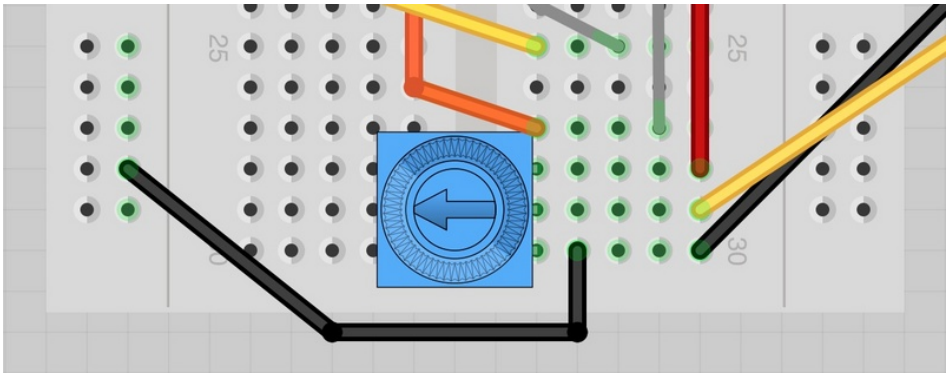
본 게시글에서는 아날로그 입력을 사용하여 어떻게 광도를 측정하는지 알아보도록 하겠습니다. 이전 강좌 8번에 서 사용된 회로를 응용하겠습니다.



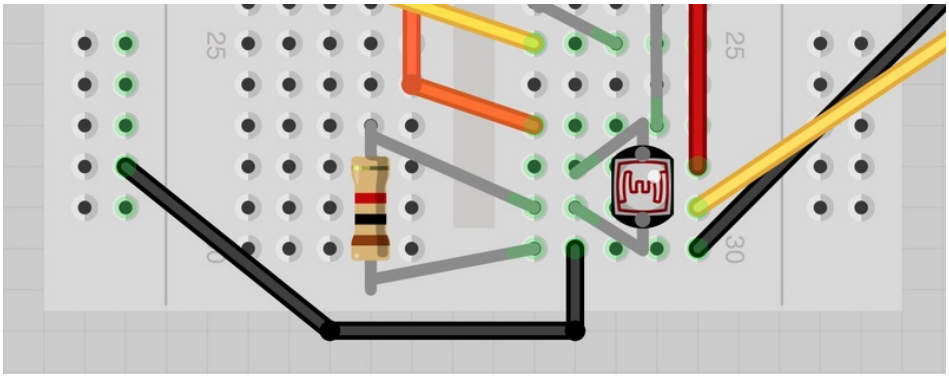
사진을 보면 포토셀 센서는 브레드보드 맨 하단에 위치하고 있습니다.

브레드보드 레이아웃

본 게시글에서 사용하는 브레드보드 레이아웃은 강좌 8에서 사용하였던 레이아웃과 동일합니다. 다만 포텐셔미터 대신 LDR과 1K오옴 저항이 사용됩니다.

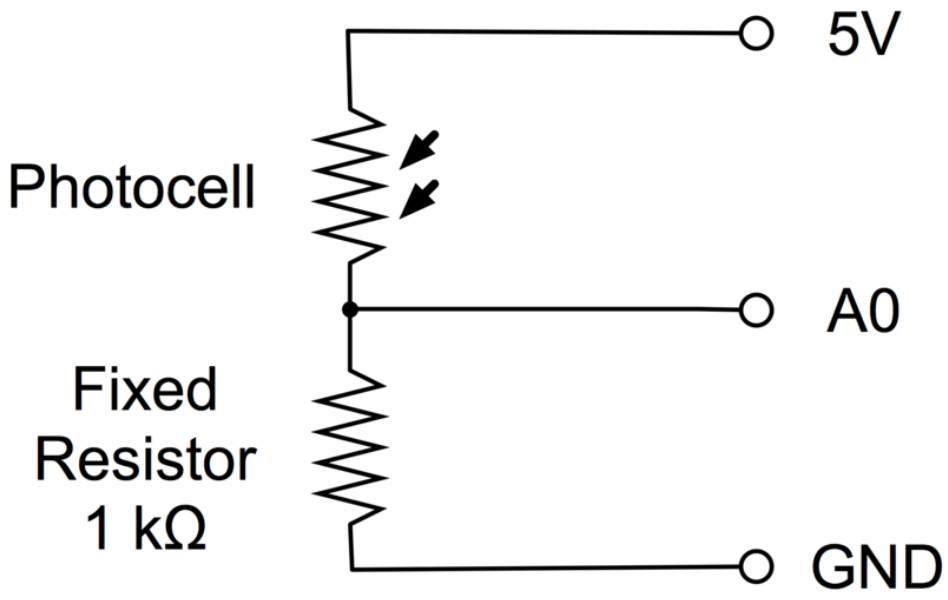


포센셔미터를 제거하고 아래와 같이 포토셀과 저항을 셋업하십시오.



포토셀

포토셀은 LDR(Light Dependent Resistor)라고 불리며 그 이름대로 빛의 세기에 따라 저항값이 달라지는 소자입니다. 실험에 사용되는 포토셀은 어둠속에서 50K오옴 밝은 빛에서 500k오옴을 가지는 저항입니다. 계속적으로 변하는 저항을 아두이노의 아날로그 입력에서 측정하려면 전압으로 바꾸어 주어야 합니다. 가장 간단한 방법은 포토셀을 고정된 값을 가진 저항과 결합하는 방법입니다.



저항과 포토셀을 합쳐 놓으면 일종의 포센셔미터처럼 동작합니다. 빛이 매우 밝으면 고정값 저항에 비했을 때 포토셀의 저항이 매우 낮아 지고, 이는 마치 포텐셔미터를 최대치로 돌렸을 때와 동일한 효과를 냅니다.

포토셀이 어두운 곳에 있을 경우 저항은 고정값 저항보다 커지게 되고 이는 포센셔미터가 그라운드로 향하는 것과 동일한 효과를 내게 됩니다.

아래에 있는 스케치를 아두이노에 업로드하고 포토셀을 어둡게 하거나 밝게 하여 보십시오.

아두이노 코드

밝기에 따라서 8개의 LED를 켜는 코드입니다. 강좌 8에서 사용한 코드도 작동하긴 하지만, 8개의 LED를 모두 켜기 위해서 필요한 밝기의 정도는 아직 모릅니다. 이것은 고정된 저항값때문인데 그래서 포토셀 저항이 얼마나 떨어지는지 상관없이 그것을 오프셋하기 위한 1k오옴의 고정저항이 있다는 사실에 대한 보충작업을 해줄 필요가 있습니다.

아래는 강좌 8에서 사용된 코드를 약간 수정한 버전입니다.

```
int lightPin = 0;
int latchPin = 5;
int clockPin = 6;
int dataPin = 4;
```

```
int leds = 0;

void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop()
{
  int reading = analogRead(lightPin);
  int numLEDSLit = reading / 57; //1023 / 9 / 2
  if (numLEDSLit > 8) numLEDSLit = 8;
  leds = 0;
  for (int i = 0; i < numLEDSLit; i++)
  {
      bitSet(leds, i);
  }
  updateShiftRegister();
}

void updateShiftRegister()
{
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, leds);
  digitalWrite(latchPin, HIGH);
}
```

가장 먼저, 기존의 potPin은 lightPin으로 이름이 바뀌었습니다. 포텐셔미터 대신 포토셀이 사용되니까요.

그 다음으로는 얼마나 많은 LED가 켜지는지를 계산하는 부분이 바뀌었습니다.

```
int numLEDSLit = reading / 57; // all LEDs lit at 1k
```

기존에 114로 나누는 부분을 114의 절반인 57로 나누도록 바꾸었습니다. 이것은 1K오옴 저항을 고려하여 이렇게 변경한 것입니다. 만약 포토셀이 고정저항과 같이 1K오옴의 저항을 가진다면 아날로그 입력값은 1023/2 = 511이 될 것이며 57로 나누면 9구간이 떨어지게 됩니다. 이 수식은 모든 LED를 켤수 있게 합니다.

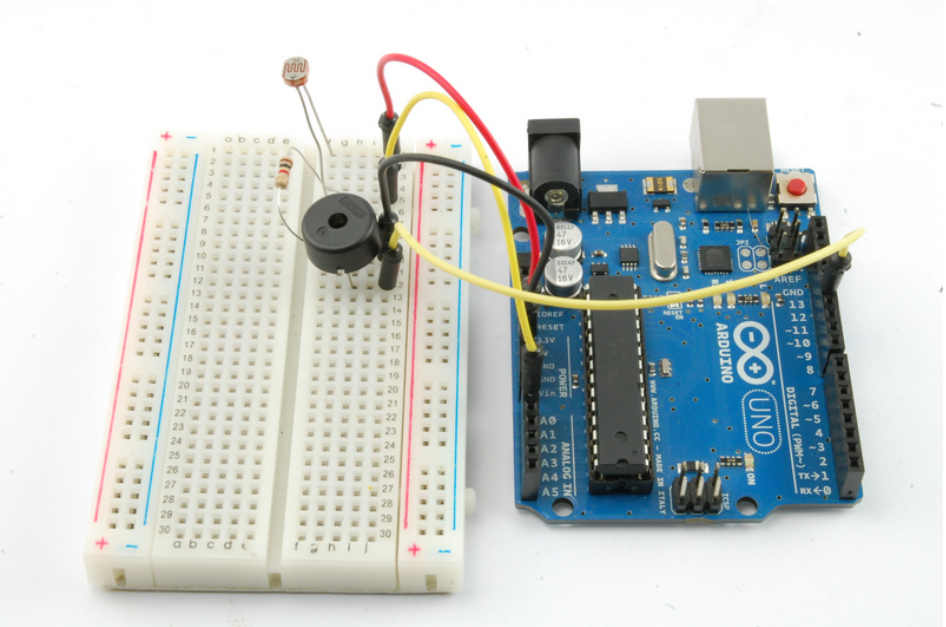
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 10. 소리 만들어 음악 연주하기

아두이노 강좌

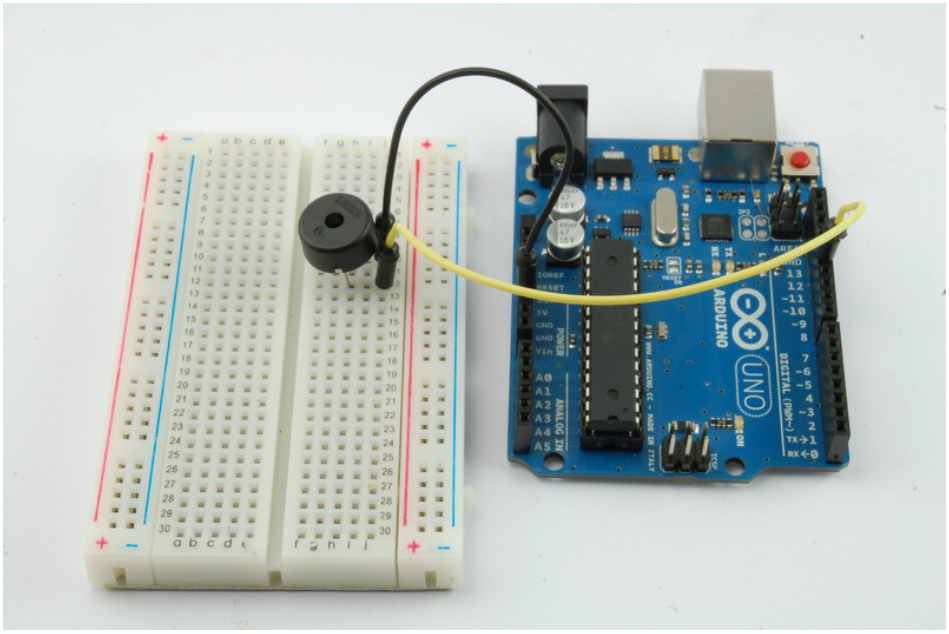
2013/06/17 11:24 http://blog.naver.com/ubicomputing/150170099450

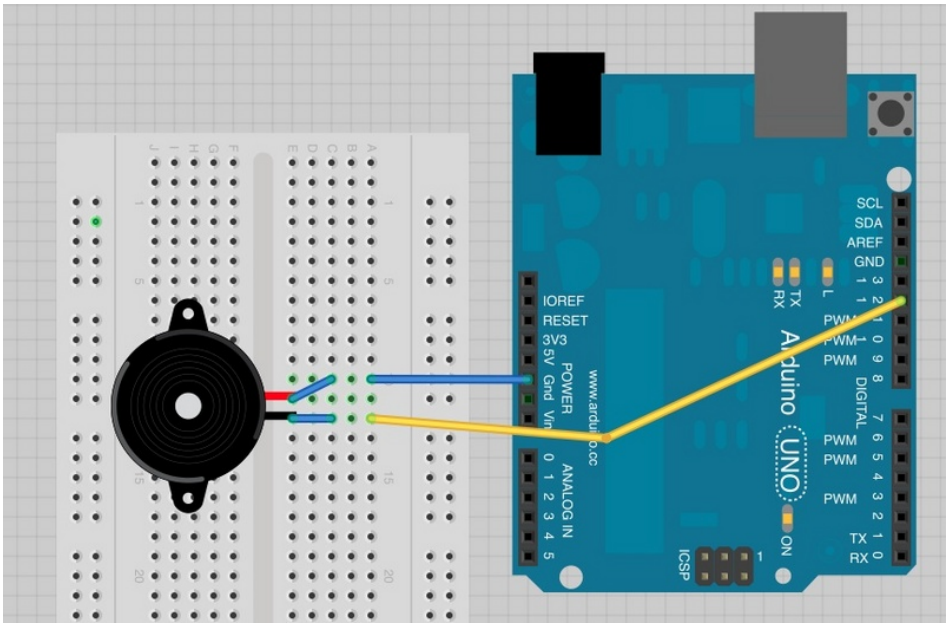
이번 게시글에서는 아두이노를 가지고 소리를 만들어 내는 법에 대해 살펴보 겠습니다. 먼저 아두이노가 음계를 연주하도록 만든 다음에 이것을 포토셀과 결합하여 조도에 따라 음을 연주하도록 만들어 보겠습니다.



음계 연주하기

음계를 연주하기 위해서 피에조 버저하나를 준비하고 브레드보드에 피에조 버저를 꼽은 다음 피에조버저의 핀 하나를 GND에 연결하고 다른 하나를 디지털 핀 12에 연결합니다.





위의 그림 참조

```
int speakerPin = 12;

int numTones = 10;
int tones[] = {261, 277, 294, 311, 330, 349, 370, 392, 415, 440};
// mid C C# D D# E F F# G G# A

void setup()
{
  for (int i = 0; i < numTones; i++)
```

```
  {
    tone(speakerPin, tones[i]);
    delay(500);
  }
  noTone(speakerPin);
}

void loop()
{
}
```

특정 음을 연주하기 위해서는 주파수를 정해 주어야 합니다. 각각의 음은 서로 다른 주파수를 가지고 있고 배열에 저장이 되어 있습니다. 배열에 있는 음을 순서대로 연수하면 음계를 연주할 수 있습니다.

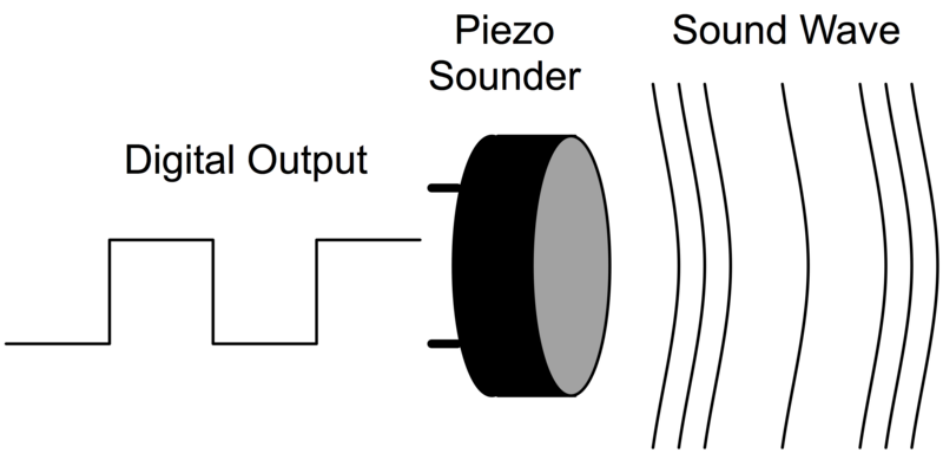
for loop문에서 각각의 음은 tone[i]를 사용하여 연주합니다. 아두이노 함수 tone()은 두 개의 파라메터를 가지는데 처음 값은 음을 플리이할 스피커핀이며, 다른 것은 플레이할 음의 주파수 입니다.

모든 음이 연주가 되면 noTone함수가 연주되고 있는 음을 멈추게 됩니다.

for loop는 메인 loop문 안에 위치하여도 되지만 setup함수안에 위치시켜 한번만 플레이 되게 만들어 놓았습니다. 또 한번 연주되는 음을 듣고 싶으면 아두이노의 리셋버튼을 누르십시오.

소리

소리파동은 공기압의 진동입니다. 진동의 속도가 소리를 만들게 되는 것이죠. 높은 주파수의 진동이 더 높은 피치를 내게 됩니다.

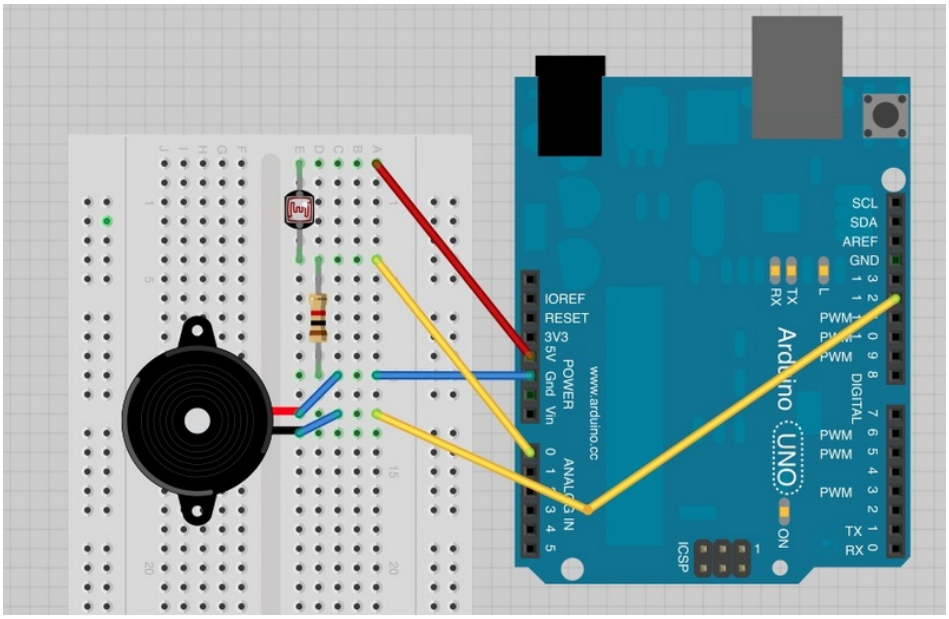


C 음은 보통 277Hz로 정의 되어 있습니다. 디지털 출력을 277번 on/off 시키면 C음을 얻을 수 있는데 이 소리를 들을려면 전기적인 신호를 파동으로 바꾸어주는 무언가를 연결해야 합니다. 이것이 스피커 혹은 피에조 버저가 됩니다.

비에조 버저는 전기적인 신호에 늘어났다 줄어드는 특별한 크리스탈을 사용하여 우리가 들을 수 있는 소리를 만들어 냅니다.

빛에 따라 연주하는 아두이노

아래와 같이 브레드 보드를 셋업합니다.



아두이노 코드

다음의 아두이노 코드를 업로드합니다.

```
int speakerPin = 12;
int photocellPin = 0;

void setup()
```

```
{
}

void loop()
{
    int reading = analogRead(photocellPin);
  int pitch = 200 + reading / 4;
  tone(speakerPin, pitch);
}
```

이 스케치 코드는 매우 간단합니다. 단순히 A0에서 조도값인 아날로그 값(0~700)을 읽어서 4로 나눈뒤 200을 더하여 제일 낮은 주파수를 200Hz로 만들고 플레이 시키는 코드입니다. 주파수대역은 200Hz~370Hz가 되게 됩니다. 포토셀앞에서 손을 흔들어 음악을 플레이하여 보십시오.

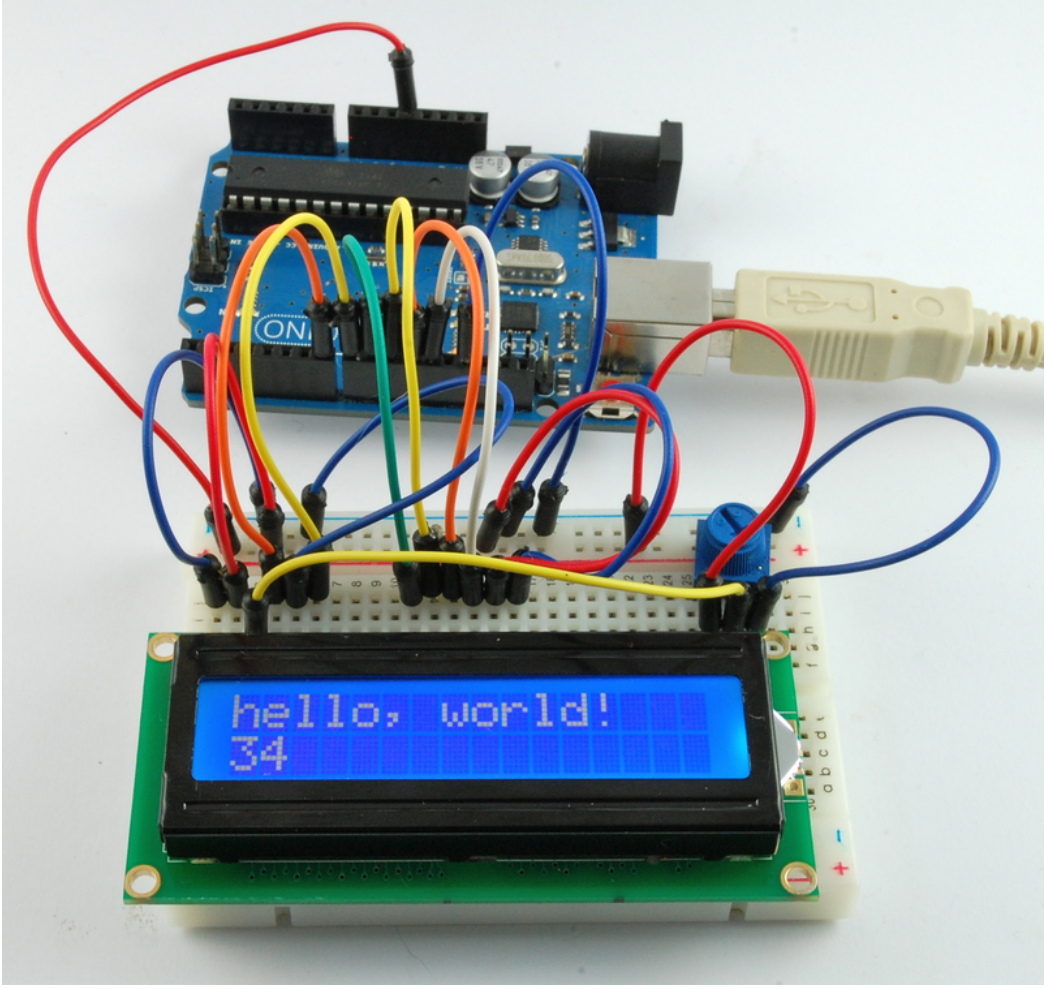
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 11. LCD 디스플레이 연결하기

아두이노 강좌

2013/06/17 20:53 http://blog.naver.com/ubicomputing/150170136407

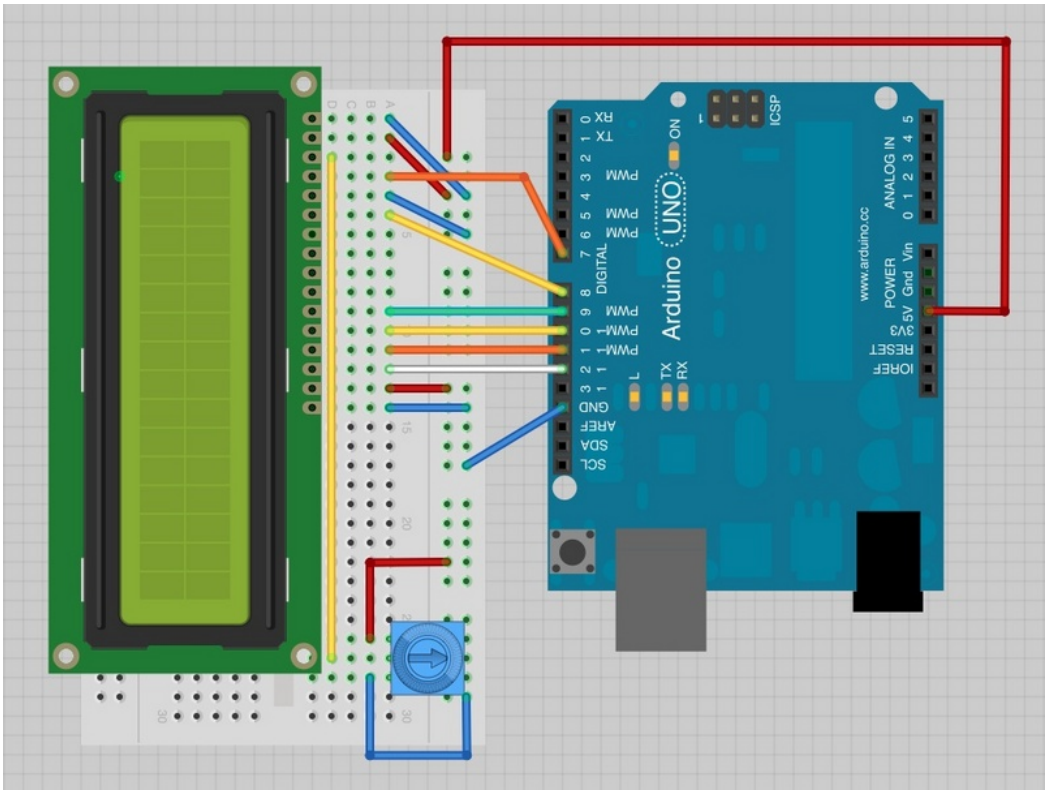
본 게시글에서는 character LCD 디스플레이를 결선하고 사용하는 방법에 대하여 설명하도록 하겠습니다.



사용되는 디스플레이는 LED 백라이트가 있으며, 두개열에 16개의 문자를 표시할 수 있습니다. 본 강좌에서는 LCD 라이브러리를 사용하는 아두이노 예제 프로그램을 실행하여 보고 다음 강좌에서는 센서를 이용하여 온도나 조도를 LCD에 표시하여 보도록 하겠습니다.

브레드보드 레이아웃

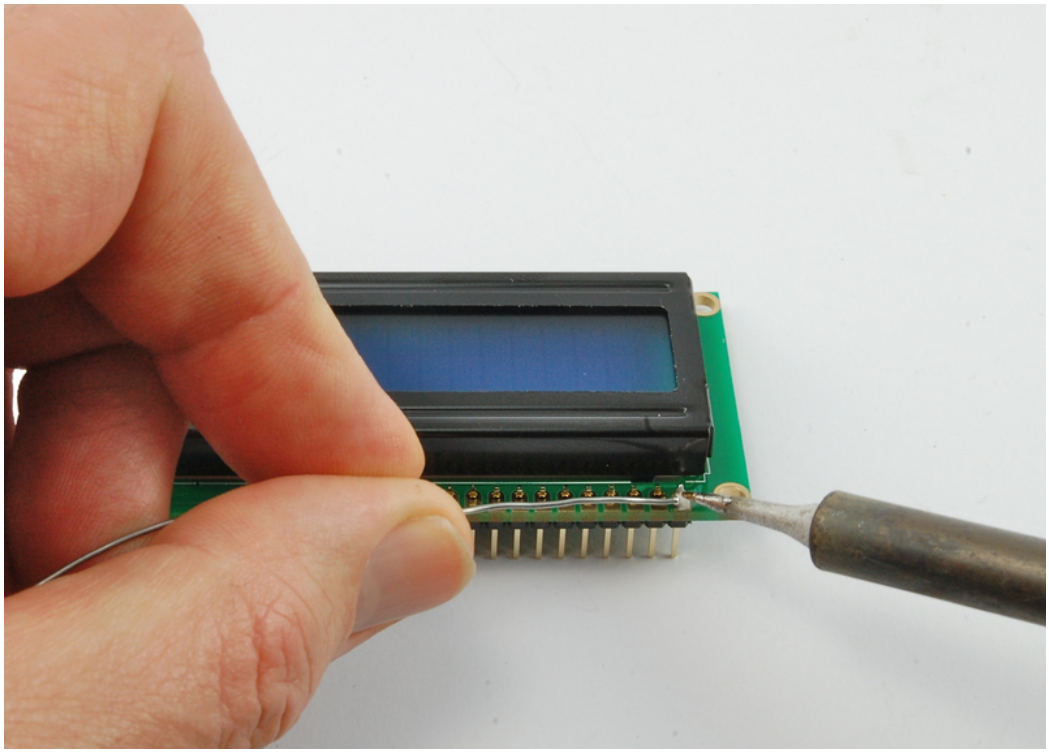
LCD 디스플레이는 동작을 위해서 6개의 아두이노 핀이 필요하며 6개의 핀은 모두 디지털 출력으로 셋팅되어야 합니다. 5v와 GND 연결은 당연히 필요하겠죠.



위의 그림과 같이 결선합니다. 포텐셔미터가 하나 연결되어 있는데, 이 포텐셔미터는 LCD의 constrast를 조절하는데 사용이 됩니다.

LCD에 핀 납땜하기

LCD가 납땜이 되어 있지 않다면 납땜을 합니다.



아두이노 코드

아두이노 IDE는 LCD 라이브러리를 사용한 예제코드를 가지고 있습니다. IDE에서 Examples --> Liquid Crystal --> HelloWorld 를 찾아 보십시오.

이 예제는 우리가 사용하는 핀과 다른 핀을 LCD용으로 사용하므로 코드를 수정하여 주어야 합니다. 이 부분은 사용하고자 하는 LCD의 종류에 따라 다를 수 있으므로 LCD 데이터 쉬트를 참고하여 고쳐주어야 합니다.

예제에서 아래의 코드를 찾아서,

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
아래와 같이 변경합니다:

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
코드를 아두이노에 업로드하면, 다음과 같이 'hello, world'가 디스플레이 되는 것을 볼 수 있습니다.

에제 코드에서 첫번째로 나오는 것은 아래의 코드입니다.

#include <LiquidCrystal.h>

이 코드는 아두이노에 LCD라이브러리를 사용할 것이라고 알려주는 코드입니다. 다음에 나오는 코드는 우리가 수정한 코드로 아두이노와 LCD가 어떤 핀으로 연결되었는지를 알려주는 역활을 합니다.

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
함수에 넘겨주는 파라메터는 다음과 같습니다.

디스플레이 핀이름	디스플레이 핀번호	아두이노 핀
RS	4	7
E	6	8
D4	11	9
D5	12	10
D6	13	11
D7	14	12

이 예제를 업로드하고 글자가 나오지 않는다면 포텐셔미터를 조절하여 글자가 표시될때까지 명암을 조절하십시오.

setup함수에서는 다음의 두개의 명령이 있습니다.

lcd.begin(16, 2);
lcd.print("hello, world!");
첫번째 줄은 디스플레이가 가진 행과 열 정보를 LCD라이브러리에 알려주는 코드이며, 두번째 줄은 "hello, world"를 LCD에 디스플레이하라는 코드입니다.

loop함수에는 두개의 명령이 있습니다.

lcd.setCursor(0, 1);
lcd.print(millis()/1000);
첫번째 명령은 다음 텍스트가 위치할 커서를 설정하는 명령입니다. 여기서는 행이 0, 열이 1이 되겠네요. 두번째 줄은 밀리세컨드의 숫자를 출력합니다.

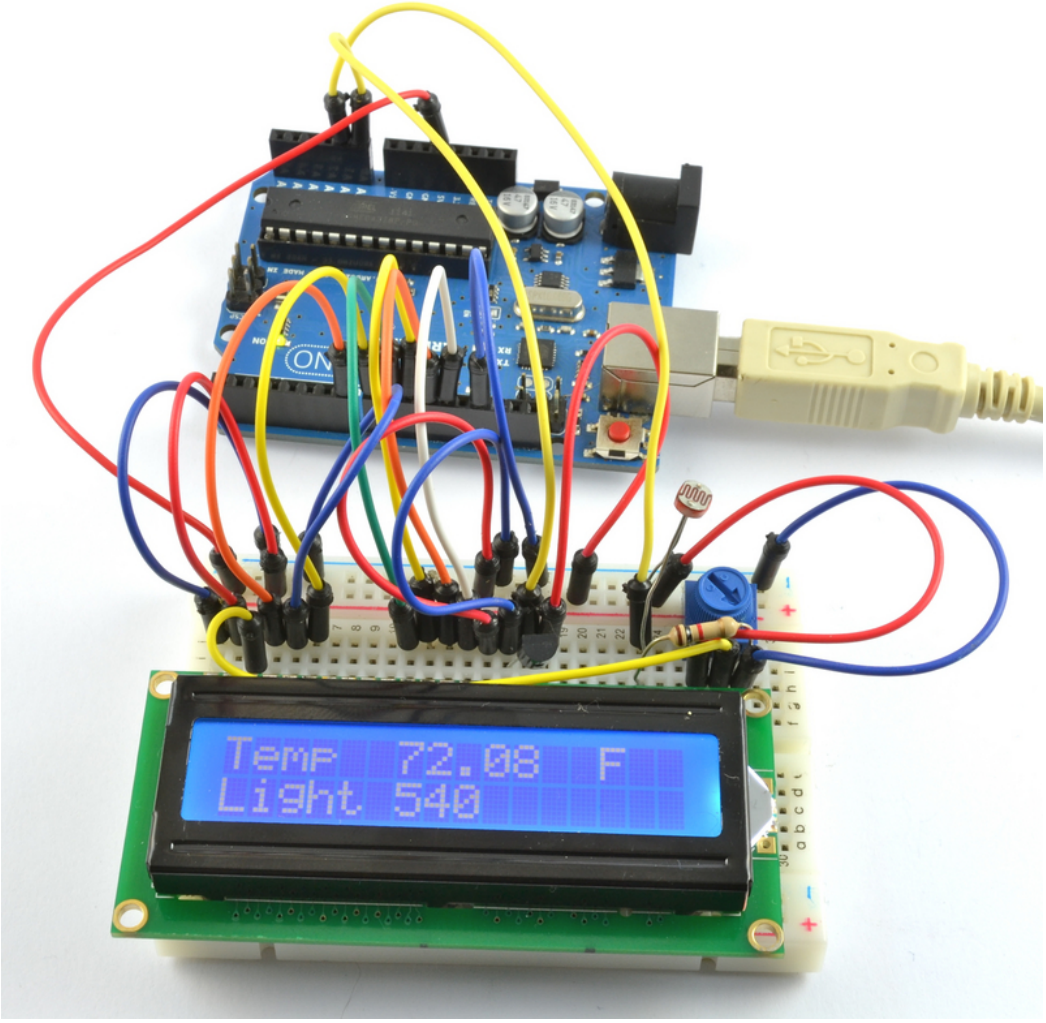
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 12. LCD에 온도 및 조도 표시하기

아두이노 강좌

2013/06/18 11:33 http://blog.naver.com/ubicomputing/150170169367

본 게시물에서는 강좌 11에서 연결한 LCD를 이용하여 온도와 조도를 LCD에 디스플레이 시켜 보도록 하겠습니다.

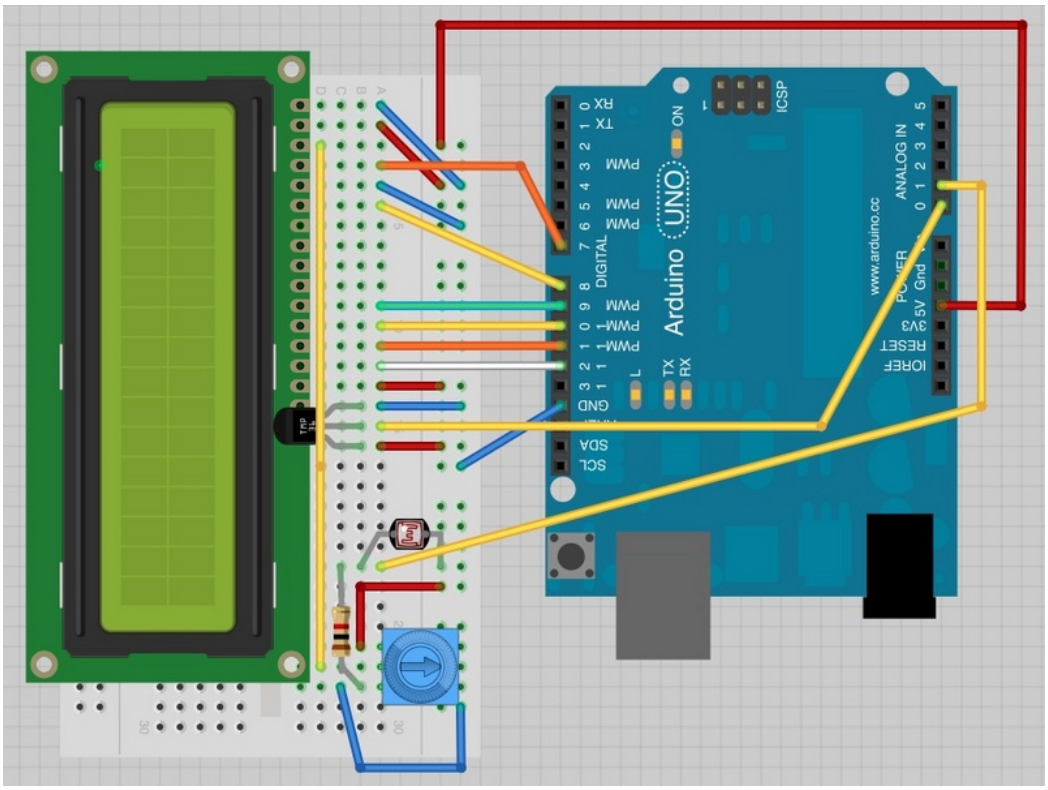


조도는 이전 강좌9에서 사용하였던 포토셀을 이용합니다. 온도를 측정하기 위해서 TMP36 온도 센서를 이용합니다. 이 온도센서는 3개의 리드선을 가지고 있으며 그중 두개는 각각 5V, GND연결에 사용되고 나머지 한개는 온도출력으로 아두이노의 아날로그 입력에 연결되게 됩니다.

〈TMP36 온도센서〉

브레드보드 레이아웃

이전 강좌 11에서 사용하였던 레이아웃입니다. 아래와 같이 브레드보드를 셋업하세요.



포토셀과 1k오옴 저항, TMP36은 새롭게 브레드보드에 추가되었습니다. TMP36은 평평한 쪽을 아두이노를 바라보게 연결해야 합니다.

아두이노 코드

아래의 스케치 코드를 아두이노에 업로드하고 손가락을 센서에 갖다 대어서 온도를 올려 보십시오. 디스플레이에 온도가 올라가는 것을 확인 할 수 있을 것입니다. 비슷하게 포토셀 앞에서 손을 흔들어 빛을 가리면 조도 역시 변경되는 것을 확인 수 있습니다.

```
#include <LiquidCrystal.h>

int tempPin = 0;
int lightPin = 1;

// BS E D4 D5 D6 D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup()
{
  lcd.begin(16, 2);
}

void loop()
{
  // Display Temperature in C
  int tempReading = analogRead(tempPin);
  float tempVolts = tempReading * 5.0 / 1024.0;
  float tempC = (tempVolts - 0.5) * 100.0;
  float tempF = tempC * 9.0 / 5.0 + 32.0;
  // ----------------
  lcd.print("Temp F ");
```

```
  lcd.setCursor(6, 0);
  lcd.print(tempF);
  // Display Light on second row
  int lightReading = analogRead(lightPin);
  lcd.setCursor(0, 1);
  // ----------------
  lcd.print("Light ");
  lcd.setCursor(6, 1);
  lcd.print(lightReading);
  delay(500);
}
```

아래와 같이 lcd()함수에 주석을 붙여 알아보기 쉽게 하였습니다.

```
// BS E D4 D5 D6 D7
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

loop함수에서는 두개의 흥미있는 부분이 있는데, 첫번째로 온도센서로부터 나오는 아날로그 값을 실제 온도로 변환하여 는 부분이며, 두번째로는 그것을 디스플레이 하는 부분입니다. 먼저 온도를 계산하는 부분을 살펴보도록 하겠습니다.

```
int tempReading = analogRead(tempPin);
float tempVolts = tempReading * 5.0 / 1024.0;
float tempC = (tempVolts - 0.5) * 100.0;
float tempF = tempC * 9.0 / 5.0 + 32.0;
```

analogRead()함수에 의해 읽혀진 0~1023의 값(tempPin 아날로그 입력)은 0~5V의 값을 구하기 위해 5를 곱한 후 1024로 나누게 됩니다.

TMP36에서 오는 전압을 온도(C)로 변환하기 위해 0.5V를 측정값에서 뺀뒤 100을 곱합니다. 참고: TMP36의 데이터 쉬트를 보면 25도에서 750mV의 전압출력은 낸다고 적혀 있으며, 전압출력은 온도에 linear(1도당 10mV)합니다.

Fahrenheit온도로 계산하기 해서는 9/5를 곱한후 32를 더합니다.

계속 변하는 숫자를 LCD에 디스플레이할 때는 이전에 쓰여졌던 숫자가 이후에도 남아있는 것을 방지하기 위해 전체 LCD화면을 다시 써주게 들어야 합니다.

```
// ----------------
```

```
lcd.print("Temp F ");
lcd.setCursor(6, 0);
lcd.print(tempF);
```

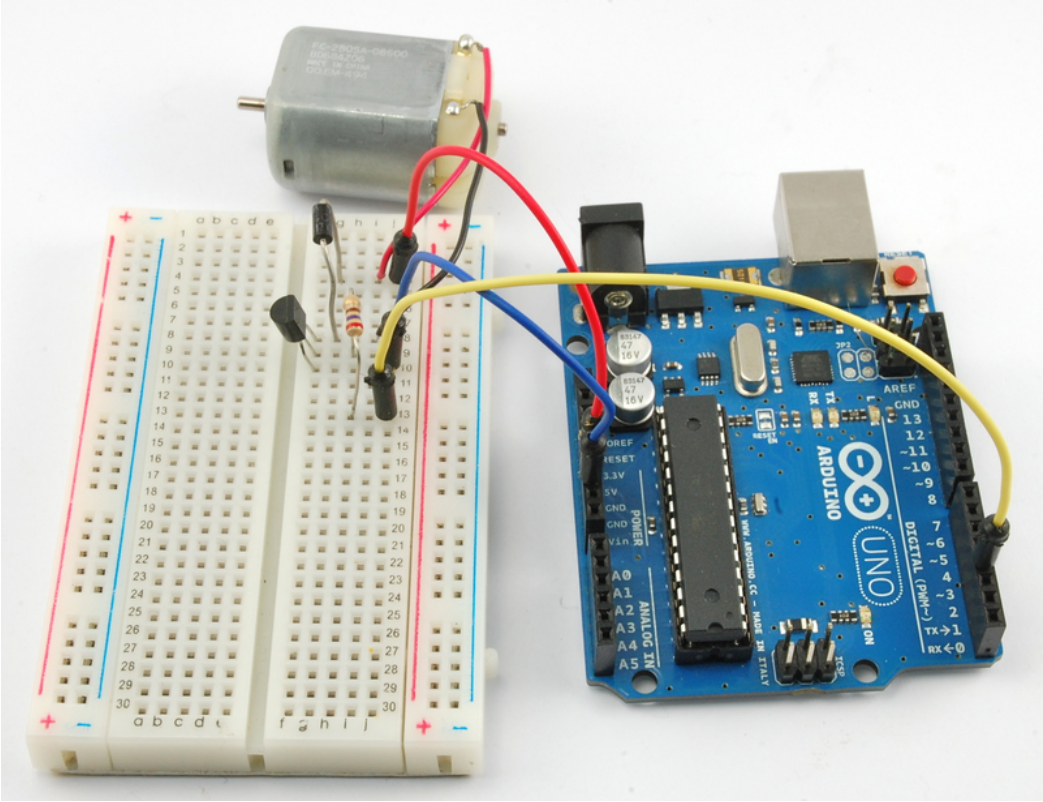
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 13. 아두이노를 사용하여 DC모터 제어하기

아두이노 강좌

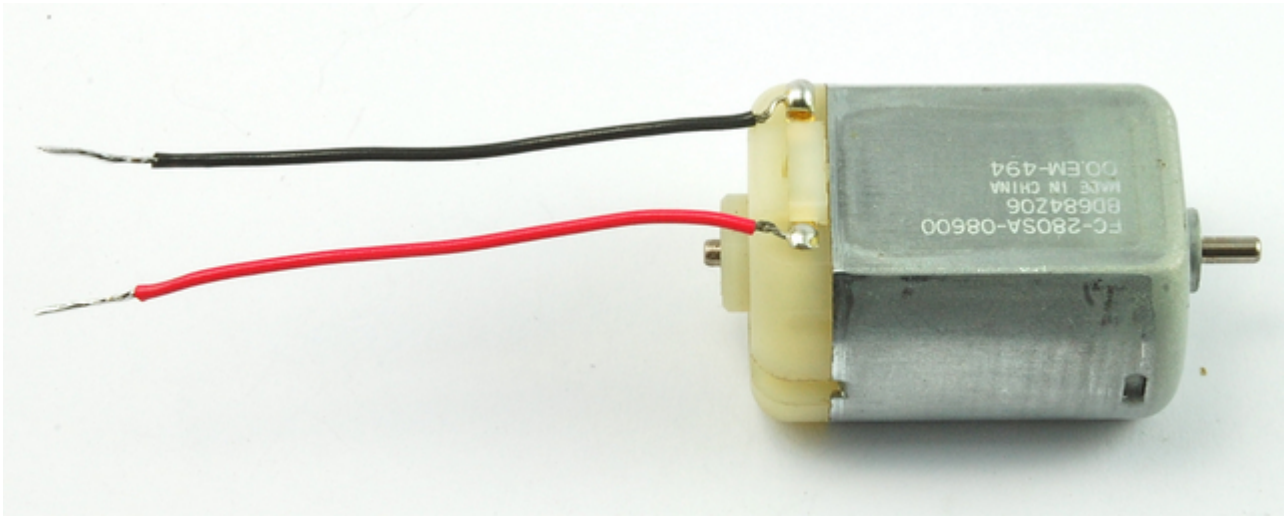
2013/06/18 14:03 http://blog.naver.com/ubicomputing/150170178581

이번 게시물에서는 아두이노와 트랜지스터를 이용하여 어떻게 작은 DC모터를 제어하는지 설명하겠습니다.

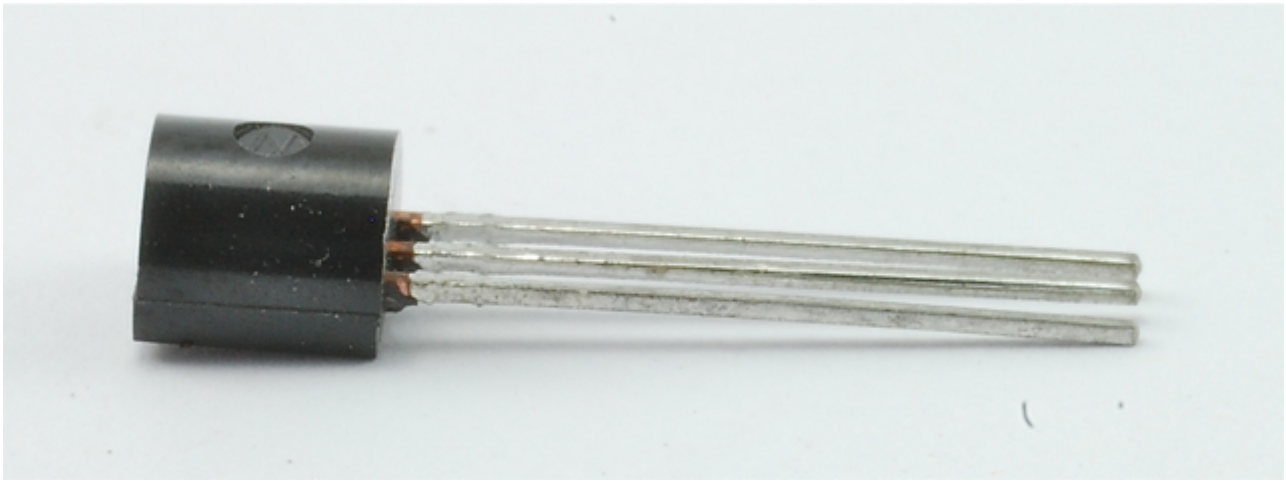


아두이노 아날로그 출력(PWM)을 모터의 속도를 조절하기 위하여 사용할 것입니다. 모터의 속도는 아두이노 IDE상의 시리얼 모티너 프로그램에서 입력되며 0~255의 값을 가집니다.

사용하려는 DC 모터입니다.



사용하려는 PN2222 트랜지스터입니다.



사용되는 다이오드 1N4001입니다.

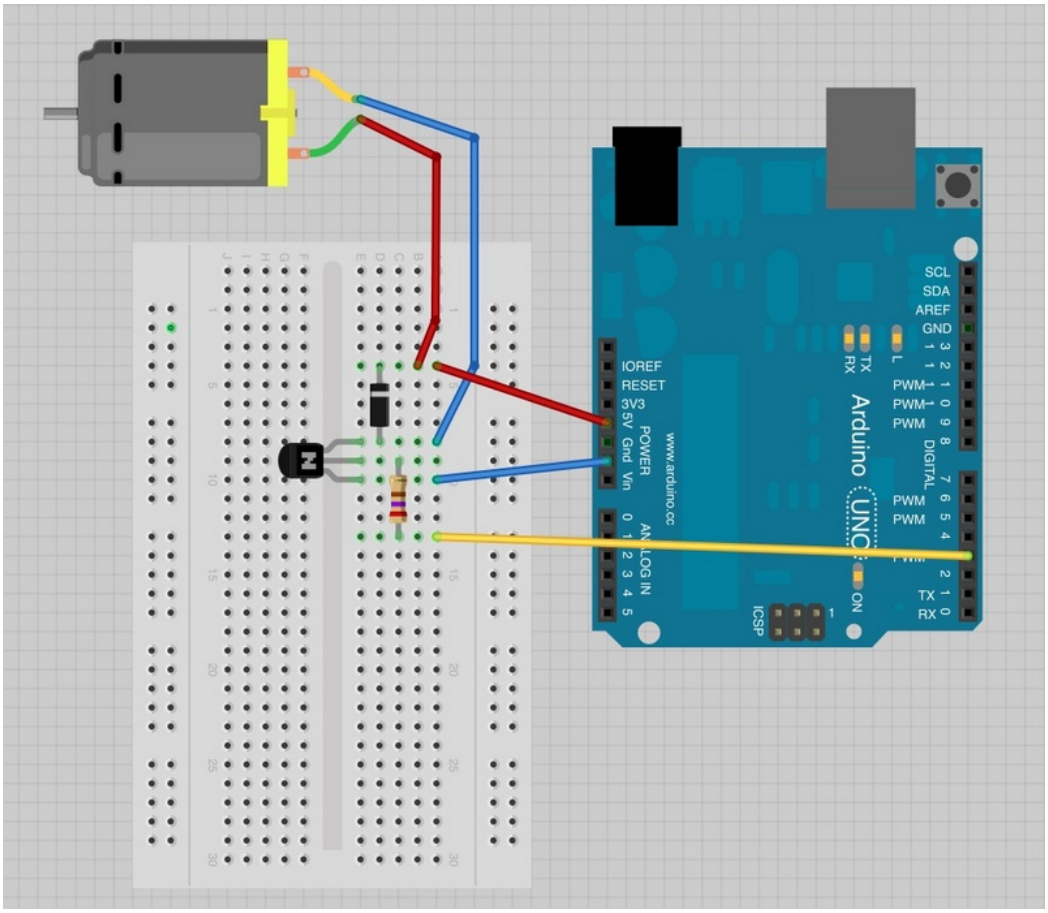
저항은 270오옴이 사용되었습니다.

브레드보드 레이아웃

브레드보드를 아래와 같이 셋업합니다. 셋업시 두가지를 살펴보아야 하는데, 첫번째로 트랜지스터가 올바르게 연결되어 있는지 확인하여야 합니다. 트랜지스터의 평평한 면이 블레드보드 오르쪽으로 오게 만들어야 합니다.

두번째로 다이오드의 줄무늬가 5V전원쪽으로 위치하여야 합니다.

> 사용하는 모터가 250mA이상의 전류를 소비한다면, 이것은 USB포트가 공급할 수 있는 범위가 넘서 설수 있기 때문에 USB포트 대신 전원어댑터를 아두이노에 연결하여야 합니다.



위의 그림과 같이 브레드보드를 셋업합니다.

아두이노 코드

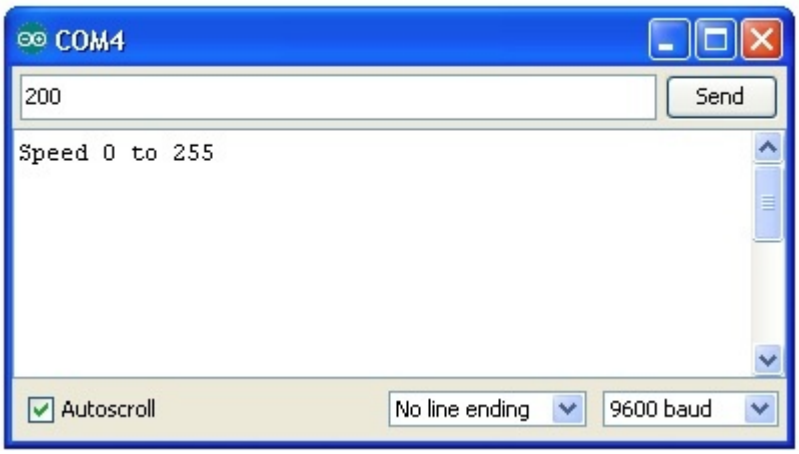
아래의 코드를 아두이노에 업로드 합니다.

```
int motorPin = 3;
```

```
void setup()
{
    pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  while (! Serial);
  Serial.println("Speed 0 to 255");
}
void loop()
{
  if (Serial.available())
  {
      int speed = Serial.parseInt();
      if (speed >= 0 && speed <= 255)
      {
          analogWrite(motorPin, speed);
      }
  }
}
```

트랜지스터는 스위치처럼 동작하여 모터로 가는 전력을 제어합니다. 아두이노 핀3번이 트랜지스터를 on/off하는데 사용되어 motorPin이라고 스케치에서 명명되었습니다.

스케치가 시작하면 시리얼 모니터에 제어하고 싶은 모터의 속도를 입력하라는 메세지가 뜨게 됩니다. 입력할 수 있는 값은 0에서 255까지 입니다.

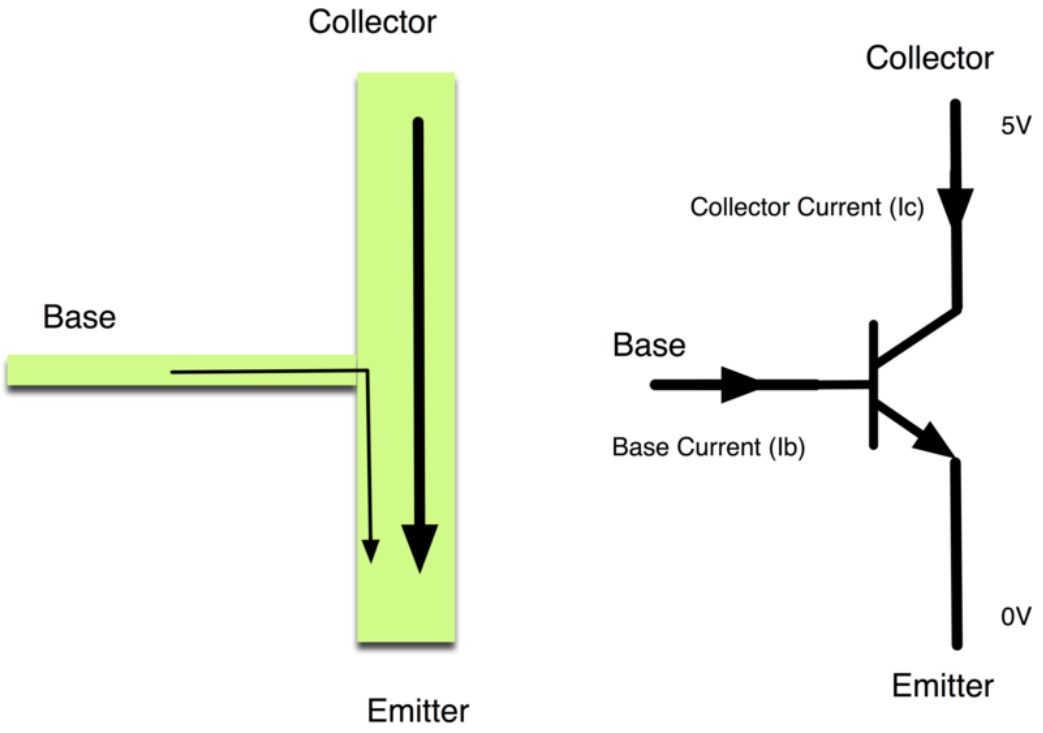


loop함수에서는 Serial.parseInt함수가 시리얼 모니터에 입력된 숫자를 스트링형태로 읽어 int 타입으로 변환합니다. 시리얼 모니터창에는 아무 숫자나 입력하여도 loop함수내의 if문에서 0~255사이의 값만 analogwrite합니다.

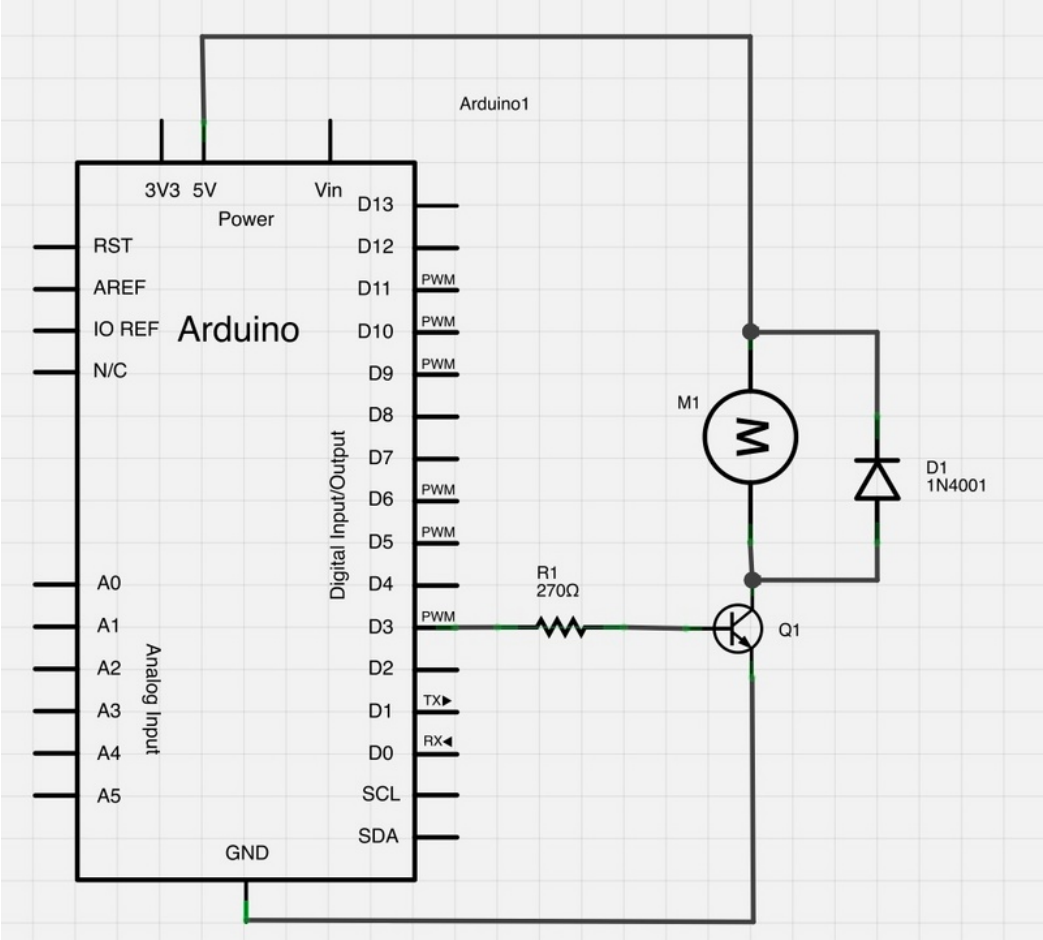
트랜지스터

DC모터는 보통 아두이노 디지털 출력 핀보다 더 많은 전력을 사용하기 때문에 직접적으로 제어를 합니다. 만약 모터를 아두이노 핀에 직접 연결한다면 아두이노 보드가 손상 받을 확률이 농후 합니다.

PN2222와 같은 트랜지스터는 아두이노의 디지털 핀에서 나오는 작은 전류를 사용하여 동작이 될 수 있기 때문에 , 스위치로 사용하여 모터와 같이 부하가 많이 걸리는 부품을 제어할 수 있습니다.



트랜지스터는 3개의 다리가 있습니다. 작은 양의 전류를 베이스로 흘려보내면 전기는 콜렉터에서 에미터로 흐르게 됩니다. 베이스에 흘려 보낼 작은 전류는 아두이노 디지털 출력으로 가능합니다.



위의 회로도를 보면 아두이노의 D3핀이 저항을 통하여 트랜지스터의 베이스로 연결되어 있습니다. 저항은 트랜지스터에 과도한 전류가 들어가 트랜지스터가 망가지는 것을 방지하여 주는 역활을 합니다.

모터쪽에는 다이오드가 연결되어 있습니다. 다이오는는 전기가 한방향으로만 흐르도록 만들어주는 부품입니다. 모터를 끌때, 음극 전압 스파이크가 생기고 이것은 아두이노나 트랜지스터를 망가트릴 수 있습니다. 다이오드는 모터로부터 꺼꾸로 흐르는 전류로부터 트랜지스터와 모터를 보호합니다.

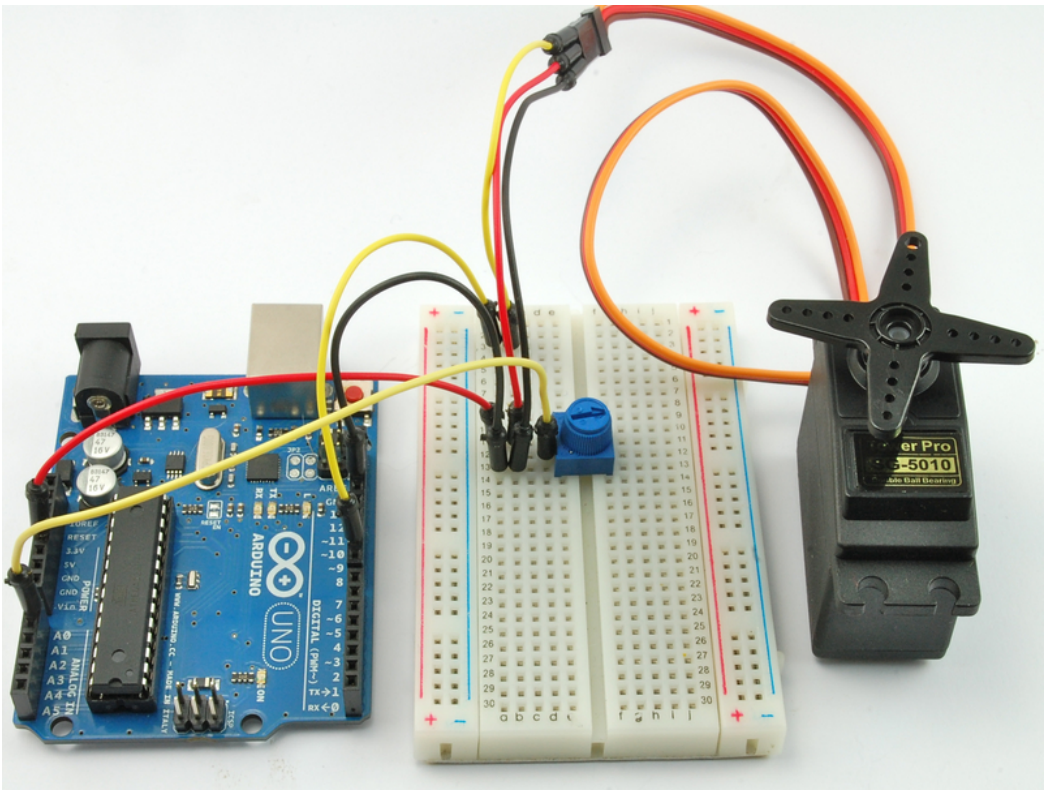
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 14. 아두이노로 서보모터 제어하기

아두이노 강좌

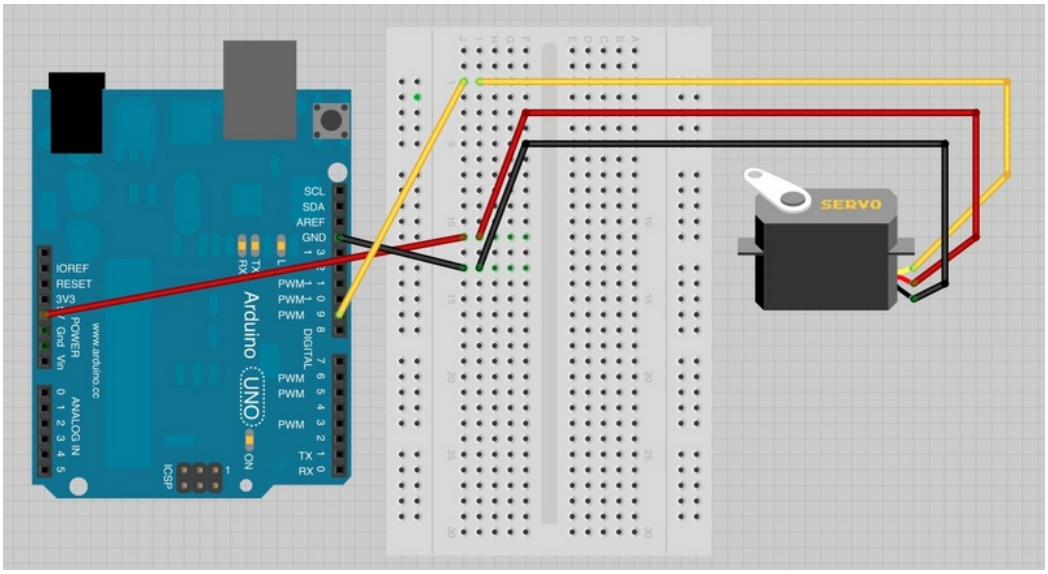
2013/06/18 18:24 http://blog.naver.com/ubicomputing/150170197643

본 게시글에서는 아두이노를 이용하여 서보모터를 어떻게 제어하는지 살펴보도록 하겠습니다. 서보를 자동으로 앞뒤로 움직이게 만들어보고, 포텐셔미터를 추가하여 서보의 포지션을 제어하도록 하겠습니다.



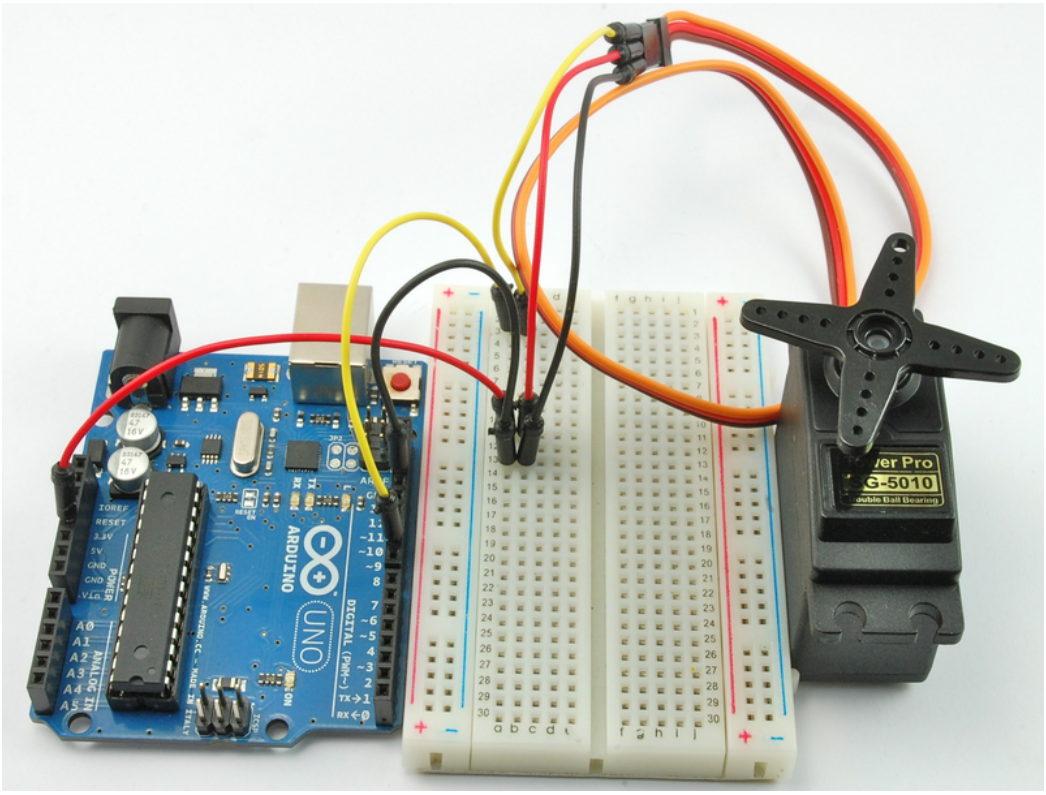
브레드보드 레이아웃1

아래의 브레드보드에서는 서보 모터만이 아두이노에 연결되어 있습니다. 아래와 같이 브레드보드를 셋업합니다.



서보모터는 3개의 리드선을 가지고 있는데 보통 빨강이 5V, GND는 검정이나 갈색입니다. 나머지 리드선은 제어 리드선으로 보통 오렌지색이나 노랑색을 띄고 있습니다. 제어 리드선을 디지털 핀 9번에 연결합니다.

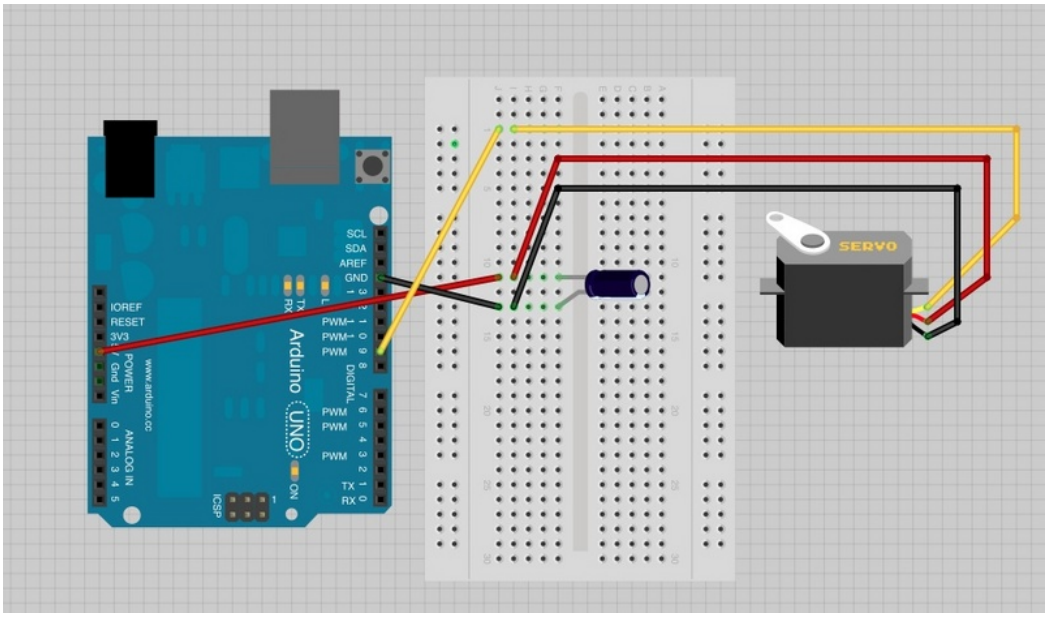
서보 리드선 끝의 소켓에 점퍼를 연결하여 아래와 같이 브레드보드에 셋업합니다.



만약 서보가 오동작을 한다면,

위와 같이 셋업을 하였는데 만약 서보가 이상하게 동작을 한다면, 그것은 서보가 너무 많은 전력을 끌어다 쓰기 때문일 수 있습니다. USB로 아두이노에 전원을 공급하기 때에만 이러한 현상이 발생할 수 있습니다. 보통 모터가 움직이기 시작할때 전력을 많이 쓰기 때문에 이는 아두이노 보드의 전압을 떨어 뜨릴 수 있으며, 이는 다시 아두이노 보드를 리셋하게 만듭니다.

만약 이러한 현상이 발생한다면, 캐페이시터(470uF이상)를 GND와 5V 사이에 추가함으로써 해결할 수 있습니다.



캐패에시터는 모터가 사용하는 전력 저장소와 같은 역활을 하여 모터가 시작할때 아두이노 전원뿐아니라 캐패이시터에서 전기를 끌어다 쓸수 있습니다.

캐패이시터의 긴 다리가 양극이며 이 리드선이 5V에 연결되어야 합니다. 음극 리드선은 보통 '-' 심볼로 표시됩니다.

아두이노 코드 1

아래의 스케치 코드를 아두이노에 업로드하면 서보가 한방향으로 돌다가 다른 방향으로 다시 도는 것을 확인 할 수 있습니다.

본 스케치 코드는 아두이노 servo 예제 폴더에 있는 sweep 코드를 기반으로 작성된 코드입니다.

#include <Servo.h>

```
int servoPin = 9;
Servo servo;
int angle = 0; // servo position in degrees
void setup()
{
   servo.attach(servoPin);
}
void loop()
{
  // scan from 0 to 180 degrees
  for(angle = 0; angle < 180; angle++)
  {
      servo.write(angle);
      delay(15);
  }
  // now scan back from 180 to 0 degrees
  for(angle = 180; angle > 0; angle--)
  {
      servo.write(angle);
      delay(15);
  }
}
```

서보 모터는 펄스에 의해 제어되어 사용하기 쉽습니다. 서보용 아두이노 라이브러리가 있어 서보에게 단지 동작할 각도를 알려주기만 하면 됩니다.

이러한 서보 라이브러리를 사용하기 위해서는 아래와 같은 코드로 아두이노 IDE에게 서보 라이블러리를 사용할 것임을 알려주어야 합니다.

- #include <Servo.h>

그리고, 서보를 제어하기 위해 servoPin이라는 이름의 변수를 정의하였습니다.

```
Servo servo;
```

위의 코드에서 보면 Servo 타입 servo변수를 정의한 것을 볼 수 있습니다. Servo는 라이브러리에서 제공하는 변수 타입으로 서보사용시 사용합니다. 8개까지의 서보를 정의할 수 있습니다. 만약 두개의 서보를 가지고 있다면, 아래와 같이 정의 할 수 있을 것입니다.

```
Servo servo1;
Servo servo2;
```

servo변수에게는 실제로 제어할 서보의 제어핀이 어떤 핀인지를 알려주어야 니다. 아래의 코드로 servo변수에게 제어핀을 알려줍니다.

```
servo.attach(servoPin);
```

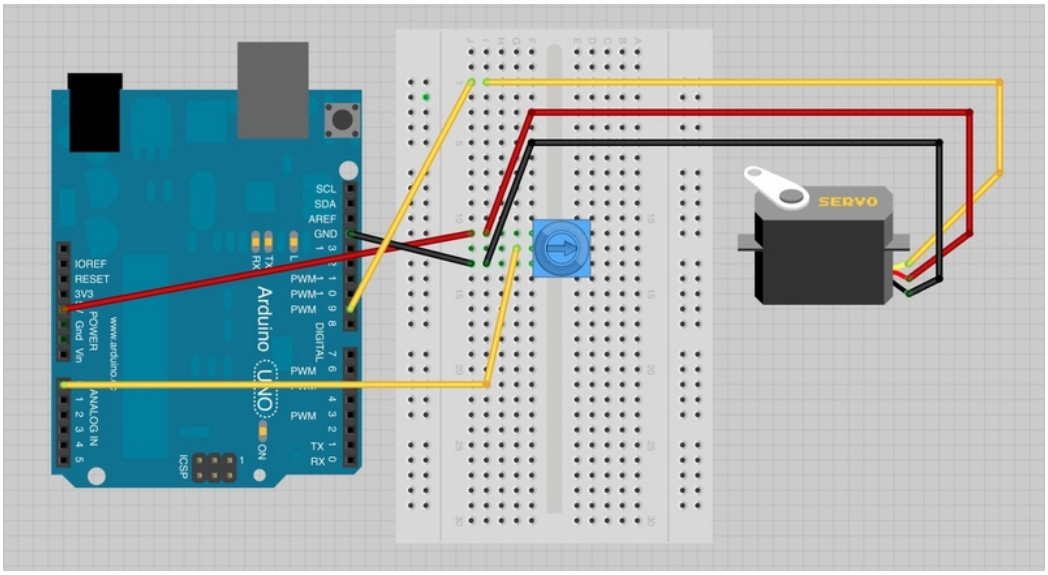
변수 angle은 서보의 현재각도를 저장하기 위해 사용됩니다. loop함수에서 우리는 두개의 for loop를 사용하는 것을 볼 수 있는데, 첫번째로 각도를 한방향으로 180도까지 증가 시키고, 다음 for 루프에서는 반대방향으로 움직이게 합니다.

```
servo.write(angle);
```

위의 코드는 서보에게 파라메터로 들어온 각도로 위치를 업데이트 하는 명령입니다.

브레드보드 레이아웃2

그럼 두번째 브레드보드 레이아웃을 살펴보겠습니다. 이 레이아웃에는 포텐셔미터가 포함되어 포텐셔미터를 돌려 서보의 위치를 조절할 수 있습니다. 포텐셔미터의 슬라이더 리드를 아두이노의 A0에 연결합니다.



아두이노 코드

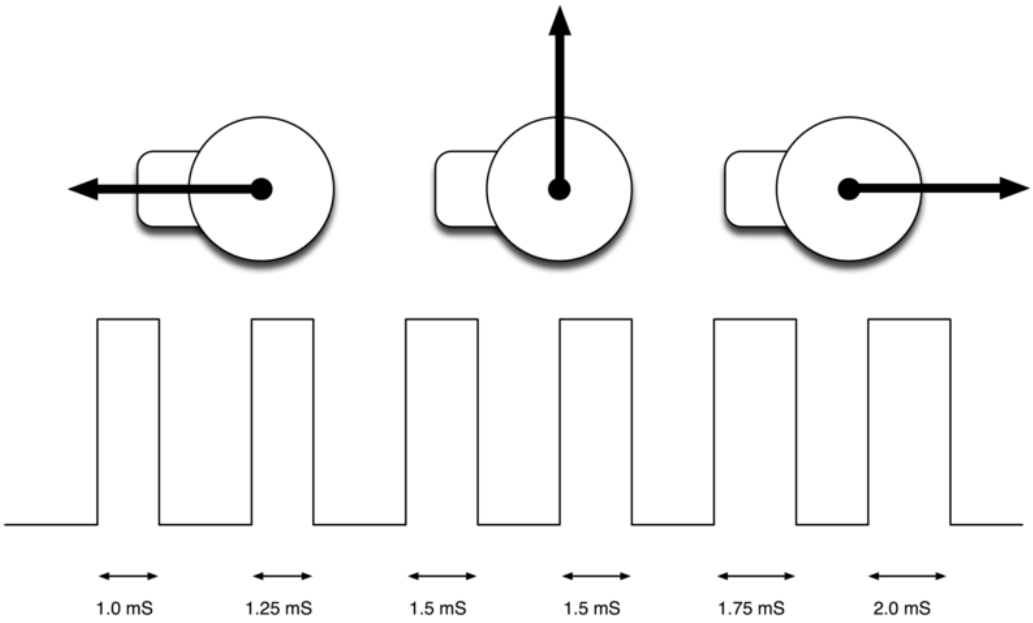
```
#include <Servo.h>

int potPin = 0;
int servoPin = 9;
Servo servo;
void setup()
{
  servo.attach(servoPin);
}
void loop()
{
  int reading = analogRead(potPin); // 0 to 1023
  int angle = reading / 6; // 0 to 180-ish
  servo.write(angle);
}
```

두번째 코드도 첫번째 코드와 비슷한데 potPin이라는 변수가 추가되었습니다. 서보의 위치를 셋팅하기 위해 포텐셔미터로부터 나오는 아날로그 값을 A0핀으로부터 analogRead함수를 통하여 읽어 드리면, 0에서 1023의 값이 읽히게 됩니다.

서보는 오직 180도만 회전할 수 있기 때문에 180도에 맞게 0~1023까지의 값을 스케일 하여야 합니다. 0-1023을 6으로 나누면 0~170이 되어 알맞게 스케일 되었습니다

서보 모터

서보 모터의 위치는 펄스의 길이에 따라서 설정됩니다. 서보는 대략 매 20ms마다 펄스를 받게되는데, 만약 이 펄스가 1ms동안 high이면 각은 0 이며, 1.5ms동안 high이면 중간위치에 위치하게 되고 2ms인 경우는 180도가 되게 됩니다..



서버가 움직일 수 있는 끝부분은 제품에 따라 차이가 있는데 많은 서보가 170도까지만 회전을 합니다. 360도까지 회전하는 서보도 있으니 필요하다면 제품을 찾아보는 것도 좋습니다.

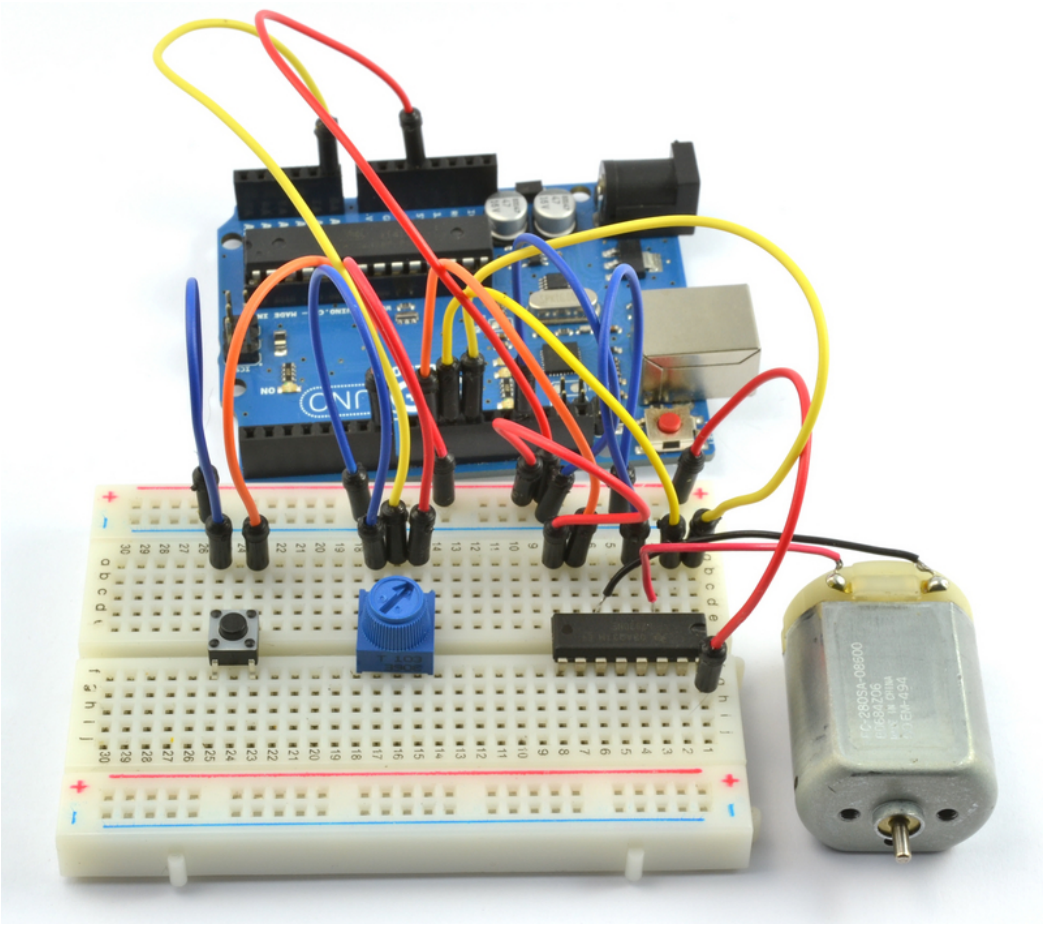
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 15. DC 모터 양방향으로 제어하기

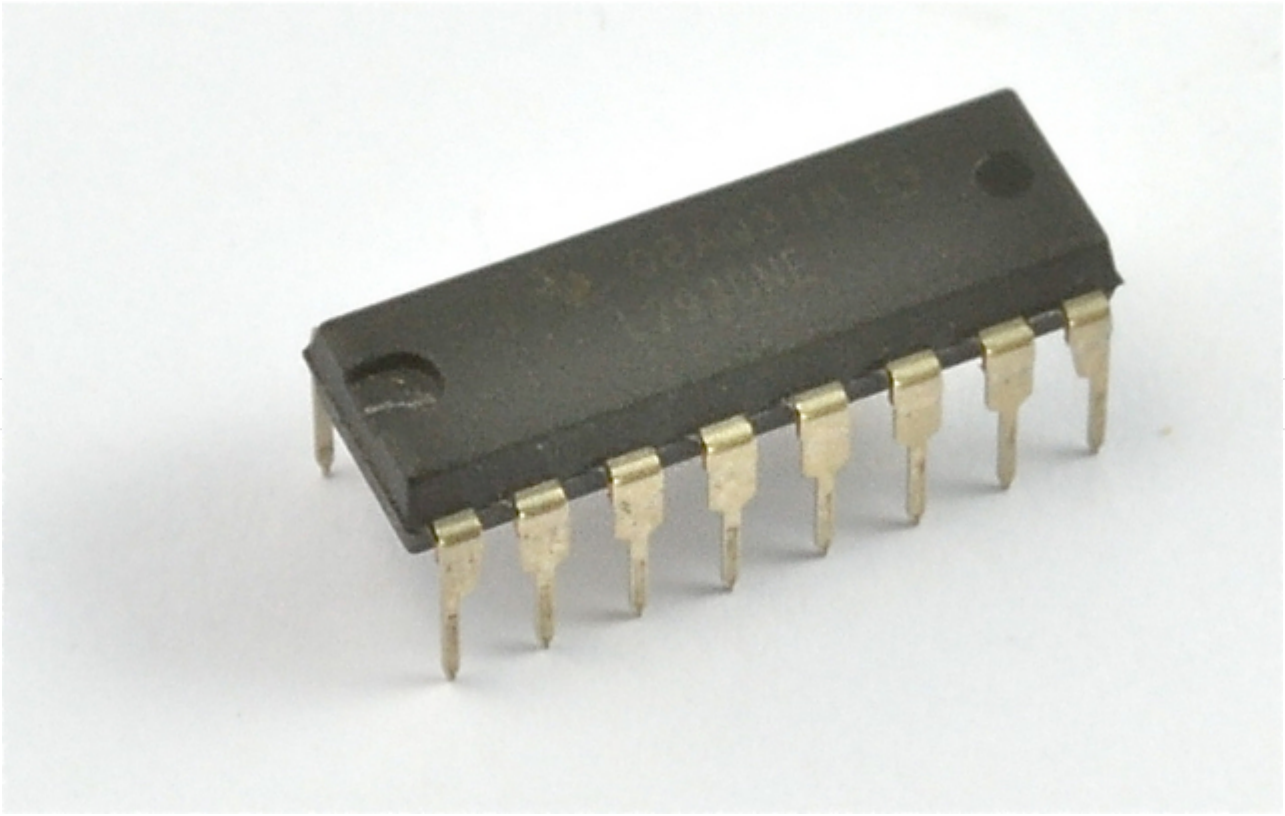
아두이노 강좌

2013/06/19 14:55 http://blog.naver.com/ubicomputing/150170252237

본 게시물에서는 아두이노와 L293D 모터 드라이버 칩을 이용하여 DC모터의 속도와 동작방향을 제어하는지에 대해 설명하겠습니다.



본 프로젝트에서는 포텐셔미터로 모터의 속도를 조절하고 푸쉬버튼으로 모터의 방향을 조절합니다.

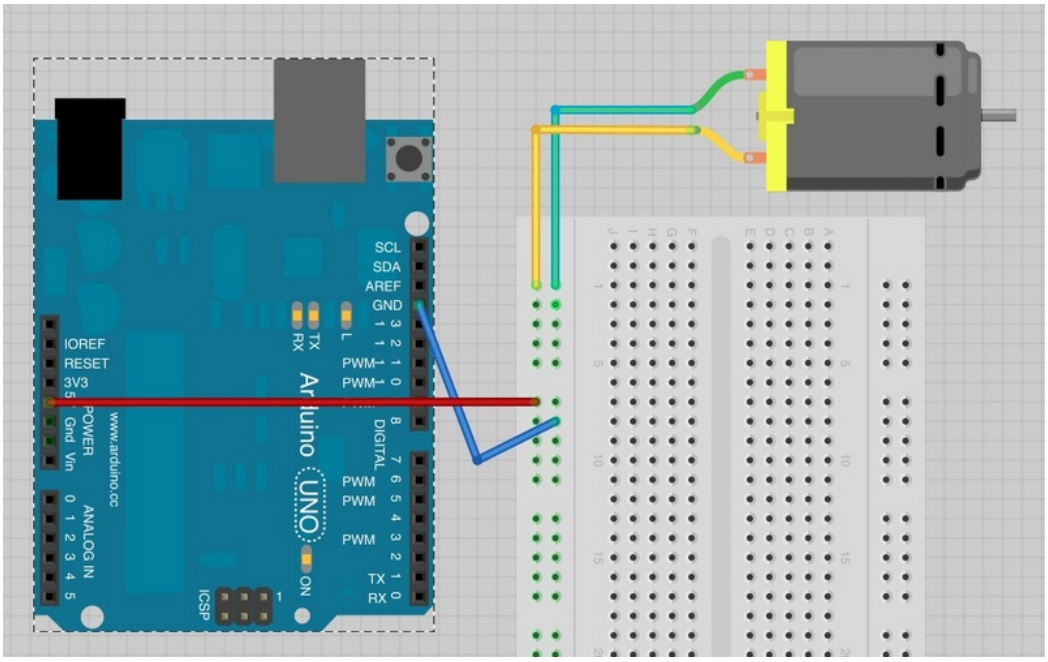


사용되는 L293D IC 입니다.

실험

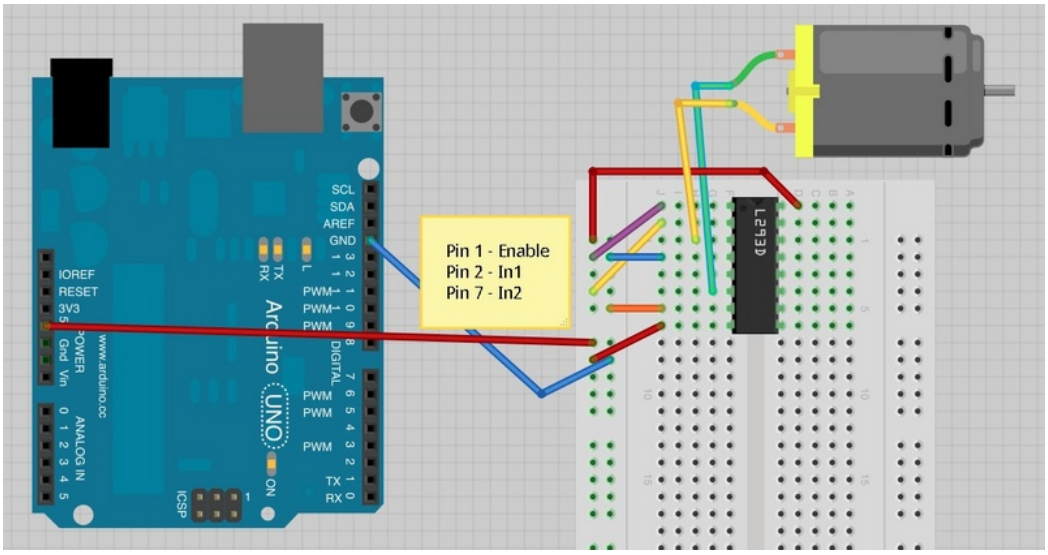
아두이노로 모터를 제어하기에 앞서, 먼저 L293D 모터 제어 칩이 어떻게 동작하는 지 알아 보기 위한 간단한 실험을 하겠습니다.

먼저 모터에 5V를 공급하는 것부터 시작합니다..



손가락을 모터 샤프트에 살짝 갖다대면 어느 방향으로 모터가 회전하는지 알 수 있는데, 모터가 어느방향으로 회전하는지 확인하여 보십시오. 모터에 연결된 선을 서로 바꾸면 모터는 아까와는 반대방향으로 회전합니다.

L293D 칩은 위와 같이 극성에 따라 동작방향이 변하는 점을 이용한 칩입니다. 아래와 같이 브레드보드를 셋업하십시오. 먼저 아두이노가 아닌 수동으로 실험을 진행하여 보겠습니다.



우리가 관심있는 것은 L293D의 세개의 핀인데, 그 핀들은 Pin1(Enable), Pin 2 (In1), Pin 7 (In2)입니다. 이 핀들은 5V 혹은 GND에 보라, 노랑, 오렌지 색 점퍼로 연결되어 있습니다.

위의 그림에서 보듯, 모터는 한방향으로 움직입니다. 이 방향을 편의상 A라고 부릅시다.

Pin1(enable)을 GND로 옮기면 모터는 멈추어 서게 되고, pin In1과 In2에 무엇을 하던 모터는 동작하지 않습니다. Enable핀은 On/Off 스위치와 같으며, PWM출력을 이용하여 모터의 속도를 조절하는데 유용하게 사용이 됩니다. enable핀을 다시 5V로 연결합니다.

이제 In1(pin2, 노랑)을 5V에서 GND로 옮깁니다. 그러면 In1과 In2는 지금 모두 GND에 연결된 상태로 모터는 다시 멈추게 됩니다.

In2핀을 GND에서 5V로 옮깁니다. 그러면 모터는 반대방향(방향B)로 움직이게 됩니다.

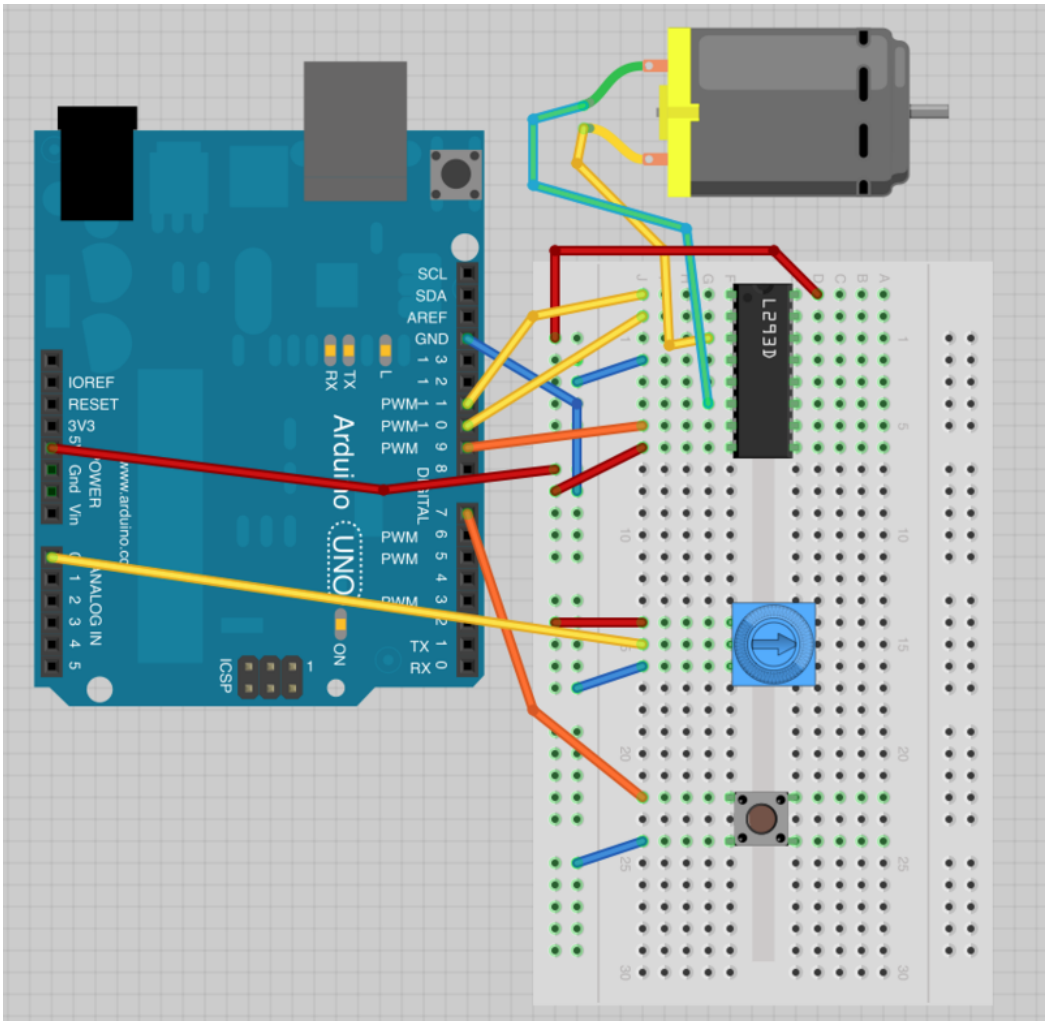
마지막으로 In1을 5V로 다시 옮깁니다. 그러면 In1, In2가 모두 5V가 되는데, 이 경우 모터가 멈추게 됩니다.

In1과 In2 핀의 조합에 따른 모터의 동작을 아래의 표에 정리하였습니다.

In1	In2	Motor
GND	GND	멈춤
5V	GND	방향A로 회전
GND	5V	방향B로 회전
5V	5V	멈춤

브레드보드 레이아웃

자, 이제는 모터의 enable, In1, In2핀을 아두이노에 연결하여 제어를 하여 보도록 하겠습니다. IC를 브레드보드에 연결할 때 IC의 U형 노치가 브레드보드 상단을 향하도록 설치가 되어야 합니다.



아두이노 코드

아래의 코드를 아두이노에 업로드 하십시오.

```
int enablePin = 11;
int in1Pin = 10;
int in2Pin = 9;
int switchPin = 7;
int potPin = 0;

void setup()
{
```

```
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  pinMode(switchPin, INPUT_PULLUP);
}

void loop()
{
  int speed = analogRead(potPin) / 4;
  boolean reverse = digitalRead(switchPin);
  setMotor(speed, reverse);
}

void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}
```

setup함수안에 핀과 핀의 모드가 정의 되어 있습니다. loop함수내에서는 모터속도 값은 포텐셔미터로부터 얻은 아날로그 값을 4로 나눈 값으로 할당되어 있습니다.

4로 나눈 이유는 아날로그 입력값은 0에서 1023이며 실제로 우리가 필요한 값은 0에서 255사이의 값이기 때문입니다.

만약 버튼이 눌리게 되면 모터는 전진하고, 그렇지 않으면 반대로 돌게 됩니다. reverse 변수값은 스위치 핀으로부터 읽어들인 값으로 설정되기 때문에, 버튼이 눌리면 False, 아니면 True입니다.

모터의 속도와 방향의 값은 setMotor()함수의 파라메터로 전달되어 모터 드라이버 칩의 해당 핀을 셋팅함으로써 모터를 제어하게 됩니다.

```
void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, ! reverse);
```

```
  digitalWrite(in2Pin, reverse);
}
```

먼저, enable 핀에 analogWrite함수를 이용하여 속도를 설정합니다.L293의 enable핀은 In1, In2가 어떻게 셋팅되어 있던지 상관없이 모터를 on/off합니다. 뱡향을 제어하기 위해서는 In1과 In2핀은 반드시 반대되는 값으로 설정되어야 합니다.

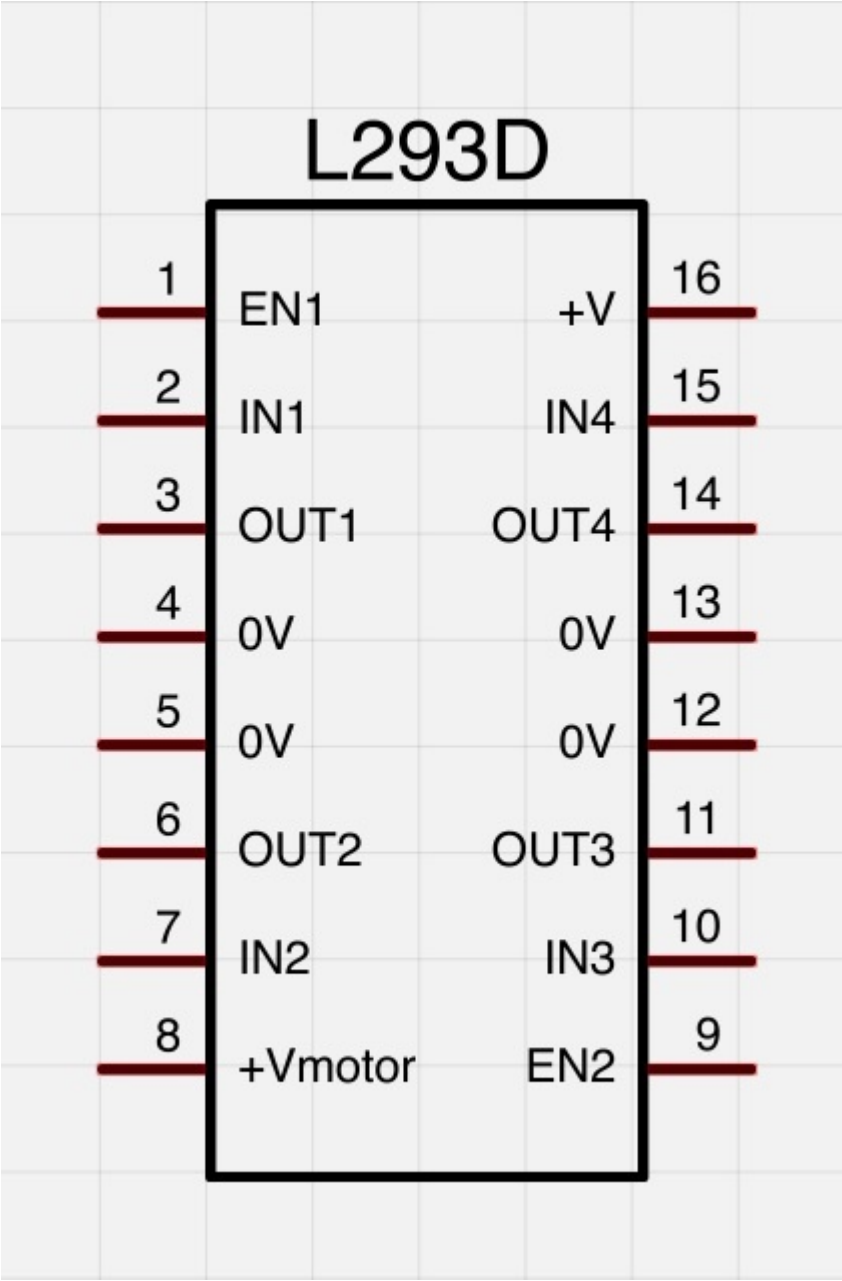
만약 In1이 high이고 In2가 low이면, 모터는 한방향으로 돌게 됩니다. 반대로 In1이 low이고 In2가 high라면 모터는 그 반대방향으로 돌게 됩니다.

! 명령은 not을 의미합니다. 그래서 reverse의 값이 무엇이던지 In1의 값은 reverse의 값과 반대되는 값으로 설정이 됩니다.

두번째 digitalWrite는 In2에 reverse의 값을 설정합니다. In1값은 reverse값의 반대값이 설정되기 때문에 In2의 값은 In1의 값과 항상 반대값이 됩니다.

L293D

L293D는 매우 유용한 칩으로 실제적으로 두개의 모터를 독립적으로 제어할 수 있습니다. 본 강좌에서는 칩의 반만을 사용한 것이죠. 칩의 왼쪽, 오른쪽으로 모터를 하나씩 연결할 수 있습니다.



만약 두번째 모터를 연결한다면 OUT3, OUT4에 연결되어야 하며 추가적인 세개의 제어 핀이 필요합니다..

- EN2는 아두이노의 PWM핀에 연결되어야 합니다.
- IN3과 IN4은 아두이노의 디지털 출력 핀에 연결되어야 합니다.

L293D는 두개의 +V핀(8번, 16번)을 가지고 있습니다. +Vmotor(8)은 모터에 전원을 공급하고 +V(16)은 칩 로직에 전원을 공급합니다. 여기서는 아두이노의 5V핀을 가지고 두개를 다 연결하였습니다만, 만약 높은 출력의 모터를 사용하거나 좀더 높은 전압의 모터를 사용한다면, 모터에게 별도의 전원을 연결하여 주는 것이 좋습니다.

가치창조기술 | www.vctec.co.kr

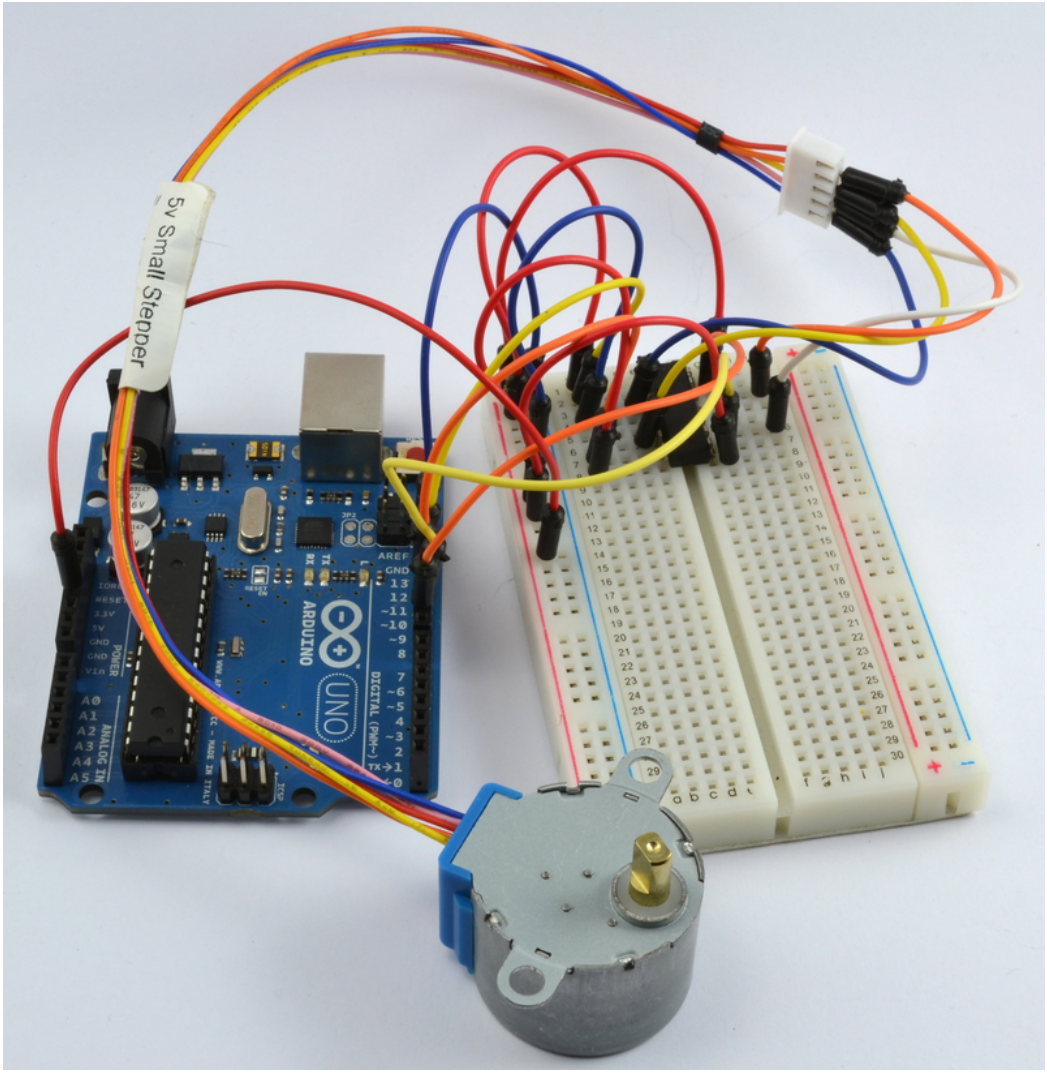
[아두이노 강좌] 16. 스텝퍼 모터 제어하기

아두이노 강좌

2013/06/20 17:09 http://blog.naver.com/ubicomputing/150170329589

스텝퍼 모터는 DC모터와 서보 모터의 중간쯤 되는 성질을 가진 모터입니다. 정교하게 위치를 조절할 수 있고, 앞뒤방향으로 한번에 한 스텝단위로 움직일수 있습니다. 그리고 일반적인 서보가 180도만 회전하는 것과는 달리 끊임없이 회전할 수 있습니다.

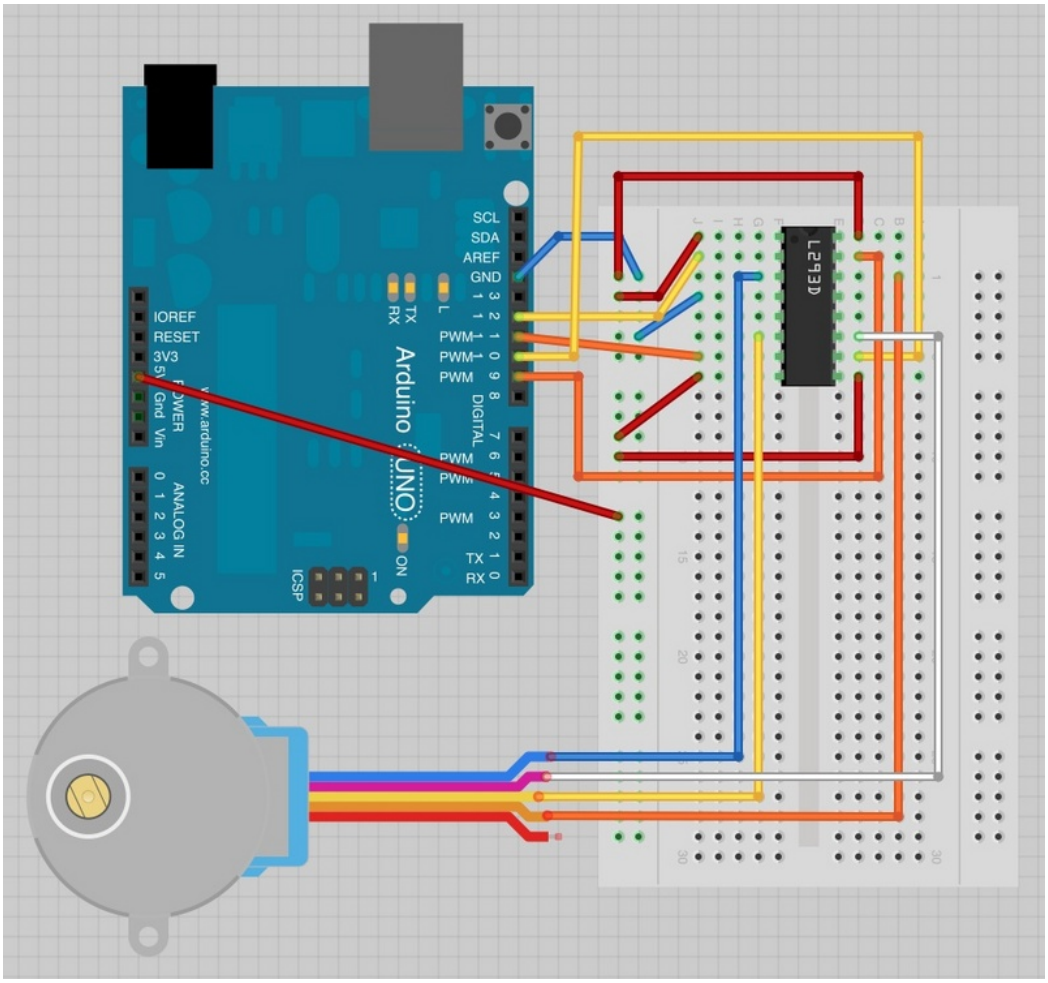
본 강좌에서는 아두이노와 L293D 모터 컨트롤 칩을 이용하여 5V 스텝퍼모터를 어떻게 제어하는지 알아 보겠습니다



브레드보드 레이아웃

스텝퍼 모터는 5개의 리드선을 가지고 있습니다. 이전 강좌에서 DC모터를 제어하기 위해서 L293D칩의 절반만 사용을 하였는데, 이번에는 양쪽 사이드를 다 사용하겠습니다.

스텝퍼 모터의 리드선 끝부분에는 소켓이 하나 있습니다. 소켓에 점퍼를 연결하여 모터를 브레드보드에 연결하기 쉽게 만들어 주십시오.



스텝퍼모터의 빨강색 리드선은 아무것과도 연결하지 않습니다.

아두이노 코드

아래의 스케치코드는 시리얼 모니터를 사용합니다. 그렇기때문에 스케치를 실행시키면 시리얼 모니터를 열고, 스텝퍼 모터를 동작시킬 step 수를 입력하여 주어야 합니다. 500정도를 입력하면 모터가 대략 360도정도 회전합니다. -500을 입력하여 보십시오. 모터가 역방향으로 회전하여 원래 자리로 되돌아 옵니다.

stepper library는 최신버전의 아두이노 IDE에 포함되어 있기때문에 최신버전이 아니라면 IDE를 업그레이드 하십시오.

```
#include <Stepper.h>

int in1Pin = 12;
int in2Pin = 11;
int in3Pin = 10;
int in4Pin = 9;

Stepper motor(768, in1Pin, in2Pin, in3Pin, in4Pin);

void setup()
{
   pinMode(in1Pin, OUTPUT);
   pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);

  // this line is for Leonardo's, it delays the serial interface
  // until the terminal window is opened
  while (!Serial);
  Serial.begin(9600);
  motor.setSpeed(20);
}

void loop()
{
  if (Serial.available())
  {
```

```
    int steps = Serial.parseInt();
    motor.step(steps);
  }
}
```

코드를 보면 스텝퍼모터를 지원하는 아두이노 라이브러리가 있습니다. 이 라이브러리 덕분에 모터를 쉽게 사용할 수 있습니다. stepper.h를 include한뒤에 in1부터 in4까지 정의되었습니다.

아두이노 라이브러리에게 어떤 핀이 모터 컨트롤러에 연결되어 있는지 알려주기 위해서 아래의 명령이 사용됩니다.

Stepper motor(768, in1Pin, in2Pin, in3Pin, in4Pin);
첫번째 파라메터는 모터가 움직일 스텝수입니다. 스텝퍼모터는 한번에 한 스텝만 움직여서 정교한 위치제어가 가능합니다.

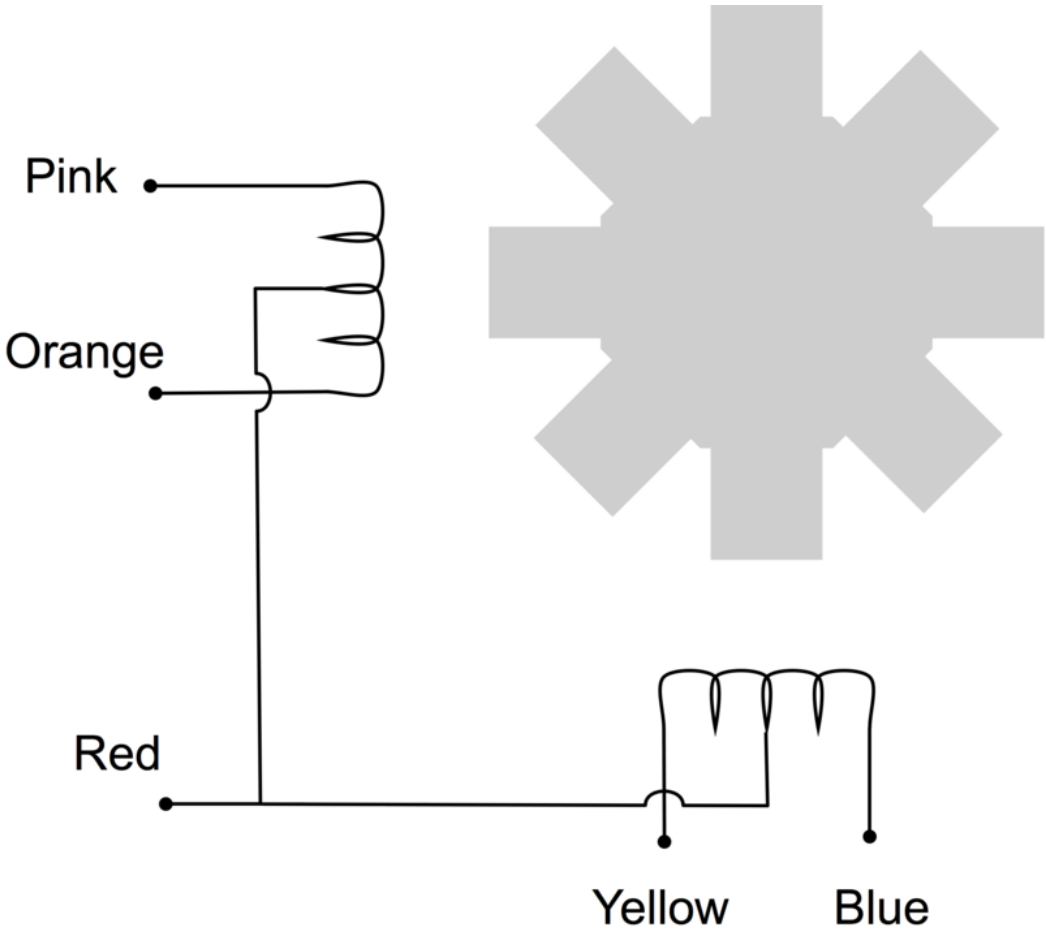
그 다음에 시리얼 통신이 시작되고, 아두이노는 시리얼 모니터를 통하여 명령을 받을 수 있는 상태가 됩니다.

마지막으로 아래의 명령을 통해 모터가 돌아가는 속도를 정해줍니다.

motor.setSpeed(10);
loop 함수는 매우 간단한데, 시리얼 모니터로부터 들어오는 명령을 기다리다가 들어오는 텍스트를 숫자로 변환하여 모터가 얼마나 돌지 명령하는 내용으로 끝이 납니다.

스텝퍼 모터

스텝퍼 모터는 톱니가 달리 휠과 전자석을 사용하여 한번에 한 스텝씩 움직이게 만듭니다.



코일을 순서에 맞게 전원을 공급하면 모터가 회전하게 됩니다. 모터가 360도를 돌기 위한 스텝은 실제로 톱니의 이빨수와 같습니다.

여기서 사용하는 모터는 48스텝을 가지고 있지만 추가적인 1:16 기어를 가지고 있어서 16 X 48 = 768 스텝이 필요합니다.

여기서는 빨강 리드선을 사용하지 않았습니다. 이 선은 다른 종류의 드라이브 회로를 사용할 때 사용되는데, 역방향으로 돌기 위한 전류를 각 코일에 허용하지 않는 드라이버 회로를 사용할 경우만 사용이 됩니다.

우리가 사용하는 L293D는 전류를 역으로 돌리는데 매우 훌륭하므로, 이 빨강 리드를 연결하지 않습니다. L293D는 각각의 코일에 어떤 방향으로든 전류의 공급이 가능합니다.

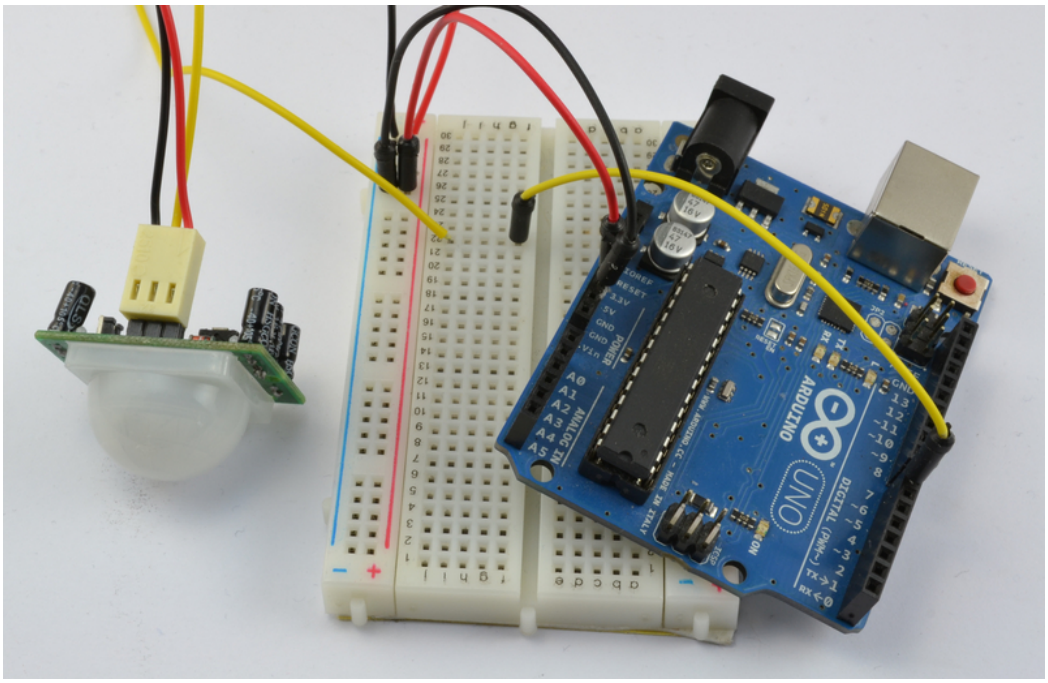
가치창조기술 | www.vctec.co.kr

[아두이노 강좌] 17. 침입자 움직임 탐지시 이메일 경고 보내기

아두이노 강좌

2013/06/21 12:58 http://blog.naver.com/ubicomputing/150170380969

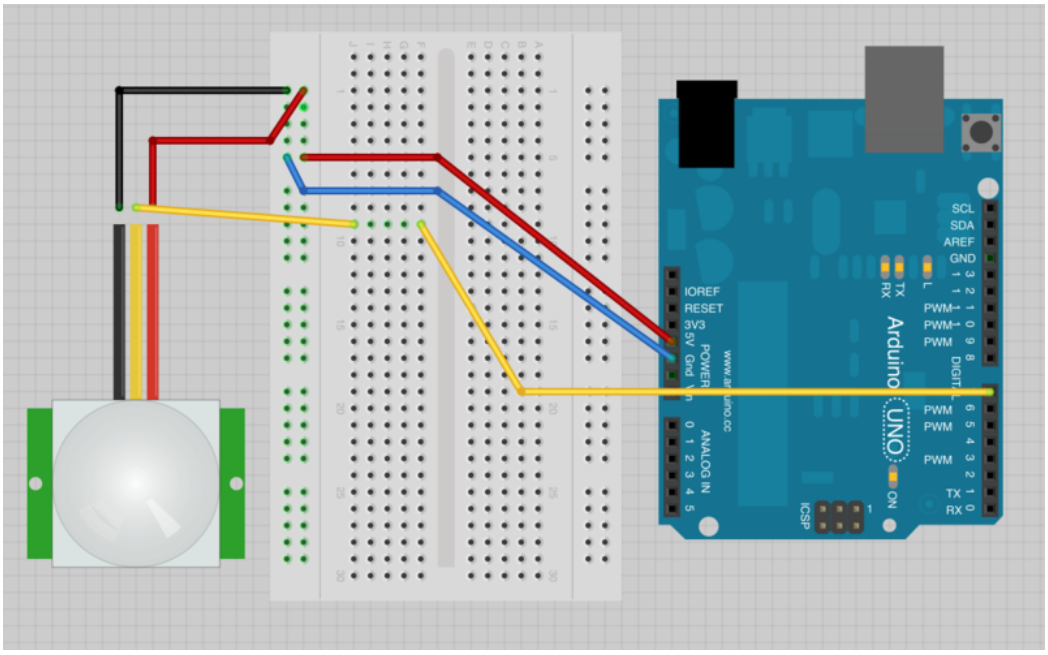
본 게시글에서는 PIR 움직임 탐지 센서를 이용하여 아두이노가 PC상의 파이선 프로그램과 통신하여 움직임이 검출되면 파이선프로그램이 이메일을 보내는 프로젝트를 진행하여 보겠습니다.



사진에 보이는 반투명한 반구 모양의 모듈이 PIR센서입니다.

브레드보드 레이아웃

브레드보드를 아래와 같이 셋팅합니다. 아두이노에 연결할 것은 PIR센서 하나입니다.



아두이노 코드

아두이노는 움직임이 감지 될때 마다 USB Serial 연결을 통하여 메일 보낼 것입니다. 하지만 이러한 방식은 매우 많은 이메일을 보내게 될 가능성이 있기 때문에 일정 시간이내에서는 메일을 또 보내지 않도록 작성되었습니다.

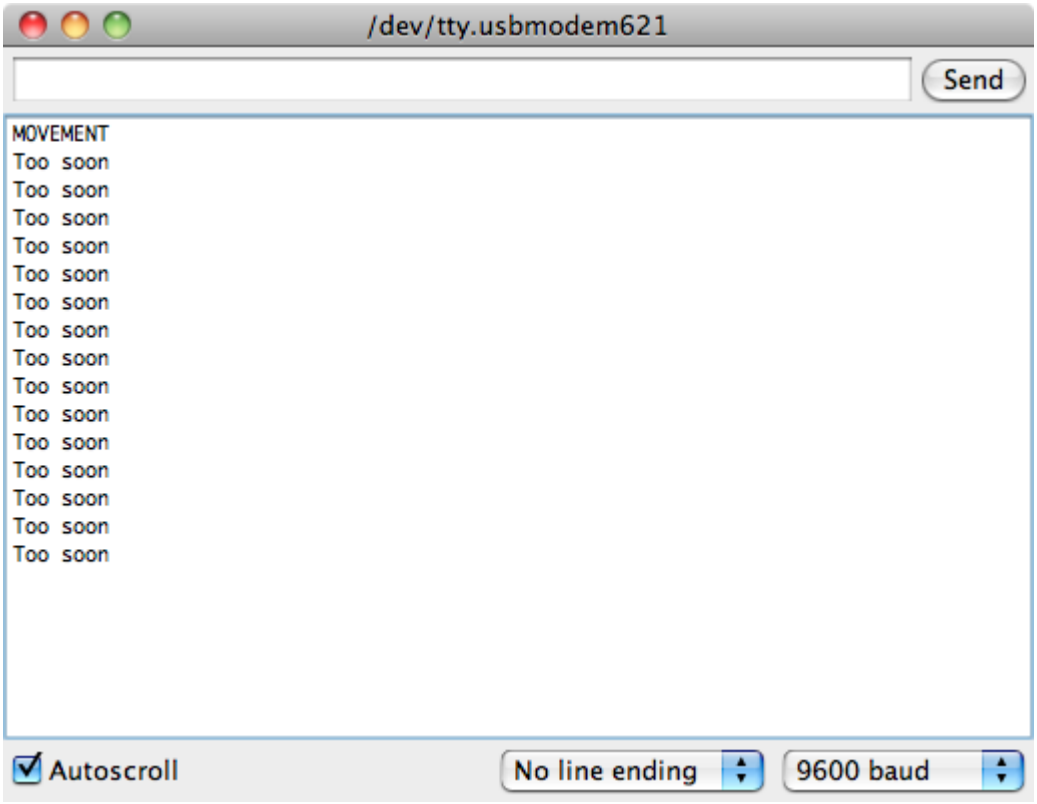
```
int pirPin = 7;

int minSecsBetweenEmails = 60; // 1 min

long lastSend = -minSecsBetweenEmails * 1000l;

void setup()
{
   pinMode(pirPin, INPUT);
   Serial.begin(9600);
}

void loop()
{
```

```
  long now = millis();
  if (digitalRead(pirPin) == HIGH)
  {
     if (now > (lastSend + minSecsBetweenEmails * 1000l))
     {
        Serial.println("MOVEMENT");
        lastSend = now;
     }
     else
     {
        Serial.println("Too soon");
     }
  }
  delay(500);
}
```

변수 minSecBetweenEmails는 60초로 설정되었지만 원하는 숫자로 변경이 가능합니다. 여기서는 60초이므로 1분이내에는 이메일이 다시 발송되지 않습니다. 언제 마지막 이메일이 발송되었는지를 확인하기 위해 lastSend변수가 사용되었습니다. 이 변수는 음수로 초기화되고 minSecsBetweenEmails의 음의 값을 가지게 됩니다. 이렇게 하면 스케치가 시작하자마자 PIR센서가 즉시 트리거 될수 있게 됩니다.

loop안에서는 함수 millis()가 사용되어 밀리세컨드를 얻을 수 있습니다. 아두이노가 시작되면 밀리세컨드를 마지막 알람이 트리거 되었을 때와 비교하고, 정해진 시간보다 넘었을 경우에만 MOVEMENT 메세지를 보내게 됩니다. 그 시간이 넘지 않았다면 Too soon메세지를 보내게 됩니다.

파이썬 프로그램을 실행하기 전에 아두이노 IDE의 시리얼 모니터를 열어서 아두이노가 올바르게 동작하는 지 테스트 할 수 있습니다.



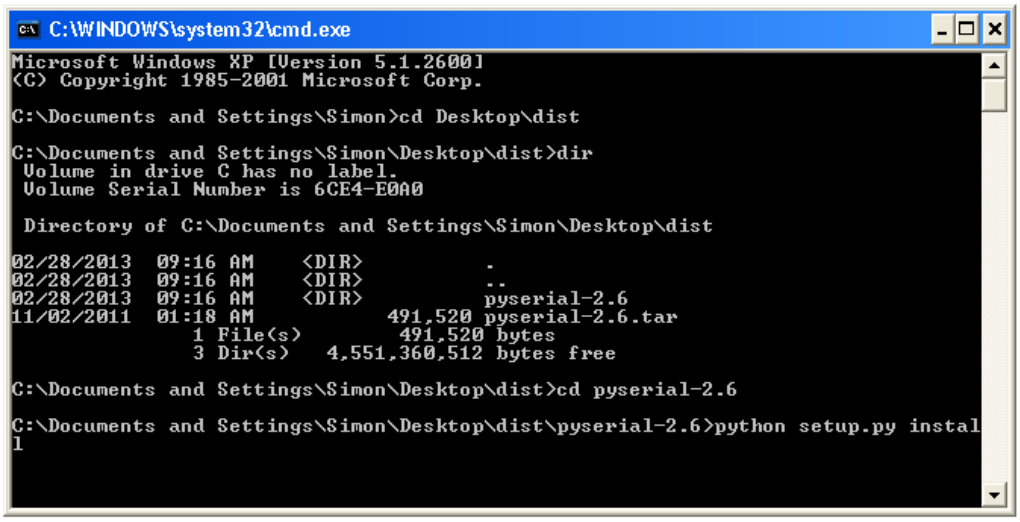
Python과 PySerial 인스톨하기

먼저 PC에 파이썬을 설치합니다.

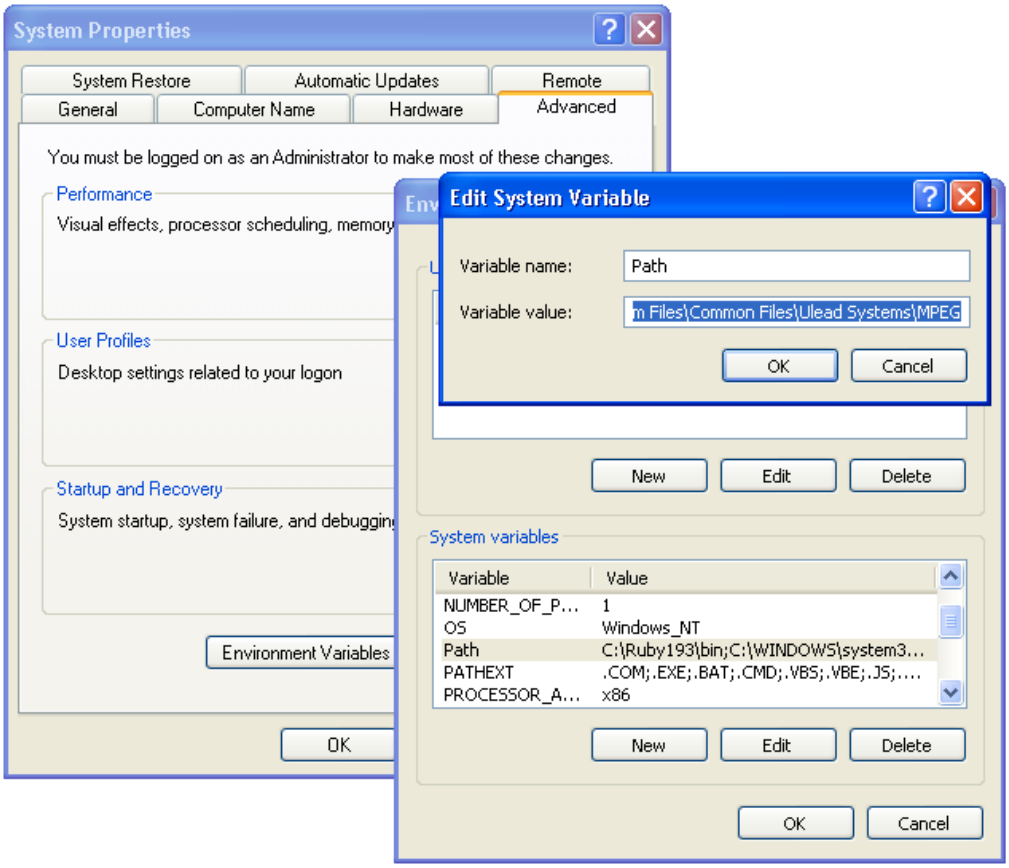
윈도우에 파이썬 설치하기

http://www.python.org/getit/ 에서 파이썬 인스톨 파일을 다운로드 받습니다.

본 프로젝트에서는 Python 2.7.3을 사용합니다. Python3에서는 PySerial을 사용시 알려진 버그가 있으므로 Python 2.7.3을 사용합니다.

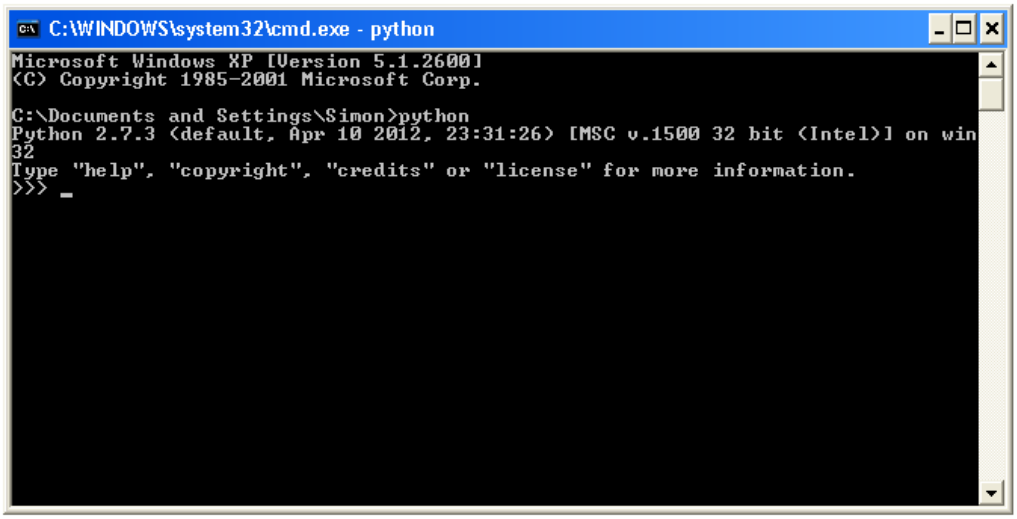


파이썬은 설치한뒤 명령창에서 사용하기 위해 아래의 그림과 같이 PATH 환경 변수를 수정합니다.



윈도우의 제어판-->시스템속성-->환경변수를 선택하여 위의 창이 나오면 아래의 Path를 선택하고 Edit버튼을 눌러서 끝부분에 *;C:₩Python27* 을 추가합니다.

만약 제대로 추가가 잘 되었다면 cmd 창에서 python 을 타이핑 할 경우 아래와 같은 화면이 나올 것입니다. 만약 안나온다면 파이선이 설치된 PC의 폴더위치를 다시 한번 확인하여 오타나 ; 가 빠지지 않았는지 확인합니다.



PySerial 설치하기

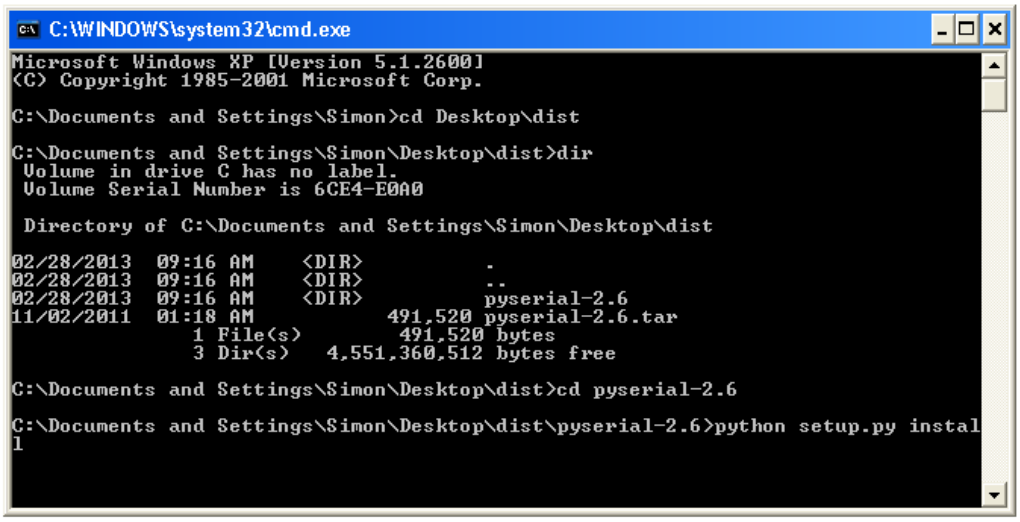
다음의 링크에서 PySerial 2.6 설치 패키지를 다운로드 받습니다. https://pypi.python.org/pypi/pyserial

파일명: pyserial-2.6.tar.gz

다운로드한 파일의 압축을 풉니다. 윈도우 사용자는 7-zip(www.7-zip.com)과 같은 툴을 다운 받아 압축을 풀수 있습니다. 리눅스나 맥사용자는 아래의 명령으로 압축을 풀수 있습니다.

$ tar -xzf pyserial-2.6.tar.gz
압축이 풀린 폴더로 들어가 설치 파일을 실행합니다. 리눅스에서는 아래와 같이 실행합니다.

sudo python setup.py install



파이썬 코드

자 이제 파이썬의 선치가 끝났으므로, 아두이노와 통신할 파이썬 코드를 작성하여야 합니다. 아래의 코드를 카피하여 "movement.py"라는 이름의 파이선코드를 만듭니다.

```
import time
import serial
import smtplib

TO = 'putyour@email.here'
GMAIL_USER = 'putyour@email.here'
GMAIL_PASS = 'putyourpasswordhere'

SUBJECT = 'Intrusion!!'
TEXT = 'Your PIR sensor detected movement'
ser = serial.Serial('COM4', 9600)

def send_email():
  print("Sending Email")
  smtpserver = smtplib.SMTP("smtp.gmail.com",587)
  smtpserver.ehlo()
```

```
  smtpserver.starttls()
  smtpserver.ehlo
  smtpserver.login(GMAIL_USER, GMAIL_PASS)
  header = 'To:' + TO + '₩n' + 'From: ' + GMAIL_USER
  header = header + '₩n' + 'Subject:' + SUBJECT + '₩n'
  print header
  msg = header + '₩n' + TEXT + ' ₩n₩n'
  smtpserver.sendmail(GMAIL_USER, TO, msg)
  smtpserver.close()
while True:
  message = ser.readline()
  print(message)
  if message[0] == 'M' :
     send_email()
  time.sleep(0.5)
```

파이선 코드에서 맨 위에 이메일과 구글 사용자 계정등 필요한 정보를 자신에 맞게 수정합니다. 위의 프로그램은 구글 이메일 기능을 이용하므로 만약 구글 계정이 없다면 하나 만드셔야 합니다.

To 뒤에 나오는 이메일 주소를 알람을 받을 주소로 변경합니다. GMAIL_USER 뒤에 나오는 이메일 주소를 본인의 gmail주로로 변경합니다. 암호도 변경하여 줍니다.

아래와 같이 코드를 정정하여 자신의 컴퓨터에 설치된 아두이노의 시리얼 포트를 설정하여 줍니다.

```
ser = serial.Serial('COM4', 9600)
```

아두이노의 com포트는 IDE의 오른쪽 하단에 나타나 있습니다. 아래는 리눅스에서 IDE를 실행시켰을 때 나오는 화면으로 /dev/tty.usbmodem621에 연결된 것을 볼수 있습니다.

이러한 내용을 자신의 환경에 맞게 변경하였으며 명령창에서 아래와 같이 명령을 입력하여 파이썬 프로그램을 실행합니다.

python movement.py

움직임이 센서에 검출되면 아래와 같은 이메일이 도착하는 것을 확인 할 수 있습니다.