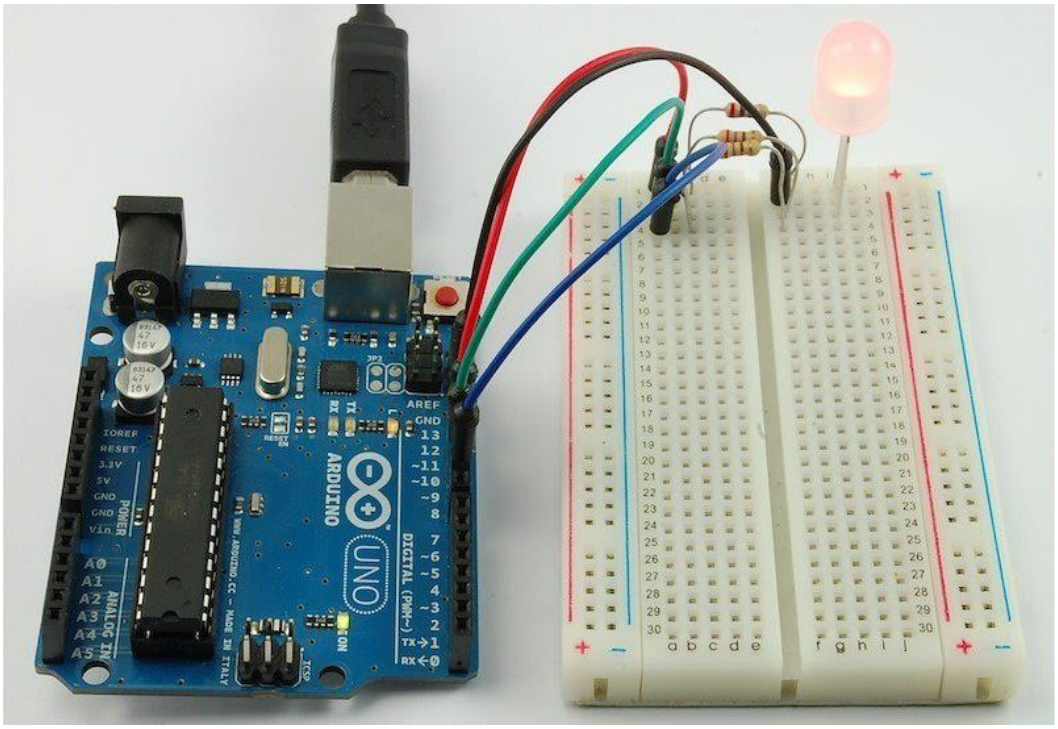
Overview

In this lesson, you will learn how to use a RGB (Red Green Blue) LED with an Arduino.

You will use the *analogWrite* function of Arduino to control the color of the LED.

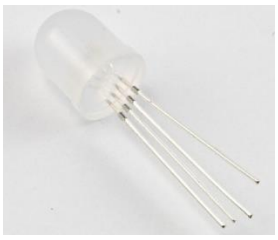


At first glance, RGB (Red, Green, Blue) LEDs look just like regular LEDs, however, inside the usual LED package, there are actually three LEDs, one red, one green and yes, one blue. By controlling the brightness of each of the individual LEDs you can mix pretty much any color you want.

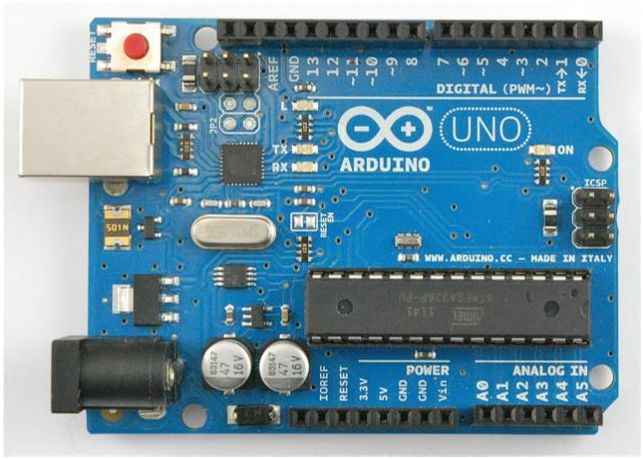
We mix colors just like you would mix audio with a 'mixing board' or paint on a palette - by adjusting the brightness of each of the three LEDs. The hard way to do this would be to use different value resistors (or variable resistors) as we played with in lesson 2. That's a lot of work! Fortunately for us, the Arduino has an *analogWrite* function that you can use with pins marked with a ~ to output a variable amount of power to the appropriate LEDs.

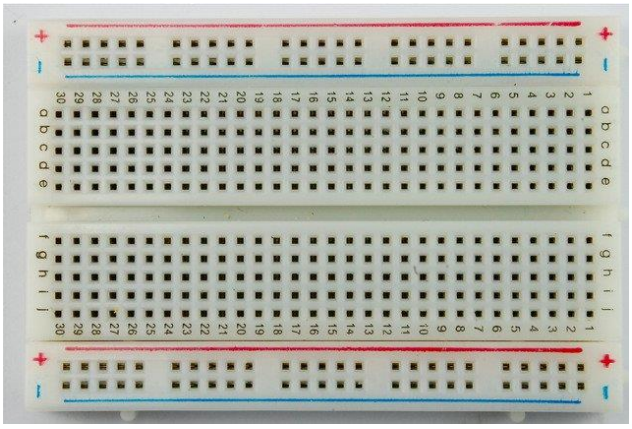
Parts

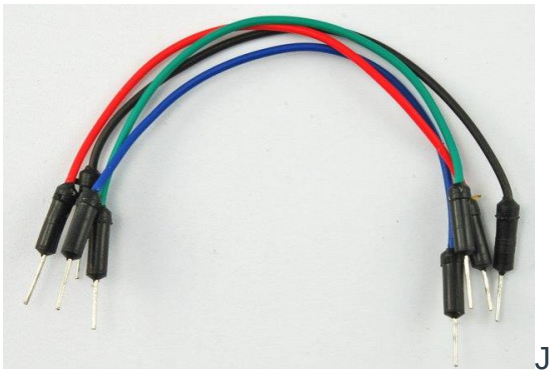
To build the project described in this lesson, you will need the following parts.

 RGB LED - 1

 1 220 ohm resistor (red, red, black, black, brown) - 3

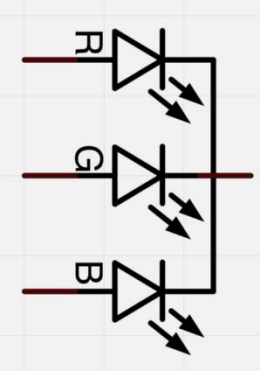
 Arduino UNO R3 - 1

 breadboard - 1

 Jumper wires - 4

Breadboard Layout

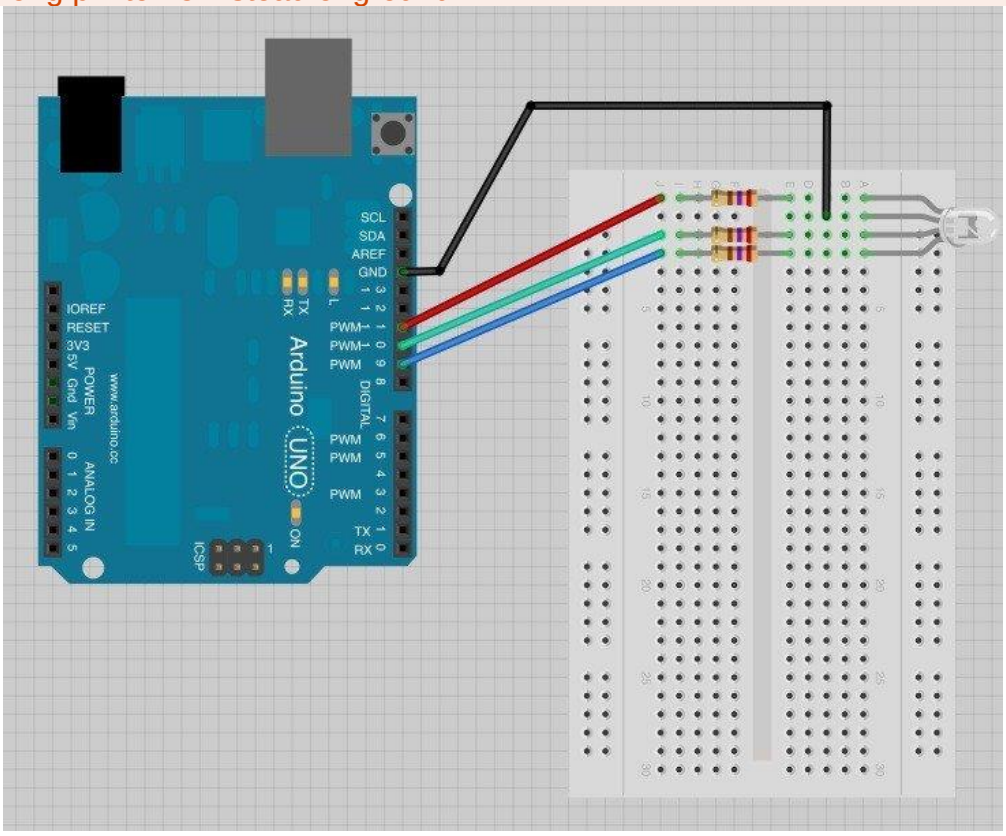
The RGB LED has four leads. There is one lead going to the positive connection of each of the single LEDs within the package and a single lead that is connected to all three negative sides of the LEDs.



The common negative connection of the LED package is the second pin from the flat side of the LED package. It is also the longest of the four leads. This lead will be connected to ground.

Each LED inside the package requires its own 270Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to Arduino output pins using these resistors.

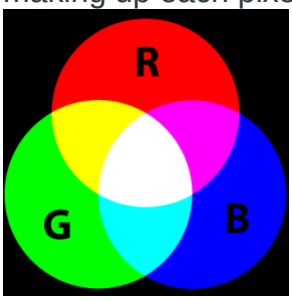
If you are using a common ANODE LED instead of common CATHODE, connect the long pin to +5 instead of ground



Colors

The reason that you can mix any color you like by varying the quantities of red, green and blue light is that your eye has three types of light receptor in it (red, green and blue). Your eye and brain process the amounts of red, green and blue and convert it into a color of the spectrum.

In a way, by using the three LEDs we are playing a trick on the eye. This same idea is used in TVs, where the LCD has red, green and blue color dots next to each other making up each pixel.



If we set the brightness of all three LEDs to be the same, then the overall color of the light will be white. If we turn off the blue LED, so that just the red and green LEDs are the same brightness, then the light will appear yellow.

We can control the brightness of each of the red, green and blue parts of the LED separately, making it possible to mix any color we like.

Black is not so much a color as an absense of light. So the closest we can come to black with our LED is to turn off all three colors.

Arduino Sketch

Get a copy of the sketch from your teacher. You will type this into the Arduino program on your computer.

The sketch will cycle through the colors red, green, blue, yellow, purple, and aqua. These colors being some of the standard Internet colors.

The sketch starts by specifying which pins are going to be used for each of the colors:

```
1.  int redPin = 11;
2.  int greenPin = 10;
3.  int bluePin = 9
```

The next step is to write the 'setup' function. As we have learned in earlier lessons, the setup function runs just once after the Arduino has reset. In this case, all it has to do is define the three pins we are using as being outputs.

Then create your 'loop' function. Look at the last function in the sketch.

This function takes three arguments, one for the brightness of the red, green and blue LEDs. In each case the number will be in the range 0 to 255, where 0 means off and 255 means maximum brightness. The function then calls 'analogWrite' to set the brightness of each LED.

If you look at the 'loop' function you can see that we are setting the amount of red, green and blue light that we want to display and then pausing for a second before moving on to the next color.
Try adding a few colors of your own to the sketch and watch the effect on your LED.

Using Internet Colors

If you have done any Internet programming, you will probably be aware that colors are often represented as a 'hex' number. For example the color red has the number #FF0000. You can find the numbers associated with a particular color using tables like these: https://htmlcolorcodes.com/color-names/ (Links to an external site.)Links to an external site.
The six digits of the number are actually three pairs of numbers; the first pair being the red component of the color, the next two digits the green part and the final pair the blue part. Red is #FF0000, because its maximum red (FF is 255 in hex) and it has no green or blue part.

It would be pretty useful to be able to dial up one of these color numbers so that it is displayed on the RGB LED.

Try to make the color indigo (#4B0082).

The red, green and blue parts of indigo are (in hex) 4B, 00 and 82 respectively. We can plug those into the 'setColor' function like this:

```
setColor(0x4B, 0x0, 0x82); // indigo
```

We have used hex numbers for the three parts of the color by putting '0x' in front of them.

When you are ready, have your teacher check your work. Try adding a few colors of your own to the 'loop' function. Don't forget to add a delay after each one. Try changing the delays to speed up or slow down the color changing.