RGB LED

Overview

RGB LEDs are a fun and easy way to add some color to your projects. Since they are like 3 regular LEDs in one, how to use and connect them is not much different. They come mostly in 2 versions: Common Anode or Common Cathode.

Common Anode uses 5V on the common pin, while Common Cathode connects to ground.

As with any LED, we need to connect some resistors inline (3 total) so we can limit the current being drawn.

Component Required:

1 x Arduino Uno

1 x 830 Tie Points Breadboard

4x M-M wires (Male to Male jumper wires)

1 x RGB LED

3 x 220 ohm resistors

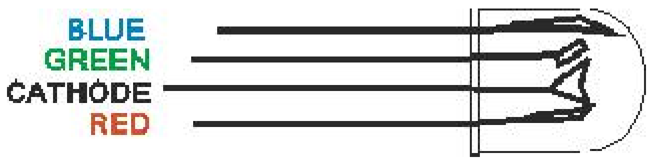
Component Introduction

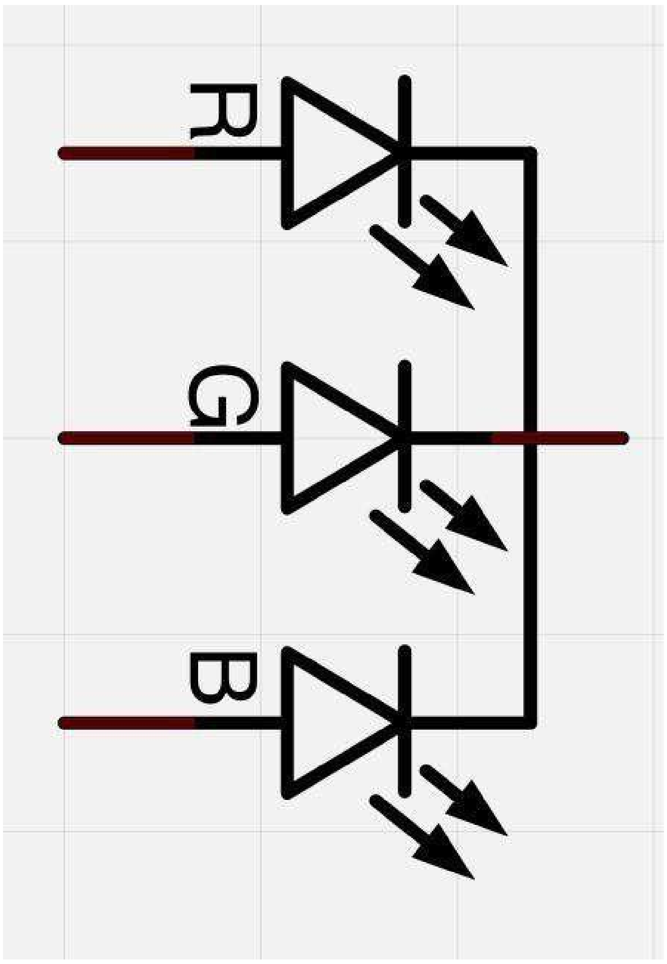
RGB:

At first glance, RGB (Red, Green and 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 the same way you would mix 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 did with in Lesson 2, but that's a lot of work! Fortunately for us, Uno board has an analogWrite function that you can use with pins marked with a ~ to output a variable amount of power to the appropriate LEDs.

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.





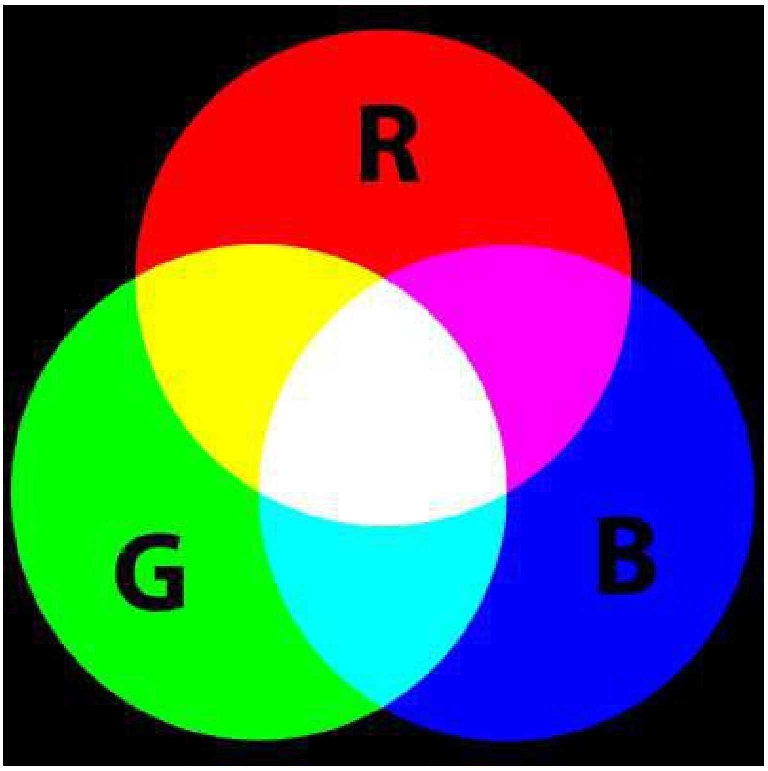
Here on the photographs you can see 4 electrode LED. Every separate pin for Green or Blue or Red color is called Anode. You will always connect "+" to it. Cathode goes to "-" (ground). If you connect it other way round the LED will not light.

The common negative connection of the LED package is the second pin from the flat side. It is also the longest of the four leads and will be connected to the ground. Each LED inside the package requires its own 220Ω 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 UNO output pins using these resistors.

COLOR:

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.

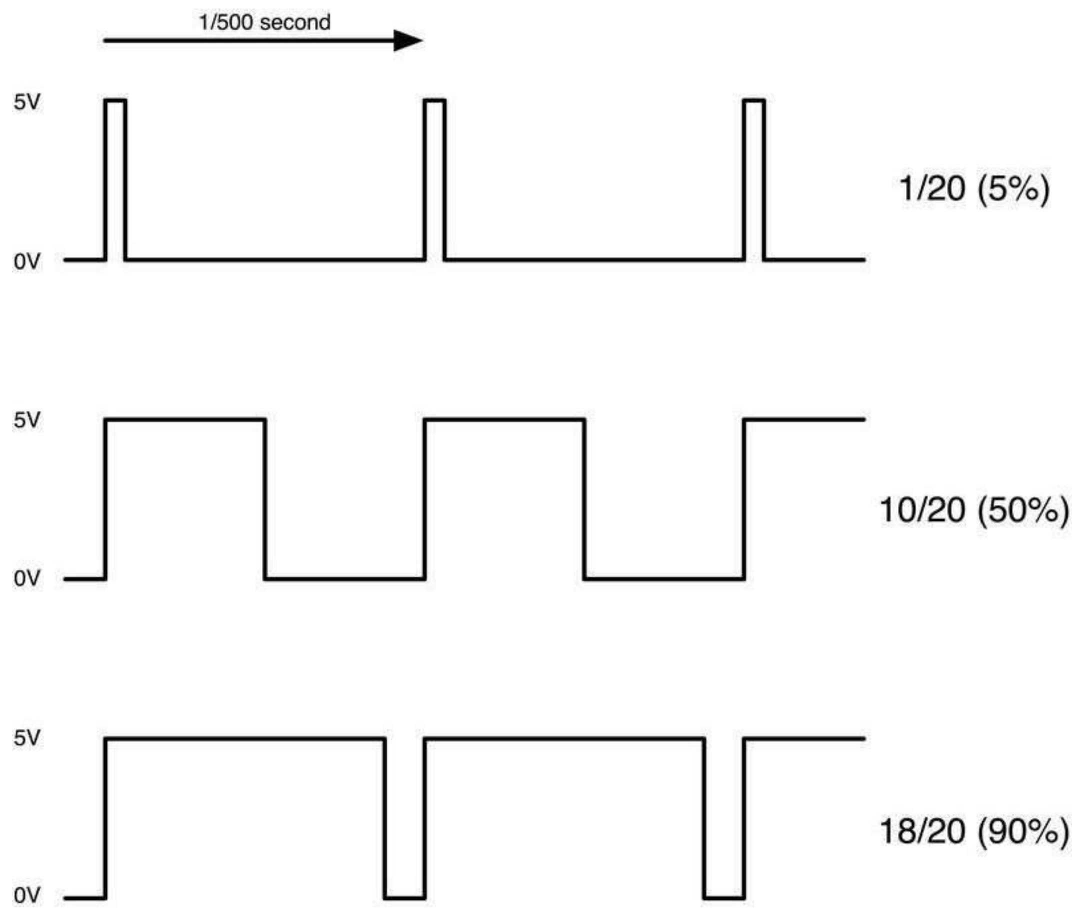
Black is not so much a color as an absence of light. Therefore, the closest we can come to black with our LED is to turn off all three colors.

Theory (PWM)

Pulse Width Modulation (PWM) is a technique for controlling power.

We also use it here to control the brightness of each of the LEDs.

The diagram below shows the signal from one of the PWM pins on the Uno.



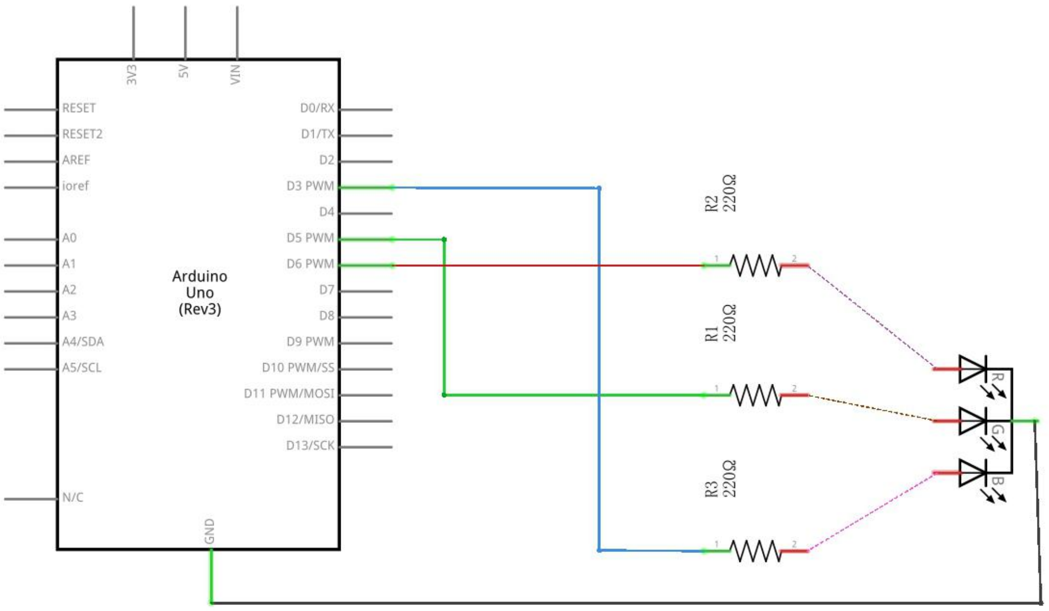
Roughly every 1/500 of a second, the PWM output will produce a pulse. The length of this pulse is controlled by the 'analogWrite' function. So 'analogWrite(0)' will not produce any pulse at all and 'analogWrite(255)' will produce a pulse that lasts all the way until the next pulse is due, so that the output is actually on all the time.

If we specify a value in the analogWrite that is somewhere in between 0 and 255, then we will produce a pulse. If the output pulse is only high for 5% of the time, then whatever we are driving will only receive 5% of full power.

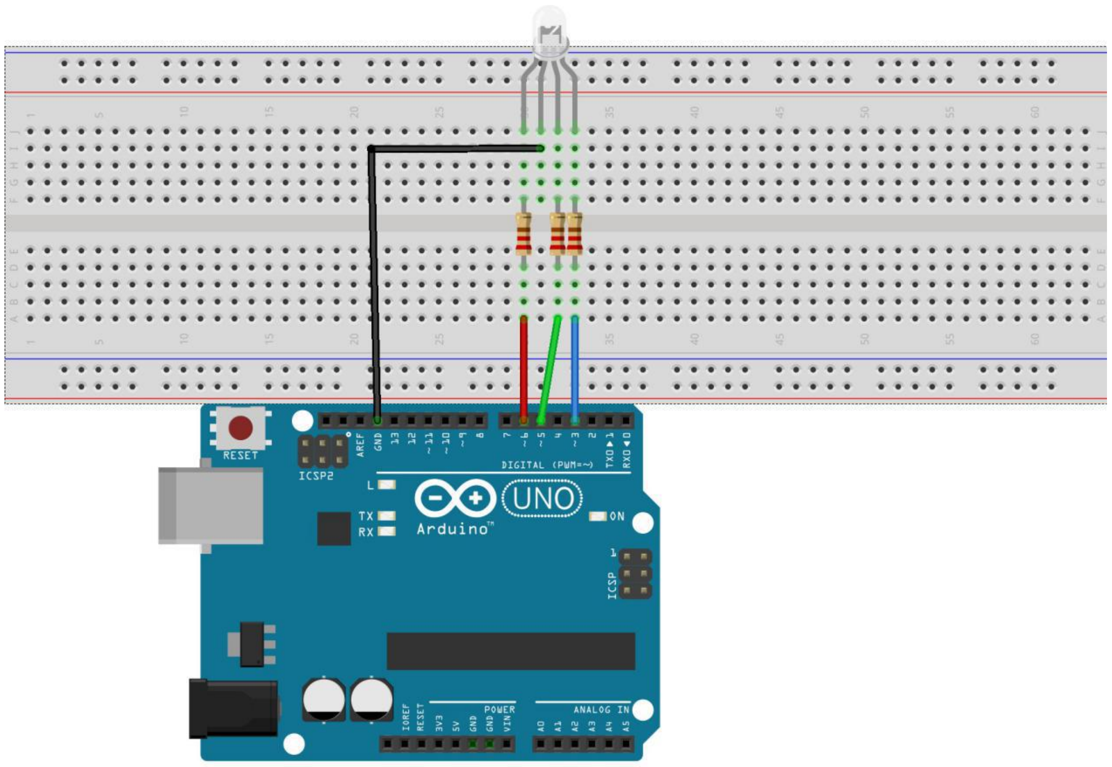
If, however, the output is at 5V for 90% of the time, then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is changing.

Connection

Schematic



Wiring diagram



Code

After wiring, please open the program in the code folder- " RGB LED", and click UPLOAD to upload the program. See "Blink"for details about program uploading if there are any errors.

Our code will use FOR loops to cycle through the colors.

The first FOR loop will go from RED to GREEN.

The second FOR loop will go from GREEN to BLUE.

The last FOR loop will go from BLUE to RED.

Try the sketch out and then we will dissect it in some detail. ....

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

```
// Define Pins

int    BLUE= 3;

Int    GREEN =5;

Int    RED =6;
```

The next step is to write the 'setup' function. As we have learnt 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.

```
void setup()

{
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
digitalWrite(RED, HIGH);
digitalWrite(GREEN, LOW);
digitalWrite(BLUE, LOW);

}
```

Before we take a look at the 'loop' function, let's look at the last function in the sketch.

The define variables

```
redValue = 255; // choose a value between 1 and 255 to change the color.

greenValue = 0;

blueValue = 0;
```

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.

```
#define delayTime 10 // fading time between colors

delay(delayTime);
```

Try adding a few colors of your own to the sketch and watch the effect on your LED.