AD32600 Physical Computing
Prof. Fabian Winkler
Fall 2014

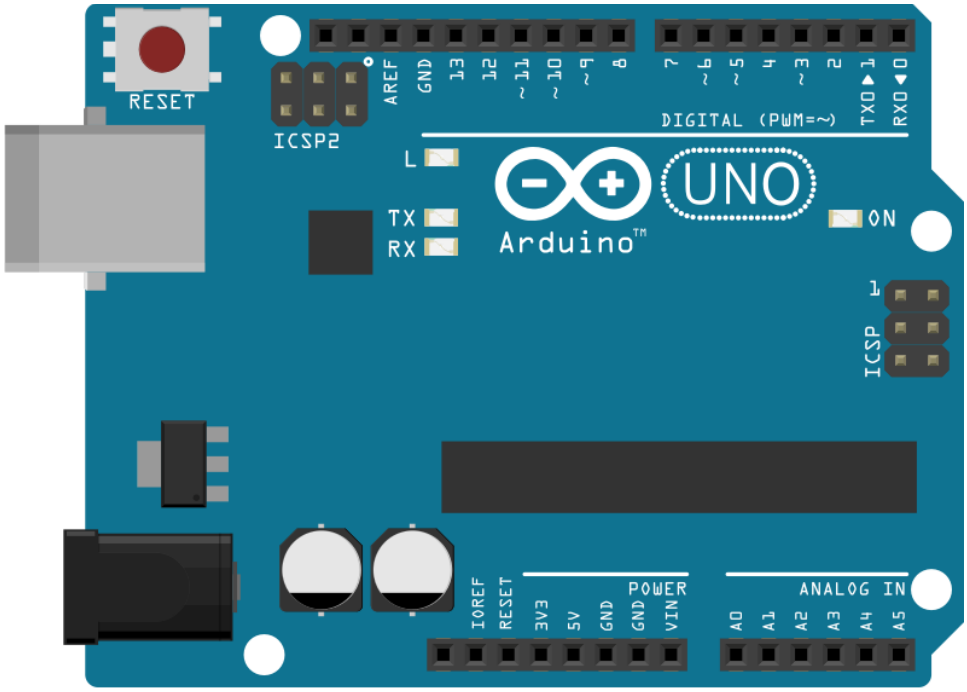
Arduino Workshop 01

This workshop provides an introductory overview of the Arduino board, basic electronic components and closes with a few basic circuits and code examples connecting LEDs to the Arduino board.

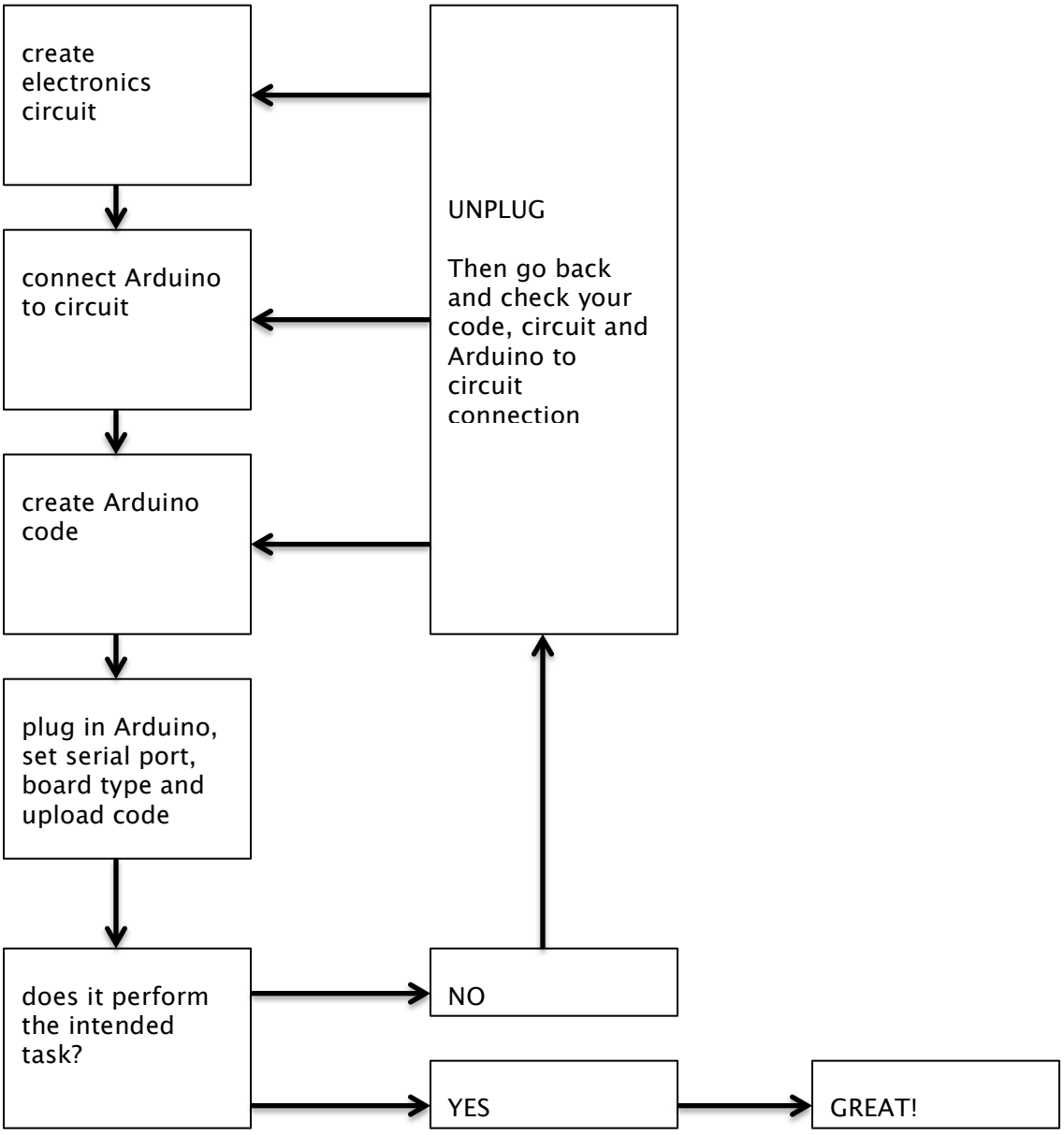
Getting started – the folks at Fritzing made two great introductory tutorial videos (each around 5 minutes in length) that give you a great basic understanding of electricity and electronics (i.e. the force the drives the Arduino board and ways to control and measure it) as well as the anatomy of the Arduino board:

- http://www.youtube.com/watch?v=9E779EfPLI4&list=PL8CD32146ED5CD04E
- http://www.youtube.com/watch?v=xKwox3dd-dE&index=2&list=PL8CD32146ED5CD04E

Here is again a top view of the Arduino board (UNO R3, other types of boards are very similar) with its most important components:



Arduino workflow

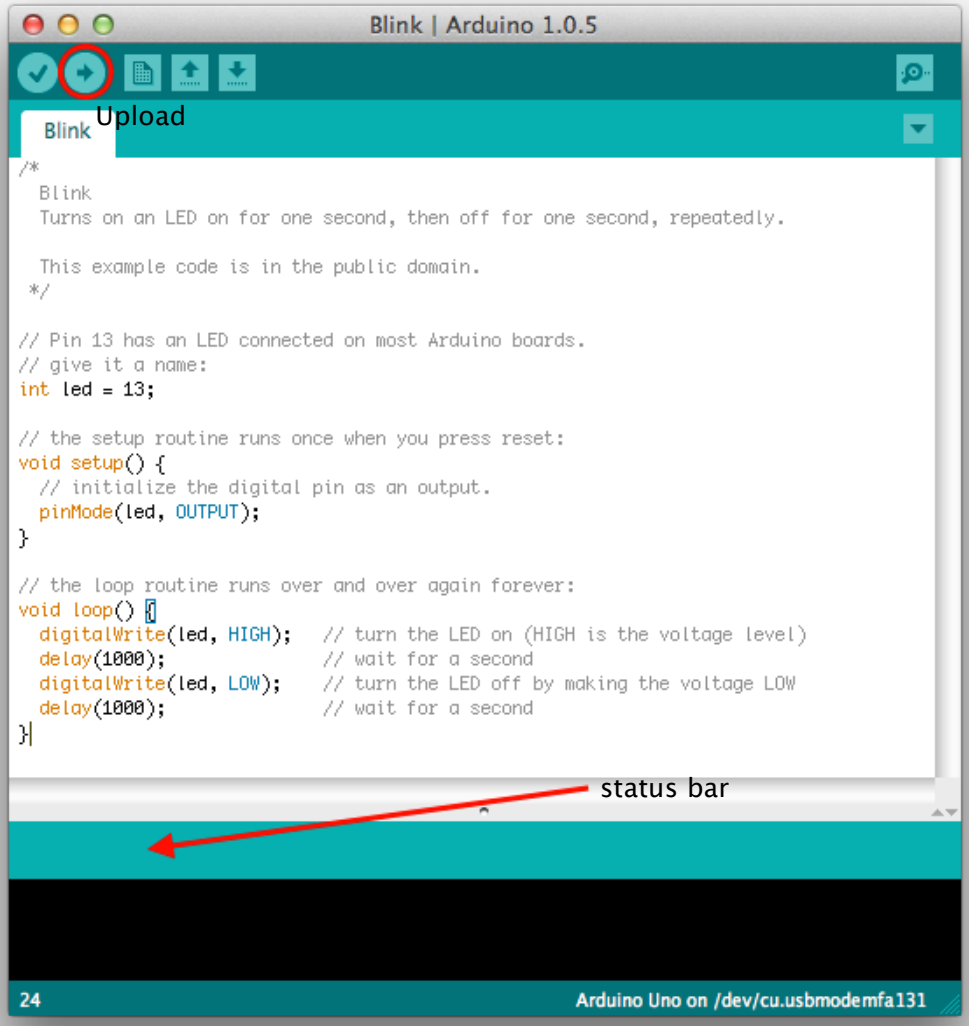


Your first Arduino program

Open up `File > Examples > 01. Basics > BareMinimum`. This will give you the anatomy of a very basic Arduino sketch. You'll immediate recognize its similarity to Processing.

Now open `File > Examples > 01. Basics > Blink`. This code will make the on-board LED blink at a frequency of one second on, one second off. Again compare the similarity to Processing looking at variable declarations, setup() and loop() functions (loop() is called draw() in Processing, otherwise it is identical). The Blink example is actually pre-loaded on all new Arduinos, so without uploading anything, once you plug in your Arduino board the on-board LED should blink accordingly.

Quickly change some of the timing, or add some more time intervals and then upload this sketch with its new behavior.



Connect the Arduino board, make sure that you select the proper board from `Tools > Board > Arduino UNO` (if you are using the UNO, otherwise choose the board you are using from the list) and select the correct serial port: `Tools > Serial Port` on the Macintosh it is something like this `/dev/cu.xxxxxxUSB` on the PC you'll have to try the different COM ports that will show up in the list. Then push the upload button.

Once the code is uploaded you should receive a confirmation about the successful upload of the sketch in your Arduino window's status bar: "Done uploading."

We are now ready to take the next step and look at electronic components and how to connect them to the Arduino board.

Electronics: Concepts, Conventions, Components

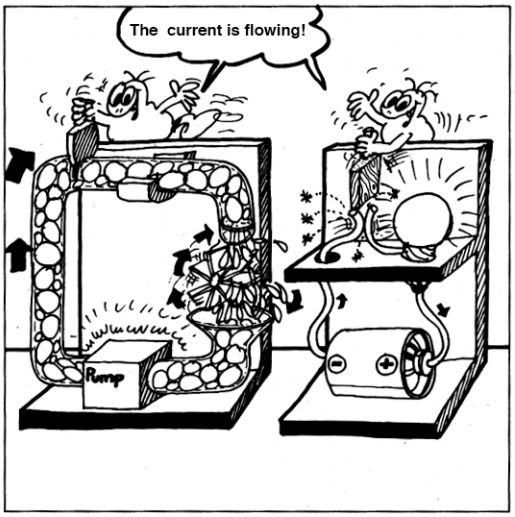
A. Concepts and Conventions

- All our circuits use DC electricity (a battery, wall adapter or power from the computer's USB port), do not connect them to AC (i.e. outlet) electricity.

- For all our circuits we assume that the flow of **electrons** is in one direction only from + (positive) to – (negative). The flow of electrons is called **current**.

- As a convention, positive power is marked with red, negative (or ground, "GND") is marked with black or blue.

- We distinguish between **voltage (V or E)** (potential energy – "pressure" of electrons into the circuit) and **current (I)** (amount of electrons passing a point in the circuit) – look at water analogy.



- Narrow passages in the water circuit would correspond to areas of "resistance" and can be linked to the function of resistors in an electronic circuit which limit the amount of electrons that pass through them at any moment in time.

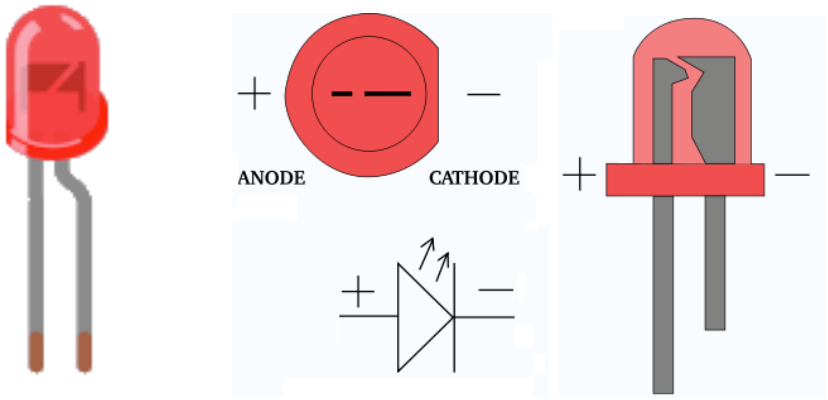
- One of the most important relationships between Voltage, Current and Resistance in an electronic circuit can be expressed mathematically using Ohm's law (named after German physicist Georg Ohm who in a publication in 1827 laid the foundation for this formula): E = I x R. We will see later how this formula is extremely useful, e.g. when we need to calculate resistor values to use with LEDs. See also: https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law

- We distinguish between polarized and non-polarized parts/electric components. The orientation of a non-polarized part in a circuit does not matter, it does matter however for a polarized part.

B. Components

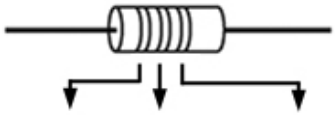
We begin our circuit building with two very basic electronic components and will get to know more the further we move along in our workshop series.

- LEDs – polarized, light up when electrons pass through them



- Resistors – non-polarized, limit amount of electrons that pass through them

Resistor color chart: use this table to determine a resistor's value based on its colored rings

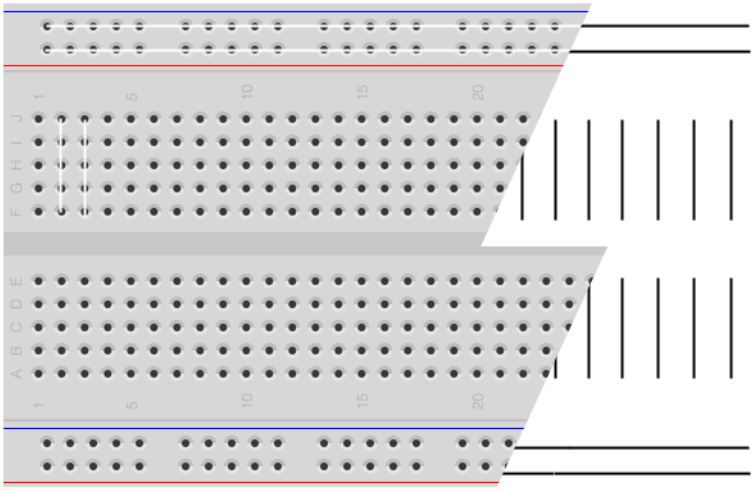


BLACK	0	0	x	1
BROWN	1	1	x	10
RED	2	2	x	100
ORANGE	3	3	x	1,000
YELLOW	4	4	x	10,000
GREEN	5	5	x	100,000
BLUE	6	6	x	1,000,000
VIOLET	7	7	x	10,000,000
GRAY	8	8	x	100,000,000
WHITE	9	9		--

The fourth band indicates accuracy (GOLD = +/- 5%, SILVER = +/- 10%, NO FOURTH BAND = +/- 20%)

C. Building Tips

- How to use the breadboard: look at the picture below to see how rows and columns connect parts on the breadboard

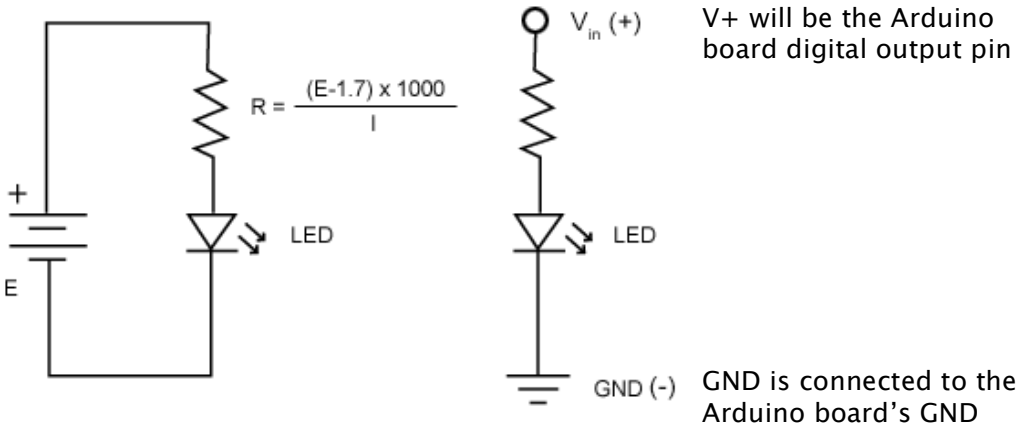


- Keep all your jumper cables, component legs, etc. short, straight and as close as possible to the surface of the breadboard.

- Only modify your circuits when the Arduino board is unplugged. Unplugging/plugging in new components while there is current flowing through your beadboard could possible break electronic parts or your Arduino board

Your first circuit – Blinking Breadboard

How do we light up and LED? Look at the circuit below. Observe the LEDs polarity and use the formula below to determine the resistance of R1 (resistor 1). Look at the colored bands around the resistor to specify its value (see resistor color codes in 1.B. components section.

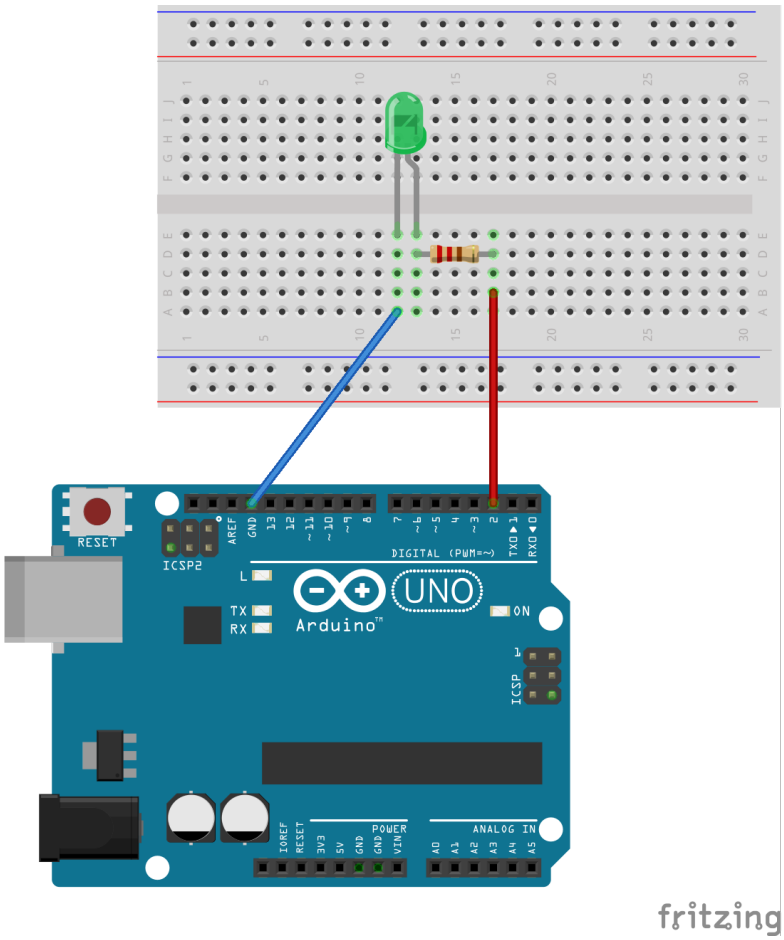


V+ will be the Arduino board digital output pin

GND is connected to the Arduino board's GND

The formula next to resistor R on the left (based on Ohm's law) helps you to determine the correct resistor value for R, so you don't burn the LED. As a rule of thumb you can use 1.7V @ 20mA for standard LEDs (colored, i.e. non-clear plastic cap). Here is what this should look like on your breadboard. Since it is hard to find a 165Ω resistor (and

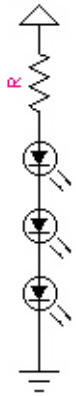
because of value fluctuation of 10-20%) it is safe to **use a 220Ω resistor** with a standard LED:



Change the LED pin in your Arduino sketch from pin 13 to pin 2 and re-upload the code. Here is the code: http://web.ics.purdue.edu/~fwinkler/AD32600_F14/Blink.zip

If you would like to connect more than just one LED you can connect multiple LEDs in **series** or **parallel**, and the value of the resistor changes like this:

Series circuit:



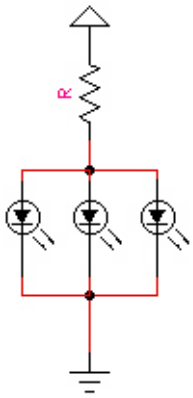
LED1 = LED2 = LED3 = 1.7V, 20mA

$$R = \frac{(E - (\text{number of LEDs} \times 1.7)) \times 1000}{20\ \text{mA}}$$

$$R = \frac{(5V - (3 \times 1.7)) \times 1000}{20\ \text{mA}} = 0\Omega$$

So this would be the maximum number of LEDs you can drive in series from a single Arduino pin (the output voltage of the Arduino pins goes only to 5V).

Parallel circuit:



LED1 = LED2 = LED3 = 1.7V, 20mA

$$R = \frac{(E - 1.7) \times 1000}{\text{number of LEDs} \times 20\ \text{mA}}$$

$$R = \frac{(5V - 1.7)) \times 1000}{3 \times 20\ \text{mA}} = 55\Omega$$

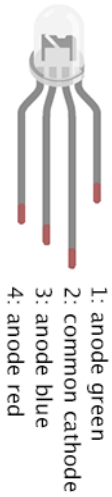
Again, 55 Ω is an odd value for a resistor. Also, including possible imprecision, R=100Ω would be a safe choice. **However, with this example it is important to watch the current. Any single digital pin can only source (provide positive current) or sink (provide negative current) up to 40 mA (milliamps) of current to other devices/circuits. So with our 60mA we are running the danger to break our Arduino board!**

What if you still wanted to drive more than 3 LEDs from any Arduino pin? You will need to use a transistor that will be turned on or off by the pin but directs much larger amounts of voltage and/or current to series and parallel arrangements of LEDs. We will learn about transistors and get back to this specific example in the Arduino workshop on output.

RGB Tri-color LED

In this final example we are using a tri-color (or RGB) LED, Jameco part no. 2125181. This component has 3 LEDs embedded in one package that will allow us to mix their colors based on the RGB (additive) color mixing principle using Arduino code. Going to the Jameco website you can find the part's datasheet here:
http://www.jameco.com/Jameco/Products/ProdDS/2125181.pdf

The datasheet is an important document that tells you about a component's dimensions, pinout and pin functionality as well as voltage and current ratings.



From this part's datasheet we learn that this LED has a "common cathode" e.g. one pin that is connected to GND and individual anodes for the red (ultra orange), green (ultra pure green) and blue (ultra blue) that need to be connected via resistors to the supply voltage (V+). The voltage range for these pins at a current of 20mA is as follows:

red: 2.25 – 2.8V
green: 3.5 – 4.0V
blue: 3.5 – 4.0V

Using the above formula to determine the resistors needed at a supply voltage of +5V (from the Arduino board):

red: (5.0V – 2.5V)/20mA x 1000 = 125Ω
green: (5.0V – 3.7V)/20mA x 1000 = 65Ω
blue: (5.0V – 3.7V)/20mA x 1000 = 65Ω

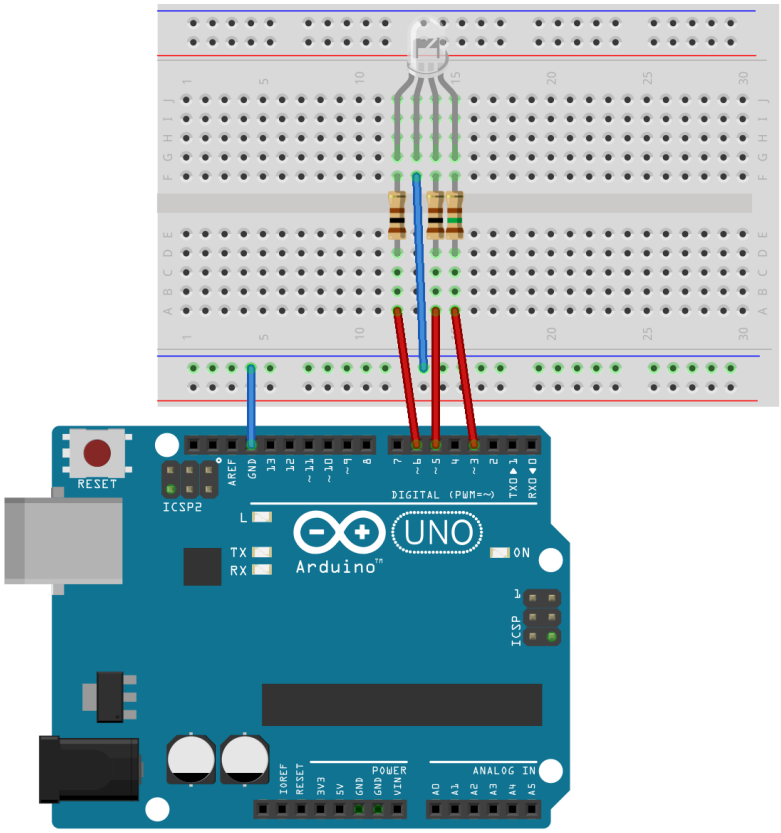
To be on the safer side let's use a 150Ω resistor for the red anode, and 100Ω resistors for the blue and green anodes respectively.

The resulting circuit looks like the one on top of the following page.

Breaking more complex processes into simpler steps

Let's first program the Arduino to first turn each of the LEDs on for a certain amount of time, one after the other. This is a good start to make sure all connections and all code elements work. It is often a good strategy to break down more complex task (do some awesome color mixing with a tri-color LED and Arduino) into simpler steps (turn on/off each LED separately first, then try and turn them on at a percentage of their maximum power individually and finally make the awesome color mixer). This will allow you to check for mistakes early on, an in isolation rather than having to troubleshoot a quite complex code/hardware setup immediately. Try and develop your own ways of breaking up more complex processes into simple steps for your own project work – the workshops in class will help you in developing this skill.

Here is a link to the program for the first step: turning on each LED one after the other:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/Blink_RGB.zip



Fading in/out each LED separately example:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/Fading_RGB.zip

We use pins 3, 5 and 6 since they are pins which allow PWM (pulse width modulation). Pulse Width Modulation is a technique of quickly turning on and off a digital output pin to approximate an analog voltage. See: http://arduino.cc/en/Tutorial/PWM. The above code example is based on `File > Examples > 03. Analog > Fading`

Finally you may want to check out this cross-fading example for the most interesting:
http://www.arduino.cc/en/Tutorial/ColorCrossfader
Just make sure to adjust the pin numbers to your color LEDs (red: pin 3, blue: pin 5 and green: pin 6).

Once we learn how to interface Processing with Arduino we can program simple sliders for the red, green and blue components on the computer screen that allow us to interactively set the color of the LED.