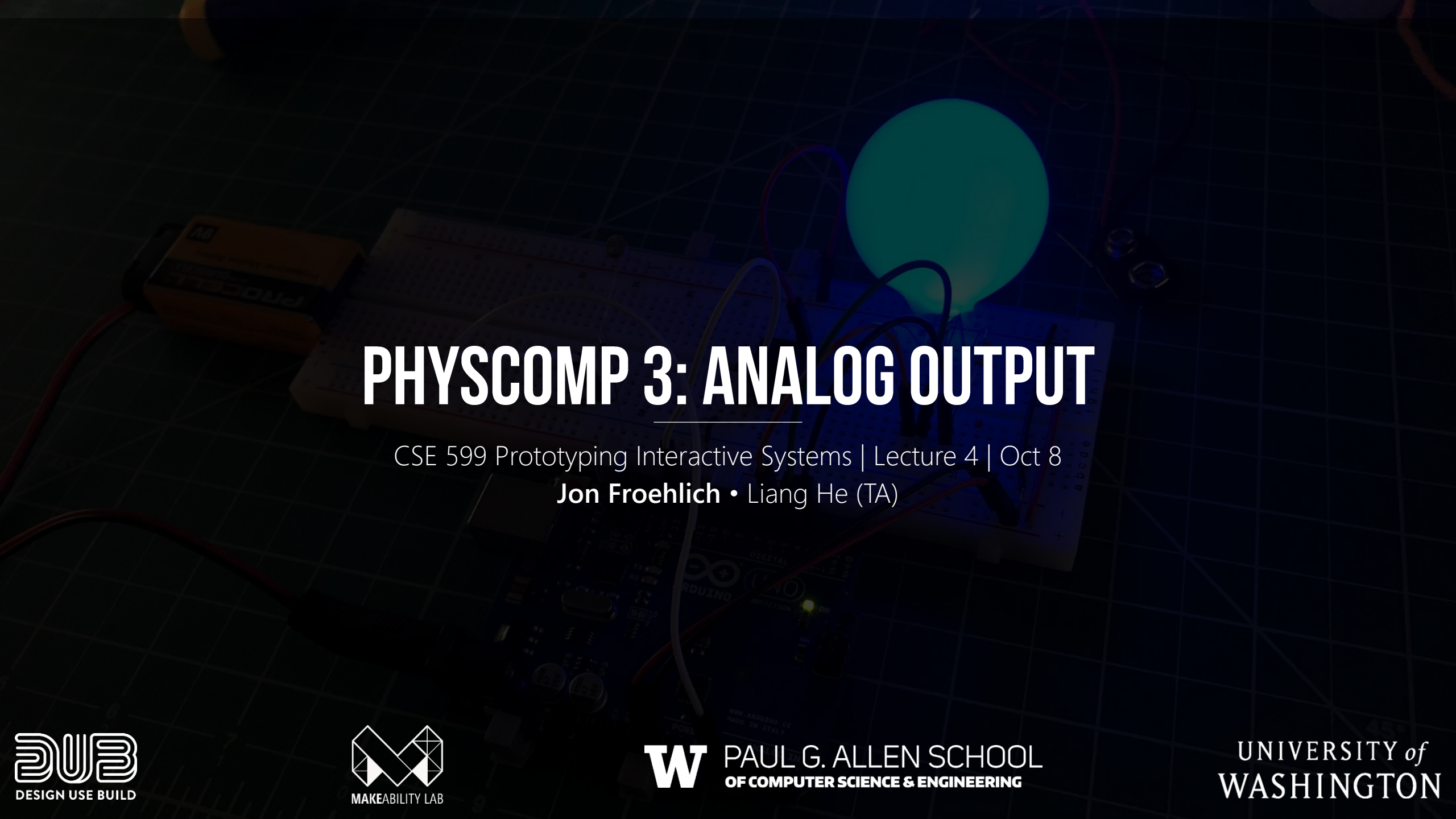


PHYSCOMP 3: ANALOG OUTPUT

CSE 599 Prototyping Interactive Systems | Lecture 4 | Oct 8

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

PHYSICAL COMPUTING 3

TODAY'S LEARNING GOALS

What is **analog output?**

What is **PWM** and why does it matter?

How to use **analogWrite()**

How to use **RGB Leds**

(Partial) introduction to **potentiometers** and **analog input** (if time)

ASSIGNMENT 1

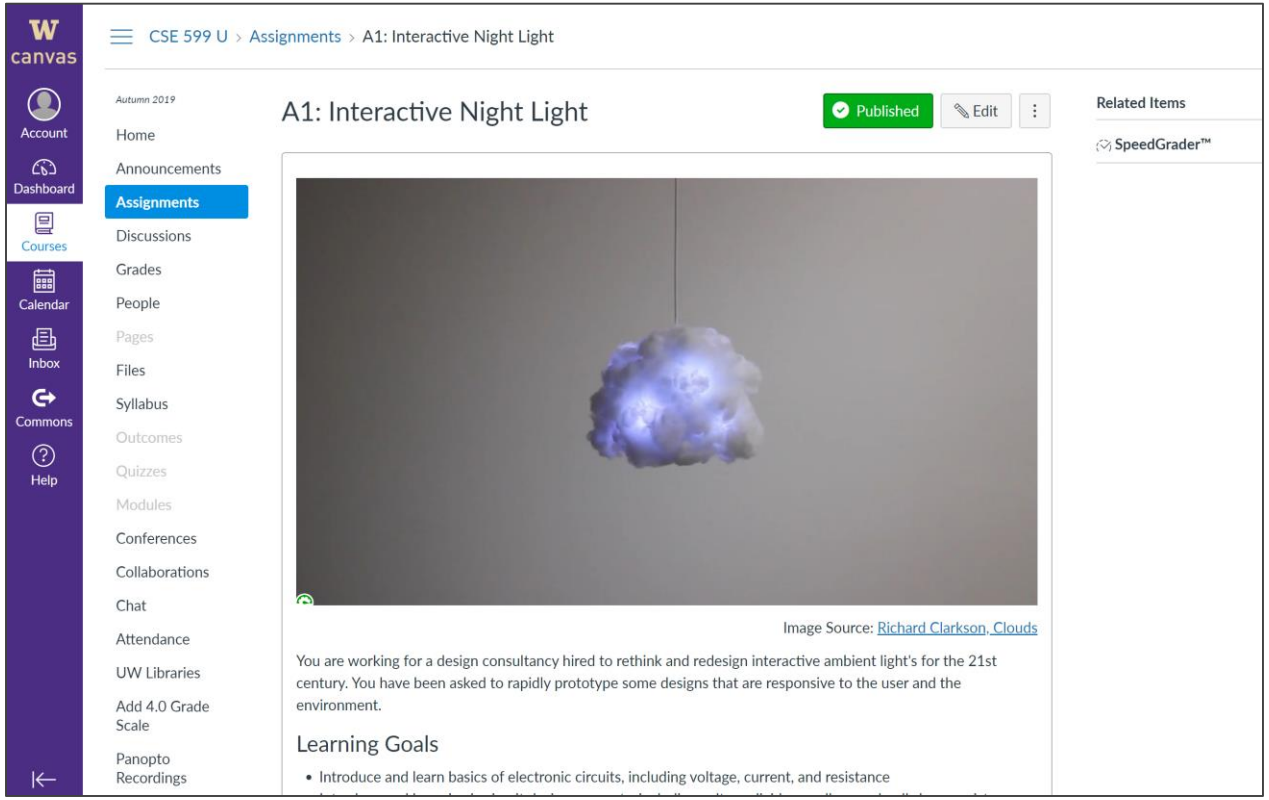
A1: INTERACTIVE NIGHT LIGHT

Due Oct 15th (one week!)

We'll do live demos + feedback

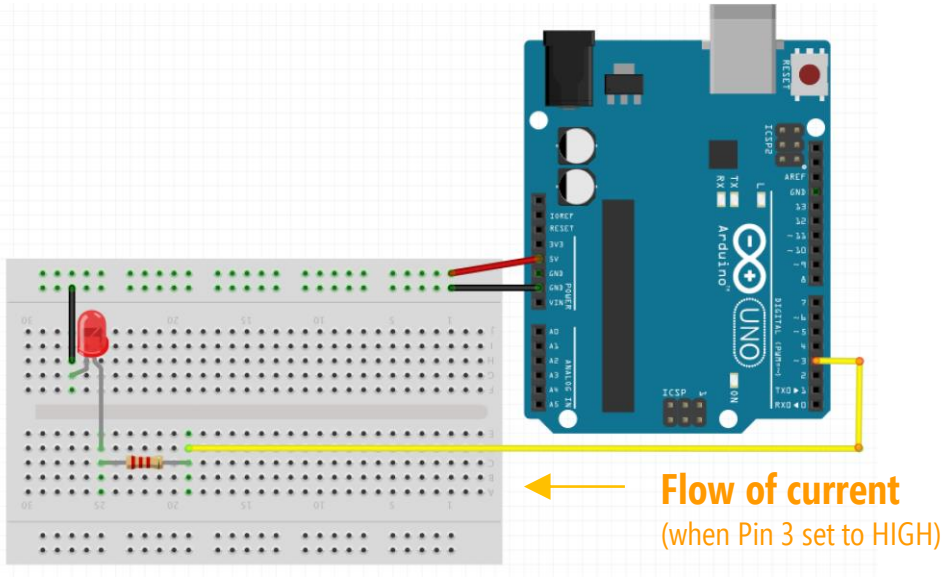
You should be working on this!

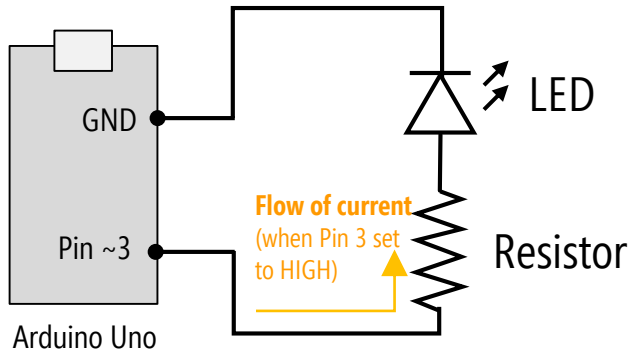
I want you to learn concepts on your own through making and reinforce concepts from lecture



LAST TIME: WE BUILT THIS! LET'S DO IT AGAIN…

Build Circuit: Flash LED on/off via the pin 3



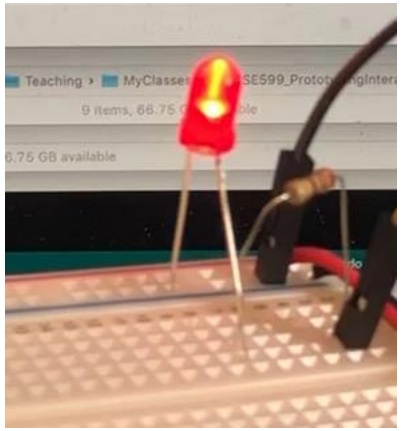


Write Code: Flash LED on/off via the pin 3

Blink | Arduino 1.8.8

Blink

```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

const int LED_OUTPUT_PIN = 3;

// The setup function runs once when you press reset or power the board
void setup() {
  // Because pins 0 - 13 can either be input or output, we must specify
  // how we're using the pin by using pinMode. In this case, we want to
  // control an LED, so set the pin to OUTPUT
  pinMode(LED_OUTPUT_PIN, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_OUTPUT_PIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                          // delay is in milliseconds; so wait one second
  digitalWrite(LED_OUTPUT_PIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                          // wait for a second
}
```



Now let's **fade** our LED on and off. To do this, we have to use **analogWrite** rather than **digitalWrite**.

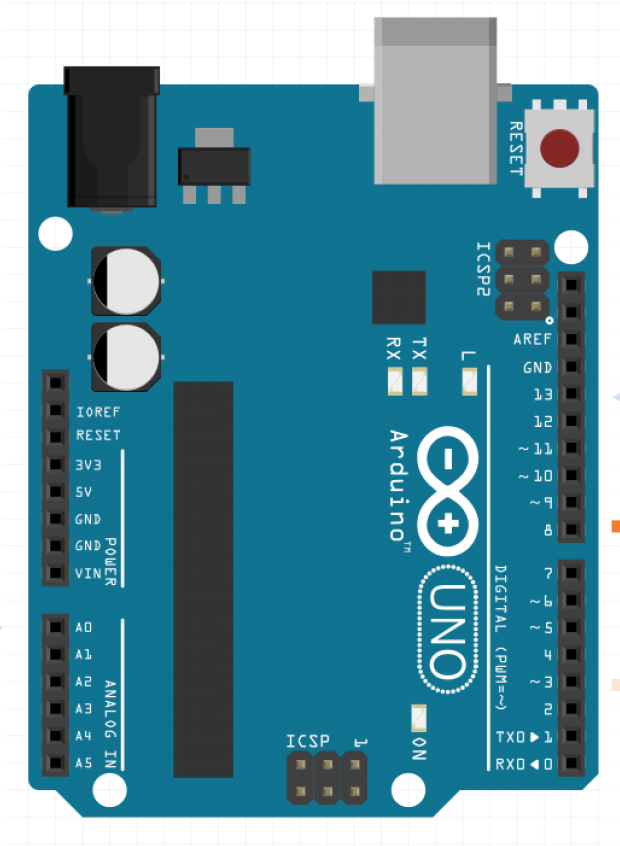
INTRODUCTION TO I/O

INPUT VS. OUTPUT

Input and output is with respect to the microcontroller board. Input is anything that the Arduino Uno reads in (*e.g.*, from a sensor) and output is anything that the Arduino Uno writes out (*e.g.*, to a light or motor)

Analog Input on A0

Reads in any value between 0V or 5V using the analogRead function. In this case, the value of a photocell



Digital Input on pin 13

Reads in a digital input signal (anything below 2.5V converted to LOW, anything above 2.5V converted to HIGH). In this case, the value of switch.

Digital Output on pin 8

Writes out 0V or 5V using digitalWrite function. In this case, turning on and off an LED.

Analog Output on pin 3

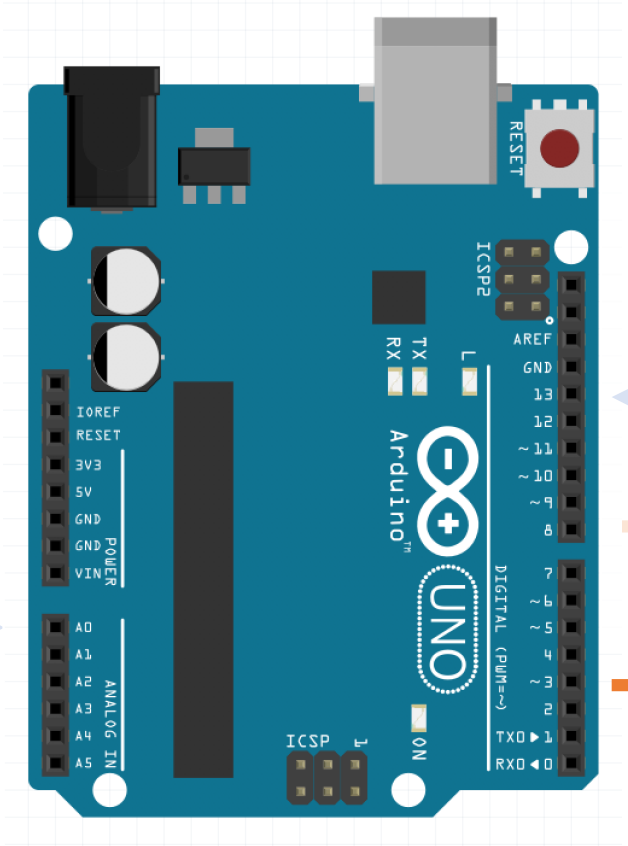
Writes out any value between 0V or 5V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage). Only pins with a tilde ~ can be analog outputs.

INPUT VS. OUTPUT

Input and output is with respect to the microcontroller board. Input is anything that the Arduino Uno reads in (*e.g.*, from a sensor) and output is anything that the Arduino Uno writes out (*e.g.*, to a light or motor)

Analog Input on A0

Reads in any value between 0V or 5V using the analogRead function. In this case, the value of a photocell



Digital Input on pin 13

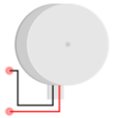
Reads in a digital input signal (anything below 2.5V converted to LOW, anything above 2.5V converted to HIGH). In this case, the value of switch.

Digital Output on pin 8

Writes out 0V or 5V using digitalWrite function. In this case, turning on and off an LED.

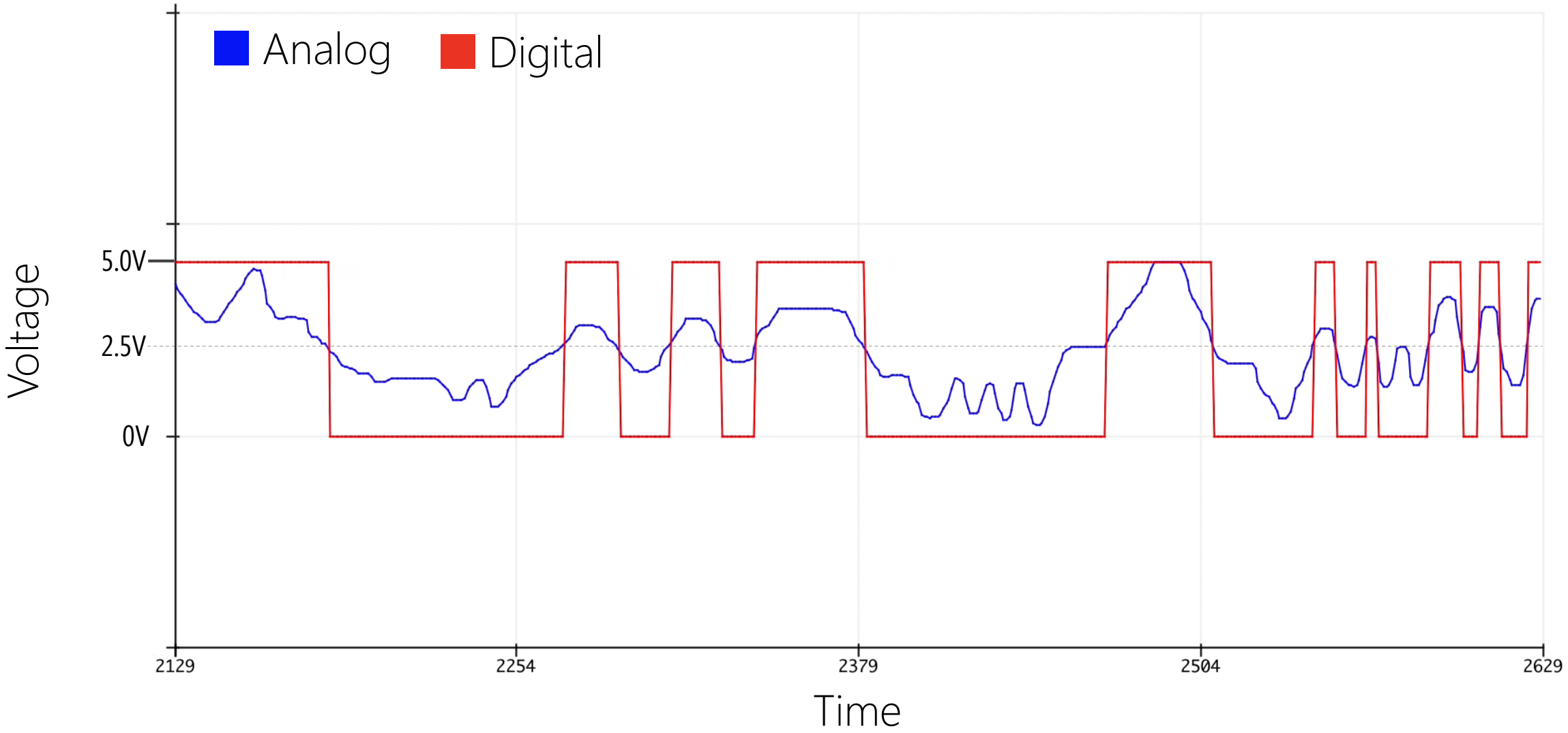
Analog Output on pin 3

Writes out any value between 0V or 5V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage). Only pins with a tilde ~ can be analog outputs.



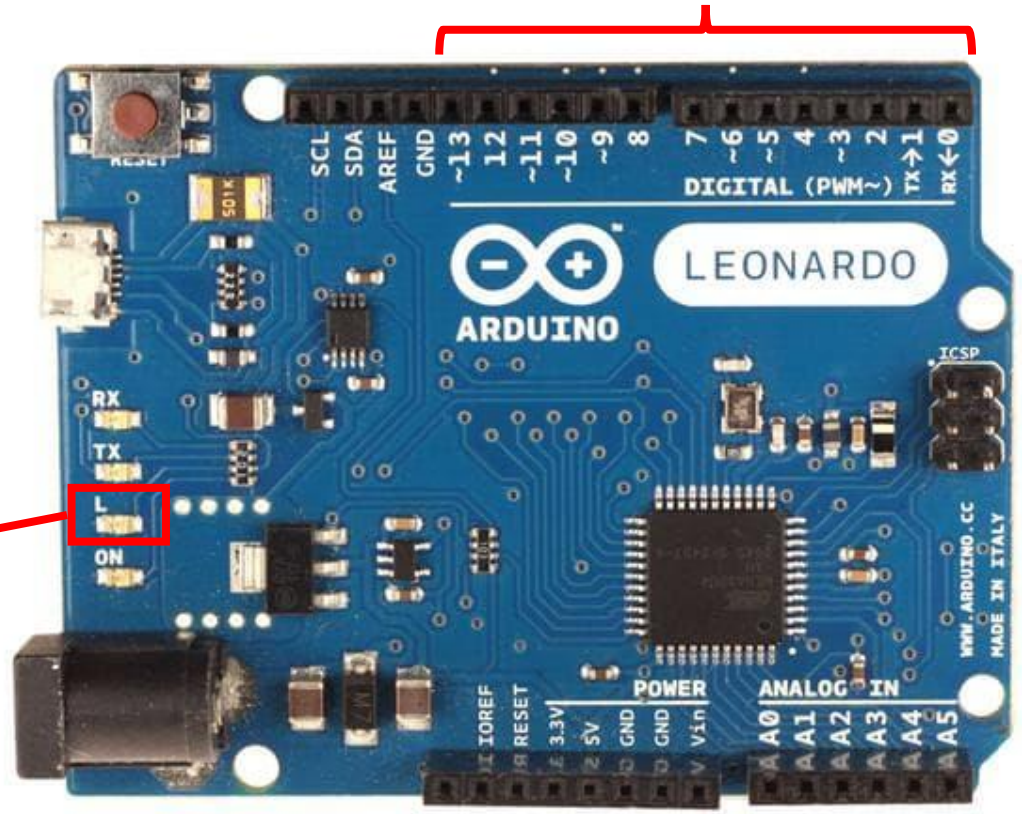
ANALOG VS. DIGITAL

digitalRead and digitalWrite can only be LOW or HIGH corresponding to 0V and 5V.



ARDUINO LEONARDO DIGITAL I/O

14 digital I/O pins: these pins can be used for either input or output (set the mode using the pinMode function). Max output current is 40mA (total across all pins is 200mA)

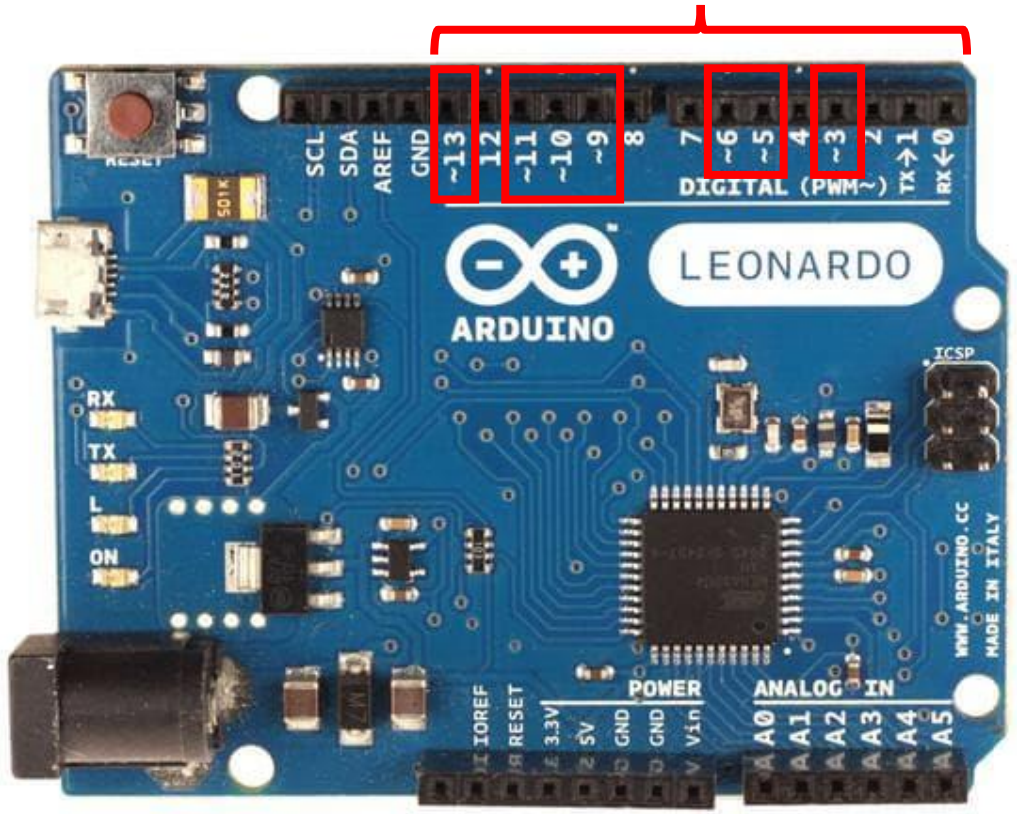


'L' LED: The 'L' LED lights up when pin D13 goes HIGH. This is useful for debugging.

Source: https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro; https://store.arduino.cc/usa/Leonardo; https://learn.sparkfun.com/tutorials/arduino-comparison-guide/atmega32u4-boards

ARDUINO LEONARDO ANALOG OUTPUT

7 analog output pins: pins with the '~' symbol can be used to simulate analog output using 8-bit pulse-width modulation (PWM). Again, max current is 40mA per pin.



Source: https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro; https://store.arduino.cc/usa/Leonardo; https://learn.sparkfun.com/tutorials/arduino-comparison-guide/atmega32u4-boards

DIGITAL OUTPUT

`digitalWrite(pin, value)`

Writes a **HIGH (5V)** or **LOW (0V)** value to a digital pin

Syntax

```
digitalWrite(pin, value)
```

Parameters

`pin`: the pin number

`value`: HIGH (5V) or LOW (0V)

ANALOG OUTPUT

`analogWrite(pin, value)`

Writes an analog value **between 0 and 5V** to a pin using PWM.

Syntax

```
analogWrite(pin, value)
```

Parameters

`pin`: the pin to write to.

`value`: an integer value between 0 & 255 which roughly maps to 0 – 5V on the Arduino Uno & Leonardo.

https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

ANALOG OUTPUT

analogWrite(pin, value)

Writes an analog value between 0 and 5V to a pin using pulse-width modulation (PWM).

Syntax

`analogWrite(pin, value)`

Parameters

`pin`: the pin to write to.

`value`: an integer value between 0 & 255 which roughly maps to 0 – 5V on the Arduino Uno.

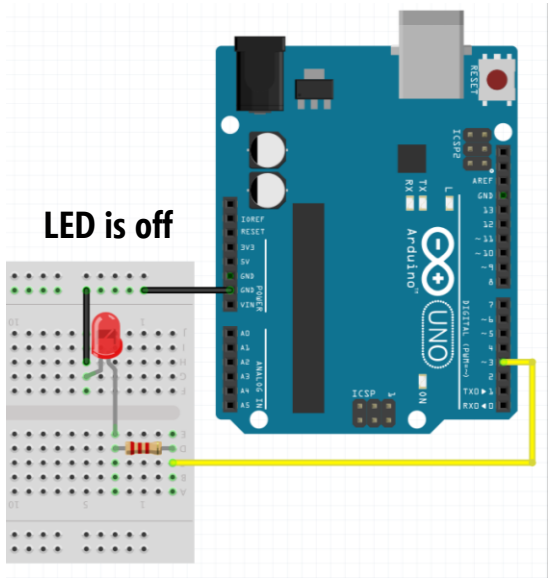
FadeOneDirection §

```
 * Adapted from http://www.arduino.cc/en/Tutorial/Fade
 *
 */

const int LED_OUTPUT_PIN = 3;
int _curBrightness = 0;   // how bright the LED is

// The setup function runs once when you press reset or power the board
void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {

  // set the brightness of the LED pin:
  analogWrite(LED_OUTPUT_PIN, _curBrightness);

  // change the brightness for next time through the loop:
  _curBrightness = _curBrightness + 1;

  // the maximum value that we can write out to analogWrite is 255
  // so check to see if the current brightness value is greater than 255
  // and if it is, reset the brightness to 0 (which is off)
  if (_curBrightness > 255){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

ANALOGWRITE EXAMPLE

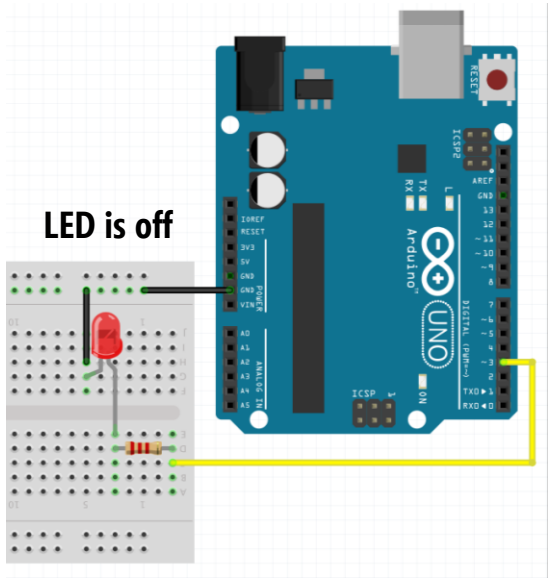
Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V




```
analogWrite(3, ?);
```

ANALOGWRITE EXAMPLE

Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



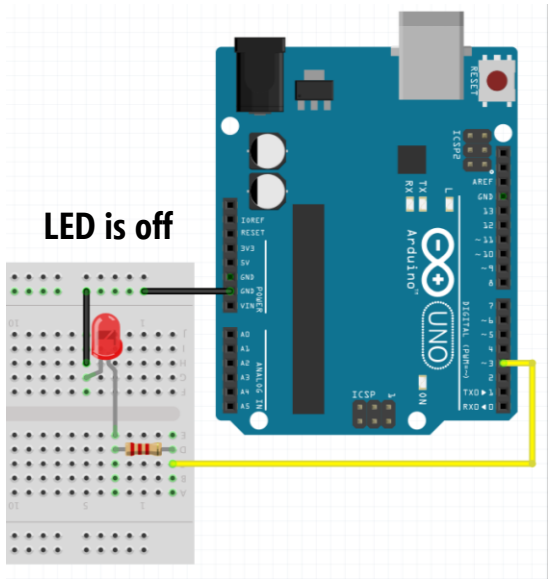

```
analogWrite(3, 0);
```

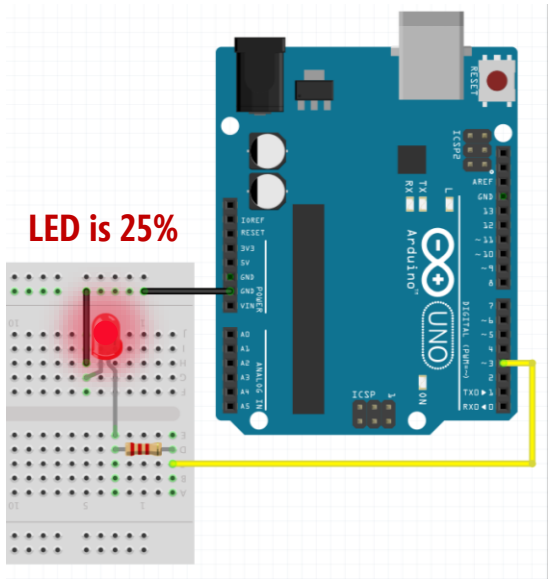
0% brightness (off)

(because 0/255 = 0)

ANALOGWRITE EXAMPLE

Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V






```
analogWrite(3, 0);
```

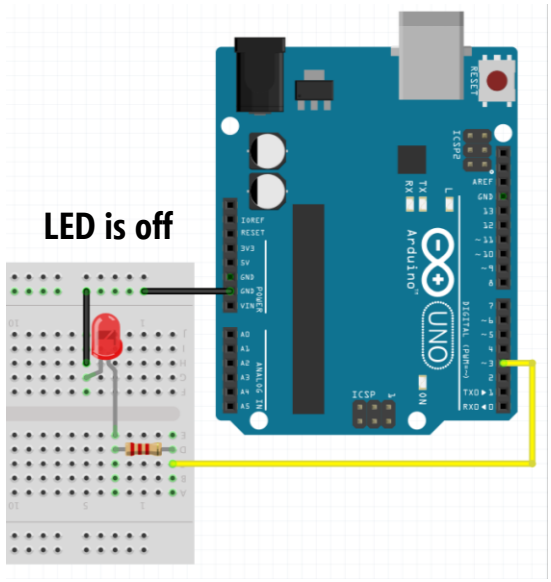
0% brightness (off)

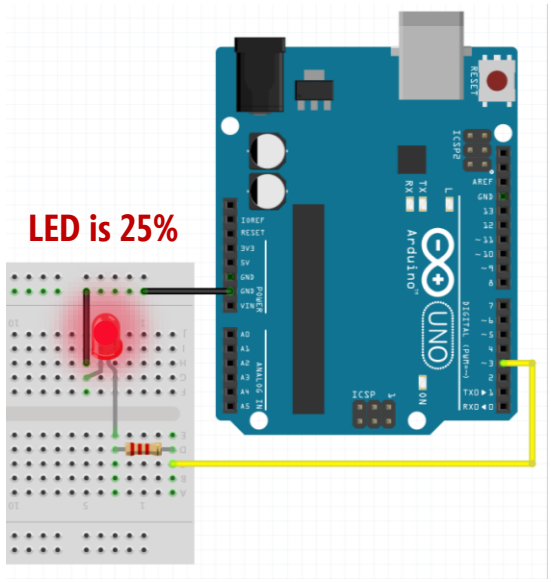
(because 0/255 = 0)

```
analogWrite(3, ?);
```

ANALOGWRITE EXAMPLE

Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



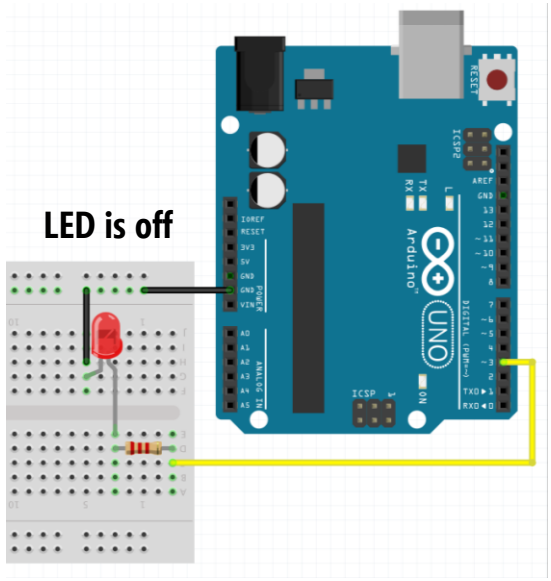


`analogWrite(3, 0);`
0% brightness (off)
(because 0/255 = 0)

`analogWrite(3, 64);`
25% brightness (1.25V)
(because 64/255 = 25%)

ANALOGWRITE EXAMPLE

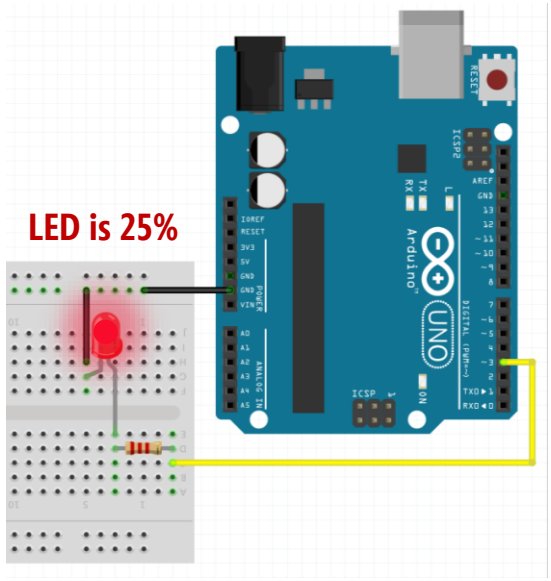
Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



`analogWrite(3, 0);`

0% brightness (off)

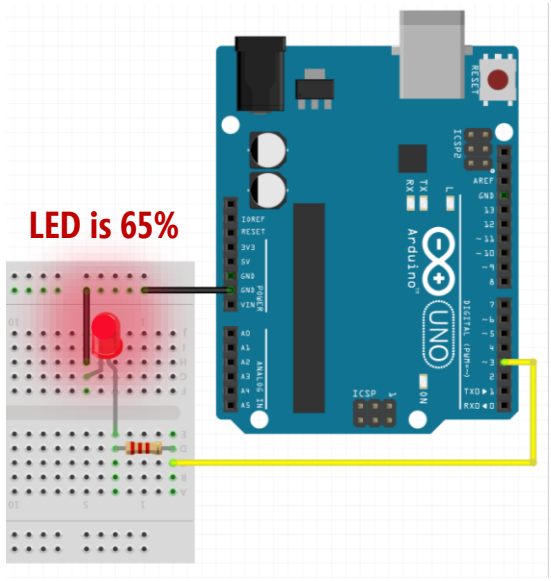
(because 0/255 = 0)



`analogWrite(3, 64);`

25% brightness (1.25V)

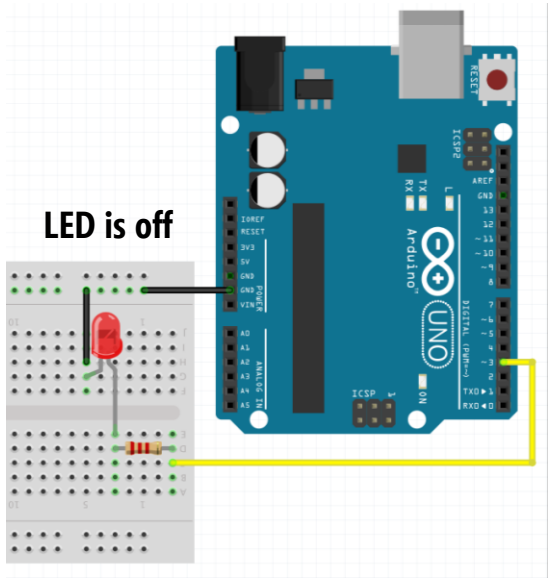
(because 64/255 = 25%)



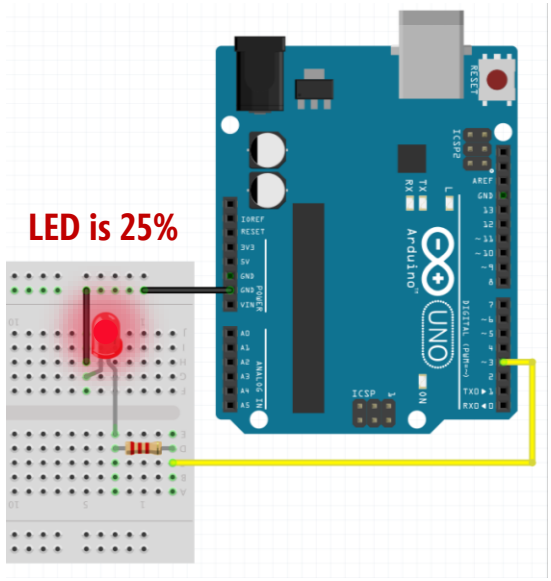
`analogWrite(3, ?);`

ANALOGWRITE EXAMPLE

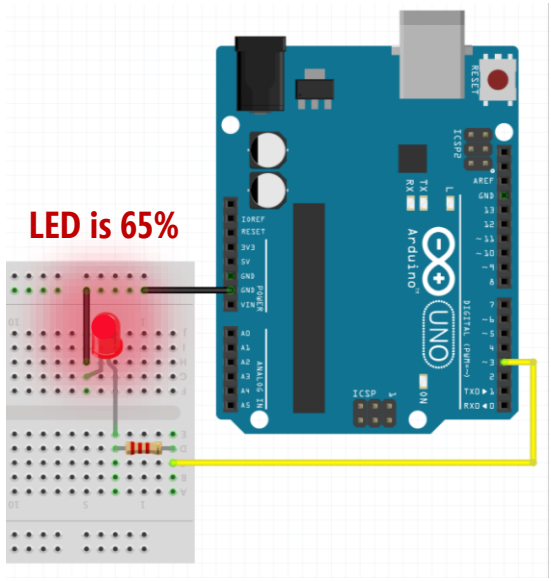
Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



`analogWrite(3, 0);`
0% brightness (off)
(because 0/255 = 0)



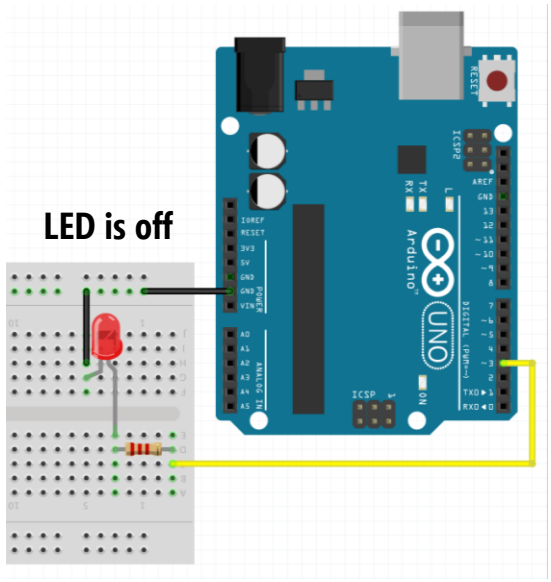
`analogWrite(3, 64);`
25% brightness (1.25V)
(because 64/255 = 25%)



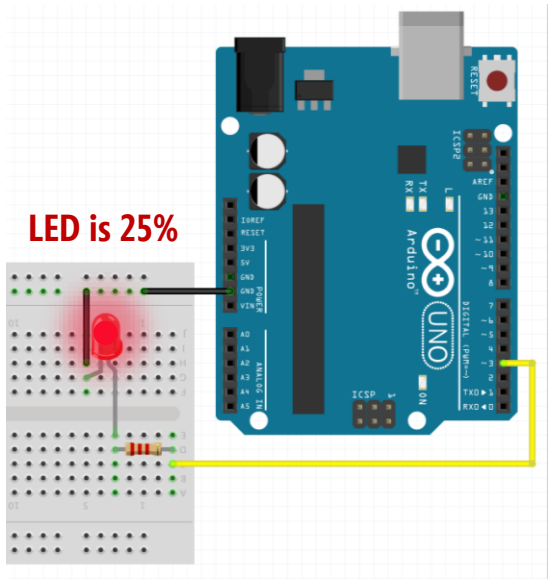
`analogWrite(3, 166);`
65% brightness (3.25V)
(because 166/255 = 65%)

ANALOGWRITE EXAMPLE

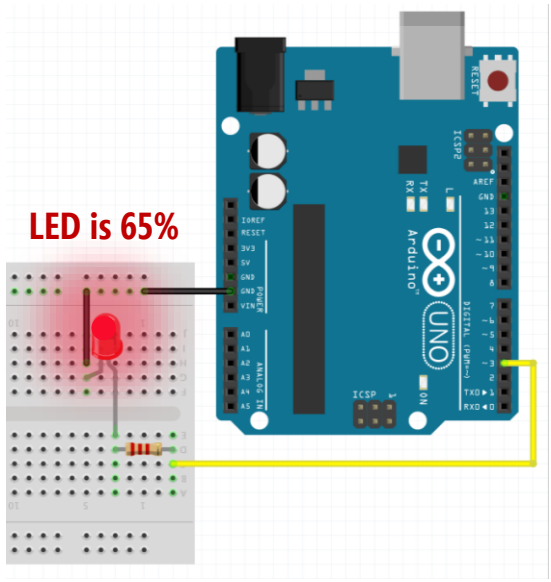
Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



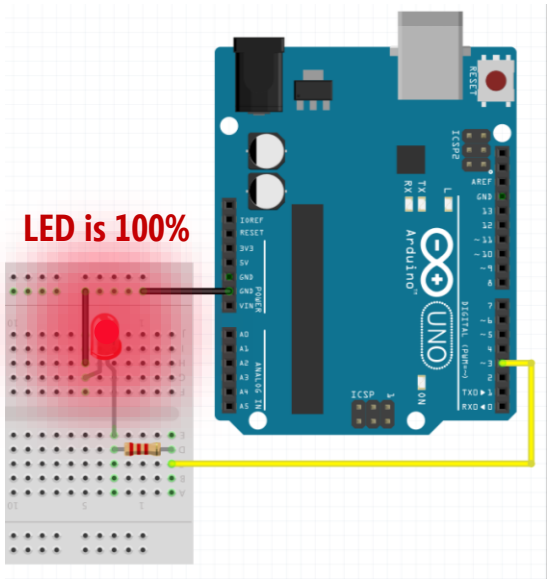
`analogWrite(3, 0);`
0% brightness (off)
(because 0/255 = 0)



`analogWrite(3, 64);`
25% brightness (1.25V)
(because 64/255 = 25%)



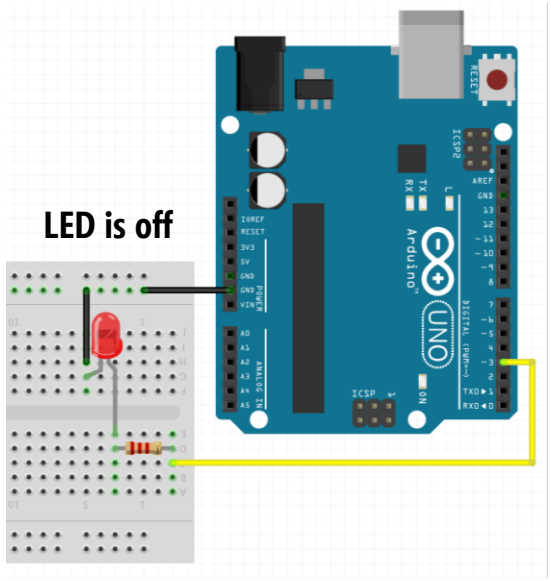
`analogWrite(3, 166);`
65% brightness (3.25V)
(because 166/255 = 65%)



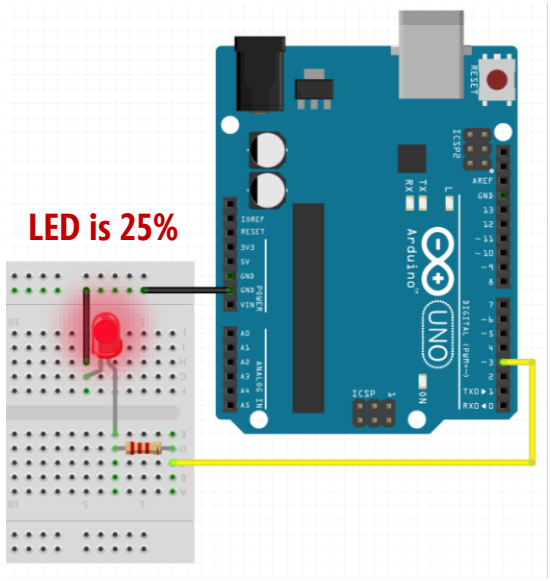
`analogWrite(3, ?);`

ANALOGWRITE EXAMPLE

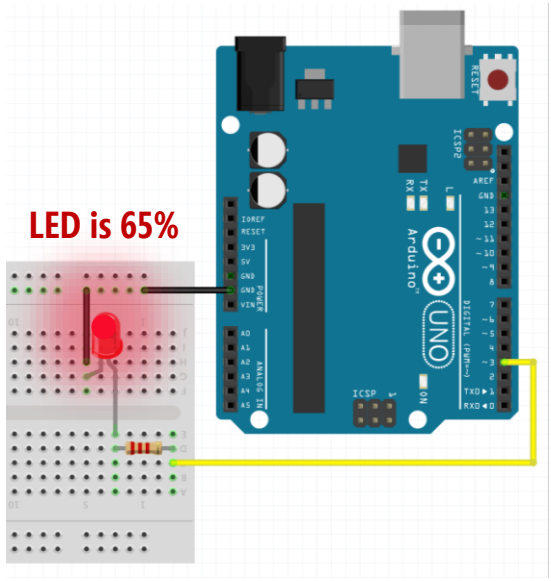
Recall that analogWrite(pin, value) where value is 0 – 255, which is linearly mapped to 0 – 5V



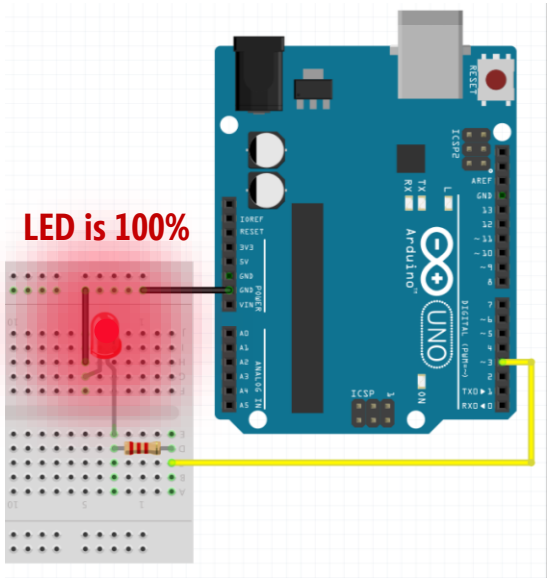
`analogWrite(3, 0);`
0% brightness (off)
(because 0/255 = 0)



`analogWrite(3, 64);`
25% brightness (1.25V)
(because 64/255 = 25%)



`analogWrite(3, 166);`
65% brightness (3.25V)
(because 166/255 = 65%)



`analogWrite(3, 255);`
100% brightness (5V)
(because 255/255 = 100%)

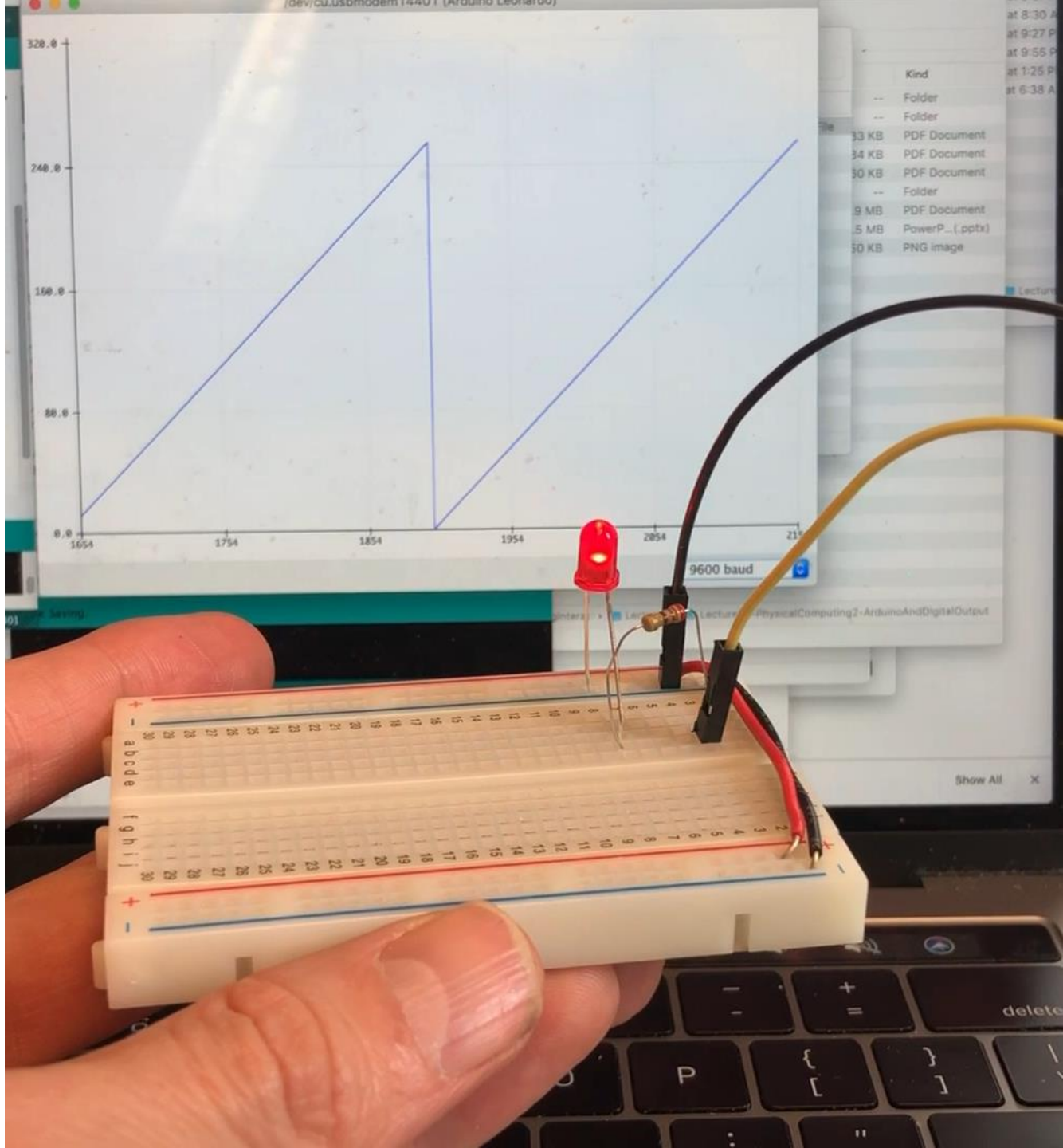
ANALOG OUTPUT

ACTIVITY: FADE LED ON

Fade the LED on using analogWrite

Remember, the **analog output** is limited to 8-bits (0-255)

Try it yourself! Unlike last lecture, I'm not going to first show you the circuit diagram or the code.

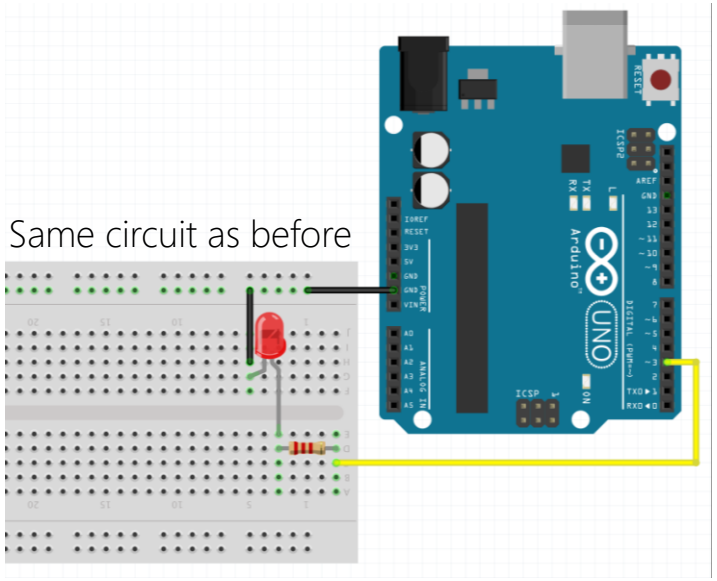


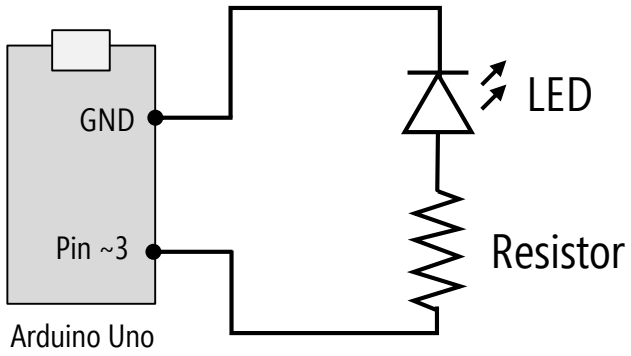
ANALOG OUTPUT

ACTIVITY: FADE LED ON

The circuit is the same as the flashing LED example but the code needs to change.

Circuit: Fade LED on via the pin 3





Code: Fade LED on via pin 3

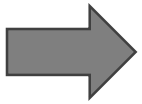
FadeOnSimple §

```
int _curBrightness = 0; // how bright the LED is (between 0 - 255)

void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  // set the brightness of the LED pin
  analogWrite(3, _curBrightness);

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + 1;

  // check to see if we are beyond the max val (255)
  // and if we are, reset the brightness to 0 (which is off)
  if (_curBrightness > 255){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

FADE LED ON WITH FOR LOOP OR WITHOUT?

FadeOnSimple §

```
int _curBrightness = 0; // how bright the LED is (between 0 - 255)

void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  // set the brightness of the LED pin
  analogWrite(3, _curBrightness);

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + 1;

  // check to see if we are beyond the max val (255)
  // and if we are, reset the brightness to 0 (which is off)
  if (_curBrightness > 255){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```



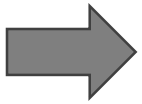
FadeOnForLoop §

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  for (int brightness = 0; brightness < 255; brightness++){
    // set the brightness of the LED pin
    analogWrite(3, brightness);

    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```

FADE LED ON WITH FOR LOOP OR WITHOUT?

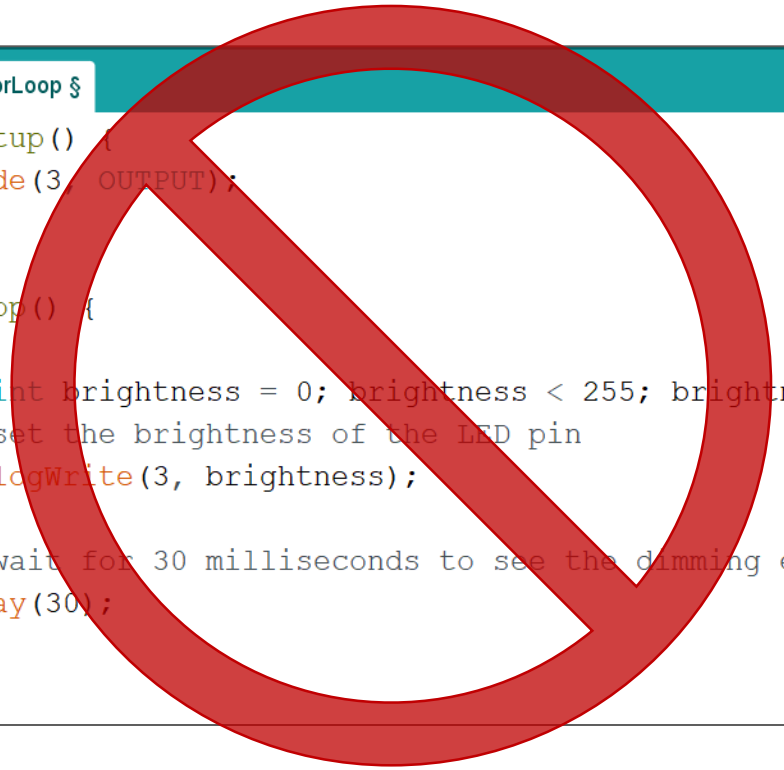
FadeOnSimple §

```
int _curBrightness = 0; // how bright the LED is (between 0 - 255)

void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  // set the brightness of the LED pin
  analogWrite(3, _curBrightness);

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + 1;

  // check to see if we are beyond the max val (255)
  // and if we are, reset the brightness to 0 (which is off)
  if (_curBrightness > 255){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```



FadeOnForLoop §

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  for (int brightness = 0; brightness < 255; brightness++){
    // set the brightness of the LED pin
    analogWrite(3, brightness);

    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```



We typically want to **avoid loops** so we don't "**starve**" other parts of our program (especially **input!**). However, in this case, it would be fine…

ANALOG OUTPUT

MY IMPLEMENTATION

FadeOnSimple §

```
int _curBrightness = 0; // how bright the LED is (between 0

void setup() {
  pinMode(3, OUTPUT);
}

void loop() {

  // set the brightness of the LED pin
  analogWrite(3, _curBrightness);

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + 1;

  // check to see if we are beyond the max val (255)
  // and if we are, reset the brightness to 0 (which is off)
  if (_curBrightness > 255){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

FadeOn §

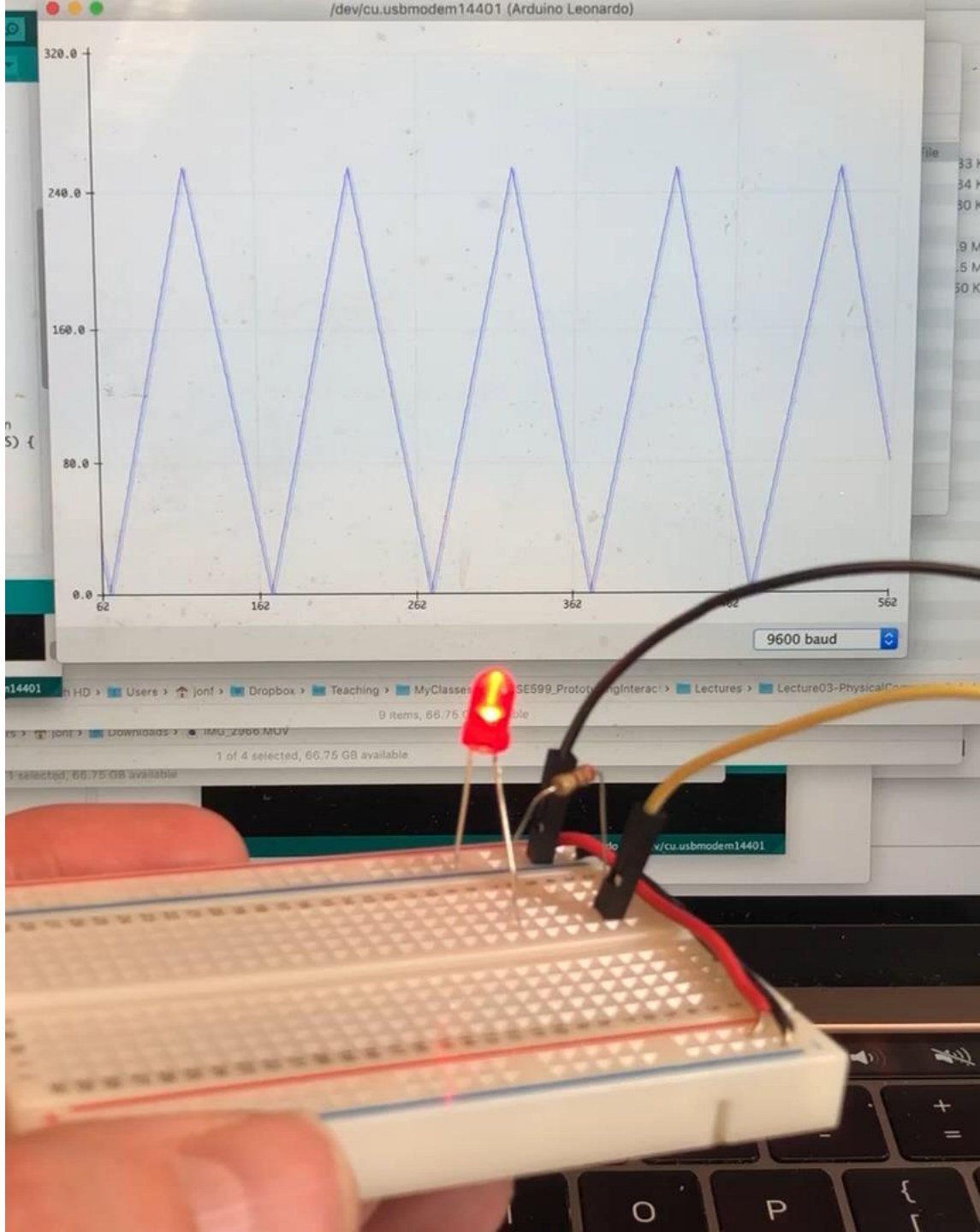
```
const int LED_OUTPUT_PIN = 3;
const int MAX_BRIGHTNESS = 255; // max digital out on Uno, etc. is 255
int _curBrightness = 0;   // how bright the LED is (between 0 - 255)
int _stepVal = 1; // how much to change brightness on each loop

// The setup function runs once when you press reset or power the board
void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  Serial.begin(9600); // for using Serial.println
}

// The loop function runs over and over again forever
void loop() {

  // set the brightness of the LED pin
  analogWrite(LED_OUTPUT_PIN, _curBrightness);
  Serial.println(_curBrightness); // print out current brightness

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + _stepVal;

  // check to see if we are beyond the max brightness val
  // and if we are, reset the brightness to 0 (which is off)
  if (_curBrightness > MAX_BRIGHTNESS){
    _curBrightness = 0;
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

ANALOG OUTPUT

ACTIVITY FADE LED ON & OFF

Now fade the LED on **and** off

What **changes** do you need to make to your code to support this behavior?

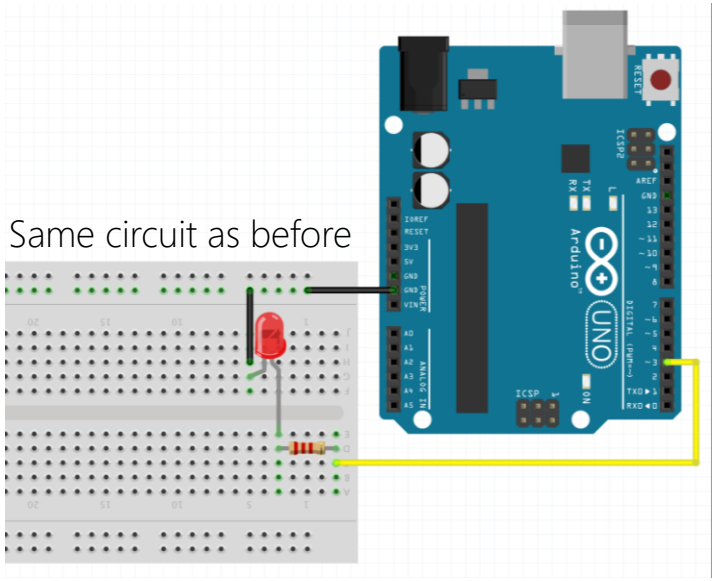


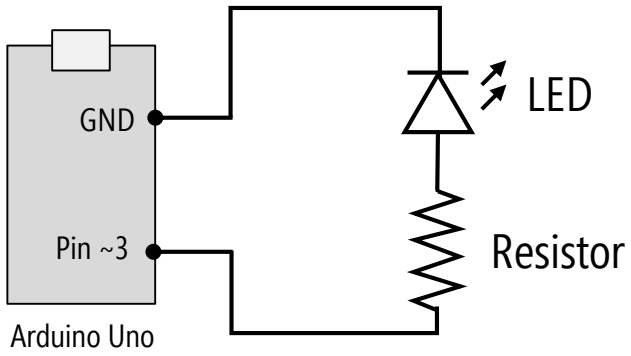
ANALOG OUTPUT

FADE LED ON AND FADE LED OFF

The same circuit as the one before but we need to update the code to fade on and off

Circuit: Fade LED on and off via the pin 3





Code: Fade LED on and off via pin 3

FadeOnAndOff

```
const int LED_OUTPUT_PIN = 3;
const int MIN_BRIGHTNESS = 0; // the min brightness
const int MAX_BRIGHTNESS = 255; // the max analog out value on Uno, Leonardo, etc. is 255 (8-bit max)

int _fadeAmount = 5;      // the amount to fade the LED by on each step
int _curBrightness = 0;   // how bright the LED is

// The setup function runs once when you press reset or power the board
void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  Serial.begin(9600); // for using Serial.println
}

// The loop function runs over and over again forever
void loop() {

  // set the brightness of the LED pin
  analogWrite(LED_OUTPUT_PIN, _curBrightness);
  Serial.println(_curBrightness); // print out current brightness

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + _fadeAmount;

  // reverse the direction of the fading at the end of each fade direction
  if (_curBrightness <= MIN_BRIGHTNESS || _curBrightness >= MAX_BRIGHTNESS) {
    _fadeAmount = -_fadeAmount; // reverses fade direction
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

FadeOnAndOff

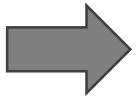
```
const int LED_OUTPUT_PIN = 3;
const int MIN_BRIGHTNESS = 0; // the min brightness
const int MAX_BRIGHTNESS = 255; // the max analog out value on Uno, Leonardo, etc. is 255 (8-bit max)

int _fadeAmount = 5;      // the amount to fade the LED by on each step
int _curBrightness = 0;   // how bright the LED is

// The setup function runs once when you press reset or power the board
void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  Serial.begin(9600); // for using Serial.println
}

// The loop function runs over and over again forever
void loop() {

  // set the brightness of the LED pin
  analogWrite(LED_OUTPUT_PIN, _curBrightness);
  Serial.println(_curBrightness); // print out current brightness

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + _fadeAmount;

  // reverse the direction of the fading at the end of each fade direction
  if (_curBrightness <= MIN_BRIGHTNESS || _curBrightness >= MAX_BRIGHTNESS) {
    _fadeAmount = -_fadeAmount; // reverses fade direction
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

ANALOG OUTPUT

AGAIN, YOU COULD USE A FOR LOOP...

FadeOnAndOff

```
const int LED_OUTPUT_PIN = 3;
const int MIN_BRIGHTNESS = 0; // the min brightness
const int MAX_BRIGHTNESS = 255; // the max analog out value on Uno, Leonardo, etc.

int _fadeAmount = 5;      // the amount to fade the LED by on each step
int _curBrightness = 0;   // how bright the LED is

// The setup function runs once when you press reset or power the board
void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  Serial.begin(9600); // for using Serial.println
}

// The loop function runs over and over again forever
void loop() {

  // set the brightness of the LED pin
  analogWrite(LED_OUTPUT_PIN, _curBrightness);
  Serial.println(_curBrightness); // print out current brightness

  // change the brightness for next time through the loop
  _curBrightness = _curBrightness + _fadeAmount;

  // reverse the direction of the fading at the end of each fade direction
  if (_curBrightness <= MIN_BRIGHTNESS || _curBrightness >= MAX_BRIGHTNESS) {
    _fadeAmount = -_fadeAmount; // reverses fade direction
  }

  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```



FadeOnAndOffForLoop

```
const int LED_OUTPUT_PIN = 3;

void setup() {
  // set the LED pin to as an output
  pinMode(LED_OUTPUT_PIN, OUTPUT);
}

void loop(){
  // fade on
  for(int i = 0; i <= 255; i += 1){
    analogWrite(LED_OUTPUT_PIN, i);
    delay(30);
  }

  //fade off
  for(int i = 255; i >= 0; i -= 1){
    analogWrite(LED_OUTPUT_PIN, i);
    delay(30);
  }
}
```

HOW DOES ANALOG WRITE WORK?

How does `analogWrite` work?

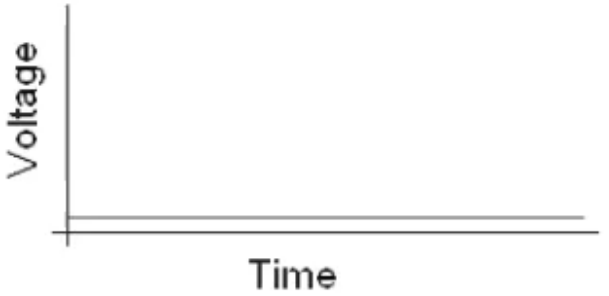
That is, how can the microcontroller **dynamically control the voltage** output?

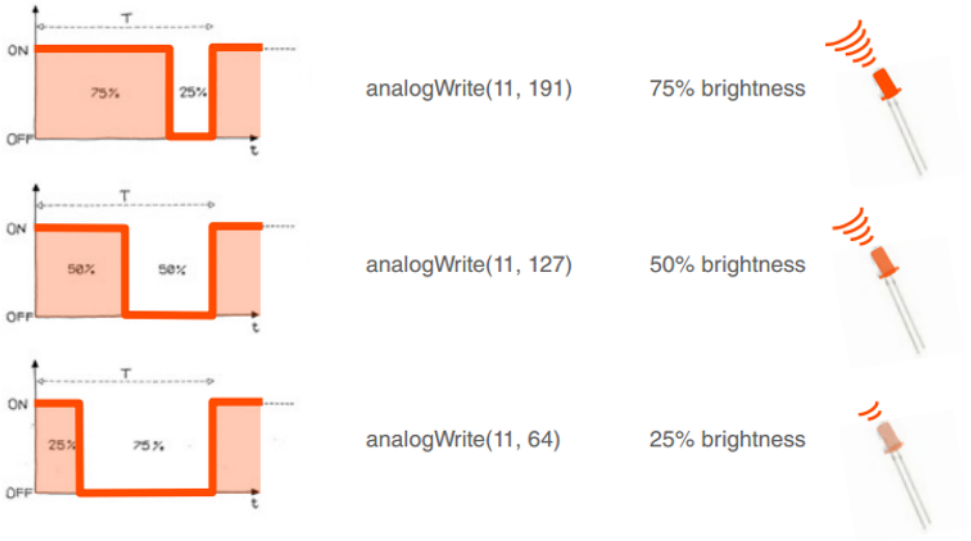
HOW DOES ANALOG WRITE WORK?

How does `analogWrite` work?

That is, how can the microcontroller **dynamically control the voltage** output?

Using **PWM!**





Slide adapted from Kimiko Ryokai's Theory and Practice of Tangible User Interfaces course; Animted gif from: https://i.gifer.com/I0Yv.gif

How do we know? To the **documentation**…

LANGUAGE
FUNCTIONS
VARIABLES
STRUCTURE
LIBRARIES
GLOSSARY

Reference > Language > Functions > Analog io > Analogwrite

analogWrite()

[Analog I/O]

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

* In addition to PWM capabilities on the pins noted above, the MKR and Zero boards have true analog output when using `analogWrite()` on the `DAC0` (`A0`) pin.
** In addition to PWM capabilities on the pins noted above, the Due has true analog output when using `analogWrite()` on pins `DAC0` and `DAC1`.

You do not need to call `pinMode()` to set the pin as an output before calling `analogWrite()`. The `analogWrite` function has nothing to do with the analog pins or the `analogRead` function.

Syntax

```
analogWrite(pin, value)
```

Parameters

`pin`: the Arduino pin to write to. Allowed data types: `int`.
`value`: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: `int`.

https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

analogWrite()

[Analog I/O]

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz

LANGUAGE
FUNCTIONS
VARIABLES
STRUCTURE
LIBRARIES
GLOSSARY

Find anything that can be improved? Suggest corrections and new documentation via GitHub.

Doubts on how to use Github? Learn everything you need to know in this tutorial.

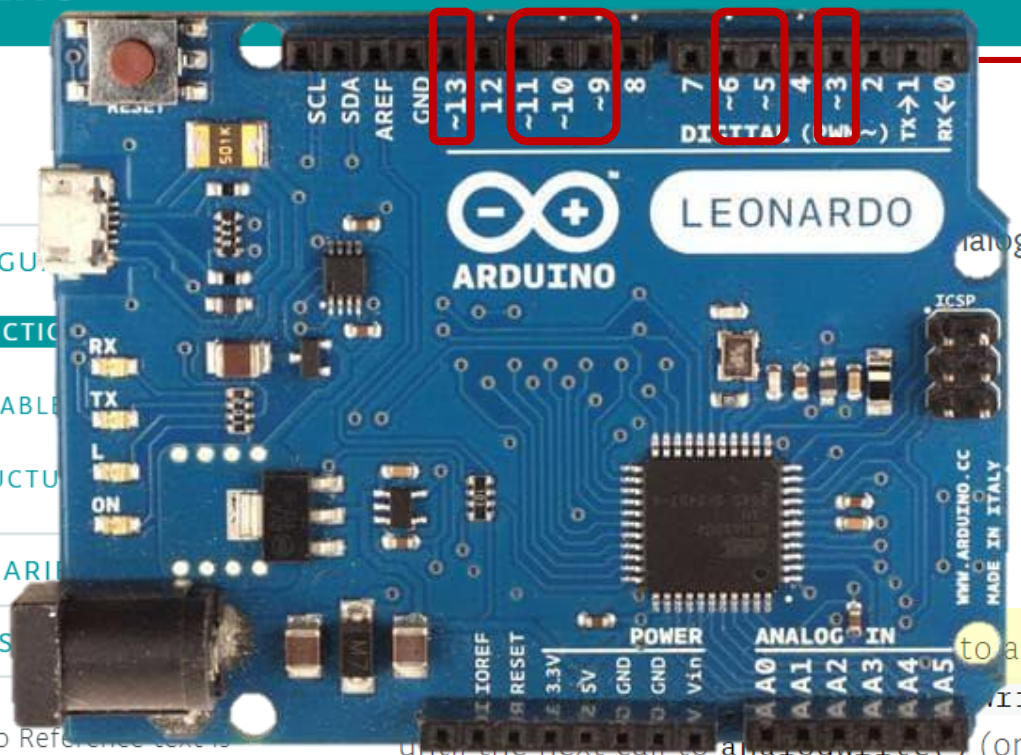
analogWrite()

[Analog I/O]

Description

Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz



to a pin. Can be used to light a LED at varying brightnesses or drive a motor at
`rite()`, the pin will generate a steady rectangular wave of the specified duty cycle
until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

BOARD	PWM PINS	PWM FREQUENCY
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz

know in this tutorial.

Leonardo, Micro, Yun	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

* In addition to PWM capabilities on the pins noted above, the MKR and Zero boards have true analog output when using `analogWrite()` on the `DAC0` (`A0`) pin.
** In addition to PWM capabilities on the pins noted above, the Due has true analog output when using `analogWrite()` on pins `DAC0` and `DAC1`.

You do not need to call `pinMode()` to set the pin as an output before calling `analogWrite()`. The `analogWrite` function has nothing to do with the analog pins or the `analogRead` function.

Last Update: 7/2/2019

EDIT THIS PAGE

Syntax

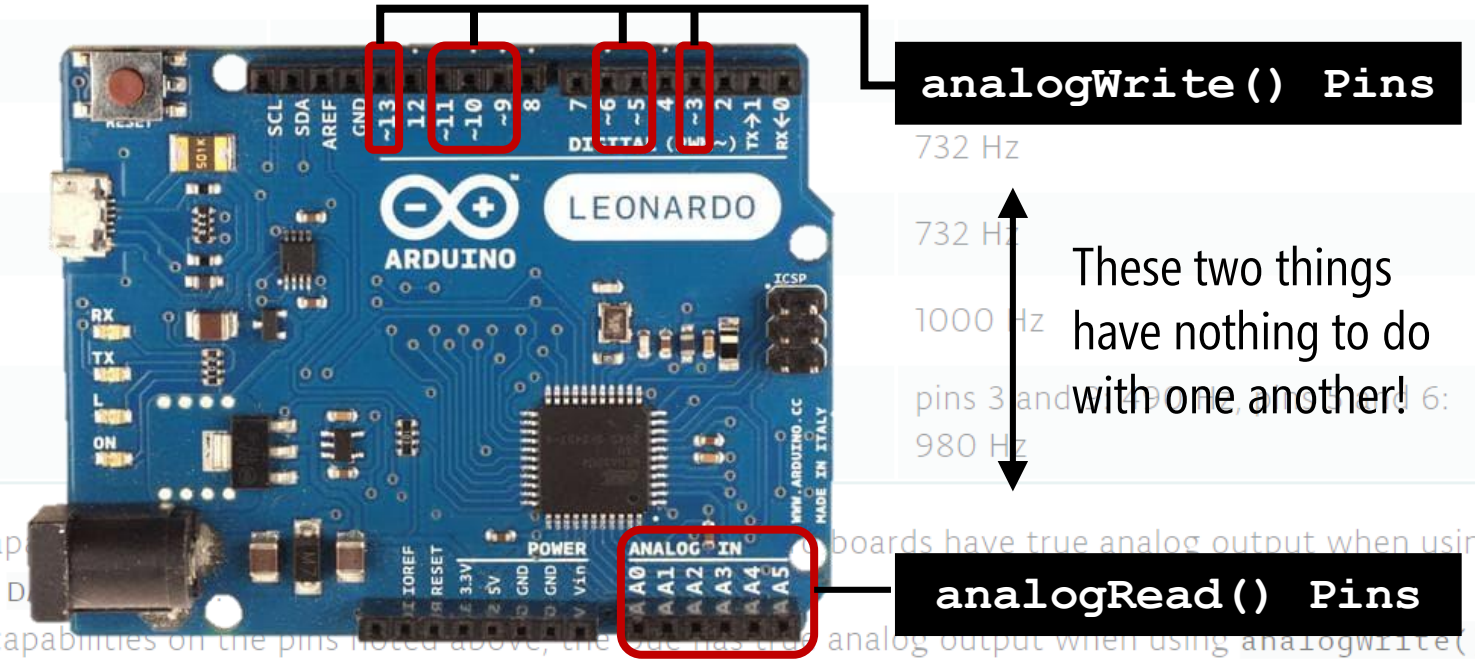
```
analogWrite(pin, value)
```

Parameters

`pin`: the Arduino pin to write to. Allowed data types: `int`.
`value`: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: `int`.

know in this tutorial.

Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz
MKR boards *		
MKR1000 WiFi *		732 Hz
Zero *		732 Hz
Due **		1000 Hz
101		pins 3 and [illegible], 5, 6: 980 Hz



* In addition to PWM cap[illegible] boards have true analog output when using `analogWrite()` on the D[illegible]
** In addition to PWM capabilities on the pins noted above, the Due has true analog output when using `analogWrite()` on pins `DAC0` and `DAC1`.

Last Update: 7/2/2019

EDIT THIS PAGE

You do not need to call `pinMode()` to set the pin as an output before calling `analogWrite()`. The `analogWrite` function has nothing to do with the analog pins or the `analogRead` function.

Syntax

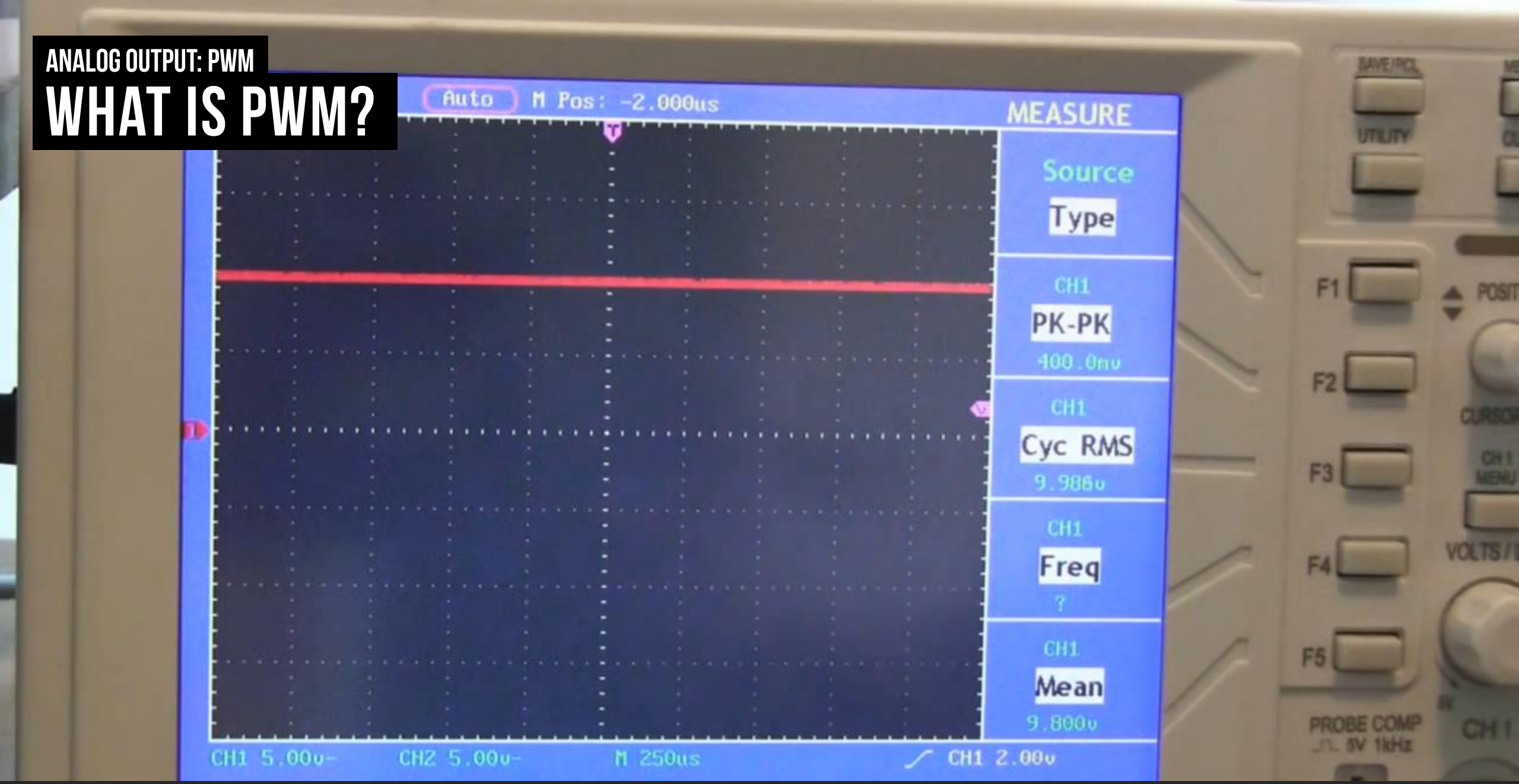
```
analogWrite(pin, value)
```

Parameters

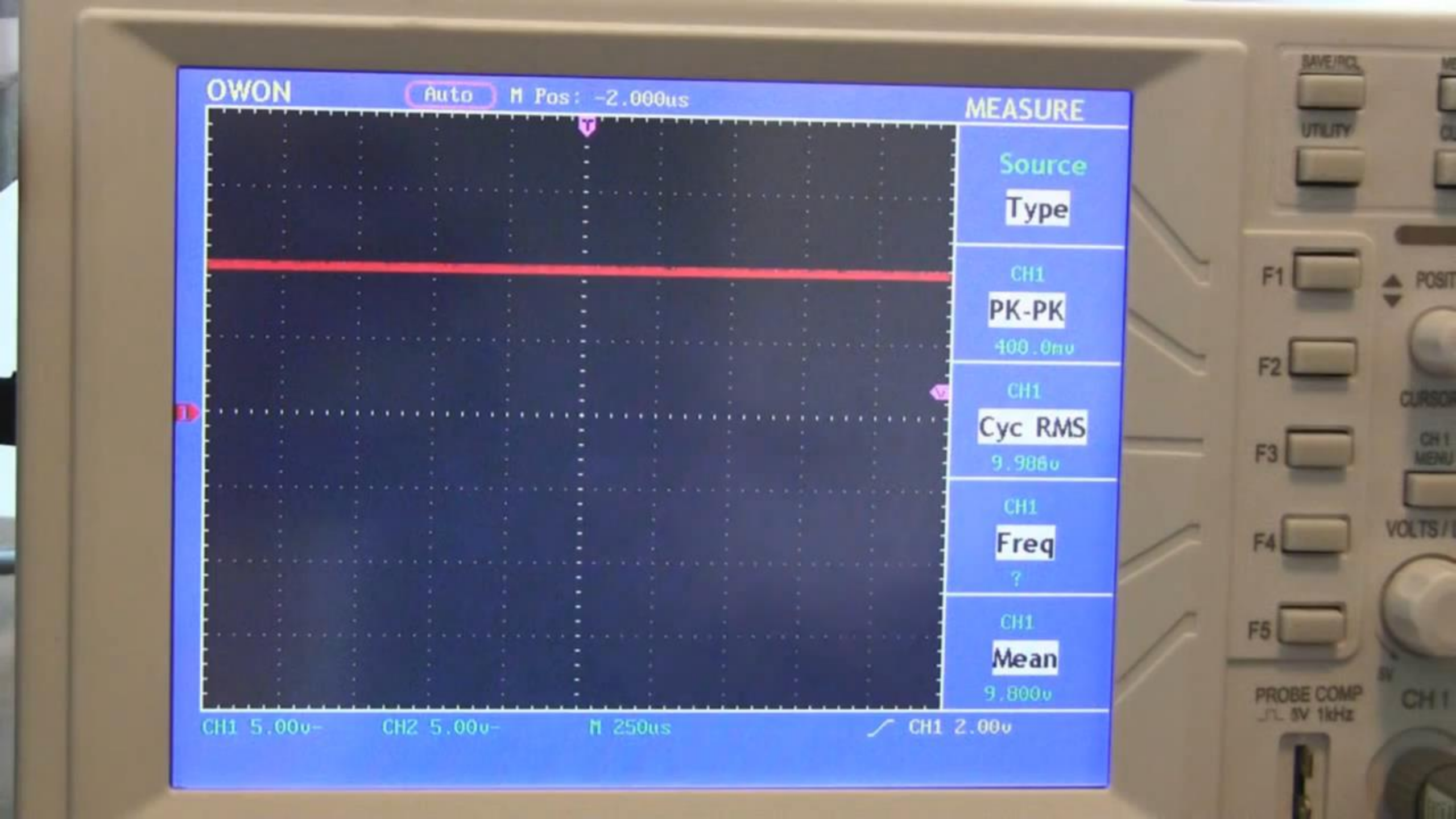
`pin`: the Arduino pin to write to. Allowed data types: `int`.
`value`: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: `int`.

ANALOG OUTPUT: PWM

WHAT IS PWM?



Source: Afrotechmods, https://youtu.be/YmPziPfaByw


OWON
Auto
M Pos: -2.000us
MEASURE
Source
Type
CH1
PK-PK
400.0mv
CH1
Cyc RMS
9.986v
CH1
Freq
?
CH1
Mean
9.800v
CH1 5.00v-
CH2 5.00v-
M 250us
CH1 2.00v
SAVE/RCL
UTILITY
F1
F2
F3
F4
F5
PROBE COMP
5V 1kHz
CH 1

ANALOG OUTPUT: PWM

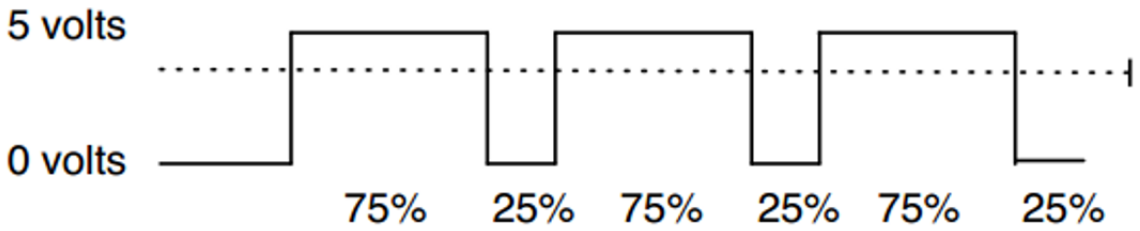
PULSE WIDTH MODULATION

The voltage is max_voltage * duty_cycle_val

$$V_{out} = Voltage_{max} * Duty\ Cycle_{fraction}$$

What is the V_{out}?

75% Duty Cycle
analogWrite(191)



Source: https://www.arduino.cc/en/Tutorial/PWM

ANALOG OUTPUT: PWM

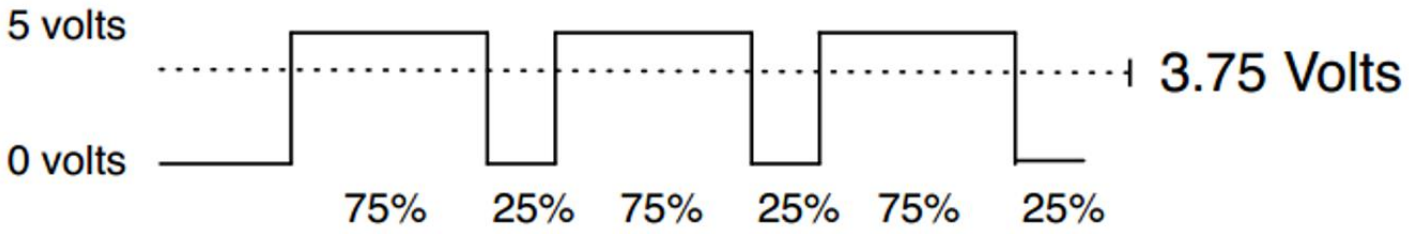
PULSE WIDTH MODULATION

The voltage is max_voltage * duty_cycle_val

$$V_{out} = Voltage_{max} * Duty\ Cycle_{fraction}$$

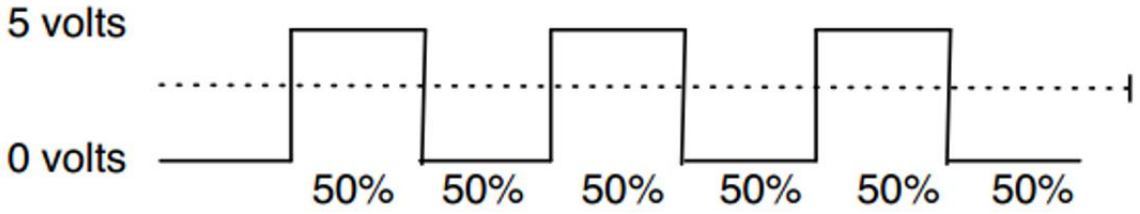
What is the V_{out}?

75% Duty Cycle
analogWrite(191)



3.75 Volts

50% Duty Cycle
analogWrite(127)



Source: https://www.arduino.cc/en/Tutorial/PWM

ANALOG OUTPUT: PWM

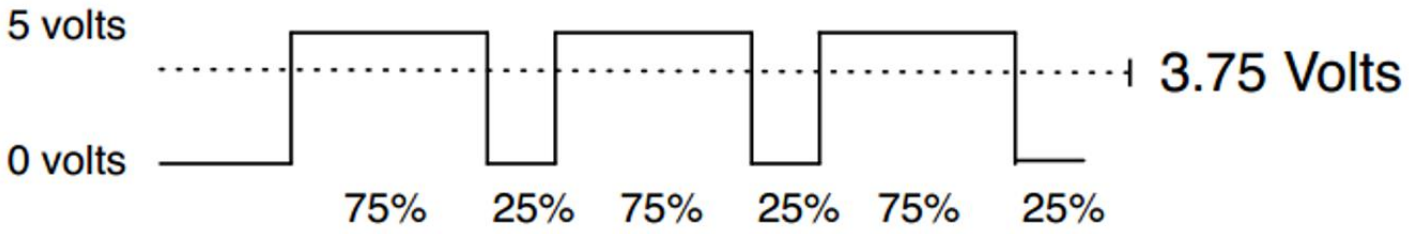
PULSE WIDTH MODULATION

The voltage is max_voltage * duty_cycle_val

$$V_{out} = Voltage_{max} * Duty\ Cycle_{fraction}$$

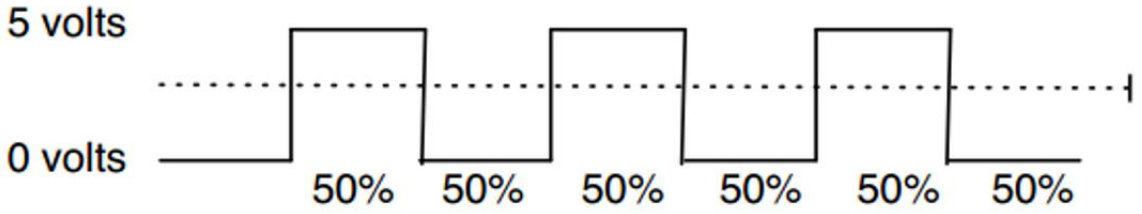
What is the V_{out}?

75% Duty Cycle
analogWrite(191)



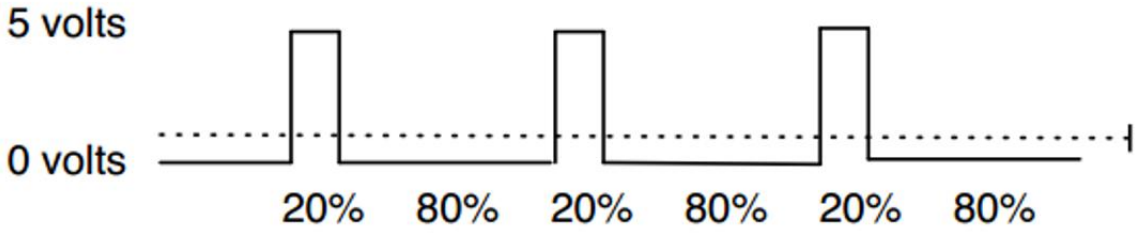
3.75 Volts

50% Duty Cycle
analogWrite(127)



2.5 Volts

20% Duty Cycle
analogWrite(51)



Source: https://www.arduino.cc/en/Tutorial/PWM

ANALOG OUTPUT: PWM

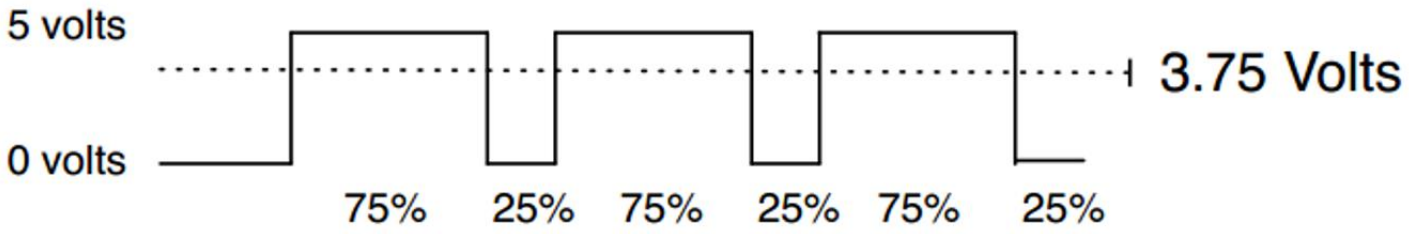
PULSE WIDTH MODULATION

The voltage is max_voltage * duty_cycle_val

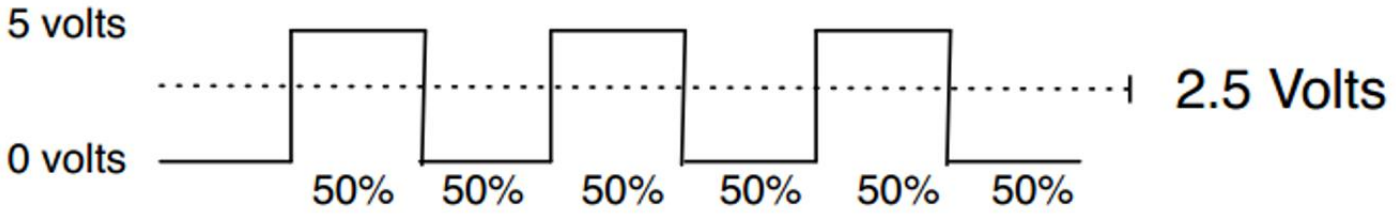
$$V_{out} = Voltage_{max} * Duty\ Cycle_{fraction}$$

What is the V_{out}?

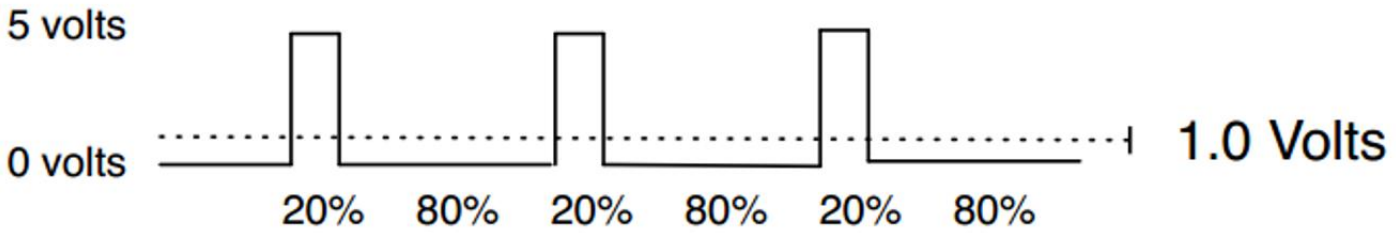
75% Duty Cycle
analogWrite(191)



50% Duty Cycle
analogWrite(127)

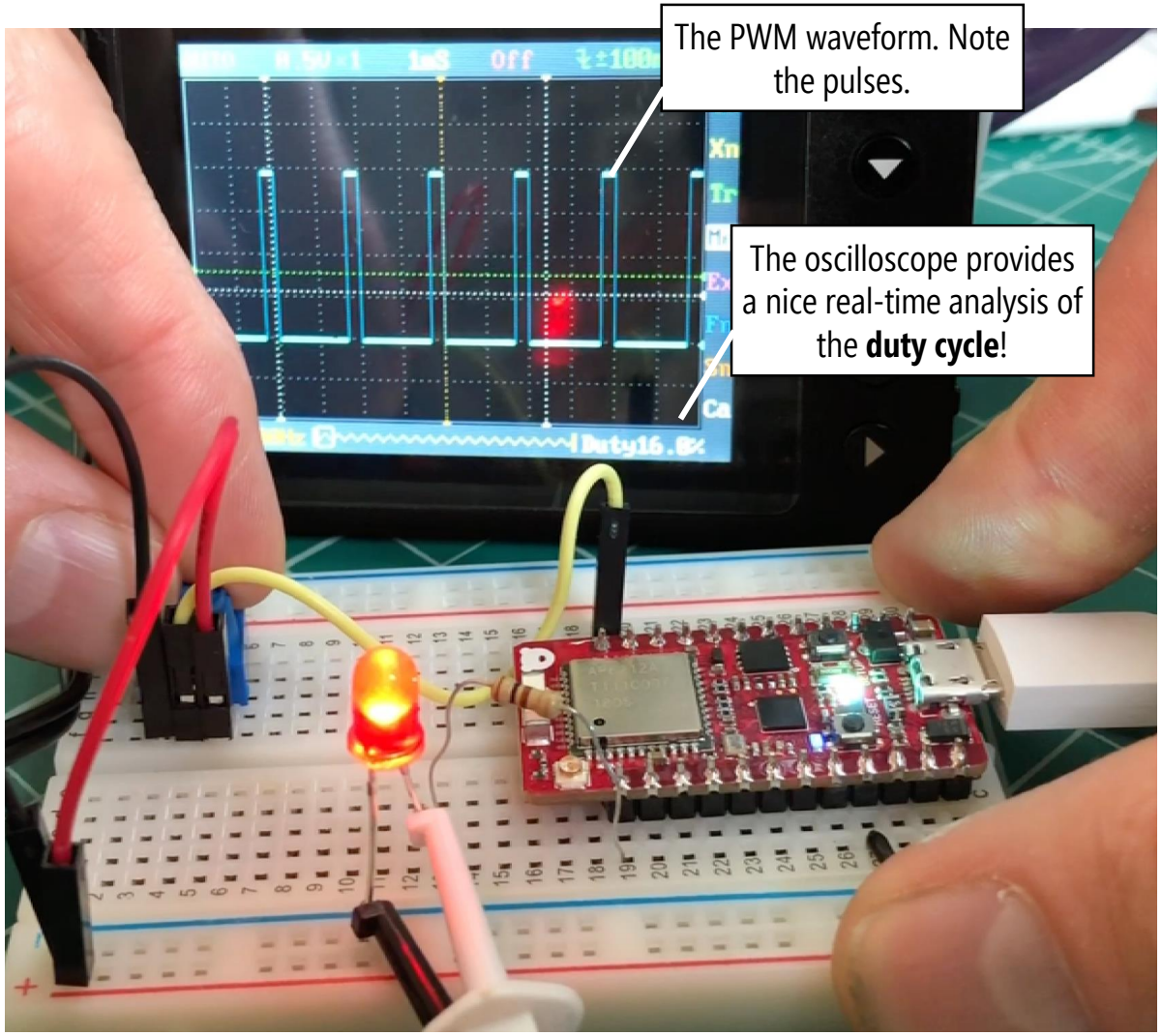


20% Duty Cycle
analogWrite(51)



Source: https://www.arduino.cc/en/Tutorial/PWM

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE

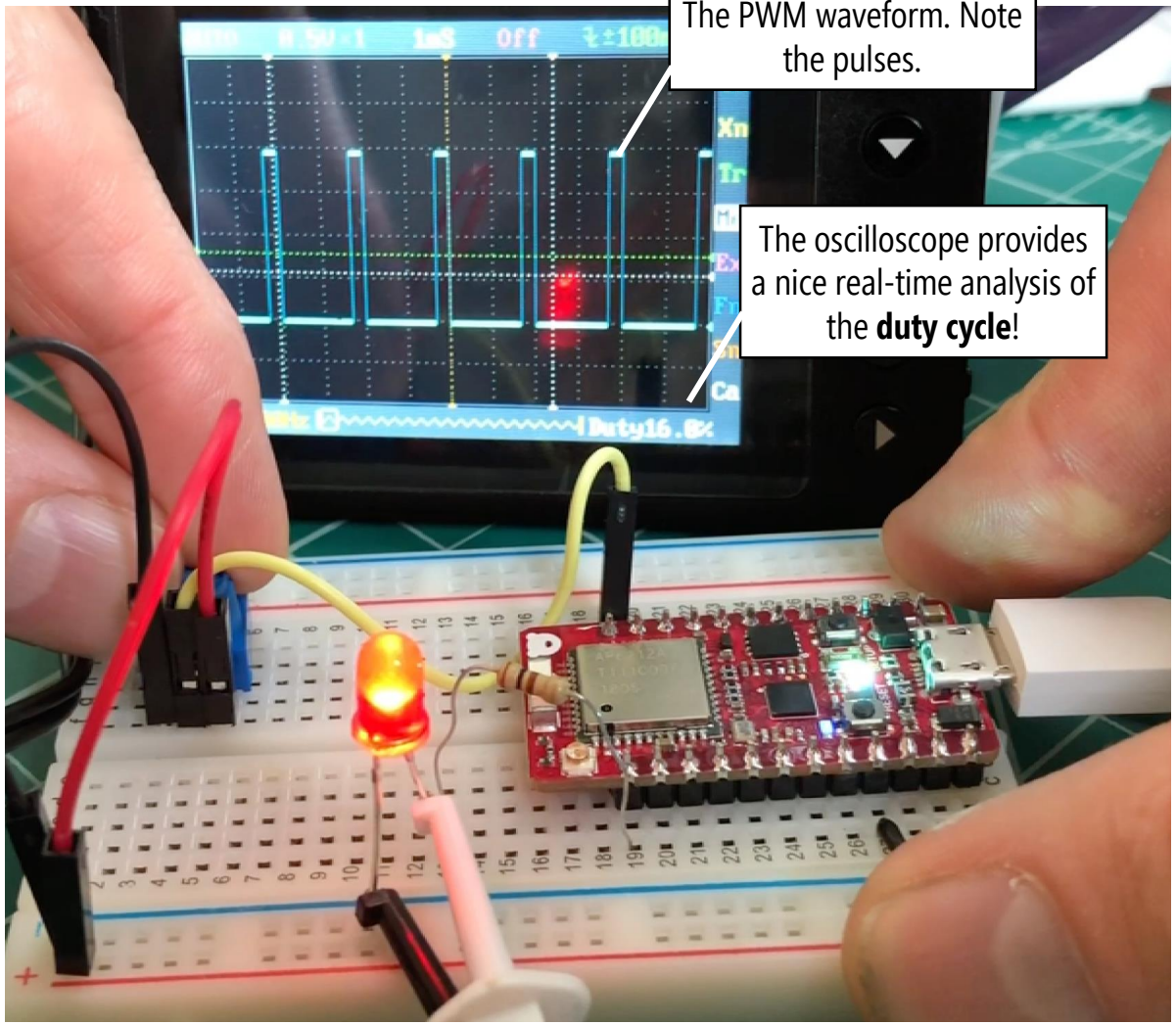


RedBearDuoReadPotSetLED | Arduino 1.8.5

RedBearDuoReadPotSetLED

```
SYSTEM_MODE(MANUAL);

const int LED_OUTPUT_PIN = D0;
const int POT_INPUT_PIN = A0;

void setup() {
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  pinMode(POT_INPUT_PIN, INPUT);
}

void loop() {

  // read the potentiometer value
  int potVal = analogRead(POT_INPUT_PIN);

  // the analogRead on the RedBear Duo seems to go from 0 to 4092 (and not 4095
  // as you would expect with a power of two--e.g., 2^12 or 12 bits). On the Arduino
  // Uno, the analogRead ranges from 0 to 1023 (2^10 or 10 bits). Regardless,
  // we need to remap this value linearly from the large range (0-4092) to
  // the smaller range (0-255) since the analogWrite function can only write out
  // 0-255 (a byte--2^8).
  int ledVal = map(potVal, 0, 4092, 0, 255);

  // write out the LED value. This value is translated to voltage by:
  // voltageVal = max_voltage * val/255 or voltageVal = 3.3V * val/255 in
  // the case of the RedBear Duo
  analogWrite(LED_OUTPUT_PIN, ledVal);

  delay(100);
}
```

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE



RedBearDuoReadPotSetLED | Arduino 1.8.5

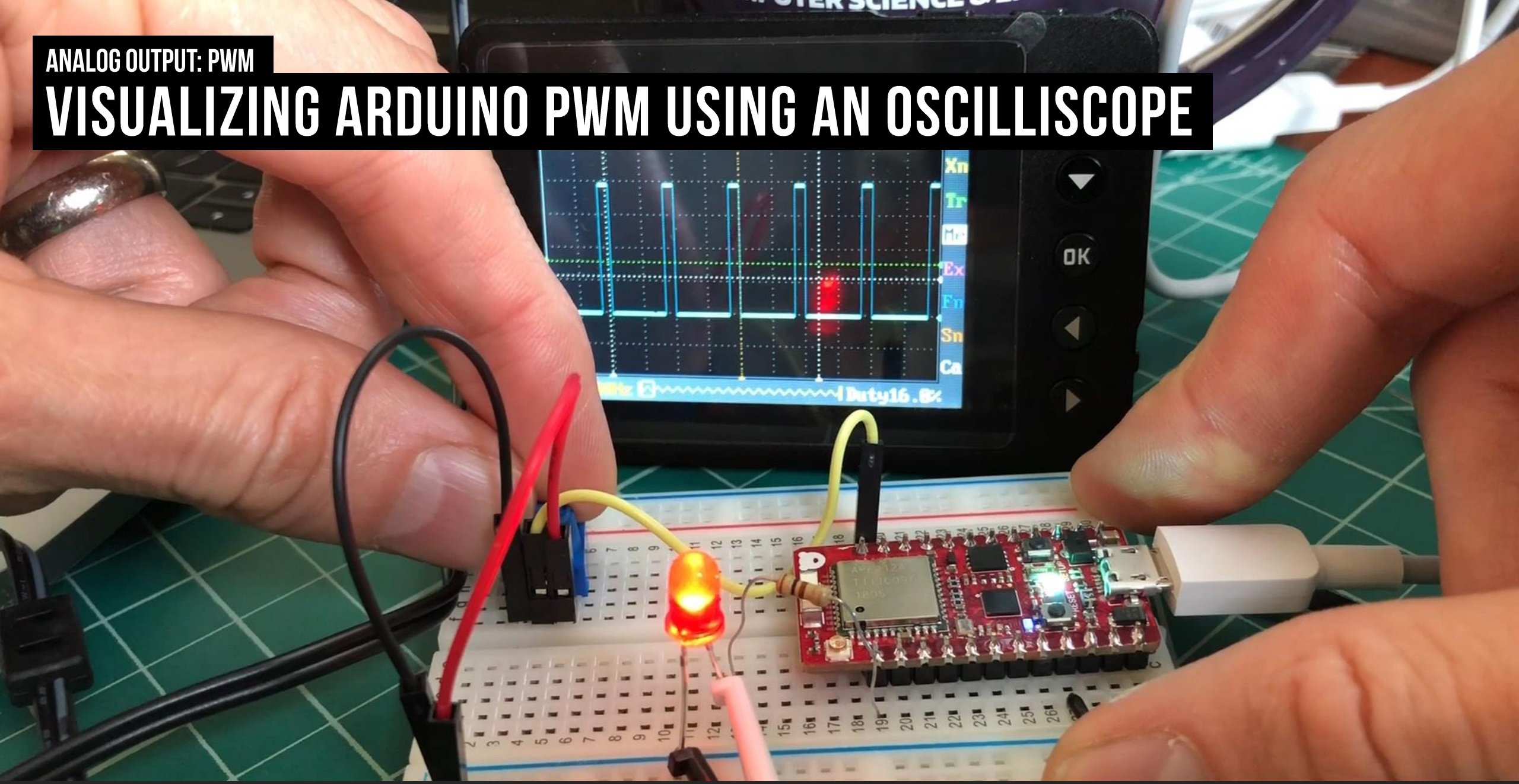
RedBearDuoReadPotSetLED

```
SYSTEM_MODE(MANUAL);

const int LED_OUTPUT_PIN = D0;
const int POT_INPUT_PIN = A0;

void setup() {
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  pinMode(POT_INPUT_PIN, INPUT);
}

void loop() {

  // read the potentiometer value
  int potVal = analogRead(POT_INPUT_PIN);

  // the analogRead on the RedBear Duo seems to go from 0 to 4092 (and not 4095
  // as you would expect with a power of two--e.g., 2^12 or 12 bits). On the Arduino
  // Uno, the analogRead ranges from 0 to 1023 (2^10 or 10 bits). Regardless,
  // we need t                                                    092) to
  // the small                                                    ly write out
  // 0-255 (a
  int ledVal =

  // write ou                                                     :
  // voltageVal = max_voltage * val/255 or voltageVal = 3.3V * val/255 in
  // the case of the RedBear Duo
  analogWrite(LED_OUTPUT_PIN, ledVal);

  delay(100);
}
```

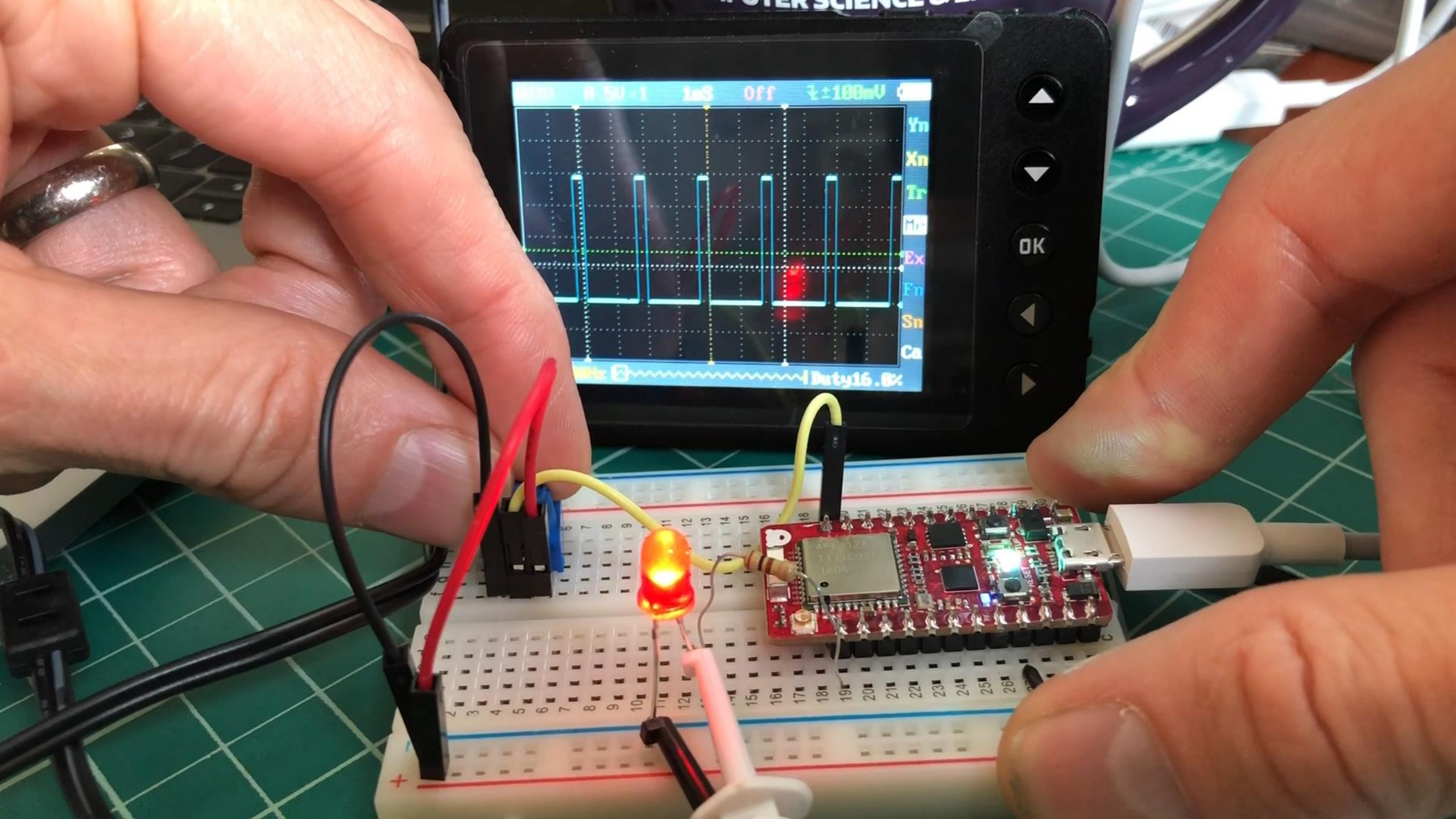
The **analogWrite call sets the duty cycle** of the PWM waveform. Note: you cannot set the PWM frequency. See the video on the next slide

ANALOG OUTPUT: PWM

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE



Source: video by Jon Froehlich


OK

PULSE WIDTH MODULATION

ANOTHER GREAT VIDEO ON PWM, DUTY CYCLES, & FREQUENCY

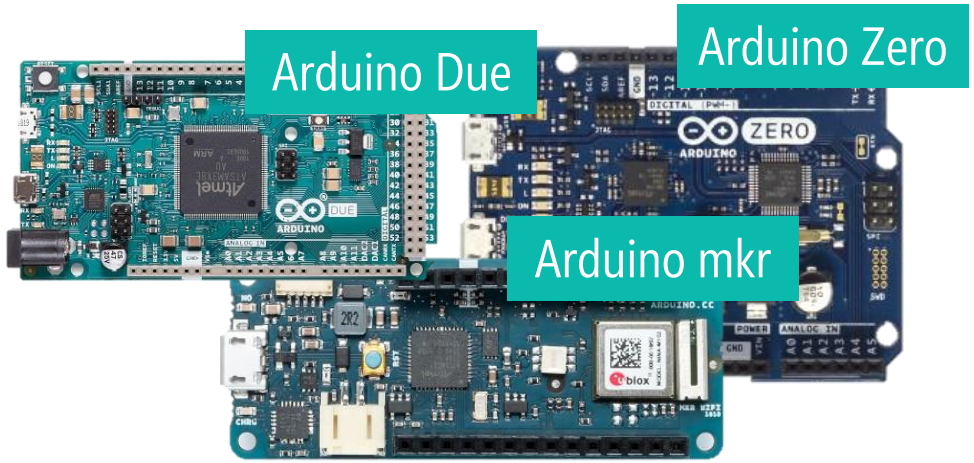
Source: https://youtu.be/rBQVfCUuhfs

But what if we want to write out **true analog voltages**?

But what if we want to write out **true analog voltages**?

We need a DAC (Digital-Analog-Converter)

TRUE ANALOG OUTPUT: 3 PRIMARY OPTIONS



Switch to a different board

Some boards have built in DACs that output true analog voltages such as the Arduino Due

LANGUAGE
FUNCTIONS
VARIABLES
STRUCTURE
LIBRARIES
GLOSSARY

analogWriteResolution()

[Zero, Due & MKR Family]

Description

`analogWriteResolution()` is an extension of the Analog API for the Arduino Due.

`analogWriteResolution()` sets the resolution of the `analogWrite()` function. It defaults to 8 bits (values between 0-255) for backward compatibility with AVR based boards.

The **Due** has the following hardware capabilities:

- 12 pins which default to 8-bit PWM, like the AVR-based boards. These can be changed to 12-bit resolution.
- 2 pins with 12-bit DAC (Digital-to-Analog Converter)

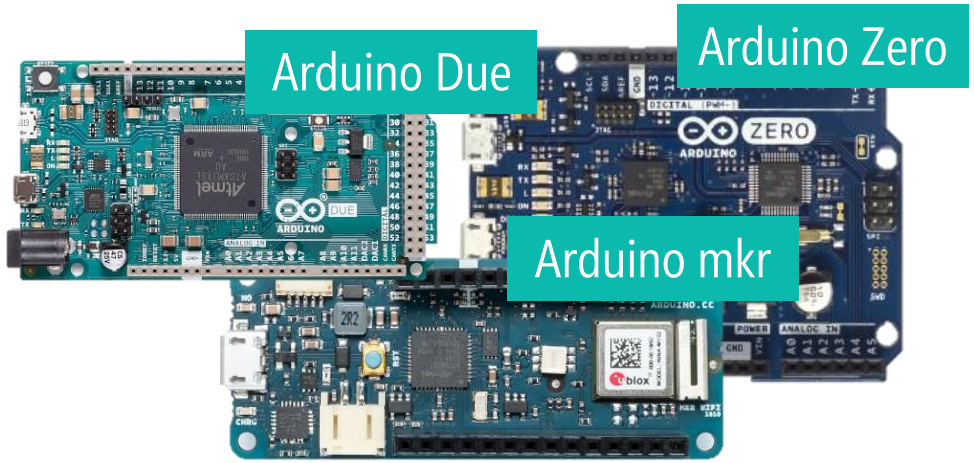
By setting the write resolution to 12, you can use `analogWrite()` with values between 0 and 4095 to exploit the full DAC resolution or to set the PWM signal without rolling over.

The **Zero** has the following hardware capabilities:

- 10 pins which default to 8-bit PWM, like the AVR-based boards. These can be changed to 12-bit resolution.
- 1 pin with 10-bit DAC (Digital-to-Analog Converter).

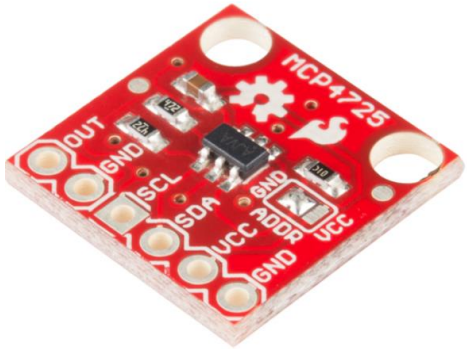
By setting the write resolution to 10, you can use `analogWrite()` with values between 0 and 1023 to exploit the full DAC resolution

TRUE ANALOG OUTPUT: 3 PRIMARY OPTIONS



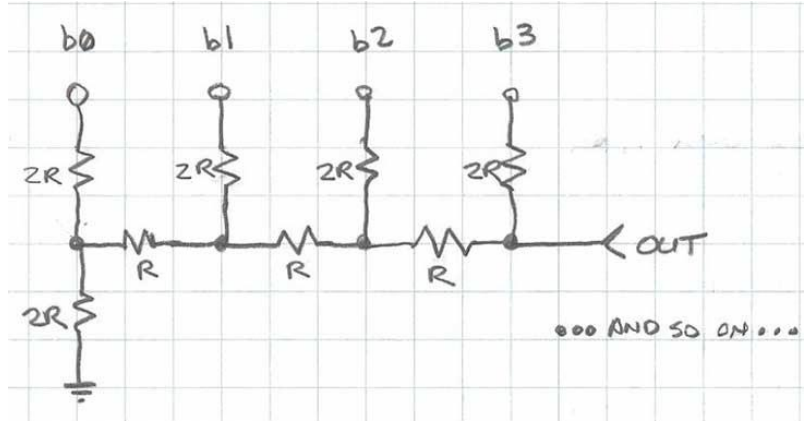
Switch to a different board

Some boards have built in DACs that output true analog voltages such as the Arduino Due



Use external DAC chip

Use an external DAC chip like the MCP4725. Both Sparkfun and Adafruit sell their own versions



Build a resistor ladder

Cheapest and most inaccurate solution is to use a ladder of resistors to convert to analog voltages.

ANALOG OUTPUT

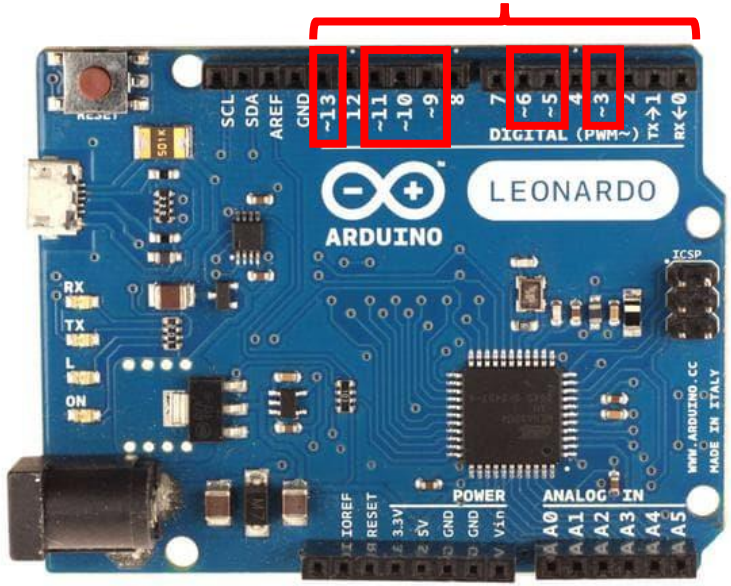
SUMMARY

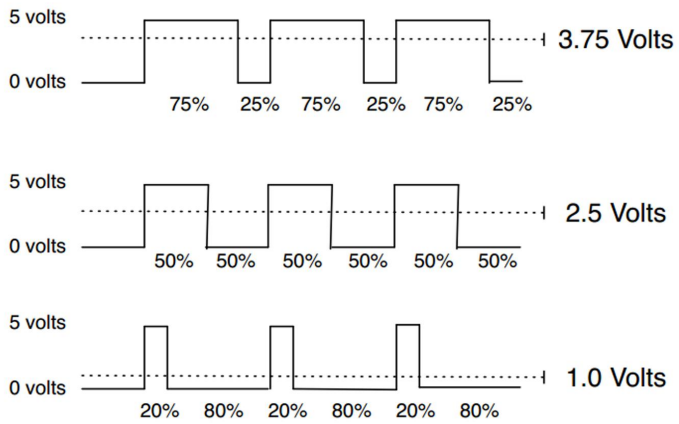
The **microcontroller** uses **PWM** to **vary** the **output voltage**

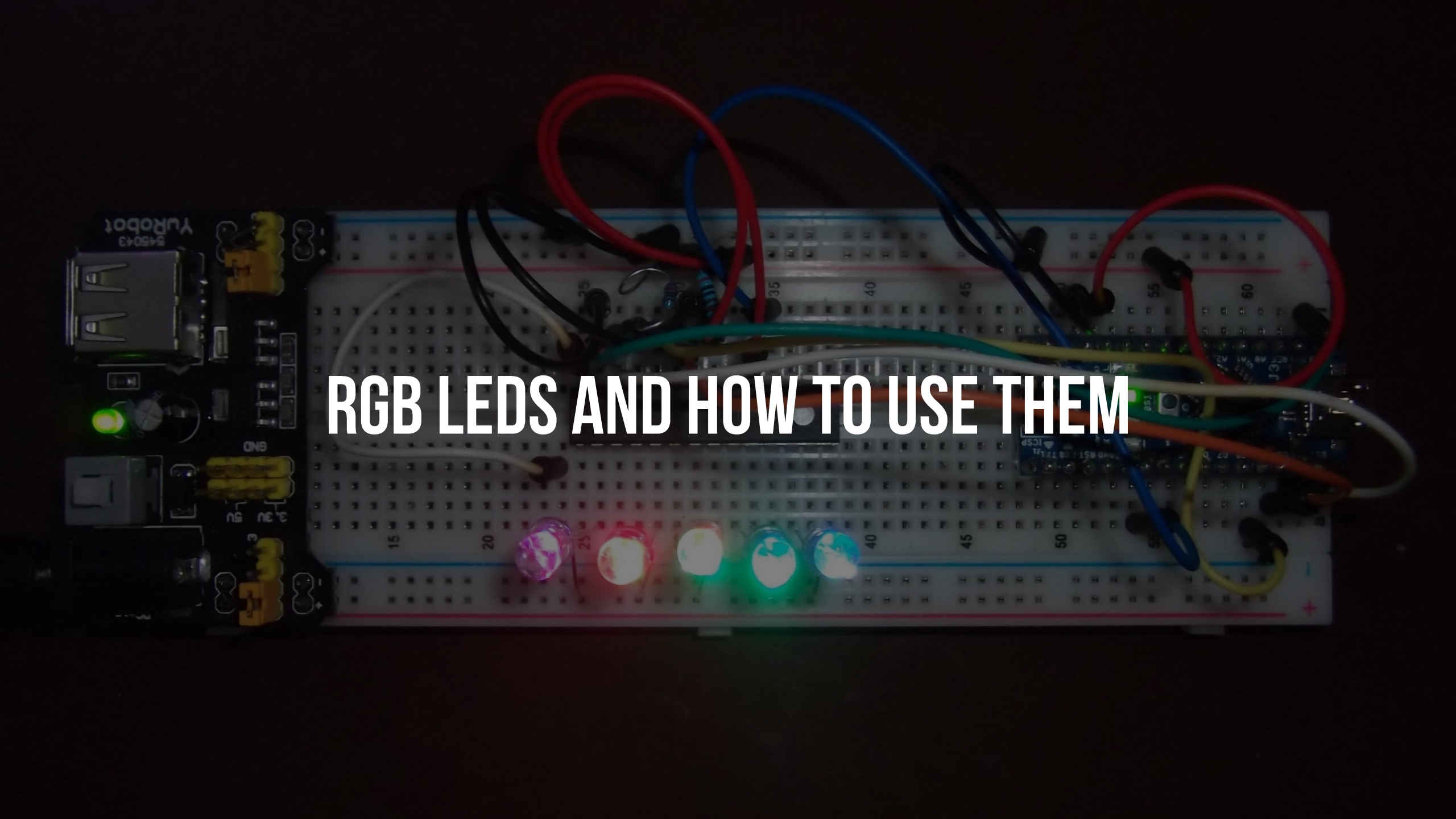
analogWrite changes the **duty cycle** but the waveform frequency is fixed at 490Hz or 980Hz depending on the pin

The analog **output voltage** is equal to **max_voltage * duty_cycle_val**

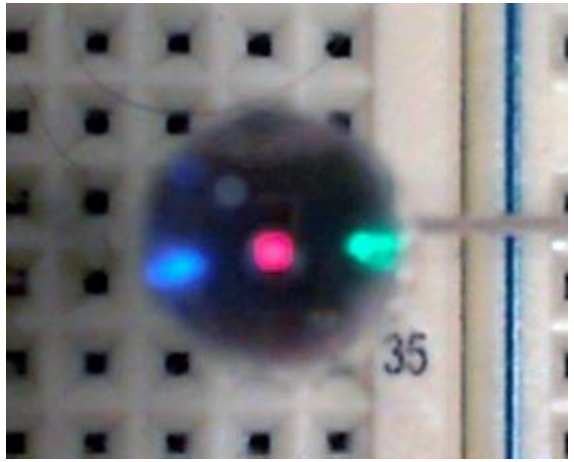
7 analog output pins: pins with the '~' symbol can be used to simulate analog output using 8-bit pulse-width modulation (PWM). Pins 3, 9, 10, 11, & 13 have PWM frequency of ~490Hz, Pins 5 & 6 are ~980Hz. Max current is 40mA per pin

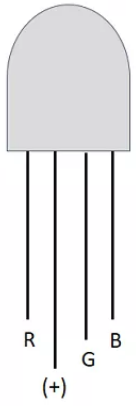


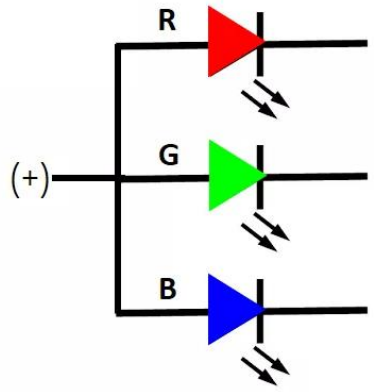



RGB LEDS AND HOW TO USE THEM

WHAT ARE RGB LEDS







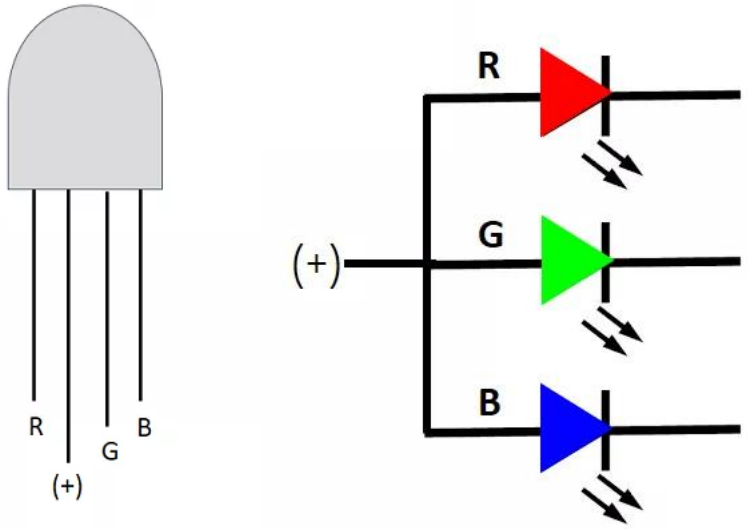
RGB LEDs are actually **three LEDs** in one**:** a **red** LED, a **green** LED, and a **blue** LED

You can **control the color** by setting the input voltages of each R, G, and B leg to a different value.

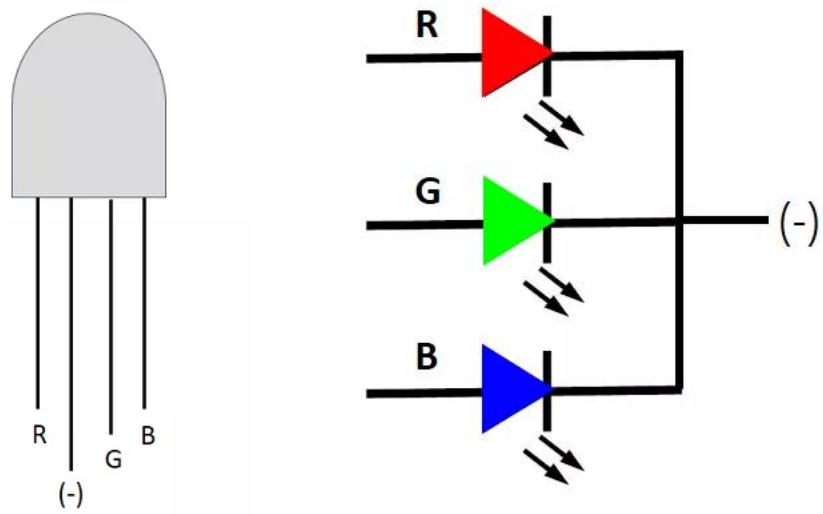
RGB LEDS

COMMON ANODE VS. COMMON CATHODE

Common Anode (+)



Common Cathode (-)



Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

1 ADD TO CART

QTY	DISCOUNT
1-9	$2.00
10-49	$1.75
50+	$1.50

ADD TO WISHLIST

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

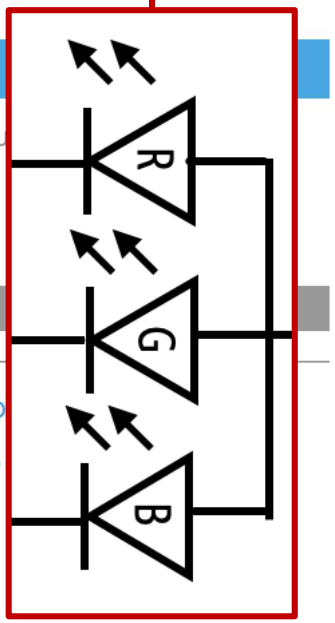
1

QTY	DISCOU
1-9	$2.00
10-49	$1.75
50+	$1.50

DESCRIPTIO

TECHNICAL

LEARN



Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

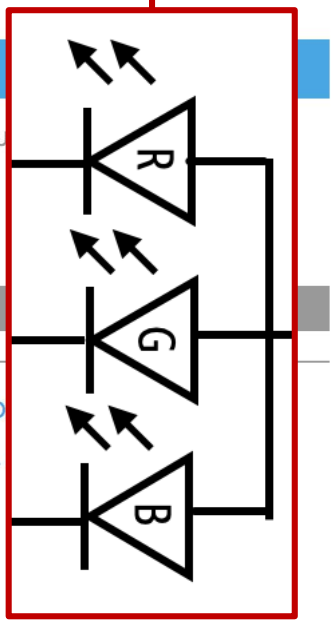
1

QTY	DISCOU
1-9	$2.00
10-49	$1.75
50+	$1.50

DESCRIPTIO

TECHNICAL

LEARN



adafruit SHOP BLOG LEARN FORUMS VIDEOS ADABOX

LEDS / BARE LEDS / DIFFUSED RGB (TRI-COLOR) LED

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

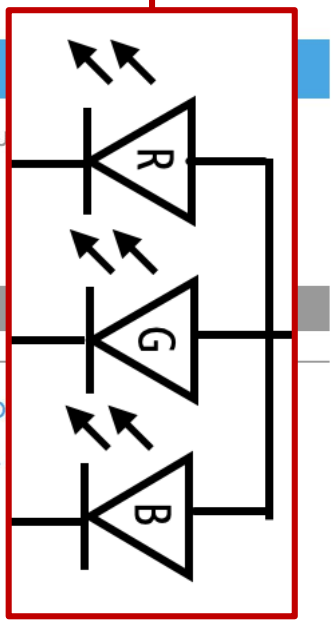
$2.00

49 IN STOCK

1

QTY DISCOU
1-9 $2.00
10-49 $1.75
50+ $1.50

DESCRIPTIO
TECHNICAL
LEARN

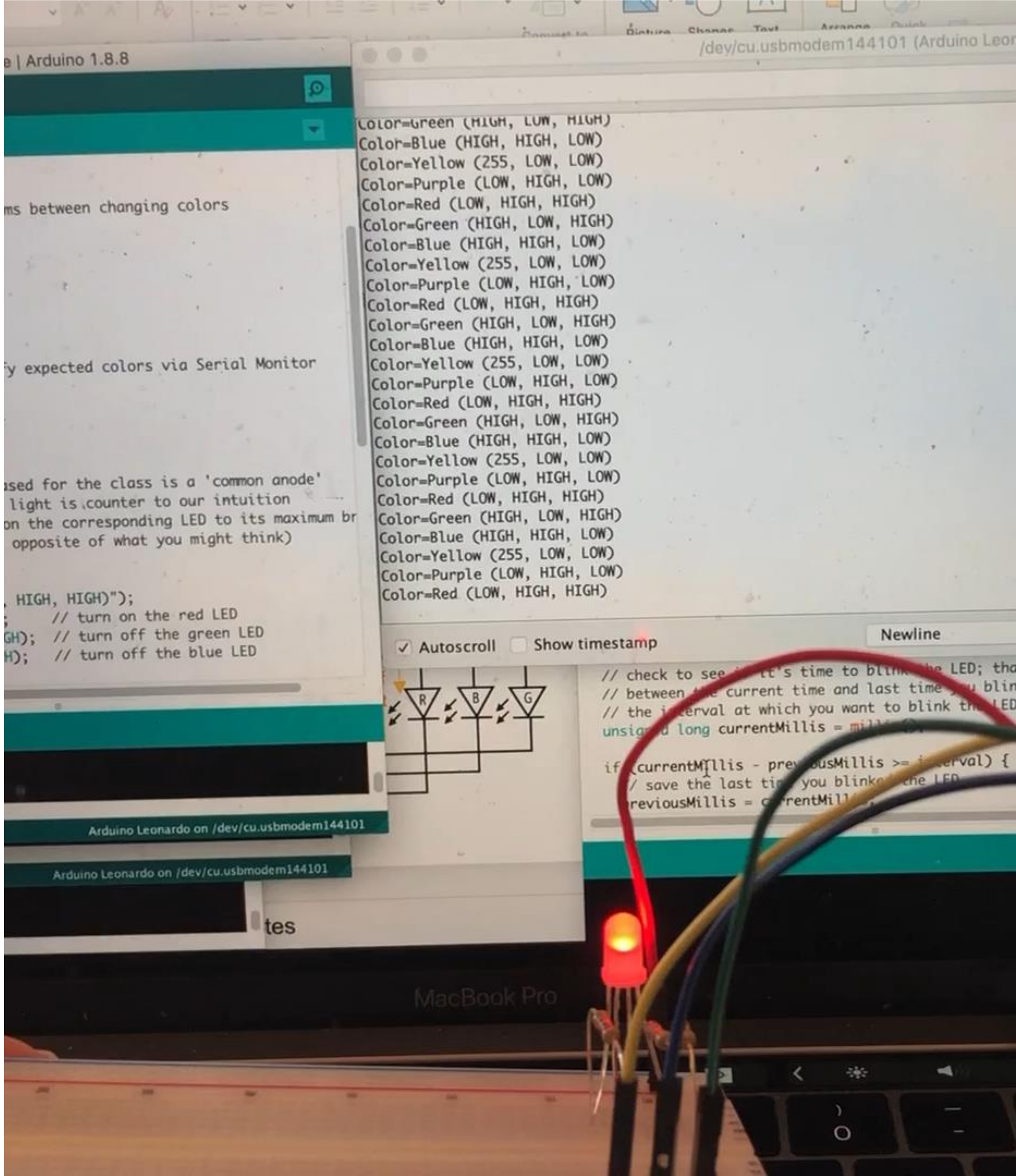


ACTIVITY

FLASH RGB LED COLORS

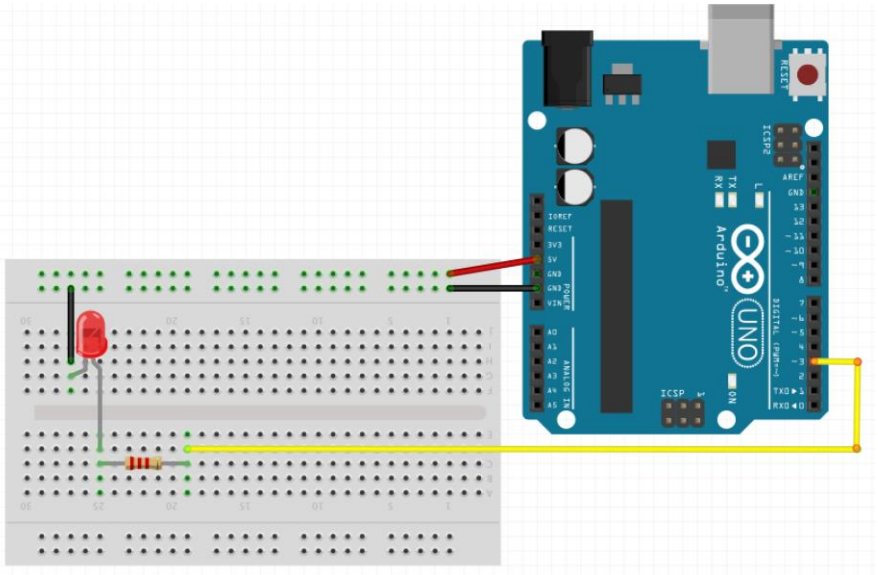
Build a circuit & write code to flash through **different RGB LED colors**

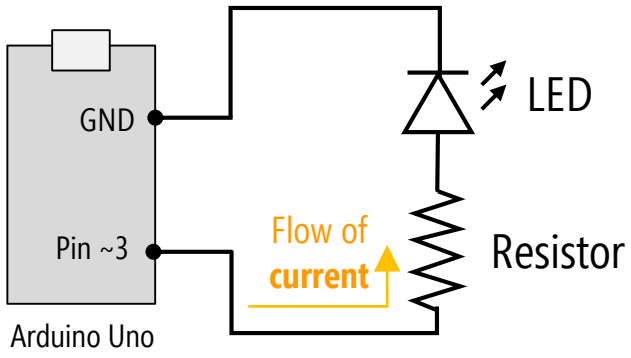
Choose whatever colors you want ☺



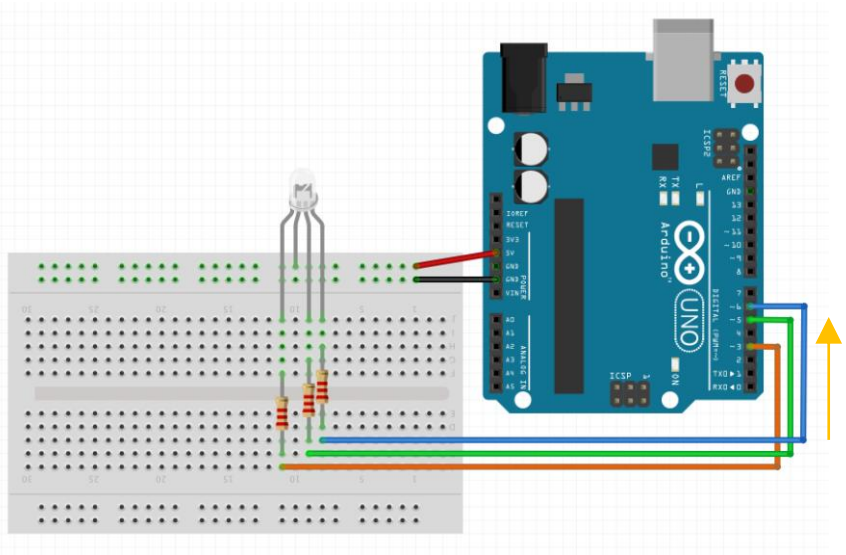
USING AN RGB LED: THE CIRCUIT

Old Circuit: Turn LED on/off via pin 3

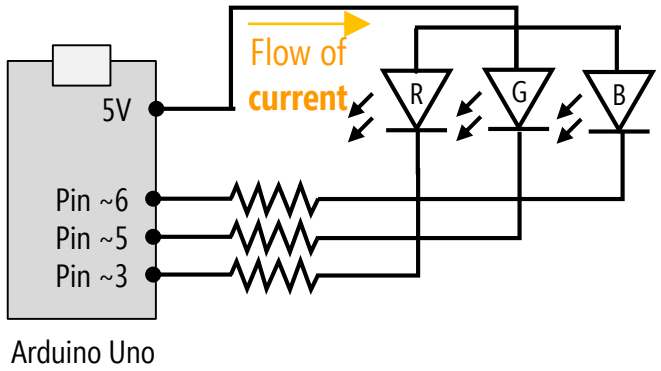




New Circuit: Set R, B, G values via pin 6, 5, and 3

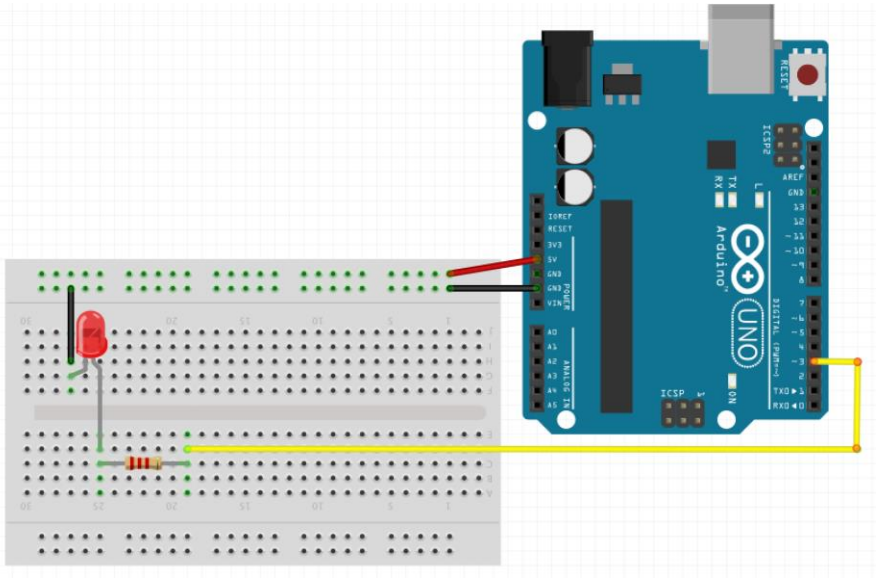


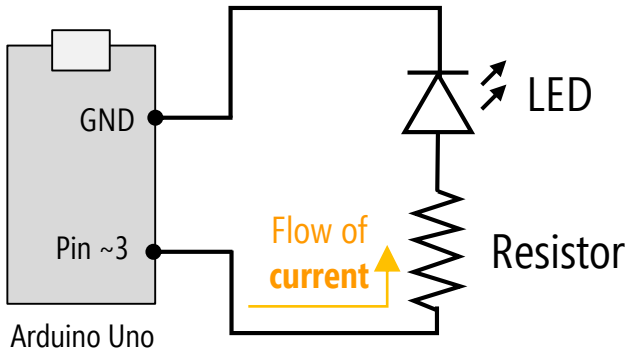
Flow of current
Because this is a "common anode" RGB. We set these pins to zero to turn on the light and 5V to turn off the light!



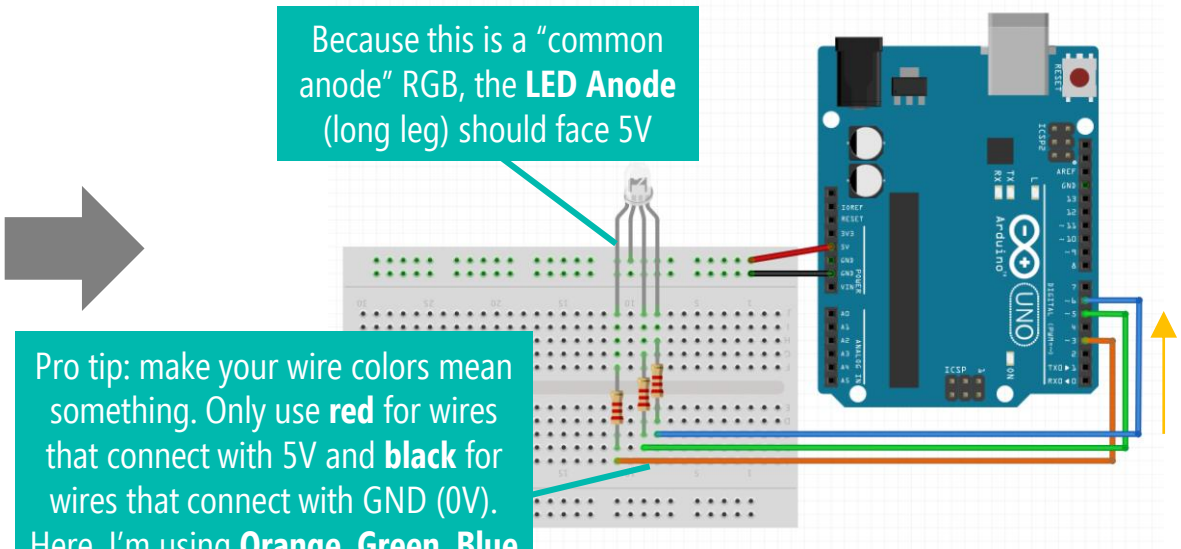
USING AN RGB LED: THE CIRCUIT

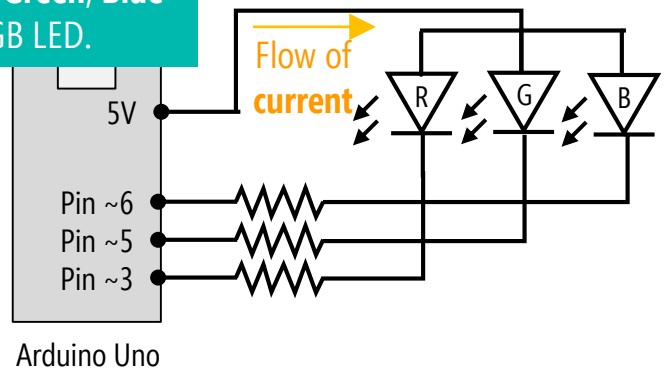
Old Circuit: Turn LED on/off via pin 3





New Circuit: Set R, B, G values via pin 6, 5, and 3

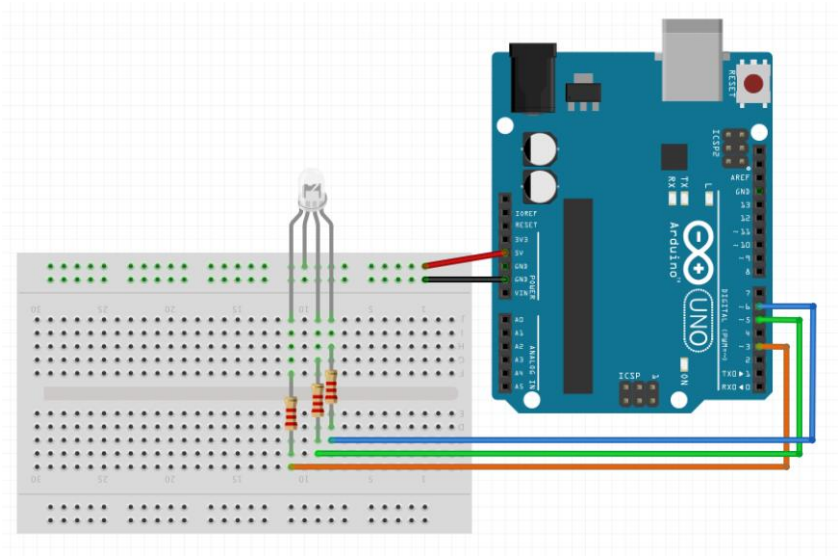


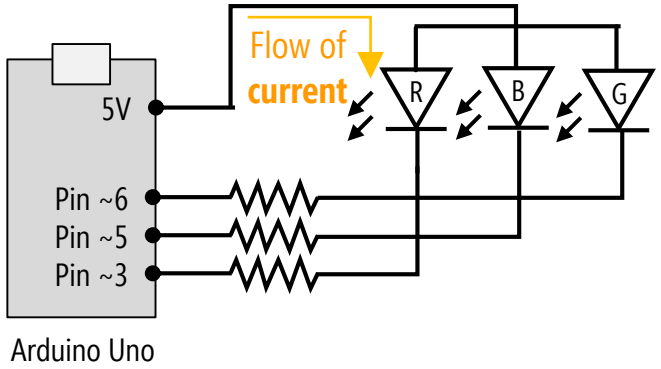


RGB LEDS

USING AN RGB LED: THE CODE

Circuit: Set R, B, G values via pin 6, 5, and 3





BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LEDS

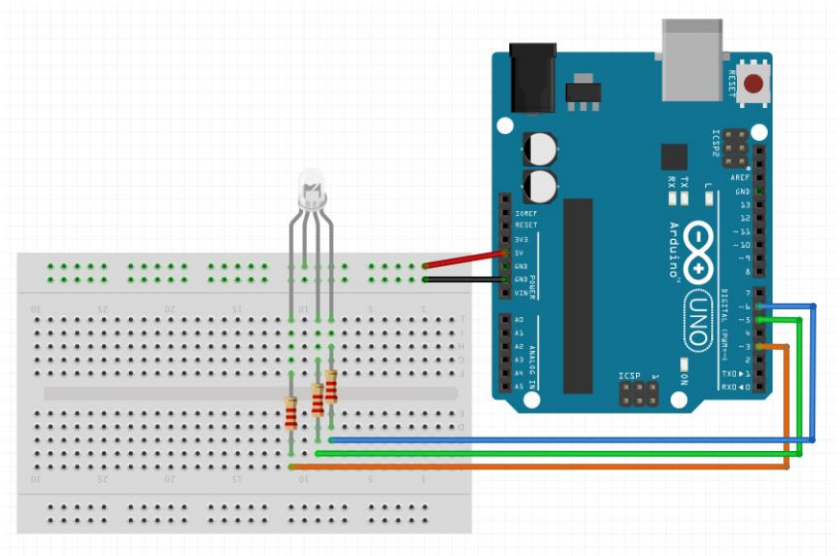
USING AN RGB LED: THE CODE

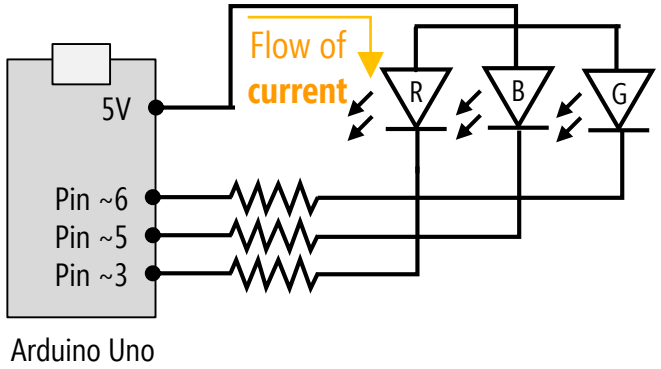
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}
```

```
void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





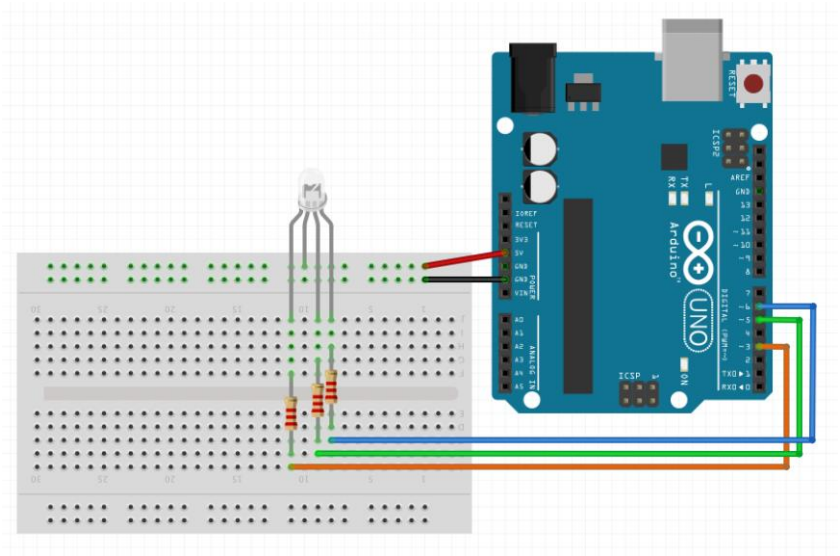
Arduino starts. **setup()** is called immediately once and only once.

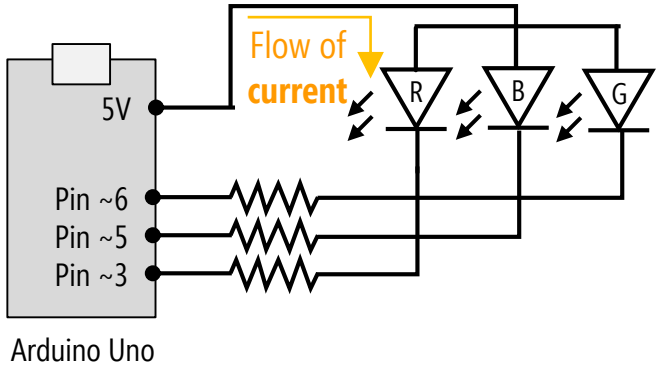
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





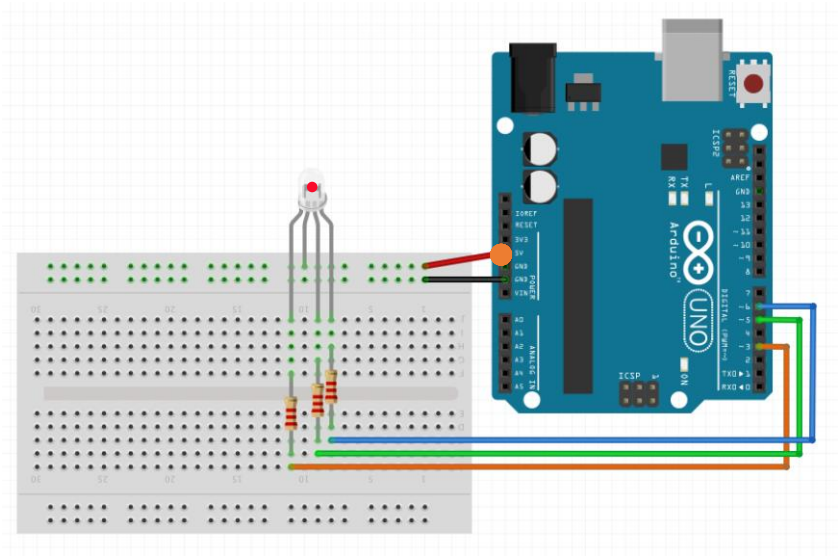
The **RGB LED has three legs** (one for red, one for green, and one for blue), so we need **to set all three pins** as **output pins**

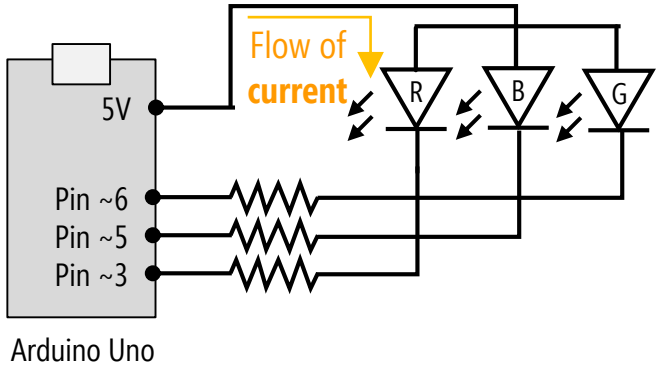
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





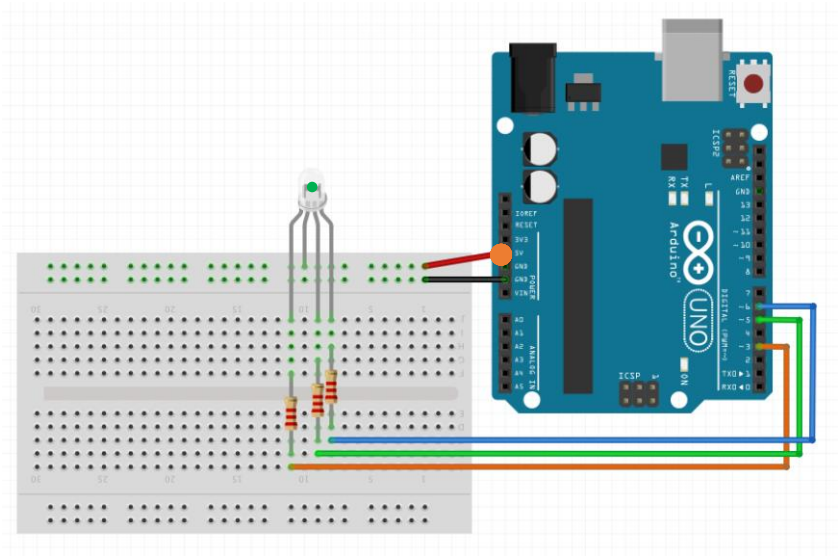
Sets the RGB LED to **red** by **writing out 0 (0V)** to the red pin of the RGB LED and 255 (5V) to the green and blue pins. Wait for one second

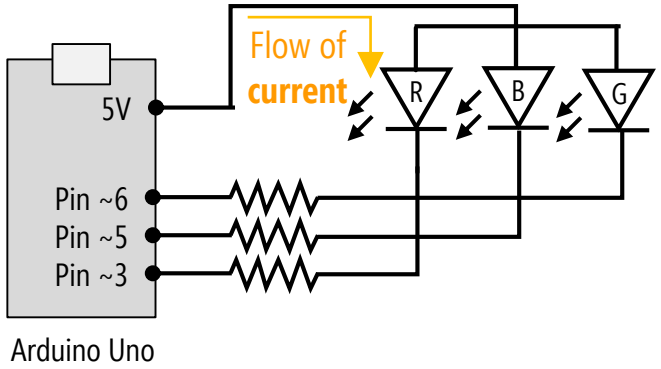
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





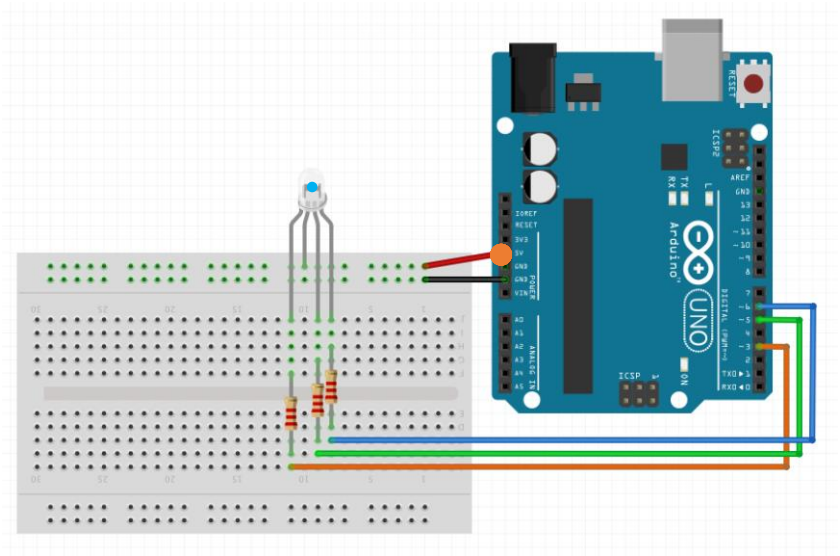
Sets the RGB LED to **green** by writing out 0 (0V) to the green pin of the RGB LED and 255 (5V) to the red and blue pins. Stay like this for one second (as caused by the delay line)

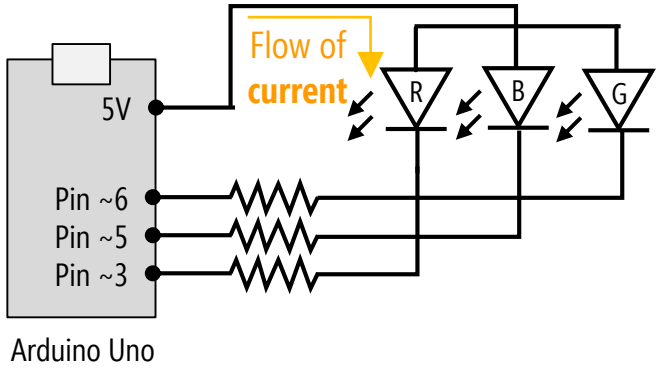
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





Sets the RGB LED to **blue** by writing out 0 (0V) to the blue pin of the RGB LED and 255 (5V) to the red and green pins. Stay like this for one second (as caused by the delay line)

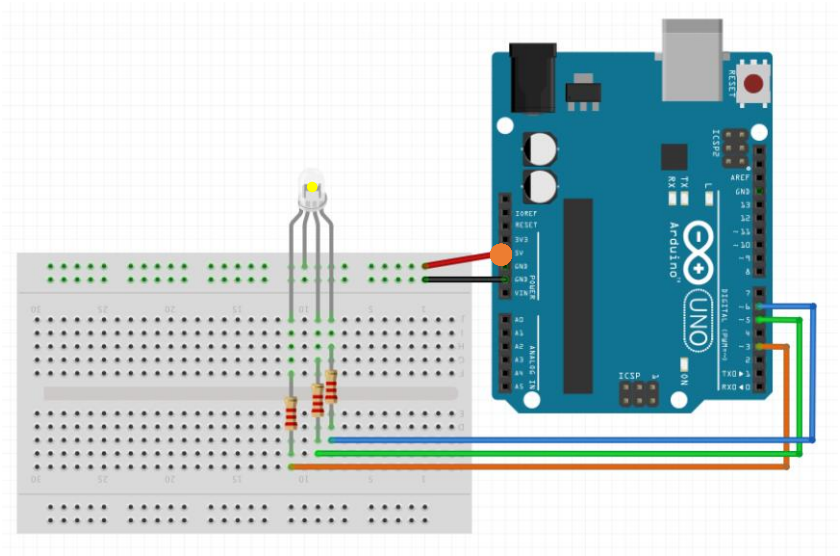
BlinkRGBSimple §

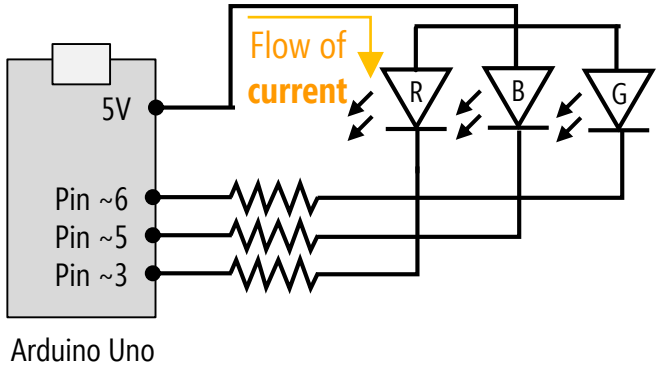
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LEDS

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





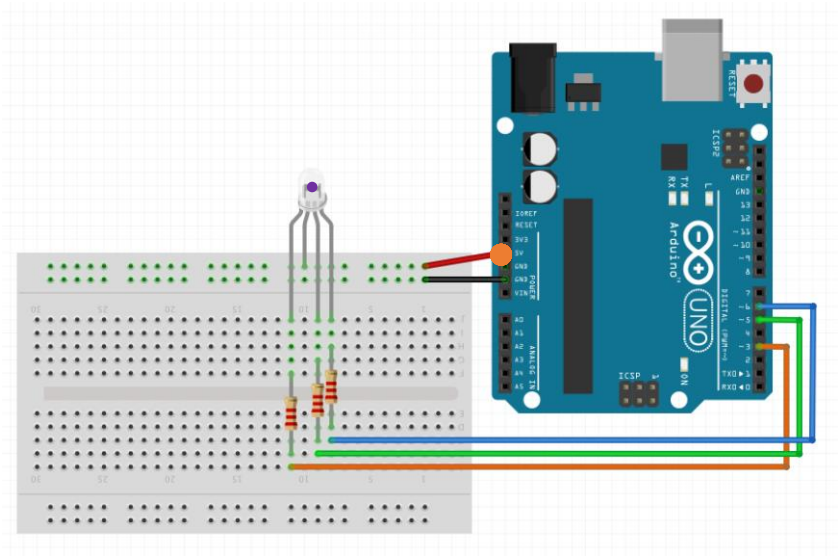
Sets the RGB LED to **yellow** by writing out 0 (0V) to the green and blue pins of the RGB LED and 255 (5V) to the red pin. Stay like this for one second (as caused by the delay line)

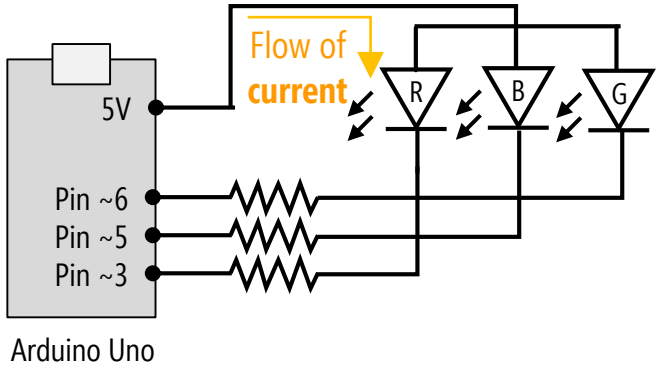
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3





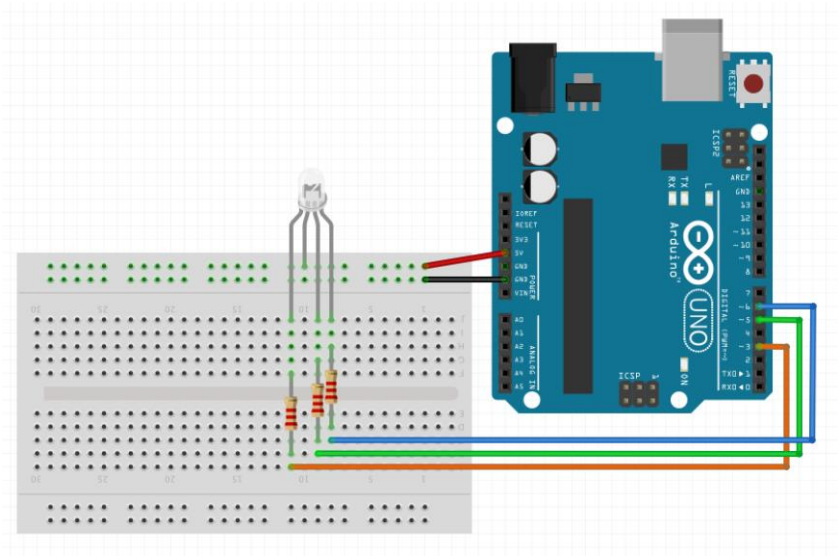
Sets the RGB LED to **purple** by writing out 0 (0V) to the red and blue pins of the RGB LED and 255 (5V) to the green pin. Stay like this for one second (as caused by the delay line)

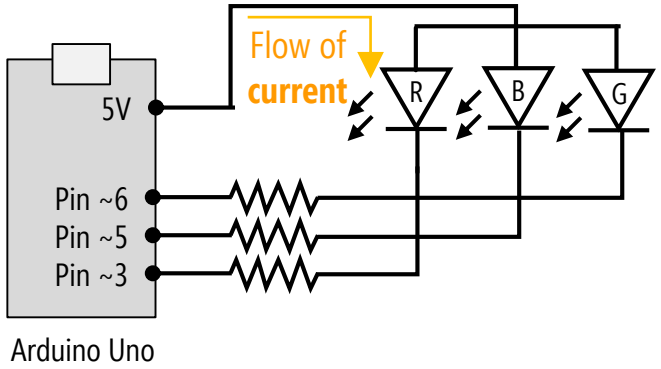
BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

RGB LED WALKTHROUGH

Circuit: Set R, B, G values via pin 6, 5, and 3



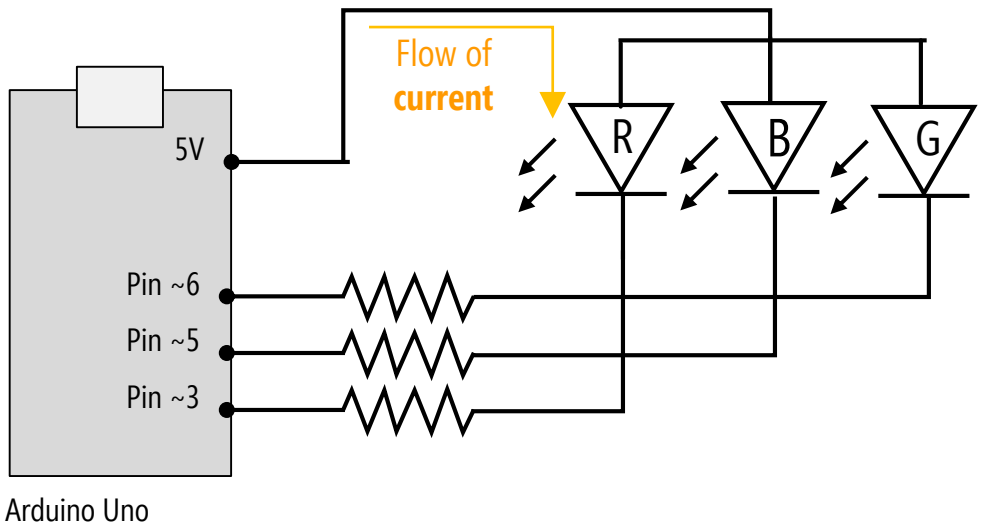


When the loop() function ends, it's automatically called again by the Arduino (and will continue looping forever)

BlinkRGBSimple §

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}

void loop() {
  // Set the RGB LED to red
  Serial.println("Color=Red (LOW, HIGH, HIGH)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH);   // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  Serial.println("Color=Blue (HIGH, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH);    // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW);   // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  Serial.println("Color=Purple (LOW, HIGH, LOW)");
  digitalWrite(RGB_RED_PIN, LOW);     // turn on the red LED
  digitalWrite(RGB_GREEN_PIN, HIGH);  // turn off the green LED
  digitalWrite(RGB_BLUE_PIN, LOW);    // turn on the blue LED
  delay(DELAY);
}
```

FOR MORE PRECISE COLOR, USE DIFFERENT RESISTORS FOR EACH LED LEG



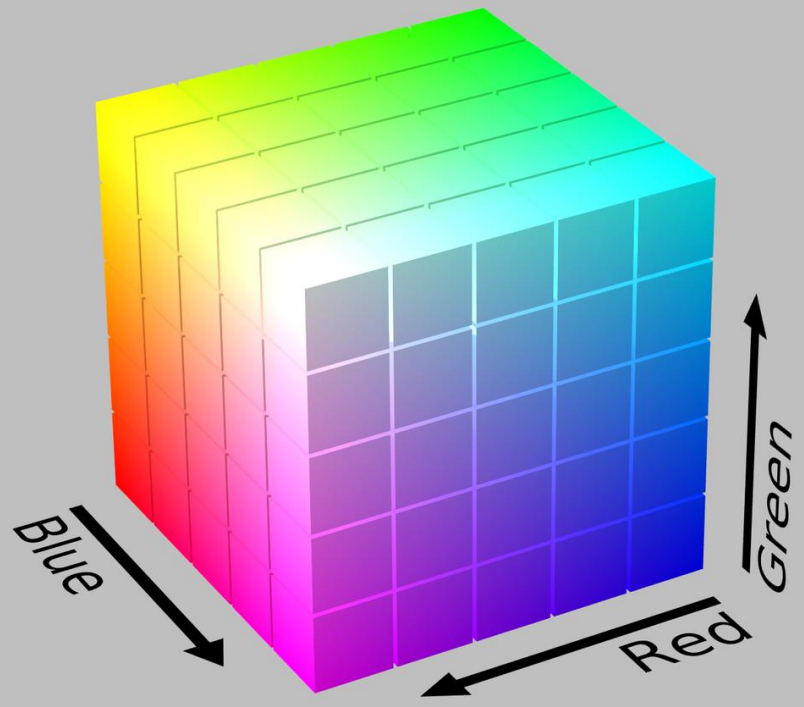
The **three LEDs** embedded inside an RGB LED typically have different **operating voltages** and current levels

For example, a **red LED** may need **2 volts** while a **blue LED** may need **3**

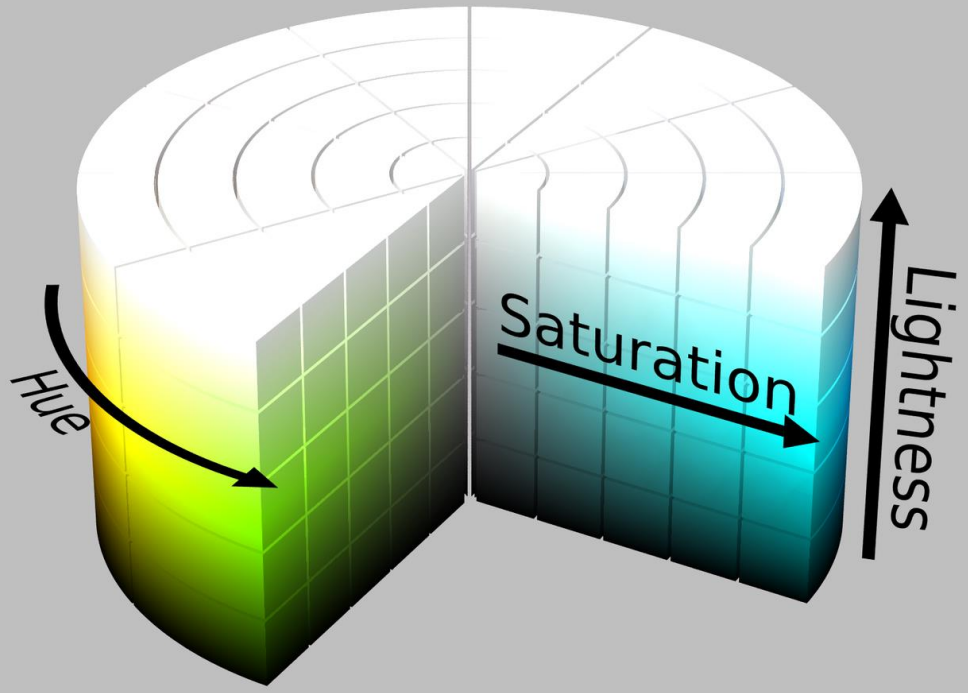
Consult the **datasheet** and set the resistor values appropriately

EASIER TO MANIPULATE COLOR IN HSL COLORSPACE

RGB Colorspace

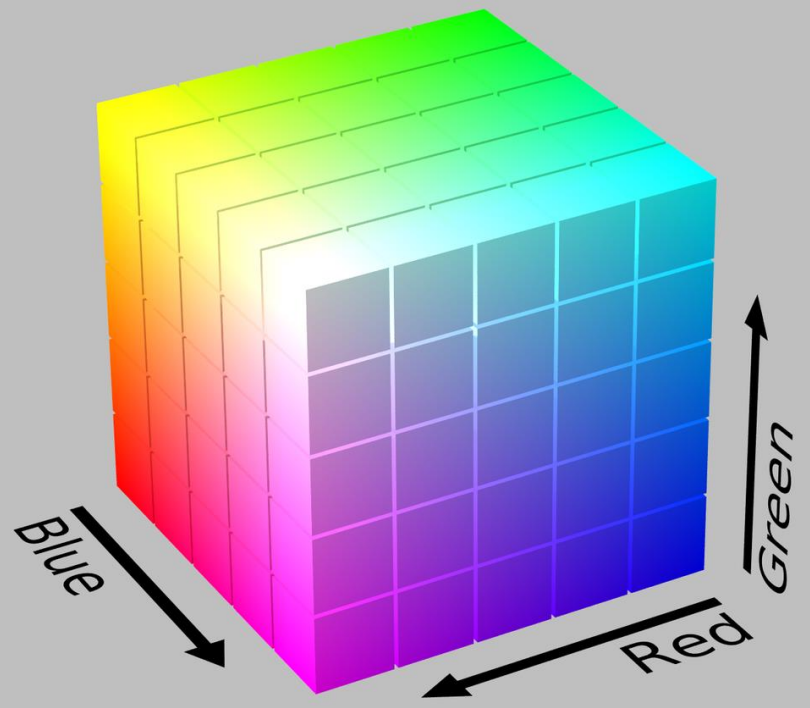


HSL Colorspace



COLOR SPACES

RGB COLORSPACE



RGB colorspace can be represented by a cube. Each dimension corresponds to the amount of **Red**, **Green**, **Blue** intensities.

```
(255, 0, 0) => Red
(0,0,255) => Blue
(0,255,255) => Yellow
(255,255,0) => Violet
```

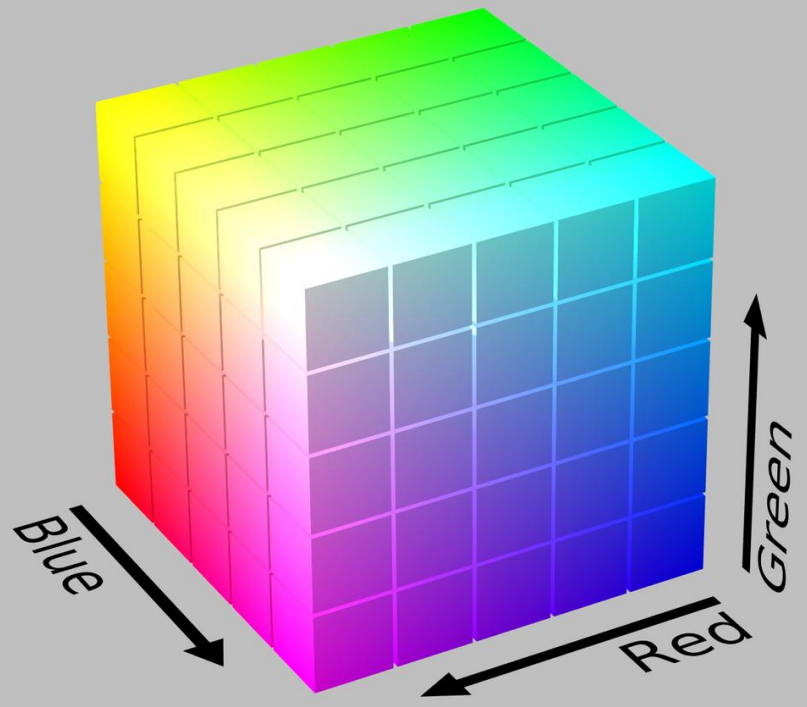
But how can you independently control **brightness**?

Source: https://en.wikipedia.org/wiki/HSL_and_HSV

COLOR SPACES

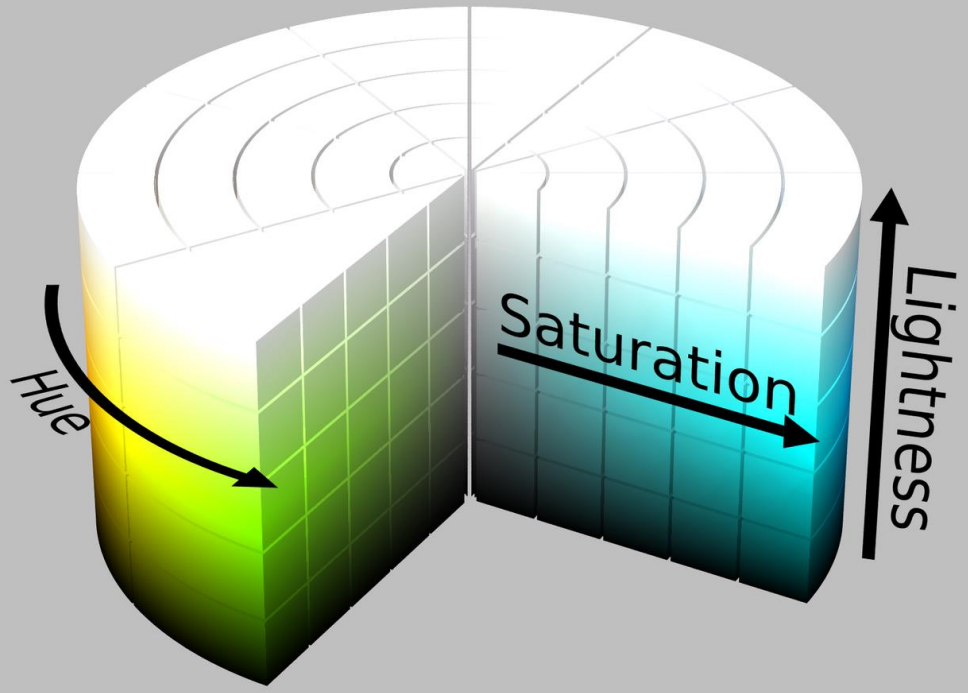
RGB VS. HSL (HUE, SATURATION, LIGHTNESS)

RGB Colorspace



RGB => (0-1.0, 0-1.0, 0-1.0)

HSL Colorspace



HSL => (0-360, 0-1.0, 0-1.0)

Source: https://en.wikipedia.org/wiki/HSL_and_HSV

COLOR SPACES

HSL (HUE, SATURATION, LIGHTNESS)

SATURATION

Intensity of the hue

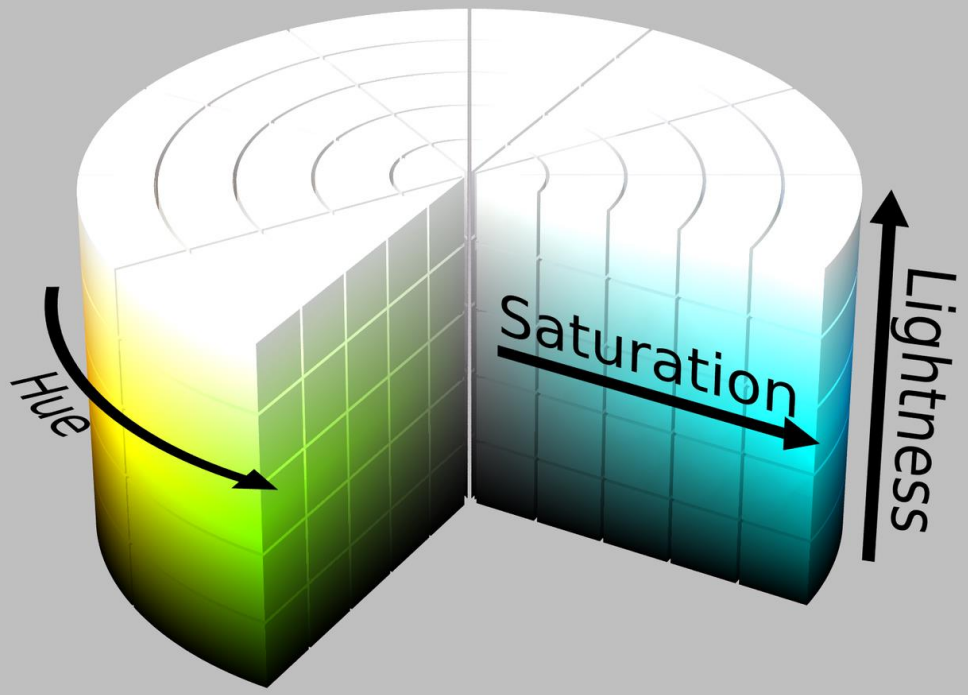
0% 100%

LIGHTNESS

Measure of brightness from dark (black) to bright (white)

0% 50% 100%

HSL Colorspace

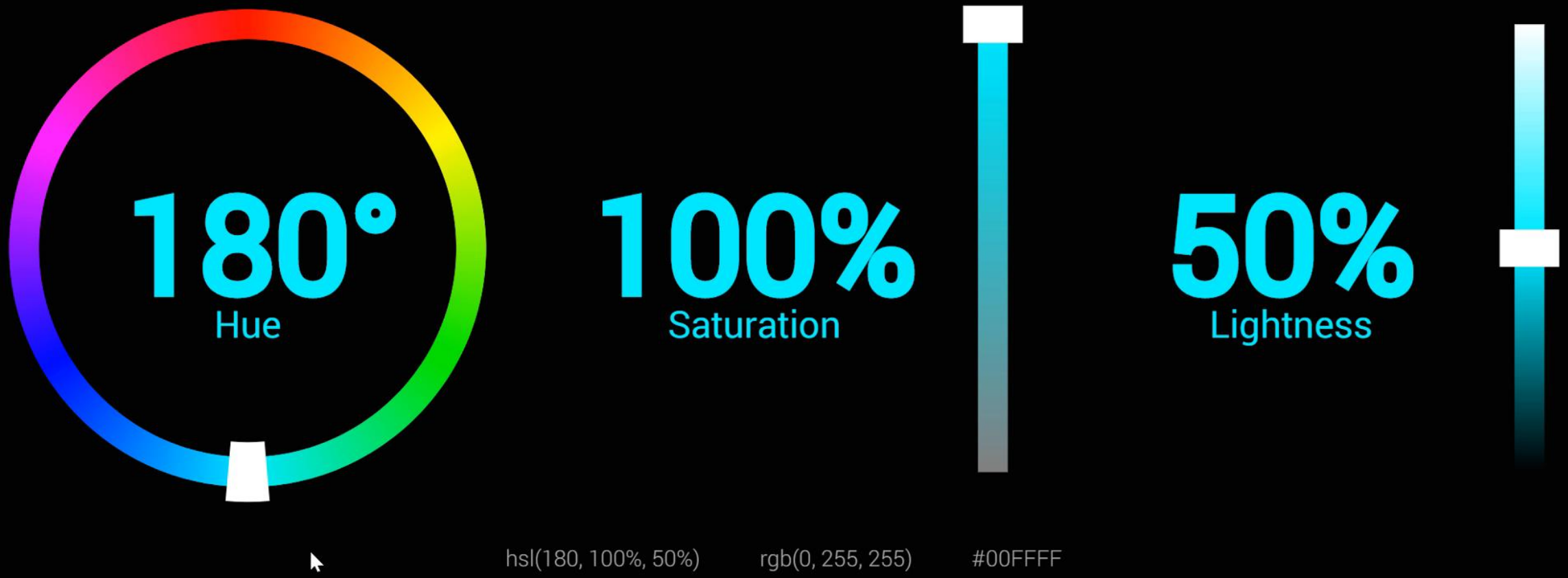


HSL => (0-360, 0-1.0, 0-1.0)

Source: https://en.wikipedia.org/wiki/HSL_and_HSV

COLOR SPACES

HSL COLORSPACE DEMO



Source: https://codepen.io/HunorMarton/pen/dvXVvQ/

CROSS FADING COLORS USING HUE

CrossFadeHue §

```
const boolean COMMON_ANODE = true;
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY_INTERVAL = 10; // interval in ms between incrementing hues
float _hue = 0; //hue varies between 0 - 1
float _step = 0.1f;

RGBConverter _rgbConverter;

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);

  // Turn on Serial so we can verify expected colors via Serial Monitor
  Serial.begin(9600);
}
```

```
void loop() {

  // Convert the current hue to RGB
  byte rgb[3];
  _rgbConverter.hslToRgb(hue, 1.0, 1.0, rgb);


  // Set the LED color
  setColor(rgb[0], rgb[1], rgb[2]);

  // Reset the hue to 0.0
  _hue += _step;
  if(_hue > 1.0){
    _hue = 0;
  }

  delay(DELAY_INTERVAL);
}
```

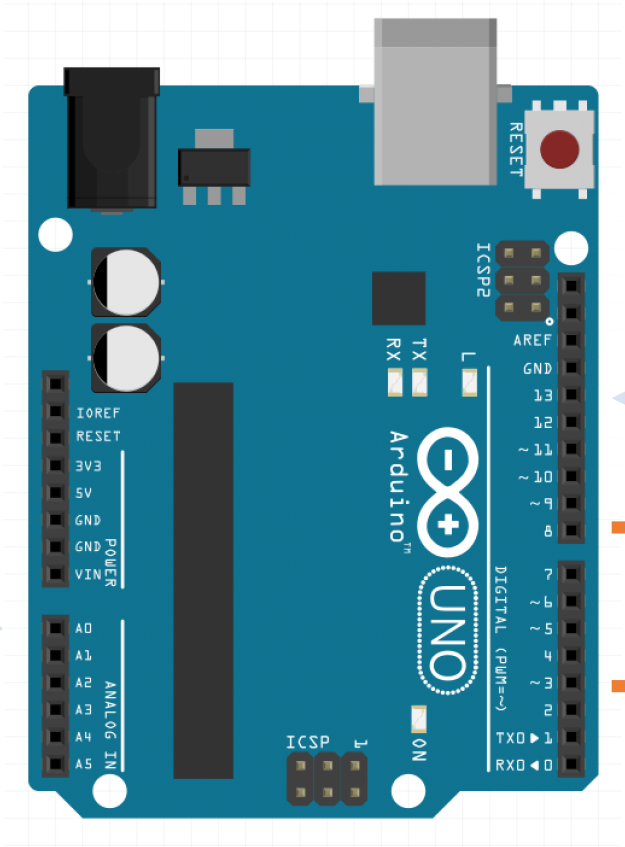
Source: https://github.com/jonfroehlich/CSE599Au2019/tree/master/Code/Arduino/CrossFadeHue

DIGITAL AND ANALOG OUTPUT

This completes our quick tour of digital and analog output on the Arduino

Analog Input on A0

Reads in any value between 0V or 5V using the analogRead function. In this case, the value of a photocell



Digital Input on pin 13

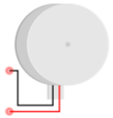
Reads in a digital input signal (anything below 2.5V converted to LOW, anything above 2.5V converted to HIGH). In this case, the value of switch.

Digital Output on pin 8

Writes out 0V or 5V using digitalWrite function. In this case, turning on and off an LED.

Analog Output on pin 3

Writes out any value between 0V or 5V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage). Only pins with a tilde ~ can be analog outputs.

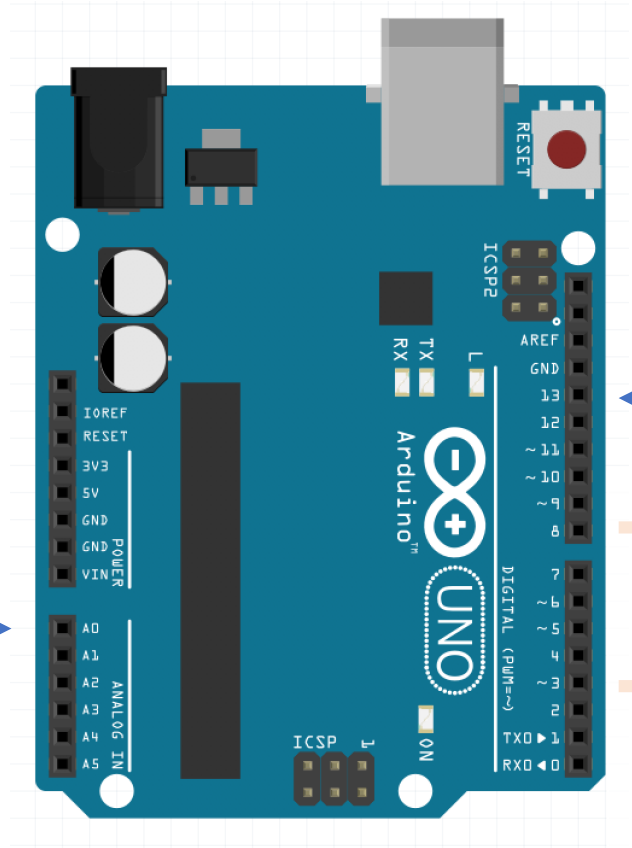


DIGITAL AND ANALOG OUTPUT

Now we are going to switch from talking about output to talking about input

Analog Input on A0

Reads in any value between 0V or 5V using the analogRead function. In this case, the value of a photocell



Digital Input on pin 13

Reads in a digital input signal (anything below 2.5V converted to LOW, anything above 2.5V converted to HIGH). In this case, the value of switch.

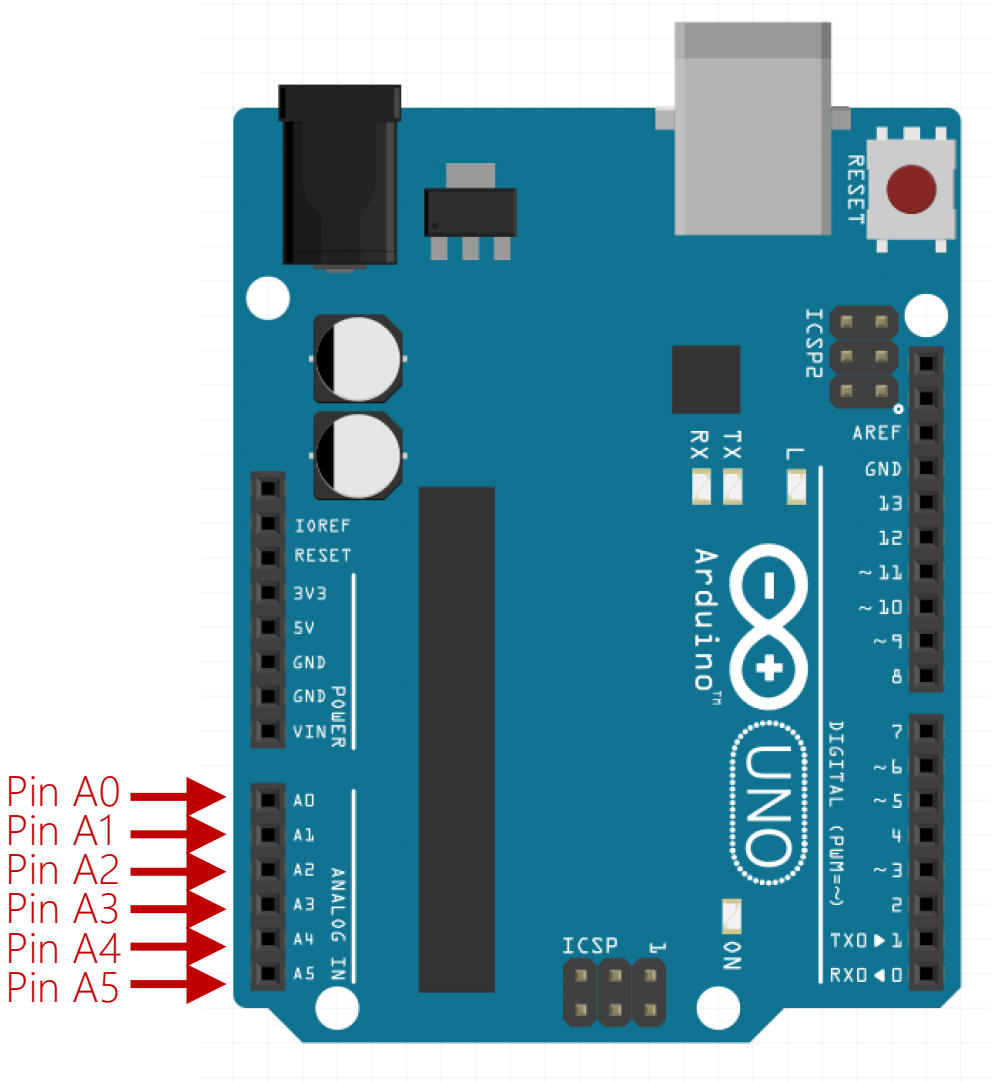
Digital Output on pin 8

Writes out 0V or 5V using digitalWrite function. In this case, turning on and off an LED.

Analog Output on pin 3

Writes out any value between 0V or 5V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage). Only pins with a tilde ~ can be analog outputs.

RECALL THAT WE ONLY HAVE SIX ANALOG INPUTS

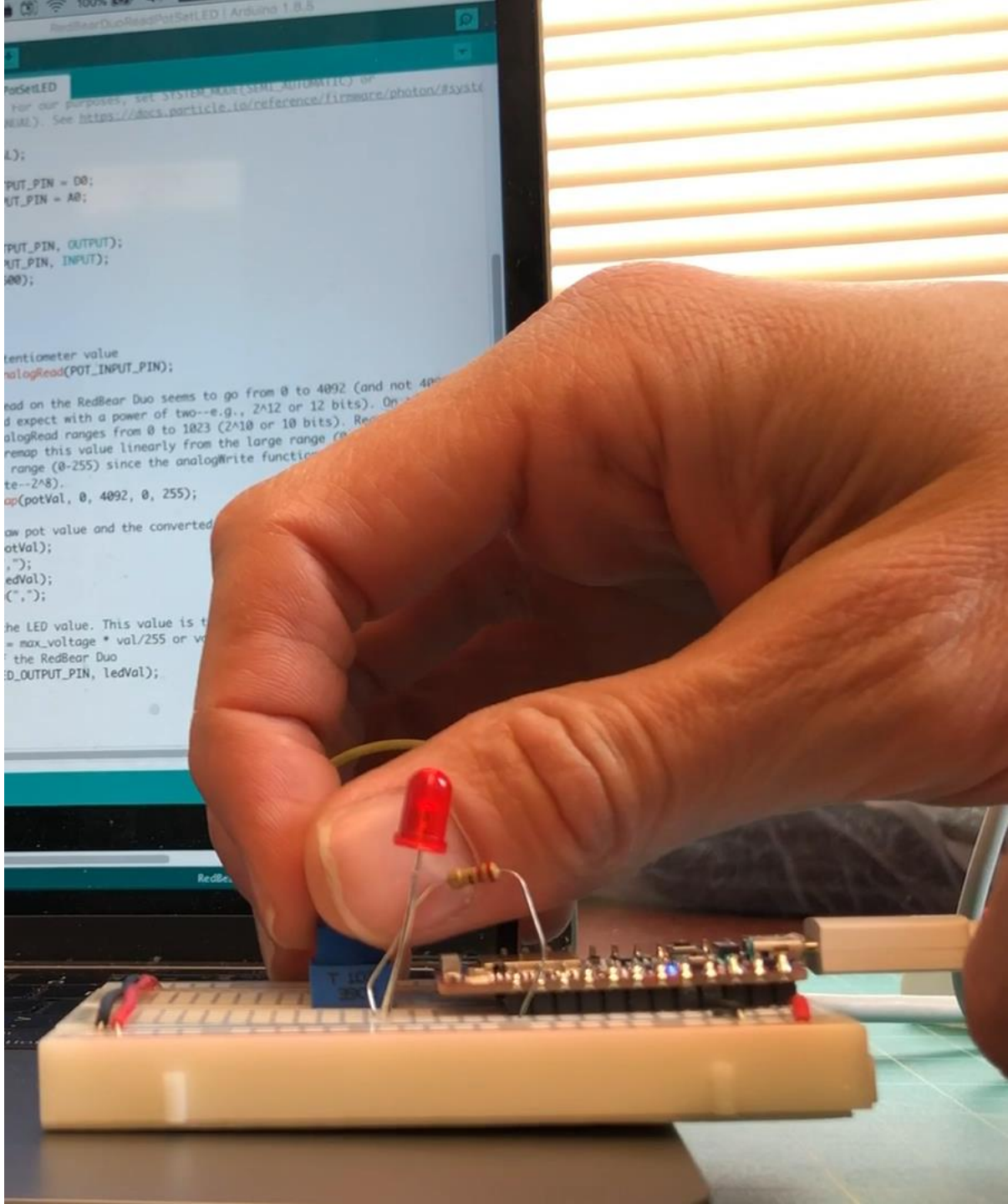


Let’s build a circuit and write code to **fade an LED based on the value of a trim potentiometer**. For this, we will use analog input (analogRead) and output (analogWrite)

ACTIVITY: CONTROL LED BRIGHTNESS WITH TRIMPOT

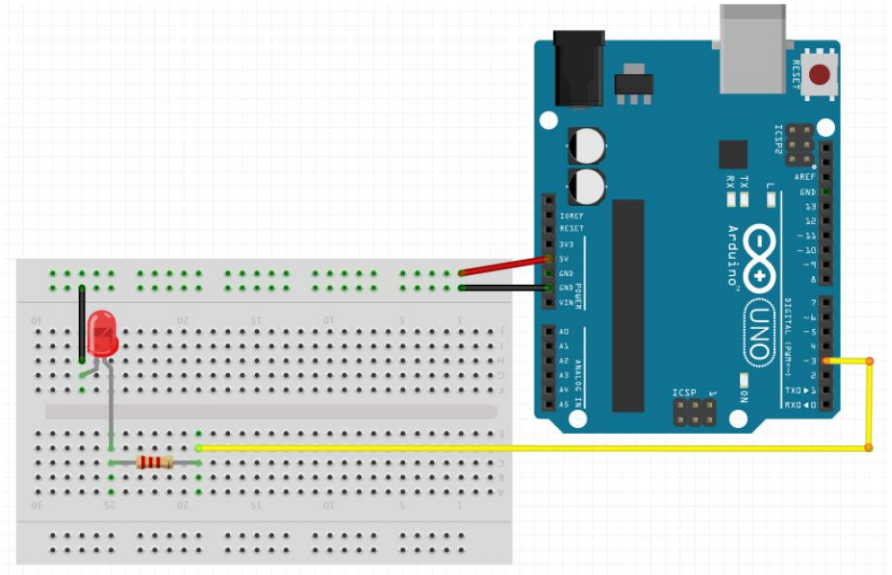
Same output circuit as before

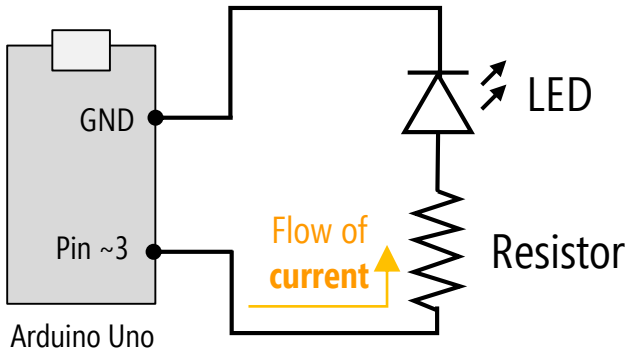
Now we need to add input



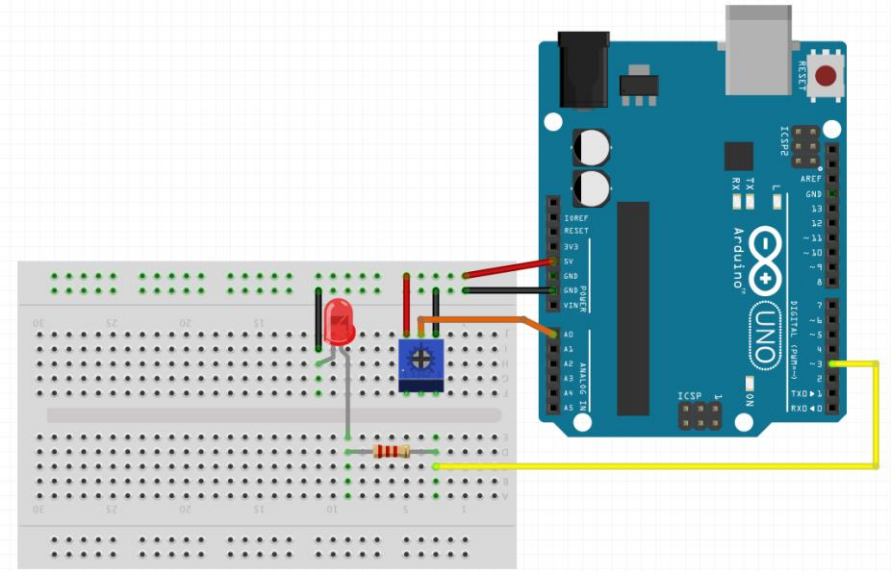
ACTIVITY: CONTROL LED BRIGHTNESS WITH POTENTIOMETER

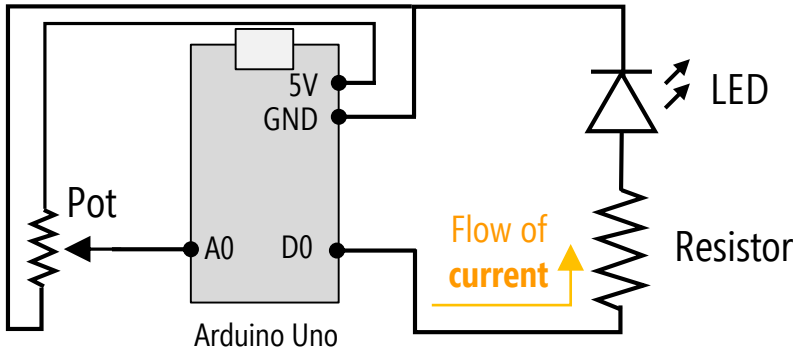
Old Circuit: Fade LED on/off via pin 3





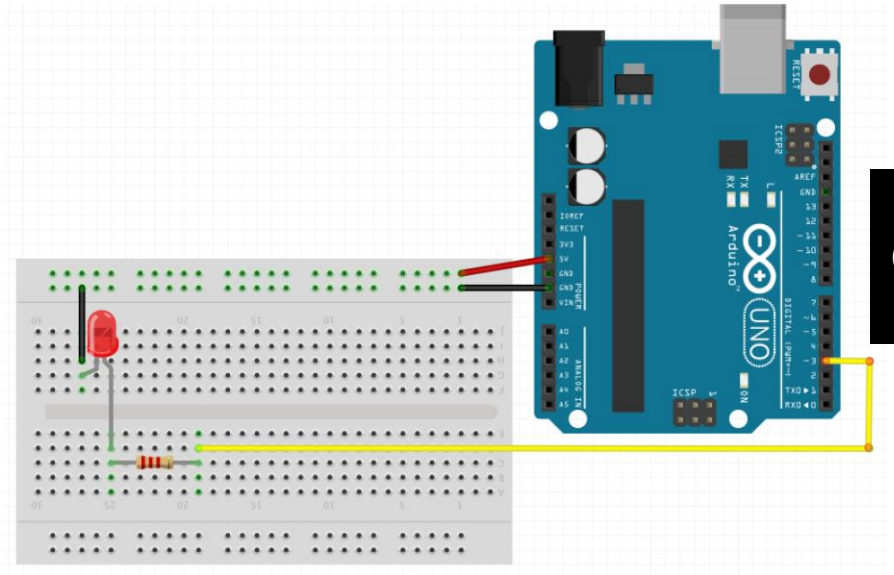
New Circuit: Fade LED on/off via pot value

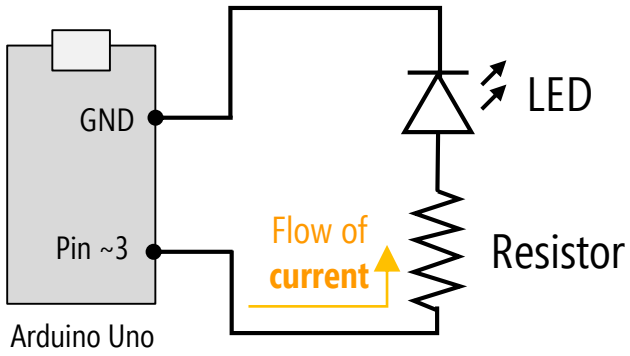




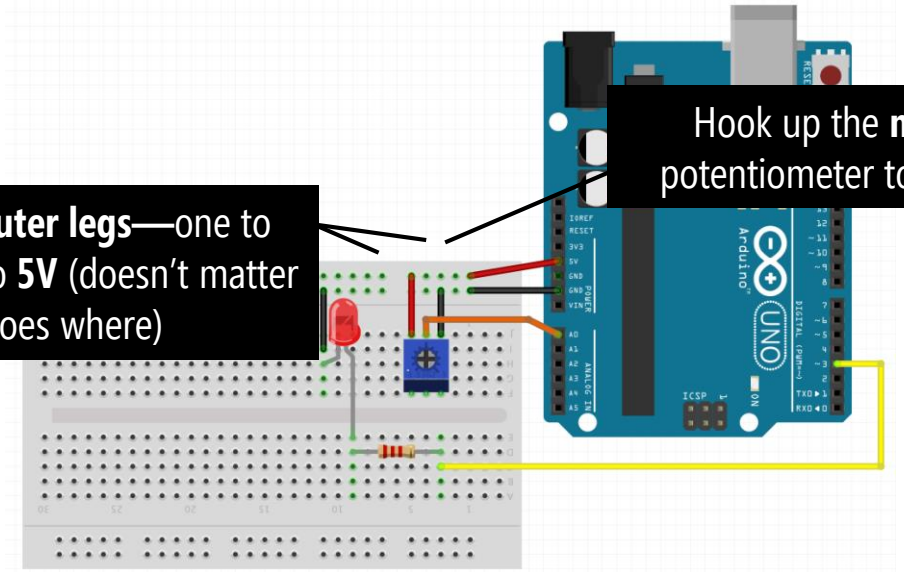
ACTIVITY: CONTROL LED BRIGHTNESS WITH POTENTIOMETER

Old Circuit: Fade LED on/off via pin 3



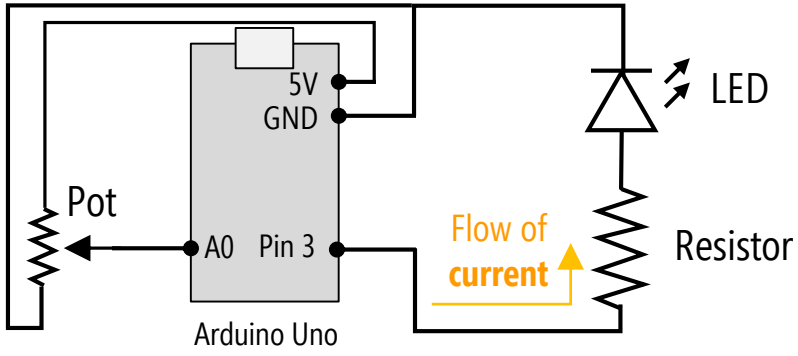


New Circuit: Fade LED on/off via pot value



Hook up the **two outer legs**—one to **GND** and the other to **5V** (doesn't matter which one goes where)

Hook up the **middle leg** of the potentiometer to **A0** on **Analog In**.



ANALOG INPUT

int analogRead()

Reads the voltage from the specified analog pin & returns an integer (0 – 1023)

Syntax

`analogRead(pin)`

Parameters

`pin`: the analog pin to read from

Returns

An integer between 0 and 1023 (inclusive).

TrimpotLED

```
// The Arduino Uno ADC is 10 bits (thus, 0 - 1023 values)
#define MAX_ANALOG_INPUT_VAL 1023

const int LED_OUTPUT_PIN = 3;
const int POT_INPUT_PIN = A0;

void setup() {
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  pinMode(POT_INPUT_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {

  // read the potentiometer value
  int potVal = analogRead(POT_INPUT_PIN);

  // the analogRead on the Arduino Uno goes from 0 to 1023. We need to remap
  // this value to the smaller range (0-255) since the analogWrite function can
  // only write out 0-255 (a byte--2^8). The map function provides a linear
  // mapping to do this (however, a better way would likely be some sort of
  // non-linear mapping given that perceived LED brightness is not linear with current,
  // perhaps logarithmic)
  int ledVal = map(potVal, 0, MAX_ANALOG_INPUT_VAL, 0, 255);

  // print the raw pot value and the converted led value
  Serial.print(potVal);
  Serial.print(",");
  Serial.println(ledVal);

  // write out the LED value. This value is translated to voltage by:
  // voltageVal = max_voltage * val/255 or voltageVal = 3.3V * val/255 in
  // the case of the RedBear Duo
  analogWrite(LED_OUTPUT_PIN, ledVal);

  delay(100);
}
```

https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

ANALOG INPUT

int analogRead()

Reads the voltage value from the specified analog pin and returns it as an integer (0 – 1023)

Syntax

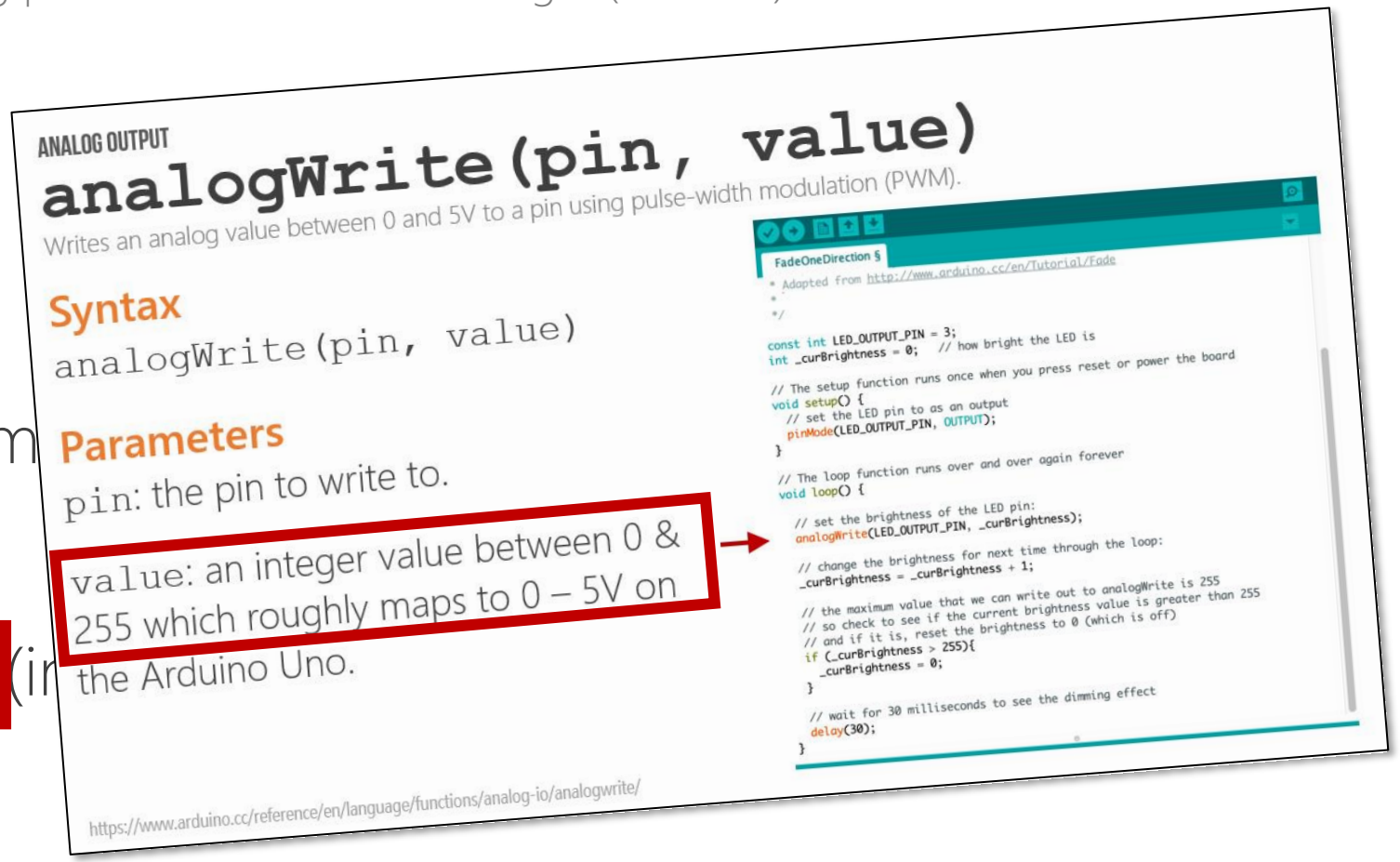
`analogRead(pin)`

Parameters

`pin`: the analog pin to read from

Returns

An integer between 0 and 1023 (in



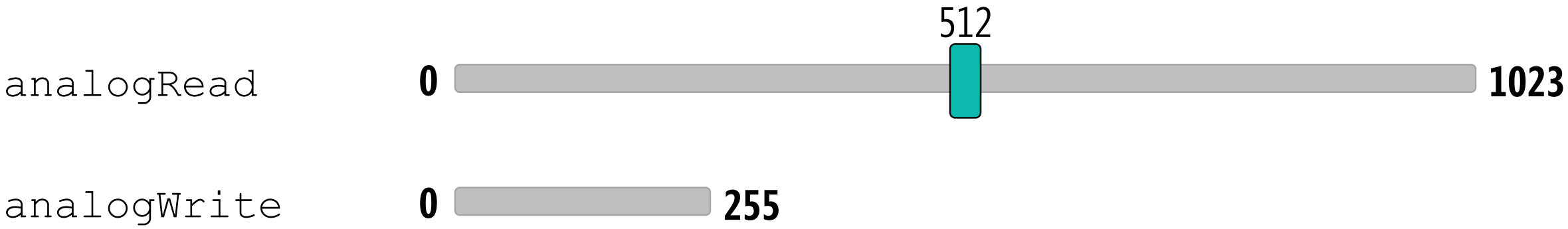
https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

USING ANALOGREAD & ANALOGWRITE TOGETHER

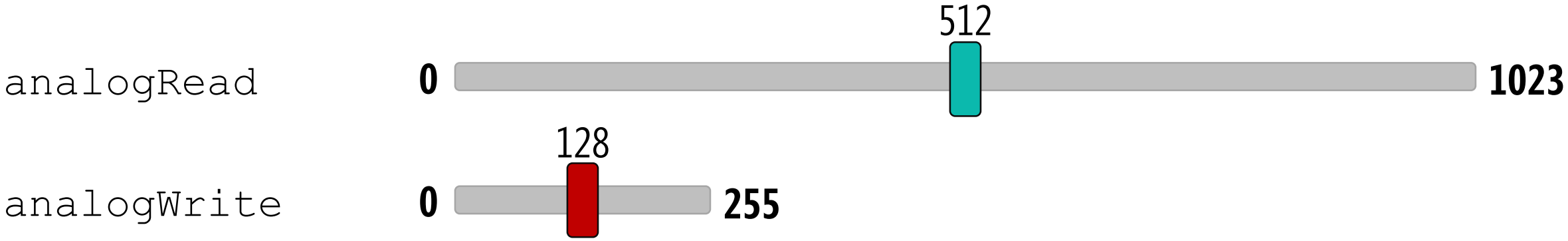
We need to convert a value in the 0 – 1023 range to an equivalent value in the 0 – 255 range.

How do we do this?

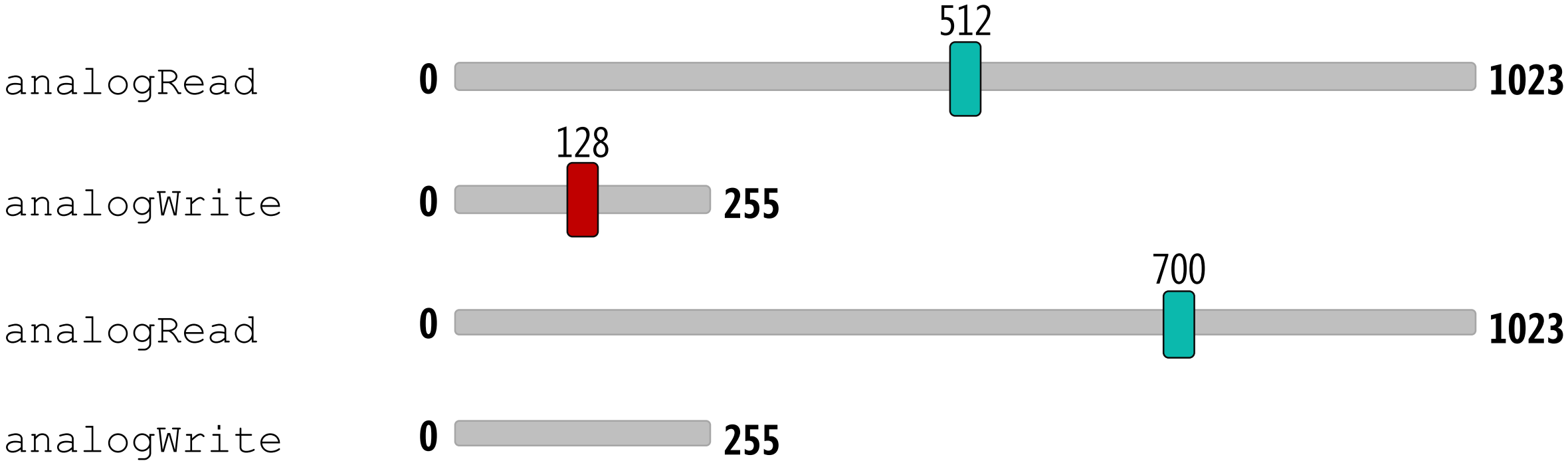
CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



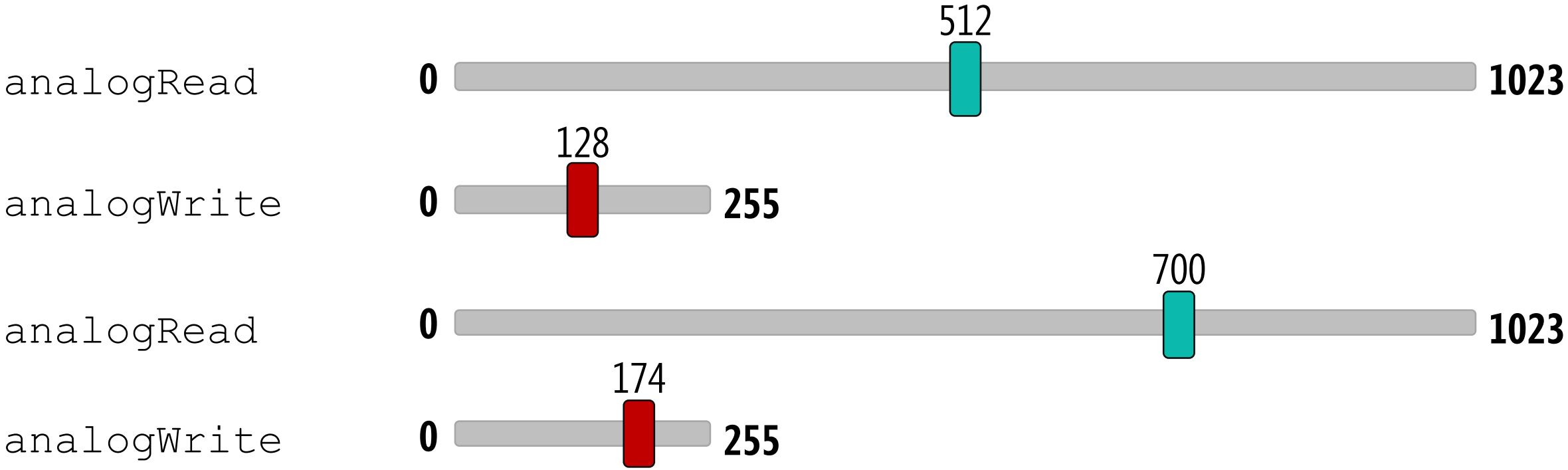
CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



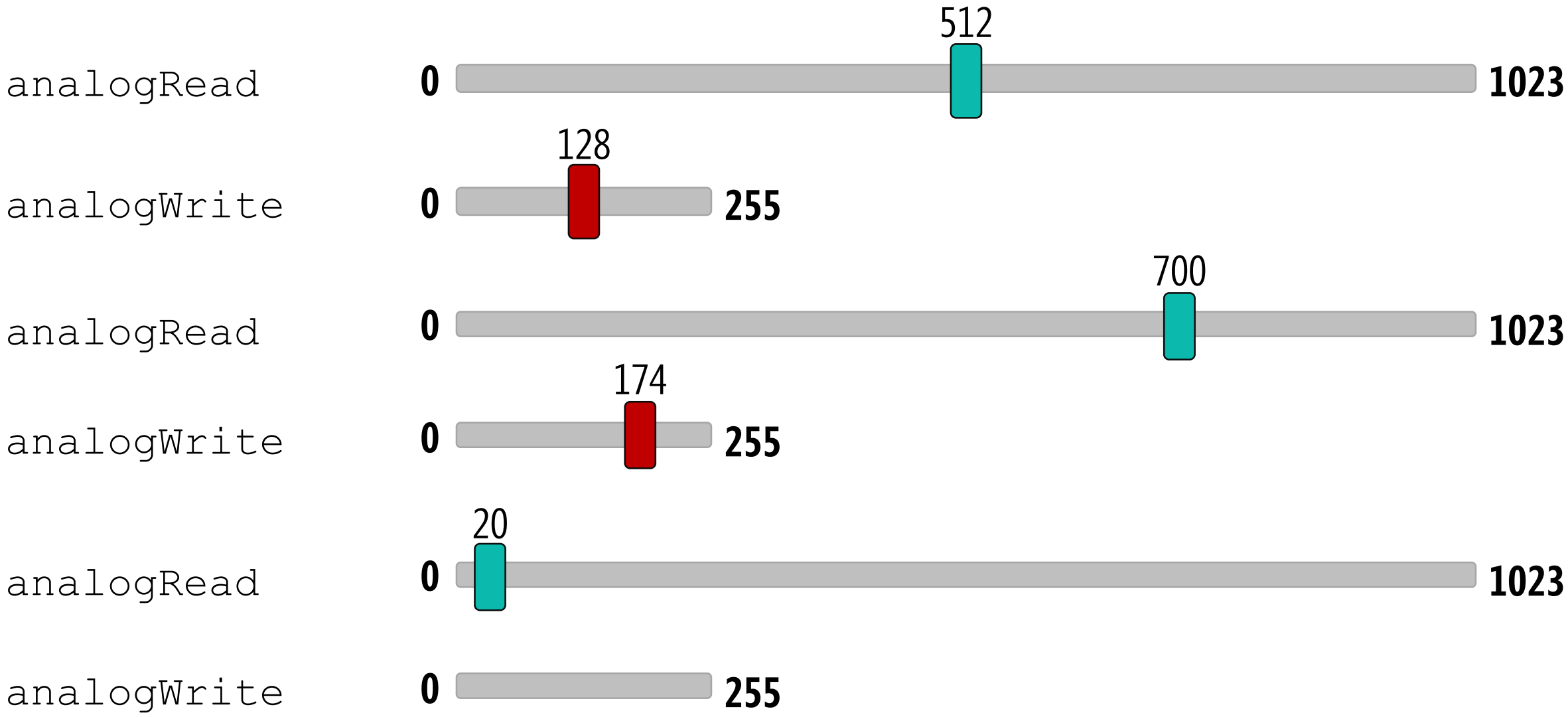
CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



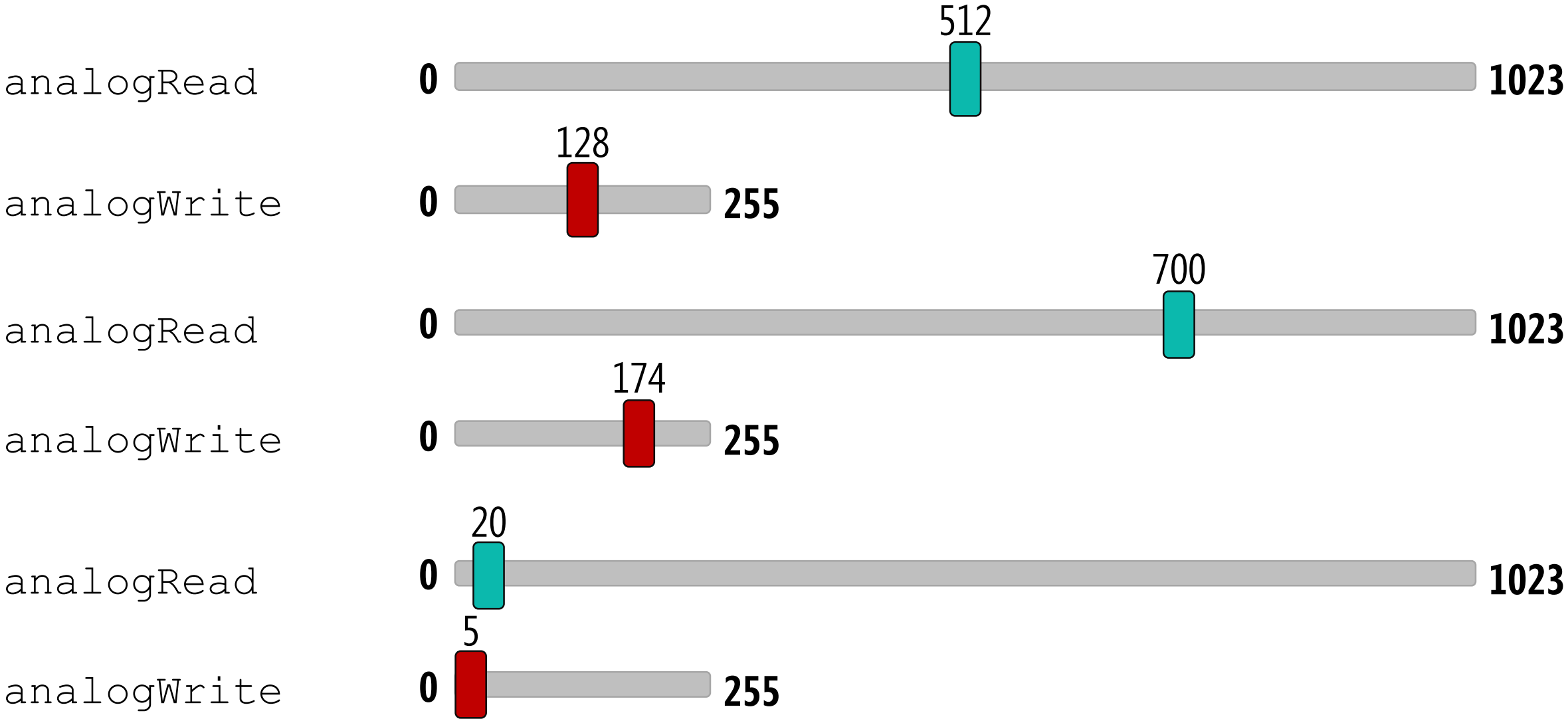
CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



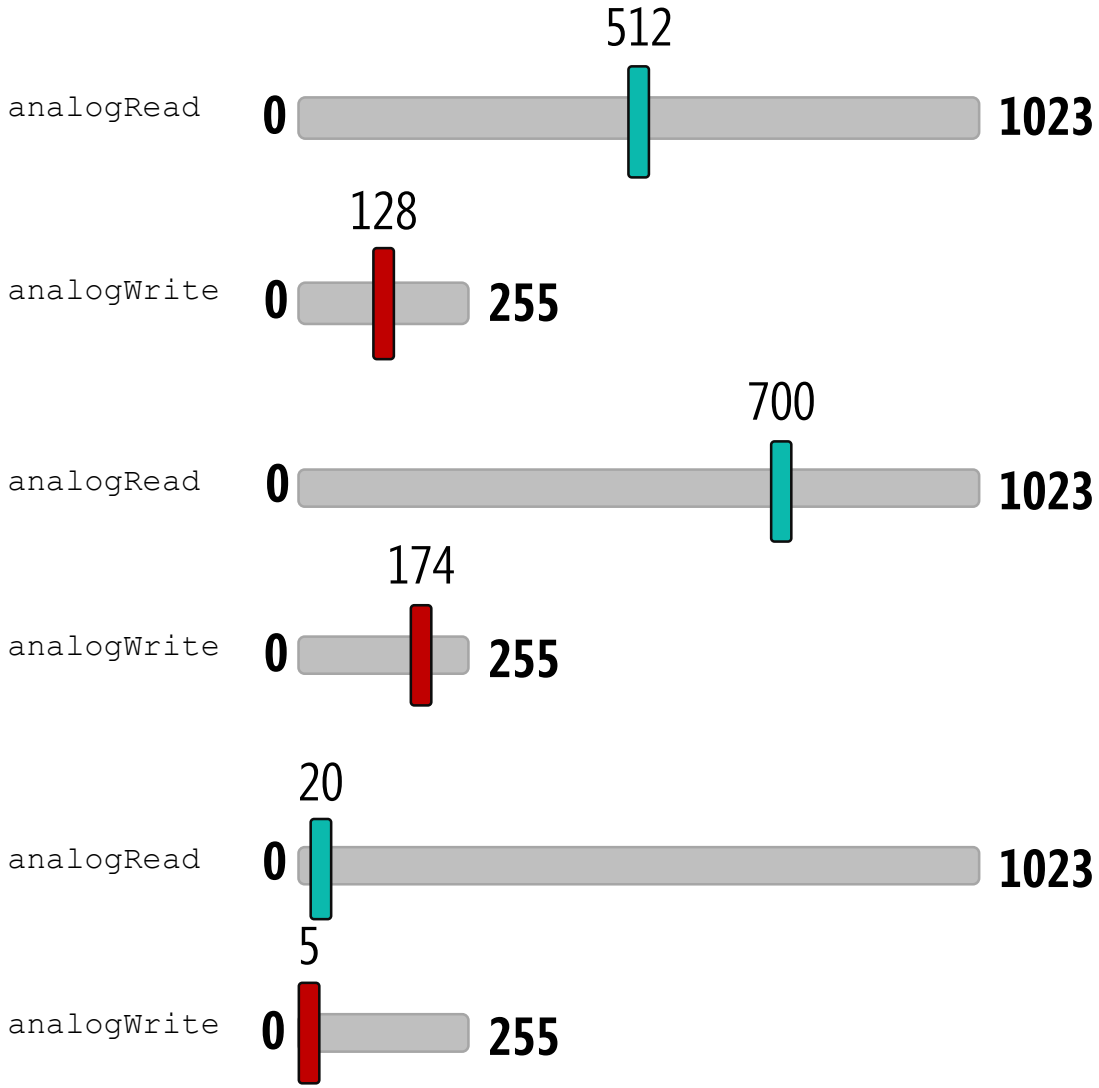
CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



CONVERTING ANALOGREAD VALS TO ANALOGWRITE VALS



`analogRead`

$$Analog_{input\_fraction} = \frac{Analog_{input\_value}}{Analog_{input\_max}}$$

`analogWrite`

$$Analog_{output\_value} = Analog_{input\_fraction} * Analog_{output\_max}$$

CONVERSION EQUATION GENERALIZED

`analogRead`

$$Analog_{input\_fraction} = \frac{Analog_{input\_value}}{Analog_{input\_max}}$$

`analogWrite`

$$Analog_{output\_value} = Analog_{input\_fraction} * Analog_{output\_max}$$

Generic form to support non-zero ranges

`analogRead`

$$Analog_{input\_fraction} = \frac{(Analog_{input\_value} - Analog_{input\_min})}{(Analog_{input\_max} - Analog_{input\_min})}$$

`analogWrite`

$$Analog_{output\_value} = Analog_{output\_min} + Analog_{input\_fraction} * (Analog_{output\_max} - Analog_{output\_min})$$

CONVERSION EQUATION GENERALIZED

`analogRead`

$$Analog_{input\_fraction} = \frac{Analog_{input\_value}}{Analog_{input\_max}}$$

`analogWrite`

$$Analog_{output\_value} = Analog_{input\_fraction} * Analog_{output\_max}$$

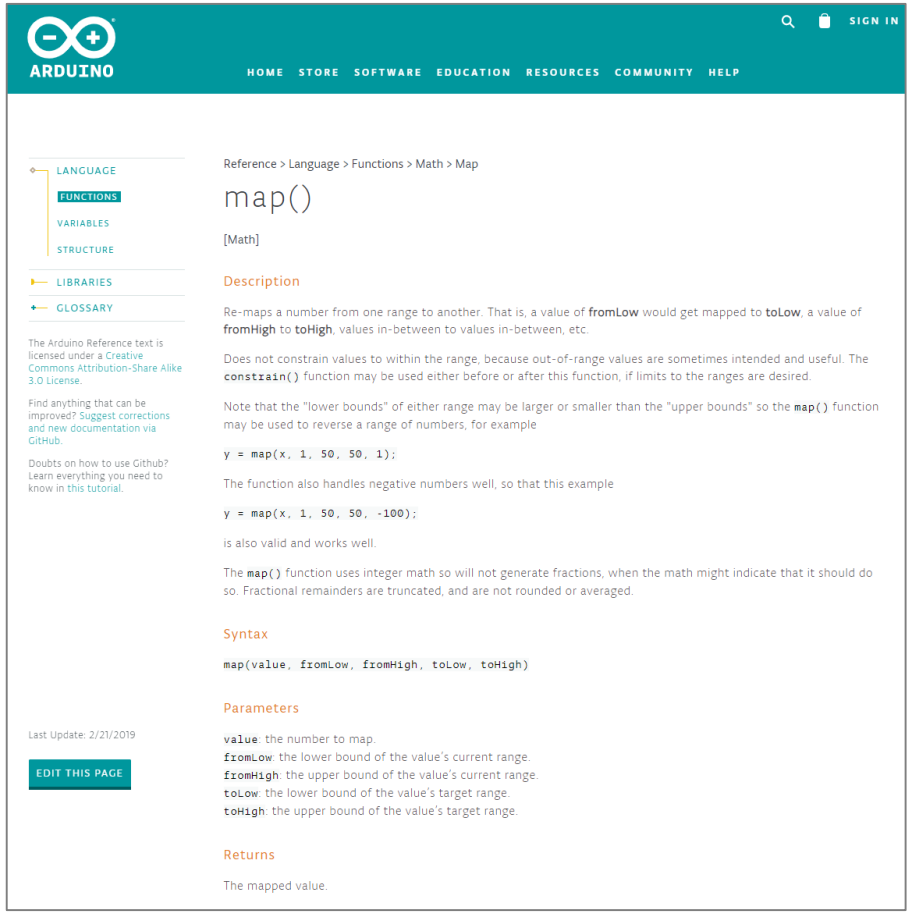
Generic form to support non-zero ranges

`analogRead`

$$Analog_{input\_fraction} = \frac{(Analog_{input\_value} - Analog_{input\_min})}{(Analog_{input\_max} - Analog_{input\_min})}$$

`analogWrite`

$$Analog_{output\_value} = Analog_{output\_min} + Analog_{input\_fraction} * (Analog_{output\_max} - Analog_{output\_min})$$



HELPER FUNCTION

`map(value, fromLow, fromHigh, toLow, toHigh)`

Remaps a number from one range to another. Warning: it does not constrain values to be within the range.

Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Parameters

`value`: the number to map

`fromLow`: the lower bound of the value's current range

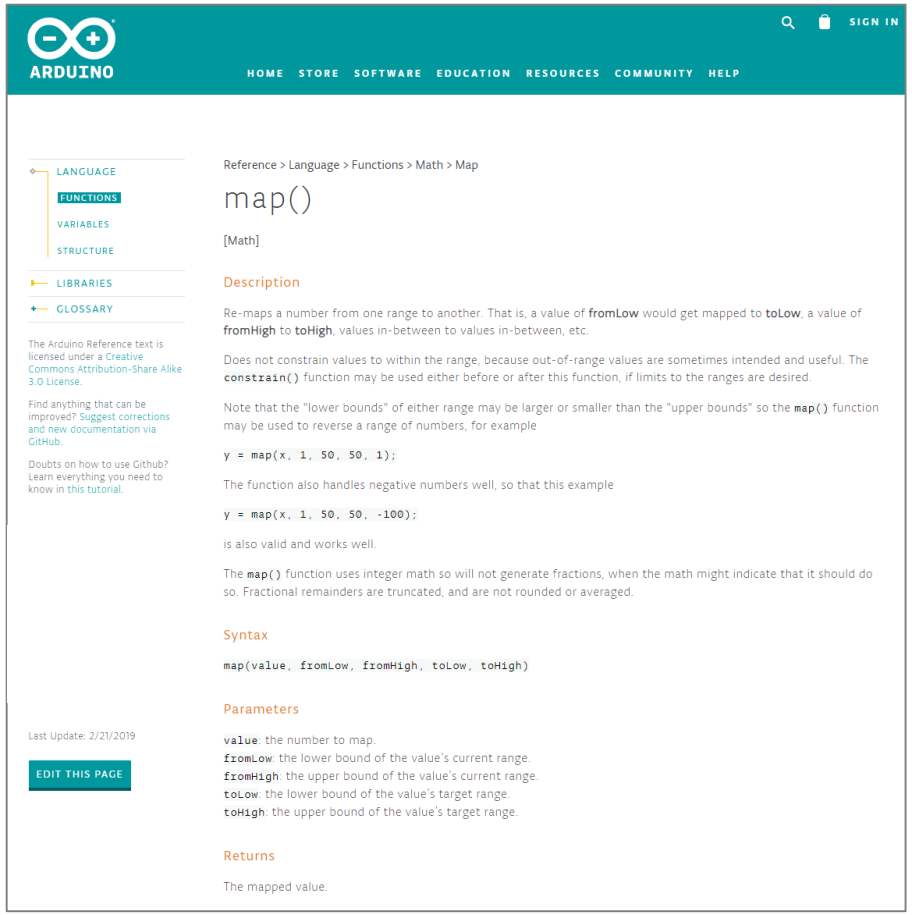
`fromHigh`: the upper bound of the value's current range

`toLow`: the lower bound of the value's target range

`toHigh`: the upper bound of the value's target range

Returns

The mapped value



https://www.arduino.cc/reference/en/language/functions/math/map/

HELPER FUNCTION

map(value, fromLow, fromHigh, toLow, toHigh)

Remaps a number from one range to another. Warning: it does not constrain values to be within the range.

Syntax

`map(value, fromLow, fromHigh, toLow, toHigh)`

Parameters

`value`: the number to map

`fromLow`: the lower bound of the value's current range

`fromHigh`: the upper bound of the value's current range

`toLow`: the lower bound of the value's target range

`toHigh`: the upper bound of the value's target range

Returns

The mapped value

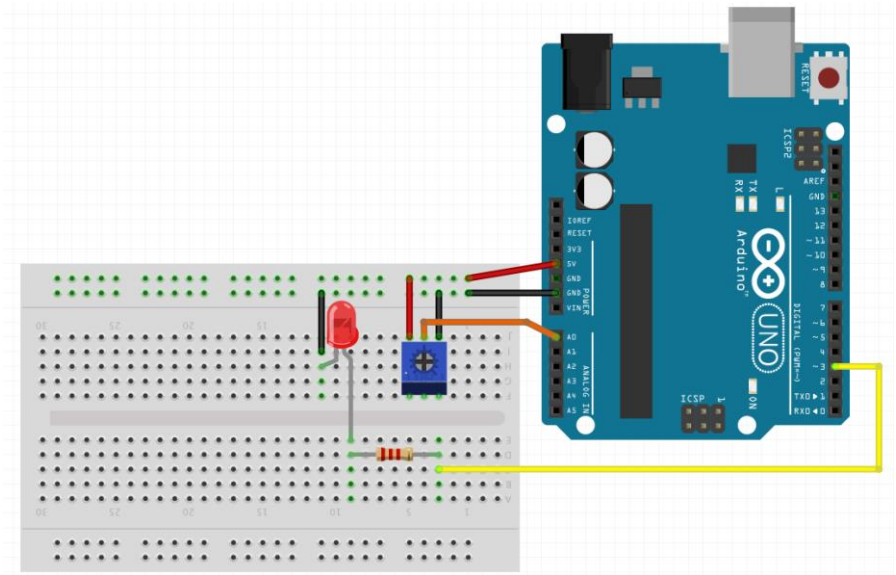
TrimpotLED

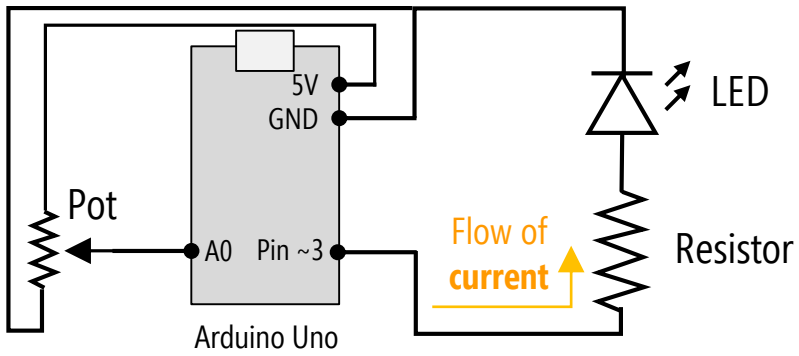
```
// The Arduino Uno ADC is 10 bits (thus, 0 - 1023 values)
#define MAX_ANALOG_INPUT_VAL 1023

const int LED_OUTPUT_PIN = 3;
const int POT_INPUT_PIN = A0;

void setup() {
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  pinMode(POT_INPUT_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {

  // read the potentiometer value
  int potVal = analogRead(POT_INPUT_PIN);

  // the analogRead on the Arduino Uno goes from 0 to 1023. We need to remap
  // this value to the smaller range (0-255) since the analogWrite function can
  // only write out 0-255 (a byte--2^8). The map function provides a linear
  // mapping to do this (however, a better way would likely be some sort of
  // non-linear mapping given that perceived LED brightness is not linear with current,
  // perhaps logarithmic)
  int ledVal = map(potVal, 0, MAX_ANALOG_INPUT_VAL, 0, 255);

  // print the raw pot value and the converted led value
  Serial.print(potVal);
  Serial.print(",");
  Serial.println(ledVal);

  // write out the LED value. This value is translated to voltage by:
  // voltageVal = max_voltage * val/255 or voltageVal = 3.3V * val/255 in
  // the case of the RedBear Duo
  analogWrite(LED_OUTPUT_PIN, ledVal);

  delay(100);
}
```

https://www.arduino.cc/reference/en/language/functions/math/map/

CONTROL LED BRIGHTNESS WITH POTENTIOMETER

Circuit: Fade LED on/off via pot value





Code: Fade LED on/off via pot value

TrimpotLED

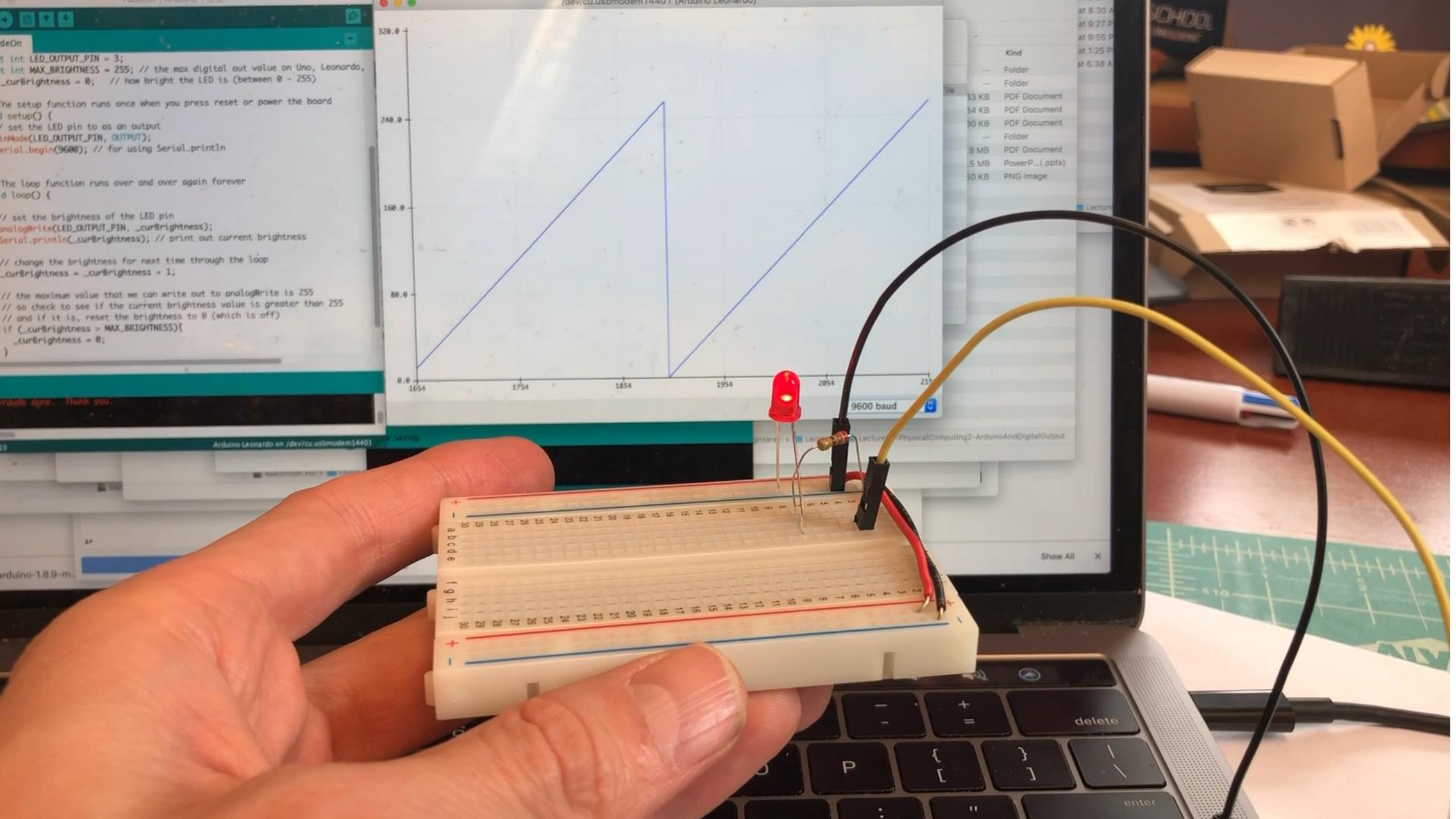
```
// The Arduino Uno ADC is 10 bits (thus, 0 - 1023 values)
#define MAX_ANALOG_INPUT_VAL 1023

const int LED_OUTPUT_PIN = 3;
const int POT_INPUT_PIN = A0;

void setup() {
  pinMode(LED_OUTPUT_PIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {

  // read the potentiometer value
  int potVal = analogRead(POT_INPUT_PIN);

  // the analogRead on the Arduino Uno goes from 0 to 1023. We need to remap
  // this value to the smaller range (0-255) since the analogWrite function can
  // only write out 0-255 (a byte--2^8). The map function provides a linear
  // mapping to do this (however, a better way would likely be some sort of
  // non-linear mapping given that perceived LED brightness is not linear with current,
  // perhaps logarithmic)
  int ledVal = map(potVal, 0, MAX_ANALOG_INPUT_VAL, 0, 255);

  // print the raw pot value and the converted led value
  Serial.print(potVal);
  Serial.print(",");
  Serial.println(ledVal);

  // write out the LED value. This value is translated to voltage by:
  // voltageVal = max_voltage * val/255 or voltageVal = 3.3V * val/255 in
  // the case of the RedBear Duo
  analogWrite(LED_OUTPUT_PIN, ledVal);

  delay(100);
}
```

DEBUGGING ARDUINO

Modularize.

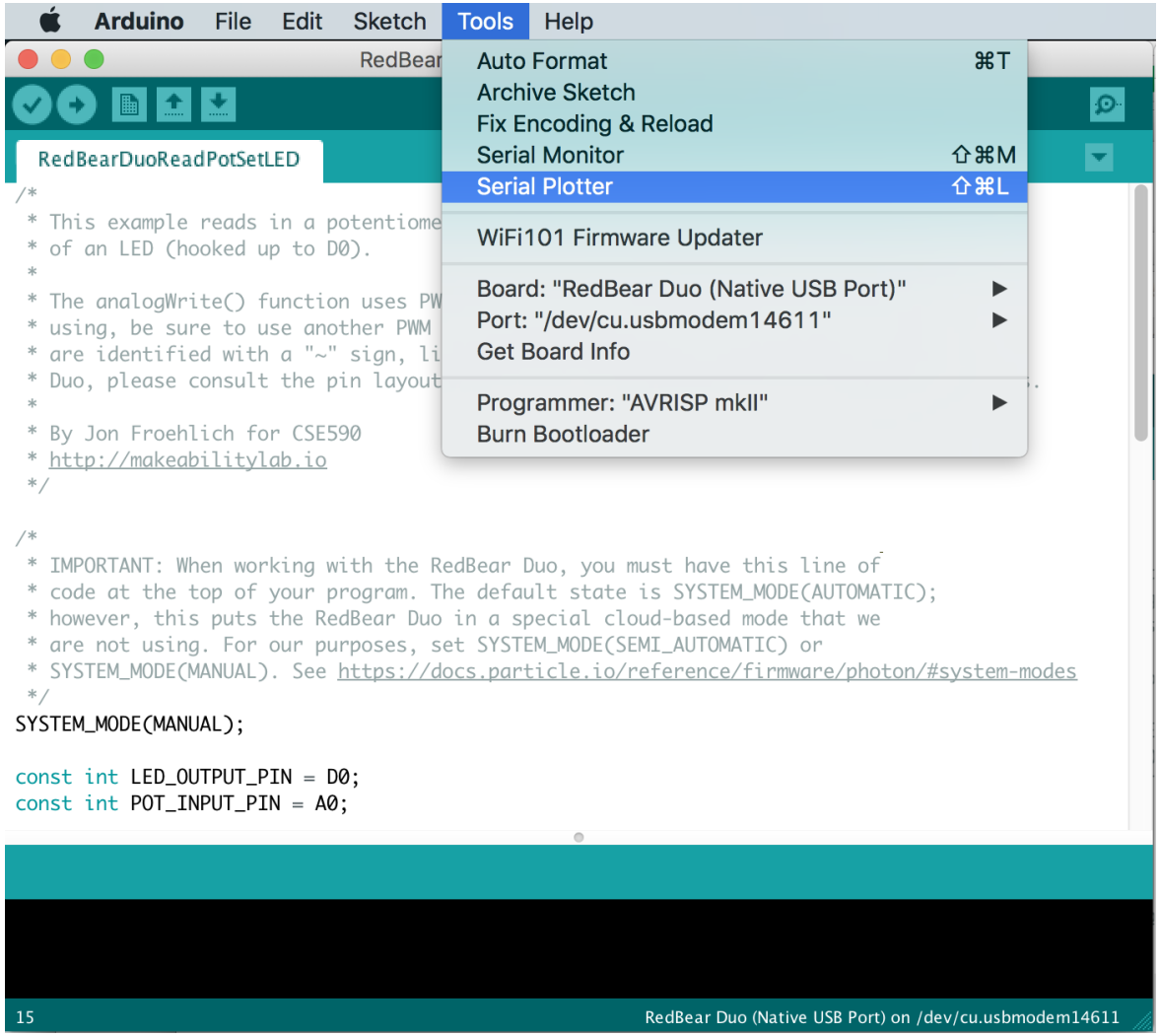
Use Serial.println -> Serial Monitor and Serial Plotter

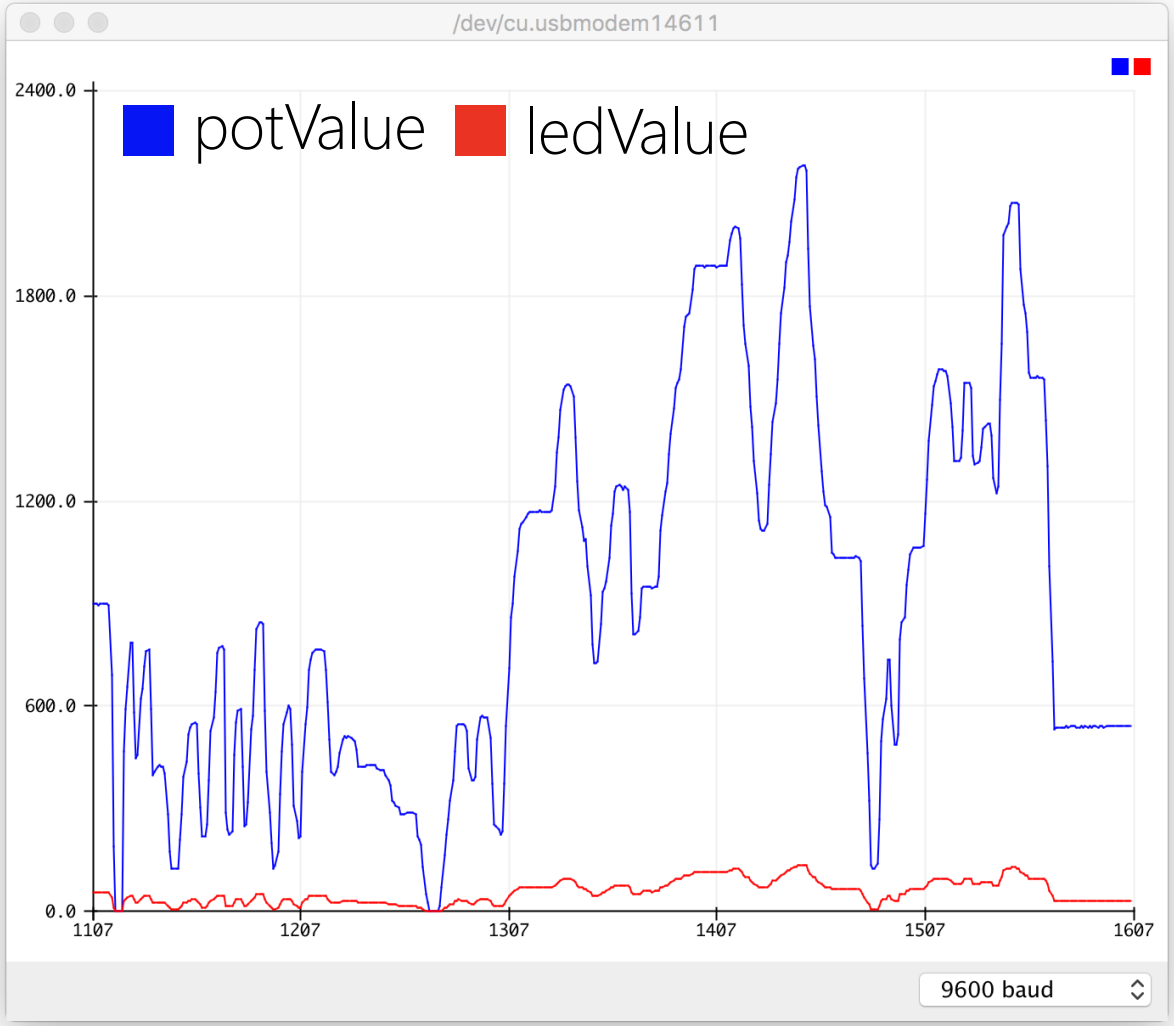
Use a multimeter


int LED_OUTPUT_PIN = 3;
int MAX_BRIGHTNESS = 255; // the max digital out value on Uno, Leonardo,
_curBrightness = 0;   // how bright the LED is (between 0 - 255)
The setup function runs once when you press reset or power the board
setup() {
set the LED pin to as an output
inMode(LED_OUTPUT_PIN, OUTPUT);
erial.begin(9600); // for using Serial.println
The loop function runs over and over again forever
loop() {
// set the brightness of the LED pin
analogWrite(LED_OUTPUT_PIN, _curBrightness);
Serial.println(_curBrightness); // print out current brightness
// change the brightness for next time through the loop
_curBrightness = _curBrightness + 1;
// the maximum value that we can write out to analogWrite is 255
// so check to see if the current brightness value is greater than 255
// and if it is, reset the brightness to 0 (which is off)
if (_curBrightness > MAX_BRIGHTNESS){
_curBrightness = 0;
}
320.0
240.0
160.0
80.0
0.0
1654
1754
1854
1954
9600 baud
Kind
Folder
PDF Document
PNG image
Show All
delete
enter

SERIAL PLOTTER

USING THE SERIAL PLOTTER



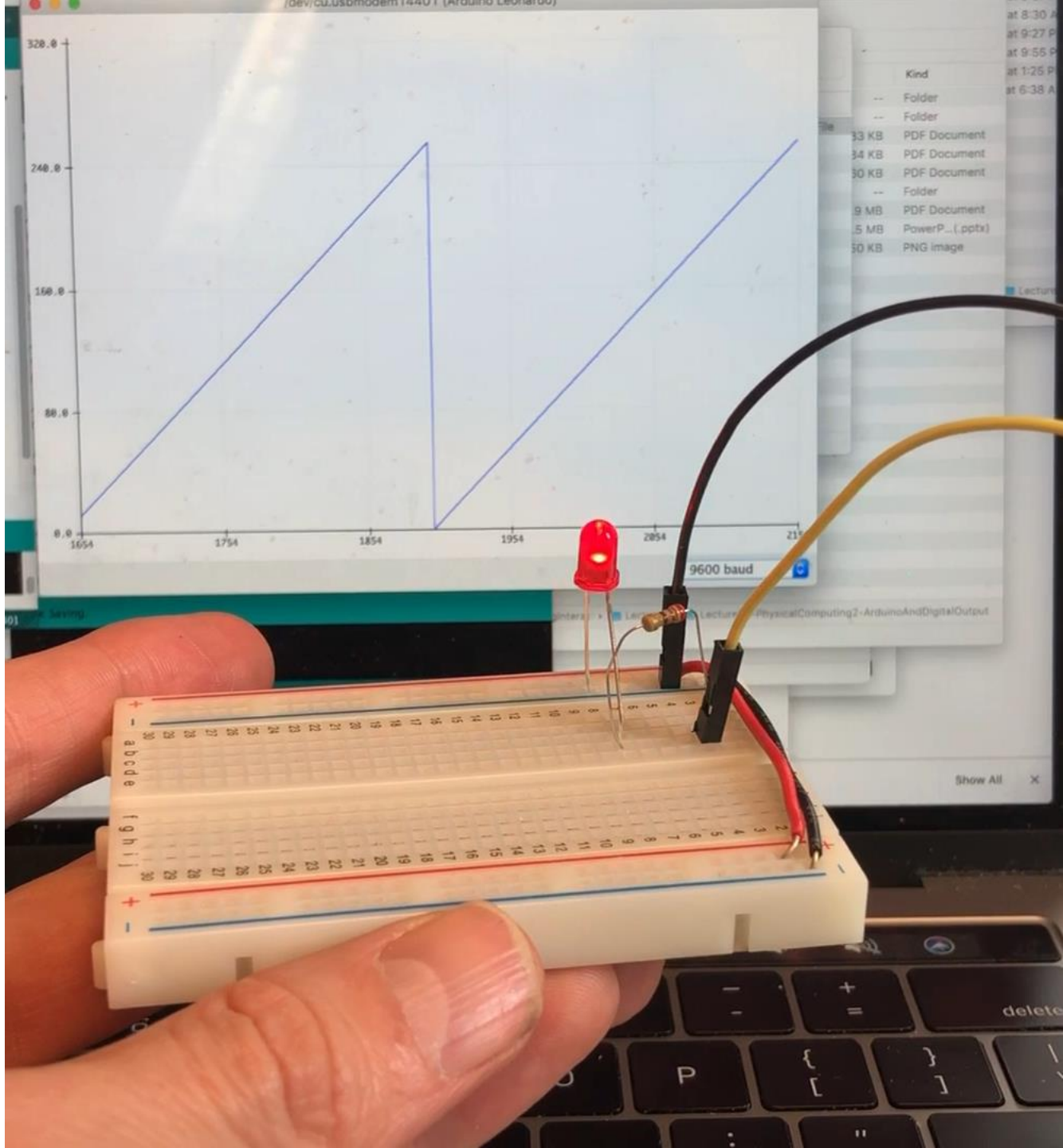


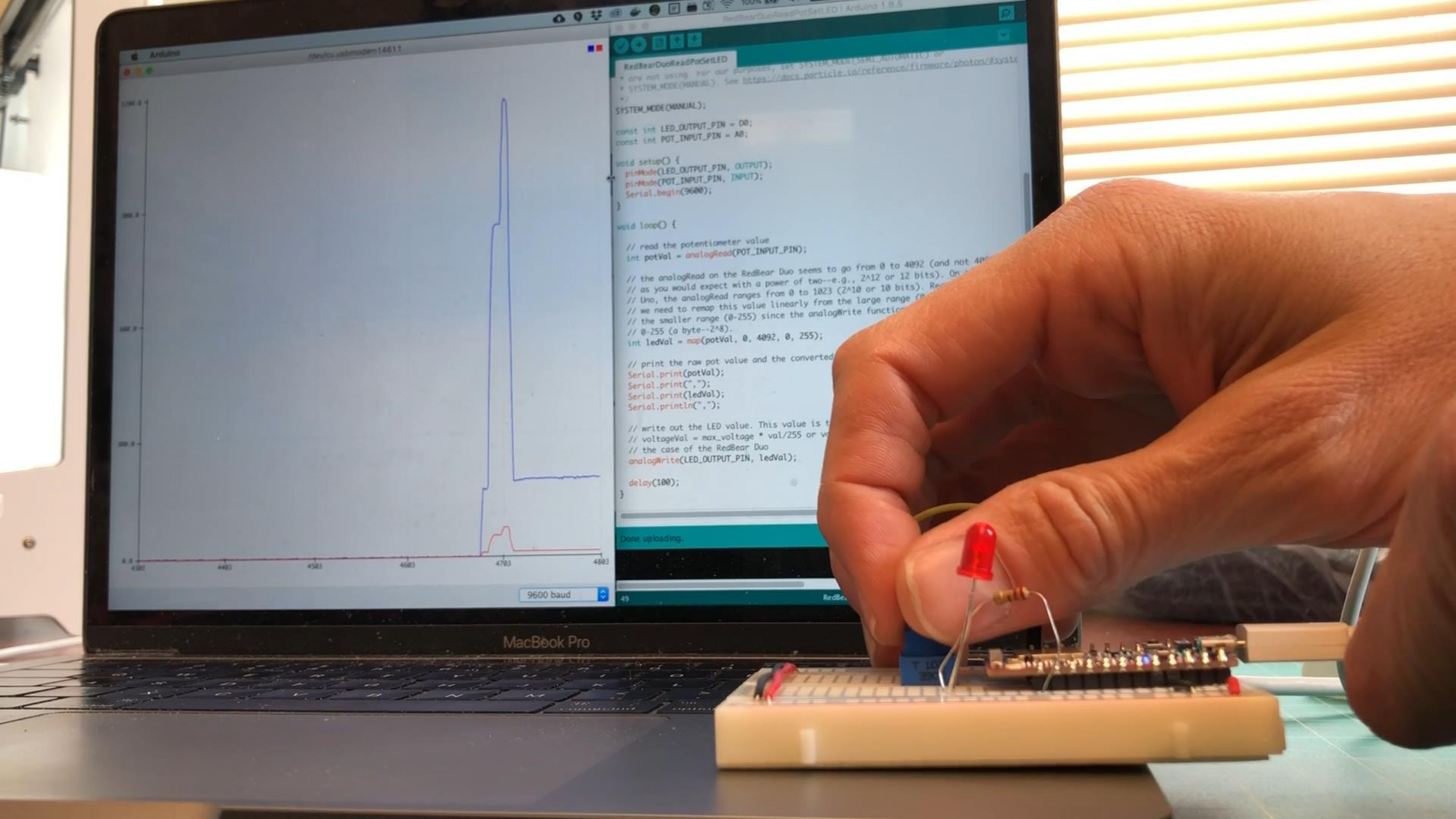
DEBUGGING

SERIAL PLOTTER

Visualizes numeric values off of Serial port

To **visualize multiple values**, use a **CSV** (comma separate value) format

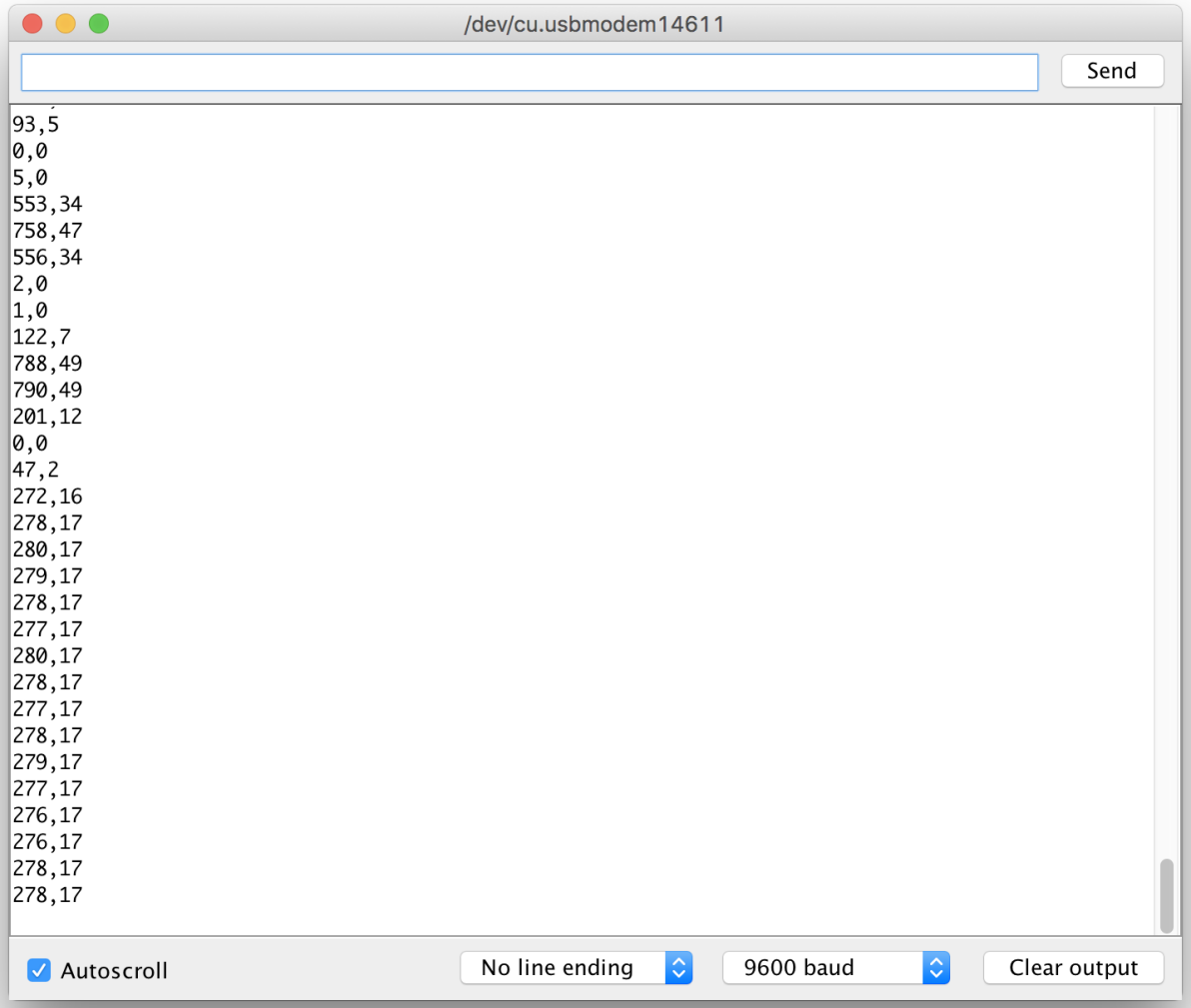


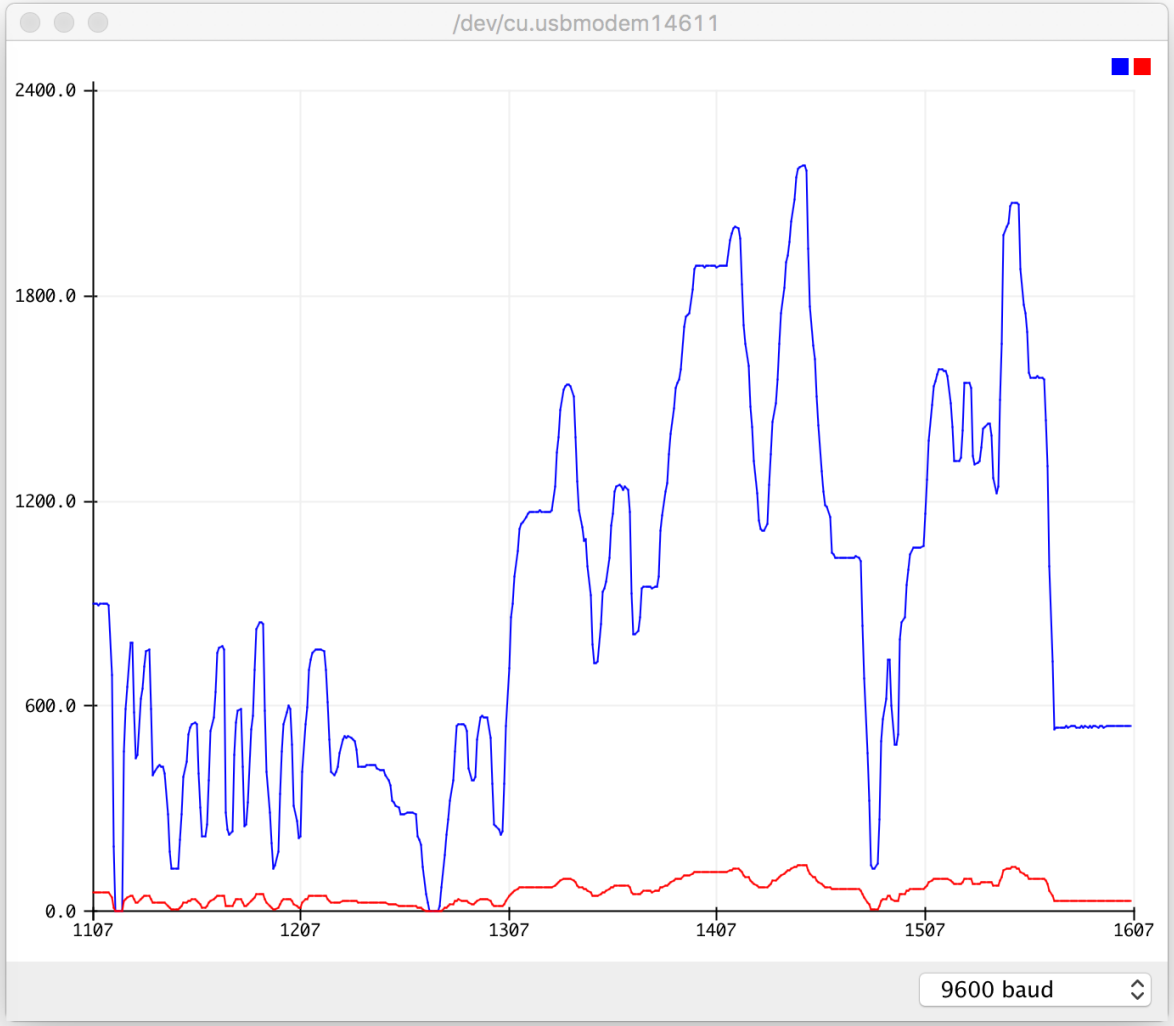

Arduino
/dev/cu.usbmodem14611
9600 baud
RedBearDuoReadPotSetLED
SYSTEM_MODE(MANUAL);
const int LED_OUTPUT_PIN = D0;
const int POT_INPUT_PIN = A0;
void setup() {
pinMode(LED_OUTPUT_PIN, OUTPUT);
pinMode(POT_INPUT_PIN, INPUT);
Serial.begin(9600);
}
void loop() {
// read the potentiometer value
int potVal = analogRead(POT_INPUT_PIN);
// print the raw pot value and the converted
Serial.print(potVal);
Serial.print(",");
Serial.print(ledVal);
Serial.println(",");
// write out the LED value. This value is t
// the case of the RedBear Duo
analogWrite(LED_OUTPUT_PIN, ledVal);
delay(100);
}
Done uploading.
MacBook Pro

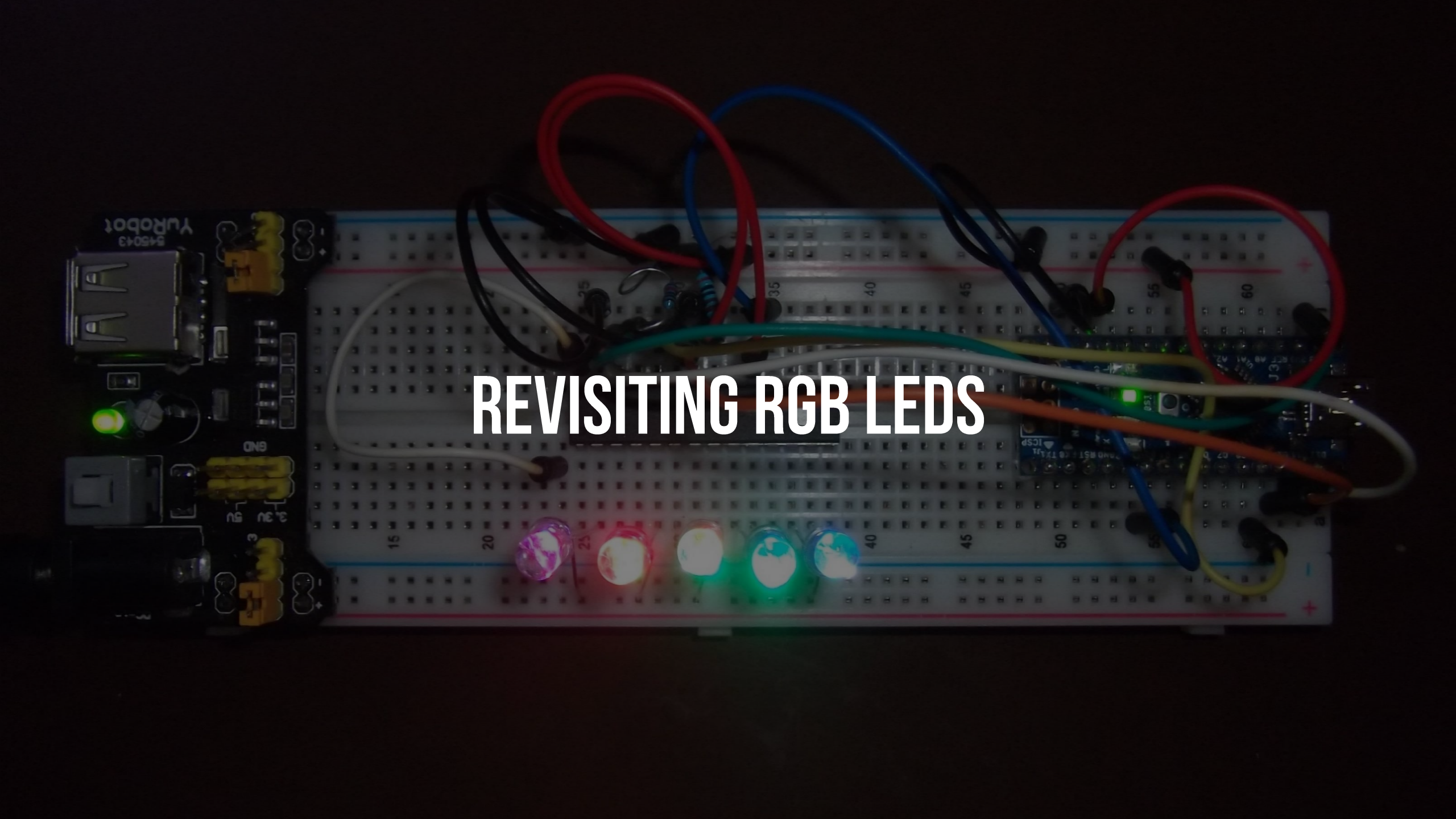
SERIAL PLOTTER

SERIAL MONITOR VS. PLOTTER

You can only have one of these windows open at once.






REVISITING RGB LEDS

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

1 ADD TO CART

QTY	DISCOUNT
1-9	$2.00
10-49	$1.75
50+	$1.50

ADD TO WISHLIST

adafruit SHOP BLOG LEARN FORUMS VIDEOS ADABOX

LEDS / BARE LEDS / DIFFUSED RGB (TRI-COLOR) LED

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

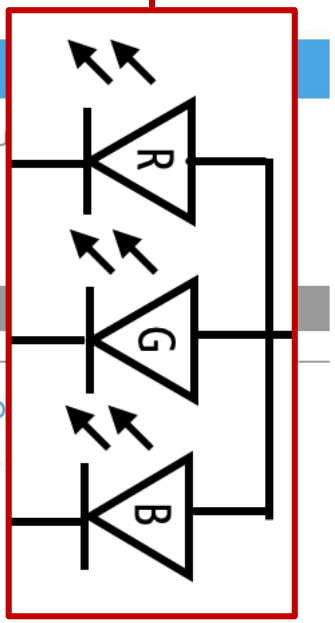
$2.00

49 IN STOCK

1

QTY DISCOU
1-9 $2.00
10-49 $1.75
50+ $1.50

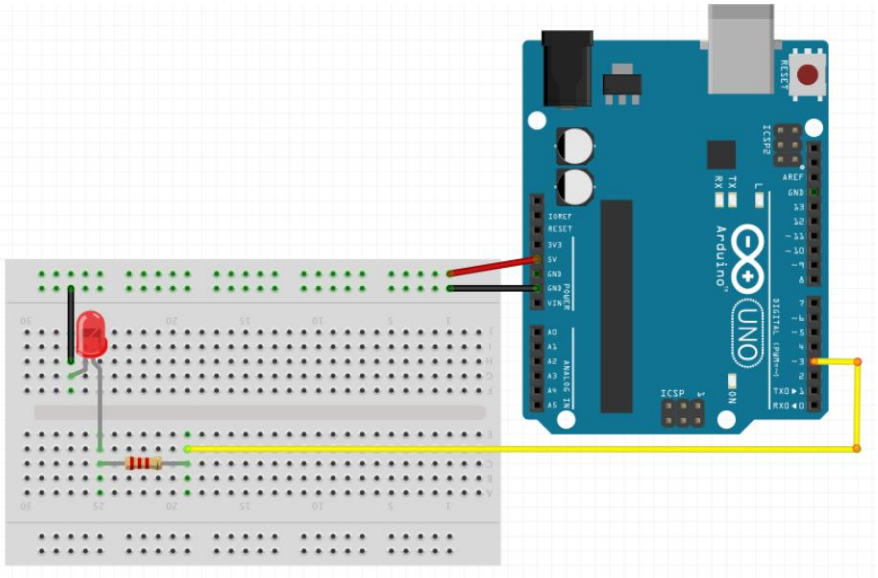
DESCRIPTIO
TECHNICAL
LEARN

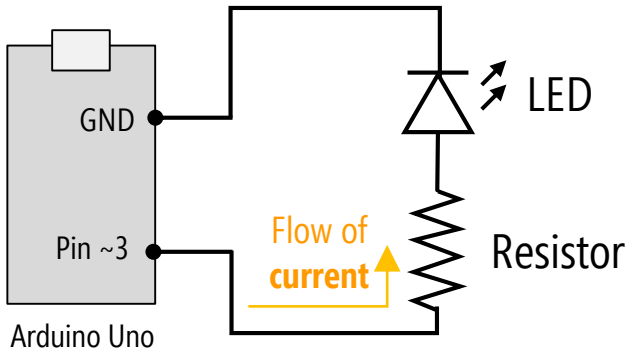


ANALOG OUTPUT

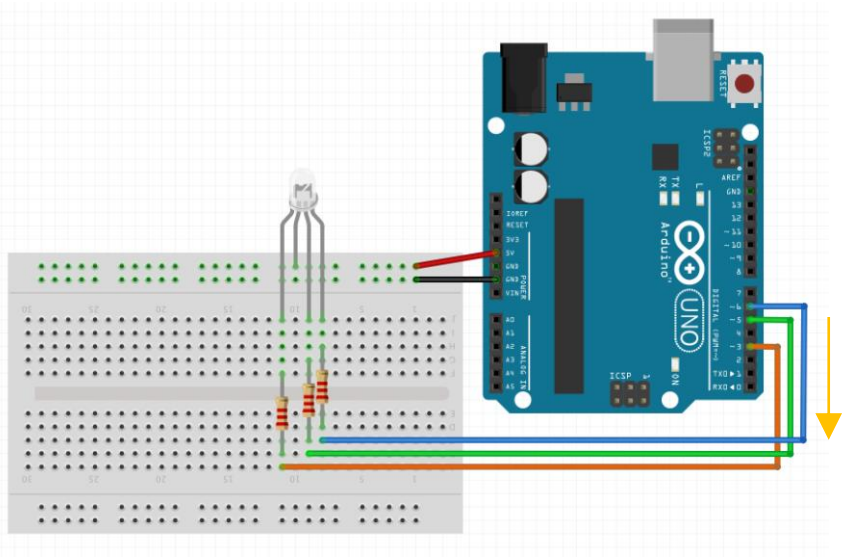
USING AN RGB LED: THE CIRCUIT

Old Circuit: Fade LED on/off via pin 3

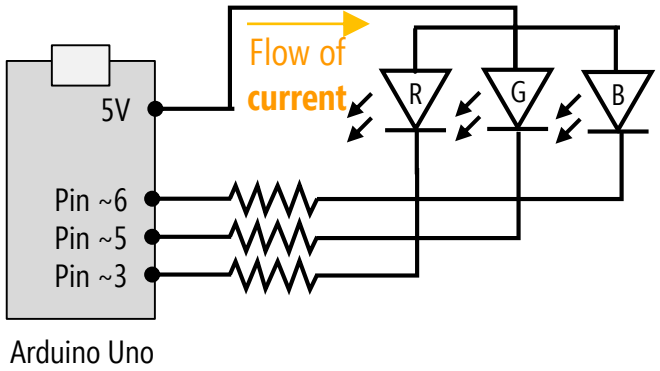




New Circuit: Set R, B, G values via pin 6, 5, and 3

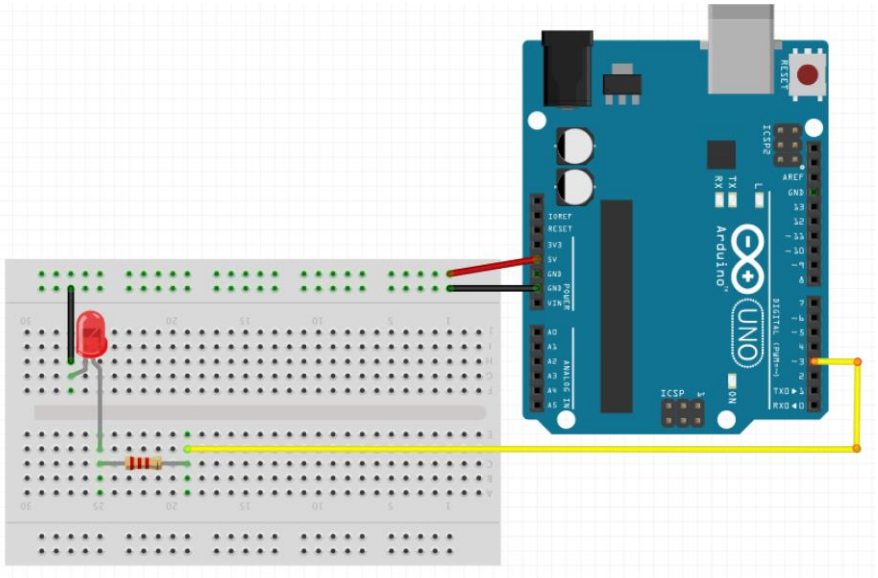


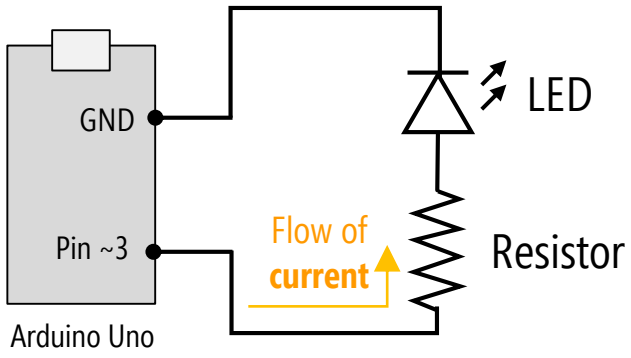
Flow of current
Because this is a "common anode" RGB. We set these pins to zero to turn on the light and 5V to turn off the light!



USING AN RGB LED: THE CIRCUIT

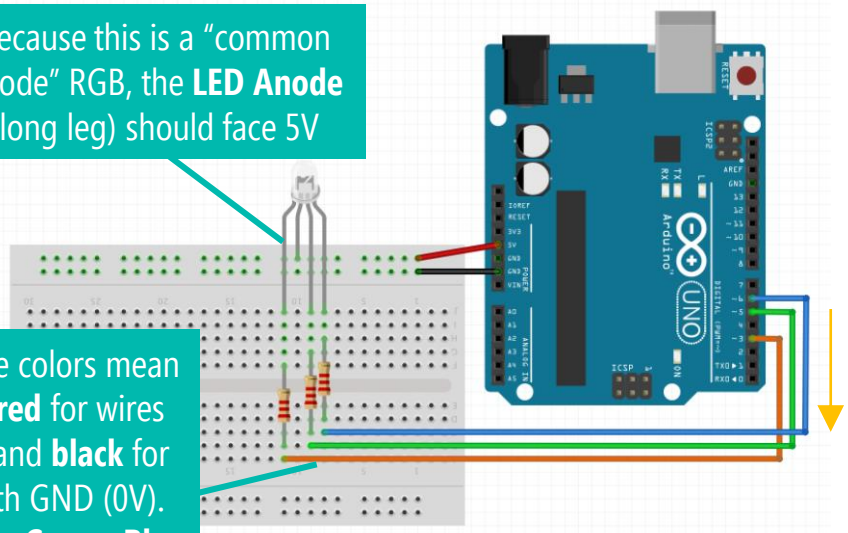
Old Circuit: Fade LED on/off via pin 3





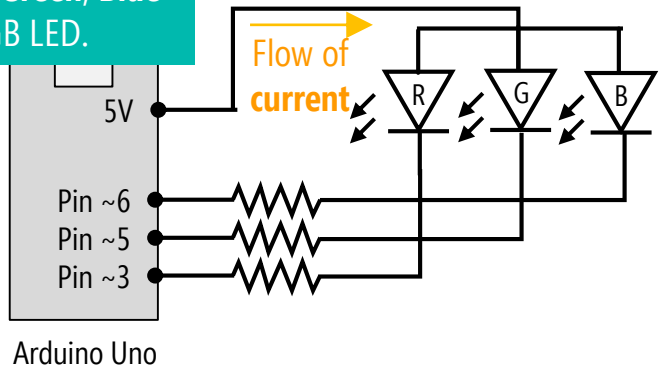
New Circuit: Set R, B, G values via pin 6, 5, and 3

Because this is a "common anode" RGB, the **LED Anode** (long leg) should face 5V



Pro tip: make your wire colors mean something. Only use **red** for wires that connect with 5V and **black** for wires that connect with GND (0V). Here, I'm using **Orange**, **Green**, **Blue** to represent the **Red**, **Green**, **Blue** signal for my RGB LED.

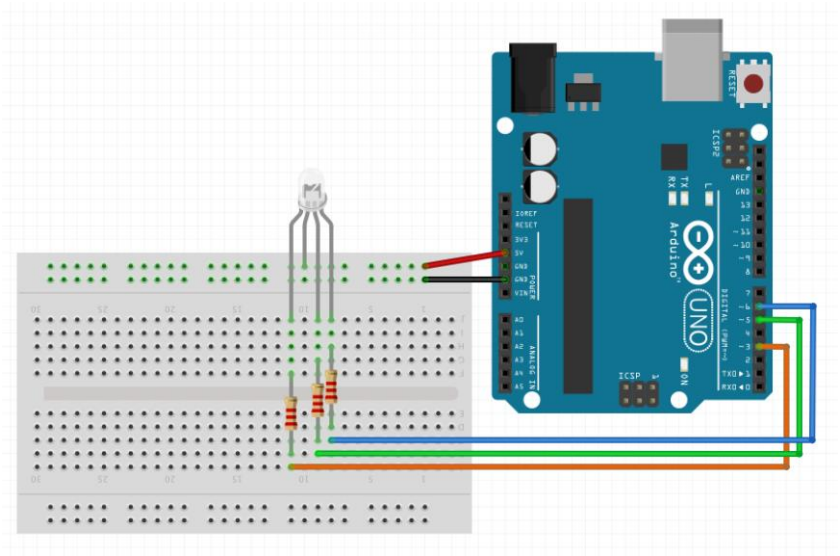
Flow of current
Because this is a "common anode" RGB. We set these pins to zero to turn on the light and 5V to turn off the light!

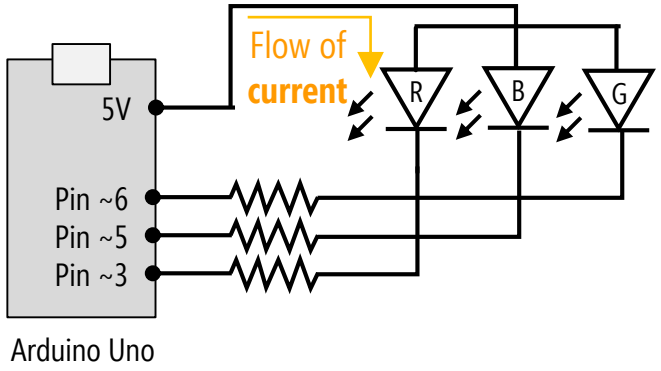


ANALOG OUTPUT

USING AN RGB LED: THE CODE

Circuit: Set R, B, G values via pin 6, 5, and 3

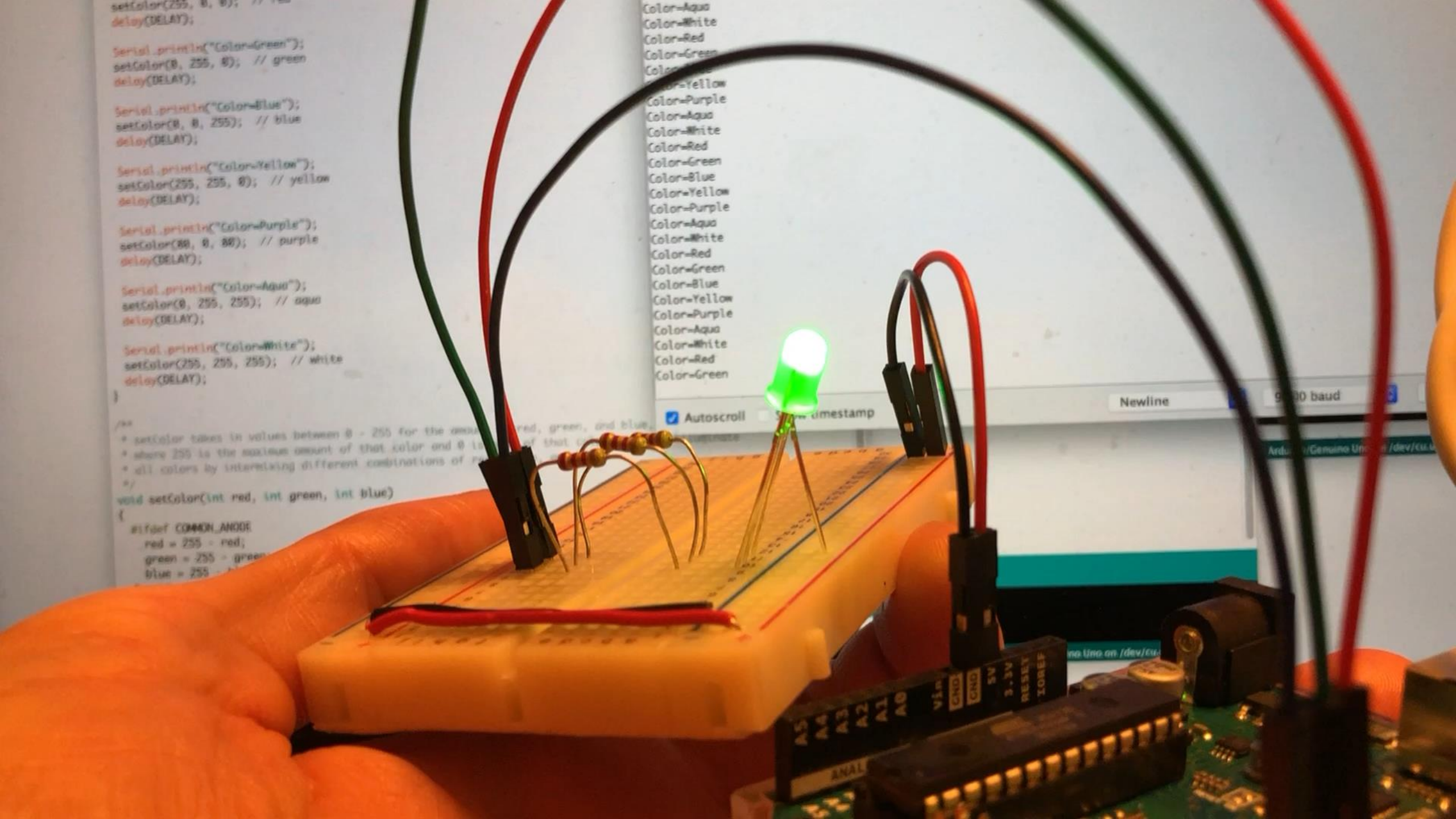




Arduino Uno

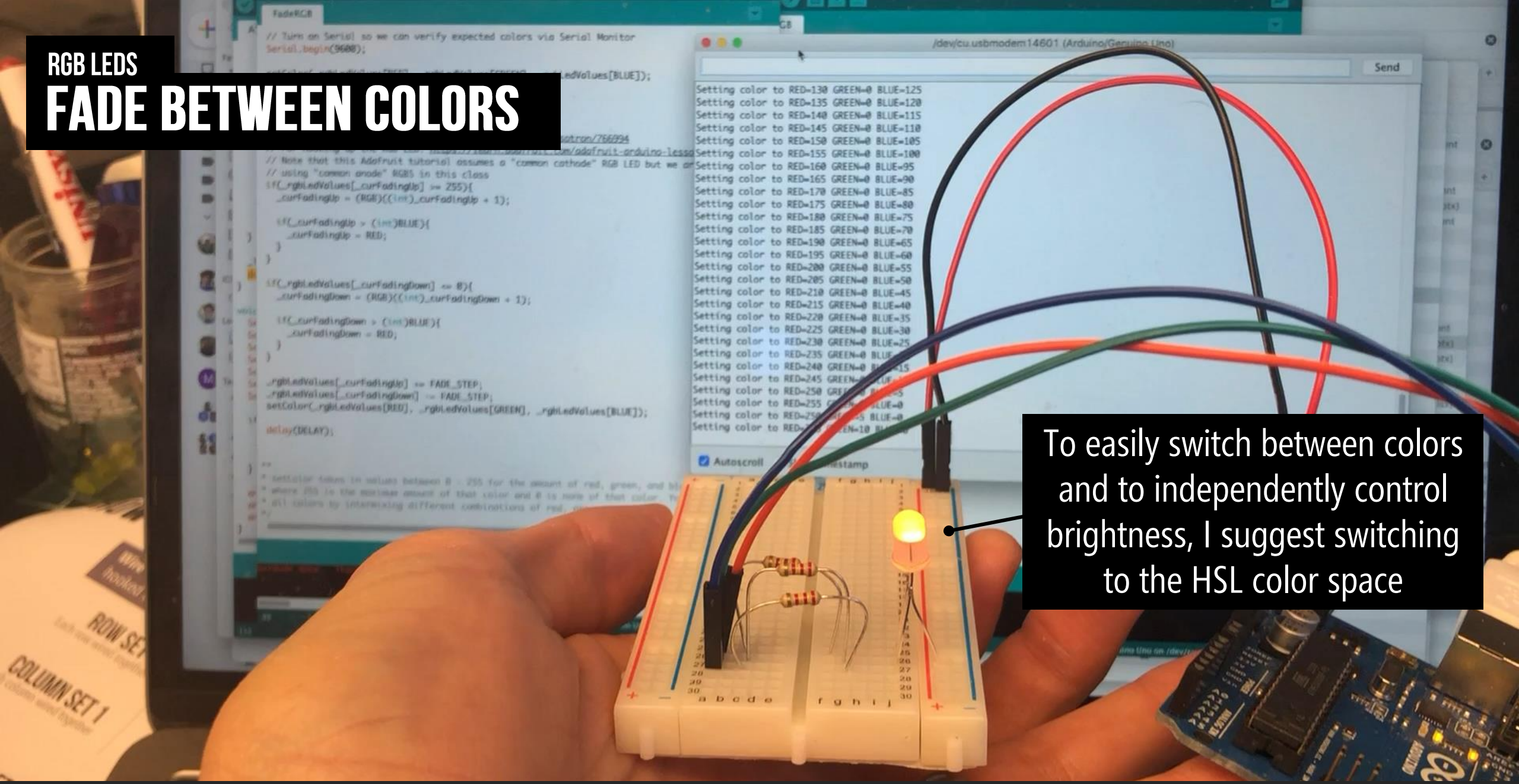
BlinkRGBSimple

```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

void setup() {
  // Set the RGB pins to output
  pinMode(RGB_RED_PIN, OUTPUT);
  pinMode(RGB_GREEN_PIN, OUTPUT);
  pinMode(RGB_BLUE_PIN, OUTPUT);
}

void loop() {

  // Because the RGB LED we purchased for the class is a 'common anode'
  // RGB, the way we turn on each light is counter to our intuition
  // We set a pin to 0 to turn on the corresponding LED to its maximum brightness
  // and 255 to turn it off (the opposite of what you might think)

  // Set the RGB LED to red
  analogWrite(RGB_RED_PIN, 0);      // turn on the red LED
  analogWrite(RGB_GREEN_PIN, 255);  // turn off the green LED
  analogWrite(RGB_BLUE_PIN, 255);   // turn off the blue LED
  delay(DELAY);

  // Set the RGB LED to green
  analogWrite(RGB_RED_PIN, 255);    // turn off the red LED
  analogWrite(RGB_GREEN_PIN, 0);    // turn on the green LED
  analogWrite(RGB_BLUE_PIN, 255);     // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to blue
  analogWrite(RGB_RED_PIN, 255);    // turn off the red LED
  analogWrite(RGB_GREEN_PIN, 255);  // turn off the green LED
  analogWrite(RGB_BLUE_PIN, 0);     // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to yellow (by turning on green and blue!)
  analogWrite(RGB_RED_PIN, 255);    // turn off the red LED
  analogWrite(RGB_GREEN_PIN, 0);    // turn on the green LED
  analogWrite(RGB_BLUE_PIN, 0);     // turn on the blue LED
  delay(DELAY);

  // Set the RGB LED to purple (by turning on red and blue!)
  analogWrite(RGB_RED_PIN, 0);      // turn on the red LED
  analogWrite(RGB_GREEN_PIN, 255);  // turn off the green LED
  analogWrite(RGB_BLUE_PIN, 0);     // turn on the blue LED
  delay(DELAY);
}
```


Color=Aqua
Color=White
Color=Red
Color=Green
Color=Yellow
Color=Purple
Color=Aqua
Color=White
Color=Red
Color=Green
Color=Blue
Color=Yellow
Color=Purple
Color=Aqua
Color=White
Color=Red
Color=Green
Color=Blue
Color=Yellow
Color=Purple
Color=Aqua
Color=White
Color=Red
Color=Green
Autoscroll
Newline
baud
A0
A1
A2
A3
A4
A5
VIN
GND
GND
5V
3.3V
RESET
IOREF
ANAL

RGB LEDS

FADE BETWEEN COLORS



Video: Jon Froehlich

PHYSICAL COMPUTING 3

TODAY'S LEARNING GOALS

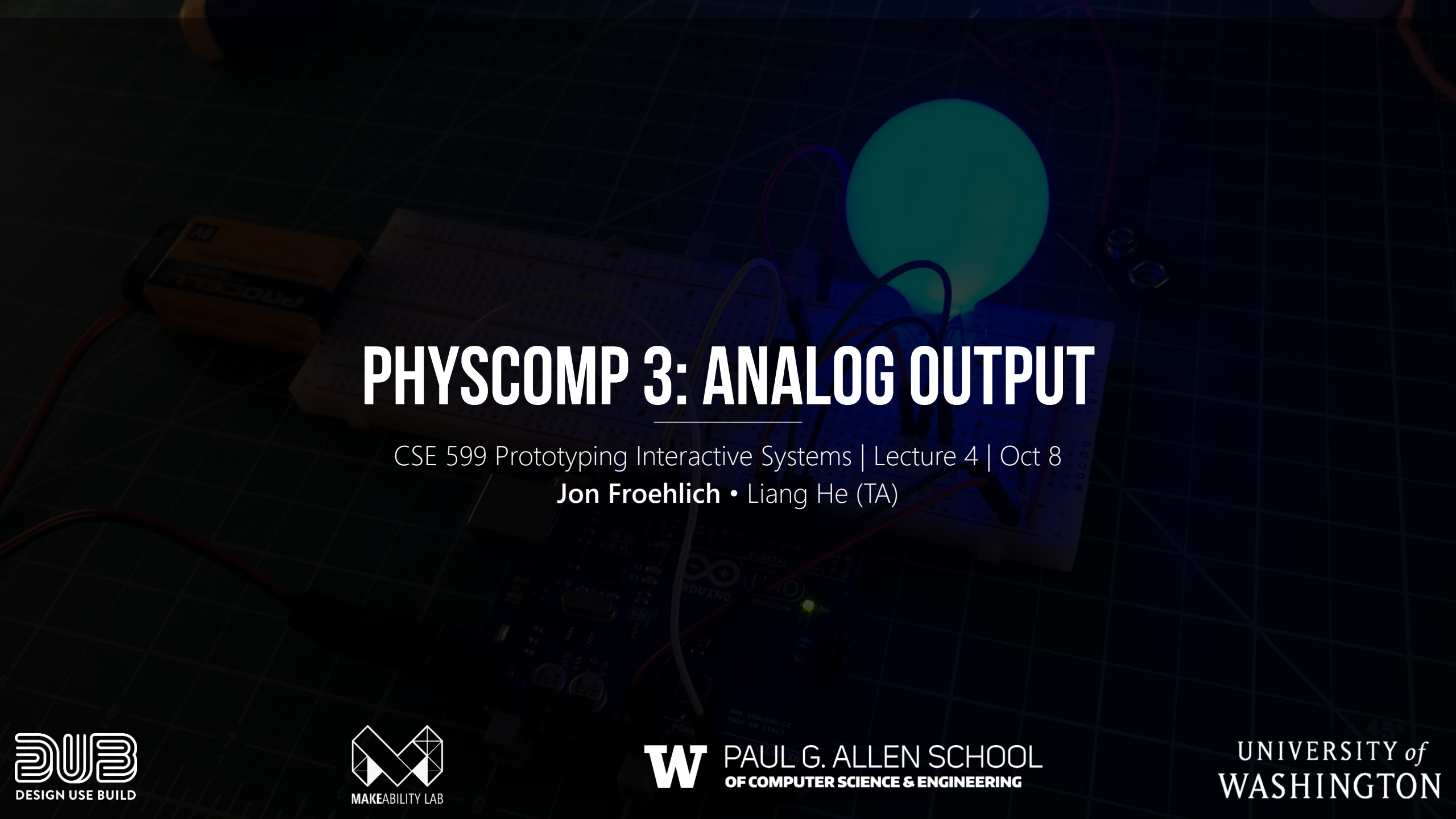
What is **analog output?**

What is **PWM** and why does it matter?

How to use **analogWrite()**

How to use **RGB Leds**

(Partial) introduction to **potentiometers** and **analog input** (if time)



PHYSCOMP 3: ANALOG OUTPUT

CSE 599 Prototyping Interactive Systems | Lecture 4 | Oct 8

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON