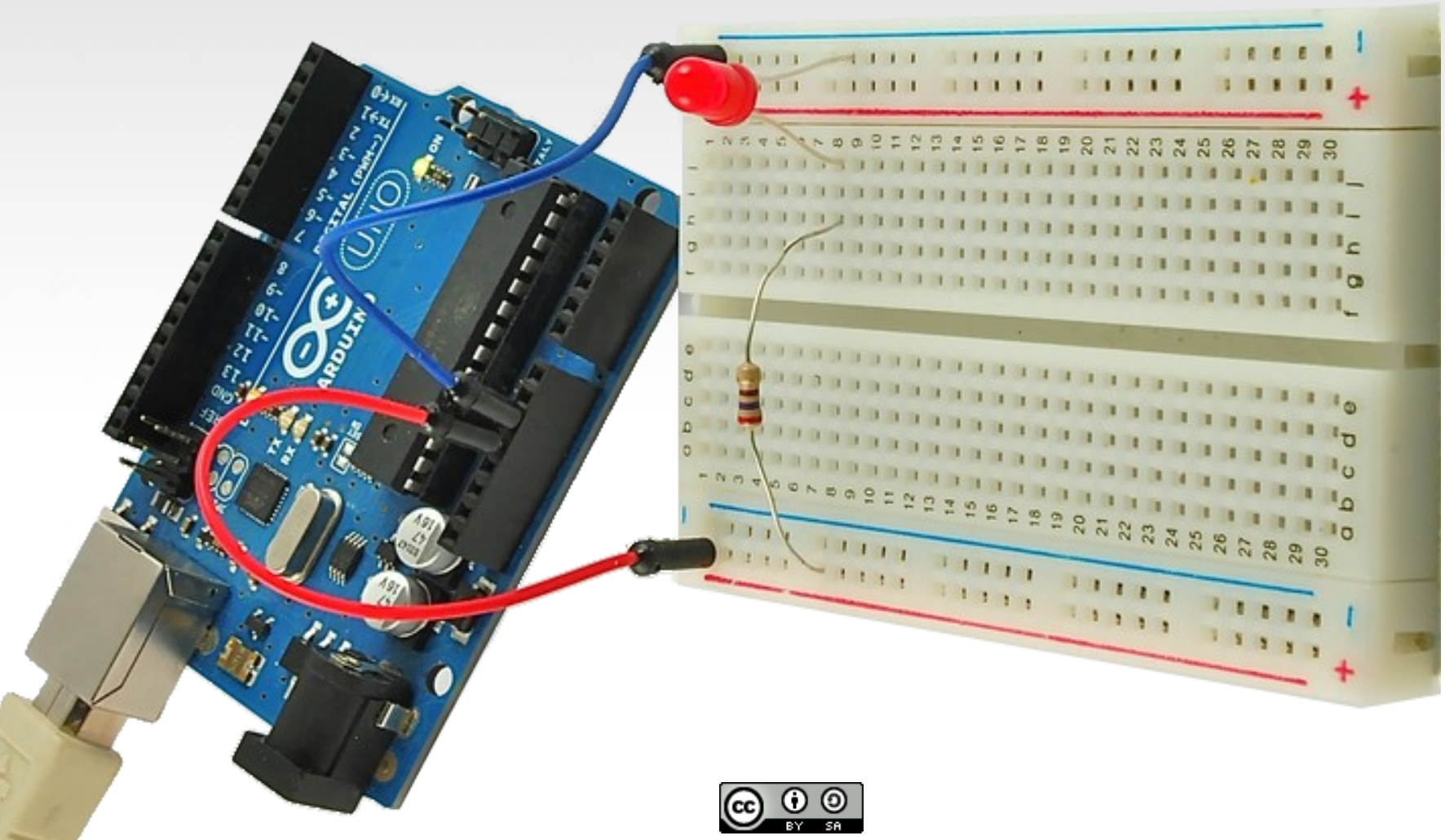
In The Blink Of An LED:

An introduction to physical computing



Overview of Class

Getting Started:

- Context
- Components
- Software Installation

Electrical:

- Ohm's Law
- Circuits
- Multimeters
- Inputs and Outputs
- Analog vs Digital

Microcontrollers:

- Digital Outputs
- Analog Outputs
- Digital Inputs
- Analog Inputs
- Serial Communication

Wiring Library, Processing (and Arduino)

Open Source Hardware/Software

Processor

Coding is accessible & transferrable

(C++, Processing, java)

Quick Definitions

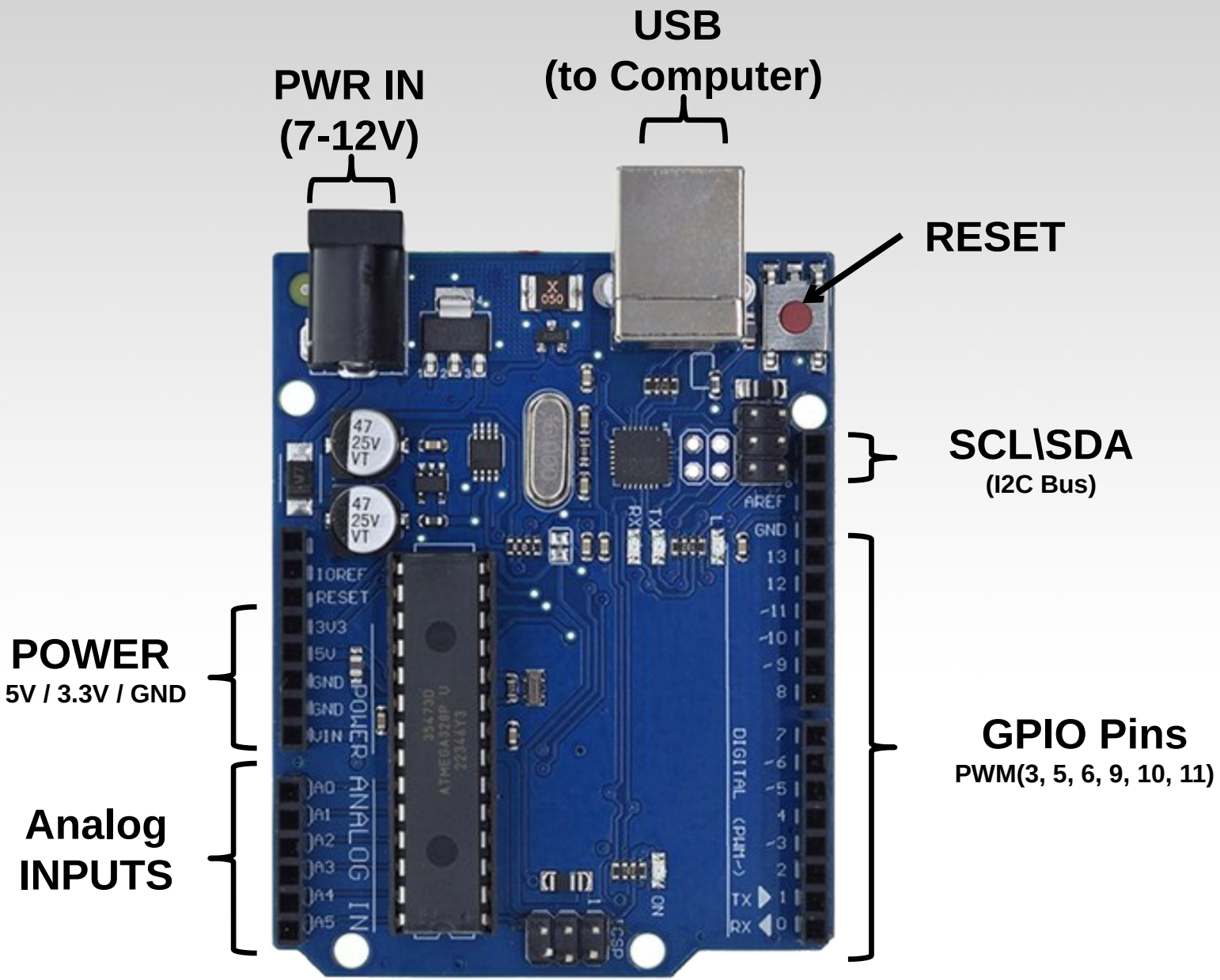
Arduino – can refer to the Amtel ATMEGA328p microcontroller, the development board or th programming environment

microcontroller – shitty computer, but generally one with GPIO and analog input pins

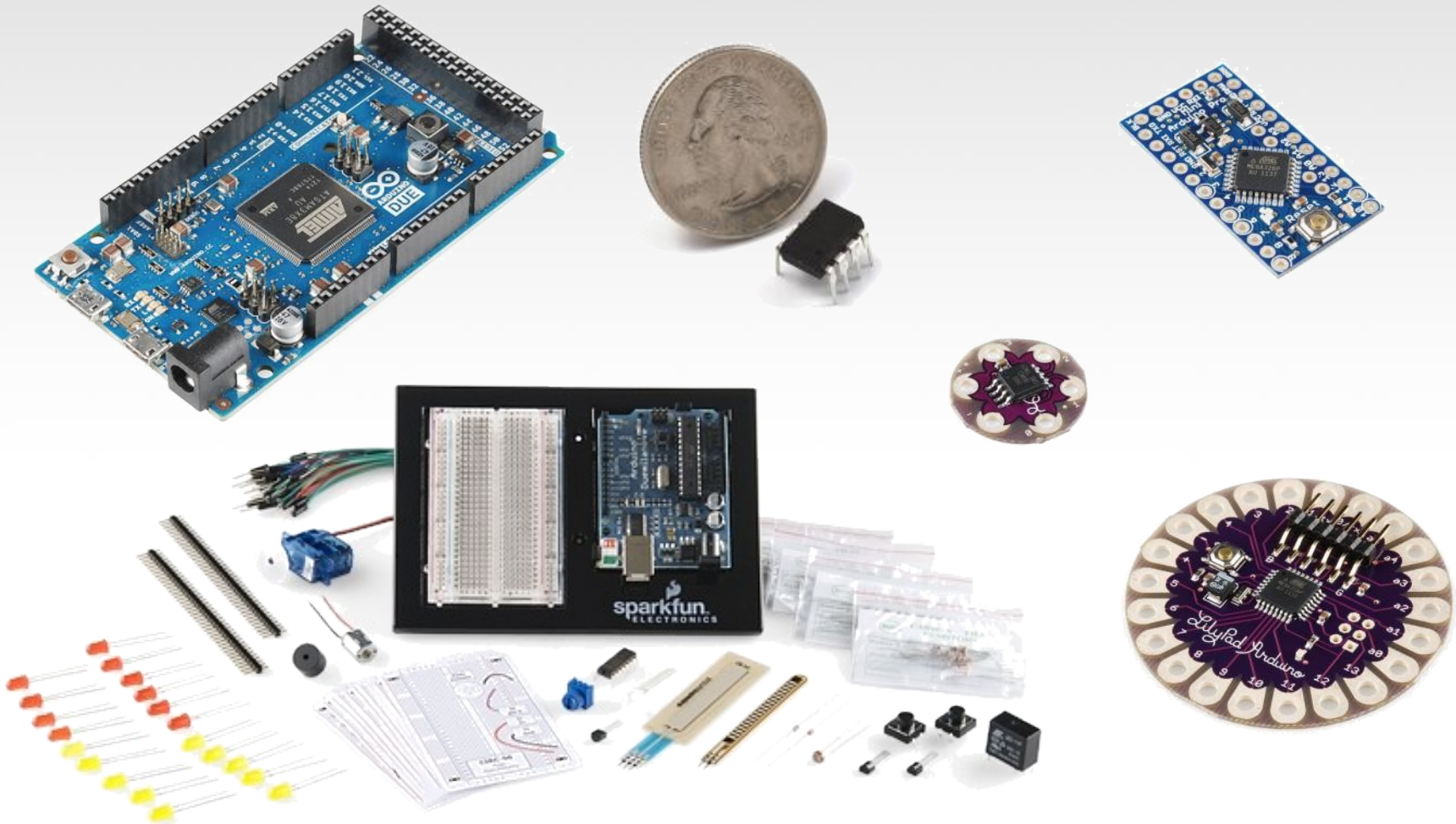
GPIO pins – general purpose input/output pins, send and receive signals from outside the chip

physical computing – using a computer to interact with the real world

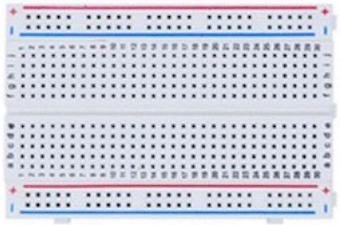
development board – microcontroller with connectors to easily access the GPIO, analog input and power



Arduino & Arduino Compatible Boards



Included Components

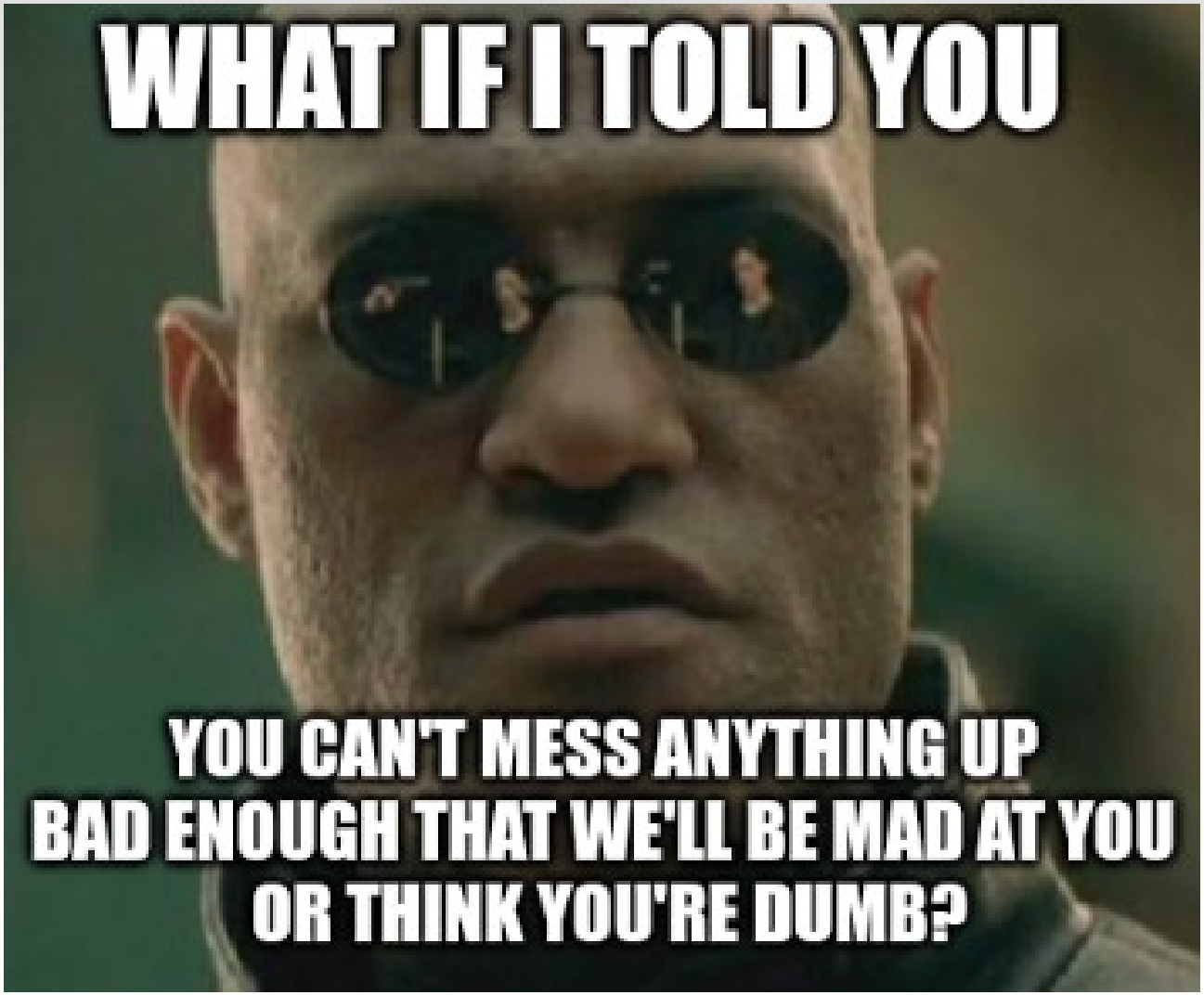
Name	Image	Type	Function	Notes
Push Button		Digital Input	Switch - Closes or opens circuit	
Trim potentiometer		Analog Input	Variable resistor	Also called a Trimpot.
Resistor		Passive Component	Drops Voltage	Color-Coded for Different Values
Photoresistor		Analog Input	Light Dependent Resistor (LDR)	Resistance varies with light.
LED		Digital & Analog Output	Emits a single wavelength light	
RGB LED		Digital & Analog Output	16,777,216 different colors	Ooh... So pretty.
Jumper Wires		Interconnect	Connects other components	Your extroverted friend
Breadboard		Interconnect	Where the connections happen	The neighborhood pub

Getting Started

https://microcontrollers.smartypantsconsulting.ltd

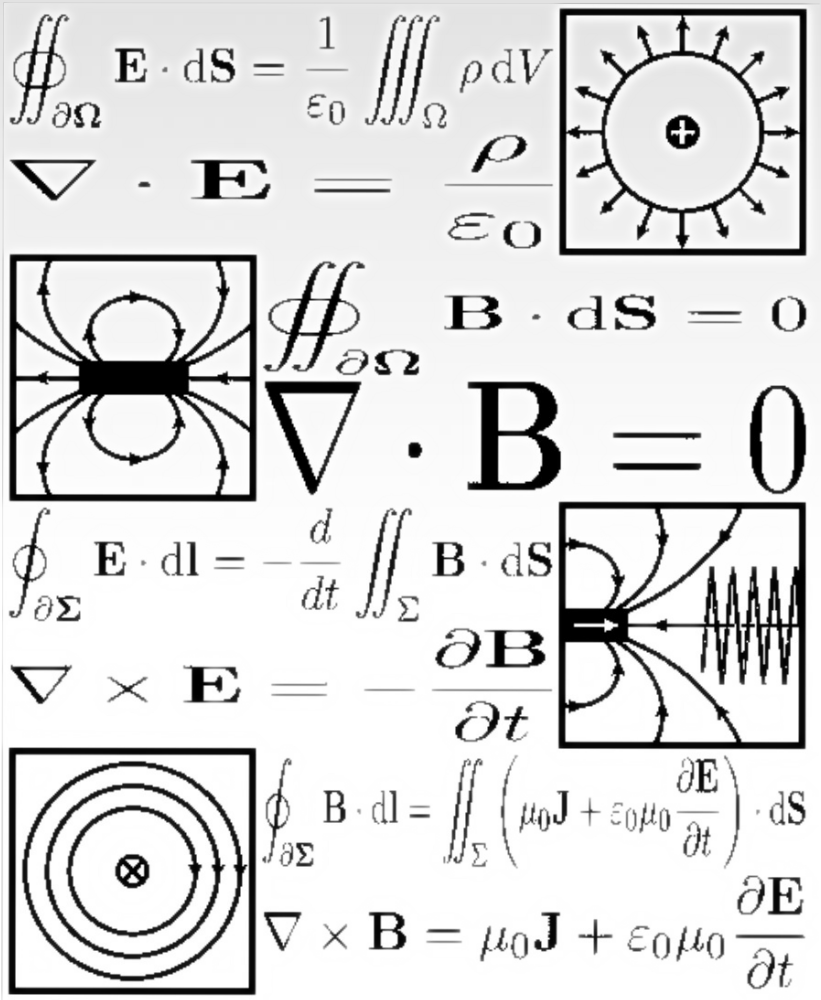
- Sample Code
- Schematic/Wiring Diagrams
- Web Resources

Getting Started



Electricity / Electronics Basic Concepts

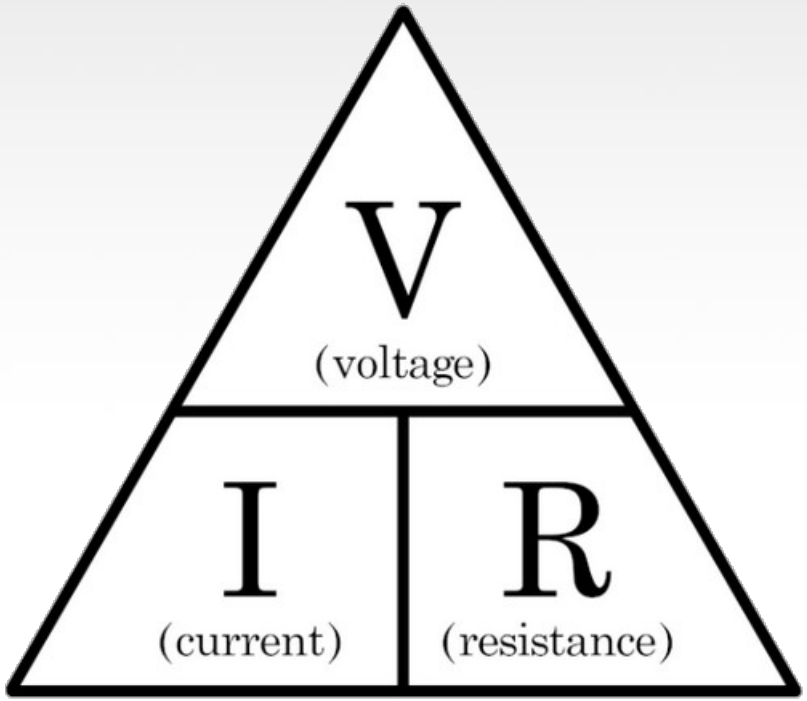
- Voltage
- Current
- Resistance
- Ohm's Law
- Using a Multimeter

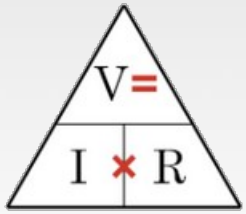


Electrical Properties

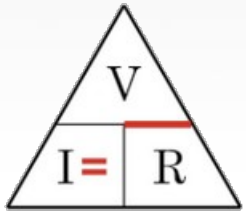
Voltage V	Current I	Resistance R
• The amount of potential energy in a circuit (how "hard" it can push electrons) • Units: Volts (V)	• Related to number of electrons that flow through a circuit every second • Units: Amperes (A)	• How difficult it is for electrons to flow through a circuit • Units: Ohms (Ω)

Ohm's Law

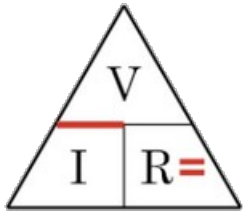




$$V = I \times R$$



$$I = V \div R$$



$$R = V \div I$$

$V = I R$

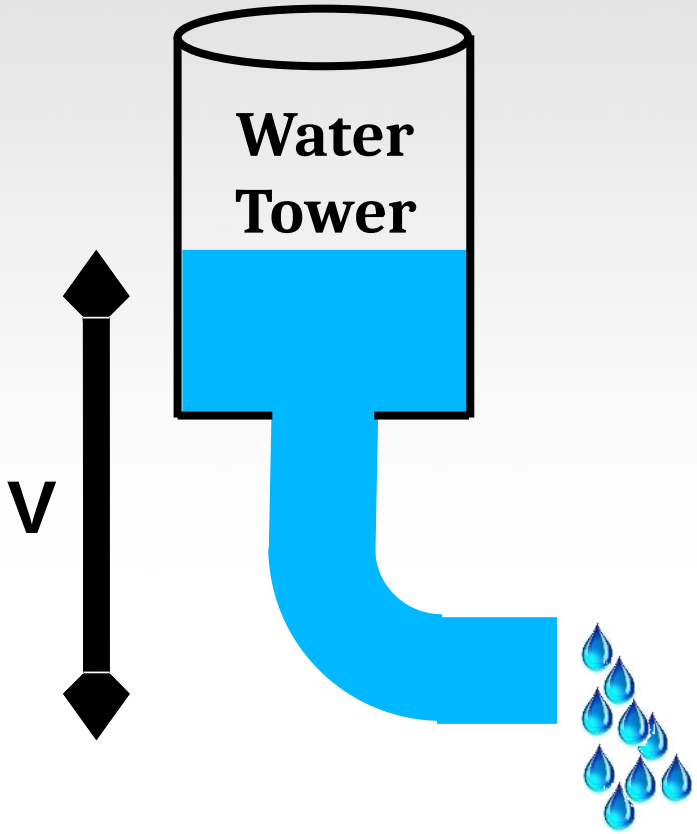
Current Flow Analogy

High Current

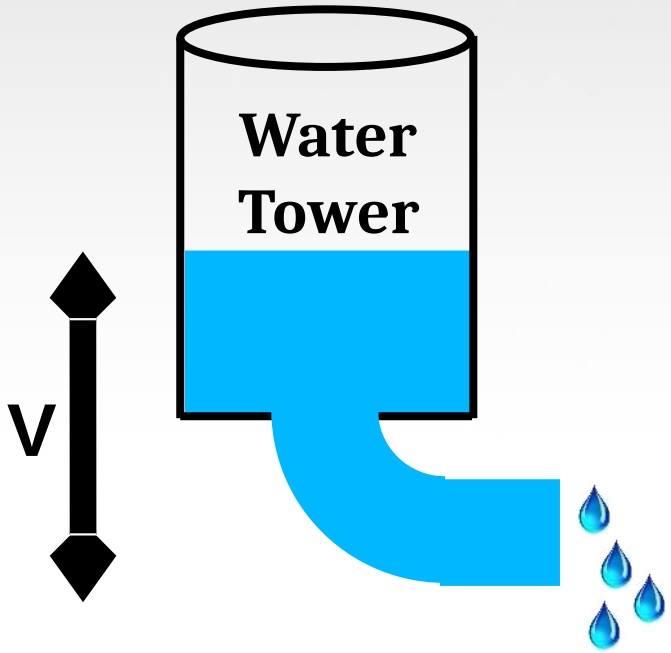
Low Current

$$V = I R$$

Voltage Analogy



More Energy == Higher Voltage



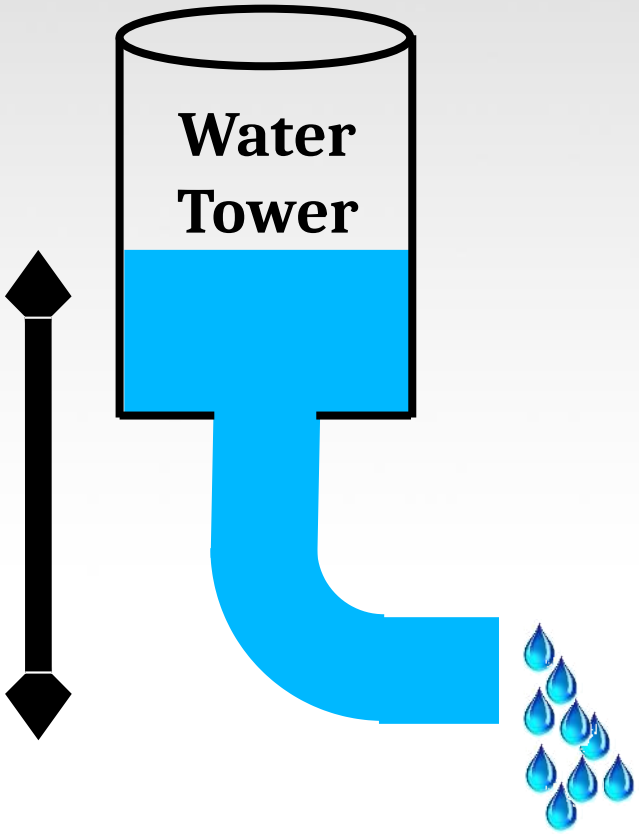
Less Energy == Lower Voltage

$$V = I R$$

$$V = I R$$

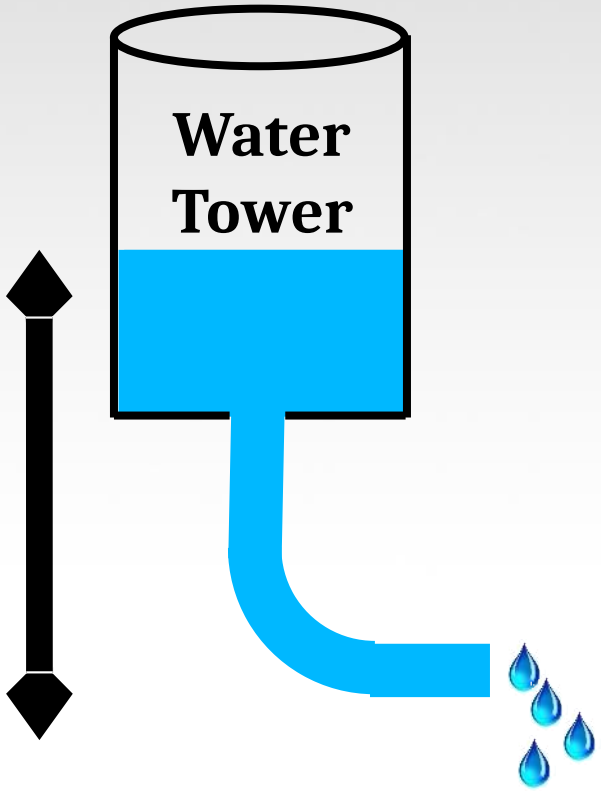
$$V = I R$$

Resistance Analogy



Big Pipe == Lower Resistance

$$V = I R$$

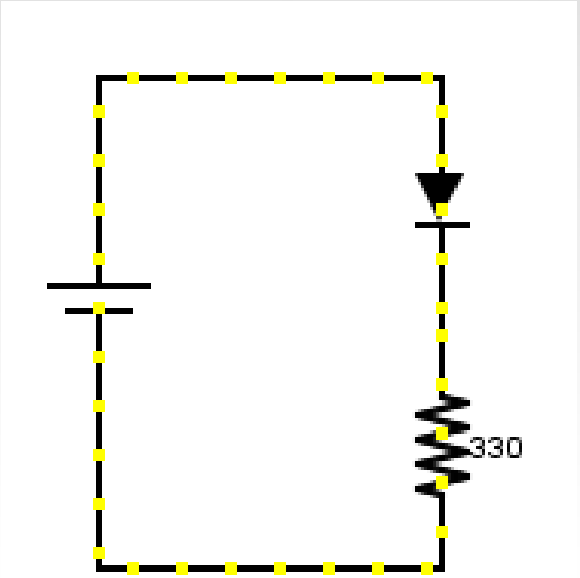


Small Pipe == Higher Resistance

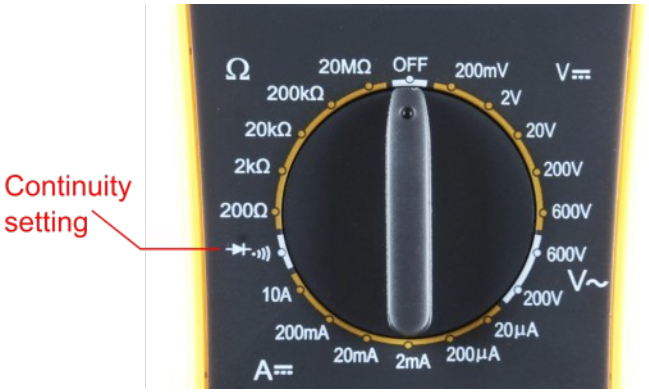
$$V = I R$$

Continuity – Is it a Circuit?

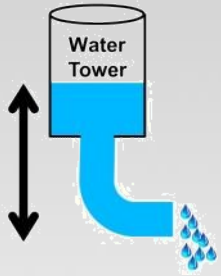
The word “circuit” is derived from the circle. An Electrical Circuit must have a continuous LOOP from Power (V_{cc}) to Ground (GND).



Continuity- it is important to make sure all the portions of circuits are connected. Continuity is the simplest and possibly the most important setting on your multimeter. Sometimes we call this “ringing out” a circuit.

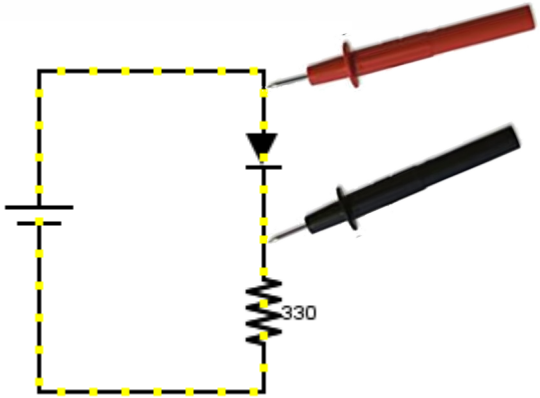


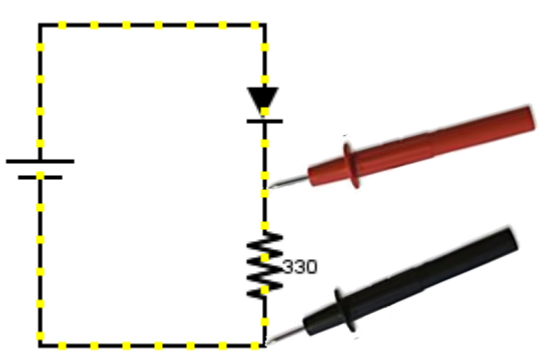
Measuring Electricity – Voltage

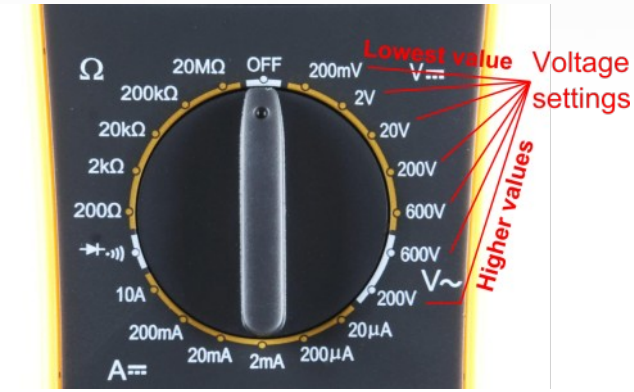


Voltage is a measure of potential electrical energy.

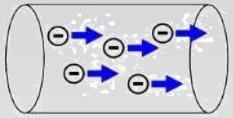
A voltage is also called a potential difference – it is measured between two points in a circuit – across a device.





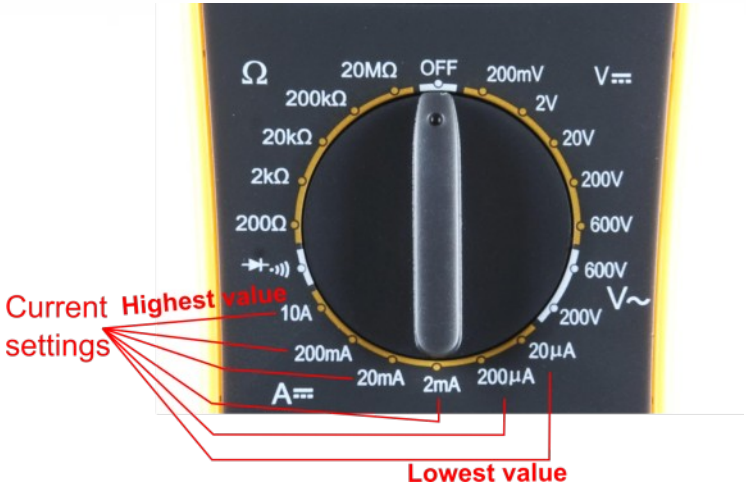


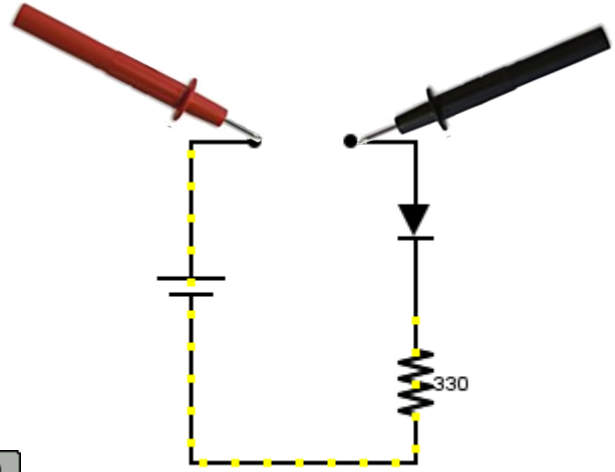
Measuring Electricity -- Current



Current is the measure of the rate of charge flow. For Electrical Engineers – we consider this to be the movement of electrons.

In order to measure this – you must break the circuit or insert the meter in-line (series).





Measuring Electricity -- Resistance

Resistance is the measure of how much opposition to current flow is in a circuit.

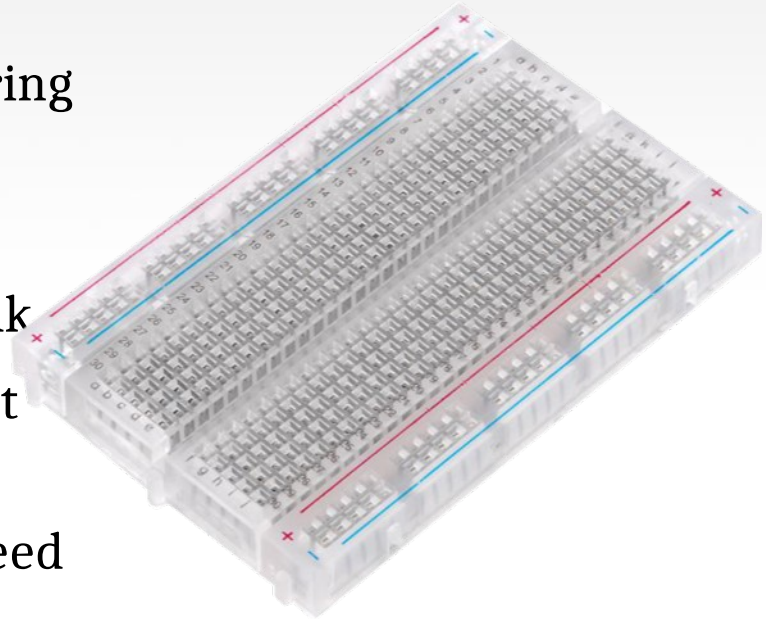
Components should be removed entirely from the circuit to measure resistance. Note the settings on the multimeter. Make sure that you are set for the appropriate range.

Resistance settings

Prototyping Circuits
Solderless Breadboard

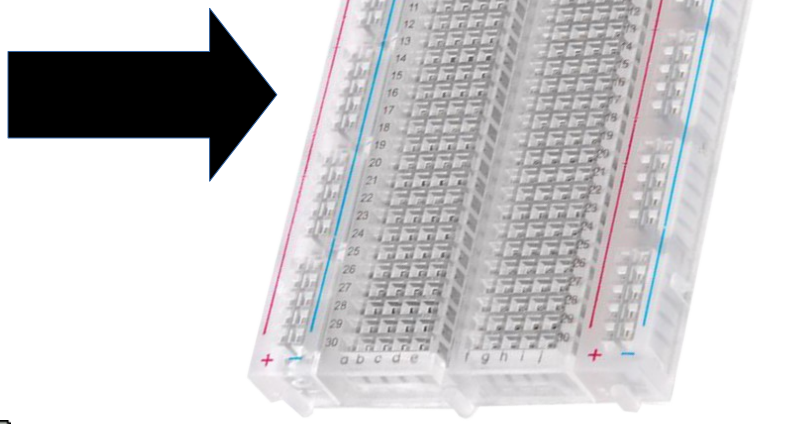
One of the most useful tools in an engineer or Maker's toolkit. Good things to remember:

- A breadboard is easier than soldering
- A breadboard isn't the same as soldering
  - Sometimes breadboards break
  - Sometimes breadboards don't make good connections
  - They're not good for high-speed applications
- Know what dots are connected to each other

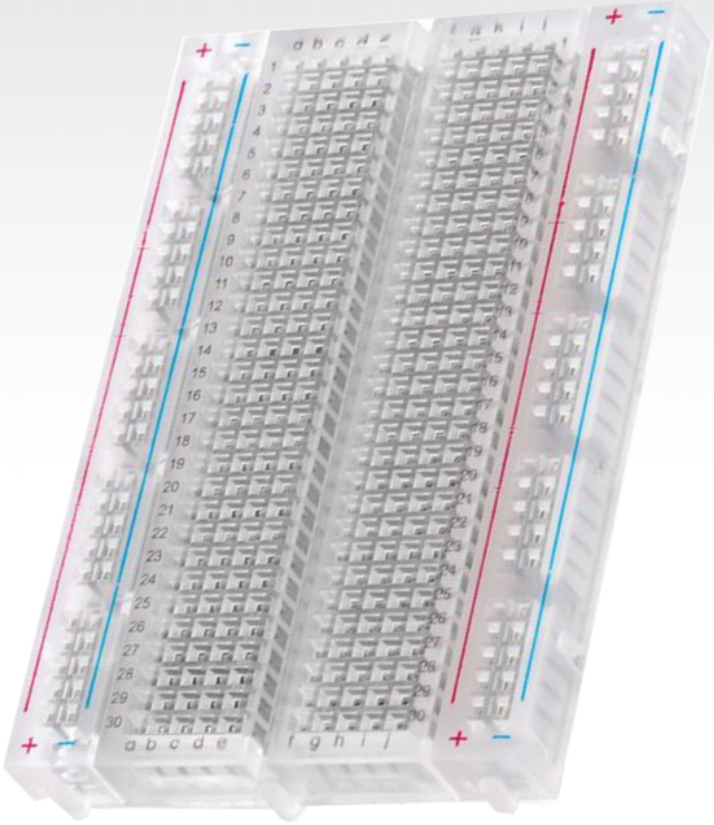


Why a Breadboard?

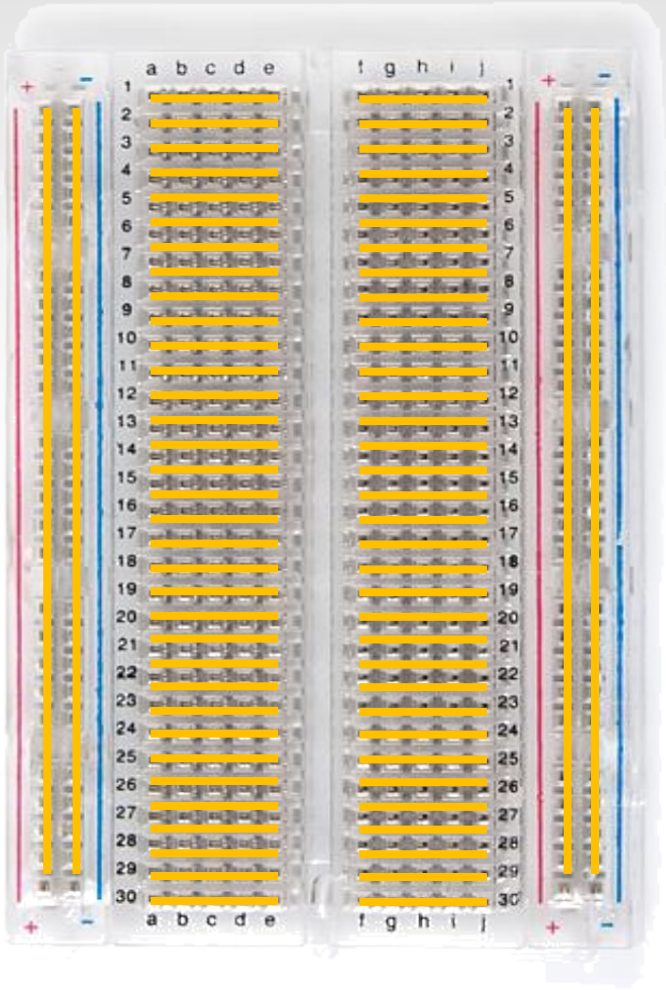
Atwater Kent Radios (ca 1923)



What's a Breadboard?

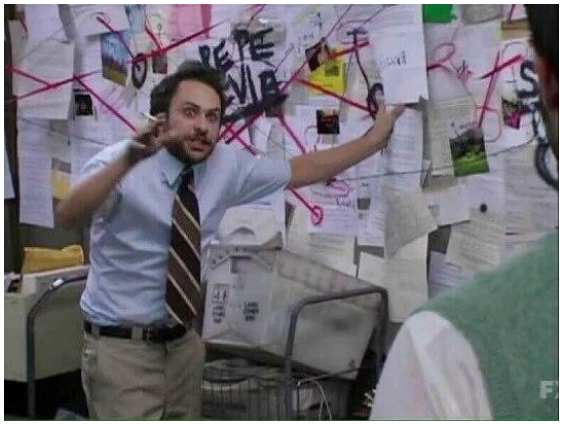


How It's All Connected

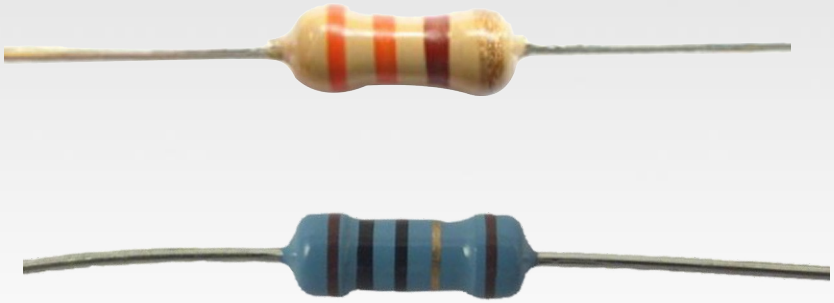


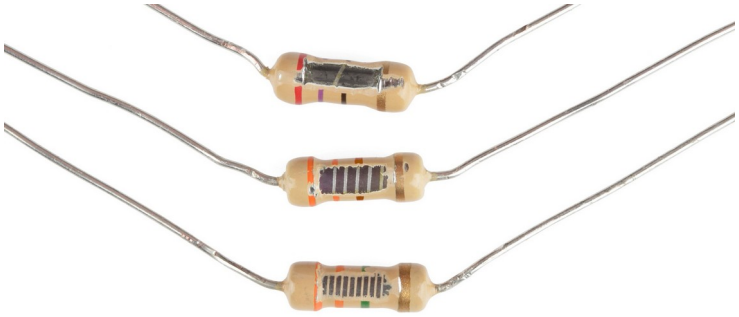
Each row (horizontal) of 5 holes are connected

Vertical columns- called power buses- are connected vertically



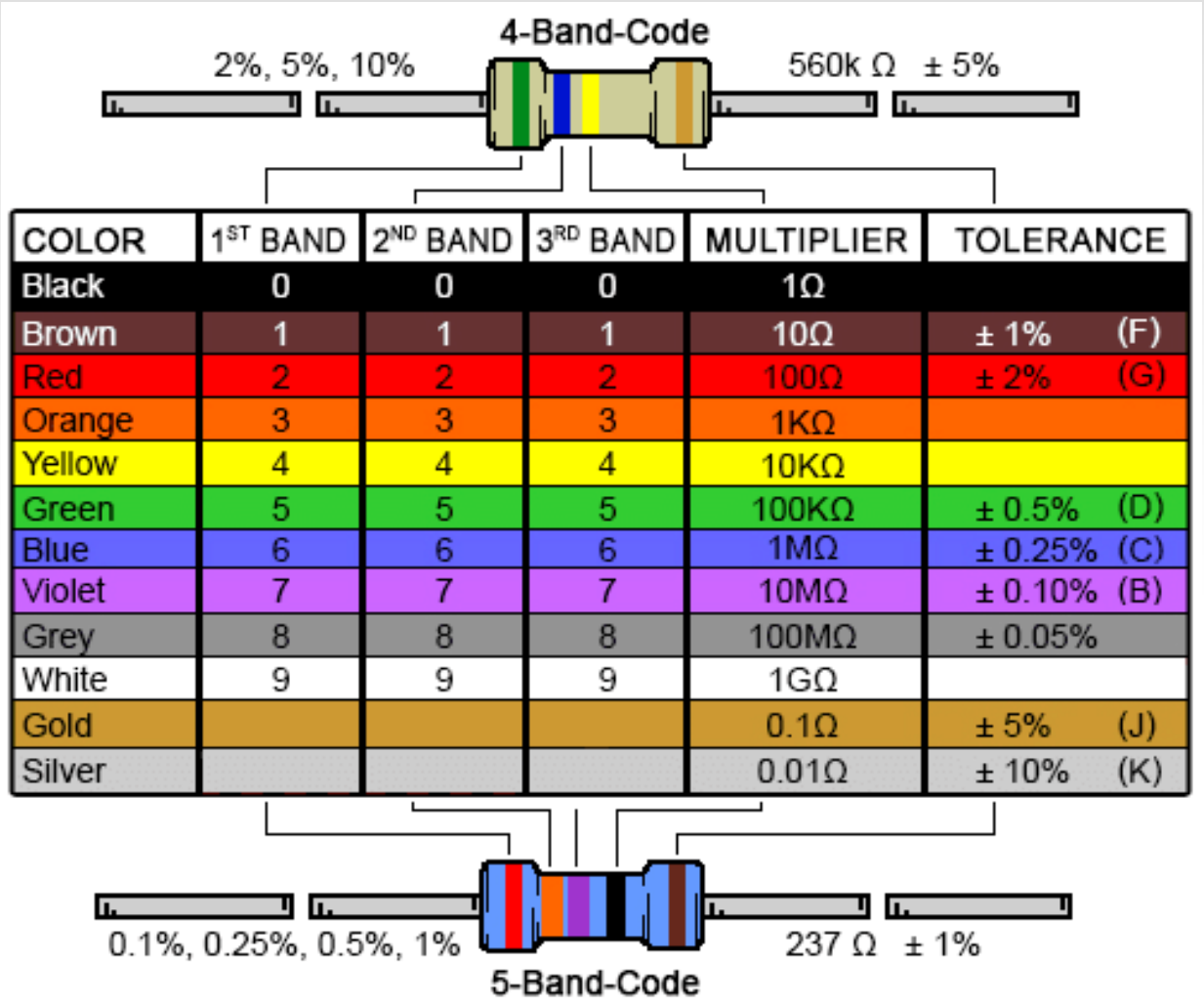
Component Spotlight: The Resistor





Ohms(Ω)

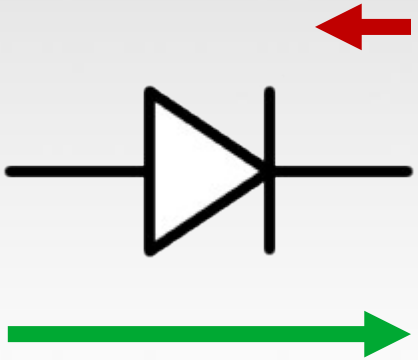
Decoding Resistor Values

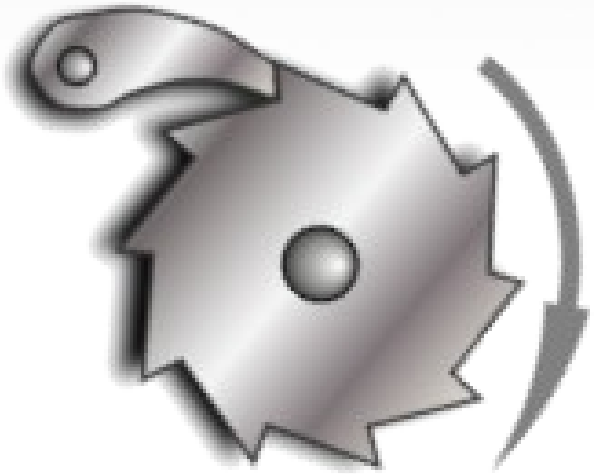


COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8	100MΩ	± 0.05%
White	9	9	9	1GΩ	
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

Component Spotlight: The LED

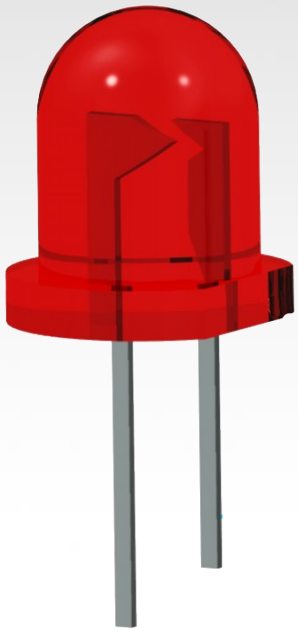
Diode

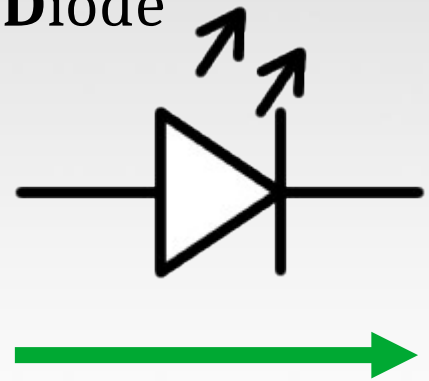


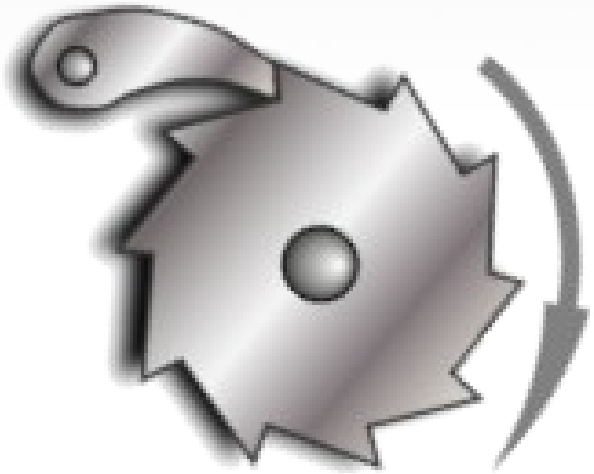


Component Spotlight: The LED

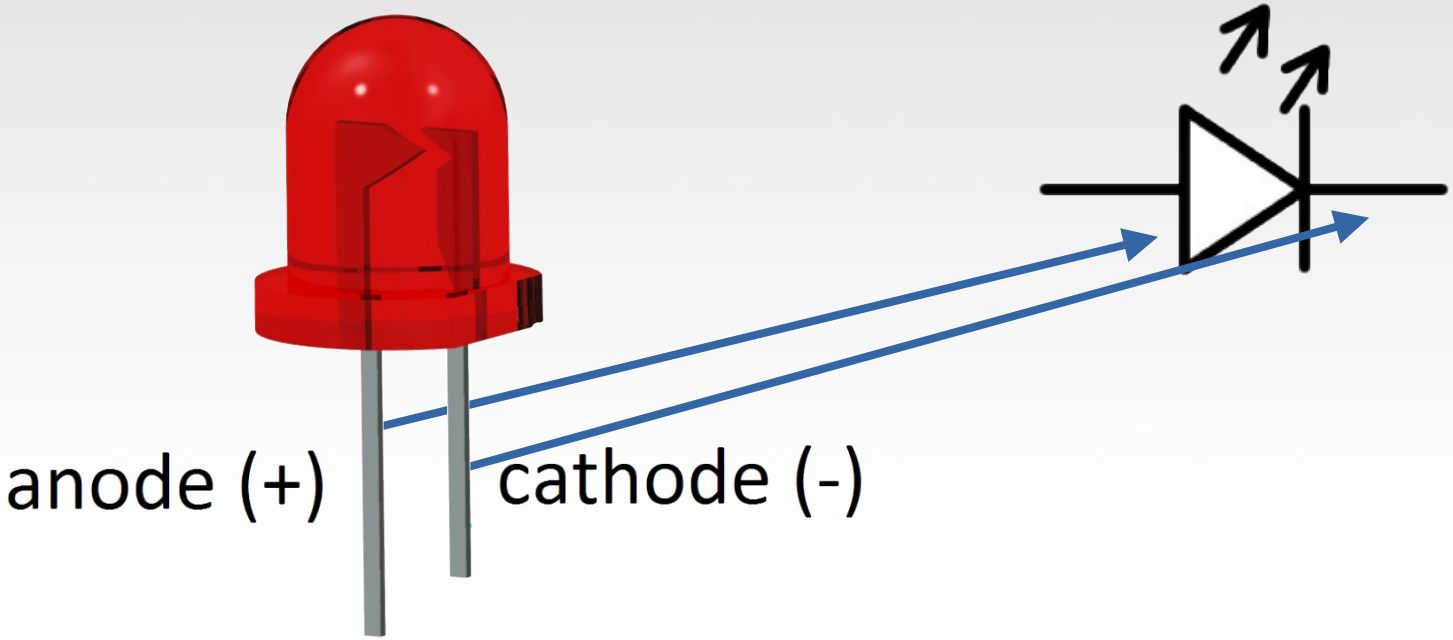
Light Emitting Diode



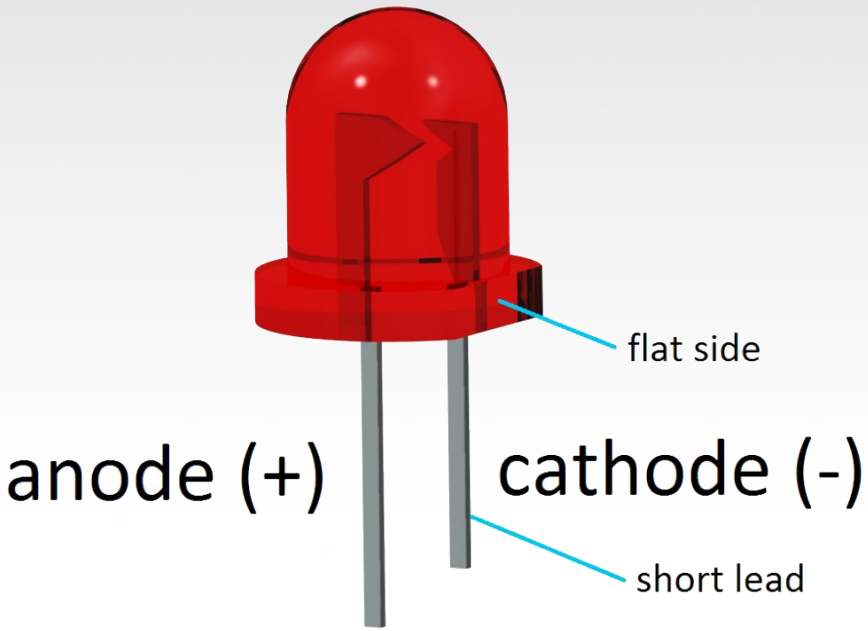




Component Spotlight: The LED

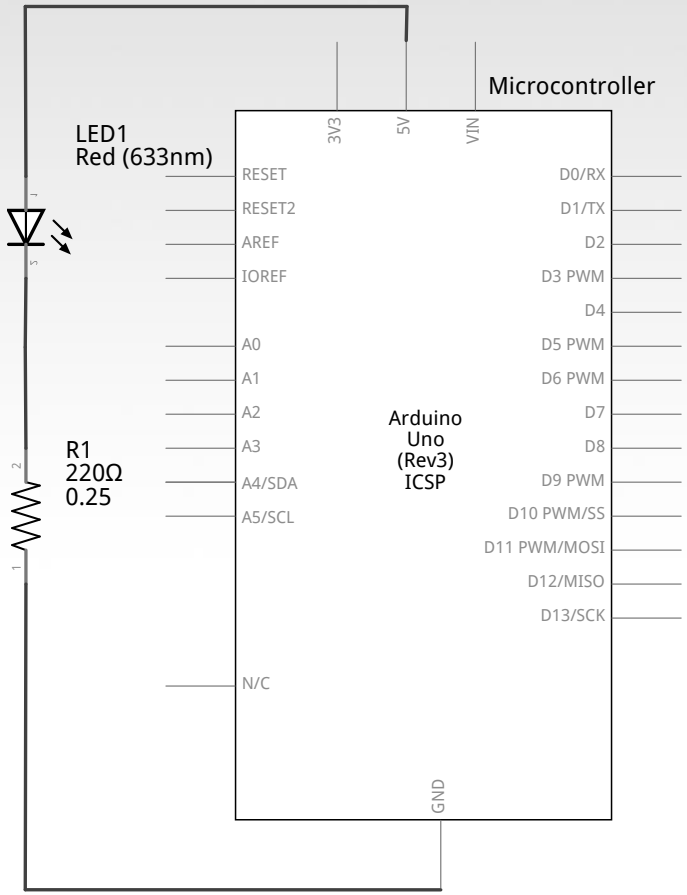


LED Mnemonic



Cathode, like Kathy, is short and negative

Using the Breadboard to built a simple circuit



Use the breadboard to wire up

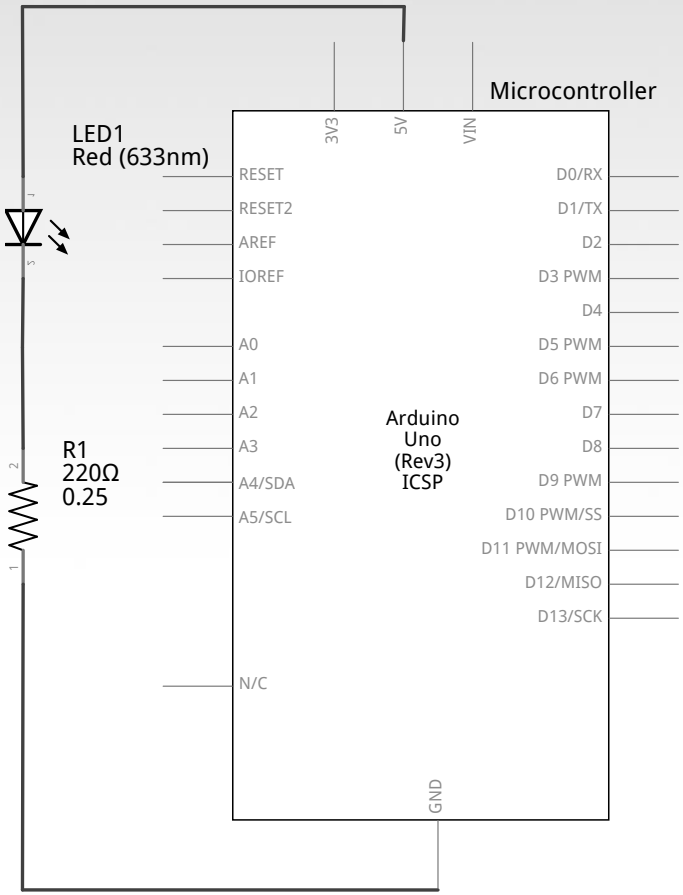
- a single LED
- a 220 Ω Resistor

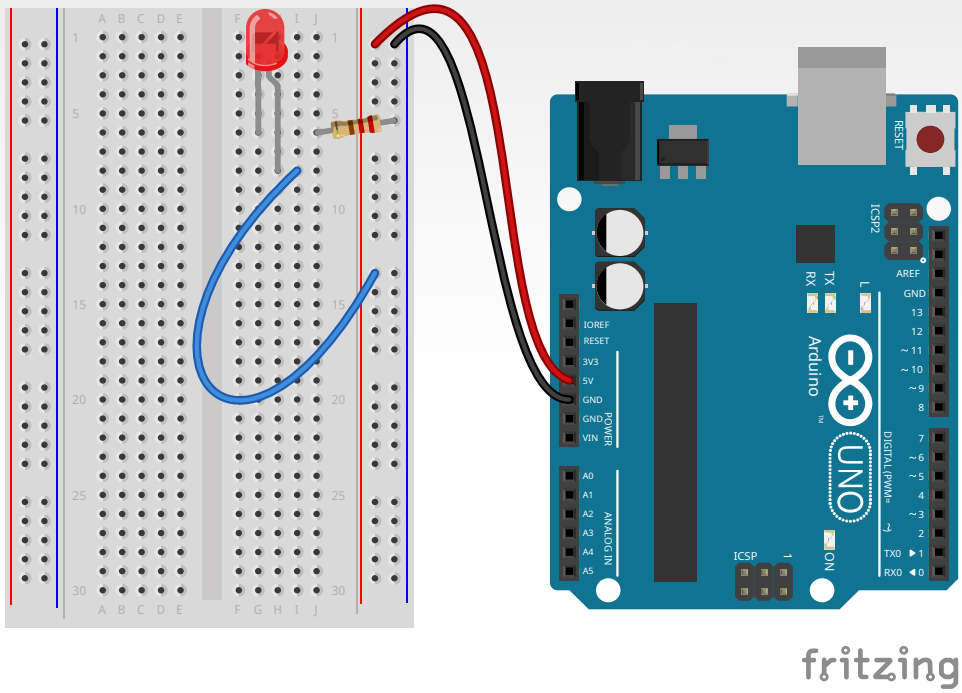
  (Red-Red-Black-Black-Gold)

  or

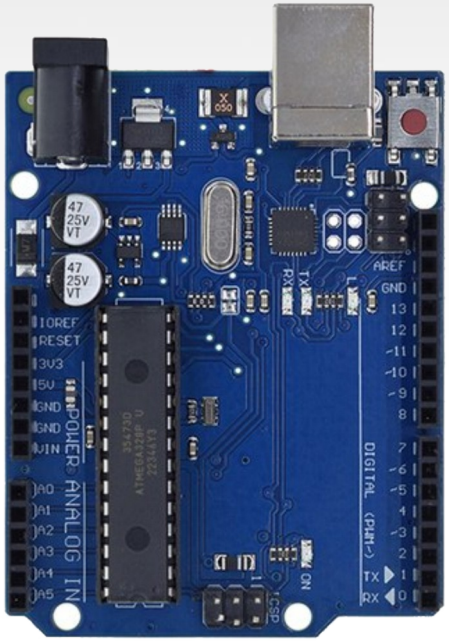
  (Red-Red-Brown-Gold)

Using the Breadboard to built a simple circuit

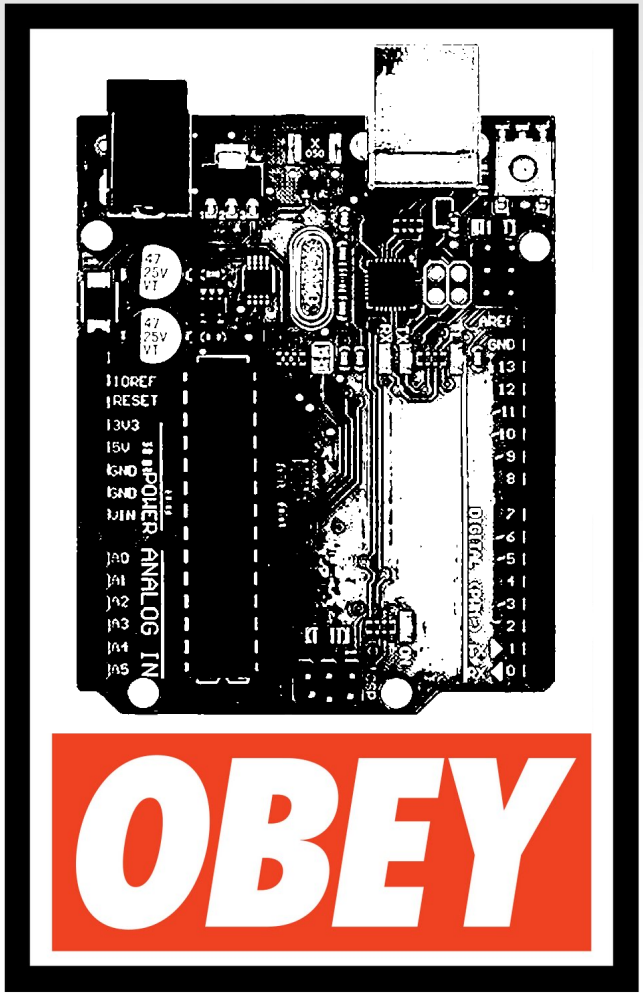




Go ahead and plug your board in!



Adding Control



Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller

Input is a signal / information going into the board.

Output is any signal exiting the board.

Almost all systems that use physical computing will have an output

What are some examples of inputs and outputs?

Concepts: INPUT vs. OUTPUT

Referenced from the perspective of the microcontroller

Input is a signal / information going into the board.

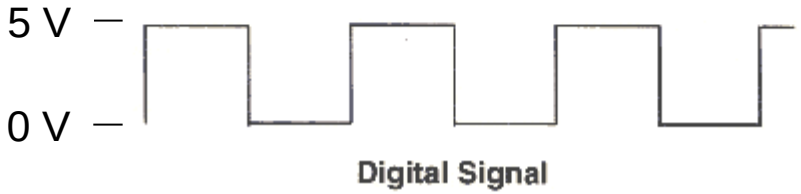
Output is any signal exiting the board.

Examples: Buttons, Switches, Microphones, Light Sensors, Touch Sensors, Flex Sensors, Humidity Sensors, Temperature Sensors…	Examples: LEDs, RGB LEDs, monitors, relays, DC motors, servo motors, buzzers, speakers

Concepts: Analog vs. Digital

Microcontrollers are **digital** devices – ON or OFF. Also called discrete.

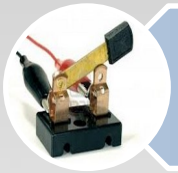
Analog signals are anything that can be a full range of values. What are some examples?



Digital Signal

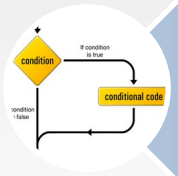
Analog Signal

BIG 6 CONCEPTS



`digitalWrite()`

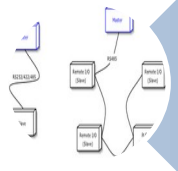
`analogWrite()`



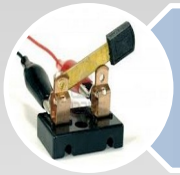
`if()` statements / Boolean logic

`digitalRead()`

`analogRead()`



Serial communication

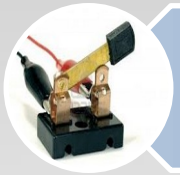


digitalWrite()

Project # 1 – Blink

"Hello World" of Physical Computing

Psuedo-code – how should this work?

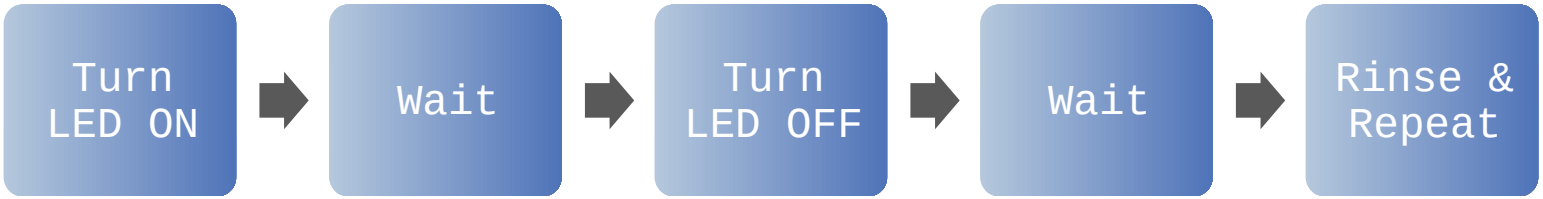


digitalWrite()

Project # 1 – Blink

"Hello World" of Physical Computing

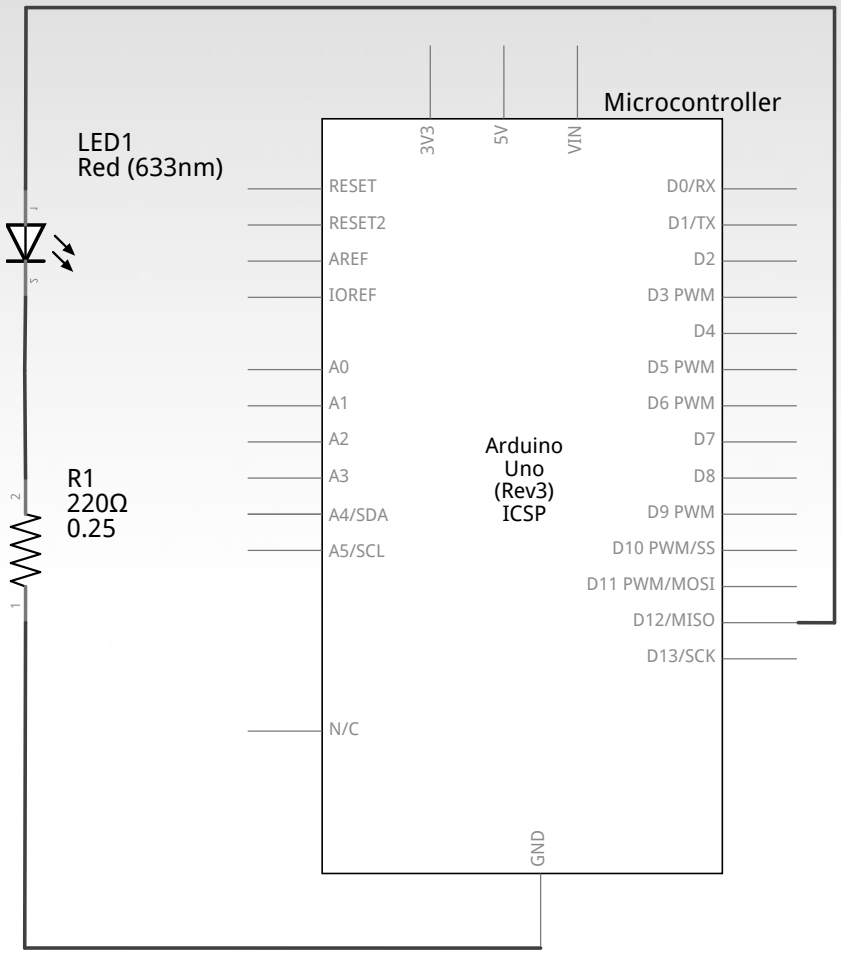
Psuedo-code – how should this work?



Project # 1 - Blink
Inputs and Outputs

Inputs	Outputs
None	LED/resistor

Project # 1 - Blink
Schematic

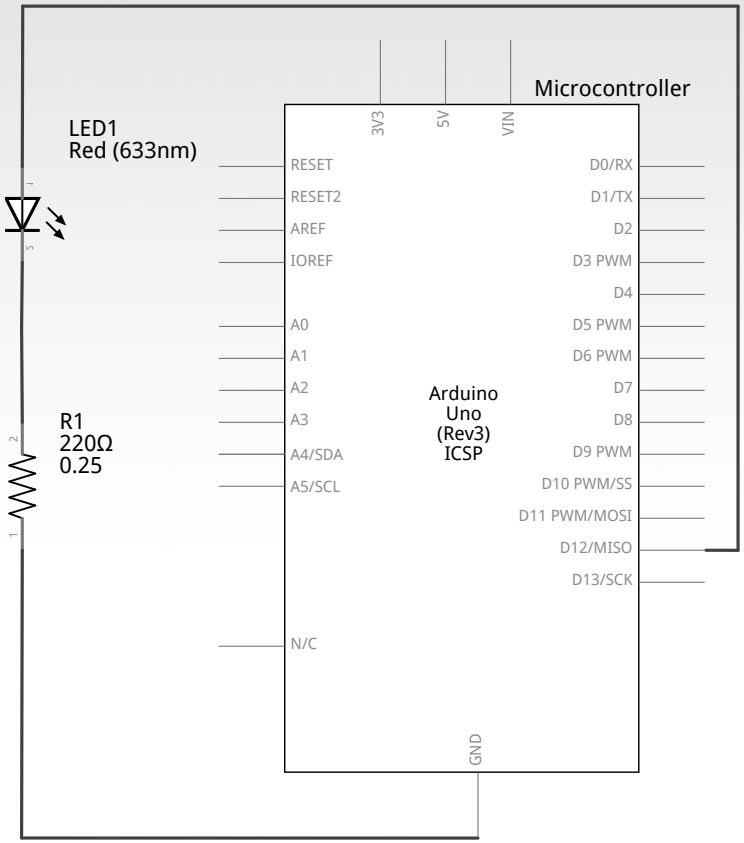


Move the blue wire from the power bus to pin 12 (or any other Digital I/O pin) on the microcontroller board.

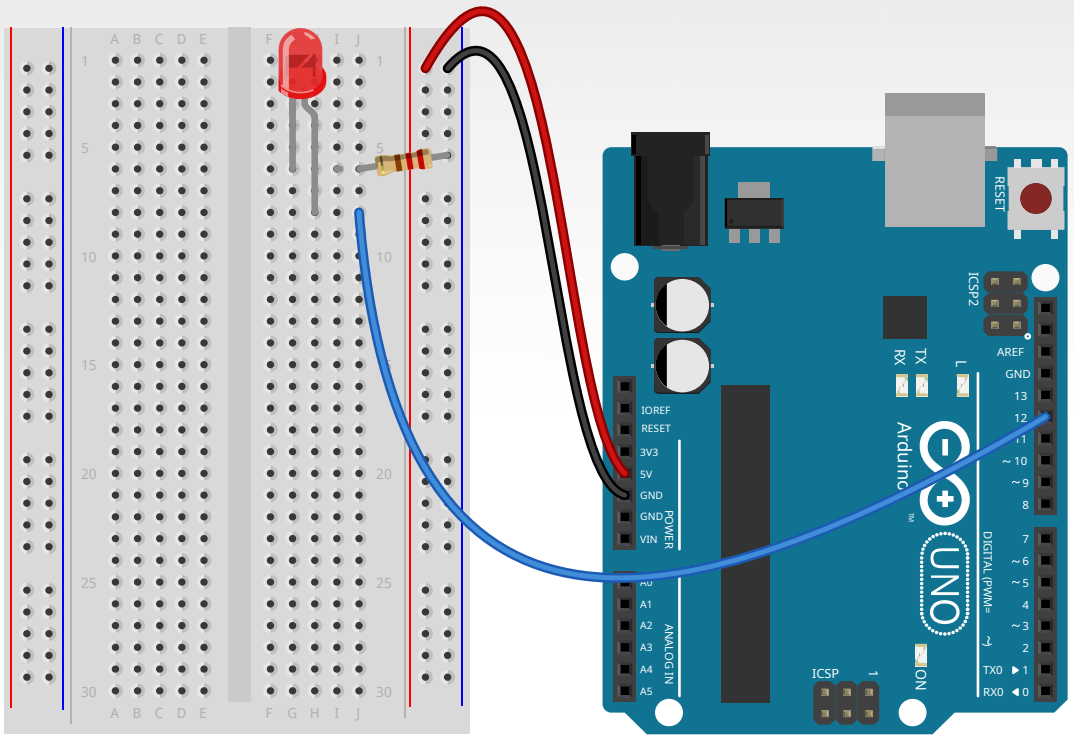
fritzing

Project # 1 – Blink

Wiring Diagram

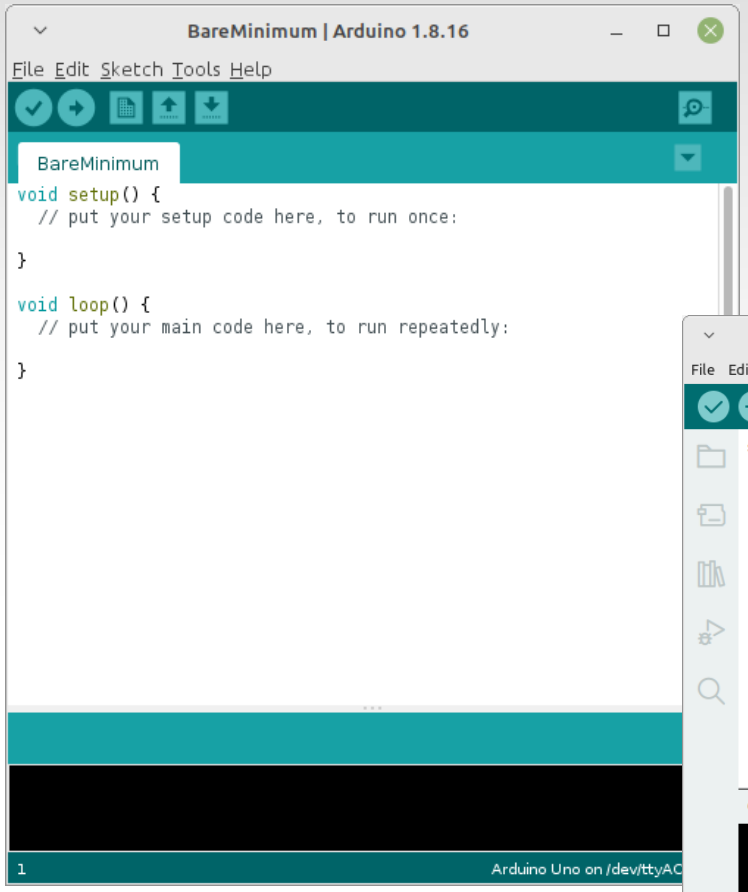


fritzing



fritzing

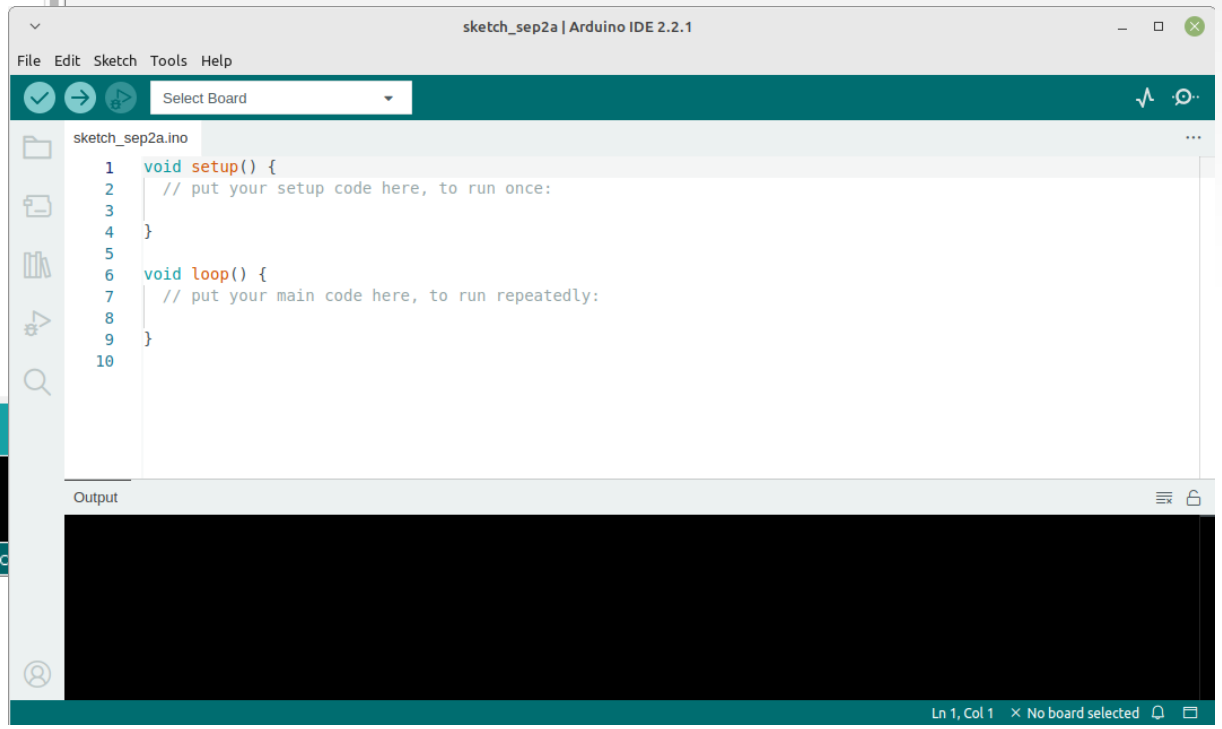
Open up Arduino IDE



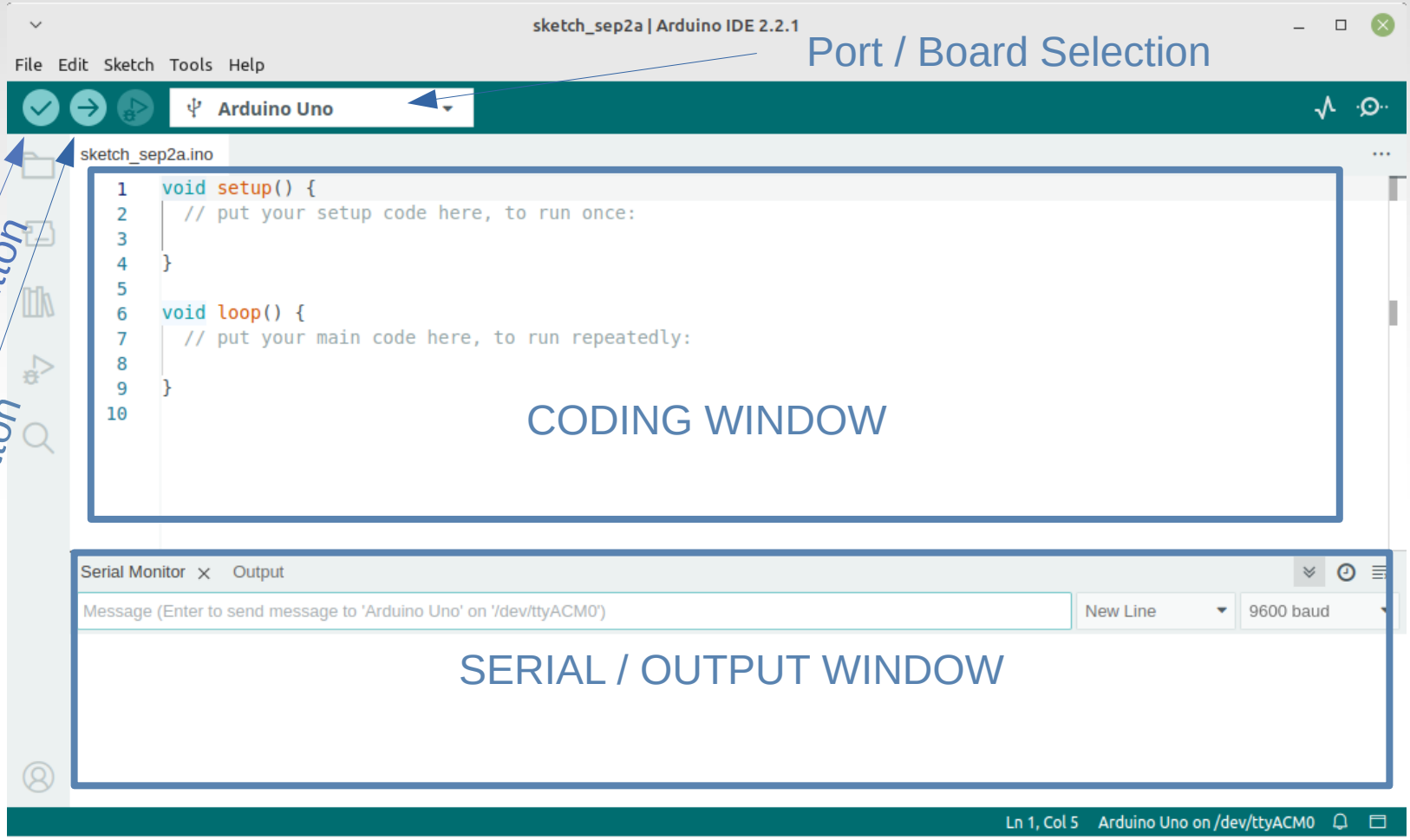
IDE → **I**ntegrated **D**evelopment **E**nvironment

Lets you write, compile (and upload) code all in the same place

Versions 1 and 2 are aesthetically different, functionally the same (for our purposes)



Exploring the Arduino IDE



Coding

This is like trying to learn ‘sports’ in a few hours, or ‘cooking’ or anything else that is a field of study unto itself

We’re going to go over some very broad concepts, in the form of commands and function, giving you just enough to be dangerous

You can copy/paste the examples, or type them in yourself, then we’ll review what each is doing line by line

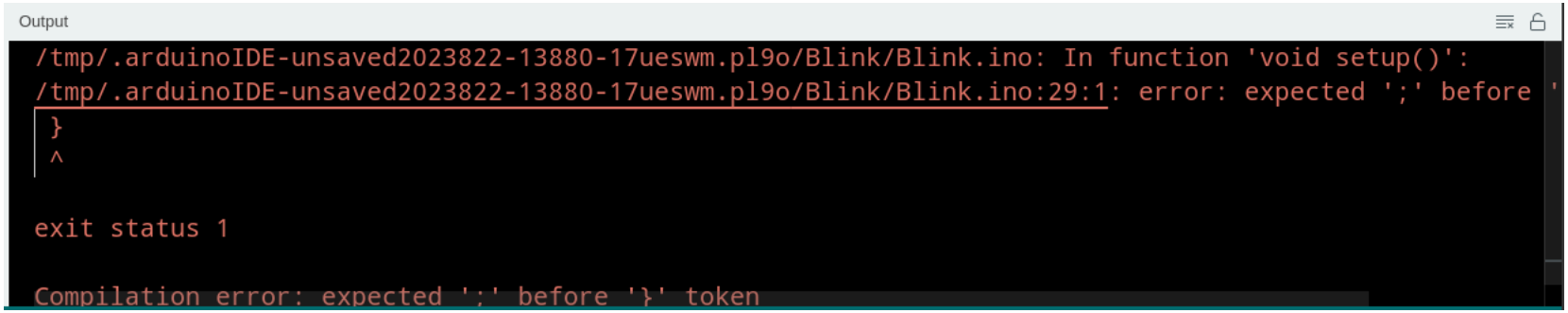
Commands

Commands are what you want the microcontroller to do. They're a code to translate between English and computer-speak

Only one command can go on a line

Because it's a code, the **syntax** or how you type the commands, is very important

If you don't get it just right, the output at the bottom of the IDE will try to tell you what's wrong



Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);




// NOTE: -> commands are CASE-sensitive
```

Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(12, OUTPUT);




// NOTE: -> commands are CASE-sensitive
```

Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(12, OUTPUT);

digitalWrite(pin, HIGH/LOW);




// NOTE: -> commands are CASE-sensitive
```

Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(12, OUTPUT);

digitalWrite(pin, HIGH/LOW);
  ex: digitalWrite(12, HIGH);



// NOTE: -> commands are CASE-sensitive
```

Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(12, OUTPUT);

digitalWrite(pin, HIGH/LOW);
  ex: digitalWrite(12, HIGH);

delay(time_ms);


// NOTE: -> commands are CASE-sensitive
```

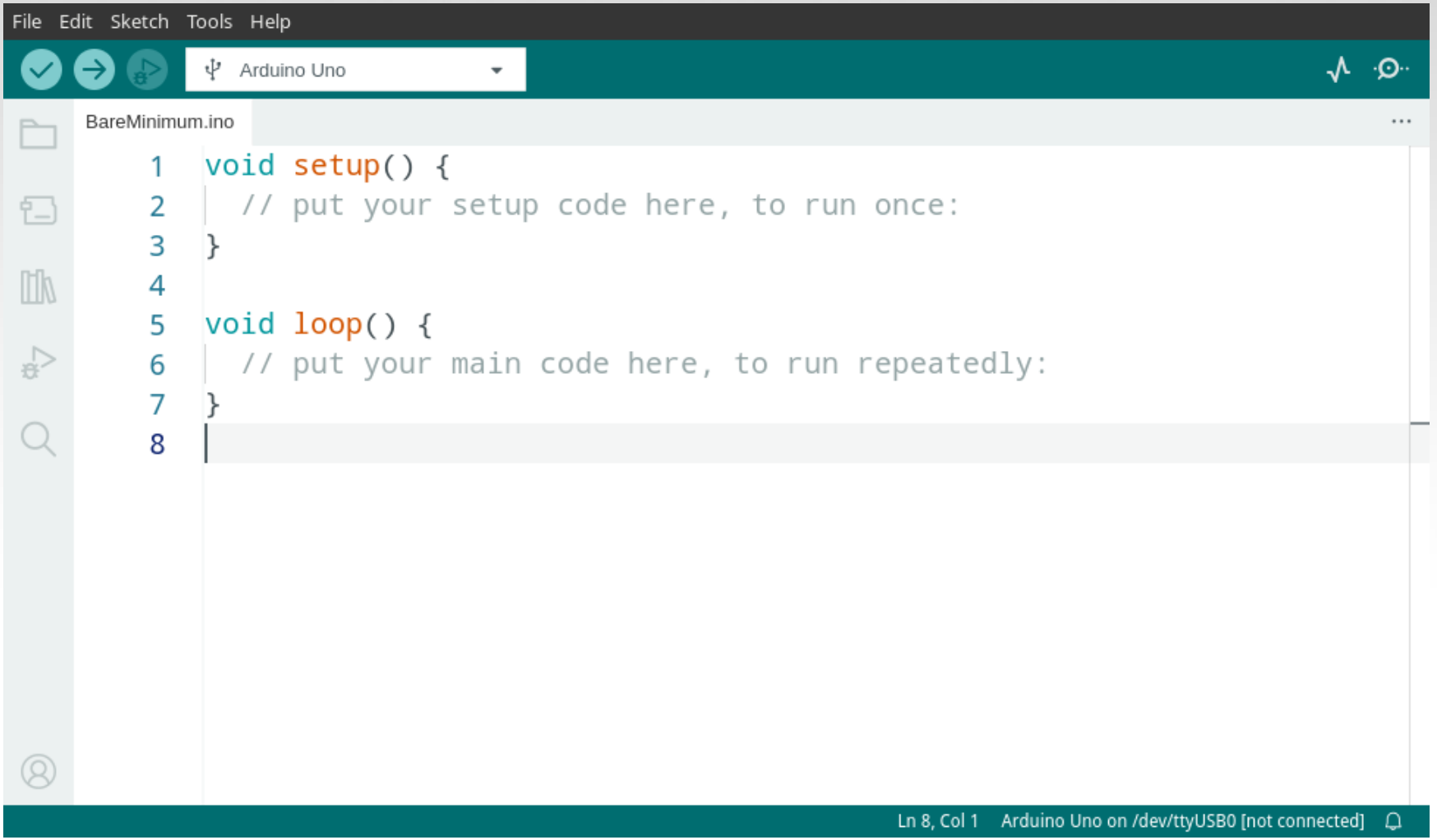
Project # 1 – Blink
Three Command to Start

```
pinMode(pin, INPUT/OUTPUT);
  ex: pinMode(12, OUTPUT);

digitalWrite(pin, HIGH/LOW);
  ex: digitalWrite(12, HIGH);

delay(time_ms);
  ex: delay(2500);

// NOTE: -> commands are CASE-sensitive
```

Functions

Functions are groups of commands

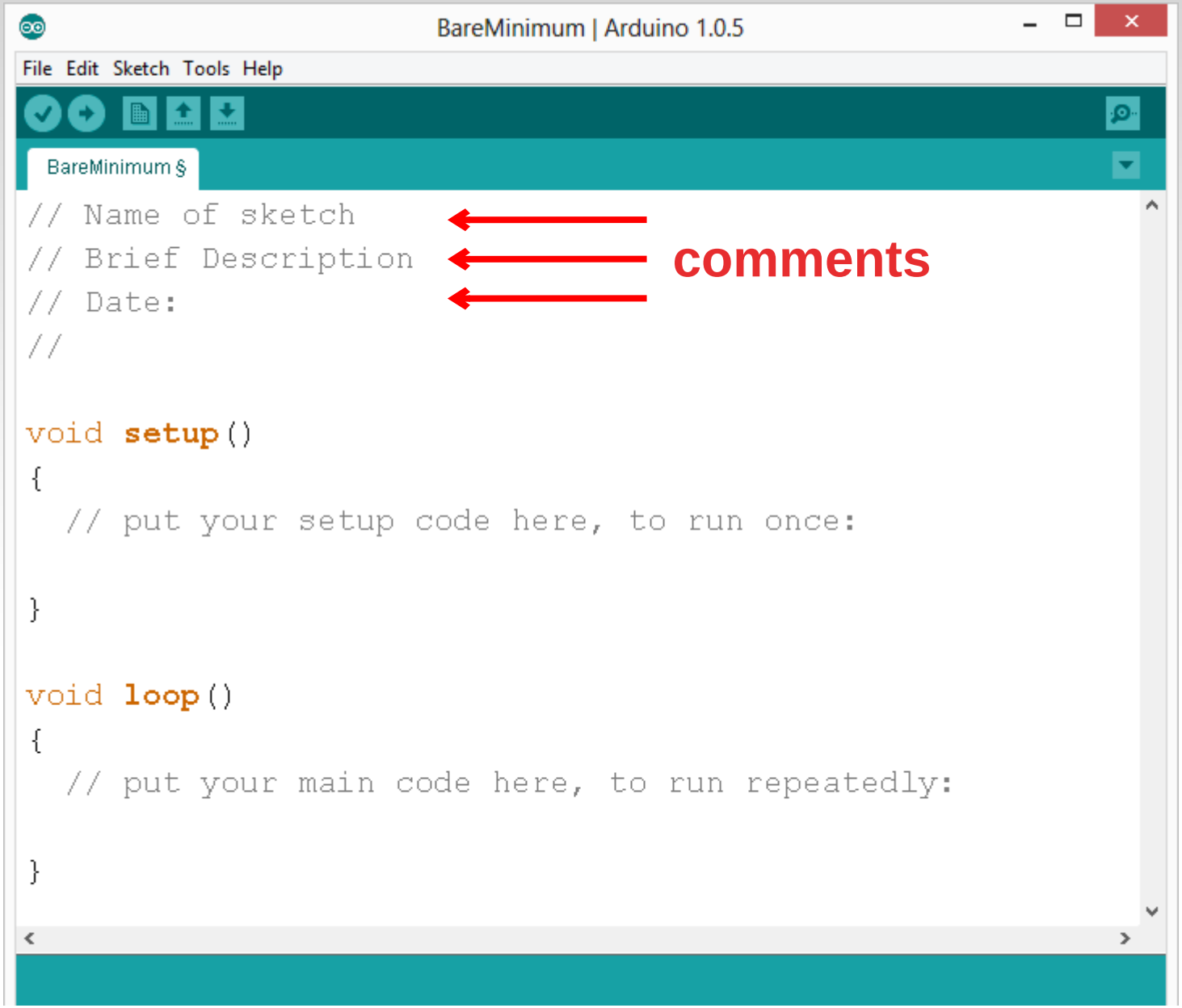
- They need a unique name
- They *can* take numbers as inputs (arguments) and spit out a number as an output (return) but don't have to
- They begin and end with curly brackets → `{ }`
- We wont be dealing with them in too much more detail, save every Arduino program **requires** two functions, the `setup()` function and the `loop()` function



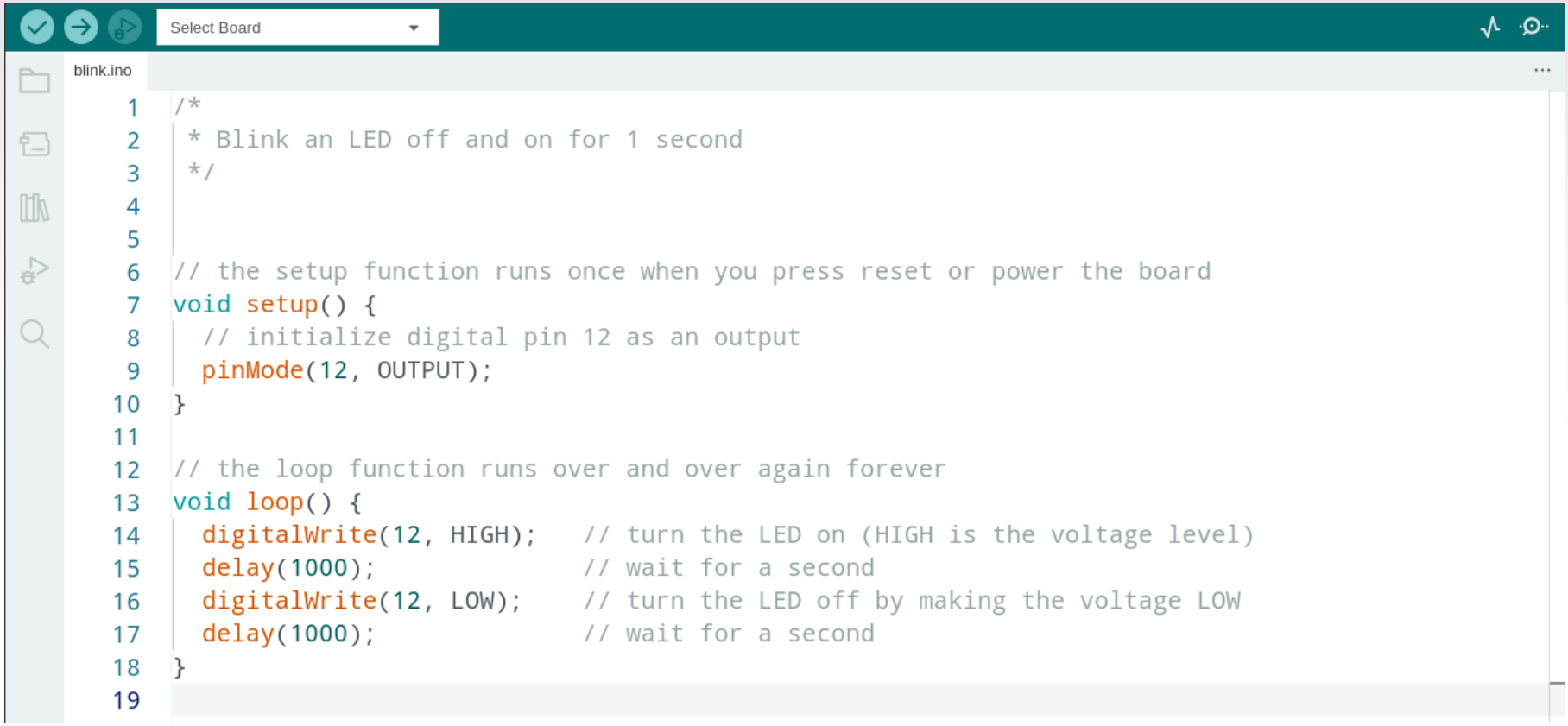
Comments, Comments, Comments

Comments are ignored by the microcontroller. They're just for you – the programmer and your friends…or anyone else human that might read your code.

```
// this is for single line comments

/* this is for multi-line comments
   Like this…
   And this….
*/
```



Project # 1 - Blink
Code Review



Project # 1 – Blink Puzzles

Challenge 1a – blink with a 200 ms second interval.

Challenge 1b – find the fastest blink that the human eye can still detect…

1 ms delay? 2 ms delay? 3 ms delay???

Challenge 1c – blink to mimic a heartbeat

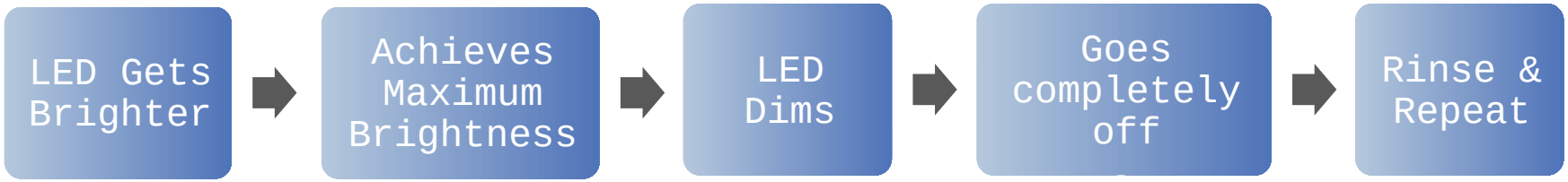
Challenge 1d – change the output pin

analogWrite()

Project # 2 – Fade

Not so fast, Jack

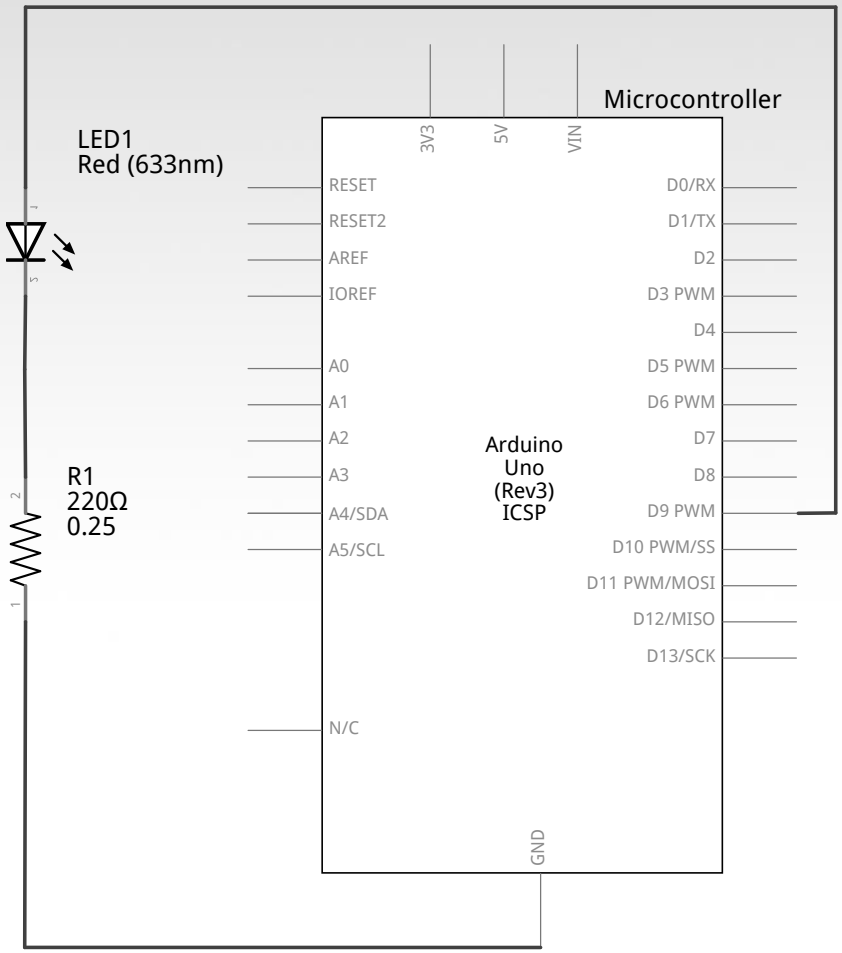
Psuedo-code – how should this work?



Project # 2 – Fade
Inputs and Outputs

Inputs	Outputs
None	LED/resistor

Project # 2 – Fade Schematic

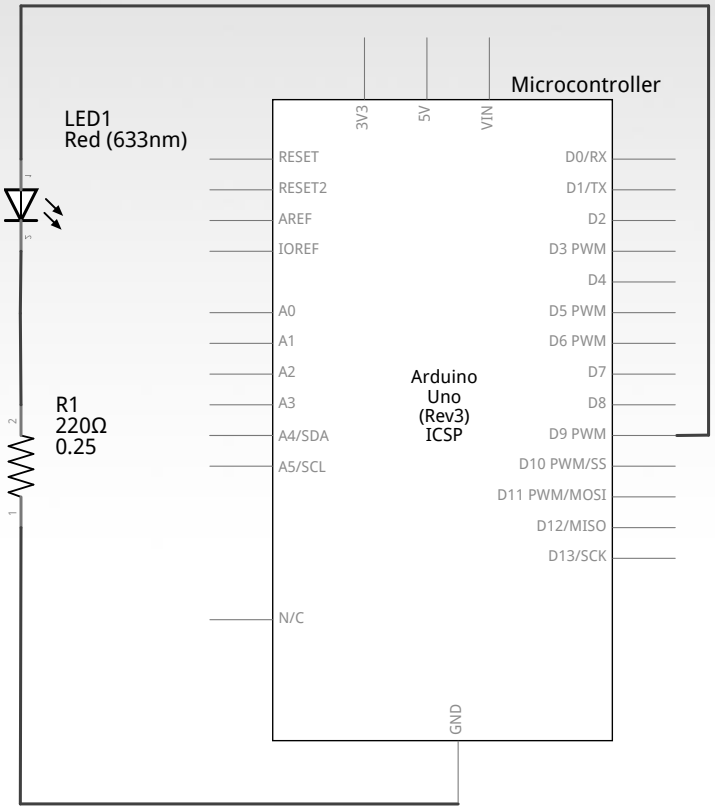


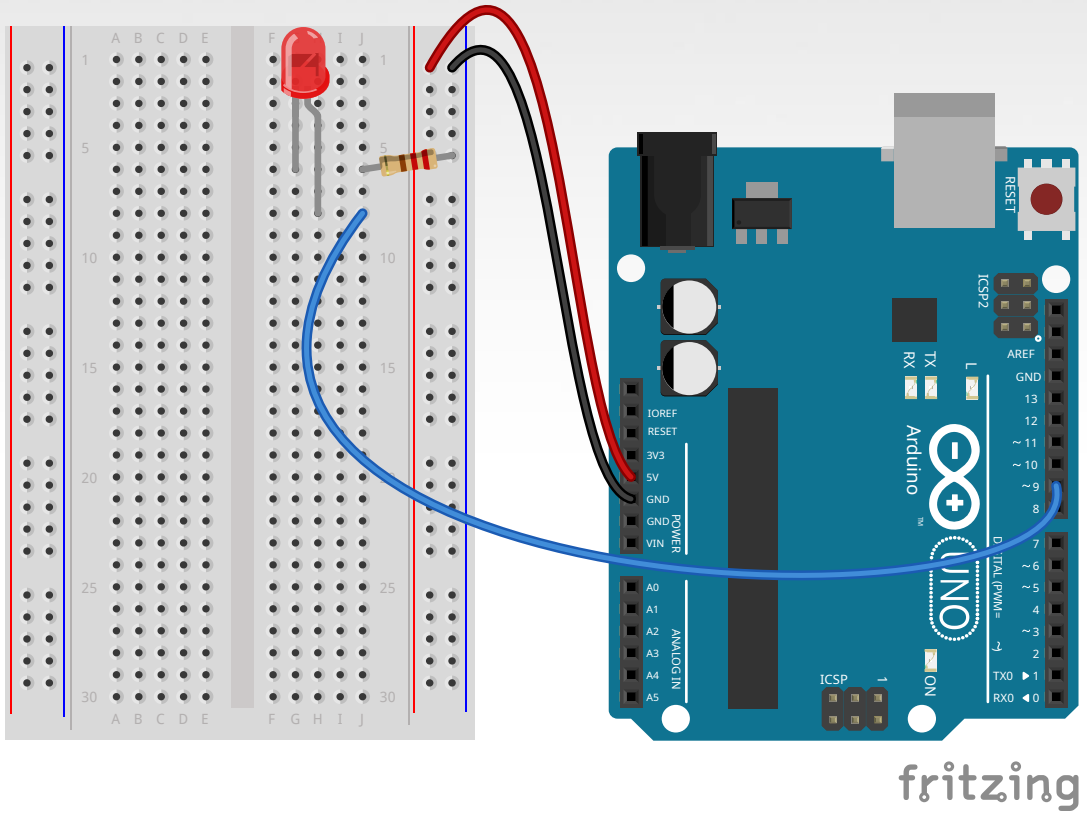
Move the blue wire to pin 9 (or 3, 5, 6, 10 or 11) on the microcontroller board.

fritzing

Project # 2 – Fade

Wiring Diagram





Fading in and Fading Out (Analog or Digital?)

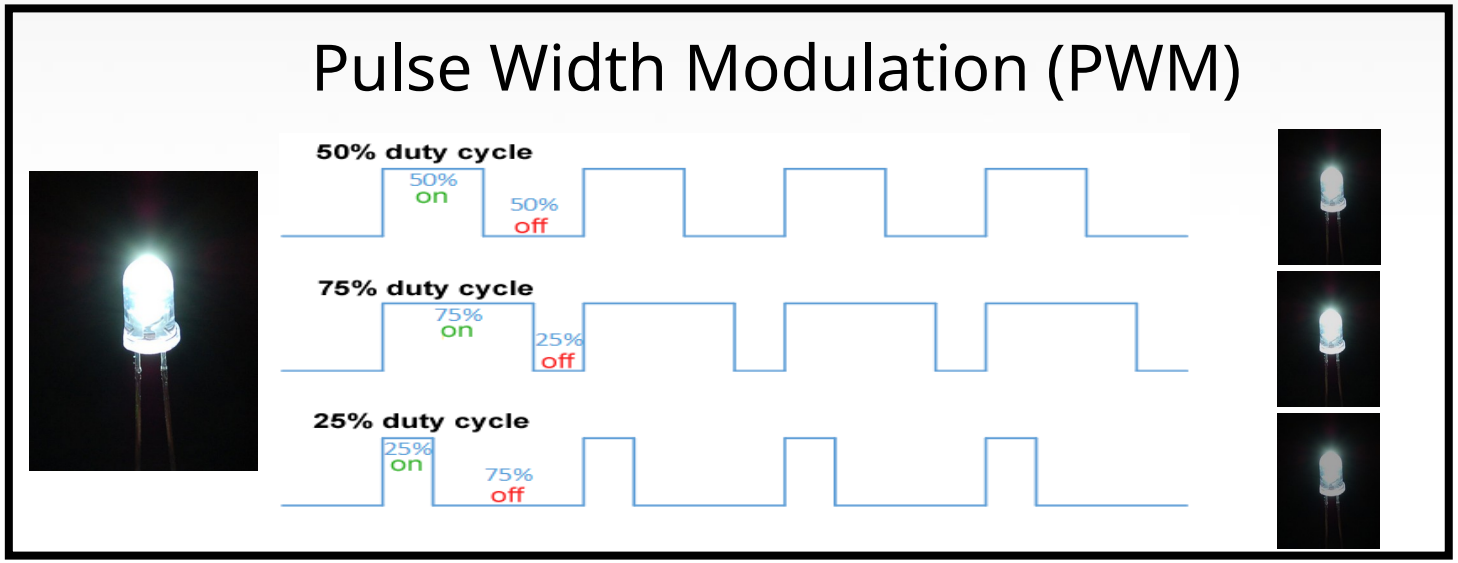
A few pins on the Arduino allow for us to modify the output to “mimic” an analog signal.

This is done by a technique called:

Pulse Width Modulation (PWM)

Concepts: Analog vs. Digital

To create an analog signal, the microcontroller uses a technique called PWM. By varying the duty cycle, we can fool the eye into seeing an average brightness.



Programming Concepts: Variable

```
1   char varA;
2   int varB = 5;
3   long varC = varB;
4
5   varA = 2 * 7;
6   varB = varB + 1;
7   varC = varA - varB;
```

Programming Concepts: Variable Types

Variable Types:

8 bits

byte/char

2^8 = 256 values
(0 - 255)

16 bits

int/ unsigned int

2^{16} = 65,536 values
(-32728 – 32727)
(0 – 65535)

32 bits

long/unsigned long

2^{32} = 4,294,967,296
(-2147483648 – 2147483647)
(0 – 4294967295)

How Big Is 2^{32}?

The function millis() returns the number of milliseconds since the program started

The type of data millis() returns is an unsigned long int

4,294,967,296 milliseconds is 49.71 days

Variables: Which Cup Should You Choose?

Variable Types:

8 bits

byte/char

2^8 = 256 values
(0 - 255)

16 bits

int/ unsigned int

2^{16} = 65,536 values
(-32728 – 32727)
(0 – 65535)

32 bits

long/unsigned long

2^{32} = 4,294,967,296
(-2147483648 – 2147483647)
(0 – 4294967295)

Variables: Which Cup Should You Choose?

Variable Types:

8 bits

byte/char

2^8 = 256 values
(0 - 255)

16 bits

int/ unsigned int

2^{16} = 65,536 values
(-32728 – 32727)
(0 – 65535)

32 bits

long/unsigned long

2^{32} = 4,294,967,296
(-2147483648 – 2147483647)
(0 – 4294967295)

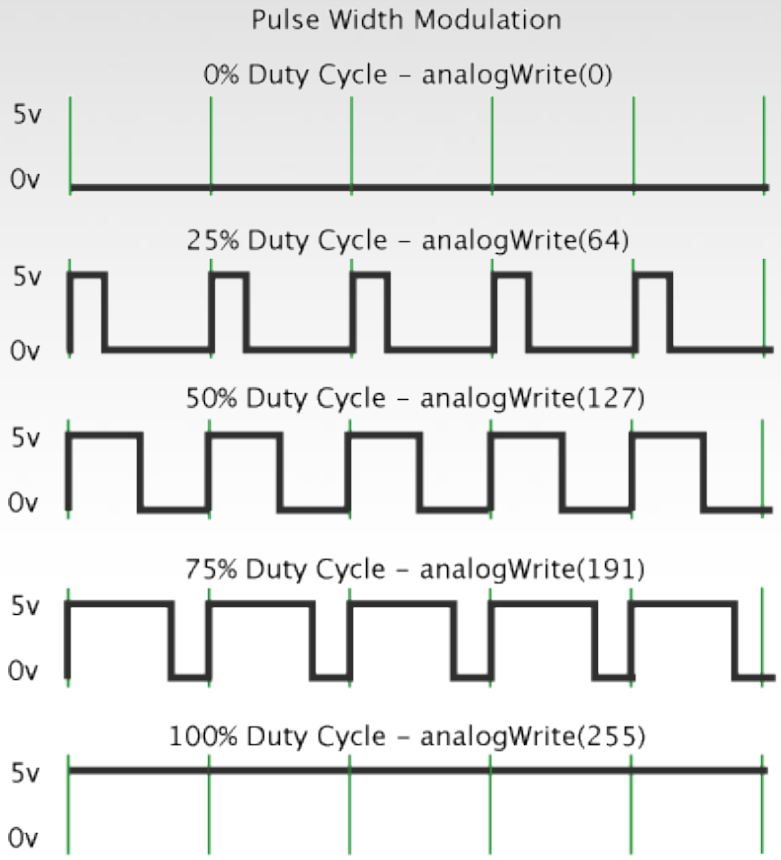
Project # 2 – Fade
New Command!

```
analogWrite(pin, val);
```

pin – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

val – byte/char value (0 – 255).

0 => 0V | 255 => 5V



Fade - Code Review

Select Board

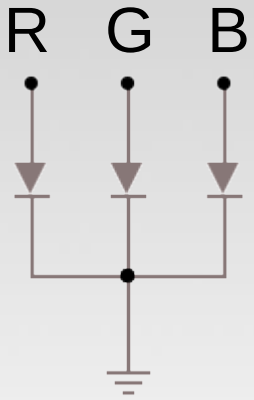
fade.ino

```
1  /*
2   * Gradually fade an LED on and off
3   */
4
5  // assign pin based off circuit
6  int led = 9;
7
8  int brightness = 0;     // holds the value for how bright the led is
9  int fade_amount = 5;    // how much the brightness changes each loop
10
11 // the setup function runs once when you press reset or power the board
12 void setup() {
13   // initialize digital pin 9 as an output.
14   pinMode(led, OUTPUT);
15 }
16
17 // the loop function runs over and over again forever
18 void loop() {
19   analogWrite(led, brightness);               // set the brightness of pin 9
20
21   brightness = brightness + fade_amount;      // update the brightness for the next loop
22
23   if (brightness <= 0 || brightness >= 255) {  // if the LED is all the way off or on
24     fade_amount = fade_amount * -1;           // change the direction of the fade
25   }
26   delay(30);                                  // pause to see the dimming effect
27 }
28
```

Project # 2 - Fade Puzzles

Challenge 2a – Change the rate of the fading in and out. There are at least two different ways to do this – can you figure them out?

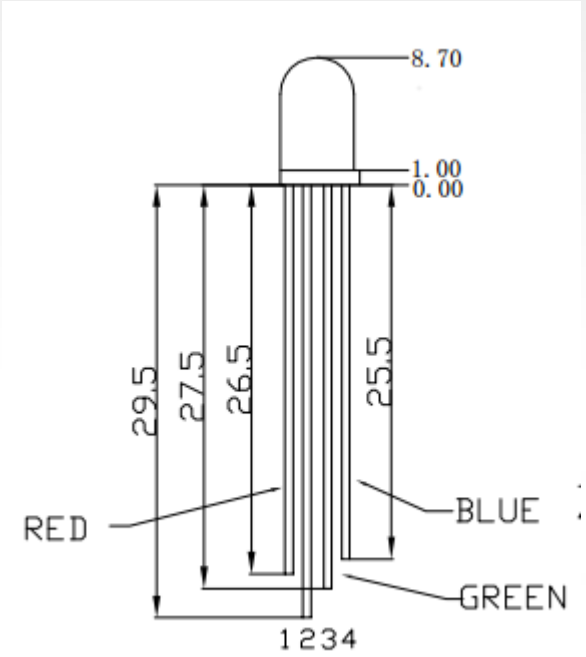
Challenge 2b – Use 2 (or more) LEDs – so that one fades in as the other one fades out.



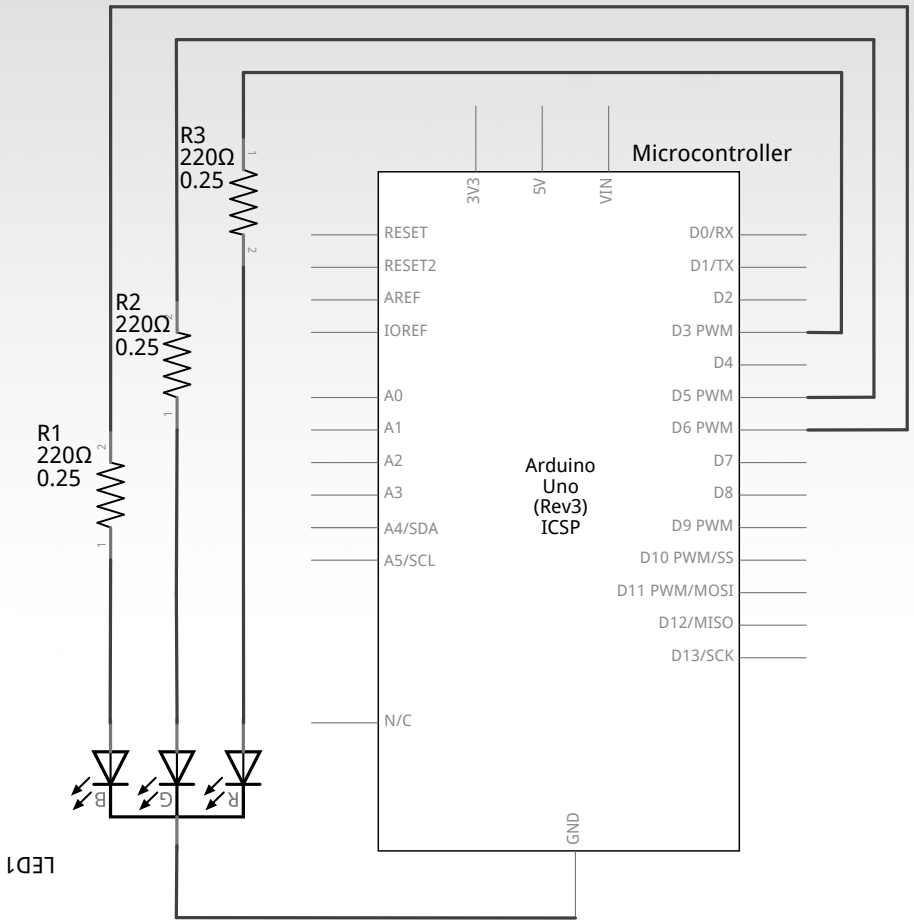
Project # 2.1- Color Mixing
RGB LED

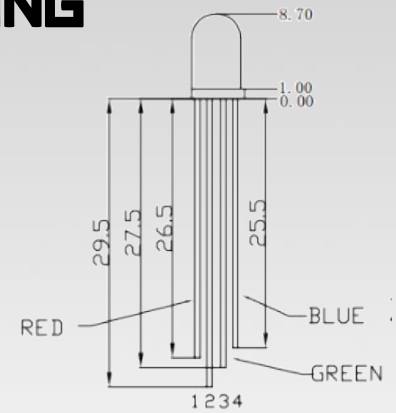
Common Cathode LED

This means the negative side of the LED is all tied to Ground.



Project # 2.1 – Color Mixing Schematic





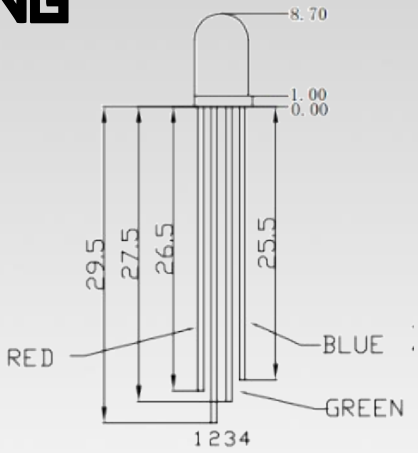
Note: The longest leg of the RGB LED is the Common Cathode. This goes to GND.

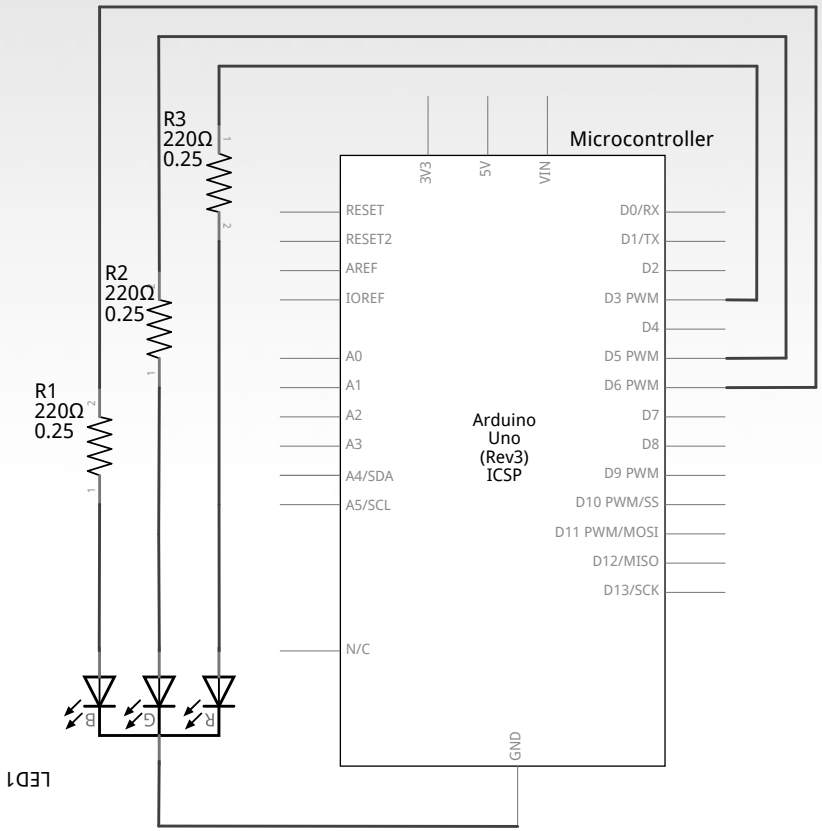
fritzing

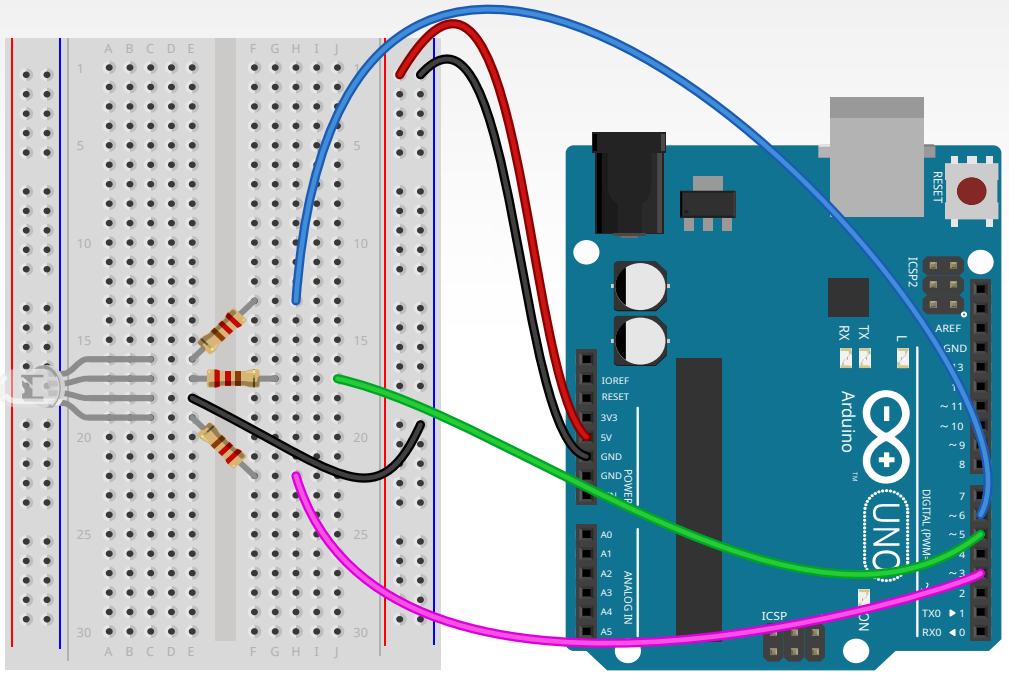
Use pins 5, 6, & 9

Project # 2.1 – Color Mixing

Wiring Diagram







fritzing

fritzing

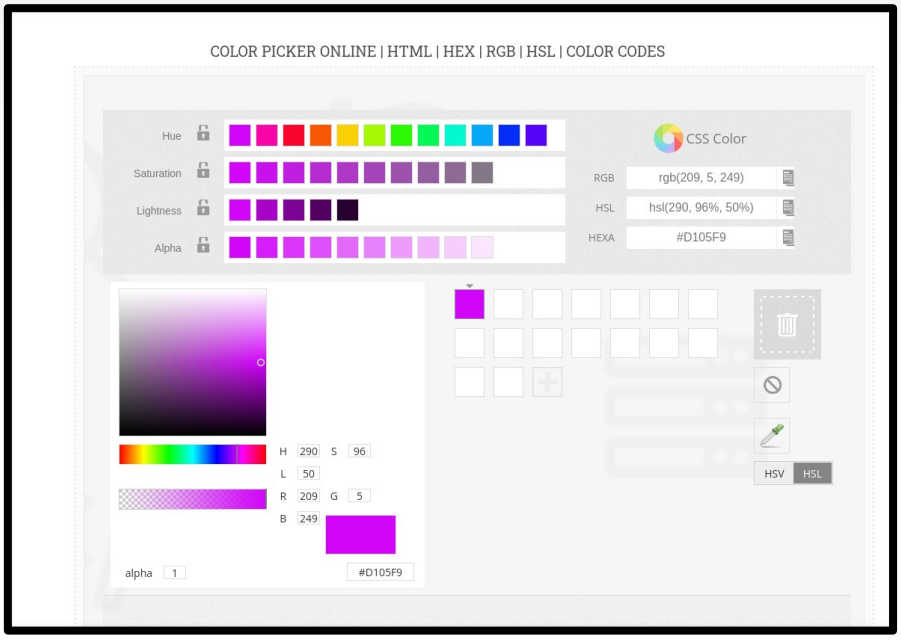
Project # 2.1 - Color Mixing
Code Review

rgb_led.ino

```
/*
 * blend the colors of an RGB LED
 */

// assign pins based off circuit
int redPin = 3;
int greenPin = 5;
int bluePin = 6;

int red_val = 209;
int green_val = 5;
int blue_val = 249;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize all three pins as outputs.
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  // set the different levels of red, green and blue to make your color
  analogWrite(redPin, red_val);
  analogWrite(greenPin, green_val);
  analogWrite(bluePin, blue_val);
}

// the loop function runs over and over again forever
void loop() {
  // nothing to do here
}
```

How many unique colors can you create?

$$\#\ of\ unique\ colors = 256 \cdot 256 \cdot 256$$
$$= 16{,}777{,}216\ colors!$$



Click the “Color Picker” link at microcontrollers.smartypantsconsulting.ltd

Pick out a few colors that you want to try re-creating for a lamp or lighting display...

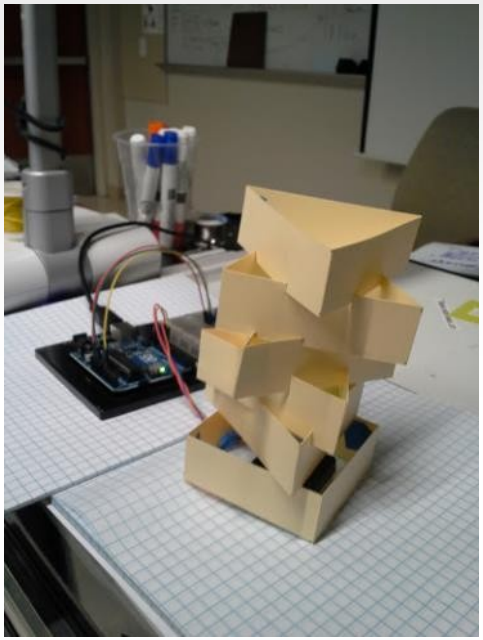
Play around with this and the **`analogWrite()`** command.

Project # 2.1 - Color Mixing
Code Review

rgb_led.ino

```
/*
 * blend the colors of an RGB LED
 */

// assign pins based off circuit
int redPin = 3;
int greenPin = 5;
int bluePin = 6;

int red_val = 209;
int green_val = 5;
int blue_val = 249;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize all three pins as outputs.
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  // set the different levels of red, green and blue to make your color
  analogWrite(redPin, red_val);
  analogWrite(greenPin, green_val);
  analogWrite(bluePin, blue_val);
}

// the loop function runs over and over again forever
void loop() {
  // nothing to do here
}
```

Project: Mood Lamp / Light Sculpture



Special Thanks:

roto

SenseICs