

# PHYSICAL COMPUTING 2: ARDUINO & BASIC OUTPUT

---

CSE 599 Prototyping Interactive Systems | Lecture 3 | April 9

**Jon Froehlich** • Jasper O'Leary (TA)

# TODAY'S LEARNING GOALS

Reinforce some **circuit concepts** from last week

What is **Arduino**? Both hardware and software

How to use **digital output** (*i.e.*, turning on/off an LED using digitalWrite)

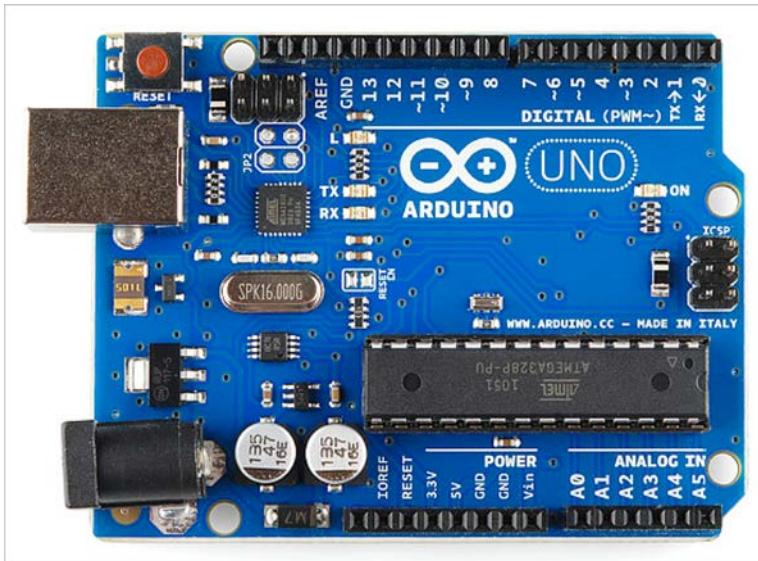
How to **debug** using **Serial Monitor** and **multimeters**

How to use an **RGB LED** (if time)

# WHAT IS ARDUINO?

Arduino is an **open-source electronics platform** based on easy-to-use hardware and software. It's intended to enable designers, artists, hobbyists, engineers, and even children to (rapidly) create new interactive experiences.

## Arduino Hardware



The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

## Arduino Software

A screenshot of the Arduino IDE interface. The window title is "PrintSingleAnalogInputForProcessing | Arduino 1.5.8". The code editor displays the following sketch:// PrintSingleAnalogInputForProcessing
// by Tom Igoe and Scott Fitzgerald
// This example code is in the public domain.
void setup() {
 // initialize the serial communication:
 Serial.begin(9600);
}

void loop() {
 // send the value of analog input 0:
 Serial.println(analogRead(A0));
 // wait a bit for the analog-to-digital converter
 // to stabilize after the last reading:
 delay(2);
}The status bar at the bottom right shows "Arduino Uno on COM6".

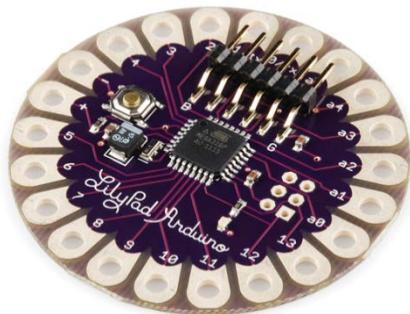
The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

# POPULAR ARDUINO MODELS



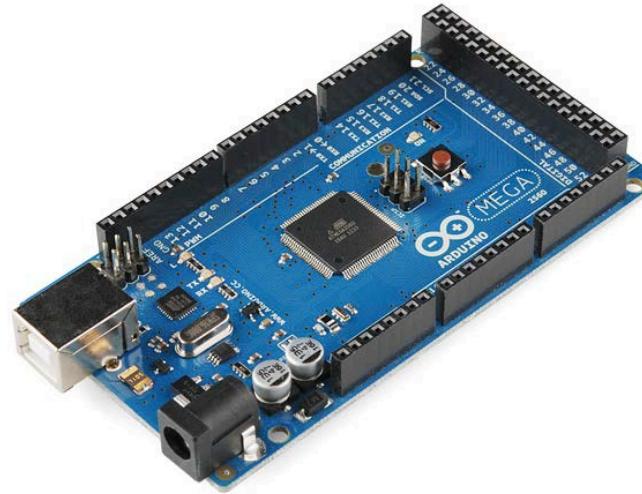
## Arduino Uno (R3)

14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, a reset button



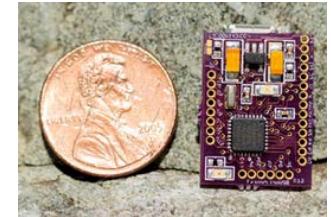
## LilyPad Arduino

Very similar to the Arduino Uno but built for wearable e-textiles. The LilyPad also has its own family of input, output, power, and sensors built specifically for e-textiles.



## Arduino Mega (R3)

The Arduino Mega is like the UNO's big brother. It has lots (54!) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button



## FemtoduinoUSB

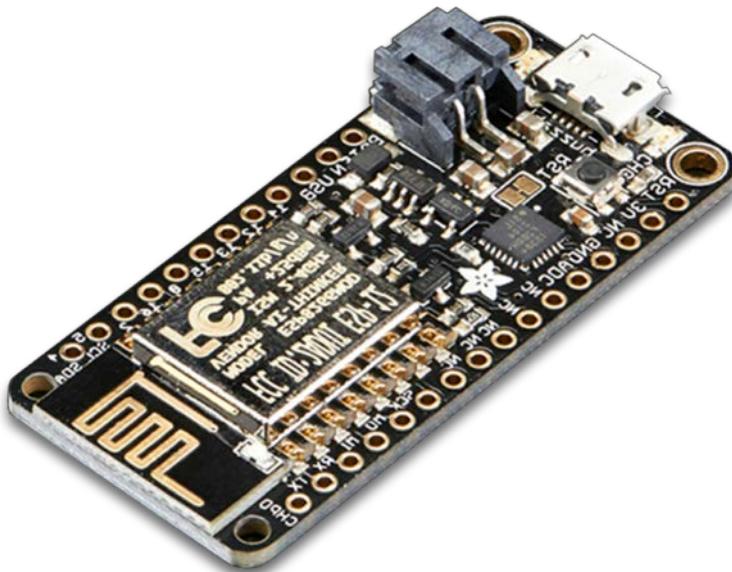
The smallest Arduino board with micro-USB, a voltage regulator, same power and pin count as an Arduino UNO

# NEWEST PLATFORMS: WIFI, BLUETOOTH, FASTER, MORE MEMORY



**REDBEAR DUO**

<https://redbear.cc/product/wifi-ble/redbear-duo.html>



**ADAFRUIT FEATHER**

<https://www.adafruit.com/feather>



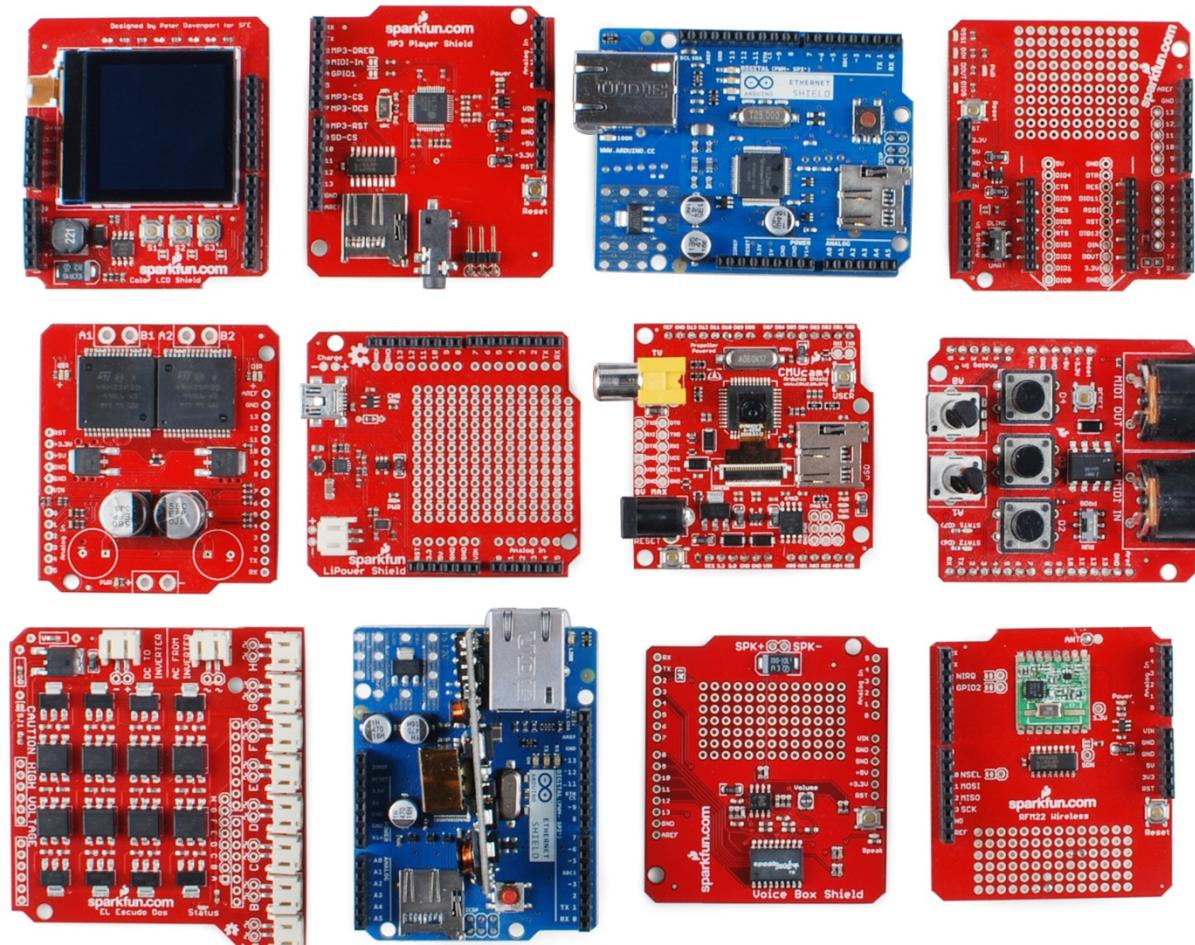
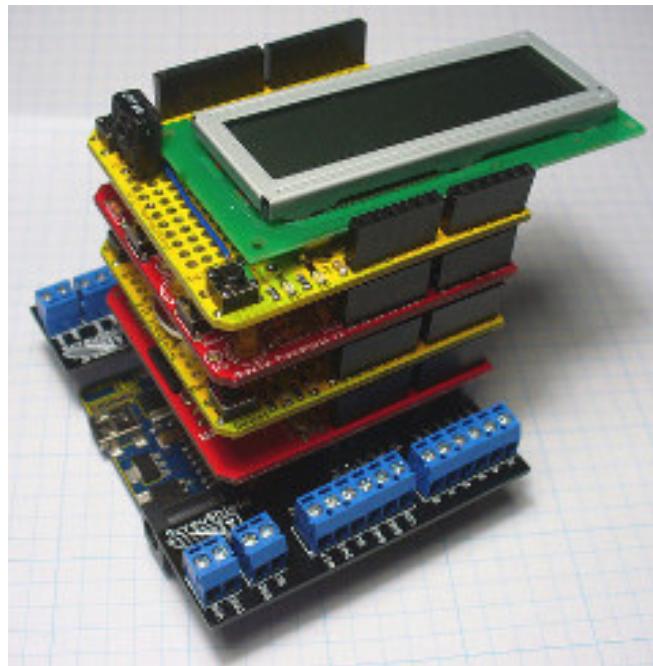
**PARTICLE PHOTON**

<https://www.particle.io/what-is-particle>

ARDUINO

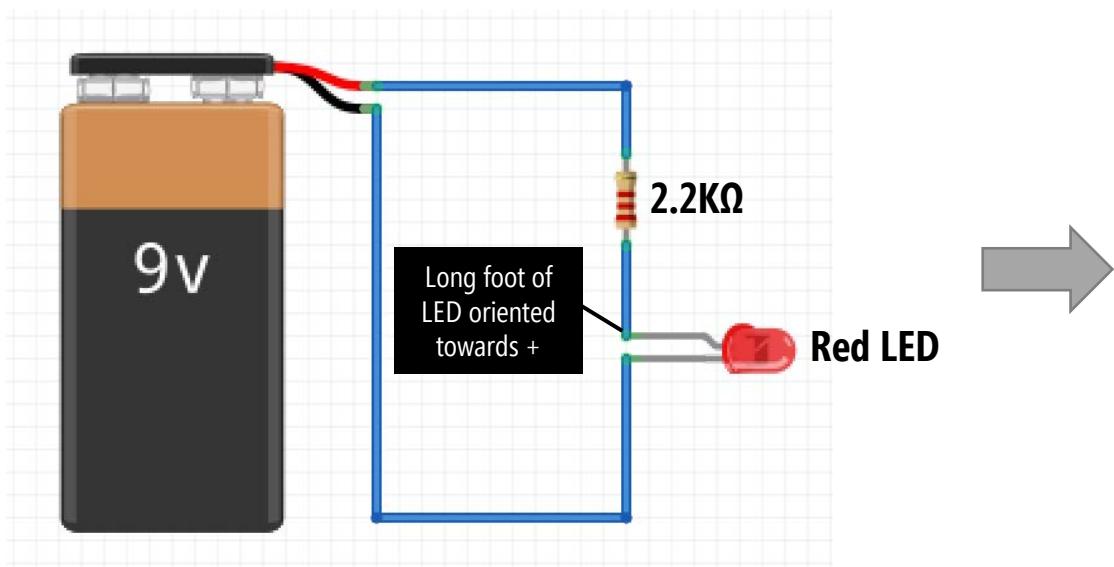
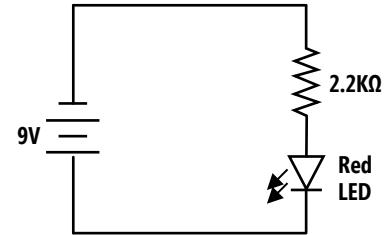
# ARDUINO UNO/LEONARDO STACKABLE SHIELDS

You can purchase stackable shields that give your Arduino more features such as WiFi, controlling motors, cellular modems, controlling LCD screens, etc.

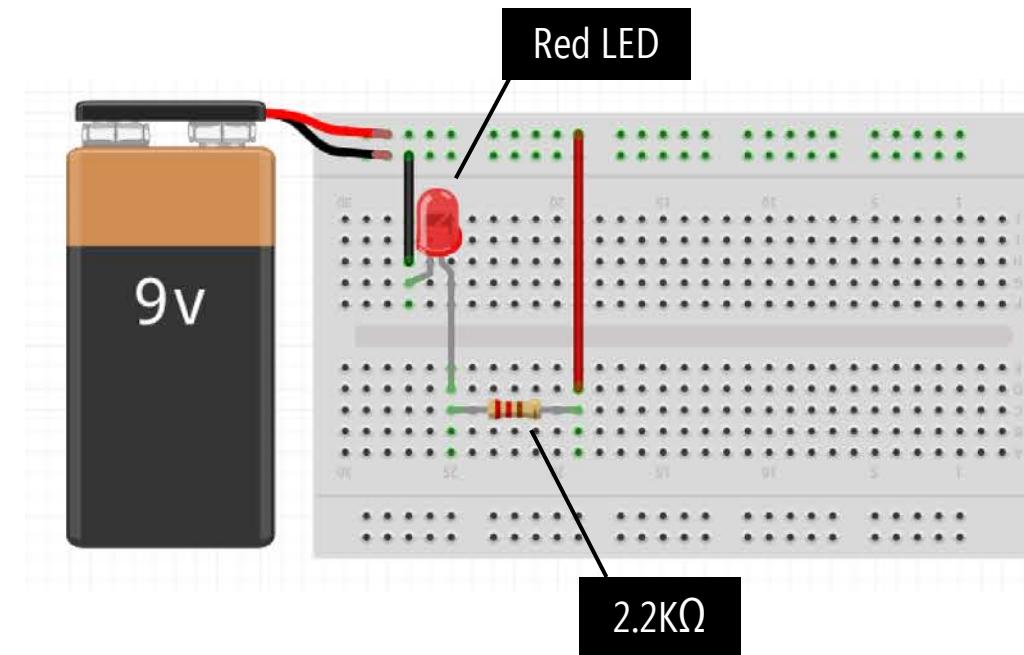


## ACTIVITY

# LAST TIME: WE BUILT A BASIC LED CIRCUIT



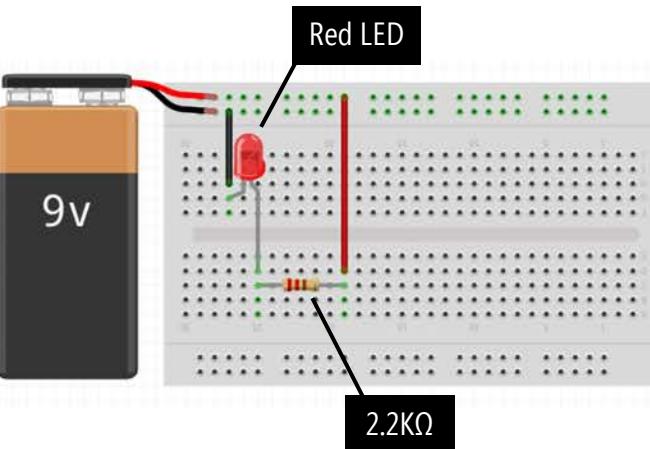
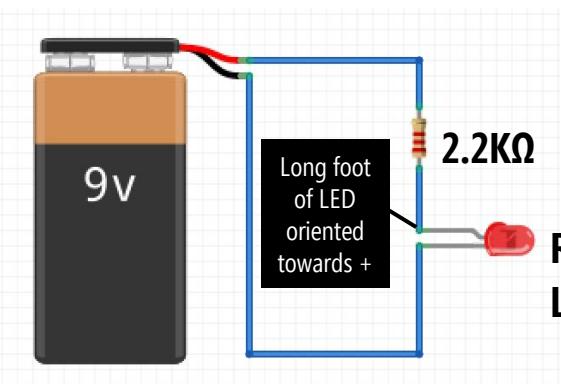
First, we built a free-form LED circuit



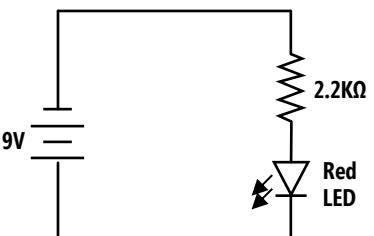
Second, we adapted this circuit to use a breadboard

## ACTIVITY

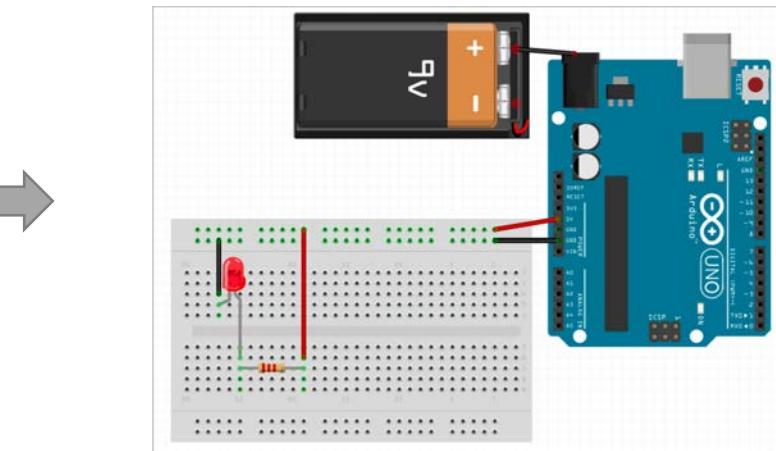
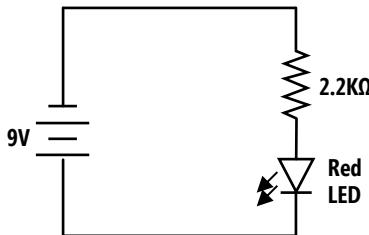
# POWER CIRCUIT VIA ARDUINO 5V



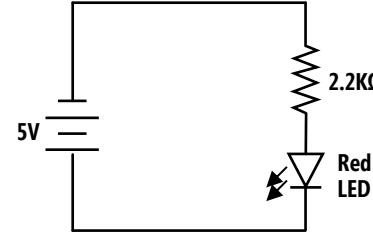
First, we built a free-form LED circuit



Second, we adapted this circuit to use a breadboard

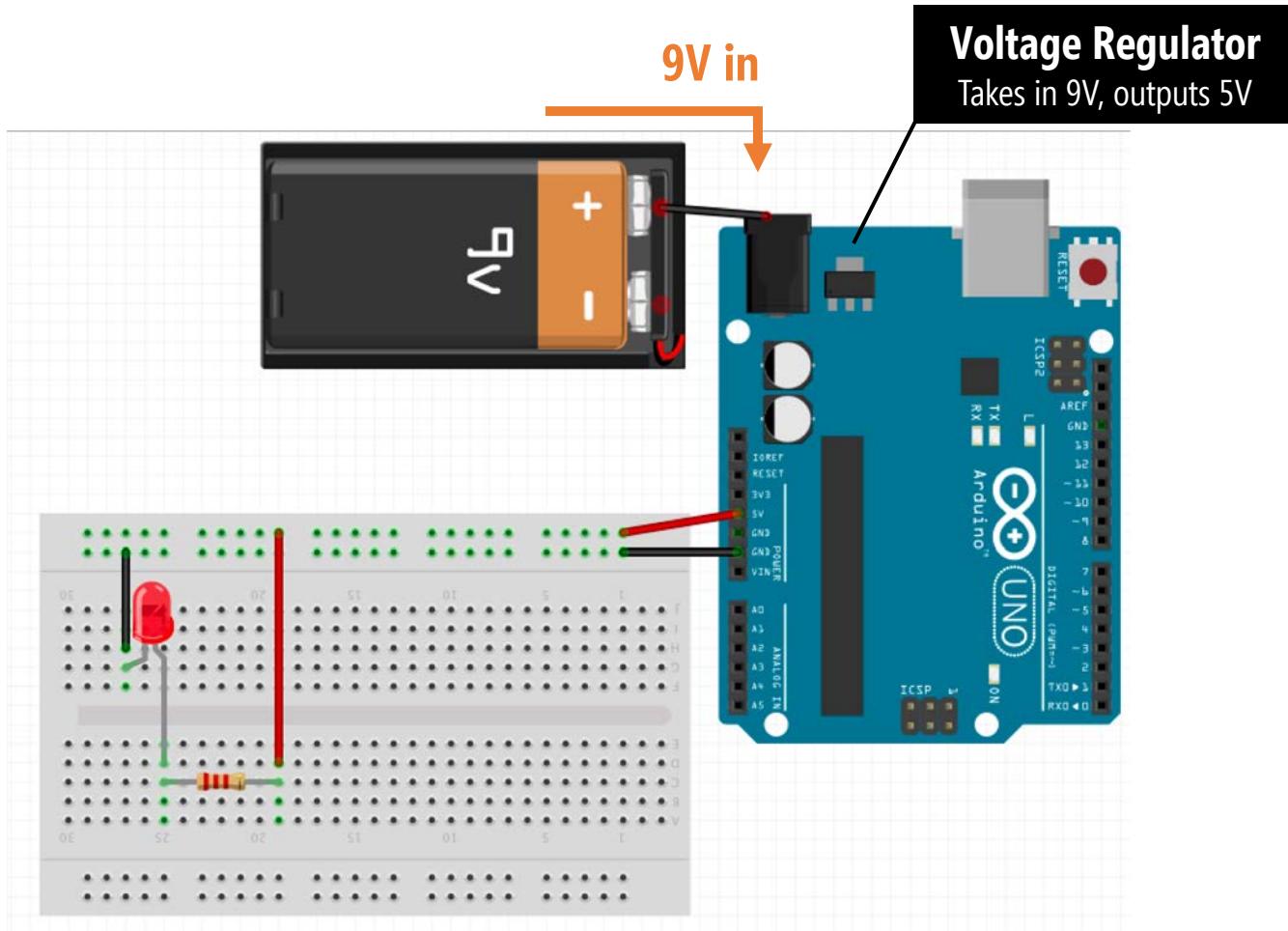


Now, let's power this circuit off of the Arduino with the 9V battery!

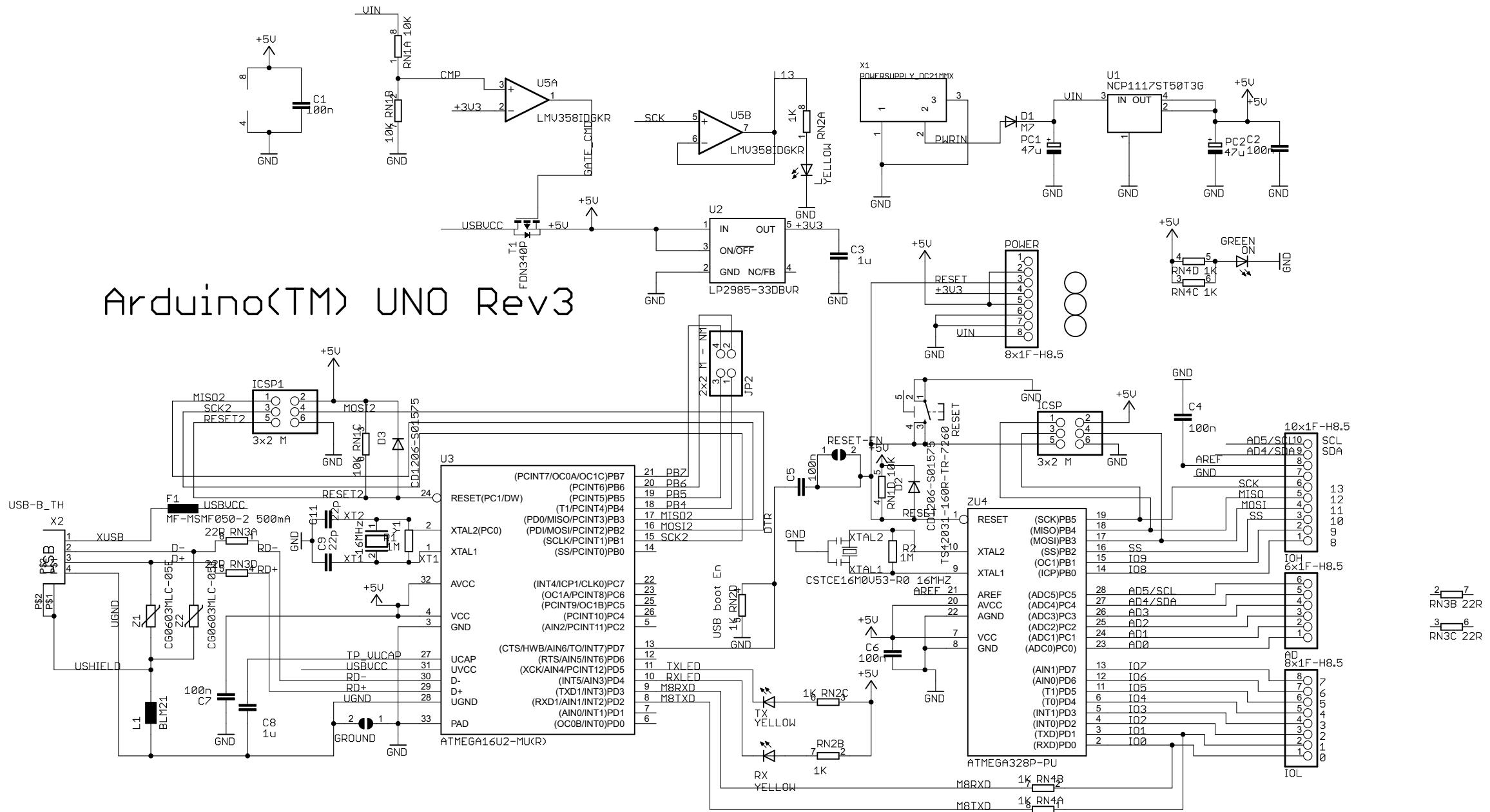


## ACTIVITY

# POWER CIRCUIT VIA ARDUINO 5V

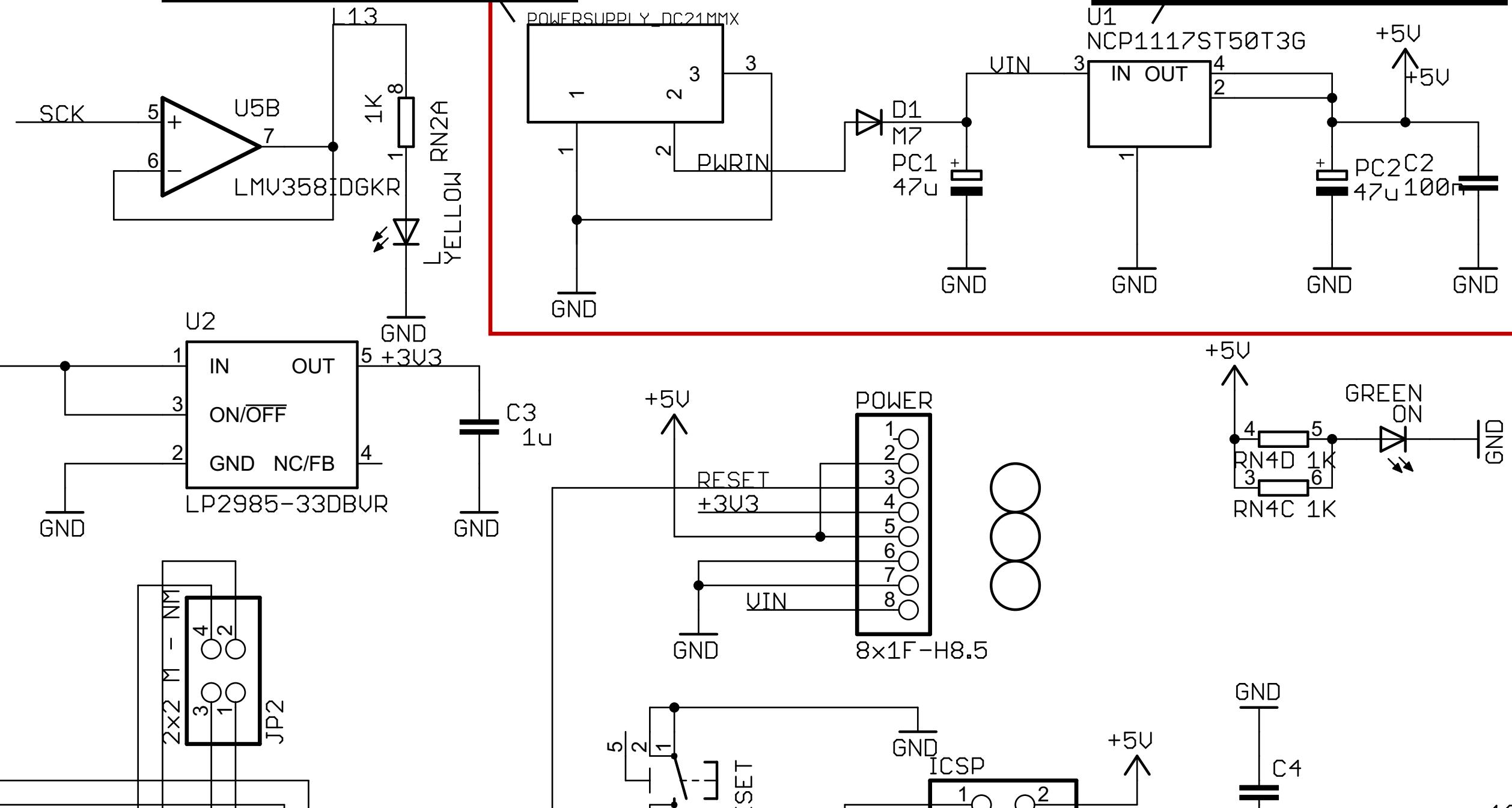


# Arduino(TM) UNO Rev3



# External Power Supply

# NCP1117 Voltage Regulator



ARDUINO

# NCP1117 VOLTAGE REGULATOR

Up to 20V input

Nine **fixed output voltages**,  
including 1.5V, 3.3V, **5V**, and 12V

Also has an **adjustable output version** that can output 1.25V to 18.8V with two external resistors

## NCP1117, NCV1117

### 1.0 A Low-Dropout Positive Fixed and Adjustable Voltage Regulators

The NCP1117 series are low dropout positive voltage regulators that are capable of providing an output current that is in excess of 1.0 A with a maximum dropout voltage of 1.2 V at 800 mA over temperature. This series contains nine fixed output voltages of 1.5 V, 1.8 V, 1.9 V, 2.0 V, 2.5 V, 2.85 V, 3.3 V, 5.0 V, and 12 V that have no minimum load requirement to maintain regulation. Also included is an adjustable output version that can be programmed from 1.25 V to 18.8 V with two external resistors. On chip trimming adjusts the reference/output voltage to within  $\pm 1.0\%$  accuracy. Internal protection features consist of output current limiting, safe operating area compensation, and thermal shutdown. The NCP1117 series can operate with up to 20 V input. Devices are available in SOT-223 and DPAK packages.

#### Features

- Output Current in Excess of 1.0 A
- 1.2 V Maximum Dropout Voltage at 800 mA Over Temperature
- Fixed Output Voltages of 1.5 V, 1.8 V, 1.9 V, 2.0 V, 2.5 V, 2.85 V, 3.3 V, 5.0 V, and 12 V
- Adjustable Output Voltage Option
- No Minimum Load Requirement for Fixed Voltage Output Devices
- Reference/Output Voltage Trimmed to  $\pm 1.0\%$
- Current Limit, Safe Operating and Thermal Shutdown Protection
- Operation to 20 V Input
- NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC-Q100 Qualified and PPAP Capable
- These are Pb-Free Devices

#### Applications

- Consumer and Industrial Equipment Point of Regulation
- Active SCSI Termination for 2.85 V Version
- Switching Power Supply Post Regulation
- Hard Drive Controllers
- Battery Chargers



ON Semiconductor®

[www.onsemi.com](http://www.onsemi.com)



SOT-223  
ST SUFFIX  
CASE 318H



DPAK  
DT SUFFIX  
CASE 369C

#### PIN CONFIGURATION



SOT-223  
(Top View)



DPAK  
(Top View)

Pin: 1. Adjust/Ground  
2. Output  
3. Input

Heatsink tab is connected to Pin 2.

#### ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 12 of this data sheet.

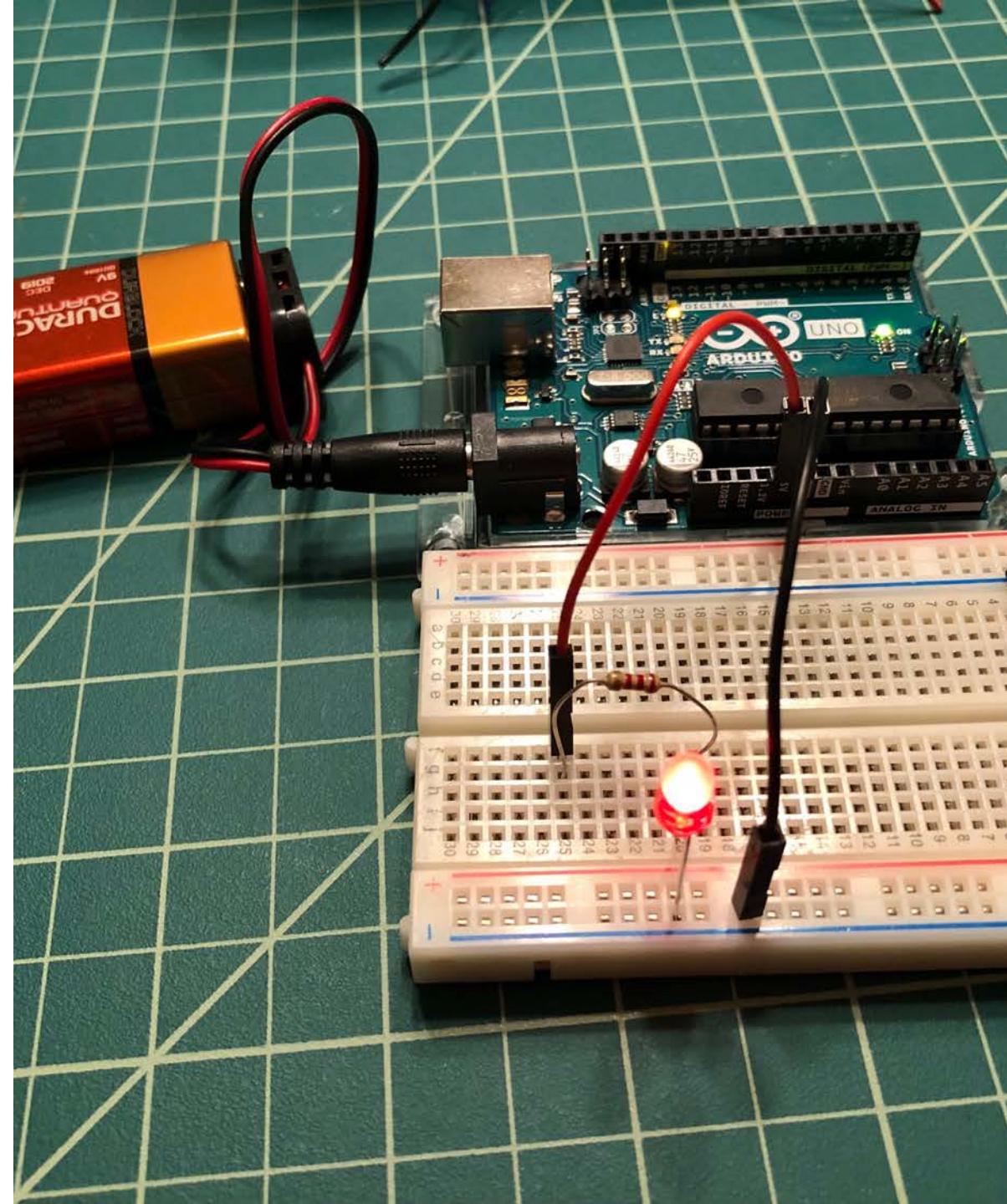
#### DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 14 of this data sheet.

# ACTIVITY: POWER YOUR CIRCUIT VIA ARDUINO FIXED 5V OUTPUT

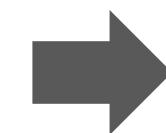
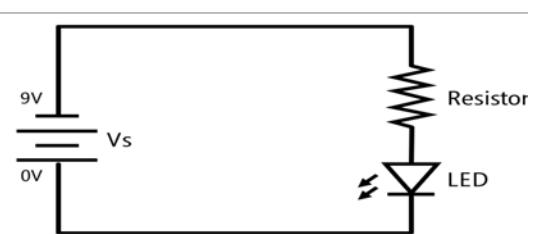
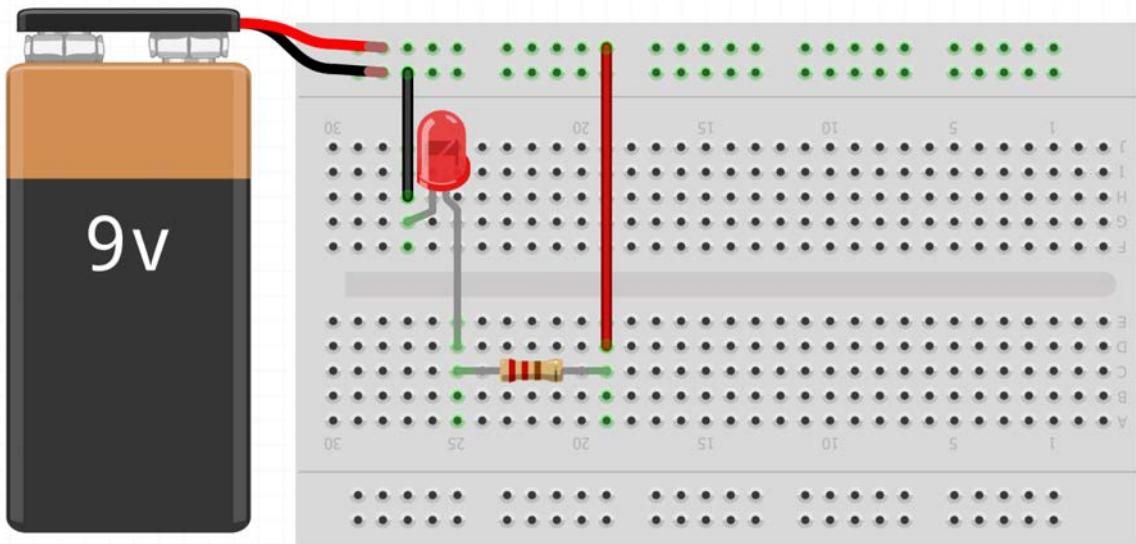
Convert the circuit you previously made with the **9V battery** to, instead, use the **Arduino Uno 5V** and **GND (0V)** connections.

This is a bit of a **toy exercise** given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! ☺

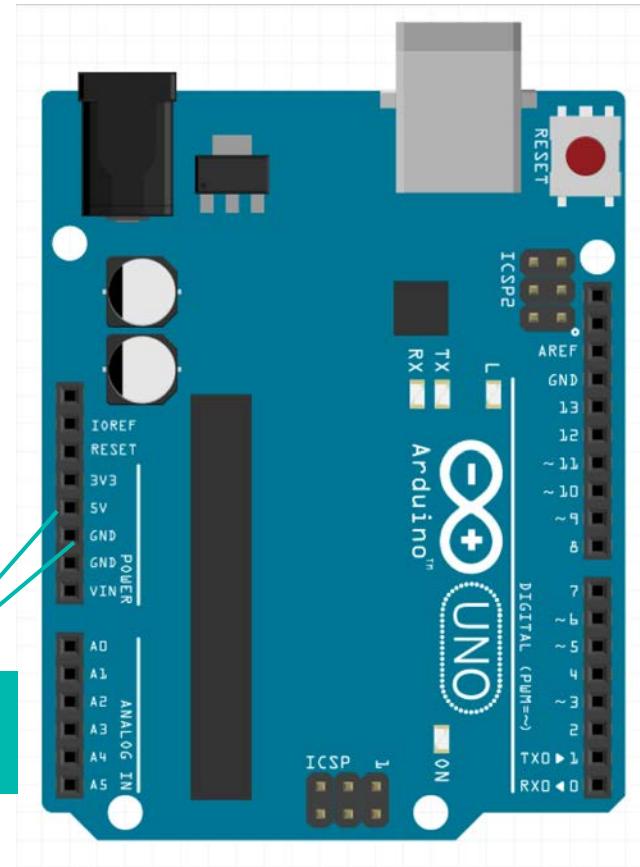


# POWER YOUR CIRCUIT VIA ARDUINO UNO 5V OUTPUT

For our first exercise, we are just going to convert the circuit you previously made with the barrel jack + 9V battery to, instead, use the Arduino Uno 5V and GND connections. This is a bit of a toy exercise given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! 😊

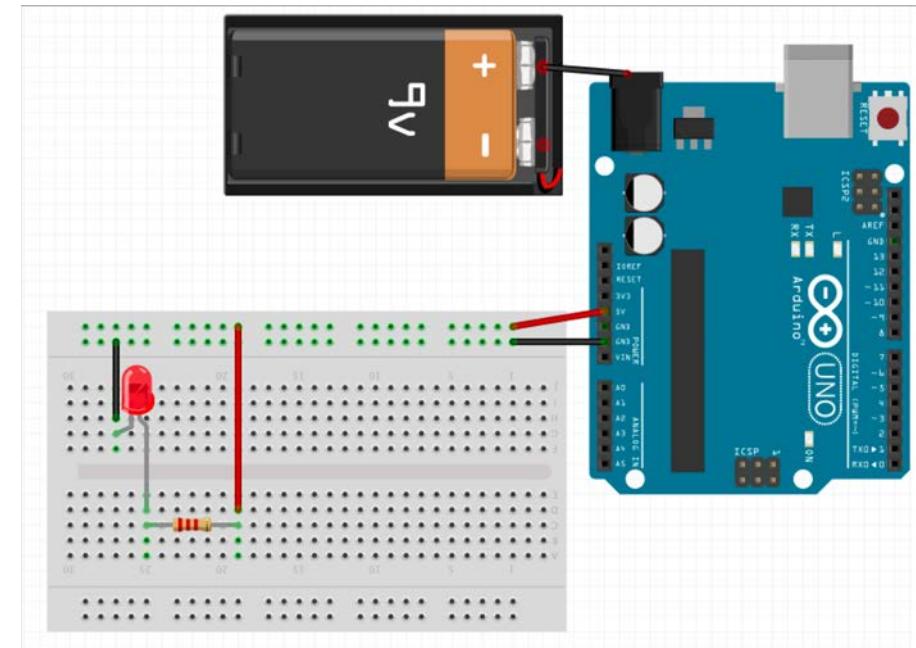
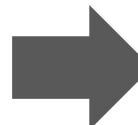
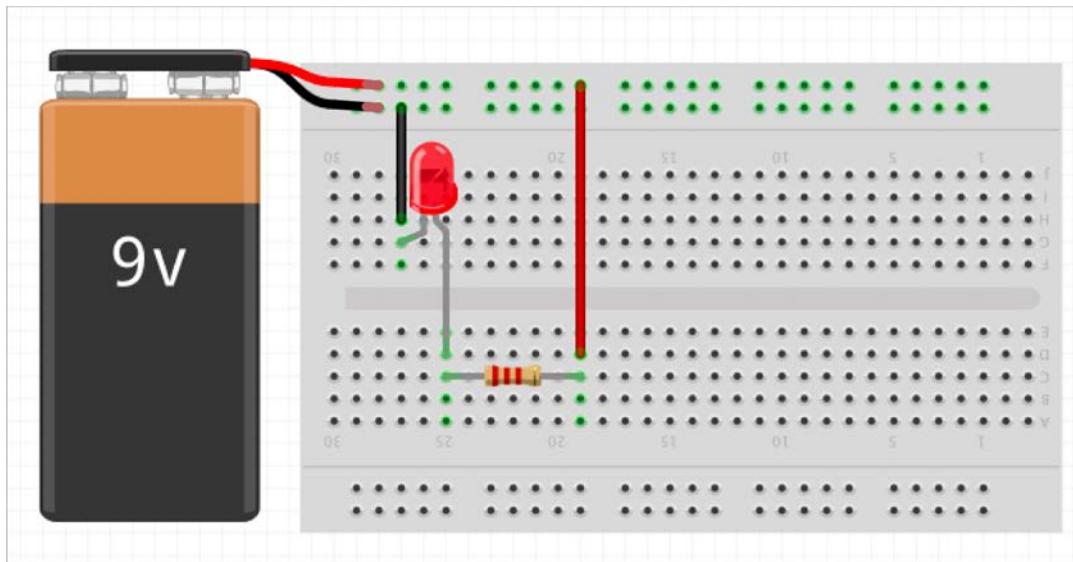


We will want to use this  
**5V pin** and the **GND pin**



# POWER YOUR CIRCUIT VIA ARDUINO UNO 5V OUTPUT

For our first exercise, we are just going to convert the circuit you previously made with the barrel jack + 9V battery to, instead, using the Arduino Uno 5V and GND connections. This is a bit of a toy exercise given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! 😊

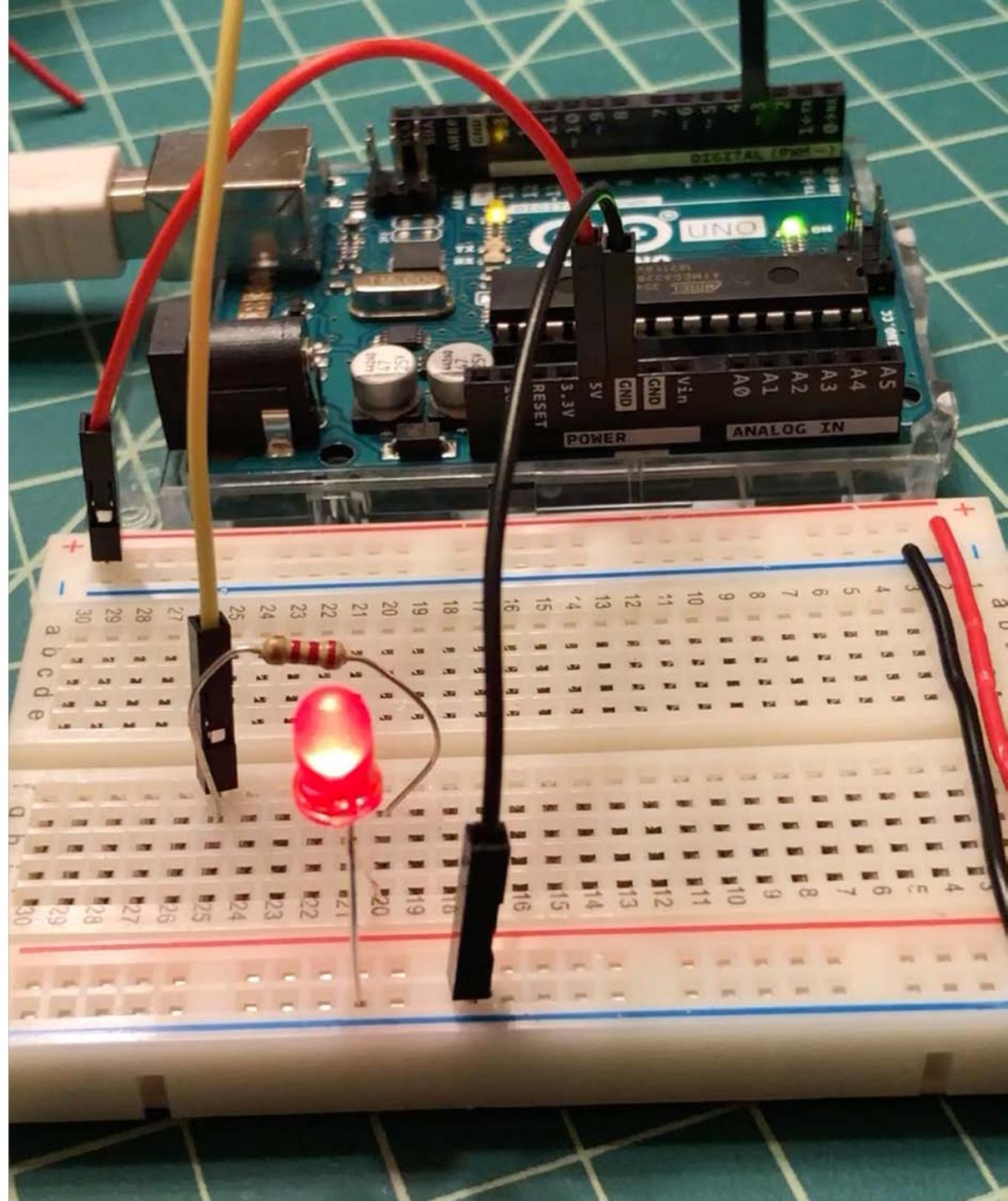


# ACTIVITY: CONTROL THE LED WITH YOUR ARDUINO

OK, now let's do something **more fun** and interactive!

Let's program the **Arduino** to control the LED.

To do this, we have to **introduce the Arduino, the programming framework, and some basics of input/output (I/O)**



# ARDUINO PLATFORM

## Arduino Hardware



The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

## Arduino Software

```
// PrintSingleAnalogInputForProcessing | Arduino 1.5.8
File Edit Sketch Tools Help
PrintSingleAnalogInputForProcessing
// by Tom Igoe and Scott Fitzgerald
// This example code is in the public domain.
void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
}

void loop() {
  // send the value of analog input 0:
  Serial.println(analogRead(A0));
  // wait a bit for the analog-to-digital converter
  // to stabilize after the last reading:
  delay(2);
}
```

The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

# ARDUINO PLATFORM

## Arduino Hardware



The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

## Arduino Software

A screenshot of the Arduino IDE interface. The window title is "PrintSingleAnalogInputForProcessing | Arduino 1.5.8". The code editor contains the following C-like pseudocode:

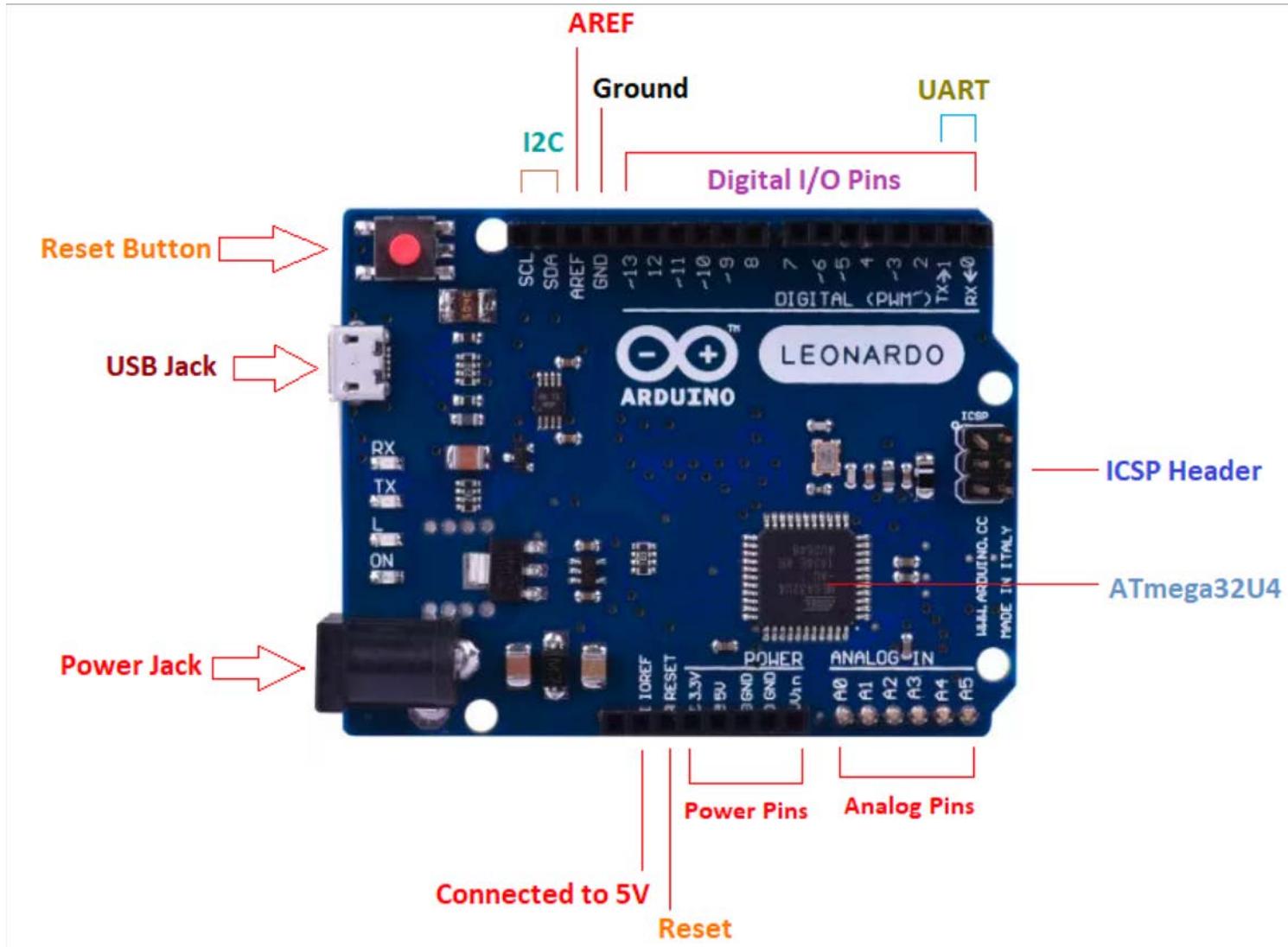
```
// by Tom Igoe and Scott Fitzgerald
// This example code is in the public domain.
void setup() {
    // initialize the serial communication:
    Serial.begin(9600);
}

void loop() {
    // send the value of analog input 0:
    Serial.println(analogRead(A0));
    // wait a bit for the analog-to-digital converter
    // to stabilize after the last reading:
    delay(2);
}
```

The status bar at the bottom right shows "Arduino Uno on COM0".

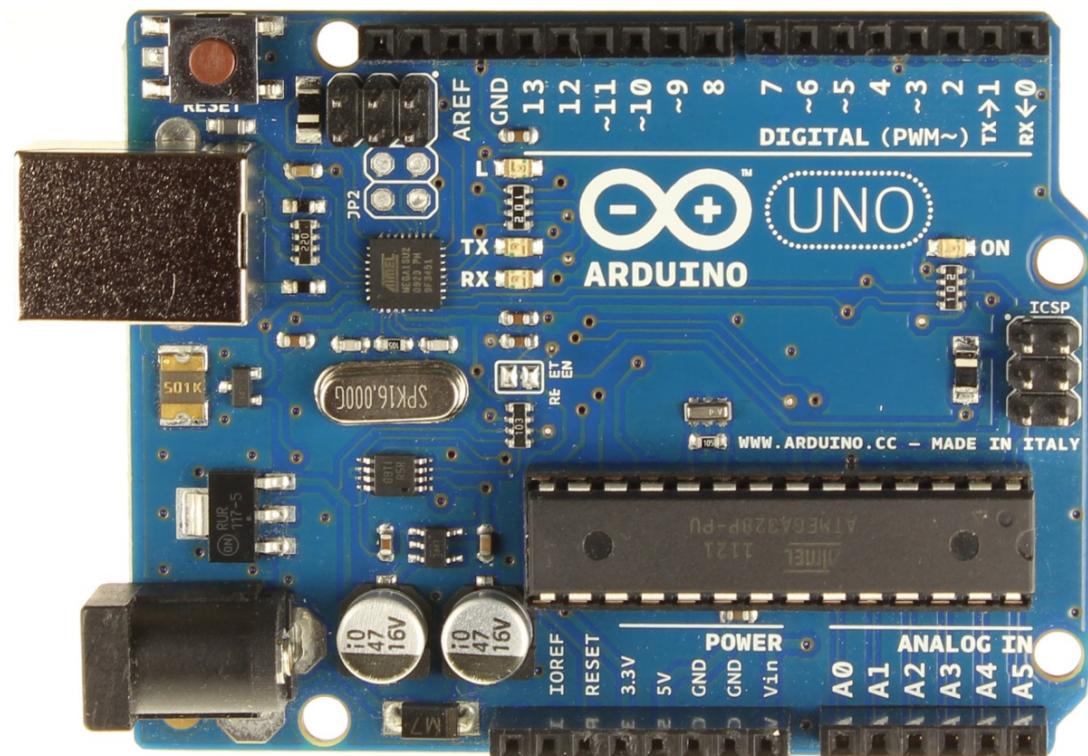
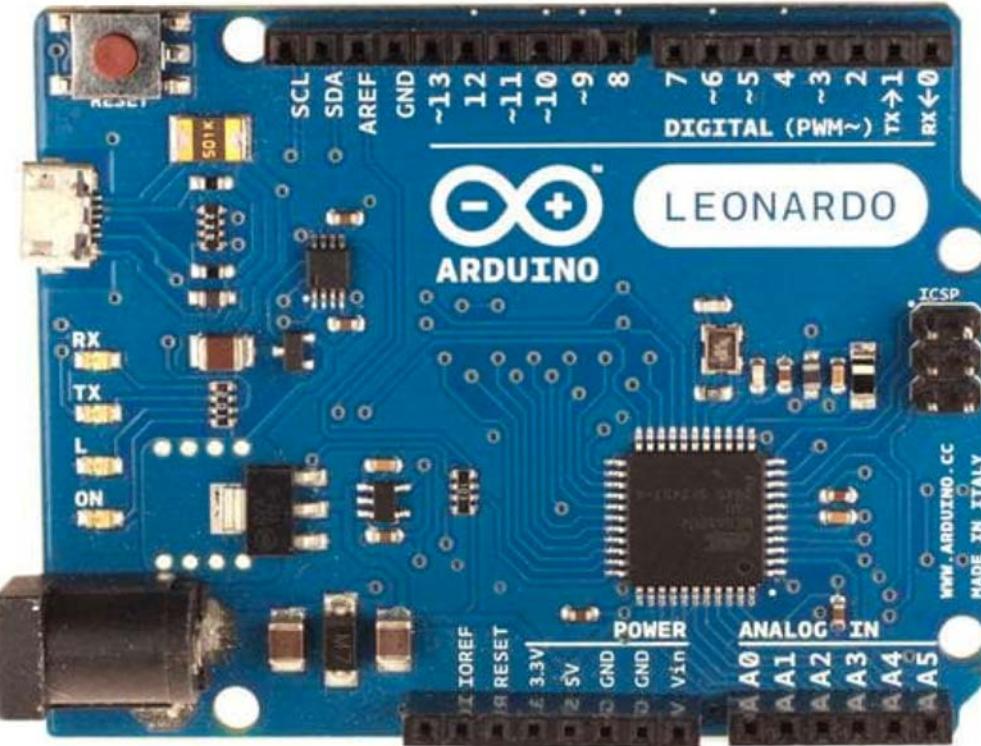
The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

# ARDUINO LEONARDO: TAKE OUT YOUR BOARD



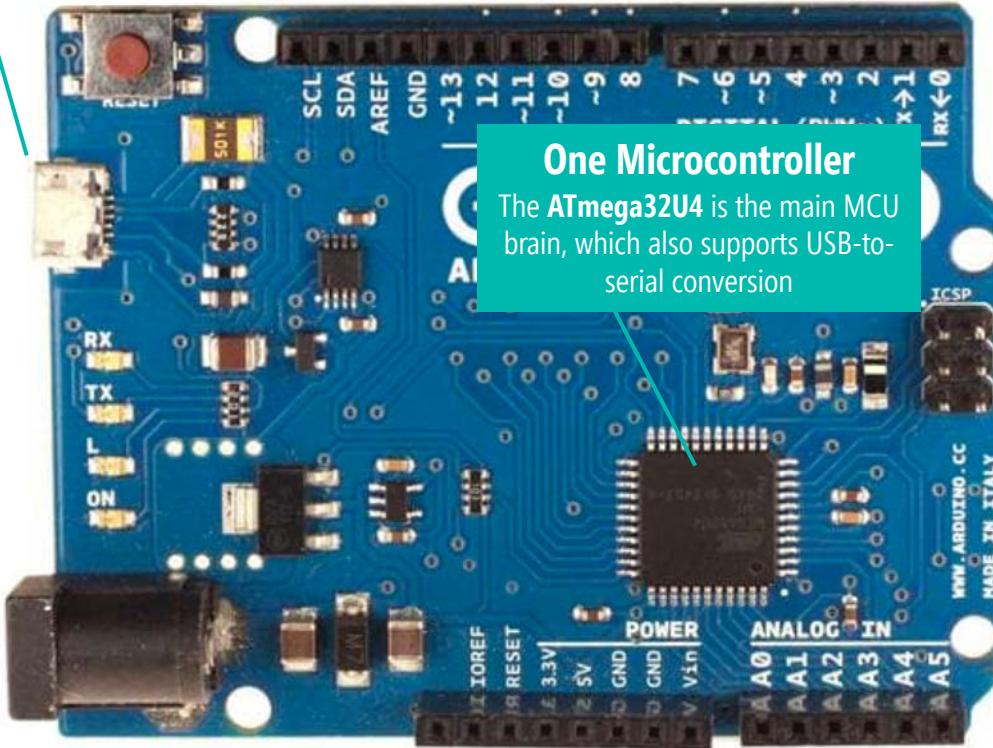
ARDUINO HARDWARE

# LEONARDO VS. UNO



# LEONARDO VS. UNO: SAME FORM FACTOR & I/O PINS

MicroUSB

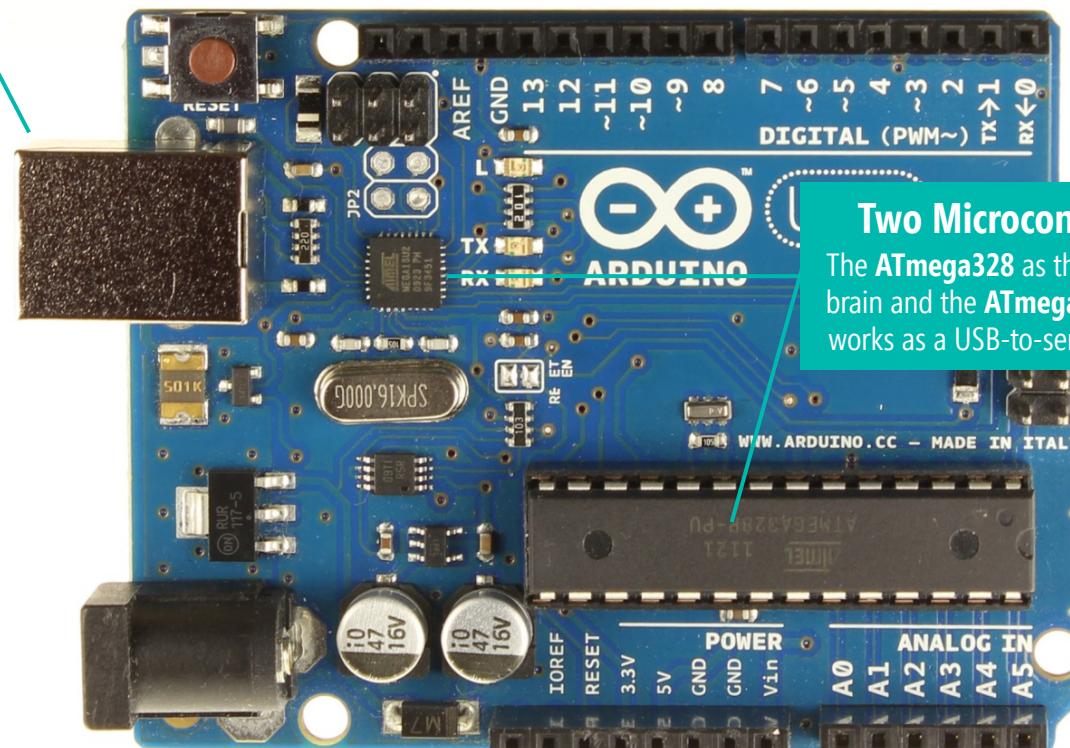


## One Microcontroller

The ATmega32U4 is the main MCU brain, which also supports USB-to-serial conversion

USB A-B

(typically used with printers)

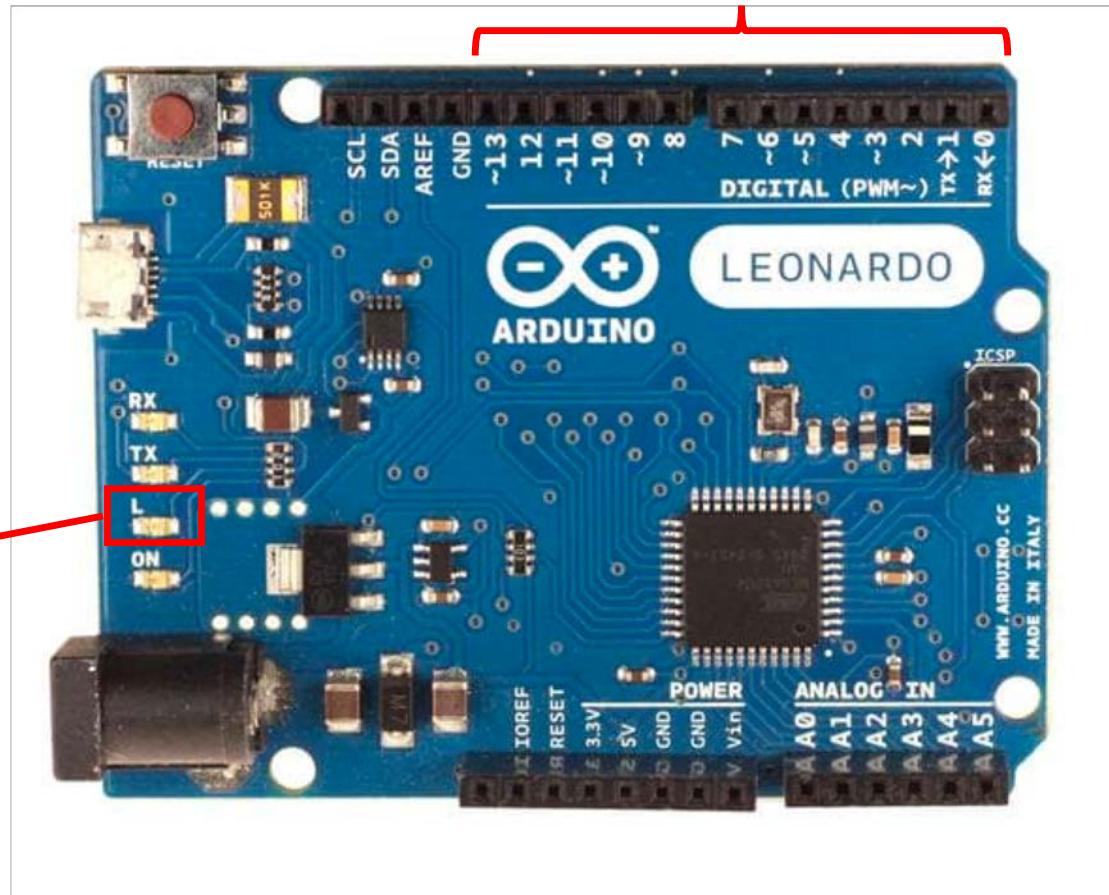


## Two Microcontrollers

The ATmega328 as the main MCU brain and the ATmega16u2, which works as a USB-to-serial converter

# ARDUINO LEONARDO DIGITAL I/O

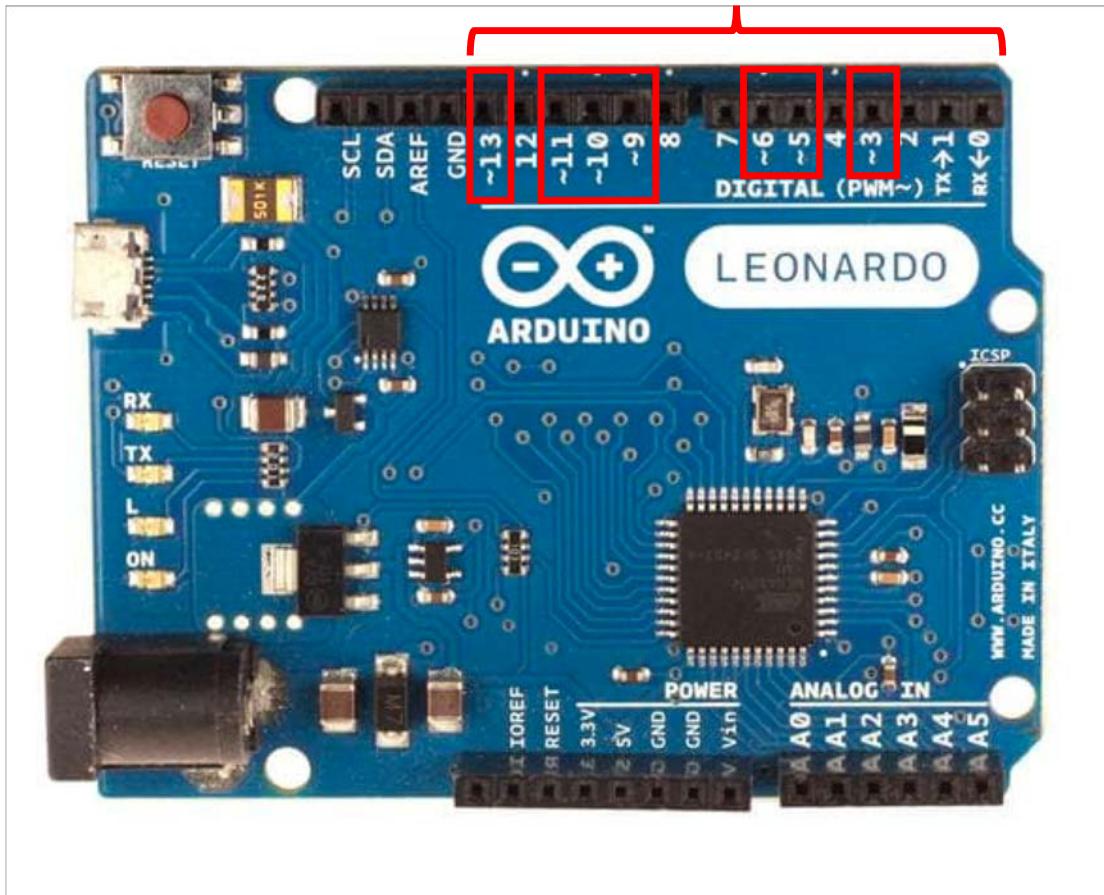
**14 digital I/O pins:** these pins can be used for either input or output (set the mode using the pinMode function). Max output current is 40mA (total across all pins is 200mA)



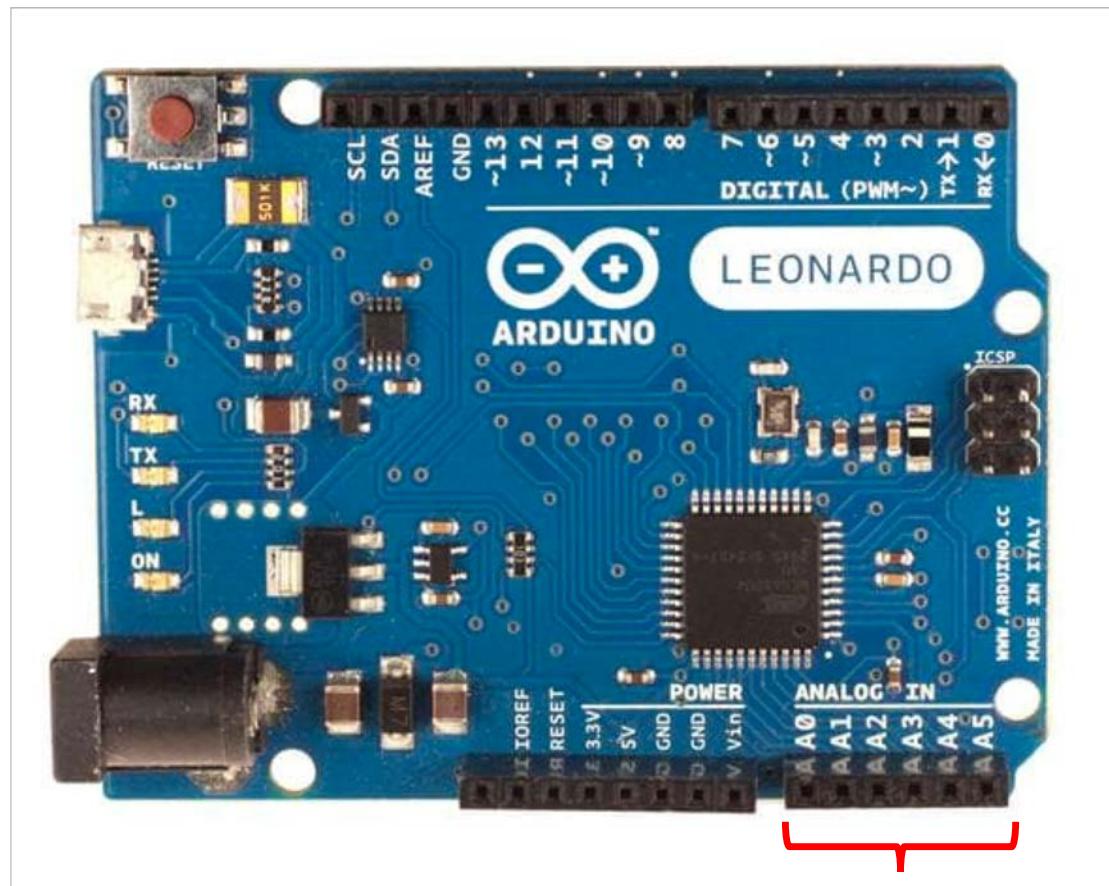
**'L' LED:** The 'L' LED lights up when pin D13 goes HIGH. This is useful for debugging.

# 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.



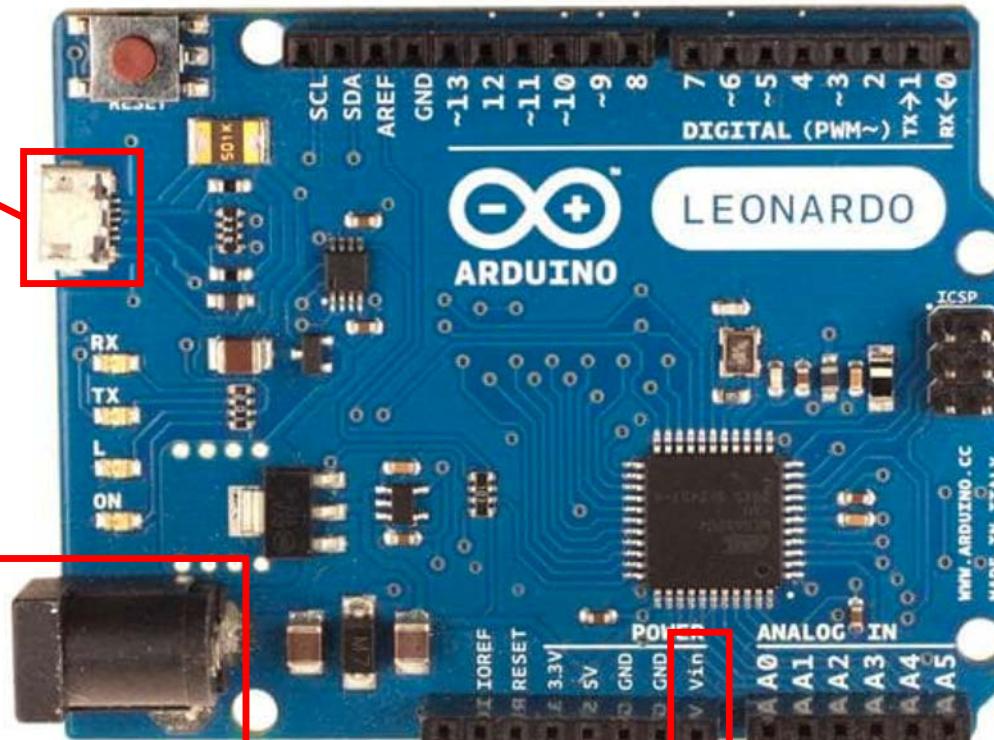
# ARDUINO LEONARDO ANALOG INPUT



**6 analog inputs:** 10 bit resolution (from 0 to 1023). By default, they measure from GND to 5V but this can be changed using the AREF pin and the analogReference function

# ARDUINO LEONARDO: INPUTS FOR POWERING BOARD

**USB:** Supplies 5V either from your laptop or an external power supply



**Barrel Jack:** The recommended input voltage is between 7-12 volts, which can be supplied via an AC-to-DC adapter (wall outlet) or battery.

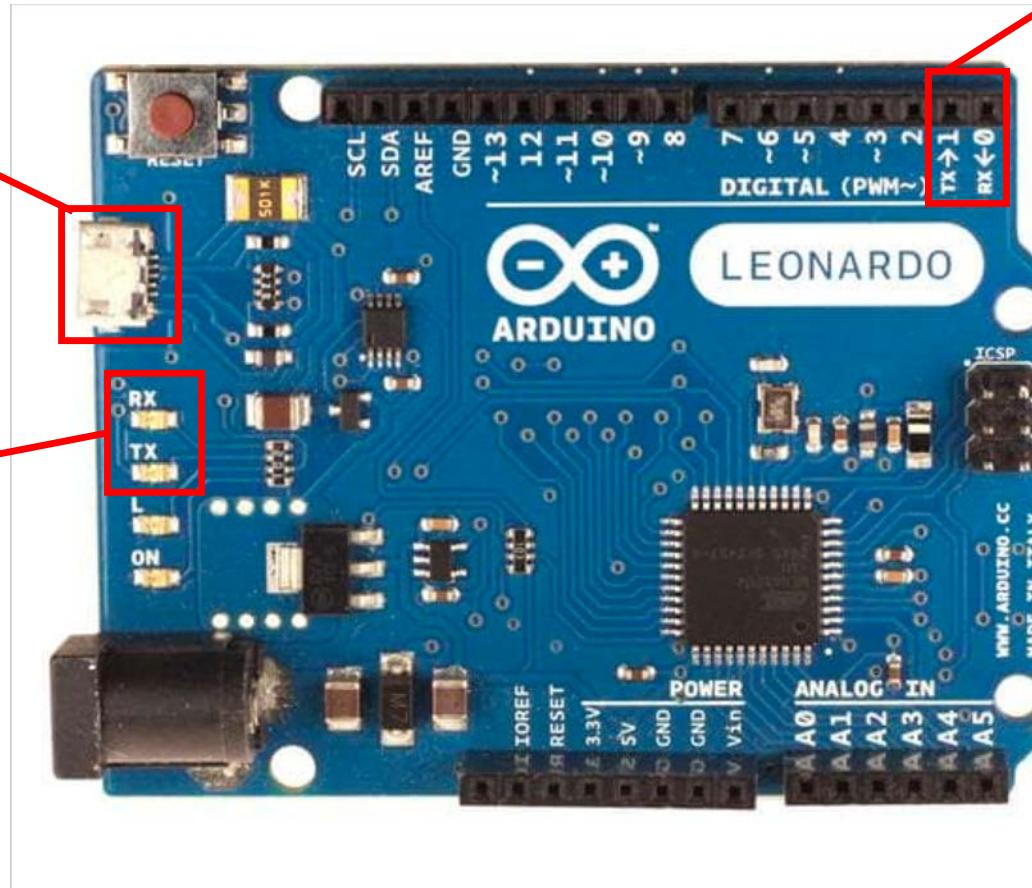


**Vin:** You can also power your Arduino via this pin; however, this is less common. Also, as a warning, though the *barrel jack* is protected by a polarity protection diode, this pin is not. So, if you accidentally connect ground here and your input supply to GND, you will damage your Arduino

# ARDUINO LEONARDO COMMUNICATION

**USB:** The USB allows you to communicate with your computer to program the Arduino via the serial communication protocol and also supplies power (5V).

**TX/RX LEDs:** These LEDs visually indicate when the Arduino is receiving or transmitting data via the serial port (like when you're uploading a program on the board or if you use the Serial library in your Arduino program)



**TX/RX Pins:** TX means 'transmit' and RX means 'receive.' These pins function like normal digital I/O pins *unless* you use the Serial library in your Arduino program.

# ARDUINO PLATFORM

## Arduino Hardware



The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

## Arduino Software

A screenshot of the Arduino IDE interface. The title bar says "PrintSingleAnalogInputForProcessing | Arduino 1.5.8". The code window displays the following C-like pseudocode:

```
PrintSingleAnalogInputForProcessing
// by Tom Igoe and Scott Fitzgerald
//
// This example code is in the public domain.
void setup() {
    // initialize the serial communication:
    Serial.begin(9600);
}

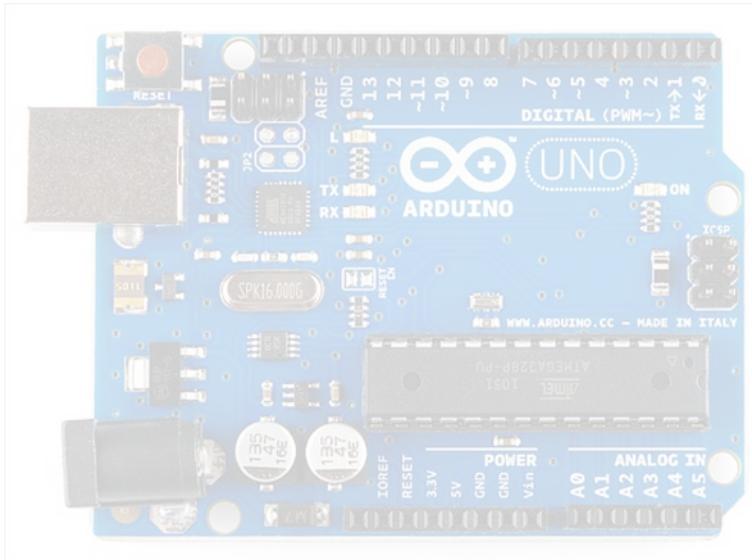
void loop() {
    // send the value of analog input 0:
    Serial.println(analogRead(A0));
    // wait a bit for the analog-to-digital converter
    // to stabilize after the last reading:
    delay(2);
}
```

The status bar at the bottom right shows "Arduino Uno on COM0".

The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

# ARDUINO PLATFORM

## Arduino Hardware



The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

## Arduino Software

```
PrintSingleAnalogInputForProcessing | Arduino 1.5.8
File Edit Sketch Tools Help
PrintSingleAnalogInputForProcessing
// by Tom Igoe and Scott Fitzgerald
// This example code is in the public domain.
void setup() {
  // initialize the serial communication:
  Serial.begin(9600);
}

void loop() {
  // send the value of analog input 0:
  Serial.println(analogRead(A0));
  // wait a bit for the analog-to-digital converter
  // to stabilize after the last reading:
  delay(2);
}
```

The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

# ARDUINO VARIABLES: DATA TYPES & CONVERSION

Arduino data types and constants

Data Types		Conversion	Variable Scope & Qualifiers
String	int	byte()	const
String()	long	char()	scope
array	short	float()	static
bool	unsigned char	int()	volatile
boolean	unsigned int	long()	
byte	unsigned long	word()	
char	void		
double			
float	word		

# ARDUINO STRUCTURE

The elements of Arduino C/C++ code

Sketch

`loop()`

`setup()`

Control Structure

`break`

`continue`

`do...while`

`else`

`for`

`goto`

`if...else`

`return`

`switch...case`

`while`

Arithmetic Operators

`%` (remainder)

`%` (remainder)

`*` (multiplication)

`+` (addition)

`-` (subtraction)

`/` (division)

`=` (assignment operator)

Comparison Operators

`!=` (not equal to)

`<` (less than)

`<=` (less than or equal to)

`==` (equal to)

`>` (greater than)

`>=` (greater than or equal to)

Pointer Access Operators

`&` (reference operator)

`*` (dereference operator)

Bitwise Operators

`&` (bitwise and)

`<<` (bitshift left)

`>>` (bitshift right)

`^` (bitwise xor)

`|` (bitwise or)

`~` (bitwise not)

Boolean Operators

`!` (logical not)

`&&` (logical and)

`||` (logical or)

Compound Operators

`&=` (compound bitwise and)

`*=` (compound multiplication)

`++` (increment)

`+=` (compound addition)

`--` (decrement)

`-=` (compound subtraction)

`/=` (compound division)

`^=` (compound bitwise xor)

`|=` (compound bitwise or)

Further Syntax

`#define` (define)

`#include` (include)

`/* */` (block comment)

`//` (single line comment)

`;` (semicolon)

`{}` (curly braces)

# ARDUINO FUNCTIONS: I/O

For controlling the Arduino board and performing computations

## Digital I/O

`digitalRead()`

`digitalWrite()`

`pinMode()`

## Analog I/O

`analogRead()`

`analogReference()`

`analogWrite()`

## Advanced I/O

`noTone()`

`pulseIn()`

`pulseInLong()`

`shiftIn()`

`shiftOut()`

`tone()`

## External Interrupts

`attachInterrupt()`

`detachInterrupt()`

`Interrupts`

`interrupts()`

`noInterrupts()`

## Communication

`Serial`

`stream`

`USB`

`Keyboard`

`Mouse`

# ARDUINO PROGRAMMING

# ARDUINO CHEAT SHEET

## Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference  
<http://arduino.cc/en/Reference/>

### Structure & Flow

```
Basic Program Structure
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}

Control Structures
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break; // Exit a loop immediately
continue; // Go to next iteration
switch (var) {
  case 1:
    ...
  break;
  case 2:
    ...
  break;
  default:
    ...
}
return x; // x must match return type
return; // For void return type

Function Definitions
ret. type <name>(params) { ... }
e.g. int double(int x) {return x*2;}
```

### Operators

**General Operators**  
= assignment  
+ add - subtract  
\* multiply / divide  
% modulo  
== equal to != not equal to  
< less than or equal to > greater than  
< less than or equal to > greater than or equal to  
&& and || or  
! not

**Compound Operators**  
++ increment  
-- decrement  
+= compound addition  
\*= compound multiplication  
/= compound division  
&= compound bitwise and  
|= compound bitwise or

**Bitwise Operators**  
& bitwise and | bitwise or  
^ bitwise xor ~ bitwise not  
<< shift left >> shift right

**Pointer Access**  
& reference: get a pointer  
\* dereference: follow a pointer

### Variables, Arrays, and Data

Data Types	
boolean	true   false
char	-128 - 127, 'a'-'\$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

**Numeric Constants**

```
123          decimal
0b1111011   binary
0x73         octal - base 8
0x7B         hexadecimal - base 16
123U        force unsigned
123L        force long
123UL       force unsigned long
123.0       force floating point
1.23e6      1.23*106 = 1230000
```

**Qualifiers**

```
static        persists between calls
volatile     in RAM (nice for ISR)
const        read-only
 PROGMEM     in flash
```

**Arrays**

```
char str[8] = {'A', 'r', '!', 'd', '!', 'u', '!', 'n', '!', 'o', '!', '\0'};
// Includes \0 null termination
char str[8] = {'A', 'r', '!', 'd', '!', 'u', '!', 'n', '!', 'o', '\0'};
// Compiler adds null termination
char str[] = "Arduino";
char str[8] = "Arduino";
```

### Built-in Functions

**Pin Input/Output**  
Digital I/O - pins 0-13 A0-A5  
pinMode(pin, [INPUT, OUTPUT, INPUT\_PULLUP])  
digitalRead(pin)  
digitalWrite(pin, [HIGH, LOW])

Analog In - pins A0-A5  
int analogRead(pin)  
analogReference([DEFAULT, INTERNAL, EXTERNAL])

PWM Out - pins 3 5 6 9 10 11  
analogWrite(pin, value)

**Advanced I/O**  
tone(pin, freq\_Hz)  
tone(pin, freq\_Hz, duration\_ms)  
noTone(pin)

shiftOut(dataPin, clockPin, [MSBFIRST, LSBFIRST], value)  
unsigned long pulseIn(pin, [HIGH, LOW])

**Time**  
unsigned long millis()  
// Overflows at 50 days  
unsigned long micros()  
// Overflows at 70 minutes  
delay(msec)  
delayMicroseconds(usec)

### Libraries

**Serial** - comm. with PC or via RX/TX  
begin(long speed) // Up to 115200  
end()  
int available() // #bytes available  
int read() // -1 if none available  
int peek() // Read w/o removing  
flush()  
print(data) println(data)  
write(byte) write(char \* string)  
write(byte \* data, size)  
SerialEvent() // Called if data ready

**SoftwareSerial.h** - comm. on any pin  
SoftwareSerial(rxPin, txPin)  
begin(long speed) // Up to 115200  
listen() // Only 1 can listen  
isListening() // At a time.  
read, peek, print, println, write  
// Equivalent to Serial library

**EEPROM.h** - access non-volatile memory  
byte read(addr)  
write(addr, byte)  
EEPROM[index] // Access as array

**Servo.h** - control servo motors  
attach(pin, [min\_us, max\_us])  
write(angle) // 0 to 180  
writeMicroseconds(us)  
// 1000-2000; 1500 is midpoint  
int read() // 0 to 180  
bool attached()  
detach()

**Wire.h** - I<sup>2</sup>C communication  
begin() // Join a master  
begin(addr) // Join a slave @ addr  
requestFrom(address, count)  
beginTransmission(addr) // Step 1  
send(byte) // Step 2  
send(char \* string)  
send(byte \* data, size)  
endTransmission() // Step 3  
int available() // #bytes available  
byte receive() // Get next byte  
onReceive(handler)  
onRequest(handler)



The Arduino Cheat Sheet handout we provided in class should be a useful reference to help you get started (until you've internalized the syntax, common functions, etc.).

# PROGRAMMING ARDUINO!

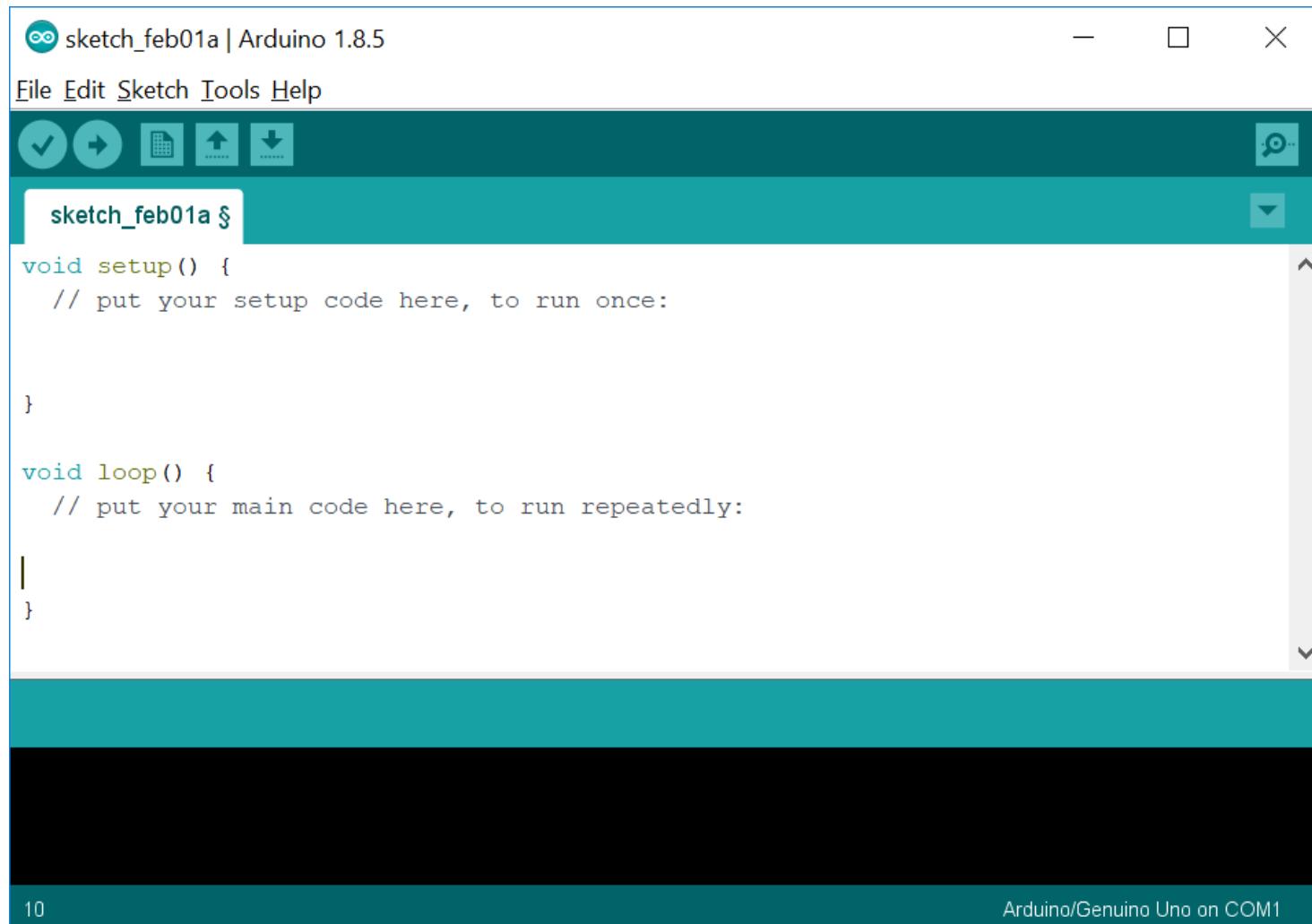
All Arduino programs are comprised of two areas: setup and loop.

## SETUP

Called once at the very beginning of execution. Use this for initialization

## LOOP

Called over and over again as fast as possible by the microcontroller. As soon as your code completes here, loop() will be called again forever



The screenshot shows the Arduino IDE interface with the title bar "sketch\_feb01a | Arduino 1.8.5". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for save, run, upload, and download. The main code editor window displays the following code:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:
}
```

The status bar at the bottom right indicates "Arduino/Genuino Uno on COM1".

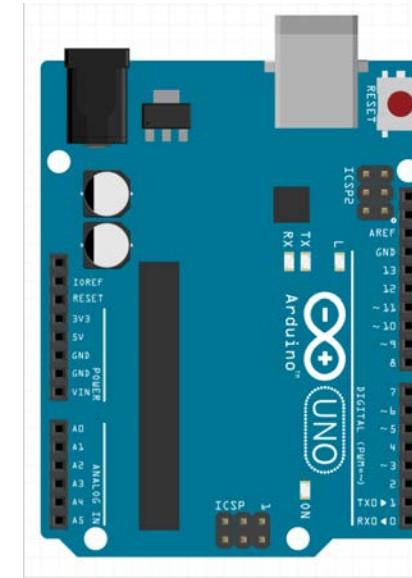
ACTIVITY: LEARN DIGITAL OUTPUT

# UPLOAD PROGRAM TO ARDUINO

To test your program, you compile and upload to Arduino



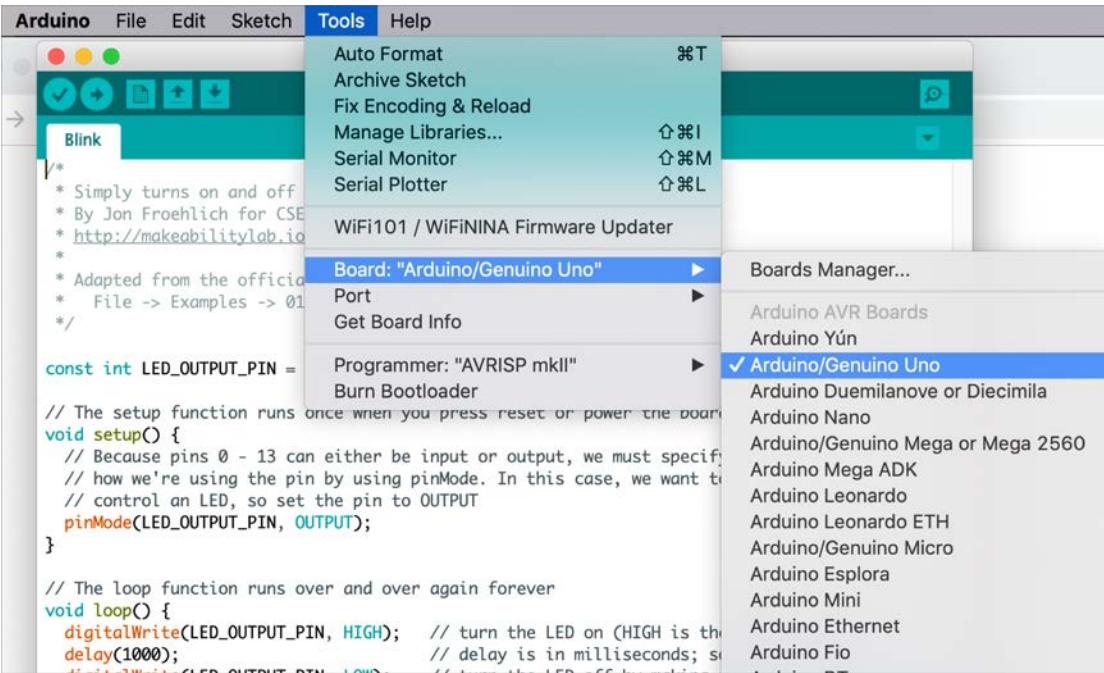
USB Cable



Once the program is loaded onto the Arduino, you can unplug it and run on battery

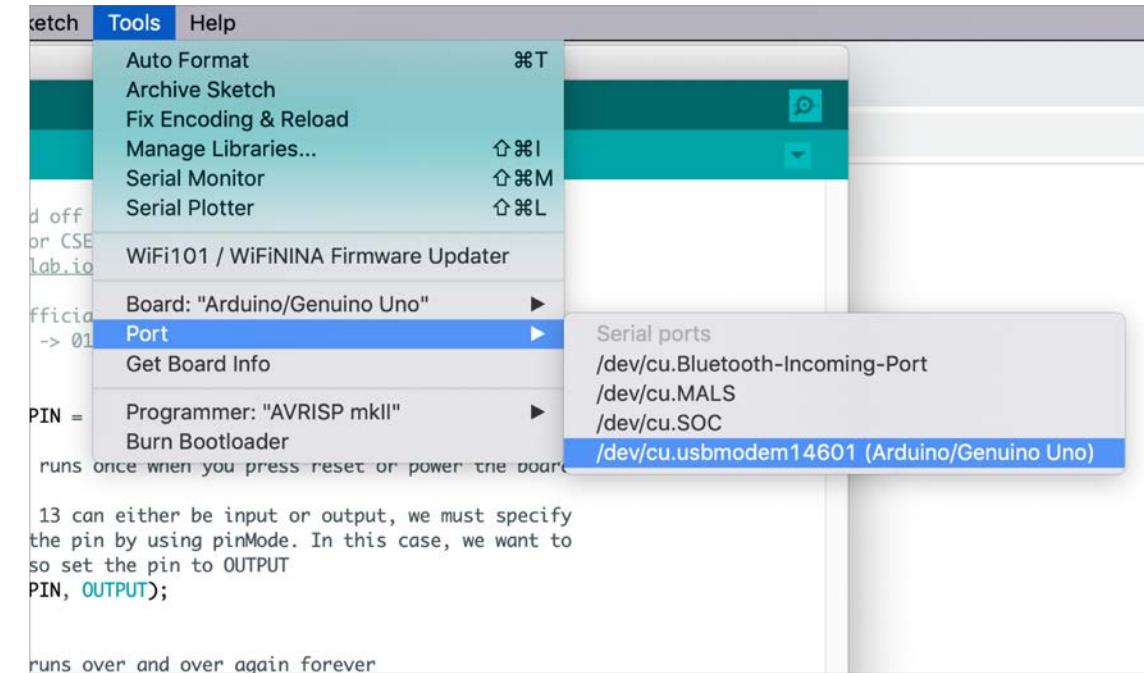
# CONFIGURE THE IDE FOR YOUR ARDUINO BOARD

You should only need to do this once (unless you switch USB ports)



## 1. SELECT THE ARDUINO PLATFORM

Go to Tools -> Board to select the Arduino board that you have plugged into USB



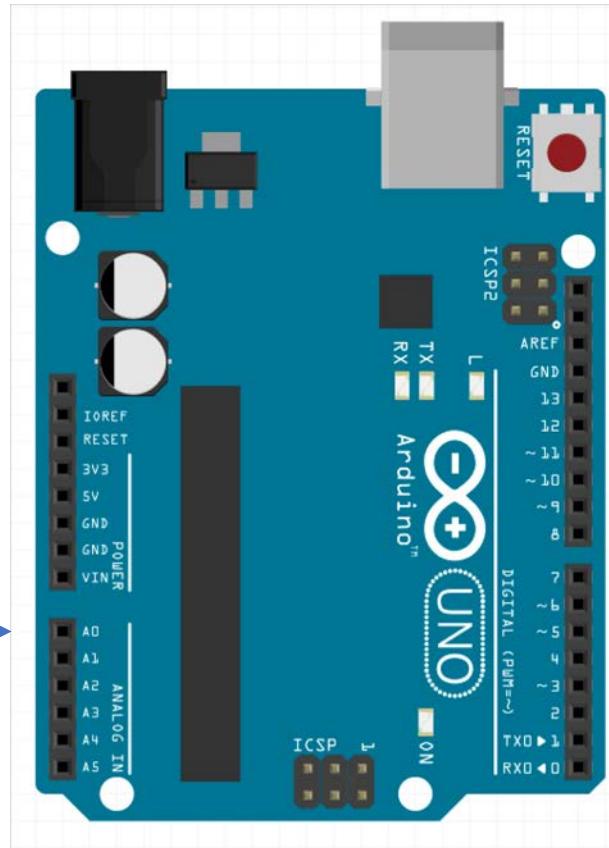
## 2. SELECT THE PORT

Go to Tools -> Port to select the port that the Arduino is using

## 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.



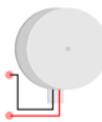
### Digital Output on pin 8

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



### Analog Output on pin 3

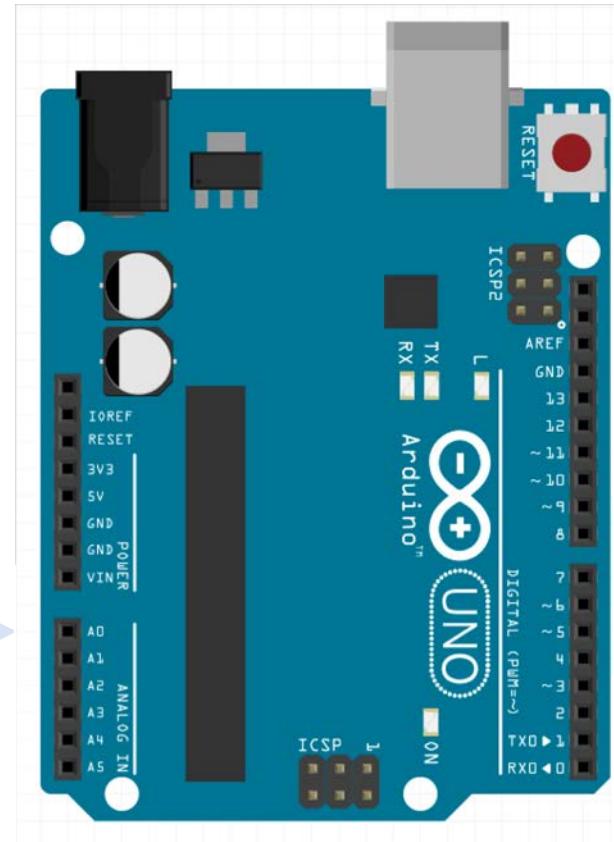
Writes out any value between 0V or 5V using `analogWrite` function. In this case, vibrating a motor (where strength of vibration proportional to voltage). Only pins with a tilde ~ can be analog outputs.



## INTRODUCTION TO I/O

# INTRO TO DIGITAL OUTPUT: WRITING 0V AND 5V

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.



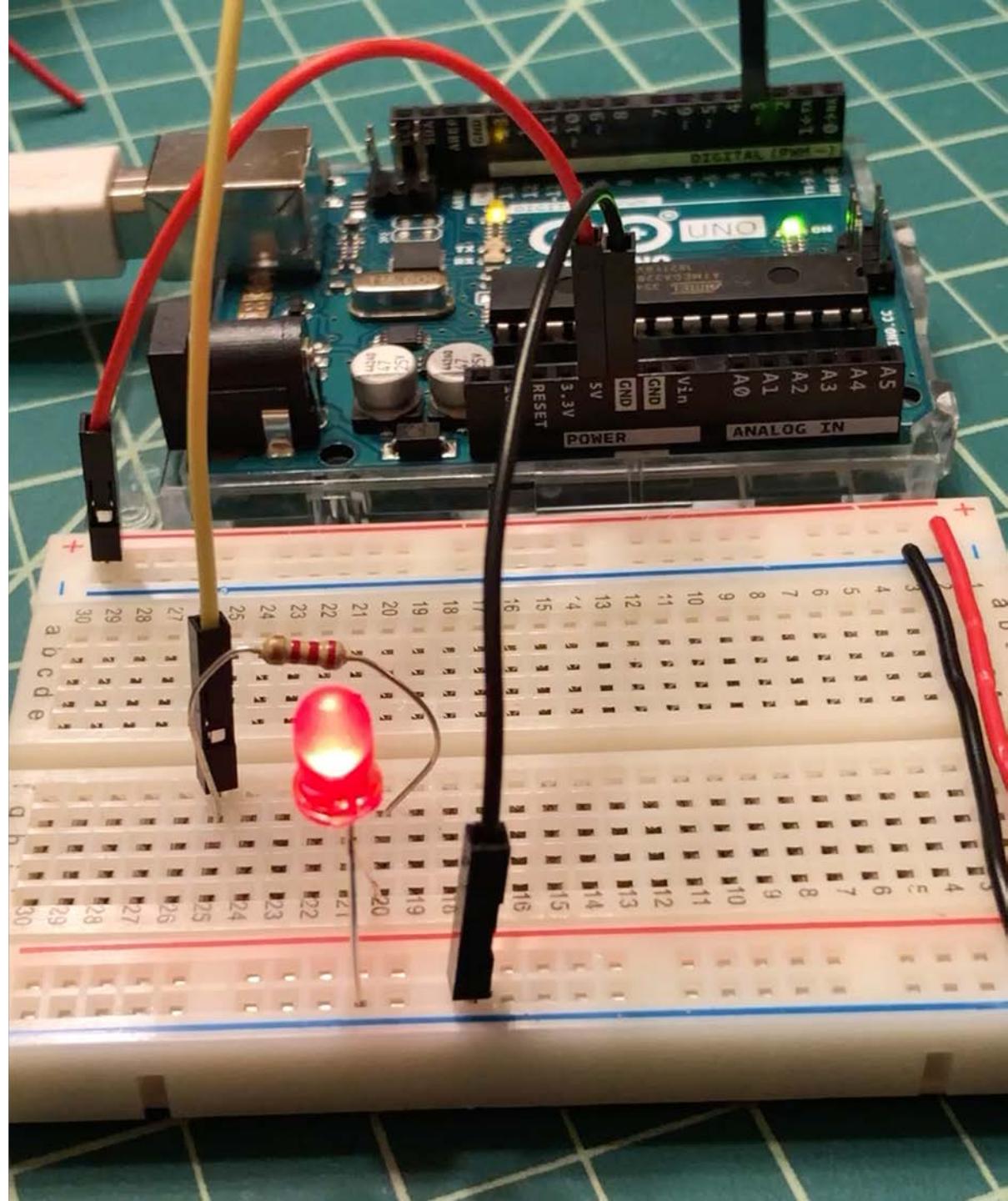
## DIGITAL OUTPUT

# ACTIVITY: FLASH LED ON/OFF USING ARDUINO UNO

Let's write our first Arduino program:  
**blink** an LED on for 1 second and off for 1 second.

When you want to turn something "on" and "off" then this implies digital output, so we will use the **digitalWrite** method.

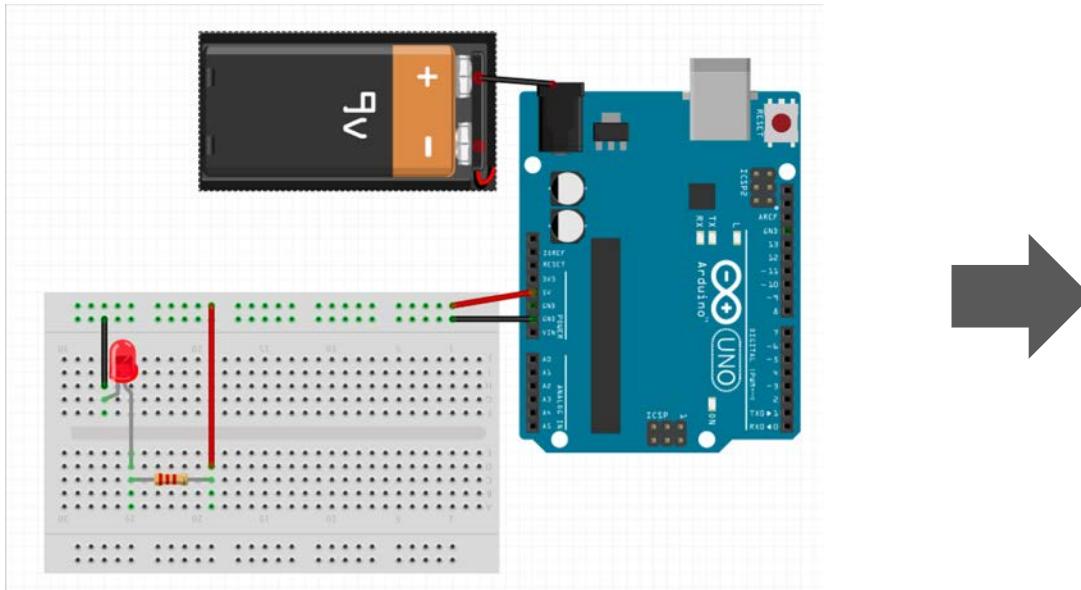
First, we'll modify our circuits and then we'll write code



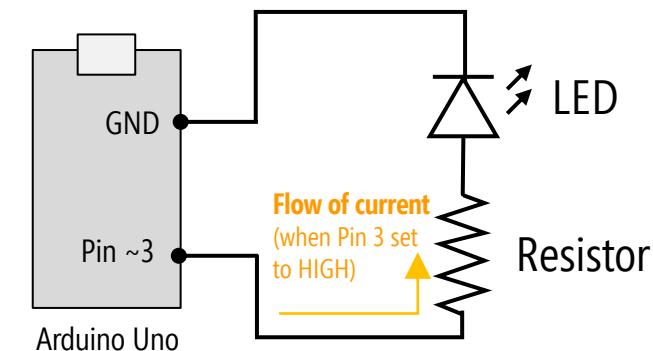
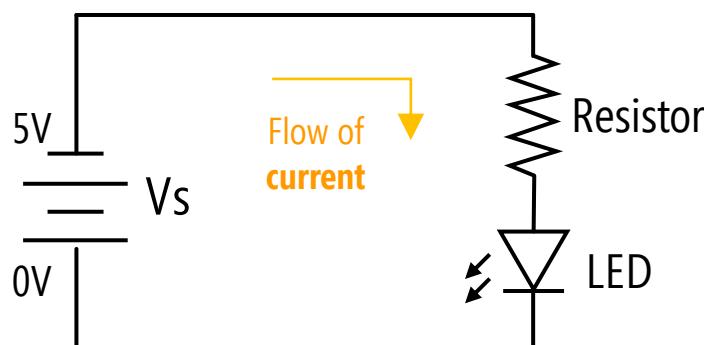
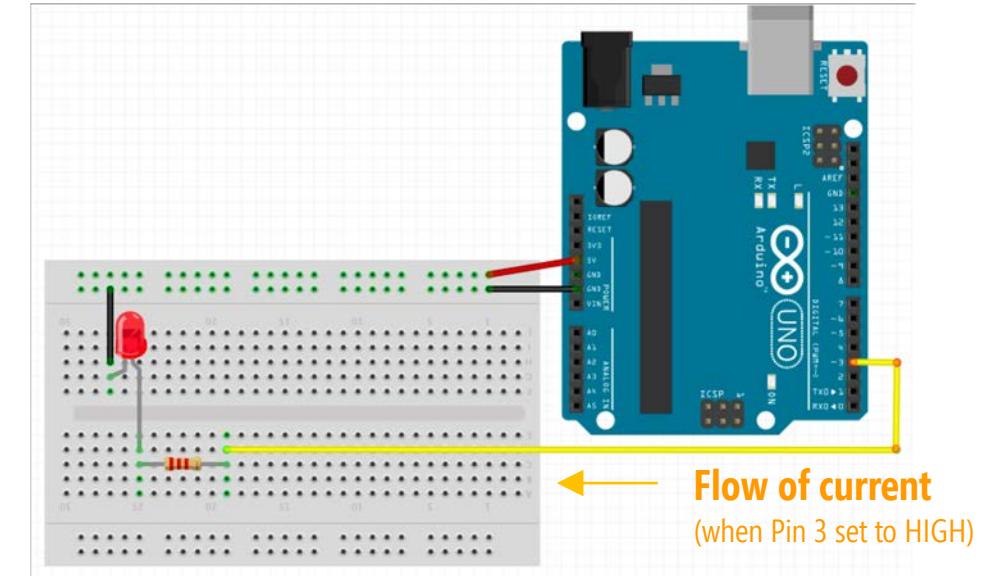
DIGITAL OUTPUT

# FLASH LED ON/OFF USING ARDUINO UNO

Previous Circuit: LED Powered via 5V pin



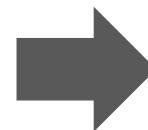
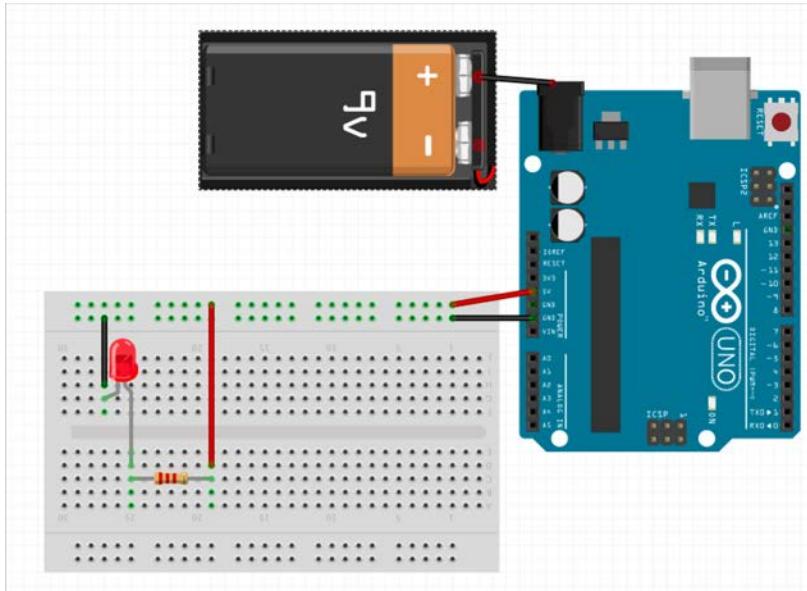
Build Circuit: Flash LED on/off via the pin 3



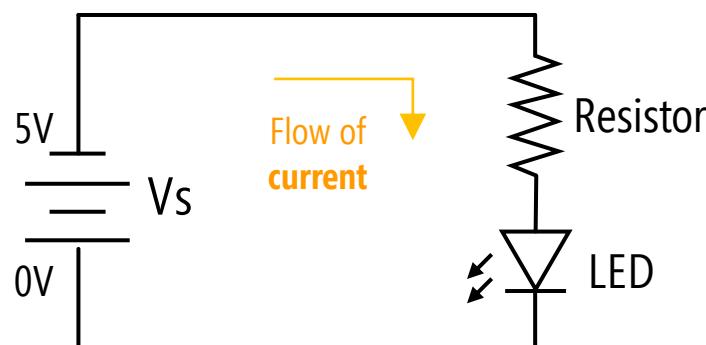
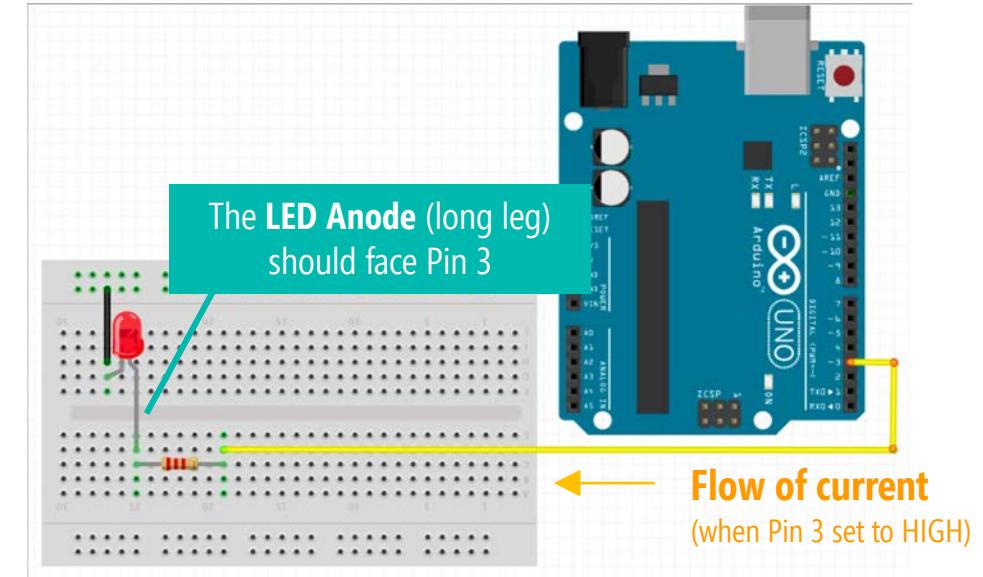
## DIGITAL OUTPUT

# FLASH LED ON/OFF USING ARDUINO UNO

**Previous Circuit:** LED Powered via 5V pin

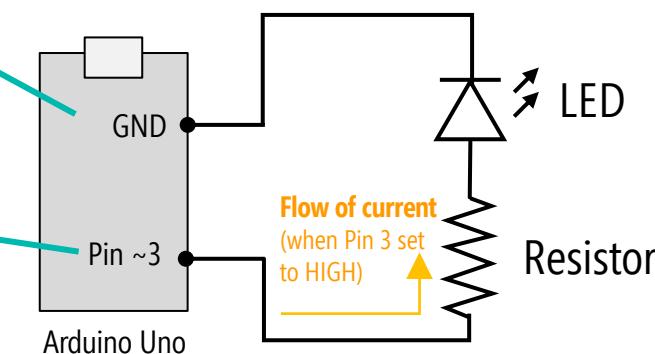


**Build Circuit:** Flash LED on/off via the pin 3



GND is always the lowest potential voltage point in our circuit

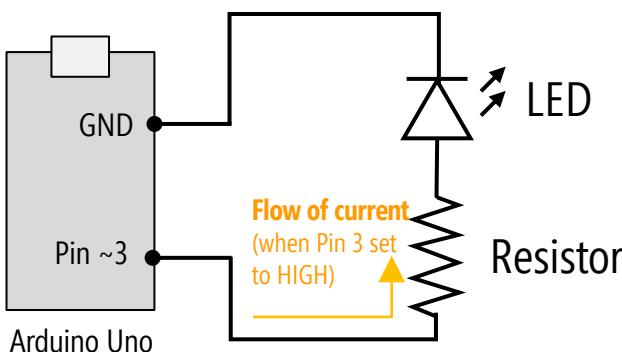
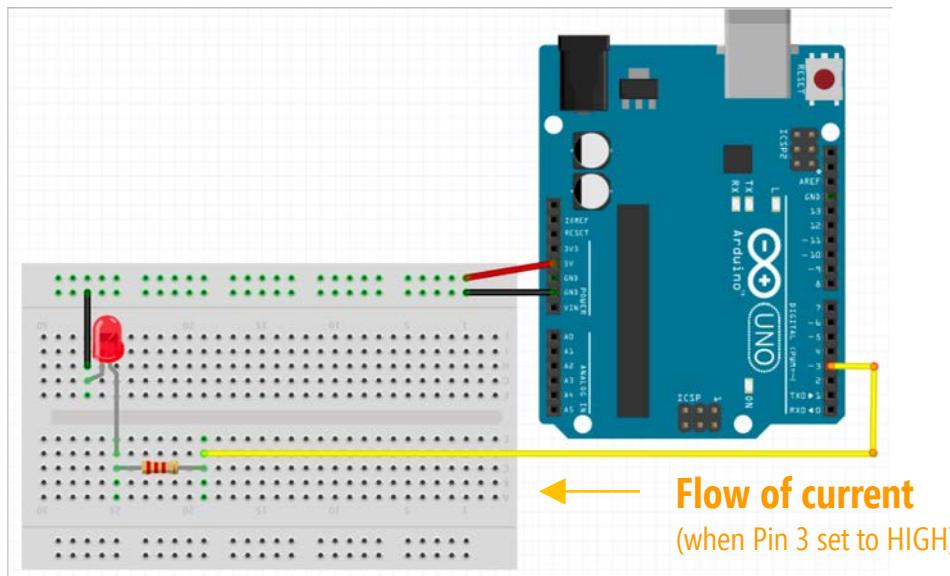
We will program Pin 3 to alternate between 5V and 0V. When 5V, the LED will turn on.



## DIGITAL OUTPUT

# FLASH LED ON/OFF USING ARDUINO UNO

**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

```
/*
 * 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
 */

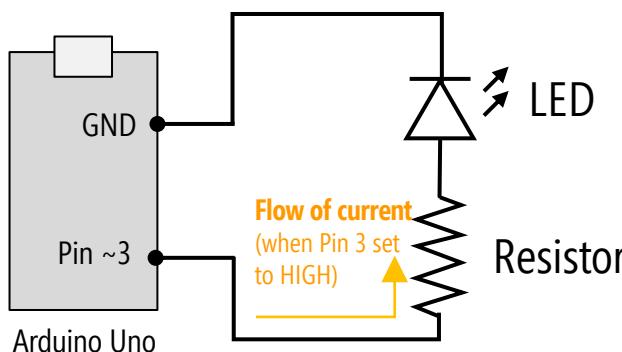
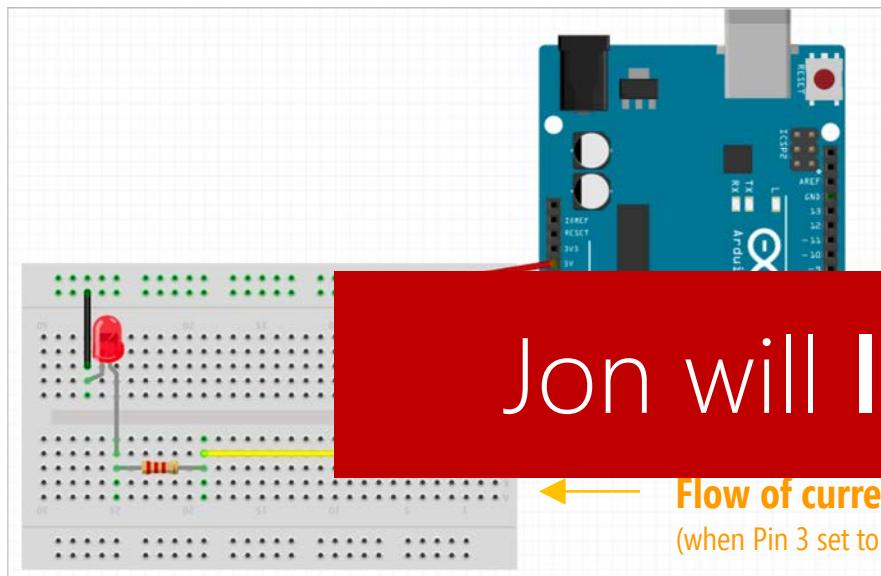
const int LED_OUTPUT_PIN = 3;

// 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
}
```

# FLASH LED ON/OFF USING ARDUINO UNO

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



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

The Arduino IDE interface showing the "Blink" sketch. The code is as follows:

```

/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 */

// 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
}

```

Jon will **live code** this with you!

# pinMode(pin, mode)

Configures the specified digital I/O pin to behave as either an input or an output. You always do this in **setup()**

## Syntax

pinMode(pin, mode)

## Parameters

pin: the pin number whose mode you wish to set

value: INPUT or OUTPUT



```
/*  
 * 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  
}
```

# digitalWrite(pin, value)

Writes a HIGH or LOW value to a digital pin

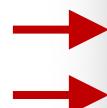
## Syntax

`digitalWrite(pin, value)`

## Parameters

`pin`: the pin number

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



```
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;

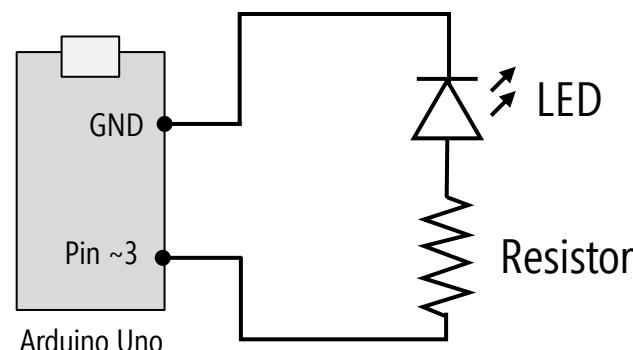
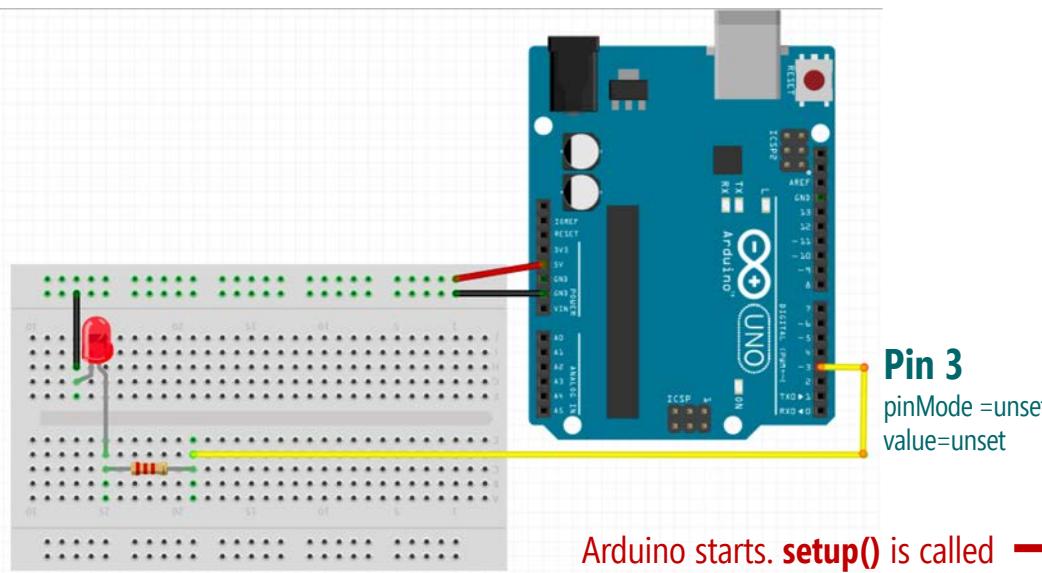
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# LET'S WALK THROUGH WHAT'S HAPPENING HERE

Circuit: Flash LED on/off via the pin 3



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

```
Blink | Arduino 1.8.8

/*
 * 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;

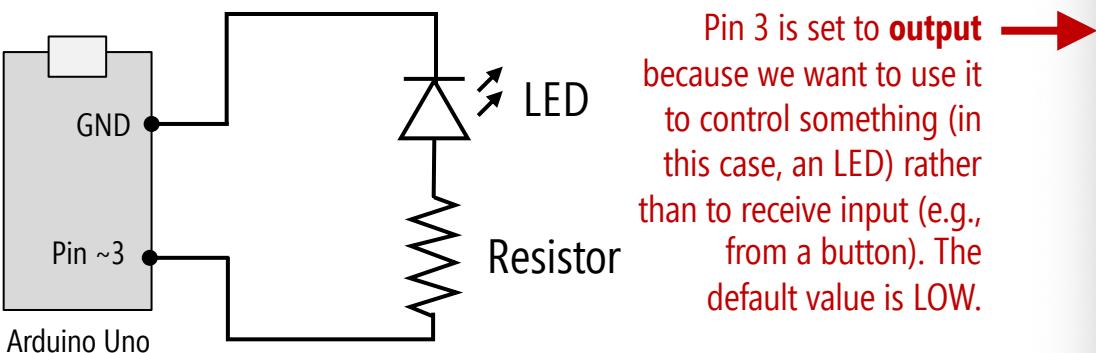
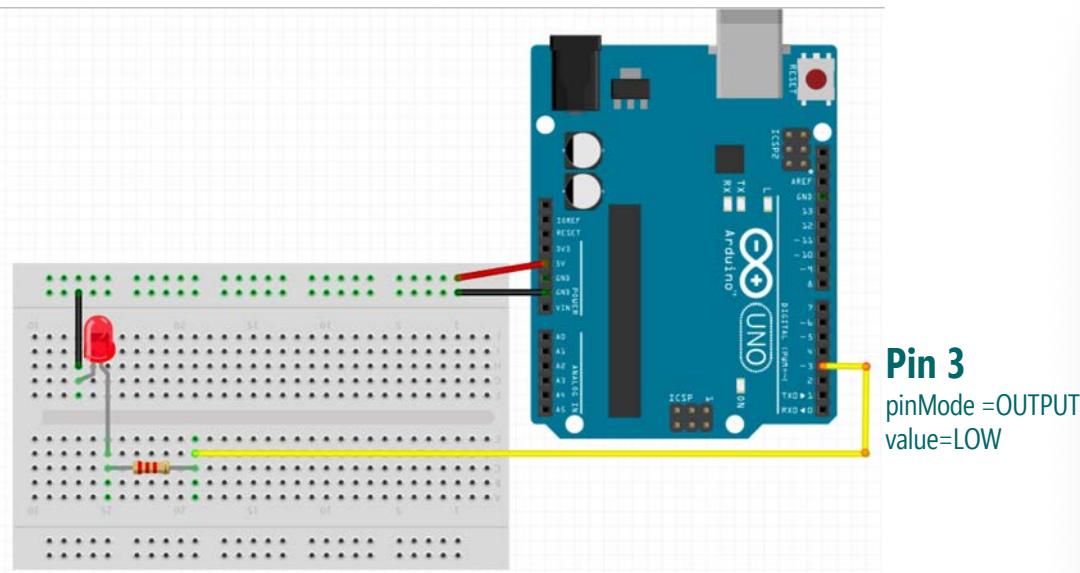
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# PIN 3 SET TO OUTPUT

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



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

```
/*
 * 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;

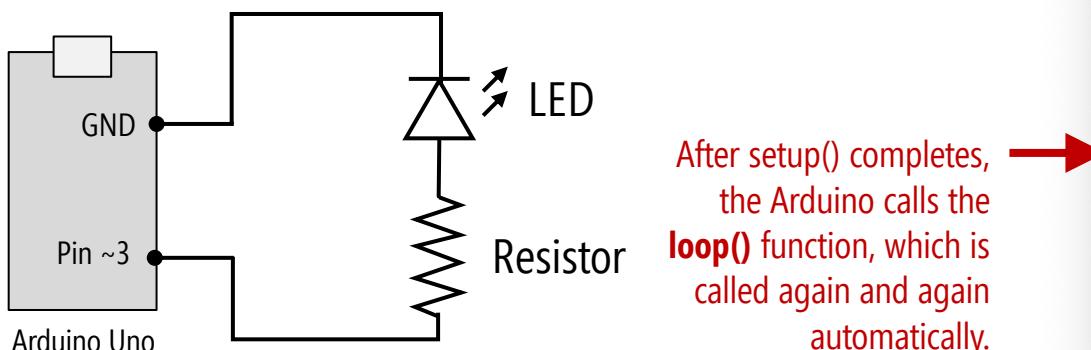
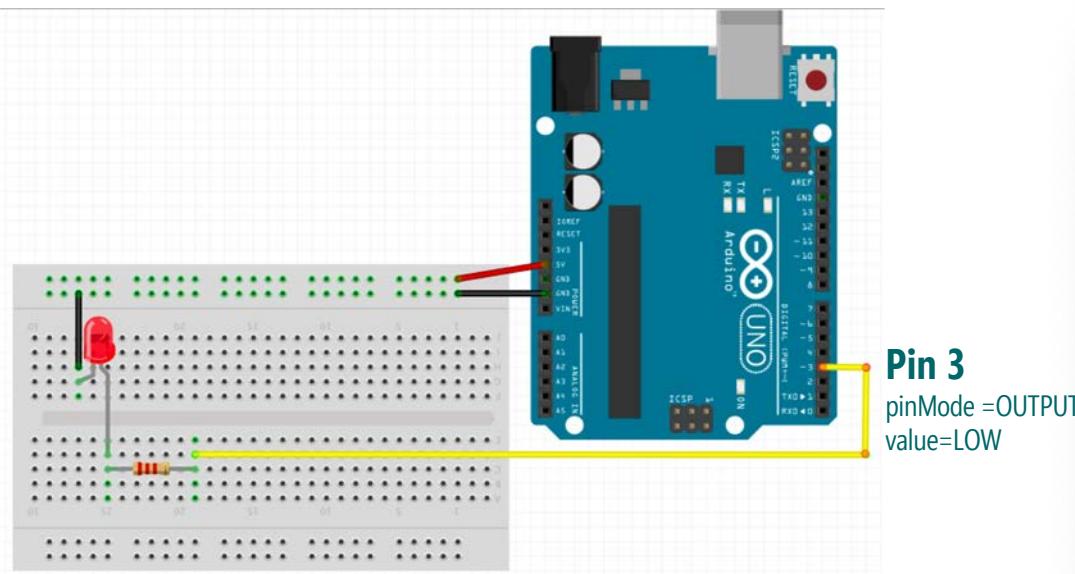
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# SETUP() COMPLETES, LOOP() FUNCTION CALLED...

Circuit: Flash LED on/off via the pin 3



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

The screenshot shows the Arduino IDE interface with the title "Blink | Arduino 1.8.8". The code in the editor is:

```
/*
 * 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;

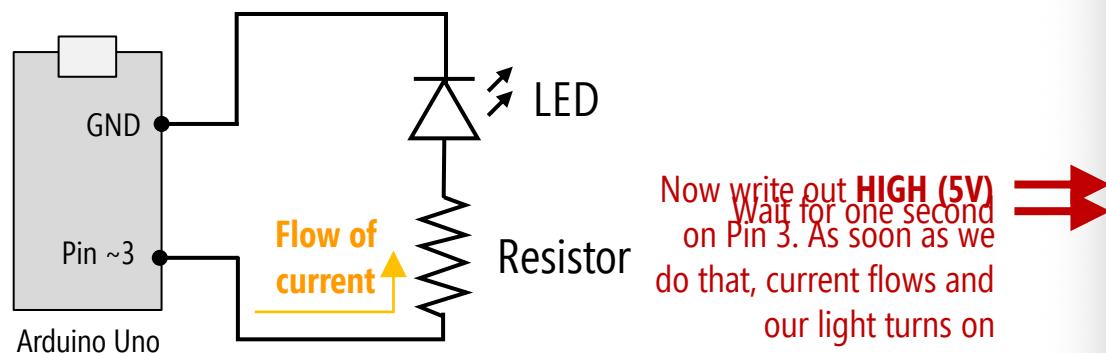
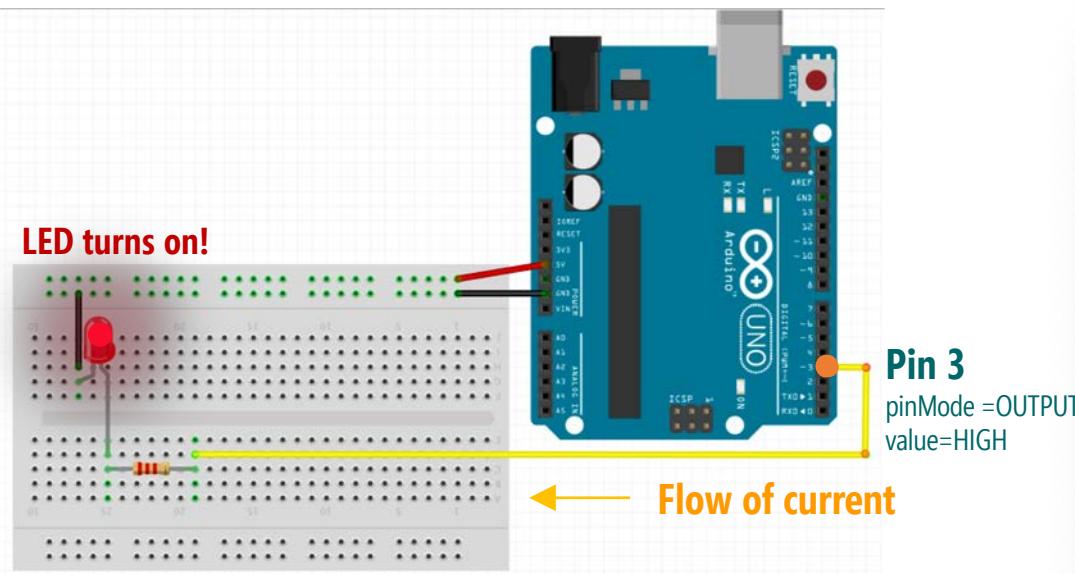
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# PIN 3 SET TO HIGH (5V) FOR 1 SECOND

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



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

```
Blink | Arduino 1.8.8

/*
 * 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;

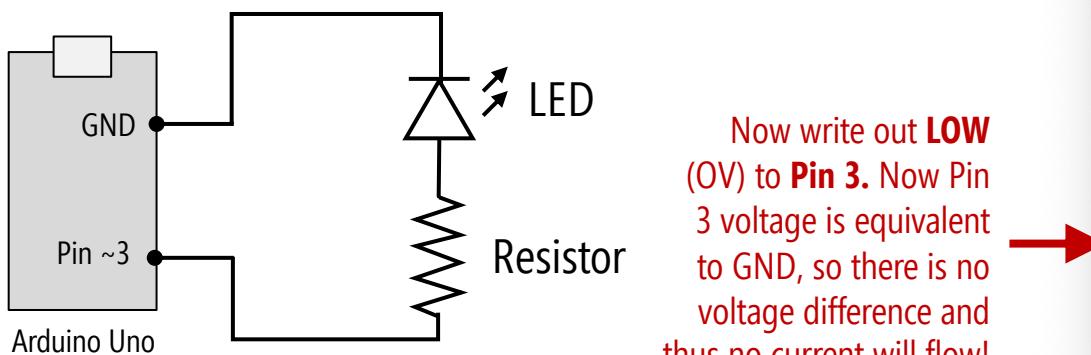
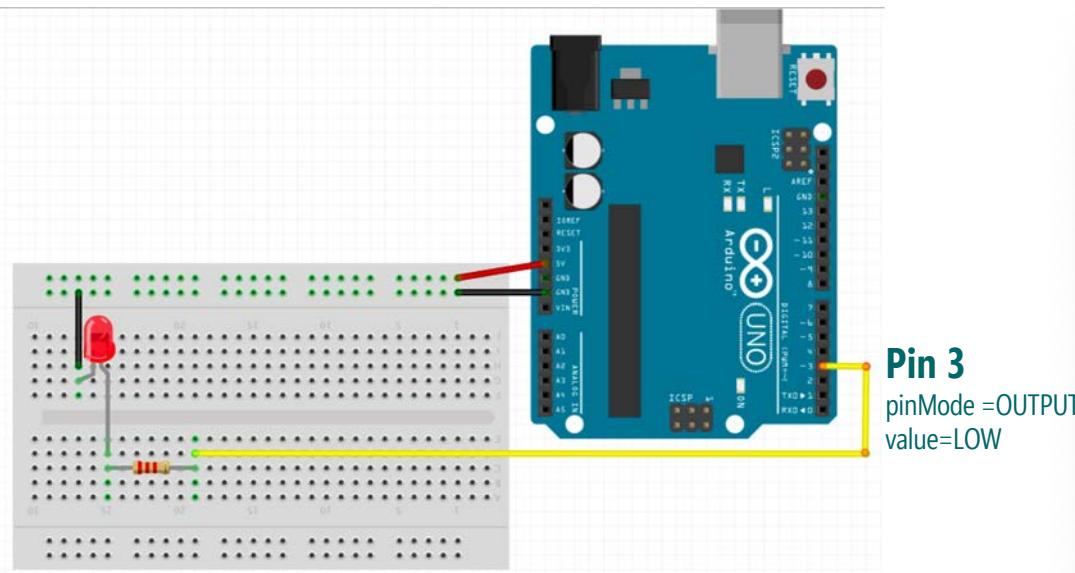
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# PIN 3 SET TO LOW (OV)

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



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

The screenshot shows the Arduino IDE interface with the title "Blink | Arduino 1.8.8". The code is as follows:

```
/*
 * 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;

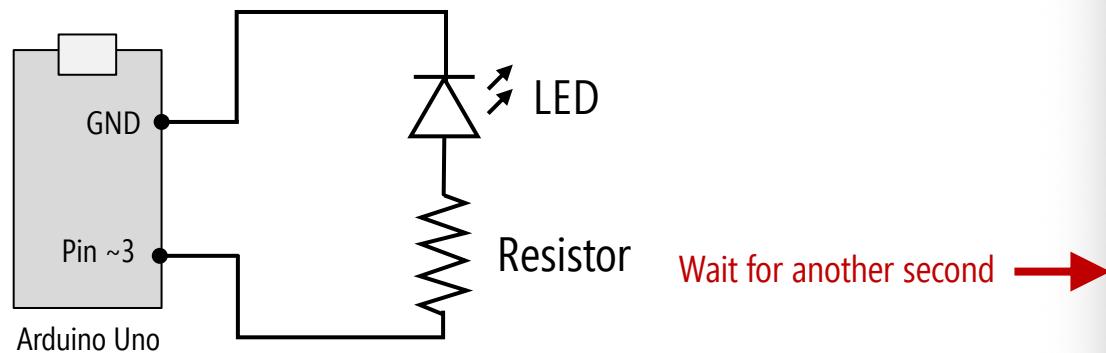
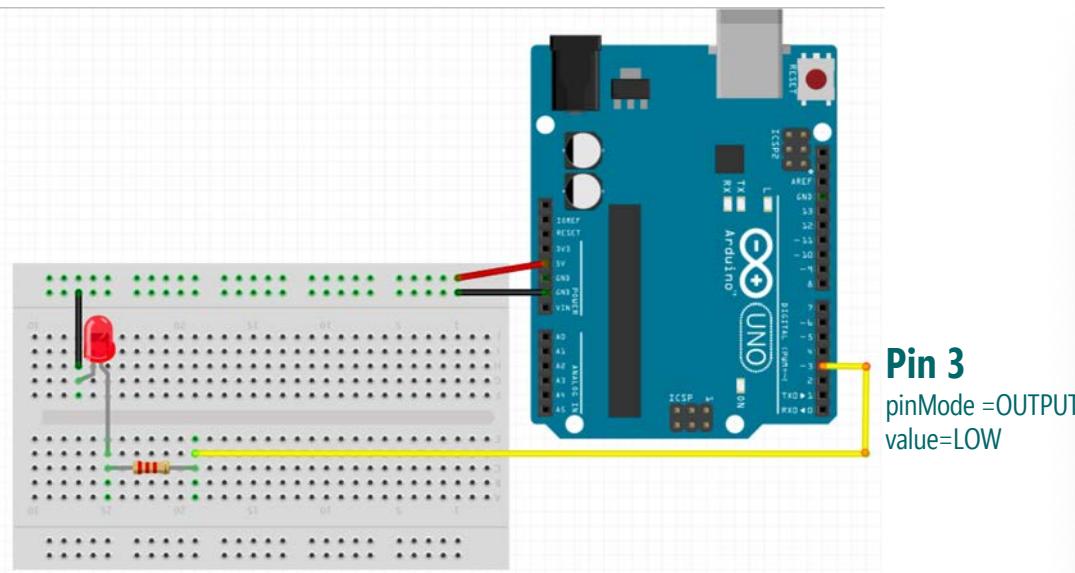
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# PIN 3 SET TO LOW (0V) FOR 1 SECOND

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



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

```
Blink | Arduino 1.8.8

/*
 * 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;

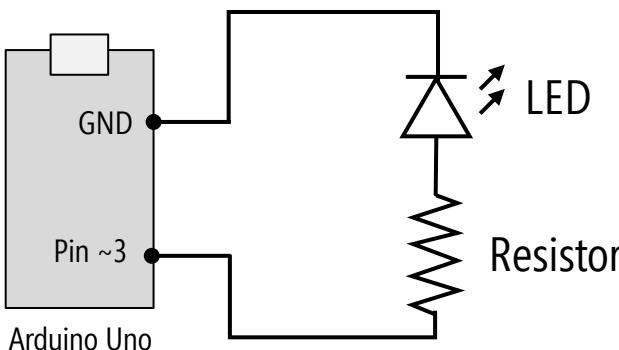
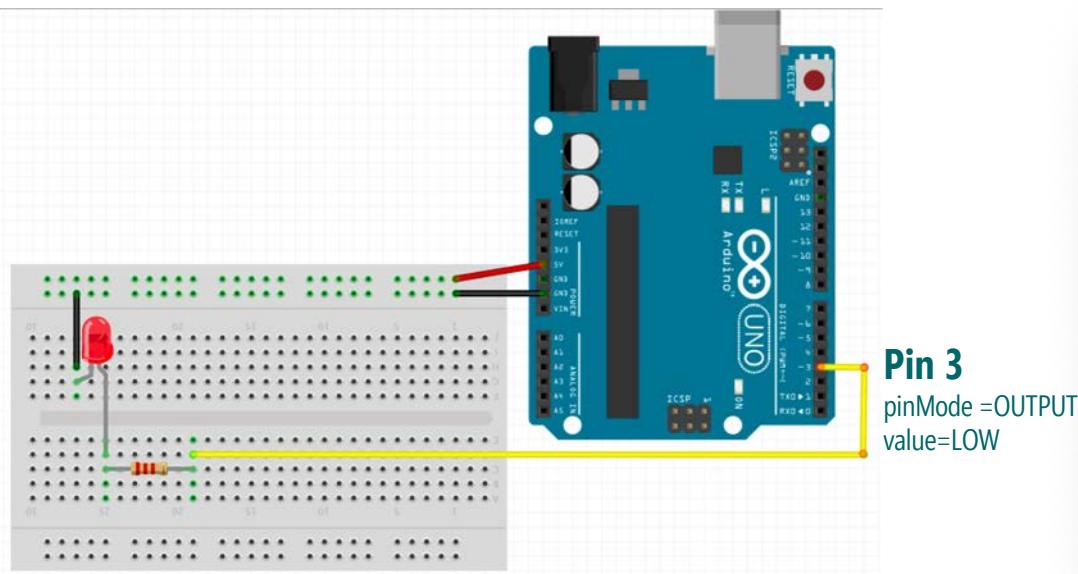
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# LOOP() FUNCTION FINISHES & IMMEDIATELY CALLED AGAIN

Circuit: Flash LED on/off via the pin 3



As soon as we get here,  
the **loop()** function  
finishes and then is  
automatically called  
again by the Arduino!  
And this repeats forever!

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;

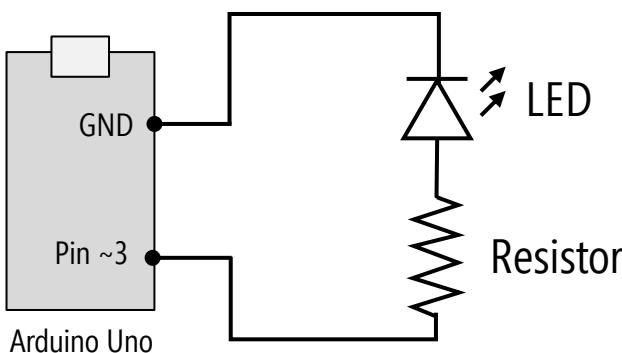
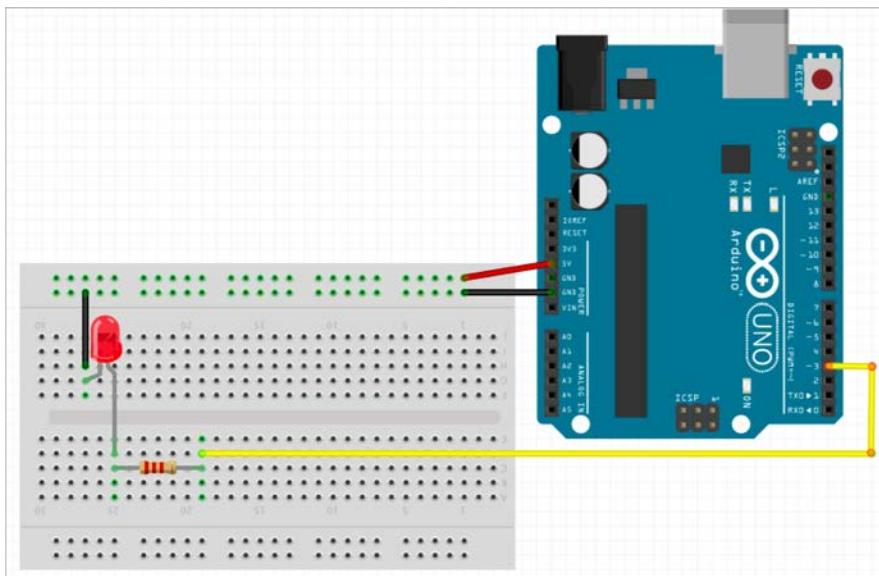
// The setup function runs once when you press reset or power the board
void setup() {
    // Because pins 0 - 13 can either be input or output, we must specify
    // how we're using the pin by using pinMode. In this case, we want to
    // control an LED, so set the pin to OUTPUT
    pinMode(LED_OUTPUT_PIN,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

## DIGITAL OUTPUT

# OK, OUR CIRCUIT IS DONE; OUR CODE IS READY...

**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

/*
 * 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
}
```

Compile your program (check for syntax errors, etc.)

Upload your program to the Arduino



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.8". The main area displays the "Blink" sketch code. A tooltip with the text "Compile your program (check for syntax errors, etc.)" points to the top-left corner of the code area. Another tooltip with the text "Upload your program to the Arduino" points to the bottom-left corner of the code area. The code itself is a standard Blink example:

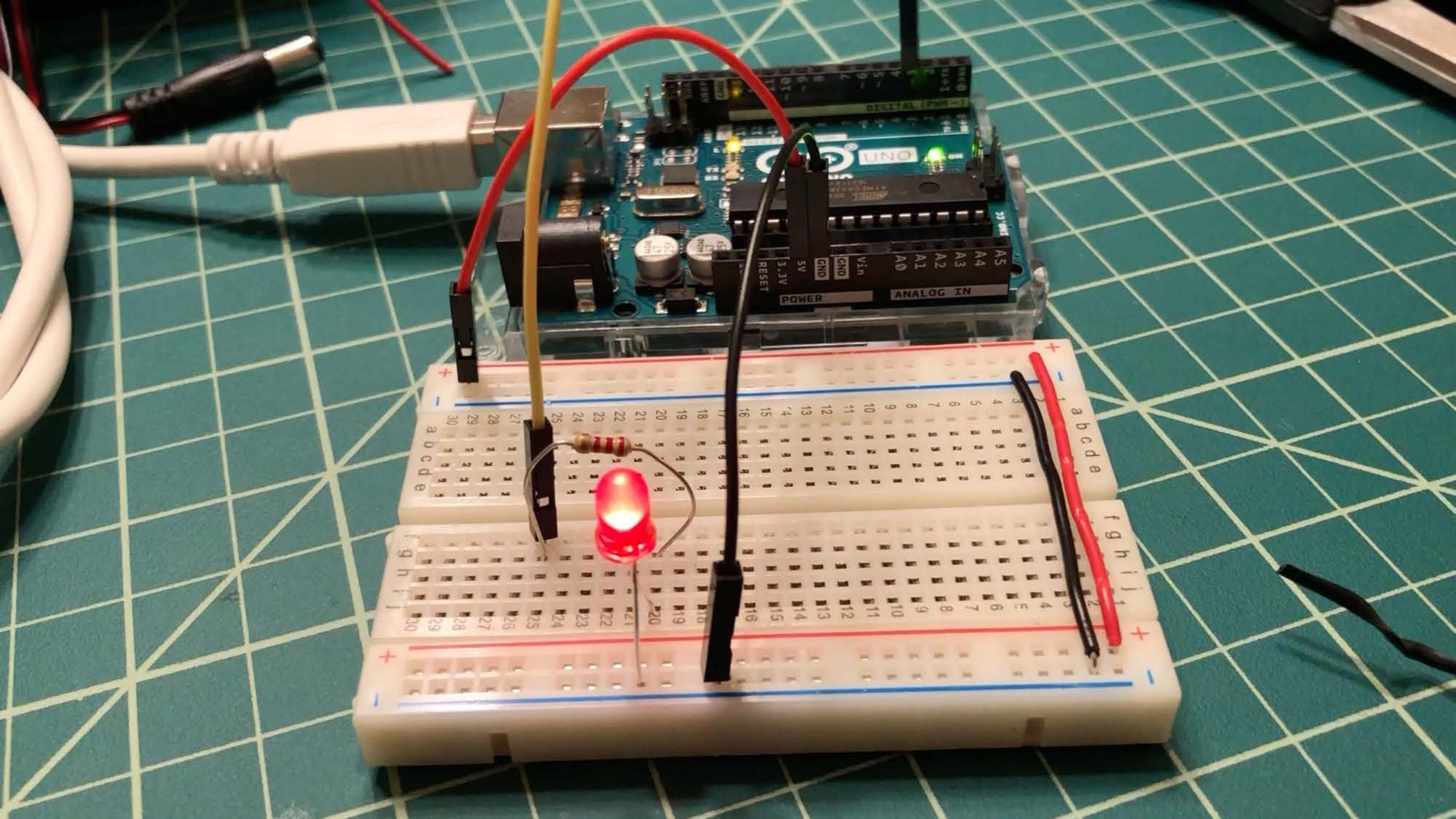
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 * File -> Examples -> 01. Basics -> Blink
 */

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
}

Done Saving.
```

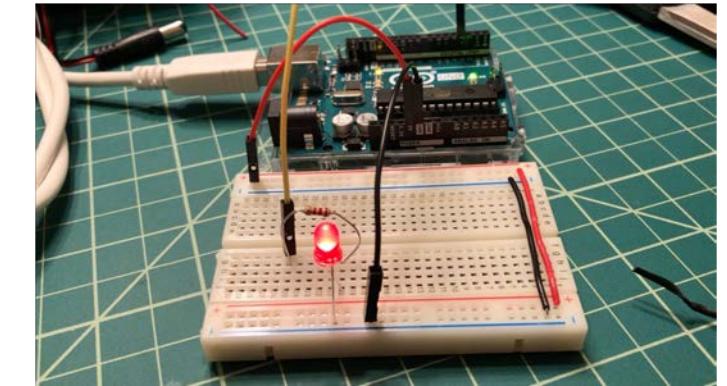


# HOW WOULD YOU MAKE THE LED FLASH FASTER?

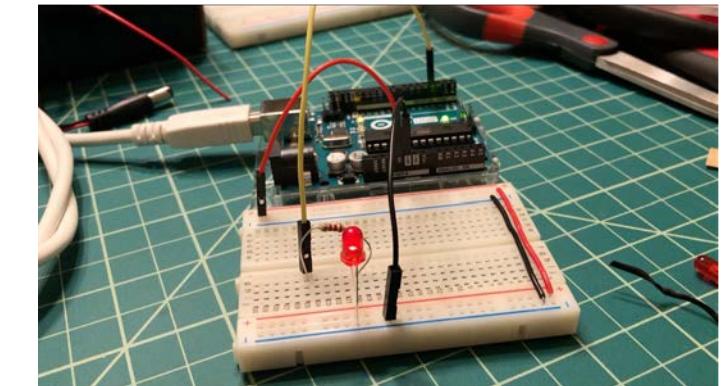
```
// The loop function runs over and over again forever
void loop() {
    digitalWrite(LED_OUTPUT_PIN, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                            // delay is in milliseconds; so wait one second
    digitalWrite(LED_OUTPUT_PIN, LOW);      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

# HOW WOULD YOU MAKE THE LED FLASH FASTER?

```
// 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
}
```



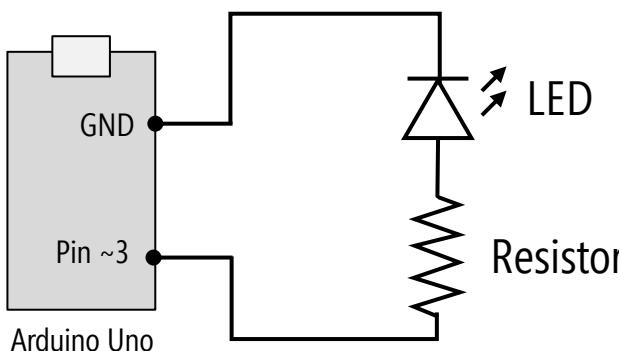
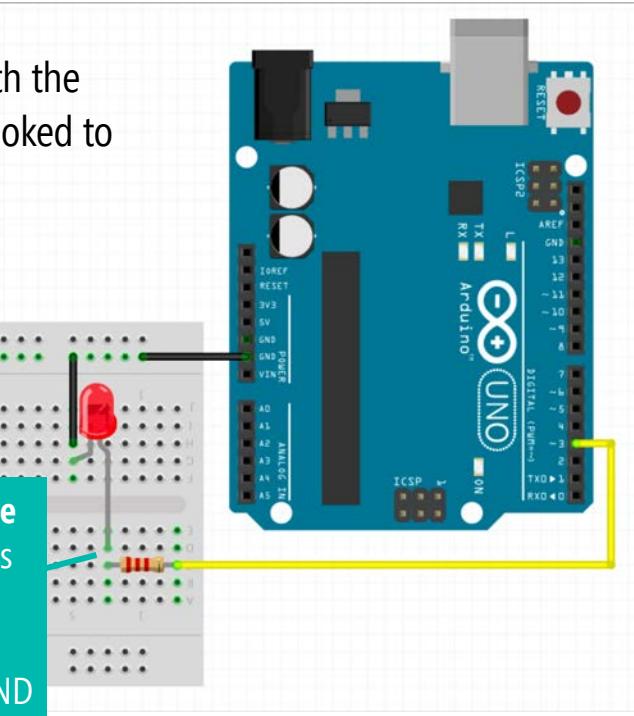
```
// The loop function runs over and over again forever
void loop() {
    digitalWrite(LED_OUTPUT_PIN, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(100);                      // delay is in milliseconds; so wait 100ms
    digitalWrite(LED_OUTPUT_PIN, LOW); // turn the LED off by making the voltage LOW
    delay(100);                      // wait a 100ms
}
```



## DIGITAL OUTPUT

# WE CAN HOOK UP THE CIRCUIT ANOTHER WAY BUT IT'S LESS INTUITIVE

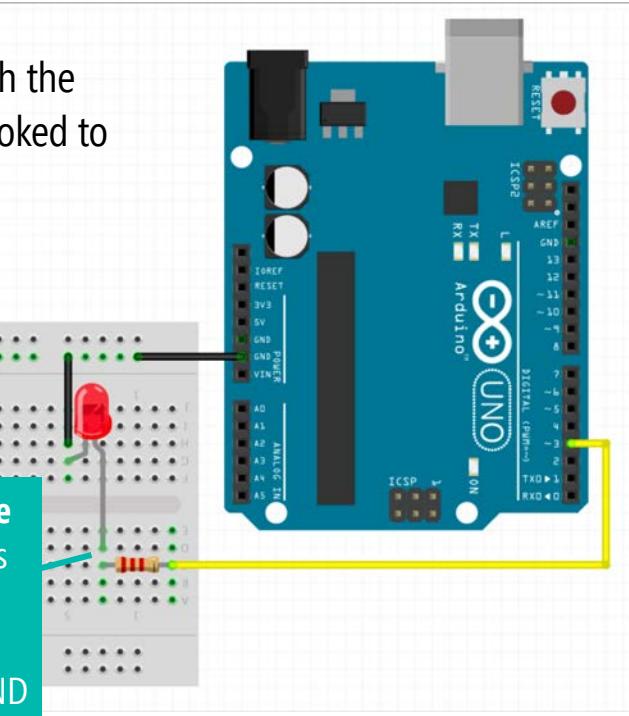
Old circuit with the LED anode hooked to Pin 3



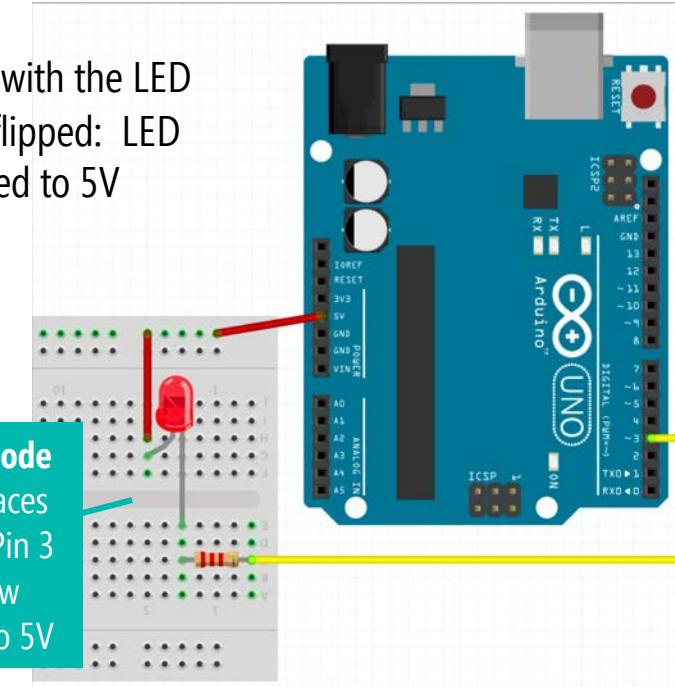
## DIGITAL OUTPUT

# WE CAN HOOK UP THE CIRCUIT ANOTHER WAY BUT IT'S LESS INTUITIVE

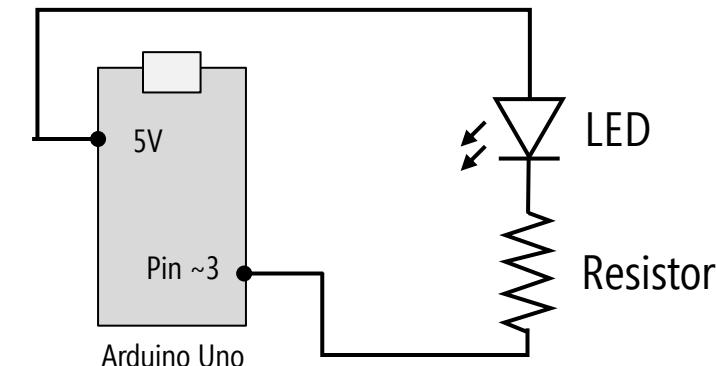
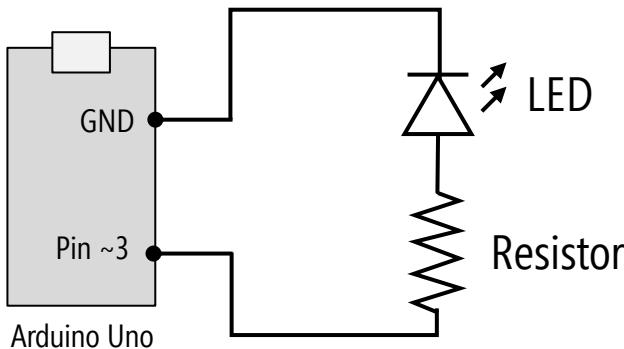
Old circuit with the LED anode hooked to Pin 3



New circuit with the LED orientation flipped: LED anode hooked to 5V



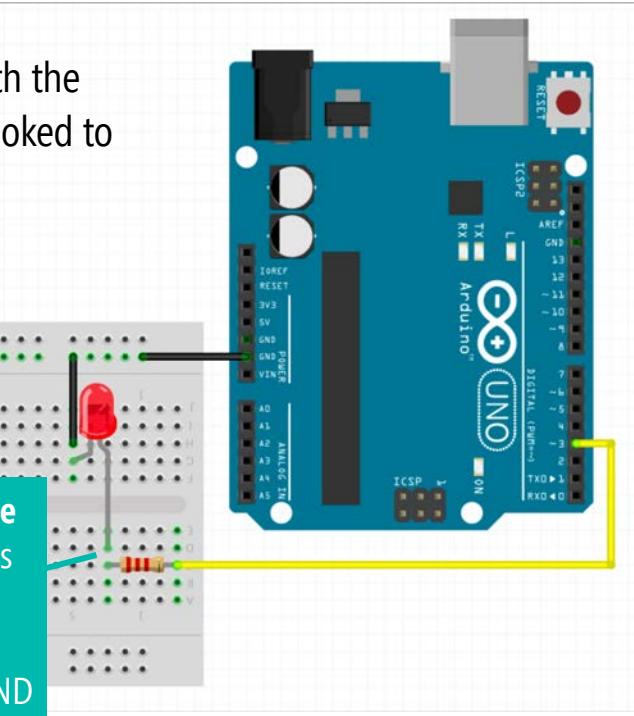
The LED Anode (long leg) faces Pin 3 and the cathode is connected to GND



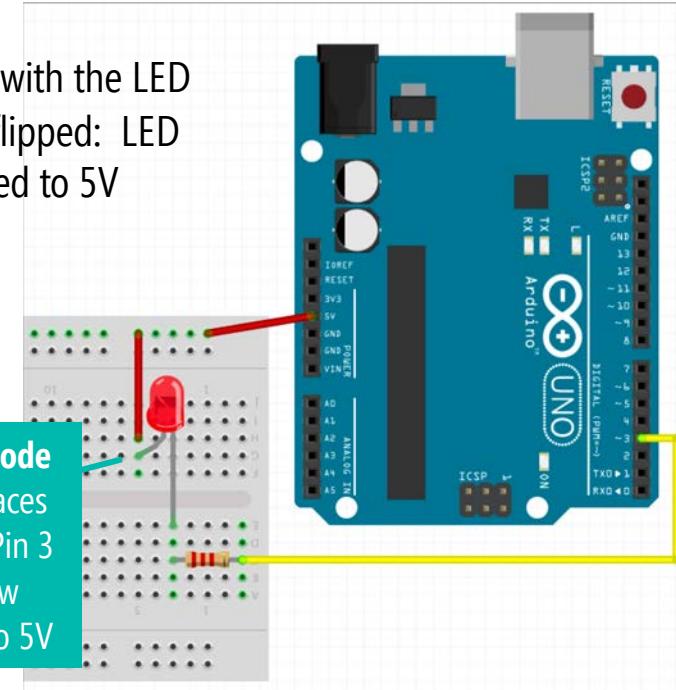
## DIGITAL OUTPUT

# WHAT HAPPENS WITH THIS NEW CIRCUIT WHEN PIN 3 SET TO LOW?

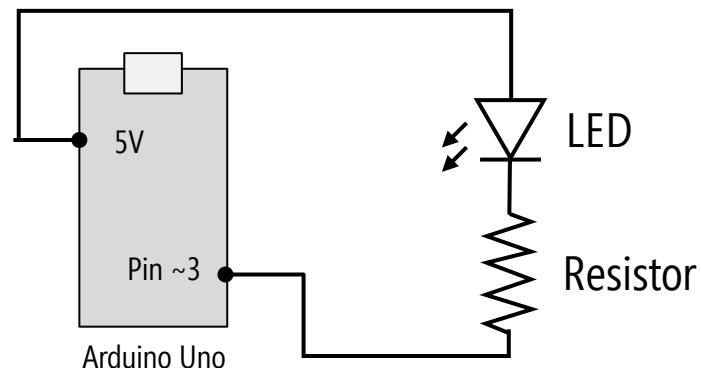
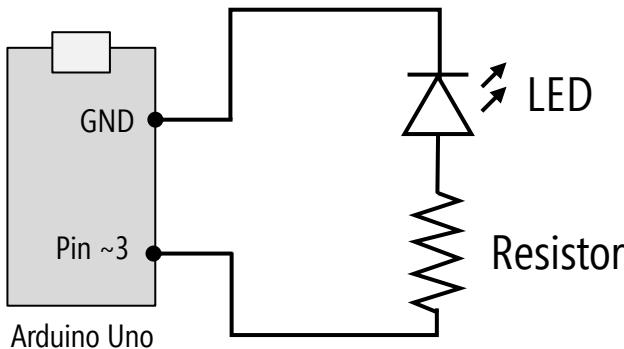
**Old circuit** with the LED anode hooked to Pin 3



**New circuit** with the LED orientation flipped: LED anode hooked to 5V



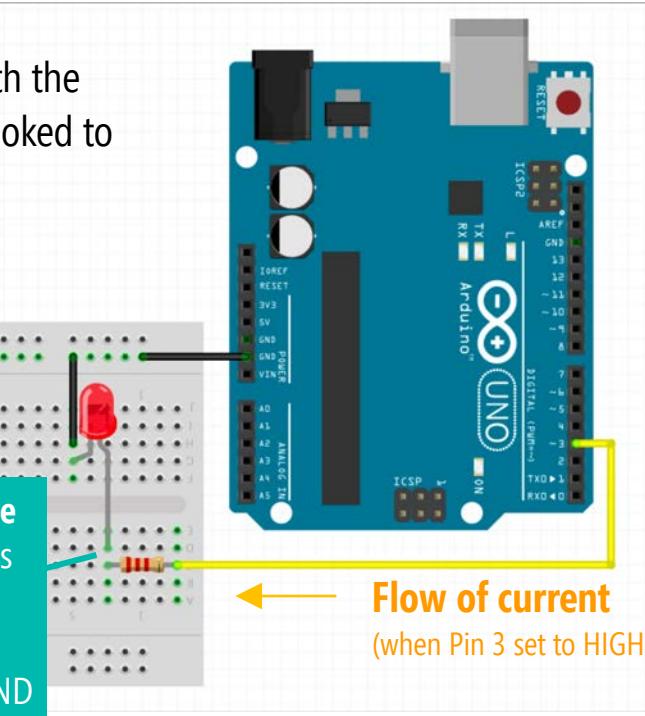
In this configuration, what happens when Pin 3 is set to LOW?



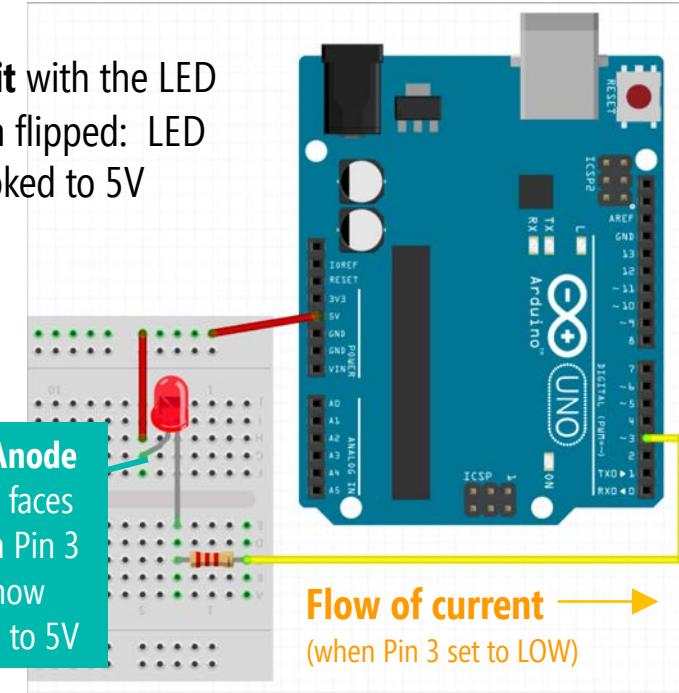
## DIGITAL OUTPUT

# WITH NEW CIRCUIT, LED TURNS ON WHEN PIN 3 IS LOW!

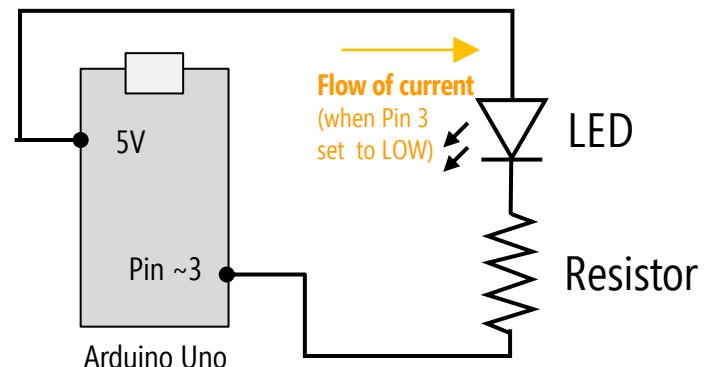
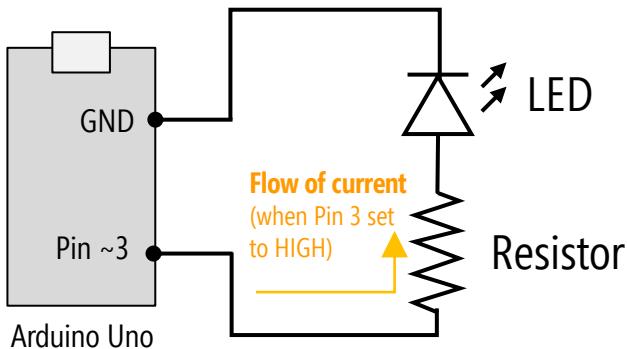
Old circuit with the LED anode hooked to Pin 3



New circuit with the LED orientation flipped: LED anode hooked to 5V



In this configuration, what happens when Pin 3 is set to LOW?

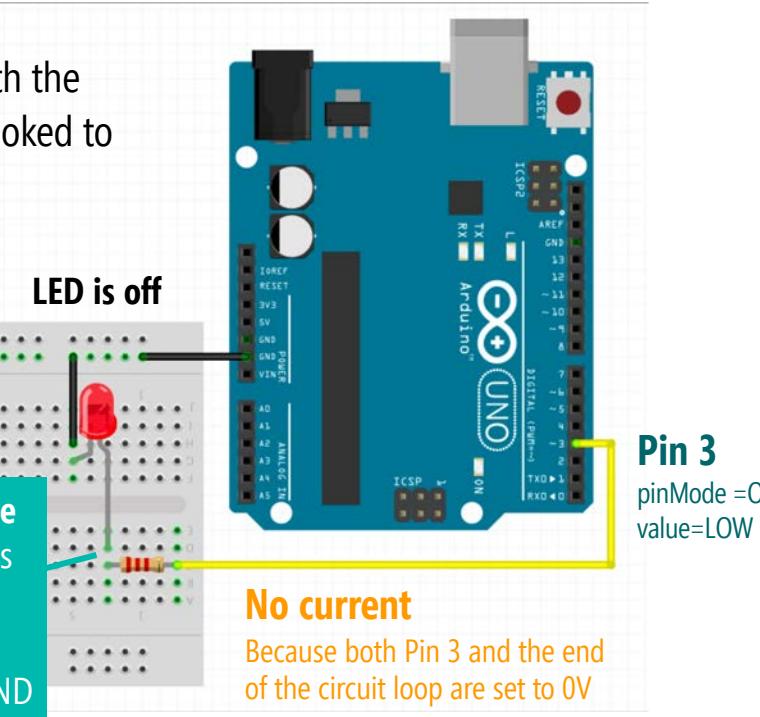


**Answer:**  
The LED turns on!

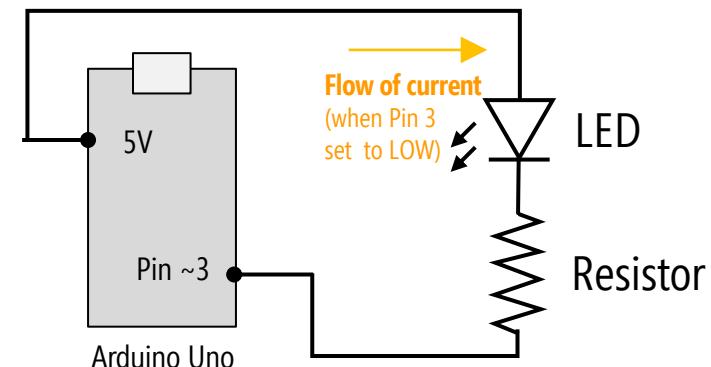
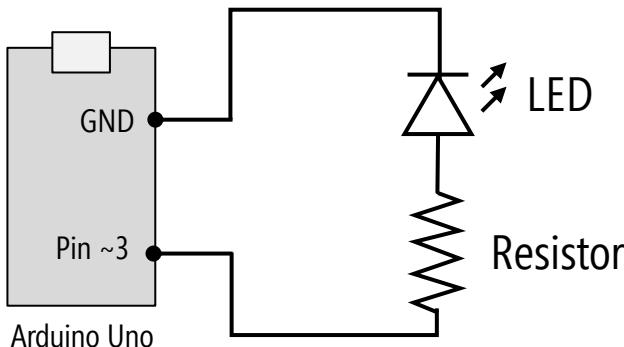
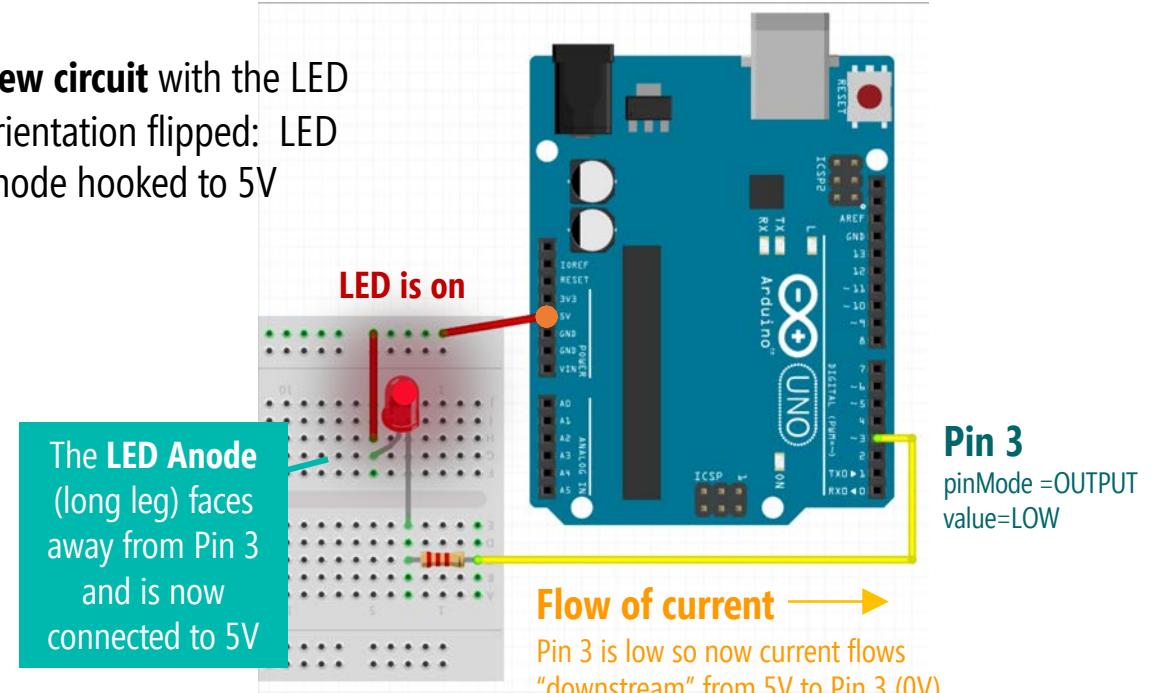
## DIGITAL OUTPUT

# HERE'S WHAT HAPPENS WHEN PIN 3 IS LOW (0V)

Old circuit with the LED anode hooked to Pin 3



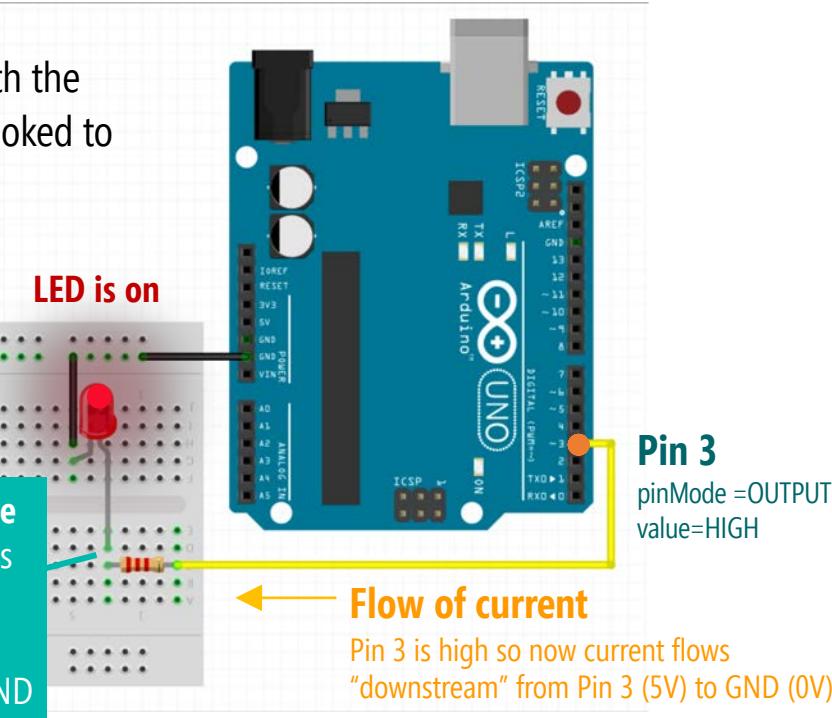
New circuit with the LED orientation flipped: LED anode hooked to 5V



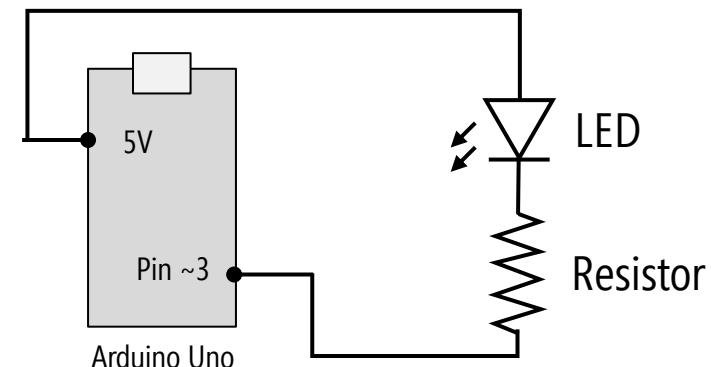
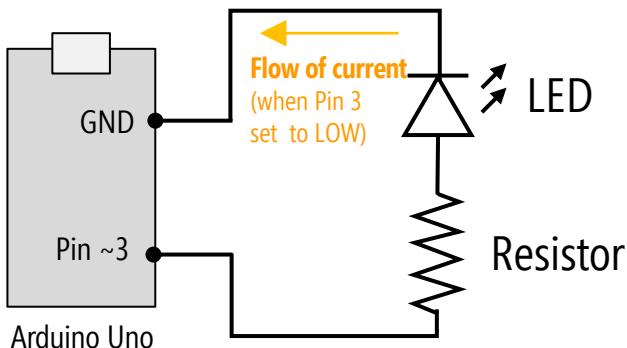
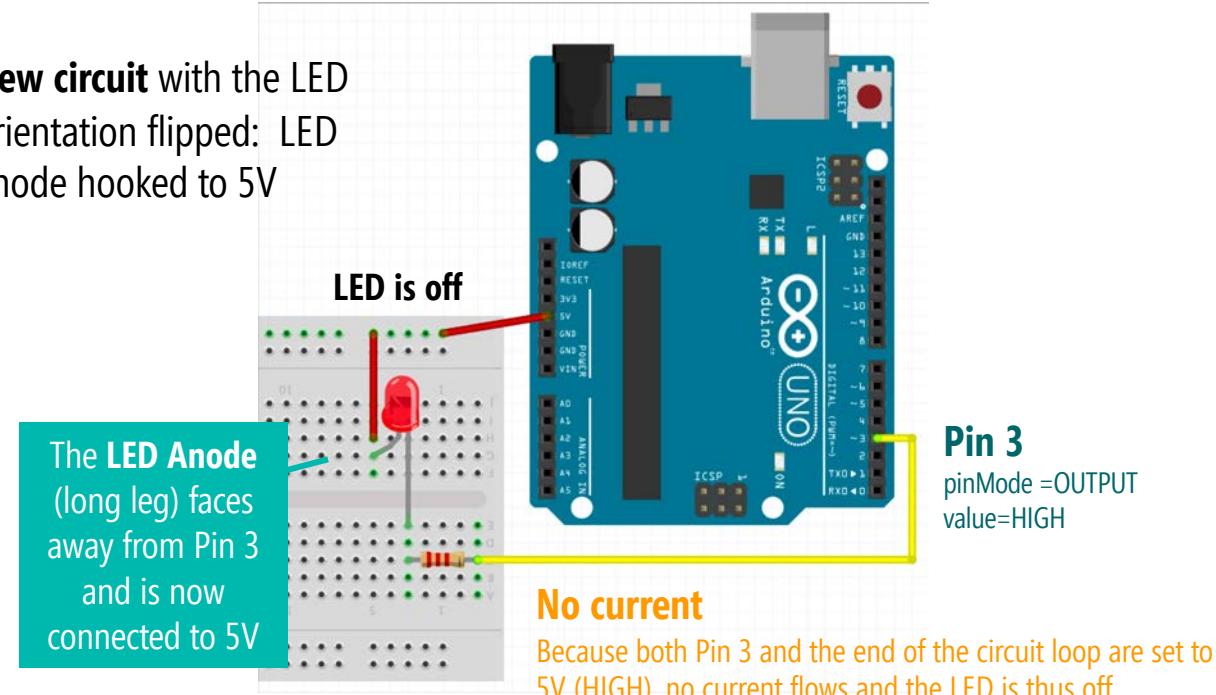
## DIGITAL OUTPUT

# HERE'S WHAT HAPPENS WHEN PIN 3 IS HIGH (5V)

Old circuit with the LED anode hooked to Pin 3

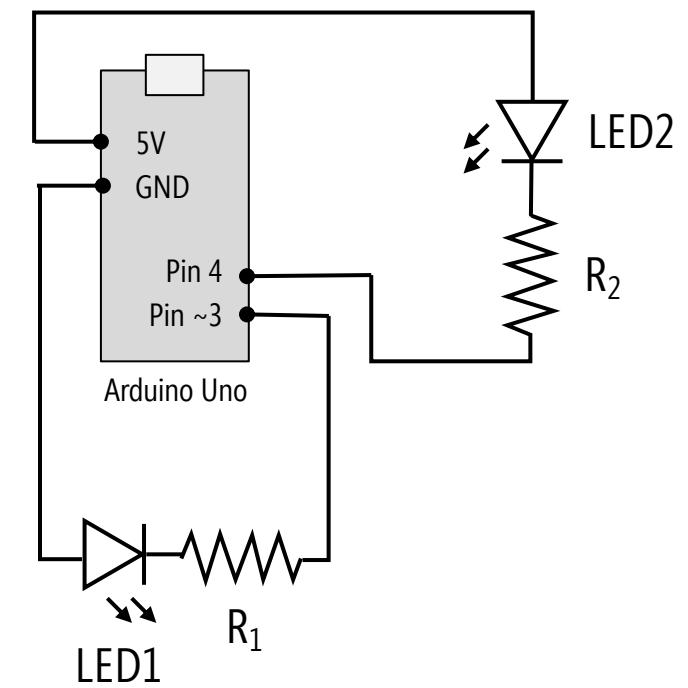
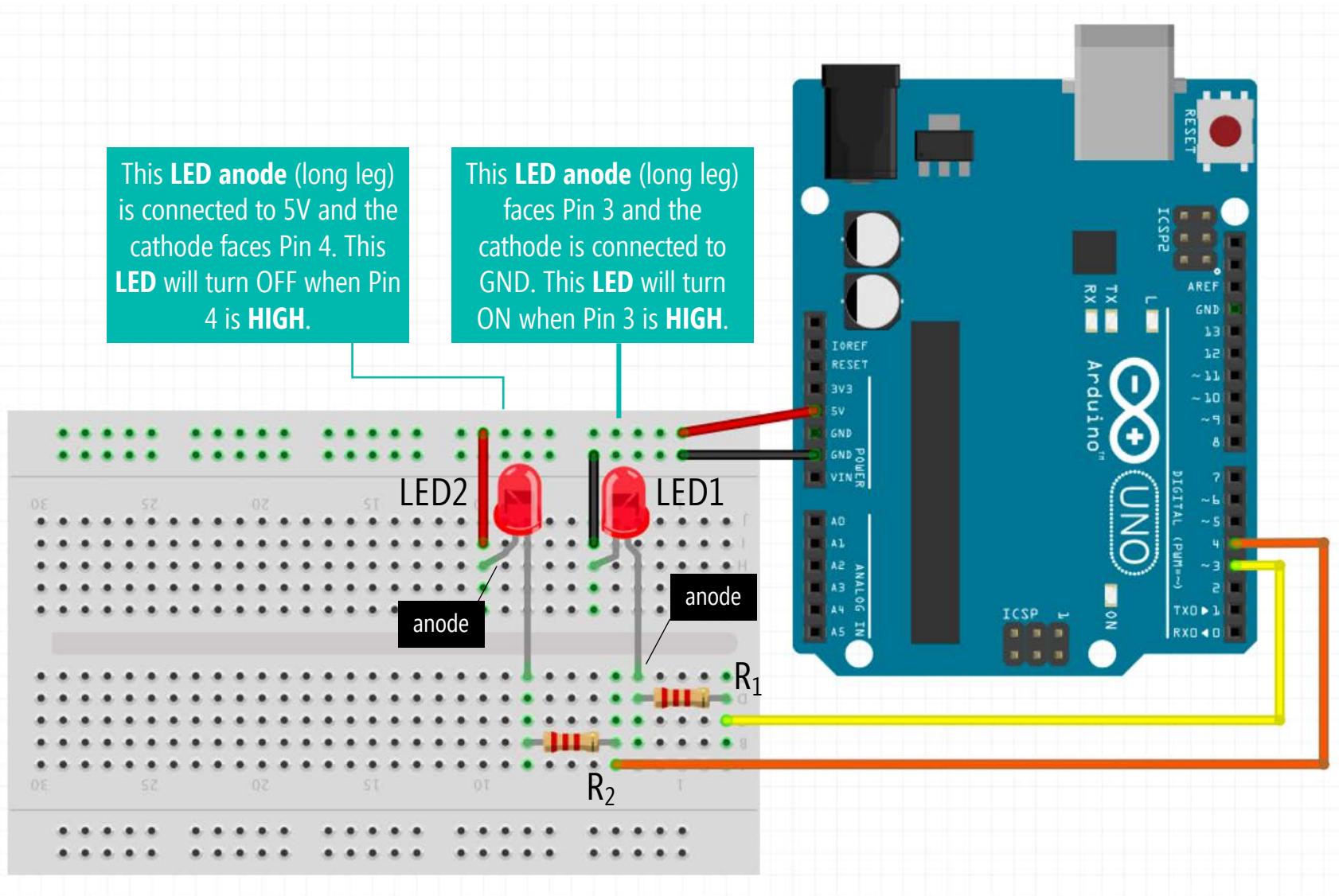


New circuit with the LED orientation flipped: LED anode hooked to 5V



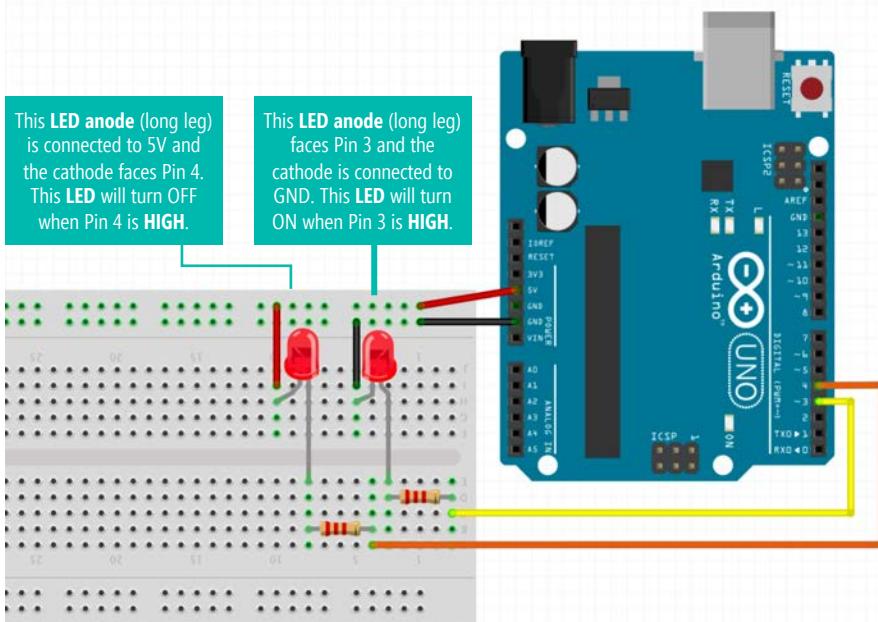
## DIGITAL OUTPUT

# LET'S WIRE UP BOTH CONFIGURATIONS AT THE SAME TIME



## DIGITAL OUTPUT

# AND HERE'S THE CODE



This LED anode (long leg) is connected to 5V and the cathode faces Pin 4. This LED will turn OFF when Pin 4 is HIGH.

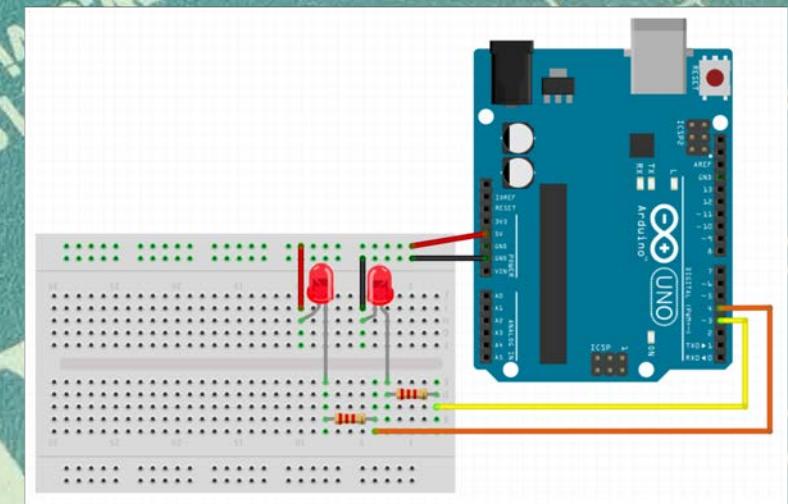
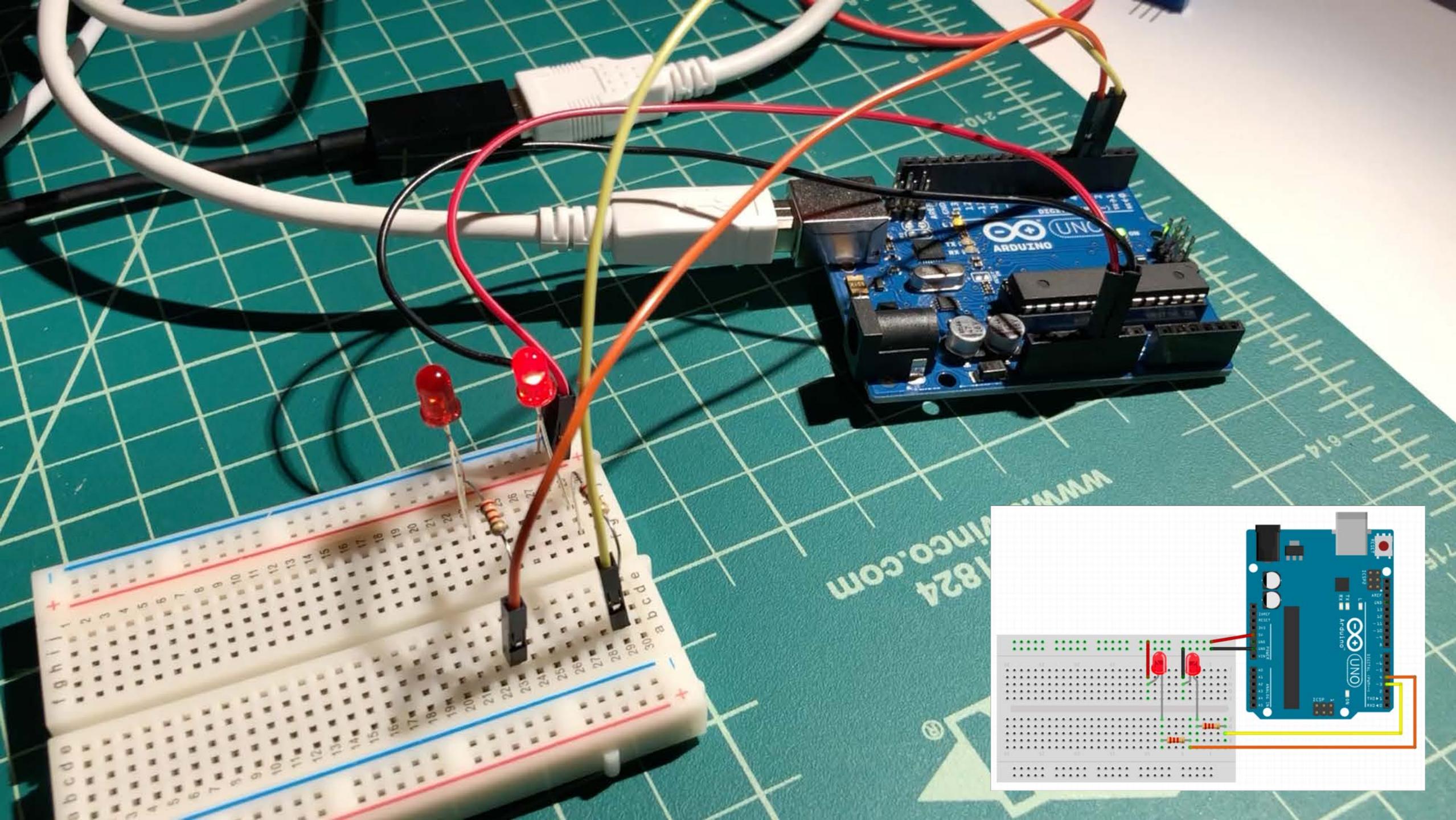
This LED anode (long leg) faces Pin 3 and the cathode is connected to GND. This LED will turn ON when Pin 3 is HIGH.

```
const int LED1_OUTPUT_PIN = 3; // Anode faces Pin 3 (cathode connected to 0V)
const int LED2_OUTPUT_PIN = 4; // Cathode faces Pin 4 (anode connected to 5V)

// 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 LEDs, so set both pins to OUTPUT
    pinMode(LED1_OUTPUT_PIN, OUTPUT);
    pinMode(LED2_OUTPUT_PIN, OUTPUT);
}

// The loop function runs over and over again forever
void loop() {
    // Below, you're going to see that driving Pin 3 HIGH will turn on LED1
    // but driving Pin 4 HIGH will actually turn *off* LED2
    digitalWrite(LED1_OUTPUT_PIN, HIGH); // turns ON LED1 (Pin 3 is now 5V and other leg of LED is 0V)
    digitalWrite(LED2_OUTPUT_PIN, HIGH); // turns OFF LED2 (Pin 4 is now 5V and other leg of LED is 5V)
    delay(1000); // delay is in milliseconds; so wait one second

    digitalWrite(LED1_OUTPUT_PIN, LOW); // turns OFF LED1 (Pin 3 is now 0V and other leg of LED is 0V)
    digitalWrite(LED2_OUTPUT_PIN, LOW); // turns ON LED2 (Pin 4 is now 0V and other leg of LED is 5V)
    delay(1000); // wait for a second
}
```



# CAREFUL WITH DELAYS!

When you **call delay()**, the MCU is in a **hold state** and unresponsive to new input

Better to, instead, **track time yourself** in loop() and perform operations based on elapsed time.

## Blink §

```
/*
 * Simply turns on and off pin 3 once a second
 *
 * By Jon Froehlich
 * 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
}
```

# BLINK WITH DELAY()

## Blink §

```
/*
 * Simply turns on and off pin 3 once a second
 *
 * By Jon Froehlich
 * 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
}
```

# BLINK WITHOUT DELAY()

## BlinkWithoutDelay §

```
const int LED_OUTPUT_PIN = 3;
const int INTERVAL_IN_MS = 1000; // interval at which to blink (in milliseconds)

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long _lastToggledTimestampMs = 0; // tracks the last time the LED was updated
int _ledState = LOW; // will toggle between LOW and HIGH

// 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() {
    unsigned long currentTimestampMs = millis();
    // check to see how much time has elapsed and if it exceeds our interval
    if (currentTimestampMs - _lastToggledTimestampMs >= INTERVAL_IN_MS) {
        _lastToggledTimestampMs = currentTimestampMs;
        _ledState = _ledState == HIGH ? LOW : HIGH;
        digitalWrite(LED_OUTPUT_PIN, _ledState);
    }
}
```

DEBUGGING

# DEBUGGING ARDUINO

Modularize.

Use Serial.println -> Serial Monitor and Serial Plotter

Use a multimeter

## SERIAL MONITOR

# SERIAL MONITOR IS USEFUL FOR DEBUGGING

RedBearDuoReadPotSetLED | Arduino 1.8.5

File Edit Sketch Tools Help

RedBearDuo

```
void setup()
{
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(POTENTIOMETER_PIN, INPUT);
    Serial.begin(9600);
}

void loop()
{
    // read the potentiometer value
    int potVal = analogRead(POTENTIOMETER_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);

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

    // 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);
}
```

/dev/cu.usbmodem14611

Send

```
93,5
0,0
5,0
553,34
758,47
556,34
2,0
1,0
122,7
788,49
790,49
201,12
0,0
47,2
272,16
278,17
280,17
279,17
278,17
277,17
280,17
278,17
277,17
278,17
279,17
277,17
276,17
276,17
278,17
278,17
```

Autoscroll

No line ending

9600 baud

Clear output

## DIGITAL OUTPUT

# USING SERIAL MONITOR FOR DEBUGGING

The image shows the Arduino IDE interface. On the left, the code for 'BlinkWithSerialPrint' is displayed:

```
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);

  Serial.begin(9600); // Turn on the serial port, which is necessary for Serial.print
}

// The loop function runs over and over again forever
void loop() {
  int ledVal = HIGH;
  digitalWrite(LED_OUTPUT_PIN, ledVal); // turn the LED on (HIGH is the voltage level)
  Serial.println(ledVal);
  delay(500); // wait 500ms
  ledVal = LOW;
  digitalWrite(LED_OUTPUT_PIN, ledVal); // turn the LED off by making the voltage LOW
  Serial.println(ledVal);
  delay(500); // wait 500ms
}
```

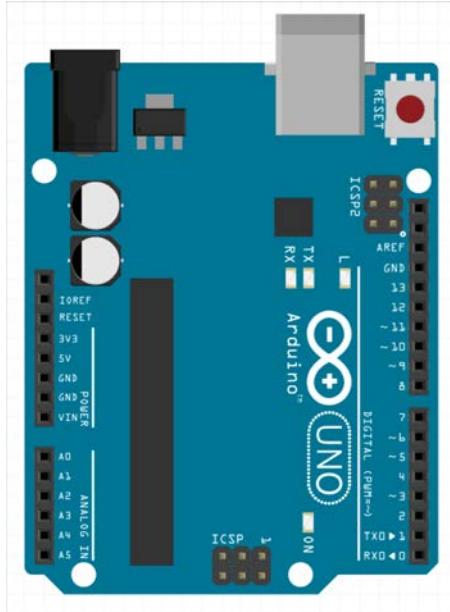
On the right, the Serial Monitor window is open, connected to the port `/dev/cu.usbmodem14601` (Arduino/Genuino Uno). The baud rate is set to 9600. The monitor displays the digital values being printed by the code, showing a repeating pattern of 0s and 1s.

Serial Monitor settings: Autoscroll (checked), Show timestamp (unchecked), Newline, 9600 baud, Clear output.

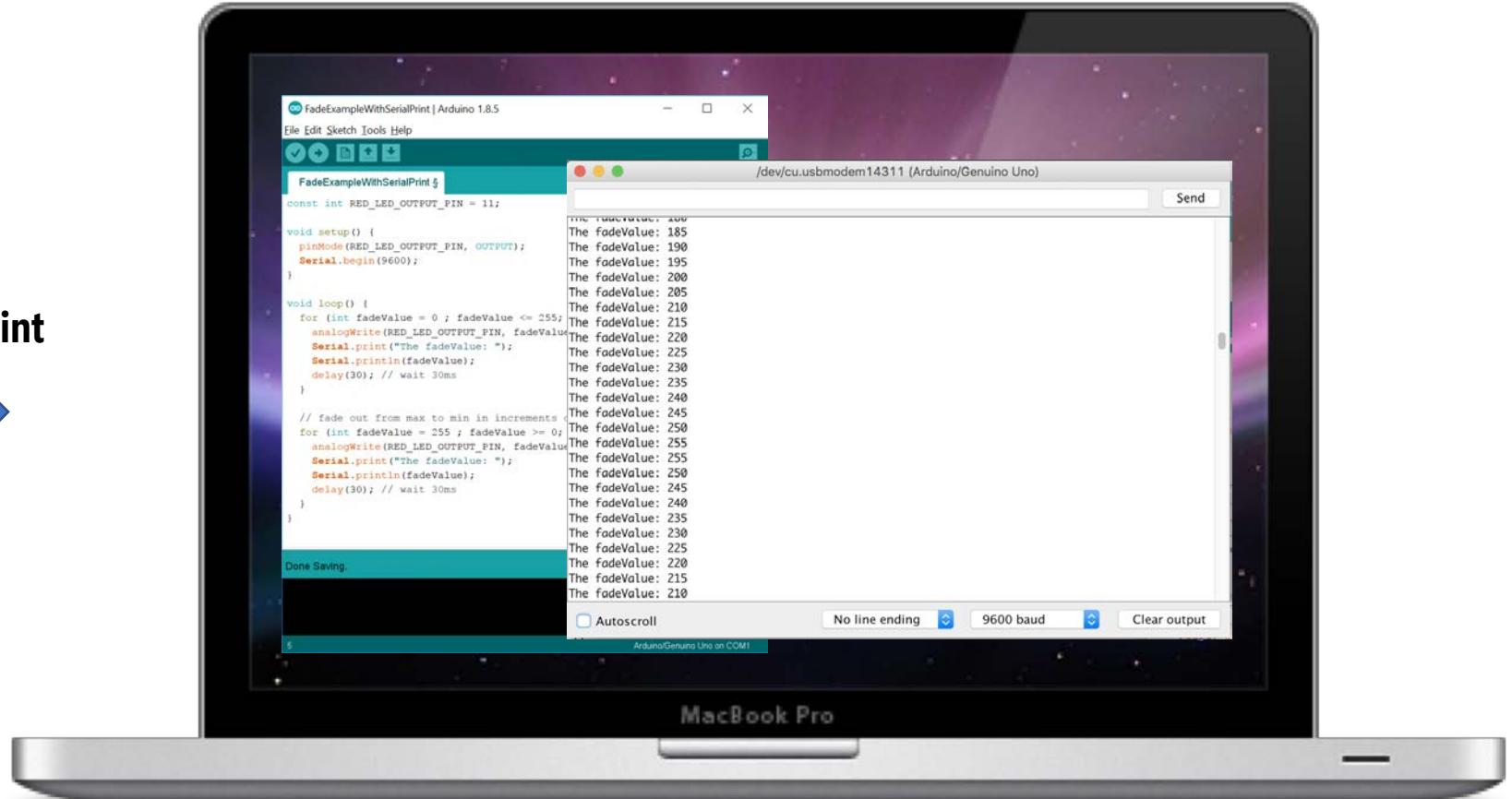


## SERIAL MONITOR

# CONCEPTUALLY UNDERSTANDING SERIAL.PRINT



Serial.print  
→



Serial.print sends data over the USB port (via the Serial protocol), which can be displayed by any terminal program including Arduino's built-in Serial Monitor and Serial Plotter.

# Serial.print(val)

Prints data to the serial port.

You must add this line of code to setup() to initialize the serial port

```
Serial.begin(9600);
```

## Syntax

```
Serial.print(val)
```

```
Serial.print(val, format)
```

## Parameters

val: the value to print

format: specifies formatting, see documentation

# Serial.print (val)

Prints data to the serial port.

You must add this line of code to setup() to initialize the serial port

```
Serial.begin( 9600 );
```

## Syntax

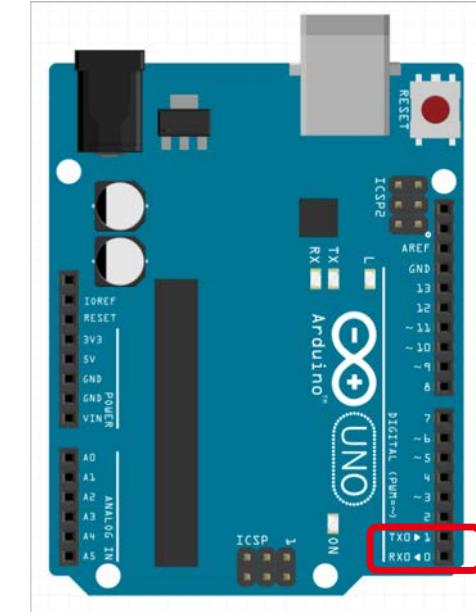
```
Serial.print(val)
```

```
Serial.print(val, format)
```

## Parameters

val: the value to print

format: specifies formatting, see documentation



As soon as you activate Serial.begin, pins 0 and 1 are used for serial communication and cannot be used for anything else!

## SERIAL MONITOR

# TRY IT YOURSELF!

FadeExampleWithSerialPrint | Arduino 1.8.5

File Edit Sketch Tools Help

FadeExampleWithSerialPrint §

```
const int RED_LED_OUTPUT_PIN = 11;

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

void loop() {
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    analogWrite(RED_LED_OUTPUT_PIN, fadeValue); // set the LED value
    Serial.print("The fadeValue: ");
    Serial.println(fadeValue);
    delay(30); // wait 30ms
  }

  // fade out from max to min in increments of 5 points:
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    analogWrite(RED_LED_OUTPUT_PIN, fadeValue); // set the LED value
    Serial.print("The fadeValue: ");
    Serial.println(fadeValue);
    delay(30); // wait 30ms
  }
}

Done Saving.
```

5

Arduino/Genuino Uno on COM1

/dev/cu.usbmodem14311 (Arduino/Genuino Uno)

The fadeValue: 100  
The fadeValue: 185  
The fadeValue: 190  
The fadeValue: 195  
The fadeValue: 200  
The fadeValue: 205  
The fadeValue: 210  
The fadeValue: 215  
The fadeValue: 220  
The fadeValue: 225  
The fadeValue: 230  
The fadeValue: 235  
The fadeValue: 240  
The fadeValue: 245  
The fadeValue: 250  
The fadeValue: 255  
The fadeValue: 255  
The fadeValue: 250  
The fadeValue: 245  
The fadeValue: 240  
The fadeValue: 235  
The fadeValue: 230  
The fadeValue: 225  
The fadeValue: 220  
The fadeValue: 215  
The fadeValue: 210

Autoscroll      No line ending      9600 baud      Clear output

DEBUGGING

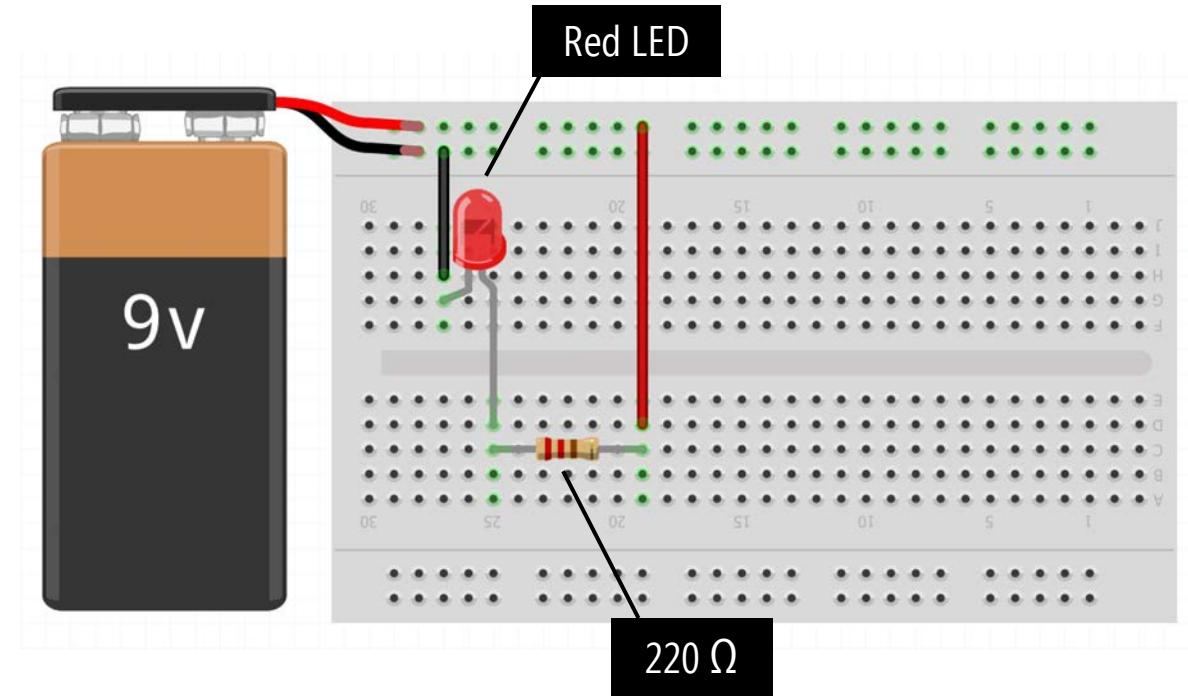
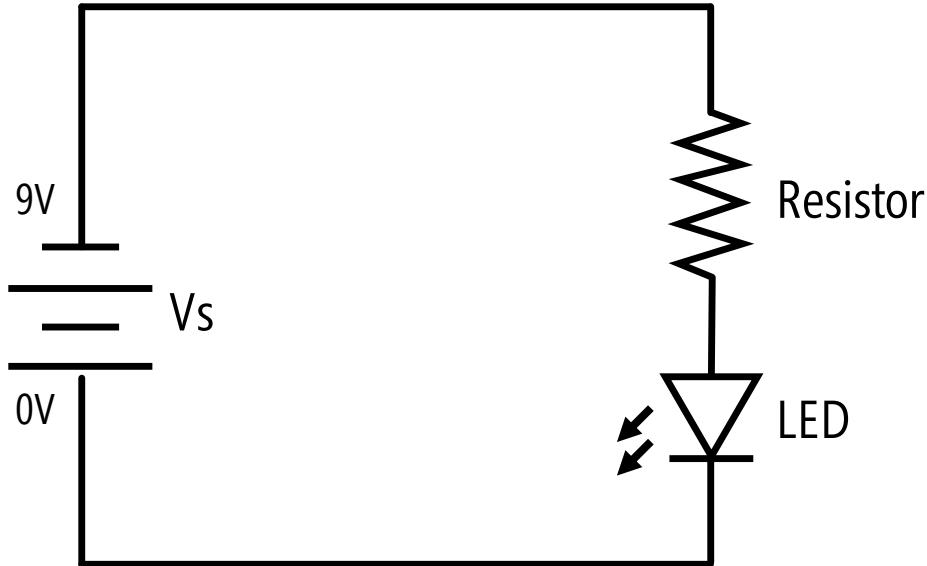
# DEBUGGING ARDUINO

Modularize.

Use Serial.println -> Serial Monitor and Serial Plotter

Use a multimeter

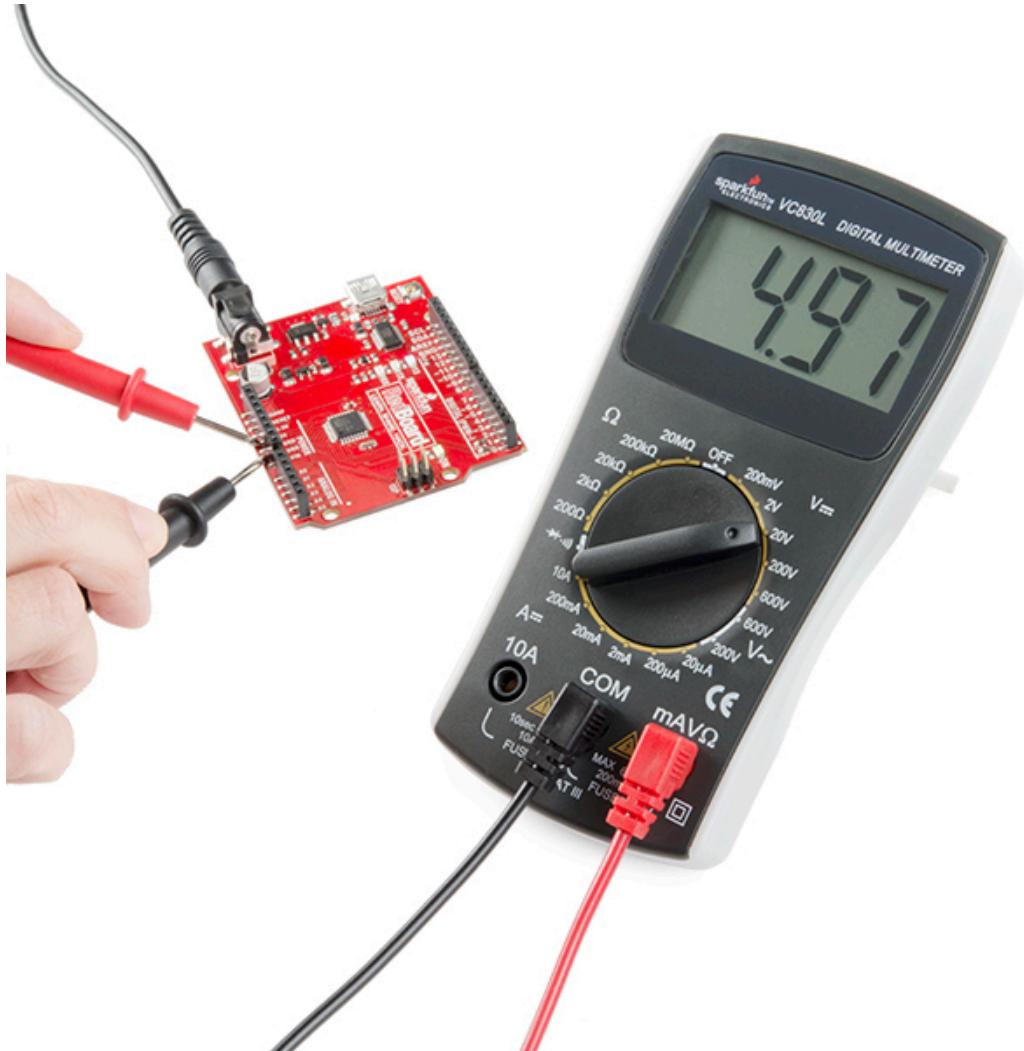
# EXERCISE: HOW MUCH CURRENT THROUGH LED?



Let's measure it empirically using a multimeter... but first, an introduction

MULTIMETERS

# YOU CAN MEASURE CIRCUITS WITH A MULTIMETER



Resistance

Voltage

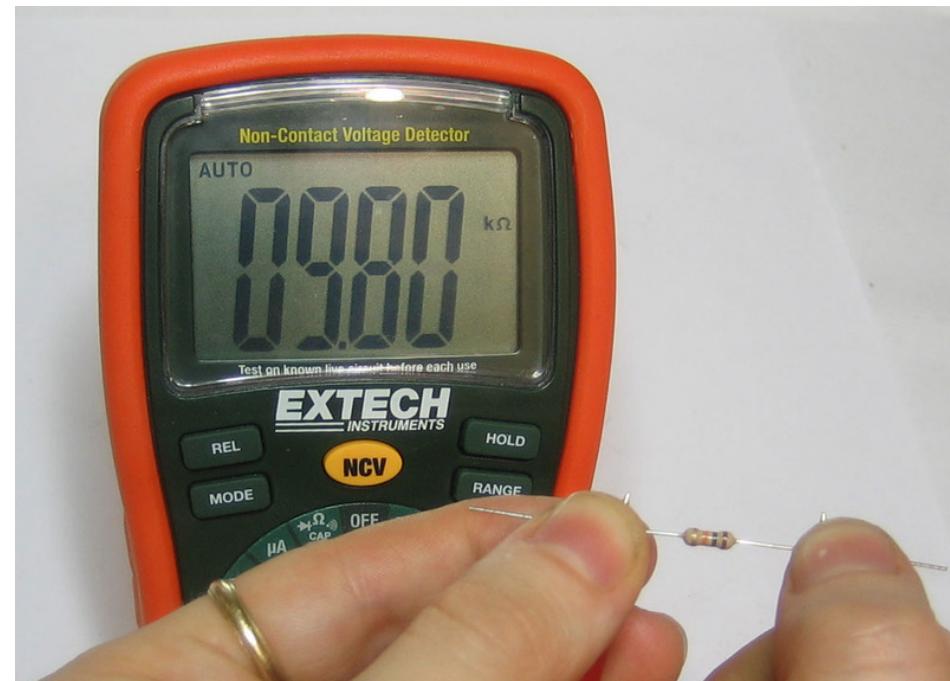
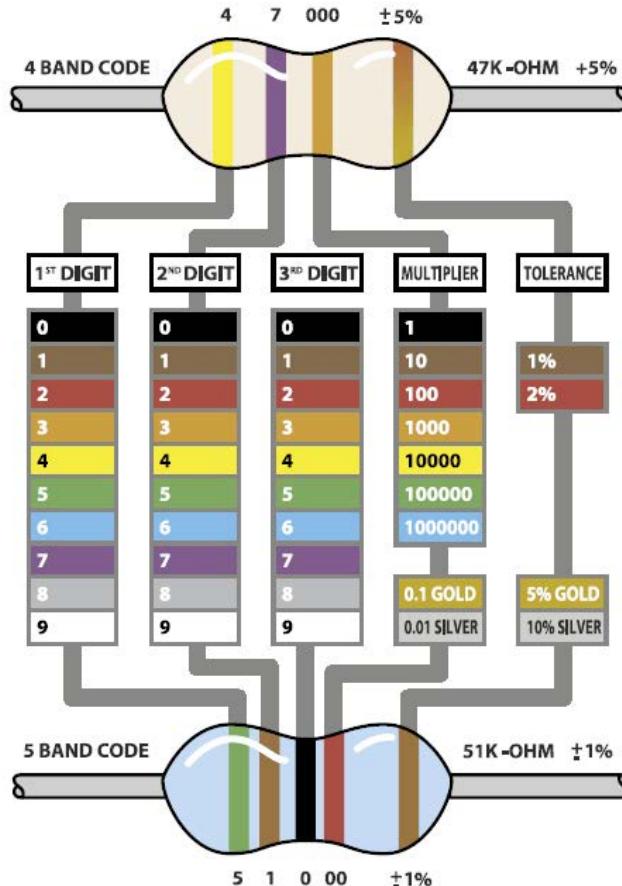
Current

Continuity (short testing)

## MULTIMETERS ACTIVITY

# MEASURE A RESISTOR

You can interpret color codes on a resistor or use a multimeter



This 10KΩ resistor is really 9.80K Ω

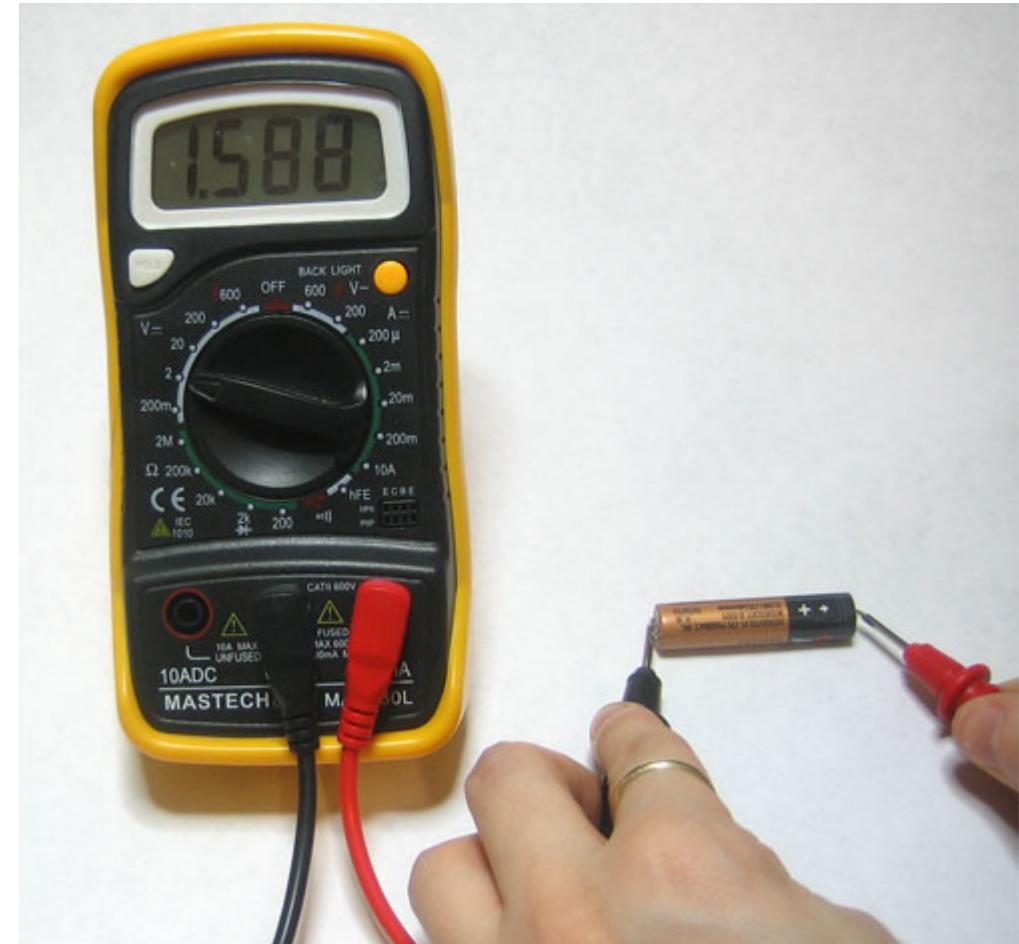
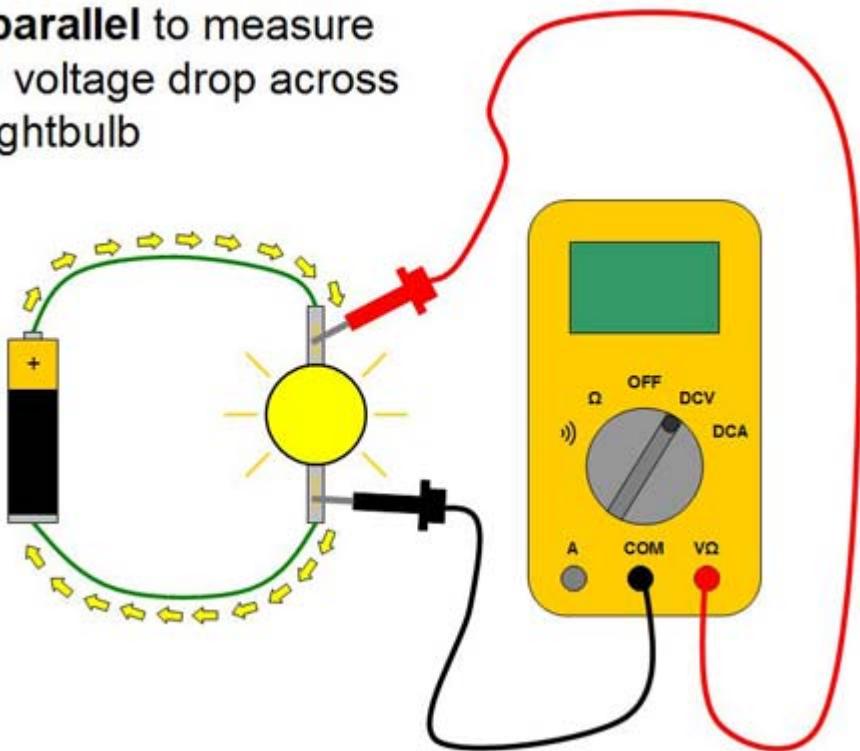
## MULTIMETERS

# EXERCISE: MEASURE RESISTANCE



# MEASURING VOLTAGE

Connect a multimeter in **parallel** to measure the voltage drop across a lightbulb



This 1.5V battery reads 1.588V. Why? The 1.5V written on the battery is a nominal voltage—or the “average” you may expect from the battery. In reality, an alkaline battery starts out higher, then slowly drifts down to 1.3V, then finally to 1.0V and even lower.

## MULTIMETERS

# MEASURING VOLTAGE: SWITCH MULTIMETER TO V



**Figure 1-54.** Each meter has a different way to measure volts DC. The manually adjusted meter (top) requires you to move a slider switch to "DC" and then choose the highest voltage you want to measure: In this case, the selected voltage is 20 (because 2 would be too low). Using the autoranging RadioShack meter, you set it to "V" and the meter will figure out which range to use.



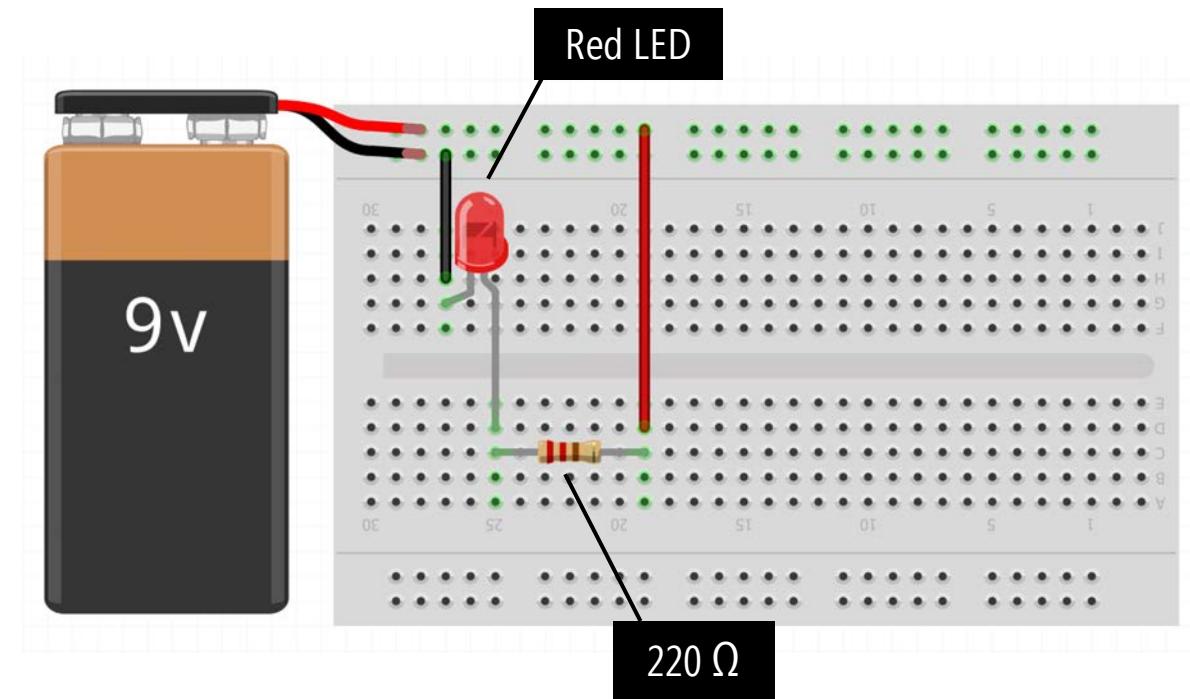
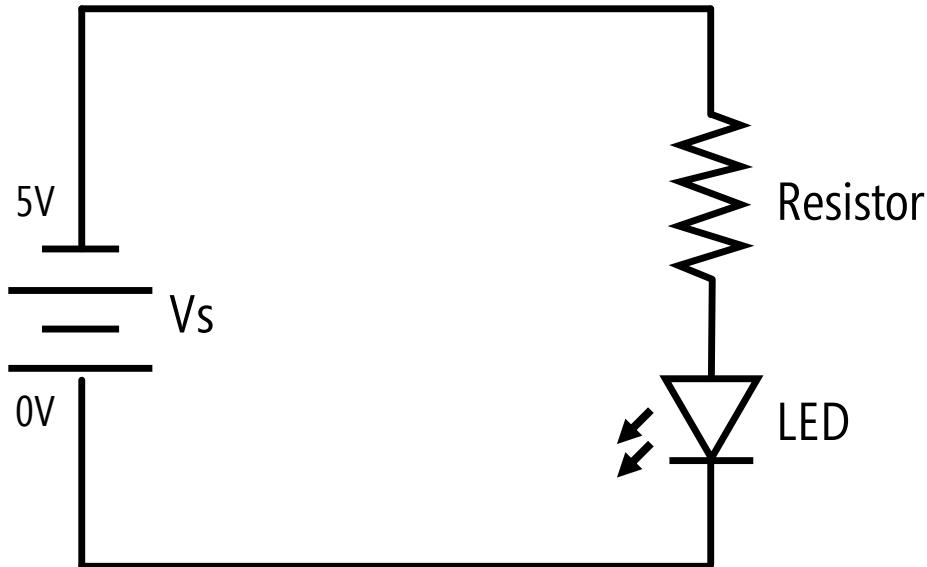
**Figure 1-53**



**Figure 1-52**

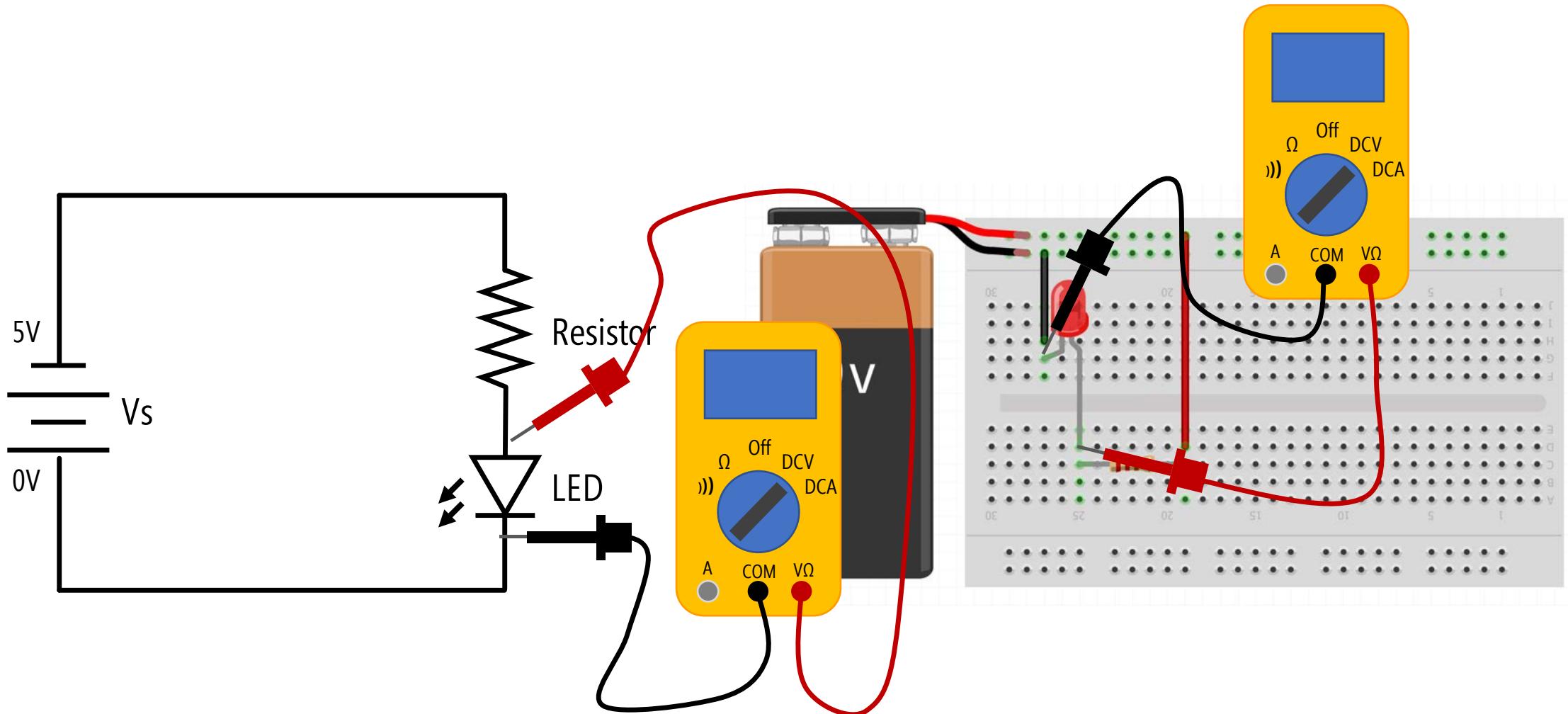
MULTIMETERS

# EXERCISE: WHAT'S THE VOLTAGE DROP ACROSS THE LED?



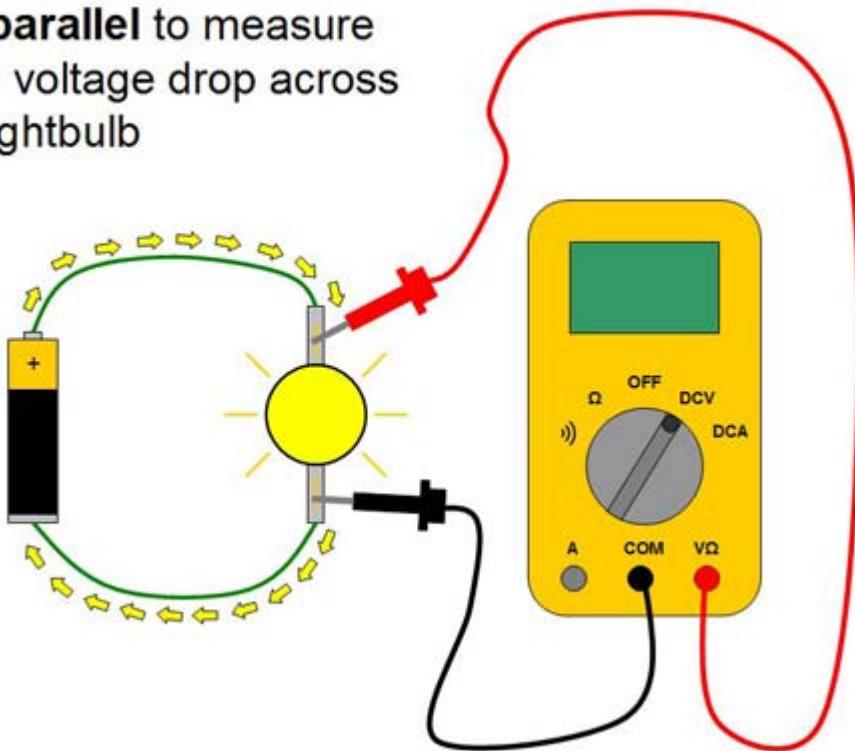
MULTIMETERS

# EXERCISE: WHAT'S THE VOLTAGE DROP ACROSS THE LED?

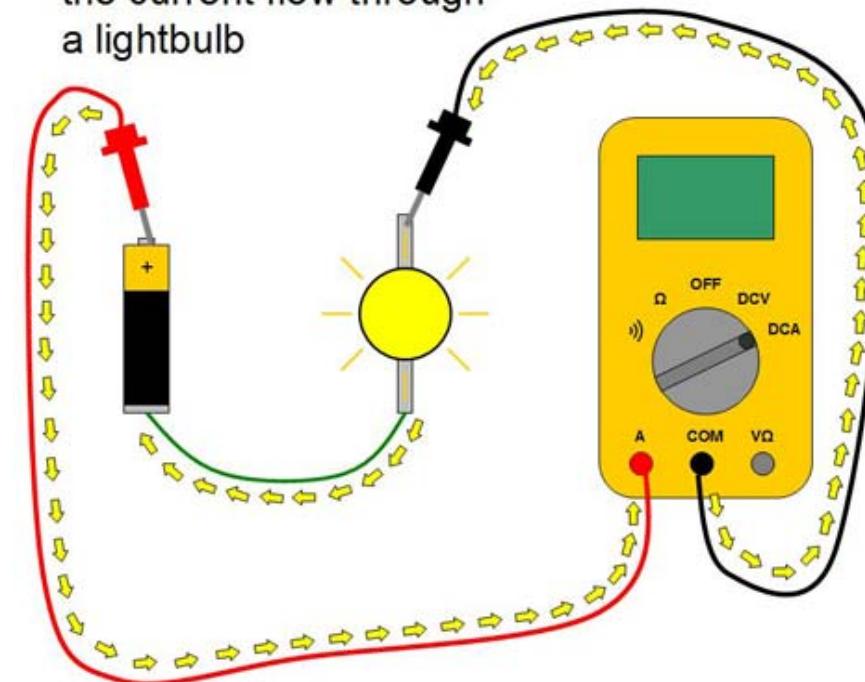


# MEASURING VOLTAGE VS. CURRENT

Connect a multimeter in **parallel** to measure the voltage drop across a lightbulb



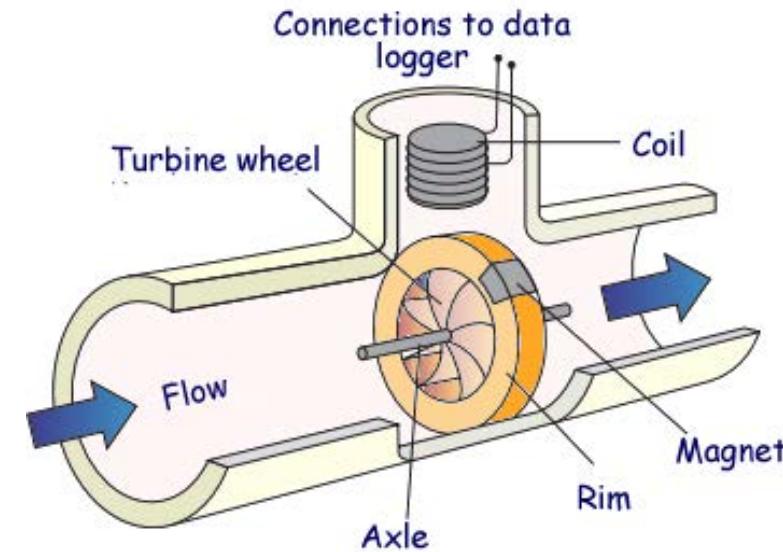
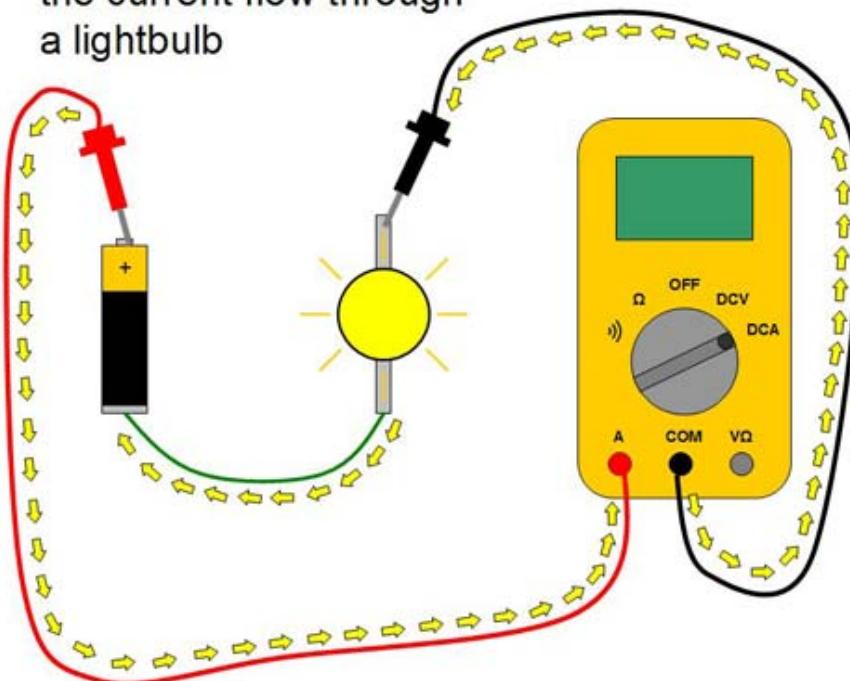
Connect a multimeter in **series** to measure the current flow through a lightbulb



# MEASURING CURRENT

When measuring current, think of the multimeter as a little turbine sensor for electrons. It needs to be inline (in series) in order to measure flow!

Connect a multimeter in **series** to measure the current flow through a lightbulb



# MEASURING CURRENT: CONFIGURE FOR MEASUREMENT



Figure 1-58. Any meter will blow its internal fuse if you try to make it measure too high an amperage. In our circuit, this is not a risk as long as you keep the potentiometer in the middle of its range. Choose "mA" for millamps and remember that the meter displays numbers that mean thousandths of an amp.

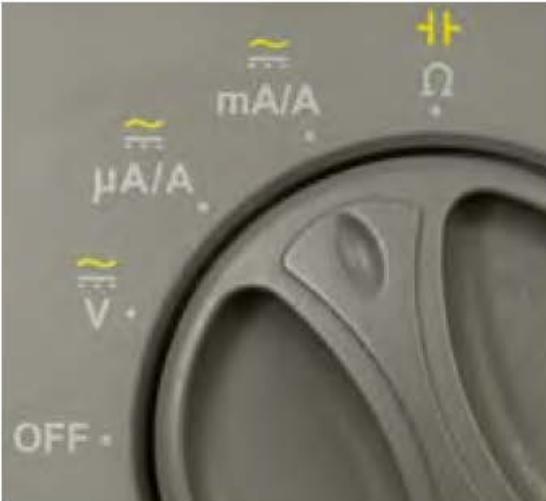
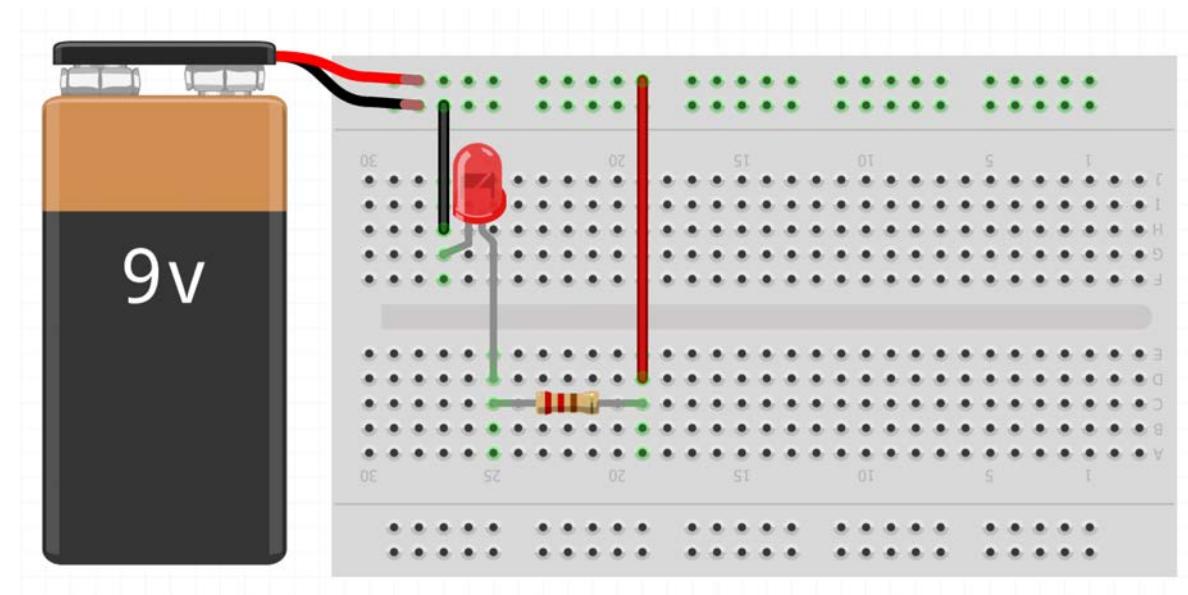
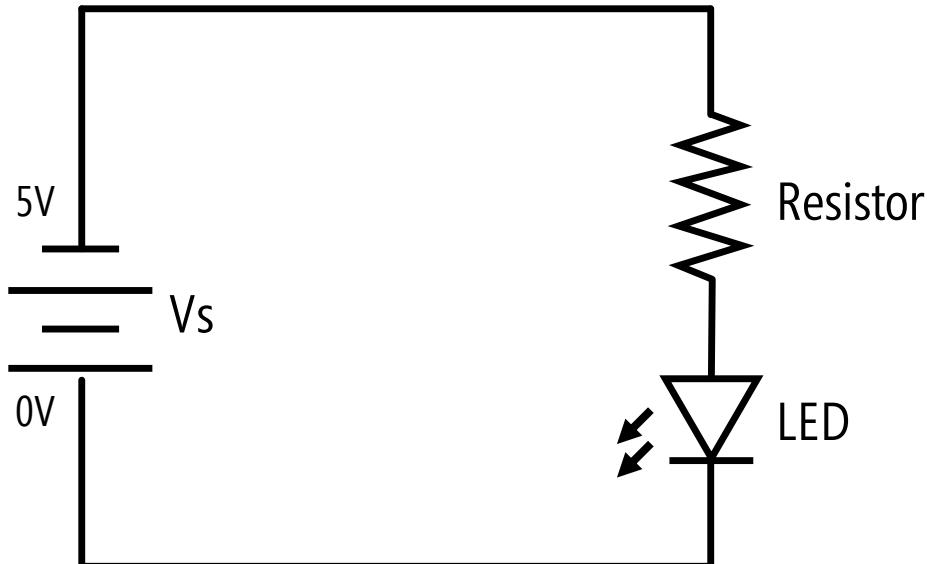


Figure 1-59



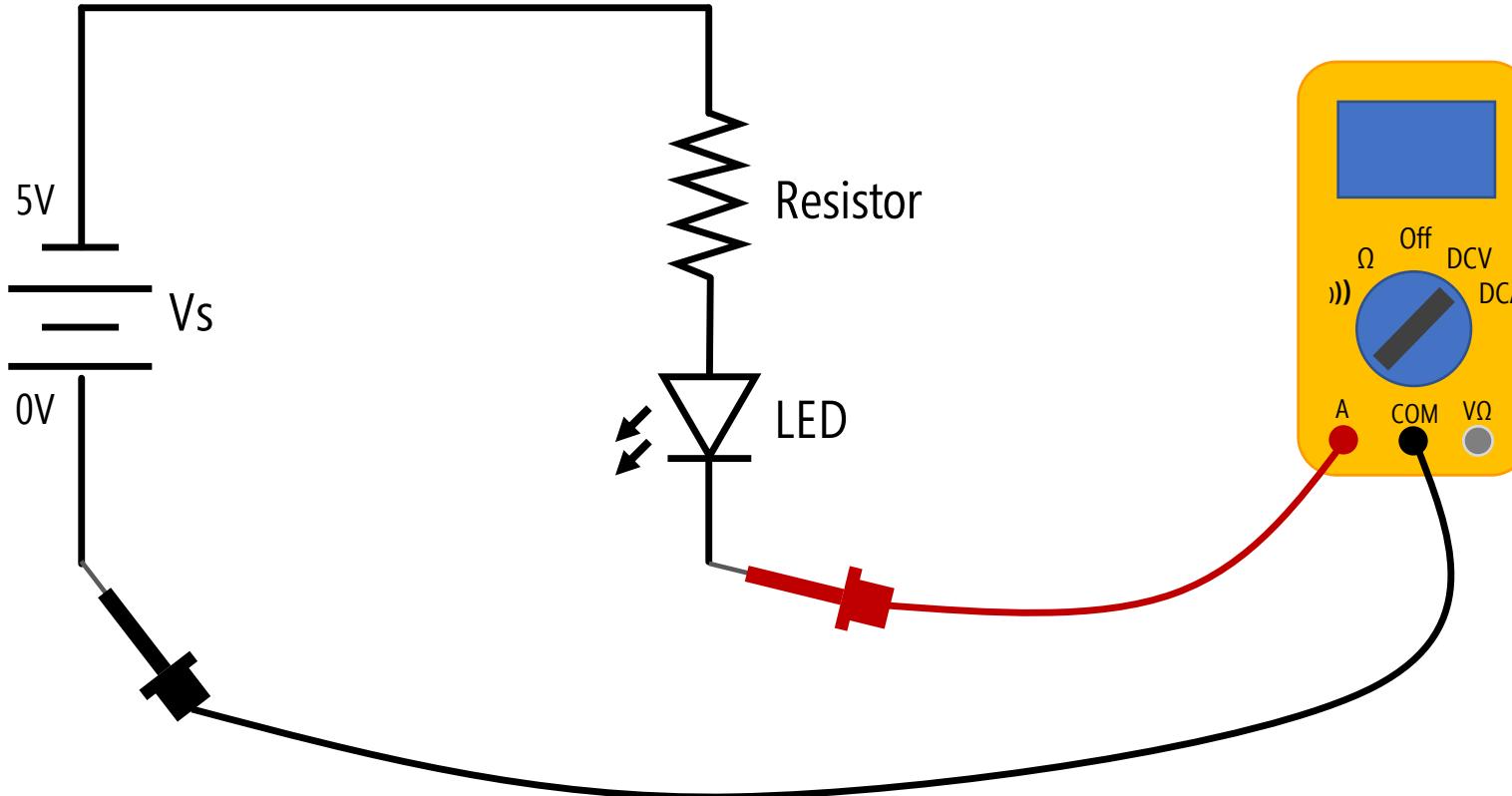
# EXERCISE: HOW MUCH CURRENT THROUGH LED?

You'll have to slightly rewire this circuit so that the current passes through the multimeter



# EXERCISE: HOW MUCH CURRENT THROUGH LED?

You'll have to slightly rewire this circuit so that the current passes through the multimeter



## RESOURCES

# HELPFUL MULTIMETER RESOURCES

### [Adafruit's Multimeter Tutorial](#)

Adafruit online tutorial series

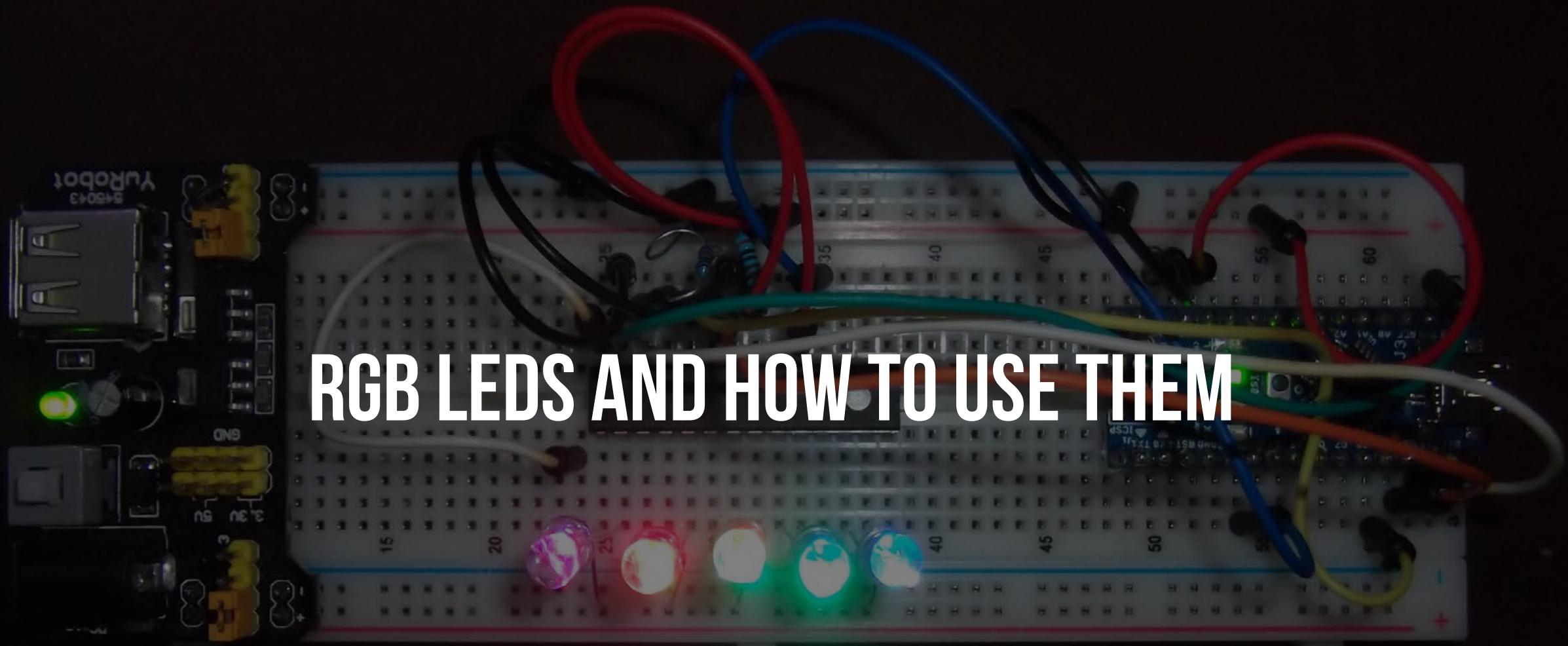
### [Collin's Lab: Multimeters Video](#)

Collin Cunningham, noted tutorialist

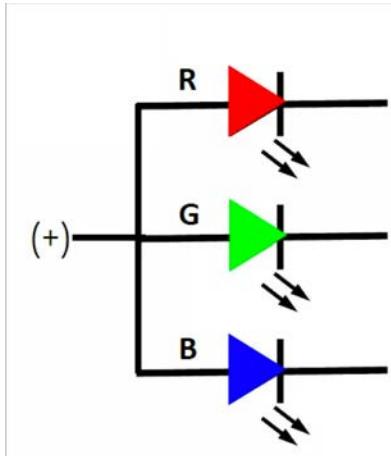
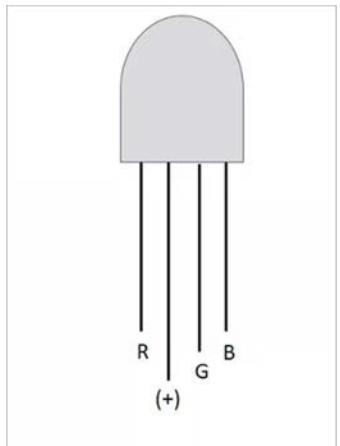
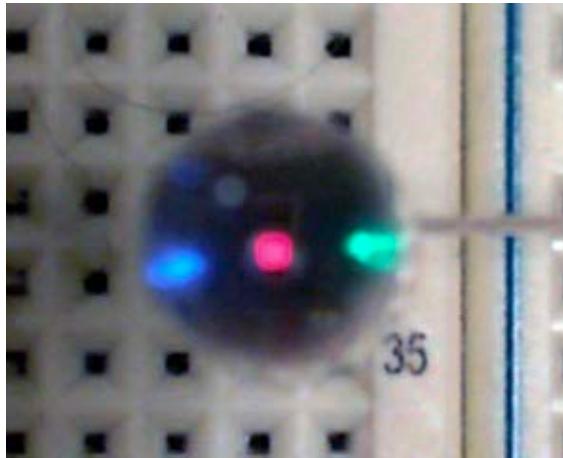
### [Make Magazine's How to Use a Multimeter](#)

Make Magazine

# RGB LEDS AND HOW TO USE THEM



# WHAT ARE RGB LEDs

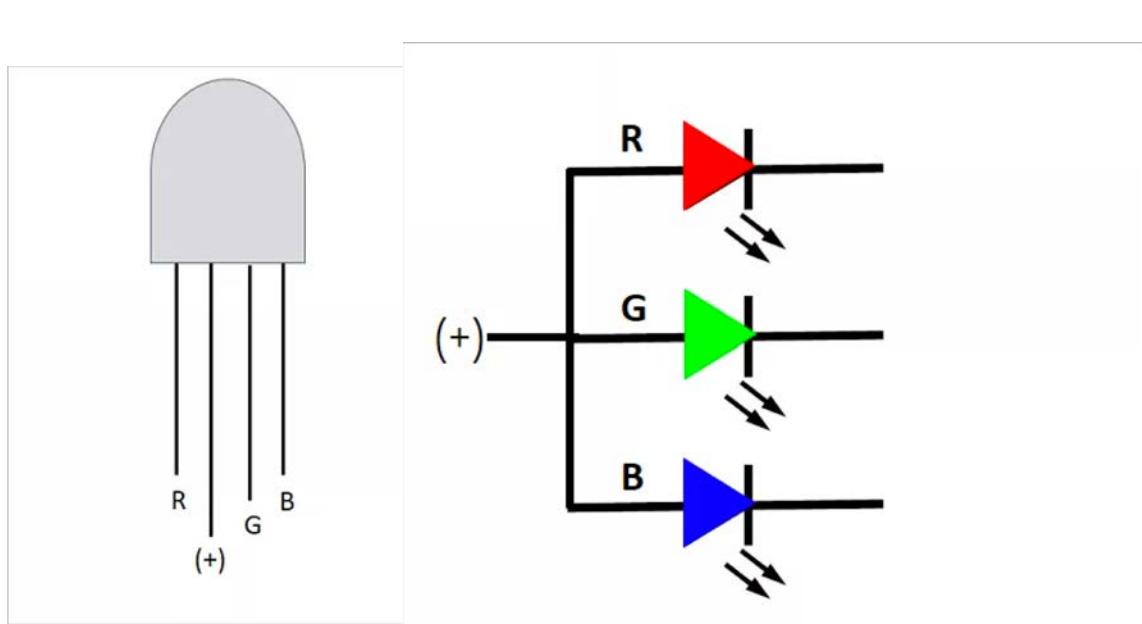


RGB LEDs are actually **three LEDs** in one: a **red** LED, a **green** LED, and a **blue** LED

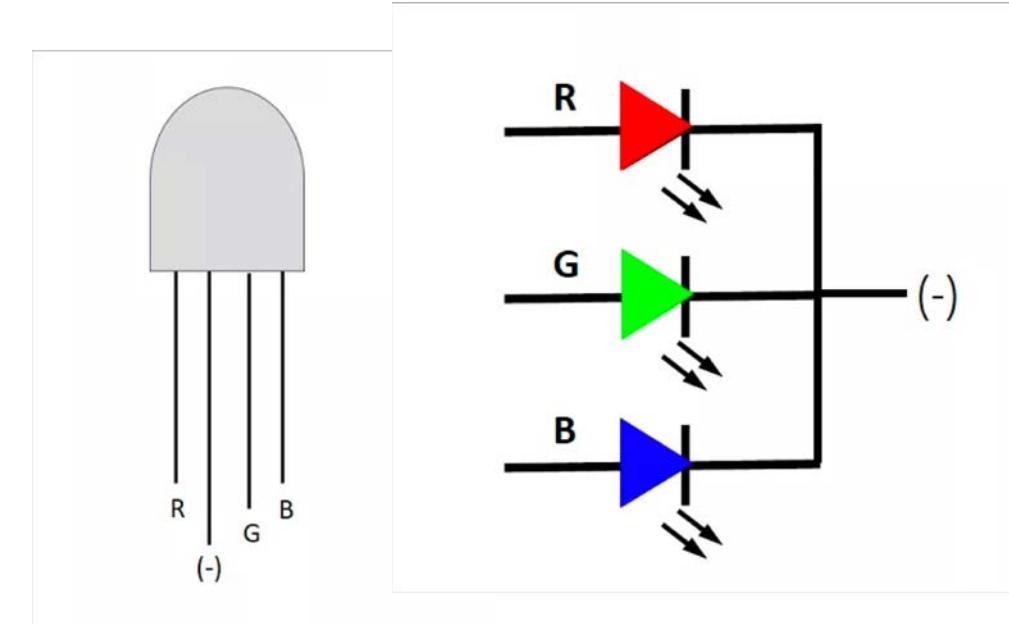
You can **control the color** by setting the input voltages of each R, G, and B leg to a different value.

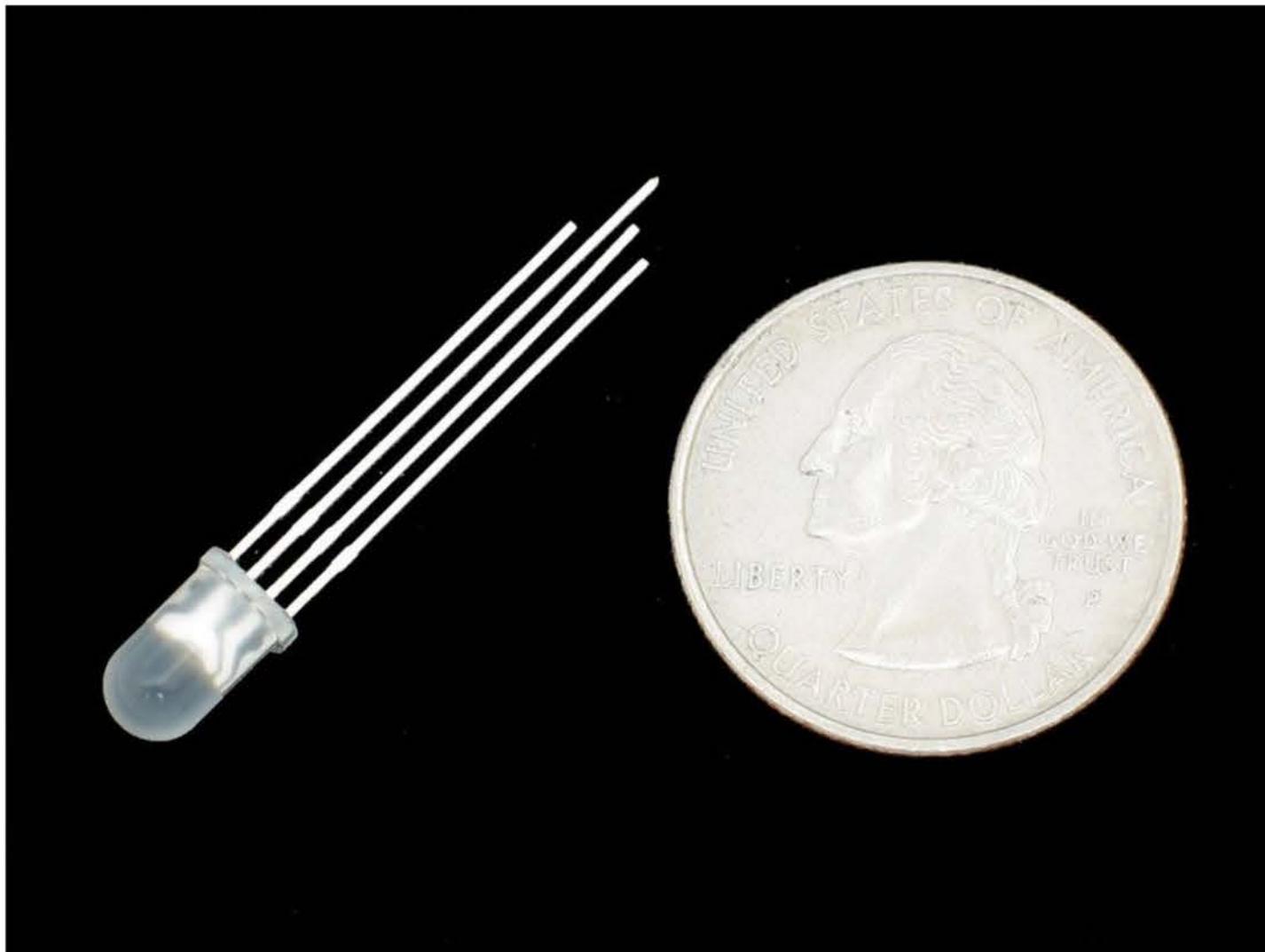
# COMMON ANODE VS. COMMON CATHODE

Common Anode (+)



Common Cathode (-)



[LEDS](#) / [BARE LEDS](#) / DIFFUSED RGB (TRI-COLOR) LED

## Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

**\$2.00**

49 IN STOCK

1

[ADD TO CART](#)

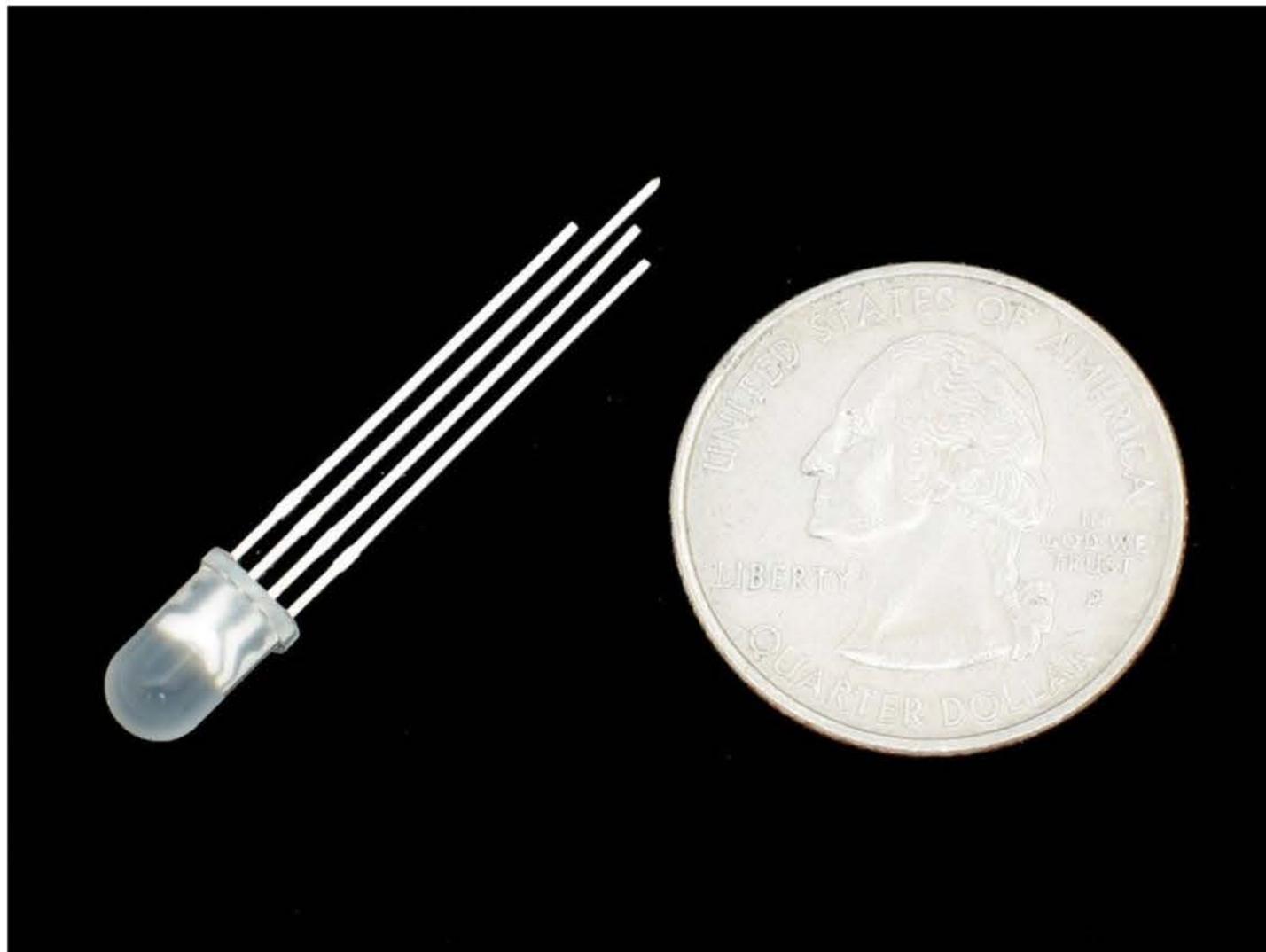
QTY DISCOUNT

1-9 \$2.00

10-49 \$1.75

50+ \$1.50

[ADD TO WISHLIST](#)[DESCRIPTION](#)[TECHNICAL DETAILS](#)[LEARN](#)

[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 DISCOUNT

1-9 \$2.00

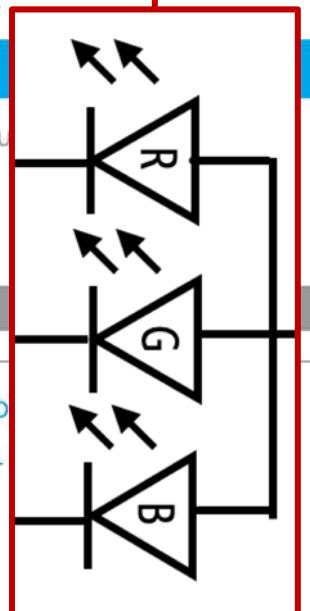
10-49 \$1.75

50+ \$1.50

DESCRIPTION

TECHNICAL

LEARN





## Diffused RGB (tri-color) LED Common Anode

PRODUCT ID: 159

\$2.00

49 IN STOCK

1

QTY DISCOUNT

1-9 \$2.00

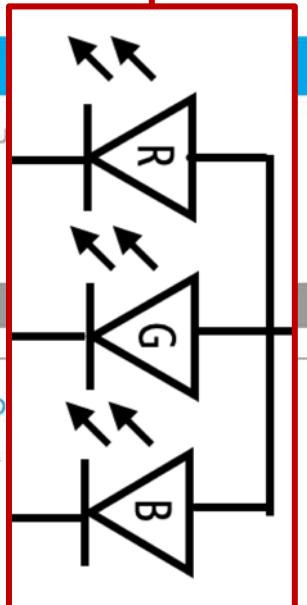
10-49 \$1.75

50+ \$1.50

DESCRIPTION

TECHNICAL

LEARN





The **Common Anode** means that all three embedded LEDs share an Anode leg (the long leg) and this leg should be connected to positive (the direction with highest voltage potential).

When using a new part, **always consult** with the part website or datasheet

## Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

\$2.00

49 IN STOCK

1

QTY DISCOUNT

1-9 \$2.00

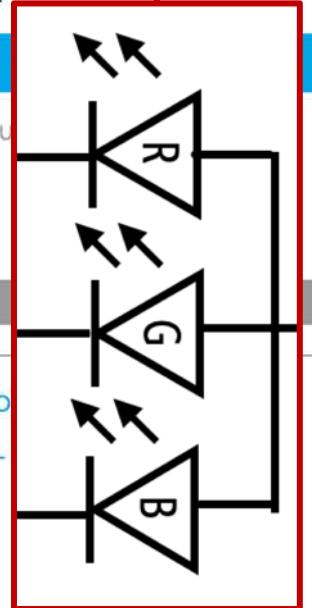
10-49 \$1.75

50+ \$1.50

DESCRIPTION

TECHNICAL

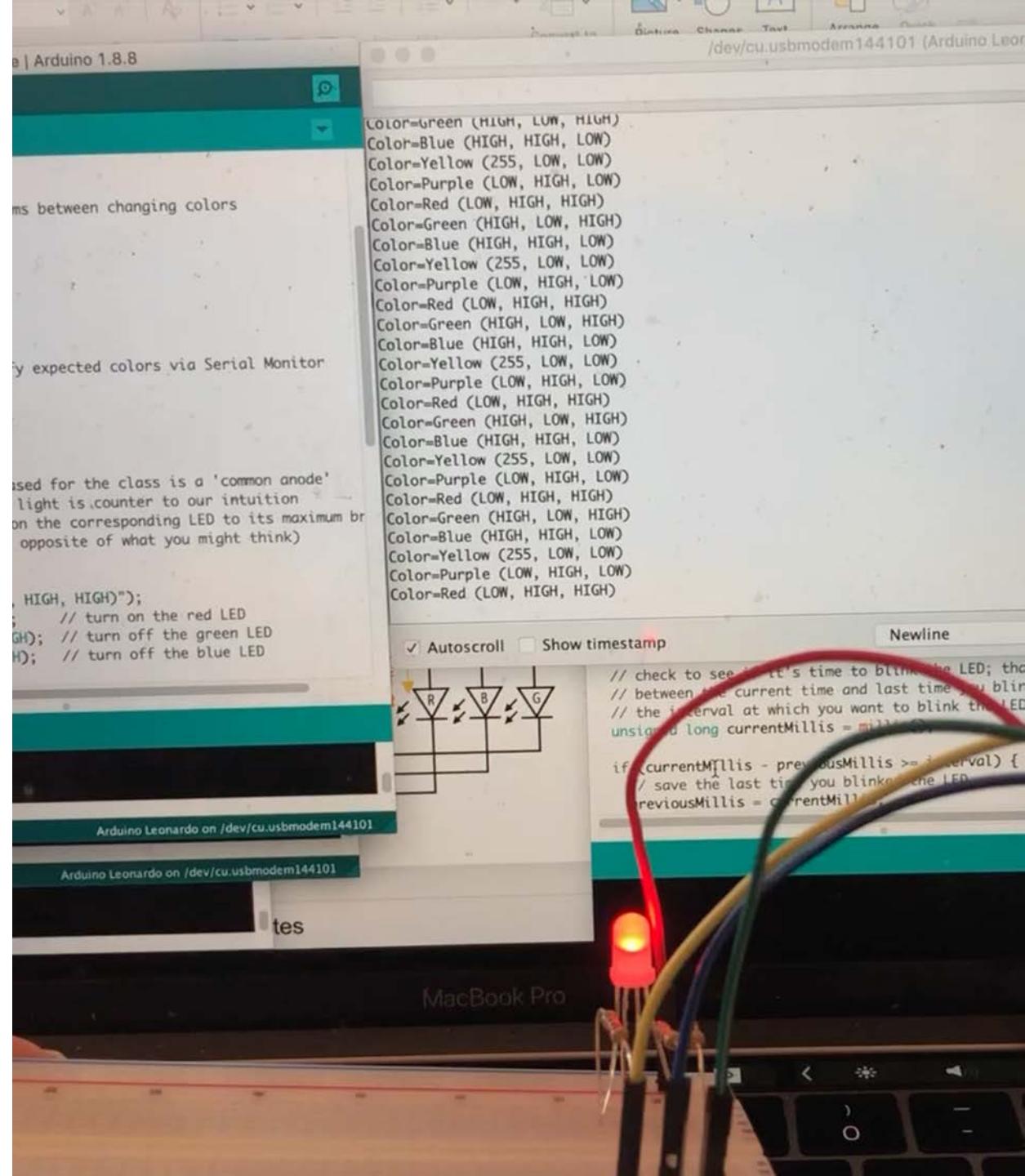
LEARN



## ACTIVITY

# FLASH RGB LED COLORS

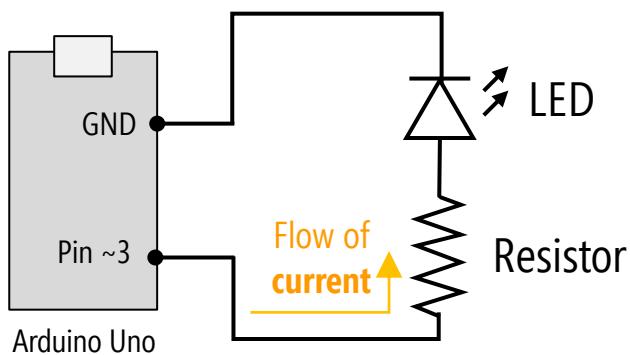
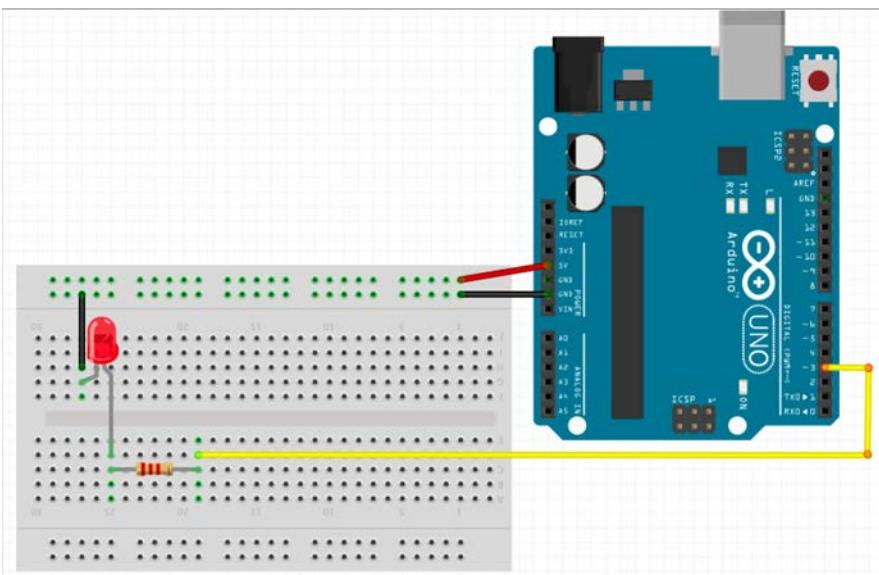
Build a circuit & write code to flash through different RGB LED colors



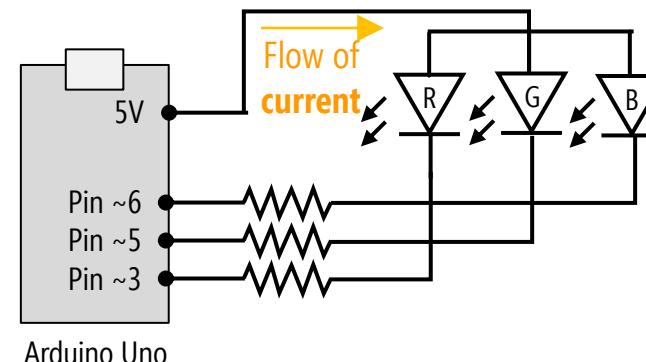
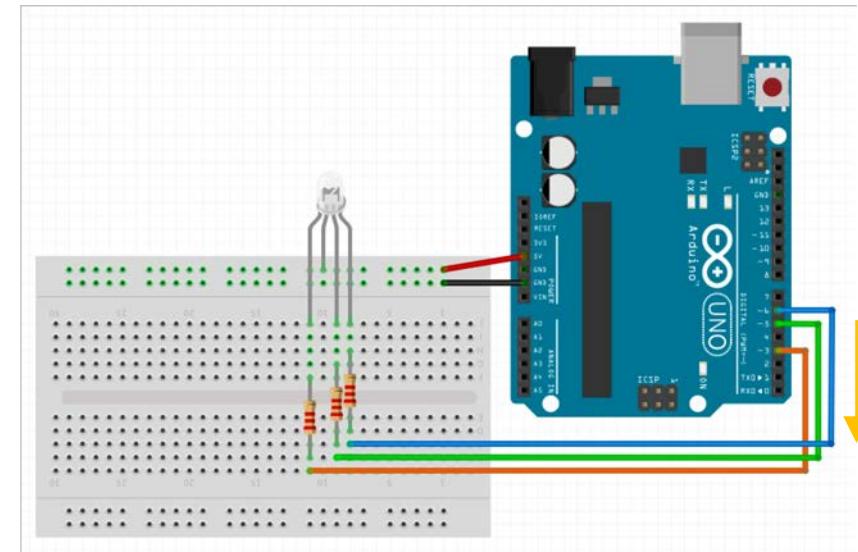
RGB LEDs

# 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

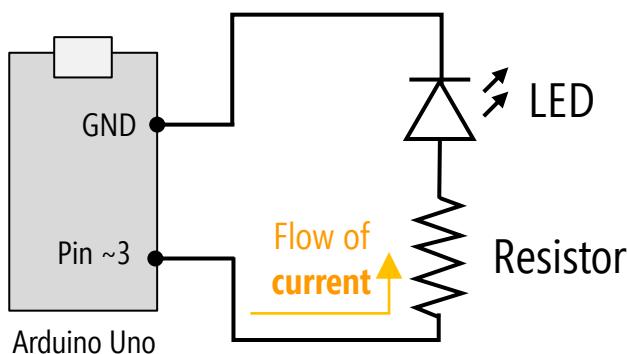
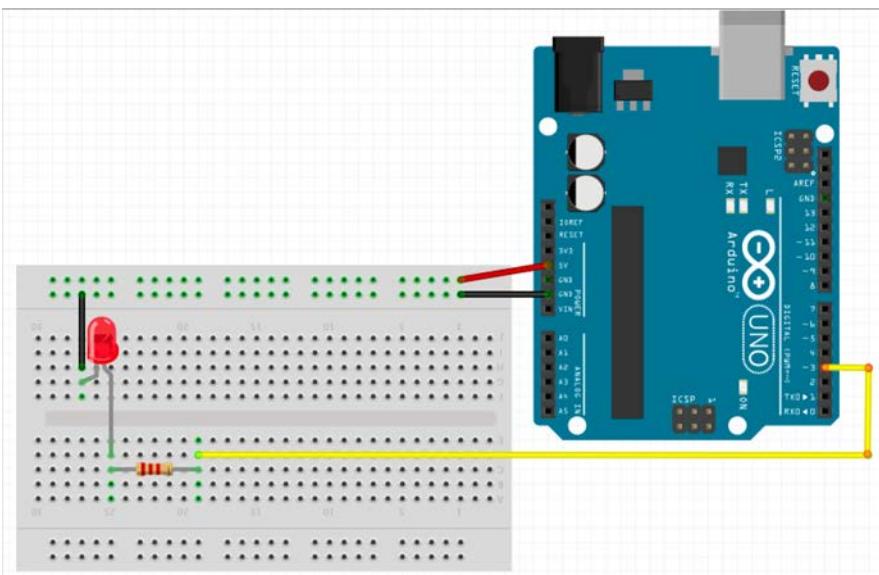


**Flow of current**  
Because this is a  
“common anode”  
RGB. We set these  
pins to zero to  
turn on the light  
and 5V to turn off  
the light!

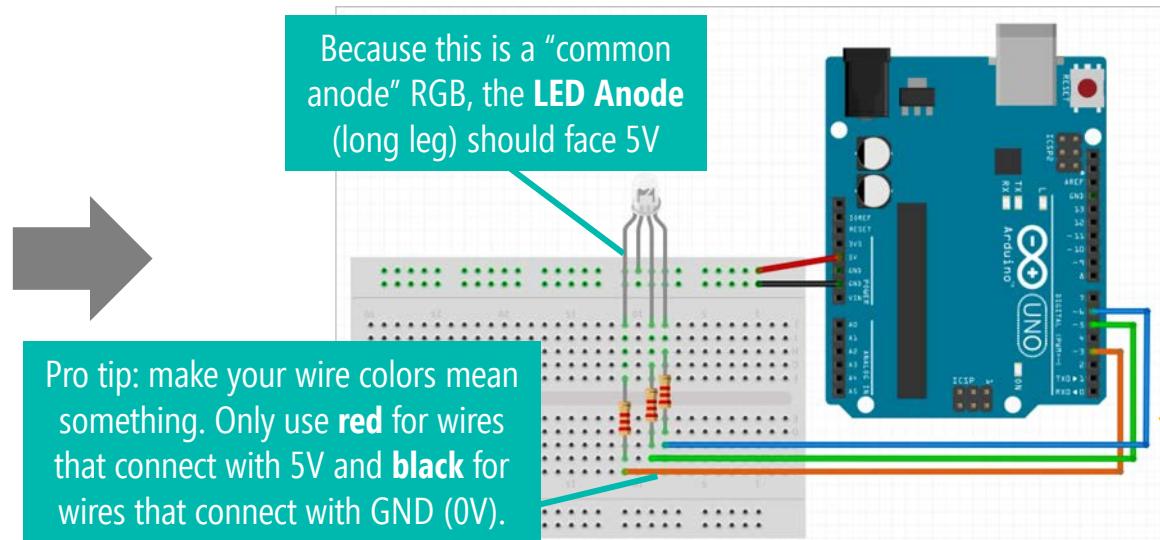
## RGB LEDs

# USING AN RGB LED: THE CIRCUIT

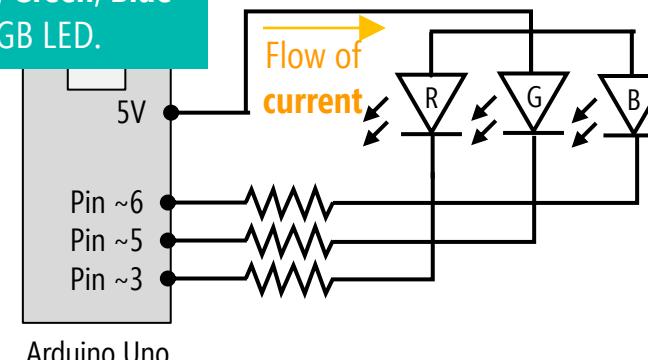
**Old Circuit:** Turn LED on/off via pin 3



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



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.

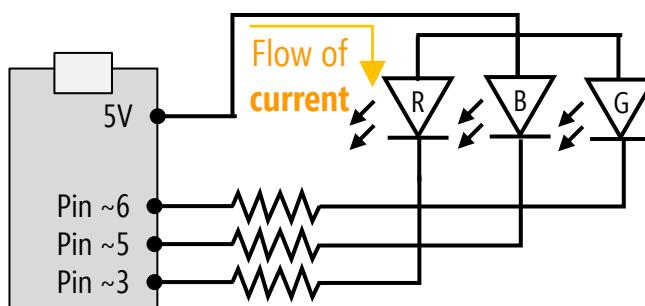
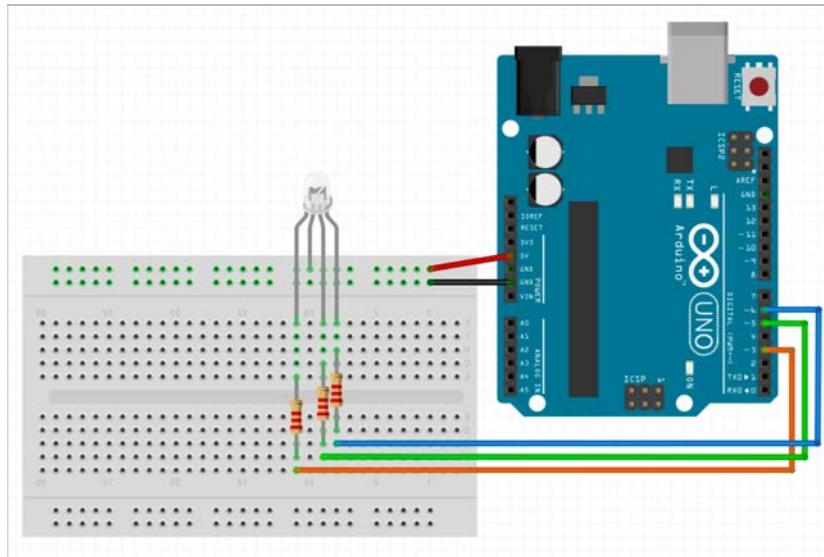


**Flow of current**  
Because this is a “common anode” RGB. We set these pins to zero to turn on the light and 5V to turn off the light!

## DIGITAL 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);

  // 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);

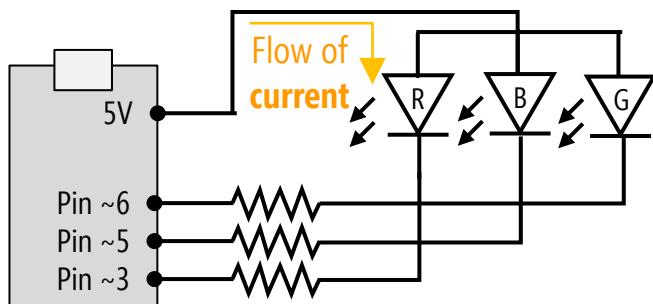
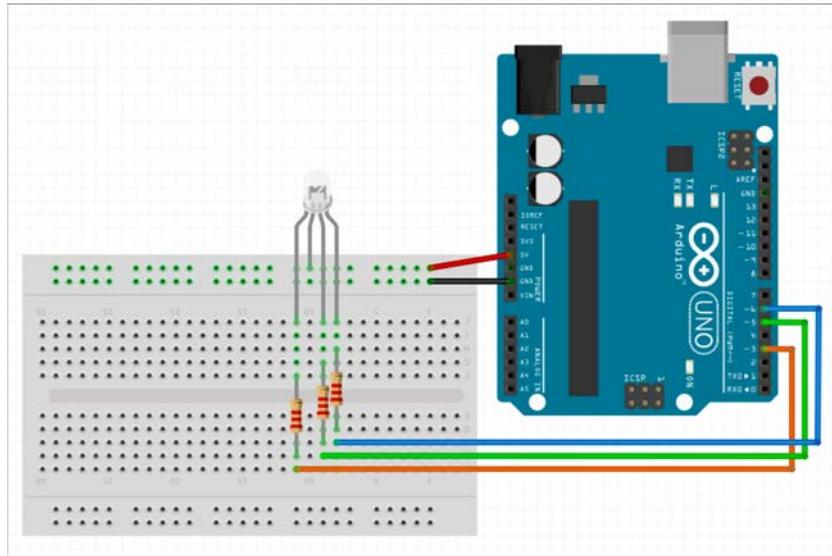
  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  delay(DELAY);

  // 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);
}
```

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

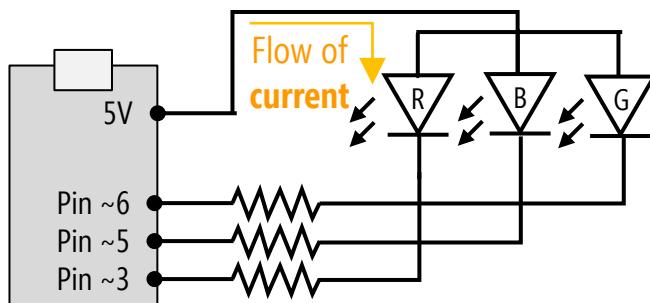
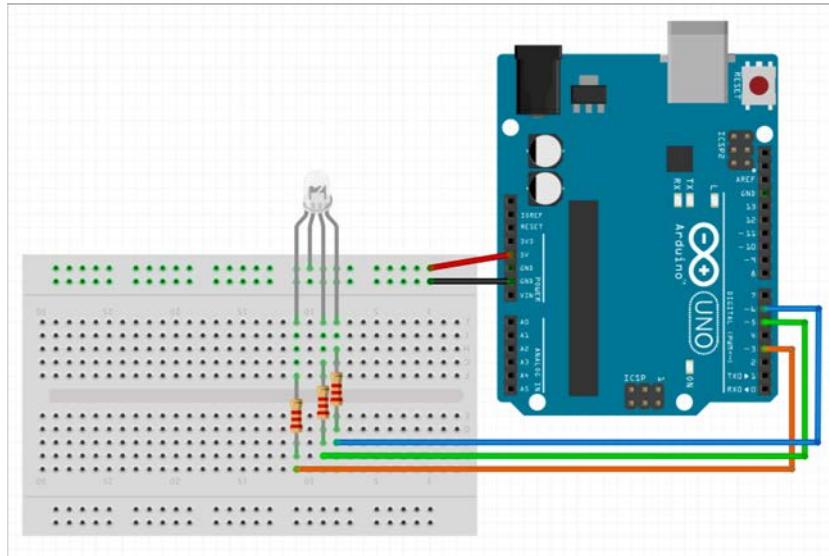
  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  delay(DELAY);

  // 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);
```

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

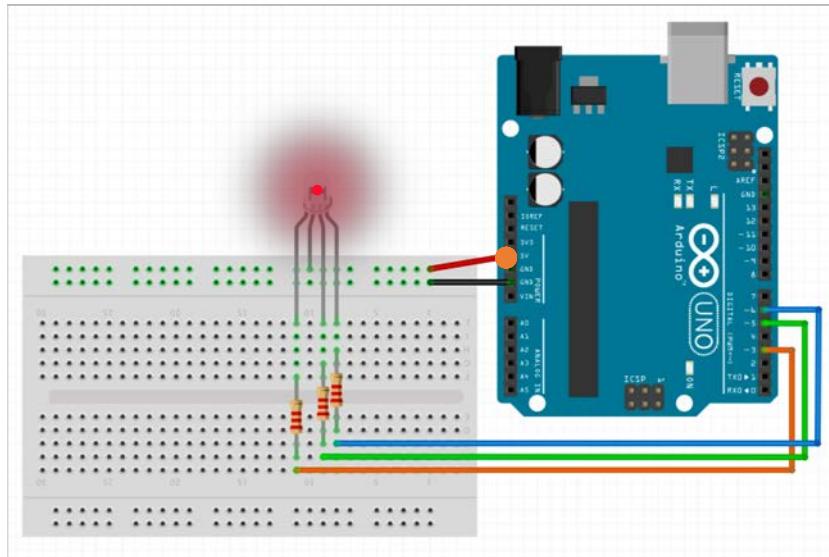
  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  delay(DELAY);

  // 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);
```

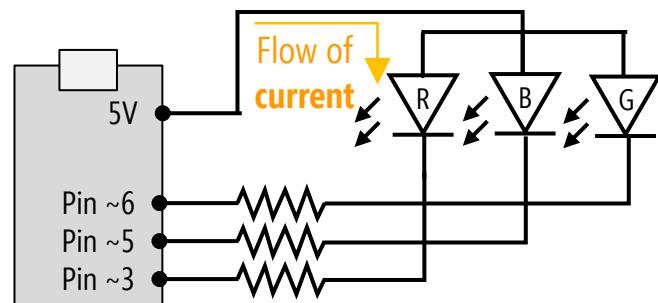
## DIGITAL OUTPUT

# 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



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);

    // 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);

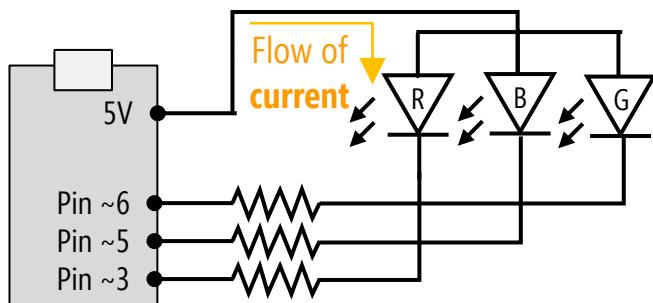
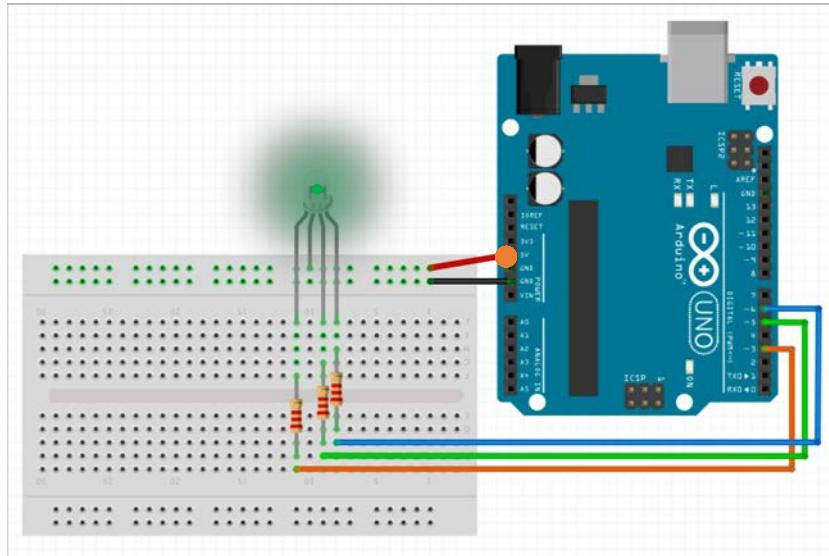
    // Set the RGB LED to yellow (by turning on green and blue!)
    Serial.println("Color=Yellow (255, LOW, LOW)");
    digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
    digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
    digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  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

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

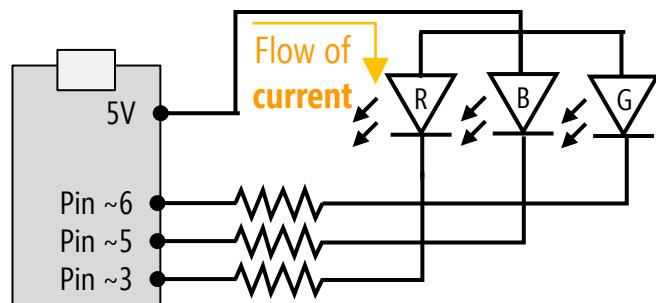
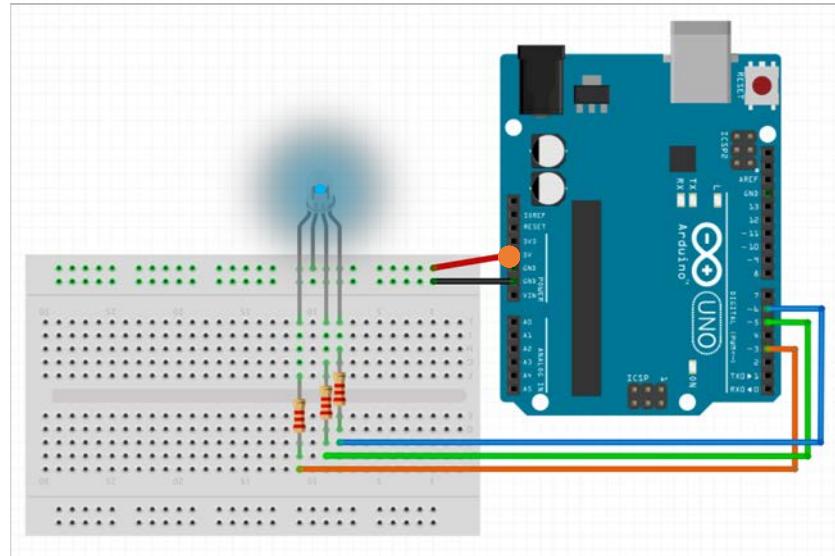
    // Set the RGB LED to yellow (by turning on green and blue!)
    Serial.println("Color=Yellow (255, LOW, LOW)");
    digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
    digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
    digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  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

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

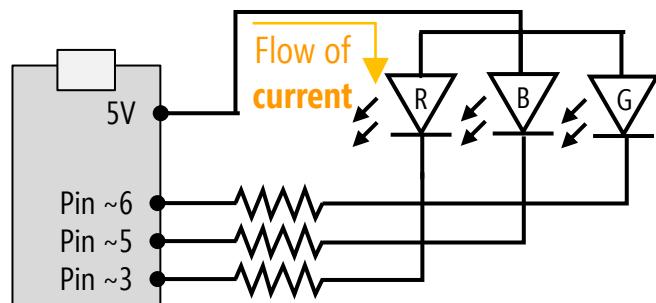
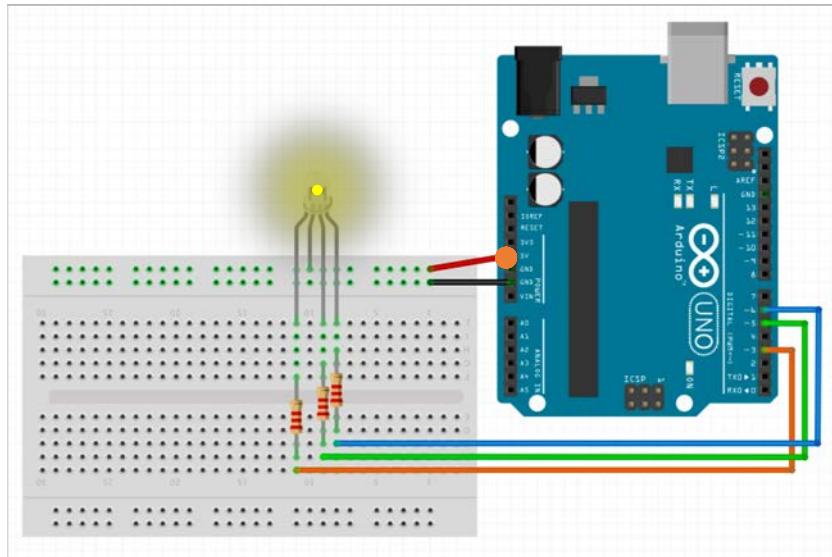
    // Set the RGB LED to yellow (by turning on green and blue!)
    Serial.println("Color=Yellow (255, LOW, LOW)");
    digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
    digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
    digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  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

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

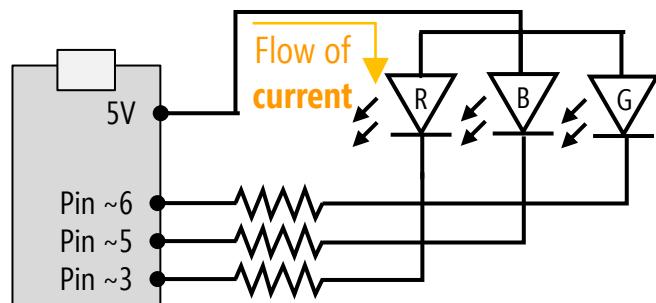
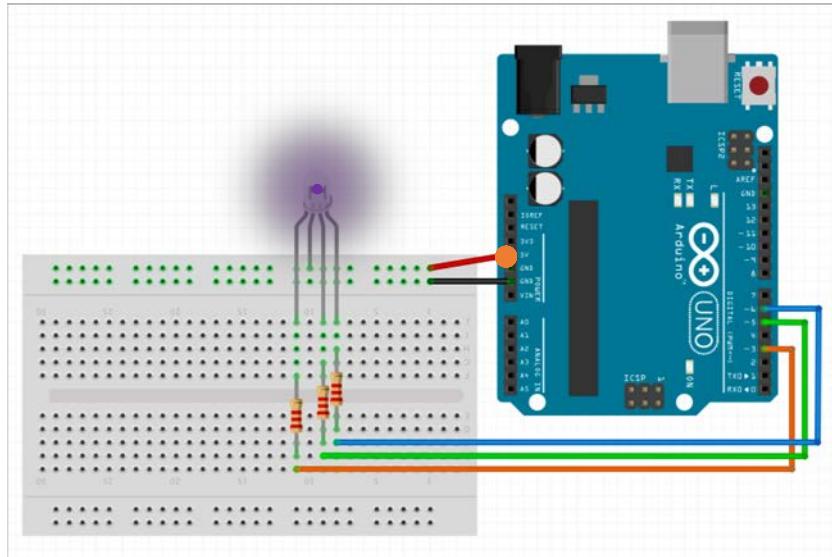
  // Set the RGB LED to yellow (by turning on green and blue!)
  Serial.println("Color=Yellow (255, LOW, LOW)");
  digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  delay(DELAY);

  // 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);
}
```

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

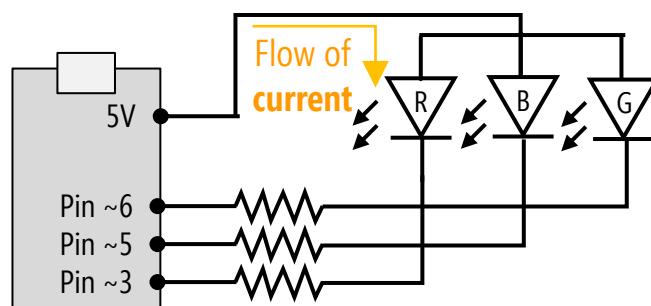
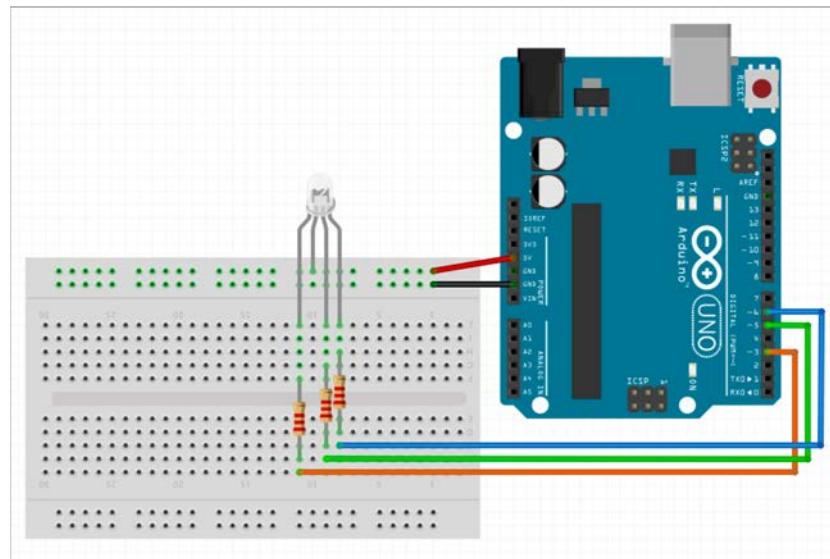
    // Set the RGB LED to yellow (by turning on green and blue!)
    Serial.println("Color=Yellow (255, LOW, LOW)");
    digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
    digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
    digitalWrite(RGB_BLUE_PIN, LOW); // turn on the blue LED
  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

## DIGITAL OUTPUT

# RGB LED WALKTHROUGH

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



Arduino Uno

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);

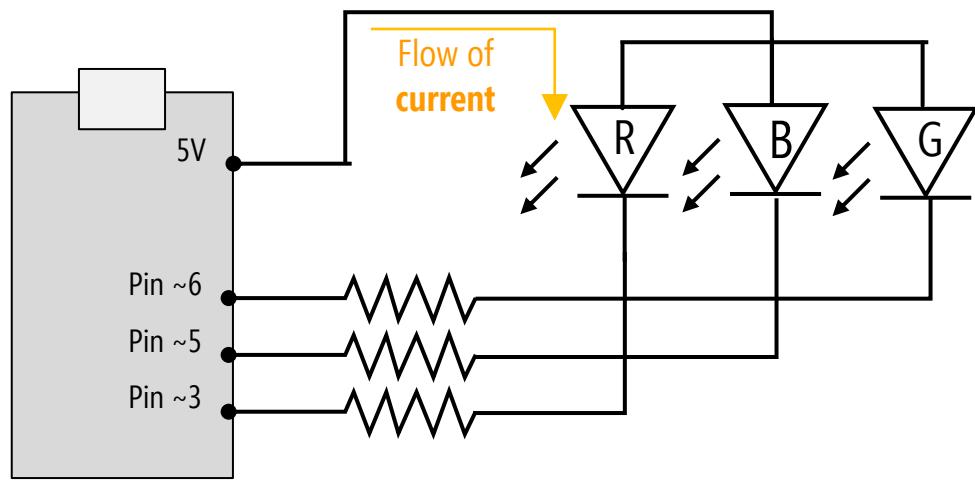
  // Set the RGB LED to green
  Serial.println("Color=Green (HIGH, LOW, HIGH)");
  digitalWrite(RGB_RED_PIN, HIGH); // turn off the red LED
  digitalWrite(RGB_GREEN_PIN, LOW); // turn on the green LED
  digitalWrite(RGB_BLUE_PIN, HIGH); // turn on the blue LED
  delay(DELAY);

  // 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



Arduino Uno

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

# TODAY'S LEARNING GOALS

Reinforce some **circuit concepts** from last week

What is **Arduino**? Both hardware and software

How to use **digital output** (*i.e.*, turning on/off an LED using digitalWrite)

How to **debug** using **Serial Monitor** and **multimeters**

How to use an **RGB LED** (if time)

# PHYSICAL COMPUTING 2: ARDUINO & BASIC OUTPUT

---

CSE 599 Prototyping Interactive Systems | Lecture 3 | April 9

**Jon Froehlich** • Jasper O'Leary (TA)