PHYSCOMP 2: ARDUINO & BASIC OUTPUT

CSE 599 Prototyping Interactive Systems | Lecture 3 | Oct 3

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

PHYSICAL COMPUTING 2

TODAY'S LEARNING GOALS

Reinforce some **circuit concepts** from last time

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**

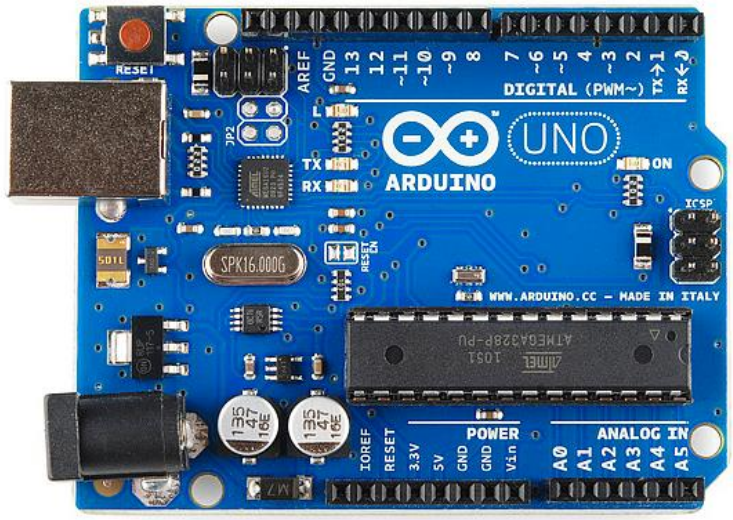
How to use an **RGB LED** (if time)

ARDUINO

WHAT IS ARDUINO?

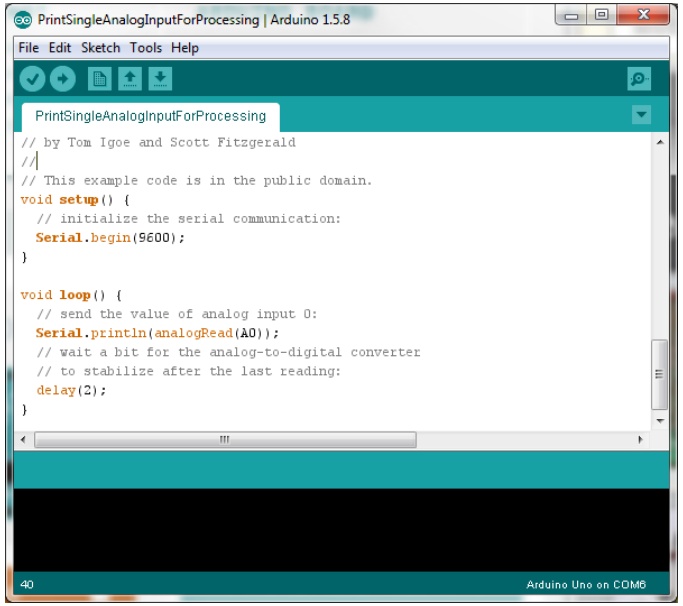
Arduino is an **open-source electronics platform** based on easy-to-use hardware and software. It's intended to enable designers, artists, hobbyists, engineers, and hardware novices to (rapidly) create new interactive experiences.

Arduino Hardware



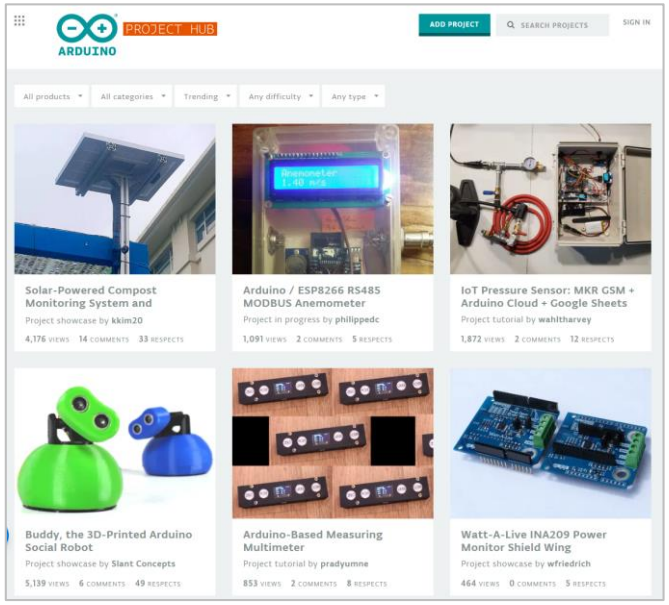
The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. & activate lights, motors…

Arduino Software



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

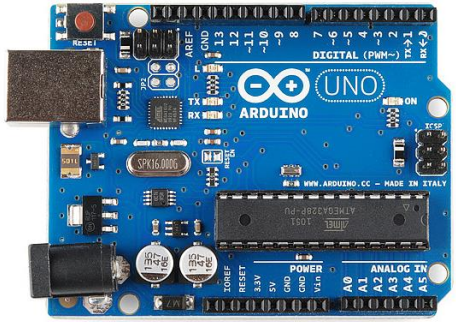
Arduino Community



The Arduino community is the **most popular open hardware movement** online with tutorials, project hubs, forums, etc.

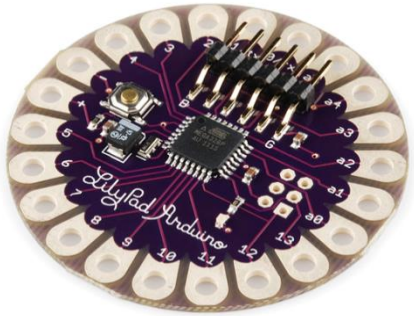
ARDUINO

POPULAR ARDUINO MODELS



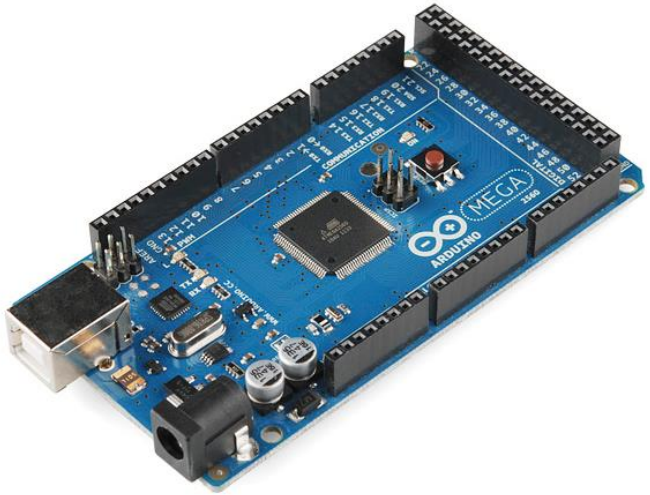
Arduino Uno (R3)

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



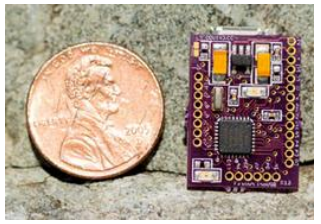
LilyPad Arduino

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



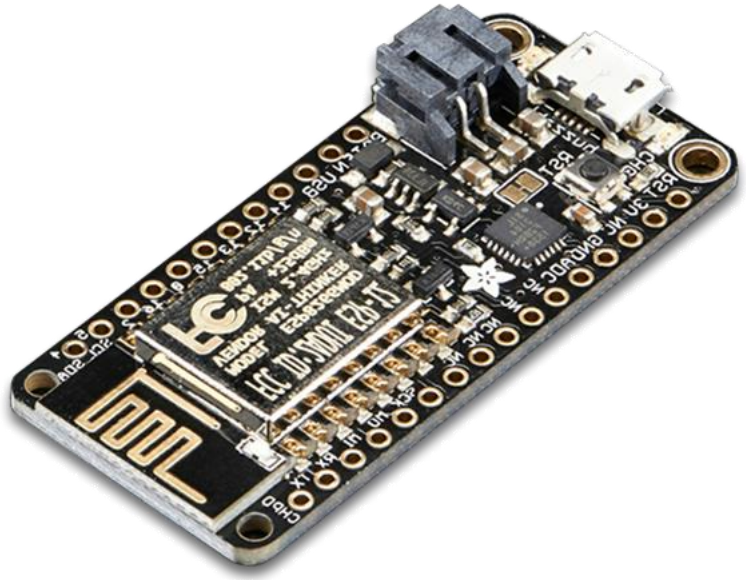
FemtoduinoUSB

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

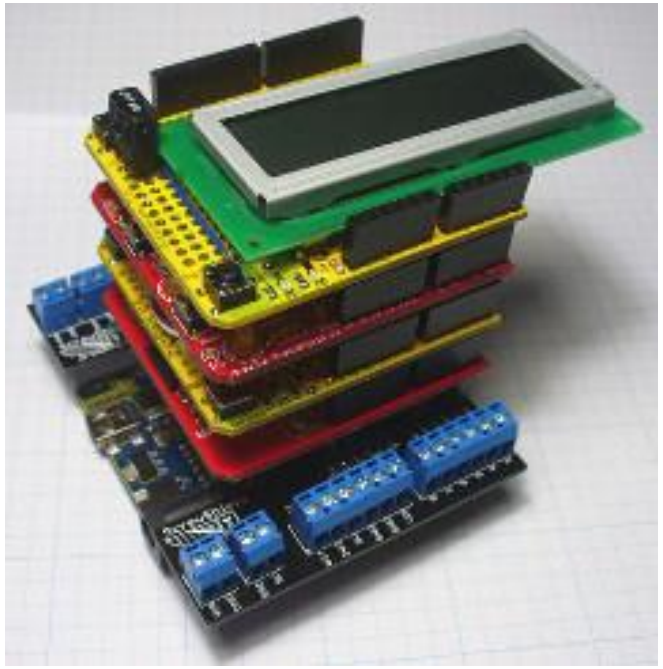
https://www.adafruit.com/feather

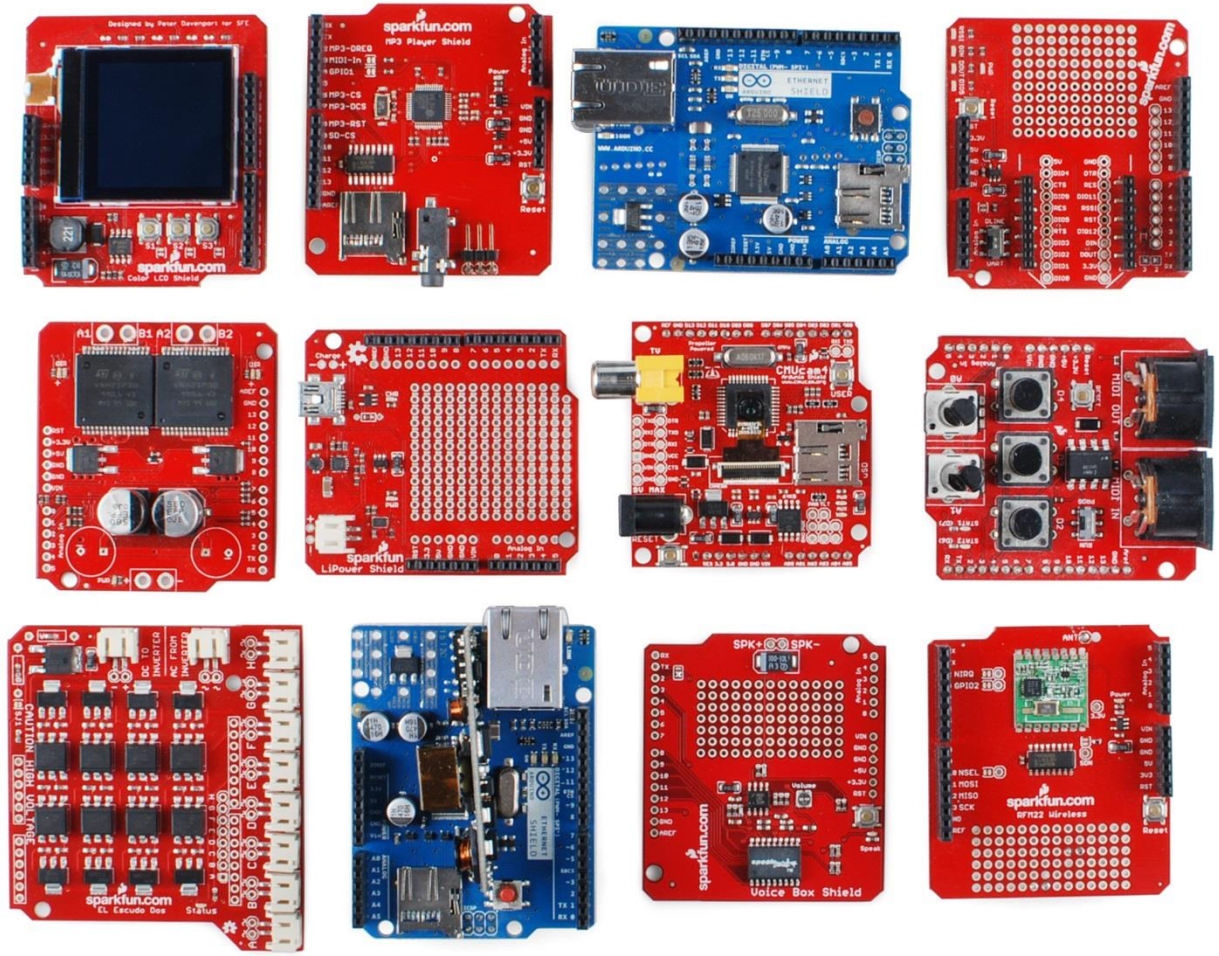
PARTICLE PHOTON

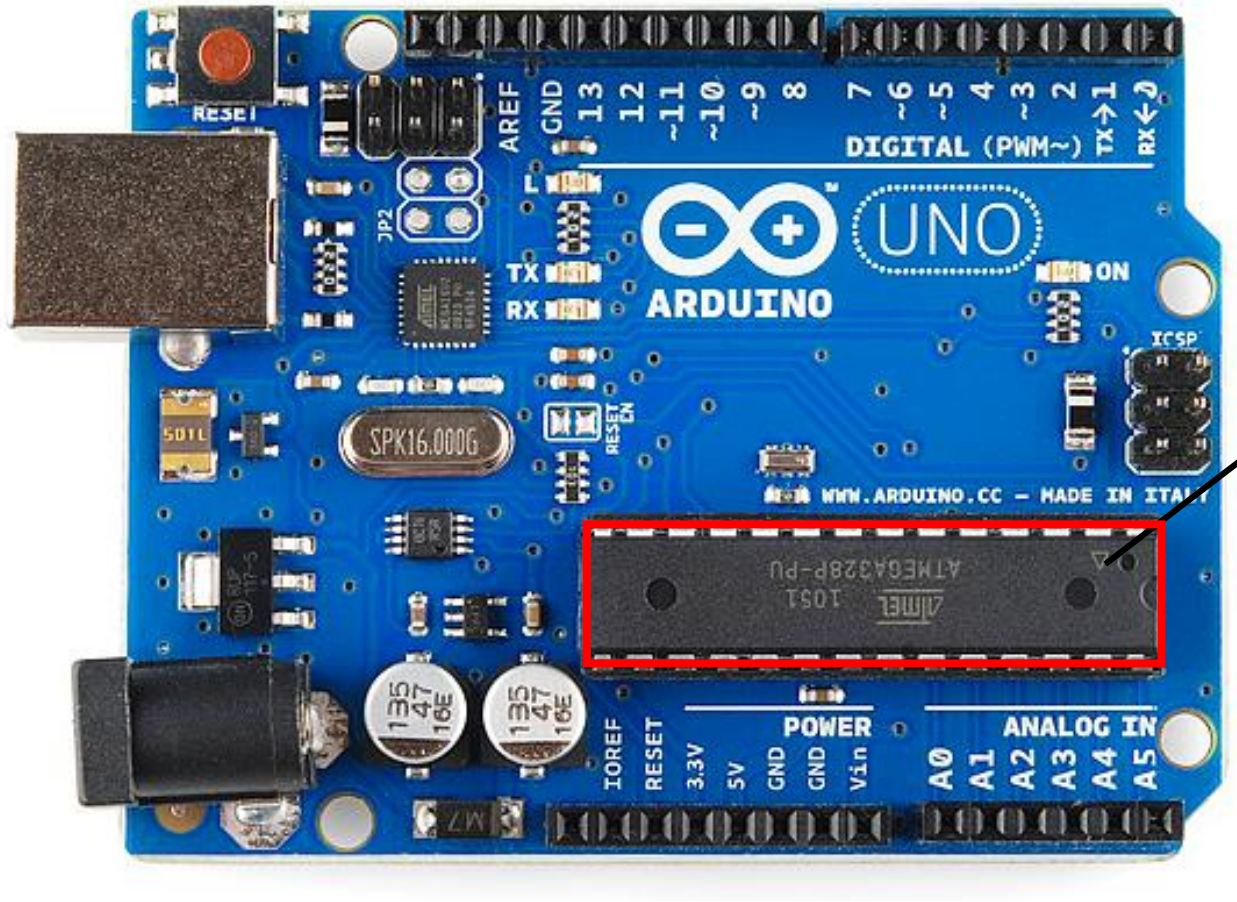
https://www.particle.io/what-is-particle

ARDUINO 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.



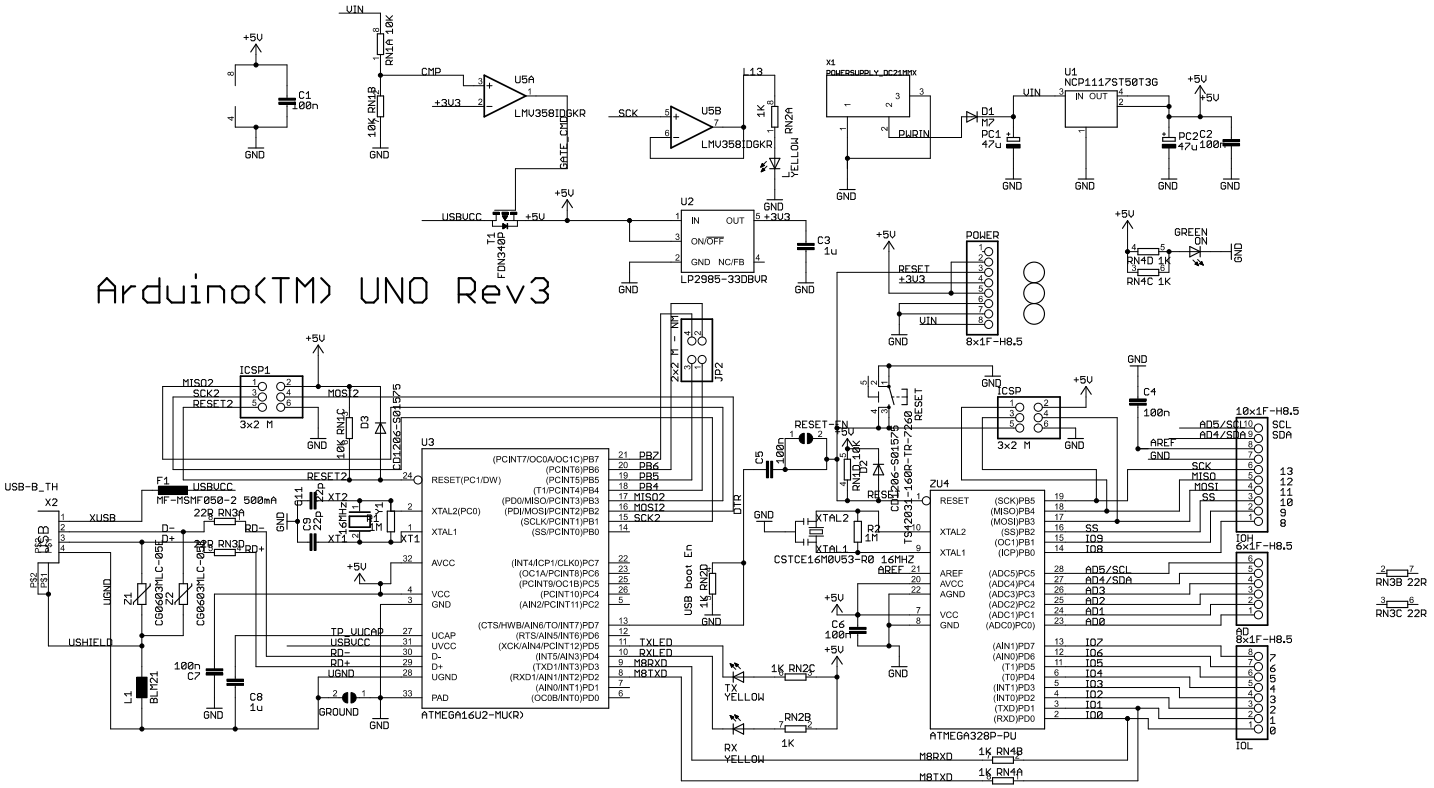




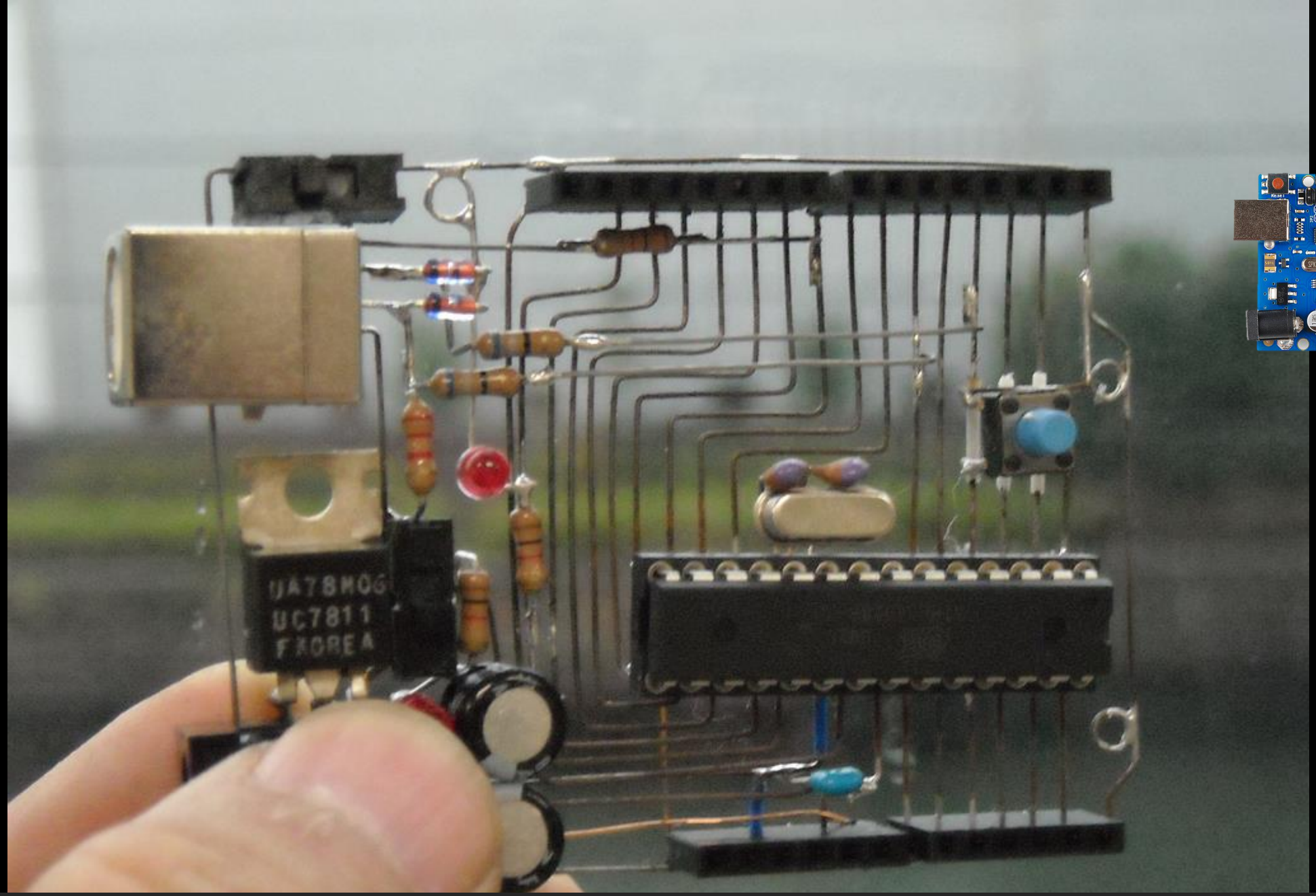
Arduino just makes it easier to interface with **this microcontroller chip**—in this case, the ATMega328—by providing a USB-based programming connection, protection circuitry, oscillator for micro, *etc.*

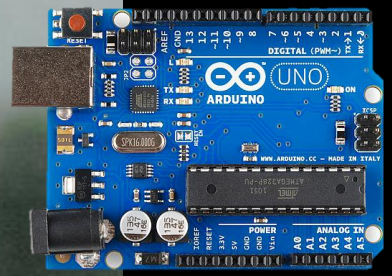
And you could, of course, simply **build your own Arduino**… and many do!

open source hardware

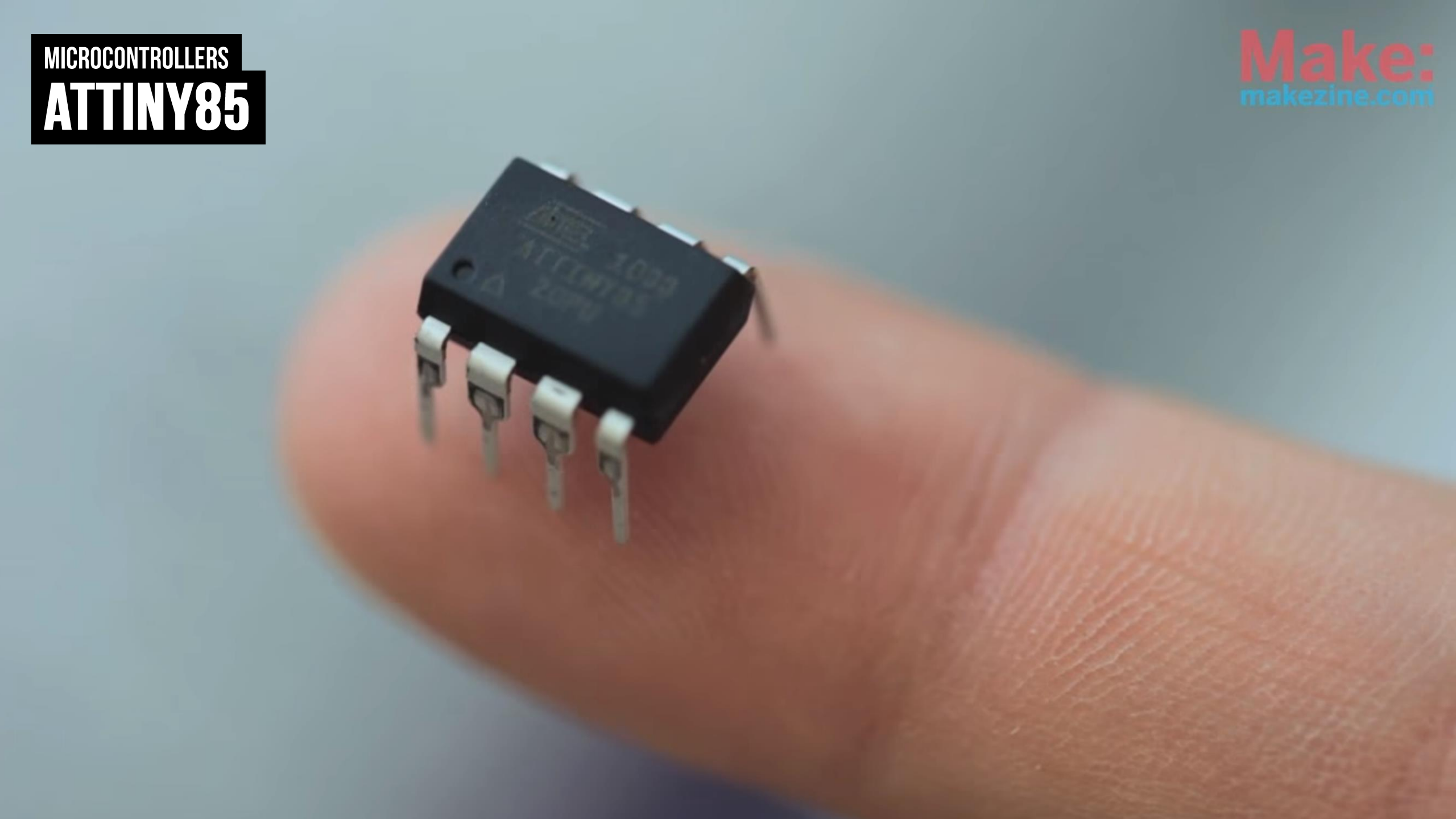


Source: https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf





Designer: Kimio Kosaka, “Arduino Skeleton”, https://make.kosakalab.com/arduino/obaka/project-7/index_en.html


MICROCONTROLLERS
ATTINY85
Make:
makezine.com

Order in the next 02:24:55 to ship **today**. Learn More

ATTINY85-20PU

More Images

Images are for reference only
See Product Specifications

Share

Mouser #:	556-ATTINY85-20PU
Mfr. #:	ATTINY85-20PU
Mfr.:	Microchip Technology / Atmel
Customer #:	Customer #
Description:	8-bit Microcontrollers - MCU 8kB Flash 0.512kB EEPROM 6 I/O Pins
Datasheet:	ATTINY85-20PU Datasheet
ECAD Model:	PCB Symbol, Footprint & 3D Model

Download the free Library Loader to convert this file for your ECAD Tool. Learn More.

Compare Product Add To Project | Add Notes

In Stock: 3,662

Stock:	3,662 Can Ship Immediately
On Order:	4550 View Delivery Dates
Factory Lead-Time:	7 Weeks
Enter Quantity:	Minimum: 1 Multiples: 1 Buy

Pricing (USD)

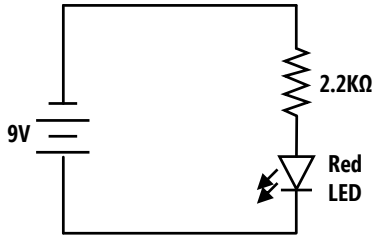
Qty.	Unit Price	Ext. Price
1	$1.23	$1.23
10	$1.21	$12.10
25	$1.13	$28.25
100	$1.03	$103.00
20,000	Quote	

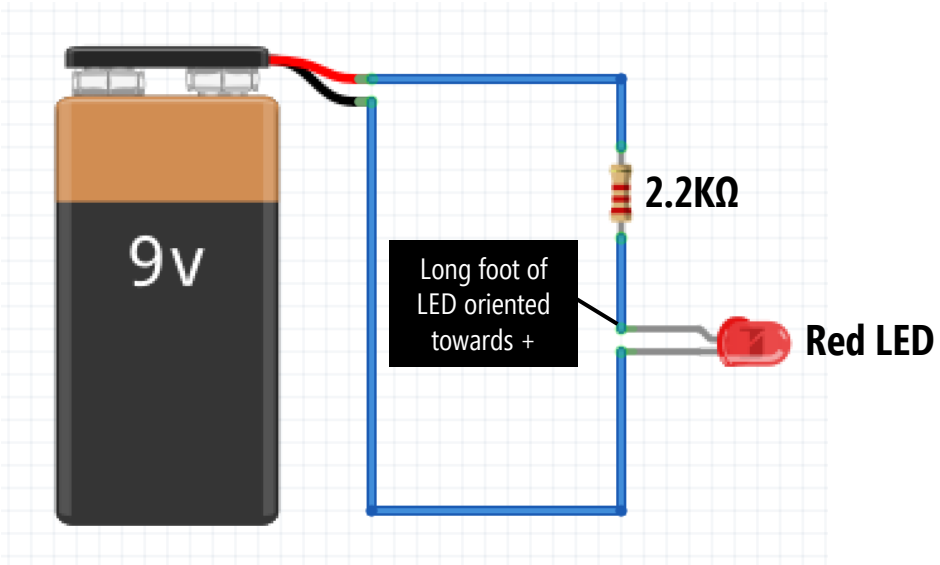
Specifications

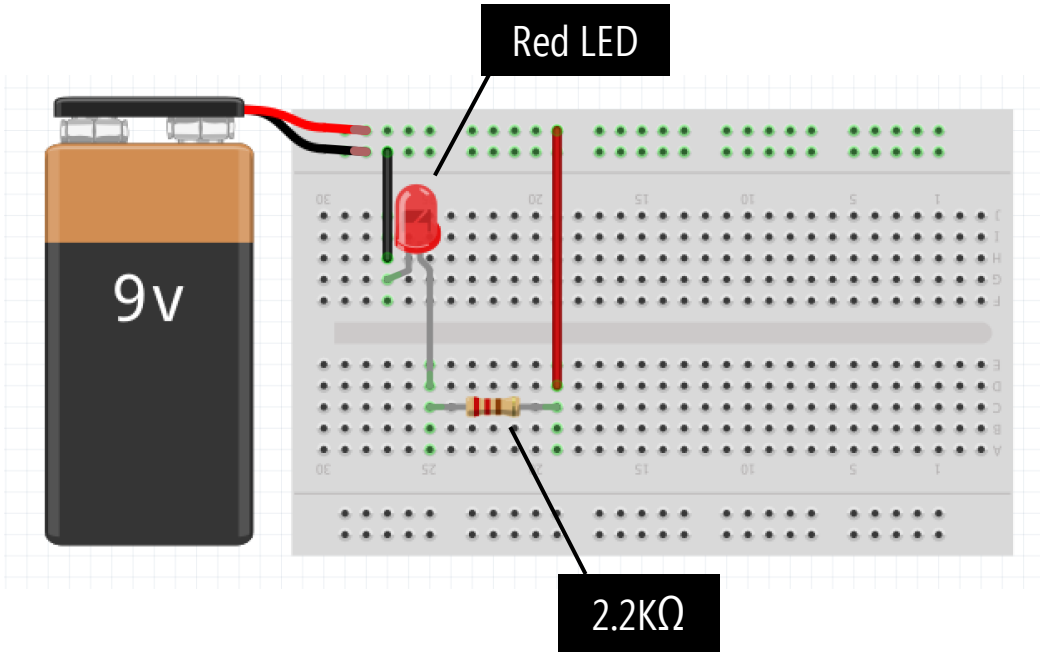
Product Attribute	Attribute Value	Search Similar
Manufacturer:	Microchip	☐
Product Category:	8-bit Microcontrollers - MCU	☑
RoHS:	RoHS Details	
Mounting Style:	Through Hole	☐
Package / Case:	PDIP-8	☐
Series:	ATtiny85	☐

ACTIVITY

LAST TIME: WE BUILT A BASIC LED CIRCUIT





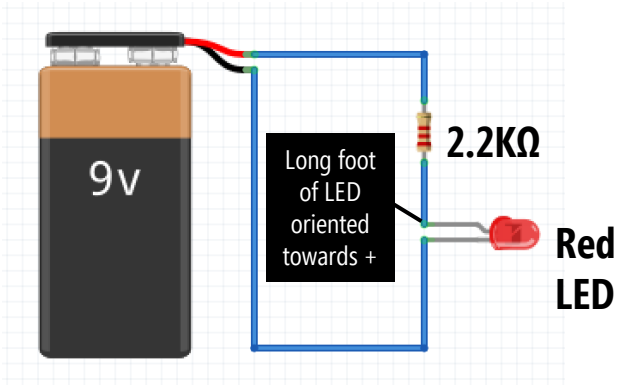


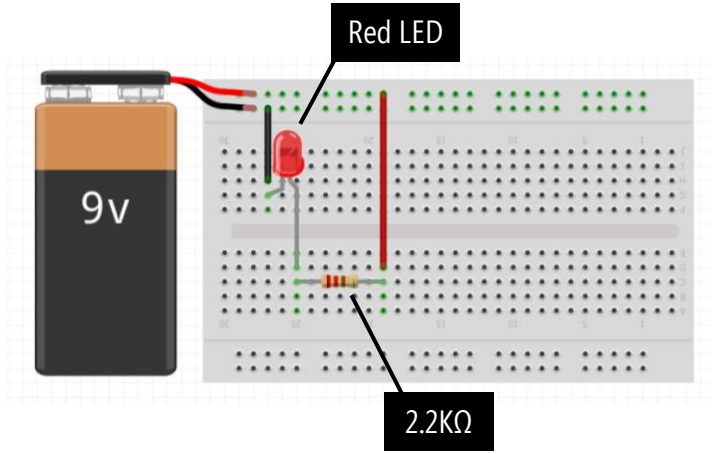
First, we built a free-form LED circuit

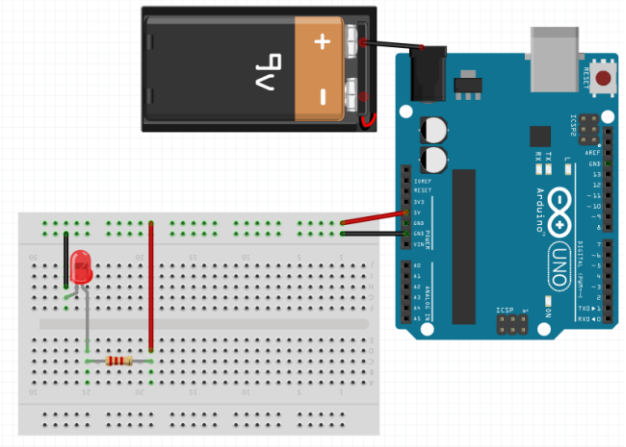
Second, we adapted this circuit to use a breadboard

ACTIVITY

POWER CIRCUIT VIA ARDUINO 5V



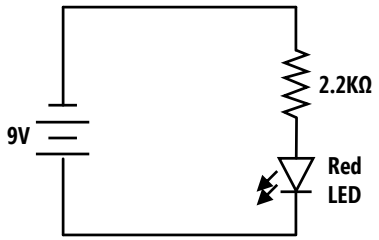


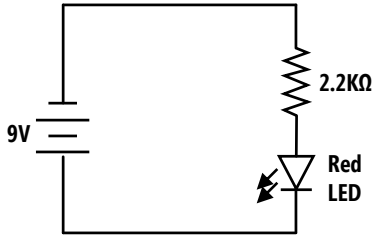


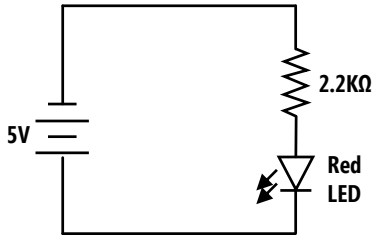
First, we built a free-form LED circuit

Second, we adapted this circuit to use a breadboard

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

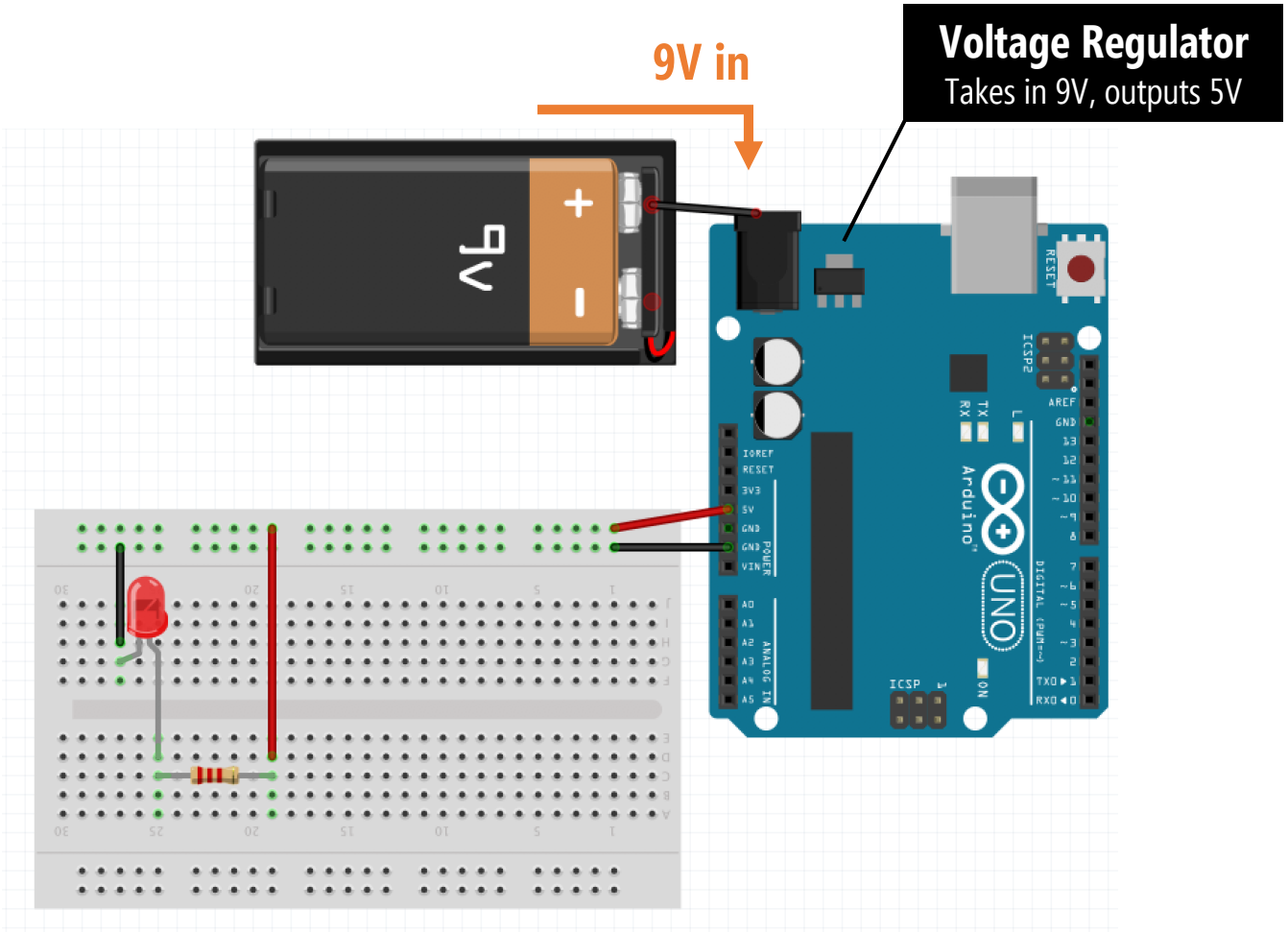


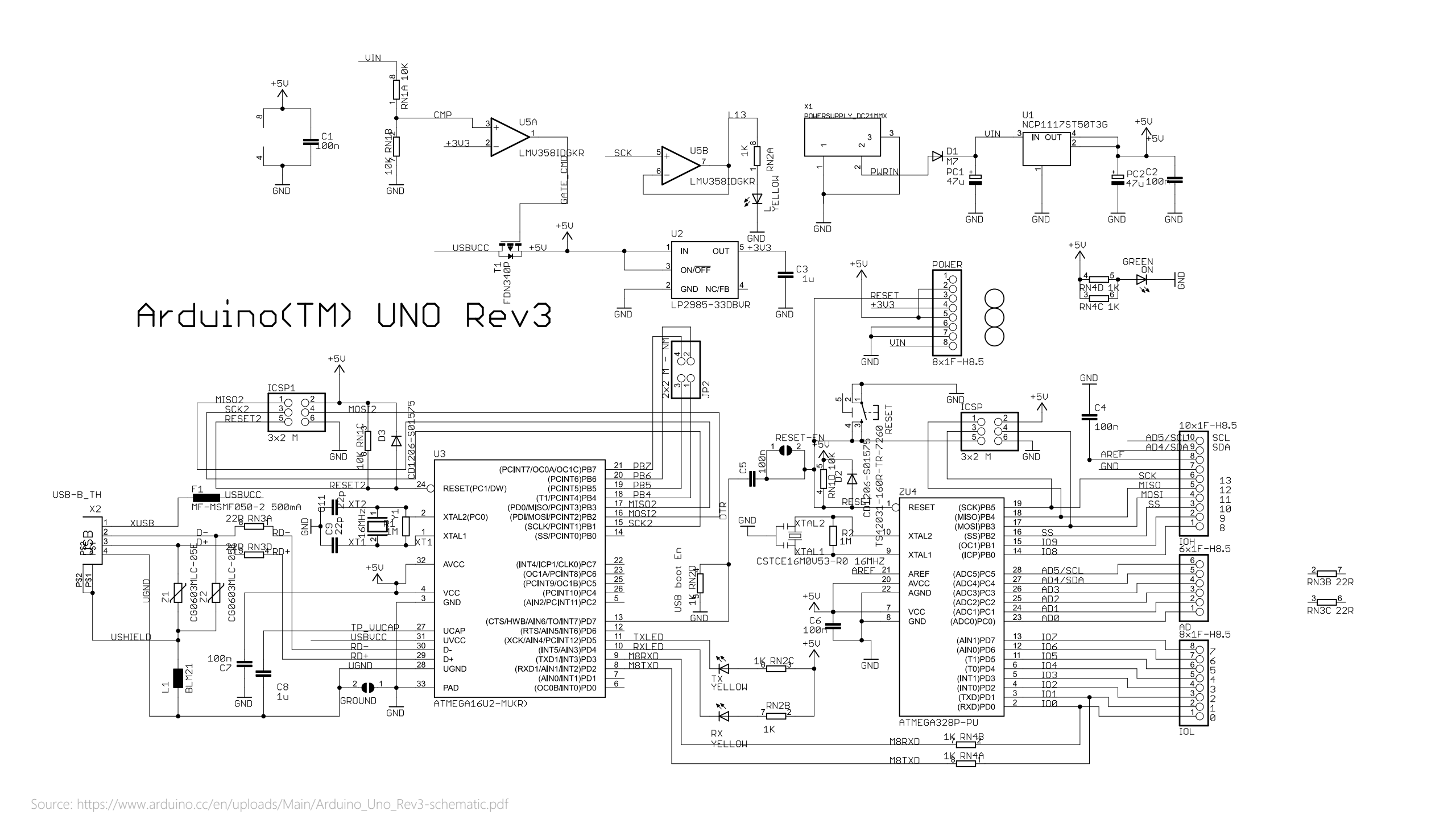




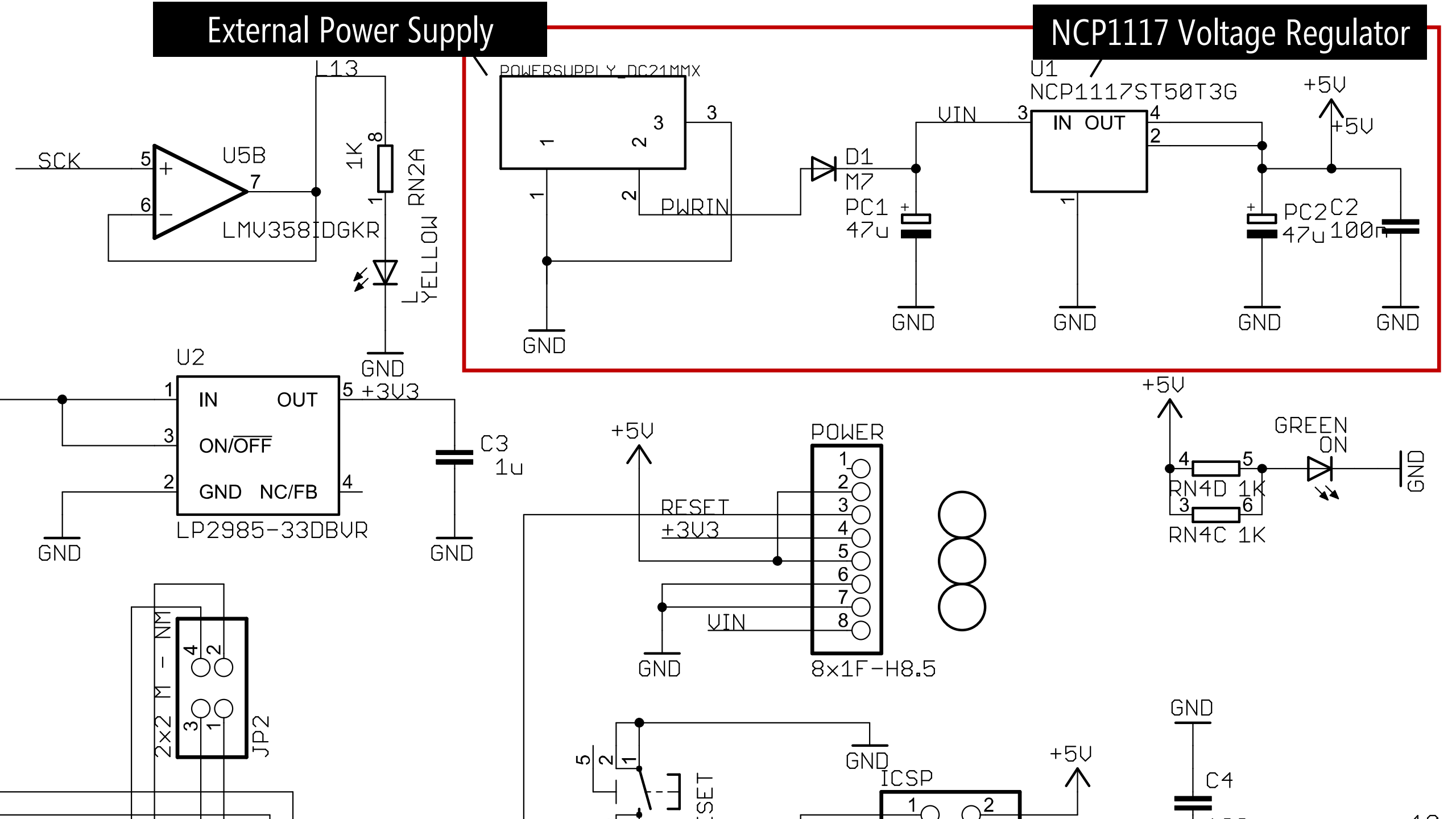
ACTIVITY

POWER CIRCUIT VIA ARDUINO 5V





Source: https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf


External Power Supply
NCP1117 Voltage Regulator
L13
SCK
U5B
LMV358IDGKR
1K
RN2A
L YELLOW
GND
POWERSUPPLY_DC21MMX
PWRIN
D1
M7
PC1
47u
U1
NCP1117ST50T3G
VIN
IN OUT
+5V
PC2
47u
C2
100n
U2
IN
OUT
ON/OFF
GND NC/FB
LP2985-33DBVR
+3V3
C3
1u
POWER
RESET
VIN
8x1F-H8.5
GREEN
ON
RN4D 1K
RN4C 1K
2x2 M - NM
JP2
ICSP
C4

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 ±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 ±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®

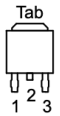
www.onsemi.com

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

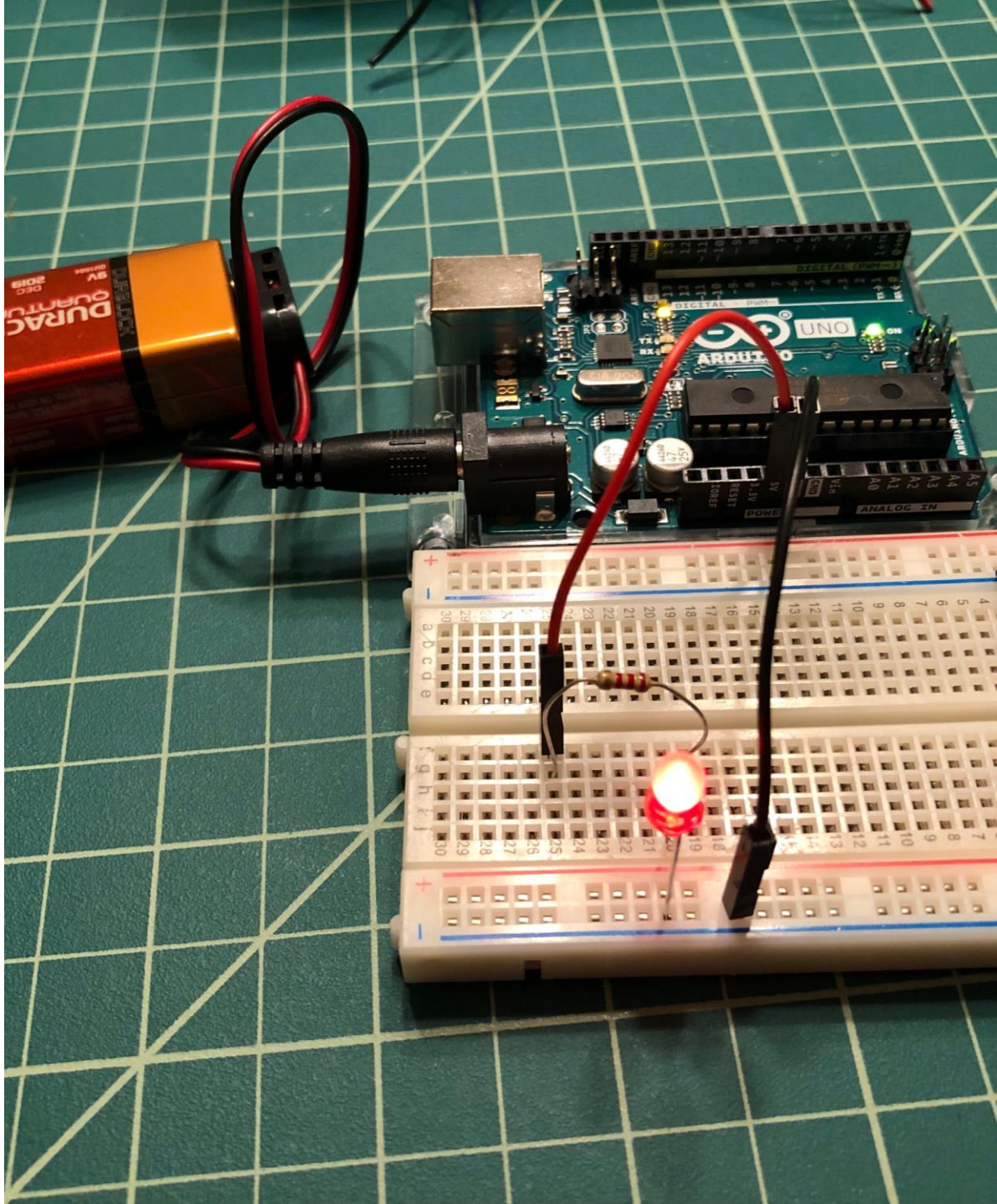
See general marking information in the device marking section on page 14 of this data sheet.

Source: https://www.onsemi.com/pub/Collateral/NCP1117-D.PDF

ACTIVITY: POWER YOUR CIRCUIT VIA ARDUINO FIXED 5V OUTPUT

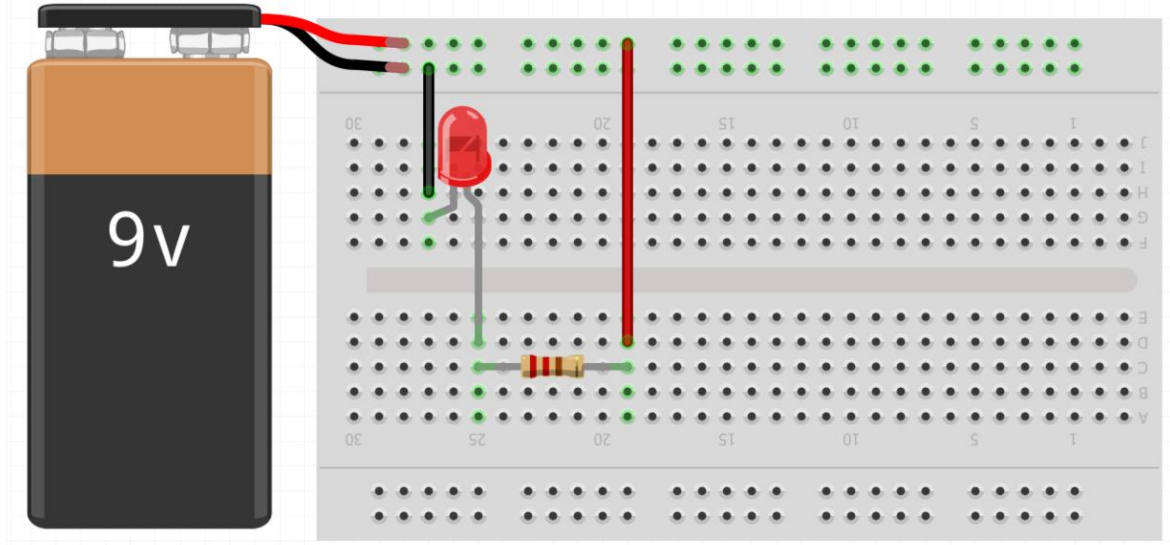
Convert the circuit you previously made with the **9V battery** to, instead, use the **Arduino Uno 5V** and **GND (0V)** connections.

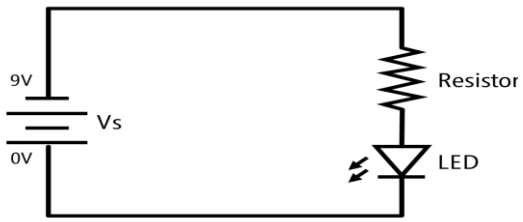
This is a bit of a **toy exercise** given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! ☺

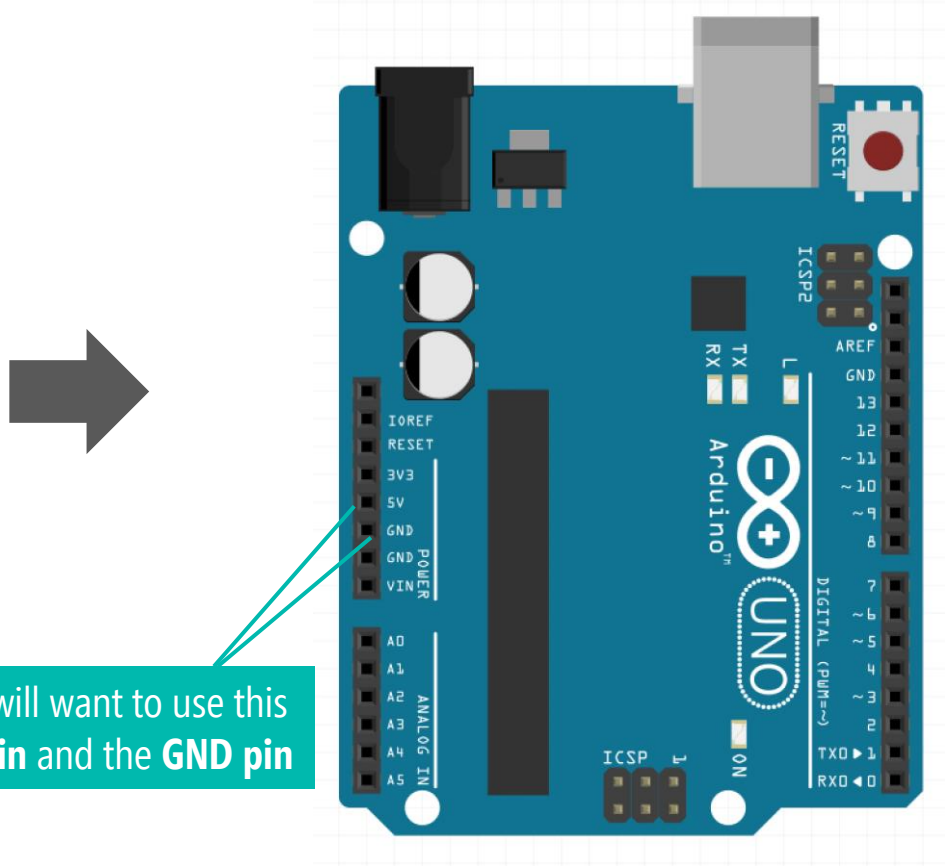


POWER YOUR CIRCUIT VIA ARDUINO UNO 5V OUTPUT

For our first exercise, we are just going to convert the circuit you previously made 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! ☺



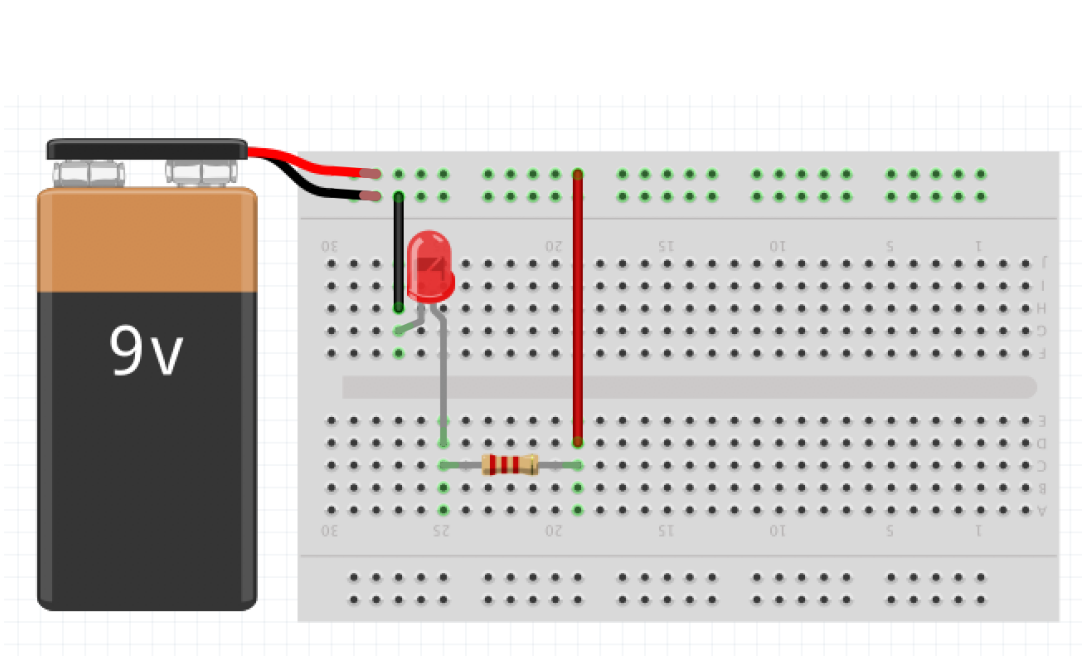


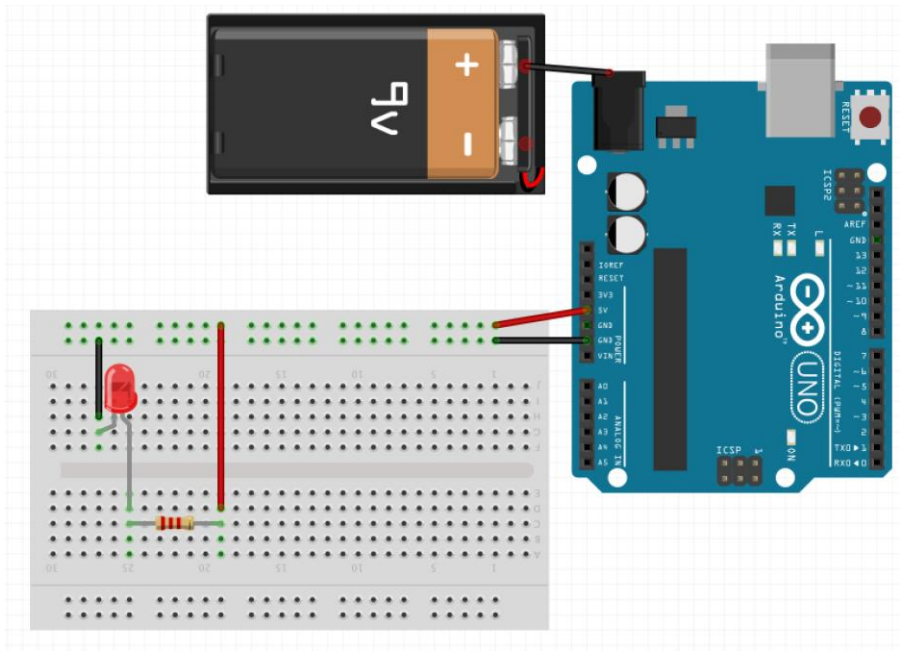


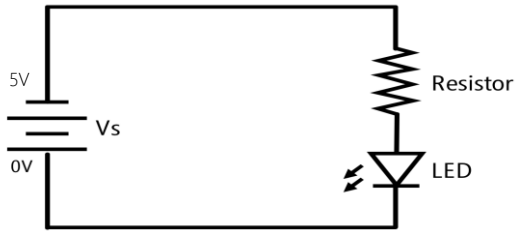
ARDUINO

POWER YOUR CIRCUIT VIA ARDUINO UNO 5V OUTPUT

For our first exercise, we are just going to convert the circuit you previously made 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! ☺





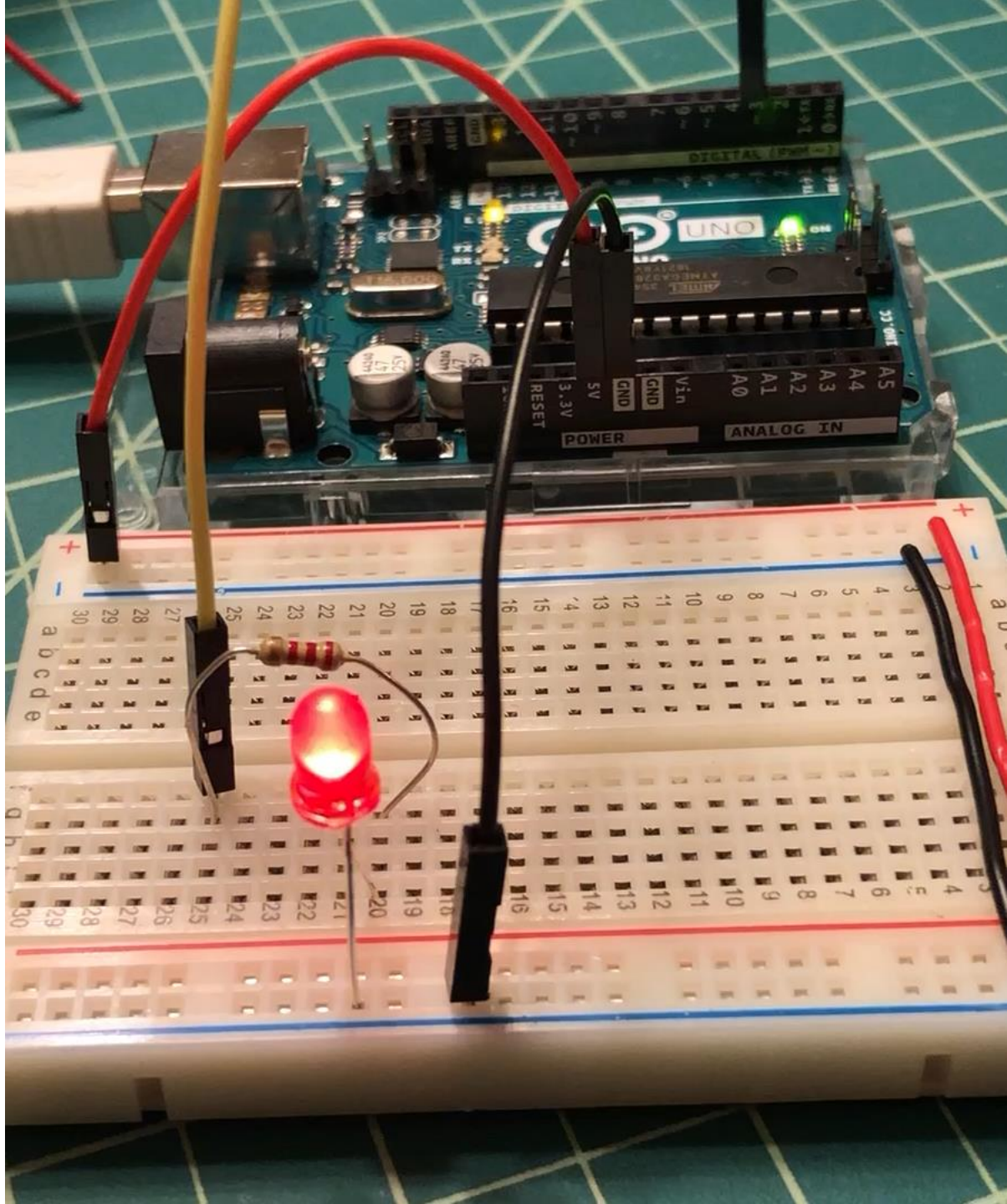


ACTIVITY: CONTROL THE LED WITH YOUR ARDUINO

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

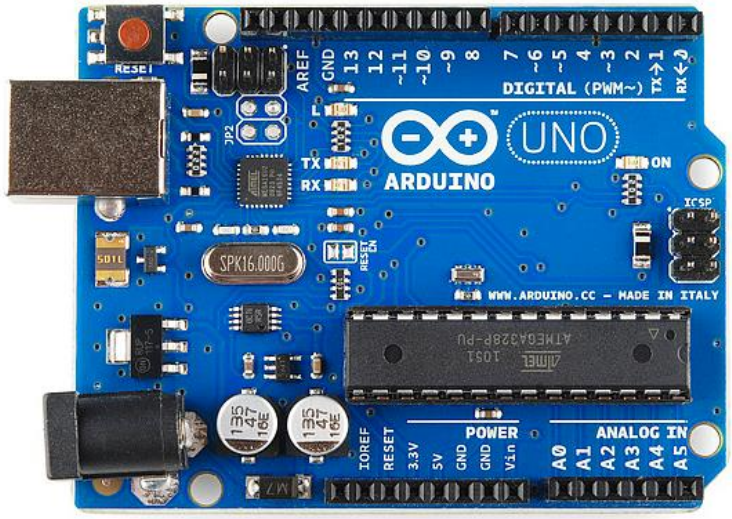
Let's program the **Arduino** to **control the LED**.

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



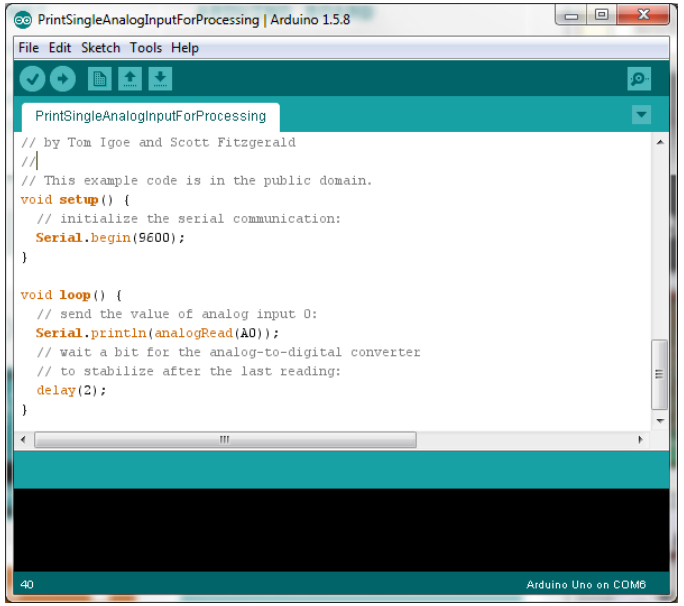
ARDUINO PLATFORM

Arduino Hardware



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

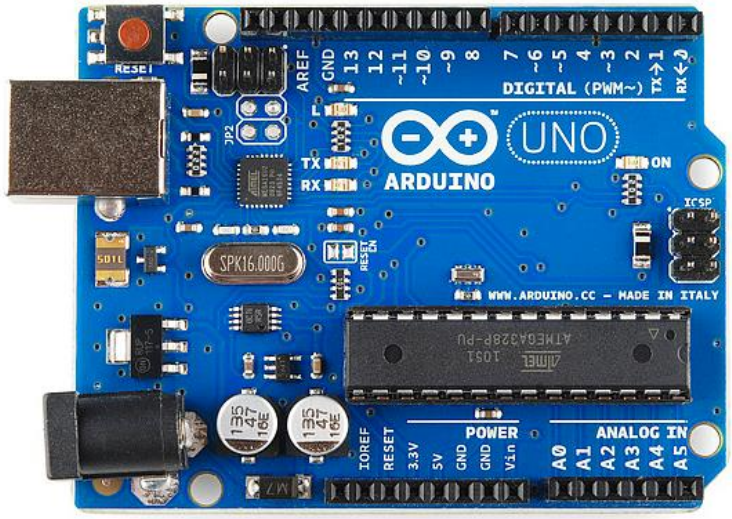
Arduino Software



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

ARDUINO PLATFORM

Arduino Hardware



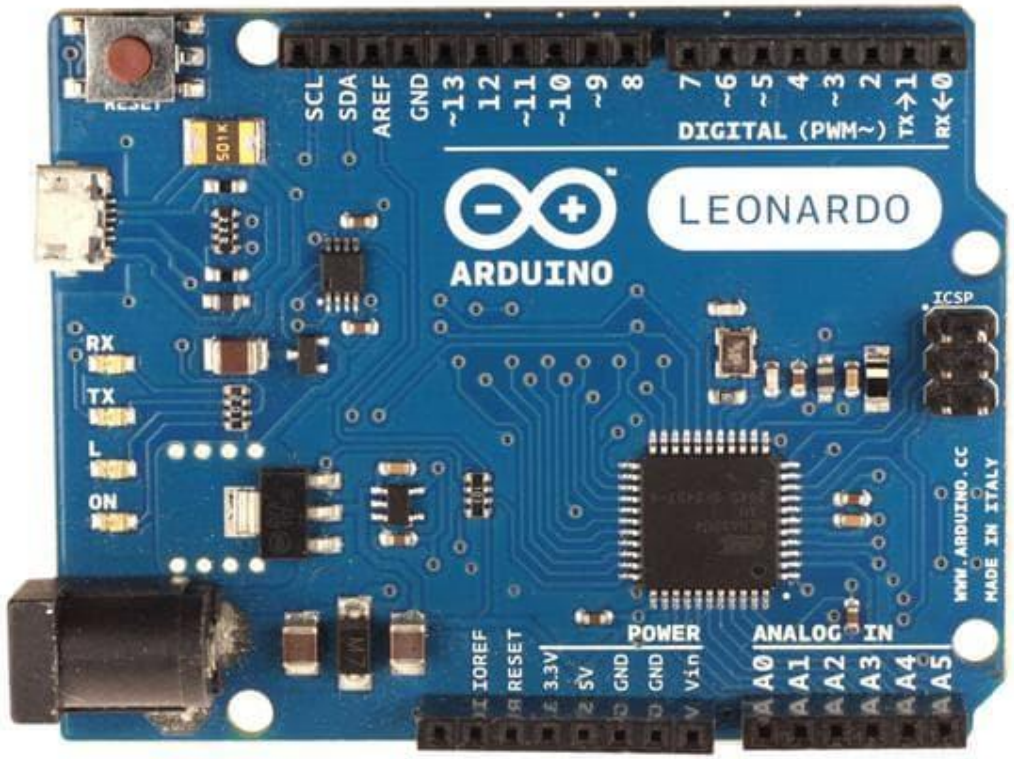
The Arduino board contains a **microcontroller** (small computer) and is used to sense the environment, people, etc. and activate lights, motors, etc.

Arduino Software

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

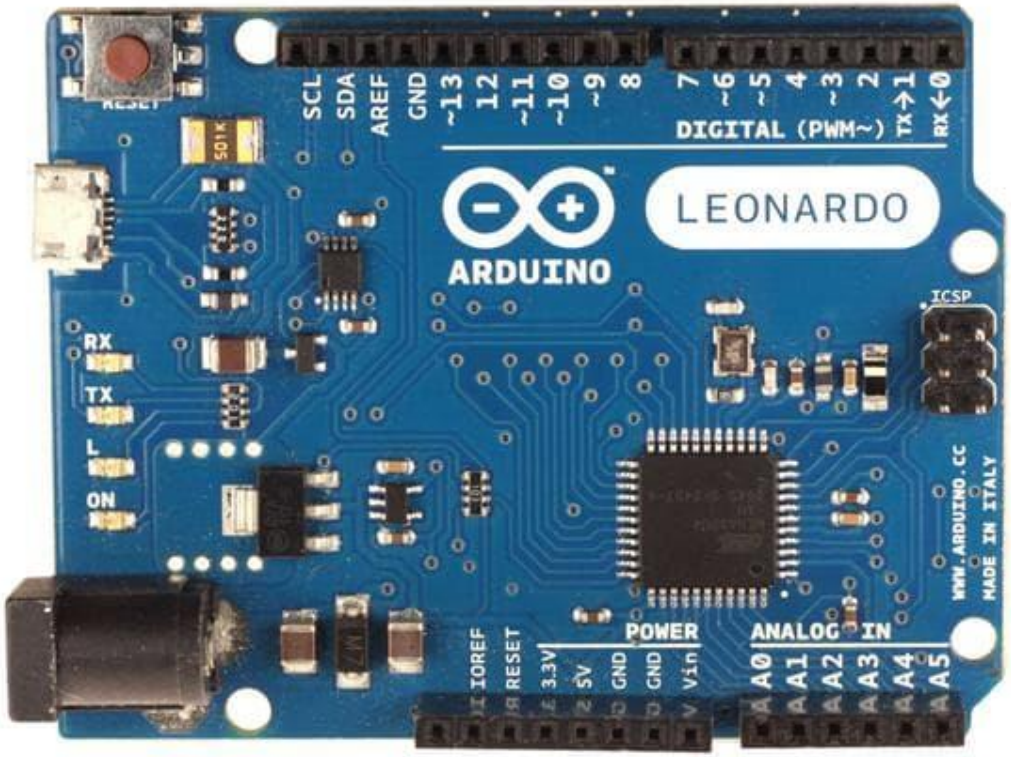
What do you see?



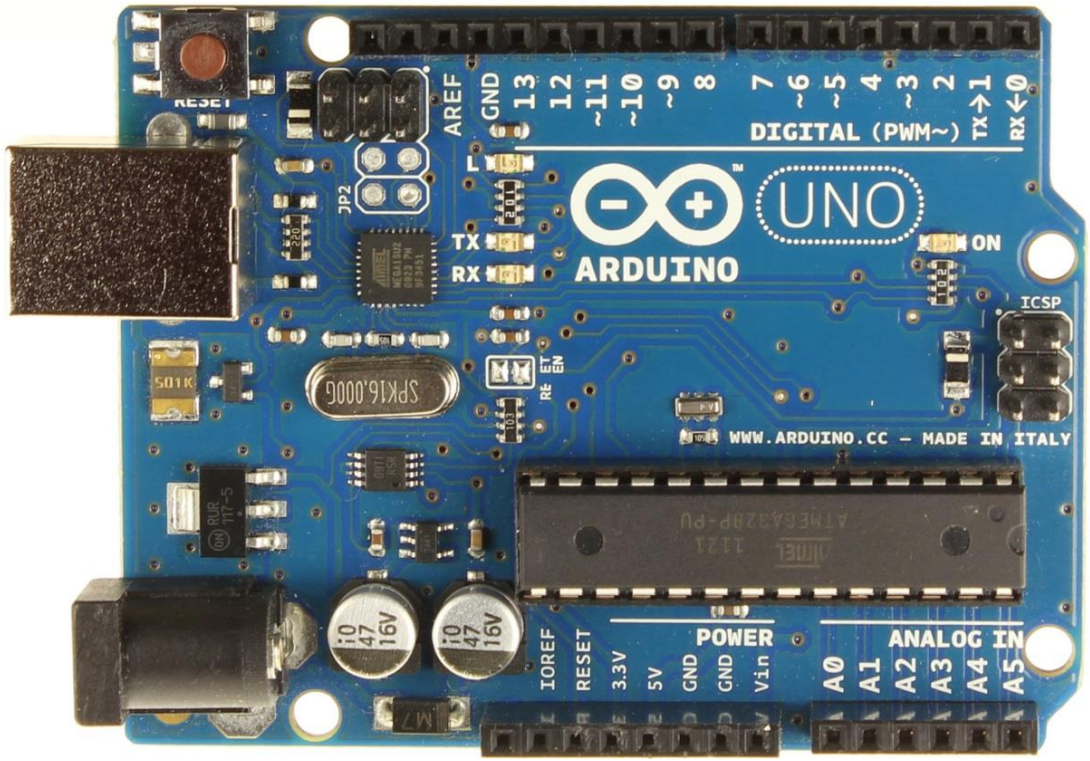
ARDUINO HARDWARE

LEONARDO VS. UNO

Leonardo



Uno



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

ARDUINO HARDWARE

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

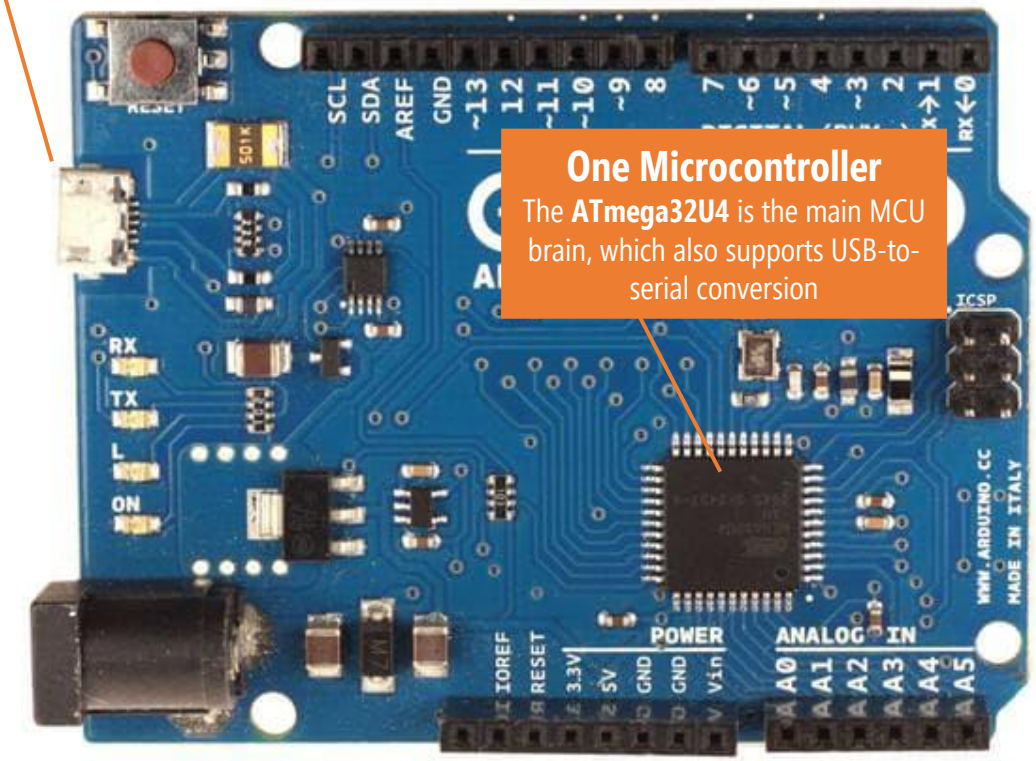
Leonardo

Uno

MicroUSB

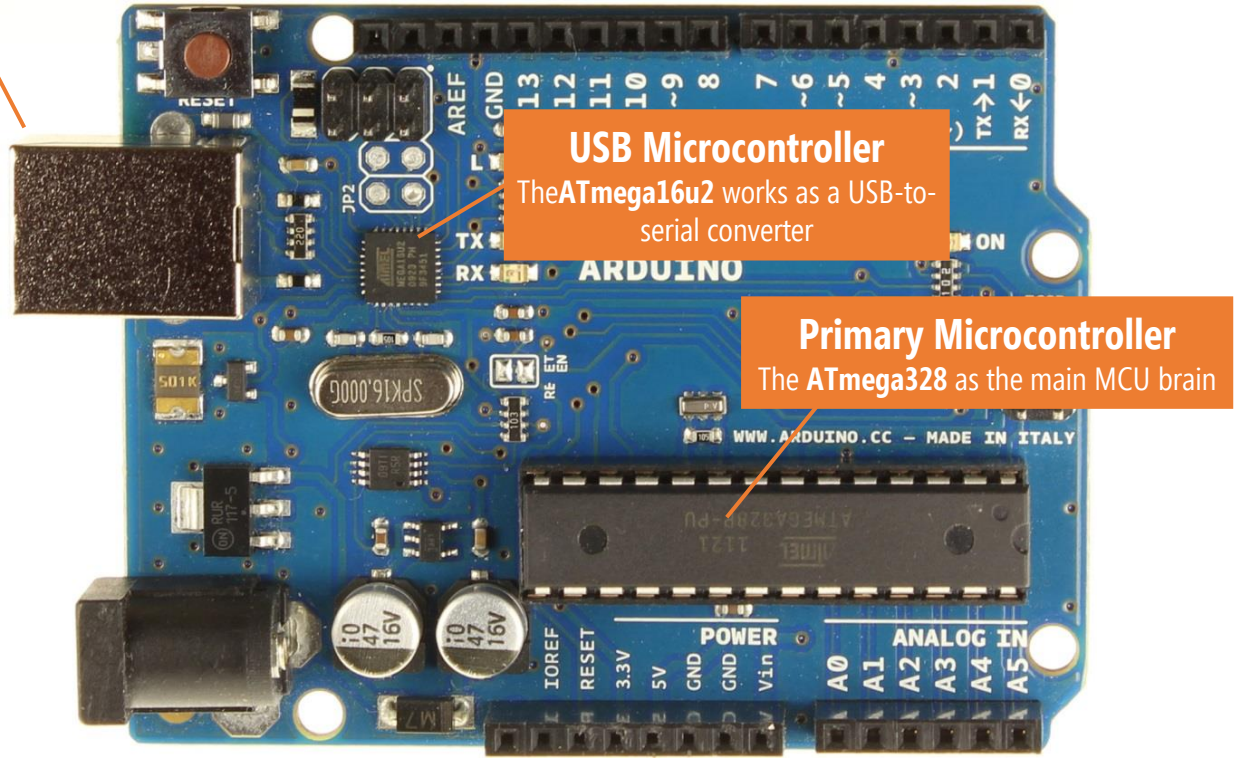
USB A-B
(typically used with printers)

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



USB Microcontroller
The**ATmega16u2** works as a USB-to-serial converter

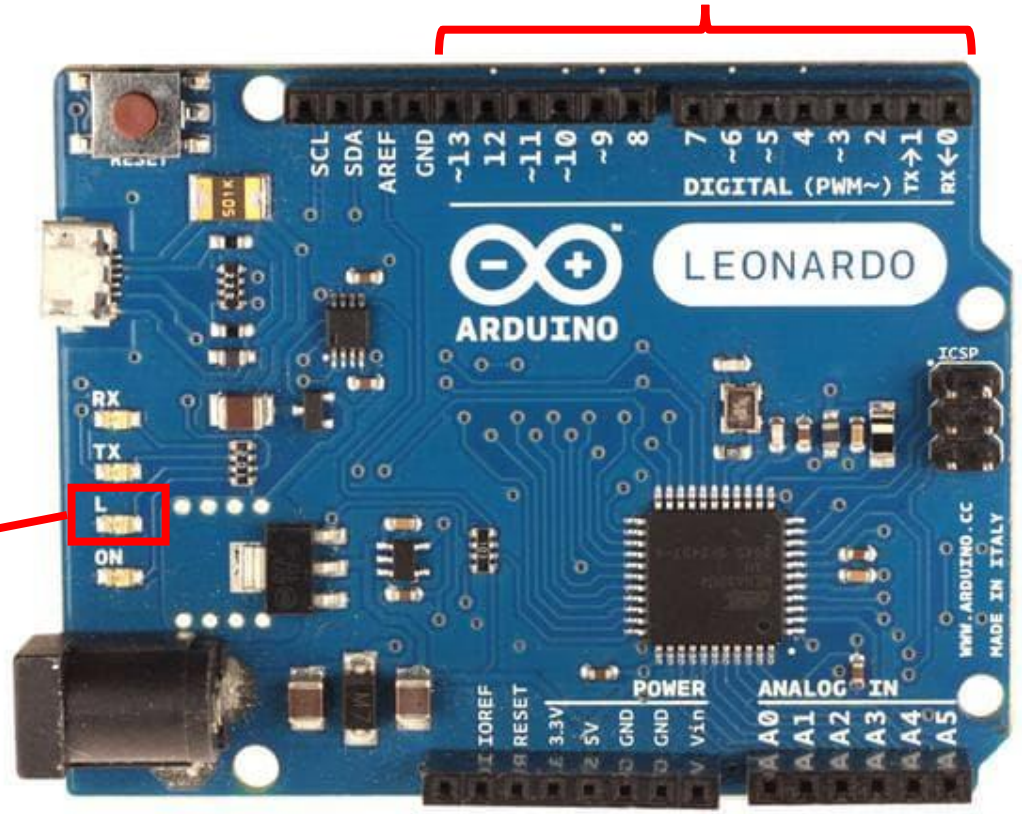
Primary Microcontroller
The **ATmega328** as the main MCU brain



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

ARDUINO LEONARDO DIGITAL I/O

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

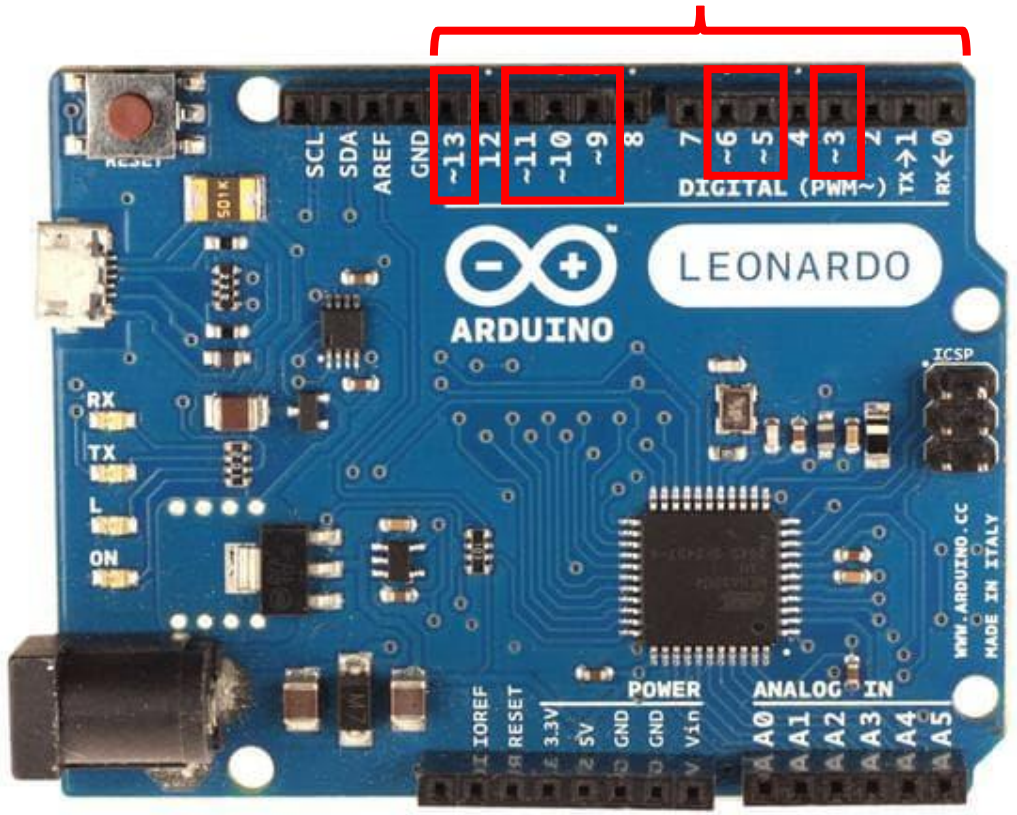


'L' LED: The 'L' LED lights up when pin D13 goes HIGH. This is useful for debugging.

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

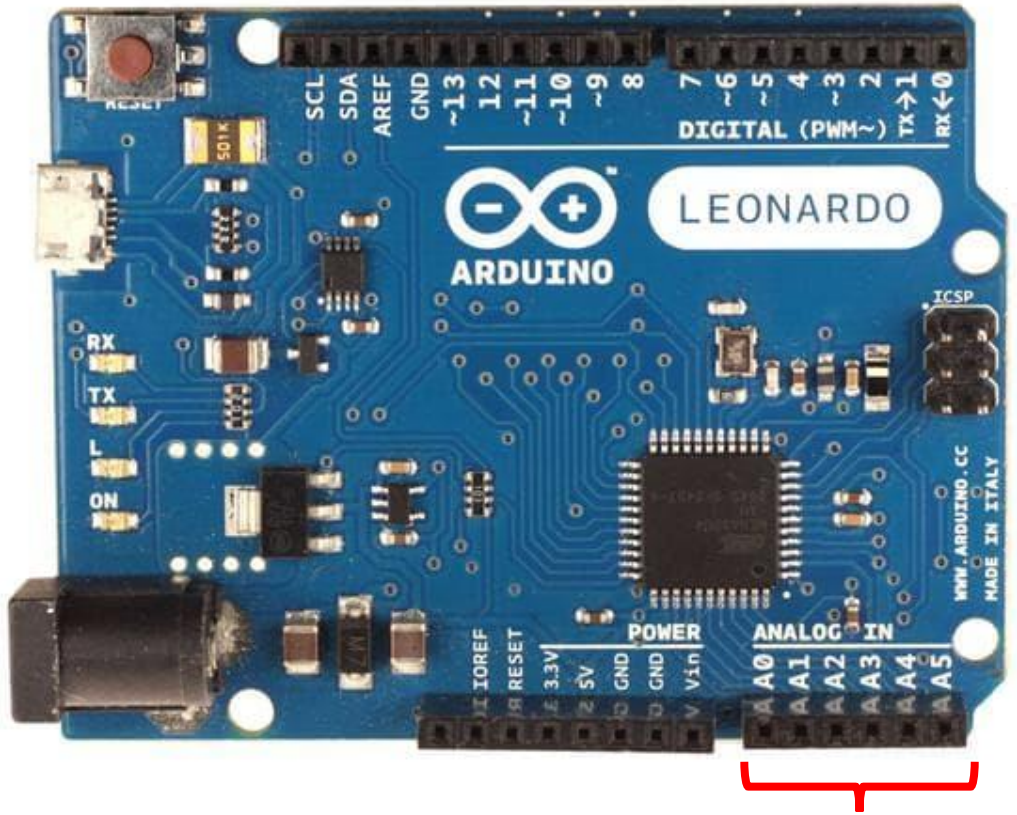
ARDUINO LEONARDO ANALOG OUTPUT

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



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

ARDUINO LEONARDO ANALOG INPUT



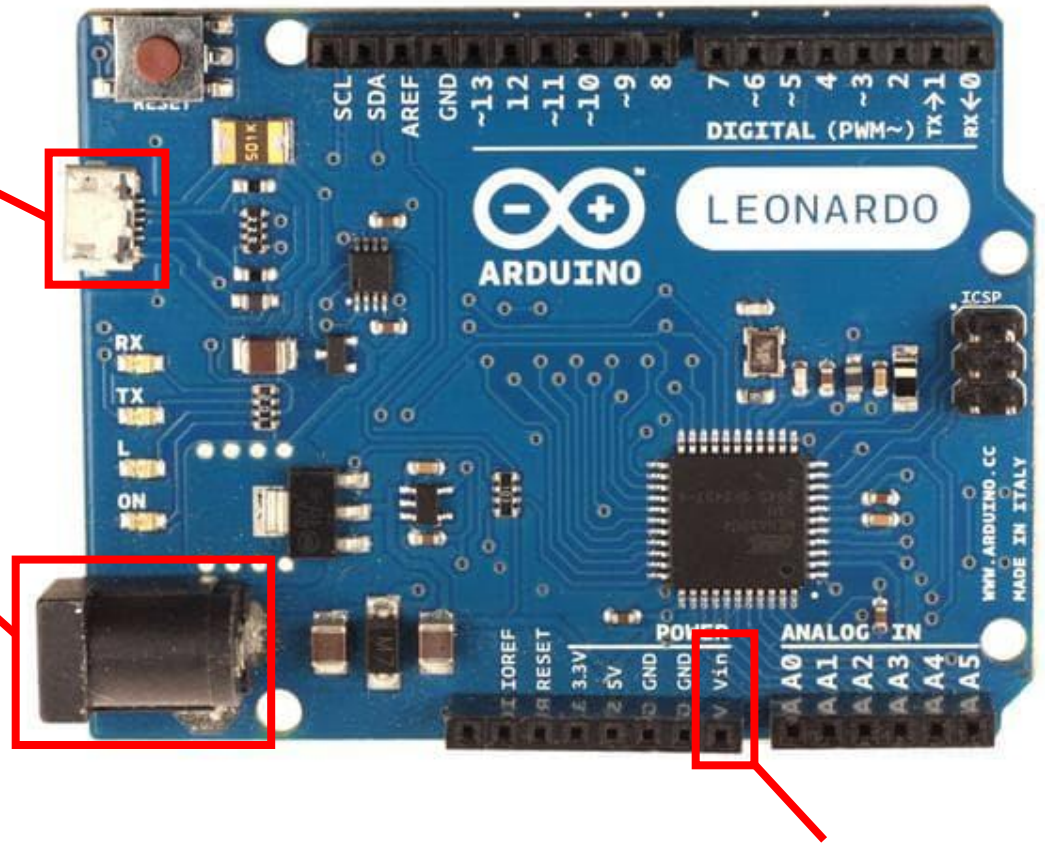
6 analog inputs: 10 bit resolution (from 0 to 1023). By default, they measure from GND to 5V but this can be changed using the AREF pin and the analogReference function

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

ARDUINO LEONARDO: 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 power supply to GND, you will damage your Arduino

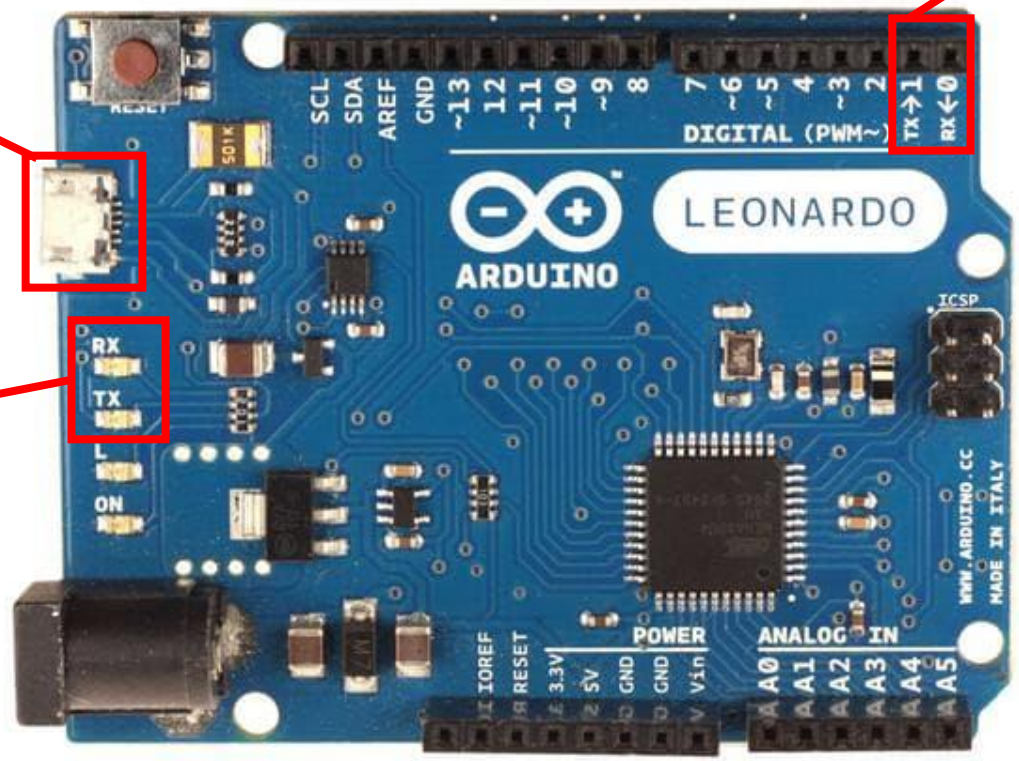
Source: https://www.arduino.cc/en/Guide/ArduinoLeonardoMicro; https://store.arduino.cc/usa/Leonardo; https://learn.sparkfun.com/tutorials/arduino-comparison-guide/atmega32u4-boards

ARDUINO HARDWARE

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)



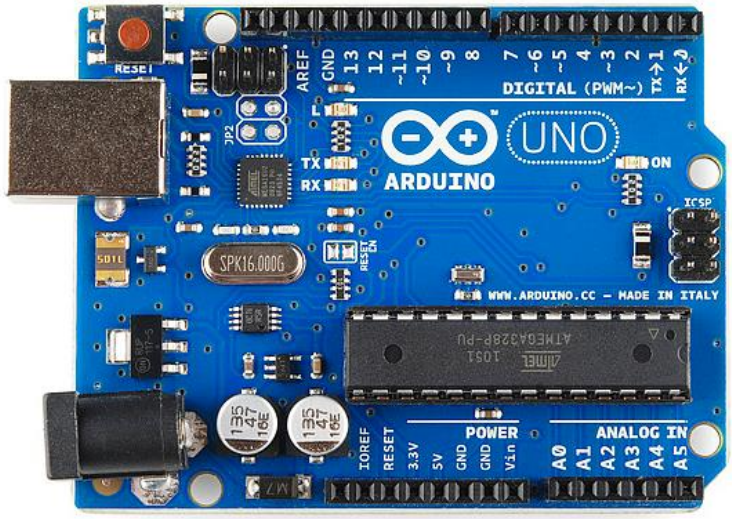
TX/RX Pins: TX means 'transmit' and RX means 'receive.' These pins function like normal digital I/O pins **unless** you use the Serial library in your Arduino program.

So, I often **avoid** using these pins because using Serial I/O is common for debugging.

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

ARDUINO 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

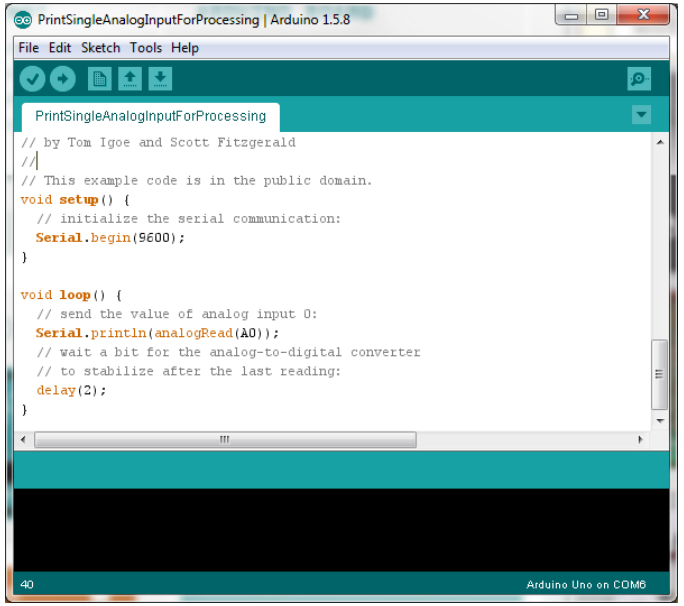
The Arduino software provides a **simplified C programming editor**. It is also used to **upload programs** to Arduino compatible boards.

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



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

ARDUINO PROGRAMMING

ARDUINO VARIABLES: DATA TYPES & CONVERSION

Arduino data types and constants

Data Types

String
String()
array
bool
boolean
byte
char
double
float
int
long
short
unsigned char
unsigned int
unsigned long
void
word

Conversion

byte()
char()
float()
int()
long()
word()

Variable Scope & Qualifiers

const
scope
static
volatile

Source: https://www.arduino.cc/reference/en/

ARDUINO PROGRAMMING

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)

Source: https://www.arduino.cc/reference/en/

ARDUINO PROGRAMMING

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

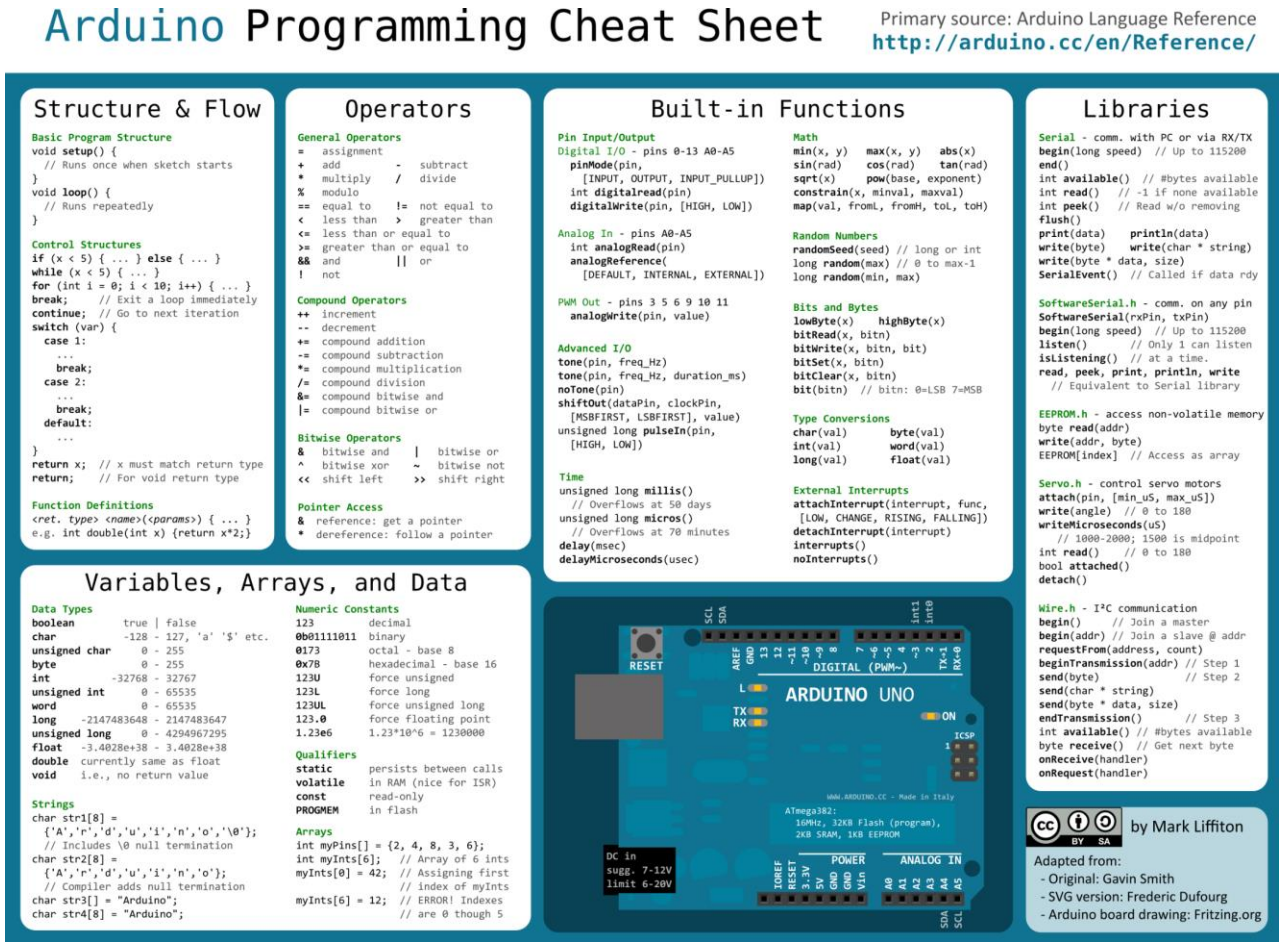
Keyboard

Mouse

Source: https://www.arduino.cc/reference/en/

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    >   greater than
<=  less than or equal to
>=  greater than or equal to
&&  and          ||  or
!   not
```

Compound Operators
```
++  increment
--  decrement
+=  compound addition
-=  compound subtraction
*=  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
```

Built-in Functions

Pin Input/Output
```
Digital I/O - pins 0-13 A0-A5
  pinMode(pin,
    [INPUT, OUTPUT, INPUT_PULLUP])
  int 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)
```

Math
```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers
```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions
```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts
```
attachInterrupt(interrupt, func,
  [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

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 rdy

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 - I²C 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)
```

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
```

Strings
```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants
```
123          decimal
0b01111011   binary
0173         octal - base 8
0x78         hexadecimal - base 16
123U         force unsigned
123L         force long
123UL        force unsigned long
123.0        force floating point
1.23e6       1.23*10^6 = 1230000
```

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

Arrays
```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];   // Array of 6 ints
myInts[0] = 42;  // Assigning first
                 // index of myInts
myInts[6] = 12;  // ERROR! Indexes
                 // are 0 though 5
```


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!

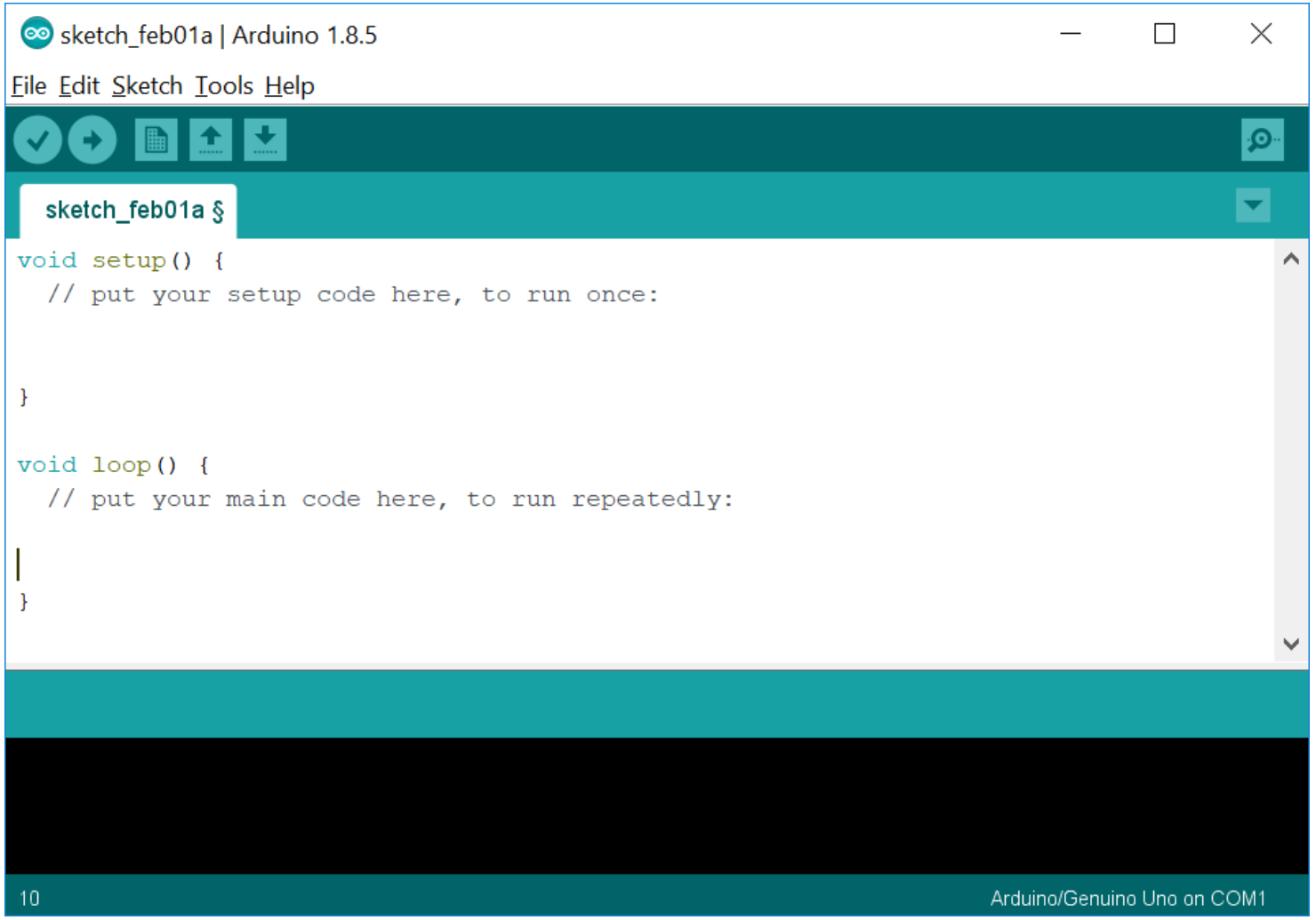
All Arduino programs are comprised of two areas: setup and loop.

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

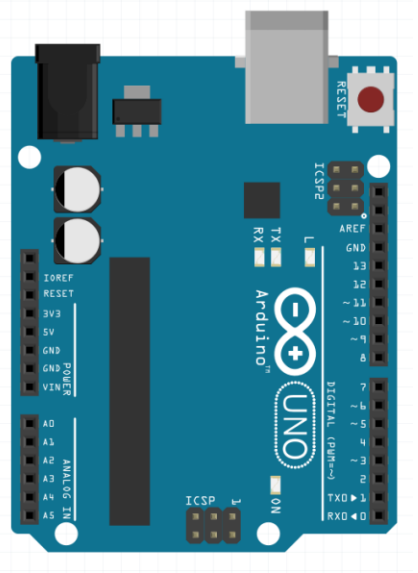


ARDUINO PROGRAMMING

UPLOAD PROGRAM TO ARDUINO

To test your program, you compile and upload to Arduino

USB Cable

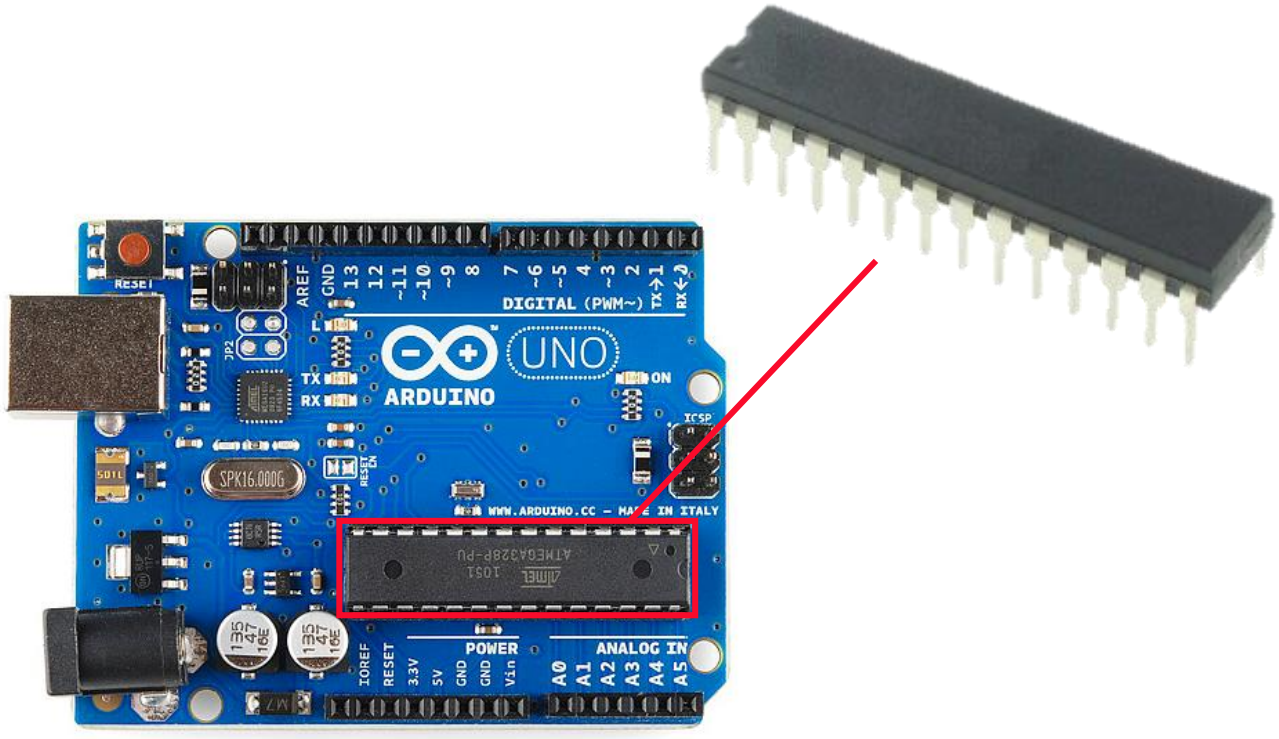


Once the program is loaded onto the Arduino, you can unplug it and run on battery. The program also persists through unpowered states.

ARDUINO ARCHITECTURE

ARDUINO UNO MEMORY

ATmega328



Specifications

Product Attribute	Attribute Value	Search Similar
Manufacturer:	Microchip	☐
Product Category:	8-bit Microcontrollers - MCU	☑
RoHS:	RoHS Details	
Mounting Style:	Through Hole	☐
Package / Case:	PDIP-28	☐
Series:	ATmega328P	☐
Core:	AVR	☐
Data Bus Width:	8 bit	☐
Maximum Clock Frequency:	20 MHz	☐
Program Memory Size:	32 kB	☐
Data RAM Size:	2 kB	☐
ADC Resolution:	10 bit	☐
Number of I/Os:	23 I/O	☐
Operating Supply Voltage:	1.8 V to 5.5 V	☐
Minimum Operating Temperature:	- 40 C	☐
Maximum Operating Temperature:	+ 85 C	☐
Packaging:	Tube	☐
Height:	4.57 mm	
Length:	34.79 mm	
Product:	MCU	
Program Memory Type:	Flash	
Width:	7.49 mm	
Brand:	Microchip Technology / Atmel	
Data RAM Type:	SRAM	
Data ROM Size:	1 kB	
Data ROM Type:	EEPROM	
Interface Type:	I2C, SPI, USART	
Number of ADC Channels:	6	
Number of Timers/Counters:	3 Timer	
Processor Series:	megaAVR	
Product Type:	8-bit Microcontrollers - MCU	
Factory Pack Quantity:	14	
Subcategory:	Microcontrollers - MCU	
Supply Voltage - Max:	5.5 V	
Supply Voltage - Min:	1.8 V	
Tradename:	AVR	
Unit Weight:	0.147798 oz	

Show Similar

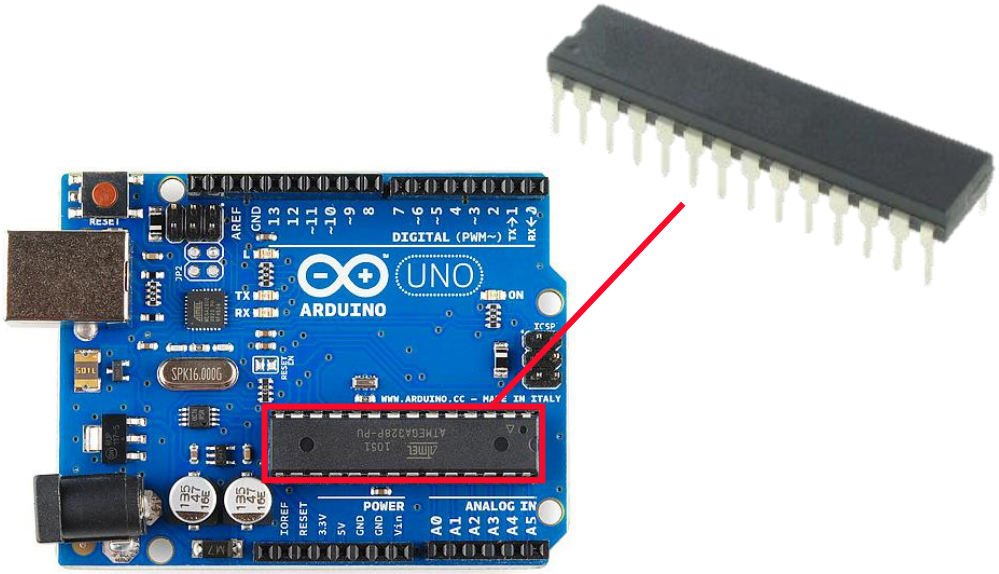
Source: https://www.microchip.com/wwwproducts/en/ATmega328

Maximum Clock Frequency:	20 MHz
Program Memory Size:	32 kB
Data RAM Size:	2 kB
ADC Resolution:	10 bit
Number of I/Os:	23 I/O
Operating Supply Voltage:	1.8 V to 5.5 V
Minimum Operating Temperature:	- 40 C
Maximum Operating Temperature:	+ 85 C
Packaging:	Tube
Height:	4.57 mm
Length:	34.79 mm
Product:	MCU
Program Memory Type:	Flash
Width:	7.49 mm
Brand:	Microchip Technology / Atmel
Data RAM Type:	SRAM
Data ROM Size:	1 kB
Data ROM Type:	EEPROM

ARDUINO ARCHITECTURE

ARDUINO UNO MEMORY: THREE TYPES

ATmega328



FLASH (32KB)

Program Memory
(31.5KB)

Bootloader
(0.5KB)

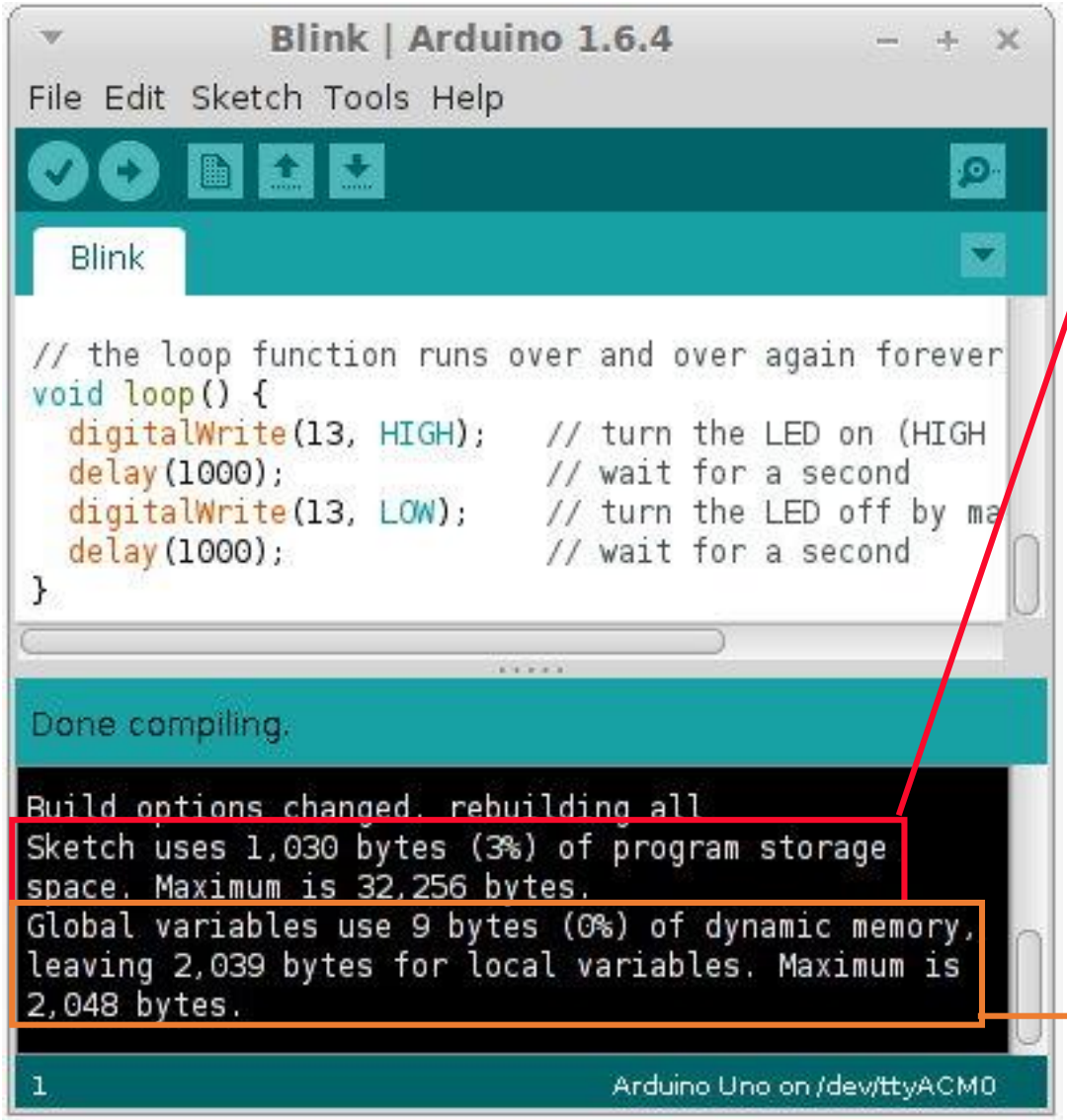
SRAM (2KB)

Data
(Global variables, local variables)

EEPROM (1KB)

Persistent Storage
(Data persists without power; much slower than SRAM)

ARDUINO UNO MEMORY: THREE TYPES



FLASH (32KB)

Program Memory
(31.5KB)

Bootloader
(0.5KB)

SRAM (2KB)

Data
(Global variables, local variables)

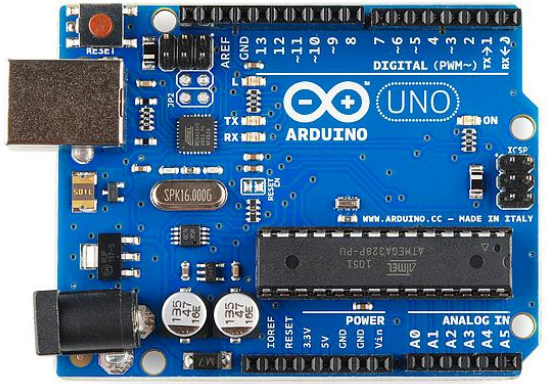
EEPROM (1KB)

Persistent Storage
(Data persists without power; much slower than SRAM)

Memory will **not** be a **huge concern**, at least not at first…

WHAT TO DO IF YOU RUN OUT OF FLASH (PROGRAM) MEMORY?

ARDUINO UNO



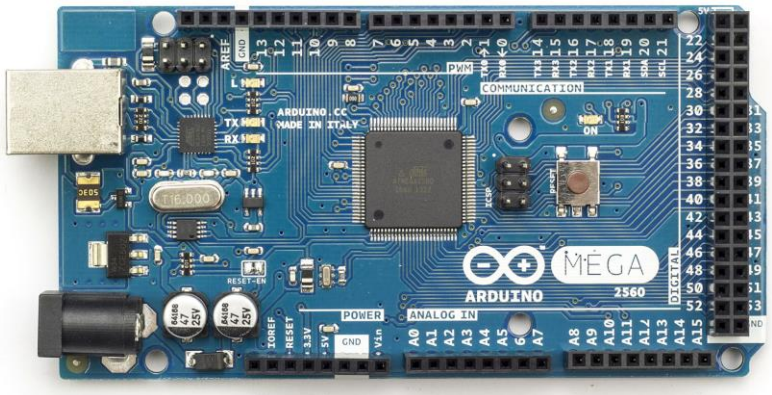
ATMega328 Microcontroller

Flash: 32KB (but 0.5K used for bootloader)

SRAM: 2KB

EEPROM: 1KB

ARDUINO MEGA



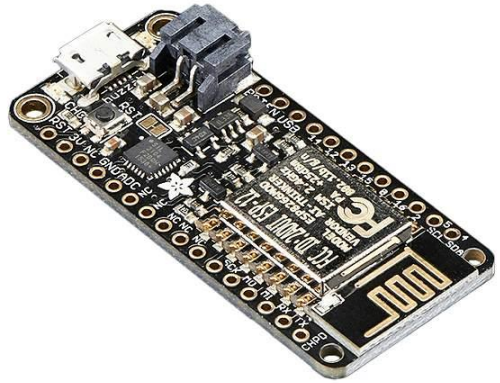
ATMega2560 Microcontroller

Flash: 256KB (but 8K used for bootloader)

SRAM: 8KB

EEPROM: 4KB

ADAFRUIT HUZZAH32 FEATHER



ESP-32 Microcontroller

Flash: 4MB

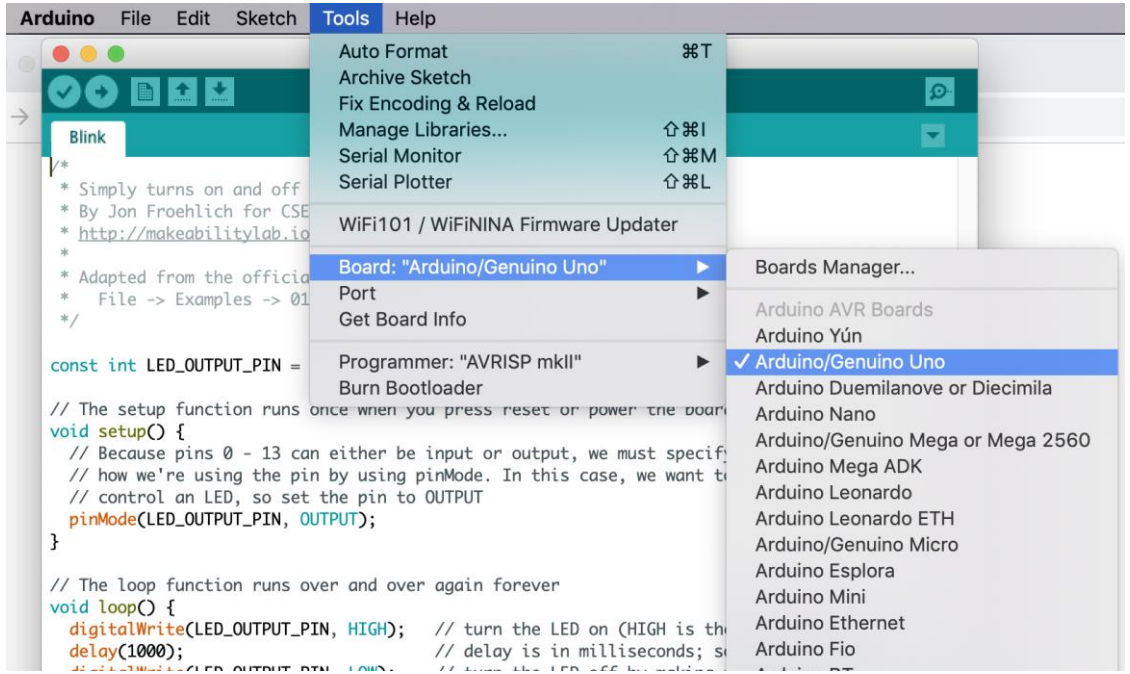
SRAM: 520KB

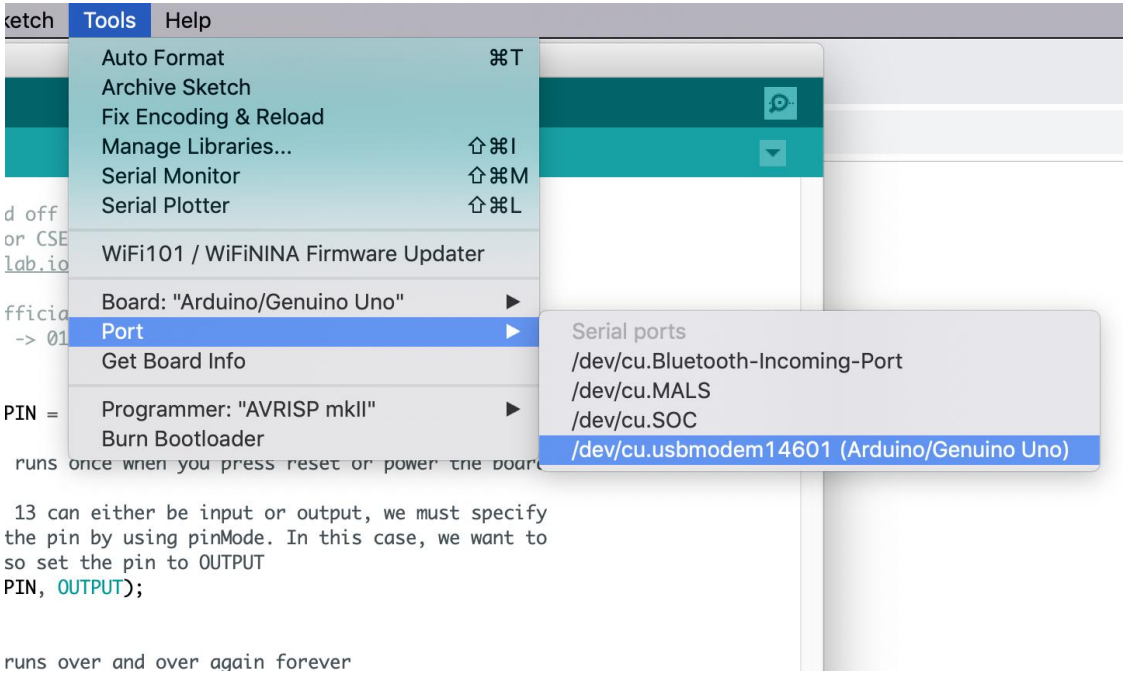
EEPROM: 512bytes (deprecated?)

ARDUINO PROGRAMMING

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

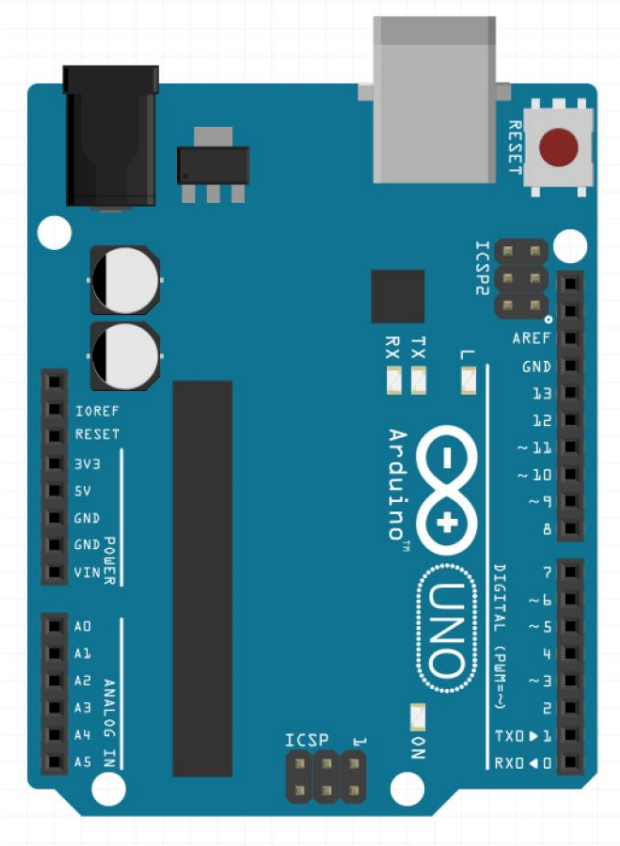
2. SELECT THE PORT

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

INTRODUCTION TO I/O

INPUT VS. OUTPUT

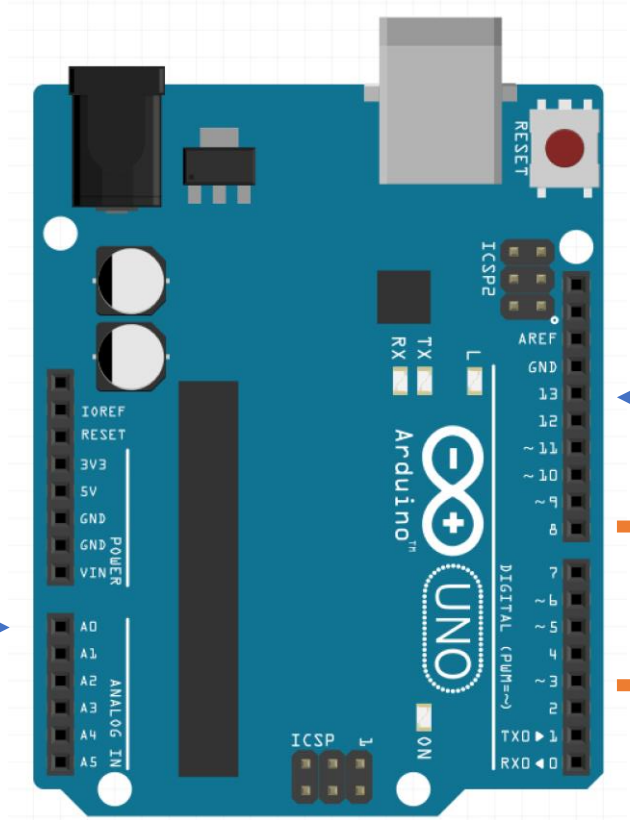
Input and output is with respect to the microcontroller board. Input is anything that the Arduino Uno reads in (*e.g.*, from a sensor) and output is anything that the Arduino Uno writes out (*e.g.*, to a light or motor)



INPUT VS. OUTPUT EXAMPLES

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)

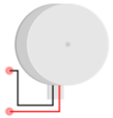
Analog Input on A0
Reads in any value between 0V or 5V using the analogRead function. In this case, the value of a photocell



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.

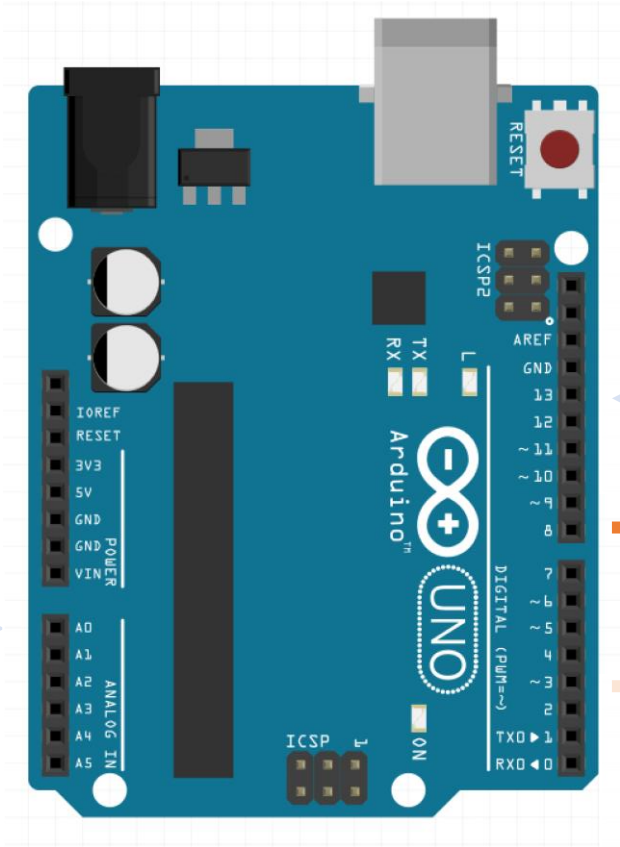


INTRO TO DIGITAL OUTPUT: WRITING 0V AND 5V

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.

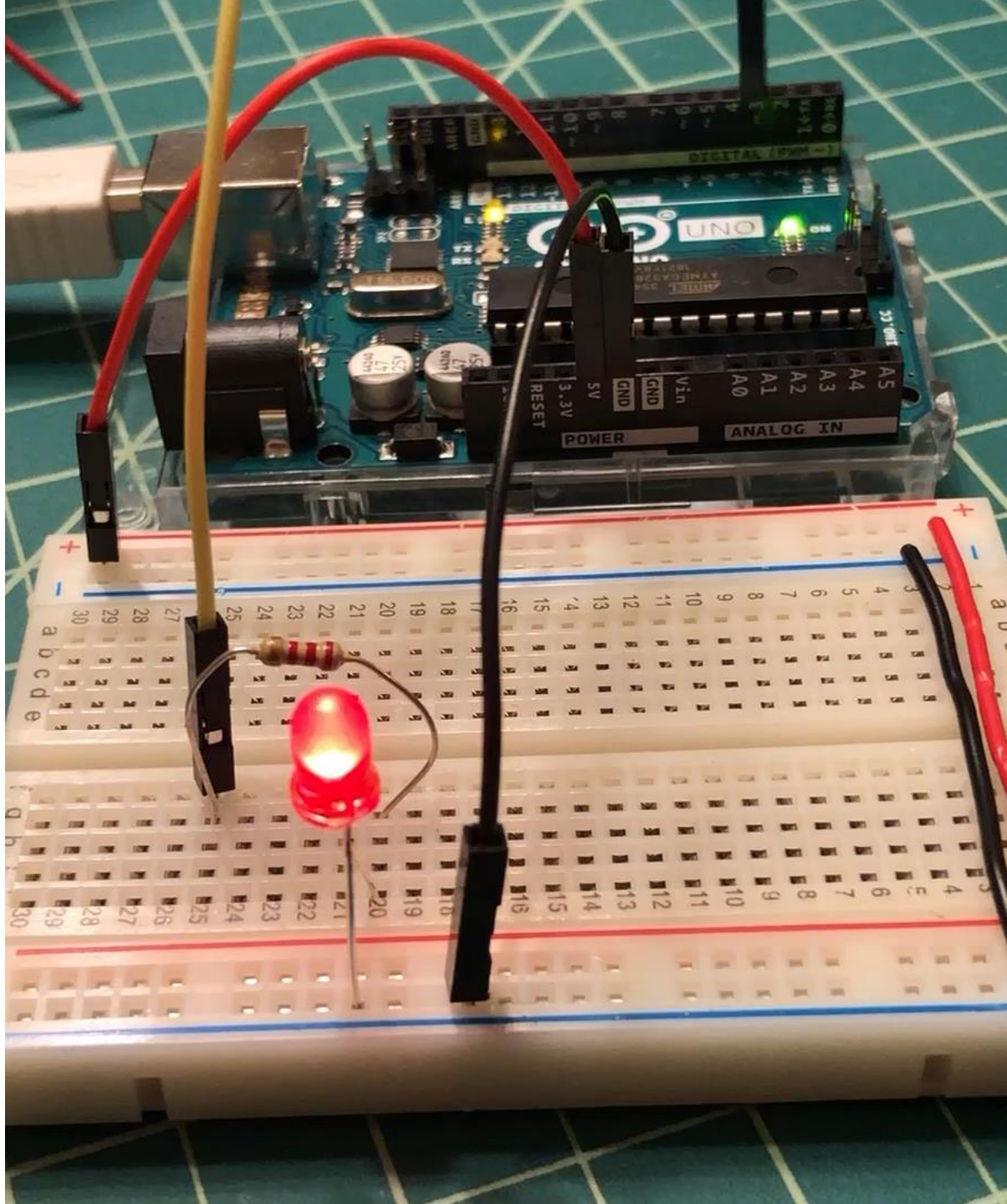
DIGITAL OUTPUT

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.

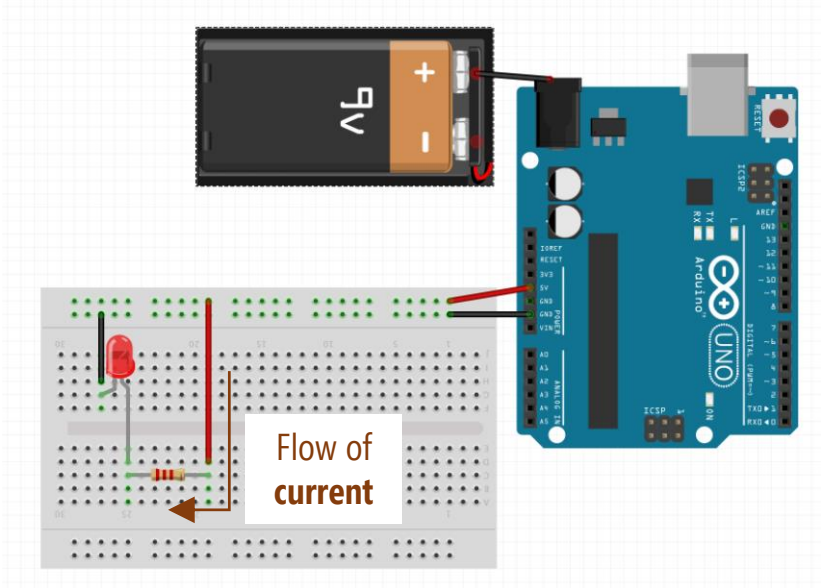
When you **only** want to turn something "on" and "off" then this implies digital output, so we will use the **digitalWrite** method.

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



BEFORE: LED ALWAYS ON VIA 5V PIN

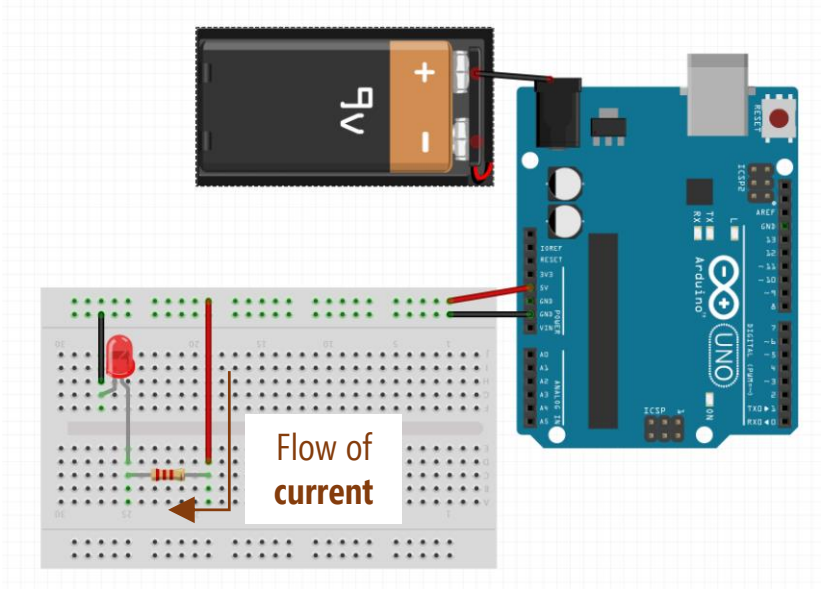
Previous Circuit: LED Powered via 5V pin



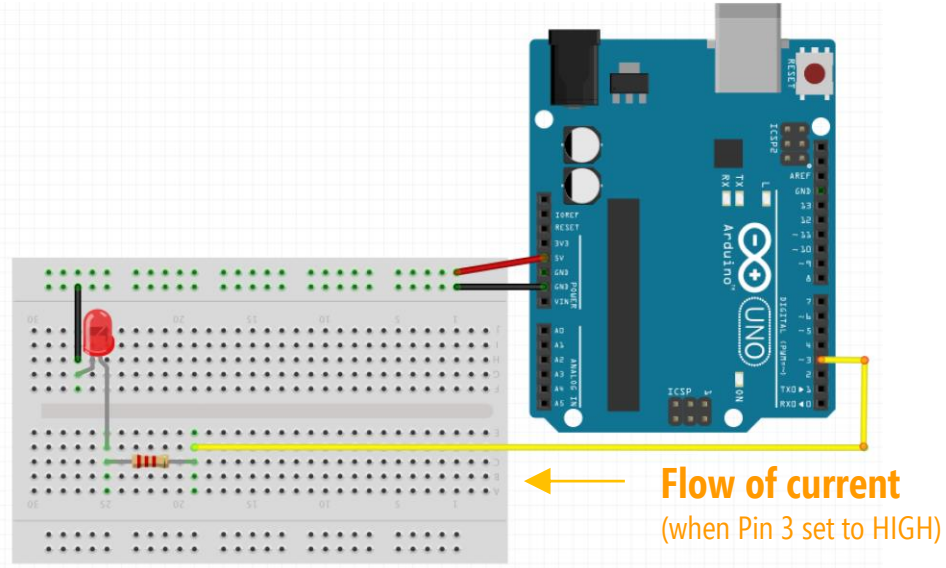
DIGITAL OUTPUT

FLASH LED ON/OFF USING ARDUINO UNO

Previous Circuit: LED Powered via 5V pin

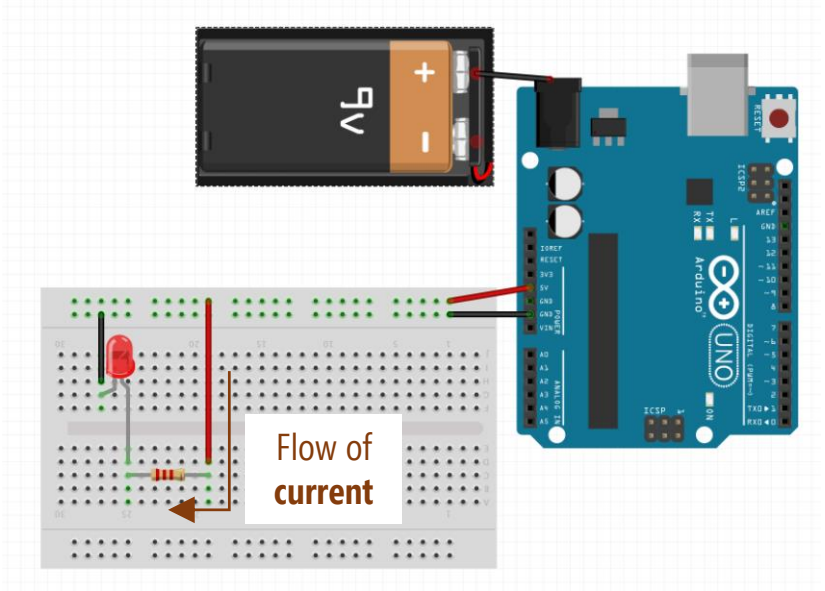


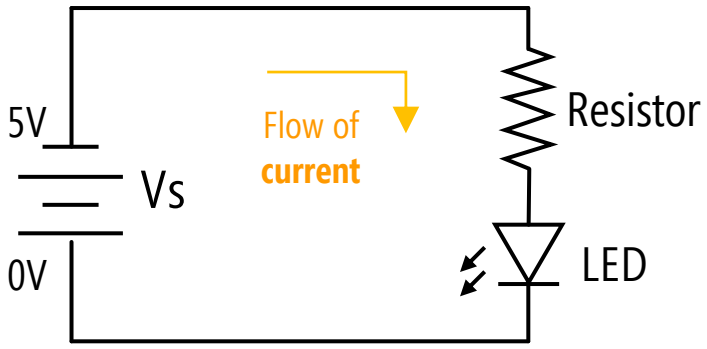
Build Circuit: Flash LED on/off via the pin 3



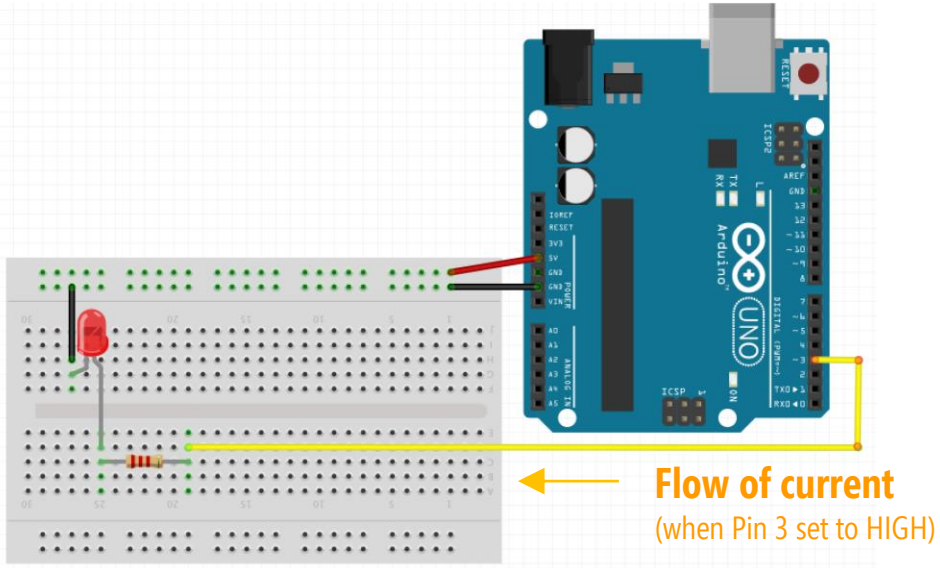
FLASH LED ON/OFF USING ARDUINO UNO

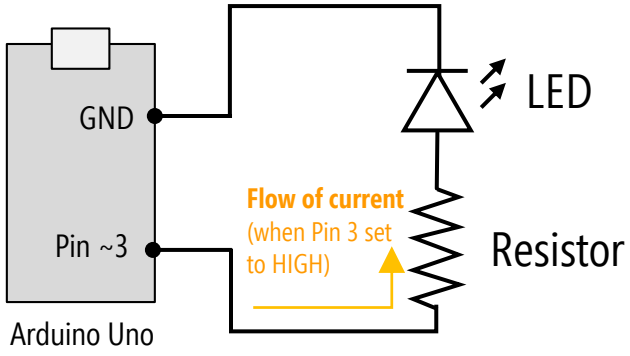
Previous Circuit: LED Powered via 5V pin





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

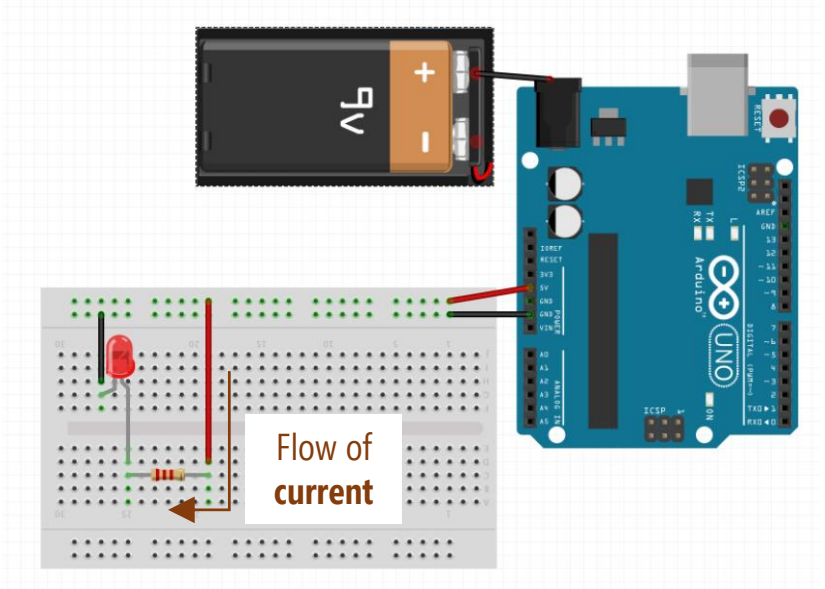


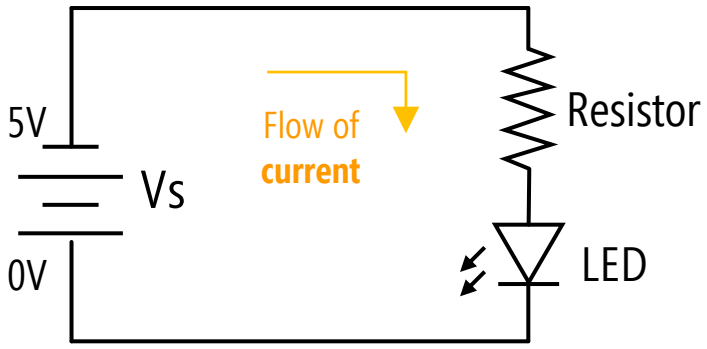


DIGITAL OUTPUT

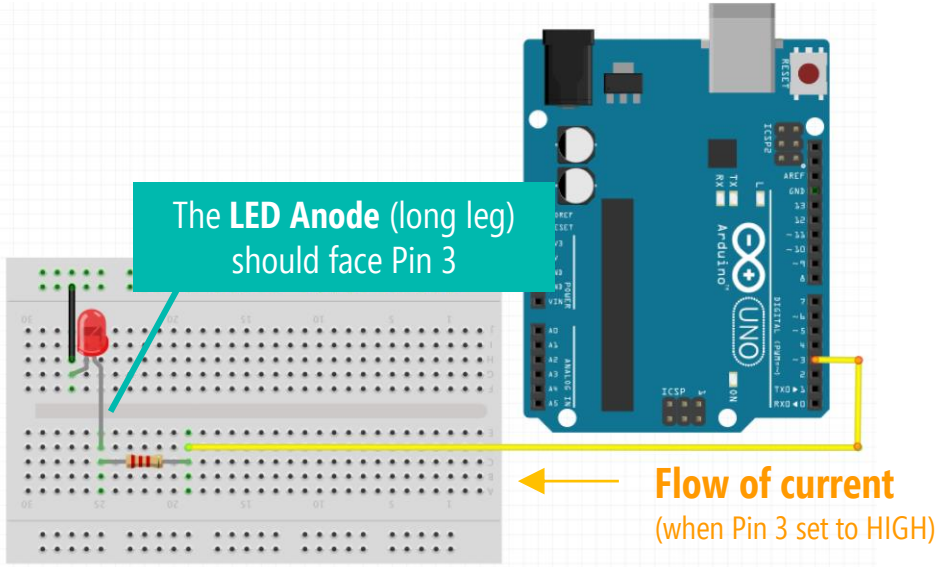
FLASH LED ON/OFF USING ARDUINO UNO

Previous Circuit: LED Powered via 5V pin



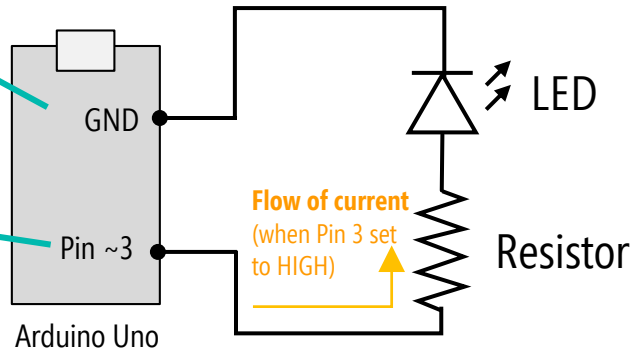


Build Circuit: Flash LED on/off via the pin 3



GND is always the lowest potential voltage point in our circuit

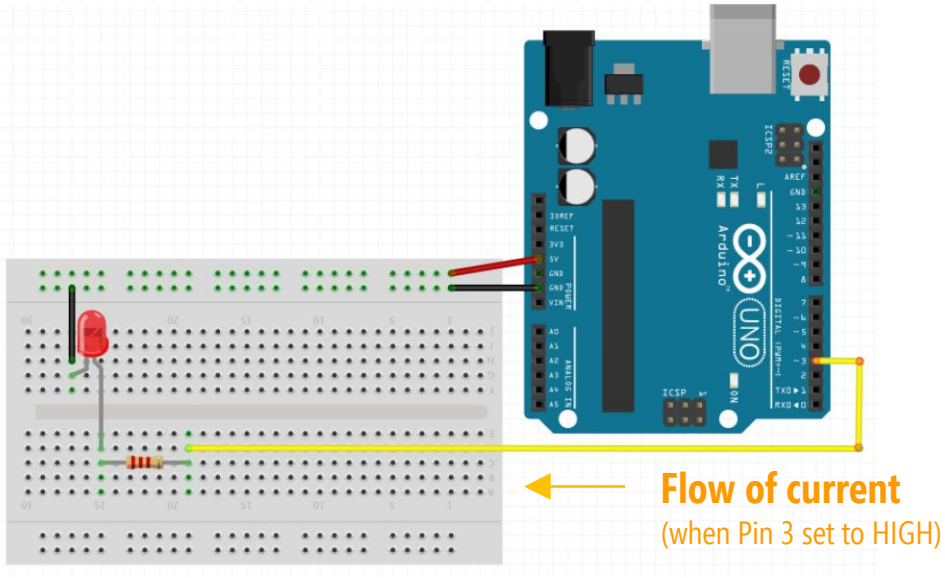
We will program **Pin 3** to alternate between 5V and 0V. When 5V, the LED will turn on.

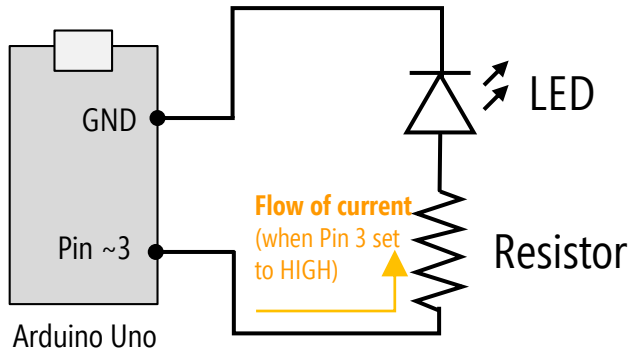


DIGITAL OUTPUT

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

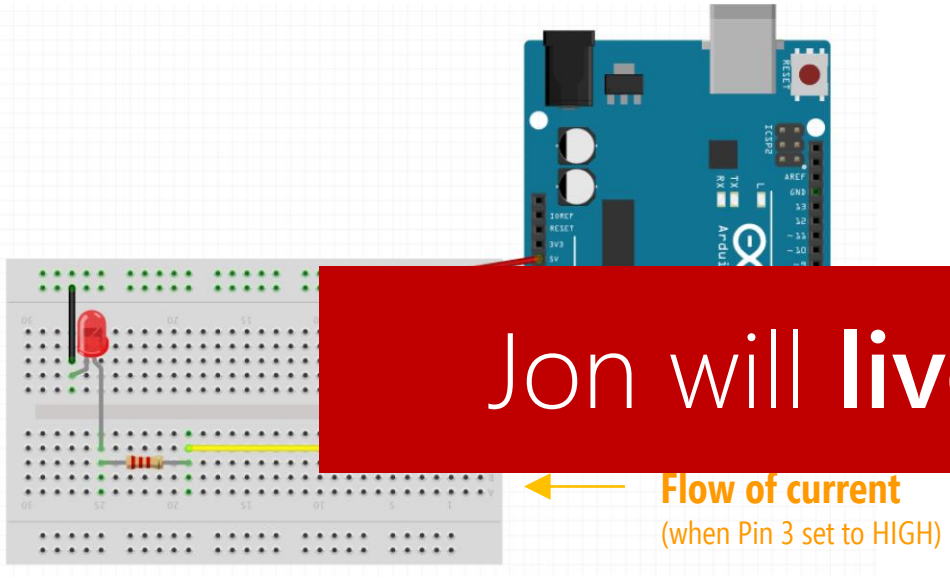
Blink

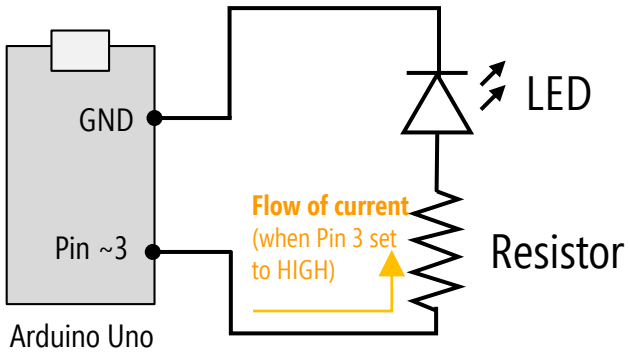
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

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

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

Blink

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

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

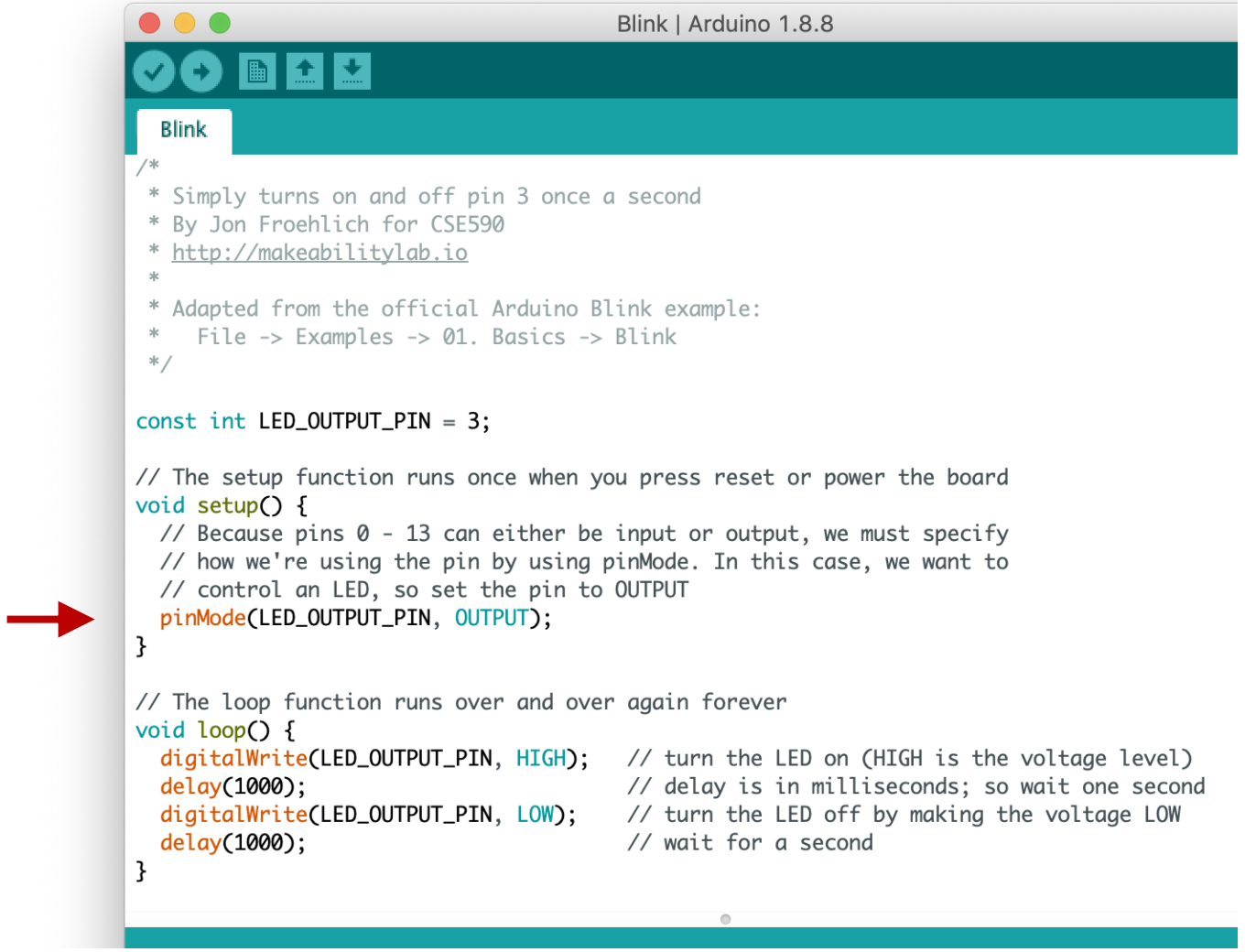
```
// The setup function runs once when you press reset or power the board
void setup() {
  // Because pins 0 - 13 can either be input or output, we must specify
  // how we're using the pin by using pinMode. In this case, we want to
  // control an LED, so set the pin to OUTPUT
  pinMode(LED_OUTPUT_PIN,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

Let's **walkthrough** what we made

DIGITAL OUTPUT

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()**



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

DIGITAL OUTPUT

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()**

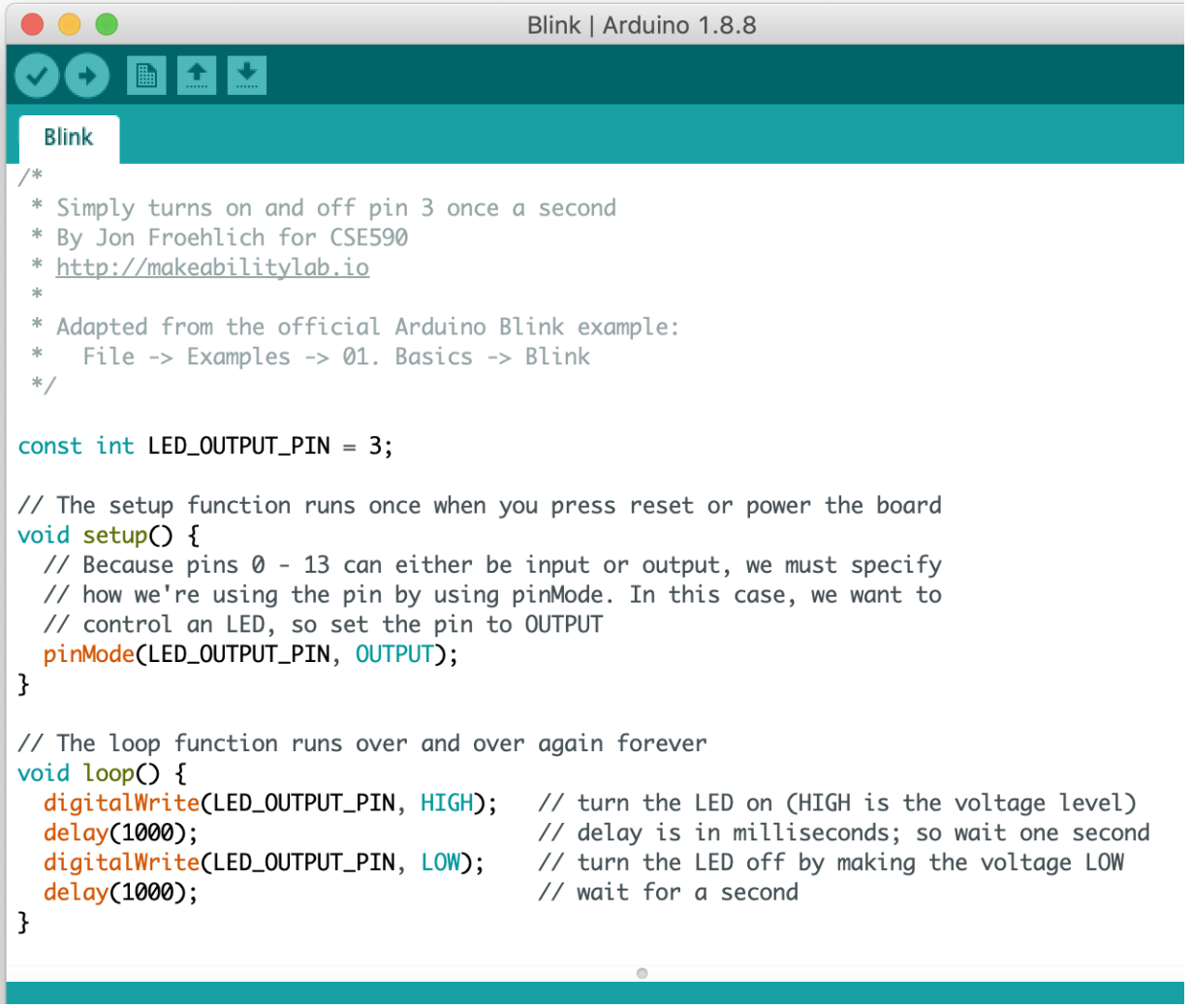
Syntax

`pinMode(pin, mode)`

Parameters

`pin`: the pin number whose mode you wish to set

`value`: INPUT or OUTPUT

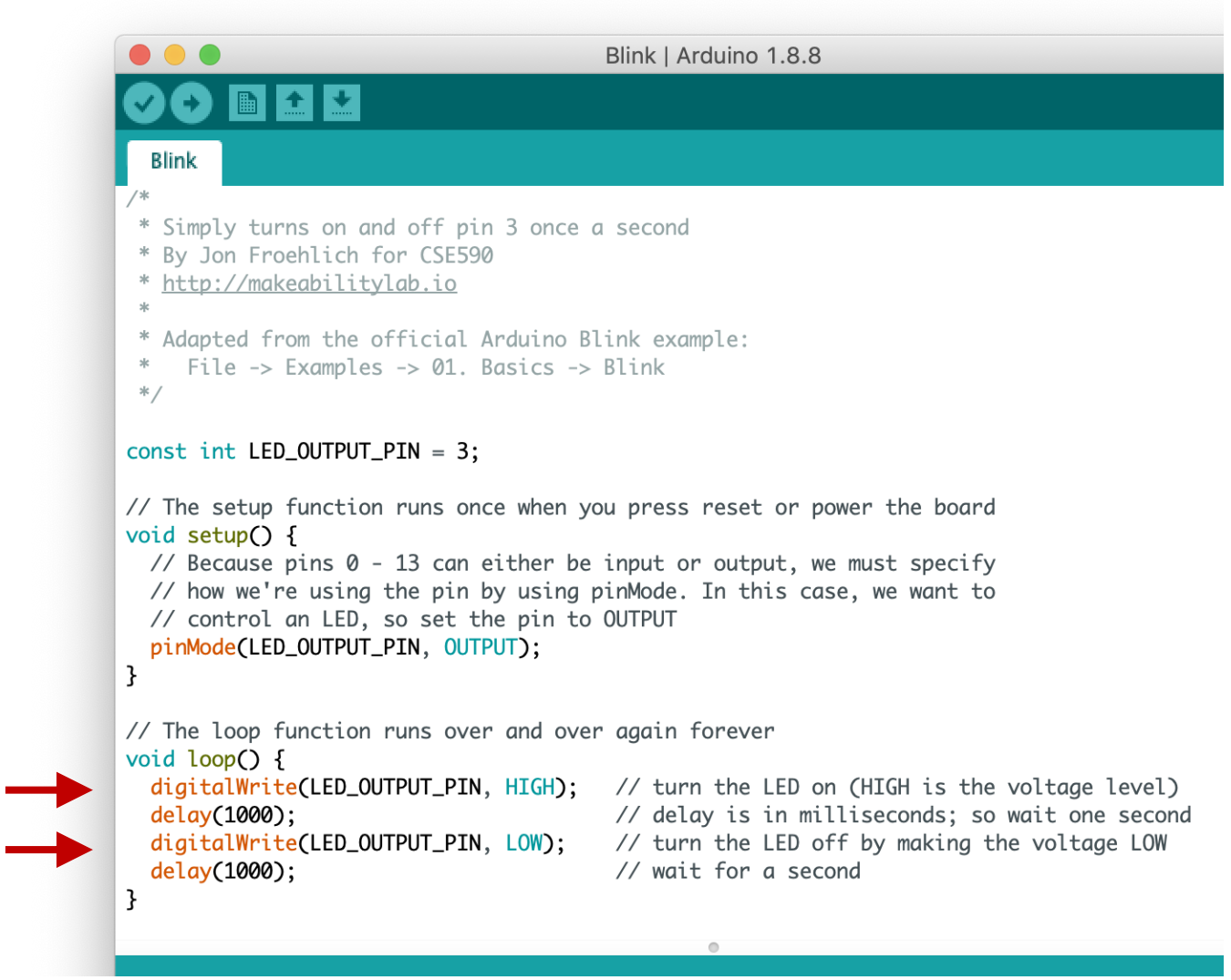


https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

DIGITAL OUTPUT

digitalWrite(pin, value)

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



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

DIGITAL OUTPUT

digitalWrite(pin, value)

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

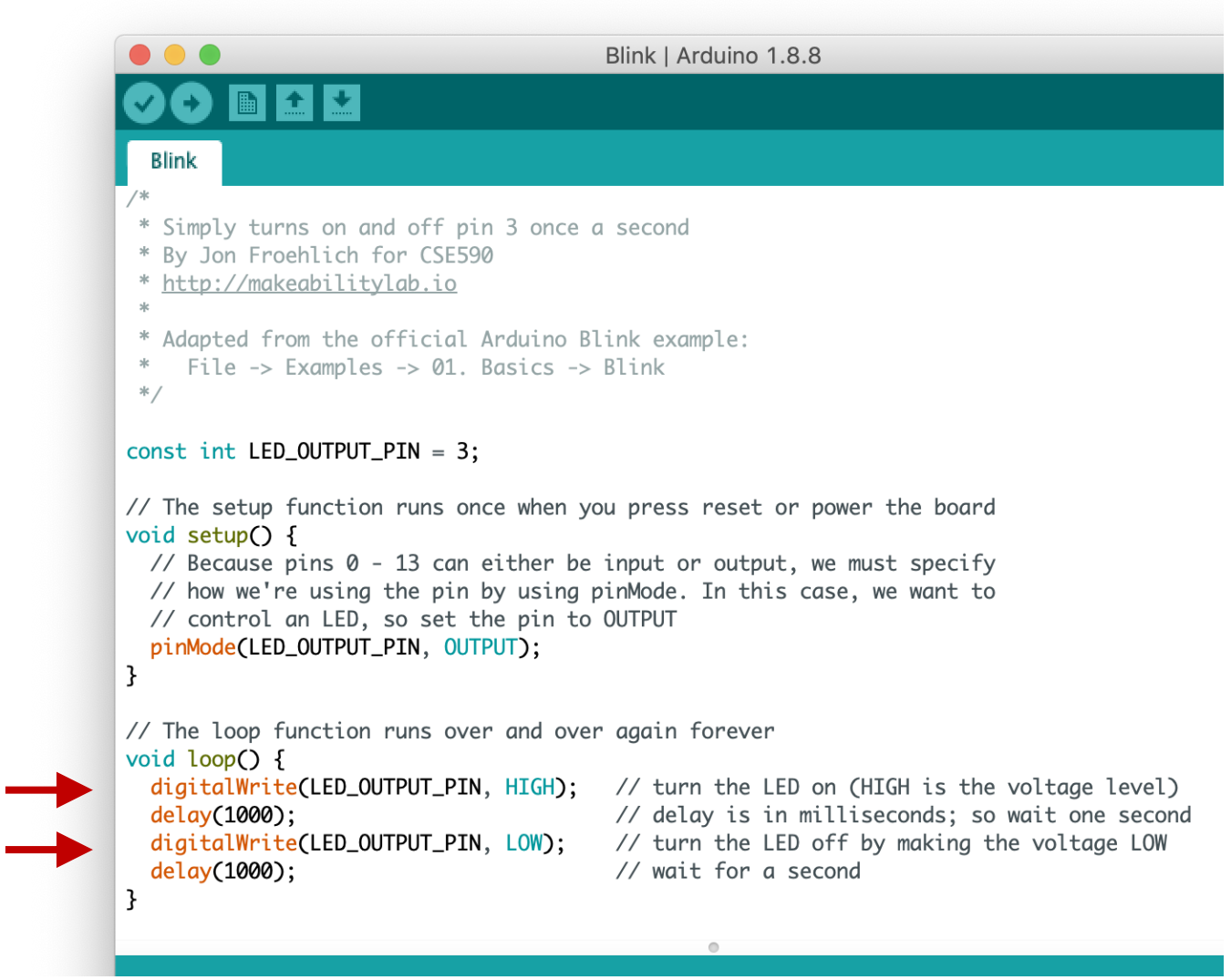
Syntax

`digitalWrite(pin, value)`

Parameters

`pin`: the pin number

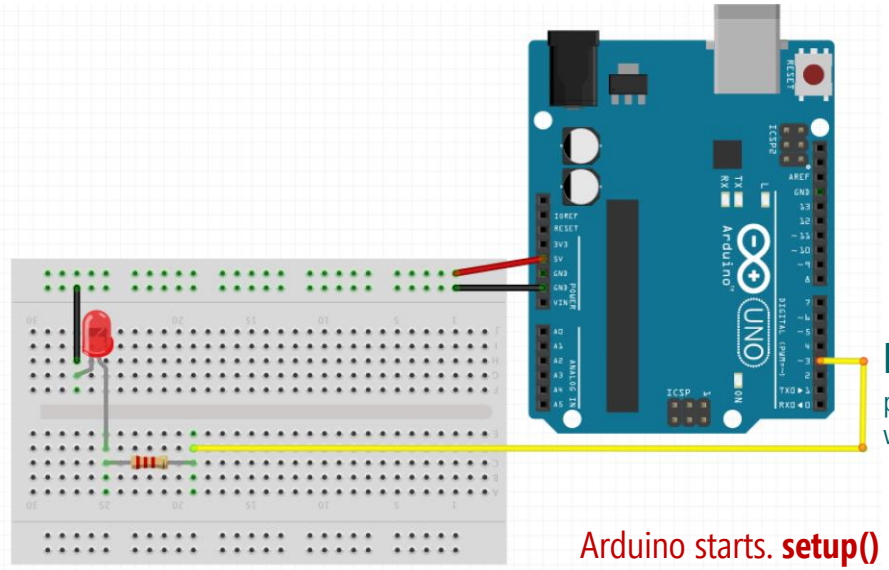
`value`: HIGH (5V) or LOW (0V)



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

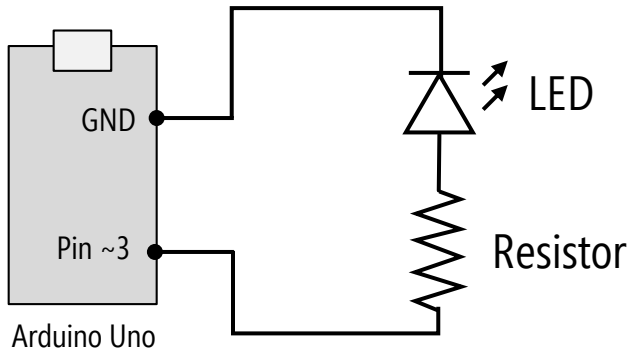
LET'S WALK THROUGH WHAT'S HAPPENING HERE

Circuit: Flash LED on/off via the pin 3



Pin 3
pinMode =unset
value=unset

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



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

Blink | Arduino 1.8.8

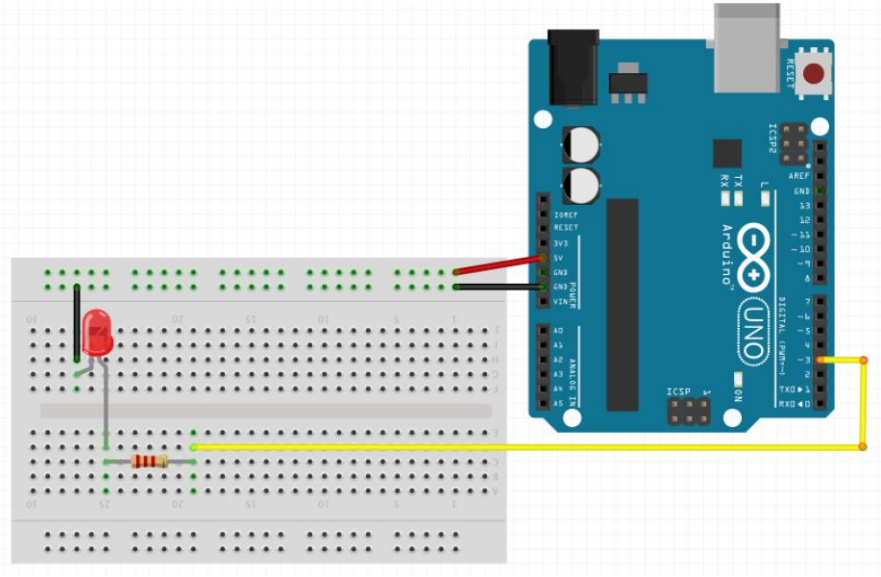
Blink

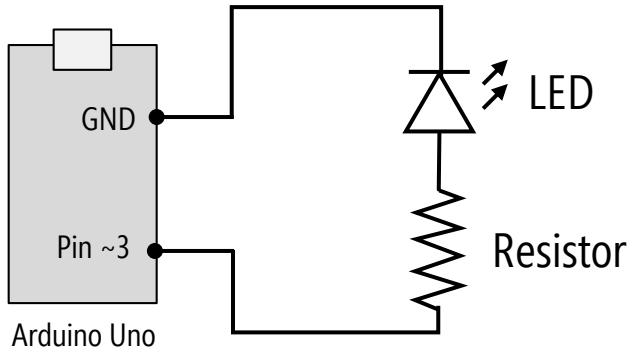
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

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

PIN 3 SET TO OUTPUT

Circuit: Flash LED on/off via the pin 3





Pin 3 is set to **output** because we want to use it to control something (in this case, an LED) rather than to receive input (e.g., from a button). The default value is LOW.

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

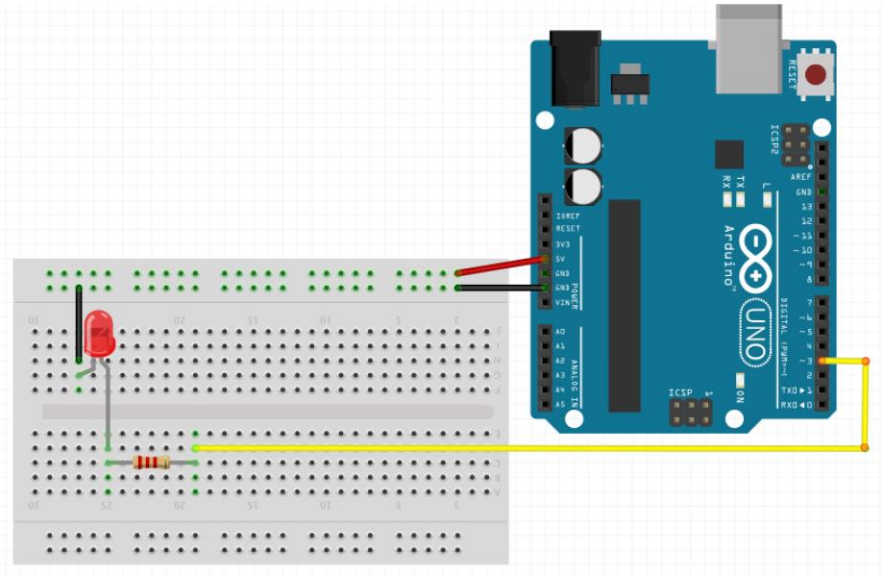
Blink | Arduino 1.8.8

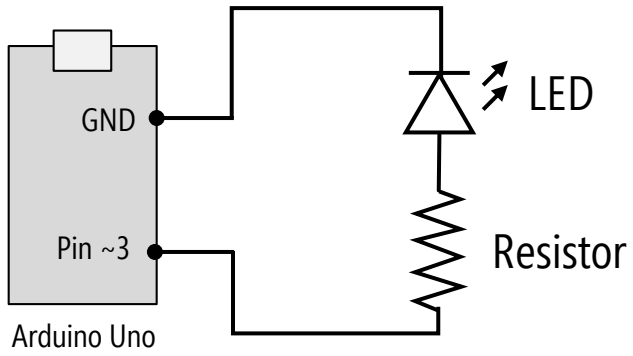
Blink

```
/*
 * 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
}
```

SETUP() COMPLETES, LOOP() FUNCTION CALLED…

Circuit: Flash LED on/off via the pin 3





After setup() completes, the Arduino calls the **loop()** function, which is called again and again automatically.

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

Blink | Arduino 1.8.8

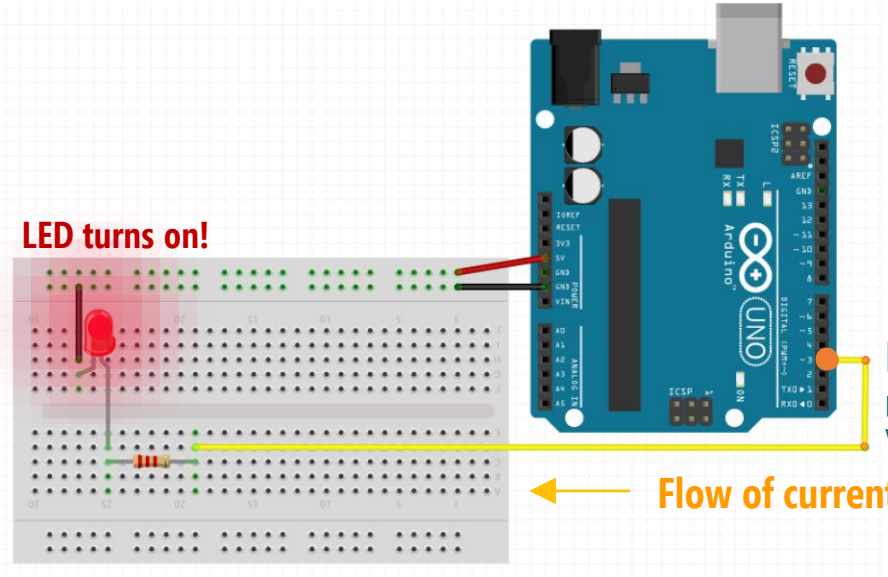
Blink

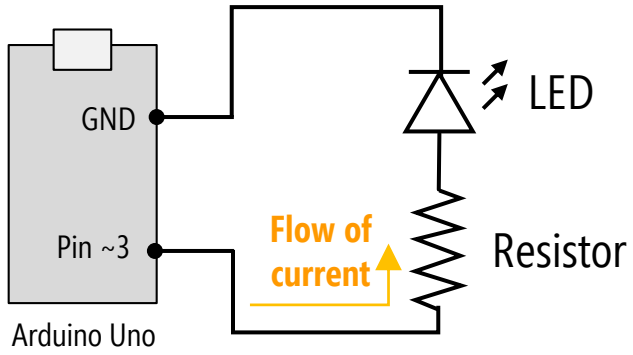
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

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

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

Circuit: Flash LED on/off via the pin 3





Now write out **HIGH (5V)**
Wait for one second
on Pin 3. As soon as we
(State is retained so Pin
do that, current flows and
3 is still HIGH)
our light turns on

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

Blink | Arduino 1.8.8

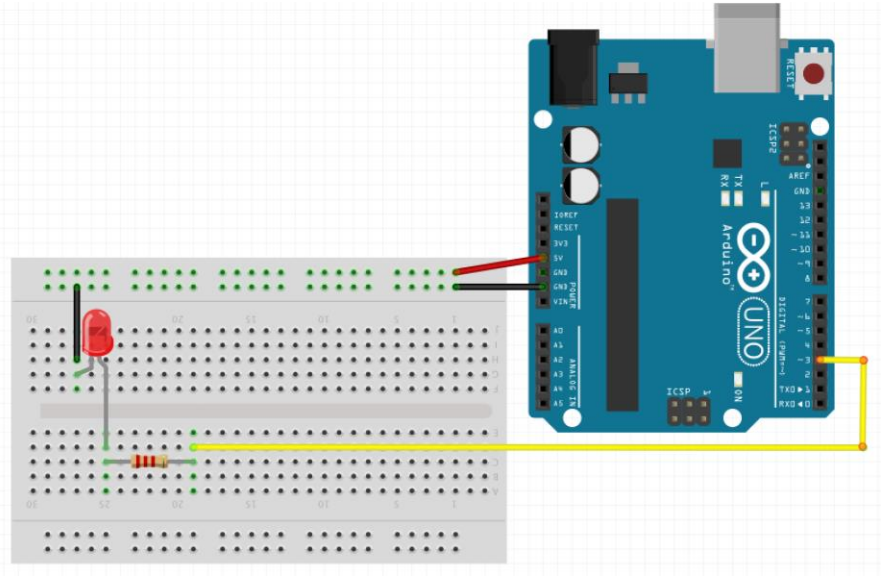
Blink

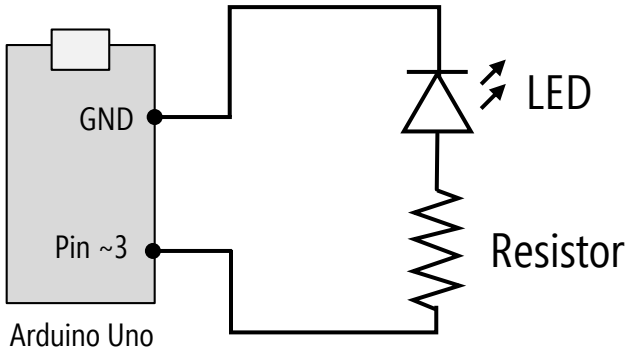
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

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

PIN 3 SET TO LOW (0V)

Circuit: Flash LED on/off via the pin 3





Now write out **LOW** (0V) to **Pin 3.** Now Pin 3 voltage is equivalent to GND, so there is no voltage difference and thus no current will flow!

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

Blink | Arduino 1.8.8

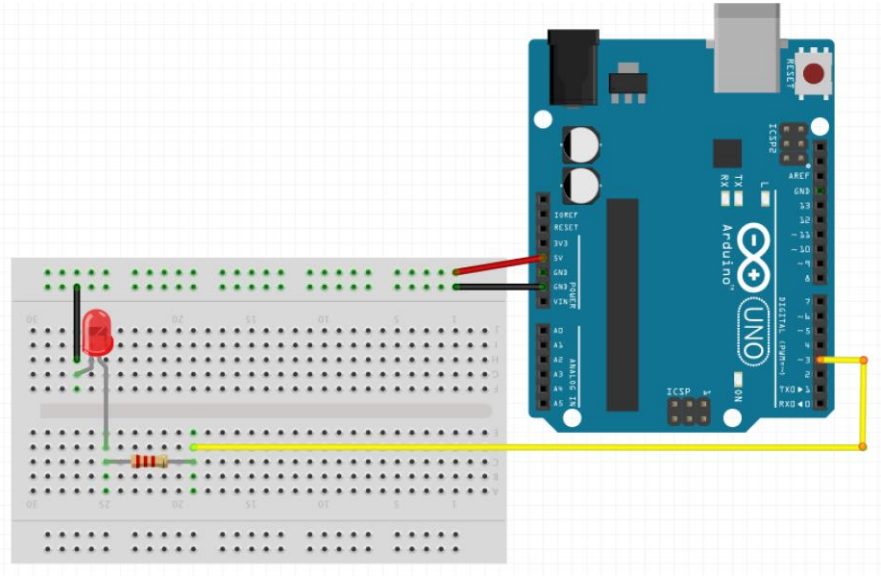
Blink

```
/*
 * 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
}
```

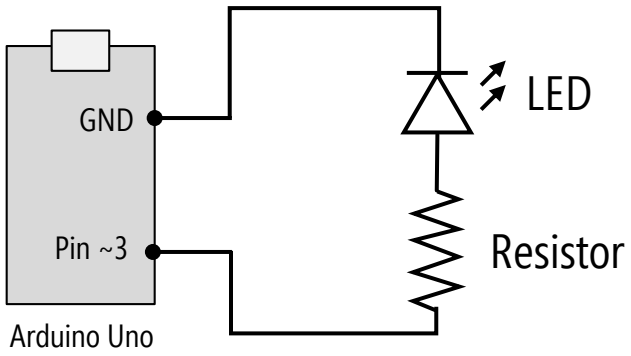
DIGITAL OUTPUT

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

Circuit: Flash LED on/off via the pin 3



Pin 3
pinMode =OUTPUT
value=LOW



Wait for another second

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

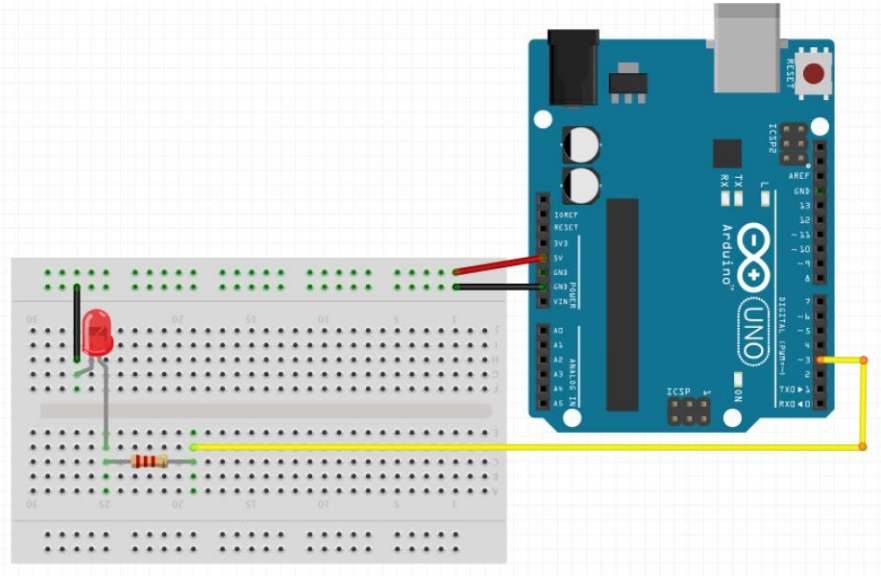
Blink | Arduino 1.8.8

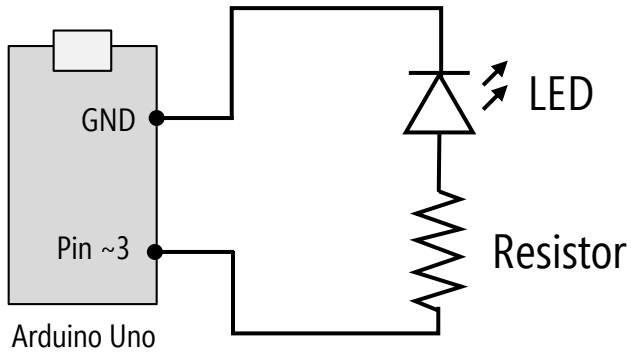
Blink

```
/*
 * 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
}
```

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

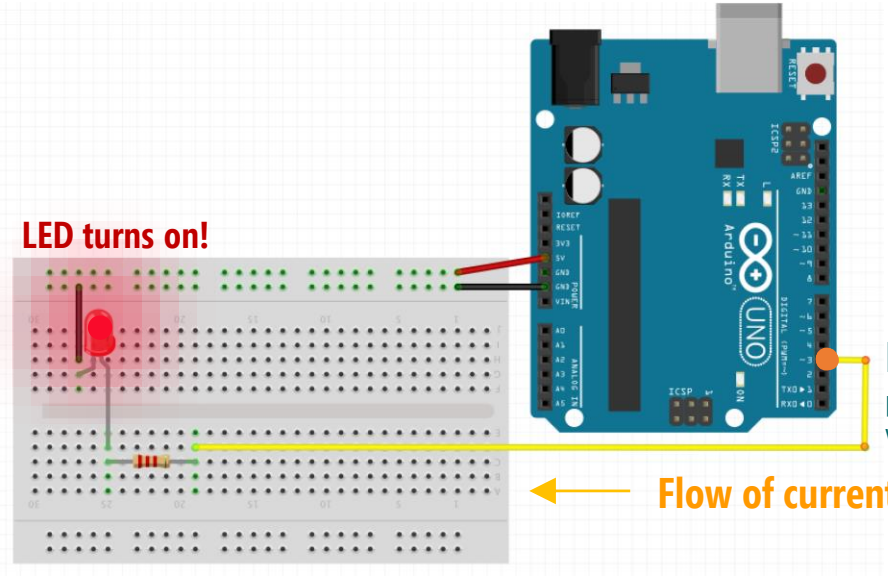
Blink

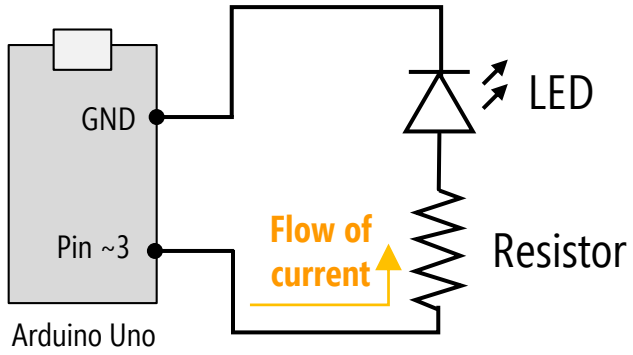
```
/*
 * Simply turns on and off pin 3 once a second
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 *
 * Adapted from the official Arduino Blink example:
 *   File -> Examples -> 01. Basics -> Blink
 */

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

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

Circuit: Flash LED on/off via the pin 3





Now write out **HIGH (5V)**
Wait for one second
on Pin 3. As soon as we
(State is retained so Pin
do that, current flows and
3 is still HIGH)
our light turns on

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

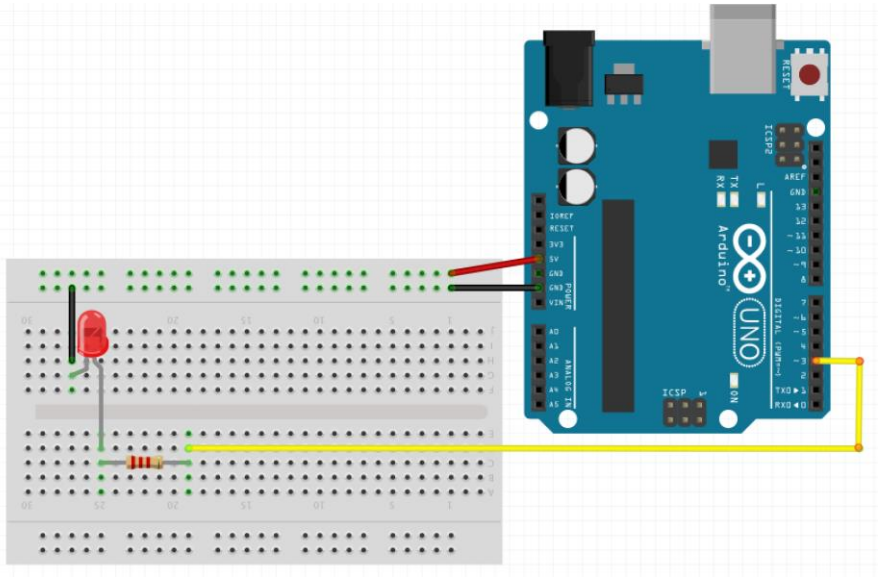
Blink | Arduino 1.8.8

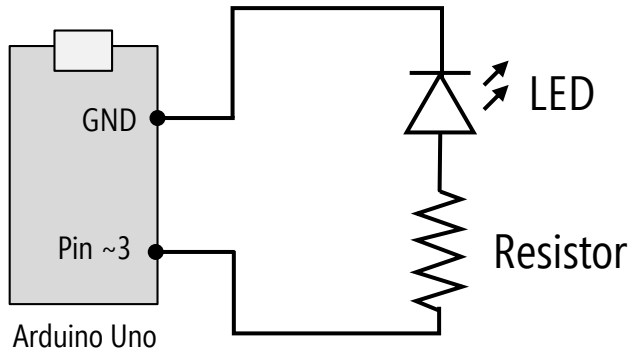
Blink

```
/*
 * 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
}
```

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

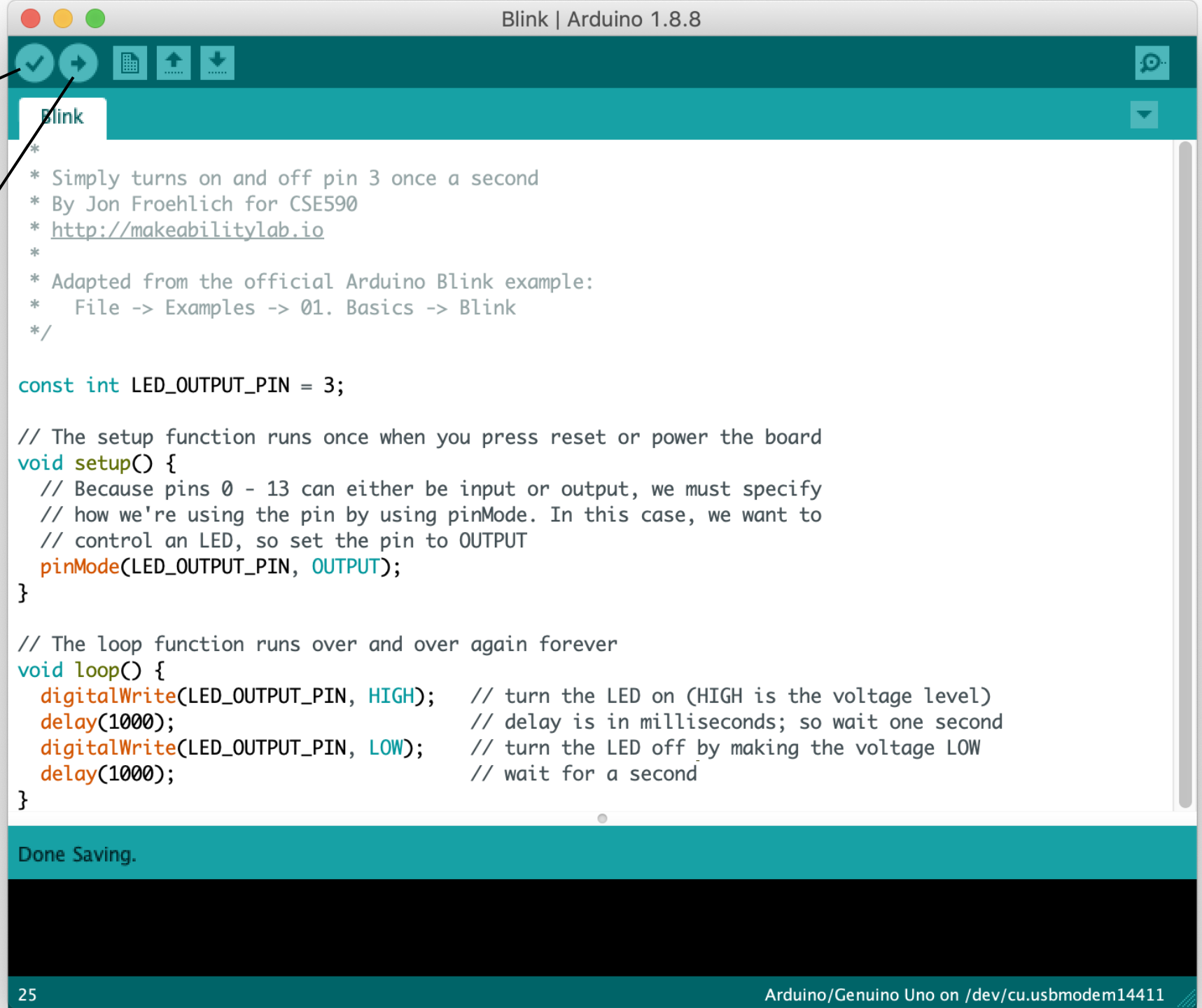
Blink | Arduino 1.8.8

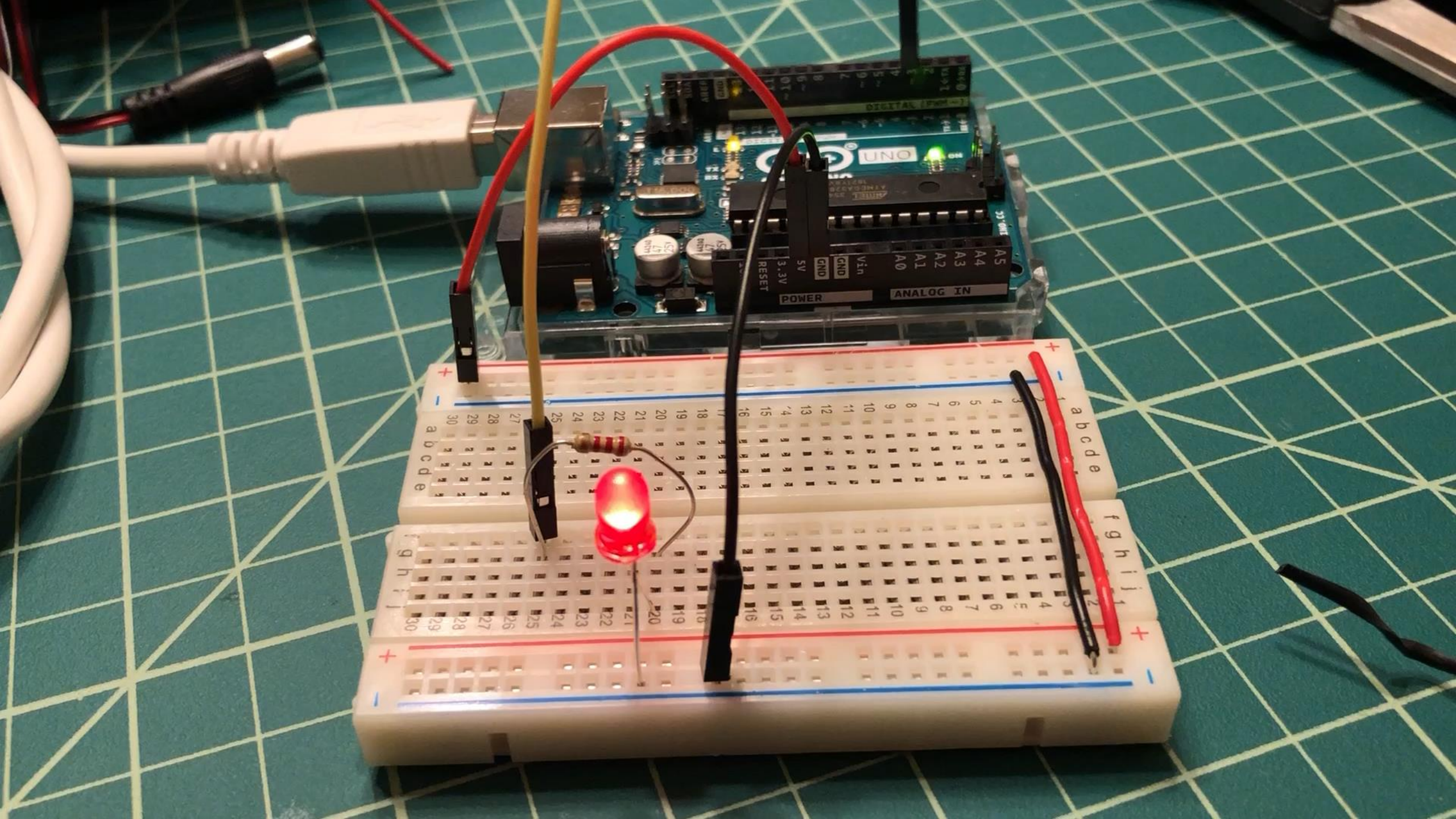
Blink

```
/*
 * 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
}
```

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

Upload your program to the Arduino




UNO
POWER
ANALOG IN
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

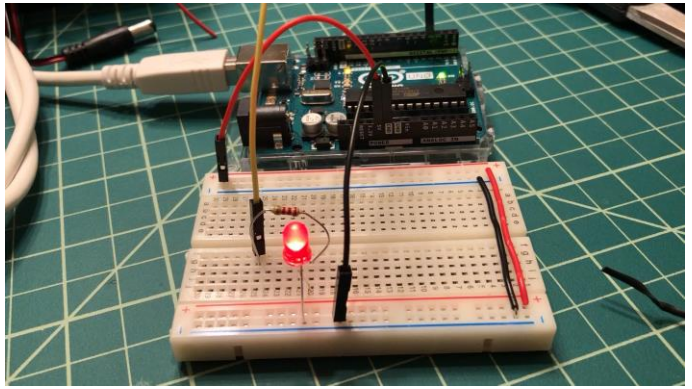
HOW WOULD YOU MAKE THE LED FLASH FASTER?

```
// 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
}
```

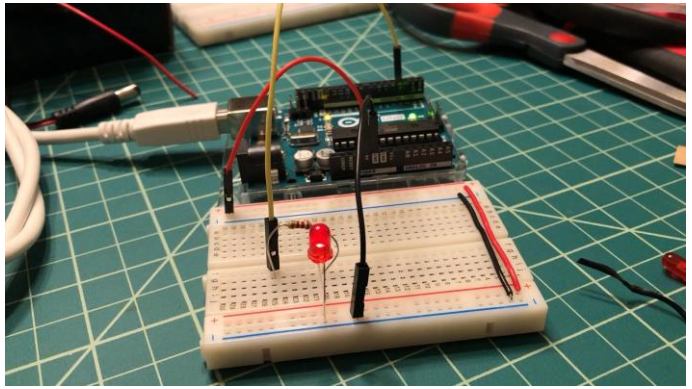
DIGITAL OUTPUT

HOW WOULD YOU MAKE THE LED FLASH FASTER?

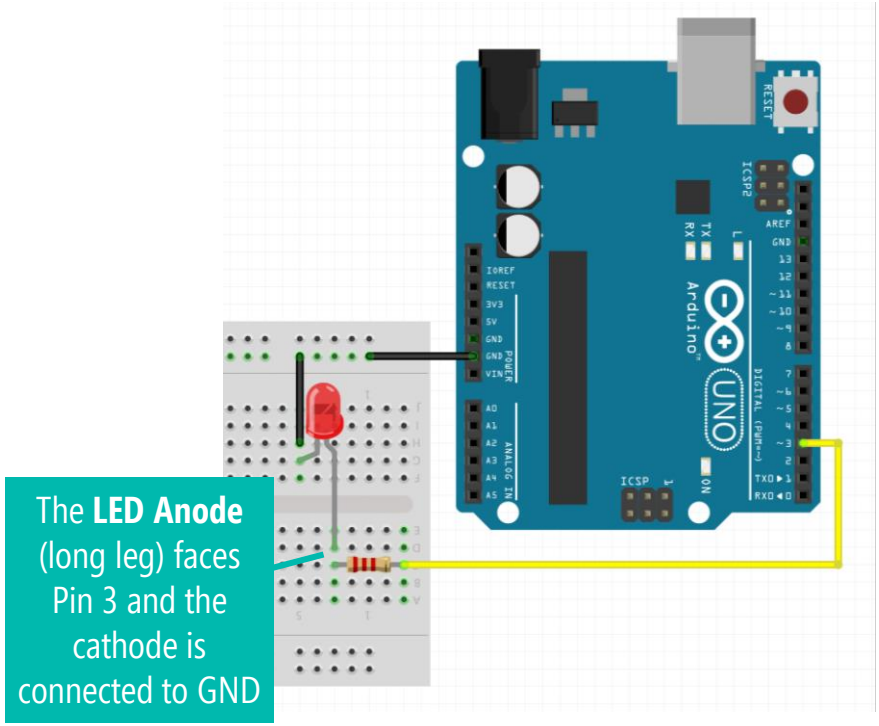
```
// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_OUTPUT_PIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                          // delay is in milliseconds; so wait one second
  digitalWrite(LED_OUTPUT_PIN, LOW);    // turn the LED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



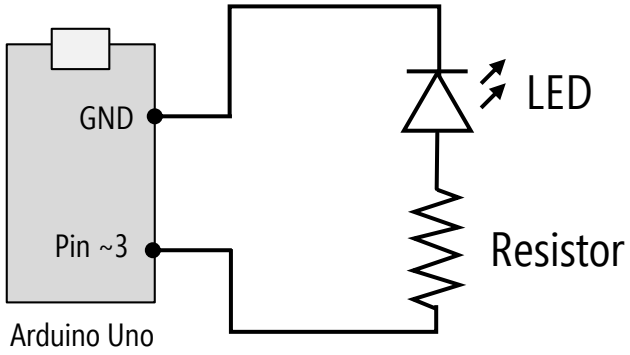
```
// The loop function runs over and over again forever
void loop() {
  digitalWrite(LED_OUTPUT_PIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(100);                          // delay is in milliseconds; so wait 100ms
  digitalWrite(LED_OUTPUT_PIN, LOW);   // turn the LED off by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



HOW COULD WE CHANGE CIRCUIT SO WRITING 'HIGH' WOULD TURN OFF LED RATHER THAN ON?

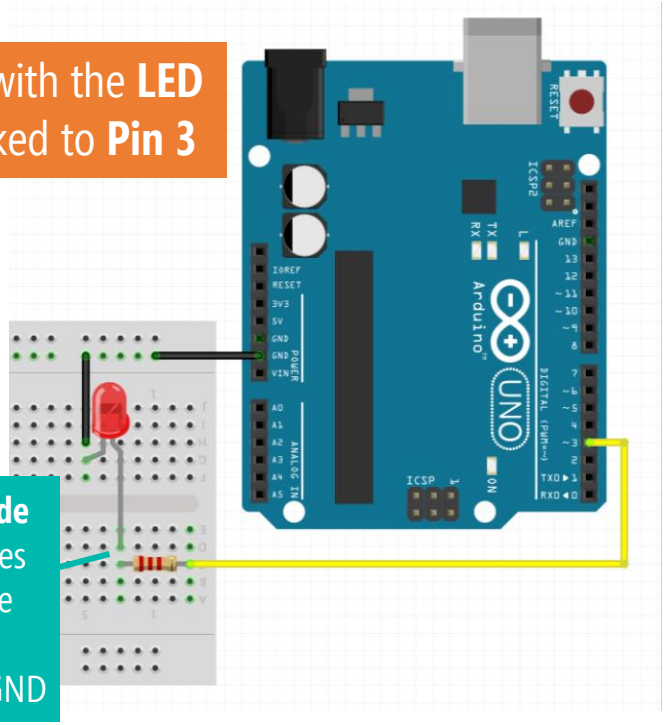


The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND

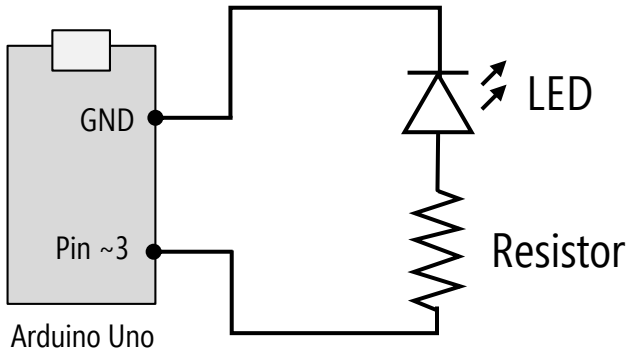


WE CAN HOOK UP THE CIRCUIT ANOTHER WAY BUT IT'S LESS INTUITIVE

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

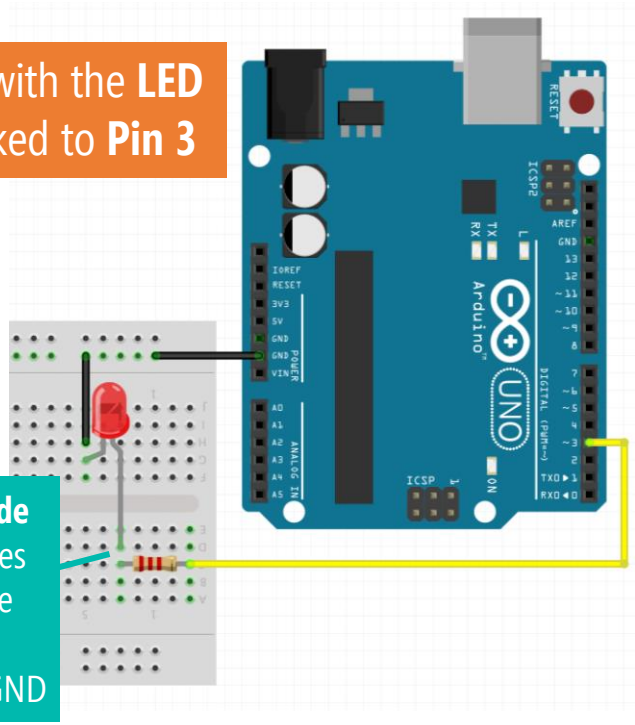


The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND

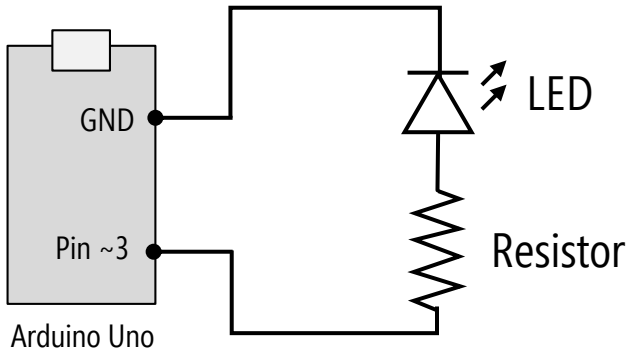


WE CAN HOOK UP THE CIRCUIT ANOTHER WAY BUT IT'S LESS INTUITIVE

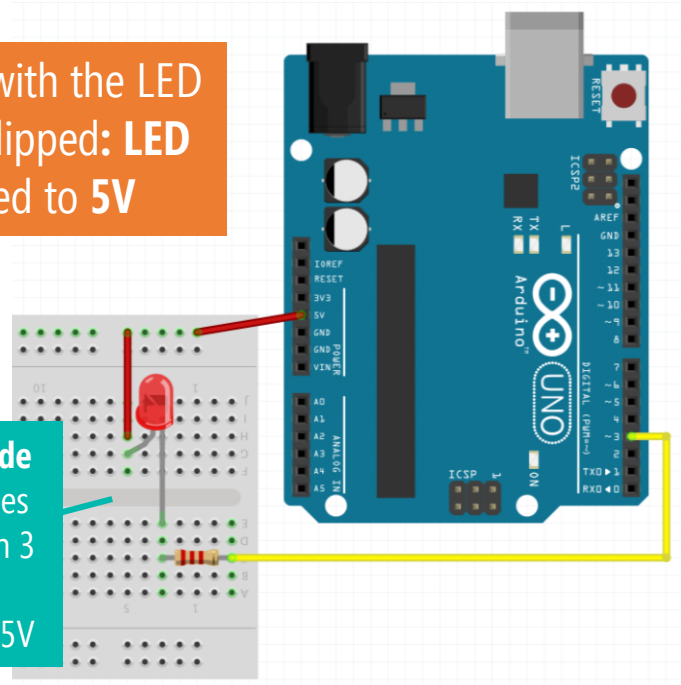
Old circuit with the **LED anode** hooked to **Pin 3**



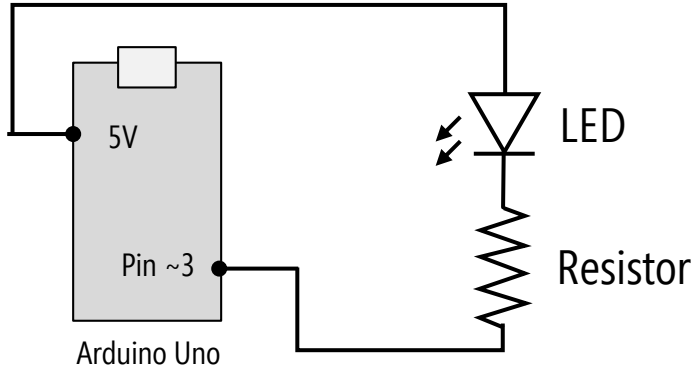
The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND



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

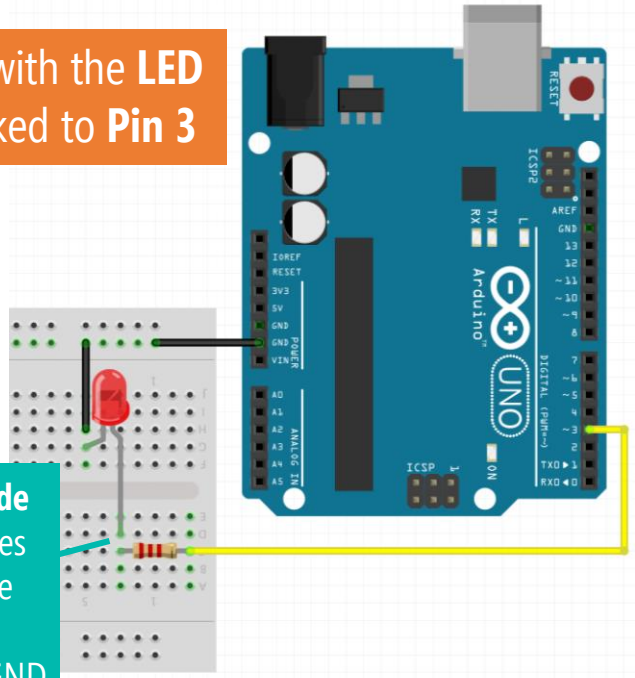


The **LED Anode** (long leg) faces away from Pin 3 and is now connected to 5V

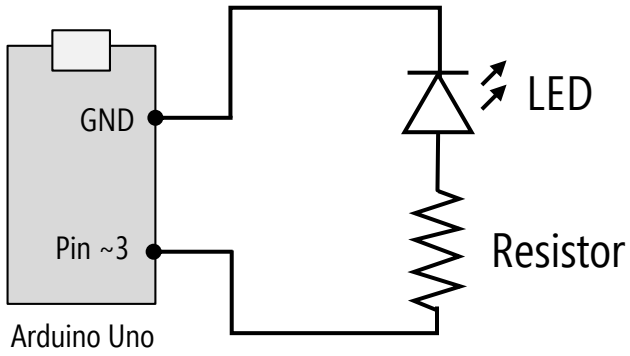


WHAT HAPPENS WITH THIS NEW CIRCUIT WHEN PIN 3 SET TO LOW?

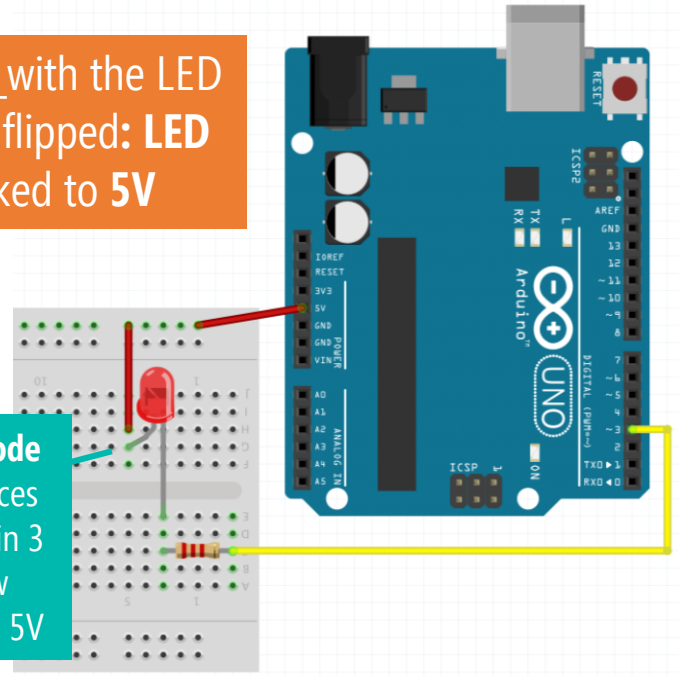
Old circuit with the **LED anode** hooked to **Pin 3**



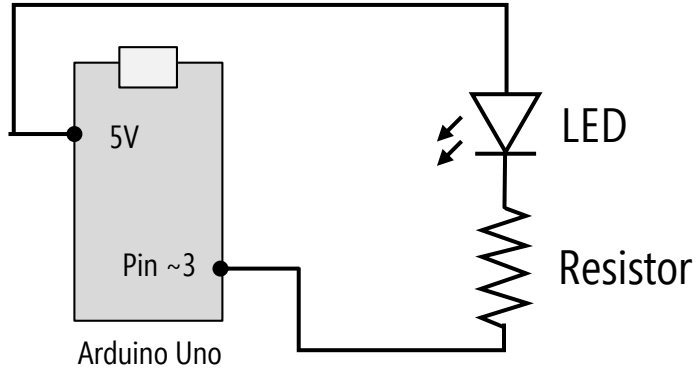
The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND



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



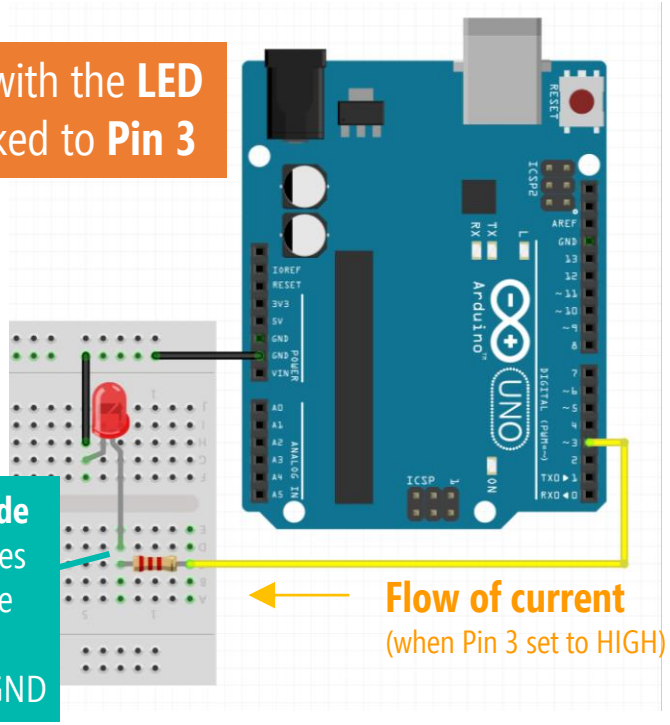
The **LED Anode** (long leg) faces away from Pin 3 and is now connected to 5V



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

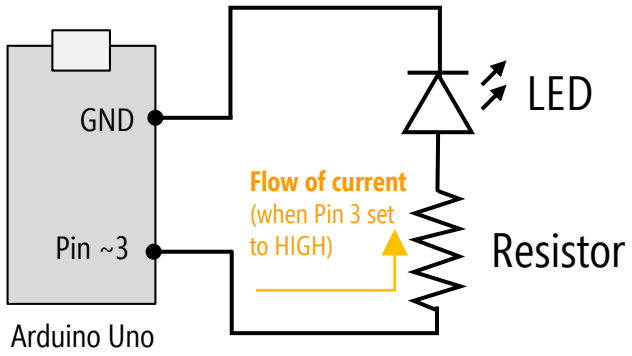
WITH NEW CIRCUIT, LED TURNS ON WHEN PIN 3 IS LOW!

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

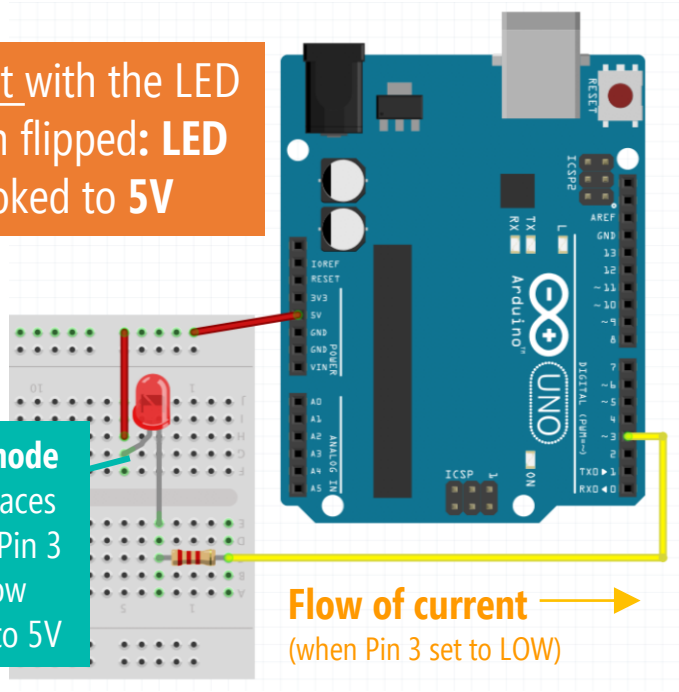


The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND

Flow of current (when Pin 3 set to HIGH)

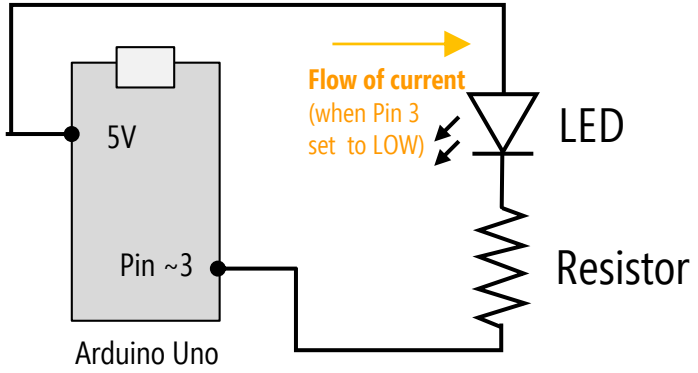


New circuit with the LED orientation flipped**: LED anode** hooked to **5V**



The **LED Anode** (long leg) faces away from Pin 3 and is now connected to 5V

Flow of current (when Pin 3 set to LOW)

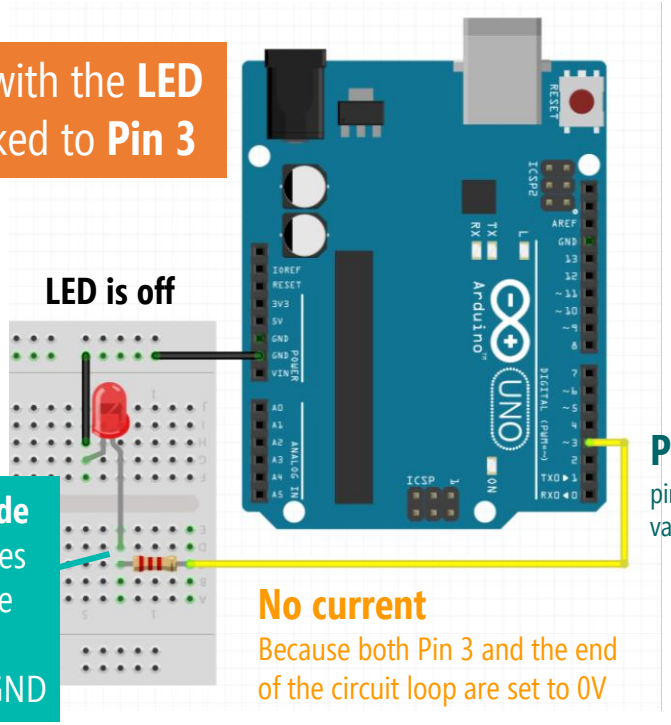


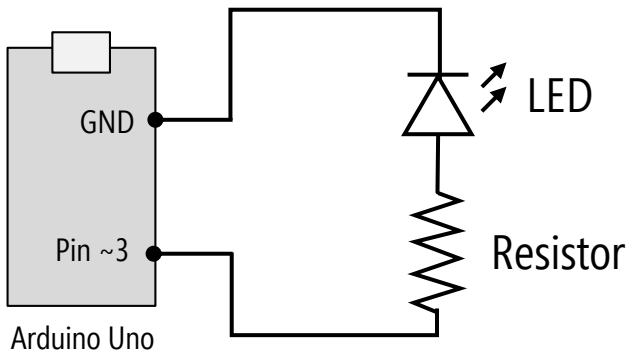
In this configuration, what happens when Pin 3 is set to LOW?

Answer:
The LED turns on!

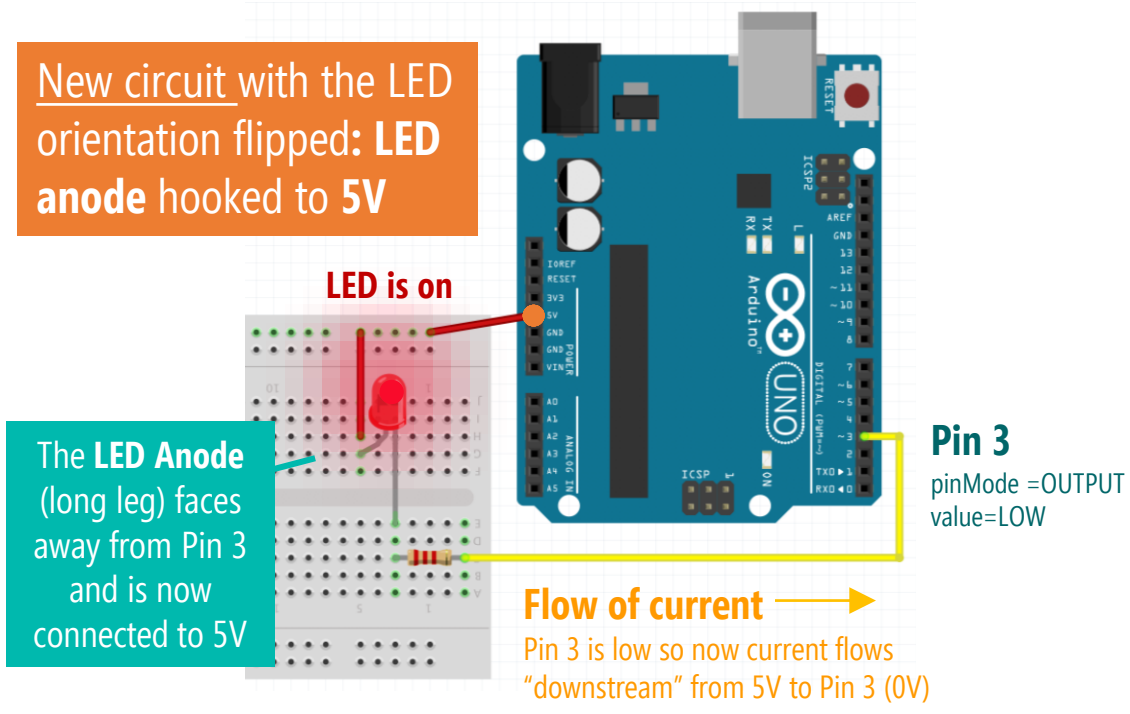
HERE'S WHAT HAPPENS WHEN PIN 3 IS LOW (0V)

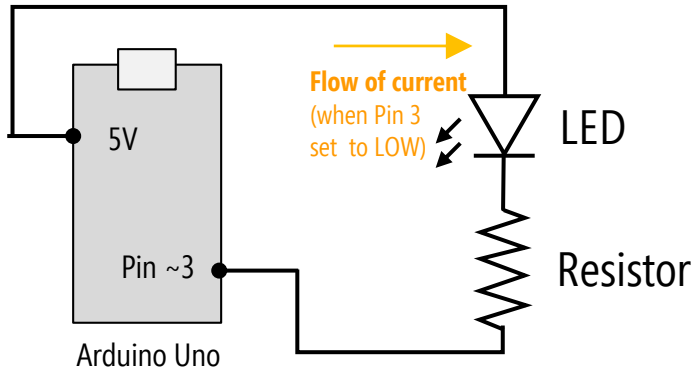
Old circuit with the **LED anode** hooked to **Pin 3**





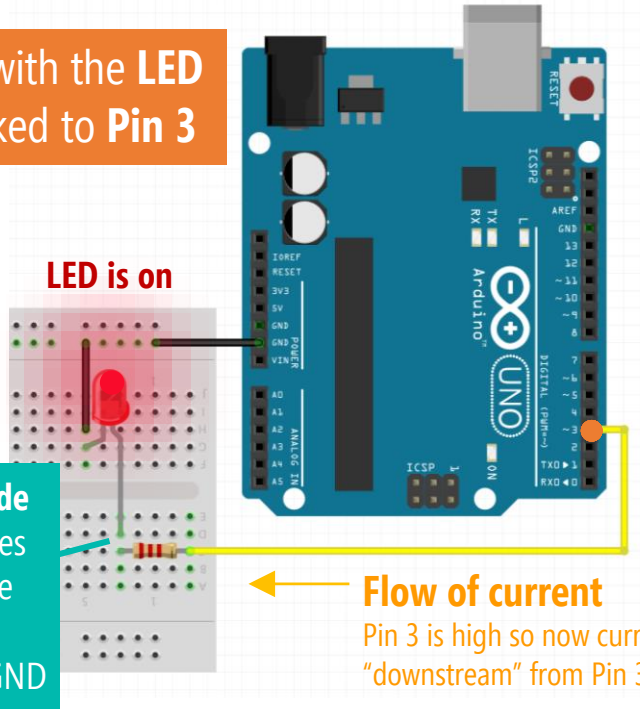
New circuit with the LED orientation flipped: **LED anode** hooked to **5V**





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

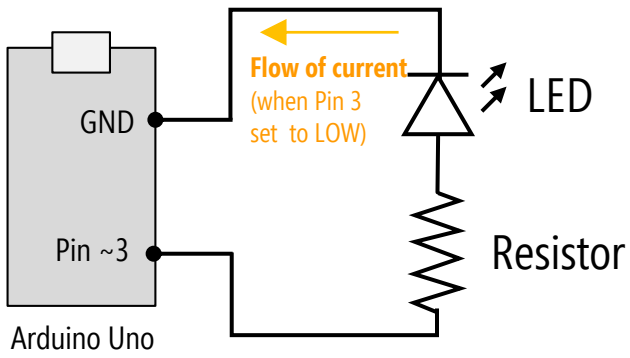
Old circuit with the **LED anode** hooked to **Pin 3**



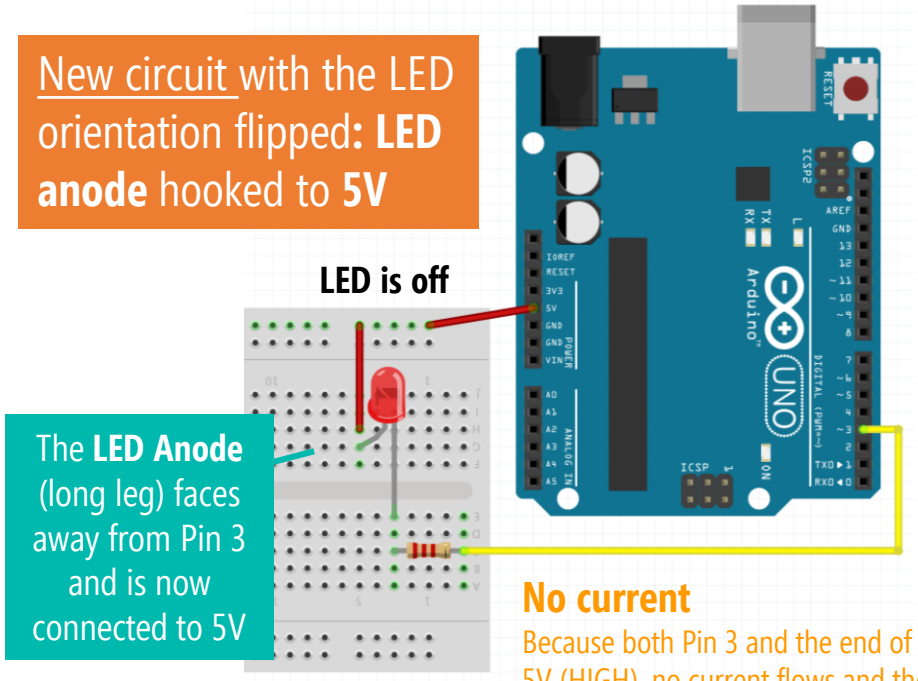
The **LED Anode** (long leg) faces Pin 3 and the cathode is connected to GND

Pin 3
pinMode =OUTPUT
value=HIGH

Flow of current
Pin 3 is high so now current flows "downstream" from Pin 3 (5V) to GND (0V)



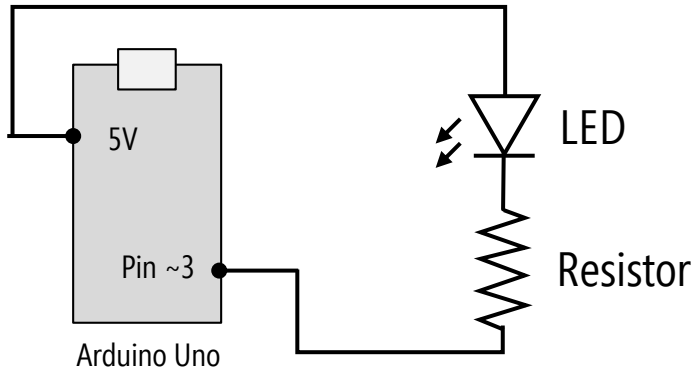
New circuit with the LED orientation flipped: **LED anode** hooked to **5V**



The **LED Anode** (long leg) faces away from Pin 3 and is now connected to 5V

Pin 3
pinMode =OUTPUT
value=HIGH

No current
Because both Pin 3 and the end of the circuit loop are set to 5V (HIGH), no current flows and the LED is thus off

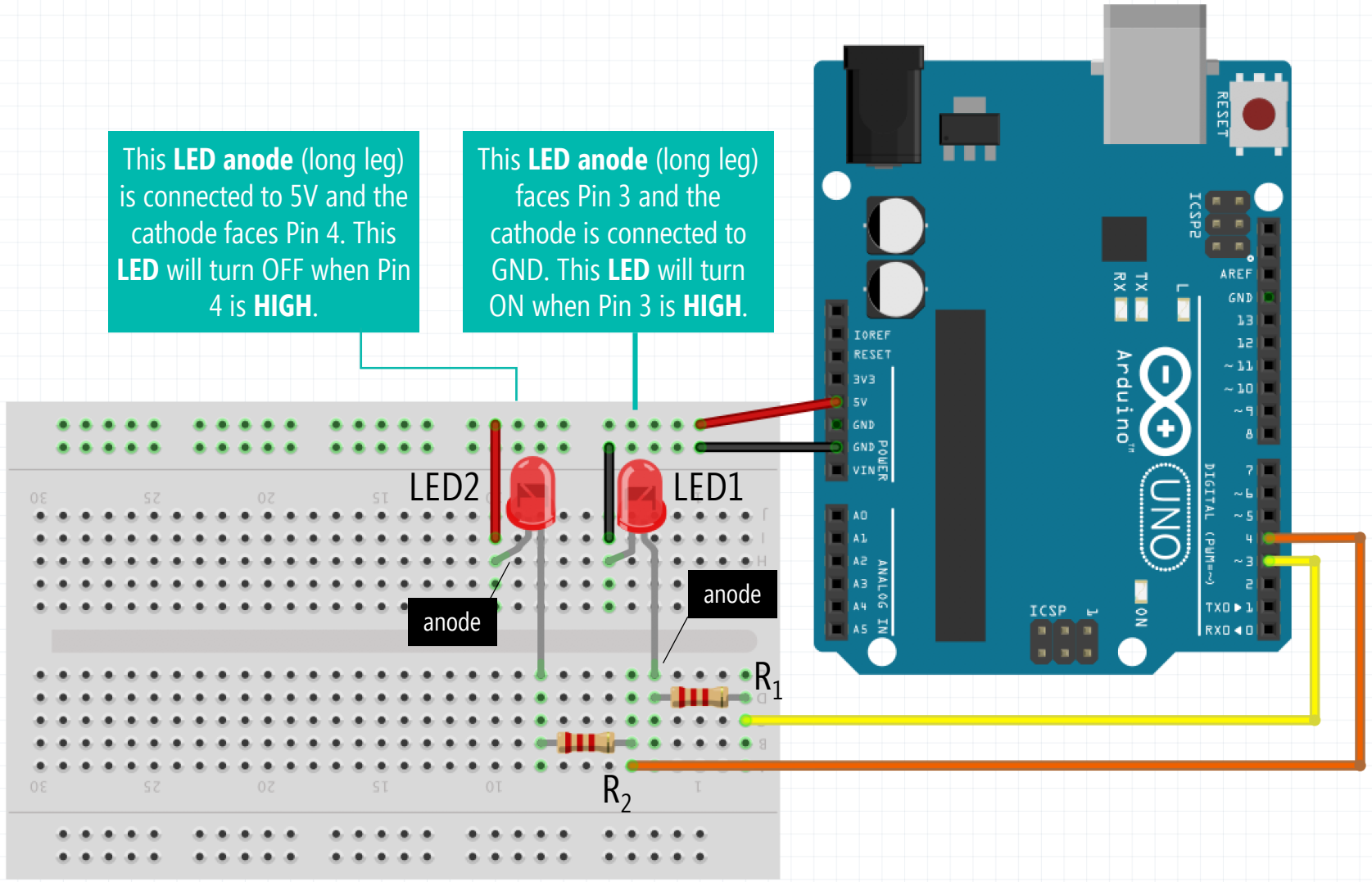


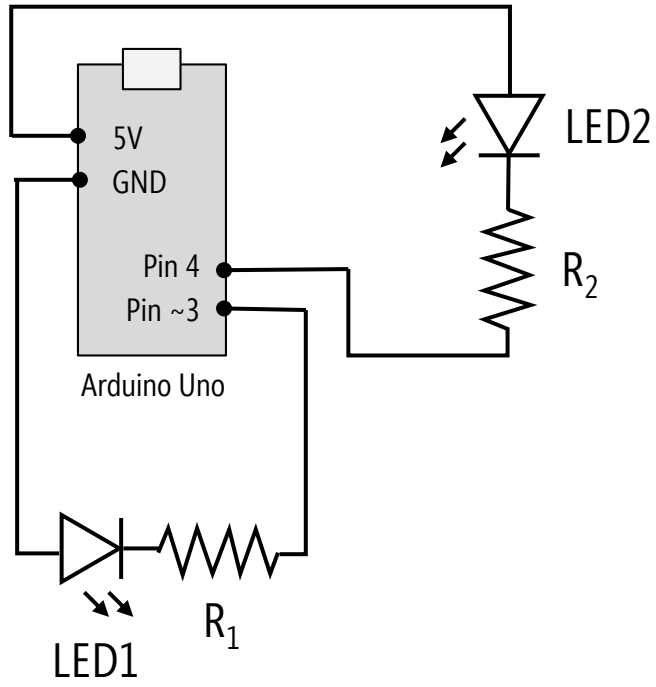
DIGITAL OUTPUT

LET'S WIRE UP BOTH CONFIGURATIONS AT THE SAME TIME

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**.

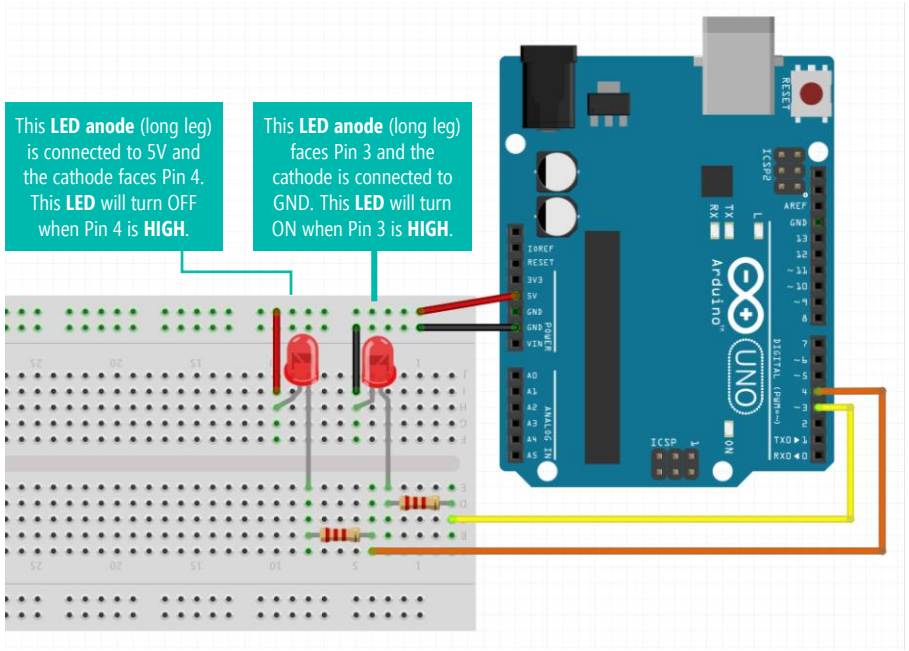
This **LED anode** (long leg) faces Pin 3 and the cathode is connected to GND. This **LED** will turn ON when Pin 3 is **HIGH**.



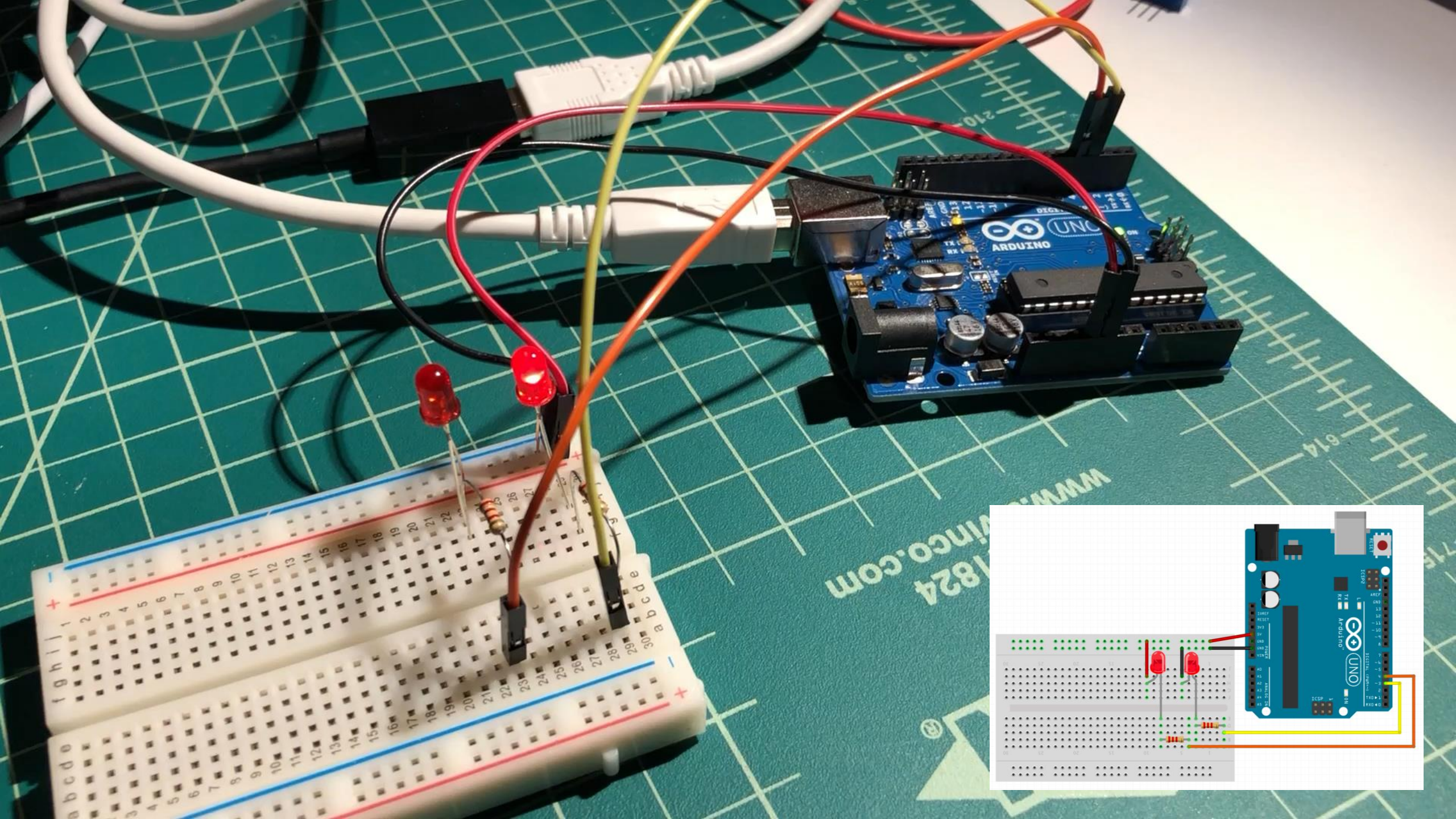


DIGITAL OUTPUT

AND HERE'S THE CODE




```
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
}
```


ARDUINO
UNO

ARDUINO

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 upd
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);
  }
}
```

Whew, that was a **rapid introduction** to Arduino…

But now we have to **debug** both **software** & **hardware**, how?!

But now we have to **debug** both **software** & **hardware**, how?!

THIS IS HARD!

DEBUGGING ARDUINO: SOME PATHS FORWARD

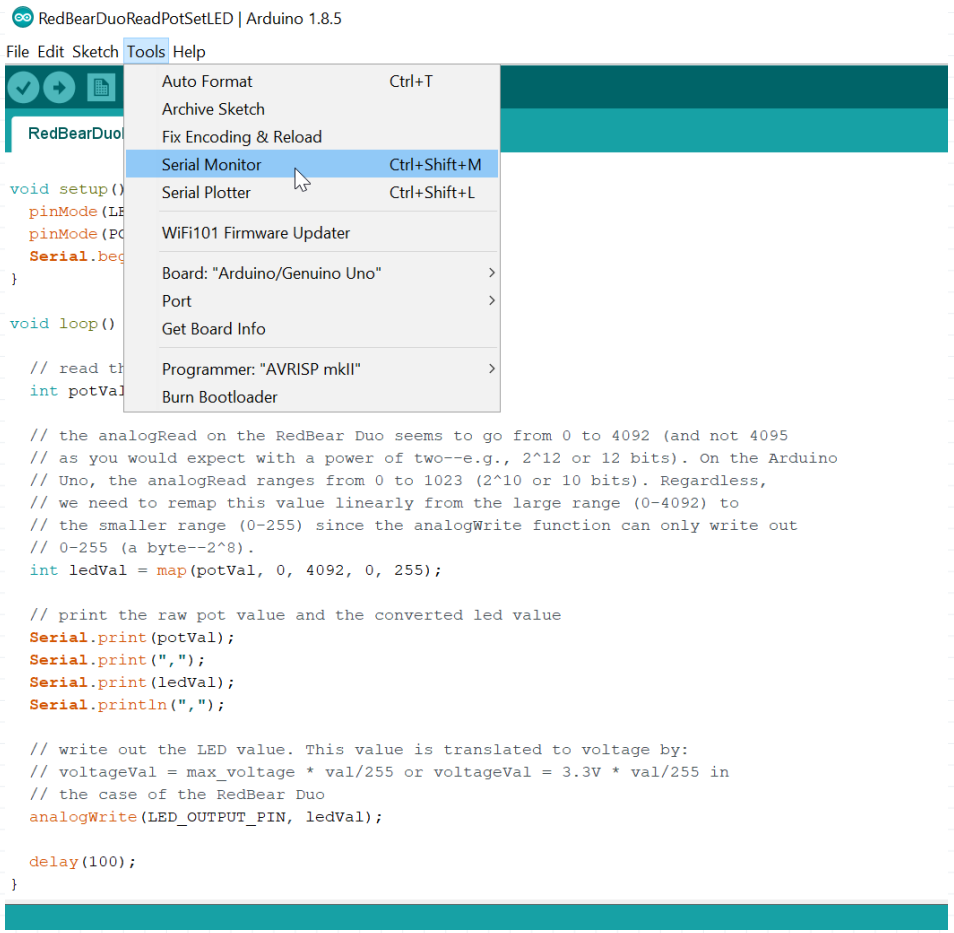
Modularize.

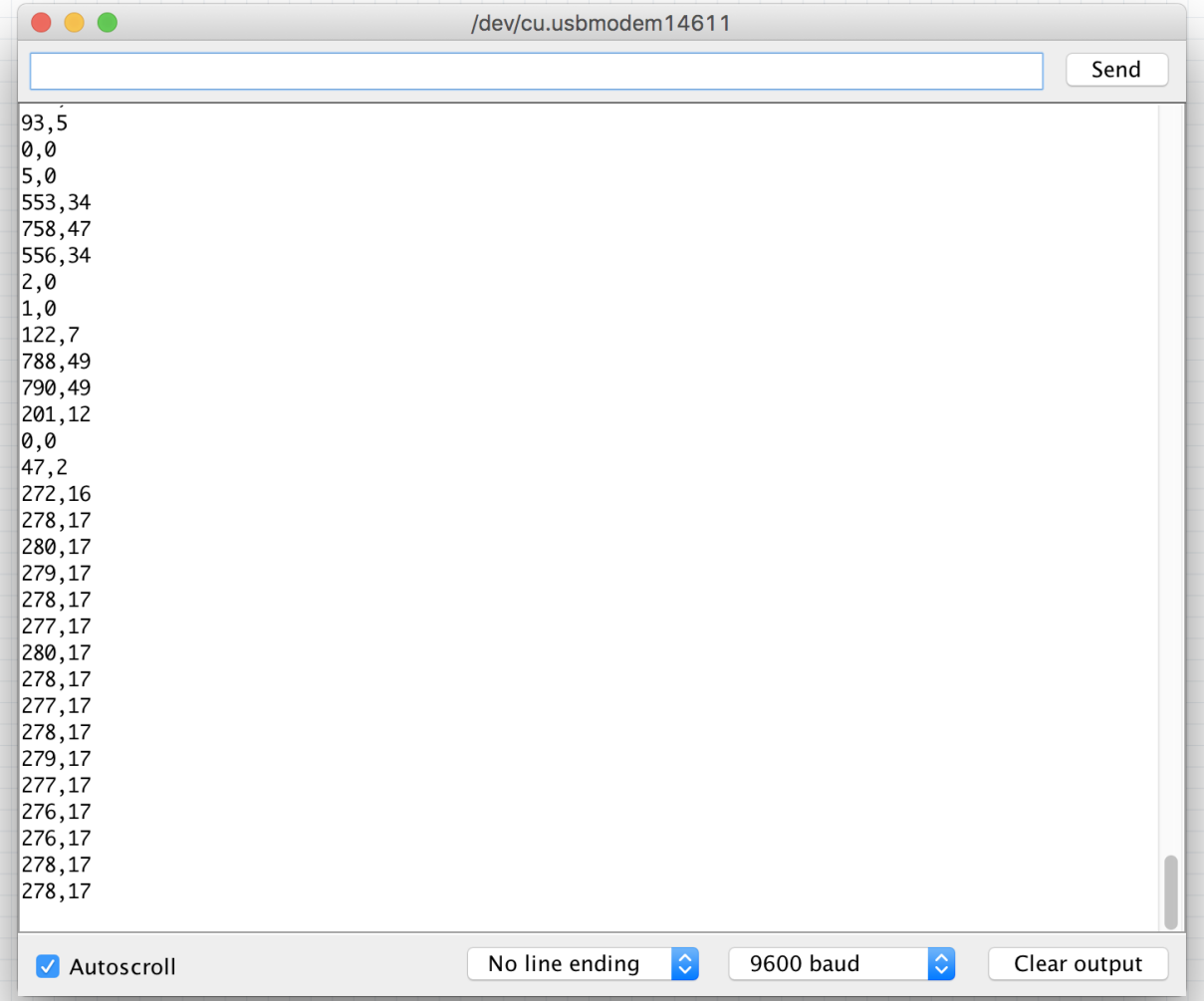
Use Serial.println -> Serial Monitor and Serial Plotter

Use a multimeter

SERIAL MONITOR

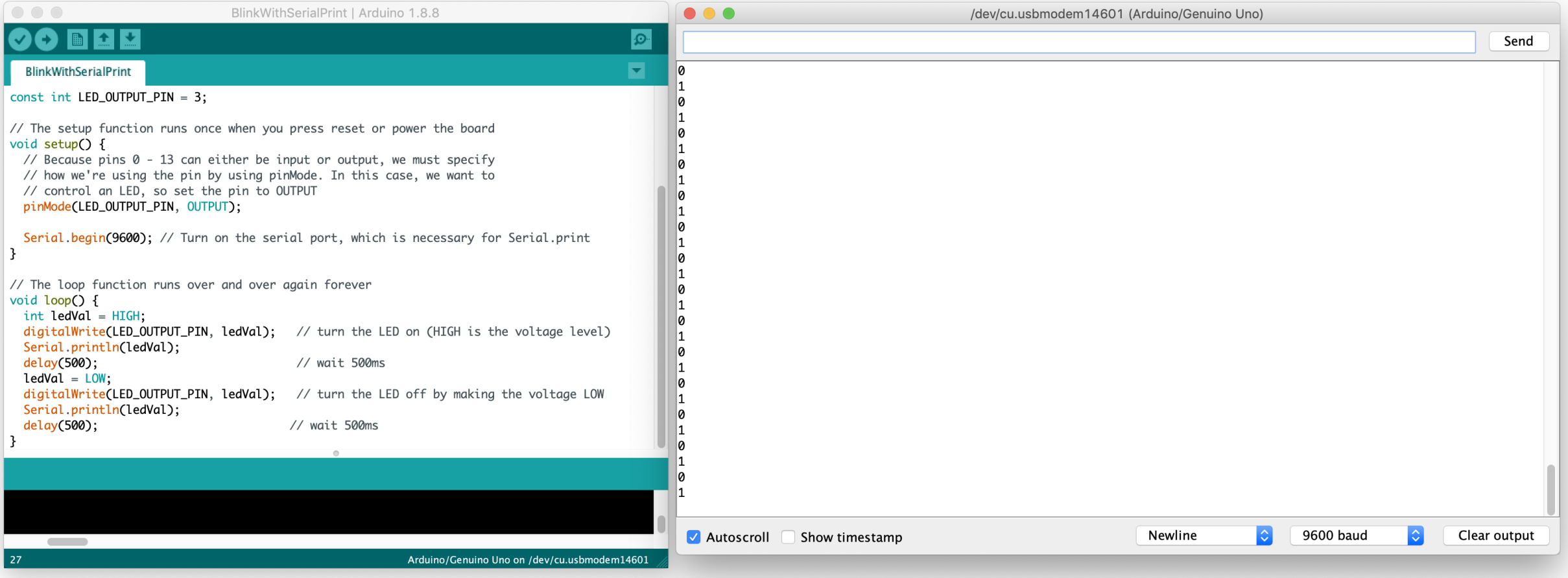
SERIAL MONITOR IS USEFUL FOR DEBUGGING

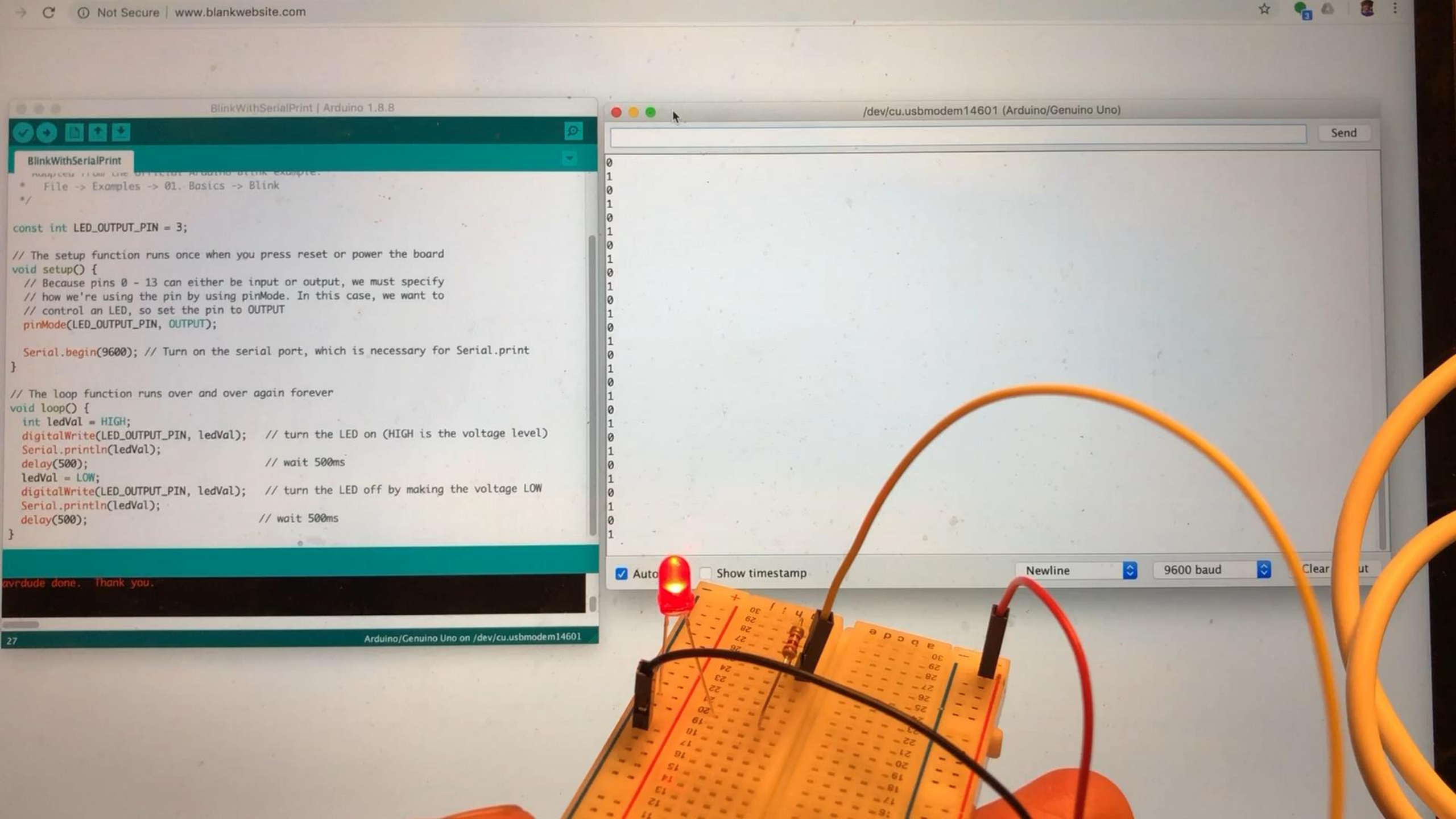




DIGITAL OUTPUT

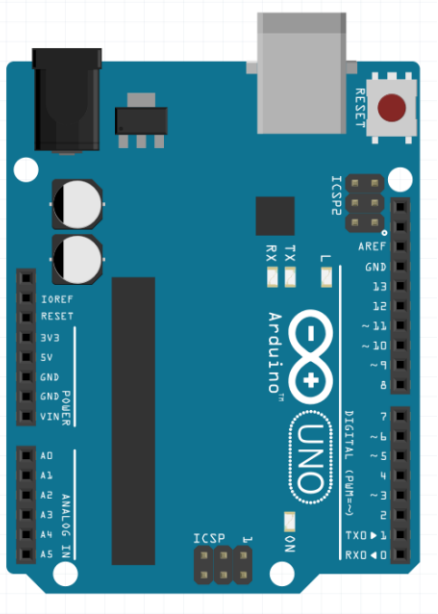
USING SERIAL MONITOR FOR DEBUGGING



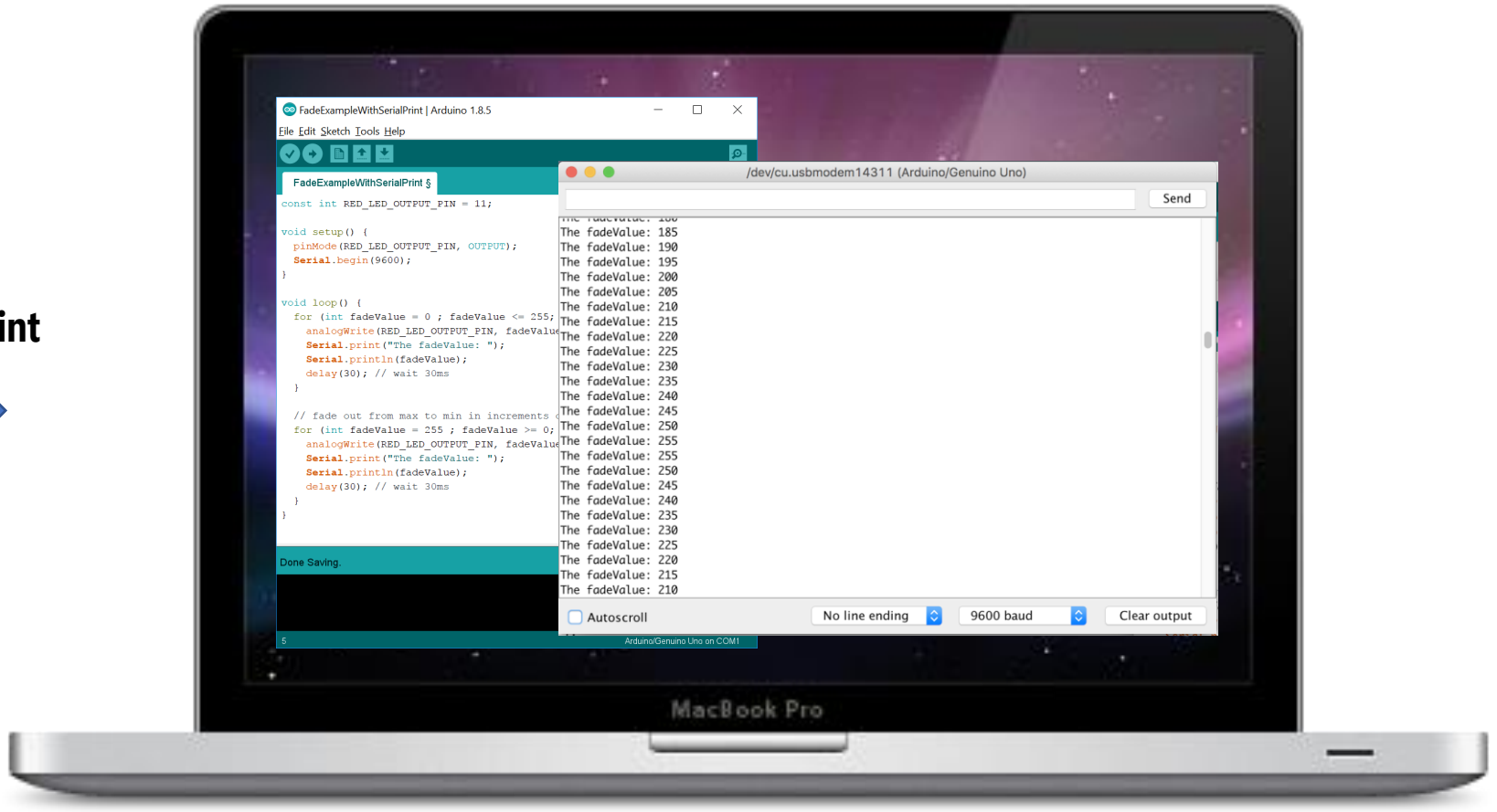

Not Secure | www.blankwebsite.com
BlinkWithSerialPrint | Arduino 1.8.8
BlinkWithSerialPrint
* File -> Examples -> 01. Basics -> Blink
*/
const int LED_OUTPUT_PIN = 3;
// The setup function runs once when you press reset or power the board
void setup() {
// Because pins 0 - 13 can either be input or output, 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);
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
}
avrdude done. Thank you.
27
Arduino/Genuino Uno on /dev/cu.usbmodem14601
/dev/cu.usbmodem14601 (Arduino/Genuino Uno)
Send
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
Auto
Show timestamp
Newline
9600 baud
Clear

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.

PRINTING TO SERIAL

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

https://www.arduino.cc/en/Serial/Print

PRINTING TO SERIAL

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);
```

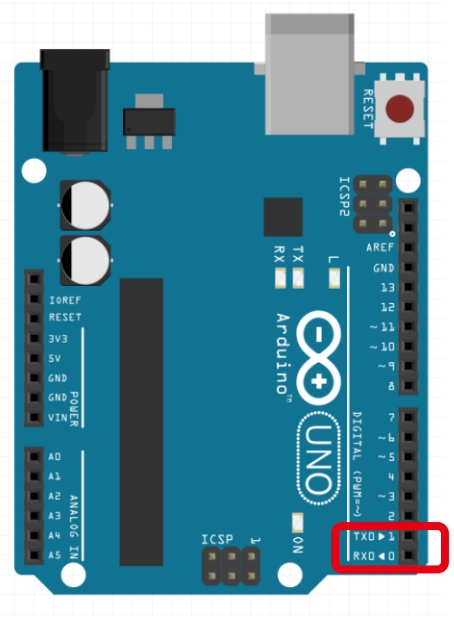
Syntax

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

Parameters

`val`: the value to print

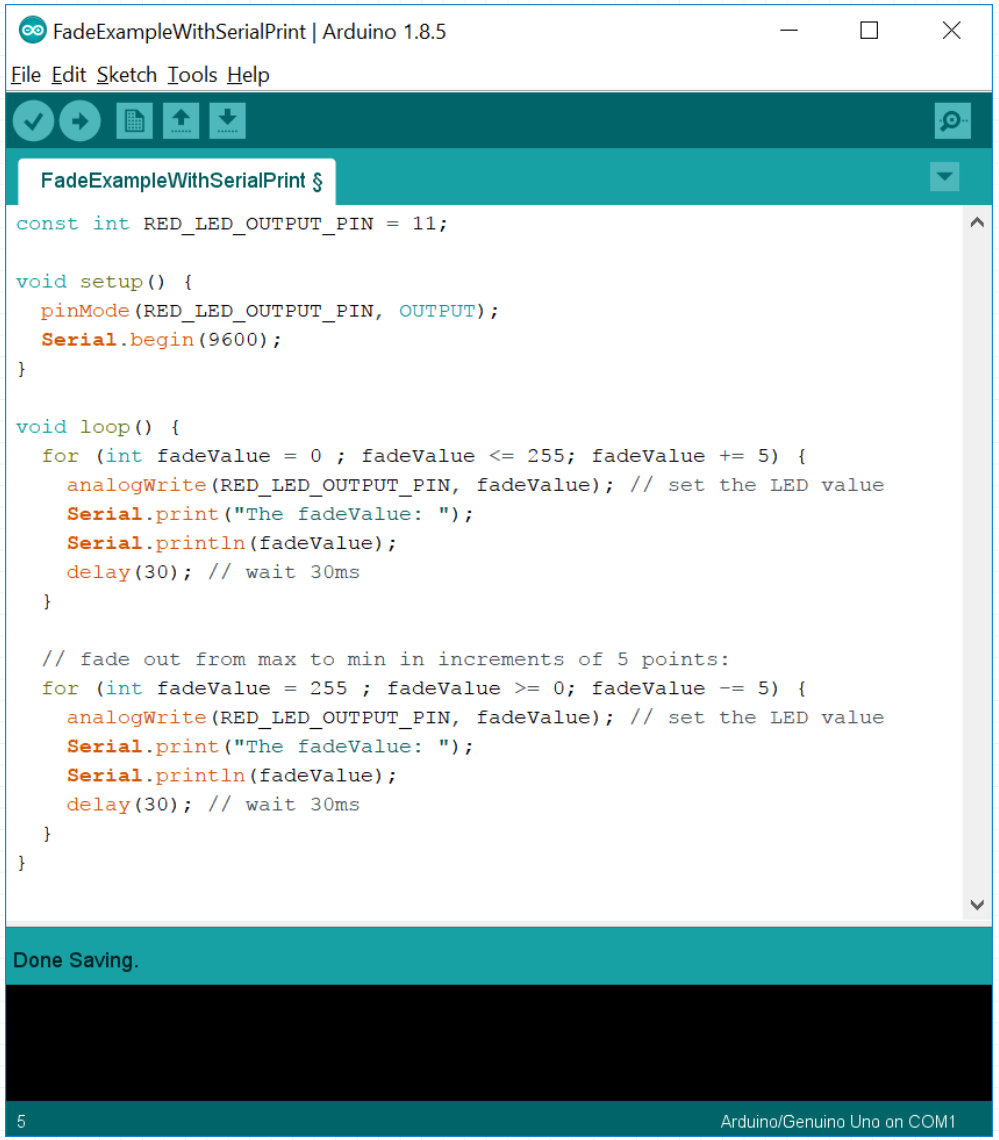
`format`: specifies formatting, see documentation

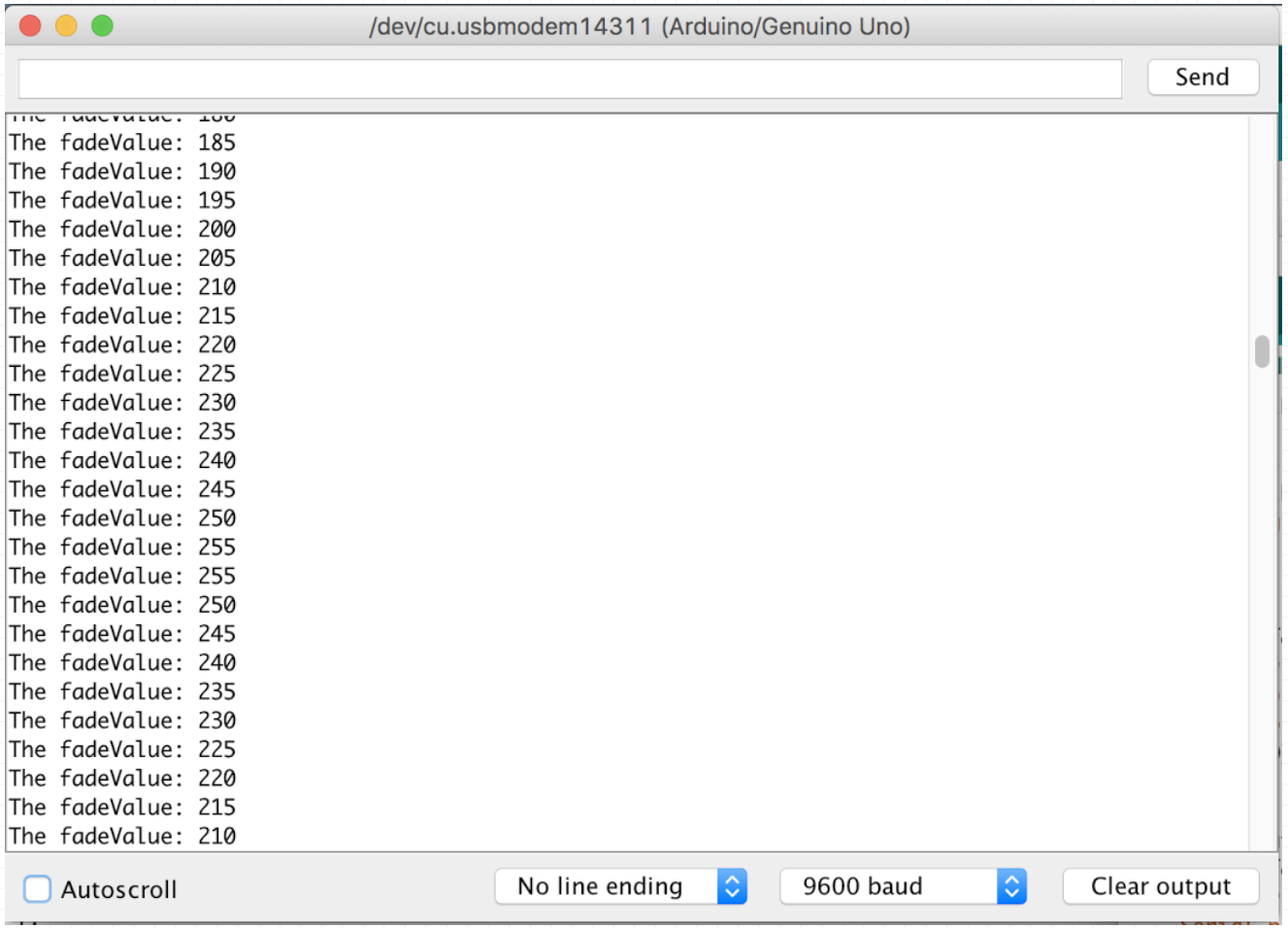


As soon as you activate Serial.begin, **Pins 0 and 1** are used for serial communication and cannot be used for anything else!

https://www.arduino.cc/en/Serial/Print

SERIAL MONITOR
TRY IT YOURSELF!





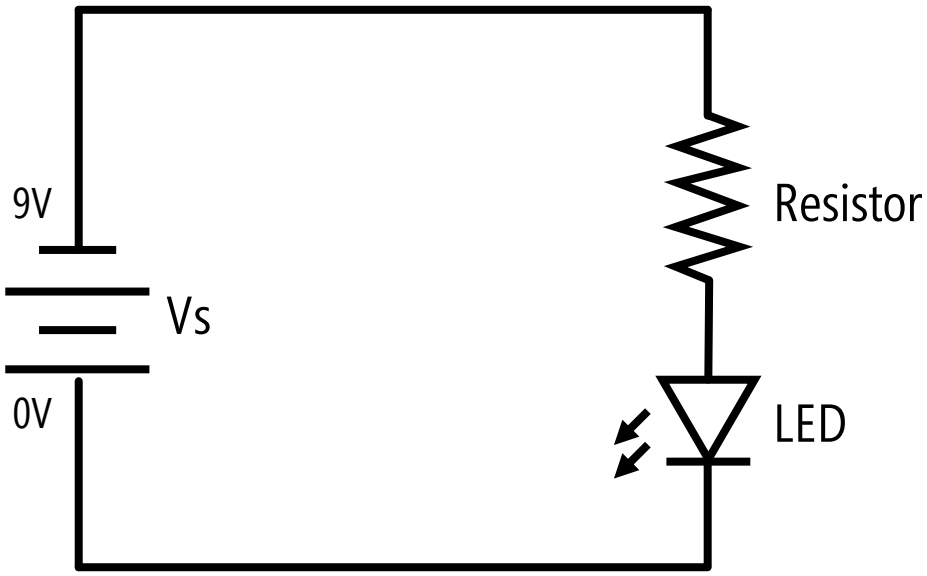
DEBUGGING ARDUINO

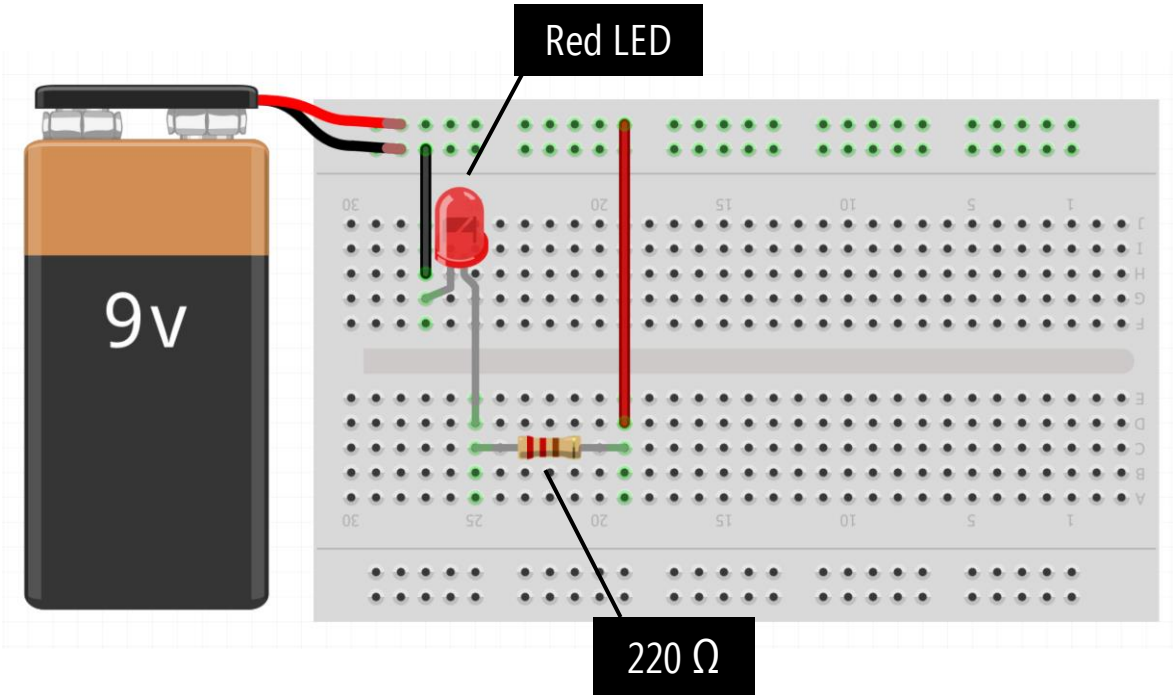
Modularize.

Use Serial.println -> Serial Monitor and Serial Plotter

Use a multimeter

EXERCISE: HOW MUCH CURRENT THROUGH LED?

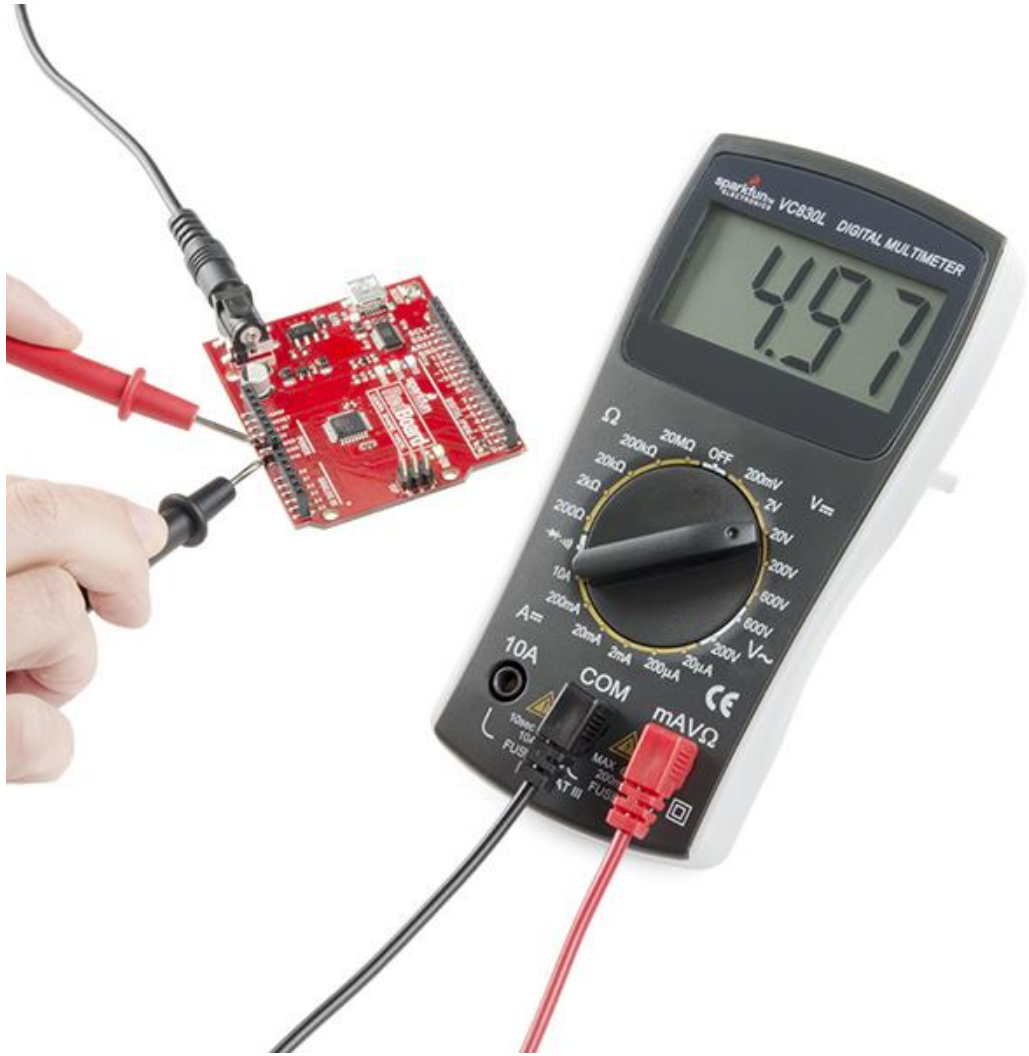




Let's measure it empirically using a multimeter… but first, an introduction

MULTIMETERS

YOU CAN MEASURE CIRCUITS WITH A MULTIMETER



Resistance

Voltage

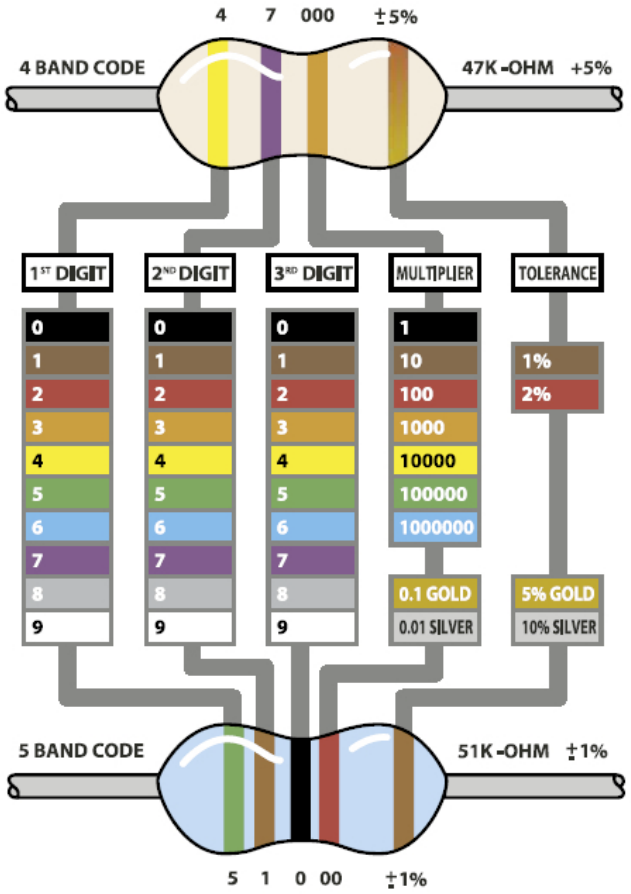
Current

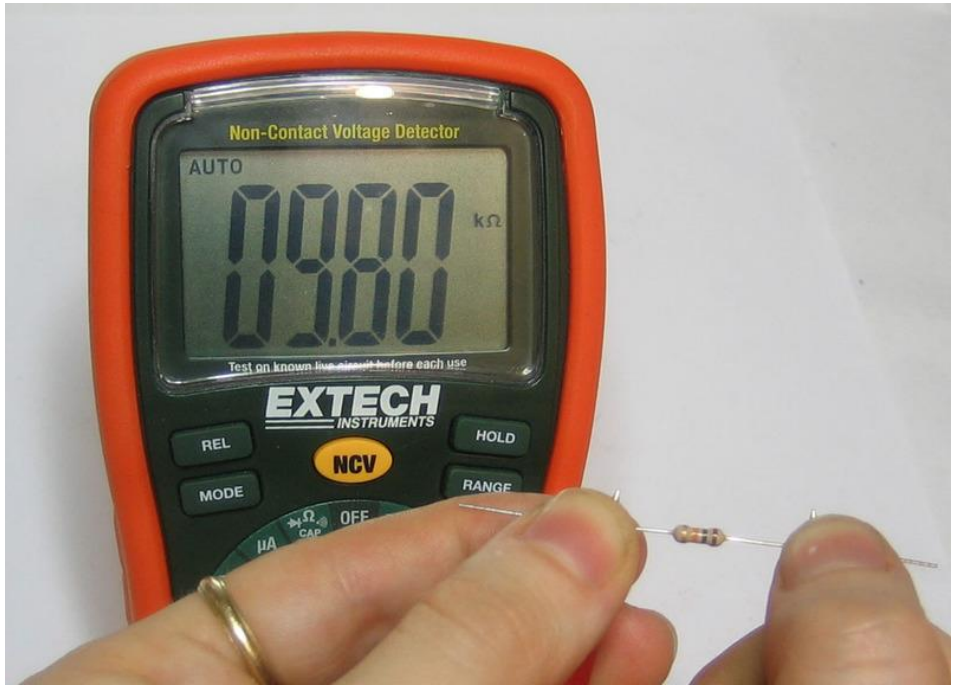
Continuity (short testing)

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

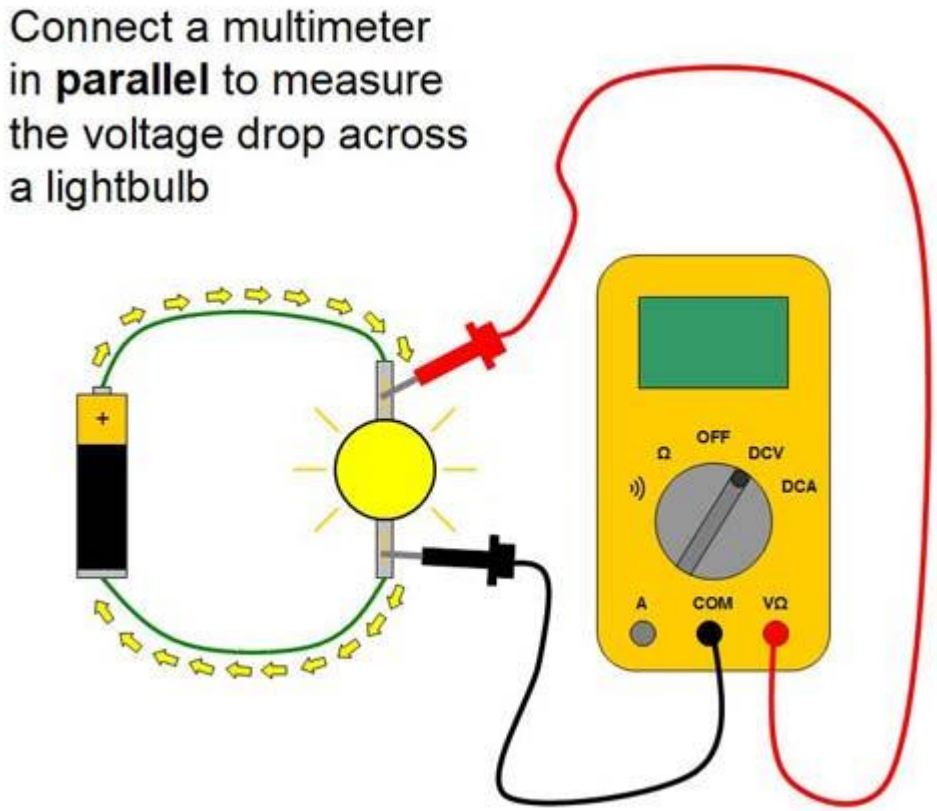
Source: https://learn.adafruit.com/multimeters/resistance

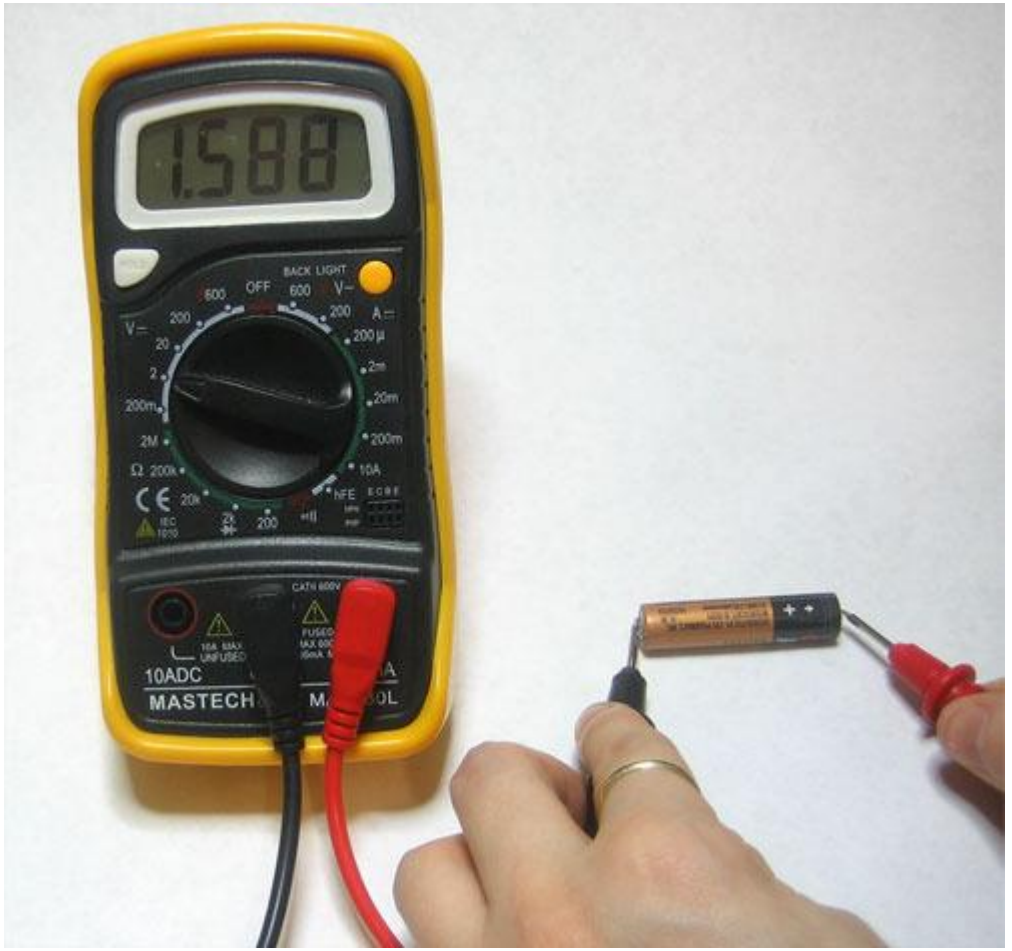
MULTIMETERS

MEASURING RESISTANCE

MULTIMETERS

MEASURING VOLTAGE





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.

Source: https://learn.adafruit.com/multimeters/resistance; http://www.sciencebuddies.org/science-fair-projects/multimeters-tutorial.shtml

MULTIMETERS

MEASURING VOLTAGE: SWITCH MULTIMETER TO V

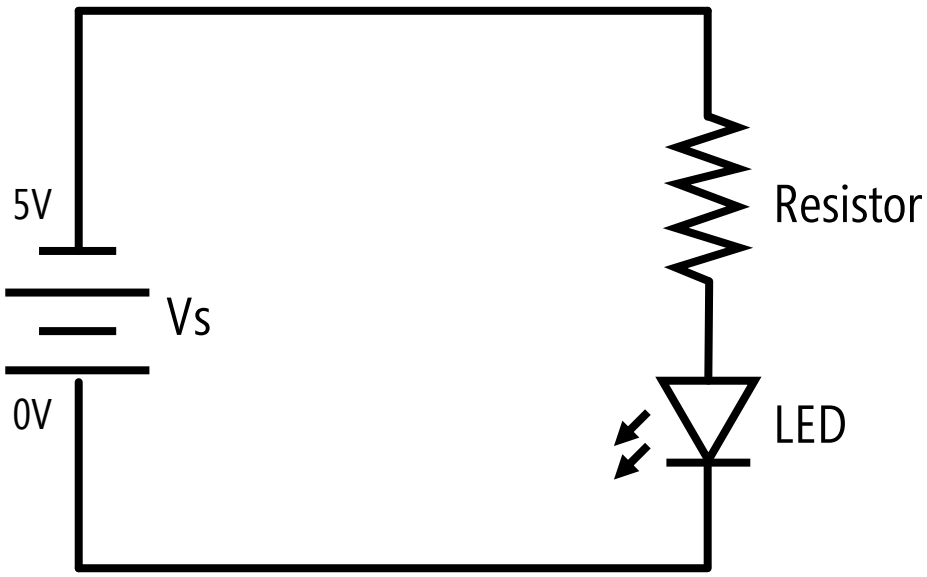
Figure 1-54. *Each meter has a different way to measure volts DC. The manually adjusted meter (top) requires you to move a slider switch to "DC" and then choose the highest voltage you want to measure: In this case, the selected voltage is 20 (because 2 would be too low). Using the autoranging RadioShack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

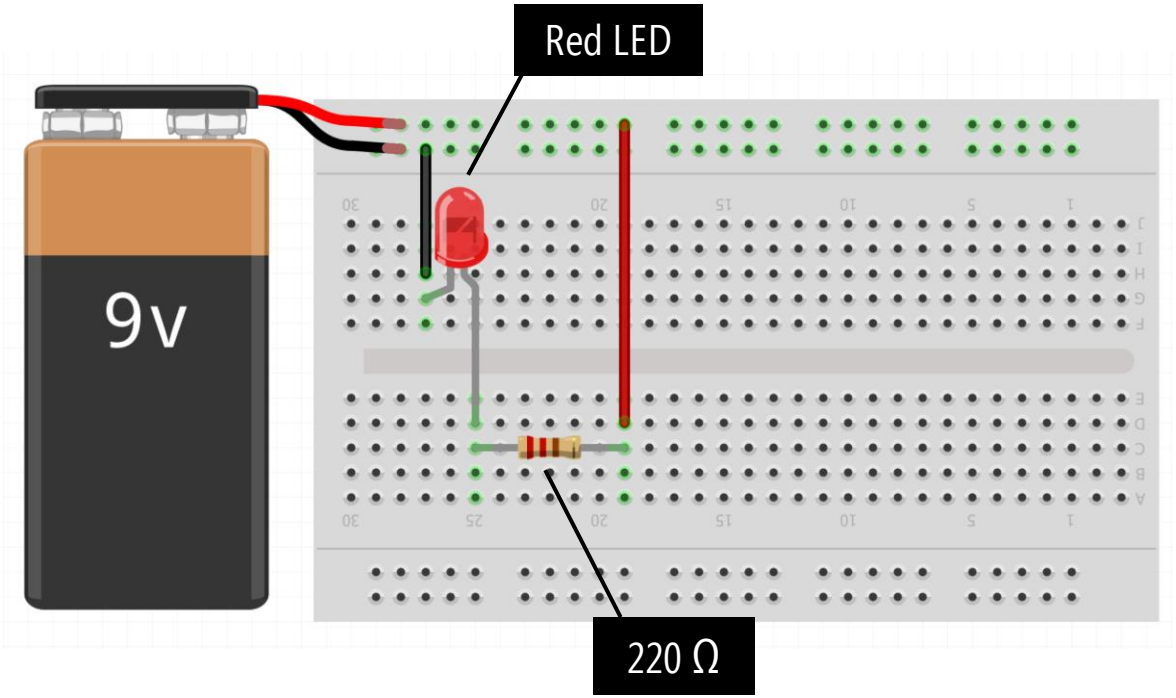
Figure 1-53

Figure 1-52

[source: Platt, Make: Electronics, 1st Edition]

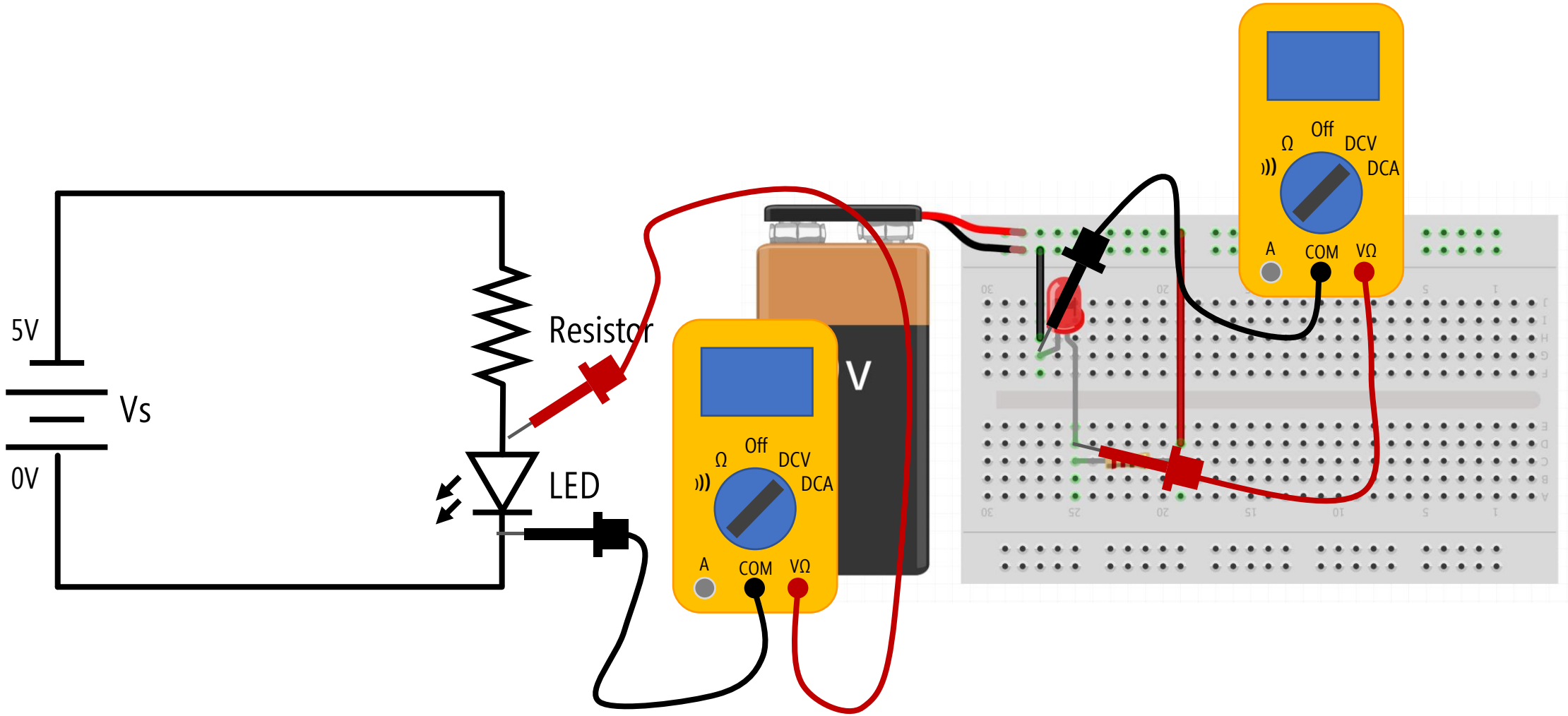
EXERCISE: WHAT'S THE VOLTAGE DROP ACROSS THE LED?





MULTIMETERS

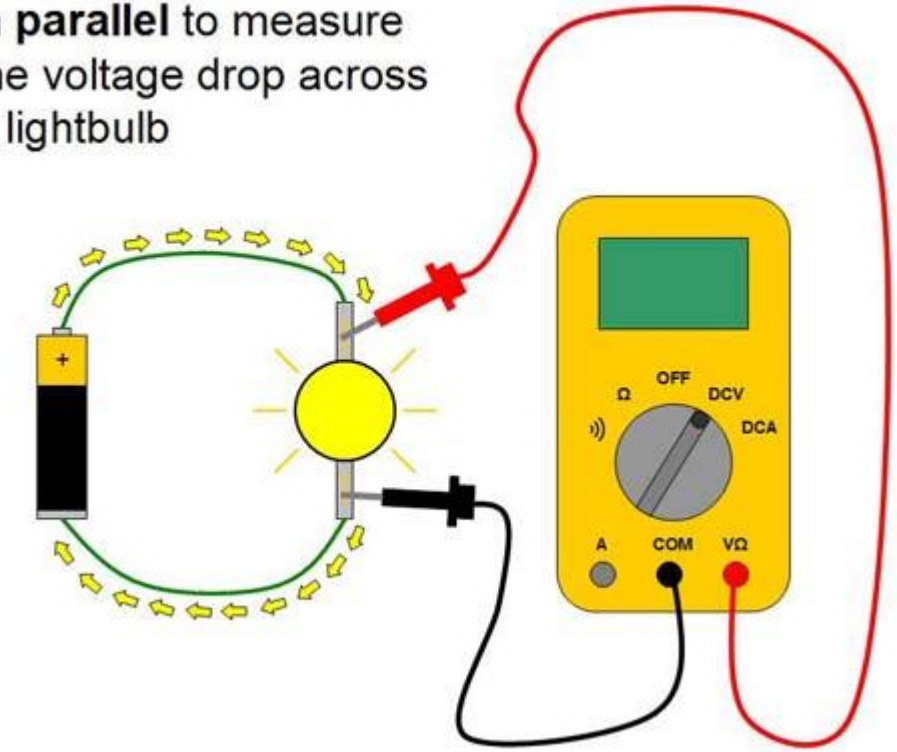
EXERCISE: WHAT'S THE VOLTAGE DROP ACROSS THE LED?



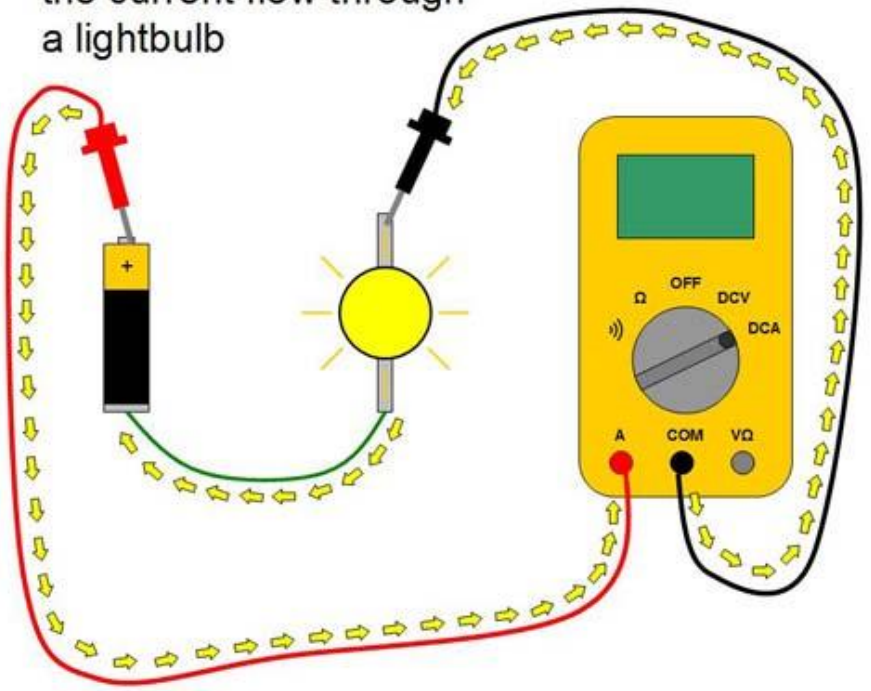
MULTIMETERS

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

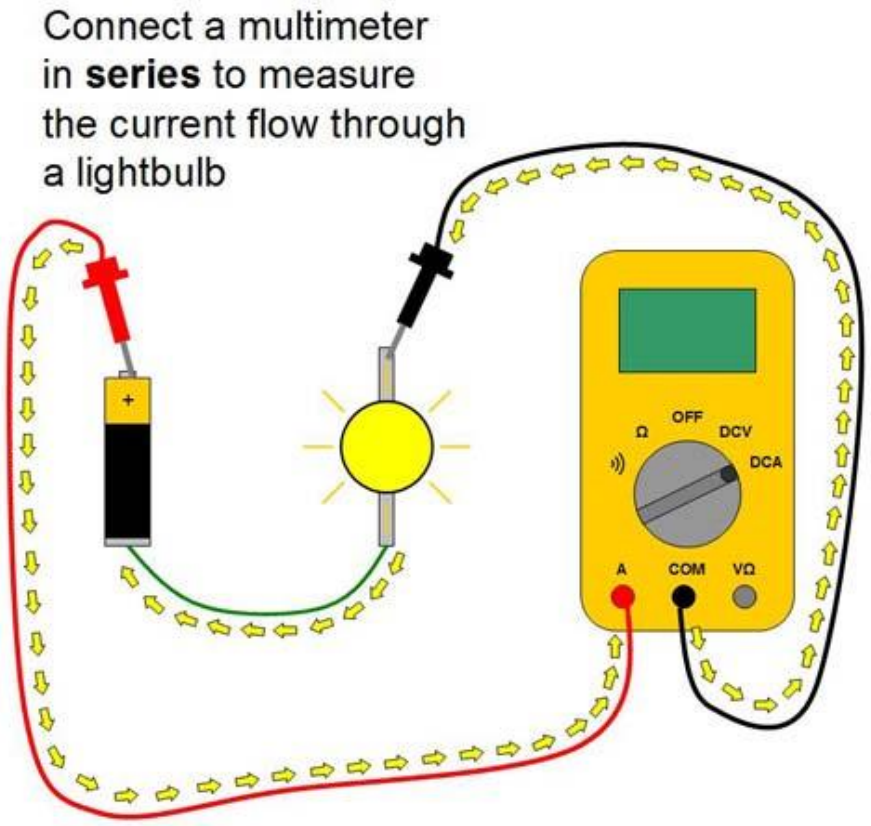


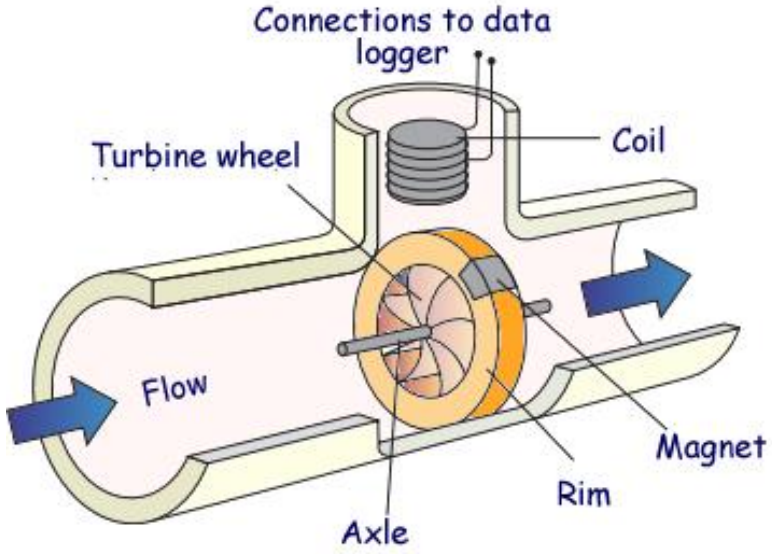
Source: https://learn.adafruit.com/multimeters/resistance; http://www.sciencebuddies.org/science-fair-projects/multimeters-tutorial.shtml

MULTIMETERS

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!





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 milliamps and remember that the meter displays numbers that mean thousandths of an amp.*

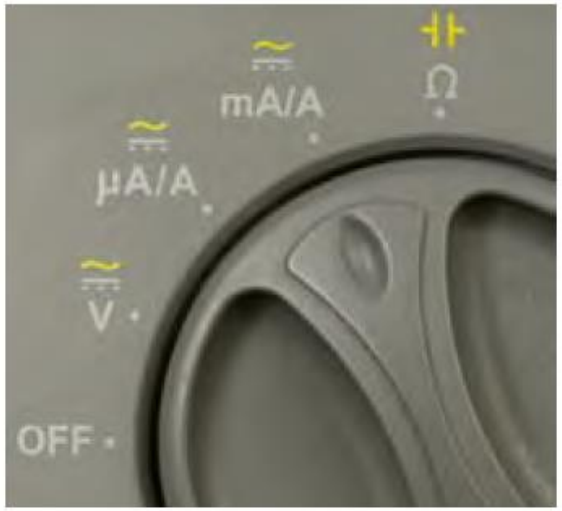
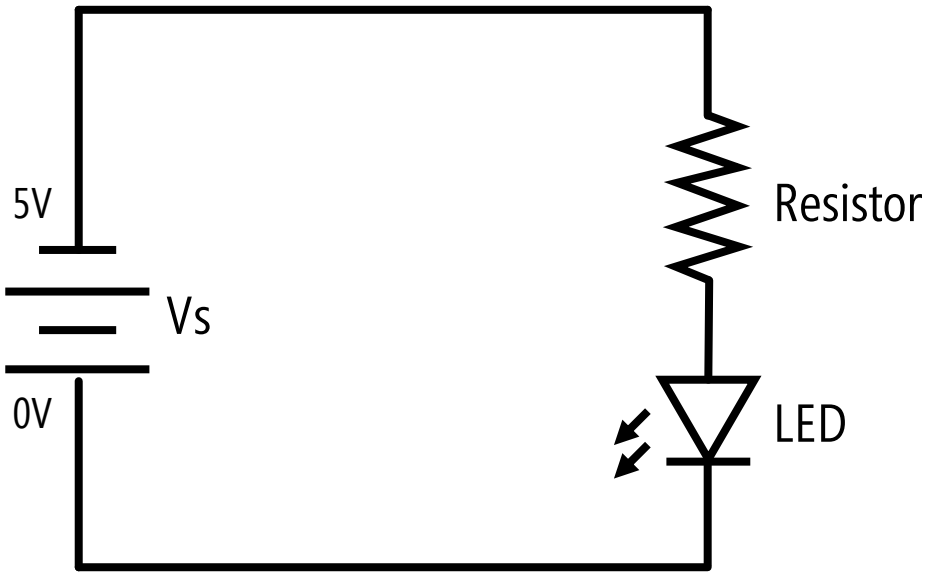
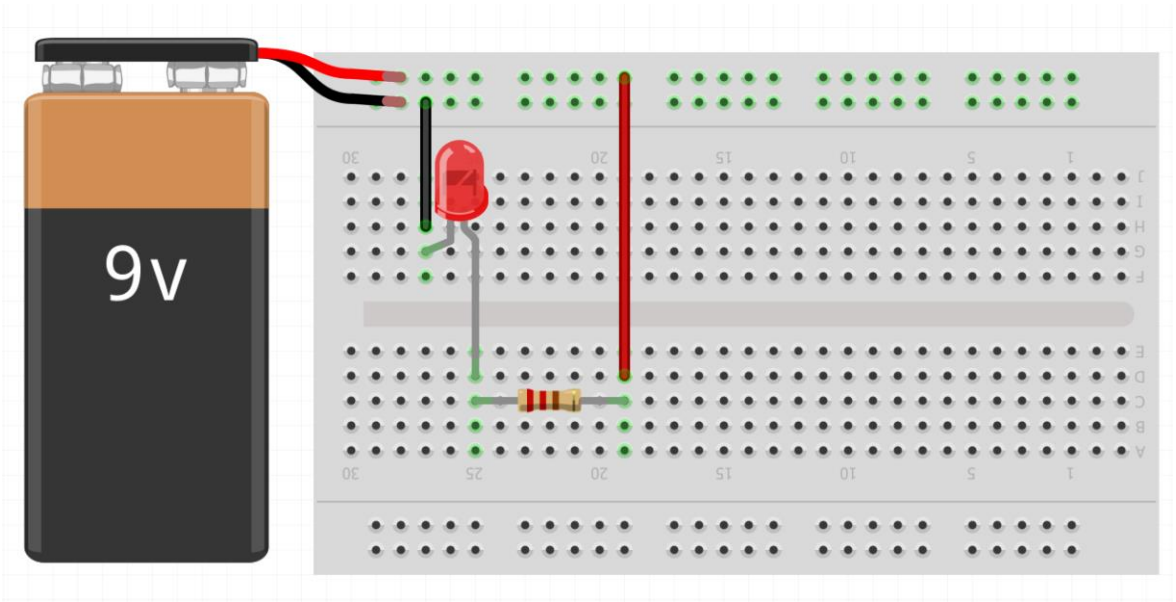


Figure 1-59

EXERCISE: HOW MUCH CURRENT THROUGH LED?

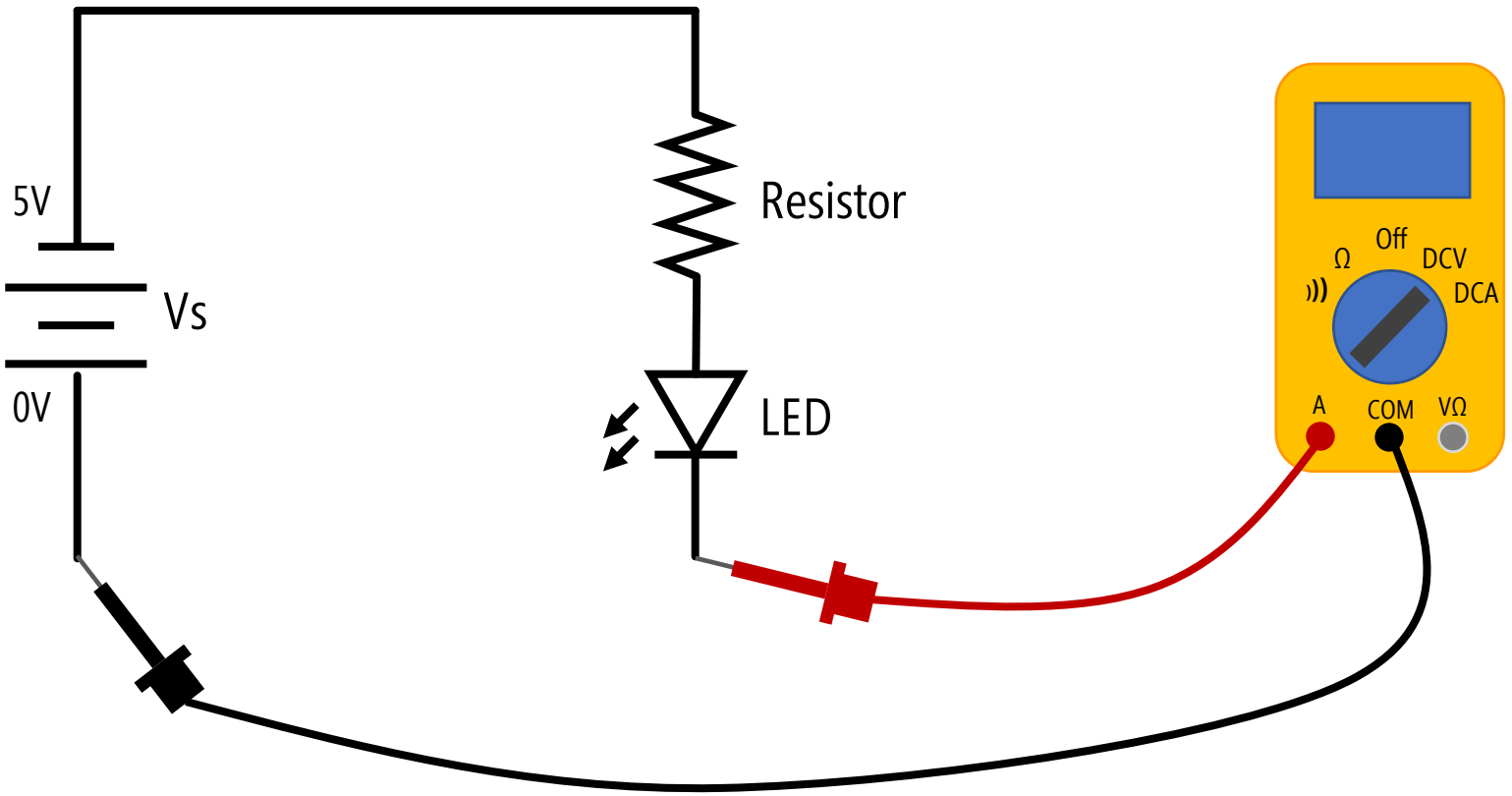
You'll have to slightly rewire this circuit so that the current passes through the multimeter





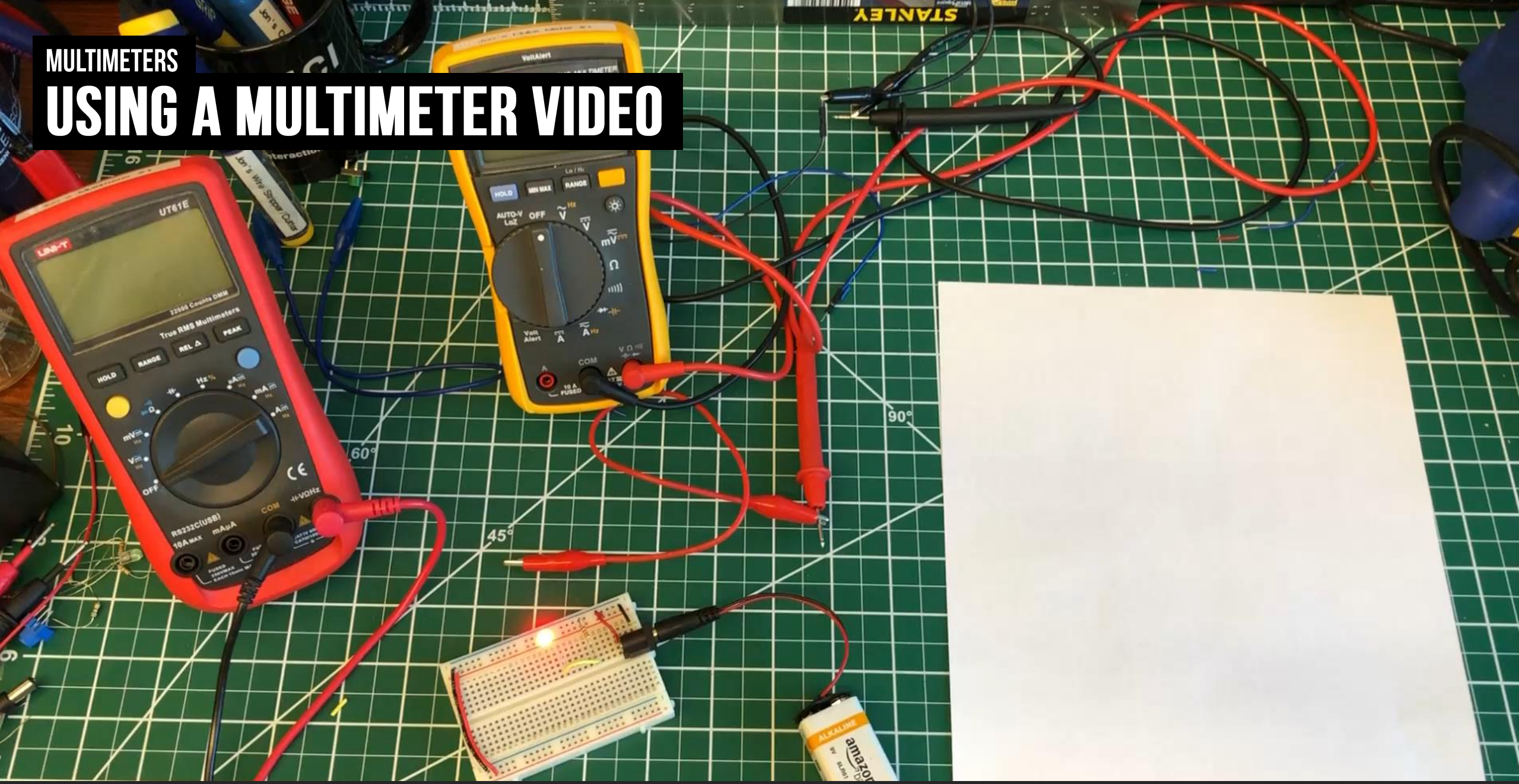
EXERCISE: HOW MUCH CURRENT THROUGH LED?

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



MULTIMETERS

USING A MULTIMETER VIDEO



Video by Jon Froehlich

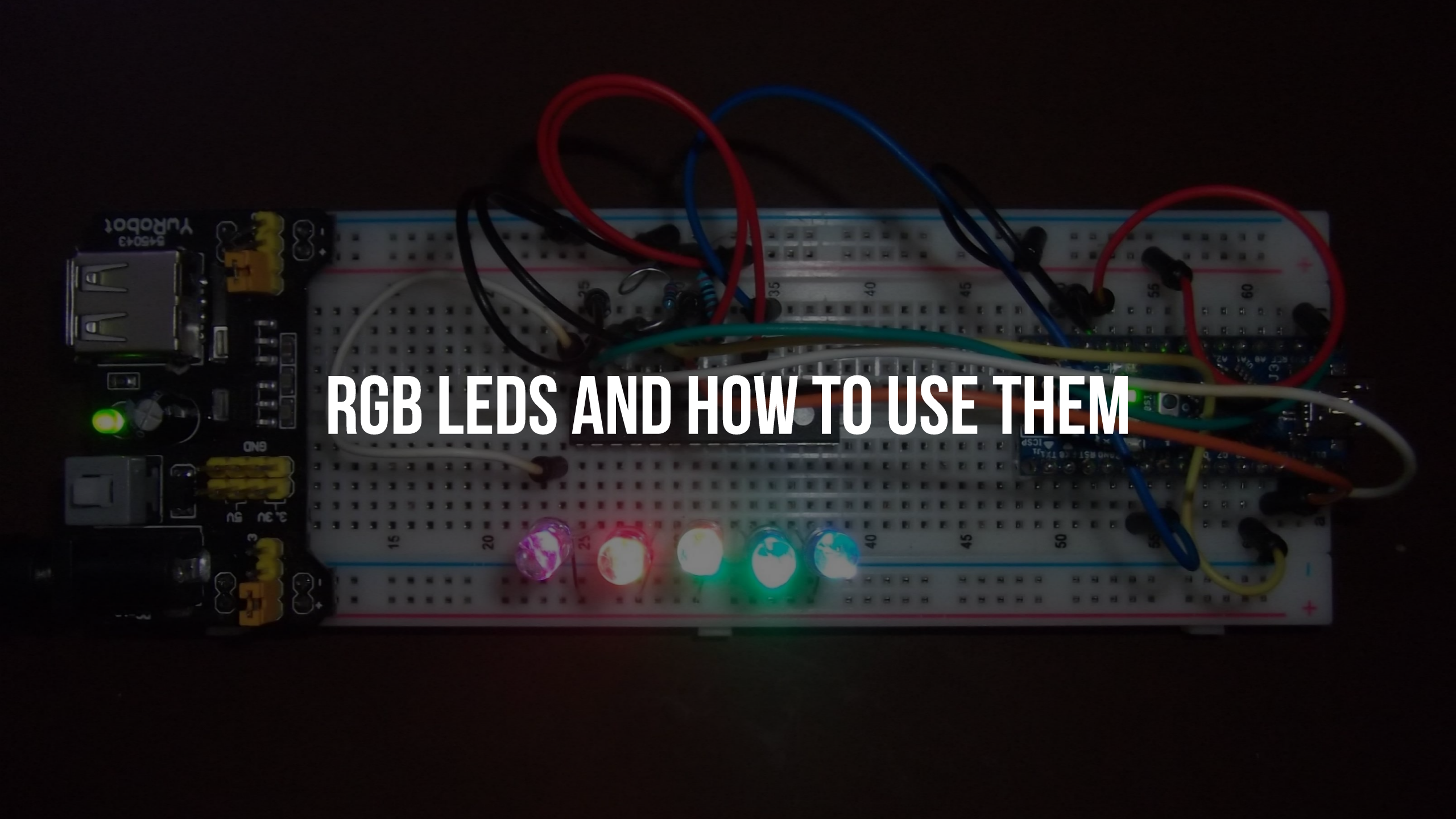
RESOURCES

HELPFUL MULTIMETER RESOURCES

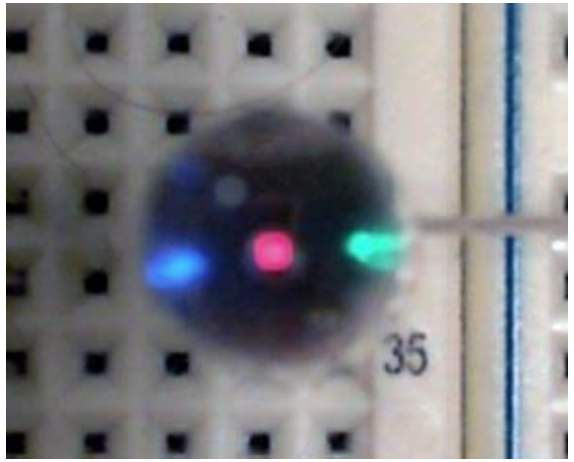
Adafruit's Multimeter Tutorial
Adafruit online tutorial series

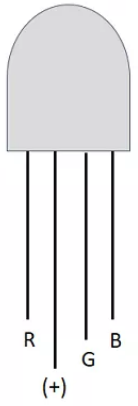
Collin's Lab: Multimeters Video
Collin Cunningham, noted tutorialist

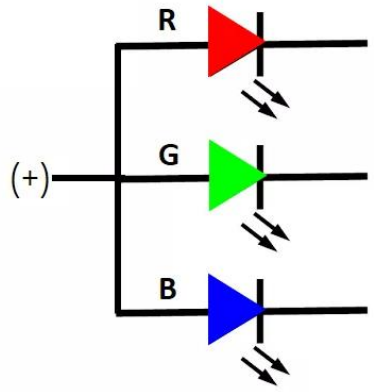
Make Magazine's How to Use a Multimeter
Make Magazine


RGB LEDS AND HOW TO USE THEM

WHAT ARE RGB LEDS







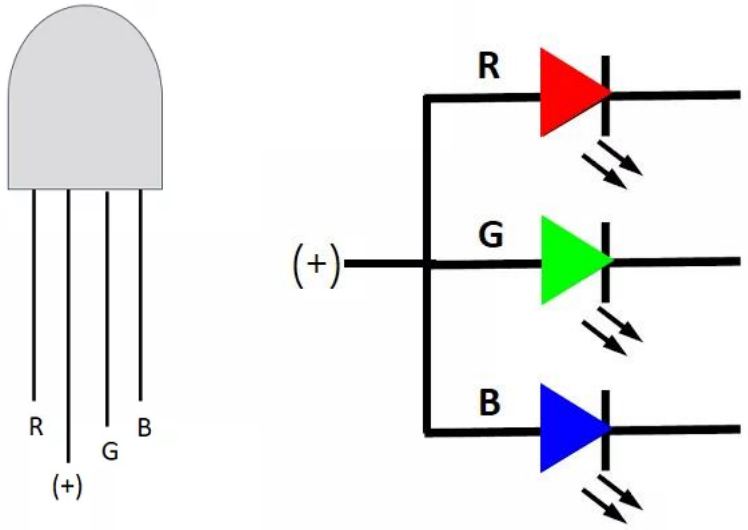
RGB LEDs are actually **three LEDs** in one**:** a **red** LED, a **green** LED, and a **blue** LED

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

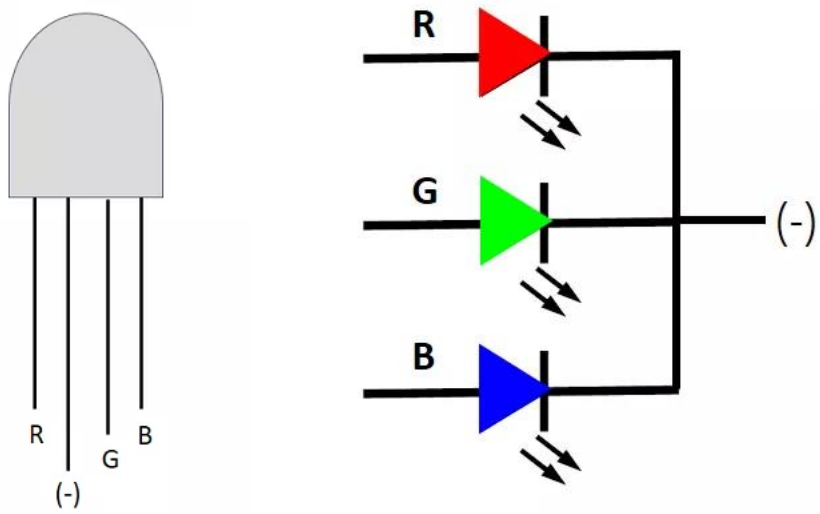
RGB LEDS

COMMON ANODE VS. COMMON CATHODE

Common Anode (+)



Common Cathode (-)



Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

1 ADD TO CART

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

ADD TO WISHLIST

DESCRIPTION

TECHNICAL DETAILS

LEARN

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

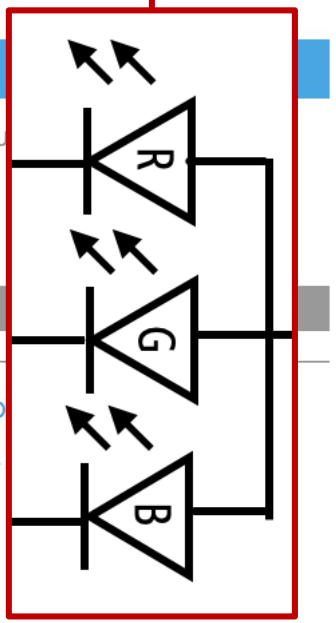
1

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

DESCRIPTIO

TECHNICAL

LEARN



Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

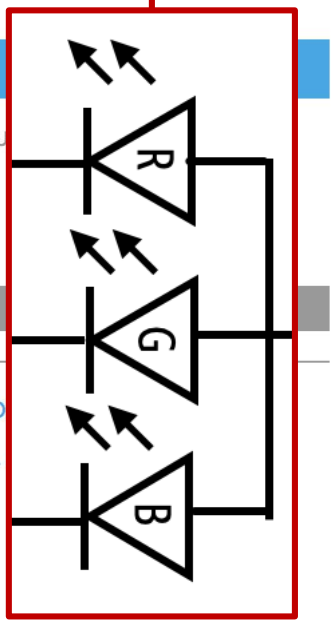
1

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

DESCRIPTIO

TECHNICAL

LEARN



adafruit SHOP BLOG LEARN FORUMS VIDEOS ADABOX

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

Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

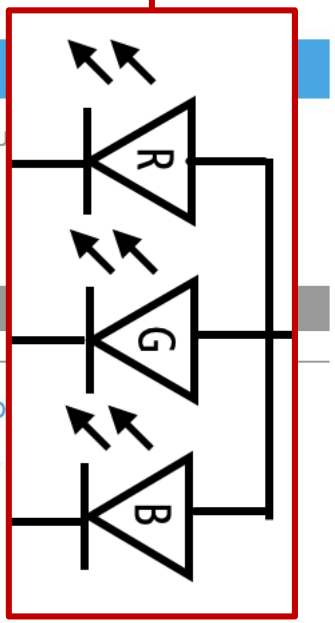
$2.00

49 IN STOCK

1

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

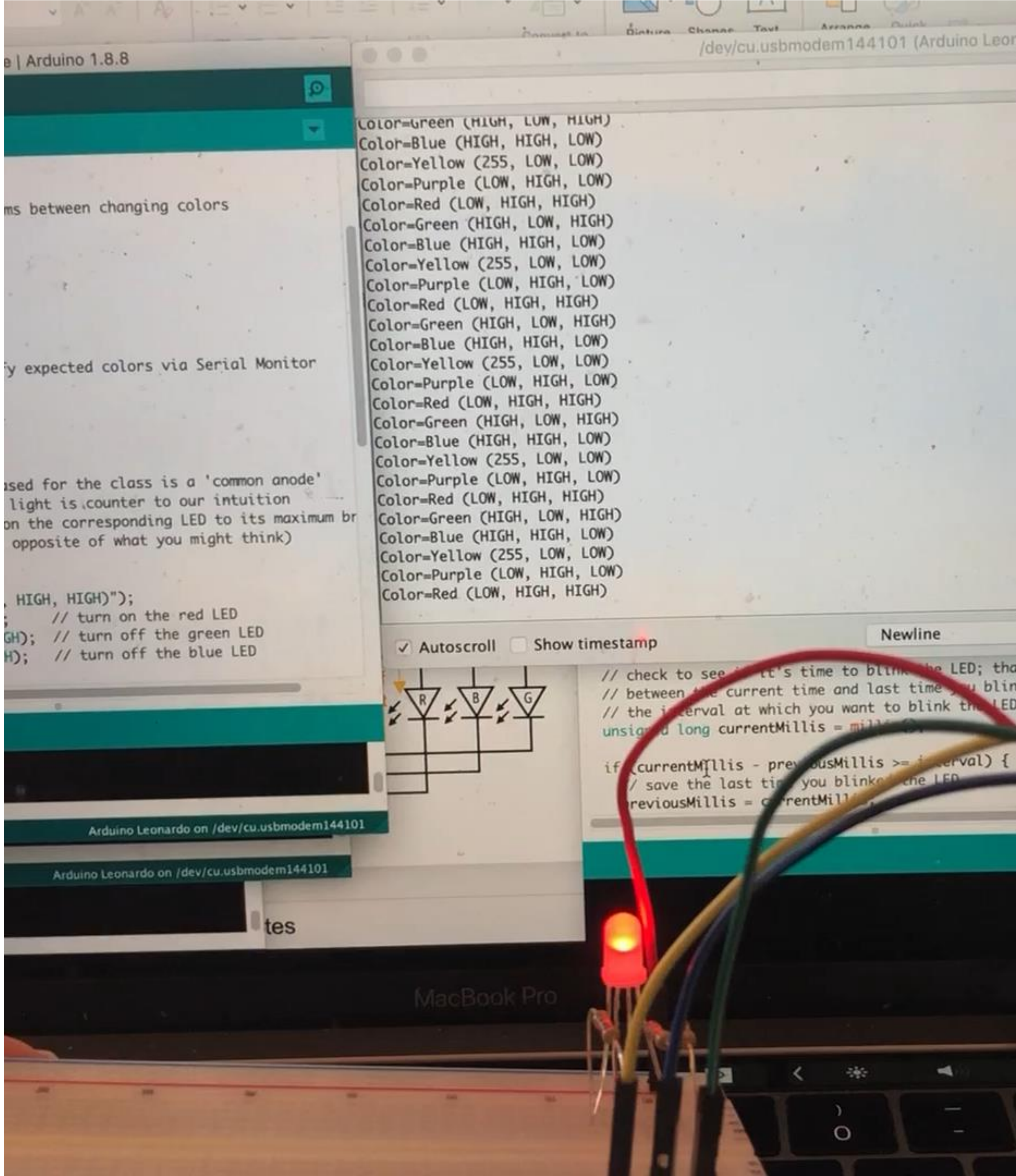
DESCRIPTIO
TECHNICAL
LEARN



ACTIVITY

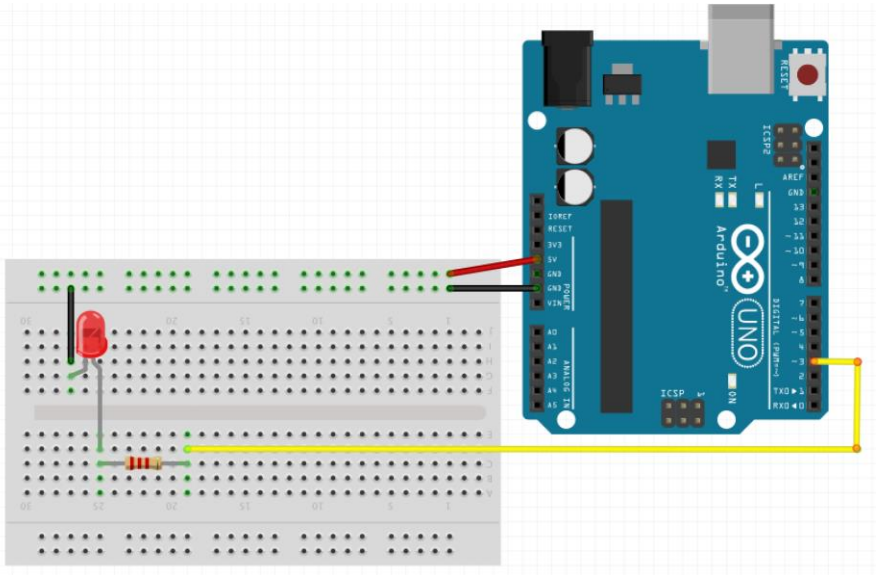
FLASH RGB LED COLORS

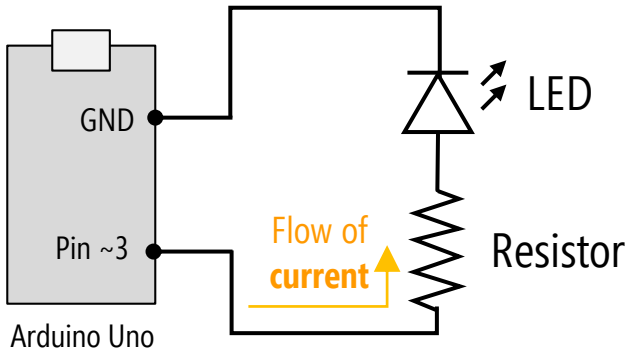
Build a circuit & write code to flash through different RGB LED colors



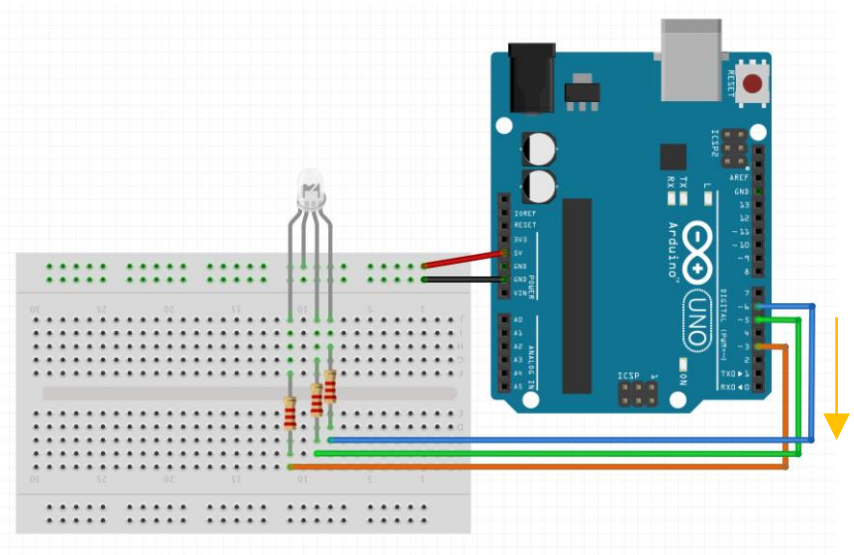
USING AN RGB LED: THE CIRCUIT

Old Circuit: Turn LED on/off via pin 3

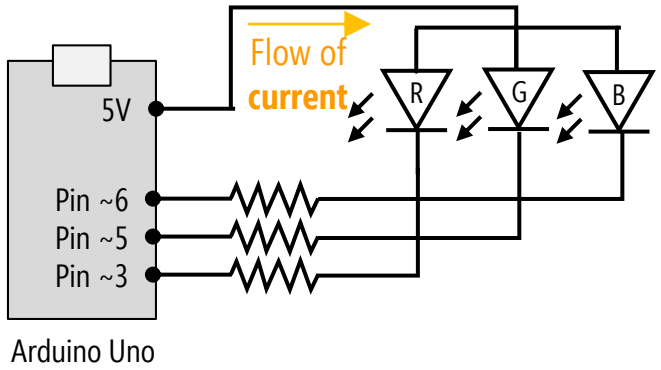




New Circuit: Set R, B, G values via pin 6, 5, and 3

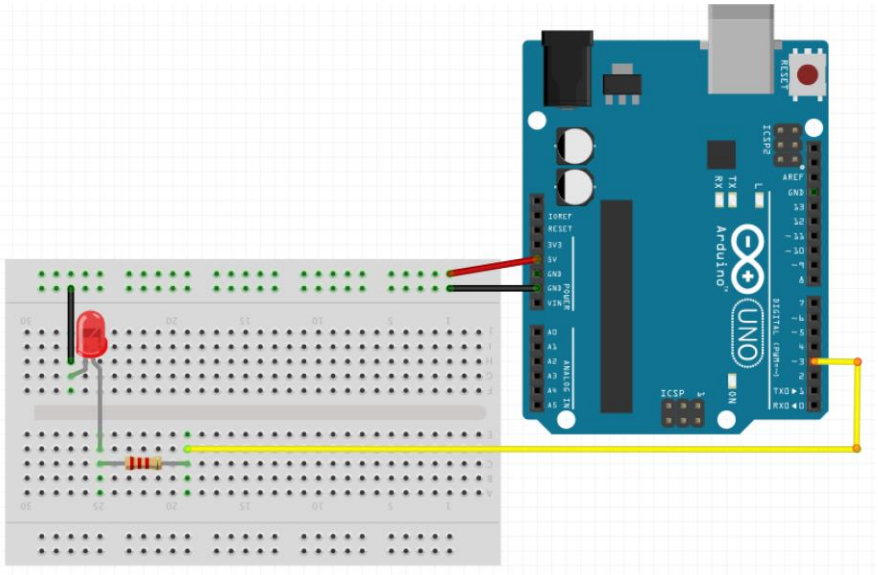


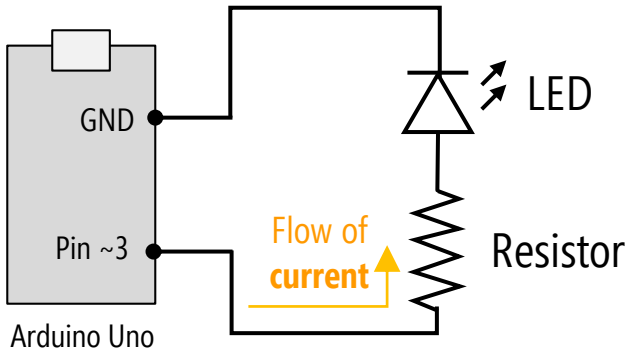
Flow of current
Because this is a "common anode" RGB. We set these pins to zero to turn on the light and 5V to turn off the light!



USING AN RGB LED: THE CIRCUIT

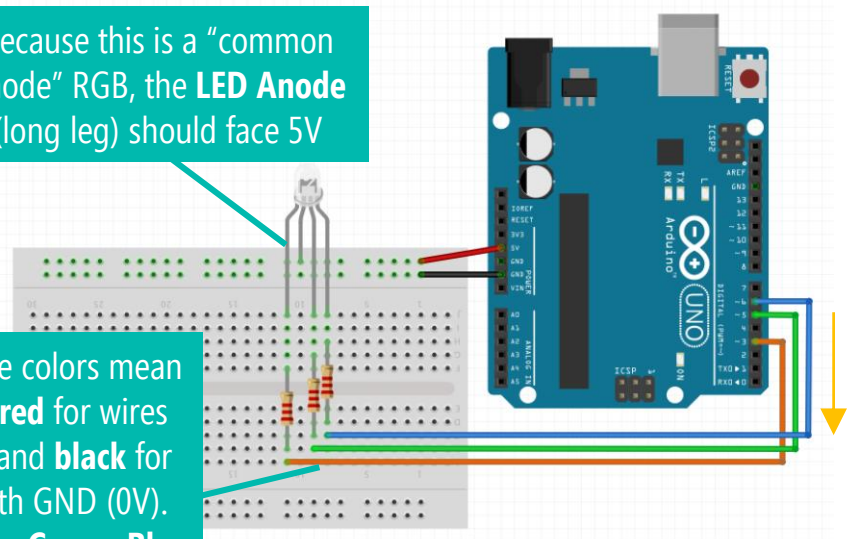
Old Circuit: Turn LED on/off via pin 3





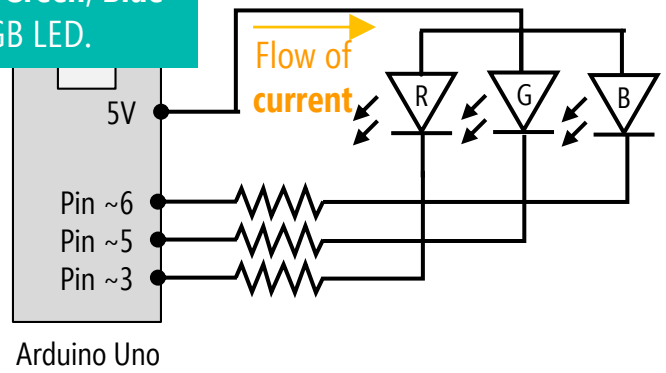
New Circuit: Set R, B, G values via pin 6, 5, and 3

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



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

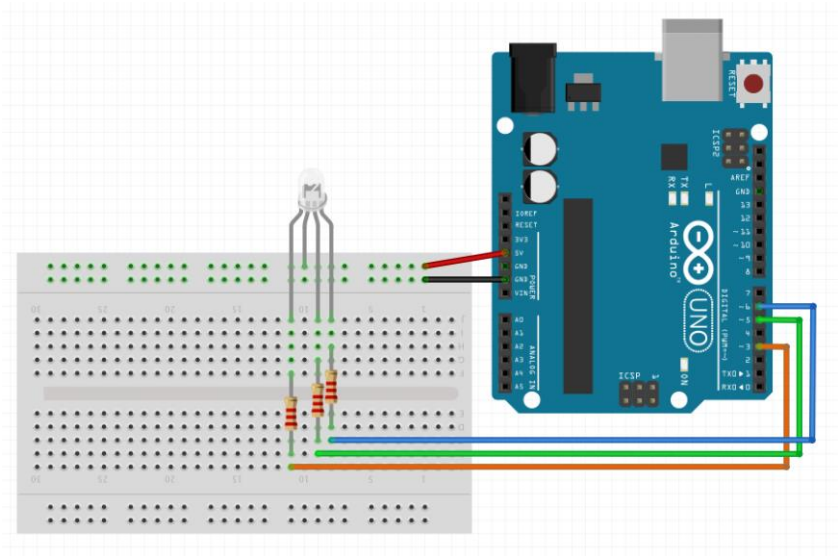
Flow of current
Because this is a "common anode" RGB. We set these pins to zero to turn on the light and 5V to turn off the light!

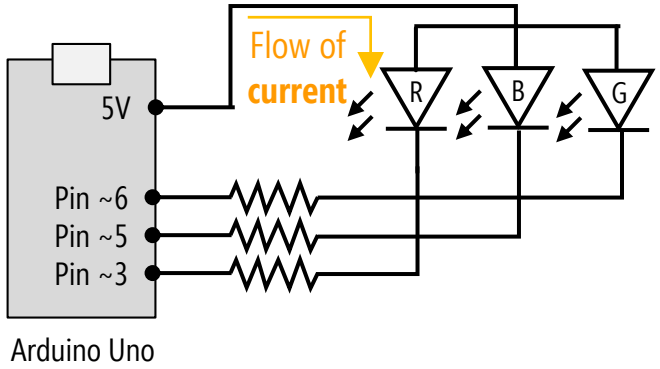


DIGITAL OUTPUT

USING AN RGB LED: THE CODE

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





BlinkRGBSimple §

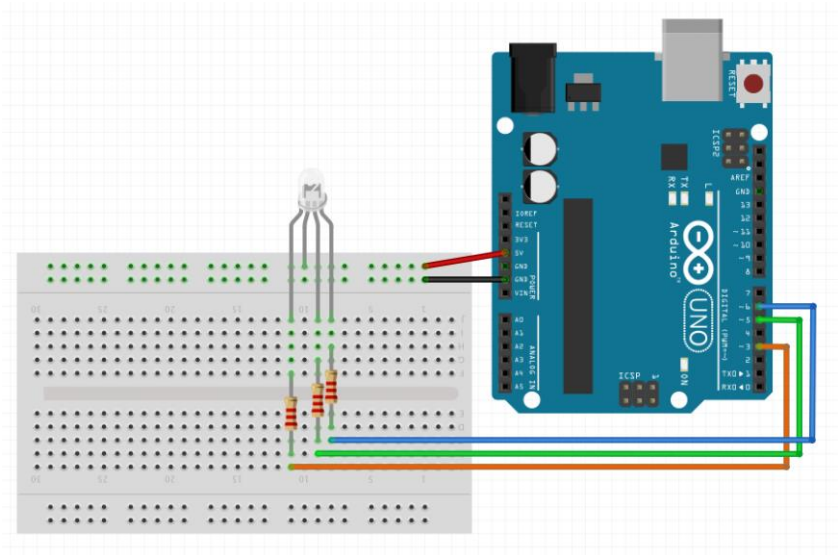
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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

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

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

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

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

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

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

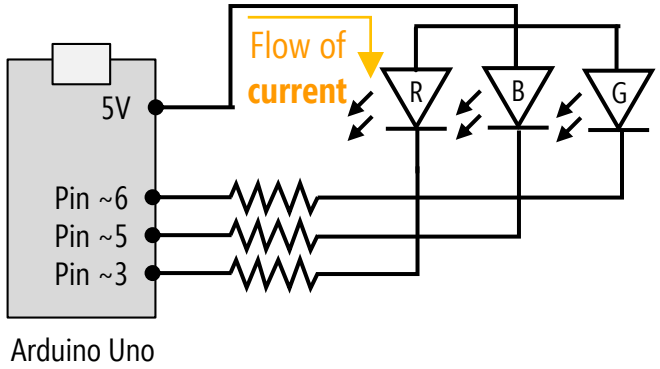
DIGITAL OUTPUT

RGB LED WALKTHROUGH

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

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





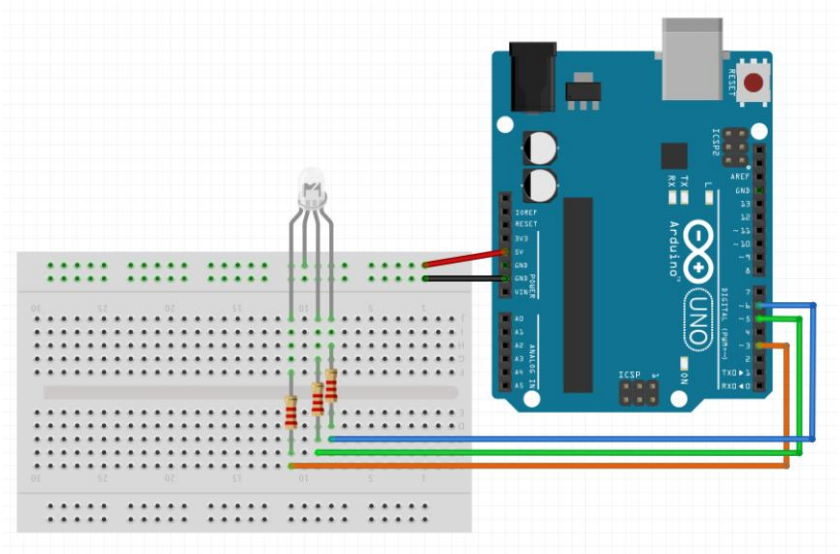
BlinkRGBSimple §

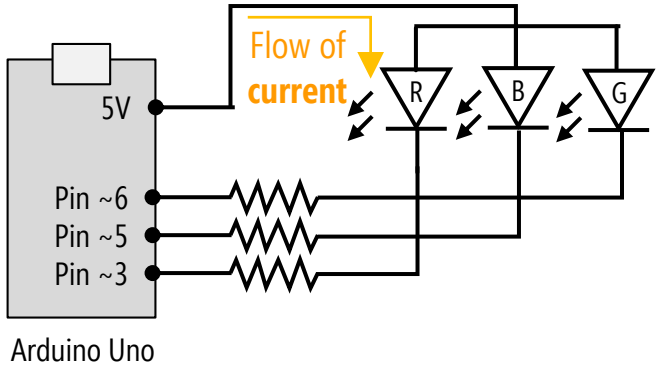
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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

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

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

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

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

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

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

DIGITAL OUTPUT

RGB LED WALKTHROUGH

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





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

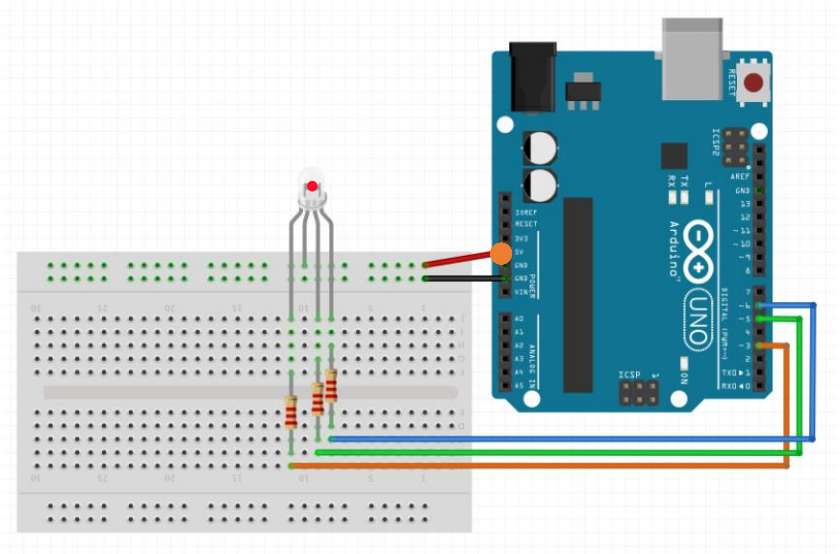
BlinkRGBSimple §

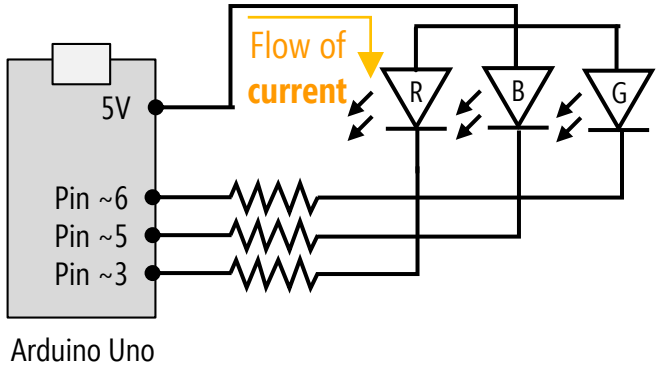
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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

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

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

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

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

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

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

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

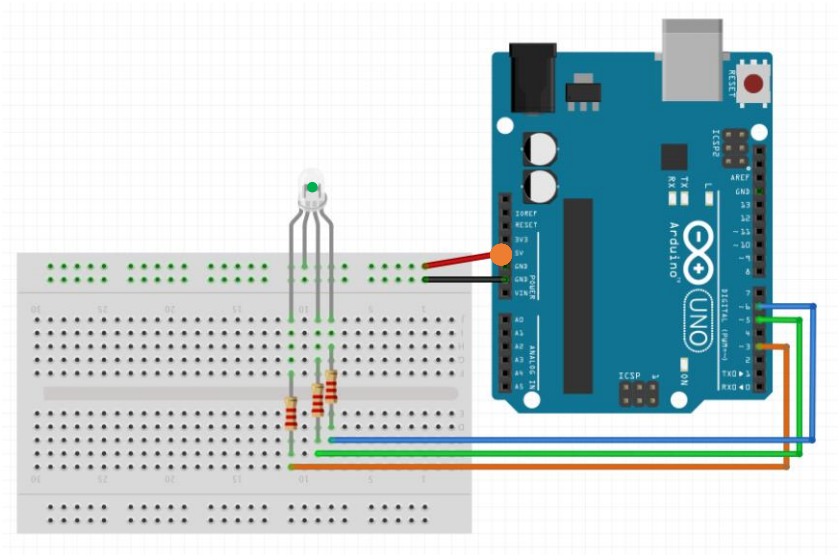
BlinkRGBSimple §

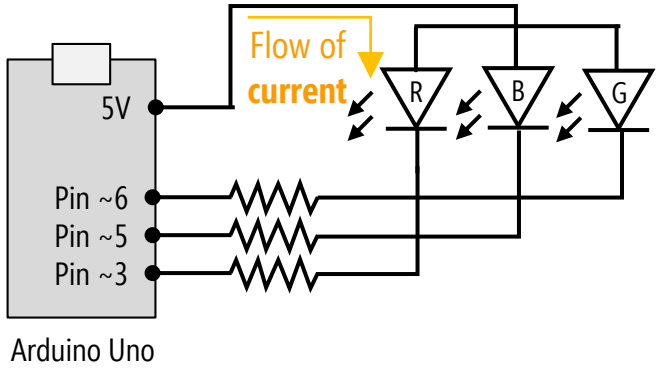
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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 **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)

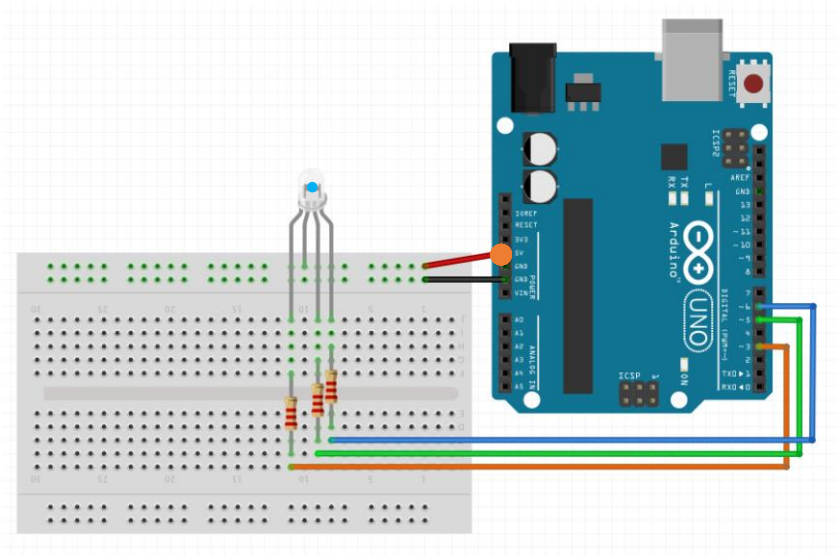
BlinkRGBSimple §

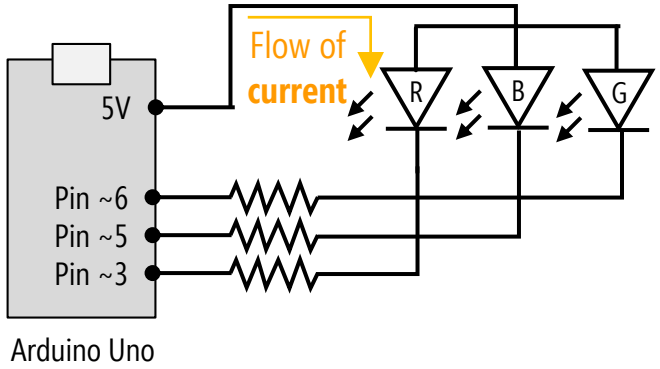
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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





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)

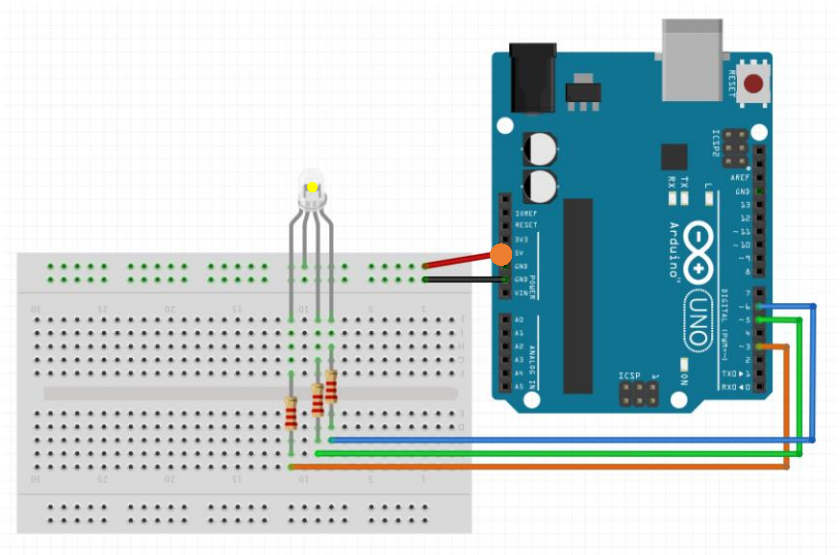
BlinkRGBSimple §

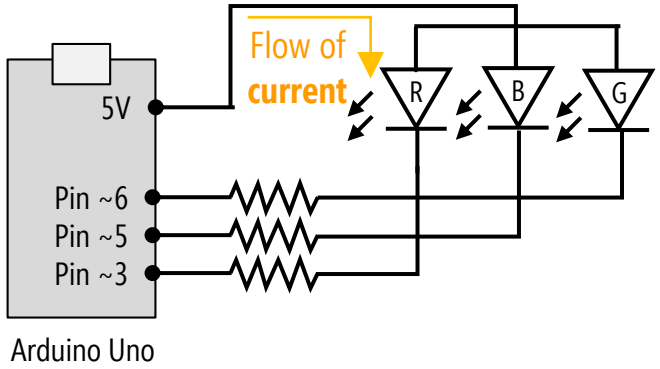
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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





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)

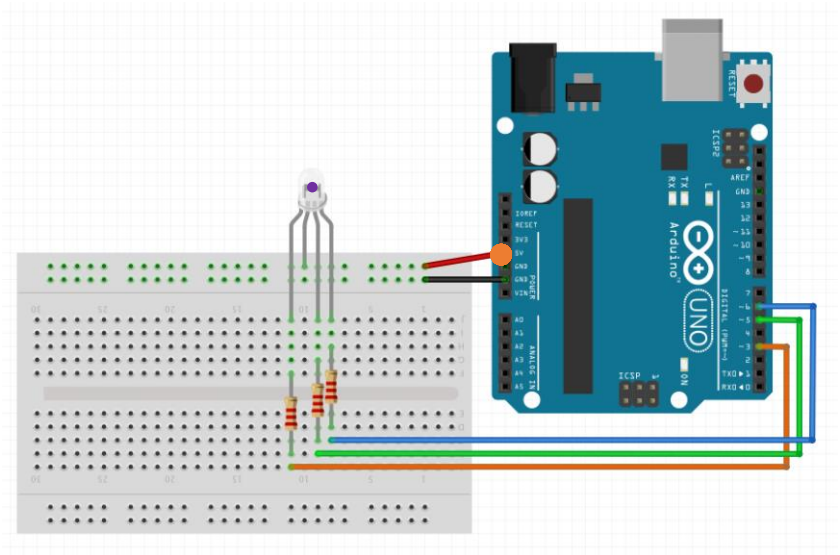
BlinkRGBSimple §

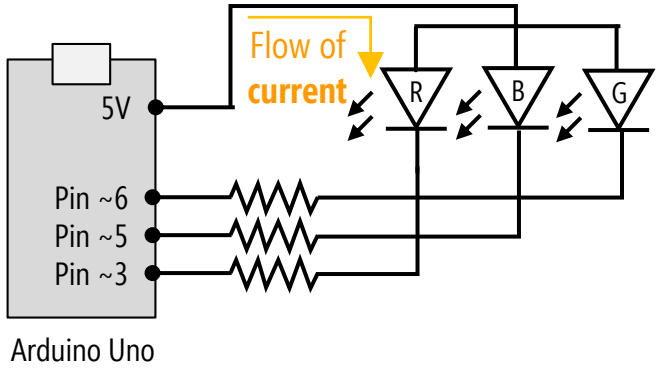
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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





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)

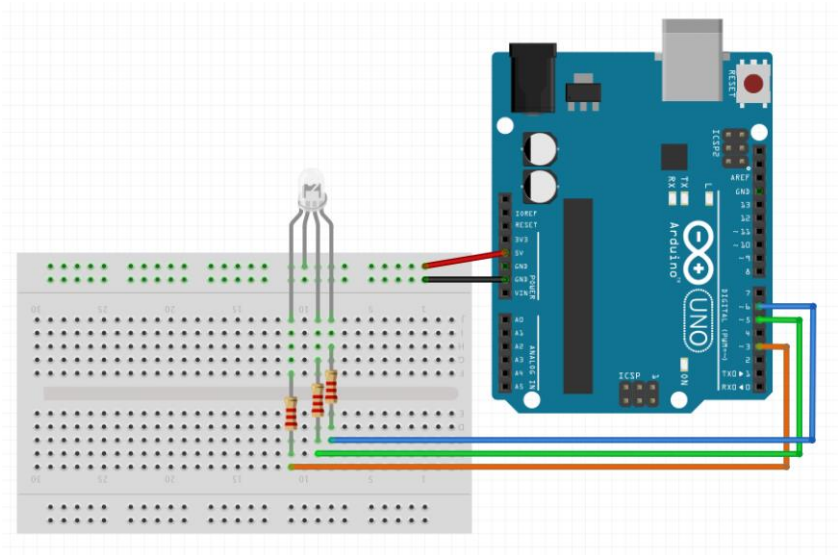
BlinkRGBSimple §

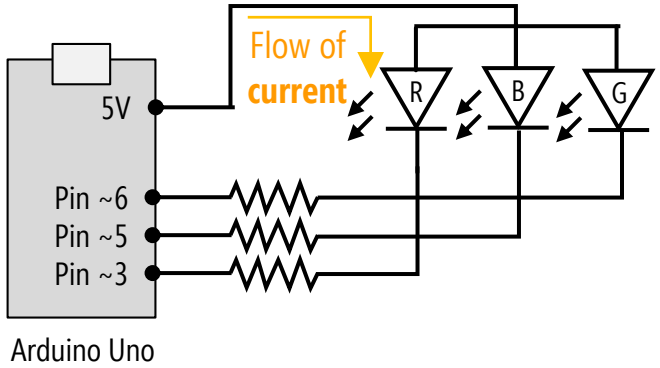
```
const int RGB_RED_PIN = 6;
const int RGB_GREEN_PIN  = 5;
const int RGB_BLUE_PIN  = 3;
const int DELAY = 1000; // delay in ms between changing colors

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



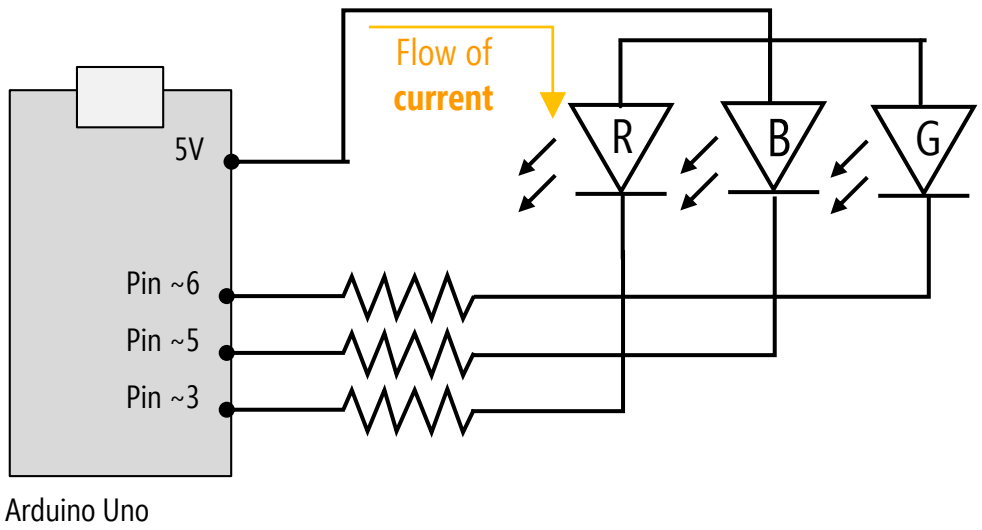


When the loop() function ends, it's automatically called again by the Arduino (and will continue looping forever)

BlinkRGBSimple §

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

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

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

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

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

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

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

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

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



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

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

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

PHYSICAL COMPUTING 2

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)

PHYSCOMP 2: ARDUINO & BASIC OUTPUT

CSE 599 Prototyping Interactive Systems | Lecture 3 | Oct 3

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON