PHYSICAL COMPUTING 2: ARDUINO

CSE 590 Ubiquitous Computing | Lecture 5 | April 26

Jon Froehlich • Liang He (TA)

DESIGN USE BUILD

MAKEABILITY LAB

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY of WASHINGTON

PHYSICAL COMPUTING 2

INTRO TO ARDUINO: LEARNING GOALS

What is **Arduino** and how to use it

Difference between **analog** and **digital**

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

How to use **analog output** (*i.e.*, fading an LED using analogWrite)

What is **Pulse Width Modulation** (PWM)?

How to use **analog input** (*i.e.*, hooking up a POT and using analogRead)

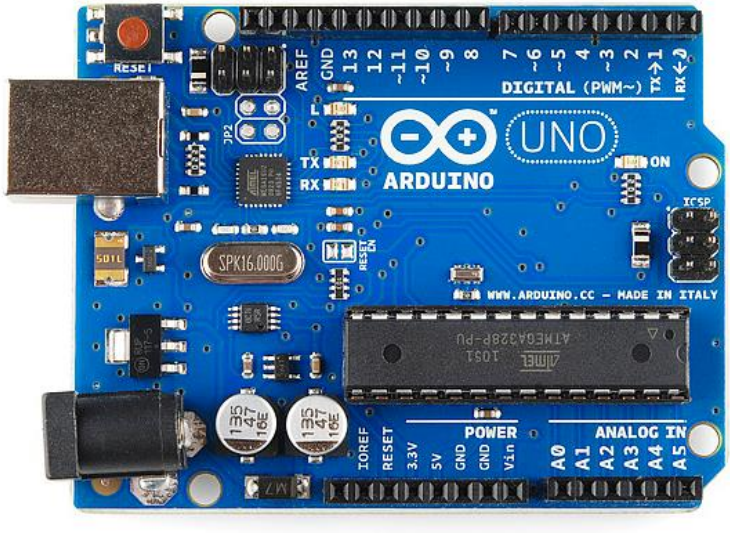
How to use **digital input** (*i.e.*, hooking up a button and using digitalRead)

ARDUINO

WHAT IS ARDUINO?

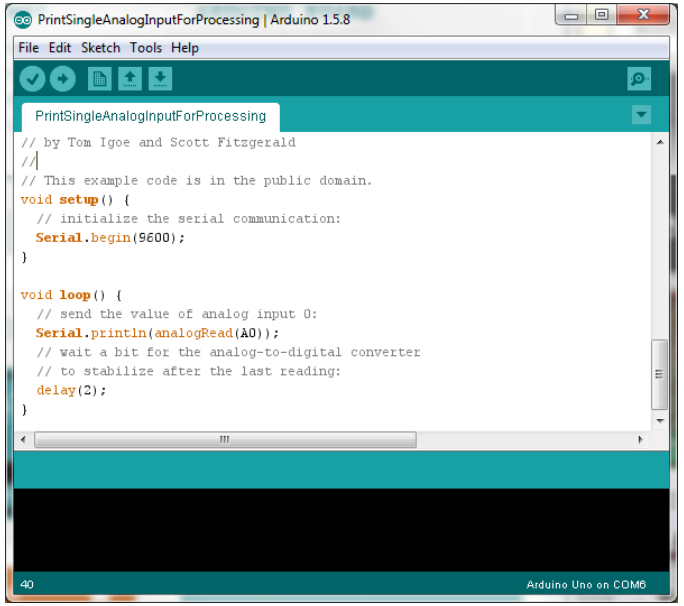
Arduino is an **open-source electronics platform** based on easy-to-use hardware and software. It's intended to enable designers, artists, hobbyists, engineers, and even children to (rapidly) create new interactive experiences.

Arduino Hardware



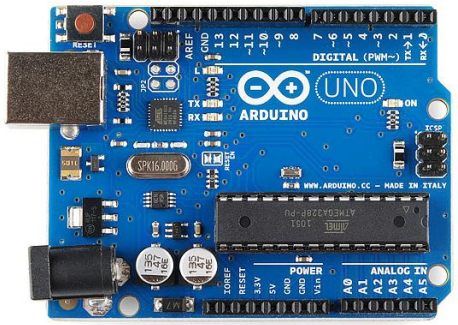
The Arduino board contains a microcontroller (small computer) and is used to sense the environment (or people, animals, things) by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators.

Arduino Software

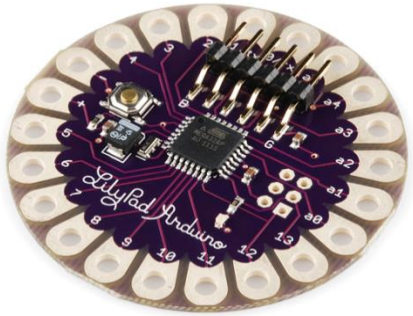


The Arduino is programmed via the Arduino programming language and using the Arduino development environment.

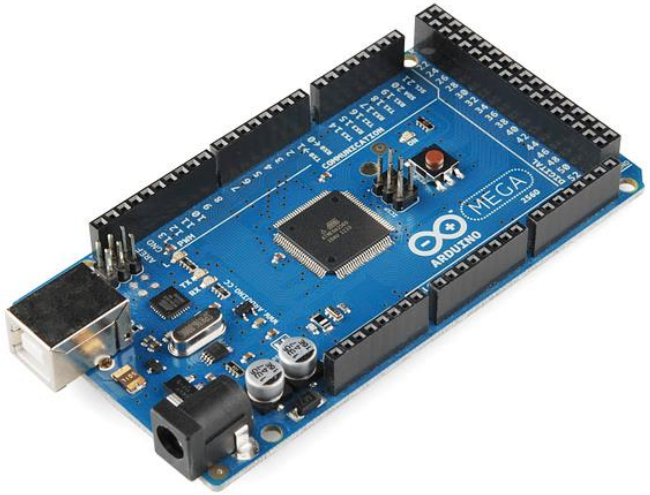
POPULAR ARDUINO MODELS



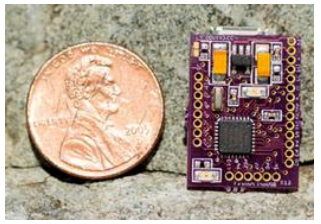
Arduino Uno (R3)
14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack, a reset button



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.



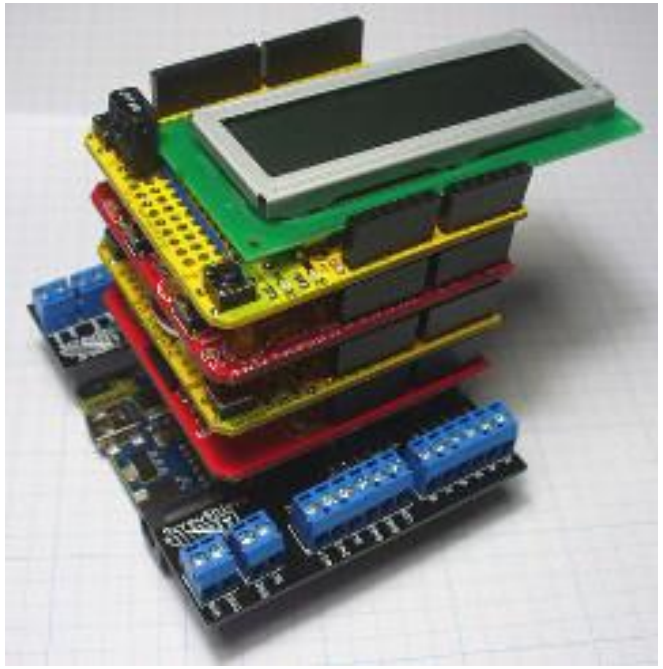
Arduino Mega (R3)
The Arduino Mega is like the UNO's big brother. It has lots (*54!*) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button

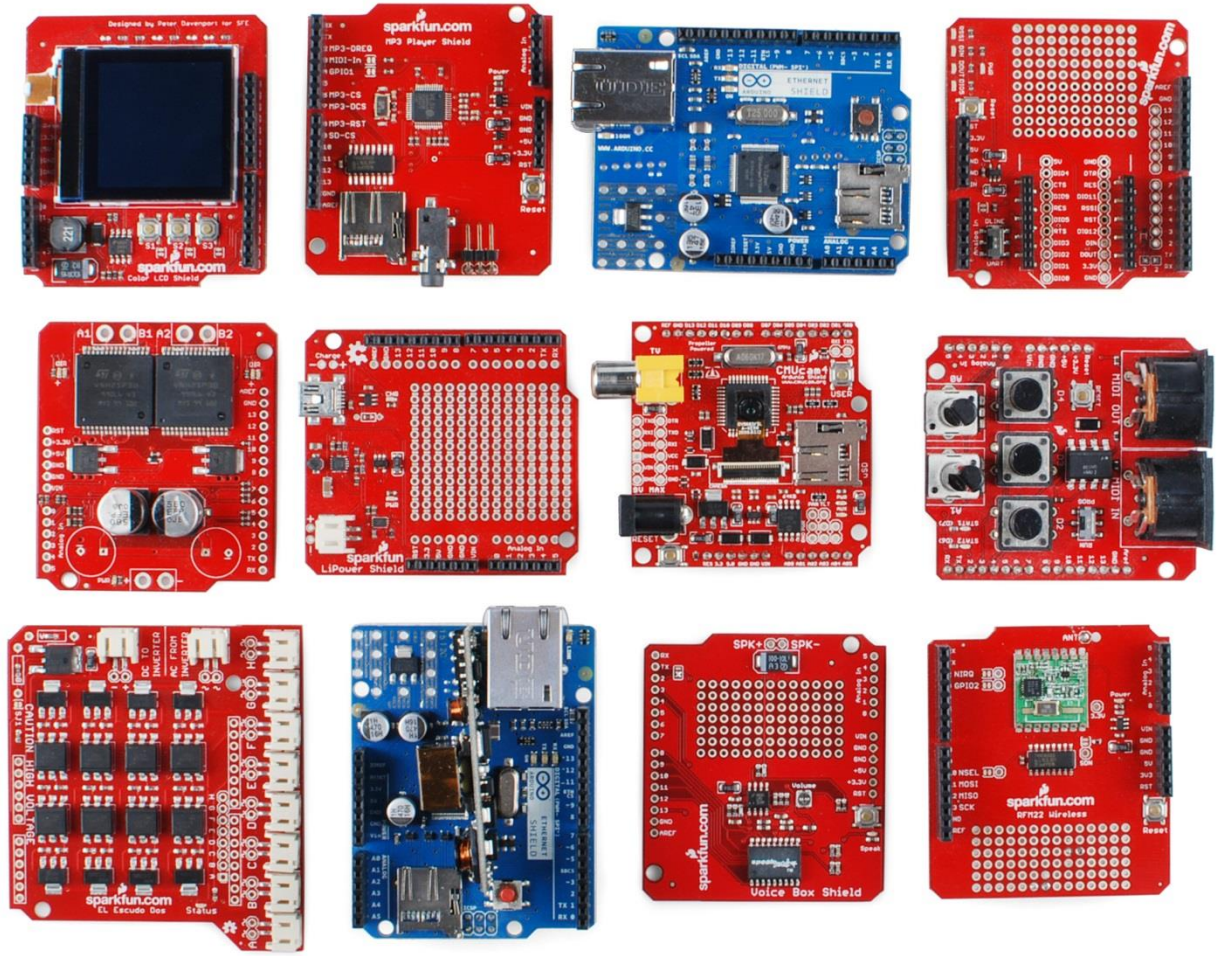


FemtoduinoUSB
The smallest Arduino board with micro-USB, a voltage regulator, same power and pin count as an Arduino UNO

ARDUINO UNO STACKABLE SHIELDS

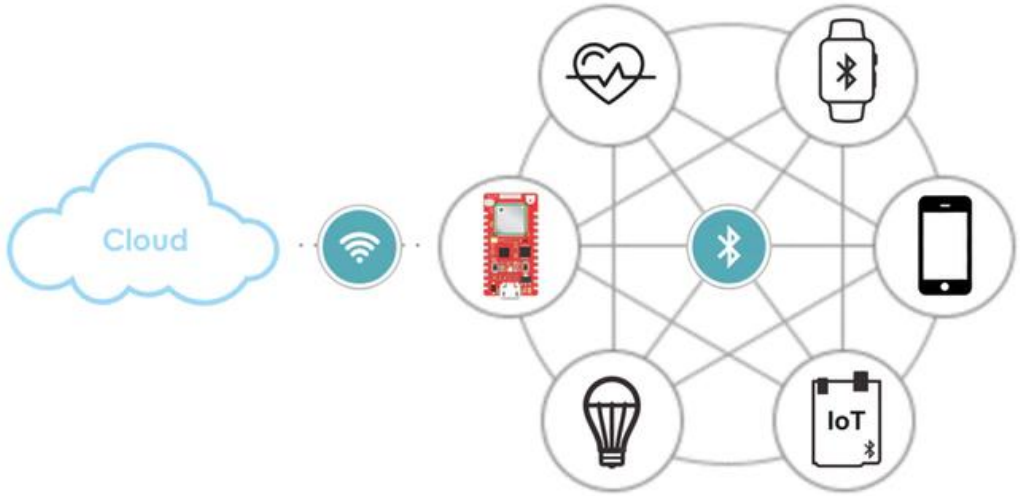
You can purchase stackable shields that give your Arduino more features such as WiFi, controlling motors, cellular modems, controlling LCD screens, etc.





ARDUINO

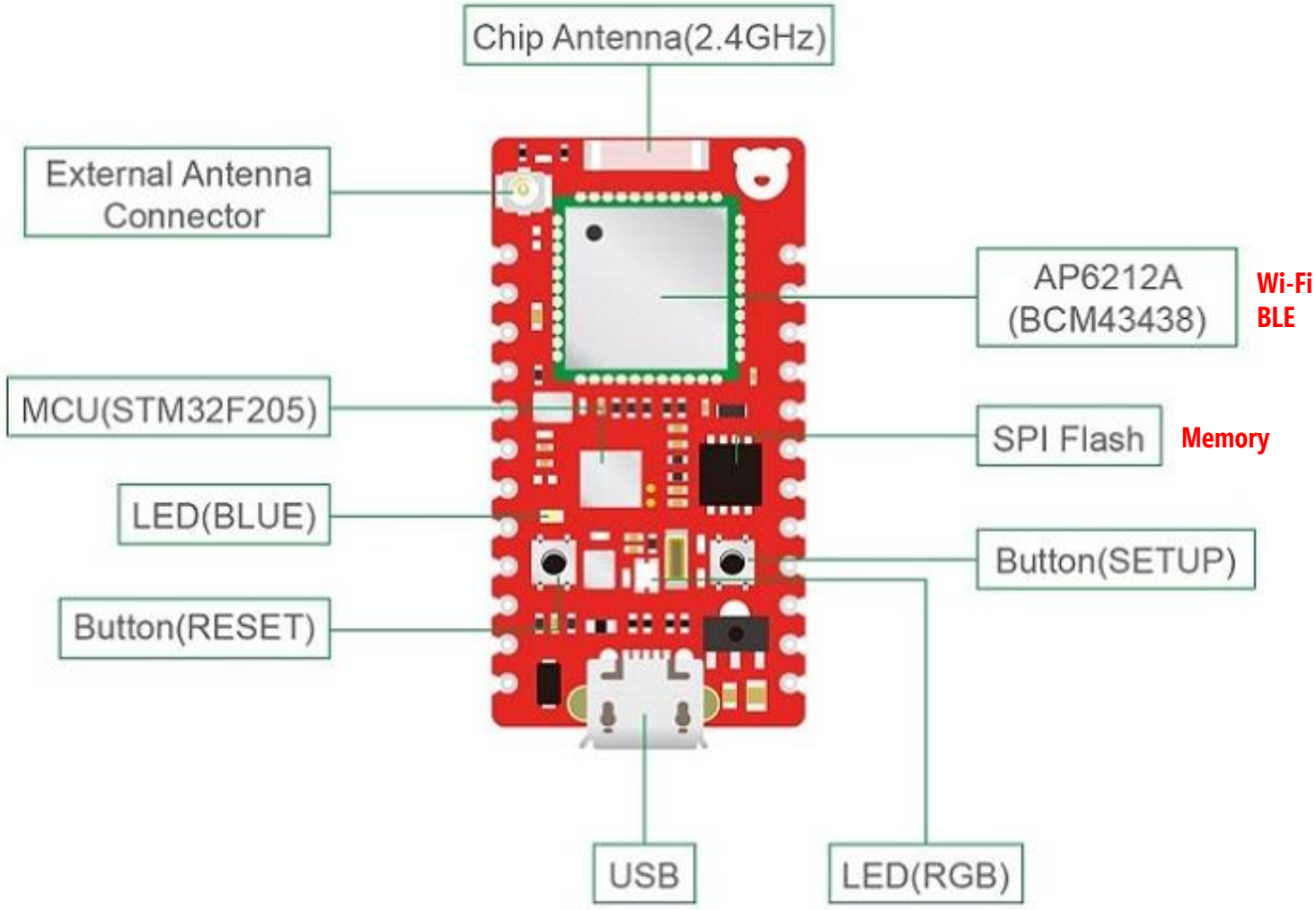
REDBEAR DUO: BLE + WIFI BOARD



Sources: https://github.com/redbear/Duo/blob/master/docs/duo_introduction.md

ARDUINO

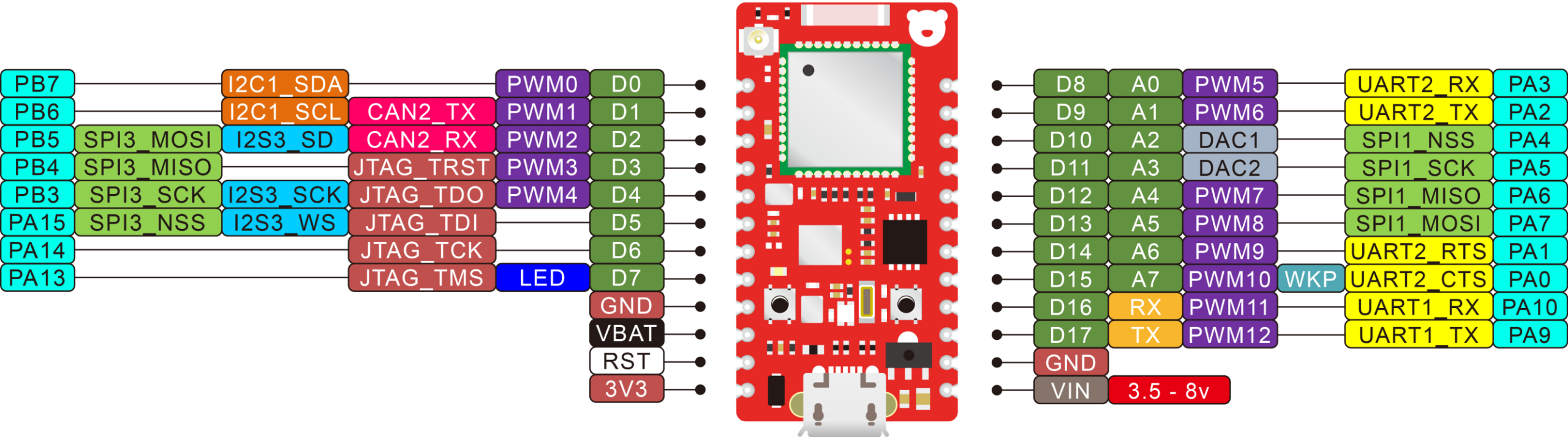
REDBEAR DUO: BLOCK DIAGRAM



Source: https://github.com/redbear/Duo/blob/master/docs/duo_introduction.md

ARDUINO

REDBEAR DUO: PIN LAYOUT

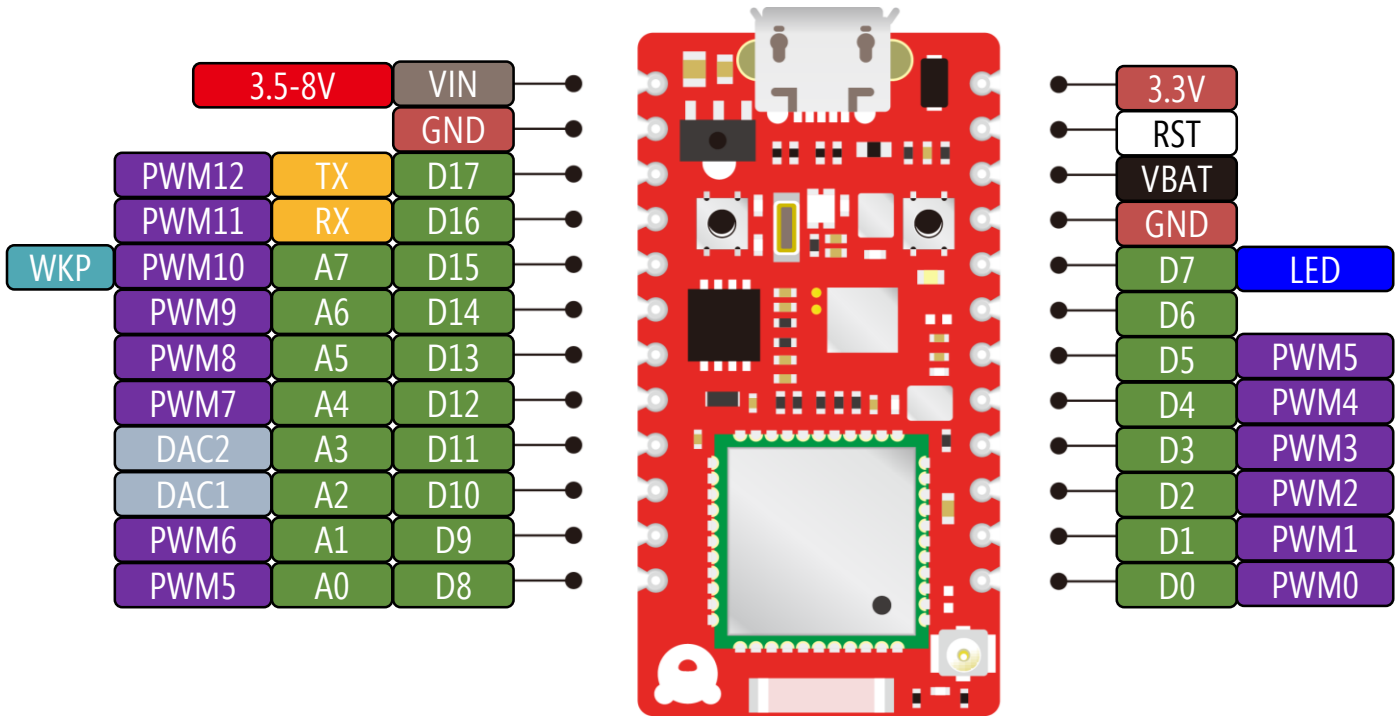


Sources: https://github.com/redbear/Duo/blob/master/docs/duo_introduction.md

ARDUINO

REDBEAR DUO: PIN LAYOUT

I've simplified the pin layout chart and flipped it around (I tend to program with the USB facing away from me)



Legend

AX	Analog Input Pin
DX	Digital Input/output Pin
PWM	Supports Analog Out via Pulse Width Modulation

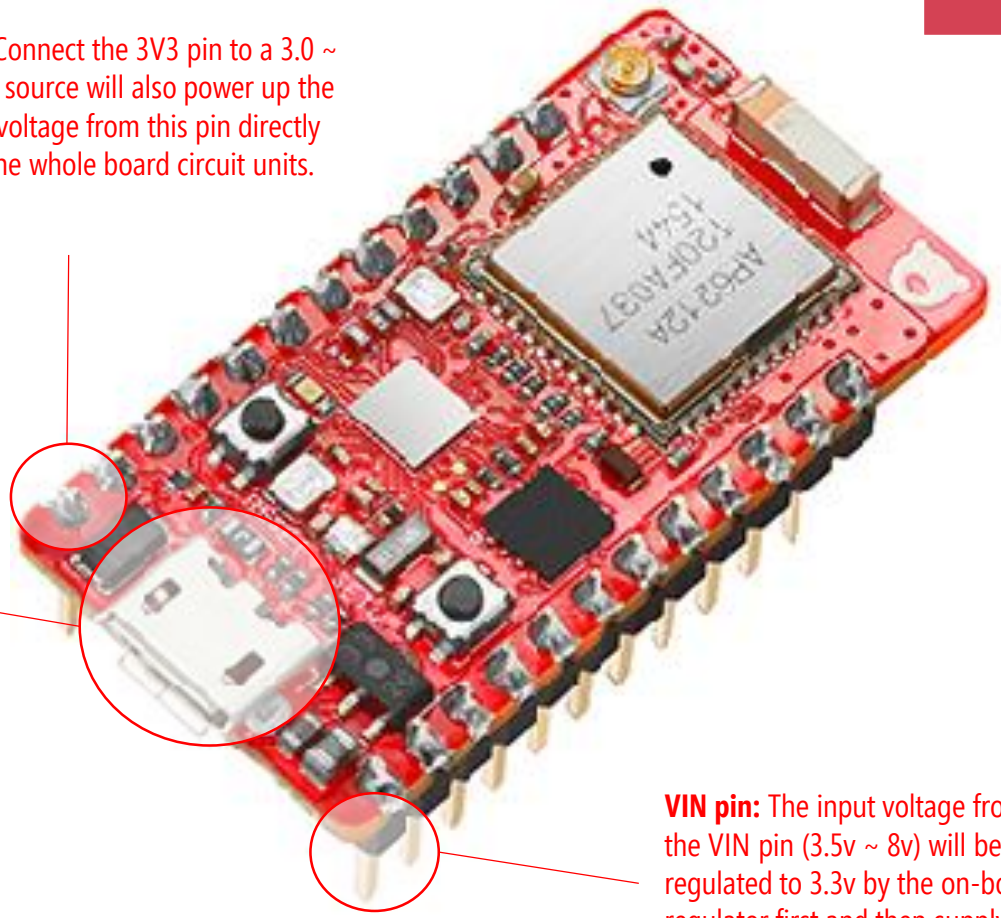
Modified from: https://github.com/redbear/Duo/blob/master/docs/duo_introduction.md

ARDUINO

REDBEAR DUO: POWER

Use only one power source as the main power supply at a time, otherwise you will damage the board!

3V3 pin: Connect the 3V3 pin to a 3.0 ~ 3.6 power source will also power up the Duo. The voltage from this pin directly supplies the whole board circuit units.



Micro USB: The Micro USB allows you to communicate with your computer to program the Arduino and also supplies power (3.3V). Easiest option for rapid prototyping.

VIN pin: The input voltage from the VIN pin (3.5v ~ 8v) will be regulated to 3.3v by the on-board regulator first and then supply the whole board circuit units.

Sources: https://github.com/redbear/Duo/blob/master/docs/duo_introduction.md

ARDUINO

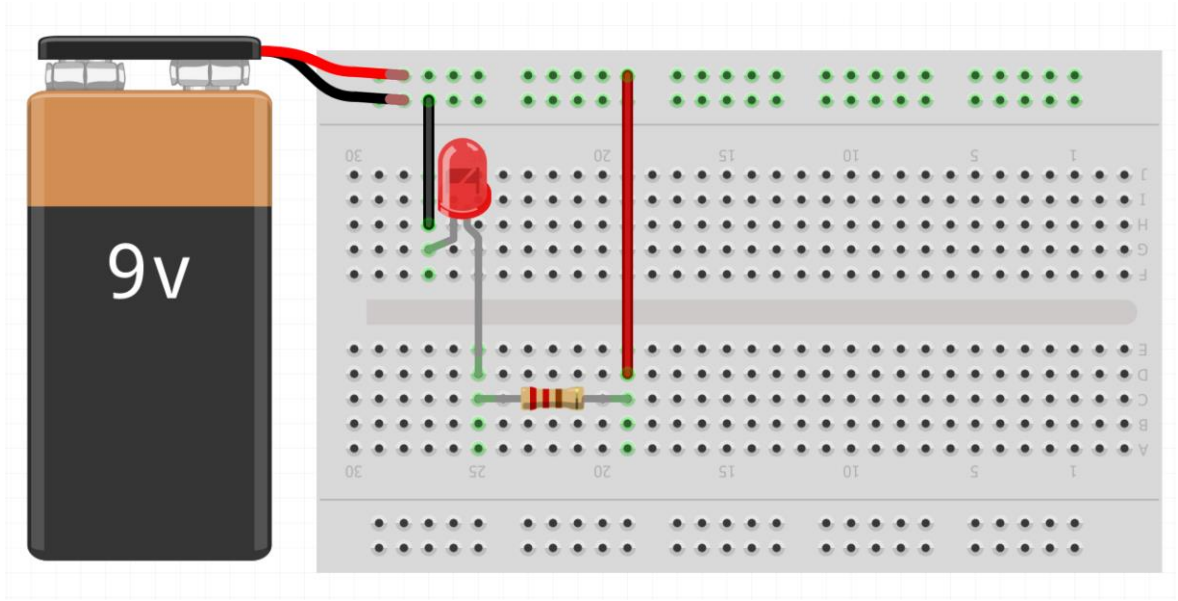
POWER YOUR CIRCUIT VIA REDBEAR DUO 3.3V OUTPUT

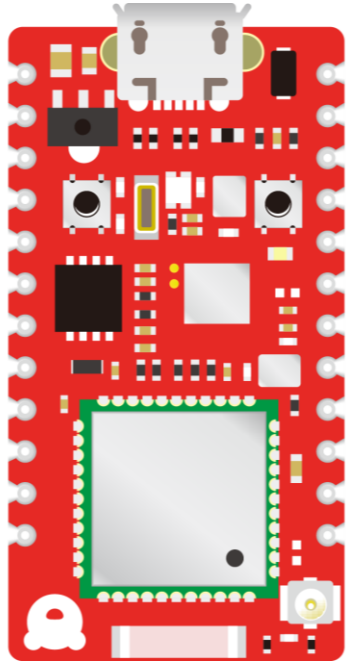
For our first exercise, we are just going to convert the circuit you previously made with the barrel jack + 9V battery to, instead, use the Arduino RedBear Duo 3.3V and GND connections. This is a bit of a toy exercise given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! ☺

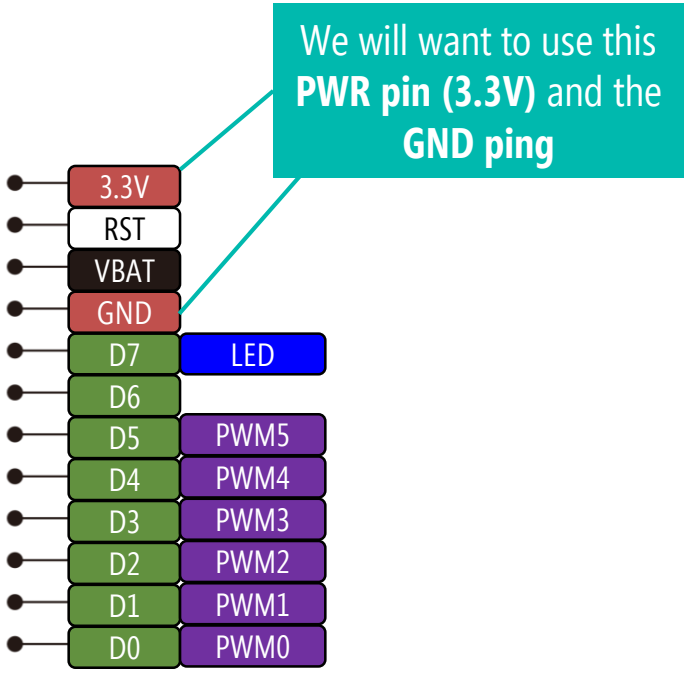
ARDUINO

POWER YOUR CIRCUIT VIA REDBEAR DUO 3.3V OUTPUT

For our first exercise, we are just going to convert the circuit you previously made with the barrel jack + 9V battery to, instead, use the Arduino RedBear Duo 3.3V 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! ☺

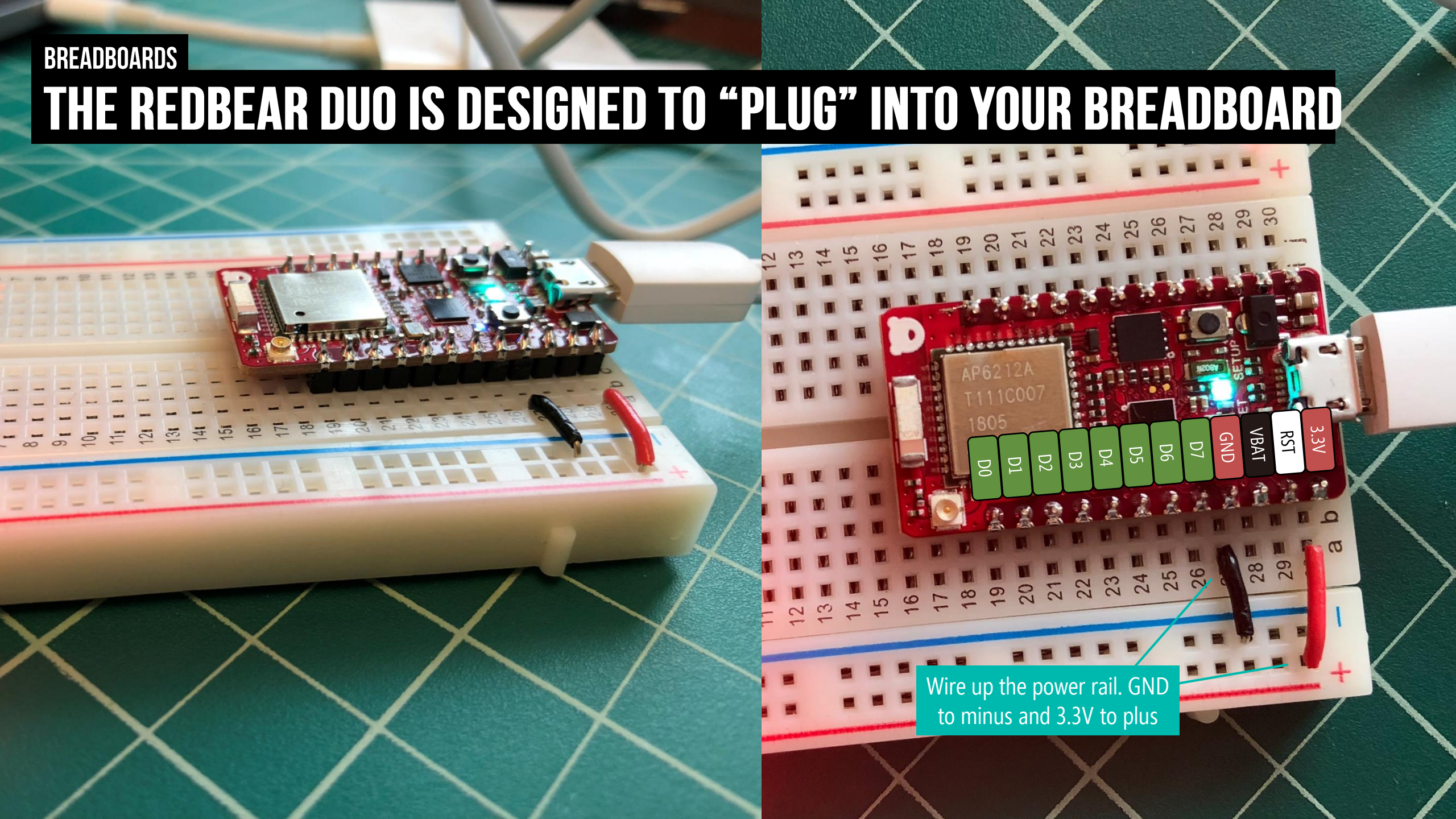






BREADBOARDS

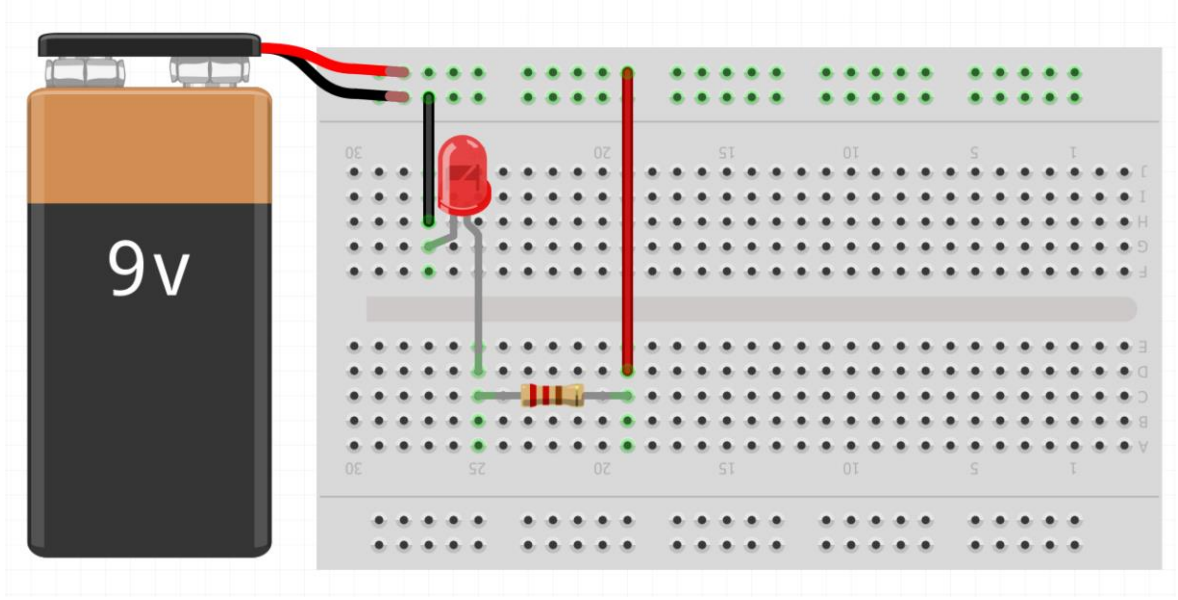
THE REDBEAR DUO IS DESIGNED TO “PLUG” INTO YOUR BREADBOARD

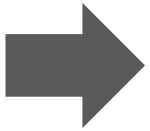


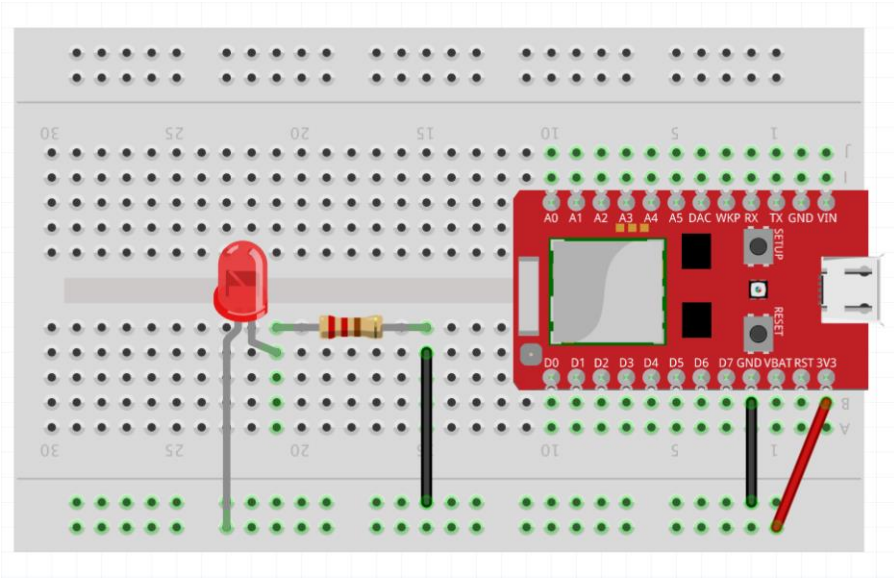
ARDUINO

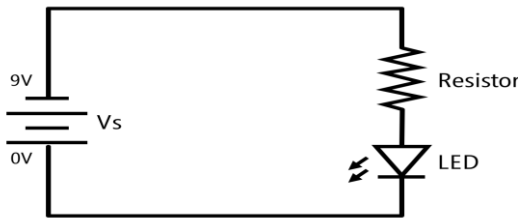
POWER YOUR CIRCUIT VIA REDBEAR DUO 3.3V OUTPUT

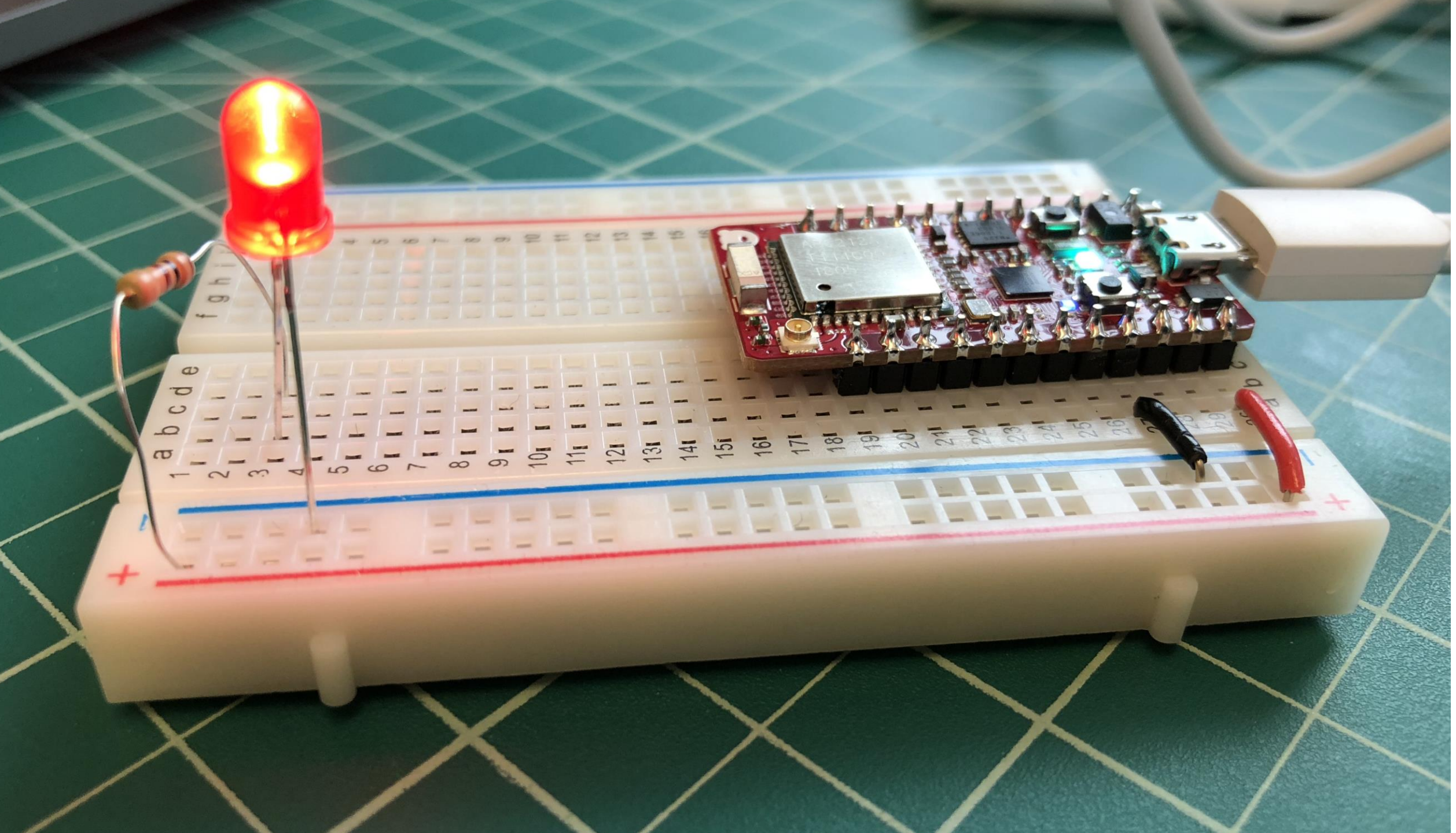
For our first exercise, we are just going to convert the circuit you previously made with the barrel jack + 9V battery to, instead, using the Arduino RedBear Duo 3.3V and GND connections. This is a bit of a toy exercise given that we are not yet taking advantage of the microcontroller platform—we are only using it to power our circuit! ☺











OK, now let's do something more fun and interactive! Let's program the Arduino to control the light. To do this, we have to introduce the Arduino programming framework.

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

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)

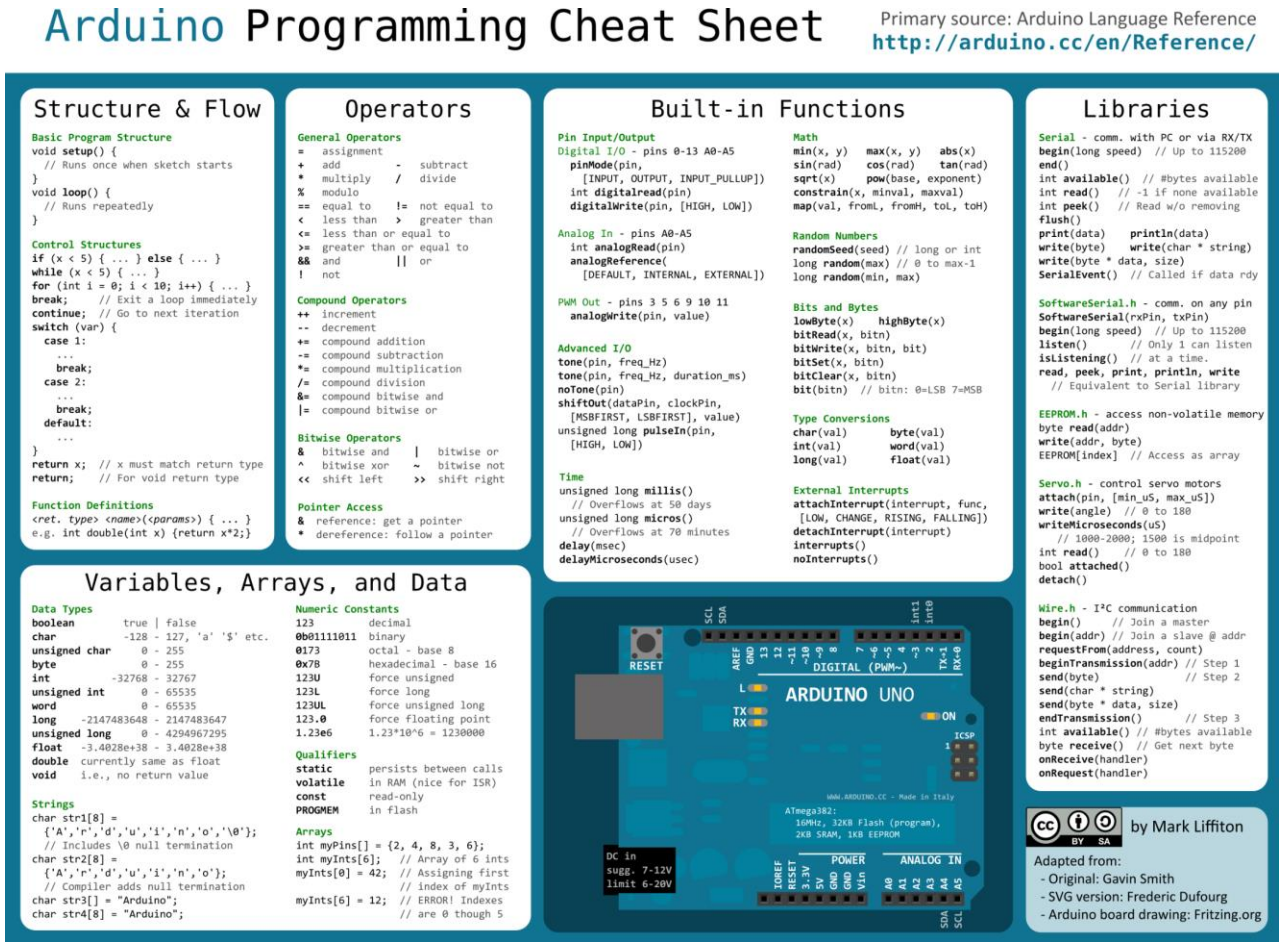
// (single line comment)

; (semicolon)

{} (curly braces)

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

ARDUINO CHEAT SHEET



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.).

While this cheat sheet was obviously written for the Arduino Uno—all of the functionality is the same. However, the **pin layout on the Uno is different** from the RedBear Duo and the **Uno runs** at **5V**, your **Duo runs** at **3.3V**!

ACTIVITY: LEARN DIGITAL OUTPUT

PROGRAM ARDUINO!

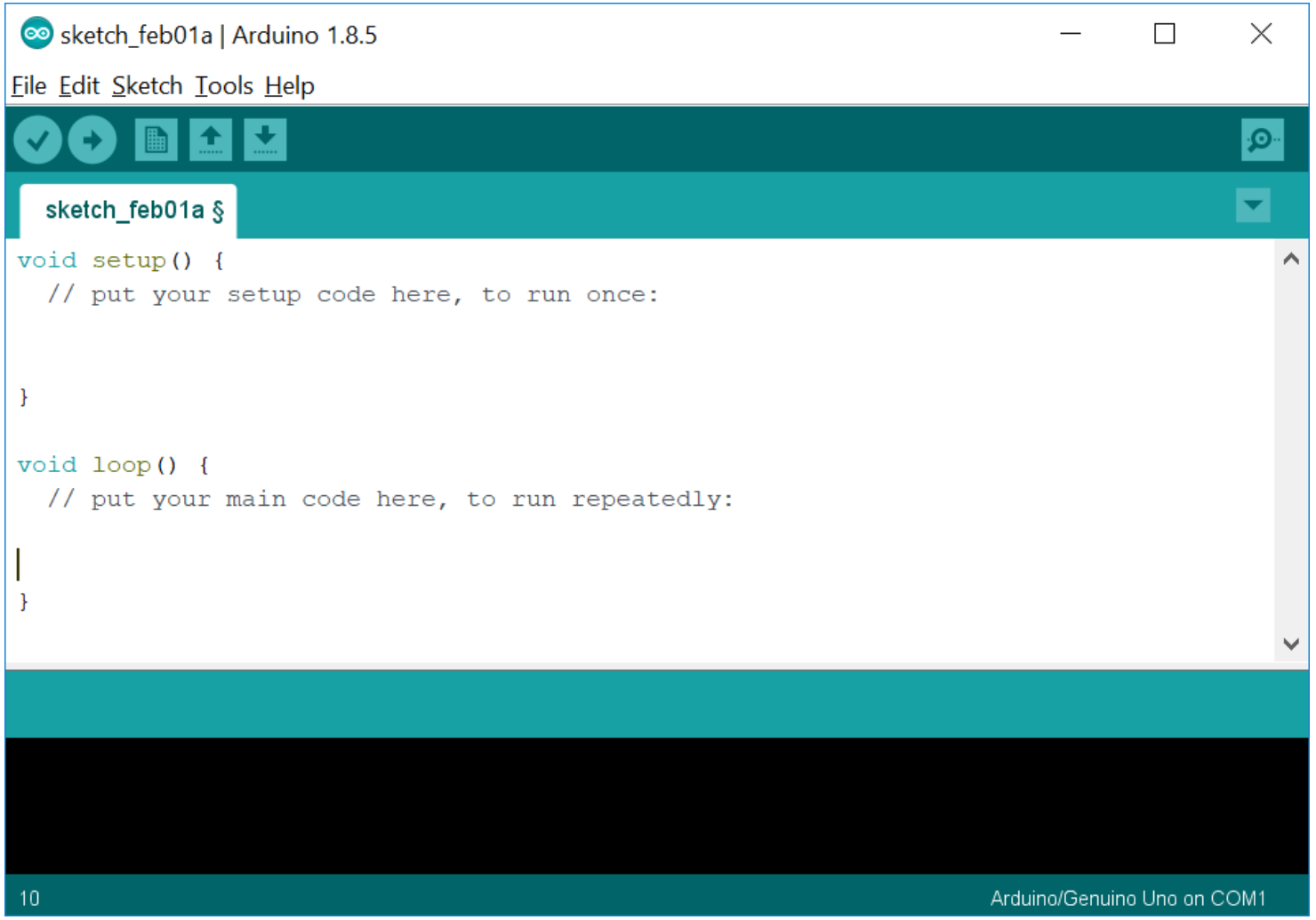
All Arduino programs are comprised of two areas: *setup* and *loop*.

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



PROGRAMMING ARDUINO

SETTING UP ARDUINO IDE FOR REDBEAR DUO

CSE590_RedBear_Duo_Programming_101

File Edit View Insert Format Tools Add-ons Help Last edit was made yesterday at 5:41 PM by Liang He

SHARE

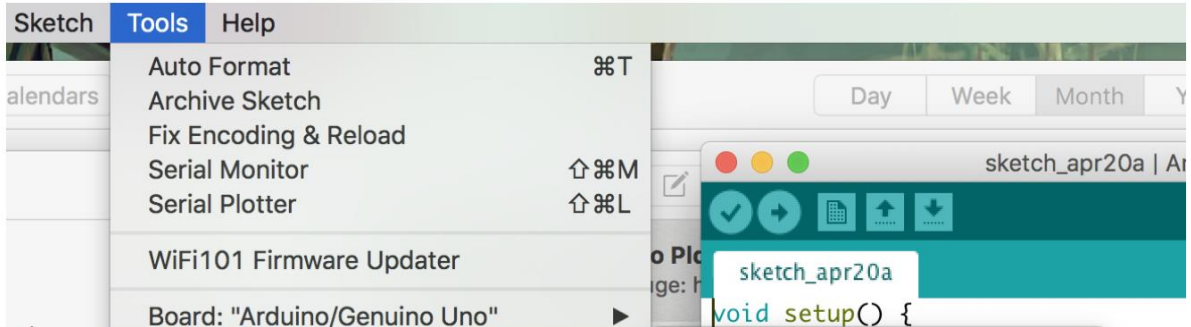
100% Normal text Arial 11

Arduino IDE Configuration for RedBear Duo Programming

1. Download and install Arduino IDE 1.6.7 or above (latest is 1.8.5): https://www.arduino.cc/en/Main/Software

2. Plug in the RedBear Duo board to your laptop/computer with a micro-USB to USB cable. Launch Arduino and select the right port under **"Tools->Port"**, select the port that the RedBear Duo board uses

   (On MacOSX) The port looks like the one in the image below (either /dev/tty.usbmodem* or /dev/cu.usbmodem*).



PROGRAMMING ARDUINO

SETTING UP ARDUINO IDE FOR REDBEAR DUO

CSE590_RedBear_Duo_Programming_101

File Edit View Insert Format Tools Add-ons Help Last edit was made yesterday at 5:41 PM by Liang He

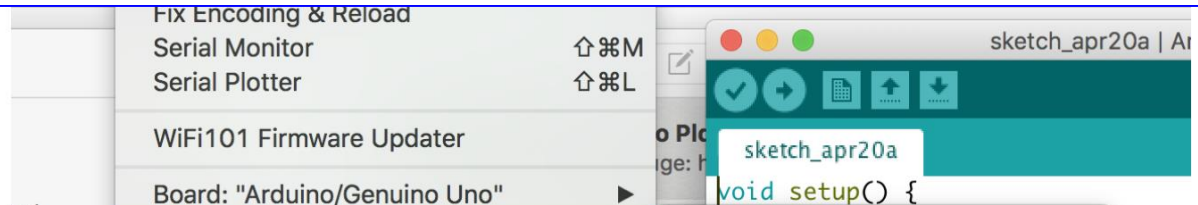
SHARE

100% Normal text Arial 11

Arduino IDE Configuration for RedBear Duo Programming

Go here for setup instructions:

https://goo.gl/BERpSG



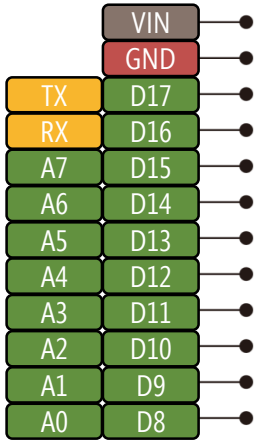
INTRODUCTION TO I/O

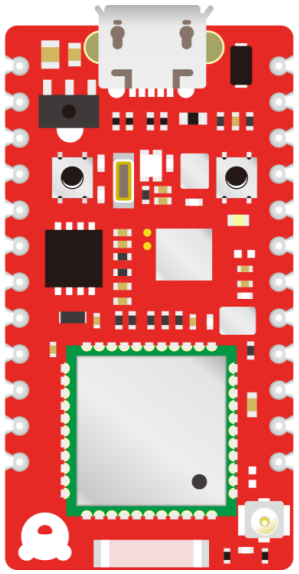
INPUT VS. OUTPUT

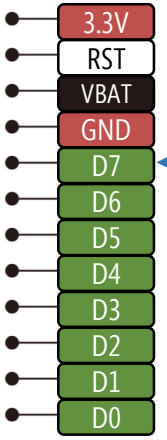
Input and output is with respect to the microcontroller board. Input is anything that the RedBear Duo reads in (*e.g.*, from a sensor) and output is anything that the RedBear Duo writes out (*e.g.*, to a light or motor)

Analog Input on A7

Reads in any any value between 0V or 3.3V using analogRead function. In this case, the value of a photocell

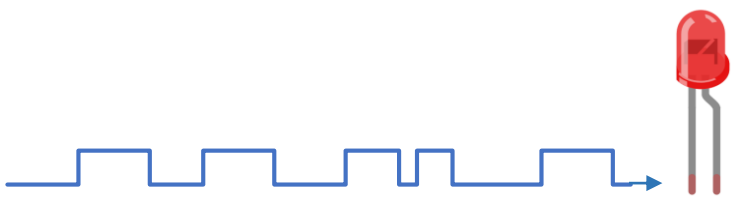






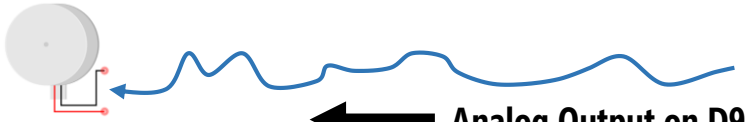
Digital Input on D7

Reads in a digital input signal (anything below 1.65V converted to LOW, anything above 1.65V converted to HIGH). In this case, the value of switch.



Digital Output on D0

Writes out 0V or 3.3V using digitalWrite function. In this case, turning on and off an LED.

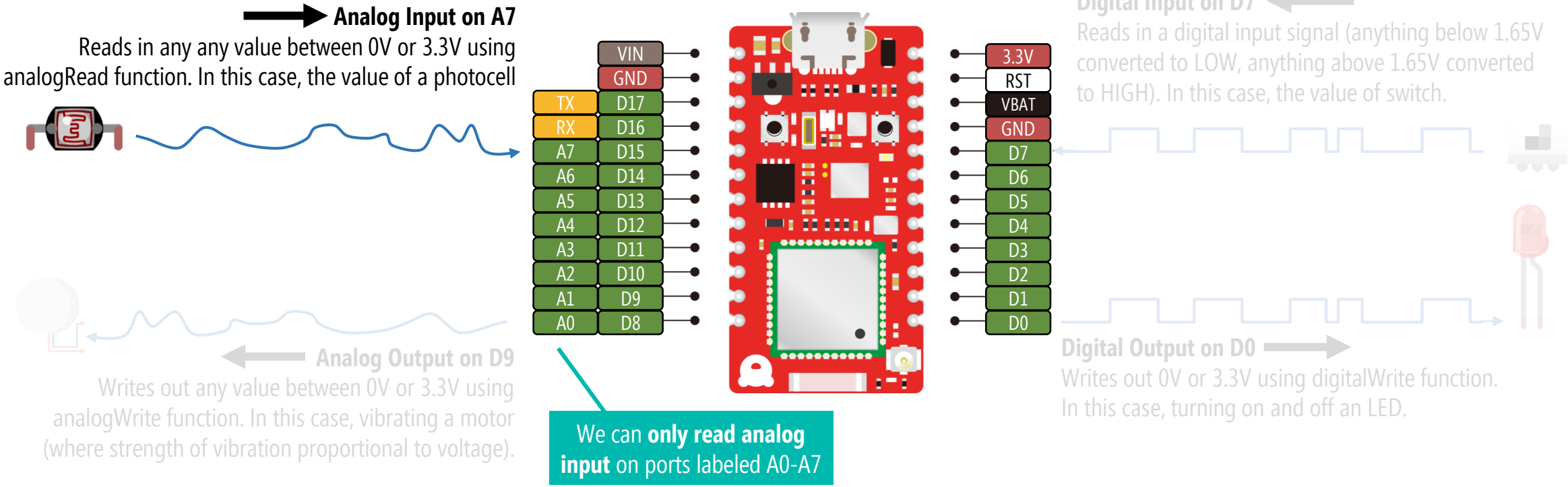


Analog Output on D9

Writes out any value between 0V or 3.3V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage).

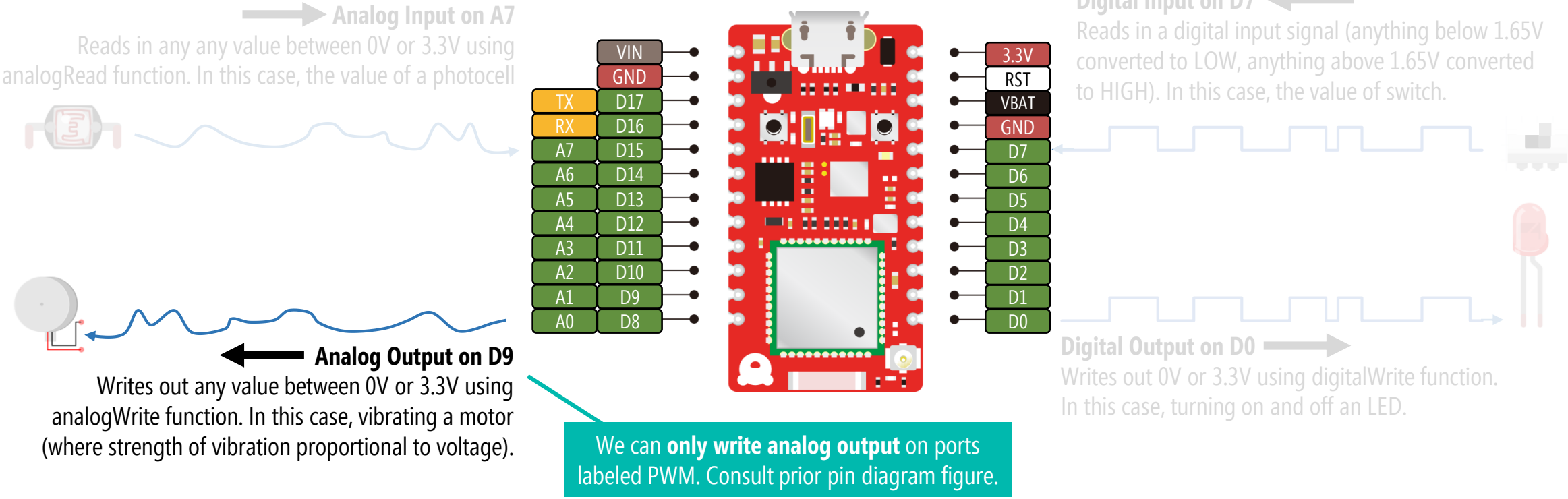
INPUT VS. OUTPUT

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



INPUT VS. OUTPUT

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



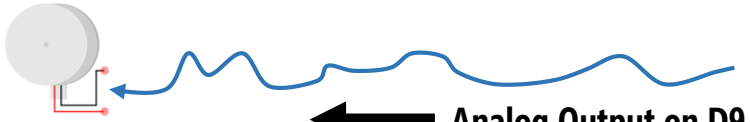
OUTPUT

DIGITAL AND ANALOG OUTPUT

We are going to start with covering digital and analog output

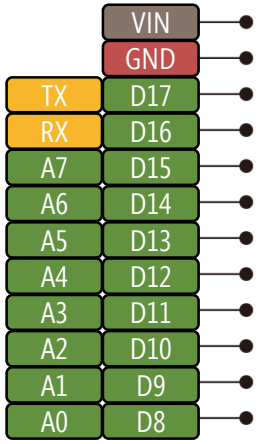
Analog Input on A7

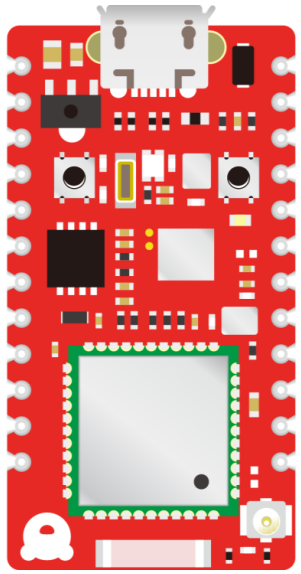
Reads in any any value between 0V or 3.3V using analogRead function. In this case, the value of a photocell

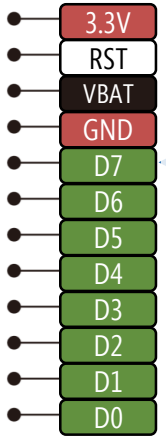


Analog Output on D9

Writes out any value between 0V or 3.3V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage).

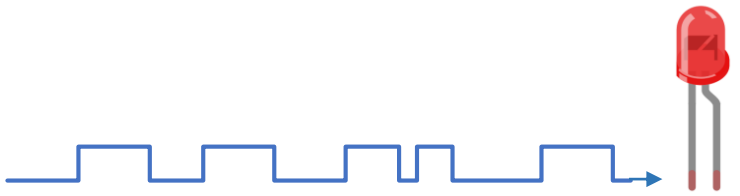






Digital Input on D7

Reads in a digital input signal (anything below 1.65V converted to LOW, anything above 1.65V converted to HIGH). In this case, the value of switch.



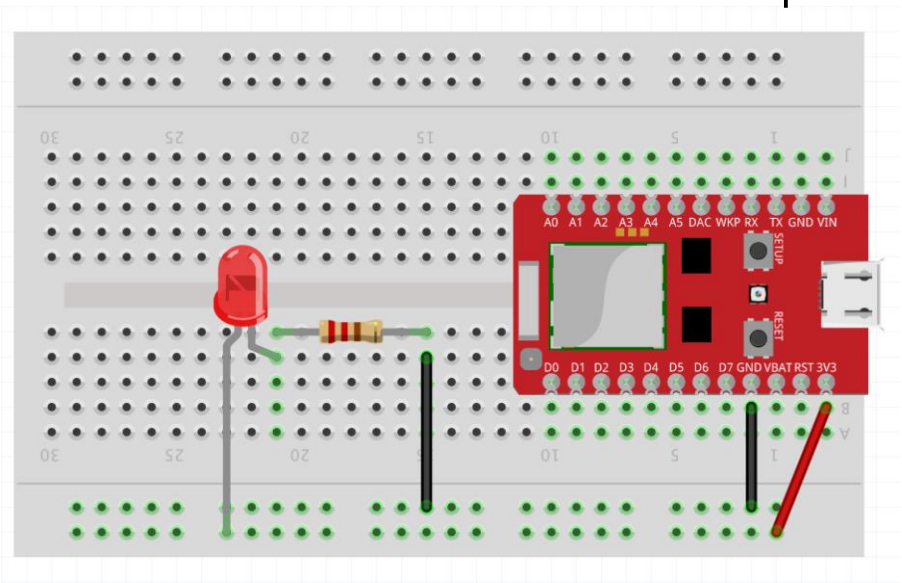
Digital Output on D0

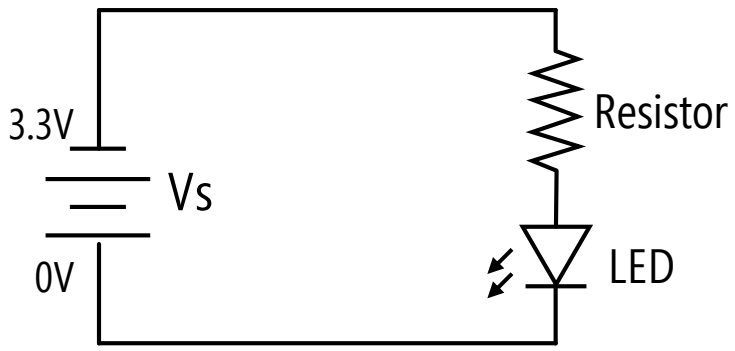
Writes out 0V or 3.3V using digitalWrite function. In this case, turning on and off an LED.

Writing our first Arduino program: **blinking** an LED on and off.
On and *off* implies digital output, so we will use digitalWrite.

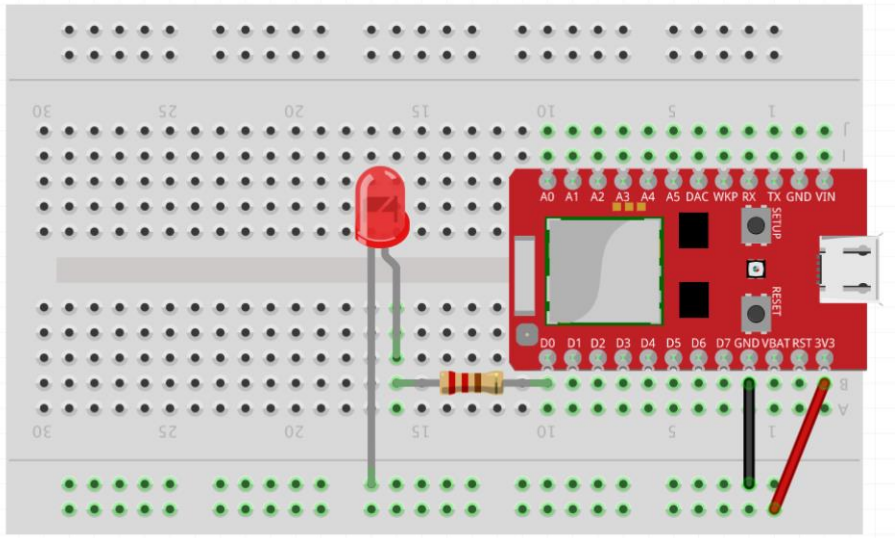
FLASH LED ON/OFF USING REDBEAR DUO

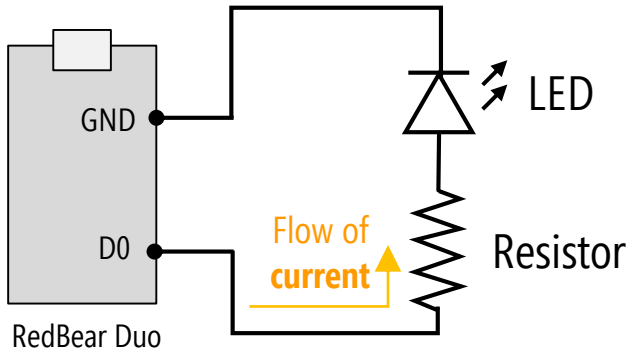
Previous Circuit: LED Powered via 3.3V pin





Build Circuit: Flash LED on/off via the D0 pin

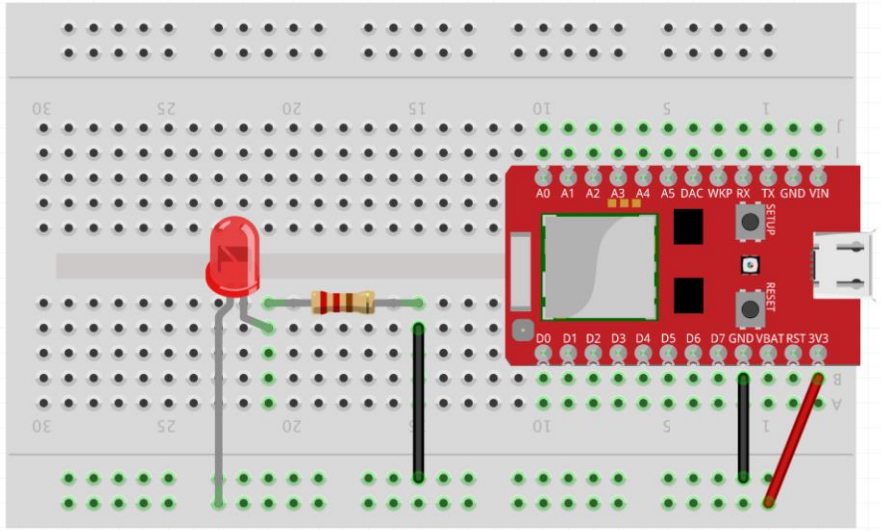




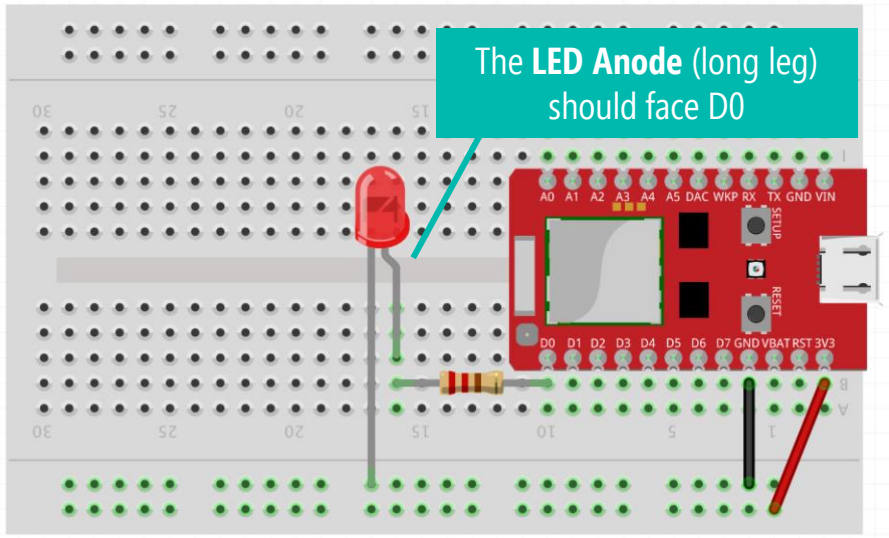
DIGITAL OUTPUT

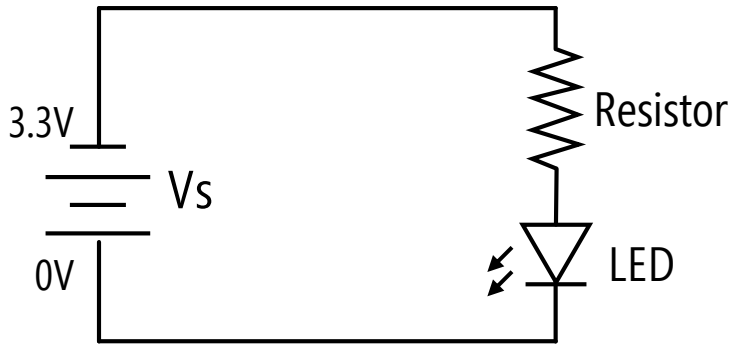
FLASH LED ON/OFF USING REDBEAR DUO

Previous Circuit: LED Powered via 3.3V pin



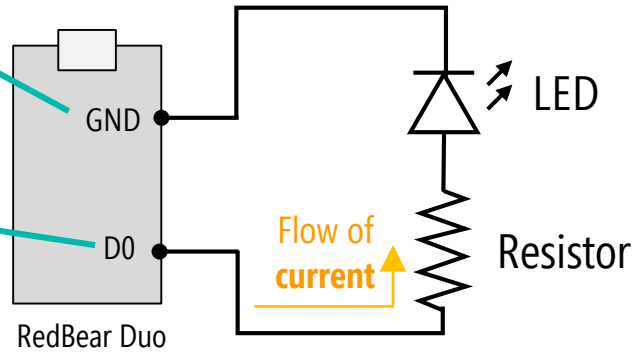
Build Circuit: Flash LED on/off via the D0 pin





GND is always the lowest potential voltage point in our circuit

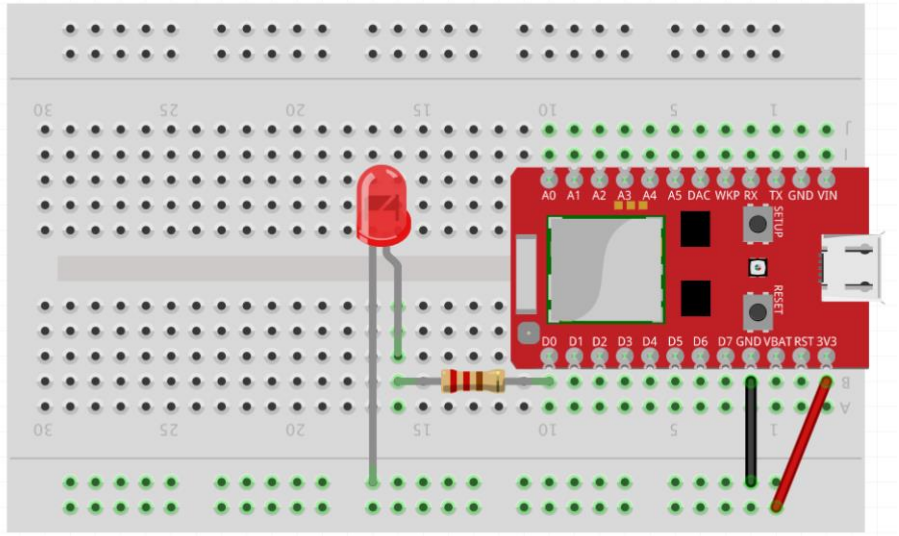
D0 will alternate between 3.3V and 0V. When 3.3V, the LED will turn on.

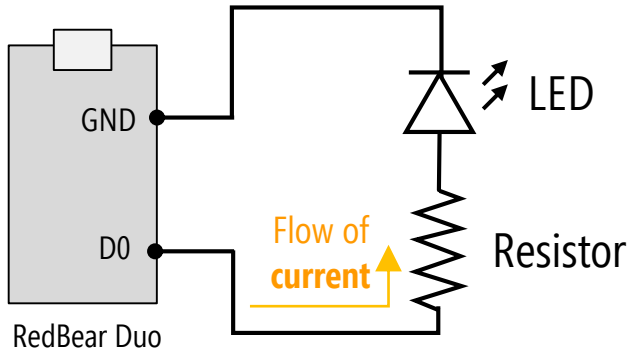


DIGITAL OUTPUT

FLASH LED ON/OFF USING REDBEAR DUO

Build Circuit: Flash LED on/off via the D0 pin





Write Code: Flash LED on/off via the D0 pin

RedBearDuoBlink | Arduino 1.8.5

RedBearDuoBlink

```
/*
 * IMPORTANT: When working with the RedBear Duo, you must have this line of
 * code at the top of your program. The default state is SYSTEM_MODE(AUTOMATIC);
 * however, this puts the RedBear Duo in a special cloud-based mode that we
 * are not using. For our purposes, set SYSTEM_MODE(SEMI_AUTOMATIC) or
 * SYSTEM_MODE(MANUAL). See https://docs.particle.io/reference/firmware/photon/#system-modes
 */
SYSTEM_MODE(MANUAL);

const int RED_LED_OUTPUT_PIN = D0;

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

void loop() {

  // Write out a HIGH value (3.3V in this case as the RedBear Duo is a 3.3V board)
  digitalWrite(RED_LED_OUTPUT_PIN, HIGH);

  // Delay 1000ms (or 1 s)
  delay(1000);

  // Write out a LOW value (0V in this case)
  digitalWrite(RED_LED_OUTPUT_PIN, LOW);

  // Wait another second
  delay(1000);
}
```

Done Saving.

https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoBlink

DIGITAL OUTPUT: PINMODE()

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

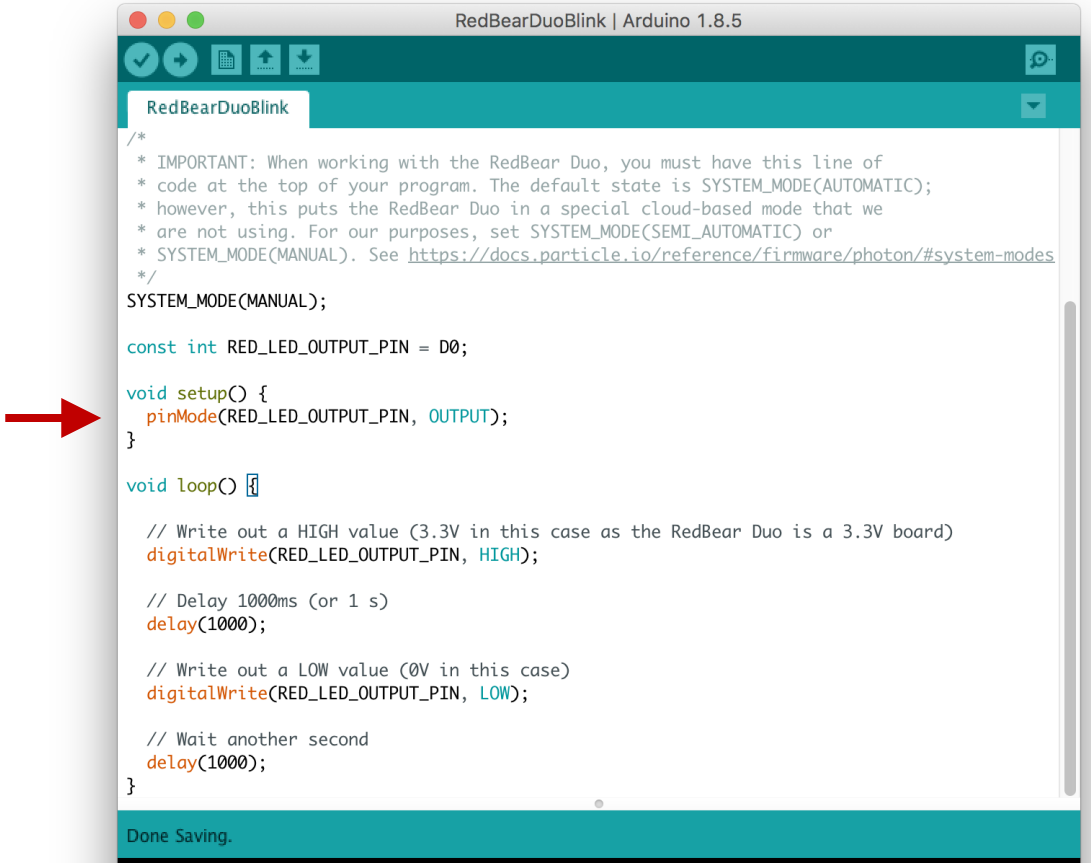
Syntax

`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()

Writes a HIGH or LOW value to a digital pin

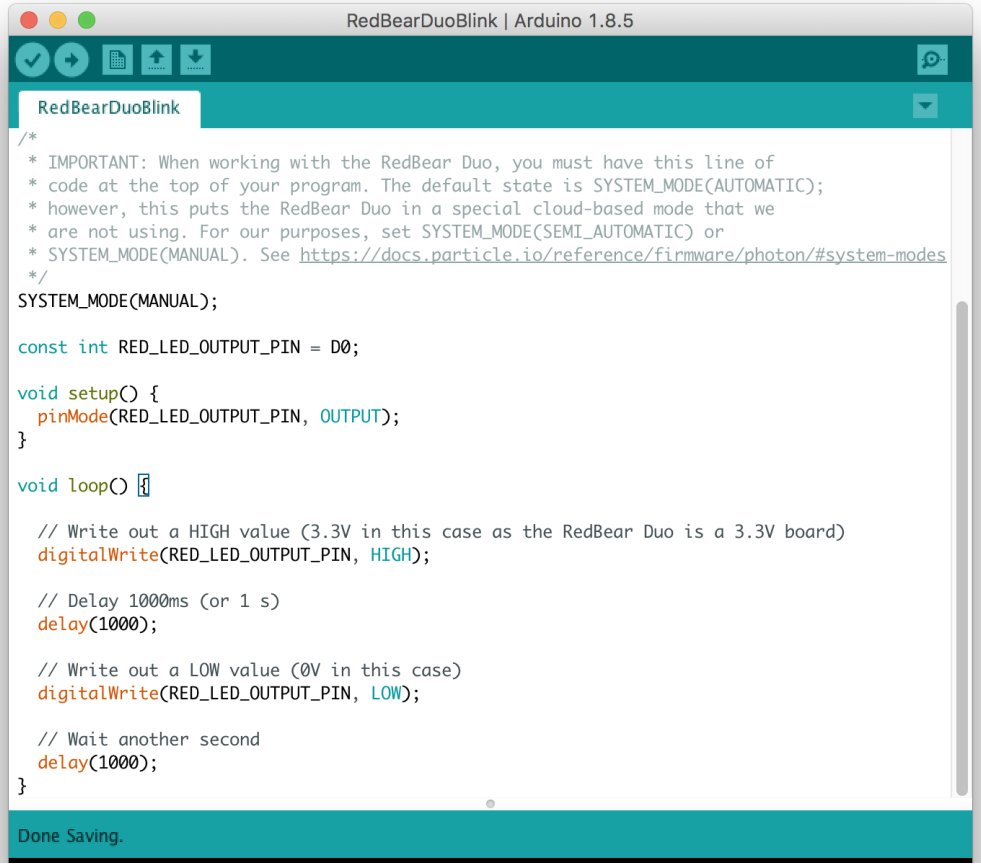
Syntax

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

Parameters

`pin`: the pin number

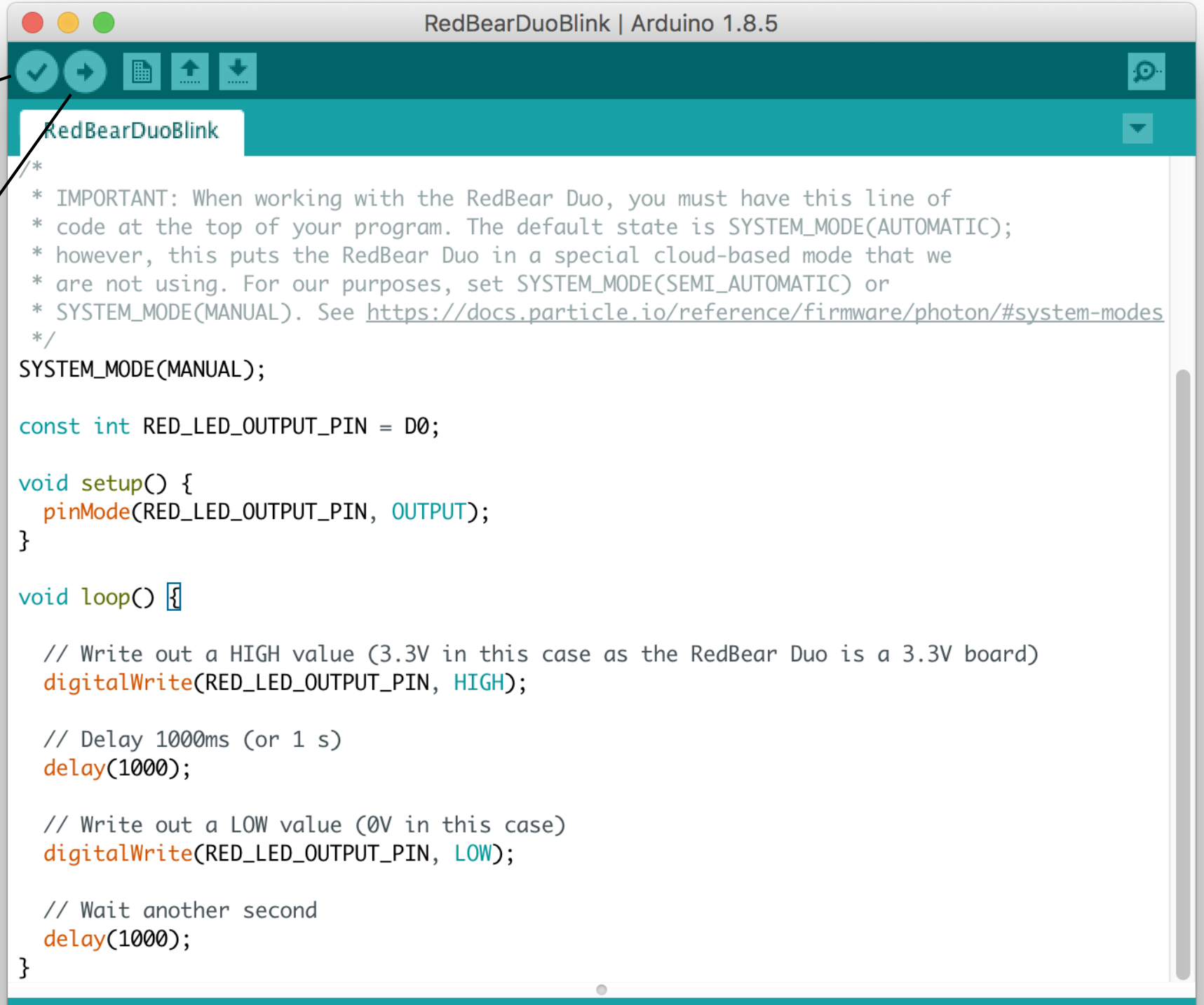
`value`: HIGH (3.3V) or LOW (0V)

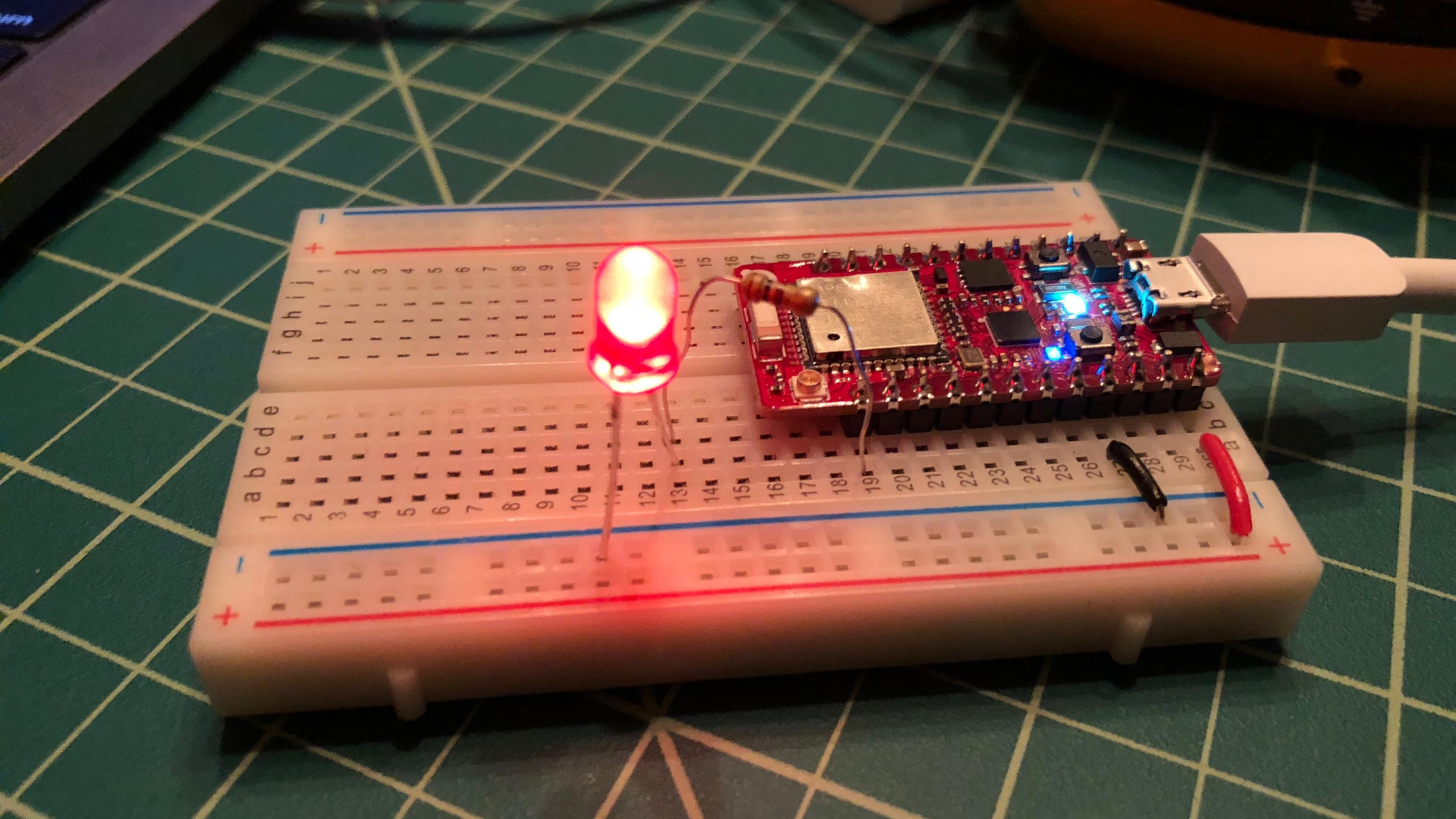


https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

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

Upload your program to the Arduino





QUICK TIP: AVOID USING CONSTANTS IN CODE

FadeExample | Arduino 1.8.5

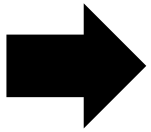
File Edit Sketch Tools Help

FadeExample §

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

Done Saving.

10 Arduino/Genuino Uno on COM1



FadeExample | Arduino 1.8.5

File Edit Sketch Tools Help

FadeExample §

```
const int RED_LED_OUTPUT_PIN = 13;

void setup() {
  // put your setup code here, to run once:
  pinMode(RED_LED_OUTPUT_PIN, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(RED_LED_OUTPUT_PIN, HIGH);
  delay(500);
  digitalWrite(RED_LED_OUTPUT_PIN, LOW);
  delay(500);
}
```

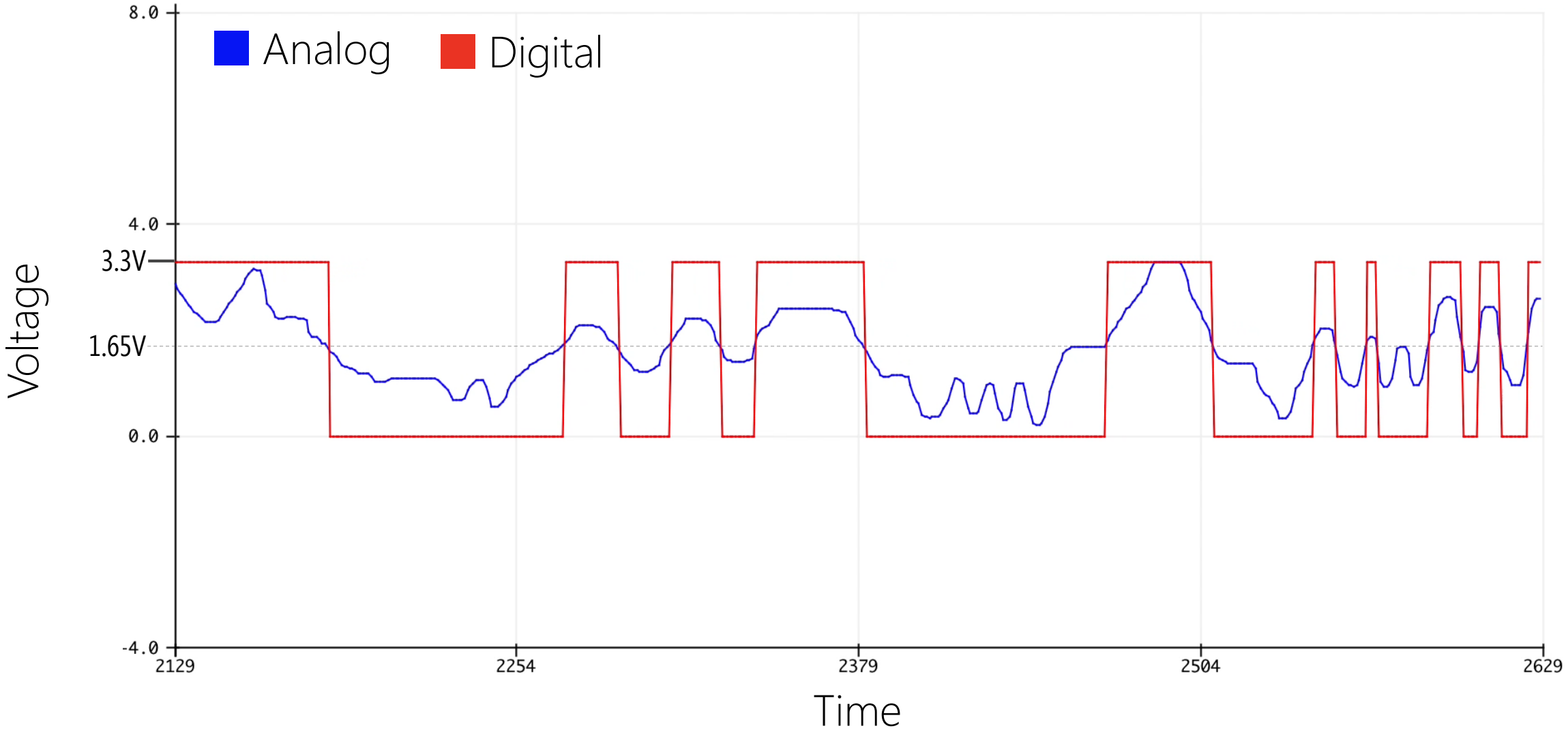
Done Saving.

1 Arduino/Genuino Uno on COM1

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

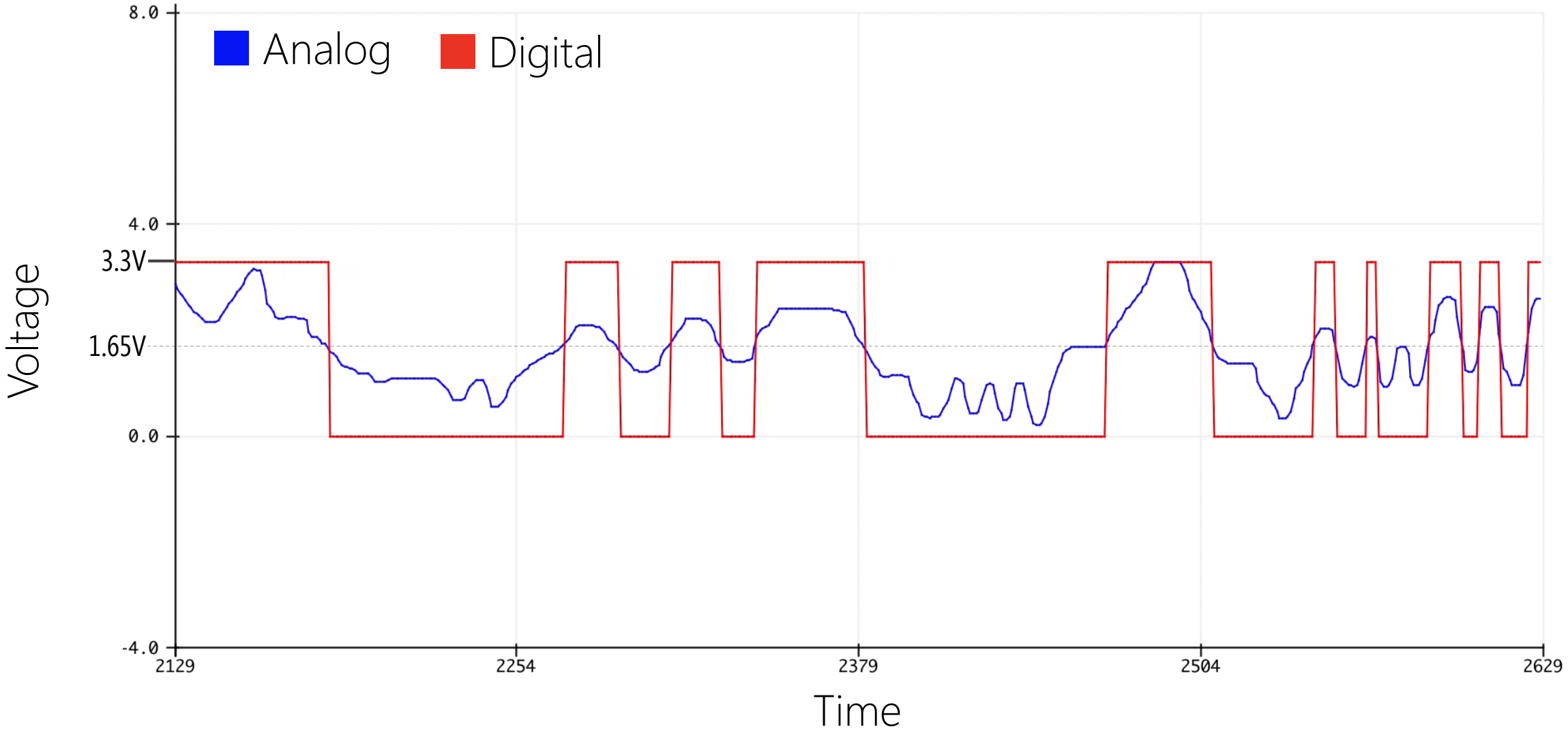
ANALOG VS. DIGITAL

digitalRead and digitalWrite can only be LOW or HIGH corresponding to 0V and 3.3V.



ANALOG VS. DIGITAL

digitalRead and digitalWrite can only be LOW or HIGH corresponding to 0V and 3.3V.

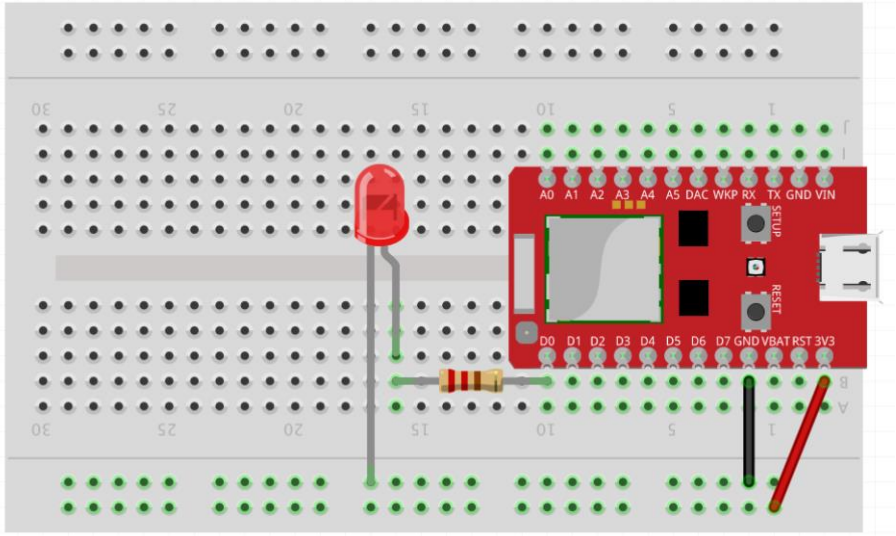


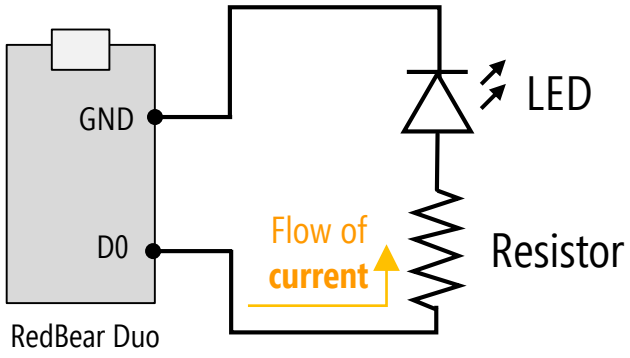
ANALOG OUTPUT

FADE LED ON/OFF USING REDBEAR DUO

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

Circuit: Fade LED on/off via the D0 pin





Write Code: Fade LED on/off via the D0 pin

RedBearDuoFade | Arduino 1.8.5

RedBearDuoFade

```
SYSTEM_MODE(MANUAL);

const int LED_OUTPUT_PIN = D0;

int _brightness = 0;    // how bright the LED is
int _fadeAmount = 5;    // the amount to fade the LED by on each step
int _maxBrightness = 128; // the max allowable analogWrite value is 255

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

void loop() {

  // set the brightness of pin D0:
  analogWrite(LED_OUTPUT_PIN, _brightness);

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

  // reverse the direction of the fading at the ends of the fade:
  if (_brightness <= 0 || _brightness >= _maxBrightness) {
    _fadeAmount = -_fadeAmount;
  }

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

Done uploading.

https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoFade

ANALOG OUTPUT: ANALOGWRITE()

Writes an analog value between 0 and 3.3V to a pin using pulse-width modulation (PWM).

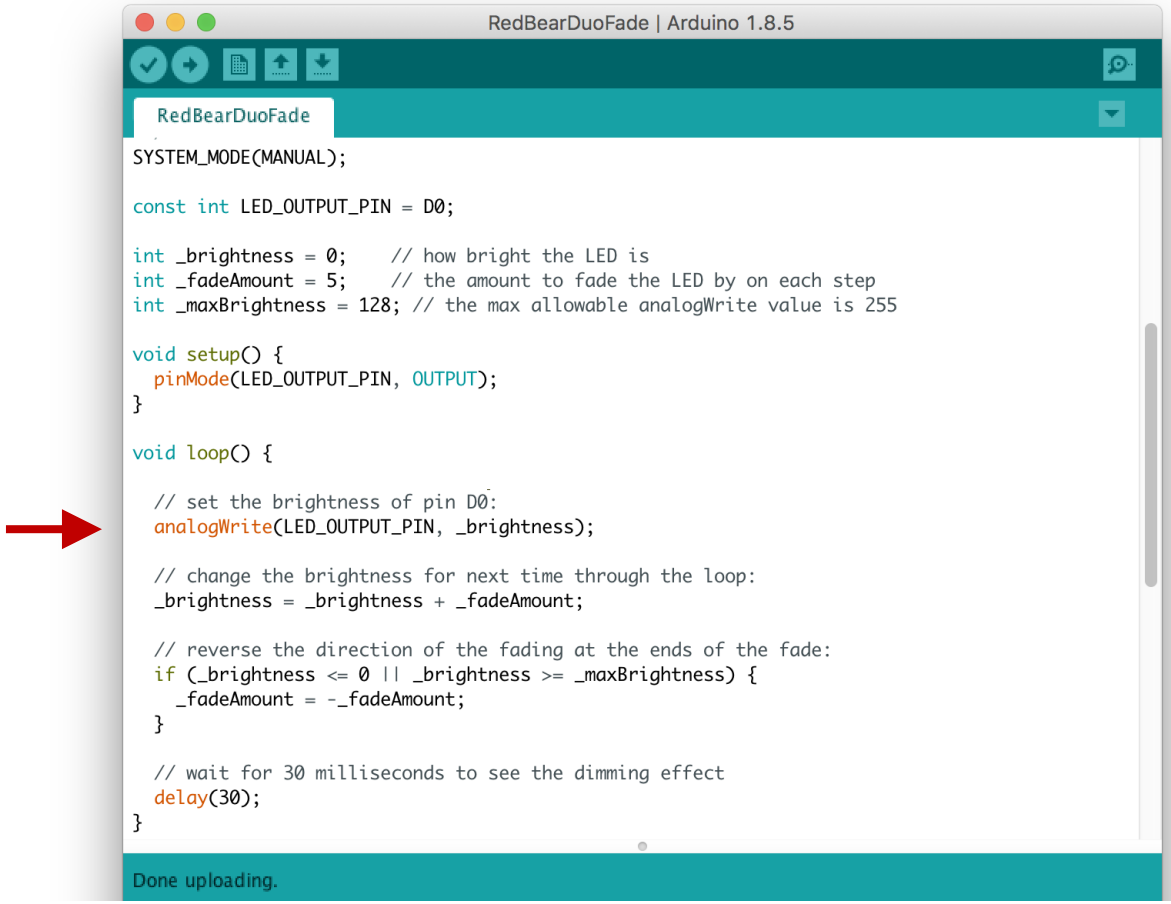
Syntax

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

Parameters

`pin`: the pin to write to.

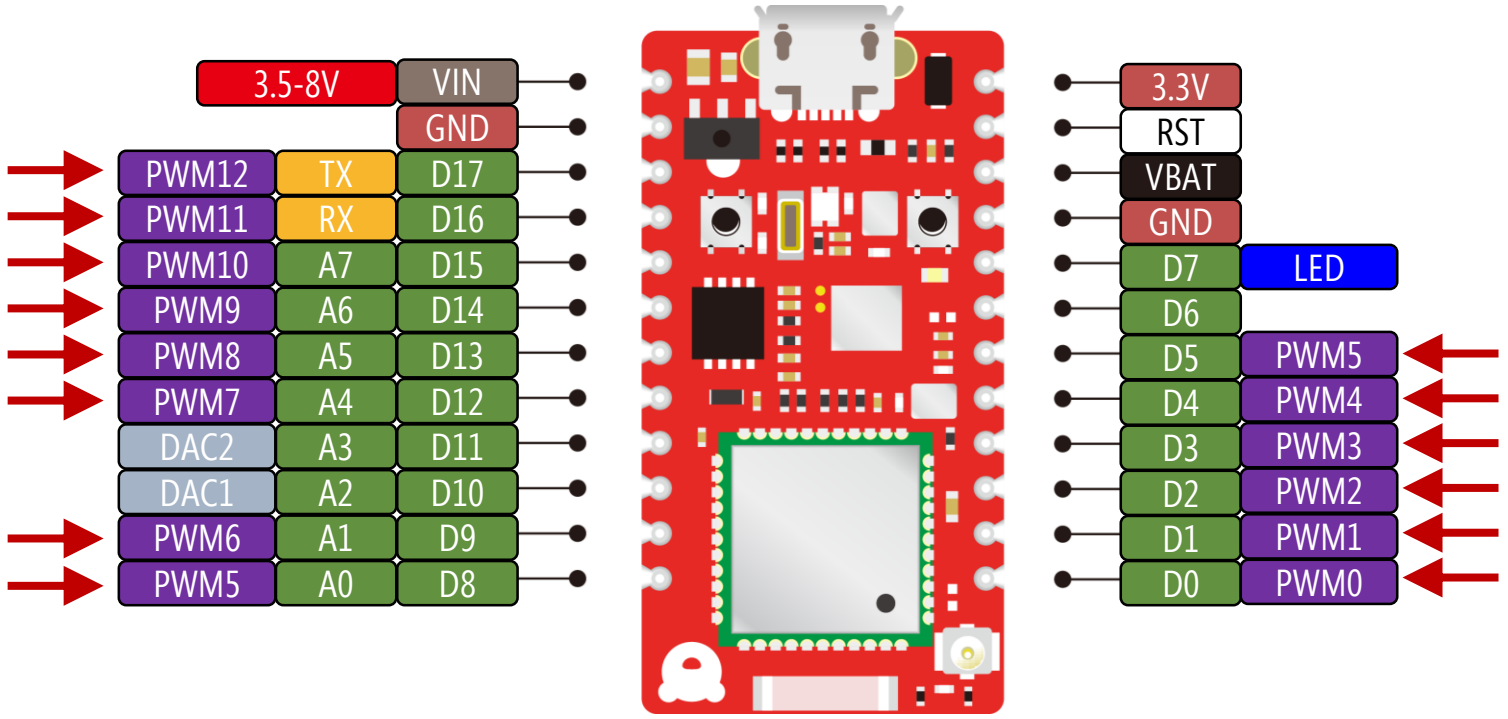
`value`: an integer value between 0 & 255 which roughly maps to 0 – 3.3V on the RedBear Duo.



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

ANALOG OUTPUT: PULSE WIDTH MODULATION

You can only use analogWrite on those pins marked with PWM.

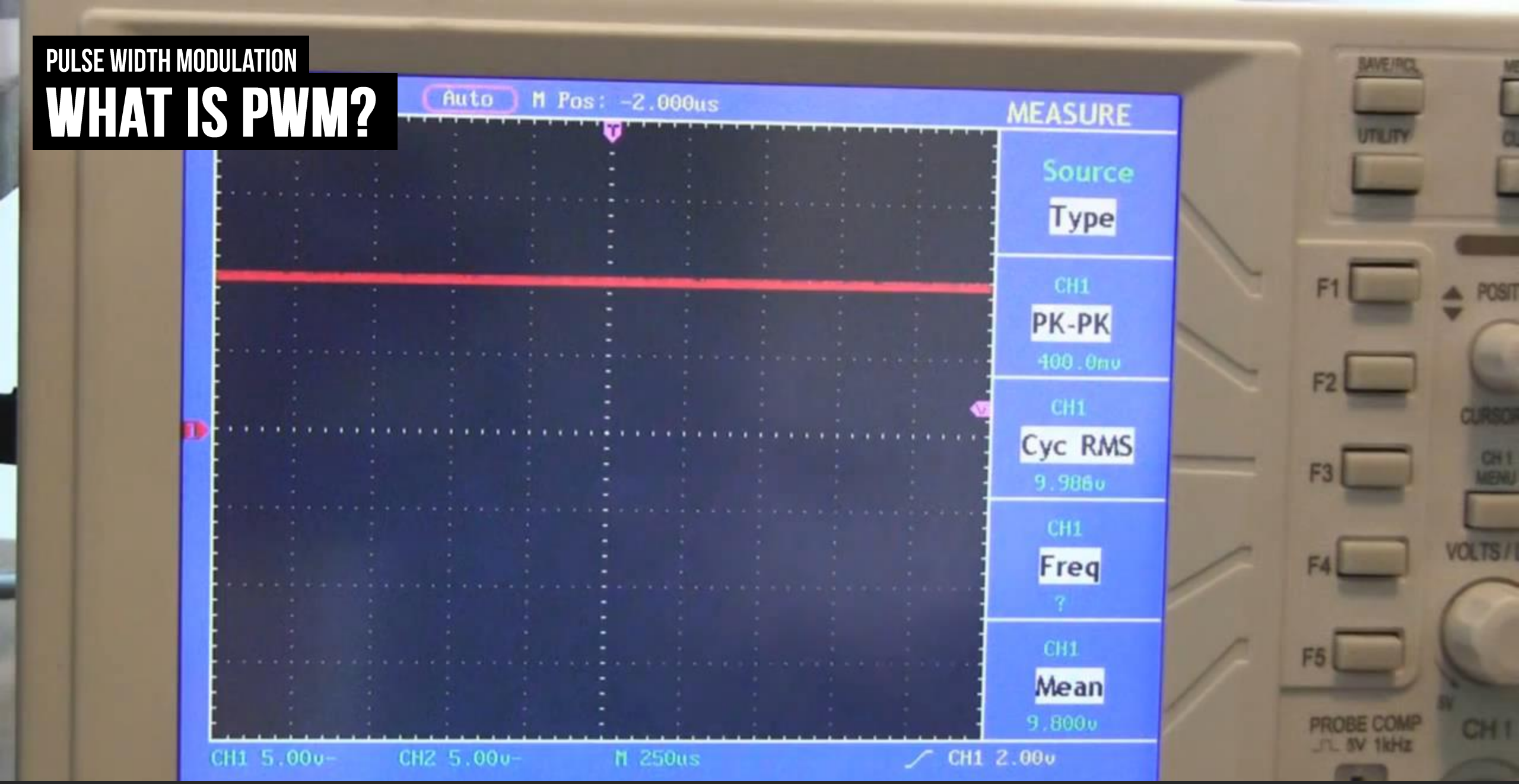


Legend

AX	Analog Input Pin
DX	Digital Input/output Pin
PWM	Supports Analog Out via Pulse Width Modulation

PULSE WIDTH MODULATION

WHAT IS PWM?

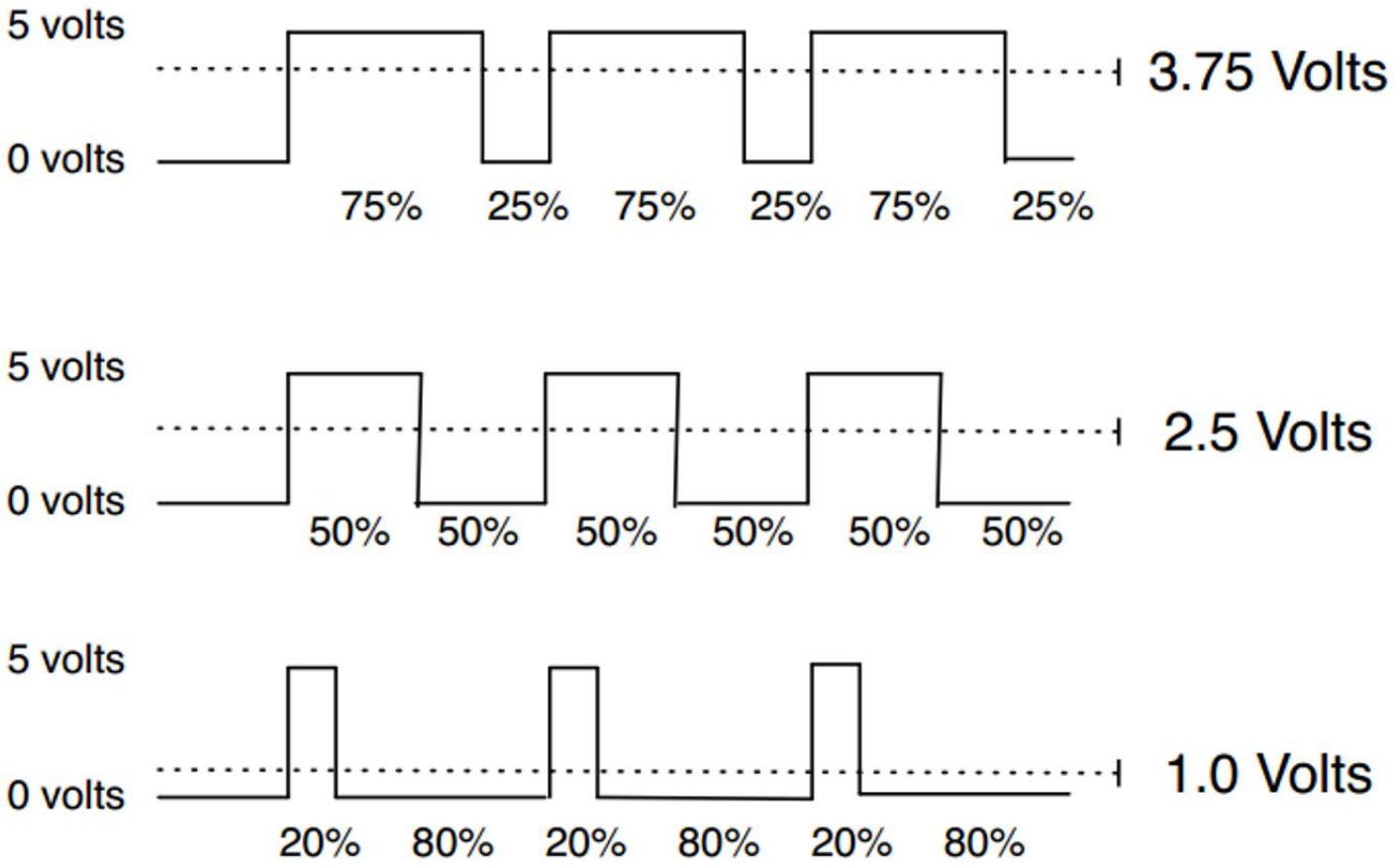


Source: Afrotechmods, https://youtu.be/YmPziPfaByw

ANALOG OUTPUT

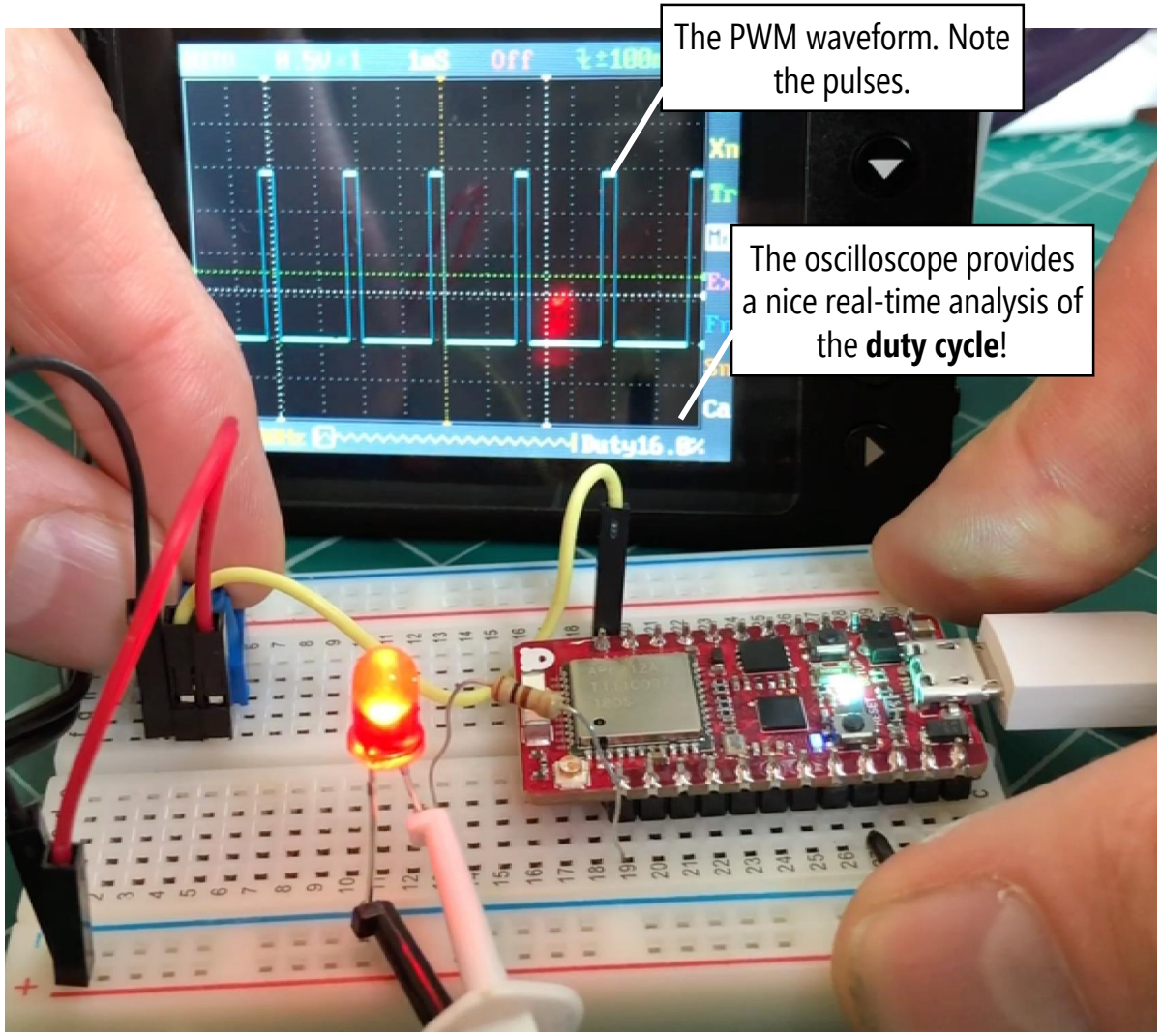
PULSE WIDTH MODULATION

The voltage is max_voltage * duty_cycle_val



ANALOG OUTPUT

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE

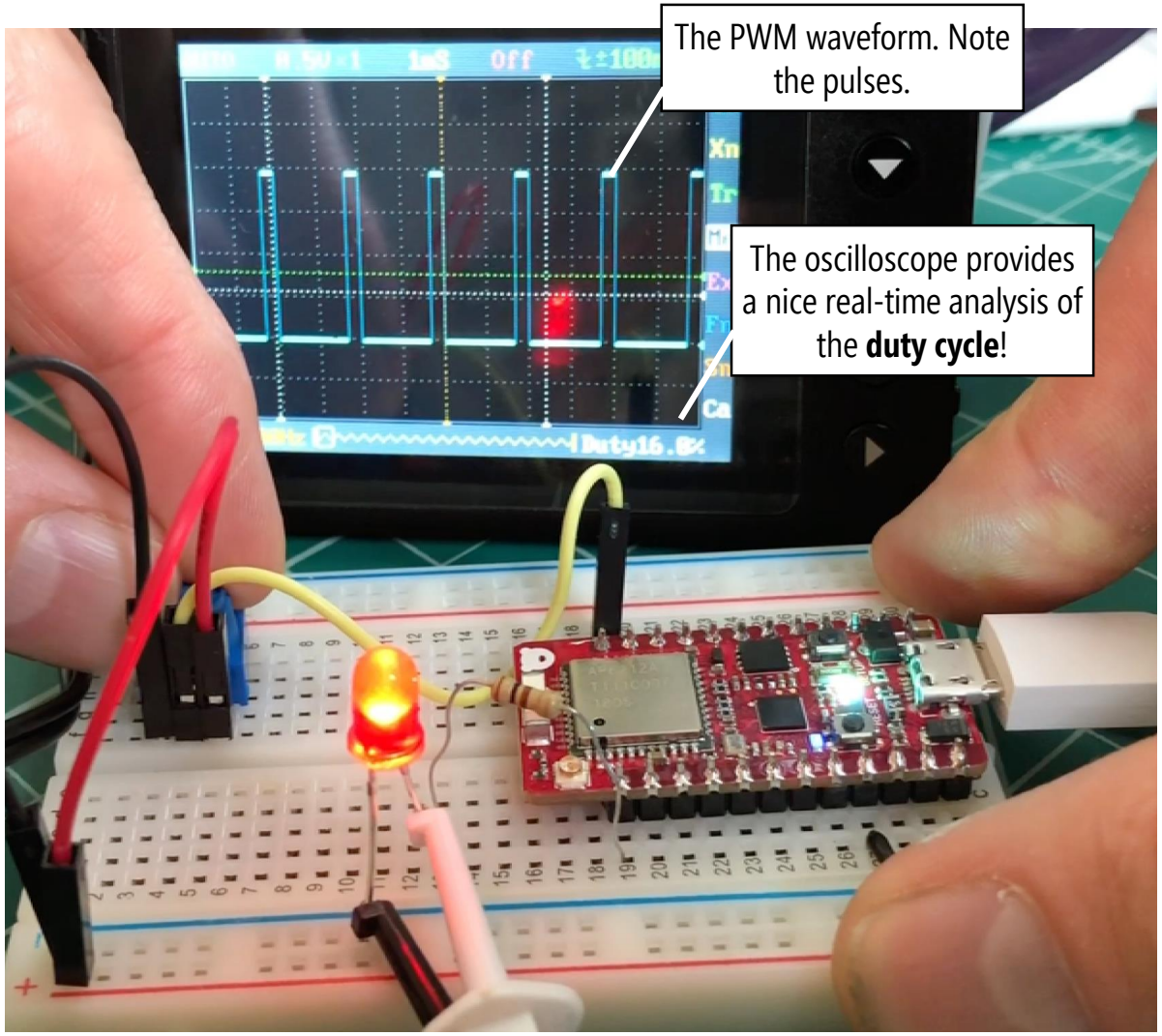


RedBearDuoReadPotSetLED | Arduino 1.8.5

RedBearDuoReadPotSetLED

```
SYSTEM_MODE(MANUAL);

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

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

void loop() {

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

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

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

  delay(100);
}
```

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE



RedBearDuoReadPotSetLED | Arduino 1.8.5

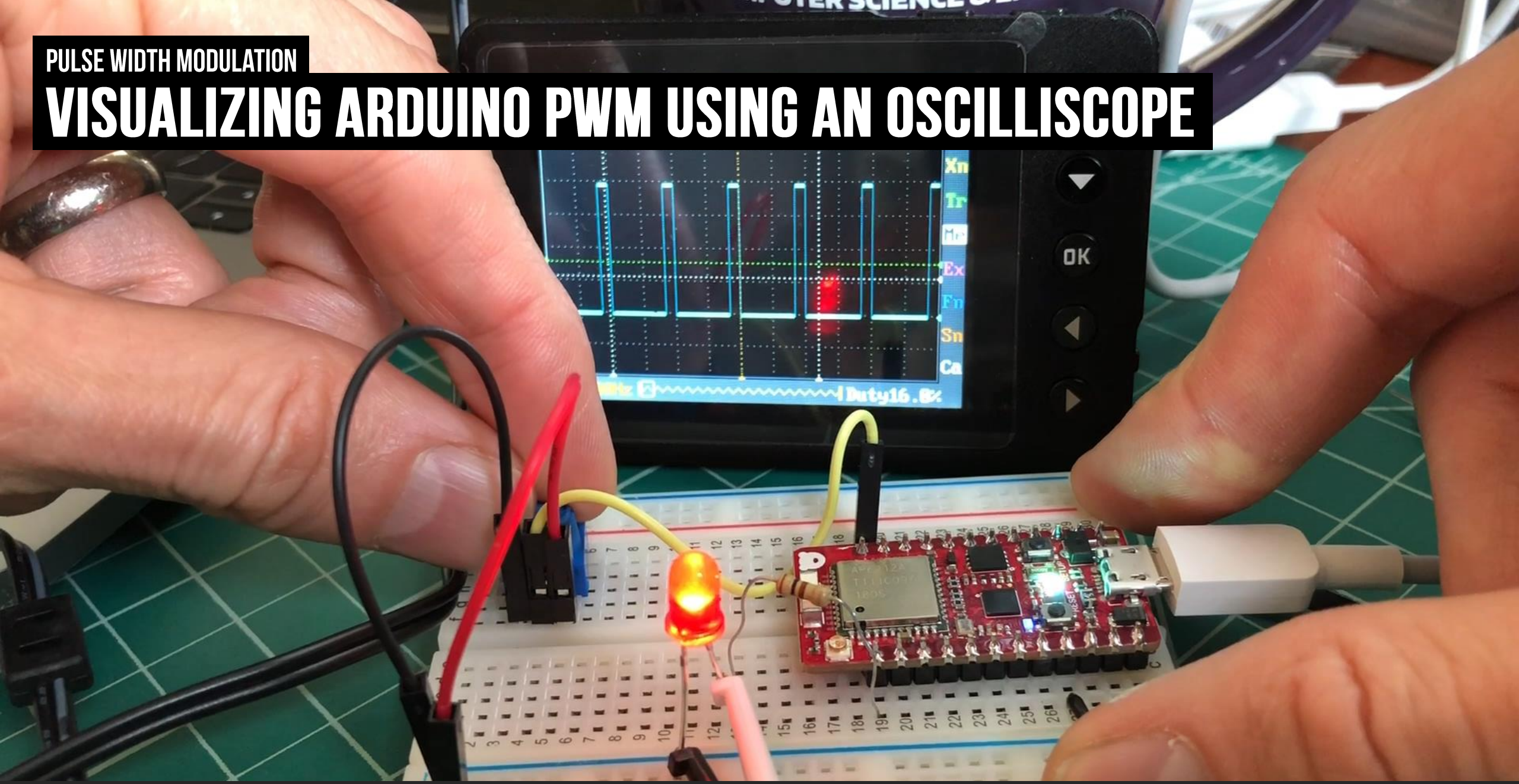
RedBearDuoReadPotSetLED

```
SYSTEM_MODE(MANUAL);

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

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

void loop() {

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

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

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

  delay(100);
}
```

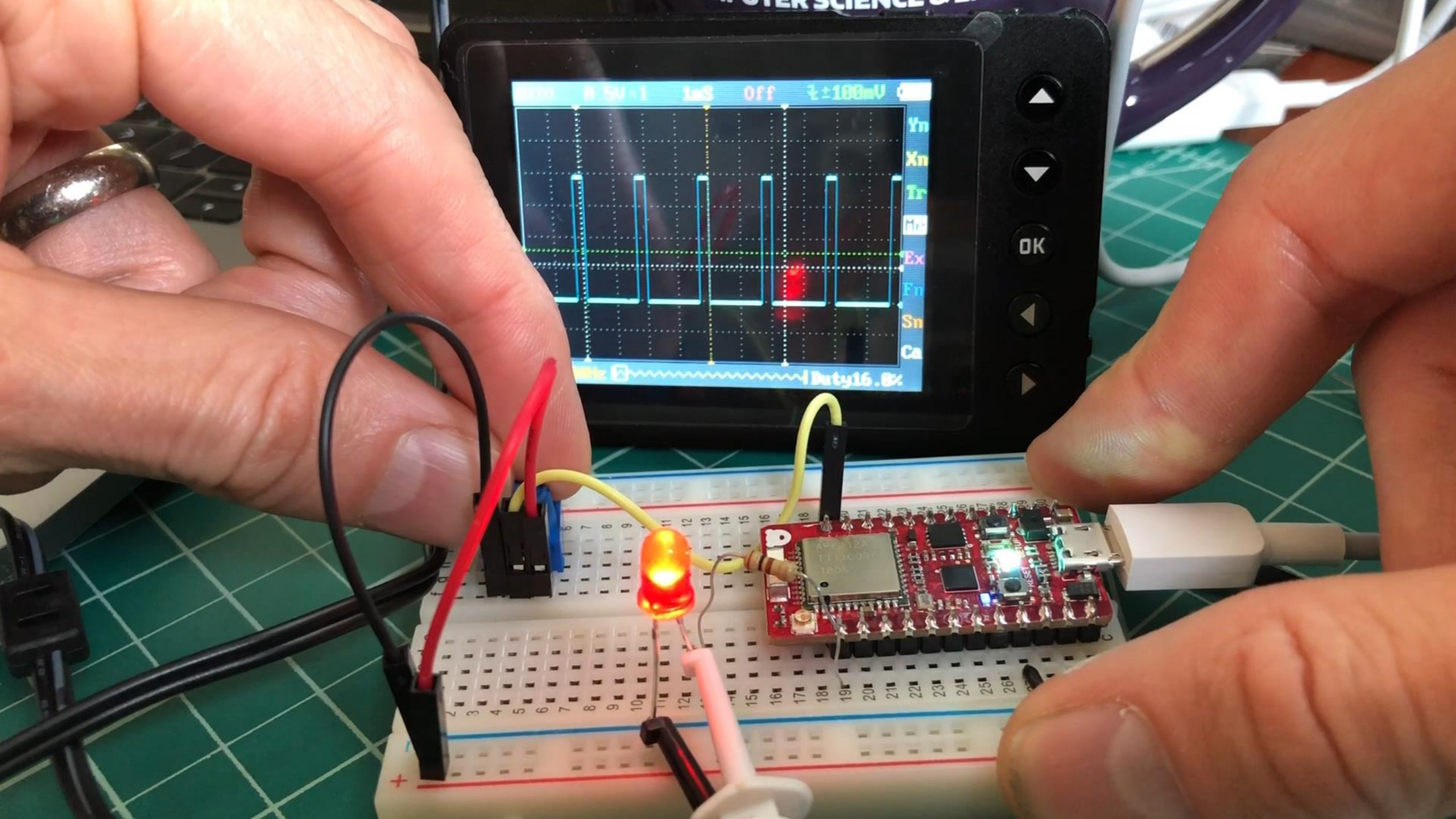
The **analogWrite call sets the duty cycle** of the PWM waveform. Note: you cannot set the PWM frequency. See the video on the next slide

PULSE WIDTH MODULATION

VISUALIZING ARDUINO PWM USING AN OSCILLISCOPE



Source: video by Jon Froehlich


OK

Phew, OK, we get it! **Arduino does not actually write out a truly analog signal**. Instead, it "fakes" it by using PWM. When we use analogWrite, we are changing the duty cycle but the waveform frequency is fixed at ~500Hz (see video!). Now that we've got that covered, you don't really need to think about this distinction anymore—at least not for A3.

PULSE WIDTH MODULATION

ANOTHER GREAT VIDEO ON PWM, DUTY CYCLES, & FREQUENCY

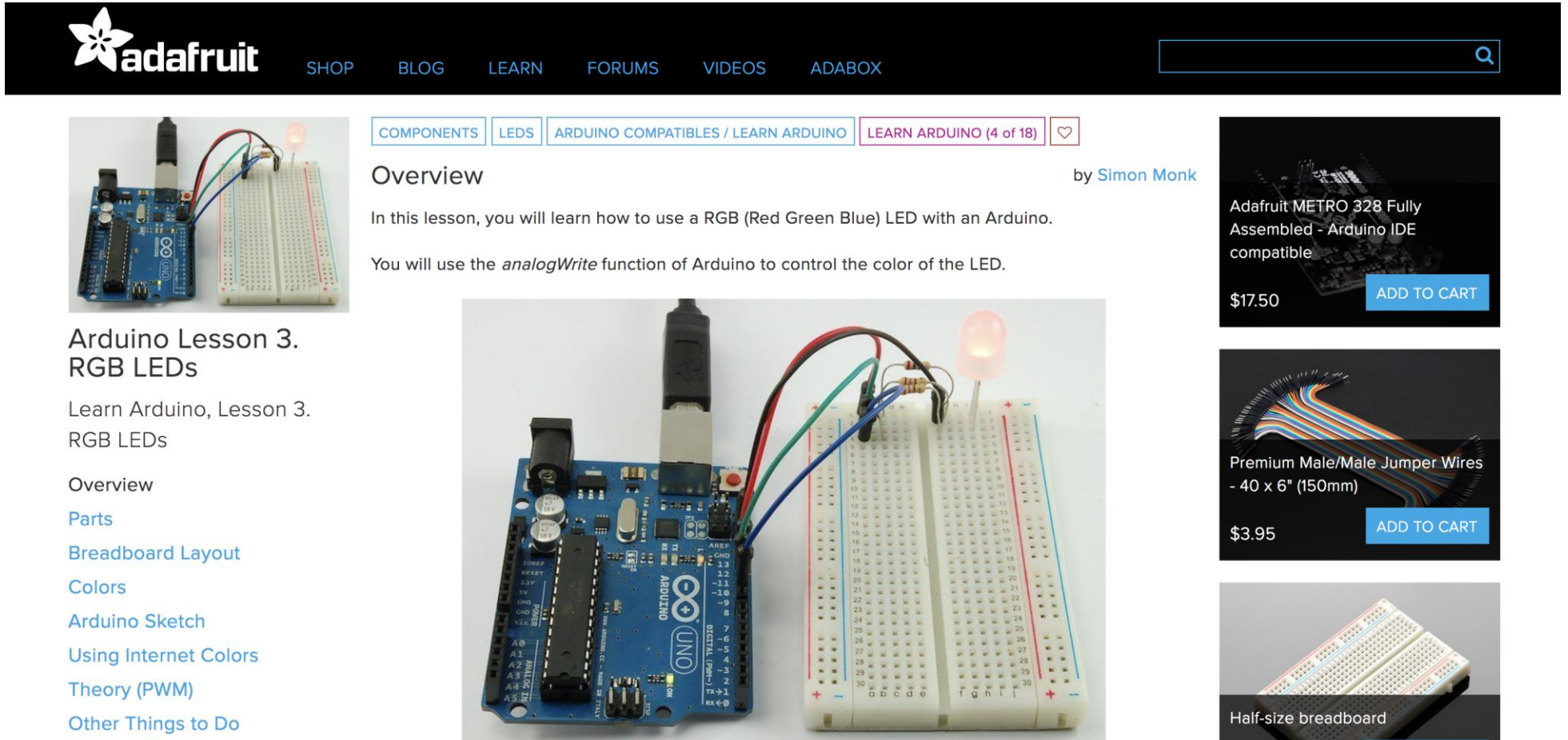
Source: https://youtu.be/rBQVfCUuhfs

Let's switch things up. We've been using a RED LED for all of the examples so far but of course there is a ton of different kinds of output (sound, vibration, motors, *etc.*). Let's stick with light for now and **play around with an RGB LED.**

ANALOG OUTPUT

USING AN RGB LED

I've already pointed you to the fantastic series of Adafruit Arduino tutorials, including this one on RGB LEDs. But I'm going to show you how to use the particular RGB LED in our kits with the RedBear Duo. Note: as with other material covered in class, please feel free to Google around for tutorials, answers to questions, and please post questions on Canvas! ☺



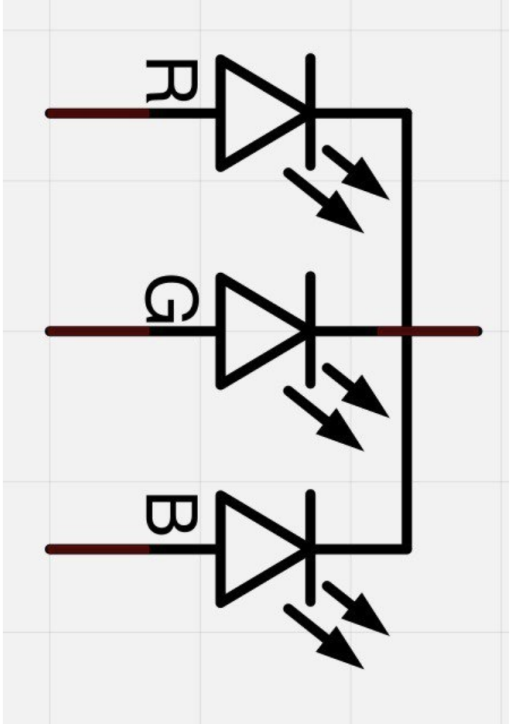
ANALOG OUTPUT

USING AN RGB LED: FROM ADAFRUIT TUTORIAL

Breadboard Layout

by Simon Monk

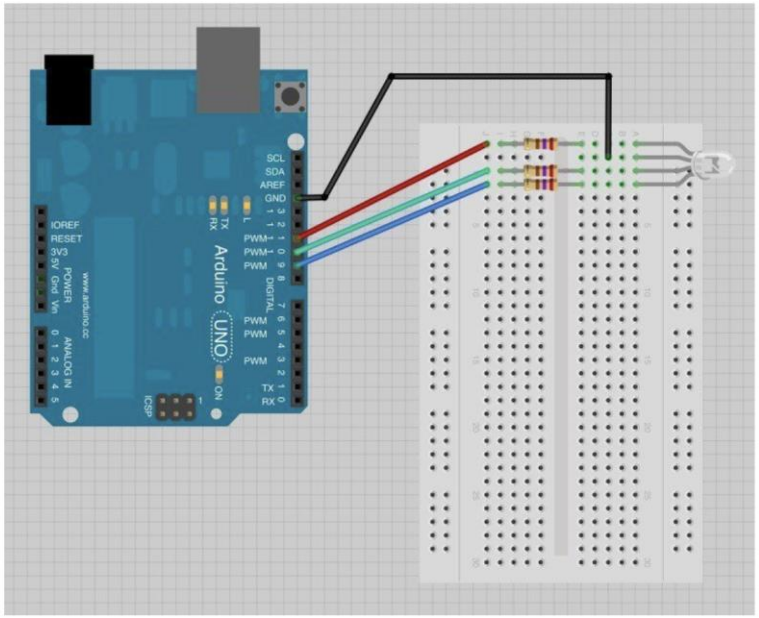
The RGB LED has four leads. There is one lead going to the positive connection of each of the single LEDs within the package and a single lead that is connected to all three negative sides of the LEDs.



The common negative connection of the LED package is the second pin from the flat side of the LED package. It is also the longest of the four leads. This lead will be connected to ground.

Each LED inside the package requires its own 270Ω resistor to prevent too much current flowing through it. The three positive leads of the LEDs (one red, one green and one blue) are connected to Arduino output pins using these resistors.

If you are using a common ANODE LED instead of common CATHODE, connect the long pin to +5 instead of ground



Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

1 ADD TO CART

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

ADD TO WISHLIST

Diffused RGB (tri-color) LED - Common Anode

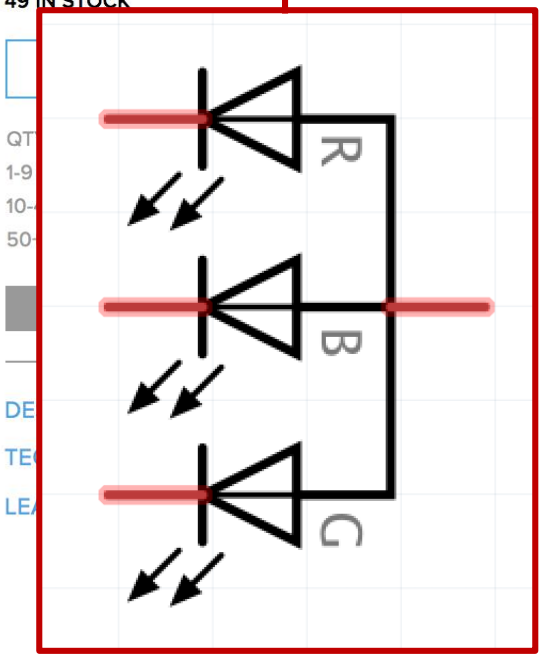
PRODUCT ID: 159

$2.00

49 IN STOCK

QT
1-9
10-
50-

DE
TE
LE

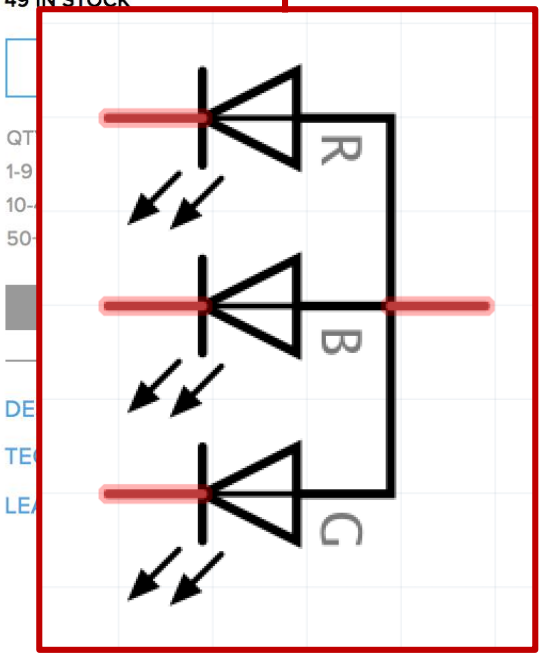


Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

$2.00

49 IN STOCK

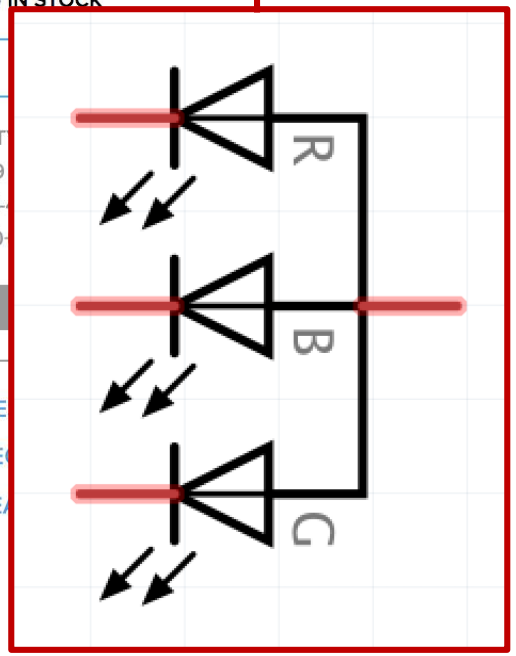


Diffused RGB (tri-color) LED - Common Anode

PRODUCT ID: 159

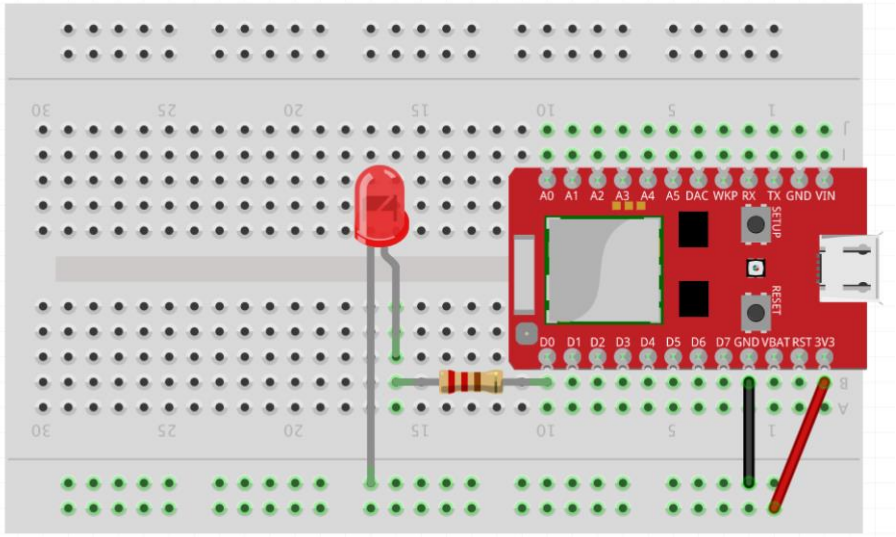
$2.00

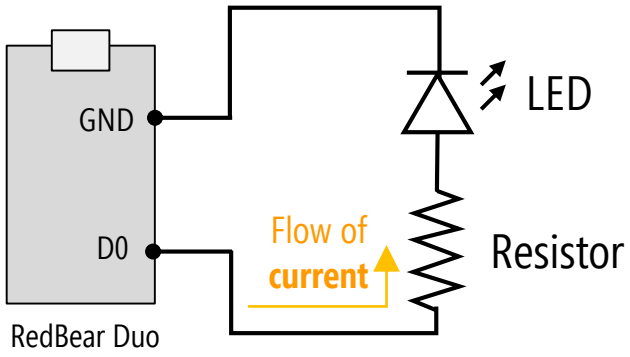
49 IN STOCK



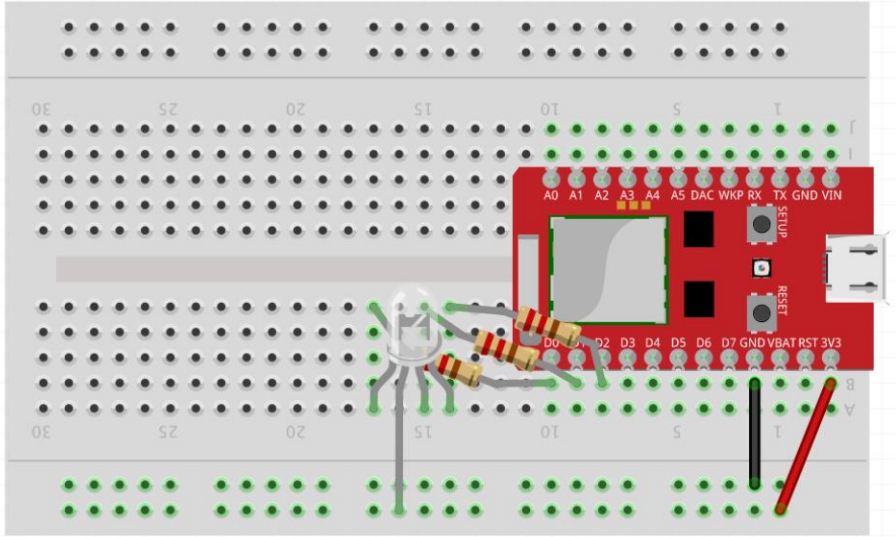
USING AN RGB LED: THE CIRCUIT

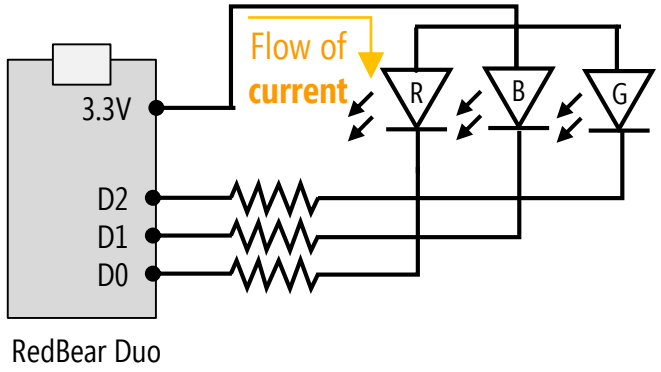
Old Circuit: Fade LED on/off via the D0 pin





New Circuit: Set R, B, G values via D0, D1, & D2

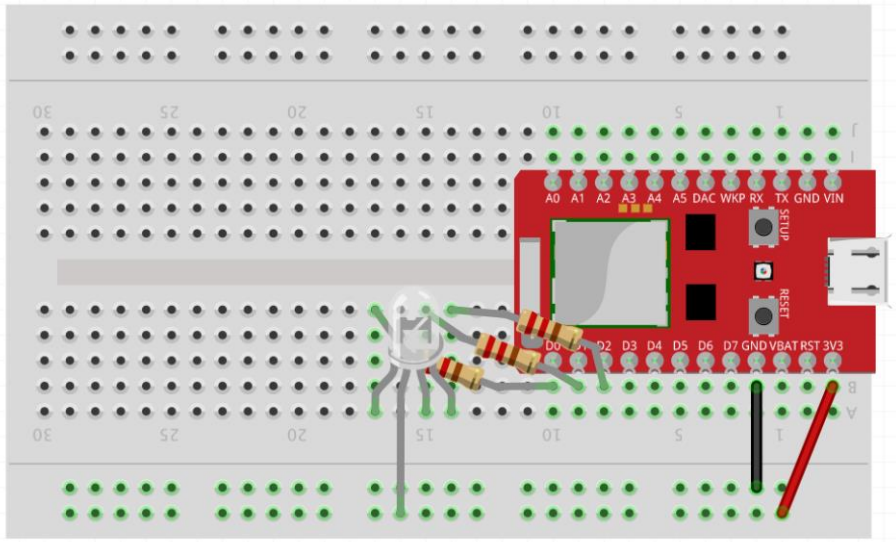


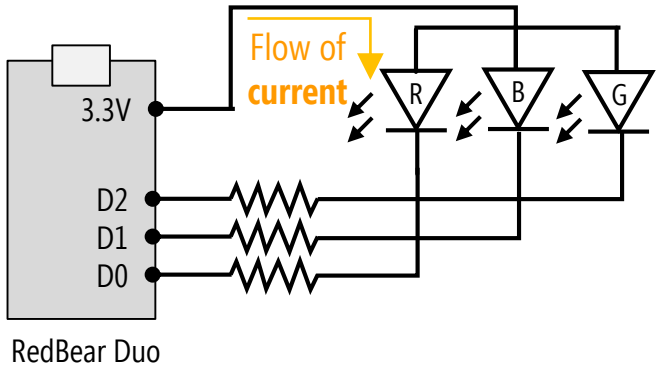


ANALOG OUTPUT

USING AN RGB LED: THE CODE

Circuit: Set R, B, G values via D0, D1, & D2

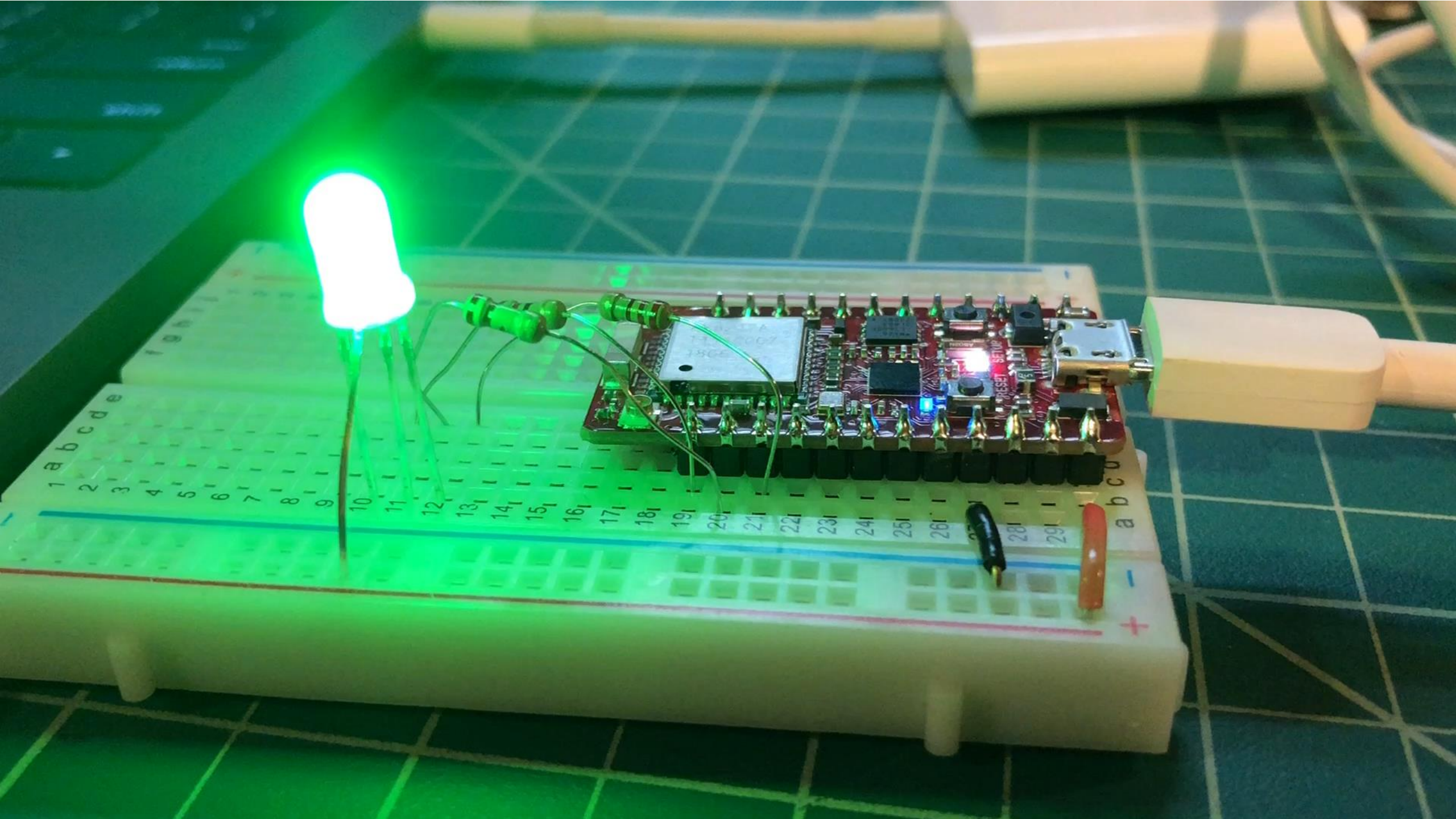




RedBearDuoRGB §

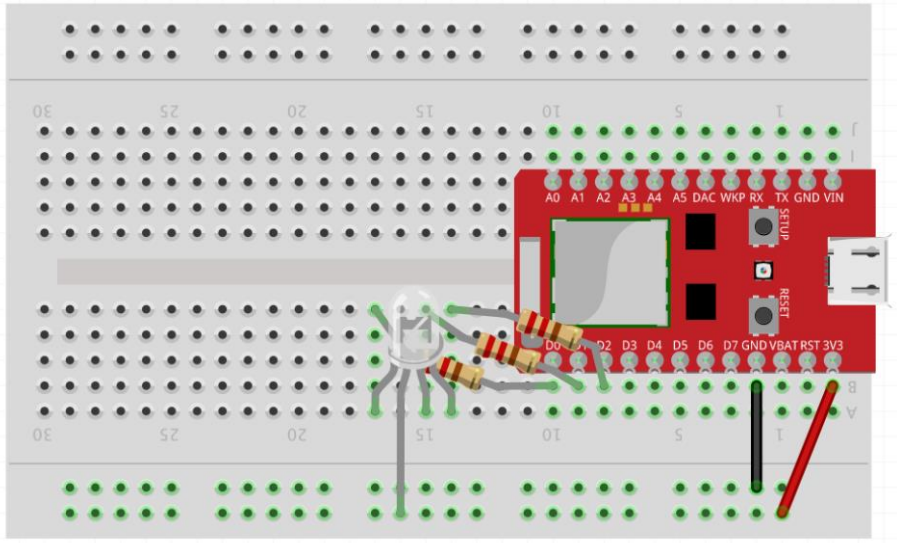
```
#define COMMON_ANODE 1

const int RGB_RED_PIN = D0;
const int RGB_GREEN_PIN  = D1;
const int RGB_BLUE_PIN  = D2;
const int DELAY = 1000; // delay between changing colors

void setup() {
  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() {

  Serial.println("Color=Red");
  setColor(255, 0, 0);  // red
  delay(DELAY);

  Serial.println("Color=Green");
  setColor(0, 255, 0);  // green
  delay(DELAY);

  Serial.println("Color=Blue");
  setColor(0, 0, 255);  // blue
  delay(DELAY);

  Serial.println("Color=Yellow");
  setColor(255, 255, 0);  // yellow
  delay(DELAY);

  Serial.println("Color=Purple");
  setColor(80, 0, 80);  // purple
  delay(DELAY);

  Serial.println("Color=Aqua");
  setColor(0, 255, 255);  // aqua
  delay(DELAY);
}

void setColor(int red, int green, int blue)
{
  #ifdef COMMON_ANODE
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
  #endif
  analogWrite(RGB_RED_PIN, red);
  analogWrite(RGB_GREEN_PIN, green);
  analogWrite(RGB_BLUE_PIN, blue);
}
```

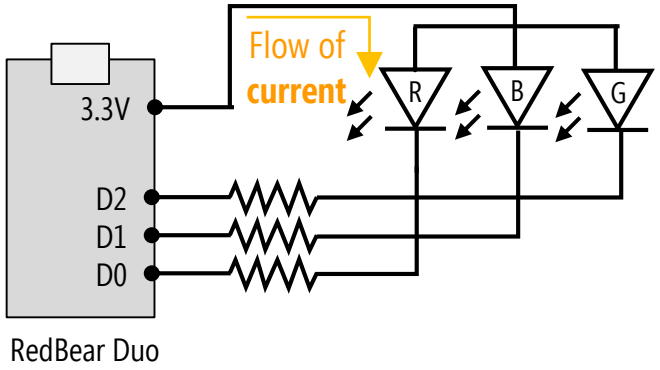
https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoRGB



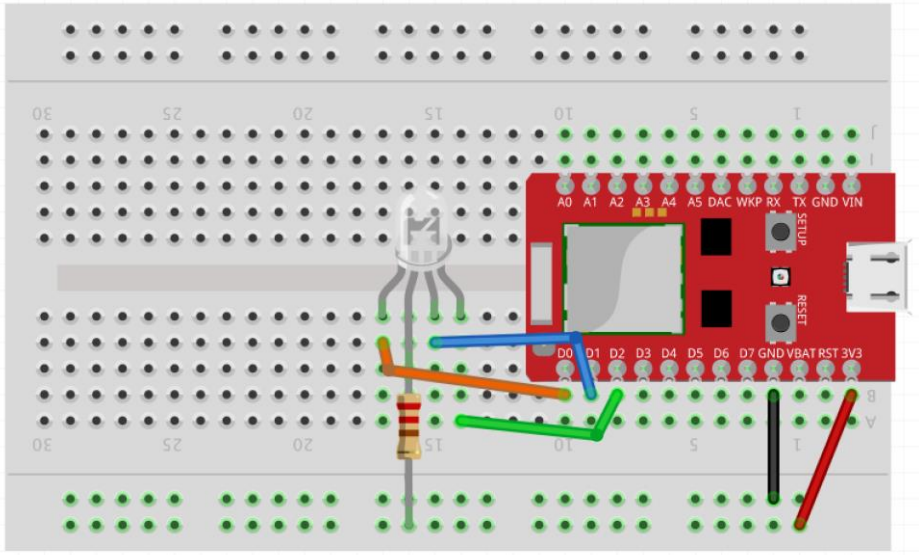
USING AN RGB LED: THE CIRCUIT (AN ALTERNATIVE)

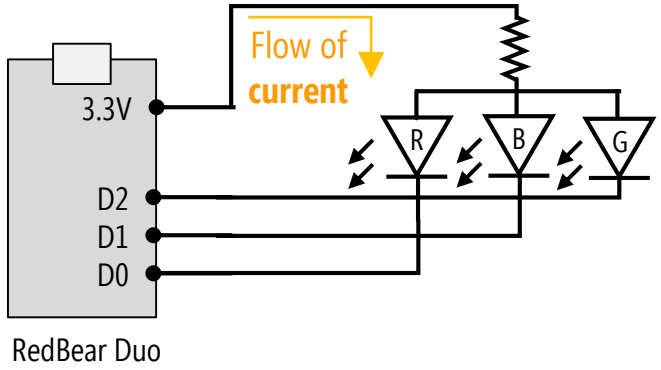
RGB Circuit 1: Set R, B, G values via D0, D1, & D2





RGB Circuit 2: Set R, B, G values via D0, D1, & D2

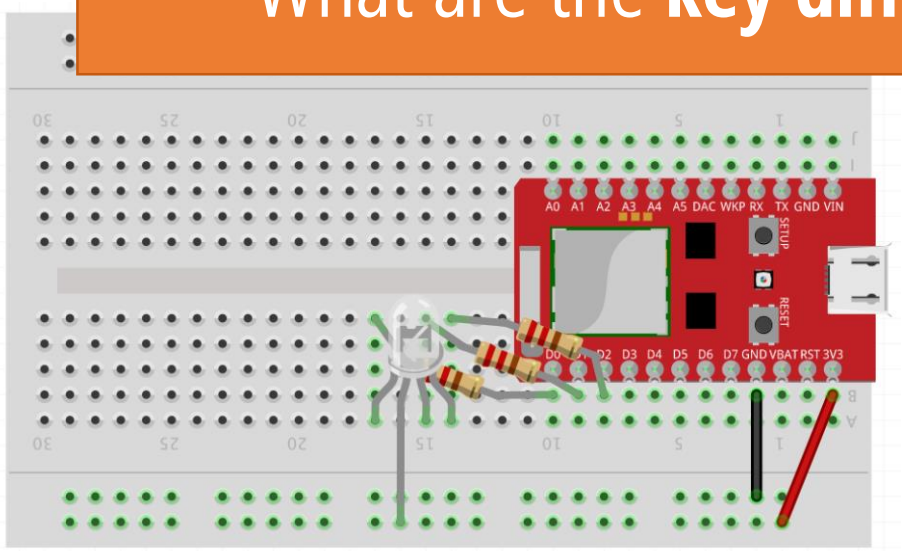


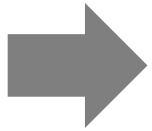


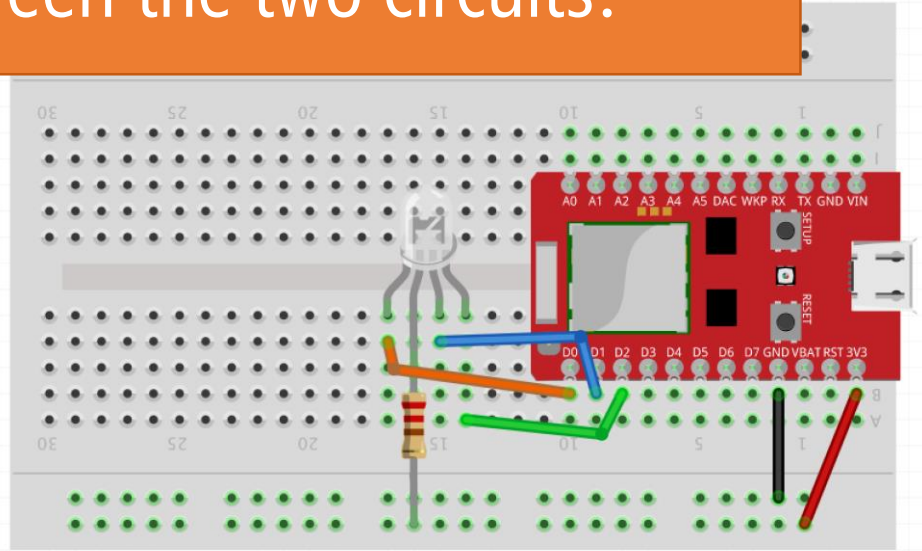
USING AN RGB LED: THE CIRCUIT (AN ALTERNATIVE)

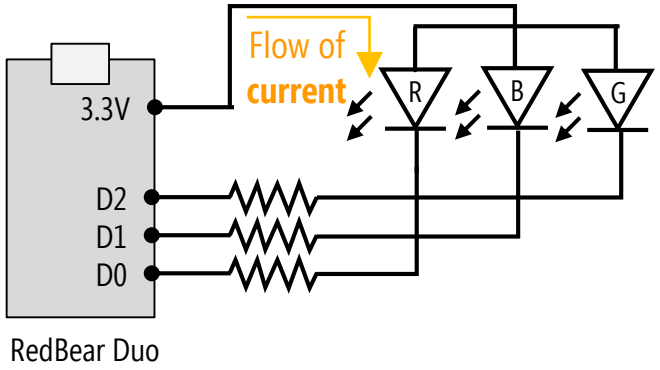
RGB … D1, & D2

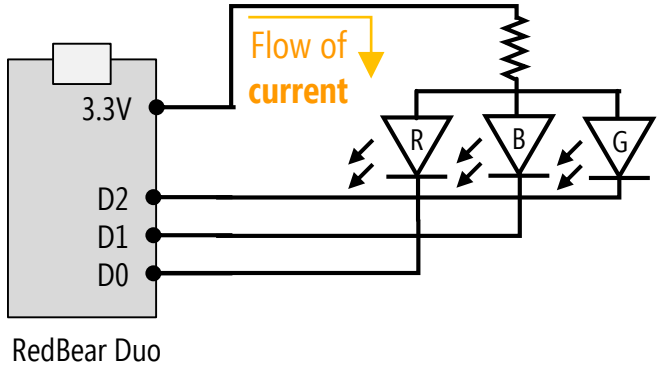
What are the **key differences** between the two circuits?











Diffused RGB (tri-colo LED - Common Anod

PRODUCT ID: 159

$2.00

46 IN STOCK

1 ADD TO CART

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

ADD TO WISHLIST

DESCRIPTION

Diffused 5mm tri-color LED with separate red, green and blue LED chips inside! Nice indicator, and fun to color-swirl. 60 degree viewing angle. We like diffused RGB LEDs because they color mix inside instead of appearing as 3 distinct LEDs.

These are Common-Anode type which means you connect one pin to 5V or so and then tie the other three legs to ground through a resistor. We carry and use CA more than CC because multi-LED driver chips (such as the TLC5940/TLC5941) are often designed exclusively for CA and can't be used with Common-Cathode.

- 5mm diameter
- Red: 630 nm wavelength, Green: 525 nm, Blue: 430 nm
- Red: 2.1-2.5V Forward Voltage, at 20mA current, Green: 3.8-4.5V, Blue: 3.8-4.5V
- Red: 500 mcd typical brightness, Green: 600 mcd, Blue: 300 mcd
- Datasheet

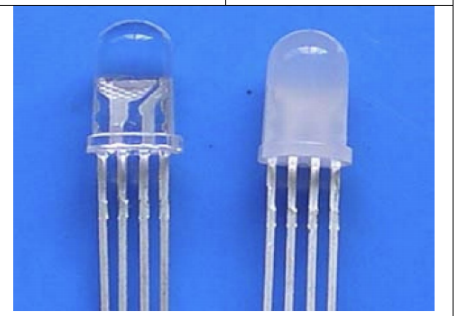
If you need some help using LEDs, please read our "Introduction to using LEDs" tutorial for any electronics project.

TECHNICAL DETAILS

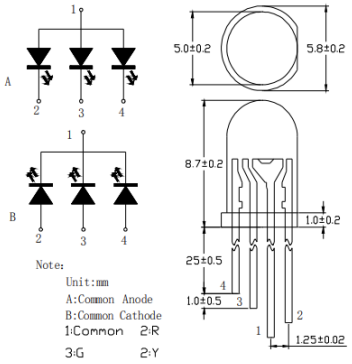
FEDY® 飞帝·中国	Shenzhen Fedy Technology Co., LTD	FD-5WSRGB-A

Features:

* 5mm Lamp LED
* Emitting Color Red Green Blue
* Low Power Consumption
* Ideal For Backlight Toy Light-Gift Indicator



Circuit and Package Dimension



Absolute Maximum Rating

Item	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current*	I_{FP}	100	mA
Reverse Voltage	V_R	5	V
Reverse Current	I_R	10	uA
Power Dissipation	P_D	150	mW
Electrostatic discharge	E_{SD}	800	V
Operation Temperature	T_{OPR}	-25～+80	℃
Storage Temperature	T_{STG}	-5～+45	℃
Lead Soldering Temperature	T_{SOL}	Max.260℃ for 5sec Max(3mm from the base of the epoxy bulb)	

Typical Optical/Electrical Characteristics

Product Type	Lens Color	Emitting Color	Forward Voltage(v)		Prcp Wavelength (nm)		Luminous Intensity(mcd)		50% Power Angle(deg)
			Type	Max.	Type.	Max.	Min.	Type	Type
5TSRGB-A/C	Water Clear	R	1.80	2.40	630	640	3000	4000	18
		G	3.20	3.60	515	525	4000	6000	
		B	3.20	3.60	460	470	900	1000	
5WSRGB-A/C	Diffused	R	1.80	2.40	630	640	2000	3000	45
		G	3.20	3.60	515	525	3000	5000	
		B	3.20	3.60	460	470	700	900	

Absolute Maximum Rating

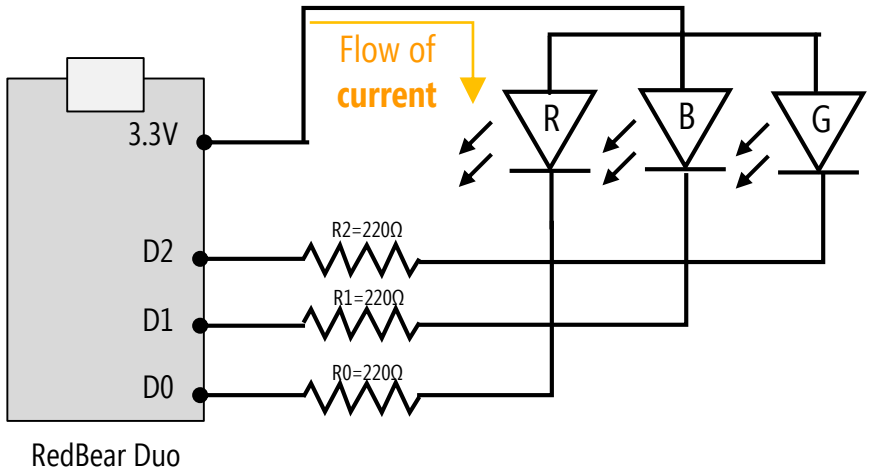
Item	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current*	I_{FP}	100	mA
Reverse Voltage	V_R	5	V
Reverse Current	I_R	10	uA

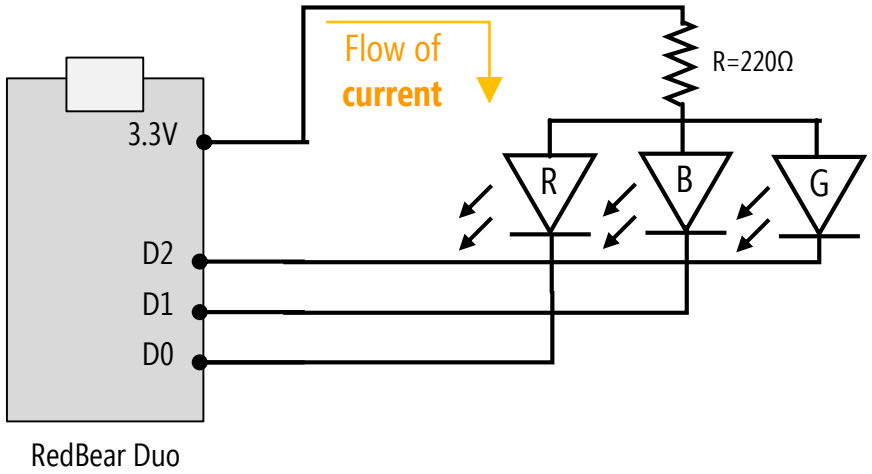
Typical Optical/Electrical Characteristics

Product Type	Lens Color	Emitting Color	Forward Voltage(v)		Prcp Wavelength (nm)	
			Type	Max.	Type.	Max.
5TSRGB-A/C	Water Clear	R	1.80	2.40	630	640
		G	3.20	3.60	515	525
		B	3.20	3.60	460	470
5WSRGB-A/C	Diffused	R	1.80	2.40	630	640
		G	3.20	3.60	515	525
		B	3.20	3.60	460	470

KEY DIFFERENCES BETWEEN RGB CIRCUIT DESIGNS

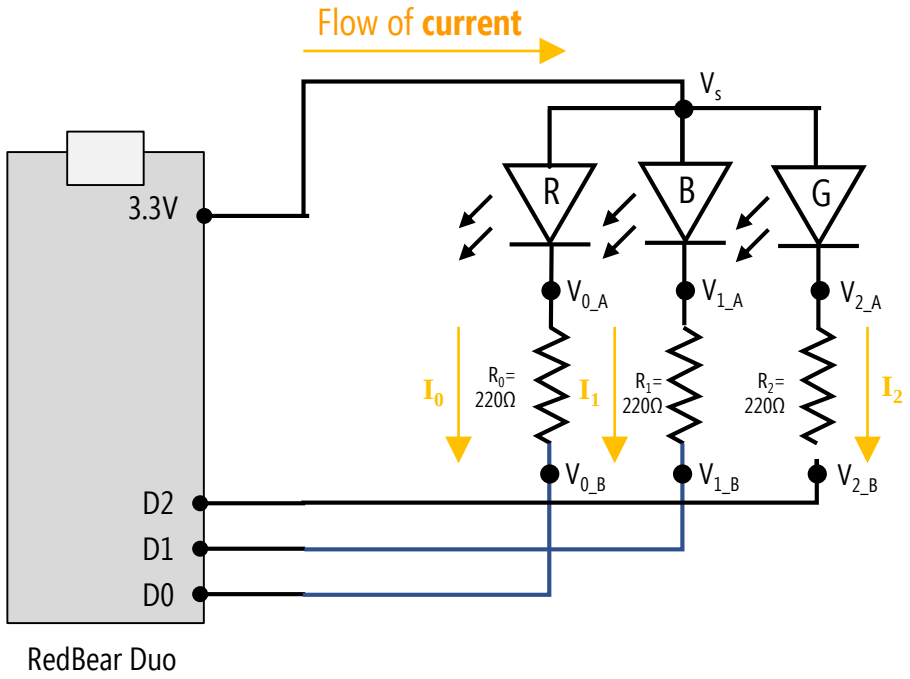
Assume D2, D1, and D0 are all analogWrite(0) so all are 0V (and, consequently, the light is white)

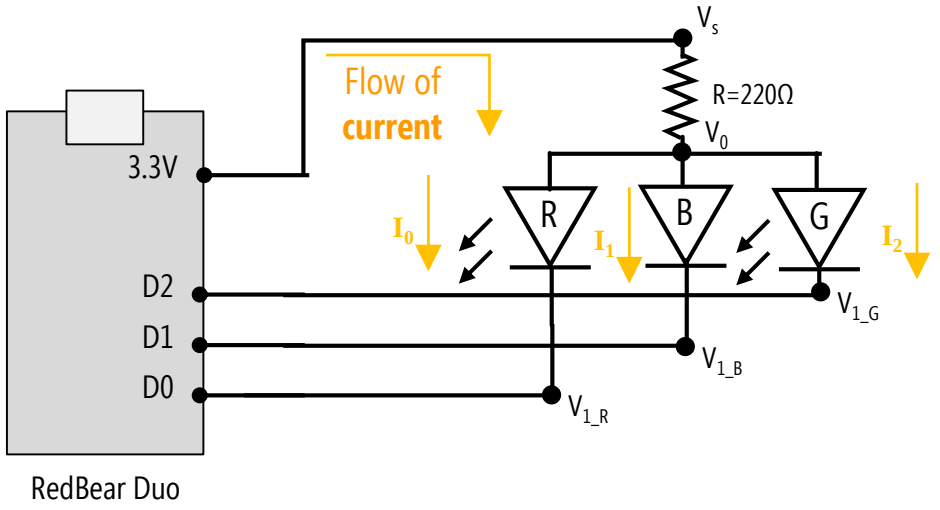




KEY DIFFERENCES BETWEEN RGB CIRCUIT DESIGNS

Assume D2, D1, and D0 are all analogWrite(0) so all are 0V (and, consequently, the light is white)





+ This circuit provides **more control** over the brightness of each LED

+ This circuit allows you to **account for the different voltage drops** of each LED (*e.g.,* red requires a smaller forward voltage than green)

- This circuit **requires more parts** (but is still, in general, a very simple circuit)

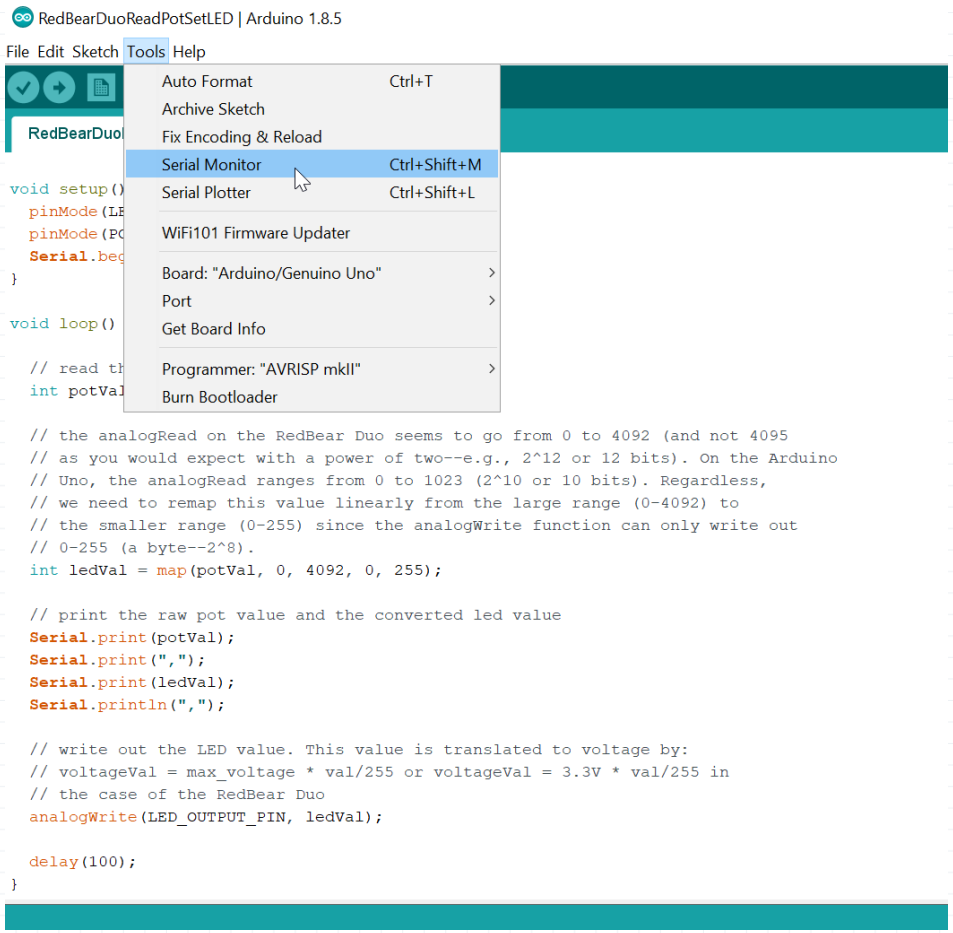
+ This circuit is **simpler** and uses **fewer parts**

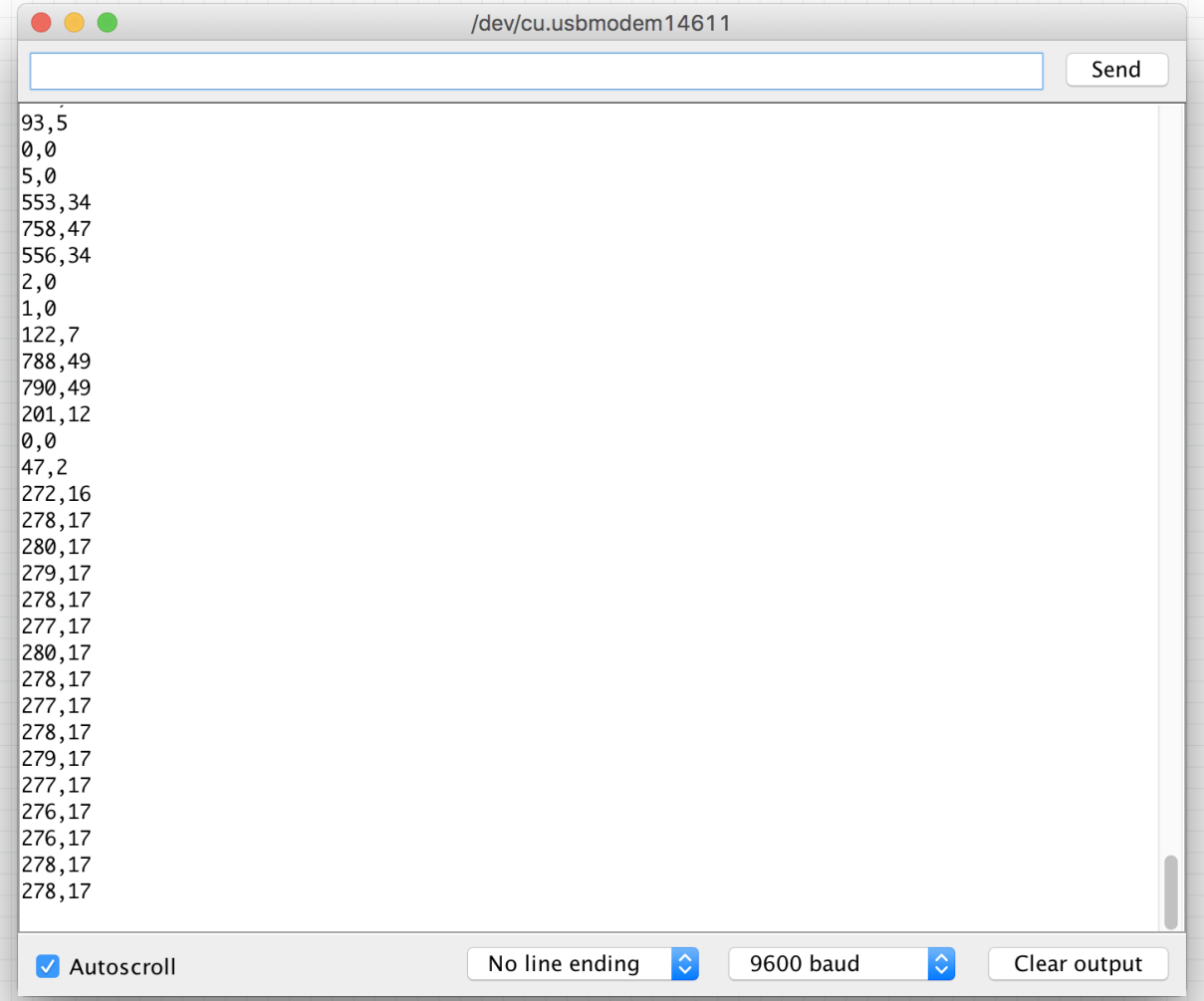
- This circuit does **not allow you to individually control** the amount of current (*i.e.,* brightness) of each LED

So, now we've changed the color of the RGB LED. **How would you dynamically change the brightness using Arduino?**
(This is part of your A3 assignment, so I'll leave this up to you).

SERIAL MONITOR

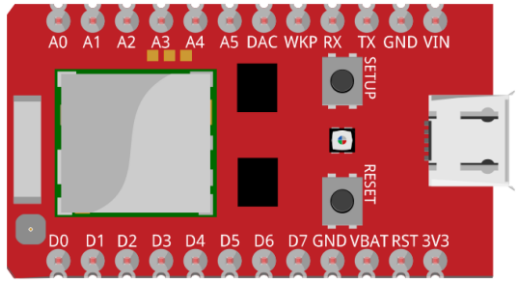
SERIAL MONITOR IS USEFUL FOR DEBUGGING



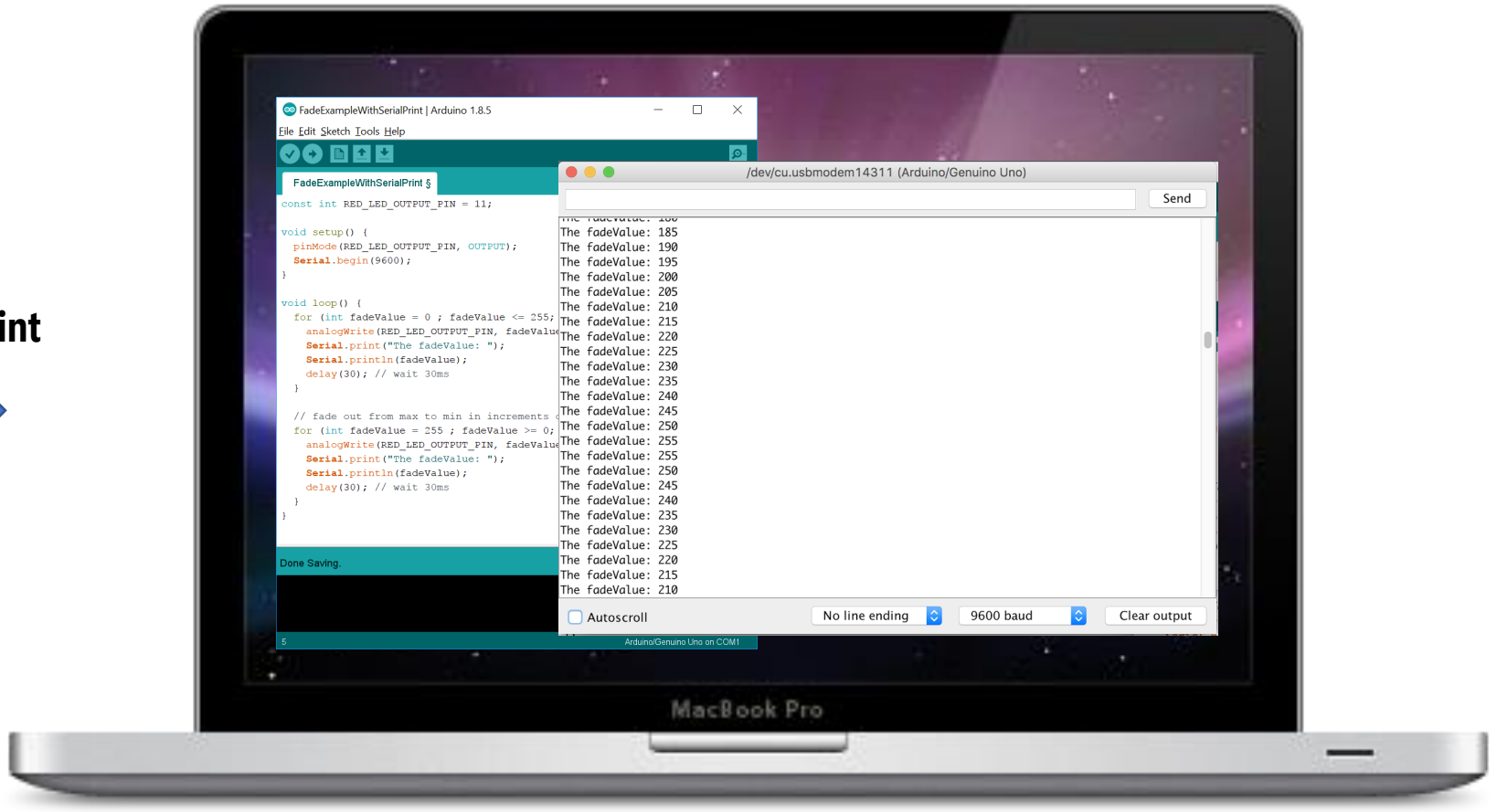


SERIAL MONITOR

CONCEPTUALLY UNDERSTANDING SERIAL.PRINT



Serial.print



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

SERIAL MONITOR

PRINTING TO SERIAL MONITOR: PRINT()

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

SERIAL MONITOR
TRY IT YOURSELF!

FadeExampleWithSerialPrint | Arduino 1.8.5

File Edit Sketch Tools Help

FadeExampleWithSerialPrint §

```
const int RED_LED_OUTPUT_PIN = 11;

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

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

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

Done Saving.

5 Arduino/Genuino Uno on COM1

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

Send

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

Autoscroll No line ending 9600 baud Clear output

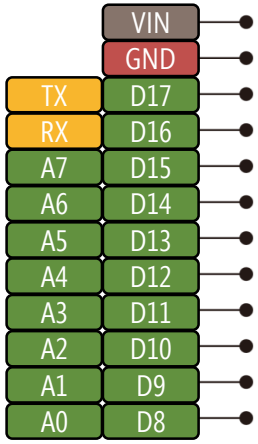
OUTPUT

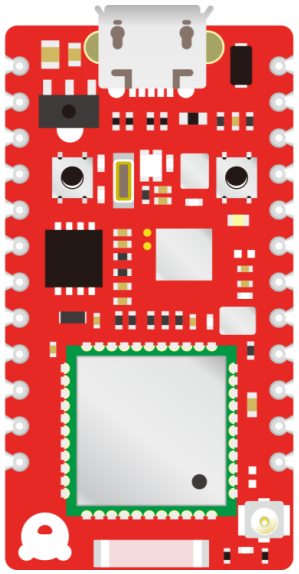
DIGITAL AND ANALOG OUTPUT

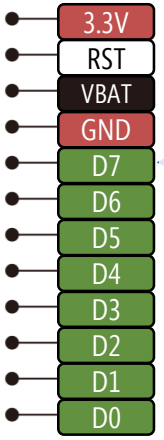
This completes our quick tour of digital and analog output on the Arduino

Analog Input on A7

Reads in any any value between 0V or 3.3V using analogRead function. In this case, the value of a photocell

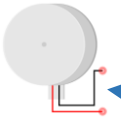






Digital Input on D7

Reads in a digital input signal (anything below 1.65V converted to LOW, anything above 1.65V converted to HIGH). In this case, the value of switch.



Analog Output on D9

Writes out any value between 0V or 3.3V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage).

Digital Output on D0

Writes out 0V or 3.3V using digitalWrite function. In this case, turning on and off an LED.

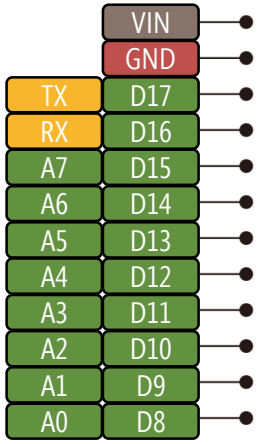
INPUT

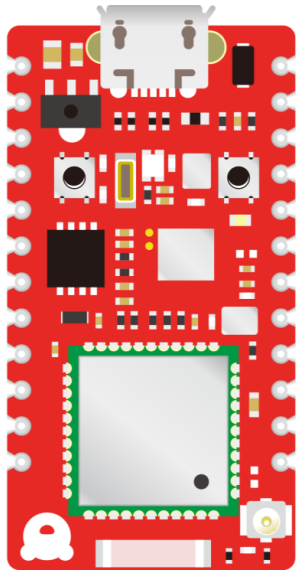
DIGITAL AND ANALOG INPUT

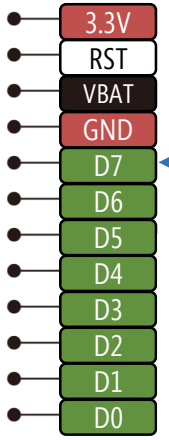
Now we are going to switch from talking about output to talking about input

Analog Input on A7

Reads in any any value between 0V or 3.3V using analogRead function. In this case, the value of a photocell







Digital Input on D7

Reads in a digital input signal (anything below 1.65V converted to LOW, anything above 1.65V converted to HIGH). In this case, the value of switch.

Digital Output on D0

Writes out 0V or 3.3V using digitalWrite function. In this case, turning on and off an LED.

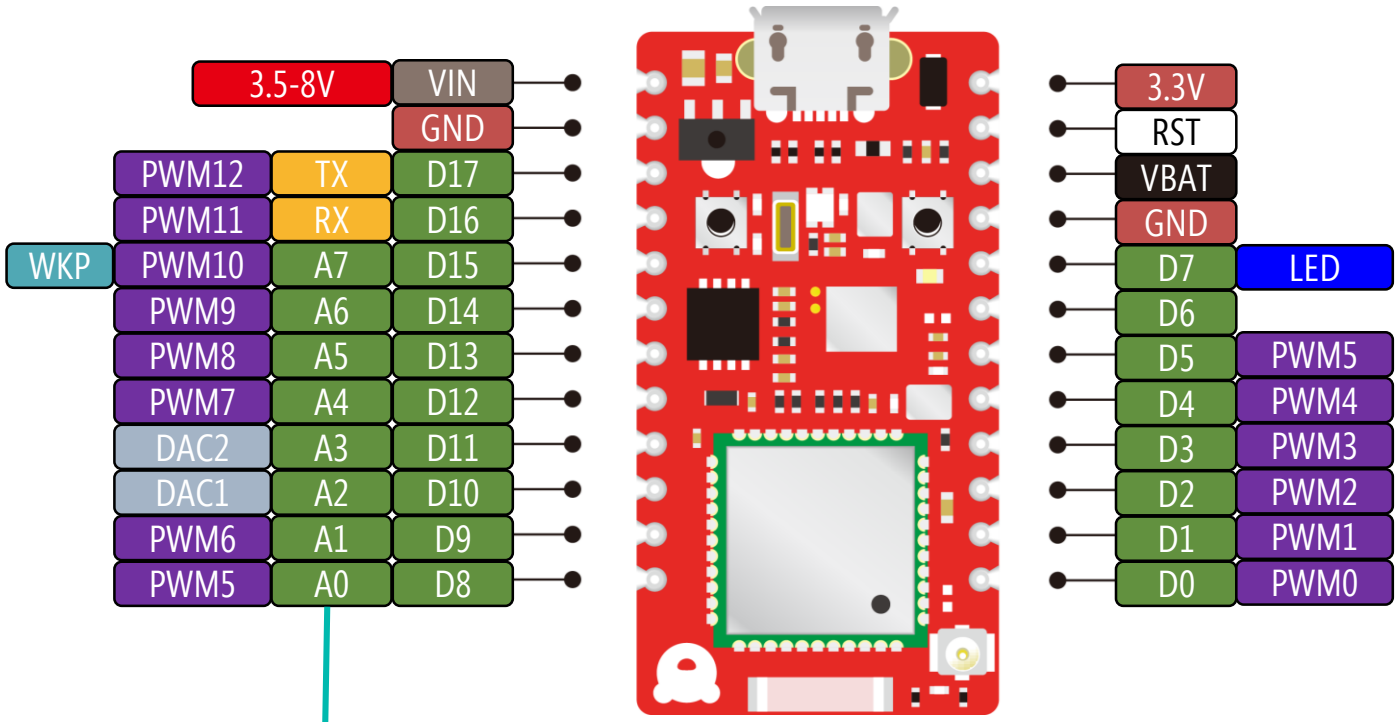
Analog Output on D9

Writes out any value between 0V or 3.3V using analogWrite function. In this case, vibrating a motor (where strength of vibration proportional to voltage).

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

ANALOG INPUT

RECALL THAT WE ONLY HAVE EIGHT ANALOG INPUTS



We can only read in analog values on **A0-A7**

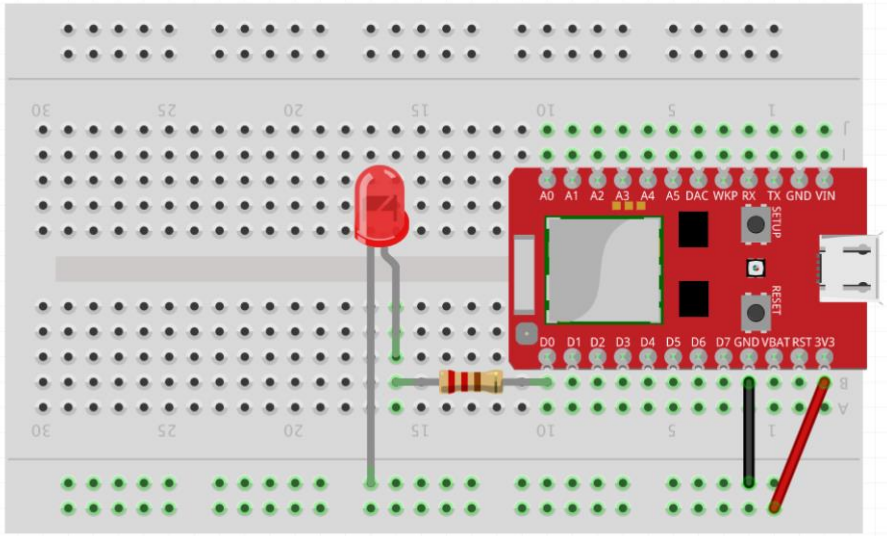
Legend

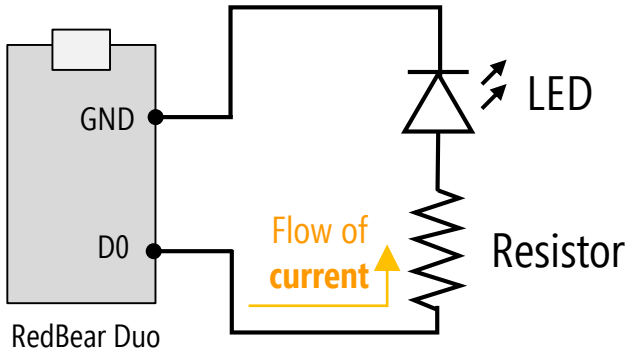
AX	Analog Input Pin
DX	Digital Input/output Pin
PWM	Supports Analog Out via Pulse Width Modulation

ANALOG INPUT

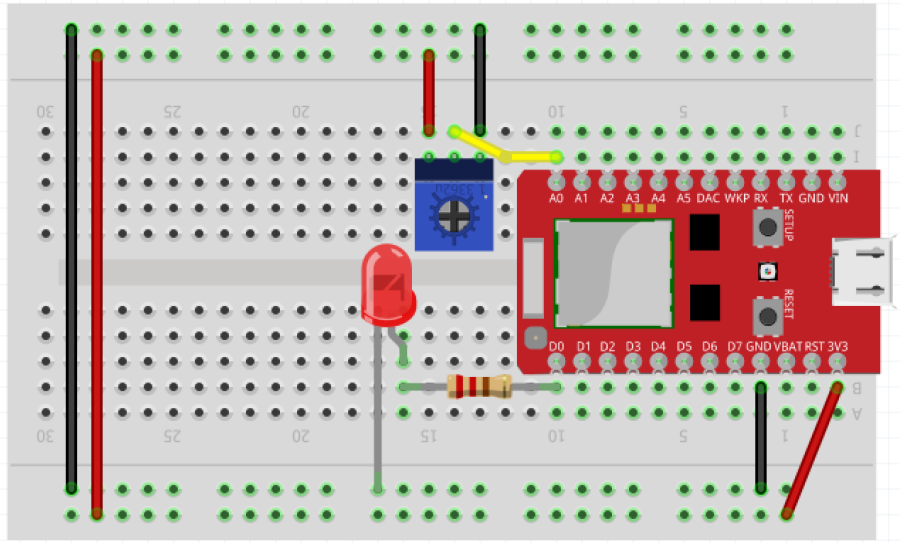
CONTROL LED BRIGHTNESS WITH POTENTIOMETER

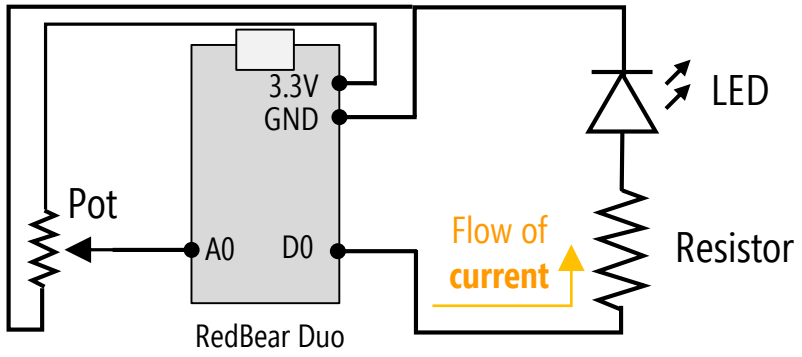
Old Circuit: Fade LED on/off via the D0 pin





New Circuit: Fade LED on/off via pot value

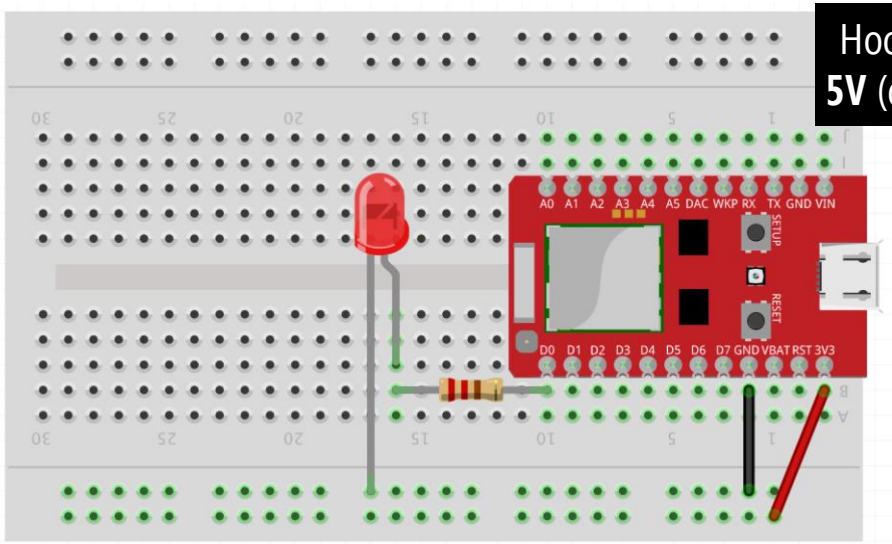


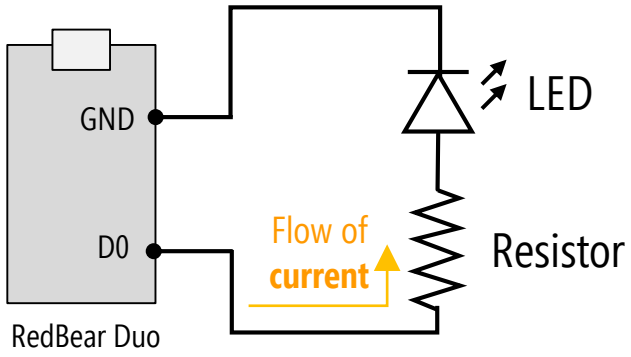


ANALOG INPUT

CONTROL LED BRIGHTNESS WITH POTENTIOMETER

Old Circuit: Fade LED on/off via the D0 pin

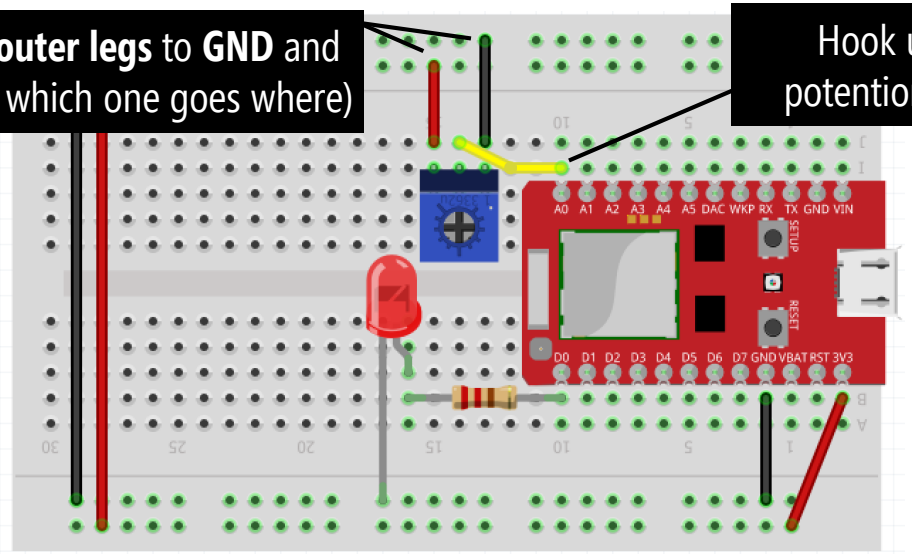


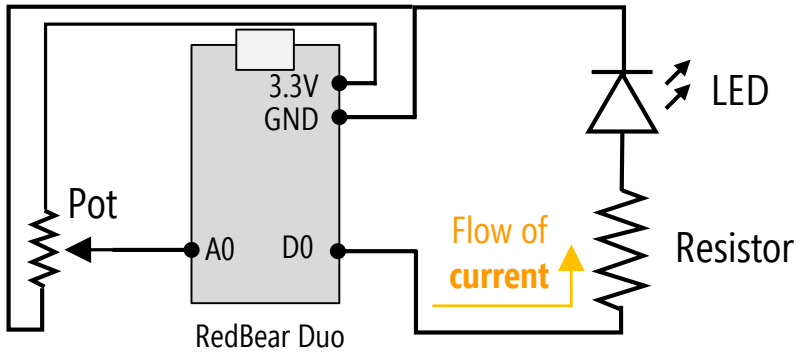


New Circuit: Fade LED on/off via pot value

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

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





ACTIVITY: LEARN ANALOG INPUT

ANALOG INPUT: ANALOGREAD()

Reads the value from the specified analog pin. The input voltage (0-5V) is mapped to 0-1023.

Syntax

```
analogRead(pin)
```

Parameters

`pin`: the analog pin to read from

Returns

An integer between 0 and 1023 (inclusive). Note: on the RedBear Duo, it appears that this is actually more like 0 to 4096 (which is 2^{12} rather than 2^{10} on the Arduino Uno—so, 12 bits reserved for ADC on the Duo). Emprically, I've found that the range is more like 0 to 4092.

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

ACTIVITY: LEARN ANALOG INPUT

HELPER FUNCTION: MAP()

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

Syntax

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

Parameters

`value`: the number to map

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

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

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

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

Returns

The mapped value

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

ACTIVITY: LEARN ANALOG INPUT

HELPER FUNCTION: MAP()

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

Syntax

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

Parameters

`value`: the number to map

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

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

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

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

Returns

The mapped value

There is no magic here. The full utility function is simply:

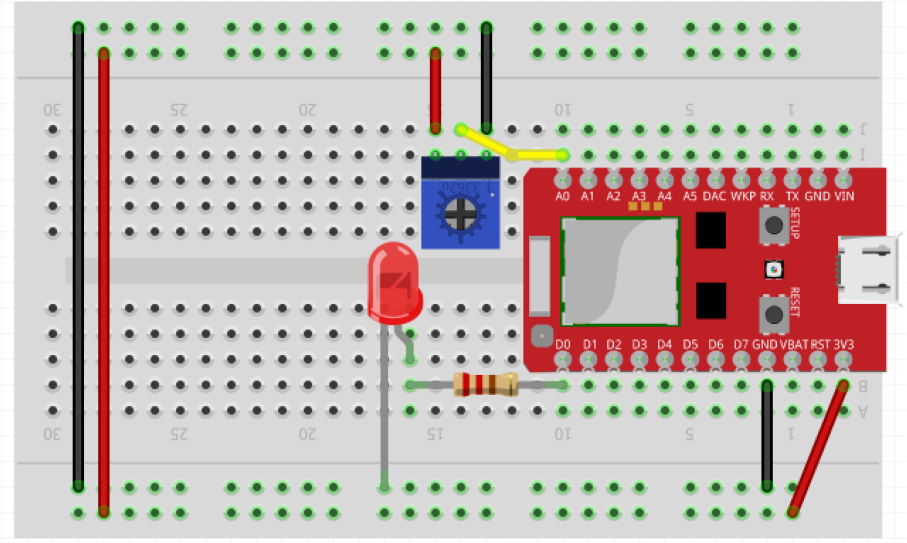
```
long map(long val, long fromLow, long fromHigh, long toLow, long toHigh)
{
  float rangeFraction = (toHigh - toLow) / (fromHigh - fromLow);
  return toLow + (x - fromLow) * rangeFraction;
}
```

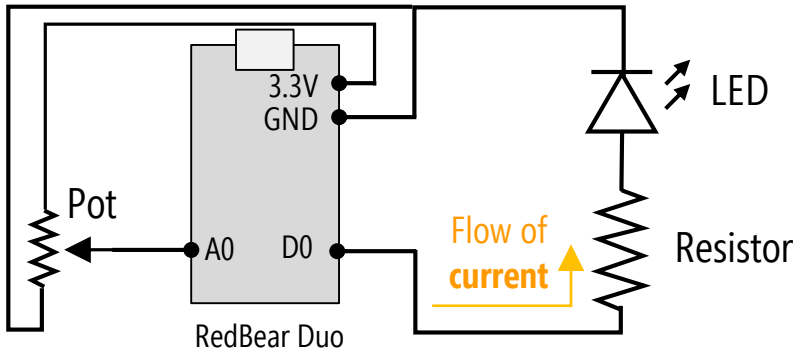
https://www.arduino.cc/reference/en/language/functions/math/map/

ANALOG INPUT

CONTROL LED BRIGHTNESS WITH POTENTIOMETER

Circuit: Fade LED on/off via pot value





Write Code: Fade LED on/off via pot value

RedBearDuoReadPotSetLED | Arduino 1.8.5

File Edit Sketch Tools Help

RedBearDuoReadPotSetLED

```
SYSTEM_MODE(MANUAL);

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

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

void loop() {

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

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

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

  delay(100);
}
```

https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoReadPotSetLED

RESET
SETUP

When I taught Arduino at UMD, one of my first assignments was to make a visualizer for Arduino signals (see ArduinoGraphTutorials), now the Arduino IDE ships with its own signal visualizer called **Serial Plotter**—a great way to visualize I/O signals. Let's modify our potentiometer code to take advantage of it.

SERIAL PLOTTER

USING THE SERIAL PLOTTER

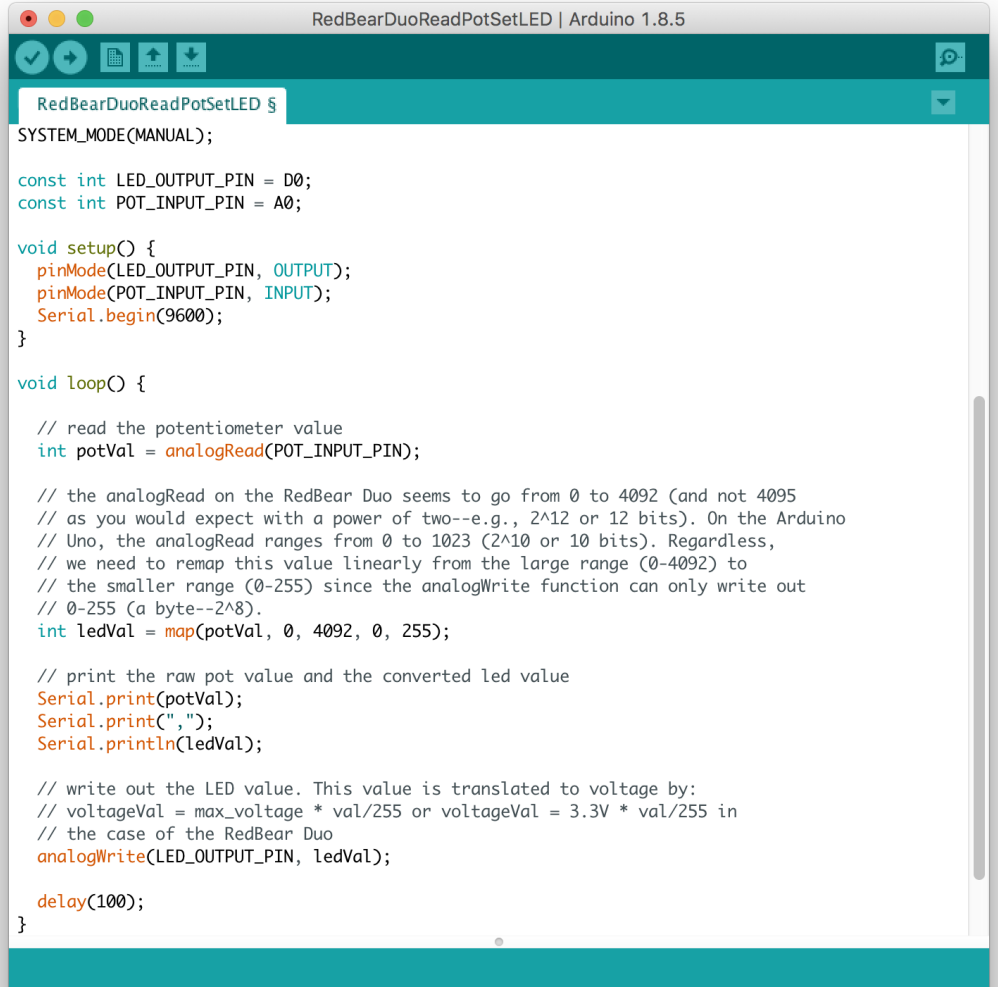
Serial Plotter automatically visualizes comma separated numerical values coming in off Serial.

Old Code: Fade LED on/off via pot value


```
SYSTEM_MODE(MANUAL);

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

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

void loop() {

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

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

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

  delay(100);
}
```

https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoReadPotSetLED

Updated Code: Adds in Serial.println of pot value

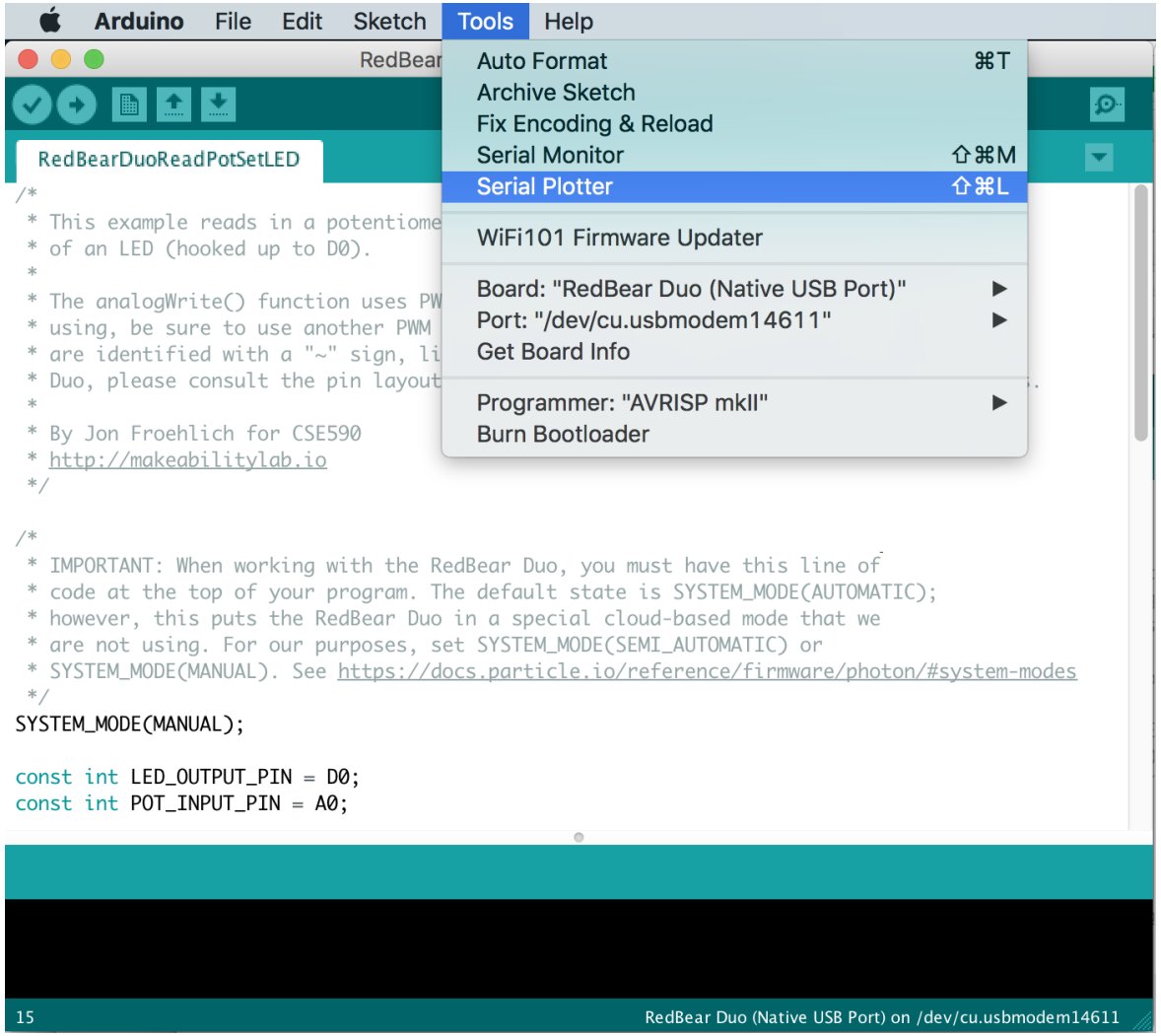


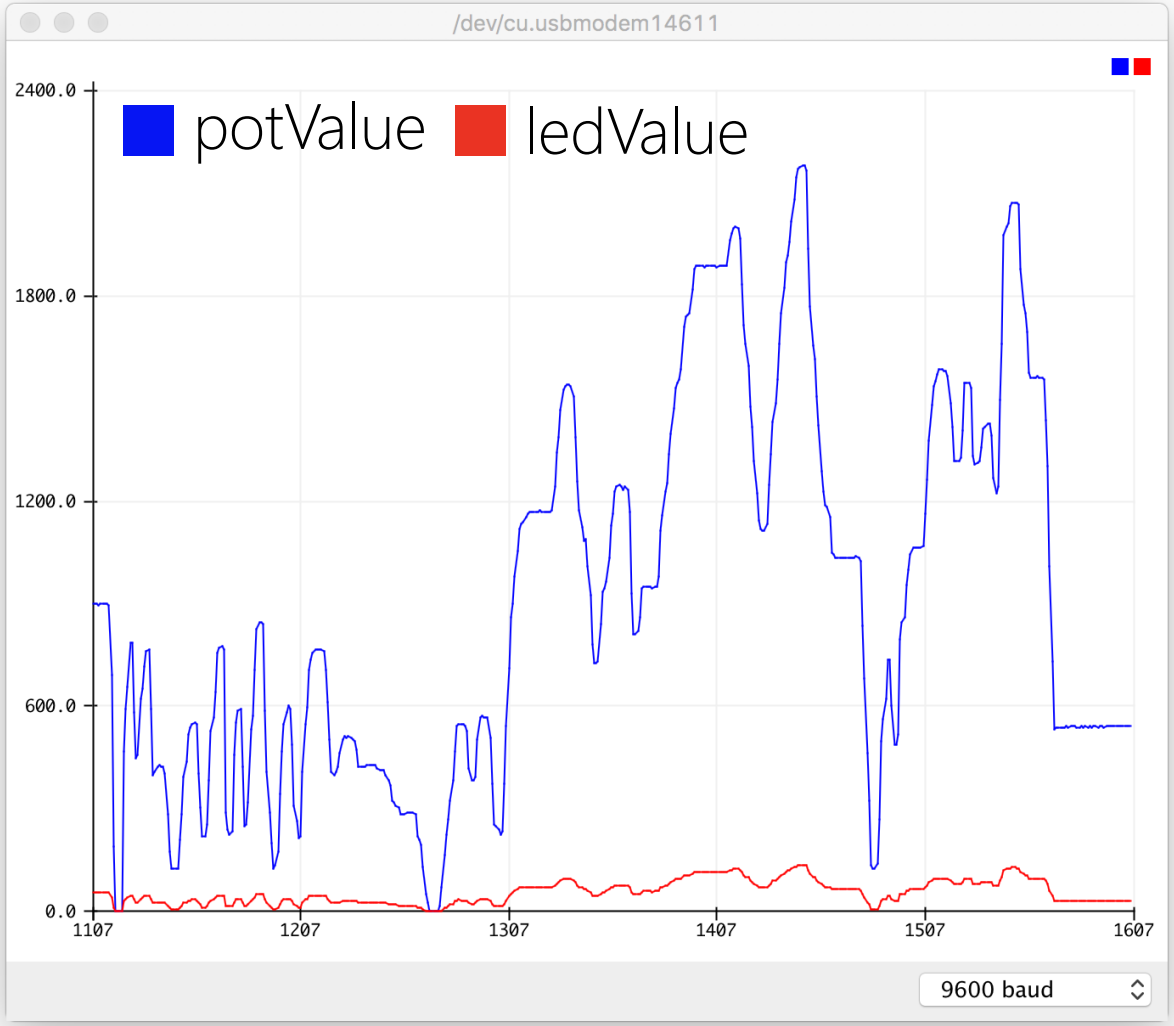

```
SYSTEM_MODE(MANUAL);

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

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

void loop() {

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

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

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

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

  delay(100);
}
```

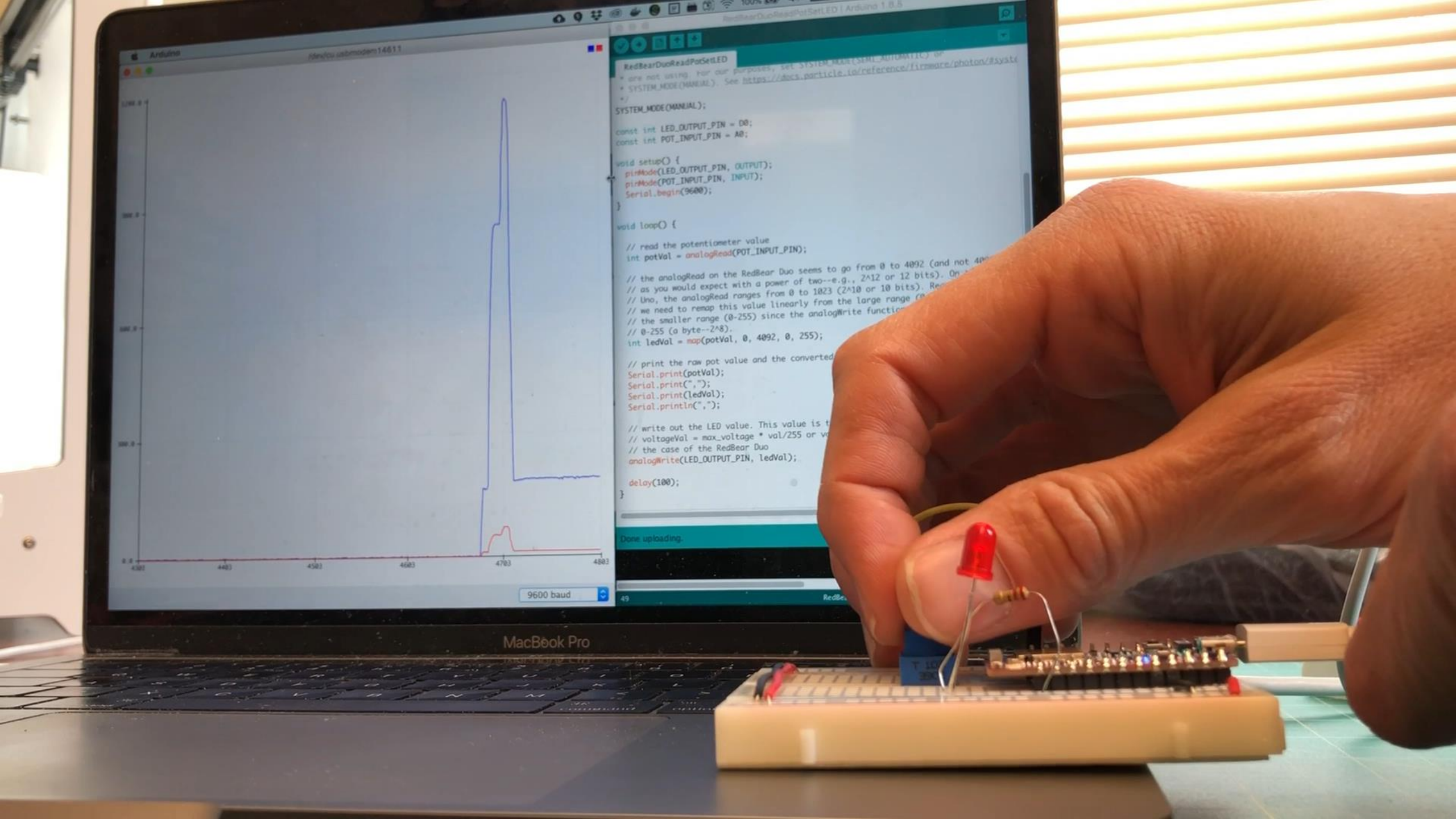
https://github.com/jonfroehlich/CSE590Sp2018/tree/master/L05-Arduino/RedBearDuoReadPotSetLED

SERIAL PLOTTER

USING THE SERIAL PLOTTER



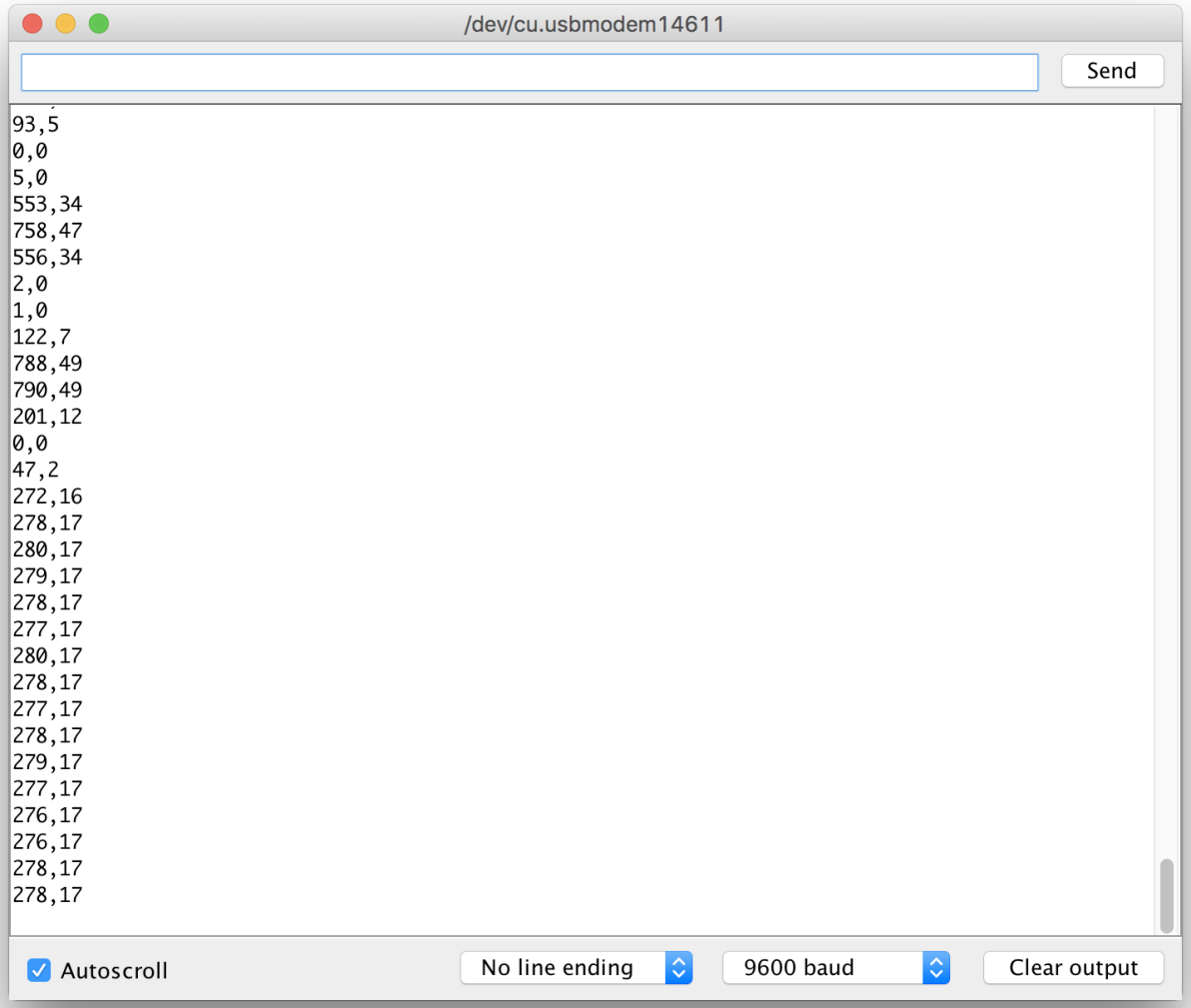


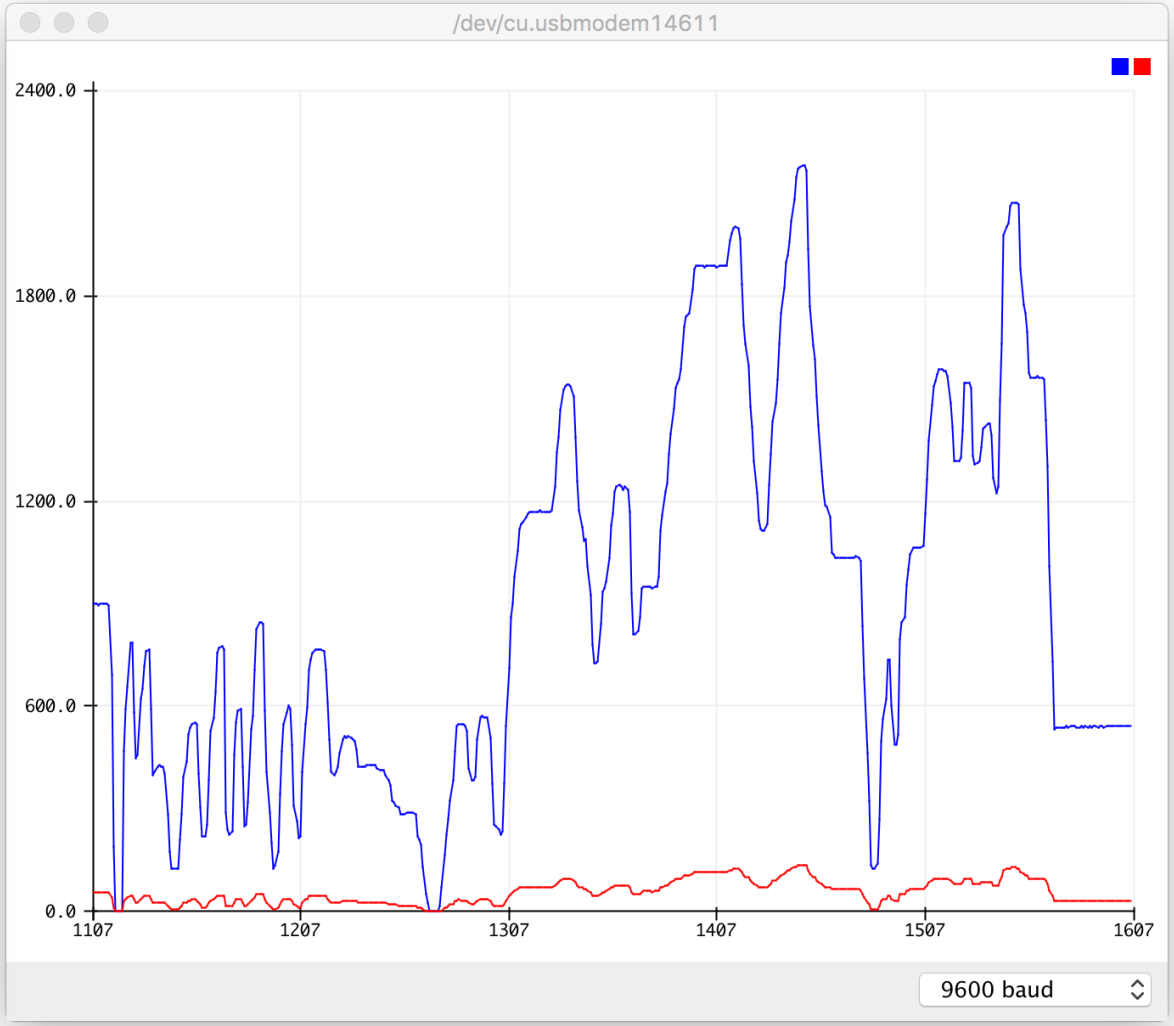

Arduino
/dev/cu.usbmodem14611
9600 baud
RedBearDuoReadPotSetLED
SYSTEM_MODE(MANUAL);
const int LED_OUTPUT_PIN = D0;
const int POT_INPUT_PIN = A0;
void setup() {
pinMode(LED_OUTPUT_PIN, OUTPUT);
pinMode(POT_INPUT_PIN, INPUT);
Serial.begin(9600);
}
void loop() {
// read the potentiometer value
int potVal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

SERIAL PLOTTER

SERIAL MONITOR VS. PLOTTER

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

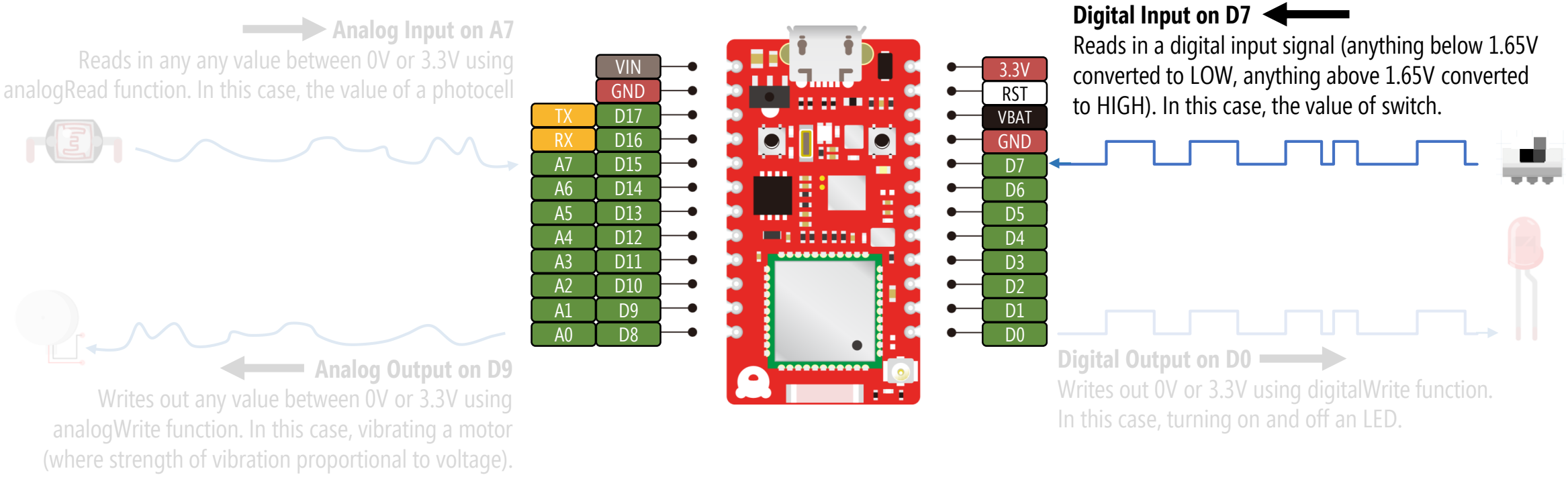




INPUT

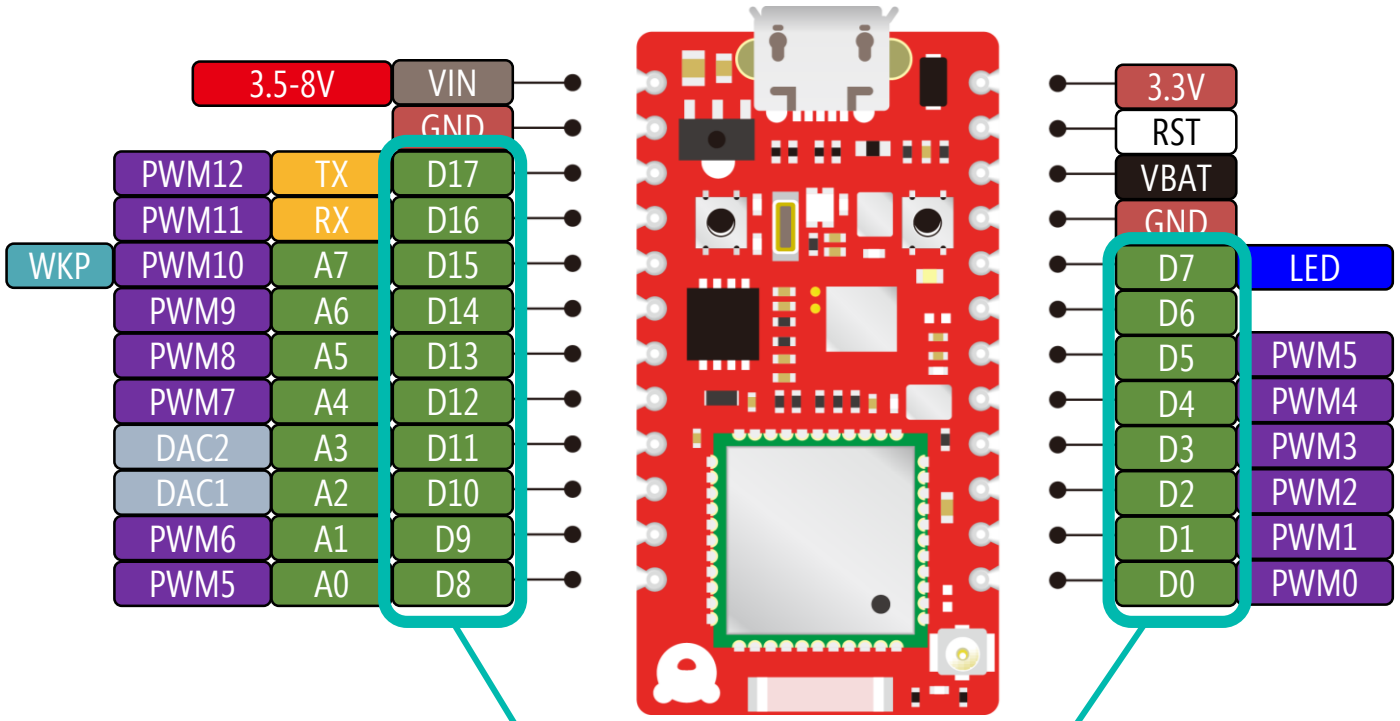
DIGITAL INPUT

Finally, the last thing we are going to cover is digital input—which is not as straightforward as you might initially suspect.



DIGITAL INPUT

ANY OF THE D PINS CAN BE SET FOR DIGITAL INPUT OR OUTPUT



Legend

AX	Analog Input Pin
DX	Digital Input/output Pin
PWM	Supports Analog Out via Pulse Width Modulation

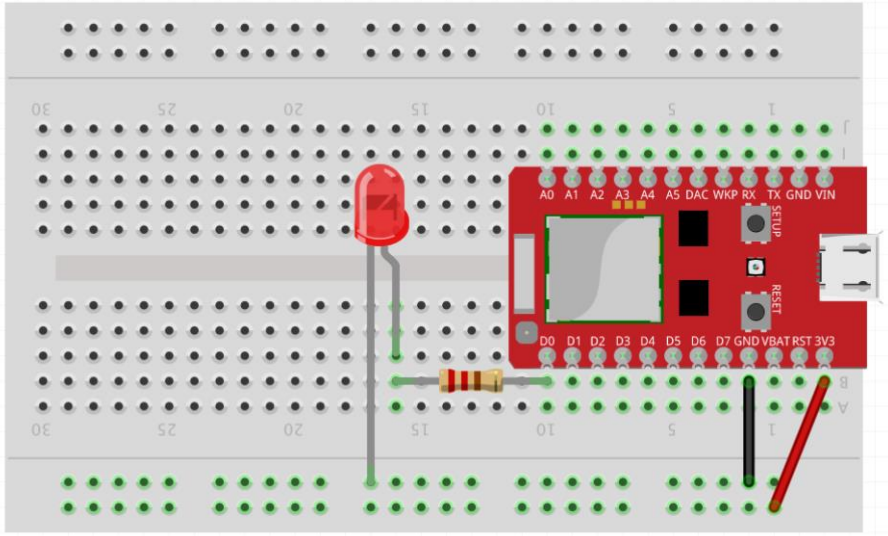
Any of the D pins can be set for input or output in setup() using pinMode()

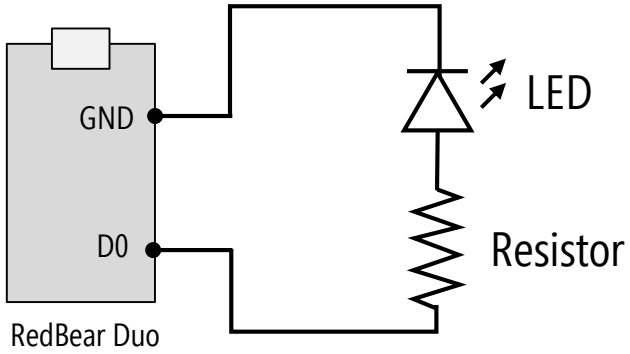
Let’s design a circuit& write code to
turn **on** and **off** an **LED** with a **button**.

DIGITAL INPUT

TURN ON/OFF LED WITH A BUTTON

Basic Circuit: Control LED via the D0 pin

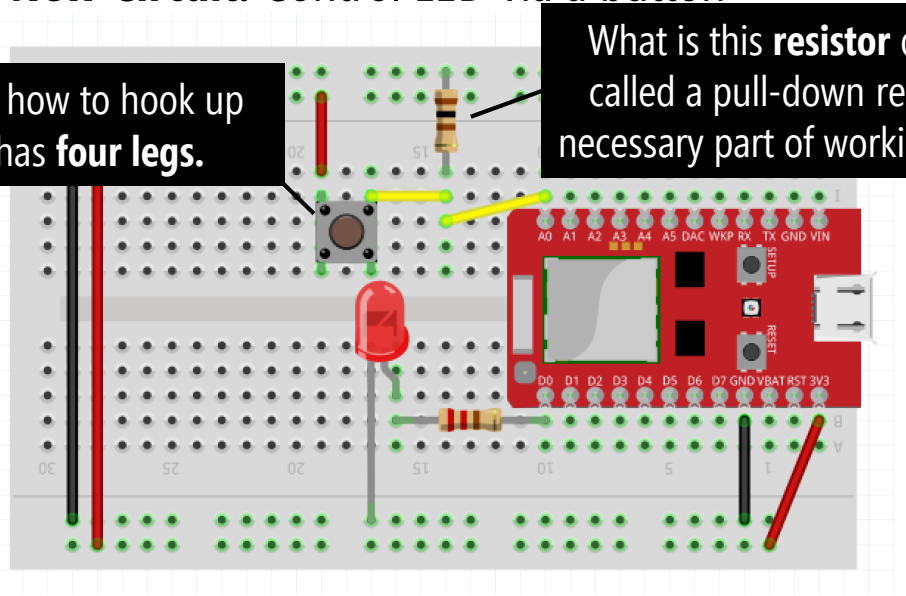


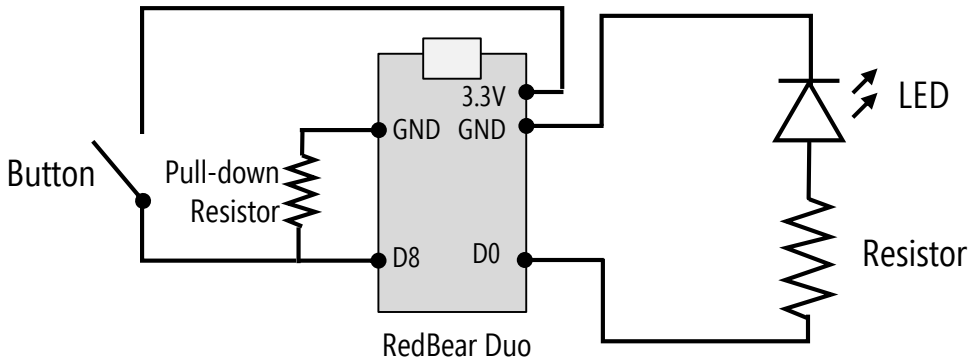


New Circuit: Control LED via a button

How do we know how to hook up this button? It has **four legs.**

What is this **resistor** doing here? It's called a pull-down resistor and it's a necessary part of working with switches.





HOOKING UP BUTTON: CONSULT THE DATASHEET

OMRON

Tactile Switch
B3F

Miniature, Space-Saving Tactile Switch Provides Long Service Life and Easy Mounting

- Extended mechanical/electrical service life: 10 x 10^6 operations for 12 x 12 mm type and 1 x 10^6 operations for the 6 x 6 mm type
- Ideal for applications such as audio, office and communications equipment, measuring instruments, TVs, VCRs, etc.
- Taped radial type, vertical type and high force types are available as series versions
- Flux-tight base structure allows automatic soldering of the tactile switches onto a PC board
- Gold plated models available for increased contact reliability, resistance to corrosive gas and insulation fault prevention for ion migration in harsh environments
- RoHS Compliant

Ordering Information

■ **B3F-1□□□, B3F-3□□□**

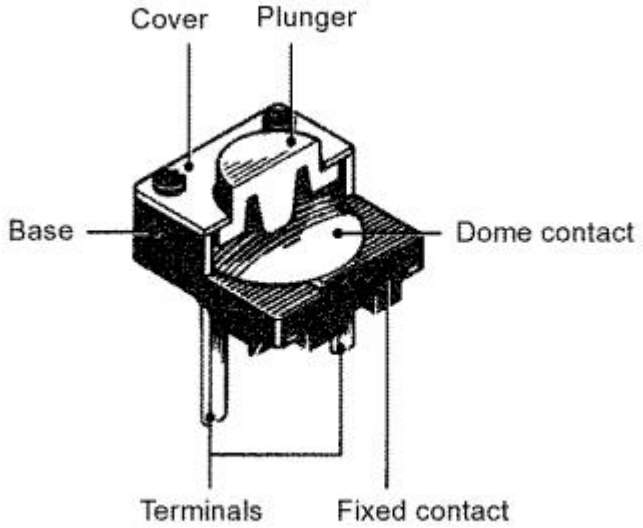
6 x 6 mm type

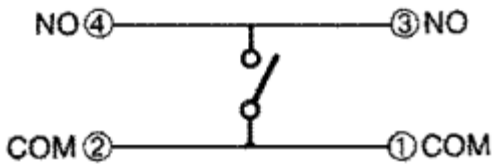
Type	Plunger	Switch height x pitch	Operating force	Model			
				Without ground terminal		With ground terminal	
				Bags	Sticks*	Bags	Sticks*
Standard	Flat	4.3 x 6.5 mm	General-purpose: 100 g	B3F-1000	B3F-1000S	B3F-1100	B3F-1100S
			150 g	B3F-1002	B3F-1002S	B3F-1102	B3F-1102S
			High-force: 260 g	B3F-1005	B3F-1005S	B3F-1105	B3F-1105S
		5.0 x 6.5 mm	General-purpose: 100 g	B3F-1020	B3F-1020S	B3F-1120	B3F-1120S
			150 g	B3F-1022	B3F-1022S	B3F-1122	B3F-1122S
			High-force: 260 g	B3F-1025	B3F-1025S	B3F-1125	B3F-1125S
		5.0 x 7.5 mm	General-purpose: 100 g	—	—	B3F-1110	—
	Projected	7.3 x 6.5 mm	General-purpose: 100 g	B3F-1050	B3F-1050S	B3F-1150	B3F-1150S
			150 g	B3F-1052	B3F-1052S	B3F-1152	B3F-1152S
			High-force: 260 g	B3F-1055	B3F-1055S	B3F-1155	B3F-1155S
Vertical	Flat	3.15 mm	General-purpose: 100 g	—	—	B3F-3100	—
			150 g	—	—	B3F-3102	—
			High-force: 260 g	—	—	B3F-3105	—
		3.85 mm	General-purpose: 100 g	—	—	B3F-3120	—
			150 g	—		B3F-3122	—
			High-force: 260 g	—	—	B3F-3125	—
	Projected	6.15 mm	General-purpose: 100 g	—	—	B3F-3150	—
			150 g	—	—	B3F-3152	—
			High-force: 260 g	—	—	B3F-3155	—

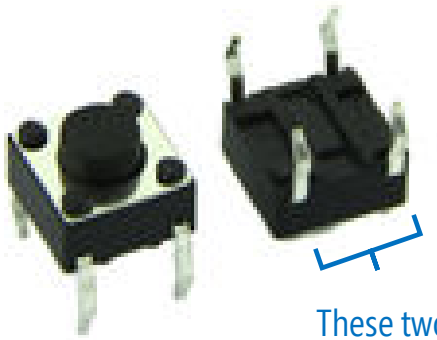
* Number of switches per stick: Without ground terminal 90/stick
With ground terminal 75/stick

Important Note: Switches cannot be water-washed.

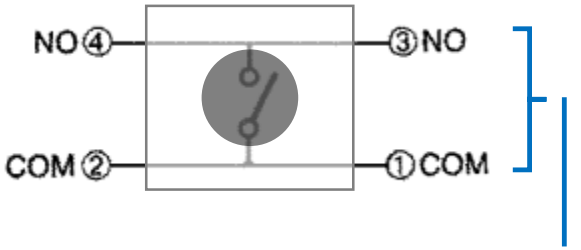
1128 Tactile Switch **B3F**







These two sides become connected when button is pressed down



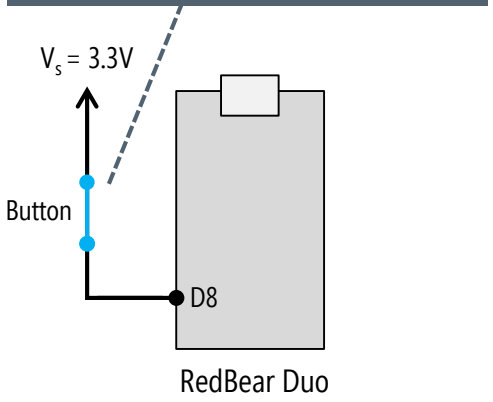
These two sides become connected when button is pressed down

https://www.adafruit.com/product/1119

DIGITAL INPUT

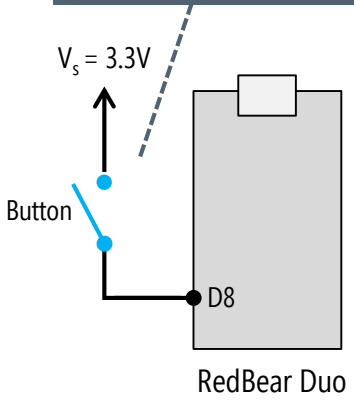
HOOKING UP A BUTTON: PULL-DOWN RESISTORS

When the this button is closed (as it is below), what would the **microcontroller** (MCU) **read** from the **input pin**?



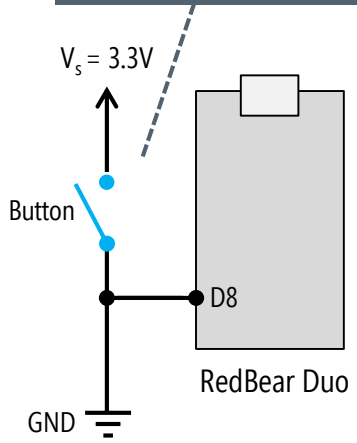
Answer: The MCU would read 3.3V (or HIGH)

Now, when the **button switches** to **open**, what would the MCU read?



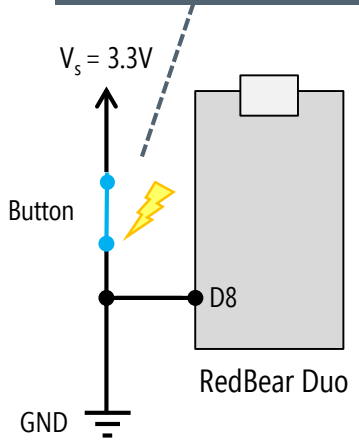
Answer: Uh, oh, the input pin is in an unknown state (commonly called "floating")—this is bad!

Well, can't we just solve that by **adding GND** here that "pulls" D8 to 0V when the switch is open?



Answer: You're on the right track but this will create a short circuit when the button is closed!

Now, when the button is closed, we have a **short circuit** (V_s is connected to GND. This could damage our MCU)!

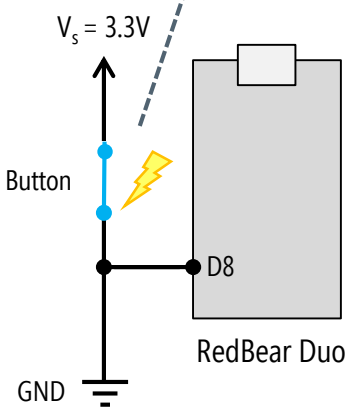


Question: So, what do we do? We add what's called a pull-down resistor.

See: https://learn.sparkfun.com/tutorials/switch-basics/all; https://learn.sparkfun.com/tutorials/pull-up-resistors/all

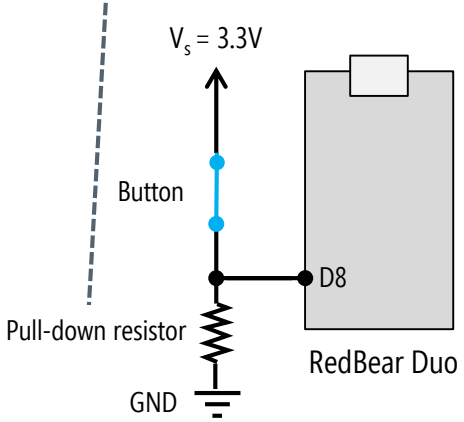
HOOKING UP A BUTTON: PULL-DOWN RESISTORS

Now, when the button is closed, we have a **short circuit** (V_s is connected to GND. This could damage our MCU)!



Question: So, what do we do? We add what's called a pull-down resistor.

This resistor is called a "**pull-down resistor**" because it biases the input low (to GND) when the switch is open.



Now, when the switch is open, the MCU is pulled to GND. When the switch is closed, the MCU is 3.3V (HIGH).

How do you choose the resistor value for the pull-down?

If you choose a low resistor value, more current will be "wasted" by flowing from V_s to GND when the button is closed. In contrast, a high resistor value (*e.g.*, 4MΩ) may not work as a pull-down (not enough current will flow).

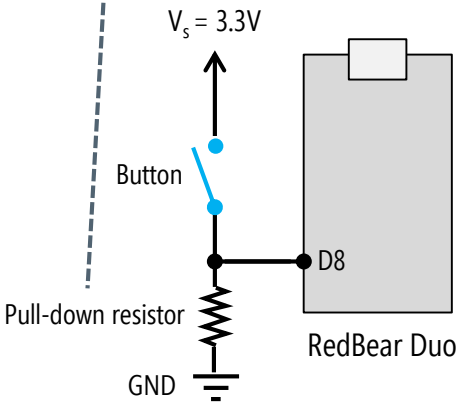
Arduino recommends using a 10K Ω resistor.

See: https://learn.sparkfun.com/tutorials/switch-basics/all; https://learn.sparkfun.com/tutorials/pull-up-resistors/all

DIGITAL INPUT

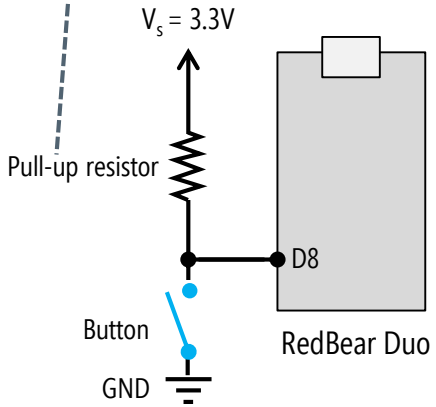
PULL-UP VS. PULL-DOWN RESISTORS

This resistor is called a "**pull-down resistor**" because it biases the input low (to GND) when the switch is open.



Pull-Down Resistor Configuration
When the switch is open, the MCU is pulled to GND. When the switch is closed, the MCU is 3.3V (HIGH) as it becomes directly connected to V_s.

This resistor is called a "**up resistor**" because it biases the input high (to V_s) when the switch is open.



Pull-Up Resistor Configuration
When the switch is open, the MCU is pulled to V_s. When the switch is closed, the MCU is 0V (LOW) as it becomes directly connected to GND.

You can wire up either pull-down resistor configurations or pull-up resistor configurations.

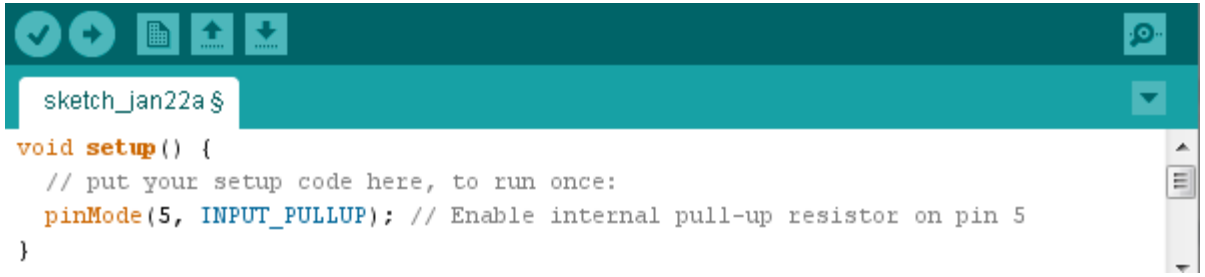
I have a preference towards pull-down resistors because it makes more sense to me that a switch is 0V (LOW) when open and 3.3 (HIGH) when closed.

See: https://learn.sparkfun.com/tutorials/switch-basics/all; https://learn.sparkfun.com/tutorials/pull-up-resistors/all

DIGITAL INPUT

MANY MCUS HAVE PULL-UP RESISTORS BUILT IN

Pull-up resistors are so commonly needed that many MCUs like the ATmega328 microcontroller have internal pull-up resistors (20kΩ) that can be enabled or disabled.

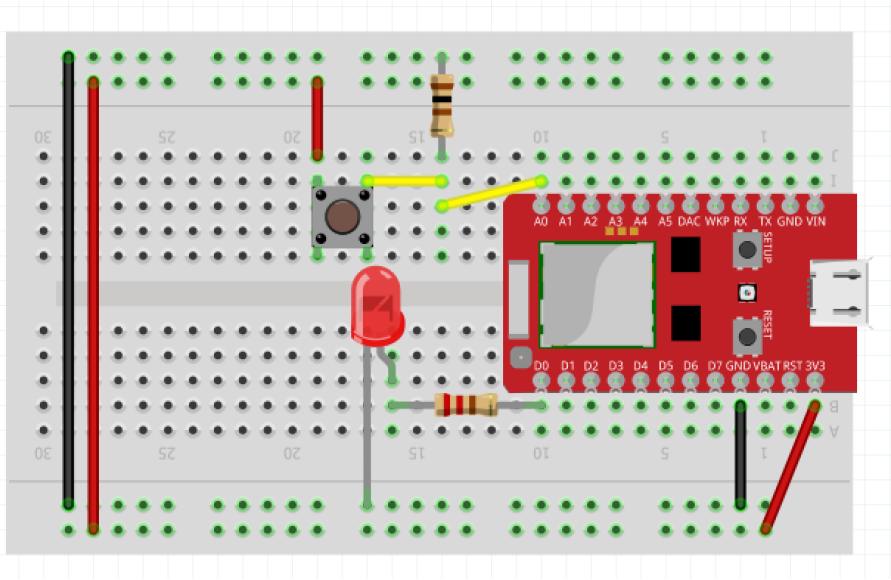


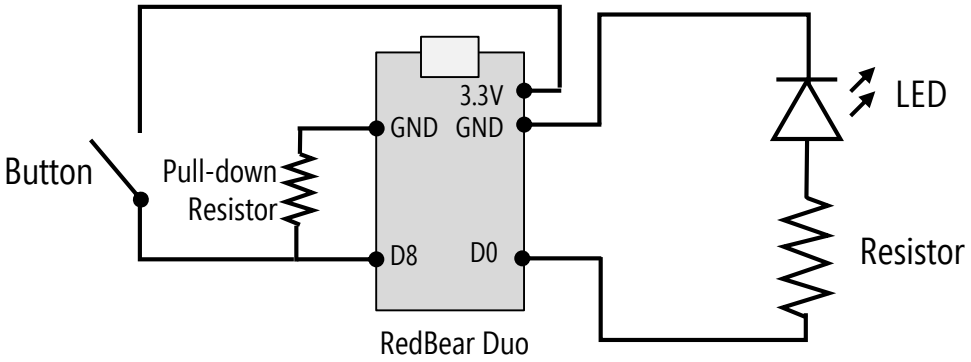
When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to be read as HIGH when the switch is open and LOW when the switch is pressed (which is somewhat counterintuitive!).

DIGITAL INPUT

TURN ON/OFF LED WITH A BUTTON

Circuit: Control LED via a button





Now we just have to write the Arduino code to take in the button state and turn on/off the LED accordingly.

ACTIVITY: LEARN DIGITAL INPUT

DIGITAL INPUT: DIGITALREAD()

Reads the value from a specified digital ping and returns either HIGH or LOW

Syntax

```
digitalRead(pin)
```

Parameters

`pin`: the digital I/O pin you want to read

Returns

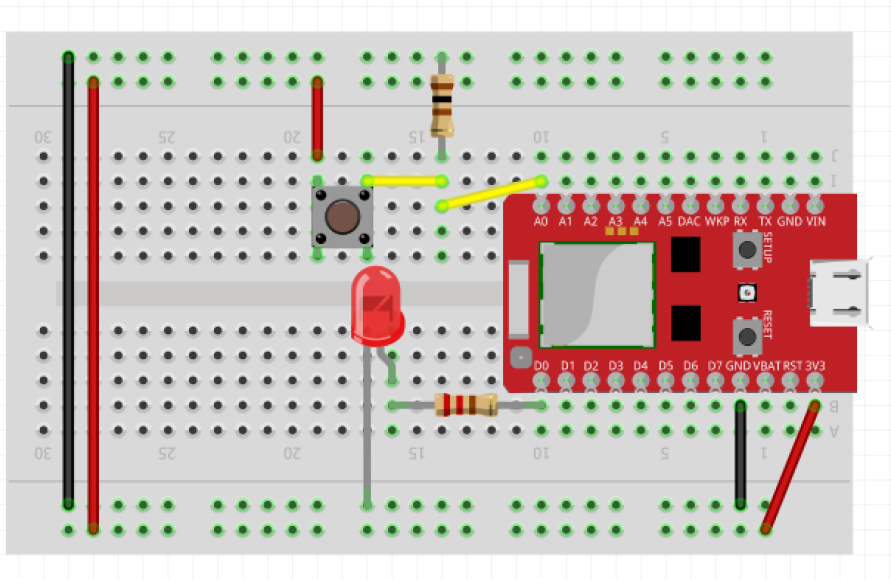
HIGH or LOW

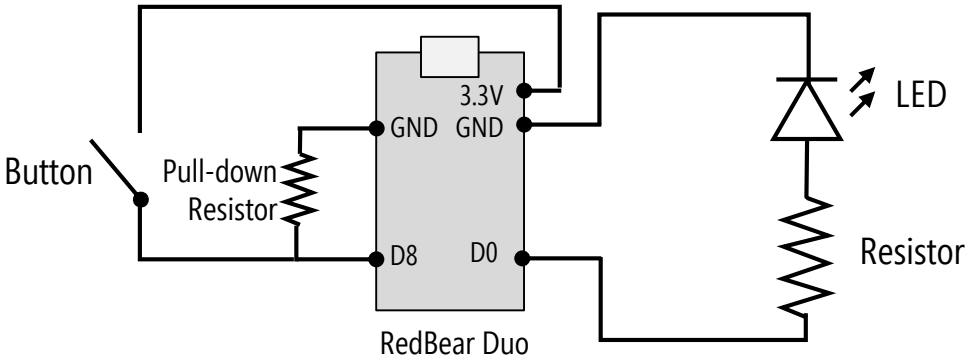
https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/

DIGITAL INPUT

TURN ON/OFF LED WITH A BUTTON

Circuit: Control LED via a button





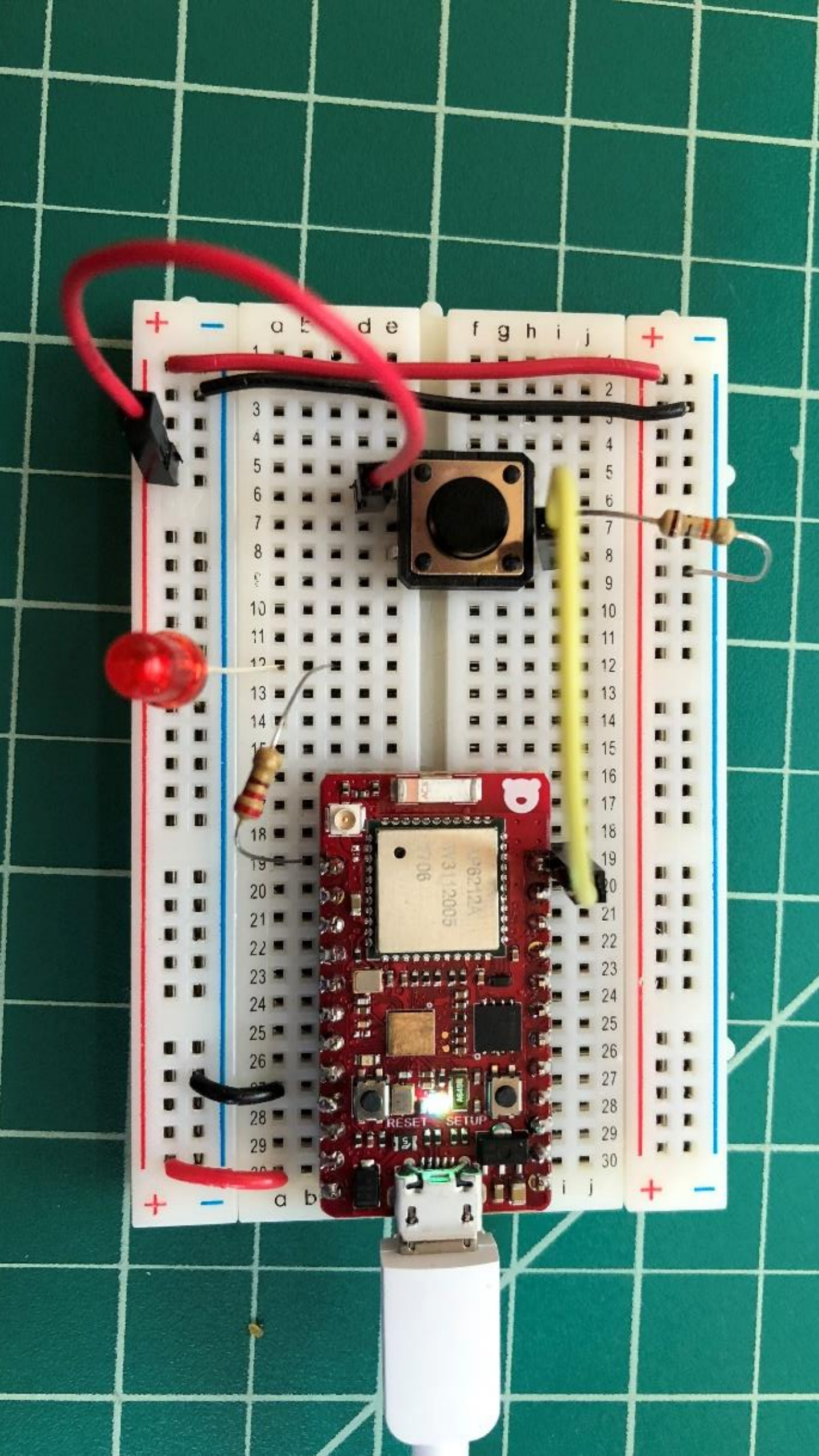
Code: Read button state and set LED on/off accordingly

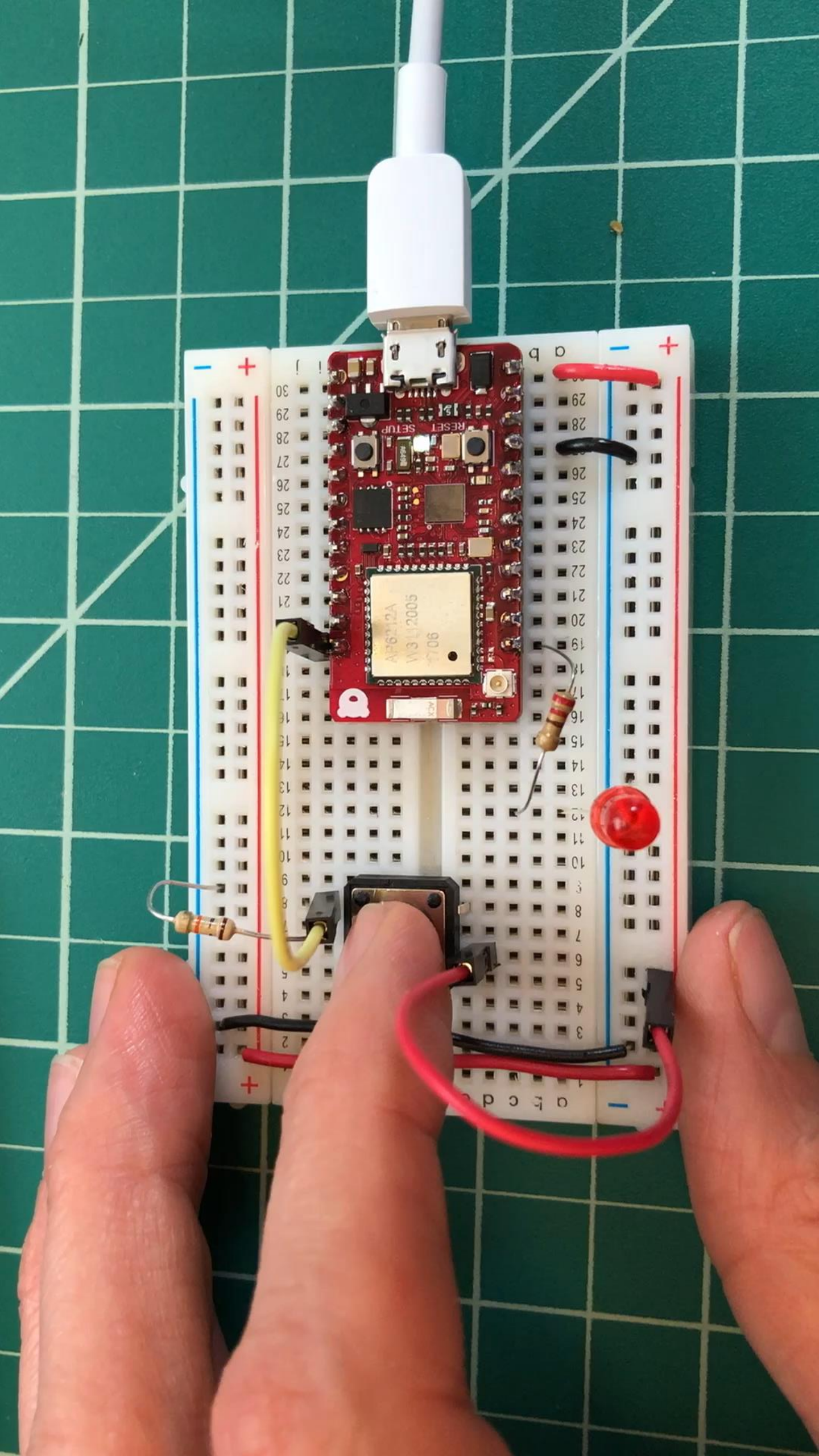
RedBearDuoReadButtonSetLED | Arduino 1.8.5

File Edit Sketch Tools Help

RedBearDuoReadButtonSetLED §

```
/*
 * This example reads in a button input on D8 (with a pull-down resistor configuration)
 * and turns on/off an LED on D0 accordingly
 *
 * By Jon Froehlich for CSE590
 * http://makeabilitylab.io
 */
SYSTEM_MODE(MANUAL);

const int LED_OUTPUT_PIN = D0;
const int BUTTON_INPUT_PIN = D8;

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

void loop() {

  // read the button value. It will be HIGH when pressed and
  // LOW when not pressed
  int buttonVal = digitalRead(BUTTON_INPUT_PIN);

  // Write out HIGH or LOW
  digitalWrite(LED_OUTPUT_PIN, buttonVal);

  // Check for new input every 100ms
  delay(100);
}
```


RESET
SETUP


RESET
SETUP