Intro to Lighting

With Arduino

Version: 4

Last updated: Jan 30, 2016

Social media

Vancouver Hackspace
@VHS http://hackspace.ca

Steven Smethurst (Funvill)
@Funvill http://abluestar.com

Slides and Example source code

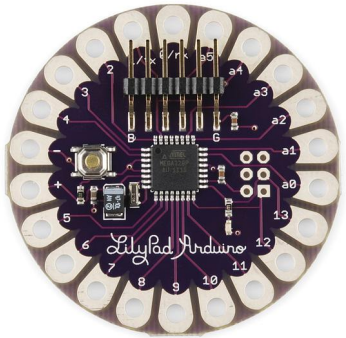
https://github.com/funvill/ArduinoLightingWorkshop/

What you should have in your kit

What is an Arduino?

- Open source microcontroller
- Many hardware configurations to choose from
- Great for prototyping

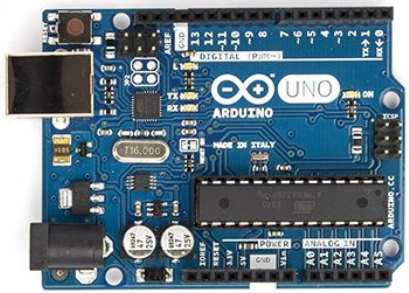
Board Form Factors



LilyPad

lilypadarduino.org

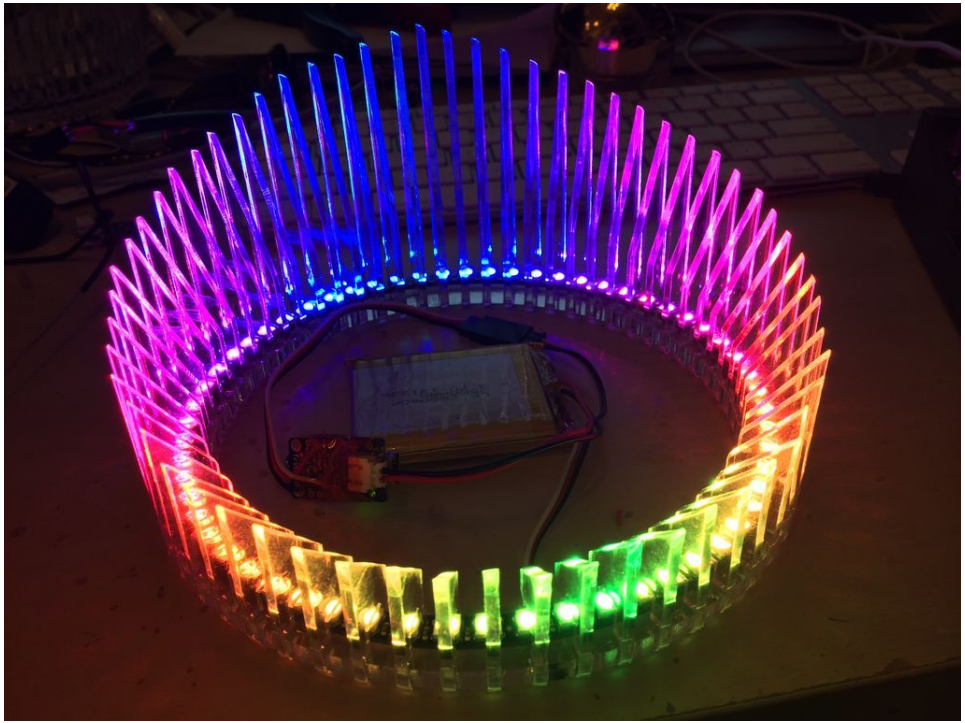
Nano



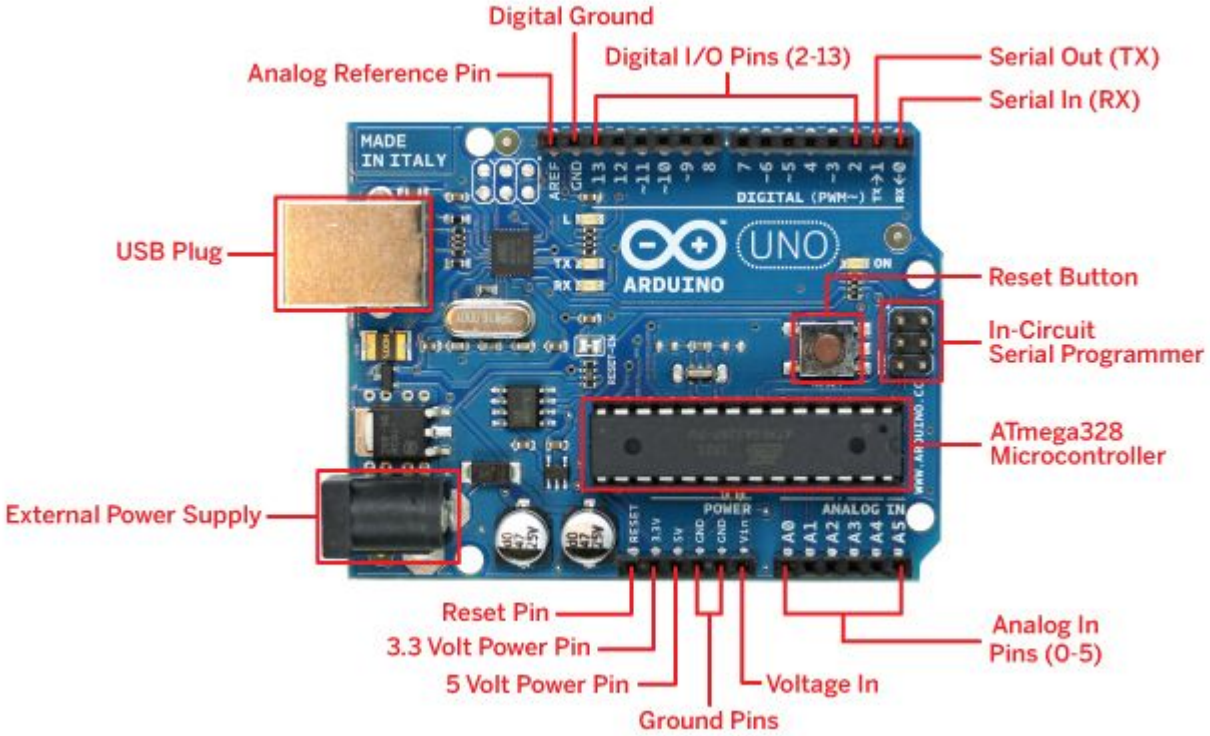
Uno

What can I make with an Arduino?

- Air Pollution Monitor
- Turn Signal Bike Jacket
- Tweeting Coffee Pot
- Robots
- LED “Laser” crown
- ...and so much more!



Anatomy of an Arduino

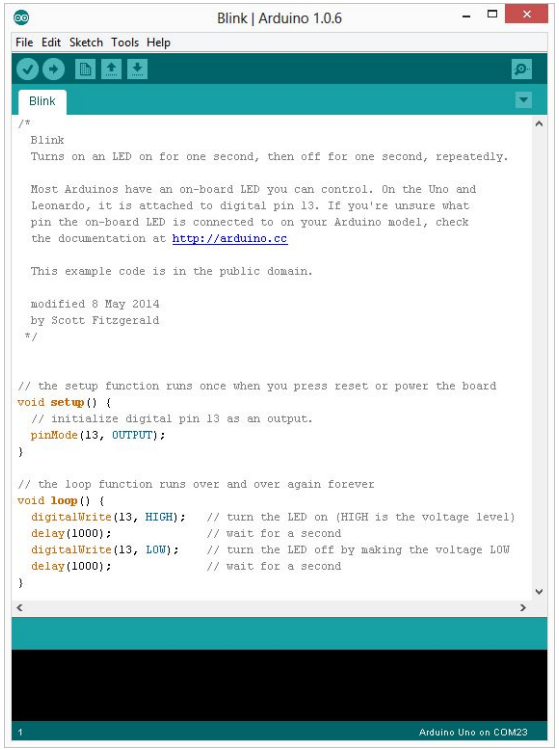


Your arduino might be slightly different.

Setting up for Programming

1. Install Arduino Editor (http://arduino.cc/)
2. Plug in your Board to USB
3. Select your board type
   Tools ⇒ Board ⇒ Arduino Leonardo
   (Yours might be different)
4. Select connection type
   Tools ⇒ Port ⇒ COM1 (Yours might be different)

Load the Blink sketch



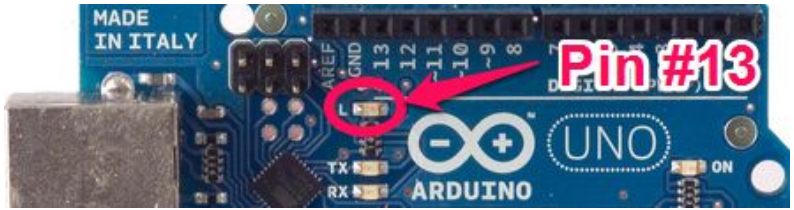
1. File ⇒ Examples ⇒ 01.Basics ⇒ Blink

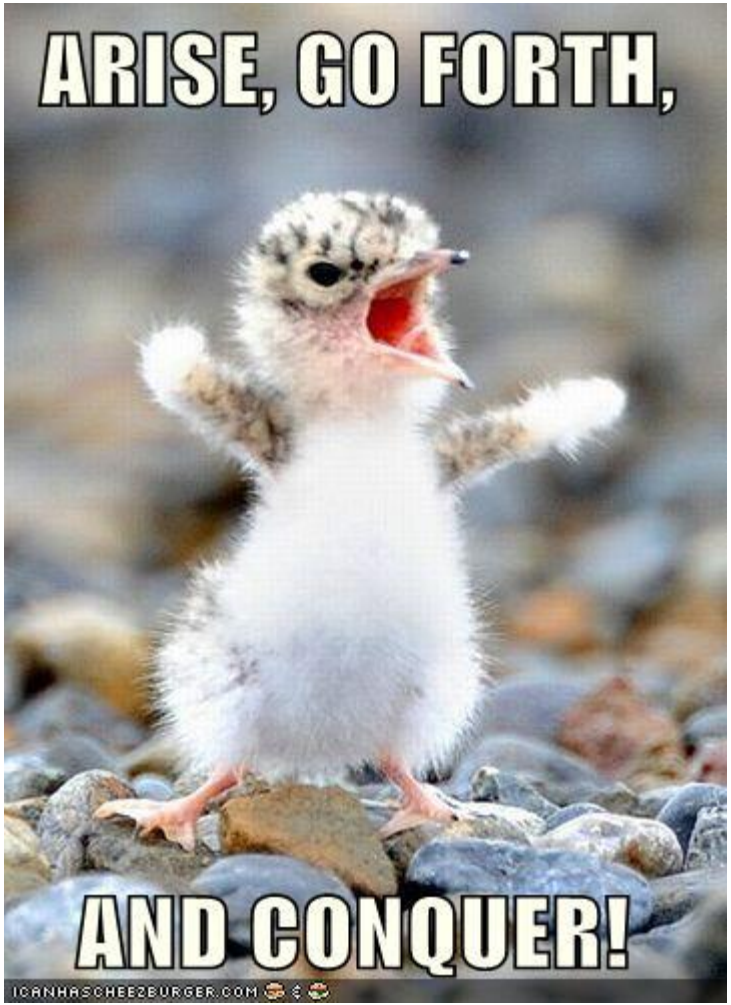
There are many other example sketches that you can load from the example folder.

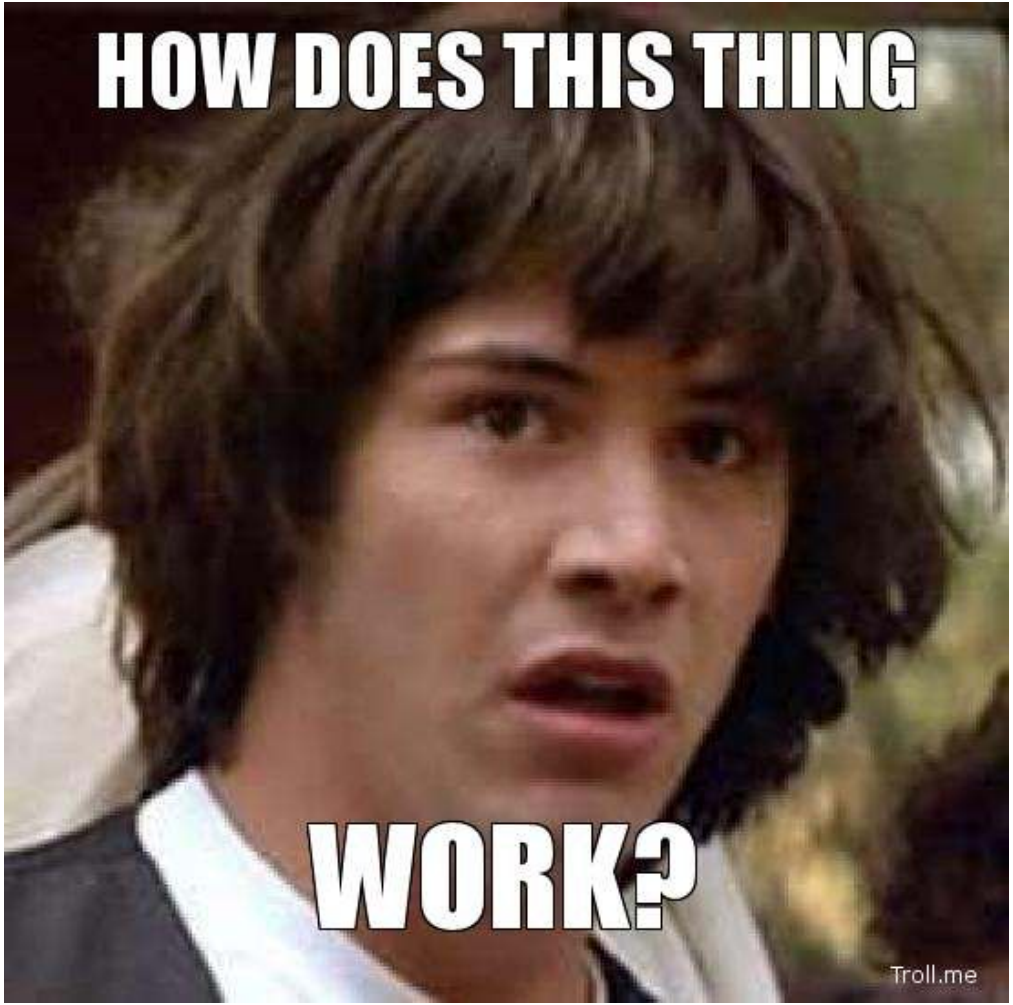
Upload !

1. Click the upload button

The LED should be blinking!




ARISE, GO FORTH,
AND CONQUER!
ICANHASCHEEZBURGER.COM


HOW DOES THIS THING
WORK?
Troll.me

Blink Sketch

File ⇒ Examples ⇒ 01.Basics ⇒ Blink)

```
// The setup function runs once when you press reset or power the board
void setup() {
  pinMode(13, OUTPUT);    // Initialize digital pin 13 as an output.
}

// The loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // Turn the LED on (HIGH is the voltage level)
  delay(1000);            // Wait for a second
  digitalWrite(13, LOW);  // Turn the LED off by making the voltage LOW
  delay(1000);            // Wait for a second
}
```

Functions

Segmenting code into functions allows a programmer to create modular pieces of code that perform a defined task and then return to the area of code from which the function was "called".

Functions has several advantages:

- Stay organized
- Functions make the whole sketch smaller and more compact because sections of code are reused many times.
- They make it easier to reuse code in other programs by making it more modular, and as a nice side effect, using functions also often makes the code more readable.

Function: digitalWrite()

Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V (ground) for LOW.

Syntax:

digitalWrite(pin, value)

Parameters:

pin: The pin number
value: HIGH or LOW

Function: delay()

Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

```
 1,000 =  1 sec
30,000 = 30 sec
60,000 =  1 min
```

Syntax:

delay(ms)

Parameters:

ms: the number of milliseconds to pause (unsigned long)

Try this

- Change the blink sketch to make the LED blink faster or slower. *Hint:* You can change the value in the delay function.
- Change the blink sketch to make a pattern. Do SOS in morse code. Three short blinks, three long blinks, three short blinks. *Hint:* Copy and paste the code a few times.

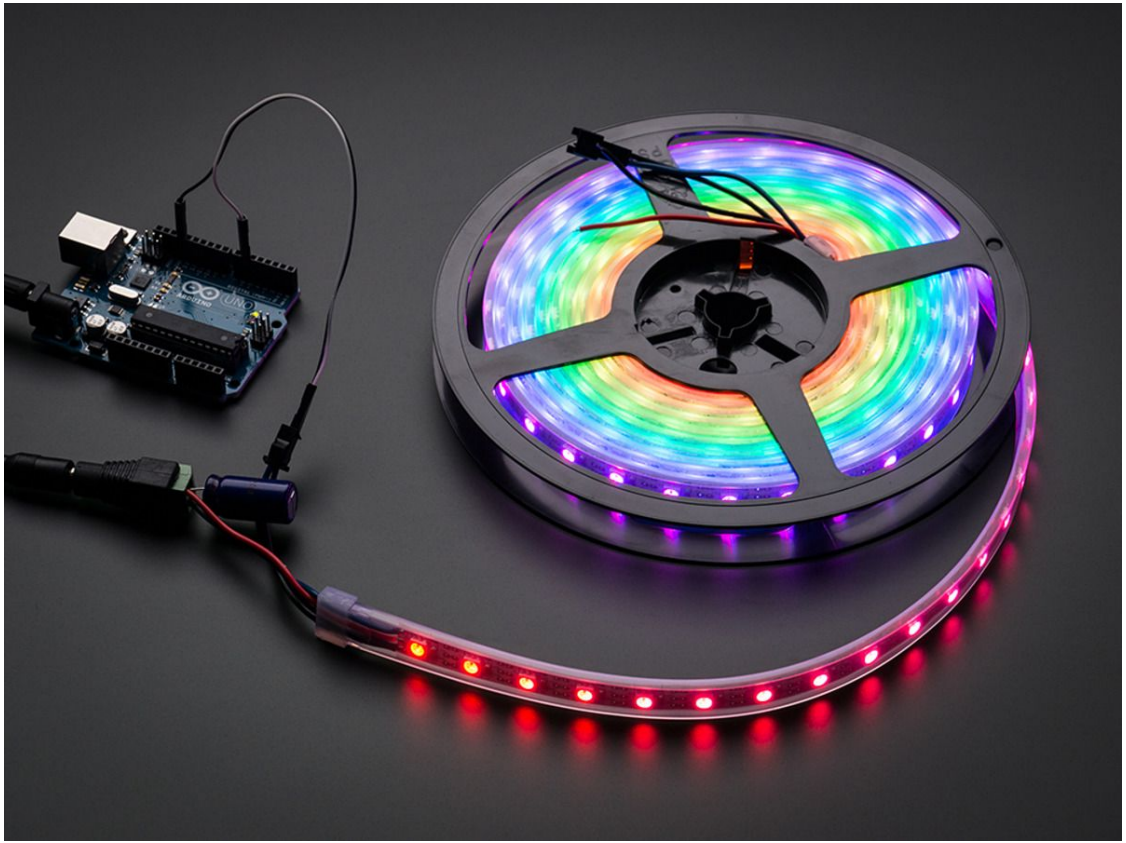
Blink Faster, and Slower

```
void setup() {
  pinMode(13, OUTPUT);     // Initialize digital
                           // pin 13 as an output.
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // Turn the LED on
  delay(500);              // Wait for half second
  digitalWrite(13, LOW);  // Turn the LED off
  delay(2000);             // Wait for two second
}
```

Make your own functions

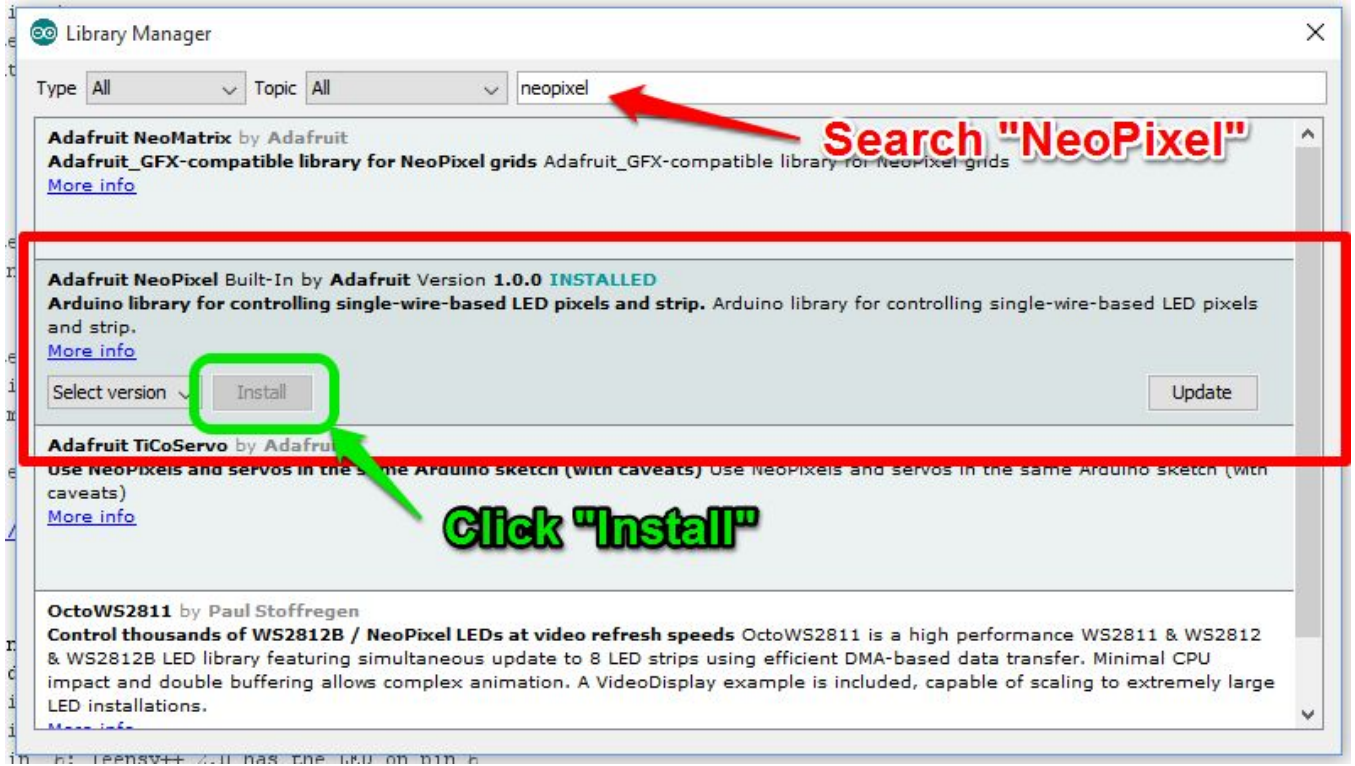
```
void setup() {
  pinMode(13, OUTPUT);    // Initialize digital
                          // pin 13 as an output.
}

// the loop function runs over and over again forever
void loop() {
  Blink();
}

void Blink() {
  digitalWrite(13, HIGH); // Turn the LED on
  delay(500);             // Wait for half second
  digitalWrite(13, LOW);  // Turn the LED off
  delay(1000);            // Wait for one second
}
```

Let’s Add the LED strip

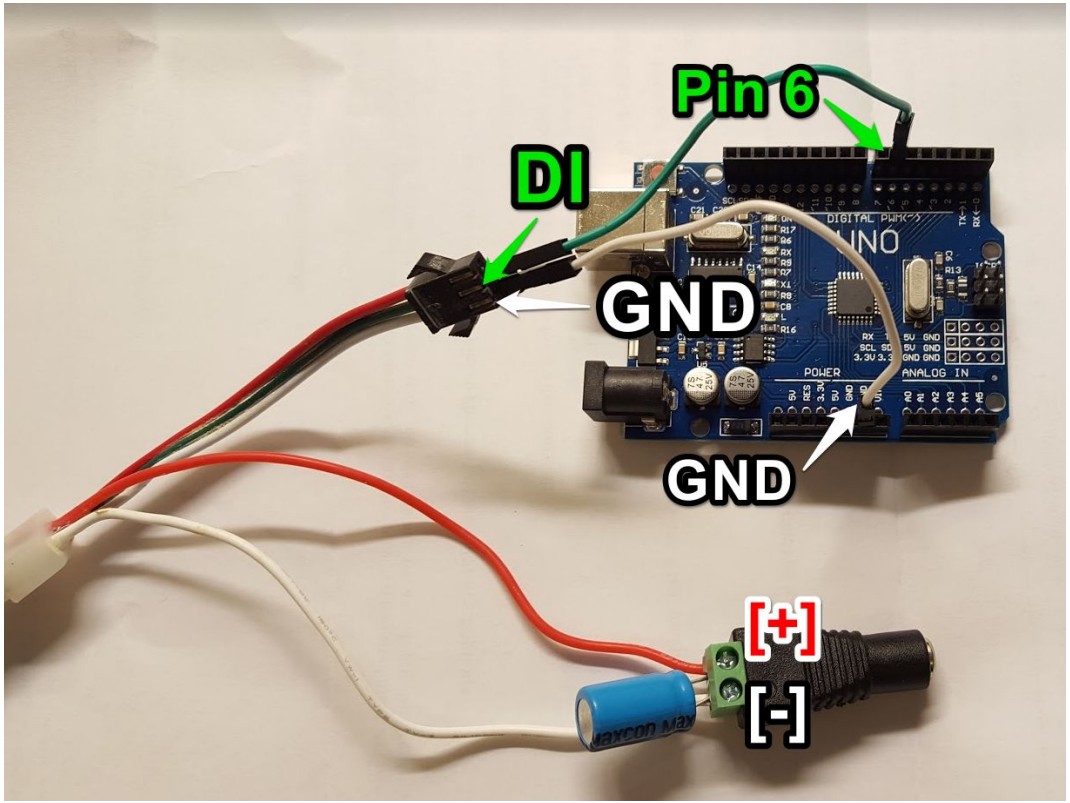


Adding NeoPixel library to Arduino

Sketch => Include Libraries => Manage Libraries



Connect the NeoPixel LEDs

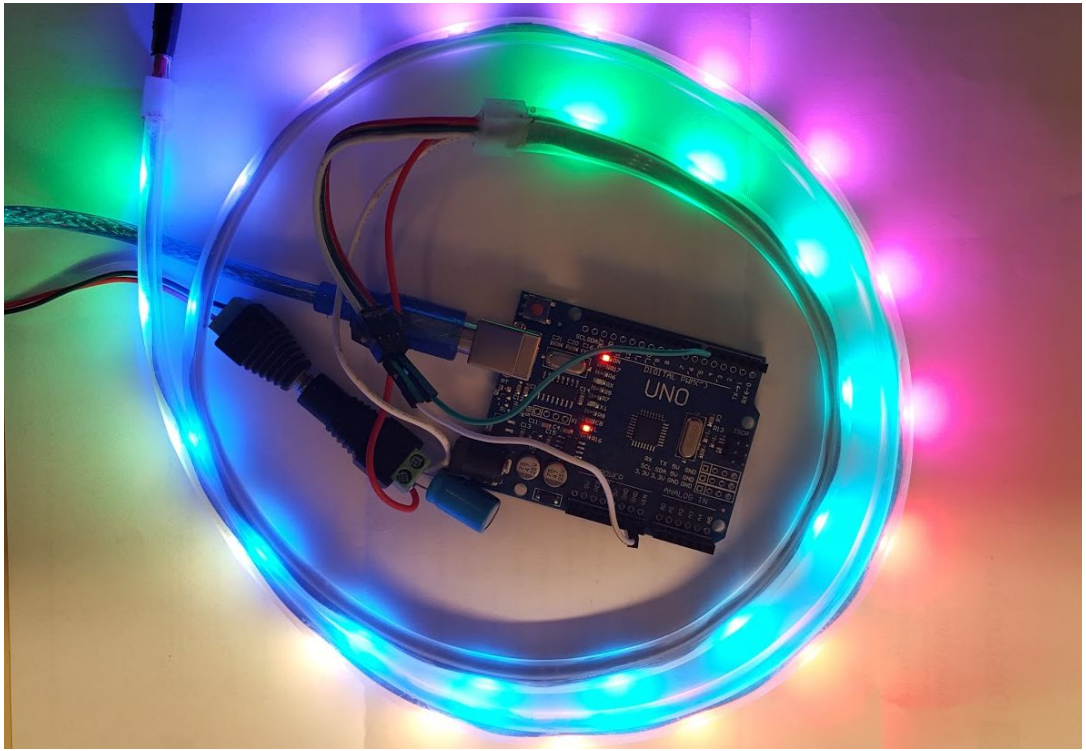


Arduino		LED Strip
GND	=>	GND
PIN D6	=>	DI
+5V	=>	+5V

Add 1000uf cap
between [+] and [-]

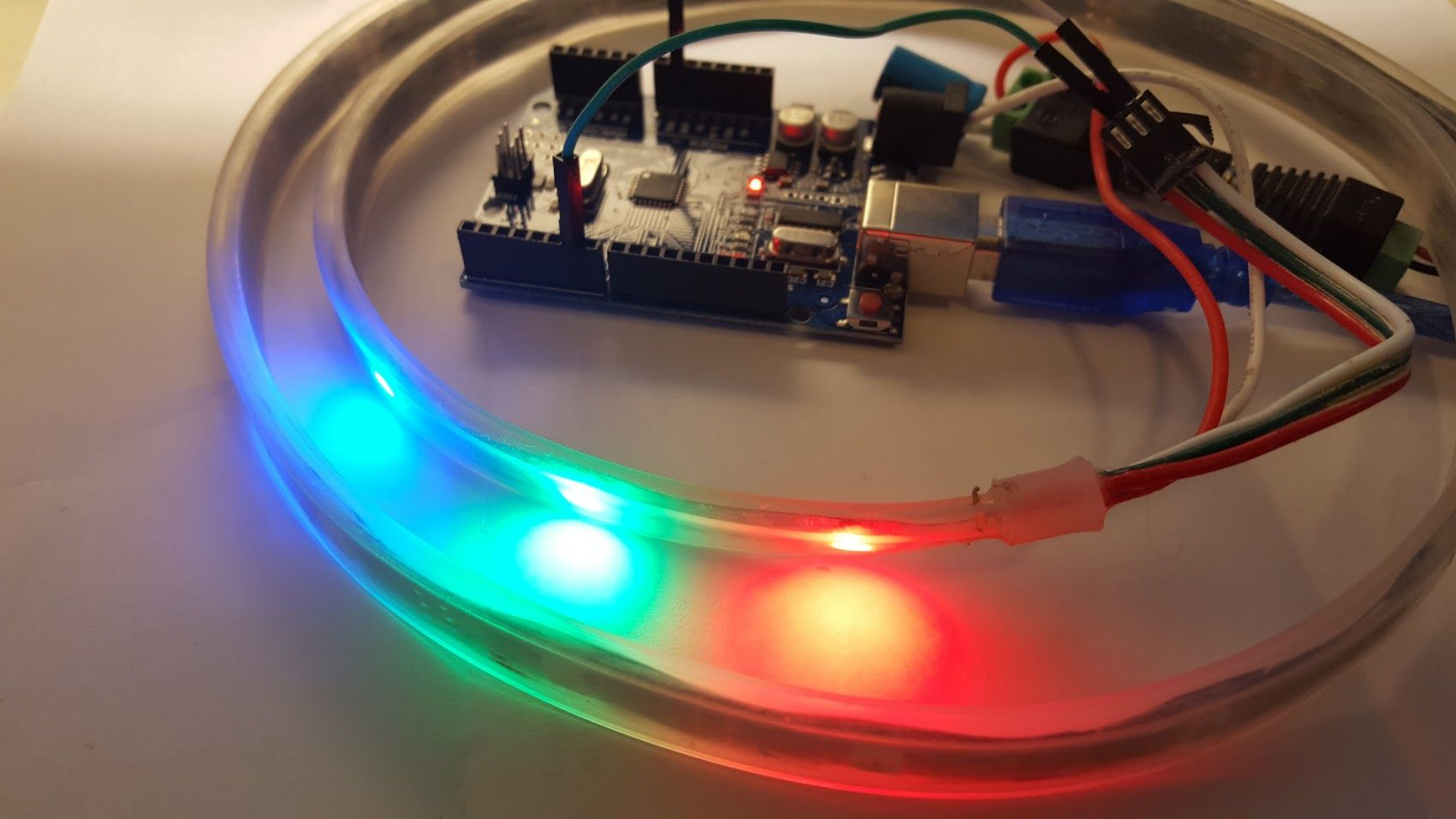
Test the NeoPixel LEDs

1. File => Examples => Adafruit Neopixel => Strandtest



Start a new sketch (SimpleLEDStrip.ino)

```
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

const int NEOPIXEL_PIN    = 6 ;
const int NUMPIXELS       = 30 ;

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMPIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin(); // This initializes the NeoPixel library.
}

void loop() {
  strip.setPixelColor(0, strip.Color(150,0,0)); // Moderately bright red color.
  strip.setPixelColor(1, strip.Color(0,150,0)); // Moderately bright green color.
  strip.setPixelColor(2, strip.Color(0,0,150)); // Moderately bright blue color.
  strip.show(); // This sends the updated pixel color to the hardware.
}
```



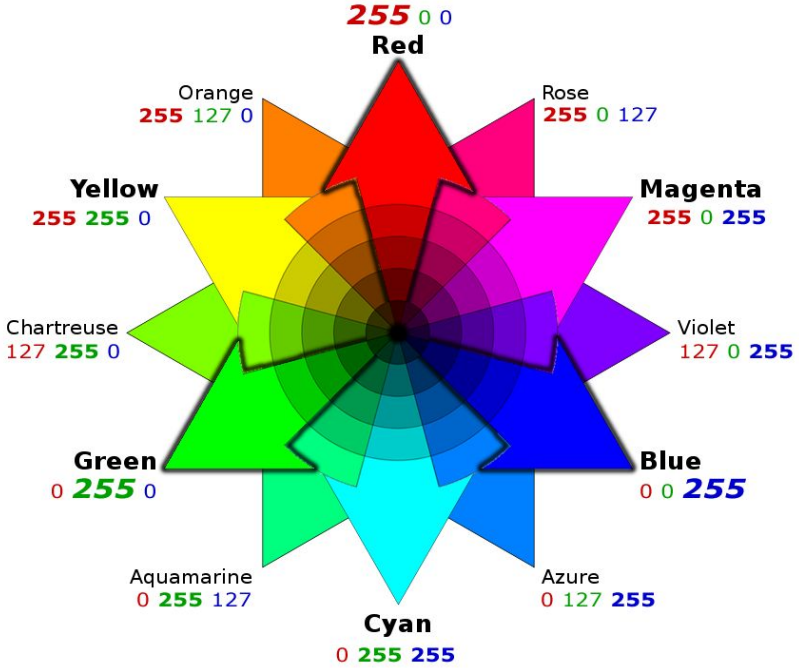
Function: strip.Color(red, green, blue)

Parameters:

red, green, blue: Arguments are the pixel color, expressed as red, green and blue brightness levels, where 0 is dimmest (off) and 255 is maximum brightness.

```
purple     = strip.Color(128,   0, 128)
skyblue    = strip.Color(135, 206, 235)
limegreen  = strip.Color( 50, 205,  50)
darkorange = strip.Color(255, 140,   0)
```

RGB color wheel



The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue

http://www.colorspire.com

Function: strip.setPixelColor(n, color);

Parameters:

n: Is the pixel number along the strip, starting from 0 closest to the Arduino. If you have a strip of 30 pixels, they're numbered 0 through 29. It's a computer thing.

color: Is a 32-bit type that merges the red, green and blue values into a single number. strip.Color(255, 0, 255);

```
strip.setPixelColor(0, strip.Color(150,0,0));
strip.setPixelColor(1, strip.Color(0,150,0));
strip.setPixelColor(2, strip.Color(0,0,150));
```

Function: strip.show();

This updates the whole strip at once, and despite the extra step is actually a good thing. If every call to setPixelColor() had an immediate effect, animation would appear jumpy rather than buttery smooth.

```
strip.show();
```

Try this

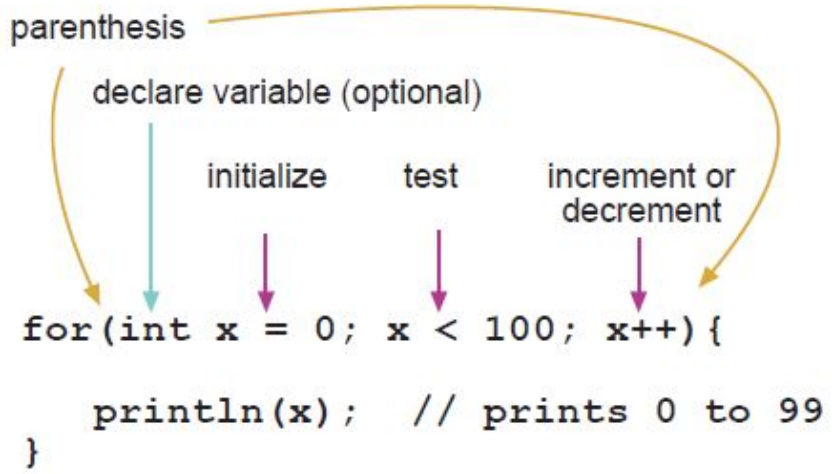
- Make an LED Yellow
- Make an LED Magenta
- Do a pattern of Yellow, Green, Blue, Magenta, Purple, Red

For Statement

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

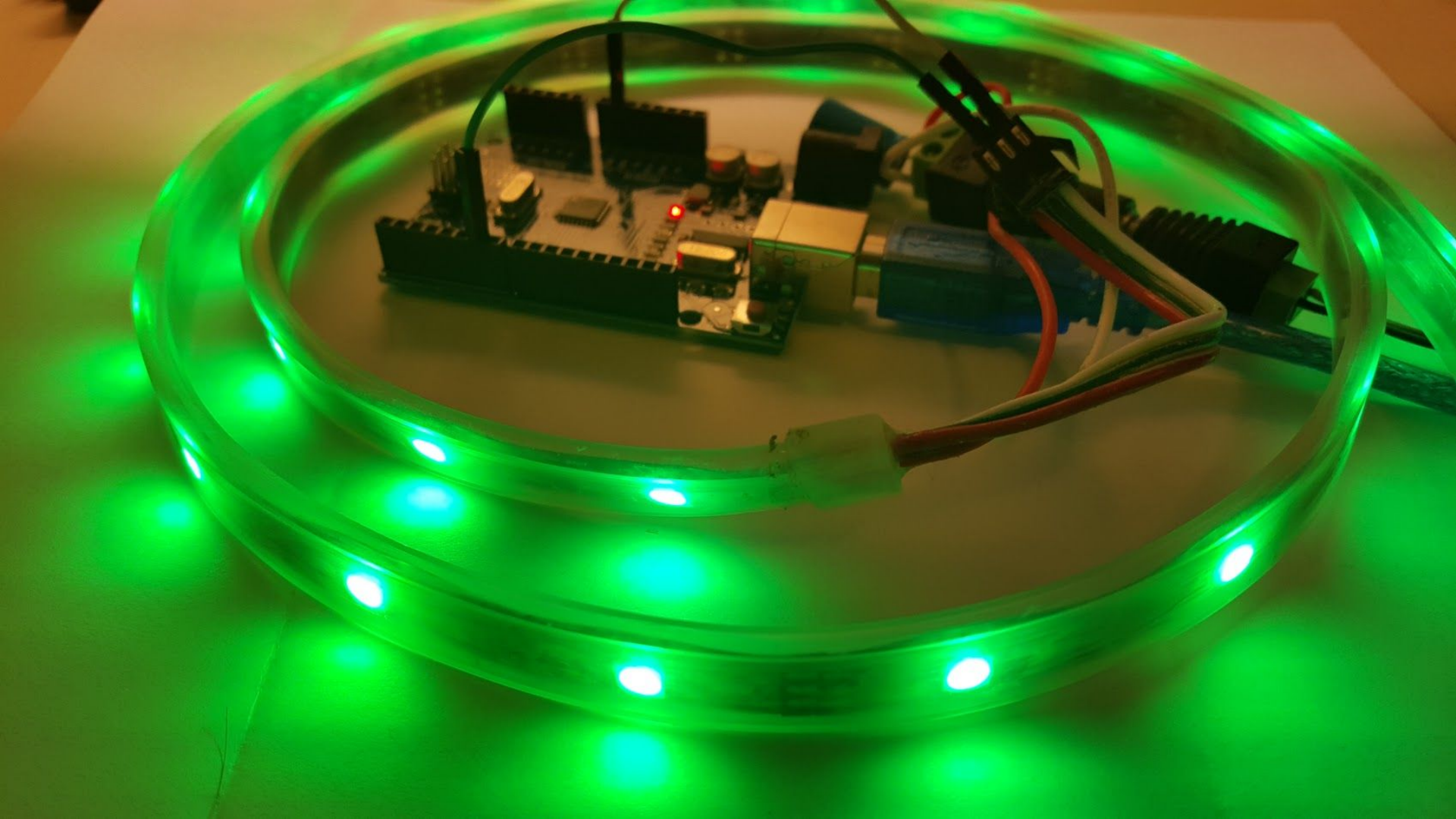
There are three parts to the for loop header:

```
for (int i=0; i <= 25; i += 5 ) {
 strip.setPixelColor(i,
           strip.Color(0,150,0));
}
```



Make all the LEDs Green (AllLEDsGreen.ino)

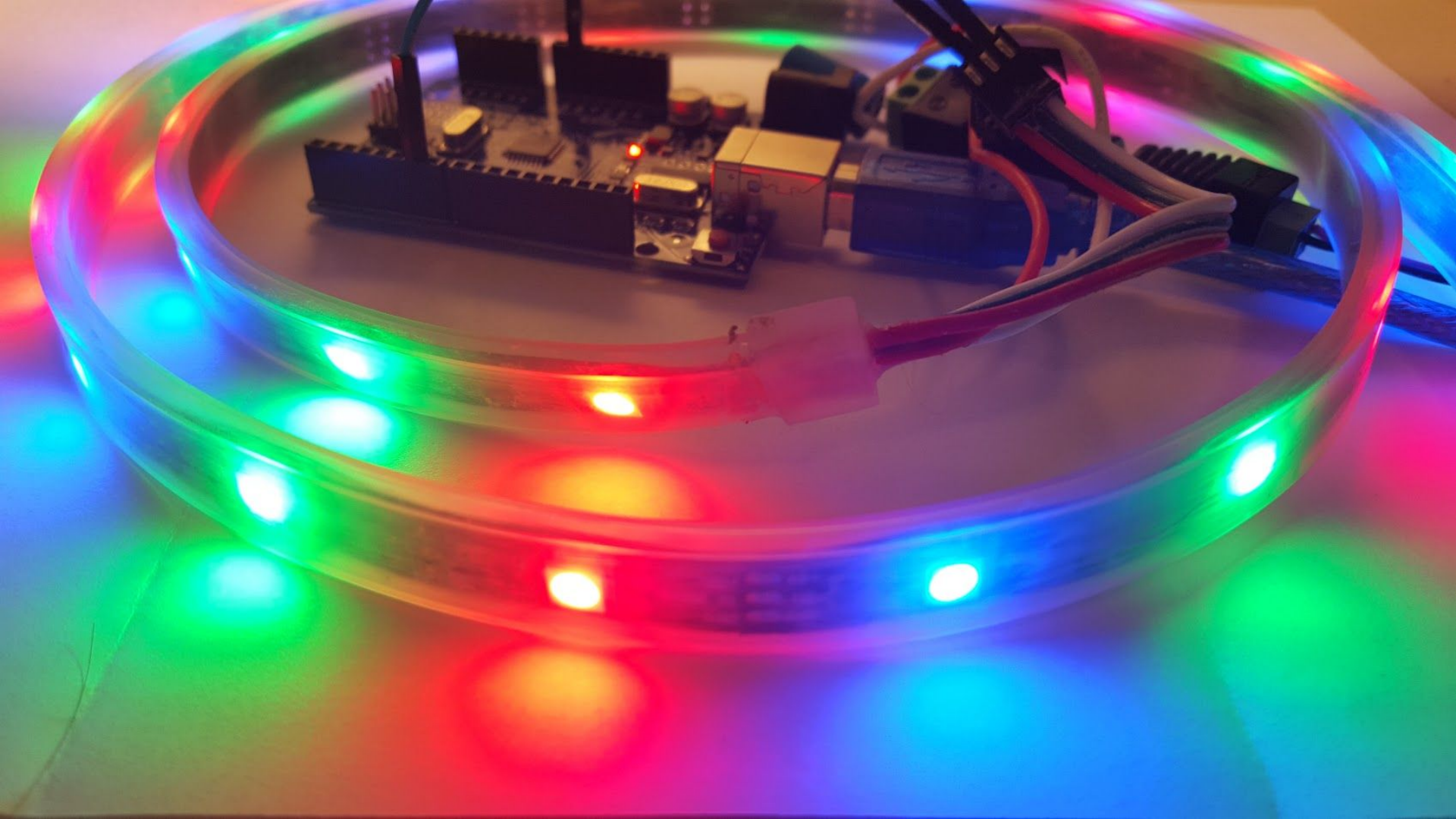
```
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>
const int LEDS_PIN       =  6 ;
const int NUMPIXELS      = 30 ;

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMPIXELS, LEDS_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin(); // This initializes the NeoPixel library.
}

void loop() {
  for( int pixel = 0; pixel < strip.numPixels(); pixel++ ) {
     strip.setPixelColor(pixel, strip.Color(0,150,0)); // Moderately bright green color.
  }
  strip.show(); // This sends the updated pixel color to the hardware.
}
```



Make a pattern (MakeAPattern.ino)

Repeat the the sequence of red, green, blue across the entire strip of LEDs

```
void loop() {
  // Instead of incrementing the for loop by 1, increment by 3
  for( int pixel = 0; pixel < strip.numPixels(); pixel +=3 ) {
     strip.setPixelColor(pixel  , strip.Color(150,0,0)); // bright red color.
     strip.setPixelColor(pixel+1, strip.Color(0,150,0)); // bright green color.
     strip.setPixelColor(pixel+2, strip.Color(0,0,150)); // bright blue color.
  }
  strip.show(); // This sends the updated pixel color to the hardware.
}
```



Estimating Power Requirements

Each individual NeoPixel draws up to 60 milliamps at maximum brightness white (red + green + blue). In actual use though, it's rare for all pixels to be turned on that way. When mixing colors and displaying animations, the current draw will be much less. It's impossible to estimate a single number for all circumstances, but we've been using 1/3 this (20 mA per pixel) as a gross rule of thumb with no ill effects. To estimate power supply needs, multiply the number of pixels by 20, then divide the result by 1,000

30 LEDs * (20 mA) / 1000 = 0.6 Amp Probably okay

30 LEDs * (60 mA) / 1000 = 1.8 Amp Guaranteed to work

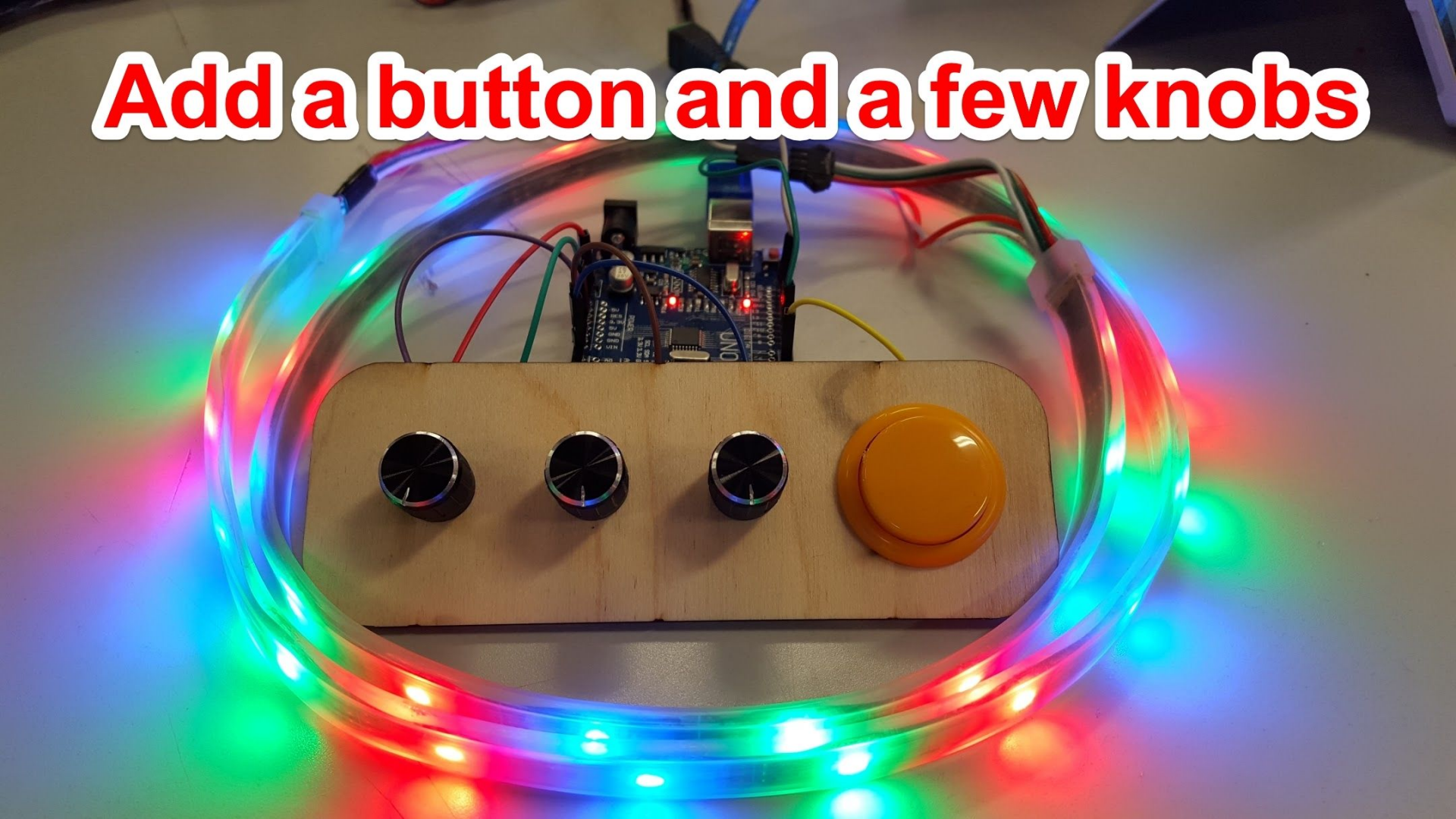
Voltage drops across long LED strips

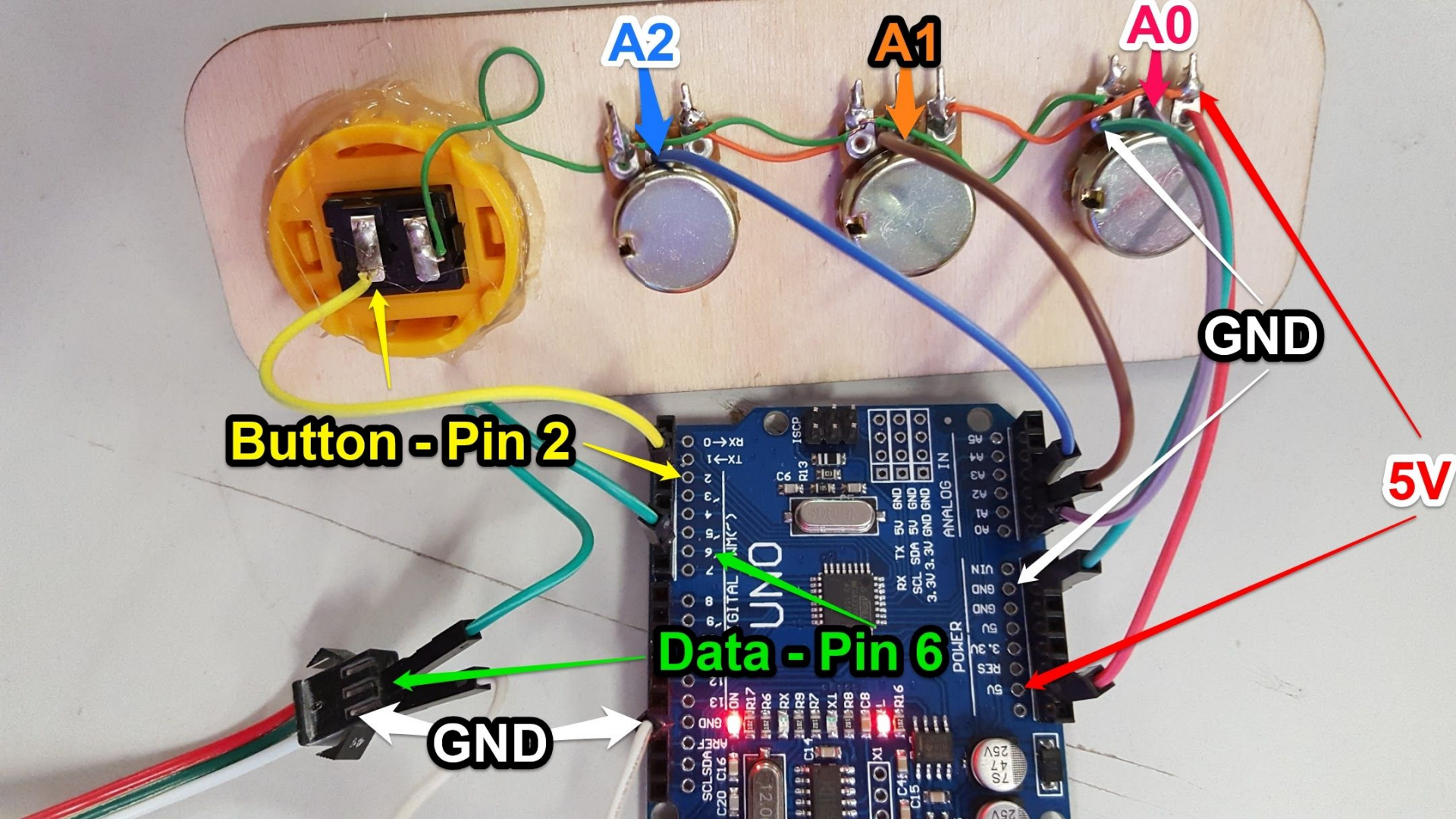
The longer a wire is, the more resistance it has. The more resistance, the more voltage drops along its length. If voltage drops too far, the color of NeoPixels can be affected. Consider a full 4 meter reel of NeoPixels. With 5V applied at one end of the strip, for those pixels closest to this end, power traverses only a few inches of copper. But at the far end of the strip, power traverses 8 meters of copper — 4 meters out on the +5V line, 4 meters back on the ground line. Those furthest pixels will be tinted brown due to the voltage drop (blue and green LEDs require higher voltage than red).

Power taps for long series of LEDs

The way to resolve this is to separate the LEDs power connections into smaller sections. 1 meter or 30 LEDs work best but you can use up to 3 meters and 90 LEDs without too much effect.

http://www.seeedstudio.com/depot/AllPixel-Power-Tap-Kit-p-2380.html


Add a button and a few knobs


A2
A1
A0
GND
5V
Button - Pin 2
Data - Pin 6
GND

The button

Pushbuttons or switches connect two points in a circuit when you press them.

Variable Declaration

A variable is a way of naming and storing a value for later use by the program, such as data from a sensor or an intermediate value used in a calculation.

Declaring Variables

Before they are used, all variables have to be declared. Declaring a variable means defining its type, and optionally, setting an initial value (initializing the variable). Variables do not have to be initialized (assigned a value) when they are declared, but it is often useful.

```
int inputVariable1;
int inputVariable2 = 0;  // both are correct
```

Function: digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW. If the pin isn't connected to anything, digitalRead() can return either HIGH or LOW (and this can change randomly).

```
int inPin = 7; // pushbutton connected
               // to digital pin 7
void setup() {
   // sets the digital pin 7 as input
   pinMode(inPin, INPUT);
}
void loop() {
   // read the input pin
   int val = digitalRead(inPin);
}
```

Syntax

digitalRead(pin)

Parameters

pin: The number of the digital pin you want to read (int)

Returns

HIGH or LOW

If Statement

The if() statement is the most basic of all programming control structures. It allows you to make something happen or not, depending on whether a given condition is true or not. It looks like this:

```
if (someCondition) {
   // do stuff if the condition is true
}
```

There is a common variation called if-else that looks like this:

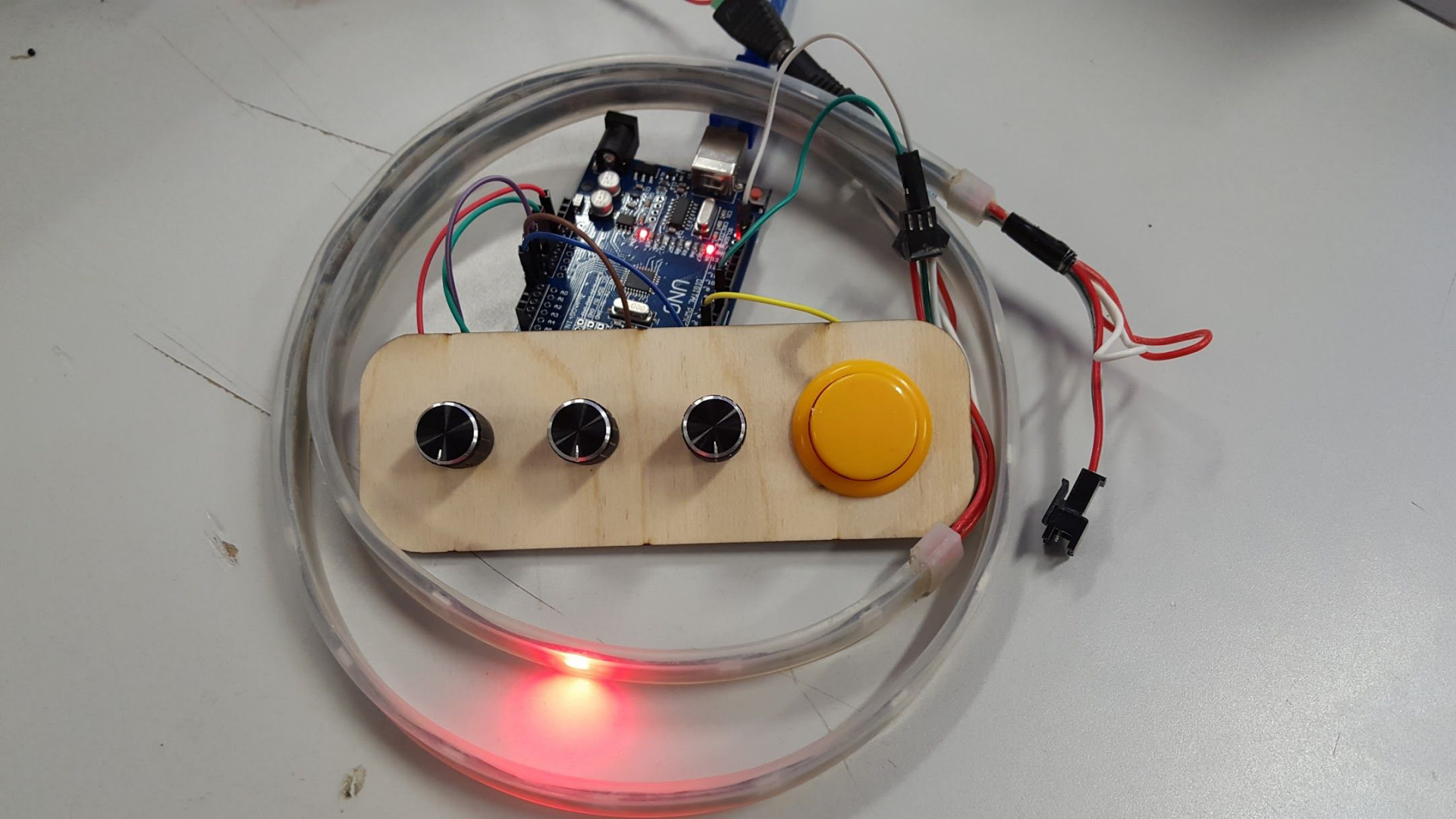
```
if (someCondition) {
   // do stuff if the condition is true
} else {
   // do stuff if the condition is false
}
```

Button Count (ButtonCount.ino)

```
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

const int BUTTON_PIN     =  2 ;  // The number of the pushbutton pin
const int NEOPIXEL_PIN   =  6 ;  // The number of the LED pin
const int NUMPIXELS      = 30 ;  // The number of LEDs in the LED strip.

int  pixelOffset         =  0 ;
int  lastButtonState     = HIGH; // The previous reading from the input pin

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMPIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin(); // This initializes the NeoPixel library.
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Initialize the button pin as an input
}
```

Button Count (ButtonCount.ino)

```
void loop() {
  int buttonState = digitalRead(BUTTON_PIN); // read the state of the button.
  if (buttonState != lastButtonState ) { // Check to see if the button has changed state
    lastButtonState = buttonState ; // The button changed, update the last button state.
    if( buttonState == LOW ) { // check to see if the button is currently down.
      // Set the previous led to black.
      strip.setPixelColor(pixelOffset, strip.Color(0,0,0));

      pixelOffset++ ; // Increment the pixel offset
      if( pixelOffset >= NUMPIXELS ) {
        pixelOffset = 0 ;
      }

      strip.setPixelColor(pixelOffset, strip.Color(150,0,0)); // RED
      strip.show(); // This sends the updated pixel color to the hardware.
    }
  }
}
```



Try This:

- Make all the LEDs green and count with a blue LED when the button is pressed.
- Make the LED turn red when the button is pressed and green when the button is not pressed.
- Make all the LEDs cycle through different colors when pressed.

Let’s add a knob (Potentiometer)

A potentiometer is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value.

Function: analogRead()

Reads the value from the specified analog pin. The Arduino board contains a 6 channel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.

```
analogRead(analogPin);
```

Syntax

analogRead(pin)

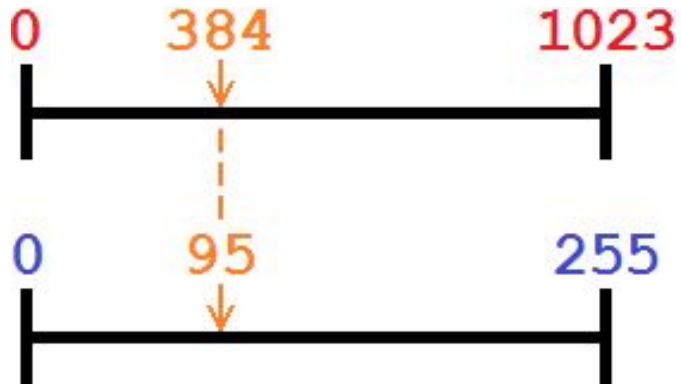
Parameters

pin: The number of the analog input pin to read from 0 to 5 on most boards.

Function: map()

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

```
output = map(value,0,1023,0,255);
output = map(value,0,1023,0,100);
```



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.

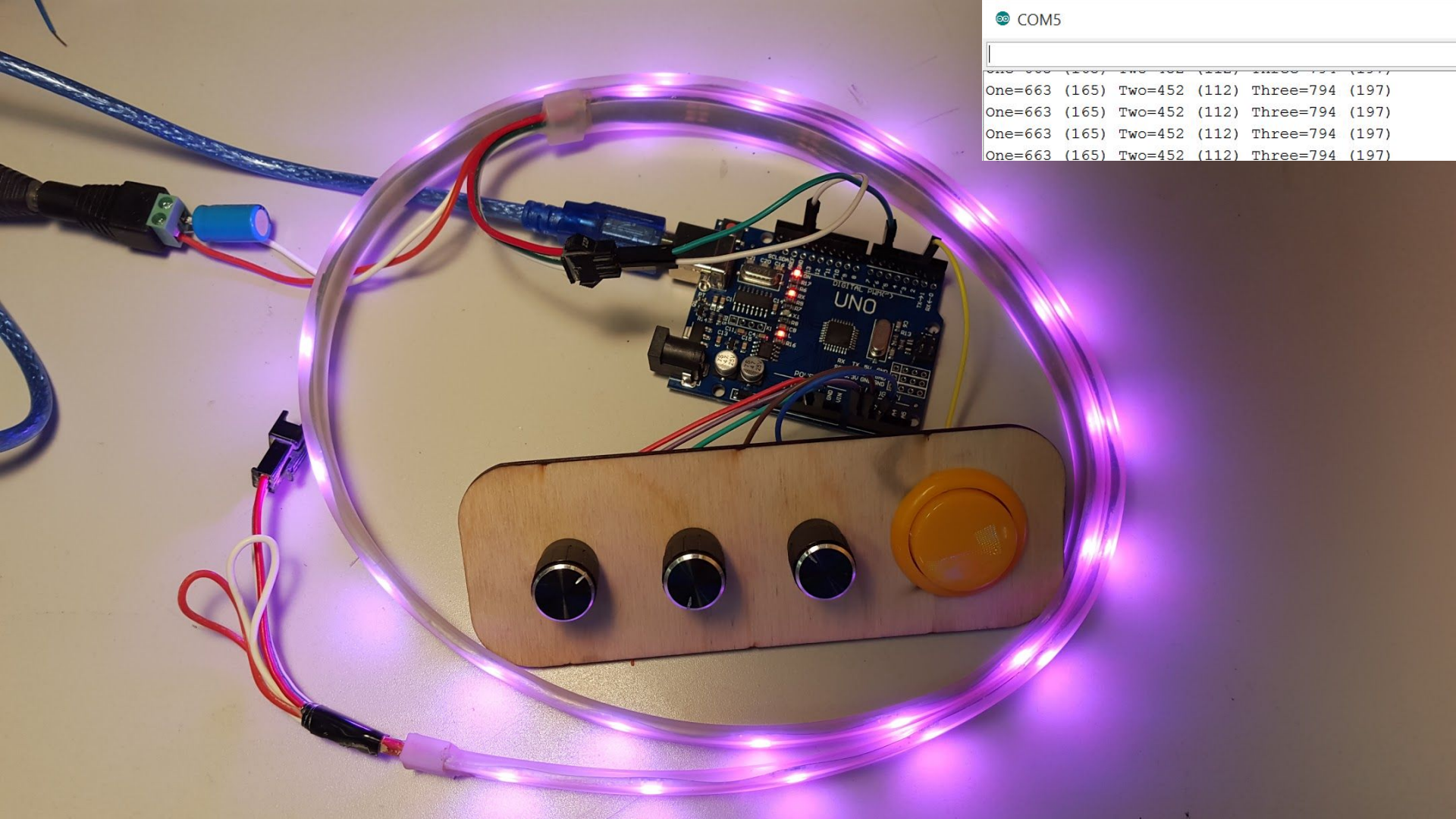
toHigh: The upper bound of the value's target range.

Knob change RGB (RGBWithKnobs.ino)

```
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

const int KNOB_ONE_PIN   = A0 ;
const int KNOB_TWO_PIN   = A1 ;
const int KNOB_THREE_PIN = A2 ;
const int BUTTON_PIN     =  2 ;  // The number of the pushbutton pin
const int NEOPIXEL_PIN   =  6 ;  // The number of the LED pin
const int NUMPIXELS      = 30 ;  // The number of LEDs in the LED strip.

Adafruit_NeoPixel strip = Adafruit_NeoPixel(NUMPIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin(); // This initializes the NeoPixel library.
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Initialize the pushbutton pin as an input
                                     // and high by default.
  Serial.begin(9600); // initialize serial communications at 9600 bps:
}
```

Knob change RGB (RGBWithKnobs.ino)

```
void loop() {
  int potOneValue   = analogRead(KNOB_ONE_PIN);
  int potTwoValue   = analogRead(KNOB_TWO_PIN);
  int potThreeValue = analogRead(KNOB_THREE_PIN);

  unsigned char red   = map(potOneValue,0,1023,0,255) ;
  unsigned char green = map(potTwoValue,0,1023,0,255) ;
  unsigned char blue  = map(potThreeValue,0,1023,0,255) ;

  // print the results
  Serial.print("One=" + String(potOneValue) + " (" + String(red) + ") ");
  Serial.print("Two=" + String(potTwoValue) + " (" + String(green) + ") ");
  Serial.println("Three=" + String(potThreeValue) + " (" + String(blue) + ") ");

  for( int pixel = 0 ; pixel < NUMPIXELS ; pixel++ ) {
    strip.setPixelColor(pixel, strip.Color(red,green,blue));
  }
  strip.show(); // This sends the updated pixel color to the hardware.
}
```


COM5
One=663 (165) Two=452 (112) Three=794 (197)
One=663 (165) Two=452 (112) Three=794 (197)
One=663 (165) Two=452 (112) Three=794 (197)
One=663 (165) Two=452 (112) Three=794 (197)
UNO

Try this:

- Using a knob make a red LED travel from one end of the strip to the other depending on the position of the knob.

Function: strip.setBrightness(brightness);

setBrightness() sets the max brightness of the LEDs.

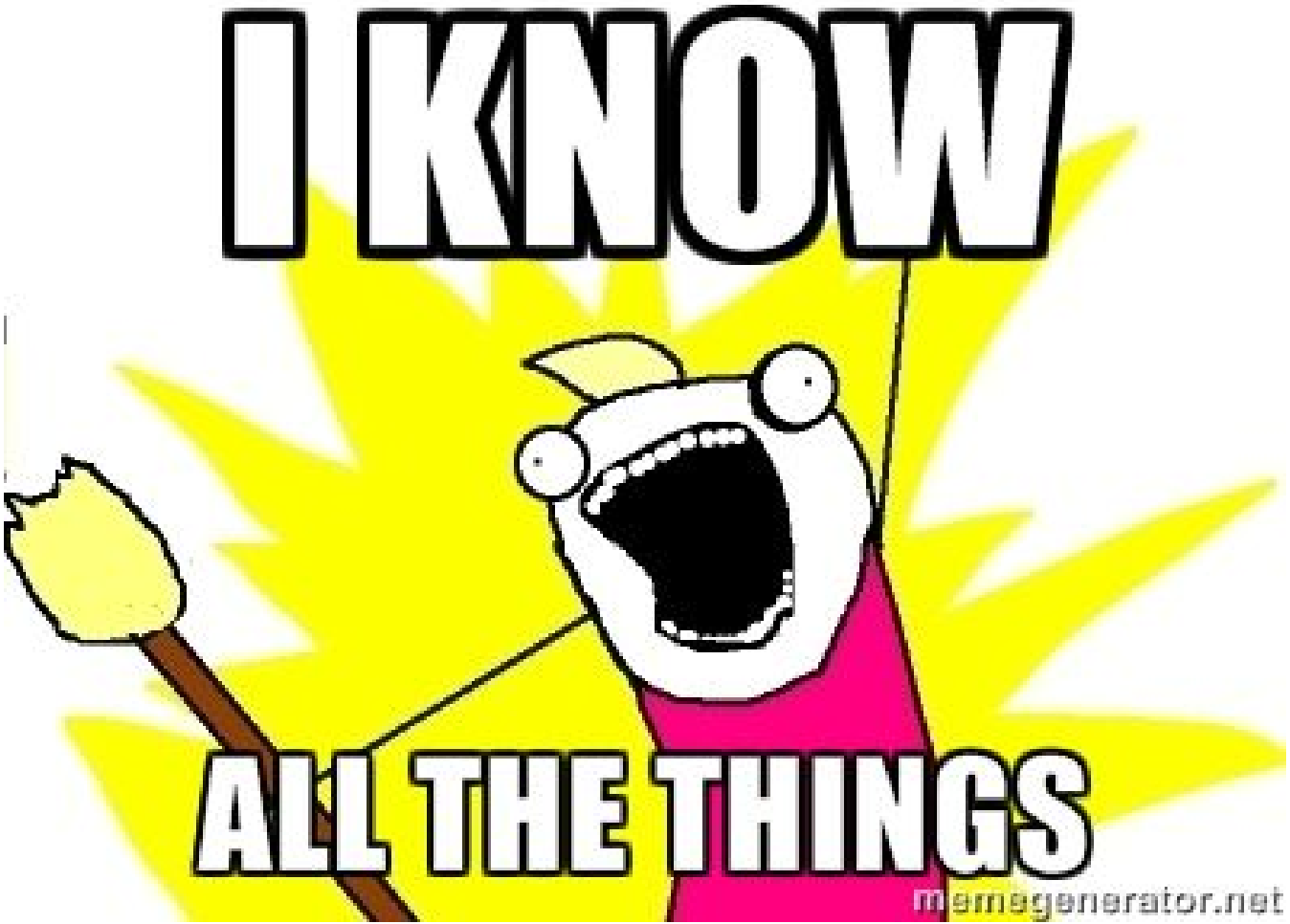
setBrightness() was intended to be called once, in setup(), to limit the current/brightness (battery life) of the LEDs throughout the life of the sketch. The operation of this function is “lossy” you will lose the color range.

brightness: The level of brightness 0-255. Lower number is darker

```
strip.setBrightness(128); // Half brightness
```

What you have learned so far

- IF, For, and Functions statements.
- How variables declaration works
- How to check to see if a button has been pressed
- How to check to see the value of a knob (Potentiometer)
- How to install 3rd party libraries into the Arduino IDE
- How to change the color and turn on and off LEDs in a Neopixel LED strip
- How to make patterns with the Neopixel LED strip
- How to switch between patterns with a button.


I KNOW
ALL THE THINGS
memegenerator.net

But wait… There's more.

Try this

- Try FullExample.ino example, It shows many of the things you have learned so far.
- FastLED Animation Library - http://fastled.io/