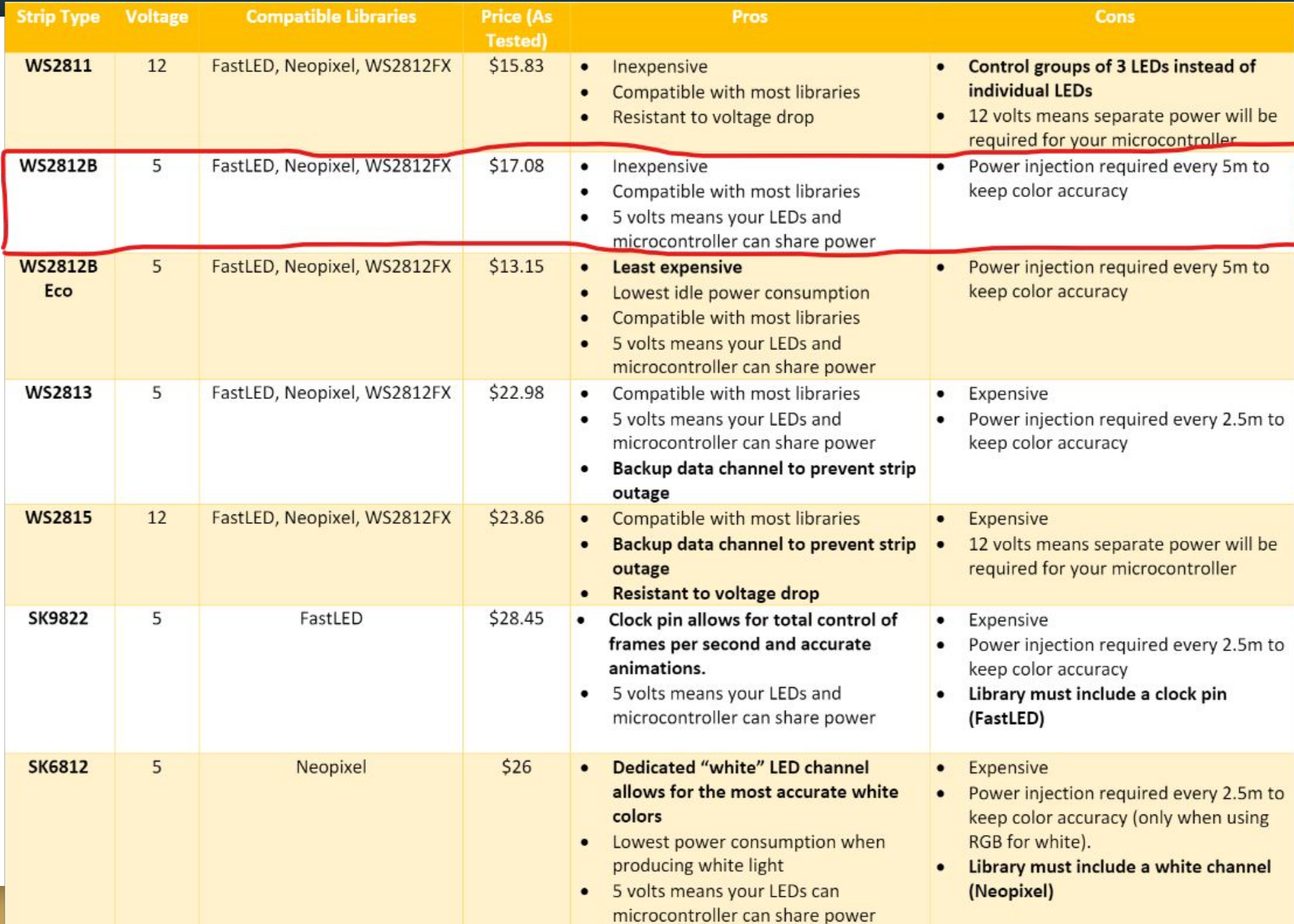
Programming an LED strip

By Alex Torres

We are going to understand the way WS2812b leds work and program our own animations

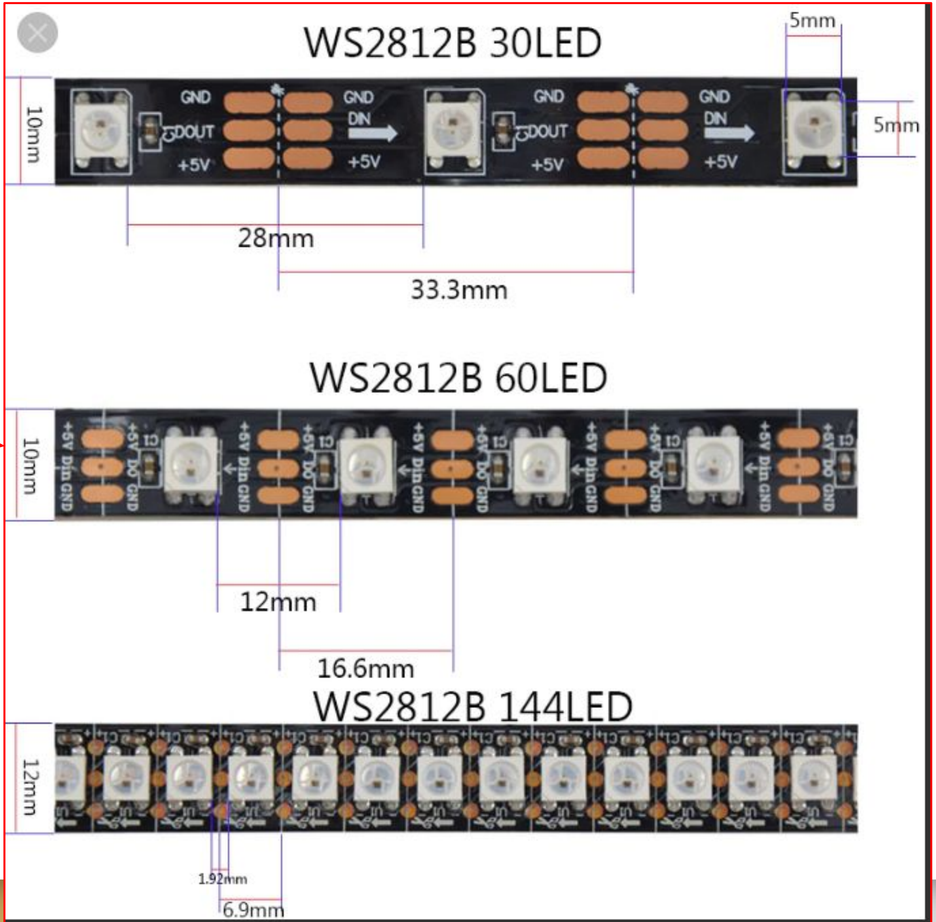
WS2812B Vs. Others



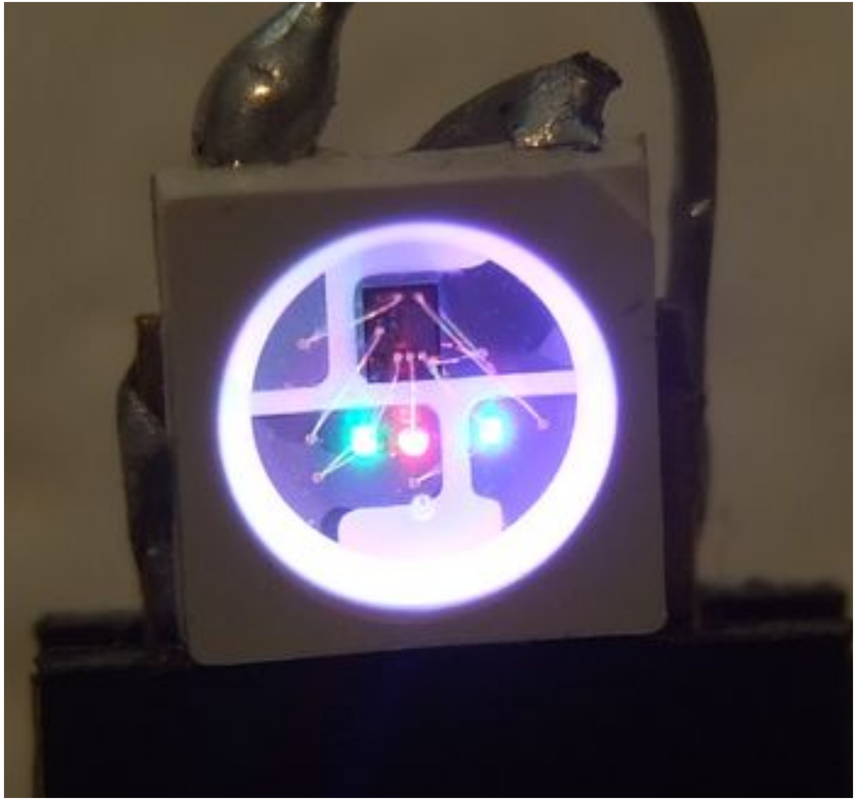
Strip Type	Voltage	Compatible Libraries	Price (As Tested)	Pros	Cons
WS2811	12	FastLED, Neopixel, WS2812FX	$15.83	• Inexpensive • Compatible with most libraries • Resistant to voltage drop	• **Control groups of 3 LEDs instead of individual LEDs** • 12 volts means separate power will be required for your microcontroller
WS2812B	5	FastLED, Neopixel, WS2812FX	$17.08	• Inexpensive • Compatible with most libraries • 5 volts means your LEDs and microcontroller can share power	• Power injection required every 5m to keep color accuracy
WS2812B Eco	5	FastLED, Neopixel, WS2812FX	$13.15	• **Least expensive** • Lowest idle power consumption • Compatible with most libraries • 5 volts means your LEDs and microcontroller can share power	• Power injection required every 5m to keep color accuracy
WS2813	5	FastLED, Neopixel, WS2812FX	$22.98	• Compatible with most libraries • 5 volts means your LEDs and microcontroller can share power • **Backup data channel to prevent strip outage**	• Expensive • Power injection required every 2.5m to keep color accuracy
WS2815	12	FastLED, Neopixel, WS2812FX	$23.86	• Compatible with most libraries • **Backup data channel to prevent strip outage** • **Resistant to voltage drop**	• Expensive • 12 volts means separate power will be required for your microcontroller
SK9822	5	FastLED	$28.45	• **Clock pin allows for total control of frames per second and accurate animations.** • 5 volts means your LEDs and microcontroller can share power	• Expensive • Power injection required every 2.5m to keep color accuracy • **Library must include a clock pin (FastLED)**
SK6812	5	Neopixel	$26	• **Dedicated "white" LED channel allows for the most accurate white colors** • Lowest power consumption when producing white light • 5 volts means your LEDs can microcontroller can share power	• Expensive • Power injection required every 2.5m to keep color accuracy (only when using RGB for white). • **Library must include a white channel (Neopixel)**

Come in different number of LED’s on a strip

We will be using the 60led variant today.



Inside each LED

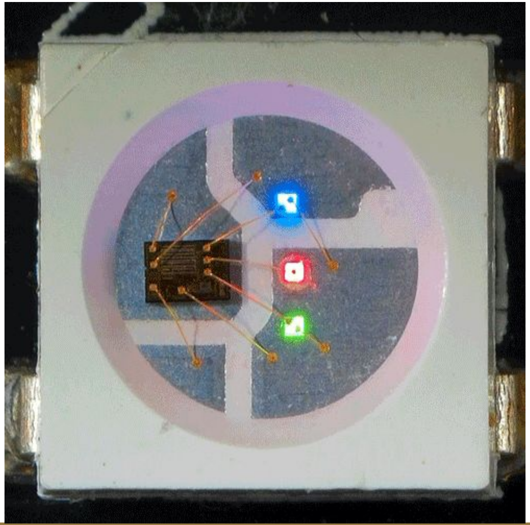


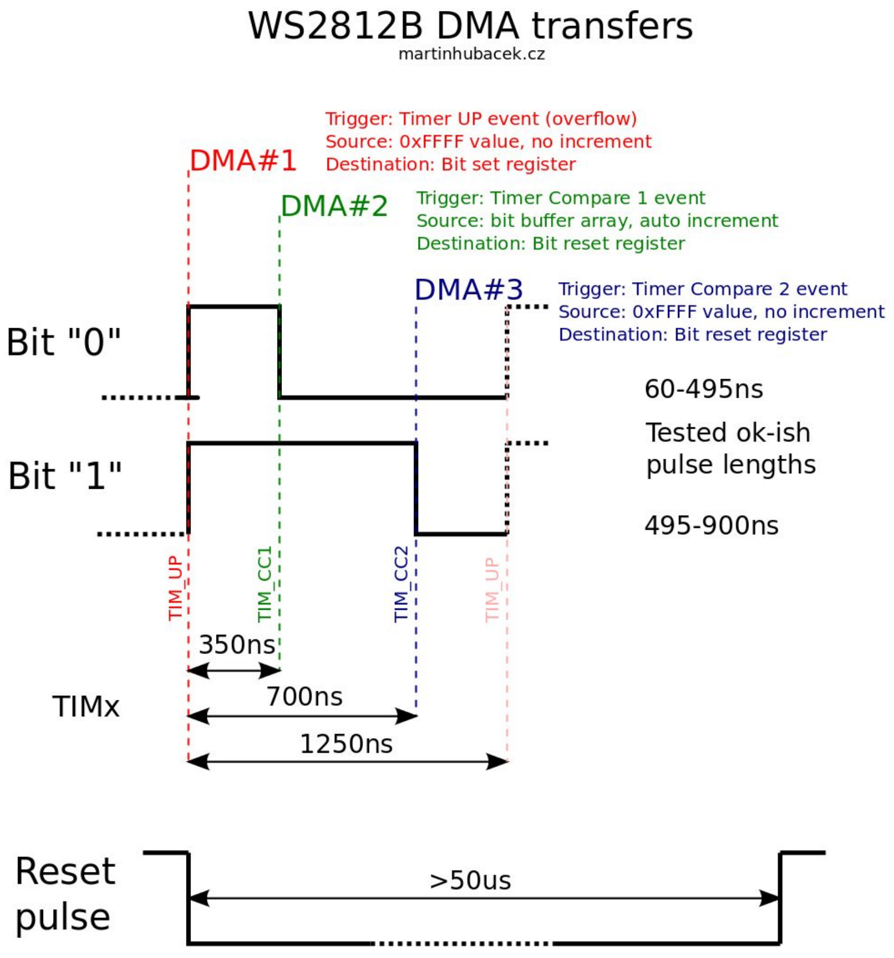
Uses different pieces of semiconductor to produce the green red and blue lights.

The black box is a chip that will receive data to to distinguish what leds to turn on and at what brightness.

The way it works

The small chip receives serial data in 24bit and shift the new data on to the next led in an asynchronous fashion and depending on the time requirements it must meet it will then understand what leds to turn on.





WS2812B DMA transfers

martinhubacek.cz

DMA#1
Trigger: Timer UP event (overflow)
Source: 0xFFFF value, no increment
Destination: Bit set register

DMA#2
Trigger: Timer Compare 1 event
Source: bit buffer array, auto increment
Destination: Bit reset register

DMA#3
Trigger: Timer Compare 2 event
Source: 0xFFFF value, no increment
Destination: Bit reset register

60-495ns

Tested ok-ish pulse lengths

495-900ns

So how do we send that precise information to the leds?

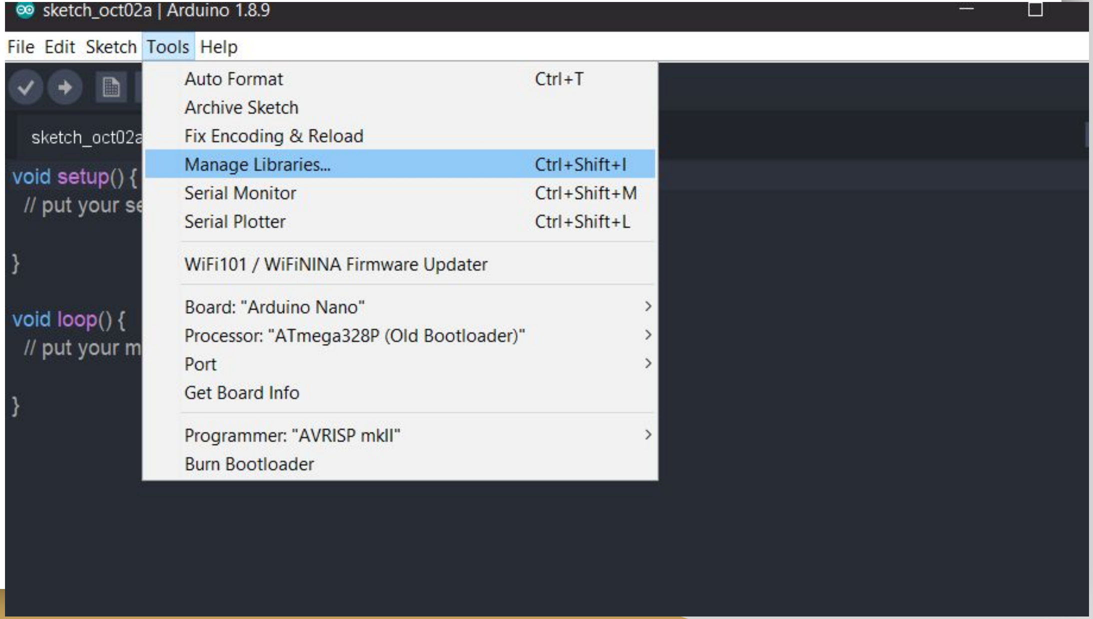
Any Microcontroller will work such as the arduino Uno we will be using today.

With the help of the neopixel library we can easily make animations in C.

Setting up ide

1.Open up arduino

2. Select “Manage Libraries” from the tools drop down menu.

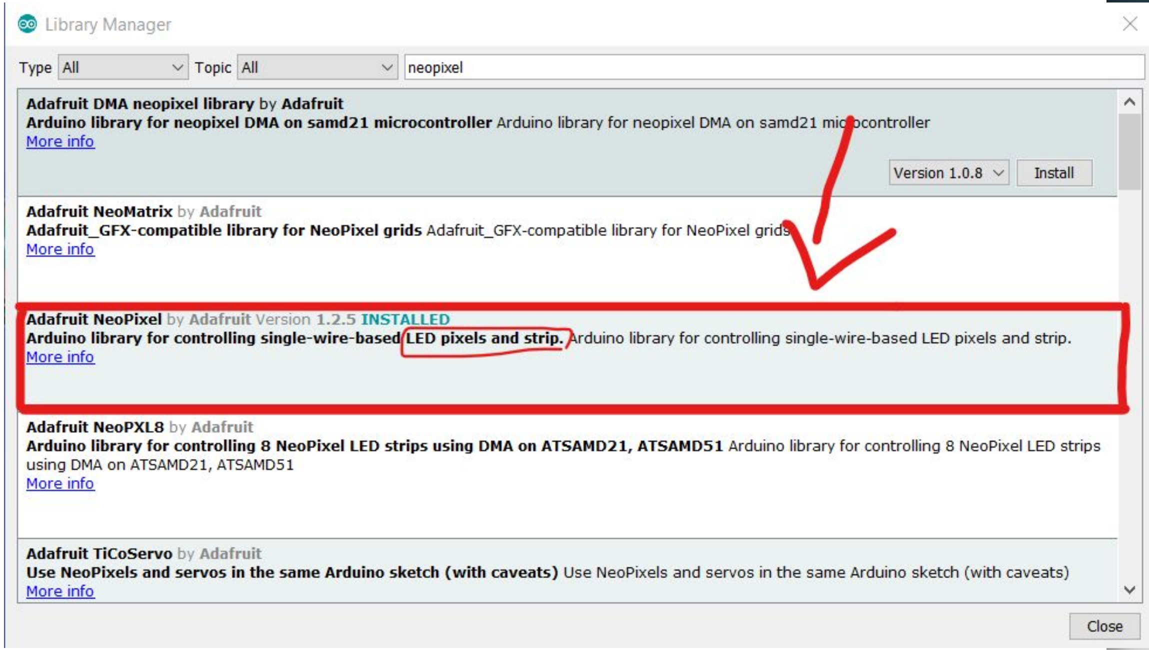

sketch_oct02a | Arduino 1.8.9
File Edit Sketch Tools Help
sketch_oct02a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Nano"
Processor: "ATmega328P (Old Bootloader)"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

```
void setup() {
  // put your se

}

void loop() {
  // put your m

}
```

Setting up ide

3. Type neopixel in the search box.

4. Install the library that mentions LED strip



Library Manager

Type All Topic All neopixel

Adafruit DMA neopixel library by **Adafruit**
Arduino library for neopixel DMA on samd21 microcontroller Arduino library for neopixel DMA on samd21 microcontroller
More info

Adafruit NeoMatrix by **Adafruit**
Adafruit_GFX-compatible library for NeoPixel grids Adafruit_GFX-compatible library for NeoPixel grids
More info

Adafruit NeoPixel by **Adafruit** Version **1.2.5** **INSTALLED**
Arduino library for controlling single-wire-based LED pixels and strip. Arduino library for controlling single-wire-based LED pixels and strip.
More info

Adafruit NeoPXL8 by **Adafruit**
Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51 Arduino library for controlling 8 NeoPixel LED strips using DMA on ATSAMD21, ATSAMD51
More info

Adafruit TiCoServo by **Adafruit**
Use NeoPixels and servos in the same Arduino sketch (with caveats) Use NeoPixels and servos in the same Arduino sketch (with caveats)
More info

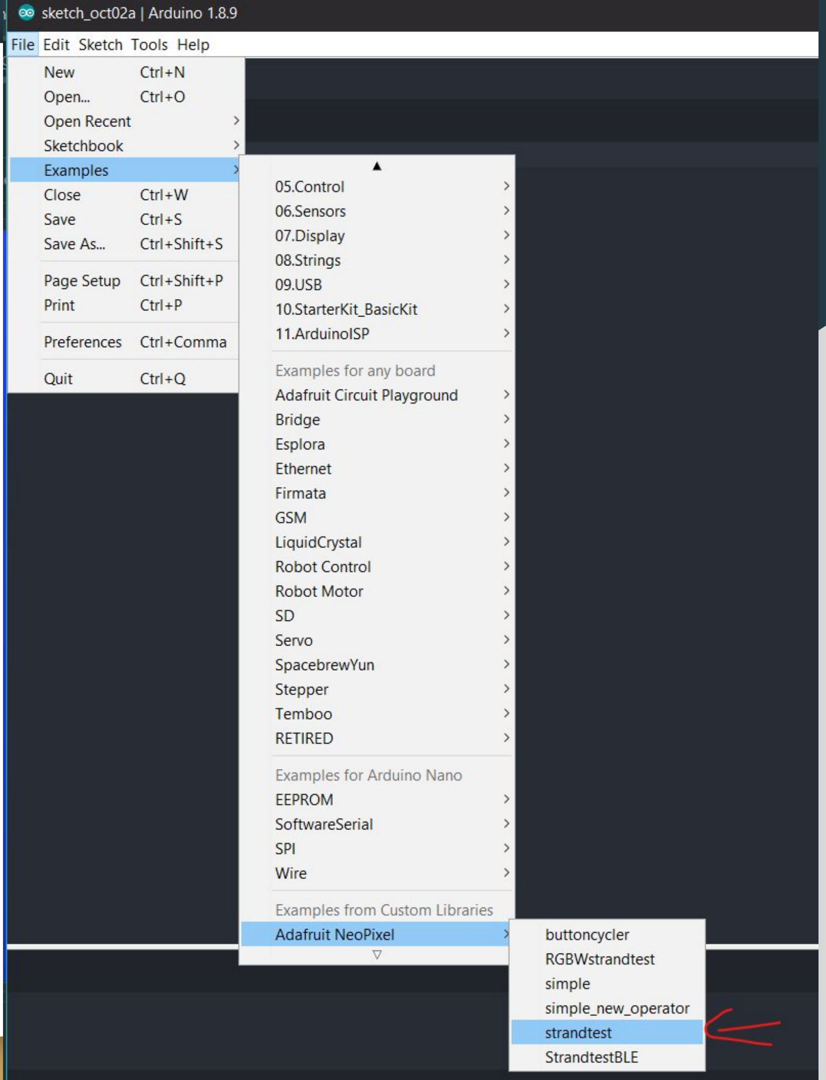
Setting up ide

5. Once installed navigate to “FIle”, “Examples”,

“Adafruit NeoPixel”,

then click on “strandtest”.

A new window should appear.



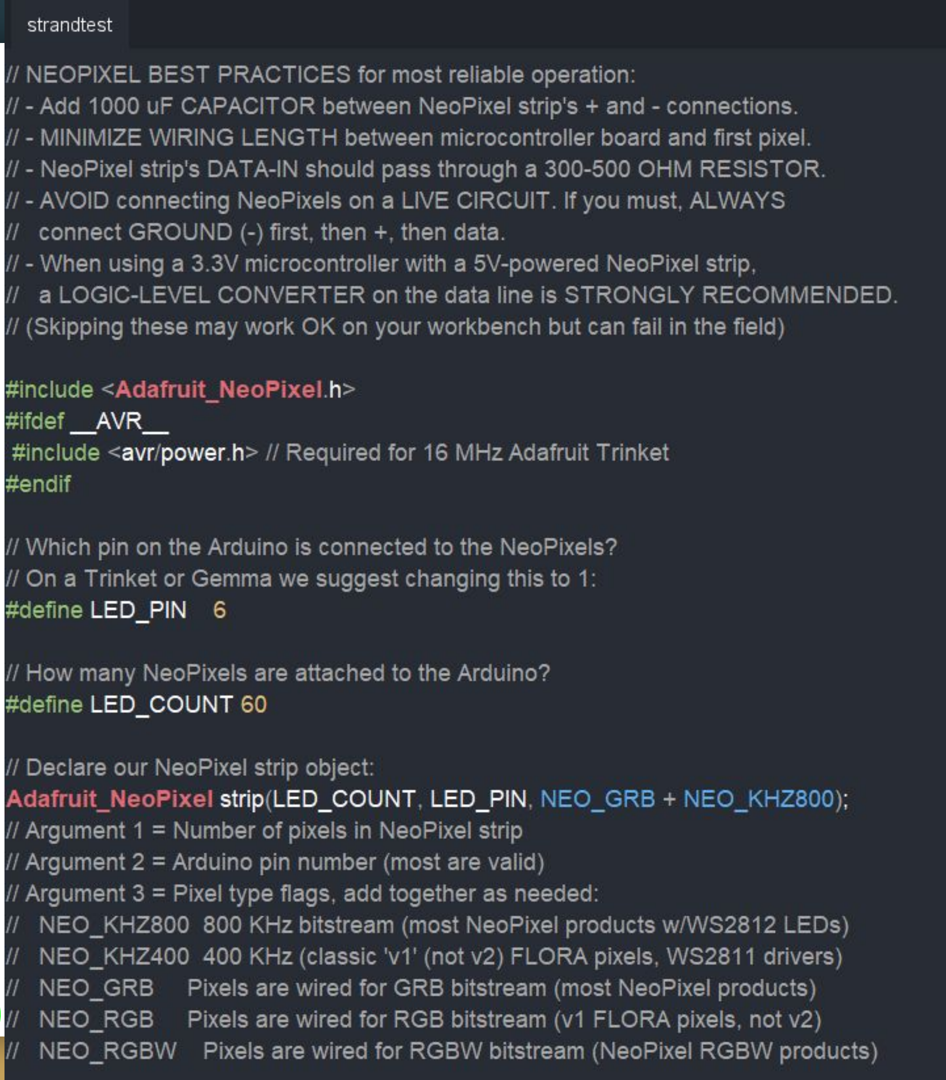
What do we see?

Most of the words are just comments written to understand what to adjust.

The adafruit library is imported with the first line of code.

Most things are best left alone for this demonstration.

The library is used to support other strips as well giving us a wide variety of options.

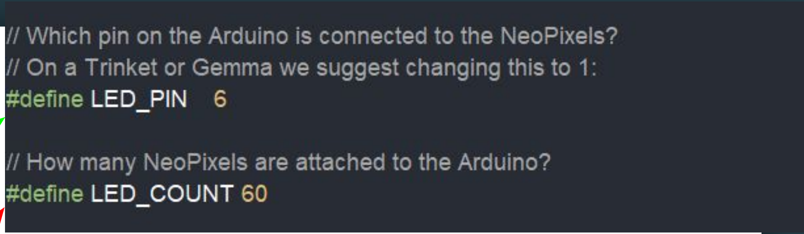


```
// NEOPIXEL BEST PRACTICES for most reliable operation:
// - Add 1000 uF CAPACITOR between NeoPixel strip's + and - connections.
// - MINIMIZE WIRING LENGTH between microcontroller board and first pixel.
// - NeoPixel strip's DATA-IN should pass through a 300-500 OHM RESISTOR.
// - AVOID connecting NeoPixels on a LIVE CIRCUIT. If you must, ALWAYS
//   connect GROUND (-) first, then +, then data.
// - When using a 3.3V microcontroller with a 5V-powered NeoPixel strip,
//   a LOGIC-LEVEL CONVERTER on the data line is STRONGLY RECOMMENDED.
// (Skipping these may work OK on your workbench but can fail in the field)

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
#define LED_PIN    6

// How many NeoPixels are attached to the Arduino?
#define LED_COUNT 60

// Declare our NeoPixel strip object:
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
// Argument 1 = Number of pixels in NeoPixel strip
// Argument 2 = Arduino pin number (most are valid)
// Argument 3 = Pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
//   NEO_RGBW    Pixels are wired for RGBW bitstream (NeoPixel RGBW products)
```

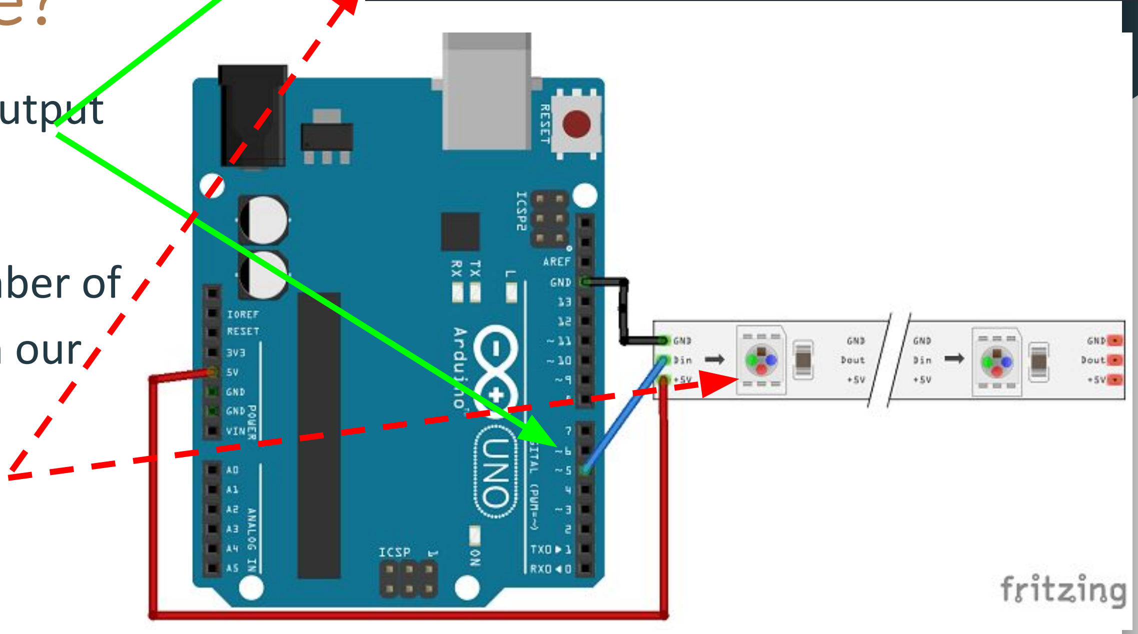
What do we see?

LED_PIN refers to the digital output pin on the arduino.

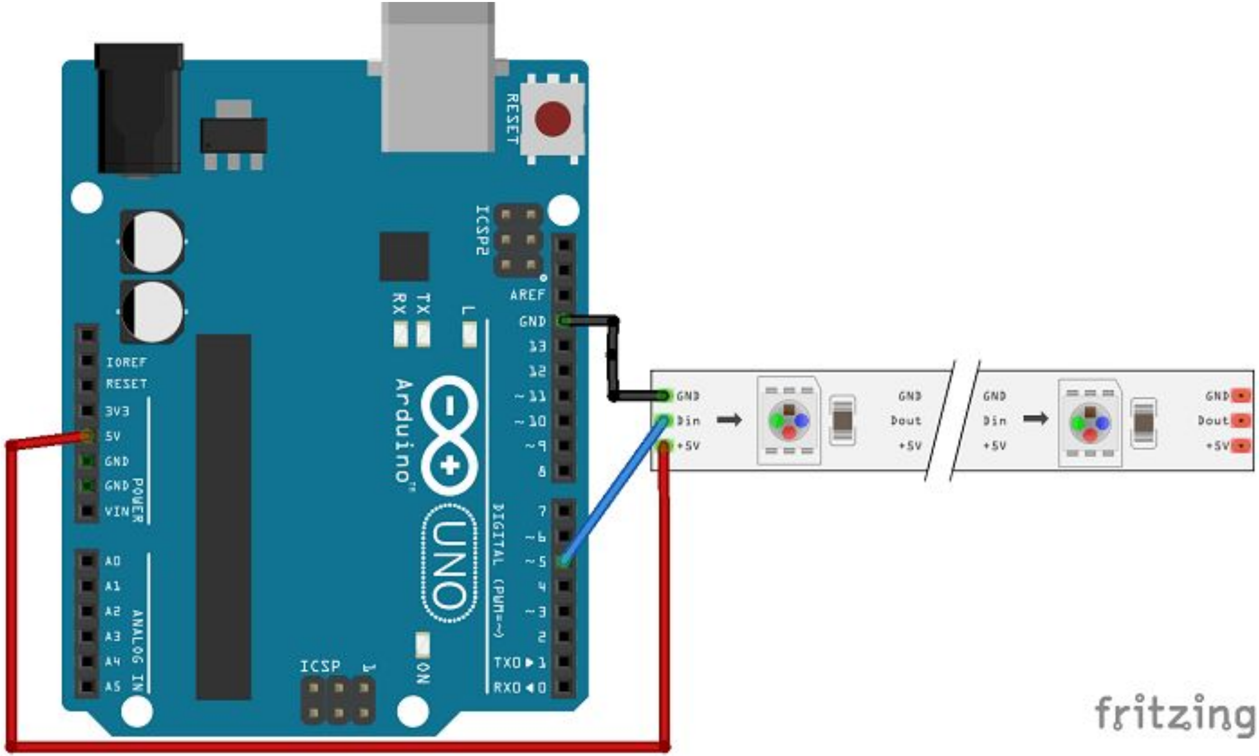
LED_COUNT refers to the number of leds on the strip which is 10 in our case.



```
// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
#define LED_PIN    6

// How many NeoPixels are attached to the Arduino?
#define LED_COUNT 60
```



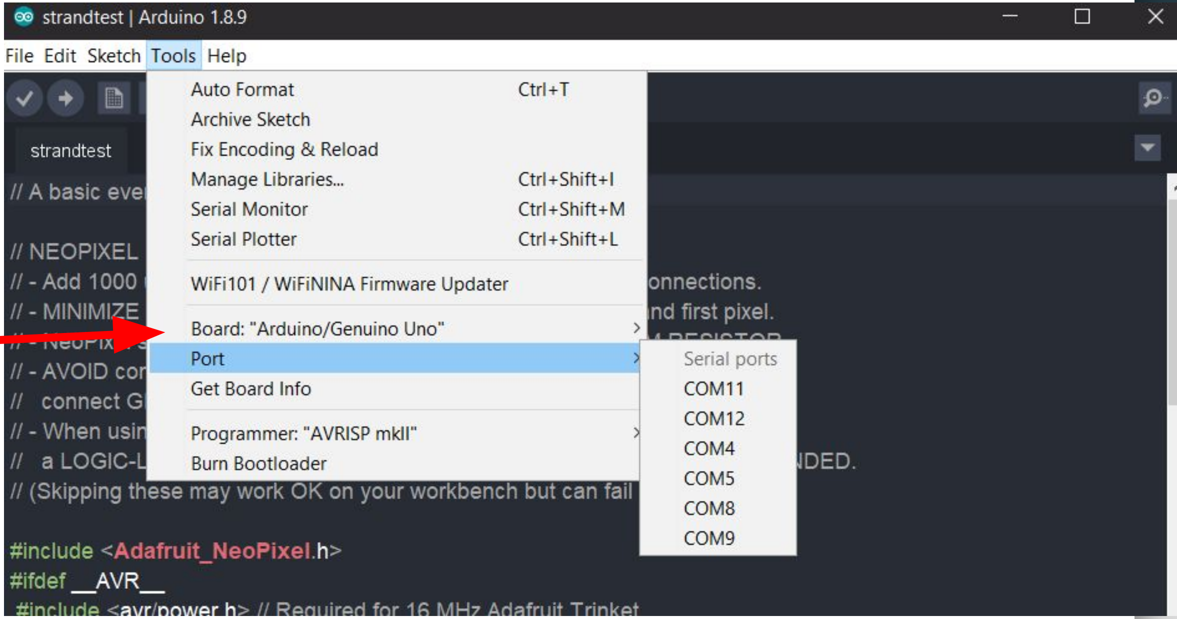
Connect led strip and arduino



Uploading code to arduino

Make sure under tools/Board is set to Arduino/Genuino Uno.

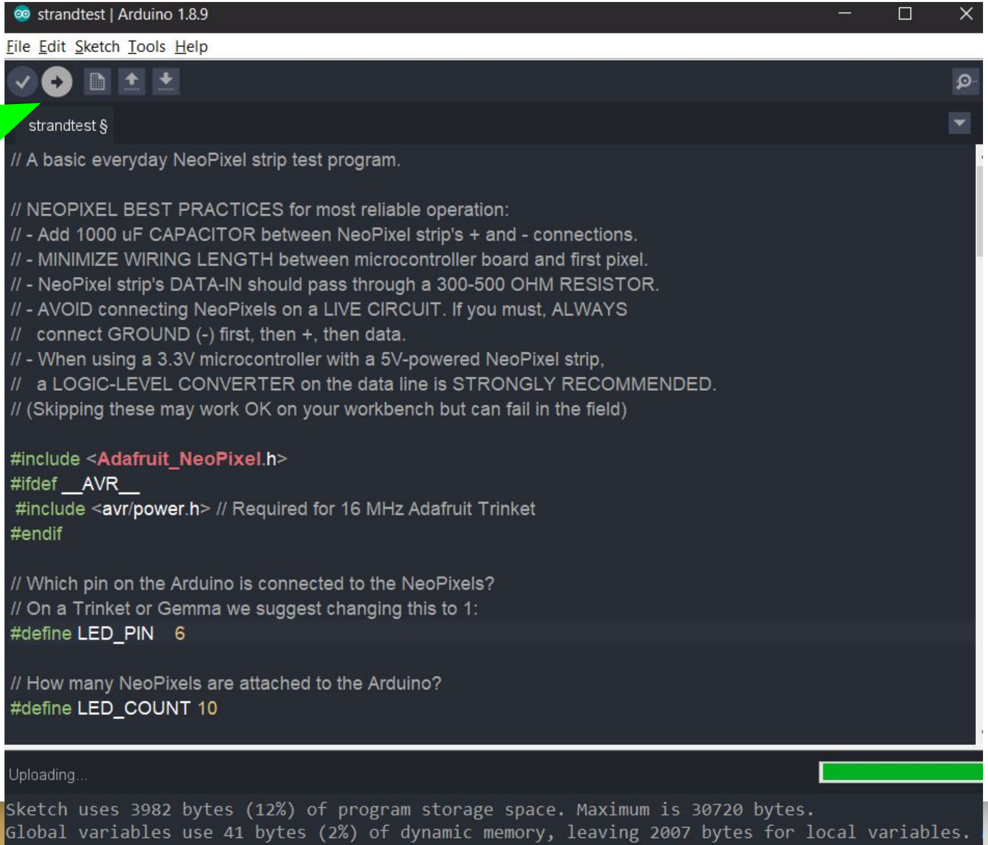
And designated port is also selected.


strandtest | Arduino 1.8.9
File
Edit
Sketch
Tools
Help
strandtest
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM11
COM12
COM4
COM5
COM8
COM9

```
// A basic eve

// NEOPIXEL
// - Add 1000                                          onnections.
// - MINIMIZE                                          nd first pixel.
// - NeoPix
// - AVOID cor
//   connect G
// - When usin
//   a LOGIC-L                                                        NDED.
// (Skipping these may work OK on your workbench but can fail

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
```

Uploading code to arduino

Now click the upload button on the upper left hand corner.



```
// A basic everyday NeoPixel strip test program.

// NEOPIXEL BEST PRACTICES for most reliable operation:
// - Add 1000 uF CAPACITOR between NeoPixel strip's + and - connections.
// - MINIMIZE WIRING LENGTH between microcontroller board and first pixel.
// - NeoPixel strip's DATA-IN should pass through a 300-500 OHM RESISTOR.
// - AVOID connecting NeoPixels on a LIVE CIRCUIT. If you must, ALWAYS
//   connect GROUND (-) first, then +, then data.
// - When using a 3.3V microcontroller with a 5V-powered NeoPixel strip,
//   a LOGIC-LEVEL CONVERTER on the data line is STRONGLY RECOMMENDED.
// (Skipping these may work OK on your workbench but can fail in the field)

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1:
#define LED_PIN    6

// How many NeoPixels are attached to the Arduino?
#define LED_COUNT 10
```

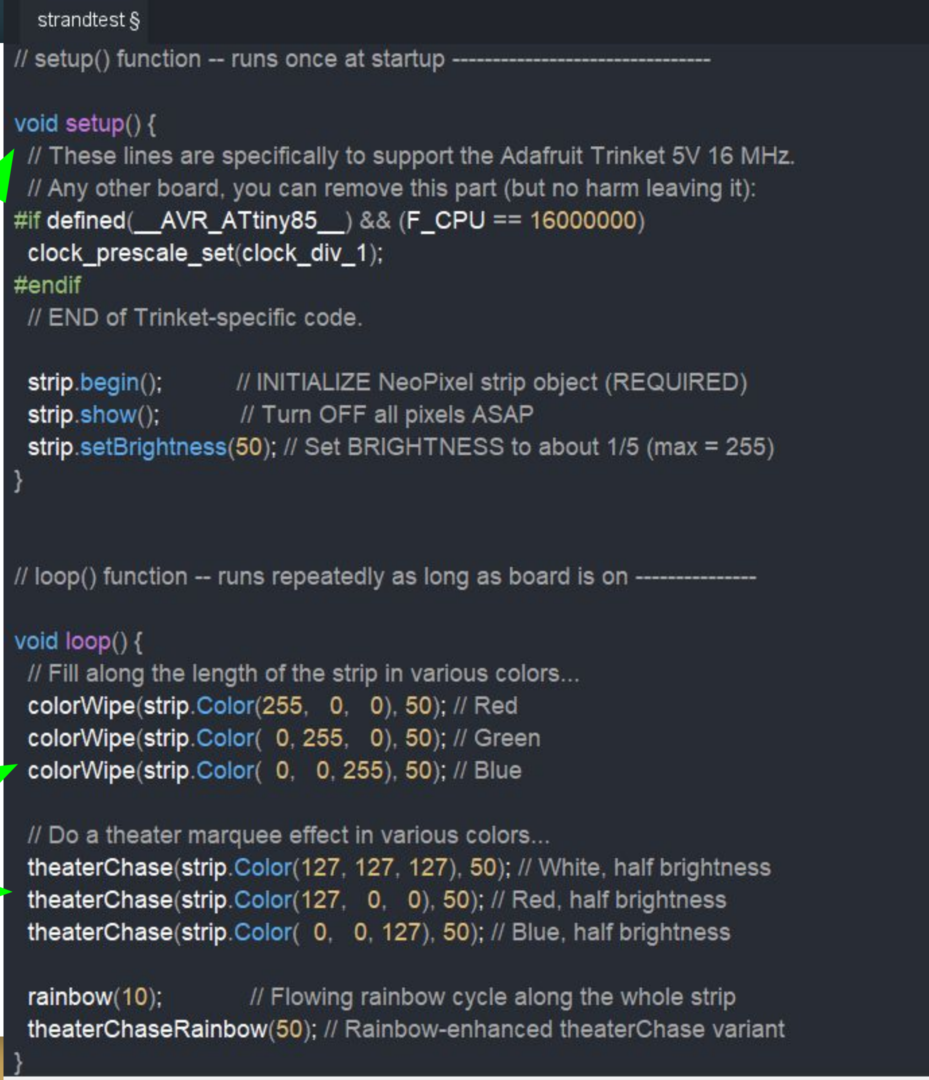
```
Sketch uses 3982 bytes (12%) of program storage space. Maximum is 30720 bytes.
Global variables use 41 bytes (2%) of dynamic memory, leaving 2007 bytes for local variables.
```

OMG!!! It works!!

What makes it do that?

void setup() is ran once upon startup used for initializing the strip object.

void loop() will continue to run repeatedly. In this example different functions are declared such as colorWipe and theaterChase with a given parameter.

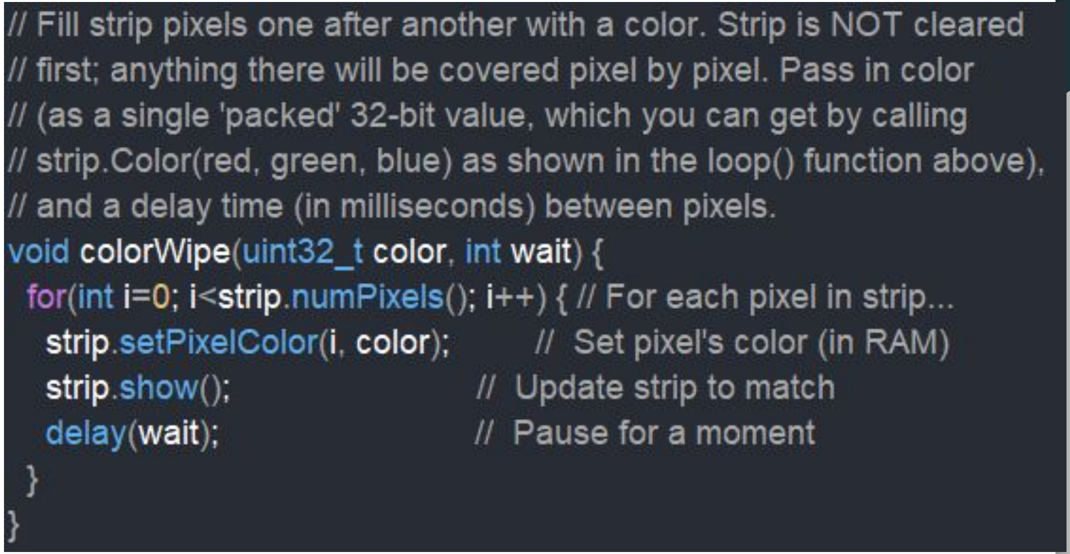


strandtest §

```
// setup() function -- runs once at startup --------------------------------

void setup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)
}


// loop() function -- runs repeatedly as long as board is on ---------------

void loop() {
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(255,   0,   0), 50); // Red
  colorWipe(strip.Color(  0, 255,   0), 50); // Green
  colorWipe(strip.Color(  0,   0, 255), 50); // Blue

  // Do a theater marquee effect in various colors...
  theaterChase(strip.Color(127, 127, 127), 50); // White, half brightness
  theaterChase(strip.Color(127,   0,   0), 50); // Red, half brightness
  theaterChase(strip.Color(  0,   0, 127), 50); // Blue, half brightness

  rainbow(10);             // Flowing rainbow cycle along the whole strip
  theaterChaseRainbow(50); // Rainbow-enhanced theaterChase variant
}
```

Let’s take a look at the function colorWipe()

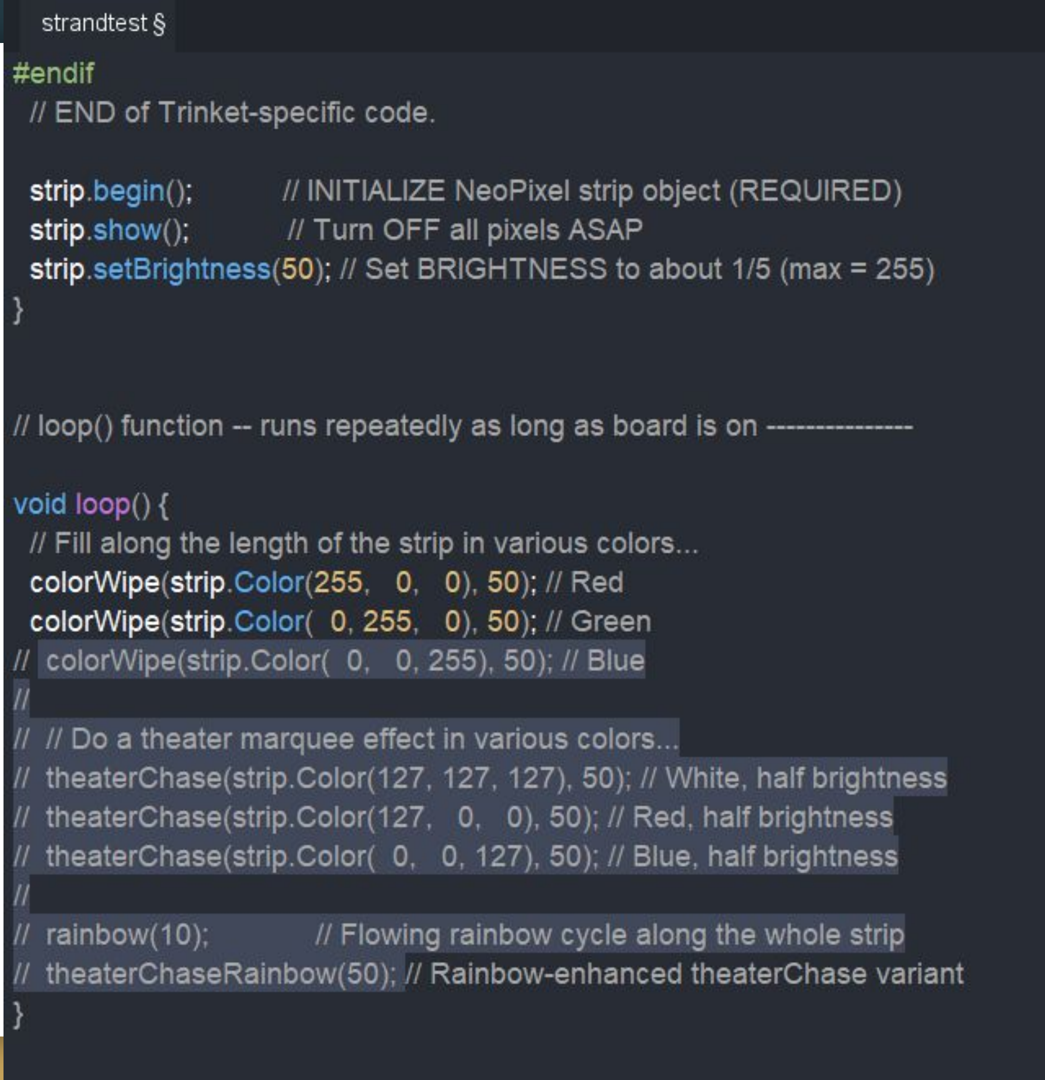
colorWipe() takes in two parameters. A uint32_t color which an example is strip.Color(x,y,z). And the amount to wait in milliseconds.



```
// Fill strip pixels one after another with a color. Strip is NOT cleared
// first; anything there will be covered pixel by pixel. Pass in color
// (as a single 'packed' 32-bit value, which you can get by calling
// strip.Color(red, green, blue) as shown in the loop() function above),
// and a delay time (in milliseconds) between pixels.
void colorWipe(uint32_t color, int wait) {
  for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...
    strip.setPixelColor(i, color);         //  Set pixel's color (in RAM)
    strip.show();                          //  Update strip to match
    delay(wait);                           //  Pause for a moment
  }
}
```

What does it do?

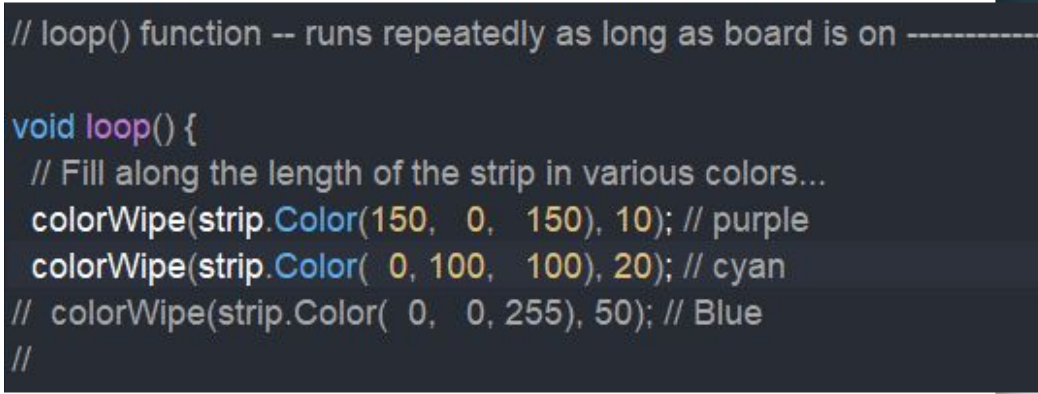
Lets comment out the functions in the loop() by highlighting and using the shortcut “ctrl+/” and leave only two colorWipe(). Then upload the code once more.



```
#endif
  // END of Trinket-specific code.

  strip.begin();           // INITIALIZE NeoPixel strip object (REQUIRED)
  strip.show();            // Turn OFF all pixels ASAP
  strip.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)
}


// loop() function -- runs repeatedly as long as board is on ---------------

void loop() {
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(255,   0,   0), 50); // Red
  colorWipe(strip.Color(  0, 255,   0), 50); // Green
//  colorWipe(strip.Color(  0,   0, 255), 50); // Blue
//
//  // Do a theater marquee effect in various colors...
//  theaterChase(strip.Color(127, 127, 127), 50); // White, half brightness
//  theaterChase(strip.Color(127,   0,   0), 50); // Red, half brightness
//  theaterChase(strip.Color(  0,   0, 127), 50); // Blue, half brightness
//
//  rainbow(10);             // Flowing rainbow cycle along the whole strip
//  theaterChaseRainbow(50); // Rainbow-enhanced theaterChase variant
}
```

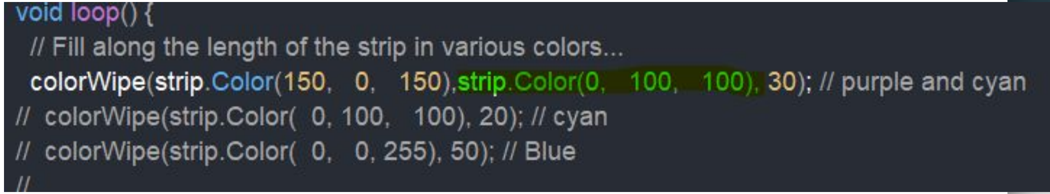
Let's change some things!

As you can see, it individually changes the color of each led with the new color it was given. Now let's mess around with the time and color. And upload.



```
// loop() function -- runs repeatedly as long as board is on ----------

void loop() {
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(150,   0,   150), 10); // purple
  colorWipe(strip.Color(  0, 100,   100), 20); // cyan
//  colorWipe(strip.Color(  0,   0, 255), 50); // Blue
//
```

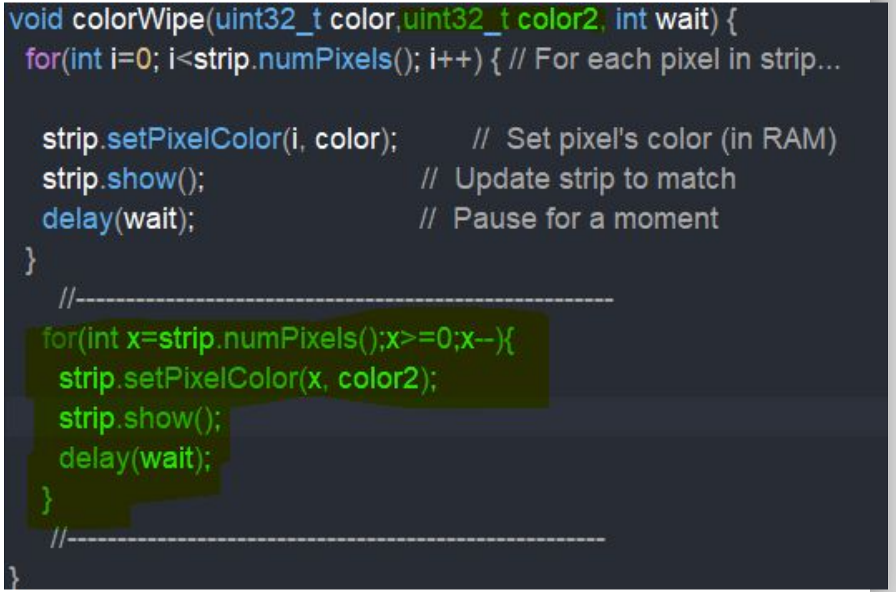
What if we modify the function?



```
void loop() {
  // Fill along the length of the strip in various colors...
  colorWipe(strip.Color(150,   0,   150),strip.Color(0,   100,   100), 30); // purple and cyan
//  colorWipe(strip.Color(  0, 100,   100), 20); // cyan
//  colorWipe(strip.Color(  0,   0, 255), 50); // Blue
//
```

In our loop we must add another strip.Color(x,y,z).

In the function we add another parameter uint32_t and name it color2. After our first for loop we add another with modified instructions

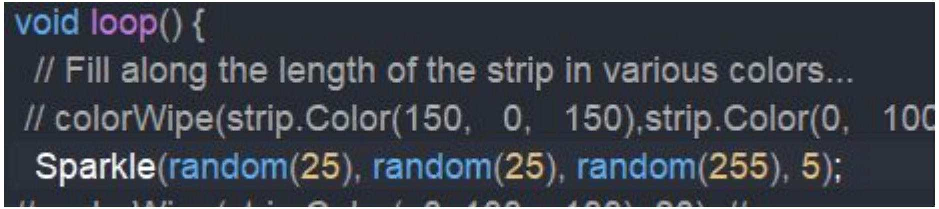


```
void colorWipe(uint32_t color,uint32_t color2, int wait) {
  for(int i=0; i<strip.numPixels(); i++) { // For each pixel in strip...

    strip.setPixelColor(i, color);         //  Set pixel's color (in RAM)
    strip.show();                          //  Update strip to match
    delay(wait);                           //  Pause for a moment
  }
   //--------------------------------------------------
   for(int x=strip.numPixels();x>=0;x--){
    strip.setPixelColor(x, color2);
    strip.show();
    delay(wait);
   }
   //--------------------------------------------------
}
```

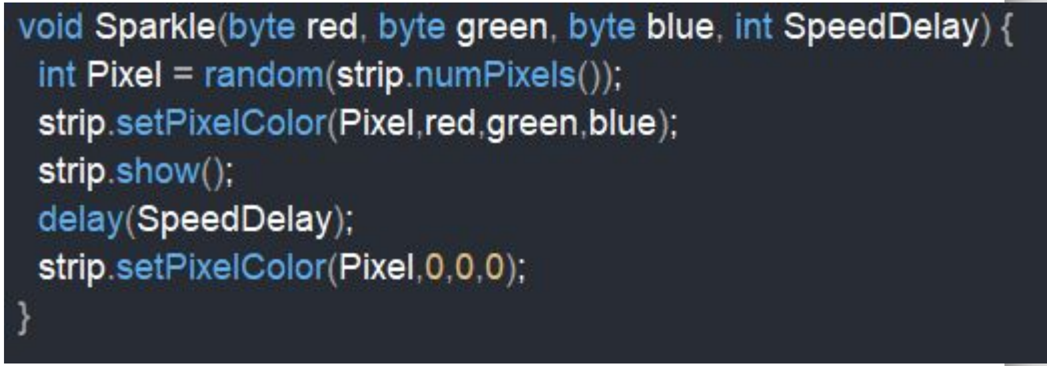
Lets try another

The function will be called sparkle().

Once done upload.

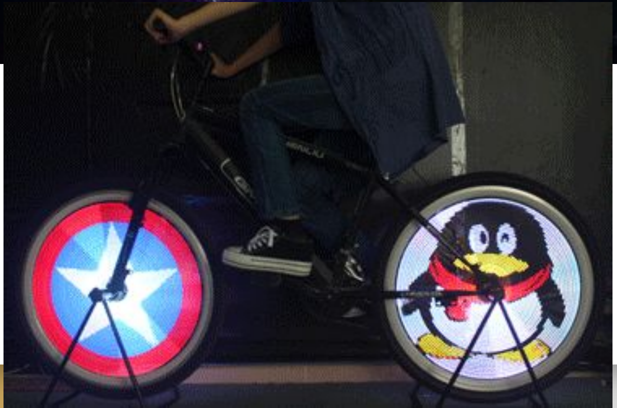


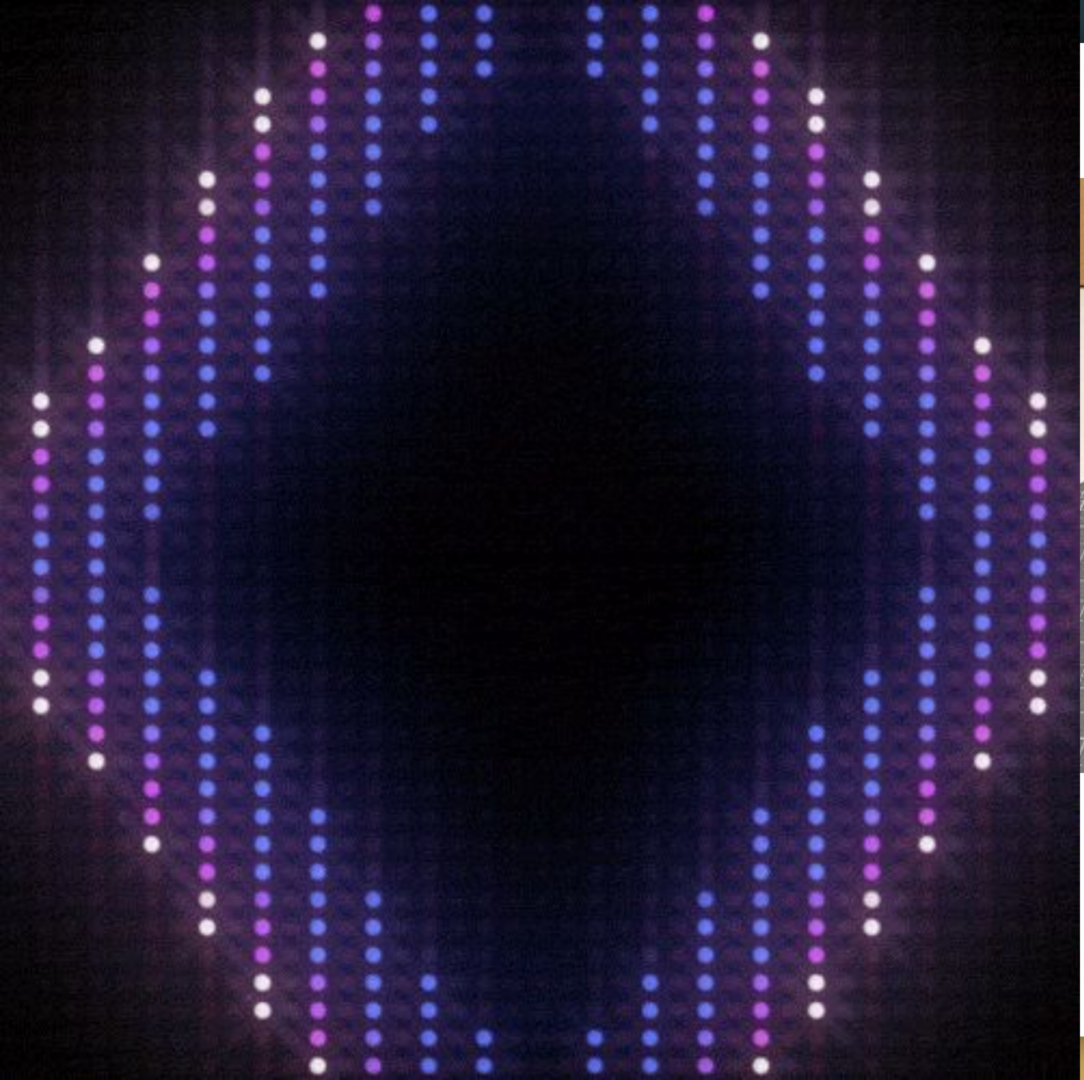
```
void loop() {
  // Fill along the length of the strip in various colors...
 // colorWipe(strip.Color(150,  0,  150),strip.Color(0,  100
  Sparkle(random(25), random(25), random(255), 5);
```

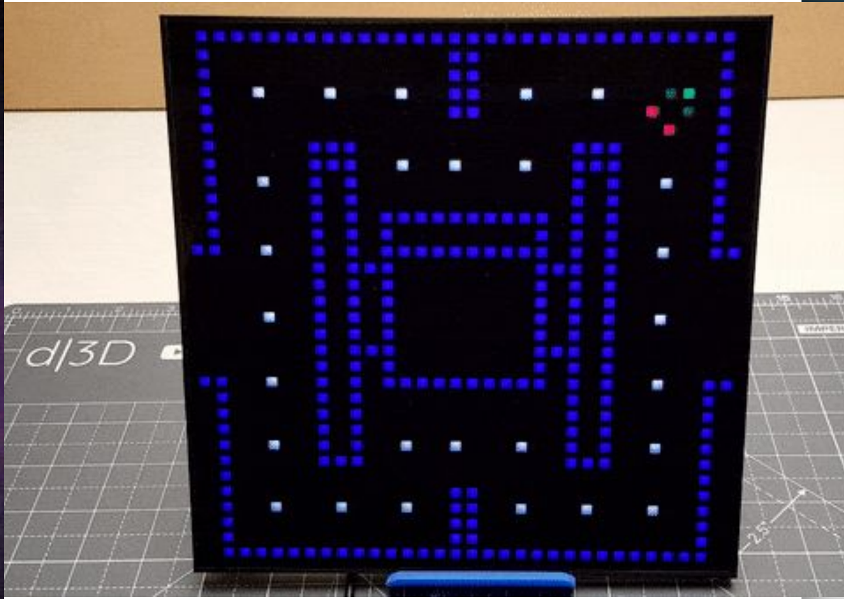


```
void Sparkle(byte red, byte green, byte blue, int SpeedDelay) {
  int Pixel = random(strip.numPixels());
  strip.setPixelColor(Pixel,red,green,blue);
  strip.show();
  delay(SpeedDelay);
  strip.setPixelColor(Pixel,0,0,0);
}
```

So.....what else can you make with this??

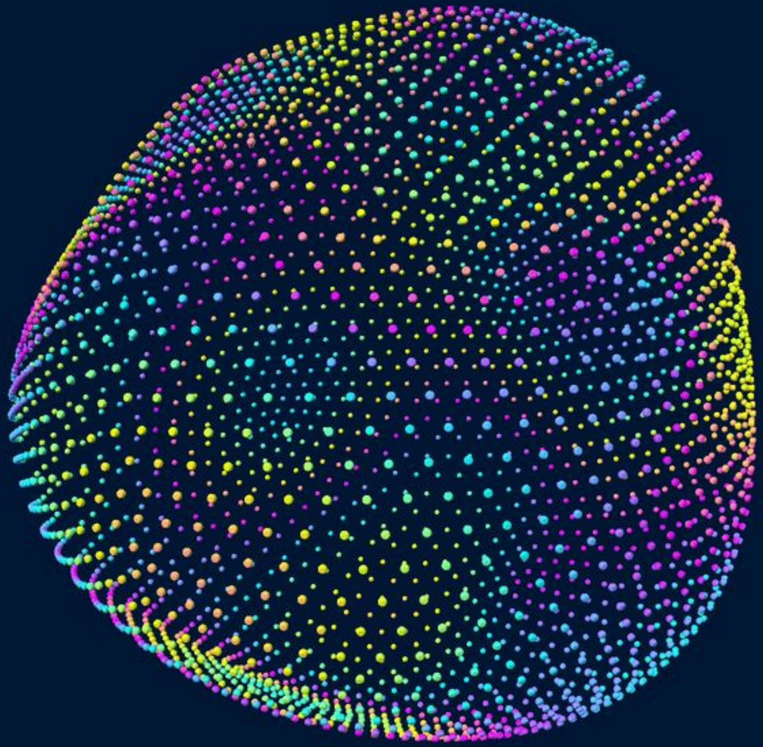





d|3D

DIVERGE

DEEPLIGHT LABS LIGHTSHOW



Q/A