PAT 451/551 INTERACTIVE MEDIA DESIGN I

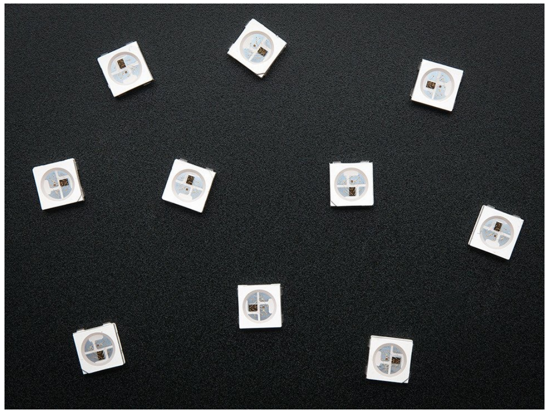
DOTSTAR LEDS

ADDRESSABLE RGB LEDS

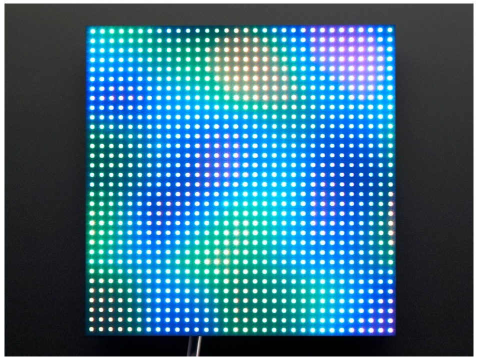
Concept

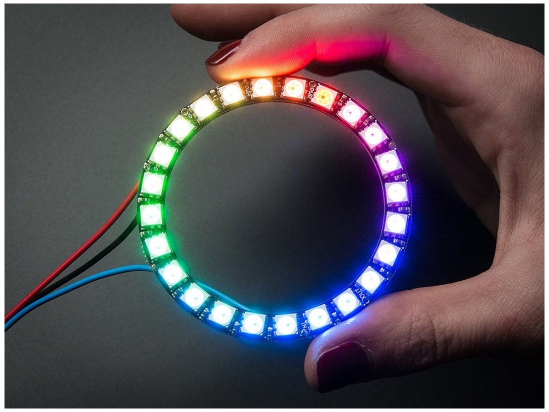
Small (5mm x 5mm) RGB LEDs that include a driver and communication interface.

LEDs are frequently sold in prefabricated form factors, such as linear flexible strips, rings, matrices, or rigid strips.


Individual LED elements

Flexible strip

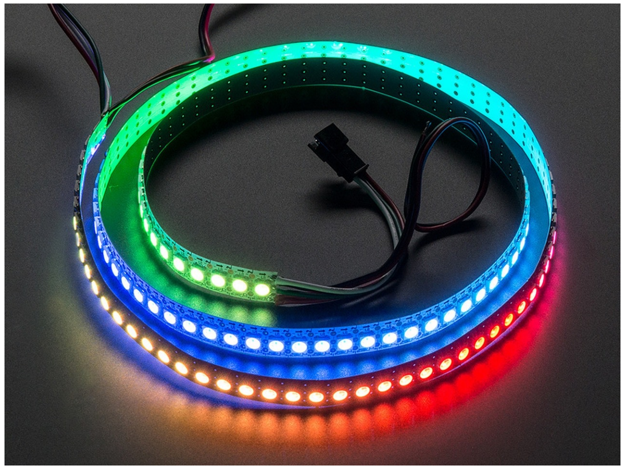

Matrix


Ring

ADDRESSABLE RGB LED STRIPS

Different spacings (densities): usually 30, 60, or 144 per meter

Laminated (outdoor) or bare



LED MODELS

Often casually called “5050” LEDs, indicating the dimensions 5mm x 5mm

In reality, several different LED parts are available.

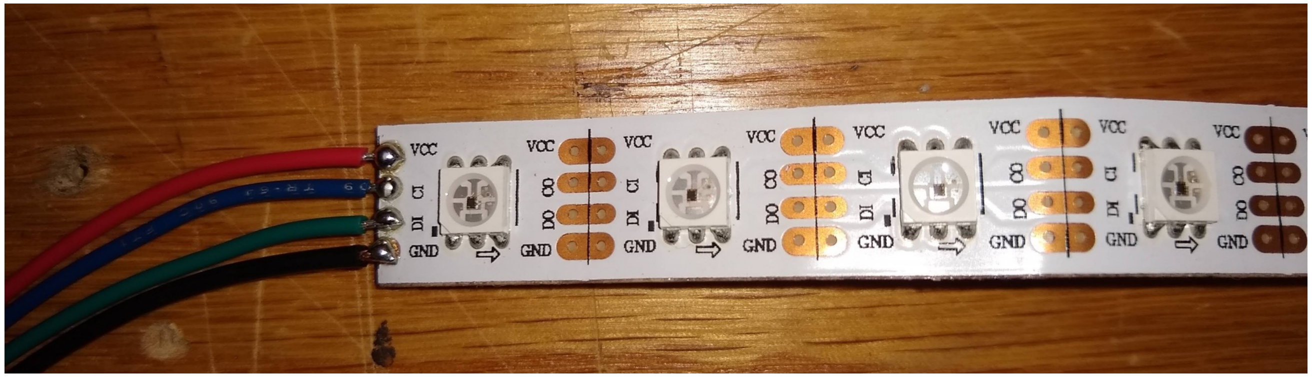
APA102 / SK9822 (DotStar)	WS2812 (NEOPIXEL)
+ Extremely fast data rates, suitable for persistence-of-vision displays. + Easier to interface to a broader range of devices; no strict signal timing requirements. + Don't have to worry about special pins, interrupt management (e.g. Arduino Servo library or tone() function.	+ More affordable. + Wide range of form-factors (pixels, rings, matrices, etc.) + Works from a single microcontroller pin. + RGBW (RGB+white) variants available.
– Slightly more expensive. – Fewer available form factors. – Needs two pins for control.	– Strict 800 KHz data rate; bottleneck on very long strands. – 400 Hz refresh/PWM rate not suitable for persistence-of-vision effects. – Not compatible with programs that require interrupts (e.g. Arduino Servo library or tone() function).

https://learn.adafruit.com/adafruit-dotstar-leds/overview

DOTSTAR LED STRIP

Each chip has 6 pins:

VCC:	+5V
CI:	Clock In
DI:	Data In
CO:	Clock Out
DO:	Data Out
GND:	Ground



We make 4 connections at one end of the strip:
- **VCC connects to +5V (or Vin with a 5VDC power supply)**
- **GND connects to Ground**
- **CI connects to Arduino digital pin 13 (hardware SPI – CKA)**
- **DI connects to Arduino digital pin 11 (hardware SPI – COPIA)**

After that, all the 5V and GND are connected together.

CO and DO of first chip are connected to CI and DI of second chip (and so on).

ARDUINO LIBRARY

Install from:

Sketch > Include Library > Manage Libraries

Search for “Adafruit DotStar” (current v. 1.2.3).

Click “Install”, then close.

DOTSTAR FUNCTIONS

```
// constructor
Adafruit_DotStar strip(NUMPIXELS, DATAPIN, CLOCKPIN);

// Initialize pins
strip.begin();

// set color of pixel p, but don’t update display
strip.setPixelColor(p, color);

// Update display, reflecting ALL prior calls to setPixelColor()
strip.show();
```

`color` is an unsigned 32-bit int, where

- Bits 0-7 are Blue color intensity (8 bits)
- Bits 8-15 are Green color intensity (8 bits)
- Bits 16-23 are Red color intensity (8 bits)

HEXADECIMAL NUMBERS

Hexadecimal numbers are base 16.

An *n*-digit hexadecimal number has a range of

0 to $16^n - 1$

So a **2-digit** hexadecimal number has a range of

0 to $16^2 - 1$ or

0 to 255

In C, we use the prefix `0x` to indicate a hexadecimal number.

The biggest 2 digit hexadecimal number is

`0xFF`

$15*16^1 \;+\; 15*16^0 = 255$

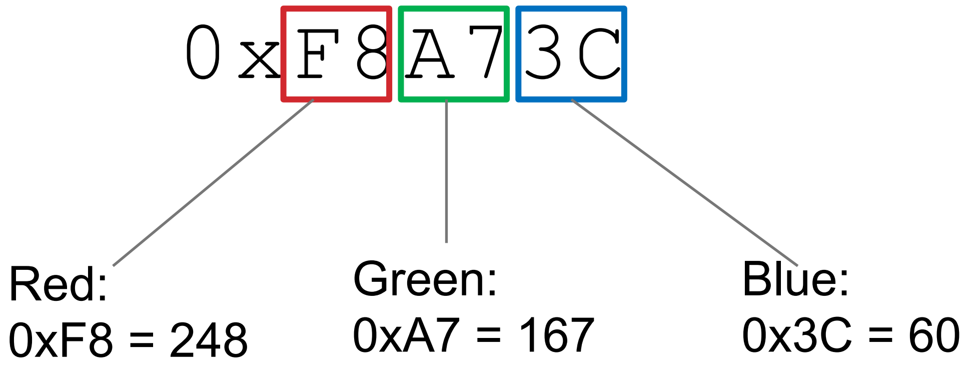
Digits are 0-9 and A-F:
A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

HEXADECIMAL NUMBERS

A **2-digit** hexadecimal number has a range of
0 to 255

We know that 255 is also the biggest 8-bit number, so a 2-digit hexadecimal number is a convenient way to represent an 8-bit value.

We can then represent 3 different 8-bit numbers in a larger variable as a series of bit-shifted 8-bit numbers. This is a common way to represent color values, where we have 8 bits for each channel (Red, Green, and Blue). For example:



RGB LED COLORS

For each color channel, the value 0-255 sets the intensity or brightness, where
00 = Off
FF = Max brightness

So, for basic colors:
0xFF0000 is red, full intensity
0x7F0000 is red, ½ intensity
0x3F0000 is red, ¼ intensity

0x00FF00 is green, full intensity
0x0000FF is blue, full intensity

0xFFFF00 is yellow
0xAAAA00 is yellow, but not as bright

0x00FFFF is cyan
0xFF00FF is magenta

0xFFFFFF is white, maximum intensity

POWER CONSIDERATIONS

- Each color of a single DotStar LED draws about 20mA of current at maximum intensity.
- Setting the color to 0xFFFFFF (white, maximum brightness) draws about 60mA.
- You have 7 leds on your strip, so turning all of them on at 0xFFFFFF will draw about 420mA.
- This is just about the limit of what you want to safely draw from your USB bus and the Arduino's voltage regulator.
- So: it's ok to power your LED strip from your Arduino 5V pin, as long as you don't have them all on at maximum brightness for any length of time.
- It's best practice to connect your 5V / 2000mA DC power supply to your Arduino, and then power your LED strip with **Vin** and **GND.**
- If you ever have more than ~10 dotstar LEDs, you definitely need an external supply.

POWER CONSIDERATIONS

- If you are running your LED strip from an external power supply via Vin, it's a good idea to connect a large capacitor between the power and ground of your LED strip. This protects the LEDs in case of any large voltage spikes.
- You have a large-ish capacitor in your lab kit (10 μF). It's the biggest one, an it's brown in color. Connect it between the Vin and GND pins. The short leg of the capacitor, adjacent to the silver stripe, goes to ground