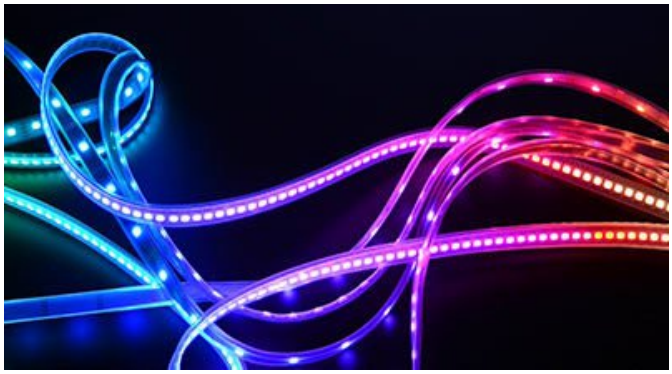
Adafruit DotStar LEDs

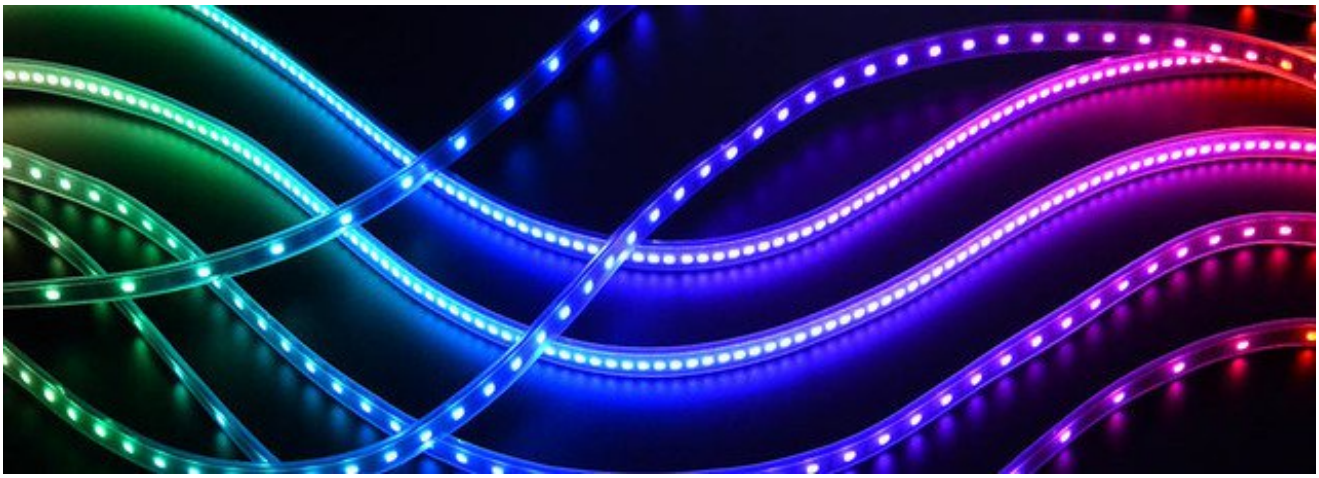
Created by Phillip Burgess



Last updated on 2016-10-04 07:27:00 PM UTC

Guide Contents

Overview



NeoPixel LEDs are the bee's knees, but in a few scenarios they come up short… connecting odd microcontrollers that can't match their strict timing, or fast-moving persistence-of-vision displays. Adafruit DotStar strips deliver high speed PWM and an easy-to-drive two-wire interface, bridging the gaps in the spectrum of awesome.

DotStars vs NeoPixels

The basic idea behind DotStars and NeoPixels is the same: a continuous string of individually-addressable RGB LEDs, driven by a microcontroller. The way each goes about it is a little different. DotStars *aren't necessarily a better thing in every situation*…there are tradeoffs, each has its pros and cons to consider…

DotStars

+ Extremely fast data[1] and PWM[2] rates, suitable for persistence-of-vision displays.

+ Easier to interface to a broader range of devices; no strict signal timing requirements.

NeoPixels

+ More affordable.

+ Wide range of form-factors (pixels, rings, matrices, etc.).

+ Works from a single microcontroller pin.

– Strict 800 KHz data rate; not all systems can

+ Works with Servo library, tone() function, interrupts, etc.

– Slightly more expensive.

– Fewer available form factors; strips and flex matrices only.

– Needs two pins for control.

generate this, and speed is a bottleneck on very long strands.

– 400 Hz refresh/PWM rate not suitable for persistence-of-vision effects.

– Not compatible with interrupts (e.g. Arduino Servo library or tone() function).

[1] Up to 8 MHz on Arduino, up to 32 MHz on Raspberry Pi.

[2] 19.2 KHz.

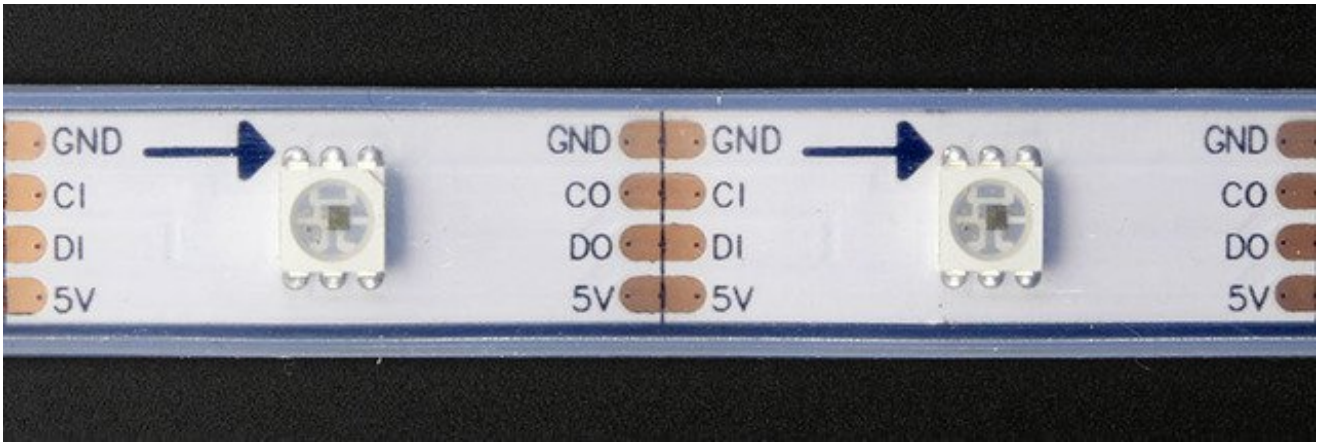
Powering DotStar LEDs

The power requirements for DotStars are pretty much identical to NeoPixels…in fact, we'll simply refer you to the relevant page of the NeoPixel Überguide for pointers on estimating and routing power (http://adafru.it/iZe). In summary:

- Estimate up to 60 milliamps peak for each pixel at full brightness white.
- For long strips, try to add a power tap every meter or so. This prevents a brown-out effect toward the end of the strip.
- It remains to be seen whether DotStars are as delicate as NeoPixels…but adding the power capacitor is a good idea anyway, do it!
- A ground connection is required between the microcontroller and strip, in addition to the signal lines.

Connecting DotStar LEDs

Our LED suppliers sometimes make unannounced production changes to the wiring. Therefore, **the best way to identify connections is a close visual examination of the strip.**



First, look for arrows printed along the strip next to each LED. These show the direction of data moving down the strip…your microcontroller connects at the originating (“in”) end, the arrows point toward the “out” end. (In the photo above, our microcontroller would be located off the left side.)

Second, look for labels on the strip to identify the function and order of the four wires: ground, 5 Volts, data and clock…usually labeled GND, 5V, D or DI (data input) and C or CI (clock input). These will go to corresponding pins on your microcontroller and power supply.

Again, due to production variations, you can’t always count on wire colors or plug genders as a reliable indication of function, even if ordered at the same time. Take a close look to confirm before connecting anything.

DotStars are 5V devices. While you might get them to respond to 3.3V signals,*this is not a guaranteed thing and should not be counted on.* For low-voltage microcontrollers and systems such as Raspberry Pi, a logic level shifter (e.g. 74AHCT125 (http://adafru.it/e5g)) is recommended for both the data and clock pins.

Libraries and Example Code

We have an Arduino library (http://adafru.it/ej7) to get you started:

Click to download Adafruit_DotStar library for Arduino
http://adafru.it/eio

Unzip and rename the resulting folder “Adafruit_DotStar”, move it to your Arduino/Libraries folder and restart the Arduino IDE. Library installation is a frequent stumbling block…if you need assistance, our All About Arduino Libraries (http://adafru.it/dit) guide spells it out in detail!

The “strandtest” example shows basic library use; declaring a strip object, setting LED colors and pushing this data to the strip via the show() method. For more advanced examples, nearly any NeoPixel code should compile and run with DotStars, just changing the library #include and the strip declaration.

Another production batch variation is the order in which color bytes are sent. Your own code can always use red, green, blue order when specifying colors, but the library needs to know the LED color sequence so it can adapt…you can see this as an optional last parameter to the constructor (change 'BRG' to 'GRB' or whatever data order your LEDs require):

```
Adafruit_DotStar strip = Adafruit_DotStar(60, 4, 5, DOTSTAR_BRG);
```

Another option for Arduino is the FastLED library (http://adafru.it/eip), featuring cutting-edge code with more color-handling and mathematical support functions. However, it’s a little more tricky to use…so, if connecting DotStars for the first time, we ask that you **start with the Adafruit_DotStar library.** Once the hardware is confirmed working, you can then graduate to whatever code or library you’d like!

There’s also a Python module (http://adafru.it/eiq) that works with **Raspberry Pi:**

Click to download Adafruit_DotStar Python module for Raspberry Pi
http://adafru.it/ek2

The Python code includes three examples. One is a simple strand test just to confirm that the LEDs are working (start here). The other two show how light painting and persistence-of-vision can be implemented using the Python Image module, which must first be installed with:

```
sudo apt-get install python-imaging
```

To use the SPI output mode in the examples (as opposed to the default "bit-bang" mode), you must enable the SPI kernel module in raspi-config (under "Advanced Options").

The Python code is modeled after the Arduino library (in turn based on the NeoPixel library), so the available functions are essentially the same, and existing NeoPixel examples can probably be ported with minimal fuss.

The Raspberry Pi is a 3.3V device. See notes above (“Connecting DotStar LEDs”) regarding level-shifting.

DotStar Matrices

We also sell these lovely DotStars in MATRIX form! This gives you a grid of LEDs for 2D blinkin-fun.

You can easily turn a flexible DotStar matrix into your very own glowing canvas by using the Adafruit DotStarMatrix library (http://adafru.it/iZf).

We have the flexible matrices in multiple size:

- Rectangular 8x32 (256 pixels) (http://adafru.it/2736)
- Square 16x16 (256 pixels) (http://adafru.it/2735)
- Square 8x8 (64 pixels) (http://adafru.it/2734)

To get going, install the Adafruit DotStarMatrix library (http://adafru.it/iZf)

You can download a zip of the latest DotStarMatrix library by clicking below

Download Adafruit_DotStarMatrix
http://adafru.it/iZA

Unzip and rename the resulting folder “Adafruit_DotStarMatrix”, making sure it contains Adafruit_DotStarMatrix.cpp. Move that folder to your ArduinoSketchBook/Libraries folder and restart the Arduino IDE. Library installation is a frequent stumbling block…if you need assistance, our All About Arduino Libraries (http://adafru.it/dit) guide spells it out in detail!

You'll **also** need the Adafruit_GFX library, if you dont have it already,download it from here (http://adafru.it/aJa). Or by clicking below for the zip

Download Adafruit_GFX
http://adafru.it/aJa

Once you have both libraries, restart the IDE and open up the**File->Examples->Adafruit_DotStarMatrix->matrixtest** example, and adjust the data/clock lines to what you're planning on using

```
#define DATAPIN  4
#define CLOCKPIN 5
```

Then, depending on your matrix size & style, you'll have to also adjust this line:

```
Adafruit_DotStarMatrix matrix = Adafruit_DotStarMatrix(
```

```
  8, 8, DATAPIN, CLOCKPIN,
  DS_MATRIX_TOP     + DS_MATRIX_RIGHT +
  DS_MATRIX_COLUMNS + DS_MATRIX_ZIGZAG,
  DOTSTAR_BGR);
```

The first four numbers are easiest to set, the first 2 are the size of the matrix, just put in 8, 8 or 16, 16 or whatever. The Data and Clock pins are also set.

The argument to the object is a bunch of 'flags' letting it know which way the matrix goes. You will likely have to mess with these a little depending on how your matrix is set up.

Chances are you want to use:

- **DS_MATRIX_PROGRESSIVE** for purchased flexible matrices, rather than **DS_MATRIX_ZIGZAG** which is often used for DIY matrices
- **DOTSTAR_BGR** for almost all matrices, Dotstars had a 'color swap' but that was in late 2014 so try this one first

Then switch around **DS_MATRIX_TOP**/**DS_MATRIX_BOTTOM**, **DS_MATRIX_RIGHT**/**DS_MATRIX_LEFT** and **DS_MATRIX_ROWS**/**DS_MATRIX_COLUMNS** until you get the orientation and pixel direction you want.