

---

# **Adafruit DotStar Library Documentation**

***Release 1.0***

**Scott Shawcroft, Limor Fried  
Damien P. George**

**Feb 26, 2018**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_dotstar - DotStar strip driver . . . . .	12
<b>6</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



Higher level DotStar driver that presents the strip as a sequence. It is the same api as the [NeoPixel library](#).

Colors are stored as tuples by default. However, you can also use int hex syntax to set values similar to colors on the web. For example, `0x100000` (`#100000` on the web) is equivalent to `(0x10, 0, 0)`.

---

**Note:** The int hex API represents the brightness of the white pixel when present by setting the RGB channels to identical values. For example, full white is `0xffffffff` but is actually `(0, 0, 0, 0xff)` in the tuple syntax. Setting a pixel value with an int will use the white pixel if the RGB channels are identical. For full, independent, control of each color component use the tuple syntax.

---



# CHAPTER 1

---

## Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).





## CHAPTER 2

---

### Usage Example

---

This example demonstrates the library with the single built-in DotStar on the [Trinket M0](#) and [Gemma M0](#).

```
import board
import adafruit_dotstar

pixels = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1)
pixels[0] = (10, 0, 0)
```



## CHAPTER 3

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.



## CHAPTER 4

---

### Building locally

---

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-dotstar --
↳library_location .
```

### 4.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 5.1: examples/dotstar\_simpletest.py

```
1 import time
2 import random
3 import board
4 import adafruit_dotstar as dotstar
5
6 # One pixel connected internally on a GEMMA M0
7 dots = dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1, brightness=0.2)
8
9 # With a Dotstar Digital LEB Strip with 30 lights
10 #dots = dotstar.DotStar(board.SCK, board.MOSI, 30, brightness=0.2)
11
12 ##### HELPERS #####
13
14 # a random color 0 -> 224
15 def random_color():
16     return random.randrange(0, 7) * 32
17
18 ##### MAIN LOOP #####
19 n_dots = len(dots)
20 while True:
21     #fill each dot with a random color
22     for dot in range(n_dots):
23         dots[dot] = (random_color(), random_color(), random_color())
24
25     # show all dots in strip
26     dots.show()
27
28     time.sleep(.25)
```

## 5.2 adafruit\_dotstar - DotStar strip driver

- Author(s): Damien P. George, Limor Fried & Scott Shawcroft

**class** `adafruit_dotstar.DotStar` (*clock*, *data*, *n*, \*, *brightness*=1.0, *auto\_write*=True)

A sequence of dotstars.

### Parameters

- **clock** (*Pin*) – The pin to output dotstar clock on.
- **data** (*Pin*) – The pin to output dotstar data on.
- **n** (*int*) – The number of dotstars in the chain
- **brightness** (*float*) – Brightness of the pixels between 0.0 and 1.0
- **auto\_write** (*bool*) – True if the dotstars should immediately change when set. If False, *show* must be called explicitly.

Example for Gemma M0:

```
import adafruit_dotstar
import time
from board import *

RED = 0x100000

with adafruit_dotstar.DotStar(APA102_SCK, APA102_MOSI, 1) as pixels:
    pixels[0] = RED
    time.sleep(2)
```

### **brightness**

Overall brightness of the pixel

### **deinit** ()

Blank out the DotStars and release the resources.

### **fill** (*color*)

Colors all pixels the given *\*color\**.

### **show** ()

Shows the new colors on the pixels themselves if they haven't already been autowritten.

The colors may or may not be showing after this function returns because it may be done asynchronously.



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### a

adafruit\_dotstar, [11](#)



### A

`adafruit_dotstar` (module), [11](#)

### B

`brightness` (`adafruit_dotstar.DotStar` attribute), [12](#)

### D

`deinit()` (`adafruit_dotstar.DotStar` method), [12](#)

`DotStar` (class in `adafruit_dotstar`), [12](#)

### F

`fill()` (`adafruit_dotstar.DotStar` method), [12](#)

### S

`show()` (`adafruit_dotstar.DotStar` method), [12](#)