**# Particle-NeoPixel**

A library for manipulating NeoPixel RGB LEDs for the:

Particle Core, Photon, P1, Electron, Argon, Boron, Xenon and RedBear Duo

Implementation based on Adafruit's NeoPixel Library.

**## Supported Pixel Types**

- 800 KHz WS2812, WS2812B, WS2813 and 400kHz bitstream and WS2811

- 800 KHz bitstream SK6812RGBW (NeoPixel RGBW pixel strips)

    (use 'SK6812RGBW' as PIXEL\_TYPE)

The most common kinds are WS2812/WS2813 (6-pin part), WS2812B (4-pin part) and SK6812RGBW (3 colors + white).

**#### Also supports these less common pixels**

- Radio Shack Tri-Color LED Strip with TM1803 controller 400kHz bitstream.

- TM1829 pixels, many [details here.](https://community.particle.io/t/neopixel-library-for-tm1829-controller-resolved/5363)

- Some functions from the [MessageTorch library](https://github.com/plan44/messagetorch/blob/master/messagetorch.cpp#L58-L134) have been added.

- SK6812MINI "NeoPixel Mini" (use 'WS2812B' as `PIXEL\_TYPE`)

**## Usage**

Set up the hardware:

- A NeoPixel digital RGB LED (get at [adafruit.com](https://www.adafruit.com))

- or a Radio Shack Tri-Color LED Strip (get at [radioshack.com](https://www.radioshack.com))

- A power supply or breakout board to supply NeoPixel's with 5V

Flash the [rainbow example](examples/a-rainbow/a-rainbow.cpp). With the

[Particle CLI](https://docs.particle.io/guide/tools-and-features/cli/)

do `particle flash <my\_device> examples/a-rainbow`

Adapt it to your needs while keeping this general structure:

```cpp

Adafruit\_NeoPixel strip(PIXEL\_COUNT, PIXEL\_PIN, PIXEL\_TYPE);

void setup() {

  strip.begin();

  strip.show();

}

void loop() {

  // change your pixel colors and call strip.show() again

}

```

**## Documentation**

**### `Adafruit\_NeoPixel`**

```

// IMPORTANT: Set pixel COUNT, PIN and TYPE

#define PIXEL\_COUNT 10

#define PIXEL\_PIN D2

#define PIXEL\_TYPE WS2812B

Adafruit\_NeoPixel strip(PIXEL\_COUNT, PIXEL\_PIN, PIXEL\_TYPE);

```

Creates an object to interact wth a NeoPixel strip.

`PIXEL\_COUNT` is the number of pixels in strip.

*\_Note: for some stripes like those with the TM1829, you need to count the number of segments, i.e. the number of controllers in your stripe, not the number of individual LEDs!\_*

`PIXEL\_PIN` is the pin number where your NeoPixel are connected (A0-A7, D0-D7, etc). If omitted, D2 is used.

On Photon, Electron, P1, Core and Duo, any pin can be used for Neopixel.

On the Argon, Boron and Xenon, only these pins can be used for Neopixel:

- D2, D3, A4, A5

- D4, D6, D7, D8

- A0, A1, A2, A3

In addition on the Argon/Boron/Xenon, only one pin per group can be used at a time. So it's OK to have one Adafruit\_NeoPixel

instance on pin D2 and another one on pin A2, but it's not possible to have one on pin A0 and another

one on pin A1.

`PIXEL\_TYPE` is the type of LED, one of WS2811, WS2812, WS2812B, WS2812B2, WS2813, TM1803, TM1829, SK6812RGBW. If omitted, WS2812B is used.

\_Note: For legacy 50us reset pulse timing on WS2812/WS2812B or WS2812B2, select WS2812B\_FAST or WS2812B2\_FAST respectively.  Otherwise, 300us timing will be used.\_

*\_Note: RGB order is automatically applied to WS2811, WS2812/WS2812B/WS2812B2/WS2813/TM1803 is GRB order.\_*

**### `begin`**

`strip.begin();`

Sets up the pin used for the NeoPixel strip.

**### `setPixelColor`**

**### `setColor`**

```

strip.setPixelColor(num, red, green, blue);

strip.setPixelColor(num, red, green, blue, white);

strip.setPixelColor(num, color);

strip.setColor(num, red, green, blue);

strip.setColor(num, red, green, blue, white);

```

Set the color of LED number `num` (0 to `PIXEL\_COUNT-1`). `red`,

`green`, `blue`, `white` are between 0 and 255. White is only used for

RGBW type pixels. `color` is a color returned from [`Color`](#color).

The brightness set with `setBrightness` will modify the color before it

is applied to the LED.

**### `show`**

`strip.show();`

Displays the colors on the NeoPixel strip that were set with `setPixelColor` and other calls that change the color of LEDs.

This function takes some time to run (more time the more LEDs you have) and disables interrupts while running.

**### `clear`**

`strip.clear();`

Set all LED color to off. Will take effect on next `show()`.

**### `setBrightness`**

`strip.setBrightness(brightness);`

Make the LED less bright. `brightness` is from 0 (off) to 255 (max brightness) and defaults to 255.

This factor is not linear: 128 is not visibly half as bright as 255 but almost as bright.

**### `getBrightness`**

`uint8\_t brightness = strip.getBrightness();`

Get the current brightness.

**### `setColorScaled`**

```

strip.setColorScaled(num, red, green, blue, scaling);

strip.setColorScaled(num, red, green, blue, white, scaling);

```

Set the color of LED number `num` and scale that color non-linearly according to the `scaling` parameter (0 to 255).

**### `setColorDimmed`**

```

strip.setColorDimmed(num, red, green, blue, brightness);

strip.setColorDimmed(num, red, green, blue, white, brightness);

```

Set the color of LED number `num` and dim that color linearly according to the `brightness` parameter (0 to 255). In this case 128 should look half as bright as 255.

**### `Color`**

```

uint32\_t color = strip.Color(red, green, blue);

uint32\_t color = strip.Color(red, green, blue, white);

```

Make a color from component colors. Useful if you want to store colors in a variable or pass them as function arguments.

**### `getPixelColor`**

`uint32\_t color = strip.getPixelColor();`

Get the current color of an LED in the same format as [`Color`](#color).

**### `setPin`**

`strip.setPin(pinNumber);`

Change the pin used for the NeoPixel strip.

**### `updateLength`**

`strip.updateLength(n);`

Change the number of LEDs in the NeoPixel strip.

**### `getPixels`**

`uint8\_t \*pixels = strip.getPixels();`

Get the raw color data for the LEDs.

**### `getNumLeds`**

**### `numPixels`**

```

uint16\_t n = strip.getNumLeds();

uint16\_t n = strip.numPixels();

```

Get the number of LEDs in the NeoPixel strip. `numPixels` is an alias for `getNumLeds`.

**## Nuances**

- Make sure get the # of pixels, pin number, type of pixels correct

- NeoPixels require 5V logic level inputs and the Spark Core and Photon only have 3.3V logic level digital outputs. Level shifting from 3.3V to 5V is necessary, the Particle Shield Shield has the [TXB0108PWR](http://www.digikey.com/product-search/en?pv7=2&k=TXB0108PWR) 3.3V to 5V level shifter built in (but has been known to oscillate at 50MHz with wire length longer than 6"), alternatively you can wire up your own with a [SN74HCT245N](http://www.digikey.com/product-detail/en/SN74HCT245N/296-1612-5-ND/277258), or [SN74HCT125N](http://www.digikey.com/product-detail/en/SN74HCT125N/296-8386-5-ND/376860). These are rock solid.

- To reduce NeoPixel burnout risk, add 1000 uF capacitor across pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input and minimize distance between device and first pixel.  Avoid connecting on a live circuit. If you must, connect GND first.

- Don't use `getPixelColor()` to move pixel data around when you are also using `setBrightness()`.  When the brightness is set, all `setPixelColor()` calls will end up scaling colors to dim them before they are stored in memory.  When using `getPixelColor()` the stored dimmed color is rescaled back up to the original color.  However, due to some loss of precision with the math, it is not possible to recreate this color data perfectly.  This is especially true with low brightness values.  If you `get` and `set` color data repeatedly with a dimmed pixel, it will eventually continue to decrease in value until it is equal to zero.

- When changing the brightness, always call `setPixelColor()` first with fresh un-dimmed color data, then call `setBrightness()`, and finally `show()`.

**## References**

- NeoPixel Guide: https://learn.adafruit.com/adafruit-neopixel-uberguide

- Quad Level Shifter IC: [SN74ACHT125N](https://www.adafruit.com/product/1787) (Adafruit)

- Quad Level Shifter IC: [SN74HCT125N](http://www.digikey.com/product-detail/en/SN74HCT125N/296-8386-5-ND/376860) (Digikey)

- Quad Level Shifter IC: [SN74AHCT125N](http://www.digikey.com/product-detail/en/SN74AHCT125N/296-4655-5-ND/375798) (Digikey)

**## License**

Copyright 2014-2018 Technobly, Julien Vanier, Cullen Shane, Phil Burgess

Released under the LGPL license