Arduino NeoPixel Example Guide

NeoPixel Library Reference

The Adafruit Neopixel library has a few functions already written so that you don't have to. The code is not very well documented if you don't know where to look sadly (welcome to the world of free code). This is a brief run down of what is going on so you can reference it later.

When you create a NeoPixel object in code, it will always be in the format:

```
Adafruit_NeoPixel pixelRing = Adafruit_NeoPixel(numberOfPixels, pixelDataPin, pixelType);
```

Thankfully you shouldn't need to worry about the `pixelType` as it should always be the same (i.e. `NEO_GRB + NEO_KHZ800`) if you are using the pixel rings you were given. If you buy extra NeoPixels or other addressable LEDs then you may need to change this.

An `Adafruit_NeoPixel` object can perform the functions listed below. Functions are always called in the form `.functionName` written after whatever name you gave the NeoPixel object
(e.g. `nameGivenToPixelRing.clear()`)

These functions should always change to a nice orange so that you know you are on the right track (apart from `.fill`, at least for me).

`.begin()`

You will need to use this in the `setup()` of your code. It initialises the pixels and does some background magic so that all the functions work properly.

`.setPixelColor(n, r, g, b)`

Set the colour of pixel index n to the RGB colour with values `r`, `g` and `b`. We always index starting from 0, so if you have 10 pixels, the first pixel would be 0 and the last would be 9. Setting the first pixel to red would look like `.setPixelColor(0, 255, 0, 0)`

`.setPixelColor(n, c)`

Just like the above, except the RGB value is bundled into one Hex value. If you have used Adobe or digital colour swatches, then this will look familiar. Writing the colour in hex, the first 2 digits are the red value, the next 2 are green and the last 2 are blue. A hex number always begins with 0x. So, a bright green colour would have a hex value `0x00FF00`. Setting the first pixel to red would look like `.setPixelColor(0, 0xFF0000)`. You can see this used in the SimpleNeoPixels example (line 25).

`.fill(c, i, n)`
Fill `n` number of pixels to the colour `c` starting from pixel `i`. This is used in the ColourWheel example (line 26).

`.show()`
After you have changed the colour of the pixels we need to let them know that they should update. This is when we call `.show()` .

`.clear()`
To turn all the pixels off, call this function. Don't for get to call `.show()` afterwards.

`.setBrightness(b)`
Changes the intensity of the current RGB colour of each pixel. Remember, that since there is a limit to the RGB values, you can lose some of the original hue. Used in the PixelSensorBrightnessExample

`.getBrightness()`
Get the value of the brightness that is currently set

`.getPixelColor(n)`
Get the colour value of pixel `n`

`.numPixels()`
Get the number of pixels that are currently controllable.

Examples

SimpleNeoPixels

A concise example showing how to light up some pixels, one by one, and demonstrating how to use `delay()` and `for` loops to achieve this. Contrast the number of lines and comments in this example with the Adafruit ‘simple’ example. Both achieve the same result.

Colour Wheel

This example demonstrates writing and then using a function. Line 10 and 11 should be changed to match your wiring and pixel ring. Line 26 is where we set the actual colour

PixelSensor

No we start getting into some fun stuff. This example will light up `n` number of pixels depending on the distance that is read. You will need to download the Ultrasonic library (same way you downloaded the NeoPixel Library).

PixelSensorBrightness

Control brightness of pixels if distance measured is in a certain range.

PixelSensorBrightnessSmooth

Same as above except the transition between brightness values should be smoother. This I because we use a current and target brightness value. The speed at which we increment towards are target dictates how smooth the transition is,.