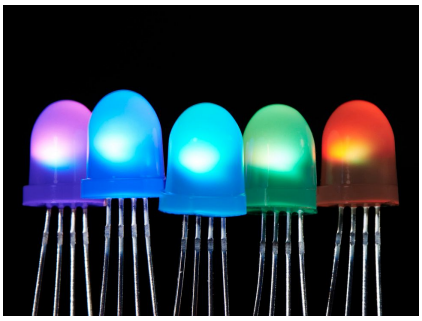
Individually Addressable RGB LEDs

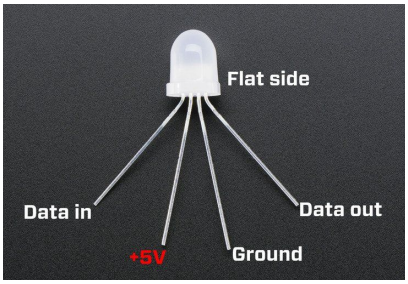


Source: https://www.adafruit.com/products/1734

The individually addressable RGB LED is both a functionally useful display element and can be used to create some fun outfits.

Individually addressable

These LEDs are part of a class of LEDs of all shapes and sizes that are called individually addressable.This means that when wired up in a chain, every LED can be individually set to a different color. Each LED requires a shared power supply and ground pin. Additionally, each LED has a Data_In and Data_Out pin. When connected in series, the first LED in the string grabs the first color setting and then passes the rest of the data to the next LED down the chain, which grabs the second color setting, etc.



Pin Description
Note: These devices need **+5V power supplies**.

The site https://www.adafruit.com/ provides a whole range of these LEDs under the brand name NeoPixel. Though just about any LED running on the same underlying chipsets can be controlled. The chipsets supported are often referred to as

WARNING: Applying power to these device backwards will release the magic smoke (i.e., destroy the device).

WS2812, WS2812B, WS2812B2, WS2811, TM1803, and TM1829.

NeoPixel Library

The library we need to operate these devices is called NEOPIXEL after a brand name of LED from www.adafruit.com .This library operates the NeoPixel as an object. First, we need to initialize the object:

```
#define PIXEL_PIN D4
#define PIXEL_COUNT 3
#define PIXEL_TYPE WS2811

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);
```

Then inside setup():

```
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
```

Once setup, there are a variety of functions to control the colors of the LEDs.Here's a short summary of the simplest functions:

Method (Function) Definition	**Description**
`strip.setPixelColor(PixelID, PackedColor)` `void = strip.setPixelColor(<uint16>,<uint32>)`[1]	Store the PackedColor (a 32-bit 'packed' RGB integer) for the n-th PixelID (zero indexed integer) in internal memory

`PackedColor = strip.Color(Red,Blue,Green)`
 	
`<uint32_t> = strip.Color(<uint8>,<uint8>,<uint8>)`
	Returns a 32-bit 'packed' integer representing the RGB values
	(ie., Red*256^2 + Blue*256 + Green)

`strip.show()`	Send stored configuration to LED strip in single burst transmission

Example Function	**Description**
`strip.setPixelColor(0,255)`	Store the intended color (bright green) for Pixel 0 (first pixel) in internal memory
`strip.Color(0,0,255)`	Returns a 32-bit 'packed' integer representing the RGB values (ie., bright blue)
`strip.show()`	Send stored configuration to LED strip in single burst transmission

[1] You can actually see the definition of this method in the file "neopixel.c" on line 699. This is a useful place to look if you need to figure out how an undocumented library functions.

The intended state of the whole string of NeoPixels is stored in memory with strip.setPixelColor() + strip.Color() and only updated when strip.show() is called.For example inside loop():

```
strip.setPixelColor(0, strip.Color(60, 50, 0));
strip.show();
```

Minimal Working Example

Wiring Diagram

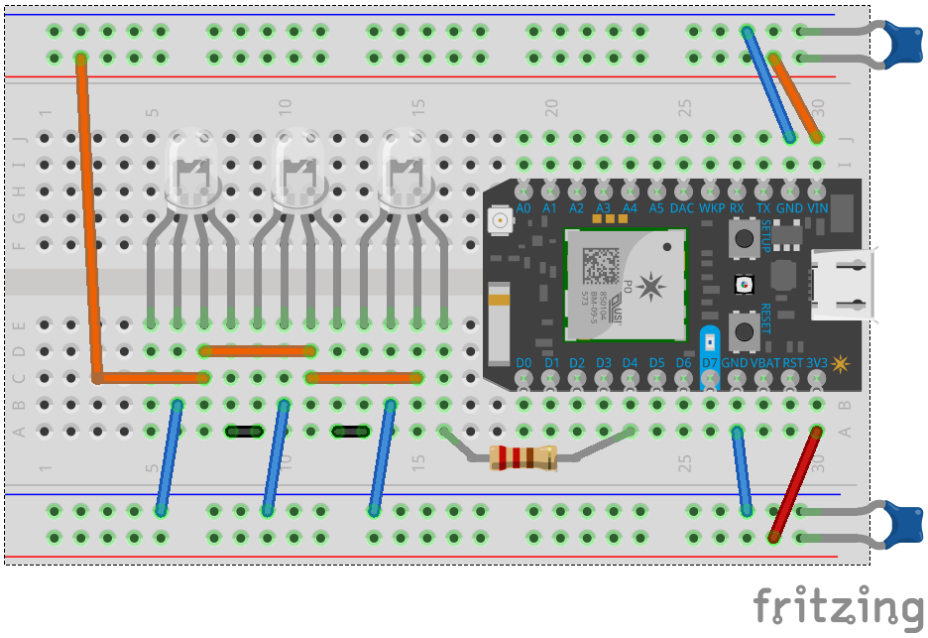


Figure 1: Example Circuit Wiring Diagram
(Note: LEDs' flat side is different than the previous diagram)

Example Code

```
#include "application.h"
#include "neopixel/neopixel.h"

SYSTEM_MODE(AUTOMATIC);

// IMPORTANT: Set pixel COUNT, PIN and TYPE
#define PIXEL_PIN D4
#define PIXEL_COUNT 3
#define PIXEL_TYPE WS2811

//Initialize the NeoPixel object
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN,
```

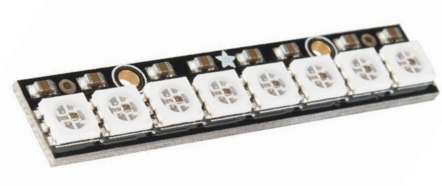
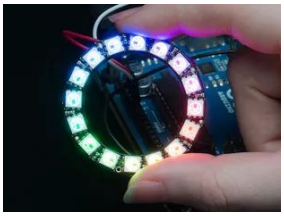
```
PIXEL_TYPE);

//Initialize loop index variable
int i;

void setup()
{
  Serial.begin(9600);
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
  i = 0;
}
void loop()
{

    if(i<128){
        //Set first pixel to green
        strip.setPixelColor(0, strip.Color(0, i, 0));
        //set second pixel to red/blue transition
        strip.setPixelColor(1, strip.Color(128-i, 0, i));
        //set third pixel to Gopher Gold!
        strip.setPixelColor(2, strip.Color(60, 50, 0));

        i=i+1;                  //increment loop index
    } else {
        i=0;  // Reset loop(max brightness is 255, but 128 is plenty bright)
    }
    strip.show();

    delay(20); // Wait 20ms before changing color again

    Serial.println(i);  //Print some debug information to the Serial port
    Serial.println(strip.Color(128-i, 0, i));

}
```

Additional Functions

Example Function	Description
`strip.getPixelColor(0)`	Returns 32-bit ‘packed’ RGB integer of the color of pixel 0 stored in local memory (first pixel).
`strip.numPixels()`	Returns the number of pixels
`strip.setBrightness(127)`	Attempts to scale the brightness of the stored colors by a factor of 127/255. These devices aren’t very linear and you can get color shifts doing this.

Other Shapes and Sizes of LEDs

Individually Addressable RGB LEDs come in many shapes and sizes:

	 
LED Arrays	LED Strips

 
Flexible substrates

Look around on Amazon, SparkFun, and AdaFruit for inspiration.

Warnings!!!

There are a couple of warnings that are *very* important when working with these super bright LEDs.

Current Limit on VIN

Warning!!! - You can damage your Photon if you connect too many LEDs to it. As a general rule of thumb, if you connect more than 8 LEDs to your Photon use the command "strip.setBrightness(127);" to adjust the maximum brightness to half. This should allow you to run safely up to 60 LEDs on a good USB port/powerstick.

Always use a protection resistor, decoupling cap, and never connect / disconnect the LEDs when they are operating.

More detail: Each Pixel will use about 60mA when bright white (RGB = 255,255,255).The VIN supply is limited by the capabilities of your USB port (Usually 500mA, but can range from 100mA to 2000mA).

A reasonable rule of thumb for the max LEDs to operate using most modern USB ports is:

$$N_{max,LEDs} = \frac{500mA}{60mA} \approx 8$$

When prototyping it is possible to operate a larger number of LEDs at lower brightness. The following formula applies when operating at lower than 255 brightness:

$$N_{max,LEDs} = \frac{500mA}{60mA} \frac{Brightness}{255}$$

For a project with a large number of LEDs (ie., a matrix display or flexible strip) it is often a good idea to get an external 5V power supply. For example:
http://www.amazon.com/Power-Adapter-2-1mm-Regulated-Supply/dp/B006QYRVU6

Microprocessor Protection

In certain circumstances, turning on and off large amounts of current in LEDs can damage the microprocessor. A resistor between the LEDs and the microprocessor will prevent this damage. This is the purpose of the 220 Ohm resistor shown in the example above. Adding this resistor is critical if you use an external 5V power supply for your LEDs.

Reverse Voltage

If you apply 5V in the wrong direction (ie., GND = 5.0V and +5V = 0.0V), you will destroy these devices.This commonly results in a popping noise and the magic smoke being released, however, it can potentially result in a small explosion and shards of plastic being ejected. Please be very careful and double check your wiring before applying power.

Additional Reading

https://learn.adafruit.com/adafruit-neopixel-uberguide/overview
https://learn.adafruit.com/adafruit-neopixel-uberguide/best-practices