Simple and Beautiful NeoPixel Holiday Lights

Created by Erin St Blaine

https://learn.adafruit.com/simple-beautiful-color-changing-light-strand

Last updated on 2024-06-03 02:16:09 PM EDT

Table of Contents

Introduction

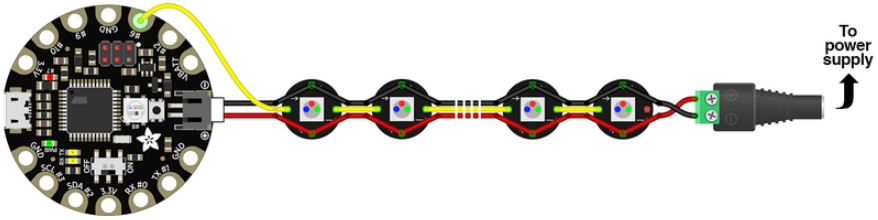
Get your Holiday cheer up and running with NeoPixels! Make a beautiful color shifting light strand to put on your Christmas tree, your house, or add color and light to your artwork. These weatherproof LED pixels work great indoors or outdoors. We've included some fancy software to run the lights in an ever-shifting and changing array of color palettes. It's not much more work to choose and customize your own color palettes to create the exact look you want.

This project is great for beginners. There is just a little bit of easy soldering involved. Most of the challenge is in getting the software set up, but once you've got that working, customization is as easy as copy and paste.

Pictures in this guide show either a Flora (http://adafru.it/659) or Circuit Playground (http://adafru.it/3333) microcontroller — either one works fine, or you might even consider the Gemma M0 (http://adafru.it/3501). Gemma M0 and Flora are a little less expensive, but Circuit Playground has a lot more options for future expansion: it has ten LEDs on its face, so with a few extra code tweaks you can use it as a glowing tree ornament, or the onboard microphone could make things sound-reactive if you add suitable code. Other microcontroller boards may work as well, as long as it's Arduino- and FastLED-compatible.

These light strands are sturdy and versatile. Have fun making everything glow!

Wiring Diagram

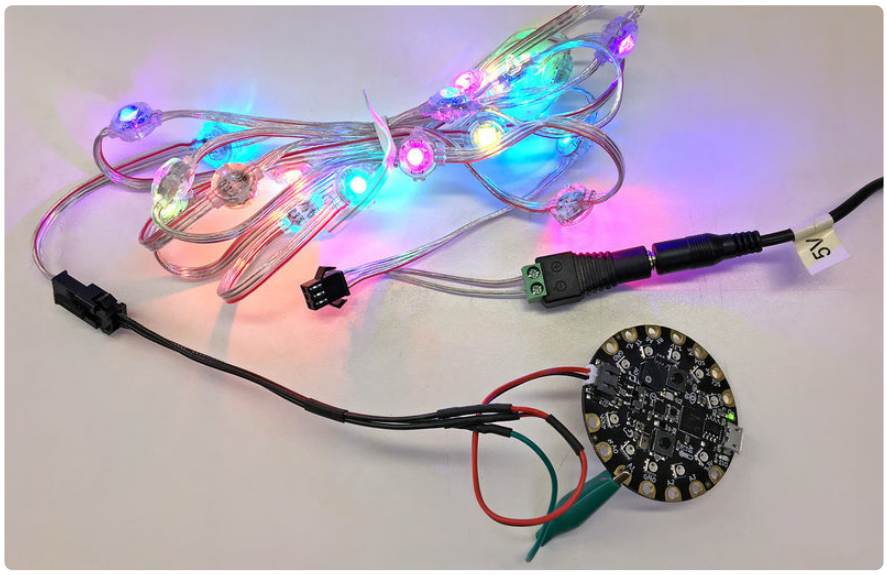


The great thing about these NeoPixel strands is that most of the wiring is already done for you, we just need to make some connections at the ends.

We'll show making an adapter for the microcontroller, then attach the NeoPixels with their included connector. Power connects at the opposite end of the light strand. That's something frequently overlooked with NeoPixel projects: while color data must

travel in a specific direction from “in” to “out,” **power can go either way.** In the diagram above, a 5V power supply is connected at the end of the strand, and we tap off this at the start of the strand to power the microcontroller, where the strand’s data input is also connected.

You can easily chain multiple light strands together with their included connectors. Just remember that if you have more than a few strands chained together, you may need a beefier power supply. (http://adafru.it/658) A modest 2 Amp supply is good for a couple strands, or the 4A power supply listed in the sidebar should work for at least 5-8 strands, depending on how you set the brightness level.



Software

It's a great idea to get your software all set up and loaded onto your board right away, to make testing your connections easier later on.

To get the code running you'll need:

1. Arduino IDE (1.8 or newer preferred)
2. Adafruit Board support (for Flora, Circuit Playground or Gemma M0)
3. FastLED Library

It's also helpful to install the Adafruit NeoPixel library, for testing purposes or in case FastLED is not working. (Use `Sketch→Include Library→Manage Libraries…` and search for Adafruit NeoPixel)

1. Arduino IDE

If you're not using a recent version of the Arduino IDE (1.8.3 or newer), this would be a good time to upgrade (https://adafru.it/fvm). If this is your first time using Arduino, head over to this guide to get it installed (https://adafru.it/jDQ). It's free and fairly simple to get set up.

2. Board Support

You'll need to tell the Arduino IDE which board you're using. This takes just a few minutes of setup, and you'll only need to do it once.

- Here's a step-by-step tutorial using the Flora (https://adafru.it/jDQ)
- Here is a step-by-step tutorial using the Circuit Playground (https://adafru.it/Cia)
- And one for Gemma M0 (https://adafru.it/Cib)

3. FastLED Library

This project uses the FastLED library, which provides high-performance math and color functions for NeoPixel projects. You'll need to download and install this manually:

Download FastLED Library for Arduino

https://adafru.it/Alj

Installing Arduino libraries is a frequent stumbling block, but we have a guide explaining the procedure (https://adafru.it/dit).

Upload the Code

Plug your microcontroller into your computer with a USB cable. In the Arduino IDE, go to **Tools** > **Boards** and select the name of the board. Then go to **Tools** > **Port** and select the board there too. (If it's not showing up there, be sure your microcontroller is plugged into your computer via USB)

This code was written by Mark Kriegsman. There are a few simple variables you can change: brightness of the LEDs and how quickly the lights shift between palettes. I'll get more into how to add and configure your own color palettes later on.

Don't forget to change the DATA_PIN and NUM_LEDS values to match your device and strand, before uploading!

```
// SPDX-FileCopyrightText: 2017 Erin St. Blaine for Adafruit Industries
//
// SPDX-License-Identifier: MIT


#include "FastLED.h"

// ColorWavesWithPalettes
// Animated shifting color waves, with several cross-fading color palettes.
// by Mark Kriegsman, August 2015
//
// Color palettes courtesy of cpt-city and its contributors:
//   http://soliton.vm.bytemark.co.uk/pub/cpt-city/
//
// Color palettes converted for FastLED using "PaletteKnife" v1:
//   http://fastled.io/tools/paletteknife/
//

//#if FASTLED_VERSION < 3001000
//#error "Requires FastLED 3.1 or later; check github for latest code."
//#endif

#define DATA_PIN   6
#define LED_TYPE    WS2812B
#define COLOR_ORDER GRB
#define NUM_LEDS    400       // Change this to reflect the number of LEDs you have
#define BRIGHTNESS  80      // Set Brightness here

CRGB leds[NUM_LEDS];

// ten seconds per color palette makes a good demo
// 20-120 is better for deployment
#define SECONDS_PER_PALETTE 20


void setup() {
  delay(3000); // 3 second delay for recovery

  // tell FastLED about the LED strip configuration
  FastLED.addLeds<LED_TYPE,DATA_PIN, COLOR_ORDER>(leds, NUM_LEDS)
    .setCorrection(TypicalLEDStrip) // cpt-city palettes have different color
balance
    .setDither(BRIGHTNESS < 255);

  // set master brightness control
  FastLED.setBrightness(BRIGHTNESS);
}

// Forward declarations of an array of cpt-city gradient palettes, and
// a count of how many there are.  The actual color palette definitions
// are at the bottom of this file.
extern const TProgmemRGBGradientPalettePtr gGradientPalettes[];
extern const uint8_t gGradientPaletteCount;

// Current palette number from the 'playlist' of color palettes
uint8_t gCurrentPaletteNumber = 0;

CRGBPalette16 gCurrentPalette( CRGB::Black);
CRGBPalette16 gTargetPalette( gGradientPalettes[0] );
```

```
// This function draws color waves with an ever-changing,
// widely-varying set of parameters, using a color palette.
void colorwaves( CRGB* ledarray, uint16_t numleds, CRGBPalette16& palette)
{
  static uint16_t sPseudotime = 0;
  static uint16_t sLastMillis = 0;
  static uint16_t sHue16 = 0;

  uint8_t sat8 = beatsin88( 87, 220, 250);
  uint8_t brightdepth = beatsin88( 341, 96, 224);
  uint16_t brightnessthetainc16 = beatsin88( 203, (25 * 256), (40 * 256));
  uint8_t msmultiplier = beatsin88(147, 23, 60);

  uint16_t hue16 = sHue16;//gHue * 256;
  uint16_t hueinc16 = beatsin88(113, 300, 1500);

  uint16_t ms = millis();
  uint16_t deltams = ms - sLastMillis ;
  sLastMillis  = ms;
  sPseudotime += deltams * msmultiplier;
  sHue16 += deltams * beatsin88( 400, 5,9);
  uint16_t brightnesstheta16 = sPseudotime;

  for( uint16_t i = 0 ; i < numleds; i++) {
    hue16 += hueinc16;
    uint8_t hue8 = hue16 / 256;
    uint16_t h16_128 = hue16 >> 7;
    if( h16_128 & 0x100) {
      hue8 = 255 - (h16_128 >> 1);
    } else {
      hue8 = h16_128 >> 1;
    }

    brightnesstheta16  += brightnessthetainc16;
    uint16_t b16 = sin16( brightnesstheta16  ) + 32768;

    uint16_t bri16 = (uint32_t)((uint32_t)b16 * (uint32_t)b16) / 65536;
    uint8_t bri8 = (uint32_t)(((uint32_t)bri16) * brightdepth) / 65536;
    bri8 += (255 - brightdepth);

    uint8_t index = hue8;
    //index = triwave8( index);
    index = scale8( index, 240);

    CRGB newcolor = ColorFromPalette( palette, index, bri8);

    uint16_t pixelnumber = i;
    pixelnumber = (numleds-1) - pixelnumber;

    nblend( ledarray[pixelnumber], newcolor, 128);
  }
}

// Alternate rendering function just scrolls the current palette
// across the defined LED strip.
void palettetest( CRGB* ledarray, uint16_t numleds, const CRGBPalette16&
gCurrentPalette)
{
  static uint8_t startindex = 0;
  startindex--;
  fill_palette( ledarray, numleds, startindex, (256 / NUM_LEDS) + 1,
gCurrentPalette, 255, LINEARBLEND);
}




// Gradient Color Palette definitions for 33 different cpt-city color palettes.
```

```
//    956 bytes of PROGMEM for all of the palettes together,
//   +618 bytes of PROGMEM for gradient palette code (AVR).
//  1,494 bytes total for all 34 color palettes and associated code.

// Gradient palette "ib_jul01_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ing/xmas/tn/ib_jul01.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 16 bytes of program space.

DEFINE_GRADIENT_PALETTE( ib_jul01_gp ) {
    0, 194,  1,  1,
   94,   1, 29, 18,
  132,  57,131, 28,
  255, 113,  1,  1};

// Gradient palette "es_vintage_57_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/vintage/tn/
es_vintage_57.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_vintage_57_gp ) {
    0,   2,  1,  1,
   53,  18,  1,  0,
  104,  69, 29,  1,
  153, 167,135, 10,
  255,  46, 56,  4};

// Gradient palette "es_vintage_01_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/vintage/tn/
es_vintage_01.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 32 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_vintage_01_gp ) {
    0,   4,  1,  1,
   51,  16,  0,  1,
   76,  97,104,  3,
  101, 255,131, 19,
  127,  67,  9,  4,
  153,  16,  0,  1,
  229,   4,  1,  1,
  255,   4,  1,  1};

// Gradient palette "es_rivendell_15_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/rivendell/tn/
es_rivendell_15.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_rivendell_15_gp ) {
    0,   1, 14,  5,
  101,  16, 36, 14,
  165,  56, 68, 30,
  242, 150,156, 99,
  255, 150,156, 99};

// Gradient palette "rgi_15_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ds/rgi/tn/rgi_15.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 36 bytes of program space.

DEFINE_GRADIENT_PALETTE( rgi_15_gp ) {
    0,   4,  1, 31,
   31,  55,  1, 16,
   63, 197,  3,  7,
   95,  59,  2, 17,
  127,   6,  2, 34,
  159,  39,  6, 33,
```

```
  191, 112, 13, 32,
  223,  56,  9, 35,
  255,  22,  6, 38};

// Gradient palette "retro2_16_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ma/retro2/tn/
retro2_16.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 8 bytes of program space.

DEFINE_GRADIENT_PALETTE( retro2_16_gp ) {
    0, 188,135,  1,
  255,  46,  7,  1};

// Gradient palette "Analogous_1_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/red/tn/
Analogous_1.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( Analogous_1_gp ) {
    0,   3,  0,255,
   63,  23,  0,255,
  127,  67,  0,255,
  191, 142,  0, 45,
  255, 255,  0,  0};

// Gradient palette "es_pinksplash_08_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/pink_splash/tn/
es_pinksplash_08.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_pinksplash_08_gp ) {
    0, 126, 11,255,
  127, 197,  1, 22,
  175, 210,157,172,
  221, 157,  3,112,
  255, 157,  3,112};

// Gradient palette "es_pinksplash_07_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/pink_splash/tn/
es_pinksplash_07.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_pinksplash_07_gp ) {
    0, 229,  1,  1,
   61, 242,  4, 63,
  101, 255, 12,255,
  127, 249, 81,252,
  153, 255, 11,235,
  193, 244,  5, 68,
  255, 232,  1,  5};

// Gradient palette "Coral_reef_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/other/tn/
Coral_reef.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 24 bytes of program space.

DEFINE_GRADIENT_PALETTE( Coral_reef_gp ) {
    0,  40,199,197,
   50,  10,152,155,
   96,   1,111,120,
   96,  43,127,162,
  139,  10, 73,111,
  255,   1, 34, 71};
```

```
// Gradient palette "es_ocean_breeze_068_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/ocean_breeze/tn/
es_ocean_breeze_068.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 24 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_ocean_breeze_068_gp ) {
    0, 100,156,153,
   51,   1, 99,137,
  101,   1, 68, 84,
  104,  35,142,168,
  178,   0, 63,117,
  255,   1, 10, 10};

// Gradient palette "es_ocean_breeze_036_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/ocean_breeze/tn/
es_ocean_breeze_036.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 16 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_ocean_breeze_036_gp ) {
    0,   1,  6,  7,
   89,   1, 99,111,
  153, 144,209,255,
  255,   0, 73, 82};

// Gradient palette "departure_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/mjf/tn/departure.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 88 bytes of program space.

DEFINE_GRADIENT_PALETTE( departure_gp ) {
    0,   8,  3,  0,
   42,  23,  7,  0,
   63,  75, 38,  6,
   84, 169, 99, 38,
  106, 213,169,119,
  116, 255,255,255,
  138, 135,255,138,
  148,  22,255, 24,
  170,   0,255,  0,
  191,   0,136,  0,
  212,   0, 55,  0,
  255,   0, 55,  0};

// Gradient palette "es_landscape_64_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/landscape/tn/
es_landscape_64.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 36 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_landscape_64_gp ) {
    0,   0,  0,  0,
   37,   2, 25,  1,
   76,  15,115,  5,
  127,  79,213,  1,
  128, 126,211, 47,
  130, 188,209,247,
  153, 144,182,205,
  204,  59,117,250,
  255,   1, 37,192};

// Gradient palette "es_landscape_33_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/landscape/tn/
es_landscape_33.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 24 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_landscape_33_gp ) {
```

```
    0,   1,  5,  0,
   19,  32, 23,  1,
   38, 161, 55,  1,
   63, 229,144,  1,
   66,  39,142, 74,
  255,   1,  4,  1};

// Gradient palette "rainbowsherbet_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ma/icecream/tn/
rainbowsherbet.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( rainbowsherbet_gp ) {
    0, 255, 33,  4,
   43, 255, 68, 25,
   86, 255,  7, 25,
  127, 255, 82,103,
  170, 255,255,242,
  209,  42,255, 22,
  255,  87,255, 65};

// Gradient palette "gr65_hult_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/hult/tn/gr65_hult.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 24 bytes of program space.

DEFINE_GRADIENT_PALETTE( gr65_hult_gp ) {
    0, 247,176,247,
   48, 255,136,255,
   89, 220, 29,226,
  160,   7, 82,178,
  216,   1,124,109,
  255,   1,124,109};

// Gradient palette "gr64_hult_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/hult/tn/gr64_hult.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 32 bytes of program space.

DEFINE_GRADIENT_PALETTE( gr64_hult_gp ) {
    0,   1,124,109,
   66,   1, 93, 79,
  104,  52, 65,  1,
  130, 115,127,  1,
  150,  52, 65,  1,
  201,   1, 86, 72,
  239,   0, 55, 45,
  255,   0, 55, 45};

// Gradient palette "GMT_drywet_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/gmt/tn/GMT_drywet.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( GMT_drywet_gp ) {
    0,  47, 30,  2,
   42, 213,147, 24,
   84, 103,219, 52,
  127,   3,219,207,
  170,   1, 48,214,
  212,   1,  1,111,
  255,   1,  7, 33};

// Gradient palette "ib15_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ing/general/tn/ib15.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 24 bytes of program space.
```

```
DEFINE_GRADIENT_PALETTE( ib15_gp ) {
    0, 113, 91,147,
   72, 157, 88, 78,
   89, 208, 85, 33,
  107, 255, 29, 11,
  141, 137, 31, 39,
  255,  59, 33, 89};

// Gradient palette "Fuschia_7_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/ds/fuschia/tn/
Fuschia-7.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( Fuschia_7_gp ) {
    0,  43,  3,153,
   63, 100,  4,103,
  127, 188,  5, 66,
  191, 161, 11,115,
  255, 135, 20,182};

// Gradient palette "es_emerald_dragon_08_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/emerald_dragon/tn/
es_emerald_dragon_08.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 16 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_emerald_dragon_08_gp ) {
    0,  97,255,  1,
  101,  47,133,  1,
  178,  13, 43,  1,
  255,   2, 10,  1};

// Gradient palette "lava_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/neota/elem/tn/lava.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 52 bytes of program space.

DEFINE_GRADIENT_PALETTE( lava_gp ) {
    0,   0,  0,  0,
   46,  18,  0,  0,
   96, 113,  0,  0,
  108, 142,  3,  1,
  119, 175, 17,  1,
  146, 213, 44,  2,
  174, 255, 82,  4,
  188, 255,115,  4,
  202, 255,156,  4,
  218, 255,203,  4,
  234, 255,255,  4,
  244, 255,255, 71,
  255, 255,255,255};

// Gradient palette "fire_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/neota/elem/tn/fire.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( fire_gp ) {
    0,   1,  1,  0,
   76,  32,  5,  0,
  146, 192, 24,  0,
  197, 220,105,  5,
  240, 252,255, 31,
  250, 252,255,111,
  255, 255,255,255};

// Gradient palette "Colorfull_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/atmospheric/tn/
```

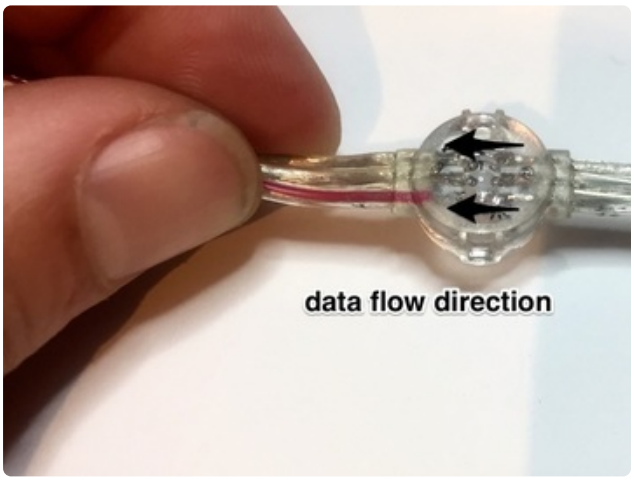
```
Colorfull.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 44 bytes of program space.

DEFINE_GRADIENT_PALETTE( Colorfull_gp ) {
    0,  10, 85,  5,
   25,  29,109, 18,
   60,  59,138, 42,
   93,  83, 99, 52,
  106, 110, 66, 64,
  109, 123, 49, 65,
  113, 139, 35, 66,
  116, 192,117, 98,
  124, 255,255,137,
  168, 100,180,155,
  255,  22,121,174};

// Gradient palette "Magenta_Evening_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/atmospheric/tn/
Magenta_Evening.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( Magenta_Evening_gp ) {
    0,  71, 27, 39,
   31, 130, 11, 51,
   63, 213,  2, 64,
   70, 232,  1, 66,
   76, 252,  1, 69,
  108, 123,  2, 51,
  255,  46,  9, 35};

// Gradient palette "Pink_Purple_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/atmospheric/tn/
Pink_Purple.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 44 bytes of program space.

DEFINE_GRADIENT_PALETTE( Pink_Purple_gp ) {
    0,  19,  2, 39,
   25,  26,  4, 45,
   51,  33,  6, 52,
   76,  68, 62,125,
  102, 118,187,240,
  109, 163,215,247,
  114, 217,244,255,
  122, 159,149,221,
  149, 113, 78,188,
  183, 128, 57,155,
  255, 146, 40,123};

// Gradient palette "Sunset_Real_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/atmospheric/tn/
Sunset_Real.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( Sunset_Real_gp ) {
    0, 120,  0,  0,
   22, 179, 22,  0,
   51, 255,104,  0,
   85, 167, 22, 18,
  135, 100,  0,103,
  198,  16,  0,130,
  255,   0,  0,160};

// Gradient palette "es_autumn_19_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/es/autumn/tn/
es_autumn_19.png.index.html
```

```
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 52 bytes of program space.

DEFINE_GRADIENT_PALETTE( es_autumn_19_gp ) {
    0,  26,  1,  1,
   51,  67,  4,  1,
   84, 118, 14,  1,
  104, 137,152, 52,
  112, 113, 65,  1,
  122, 133,149, 59,
  124, 137,152, 52,
  135, 113, 65,  1,
  142, 139,154, 46,
  163, 113, 13,  1,
  204,  55,  3,  1,
  249,  17,  1,  1,
  255,  17,  1,  1};

// Gradient palette "BlacK_Blue_Magenta_White_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/basic/tn/
BlacK_Blue_Magenta_White.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( BlacK_Blue_Magenta_White_gp ) {
    0,   0,  0,  0,
   42,   0,  0, 45,
   84,   0,  0,255,
  127,  42,  0,255,
  170, 255,  0,255,
  212, 255, 55,255,
  255, 255,255,255};

// Gradient palette "BlacK_Magenta_Red_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/basic/tn/
BlacK_Magenta_Red.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( BlacK_Magenta_Red_gp ) {
    0,   0,  0,  0,
   63,  42,  0, 45,
  127, 255,  0,255,
  191, 255,  0, 45,
  255, 255,  0,  0};

// Gradient palette "BlacK_Red_Magenta_Yellow_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/basic/tn/
BlacK_Red_Magenta_Yellow.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 28 bytes of program space.

DEFINE_GRADIENT_PALETTE( BlacK_Red_Magenta_Yellow_gp ) {
    0,   0,  0,  0,
   42,  42,  0,  0,
   84, 255,  0,  0,
  127, 255,  0, 45,
  170, 255,  0,255,
  212, 255, 55, 45,
  255, 255,255,  0};

// Gradient palette "Blue_Cyan_Yellow_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/nd/basic/tn/
Blue_Cyan_Yellow.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 20 bytes of program space.

DEFINE_GRADIENT_PALETTE( Blue_Cyan_Yellow_gp ) {
    0,   0,  0,255,
```

```
   63,   0, 55,255,
  127,   0,255,255,
  191,  42,255, 45,
  255, 255,255,  0};

  // Gradient palette "bhw1_28_gp", originally from
// http://soliton.vm.bytemark.co.uk/pub/cpt-city/bhw/bhw1/tn/bhw1_28.png.index.html
// converted for FastLED with gammas (2.6, 2.2, 2.5)
// Size: 32 bytes of program space.

DEFINE_GRADIENT_PALETTE( bhw1_28_gp ) {
    0,  75,  1,221,
   30, 252, 73,255,
   48, 169,  0,242,
  119,   0,149,242,
  170,  43,  0,242,
  206, 252, 73,255,
  232,  78, 12,214,
  255,   0,149,242};



// Single array of defined cpt-city color palettes.
// This will let us programmatically choose one based on
// a number, rather than having to activate each explicitly
// by name every time.
// Since it is const, this array could also be moved
// into PROGMEM to save SRAM, but for simplicity of illustration
// we'll keep it in a regular SRAM array.
//
// This list of color palettes acts as a "playlist"; you can
// add or delete, or re-arrange as you wish.
const TProgmemRGBGradientPalettePtr gGradientPalettes[] = {
  bhw1_28_gp,
  Sunset_Real_gp,
  es_rivendell_15_gp,
  es_ocean_breeze_036_gp,
  rgi_15_gp,
  retro2_16_gp,
  Analogous_1_gp,
  es_pinksplash_08_gp,
  Coral_reef_gp,
  es_ocean_breeze_068_gp,
  es_pinksplash_07_gp,
  es_vintage_01_gp,
  departure_gp,
  es_landscape_64_gp,
  es_landscape_33_gp,
  rainbowsherbet_gp,
  gr65_hult_gp,
  gr64_hult_gp,
  GMT_drywet_gp,
  ib_jul01_gp,
  es_vintage_57_gp,
  ib15_gp,
  Fuschia_7_gp,
  es_emerald_dragon_08_gp,
  lava_gp,
  fire_gp,
  Colorfull_gp,
  Magenta_Evening_gp,
  Pink_Purple_gp,
  es_autumn_19_gp,
  BlacK_Blue_Magenta_White_gp,
  BlacK_Magenta_Red_gp,
  BlacK_Red_Magenta_Yellow_gp,
  Blue_Cyan_Yellow_gp };
```

```
// Count of how many cpt-city gradients are defined:
const uint8_t gGradientPaletteCount =
  sizeof( gGradientPalettes) / sizeof( TProgmemRGBGradientPalettePtr );



  void loop()
{
  EVERY_N_SECONDS( SECONDS_PER_PALETTE ) {
    gCurrentPaletteNumber = addmod8( gCurrentPaletteNumber, 1,
gGradientPaletteCount);
    gTargetPalette = gGradientPalettes[ gCurrentPaletteNumber ];
  }

  EVERY_N_MILLISECONDS(40) {
    nblendPaletteTowardPalette( gCurrentPalette, gTargetPalette, 16);
  }

  colorwaves( leds, NUM_LEDS, gCurrentPalette);

  FastLED.show();
  FastLED.delay(20);
}
```

Assembly

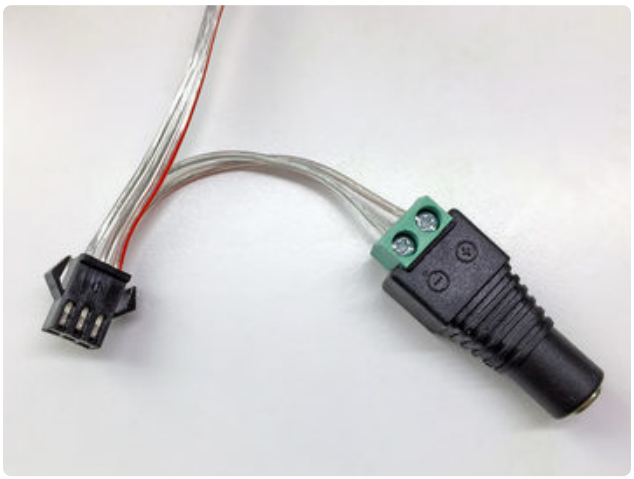


NeoPixel color data must travel from the “in” end of the strand to the “out” end. Examine the backs of the pixels very closely. Printed on the PCB, you might see either some data direction arrows, or the words “IN” and “OUT” between pads. The weatherproofing epoxy makes things murky and you may need to look at several pixels before finding one. This is important! If you connect the microcontroller to the wrong end, the lights won't work.

If you have multiple LED strands, plug them into each other to create one long überstrand.

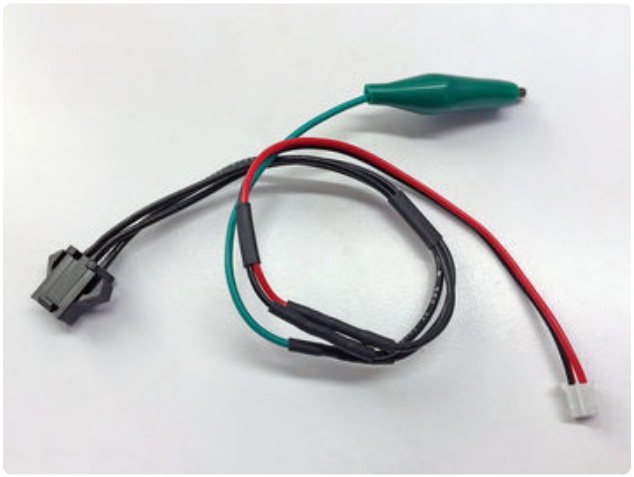
Each strand should have a pair of extra wires with exposed ends, for connecting power. Only **one** of these pairs is required. The rest should have their exposed tips trimmed flush or covered with tape or heat-shrink tubing to prevent electrical shorts.

For 1 to 3 linked strands, let’s use the pair of wires at the **end**. For longer strands, choose a pair near the **middle.**



Connect a female DC power adapter using the screw connectors, being **super extra careful** to **follow the polarity markings** stamped on the DC jack:

+ (plus) connects to the strand's **red** wire
– (minus) to the opposite wire
You may need to strip away a little extra insulation from the wires to make a good connection with the terminals. Tug a little on each wire to make sure it's firmly in place.



To connect the microcontroller at the "input" end of the strand, I soldered up this little **adapter cable** using a **3-pin JST plug** (to the LEDs), a **2-pin JST cable** (to the Circuit Playground, Flora or Gemma board) and one end of an **alligator jumper cable**.

Don't just follow the picture, take a good look at your actual hardware and be **super extra careful** to get the connections right:

The **RED** wire from the LED strand should connect to the **RED** wire on the 2-pin JST plug. (If using an all-black 3-pin JST cable, as shown here, carefully follow the wires from the LED strand all the way through.) The **OPPOSITE** wire from the LED strand (furthest from the red wire) should connect to the **BLACK** wire on the 2-pin JST plug. The **MIDDLE** wire should go to the '**gator clip**, which can then be clipped to the appropriate pad on the microcontroller board.

If you don't have these exact parts on-hand, it's usually okay to **improvise**.

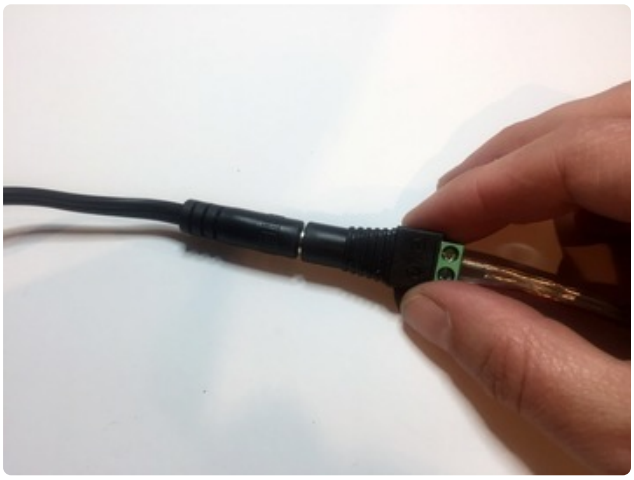
For example, if you don't have or are unwilling to sacrifice an **alligator cable**, you can **solder a wire directly** to one of the pads on the microcontroller board. This is easily

removed later by re-melting the solder and you can then use the board in other projects.

If you don't have a **3-pin JST plug**, and you're 100% okay sacrificing the "output" end of your LED strand, you can cut that plug off and use it as an input-end adapter. Do this at the mid-point between the plug and the last pixel, so there's enough wire to be useful. Was your DC jack connected to the extra power wires there? That's okay! Remember, the strand can accept power from either end.

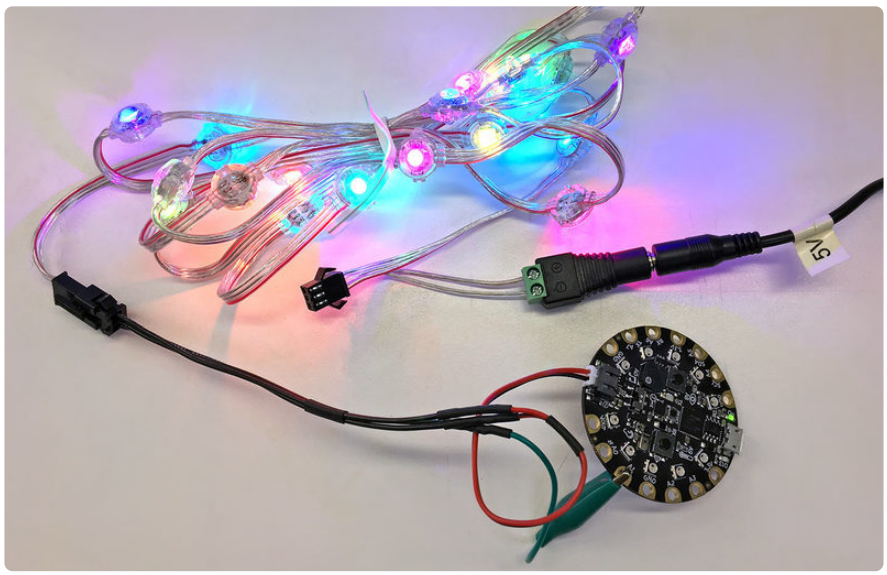
The wiring diagram and code both show the NeoPixel data connected to digital pin **6**. If using Circuit Playground or Gemma M0, the number and sequence of pads is different, and you'll need to **modify the code** to reflect what pin has the NeoPixels connected. It's fairly early in the sketch:

```
#define DATA_PIN   6
```



Double-check all your connections: 5V, ground and NeoPixel data. If you have a multimeter with a continuity beep function, the "–" (minus) terminal on the DC jack should beep in response to any "GND" pad on the microcontroller board.

If everything looks good, connect a power supply and watch the pretty light show!



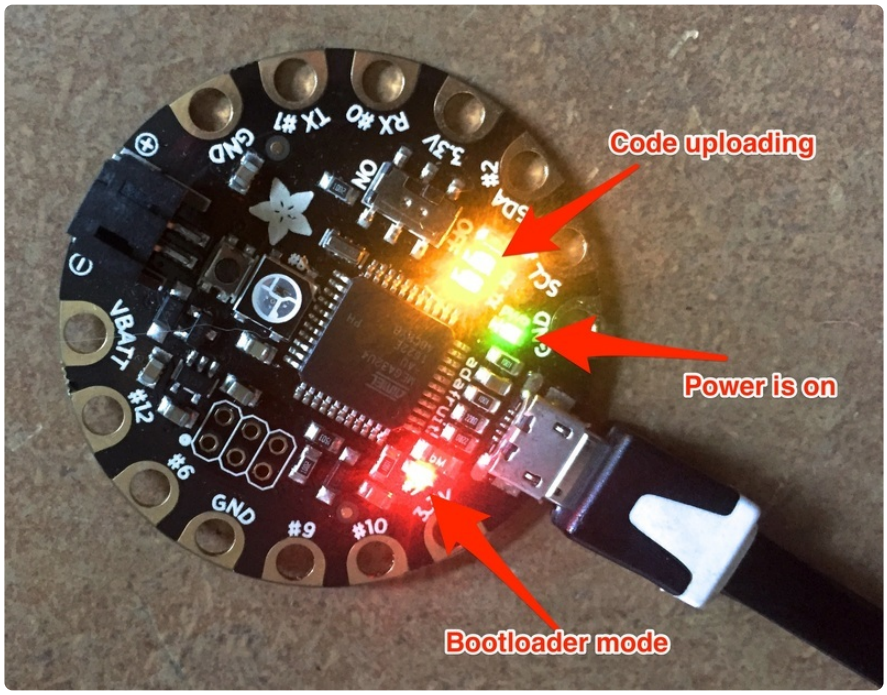
Troubleshooting

Lights Don't Come On

First, check all your connections and make sure they are tight. The screw terminal may be the culprit. Check to be sure your wires are stripped enough, and the terminal is clamping down on bare wire instead of on insulation. Tug on the wires to make sure they're in there really firmly.

On occasion you get a screw terminal that just won't hold the wires very well. Try replacing the DC jack if this seems to be the problem.

Next, check to be sure you've uploaded the code to your microcontroller. For the Flora: when an upload is successful you'll see a red light come on and slow-blink (that means you're in bootloader mode), then you'll see a yellow LED flash for a moment (that means the Flora is receiving code). If you don't see this light sequence on the Flora, your code didn't upload correctly. Try hitting the reset button and uploading again.



If it seems to be uploading correctly but the light strand still doesn't come on:

Be sure that your code and your actual physical pin connections match. The code expects you to have connected your lights to pin #6. If you've connected to a different pin, update the code to match.

Also make sure you've got the three connector wires going to the correct pins -- it's easy to get them mixed up.

Lastly, check again on the back of the pixels to make sure your Flora is attached at the "in" end of the strand and not the "out" end.

Code Won't Upload

Check to be sure that the correct board is selected from your Boards menu. Don't see it there? Head back to the **Software** page and follow the directions to get it installed.

Also check to be sure that under `Tools > Port ..` you can see and select your microcontroller. Don't see it there? Try a different USB cable, or try plugging your USB cable directly into your computer if it's going through a hub. Still don't see it? Try rebooting your computer.

Be sure you've installed the **FastLED** library (`Sketch > Include Library > Library Manager..`) You may need to update the library if you have an older version.

If you still can't get the code to upload, back up and try uploading sample sketches from the NeoPixel library Examples folder. Be sure you can get the **StrandTes**t sketch to work, then try this sketch again.

Lights Come on but Misbehave

If only some of the lights come on, take a look at the code you uploaded and make sure you've changed this line:

```
#define NUM_LEDS 400 // Change this to reflect the number of LEDs you have
```

..to reflect the actual number of LEDs you have. It's OK to make this number too big, but it's a problem if it's too small.

If you get just one or two lights coming on in white or green: You may have your power and ground connections switched. Double check to make sure you soldered the connector on right.

Creating Color Palettes

FastLED Color Palettes

Using the FastLED library opens up a world of LED coding tools that are fairly easy to use.

There's a fantastic tool called **PaletteKnife** that works great for artist-minded people who think in colors instead of in code, and this project uses it.

Watch the video above for a demo of just how easy this is.

Want to try PaletteKnife and grab some new color palettes?
Here's what you'll definitely need:

1. A basic understanding of how FastLED palettes are used. Check out the FastLED ColorPalette example, as well as FastLED Fire2012WithPalette example, both of which can be found in the FastLED library (https://adafru.it/ezj) Examples folder.
2. You'll need Chrome or Safari. PaletteKnife doesn't work with Firefox or IE.
3. You will also need a little patience; it's not terribly slick but it works a dream.

Basically, you browse to the color palette that you want, and click PaletteKnife (a bookmarklet) in your browser. Then you COPY the resulting code, and PASTE it into your FastLED animation source code.

Full instructions on how to get started with PaletteKnife are here:
http://fastled.io/tools/paletteknife/ (https://adafru.it/mcn)