Table of Contents

Introduction 3

Setup & Testing 4
• Adafruit Feather M0 Basic Proto — Arduino IDE Setup

Software 6
• Processing Downloads Page
• Edit Arduino Sketch
• Upload

OPC Strandtest 10

Wiring 12

LED Strip Wiring 13

Wig Assembly 16

Final Wiring 21

Finishing 28

Use It 30
Introduction

Imagine..

All your LED costume pieces in sync with each other. All of them listening and responding to the music around you. Your hand-made hat, wig, poi, and jacket all pulsing to the sound and visuals running on your laptop -- with no extra coding required.

Magic.

This guide uses the code and concepts from the [Lightship LED guide](https://learn.adafruit.com/lightship-led-guide) and integrates it into a costume piece, so you can roam around your dance club, playa party or DJ gig and let the visuals follow you wherever you go.

This is a pretty intense guide -- getting everything set up will take a bit of doing, but once you're done, all you have to do is press play -- and the LEDs will reflect whatever's happening on your computer screen.

I'll say that again: whatever is on your screen, is also playing on your costume. The mind boggles.

These same concepts can easily be applied to top hats, poi, LED coats, or whatever other project you've got on your workbench. The Feather M0 and Dotstar LEDs make it all possible.

Adafruit Parts

- Feather M0 Wifi
- Tactile on/off switch
- Lithium Ion battery
- about 2 m of 30 LED/meter Dotstars
- 26awg Silicone wire in various colors

Other Bits Needed

- A wig (I like [Arda Wigs](https://www.arda.com/) best but any wig will work)
- Wig clips()
- [Large tubular crin ribbon](https://www.adafruit.com/product/124) -- be sure you get the large size, the smaller won't fit the LED strips.
- 1/2" clear [heat shrink tubing](https://www.adafruit.com/product/101)

©Adafruit Industries
• Cable ties

Tools

• Hot glue gun
• Heat gun
• Soldering Iron & accessories
• Needle & thread

Setup & Testing

This is a fairly complex project with lots of steps. Each step is a fantastic opportunity for stuff to go wrong, so it’s really important to stop at every step and test to make sure the piece you’re working on is working for you.

There are two major parts to this guide: the software and computer-ish part and the hardware, soldering & building part. It’s a great idea to get your software up and running 100% before starting on the fairly complex hardware build for this project, and for that you’ll need to set up a testing strip.

Dotstar strips come with a connector pre-soldered onto the ends. For testing purposes I like to just cut the connector head off and use the pre-soldered wires to connect directly to my project.
Solder your strand of Dotstars:

- Dotstar Clock --> Feather 11
- Dotstar Data --> Feather 13
- Dotstar G --> Feather G
- Dotstar + --> Feather 3v

Find the Adafruit DotStar Strandtest code. You'll find this in the Adafruit Dotstar library for Arduino. Use the Library Manager to install this (Sketch→Include Library→Manage Libraries...), or if you’re using an older version of the Arduino IDE, it can be downloaded and installed manually:

Adafruit Dotstar Library
If you've used Feather M0 before, upload the Dotstar code and test to be sure the lights come on.

If you haven't used Feather before...intermission time! Read our Feather M0 Basic Proto guide to set this up:

**Adafruit Feather M0 Basic Proto — Arduino IDE Setup**

Newer versions of the Arduino IDE have been released since that guide...1.6.5 and 1.6.7 both seem to work pretty well with the M0 board.

Even if you have used the Feather M0 before, use the Arduino IDE Boards Manager to check that you’re using the latest Adafruit SAMD Boards support files. (For the Arduino Zero, install the Arduino SAMD Boards package.)

Before continuing, make sure you can compile and upload sketches to the board. Try the basic “Blink” example as a test.

Once you have "Blink" working, upload the Dotstar Strandtest code and be sure your LEDs light up.

**Software**

For WiFi support and for our code to use certain M0-specific features, download and install the Adafruit_WINC1500 library (for Feather M0 WiFi) or WiFi101 (for Arduino Zero w/WiFi Shield 101 — install via Arduino Library manager), plus the Adafruit Arduino Zero ASF Core Library and Adafruit_ZeroDMA:

- Download Adafruit_WINC1500 library from Github
- Download ASF Core Library from Github
- Download Adafruit_ZeroDMA library for Github

Library installation is a frequent stumbling block...if you need assistance, our All About Arduino Libraries guide spells it out in detail!
Then download and extract the ZIP file containing the code for this project:

Download Adafruit_Lightship from Github

In this archive are two folders:

The first, “Arduino,” contains the OPCserver sketch for the Feather board (or Arduino Zero).

The second, “Processing,” contains several Open Pixel Control client demos for use with the Processing programming language. These will run on your main computer (desktop or laptop).

The client/server nomenclature may seem odd...the OPC “server” runs on the tiny Feather board, while OPC “clients” are programs running on a larger and more capable system. All the color and animation decisions are made in the client applications...the server just passes these through to the LEDs.

OPC clients can be written in many programming languages, but we’ll use Processing (a derivative of Java) as it’s free, cross-platform (runs on Windows, Mac and Linux) and is focused on visual arts.

Processing Downloads Page ()

For now, we recommend downloading the version 2.2.1 release of Processing for your operating system. The 3.X series introduced some significant changes that aren’t always compatible with existing Processing code.

Processing looks a lot like the Arduino IDE (in fact, the Arduino IDE derived from the same code base). This can be confusing because Arduino sketches don’t work in Processing, nor vice versa. Make sure you’re loading sketches into the correct IDE for each.

Edit Arduino Sketch

In the Arduino IDE, open the “OPCserver” sketch. We’ll make some changes before uploading to the board, to configure for your particular hardware and network.
Toward the top of the code, the following lines are of interest. They’re not in one place, but all appear in the first 50 or so lines of the sketch:

```
#define ADAFRUIT_ATWINC
```

This tells the code to use the Adafruit_ATWINC library. If using an Arduino Zero w/ WiFi Shield 101 and Native USB comment this line out.

```
//#define Serial SerialUSB // Enable if using Arduino Zero 'Native USB' port
```

Commented out by default. Enable this line only if you’re using an Arduino Zero and are connected to the “Native USB” port (rather than the “Programming” port).

```
#define IP_TYPE IP_STATIC // IP_STATIC | IP_DYNAMIC | IP_BONJOUR
```

You’ll probably leave this line as-is, which tells the WiFi module to use a static IP address (rather than dynamically assigning an address from your WiFi router). A static address makes it easier for OPC clients to access the device, as it’s always in a known location.

This requires some knowledge of how your WiFi router doles out IP addresses. Most will have a numeric range of IP addresses to assign dynamically…for example, my WiFi router starts issuing dynamic IP addresses at 192.168.0.100 and above. I can assign fixed addresses below that (except for 0 and 1) to specific devices, as long as others aren’t using them.

```
char      *ssid = "NETWORK_NAME",   // WiFi credentials
*pass = "NETWORK_PASSWORD";
```

Replace these two strings with the login credentials for your wireless network.

```
IPAddress  ipaddr(192, 168, 0, 60); // Static IP address, if so configured
```

This is the static IP address to be assigned to the device on your wireless network. As mentioned above, this requires knowledge of your router’s DHCP policy. Most will start with 192.168.0.X, 192.168.1.X, 10.0.0.X or 10.1.1.X — but not all.

Whatever address you use, you’ll need this later for the Processing sketches.

A little further down in the code (around line 100) are these lines:
#define DOTSTAR_BLUEBYTE  0
#define DOTSTAR_GREENBYTE 1
#define DOTSTAR_REDBYTE   2

You probably don’t need to edit these. But later, if you find the test examples are producing the wrong colors, you may need to return here and edit these numbers. This is the “native color order” used by the DotStar LEDs…it’s changed at least once in different production runs of these devices. If your LEDs are from a different batch, you’ll need to swap or rearrange some of these numbers (they’ll always be the values 0, 1 and 2, only the order changes).

Upload

Make sure Adafruit Feather M0 (or Arduino Zero if using that board) is selected in the Tools → Board, then upload the code to the board.

If the code doesn’t compile (throws an error):

  - Confirm the right board type and USB port are selected in the Tools menu. Have you installed the correct files for this board using the Boards Manager?
  - Confirm the correct libraries are installed: Adafruit_ASFcore, Adafruit_ZeroDMA and Adafruit_WINC1500 (for Feather M0) or WiFi101 (for Arduino Zero + WiFi Shield 101).
  - There may be problems if both the WiFi101 and Adafruit_WINC1500 libraries are installed. If so, remove the WiFi101 library…the WiFi Shield 101 works fine with the WINC1500 library, you’ll just need to edit some pin numbers (this is commented in the code).
  - Check that you haven’t mangled the syntax on any of the lines edited above. For example…the IP address has commas (not periods) between each value.

If the code compiles but doesn’t upload:

  - If you built the circuit with the DPDT switch, this needs to be in the “RUN” position to upload code.
  - Try uploading again. Or tap reset once or twice on the board, then upload. The M0 boards are a new thing and can be a bit persnickety what with all the different operating systems and USB port types.

Now open the Arduino IDE Serial Monitor.
If you see nothing at all, that's actually a good sign. Or if you see a “Server listening” message, that's good too.

If you get an unending series of periods (...), the board isn’t connecting to your wireless network. Confirm that the network name, password and provided IP address are all valid.

If everything checks out, let's try the examples...

**OPC Strandtest**

Hello world! Launch the Processing (2.2.1) IDE and load the first of our example programs: OPCstrandtest.

At the top of the code are these two lines:

```java
OPC opc       = new OPC(this, "192.168.0.60", 7890);
int numPixels = 256; // Set this to actual strand length
```

The first line holds the address of your OPC server device. We previously configured that in the Arduino code. The format here is just a little different though...instead of four comma-delimited values (192, 168, 0, 60), here it’s a string with period separators "192.168.0.60". Edit the numbers to match the Arduino sketch.

Next line is the number of LEDs in your DotStar chain. This might be 64 for an 8x8 matrix, or 60 (or 144, etc.) for one meter of DotStar strip.

When you run this code (the top-left icon in the Processing window), after just a moment’s delay you should see the message “Connected to OPC server” and get a
“chaser” down the DotStar chain that cycles between red, green and blue for each pass. Press the ESC key to stop the program.

If the colors appear in the wrong order (not red, green, blue), you'll need to edit the Arduino sketch to match your particular DotStar hardware. This is explained near the bottom of the “Software” page.

If it runs slowly or stutters, especially with long LED runs, you may need to reduce the frame rate. Inside the setup() function you'll see a call to frameRate(). It takes a single argument, the number of frames per second. The default (if not specified) is 60 frames per second...that's usually fine with small projects, but with lots of LEDs you may need to dial it back, to perhaps 30 frames per second.

If you don't see the “Connected” message and don't get any LEDs: either the Server sketch on the Arduino can't connect to the wireless network, or the address at the top of the Processing sketch is incorrect.

Do not continue until you have the led “chaser” working, and it's cycling from red to green to blue.
Wiring

This wig has 6 Dotstar strips. Each strip's power and ground wires are wired in parallel (starfish pattern, or lots of little "spokes") and the Clock and Data wires are wired in sequence (serpentine pattern, or one long wire from start to finish).

This means that if one of your clock or data lines comes un-soldered, everything "downstream" of that bad solder joint will stop working. So be sure your soldering and gluing is rock-solid for this project!

- Dotstar Clock --> Feather 11
- Dotstar Data --> Feather 13
- Dotstar G --> Feather G
- Dotstar + --> Feather 3v
LED Strip Wiring

Prepare LED Strips

Cut 6 strands of dotstars -- 3 for each side. My strips have 8, 10, and 12 pixels respectively since I like a "layered" look on the finished wig. Cut carefully between the copper pads and leave the silicone sleeve intact.

Clock & Data Out

Start by finding the "out" ends of all the strands and line them up so they're all correctly faced. Solder a long (15-20") clock and data wire onto the pins at this end of 5 of the strips (you don't need clock-out or data-out on the last strip).
Slide the strips out of their silicone housing, then slide them back in, with the wires trailing along the back of the strip. The idea here is to get the wires to feed neatly back through the silicone strip so they won't ever get tugged or bumped.

Clock & Data In

Once you've got the sleeves back on, expose the "in" end of the strip.

On strip #1, solder a long clock wire and data wire to the "in" pads. These will later connect to the Feather.
Turn this strip over and find the wires poking out the back of its silicone sleeve. Trim to a good length and then solder these "out" wires from strip #1 to the "in" pads on strip #2.

Find strip #2's "out" wires and solder those to the "in" pads on strip #3. You now have a daisy chain of 3 LED strips all connected to each other.

Repeat this sequence with the other 3 strips. Note: Do not daisy-chain all 6 strips together at this point! That will make it impossible to thread the wires into the wig for attachment. We'll connect the clock and data lines from strip #3 to strip #4 together later.

For now, take a sharpie and mark the "out" clock and data wires on strip #3 and the "in" wires on strip #4, so you'll know which is which later on.

Power & Ground Wires

Cut 6 short (3") red and black wires and solder onto the + and - pads of each strip.

Cut a 6" power wire and twist it together with the power wire from strip #1. Then, twist together the power wires from strips #2 and #3. Splice all these wires together and cover with heat shrink. You should end up with one red wire that splits 3 ways and connects to each strip.
Repeat with the ground wires, and do the same for strips #4, 5 and 6.

At this point you'll want to test all your connections and make sure everything is solid and all the strips work. Hook each set of 3 strips up to a gemma or flora running the Dotstar Strandtest code and be really sure everything works perfectly. In the next step we'll be coating these connections in glue and heat shrink and they'll be very hard to fix! So be sure everything works before proceeding.

Wig Assembly

Get out your heat gun and plug in your hot glue gun. You'll need both tools ready in order to create a good, solid, rugged seal that will enforce your wiring and keep the strands safe.

Cut a piece of clear 1/2" heat shrink and slip it over the bottom of each strand. Before you shrink it down, pump some hot glue in -- just enough to cover the connections and fill the end of the silicone sleeve.

Hit it with your heat gun to shrink down the heat shrink, before the glue cools fully. If the glue squishes out all over the place, you've done it right. Let it cool completely and then trim off any excess glue.
Do the same for the top of each strip (you'll need to slide the heat shrink on from the bottom).

Give your wires some gentle tugs. You'll be supporting the weight of each strand on the wires, so be really sure each one is secure and solid.
Measure out your tubular crin and cut it to the length you want.

Slide one piece of crin over each LED strip and secure it with a cable tie. Then, using more ribbon or cable ties, secure each set of 3 strips sturdily together. Add in any leftover crin ribbon to create a filled-out cyber-dred.

Turn the ends of all the crin pieces inside out so they don’t unravel.
Insert all the wires from each cyber-dred through the cap of your wig. One dred will have 6 wires and the other will have 4. Make sure you're happy with the placement, and add a whole bunch more zip ties to keep the dreds in place.
Final Wiring
Power Connector

Plug your male JST connector into the female plug coming from your battery and take careful note of which side is power and which is ground.

Slide a small piece of heat shrink onto one of the power and one of the ground wires you just attached to your LED strand. Carefully solder each of these wires onto the JST connector and cover with the heat shrink.

Now we're going to add hardcore strain relief to this connector. It's the piece that takes the most tugging and wear and tear, so we want to be sure it won't break while you're changing batteries (in the dark, in the middle of the night, most likely).
Slide a large 1/2" or 3/8" piece of heat shrink over the connector, leaving the front open but covering the back and the connections you just made completely. Don't heat it yet!

Take a hot glue gun and fill the back side of the heat shrink with hot glue. Then, while your glue is still liquid, hit the whole assembly with a heat gun to shrink the heat shrink in a controlled manner. Be careful not to get glue inside the connector itself.
Solder your on/off switch inline with the red wire coming from your power connector.

Then, create your power harness: Twist together the red wires from each cyber-dread. Twist another red wire to the one coming from your power switch. Connect all 4 of these wires together and cover with heat shrink.

Repeat with the ground wires.
Connect the "out" wires from strip #3 to the "in" wires for strip #4. Aren't you glad you marked these earlier?

Now you have one free power wire, one ground wire, one data wire and one clock wire. Connect these to your Feather as shown. Clock goes to 11 and data goes to 13.
Gently set your wig down or place it on a wig head and plug a battery in. Make sure your feather powers up! It's time for a test run.
First, upload the dotstar Strandtest code to your Feather. All 6 strands should light up in sequence. Be sure you can run Strandtest before proceeding any further.

Next, upload the OPCserver code to your Feather in the Arduino environment, making sure you've edited the sketch with your local wifi network IP address and password.

Launch Processing and open the OPCstrandtest code. Edit these lines to reflect your IP address and strand length.

```java
OPC opc = new OPC(this, "192.168.0.101", 7890); //your IP adress here
int numPixels = 60; // Set this to actual strand length
```

Press "play". If all goes according to plan, you'll see the strandtest code run on your wig. Congratulations! Let's secure the components and then it's time for the fun stuff.
If your strandtest code doesn’t run, don’t despair. This is pretty complicated. Some things to check:

- Is your wig powered up with a battery and the switch turned on?
- Are your IP address and password correct in the OPCserver Arduino sketch?
- Is the IP address correct in the OPCstrandtest Processing sketch?
- Is your computer on the same Wifi network as the OPCserver sketch?
- Are all the requisite libraries installed?

More troubleshooting tips over at [Phil Burgess’ Lightship guide](https://example.com).

Finishing

Sew your Feather to your wig using the mounting holes. It works well to sandwich any extra wires behind the feather, but be sure your USB port is accessible.

Add a battery pocket made from a scrap of fabric or netting.
Sew wig clips into the front and sides of the wig. The added weight of the electronics and dreds may make it slide off your head otherwise. Wig clips are beautiful things.

Since I dance pretty hard in my wig, I also added a little scrap of oilcloth over the Feather, so it's not in any danger of shorting out if I sweat or wear the wig with damp hair.
Finish up by adding some ribbons or flowers or whatever your imagination desires to hide the zip ties at the top.

Use It

The OPC folder has 4 files in it. One is OPCstrandtest, which you are already familiar with.

Check out Phil's Lightship guide () for descriptions of how to use OPCpianoroll and OPCvideo.

New to this guide is the OPCscreencap sketch. This sketch will capture whatever is playing on your screen in realtime and send it out to your wig over wifi.
Open OPCscreencap and update the code to reflect your device's IP address. Also, change the arrayWidth and arrayHeight numbers to match your LED strands -- I used 6 for width since we have 6 strands, and 12 for height since my longest strip is 12 pixels long.

```
OPC opc       = new OPC(this, "192.168.0.101", 7890);
int arrayWidth  = 6,   //change these numbers to reflect your LED strands
arrayHeight = 12,
pixelSize   = 30,
numPixels   = arrayWidth * arrayHeight,
screenNum   = 0  ;  //if you have multiple computer monitors, try changing this to 1
```

Press "Play" in Processing. You'll see a pixelated version of your screen that's 6x12. Move some windows around on your screen and see if the lights in the wig change along with you.

To see the power of this app, pull up a music visualizer program (iTunes has a good native one, or head over to Electric Sheep () on YouTube). Make the video full screen while your Processing script is running, and Be Amazed.