adafruit learning system

1,500 NeoPixel LED Curtain with Raspberry Pi and Fadecandy

Created by Phillip Burgess

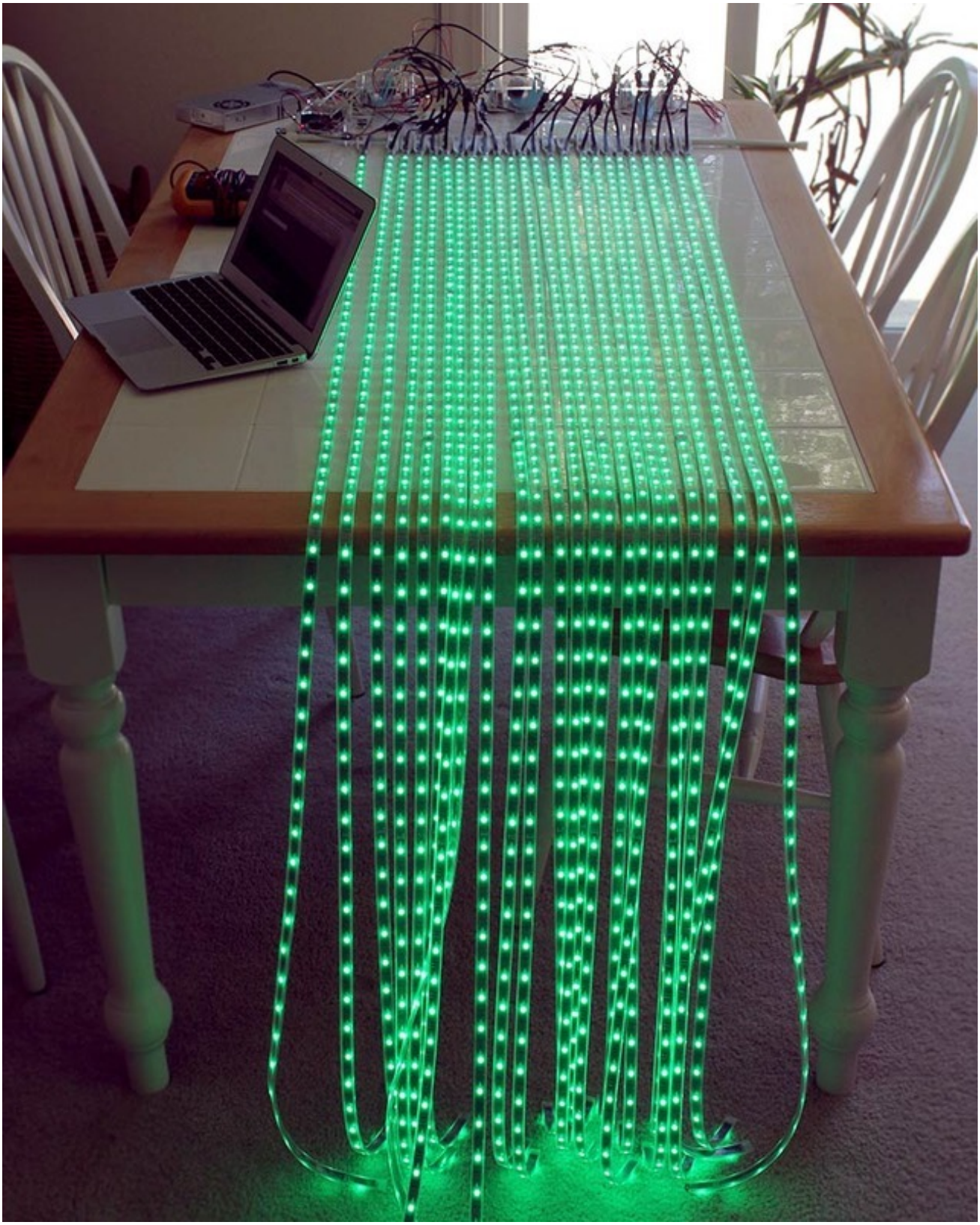
Last updated on 2014-09-15 03:30:13 AM EDT

Guide Contents

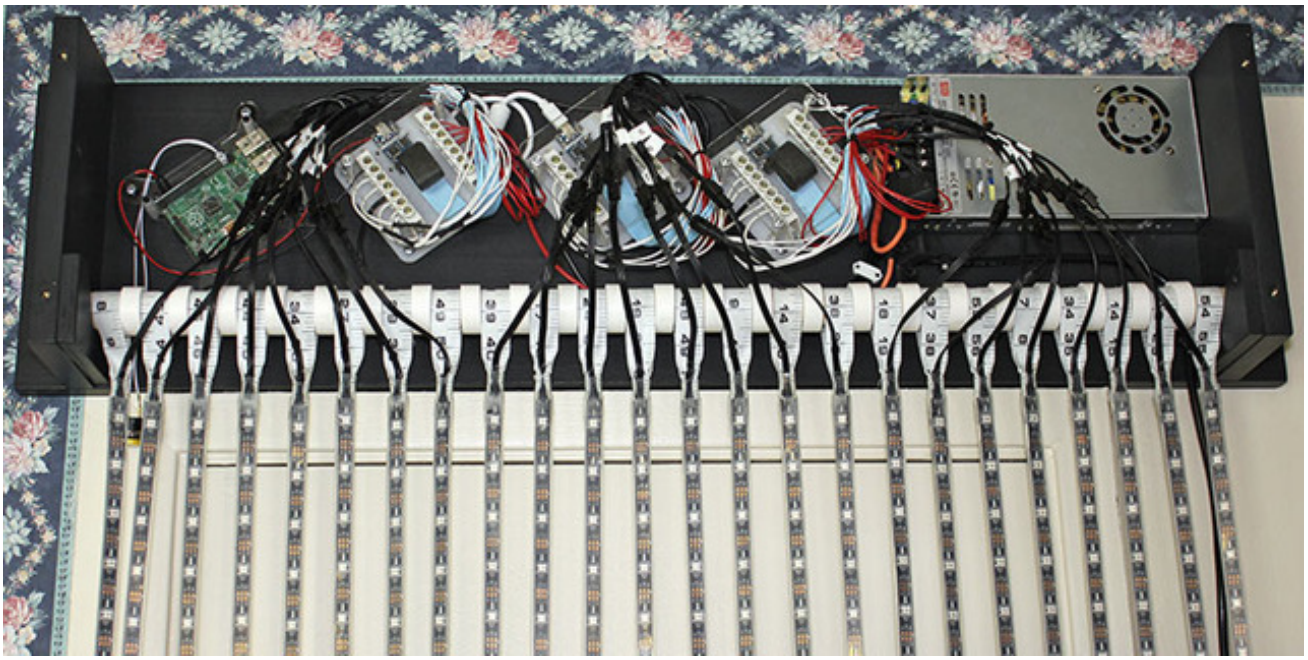
Overview

What your new apartment really needs is an upgrade to that beaded curtain - why not liven it up with a *massive* ~1,500-LED* NeoPixel curtain? NeoPixels + Raspberry Pi + Fadecandy == AWESOME!

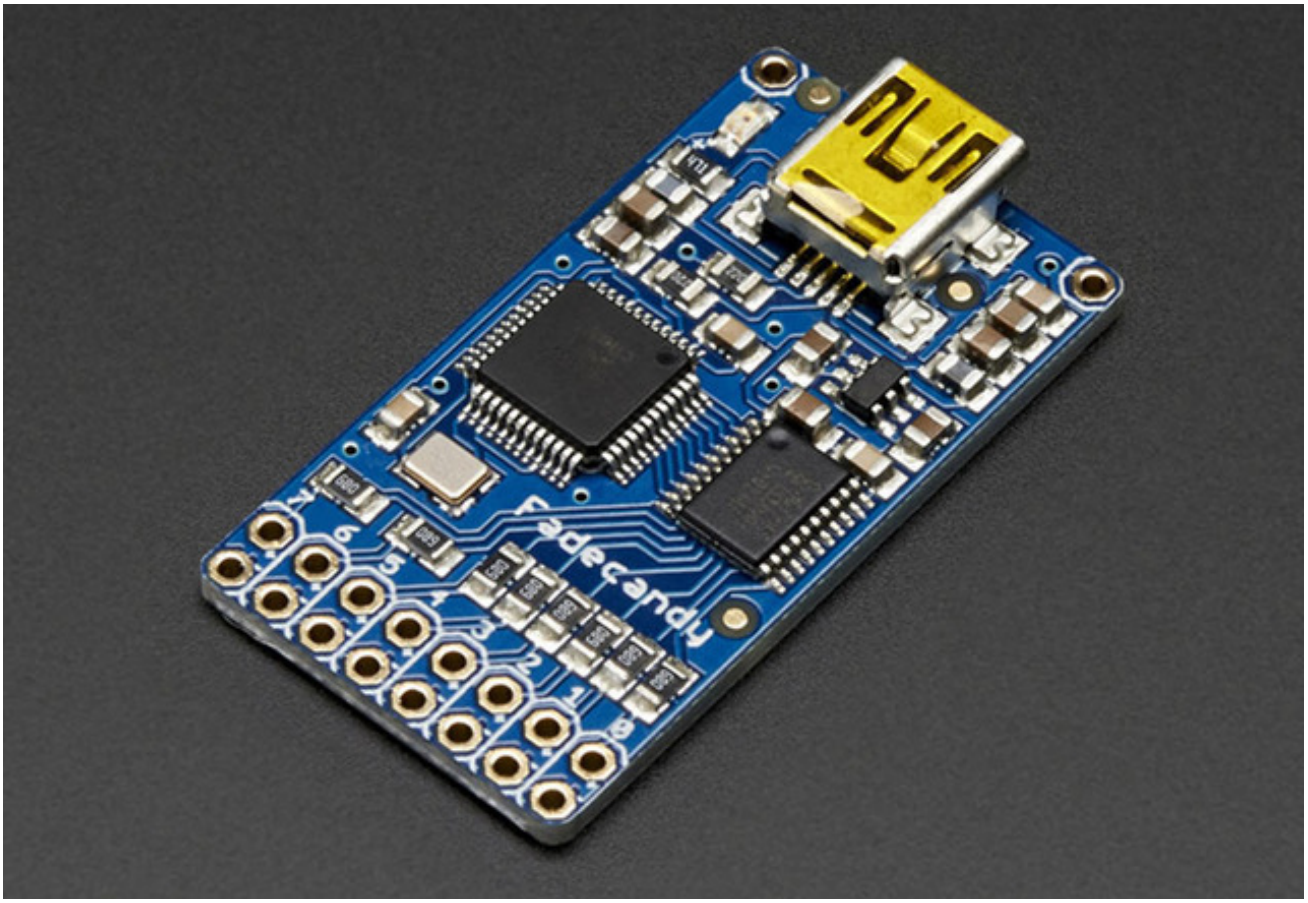
* *(technically, its 1440 LEDs but you can make your curtain any # you want!)*



This project will show you how you too can build this massive and enjoyable flexible display. If you're interested in any other kind of large-scale NeoPixel project, this guide will help with all your detailed questions with Phil Burgess' hard-earned tips and techniques.



The curtain is powered by a massive 5V brick supply, a Raspberry Pi Model B+ and three **Fadecandy** NeoPixel driver boards! You can display any kind of images, video, or effects thanks to the wireless server system. Data is beamed from your desktop to the curtain over WiFi to the Pi server.



Fadecandy is a powerful NeoPixel driver designed by Micah from Scanlime (http://adafru.it/cWk). Calling Fadecandy "a NeoPixel driver" is only half the picture. It's a combination of hardware *and software* working together to create **subtle, expressive LED art,** far more nuanced than the usual frenetic blinky rainbow fade.

Fadecandy is:

- **Scalable.** Each board drives up to 512 NeoPixels (8 chains of 64 pixels); add more boards for more pixels. Massive installations become practical.
- **Open source** hardware, software and protocols. Program in your "native tongue"… Processing, C, Python…
- **USB-connected**, offloading the low-level task of NeoPixel communication; the host computer is free to dwell on **art and animation**, not shuffling and timing bits. A **client/server architecture** can further spread the task on a network — one system synthesizing visuals, another working with the USB devices.
- Designed with **visual quality** in mind. Temporal dithering provides smoother colors. Gamma and color balance can be finely adjusted.

We'll demonstrate Fadecandy's *flexibility* rather literally, with a hanging "soft" NeoPixel array containing nearly 1,500 pixels (1,440, to be specific). It's programmed using the Processing environment on our favorite laptop on a WiFi network, with a Raspberry Pi computer as a go-between.

Here Be Dragons

Rawr. This is a "Skill Level 5" project* — one requiring a heap of tools and techniques that only come with lots of prior making.

It's not quite a step-by-step guide. There remains a lot of *hand-waving*…gaps that can only be filled by your own improvised methods. Adafruit doesn't offer all of the parts shown; you need your own resources. And it's expensive. And there's risk.

Our goal was a dramatic NeoPixel/Fadecandy showcase project. Maybe you won't build one of these *exactly.* It's here to fuel your own ideas!

* *We don't really have a numeric skill level system…it's a nod to the Estes model rockets of my youth. Attempting a rocket project far beyond one's experience usually ended in heartbreak. If this looks too daunting, we have lots of other NeoPixel projects that are much more approachable!*

Some of the *elsewhere* components include:

- Scary bigass 5V DC power supply
- Heavy-gauge copper wire
- Fuse holders, 25A fuses

- Heat-shrink tubing: 1/2" clear, plus several smaller sizes
- Lumber, fasteners, adhesives, acrylic

Tools include:

- Quality soldering iron
- Multimeter (w/continuity test mode)
- Heat gun
- Woodworking tools (e.g. bandsaw, sander)
- Safety glasses, fire extinguisher, common sense

Sources for parts have included:

- Online electronic/industrial distributors (Mouser, Jameco, etc.)
- Craft store
- Auto parts store
- Well-stocked independent hardware store
- Electronic/industrial surplus store

These lists (and the featured products at right) don't cover everything. Read through, I may have missed some items, or you might have ideas for substitutes of your own.

Gross Planning

Our idea is a futuristic version of a beaded curtain, using vertical strands of LEDs. Hung from a rod, it might fill a doorway or act as a room divider. In order to be similarly functional — allowing a person (or the random cat) to walk through — there are some physical constraints on the design. Free-hanging strands like this requires they have connections only at the top; bottom connections or zig-zag wiring would snag passers-by.

Planning this out requires some simple “napkin calculations”…

A typical interior door measures about **30 inches wide by 80 inches tall (76 cm x 2 m)**, so we’ll aim for something close to that.

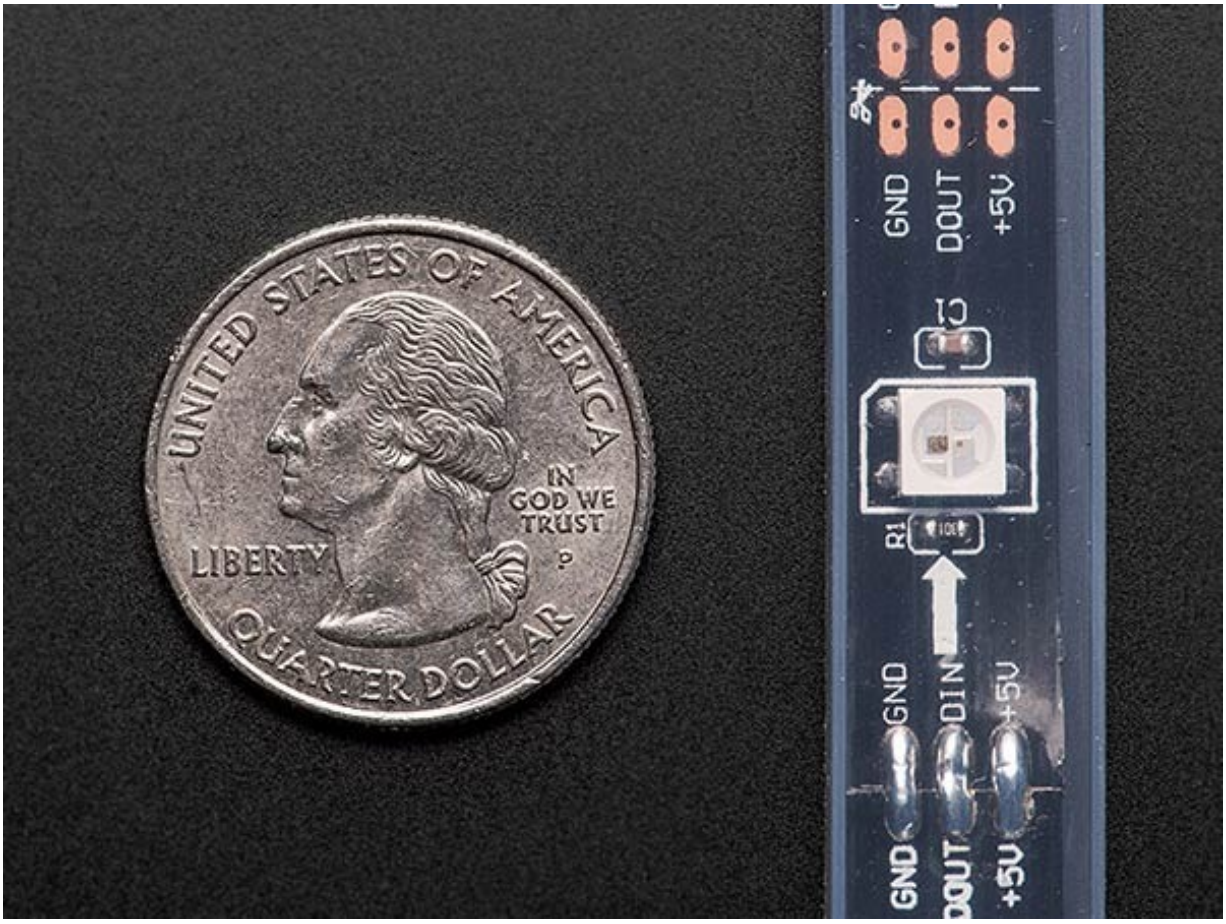
Goal: vertical LED strands to fill a 30 by 80 inch doorway, connections at top only.

Then there are technical factors to consider. A **single Fadecandy board** is designed to control up to 512 pixels: **8 strands of 64 pixels** each, maximum. With our data connections all residing at the top of the curtain, this sets an upper limit to the number of LEDs we can hang vertically: **64**.

We’d like to use flexible NeoPixel strips, they’re amazingly handy. These are sold in various pixel “densities”: **30, 60 or 144 LEDs** per meter.

Given some of our constraints…the 64 pixel limit and the 2 meter door height…we can quickly determine the higher densities won’t work for this application. So…

Established: we’ll use 30 LED/meter NeoPixel strips.



The door's about 2 meters tall, and NeoPixel strips are sold in 1-meter increments (5 meters on a reel). So…

Established: each vertical strip will be 2 meters long, containing 60 pixels.

This is quite convenient, it avoids lots of little fractional bits of strip. At worst a few strips will require joining two 1-meter sections. The 2m length comes close to using the full 64 pixel capability of the Fadecandy board. *(This 64 pixel limit only applies because of our "single connection at the top" constraint. If you're building something to hang on a wall, connections can be made anywhere, and much larger and more freeform layouts become possible.)*

At 30 LEDs/meter, there's about **33 millimeters spacing** between pixels down the strip. To create a pleasing and proportional grid arrangement, we should try to match that spacing (or come close) on the horizontal axis. 76 cm door width (760 mm) ÷ 33 mm spacing = 23 strips.

Add one to avoid a ***fencepost error (http://adafru.it/aJ9)***. **24 strips** total. This is perfect! Each Fadecandy board can control up to 8 strips in parallel. **8 × 3 = 24**. And the X/Y grid spacing will be very nearly uniform.

Established: there will be 24 vertical strips, 2 meters each (48 meters total),

controlled by 3 Fadecandy boards. Math!

So the finished curtain will be comprised of **48 meters** of NeoPixels. There's one more number to determine, and it *can't* be derived from simple math: **what's your tolerance for risk?** How much, if any, spare strip should be ordered? And how much can you really budget for?

Accidents happen.

Sometimes wires get crossed, NeoPixels get fried. Occasionally a reel escapes the factory with a bad pixel or two. Defective strips can be replaced…but can you afford the downtime? Should you have some spare materials on hand, ready to swap out? ***Scratch monkeys?*** (http://adafru.it/dMs)

Since the 30 LED/m strip comes on 5-meter reels, I started with **50 meters** total (10 reels) …leaving just enough for *one* spare 2m strip. If a strip in my curtain should fail, an extra is immediately on-hand to replace it…but that's it, just the one. That's cutting it close. Though I've always had good luck on NeoPixel projects, my personal safety buffer would perfer two extra reels…a 25% surplus…and maybe even a spare Fadecandy board. **Your personal buffer might be different**, and there's no hard number that I can provide here. Stuff costs money, and it's uncomfortable looking at idle backup hardware just sitting on a shelf.

Find your personal balance zone between cost and readiness.

Data Topology

One of the stellar features of Fadecandy is that the software is **cross-platform**, with versions for Windows, Mac and Linux.

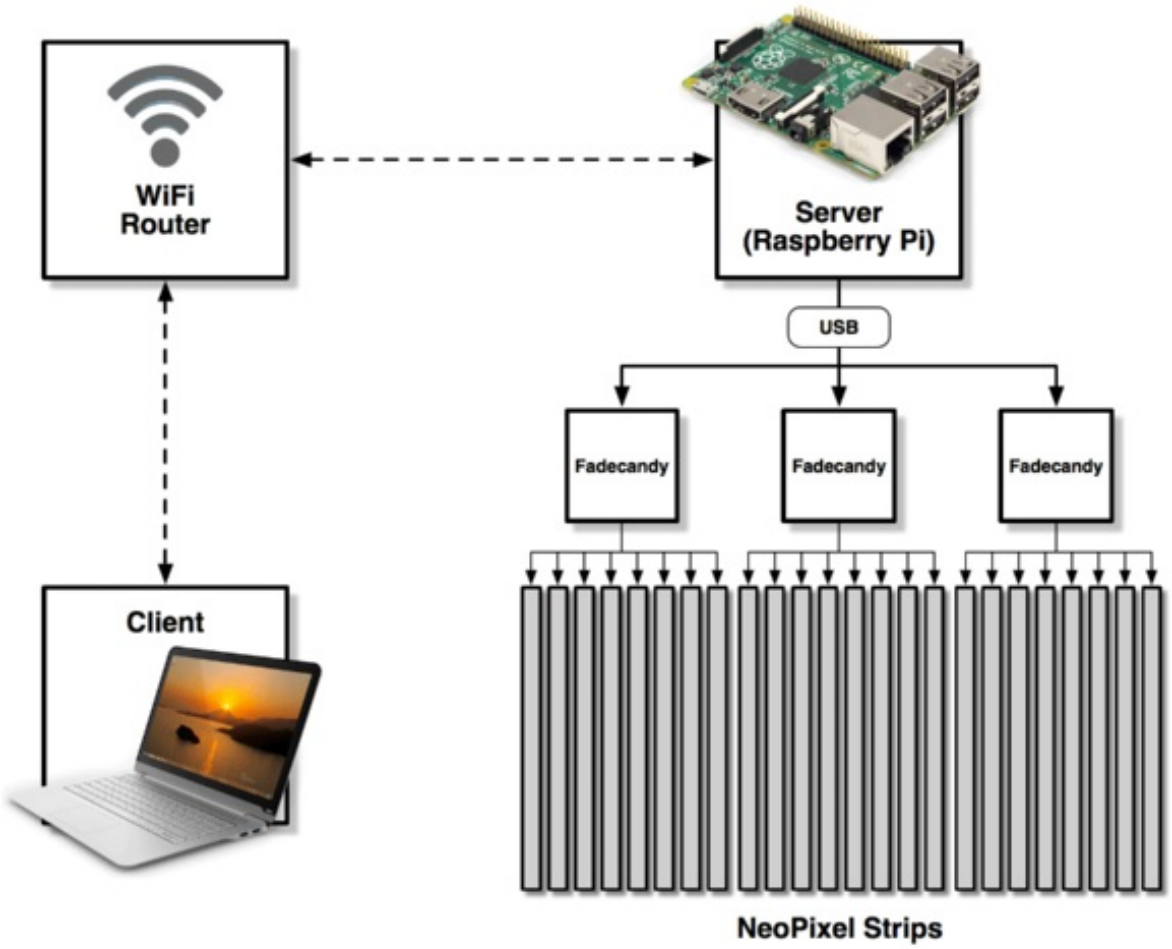
It also uses a **client/server architecture,** meaning the tasks of *rendering animation* and *communicating with the Fadecandy USB boards* can take place on *separate machines on a network* (or on the same system, if you'd prefer).

With tiny and super-affordable Linux systems like the **Raspberry Pi,** it's not unthinkable to *dedicate a whole machine to the server task.* Linked to a wireless network, it accepts connections from other systems and forwards data to the Fadecandy USB hardware.

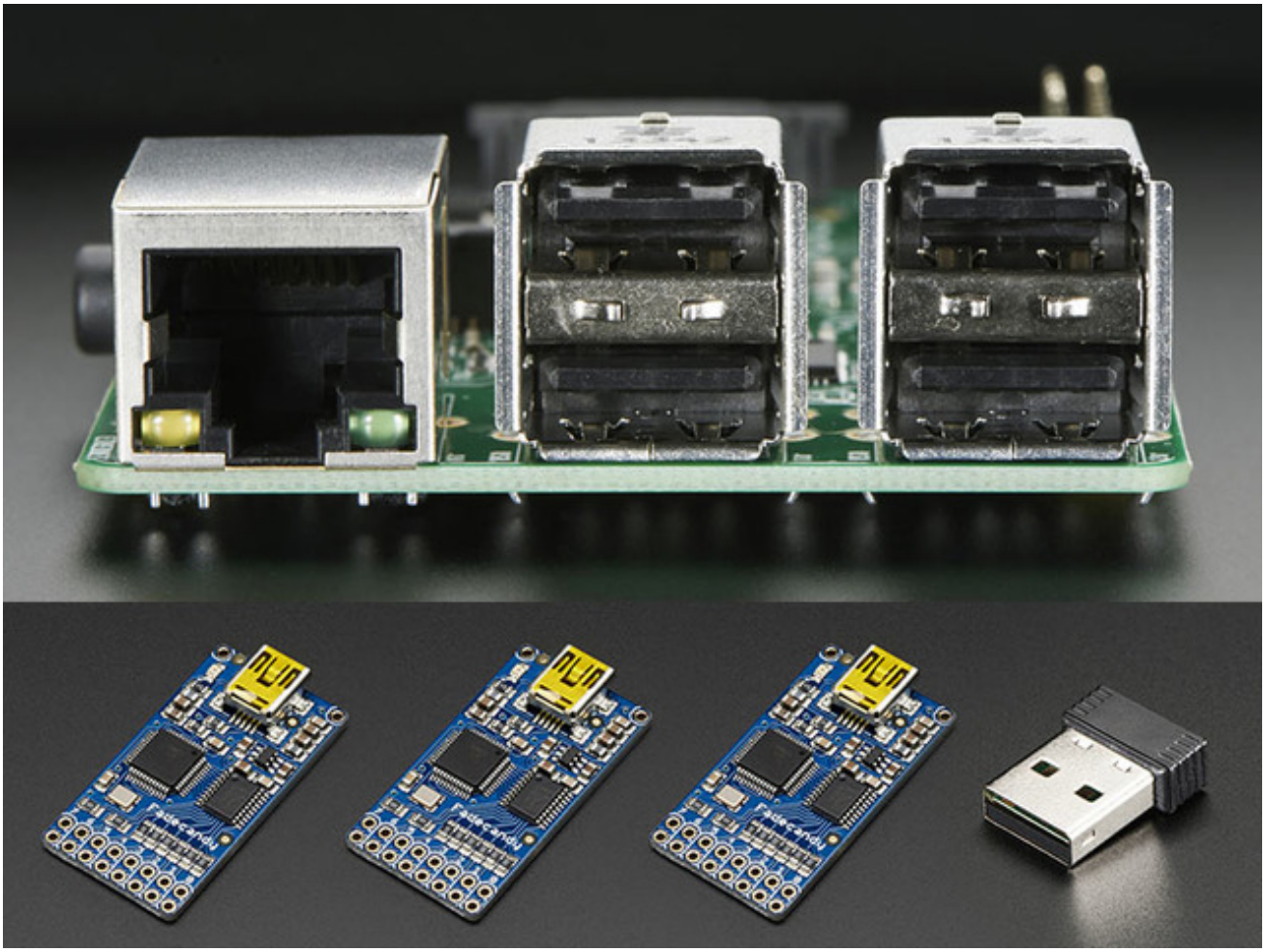
This all means…

1. The server-side hardware can be assembled into a tidy package with just a *single power cord* protruding…no visible rat's nest of wires…you don't even need to run a USB or Ethernet cable to the outside.
2. You can kick back with your favorite laptop and write animation code without being tethered to the Fadecandy hardware. How cool is that?

Disregarding power for the moment, here's how the parts communicate:



The new Raspberry Pi **Model B+** seems tailor-made for this. On the prior page we determined that **three Fadecandy USB boards** will meet our needs. The Model B+ has **four USB ports**...sufficient for the Fadecandy boards plus a **USB WiFi adapter**. Once set up, the Raspberry Pi will operate "headless," with no monitor or keyboard attached...a USB hub is not required, further simplifying the cabling.



Power Topology

This is the area where most folks run into trouble. **Power conversion and distribution on a large LED installation is not a thing to be trifled with**…done wrong it can lead to **component failure, fire and/or injury.** Really!

> Don't mess around.

Think about the trunk and branches of a tree: hefty at the core, progressively thinner toward the extremities. Why is that? Well yes, of course, the tree needs to hold itself up. But also…*you can't squeeze a whole tree's worth of water and nutrients through a narrow branch.*

Power distribution is exactly like that.

Wire has resistance. Thicker wires have less resistance; they can transport more current safely. **The rest is lost as heat.** In extreme cases, a *lot* of heat…enough to start a fire.

Google around for "wire gauge ampacity." You'll turn up charts with recommended limits for various gauges of wire. Stick within these bounds and you're safe. A couple points to keep in mind:

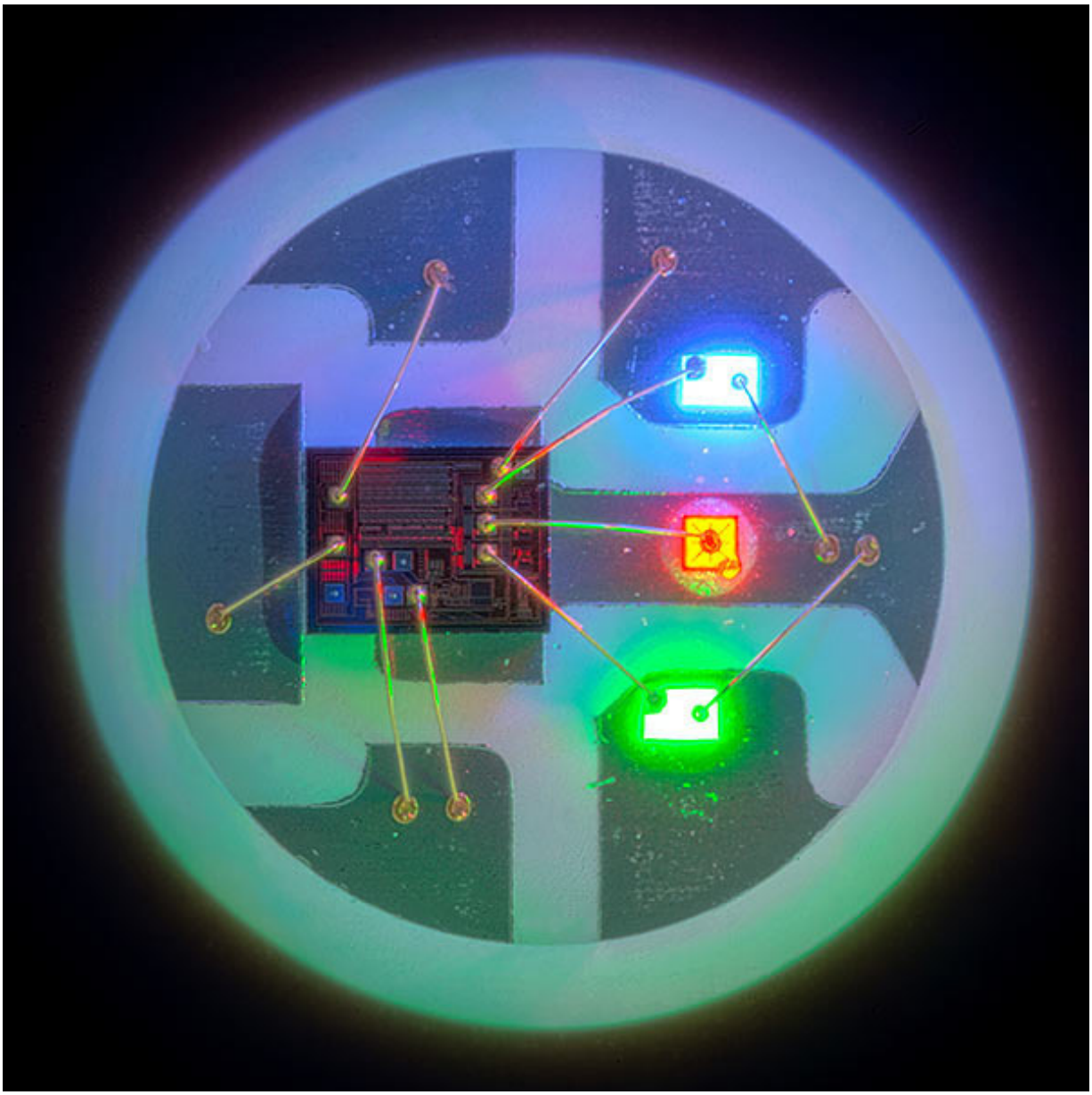
- Most charts show different limits for "free air" vs "enclosed" wire (or "chassis" vs "power distribution"). The former is for short runs where ambient air provides thermal relief; latter is generally for long runs through conduit, with a lower safe threshold. For a project like this one, we can consider it chassis wiring…but that's no reason to push the limits. Allow for some overhead!
- The charts are almost always for solid-core wire. Stranded wire is nice for its flexibility, but its *equivalent cross section* is lower — the gaps between strands mean there's less current-carrying copper, so you need to scale back the numbers a bit.

Being overly dramatic, aren't you? LEDs are super efficient, they hardly use any power!
Indeed they are. It's not a matter of efficiency though, but *scale...*

ALL THE LEDS!

A single LED uses so little power, it can run off a tiny coin cell battery for hours, even *days!* (http://adafru.it/dMu)

One NeoPixel contains three LEDs: one each for red, green and blue, plus a tiny embedded controller chip:



At full brightness, one NeoPixel (producting enough light that it kind of hurts to look directly at it) draws about **60 milliamps** of current at 5 Volts. That's a modest amount of power.

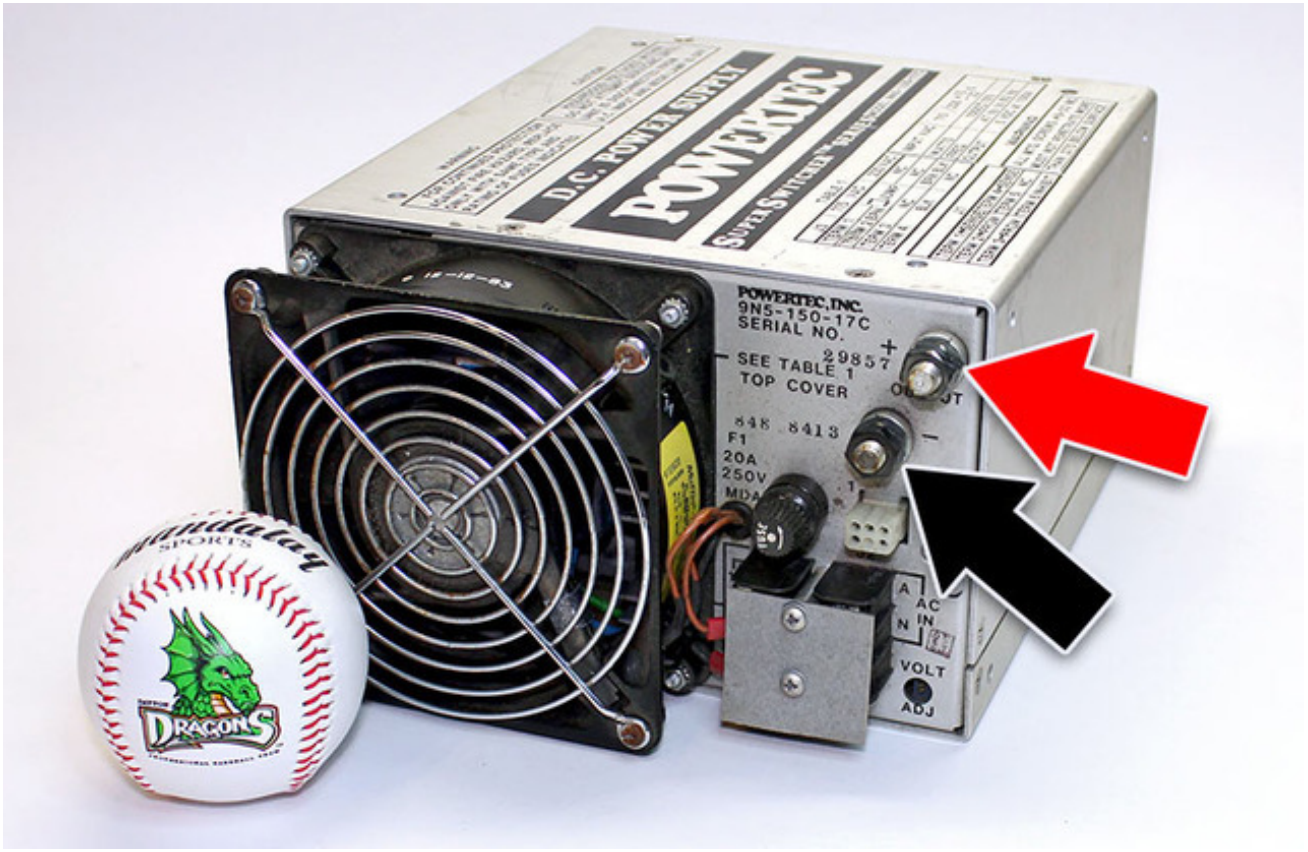
Efficient!

Our 2-meter strips each hold **60 NeoPixels.** 60 NeoPixels × 60 milliamps = 3,600 milliamps…**3.6 Amps!** Suddenly, that's a real amount of current. Yet we haven't lost efficiency, *we're just dealing with a whole crapton of LEDs!*

Multiply that by the number of strips: 3.6A × 24 strips = **86.4 Amps** at 5 Volts. By comparison, the power brick for a phone charger might provide 1-2 Amps, tops. 86.4 Amps is a *lot.* If it makes you feel better, run around screaming it like Doc Brown's *"1.21 Jiggowatts!"*

Referring back to those ampacity charts, you'll see this would require cables with solid copper as thick as a pencil. **This is where people run into trouble.** The usual electronics project hookup wire and "wall wart" power supply aren't gonna cut it…**a project of this scope requires a change of materials and techniques.** Some of these parts we don't offer at Adafruit; it's beyond the scope of the hobbyist…you'll need to turn to industrial suppliers.

Look at this thing:



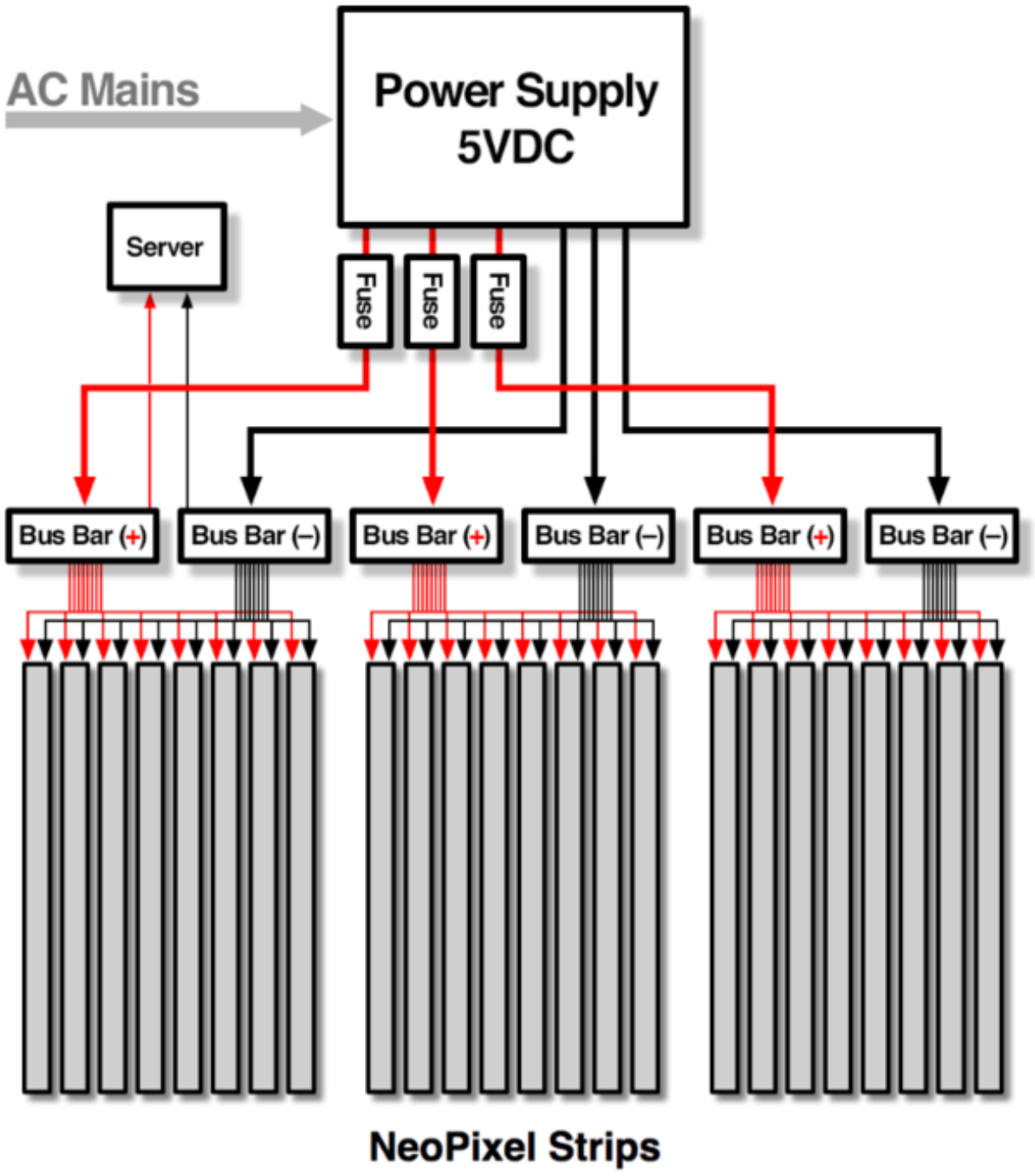
This is a 5 Volt, 150 Amp DC power supply bought at a local hamfest (the baseball is just there to provide some sense of scale). See those two bolts? *Those are the DC output!* Bolts! This thing requires cabling like the battery in your car.

It's okay to use an oversize power supply — one with a higher amperage rating than our LEDs need — the circuit will pull only what it needs, and the supply will run a bit cooler. Just don't use something with a higher *voltage.* **5 Volts** is it!

If you're a surplus scrounger, be certain what you're getting is a **DC** supply. This is easily overlooked! AC will kill your pixels.

Shopping around sites like Mouser and PowerGate Express, I ultimately settled on this supply:

Instead of bolts, it features **three pairs** of **spade terminals** branching directly off the power supply. Since each pair only needs to handle 1/3 the current, this allows for more conventional wiring (albeit still heavy gauge). The three-way split also correlates nicely with the three Fadecandy boards, providing some organization to the system: for each group, data and power branch out to 8 NeoPixel strips. Also, each set could be separately fused — an electrical short in one area won't bring down the whole system:



The Raspberry Pi taps off one set of bus bars. That's fine…it uses so little power as to be negligible.

A Chain is as Strong as its Weakest Link…

The extra per-section fuses add a margin of safety…but, due to haste, also created an unexpected bottleneck that would determine the scale of the power system.

Not wanting to wait around for special-order parts, I found some nice inline fuse holders at the local auto parts store. These came pre-molded around **12 gauge** stranded wire pigtails…which had a recommended max rating of **22 Amps**. 3 × 22 = **66 Amps**, about 25% short of our desired 86.4 Amps.

I could have done something stupid at this point. I could have crossed my fingers and "redlined" the system, or skipped the fuses altogether. Or I could be sensible and delay the project to locate some 10-gauge fuse holders. Or…the path ultimately chosen…I simply **limit the maximum brightness of the LEDs to 70%** in code, so they never exceed 60 Amps total. The Fadecandy software has a setting to handle this!

This is not necessarily a bad thing.

1,440 NeoPixels at full throttle will knock you on your ass. 1,440 NeoPixels at 70% brightness will still knock you on your ass. Honestly, we're NOT left wanting for photons. Scaling the system down to 60 Amps pays dividends: there are more power supply options at this size (and at lower cost), and 12 gauge (vs 10 gauge) wire is more manageable and saves money. (No point using 10 gauge wire if the fuse holders are 12 gauge…we can't "get back" that current…it truly is a matter of the weakest link.)

The power supply in the photo above is a Mean Well RSP-320-5, rated for 5 Volts at 60 Amps. I chose this one because it's slim and has the multiple spade connectors and a decent rep…but there are many options from many manufacturers that'll do the job just as well. (I also liked that the spade connectors were recessed…safer that way, unlike the one with bolts sticking out, just waiting for a dropped wrench.)

Could I instead use a whole bunch of 5V power bricks working together?
Could I drive framing nails with a whole bunch of upholstery tack hammers working together? It's a matter of *the right tool for the job.* Large DC supplies are purpose-built for this kind of load, providing consistent voltage across the system. They're not *that* much more expensive.

Besides, where would you plug in all those bricks? It would look *terrible!*

How about an ATX power supply?
That's a fine hack for medium-sized projects…ATX supplies can be found as free scrap sometimes, and it just takes a single wire jumper to turn them on. They're inadequate for really large projects like our full NeoPixel curtain though…even the beefiest ATX supplies top out around 40A total on the 5V lines. And the distribution is rarely even…you may have six Molex connectors on a single cable, two on another…it doesn't balance out, and you don't know the safe limits on each wire.

Is this cheap power supply on eBay any good?
Speaking personally, I would never plug in an imported no-name thing that I can't pick up and carry the fire outside. I *might* consider a surplus or secondhand unit if it's a reputable

brand normally carried by a distributor like Jameco, Mouser, etc. (like that Powertec hamfest find) and in good condition.

Whatever you decide, I do strongly recommend buying it domestically. I've ordered big-ticket items from overseas and had entirely different items arrive…and then return shipping for a refund would cost more than the original price. Buying closer to home costs more, but you won't get stung if an exchange is needed.

What if the software brightness throttle fails?
Then we'll know for sure whether the fuses work!

This power supply says it can provide X Amps continuously…but also X+Y for 1 minute or X+Z for 5 seconds. Since my LEDs won't ALWAYS be full white, could I use a smaller supply this way?
Two good reasons why this is a **bad idea:**

1. When a power supply is pushed to its limits (e.g. LEDs exceeding the continuous-rated current), even briefly, the output voltage often sags. We'll be powering our Fadecandy server off this same supply…and with a power brownout, the computer may lock up. When this happens, the *NeoPixels retain their last color*…which we've already established is more than the power supply can sustain. *Brrzap, fizz, pop.*
2. When engineers design a bridge, they *don't* just do the math based on trucks filling the span bumper-to-bumper…they take that and perhaps *double it,* even though it's physically impossible. This is the *engineering overhead.* Nature excels at throwing curveballs, and systems are often put to unexpected demands. When success *depends* on that "X+Y for 1 minute," you're not just leaving yourself no overhead…*you're purposefully dipping into the red every time!*

Either get a bigger power supply that can sustain the peak estimated LED current, or implement the software brightness throttle (demonstrated on the Fadecandy Server Setup page). If I wanted to be *really* good about it, I'd dial back the brightness a bit further.

"With Great Power Comes Great Responsibility."

To reiterate the opening message, please be *super extra careful* around this stuff.

Ever had a component on a breadboard explode? One little capacitor or an LED or something? It throws shrapnel at your face and *it hurts*…and that's just a tiny amount of power in one tiny part!

We're not in Kansas anymore. Dropping a wrench or a screwdriver across the terminals of a high-current power supply can spot-weld the tool in place. The arc can burn or even blind you. Sparks can cause fires. Please:

- Avoid working on a live circuit. Unplug power and let any charge bleed off before poking around. (Sometimes you have to, obviously…metering voltages and such…be careful.)
- As you build up a system, test subassemblies; don't throw the switch on Las Vegas all at once.
- Never bypass safety devices such as fuses, interlocks, covers or terminal partitions.
- Cover power terminals. Insulate wire splices. Nothing conductive should protrude.
- Remove electrically conductive jewelry, including rings, watches and necklaces.
- Ampacity is all about cross-sections and contact areas. Don't skimp on wire gauges. Don't file down a terminal that doesn't fit…get the right terminal.
- Work methodically, check everything. Twice. Even the "obvious."

Bigger?

It's possible. You'll need a USB hub, then add one extra Fadecandy board per 512 pixels. Also a bigger power supply, proportionally larger cables and bus bars and an extra helping of common sense. Very industrial stuff…suppliers for large boats and RVs *might* have suitable components…such vehicles also rely on low-voltage, high current power systems.

Personally, even the 60A project scares the crap out of me.

Spotted this terrifying bus bar in an electronics/industrial surplus shop. Notice at this scale how everything's bolted. This might fan out to smaller bus bars, and eventually to conventional wiring at the extremities.

At some point (around 250A on the DC side, I estimate) you run up against the limits of what a standard house or office circuit can safely provide…to push it further is to blow a circuit breaker. Stop and think. When one's work approaches such a threshold that you're

considering *rewiring a building,* you really have to ask yourself: **is this *art,* or just MOAR LEDS?**

Test NeoPixel Reels

Testing all of the components before any serious work commences can save heartache later. Though the testing process can get a bit tedious, it's so much easier to exchange defective parts now when they're not wired into something!

You'll need an **Arduino** microcontroller board (any model should do — Uno, Leonardo, etc.) and a suitable USB cable. **The Arduino is only used during testing,** it's not a permanent part of the project. If you don't have an Arduino board around, maybe you can borrow one from a geeky friend, or just buy one (they're fun to experiment with)…then work through a couple introductory tutorials (http://adafru.it/dMN) to understand how the software gets installed and code gets uploaded to the board.

If you don't already use it, download and install the **Adafruit NeoPixel Library** for Arduino to run the test code. We have a tutorial for library installation too (http://adafru.it/dit).

Click to download Adafruit NeoPixel library for Arduino

http://adafru.it/cDj

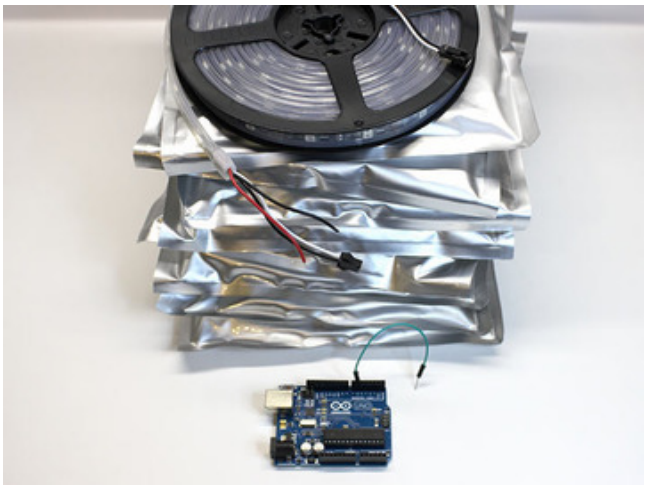
Testing Full NeoPixel Reels

For these tests, do not use the strandtest example code that comes with the NeoPixel library. Instead, copy and paste the code below into a new Arduino sketch, then upload it to the board.

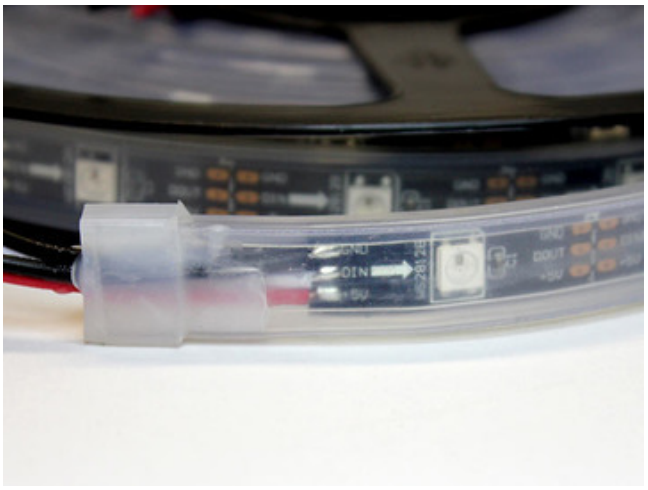
Don't connect any NeoPixels yet. There's a procedure to follow.

```
// Simple NeoPixel test.  Lights just a few pixels at a time so a
// long strip can safely be powered from Arduino 5V pin.  Arduino
// may nonetheless hiccup when LEDs are first connected and not
// accept code.  So upload code first, unplug USB, connect pixels
// to GND FIRST, then +5V and digital pin 6, then re-plug USB.
// A working strip will show a few pixels moving down the line,
// cycling between red, green and blue.  If you get no response,
// might be connected to wrong end of strip -- look for the data
// direction arrows printed on the strip.

#include <Adafruit_NeoPixel.h>

#define PIN    6
#define N_LEDS 150 // 5 meter reel @ 30 LEDs/m

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, PIN, NEO_GRB + NEO_KHZ800);
```

```
void setup() {
  strip.begin();
}

void loop() {
  chase(0xFF0000); // Red
  chase(0x00FF00); // Green
  chase(0x0000FF); // Blue
}

static void chase(uint32_t c) {
  for(uint16_t i=0; i<strip.numPixels()+4; i++) {
      strip.setPixelColor(i  , c); // Draw new pixel
      strip.setPixelColor(i-4, 0); // Erase pixel a few steps back
      strip.show();
      delay(25);
  }
}
```



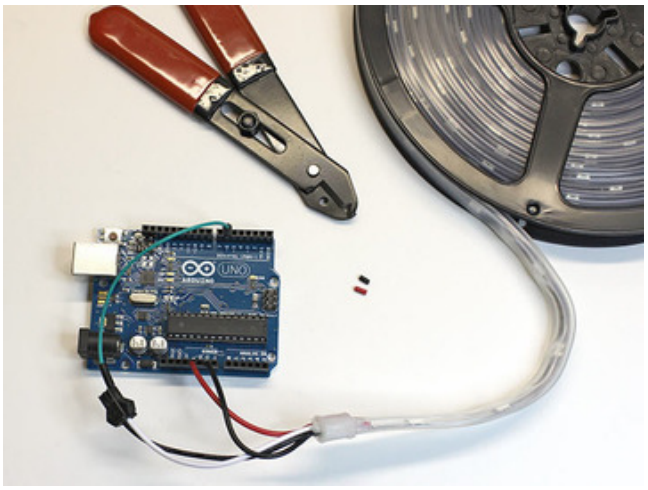
Gather up all your LED reels and the Arduino. You'll also need a small jumper wire (possibly some additional components…more on that in a moment).



Examine each reel to find the “input” end of the strip. Most are coiled with this at the outer edge, but occasionally one’s wound backwards for whatever reason.

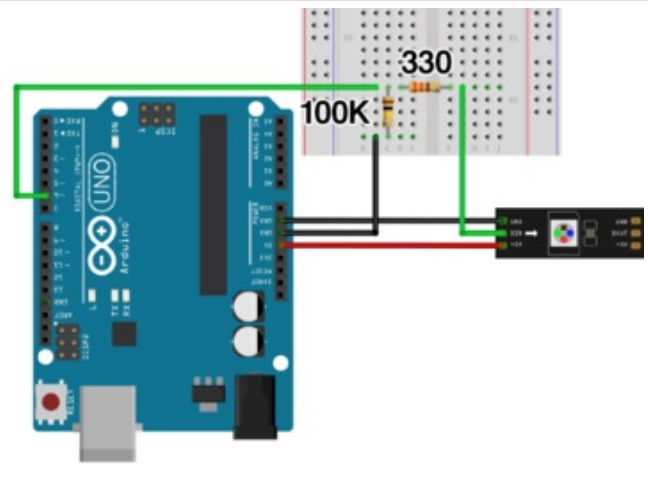
Look for the solder pad labeled “**DIN**” (data input), with the arrow pointing toward the first LED (this shows the direction of data moving down the strip).

Both ends of the strip have a 2-pin JST connector, plus two extra wires for power.



Strip a little extra insulation from the tips of the power wires. Connect the **black** wire to any **GND** on the Arduino, then the **red** wire to **5V** (the Arduino should not be connected to USB yet…try to avoid connecting NeoPixels to a live circuit).

Look closely at the JST plug. One of the two wires (usually white) leads to the DIN pin on the strip. Using a jumper wire, connect **DIN** to Arduino **pin 6**.



RECOMMENDED: if you’re in a dry or static-prone climate, don’t connect the NeoPixel DIN line directly. Use this simple circuit instead:

- Put a 330 Ohm resistor between Arduino pin 6 and DIN on the strip (a slightly larger resistor like 500 Ohms is fine, if that’s what you have around).
- Put a 100K resistor between DIN and GND.

(This is a simplified diagram; the actual NeoPixel strip has a plug on the input.)

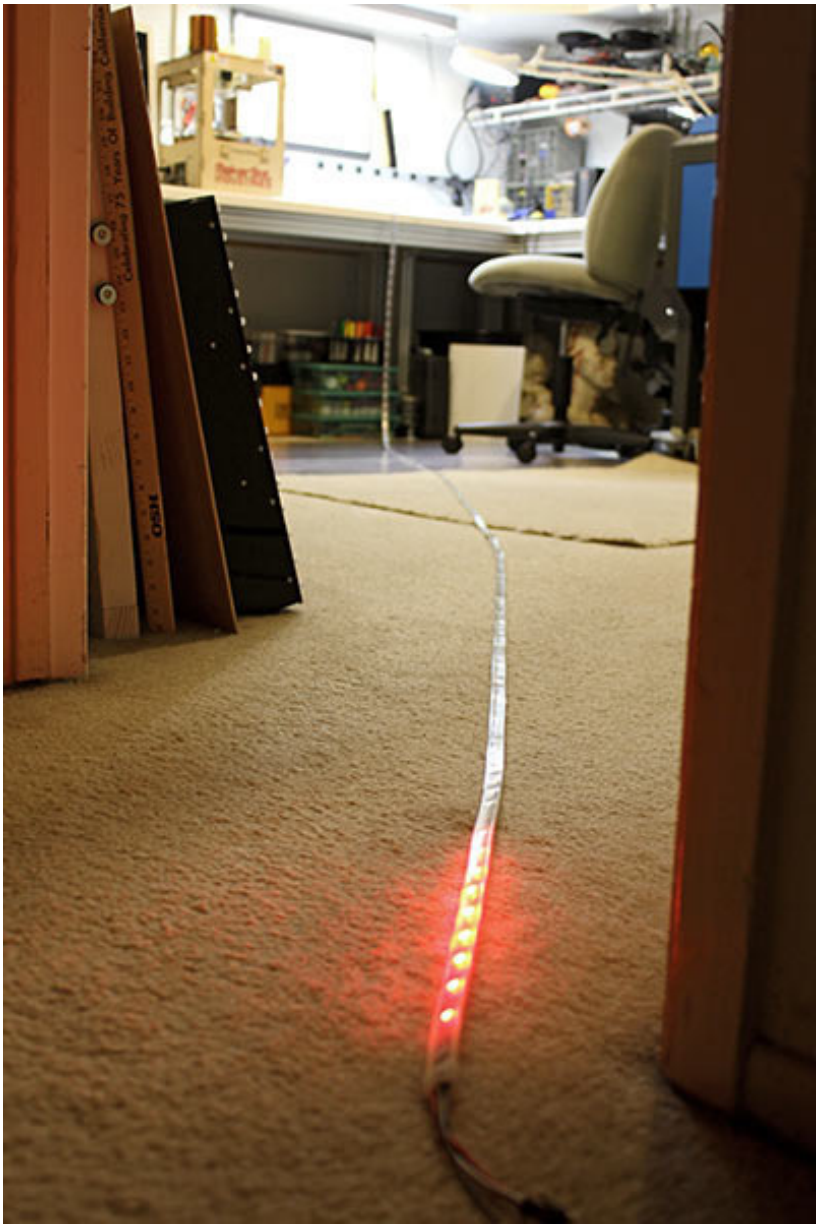
You can do this even if you’re not in a dry climate. Better safe than sorry.

Unroll the NeoPixel strip along the ground or a long workbench, so it all lies flat and facing up. Don't test on the reel…it obscures parts of the strip. **You need a clear view of every pixel!**

Make sure the power wires at the "out" end aren't touching anything or each other!

Now connect the USB cable between the Arduino and a computer's USB port or a USB wall charger. After a moment you should see a few pixels chase down the strip. Red, green, blue, repeat.

If it's working, let this run for a few minutes. Look down the strip, watching closely for any pixels that don't light the right color, or don't light at all. Does the "chase" make it all the way down the strip, or does it cut out part way down?



I am a horrible person, testing electronics on carpet. The concrete floor in the garage would be smarter.

If everything checks out, dismantle it in the reverse order: unplug USB, disconnect DIN, then 5V, then GND. Roll it back up and proceed to the next reel.

Troubleshooting

Nothing lights up!

- If you're using a computer's USB port for power: does the computer report a device is drawing too much current? This is almost certainly an electrical short. Are the power wires at the "out" end touching each other, or anything electrically conductive? How about the Arduino? Is it sitting on one of the anti-static bags? Watch it…they're mildly conductive.
- Check the strip and Arduino wiring against the diagram: GND, 5V and DIN.
- Are you connected to the input end? DIN? Not DOUT.
- Did you upload the test program to the Arduino first?
- If you have a multimeter, check the voltage across the power wires at the "out" end. It'll be lower than 5 Volts, that's normal…but it should not read 0V.

Or it could simply be a bum strip; if nothing else works, set it aside and try the next. If you encounter the same problem a second time, then "user error" is more likely the cause. **Stop immediately** and see "Next Steps" below.

The LEDs cut out part way down the strip!
If you're using the test code exactly as written (150 pixels) on a 5 meter, 30 LED/m reel (150 pixels) and it cuts off, then this is likely a dead NeoPixel or a bad solder joint.

- Pinch down on the strip on the last good pixel and the first bad one. Any change? If so, there's a cracked solder joint. There's ways you can try repairing this, or you can set up an exchange.
- NeoPixel strips are manufactured in 1/2 meter lengths, which are then soldered together to produce a reel. Look closely between the last working pixel and the first bad one. Is this one of the half-meter join spots? If so, give a pinch there too.

One pixel (or a few) don't light, but the chase otherwise makes it down the strip.
It's a defective LED; no amount of pinching or coercion will correct for this. It needs replacing.

Next Steps

As long as they're testing OK, repeat the procedure for every reel. It goes faster if you have friends helping out, assembly-line style.

If one reel is misbehaving, it's probably a defect. If a second reel misbehaves the same,

STOP IMMEDIATELY. You might be killing them!

If you encounter any problems, post on the Adafruit Forums (http://adafru.it/cer) (in the "Glowy Things" section). It's very helpful if you can provide a photo (or several) clearly showing your wiring between the Arduino and NeoPixel strip, and please describe your test setup and the symptoms observed. We'll look it over for gremlins and try to correct any user error. If it's defective, we can then set up an exchange.

For defective parts, you have options:

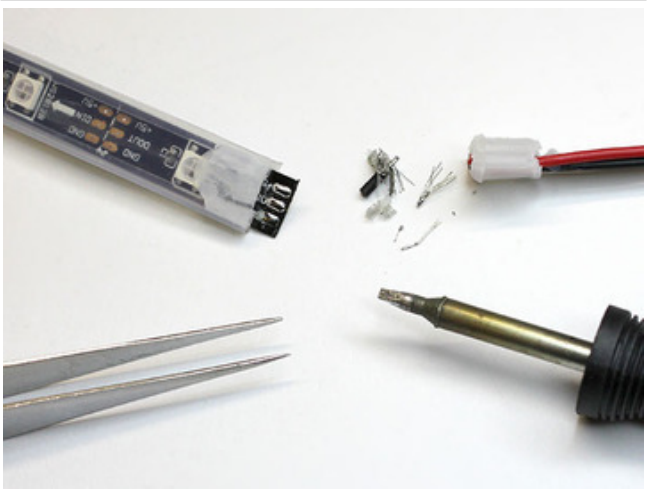
- If you prefer to keep it simple, we can just exchange the complete reel(s).
- Maybe you're eager to make progress. On the next pages, we'll be cutting 2-meter sections from these reels and joining pairs of the remaining 1-meter sections. If you're able and willing to salvage some working 2-meter or 1-meter sections from the 5-meter reel, we can exchange the defective section(s) for an equivalent length of new strip.

(This is one of the reasons for ordering some extra strip. You can continue making progress on the project while replacement parts are en route.)

Prep LED Strips

Now we need to convert these **5-meter** reels into **2-meter** strips for hanging. Each reel provides **two 2-meter strips**, plus **one 1-meter strip**. The latter can be **joined** in pairs to produce a few additional 2-meter lengths.

Peel back the rubber weatherproofing cap on the end of the strip (you might need to cut or tear it away…that's okay, we won't be needing it). Then cut the wires, close to the strip.



Using a soldering iron, scrape the remaining wire tips off the solder pads.

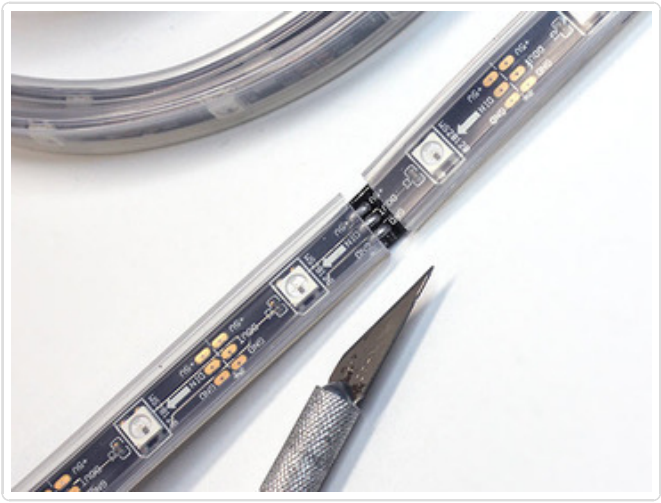
Be **super extra careful** to avoid getting any tiny wire strands or solder balls down inside the sleeve! Also watch out for solder bridges between the pads.

There may be little bits of rubber stuck here and there. Peel them away with tweezers, or scrape them with the iron if they're on the solder pads.

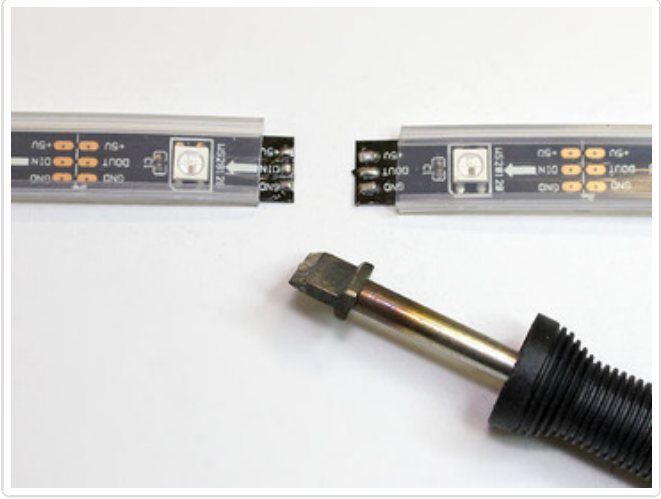
Unroll the strip and repeat for the connector at the opposite end. We won't be using the JST plugs for this project, but save them for future things…they're quite handy!

NeoPixel strips are all manufactured in half-meter lengths. These are joined at the factory to produce a complete reel. If you look along the strip, you can clearly see where the sections are joined…there are solder connections there.

Measure 2 meters down the strip. Or simply count four half-meter sections.



Using a hobby knife or box cutter, carefully cut through the sleeve (but not the strip) at the 2m mark, where the solder blobs are.

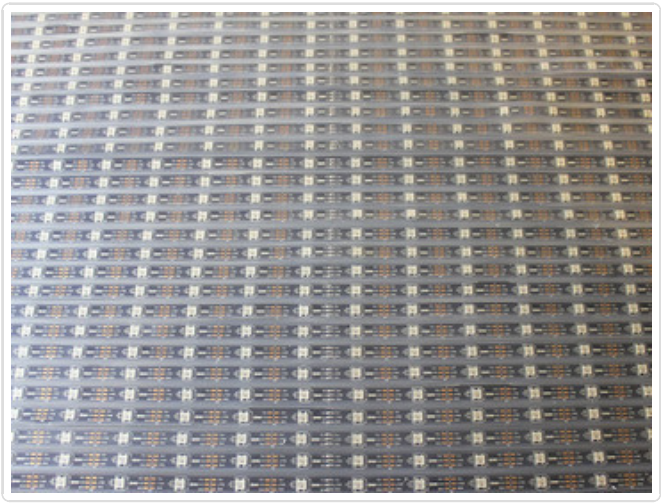


Separate the strip using a soldering iron.

Normally you have to heat the three pads and gently twist the strip away (it may take several passes). Or if you're lucky, you have an iron that they make these extra-wide tips for.

As with the end of the strip, watch out for solder balls and bridged pads.

Measure, cut and desolder a second 2m length of strip. You should then have a 1m section left over.

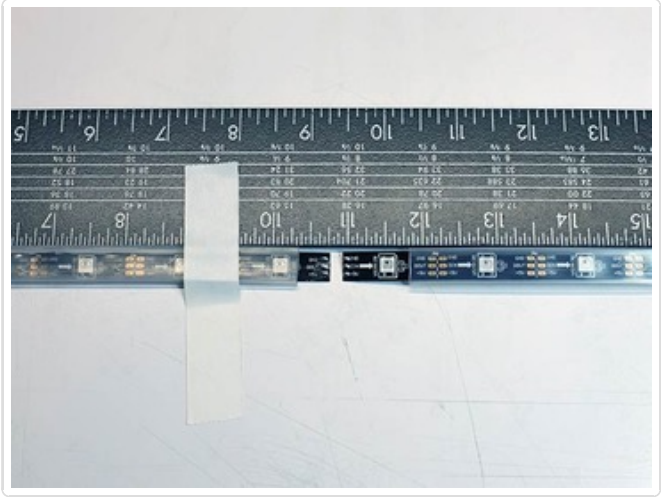


Repeat for all of the NeoPixel reels.

Later steps will be easier and less error-prone if all of the strips (both the 2m and 1m varieties) are laid out flat, all oriented the same way. Don't leave them in a heap of disarray.

Joining the 1m Sections

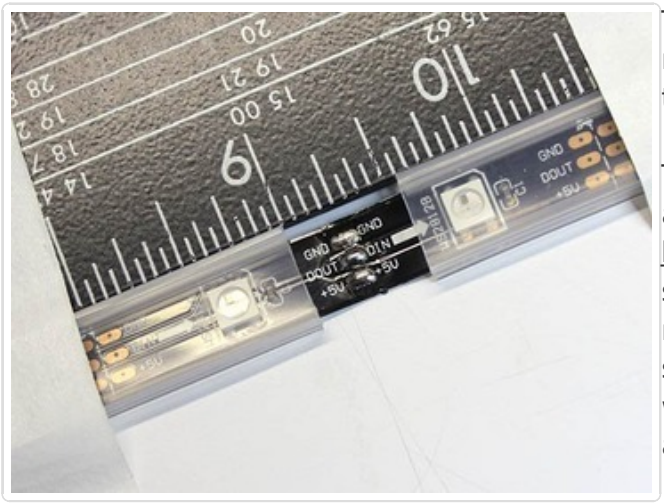
Working with 10 reels of NeoPixels, there's now **20 two-meter strips** and **10 one-meter strips**. The latter will be reconnected into 5 two-meter strips, for 25 two-meter strips total. **24** will be assembled into the curtain, **one** is left over as a spare (for future repairs). With better planning I'd have allowed for a few additional spares!



It's extremely important that the two sections are straight. Even a tiny kink will throw off the whole grid!

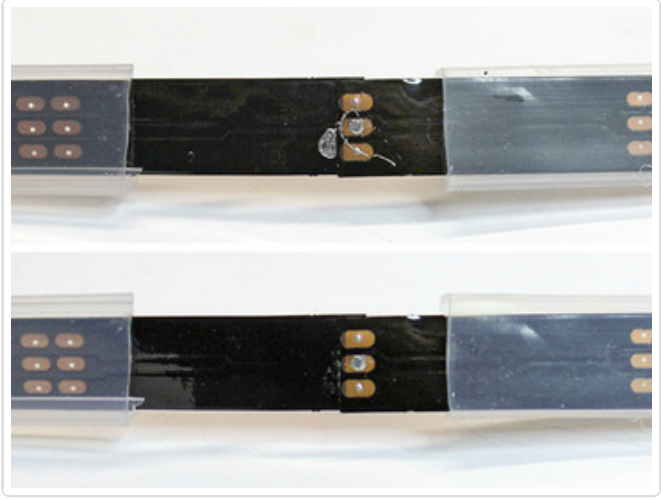
A straightedge is used as an alignment guide; this and the LED strip are taped down to the work surface.

Make sure you're connecting DOUT to DIN. Don't join strips back-to-back!



The end of one strip overlaps the start of the next (one set of pads should be directly over the other) and the connections are re-soldered.

These strips are going to see a lot of abuse, and **the solder connections better be bulletproof!** If you're having trouble successfully re-joining strips, you may need to remove the old solder from the pads (using solder wick) and start fresh. Old solder gets weird and sticky when all the flux has boiled away.



Check for shorts between pads using a multimeter. This strip looked great on top…but it turns out all three pads were bridged underneath! Clean these up with the soldering iron and some solder wick if needed. And once again, watch out for solder balls getting into the sleeve.

Later we'll seal the gap in the sleeve…but **not yet**. It's prudent to test all the strips before making those connections inaccessible.

Adding Wires

We'll now add JST plugs to each strip. Wait…didn't we just *remove* a bunch of JST plugs?

These are different. They're *three-pin* connectors, to match the three pads on the NeoPixel strip. And there's one for every 2m strip (at least 24, plus any spares).

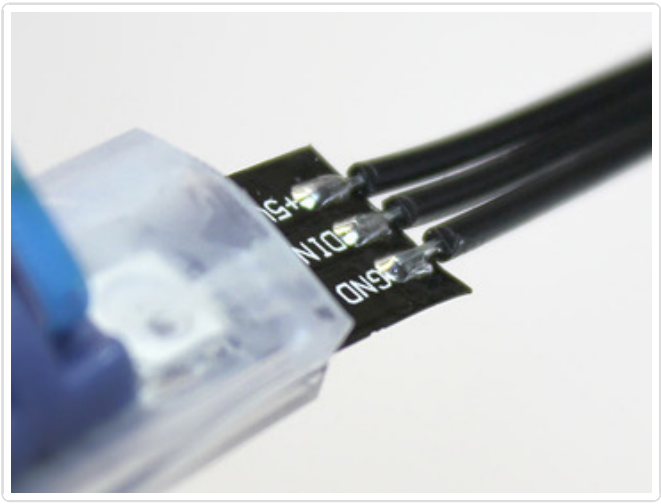
Using plugs (rather than hard-wiring the strips) makes maintenance much easier, if any strips need replacing later.

IMPORTANT: Set the female (socket) receptacles aside for later, **use male (pin) plugs on the strips.** This is the *opposite* of the factory connectors, but it's *safer* (sticky-outy things on the power supply side is a *bad idea*).

The rubber sleeve is cinched back a bit and clamped in place, then the wires from the JST plug are soldered to the strip. *(In hindsight, I might've cut these to half their original length and stripped new ends...but your overall design might be different.)*

JST plugs feature a polarity "key" so they don't plug in backwards. Doesn't matter which orientation you choose, just **make sure every plug/strip is aligned the same way!**

Make sure you're connecting to the DIN end.



Like the joined strips, these solder connections are going to see a *lot* of abuse, so you'd better have some **Nobel Prize-worthy solder connections**, shiny and flowing smoothly between the copper pads and the wires. Not too much, not too little, and not "balled up" on the surface. It may work best to wick off any old solder and apply fresh.

And you know the drill by now: don't let any solder bits get down inside the sleeve.

Repeat this for all 24 strips plus the spares. When you're done, check these things on every strip:

1. Are all the JST plugs oriented the same way on every strip? With the "key" facing either up or down? Doesn't matter which way, just **be consistent.**
2. Did you connect to the DIN end?
3. Use a multimeter to test for electrical shorts: GND to DIN, DIN to +5V, and GND to +5V.
4. For the joined strips, confirm DOUT connects to DIN; arrows should point the same way down the whole strip.

Do not continue until every strip checks out!

Test Again

Sorry for the repetition, but it's really important.

Return to the Arduino code that was used for testing reels, and make one small change near the top:

```
#define N_LEDS 60 // 2 meter strip @ 30 LEDs/m
```

Upload this modified code to the Arduino, then disconnect the USB cable.

Take one of the mating JST receptacles (female socket) and strip a little extra insulation from the wires. Plug it into one of the strips and trace the wires so you know exactly which is +5V, GND and DIN. Double check, then maybe even stick labels on them.

Connect the wires to the corresponding points on the Arduino: 5V, GND and pin 6. If

possible, route DIN through a static-protection circuit as shown on the prior page.

Re-connect the USB cable. You should get the LED “chase” again, red, green and blue. Let it run for a couple minutes as you examine the entire length of the strip. If everything checks out, unplug USB first, then the JST connector, then reverse these steps as you connect the next strip for testing (we’re avoiding connecting the strips to a live circuit).

Troubleshooting is about the same as for reels. If there’s a problem, stop immediately and refer to the checklists on the prior page, and also look for the following:

- Confirm that 5V, GND and DIN are properly routed from the strip, through the JST connectors, to the Arduino.
- Confirm you’ve soldered the plug to the DIN end.
- Confirm all the JST plugs are oriented the same way. You can have all the “keys” on the NeoPixel side of the strip, or all on the back, but not a mix of the two.
- For joined strips, confirm you’ve soldered them DOUT to DIN (arrows all pointing the same way), not back-to-back.

Sealing the Gaps

Do not continue until every strip has been tested. Going forward, all those solder connections become inaccessible…if something was mis-wired, it might not be salvageable after this.

Set aside all the full 2m strips. This next part applies only to the joined (2X 1m) strips.

Lets fix those gaps in the sleeve, where the two pieces meet…

Cut some 1-inch sections of clear heat-shrink tube (1/2" diameter). Slide one over the strip… close to, but not yet on top of, the gap.

Wriggle the sleeves so the ends butt together close to the solder connection. You may need to hold the end of the strip with pliers and “inchworm” the sleeve down a few millimeters over several passes.

Squeeze some hot-melt glue or Permatex 66B under both sleeves, top and bottom (tested both…hot glue is cleaner). Re-butt them, slide the tubing over the gap and shrink it using a heat gun.

You really need to **use a heat gun** for this. Flame does not work well on large heat-shrink, and may be unsafe for certain glues!

It's not beautiful...but with the LEDs turned on, and from a couple feet away, nobody will notice.

Aside from weatherproofing, the glue + heat shrink provide some much-needed strain relief for the solder connections.

Hang ’Em High

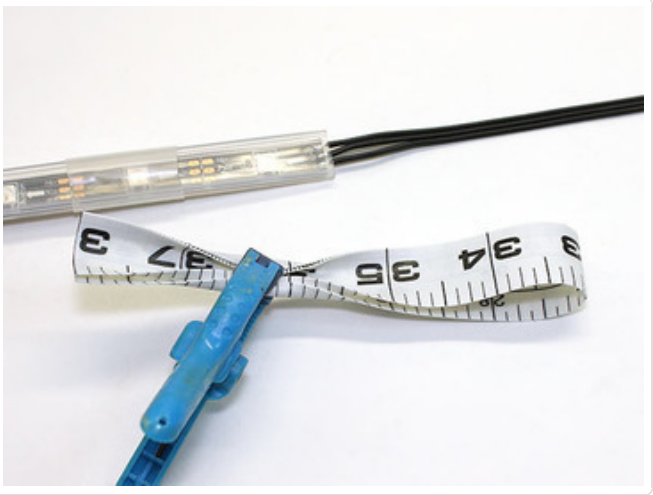
The most challenging aspect of the project wasn’t software or electrical, but *mechanical.* How to suspend all these strips while simultaneously meeting several seemingly incompatible criteria?

- Keep the strips all facing “flat,” but still allow for a bit of twist.
- Durable enough to withstand the occasional tug of the curious; won’t snap or tear.
- Doesn’t rely on the wires for load-bearing.
- Nonconductive; doesn’t short the electrical parts.

Wasn’t satisfied with some 3D printed and laser-cut parts. Brought home about half the items in the hardware store to experiment with…

The winner? Fiberglass measuring tape, of all things. Must be **fiberglass** tape! Vinyl tape isn’t as strong, and anything paper-based would go pulpy in a humid setting. Plastic pallet strapping tape might also be a good choice.

I bought several tapes, cut off the first inch (with the metal tab) and cut the remainders into **10" lengths.**

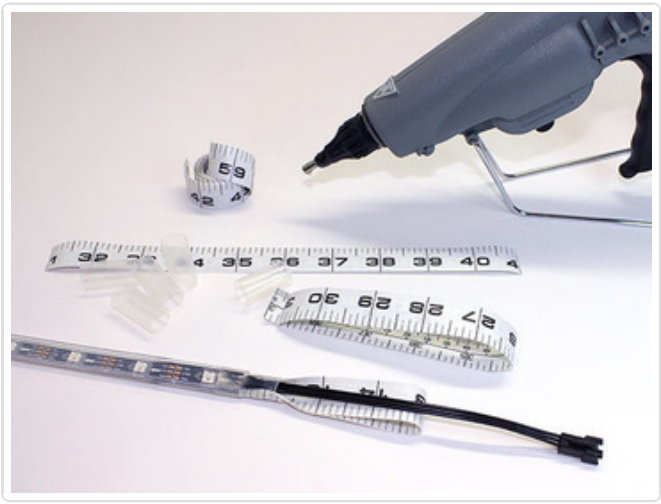


These were doubled over to form a **loop** for hanging. The leader needed to be rolled in order to fit down the sleeve…narrower tape would avoid that.

A 1" piece of **heat-shrink tube** (1/2" diameter) is slid over the strip, close to the end.

The tape gets crammed down the *back* of the strip…the side without NeoPixels.

Make sure the distance is the same on every strip, so they all hang at the same height to form a grid. Also, enough contact area for durability. 1.5 inches felt about right.



Hot-melt glue is squeezed down into the strip, both front and back, then the heat-shrink tube is slid over the end and finished with a heat gun.

This is a balancing act. The tape will melt if exposed to too much heat, but the tubing won't properly shrink with too little. Practice with some scrap pieces first to find the proper heat setting and distance.

These loops bear the weight of the strips, reducing strain on the wires and solder joints.

The tail ends of the strips also need to be sealed using hot glue and shrink tube. Immediately after heating, pinch the tip flat using pliers to seal it, and trim away any excess after it cools.

Having been stored on reels, the strips tend to have some curl to them. A little extra weight at the tip can help…something like these tiny glass marbles (glass = nonconductive) can go inside the heat-shrink. Totally optional though, maybe not worth the extra effort.

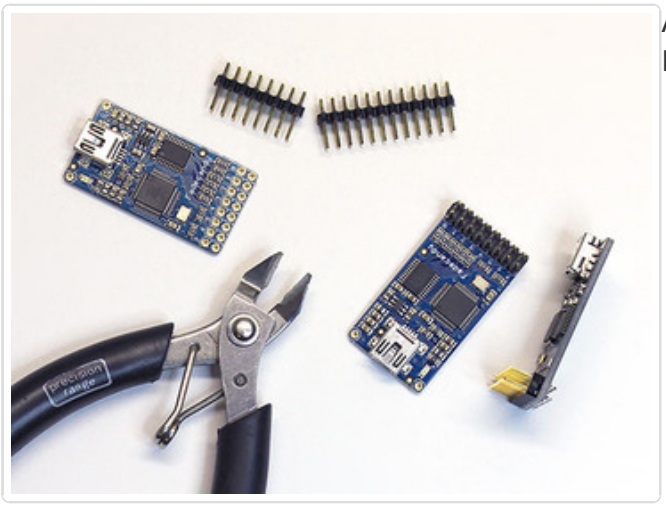
All this manhandling and hot glue can be rough on the strips. Hate to say this, but you may want to repeat the Arduino test on all of them. It's easier to resolve problems now than when the whole curtain is assembled.

Wiring Harnesses

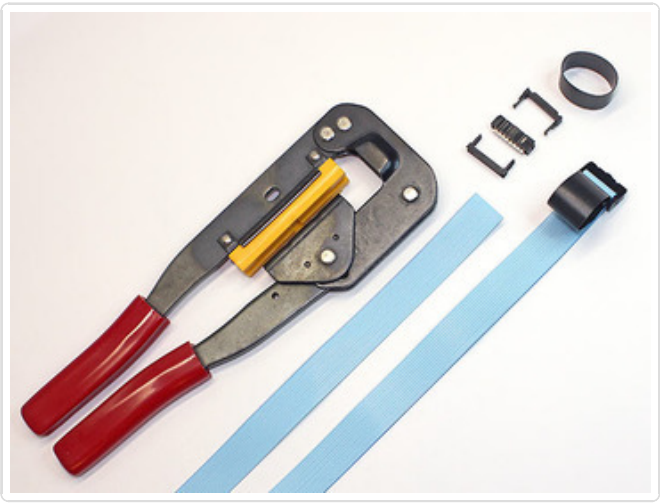
For the sake of troubleshooting and maintenance, I did not want to hard-wire everything together. If a Fadecandy board or something else needs to be replaced, there's a degree of modularity and hierarchy to the design.

Keeping with the "three groups" motif…three Fadecandy boards, three sets of power terminals…three identical wiring harnesses were built, each distributing power and data to 8 NeoPixel strips. Each block consists of:

- 1X USB cable
- 1X Fadecandy board
- 1X 2-row, 16-pin IDC header and plug
- 1X 16-conductor ribbon cable
- 8X JST **female** receptacle
- 8X 22 gauge wires for 5V
- 8X 22 gauge wires for GND
- 4X 18 gauge wires for 5V
- 4X 18 gauge wires for GND
- 1X 12 gauge wire for 5V
- 1X 12 gauge wire for GND
- 2X Power distribution bus bars (1 ea. for 5V and GND)
- Lots of **heat-shrink tubing** in various sizes (**do not** use electrical tape)



A 16-pin (8x2) header is soldered to each Fadecandy board.



A mating IDC header is assembled using 16-conductor ribbon cable, sufficiently long to bridge the anticipated distance between the Fadecandy board and strips. Use an IDC crimper or a vise for this. Do not use pliers.

Goal is to get pin 1 of the cable (usually marked with a dark stripe) connected to pin 0 on the Fadecandy board.

IDC cables can be a topological nightmare; accounting for the doubled-back strain relief always gets me.

Examine the header for a pin 1 mark (if any), align this end with pin 1 on the cable and the "0" pin on the Fadecandy board. You may need to try a dummy build and test with a multimeter.

Take one of the JST female receptacles and plug it into one of your assembled strips, tracing back the 5V, data and GND wires. Or if you still have the receptacle from your Arduino testing, use that for reference.

For each receptacle, add an extension to the 5V wire, **22 gauge** or heavier. Make sure they're all oriented the right way, with the key in the same position.
Use a robust (http://adafru.it/dOA)**inline splice (http://adafru.it/dOA)** for the connection: twist the wires around each other, solder and cover with heat-shrink.

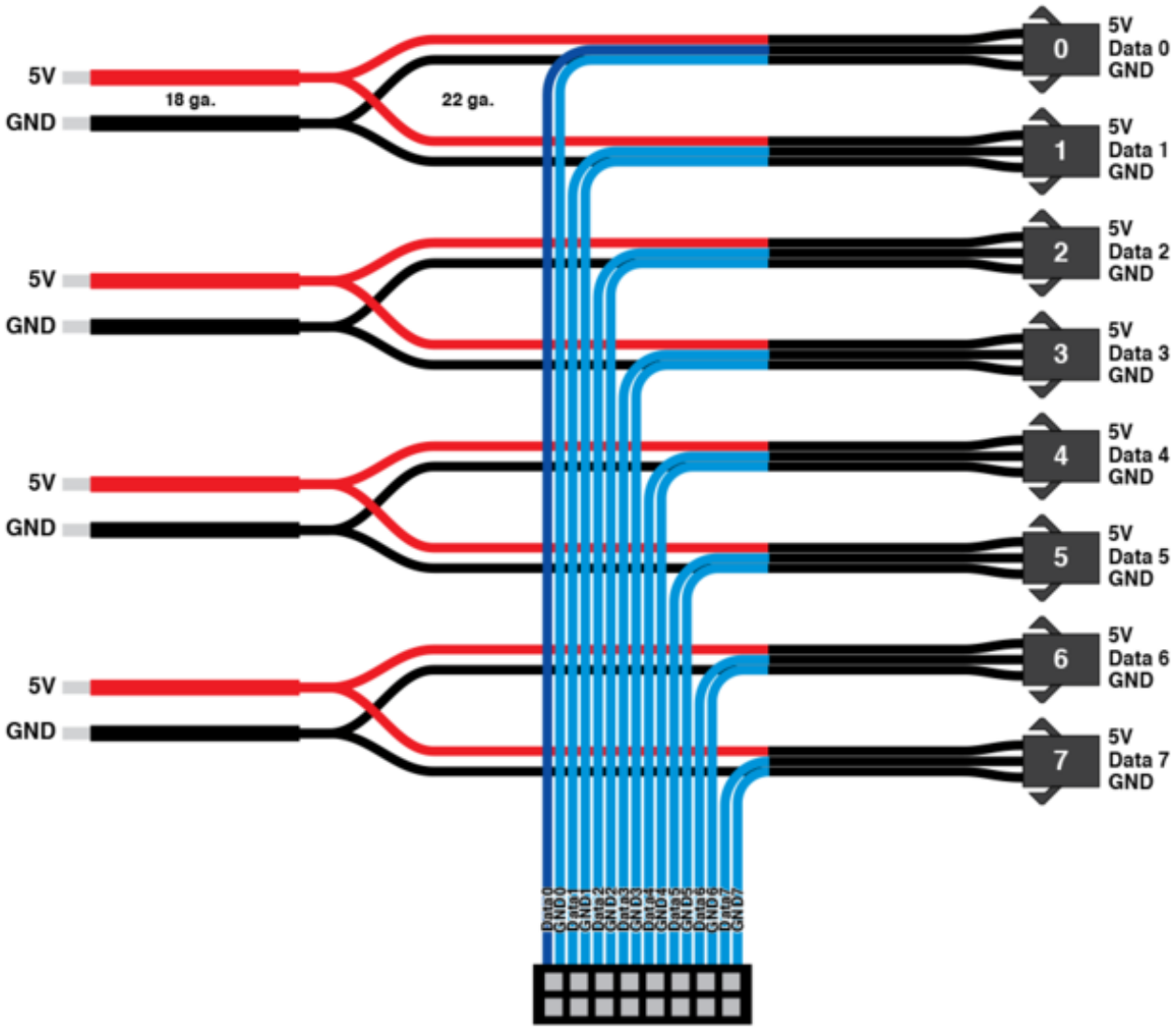
Take the first two conductors of the ribbon cable and peel away a few inches. The first conductor — **pin 1** — gets connected to the **data line for strip 0**. Slide the heat-shrink tube over the wire *before* making the inline splice.

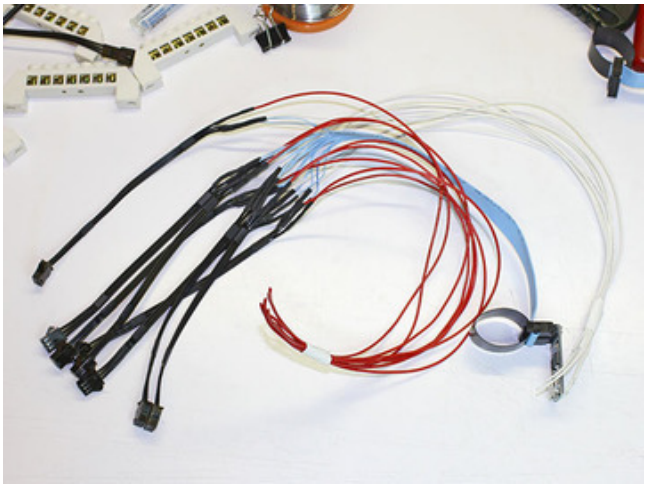
The second conductor — **pin 2** — then goes to that strip's GND wire…**but**, like was previously done for the 5V wires, this wire *also* needs a 22 ga. extension. It's a **three-way inline splice.**

Repeat these steps, peeling pairs of wires off the ribbon cable and connecting to each receptacle's data & GND wires.

The eight 5V and GND wires all need to meet up with 12 gauge wire to the power supply… that's 9 wires total, but the power distribution bus bars only accommodate 7 wires each (and don't take well to doubling-up). So pairs of 5V and GND wires were joined with 18 gauge pigtails (which can handle two strips worth of current) using three-way splices.

Conceptually, it's similar to this:

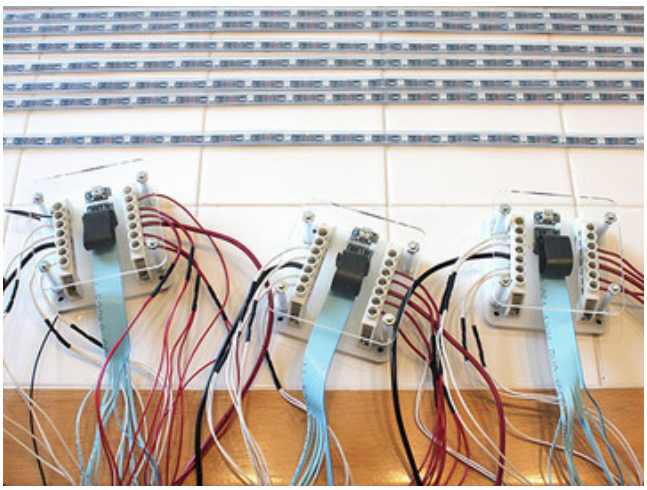




Here's one of the wiring harnesses nearly complete. The 18 gauge power pigtails haven't yet been added.

The project requires three of these harnesses. It's really quite laborious. This is why Burning Man folks have soldering parties.

If you have a label-maker, it's extremely helpful to add strip number "flags" to each JST cable! They need to connect to strips in a specific physical order.



Being a stickler for details, I made these laser cut holders, but you DON'T need to go to such lengths! Fastening everything to a board would work just fine.

Each one holds a Fadecandy board, two power distribution bars (5V and GND) and the wiring harness for 8 NeoPixel strips.

Using a multimeter's continuity beep feature, do a "sanity check" on each section. There should be no connection between 5V and GND. Also test the data and ground pins on the Fadecandy boards: there should be no shorts between any of the data wires, but the ground wires should all connect.

With **no strips attached** and with the **power supply switched off**, each unit was wired in one at a time. Power up, confirm power supply is happy, power down and repeat with the next.

Raspberry Pi Setup

We'll start with a fresh install of the latest **Raspbian** OS release. As we're using a Model B+, any version from 2014-06-20 or later should work:

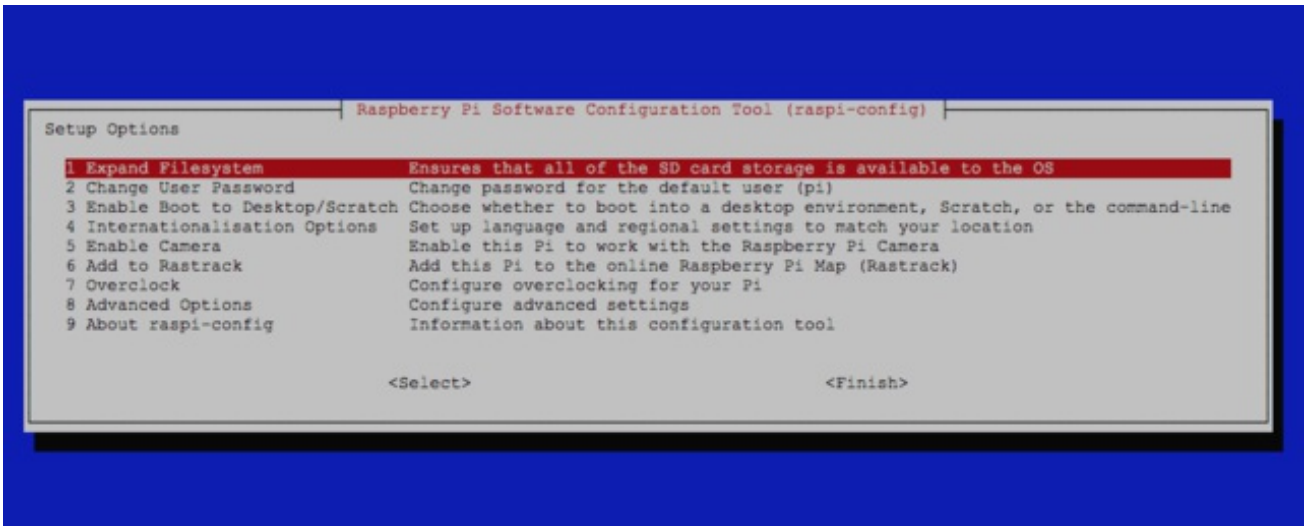
Raspberry Pi Downloads Page (http://adafru.it/dpb)

If you're new to Raspberry Pi, we strongly suggest working through the first few guides in the **Learn Raspberry Pi** (http://adafru.it/dpe) tutorial series…know how to "burn" an SD image, perform a first-time setup and get the Raspberry Pi connected to a network. Some familiarity with one of the text editors (such as the simple *nano* or the more daunting *vi* or *emacs*) is also recommended.

Install the Raspbian image on a **4GB or larger** microSD card. You'll need to connect a **monitor** and **USB keyboard** for basic system configuration, but this is only temporary… we'll set it up to run "headless" later. For networking, connect either an Ethernet cable or a USB WiFi adapter.

Basic Setup

On first boot, the raspi-config utility will automatically start:



The following selections are **required**:

- Expand Filesystem

The following are **optional**, but **strongly recommended**:

- Change User Password (because everybody knows the default).
- Under "Internationalization Options," select "Change Locale," "Change Timezone" and "Change Keyboard Layout" to suit your location. If your keyboard isn't producing the

expected symbols, this is why.
- Change Hostname (under "Advanced Options"). I named mine "**curtain**" to distinguish it from other Raspberry Pi systems on the network.
- Enable SSH (also under "Advanced Options"). This allows remote login from another system on the network, for performing administration tasks without a display attached.
- You can disable Overscan (Advanced Options) if you like — we'll reboot a few times during the setup process, and this provides a little extra screen real estate on HDMI monitors.

You do not need to change the Memory Split or Overclock settings. The Pi can handle Fadecandy just fine with the standard configuration.

Tab over to the "Finish" button, press Return and confirm you'd like to reboot the system when prompted. You'll need to log in manually now, using the password you established above.

Network Setup

Even if your goal is a standalone system (with all the LED animation being generated on the Raspberry Pi, not streamed over a network), it's still helpful to get networking up at least temporarily, to install packages and updates, or for remote administration.

If using a wired network, just connect an Ethernet cable and you're done…skip ahead to the next section below. Wireless networks are a little more involved…

For most WiFi networks — those that broadcast their SSID (network name) — you can type startx and (with a USB mouse connected) click the WiFi Config icon on the desktop to configure your network adapter. Super easy.

For WPA2-protected "hidden" WiFi networks — those that don't broadcast their SSID — you can try the following steps. **This is not guaranteed to work**…we strongly recommend just using a broadcast network name (with WPA2 password), since *hidden networks are not inherently more secure!* But you can give it a shot…

First, edit the WPA Supplicant configuration file:

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

Use the following, completely replacing the contents of that file:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
ap_scan=2
eapol_version=1
network={
 ssid="WIFI_NAME_GOES_HERE"
```

```
 scan_ssid=1
 mode=0
 proto=WPA2
 auth_alg=OPEN
 pairwise=CCMP
 group=CCMP
 key_mgmt=WPA-PSK
 psk="WIFI_PASSWORD_GOES_HERE"
}
```

Edit the ssid and psk strings to match your WiFi network name and password.

Make sure those lines in the network section are indented using tabs, not spaces. It seems to be quite picky about formatting.

If using a WiFi adapter based on the popular Realtek 8192CU chipset, disabling WiFi power management seems to help with reliability:

```
echo "options 8192cu rtw_power_mgnt=0 rtw_enusbss=0" | sudo tee --append /etc/modprobe.d/819
```

WiFi should then be active on the next boot:

sudo reboot

Log in again, and you should be able to access the outside world now:

ping adafruit.com

Do not proceed until internet access is working. If WiFi refuses to cooperate, make sure every step above has been followed carefully. If you're trying to use a hidden network and it just won't play nice, change the router configuration to broadcast the network name.

A Little More Network Setup

The monitor and keyboard are needed for just a moment more…

Installing *netatalk* enables *Zeroconf* (aka Bonjour) networking, so the system appears on the network as "curtain.local" (or whatever hostname you configured) instead of a numeric IP address:

sudo apt-get -y install netatalk

As a bonus for Mac users, this also enables AppleTalk sharing, which can make it easier to

transfer files to and from the system if needed.

With netatalk installed, you can easily access the Raspberry Pi remotely using an ssh client from another system on the network. For example, using the Terminal application in Mac OS X, one would type:

ssh pi@curtain.local

You should get a password prompt. Once logged in, you can perform all administration duties remotely (including the steps that follow), and the monitor and keyboard are no longer needed on the Raspberry Pi.

As an **optional** step, you can make sure your Pi is running the most up-to-date packages:

sudo apt-get -y update
sudo apt-get -y upgrade

This can be really time-consuming, and honestly your Pi is fine running the OS as it arrived on the disk image…some folks are just sticklers for detail.

Remember…

We're not done yet, but something to keep in mind for the end of the day: so as not to lose all this hard work, make sure the Raspberry Pi is always shut down using the proper procedure. *Never* just pull the plug on a Linux system, they *hate* that!

sudo shutdown -h now

Wait about 15 seconds and you can then safely disconnect power. Or, if a monitor is attached, wait for the "Will now halt" message.

Logging in remotely every time you want to turn off the curtain can be a bit of a nuisance. We wrote a small utility that handles this task just by pressing a button connected to the Pi's GPIO header. The software can be installed with:

git clone git://github.com/adafruit/Adafruit-GPIO-Halt
cd Adafruit-GPIO-Halt
make
sudo make install
sudo nano /etc/rc.local

Insert this one line before the final 'exit 0':

```
/usr/local/bin/gpio-halt &
```

An alternate pin number can optionally be specified before the “&”. The default — GPIO21 — was chosen because this and a GND are the last two pins on the Model B+ GPIO header, making it very simple to add a momentary button (http://adafru.it/1445) using one of these quick-connects (http://adafru.it/dMr). On a “classic” Pi (original Model A or B), GPIO7 is similarly convenient, with an adjacent GND pin at the end of the header…just be sure to use “gpio-halt 7 &” in that case.

Once installed, gpio-halt will be active on the next reboot.

After pressing the button, you’ll still need to wait about 15 seconds before disconnecting power.

This is NOT an emergency off button, nor does it disconnect power. You may want a dedicated switch on the input to the power supply.

Fadecandy Server Setup

At this point you should have gone through the network configuration steps from the prior page, and should have working internet access before continuing. There's a little more software to be fetched…

Recent Raspbian versions have the git tool preinstalled, but for posterity let's confirm:

sudo apt-get -y install git

Then retrieve the Fadecandy software from Github:

git clone git://github.com/scanlime/fadecandy

The package includes a pre-built executable for Raspberry Pi, but it's built on an older version of Raspbian and won't work on the current system. Not to worry, a new one can be compiled in just a few steps:

cd fadecandy/server
make submodules
make

This takes about 10 minutes to complete, depending on your internet connection. Once it's finished, type:

sudo mv fcserver /usr/local/bin

To make the fcserver program start automatically when the system boots:

sudo nano /etc/rc.local

Just above the final "exit 0" line, copy and paste the following:

```
/usr/local/bin/fcserver /usr/local/bin/fcserver.json >/var/log/fcserver.log 2>&1 &
```

Then create a new configuration file:

sudo nano /usr/local/bin/fcserver.json

Copy and paste the following block into the new file:

```
{
 "listen": [null, 7890],
 "verbose": true,
```

```
"color": {
 "gamma": 2.5,
 "whitepoint": [0.7, 0.7, 0.7]
},

"devices": [
 {
  "type": "fadecandy",
  "serial": "TVUCPRXHXJQJOFKR",
  "map": [
   [ 0,   0,   0, 60 ],
   [ 0,  60,  64, 60 ],
   [ 0, 120, 128, 60 ],
   [ 0, 180, 192, 60 ],
   [ 0, 240, 256, 60 ],
   [ 0, 300, 320, 60 ],
   [ 0, 360, 384, 60 ],
   [ 0, 420, 448, 60 ]
  ]
 },
 {
  "type": "fadecandy",
  "serial": "DBDJIGEEZLWFYVYJ",
  "map": [
   [ 0, 480,   0, 60 ],
   [ 0, 540,  64, 60 ],
   [ 0, 600, 128, 60 ],
   [ 0, 660, 192, 60 ],
   [ 0, 720, 256, 60 ],
   [ 0, 780, 320, 60 ],
   [ 0, 840, 384, 60 ],
   [ 0, 900, 448, 60 ]
  ]
 },
 {
  "type": "fadecandy",
  "serial": "SIUEOKCKPNKVOLSN",
  "map": [
   [ 0, 960,   0, 60 ],
   [ 0, 1020,  64, 60 ],
   [ 0, 1080, 128, 60 ],
   [ 0, 1140, 192, 60 ],
   [ 0, 1200, 256, 60 ],
   [ 0, 1260, 320, 60 ],
   [ 0, 1320, 384, 60 ],
   [ 0, 1380, 448, 60 ]
  ]
```

```
  }
 ]
}
```

The “serial” strings refer to the unique serial numbers of each Fadecandy board: left, center and right. The default serial numbers above are meaningless…you’ll need to change them to the serial numbers of your own boards.

There are a couple of ways to get your serial number(s). If you haven’t yet rebooted the system since installing the fcserver program above, run it manually:

fcserver

Then connect each of your Fadecandy boards to a USB port. You’ll see messages such as this one:

```
USB device Fadecandy (Serial# TVUCPRXHXJQJOFKR, Version 1.07) attached.
```

There’s your serial number, which you can copy to the serial: line in the fcserver.json file (make sure it’s between quotes). Repeat for each board. Press Control+C to stop the fcserver program.

If you’ve rebooted since installing fcserver, it’s already running in the background now, and you’ll instead find the connection messages in the log file:

tail -f /var/log/fcserver.log

Once configured, you can hunt down and restart the fcserver process, but I find it much easier just to reboot:

sudo reboot

If fcserver doesn’t seem to be starting up, check the log as shown above. It’ll warn of configuration file errors (the format is very exacting…missing or misplaced characters are not uncommon).

Other Configuration Highlights

Notice this line near the top of the fcserver.json file:

```
"whitepoint": [0.7, 0.7, 0.7]
```

This sets the overall color balance (red, green and blue, range 0.0 to 1.0) and maximum brightness of the LED matrix. Brightness has intentionally been dialed back to 70% (0.7), as explained on the "Power" page.

At their brightest, each NeoPixel can draw up to 60 milliamps of current. With 1,440 of them, that's up to 86.4 Amps, exceeding the limits of our 60A power supply. 60 ÷ 86.4 = 0.746, but we'll round down to 0.7 for a slight safety margin. **If using a different power supply or a different number of NeoPixels, adjust these numbers to suit.**

Then there's these "map" sections in the file:

```
"map": [
 [ 0,   0,   0, 60 ],
 [ 0,  60,  64, 60 ],
 [ 0, 120, 128, 60 ],
 [ 0, 180, 192, 60 ],
 [ 0, 240, 256, 60 ],
 [ 0, 300, 320, 60 ],
 [ 0, 360, 384, 60 ],
 [ 0, 420, 448, 60 ]
]
```

Each Fadecandy board controls exactly 512 NeoPixels: 8 lines of 64 pixels each. This is unvarying and by design.

Our 2-meter curtain strands each contain **60** NeoPixels. We *could* just produce data for 64 pixels and let the surplus be ignored off the end of the strip. But as a persnickety thing I didn't like having to account for the extra nonexistent pixels in code. These map sections let us address the LEDs *contiguously* in software, despite the *gaps* in the physical layout. Each line contains four values:

1. A channel number. This will always be **0** in our application.
2. A starting pixel number *as we'd like to address them.* For example, the first pixel of the second strand will be 60 (normally it would be 64). First pixel of the third strand will be 120. And so forth through 1380, the first pixel of the last strand.
3. The corresponding pixel number *as handled by this Fadecandy board.* Remember, it regards every strand as 64 pixels, always, even if physically shorter. So we've mapped the second strand to position 64, third to 128, etc. Unlike the prior value, these numbers apply to the current board only, not globally, so they'll always be from 0 to 511, never larger.
4. The number of pixels being remapped. 60 in this case, the length of our strips.

If you've created a different strip layout, you'll need to adjust these tables accordingly. Or if using a Fadecandy board in its ideal configuration — 8 strips of 64 pixels — you can leave the map section out altogether.

Dry Run

Given both the cost and the power consumption of all those LEDs, it's prudent to **test the system in stages**; any failure that comes up can be diagnosed and repaired on a more sensible scale. We'll try **one strip** first, then three (one for each Fadecandy), then the complete matrix, powering down the system between each test (NeoPixels don't like being hot-swapped in a live circuit).

Find a spot where you can spread things out, providing a safe buffer between all electrically conductive parts and a good view of the whole system.

Download the same **Fandycandy package from Github** (http://adafru.it/cDN) to whatever system will be synthesizing your animation. A laptop, for instance.

Unzip the file and look inside the examples/html folder. Open the file ganzfeld.html in a text editor. Look for the following line, around line 78 last time I checked, in the writeFrame() function:

```
    var leds = 512;
```

Change this to the actual size of the LED curtain array. You can make this an expression if you like…for example, three Fadecandy boards, 8 strips each, 60 pixels per strip:

```
    var leds = 3 * 8 * 60;
```

Then, around line 131, look for this WebSocket line:

```
var socket = new WebSocket('ws://localhost:7890');
```

Change this to:

```
var socket = new WebSocket('ws://curtain.local:7890');
```

(Change "curtain.local" to your actual hostname if you used something different. But leave the 7890 part.)

On your first test, you should only have **one strip connected**, on a single Fadecandy board. We'll add more with each subsequent pass.

Save the changes, then drag-and-drop this file to a web browser.

All currently-connected NeoPixels should flash white. You can fiddle with the sliders or the

color value to change the effect. *If you or anyone nearby is sensitive to flashing lights, I'd suggest setting the "Frequency" slider to the lowest value immediately, no joke.*

This provides an opportunity to test and troubleshoot the LED matrix. Look for gaps, disabled strips or entire missing sections. If everything's working, allow it to run for a few minutes. Feel the power wires…if they're hot, the gauge of wire is inadequate for carrying that much current.

Nothing lights up!

- Did you insert the correct hostname or address in the WebSocket() call?
- Confirm the Raspberry Pi has booted and is connected to the local network. Following steps on prior pages, you should be able to log in remotely via ssh.
- Confirm the fcserver program is running. Log into the Raspberry Pi and check the contents of /var/log/fcserver.log. If this file doesn't exist, perhaps you forgot to edit /etc/rc.local to launch fcserver, or made a typo?
- Check fcserver.log to confirm that all the Fadecandy boards are connected. Check the serial numbers against the values in fcserver.json.
- Using a multimeter, check that the power supply is outputting 5V DC. Or if the power supply has a fan, confirm it's spinning. If not, probably a blown fuse in the supply due to an electrical short in your wiring.

A whole section of 8 strips isn't lighting!

- Often a typo in the serial numbers. Check fcserver.log to get the numbers for the currently-connected boards, and copy-and-paste these exactly to fcserver.json, making sure they're in quotes.
- If that's not it, check the number ranges in the map sections of fcserver.json. Remember that the third column of numbers applies *per board* — it's always from 0 to 511 — it's *not* a global index across the whole system.
- Could also be a blown fuse, if you've fused each group separately. This can happen if there's an electrical short in that section. Check with a multimeter.

One (or more) individual strips aren't lighting!
Swap the connections between a working and non-working strip.

If the problem moves to the other strip — the previously non-working strip now works, and the previously good strip now won't light — this is usually an error in fcserver.json, probably a map section.

Or, if the same strip doesn't light on a different (known good) connection, it might be wired wrong, there may be a problem with the strip itself…a bad solder connection, blown NeoPixel or conductive detritus inside the strip casing. Test the strip in isolation on an Arduino.

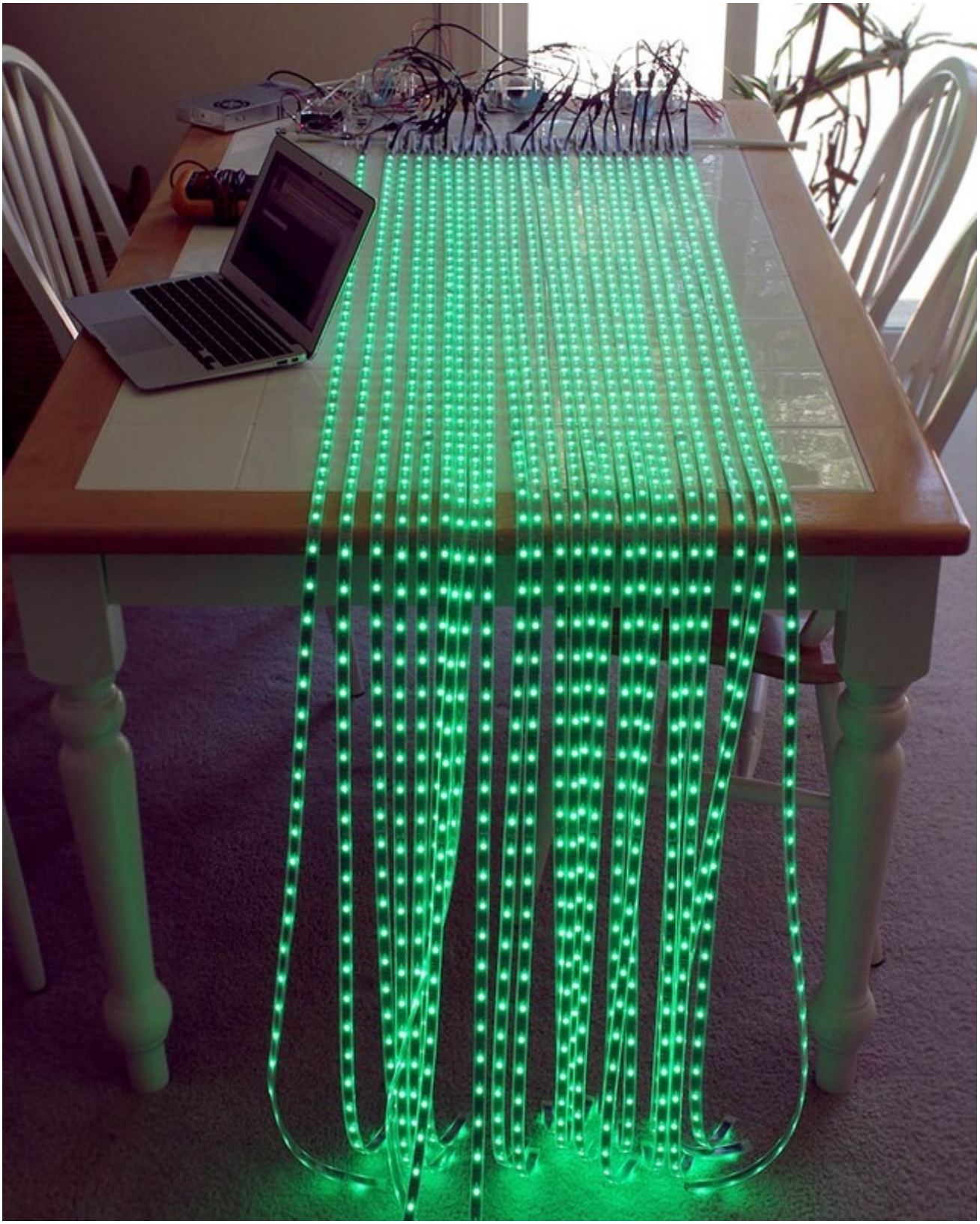
Part of a strip isn't lighting!
Same troubleshooting as above; swap "good" and "bad" strips and observe any changes.

The animation gets “chunky” after a moment!
This is normal, especially on a laptop, as the JavaScript interpreter in the browser enforces power-saving measures.

Later we’ll try some more sophisticated clients that won’t exhibit this symptom.

Success!

If the test works with one strip, power down the system (try the halt button, confirm that’s working) and add one strip to each of the other Fadecandy boards (three strips total). Power it up and repeat the test. If that works, try for the whole enchilada, but be ready to pull the plug at the first sign of trouble.



Construction

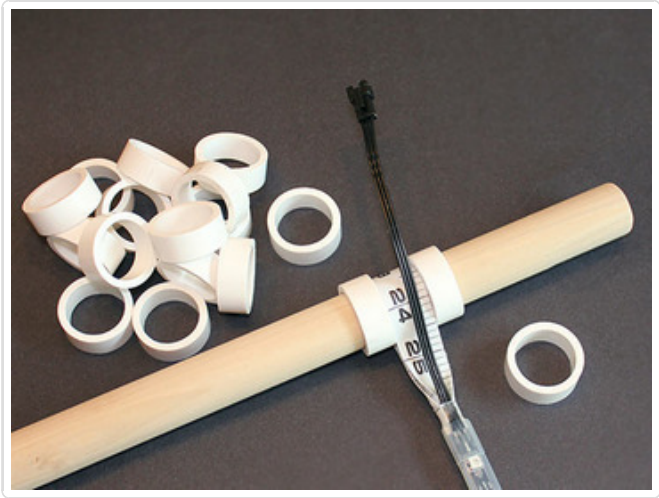
Now to get that jumble of strips off the table and over the doorway…

This is the ***Arts and Crafts*** part of the project! The design shown here is just one of many possibilities. How you approach it will depend on your goals and available tools and materials.

Initially I'd planned to hang the strips from a curtain rod with the electronics exposed on a board above the doorway. After encountering some durability issues, all the hardware was instead moved into a wooden box with top and bottom open for ventilation. It looks a lot nicer with all the strip-hanging hardware and wiring hidden away…just a single power cord protrudes.

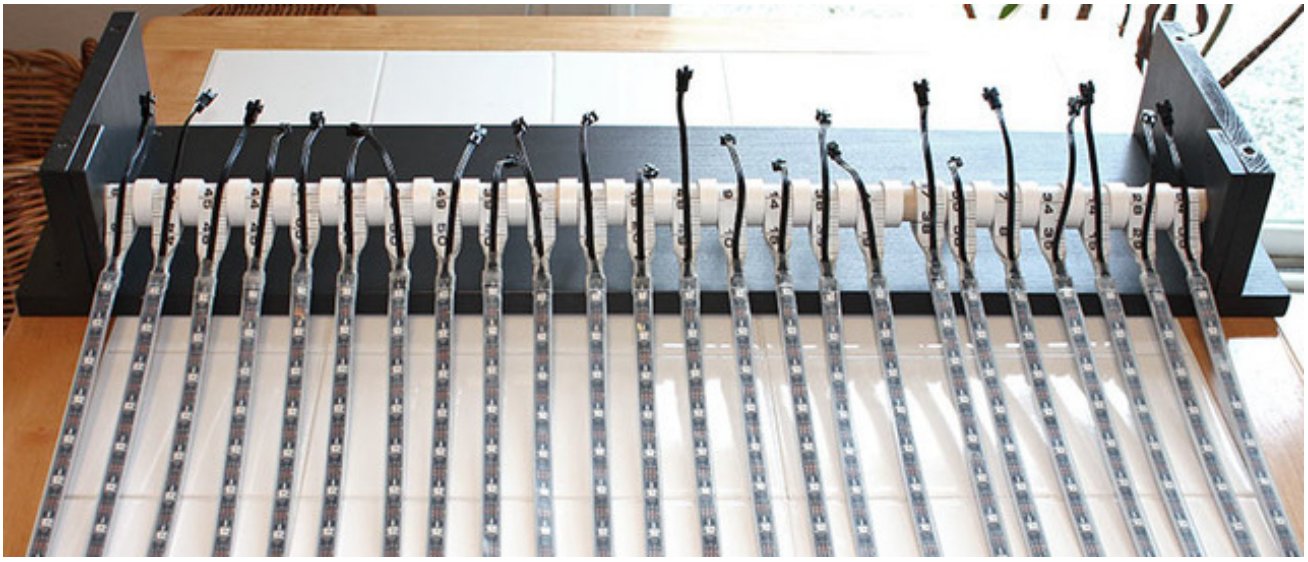
I don't have plans for this box. It's a simple shape that was improvised, with few measurements.

All those strips weigh a lot. About 5 pounds total. A beefy 1" wooden dowel replaced the (now pretzel-shaped) flimsy adjustable metal rod. Holders at the sides allow the dowel to be lifted out for access to strips.

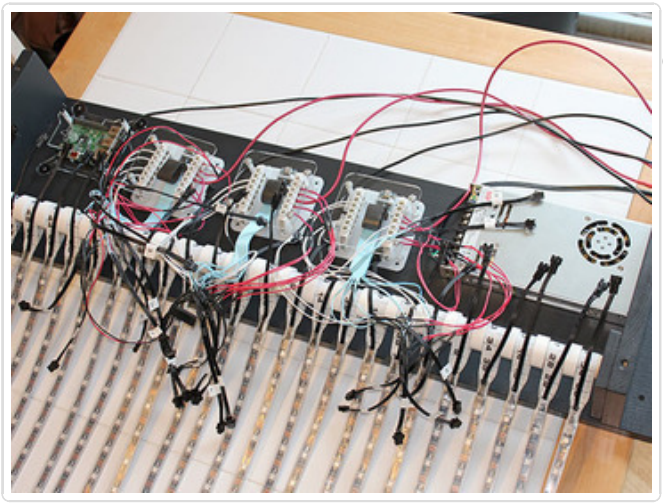


Rings cut from PVC pipe keep the strips uniformly spaced. It was quicker than 3D printing.

Here's all the strips on the rod, hung in the now-painted box:



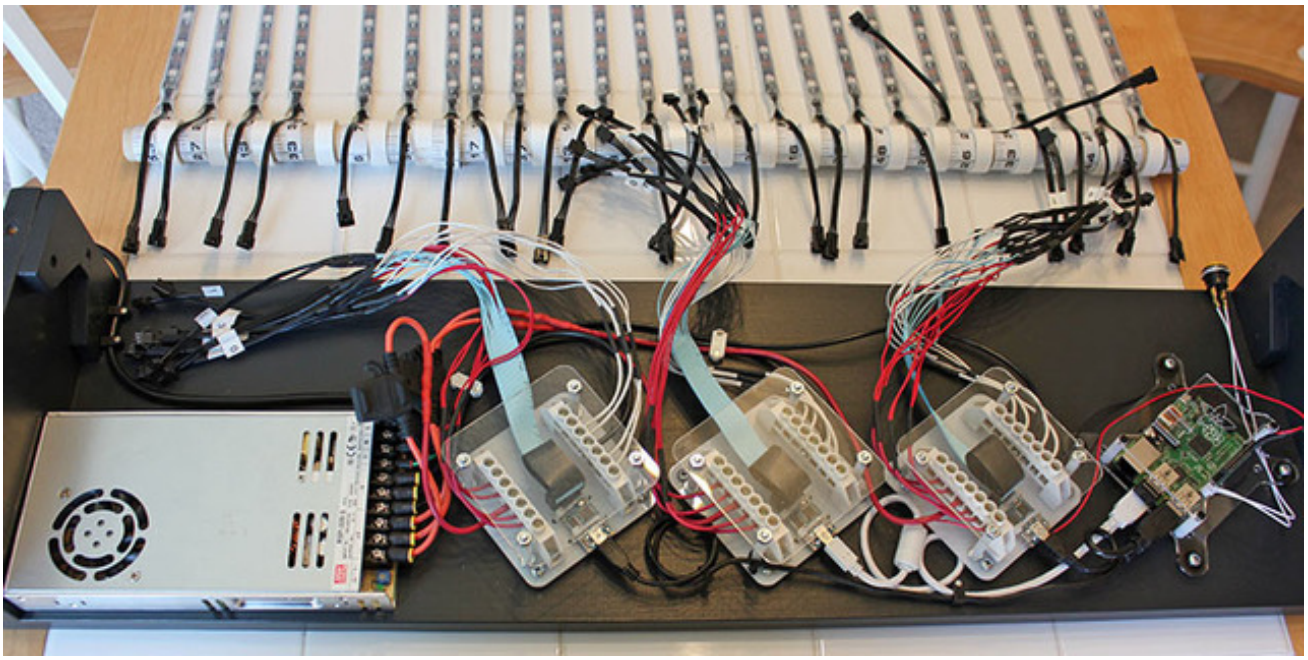
I'd planned to center each "alien facehugger" power/data distribution box over its corresponding group of strips, on the back side of the front face, while the Pi and power supply would be on the facing side. That's one reason for the depth of the box.



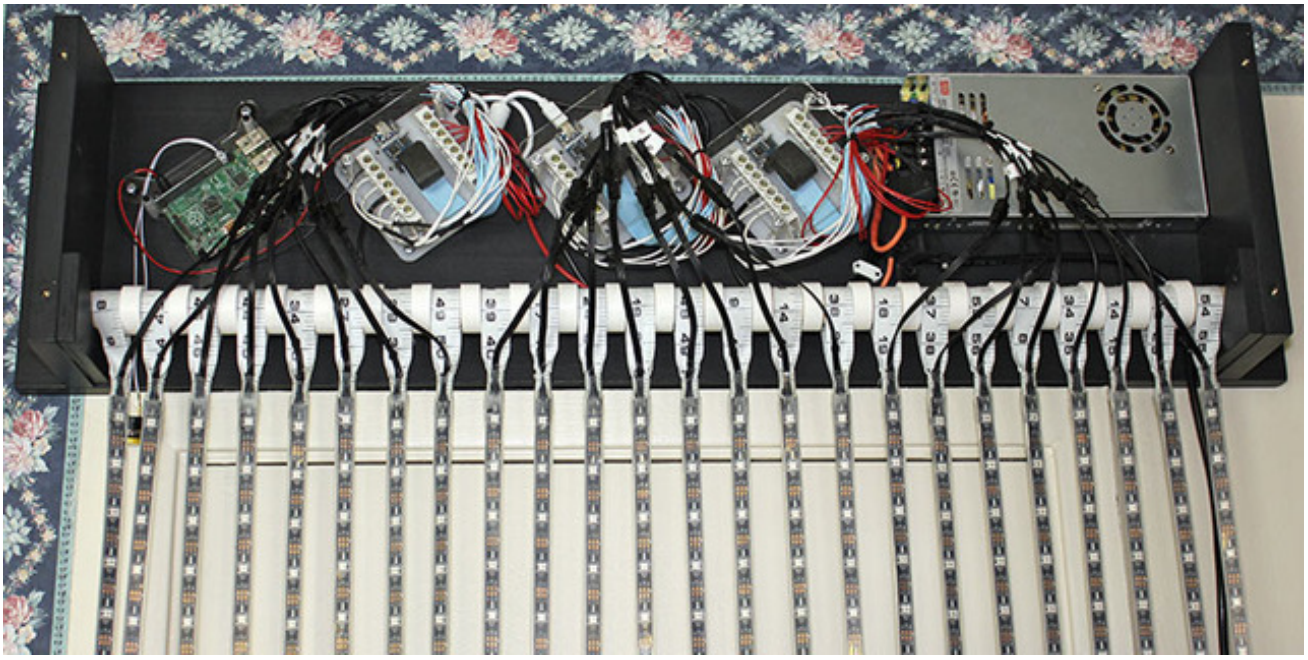
Experimenting with placement, it looked like everything might simply fit on the rear board…

And indeed it would. Some parts were mounted at an angle so the USB plugs wouldn't poke out the top, and for better access to the Pi's ports (in case WiFi is switched for Ethernet later).

Here's most of the parts fastened to the board. The 12-gauge power wires have been trimmed to length and are held with cable clips…



With a friend's help, it's then moved and secured to the wall (using either lag screws or wall anchors)…



It's plugged in and given another test flight before the final wire adjustments and installing the front cover.

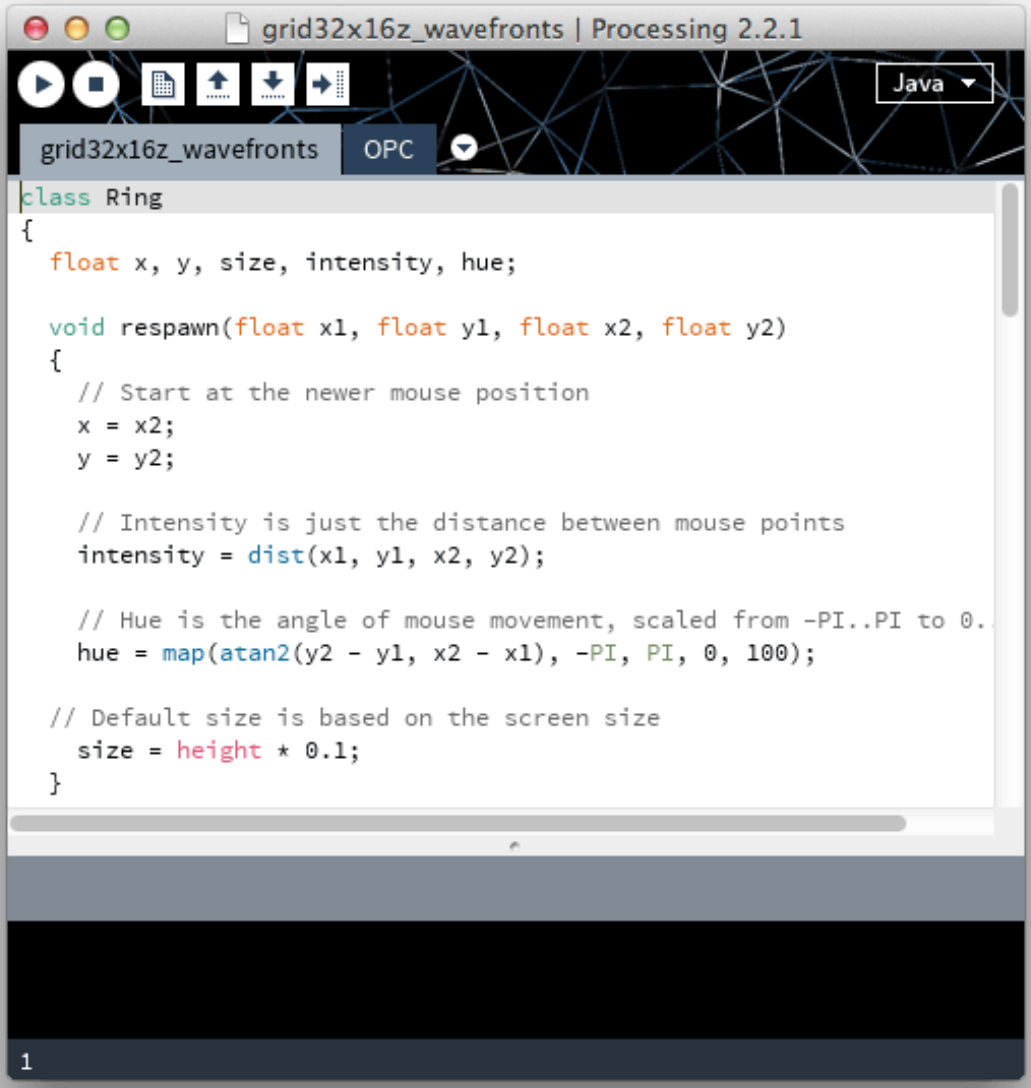
Advanced Clients

Some of the more complex demos are written in *Processing,* a cross-platform multimedia programming environment for Windows, Mac and Linux. If you're not already using Processing…

(http://adafru.it/dOs)Download from Processing.org (http://adafru.it/cK1)

When you unpack and install Processing, you'll notice it looks very similar to the Arduino IDE, and it's easy to get the two confused if they're both running. Unlike the Arduino IDE, which is specifically for microcontrollers, Processing is for writing code for your regular computer.

Let's open one of the example projects. Let's say grid32x16z_wavefronts…



You'll see this sketch has a second tab called "OPC." Every Processing sketch for Fadecandy requires this code for communicating with fcserver; you'll want to copy it into any new Fadecandy sketches you write.

As written, this demo sketch won't work with our LED curtain. It's designed for several 8x8 NeoPixel matrices. But we can rework it quite easily.

Look for this block of code in the setup() function:

```
size(640, 320, P3D);
colorMode(HSB, 100);
```

```
texture = loadImage("ring.png");

opc = new OPC(this, "127.0.0.1", 7890);
opc.ledGrid8x8(0 * 64, width * 1/8, height * 1/4, height/16, 0, true);
opc.ledGrid8x8(1 * 64, width * 3/8, height * 1/4, height/16, 0, true);
opc.ledGrid8x8(2 * 64, width * 5/8, height * 1/4, height/16, 0, true);
opc.ledGrid8x8(3 * 64, width * 7/8, height * 1/4, height/16, 0, true);
opc.ledGrid8x8(4 * 64, width * 1/8, height * 3/4, height/16, 0, true);
opc.ledGrid8x8(5 * 64, width * 3/8, height * 3/4, height/16, 0, true);
opc.ledGrid8x8(6 * 64, width * 5/8, height * 3/4, height/16, 0, true);
opc.ledGrid8x8(7 * 64, width * 7/8, height * 3/4, height/16, 0, true);
```

Change the size() call to match the aspect ratio of our curtain (24 columns wide, 60 rows tall). We'll scale it up by a factor of 10 so the window is easier to see:

```
size(240, 600, P3D);
```

Change the OPC() call to contact the Raspberry Pi across the network:

```
opc = new OPC(this, "curtain.local", 7890);
```

And all the ledGrid8x8() calls are replaced with 24 vertical ledStrip() calls. We'll use a loop for brevity:

```
for(int i=0; i<24; i++) {
  opc.ledStrip(i * 60, 60, i * width / 24.0 + width / 48.0,
    height * 0.5, width / 24.0, PI * 0.5, false);
}
```

The parameters to opc.ledStrip() are:

1. The index of the first pixel in the strip. Because of the map we set up in fcserver.json, and the length of the strips, this is equal to the column number (i, ranging from 0 to 23) times the strip length (60 pixels).
2. Strip length, 60 pixels.
3. In window coordinates, the centerpoint of the strip on the X (horizontal) axis. For our curtain, this is equal to the column number (0 to 23) multiplied by 1/24 of the total window width in pixels (the variable 'width' in Processing). The additional half column (+0.5) is to center the full set of columns in the window, so they're not pressed up

along the left edge.
4. Centerpoint of the strip on the Y (vertical) axis. Our vertical strips will always be centered along that direction, hence height * 0.5.
5. Spacing between LEDs, in pixel units. With 24 columns, this is width / 24.0.
6. Angle of the strip, in radians. For vertical strips with the first pixel at the top, this is PI * 0.5(90 degrees).
7. false = don't reverse the order of the LEDs.

Before running this sketch, close the browser window running the *ganzfeld* test; the two programs will interfere.

When you move the mouse over the *wavefronts* window, you'll get some colorful ripples that follow the cursor. The animation should be mirrored on the LED curtain.

No response from the LEDs, or some sections aren't lighting!
Use the same troubleshooting process as for the web example above.

It's flickering madly!
Close the browser window running the *ganzfeld* test; the two programs will interfere.

The brilliance of the Fadecandy OPC code is that it's easily incorporated into existing Processing sketches. There's already *tons* of visual effects code out there.

Here's a Processing sketch for playing video. Select a movie file (AVI, MOV, etc.) and it'll crop and scale a 24x60 vertical box from the center. Remember to copy the contents of the OPC tab from another Fadecandy sketch to use this.

```
// Processing 2.X movie example for
// Adafruit NeoPixel/Fadecandy LED curtain

import processing.video.*;

OPC           opc;
Movie         movie;
PGraphics     g; // Offscreen buffer for scaling
DisposeHandler dh;
int           xres = 24, yres = 60, scale = 10, mx, mw;

void setup() {
  size(xres * scale, yres * scale, P2D);
  g = createGraphics(xres, yres, P2D);

  opc = new OPC(this, "curtain.local", 7890);
  dh  = new DisposeHandler(this);
```

```
  for(int x=0; x<xres; x++) {
    opc.ledStrip(x * yres, 60, (x + 0.5) * scale,
      height * 0.5, scale, PI / 2.0, false);
  }

  selectInput("Select a file to process:", "fileSelected");
}

void fileSelected(File selection) {
  if(selection == null) exit();
  movie = new Movie(this, selection.getAbsolutePath());
  movie.loop();
  mw = (int)((float)yres * ((float)movie.width / (float)movie.height));
  mx = (xres - mw) / 2;
}

void movieEvent(Movie m) {
  m.read();
}

void draw() {
  if(movie == null) {
    background(0);
  } else {
    g.beginDraw();
    g.image(movie, mx, 0, mw, g.height); // movie->buf scale/crop
    g.endDraw();
    image(g, 0, 0, width, height); // buf->window scale
  }
}

public class DisposeHandler { // LEDs off when exiting
  DisposeHandler(PApplet pa) {
    pa.registerMethod("dispose", this);
  }
  public void dispose() {
    for(int i=0; i < xres * yres; i++) opc.setPixel(i, 0);
    opc.writePixels();
  }
}
```

The *DisposeHandler* section takes care of turning off the matrix when the program exits. You may want to copy this into your own Fadecandy sketches.

Curating videos, selecting material that looks good on the vertical curtain can be time-consuming; not everything frames well in this format. The stargate segment in *2001: a*

Space Odyssey looks great though.

For Power Users...

You can write Fadecandy applications in almost any language...anything that provides access to TCP sockets...you just need to follow the Open Pixel Control protocol, as detailed in the Fadecandy documentation. (http://adafru.it/cWf) Or you can pick apart the OPC code in Processing.

Python and C both perform quite well on the Raspberry Pi. An expert programmer could make this a self-contained system, not reliant on a networked computer to drive the animation.

Care and Feeding

The NeoPixel curtain works best as a wall decoration or for *infrequent* doorway use; it probably won't hold up to "main entrance" traffic.

Although NeoPixel strips are flexible, they don't like harsh or repeated bending. Care should be taken to avoid strong pulling, or pinching underfoot or in closing doors.

If you do lose a strip, have replacements ready. You may be able to repair the damaged strip later, but it's hard to do in the field. Also have a supply of fuses compatible with your power system…I put several in a Ziploc baggie, pinned inside the box so they're never misplaced.

Occasionally wipe the strips down with a damp washcloth to remove any dust, so they slide over things more easily. Also periodically blow out the power supply; don't let dust bunnies accumulate.

Remember to shut the system down correctly before pulling power, otherwise you may corrupt the SD card. You may want to make a backup image just in case.