Supernova Poi

Created by Erin St Blaine

https://learn.adafruit.com/supernova-poi

Last updated on 2024-06-03 01:52:14 PM EDT

Table of Contents

Introduction

Dotstar LEDs are super fast and super bright -- they're the best LEDs available for drawing persistence of vision images or light painting. This guide combines Dotstars with a Teensy 3.2 -- a super fast Arduino-like board with 256K Flash Memory and 64K RAM. The Teensy has enough space for hundreds of images and patterns, so with a little time and a little imaging knowhow, these poi can go all night without ever repeating a pattern.

We even added IR remote support so you can change programs with the press of a button!

Materials

To make one poi (get two of everything for two poi):

- 1/2 m 144 LED/m Dotstars
- Teensy 3.2
- Pro Trinket Lion/LiPoly Backpack Charger
- Lithium Ion Cylindrical battery - 3.7 v 2200mAh
- On/off switch - SPST Submini Slide Siwtch from Radio Shack (https://adafru.it/sB1)
- IR receiver sensor
- Mini remote control

Housing

- 14" x 1" polycarbonate tube (be sure inner diameter is **at least** 7/8")
- 2 3d-Printable End Caps (or order them online here) (https://adafru.it/sAZ)
- 8" x 5/8" wooden dowel
- Poi leash & handle (I like the ones from FlowToys (https://adafru.it/10br))
- Holographic sticky-backed vinyl (https://adafru.it/19sb) (optional, of course, but.. SO SPARKLY)

Tools

- Soldering iron & accessories
- Drill and Rotary tool or saw for cutting tubing & dowels to length
- Hot glue gun
- E6000 glue
- Micro USB cable for programming & charging

Code

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

To get the code running on the Teensy you'll need:

1. Arduino IDE (1.6.5 or newer preferred)
2. Teensyduino Installer
3. Teensy's version of the IRremote Library
4. Adafruit Dotstar Library
5. Kinetic POV Code

1. Arduino IDE

If you're not using a recent version of the Arduino IDE (1.6.5 or newer), this would be a good time to upgrade (https://adafru.it/fvm).

2. Teensyduino Installer

Once you have that software installed and working, download and run the Teensyduino installer (https://adafru.it/iwF), which adds support for the full line of Teensy microcontroller boards in the Arduino IDE.

In the Arduino IDE, from the **Tools** menu, select **Board→Teensy 3.2** and **CPU Speed→72 MHz (Optimized)**. Confirm that you can compile and upload the classic "blink" sketch to the Teensy board. Be sure you can blink the light before continuing.

3. Teensy's Version of IRremote Library

The IRremote library is a really popular way to add IR support to arduino projects. The folks who make the Teensy have updated it to work with Teensy 3.1-3.2, so you'll need to be sure you have the right version of IRremote.

This library can be fetched from PJRC.com (https://adafru.it/lgD) and installed manually (https://adafru.it/dNR).

4. Adafruit Dotstar Library

This project also requires 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 (https://adafru.it/eio) and installed manually (https://adafru.it/dNR).

5. Kinetic POV Code

Software for the Supernova poi project can be fetched from GitHub (https://adafru.it/E-O):

Click to download Kinetic POV archive for Arduino

https://adafru.it/E-O

The "supernova_poi" folder contains the Arduino sketch for this project. The "convert" folder contains a utility for processing images — we'll cover that on a later page. The other folders can be ignored — they're for other projects that evolved from the same code base.

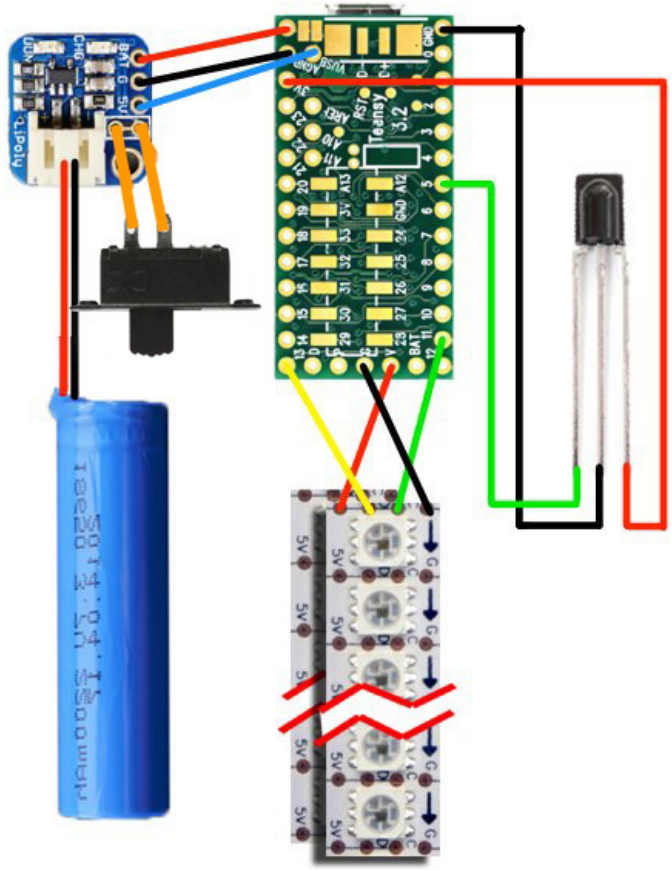
There are two files in the "supernova_poi" folder, which will open as two tabs in the Arduino sketch. The second file/tab — graphics.h — contains the bitmaps and color palettes for the different modes. We'll explain how to add different ones later on.

I can compile the code but it won't upload to the Teensy!

You might have a "charge only" USB cable. Definitely need the normal "charge+data" type for this. Switch it out for a different cable and try again.

Also, be sure you've selected "Teensy 3.2" from the tools dropdown.

Wiring Diagram

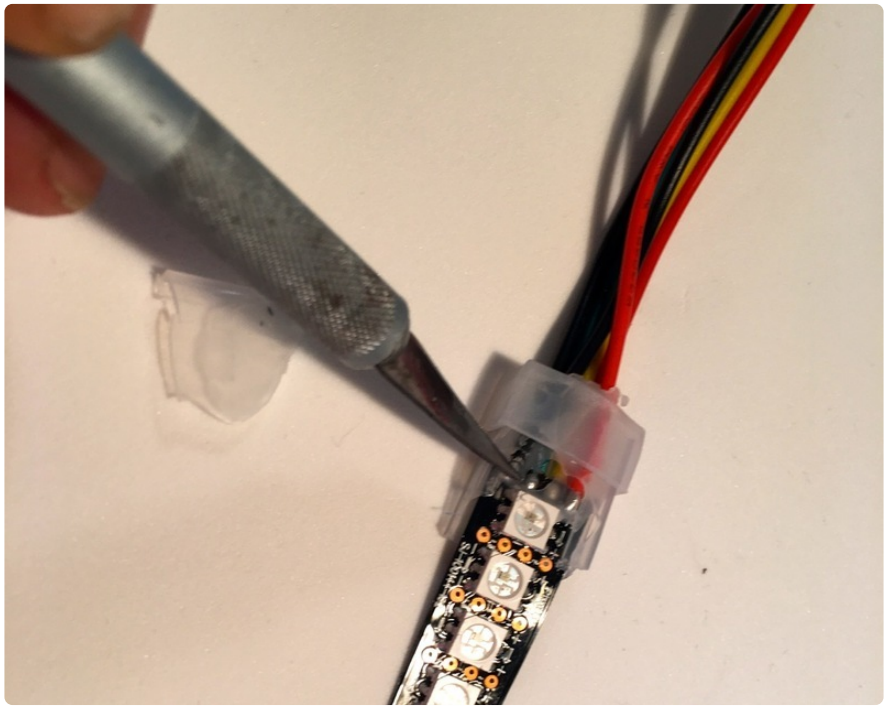


This is a schematic layout of the parts, to clearly show all the connections, and not representative of the actual placement of all the parts or wires. We'll detail that in the following pages.

The Teensy 3.2, charger, and IR sensor will be at one end of the poi, and the battery and on/off switch will be at the other end.

It's best to place the battery at the bottom end of the poi, since it's the heaviest part and will add a very nice heft and swing to your poi.

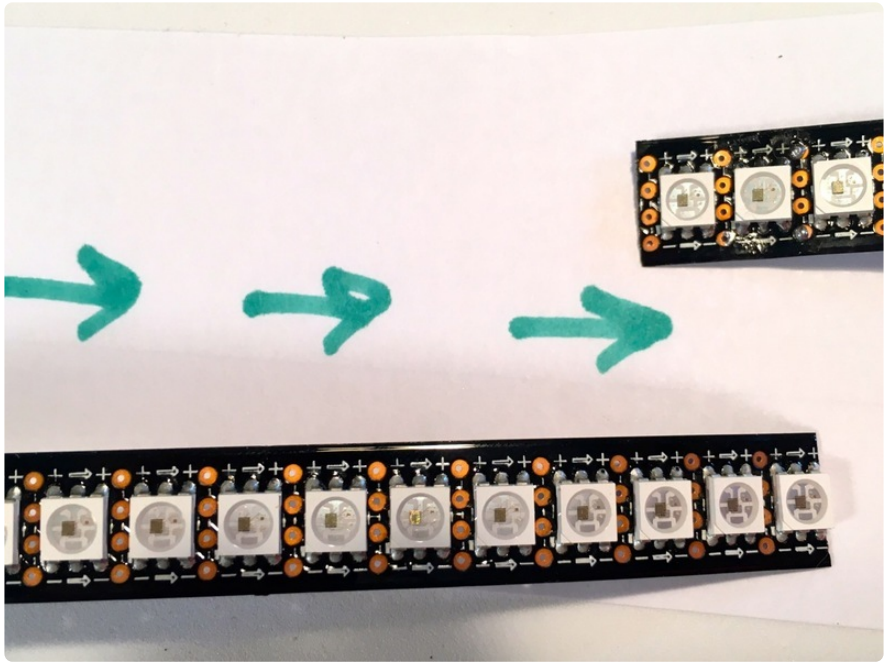
Layout & Prep



Prep the LEDs

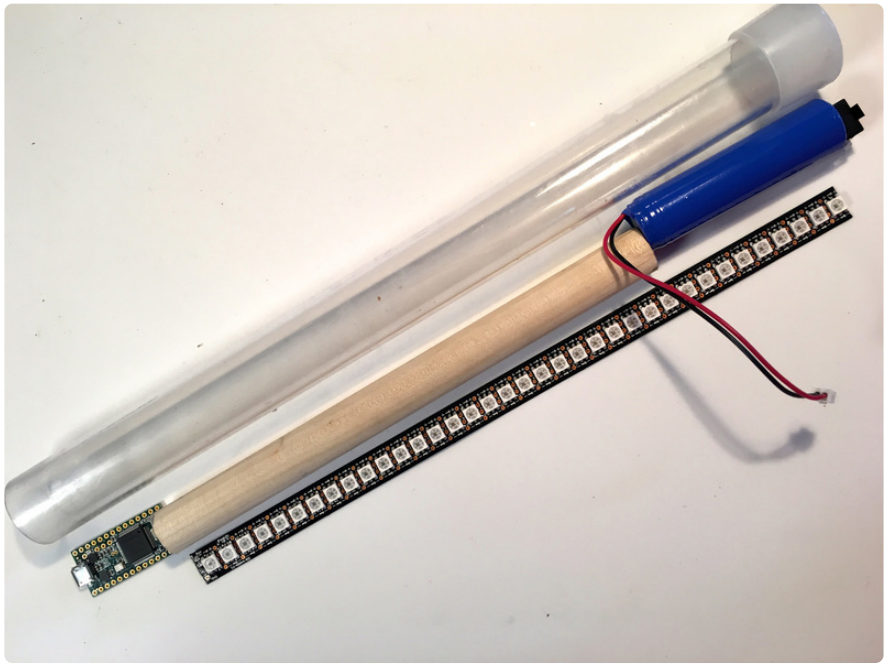
Use a utility knife to carefully remove the silicone sheathing and most of the covering and glue from the ends of your 1M Dotstar LED strip where the connector is soldered on. Get the ends as clean and tidy as you can. De-solder the wires that came with your Dotstar strip, we'll be using more flexible wires instead.

Go to the "in" end and count out 36 pixels. Count them once more. Then, with tiny scissors or a utility knife, carefully cut between the pixels. Cut very close to pixel #36, leaving the copper pads on the side that now starts with pixel #37 (this is the "in" side and you'll need to solder on this end).



Tubing & Caps

Lay out your battery and Teensy and switch alongside your LEDs and tubing to visualize how everything should be sized. You want the switch all the way at one end of the tube, and the USB port on the Teensy all the way at the other end. A tight fit will help make sure your poi don't slip around and get damaged as you spin them.



Cut the dowel and tubing to the right length, leaving a little room to attach the leashes. My dowels ended up at 7 3/8" and my tubes came to 12".

End Caps

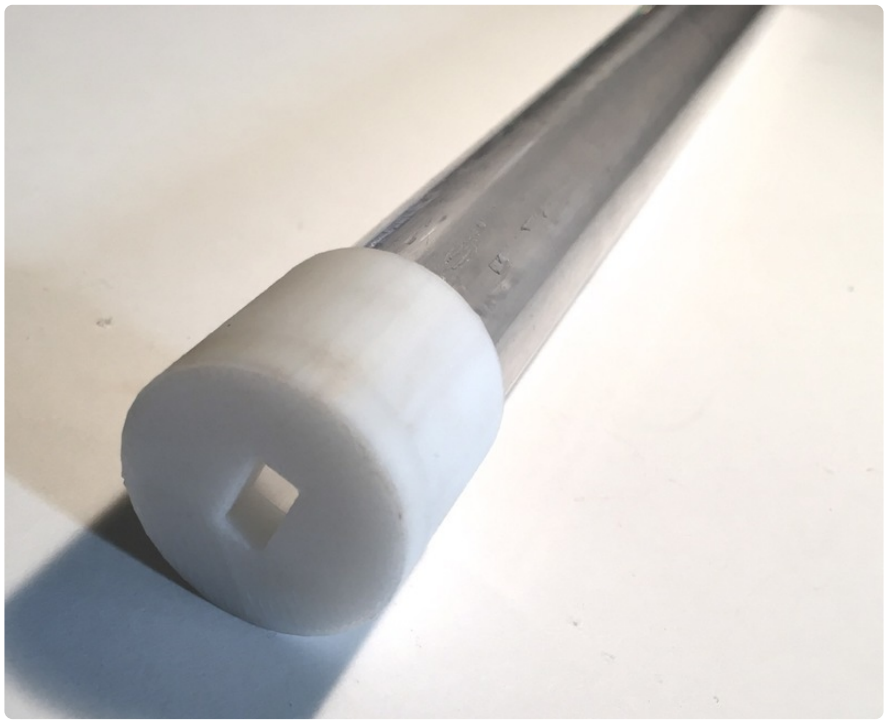
The end caps are designed to sit flush with the top of the on/off switch so the poi won't get accidentally turned on or off in the switch gets bumped while you're spinning.

You can print them in either ABS or PLA -- the PLA is slightly stronger and will hold up a little bit better to whacks and bumps.

Download .stl from Thingiverse

https://adafru.it/sAY

If you don't have access to a 3-d printer, you can order the caps from Shapeways (https://adafru.it/sAZ).



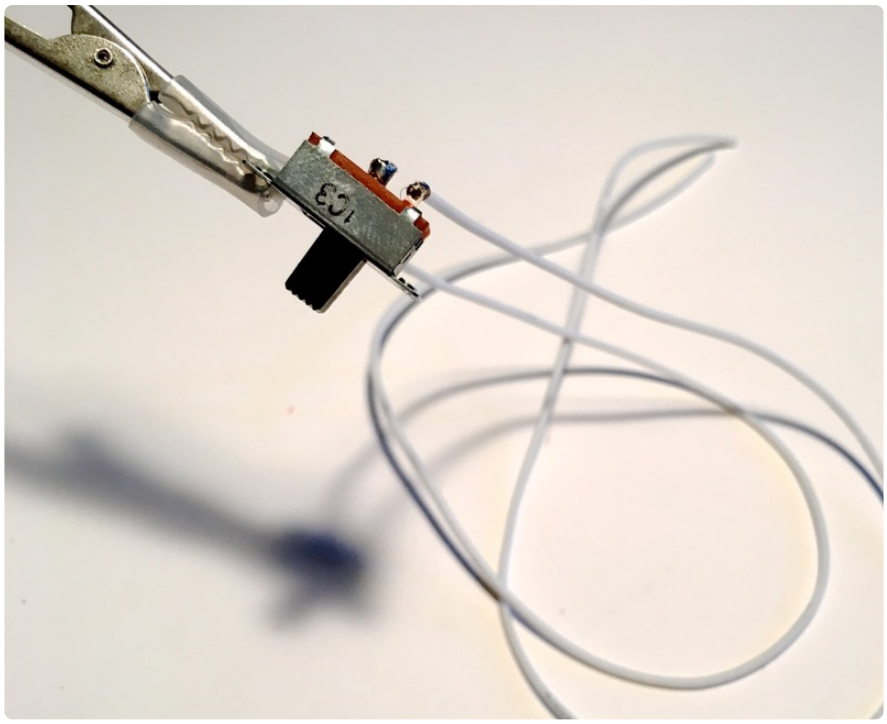
For the leash end, drill a couple of holes in the tubing about 1/2" from the end. Thread your leash through the holes and tie it off securely. Melt the end of the cord to ensure that knot will never slip.

Assembly: Wiring

I'm using 26awg (http://adafru.it/1880006328647588) (power & ground wires) or 30 awg silicone-coated wire (http://adafru.it/2051) (for data lines) for all steps. This wire is super flexible, heat-resistant, easy to use, and very hard to break. It makes the wiring on this project much easier using than traditional wire.

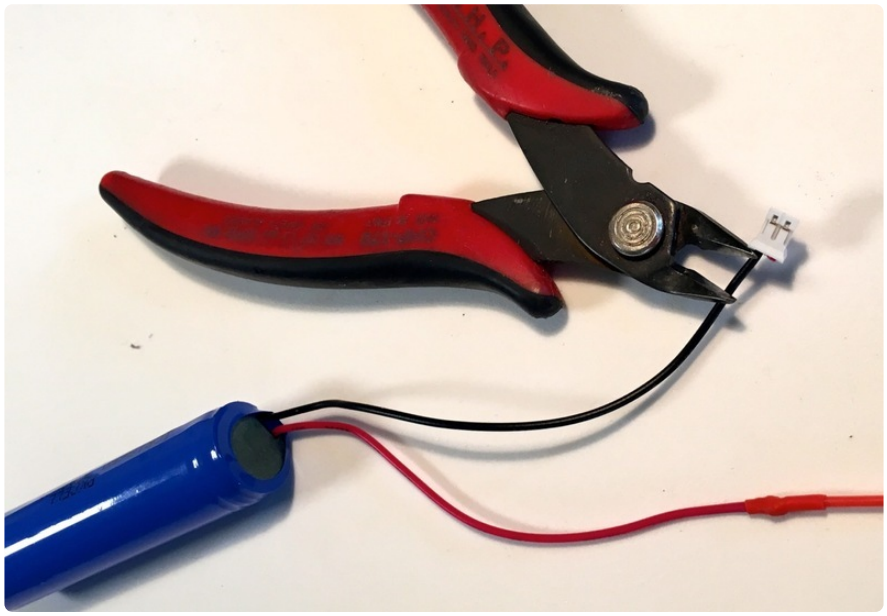
Color-coding the wire is very helpful too, as the wiring can get a little crazy otherwise. I'm going with the color-coding as I found it on my Dotstar strips. Manufacturing sometimes changes these colors around, so notice if your wiring colors are different and adjust accordingly.

For me: Power (5v) is red, G is black, Clock is yellow and Data is green. White is for the power switch.



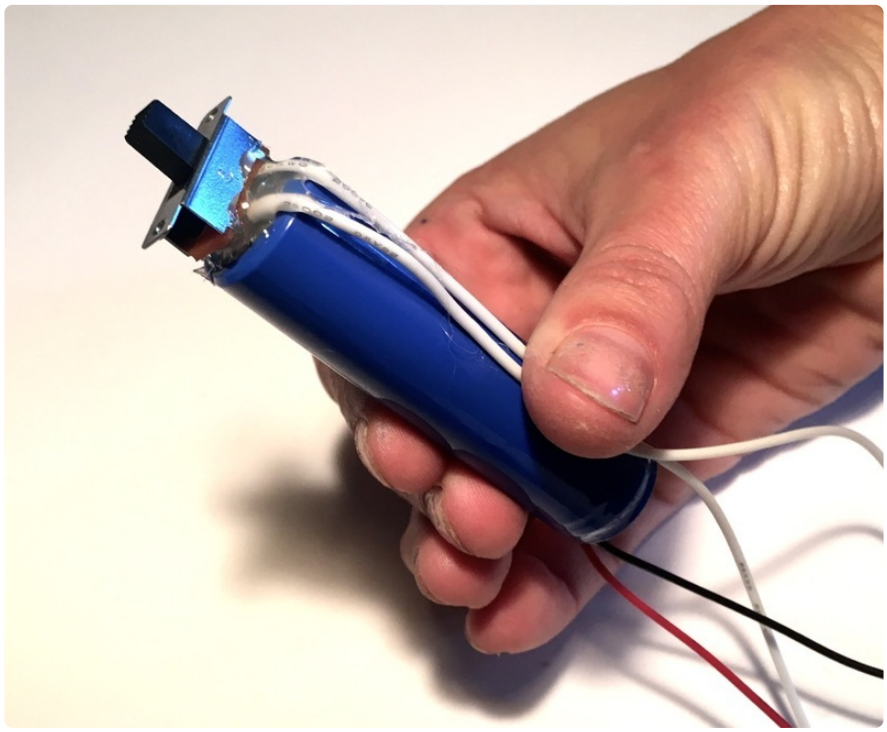
Battery & Switch

Cut 2 white wires, one red wire and one black wire to just over a foot in length. Solder the white wires to the two legs of the on/off switch. Be sure to go through the tiny holes with your wire instead of just around -- a tight physical connection will be much stronger and more secure than just soldering.



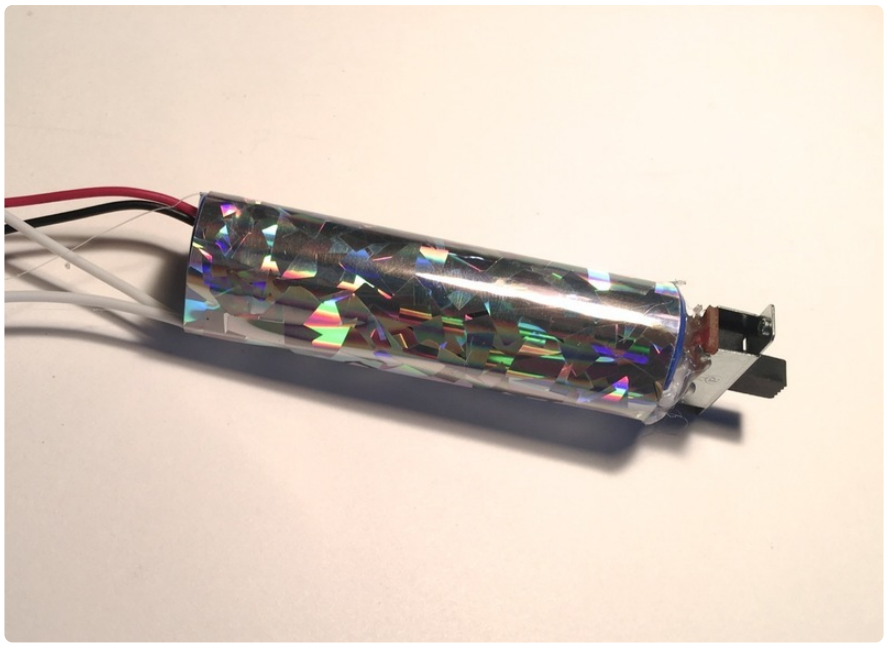
Carefully cut the red wire coming from the battery, and splice your red wire on to the battery's power wire. Once you've finished soldering and heat-shrinking, do the same with the black wire.

DO NOT cut both these battery wires at once! You might accidentally short the battery with your wire cutter, and could damage your battery or your body. These battery-things are filled with powerful lightning. Respect them.



Place the switch on the bottom (non-wired) end of the battery. Line it up so the white wires are on the opposite side of the cylinder as the battery wires. Secure the switch with a dab of hot glue.

Cut a piece of sparkly vinyl and wrap the battery, being sure the wires are spaced evenly down two sides and not crossing. The battery with the LEDs will be a tight fit inside the tube, so get the vinyl and wires wrapped down as tightly as possible to make a slim, sparkly little package.



Hot glue the dowel to the other end of the battery. Wrap the dowel and wires in another piece of sparkly vinyl, keeping the wires as tidy as possible and making sure they stay on opposite sides.

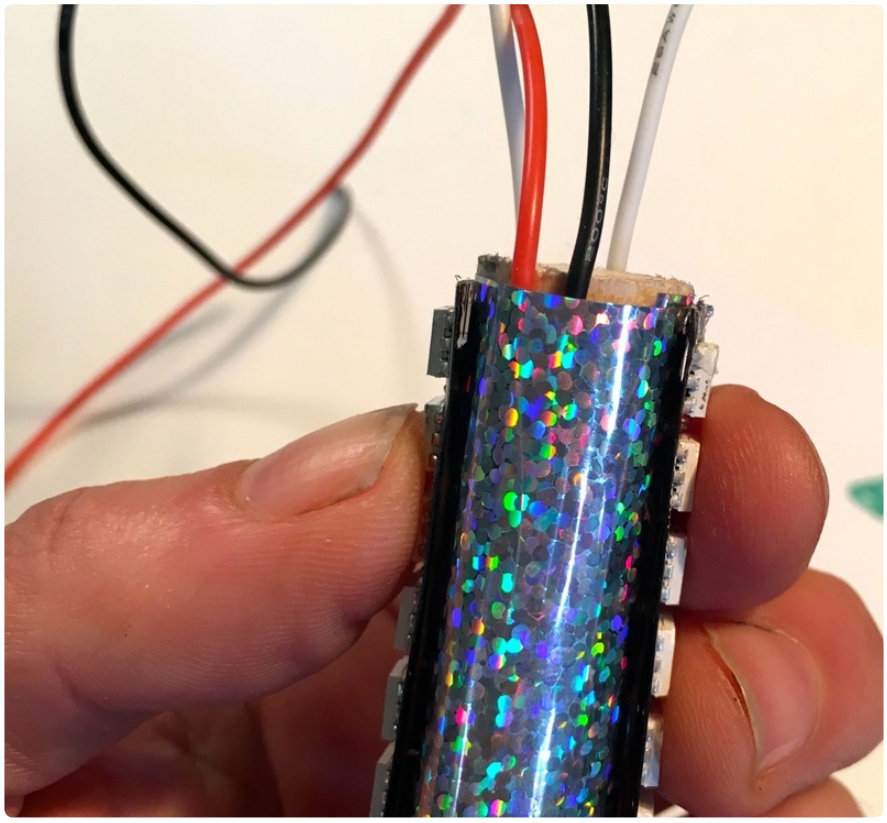
Wiring Everything Together

The Teensy needs to connect to 3 things: the LED strips, the battery charger, and the IR sensor. We'll solder 3 sets of wires into different areas on the Teensy, one for each of these connections.

LED Strips - Prep

Find the "in" end of the LED strips and put them at the opposite end from the battery. Glue the strips down along the sides without wires. Be sparing with the glue -- try not to add any bulk or your battery won't fit inside the tube.

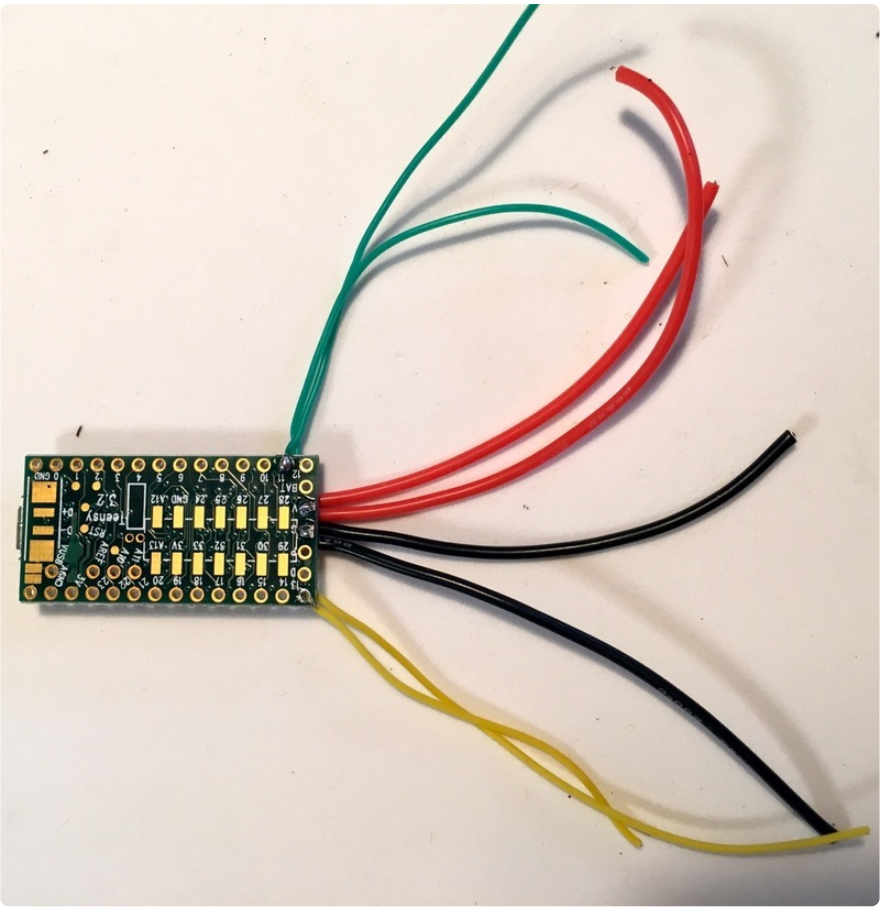
Line the LEDs up as perfectly as you can. A misalignment will make for a blurry picture when the poi are spun. Take your time and get this right.



LED Strips: Teensy Wiring

Put the poi assembly aside for now and grab your Teensy. The wires to the LED strips will be soldered into the holes near the bottom of the Teensy, since that's where they'll line up inside the poi. Since we have two LED strips, solder **two wires each** into the G (ground, black) and 3.3 v (power, red) holes along the Teensy's bottom edge, and **two wires each** into holes 11 (data, green) and 13 (clock, yellow).

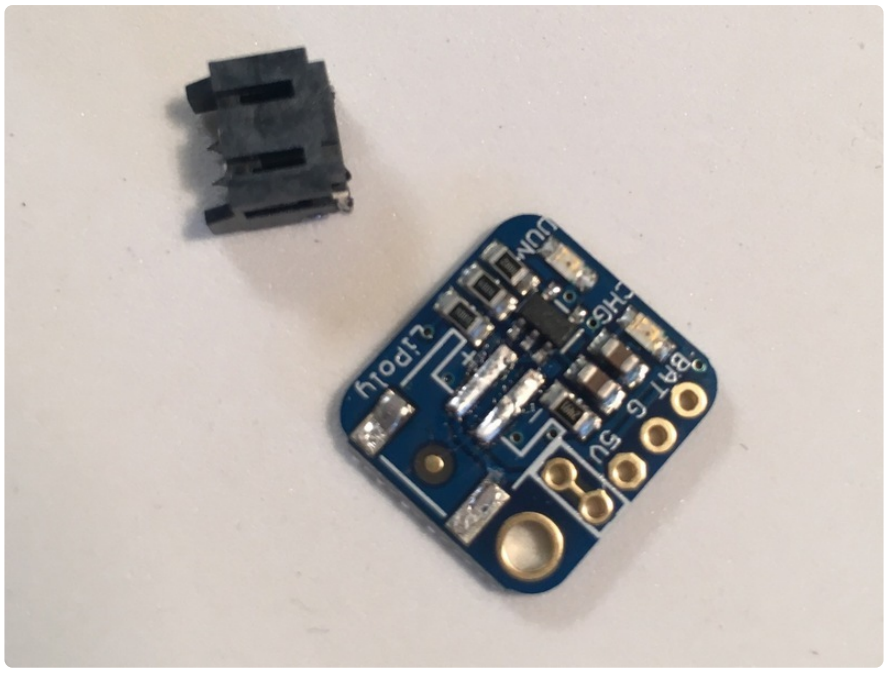
Don't solder the other end of the wires to the LEDs just yet. It's easier to get everything in place on the Teensy first.

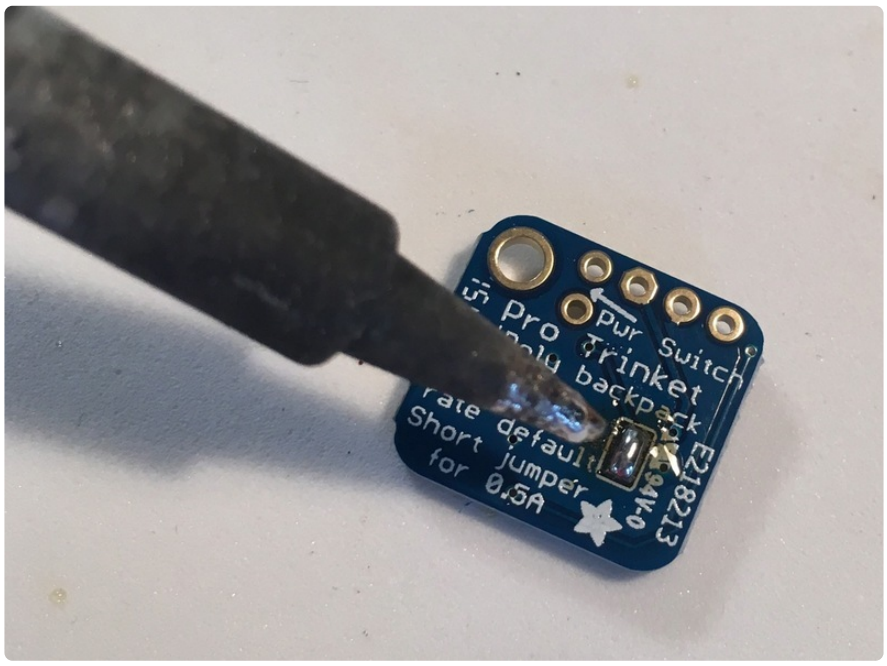


Battery Charger - Prep

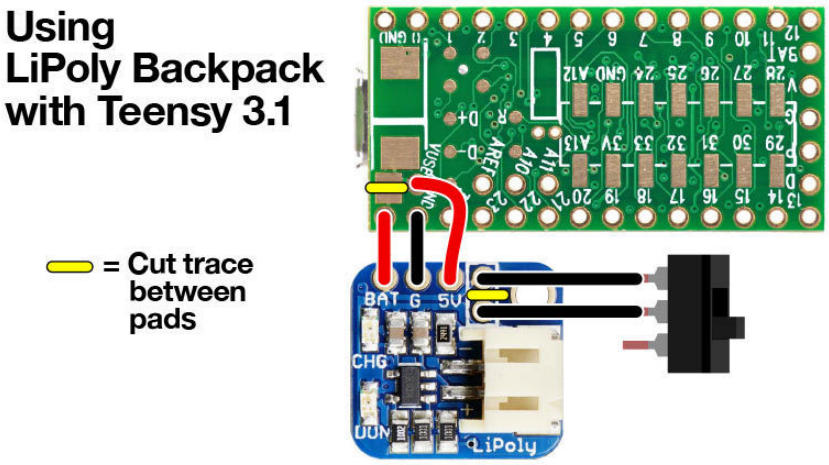
Clip the JST battery connector off the LiPoly backpack and clean up the + and - pads with your soldering iron.

Flip the charger over and look at the back. There's a little silvery patch. Take your soldering iron and bridge the two pads here -- this will make your poi charge a lot faster.





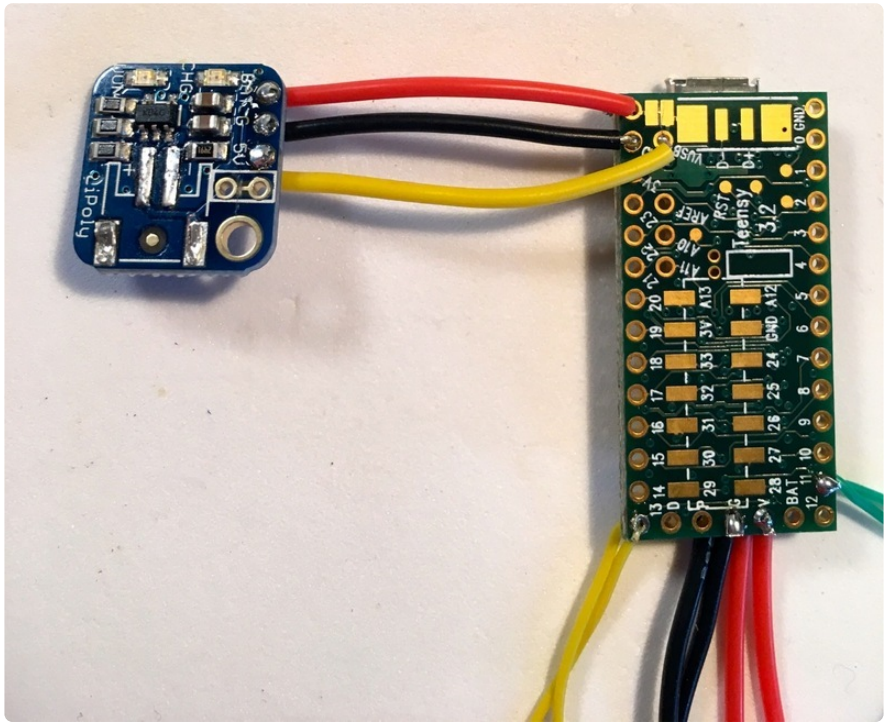
To use the Teensy board with the Adafruit LiPoly Backpack (allowing USB charging), first **two copper traces need to be cut:** between the two pads next to the Teensy's VUSB pin, and between the switch pads on the LiPoly Backpack (marked on back).



Battery Charger: Wiring

Add these three wires between the boards:

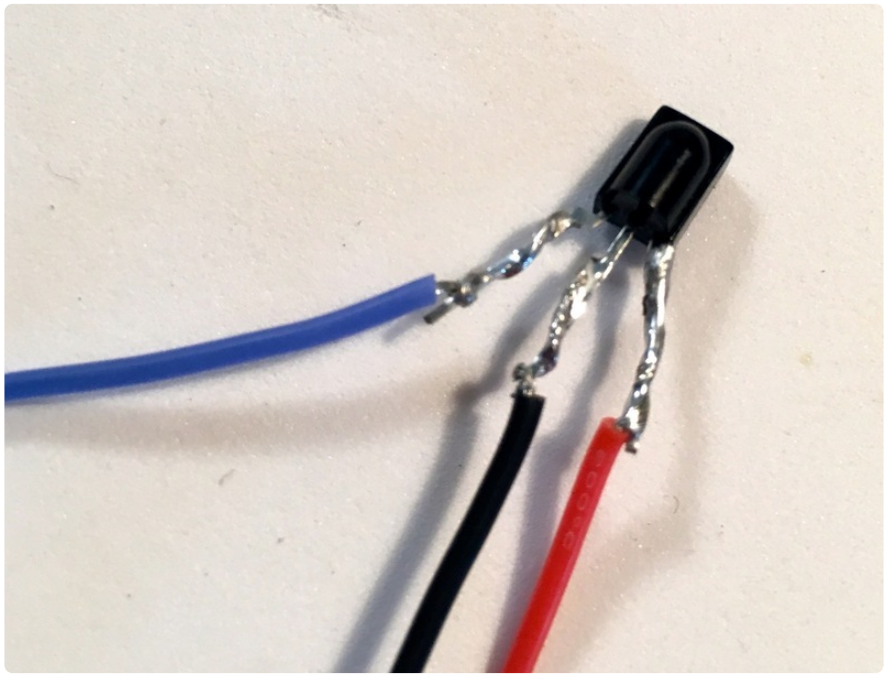
- LiPoly **BAT** to Teensy **VIN/BAT+** (unmarked pin at corner)
- LiPoly **G** to Teensy **GND**
- LiPoly **5V** to Teensy **USB+** pin



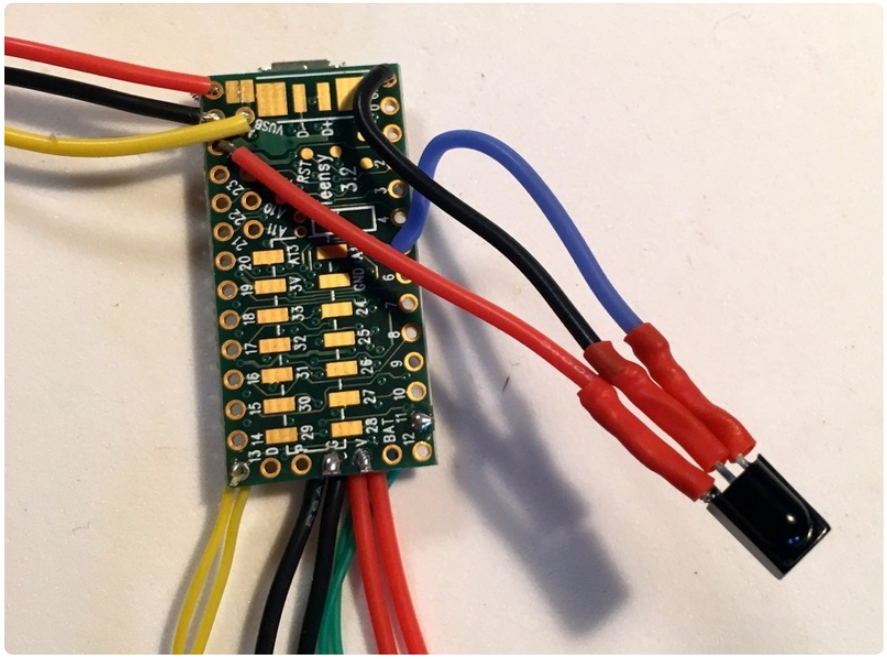
IR Sensor: Prep

Trim the leads on the IR sensor down to about half their length.

Strip about 1/2 inch (yes, that's a lot!) of shielding from a blue, red, and black wire and slip a large piece of heat shrink onto each wire.

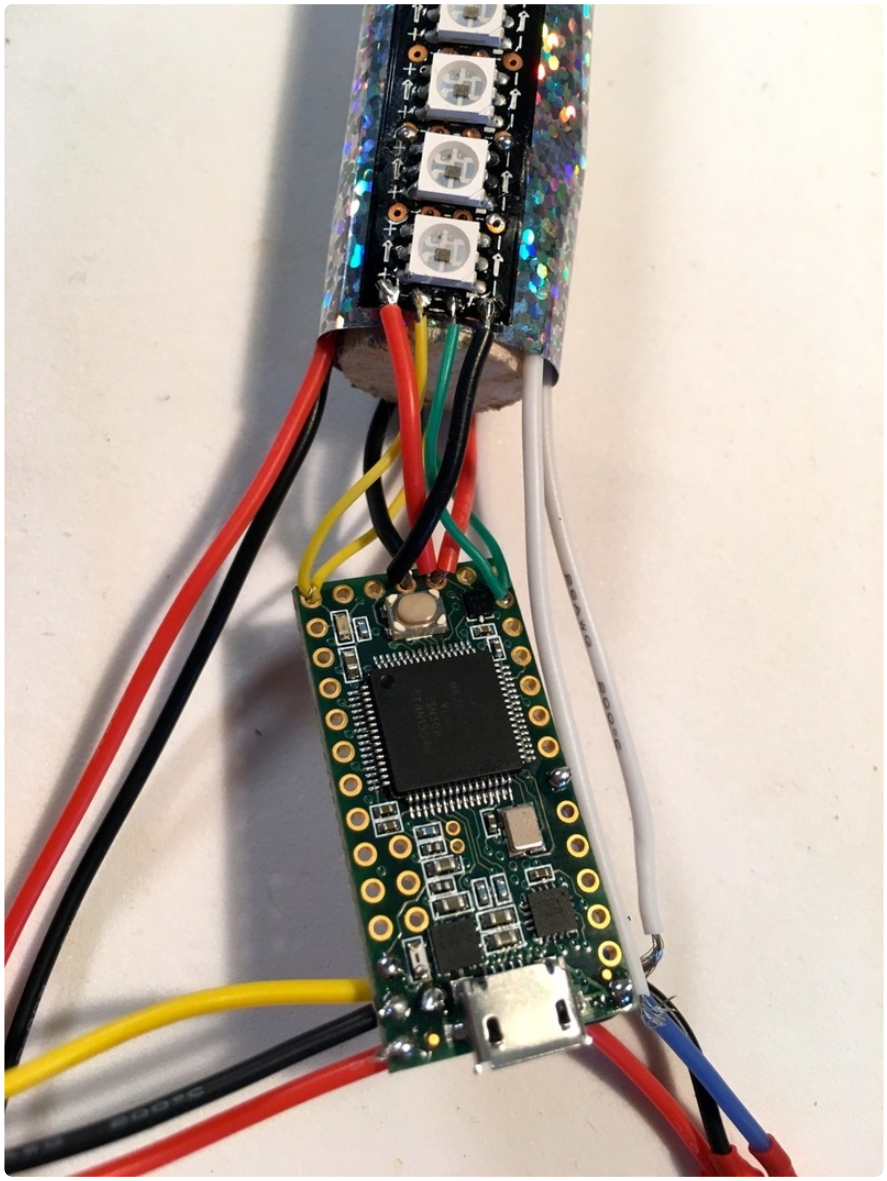


As you're looking at the sensor with the bump facing you and the legs down, coil the blue wire around the left leg, the black wire around the center leg, and the red wire around the right leg. Solder and secure with heat shrink.



IR Sensor: Wiring

Solder the black wire into the remaining G pin on the Teensy, the red wire into 3.3V, and the blue wire into pin 5.

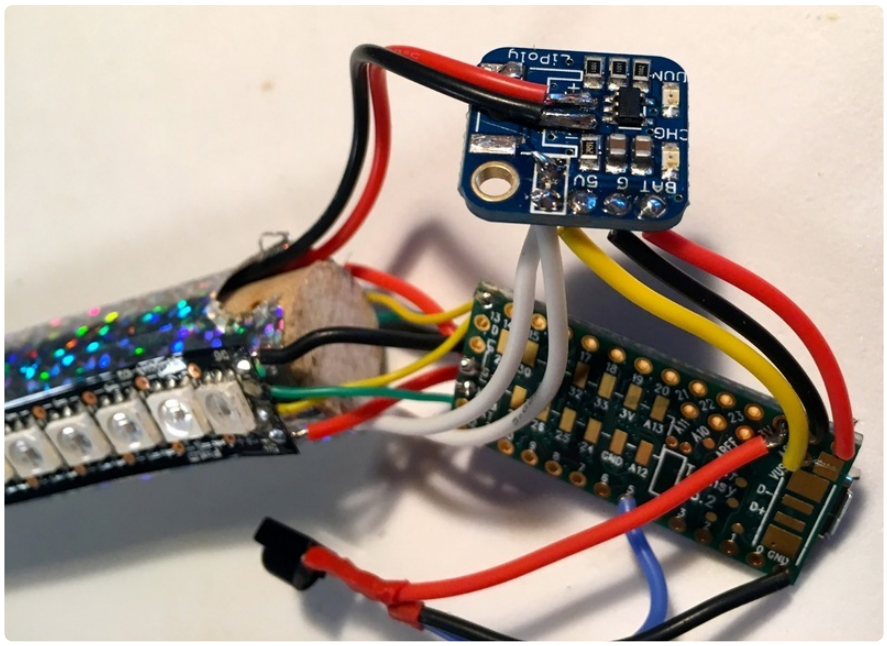


Solder the LED Strips

Place the Teensy next to your LED strip assembly. Trim all the remaining LED wires and solder:

- Teensy G > Dotstar G
- Teensy 3.3v > Dotstar VIN
- Teensy 11 > Dotstar Data
- Teensy 13 > Dotstar Clock

Note: some versions of the Dotstar LEDs have the pins laid out in a different order, so double check your strip layout before soldering.



Battery & Switch Connection

Connect the two white wires from the switch to the two power switch holes on the battery charger.

Finally, carefully solder the two battery wires onto the + and - pads on the battery charger. If you see lights come on while you're soldering, click your switch once to turn the battery off.

Once all the wires are soldered into place, press your power button to turn the poi on. Press the left and right buttons on the remote control to be sure the IR control is working.

Do your best to resist the compelling urge to fling your proto-poi around through the air at this point to see the images.

Troubleshooting

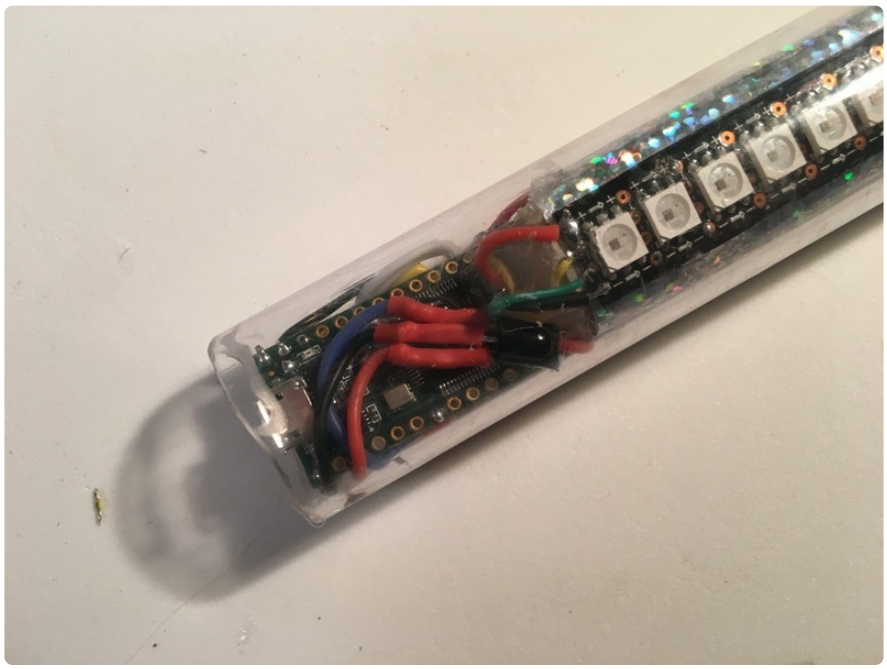
If the LEDs don't come on, here are a few common mistakes to check for:

1. Plug the poi in to the USB cable and be sure the battery has some charge and didn't get shorted while you were soldering
2. Make sure your clock and data wires are not cross-wired to the opposite pads
3. Be sure the traces on the battery charger and Teensy have been carefully cut and you didn't scratch through anything critical
4. Try re-soldering the Dotstar wires using the pads on the back of the strip instead of the front. I had more success soldering the data wires to the back of the strip and the power and ground wires to the front.

5. If you fry the first pixel in line, you can cut just that pixel off and try soldering to the next one in line. The poi will still look fine with only 35 pixels on one side and 36 on the other, so don't worry!
6. Re-upload the code to your Teensy
7. Try uploading the Dotstar strandtest from the dotstar library to see if it's a soldering issue or a code / board issue

---

Case Assembly



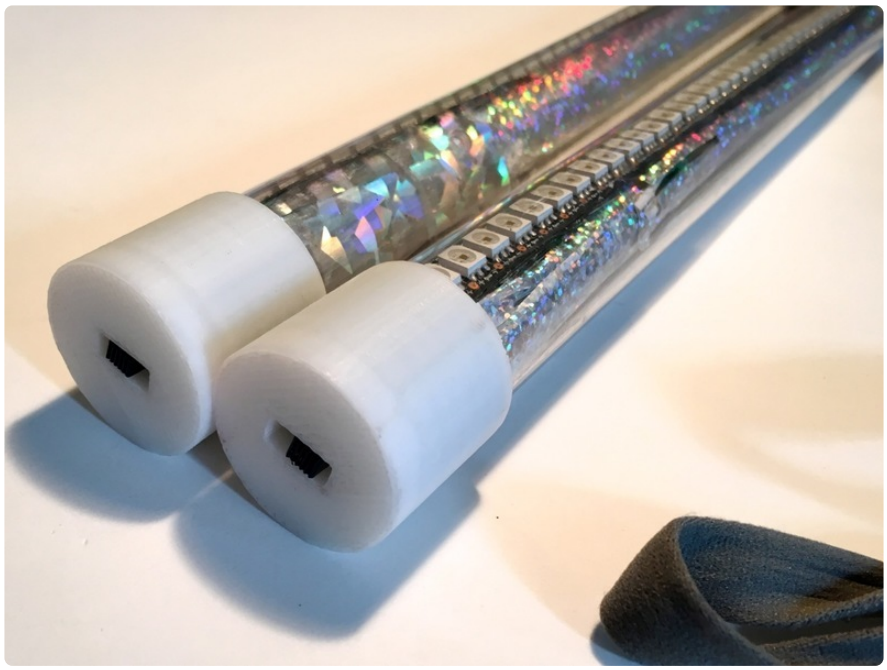
Once you're sure that everything is working, it's time to secure your connections.

- Put a dab of hot glue over the LED solder joints to keep them from popping off if your poi get a good whack.
- Glue the battery charger to the back of the Teensy with a big dollop of hot glue (enough to ensure they don't touch and short each other out
- Wind the wires to the IR sensor around the teensy and glue in place near the bottom end of the teensy (away from the USB port) with the bump on the IR sensor facing out
- Glue the Teensy down to the top of the wooden dowel. I used a liberal amount of hot glue so it stays firmly upright.

Once everything is secure, carefully slide the whole assembly into your polycarbonate tube. I found it easiest to start with the Teensy end, since the battery end was a pretty tight fit.

Once it's inside, fit the bottom cap over the power switch. Make sure the switch is accessible and works nicely, then glue the cap on using E6000 glue (or your favorite acrylic glue).

If you haven't already, thread your leash through the holes you drilled at the top of the poi and secure it. Turn it on and give it a twirl!



Adding Images

The poi can display **BMP or GIF** images. Images need to be 36 pixels tall (or however many LEDs your poi have).

The process for converting images is a little gritty right now, requiring a command-line tool written in Python. It also requires the Python Imaging Library (PIL).

Probably the least-bothersome way to do this right now is on a Raspberry Pi computer, where most of the tools are already built-in, though this requires some familiarity with the Linux operating system.

Phil says he'd like to make a more user-friendly tool for this in the future. But for the time being these steps remain a bit technical.

Installing and using Python varies from system to system. On the Raspberry Pi, Python is already installed by default, though PIL must be added manually:

```
sudo apt-get install python-imaging
```

Things will be entirely different on Windows or Mac or even on other Linux distributions. Unfortunately setting up Python is way beyond the scope of this guide, so you might Google 'round for tutorials elsewhere. If this gets too dry and technical, don't fret...I suspect that given time other users will post some good poi-ready images to the Adafruit Forums (https://adafru.it/cer). And the code as posted has a good number of images already installed.

Inside the “convert” folder included with the project is a file called “convert.py” — the Python script which readies images for the poi.

As written, this code is designed for the Genesis Poi (https://adafru.it/lgE) project, which use fewer LEDs and a much smaller battery. For maximum eye-searing potential on our supernova poi, we need to make some small changes to the script. Open the file in a text editor and look for these three lines near the beginning:

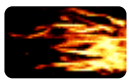
```
batterySize    = 150  # Battery capacity, in milliamp-hours (mAh)
runTime        = 1.1  # Est. max run time, in hours (longer = dimmer LEDs)
parallelStrips = 2    # Same data is issued to this many LED strips
```

Change these to:

```
batterySize    = 2200 # Battery capacity, in milliamp-hours (mAh)
runTime        = 2.5  # Est. max run time, in hours (longer = dimmer LEDs)
parallelStrips = 2    # Same data is issued to this many LED strips
```

Save the changes to the file. But if you’ll ever be converting images for the Genesis Poi in the future, you’ll need to change those lines back.

Once everything is installed and your scripts are up to date, here's a video detailing how to add images to your poi.



So, let’s suppose we have this little flames image, which is 36 pixels tall (the same number as my LED count on each end):

The top of the image will correspond to the top of the poi, so if you're showing text or a directional image you may want to flip the image upside down so it looks upright at the top of the swing.

To convert this using the Python script, you’d type:

```
python convert.py images/flames.gif &gt; graphics.h
```

Or you can convert a whole list of images:

```
python convert.py images/*.gif &gt; graphics.h
```

The “> graphics.h” redirects the output of the convert.py script to the plain-text file graphics.h, which can then be incorporated into an Arduino sketch.

Inside the file you'll see one or more sections like this:

```
// usa.gif -------------------------------------------------------------

const uint8_t PROGMEM palette04[][3] = {
  {  56,  56,  56 },
  {  56,   0,   0 },
  {   0,   0,   0 },
  {   0,   3,  56 } };

const uint8_t PROGMEM pixels04[] = {
  0X22, 0X22, 0X22, 0X22, 0X22, 0X22, 0X22, 0X22,
  0X22, 0X22, 0X22, 0X22, 0X22, 0X22, 0X22, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X10, 0X10, 0X10, 0X10, 0X10, 0X10, 0X12, 0X22,
  0X33, 0X33, 0X33, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X30, 0X30, 0X30, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X33, 0X03, 0X03, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X30, 0X30, 0X30, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X33, 0X03, 0X03, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X30, 0X30, 0X30, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X33, 0X03, 0X03, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X30, 0X30, 0X30, 0X30, 0X10, 0X10, 0X12, 0X22,
  0X33, 0X33, 0X33, 0X30, 0X10, 0X10, 0X12, 0X22 };
```

Above is the data for an American flag…a four-entry color palette (white, red, black, blue) followed by the pixel data (packed two pixels per byte).

Then, near the bottom of the file, you'll see a block like this:

```
const image PROGMEM images[] = {
  { PALETTE1 ,  100, (const uint8_t *)palette00, pixels00 },
  { PALETTE4 ,   48, (const uint8_t *)palette01, pixels01 },
  { PALETTE4 ,   54, (const uint8_t *)palette02, pixels02 },
  { PALETTE4 ,    1, (const uint8_t *)palette03, pixels03 },
  { PALETTE4 ,   24, (const uint8_t *)palette04, pixels04 },
  { PALETTE4 ,    9, (const uint8_t *)palette05, pixels05 },
  { PALETTE4 ,   26, (const uint8_t *)palette06, pixels06 }
};
```

This table holds references to all of the images in the file, along with their widths in pixels (height is always 36) and format.

Paste this code into your arduino sketch (in the graphics.h tab) and upload it to your poi. You're good to go!

Using the Remote

This remote can be reconfigured in the code to work however you want. Here's how I did it: