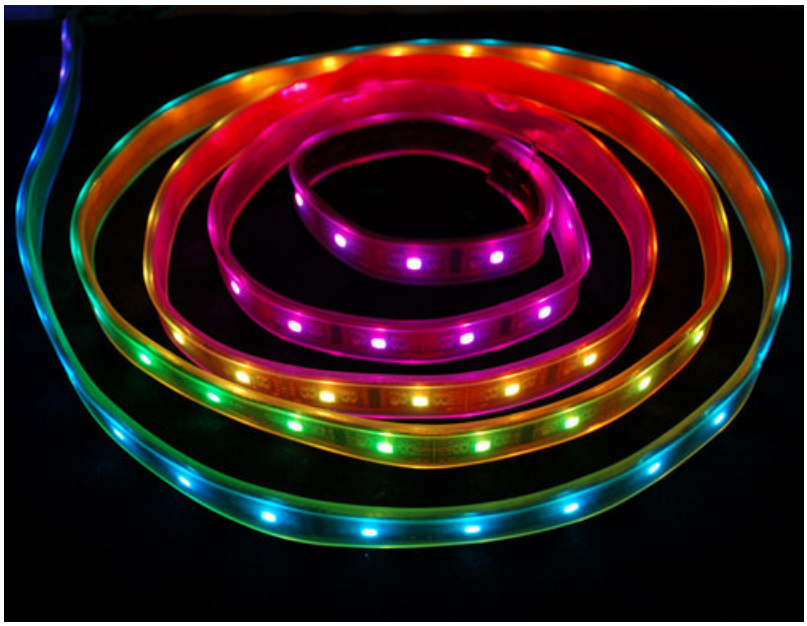
adafruit learning system

LPD8806 Digital RGB LED Strip

Created by Phillip Burgess

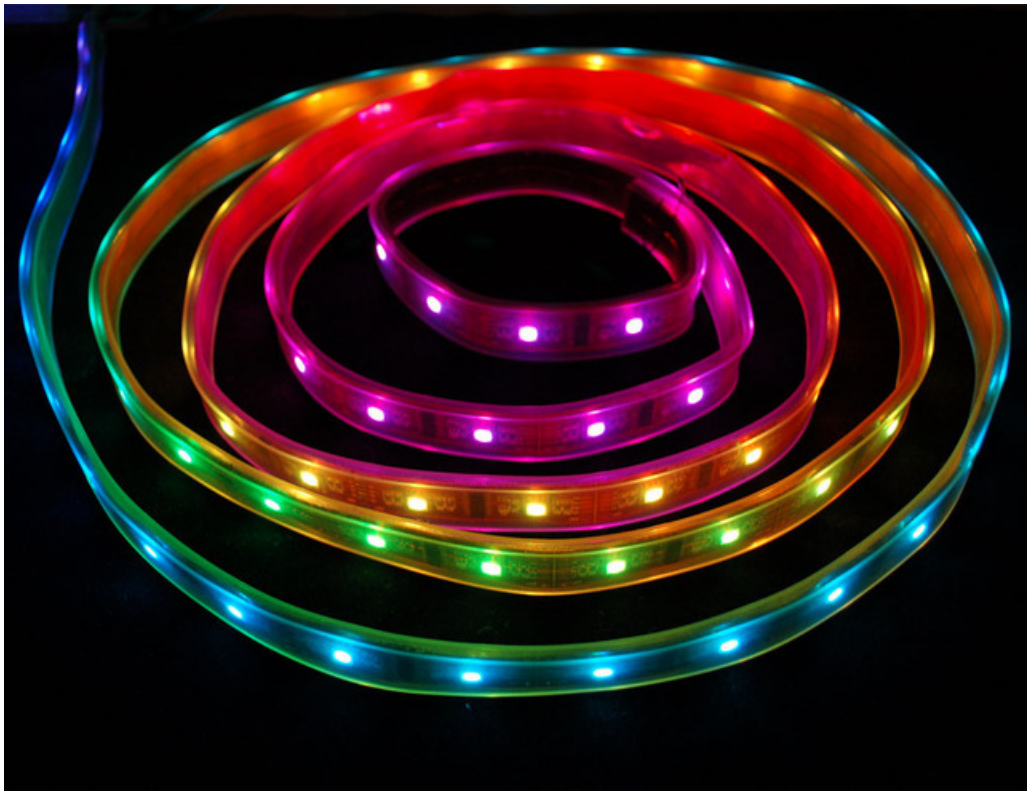


Last updated on 2018-08-22 03:31:03 PM UTC

Guide Contents

Overview

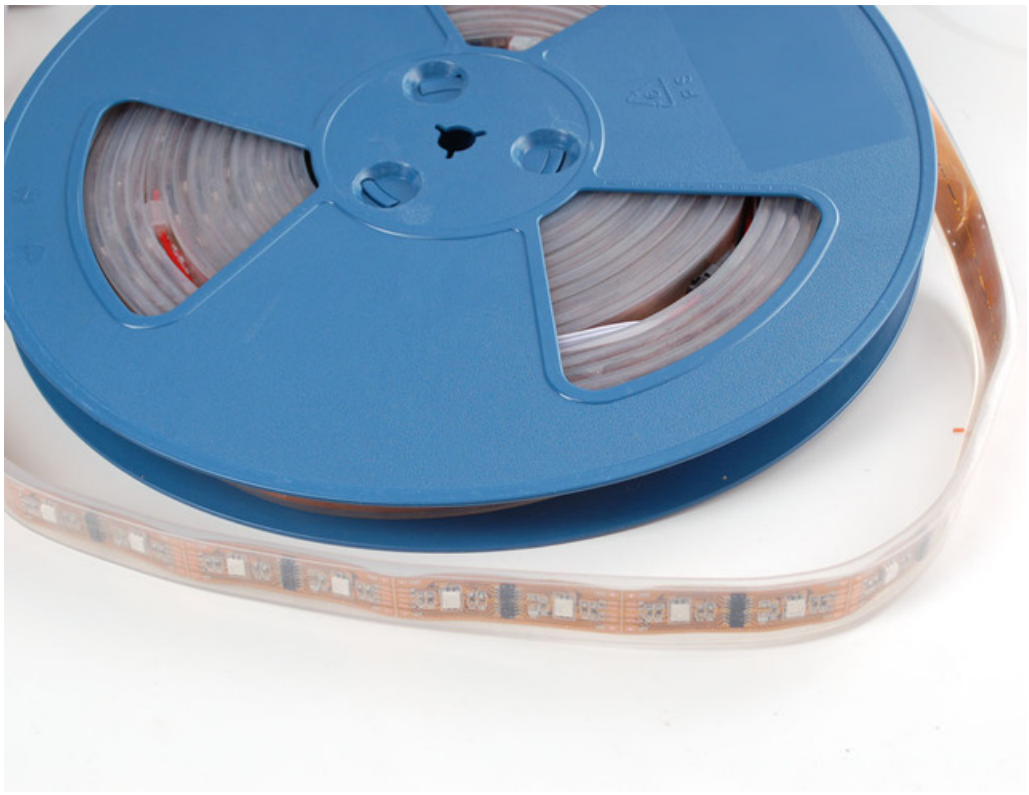
This guide is for LPD8806 LED strips, if you have NeoPixels check out our guide here https://learn.adafruit.com/adafruit-neopixel-uberguide



We love some good LED blinking as much as the next person but after years of LED-soldering we need something cooler to get us excited. Sure there are RGB LEDs and those are fun too but what comes after that? Well, we have the answer: **Digital LED Strips**! These are *flexible* circuit boards with full color LEDs soldered on. They take a lot of LED-wiring-drudgery out of decorating a room, car, bicycle, costume, etc. The ones we carry come with a removable waterproof casing.

There are two basic kinds of LED strips, the "analog" kind and "digital" kind. Analog-type strips have all the LEDs connected in parallel and so it acts like one huge tri-color LED; you can set the **entire** strip to any color you want, but you can't control the individual LED's colors. They are very very easy to use and fairly inexpensive.

The Digital-type strips work in a different way. They have a chip for each LED, to use the strip you have to send digitally coded data to the chips. However, this means you can control each LED individually! Because of the extra complexity of the chip, they are more expensive.

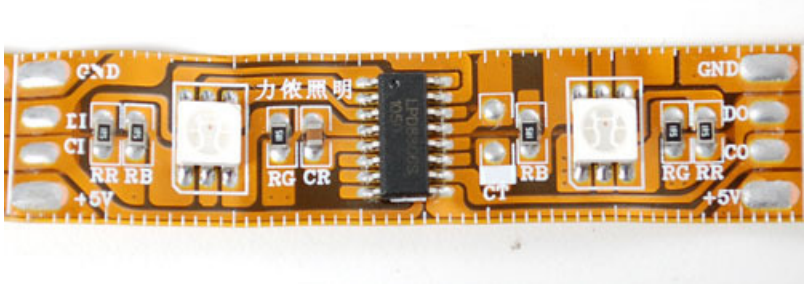


Technical specs:

- 32 LEDs per meter (16 segments)
- 2 common-anode RGB LEDs per segment, individually controllable
- Removable IP65 waterproof casing
- Maximum 5V @ 120mA draw per 2.5" strip segment (all LEDs on full brightness) - about 2A per meter
- 16.5mm (0.65") wide (LPD8806), 4.5mm (0.18") thick with casing on, 62.5mm (2.45") long per segment
- LED wavelengths: 630nm/530nm/475nm
- Microcontroller required to control strip

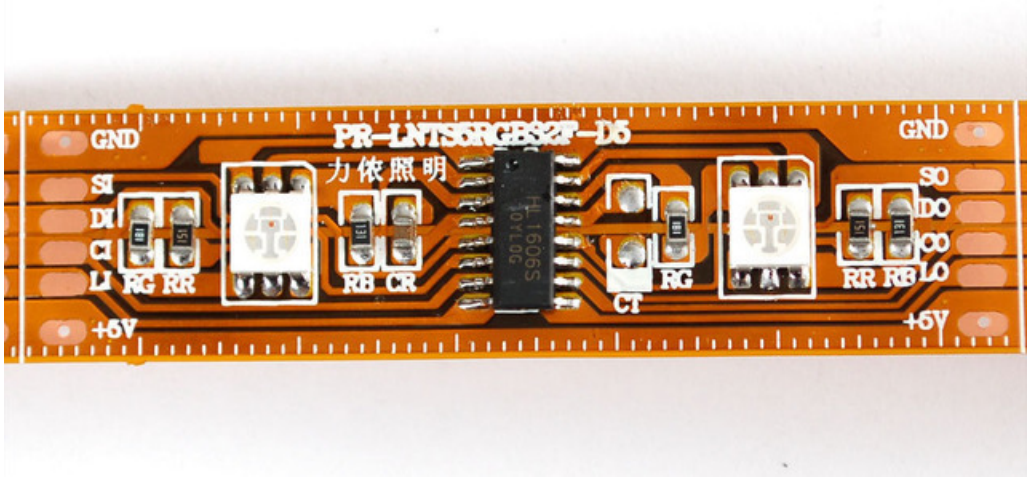
Know Your LED Strips

Adafruit carries several types of flexible LED strip. Examine your strip closely. If it looks like this:



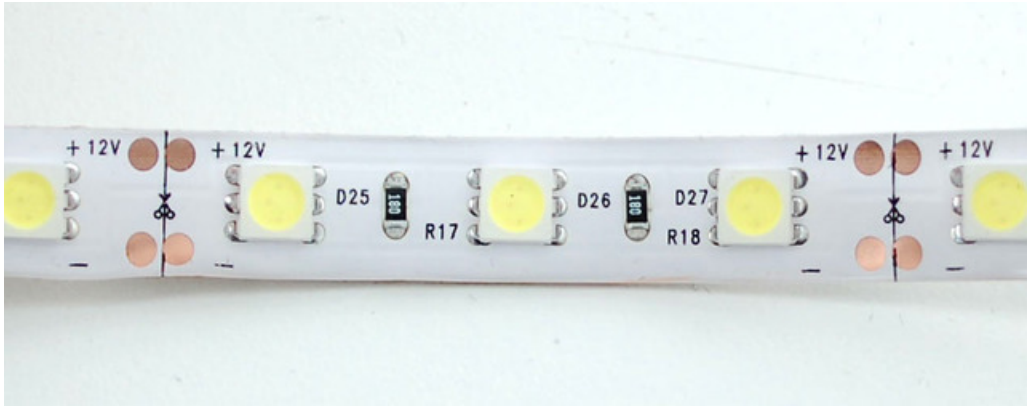
Then you're reading the right tutorial. Proceed!

If your strip looks like this:



Then you have the prior generation HL1606 strip. Adafruit no longer sells this model, but our tutorial is still available for reference (https://adafru.it/aHL).

If your strip resembles this:



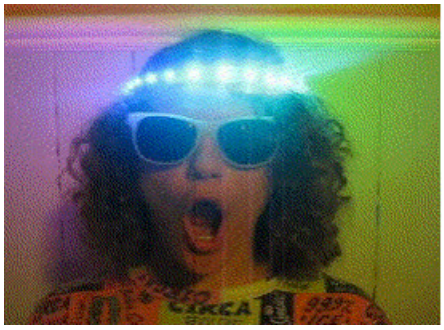
Then you have one of our “analog” LED strips, a different animal altogether. Here’s the appropriate tutorial to get you started (https://adafru.it/cl6).

Project Ideas

Here's just a few ideas what these strips could be used for:

A large wall-mounted LED bandwidth monitor (https://adafru.it/aHP):

A trippy head band. Duuuuuude!



One-dimensional cellular automata:

Light-edge clock:

Wiring

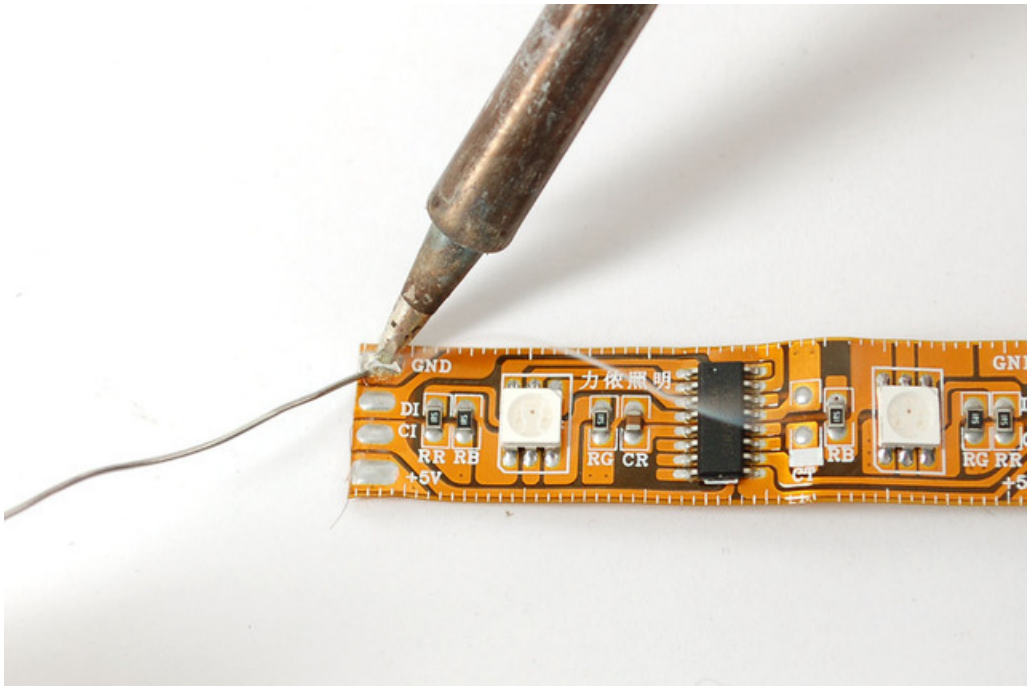
This guide is for LPD8806 LED strips, if you have NeoPixels check out our guide here https://learn.adafruit.com/adafruit-neopixel-uberguide

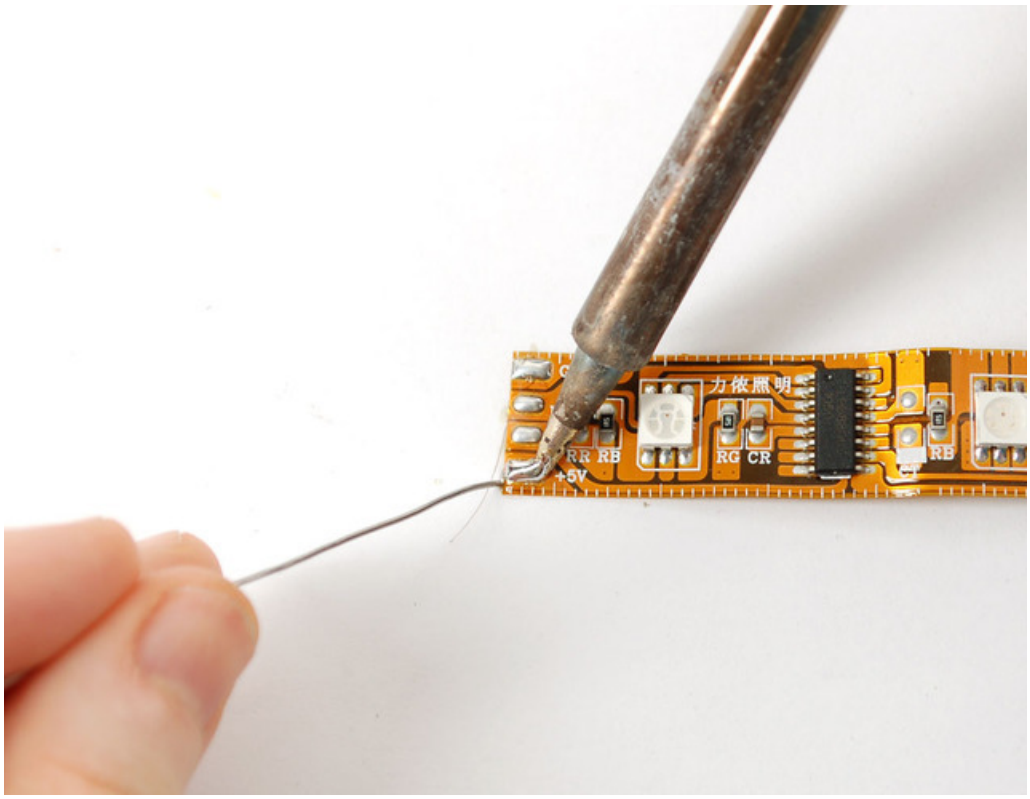
Data moves only one direction along these strips — they have a definite “in” and “out” end, and the microcontroller must be attached to the “in” end. At the ends of each strip are four solder pads. If you look closely, the two middle pads are labeled either DI and CI or DO and CO. These stand for “data in” and “clock in,” and “data out” and “clock out,” respectively. We want to connect wires to the “**in**” end, so we’ll use the group of 4 pads labeled **GND/DI/CI/+5V**.

Some strips arrive with plugs pre-wired for testing at the factory, but these might be at *either end* of the strip and aren’t necessarily useful for connecting to the microcontroller — in most cases you’ll still have to solder your own wires. Check three times to make sure you’re connecting to the INPUT side! Just because your strip includes a plug does not mean it is useful. You might get lucky, but odds are against it.

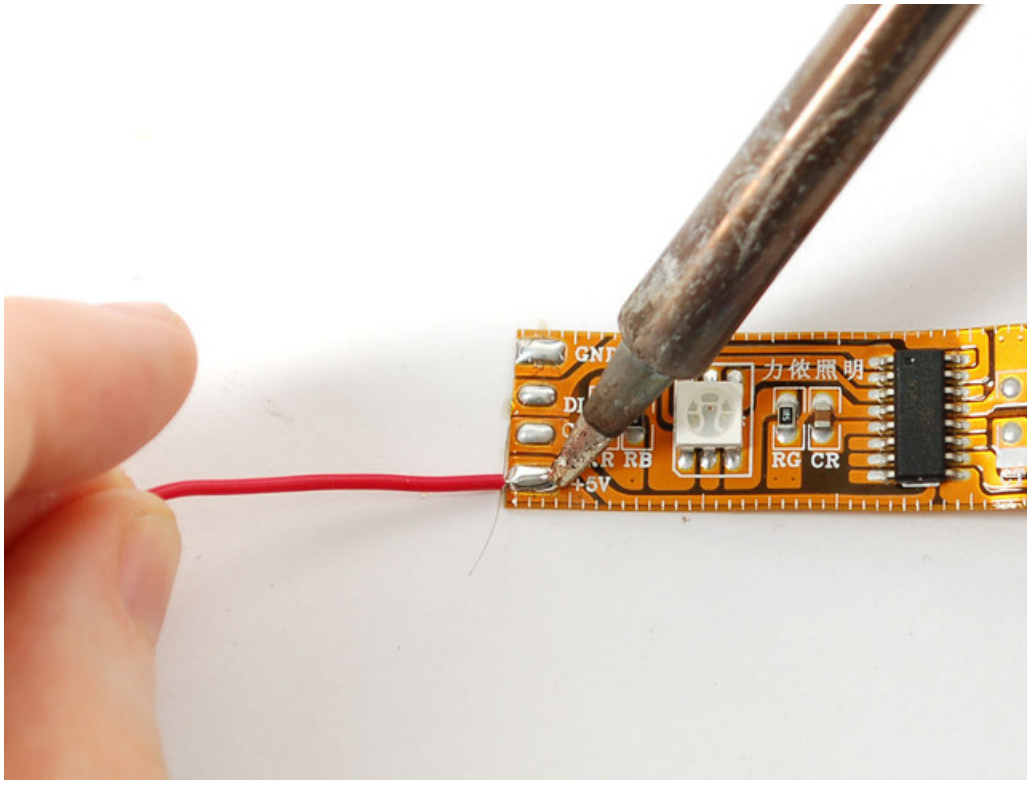
To make the images clearer, we removed the plastic covering for this tutorial. However, you can just cut a little bit away on yours to allow you to access the pads. Once you take off the plastic cover its a bit difficult to get it back on.

Tin all four “in” pads by carefully melting a little solder onto the pads:

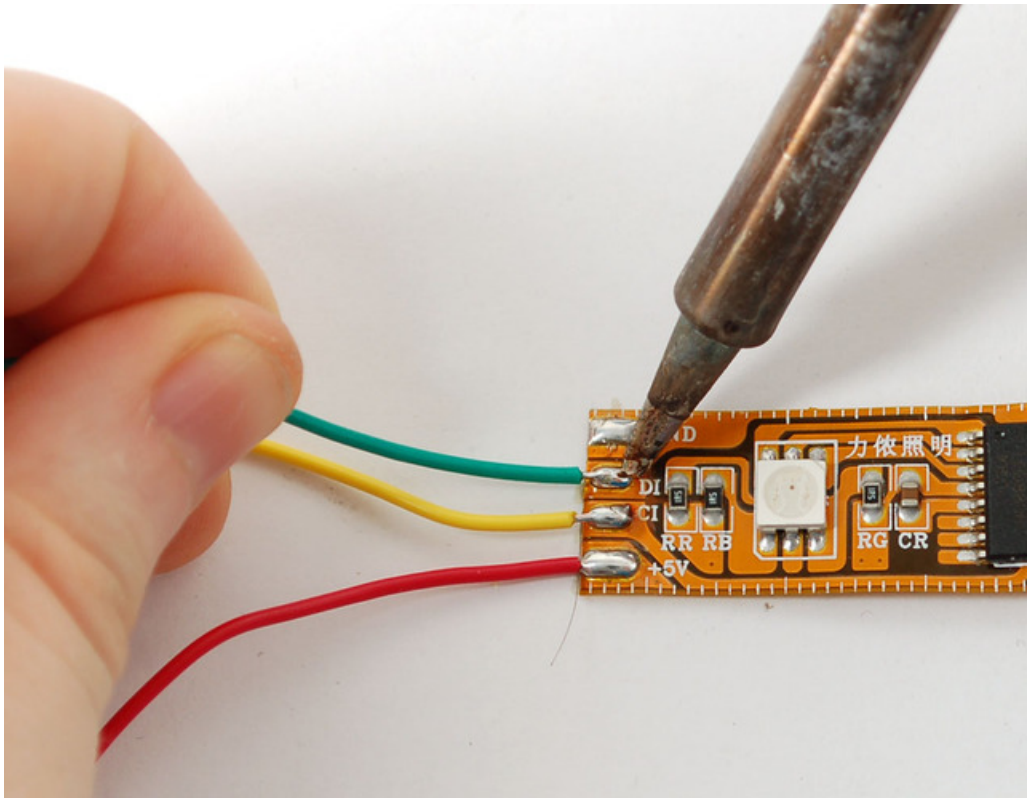




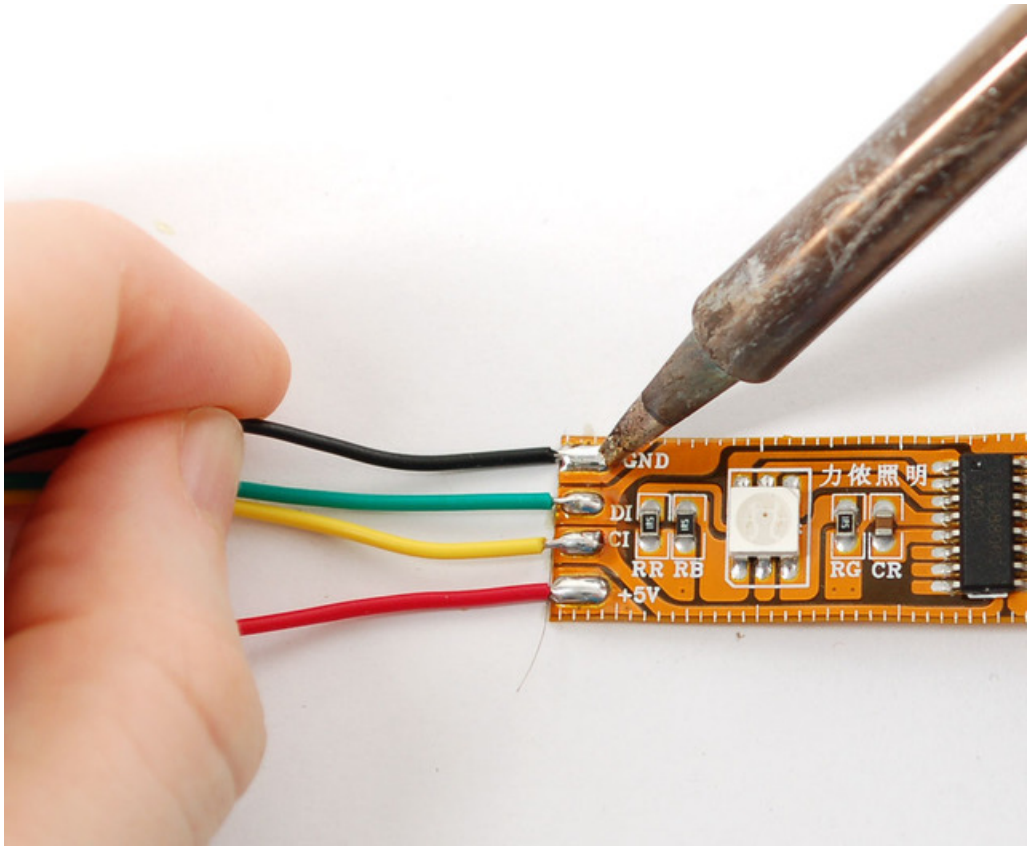
Start by soldering a red wire to the +5V power line:



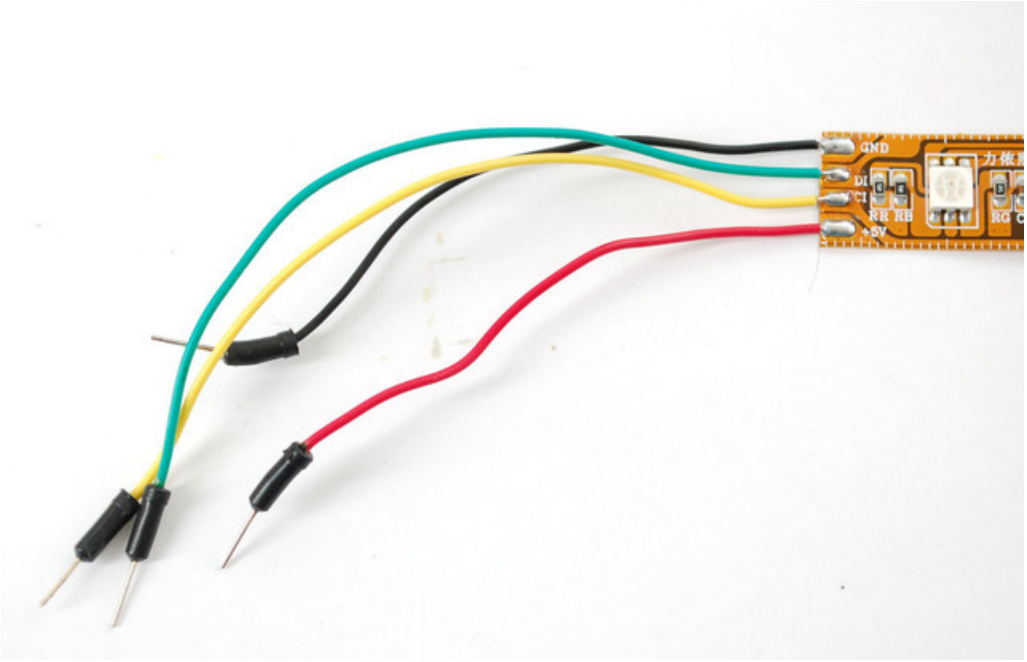
Next connect two wires to the data and clock pads. We'll use yellow for the Clock pin (CI) and green for the data pin (DI):



Finally, solder a black wire to ground:

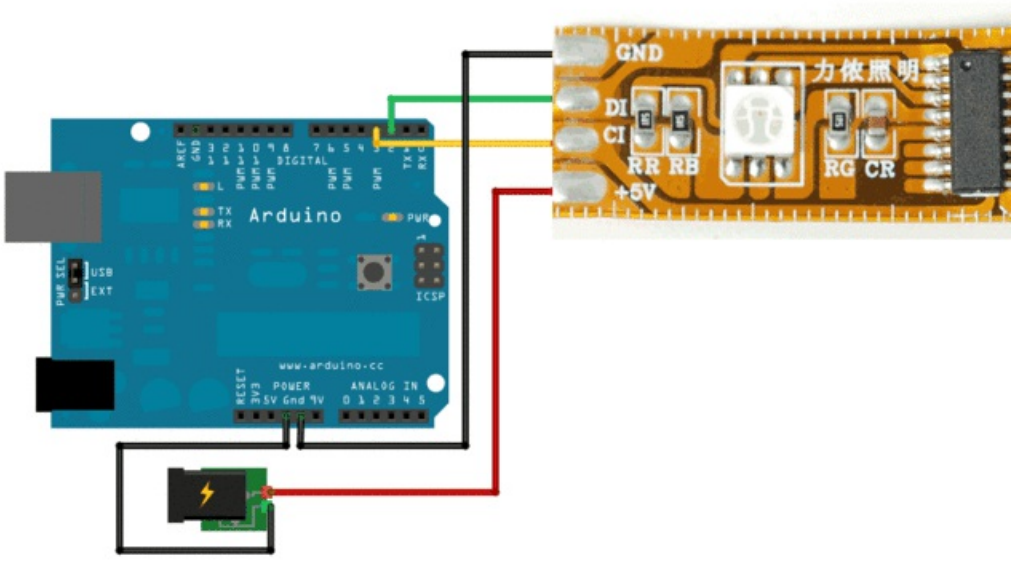


That's it! Now you're ready to use the strip. You may want to use heatshrink to provide a secure cover for the wires, or stuff hotglue in the end, which will do the same.



Connecting to Arduino

- Connect the Black Ground to any ground pin of the microcontroller (this is for data and power ground)
- Connect the Yellow Clock wire to digital Pin 3 (you can change this later)
- Connect the Green Data wire to digital Pin 2 (you can change this later)
- Connect the Red +5V power wire to your 5V power supply.



Power

When running a lot of LEDs, it's important to keep track of power usage. Individual LEDs don't get very hot or use tons of power, but they add up fast!

Each single RGB LED can draw up to 60mA from a 5V supply. That means a full meter can use nearly 2 Amps. That's a peak rate, which assumes that all the LEDs are on at full brightness. If most of the LEDs are kept dim or off (as when animating patterns), the power usage can be 1/3 this or less.

As shown in the previous wiring diagram, connect ground to both your power supply and microcontroller. Then connect the 5V line from the power supply to the red wire on the LED strip.

> Never, NEVER connect more than 5 Volts to the strip! This will permanently damage it!

We suggest a nice switching supply for driving LED strips:

Our 5 Volt, 2 Amp power supply (http://adafru.it/276) is ideal for a one meter strip.

For larger projects, our 5 Volt 10 Amp power supply (http://adafru.it/658) is good for up to 5 meters (160 pixels total).

The female DC power adapter (http://adafru.it/368) mates with either of the above power supplies. Screw terminals clamp down on power wires at the end of a strip.

Note the embossed polarity markings. Connect the **red wire** to the + terminal and the **black wire** to the - terminal.

Tips for powering LED strips:

- When creating longer runs, power should be split and applied *every* meter. If you try to power too many LEDs from just one end of the strip, they'll start to "brown out" the further they are from the power supply.
- Strands can be powered from *either* end — "input" and "output" doesn't apply to power, only the data signals from the microcontroller.
- If the 10 Amp power supply isn't large enough for your project, a slightly modified ATX computer power supply (https://adafru.it/CbO) can provide 30 Amps to power upwards of 500 pixels!
- Generally speaking, you should not try to power an LED strip from the Arduino's 5V line. This is okay if just a few pixels are lit, but is not adequate for driving a full strand.
- For a standalone application (not USB connected to a computer), you can power the Arduino from the same regulated 5V supply as the LEDs — connect to the 5V pin on the Arduino, *not* Vin, and don't use the DC jack on the Arduino.

Code

Installation

To download the Arduino library, visit the repository on GitHub (https://adafru.it/aHT) . Click the DOWNLOAD ZIP button near the upper left, extract the archive and then rename the uncompressed folder to LPD8806. Confirm that this folder contains the files LPD8806.cpp and LPD8806.h and the examples folder.

Place the LPD8806 folder inside your Arduino libraries folder. You may need to create this folder if it does not yet exist. In Windows, this would be (home folder)\My Documents\Arduino\Libraries and for Mac or Linux is (home folder)/Documents/Arduino/Libraries. We also have a tutorial on library installation (https://adafru.it/aYG).

After installing the LPD8806 library, restart the Arduino IDE. You should now be able to access the sample code by navigating through menus in this order: File→Sketchbook→Libraries→LPD8806→strandtest

strandtest is written for a one meter LED strip attached to Arduino pins 2 and 3. If you're using a longer or shorter strip, you should change this line:

```
LPD8806 strip = LPD8806(32, dataPin, clockPin);
```

so that the first argument to the object is the number of LEDs in your strip. Each meter has 32 LEDs so count them or do the math. Now upload it to your Arduino, your strip should start to perform a bunch of demonstration tests! (The 'longstrandtest' sketch already has this change made for a full 5 meter strip.)

The other two example sketches — LEDbeltKit and advancedLEDbeltKit — are part of a kit based on a different controller board, and won't work on a standard Arduino without modification. This is just a matter of pin assignments...if you're crafty you should be able to figure it out.

Netduino and LPD8806 based strips

Driving these strips from Netduino (or other .Net Micro Framwork boards like FEZ Panda) is very convenient and doesn't require a code library if SPI ports are used. The following sample shows driving a 32 pixel light strip connected to SPI1 (SCLK on pin 13 and MOSI on pin 11):

```
using Microsoft.SPOT.Hardware;
...
        public static void LightStripSpi()
        {
            var spi = new SPI(new SPI.Configuration(Cpu.Pin.GPIO_NONE,
                false, 0, 0, false, true, 10000, SPI.SPI_module.SPI1));
            var colors = new byte[3 * 32];
            var zeros = new byte[3 * ((32 + 63) / 64)];

            while (true)
            {
                // all pixels off
                for (int i = 0; i < colors.Length; ++i) colors[i] = (byte)(0x80 | 0);
                // a progressive yellow/red blend
                for (byte i = 0; i < 32; ++i)
                {
                    colors[i * 3 + 1] = 0x80 | 32;
                    colors[i * 3 + 0] = (byte)(0x80 | (32 - i));
                    spi.Write(colors);
                    spi.Write(zeros);
                    Thread.Sleep(1000 / 32); // march at 32 pixels per second
                }
            }
        }
```

Arduino Library Docs

Arduino Library Docs (https://adafru.it/Au9)

Troubleshooting

Many support issues arise from eager users getting ahead of themselves, changing the code and wiring before confirming that all the pieces work in the standard configuration. We recommend always starting out with the examples as shown. Use the pinouts and wiring exactly as in the tutorial, and run the stock, unmodified "strandtest" example sketch. Only then should you start switching things around.

Here are the most common issues and solutions...

The pixels are wired and powered exactly as in the tutorial, the sketch compiles and uploads successfully, but nothing happens.

- Double-check all wiring. Are the clock and data wires swapped? Is ground connected to the Arduino?
- Confirm the Arduino is connected to the INPUT end of the strip. If your strip came with a plug pre-soldered, there is only a 50/50 chance this is the correct end. Examine the strip closely, and solder wires to the INPUT end if needed.
- Check power supply polarity and voltage. Are + and - swapped? If you have a multimeter, confirm 5V DC output (±10%) from the power supply.
- Is the correct board type selected in the Arduino Tools→Board menu?

A few LEDs randomly turn on when power is applied, but then nothing happens.

The power supply is probably OK. Check for any of the following:

- Double-check all wiring. Are the clock and data wires swapped? Is ground connected to the Arduino?
- Confirm the Arduino is connected to the INPUT end of the strip. If your strip came with a plug pre-soldered, there is only a 50/50 chance this is the correct end. Examine the strip closely, and solder wires to the INPUT end if needed.
- Is the correct board type selected in the Arduino Tools→Board menu?
- Did the strandtest code successfully compile and upload?

Only the first few LEDs respond. The rest remain off or flicker randomly.

- Confirm that the number of LEDs in the LPD8806() constructor match the number of LEDs in the strip (both will be 32 if using the strandtest example and a single meter of LEDs).
- Check for conductive detritus that may have gotten into the strip: solder balls, frayed bits of wire, etc.
- Between each pair of LEDs there's a small chip. Find the chip corresponding to the first bad LED, and give the strip a firm squeeze here — it may simply be a dodgy connection. Try this with the *prior* chip as well. If that works, you can either touch up the connections with a soldering iron, cut out the offending section of strip and join the two sub-strips, or arrange for a replacement strip if it's new.

The LEDs flicker randomly, not the way they're programmed to.

Are the clock and data wires swapped? Is ground connected to the Arduino?

"White" LEDs are showing pink instead.

- This can happen when trying to power too long of a strip from one end. Voltage will drop along the length of the strip and the furthest pixels will "brown out." Connect power *every* meter.

Sketch will not compile: error message is "LPD8806 does not name a type"

- Confirm the library is unzipped prior to installation.
- Confirm the library is properly named and located. The folder should be called LPD8806, and placed inside your personal Documents/Arduino/Libraries folder — *not* inside the Arduino application folder!

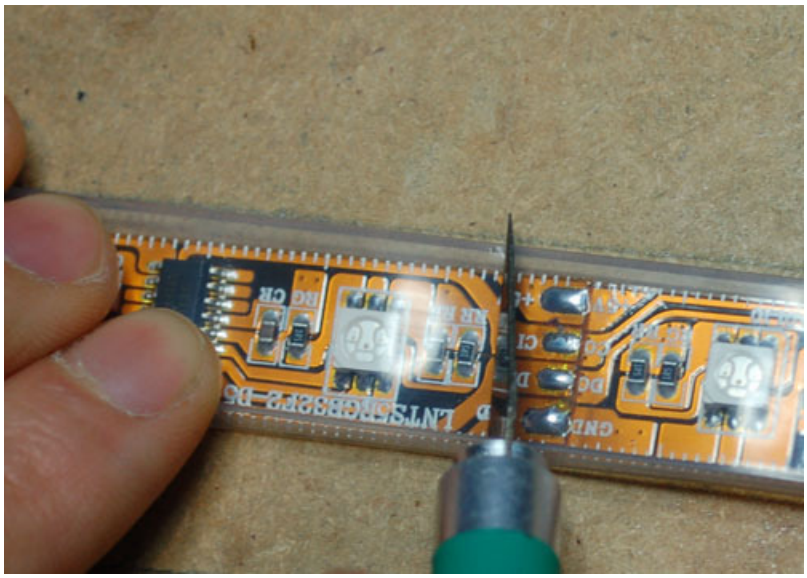
- After installation, the Arduino IDE needs to be restarted for new libraries to be used.

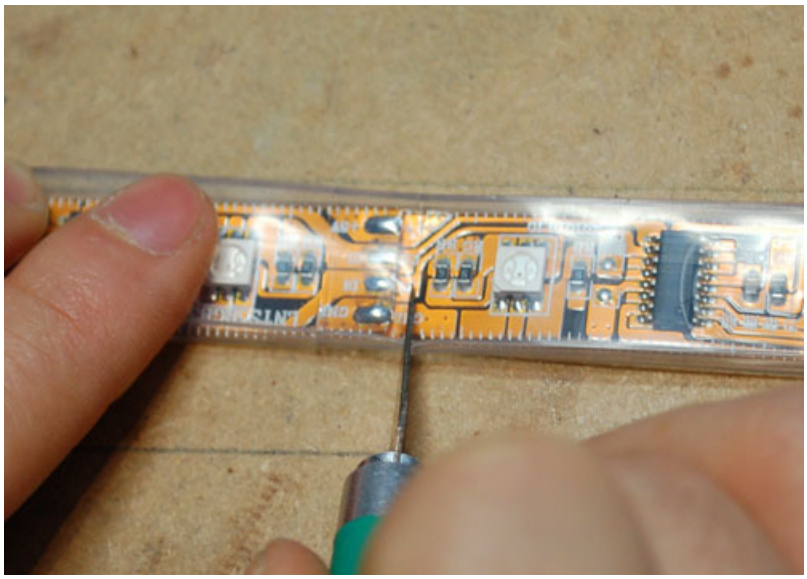
Advanced: Separating Strips

All our LED strips are manufactured in 1/2 meter long sections, and these sections are then joined to produce strips up to 5 meters long. For other lengths, you can normally cut the flex strip between each pair of LEDs...but if this coincides with one of those 1/2 meter joins, you must instead pull apart the soldered sections. It's not terribly hard, but worth documenting!

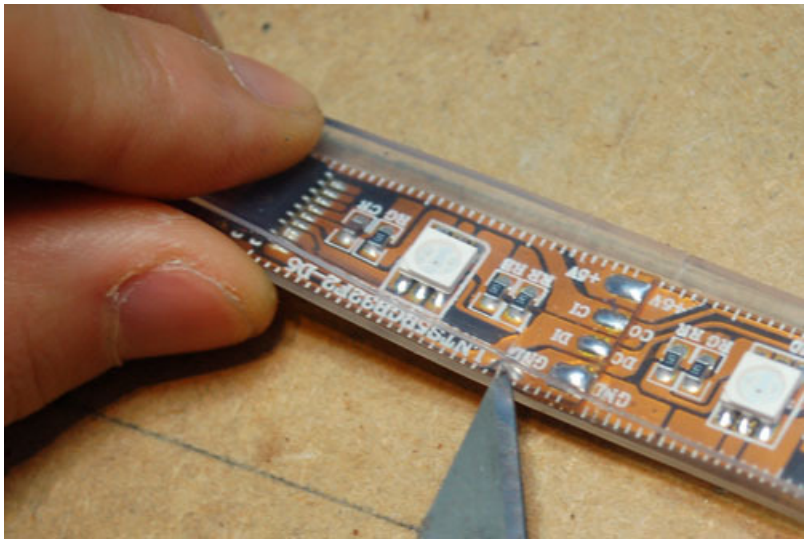
For this, we use the wide flat tip on our METCAL iron. A wide tip isn't required but it sure is handy!

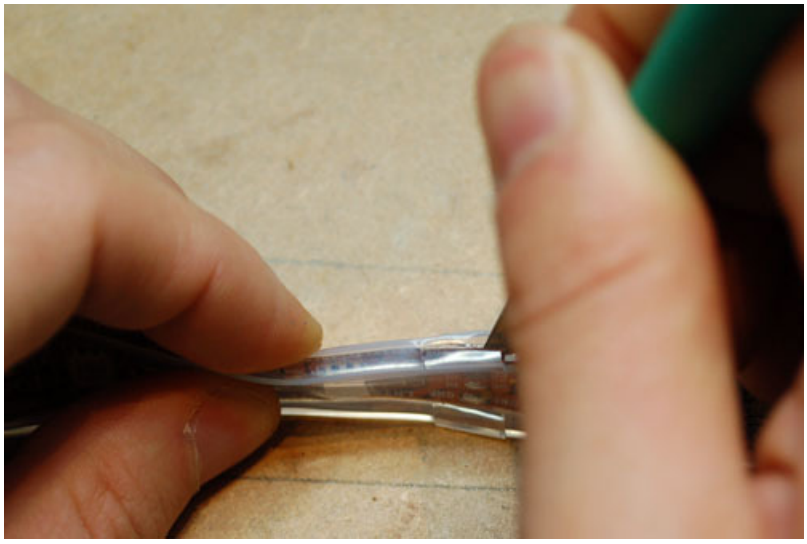
Using an X-acto knife, cut two slices from the top on either side of the solder connection:



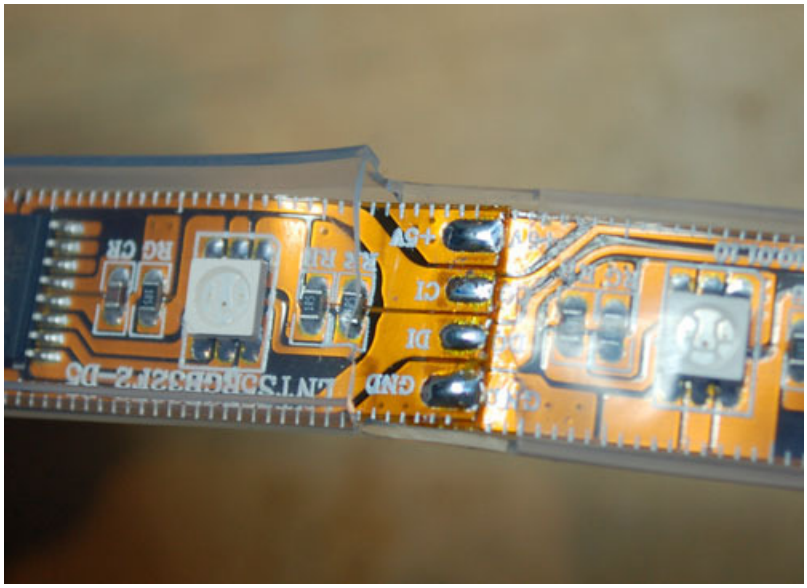


Then slice the rubber coating between the two cuts you made:

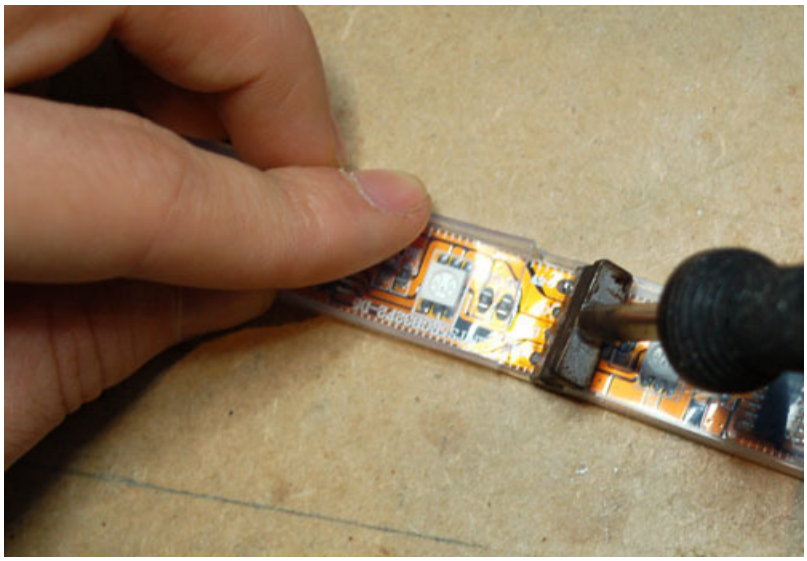


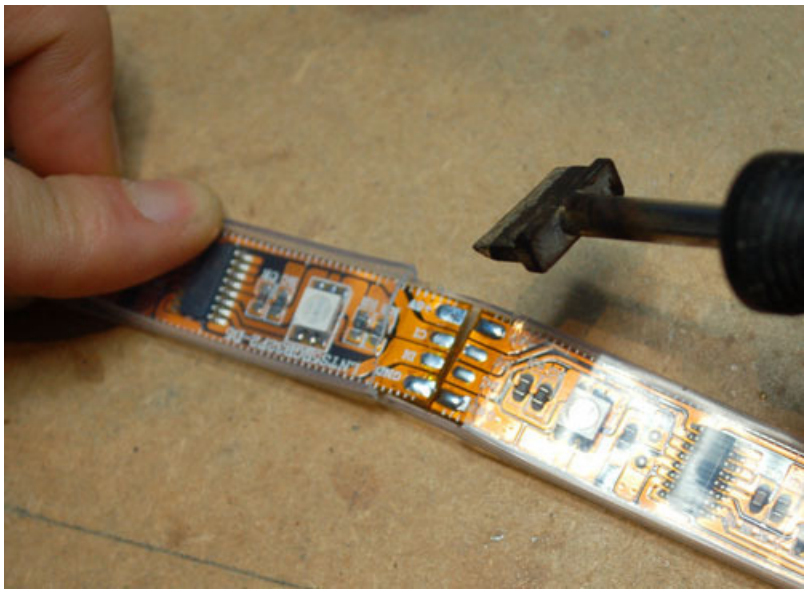


Pull off the thin rubber piece between the cuts:

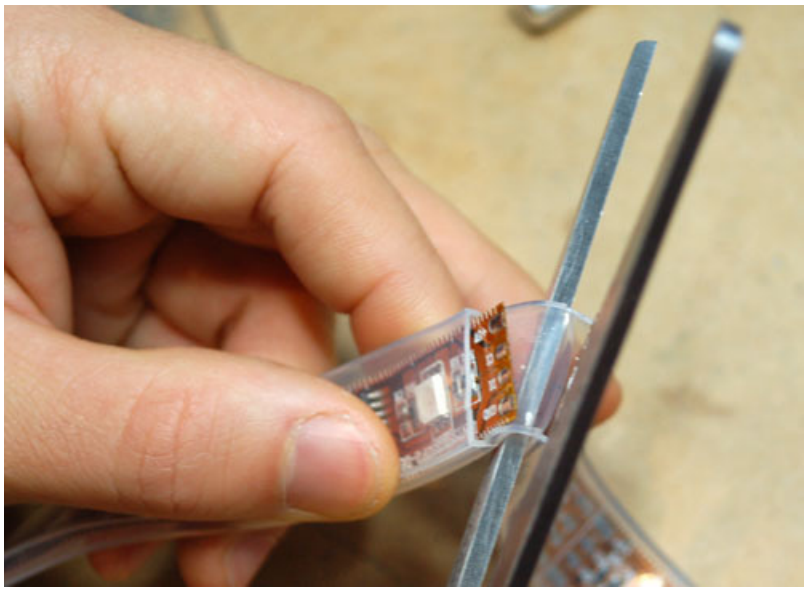


Press the soldering iron against the four joints while pulling the two pieces apart:

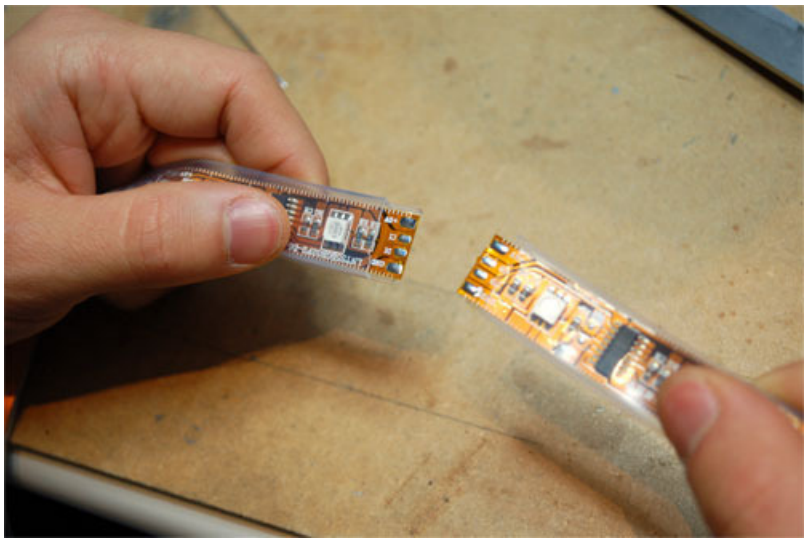




Cut the remaining rubber underneath with scissors:



Thats it!



F.A.Q.

How many pixels can I control at once?

The theoretical limit for pixels - assuming you can power them all - is infinite!

In reality, there's a limit because the controller chip must store the entire LED strip pixel data in memory. Each pixel takes up 3 bytes. For an Arduino Uno (or other '328 based 'duino), the processor has very limited memory, that means that you can't really run more than 250 or so pixels. That number is a little dependant on what else the Arduino is storing - for example the SD library takes up a massive amount of RAM memory. In our experience, most people have issues with over 150 pixels.

There is no easy way to check runtime RAM usage (unlike the FLASH storage which is printed out on compilation) so if you're having issues with longer strips, shorten the strip or update to a Mega which has tons more RAM.

There are a few things you can do to minimize RAM usage. Minimizing string usage is often a good place to start. This page has some good tips: http://itp.nyu.edu/~gpv206/2008/04/making_the_most_of_arduino_mem.html

Long strips of pixels start failing!

See above, you are likely running out of RAM storage.