ADVANCED COSPLAY LIGHTS

ANIMATED LEDS

BY SVETLANA QUINDT

GO MAKE SOMETHING!

Interested in armor, prop making, painting or sewing? Check out my other books!

kamuicosplay.com/books

If you've read my previous volume **The Book of Cosplay Lights**, you've already learned a lot about the basics of how LEDs work. From serial and parallel connections to batteries and soldering - even clever ways to hide your light effects in your props and costumes. Since you're reading this new book however, I can only assume that this was not enough for you - that you wanted *more*.

Let me introduce you to the colorful world of animated LEDs! Instead of working with steady lights, this time you'll learn how to animate bright cascades of shiny rainbows or let your costumes pulse as if they have a heartbeat. Sounds pretty neat, right? Well, once you begin, you won't be able to stop! Just turn your soldering iron on and get your fingers ready for some coding: It's time to give the world some color!

Before we start, I want to make one thing clear: I am not a programmer and I don't want to turn you into one either. Instead I'll tell you something about *reverse engineering*. Fancy words, right? It just means that we will use code that's already written and change it so we get the results we want for our projects. No need to write anything from scratch! To do this however we need to at least understand what the code does. So we'll play around with some fancy lights, copy, paste and edit bits of it until we actually know what we're doing. Sounds fun right? Let's start!

While I always try to keep things as simple as possible, please take note that this is an 'advanced book' and the follow up of **The Book of Cosplay Lights**. I won't repeat all the basics here again. So if you don't know what volt is or how to solder two cables together, please go check out the first volume! I'll wait here for you.

One final tip: You'll get the most out of this book if you actually follow along with my examples. Some things will only become clear if you do them for yourself.

ABOUT THE AUTHOR

KAMUI COSPLAY

We're Svetlana and Benni but most of you just know us as Kamui Cosplay. If you're into cosplay or prop making, chances are you've already stumbled upon one of our videos, pictures or tutorials before. We consider ourselves very lucky to be able to make our living by helping creative people like you with their crafting dreams. We're humbled by all the support you give us and try to pass it on as much as we can. We love you!

Thank you for your support by buying this book! Hopefully you will find it helpful and inspirational!

Content

Hello you shiny rainbow beauties!

I already wrote *a lot* about regular LEDs in my previous book, so this time we'll check out the so called **Digital RGB LED Strips**. These useful babies carry 30, 60 or even 144 lights (pixels) per meter. Each pixel consists of a tiny red, green and blue LED as well as a really small chip to process data. This allows you to control them all individually!

LED strips are really nothing else than programmable Christmas lights and that's exactly what makes them so incredibly valuable for costumes and props. By connecting them with a microcontroller (which is like a tiny computer), you can basically tell every single LED what it should do. Light up, change colors, slowly fade on and off or flicker like fire. The possibilities are endless!

In this book I'll mainly use LED strips called **Neopixels**, which are distributed by a company in the United States called **Adafruit**. You can also buy your LED strips from anywhere else in the world as long as they use the same drivers (more on that in the materials and tools chapter). When searching for a strip it's also important to know that they run with **5V**.

Unlike regular LEDs, the strips won't light up when you solder only a battery to their plus and minus pole - they wouldn't know what to do. Instead they have an additional **data pin** which we will use to give them instructions. This is also where our microcontroller comes into play!

There are many different microcontrollers to choose from but I always found the **Arduino** the easiest to understand. It's a very beginner friendly yet powerful microchip designed mainly for students, hobbyists and even kids. It runs with a programming language close to C or C++ and has a huge supportive online community that loves to help, share code for free and constantly comes up with new innovative ideas. A great place for any cosplayer to feel at home!

NOTE

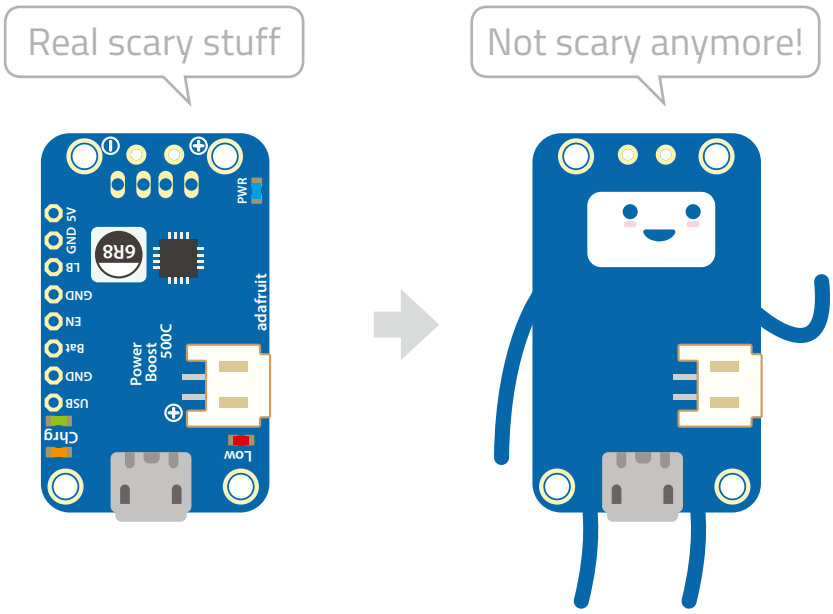
Before we dive deep into electronic circuits, code, and solder fumes, I want you to always keep in mind how awesome LEDs and LED strips are. During the last few years I lit up translucent blades, soul hungry axes and even full glowing armor sets. None of these projects were quick to figure out and things rarely worked the first time.
I know how it feels to get stuck. I've been there often enough. There will be times when you feel like you know nothing, Jon Snow! This topic is no child's play, but when you eventually manage to pull through the result will be worth your effort tenfold.

You're in for a treat!

What you can expect:

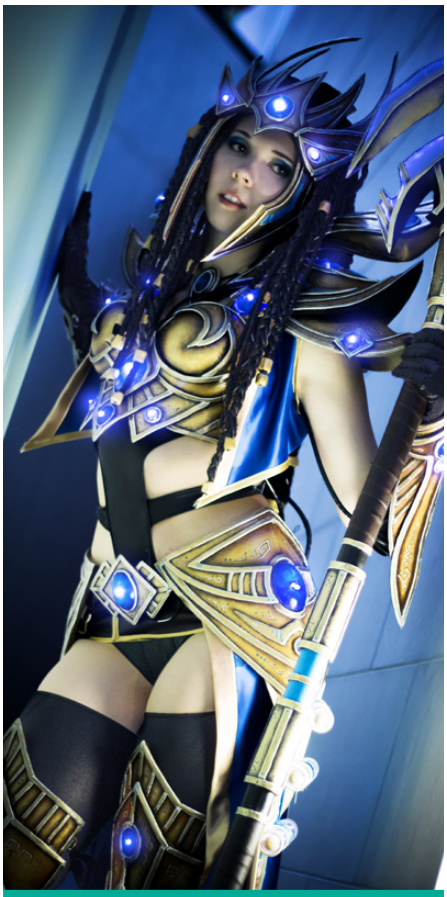
The following chapters will show a lot of circuits and codes that will no doubt look intimidating at first. I spent a lot of time and effort breaking everything down to be as easy and understandable as possible however. I'm sure if you understood my last LED book this will be just as easy for you. My goal is to teach you all the basics and show you with a few easy to follow work examples that despite looking like a scary thing, working with microchips and code is actually pretty fun.

It's really just soldering on a few wires, clicking the right buttons and you are good to go! Even my dog can do it! Well, *not really* - but Zelda still loves to wrap herself in all the shiny lights! Just like a cosplayer! I mean who doesn't want to dress up as a magical, sparkling, shiny unicorn, right? So, don't worry about the scary stuff and instead focus your thoughts on how awesome your costume or your props will look with animated LEDs in them!



A hub for inspiration: adafruit.com

When I started working with LED strips years ago, the website adafruit.com was my main source for inspiration, helpful tutorials and easy to understand code. As a beginner, this website and its countless project related tutorials were actually one of the reasons why I fell in love with animated LED effects and dared to tackle some of my more complicated costumes. Even today I still visit it very often.



'But Kamui! I spent money on this book. Couldn't I have just gone to this website to learn everything?' Yes and you totally fell for it, haha! It's too late now! I already spent it all on dog food! Seriously though, by going through the following chapters, you'll notice that I often refer to Adafruit. I use their products and work with their code but it's all because of a good reason. While there may be other more powerful alternatives, more efficient programming languages or more impressive work examples, Adafruit is a great place for beginners. I know that if you finish this book and feel inspired to learn even more you can just go there and find more easy soldering tutorials, interesting code examples, a helpful online forum and an always growing range of products made for those who are getting started and want to learn more. I'll still try to mention as many alternatives as I can – just keep in mind that I wrote this book for beginners and want to make sure that you can continue learning even when you're done with reading the last page.

adafruit.com

NOTE

The Protoss Wizard was my very first costume that included animated LEDs. I finished this project in 2013 when my English skills were still pretty bad. So sadly I had a lot of trouble understanding the tutorials, let alone the programming code! Luckily I still somehow managed to wire up around 200 LEDs and give them a cool pulsing effect. The biggest lesson I took away from this mammoth of a project was to never give up, even when I'm drowning in code I don't understand.

NOTE

Apart from LED strips, Adafruit offers many different types of products. If you look into their store you'll find motors, sound sensors, wireless controllers, EL wire and tape, single LEDs, tiny displays and a bunch of other cool ideas for cosplay and costuming. They can be a bit on the pricey side but even if you buy your products at other stores like ebay it's still worth a visit just for the tutorials. The important thing is to try out new things! Don't be afraid to wire up a motor and make a part of your costume movable. The future belongs to the fearless!

Let it glow! Let it glooow! ♪

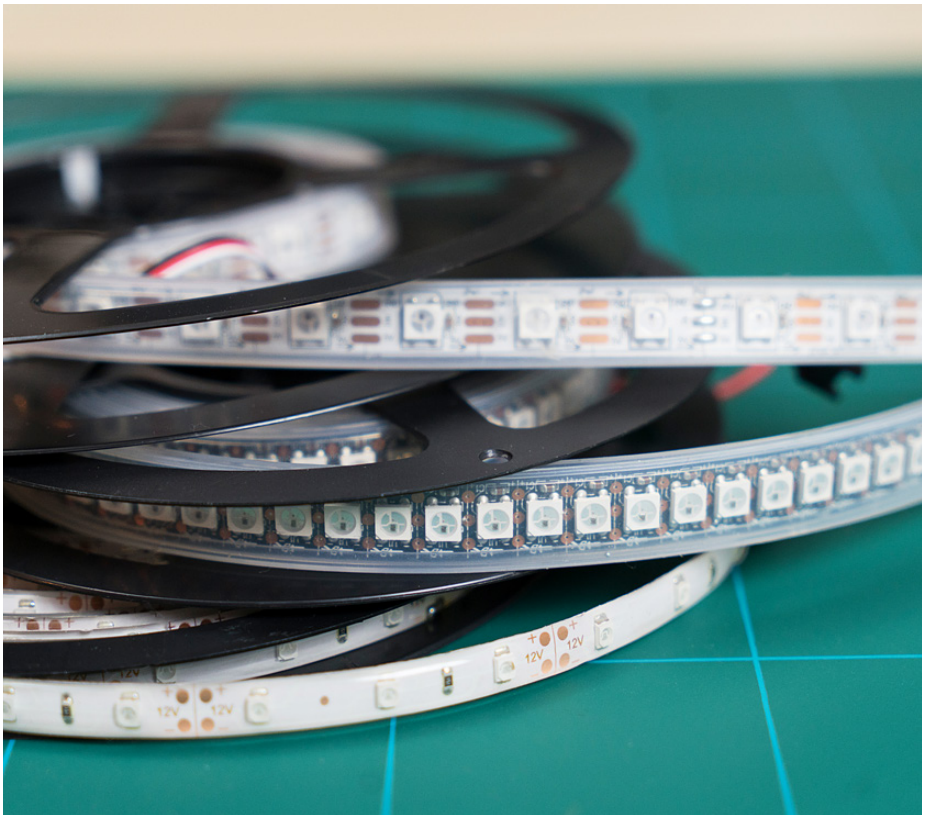
Before we get started with wrapping everything in shiny lights, let's go shopping first! You won't need that much, only some LED strips, a few microchips, batteries and some soldering stuff. Let's quickly go over some of the essentials.

Digital RGB LED Strips

This is what you will need most of so it's a good idea to try and find a good supplier. Adafruit offers strips with 60 LEDs per meter for around 25 USD. The version with only 30 LEDs comes cheaper but is also not as bright. Don't want to buy from Adafruit? Simply Google search 'digital RGB LED strip **WS2811**' or '**WS2812**'. Other than that just make sure they run with **5V**.

Apart from strips you can also get single RGB LEDs, Neopixel rings, pixels that you can sew onto fabric, a matrix and more! Don't limit yourself to LED strips if you want to try out something else. All of these products are programmable as well and work with the same hardware and software.

'Digital RGB' means that you can tell each LED to light up in a specific color by using a microcontroller board. There are also regular LED strips that are *not programmable* and only light up in certain colors. So just be carefully what you buy.

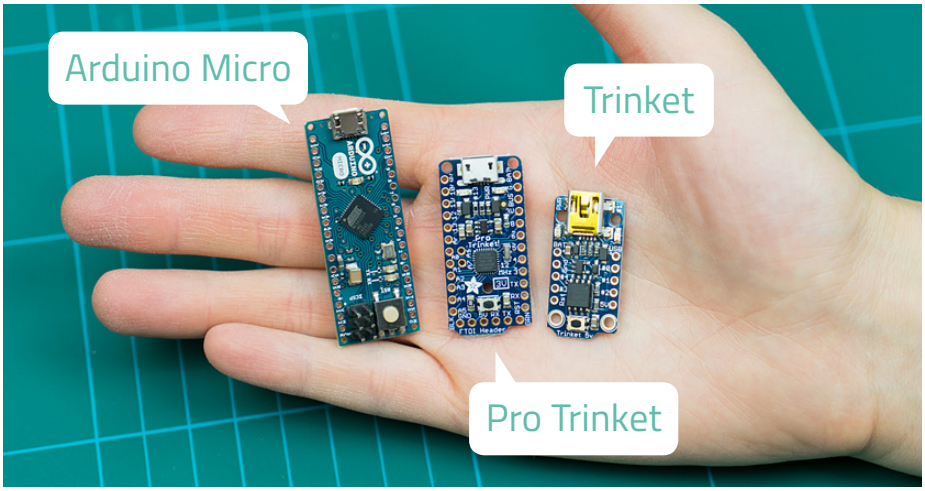


Arduino Uno + Starter Set

This box set is marked as 'optional' on your shopping list, but still very interesting if you want to try a few things out without fully committing yet. The Starter Set will show you how to work with the **Arduino Uno** microcontroller board, LEDs, motors, switches and introduce you to the basics of programming and setting up a circuit. You can also just buy the Arduino Uno separately for around 10-15 USD. We'll need it often!

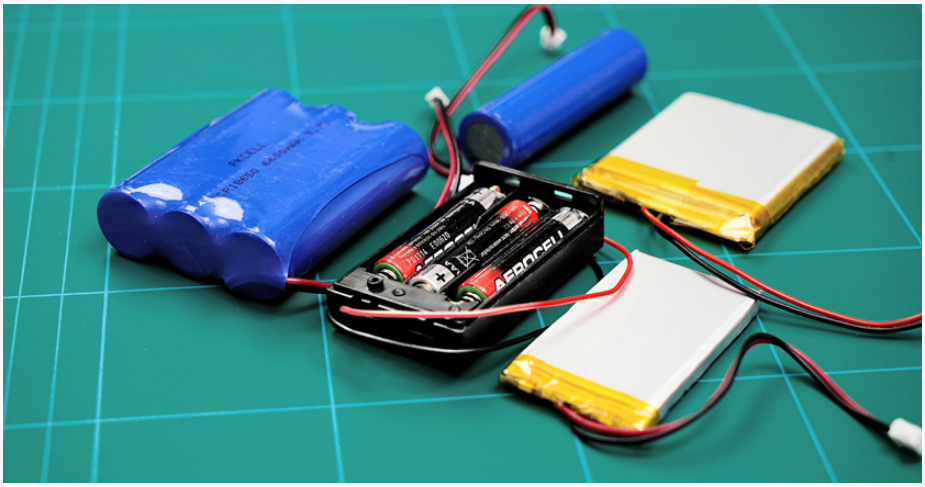
Arduino Software (free)

Instead of just soldering the LEDs together, we'll have to program them this time! The right software to do that is the **Arduino IDE**. You'll also need a USB cable and a computer. The program runs on Mac as well as on a PC, though it might be an advantage if you've passed Windows 98 already. You'll find out where to get and how to install the software by continue reading or by checking out **www.arduino.cc** right now.



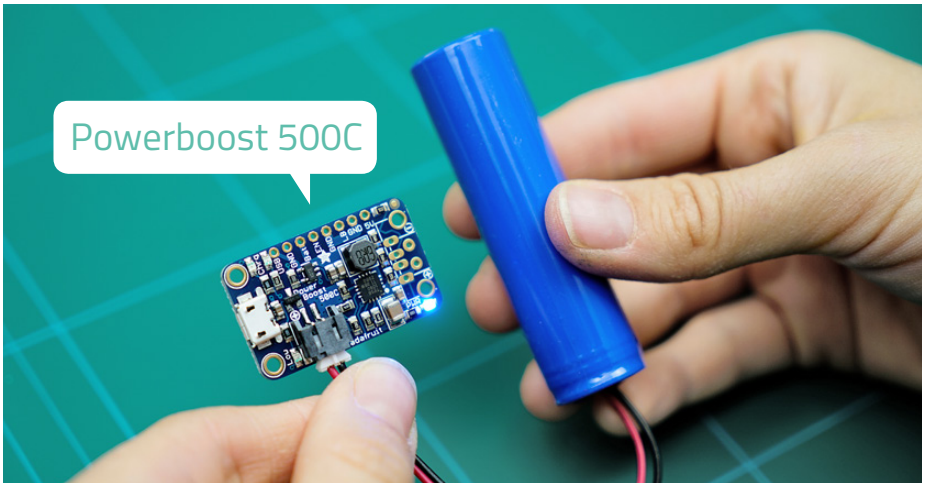
Microchips

The Arduino Uno is too big to put into most costumes and props, so you'll need a smaller chip as well. There is a huge variety of great products with all kinds of different functions. The ones I use are the **Adafruit Trinket 5V** and **Pro Trinket 5V**, which are counterparts to the almost equally small **Arduino Micro**. They are pretty powerful, provide plenty of pins even for bigger projects and are super compact – perfect for cosplay! I'll mainly stick to the **Trinket** for my examples because Adafruit offers a lot more tutorials for it.



Batteries

To keep your costume and light installation mobile, you will need an equally small power source. I use rechargeable and environment friendly Lithium Ion (LiPo or LiPoly) batteries that I just fill up again when they're empty. This saves money and reduces waste at the same time! While battery holders for simple AA or AAA batteries work too, they will not keep your LED strip lit up for long. Adafruit's lipo batteries have 3.7V output and start in super tiny shapes of 100mAh up to 6600 mAh. Luckily when connecting them to a Trinket or Powerboost (see below), the chip will automatically convert the 3.7V to the 5V that are necessary for the LED strip.



Charger

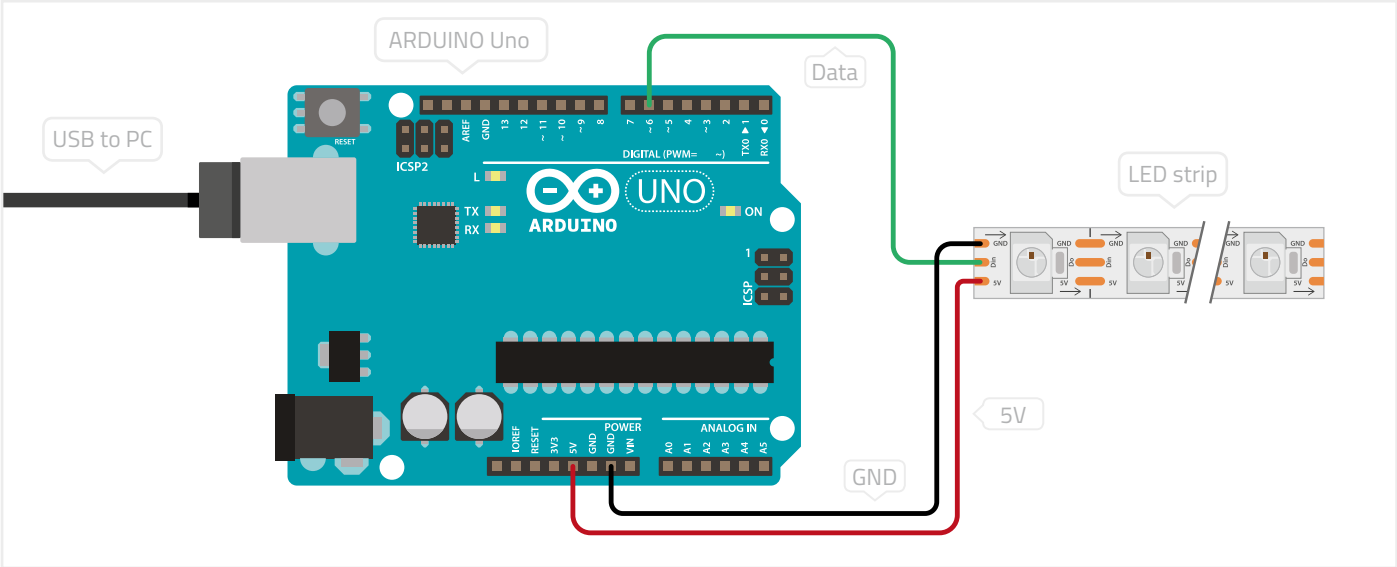
You'll need a special chip to fill up empty lipo batteries. The one I use is called the **Powerboost Charger**. You can choose between a Powerboost 100, 500 or 1000 – the number indicates the charging speed in mAh. While it's not required to power just a few LEDs, you'll definitely need one if you want longer LED strips. More about this in a later chapter.

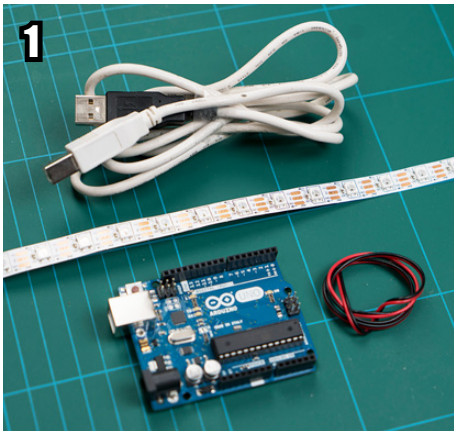
Soldering tools & more

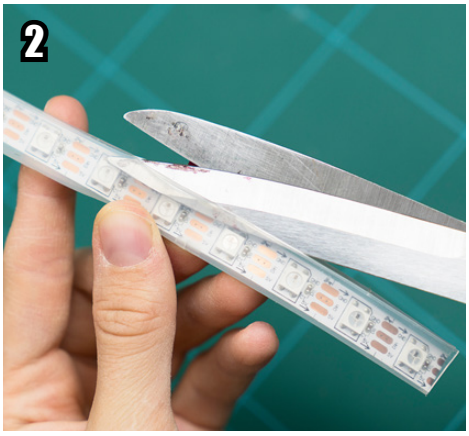
This stuff is pretty basic and I already covered most of it in my previous LED book. Since you'll need to solder your circuit together, make sure you have a soldering iron, cables, lead free solder, a wire striper, a helping hand, tongs, insulation tape and shrinking tube. I also recommend that you buy a few slide switches, push buttons and plugs. Most of these are super cheap and it always pays off to have more than you need. The time will come when you hope and pray that there is still a slide switch in your collection at 2am in the morning before the con.

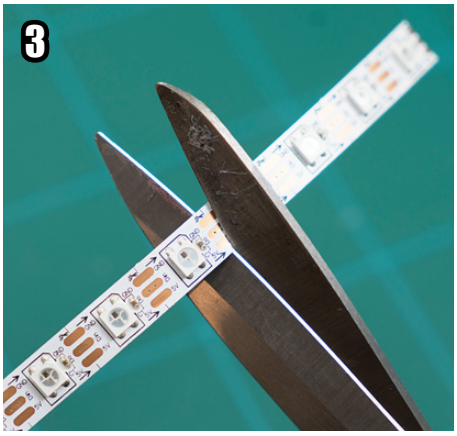
Step 1: Let there be light!

Compared to wiring a simple static LED circuit, lighting up a LED strip is a little bit more advanced. After breaking it down however it's still basically just soldering a few wires to the right places. So, just take a look at the drawing below and let's start!





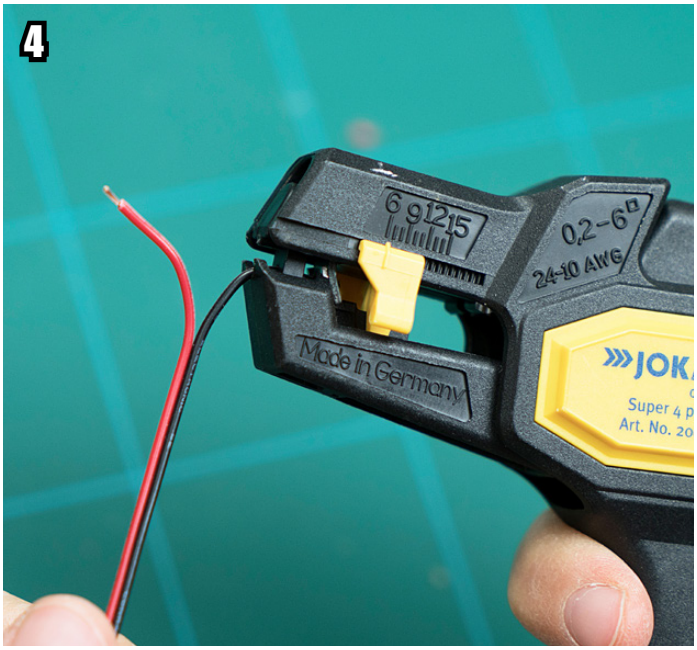


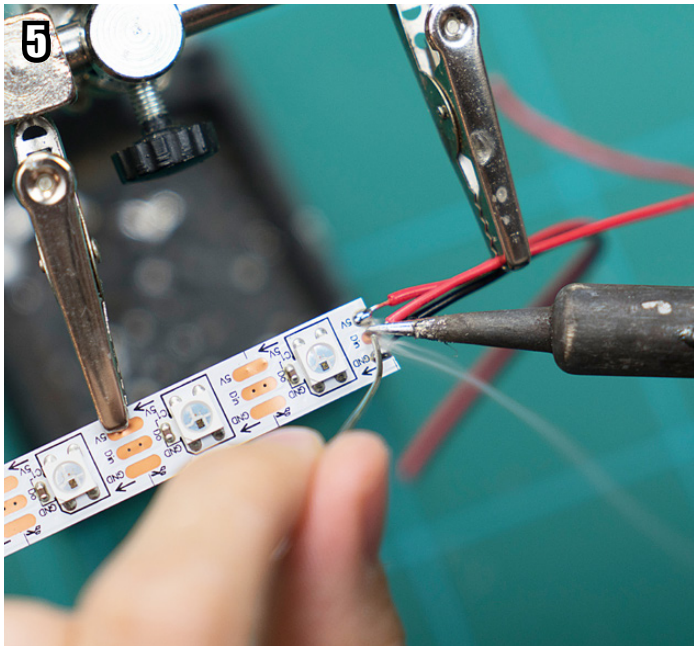


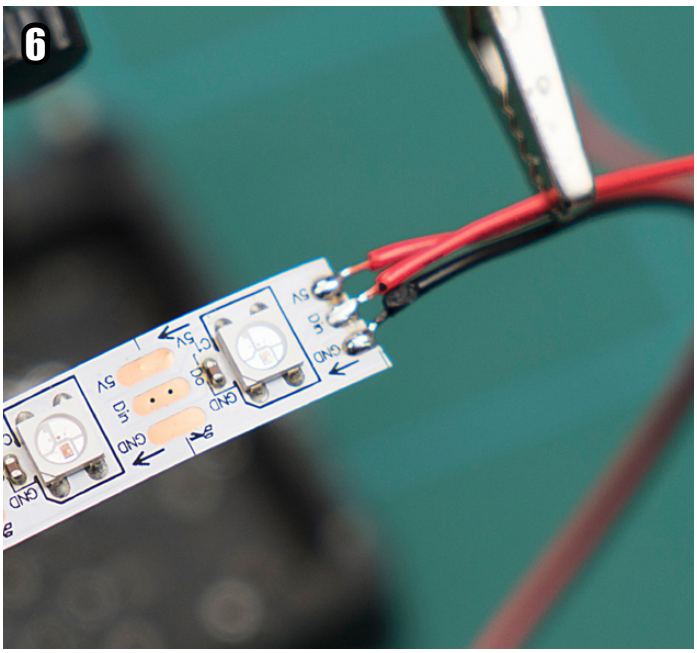
To get a better understanding of what I'm about to show you I recommend buying an **Arduino Uno** and following along with this example. All the code examples in this book work with this chip as well so it's perfect for beginners! To light up our first LED strip we also need a fitting USB cable, some wire and basic solder equipment **[1]**. LED strips generally come in rolls, so if you want to keep your circuit small and compact instead of using the whole thing, first get rid of the plastic water protection (not all of them have that) **[2]** and then cut the strip along the marked lines with regular scissors **[3]**. These kinds of strips can be separated after every pixel, but take care - there are also some that only allow you to cut after every third LED for example. Just look for the correct markings.

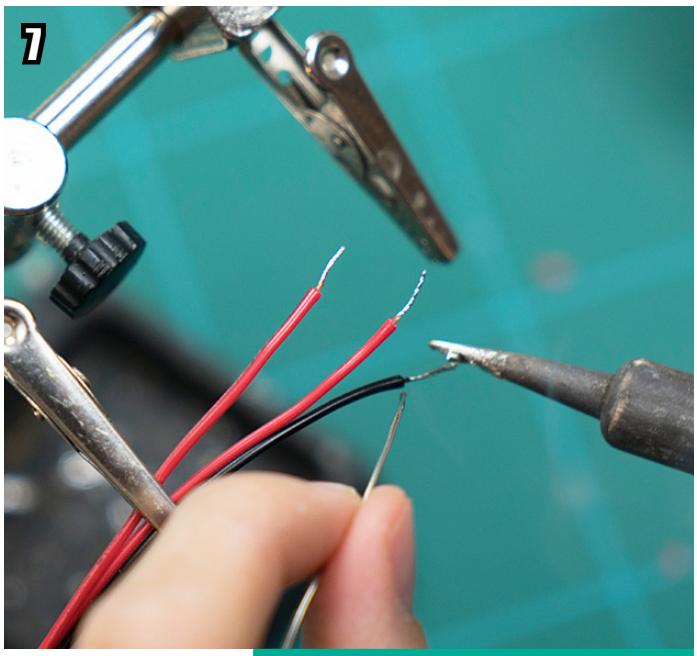
KEEP IN MIND

Always be careful when working with electronics, soldering irons and electrical current. Accidents happen faster than you think! Stay away from the hot tip of your soldering iron and turn if off when you're done working. Don't touch the metal contacts of your cables or your microchip after you've plugged the power in. Try to avoid short circuits and please don't stick your tongue into a power plug!









Next, grab three wires and strip their ends. You can either do that with a regular cutter or use a much handier wire stripper **[4]**. Try to get rid of only around 3mm (0.1inch) of insulation so you don't need to cover up everything again later.

Now solder all three wires to the bronze pins on your LED strip **[5]** and make sure the arrow leads away from the pins. You can wire them up in only one direction and the other won't work. I tend to use a double colored wire. Red goes to **5V** (plus pole) and black to **GND** (minus pole).

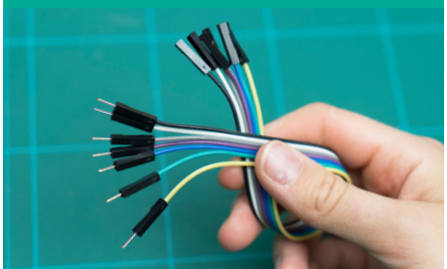
Finally add a third, separate cable in either red, black or another color to the data pin (**Din**) in the middle **[6]**. Using different colors greatly reduces the chance of getting confused which cable belongs to which pin.

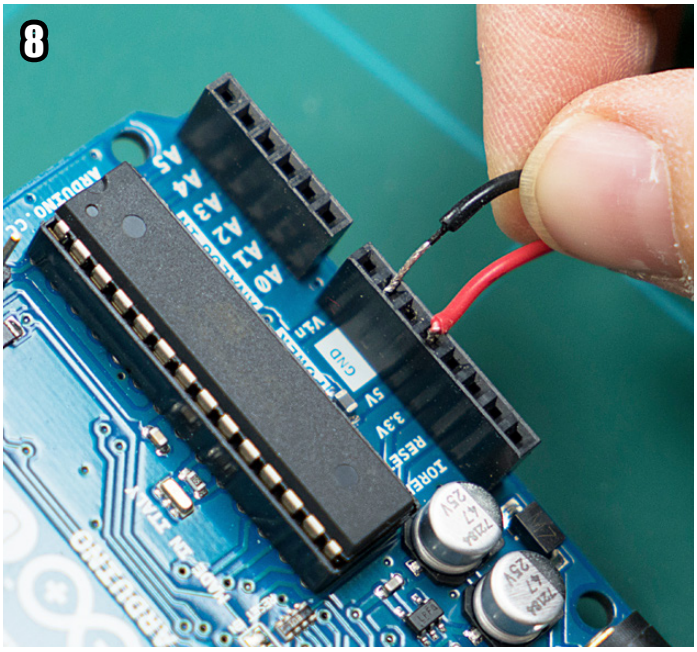
Adafruit actually suggests in their comments to wire a 300-500 Ohm resistor to the data pin to minimize the 'burnout risk'. I honestly never did this nor did I ever have a pixel burnout on me. It's up to you if you want to do it or not.

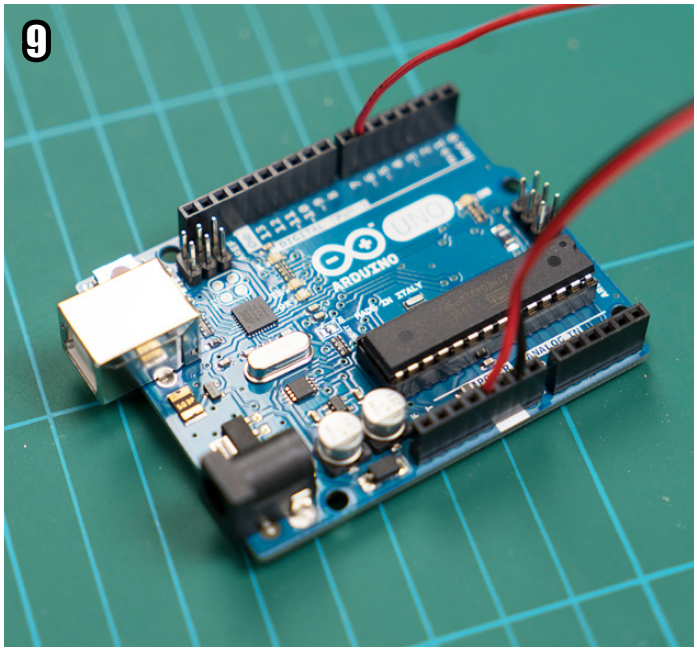
Let's keep the next part simple too. Strip around 1cm (0.4inch) at the ends of your cables, twirl the copper threads together between your fingers and cover them with a thin layer of solder **[7]**. These pointy ends should be straight, solid and thin. We're gonna use them as plugs now!

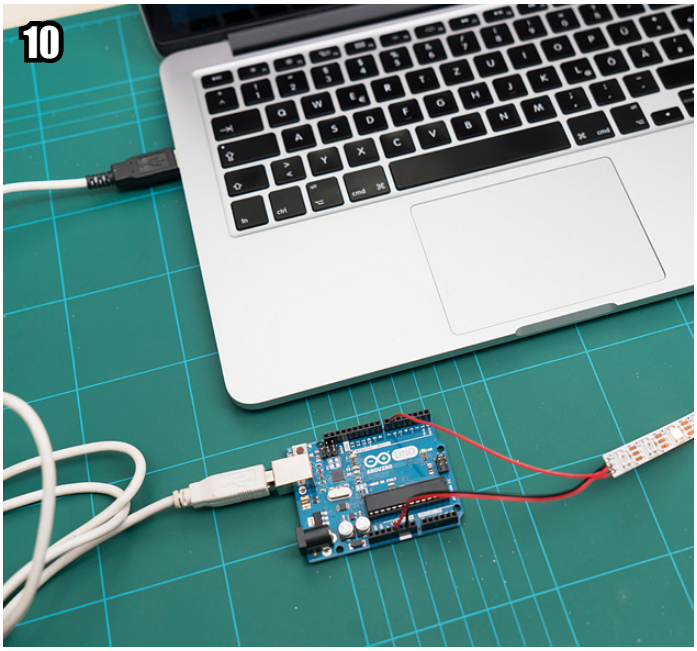
NOTE

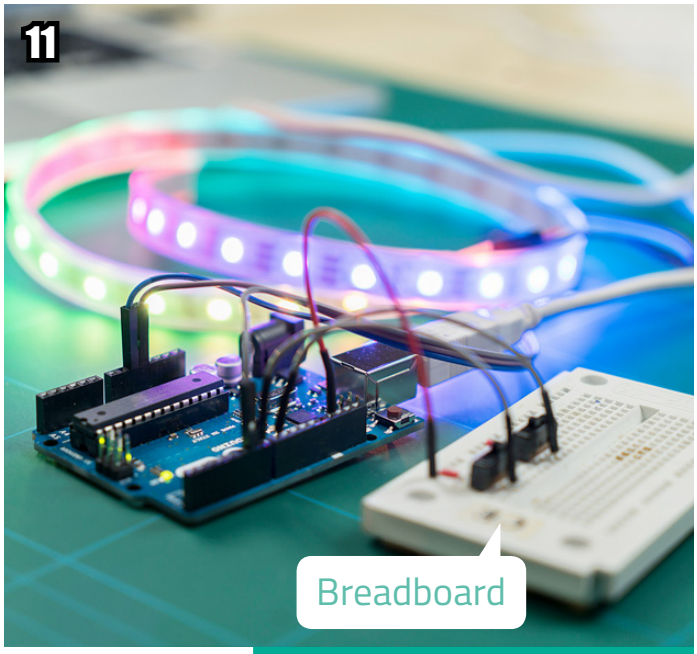
If you're a technician you will probably rip your hair out, disgusted by this step. I *guess* it's more professional and durable to solder proper jumper plugs to the end of the cables. But covering the wire with solder does the job too. Leave me alone!











It's time to plug your *beautiful* cables into the Arduino **[8]**. Just follow the schematic from before. 5V goes to 5V, GND to GND and pin number 6 is usually a good choice for the data cable since many codes use it as a default **[9]**.

By plugging a USB/FireWire cable from the Arduino to your computer your LED strip is all ready to light up. It's powered by your PC/Mac now and eagerly awaits your commands **[10]**.

Even if this setup is probably too big to put into your costume or prop, it's more than enough to play around with the software (see following chapter) and test out some neat LED animations. The good thing is that you won't have to think about batteries since it's all conveniently powered by the USB connection.

NOTE

I just love my Arduino! There is no faster way to try out an idea or see if your code adjustments work. Connected to a breadboard this chip is just perfect to build up new circuits without any soldering. Plugging in switches, additional chips and batteries happens within seconds. Additionally I often take a little box including the Arduino, a LED strip, wires and switches with me when we're traveling. It's great to be able to keep on working on my props and costumes even when they are not with me.

Now that your microcontroller is
ready and your LED strip connected,
it’s finally time to let it shine!

This is where the controller software comes
into play, the **Arduino IDE**.
Naturally the next few pages will show
a lot of code, but don’t you worry: It's just some
editing and a little bit of hammering your
head on the keyboard.

You'll get the hang of it quickly!

Gotta code them all!

Lucky for us, the Arduino is a chip designed mainly for kids, beginners and hobbyists – perfect for cosplayers! Setting up the hardware was already pretty easy, right? Now we just need to download the Arduino software (Arduino IDE) from **www.arduino.cc**, so simply click on the download button and install the most current version.

Before starting the software, make sure your chip is connected to your computer via USB. This ensures the program will recognize your controller and you don't have to set up anything extra. If you do however encounter a problem at any point, just check out one of the tutorials on the Arduino website.

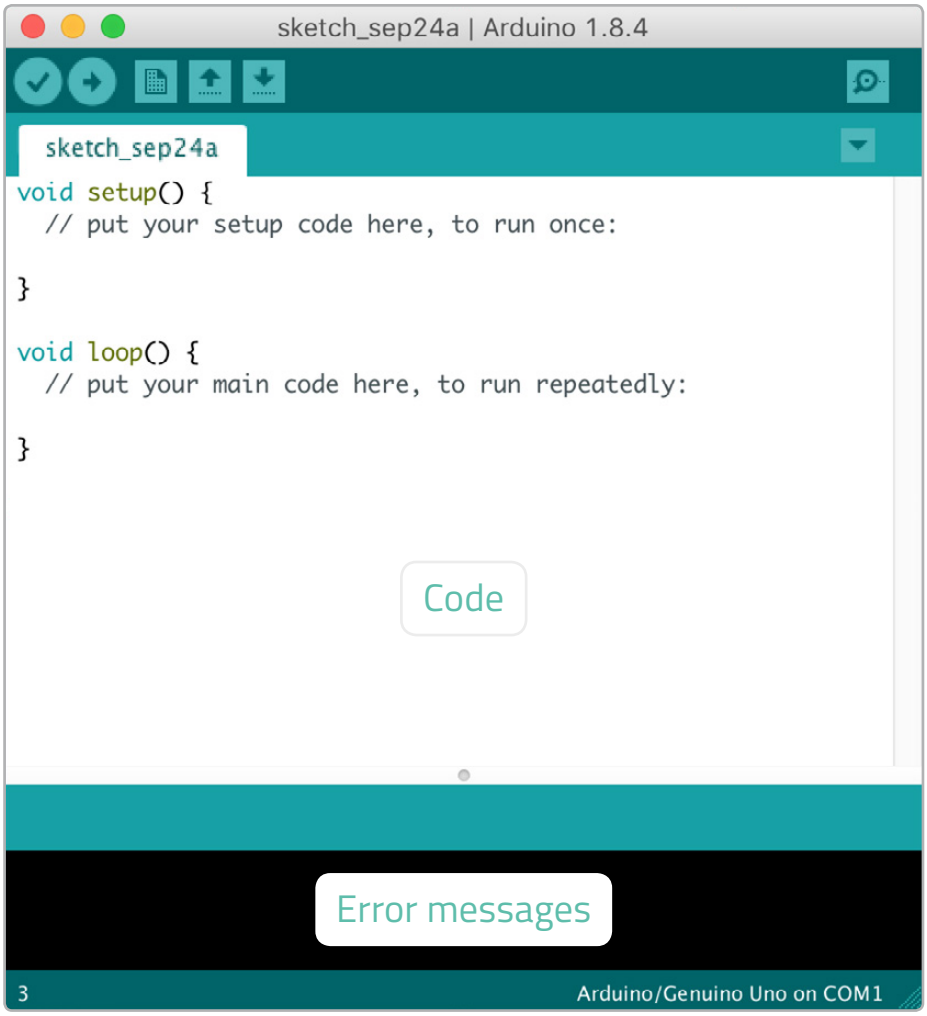
Now, let me introduce you to the program and it's most important functions you will need.

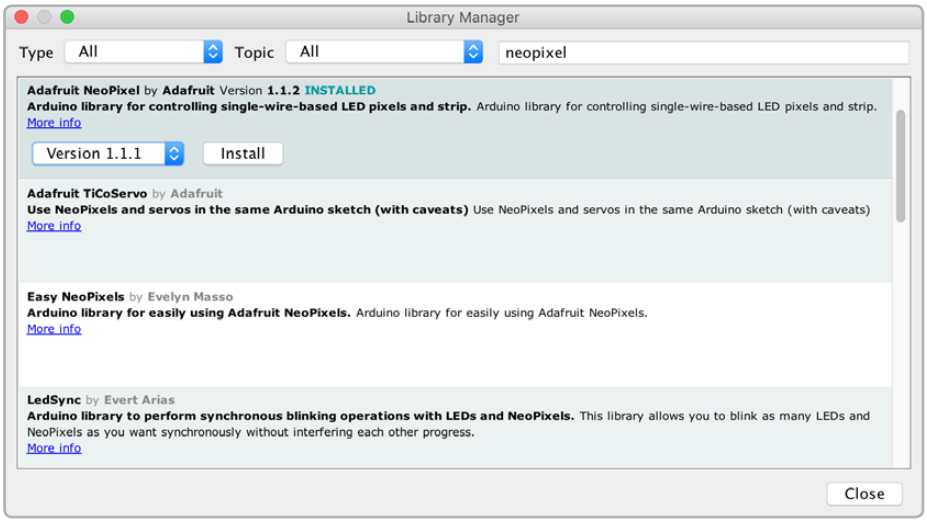
The Arduino IDE allows you to write and edit code, upload it to your microcontroller (board) and let your LEDs do a lot of fancy stuff. Since I want to keep it simple, we'll mainly concentrate on the the *edit* part. However it never hurts to get some basic know-how!

File Edit Sketch Tools Help

When you open the program, you'll find 'Arduino, File, Edit, Sketch, Tools, Help' in the top bar. It's a wild guess but I'm pretty sure this is not the first time you're sitting in front of a computer, so I won't explain what 'file' and 'help' does. The first step for you now should be to add a **library** to your program folder. Libraries are collections of pre-written code for all kinds of functions your microcontroller shall perform for you. They have everything from letting a single LED blink, over reading all kind of sensors to even controlling the motors in a little robot.

The library I mostly work with is called **Adafruit_Neopixel**. To add it, just go to **Sketch** > **Add library** > **Manage libraries** and search for its name in the upper right corner. Now click on '**more info**' and hammer the install button. Afterwards you should find a new folder under **File** > **Examples** > **Adafruit Neopixel**, that contains codes like buttoncycler, simple and strandtest. The one we will use the most is called **strandtest**, so just open this one now! These pre-written codes are called 'sketches'. by the way.





When you open strandtest for the first time, a bunch of confusing looking text will appear. Hey don't close the program yet! It's really not that difficult! See all those lines starting with two slashes (**//**)? These are just grayed out comments for some additional explanation. Just delete these lines if you want. See? It already looks a lot more inviting. All of the Adafruit sketches have these comments included, so even people who can't read code can get a general grasp of what's going on. The rest of the lines that appear in black, blue, green and orange are commands that will tell your microcontroller and LED strip what to do. The different colors are just markers for different tasks in the code.

So I bet by now you are really curious what the strandtest code actually does? Just click with your mouse on the little green arrow in the top left corner and wait until the program says '**upload successful**'. This means the computer successfully sent the code to your Arduino and your LED strip should now shine and blink in some very lovely colors and animations.

It's okay to stop and marvel at the pretty LEDs for a minute. Looks great and it wasn't that complicated, right? Now let's see how we can gain some control over these lights. Just keep reading!

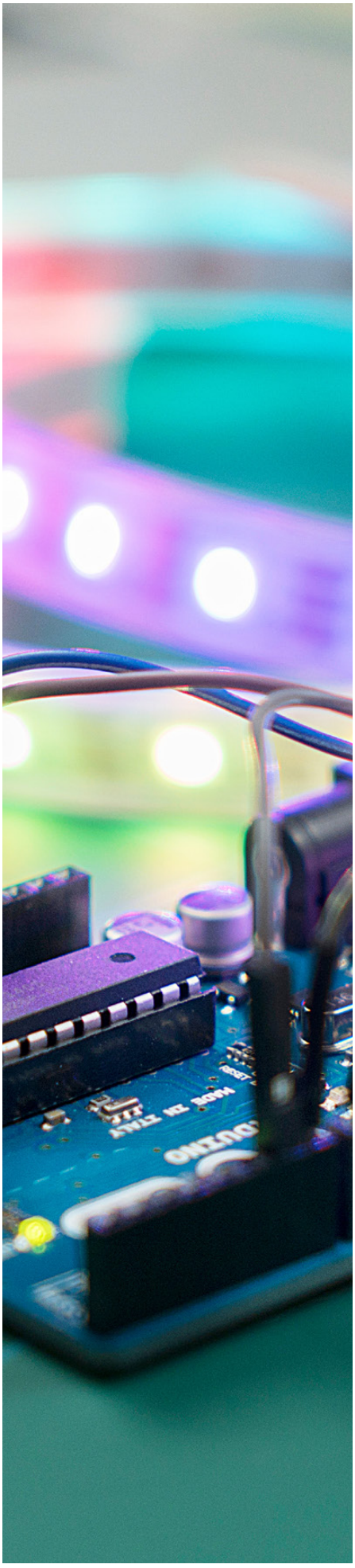
NOTE

The software should automatically detect which controller is connected. If this doesn't happen for some reason you can always just go to Tools > Board and just pick the right one.

If your upload was successful, the black window in the lower area isn't very interesting. As soon as we start to edit code however, you'll notice that it's purpose is to show you if an error occurred. Jump to the troubleshooting chapter at the end of this book or Google search for your error message to solve it.

All those pretty programs!

You've soldered your first circuit with a microcontroller and uploaded the strandtest program to your chip. Congratulations! You now have a crazy blinking LED strip! I know it's very pretty, but wouldn't it be even more awesome to be able to understand and control what it does?



The full code you get for the **strandtest** example is a little bit longer than what I show below, but I simply left out everything we don't need at the moment. Even if it's the first example in this book, it's actually still the same code I use for about 90% of my projects. Just changed slightly. First however, let me translate this chaos for you!

This first line of the program just loads the library **Adafruit_Neopixel** and all its functions.

```
#include <Adafruit_NeoPixel.h>
```

These lines tell the microcontroller, that you've connected a LED strip (doesn't have to be a Neopixel) with 60 LEDs to pin 6. You'll have to change these numbers when you use more or fewer LEDs and if you've connected them to a different pin. While defining the right pin is essential, it's actually okay to write down a different amount of LEDs than you actually wired up. This mostly just changes the speed of the light animation.

```
#define PIN 6
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
```

This section just tells the LED strip to be prepared for incoming signals from the microchip. It's basically just a *'Get ready, it's time for fun!'*

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

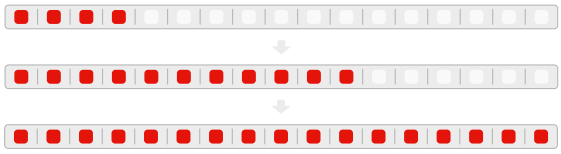
Here is a collection (loop) of all the functions that your sparkling LED strip is currently cycling through. Think of it as a **table of contents**, listing the names of the different animations. You'll find the same descriptions a little bit further down again but this time with more code describing what they do.

```
void loop() {
  colorWipe(strip.Color(255, 0, 0), 50);
  colorWipe(strip.Color(0, 255, 0), 50);
  colorWipe(strip.Color(0, 0, 255), 50);
  theaterChase(strip.Color(127, 127, 127), 50);
  theaterChase(strip.Color(127, 0, 0), 50);
  theaterChase(strip.Color(0, 0, 127), 50);
  rainbow(20);
  rainbowCycle(20);
  theaterChaseRainbow(50);
}
```

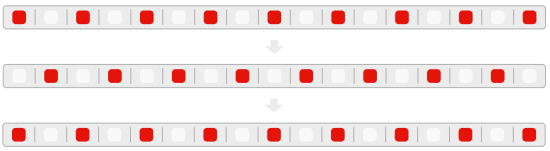
A sketch always consists of a section to **define** the pins and what's connected to them, a **void setup()**, which tell your connected parts what they are and a **void loop()**, which continuously cycles through the programs you wrote. Every sketch needs these three parts to work.

Now let me explain what each of these lines do:

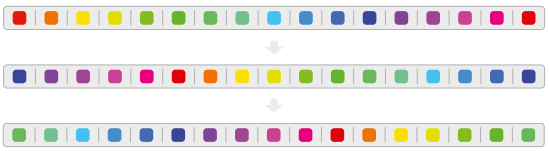
colorWipe is an animation that fills all pixels, one after the other, with a specific color, starting with red (more about the numbers in a bit). Afterwards comes green and then blue. It's super handy if you want to light up your costume or prop in just one color without any other animations. The number at the end of each line is the **delay**, which tells you how fast the strip fills up.



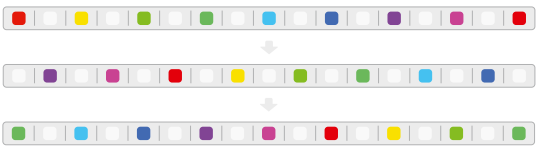
Next **theaterChase** lets the LEDs sparkle in a specific color. It's like those neon signs at a country fair. This animation might be useful if you... well, if want to let something sparkle!



rainbow and **rainbowCycle** are pretty similar and create a smooth color change through the entire rainbow spectrum. These animations are the ones I use for pretty much all of my projects. I'll explain later, why! The delay number here defines how quick your lights change color. If you lower it, the animation will be faster.



And finally **theaterChaseRainbow** turns your LED strip into a blinking rainbow animation. If you need some crazy disco lights, theaterChaseRainbow is for you.



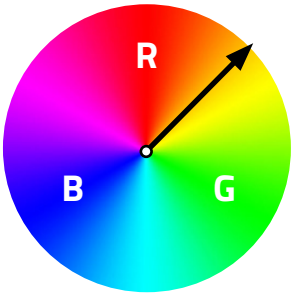
To choose which program(s) you want to use, simply disable all the others. To do this, just write two slashes **//** before the lines and the whole text will be deactivated and turned gray, just like the comments in the code. This is also called 'to comment out' something. If I only want to use the blinky rainbow program, it would look like this:

```
void loop() {
  //colorWipe(strip.Color(255, 0, 0), 50);
  // colorWipe(strip.Color(0, 255, 0), 50);
  //colorWipe(strip.Color(0, 0, 255), 50);
  // theaterChase(strip.Color(127, 127, 127), 50);
  // theaterChase(strip.Color(127, 0, 0), 50);
  //theaterChase(strip.Color(0, 0, 127), 50);
  // rainbow(20);
  // rainbowCycle(20);
  theaterChaseRainbow(50);
}
```

At the bottom of the code you'll find the additional lines. These functions describe which colors the animations **rainbow**, **rainbowCycle** and **theaterChaseRainbow** are going to use. These standard settings lead to the rainbow colors you see when you upload the code to the strip. They are described in three color channels: **Red**, **Green** and **Blue**, or **strip.Color(r,g,b)** in the code. I marked them in color so you can see where they are.

```
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;  // This inverts the color animation
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  } else {
    WheelPos -= 170;
    return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  }
}
```

0 is the minimum for each color's brightness (or better intensity) and 255 is the maximum. So **strip.Color(0,0,0)** means the strip is turned off, while **strip.Color(255,255,255)** means it's completely white. Red is **(255,0,0)** and blue is **(0,0,255)**. You get the idea. It's pretty easy to set up red, green and blue this way, but getting a different color will require you to mix the channels. Just do a quick Google search for RGB color guides and you will be good to go!



Back to the rainbows! As you can see, the color of each pixel constantly changes with a slight delay to the pixel next to it. This creates the illusion that a rainbow is running over the LED strip. The animation is described by the **WheelPos**(ition), which creates a number between 0 to 255 depending on the position at the time. It's like the pointer of a clock, spinning in wheels, again and again.

The entire function is basically just math that tells each LED which color mix it needs to show at any given time. For example: If **WheelPos** is **1**, the color code would be **strip.Color(252,0,3)**, which would be bright red with just a tiny hint of blue. When WheelPos gets higher it slowly changes to blue and then to green.

If you want to change the colors yourself, this is how you do it: Should your rainbow only shine in red, change the formula for green and blue to **0**. It will look like this.

```
strip.Color(255 - WheelPos * 3, 0, 0)
strip.Color(0, 0, 0)
strip.Color(WheelPos * 3, 0, 0)
```

Blue would look like this (see original code at the top of the page for comparison).

```
strip.Color(0, 0, WheelPos * 3)
strip.Color(0, 0, 255 - WheelPos * 3)
strip.Color(0, 0, 0)
```

I'm sure you can guess how green would like, right? It's really not that difficult!

As you can see, you don't need to learn how to code or write a program completely from scratch. As long as you know how and where to change a few numbers you can get exactly the result you want! That's what I'll show you over the next few pages!

Understanding the last few pages without any programming knowledge is not so easy (it's an *advanced* book after all) so take your time. Make little changes, step by step and always keep on uploading the code to your Arduino to see what happens. I also learned everything I know today by doing simple experiments like these. You'll understand the functions a lot faster if you play around with them and soon you will be even able to try out more sketches.

If my explanations were too short, you didn't understand everything or just have some additional questions, check out the Adafruit Überguide and scroll to the section for the Neopixel library. It's a good place for additional info:

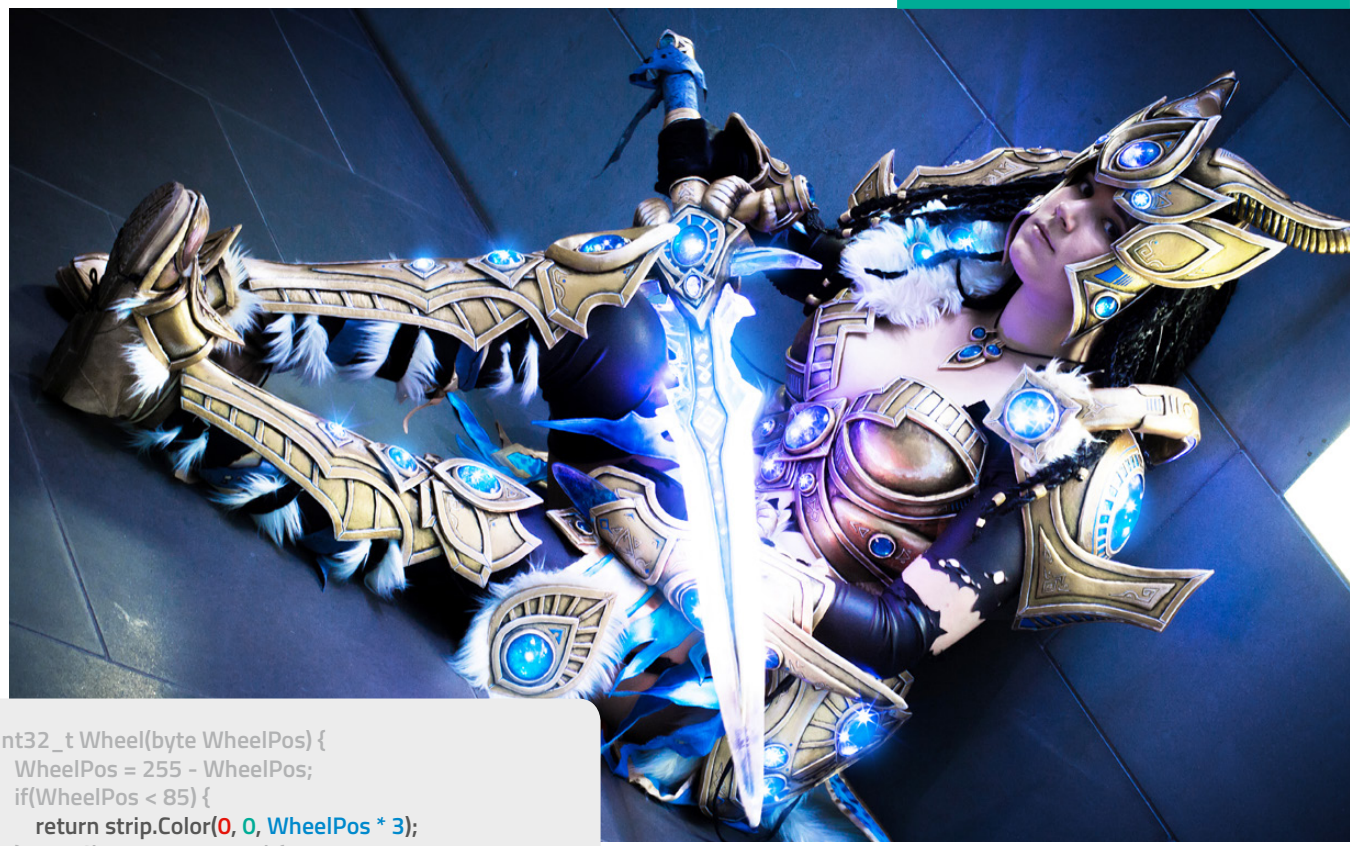
learn.adafruit.com/adafruit-neopixel-uberguide

While the Adafruit Neopixel library already contains plenty of cool effects, you might notice it's also missing a few things. Check out the 'learn' section on their website and get inspired by new ideas, flashy projects and other interesting animations. A good alternative is the **FastLED** library, which includes plenty of animations such as fire, pulsing effects or flashing and blinking. Just add it via **Tools > Managing Library**. You also might discover interesting animation tests on Youtube, codes from different electronics websites or simply want to try out something another cosplayer recently built. Don't limit yourself to what you already have. Cosplay is about growing and learning something new and you should handle it the same way when it comes to animated LEDs.

You can always just ask the community! I'm sure there is someone who can help you out!

NOTE

My friend Selina once built a sword with an animated, lit up blade. I did the programming for her, but just kept it super simple. She wanted to have a blue color running through her sword so I simply deactivated every program except the **rainbowCylce** animation. Then I turned off the red and green color channel in the functions (see below) and as you might guess the resulting effect made her super happy! It literally only took me a minute!



```
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(0, 0, WheelPos * 3);
  } else if(WheelPos < 170) {
  WheelPos -= 85;
    return strip.Color(0, 0, 255 - WheelPos * 3);
  } else {
  WheelPos -= 170;
    return strip.Color(0, 0, 0);
  }
}
```

Make your circuit smaller

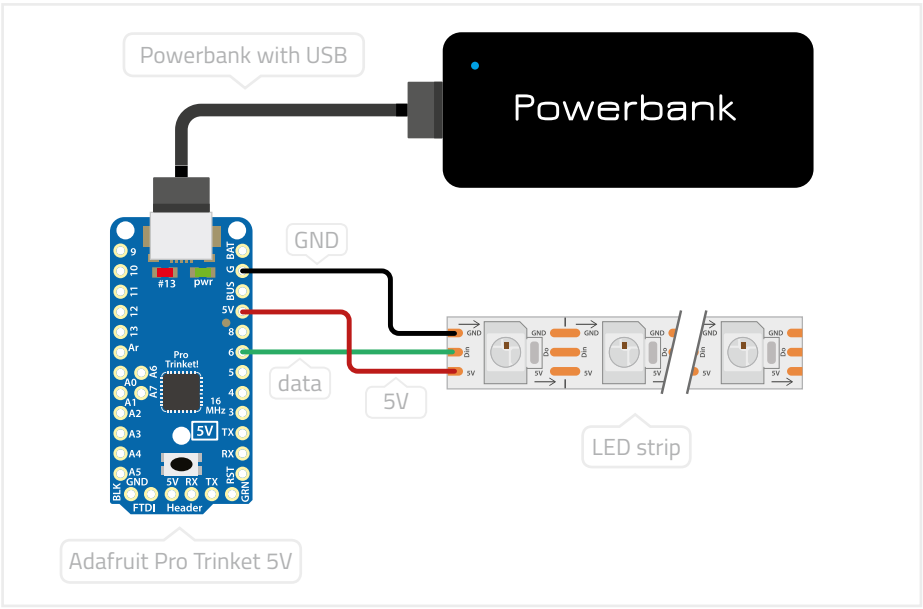
Now you know the circuit and the code! The only problem is that it will be a little bit difficult to squish the *relatively* big Arduino Uno into the tiny grip of a light up axe. You may also wonder which power source you need to be mobile and what a real-life circuit in an actual costume looks like. For that, let's get some new cool hardware!

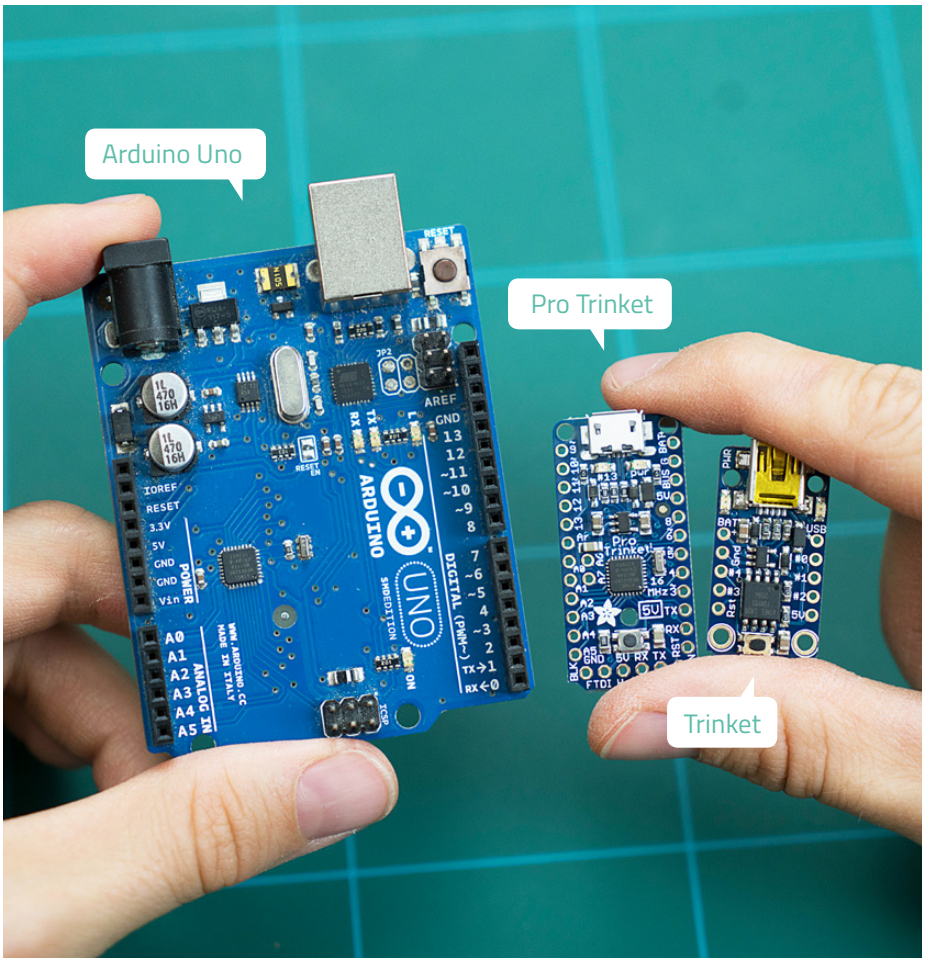
Since I'm already using the Adafruit library and their Neopixel LED strips, why not just go with their in-house microchips as well, right? (They seriously should pay me for doing this book!) A much smaller alternative is the **Adafruit Pro Trinket 5V**, which is just as powerful. Its little brother the **Trinket 5V**, doesn't have as many pins and has far less memory, but works great for smaller projects as well. Both provide enough pins to run several LED strips simultaneously, give you many options to connect a power supply and come in a nice, compact format. Besides switching to a smaller chip, we are also upgrading to a portable battery by using a **Powerbank**. You know, the things you can charge your phone with! This way you just need to turn your Powerbank on to activate the animation of the LED strip. As you know, fewer lights require less power. Therefore, this is a great solution to run simple and compact circuits with a handful of LEDs.

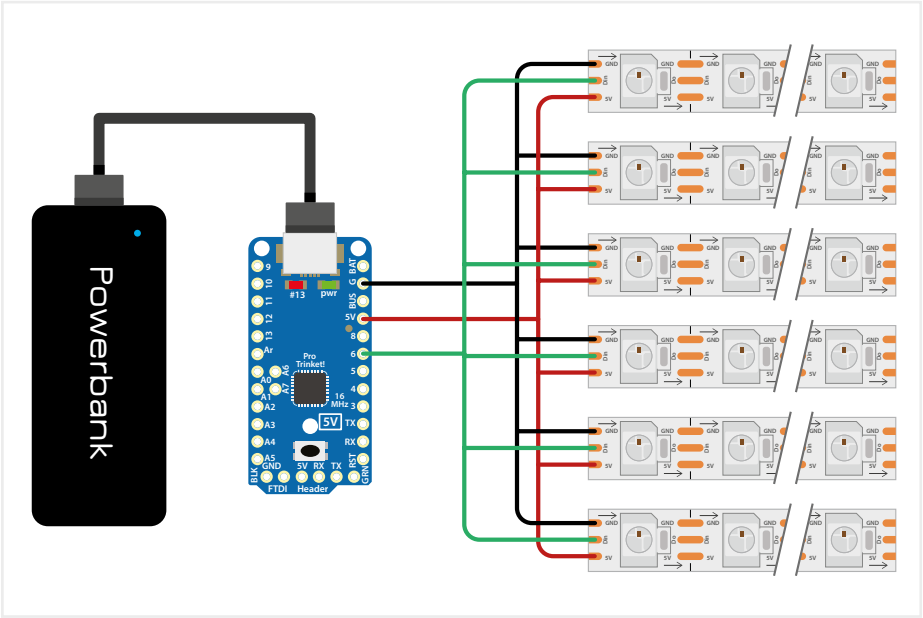
Aside from the power supply, the chips have a few minor differences to the Arduino Uno. Since they're produced by a different company, you need to adjust the Arduino IDE software so it will recognize the Trinket as well. Luckily you already did this by installing the Neopixel library, so that was easy! Additionally you *might* need to install a special driver for certain Windows systems. If it doesn't work from the get-go, you'll find a detailed step by step instruction on Adafruit's website, so take a look here if you have trouble setting up the Trinket:

learn.adafruit.com/introducing-pro-trinket

If you look at the circuit plan above you'll see that the wiring is pretty much identical to what you're already used to with the Arduino. Simply connect the **GND** and **5V** to their counterparts and **pin 6** to the **data** pin for the **Pro Trinket**. Since the regular Trinket does not have a pin number 6, just choose a different one and change the code declaration accordingly.

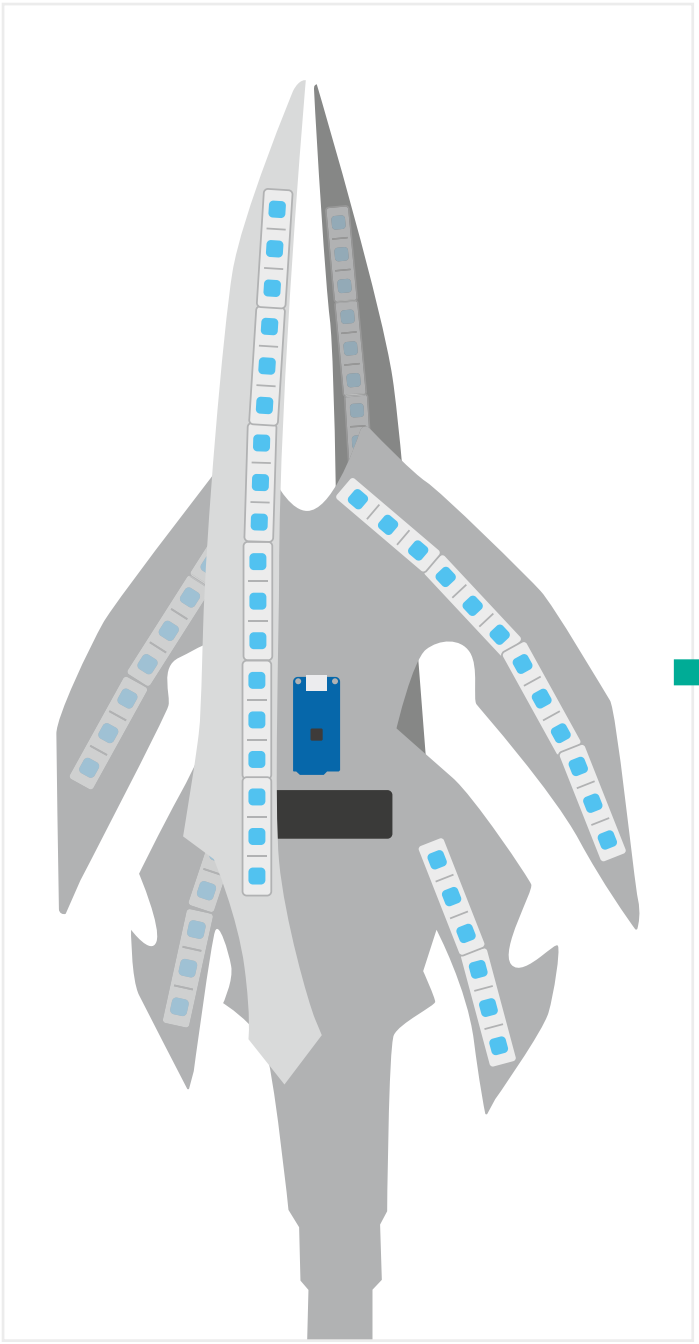


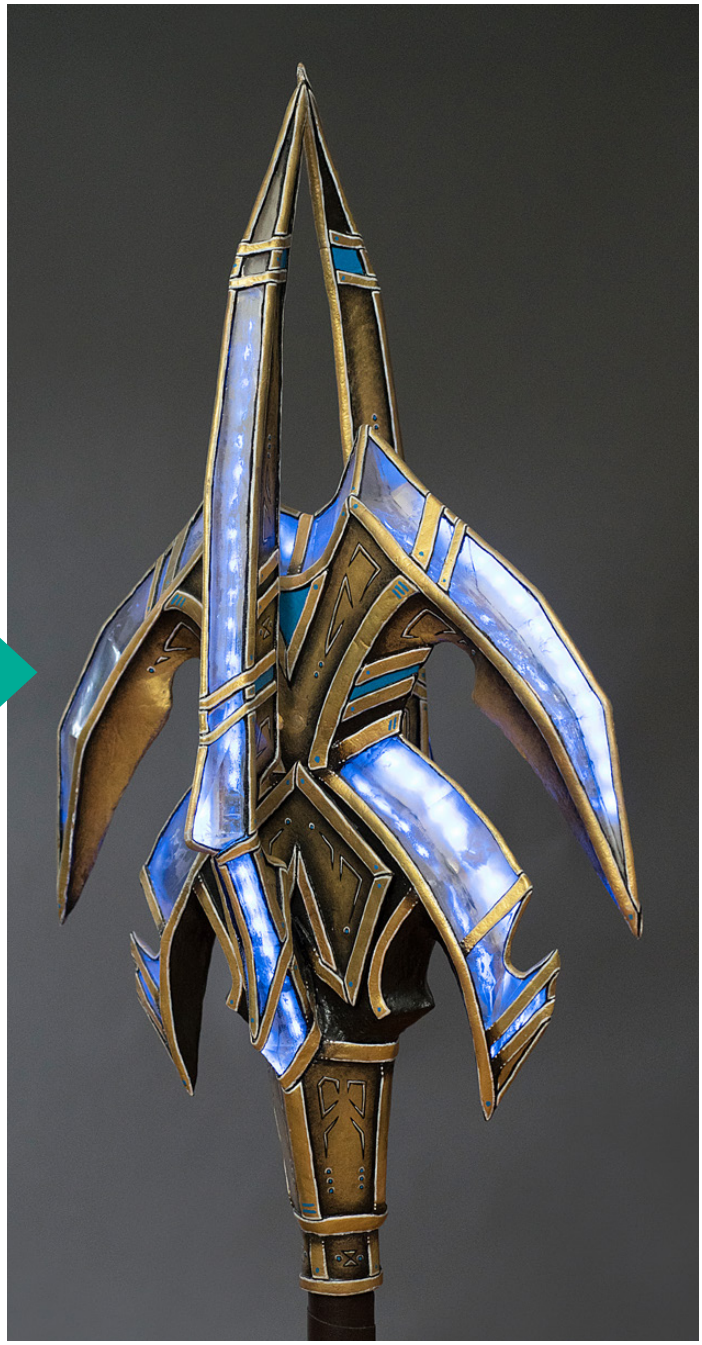


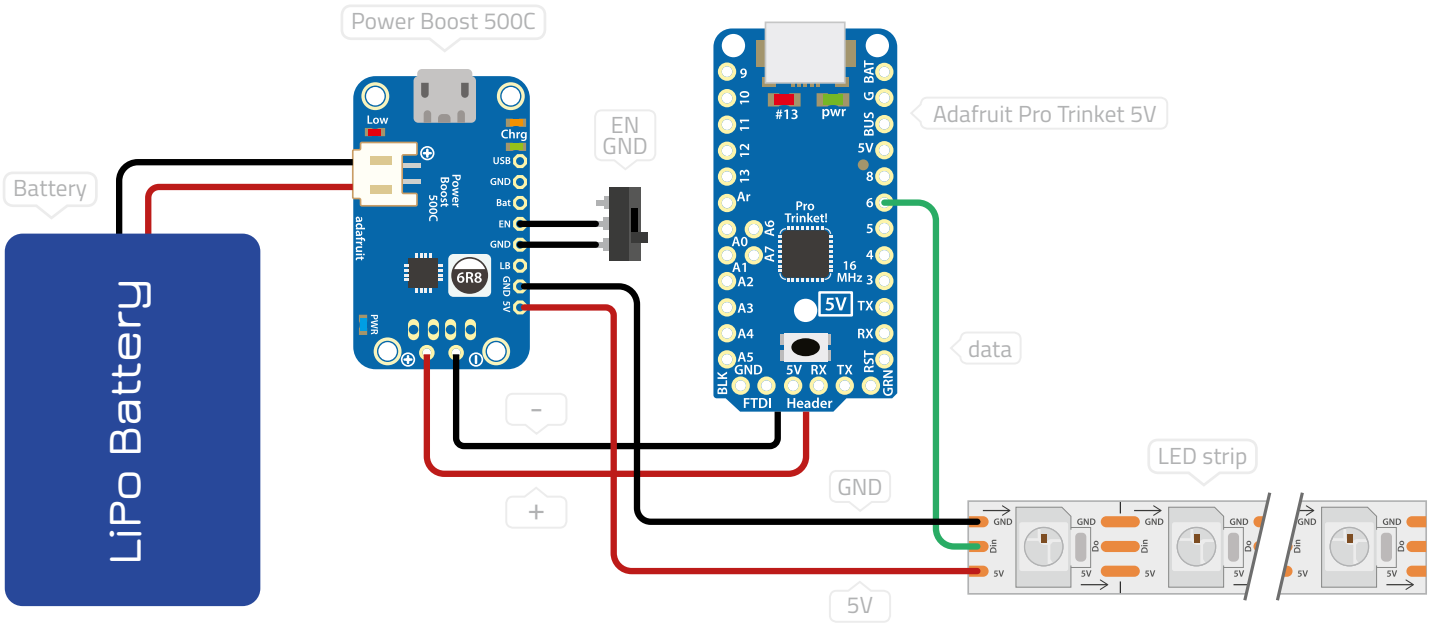


Here is a quick example:

The staff of my Protoss Wizard costume was one of my very first LED strip projects. The circuit worked exactly like I already showed you, though with a few more lights. I used a Powerbank, a Pro Trinket and simply soldered 6 LED strips in parallel to light up all the blades of the staff. Everything worked great but the animation sometimes acted a bit weird from time to time. It got stuck, started blinking or completely shut down after it just ran perfectly for several minutes. After some tests I figured out that my LED strips drained too much power (amps) over the Trinket. I wasn't able to change the circuit anymore, but instead of using all color channels, I just turned everything off except blue. Now the LED strips weren't so hungry and the Trinket stopped doing weird stuff.







You got boost power! (for more LEDs)

The Arduino as well as the Trinket can power up only a limited amount of LEDs on full brightness. The more light you'll need, the more likely your microcontroller will collapse and start acting crazy. While smaller projects will be fine, huge light up costumes will force you to upgrade your circuit.

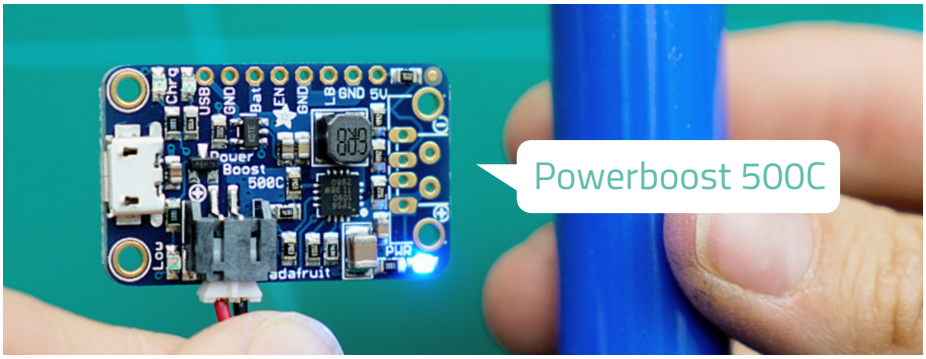
Don't worry! We're almost done with all the theory and can *finally* dive into all the exciting work examples! There is only one more thing we need to do before we have the perfect circuit! As I mentioned, the more LEDs you light up, the more power your strip draws and the higher the possibility of your chip to collapse. It won't explode or burn down your house of course, but it will start doing random stuff and give you a headache from figuring out what's wrong with it. Now you know! Time to upgrade the circuit one final time! You'll need to add a **Powerboost C** and a lipo battery, both conveniently available from Adafruit too (who would have thought?). The Powerboost is a separate chip and makes it possible for us to connect a **lipo battery** to your circuit.

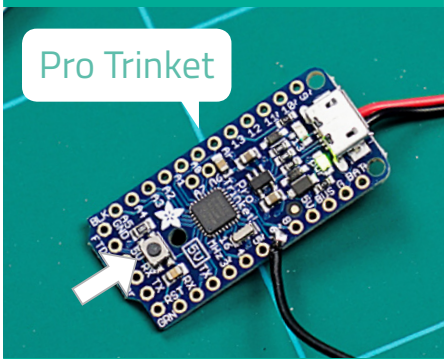
You can also use it as a charger and refill your power source again via USB! No need to waste dozens of throwaway batteries if you want to wear your costume at multiple conventions. Just connect a powerful 6600 mAh lipo and have a reliable and *relatively* environment friendly power supply for days.

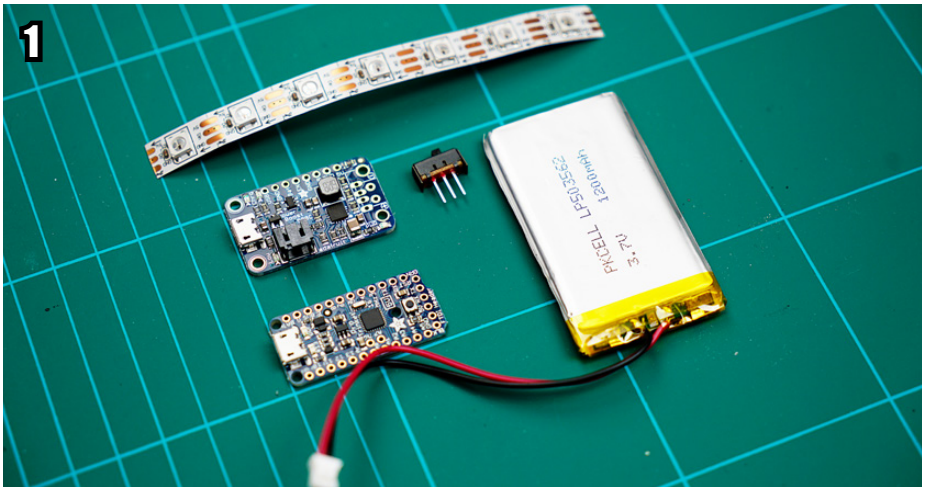
If you want to recharge a battery, there is no need to connect it to your whole circuit. Just plug the battery into a separate Powerboost, connect it via micro USB to a power plug and you're good to go! As long as the tiny LED shines orange, your lipo is charging. When it's full, it turns green.

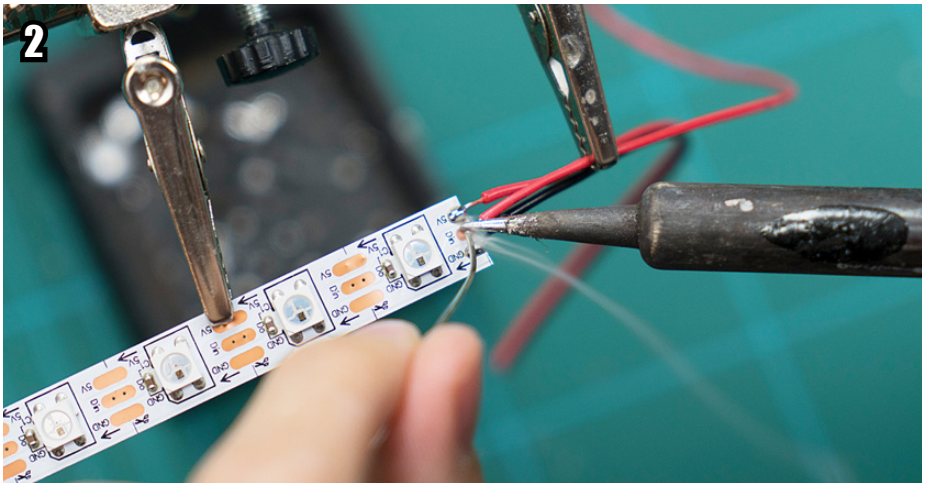
NOTE

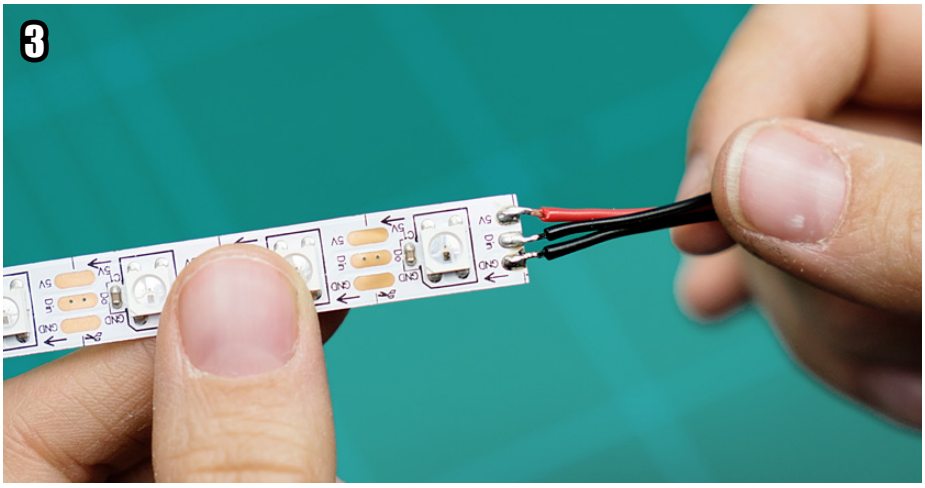
While uploading code to the Arduino Uno goes automatically, you have to press one additional button for the Trinket and Pro Trinket. If you look closely you can see a tiny push button at the lower side of the chip. Press it and an LED starts blinking red. Now you have around 10 seconds to click the button in the software to upload the code. If you miss it, you'll get an error message from the software. Just try again!











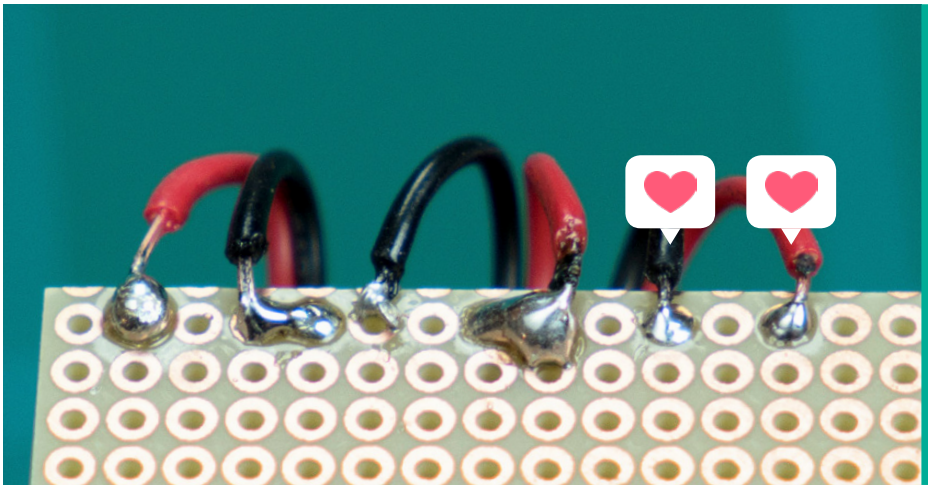
Let's get our hands dirty!

Enough with the theory, let's make something we can use! My favorite circuit consists of a **LED strip**, a **Pro Trinket 5V**, a **Powerboost 500C**, a **LiPo battery** and a **slide switch [1]**. If you managed to solder a LED strip to an Arduino, this will be a cakewalk for you (hmmm cake).

First, attach some colored cables to your LED strip **[2]**. As usual, a handy wire stripper will help you to reach the copper inside fast and without damaging it. It's recommended to have as little shiny metal exposed as possible, so only get rid of a tiny bit. I prefer lead free solder and 0.14mm² color-coded cables in red and black. Solder them to plus (**5V**), minus (**GND**) and one extra for the data (**Din**) pin **[3]**.

Sadly mother nature only gave us two hands which makes soldering things very tricky. Luckily there is this *handy* tool called a **third hand** that you can use to fix LED strip and cables into place. Now you can hold solder and soldering iron easily!

One way to attach the wires is to place a drop of solder on the copper pin, heat it up again and then 'dip' the wire into the liquid solder. I mainly just place the cable directly on the pin and add solder on top of both afterwards. Experiment a bit to see which technique you prefer.



NOTE

A good solder joint is silver, shiny and shaped like a cone. It should not look like a blob or touch any metal outside of the pin. Before soldering for the very first time, practice on a solderable breadboard! It might also happen that your solder turns gray and dull once it cooled down. That's called a cold solder joint and it's pretty unreliable to transport electricity. Reheat it, add some flux or get rid of it with your soldering tip and do it over again.

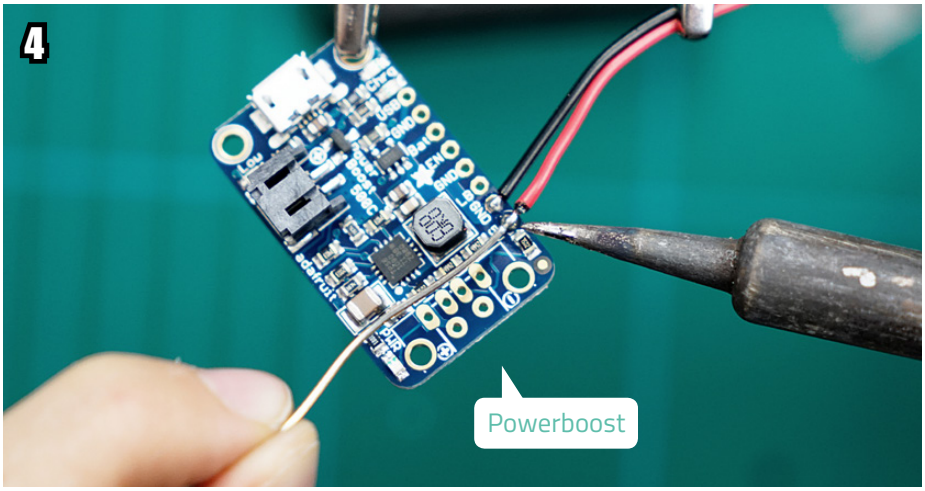
Connecting the chips

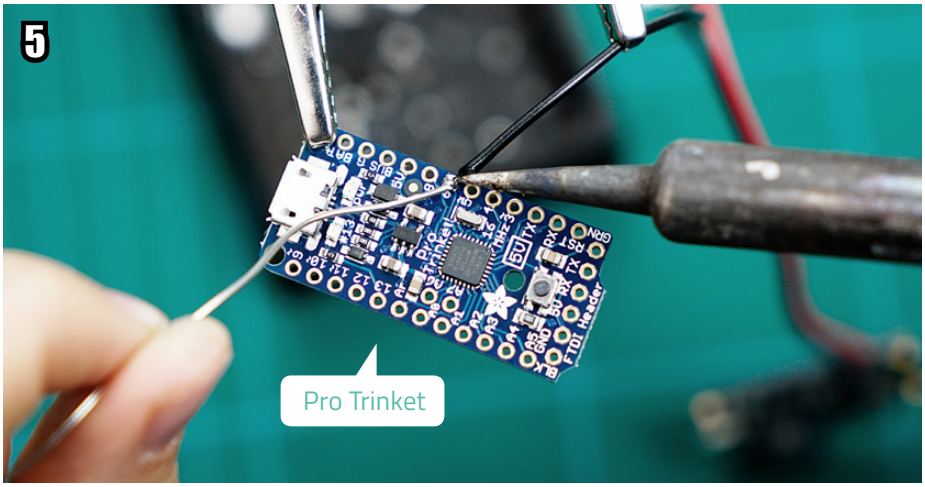
Soldering a cable to a microchip is a little different from soldering it to an LED strip pin. First twist the copper threads of your wire together between your fingers and then stick them into the pin hole of the board. Add a little drop of hot solder on top until it creates a cone shape. Make sure to cut off the wire sticking out on the backside as well. Do this to connect the **GND** and **5V** cables from the LED strip to the **Powerboost** first **[4]**.

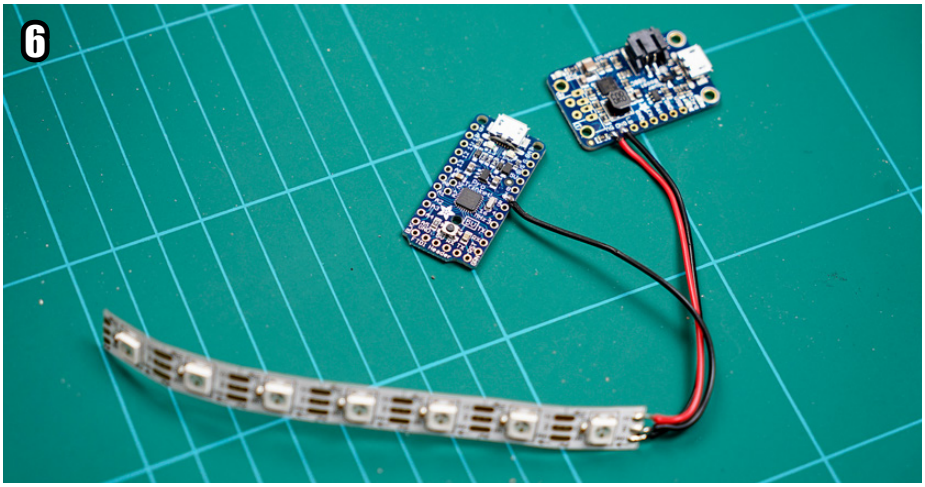
Now the **Din** (data) pin of your LED strip needs to get connected to the **Trinket**. Plenty of codes are using **pin 6** for the default setup, so it's a good idea to choose this one as well **[5]**. If you want to use another number, you can do that too, just make sure to change it in the code afterwards. A pin with a wave (~) over it or next to the number (like pin 6 has) means that it's able to send an analog and a digital signal. It's only important for specific codes so we don't need it right now. I just wanted you to know it exists in case you need it at a later point.

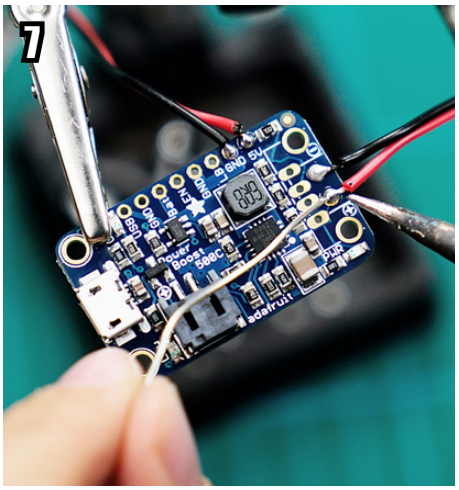
That looks pretty good so far **[6]**! Let's connect the Powerboost to the Trinket next. Strip a short double colored cable at both sides and solder their ends to the middle **(+)** and **(-)** pins at the bottom of the Powerboost **[7]**. You want your two chips to stay close together and save space, so cut the wire short. I usually just tape both of them on top of each other once the circuit is done.

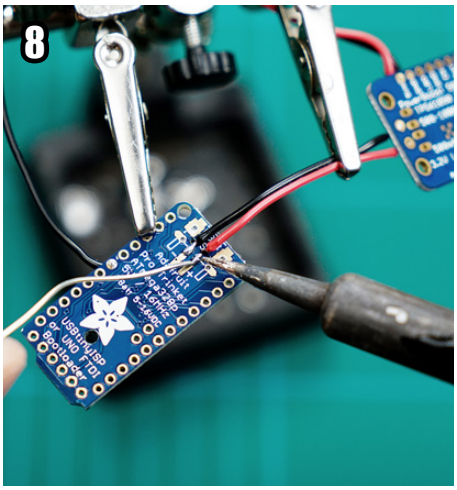
Turn the Trinket around to find the corresponding **(+)** and **(-)** pins for the other ends of the cables. They look like two longer metal bars. Soldering them here requires a pretty steady hand, but I'm sure you can manage it **[8]**! While moving your circuit or Trinket it's possible that the exposed wires on the back of plus and minus accidentally touch. This is called a short circuit and should be avoided. It won't explode or anything dramatic but you run the chance of damaging your chip once you power it up. Add some insulating tape or hot glue to prevent that.

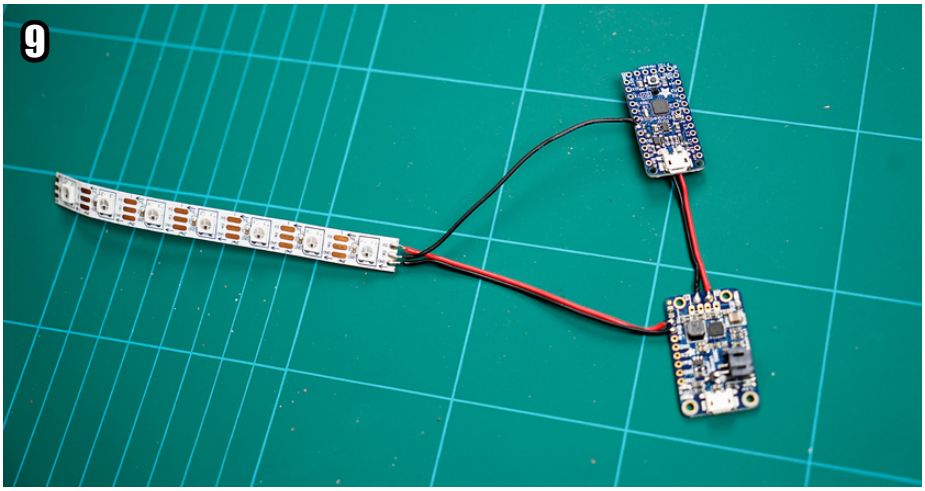


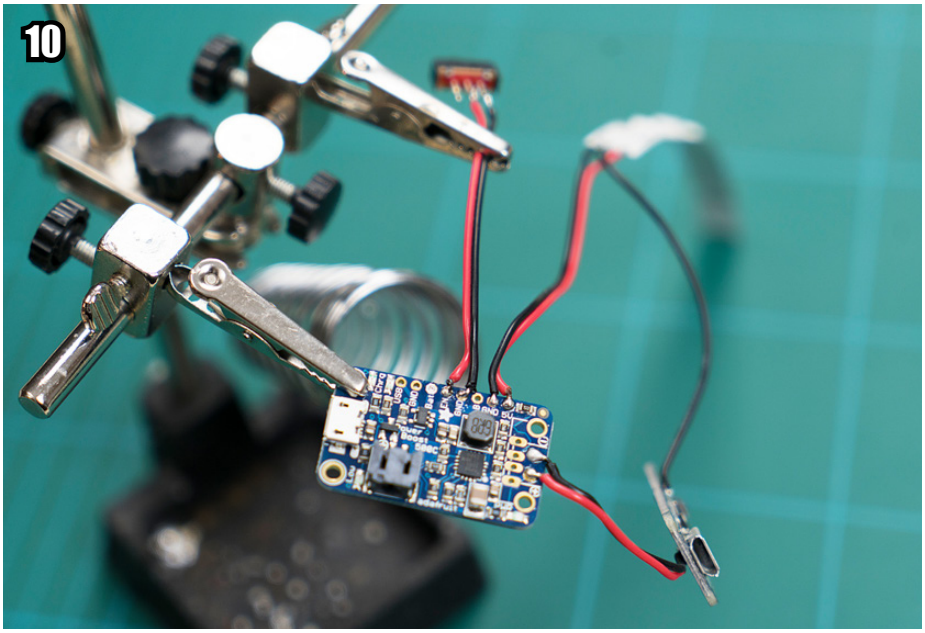


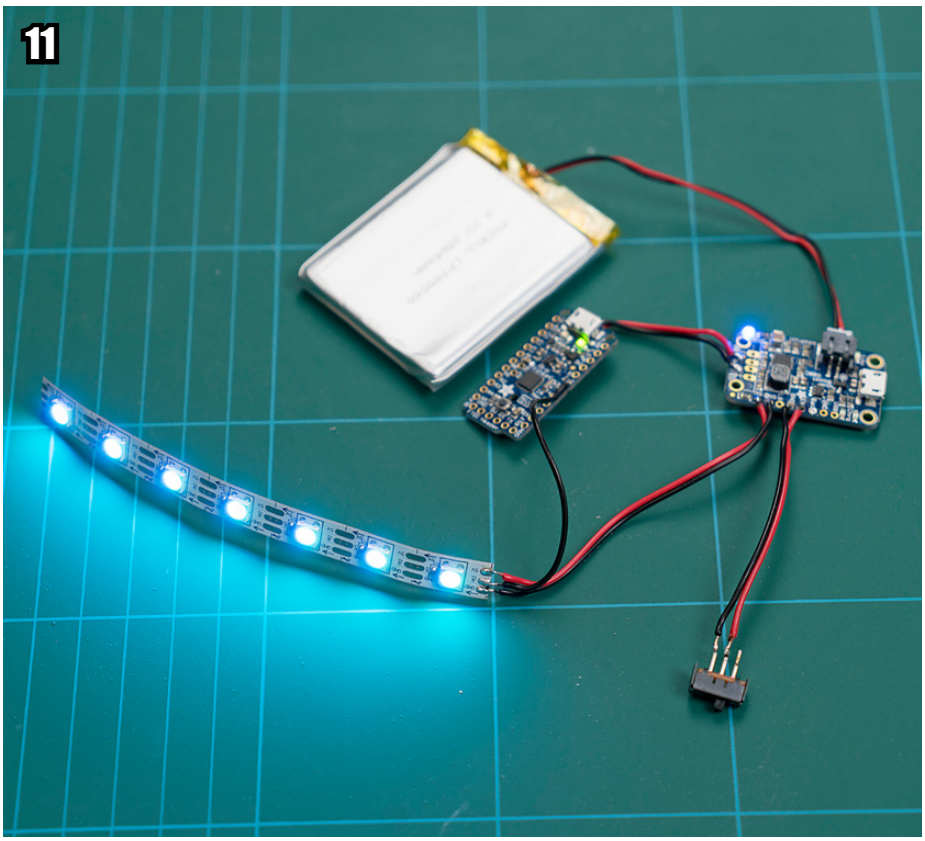












My circuit plan at the beginning of this chapter is just one possible connection **[9]**. Most chips have several pins for data, plus and minus and you can totally use different ones if you're feeling fancy. Don't get confused when you find tutorials with chips connected a totally different way. It might be a good idea to check the data sheet of a specific product to understand the functions of all pins. I never do this. Just saying it *might* be a good idea.

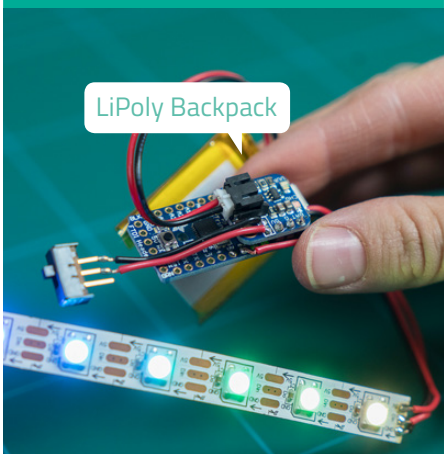
Adding an on/off switch

Last but not least, you'll need a slide switch to turn the whole circuit on and off. Even though the switch has three pins, you just need to connect the middle to the **EN** and one of the sides to the **GND** pin of the Powerboost **[10]**. Then all that's left is to plug a lipo battery (I'm using a 3.7V, 1200mAh) to the Powerboost **[11]** and connect the Trinket via a USB cable with your computer.

In the past, I connected switches to EN, GND and BAT, following older tutorials from Adafruit. While that's totally fine with plenty of slide switches, it actually turned out that a few were causing a short circuit this way. To make sure this doesn't happen, just use only two wires (EN and GND). On the following pages you'll find a few examples with 3 wires connected to them, so now you know why.

NOTE

If you need your circuit to be even smaller, consider using a **LiPoly - Backpack**. It's a much smaller version of the Powerboost and can be stacked on top of the Pro Trinket. It cannot charge batteries and the wiring is a little bit different, but it might be more suitable if you want to save space.
Pro tip: use the backpack just for connecting the battery and a separate Powerboost to charge it.



Fallout 4 - Gauss Rifle

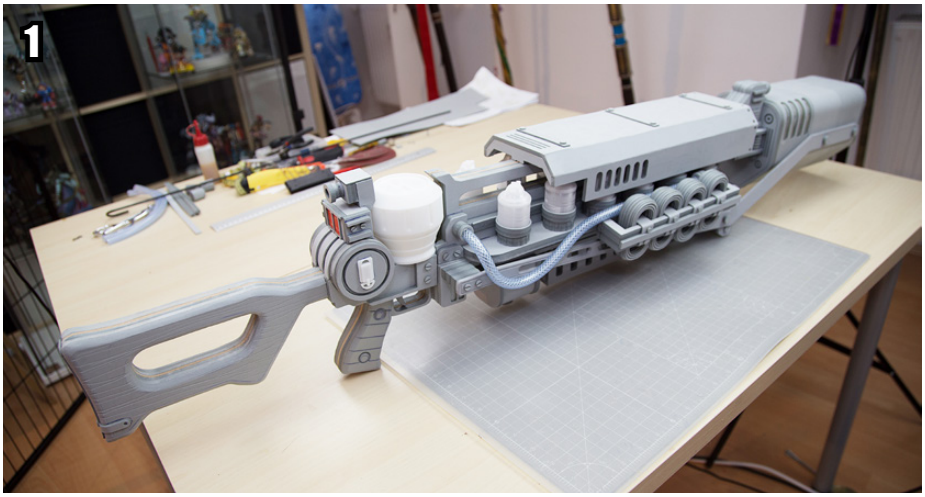
The fancy circuit you just soldered? It's what I used for this little project! As soon as I saw the Gauss Rifle light up for the first time in the game, I was in love. I've never built such a large rifle before but I immediately knew that my life wouldn't be complete without it. Looking back at this prop now, the LED wiring was probably the easiest part of the project!

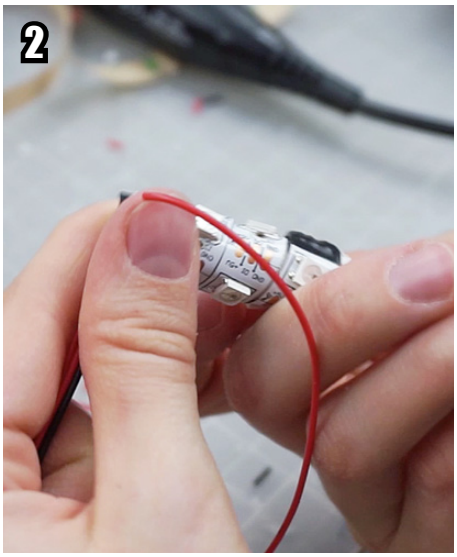
As already mentioned in my book **Advanced Painting**, the Gauss Rifle was completely laser cut out of 2, 5 and 10mm high density EVA foam and glued together piece by piece with contact cement. Despite its massive length of 120 cm (47.2 inch) and its bulky shape, I still wanted to be able to transport it easily. EVA foam was just the perfect material for that and in the end the final result weighed less then 2kg (4.4pounds). In addition I built the whole prop out of 10 individual pieces, all held together by strong neodym magnets. All in all a prop I'm pretty proud of **[1]**!

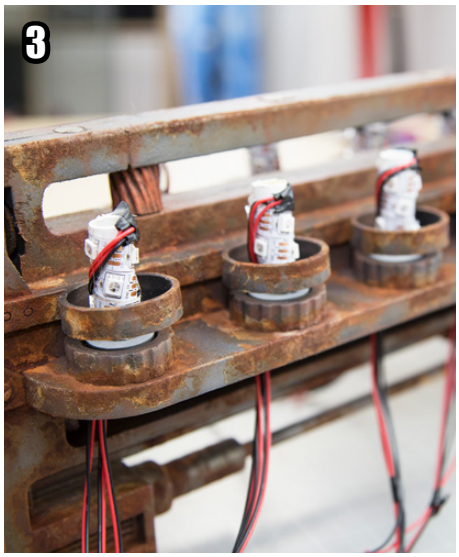
Now let's talk about the light animation. To brighten up the canisters I took a long LED strip and started cutting it into equal sections of 12 pixels each. Once I was done, I soldered six long cables to the beginning and the end of each strip. Additionally I cut a thin PVC pipe into smaller pieces and began wrapping my LED strips around them **[2]**. After I had all 12 of those LED pipes prepared, I simply stuck them onto a round piece of foam and glued them to the side of my rifle using contact cement. I also drilled a hole and let their wires hang out at the bottom **[3]**. Next it was time to cut the cables at the right length **[4]**, strip their ends and solder everything together into one long LED strip. After the left side was done, I repeated it all for the right side too.

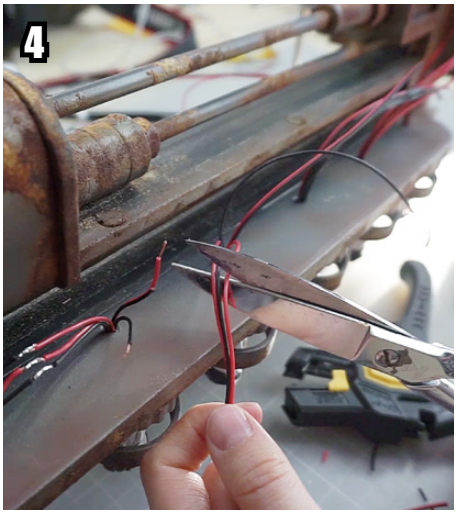
To diffuse the lights, I added thin 3D printed plastic canisters on top **[5]**. When painted with only a thin coat of color they look solid when the light is turned off but shine super bright when it's turned on. It's a really cool effect!

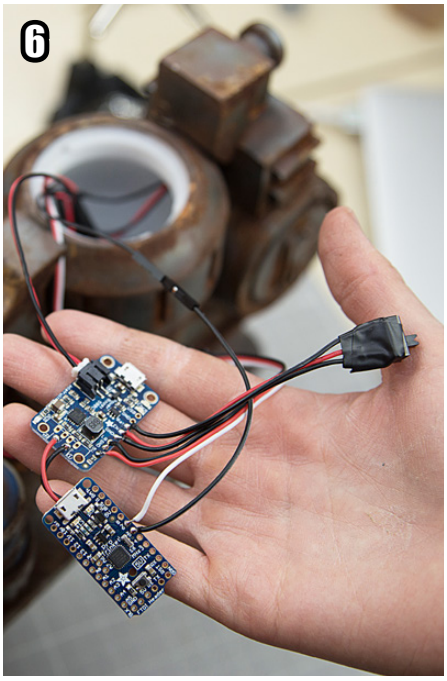
See the lights in action on our YouTube channel: https://youtu.be/vdgSBeRonbw

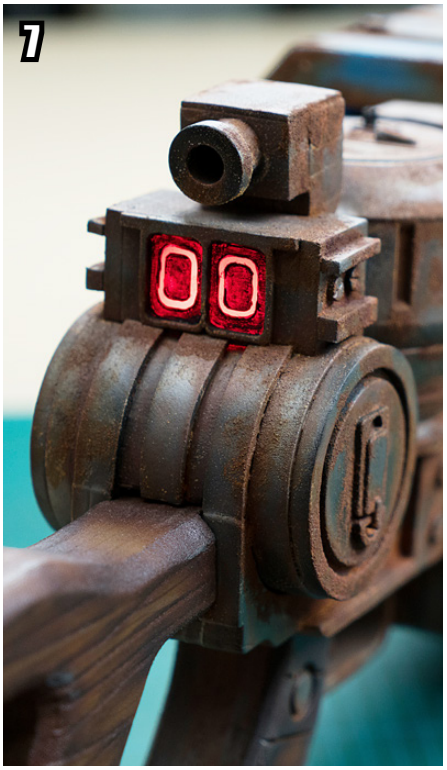












Both LED strips were connected in parallel [see circuit layout below]. Plus and minus went to the Powerboost and the data pin to the Pro Trinket. I also added a slide switch **[6]**. So far, so familiar! However I also wanted to light up the numbers at the rear sight. To do this, I cut another short strip of 4 LEDs, hid it behind a sheet of red acrylic **[7]** and connected its data wire to pin **5** on the Trinket.

As you noticed, I worked with two pins this time! In the code they had to be declared first so I simply copied everything. I dedicated **PIN2** to **pin 5** and told my code that **strip2** consisted of 4 LEDs. Following this, I just had to make sure that all commands related to strip2 need to include the name strip2 as well.

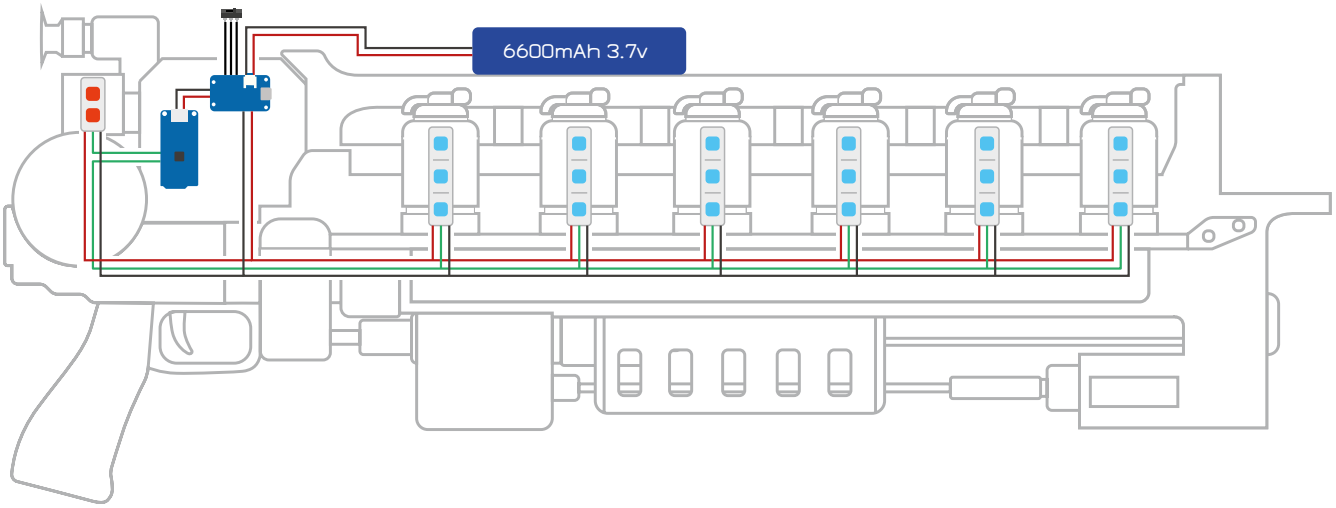
```
#define PIN 6
#define PIN2 5

Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
Adafruit_NeoPixel strip2 = Adafruit_NeoPixel(4, PIN2, NEO_GRB + NEO_KHZ800);
```

So, to get **strip2** working, I had to copy paste the commands **strip2.begin()** and **strip2.show()**.

```
void setup() {
  strip.begin();
  strip.show();
  strip2.begin();  // I added this for strip 2
  strip2.show();  // I added this for strip 2
}
```

I didn't want the numbers to pulse or have any other animation. They also had to be red, while the rest of the rifle was supposed to be blue. My solution was to use the colorWipe animation to let all 4 pixels light up in red. As you can see, I renamed the strip to strip2 to send the command to the right pin. Afterwards I deleted every other program except the rainbowCycle, which I wanted to use for the blue LED strips in the canisters.

```
void loop() {
   colorWipe(strip2.Color(255, 0, 0), 50); // I changed this to 'strip 2'
   rainbowCycle(20); // For the blue canisters
}

void colorWipe(uint32_t c, uint8_t wait) {
   for(uint16_t i=0; i<strip2.numPixels(); i++) {
      strip2.setPixelColor(i, c); // I changed this to 'strip 2'
      strip2.show(); // I changed this to 'strip 2'
   delay(wait);
   }
}
```



Getting the colors just right

The **rainbowCycle** code is great for many different light effects. Instead of using pure blue, which would have been **strip.Color(0,0,255)**, I wanted to have a brighter sky blue, which would be equal to **strip.Color(0,125,255)**. Entering fixed numbers doesn't work for the rainbowCycle though. Instead I had to add a bit of math to the original value of **WheelPos*3** (which as you know is constantly changing). WheelPos*3 here represents the current value of blue, which is anything between 0 and 255. To get sky blue, I added **1/2** of the value of blue to the green channel. It also has to change the whole time but always stay **1/2** of the blue channel. So, if I have **WheelPos*3** for blue, it would be **(WheelPos*3) / 2** for green. Isn't math amazing?

So basically, the command for setting the strip to sky blue was:

```
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(0, (WheelPos * 3) / 2, WheelPos * 3);
  } else if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, (255 - WheelPos * 3) / 2, 255 - WheelPos * 3);
  } else {
    WheelPos -= 170;
    return strip.Color(0, 0, 0); // I set all to 0
  }
}
```

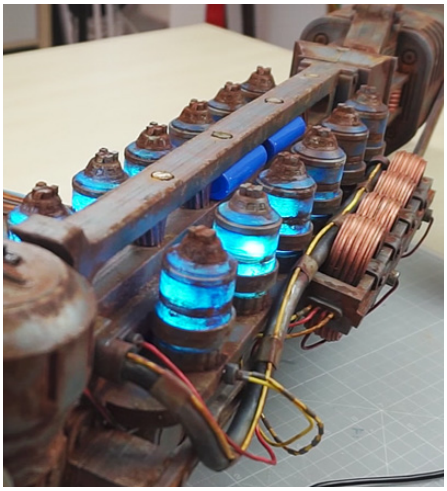
In addition I wanted to change the direction of the animation. The light should start at the stock and then crawl through the canisters to the front. The standard animation code made it run backwards though. I searched online for solutions and played around with the **void rainbowCycle()** command. This was the result. If you compare the code to the original strandtest sketch, you'll notice that I only added the small section marked in **color**. I found this code snippet on the Adafruit website and just copied it in here. Sometimes searching for a similar code on the Internet and copying it into your project to see if it helps is the best way to move forward if you're stuck!

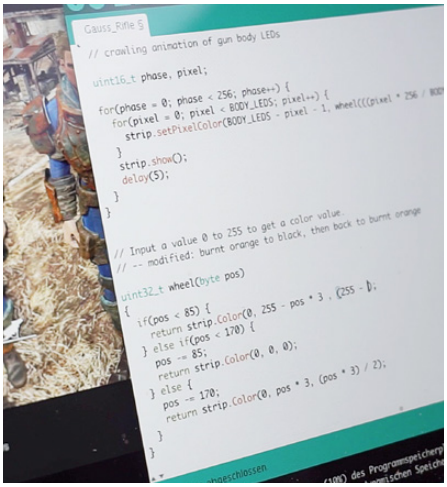
```
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(strip.numPixels() - 1 - i , Wheel(((i * 256 / strip.numPixels())
      + j) & 255));
    }
    strip.show();
    delay(wait);
    }
}
```

The animation of my power cells was now a bright sky blue, which was constantly crawling from the back to the front of the Gauss Rifle, while the back numbers were illuminated in a constant red. Exactly what I wanted! I was now able to hide the whole circuit in the hollow 3D printed canister at the back. Sadly this place was too small for even small lipo batteries.

To solve this I drilled a little hole into the canister, pulled through the plus and minus cables from the Powerboost and soldered a connector to its end. The top area above the lit up power cells would be covered up by a big blue piece later so I placed two slim 4400 mAh lipo batteries into the hollow middle of the body. I was aware that the LED strips would drain a lot of power, so it wasn't a bad idea either to use the remaining space for a spare battery!

I'm super happy how this prop turned out and the animated LED effect really added a whole new level to it. As you saw the circuit I used was really nothing special. I hope it shows that even with a little soldering knowledge you can already build very fancy looking things!





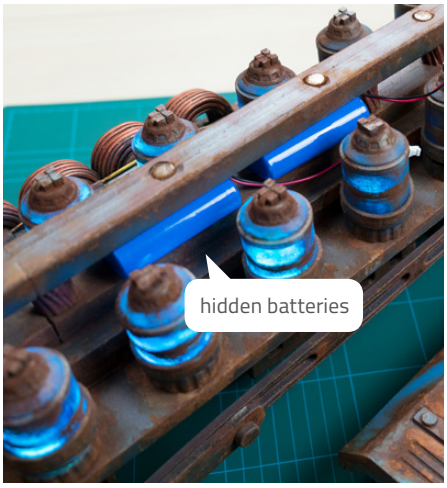
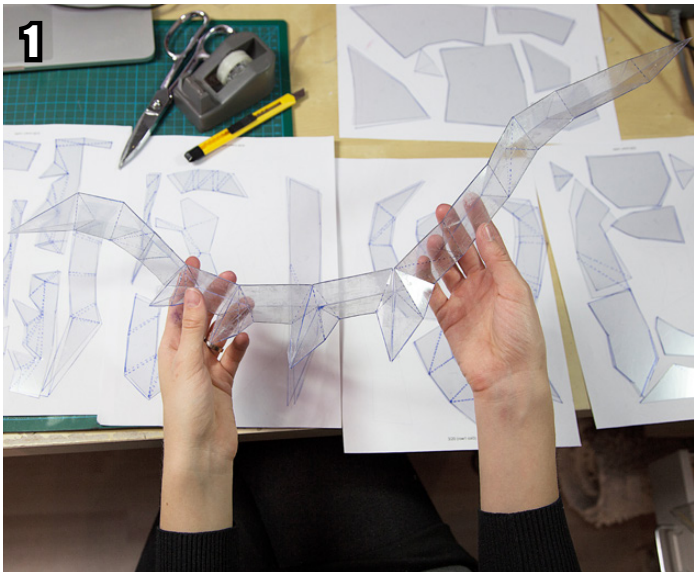
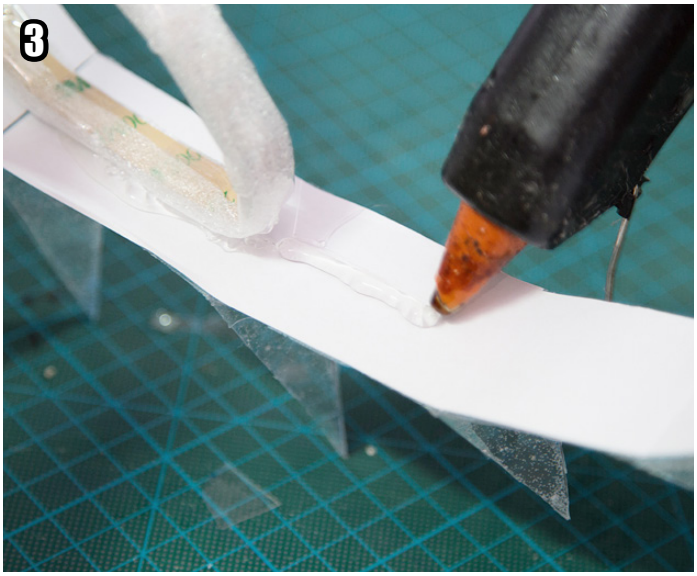


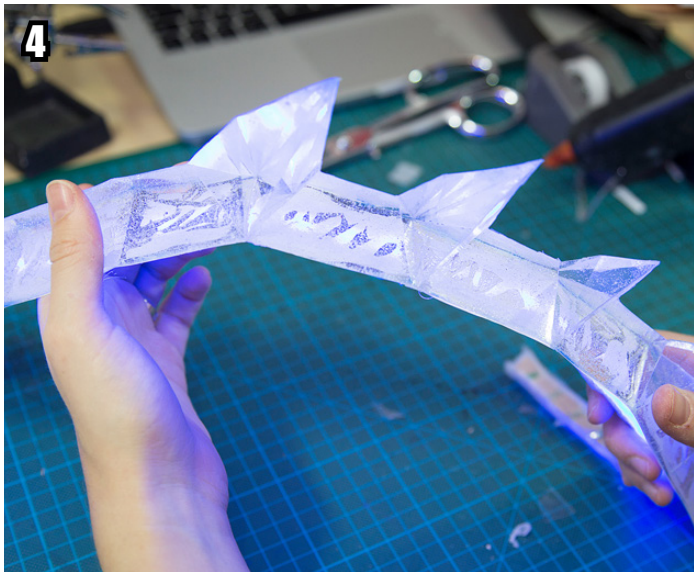

Photo by eosAndy

World of Warcraft - Shadowmourne

Building the legendary axe Shadowmourne was one of my most ambitious projects ever and taught me a lot about the wonderful world of animated LEDs. It houses around 150 lights that I used to bring the deathly shimmering ice effect to life.



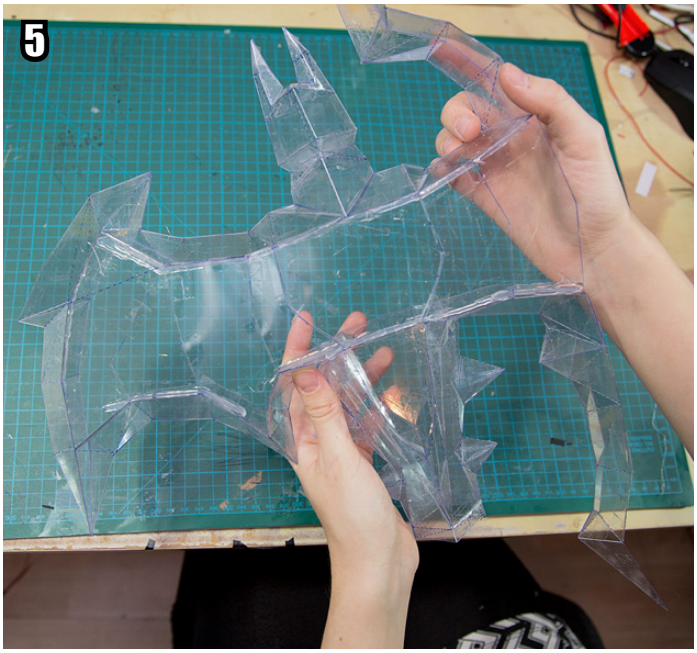


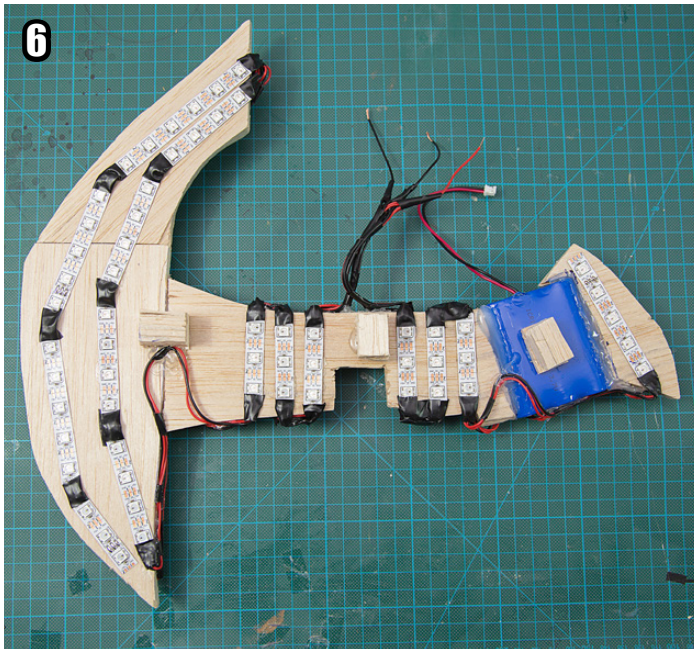


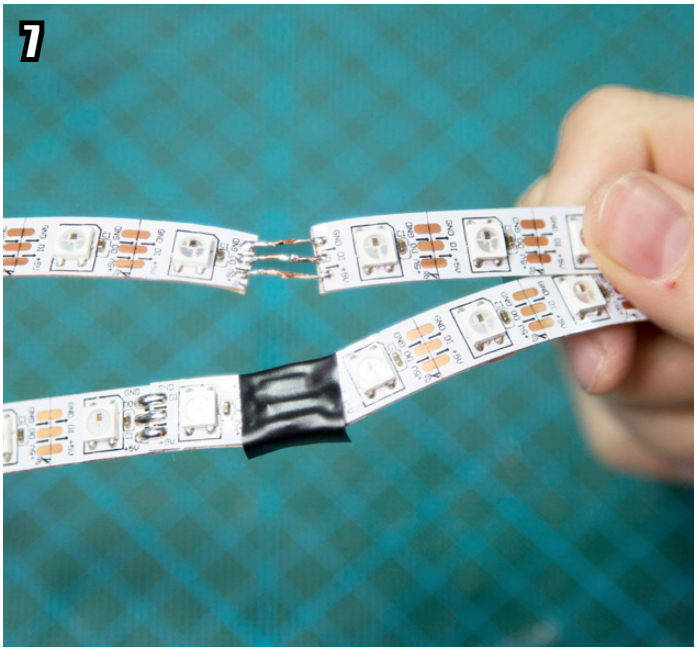
Creating this axe was challenging in every single aspect: Picking the right materials, figuring out the entire building process, setting up the wiring of the electronics and finally adapting the code to get the result that I wanted. My main reference was the original 3D model from World of Warcraft. I imported the model into the papercraft software **Pepakura Designer**, which allowed me to create a hollow life-sized model using only a very thin transparent plastic foil.

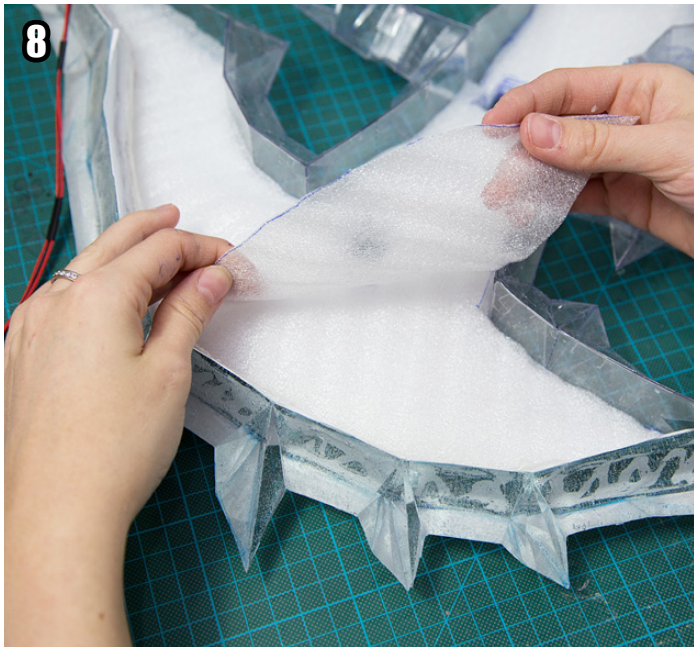
I started with the blade and traced its shape first to paper and then on foil. I cut out every piece and connected them at their edges with transparent tape. I now had a pretty fragile hollow shape **[1]**. Afterwards I mixed some clear resin (Smooth-On Crystal Clear) and carefully filled my blade with it. Once it had turned solid, I ripped away the foil again and got a perfectly clear copy of the blade of my axe (except for a few bubbles) **[2]**.

To light up the blade, I simply stuck some paper to the backside and then hot glued on a LED strip wrapped in packing material **[3]**. This was already enough to diffuse the lights. It wasn't a big deal, but the effect already looked pretty sweet **[4]**.









The rest of the body was done with a similar technique but turned out to be a little bit trickier. I started by cutting out countless plastic foil pieces and carefully connected them with tape **[5]**. Everything was super wobbly and I had to be really careful not to destroy it all with some clumsy gesture. Next, I also taped on the resin blade, which felt far too heavy for the hollow foil built of Shadowmourne's body.

I wanted to light up the runes that were placed all over the axe and the only way to do this, was to keep the body hollow and put a whole lot of LEDs inside.

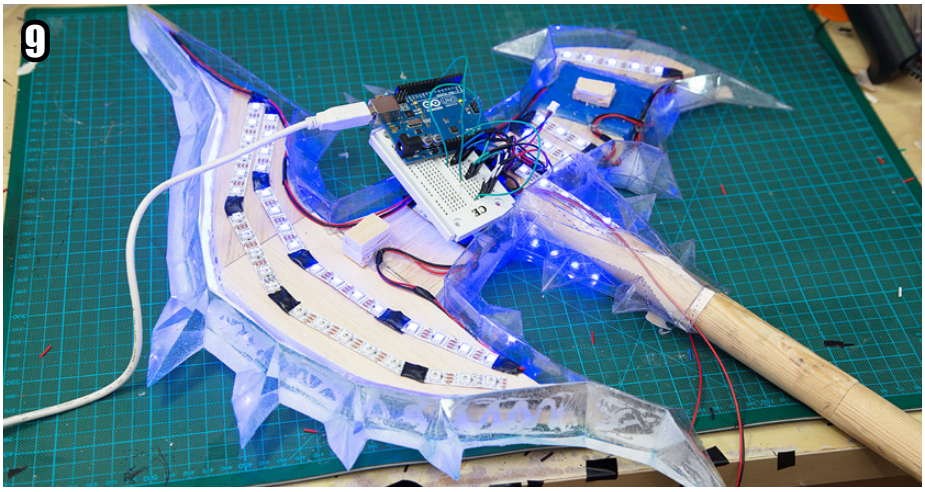
Balsa wood was solid and lightweight enough that I decided to use it as a core. I cut out a shape which would fit into the Pepakura body and hot glued a bunch of LED strips on **[6]**. To be able to bend them over the edge and get them into a curved shape I had to cut the strips into short pieces and solder them together again using short wires **[7]**. In addition to the LED strips, I also hot glued a powerful 6600 mAh lipo battery into the wooden core. With the proper wiring I would be still able to charge my power source, but I knew it would be impossible to ever reach the battery again. After running countless hours and days at conventions, the battery life actually dropped from 6 hours to around 2 hours, however this was still the best solution I found back then.

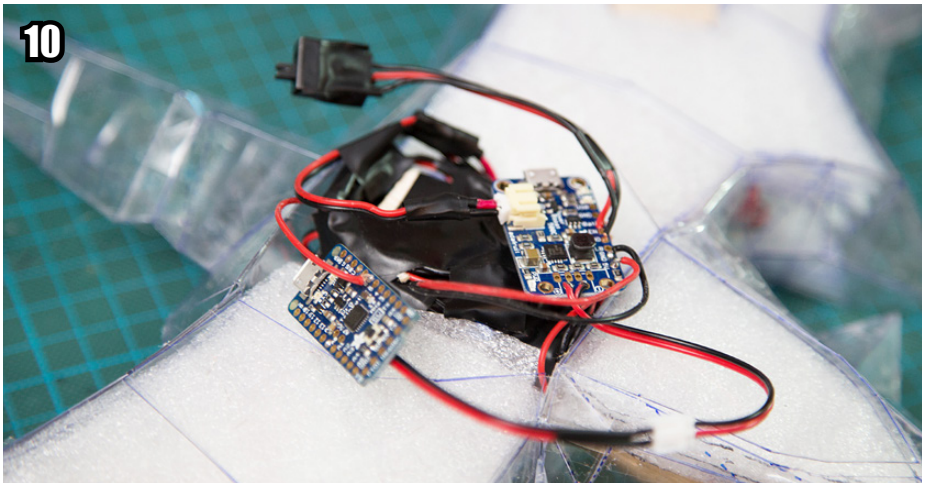
To diffuse the lights, I used translucent packing material. It's also the same bubbly plastic sheet used for footfall sound insulation when laying wooden floors. I found it in a huge 20m (21yard) roll in the hardware store for just a few bucks and it will probably last me for all eternity. I cut out a few layers, placed them into the hollow plastic foil body, added the balsa wood and my LED strips on top and covered everything with more layers of packing material **[8]**.

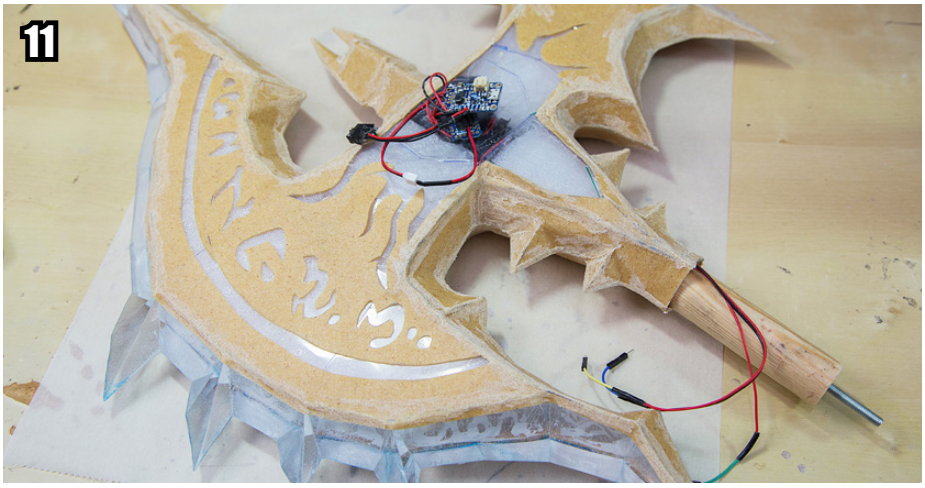
At this point, I finally did a first light test. I soldered a bunch of jumpers to the ends of my wires, plugged them into a breadboard and connected them to my Arduino Uno for a quick test-run **[9]**. I *had* to make sure all my circuits were correct, the LED strips worked properly and the diffusing material was thick enough to create an even light effect. Once the axe was closed, I would have no chance to open it up again and fix things. After I double checked that everything worked, I finally got rid of the Arduino set-up and finished my wiring and circuit with a proper Powerboost, Pro Trinket and a slide switch **[10]**. I wrapped everything in insulation tape, added a bit more packing material and closed the remaining openings in my Pepakura built with more tape and foil.

Now my LED circuit was done but my axe was still a fragile plastic shell, ready to break at any minute. The thermoplastic material Worbla was just the perfect solution for this problem. I took the same patterns I used for the plastic foil before, but this time transferred them to Worbla. After cutting them out with scissors, I carefully heated up part after part with a hot air gun and attached them carefully on the plastic **[11]**. Only the blade and the runes I left free.

After covering the whole axe with more Worbla layers and details, I attached a hand sculpted demon skull to both sides, added a couple of further LEDs to light up their eyes, and built a grip **[12]**.

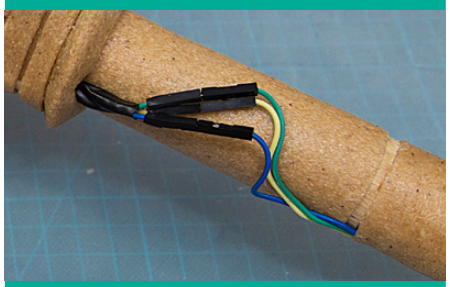


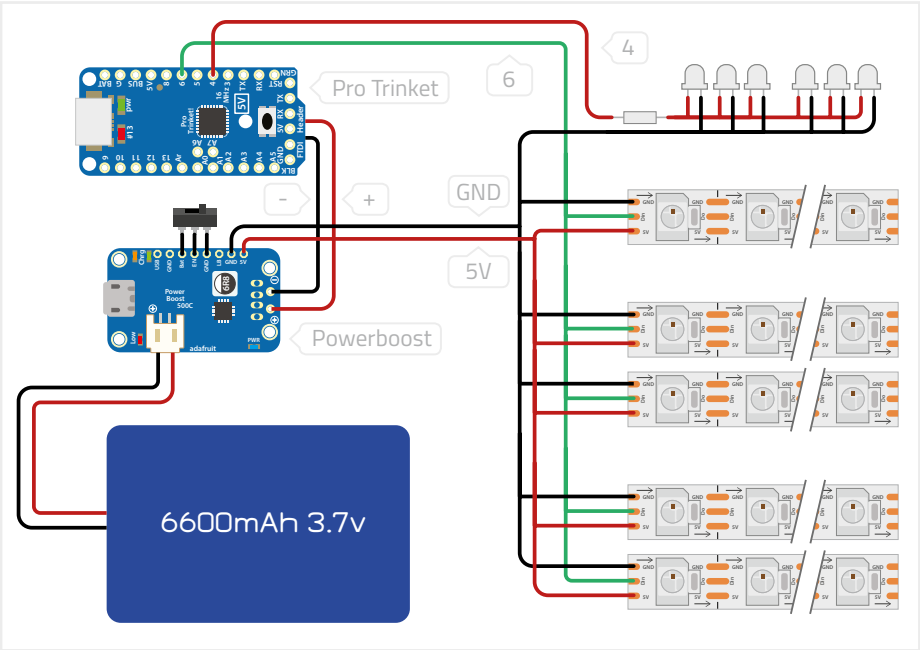




NOTE

For easier transportation I actually made the grip detachable. I cut my wires in half and soldered on plugs to disconnect and reconnect the lower part of my LED strip circuit with the upper one.

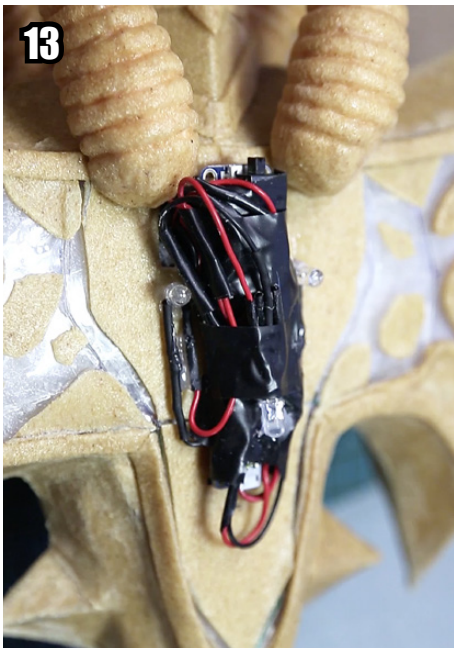




That's a lot of lights!

At this point you're surely curious, how the circuit and the code looked like, right? Just take a look at the layout on the left!

I wanted to keep it everything as straight forward as as possible, so I worked a lot with parallel connections. The LED strips on the wooden core were actually four separate strands, one to the left and one on the right side of the skull, both front and back. Then there was a fifth strip glued to the back of the blade. All of those were soldered together in parallel and then connected to a single data pin (#6). Additionally I also connected six single LEDs in parallel to pin 4. Three for the skull in the front and three in the back. Since the pin provides 5V of power output but the LEDs only need around 3V, I lowered them with a fitting resistor.



The only place to reach the switch and plug in a battery charging cable was inside the hollow front skull **[13-14]**. While the back one was permanently glued to the body, the front was mounted with strong magnets. This way I was able to reach the circuit easily, turn the lights on and off and recharge the lipo battery inside at any time.

NOTE

The bottom part of my axe was done in a similar fashion to the rest of the prop. I first created a hollow shell out of plastic foil and filled it with an LED strip and packing material for diffusion. The following Worbla skin then made it durable and blocked the light except for the few areas I left open. I carved a thin trench into the wood of my handle to hide the cable running back to the top of the axe. Later I would also glue on fake fur on top so it would no longer be visible.

To light up the LED eyes without any animation, I checked the sketch **Examples > Digital > Button**. It's a program that lets an LED light up at the push of a button. By taking a look to the code and uploading it, I understood that **const int ledPin = 13** defined the number of the pin used for the LED(s), **pinMode(ledPin, OUTPUT);** set the pin as an output (so it will send a signal to the LED) and **digitalWrite(ledPin, HIGH);** then turned the lights on.

I simply had to copy these three lines to my strandtest sketch to turn the LEDs for my skulls on. The necessary commands had to be placed in void setup() and void loop(), so I added them and it worked.

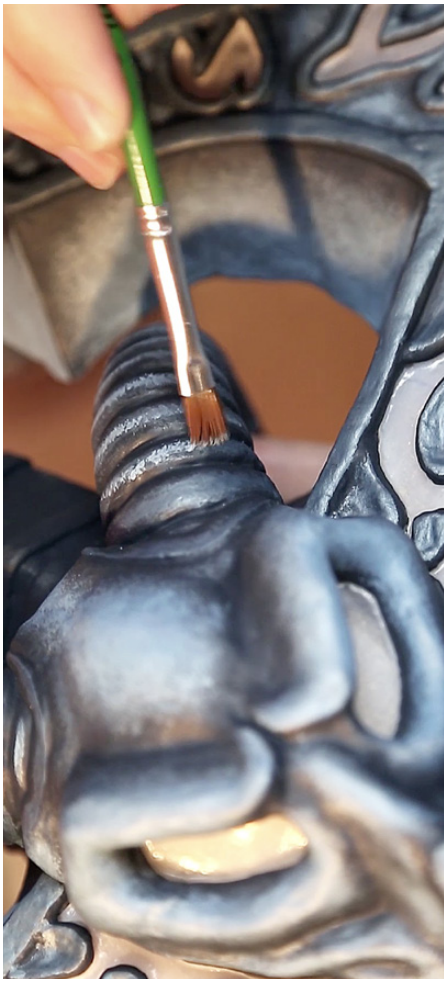
```
#define PIN 6
const int ledPin = 4; //declares that my LEDs are connected to pin 4

Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
    strip.begin();
    strip.show();
    pinMode(ledPin, OUTPUT); // tells pin #4 give an output
}

void loop() {
rainbowCycle(20);
digitalWrite(ledPin, HIGH); // turns the LEDs on
}
```

The code for the LED strips was again the basic strandtest **rainbowCylce** animation in sky blue. Pretty much the same I already used for the Gauss Rifle!

```
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
      return strip.Color(0, (WheelPos * 3) / 2, WheelPos * 3); // I changed color
  } else if(WheelPos < 170) {
      WheelPos -= 85;
 return strip.Color(0, (255 - WheelPos * 3) / 2, 255 - WheelPos * 3); // changed color
  } else {
      WheelPos -= 170;
      return strip.Color(0, 0, 0); // I set all to 0
  }
}
```

Finally, once everything was built, wired and programmed, my husband Benni painted the axe using brushes and acrylics. To give the runes and the eyes of the skulls even more of an ice effect, he experimented and then painted them with translucent sparkly blue nail polish. The result was a pretty convincing looking prop, which I still love to bring to conventions as an example for animated light effects.

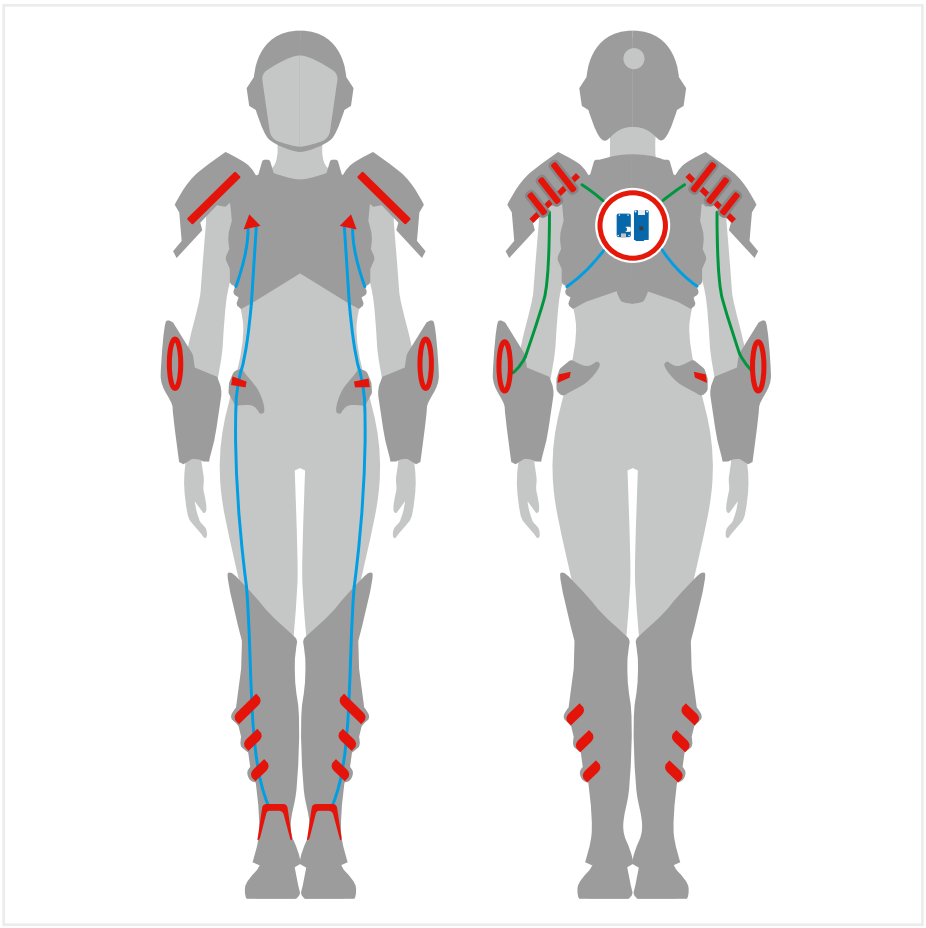
This was definitely one of my more complicated builds. Not because of the LED circuits or the code, but rather because of the entire construction.



Check out the entire making-of video of Shadowmourne on our YouTube channel:
https://youtu.be/McB0fSMyBWs

Heroes of the Storm - Nova

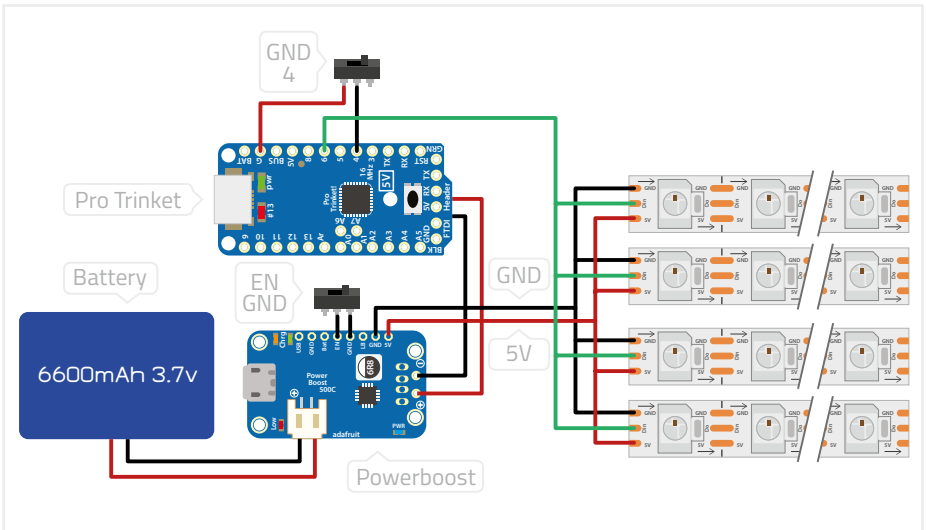
Up until this point I only showed you how to integrate LEDs into your props. Now it's time to take a look at a full costume! To light up Nova's Elite Agent skin from Heroes of the Storm, I wired up more than 800 LEDs! Aside from just building her EVA foam armor, I had to install a single circuit that was running through the entire costume, create a LED ponytail and make it comfortable and transportable at the same time.



Everything is connected

Planning this project was no easy task. Almost every piece of Nova's EVA foam armor had to be lit up individually. From the very beginning, I wanted to utilize fully animated LED strips instead of working with simple static LEDs. This meant I had to control and power up every strip from one single, central source. Adding a separate microcontroller to every costume piece would have been too much trouble.

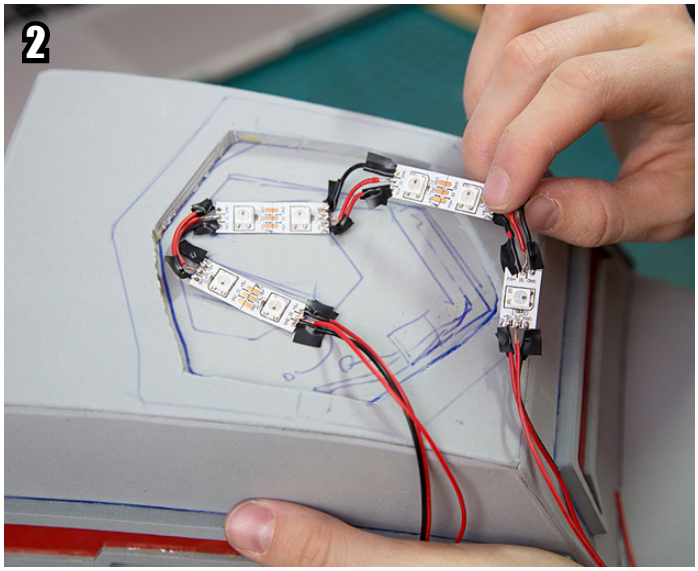
To the left you can see a plan I made to show my set-up, but I think it still needs some further explanations. The 'heart' of my circuit was the circular backpack. Here I hid my Pro Trinket, the Powerboost and my LiPo battery. Connected by a lot of plugs, all the armor parts were then lit up by four separate LED strips that I soldered in parallel. Two strips (left and right) were running from the backpack over my shoulder armor to the bracers (green), and two more went from the backpack to the front of my breastplate, then to the hip pieces and finally ended in the shin armor (blue). Aside from adding one switch to turn my lights on and off, I also added a second one to change the animations. You might imagine what effort it took me just to plan and then wire this monster.

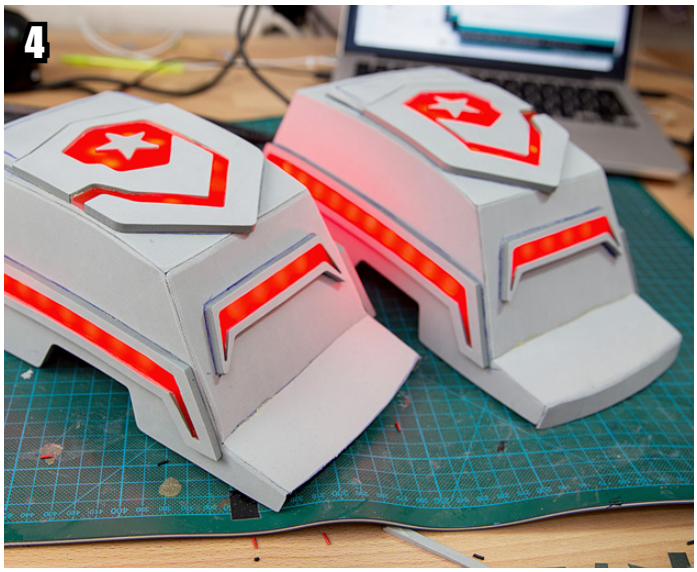


NOTE

By now you probably don't need that many explanations anymore regarding the circuit. One thing is new however:

At the top you see a second slide switch used to control the different animation modes. Using two wires, I connected the middle leg to the data pin #4 and one of the outer legs to GND pin next to the USB connection. More about that in a bit!





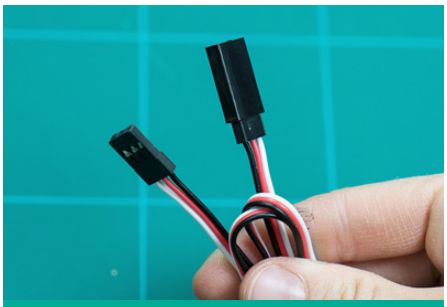
Instead of gluing the LEDs directly on top of my EVA foam, I had to create lower areas by adding more layers of foam on top **[1]**, or by cutting out and then gluing some material back on at a lower point **[2]**. This way I was able to place the LEDs *deeper* in my construction and therefore had additional room to diffuse the lights. Without this, the 'dots' of my strips would have been a lot more visible (more about diffusing in a bit).

To place the shortened LED strips I first soldered wires to their pins and then burned little holes with my soldering iron to pull them through **[3]**. A bit of contact cement glue was enough to stick them on.

On top of the LED strips came a few layers of packing material, a laser cut piece of milky acrylic and a final frame out of EVA foam around. I could still see the individual spots of light through this diffusion, but the effect still looked good and I was happy with the result **[4]**. Getting a nice and even light without ugly spots is one of the harder challenges when working with LED strips.

Next, I soldered together all the strip pieces on the inside of the armor. It was a little bit tricky to reach them, so I had to leave the wires long enough and glue them to the inside of my armor later.

Finally, I cut some servo extension cables in the middle and soldered them to the beginning and the end of the LED strip in each armor piece. To make sure that the animations run into the right direction and all mirrored parts were soldered correctly, I made a detailed drawing of the wiring and checked it 10 times before I glued a LED strip on.



NOTE

To connect the circuits hidden inside my armor pieces I used servo extension cables with three pins. They have a male and a female plug (sexy), are small enough and wired in three different colors. Instead of using them for a motor however, they were actually perfect to connect my LED strips with each other. I kept them nearly invisible by simply sewing them to the inside of my bodysuit.

Changing the light mode with a switch

The code this time was a mix of strandtest and Button, which I already used for my Shadowmourne. Button is a sketch that lets a LED light up when you push? You guessed it! A button. When you press it, the LED glows, otherwise it's turned off. Instead of a push button, I wanted to use a **slide switch** though. And by combining both codes, I wanted to be able to switch between two different modes: One would let all LEDs light up evenly in orange. This way my photographer could easily get good pictures because my lights were always on. The other mode should activate a rainbowCylcle in orange, which would be a cool effect for wearing the costume at a convention. Using a regular push button here would only activate the code as long as I would keep my finger on it.

As always, I first had to define my LED strip and the amount of LEDs I used. To light up all LEDs, I simply entered the longest strand in my circuit (60). This was the one that went over my breastplate and all the way down to my legs. In addition I also had to define the pin for the slide switch.

```
#define PIN 6
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);

const int buttonPin = 4; // I chose pin 4 for the switch
int buttonState = 0; // I copied this as well as it's required for the switch to work
```

I kept looking for commands that included the word '**button**'. **pinMode(buttonPin, INPUT)** was written down in the void setup() area of the original button code, so I simply copied and pasted it to the same position in my strandtest sketch. This line tells the program that the slide switch sends an input to the chip, which is either it's set to the *right* or to the *left*. You'll maybe notice that I changed INPUT to INPUT_PULLUP. Button is a sketch created for the Arduino Uno, but the Trinket works a little bit different. To get it working, you need to make this change. Took me some Google to find this out.

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

    pinMode(buttonPin, INPUT_PULLUP); // copied to strandtest and added _PULLUP
}
```

The second piece was **buttonState = digitalRead(buttonPin)**, which I also copied and pasted from the Button sketch. It tells the program to read the button and see how it's set. Is the switch set to **right (HIGH)**, the program is supposed to use the colorWipe command and fill each strip with the color orange. Otherwise **(else)** the strip will be animated with the rainbowCycle code.

```
void loop() {
  buttonState = digitalRead(buttonPin); //read how the switch is set

  if (buttonState == HIGH) {  // If switch is right
    colorWipe(strip.Color(255, 51, 0), 0); // set to orange
  } else { // If switch is left
    rainbowCycle(20);
  }
}
```

As always, I just had to change some of the void codes to run the animations like I wanted. For the colorWipe mode, I set the very last number to 0, so it instantly filled all pixels with my color instead of turning them on one by one. And finally I also adjusted the colors of the rainbowCycle to get a pretty orange. For this I simply turned the blue channel off and added 1/5 of red to the green channel.

As you see, adding a switch to change light modes is pretty easy! Use it to give your costumes a disco mode for parties or even to turn your peaceful prop into a deadly prop by changing the light from blue to a red.

Add a 'pulsing' animation

I also thought about adding a pulsing effect to the whole armor. This would slowly turn the LEDs on until they are fully lit and then dim them down again afterwards. This effect is also called breathing, since it looks like the LEDs are inhaling and exhaling.

To achieve this, I combined three codes this time. Neopixel's strandtest, Button and the new one, **Fade**. You can find it under **Examples > Basic > Fade**. It's a pretty basic sketch, which lets a single LED evenly light up and dim down. Instead of using the program for a single light though, I changed it to control a whole array of LED strips.

The main code stayed pretty much as it was. I just had to exchange the colorWipe animation with the new fade animation. To do this, I first copied and pasted the declaration for **brightness** and **fadeamount**, which were a part of the fade sketch. By playing around with the code, I noticed that the speed of the breathing depended on the amount of LEDs connected. The more there were, the slower the animation. To speed it up again, I had to increase the value for fadeAmount.

```
int brightness = 0; // defines how bright the LEDs will become
int fadeAmount = 5; // The higher this number the faster it breathes
```

While I kept most of the program similar, I added the command **strip.setBrightness(brightness)**, which I found in the Neopixel Überguide on the Adafruit website. Using these few additional lines, the brightness of the strip now depends on the value of the variable 'brightness', which is set by the function following right afterwards. This let's the animation *pulse*.

In addition I got rid of all delay() commands and set the last number in colorWipe to 0 (which was also a delay command). This gave me full control over the animation speed by simply changing the fadeAmount number in the beginning.

```
void loop() {

   buttonState = digitalRead(buttonPin);

   if (buttonState == HIGH) {

   strip.setBrightness(brightness);  // I added this. Found in the Überguide
   colorWipe(strip.Color(255, 0, 0), 0);  // I set the last number to 0

   brightness = brightness + fadeAmount;

   if (brightness <= 0 || brightness >= 255) {
      fadeAmount = -fadeAmount;
      }
     //delay(0);  // I commented this part out by adding two slashes in front

   }
   else {
     rainbowCycle(20);
    }
}
```

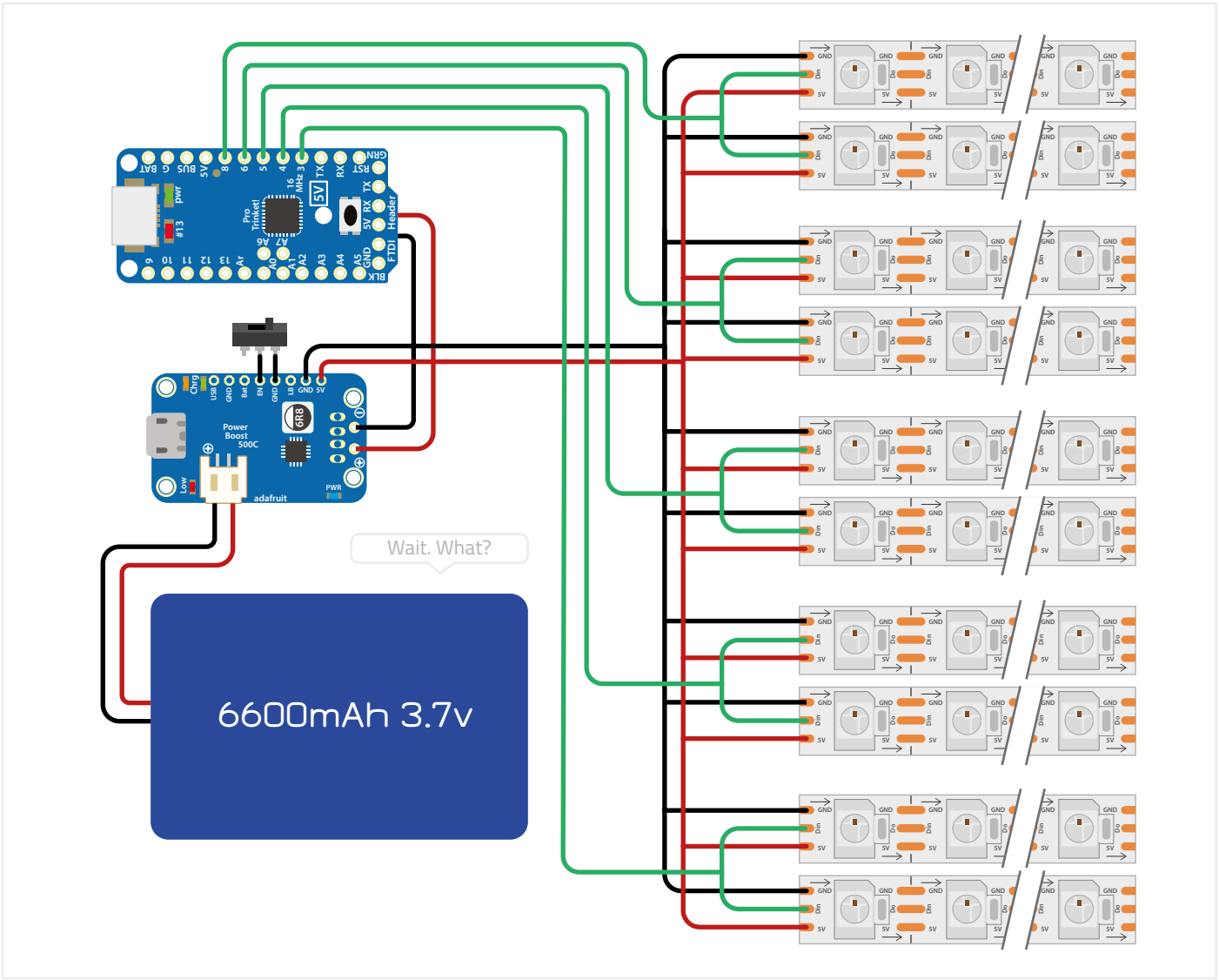
Other than that I just included the standard void (loop) programs for colorWipe and rainbowCycle.

NOTE

This was a pretty advanced project and I had my fair share of issues getting the circuit and code to work. The Arduino works a little bit different than the Trinket and it's impossible to know everything. To get help, I made a post in the Adafruit support forum. It didn't even take an hour before I got a reply. The moderator answered my questions and I finally found the mistake! So if you are ever stuck, don't hesitate to ask for help!

Nova's animated ponytail

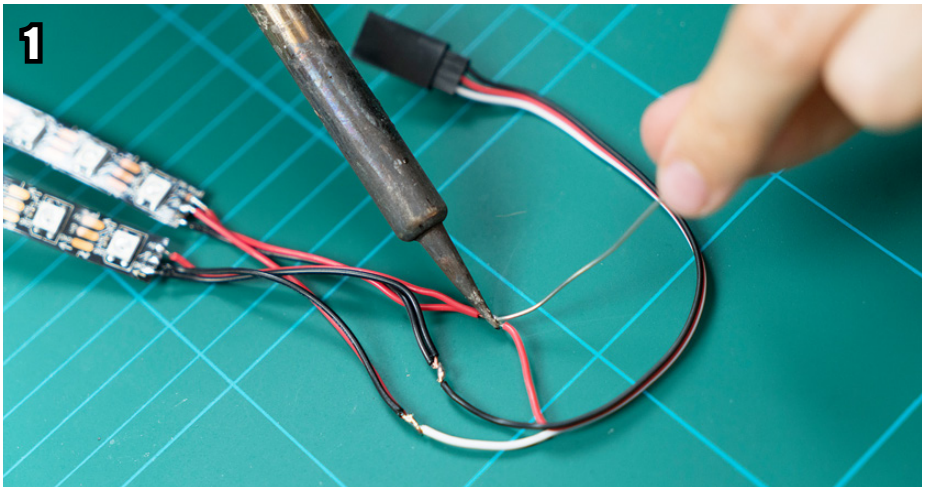
Sometimes working with animated LEDs is less about writing code or finding the right circuit. Sometimes it's just to copy and paste something way above your skill level and then try to make it work. I knew illuminating Nova's light up hair was beyond what I usually do, so I decided to look for inspiration. As always I found something that sparked my interest on the Adafruit website. Okay, let's do this! Leeroooooooyyyyy...

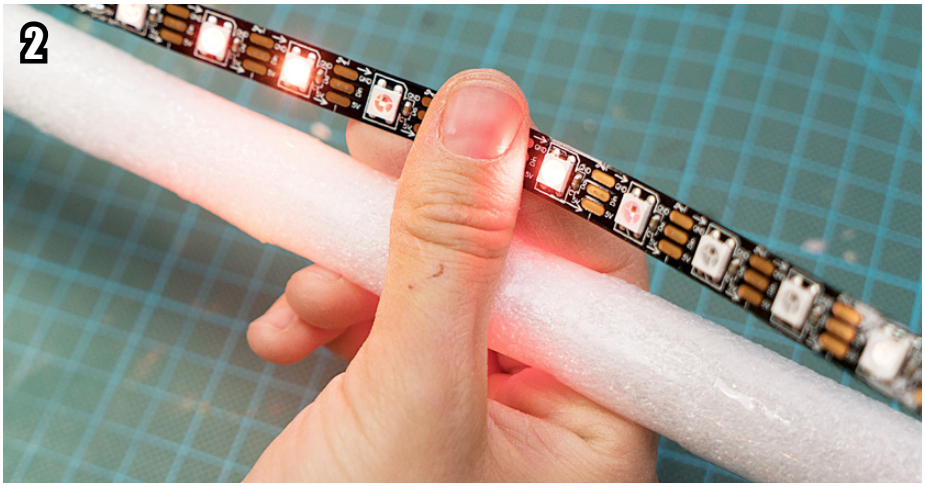


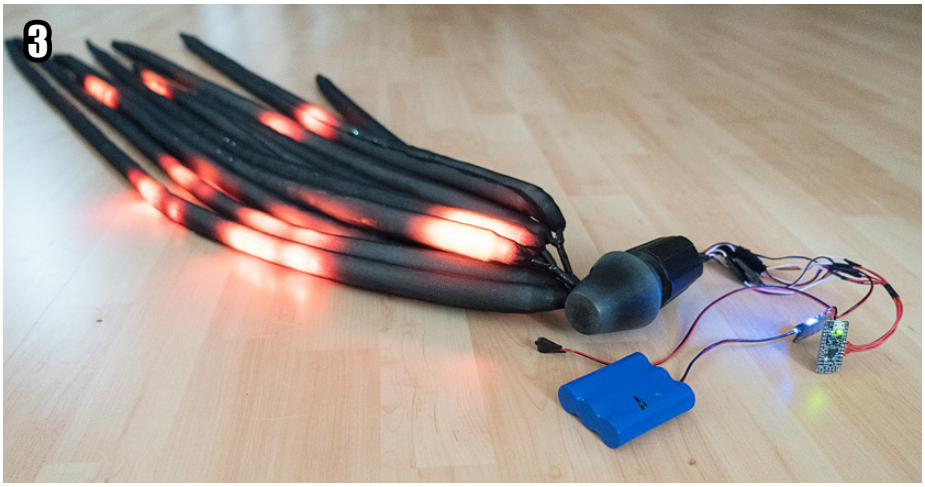
I had a rough image of how I wanted to animate Nova's LED ponytail. Her *'hair'* should consist out of several strands, which would be LED strips covered in diffusing material. Instead of letting them pulse, float or light up evenly, I wanted a random crawling effect Just like raindrops. By scrolling through Adafruit's **Learn > Wearables** page, I found a light up party wig that looked exactly like what I needed. I loved the effect and knew that I would probably just need to adjust the color, speed, amount and length of the 'drops'. Adafruit's wig was reaching only the shoulders, while Nova's hair would go all the way down to my legs.

So, first the circuit. It looks a little bit more complicated than usual but if you look closely, it's almost *exactly* the same as I used for my Shadowmourne. I used 10 LED strips in total and two of them were always connected in parallel.

While all LED strips were connected to the same 5V and GND on the Powerboost, each 'pair' got it's own data pin on the Pro Trinket. As I mentioned, plenty of other projects would work with the smaller regular Trinkets. But especially for this costume I had to use the Pro version since it has enough pins and memory.







It's a wrap!

When working on such a complicated circuit, it's important to keep track of what goes where. I directly soldered my parallel connected LED strip pairs to a plug with cables in three colors **[1]**. This helped me immensely to know which color had to go where when I connected them again later.

To give my skinny LED dreadlocks some volume, I first wrapped them in three thick layers of packing material **[2]**. This also helped diffuse the lights at the same time. Since my hair also had to look cool when the lights were turned off, I covered all of my hair sausages in a skin I sewed out of stretchy black fabric **[3]**.

In total I soldered 10m (33feet) of Neopixel strip with around 600 individual LEDs, just for the hair. To complete this 'hairy' behemoth, all I needed was a powerful 6600 mAh lipo battery and a switch to turn it on and off.

I knew that it wouldn't be a good idea to light up all strips with the basic strandtest code. Especially turning all LEDs on with the colorWipe program would draw far too much power at the same time for the battery to handle. Instead, I copied the power saving code from the Neopixel Cyber Falls Wig example on Adafruit's website. This light drop animation draws a lot less power, since only a small portion of all LEDs is lit up at the same time. After uploading it to my Pro Trinket, the effect worked and looked amazing. I was aware however that I still needed to make a few changes and that, if I ever wanted to wear the costume at a convention, I would probably need a few spare lipo batteries.

NOTE

Once the ponytail was done, it was mounted into a 3D printed helmet. I thought about using Worbla or EVA foam for this, but the material would not have been strong enough. My 3D print however was perfect to carry the weight of the ponytail properly, hide the whole circuit and two big 6600 mAH batteries inside. In addition I made it detachable so transportation wouldn't be a problem either.

The code for this example is pretty massive, so I won't even show it here. It would be beside my point anyway. I honestly have a very hard time understanding how it works!

What I am trying to say is: I believe diving head-on into a project that you don't understand is a great way to grow and get better. Even if you don't know how the program works, your basic knowledge should already be enough to spot the sections for any necessary adjustments. Code snippets from Adafruit always include helpful comments, and just by reading them, changing the code and uploading it again, you can learn a lot.

I'm actually still searching the right command to change the length of the drops, but I'm sure I'll get there soon. You see, just like you I'm not a programmer. I simply keep on trying different things, am patient and learn slowly (sometimes super slowly).

Check out Nova's Rifle on our YouTube channel:
https://youtu.be/oz8LUA2YtaA

NOTE

Congratulations for reaching this point in the book! I'm aware that soldering circuits, programming microchips and animating LED strips is anything but easy. I hope I'm still able to give you a good starting point for your very first project and made you eager and excited to become a crazy blinking Christmas tree!

The most important step is not to be afraid and try everything out, even if it looks complicated! Just like I did with Nova's ponytail!

NOTE

While the animation of Nova's ponytail looks super cool in videos, it's not very suitable for photos. To solve this, I decided to turn all 600 LEDs in my ponytail on at the same time. The maximum brightness my battery could handle was strip.Color(100,10,0). Nothing higher worked, but with this I got at least 5 minutes of time to take a few pics before the massive 6600mAh battery was completely empty again. Still, it looked pretty cool!

What to do when nothing works

Finding a mistake often takes more than just checking for error messages in your code. Sometimes everything spontaneously shuts down without a reason, even after it worked just a moment ago. It can be really frustrating. Before you rip out your hair though, let me tell you a few helpful stories from when I was building Nova.

Nova's cursed LED strip

I had one of the strangest issues with Nova's ponytail. When I did a final test with my shiny LED dreadlocks, I noticed that two parallel soldered strips suddenly lit up yellow instead of orange like the rest. Instantly I thought that I did something wrong during my soldering work, so I checked all visible joints. I couldn't find any mistakes. Next, I checked the code and unplugged all strips except those who turned yellow. Despite of running on the same code as the orange ones, the strips didn't want to light up in the right color. As a last resort, I opened a new package of unused LED strips, soldered them to my circuit and turned them on. You guessed right: The fresh LED strip lit up in the correct color! Turned out I received four meters of LED strip with a slightly weaker red channel. To get all 10 meters of LED strips for Nova's ponytail, I made multiple orders from Adafruit. Apparently that last LED strip I got had a slight color variation. I didn't even know this was possible but happy that I finally found the problem. I just changed the strips to some old ones I found in a box and everything was fine again!

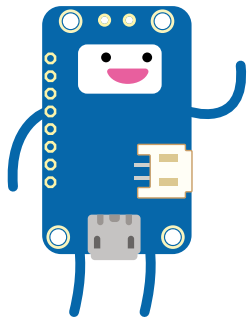
Slide switch mystery

After I finished Nova's armor, I did a final test of my circuit and then re-soldered everything to make it look nice and pretty. I turned the LEDs on again after that, but this time they would only run for 20 seconds before they turned off. The circuit dragged so much power that a fully charged 6600 mAh lipo battery was empty within mere seconds. My Powerboost got super hot too! There was clearly a short circuit, but where? All soldering joints were fine and the circuit looked as it was supposed to look. I started isolating the problem by unplugging the whole armor until only the Trinket, the Powerboost, the battery and a few cables and switches were left. It still got grilled when I turned it on. I got rid of the switch on the Powerboost next. Surprisingly everything was fine now! At this point I remembered that I actually changed the slide switch to an identical looking one during my re-soldering. After some research I found out that a few slide switches tend to create short circuits when they are connected to BAT, EN and GND on a microcontroller. I only had to get rid of the cable to BAT and it worked again.

Trinket has a bad day

And finally something really weird. I just wanted to do a last edit of my code. While everything worked fine the day before, when I tried to upload my changes to the Trinket, I constantly got a 'could not find a USBtiny device' error message. Switching USB ports and cables didn't work. The only option I had left, was that something was wrong with the Trinket. To confirm this, I plugged in another Trinket and tadah, the computer got a connection and I was able to upload the code. Switching back to the original Trinket, it finally got recognized as well and I finally was able to continue working. While this issue is indeed strange, it's actually a pretty common problem. Always have a second Trinket ready, just in case.

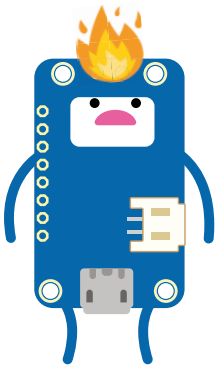
As you see, isolating the root of the problem is usually the best option to find out what's wrong. Get rid of every unnecessary component and unplug everything until you notice a change. Yes, there are some common problems like short circuits or connection issues, but it can also be something you would never even thought of. Don't rip your hair out, be patient and search smartly. We all make mistakes and hardware as well as software can be faulty, but there is nothing that cannot be solved.



If you've followed all the instructions in this book, your circuits, codes and your LED strips should work and turn your costumes into shining wonders. In theory at least. In reality you'll notice that short circuits are actually pretty common and one error message rarely comes alone. Here are a few helpful tips so at least you are better prepared!

Your Powerboost needs power too

When you connect a Powerboost to a Trinket, keep in mind that you'll also have to plug in a lipo battery to power your circuit. The USB connection alone will not be enough anymore to turn your LEDs on. I forgot this too once and then wondered what was wrong with my circuit.

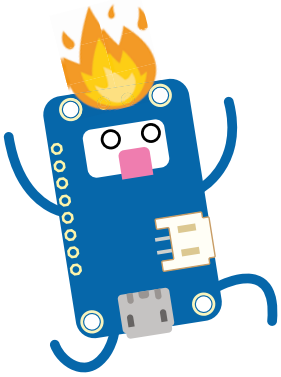


Avoid short circuits

I'm personally pretty impatient and never want to wait until I test my circuit. I just test them without insulating any wires or cleaning my workspace from left over solder and bare copper. You can imagine, that I'm basically a walking short circuit waiting to happen. Don't be like me! I even touched a bare and running Powerboost with my sweaty hands once and grilled not only my finger tip, but also the micro chip! Better make sure to insulate your soldering joints, clean your workspace and touch chips only on their edges when you're using them!

Make solder joints shiny and chrome

One of my worst enemies: Cold-soldering joints. They often appear for no reason and cause me a lot of headaches. While the circuit is just fine one moment, nothing works a second later. So keep your eyes open for dull and gray looking soldering joints and fix them immediately. Try to reheat the solder or get rid of it with a soldering pump or your soldering iron and redo this connection again. Soldering joints should always look shiny and chrome!

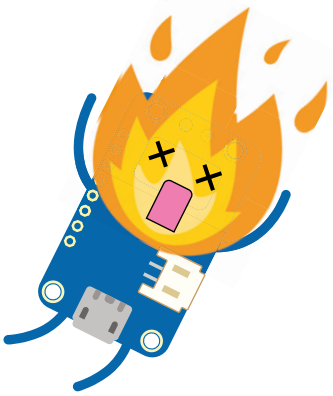


Wake up a sleeping battery

If your battery was left unused for a longer period of time, it can happen that it won't turn on your circuit anymore - even when it's fully loaded! In this case connect a Powerbank or your PC via USB to your Powerboost and turn it on while the 'sleeping' battery is connected. Now it should work again. This is the reason why I always bring a little emergency Powerbank to cons with me.

Error: Could not find USBtiny device

One of the most frequent errors I get is 'Could not find USBtiny device'. It mostly appears when I hit the reset button on a Trinket too late and I'm not fast enough to upload my code. It also occurs, when I forgot to change connected microcontroller under 'Tools' after I switched from an Arduino to a Trinket. Or - I simply forgot to plug in the USB cable. Yes, it happens and it's not a very proud moment for me.

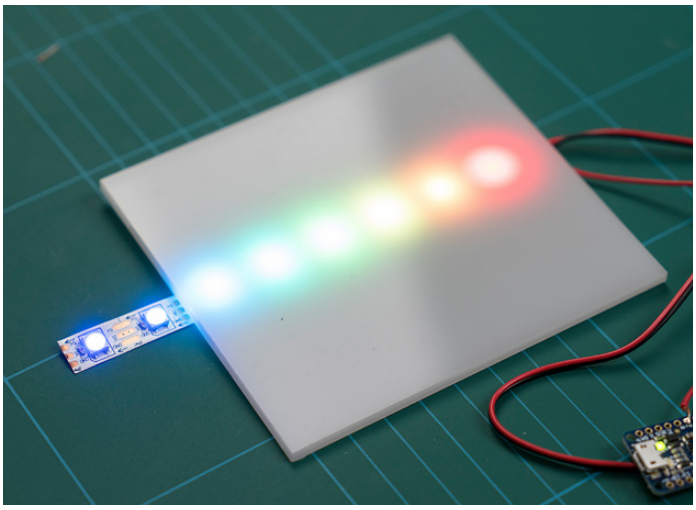


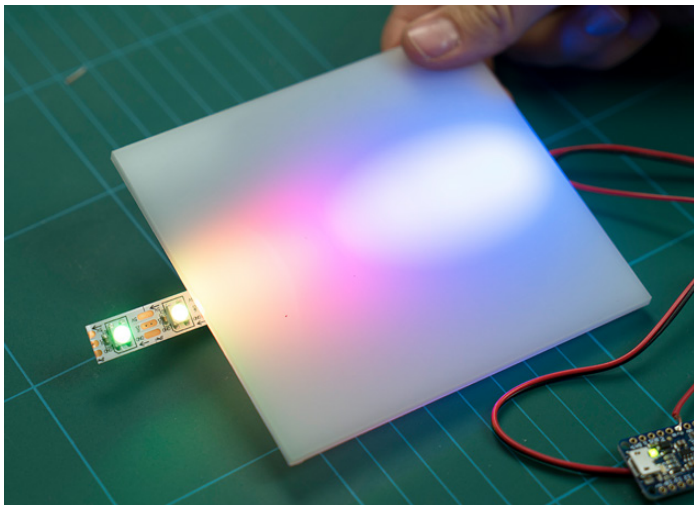
' ... ' was not declared in this scope

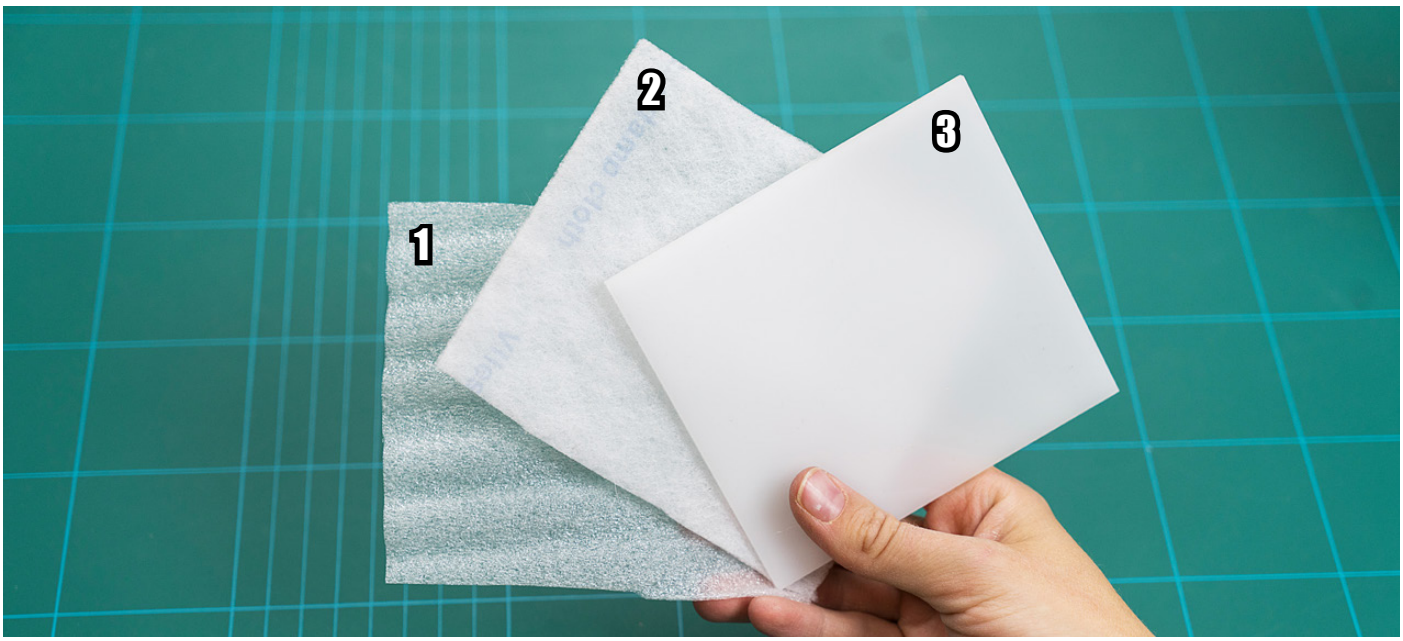
After working a bit with code now, you might have noticed that every time you open a bracket you also have to close it again. This applies to ( ), { } and [ ]. If you forget to close them, your Arduino IDE will complain with the error message *'... was not declared in this scope'*. It mostly happens to me, when I copy and paste two sketch parts together and forgot a line. Just make sure you don't miss any brackets when you copy! You can also just click on a bracket in the software and it will mark you it's counterpart. Then it's pretty easy to find the little culprit!

expected ';' before ...

The semicolon **;** might give you similar trouble. If you get the error message *'expected ';' before ... '*, check the mentioned line of code and add a **;** if necessary. Every command needs to be closed with this symbol. The Arduino software is really picky about its code!







Diffusing lights and more

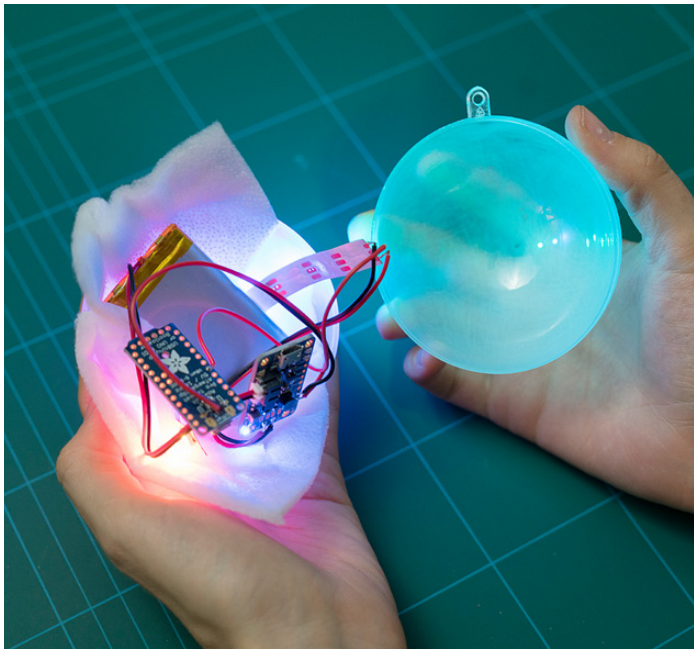
Besides having well soldered circuits and some cool light animations, you should also consider diffusing the lights of your LED strips properly. Bright dots simply look unnatural and don't really fit to any costume or prop. With the right diffusing material though it's pretty easy to fix that.

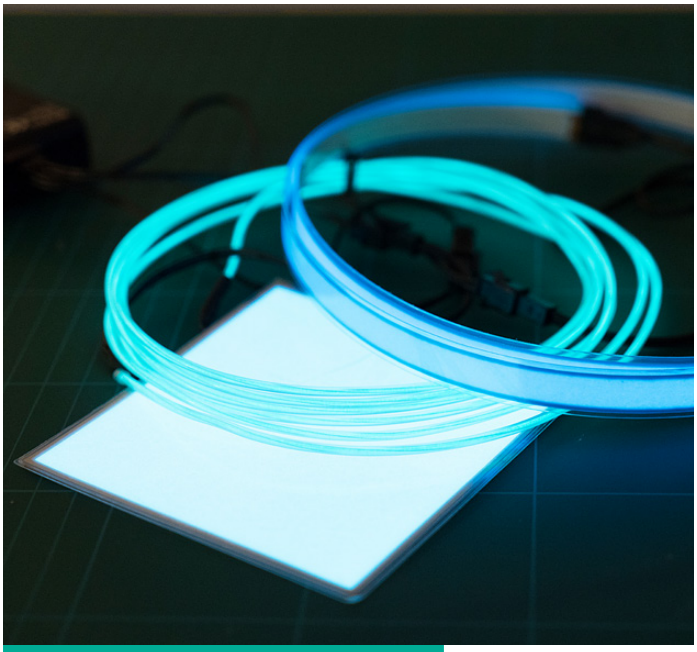
For the example at the top of the page I used milky acrylic to spread the light of the LED strip below. On the left the sheet is placed directly on the strip, while next to it added around 1cm (0.4inch) of space in between. As you see, there is already a huge difference when you compare them. However even the best diffusing material won't work if there is not enough space between so the light can spread.

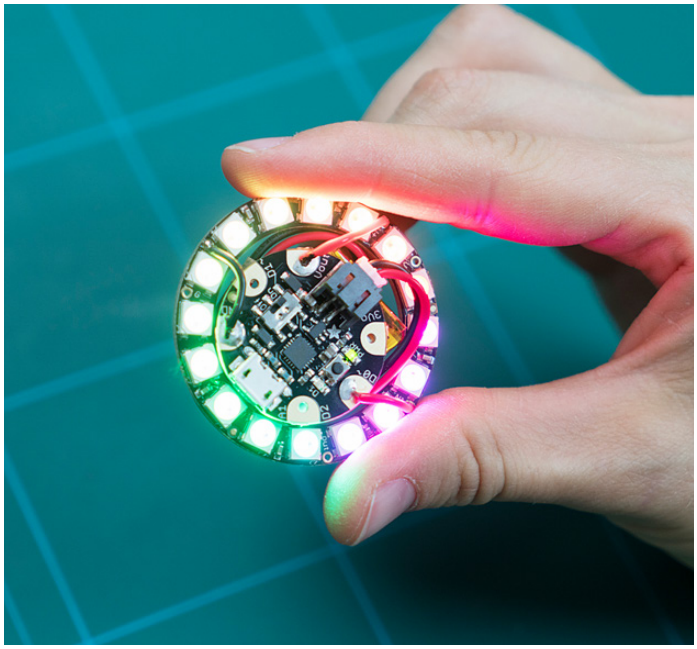
A pretty cheap alternative to acrylic is foam packing material **[1]**. I just buy them in big rolls from the floor department in my local hardware store. Apparently it's also placed underneath wooden floors to reduce noise. As you saw in my Shadowmourne example, I'm always using multiple layers to get a good effect.

Those of you who sew surely love to hear that volume vlies works great to spread lights as well **[2]**. It's just as cheap as the foam, but diffuses light even better. Volume vlies (cotton batting) and LED strips are also a great combination to create lit up clothes, which might be a really cool idea for a fun conventions or a Halloween parties.

For Nova I mainly used milky acrylic **[3]**. It diffuses light better than the other two options but can only be cut properly with a band-saw or a laser cutter. Since I had so little space between the LEDs and the diffusion in my armor parts this was the only viable option for me.







NOTE

EL wire, tape and panels are great solutions to light up your costumes quickly and with minimal soldering. They mostly come pre-wired and include a power source and a switch. Just turn on the power and enjoy the light. EL products however shine very dim compared to LED strips. You also cannot program them, they only come in specific colors and you need a bulky power inverter to light them up.

I'm not the biggest fan because of these reasons, but who knows - maybe it's just the perfect solution for you!

Stay curious and try new things out!

Congratulations! You've made it to the end of this book! I don't exaggerate when I say this was definitely my hardest book to write yet. Animating LEDs can be very intimidating from the outside but I hope I was able to show you that once you're brave enough to look into it, it's actually easy and a lot of fun!

The examples I showed you on the previous pages are only a small fraction of what is possible with the knowledge you now gained. Combining different materials is the best way to get some pretty stunning results. For the 'magical orb' above, I simply sanded the surfaced of an acrylic sphere with a 120 grit sand paper and filled it with volume vlies. Adding a short LED strip, a chip and a power source brings it to life. The result is super pretty isn't it?

There are a lot more things for you to try out that I haven't covered in this book! There are round LED strips, LED matrix panels, tiny LEDs to sew onto fabric, motors and sensors that turn your lights on when they hear sound or when you shake them. I just wanted to give you a place to start.

The rest is now in your hands! Go and have fun!

Book number nine, check!

Thank you so much for supporting us.
I hope it was helpful! Please leave me a message if you have more questions!

If you need more help, check out our website for tutorials, write-ups and videos:

kamuicosplay.com

If you've already made a costume or prop using my techniques, I want to share it in my gallery:

kamuicosplay.com/epiccosplay

You can always find us on:

facebook.com/kamuicos

instagram.com/kamuicosplay

youtube.com/kamuicosplayofficial

twitter.com/kamuicosplay

First Release, 2017
Author: Svetlana Quindt
Editing & Layout: Benjamin Schwarz
Germany

www.kamuicosplay.com