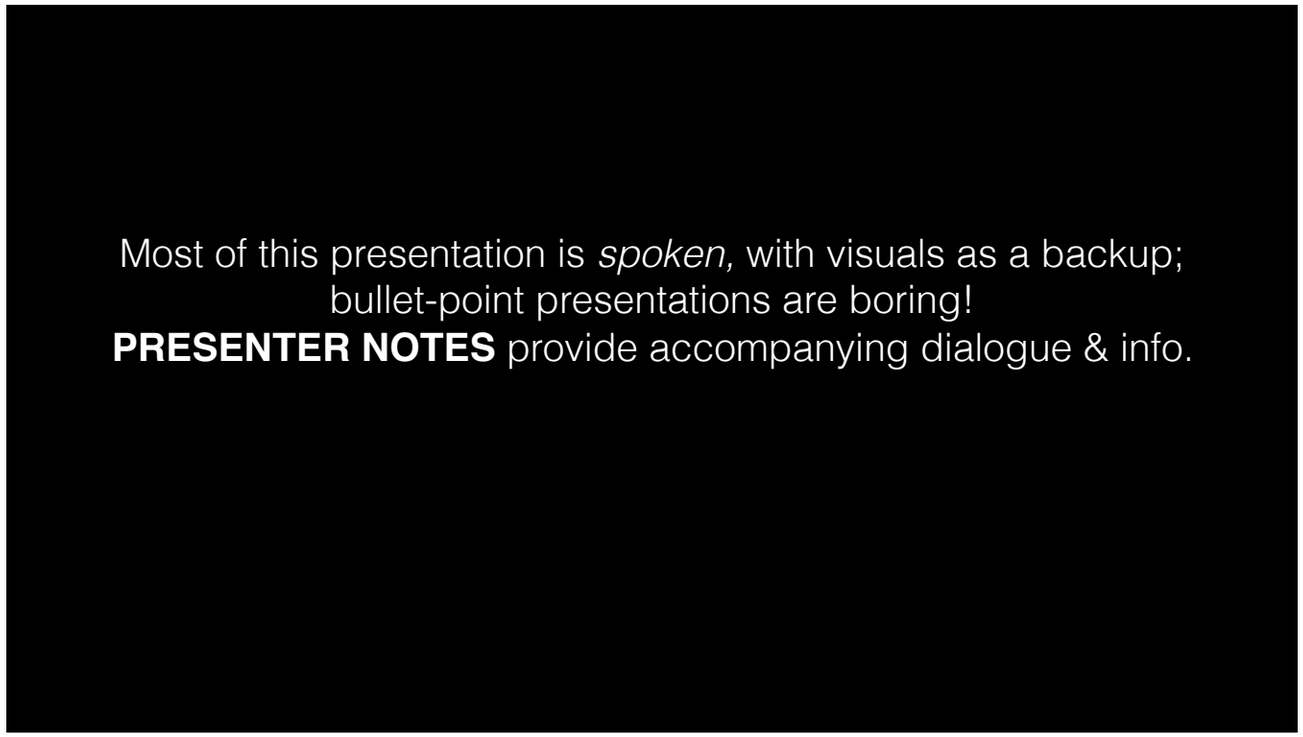
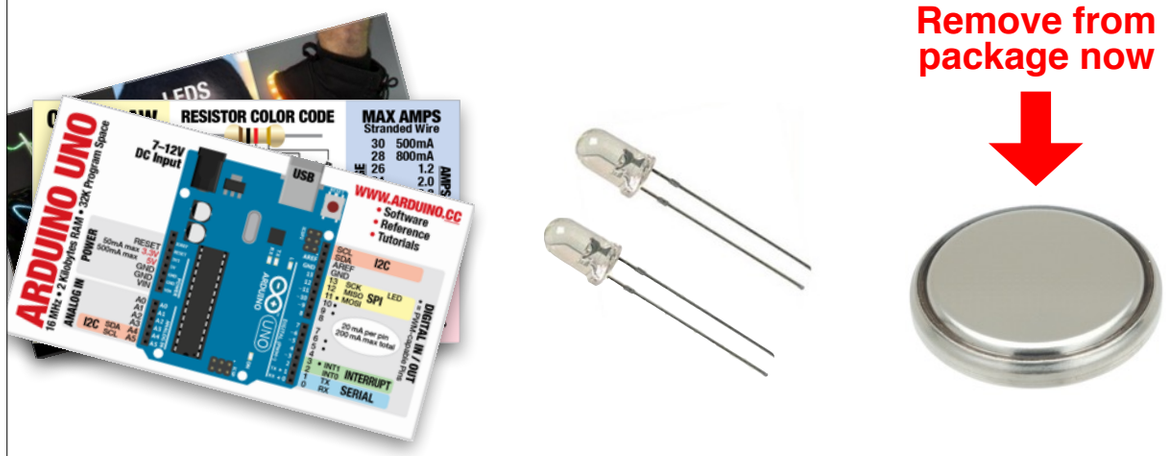


NOT DIALOGUE: This first section (about 8 minutes spoken) is an introduction that doesn't get into any actual electronic theory. This is on purpose. The average information retention in any talk is actually pretty low. My intent at the start here is as a cheerleader, to help instill confidence and curiosity while everyone's still alert. Thus, even if the audience doesn't recall all the details that follow, they'll hopefully have the desire to study further at their own pace.

A note about images: most were taken from the Adafruit web site, but a few were snagged here and there off the web. This legitimately falls under Fair Use — for teaching purposes, and not republished for sale — but as a "good neighbor" gesture it behooves me to replace these with my own images (or other Creative Commons sources), which will likely happen incrementally as I rework this for different talks.

If you can use all or part of this presentation in your own talk, or as an outline for your own, that's fantastic and I would encourage it. The presentation itself, and any photos or diagrams taken from the Adafruit site, are Creative Commons. Some images are from Wikimedia Commons and similarly usable. For anything else, you may want to substitute your own photos, which would probably be necessary anyway for targeting various audiences.



Leave this slide up during setup, as audience files in.

Each person is given a packet containing three reference cards, two LEDs in different colors, and a coin-cell battery.

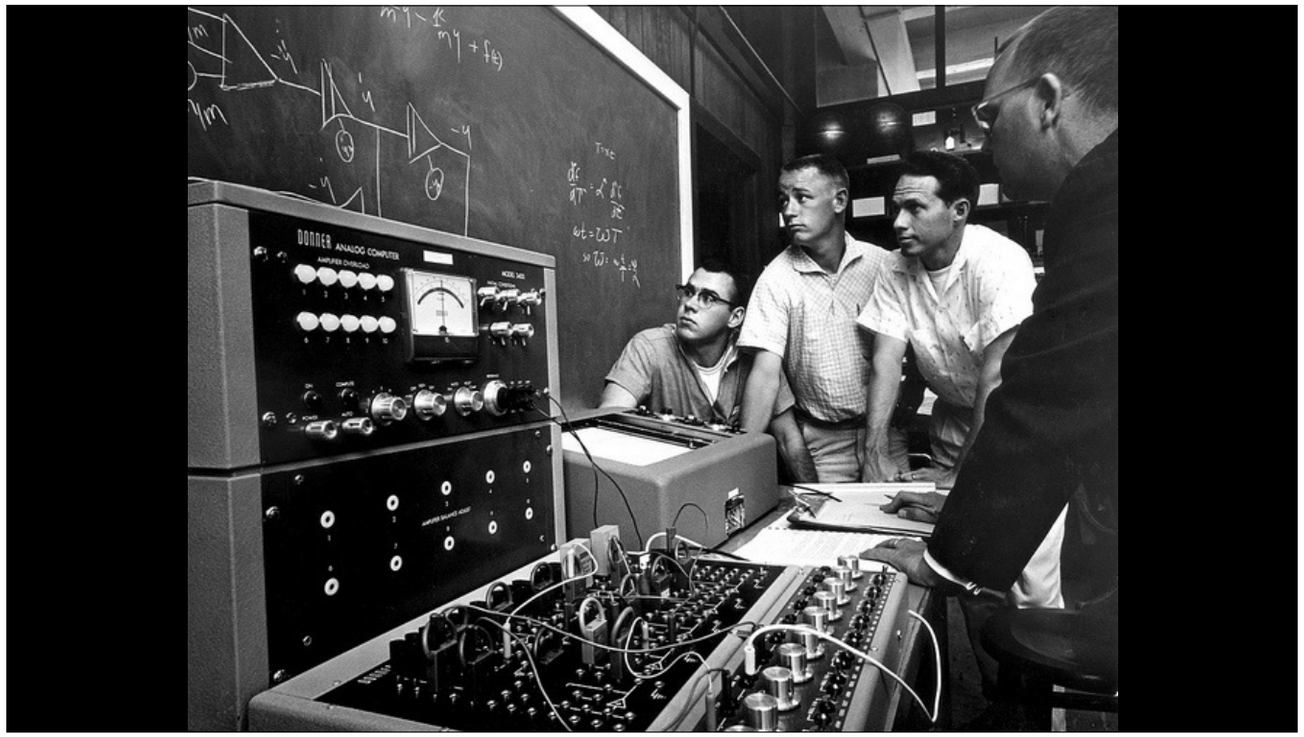
- Gives them something to do while waiting.
- They can start exploring and learning before the talk even starts.
- Battery package may be difficult to open; this gives a head start rather than struggling right when it’s needed.

Costume Electronics:
The Very Basics

Thank you for coming. This is an introduction to electronics for first-timers.
Show of hands: how many here are coming at this totally fresh?
How about any professionals, or serious hobbyists?
This talk is really targeted at beginners, but I'll try to keep it entertaining for everyone.
Also, any current or aspiring costume-makers?
That's the context of WHY we're here. Electronics in costume-making.
Just a heads-up: if you have photosensitivity, there will be some blinking lights later.

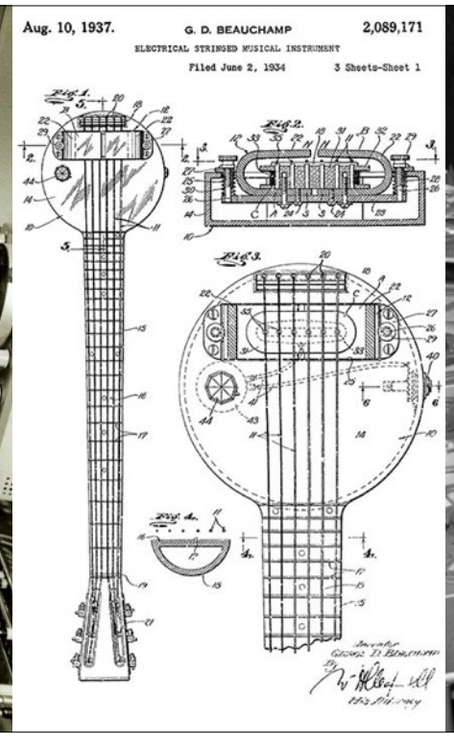
(Show-of-hands allows gauging the audience experience level, so talk can be adjusted accordingly.)

So…electronics!
ALL of us are ready USERS of electronics. Our pockets are FULL of things.
When it comes to MAKING electronics…electrical engineering…



…it tends to conjure images of nerds and equations and wires, and…okay, some parts of electronics ARE a little like that…but by and large, electronics is an extremely creative endeavor, so let's put this image to rest.

Take music, for example…


Aug. 10, 1937. G. D. BEAUCHAMP 2,089,171

ELECTRICAL STRINGED MUSICAL INSTRUMENT

Filed June 2, 1934 3 Sheets-Sheet 1

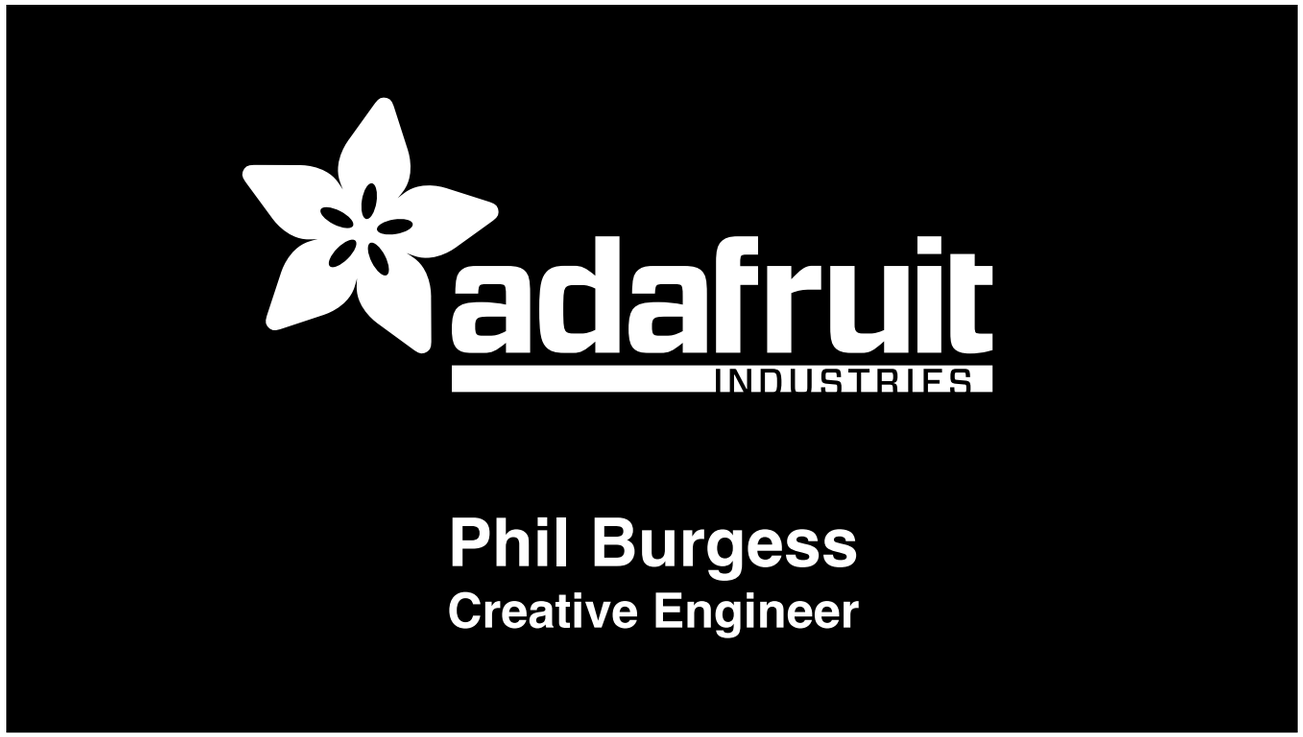
It took visionary, creative lunatics to invent ENTIRELY NEW KINDS of instruments and music using electronics…

…and still other creative lunatics to explore this new medium to its fullest.

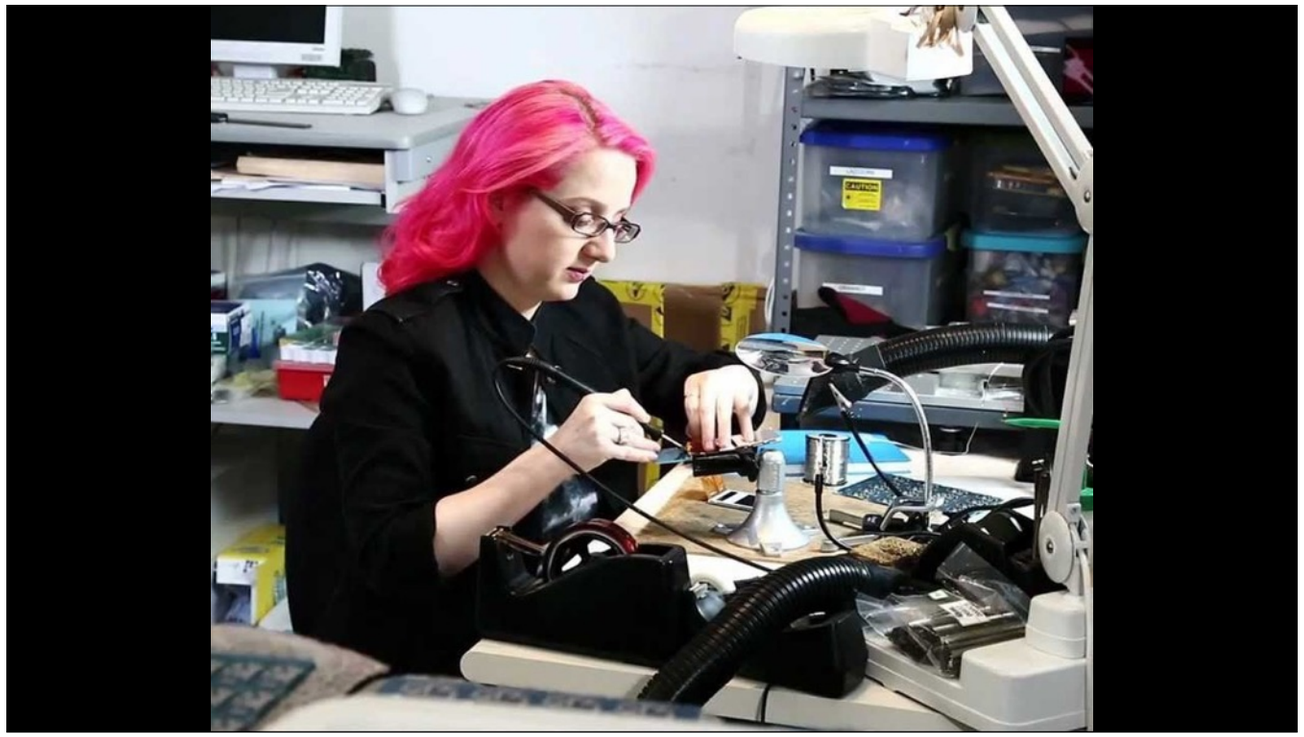
Well, you look around a convention like this one, and what do you see?
I see A FEW THOUSAND creative lunatics…

…little surprise then to see electronics creeping into our medium of costuming; it’s a natural fit for the culture.

These are some pretty sophisticated projects, but I’d like to start you on that path.



My name’s Phil, I’m a creative engineer for Adafruit Industries. We cater to creative lunatics, and others wanting to learn more about electronics.



My boss Limor. She has pink hair.

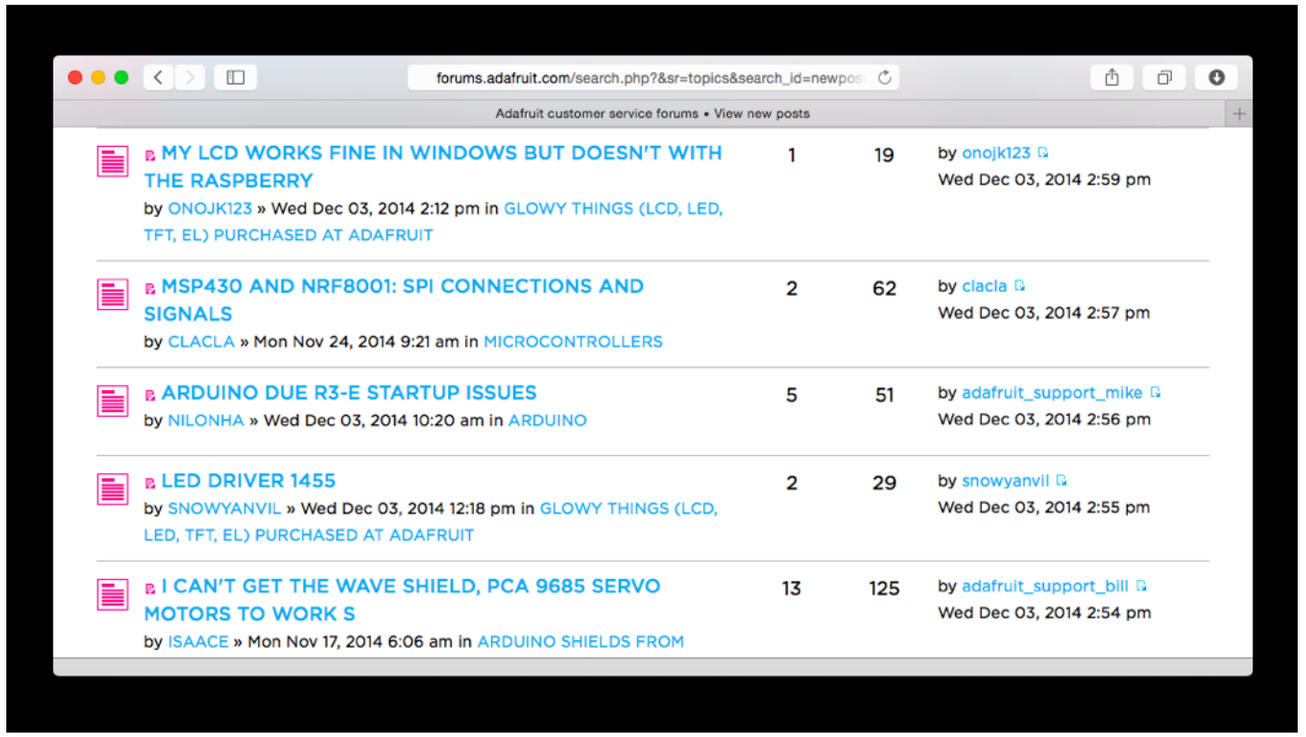
Part of my job at Adafruit is as a designer. I'm given some new part and asked to make an interesting item around it that people might like to have.

But here's the weird thing: once those recipes are finished, we don't make or sell these items. We sell parts. We GIVE AWAY the recipes.

There are a couple of reasons for this. One's very simple: we're an electronics teaching company; we're not interested in manufacturing shoes or hats.

The other's a little more philosophical. There's something hypocritical when a company sells you a mass-produced item to "express your individuality." But when you're invited into the creation process itself, you have the opportunity to understand that object inside and out…to learn from it, improve upon it, remix and truly make it a medium of SELF-EXPRESSION.

That’s an idea I think will resonate with this group.
I visit a lot of events…anime, sci-fi, comic conventions…the passion and craft on display are always amazing, but the creation of one’s own worlds and characters is something fairly unique to this fandom right here.



Another part of my job is technical support. Customers follow these recipes…but suppose they encounter problems? I help troubleshoot the issue and get them rolling again.

Doing this for a few years, you notice certain patterns…stumbling blocks that beginners consistently encounter. Those are the things I want to focus on in this talk.

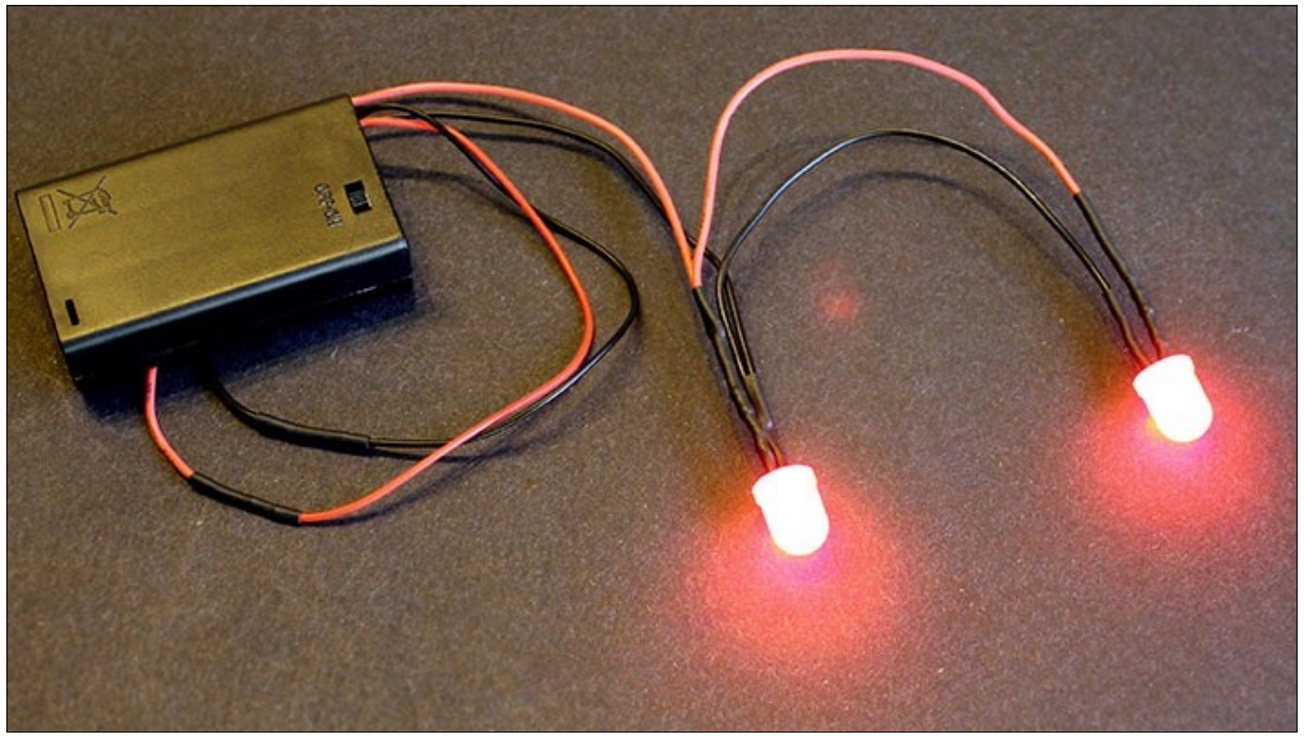
And when I'm not doing Adafruit work…

…I’m a cosplay nerd myself…I enjoy making and performing in crazy getups.
Lots of folks can teach you about electronics. But not everyone shares your perspective of surviving a big, sweaty costume. Hopefully I can make the topic more relevant.

My buddy Philosoraptor has a great saying…

KNOWLEDGE is learning that “Frankenstein” isn’t the monster.
WISDOM is realizing that Frankenstein IS the monster.

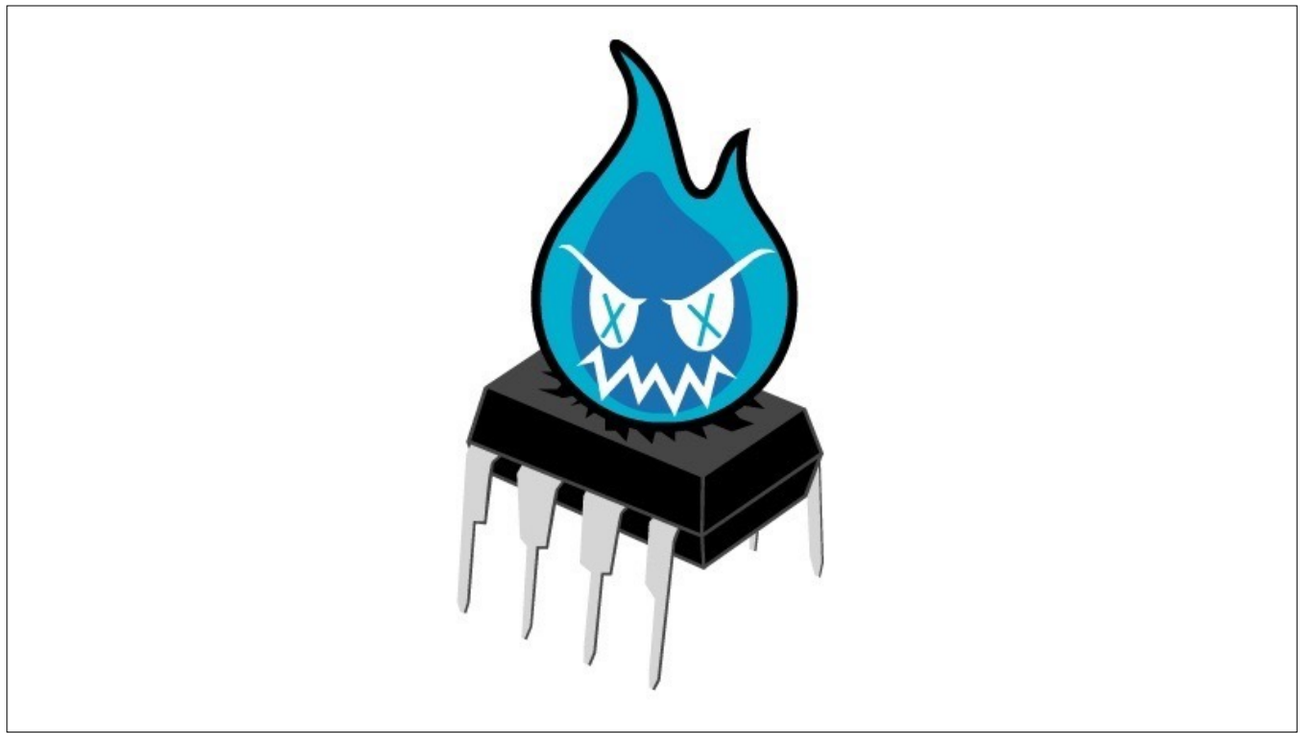
Electronics is a huge topic, and with only about an hour to talk here, I’ll be focusing on WISDOM. If you want KNOWLEDGE…



…my best advice is to dive in. Get a book, or go online, find a small project and start exploring! As tool-using monkeys, we're evolved to learn through our fingers. Get them dirty!

As for WISDOM…

If you take NOTHING ELSE away from this talk, the SINGULAR MOST IMPORTANT POINT I need to make…
To succeed at electronics, you HAVE to give yourself a free pass to SCREW UP. This goes against how we're taught in school and in life. But electronics can be abstract and invisible…talk to anyone who's done this a long time, I can GUARANTEE they'll tell you that some of their most profound insights came when they utterly botched something. Let it go.



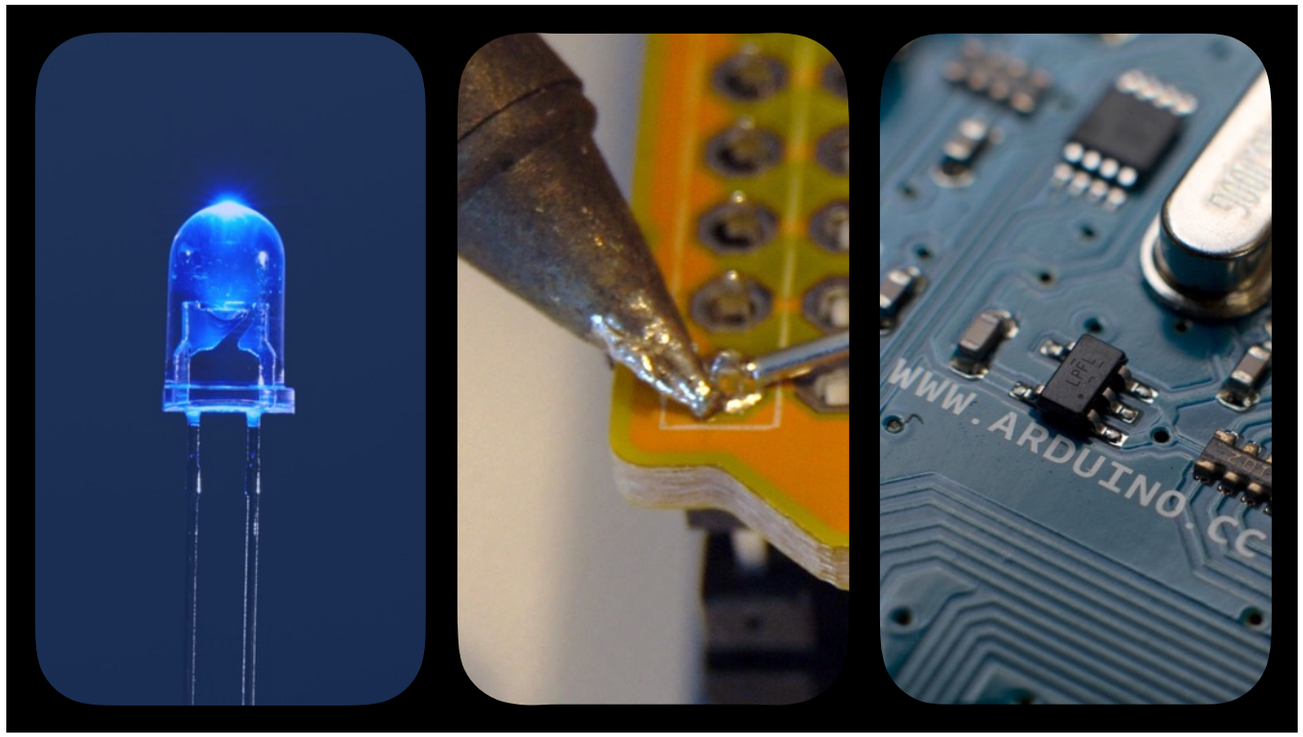
It’s practically a rite of passage, “letting the magic blue smoke out.”

Embrace failure. Pick yourself up and try again, a little smarter.

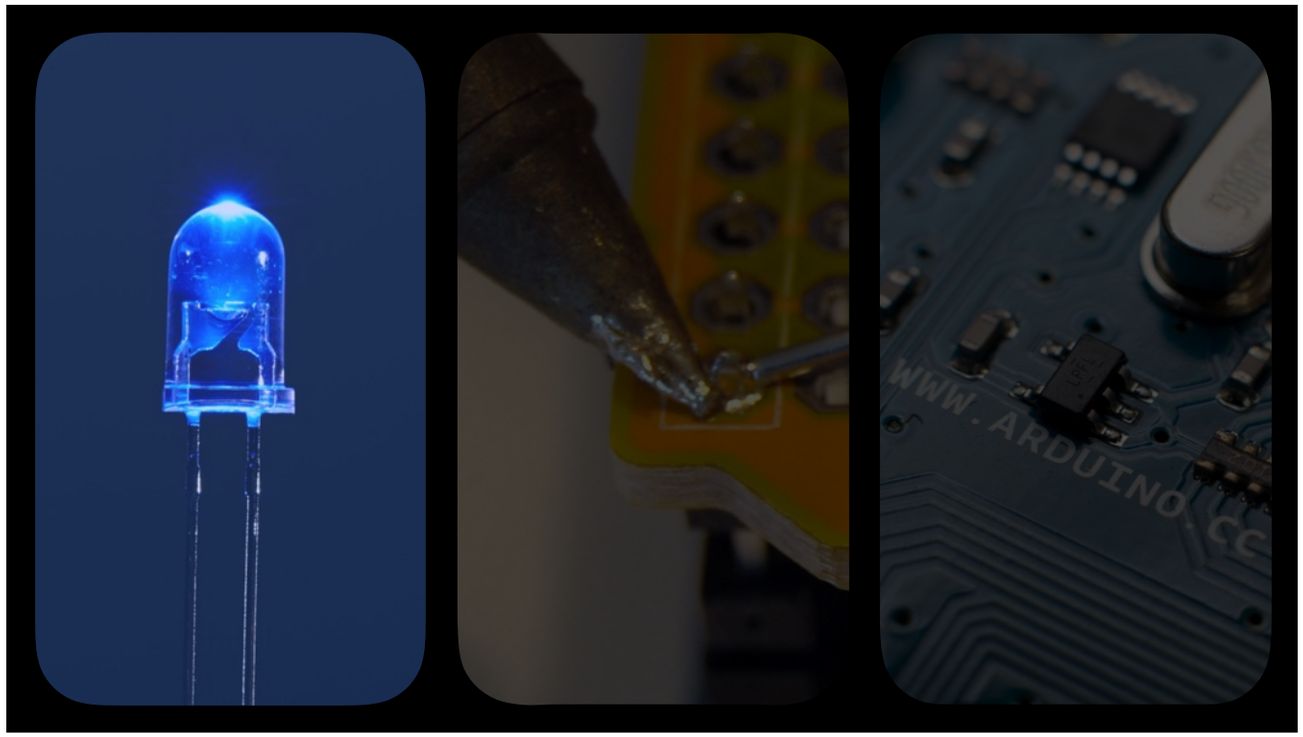
Interestingly, one of the few other places I’ve seen this sort of encouragement…

…is in sewing and costuming. Like that first time you try to sew a zipper, and it ALWAYS comes out wrong, maybe inside-out and backwards. You feel ashamed but your mentors smile knowingly. Their first zipper was just as bad. You’re on the right track.

You’re in good company here.

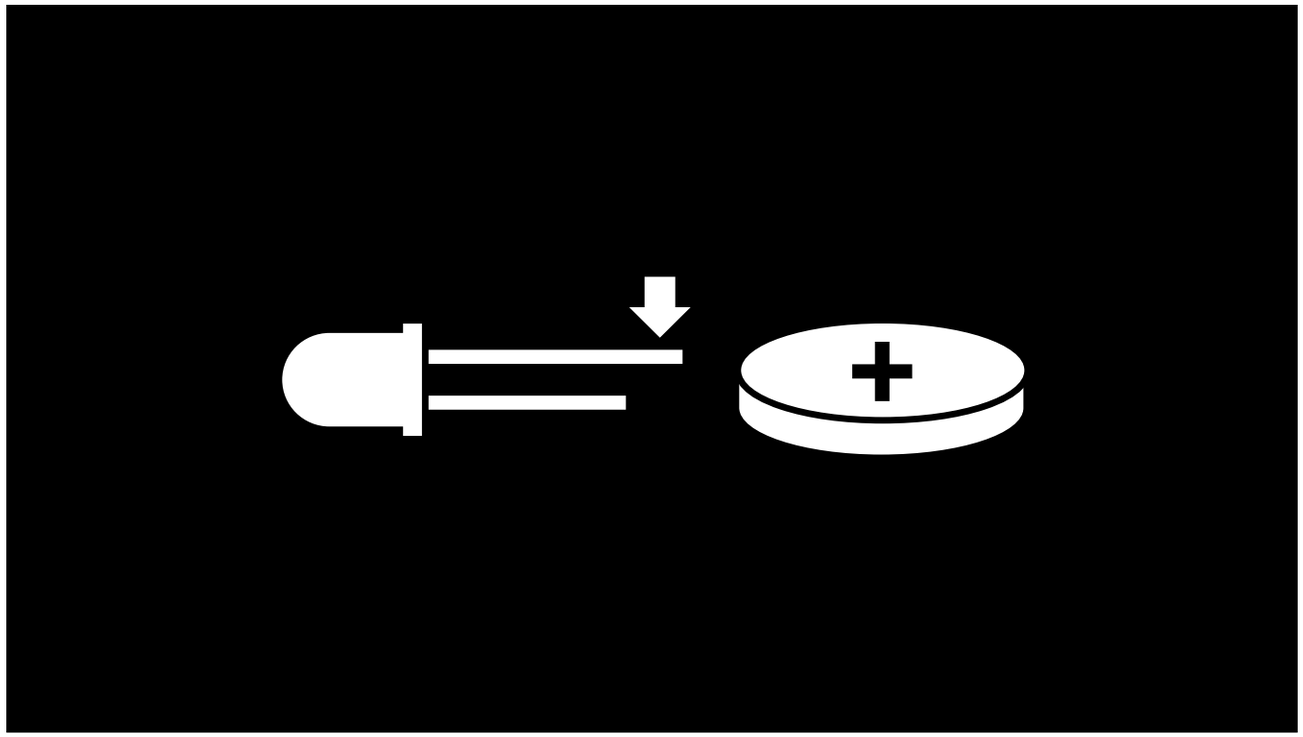


There’s three parts to this talk, each focusing on one of those core topics that stymie beginners. Afterward I’ll provide some resources for learning and finding things and we’ll have a Q&A session.

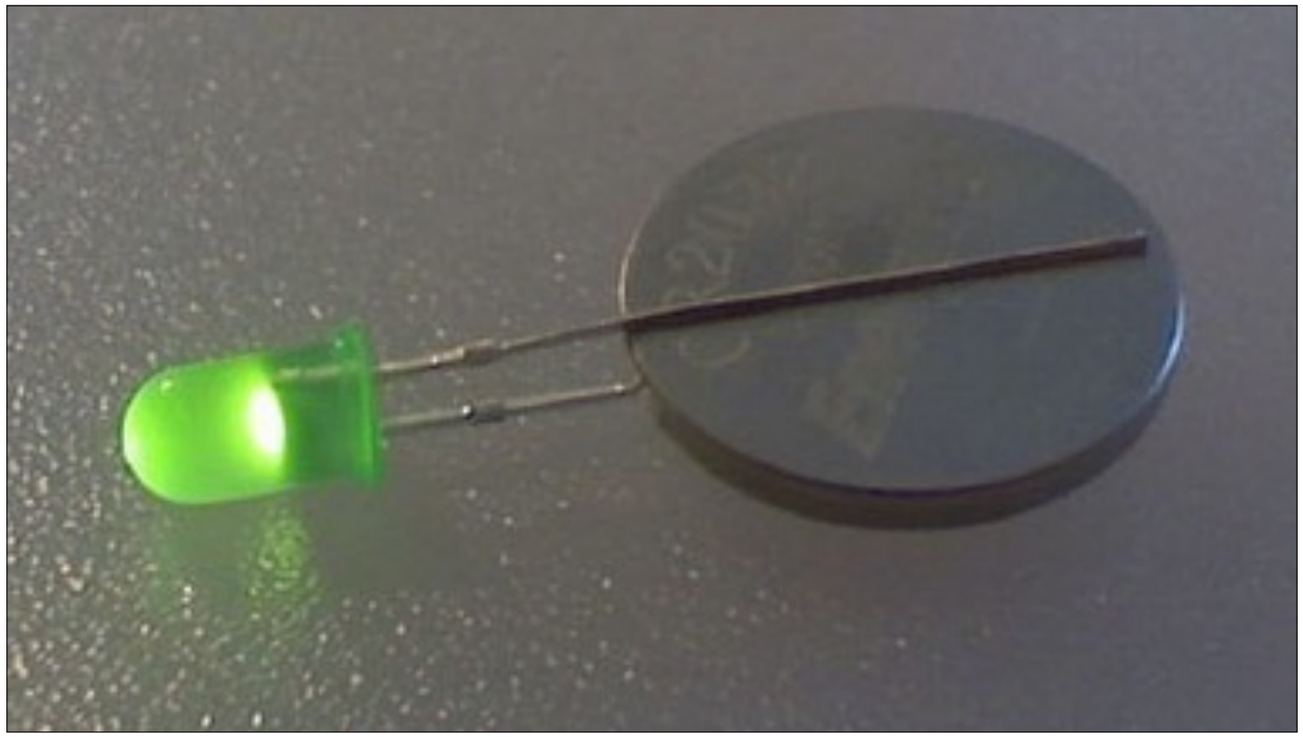


First part is about LEDs. We’ll use this as an opportunity to learn about the foundations of all electronics.

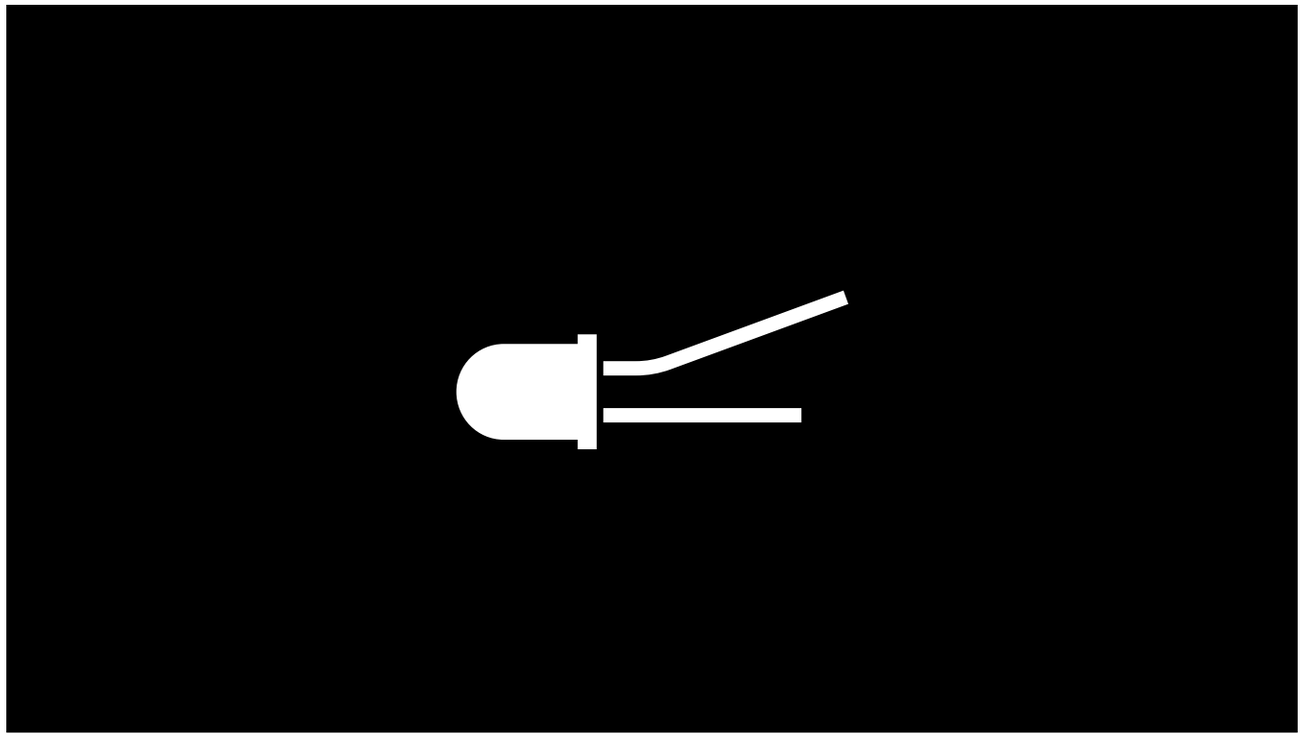
A detail like these glowing eyes adds a LOT of impact to a costume! What’s more, it’s simple, inexpensive, and teaches everything you’ll need to get into more advanced projects later. I really encourage something like this as a starter project!



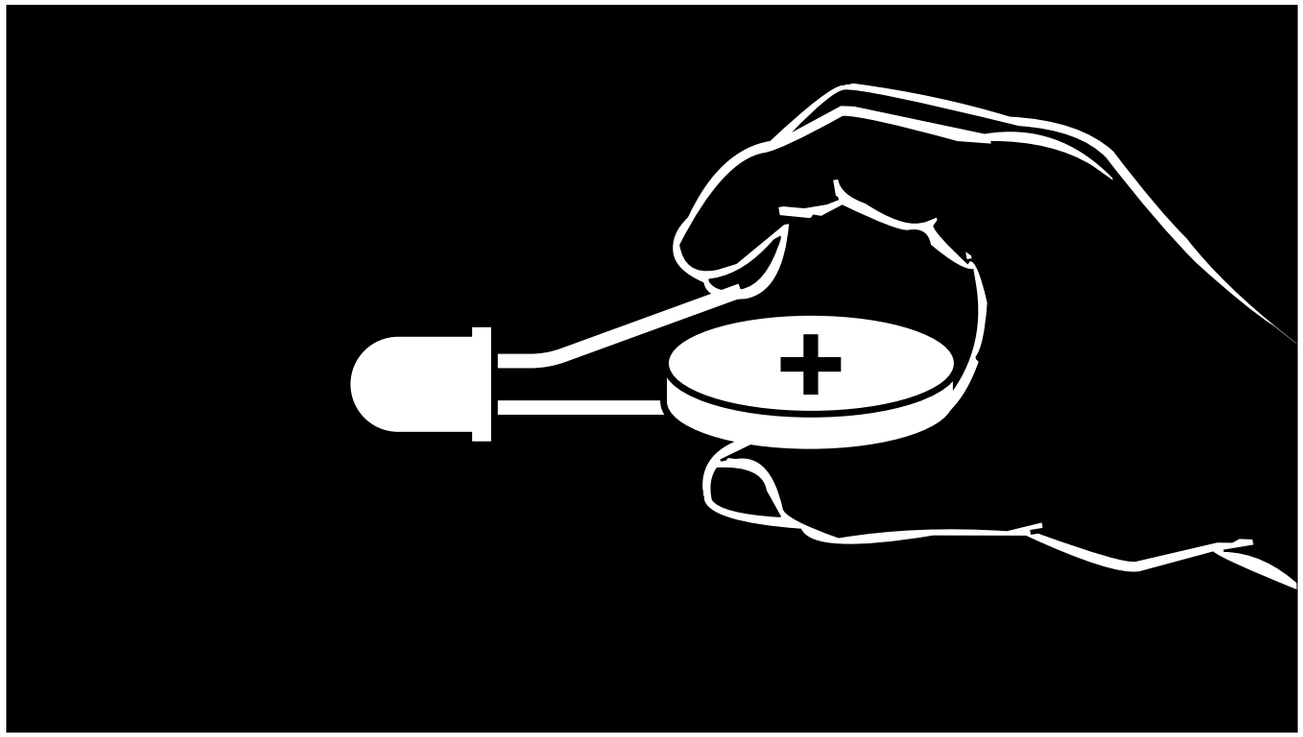
Take the battery and one of the LEDs out of the packet you received. Look closely, you'll notice one leg is a little longer than the other. Press this longer leg to the '+' side of the battery, and the other leg to the opposite face…



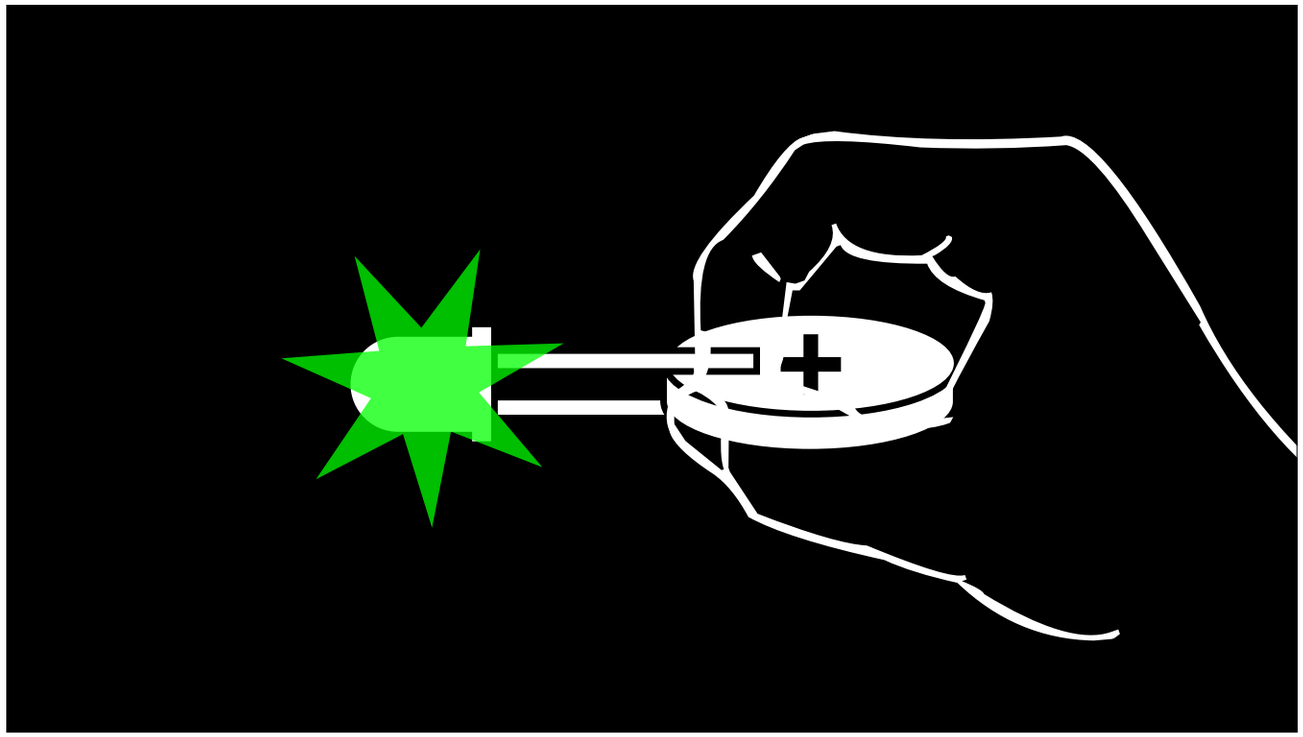
It should light up. Cool! Flip it over if it doesn’t.



Now let’s try a thing here…
Bend the longer leg up a bit, so they’re splayed…



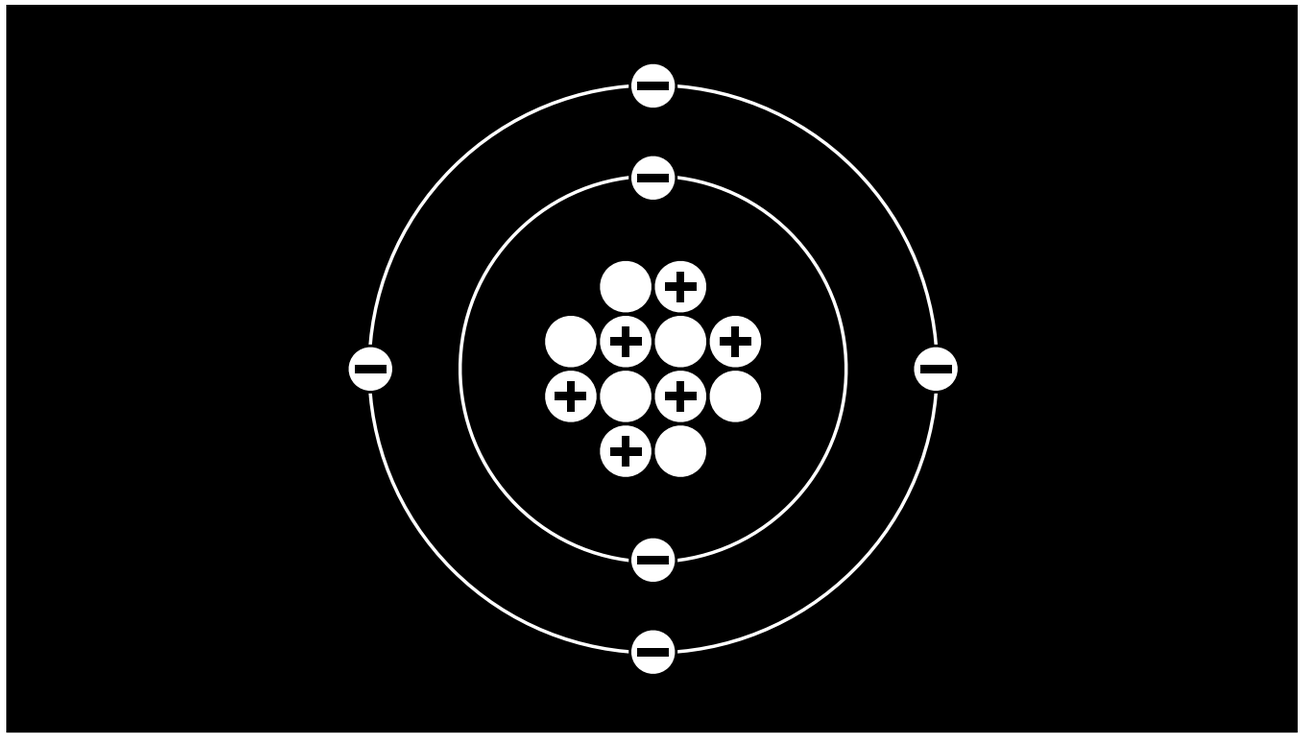
Hold the other leg to the battery, and give it a pinch…



…the LED flashes on and off.

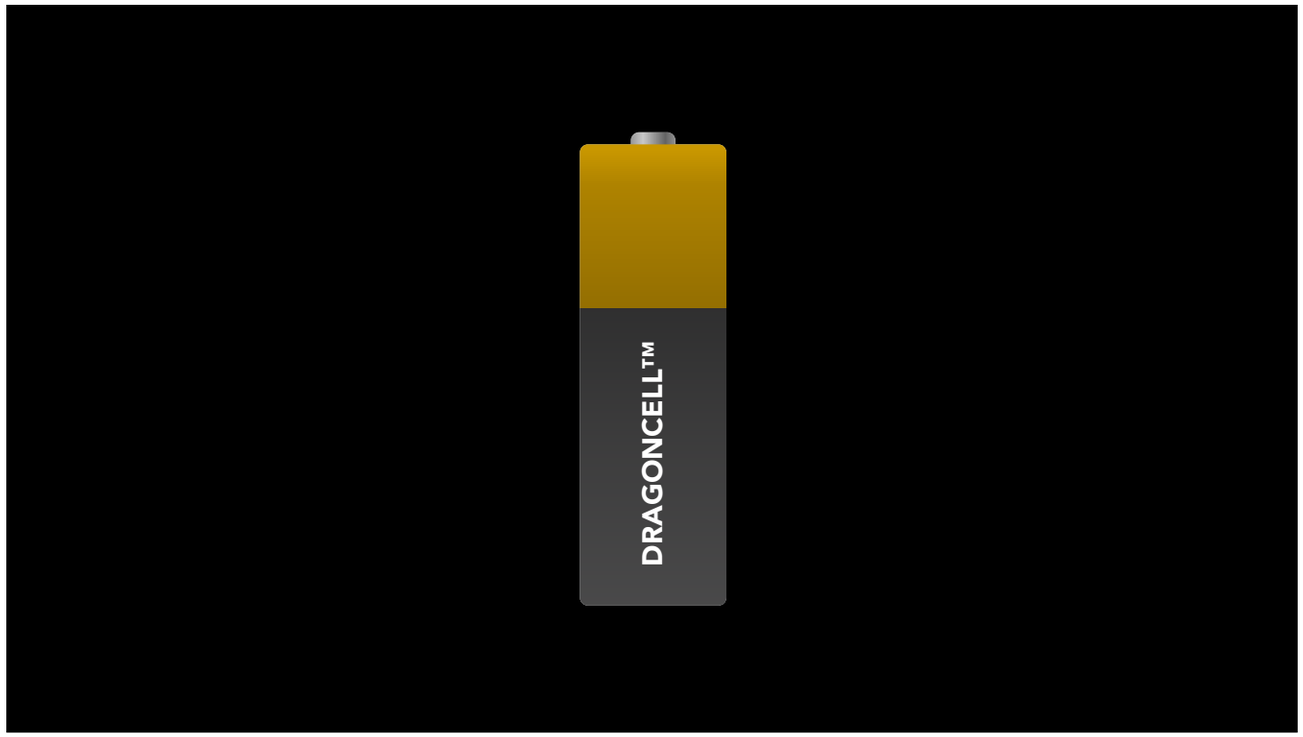
(Go back & forth between this & prior slide to animate.)

That’s cute, but actually we can learn a LOT about electronics from just this simple thing! I’ll break it down for you…in fact, lets break it ALL THE WAY down…



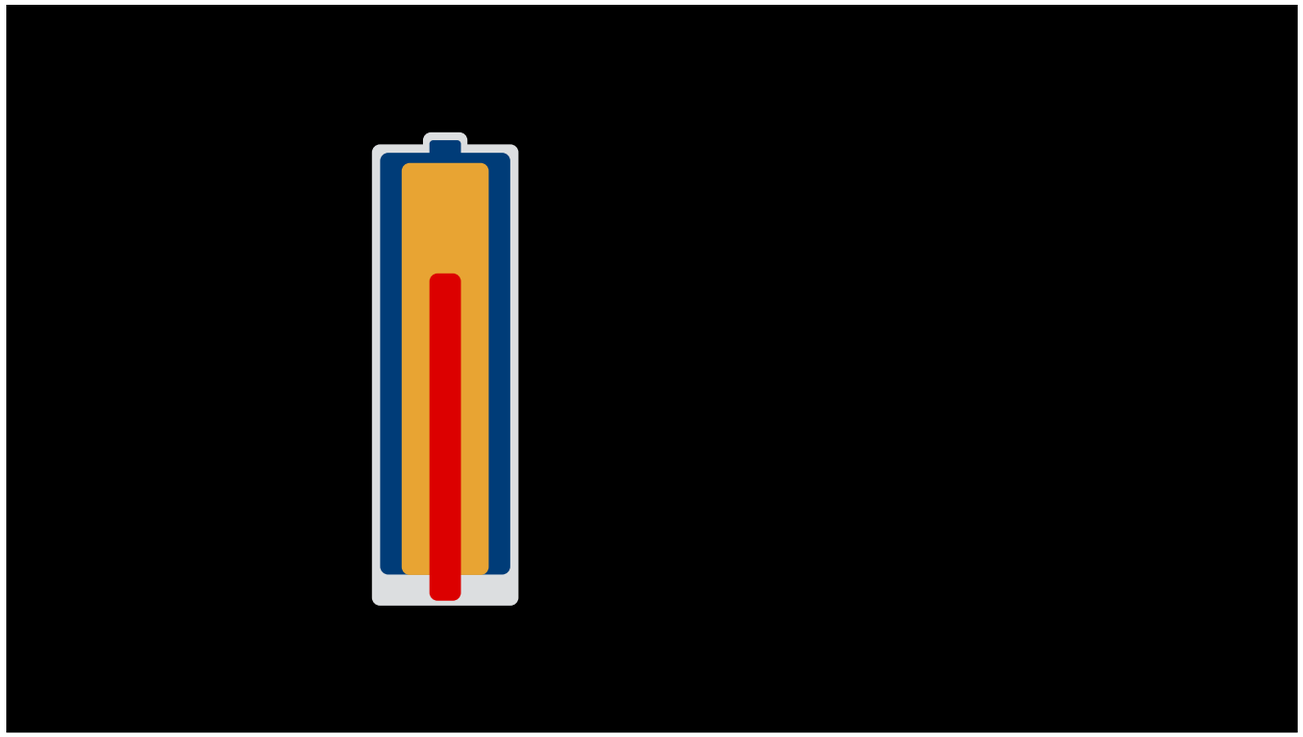
…to individual atoms.

Orbiting the nucleus of an atom, you have these guys here…electrons. They orbit in these very specific arrangements…shells…each shell becoming more stable as it approaches a particular number of electrons.
When different atoms come together, they'll share or exchange electrons to better pack these shells; a chemical reaction. It's why things rust, for example. This compulsion for stability is an INCREDIBLY potent force of nature.

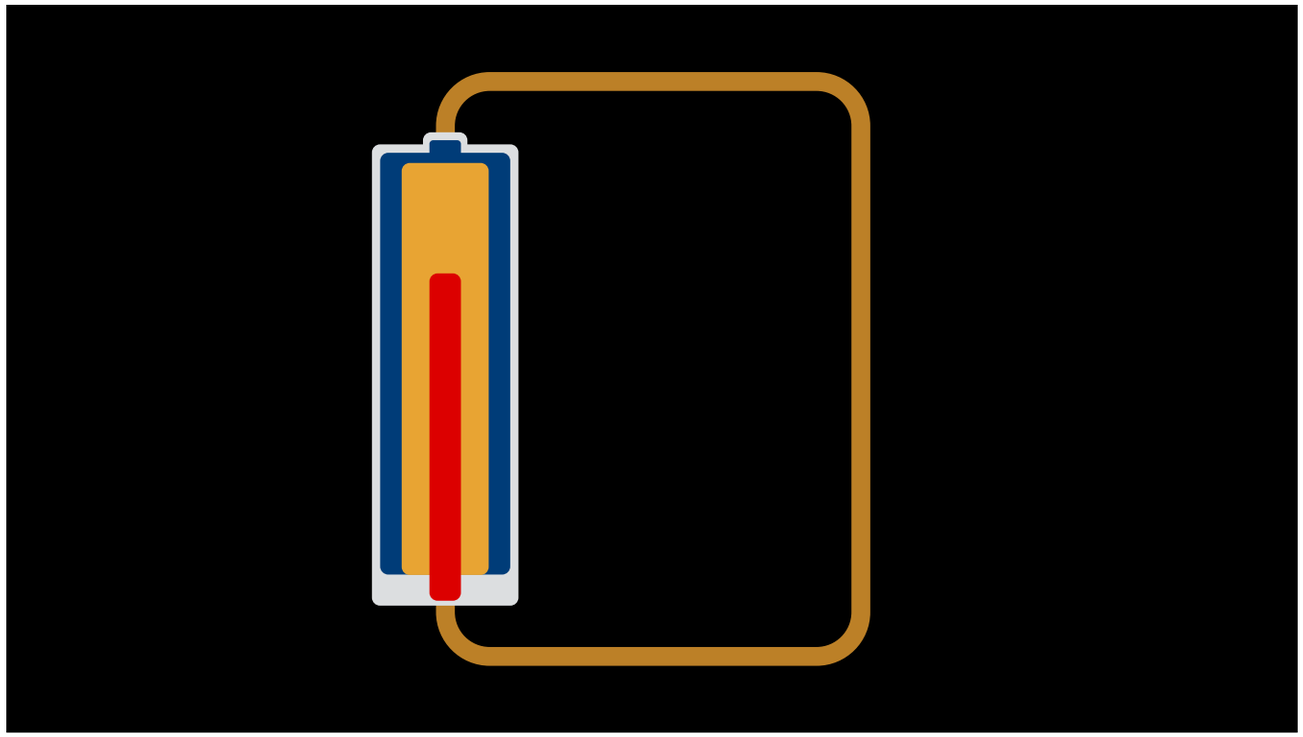


Suppose you took a battery and cut it in half with a band saw…

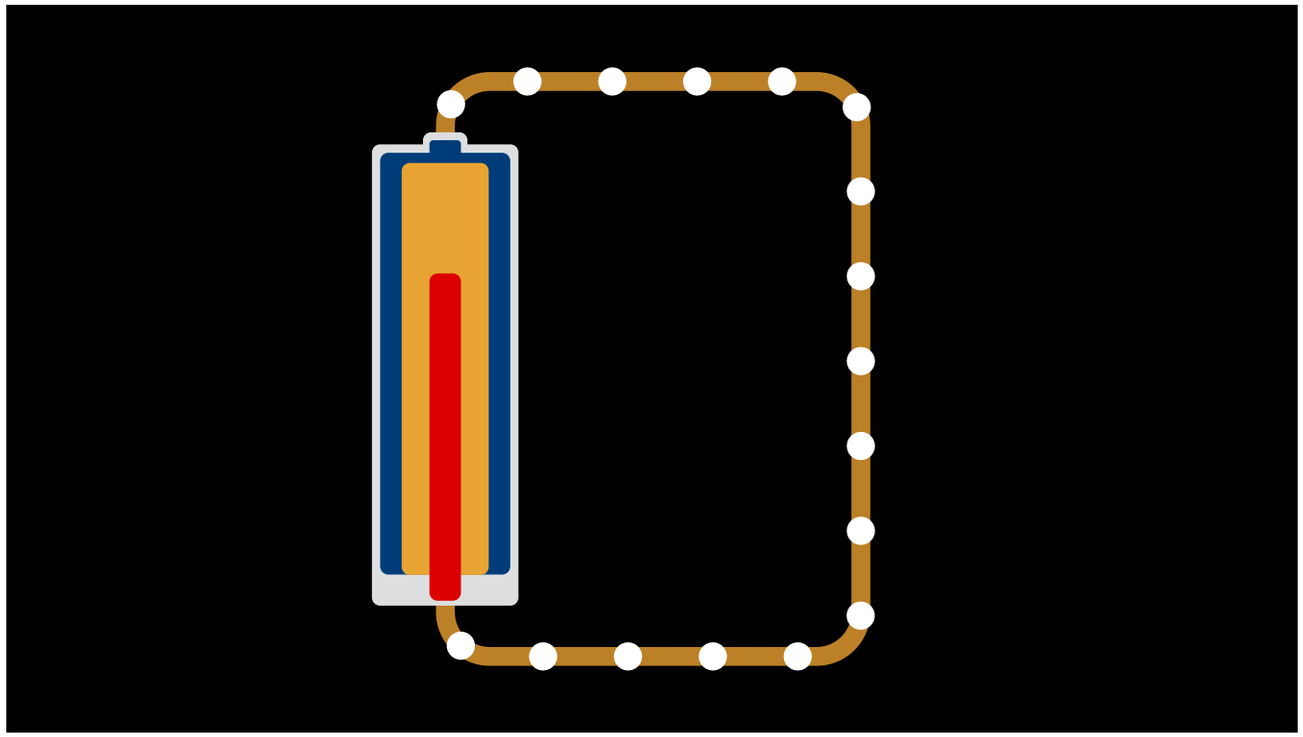
Don’t actually do that! But as a thought exercise…



Inside, you’d see a specific arrangement of chemical compounds that for all the world would love to engage in one of those reactions. But their careful placement makes this impossible; there’s no direct contact between the materials wanting to react.

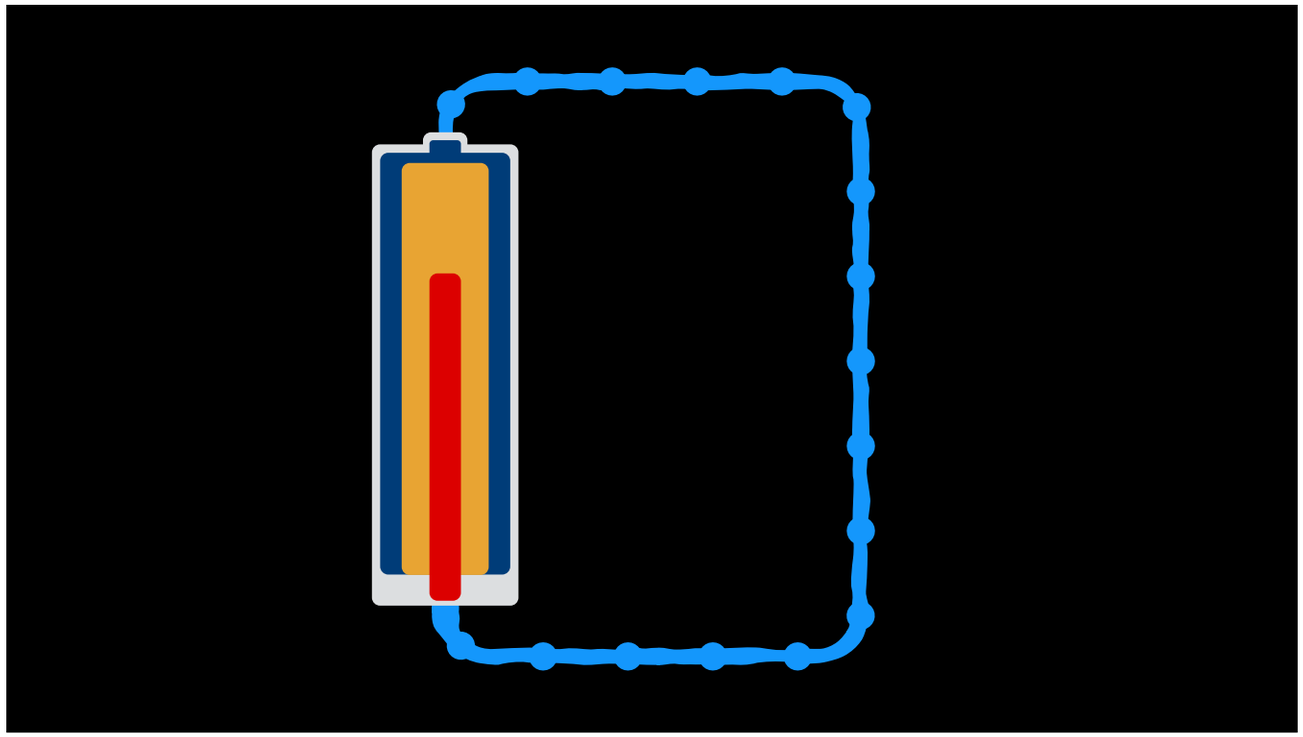


But if you provide a conductive path — a piece of copper wire, for example — between the two ends of the battery…don’t do this either, still a thought exercise…



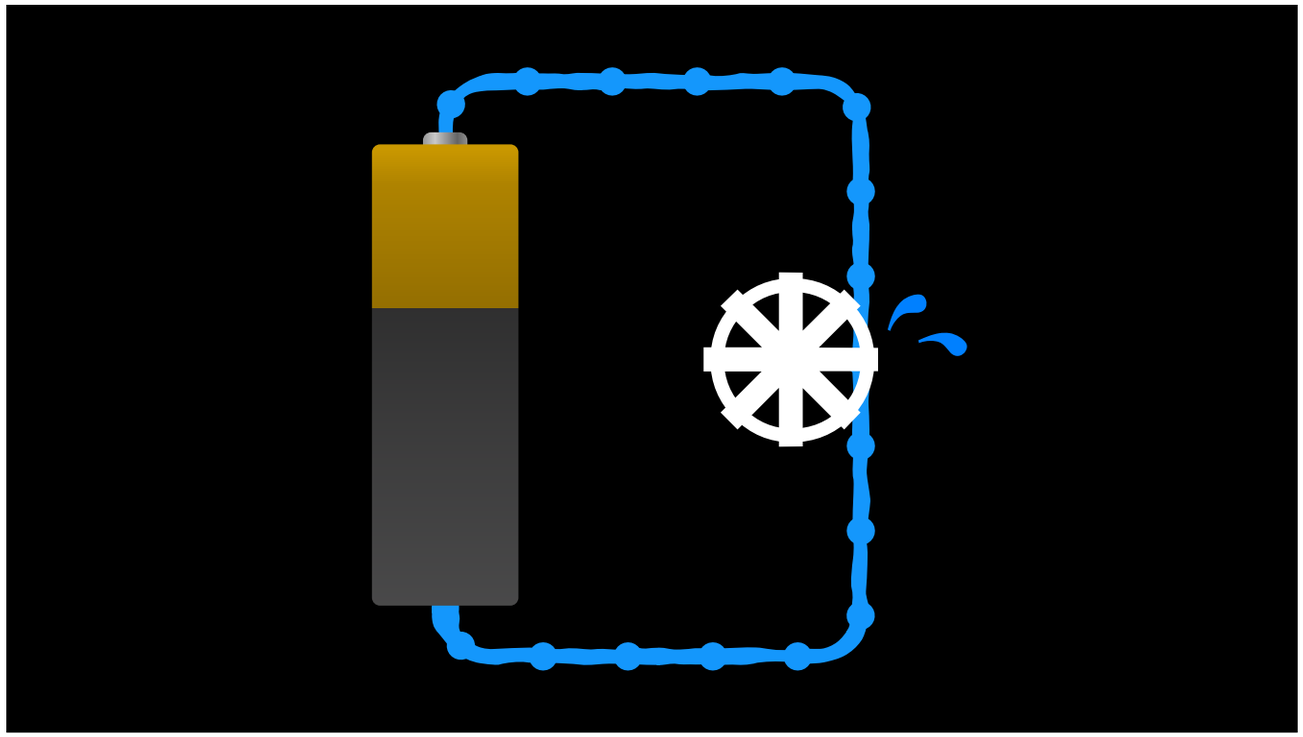
That compulsion to complete those chemical reactions is SO overpowering that electronics will up and EXIT one end of the battery, travel along the wire, and enter the other end to complete the reactions!

So strong is this compulsion, theoretically there's no limit to the length of the wire…there are practical limitations, sure…but theoretically at least, you could run that wire clear around the planet and this impulse is still present, that's what a force of nature this is.

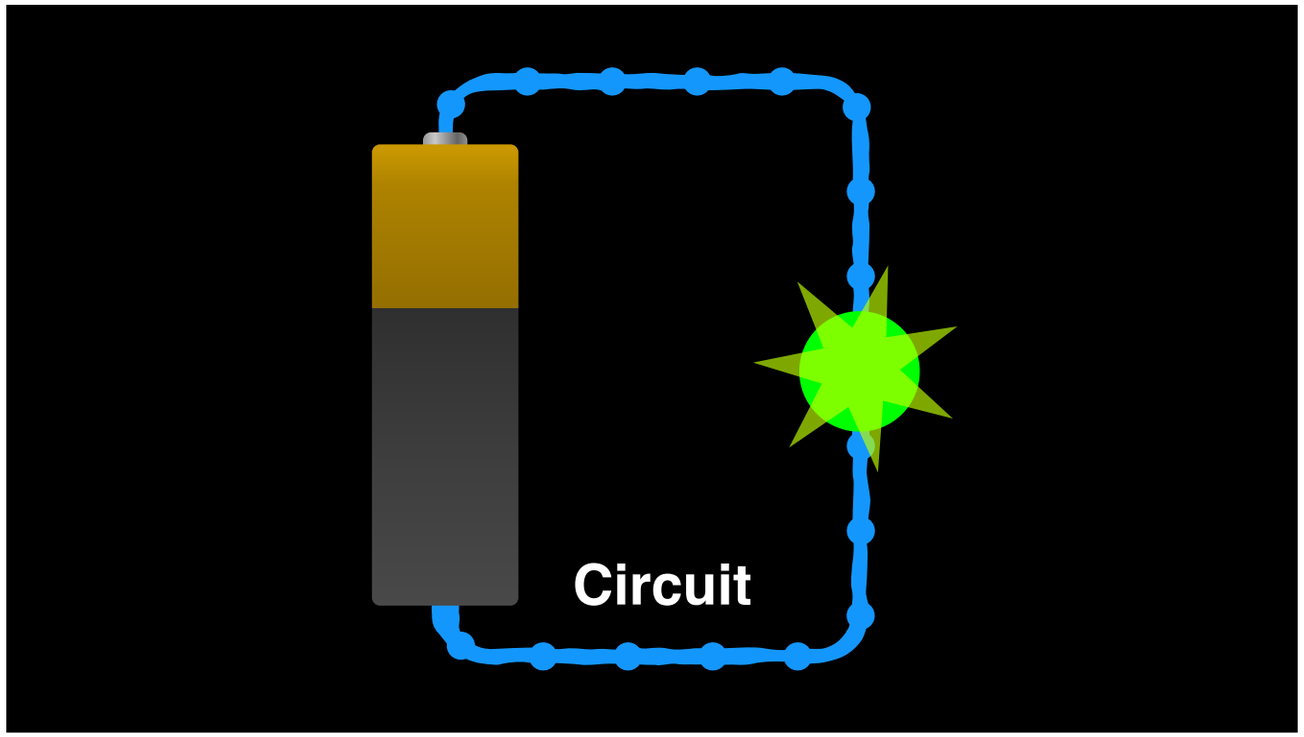


Sometimes, that flow of electrons through the wire is visualized like water flowing through a pipe or a hose…

And just like we can stick a wheel in a current of water to extract useful work from it…

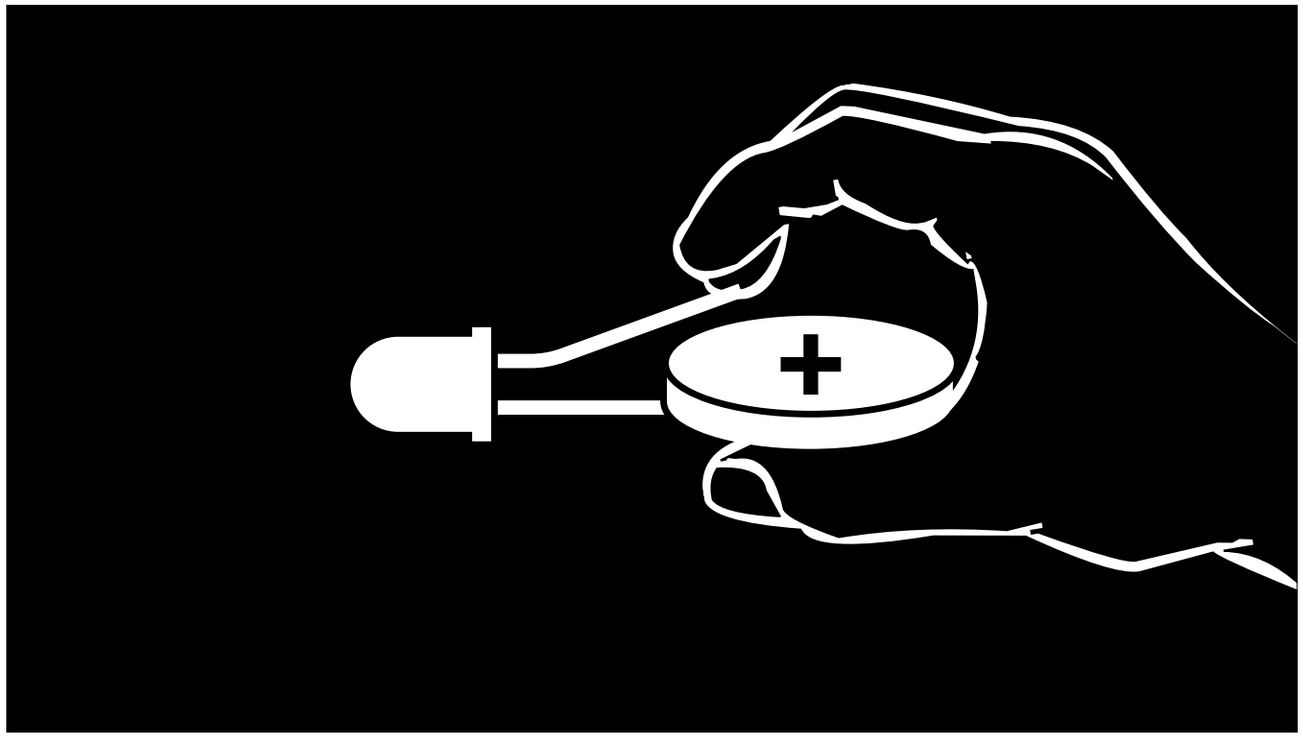


We can stick things in the flow of electrons to extract work too!

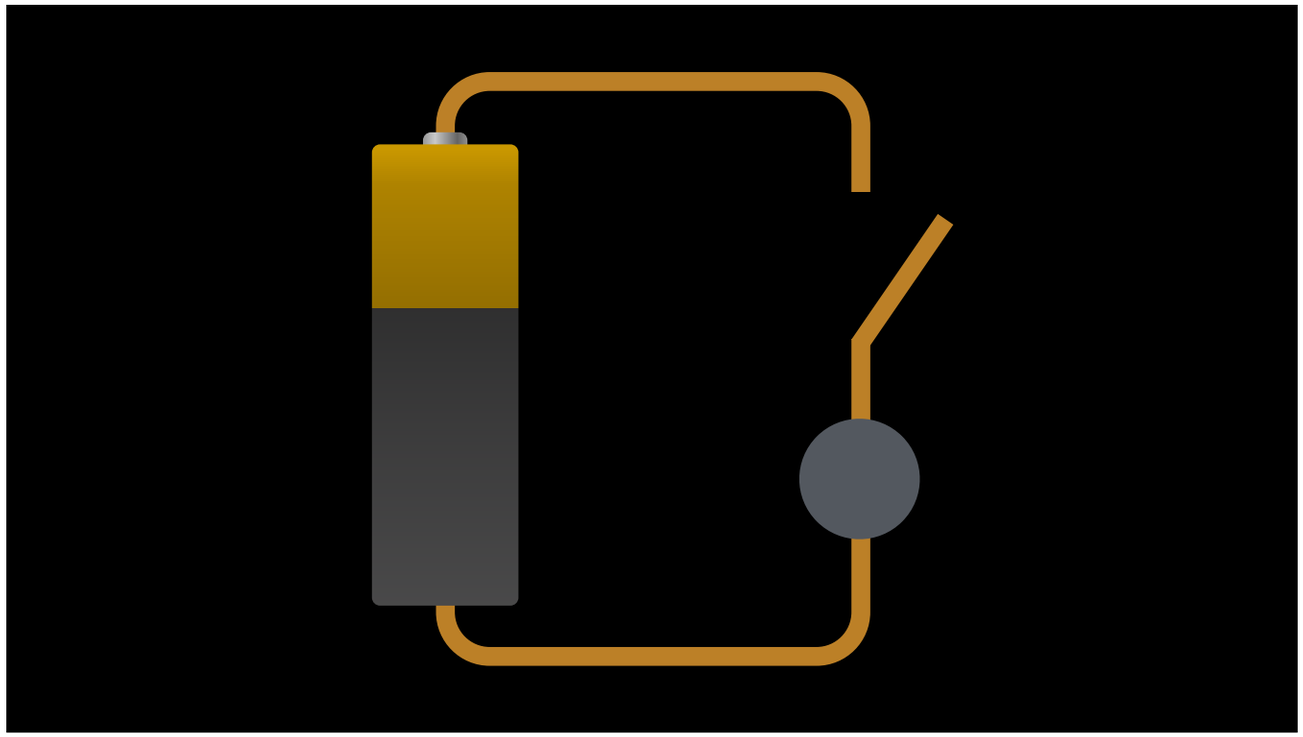


The LED is one of those things; it trades electrons for photons, providing light.

This is the basis of the electronic CIRCUIT. Most circuits are more complex than this, they may split off and do other things, but you'll always see them coming from and returning to a power source at each end.



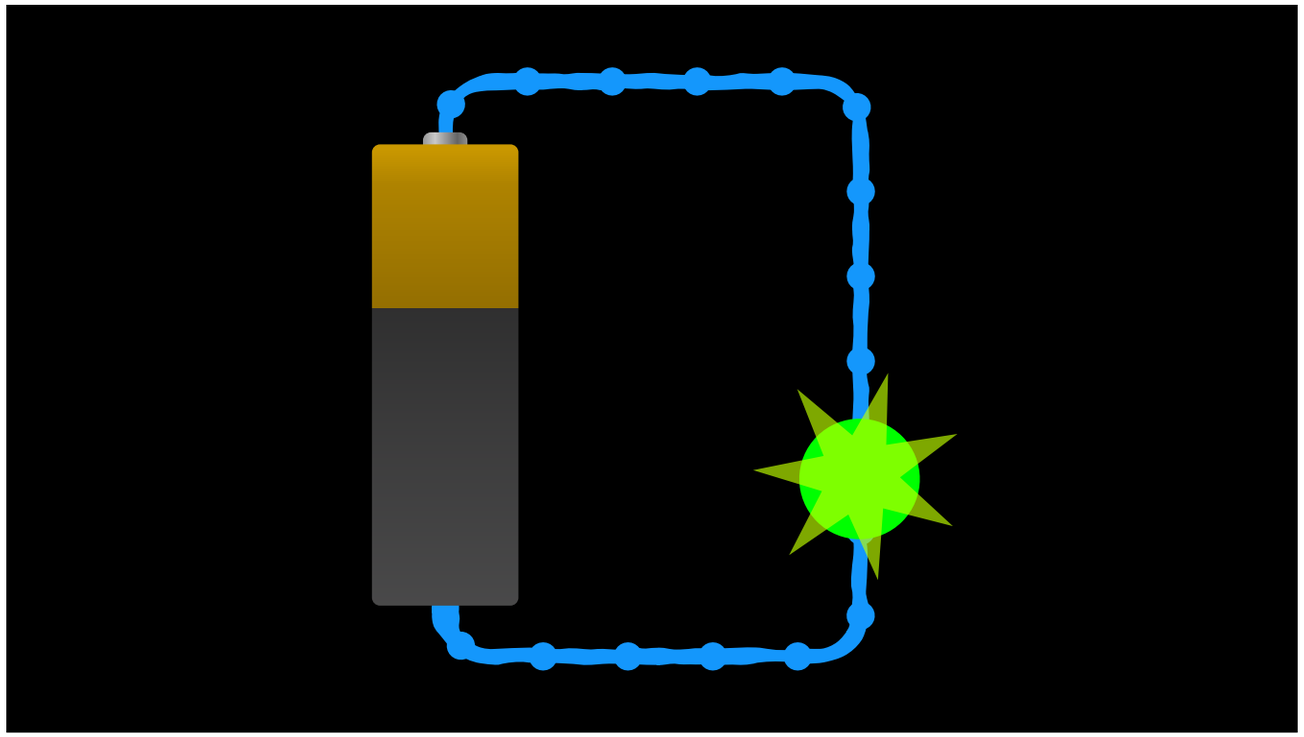
When we release the leg of the LED…



…the conductive path is broken, and current stops.

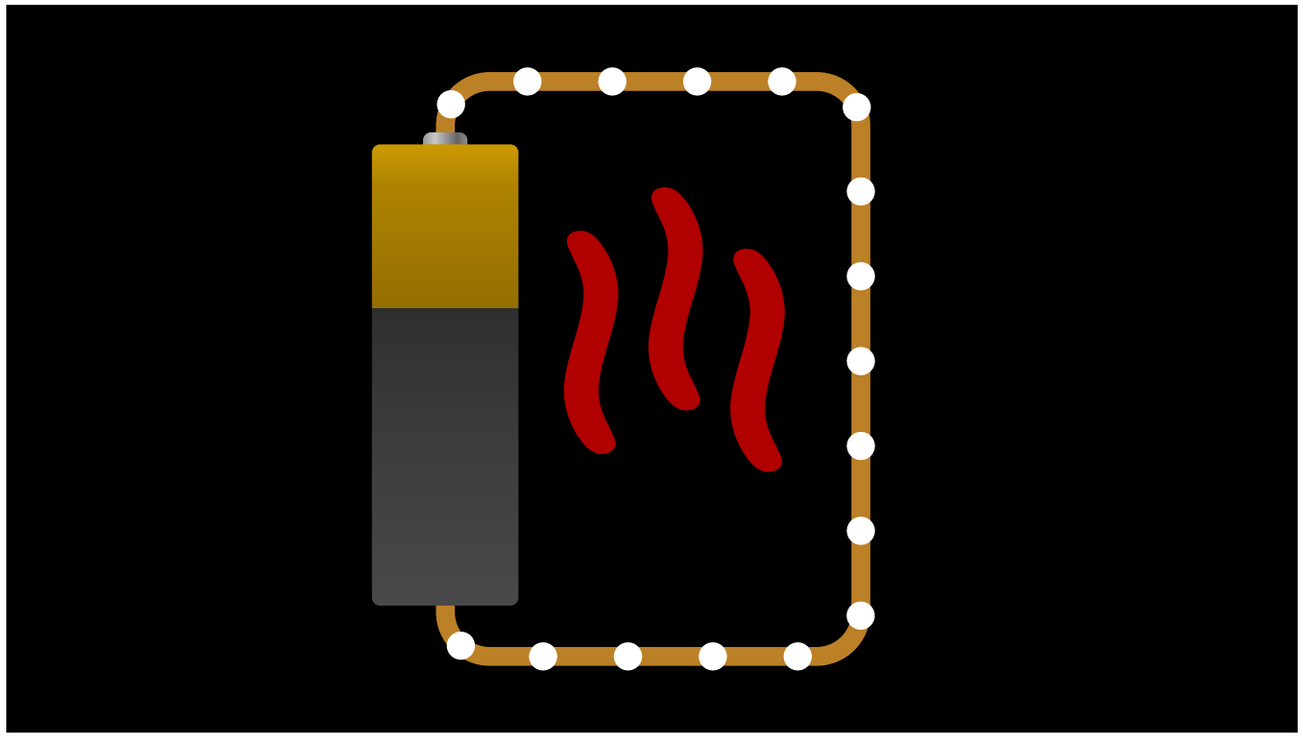
This is one way electricity ISN’T like water…break a pipe, water spills out. Electricity just stops.

This is called an OPEN circuit.



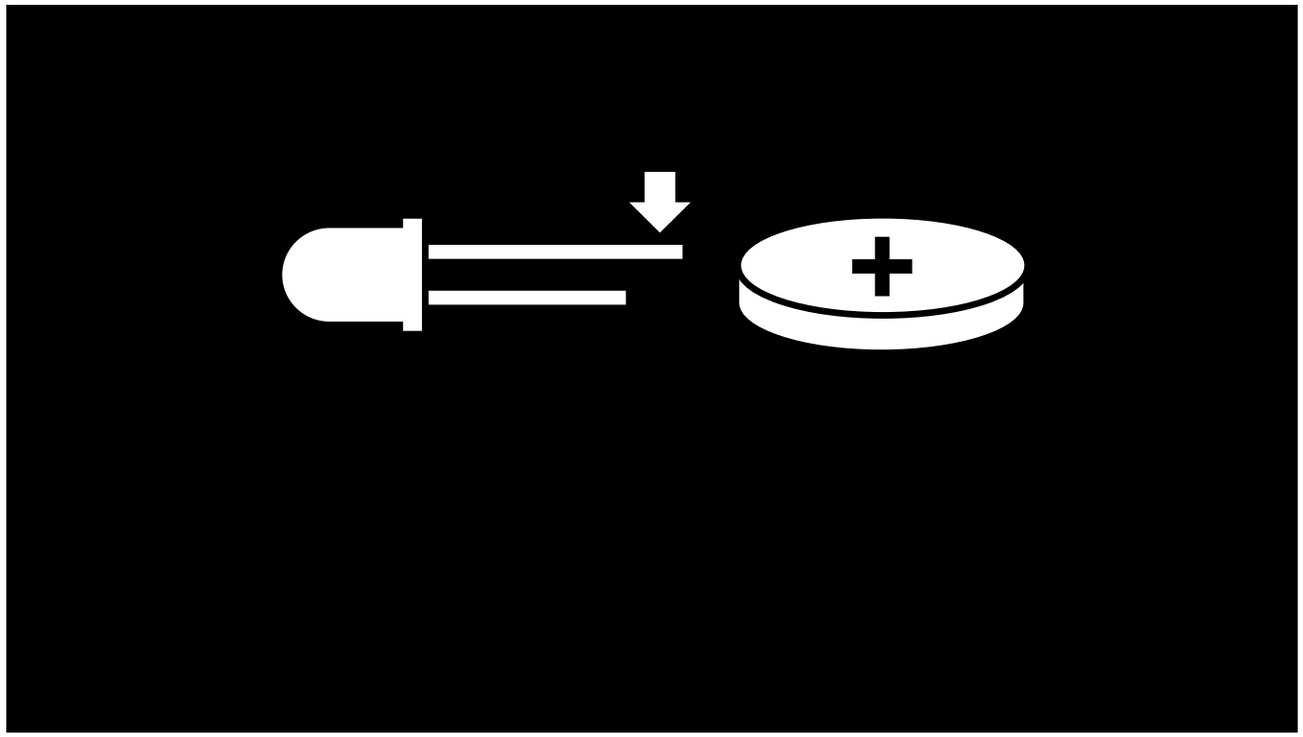
Pinch the LED and the conductive path is restored; the LED lights. This is a CLOSED circuit.

(Back and forth between this and prior slide. Open circuit. Closed circuit. Open circuit. Closed circuit.)

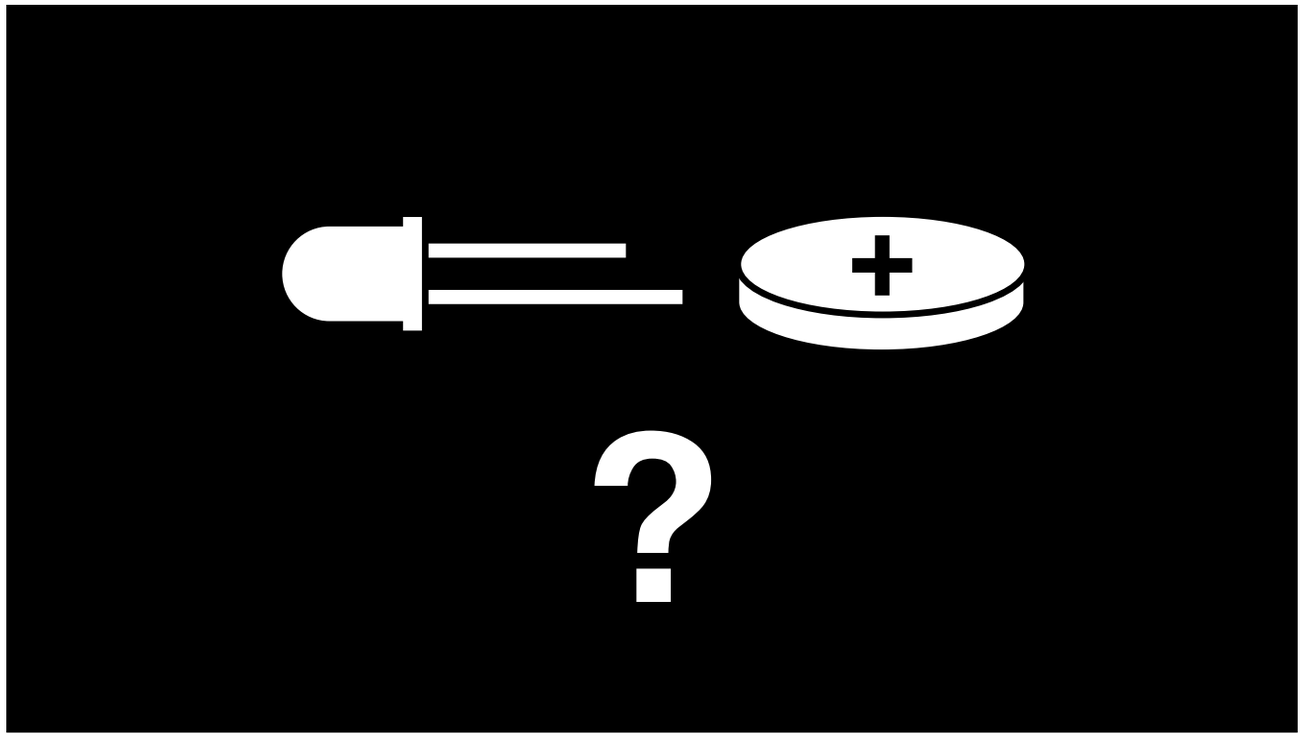


This initial picture I showed you, by the way…a circuit with NO components in its path…this is called a SHORT circuit.

These are rarely as pyrotechnic as Hollywood would have you believe, but in general with few exceptions, a short circuit is something you want to avoid.



So hey, remember how we connected the longer leg of the LED to the + side of the battery?



What happens if we try the other way? Go ahead, try it. What do you get?

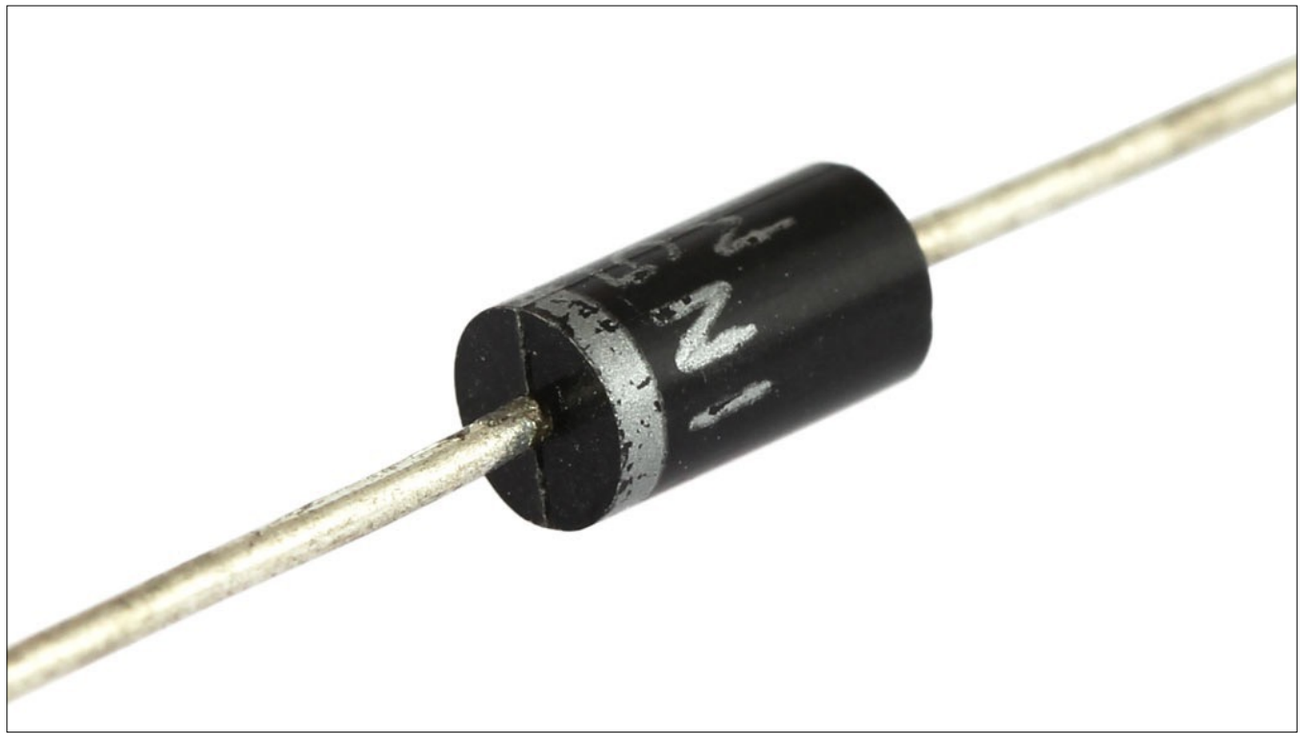
Nothing! Why is that?

LED

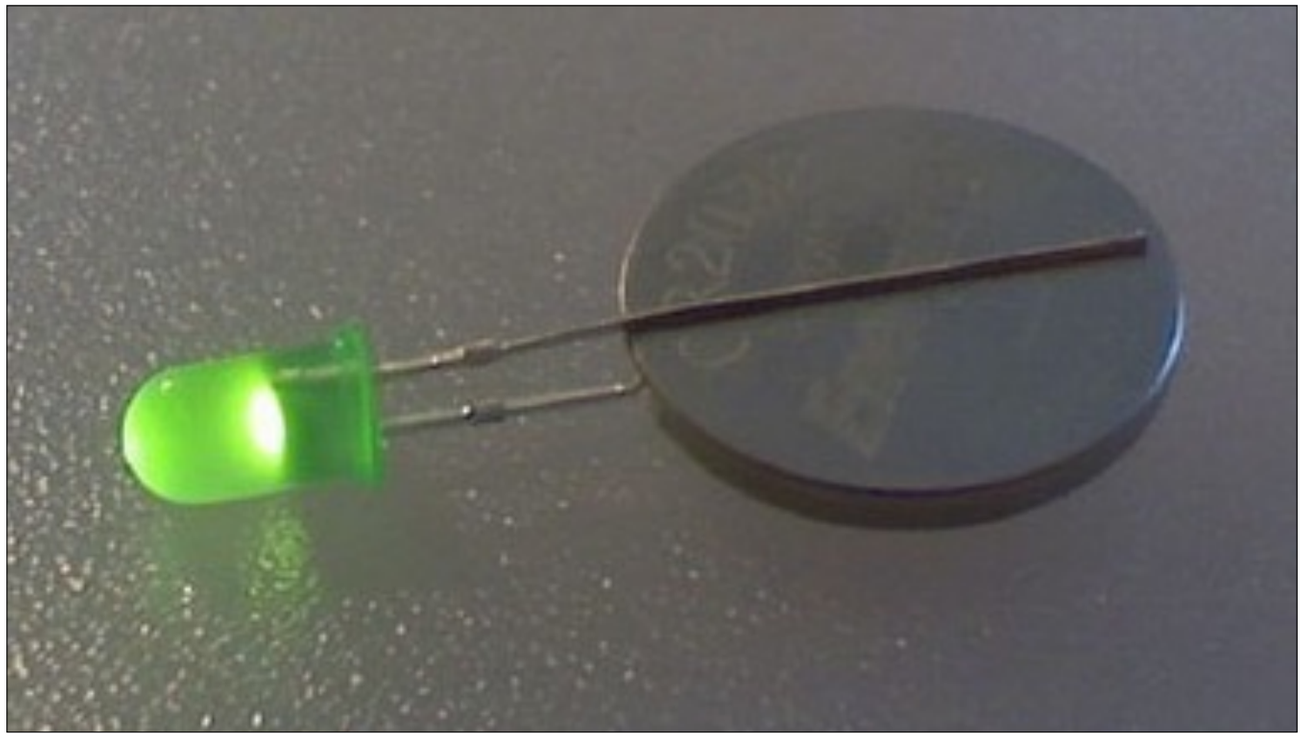
=

Light-Emitting Diode

This is because “LED” stands for Light Emitting DIODE…



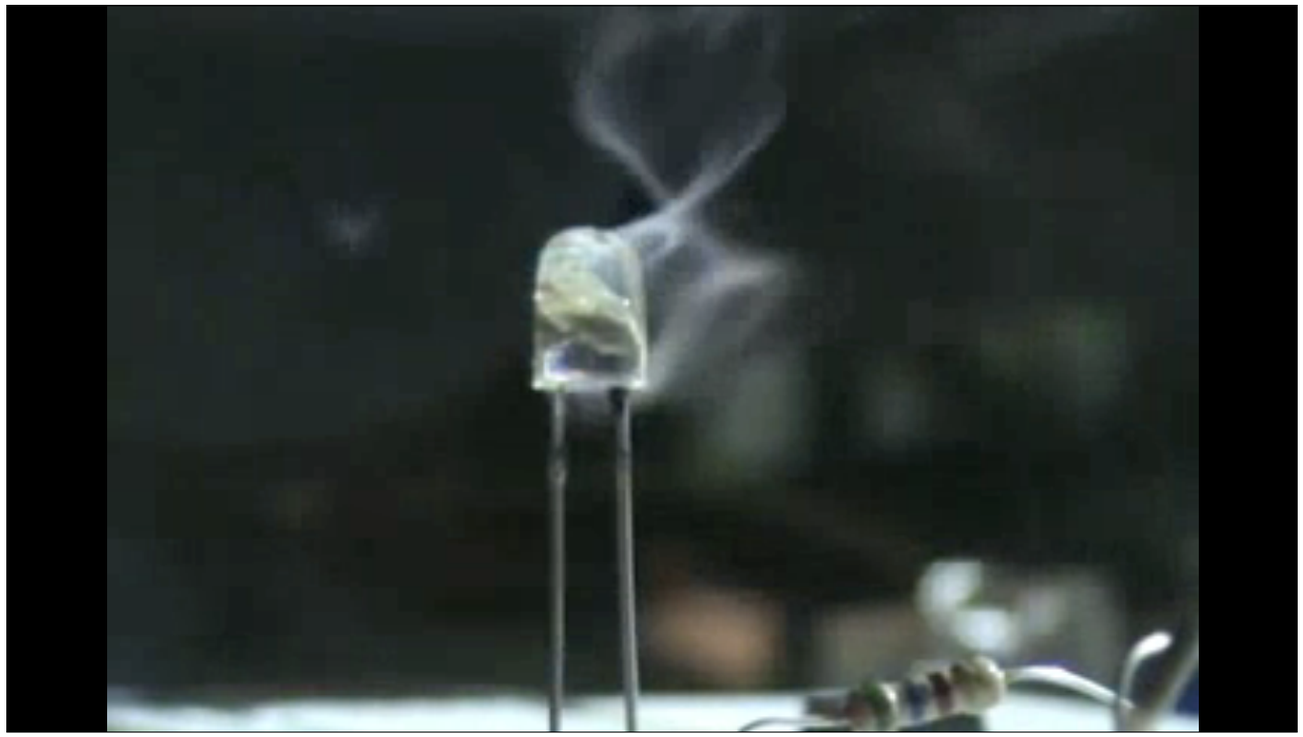
A diode is like a one-way valve for electrons; they can flow one way, but not the other. Not all diodes emit light…most are mundane-looking things like this one…but they provide a vital function in some circuits. This is just one of many electronic components you'll eventually learn about.



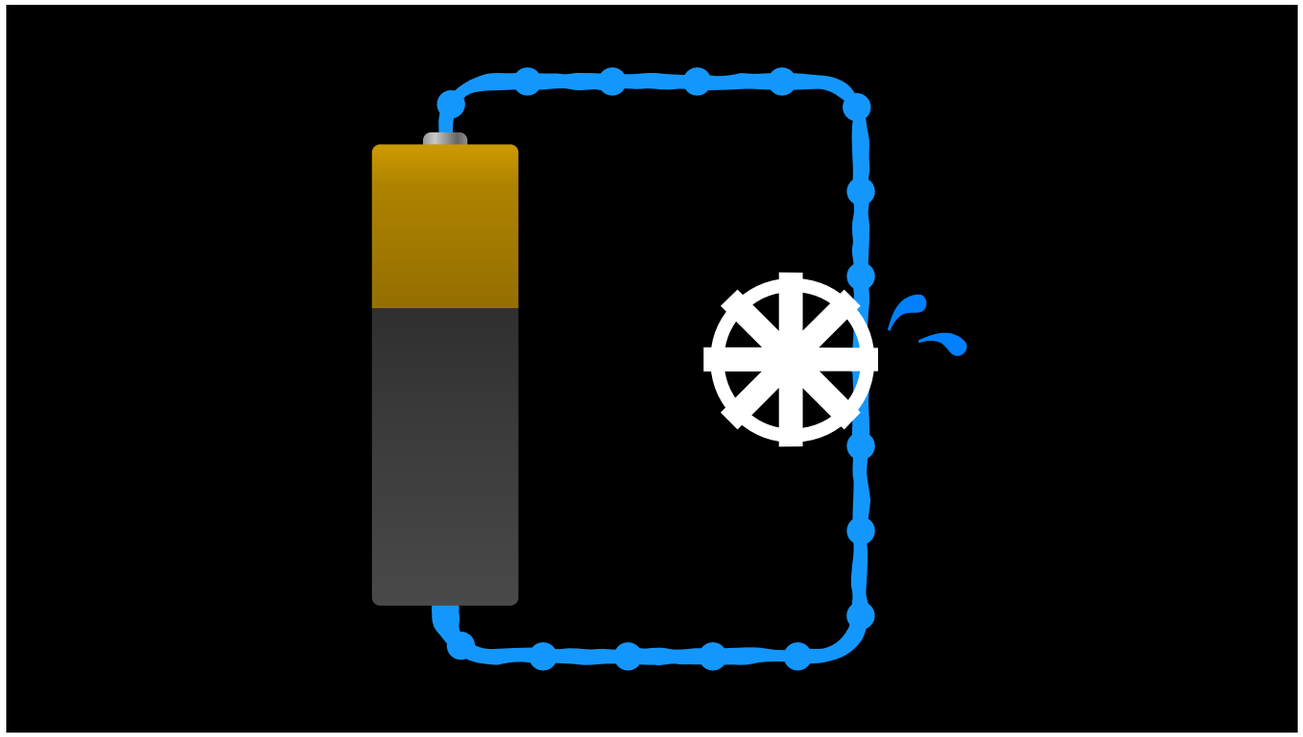
So, right there, you've made an actual working electronic circuit with just your hands! That's great. And some folks use them just like that…parts taped together, adding a few points of sparkle to a costume or prop. It'll run for a few hours like that. But it's not very economical. If you have lots of LEDs, and want longer run times before replacing batteries, we'll have to step up our game…

Look at this thing! It’s much bigger than our little coin cell battery. This is a 9 Volt battery, like you might find in a smoke detector. Something like this ought to run a whole slew of LEDs all day, shouldn’t it?

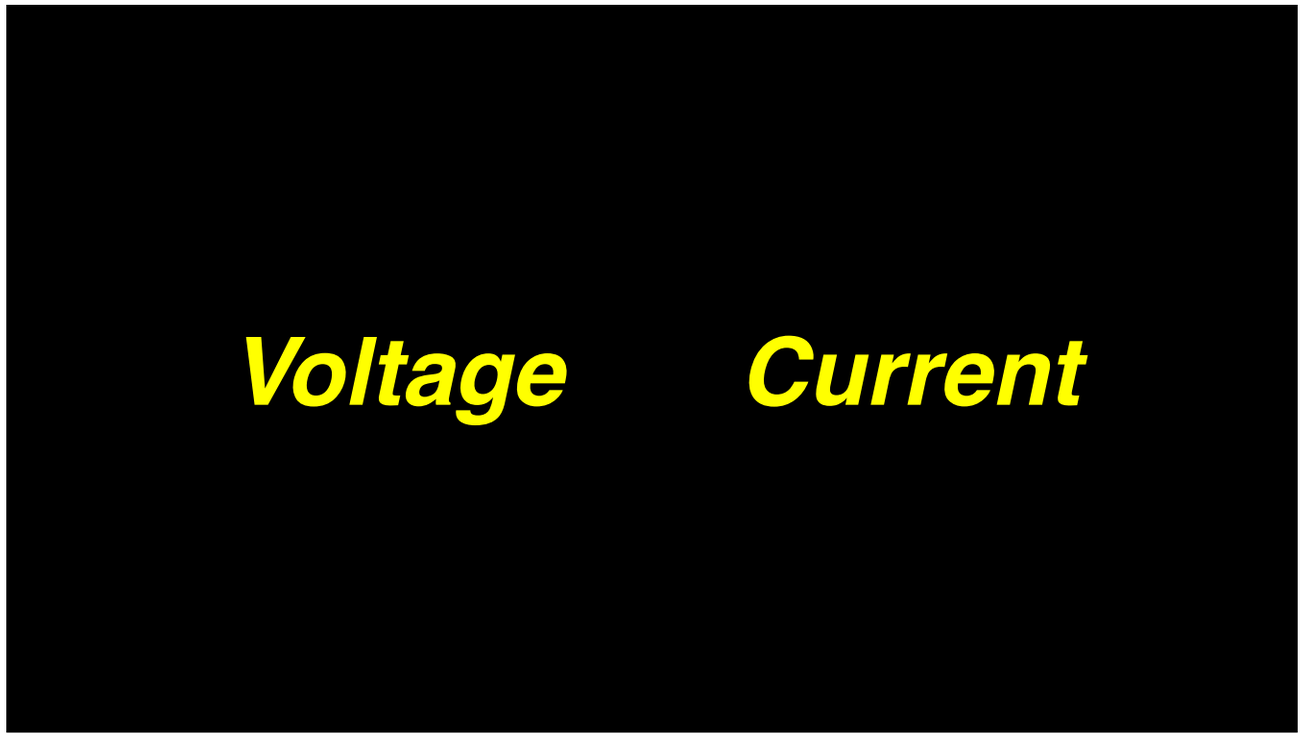
Which leads to our first classic beginner mistake…hooking an LED directly to a battery like this…



The LED fails almost instantly. Sometimes even dramatically!
But WHY? And how can we prevent this? To understand, we'll need to get a little more intimate with electricity…



So far, we've talked about electricity as either there or not; closed circuit or open circuit, on or off. But electricity has nuances…attributes…just like you and I are different heights or have different tolerances for spicy foods.



Two of these attributes are VOLTAGE and CURRENT.

Voltage

Electrical
Potential

VOLTAGE is a measure of ELECTRIC POTENTIAL. To use our water analogy; it’s the height of the waterfall. The further and harder it drops, the bigger wheel we can stick in there.

Voltage	***Current***
	
Electrical *Potential*	**Electron *Flow***

CURRENT is a measure of ELECTRON FLOW. It’s how fast the bathtub fills…regardless how high the tap is.

Voltage ***Current***

Volts
V

0V = “Ground”

Voltage is measured in units called VOLTS, abbreviated V.

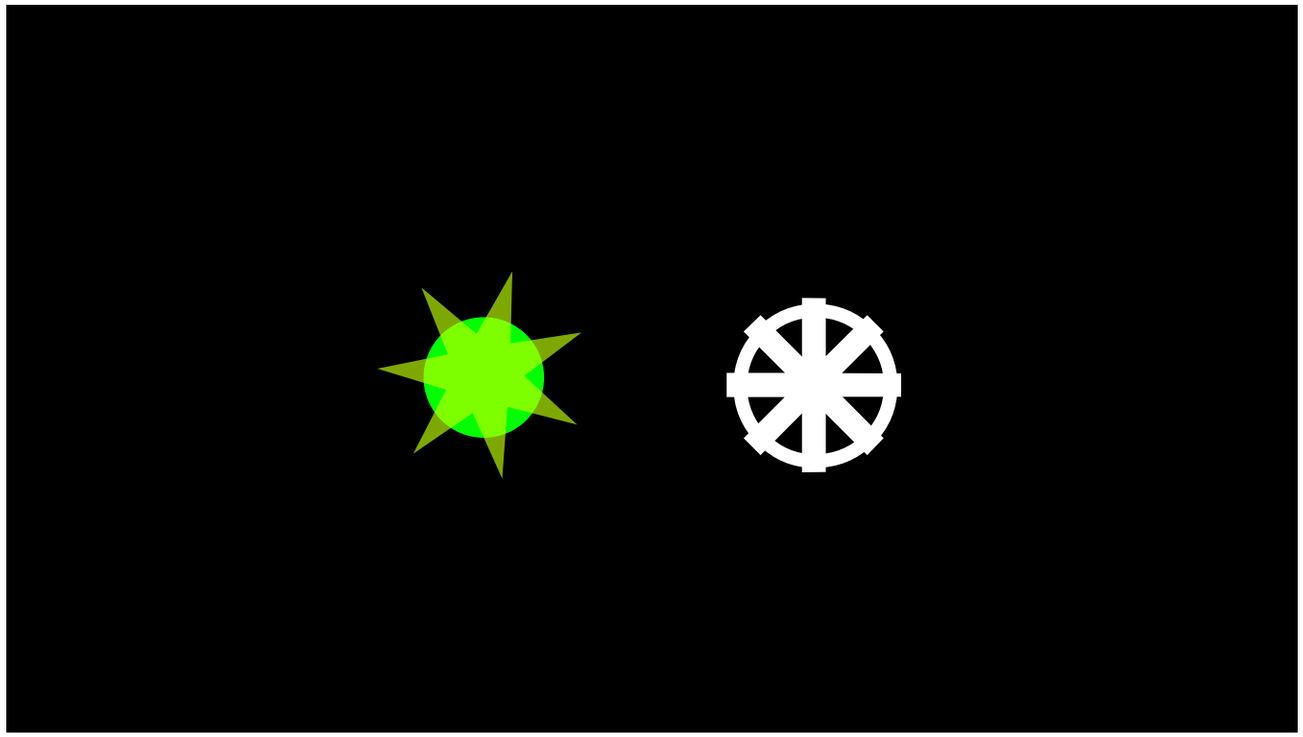
In a circuit, 0 Volts is sometimes called GROUND. Thinking of water again…if you pour out a glass of water, it may hit a few things on the way down, but it STOPS at the GROUND. That’s zero.

Voltage	***Current***
Volts	***Amperes***
V	***Amps***
0V = “Ground”	***1000 mA = 1A***

Current is measured in units called AMPERES or AMPS, abbreviated A.
One Amp is a lot of current, in small circuits usually we’ll use smaller units called MILLIAMPS; 1/1000 of an Amp.

Voltage	***Current***
	
Electrical ***Potential***	**Electron** ***Flow***

Voltage and current. Potential and flow.



So. These components we stick in the electrons’ path…the LED, for instance…these things introduce a new attribute…

Resistance

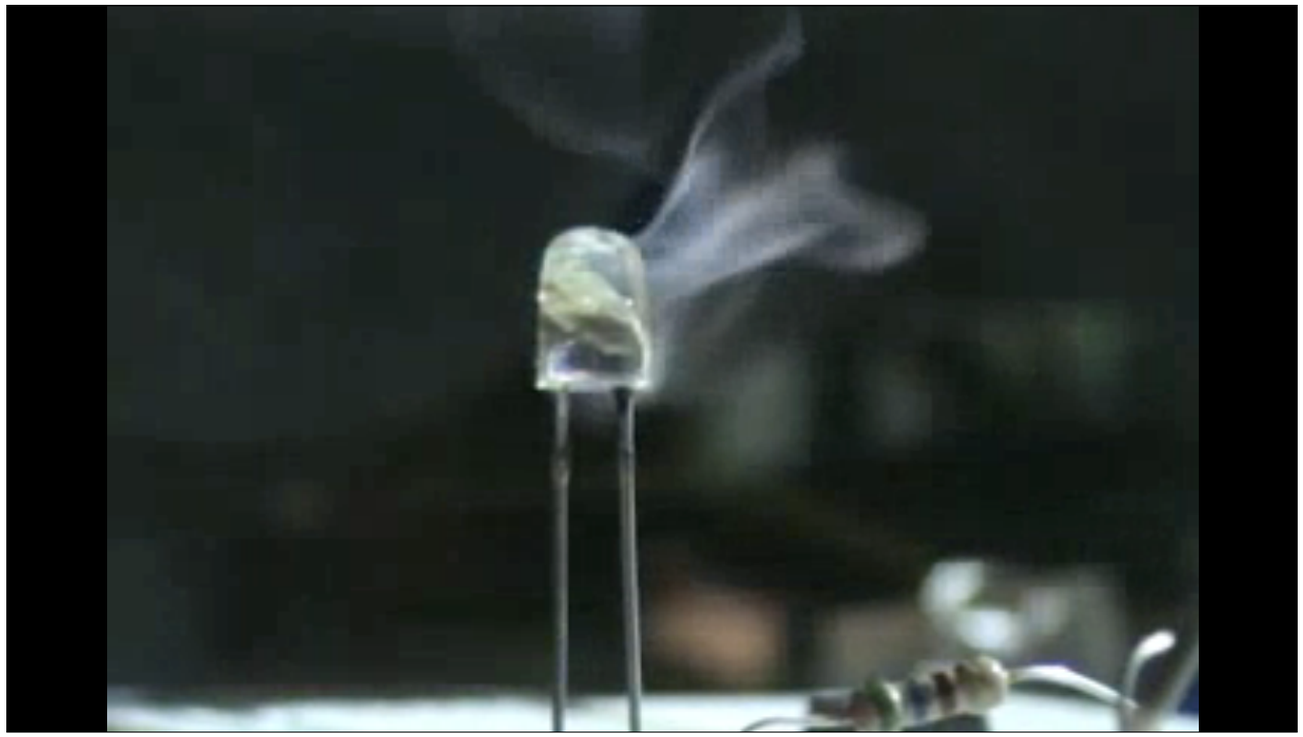
Restricts current
“Narrower pipe”

RESISTANCE. This works to oppose current; like a narrower pipe makes less water flow, or pinching a hose achieves the same. That’s resistance.

Resistance

Ohms
Ω

Resistance is measured in units called OHMS, abbreviated with this Omega symbol.

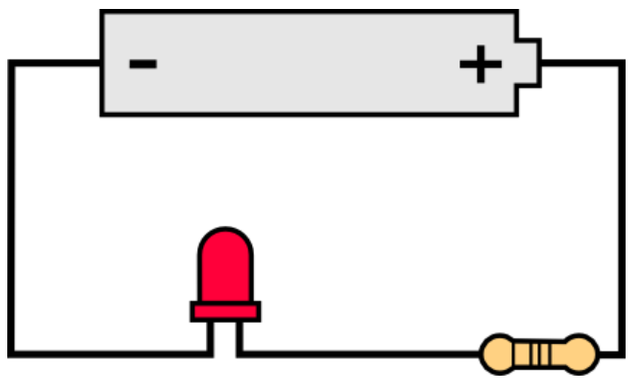


Why does the LED explode? Because of current. It’s a thermal problem…the tiny electrical junction inside just doesn’t have the capacity to pass a lot of current.

Ever had a dog get into a bunch of food and it just eats and eats until it gets sick? An LED is kind of like that…it has no concept of self-preservation.

We need to fit a narrower pipe onto the dog.

This is done with another electronic component…the RESISTOR.



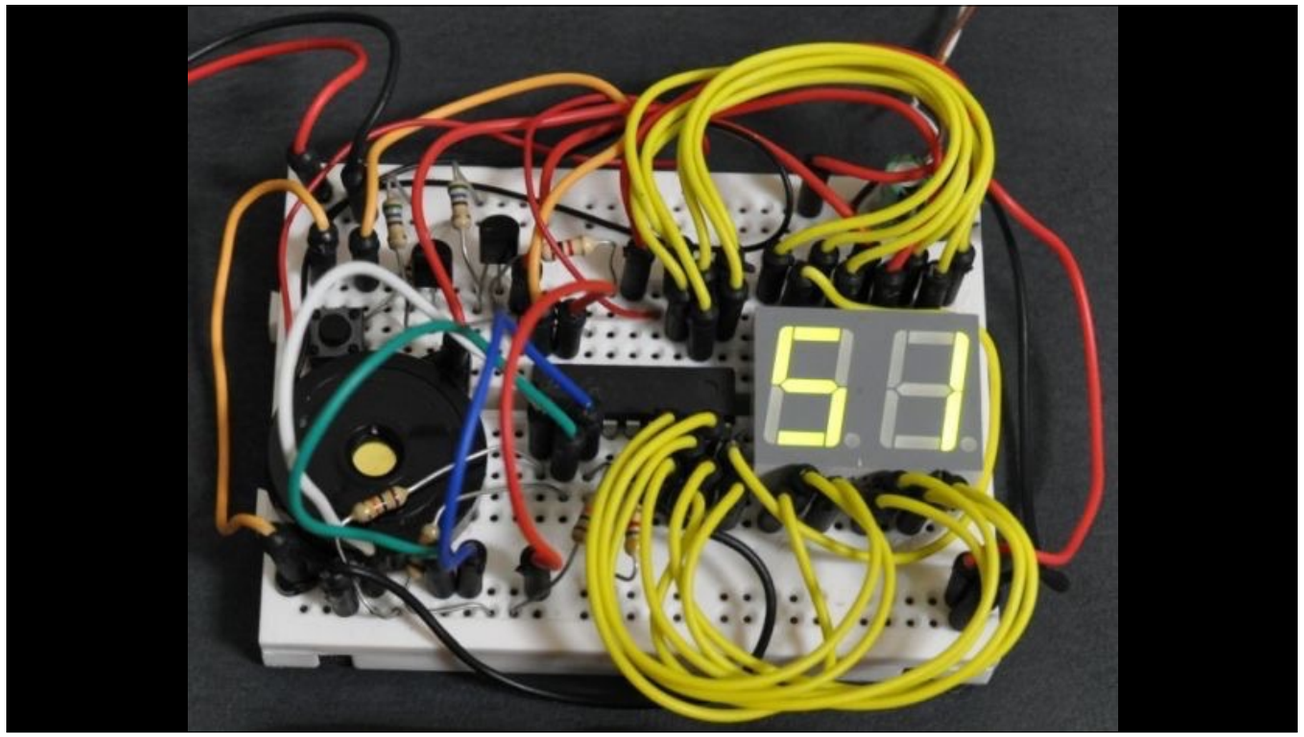
LED + Resistor

“In series”

Adding the resistor “in series” — that is, one component follows the other around the circuit — this throttles back the current and preserves the LED. BUT…

Resistors come in different values…remember, resistance is a measurement, in units called OHMS. We need just the RIGHT amount of resistance; not too much (or the LED will be dim), not too little (or the LED fails).

See these colored stripes? That's a code that tells us the resistance in Ohms. When resistors were first made, printing tiny numbers wasn't possible, so this system was invented and has stuck around. If you deal with resistors every day, you eventually learn to read the stripes…but if you just do electronics casually, it's okay to look it up. One of the cards I gave you has a reference chart, or there are smartphone apps to help decode.



I want to sidetrack for just a moment here though, to address another frequent beginner misconception…

Quite often you'll see circuits with all these colorful wires. It's a common mistake (and Hollywood reinforces the idea, "cut the green wire!") to think these different wires have different properties, just like the colored resistors. This is rarely true! Wires are usually color-coded for HUMAN convenience…they're the same inside, but the surface color gives a visual indication of their function. For example, it's a pretty standard convention to use red and black wires leading back to the + and – terminals of a battery. Setting the circuit aside and coming back to it later, there's an implicit documentation as to what's going on.

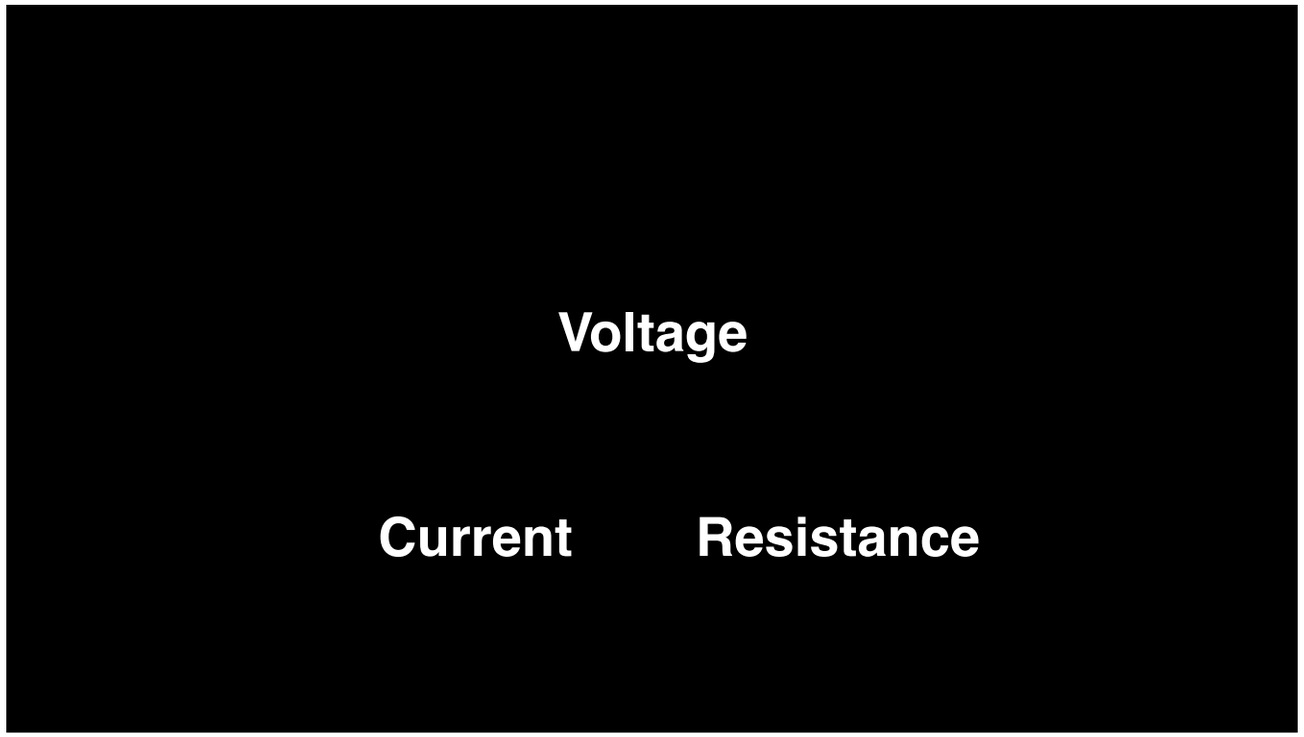
But I digress. We’re trying to find the right resistor to save our LED. Which one?

This is going to require just a little bit of math. Most people RESIST when they hear that, but trust me, it’s very small and simple…

First, recall that it’s CURRENT that destroys the LED.
A typical LED like this one…a 5mm “gumdrop” style…can withstand about 20 milliamps of current.

That’s a MAXIMUM though. We’d be redlining it continuously. To ensure a long life, let’s back off a little bit, maybe 10 percent…let’s aim for 18 milliamps, which is 0.018 Amps.

And here’s our battery, which is a 9 Volt type. We’ll need these numbers in a moment.



These three attributes I mentioned earlier…voltage, current and resistance…in any given circuit, there always exists a simple relationship between all of these…

Ohm’s Law

Voltage

Current **Resistance**

It’s a principle called OHM’S LAW, and what it tells us…

Ohm's Law

V

I **R**

…is that when we know ANY TWO of these values in a circuit, we can find the third, missing value.

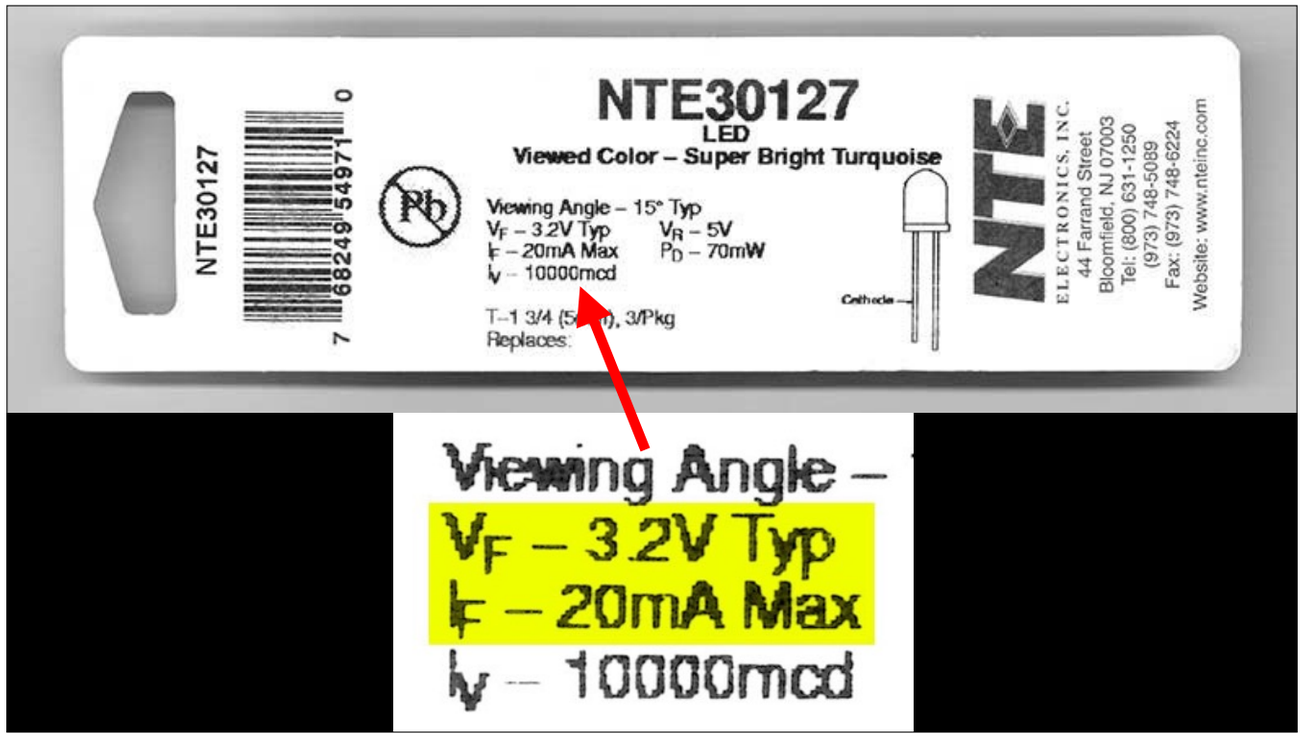
Ohm's Law

Voltage	$V = I \times R$
Current	$I = V \div R$
Resistance	$R = V \div I$

And it's a TRIVIAL calculation, just one multiply or divide to find the missing value.

Resistance $R = V \div I$

In this case, we're looking for a certain resistor. Resistance can be found by dividing voltage by current.
(Current is indicated by the letter 'I' in electronics; from French, intensite)



There’s ONE more number we need. Something called the FORWARD VOLTAGE of the LED. This is usually a function of the LED’s color, and can be found on the package. “VF” here — forward voltage — is 3.2V for this turquoise LED. And hey, there’s that 20 milliamp maximum rating again.

But we’re backing off a little to 18 milliamps…

$$R = V \div I$$

And resistance is calculated by dividing voltage by current…

$$R = \frac{9V - 3.2V}{0.018A}$$

We take the DIFFERENCE between the BATTERY VOLTAGE and the LED FORWARD VOLTAGE, and divide that by the current in Amps…

$$R = 322.22\ \Omega$$

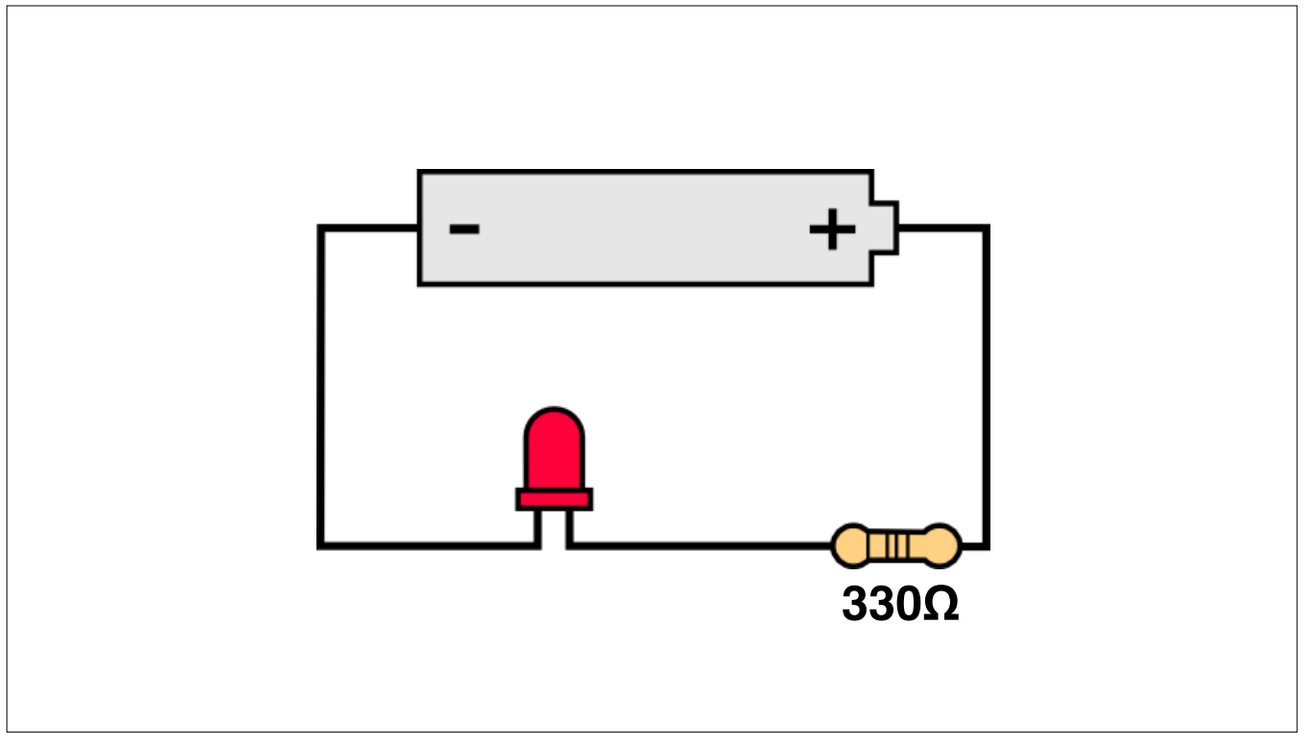
And there’s our desired resistance, in Ohms.
322.22 Ohms. That’s it.

BUT…

Resistors only come in certain standard values. You won’t actually find a 322.22 Ohm resistor!

R = 330 Ω

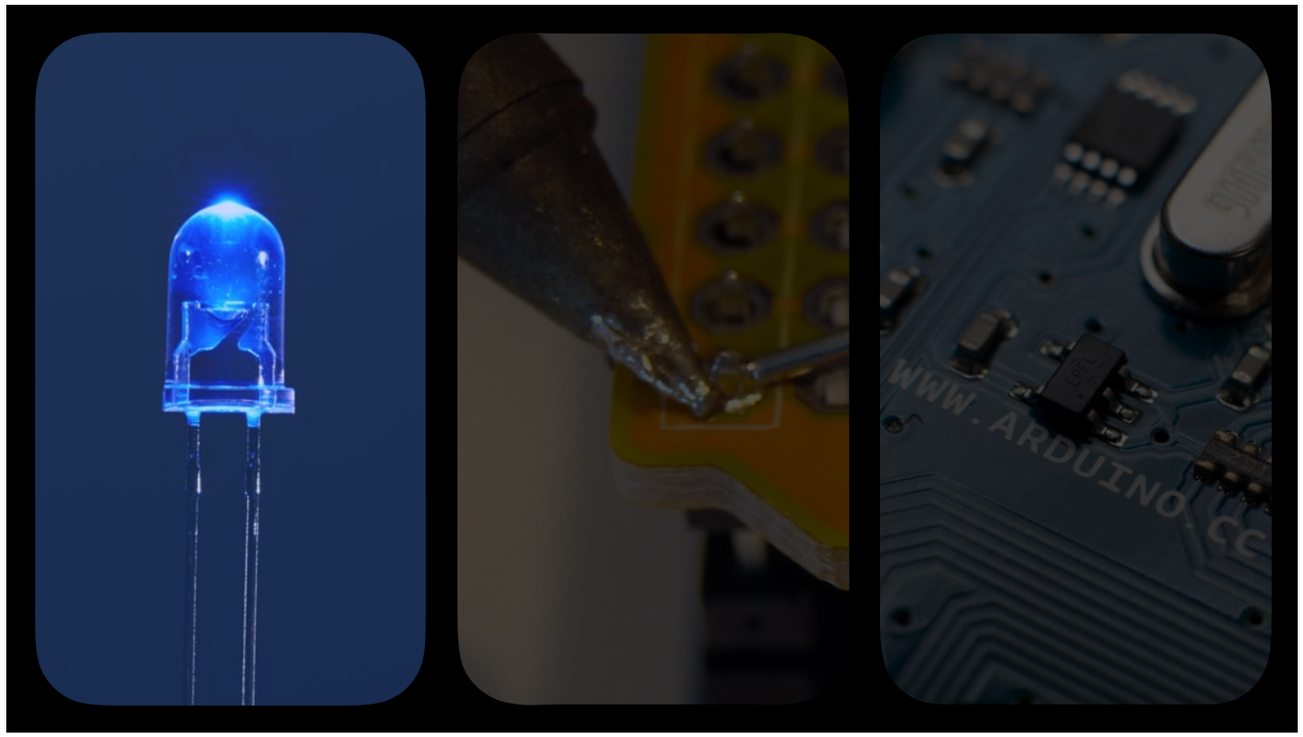
What you do in the case of these LED circuits is to round UP to the next standard value. In this case, that would be 330 Ohms.



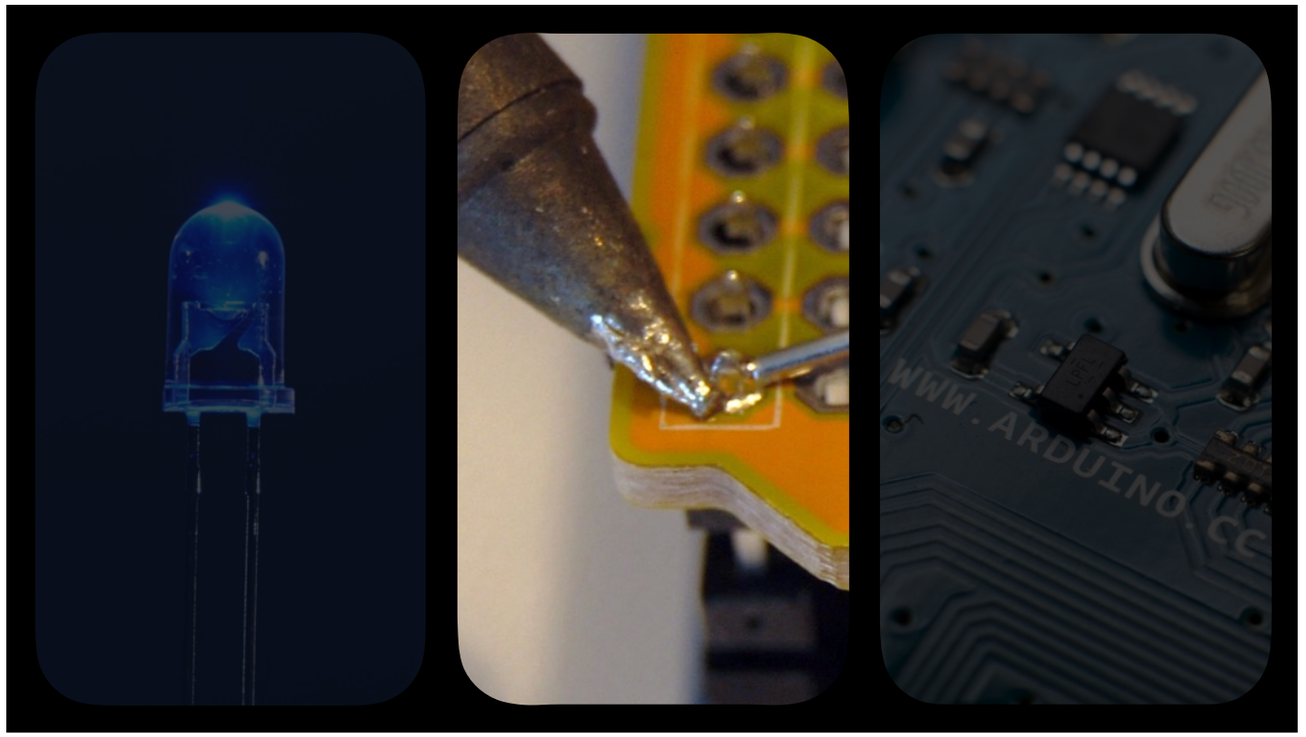
Add a 330 Ohm resistor “in series,” and the LED is now saved!

It’s important to note that this value works for that specific battery and LED combination. Using a different battery, or a different type of LED, you’ll need to do that calculation to find the right resistor for the situation.

There are smartphone apps and web sites with LED resistor calculators to help with this. They're a great time-saver. But I really REALLY recommend, every few LEDs, doing the math "the long way" and then checking against the app. This helps you internalize what the concepts of voltage, current and resistance are all about. Ohm's Law is the very bedrock of all electronics…the sooner you grasp this, the faster and better you'll get at designing more complex circuits.



That’s a basic introduction to circuits, LEDs and Ohm’s Law. Thanks for bearing with me through all that, but it’s vital, absolutely foundational stuff! Any questions on this part of the talk?



Second, I’d like to talk about SOLDERING. Soldering is THE assembly medium of electronics.

As you know with costuming, staples and duct tape will only get you so far…inevitably you’ll be learning to sew, whether by hand or machine.

The situation is similar with electronics…soldering is a vital skill you’ll NEED to acquire. But misconceptions abound for beginners, so I’d like to get into this now.

The tool of soldering is the SOLDERING IRON. You plug it in, the tip gets hot enough to melt certain metals…solder.

You WILL be needing one of these. And as a beginner, you might be looking to economize…

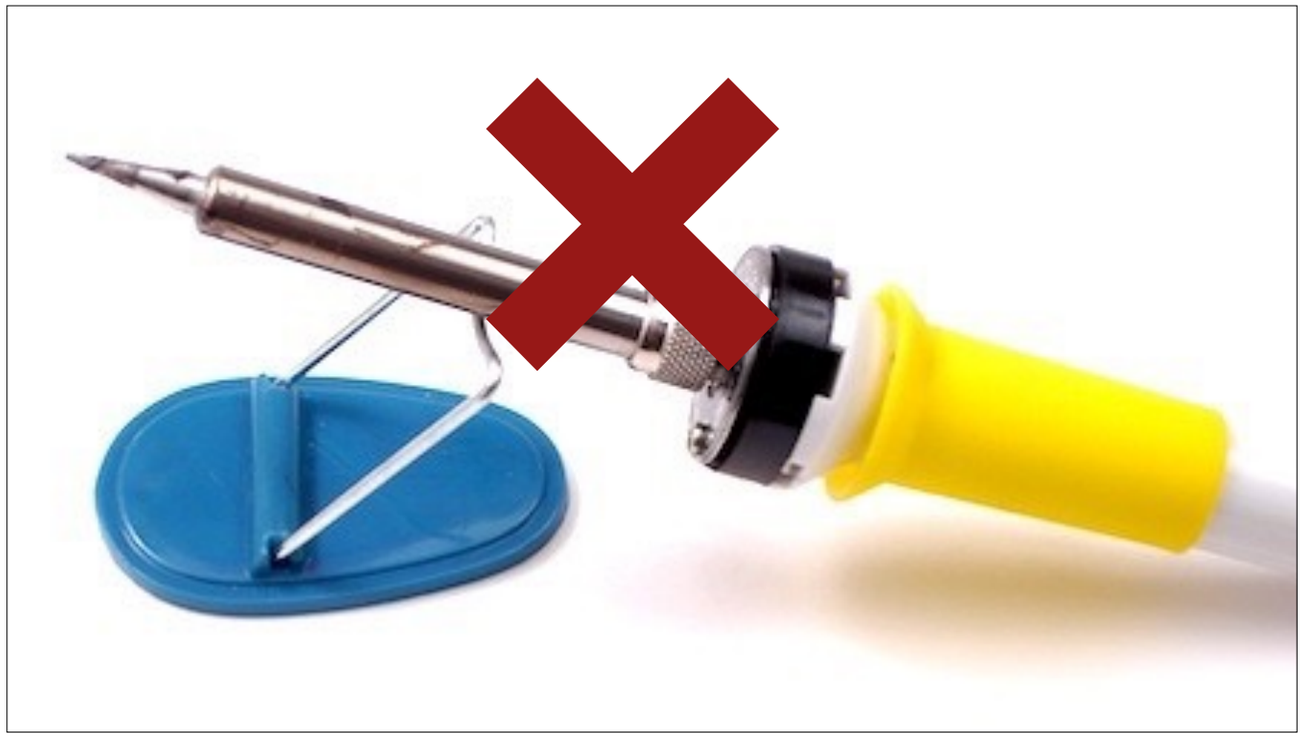
…but if your grandfather or a weird uncle offers a hand-me-down soldering gun like this, politely decline. These are for really large jobs and electronics from another era. It’s a bazooka.

And something called a “Cold Heat” soldering iron is just so much snake oil and is not suitable for electronics; avoid!

And For The Love Of God And All That Is Holy, do not, DO NOT buy the cheapest, piece-of-crap soldering iron you can find!

That’s not just advice for electronics, that’s advice for life. Never buy a bottom-of-the-barrel tool for ANY job. You’re just making it harder on yourself in the long run.

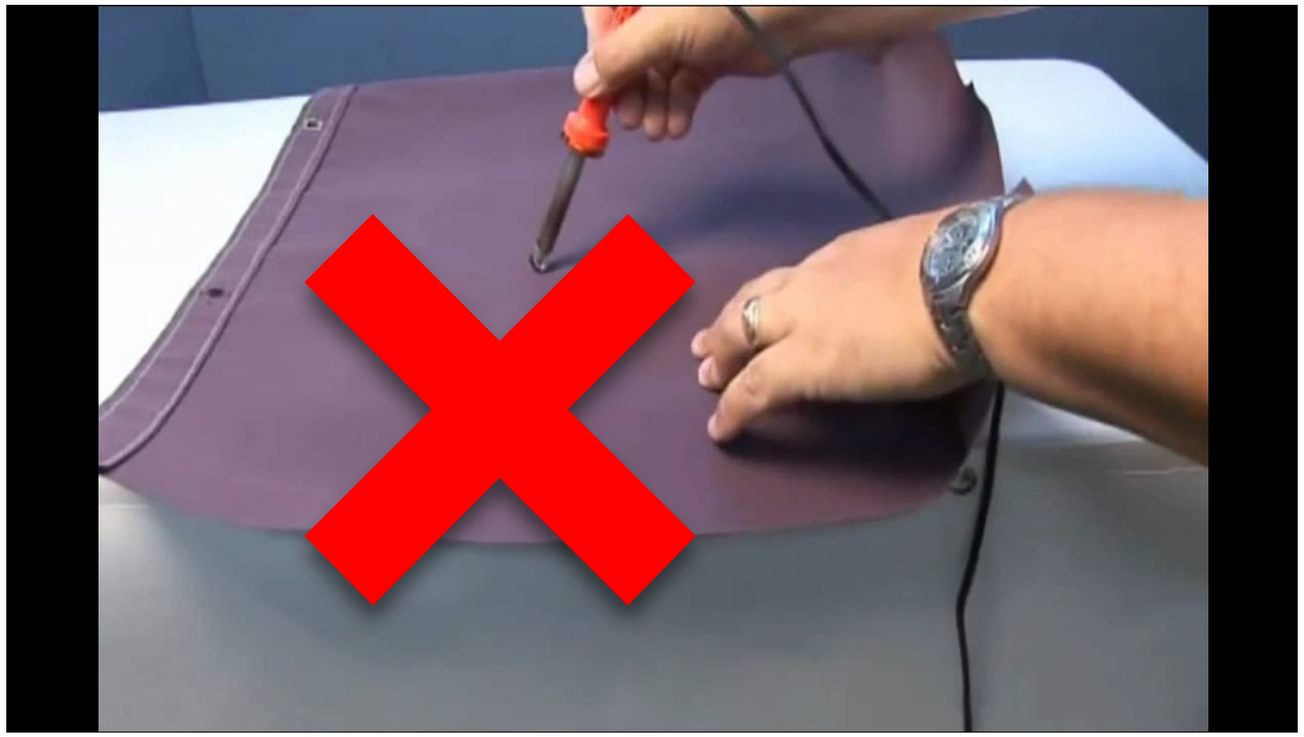
This doesn’t mean you have to break the bank. This Hakko soldering iron is amazing, but will set you back about a hundred bucks! If you’re just starting out, a simple but well-made iron might cost 20 to 40 dollars and can last for years.



If you get an economy iron and it comes with one of these little flip-out stands, THROW IT AWAY IMMEDIATELY. These are dangerous and should be illegal.

The tip of a soldering iron is about twice the temperature of a hot glue gun, and focused in a much smaller area. A stand like this is an accident waiting to happen. So…

If your iron doesn’t include a proper stand like this, GET ONE. They only cost a few dollars extra and are completely, totally worth it.



Your soldering iron, by the way, is for SOLDERING, period. Not for poking holes in things, or cauterizing the cut ends of nylon webbing, or any other non-soldering tasks.

It’s like fabric scissors in that regard. Fabric scissors have only two jobs: cutting fabric, and cutting people who use them on paper.

60/40 Rosin Core Solder

The next item you need is solder. This is what joins components together.

Look for something labeled “60/40 rosin core” solder. “Rosin core” is very important — do not use “acid core” or anything else, that’s typically for plumbing. Rosin is basically tree sap…it’s what’s known as a “flux,” which helps molten solder flow more smoothly between parts.

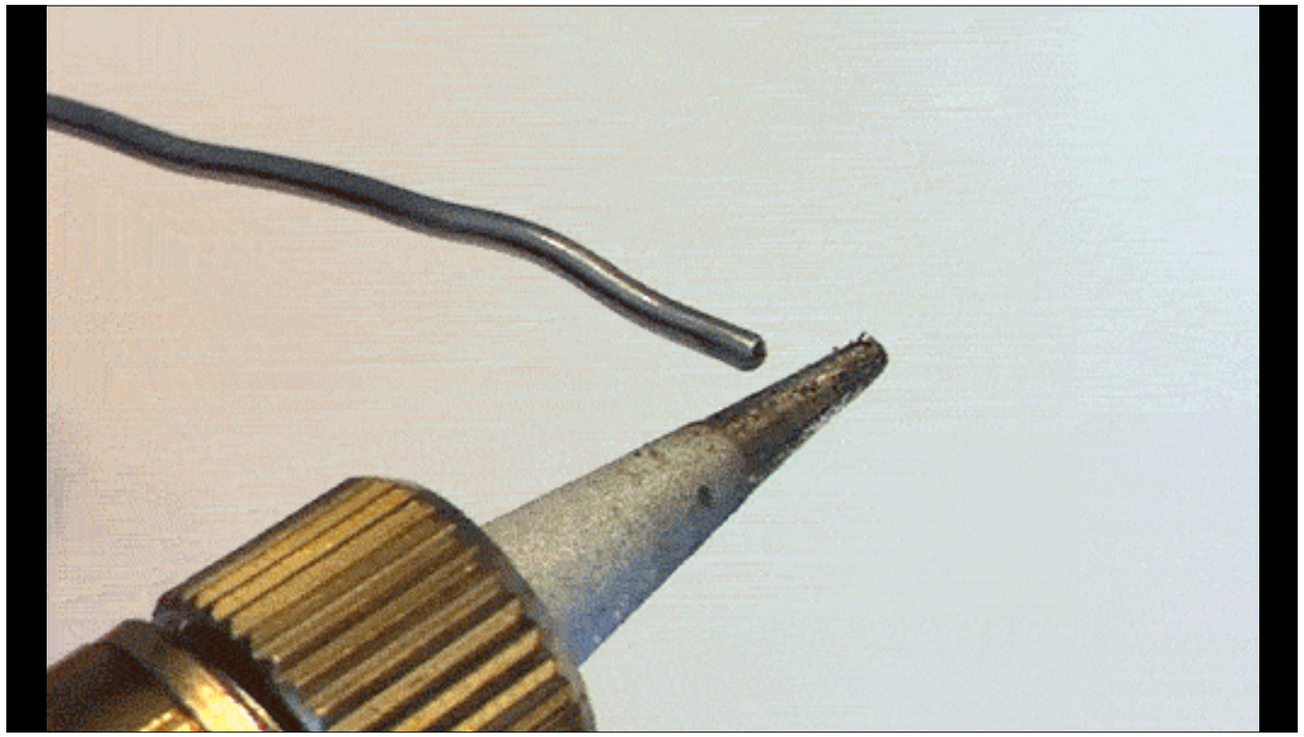
60/40 refers to the metal alloy: 60 percent tin, 40 percent lead. The mention of lead is understandably concerning…lead can be toxic…but you can take some simple precautions to work with this safely.

First: if you have children or pets, store the solder well out of reach when not in use. Treat it the same way you would any nasty household cleaners, for example.

Second: when you’re finished soldering, go wash your hands thoroughly. Do this in a bathroom or a garage sink if you have one, NOT in a kitchen sink full of dishes!

Follow those precautions and you’ll be fine.

If anyone here is visiting from Europe, you can’t get leaded solder; only lead-free can be sold…it’s a different alloy, mostly tin with some silver. Lead-free solder is a bit harder to work with and requires a slightly higher temperature. You can get this in the United States too but I don’t recommend it for hobby use.

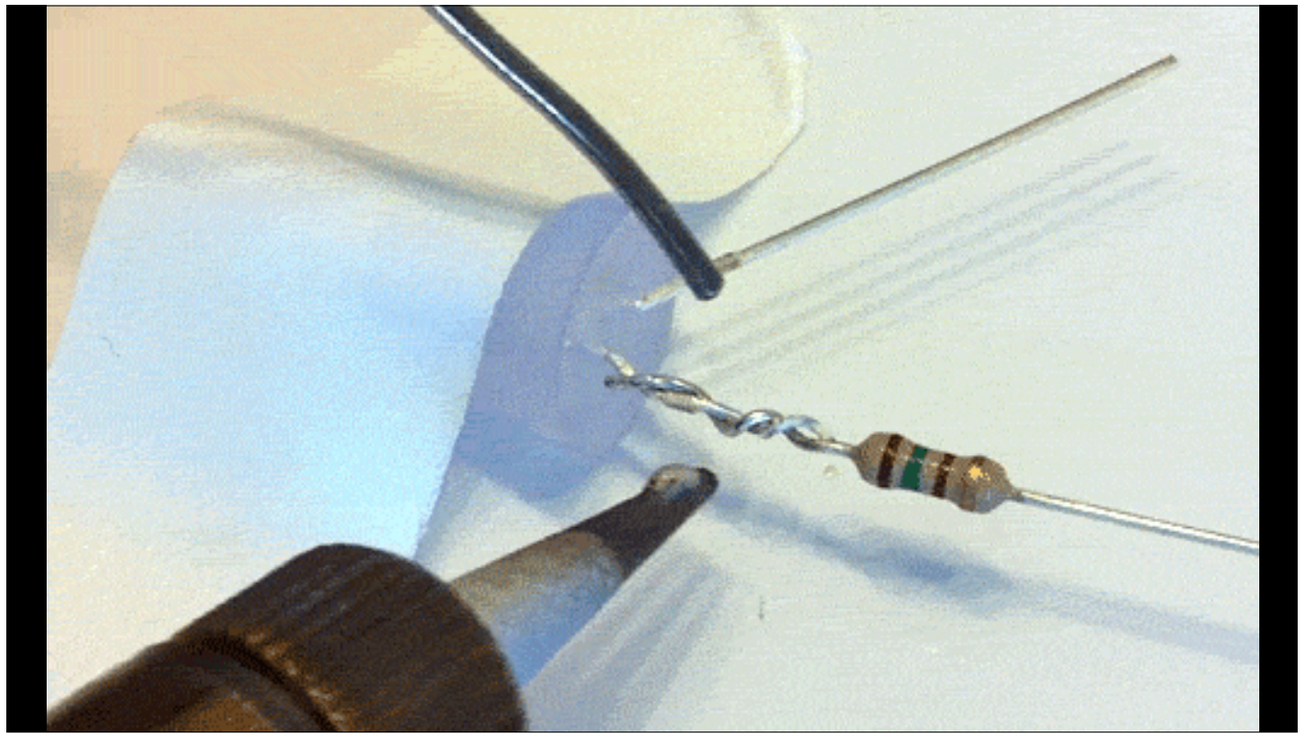


So! Time to solder. I WISH we could all do this hands-on here, but the (hotel/convention center) won't allow that. Not to worry, I can demonstrate pretty well with video…

You switch on your iron and give it a few minutes to reach full temperature. While you're waiting…see this sponge in the base? Go in the bathroom (NOT kitchen), wet the sponge and wring it out. Some bases have a brass sponge, like a Brillo pad; no water needed.

Once the iron's hot, wipe the tip on this sponge to remove any charred old gunk on there.

Then melt just a small drop of solder on the tip, like this. It's called "tinning" the iron, and is one of the very few times you melt solder directly on the tip. A VERY common misconception among beginners it treating the iron like a hot glue gun, melting solder and wiping it on parts. But good soldering is really about heat TRANSFER. That little drop of solder helps move heat from the tip to the parts being soldered. The parts heat up, then we add more solder to THAT. Let me show you…



This is an example of component-to-component soldering, or component-to-wire.

Notice the iron comes in, the tinned tip contacts the parts and they're heated up for just a moment, then more solder is added to the connection. Allow a second or two for the solder to flow — capillary action pulls it into every nook — then pull the iron away.

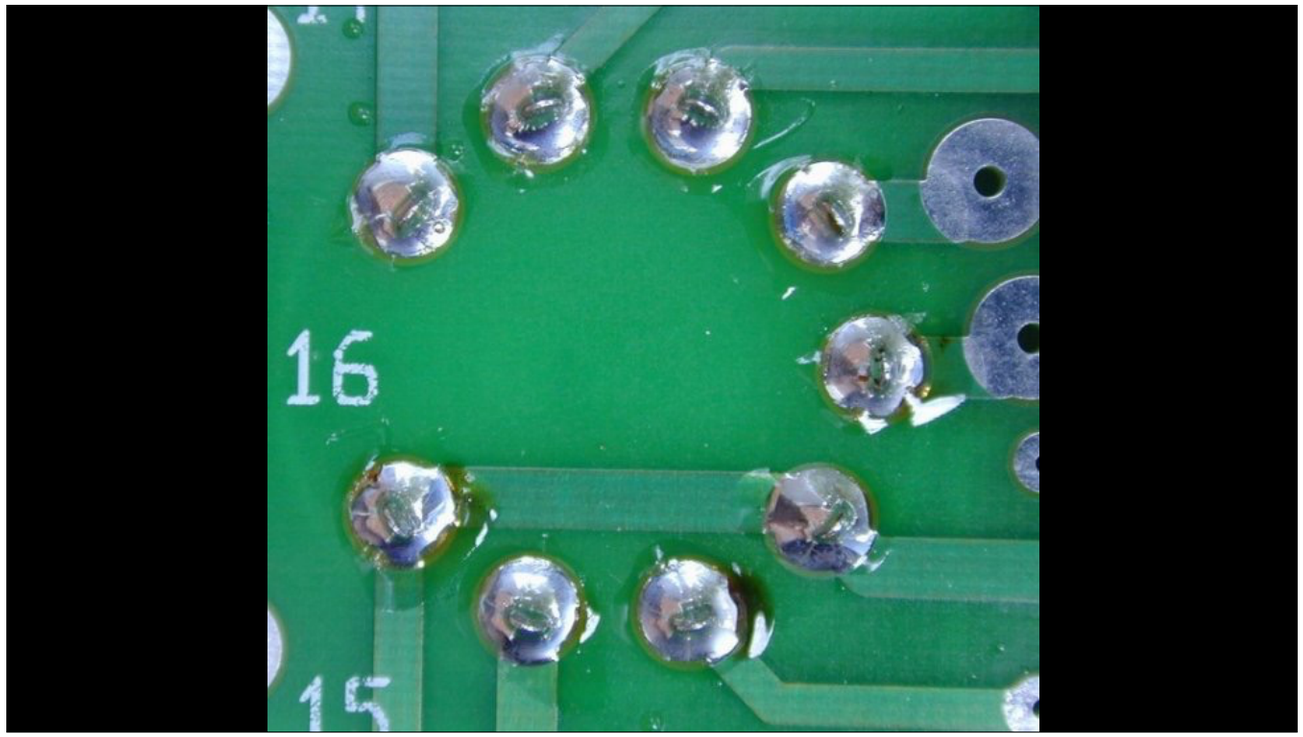
See that little puff of smoke? That's the rosin flux boiling away. Sometimes people see that and worry, "Oh no, lead fumes!" That's not the case…lead boils at THOUSANDS of degrees, this is just a few hundred. If you find yourself doing a lot of soldering, and find the smoke irritating, they make little filter fans that suck it away.

By the way, don't blow on the connection after soldering. Let it cool down at its own rate, it only takes a few seconds. Blowing on it makes the connection weaker.

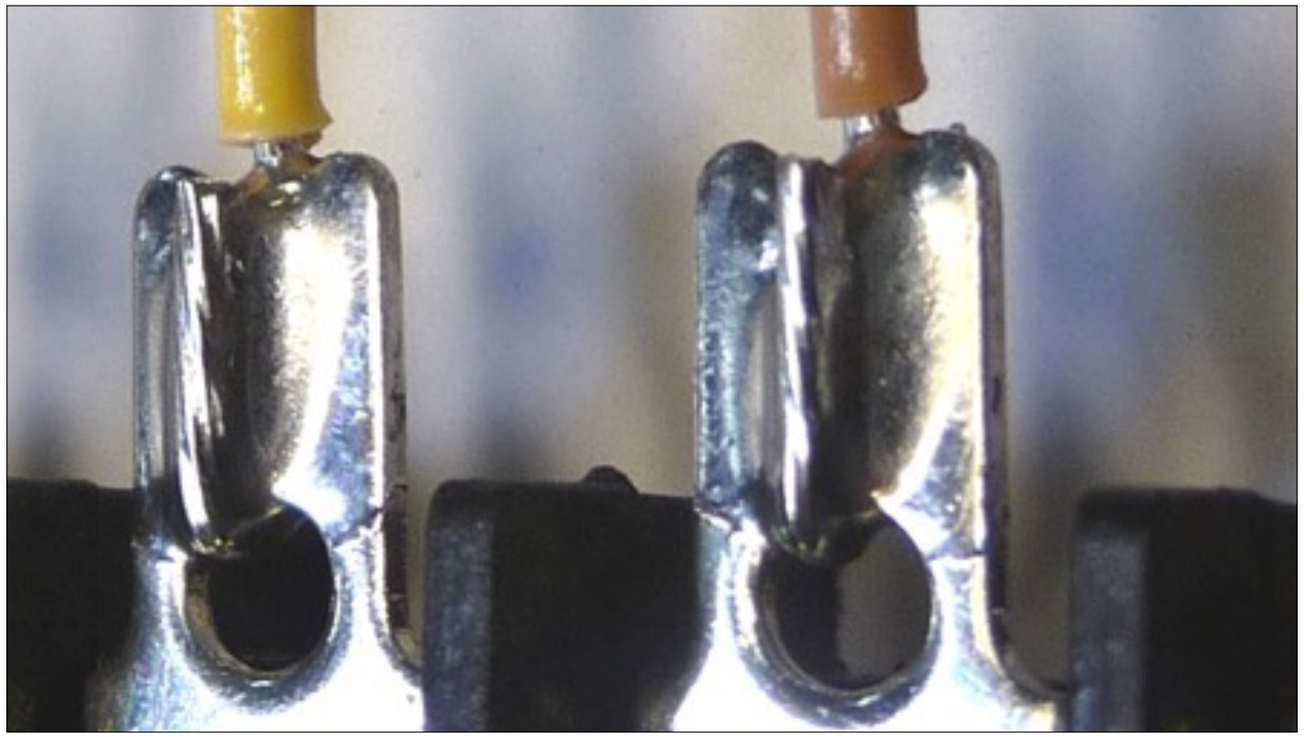
This is component-to-board soldering. The parts are different, but the technique is nearly identical…tinned iron heats the part for a moment, more solder is then added and allowed to flow for a second or two before pulling away. A little solder is left on the tip, so it's already tinned for the next part…you can work pretty quickly down a whole line like this.

Every 5 or 10 connections or so, you might find the solder getting "sticky" as all the rosin flux has boiled off. When that happens, go back to step one…wipe the tip on the sponge to clean it, and add a little drop of fresh solder, then resume.

And boom, that's it, that's really all there is to good soldering!



Here’s an example of some good soldering. Notice all the connections are shiny…meaning the iron is hot enough, and the solder was allowed to cool at its own rate… and also, they’re concave, like little Hershey’s Kisses. That shows that the solder FLOWED well. If it forms a little ball instead…there might not be any connection at all to the board underneath…the solder may be sitting on a layer of oxide or filth!

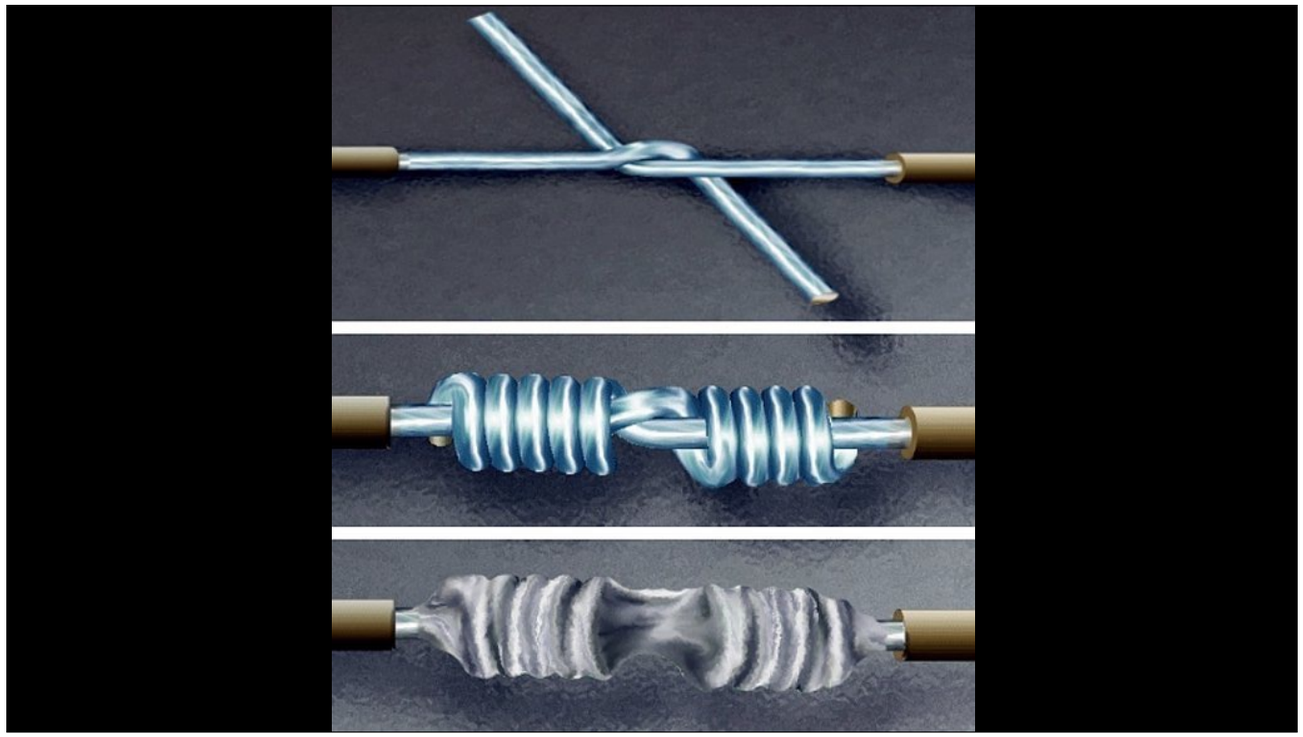


More good soldering, this one shows some wires connected to lugs. But notice again, the connections are shiny and they’re concave.

(Not dialogue, just a note here: I’ve foregone showing bad soldering; only good connections are shown. Partly this is a time-saver. But also, following a principle from mountain biking: hazard or target fixation. Basically, visually fixating on a bad thing can make you run right into it; goal is to look at the successful path through.)

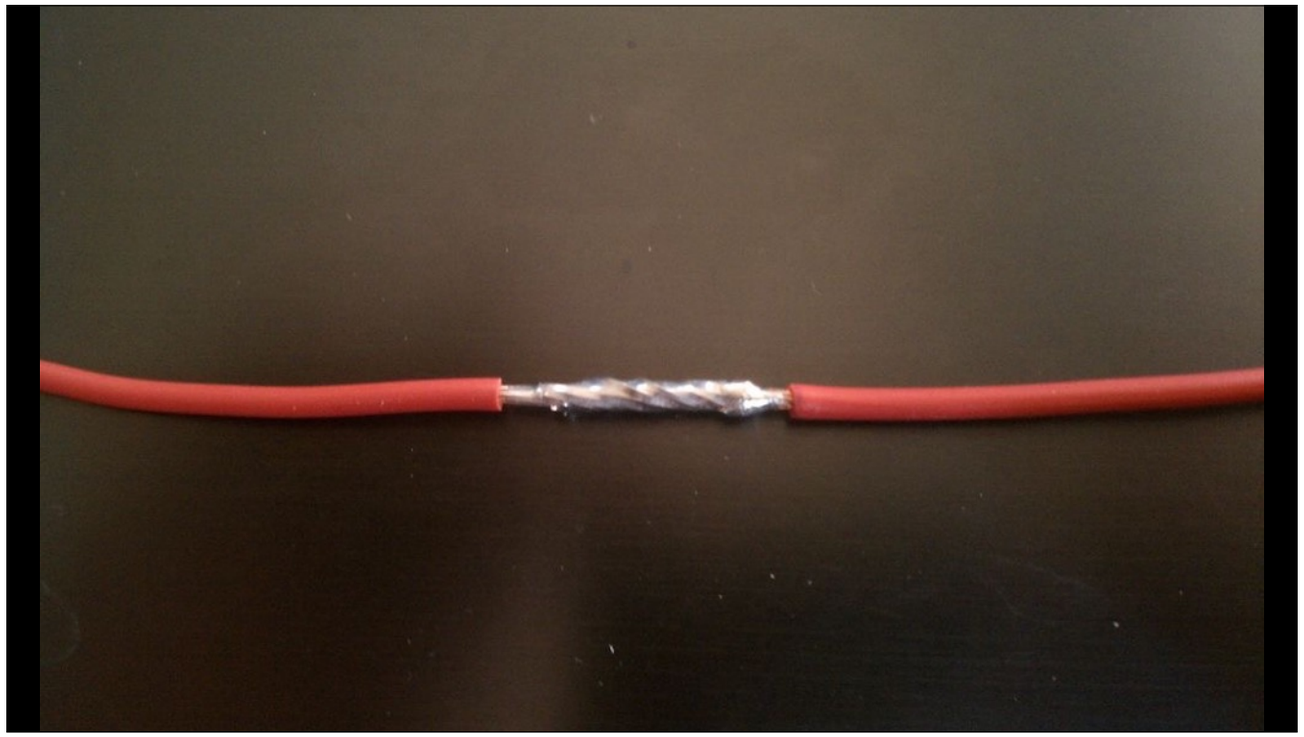
As an aside here…if you're doing wire-to-wire soldering, don't do this. Don't join the wire's side-by-side.

This is called a "pigtail." It's perfectly fine for something like residential wiring, like the lights in your house. But houses generally don't get up and walk around. In the rough-and-tumble world inside a costume, this kind of connection is very weak…if the two wires are pulled in opposite directions, it'll break.



What you want instead is a “linesman splice,” also called a “Western Union splice.”

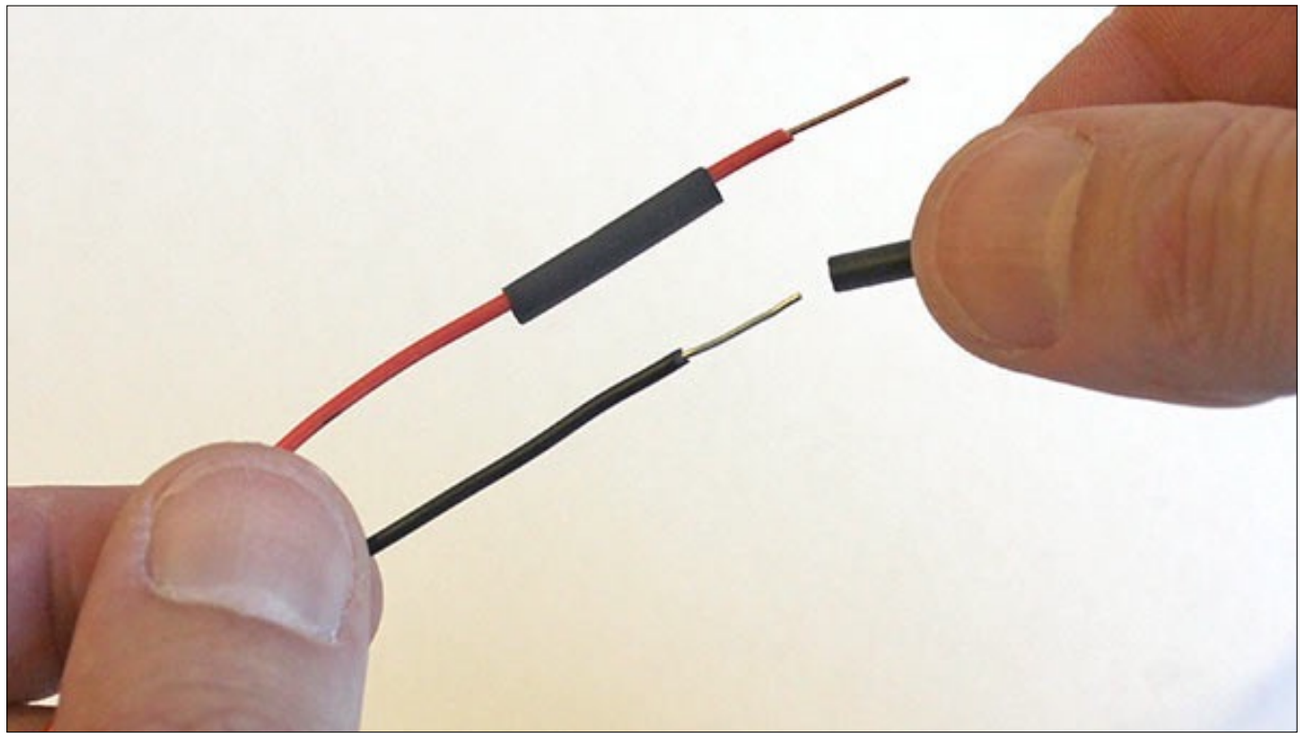
The two wires are joined end-to-end. Each twists around the other, then they’re soldered together. This withstands pulling much better.



Here’s a decent wire splice. They’re twisted around each other, and we can see good soldering form again…shiny and concave.

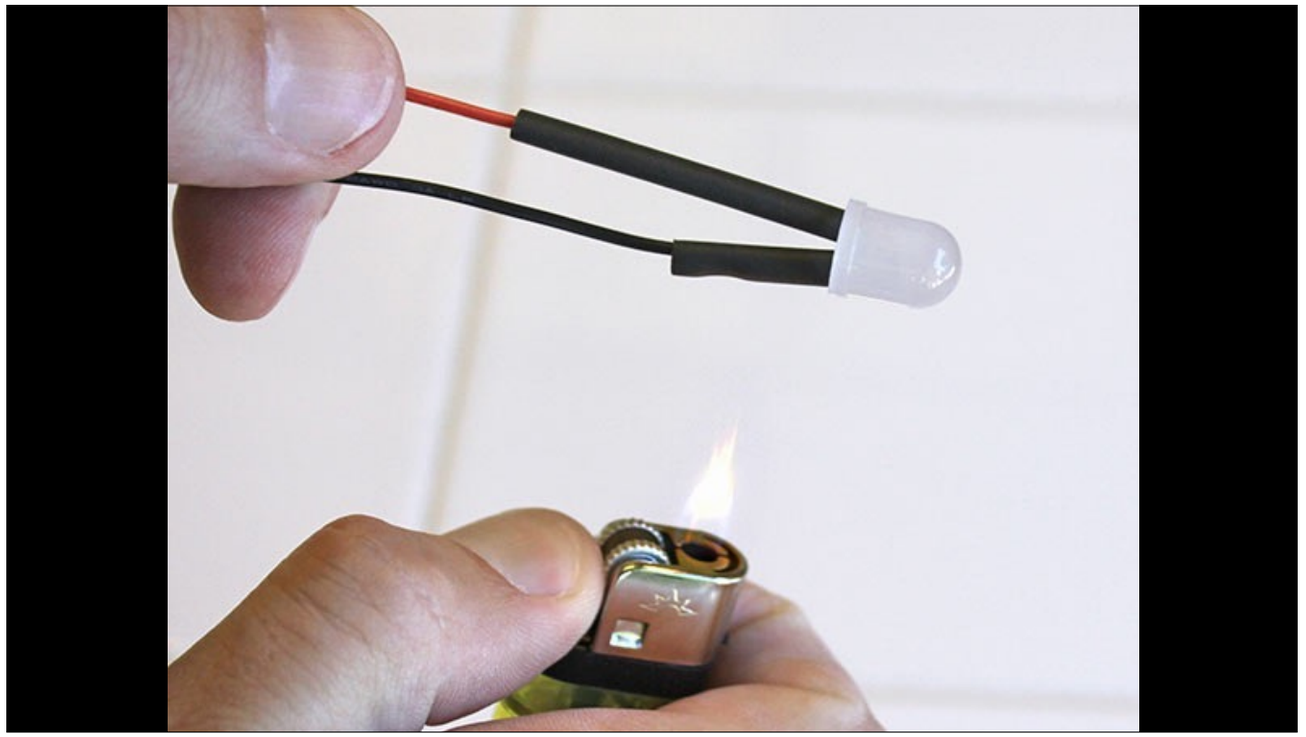
HOWEVER…we can’t just leave a connection out in the open like this. I’d mentioned “short circuits” earlier. An exposed connection like this is asking for trouble.

The antidote is HEAT-SHRINK TUBING. It’s a type of plastic tubing that contracts when heated. Comes in all different sizes…you’ll probably want an assortment of sizes on hand. If we apply this over the connection, it’ll prevent electrical shorts and also helps keep moisture out. Costumes and masks are FULL of moisture!



Thing to remember about heat-shrink tubing is that you put it over the wire BEFORE soldering. Slide it way down, away from the hot iron, make your solder connections, then slide it back…

Been doing this for decades and I STILL make that mistake sometimes! We all do.



Slide the tubing over the connection, then heat it up. A cigarette lighter can work for this, just leave some distance…use the rising heat, don’t char the plastic in the flame.

They make a special tool for this…a mini heat gun…but if you go to Michael’s and look for an “embossing heat tool” in the scrapbooking department, this is exactly the same thing!

(Don’t have a photo for this — just held up the actual item.)

This is one of my favorite tools. If a big wave came and washed away all my electronics stuff, first thing I’d replace is my soldering iron, but second would be this heat gun, it’s that nice to have.

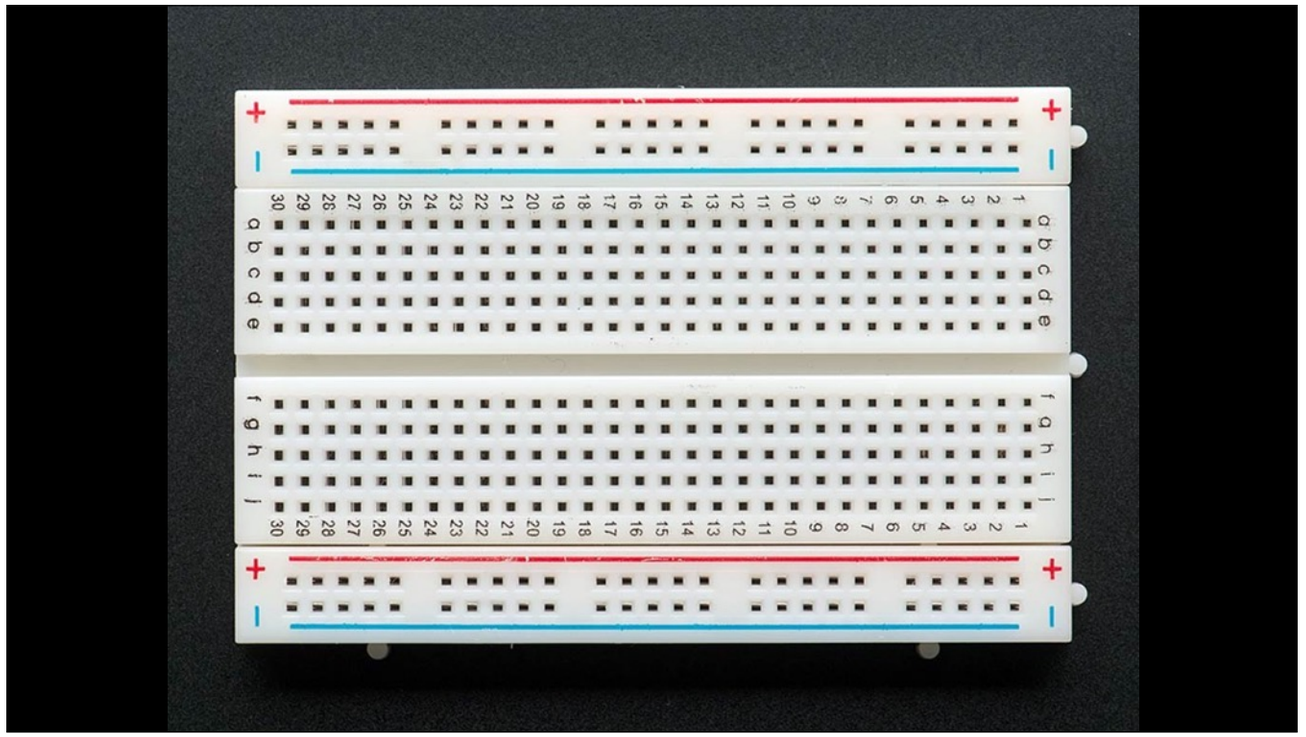
Avoid using electrical tape. It doesn’t seal well, it eventually comes unstuck and leaves a horrible reside behind. It’s just the wrong tool for any job.

When you're soldering, accidents will happen. Sooner or later, you'll need to UN-solder something. There are a couple of tools for this…

The vacuum desoldering pump, or "solder sucker," slurps up molten solder that you've re-heated with your iron. You cock this lever, melt the solder with your iron, then bring this in and press the button.

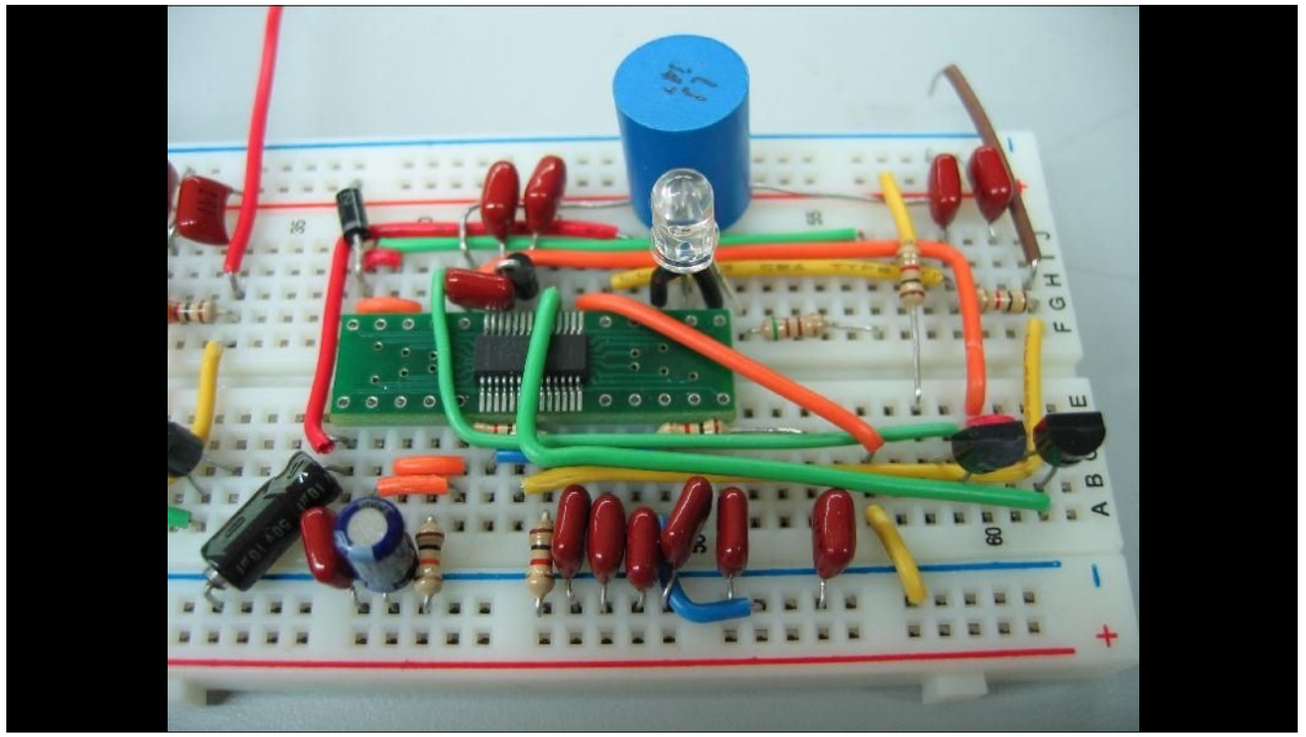
Copper desoldering braid is like a paper towel for solder. You press this on top of the solder mistake and heat it up with your iron, and capillary action pulls the solder into the fibers.

Sometimes you need to use both. And still other times, there's just no recovering. Fortunately most electronic components are pretty cheap…sometimes you just have to clip off the mistake and start over.

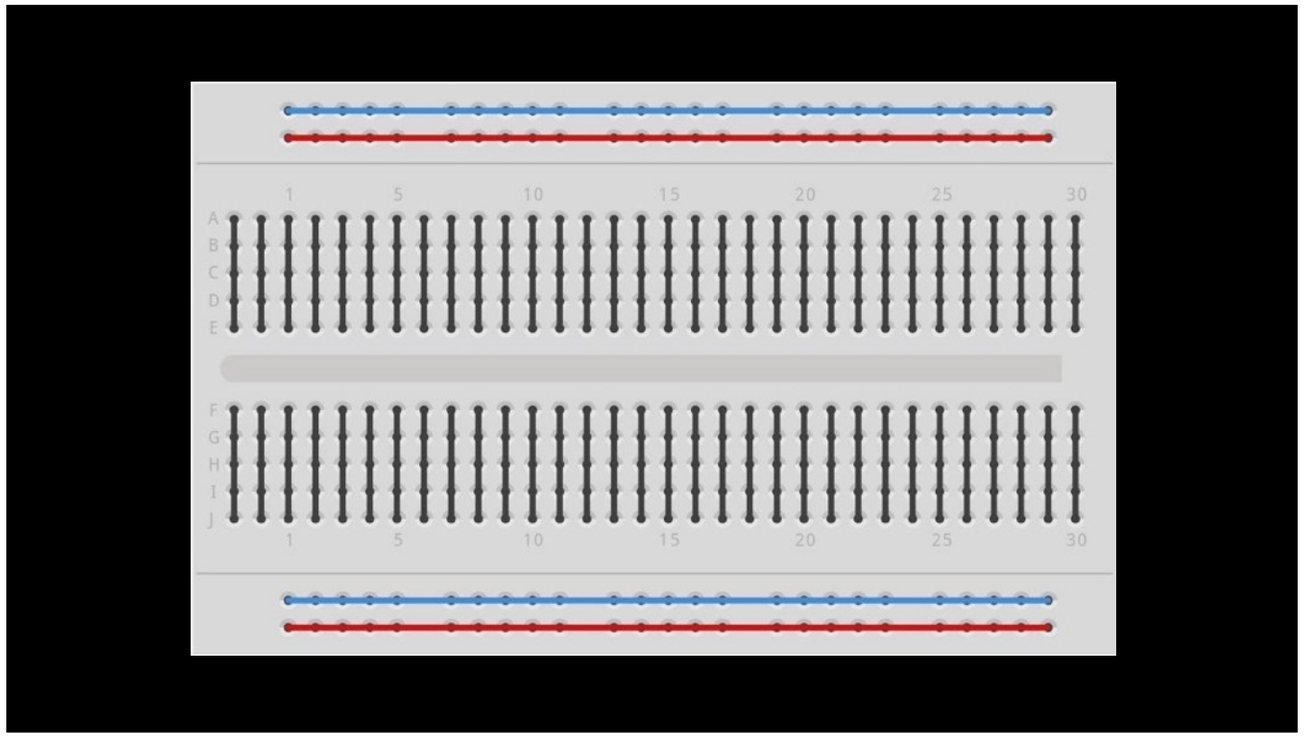


One last thing on this topic. In this case, NOT soldering.

Sometimes you’re just testing ideas out and might not want to commit to soldering parts together yet. Prototyping! There’s a tool specifically for this, called a SOLDERLESS BREADBOARD.



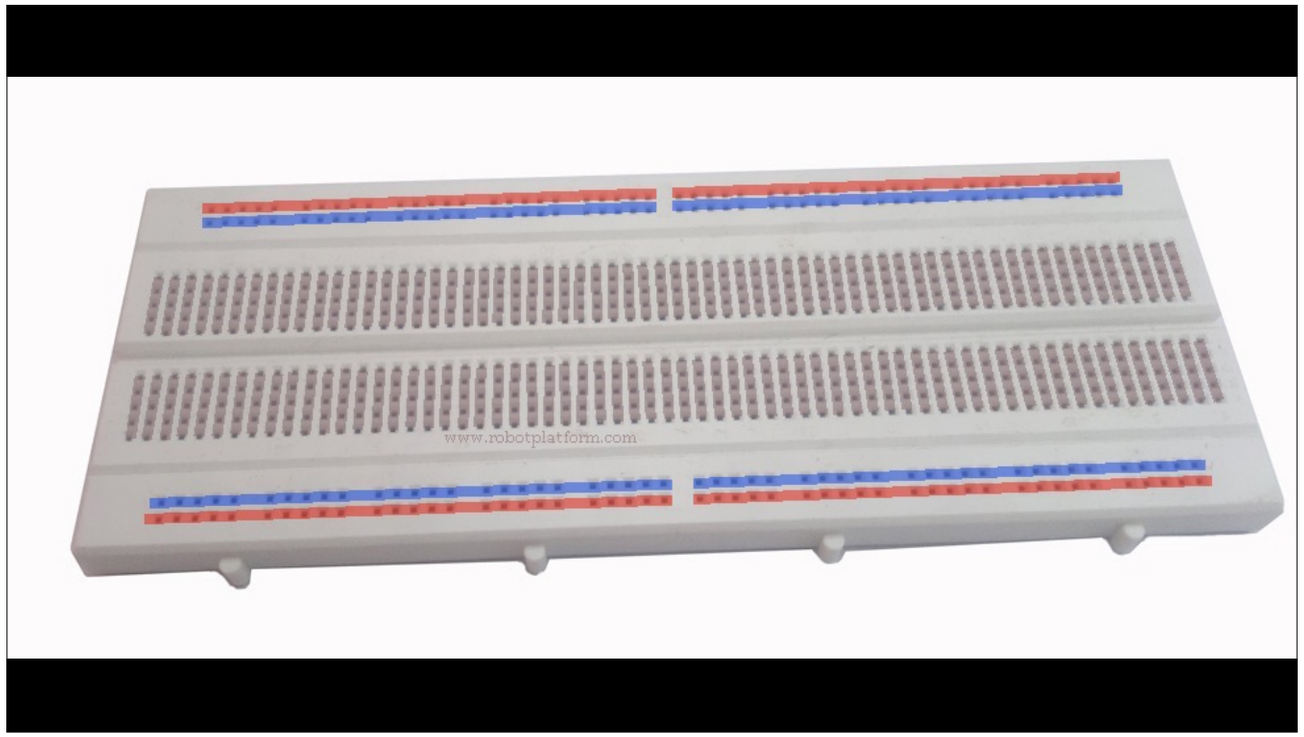
You can press components into this and make temporary connections. Test things out, move parts around. When you’re satisfied it all works, you can then solder together a more robust version. All the parts can be pulled out and re-used again in future projects.



It seems almost magical…how are connections made between the different parts?

Here's an "X ray view" of what's inside a breadboard. There are metal spring clips that connect each of these rows of 5 pins together. Anything pressed into the same row gets connected together. If you want something in the other direction, you use a short wire.

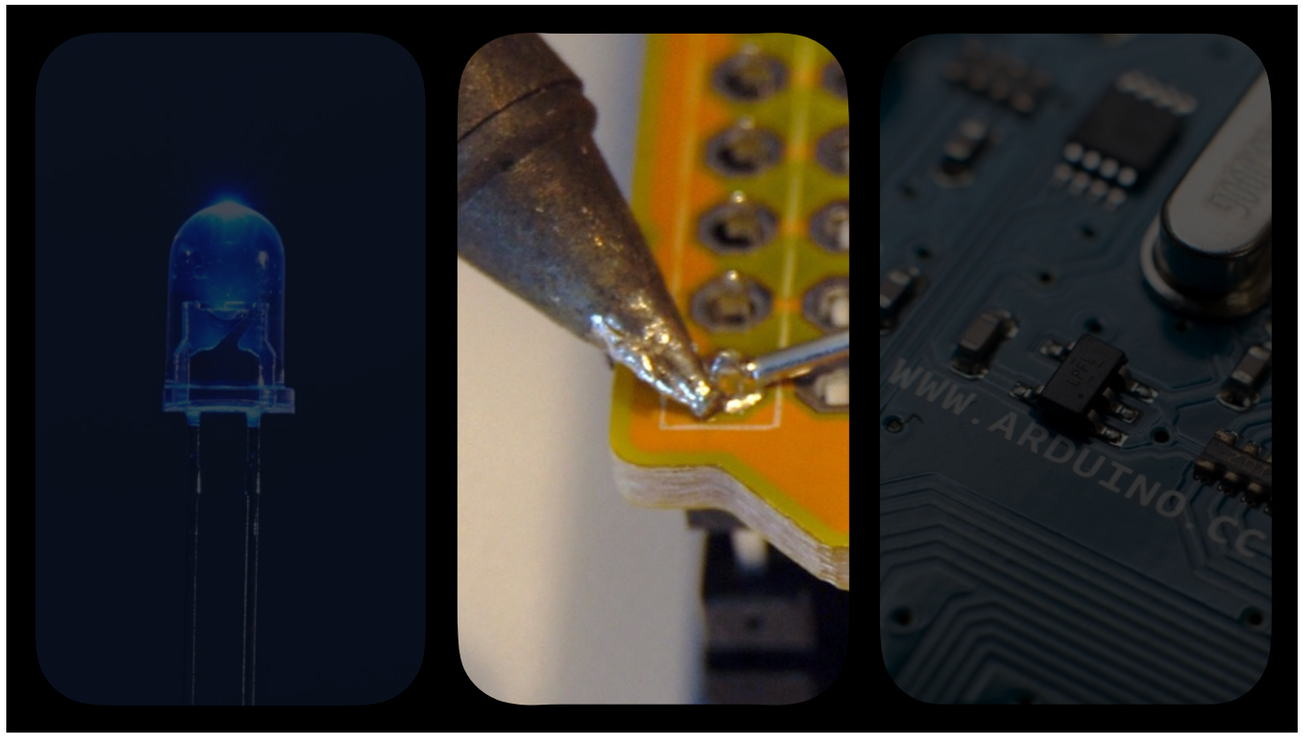
Along the outer edges are two long strips called "power rails." In most circuits you'll need multiple connections to power and ground. That's what these are for.



Common support issue I see is half a circuit on a breadboard not working.

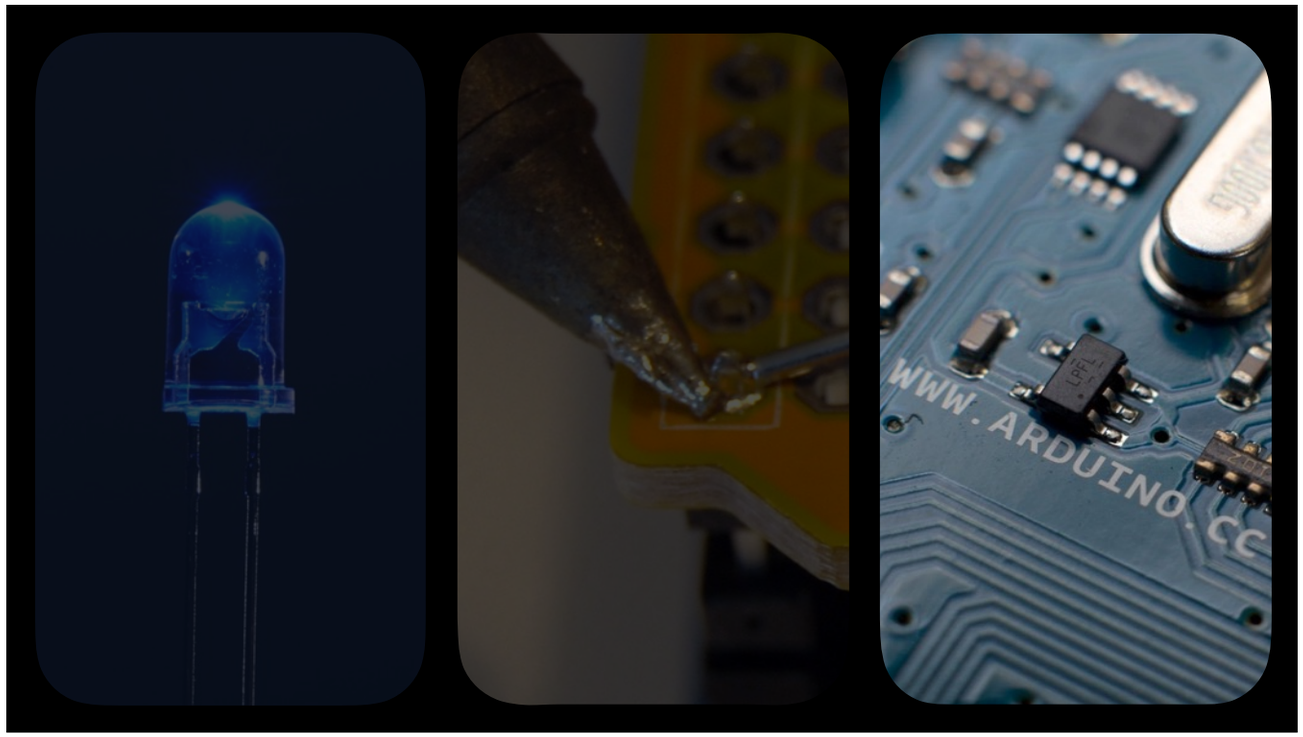
If you have one of these larger breadboards, be aware that the power rails usually aren’t contiguous down the entire length. There’s often a break in the middle. This is on purpose, because some circuits need multiple voltages.

If you have one of these, and need the power rails down the entire length, remember to add wires across these gaps in the middle.



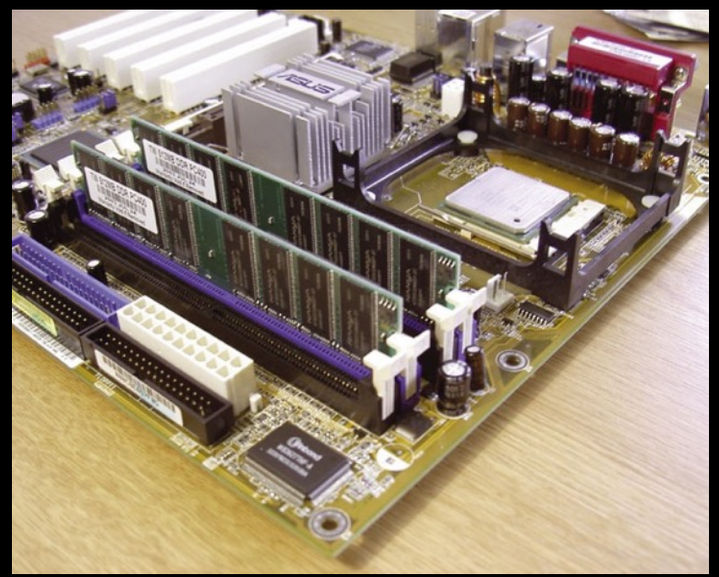
That wraps up the soldering part of the talk…

…if there’s any questions, I’m just gonna leave this loop running while we talk, to really drive home the “proper soldering” message.



Last topic I’d like to bring up is MICROCONTROLLERS.

(Note: this was a 90 minute panel and with Q&A filled the entire time slot. For a shorter 60-minute talk, I might trim this out and make it an entirely separate 60-minute talk that goes slightly more in-depth.)

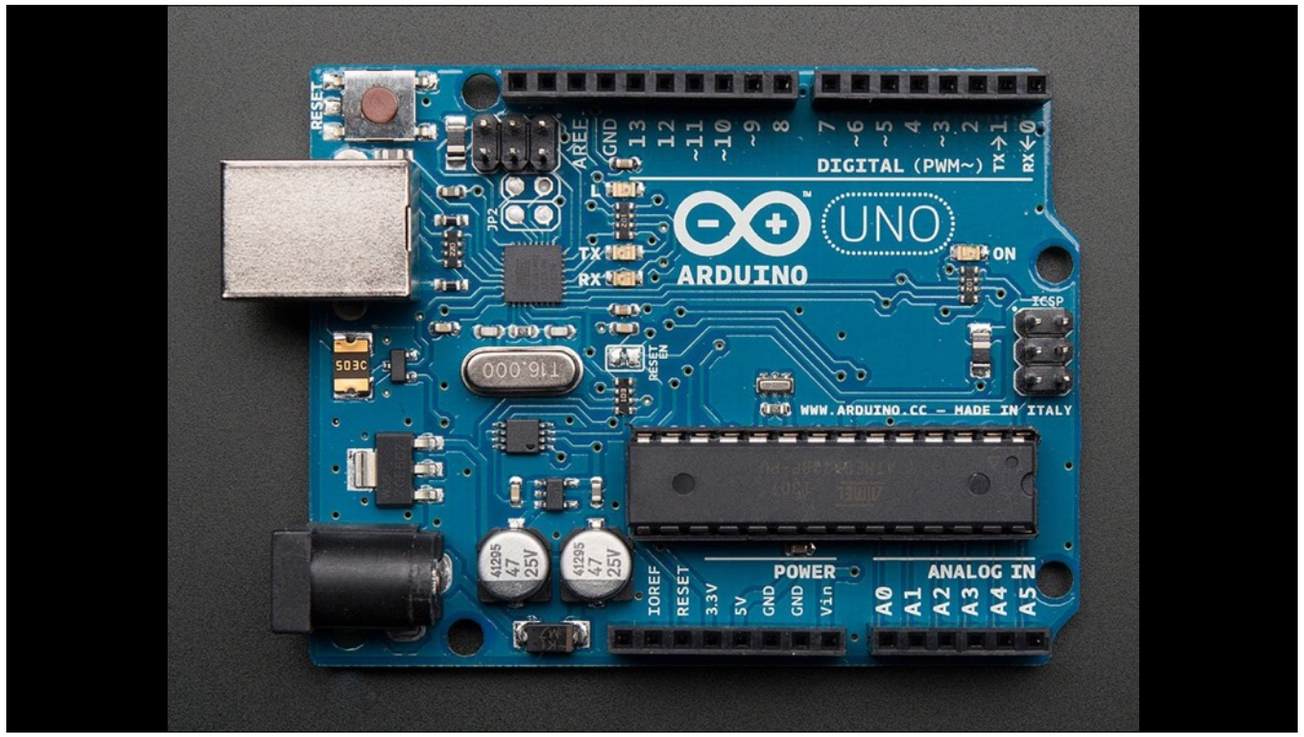


A microcontroller, as the name kind of implies, is a teeny tiny computer.

Everything that's present in your typical desktop or laptop computer — a central processor, some RAM and storage — all these things are present in a microcontroller, but in a very limited amount. Whereas your computer may run at gigahertz speed and have gigabytes of RAM and terabytes of storage, a microcontroller runs at just a few megahertz, with a couple kilobytes of RAM.

This might seem extremely limiting…but we give microcontrollers singular, simple tasks to focus on, and they do this exceedingly well. They're inexpensive and very power-efficient.

Now…this bare chip here…you can't do much with that as it is…



When working with microcontrollers, they're usually built into what's called a "development board." This gives us an easier way to connect power and wires, and often a USB port to upload software from a computer.

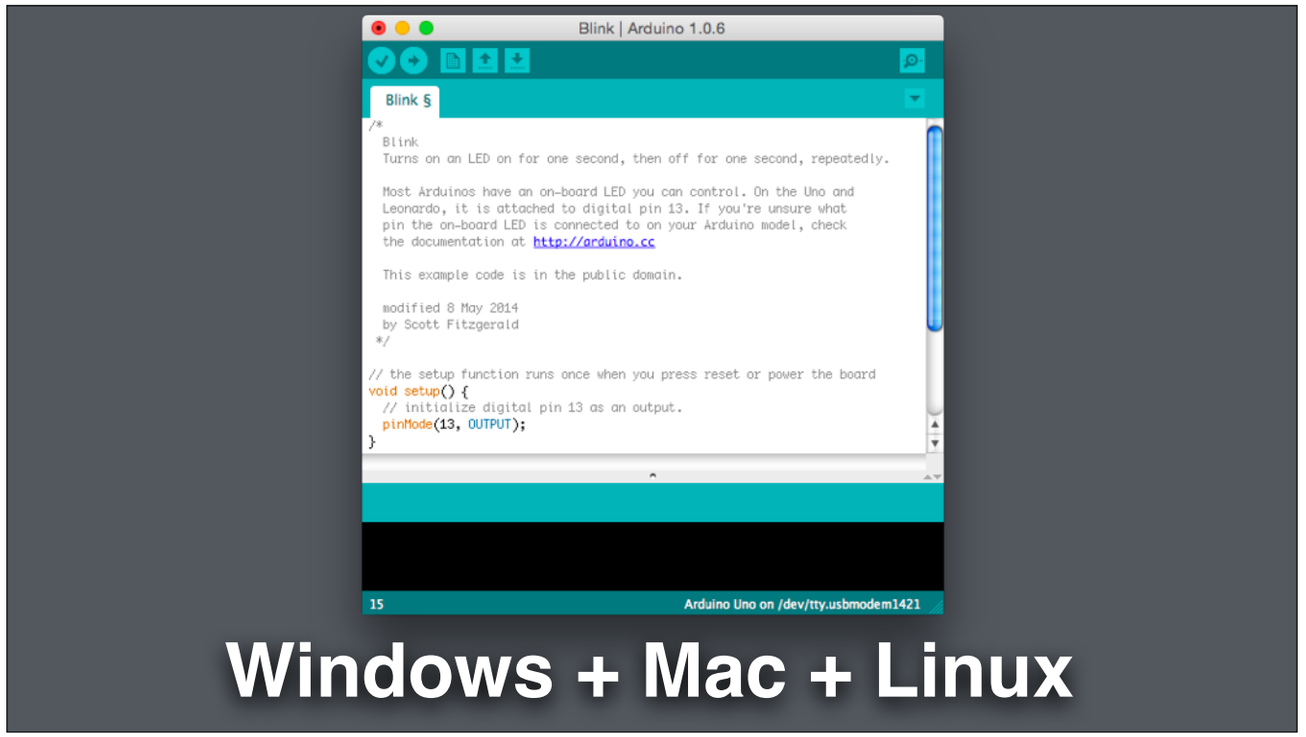
There are dozens, maybe hundreds of development boards. I'm going to focus on one in particular, called ARDUINO.

Funny thing about Arduino. It's not the most powerful, nor the cheapest, doesn't have the most RAM or anything. And yet, if you're just learning this stuff for the first time, I really strongly recommend starting out with this board, for a few reasons…

What sets Arduino apart from others is the available learning resources. There are entire books on the subject. Web sites packed with free lessons and projects. And even these ready-to-go circuit boards, called “shields,” that plug right on top to add functionality.

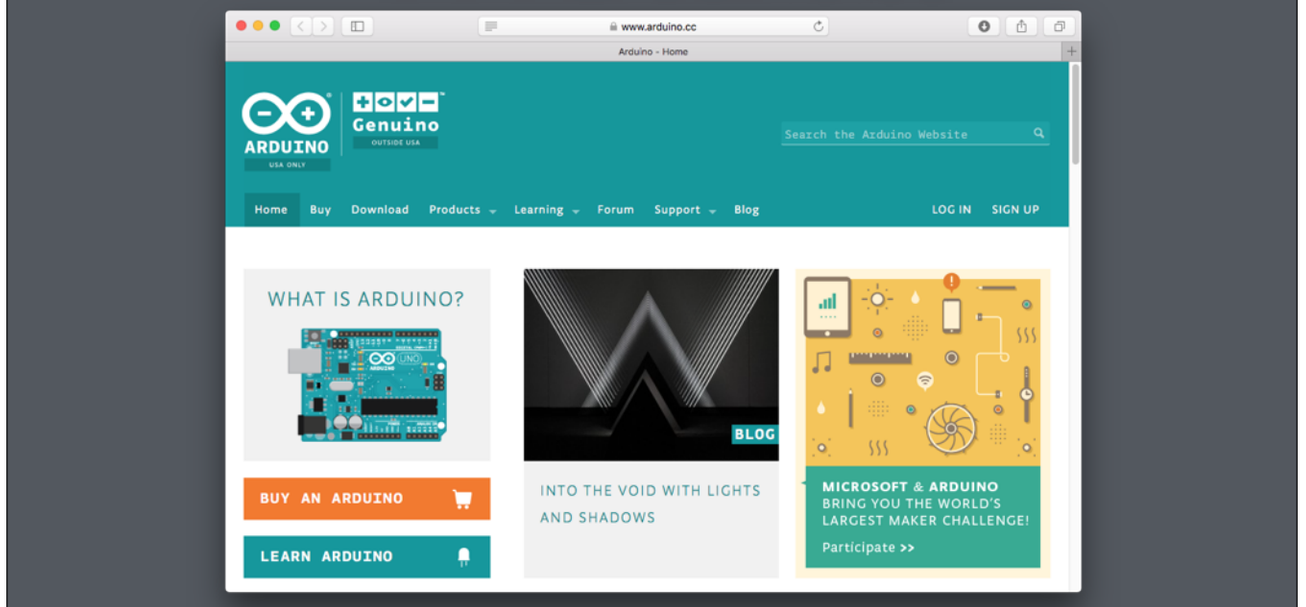
You plug Arduino into a USB port to upload programs.

To write these programs, you need special software, the Arduino development environment…

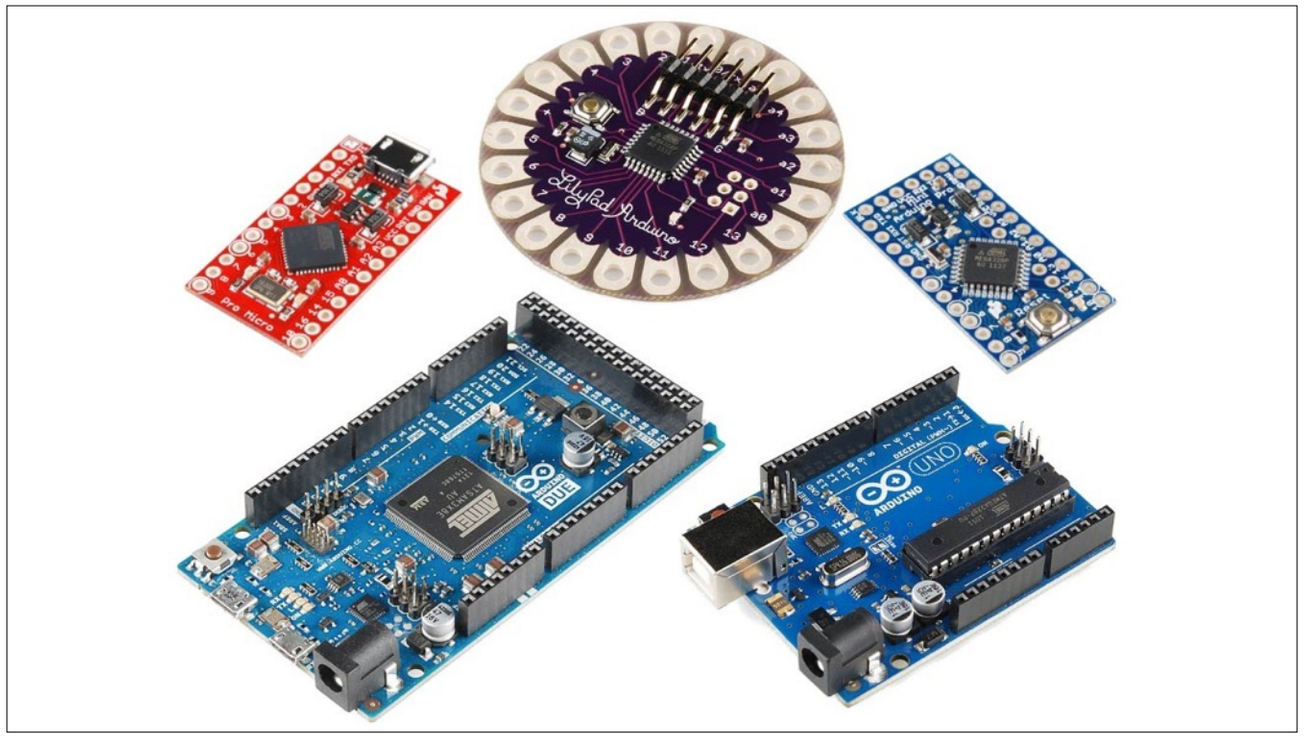


This software runs on most anything…Windows, Mac, Linux, it’s very inclusive…there’s even Android tablet versions.

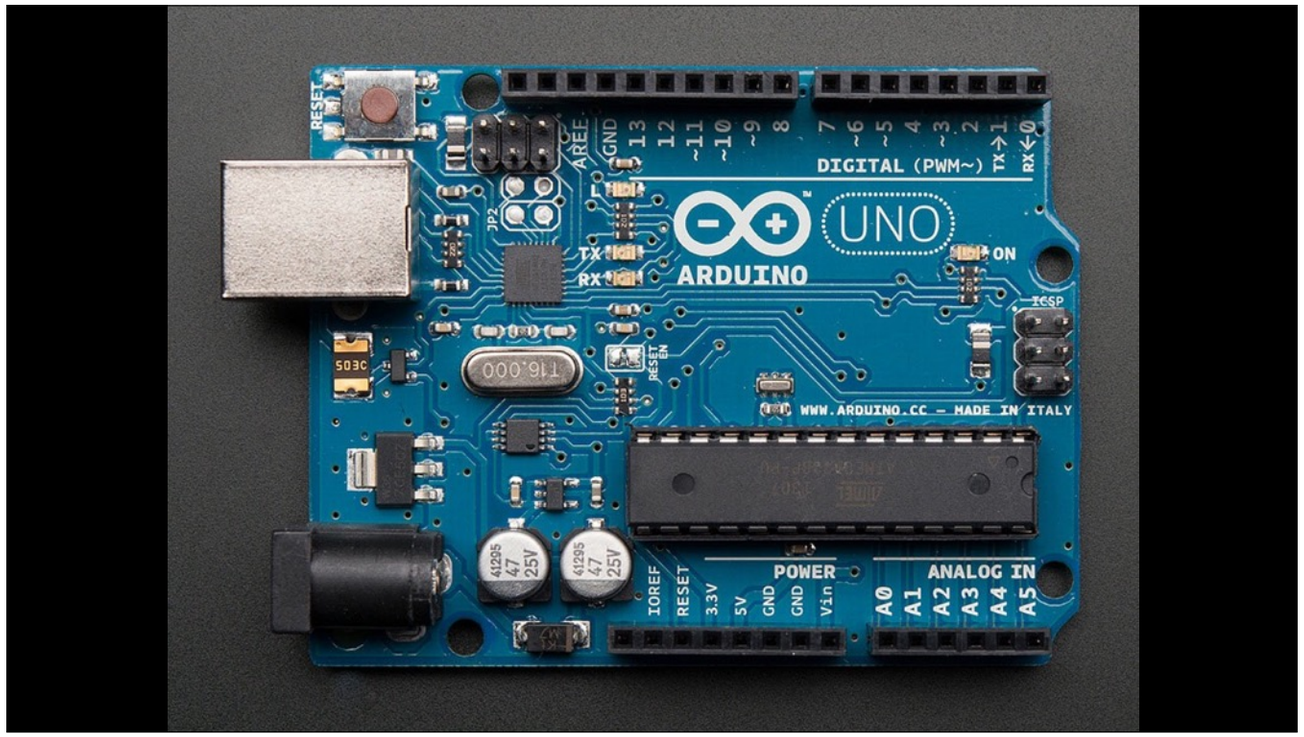
www.arduino.cc



And the software doesn’t cost a thing. You buy the Arduino board, the software’s free. You can download it from www.arduino.cc. DOT SEE SEE!



Now, if you go to buy an Arduino board, you'll find there are quite a few with "Arduino" in the name. And some of them have tempting names like "Mega" tacked on. You might think that's better, because "Mega!" But if you're starting out fresh, I really REALLY recommend getting the "Arduino Uno" board.



Reason being that most of those learning resources I mentioned are designed with the Arduino Uno in mind. You can always get a more sophisticated board later, when you need it, but start with the Uno for learning.

Now, understandably, you'll be looking to save money. But I need to warn you about something here.

Arduino is so popular now, there are actually fake knock-off boards out there. They may include the Arduino Uno logo, but are made using counterfeit chips and sub-par manufacturing processes. This is actually a problem! Remember I mentioned technical support? I see this all the time…these boards can handle simple programs, but as soon as you put it to a demanding task, issues crop up.

Even if you buy from a "reputable" source like Amazon…keep in mind, what you may be dealing with is an "Amazon affiliate" — not Amazon directly, but a fly-by-night distributor in China selling these low-grade boards.

If it's under $20, I can pretty much guarantee it's not a legitimate Arduino Uno. Trust me, the difference in price isn't worth the frustration.

Programming:

- **Logic**
- **Syntax**

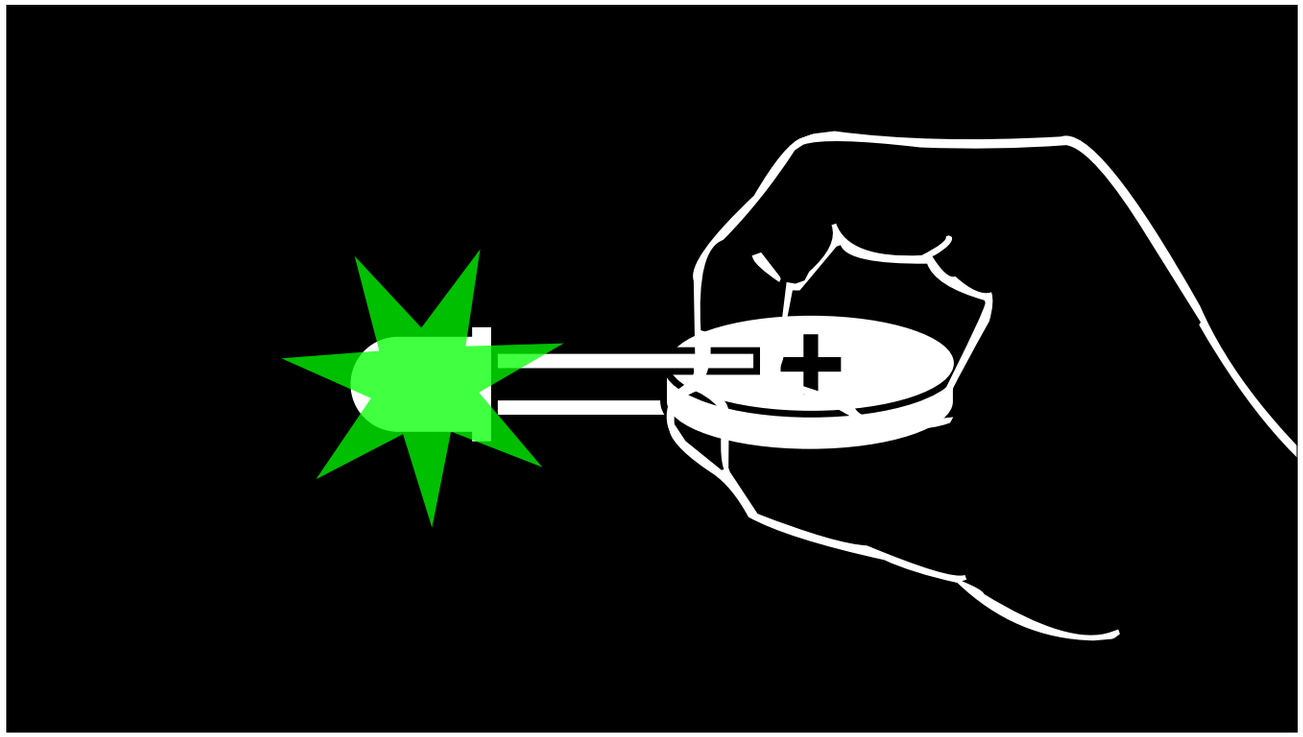
Now, I'd mentioned we'd be writing programs…software that runs on the Arduino board.

Programming isn't too difficult, but I think what trips people up is that you're actually learning TWO things at once…

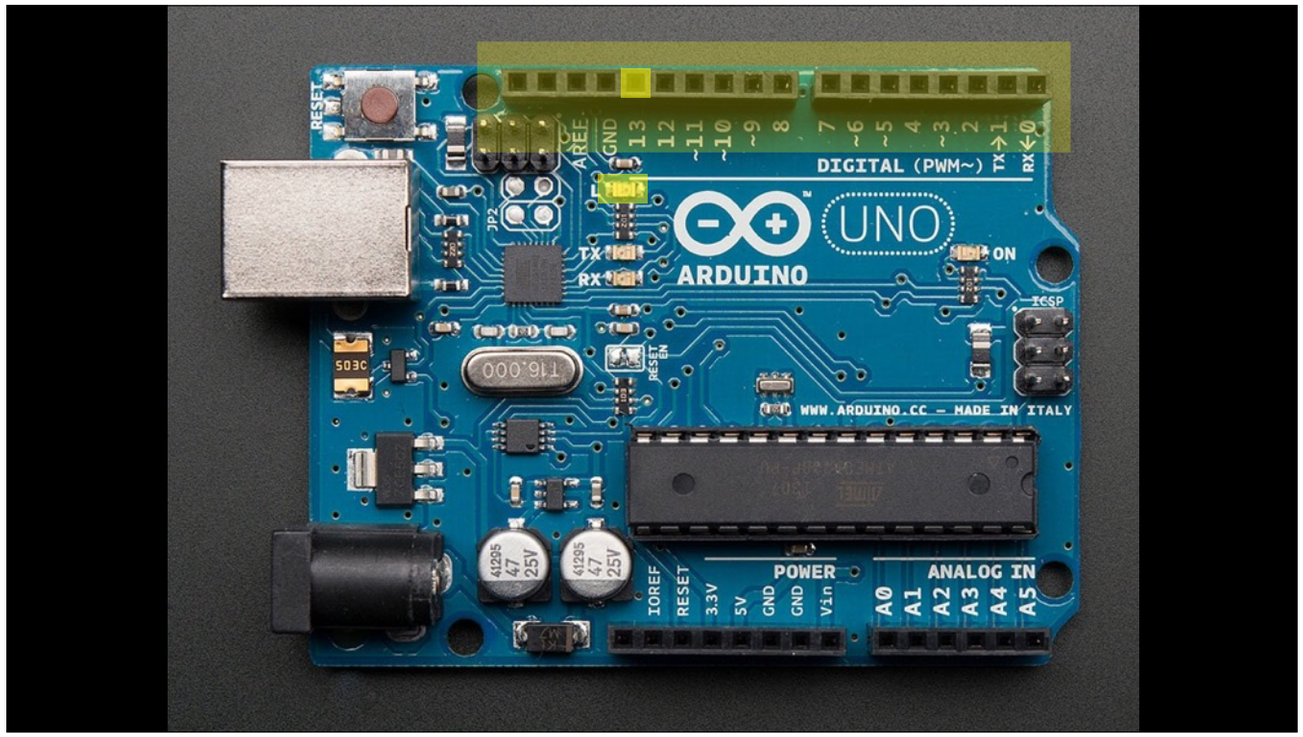
LOGIC is how we break a problem down into simple steps that a computer can follow. SYNTAX is how we express that. It's the dotting-the-i's and crossing-the-t's. Syntax is a DESTINATION; once learned, you pretty much know it. Logic is a JOURNEY; there's always new approaches to be discovered.

If you've never programmed before, I think the closest thing in the real world would be a recipe. You have a list of ingredients, and a list of steps to follow…always top-to-bottom, in order. Programs work just like that.

BUT ALSO…you know when you bake a cake, and there's that part where they ask "Turn the card over and follow the directions for making frosting"? There's something in programming EXACTLY like that. It's called a "function." It's a series of steps that have been set aside and can be referenced later by name. When baking a cake, you have a "frosting" function.



So recall earlier, how we pinched an LED to a battery and made it blink? Suppose we want to make the Arduino handle this task automatically?



In fact, this is SUCH the canonical first program, they've added an LED right on the Arduino board for this.

Up along this edge of the board, we have what are called the DIGITAL I/O PINS. "Digital" meaning they have one of two states — on/off, yes/no, one/zero. And "I/O" is short for "input/output" — each of these pins can either be an input, sensing things in the world, or an output, controlling something in the world. This socket connector lets you jam wires or components right into it. But this LED, on digital pin #13, is already installed, soldered directly to the board. You can plug in others later!

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

This is the whole LED blink program…but I'll break it down for you line-by-line…

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

These first two lines here…I'd like to show you an example of SYNTAX.

That double-slash? That's called a COMMENT. It means the rest of this line isn't actually computer code…it's notes that the programmer left for her future self, or for other programmers to help them understand what's going on.

Comments don't cost anything; they don't make your program any bigger or slower, so use them! It's very common to forget what a program was doing, so comments let you add a little documentation.

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

Next…remember I'd mentioned "functions"? A named list of instructions, like "frosting" in a cake recipe.

Every Arduino program has at least TWO functions. There can be more, but these two are required…

The SETUP function configures the hardware for some task we're about to do. It runs just once, at the beginning.

Following setup, the LOOP function then runs again and again, forever, or until you disconnect power. This is usually where the real work of your program happens.

See all these little curly braces and parenthesis and things? That's another example of SYNTAX. The braces mark the start and end of functions. I won't be going into every detail of syntax here, just be aware that these all do serve a purpose. There's a misconception that programming is intentionally arcane, as a form of job security, but these details really do mean something.

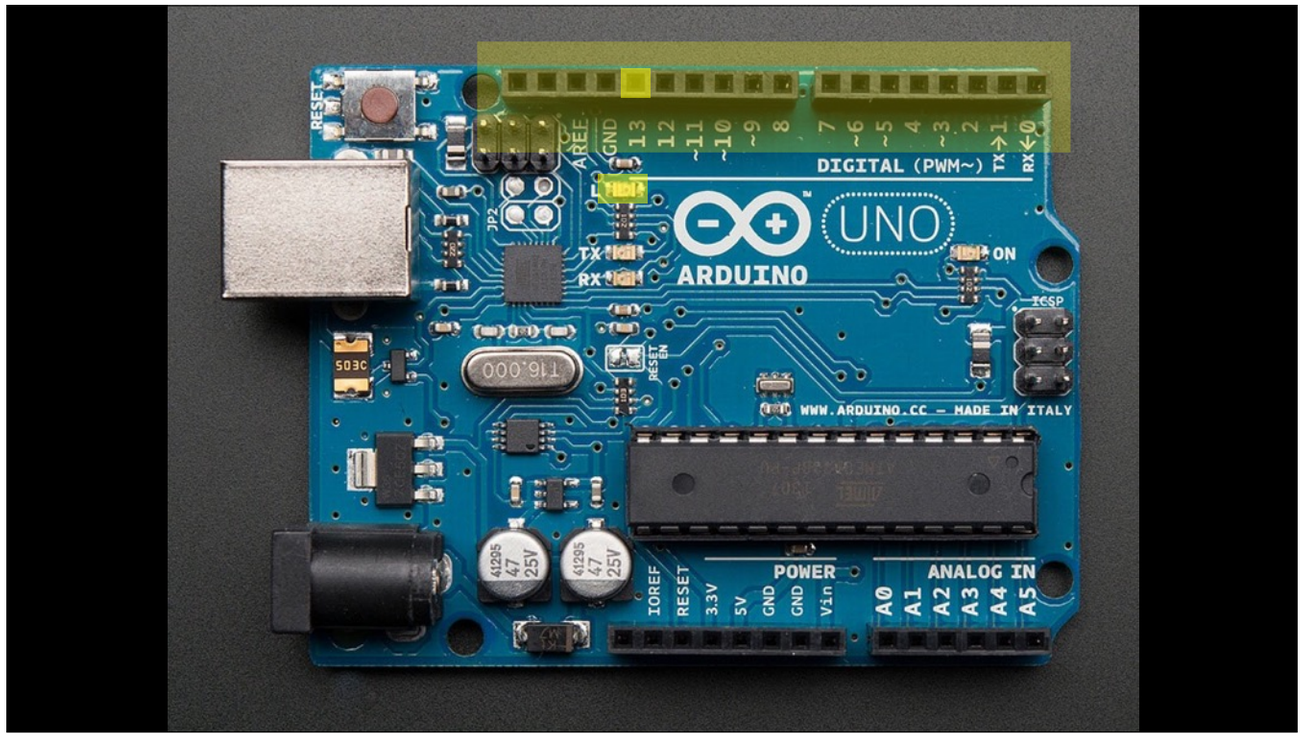
And look, more comments!

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

So, within the setup function, we want to configure the board for the task at hand…



We want to control the built-in LED that’s connected to pin 13…

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

To do this, we call one of Arduino's BUILT-IN functions…there are lots of these…in this case, one called pinMode, which lets us change a pin to an input or output. We want to CONTROL an LED, so we ask to make that pin an OUTPUT.

And that's it inside setup(), just one line.

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

Setup’s done, so we go into the loop function.

First thing there, we call another built-in function, digitalWrite, telling it a pin number and a “logic state” — HIGH or LOW — basically on or off. We want the LED ON, so we set pin 13 HIGH.

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

Next, we'd like the blink to be VISIBLE. The microcontroller handles tasks incredibly fast, and if we just turn the LED on and off at full speed, it's simply a blur — it appears to be constantly on.

To slow it down, we call another built-in function, delay. This stops the processor for some number of milliseconds. 1000, in this case, stops the code here for one second, during which the LED stays on.

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

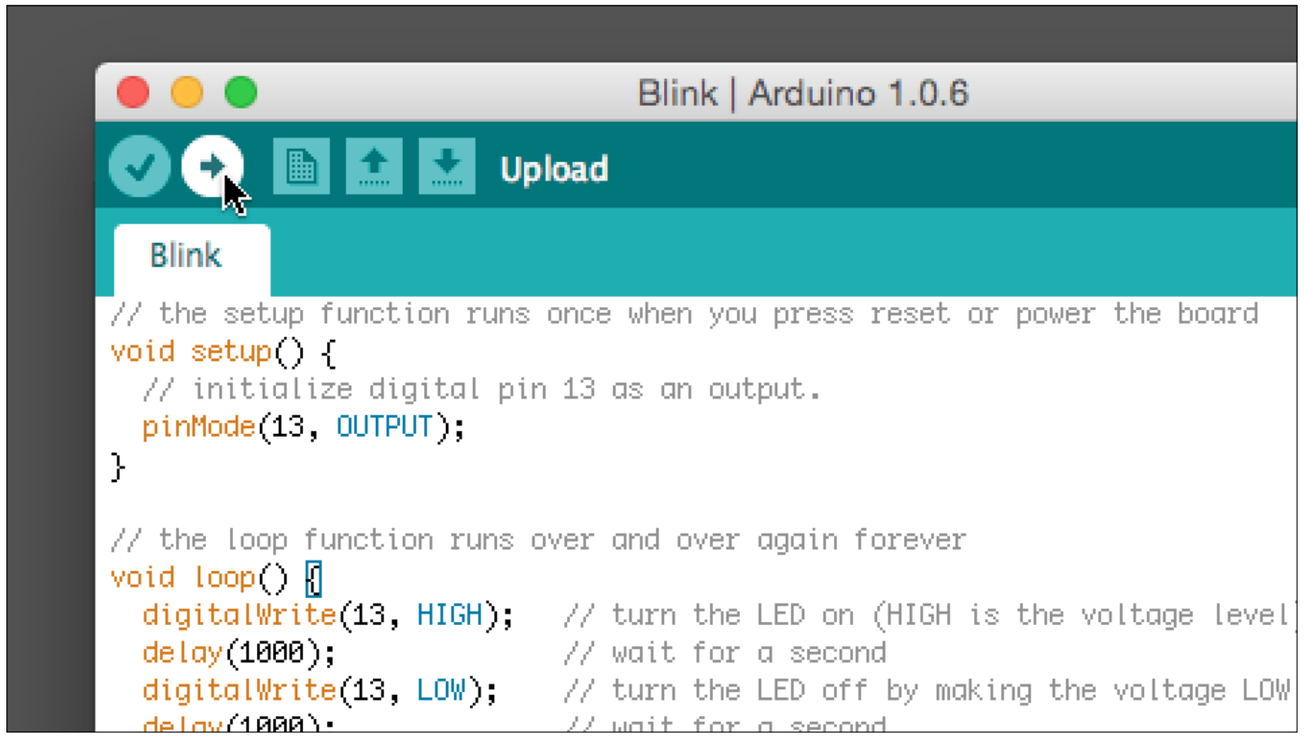
Then we turn the LED OFF and delay for another second. One second on, one second off. That’s the end of the loop function, which will start again from the top.

```
// These 2 lines are comments…a note
// for humans, not the microcontroller

void setup() { // Runs once
  pinMode(13, OUTPUT);    // Enable LED
}

void loop() { // Runs forever
  digitalWrite(13, HIGH); // LED on
  delay(1000);            // 1 second
  digitalWrite(13, LOW);  // LED off
  delay(1000);            // 1 second
}
```

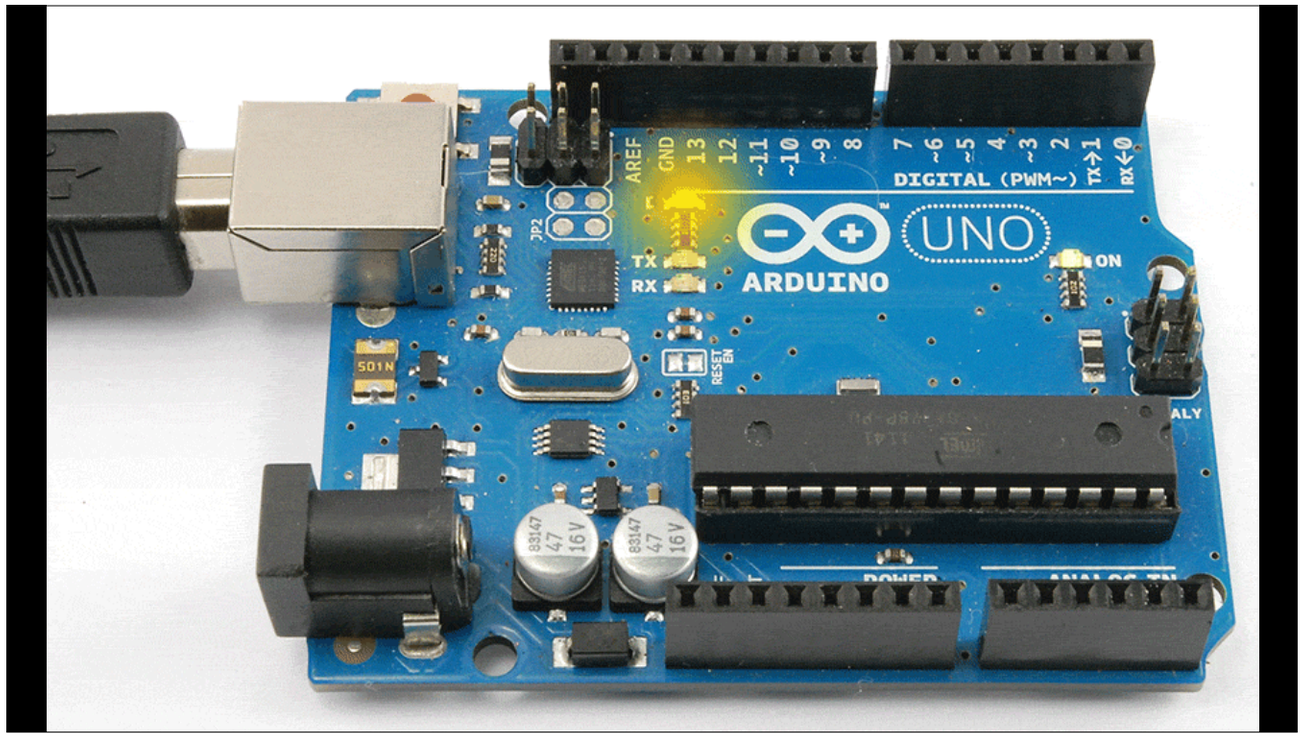
So here’s the complete program again…



Clicking this icon will then upload the program to the Arduino board…

BUT FIRST, it will scan your code to make sure the syntax all makes sense. If not, it will stop and tell you where there's a problem. The software can check for syntax errors, but not logic errors…only you can puzzle those out.

If the code's good though, it'll transfer through the USB cable, and…



…the LED blinks, just as we requested.

(Note: to improve visibility, I demonstrate this with a large 10mm LED on a proto shield. However, this may present a problem for audience members with photosensitivity; possibly even trigger a seizure. Going forward, I'll use a small buzzer instead as a stand-in for the LED.)

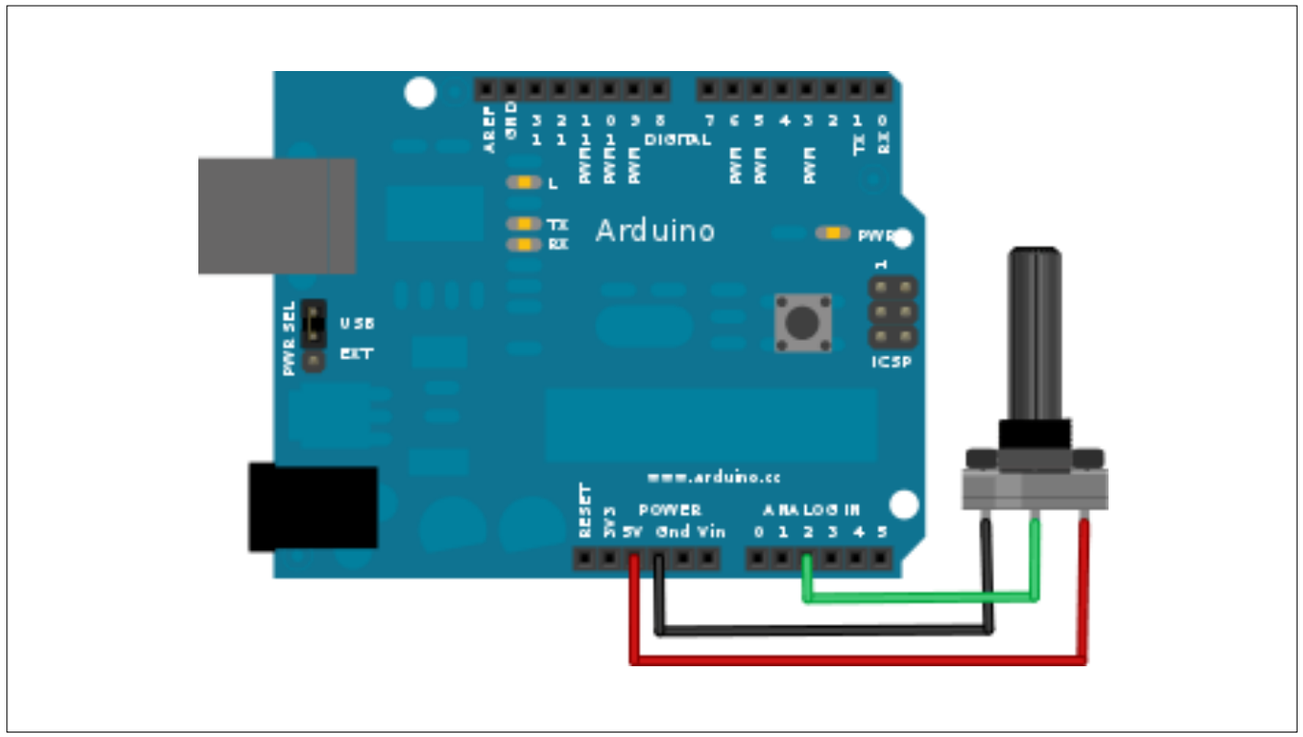
Now that's not terribly useful in itself, but we learned all the basics of programming right there. And it gives us a springboard into a vital concept…

The real power of programming is that you're rarely beginning from scratch. Most programs build atop other programs, whether your own or someone else's; there's tremendous intellectual leverage.

Let's suppose we wanted to add a dial to change the blink speed…

This is another component, called a POTENTIOMETER. Earlier I mentioned resistors. A potentiometer is an ADJUSTABLE resistor.

It has three pins on it…



The two outer pins connect to 5 Volts and ground, while the middle pin connects to this header over here…these are the ANALOG INPUT pins.

Unlike the digital pins, which have just two states…two voltage levels…the analog pins can sense any voltage in-between.

```
void setup() {
  pinMode(13, OUTPUT);     // Enable LED
}

void loop() {
  int dialValue;
  dialValue = analogRead(A2); // 0-1023
  digitalWrite(13, HIGH); // LED on
  delay(dialValue);
  digitalWrite(13, LOW);  // LED off
  delay(dialValue);
}
```

The code is mostly the same as before, just some small changes…

```
void setup() {
  pinMode(13, OUTPUT);     // Enable LED
}

void loop() {
  int dialValue;
  dialValue = analogRead(A2); // 0-1023
  digitalWrite(13, HIGH); // LED on
  delay(dialValue);
  digitalWrite(13, LOW);  // LED off
  delay(dialValue);
}
```

Inside the loop function, first we “declare a variable” — a place to store a piece of information and refer back to it later. Variables have names too…we’ll give ours a descriptive one, “dialValue.”

Another built-in function, analogRead, reads the voltage on one of the analog input pins.

analogRead gives us an INTEGER — a whole number with no fraction — between 0 and 1023, representing 0 to 5 Volts. We store that number in our variable, dialValue.

```
void setup() {
  pinMode(13, OUTPUT);     // Enable LED
}

void loop() {
  int dialValue;
  dialValue = analogRead(A2); // 0-1023
  digitalWrite(13, HIGH); // LED on
  delay(dialValue);
  digitalWrite(13, LOW);  // LED off
  delay(dialValue);
}
```

They we just change the delay() calls to use the stored delayValue instead of a fixed number of 1000 milliseconds.

```ino
void setup() {
  pinMode(13, OUTPUT);     // Enable LED
}

void loop() {
  int dialValue;
  dialValue = analogRead(A2); // 0-1023
  digitalWrite(13, HIGH); // LED on
  delay(dialValue);
  digitalWrite(13, LOW);  // LED off
  delay(dialValue);
}
```

And here’s the whole program again. Upload it to the board, and you can turn a dial and change the LED blink speed.

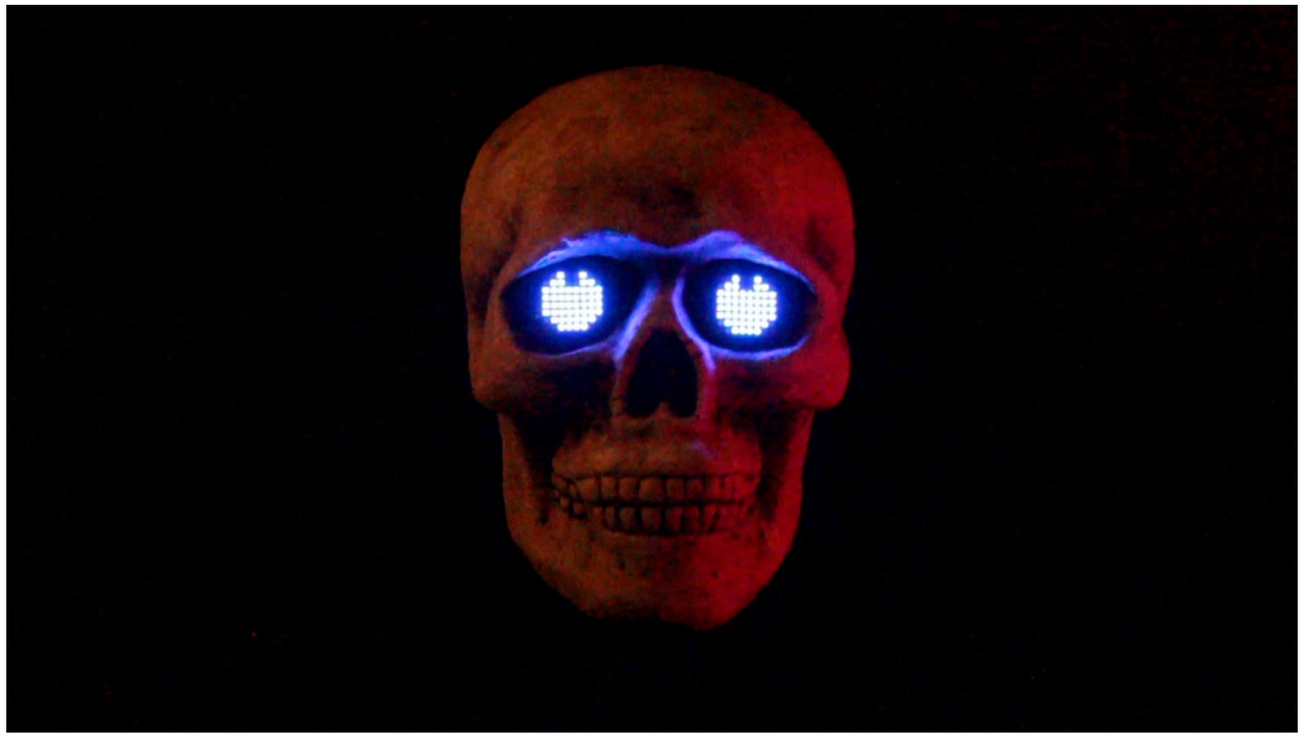
These are just simple starter programs…

```
// Render current blink (if any) into brightness map
if(blinkCounter <= blinkFrames * 2) { // In mid-blink?
  if(blinkCounter > blinkFrames) {    // Eye closing
    outer = blinkFrames * 2 - blinkCounter;
    inner = outer + 1;
  } else {                            // Eye opening
    inner = blinkCounter;
    outer = inner - 1;
  }
  y1 = upperLidTop    - (upperLidTop - upperLidBottom) * outer / blinkFrames;
  y2 = upperLidTop    - (upperLidTop - upperLidBottom) * inner / blinkFrames;
  y3 = lowerLidBottom + (lowerLidTop - lowerLidBottom) * inner / blinkFrames;
  y4 = lowerLidBottom + (lowerLidTop - lowerLidBottom) * outer / blinkFrames;
  for(i=0; i<16; i++) {
    y = pgm_read_byte(&yCoord[i]);
    if(y > y1) {        // Above top lid
      iBrightness[i] = 0;
    } else if(y > y2) { // Blur edge of top lid in motion
      iBrightness[i] = brightness * (y1 - y) / (y1 - y2);
    } else if(y > y3) { // In eye
      iBrightness[i] = brightness;
    } else if(y > y4) { // Blur edge of bottom lid in motion
      iBrightness[i] = brightness * (y - y4) / (y3 - y4);
    } else {            // Below bottom lid
      iBrightness[i] = 0;
    }
  }
} else { // Not in blink -- set all 'on'
  memset(iBrightness, brightness, sizeof(iBrightness));
}
```

Programs can grow and grow, sometimes hundreds of lines. Fear not…remember, you're often building this up incrementally. "Eat an elephant one bite at a time," as they say.

Also, some folks see a lot of math going on and want to call it quits. "I'm no good at math!" It's really not that daunting…truth is, NOBODY'S good at math. That's why we invented computers!

Back to elephants though…this code here…

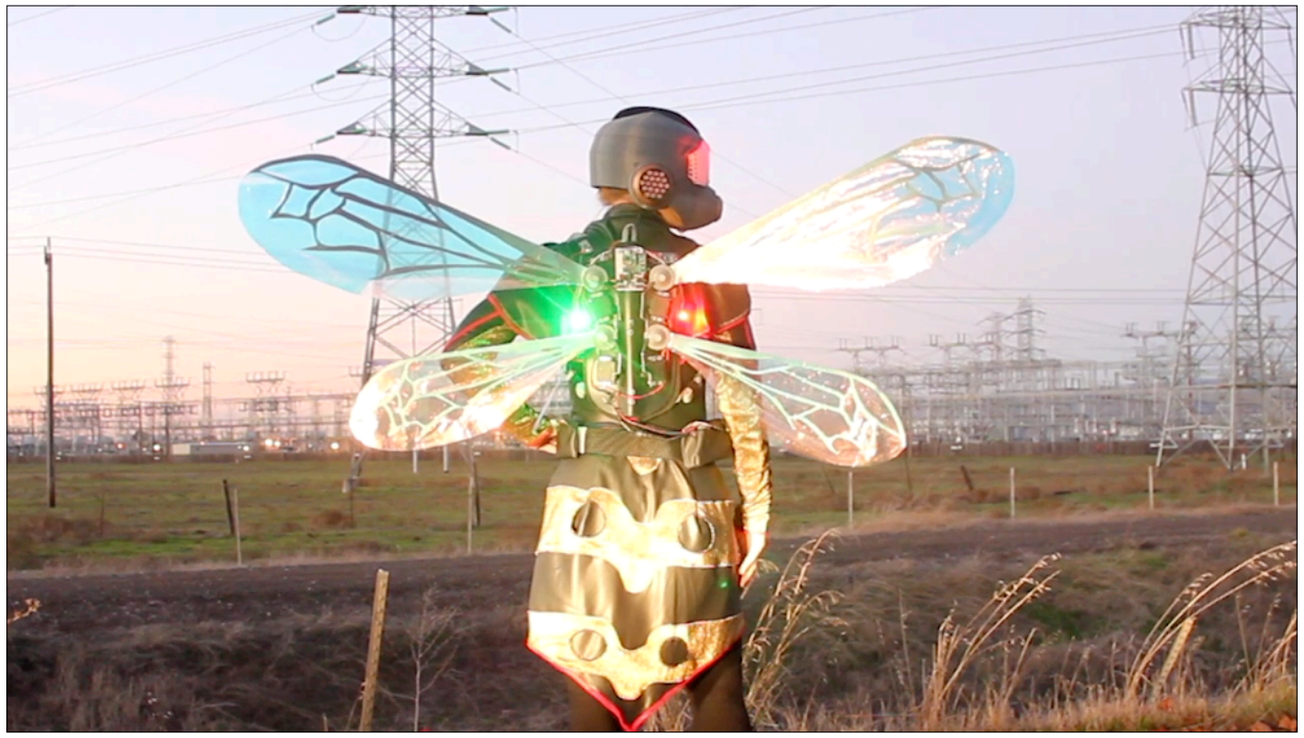


…animates these eyes and makes them blink.

A couple years later, I revisited this project, using better displays…

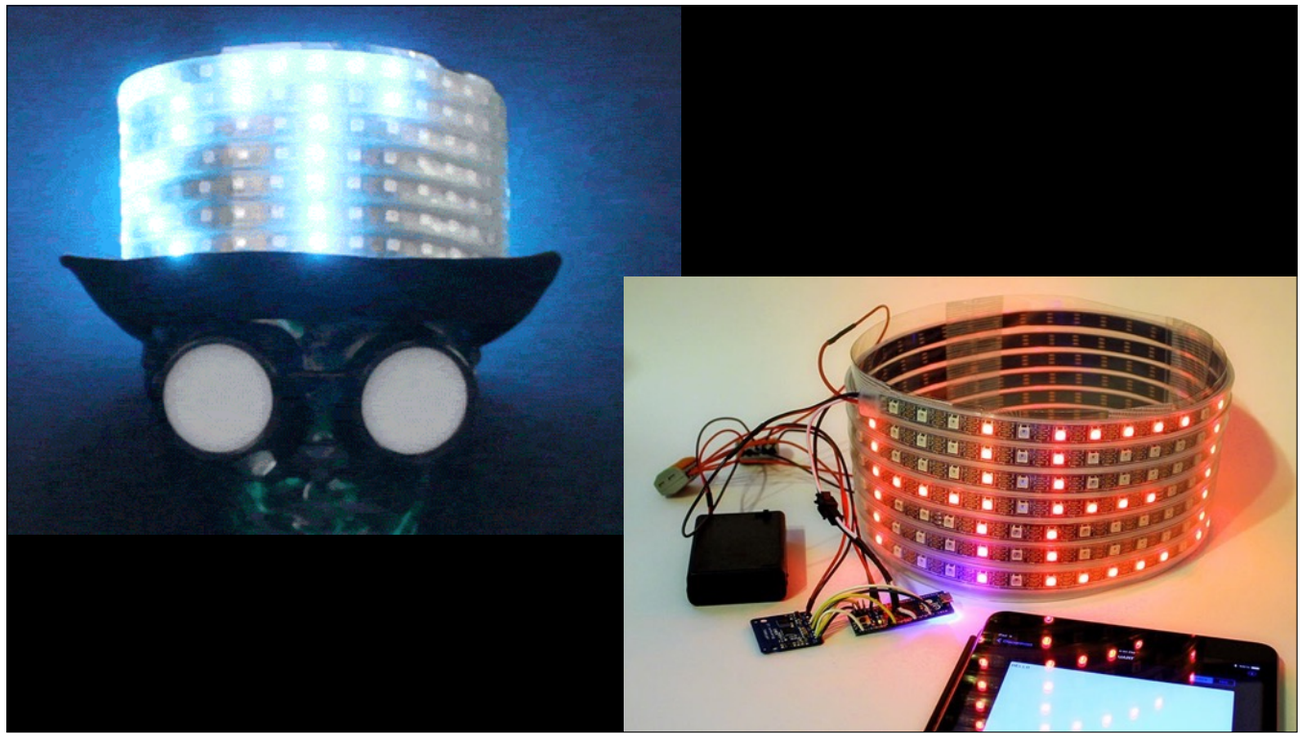
The images are sharper, but the logic that decides where to look and when to blink…that's exactly the same code from the earlier project! It's very rare that you need to start from scratch.

And Arduinos are good for more than just blinking…

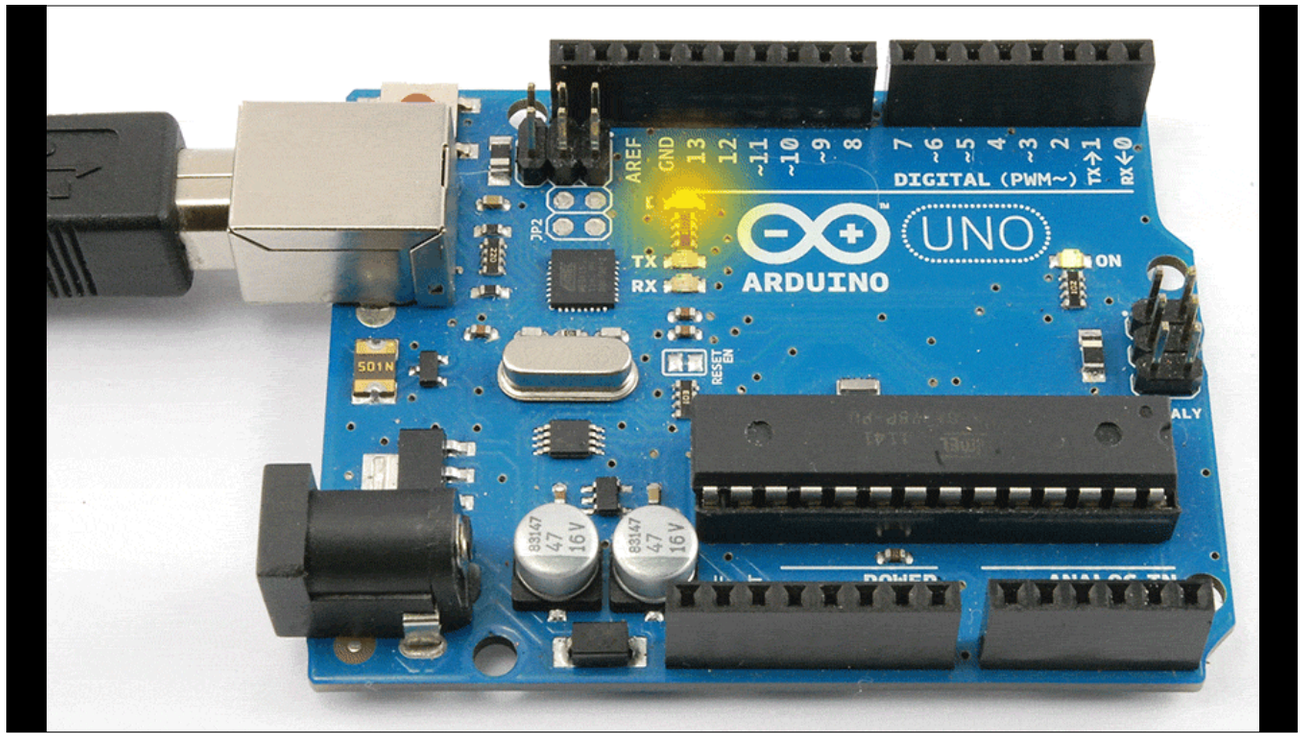


These animatronic wings, for example. There’s an Arduino board controlling four RC hobby servos.

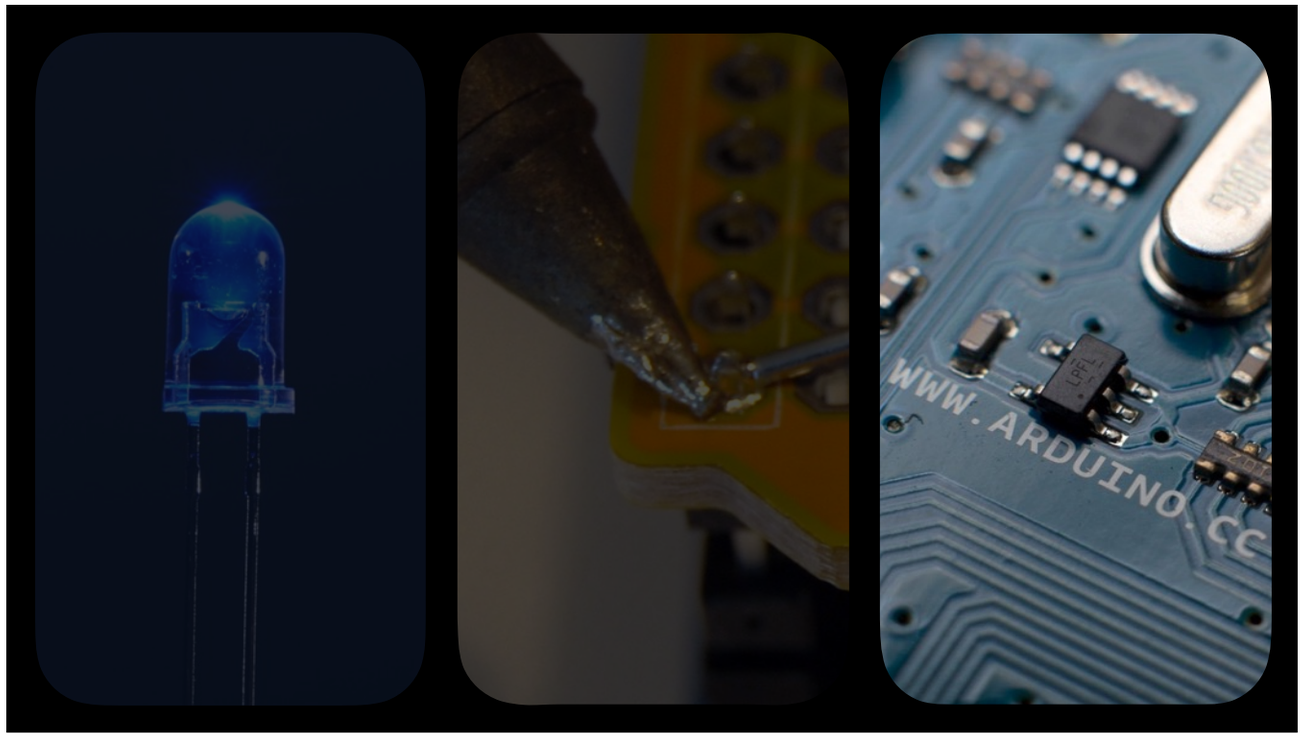
A laser harp! An avant-garde musical instrument. This uses an Arduino as well.



Or this hat that connects to your phone or tablet over Bluetooth and displays scrolling messages. Again, Arduino.



All of that starts with the simple blinking LED.



And that’s microcontollers in a nutshell. Any questions on this part?

Resources

Before we wrap this up, I can offer some places to go for more information…

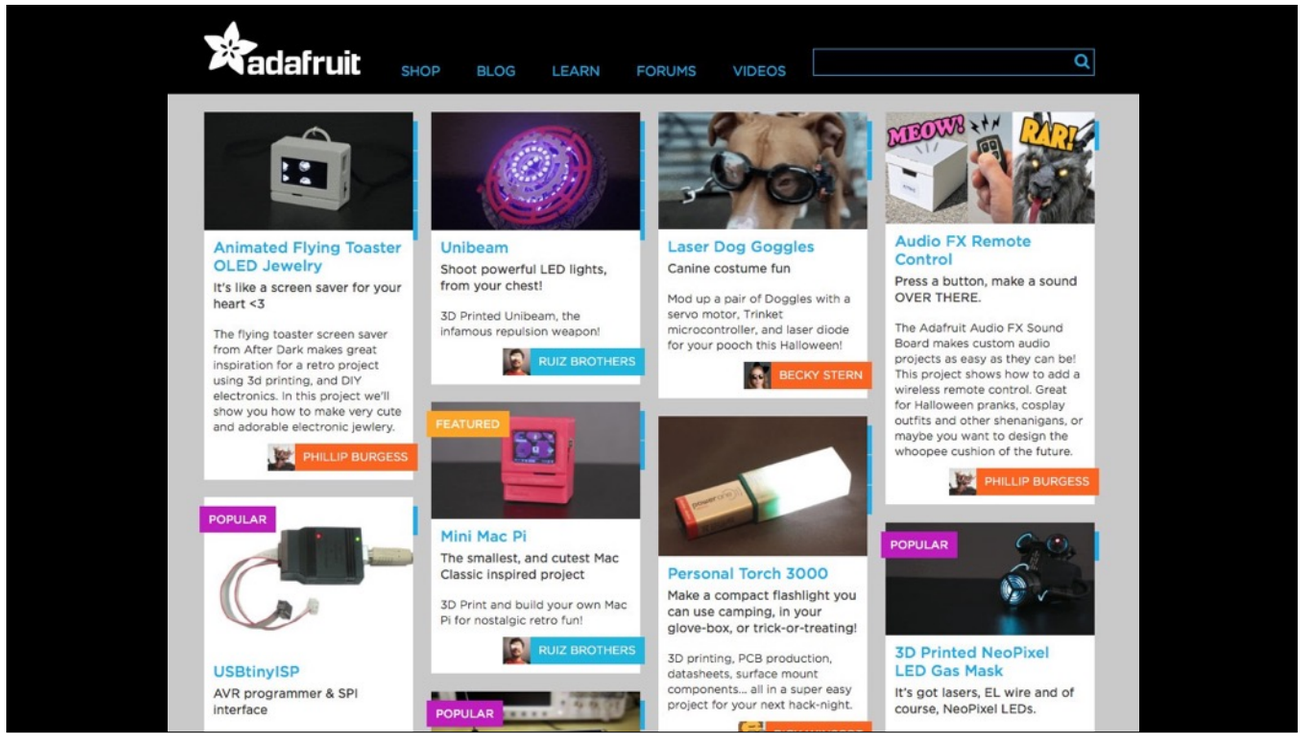
Parts, Tools, Forums & Projects:

Adafruit.com

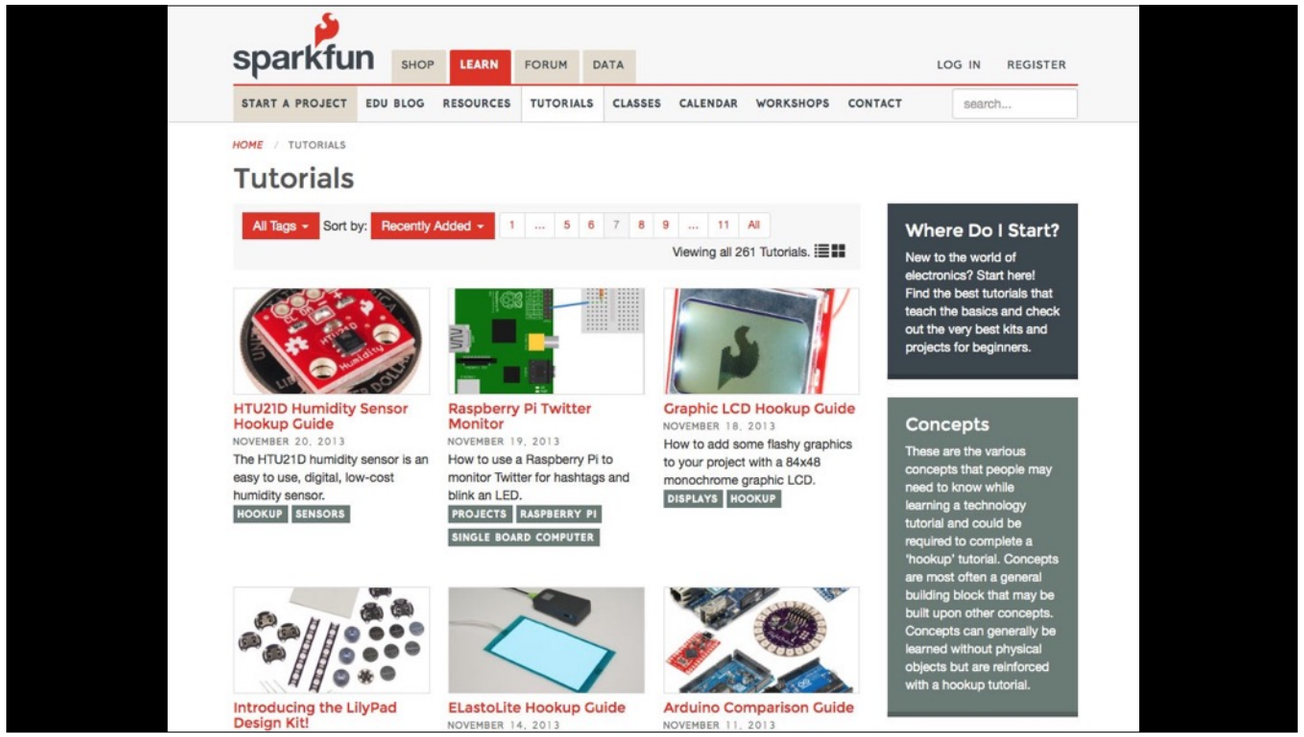
SparkFun.com

If you're looking for components or tools, have technical questions, or want some starter project ideas, both of these sites are great resources.

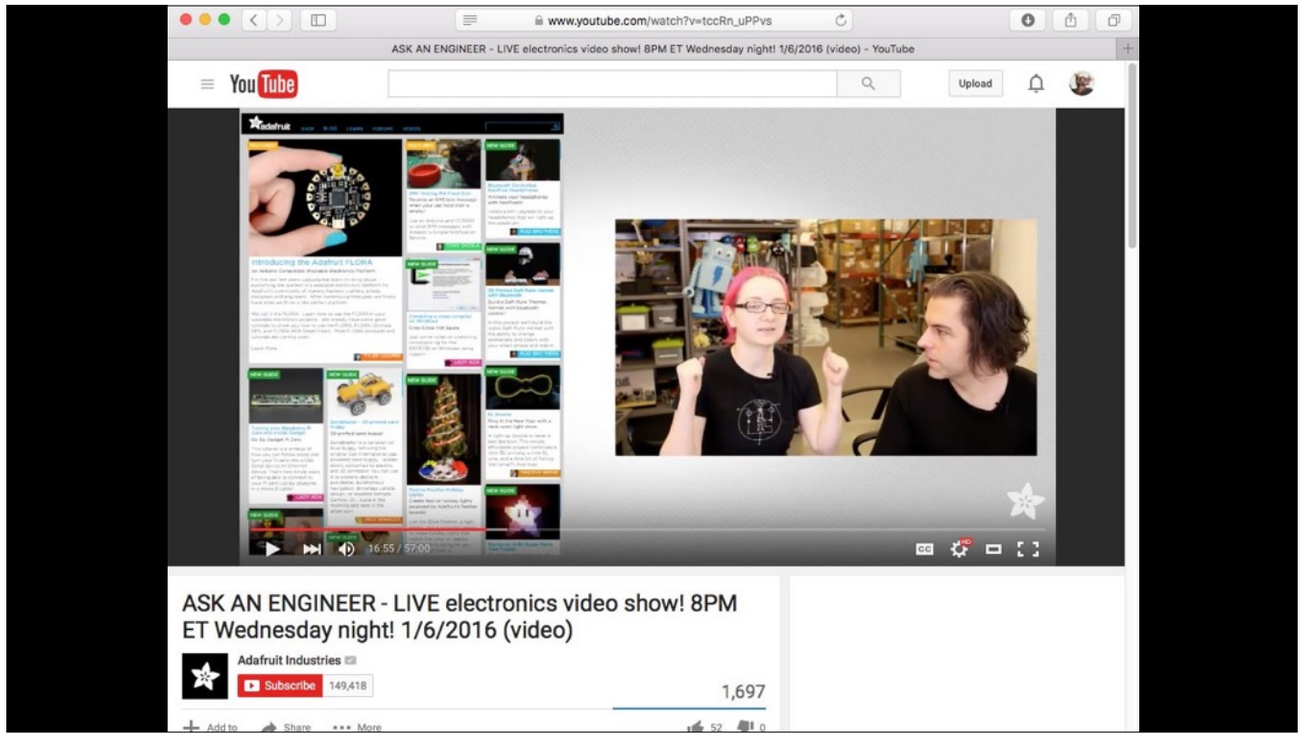
Naturally, I'm slightly biased toward one of them…but seriously, if Adafruit doesn't have what you're looking for, check out SparkFun. They're good people and share a lot of the same values.



Tack “learn dot” on the start of the URL — learn.adafruit.com — and you’ll find hundreds of projects and lessons, all free!



Likewise, learn.sparkfun.com.



If building up your toolkit or parts collection…every Wednesday, Adafruit does a live streaming show on YouTube, and there's a 10% discount code that's good from 8pm to midnight (eastern time) that day.

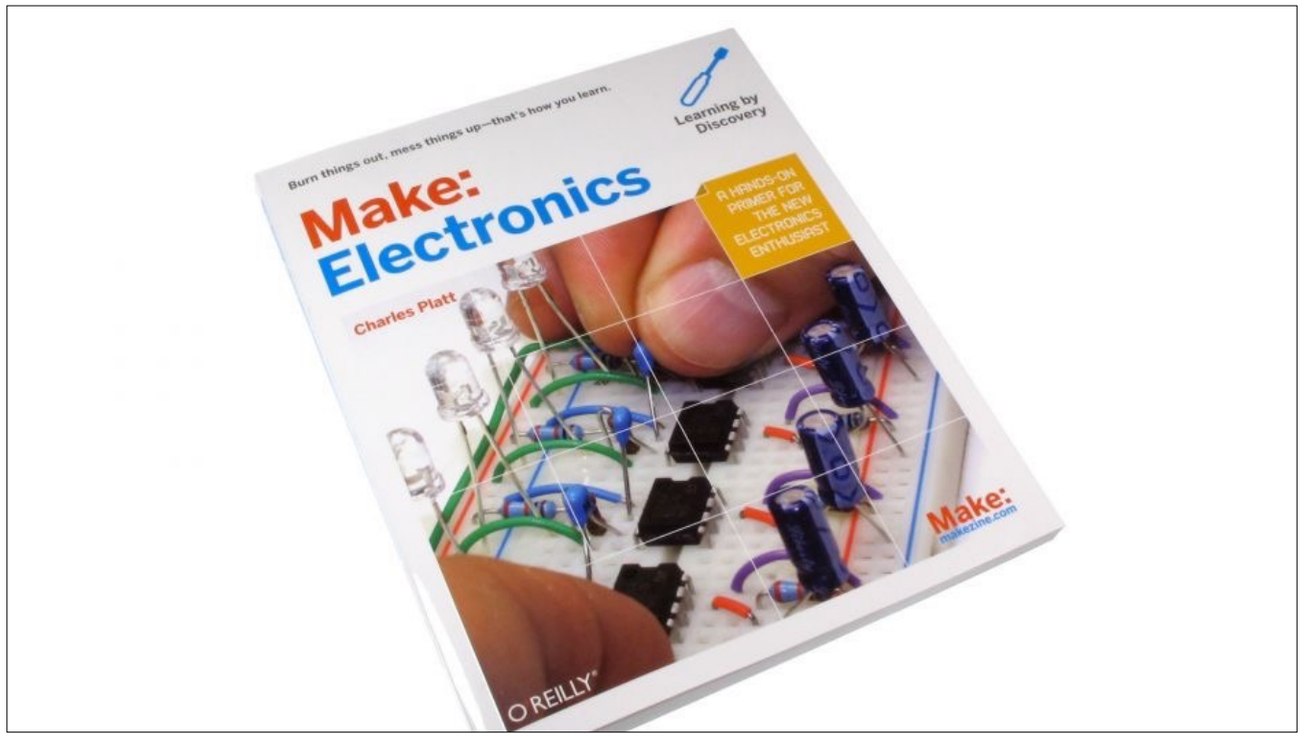
Components:

Digi-Key.com

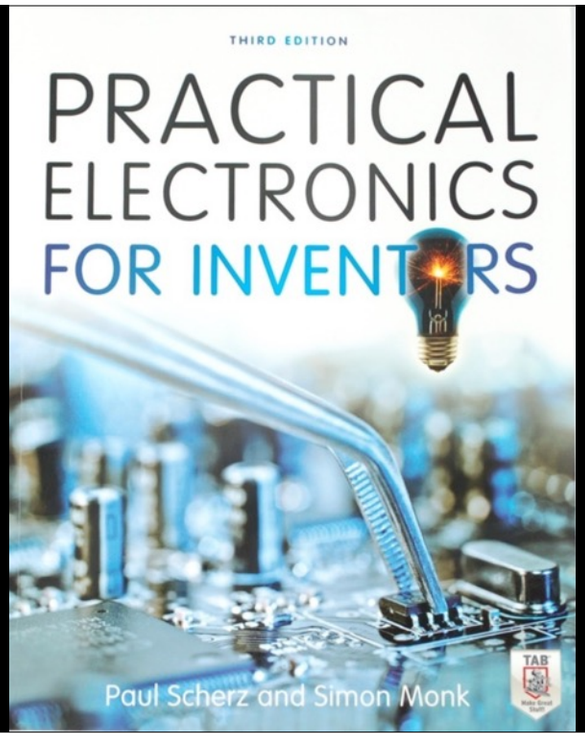
mouser.com

If you need just some small parts — LEDs, resistors, buttons — DigiKey and Mouser are good sources.

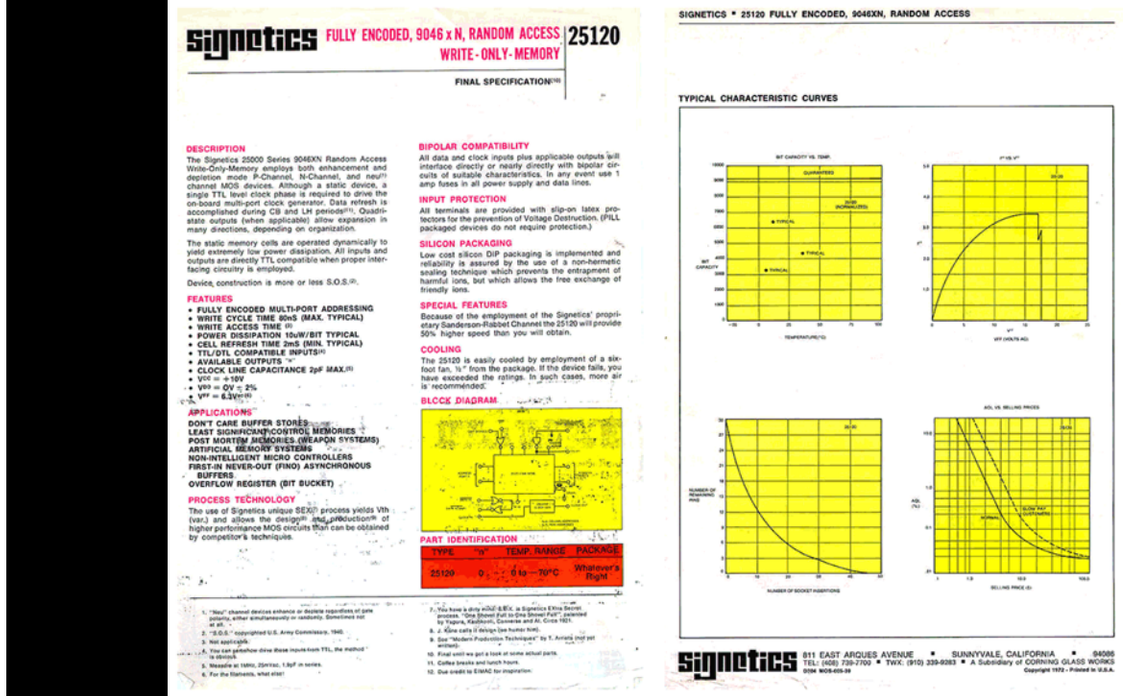
Important tip: don’t buy small parts onesy-twosey. Go ahead and get a handful or a dozen or a hundred. You’re going to spend more on shipping than on the components…small parts are super cheap…and I can practically guarantee that if you’re using a particular component today, some kind of LED or button, you’ll be using the same thing again in a future project. It doesn’t cost much more to build up a stash…and you can trade with other electronics friends!



Books! I like this one, Make: Electronics by Charles Platt. This is one you can read front-to-back and follow along with the projects.



Practical Electronics for Inventors by Simon Monk is a more “encyclopedic” reference — you won’t be reading this front-to-back, but if you need to study a particular topic, it’s got you covered.



signetics FULLY ENCODED, 9046 x N, RANDOM ACCESS WRITE-ONLY-MEMORY 25120

FINAL SPECIFICATION[1][10]

DESCRIPTION

The Signetics 25000 Series 9046XN Random Access Write-Only-Memory employs both enhancement and depletion mode P-Channel, N-Channel, and neu[11] channel MOS devices. Although a static device, a single TTL level clock phase is required to drive the on-board multi-port clock generator. Data refresh is accomplished during CB and LH periods[11]. Quadri-state outputs (when applicable) allow expansion in many directions, depending on organization.

The static memory cells are operated dynamically to yield extremely low power dissipation. All inputs and outputs are directly TTL compatible when proper interfacing circuitry is employed.

Device construction is more or less S.O.S.[2].

FEATURES

- FULLY ENCODED MULTI-PORT ADDRESSING
- WRITE CYCLE TIME 80nS (MAX. TYPICAL)
- WRITE ACCESS TIME [3]
- POWER DISSIPATION 10uW/BIT TYPICAL
- CELL REFRESH TIME 2mS (MIN. TYPICAL)
- TTL/DTL COMPATIBLE INPUTS[4]
- AVAILABLE OUTPUTS "n"
- CLOCK LINE CAPACITANCE 2pF MAX.[5]
- VDD = +10V
- VSS = 0V ± 2%
- VFF = 6.3V[6][9]

APPLICATIONS

DON'T CARE BUFFER STORES
LEAST SIGNIFICANT-CONTROL MEMORIES
POST MORTEM MEMORIES (WEAPON SYSTEMS)
ARTIFICIAL MEMORY SYSTEMS
NON-INTELLIGENT MICRO CONTROLLERS
FIRST-IN NEVER-OUT (FINO) ASYNCHRONOUS BUFFERS
OVERFLOW REGISTER (BIT BUCKET)

PROCESS TECHNOLOGY

The use of Signetics unique SEX[7] process yields Vth (var.) and allows the design[8] and production[9] of higher performance MOS circuits than can be obtained by competitor's techniques.

BIPOLAR COMPATIBILITY

All data and clock inputs plus applicable outputs will interface directly or nearly directly with bipolar circuits of suitable characteristics. In any event use 1 amp fuses in all power supply and data lines.

INPUT PROTECTION

All terminals are provided with slip-on latex protectors for the prevention of Voltage Destruction. (FILL packaged devices do not require protection.)

SILICON PACKAGING

Low cost silicon DIP packaging is implemented and reliability is assured by the use of a non-hermetic sealing technique which prevents the entrapment of harmful ions, but which allows the free exchange of friendly ions.

SPECIAL FEATURES

Because of the employment of the Signetics' proprietary Sanderson-Rabbet Channel the 25120 will provide 50% higher speed than you will obtain.

COOLING

The 25120 is easily cooled by employment of a six-foot fan, ½" from the package. If the device fails, you have exceeded the ratings. In such cases, more air is recommended.

BLOCK DIAGRAM

PART IDENTIFICATION

TYPE	"n"	TEMP. RANGE	PACKAGE
25120	0 - 6	0 to +70°C	Whatever's Right

1. "Neu" channel devices enhance or deplete regardless of gate polarity, either simultaneously or randomly. Sometimes not at all.
2. "S.O.S." copyrighted U.S. Army Commissary, 1940.
3. Not applicable.
4. You can somehow drive these inputs from TTL, the method is obvious.
5. Measure at 1MHz, 25mVac, 1.5pF in series.
6. For the filaments, what else!
7. You have a dirty mind. S.E.X. is Signetics Ethra Secret process. "One Shovel Full to One Shovel Full", patented by Yagura, Kashiyoti, Cawwise and Al. Circa 1921.
8. J. Kline calls it design (see humor link).
9. See "Modern Production Techniques" by T. Arrieta (not yet written).
10. Final until we get a look at some actual parts.
11. Coffee breaks and lunch hours.
12. Due credit to ENIAC for inspiration.

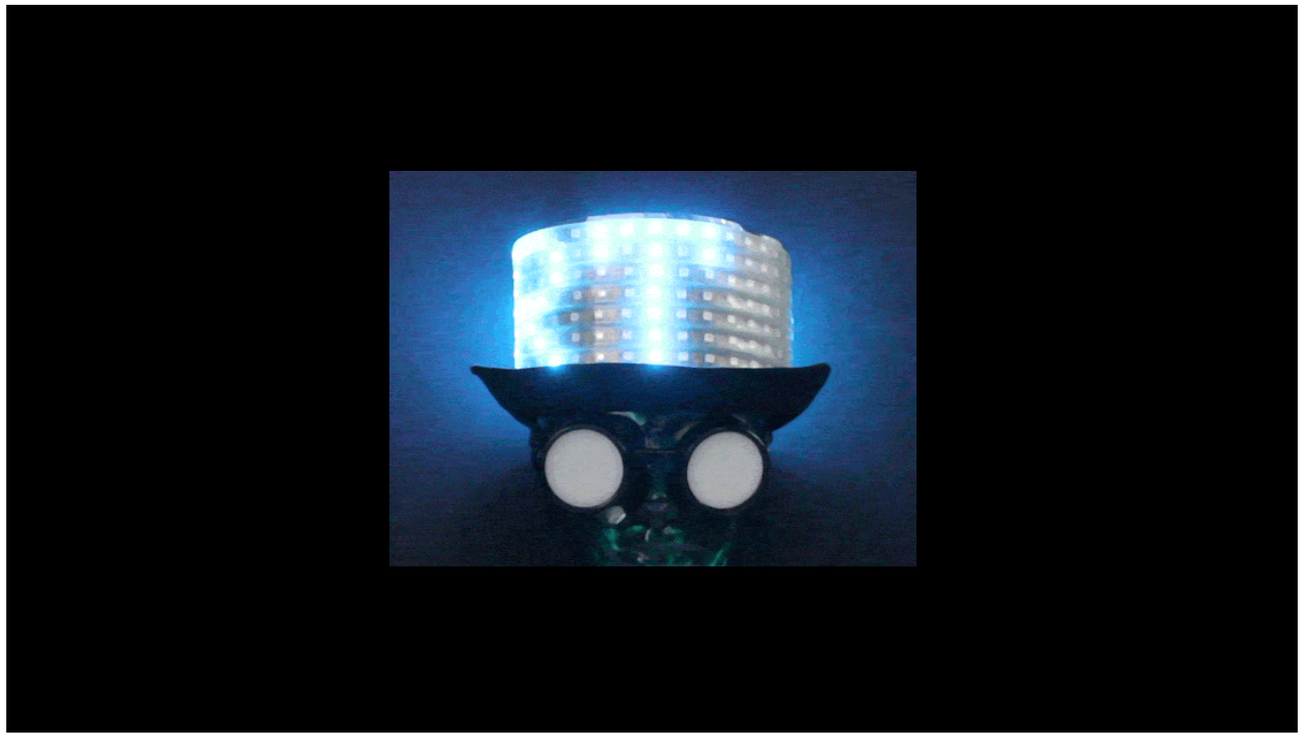
TYPICAL CHARACTERISTIC CURVES

signetics 811 EAST ARQUES AVENUE • SUNNYVALE, CALIFORNIA • 94086
TEL: (408) 739-7700 • TWX: (910) 339-9283 • A Subsidiary of CORNING GLASS WORKS

DSM 9025-025-39

And for any component you buy, down to the tiniest individual LED, the manufacturers of these components publish documents called DATASHEETS that spell out every little detail.

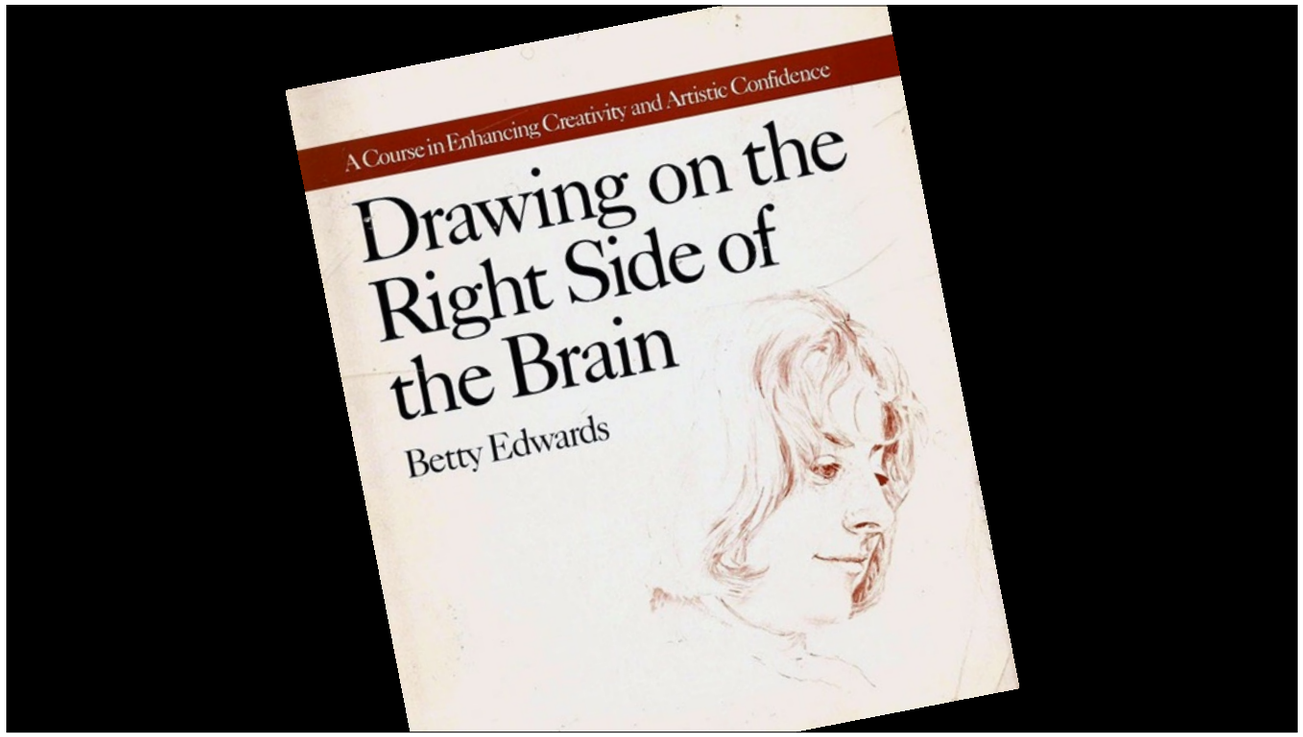
Nowadays, these datasheets can be downloaded from the web for FREE. Usually in PDF format. You can load up your tablet or book reader with all this information. It's an absolute gold mine!



If we’ve got a moment, one more book I’ll recommend…

Up to this point, I’ve mostly been explaining technical things to art people.

For the technical people in the audience, I’d like to give you a homework assignment too…



Go on Amazon and look up this book, “Drawing on the Right Side of the Brain” by Betty Edwards. Buy yourself a secondhand copy. In fact, buy the oldest, most dog-eared, earliest edition you can find…reason being that each successive edition gets a little more wordy, and strays a little further from the key point she’s trying to make…

Non-artists often see artists as having a special gift, almost a magical power. But too often, we confuse that gift as TECHNIQUE…how one holds a pencil, or moves their arm. The point Edwards makes is that we see the world through a perceptual filter. This is really a psychology book. People become stick figures, eyes become football shapes, cat ears become triangles. It’s a convenient shorthand for understanding the world, but it’s not the true shape of the world. The artist’s gift is really one of OBSERVATION, and she shows its a skill we can all learn. Follow along with the exercises, it’s pretty amazing.

So why am I mentioning all this?

Here’s an example from animatronics, but it could be anything technical.

This clown over here. I can’t fault it technically. Moving ANYTHING around is one of the most difficult tasks in electronics! But we clearly know it’s a robot, because it’s all abrupt motions and straight lines. Life just doesn’t move like that.

Now the werewolf…to be fair, this is some incredibly high-end movie animatronics…but the point here is, this is what happens when artists and engineers are speaking the same language. There’s a subtlety of movement here, with secondary and tertiary motions, that makes you wonder for a moment if this pile of motors and aluminum and rubber is alive. Creating something like this is only possible when you bypass that perceptual filter and objectively see the shapes of nature.

Q&A

Final round of questions.

Also, door prizes!