From the makers of HackSpace

# WEARABLE TECH PROJECTS

by Sophy Wong

30 PROJECTS TO MAKE, SEW, AND WEAR!

# Welcome to WEARABLE TECH PROJECTS

When you think of the word technology, what does it conjure up? Computers? Mobile phones? Bluetooth headsets? These are all technology, but they don't define it. We're makers – we can make tech in whatever form we want, and we want technology we can wear, technology that looks great, and technology that we can use to show off.

We want technology we can wear, technology that looks great, and technology that we can use to show off

We've gathered up the best bits of wearable technology from the first two years of HackSpace magazine and gathered them into one place. Here, you'll find projects for sewing, 3D printing, laser cutting and programming, so whatever medium you choose, you'll find inspiration and help getting started.

Let's get stuck in and let's make technology work the way we want it to. Turn the page and let's make the world a bit brighter and more colourful.

**BEN EVERARD**
**Editor** ben.everard@raspberrypi.org

Got a comment, question or thought about HackSpace magazine?

get in touch at **hsmag.cc/hello**

**GET IN TOUCH**
hackspace@raspberrypi.org
hackspacemag
hackspacemag

**ONLINE**
hsmag.cc

Available on the App Store
GET IT ON Google Play

**EDITORIAL**
**Editor**
Ben Everard
ben.everard@raspberrypi.org

**Features Editor**
Andrew Gregory
andrew.gregory@raspberrypi.org

**Book Production Editor**
Phil King

**Writer**
Sophy Wong

**DESIGN**
**Critical Media**
criticalmedia.co.uk

**Head of Design**
Lee Allen

**Designers**
Harriet Knight, Sam Ribbits

**ADDITIONAL CONTRIBUTORS**
Helen Leigh, Poppy Mosbacher, Ben Muller, Angela Sheehan, Kitty Yeung

**PUBLISHING**
**Publishing Director**
Russell Barnes
russell@raspberrypi.org

**Advertising**
Charlie Milligan
charlotte.milligan@raspberrypi.org
+44 7725 368 887

**DISTRIBUTION**
Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

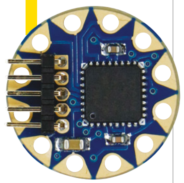
This book is printed on paper sourced from sustainable forests. The printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Wearable Tech Projects is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised. Except where otherwise noted, content in this book is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISBN: 978-1-912047-63-5.

# Contents

70 NeoPixel mask

TinyLily Mini 152



08 Wearable tech

82

Wearable game controller

**DOWNLOADABLE ASSETS**

**To download code files, designs, and sounds for the projects in this book, visit: hsmag.cc/book2-assets**

# GETTING STARTED

These beginner-friendly projects are ideal for getting started with wearables

08

# WEARABLE TECH

## DIVE INTO THE WORLD OF WEARABLES

**What do you think of when you hear the words 'wearable electronics'?** Maybe you think of Snapchat's Spectacles, the pop-coloured sunglasses with a built-in camera. If you're a cosplayer, you might think of a glowing, chest-mounted 'arc reactor' à la Tony Stark. Wearables can be fitness trackers, virtual reality headsets, spacesuits, cosplay, and more. If it needs a battery and you can wear it, it's a piece of wearable electronics!

As we speed toward ever tinier technology, it is now possible to build the kind of wearable devices that we used to only dream about in comic books. With access to components like sewable microcontrollers and flexible LED strips, tinkering with wearables has never been easier. A little foundational knowledge will set you up for success with your first projects, so let's dive into the concepts, tools, and best practices for wearables. →

## GETTING STARTED

Above
This cute and cuddly Jawa costume features glowing eyes

### LED HAT

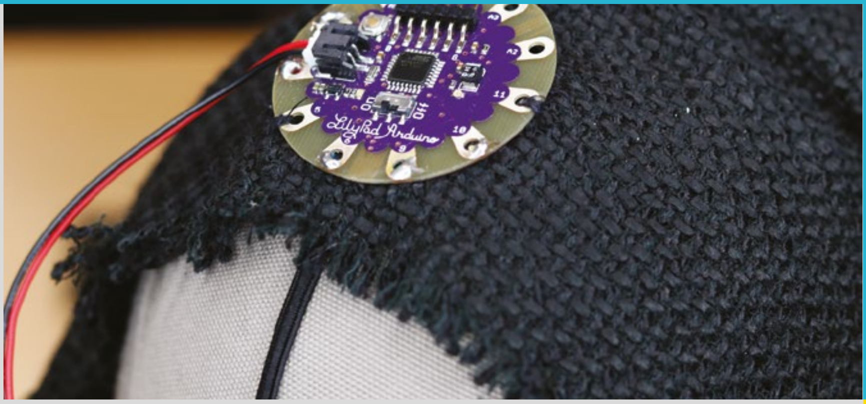
The best way to learn any subject is to start small. Turn to page 20 for a quick and easy introduction to sewing circuits.

# BUILDING FOR THE HUMAN BODY

**Your idea might be as complex as a holographic computer on your head, or as simple as an LED sewn into a wristband.** But because they all go on the human body, all wearables face a shared set of challenges. Designers often solve these challenges in similar ways, and you may notice underlying similarities between wearables that seem very different at first glance.

One of the biggest challenges for wearable designers is how to apply the rigid, flat materials of electronic components to the fleshy, round surfaces of the human body. Wearable designs generally avoid sharp edges and pointy corners, for the comfort and safety of the wearer. You can see this idea at work in both the rounded corners of Microsoft's HoloLens and the circular form factor of the LilyPad Arduino. Rounded corners also protect soft materials like fabric from snagging during movement. Keep this in mind when selecting materials and components for your wearable projects.

Wearables are often circular or tubular in form. This is because the human body is basically made up of cylindrical shapes – you can think of your arms, legs, torso, and even your fingers, as cylinders. Fitness trackers, twinkle LED skirts, and VR headsets all loop around the body in a circle, tube, or arc. In this sense, the human body is a perfect fit for electronic circuits, which are also loops. The challenge is how to make an electrical loop that can open and close, stretch, or bend around the human body. Think about this when looking at consumer wearable gadgets – how do they solve this challenge? →

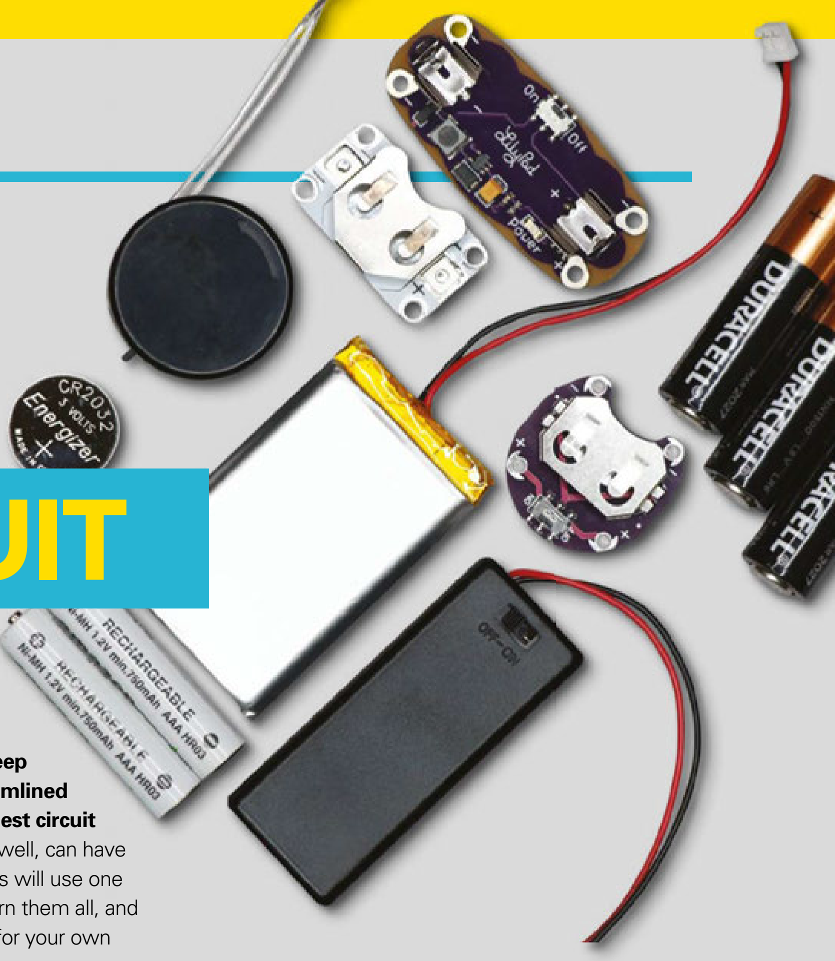


**Above** The Jawa costume uses a LilyPad Arduino to power the glowing eyes

## FIVE TIPS FOR EVERY WEARABLE PROJECT

**There's a lot to consider when designing your wearable project. Whatever form your project takes, these five tips will help you make something you'll enjoy wearing when you've finished building it:**

- Don't forget the on/off switch. If at all possible, include an on/off switch in your design, and make it accessible during wear. Resist the temptation to rely on removing and inserting batteries for power control. Ideally, you can switch your project off quicker than someone else can say, "Ouch, those lights are bright!"
- Put the battery where you can reach it. Like the power switch, keep your battery accessible. Often your battery will be the heaviest, bulkiest part of your project, and you'll want to hide it, but don't bury it too deeply. If you're sporting your wearable for more than a few hours, chances are you'll need to change the battery at some point. Bonus points if you can change it in public!
- Figure out how to clean it. Protect your hard work by ensuring that your project is cleanable in a practical manner. Some wearable components are actually hand-washable, but wash carefully and follow manufacturer's instructions. Another option is to make your electronic circuit removable using hook-and-loop fasteners or snaps, and launder the garment as you normally would. You can also wear a 'laundry layer', something that can be easily washed, underneath your project. For rigid accessories like headsets and helmets, spot-clean the areas that touch skin with alcohol-free baby wipes.
- Solder big projects. Sewn circuits are a great way to get started with wearable electronics, but be prepared for your imagination to outgrow the limitations of conductive thread. The resistance of conductive thread makes it unsuitable for long circuits, and complexity is easier to manage with insulated wire in different colours.
- Think about the wearer. When working with wearables, keep the wearer in mind. Make comfort your goal, and don't build anything that puts their safety at risk. Keep circuits and batteries away from skin, and don't impede vision or movement. Your wearable will look much better if the person wearing it is smiling!



# BUILDING YOUR CIRCUIT

**Wearable circuits don't necessarily have to be complex. Keep your project streamlined and use the simplest circuit you can.** A simple circuit, implemented well, can have a big impact. Most tutorials for wearables will use one of these three common techniques. Learn them all, and you'll be able to choose the best option for your own wearable designs.

Choosing a battery for your project is a balance between size and power

## PREMADE CIRCUITS

A quick and easy way to get started is to use a premade circuit. Instead of building the circuit from scratch, you can skip ahead to figuring out how to put it into your garment. Fairy lights are great premade circuits, as most come with a simple battery pack and a built-in power switch. Another type of premade circuit is an all-in-one board like Adafruit's Circuit Playground Express, a microcontroller with built-in lights and sensors.

When using premade circuits, your main challenge will be how to embed them into your project. For soft garments, get creative with snaps or hook-and-loop fasteners so that the electronics can be removed for washing. If your sewing machine has a buttonhole function, adding buttonholes to a garment is an easy way to make passthroughs for wires. For rigid accessories like helmets and headsets, use hot glue or strong adhesive tape to attach your circuit.

## SEWN CIRCUITS

Sewn circuits, also known as e-textiles, are also a great place to start. Sewn circuits use conductive thread for electrical connections, no soldering necessary! To build your circuit, you simply sew from one component to the next. There are many types of sewable components available, including microcontrollers, sensors, LEDs, and more. Sewable components have large, open holes for making electrical connections with conductive thread, and some are even washable.

You may already have the hand-sewing tools you need for sewn circuits: large-eye needles, scissors, and a thimble. You'll also need conductive thread and sewable electronic components for your circuit. Conductive thread is thicker than regular thread, and a dab of Fray Check or superglue will help keep knots tied.

Because conductive thread is not insulated, it can be difficult to avoid short circuits, and you will need to insulate your sewn circuit when it's complete. Cover the thread lines with puff paint or strips of fusible interfacing. This usually works well on the inside of a soft project, but think about the outside too – a long run of conductive thread can fold back on itself and cause a short circuit. Because of the resistance factor of conductive thread, insulated wire is a better choice for long circuits.

Despite the challenges of building large projects with sewn circuits, great complexity can be achieved with clever engineering and imagination. Masters of sewn circuits embrace the constraints of conductive thread and turn circuits themselves into works of art. However, if sewn circuits are your entry point to electronics, it's likely that the limits of conductive thread will drive you to learn to solder. We love that idea!

## SOLDERED CIRCUITS

If you're new to it, learning how to solder may seem daunting, but it's fun and easy once you learn the basics. A quick lesson from a makerspace or an experienced friend can get you started. Then it's a matter of practice, and investing in the tools: a soldering iron, metal solder, wire strippers, and wire for your project. Silicone-coated stranded wire is great for wearables; it's soft, flexible, and can be

**Above** There are lots of power options to choose from

incredibly thin. You can still use sewable components designed for conductive thread; just solder your wires into the holes. Soldering opens the door to the wider world of electronics, and you'll be able to make use of components not specifically designed for wearable applications.

Even if you already know how to solder, there are a few things to keep in mind when soldering for wearables. Soldered joints aren't flexible, and will break if flexed repeatedly during wear. A dab of hot glue can reinforce a soldered connection and make it last longer. Fabrics stretch, but wires don't – make sure there is enough length in your wires to allow for movement. And while your wires may be insulated, you'll still need to keep soldered connections and components away from moisture and skin. Coat them with a clear acrylic spray or clear nail polish when your build is complete.

## BATTERIES

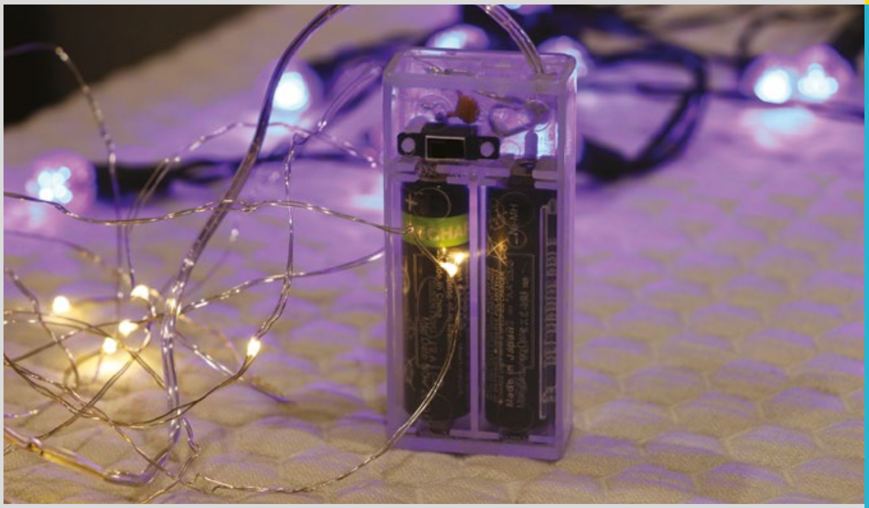
Choosing a battery for your project is a balance between size and power. For wearables, you'll want the smallest battery you can get away with, given the power requirements of your project. To estimate your power needs, check the datasheets for each component in your circuit. Find the maximum current draw for each, and add them together. It's better to provide more amps than not enough, so choose a battery that meets or exceeds this total. If you're just starting out, stick with the batteries recommended in tutorials and example projects. As you build more and become familiar with components, you'll get a better sense of what batteries to choose.

## COIN CELL BATTERIES

These are your tiniest option for battery power. Coin cells are great for low-power projects like powering a few LEDs, and maybe even a tiny microcontroller. Battery holders often come with a built-in on/off switch, and come in sewable form. Coin cell batteries are low-capacity and don't last long, but they're small enough to carry a spare set.

## ALKALINE AND NI-MH RECHARGEABLE BATTERIES

Like coin cells, these are easy to find and you already know how to use them. But unlike coin cells, rechargeable versions are available, which is great for the environment and your wallet. Three or four AAs should work for most wearable microcontroller projects. On the downside, AA and AAA batteries are heavy in multiples, and hiding a bulky battery pack is tricky in a small project.



## LIPO BATTERIES

Because they are small, rechargeable, and powerful enough to drive complex wearable projects, LiPo batteries are the battery of choice for many wearables. However, they are also more delicate than other types of batteries and require careful handling for safe use. Never puncture, compress, or expose them to heat. Don't place them directly against skin, and don't store them in your project. If your project is made of fabric, remove the LiPo battery before recharging it.

## USB POWER BANKS

For projects that need a hell of a lot of juice, a USB power bank is a great choice. They are high-capacity, safer, and more durable than LiPo batteries, and most come with some kind of on/off switch. Often, the power bank's specifications are printed right on the case, check them against the power needs of your project. USB power banks come in many shapes and sizes, but tend to be heavy. →

**Above**
Wearables projects don't have to start with specialised kit

**Below**
Conductive thread blends invisibly with fabric, but is difficult to work into complex circuits

# MICROCONTROLLERS FOR WEARABLES

For wearable tutorials, coding lessons, and more, check out:

- learn.adafruit.com
- learn.sparkfun.com
- wearabletutorials.com
- kobakant.at/DIY
- instructables.com/howto/wearables

**Arduinos and other microcontrollers are basically tiny computers that let you add programmability to your project.** With a microcontroller, you can incorporate light animations, sensors, motors, sounds, and more, all driven by code. Wearable microcontrollers are often circular in form, and can be both sewn and soldered to.

There are many wearable microcontrollers to choose from, and scores of wearable components that you can connect to. Let's take a look at some great microcontrollers specifically designed for wearables:

## 1 LILYPAD ARDUINO

LilyPad is the original sewable microcontroller, invented by Leah Buechley and produced by SparkFun. This is the board that ignited the DIY wearable movement when it was released in 2007. Today, there are other boards based on Buechley's iconic design, and even the LilyPad itself comes in several different flavours. The LilyPad line also includes sewable LEDs, sensors, buttons, switches, battery holders, and more.

This is the board that ignited the DIY wearable movement when it was released in 2007

## 2 ADAFRUIT FLORA

Inspired by the original LilyPad Arduino, the Flora wearable platform by Adafruit is powerful, easy to use, and supported by a massive library of tutorials created by Adafruit and its community. Work through a few Adafruit tutorials and you'll be designing your own wearable projects in no time. The Flora line includes sewable versions of powerful components like GPS, a Bluetooth module, and NeoPixels – Adafruit's highly addictive, individually addressable LEDs.

## 3 ADAFRUIT GEMMA

Gemma is a 1-inch diameter version of Flora that's perfect for smaller projects that only require a few inputs and outputs. It's great for beginners, and can feel less intimidating than a bigger board with lots of bells and whistles. Despite its small size, Gemma still has convenient features like a built-in on/off switch, a JST battery connector, and micro USB for programming.

## 4 CIRCUIT PLAYGROUND

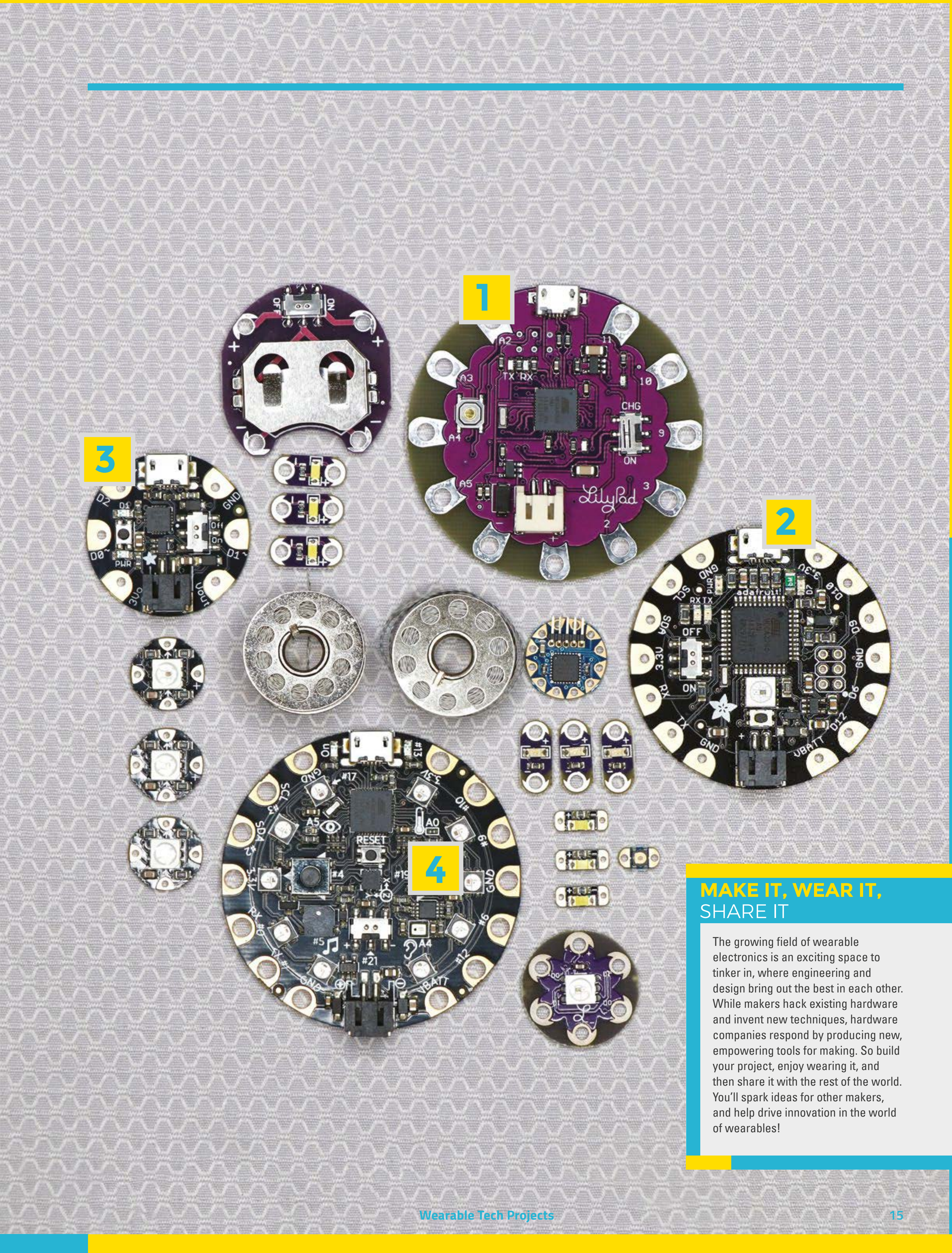
This board by Adafruit is packed full of features and brings more to your projects than just a microcontroller. There are ten multicolour LEDs and a speaker for output. For input, there's a temperature sensor, microphone, and two buttons. Along with this, it features all the usual digital and analogue input and output. There's an Express version that comes with a powerful ARM processor that we like so much, we give one away to every 12-month HackSpace subscriber. Turn to page 90 for details.

## ALSO CONSIDER: STITCHKIT

StitchKit is a new wearable microcontroller designed specifically for fashion tech and wearables by MakeFashion. Based on its makers' experience running fashion shows, the StitchKit is designed to combine durability with ease of use for people who are new to working with hardware. It's on sale on the Indiegogo crowdfunding site: **hsmag.cc/HdVLiP**. →

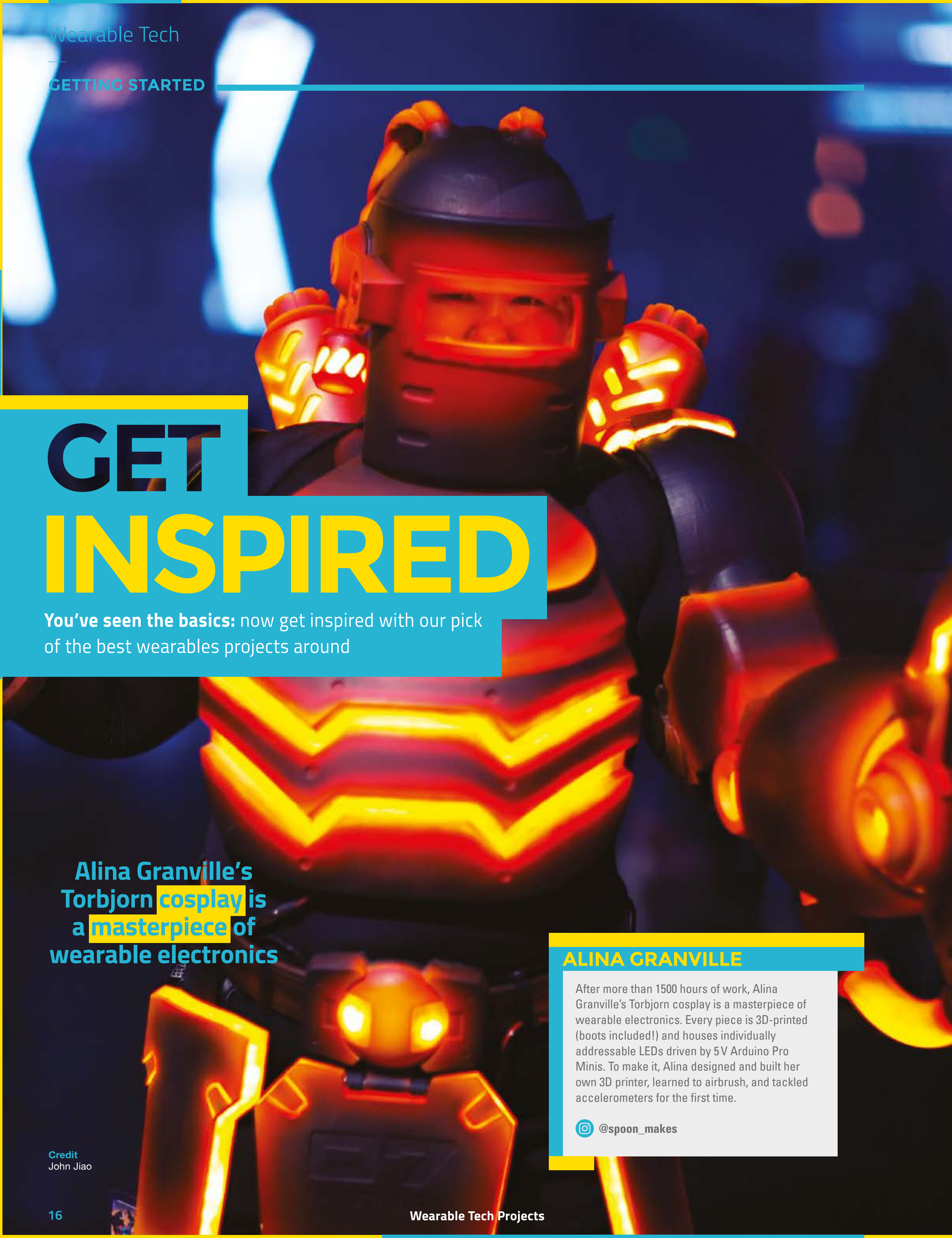
## CODING HELP

If you're new to code, fear not – tutorials and example projects abound on the internet and it takes minutes to get up and running with example sketches in the Arduino software. Start with simple code provided in tutorials and you'll learn to modify it to suit your needs and do more. There are also visual programming aids that make coding simpler: try MakeCode by Microsoft, or XOD. When you're ready to write your own code, check out CircuitPython, a derivative of the programming language MicroPython, released and supported by Adafruit. CircuitPython aims to make it easy for complete beginners to write code for their microcontrollers.



## MAKE IT, WEAR IT, SHARE IT

The growing field of wearable electronics is an exciting space to tinker in, where engineering and design bring out the best in each other. While makers hack existing hardware and invent new techniques, hardware companies respond by producing new, empowering tools for making. So build your project, enjoy wearing it, and then share it with the rest of the world. You'll spark ideas for other makers, and help drive innovation in the world of wearables!



# GET INSPIRED

**You've seen the basics:** now get inspired with our pick of the best wearables projects around

Alina Granville's Torbjorn cosplay is a masterpiece of wearable electronics

## ALINA GRANVILLE

After more than 1500 hours of work, Alina Granville's Torbjorn cosplay is a masterpiece of wearable electronics. Every piece is 3D-printed (boots included!) and houses individually addressable LEDs driven by 5V Arduino Pro Minis. To make it, Alina designed and built her own 3D printer, learned to airbrush, and tackled accelerometers for the first time.

@spoon_makes

Credit
John Jiao

## LEAH BUECHLEY

Leah Buechley's ground-breaking work includes more than just inventing the seminal LilyPad Arduino and founding the High-Low Tech Group at the MIT Media Lab. As a maker, she explores the intersection of engineering, art, and design. She currently runs a design firm and is experimenting with generative, code-based forms in fabric and wearables.

leahbuechley.com

Credit
Leah Buechley

## MAKEFASHION

MakeFashion is a Calgary-based initiative that seeks to merge fashion with cutting-edge electronics. Through hands-on workshops and international runway shows, MakeFashion brings designers and engineers together to create high-tech fashion.

makefashion.ca

**MakeFashion brings designers and engineers together to create high-tech fashion**

**Credit**
Pretty Flowers by Maria Orduz Pinto for MakeFashion

**Photo**
Kelly Hofer

## KOBAKANT

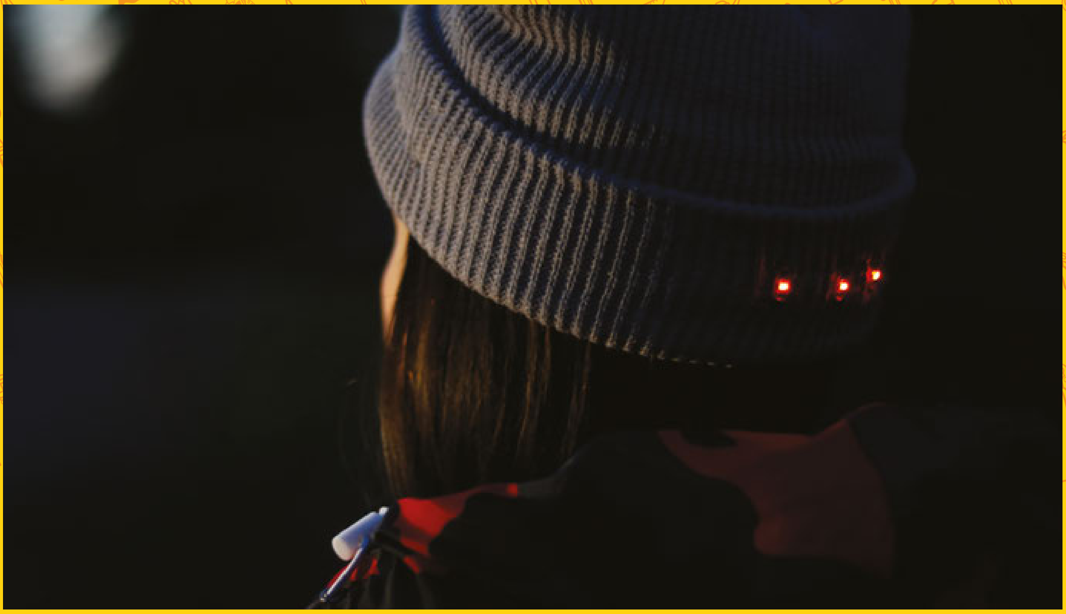
Hannah Perner-Wilson and Mika Satomi have been pioneering e-textiles together since 2006. Their projects include a crying dress and a beautiful collection of sensor-laden conductive textiles. In December 2017 they launched KOBA by KOBAKANT, a true tailoring shop for e-textiles in Berlin. For one year only, they built bespoke wearables and shared every build as an open-source tutorial online.

kobakant.at

Credit
KOBAKANT

# Sew lighting into a hat

Keep yourself warm and visible on dark nights with a simple circuit



**This is a simple and easy sewn circuit project, and it's great for getting started with wearables.** We'll sew three LEDs onto a knit hat for more visibility at night. Once you've gotten the basics down with this project, you can use this circuit to sew LEDs into other garments too. Let's get started!

## CIRCUIT OVERVIEW

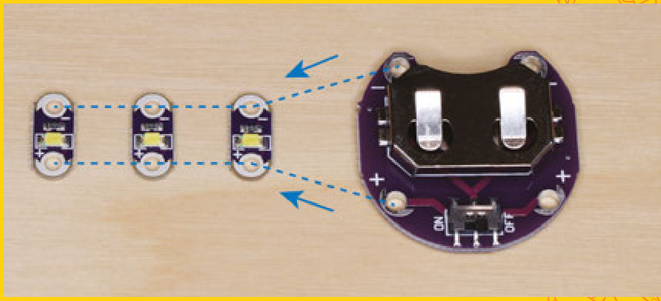
We'll be sewing the LEDs in parallel, so that each LED gets the same amount of voltage from the battery. All the LEDs will be in the same orientation in our circuit. Look for the + and - on each LED and follow the circuit diagram (shown right) as you build. Leave the battery out of your battery holder until you've finished building the circuit.

## PREPARE THE HAT

The LEDs will be attached to the outside of the hat, and the battery holder will be hidden inside the hat's folded cuff. You'll either need to use a hat with a cuff, or fold back the edge of a longer hat. If your cuff doesn't want to stay folded, tack it up at the sides with a few stitches of regular sewing thread. The cuff should be at least 4 cm wide to accommodate the battery holder.

## ADD THE LEDS

If your sewable LEDs came in a strip, gently snap them apart with small pliers. Find the front of the hat. Place the three LEDs on the front of the hat fold with the negative (-) side up and the positive (+) side down. Space them about 1.5 cm apart. Tack each LED in place with a small amount of hot glue.



**Above** Be seen in the dark with a light-up hat

**Right** Connect the LEDs to the battery holder

**Above**
The battery holder lives on the inside of the folded-up part of the hat, its unsightliness hidden

## ADD THE BATTERY HOLDER

Since the battery holder is a rigid, flat piece, we'll place it on the side of the hat so it will be against the flat part of your head. Fold the edge down at one side of your hat. Place the battery holder so that its opening points toward the edge of the hat and the negative sew tab is on top. Use a dab of hot glue to tack it in place for now.

## SEW THE CIRCUIT

Now that we have our components tacked in place, we can sew the circuit to connect them. Thread your needle with about 50 cm of conductive thread and tie a knot at the end. This may feel like a long piece of thread, but we want to sew each leg of the circuit in one continuous run.

Start at the battery holder: begin your run by stitching four or five times around the positive sew tab of the battery holder. Pull the thread snug against the sew tab for a good connection.

Use a simple running stitch to sew to the positive sew tab of the first LED. Loop around the positive sew tab four or five times, pulling snug each time.

Continue sewing this leg of your circuit, and connect to the positive sew tabs of the other two LEDs in the same manner.

When you've connected the last positive sew tab, tie off your thread with a tight knot. Trim the thread close to the knot to prevent short circuits.

Rethread your needle with another 50 cm length of conductive thread. Go back to the battery holder and repeat this process to sew the negative leg of your circuit. Keep this second thread run at least 1 cm away from the first.

## LIGHT IT UP

Now, slide a coin cell battery into the battery holder. The LilyPad sewable battery holder has a handy on/off switch, so turn your project on and try it out. The battery holder should sit comfortably on the side of your head, and no conductive thread or component should be touching your skin.

If your LEDs don't light up properly, check to make sure they're all aligned correctly with each other, and that the battery holder is in the same orientation: negative sew tab on top. Also make sure your conductive thread tails are trimmed to prevent shorts, especially at the battery holder.

Your author doubled the LED magic by repeating this process on the back of her hat with red LEDs. How will you customise this project and make it your own? Show us your build at **@HackSpaceMag**! □

" Once you've gotten the basics down with this project, you can use this circuit to **sew LEDs into other garments too** "

**YOU'LL NEED**

- **Knit hat** (find one with a folded cuff)
- **3 × sewable LEDs**
- **1 × sewable coin cell battery holder**
- **3 V coin cell battery**
- **Conductive thread**
- **Regular sewing thread**
- **Wire cutters**
- **Needle, thimble, scissors**
- **Hot glue gun and glue sticks**

**Below**
White LEDS at the front, red at the back

**Below**
The battery holder should go along the side of the hat, where it will fit along the side of your head

# Build custom LED snowboarding goggles

Shine bright this winter with these stunning shades

**The arrival of winter means snow-blanketed days and long, dark nights.** In other words, it's the perfect season for creating flashy wearable projects! In this tutorial, we'll put a strip of programmable LEDs into a pair of snowboarding goggles for a futuristic look. Sport them in the snow or at the lodge, and bring a blinking rainbow of joy to your winter festivities.

If you're a generalist, rejoice! We'll do some soldering, sewing, hot-gluing, and even a little programming. This is a great project for beginners with a little bit of soldering experience. For advanced folks, this is a fun, one-day build.

Wearable projects are a great way to combine different types of making and design, and there are plenty of ways to customise your build. Your first design choice is: what goggles to use? Snowboarding goggles are a good base to start with: most will have ample room for the LEDs, and foam sides that are easy to pass wires through. If you've already invested in an expensive pair of snowboarding goggles, you may want to purchase a less expensive pair for this. Your author purchased the pair used here for about $40 (£30).

A strip of Adafruit NeoPixels works perfectly in this project, and can be cut to size for a custom fit. If this is your first project using NeoPixel strips, prepare to be hooked! These are flexible metal strips that are packed with individually addressable LEDs. They come in several varieties and, in this project, we're using the most densely packed option: a whopping 144 LEDs per metre. We'll remove the protective sleeve the NeoPixels come in but, for projects where the strip is exposed to the elements, you'll want to leave it on. You can choose between a black or white base strip; the black disappears nicely behind darkly tinted goggles.

An Adafruit Gemma microcontroller is another good choice for this build: it's small, sewable,

## GETTING STARTED

**YOU'LL NEED**

- **Snowboarding goggles**
- **Adafruit NeoPixel digital RGB LED strip**
- **Adafruit Gemma microcontroller**
- **30 AWG silicone-coated wire**
- **3.7 V LiPo battery** 1600 mAh
- **4-way stretch Spandex** to match your goggles, about 20 cm square
- **Black gaffer tape**

and has a built-in on/off switch. Instead of sewing the circuit, we'll solder our electrical connections and use the remaining holes to secure the Gemma in place. Programming the Gemma is easy with the Arduino IDE and, even if you're new to coding, we'll get up-and-running quickly by modifying an example sketch included with the NeoPixel library. For experienced coders, this is a great place to make this project your own with a unique animation.

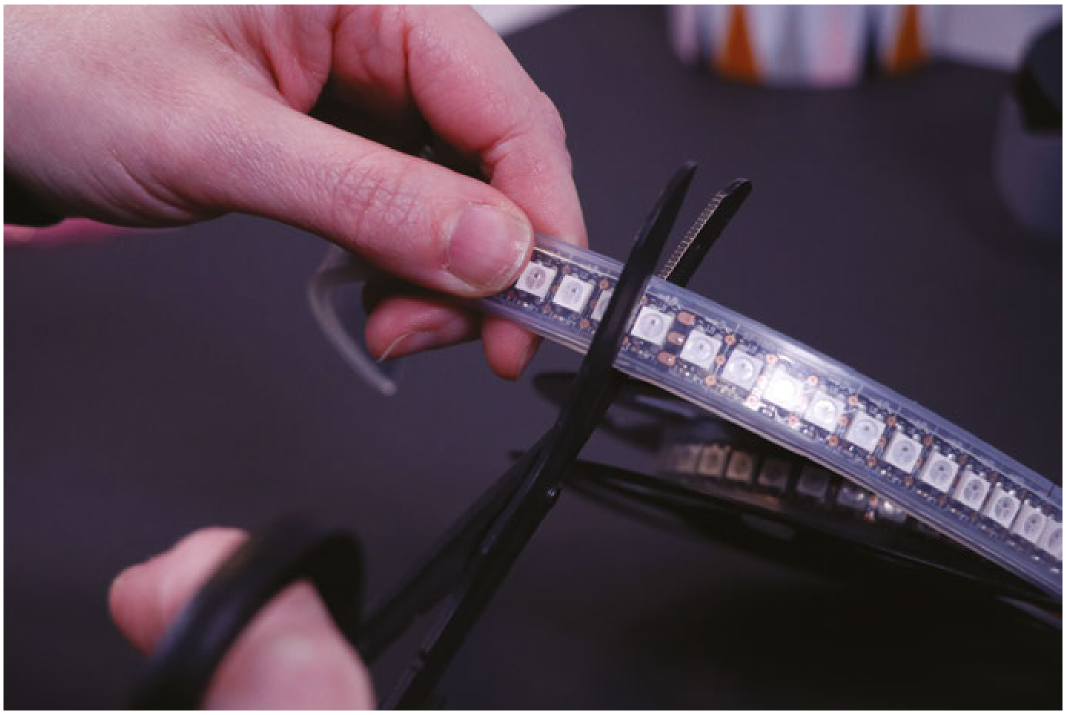
A word about safety: you should be able to wear these modified goggles comfortably over your eyes. We'll be backing the LED strip with tape, but a small amount of light may still come through. Remove the goggles if you experience discomfort. You should have good visibility out of the goggles; however, some peripheral vision may be blocked, and so you should not wear them while actually snowboarding, skiing, piloting a spaceship, or doing any other activity where impeded vision would be dangerous.

The circuit in this project is simple and versatile: it can be used in many other wearable projects where a strand of programmable LEDs would work. You could use a NeoPixel ring instead of a strand for an arc-reactor-inspired glove. Or, swap the snowboarding goggles for a dog collar, and make your furry companion festive and visible during winter walks.

### PREPARE THE NEOPIXEL STRIP

Start by measuring the inside of your goggles. Remove any protective film on the inside of the visor, and measure the length of the visor on the inside, at the very top of your goggles. Our goggles measured about 18 cm.

Next, cut your NeoPixel strip to size using your goggle measurement. Note the cutting lines on the strip. Cut through the middle of the contacts, so that each piece has usable contacts to solder to. The contacts are very tiny, so aim carefully before you cut. Wire cutters are fine for this, but trauma scissors are useful. Cut through the silicone sleeve and the NeoPixel strip at the same time, then remove the strip



**Above** Fabric tape measures are easier to bend around the goggles

**Right** A single snip and our LED strip is the correct length

**Above** The NeoPixels should leave enough space to see properly

from the sleeve. The cut corners will be sharp – it's a good idea to round them, but be careful not to cut off the contacts or any circuitry.

Test the fit by placing the strip into your goggles. It should fit comfortably at the very top of the visor, with a little bit of space at each end. If it buckles or flexes, it's too long. Cut off one LED and check again.

## INSTALL THE NEOPIXEL STRIP

Cut three lengths of 30 AWG silicone-coated wire, about 15 cm long each. Use different colours of wire to keep things organised. Strip and tin one end of each wire. Note the directional arrows that show which way data flows through the NeoPixel strip. Tin the contacts on the input end of the NeoPixel strip, and solder the wires to each contact. We used red for power, black for ground, and white for data. Add strain relief to the wires by applying a small amount of hot glue on each connection.

**Above** Measure twice, cut once and your strip should fit perfectly

Place the NeoPixel strip into your goggles with the LEDs face down against the visor and the wires pointing to the right. Thread a few centimetres of the wires through the eye of the yarn needle. Use the needle to pass the wires through the foam barrier at the side of your goggles to the inside of the goggle strap. Remove the needle and keep it handy.

Position the NeoPixel strip at the very top of the visor and apply a generous bead of hot glue to the top edge. Do one short section at a time, and be careful while holding the strip in place, as the hot glue will heat up the metal. Our strip felt secure with glue just along the top edge, but feel free to add some glue to the bottom edge if you think you need it. For strain relief, also apply a bit of hot glue to the wires on the right, just before they exit through the foam barrier.

Cover the back of the NeoPixel strip with black gaffer tape. Overlap the bottom edge slightly to secure the strip to the visor and block most of the light from the LEDs. We found it easiest to apply the tape in pieces, rather than using one long piece.

Thread the wires through the yarn needle again, and this time pass them through to the outside of the elastic strap. Give the wires a little extra slack at this point so they're not strained when putting on and removing the goggles.

"**This is a great project for beginners** with a little bit of soldering experience. For advanced folks, this is a fun, **one-day build**"

## CONNECT THE GEMMA

The Gemma microcontroller will be mounted where the wires come through the strap. Hold the Gemma in place, with the JST battery connector to the left. Find the ideal length for your wires by holding them up to their respective pins. Your power wire goes to Vout, ground to GND, and data to D0. Cut the wires accordingly, then strip and tin the ends.

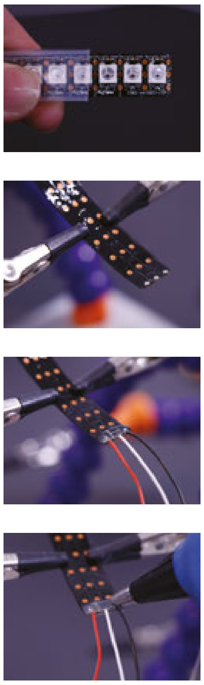
Feed the wires to their pins on the Gemma from the back to the front, and solder the connections on the front of the Gemma. Dab a bit of hot glue onto each connection on the back of the Gemma and let cool. Gently pull the slack of the wires to the inside of the strap, so the Gemma sits flat on the outside of the strap. Then, glue the Gemma in place and stitch to the elastic band through the unused GPIO holes (D1, A1, and 3Vo) with a needle and thread.

Download the Arduino IDE at **hsmag.cc/fKwgoe**.

Learn more about using Gemma and NeoPixels at **hsmag.cc/mcjcLa**. →

### TOOLS

- **Measuring tape** (the soft type)
- **Soldering iron and solder**
- **Wire strippers**
- **Helping hand tool** with clips
- **Wire cutters or trauma scissors**
- **Hot glue gun and glue sticks**
- **Sewing machine** (optional)
- **Needle and thread**
- **Computer and micro USB cable**



**Above** It takes just three solder joints to connect the LED strip

## GETTING STARTED

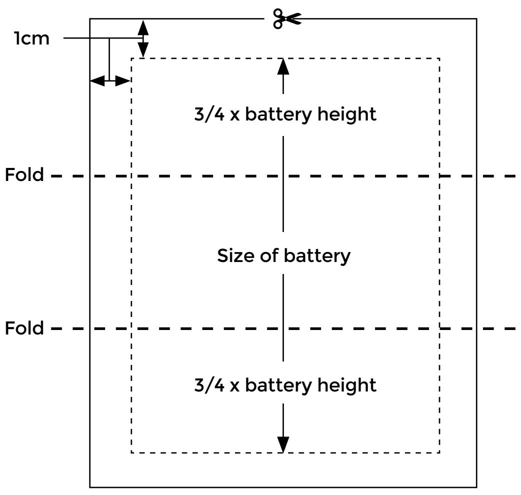


**Figure 1** ↗
The pattern for the battery holder

### PROGRAM THE GEMMA

Use a micro USB cable to connect the Gemma to your computer, then start the Arduino IDE. If this is your first time using NeoPixels, you'll have to download and install the Adafruit NeoPixel library. Find the Library Manager in the Sketch menu: Sketch > Include Library > Manage Libraries. Search for 'Adafruit NeoPixel by Adafruit' to find the library, and click install.

The library comes with some example code sketches that we can modify for use in this project. Go to File > Examples > Adafruit NeoPixel and choose Strandtest. This will pull up a sketch that cycles through several animations for testing your NeoPixel strip. First we'll test our strip, and then we'll modify the code to only show one of the animations the whole time.

Before uploading the code to the Gemma, we need to update two values in it to match our project: the pin we've connected our strip to, and the number of pixels in our strip. This sketch makes it easy to change these values by putting them at the top. Find this line near the beginning of the code:

```
#define PIN 6
```

We've connected our strip to pin 0, so change the `6` to `0`.

A few lines down from that, find this line

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

In this line, the first parameter in the brackets is the number of pixels in the strip. Change `60` to the number of pixels in your goggles (23 for us).

To upload, we need to tell the Arduino IDE what board we are using: in Tools > Board, select Adafruit Gemma 8MHz. In Tools > Programmer, select USBtinyISP. You're ready to upload!

Press the tiny black button on the Gemma to enter bootloader mode. Bootloader mode will last for about two minutes; a red light will flash on the Gemma during this mode. While the bootloader is active, click the upload arrow at the top of the Strandtest sketch and you should see the code compile and upload to the Gemma. In a few seconds, your LEDs should turn on! Watch the pretty lights cycle through the test animations, and decide which one is your favourite.

> " A strip of **Adafruit NeoPixels works perfectly in this project,** and can be cut to size for a custom fit "

Now let's go back into the code, and choose one animation for the Gemma to play. Scroll down to this line in the code:

```
void loop() {
```

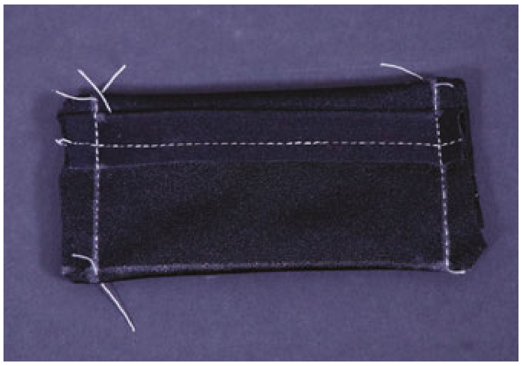
This is the beginning of the section that cycles through the example animations. We'll simply tell the program to ignore the animations we don't want by adding two slashes at the beginning of their lines. Look through the list and find the animation you want to keep (we chose rainbowCycle). Type // at the beginning of the other lines in the list. Here's how ours looked:

```
void loop() {
 // Some example procedures showing how to
display to the pixels:
 // colorWipe(strip.Color(255, 0, 0), 50); // Red
 // colorWipe(strip.Color(0, 255, 0), 50); //
Green
 // colorWipe(strip.Color(0, 0, 255), 50); //
Blue
 // Send a theater pixel chase in...
 // theaterChase(strip.Color(127, 127, 127), 50);
```

### QUICK TIP

If your project needs more I/O pins than the Gemma has, Adafruit also makes the Flora, which is similar but bigger. The Lilypad is an official Arduino board similar in size to the Flora, and makes a great option for wearables projects.

**Right** ⇨
The large pads on the Gemma are easy to solder to



```
// White
 // theaterChase(strip.Color(127, 0, 0), 50); //
Red
 // theaterChase(strip.Color(0, 0, 127), 50); //
Blue

 // rainbow(20);
 rainbowCycle(20);
 // theaterChaseRainbow(50);
}
```

Leave the rest of the code as it is. Press the black button on the Gemma to start the bootloader again, and upload your new code. If your animation plays as expected, hooray! Unplug the USB cable, and let's move on to the battery.

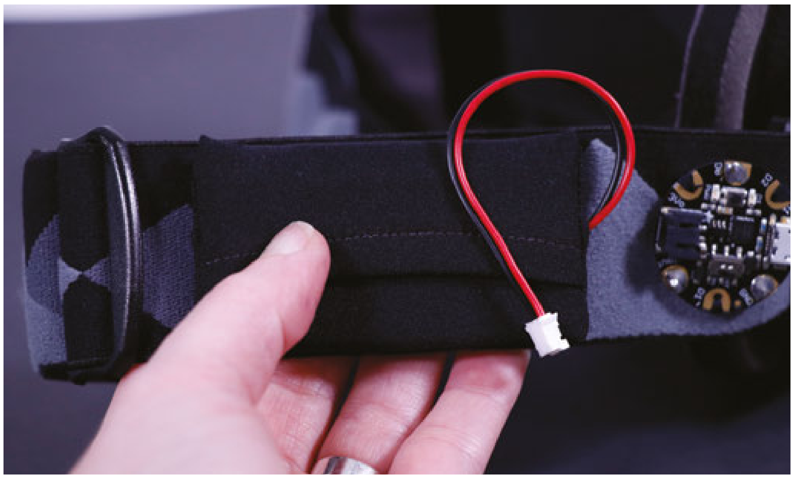
## MAKE THE BATTERY HOLDER

We suggest you cover the LiPo batteries with gaffer tape for extra protection, and mark their mAh rating on the tape. Draw up a pattern for the battery holder. Start by marking the size of your battery on a piece of four-way stretch Spandex. Then use the diagram (**Figure 1**) to mark out the rest of your pattern. Hang on to this diagram for future projects; just scale it to the size of your battery.

Cut out the holder, then fold and sew the top and bottom edges as shown. Stretch the fabric slightly as you sew – this will build some stretch into the seams so they don't break when in use. Fold along the fold lines, right sides facing in, and sew the side seams. Trim threads and corners, and flip the holder right side out.

Note where the wires come out of the battery: at the centre of one side. Snip a small hole into one side of the battery holder for the wires to feed through. Do not cut through the seam.

Now, insert the battery into its holder: feed the JST connector wire into the opening of the holder and carefully out through the hole you made in the side, then simply slide the battery into the holder. It should be a snug fit, but you shouldn't have to wrestle too desperately with the battery to get it in the right place. Note that you should always handle LiPo batteries with care.



## INSTALL THE BATTERY

Hold the battery and holder in place on the elastic band next to the Gemma, and mark the placement of the corners on the band. The wires of the battery should reach the JST battery connector on the Gemma comfortably. If your wires are a bit too long, like ours were, move the battery back or gently form them into a loop.

Remove the battery from its holder and sew the holder into place on the band, stretching it to reach the corner points you marked. Hand-sew around all four sides with a whip-stitch or back-stitch. Insert your battery back into the holder, then plug the battery into the Gemma. If you formed your battery's wires into a loop, sew the loop down so they won't catch on anything.

## LIGHT IT UP!

Flip the tiny switch on the Gemma to ON, and your LEDs should fire up! Take your new flashy goggles for a spin as they are, or turn them into a sci-fi costume piece by adding painted foam pieces or scavenged greebles. Show us **@HackSpaceMag** how you customised this project and made it your own!

**Above**
LiPo batteries are easy to connect and can power a wide range of projects

**Left**
A modified strap hides the circuit

**Below**
The coloured LEDs add a futuristic feel to the goggles

GETTING STARTED

# Light up your wardrobe

Use the same simple circuit to add a splash of colour in three different ways

**With a few modifications, one circuit can make many different projects. Here are three wearable ideas all built from one simple circuit.** First, we'll make an LED headband from start to finish. Then, we'll use the same circuit to make both a *Star Trek*-inspired combadge and a light-up hoodie.

In all three builds, we'll play with capacitive touch and make a button using conductive fabric. This is a great technique for wearables, and the Gemma M0 from Adafruit makes it easy to incorporate into projects. Let's get started!

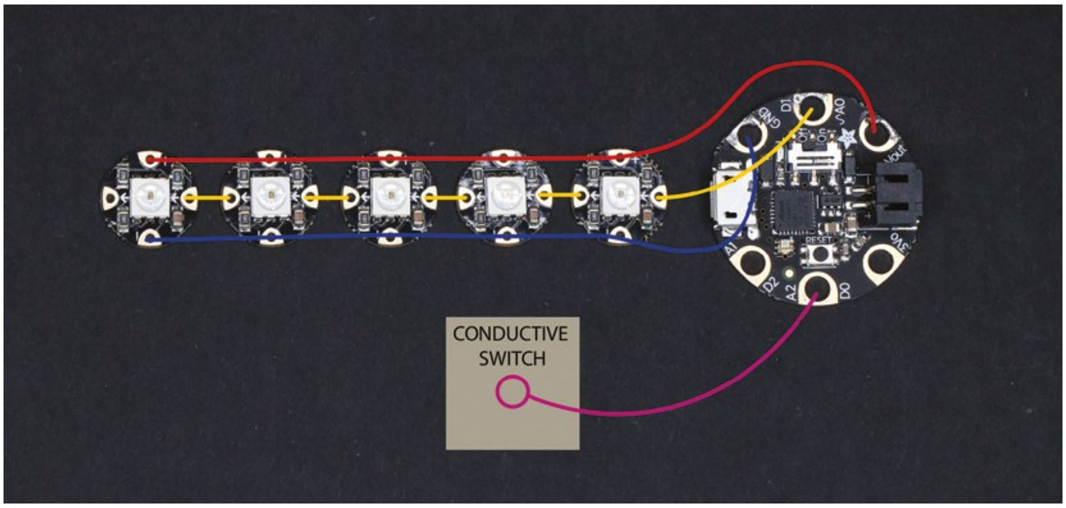
## THE CIRCUIT

This circuit is basically a strand of NeoPixels controlled by a Gemma M0 microcontroller. It's simple enough to be used for many different projects, but we've given it a fun twist and added a soft 'button' made of conductive fabric. Touching the fabric changes the colour of the NeoPixels from red to green to blue.

NeoPixels come in several different forms: strips, circles, sewable singles, and more. In most cases, the different form factors are interchangeable. With simple projects like this one, you'll just need to update the number of pixels specified in your project's code.

The Adafruit Gemma M0 microcontroller is great for wearable projects, as it comes with an on/off switch and a JST battery connector on board. We'll make use of the Gemma M0's built-in capacitive touch function, and program it using beginner-friendly CircuitPython. →

## GETTING STARTED



**Right** NeoPixels are great when you've only got a small number of GPIO pins

### YOU'LL NEED

- Gemma M0
- 3.7 V 150 mAh LiPo battery
- Conductive fabric
- Conductive thread
- Silicone-coated stranded wire
- Velcro
- NeoPixels for your project
- Soldering iron and soldering supplies

### FOR THE LED HEADBAND

- 5 × sewable NeoPixels
- Wide headband

### FOR THE COMBADGE

- 2 × NeoPixel sticks
- Pin back
- Thin craft foam

### FOR THE LED HOODIE

- 10 × sewable NeoPixels
- Hoodie
- Needle and sewing thread
- Clips or clothes pins

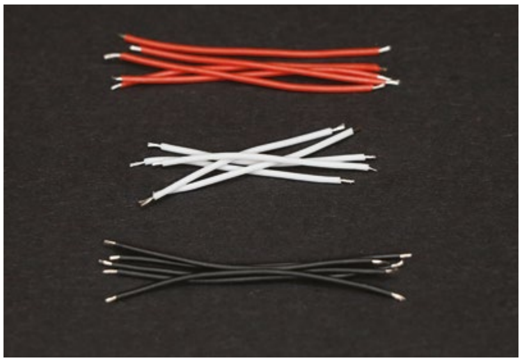
### LED HEADBAND – BUILD THE CIRCUIT

We'll start by building a chain of NeoPixels. Notice the directional arrows on each sewable NeoPixel. All NeoPixels pass data in only one direction, so always look for these arrows when soldering strips, sticks, or single NeoPixels together. The arrows should always point in the same direction.

Cut five sets of power and ground wires in 3 cm pieces. Cut five sets of data wires in 2.5 cm pieces – these need to be shorter than your other wires or the strand will not lay flat. Using different colours of wire keeps your build organised, but for a sleeker look you can use just one colour of wire. Strip and tin the ends of each wire.

Solder one data wire to the in and out pins of the first NeoPixel. Solder two power and ground wires to their pins, as shown. This is the beginning of your chain.

Solder a wire from the data-out pin of the first NeoPixel to the data-in pin of the next NeoPixel. Then solder wires to the power, ground, and data-out pins as with the first NeoPixel, and chaining the power and ground from the first NeoPixel to the second. Continue building the chain, making sure the arrows on the NeoPixels all point in the same direction.



### ADD THE GEMMA M0

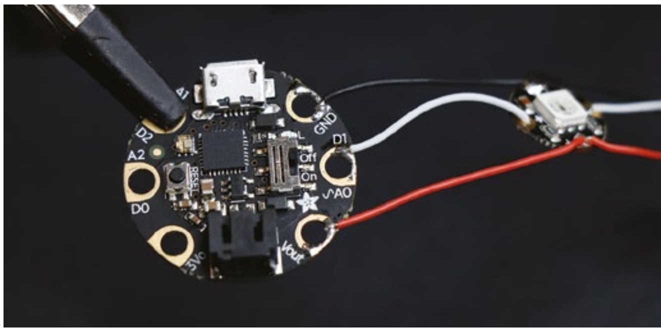
When you've completed your chain, it's time to add the Gemma M0 to the input side. Solder the connections as follows: Ground to GND, power to Vout, and data to D1.

### MAKE THE CONDUCTIVE SWITCH

To make the conductive switch, start with a piece of wire about 6 cm long. Strip about 1.5 cm of insulation off of one end, and coil it around an awl or a skinny knitting needle to make a loop. Tin the loop with solder to keep it in shape.

Cut a small piece of conductive fabric, about 1.5 cm square. Place the wire loop in the centre of the back side of the fabric and sew the loop onto the fabric using conductive thread. You can use regular thread for this, too; just be sure to pull your stitches snug to keep the wire loop tight against the conductive fabric.

Your conductive switch is complete! Solder the end of the wire to A2 on the Gemma M0. Now we're ready to code.



**Left** Get everything ready before you start and you won't need to mess around with wire-cutters when soldering

**Above** The small size of the Gemma makes it a great choice for projects like these

## PROGRAM THE GEMMA M0

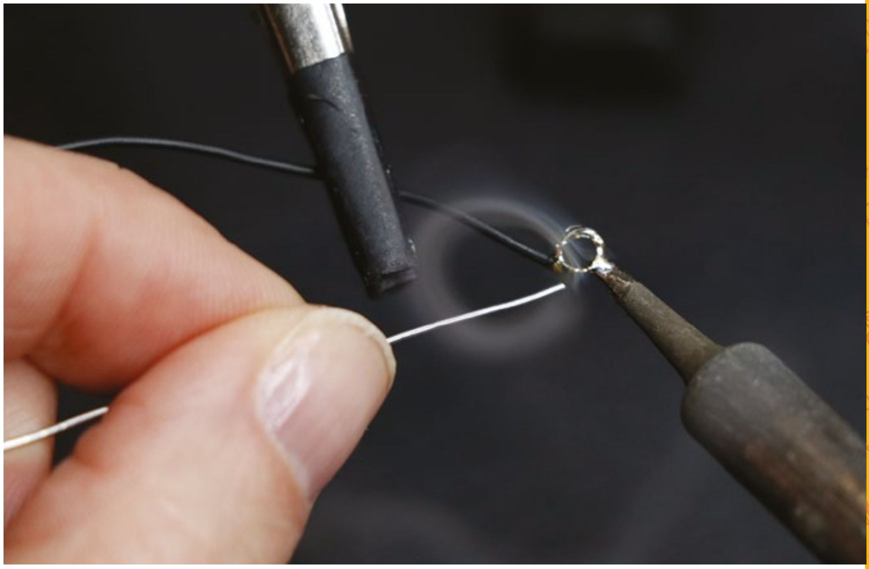
For in-depth setup help for the Gemma M0 and CircuitPython, visit **hsmag.cc/NCwBLc**.

Connect your Gemma M0 to your computer using a micro USB cable. The Gemma M0 appears as a drive called CIRCUITPY. You'll need to install the **neopixel.mpy** library for the code to work. The library is available from **hsmag.cc/AYaScW**, along with more information about using Gemma M0 and CircuitPython. Install the **neopixel.mpy** library by copying it into the **lib** folder on the CIRCUITPY drive.

Once your library has been installed, you can move on to programming the Gemma M0. With CircuitPython, you can do this by editing a text file on the board. Find the file on the drive labelled **main.py** and open it in your favourite text editor. Replace the text in that file with that in the **light-up.py** file downloadable at **hsmag.cc/book2-assets** and save the file. This will update the board, and all of your NeoPixels should light up and be red.

Tapping the conductive fabric switch should change the colour of your NeoPixels to green. Another tap should turn them blue. If the colours are cycling by themselves, or the capacitive touch doesn't behave as expected, try pressing the tiny black reset button on your board to recalibrate it.

Unplug the Gemma M0 from the computer (it's a good idea to eject the drive first). Cover the LiPo battery with electrical tape. This will help protect it from scrapes and sharp things, as well as create a barrier of insulation between the battery and the conductive fabric. Glue the conductive fabric to the front of the battery – don't use hot glue for this. Now plug the battery into the Gemma M0 and make sure everything still works.

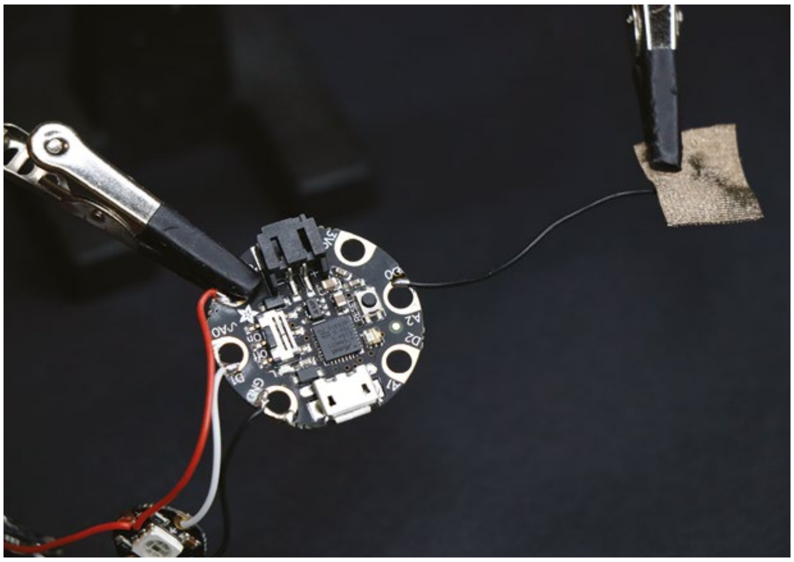


**Above** A drop of solder helps the loop on the conductive switch keep its shape

## PUT IT ALL TOGETHER

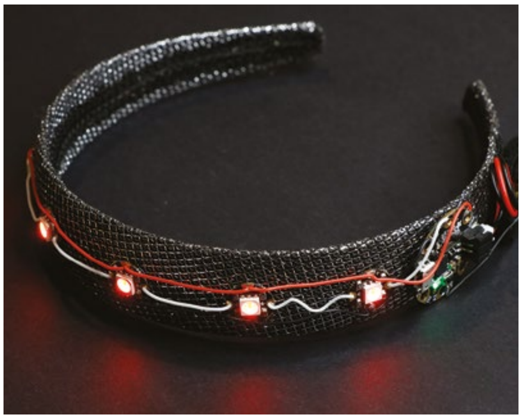
Centre the NeoPixels on top of the headband and glue them in place with hot glue. You can also use hot glue to attach the Gemma M0 at the side of the headband. Use adhesive-backed Velcro to attach the battery to the headband. This makes it easy to replace the battery or remove it for charging.

This headband is great on its own, or you can add material on top of the NeoPixels for more impact. Fabric flowers, feathers, or tulle work well for diffusing light. Just tap the conductive switch to change colour modes. Or, get creative with code and make new light animations! →

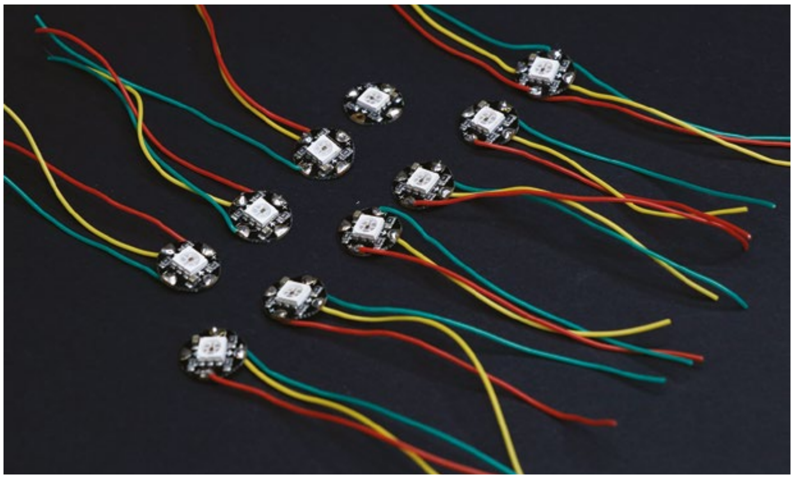


**Left** Conductive fabric allows you to sew capacitive switches into loads of wearables

**Below** You can choose wire to stand out or blend in, depending on your fashion taste



## GETTING STARTED



**Above** ↗
Soldering wires to each NeoPixel first can help avoid tangles

### LIGHT-UP HOODIE

Although the size and shape of this project is very different, it uses exactly the same circuit and components as the LED headband. We've simply doubled the number of sewable NeoPixels and sewn them into a hoodie. There's just one small change to make to the code to make this work.

Lay the NeoPixels out around the inside edge of your hood and decide how far apart you want to space them – about 7 cm or so works well. Build the NeoPixel strand as we did before, remembering to keep the data arrows pointing in one direction down the chain.

Connect the Gemma to the input side of the chain. Make the conductive switch and connect it to the Gemma M0, as described in the headband build. Glue the conductive switch to the battery as well.

**Below** ↘
The hoodie uses the same circuit as the headband, but with longer wires

### MODIFY THE CODE

We can use the exact same code we used for the headband project, with one small change: we'll need to change the number of NeoPixels specified in the code. This is a common update to make when using code from NeoPixel tutorials and examples, so it's good to get comfortable with the concept.

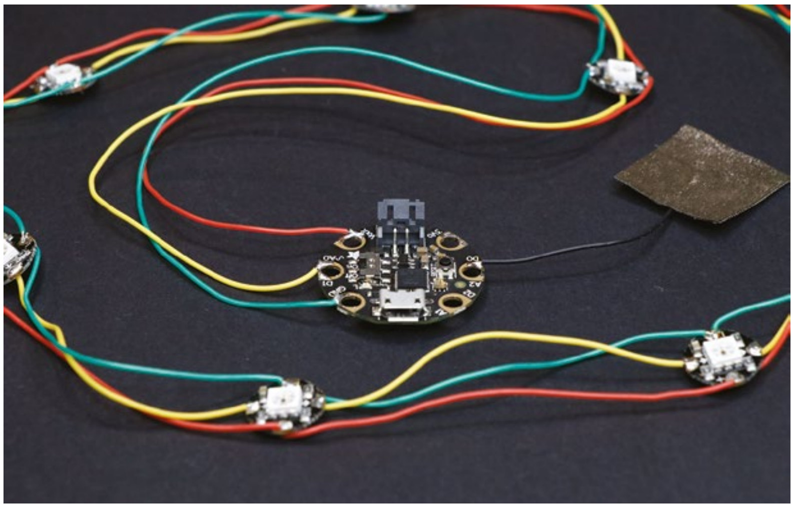
Connect the Gemma M0 to your computer and program it using the method described in the headband build. Only five of your NeoPixels will light up. To fix this, open your **main.py file** and look for this line in the code:

```
numpix = 5
```

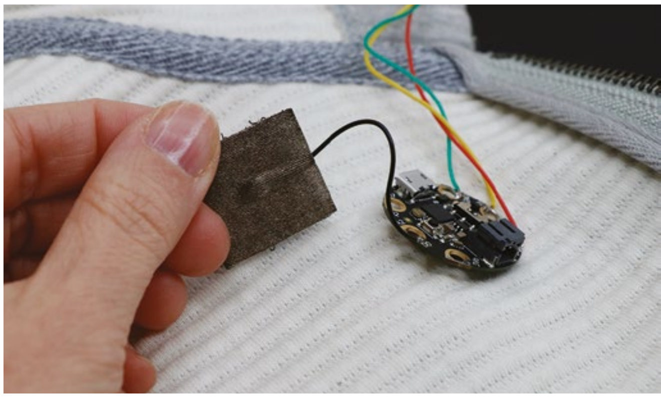
This line creates a variable for telling the microcontroller how many NeoPixels we are using. Since we have doubled the NeoPixels in this project, change the `5` to `10`. Save the file and your Gemma M0 should automatically update. Now, all ten of your NeoPixels should light up! When everything lights up and works correctly, you're ready to sew the circuit into your hoodie.

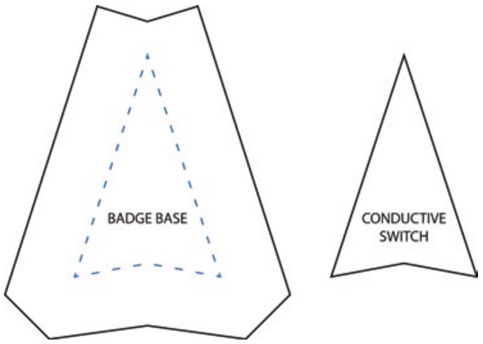
Use clothes pins to hold your circuit in place on the inside of the hood. Then hand-sew the strand of NeoPixels into place with strong sewing thread. Use large stitches about ½ cm apart. Consider your NeoPixel strand removable – to wash your hoodie, cut away your stitches to remove the strand for laundering. The rest of the circuit attaches with Velcro for easy removal.

Cut three small squares of Velcro loop (the soft side): one each for the conductive fabric pad, the Gemma M0, and the battery. Position the Velcro squares near the end of the Gemma M0 so that the conductive switch is easy to reach, and the battery can plug into the Gemma M0 comfortably. Attach the Gemma M0, the battery, and the conductive fabric switch with the hook-side Velcro, and your light-up hoodie is complete!

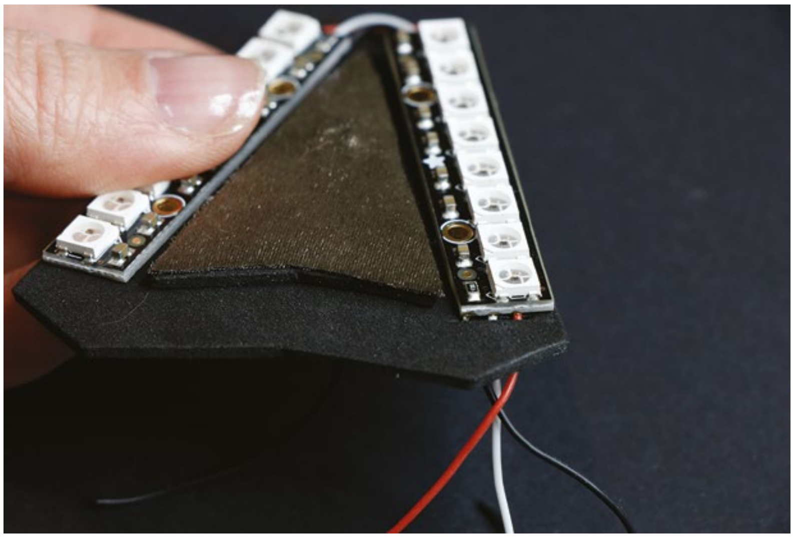


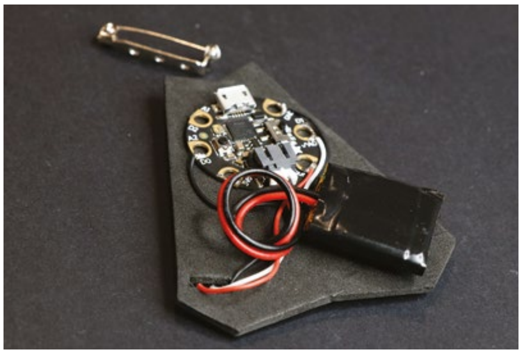
**Below** ↘
Attaching the Gemma and switch with Velcro makes it easier to wash your hoodie





**Above** Use this design in foam to make a firm base for your combadge





**Above** The conductive fabric should fit snugly between the NeoPixels

**Left** Select the right-sized LiPo and everything will fit behind the badge

**Below** Ready to boldly explore the universe – just don't wear a red shirt, obviously

## COMBADGE

This *Star Trek*-inspired combadge pin is a fun way to use the conductive fabric button. Tapping the centre of the badge changes the colour of the LEDs, which is satisfyingly next generation. The circuit is built in the same way as the previous two projects, but instead of chaining together individual NeoPixels, we'll combine two NeoPixel sticks to make two sides of the triangular badge.

Cut both triangles out of foam, and cut the smaller triangle out of conductive fabric as well. Use the same method described in the headband tutorial to create a sew loop in a small piece of wire, and attach it to the back of the conductive fabric triangle.

Glue the conductive fabric to the smaller foam triangle, making a hole for the wire to pass through to the back side.

Chain the two NeoPixel sticks together in the same way that we chained the individual sewable NeoPixels together: solder wires between the sticks to connect power, ground, and data (data-out to data-in). Solder wires to the input side for data-in, ground, and power, but don't connect the Gemma M0 just yet.

Glue the small triangle in place, making a hole for the wire to pass through to the back of the large triangle. Glue the NeoPixel sticks in place, making another hole for the wires to pass through to the back.

On the back of the badge, solder the input wires to the Gemma. Attach the Gemma M0 with hot glue. Use small squares of adhesive-backed Velcro to attach the battery to the badge, and plug it into the Gemma M0. Then, attach a pin at the top of the badge with hot glue.

The code for this project is exactly the same as the code we've been using, but again, we'll need to update the number of NeoPixels specified in the code. Go back into your code and change the `numpix` value to `16`. Save the file, and your badge should light up completely.

Pin the combadge on your chest and enjoy tapping it like they do on the Enterprise NCC-1701-D!

What else can you make this universal circuit into? Show us your builds **@HackSpaceMag**! □

# No code needed: LED shoes

Make your feet sparkle

**OMG shoes!** Here's a simple project with a high-impact end result: LED shoes. Light-up shoes are everywhere these days, and you can definitely make your own. You don't even need to know how to code or solder for this build. This a great first project for anyone who wants to get started with wearable electronics, and a fun, quick project for advanced makers.

We'll keep it simple and build a soft circuit – no soldering needed! You also won't need a sewing machine. Sewable components go together quickly with conductive thread, and hand-sewing is a useful skill to hone. We'll use two universally valuable hand stitches in this project: a running stitch and a whip-stitch. You can make almost anything with these two stitches, and you won't regret having them in your skill set!

You can use this circuit and these components in any fabric project that you want to add light to. It's a perfect fit for cosplay, and would work well in gloves, hats, or any soft accessory. Always wear socks with these shoes, so that the sewn circuit does not touch your skin. It's also a good idea to coat your finished project with a hydrophobic spray

like Scotchgard. (You'll want to stay out of puddles and skip these on rainy days.)

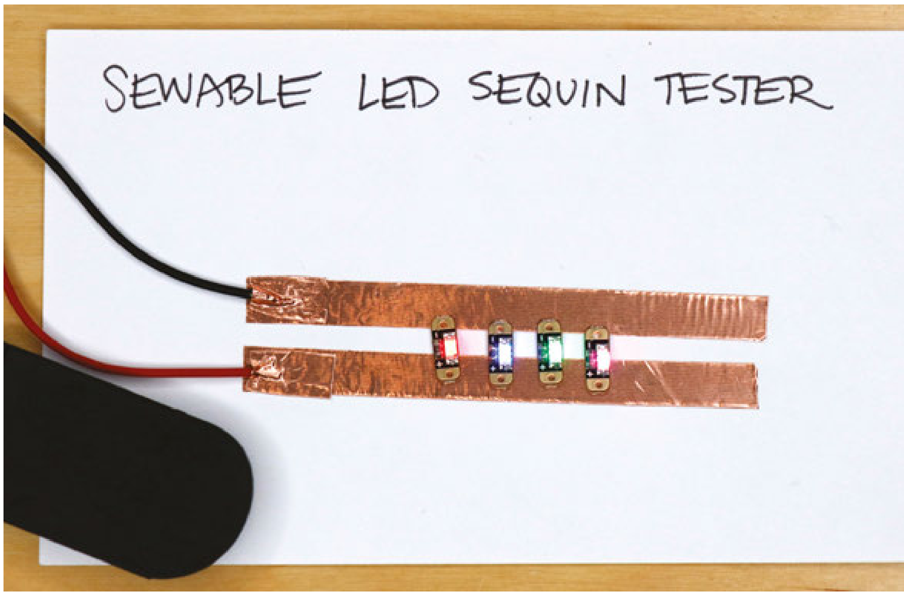
The shoes used in this tutorial are canvas high-top trainers. High-tops don't just look cool, they also provide a lot of surface area to work with for decoration and spacing your circuit out comfortably. Canvas is easy to sew through, and sturdy enough to keep its shape while supporting the weight of the electronic components. Boots would give you even more surface area to work with, but be wary of leather shoes for this project: leather is harder to sew through and less forgiving than fabric, as piercing leather leaves permanent holes.

## ELEMENTARY ELECTRONICS

Tiny, sewable LEDs are perfect for wearable projects, and are easy to get started with. We used Adafruit LED sequins ($3.95 for a pack of five) in our build, and Pimoroni sells LilyPad sewable LEDs (£3 for a pack of five) as well. These are not RGB LEDs (like NeoPixels), but single-colour LEDs that do not require a microcontroller to function. Just provide 3 to 6V DC and you've got light! Each LED has a resistor on board, so all that's needed is an appropriate power source, like a coin cell battery or two. We'll chain five of them together for a line of light on our shoes. When planning your design, you'll want to know what your LEDs look like when they are lit up, so we'll make a reusable tester card. Keep it in your toolkit for future projects!

Although the LEDs in this project are sewable components, we will use a battery holder that is not specifically made for sewing projects. We'll modify the battery holder slightly to work with our sewn circuit, and leave it accessible on the outside of each shoe for easy battery changing. Sewable battery components are available, too; just make sure to find one with an on/off switch.

**Below** PCBs can have bumps where they separate from a panel. A quick sand down tidies things up



**Above** Double-checking LEDs before committing them to the project

There are lots of ways to decorate plain trainers, from textile paint to sequins – have fun with this! White shoes make a good reflective surface for catching the light, while dark shoes would make the LEDs stand out like stars in the night sky. We kept it simple with a black marker on white shoes for a monochromatic look. When the LEDs are turned on, their colour reflects on the white canvas for a swirly, bright effect in the dark!

## MAKE AN LED TESTER

If you've never used these sewable LEDs before, it can be helpful to make a tester card with some copper tape and the battery holder. Take an index card and two strips of thin copper tape about 8cm long. Apply the tape to the card about 5mm apart. These will be the power rails of your tester card.

Strip about 1cm of insulation from the ends of the battery holder's wires. Using more copper tape, attach each wire to one of the rails, as shown above. Press the tape down securely over the wires for solid contact. Add batteries to the battery holder and flip the switch to on.

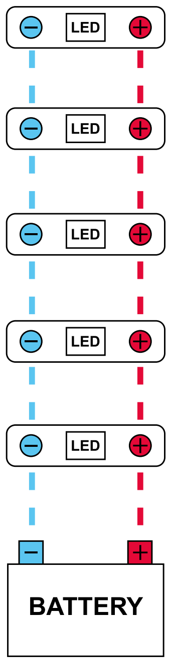
Sewable LED sequins from Adafruit come in strips of five. Snap off each LED, and use an emery board to file down the rough edge that was attached to the strip.

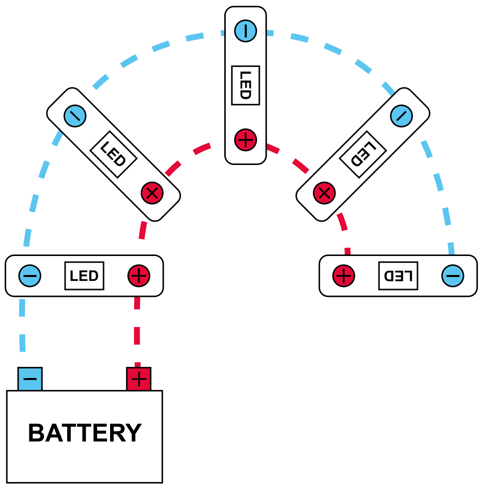
Now you can lay your LEDs across the rails, with the positive pin (+) on the positive rail (the red wire), and the negative pin (-) on the negative rail (the black wire). The LED should light up solidly. If not, try wiggling the LED around or pressing down on it for a good connection. Use this card to view the colours of your LEDs and make sure they're all working properly before you sew them into your project. Hold onto this card for future projects! →

## YOU'LL NEED

- **Canvas high-top trainers**
- **Sewable LED sequins** (Adafruit, five per shoe)
- **Battery holder with on/off switch** (one per shoe, plus one for tester)
- **2032 coin cell batteries** (two for each shoe, plus two for tester)
- **Conductive thread**
- **Erasable ink pen**
- **Markers or textile paint**
- **Index card or small piece of cardstock**
- **Copper tape**
- **E6000 or other strong fabric glue**

## GETTING STARTED





**Above** It's a simple circuit: just connect all the positives together

**Below** We'll use a regular battery holder for this project

### STITCH DICTIONARY – RUNNING STITCH

The running stitch is the most basic and universal hand stitch. Thread your needle with a single length of thread, and make a double knot at the end. Start with your needle on the back side of your fabric, and push your needle through to the front of your fabric so that the knot is hidden on the back of your work. Make small running stitches by piercing the fabric in equal amounts – about 2 or 3mm at a time – passing the needle under and over the fabric. Your stitching should look like a dashed line.

### STITCH DICTIONARY – WHIP-STITCH

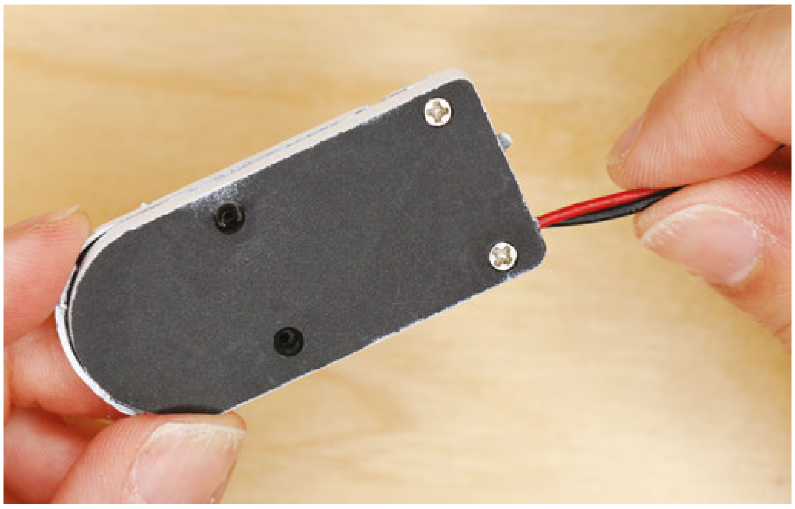
The whip-stitch is useful any time you want to attach something long and skinny, like wire or cord, to fabric. It's used a lot in wearables, and is great for attaching wires! Start the same way you would with a running stitch: thread your needle and tie a knot at the end of your thread. Push the needle through the fabric from back to front, coming up on one side of the wire you want to attach. Take your first stitch by pushing the needle down into the fabric right next to the wire, then up through the fabric coming up on the other side of the wire. For the next stitch, pass the needle back over the wire and take another stitch underneath it, in the same direction as the first stitch. Continue down the length of your wire, sewing over it, making sure not to pierce the wire's insulation as you sew. Keep your stitches a few millimetres apart for a secure hold. When you reach the end, tie off your thread on the back of your work. This method is also called 'couching' in embroidery.

### DESIGN YOUR SHOES

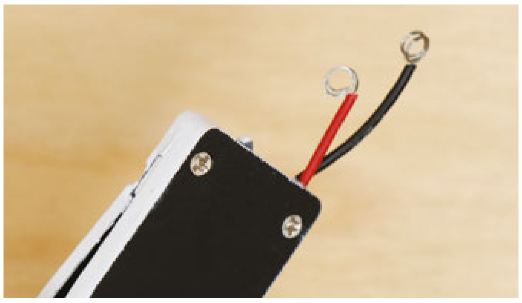
Take some time to design your shoes and decide where your LEDs will go, and how you'll decorate the shoes. It's important to do this first, so you can plan out your circuit accordingly. These canvas trainers have prominent stitching near the lacing holes, so we built our design around that. Find a nice flat area to mount the battery holder to, making sure it won't catch on anything as you walk. The inside ankle area is a good choice. We like the holder on the outside of the shoe, but if it's comfortable for you, you can try putting it on the inside.

While working on your design, keep in mind that each LED has a positive pin (+) on one end, and a negative pin (-) on the other. To chain the LEDs together, you'll need to connect all the positive pins to each other, and all the negative pins to each other, without crossing any connection lines – negative and positive should never touch. Arranging the LEDs like a ladder keeps the pins properly aligned and ensures that power and ground will not touch. If you prefer to lay the LEDs end to end, sew the positive pins together on one side and the negative pins on the other.

When you're happy with your design, it's time to start making!



**Below** You can make many electrical components sewable with a loop in the wire



**Above** The battery holder needs to be secure as it'll have to withstand quite a lot of shaking

## PREPARE THE BATTERY HOLDER

Our battery holder was black, and a quick hit of white spray paint made it blend in better with the white shoes. If painting your battery holder, cover the back of it with masking tape first so that it can be glued to the shoe later. To make our battery holder sewable, we used a tiny hand drill to make four small holes in the back of the holder.

When dry, cut the wires so that one wire is about 5 cm long, and the other is about 6 cm long. Staggering the wires helps ensure that the loops won't touch when they are sewn down near each other. Strip about 3 cm of insulation from each wire. Twist the exposed wires slightly, and then wrap them around something skinny and round (like a small screwdriver) to make tiny loops. These will be our connections for the conductive thread traces that will carry power to the LEDs. Optionally, if you have a soldering iron handy, adding solder to the loops makes them sturdier and easier to work with.

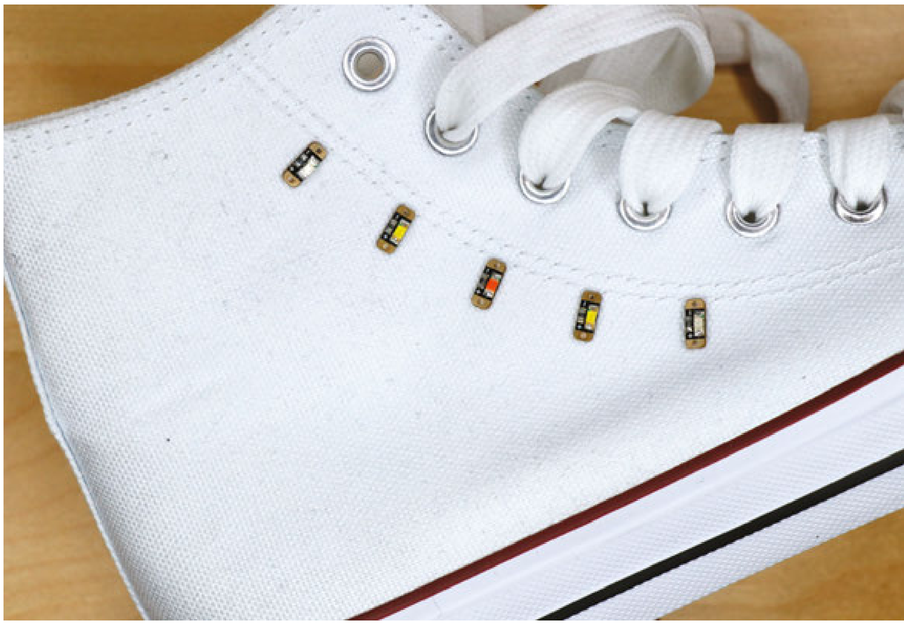
> **To make our battery holder sewable,** we used a tiny hand drill to make four small holes in the back of the holder

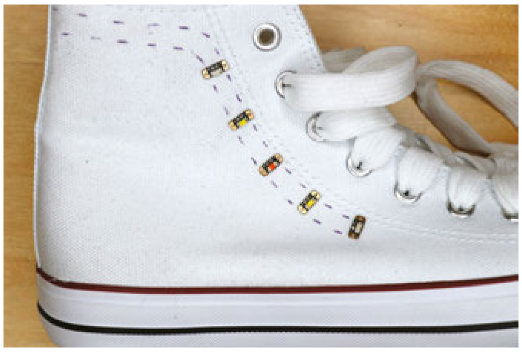
Position the battery holder and glue it in place with E6000, or other strong fabric glue. For a secure attachment, hand-sew the battery holder to the shoe using the holes we drilled. Use strong upholstery thread, or a double strand of hand-sewing thread for this step.

## ARRANGE THE LEDS

Refer to your design for the placement of the LEDs on your shoes. Arrange the LEDs on each shoe, making sure the power and ground pins are all correctly aligned. Then, use a dab of fabric glue or E6000 to hold each LED in place.

Now that we have the battery holder and LEDs attached, we can draw out our connection lines so we know exactly where to sew. Using an erasable pen, start at the wire loops from the battery holder, and draw a line connecting the negative wire to all the negative pins on the LEDs. Then repeat for the positive wire and positive pins. Your lines should never cross each other, and stop at the last LED.





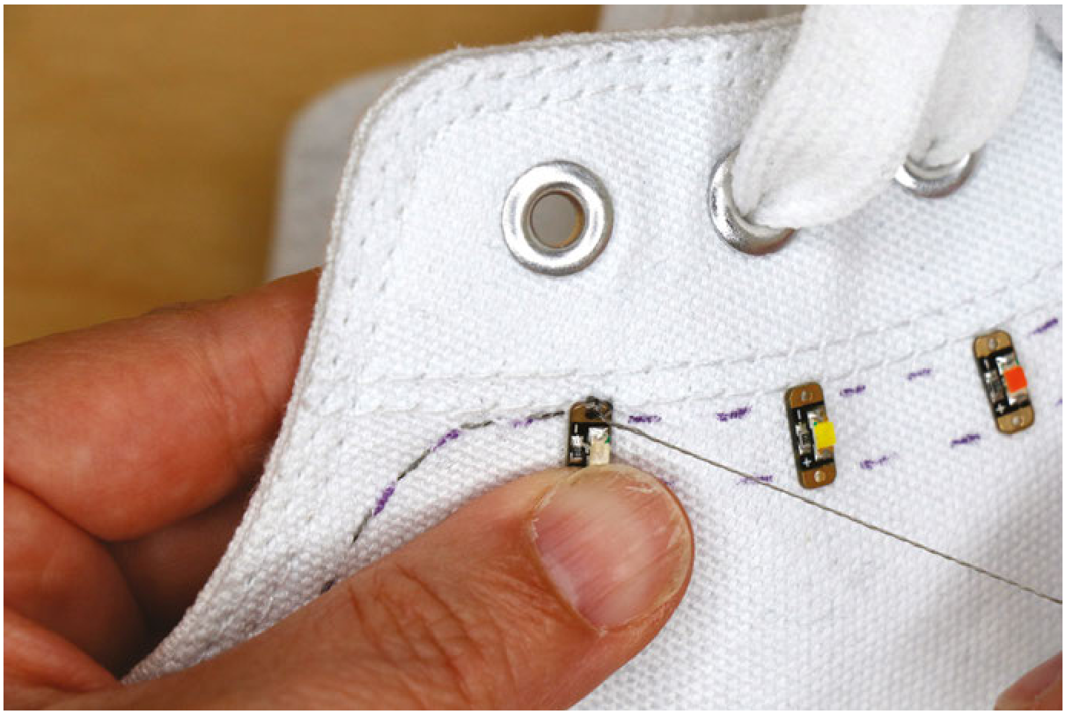
**Above** We've opted for a gentle curve of LEDs, but you can go for any design you like

**Left** Remember that your stitching will be visible, so plan your pattern

## SEW THE CIRCUIT

Thread your sewing needle with a long strand of conductive thread. Start by whip-stitching a few times around the negative wire loop from the battery holder, pulling your stitches tight for a good connection. Then use a running stitch to sew along your line to the negative pin on the first LED. Take four or five tight stitches through the negative pin to make a good connection and secure the LED to the shoe. Continue down your sketch line, until you've connected all the negative pins together. Ideally, →

## GETTING STARTED

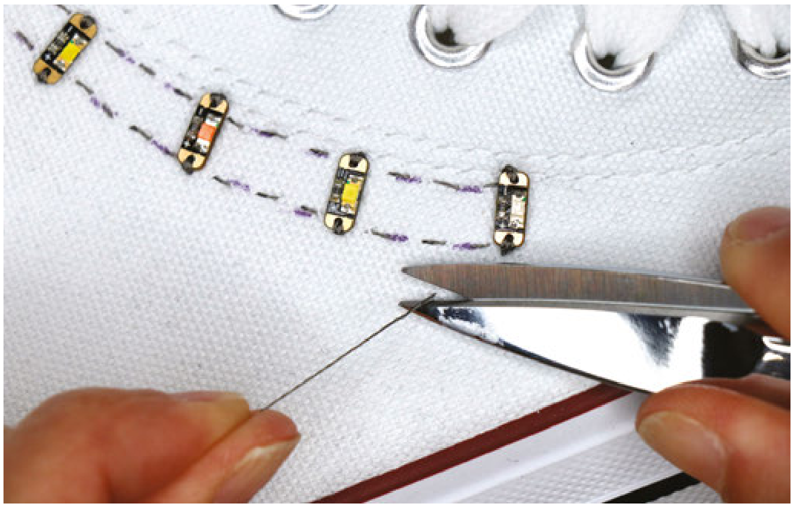


you'll sew the whole line in one go, so use a strand long enough to reach from the battery holder to the last LED in your chain. (If your thread runs out before you get to the end, no biggie! Just tie it off at one of the LEDs, then start a new strand in the same LED and continue.)

After sewing through the last LED, tie off the thread with a tight double knot. Dab a little CA glue or Fray Check on the knot for extra security. To bury the thread tail inside the fabric, take a stitch between the canvas and the lining of the shoe (the stitch shouldn't show on either the inside or outside), and pull slightly while cutting the thread flush. The thread tail will disappear safely into the fabric.

**Above** Keep your stitches through the LED tight to ensure a good connection

**Below** A careful final stitch will disappear into the shoe



" **If any LEDs flicker or don't light up,** check that the pins are correct and that your stitches are tight "

Repeat this process to connect the positive pins to the positive loop on the battery connector. When you've finished sewing your connections, pop two coin cell batteries into the battery holder and switch it on. Your LEDs should all light up solidly. If any LEDs flicker or don't light up, check that the pins are correct and that your stitches are tight. Also make sure that the power and ground lines are not touching at any point, and that you've cut any exposed thread tails short. When your LEDs are lighting up as intended, remove the batteries and set them aside.

### FINISHING TOUCHES

Now that we're sure everything works, we can put the finishing touches on our circuit. First, let's secure the battery loops to each shoe. Thread your sewing needle with regular sewing thread that is the same colour as your shoe, and whip-stitch all the way around the loop, so that the thread covers the exposed wires completely. This will keep the loop in place, as well as provide some strain relief and insulation. Tie off the thread, cut the thread tail short, and hit it with a bit of Fray Check for stability.

**Above** Make sure everything's securely attached before finishing up

Next, take a few stitches around the wires of the battery holder to keep them in place against the shoe and make them less likely to catch on things as you walk. That's all the hand-sewing needed for this project!

You're now ready to erase the sketch lines you made for your circuit, and the method will vary based on the type of marker you used. Many erasable pens for sewing require water for removal, but that's not ideal when working with electronics. The markers we used in our project disappear with heat, and a small craft iron makes quick work of erasing them.

## DECORATE YOUR SHOES

Now all that's left to do is to finish decorating your shoes so they don't just look good in the dark! Use whatever method you prefer to decorate your shoes: fabric paint, markers, or glue and sequins. For a cleaner application, stuff your shoes with stiff paper to hold their shape while you work. Sketch your design on the shoes with an erasable pen or light pencil first, then go for it with your art supply of choice. Fabric markers can be heat-set with a craft iron. To make the battery holder blend in even more, make your design flow right over it. Alternatively, you can make it stand out by painting it with a bright pop of colour. The choice is yours!

When your creation is complete, coat your shoes with hydrophobic spray to keep your design clean and your electronics dry. Then insert your batteries, flip the switches, and light up your shoes! You can use this same technique to make other accessories light up. There are endless design possibilities and lots of ways to customise your LED wearables. Show us your shiny projects **@HackSpaceMag!** □

**Above** Finish your design off with a permanent marker for some custom shoes

**Below** Time to hit the streets and reap your new-found geek-cred

# Infinity heels

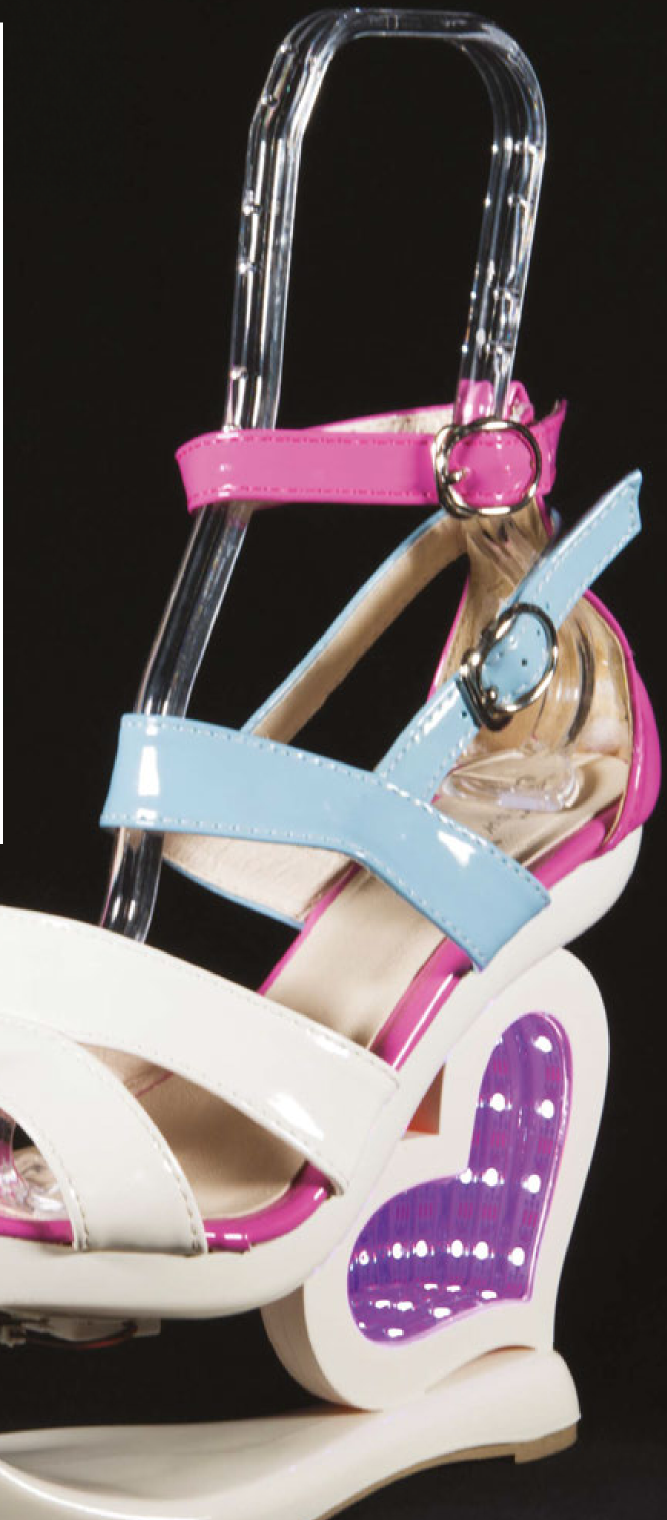
By Angela Sheehan @the_gella

**I found this wild pair of heart-shaped heels and augmented them with custom, programmable infinity mirrors.** I used a laser cutter to precisely cut mirrored acrylic to size, to fit in the heart shapes on each shoe. In between a layer of one-way and standard mirrored acrylic are 5 mm NeoPixel LED strips, each controlled by a Qduino Mini microcontroller and programmed with Arduino, utilising Adafruit's NeoPixel library.

The black wedges use six custom modules that are installed into the heart cut-outs. The Qduino and battery are hidden in the ankle strap.

The white wedges have one large infinity mirror installed in the heart shape, and the Qduino and battery are embedded in the space cut out of the platform near the toe.

Read about how I built the project and watch a video of them in action on the SparkFun blog – **hsmag.cc/TkWdbs**.



**Right** NeoPixels make any outfit at least 47.8% better (but not if you're a ninja)

# CODING

Get programming, using MicroPython or MakeCode, to make interactive wearables

88

TOP PROJECTS

## PAC-MAN HALLOWEEN

Dress to impress with these light-up ghost costumes straight from the arcade

70

GETTING STARTED

# Flaunt your skills with a light-up bag

Make an interactive tote bag with a Circuit Playground Express and a little sewing

**Need a project idea for your brand new Circuit Playground Express (CPX)?** Make it a wearable one! The CPX has oodles of sensors and NeoPixel LEDs built in, so no soldering or breadboarding is required to set up a circuit. You'll want to take it with you everywhere so you can tinker whenever inspiration strikes you. In this project, we'll sew it onto a tote for coding on the go!

The CPX is a little powerhouse, and we'll use the on-board accelerometer, capacitive touch capabilities, and ten NeoPixel LEDs to run some fun animations on the front of the tote bag. To make our program interactive, we'll add snaps for simple touch-sensitive buttons. Each button will play a different NeoPixel animation when pressed. While we're at it, we'll use the accelerometer to trigger a flashy animation if you happen to run, skip, or jump while out and about with your bag.

For portable power, we'll use a battery holder and three AAA batteries. Look for a battery holder with a power switch and a male JST connector, which will work perfectly with the JST connector on board the Circuit Playground Express. To be safe, leave the batteries out of the holder until your build is complete.

Your snaps should be metal – bare metal will work best, but these colourful enamelled snaps also worked well. If your snaps are coated, check them with a multimeter to make sure they are conductive. You can use any bag you like for this project, and you may already have the perfect bag to start with. If not, pick up a blank canvas tote at a craft store and decorate it to make it your own. The Circuit Playground Express is a perfect match for this 'hello world' text: the board fits nicely over the 'o' in 'world'. Keeping it front and centre makes it easy to plug the board into your computer any time you feel like changing the code!

## STENCIL YOUR TOTE

If you're starting with a brand new, blank canvas tote bag, be sure to wash, dry, and press it before decorating it. The fabric will take the paint better and make for a nicer finished project.

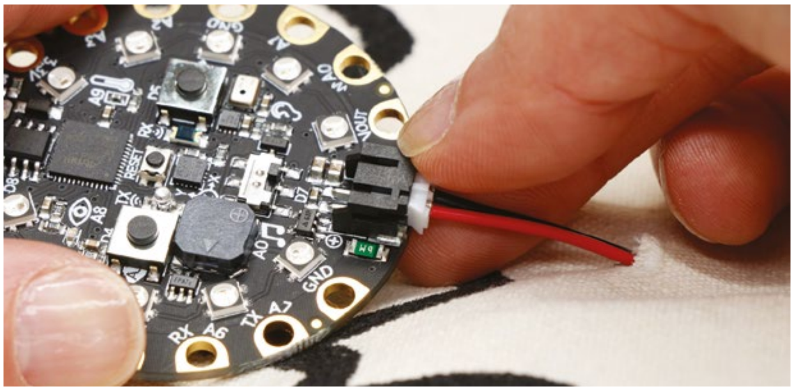
A vinyl cutter is nice for making stencils, but it's definitely not necessary. To make the stencil by hand, trace the 'hello world' text onto your contact paper, keeping a border of at least 2–3 cm around the text. Carefully cut out the text with a craft knife. Apply the stencil to the front of your tote, pressing down firmly along all the edges for good adhesion. Remember to place the hole shapes into the letters with negative spaces.

Using a craft sponge or spouncer, dab fabric paint over your stencil. For nice, crisp text, take your time and aim for an even, opaque application. Carefully remove the stencil and let the paint dry. When it's completely dry, it's a good idea to heat set your fabric paint according to the manufacturer's instructions. Now you're ready to make it light up! →

**Right** Stencilling, rather than drawing the design, means less chance of slip-ups

**Above** ↗
Time to hit the high street and let the world see your making skills

## GETTING STARTED



**Left**
The battery connector pushes firmly into place, so should hold even when the bag is in use

### YOU'LL NEED

**For the bag**

- Blank canvas tote bag
- Contact paper or adhesive shelf liner
- Craft knife or vinyl cutter
- Textile paint
- Craft sponge or spouncer
- Hand sewing kit
- Sewing machine (optional)
- Heavy-duty or upholstery thread

**For making the circuit**

- Circuit Playground Express
- AAA ×3 battery holder, with male JST connector and power switch
- Conductive thread
- Metal-capped prong snaps and setting tool (sometimes it's included)
- Fusible interfacing
- Fray Check
- Superglue
- Disappearing ink fabric pen

### MAKE A BUTTONHOLE

The Circuit Playground Express will be attached to the front of the bag, but the battery holder will be inside the bag. This means we need a neat way to pass wires through the front of the bag. A small buttonhole is perfect for this, and if you have a modern sewing machine, chances are it has a buttonhole function.

Place the Circuit Playground Express in its spot on the front of the bag, and mark where the buttonhole will go: about 1 cm below the JST battery connector on the CPX. Follow your sewing machine's procedure for making a buttonhole on your line, about 1 cm wide. Apply Fray Check along the centre of the buttonhole and let dry. Use a craft knife to cut the buttonhole open, being careful to only cut the fabric in the centre of the buttonhole.

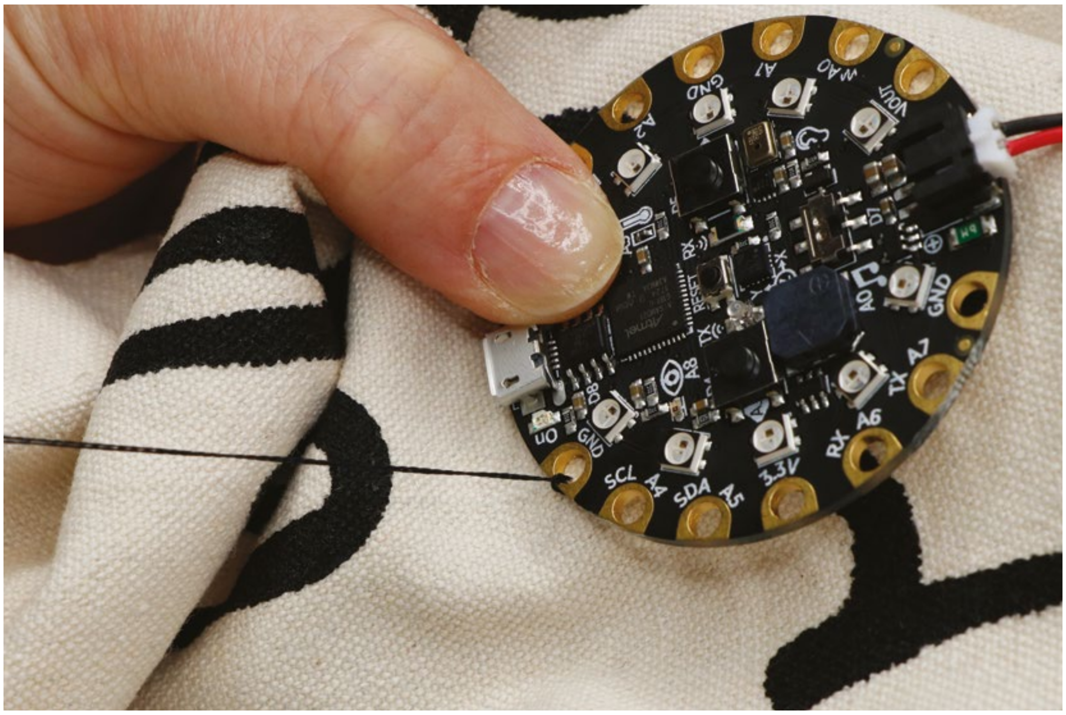
If you don't have a sewing machine, you can cut a small slit in the bag and hand-sew around the edges to bind them. Run a bead of Fray Check along the bound edges to stabilise the hole.

### ATTACH THE CIRCUIT PLAYGROUND EXPRESS

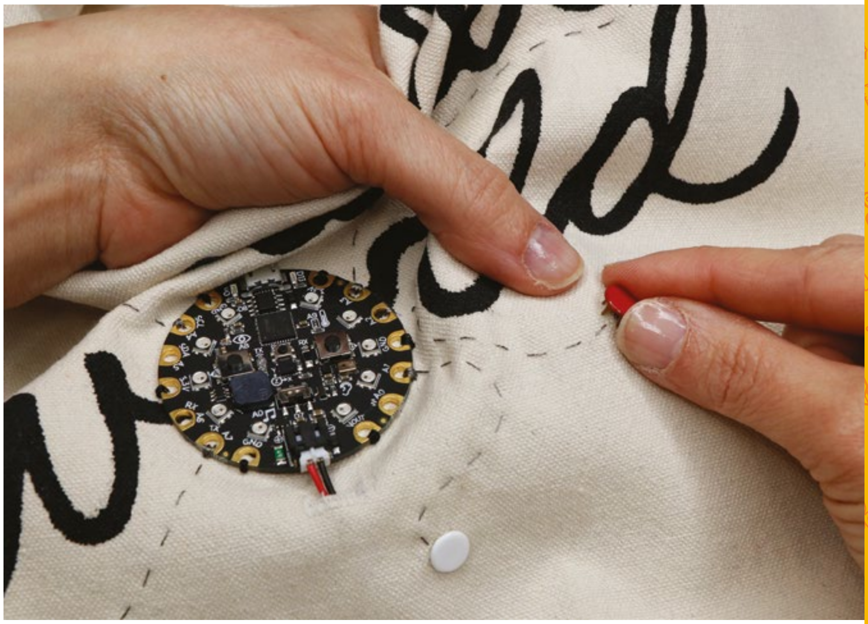
Place the empty battery holder inside the bag, and push its JST connector through the buttonhole to the front of the bag. Plug the JST connector into the black port at the bottom of the Circuit Playground Express, and lay the CPX in place on the front of the bag. You can use a dab of hot glue to hold the board in place while you sew it down.

Since we won't be using the power and ground pins in our circuit, we can use those holes to attach the board to the bag. Thread a needle with heavy-duty thread, or use a double strand of regular hand-sewing thread. Sew through the three GND pins, the two 3.3 V pins, and VOUT – that's six holes – for a secure attachment.

**Below**
The large holes on the CPX are easy to sew through, both for making the circuit and holding the board in place



**Above** Carefully mark your placements out before making any permanent incisions



**Above** You don't have to stitch in a straight line. Wearables are meant to be creative!

## ADD CONDUCTIVE BUTTONS

Decide where your metal snaps should go, and mark their locations on the front of the tote. Keep in mind that the A0 pin cannot be used for capacitive touch, but you can use any of the other I/O pins (A1 through A7). Draw the path you will sew from each snap to its I/O pin with disappearing ink. In this 'hello world' design, we are connecting to A1, A2, A3, A4, and A7.

Thread a needle with conductive thread and tie a tight knot at the end. For each button, start at the point where your snap will be, and push the needle through the fabric from the inside to the outside of the bag. Take seven or eight small stitches right on that spot to make a big dot – this will be the connection point for the snap to sit on top of. From there, sew with a running stitch along your path to the I/O pin you'll be connecting to.

When you get to the board, stitch around the pin about five or six times, and pull tight for a good connection. End with your thread on the inside of the bag, and tie a tight knot to finish off your run. Dab a bit of Fray Check or superglue onto the knot to keep it secure, and cut the thread tail short. Repeat this step for all the paths you marked.

Prong snaps come in four pieces, but for this project we'll only be using the top two pieces: the cap and the socket. To install each snap, place the pronged cap over your sewn connection point, and press the prongs through the fabric. Insert the prongs into the socket part of the snap, sandwiching the fabric and the conductive thread dot between the snap pieces. Set the snap with a setting tool and hammer.

When you've set all your snaps, turn the bag inside out so you can take a good look at your circuit. Make sure all the thread tails are cut short and not touching the other legs of the circuit. If everything looks good, you can remove the temporary markings you made. Now plug it in and see if it works!

## PROGRAM THE CIRCUIT PLAYGROUND EXPRESS

MakeCode is a great way to get started with code. It's visual and block-based, so programming is a matter of connecting the right blocks together. It's simple to get started, and perfect for unleashing all the bells and whistles packed into the Circuit Playground Express. So grab your computer, head to **makecode.adafruit.com**, and start a new project.

Click on the Light category, where you'll find lots of different blocks that can be used for animating the NeoPixels on the Circuit Playground Express. Grab the 'show animation' block and drag it into the 'forever' loop on your workspace. The CPX simulator on the left will now show the rainbow animation. →

**Left** A gentle whack fixes the snaps in place

## GETTING STARTED

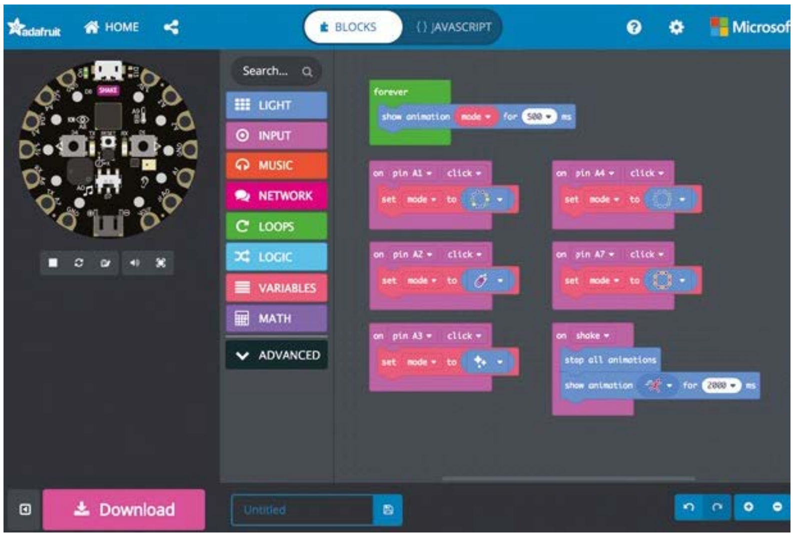
**Right** You can reprogram your CPX in place if you decide you want different animations

**Below** The MakeCode environment makes it easy to link events to touch actions

You could simply download this code to your CPX and have an awesome flashy bag. But let's go further and use our capacitive touch buttons!

MakeCode has several different animations built in, and for this project we can think of them as modes. So let's set up a variable to make it easy to change from one animation mode to another.

In the Variables category, click 'Make a Variable' and then name it 'mode'. Drag your new 'mode' variable onto your workspace, and drop it on top of the rainbow icon in the 'show animation' block. This will replace the rainbow animation with your variable.



```
forever
    show animation mode ▾ for 500 ▾ ms

on pin A1 ▾ click ▾
    set mode ▾ to ▾

on pin A2 ▾ click ▾
    set mode ▾ to ▾

on pin A3 ▾ click ▾
    set mode ▾ to ▾

on pin A4 ▾ click ▾
    set mode ▾ to ▾

on pin A7 ▾ click ▾
    set mode ▾ to ▾

on shake ▾
    stop all animations
    show animation ▾ for 2000 ▾ ms
```

Now we need to assign each of our capacitive touch buttons to play a different animation. Buttons are inputs, so head over to the Input category and drag the 'on button A click' block onto your workspace. We'll start with the button connected to pin A1, so change 'button A' to 'pin A1'. From the Variables category, grab the 'set mode to 0' block, and drop it inside your 'on pin A1 click' block.

Head into the Light category and grab the animation drop-down menu block – you'll need to scroll down to the 'More' section and it should be at the bottom. Drag the animation block on top of the '0' to place it in your 'set mode to' block. Now, when you click on the A1 pad on the CPX simulator, the rainbow animation will play!

Repeat the steps above to set up the rest of your capacitive touch buttons on pins A2, A3, A4, and A7, and choose a different animation for each one. To quickly copy the block of code you just created, right-click it and select 'Duplicate'.

Check your work by clicking on the pins in the simulator. If your animations are playing as expected, move on to setting up the accelerometer! Since you're likely to be moving around while out and about with your bag, we'll tell the CPX to flash the NeoPixels when it detects a big movement.

Just like our buttons, the accelerometer is also an input, so go back into the Input category and grab the 'on shake' block. Chances are there will already be an animation playing when the CPX is shaken, so the first thing we'll need to do is stop whatever is already running. From the Light category, grab the 'stop all animations' block and drop it into the 'on shake' block. Under this, add the 'show animation for 500 ms' block and select the running lights animation. Let's give this animation a little more airtime by changing the duration from 500 ms to 2 seconds. Check your work by clicking 'SHAKE' at the top of the CPX simulator. The lights should flash red for two seconds, and then go back to playing whatever animation was already running. If your code is working perfectly, it's time to load it onto your actual Circuit Playground Express!

Click 'Download' and follow MakeCode's instructions to load your new code onto your board. Tapping the snaps on your bag should trigger the animations you programmed, and shaking the bag should trigger the flashing runner lights. If something is not working as expected, check for crossed conductive threads and make sure your snaps are anchored well to the thread dots under them.

Check out the finished code for this project here: **hsmag.cc/ORBbKo**.

Need help? Go to **learn.adafruit.com/makecode** to learn more about programming your Circuit Playground Express with MakeCode.

" Since we won't be using the power and ground pins in our circuit, **we can use those holes to attach the board to the bag** "

**Above** Keep your circuit insulated from anything jangling around inside

## BACK WITH FUSIBLE INTERFACING

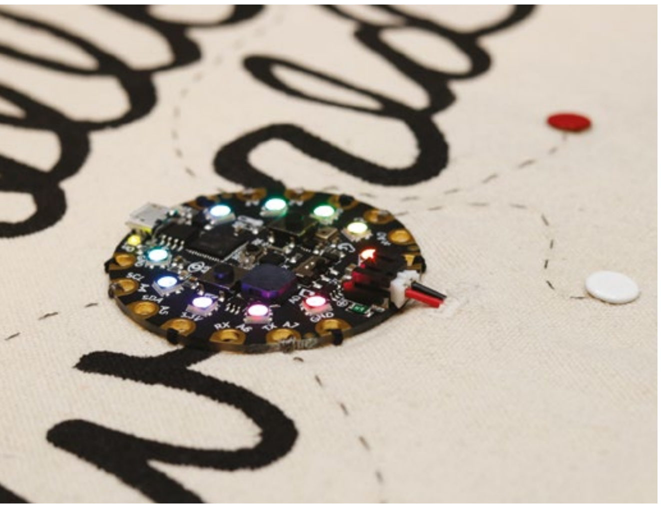
Now that we're sure everything is working, let's cover the back of the circuit with fusible interfacing. This will insulate our circuit nicely and keep the hand-sewn stitches from catching on objects inside the bag.

Turn the bag inside out again; remove the battery holder and set it aside. Cut a piece of interfacing that will cover the whole circuit. Follow the instructions for your fusible interfacing to apply it to the fabric. In general, you'll iron the interfacing on with a little bit of steam – you can iron over the conductive thread, but don't iron directly over the CPX or snaps. Just press around them so the fusible interfacing is held down securely. Make sure to avoid covering the buttonhole with interfacing.

If you've used steam to apply your interfacing, let your project dry completely before connecting it back to power. If you'd like to add some weatherproofing to the Circuit Playground Express, paint its surface with clear nail polish.

## GRAB AND GO!

It's time to plug in your battery holder, add batteries, and take your project for a spin! Flip the power switch to on, and shake your bag to see your NeoPixels flash. Try out your conductive buttons – what else can you make them do besides play animations? Whenever you have a new idea, your Circuit Playground Express will be right by your side, and we want to see what you make! Show us your projects **@HackSpaceMag**! ◻



**Left** With the battery hidden inside the bag, only the circular outline of the CPX adorns the bag

# Colour-sensing clutch bag

Use a Circuit Playground Express to create the ultimate fashion accessory

**It's no secret that at HackSpace we're hooked on the Circuit Playground Express. With so many on-board sensors and lights, we keep thinking of new ways to use it!**

This project is inspired by Angela Sheehan's fairy wings project – she uses a colour sensor in a magic wand to 'collect' colour from an object, while the lights in her wings change to match that colour – brilliant! In our build, we'll mimic this by using the built-in light sensor on the Circuit Playground Express, and a strip of NeoPixels inside a clear handbag. Change the colour of the lights in your bag to match your outfit, or a found object while you're out and about!

The circuit for this project is simple, and can be used in any project where you want to connect an external strip of NeoPixels to your Circuit Playground Express. Instead of sewing the circuit, we'll do some light soldering to connect everything together. Coding is easy with MakeCode, and once your project is complete, you can quickly modify and develop your code further, as the CPX remains easy to access on the front of your bag. The NeoPixel strip will be on the inside of your bag, so we'll need to pass wires through holes punched into the front. Choose a bag that you are willing to dedicate to this project for good. (Your author purchased the bag shown for about £25.)

The transparent 'holographic' coating on this bag makes a great surface for reflecting the NeoPixel LEDs, and it really shines in low-light conditions. Adafruit's NeoPixel strips are available in several different options, with varying densities of LEDs along the strip. The 60 LEDs per metre option (shown) is easy to work with, with bigger pads and easy-to-read markings. Advanced makers can intensify this project by sizing up to 144 LEDs per metre! NeoPixel strips are sold by the metre, but for most handbags you'll only use about 25 cm. Let's get started!

## LET'S MAKE SOME HOLES

First, we'll mark the placement of the holes we need to make to pass wires through the front of the bag. Place the Circuit Playground Express on the front of your bag, with the USB port at the top and the JST port at the bottom. Use a Sharpie to mark holes that the wires will go through: GND and VOUT on either side of the JST port, and A1. We'll also need some holes for mounting the CPX, so mark these four holes we won't be using in our circuit: GND and 3.3 V at the top, and A0 and TX/A7 at the bottom.

Set the Circuit Playground Express aside, and use an awl to make the holes you've marked. Align the awl precisely: there's no going back once you've made a hole in plastic. Twist the awl and slowly press to make holes about 1 mm in diameter. Take your time so as not to crack the plastic. →

## YOU'LL NEED

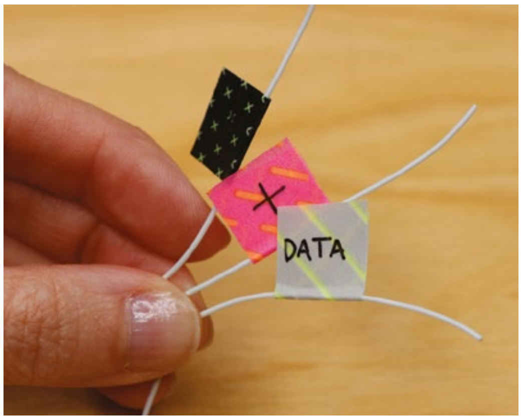
### Materials

- **Clear handbag** with a flat bottom
- **Circuit Playground Express**
- **NeoPixel strip**
- **Silicone-coated stranded wire 30 AWG**
- **AAA ×3 battery pack** with on/off switch and JST connector
- **Clear packing tape** or Gorilla tape
- **Small cable ties,** 101.6 mm length

### Tools

- **Awl**
- **Craft knife**
- **Basic soldering supplies**
- **Masking tape** or colourful washi tape
- **Scissors**
- **Third helping hand**

**Left** Gradual, even pressure is the key to making clean holes in plastic



**Right** Labels don't have to be colourful, but why not make life a bit brighter?

Made a hole in the wrong place? No problem! Make a few more holes and create a pattern – turn the mistake into a decorative feature!

## WIRING UP

Cut three lengths of wire – long enough to reach from the holes you just made, straight down to the bottom edge and over to the bottom corner of your bag. Cut the wires a little longer than you think you'll need; we'll cut them down later.

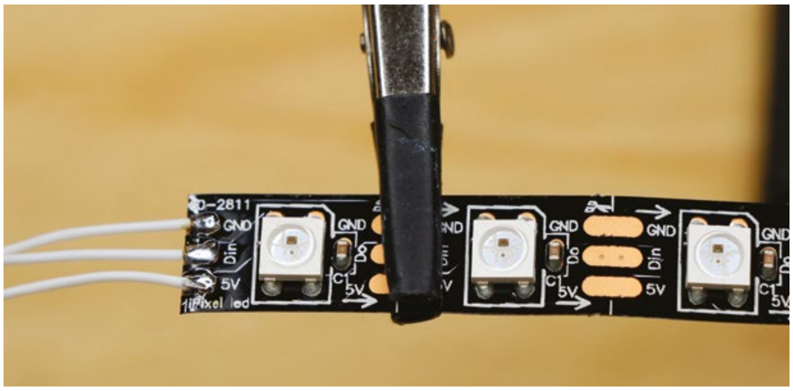
Using one colour of wire gives a clean look to this build, and is a nice option if the wires in your project will be visible, as they are on this bag. To avoid confusion while you're working, mark the wires temporarily with tape. Colourful tape is useful here, but a Sharpie and regular masking tape work just fine too. Set aside your prepared wires, and let's work on that NeoPixel strip.

NeoPixel strips come in one-metre lengths (or more), so you'll need to cut it down with a pair of scissors. Hold the NeoPixel strip against the flat bottom of your bag, and determine how many NeoPixel LEDs will fit on your bag. Allow at least 1 cm of space on each end so that the strip can be completely covered with tape later. Cut the strip through the midpoint of the conductive pads, after the last NeoPixel you'll be using. Remove the protective silicone sleeve from your cut piece.

Note the arrows printed on the NeoPixel strip: this is the direction that data flows through the strip. You'll need to solder your wires to the beginning of the strip, i.e. the end the arrows point away from. Strip and tin the ends of your wires, and tin the pads of the NeoPixel strip. Solder the wires to the strip: power to 5V, ground to GND, and data to Din.

**Below** A third-hand-style holder makes it easier to solder the wires in place

"**The circuit for this project is simple,** and can be used in any project where you want to connect an external strip of NeoPixels to your Circuit Playground Express"



## MAKING CONNECTIONS

Insert the NeoPixel strip into your bag, and feed each wire through its hole to the front of the bag. Refer to the Circuit Playground Express to make sure the wires are going through the correct holes. Once your wires are fed through their holes, mark them again with masking tape on the outside of your bag to make doubly sure they'll be connected properly to the CPX.

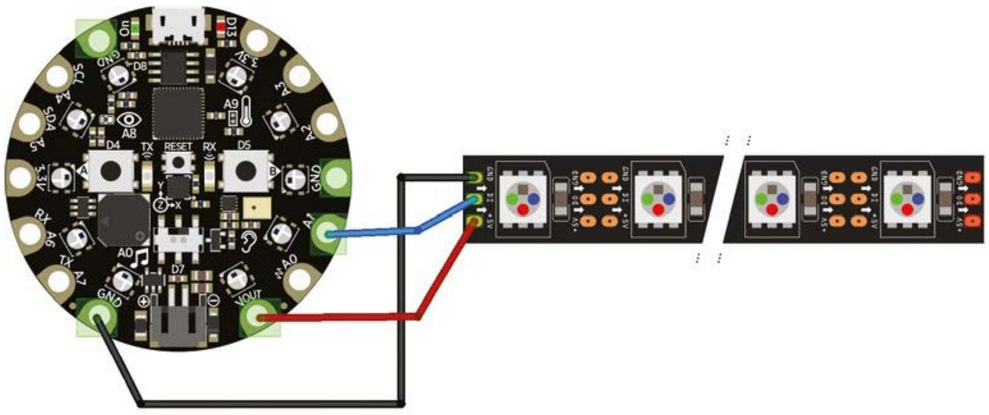
Secure the Circuit Playground Express in a set of third helping hands, to keep it from moving around while you solder. Tin the three pads you'll be using: GND and VOUT on either side of the JST connector, and A1. Strip about 5 mm of insulation off the end of the wires, and insert each wire through its hole on the CPX from back to front. Refer to the circuit diagram to make each connection correctly. For a secure connection, wrap the bare wire around the edge of the hole and solder in place. Trim off any protruding solder or wire. You don't want any pokey bits on your wearable tech!

## ENGAGE CPX

We'll use cable ties to make fasteners for attaching the board to the bag. Small black cable ties work well for this and won't be very visible when the lights are on. They're also easy to remove if you need to remove the CPX from this project later. You'll need eight cable ties in total.

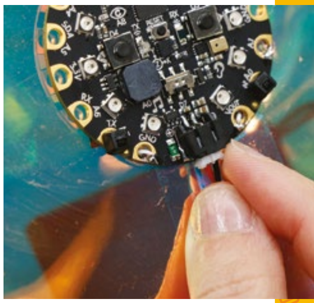
Snip the eyes off four of the cable ties, and save the long tails for another project (they're quite useful for applying tiny dabs of glue on things). These eyes will lock the other four cable ties in place.

Pull the slack in the wires to the inside of the bag, and lay the Circuit Playground Express in place against the front of the bag. Align the CPX with the holes you punched for GND and 3.3 V at the top, and A0 and TX/A7 at the bottom. From the inside of the bag, feed a cable tie through each mounting hole. Slide one eye



**Left**
The NeoPixels are really simple to wire. There's just power and one data connection

**Below**
The battery should push into the JST socket



onto each cable tie, and secure tightly. Make sure you are putting the eye onto the cable tie in the correct orientation, or the eye will slide right off. You can look at a fresh cable tie for reference. When your Circuit Playground Express is secured in place, use angle cutters to snip the end off each tie.

### ATTACH THE NEOPIXEL STRIP

Now, let's secure the NeoPixel strip to the bottom of the bag, and give it some protection at the same time. Place the strip on the bottom of your bag, close to the side that the Circuit Playground Express is mounted to. Ensure that the wires flow neatly next to the strip and up the middle of the front of the bag to the Circuit Playground Express. These will be visible, so make them as tidy as possible.

Cover the NeoPixel strip with a strip of clear, heavy-duty packing tape or clear Gorilla tape. Press the tape down well on all sides of the strip and be sure to press out any air bubbles. The clear tape disappears almost completely when the lights are on. Use another piece of the clear tape to cover the wires running up the front of the bag, and to keep them from getting jumbled up and snagged on things inside your bag.

### INSTALL THE BATTERY PACK

It's time to add the battery holder! The battery holder will live inside the bag, and the Circuit Playground Express is mounted to the outside, so we'll need to cut one more hole in the front of the bag for the battery holder's wires to pass through. Using a craft knife, cut a slit about 1 cm long below the CPX's JST port. With the battery pack on the inside of the bag, push its JST plug through the slit, to the front of the bag. Depending on the thickness of your plastic, this could be tricky, and you may need to cut a fully fledged hole to get it through.

Insert batteries into the holder, and plug the JST connector into the port on the Circuit Playground Express. Use the on/off switch on the battery holder to turn your circuit back on and off.

Now let's make everything light up with code! This code is a little more complex than our previous projects, but by the end you'll know how to extend the abilities of your Circuit Playground Express by adding external outputs, like NeoPixel strips, to pins. To get started, connect the CPX to your computer with a micro USB cable. Then open your browser and navigate to **makecode.adafruit.com**, and start a new project.

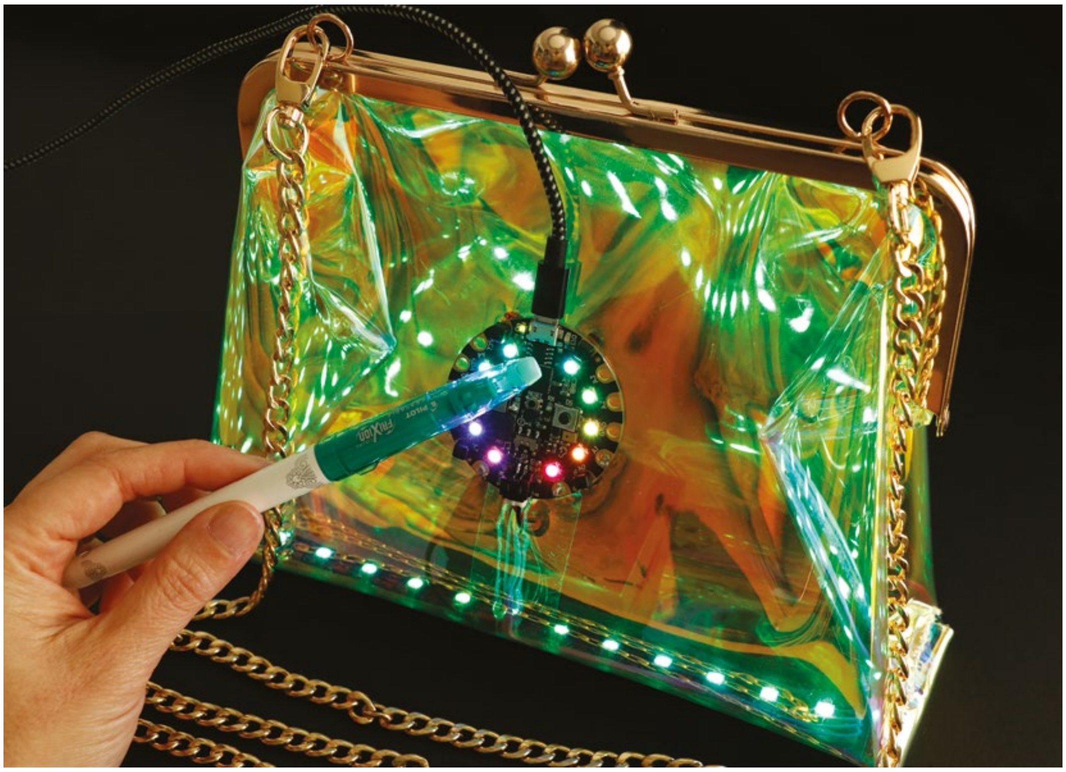
### SET UP

We won't be using the 'forever' loop in this program, so delete it from the workspace – the easiest way is to just drag it off to the left. However, we will need to set some actions to happen at the beginning of the program, so open the Loops menu and drag an 'on start' loop onto the workspace.

The light sensor and built-in LEDs are set up by default in MakeCode, but we need to create a new variable for our NeoPixel strip so we can work with it in our program. Go into the Variables menu, click 'Make a Variable' and name it 'strip'. You'll now →

**Below**
The LED strip fits unobtrusively in the bottom of the clutch

## CODING



**Right** Match your clutch bag with clothes or accessories

see that a few new items have appeared in the Variables menu.

Now click on the Light menu to reveal the nested NeoPixel submenu underneath it. Click on the latter to view more options for strips. Find the block named 'set strip to create strip on A1 with 30 pixels' and drag it onto the workspace and inside the 'on start' loop. From now on in our code, the variable 'strip' will be the placeholder for the NeoPixel strip we have connected to pin A1 of the CPX. Change the number 30 to the actual number of LEDs on your strip (ours was 12).

" Depending on the amount of light in your environment, you may need to experiment with these settings to get a **good reading from the light sensor** "

One other setup item we need to address is setting the thresholds for the light sensor. Depending on the amount of light in your environment, you may need to experiment with these settings to get a good reading from the light sensor. Even if yours works great right from the start, it's good to know where these settings are, because chances are you'll need to tweak them eventually.

The light sensor is an input, so head into the Input menu and scroll down to find the 'set dark threshold value to 0' block. Drag this block into the 'on start' loop, and just leave the value at 0 for now. Duplicate the block by right-clicking and selecting 'duplicate'. Drag the new block into place right underneath the first one, and click on the drop-down menu to change 'dark' to 'bright'. Leave this one at 0 for now also. Later on, if you've downloaded the code to your Circuit Playground Express and the sensor doesn't seem to be responding correctly, you can adjust these threshold values until the sensor works well in your particular lighting conditions.

### MAKE A STARTUP SEQUENCE

Now let's create a startup sequence so that we know the lights are working when we turn on the Circuit Playground Express. Head into the Light menu and grab the 'show rainbow animation for 500 ms' block. Add it to the 'on start' loop, and click on the drop-down menu to change the duration from 500 ms to 1 second. In the Circuit Playground Express simulator, you'll see that we just programmed the on-board NeoPixels to show a rainbow animation. Let's also make the NeoPixel strip on pin A1 show the same animation.

Click on the Light menu again to get to the NeoPixel submenu, and find the 'strip show rainbow animation for 500 ms' block. Drag it into the 'on start' loop under the previous animation, and change the duration to 1 second. Although you won't see it in the simulator, our NeoPixel strip will now show the same animation as the on-board NeoPixels.

Next, let's set all the pixels to black (off) and get ready to sample a colour. Go back into the Light menu, click on the NeoPixel menu, and grab the 'strip set all pixels to (red)' block. Drag it onto the workspace and drop it into the 'on start' loop. To change the colour setting to black, you'll need to head back into the Light menu, scroll down to the Colour section, and grab the round module labelled 'red'. Drag this module onto the workspace and drop it directly on top of the red colour block to replace it. You can then click on the word 'red' and select 'black'.

To set the on-board NeoPixel ring to black, simply duplicate this block, and drag the copy into the 'on start' loop under the original block. Head back into the Light menu, then the NeoPixel submenu and find the round module for 'onboard strip'. Drag this module onto the workspace, and drop it on top of the 'strip' variable to replace it with 'onboard strip'. That's it for setup!

## MAKE THE MAGIC HAPPEN

We're getting there! All that's left is to tell the Circuit Playground Express to read the light sensor and set the on-board strip to the same colour as it detects. Start by going into the Input menu and finding the 'on light dark' loop. Drag it onto the workspace. Now, anything we put inside this loop will happen when the light sensor is blocked by an object and it detects 'dark'.

Let's have it take a colour reading and display that colour on our NeoPixel strip. Head back into the NeoPixel submenu (remember, it is nested inside the Light menu) and grab the 'strip set all pixels to (colour)' block, and drop it into the 'on light dark' loop. Replace the colour in the block with the 'ambient color' module, which you'll find in the Input menu. Now, if you wave an object over the light sensor, the NeoPixel strip should change to match the colour of the object!

It's a magical effect, and we can embellish the magic by adding a few more actions. Let's make the on-board NeoPixel ring display the rainbow animation once the colour sensor takes its reading. From the Light menu, drag the 'show rainbow animation for 500 ms' block onto the workspace and add it to the 'on light dark' loop. The Circuit Playground Express can also play sounds, so let's add a magic wand sound as well. In the Music menu, grab the 'play sound power up until done' block and add it to the 'on light dark' loop. Click on the drop-down menu to change 'power up' to 'magic wand'. To return the CPX to its ready state, let's set the on-board pixels to black again. Simply duplicate the final block of the 'on start' loop, 'onboard strip set all pixels to black', and drop it into the bottom of your 'on light dark' loop. Your program is complete!

You can test this out on the Circuit Playground Express simulator by clicking and dragging on the yellow light sensor input icon to the left of the CPX image. You should see the on-board rainbow animation and hear the magic wand sound. If everything works as expected on the simulator, click Download, then follow the instructions in MakeCode to install the code on your Circuit Playground Express.

## HAVE FUN COLLECTING COLOURS!

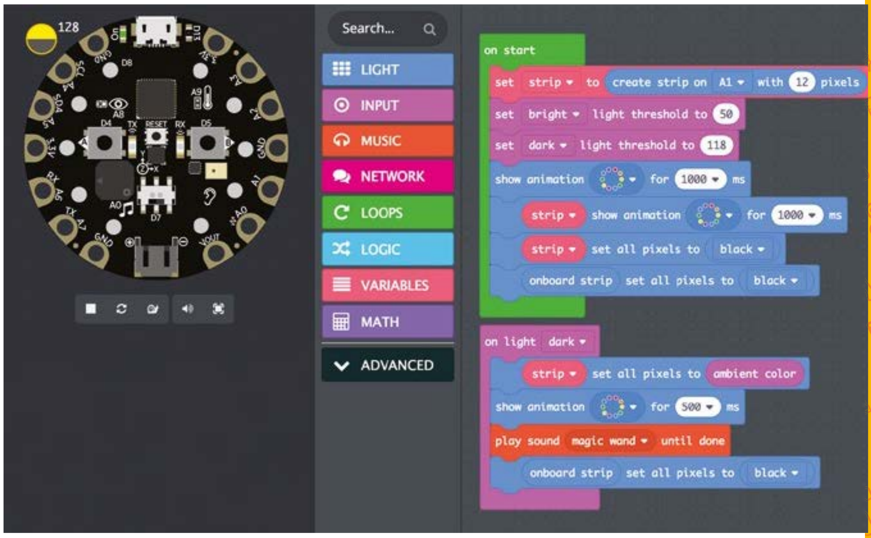
Now you're ready to sample colours from the world around you! Remember that it may take some experimenting with the light sensor threshold values to get a solid reading at first. Look for the eye icon on the Circuit Playground Express – hold the object in that area a few centimetres away from the board. When the sensor takes a reading, you'll hear the satisfying magic wand sound, and see your bag change colour. It's like magic, but better, because you made it!

We've just scratched the surface with sensors in this project, and because the Circuit Playground Express is easy to program on the front of your bag, you can modify this project in lots of ways. In addition to the light and colour sensor, the CPX also has an accelerometer, a temperature sensor, a sound sensor, and an infrared receiver and transmitter on board. How are you using these sensors in your projects? Show us at **@HackSpaceMag**! □

### QUICK TIP

Don't carry a clutch? You can apply the same circuit and code to almost any clothing or fashion accessory.

**Below** The MakeCode editor is easy to use and powerful enough for our needs



```
on start
  set strip ▾ to create strip on A1 ▾ with 12 pixels
  set bright ▾ light threshold to 50
  set dark ▾ light threshold to 118
  show animation ▾ for 1000 ▾ ms
    strip ▾ show animation ▾ for 1000 ▾ ms
    strip ▾ set all pixels to black ▾
    onboard strip set all pixels to black ▾

on light dark ▾
    strip ▾ set all pixels to ambient color
  show animation ▾ for 500 ▾ ms
  play sound magic wand ▾ until done
    onboard strip set all pixels to black ▾
```

**Above** Keep your phone topped up with power while on the move

# Smart backpack with wireless charging pocket

Never run out of power while out of the house again

**Many mobiles now come with wireless charging capabilities, allowing you to charge your phone by simply placing it over a base – no cords or plugs needed!** But what if you could charge your phone while out and about by simply slipping it into a pocket? In this project, we'll add a wireless charging pocket to a backpack for holding a portable Qi charger. A portable charging bank is also a great source of power for wearable projects, so we'll add a strip of NeoPixels to act as a visibility light and status bar. The light will glow when the charger is on, so you'll know when power is flowing!

Wireless chargers use resonant inductive coupling to transmit energy from one magnetic coil to another. The coil in the charger is the transmitter, and the coil in the phone is the receiver. While there are several standards for wireless charging, 'Qi' is currently the most widely adopted standard, and you're likely to find many portable wireless chargers labelled as 'Qi Chargers'. Most of these chargers come with USB ports for plugged charging, which can power a 5V microcontroller like an Adafruit Trinket M0. So, how could we resist adding some NeoPixels to the mix?

If you have a newer Samsung phone or iPhone, it may well be Qi-enabled and ready for wireless charging. If you have an older phone that doesn't have Qi technology built in, there's good news: you can add a wireless charging adapter pad to your phone! These adapters add an adhesive receiver coil to the back of your phone, and plug into your phone's charging port. They'll work just fine for this project!

The wireless charging pocket will be visible from the outside of the bag, so you'll want to choose fabric that will coordinate well with the design of your backpack. Fabric with a little bit of stretch in it will work well here, as the phone needs to fit snugly into the pocket so that it is held tight against the charger. A sewing machine will make quick work of the pocket, so use it if you have one, but hand-sewing is fine too. When attaching the pocket to the bag, you will need to hand-sew it in place if the location you choose is inaccessible by sewing machine.

Cotton bone casing is used in corset making to hold boning in place on the inside of the corset. It's a great material to have handy when using NeoPixel strips with wearables, because the standard NeoPixel strips fit inside perfectly. The silicone weather-proof strip that NeoPixels come with is a bit tricky to attach to fabric, but the cotton casing is easy to sew or glue to fabric. If you can't find corset bone casing, you can make your own casing by sewing a strip of cotton into a tube and ironing it flat. Bonus: white cotton casing diffuses NeoPixels nicely into a glowing bar of light!

" The wireless charging pocket will be visible from the outside of the bag, so you'll want to choose **fabric that will coordinate well with the design of your backpack** "

Adding a microcontroller gives you lots of options for adding 'smart' functionality to your bag. We'll use a Trinket M0 and NeoPixels from Adafruit to make a pulsing blue strip of light – perfect for visibility at night, or just adding a bit of sci-fi to your outfit. We'll program the Trinket with CircuitPython, and you can easily update the code when you think of a new feature to add to your bag. For protection from the elements, all electronic components will be located inside the bag, and it's a good idea to spray the NeoPixel casing with a protective hydrophobic coating like Scotchgard. →

**Above** Measure the strip against the bag to know where to cut

## YOU'LL NEED

- **Backpack or bag**
- **Portable Qi charging bank**
- **Fabric for pocket** (find something with stretch that co-ordinates with your bag)
- **Waistband elastic,** 5 cm wide
- **Hand-sewing supplies**
- **E6000 or other strong, flexible glue**
- **Trinket M0 microcontroller**
- **NeoPixel strip** about 30 cm
- **¾" wide white cotton bone casing**
- **Silicone-coated stranded wire**
- **Scotchgard, or other hydrophobic spray coating**

## DESIGN YOUR CIRCUIT

Start by working out where all of your components should go. It's best to place the light strip close to your Trinket, and the Trinket close to your charger, so that wires don't have to pass across any zip openings. A front pocket is a great choice for housing the charger and the Trinket, and the LED strip works well on a flat area, like the top flap of the pocket. The wireless charger should be able to charge through one or two layers of thin fabric, but do a quick test by holding the phone and charger in place before committing to a location. Take into account the risk of pickpockets in your local area when deciding where to put the pocket.

## MAKE THE LIGHT STRIP

Measure the NeoPixel strip against your bag to determine how long it should be. Leave about 1 cm of space on each end for the casing to attach to the bag. Use scissors to cut the strip to the correct length. Cut directly through the contacts. It's a good idea to round the corners so they don't snag on your fabric.

Note the direction of the small arrows on the NeoPixels: they point in the direction that data flows through the strip. Cut three pieces of silicone-coated stranded wire about 7–8 cm long, and solder the wires to the 'data in' side of the strip (the side the arrows point away from). Apply a bead of hot glue over the solder points for reinforcement. We'll cut the excess length off of these wires before we solder them to the Trinket.

Cut a strip of the cotton casing so that it is 4 cm longer than the NeoPixel strip. Fold 1 cm of the strip back on one end of the casing and sew it closed. Insert the 'Data Out' end of the strip into the casing so that the wires extend out of the open end of the casing. Gently fold the wires toward the back of the casing, making sure not to bend at the solder points. Be very careful not to damage the NeoPixel strip during this step. Sew down the fold with a few hand-stitches on either side of the wires, as shown.

## MAKE THE POCKET

Lay out your stretch fabric right side down, and place your phone on top. Trace around your phone with a marker. For a good connection, we need to make sure the phone does not move around inside the pocket, so trace tightly around your phone for a snug fit.

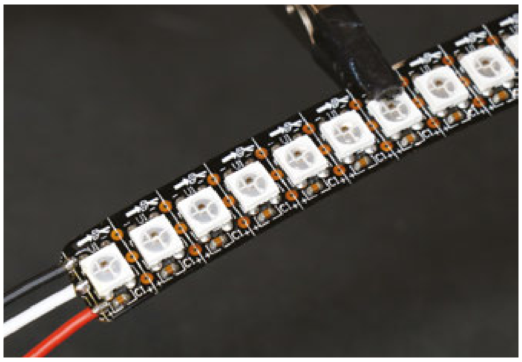
Add 2 cm to each side for seam allowance, and cut out your pocket. To keep the edges from unravelling, zigzag over all four edges with a sewing machine; if you don't have a machine, you can run a bead of Fray Check along the edges and let it dry. Fold each edge in by 1 cm and stitch in place ½ cm from the edge. Check to make sure your pocket is just about 1 cm bigger than your phone on all sides. If it looks good, set it aside and move on to making elastic straps for the charger.

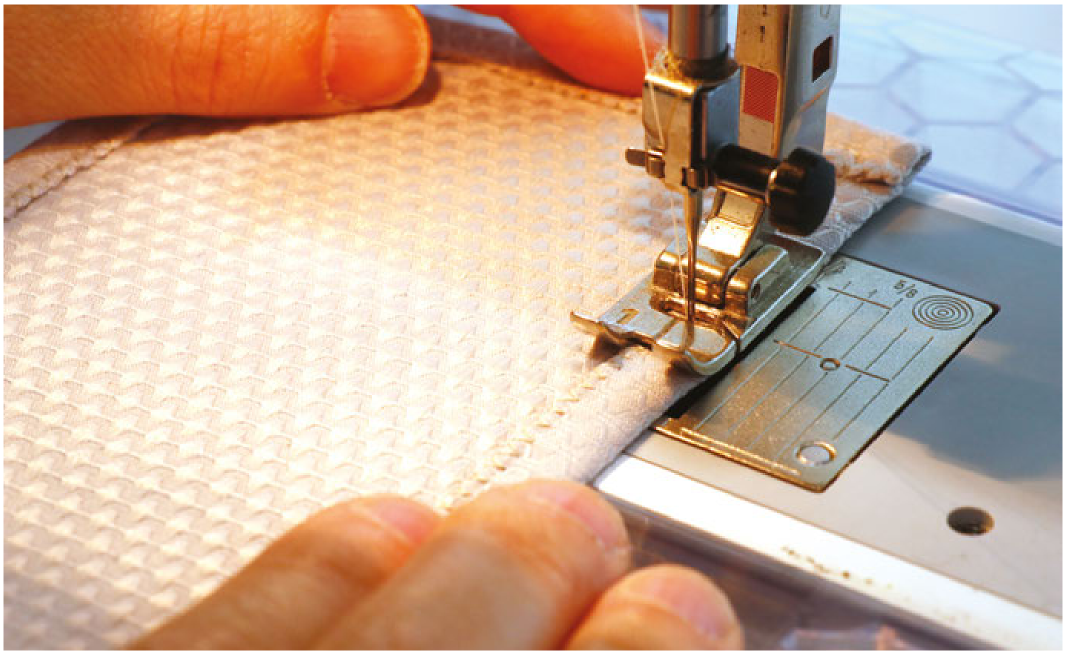
## MAKE A STRAP FOR THE CHARGER

To hold the charger tight against the front of the bag, we'll make an elastic strap for the charger to slide into. This strap will be sewn into a 'T' shape, and be attached to the inside of the bag, directly behind our new phone pocket. Cut a piece of elastic that is about 4 cm wider than your charger is, and lay it across the centre of the charger. Cut another piece of elastic to extend from the middle of this piece to about 4 cm beyond the bottom edge of your charger. Use a lighter to carefully melt the cut edges of the elastic to keep them from unravelling.

Next, pin the pieces together in their 'T' configuration. To sew the pieces together securely, stitch in a rectangle shape, then sew an 'X' inside the rectangle. Check that your strap will fit over the charger with at least 1 cm of elastic on each side for attaching to the bag.

**Below** You only need to connect one end of the strip to control all the NeoPixels





**Left**
Like many things, sewing is quicker with power tools, but still possible by hand

## SEW THE POCKET IN PLACE

Now that we've made all the individual pieces, we're ready to attach the pocket and charger straps to the bag. Start by pinning the strap in place on the inside of the bag. Test the fit by slipping the charger into the strap. Adjust the pins so that the charger is held tightly against the bag and doesn't slide around. Using a backstitch, hand-sew the elastic in place from the outside of the bag. These stitches will be covered by the phone pocket and won't show when we're done.

Pin the phone pocket in place on the outside of the bag. Slide your phone into the pocket, facing forward. To check the placement of the pocket, slip the charger into its strap, with the charging coil facing the phone. If your phone starts charging, you're ready to sew your pocket in place! Otherwise, adjust the placement of the pocket so that the back of your phone aligns properly with the inductive coil area on the charger.

Remove the charger, and hand-sew the pocket in place with a backstitch ½cm from the edge. This seam will be visible, so take your time for even, straight stitches. Your phone should fit into the pocket snugly, and align with the charger when it's in place. If everything lines up properly, it's time to move on to the NeoPixels!

## INSTALL THE LIGHT STRIP

We need to make a hole for the wires to pass through to the inside of the pocket, where the Trinket will be located. Hold the NeoPixels in place on the front of the bag, making sure the wires are on the same side of the bag as the charger. Mark the placement of the wires on the pocket flap, then set the NeoPixel strip aside. Use a hot soldering iron to make the hole, and widen it enough so that all three wires will fit through. Feed the wires through to the inside of the pocket.

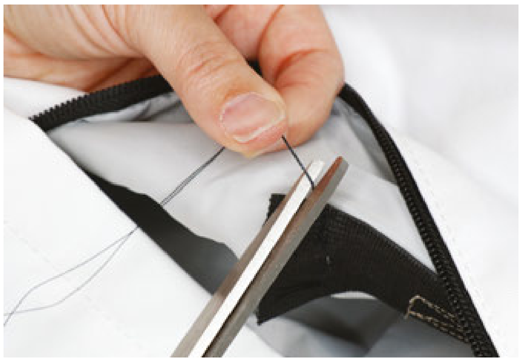
For added weather-proofing, you can spray the front of the casing with a hydrophobic coating, like Scotchgard. Glue the casing in place with a strong, flexible glue, like E6000. Hold the strip in place with clips or clothes-pegs while the glue dries. When the glue is dry, solder the wires to the Trinket: connect positive on the NeoPixels to USB on the Trinket, negative to GND, and 'data in' to pin 0.

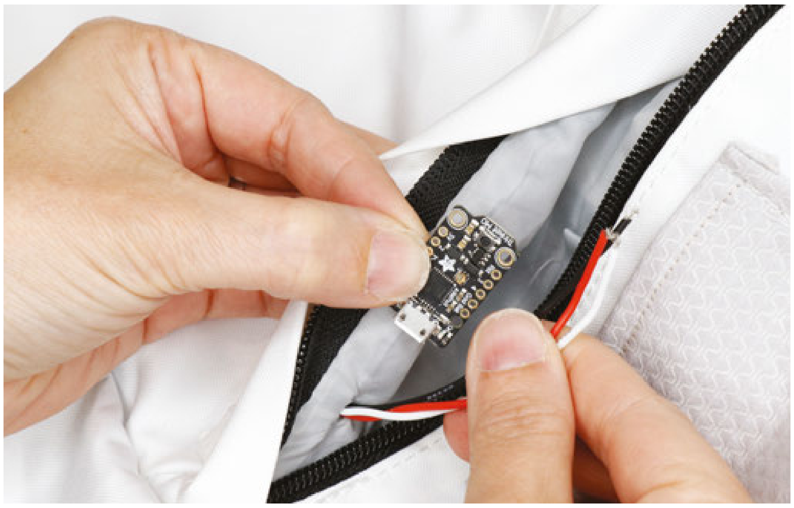
## PROGRAM THE TRINKET

Next, we'll program the Trinket M0 to animate the NeoPixel strip whenever it is on. Powering the Trinket from the charger means that the strip will act as a status bar, letting you know that the charger is on. The following code will make the strip slowly pulse in bright blue, and since the Trinket is easy to access, →

**Below**
Keep fraying elastic at bay with a brief melting

## CODING


**Right**
Keep your sewing tidy and you'll end up with a great-looking bag


**Above**
A small board like the Trinket can be hidden almost anywhere

you can change up the animation with new code at any time.

Connect the Trinket to your computer using a micro USB cable. The Trinket will appear as a drive on your computer called 'CIRCUITPY'. On the drive, locate the file called **main.py**. This is a text file that contains the code running on the Trinket – to reprogram the Trinket, we'll edit this text file. You can edit the file in any text editor, or download a Python code editor like Mu. Saving the **main.py** file will make it run automatically on the Trinket M0.

Open the **main.py** file in your text editor. If your Trinket M0 is fresh from the Adafruit factory, it will have demo code in it. Select all of the existing code, and delete it – we're going to start fresh!

The first thing we need to do is import some libraries. These libraries contain helpers that we'll refer to in our code.

```
import time
import board
import neopixel
```

Next, we'll set up some variables. These placeholders will be useful for modifying and troubleshooting the code later. Start with `numpix`, and change the number 34 to the number of NeoPixels in your strip. `pixpin` identifies the pin we've connected our NeoPixel strip to, which is D0. The `strip` variable creates the NeoPixel strip as an object, and sets the brightness to .5. You can change the brightness level as needed for the look you want. `strength` and `direction` are variables that are used to create the pulse effect; leave them at 0 and 10.

```
numpix = 34  # Number of NeoPixels
pixpin = board.D0  # NeoPixels pin
strip = neopixel.NeoPixel(pixpin, numpix,
brightness=.5, auto_write=False)
strength = 0 # blue intensity
direction = 10  # direction of intensity
```

In the main loop, we'll create one more variable for the colour `BLUE`. The three values inside it are for red, green, and blue levels, from 0 (no colour) to 255 (maximum colour). Set red and green at zero, and use the variable `strength` for blue. By changing the value of `strength` in our main loop, we'll be able to make the blue glow change over time.

```
while True:
    BLUE = (0, 0, strength)
```

We'll send this colour formula to the strip with:

```
    strip.fill(BLUE)
    strip.show()
    time.sleep(0.03)
```

At this point in the loop, the colour formula has been sent to the strip, but we won't see it yet because the `strength` variable is still set to zero. To increase the blue colour slightly over time, we'll add:

```
    strength += direction
```

This line increases the `strength` value by adding the `direction` value to it. On the first pass through the loop, `strength` = 0 and `direction` = 10, so the new value of `strength` would be 10. On every additional pass through the loop, the value will increase by 10, and the strip will show a blue glow that intensifies slowly over time.

Next, we need to add upper and lower limits to this function. A conditional statement will work great for this. First, let's say that if the `strength` value ever gets all the way down to zero, it should be reset to 10. At that point we'll also want the intensity to stop getting lower and start going up, so we'll flip the `direction` value to its opposite:

```
    if strength <= 0:
        strength = 10
        direction = -direction
```

Finally, let's set the upper limit to 255. If the **strength** value is ever larger than 255, we'll have it reset to 250 and flip the **direction** value to start going in the other direction:

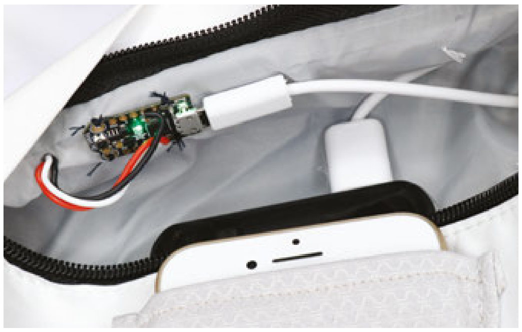
```
    elif strength >= (255):
        strength = 250
        direction = -direction
```

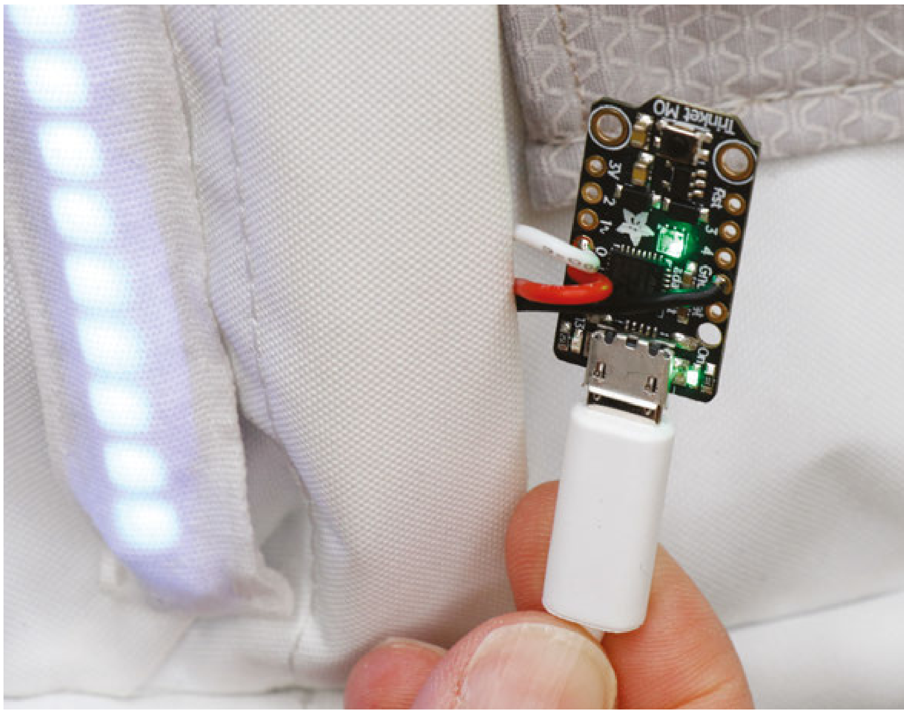
That's it for the code! Save the file to your Trinket M0, and watch your NeoPixel strip pulse. To change the speed of the pulsing, adjust the **time.sleep** value, or change the **direction** value. When you are happy with your animation, move on to installing the Trinket.

For troubleshooting help and more tutorials about programming the Trinket M0 with Circuit Python, check out **hsmag.cc/zucqvz**. Here's the full code listing:

```
import time
import board
import neopixel

numpix = 34  # Number of NeoPixels
pixpin = board.D0  # NeoPixels pin
strip = neopixel.NeoPixel(pixpin, numpix,
brightness=.5, auto_write=False)
strength = 0 # blue intensity
direction = 10  # direction of intensity

while True:
    BLUE = (0, 0, strength)
    strip.fill(BLUE)
    strip.show()
    time.sleep(0.03)

    strength += direction

    if strength <= 0:
        strength = 10
        direction = -direction
    elif strength >= (255):
        strength = 250
        direction = -direction
```

**Below** With everything in place, you're good to go out and wear your power-bank bag with pride





**Above** USB power banks can power a wide variety of things, and don't need soldering

## INSTALL THE TRINKET

We're almost done! All that is left is to securely attach the Trinket near the charger. Sew the Trinket in place through the four mounting holes located in the corners of the board. Use a short mini USB cable to connect the Trinket M0 to the charger, and turn the charger on. If your NeoPixel strip lights up and pulses in blue, you're done! If not, check your connections, and make sure your charger is charged. Also, check that your phone charges when inserted into its pocket.

## WEAR IT!

You're now ready to take your smart backpack out for a spin! Turn the charger on for visibility while out and about at night, and slip your phone into its pocket whenever you need some juice. There are a few more pins available on the Trinket M0, so you can keep adding functionality to your backpack. An accelerometer could help the lights pulse in time with your steps, a GPS module could change the colour based on your speed, or speakers could… well, we will leave ix0t up to you. However you decide to use this project, be sure to let us know at **hackspace@raspberrypi.org**!

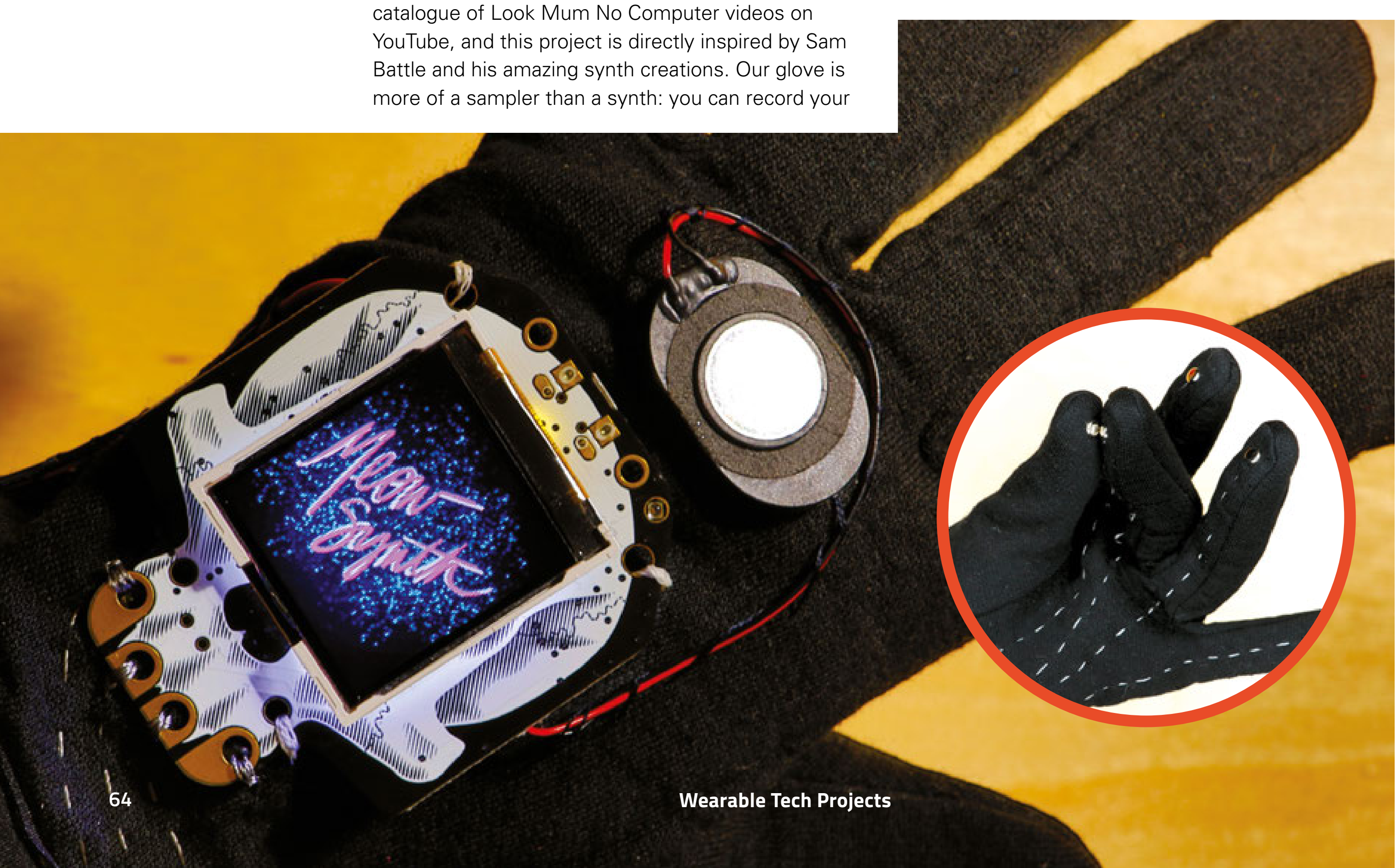
# Handy Sampler – sound glove with HalloWing

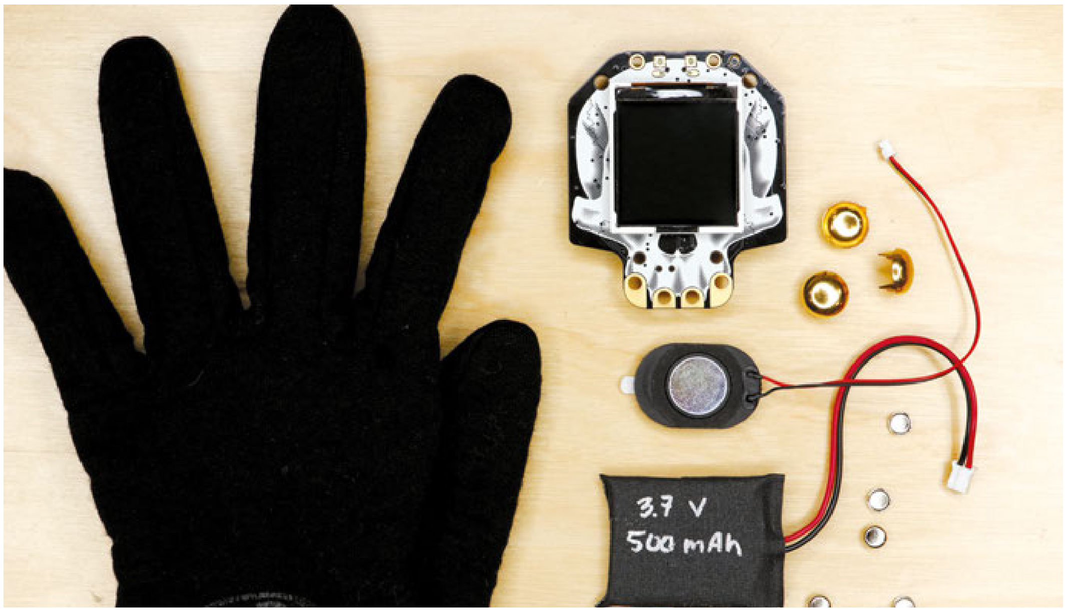
Create your own wearable audio sampler glove!

**We couldn't wait to get our hands on the new HalloWing M0 Express from Adafruit. Based on the Feather M0 Express microcontroller, the HalloWing comes packed with on-board features like a 1.44" full-colour TFT display, accelerometer, light sensor, speaker driver, and JST ports for connecting NeoPixel strips, sensor inputs, and more.** All of this, plus a built-in on-off switch, means the HalloWing is just asking to be put into a wearable project – that's what we did. Behold the Handy Sampler: a wearable audio sampler glove!

Your author has been burning through the back catalogue of Look Mum No Computer videos on YouTube, and this project is directly inspired by Sam Battle and his amazing synth creations. Our glove is more of a sampler than a synth: you can record your own sounds and load them into the 8MB of SPI flash on the HalloWing. We added capacitive touch sensors to the tips of the fingers for triggering each sound. Just press your thumb and finger together to trigger one of four pre-recorded sounds. And the TFT display shows your awesome synth-pop band logo, or a picture of your cat, or anything else you can fit into a 128×128 pixel BMP image.

To record your samples, you can use audio software like Audacity, which is free. Your author recorded her own voice saying things like 'boing', 'boop', and 'meow' and saved each sound as a separate file to create a custom sample pack. Make sure to save your samples as 16-bit mono WAV files.





**Left**
When assembling your components for a wearables project, it helps to pick ones that look good

**Below**
Mark out your traces to avoid confusion when sewing

To create a graphic for the screen, your author sketched a quick band logo in Photoshop and saved it as an RGB BMP graphic. The screen is bright, but small, so flat colours and simple shapes will look best here. The final image should be saved as a 128×128 pixel 24-bit colour BMP file.

You will definitely want to create your own samples and image (have fun with this!), but to get started quickly, you can download our sample pack and the Meow Synth band logo image from here: **hsmag.cc/book2-assets**.

Due to the HalloWing's handy ports, you won't need to solder anything to build this glove. The speaker plugs right in, and we'll use conductive thread to sew traces to each finger for capacitive touch inputs. Small metal rivets make great capacitive sensors and add a little bling to the cyberpunk aesthetic. Use your multimeter to check your rivets and make sure they're conductive. You won't need a sewing machine either, as this project is hand-sewn. Sewing components to the glove may seem more tedious than glue, but stitches hold more securely and are easy to remove – which means you can repurpose your HalloWing for another project later on.

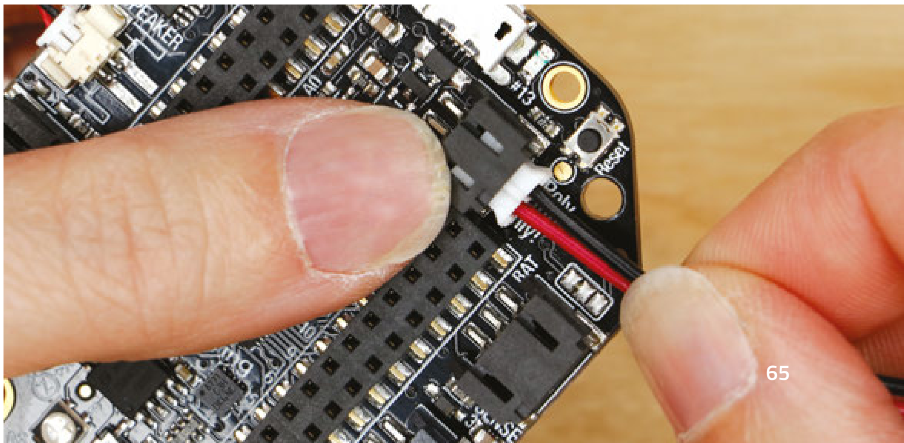
The HalloWing requires a 3.7 V LiPo battery for untethered power. LiPo batteries are tiny powerhouses that are great for wearables, but require proper handling and care for safe use. Do not puncture, compress, heat, or otherwise abuse these batteries. Ensure that you are using the correct charger for your battery, specified by the manufacturer, as the chemistry of these batteries can vary by supplier. Adafruit stocks a special 400 mAh LiPo battery that is specifically designed to fit between the headers on a Feather (or HalloWing); if you choose to use that battery, it's a simple matter of nestling it into the underside of your HalloWing. If you only have a larger LiPo battery on hand, we'll show you how to sew a custom pocket onto your glove that will fit your battery perfectly.

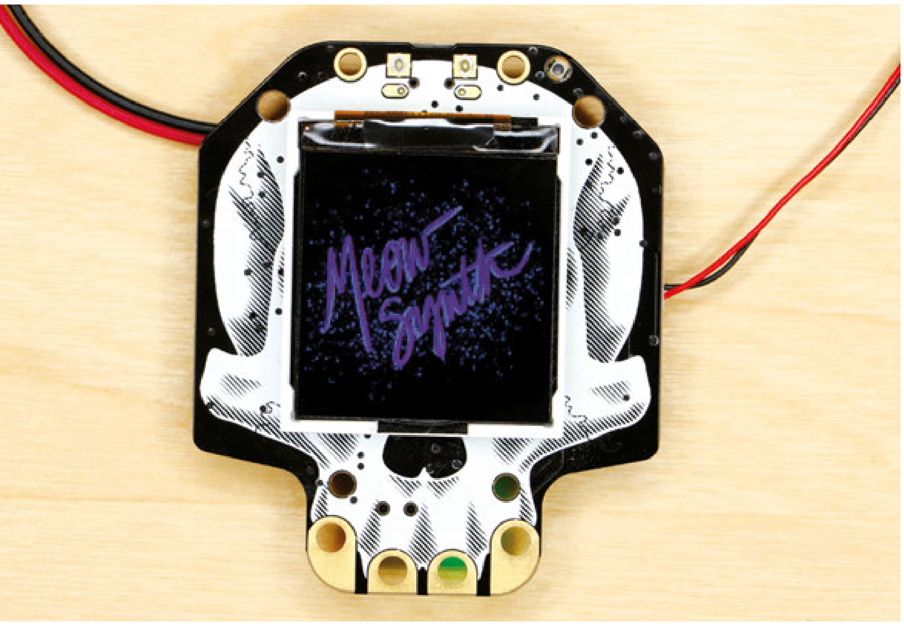
## DESIGN YOUR GLOVE

Choose a thin glove that is a snug, but comfortable fit on your hand. A natural fibre with just a little bit of stretch works nicely. You'll be sewing the traces of your circuit on the glove while wearing it, so it's easiest to build this glove for your non-dominant hand.

Plug the speaker into the speaker port, and the battery into its JST port. With the components connected, work out where everything will go on your glove, and draw out the paths for your circuit. →

**Below**
Solderless connectors make it easy to reuse components for other projects





**Above** A little creative work on the screen helps personalise this glove

Keep in mind that the on/off switch is on the back of the HalloWing, so you'll want to keep that accessible somehow. We cut a tiny hole in our glove so that the HalloWing can be switched on or off from the inside.

Draw lines from each of the four capacitive touch 'teeth' of the HalloWing to the tips of the four fingers on the underside of the glove (not the thumb). The sewn traces must not touch each other or pass under the wires of the battery or speaker. Although they are insulated, the wires can cause false triggering of the capacitive touchpads. It's best to keep everything separated on your glove as much as possible. On our glove, we routed all of the sewn traces around one side of the glove, which allowed the battery wire to run down the opposite side.

" **The first thing you'll want to do when you get your HalloWing** is turn it on and see what it can do "

## PROGRAM THE HALLOWING

The first thing you'll want to do when you get your HalloWing is turn it on and see what it can do! Then, plug the HalloWing into your computer with a micro USB cable. The HalloWing comes with a cool spooky eye animation (running on Arduino code by Phil Burgess), so ogle that for a few minutes before reformatting it with CircuitPython. Double-click the reset button on the HalloWing to enter bootloader mode; it will appear as a drive called HALLOWBOOT.

As always, the first step is to make sure that CircuitPython and the libraries on the board are up to date. This project requires a special version of CircuitPython written for HalloWing that allows for use of the TFT screen, so download the latest 4.x HalloWing-specific version of Circuit Python from Adafruit's GitHub repo: **hsmag.cc/GwKXXY**. Drag this .uf2 file onto HALLOWBOOT. This will install CircuitPython and automatically restart the HalloWing, renaming it CIRCUITPY. Note: CircuitPython 4.0 is currently in beta release, so things may change as it develops.

You'll also need the latest CircuitPython libraries for 4.x, available here: **hsmag.cc/xAHysT**. Since the HalloWing has plenty of space for it, it's easiest to drag the whole **lib** folder onto the CIRCUITPY drive.

In addition to libraries, this project uses four sounds and one image from our downloadable project pack: **meow_01.wav**, **meow_02.wav**, **meow_03.wav**, **meow_04.wav**, and **meowsynth.bmp**. Place these individual files onto the CIRCUITPY drive at the root level.

Although you can create and edit CircuitPython code with any text editor, it's best to use a code editor like Mu, which has helpful code-checking features and a REPL for debugging. In your editor of choice, create a text file called **code.py** on the CIRCUITPY drive. Now you're ready to code!

First, we'll import the libraries we need and set up the WAV files we'll be using in the code. This is where you can change the file names to other samples in the sample pack or your own recorded sounds.

```
import board
import audioio
import touchio
import pulseio
import displayio
import time

sound1 = open("meow_01.wav", "rb")
sound2 = open("meow_02.wav", "rb")
sound3 = open("meow_03.wav", "rb")
sound4 = open("meow_04.wav", "rb")

wav1 = audioio.WaveFile(sound1)
wav2 = audioio.WaveFile(sound2)
wav3 = audioio.WaveFile(sound3)
```

Next, we need to set up the capacitive touchpads as inputs, and create a variable for the capacitive touch threshold. This threshold value determines how sensitive the capacitive touchpads are – a higher value makes the pads less sensitive. Because there is a lot of exposed conductive thread running all around the glove, we found that we needed to set this value fairly high. You'll need to tune this value for your particular glove.

```
capThreshold = 3500

TOUCH1 = touchio.TouchIn(board.A2)
TOUCH1.threshold = capThreshold
TOUCH2 = touchio.TouchIn(board.A3)
TOUCH2.threshold = capThreshold
TOUCH3 = touchio.TouchIn(board.A4)
TOUCH3.threshold = capThreshold
TOUCH4 = touchio.TouchIn(board.A5)
TOUCH4.threshold = capThreshold
```

Now, we'll add the speaker to A0 on the board, and set the TFT display to show our image. If you have a different image to show, update this section with your file name.

```
speaker = audioio.AudioOut(board.A0)

backlight = pulseio.PWMOut(board.TFT_BACKLIGHT)
splash = displayio.Group()
board.DISPLAY.show(splash)

with open("meowsynth.bmp", "rb") as f:
    odb = displayio.OnDiskBitmap(f)
    face = displayio.Sprite(odb, pixel_
shader=displayio.ColorConverter(), position=(0,
0))
    splash.append(face)
   board.DISPLAY.wait_for_frame()

   for i in range(100):
        backlight.duty_cycle = i * (2 ** 15) //
100
        time.sleep(0.01)
```

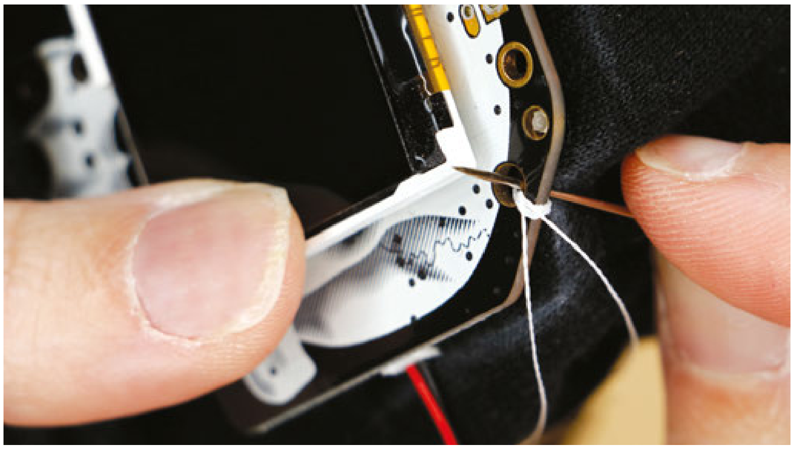
Our main loop consists of a four-part conditional statement. If any of the four capacitive touchpads are touched, the HalloWing will play one of the four sounds we loaded in the setup of our code. For good measure, we ask it to tell us in the serial monitor which sound it is playing. To keep the sounds clean, we ask it to **pass** while the sound is playing – it will not trigger another sound until the first sound stops.

```
while True:
    if TOUCH1.value:
        print("playing 1")
        speaker.play(wav1)
        while speaker.playing:
            pass
        print("stopped")

    elif TOUCH2.value:
        print("playing 2")
        speaker.play(wav2)
        while speaker.playing:
            pass
        print("stopped")

    elif TOUCH3.value:
        print("playing 3")
        speaker.play(wav3)
        while speaker.playing:
            pass
        print("stopped")

    elif TOUCH4.value:
        print("playing 4")
        speaker.play(wav4)
        while speaker.playing:
            pass
        print("stopped")
```

That's all there is to our code! Save this **code.py** file on your HalloWing, and the device will automatically restart and run the code. You should see the Meow Synth image fade in on the screen. Touch each capacitive pad (the gold 'teeth' at the bottom of the HalloWing) and make sure the sounds play correctly. For more information and help troubleshooting HalloWing, check out Adafruit's HalloWing guide at **hsmag.cc/OXwaRc**. When everything works, move on to building the glove!

## ATTACH THE COMPONENTS

Place the HalloWing on its spot on the glove, and sew it down with regular thread. There are four →



## YOU'LL NEED

- **Adafruit HalloWing M0 Express**
- **8 Ω speaker with 1.25 mm 2-pin Molex PicoBlade connector**
- **3.7 V 500 mAh or 400 mAh LiPo battery**
- **Glove**
- **Small piece of stretchy fabric**
- **Conductive prong-style metal studs or rivets and rivet tool**
- **Conductive thread**
- **Hand-sewing supplies**
- **Hot glue gun and glue sticks**

**Below**
Despite the advances of modern technology, sewing is still a great skill to have

**Above** Be careful not to pierce the wire when stitching

mounting holes that are perfect for this. Stitch through each mounting hole, and tie off each separately with a tight knot. Be sure to avoid accidentally sewing through both sides of the glove when doing this.

Next, move your speaker into its location, winding the wire into a gentle spiral or curve if it is too long. The speaker doesn't have mounting holes, so use a little hot glue to hold it in place. Then whip-stitch the wire to the glove so that it won't catch on anything.

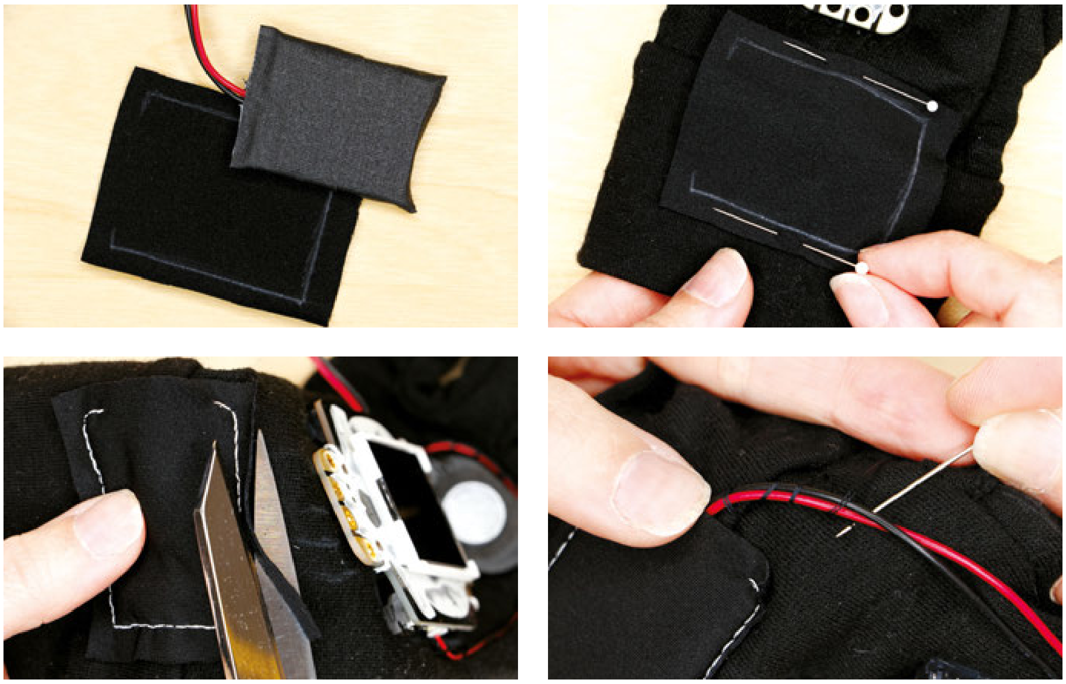
Now we need to make a pocket to hold the battery. Cut a small piece of four-way stretch fabric about 0.5 cm bigger than your battery on all sides. Trace the outline of your battery onto this fabric – this will be your stitching line. Pin this fabric in place where your battery will go. Keep in mind which side of the battery your wires extend from and the path they will take to get to the HalloWing. Pin the fabric in place on the glove.

With regular thread, sew the battery pocket down along the line on three sides, leaving the fourth side open for the battery to slide in and out. A strong back-stitch is a great choice for this seam. Sewing a few stitches beyond the two open corners helps keep the battery from falling out. Trim the edges of the fabric neatly.

Slide the battery into place, then sew down the battery wires with a whip-stitch. You're ready to switch to conductive thread and start sewing the rest of the circuit!

## SEW THE TRACES

Now that all the components are attached to the glove, it's time to sew traces to each finger tip. Thread your needle with a long piece of conductive thread (it will have to reach from the HalloWing to the tip of the finger). Start by sewing several tight stitches around one capacitive touch 'tooth' of the HalloWing, then continue to sew a running stitch along your drawn path toward the tip of the finger. When you start sewing on the underside of your glove, slip the glove on and continue sewing the trace while wearing it. Your glove is stretchy, but conductive thread does not stretch. Wearing



**Right** A little care when assembling can make a big difference to the finished product the glove while you sew will ensure that it still fits you once the trace is sewn, and you won't rip any stitches out when you try to put it on later.

When you get to the fingertip on the underside of the glove, sew several stitches over each other to make a tiny ball of conductive thread. Tie off your thread tightly, and cut the thread tail short.

Repeat this process for the other three capacitive touchpads on the HalloWing. Remember that none of your conductive thread traces can touch each other, or any other component in your glove. Trim thread tails short, especially at the capacitive touchpads where they are close together and are likely to touch. When all your traces are sewn, check to see if everything is connected correctly before moving on.

"**To trigger the sounds on this glove,** you'll need to create a capacitive point on the thumb as well"

Slip a piece of paper (or something non-conductive) inside your glove and turn it on. Touching the thread at the tip of each finger should trigger a sound.

## ADD CONDUCTIVE RIVETS

When you're happy with your sewn circuit, it's time to add rivets to the fingertips. The rivets make nice targets for you to aim for when triggering the sounds with your thumb. To apply the rivets, it's easiest to use a rivet fastener tool. Slip the concave 'anvil' of the tool into the fingertip under the thread ball. Place the rivet over the thread ball and push the prongs into the fibres of the glove. Align the rivet tool over the rivet and tap it a few times with a hammer to close the prongs of the rivet. Repeat this to apply a rivet to each finger.

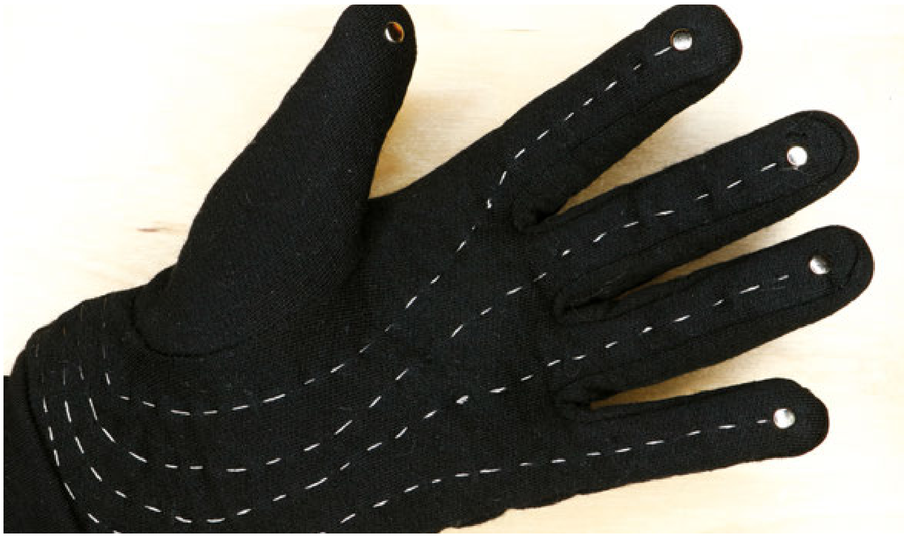
To trigger the sounds on this glove, you'll need to create a capacitive point on the thumb as well. This will simply connect your thumb's conductivity to the outside of the glove so that touching any of the four fingers to your thumb will trigger a sound. With another piece of conductive thread, sew a thread ball at the tip of the thumb. Again, it's easiest to do this while wearing the glove on your non-dominant hand. Then, cover this thread ball with another rivet.

## INSULATE THE CIRCUIT

All that's left to do is to insulate the inside of the glove so that you can actually wear it without constantly triggering sounds. To do this, carefully turn your glove inside out. Do another check, using a multimeter if necessary, to make absolutely sure none of your traces are touching. Then, cover all traces with hot glue to insulate them on the inside of your glove. When the glue is dry, turn your glove right side out again, and you're ready to play sounds!

## MAKE MEOWS (AND MORE!)

Now it's time to play! Slip on your glove, switch it on, and touch your thumb to each finger to hear a different meow. Once built, it's easy to upload new sounds to your Handy Sampler. Record new sounds, save them onto the HalloWing, and update the file names at the top of your code. And you don't have to stop there: we've just touched the surface of what can be done with HalloWing. You could add a potentiometer for tuning your samples on the fly, or add some NeoPixels for a truly gig-worthy wearable. What will you add to your Handy Sampler glove? Show us at **hackspace@raspberrypi.org**! □

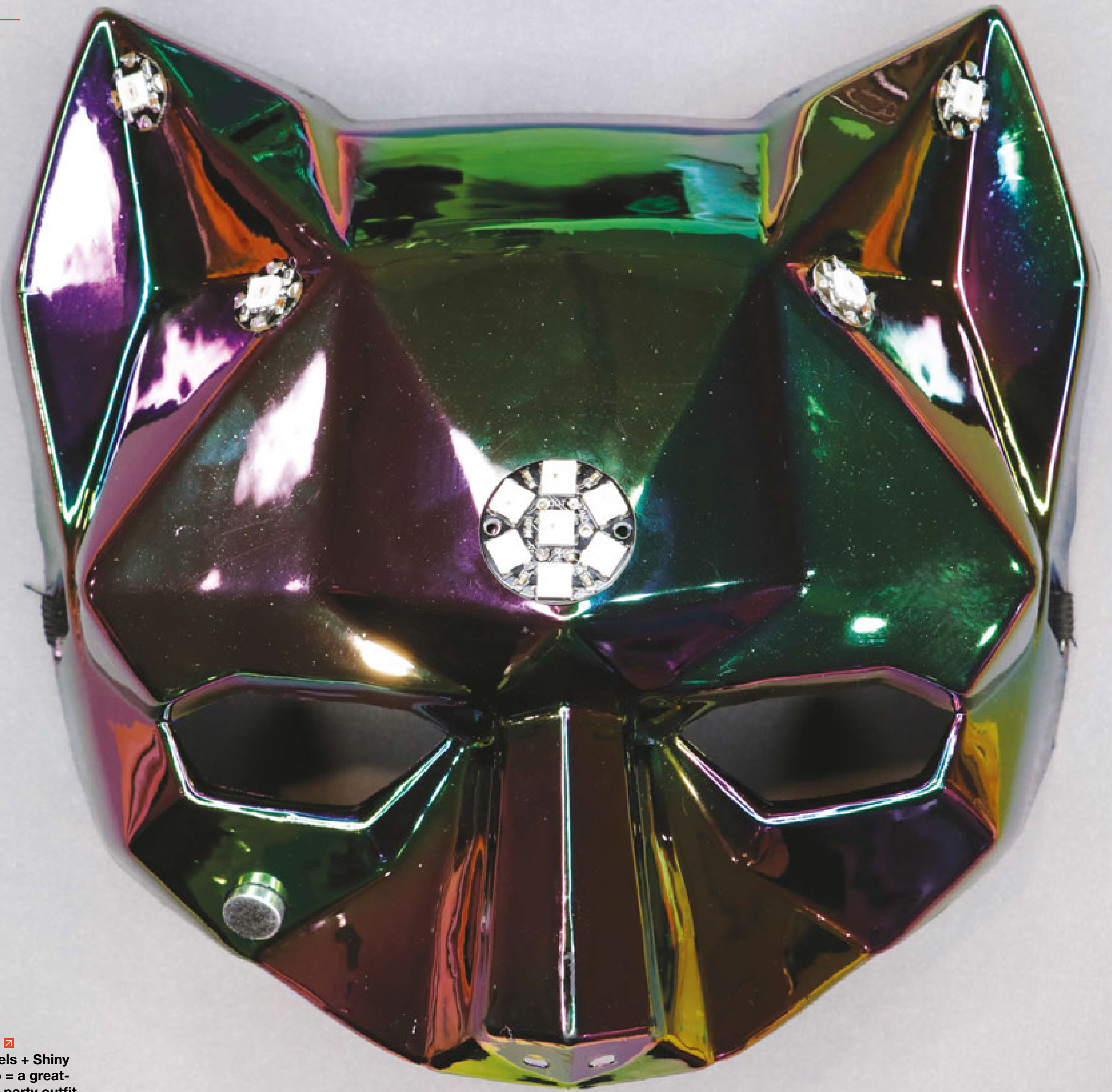

**Above** Your sewn traces will be visible, so it helps if they follow smooth lines

**Below** Don't sew the glove to your hand!

**Below** A bit of hot glue keeps your touch sensors from constantly triggering

# Audio-reactive NeoPixel mask

Light up the party with the sound of your voice



**Above** NeoPixels + Shiny + Audio = a great-looking party outfit

**Bring your project to life with sensors!** Sensors let your microcontroller react to the environment around it, by sensing sound, light, vibration, and more. If you have already used microcontrollers to control outputs like LEDs, you're ready to start using inputs to drive those outputs. In this project, we will use a microphone to sense sound and then turn that data into light, with a Gemma M0 and NeoPixels from Adafruit.

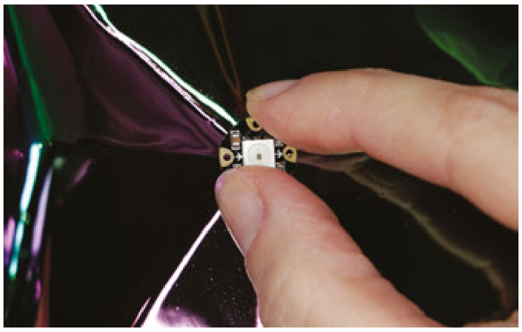
The connections on microcontrollers are commonly called GPIO pins. 'GPIO' stands for 'general-purpose input/output', meaning the pins can be used for either an output like an LED, or an input like a microphone. In our CircuitPython code, we'll set up one of our pins as an output for the NeoPixels, and another as an input for the microphone. A little bit of maths will turn sounds picked up by the microphone into colour shown by the NeoPixels.

Sensors can either be digital or analogue. Digital sensors only detect two states: on or off. Analogue sensors can detect a range of states; for example, volume from 0% to 100%. Notice that the GPIO pins on the Gemma M0 each have two labels: D0, D1, D2, and A0, A1, A2. This is because each pin can be used as either a digital or analogue pin. NeoPixels are digital devices, so we'll use that pin as a digital output (D0), while our microphone will be an analogue input (A1).

NeoPixels are a type of LED that can display any colour. They come in lots of different shapes and sizes, and we'll be using both sewable NeoPixels and a NeoPixel Jewel on this project. A Jewel is just a pre-wired collection of NeoPixels.

The mask used here was purchased by your author for about $5. Look for a mask made of plastic soft enough to drill holes into, but rigid enough to hold the electronic components securely. It's also helpful if the shape of the mask creates some recessed areas that can house the electronic components without pressing them against your face. The cat mask in our project has large, exaggerated features that worked well. Find a mask that inspires you, and let's get started!

**Below**
Sewable NeoPixels are a great form factor for wearables



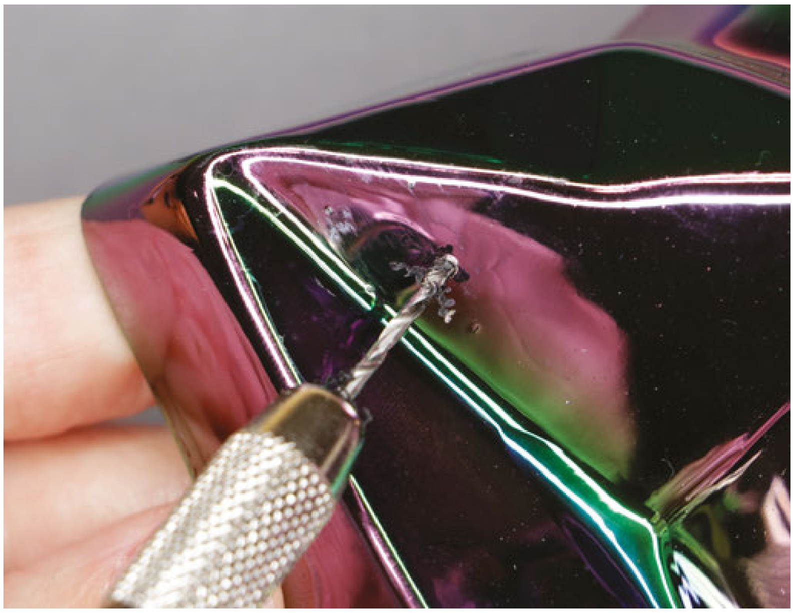
**Above**
Blu Tack is a great temporary adhesive to test out how things can look

## DESIGN YOUR MASK

The first step is to decide where the components will go on your mask. The NeoPixels will be on the outside, and all the other components will be on the inside. NeoPixels can be very bright, so think about the shape of light you will create when the LEDs are lit. Our mask is very angular, so we placed the NeoPixels symmetrically in a V shape, accentuating the angles of our cat ears. If you have a more organically shaped mask, you may want to try a swirl or a more random pattern. It can be helpful to use small bits of adhesive putty to hold things in place while you design. Holes in plastic are permanent, so make sure you like the placement and fit of everything before cutting into your mask.

Think about your circuit, and remember that all the NeoPixels will be connected together in a daisy-chain, with the Gemma M0 connected to the first pixel. The arrows on each NeoPixel indicate the direction that data flows through the chain, and must all point in the same direction: away from the Gemma M0. The NeoPixel Jewel can be inserted at any point in the chain. Try to find flat areas on the front of your mask that the NeoPixels can be adhered to securely, and keep in mind that you'll need to drill holes behind the four pads of each NeoPixel.

On the inside of the mask, look for recessed areas that do not press against your face while the mask is being worn – these are good locations for components. Find a spot near the first pixel on the inside of the mask for the Gemma M0 and the →



Above
A hand drill gives more control than a power drill

**YOU'LL NEED**

- **Plastic mask**
- **NeoPixel Jewel** (product #2226)
- **4 × Flora RGB Smart NeoPixels** (product #1260)
- **Electret microphone breakout** (I used #1713, but #1063 will also work)
- **Gemma M0** (#3501)
- **Coin cell battery holder, with JST connector** (holds two batteries)
- **Silicone-coated 30 AWG wire**
- **Soldering supplies**
- **Hot glue gun and glue sticks**
- **Awl**
- **Craft knife**
- **Small hand drill for crafts**
- **Removable adhesive putty**
- **Electrical tape or Kapton tape**
- **Masking tape**

battery holder. You can mount them separately, as we did, or if space is tight, you can glue the Gemma on top of the battery holder. Just make sure you can still change the batteries and access the on/off switches. You'll also need to find a spot for the microphone. If you want the NeoPixels to light up when you speak, place the microphone somewhere close to your mouth. You'll mount it face-down on the inside of the mask, and make a hole for the microphone to fit through.

## MAKE HOLES

When you've sorted out where you want everything to go, you're ready to make some holes. Use an awl to mark the holes you need to drill – one for each of the four pads on the NeoPixels. On the last NeoPixel in the chain, you do not need to drill a hole for the data-out pad.

Use a small hand drill to make the holes. Check to make sure that two wires will fit through the holes for power and ground.

To make the hole for the microphone, trace a circle around the microphone onto a piece of masking tape, and cut it out to make a template. Use this template to draw a circle for your microphone hole, then carefully cut the hole out with a craft knife. You may want to start the cut by drilling a smaller hole somewhere inside the circle, and then slowly widening the hole with the craft knife. Check the fit often, and remove just enough material so that the microphone slides in easily but is a snug fit.

## CONNECT THE NEOPIXELS

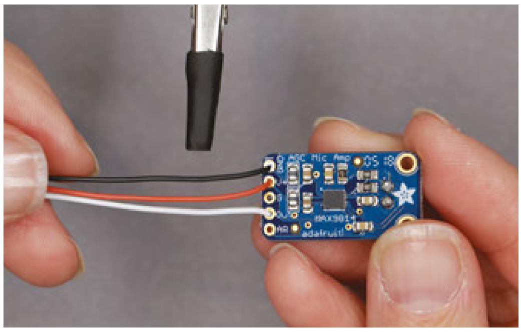
The wiring for this project is fairly straightforward. The NeoPixels all have four connections: data-in, data-out, power, and ground. Power and ground are connected to the positive and negative terminals on the battery. The NeoPixels themselves are chained so the data-in of the first pixel is connected to D0 on the Gemma, and data-out is connected to data-in on the second, data-out on the second is connected to data-in on the third and so on. The Jewel also forms a link on this chain.

Cut lengths of wire for the first NeoPixel, making them at least 5 cm longer than the distance they need to cover. Remember that both the power and ground pads will need two wires each. Feed wires through the holes from the inside to the front of the mask. To easily thread two wires through the small holes, strip some insulation off the ends of the wires and twist them together. Thread the wires through the mask, then through the NeoPixel from back to front. Check the orientation of the NeoPixel, making sure the arrows are pointing in the direction data will flow (away from the Gemma M0). Solder the wires to the pads on the front of the NeoPixel, and trim with angle cutters.

On the inside of your mask, pull the wires gently so the NeoPixel is snug against the front of the mask. (Don't glue anything in place just yet: we'll test the circuit before we make anything permanent.) Spread the wires out along the direction of your circuit. You should have three wires pointing back towards the beginning of the circuit: data-in, one power, and one ground. You should also have three wires pointing forward to the next pixel in your circuit: data-out, and the two other power and ground wires.

Repeat the soldering process for the next NeoPixel in the chain, adding another wire for power, another wire for ground, and a data-out wire. Don't

Below
Heat shrink tube over crocodile clips helps protect components



worry too much about keeping the wires very short: it's better to give yourself some extra length to make it easier while soldering. The Jewel is added in the same way.

Remember that the last NeoPixel in the chain will not have a data-out wire, and only needs one power wire and one ground wire. When your last NeoPixel is soldered, it's a great time to grab your multimeter and check the circuit for continuity. Power and ground should connect from the first NeoPixel to the last, and the data-out of each NeoPixel should connect to the data-in of the next.

> Although CircuitPython code can be written in any text editor, **we recommend using the Mu editor**

Find the free wires at the beginning of the NeoPixel chain for power, ground, and data-in. Solder these to the Gemma M0: power to Vout, ground to GND, and data to D0.

## CONNECT THE MICROPHONE

Cut three wires for the microphone breakout, making sure that they will easily reach from the microphone to the Gemma M0 on the mask. Feed the wires through the holes on the board from back to front, one each to GND (to the power supply negative), Vdd (to the power supply positive), and Out (to the Gemma A1). Solder the wires on the front of the board, and trim any bare protruding wires.

Temporarily hold the microphone in its place on the inside of the mask with adhesive putty, and solder the wires in place.

## PREPARE TO PROGRAM

Now that everything is connected, it's time to program the Gemma M0 with CircuitPython! Connect the Gemma to your computer with a micro USB cable, and turn the Gemma on. The Gemma will appear as a drive called CIRCUITPY.

If you're new to CircuitPython, head over to **hsmag.cc/ZKvDqa** for an overview and resources for learning more, and troubleshooting. Even if you've got a brand new Gemma M0, follow the installation instructions there to make sure your Gemma M0 is updated with the latest version of CircuitPython.

For this code, you'll also need the latest version of the **neopixel.mpy** library. Visit **hsmag.cc/KEvJSv** to grab the latest version of all the Adafruit CircuitPython libraries, then copy the **neopixel.mpy** file into the **lib** folder on the Gemma M0. If there is already a **neopixel.mpy** file in the **lib** folder, replace it with the new one. If you don't already have a **lib** folder on your Gemma M0, create one.

Although CircuitPython code can be written in any text editor, we recommend using the Mu editor, which includes handy helpers like code checking and a serial monitor for debugging. Adafruit has a great tutorial on using Mu here: **hsmag.cc/NtvMQr**. You may already have a text file on your Gemma M0 called **code.py**, but if not, use your editor to create one for your new code. If you already have a **code.py** file, delete its contents so we can write our program from scratch.

As always, our code begins with importing libraries. In addition to the usual `time`, `board`, and `neopixel` libraries, we'll also be using `random` and `analogio`.

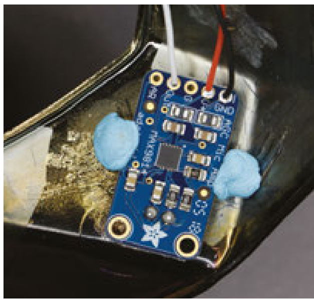
```
import time
import board
```

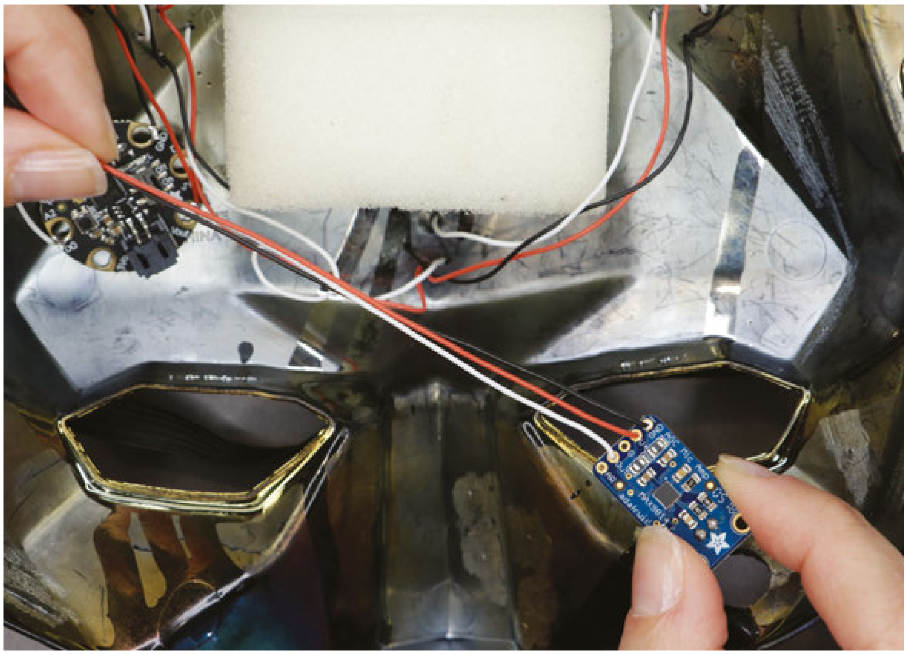
→

**Left** ◈
A microphone ready to be connected

**Below** ◈
The amplifier board sits inside with just the microphone protruding

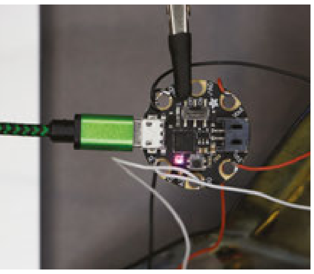
**Bottom** ◈
The inside of your mask doesn't have to be pretty, but it does have to be comfortable





Below
You'll need access to the Gemma's USB port for programming

Bottom
The reflective surface of our mask complements the NeoPixels perfectly



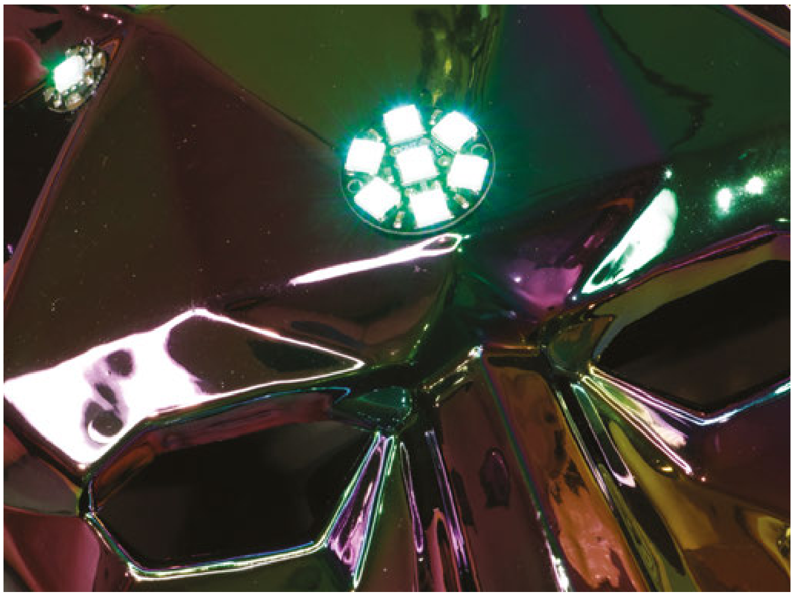
```
import neopixel
from analogio import AnalogIn
import random
```

Next, we'll set up our NeoPixels as a strip on the D0 pin. The variable **`numpix`** is the number of pixels connected together in our mask: the four individual NeoPixels plus seven on the NeoPixel Jewel makes eleven. We'll also set up the microphone as an analogue input on pin A1.

```
numpix = 11
pixpin = board.D0
pixels = neopixel.NeoPixel(pixpin, numpix,
brightness=0.3, auto_write=False)
microphone = AnalogIn(board.A1)
```

We'll create a few more variables while we're at it. **`blinkColor`** is the colour that the NeoPixels blink when the microphone doesn't detect any new sound. Let's call this the idle state. **`colorMultiplier`** is an easy way to shift the colour value of the NeoPixels during the active state, when sound is detected. Set this value to a number from one to six. Anything higher may shift the value beyond 255, and won't generate any colour on the NeoPixels.

Next, we'll tell the microphone to take a baseline reading and call it **`baseLevel`**. Finally, with **`minVol`**, we'll set the minimum volume increase needed to trigger the active state. You can fine-tune the tolerance of your setup by increasing or decreasing **`minVol`**.



```
blinkColor = 200
colorMultiplier = 4
baseLevel = int((microphone.value)/1000)
minVol = baseLevel + 2
```

Next, we have **`wheel`**, a helper function that changes a number from 1–255 into an RGB colour value that the NeoPixels can output. To learn more about **`wheel`**, check out Adafruit's tutorial on CircuitPython for NeoPixels: **hsmag.cc/cDYuuM**.

```
def wheel(pos):
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)
```

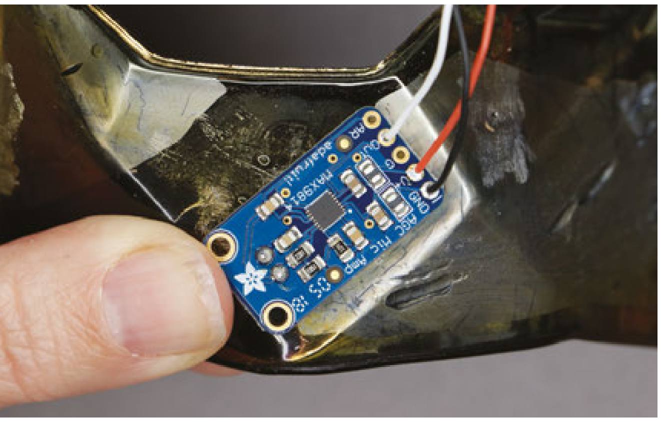
Now we're ready for the main loop! We will start by taking a reading from the microphone, and then turning this analogue value into a colour. We'll call this value **`activeColor`**, for use in the active state of our mask.

```
while True:

    micInput = microphone.value
    volume = int((micInput)/1000)
    activeColor = volume*colorMultiplier
```

Next, we'll use a conditional statement to create the two states of our mask. The **`if`** section is our active state. If the microphone detects new sound, the Gemma will set the NeoPixels to the **`activeColor`** and turn them all on.

```
    if volume >= minVol:
        pixels.fill(wheel(activeColor))
        pixels.show()
        time.sleep(.001)
```

The **`else`** section is our idle state, and it contains another conditional statement. When no new sound is detected, the Gemma will choose a random pixel and a random number between 1 and 100. If the random number is 40 or higher, the chosen pixel will flash the **`blinkColor`**. If the number is less than 40, the chosen pixel will be black (off). Blinking the NeoPixels on and off like this creates a twinkling animation that means


**Above** Coloured wires aren't essential but they help avoid mistakes

the mask still does something interesting even when there is no sound.

```
    else:
        led = random.randint(0, (numpix-1))
        blinkChance = random.randint(1, 100)
        if (blinkChance <= 40):
            color = wheel(blinkColor)
        else:
            color = (0, 0, 0)
        pixels[led] = color
        pixels.show()
        time.sleep(.001)
        pixels[led] = (0, 0, 0)
        pixels.show()
```

Saving the code in the **code.py** file on the Gemma M0 will automatically restart the device and run the program. Your mask should twinkle in its idle state, and speaking at your microphone should make all the NeoPixels illuminate. If your NeoPixels don't seem to be reacting to sounds as expected, try pressing the reset button on your Gemma to take a new baseline reading. If your microphone seems too sensitive, try increasing the tolerance by increasing the amount added to the `minVol` variable at the top of your code.

## GLUE IT ALL DOWN

Finish up by attaching everything securely to the mask. On the front of the mask, gently pull away each NeoPixel, dab a little hot glue behind it, and press it back into place.

On the inside of the mask, remove any adhesive putty and use some small dabs of hot glue to attach the microphone. Try to keep the hot glue away from any soldered components on the board. Glue the Gemma M0 in place as well, making sure that both the micro USB and JST ports are still accessible.

Plug the battery holder into the Gemma M0, and glue it in place on the inside of the mask.

To tidy up the wires, gently gather them into bundles and affix them to the inside of the mask with hot glue. Do not glue the wires of the battery holder, in case you want to switch to a different type of power source in the future, like a LiPo battery.

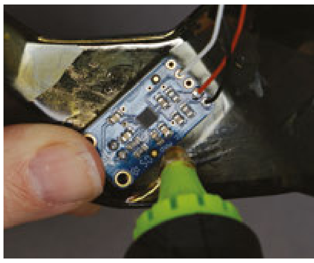
None of the components should touch skin while the mask is being worn, but it's still a good idea to insulate anything that might come into contact with skin. For example, we used some Kapton tape (electrical tape would work too) to cover the back of the microphone breakout board. For comfort, you could also cover the electronics with a bit of felt for a soft barrier.

Insert batteries, and flip on the power switch to turn on your mask! Both the Gemma M0 and the battery holder have power switches, so make sure they're both on. It's fine to leave the switch on the Gemma M0 on all the time, and use the switch on the battery holder to turn your mask on and off.

## SOUND IN, LIGHT OUT

Wear your mask to your next karaoke party and bring your own audio-reactive light show to the stage! Remember that every time the mask is turned on (or reset), the Gemma M0 will sample the ambient noise for a baseline. Press the reset button on the Gemma M0 to re-sample as needed. And don't forget some extra batteries!

Adding an audio-reactive feature is just one way to create wearables that come alive. Microphones and other sensors can give your project an awareness of the environment around it, and create ways for other people to interact with your wearable. Working on a great interactive project? Show us at **hackspace@raspberrypi.org**! □


**Above** A dab of glue holds the microphone permanently in place

**Below** A bit of tape keeps everything safe

**Bottom** Now you just need a social occasion to show off your new mask

# Add lights to a cosplay helmet

Colour-changing LEDs for special effects

**Below** Rumours that this is Elon Musk's prototype helmet are entirely untrue

**So, you've got a cool cosplay helmet and now you want to put lights inside it?** Let's do it! Whether you're starting with a purchased helmet or you've made your helmet from scratch, adding lighting can take your cosplay to the next level. There are many ways to add lighting to cosplay projects; sometimes a simple battery-powered tea light is enough to add a touch of magic to a costume piece or prop. In this tutorial, we'll take a cue from today's space movies (think Ridley Scott's *Prometheus* and *The Martian*) and add internal LEDs to light the face of our astronaut. We'll also add external LEDs that flash when our astronaut is talking: a cool interactive effect that will really bring the costume to life!

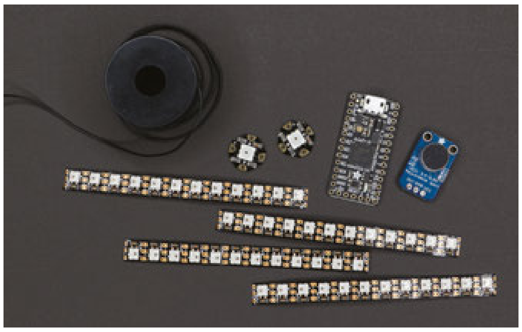
Note: adding interior lights to a helmet creates a reflective glare on the inside of the visor, especially in darker environments. Be careful when wearing this cosplay project: turn the lighting off, or remove the helmet, when moving around. Never install interior lighting in any helmet intended for safety.

We'll work with two kinds of addressable RGB LEDs in this project: a high-density NeoPixel strip, and individual NeoPixels. The strip will be used for our internal lighting; we'll trim it into four pieces to distribute light evenly around the face. The individual NeoPixels will be our external lights, just one on each side of the helmet. We'll turn these into lights that are sound-reactive, and add a microphone on the inside of the helmet to detect when our astronaut is speaking.

In this intermediate project, we'll do some light soldering, create a simple program with CircuitPython, and even craft custom diffusers for our external lights. To control the circuit, we're using the ItsyBitsy M0 Express from Adafruit. While the Gemma M0 board is our go-to for wearable projects (and would work just fine if you've got one to hand), the ItsyBitsy gives us room to grow if we want to add more functionality to the helmet in the future. Maybe a voice changer? Or sci-fi sound samples triggered by capacitive touch areas on the helmet? The possibilities are endless, and once you've got your ItsyBitsy installed, it's easy to reprogram with CircuitPython whenever you get a new idea.

For powering wearable projects, we often use battery packs or small LiPo batteries. But what if you'll be wearing your helmet all day at Comic Con and you don't want to be constantly changing batteries? For this project, our power source is a USB power bank. You may already have one for recharging your phone when you're out and about. Check the power stats, usually printed on the power bank – 5 V/1 A will do. The more mAh, the longer you can go before recharging. Find one with an on/off switch, slip it into the pocket of your spacesuit, and you're good to go all day at the con!

Because we'll be illuminating the inside of the helmet, anything that reflects light will be easily visible. Consider painting the inside of your helmet black, and using all black wires for your circuit. If you choose this method, be sure to label all of your wires as you work, and double-check every connection to avoid errors.


**Above**
Lots of lights, ready for our helmet

For this project, your author started with a purchased costume helmet and modified it with craft foam and paint to create a custom look. The base helmet used here cost about $40 on **amazon.com**. We're focusing on the electronics here, but for techniques and tips on making cosplay helmets, check out these epic cosplayers:

**Punished Props Academy**
punishedprops.com
**Kamui Cosplay**
kamuicosplay.com/startingwithcosplay
**Evil Ted Smith**
eviltedsmith.com

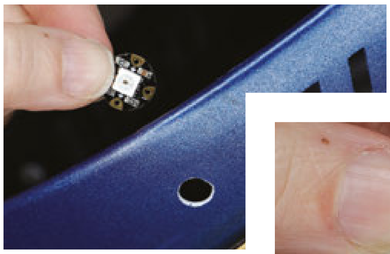
## DESIGN AND MEASURE

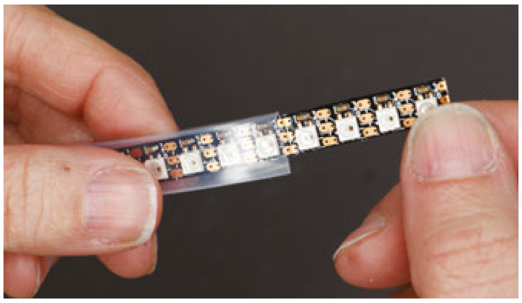
If possible, remove the visor of your helmet and any pieces that will be in the way during installation. Refer to the circuit diagram (not to scale) overleaf for an overview of how everything will be connected. Then, have a look inside the helmet and decide where the internal lighting will be installed. We are using the →

**Right**
Each helmet is different, so you'll have to measure for yourself

## YOU'LL NEED

- **Cosplay helmet**
- **1 metre NeoPixel Mini Skinny Strips, 144 LED per metre** (Adafruit part 2970)
- **2 × Flora RGB Smart NeoPixel version 2** (Adafruit part 1260)
- **1 × Electret microphone amplifier** (Adafruit part 1063)
- **1 × ItsyBitsy M0 microcontroller** (Adafruit part 3727)
- **Silicone-coated wire, in black** (Adafruit part 2003)
- **Black acrylic paint**
- **Hot glue and glue gun**
- **Mould-making putty** (optional)
- **USB power bank and micro USB cable**
- **Soldering tools and supplies**





**Right** ◆ ◆
Individual NeoPixels are easy to mount in holes

**Below** ◆
The LED strip should come out of the weather-proofing

skinniest NeoPixel strips available, so you'll only need about 1 cm of flat space to mount them to. Ideally, you'll have two pieces of NeoPixel strip at the top of the face opening, and two more at the bottom of the face opening. You'll also need to find a good spot for the ItsyBitsy microcontroller, somewhere flat and accessible at the back of the helmet. Remember that our power source will be a USB power bank, so there will be a USB cable running from the ItsyBitsy M0 out of the helmet. Lastly, find a place for the microphone on the inside of the helmet, somewhere near where your mouth will be.

Measure the opening of your helmet and determine how long your strips need to be, and how many NeoPixels you'll be using. (Each piece of our NeoPixel strip has twelve pixels, for a total of 48.) Take measurements for the distances between each piece of strip so you know how long your wires will need to be. Also, measure the distance from each component to where the ItsyBitsy M0 will be located. When cutting your wires, add one or two centimetres for insurance.

Next, find a good spot on either side of the helmet for the individual NeoPixels. These will be daisy-chained together, so try to find symmetrical locations that can be connected along the inside of the helmet. You may need to cut holes into your helmet, as we did, for the LEDs to show through.

**Below** ◆
Soldering wires to strip, but try to make sure you label them!

If the inside of your helmet is white or a light colour, it will reflect the internal lighting, making the electronic components and modifications more visible. For a more sleek look, paint the inside of your helmet black, with a few layers of acrylic paint.

## MAKE THE INTERNAL LIGHTS

Using your measurements, cut the NeoPixel strips for your helmet. Remove the silicone weather stripping, as you may want to save it for another project. It is important to note the direction of the small arrow printed on the back of the strip. This is the direction that data flows through the strip. All strips connected together must be oriented in the same direction so that the data-out of each strip connects to the data-in of the next.

It can be difficult to keep track of the direction of the strips once you've cut them down. Mark the direction of the arrow on the back of each strip. A silver marker works well for this.

Using the measurements from your layout, solder wires to your strips, connecting them together, pin by pin: power to power, ground to ground, and data-out to data-in. Double-check the orientation of each strip before soldering.

Cut wires long enough to reach from the beginning of the strip to the ItsyBitsy, and solder them to the beginning of the strip. Use masking tape to label each wire. There will be a lot of wires going to the ItsyBitsy by the time we're done soldering, and labels can mean the difference between success and hours of troubleshooting.

When finished, set the bundle aside and work on the single NeoPixels.

## PREPARE THE EXTERNAL LIGHTS AND MICROPHONE

Again, reference your measurements and prepare wires to connect the two individual NeoPixels together. Remember that the power and ground pins of the first pixel will need two wires each: one wire to reach forward to the other pixel, and one wire to reach backward to the ItsyBitsy. Make sure the wires are long enough to be routed discreetly between each pixel on the inside of your helmet. Also, remember to connect data-out on the first pixel to data-in on the second pixel.

Add a wire for data-in on the first NeoPixel and label the wires that will connect to the ItsyBitsy with masking tape.

Cut three wires for the microphone according to your measurements and solder them to the microphone. Again, label each wire.

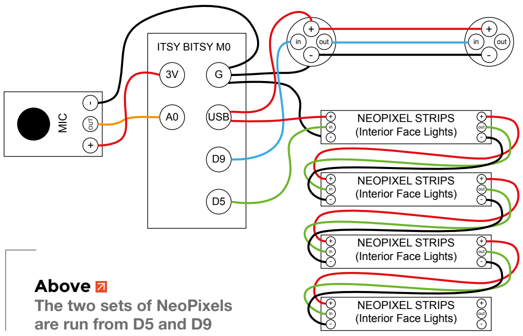
## CONNECT TO THE ITSYBITSY

With all the components prepared with wires, it's time to connect everything to the ItsyBitsy, according to the circuit diagram. This is where all those labels pay off! Leave all the labels on while you work. If you need to troubleshoot any connections later, you'll be happy that everything is still labelled.

Because several components will share the power and ground pins, let's start with those. Connect the positive wire from both the NeoPixel strip and the external lights to the USB pin on the ItsyBitsy. You can twist the wires together gently and solder them to the pin at the same time. Do the same to connect the ground wires from the NeoPixel strip, the external lights, and the microphone to the G pin on the ItsyBitsy.

To power the microphone, connect its VCC wire to the 3 V pin on the ItsyBitsy. This will add less noise to the signal than the higher USB pin, and give us a cleaner, more responsive effect. Connect the data pin on the NeoPixel strip to the D5 pin on the ItsyBitsy M0. D5 is an output-only pin specifically designed to work with the 5 V logic of NeoPixels, so it's perfect for our longer strip. Next, connect the data pin on the external lights to the D9 pin on the ItsyBitsy. And finally, connect the OUT pin on the microphone to A0.

Take a moment to gaze upon the lovely mess of wires coming out of the ItsyBitsy M0. Exciting, isn't it? Now, let's program it and make things light up!



**Above** The two sets of NeoPixels are run from D5 and D9

## PROGRAM THE ITSYBITSY M0

Plug the ItsyBitsy into your computer with a micro USB cable. The ItsyBitsy will appear on your computer as a drive called CIRCUITPY.

As always, start by making sure your microcontroller is up to date before beginning to code. Visit the online guide for ItsyBitsy – at **hsmag.cc/CAbyDw** – and follow the instructions for updating CircuitPython and the libraries on the board. For this program, you'll only need the **neopixel.mpy** library.

If this is your first time working with CircuitPython, visit **hsmag.cc/JgIQCn,** for an introduction and troubleshooting help. Technically, you can use any text editor to write and edit CircuitPython code, but it's highly recommended to use a dedicated code editor. We love Mu Editor, which has a CircuitPython mode and contains helpers and a serial monitor for debugging.

## THE CODE

Our code starts with importing the necessary libraries. Then, we set up the external lights as a NeoPixel strip, with two pixels on the D9 pin. Since these lights will react to the microphone input, we'll call this variable `voicePixStrip`.

```
import board
import neopixel

from analogio import AnalogIn

voicePixNum = 2
voicePixPin = board.D9
voicePixStrip = neopixel.NeoPixel(voicePixPin,
voicePixNum, brightness=0.3, auto_write=False)

voiceColor = (0, 250, 90)  #teal
```

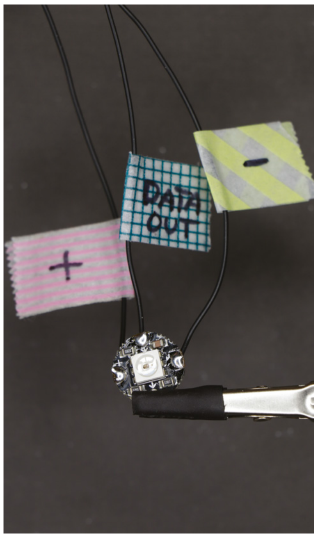
Next, we set up the internal NeoPixels as a strip called `facePixStrip`. Set `facePixNum` to the total number of pixels in your strip. For our helmet, this is 48.

```
facePixNum = 48
facePixPin = board.D5
facePixStrip = neopixel.NeoPixel(facePixPin,
facePixNum, brightness=0.3, auto_write=False)
faceColor = (200, 150, 50)  #warm white
```

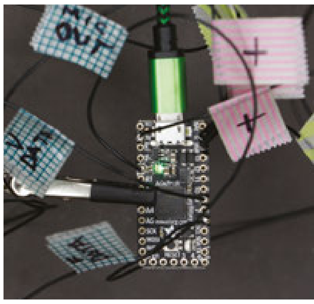
Notice that we have created variables for the colours of each strip: `voiceColor` and `faceColor`. Change the RGB values (red, green, blue) in →

**Left** The NeoPixel strips ready for use

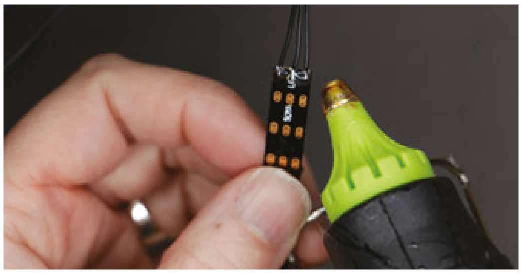
**Below** Keep your wires labelled for easy identification



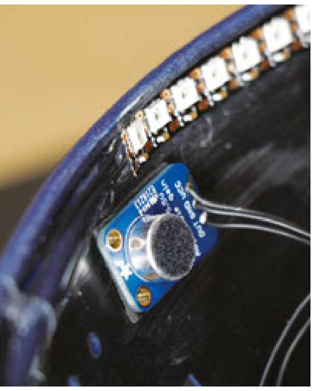
**Below** Everything connected up, ready to go



## CODING



Right
If in doubt, glue-gun it!



Above
Strips and microphone held in place

Below
The NeoPixel goes in the chin area

these variables to tune the colours of each strip to your liking. Each RGB value can be set from 0 to 255.

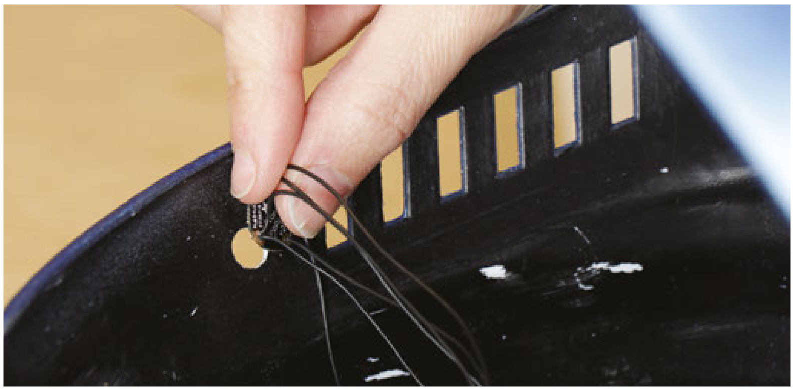
The next few lines set up the microphone on pin A0 and reads the microphone to set a base level. We'll use the **minVol** variable to set the minimum increase in sound needed to trigger the **voicePixStrip**.

```
microphone = AnalogIn(board.A0)
# Take an initial reading of the mic as the base
level
baseLevel = int((microphone.value)/1000)
# Set the minimum increase in volume needed to
trigger active state)

minVol = baseLevel + 4
```

Because we want the internal lights to stay on all the time, we can turn them on before the main loop of our code. This means the ItsyBitsy only has to turn them on once, instead of over and over again while running the main loop.

```
#  Turn on the facePixStrip pixels
facePixStrip.fill(faceColor)

facePixStrip.show()
```

The main loop of our code is simple. We start by reading the microphone and creating a new variable for volume. Next, we compare the current volume with the base level reading. If the incoming volume detected by the microphone increases by more than the value of **minVol**, the **voicePixStrip** will turn on.



Otherwise the pixels will be set to black (0, 0, 0) and will not illuminate.

```
while True:

    # Turn the microphone input value into a
manageable integer to compare with the base level.
    micInput = microphone.value
    volume = int((micInput)/1000)
    print("base: ", baseLevel)
    print("volume:", volume)

    # When the microphone detects sound louder
than the baseline level, fill all voicePixStrip
with color.
    # Otherwise, turn strip off (black).
    if volume >= minVol:
        voicePixStrip.fill(voiceColor)
        voicePixStrip.show()
    else:
        voicePixStrip.fill((0, 0, 0))
        voicePixStrip.show()
```

Saving this code as the **code.py** file on your ItsyBitsy M0 will automatically restart the device and run the code. The NeoPixel strip on D5 should illuminate as soon as the device restarts. See if the two pixels on D9 turn on when you speak into the microphone. If needed, adjust the microphone's sensitivity by increasing or decreasing the **minVol** variable. When your circuit is working as intended, it's finally time to install it!

### INSTALL THE CIRCUIT

It's a good idea to reinforce your solder points, especially on the tiny NeoPixel strips, by covering them with a small dab of hot glue. Be very careful when moving your completed circuit around so as not to snag wires or damage anything. Don't remove the labels from your wiring until the circuit is installed and you've confirmed everything is still working.

With the visor removed, place the entire circuit inside your helmet. Start by attaching the face lights. For the helmet shown, it was easiest to work from the end of the strip backwards, gluing each section in place around the inside of the face opening. Try to position the lights so that they will shine onto the face without pointing directly into the wearer's eyes. Attach the strips with hot glue or double-stick foam tape.

Next, affix the microphone to the inside of the helmet, somewhere near where your mouth will be, with double-stick foam tape or hot glue. Orient the microphone so that the wires can be routed easily toward the ItsyBitsy's location.

Now, the external lights. If needed, drill a small hole on both sides of your helmet for the lights to show through. Glue each NeoPixel in place, on the inside of the helmet, making sure the wires between them can be routed around the inside of the helmet.

Glue the ItsyBitsy in place at the back of the helmet. Make sure the ItsyBitsy's USB port faces down, so that when plugged in, the USB cable will feed out the bottom of the helmet.

Tidy up the wires inside the helmet (and prevent snagging) by tacking them down with hot glue into neat little bundles. To make the hot glue tacks invisible inside the helmet, cover them with a layer of black acrylic paint. Now is also a good time to touch up any black paint that was scraped off inside the helmet during installation.

Connect the ItsyBitsy M0 to your USB power bank and check to make sure that everything still works. If so, hooray! You can finally remove the labels from your wiring! If not, check all your connections and make repairs.

## MAKE THE DIFFUSERS

Now it's time to put the final touches on our helmet. Reinstall the visor and replace any bits and pieces that were previously removed for better access.

LEDs always look better with a little diffusion, but sometimes it's hard to find a translucent 'greeble' that's just right for your project. Here, we made custom diffusers for our helmet using a very familiar material: hot glue! Here's how to do it:

**Below**
Reassembling everything, once you're ready

First, make a mould of something that is a good shape and size for your diffuser. In our project, we used a small plastic cap (it came off your author's tripod). Two-part moulding putty is a great choice for a small object like this. Once mixed, the moulding putty cures in about 30 minutes.

Once cured, fill the mould with hot glue and let cool. This amount of hot glue takes a while to cool down. If you're too excited to wait it out, put it in the refrigerator for a few minutes. When cool, your hot glue part should slide right out of the mould! Repeat the process so that you have two identical parts.

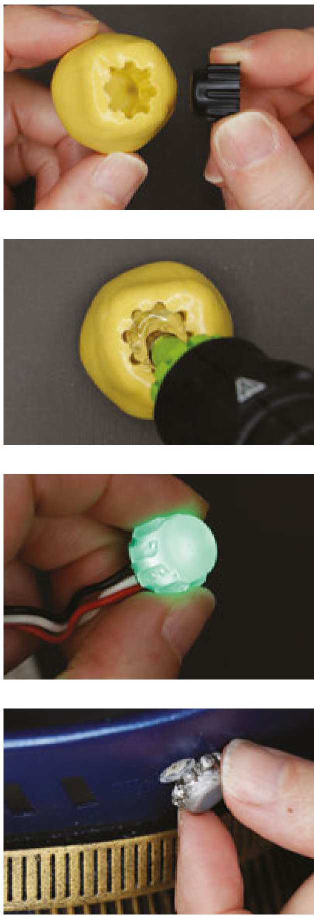
You can leave your diffusers plain, or paint them as desired to fit your helmet design. If painting, leave the top plain for maximum brightness. Have fun playing with this technique, and do experiment with paint to customise your diffusers. Then glue them in place over the NeoPixels on either side of the helmet. Your helmet is done!

## BLAST OFF!

To wear your helmet, plug the ItsyBitsy into the USB power bank and stow the power bank in your pocket (somewhere easy to reach, should you want to turn your helmet off). Turn on the helmet before putting it on. Remember that our code has the microphone take a base-level reading on start, so you may need to press the 'reset' button on the ItsyBitsy M0 to recalibrate the microphone once the USB power bank is on.

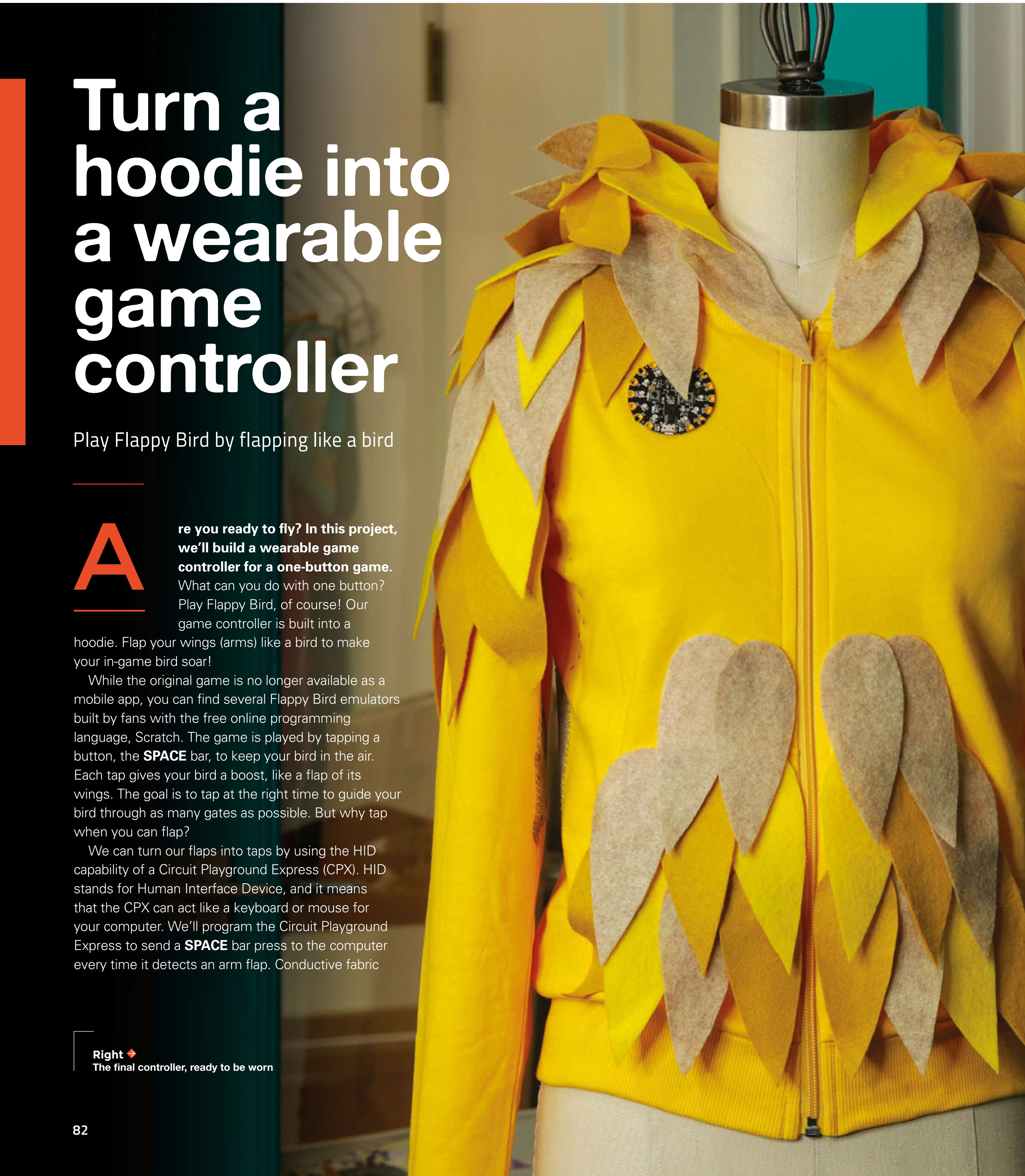
Got a great cosplay project to share? Show us your makes at **hackspace@raspberrypi.org**!

**Above**
One last touch-up



**Above**
Moulding your own LED diffusers with a glue gun

**Left**
The final helmet, ready for use



# Turn a hoodie into a wearable game controller

Play Flappy Bird by flapping like a bird

**Are you ready to fly? In this project, we'll build a wearable game controller for a one-button game.** What can you do with one button? Play Flappy Bird, of course! Our game controller is built into a hoodie. Flap your wings (arms) like a bird to make your in-game bird soar!

While the original game is no longer available as a mobile app, you can find several Flappy Bird emulators built by fans with the free online programming language, Scratch. The game is played by tapping a button, the **SPACE** bar, to keep your bird in the air. Each tap gives your bird a boost, like a flap of its wings. The goal is to tap at the right time to guide your bird through as many gates as possible. But why tap when you can flap?

We can turn our flaps into taps by using the HID capability of a Circuit Playground Express (CPX). HID stands for Human Interface Device, and it means that the CPX can act like a keyboard or mouse for your computer. We'll program the Circuit Playground Express to send a **SPACE** bar press to the computer every time it detects an arm flap. Conductive fabric

**Right**
**The final controller, ready to be worn**

**Left**
It really is as fun as it looks

**Above**
You can test out the fabric with crocodile clips

patches on the arm and torso of the hoodie will detect when the wearer's arm is down. Each real-life arm flap (think chicken dance) will be translated into an in-game bird flap.

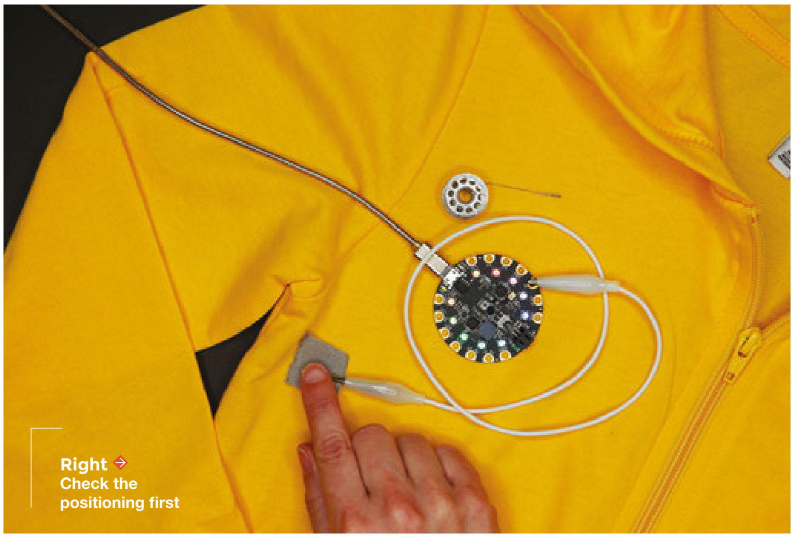
To create conductive surfaces on the hoodie, we are using conductive fabric. Conductive fabric is plated with a conductive metal like silver, and can be either woven or knit. To keep the coating from tarnishing or rubbing off, dry-cleaning is recommended for this material. We'll be hot-gluing feathers to this hoodie when we're done, so do not throw this hoodie into the washing machine – spot-clean only. When adding conductive fabric to a garment, match the stretchability of your garment: woven conductive fabric for non-stretch garments, and knit conductive fabric for stretch garments. Here, we are using knit conductive fabric because our hoodie is also made of knit fabric. Keep in mind that while knit fabrics stretch, straight seams of thread do not (such as the conductive thread traces we will be sewing). To allow for this, leave your stitches a little loose when sewing, or use a stretchable stitch, like a zigzag.

Making your own game controller is a great way to customise your gaming experience and experiment with inputs like buttons and sensors. Another great microcontroller option for this is the Makey Makey, a board designed specifically for turning bananas and other capacitive objects into human input devices. Visit the Makey Makey guide website, at **hsmag.cc/vpwvwT**, to see tons of creative input devices made with unconventional objects. Even a relatively boring game might be fun to play on bananas!

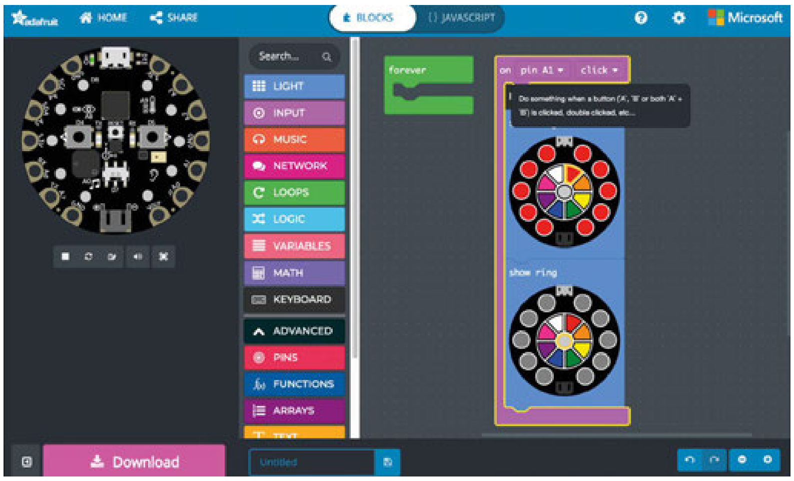
Adding movement to gaming is more than just fun; it's good for you too! In an effort to combine physical activity with coding, and promote STEAM subjects, educators John Lynch and Susan Klimczak, in Boston, Massachusetts, have created a workshop called Explode the Controller. The students use Makey Makey boards to build and hack physical game controllers, like jump-sensing platforms and slap switches. The students can also build their own games that can be played with movement using Scratch programming. For more innovative physical controllers and some inspiration, visit: **hsmag.cc/JfZoAE**. →

## YOU'LL NEED

- **Circuit Playground Express**
- **Micro USB cable (long)**
- **Conductive fabric, knit** (Adafruit part 1168)
- **Conductive thread**
- **Crocodile clip test leads**
- **Hoodie**
- **Felt (coordinating colours)**
- **Fusible web (available from sewing and craft stores)**
- **Press cloth or Teflon pressing sheet**
- **Iron**
- **Sewing needle, thread, pins**
- **Glue gun and hot glue**
- **Fray Check or CA glue**
- **Removable pen or dressmaker's chalk**



Right ➾
Check the positioning first



Above ➾
MakeCode makes programming easy

Below ➾
Measure twice, cut once

In our project, we're using the Circuit Playground Express by Adafruit, which is perfect for wearables with its easy-to-sew-through pin holes.

The Circuit Playground Express has seven capacitive touch inputs, but we're only using one for this project. For a more complex game controller, you can add more conductive fabric patches to your hoodie, or use some of the other sensors on board the CPX. Our hoodie only requires one wing to flap, but for a more accurate experience, add a second sensor to the other arm. In MakeCode, you can modify our program to require both arms to flap at the same time to give yourself more of a workout.

## PROTOTYPE AND PROGRAM

Start by making a quick prototype of the circuit. The Circuit Playground Express has seven capacitive touch inputs on pins A1 through A7. Snip a small piece of conductive fabric and clip it into one end of an alligator clip lead. Clip the other end of the lead onto the A1 pin on the Circuit Playground Express. This is a slightly simplified version of the circuit we are going to build, and we can use it to work on our code.

Connect the Circuit Playground Express to your computer using a micro USB cable, open your browser, and navigate to **hsmag.cc/qJhlwD**.

If this is your first time working with Microsoft's MakeCode, head over to the Adafruit guide at **hsmag.cc/uKjuVC** for a great introduction and more tutorials.

In the MakeCode workspace, we first need to add the Keyboard module to our project. Open the Advanced module, then open the Extensions module. In the screen that appears, click on Keyboard. You will be transported back to the MakeCode workspace, with the Keyboard module added to the list below Math. Next, we'll set up the A1 pin as a capacitive touch input. From the Input module, find the 'on button click' block and drag it onto the workspace. Click on 'button A' and change it to 'pin A1'. Now, whenever the A1 pin senses a 'click', the code we enter inside this block will be triggered.

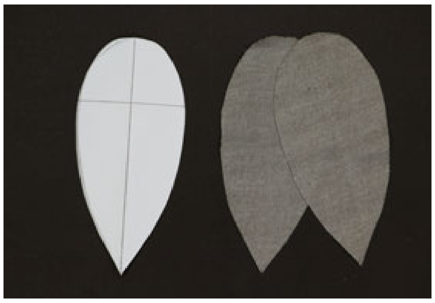
" In the MakeCode workspace, we first need to add the Keyboard module "

Think of 'click' like a mouse click – you press down and release. Selecting 'on click' means the event won't be triggered until the pin (or conductive patch) is released. There are other options in the 'click' drop-down menu (long click, down, and up) but we're going to use 'click' so that each flap is like a tap of the **SPACE** bar.

To send the **SPACE** bar key press to the computer, open the Keyboard menu and find the 'keyboard type' block. Drag this block onto the workspace and drop it into the 'on click' block. Between the quotation marks, type a single space.

Now, without the game running, it may be hard to tell if your program is working. So let's add another action to this button event so that we have some feedback on the Circuit Playground Express when A1 is clicked. This feedback will also help if we need to do any debugging later.

From the Light menu, grab the 'show ring' block and drop it in the 'on click' block below the keyboard command. This will turn on the NeoPixel ring on the Circuit Playground Express and show red. To turn the NeoPixels off, place another 'show ring' block below the first one, and turn off all the NeoPixels in this



second block (click on the grey circle in the centre of the ring, then click on each of the NeoPixels to turn them grey).

Test your code by clicking on the A1 pad in the simulator window on the left. You won't see any representation of the keyboard command, but you should see the ring flash red each time you click. Once you've confirmed your code is working, remove the 'show ring' blocks from the 'on click' block for better timing and more responsive gameplay. Click Download and follow the on-screen instructions to transfer the code to your board.

To see if the **SPACE** bar command is being sent, load up your game and give it a go. Touching the piece of conductive fabric should make your bird fly, just like pressing the **SPACE** bar on your keyboard.

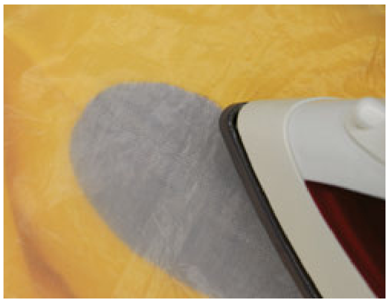
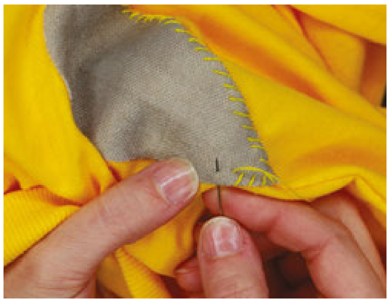
That's all the code you need for a simple one-button flying bird game. But you don't have to stop there! Now that you know how to connect inputs to keyboard commands, you've got all the on-board inputs at your disposal. Add more conductive patches to your hoodie to build more complex game controllers for games that require more than one button. Or try using the accelerometer or microphone to control your game!

## DESIGN YOUR HOODIE

With the game controller circuit prototyped, now we can work on how to integrate it into a wearable form. A hoodie is a good choice for this project, with lots of surface area for capacitive sensor patches.

To achieve the 'flapping bird' motion, we'll place one conductive fabric patch on the inside of one arm, and the other patch on the side of the torso. The patches need to touch when the wearer's arm is down. With a tape measure, find the distance from your armpit to your elbow. Subtract about 5 cm from this measurement, so the patch will sit above your elbow. On the hoodie, mark this distance with a pin on both the inner seam of the sleeve and the side seam of the torso. Or, you can simply put the hoodie on and eyeball your marks.

Decide where to put the Circuit Playground Express, and remember that it will be connected via USB cable to your computer while you are playing the game. Choose a spot on the front of the hoodie

on the same side as the sensor patches. This will make it easy to sew a trace from the board to the torso patch.

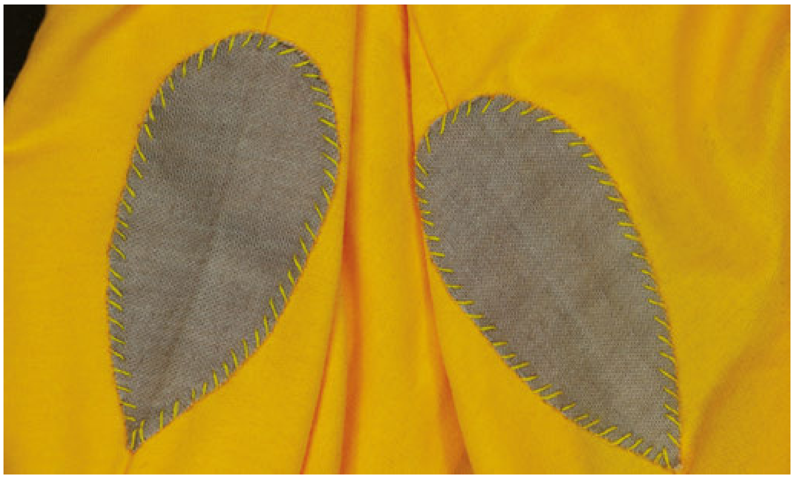
## MAKE CONDUCTIVE PATCHES

In keeping with our bird theme, our patches are shaped like feathers. Use a template to cut two patches of conductive fabric and two matching pieces of fusible web. Follow the instructions on your fusible web to apply it to the underside of the conductive patches (do not apply steam). Use a press cloth or a Teflon sheet to make sure you do not melt the conductive fabric.

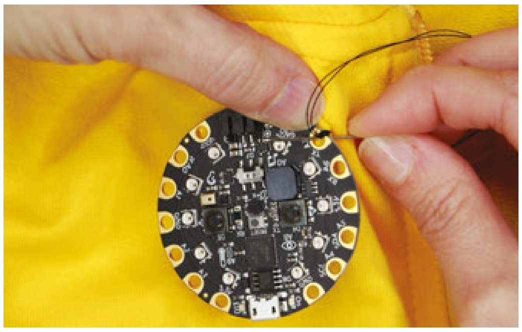
Place one patch in place on the sleeve, centring it on the side seam. Iron without steam, using your press cloth, to fuse it in place. Then repeat to affix the other patch to the side of the torso, making sure these patches will match up when worn. To prevent →

**Above** ◈
You could use any shape, but we like feathers

**Below** ◈
Yellow cotton gives a feathered edge effect

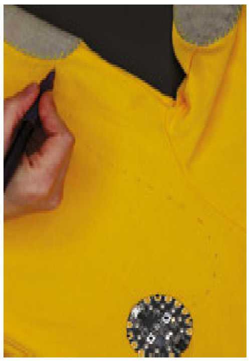


**Right** Conductive thread takes the place of wires in this circuit



**Right** A few stitches hold the CPX in place

**Below** The thread will be visible, so be neat!



the edges from peeling up, thread your needle with regular sewing thread and whip-stitch around the edges of each patch.

To make the arm patch work as a conductive signal, it needs to be connected to the inside of the sleeve so that it will be touching skin when the hoodie is worn. A line of conductive thread works perfectly for this. Thread your needle with conductive thread and sew a running stitch (under, over, under, over) around the inside perimeter of the arm patch. Tie a knot on the inside of the sleeve and dab a little Fray Check, or cyanoacrylate glue, on the knot to keep it tight. Cut the thread tail short.

## ATTACH THE CIRCUIT PLAYGROUND EXPRESS

Hold the Circuit Playground Express in place on the front of the hoodie, and make a mark through four of the pin holes: A0, A3, A4, and A7. Thread your needle with regular sewing thread. Using your marks as guides, sew a few times through each of these holes to attach the board securely to your hoodie. Tie a knot on the inside and trim the thread.

With the Circuit Playground Express in place, you're ready to sew a conductive trace from the A1 pin to the side torso patch. Use a removable marker or chalk to draw out your stitching line first. Avoid sewing over any openings or pockets.

Thread your needle with a piece of conductive thread long enough to sew the whole stitch line you just drew in one piece. Tie a knot in the end, and dab some Fray Check or CA glue on the knot. Starting at the Circuit Playground Express, stitch several times through the A1 pin, pulling your stitches tight against the board for a good connection. Then start sewing a running stitch along your line toward the patch.

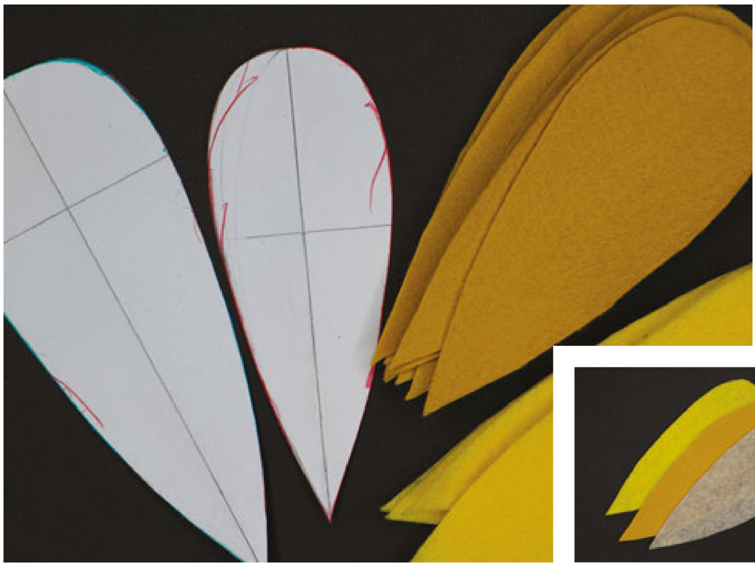
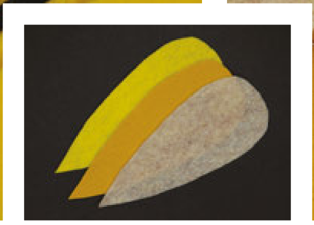
While sewing your running stitch, do not pull your stitches too tight. You just want them to lay flat against the fabric – if you pull too tight, you will gather the fabric and bunch it up. Remember that hoodies are made of stretchable fabric, but conductive thread does not stretch. When you get to the patch, take several stitches through the patch for a good connection and tie it off on the inside of the hoodie. Apply glue to the knot and trim.

" **While sewing your running stitch,** do not pull your stitches too tight "

Put your hoodie on, and check to see if everything is working. Load up your game, plug in your CPX, and flap – does your bird fly? Hooray! If not, check your connections, and make adjustments until your flaps work as intended.

## BIRD ALERT

When you need a break from flapping, you can decorate your hoodie for more bird power. We can't guarantee it will make the game any easier, but it can't hurt!

Use a template to cut feathers out of felt in a few different colours and sizes. You don't need to cover every inch of the hoodie – you can hint at wings by just covering the shoulders. For the shoulders, hood, and tummy of our hoodie, we used about 130 feathers.

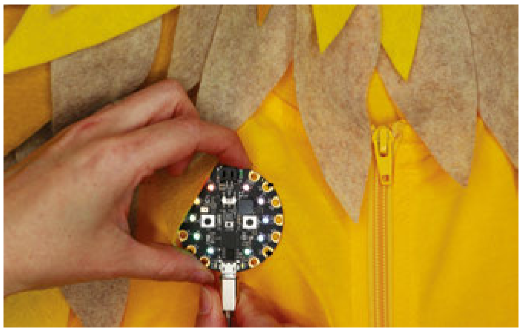
Lay your hoodie out flat on your work surface, and begin laying out your feathers. Pin the feathers in place, with one pin at the top of each feather. Mix up the colours for a mottled effect, or use them in layers for more drama. Don't worry too much about symmetry; just go for visual balance and have fun with your pattern. If you want to add more conductive feather patches, you can work them into your pattern and sew conductive thread traces to them from your Circuit Playground Express as before. Keep all the conductive traces separate from each other. Make sure that your felt feathers do not block the two conductive patches from touching each other when the arm is down.

When you are happy with your pattern, attach the feathers with hot glue. Starting at the bottom layer, lift up each feather and apply hot glue at the top only. Remove pins as you go. Continue gluing feathers down, layer by layer, bottom to top. Be sure not to cover any pocket openings and leave the zipper clear.

## LET'S PLAY!

Keep adding feathers until you're happy with your hoodie. Now you're ready for both Halloween and Game Night! Remember that you are tethered to your computer when playing – be sure to unplug before walking away!

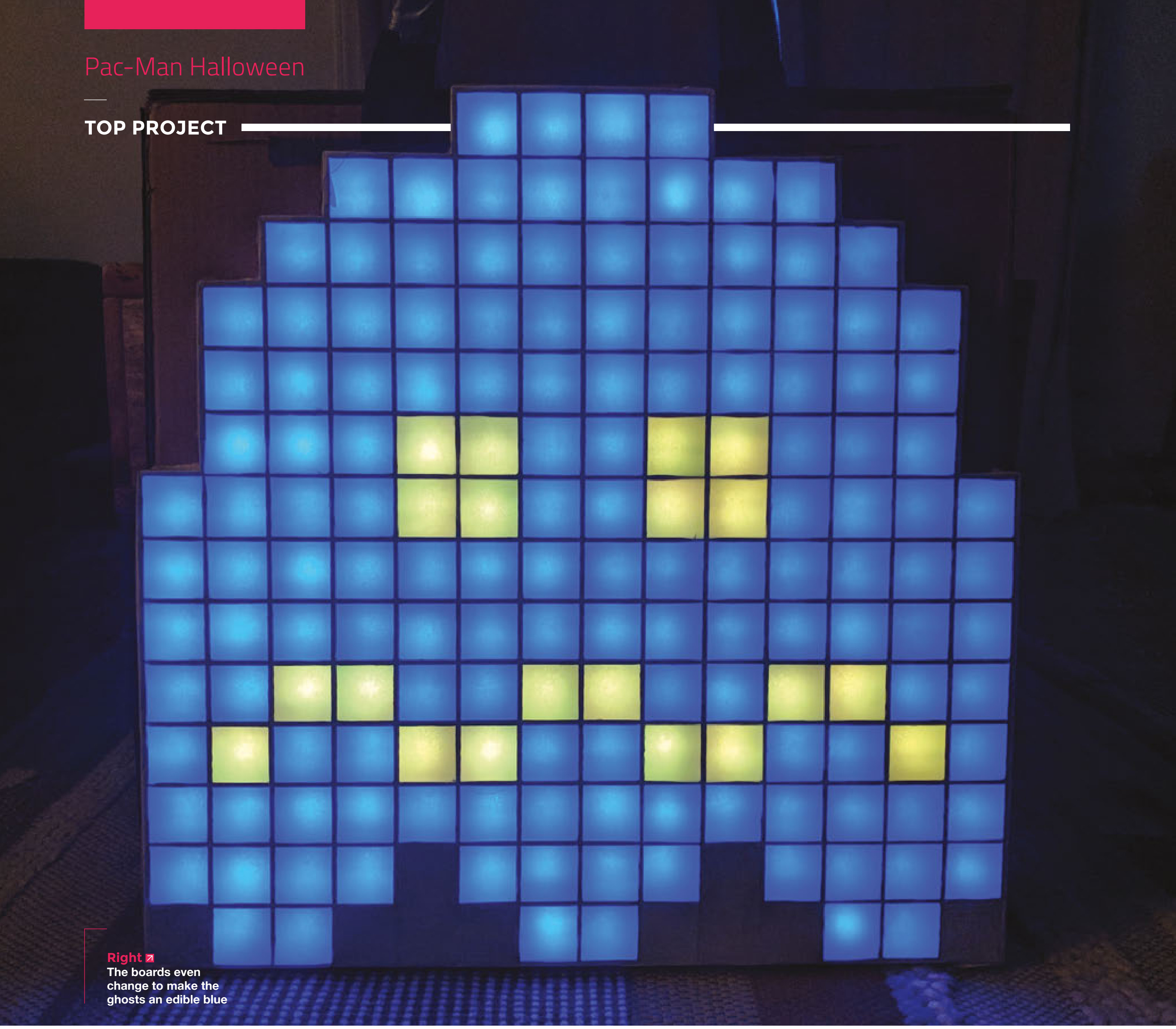
This fun project can be as simple or as complex as you make it, and we want to see how you customise it and make it your own. Show us your creative controllers at: **hackspace@raspberrypi.org**! ◻



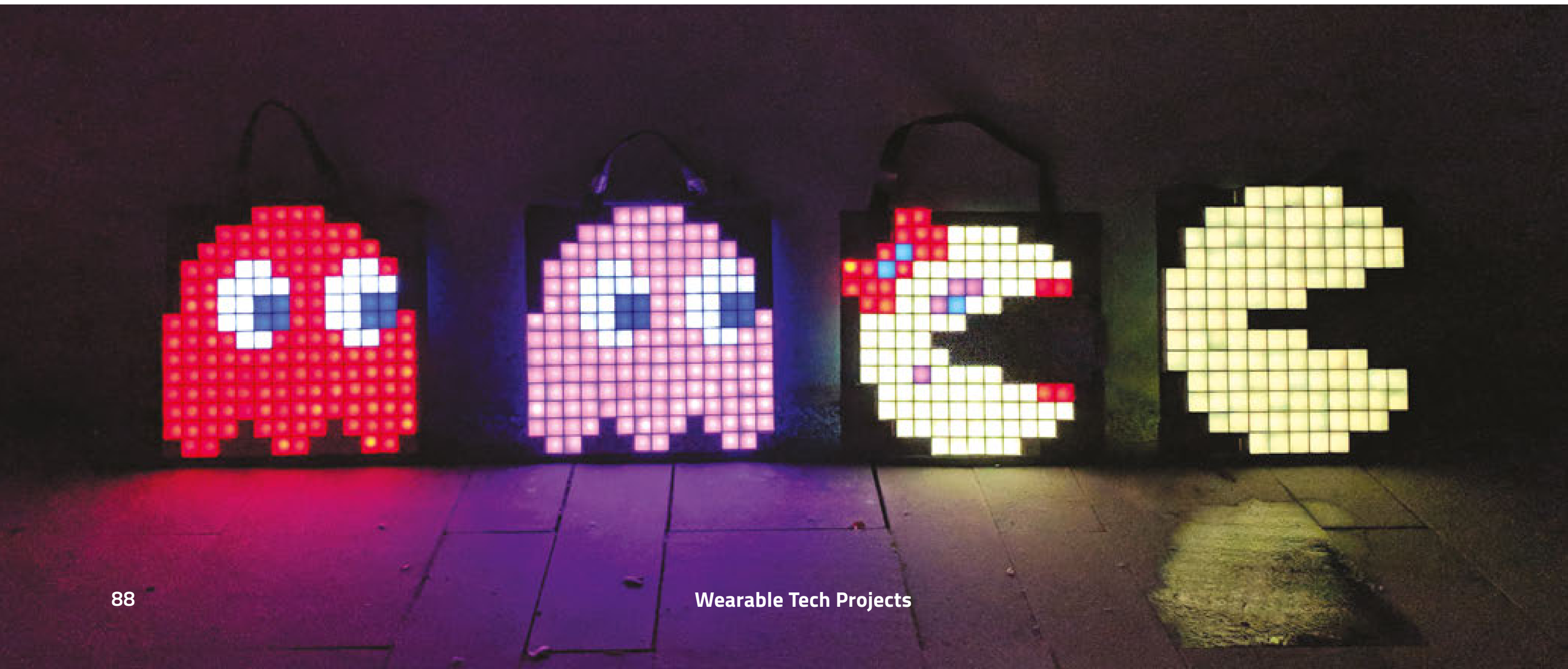
**Above**
Use a stencil to get the feathers the same size

**Left**
Birds don't have to be yellow – you can be creative!

**Left**
Once you're finished, power up and play!



**Right** ↗
The boards even change to make the ghosts an edible blue



# Pac-Man Halloween

By Ben Muller @pix3lot

**I'm an architect living in Philadelphia with my family. I have always been a tinkerer, curious how things work and how to make them on my own.** I mostly work on making things that both fill that curiosity and apply to my work as an architect, which is now focused on VR, AR, and writing custom tools for our 3D software.

Halloween is a good chance to try making something or using something that I haven't tried before. Last year we added LED strips as accents to silver clothes to look like some glowing retro silver space alien people. The costumes looked great and I liked the addition of lights to the costumes.

This year we knew we wanted to do something with LEDs again but we weren't sure what. While trying to come up with an idea, my daughter suggested Pac-Man characters. She had just been playing the Namco classic at a birthday party at a bowling alley. Once she suggested that, we ran with it.

Since we wanted it to be family costumes, it was perfect since there are enough characters and we could all be different. I would be Pac-Man, my wife Ms. Pac-Man, my daughter Pinky, and my son Blinky.

I wanted the costumes to look like they came right out of the arcade game. I wanted them to be animated and for each pixel to be legible.

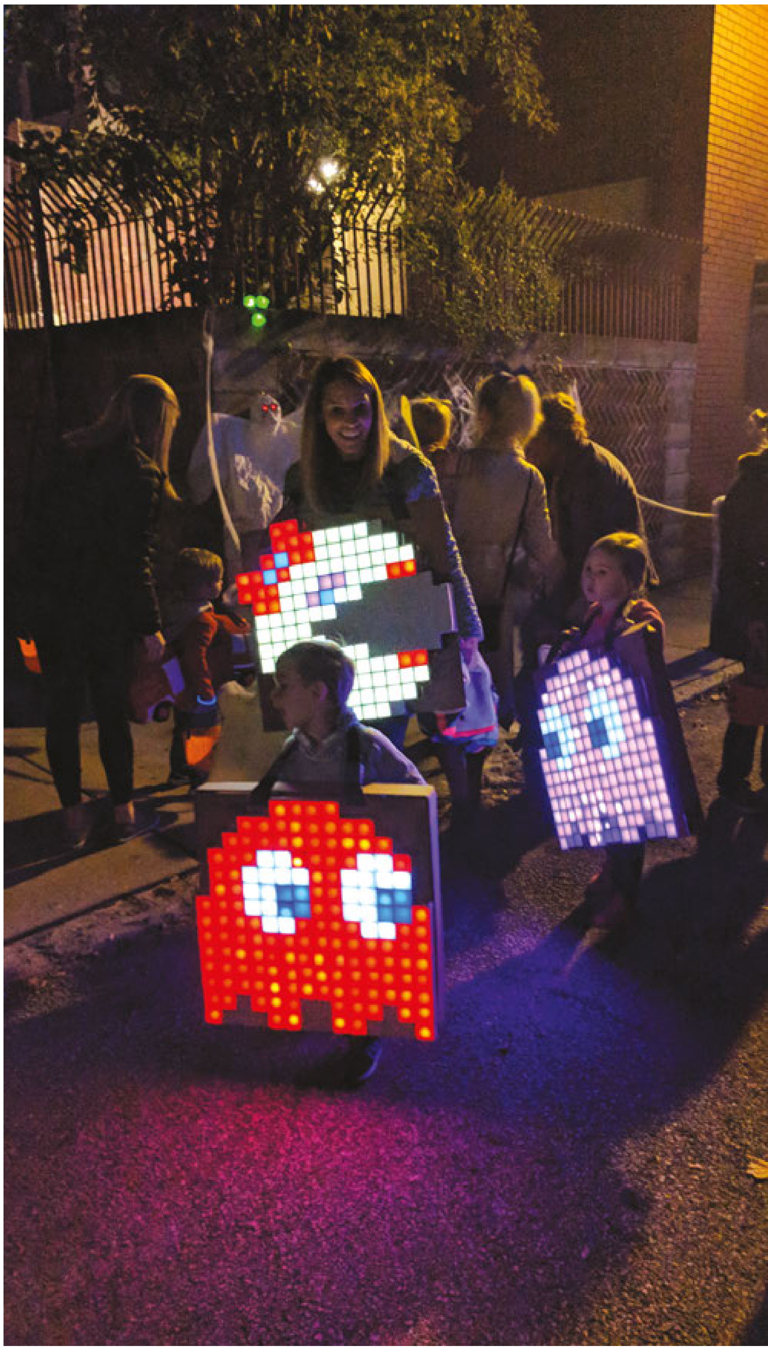
I designed the framework of the panel in the 3D modelling program Rhinoceros by McNeel. We have a laser cutter at my office and I used it to cut out the framework in ⅛" (3 mm) cardboard. The whole panel comprises the base, the slats, and the cover. The base and slats are made of the cardboard and designed so they all slide together, which minimises the need for gluing. The cover is made of drafting vellum, a type of paper.

The lights are strings of individually addressable 12 mm DC5V WS2811 LEDs that are more typically used in outdoor signage. They are inserted into 12 mm holes cut into the base. The slats create the pixels and the vellum diffuses the light to create each pixel.

The LEDs are driven by an Arduino Uno R3. The code is written, compiled, and uploaded to the board with the Arduino IDE. I used the FastLED library (**fastled.io**) to control the LEDs. It's a simple and easy-to-learn library for Arduino that is specifically made for programming individually addressable LEDs.

The LEDs and Arduino are powered by a DC 5 V battery pack (a portable phone charger) and a USB to DC adapter. I did tests on two chargers I had lying around. I got about 12h 40m out of 7800 mAh and 3h 40m out of the 2200 mAh capacity. For most nights out you really only need the small chargers, which is good news.

The board and chargers are attached to the panel with Velcro and a strap is added to make it all portable. □



# PROJECTS

Stretch your making and sewing skills with these more advanced wearable projects

134

TOP PROJECTS

## CRAZY CIRCUITS DRESS

A stylish LED dress for some seriously flashy fashion

# NFC data cuff-links

Covert data carriers for geeks and secret agents

**Have you got a fancy event to attend and need a cool accessory for your outfit that you can store about 144 bytes of data in?** We've got you covered! In this project, we'll make a pair of NFC data cuff-links, ideal for storing a website URL, a password, or a secret message. This project is perfect for a sartorial spy who loves dry Martinis, and anyone who can't remember their WiFi password.

NFC stands for near-field communication, and is a protocol that allows two devices to communicate wirelessly when they are physically near each other. An evolution of RFID, NFC is becoming increasingly popular in consumer technology, and is already commonly used in contactless payment systems and identification badges. NFC wristbands are also being used to create enhanced experiences for visitors at theme parks and other venues.

The rise of NFC hasn't bypassed hobbyists and tinkerers, and companies like Pimoroni and Adafruit sell components that make it relatively easy to add NFC functionality to your projects. Here, we'll make use of tiny NFC tags that can be read and written to by a smartphone or external NFC reader. The tags can be read through a non-metal barrier, like plastic, so we'll embed the tag in resin to make an elegant cabochon for our cuff-link. When complete, holding the cuff-link to your smartphone or NFC reader will let you read or write data to the chip inside.

For this project we used the smallest NFC tags we could find, micro NFC/RFID transponders from Adafruit (product number 2800). These 15.6 mm x 6 mm flexible tags are formatted with the now standard NDEF format, and will work as-is with newer phones and most NFC readers. If you happen to pick up older Mifare Classic formatted tags, they may need to be reformatted as NDEF to work with your reader/writer. Reformatting isn't a function of most NFC read/write

**Left** Emerald green to match your bottle of vermouth

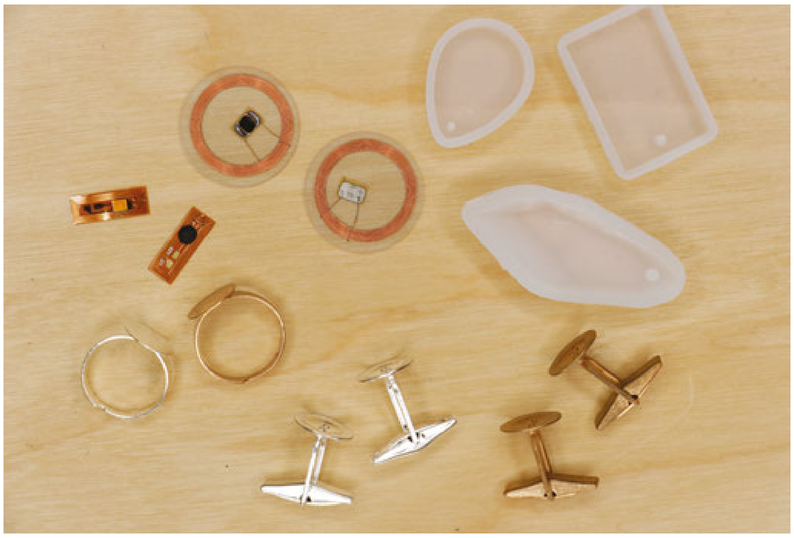
apps, but it can be done with Adafruit's PN532 NFC/RFID controller breakout board or shield.

If this is your first time working with resin epoxy, get ready for a new, fun kind of mess! Resin epoxy comes in two parts that must be mixed together in equal proportions before use. Once mixed, the resin will be workable for a short period of time before entering the curing phase and hardening completely. Figuring out exactly how much resin to mix up is definitely an art. There are even some online tools available to help calculate this. For a small project like this, just make sure you mix up a bit more than you think you'll need. You don't want to run out during the pour and have to quickly mix up more at the last minute. If you're tinting your resin, you definitely want to pour all of your pieces from the same mix, as it's almost impossible to match the colour of one batch of resin to another.

All of this means you'll undoubtedly end up with more than just two cabochons for one pair of cuff-links, and if you're going to make a mess anyway, why not go big? Pick up a few extra NFC tags and plan to pour some other pieces, like pendants or key chain fobs. These make great holiday or birthday gifts that are both technologically advanced and crafty at the same time!

Resin-cast jewellery has been made for decades and there are loads of options for resin moulds available at craft stores and online. The best moulds for resin are made of silicone. Flexible silicone moulds make it easy to remove the hardened pieces, and produce ultra-shiny surfaces. Cuff-link blanks, ring blanks, and pendant bails can also be purchased at jewellery supply stores. Refer to your moulds when choosing cuff-link →

"You'll end up with more than two cabochons… **if you're going to make a mess anyway, why not go big?**"

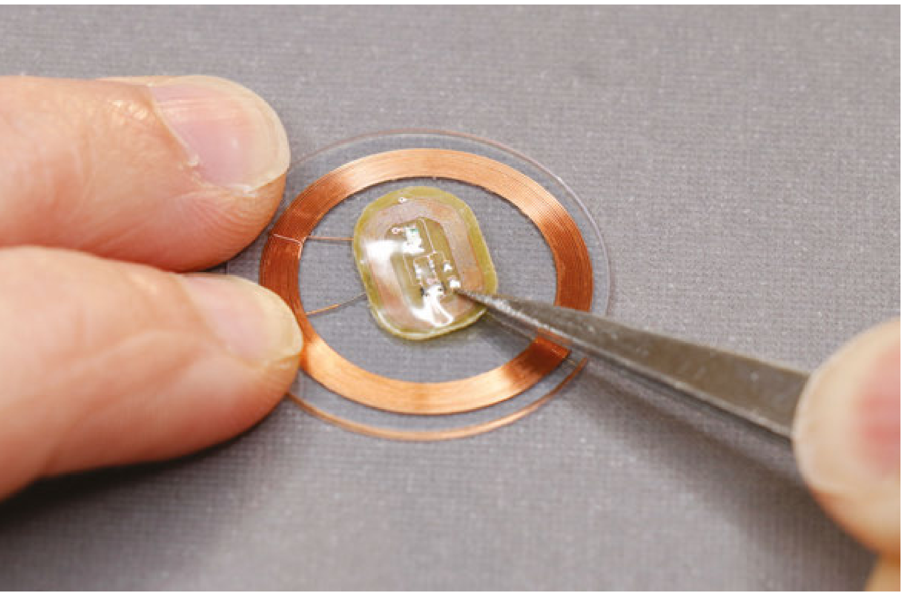


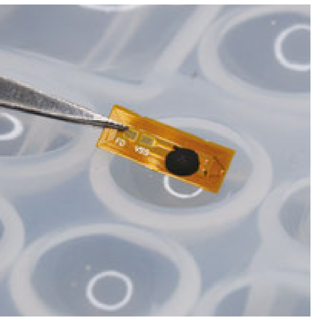
**Above** Just make sure you don't break them on your first mission

**Below** Resin: tough, transparent, and easy to work with

## YOU'LL NEED

- **Micro NFC/RFID transponders**
- **NFC-capable phone or USB NFC reader/writer and computer**
- **Clear casting two-part resin epoxy**
- **Resin tints for adding colour**
- **Small measuring/mixing cups**
- **Stirring sticks**
- **Tweezers**
- **Cabochon moulds** for jewellery making
- **Cuff-link blanks**
- **Isopropyl alcohol in a fine mist spray bottle**
- **E6000 adhesive**
- **Safety gear:** nitrile gloves, particle mask, respirator





**Top** ◈
Tiny tags can be hard to work with, so be careful

**Above** ◈
Make sure your tag fits in your mould

and ring blanks, to make sure that the blanks will work with the size of cabochon you've chosen to cast, and vice versa.

## LICENCE TO SPILL

Start by gathering your materials and setting up your workspace for working with resin. There will be a lot of stirring, pouring, and drips, and things are likely to get messy! Cover your work surface with paper and keep some paper towels nearby. Read and heed the safety warnings on your resin and hardener. Although some resins are considered non-toxic when used as directed, it's always a good idea to work in a well-ventilated area and wear nitrile gloves to keep the resin off of your skin while working.

Once the two-part resin is mixed together, you will have a limited amount of time to pour the resin before it hardens, so planning and timing is key. Check the 'pot life' indicated on your resin; this is the amount of working time you'll have after mixing before the resin begins to harden. Our resin had a pot life of 30 minutes. It can be helpful to set up a timer so you can keep track of time while you work.

If you have multiple moulds, decide which ones you will use before mixing, and make sure your NFC tags will fit into the shapes you plan to use. If you are making matching cuff-links, remember that you'll need two identical shapes. Our tiny 15.6 mm tags fit perfectly into 16 mm cabochons. Remember that you will mix more resin than you need for just two cuff-link cabochons, so it's good to have extra moulds in front of you to pour into.

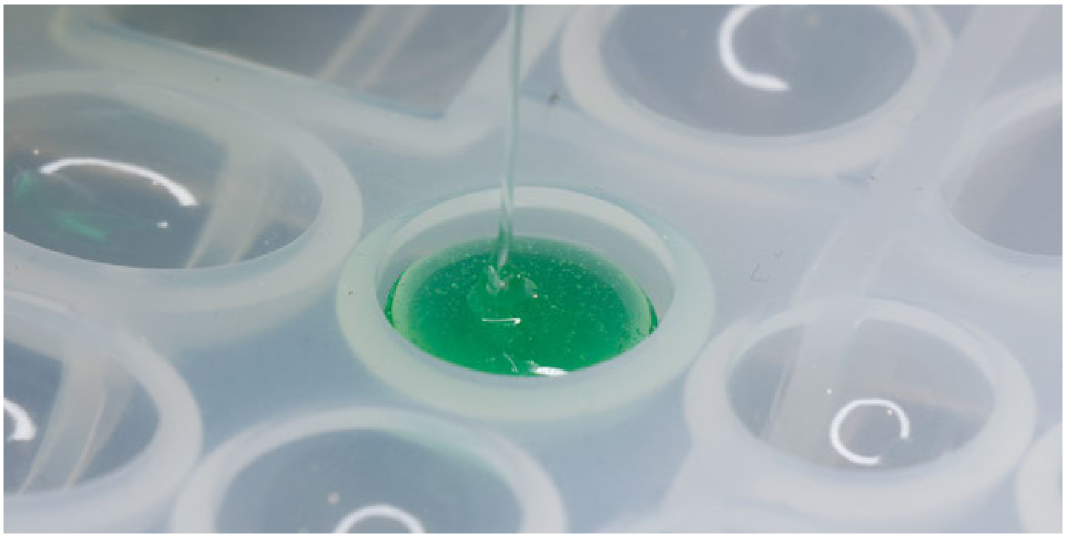
## PREPARE THE NFC TAGS

Unwrap the NFC tags and make sure they are clean and ready to be embedded in the resin. For a light-up effect, you may want to combine a data tag with an LED tag, like we did in one of our extra pieces. The back of the NFC LED nail sticker is adhesive, so it was easy to stick it directly to the larger data tag.

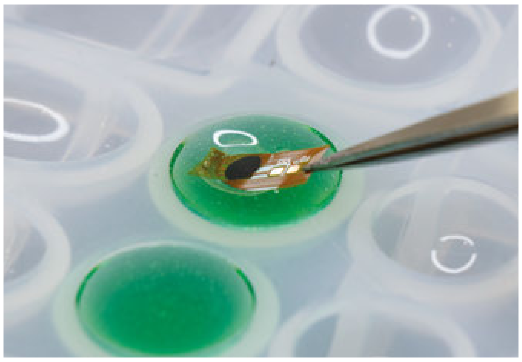
## MEASURE, MIX, AND POUR

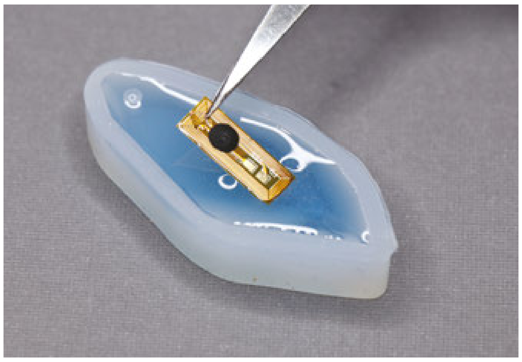
We mixed up about 6 oz (170 g) of resin, then tinted it green for a tech-emerald look. This was plenty for two cabochons and three to four extra shapes.

Follow the manufacturer's instructions to mix up your resin. Generally, it's a 1:1 ratio by volume. A good method for this is to pour each part into matching containers, up to the same measuring mark. Then, pour both into a third cup and stir. Stir slowly, but thoroughly, for at least two or three minutes, making



**Right** ◈
Pour gently, and take your time





**Left**
Long, thin moulds will fit the tags more easily

**Above**
Gently does it! A delicate touch is needed

sure to scrape the sides of your mixing cup often. If the resin is not completely and evenly mixed, it will not cure properly. If tinting your resin, add the tint to your mixed resin one drop at a time, slowly deepening the colour to your preference.

Once your resin is mixed and tinted, you'll notice lots of tiny bubbles that have been incorporated while you were stirring. Let the mixture rest for a few minutes so the bubbles can float to the top, then use a stick to move the bubbles to the side of your container and pop them.

When you've removed as many bubbles as possible, it's time to pour! Place your moulds on a level surface where they'll be able to sit undisturbed for the amount of time required to cure (check the manufacturer's instructions; ours specified 24 hours curing time). Pour the resin in a thin stream into the deepest point of your mould, and let it slowly rise to just below the top lip of your mould. Don't overfill the mould, or the resin will bow and have a convex bottom when you remove it from the mould. Pouring the resin in a thin stream can help pop larger bubbles that are still in the mix.

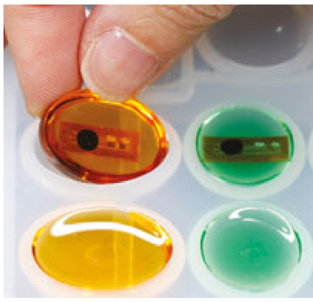
## EMBED THE NFC TAG

With the resin in your mould, you can slide the NFC tag into place. Using tweezers, dip the tag into your unpoured resin to coat it first – this will help the resin in your mould accept the tag without adding too many bubbles. Then, gently slide the tag into the mould and centre it in the resin. It will want to slowly sink to the bottom of the mould, and ideally it stays centred on the way down. You may need to wiggle it back into place with your tweezers or a thin stick, but try not to introduce any new bubbles.

After your resin is poured and the NFC tags are in place, let the resin sit in the moulds for about ten minutes. This is enough time for most of the bubbles to rise to the top surface. Then, spray a fine mist of isopropyl alcohol over the resin to pop the bubbles. This step is optional, but we noticed that it really helped achieve clearer results. →

**Left**
Clear and shiny!

**Below**
You don't have to embed tags – you can pour some clear



**Right** Larger moulds can be used for decorative items

**You could also glue the cabochons to ring blanks** to make NFC data rings

Repeat this process for all the moulds you want to pour and add NFC tags to. Check them after a few minutes to make sure your tag hasn't slid out of place, and remember to keep an eye on your pot life timer. Finish all your fiddling and bubble popping before the resin starts to harden. Then, leave your resin to cure for the amount of time specified in your resin's instructions.

## DEMOULD YOUR RESIN PIECES

When the resin has completely hardened, it's time for the exciting part: removing the cured resin from the moulds. If using silicone moulds, your piece should release from the mould without much fuss. Gently flex the silicone to let air seep between the hardened resin and the wall of the mould. Then you should be able to carefully pull the resin piece out of the mould.

Take a moment to admire your shiny cabochons! If you discover that you've over-poured your moulds, or the resin has crept up the sides of the mould, making a curved back, don't worry. Resin can be wet-sanded; just be sure to keep both the sandpaper and the piece under water while sanding, and wear a mask to keep from inhaling resin particles.

## MAKE THE CUFF-LINKS

Use glue to affix the flat-backed cabochons to the cuff-link blanks. We used E6000, which is an industrial-strength adhesive that works great on plastics. Again, be sure to work in a well-ventilated area, and wear a respirator while working with E6000.

Apply the glue to the cuff-link blank and hold the cabochon in place while the glue sets. Make two, and you're done! You could also glue the cabochons to ring blanks to make NFC data rings. For pendants, you can use jewellery findings like bails and jump rings to make necklaces or key-chain fobs.

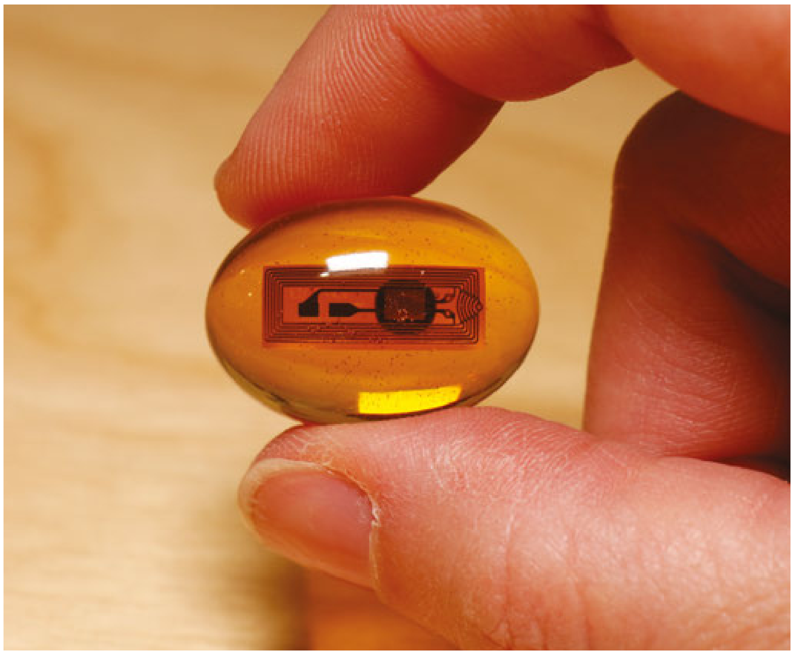
## PROGRAM THE NFC TAG

Now that you've made your NFC cuff-links, you can load them with data like a website, a password, or a secret message. There are a few methods for doing this.

If you have an NFC-capable smartphone, such as an Android phone, you won't need any additional hardware. You can download a free app like NFC Tools to write and read data on your cuff-link. NFC

**Right** Be creative when selecting your moulds

**Above** An even layer of glue helps the stone stick



**Above** Technology meets art

**Below** What your tag does is up to you

Tasks, another free app, lets you create automatic actions for your phone to perform when the NFC tag is read.

If you have an iPhone, (at the time of publishing of this article) you cannot write directly to NFC tags from your phone. But don't worry! You can still join the NFC fun by purchasing a USB NFC reader/writer. You'll be able to read and write to NFC tags with your computer using the NFC Tools desktop app. Your author purchased the NFC reader/writer shown here for about $35 on **Amazon.com**.

You can still use NFC Tools on your iPhone to read tags, and the latest version of iOS, 12.1, supports background NFC tag reading. Some basic actions, like opening a URL in a browser, can now be performed right from the home screen or lock screen – pretty cool!

For a more custom hardware/software approach, try Adafruit's PN532 NFC/RFID controller breakout board, which lets you add NFC functionality to Raspberry Pi or Arduino projects. It takes some soldering and programming to set up, but this breakout gives you lower-level control of the NFC tag, and is supported by an Adafruit NFC Arduino library. The library includes handy example code for reading and writing to tags, and reformatting Mifare Classic tags with the NDEF format.

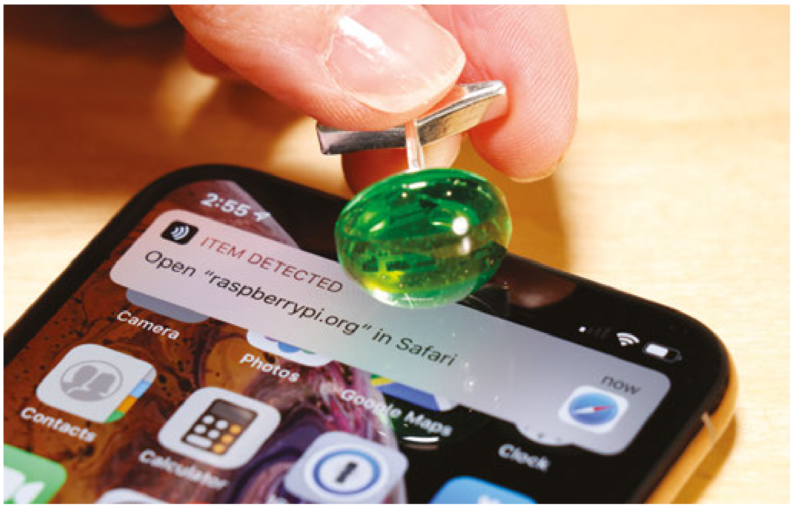
## ON HER MAJESTY'S I.T. DEPARTMENT

Sport your new cuff-links at your next dressy event, and you'll be both covert and classy! Or, gift these to your favourite snappy dresser, loaded with a secret message for their eyes only. Heading to a conference? Instead of handing out a business card to connect with someone, hold your wrist over their smartphone to bring up your webpage. It's not magic, it's technology! □

**GET IN TOUCH**

Working on a cool NFC project? Show us at **hackspace@raspberrypi.org!**

**Below** Most modern phones can detect NFC tags



# Modular NeoPixel patches

These light-up patches look cool on a denim jacket and can be swapped around

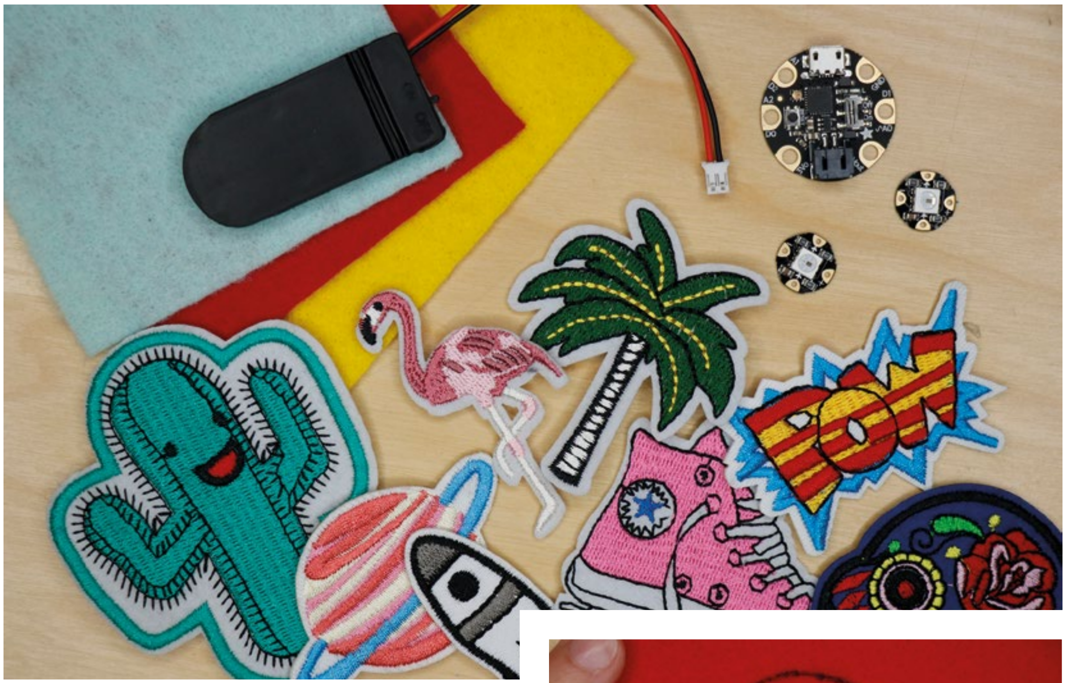
**What could possibly be more amazing than hook-and-loop tape?** Conductive hook-and-loop tape! That's right, your favourite fastener is available in a silver-coated version that is perfect for making a modular circuit with custom swappable components. In this tutorial, we'll use it to create modular NeoPixel patches for a jacket. The microcrontroller and battery for this circuit will be located in the jacket, and be connected to the loop side of the hook-and-loop tape. NeoPixel LEDs will be sewn into the removable patches. Whichever patch you slap onto your jacket will light up!

A Gemma M0 microcontroller from Adafruit will control the NeoPixels, and we'll program the Gemma with CircuitPython. Anytime you want to change the code, just plug the Gemma into your computer and edit the code directly on the device. We'll power the whole circuit with a small battery pack that can hide in a pocket. A sewn circuit is perfect for this project, as it will keep the patches thin and soft.

To achieve good swappability, the hook-and-loop tape will have to be in exactly the same position on the back of each patch, and all the patches should adequately cover the hook and loop tape underneath. The NeoPixels can be in different locations from patch to patch, but it's definitely easiest if they are vertically centred and spread out, as shown on our cactus patch.

Embroidered iron-on patches are perfect for this project. Try to choose patches that are at least 8 or 9 cm long and 6 or 7 cm wide. You'll need some space

**Right** We'll add glowing hands to this cactus using NeoPixels



**Left** A selection of patches and components

**Below** Draw a border around the patch after ironing it to the felt

to spread out the hook-and-loop, and it will be easier to avoid shorts if your conductive traces have lots of room between them. If you want to use smaller patches, iron a few to a single piece of felt to give yourself more surface area to work with. If your patches are small enough, the conductive hook-and-loop may be enough to hold them on securely, but in most cases you'll want to add some regular (non-conductive) hook-and-loop tape for extra security.

Look for patches with graphics that will be easy to incorporate the NeoPixels into. Our circuit will have two NeoPixels, which is perfect for eyes on a face, or glowing Iron Man hands on a cactus (why not?). Get creative with your designs, and keep your circuit in mind while designing. Sketch out your sewing paths before you glue anything down, to make sure none of the conductive traces will cross or touch each other. Pick out your patches and let's get started!

## DESIGN THE CIRCUIT

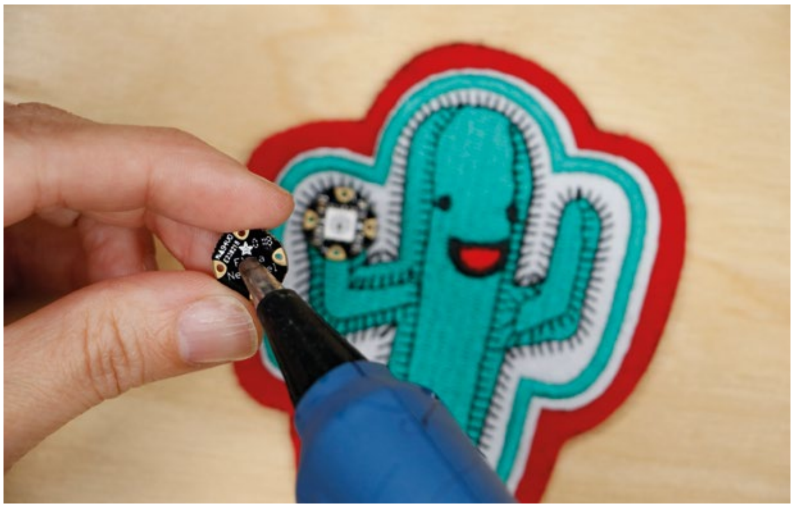
As always, start by designing your patch and sketching out the paths for sewing your conductive traces. Look at all the patches you plan to use and decide on a general layout that will work for all of them. Make sure both LEDs are in the same orientation: the negative pin should be at the top, the positive pin at the bottom, and the data pins' arrows should point to the right. When looking at the front of your patch, the NeoPixel furthest to the left is the first NeoPixel. Design your circuit so that data flows from D0 on the microcontroller to the data-in pin on the first NeoPixel. Then sketch a path from the data-out pin on the first NeoPixel to the data-in pin on the second NeoPixel.

The back of each patch will have three pieces of hook tape (the scratchy side). These should be spaced exactly the same way on each patch. For our patches, the top piece connects to the ground (-) pins, the middle piece connects to power (+), and the bottom piece connects to data-in on the first NeoPixel. Because our NeoPixels are mostly in the centre of our patches, this arrangement made it easy to sew thread traces without crossing over each other. Take the time to work your paths out before you start so you can avoid frustrating shorts and troubleshooting later.

Notice that in our design, the top two pieces of hook-and-loop tape are horizontal, and the bottom piece is vertical. This configuration acts as a visual aid to help keep track of things while working on the back of your patch. With this layout, you'll always know which side is the top of your patch, and what you're connecting to. It also helps to ensure that you'll always put the patch on your jacket the right way up, so your connections will be correct. →

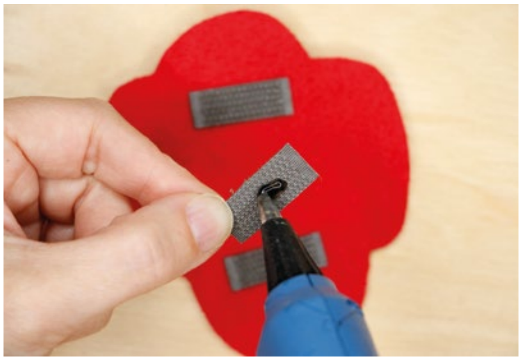
## YOU'LL NEED

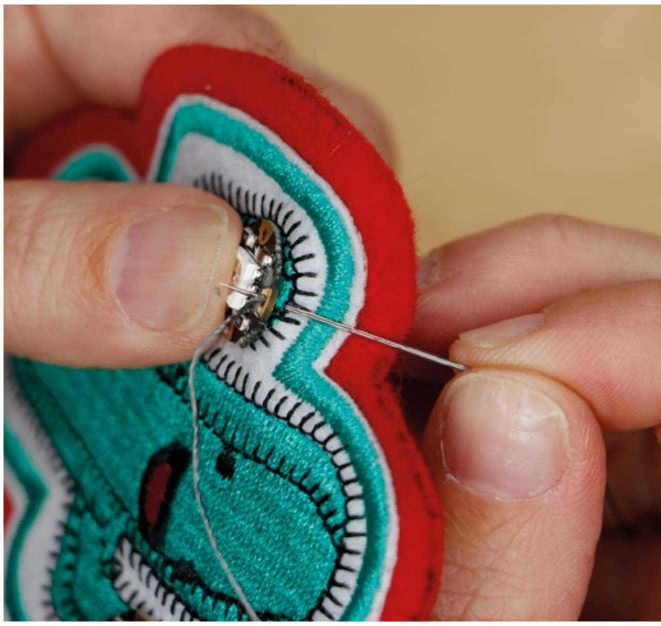
- **Purchased patches**
- **Felt in coordinating colours**
- **Sewable NeoPixels (two per patch)**
- **Conductive hook-and-loop tape (3 cm per patch)**
- **Gemma M0**
- **2 × 2032 coin cell battery holder with JST connector**
- **Conductive thread**
- **Hot glue gun**
- **Fray Check or cyanoacrylate glue**
- **Needle and sewing thread**
- **Regular (non-conductive) hook-and-loop tape**
- **Alligator clip leads for testing**



Above
Hot-glue each NeoPixel into place on the patch

Right
Glue three pieces of conductive hook tape to the back





Above
Connect the NeoPixel with conductive thread

To make the iron-on patches removable, we'll be affixing them to a piece of felt before adding the hook and loop tape. The felt will be visible, so choose colours that coordinate with your patches.

When you've got a clear sketch for your patch design, you're ready to start building the circuit!

"**When you're finished sewing, trim all thread tails short** to keep them from touching other lines of your circuit"

## MAKE THE FIRST PATCH

Place the patch on a piece of felt that is at least a few centimetres wider than the patch on all sides. Following the manufacturer's instructions, iron the patch to the felt. When the patch is securely attached, cut the patch out of the felt, leaving a border of about ½ cm on all sides.

Using a small dab of hot glue, glue each NeoPixel in place on the patch, making sure to orient each one correctly according to your design. Remember, the data arrows should be pointing to the right.

Next, cut three pieces of conductive hook-and-loop tape, making each about 1 cm long. Pull the hook and loop sides apart, and set aside the loop pieces (you'll use them for the jacket). Flip the patch over and glue the three pieces of hook tape to the back. Leave at least 1.5 cm of space between each piece to ensure they will not touch. Remember to make the top two pieces horizontal, and the bottom piece vertical.

Now, grab your sewing needle and conductive thread, and start sewing the traces of your circuit. Refer to your sketch, or draw out your paths right onto the back of your patch. First, sew the ground lines, which will go from the ground pins (-) of each NeoPixel to the top piece of hook tape. Start from the back of the patch, and take a few tight stitches around the ground pin. Then, use a running stitch to sew along the back of the patch to the top piece of hook tape. Try to keep the stitches from showing on the front by just sewing through the felt and not the patch. When you get to the hook tape, take several tight stitches at the edge for a good connection, and tie the thread off. Snip the thread tail short, and dab a little Fray Check or cyanoacrylate glue on the knot to keep it from unravelling. Then, sew the positive lines and data lines of your circuit in the same way, following the paths in your sketch.

When you're finished sewing, trim all thread tails short to keep them from touching other lines of your circuit. Dust off the back of your patch well, and make sure there aren't little conductive thread clippings stuck in your felt. Now let's program the Gemma and make sure the circuit works.

## PROGRAM THE GEMMA AND TEST THE CIRCUIT

Grab three alligator clip leads, and clip one to the Gemma: attach one to the GND pin, one to the Vout pin, and one to D0. Attach the other end of each lead to a piece of the loop tape you've set aside for the jacket portion of the circuit. Then, connect the hook-and-loop tape to the patch to mock up the circuit – ground at the top, Vout in the middle, and D0 at the bottom.

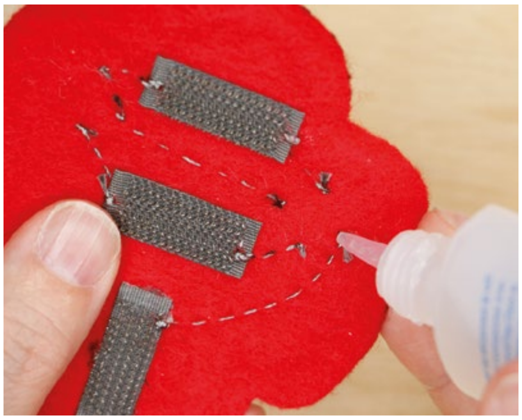
If this is your first time working with CircuitPython, visit **hsmag.cc/ZKvDqa** to get started. Follow the instructions there to update your Gemma M0 with the latest version of CircuitPython.

When connected to your computer with a micro USB cable, the Gemma will appear on your desktop as a drive. Open the drive to find the **main.py** file – this file contains the CircuitPython code that the Gemma is running, and it can be edited in any text editor. Saving this file will make it run automatically on the Gemma M0.

If your Gemma M0 is brand new, there will be sample code in your **main.py** file. Select it all and delete it. We'll write our own code for a simple, colour-cycling animation that will look good on all of our patches.

As per usual, we will start by importing the necessary libraries, setting up variables for our data pin and the number of NeoPixels we're using, and we'll create the NeoPixel strip as an object with two NeoPixels.

```
import time
import board
import neopixel

pixpin = board.D0
```



```
numpix = 2

strip = neopixel.NeoPixel(pixpin, numpix,
brightness=.5, auto_write=False)
```

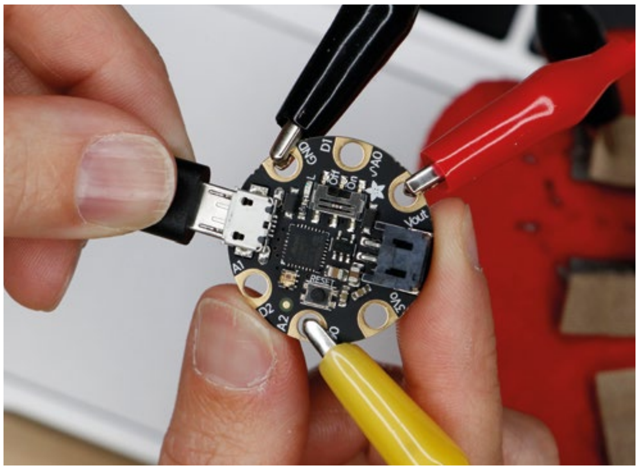
We'll also set up variables with starting values for the red, green, and blue values of our strip. These values can be anything from 0 (off) to 255 (full intensity). For dynamic colour change, we'll want to change each colour value at a different rate, so we'll also need to create a change value for each colour. You can tweak these values later to tune the colour changes to your liking:

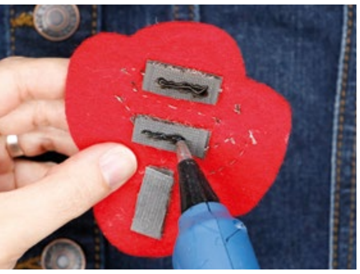
```
r = 0
g = 85
b = 170

rChange = 1
gChange = 5
bChange = 12
```

**Above & Left**
The back of the patch won't be visible, so it doesn't have to be too tidy

**Below Left**
One cactus, almost ready to light up the night

**Below**
Crocodile clips make it easy to test the circuit





**Top** Apply hot glue to the back of each piece of loop tape

**Above** Hold it in place on the jacket, then pull the badge away, for perfect tape alignment

In the main loop, start by filling the strip with the starting RGB values:

```
while True:
    COLOR = (r, g, b)
    strip.fill(COLOR)
    strip.show()
    time.sleep(0.03)
```

To make the colours change over time, let's add the increments we declared in our setup. This will increase each value by these increments every time the loop is repeated.

```
    r = r + rChange
    g = g + gChange
    b = b + bChange
```

Finally, because RGB values only range from 0 (off) to 255 (full), we need to set upper and lower limits for each value. For this, we'll use conditional statements. Starting with the **r** value, when 255 or 0 is reached, we'll flip the direction of the change by making **rChange** negative. Repeat this for both **g** and **b** values. This will make the colour values change smoothly from 0 to 255 and back again.

"**For a good connection and no shorts,** the loop tape on the jacket must line up with the hook tape on the patches"

```
    if r > 255:
        r = 255
        rChange = -rChange
    elif r < 0:
        r = 0
        rChange = -rChange

    if g > 255:
        g = 255
        gChange = -gChange
    elif g < 0:
        g = 0
        gChange = -gChange

    if b > 255:
        b = 255
        bChange = -bChange
    elif b < 0:
        b = 0
        bChange = -bChange
```

**Above** Tight stitches help make a good connection on the pad

When you're finished writing your code, save the file and the Gemma will auto-load the new code. You should see both NeoPixels on your patch light up and slowly change colours. If everything is working as expected, do a little dance and move on! If not, check your connections and make sure your thread traces do not touch each other. Use a multimeter, if necessary, to check for shorts in your circuit. When your circuit is working properly, you're ready to sew the receiving end into your jacket!

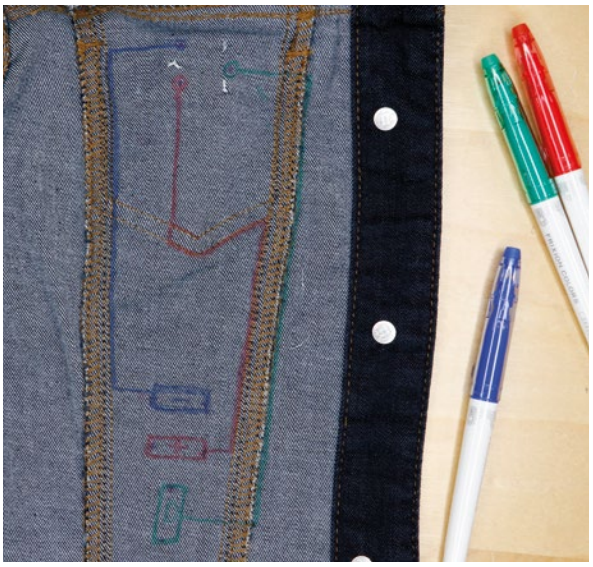
## ADD LOOP TAPE TO THE JACKET

Remove the alligator clip leads from your patch. For a good connection and no shorts, the loop tape on the jacket must line up with the hook tape on the patches. To ensure that it matches perfectly, press the loop tape pieces onto the hook tape pieces on the back of the patch. Apply hot glue to the back of each piece of loop tape, and press the badge into place on the jacket. Hold it there for a moment while the glue solidifies, and then pull the hook-and-loop tape apart to release the badge. Voilà! Your loop tape is guaranteed to be in the right place for your badge. Sew around the edges of the loop tape with regular thread for a secure hold.

You may want to add some additional pieces of regular (non-conductive) hook-and-loop tape to hold the patches securely to the jacket. Place these around the edges of your patch, and repeat the step above to line them up on your jacket.

## INSTALL THE GEMMA M0

Next, decide where your Gemma M0 will live. A pocket is the perfect place to hide it, and if you're



lucky, there will be room for the battery pack in there too. If your pocket has a metal button that might touch the Gemma, like ours did, cover the back of the button with electrical insulating tape or tuck a spare piece of felt into the pocket as a barrier.

If your jacket doesn't have a pocket, choose a spot that will protect the Gemma M0 from the elements, but keep it easily accessible for turning off and on, and reprogramming. Orient the Gemma so that the micro USB port is at the top, and the JST battery connector is at the bottom. Sew the Gemma in place with regular thread through the pin holes we won't be using for this project: D1, D2, and 3Vo.

Now have a look at the inside of your jacket, and plan out the paths you will use for sewing conductive traces from the Gemma to the three loop tape pieces. Remember that GND will connect to the top piece, Vout will connect to the middle piece, and D0 will connect to the bottom piece, and none of the traces can cross or touch each other. To hide the lines of your circuit, try to run the traces along the stitching lines of your jacket. Lightly drawing out your paths with different coloured pens is extremely helpful here.

When you've designed your circuit, start sewing the traces with conductive thread, as before. Start with a few stitches around the pin on the Gemma, then follow your drawn line and finish with a few tight stitches to the loop tape. Make sure your thread is long enough to sew the whole line in one continuous piece. Cut thread tails short and secure all knots with a dot of Fray Check or CA glue. When your traces are sewn, you're ready for batteries!

**Above Left**
To hide the circuit, sew the conductive thread along the jacket's stitch lines

**Above**
Connect the thread to the loop tape

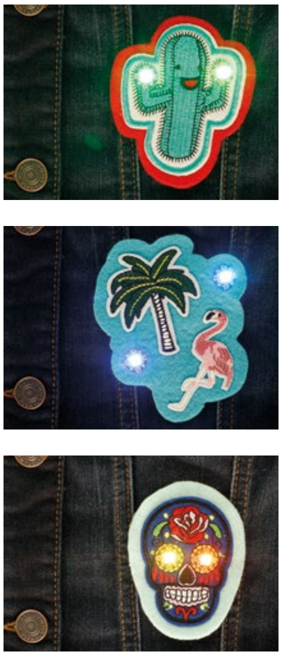
**Above**
Install the Gemma M0 in a pocket if your jacket has one

The Gemma M0 can take up to 6VDC, and can be powered with a 2× 2032 coin cell battery pack, a 3× AAA battery pack, or a small 3.7V lithium-ion battery. If you choose to go with a lithium-ion battery, protect it by wrapping it in gaffer tape and make sure it will not be compressed or punctured while wearing.

With your patch in place, plug your battery into the JST port on the Gemma M0 and switch the Gemma on. Don't be discouraged if your NeoPixels don't light up as they did before: troubleshooting is part of the process, especially with sewn circuits. Check the connections you just sewed into your jacket and eliminate any shorts until your circuit is working correctly. Don't forget to remove the batteries before cutting or sewing traces.

## MAKE MORE PATCHES!

Now that you've perfected one patch, repeat the process to build your collection of NeoPixel patches! Experiment with other shapes and themes, or make a custom patch for a special event. Conductive hook and loop tape can be used to make other modular soft circuits, as well as circuits that open and close around the body, like belts and bracelets. Show us your modular makes at **hackspace@raspberrypi.org**! or on Twitter **@HackSpaceMag**. □



**Above**
Over to you. Now get creative!

# Write with light!

Build your own light painting glove

**Above** Like electrical sparklers, wave this glove around for swirling lines of light in your photos

**Dark evenings are perfect for light painting!** Light painting is the technique of using long-exposure photography and a moving light source to create beautiful lines of light in photos. If you've never heard of this technique before, an online search of #lightpainting should inspire you to try it! In this project, we'll make a light painting glove with interchangeable NeoPixel brush shapes. We'll use CircuitPython to program our glove, and learn how to control the colour of individual pixels on a NeoPixel strip. For artistic control, we'll add a capacitive 'button' to make it easy to turn the light on and off while painting.

We're using NeoPixels in our glove so we can take advantage of the many shapes and forms they come in – lines, circles, and more – to create different 'brush shapes' that attach to the glove with Velcro. NeoPixels are bright, colourful, and easy to work with, but due to their lower PWM rate, you will start to see pixellation in your light painting when moving very quickly. But with practice, you'll find the perfect speed for painting with your glove. And if this project gets you truly hooked on light painting, you can move up to DotStar LED strips, which have a much higher PWM rate and should look buttery smooth at any speed.

This intermediate project requires soldering, some hand-sewing, and uses a LiPo battery for a lightweight power solution. As always, handle LiPo batteries carefully and ensure that they are not compressed, punctured, or otherwise damaged while in your project or being stored.

Because the effect is created in-camera, every light painting shot is a bit of an experiment. You never know quite what you're going to get until you see the captured image. That's the fun of light painting! Because every lighting situation is different, expect to spend some time dialling in your camera settings. After the build, we'll share some light painting tips that will help you get started!

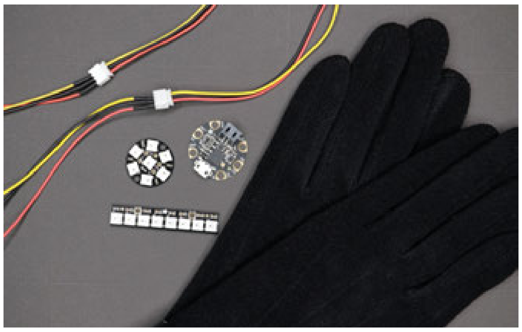
We'll make our NeoPixel shapes easily interchangeable by connecting them to the Gemma M0 with three-pin connectors. On the Gemma, we'll connect D1, GND, and Vout to a female three-pin connector. Then, we'll add male connectors to our NeoPixel shapes. Last, we'll create a capacitive switch by sewing a conductive trail from A2 on the Gemma to one of the fingers of the glove.

## INSTALL THREE-PIN CONNECTORS

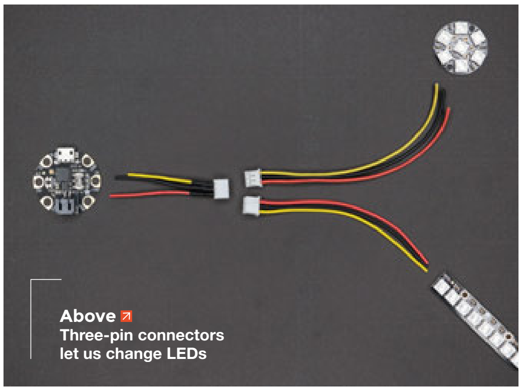
Start by connecting the female side of one three-pin connector to the Gemma M0. Trim the wires of the connector to about 3 cm, then solder the wires to the Gemma. The connector should point directly out from the right of the Gemma.

Plug one of the male three-pin connectors into the female side you just connected to the Gemma, and wrap the circuit around your hand to find the point where the wires cross your palm – this is where the NeoPixels will go. Hold the NeoPixel Jewel in place to see how long the connector's wires need to be, and trim. Solder the wires to the Jewel: D1, GND, and Vout on the Gemma connect to DIN, GND, and 5 V on the Jewel, respectively. Again, try to aim the wires to the side so that the Jewel will lay relatively flat against the glove.

Repeat this process to add a male three-pin connector to the NeoPixel stick. The connections are the same as with the Jewel. You may want to leave the connector wires a bit longer here, so that the stick can be oriented either horizontally or vertically on the glove. Why not give yourself more artistic options? →

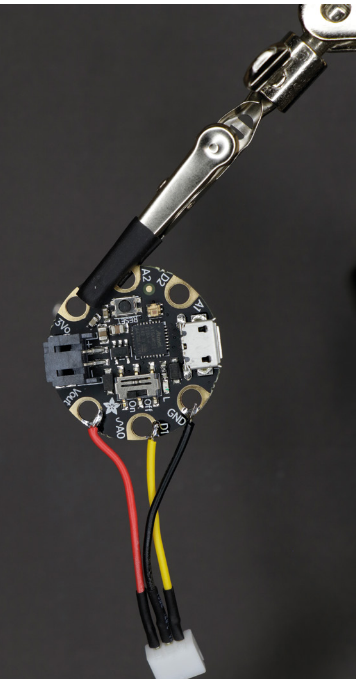

**Above**
A black glove helps accentuate the colours of the LEDs

" You never know quite what you're going to get **until you see the captured image** "


**Above**
Three-pin connectors let us change LEDs

**YOU'LL NEED**

- **Black glove**
- **Small piece of black stretch fabric**
- **Gemma M0 microcontroller** (Adafruit part 3501)
- **NeoPixel stick with 8 RGB pixels** (Adafruit part 1426)
- **NeoPixel Jewel with 7 RGB pixels** (Adafruit part 2226)
- **Small LiPo battery** (100–400 mAh or so)
- **Three-pin JST connectors for each NeoPixel 'brush'**
- **Black sew-on hook and loop,** various widths
- **Conductive thread**
- **E6000 glue**
- **Hot glue gun and glue sticks**
- **Hand-sewing kit and regular thread**



Above
The tiny Gemma is great for projects that don't need lots of input and output

Below
You could use any shape PCBs you can find, provided they use NeoPixels

## PROGRAM THE GEMMA M0

Before installing the circuit on our glove, let's program the Gemma M0 and make sure the connections work. First, take a few minutes to update your Gemma with the latest version of CircuitPython and install the newest **NeoPixel.mpy** library in your **lib** folder. Follow the Gemma M0 guide at Adafruit for complete instructions on how to make sure your board and its libraries are up to date.

CircuitPython is a coding language based on Python, and is specifically designed to make it easy to experiment with microcontrollers. It is maintained by Adafruit, and if this is your first time working with CircuitPython, we highly recommend going through Adafruit's Welcome to CircuitPython guide. It's a great resource for getting started and troubleshooting your project: **hsmag.cc/LXFSWn**.

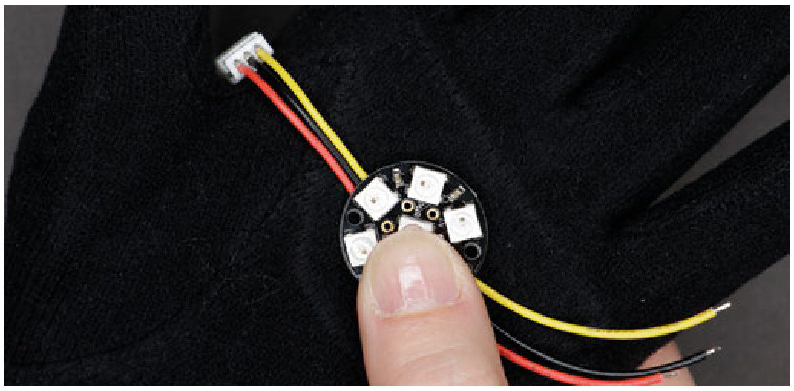
To edit the code on your Gemma M0, connect it to your computer with a data-capable micro USB cable, and locate the text file on the drive named **code.py**. If your board does not already have this file, create it using a text editor (our text editor of choice is Mu Editor). Delete any existing text in the file, and start fresh.

## THE CODE

As usual, our code starts with importing the libraries we'll be using. (Unlike the NeoPixel library, the time, board, and touchio libraries are built into CircuitPython, and so do not need to be placed into the **lib** folder.)

```
import time
import board
import neopixel
import touchio
```

Next, we create a NeoPixel object and link it to the D1 pin on the board. Because we want to be able to use this code with a few different NeoPixel shapes that have different amounts of pixels, we'll set our `numpix` variable to the largest amount of pixels we'll be using. The NeoPixel stick has the most pixels in this project: eight.



Note that when we plug in a NeoPixel shape with fewer than eight pixels, the board won't really know this. For example, the NeoPixel Jewel only has seven pixels, so the Gemma will still be illuminating an imaginary eighth pixel. And if you plug in a single NeoPixel, the Gemma will still 'think' it's illuminating eight. This is fine for our project, which doesn't have any animations or complex maths. But it's something to be aware of if you decide to use this glove with a NeoPixel shape that has more than eight pixels. In that case, you would need to increase the `numpix` value to accommodate more pixels, and assign colours to the additional pixels in the main loop below.

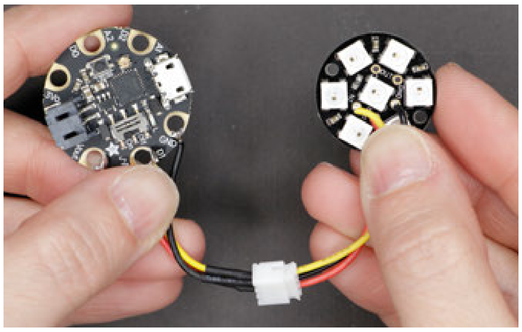
```
numpix = 8
pixpin = board.D1
pixels = neopixel.NeoPixel(pixpin, numpix,
brightness=0.3, auto_write=False)
```

We'll also set up the A2 pin as a capacitive touch input.

```
gloveTouch = touchio.TouchIn(board.A2)
```

Next, we have a useful colour wheel function. Technically, to set a pixel to a specific colour, you need to send it three values for red, green, and blue. This `wheel` function uses some maths to convert a three-digit number from 0–255 into RGB values, making it easier for us to choose colours later on in our code. Any number outside of 0–255, for example 256, is converted to black (0, 0, 0). This `wheel` function is a useful staple for making rainbow-coloured NeoPixel strips, and is documented further in the Adafruit CircuitPython guides.

```
def wheel(pos):
    # Input a value 0 to 255 to get a color
value.
    # The colours are a transition r - g - b -
back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
```



```
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)
```

Our main loop is a simple conditional statement. When the capacitive 'button' is touched, we want the NeoPixels to turn on. Otherwise, we want the NeoPixels to turn off. In the **if** portion of our conditional statement, we'll choose the colours we want the NeoPixels to be. In the **else** portion, we'll turn the strip off by setting all the pixels to black.

We've used our light painting glove in two ways in our photographs: as a single colour (red), and as multiple colours (rainbow). To set all the NeoPixels as the same colour, you can control the whole strip together with **pixels.fill** and choose a three-digit colour from the colour wheel:

```
pixels.fill(wheel(100))
pixels.show()
```

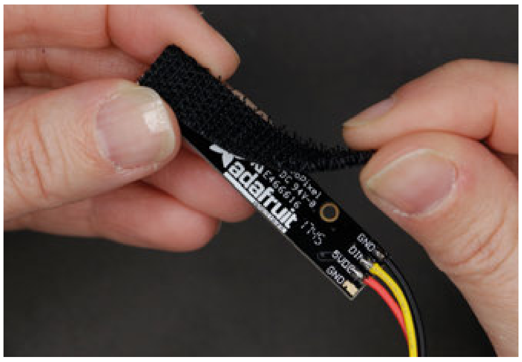
But what if you want to choose different colours for each pixel, as with our rainbow, or even turn some off completely? For that, you'll need to control each pixel individually, which is what we've done here:

```
pixels[0] = (wheel(1))
pixels[1] = (wheel(25))
pixels[2] = (wheel(50))
pixels[3] = (wheel(75))
pixels[4] = (wheel(100))
pixels[5] = (wheel(130))
pixels[6] = (wheel(165))
pixels[7] = (wheel(210))
pixels.show()
```

The number in the straight brackets **[ ]** is the pixel's position on the strip. With our **numpix** variable, we've declared that our strip has eight pixels in it. By default, their positions are 0 through 7 on the strip. This means the first pixel is position 0, the second is position 1, and so on. On each line above, we've chosen a different colour from the **wheel** function for each pixel. To get a rainbow effect, we spread the colour values out from 1 to 210 and tweaked them until we got a rainbow we liked. Our complete code (rainbow version) looks like this:

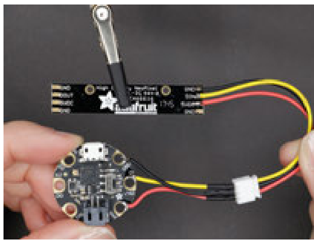
```
import time
import board
import neopixel
import touchio

numpix = 8
pixpin = board.D1
pixels = neopixel.NeoPixel(pixpin, numpix,
brightness=0.3, auto_write=False)

gloveTouch = touchio.TouchIn(board.A2)

def wheel(pos):
    # Input a value 0 to 255 to get a color
value.
    # The colours are a transition r - g - b -
back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)

while True:
    #  When cap switch is touched, show a
rainbow,
    #  otherwise show black.
    if gloveTouch.value:
        pixels[0] = (wheel(1))
        pixels[1] = (wheel(25))
        pixels[2] = (wheel(50))
        pixels[3] = (wheel(75))
        pixels[4] = (wheel(100))
```

→

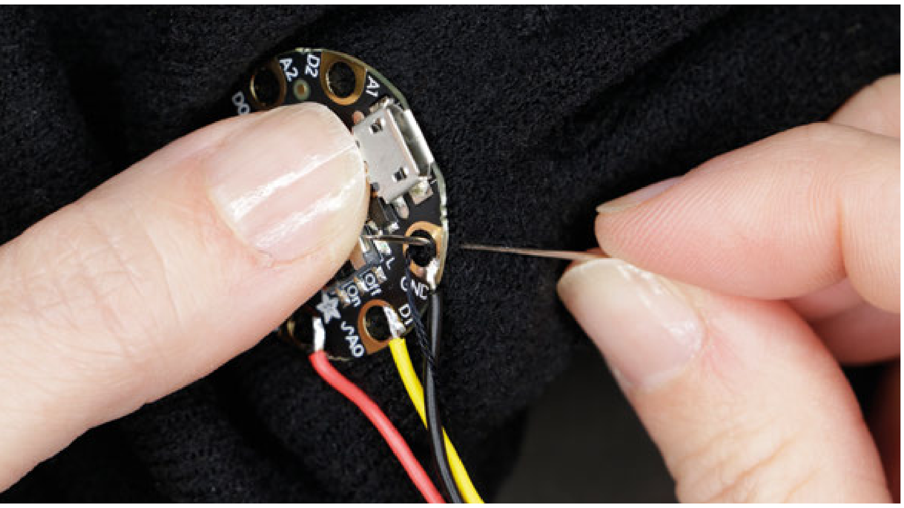


**Left** ◆
The Jewel gives us a small, compact brush to paint with

**Below** ◆
The stick gives us more defined strokes in the final image

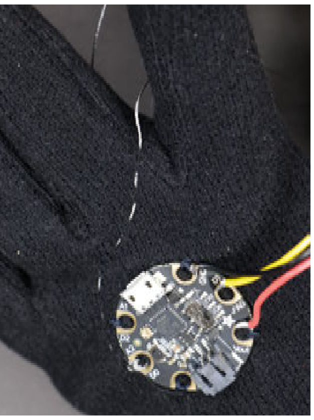


**Left** ◆
Add hook and loop tape to hold the device in place



Above ◈
A few stitches will hold your board in place

Below ◈
You can machine- or hand-stitch this project



Above ◈
Conductive thread takes the touch input to the spot we want it

Right ◈
A little pocket keeps the battery safe and secure

```
        pixels[5] = (wheel(130))
        pixels[6] = (wheel(165))
        pixels[7] = (wheel(210))
        pixels.show()
    else:
        pixels.fill(wheel(256))
        pixels.show()
```

Saving the **code.py** text file should restart your Gemma M0 and automatically run your code. Test it by touching the A2 pad – do your NeoPixels light up? If not, check the connections of your circuit.

Do you like the colours? If not, fiddle with the colour values until you're happy, then move on to assembling the glove. Because the Gemma M0 will be easy to access on top of the glove, it'll be easy to change the code any time you want to try some new colours.

## ADD HOOK AND LOOP

Next we'll create a hook-and-loop mounting area on the palm of the glove to hold the NeoPixels. Use the loop (soft side) for the glove. Extra-wide loop tape (45 mm or so) works great for this, or you can put a few narrower strips next to each other to create a generous mounting area.

Pin the loop tape in place on the glove and sew it down. The easiest method for this is to zigzag around all four edges with a sewing machine. For good measure and a super-secure hold, sew an X shape through the middle too. If you don't have a sewing machine, hand-sew the loop tape in place with regular sewing thread.

Now we'll add the hook tape to our NeoPixels. Cut the hook tape to the size of each NeoPixel object and attach with glue. E6000 glue works well for this application. Be sure to work in a well-ventilated area or outside, and give the pieces ample time to dry and cure before bringing your pieces back inside.

## ATTACH THE GEMMA M0

While your glue is drying, you can turn to the hand-sewing portion of the project. Start by attaching the Gemma M0 to the back of the glove. Hold the Gemma in place and hand-sew through four of its holes – D2, 3Vo, Vout, and GND – for a secure hold. The connector wires will wrap to the front of the glove between the thumb and index finger.

## ADD THE CAPACITIVE TOUCH 'BUTTON'

Thread your needle with conductive thread, and tie a big, tight knot at the end. Coat the knot with CA glue and let dry. Feed the needle from the inside of the glove, and pull it up through the A2 pin of the Gemma M0. Sew several tight stitches around the A2 pin for a good connection, then start sewing a running stitch towards the valley between the index and middle finger of the glove.

Low down on the side of the middle finger, halt your running stitch and take several stitches in one spot to make a raised bead of conductive thread. Tie the thread off with a tight knot and dab the knot with CA glue to hold it securely. Pull the thread after the knot through to the inside of the glove before cutting short.

With a new piece of conductive thread in your needle, sew stitches into the side of the index finger across from the first bead of thread. When the fingers are held together, the thread beads should touch. Make sure this second thread bead goes all the way through to the inside of the glove, so that your finger will touch it and provide the capacitance needed to trigger the 'button'. It can be helpful to wear the glove on your non-dominant hand while sewing, so that you can carefully feel for the needle on the inside while sewing.

## INSULATE THE CIRCUIT

Heat up your glue gun and turn the glove inside out so you can see the conductive thread stitches you sewed from the Gemma M0 to the fingers. Cover

the stitches with hot glue to insulate the thread from your hand while the glove is being worn. Make sure you only insulate the trace running from the Gemma M0 to the fingers – do not insulate the second bead of conductive thread on the index finger.

While you've got the glue gun out, add small dabs of glue to the soldered pins on the NeoPixel stick and Jewel to keep the wires from bending directly at their solder points. Do the same for the soldered pins of the Gemma M0.

## MAKE THE BATTERY POCKET

To house the battery, take a small piece of black stretch fabric and lay the battery on top. Trace a rectangle around the battery with at least 0.5 cm of clearance on all sides, or more to accommodate larger batteries. Our battery pocket is 4.5 × 4 cm.

Cut out the rectangle and pin in place on the glove about 2 cm away from the JST port on the Gemma M0. Sew along three sides to attach the pocket, leaving the side facing the Gemma open. Again, you can either zigzag these edges with a sewing machine, or practise your hand-sewing. If hand-sewing, use a whip-stitch or back-stitch for a secure hold.

## LIGHT IT UP

Now, all that's left is to plug the battery in and pop it into its pocket. Put your glove on, plug in your NeoPixel 'brush' of choice, and start playing with light painting! Turn the Gemma off before changing the NeoPixel shape. Holding your fingers together will activate the NeoPixels; spreading your fingers apart will turn them off!

Start with our settings below, or just play until you get something you like.

## LIGHT PAINTING 101

**Setup:** Anything you don't want to appear in the photograph should be black, so it does not reflect your light source. For example, use a black glove for this project, and black wires for extra invisibility. Dress in black if you'll be standing in front of the camera and don't want to appear as a ghost in your images.

**Equipment:** You don't need fancy equipment for light painting, but you will need a camera with a variable shutter speed. Your camera must be stationary while its shutter is open, so place the camera on a tripod or table. If working solo, it can be helpful to have a remote control or foot pedal to trigger your camera. An external flash (or other source of bright, diffuse light – like a smartphone flashlight) can be used to selectively expose other elements you want to add to your image.

**Settings:** For grain-free images, stick with 100 ISO. Set focus to manual and focus the camera on a placeholder object right about where you will be painting. Start with a 10–15 second shutter or use a 'bulb' setting to leave the shutter open while you paint. For a better chance at clear lines, start with your aperture stopped down to around f/8–f/11. Experiment with shutter speed and aperture to achieve the effect you want.

**Lighting:** You'll want to be in a low-light environment to make your light painting stand out. You don't need complete darkness, but keep in mind that leaving the shutter open for extended amounts of time will magnify any light present – your image may look much brighter than your real-life environment!

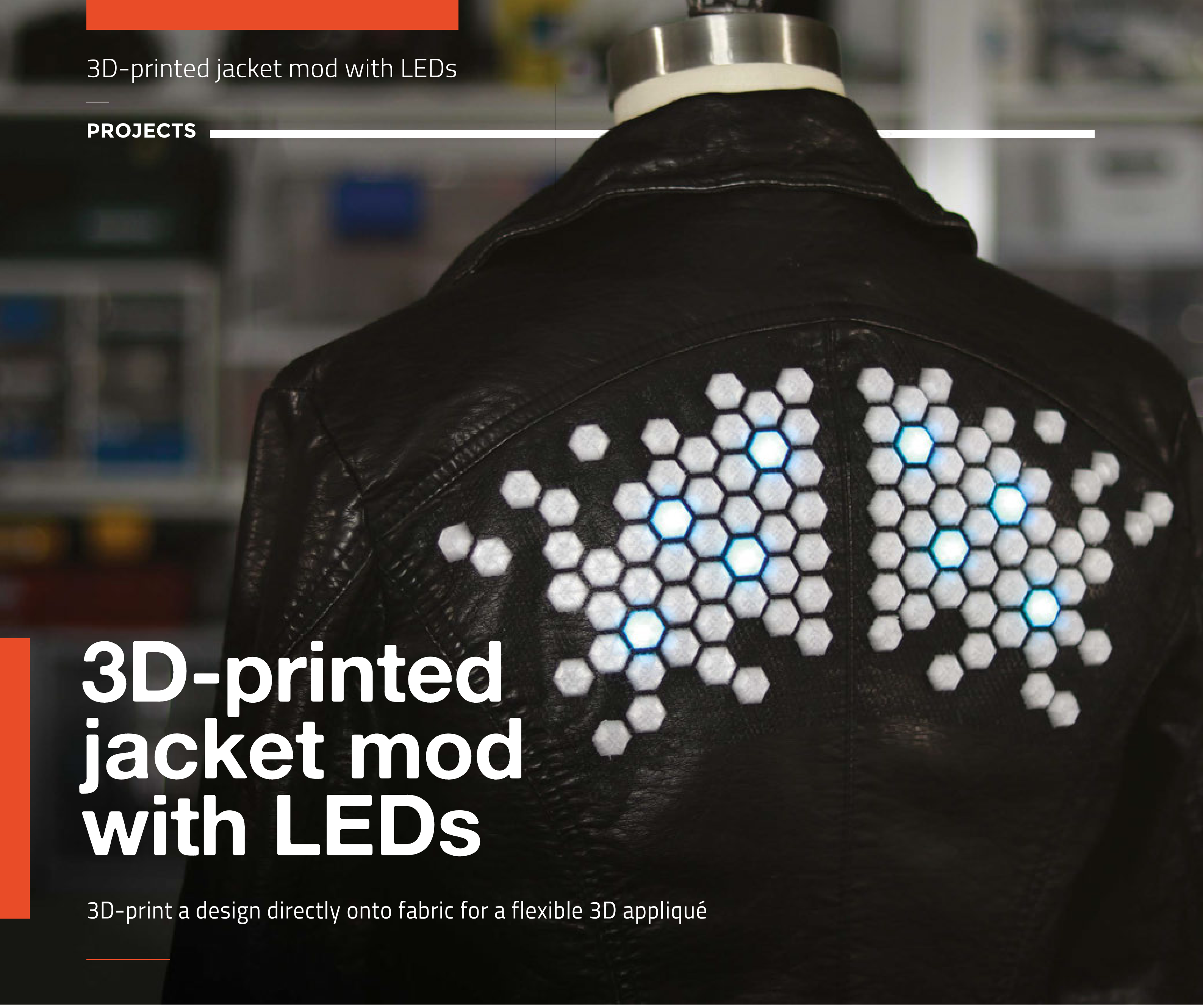
**Procedure:**

- Set up your shot, put your camera in place, and sort out your settings and focus.
- Trigger the camera's shutter to open.
- Do your light painting magic! Activate the NeoPixels on your glove by holding your fingers together. Wave your lights around in front of the camera, making squiggles, lines, words, and whatever comes to mind!
- Trigger the camera's shutter to close (shutter will close automatically if using a timed shutter).
- View the masterpiece you've created in your camera.
- Repeat!

We'd love to see your light-writing creations! Share them with us **@HackSpaceMag** on Twitter. ◻

**Above**
Ready to paint with the Jewel PCB

**Below**
Just wave your hands around for some cool photo effects



# 3D-printed jacket mod with LEDs

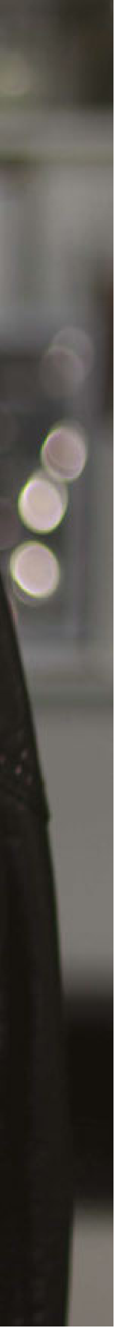
3D-print a design directly onto fabric for a flexible 3D appliqué

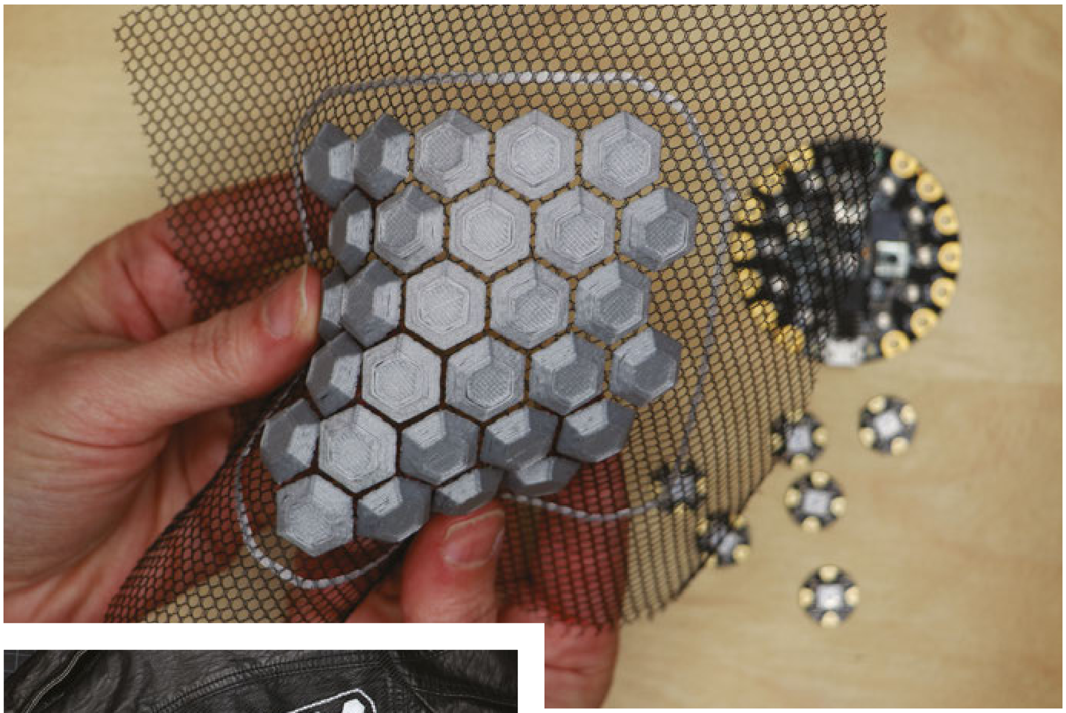
**3D printing is a great way to make custom parts for projects, and it's perfect for wearables!** Making your own models in 3D takes time to master, but sites like **Thingiverse.com** and **MyMiniFactory.com** can get you started with models designed by others in the 3D printing community. There are lots of ways to incorporate 3D printing into wearables, from custom enclosures for your electronics, to sewable diffusers for LEDs, and more. In this project, we'll print directly onto fabric for a flexible 3D appliqué that can be sewn to any garment.

If you haven't already seen the technique of 3D printing onto net or tulle fabric, seek out the work of David Shorey, Penelopy Bulnick, Amie Dansby, and others – and prepare to be amazed! Printing a repeating pattern on fabric can create fluid, flexible panels that mimic feathers, chain-mail, or even dragon scales.

In our project, we'll use a simple pattern of extruded hexagons, and make two 3D-printed panels that will be sewn to the back of a jacket. We'll pattern our pieces directly off the jacket so that they fit perfectly. Translucent filament is a fantastic diffuser for LEDs, so we'll add some lights to the back of the jacket. We'll control our LEDs with a Circuit Playground Express, using its on-board buttons to switch between animations. If you've never programmed a microcontroller before, the Circuit Playground Express (CPX) is the perfect place to start. Coding the CPX is simple with MakeCode, an online block-based visual programming tool from Microsoft. The finished piece will be a programmable, 3D-printed LED jacket, fit for your next cyberpunk adventure!

Printing on fabric is definitely an experimental technique, so expect to do some tests to dial in the settings on your printer first. Use synthetic netting, mesh, or tulle fabric for this. These fabrics have large





**Left**
Our 3D patch, ready to sew onto the jacket

**Above**
The main components for the project

openings between their yarns that allow the 3D-printed layers to fuse together around them. In a nutshell, the process is as follows. Print the first few layers directly on the printer bed, as you'd normally do. Then, pause your print and lay the fabric down over the already printed layers. With the fabric held in place using tape or clips, resume the print and let it finish. When complete, the fabric will be sandwiched into the base of your print, connecting the separate shapes of your pattern together into a flexible panel.

For this project, we recommend laying the fabric down after just one layer, so that the finished panel will be as flat on the underside as possible. You can start with our hexagon model, but you may need to rearrange the hexagons in a 3D modelling program to fit your jacket. You can create your own model, or search on **Thingiverse.com** or **MyMiniFactory.com** for other models designed for printing onto fabric. It is well worth doing a web search before starting, to get a better idea of the process. If you're new to 3D printing, practise by printing regular models (not on fabric), to familiarise yourself with your printer and the troubleshooting required for great results with this medium.

Choose a jacket with a lining so that the wiring can be completely hidden and protected during wear. You'll also need a pocket for housing the CPX and the battery pack. Black faux leather is easy to cut holes in, won't fray, and looks positively Daft Punk!

## EXPERIMENT WITH PRINTING ON FABRIC

Start by doing some test pieces and dialling in your settings for 3D printing on the net fabric. Use a slicer program to convert our **jacket-panel-3D.stl** file (from **hsmag.cc/book2-assets**) into Gcode for your printer. We used Cura, which provides an option to automatically pause the print after a specified layer (in Extensions > Post Processing > Modify Gcode > Pause At Height). The script also moves the printer head out of the way for easy access. Set the standby and resume temperatures to match your print temperature, so that the printer can resume quickly without needing to reheat. Note: modifying Gcode is an advanced technique, and can cause damage to the printer if done incorrectly. Proceed with caution, and do not attempt without full understanding of the modifications you are making.

Whether using a Gcode script or pausing manually, do some research on how to pause a print with your specific slicer and printer. The 3D-printing community is full of wonderful contributors generously sharing →

## YOU'LL NEED

- Jacket with lining and pockets
- Synthetic net or tulle fabric
- Blue painter's tape
- 3D printer
- Translucent filament (shown: Matterhackers clear PLA)
- Circuit Playground Express
- Flora sewable NeoPixels
- AAA ×3 battery pack with on/off switch
- Set of three-pin connectors such as JST PH
- Silicone-coated stranded wire
- Soldering tools and supplies
- Heat-shrink tubing
- Hand-sewing supplies

"**Synthetic fabrics are sensitive to heat.** It's a balancing act between good adhesion and protecting your fabric from damage"

**Above** Measure the panel on your jacket so you can make your 3D print the correct size

their experiences and methods, and that information is extremely helpful when attempting this project. Also, consider helping the community grow by sharing your experience and results so others can learn from you too!

Other settings you'll want to tune include:

**Bed Temperature** – Synthetic fabrics are sensitive to heat (they're basically plastic too!) and in some cases you may not want to expose them to a heated printer bed. It's a balancing act between good adhesion and protecting your fabric from damage. In our project, the synthetic netting held up fine with our regular bed temperature for PLA (60 °C).

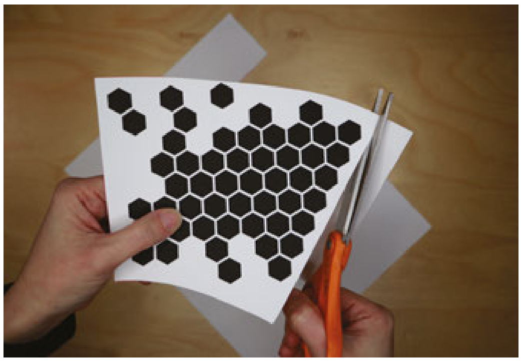
**Skirt** – Printing with a skirt (a solid line around the model, but not touching it, a few layers high) can be useful for holding the tulle in place during printing. We were able to get satisfactory results without one, but if the tape is not holding your fabric down well enough, you might try printing with a skirt. After printing, you can cut the skirt off, but keep in mind that you will need 1 cm or so of seam allowance for sewing the panel to your jacket. You could also choose to incorporate it into your design as an outline.

Run through the entire procedure several times:

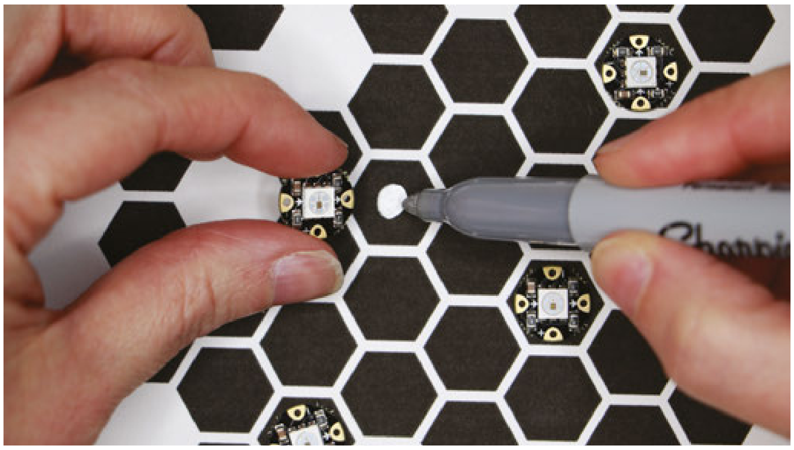
1. Start the print.
2. Pause your printer after the first layer. (Or, wait for the automatic pause if you inserted a pause command in your Gcode.)
3. Tape your fabric in place on the printer bed, then resume the print. When placing the fabric, be very careful not to let it touch the hot printer nozzle or it will melt. Tape it down on all sides so that the fabric is held flat and taut directly on top of the print. Place the tape far enough outside the print so as not to be in the way while printing. You'll want to keep the pause as brief as possible, for good adhesion of the two layers separated by the pause. This is the trickiest part of the process, and doing a few practice runs before your final pieces will help.
4. When finished, let the printer bed cool completely before trying to remove the print. Do not pull on the fabric to release the print, or it may tear.
5. When you've got satisfactory results, you're ready to move on!

## MAKE A PATTERN

If you're using our hexagon model as is, you can skip right to the printing. But if you're making a design



Above
3D-printing the panels on fabric



Far Left
The 2D print, ready for the circuit design

Left
Little lumps of 3D-printed loveliness

Below
Marking out the position of the LEDs on the jacket

from scratch, you'll want to create a custom pattern based on your jacket. Here's how to do it.

Use blue painter's tape to make a pattern of the upper back of the jacket. The panels will be symmetrical, so you only need to do one side. Place overlapping pieces of tape on the jacket on an area larger than your design will be. Then, use a marker to draw the shape you want your fabric panel to be. Pull all the tape off in one piece, and stick it down to a piece of paper. There's your pattern!

Now you can scan this pattern into your computer and trace it in a vector program, like Adobe Illustrator, to digitise it. Then you can create your 3D design using the traced vector shape as a guide, and know that it will be the correct size and proportions for your jacket. Remember to leave 1 cm or so around the edges plain, for sewing. If your design is symmetrical, like ours, you can mirror the design to create the second side.

To check your design, print a 2D version on paper and cut it out. Lay it over your traced pattern to make sure the size is still correct. Then, lay the printed pattern over your jacket to make sure that you like the placement of everything.

## PRINT YOUR FABRIC

Using the settings you determined with your test prints, start printing your first panel. Use a piece of your net fabric large enough so that the painter's tape will be placed close to the edges of your printer bed. You'll cut the panel down to size after printing. After the first layer, pause, tape your fabric in place, and resume. This is where all that practice pays off. Watch the print for a couple of layers to make sure the insertion was successful.

## BUILD CIRCUIT

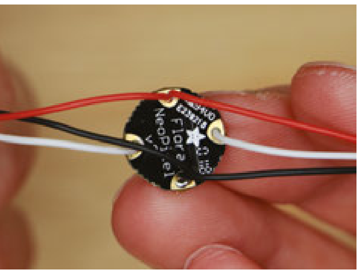
While your fabric is being printed (each of our panels took about 2.5 hours), work on your circuit! Print and cut our 2D design file, making sure to print at 100% scale (or use the printed pattern piece you made for checking your design). Lay out your NeoPixels where you want them to go in the design. You'll get great diffusion by placing them directly under the 3D-printed hexagons. Mark the centre of each hexagon that will have a NeoPixel, and use an awl to make a hole at each mark. Then, lay the pattern in place on your jacket and mark the placement of each hole. Flip the pattern over to mark the second side.

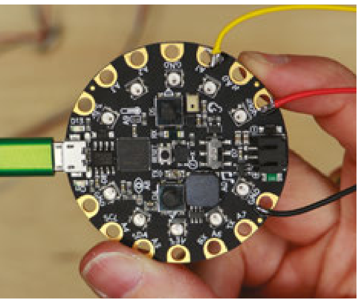
On the inside of your jacket, use a seam ripper to open the centre seam of the jacket lining, exposing the inner workings of your jacket. Make the opening large enough so that you can move the lining out of the way when cutting the holes for the NeoPixels. →

**Above** Using blue tape as a guide for cutting a hole for the NeoPixel

**Below** Wiring a Flora NeoPixel

**Bottom** The first NeoPixel in the chain is connected to the CPX





Use a craft knife to cut a square hole at each mark for the NeoPixel to show through, making sure not to cut into the jacket lining. A small square of blue tape works well as a cutting guide.

Use your template to measure the distances between each NeoPixel in your design and cut wires to length for your NeoPixel chain. For the first NeoPixel in the chain, cut longer wires that will reach the pocket on the front of the jacket where your Circuit Playground Express will live. Add a few centimetres to all your wire lengths for insurance. Note the small white arrow on each NeoPixel that indicates the direction that data flows, from in to out. When chaining the NeoPixels together, all the arrows must point the same way.

To allow the NeoPixels to lay flat against the jacket fabric, feed wires into the back of each NeoPixel and solder on the front. Start by soldering a long wire to the data-in pin on the first NeoPixel. The power and ground pins will have two wires each: one long wire (for connecting to the Circuit Playground Express), and one shorter wire (for connecting to the next NeoPixel). Lightly twist the two wires together before feeding them through the hole and soldering in place on the first NeoPixel. Then, solder a short wire to the data-out pin, and move on to the next NeoPixel.

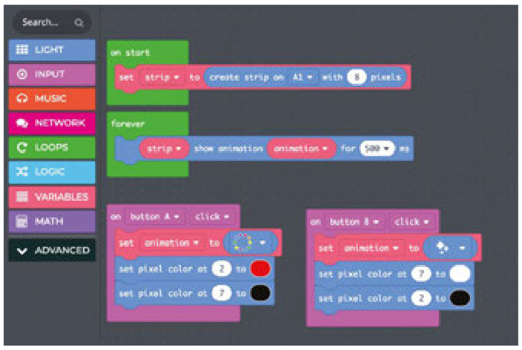
Repeat the above process to keep adding NeoPixels to the chain, continuing to connect power to power, ground to ground, and data-out to data-in on each subsequent NeoPixel. The last pixel will have only one wire for power, ground, and data-in.

"**We'll use two NeoPixels on the CPX** to tell us which animation is currently showing on the rear"

For easy installation, add a three-pin connector between the first NeoPixel (at the beginning of the long wires) and the Circuit Playground Express. This will make it possible to attach and detach the Circuit Playground Express, and make installation easier. To connect the Circuit Playground Express to the first NeoPixel, solder the connector wires so that Vout connects to power, GND connects to ground, and A1 connects to data-in.

## PROGRAM CPX

Your print is probably still going, so let's do some coding while we wait. We're going to program the Circuit Playground Express with MakeCode, and use the on-board buttons to choose between two animations. Since you won't be able to see the lights on your back, we'll use two NeoPixels on the CPX to tell us which animation is currently showing.



```
on start
    set strip to create strip on A1 with 8 pixels

forever
    strip show animation animation for 500 ms

on button A click
    set animation to
    set pixel color at 2 to
    set pixel color at 7 to

on button B click
    set animation to
    set pixel color at 7 to
    set pixel color at 2 to
```

**Figure 1** The MakeCode program for our jacket

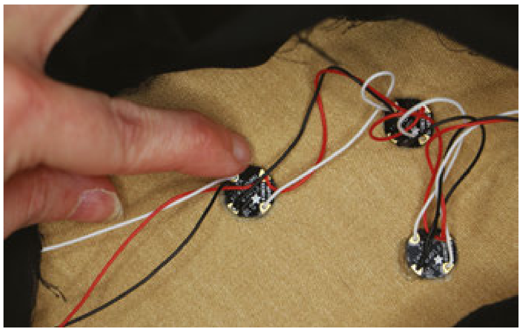
Before running any power through your circuit, lay the chain of NeoPixels out flat on your table so that no uninsulated connections are touching each other. A short circuit can cause damage and be dangerous, so don't skip this step. Plug the CPX into your computer with a micro USB cable, and navigate to **makecode.adafuit.com** to get started!

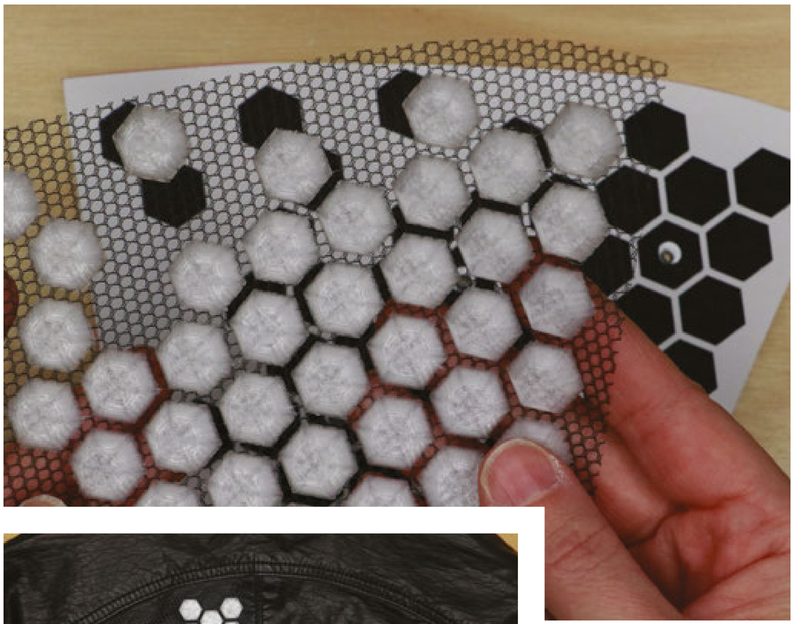
If this is your first time working with MakeCode, Adafruit has a handy guide that will get you up and running in no time: **learn.adafruit.com/makecode**. The guide also contains helpful information on downloading and flashing your code to the Circuit Playground Express when you're all done. Here's a look at how the MakeCode program (**Figure 1**) for this project works…

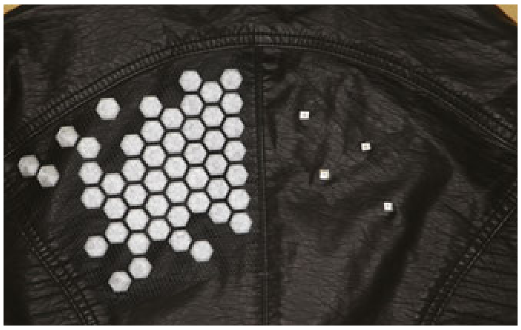
In the 'on start' block, we use a NeoPixel block from the Light menu to create our NeoPixel chain as an object called strip. In this block, we also tell the CPX that the chain is connected to the A1 pin, and contains eight NeoPixels.

In the 'forever' loop, we simply tell the Circuit Playground Express to play an animation for 500 milliseconds. Because it is a loop, it will play this animation over and over seamlessly. We could choose just one animation to play forever, but we want to use the two buttons on board the Circuit Playground Express to show two different animations. To do this, we've also created a variable in this line called 'pattern'.

Finally, we've created two blocks, one for button A and one for button B. In each block, we've chosen to set the variable pattern to a different animation







**Above** Everything coming together in the final assembly

pattern – button A sets it to a rainbow, and button B sets it to a white twinkle. We've also selected a nearby pixel from the on-board ring on the Circuit Playground Express to illuminate depending on which button has been pressed. This will make it easy to keep track of which button you've just pressed, and which animation pattern is currently showing. Button A will illuminate turn the pixel next to it red, and button B will turn the pixel next to it white.

When your code is complete, click Download and follow the on-screen instructions to install the code on your board. Press the buttons on your board and watch the animations change. When everything is working as you want it to, you're ready to sew it in!

## INSTALL CIRCUIT

Unplug the Circuit Playground Express from the NeoPixel chain at the three-pin connector. Behind the lining on the inside of your jacket, locate the back of the front pocket that will hold the Circuit Playground Express. Use a seam ripper to open a bit of the seam so you can feed the three-pin connector end of the NeoPixels into the pocket. Pull the connector through, and out of the pocket a little way. On the inside of the lining, before the wires enter the pocket, make a loop in the wires and anchor it with a few hand-stitches to prevent pulling on the NeoPixel chain. Sew the hole in the pocket closed around the wires.

Then, start gluing the NeoPixels in place on the wrong side of the fabric at the back of the jacket. Faux leather should take hot glue well, but it's safest to use a low temperature setting. Test in a small inconspicuous area first. Apply glue to the front of each NeoPixel, then place them so that the raised square LEDs show through the square holes you made. On the back of each NeoPixel, cover each solder point with a dab of hot glue for strain relief and added protection.

## ATTACH PANELS

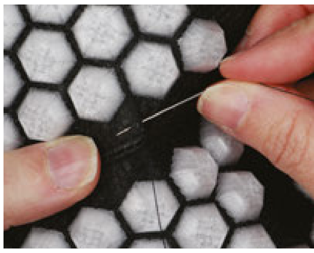
When both of your 3D-printed fabric panels are done printing, rejoice! Using your printed paper patterns, cut each panel into the correct shape. Then lay the panel on top of your LEDs and make sure everything lines up nicely, and the LEDs will be covered correctly.

You can use a tiny misting of spray glue on the back of each panel to hold it in place while sewing. Hand-sew around the edges of each panel with a long straight stitch to attach securely. Be deliberate about each needle hole you make, as all holes are permanent in leather, pleather, and vinyl.

## FINISHING

Close up the lining at the centre seam by whip-stitching over the section you opened up. Plug the battery pack into your CPX, and flip the switch on to light up your 3D-printed fabric panels!

We've touched on a lot of different processes in this project, and now you're primed to take them further! Share your experiments with us at **hackspace@raspberrypi.org**!



**Above** A few stitches hold the panel in place

# Laser-cut NeoPixel necklace

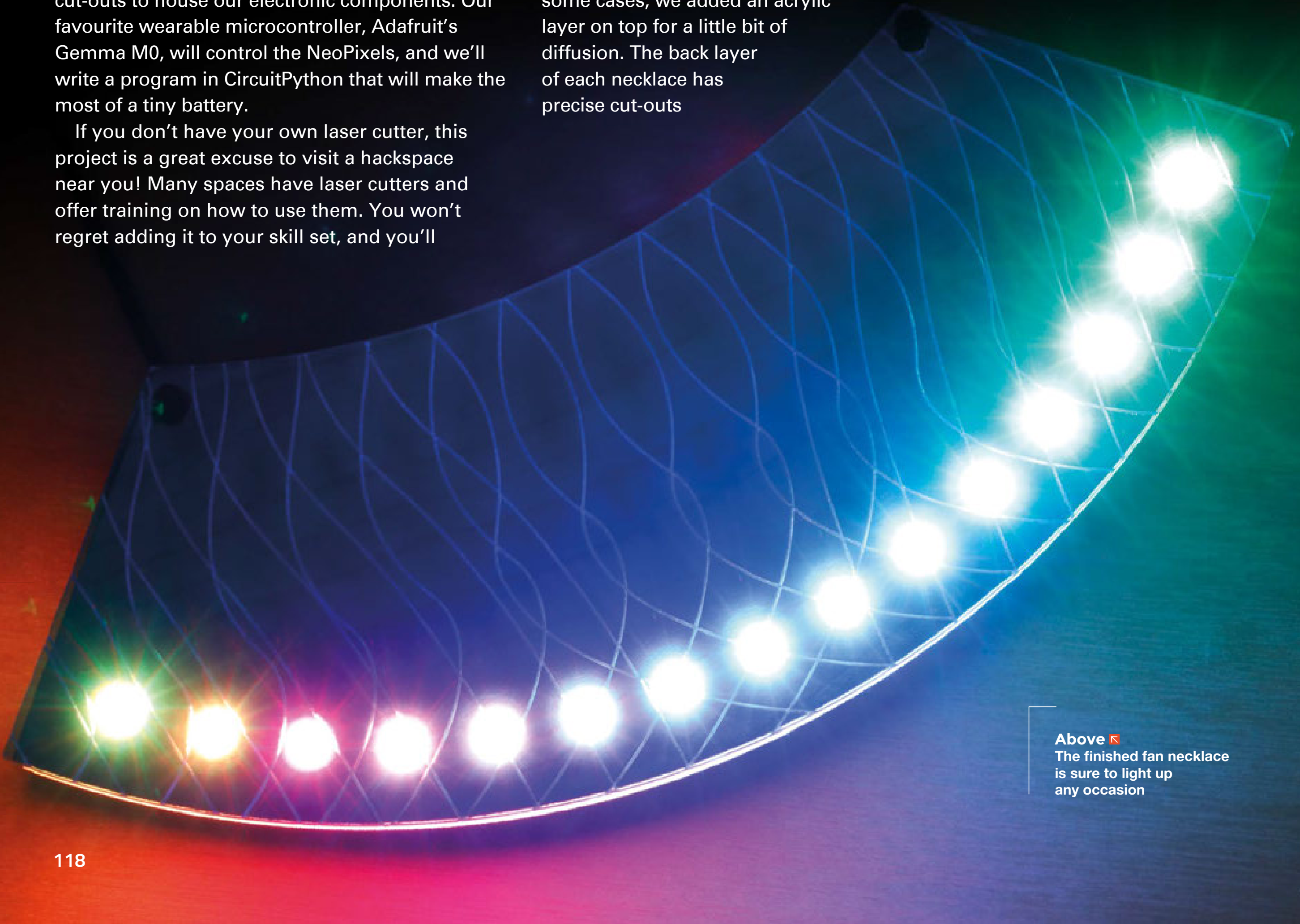
Create a stunning piece of jewellery that really shines

**Jewellery is meant to be shiny, so it's the perfect place for NeoPixels!** The trick is: how to get everything you need for a NeoPixel circuit to fit inside a small piece like a pendant. Enter the laser cutter!

Laser cutters are great for precision cuts in materials like thin wood and acrylic. In this project, we'll use a laser cutter to create a sleek necklace that shines bright with NeoPixels from Adafruit. We'll build the necklace in layers with precise cut-outs to house our electronic components. Our favourite wearable microcontroller, Adafruit's Gemma M0, will control the NeoPixels, and we'll write a program in CircuitPython that will make the most of a tiny battery.

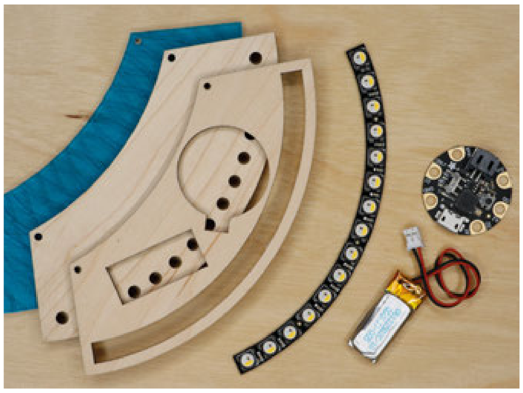
If you don't have your own laser cutter, this project is a great excuse to visit a hackspace near you! Many spaces have laser cutters and offer training on how to use them. You won't regret adding it to your skill set, and you'll be supporting your local hackspace too. If you simply can't access a laser cutter on your own, try sending the files out to a service like Ponoko. Or, take this concept of layering and apply it to materials you can cut by hand, like balsa wood, foam, or even cardboard.

Each of our necklace designs consists of several layers. All have a layer that hides the electronics but shows the NeoPixels through tiny holes that align perfectly with each LED. In some cases, we added an acrylic layer on top for a little bit of diffusion. The back layer of each necklace has precise cut-outs



**Above** The finished fan necklace is sure to light up any occasion

Below
We don't have scratch and sniff, but this smells lovely. Trust us


Above
Ready for assembly

to hold the NeoPixels, microcontroller, and battery in place. The NeoPixels face the front of the necklace, and the Gemma M0 faces the back, making it easy to access the on/off switch while wearing. This tutorial will step you through the build process for the fan necklace, which illustrates this layering concept. Then, you can use what you've learned to make the other necklaces, or design your own!

The arc of NeoPixels in our fan necklace is actually one quarter of a giant ring of NeoPixels – you can solder four of these arcs together to make a complete circle. Instead of using it for its intended purpose, we're using the quarter-circle as the focal point of a bib necklace. We started with the shape of the NeoPixel arc, and widened it to create the fan shape. Then we topped it off with a piece of blue acrylic for diffusion, and added delicate, swirling line art so that the necklace will look just as great when the NeoPixels are powered off.

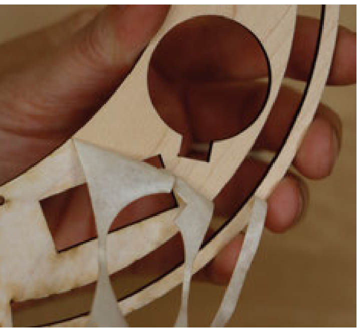
The battery in this project is a tiny lithium-ion one. These are great little powerhouses that are perfect for driving NeoPixels while keeping the project small. However, these batteries can be dangerous if not handled properly and must be treated with care. Be careful not to puncture, compress, or otherwise abuse this type of battery. Ensure that it is held securely but not squished inside your project. The chemistry in these batteries can vary among brands, so it's important to only charge them with the chargers recommended by the battery's manufacturer. Do not cut the wires of the battery, and do not pull on the wires to unplug the JST from the Gemma or charger. Inspect the battery often for damage or bulging. If any change is observed in the battery's shape, dispose of it properly according to your local hazardous waste regulations. Because these batteries require such careful handling, this project is for intermediate to advanced makers who are comfortable working with power supplies.

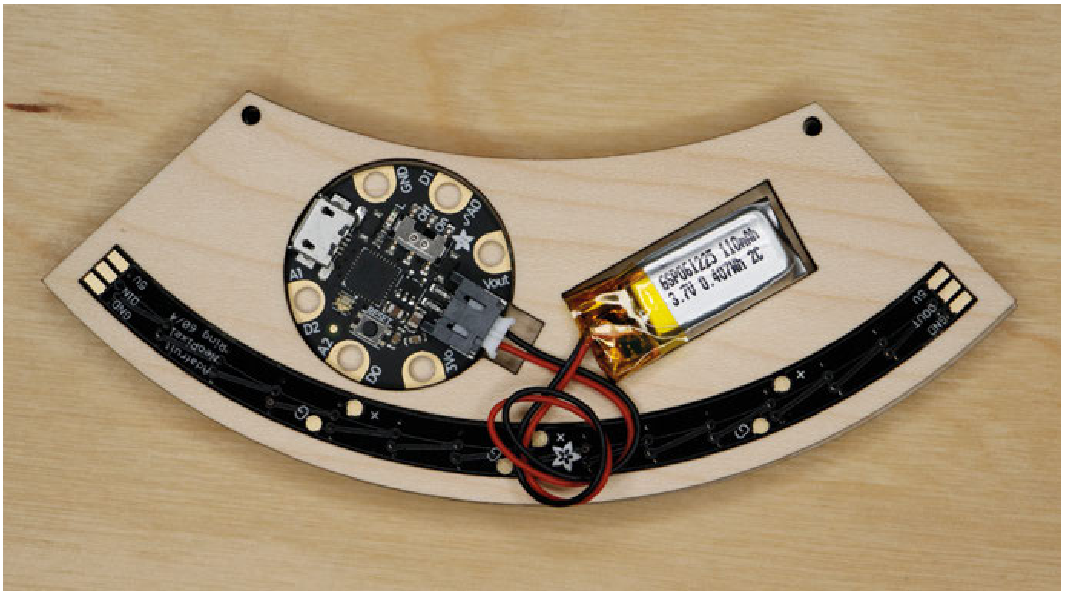
## CUT THE PIECES

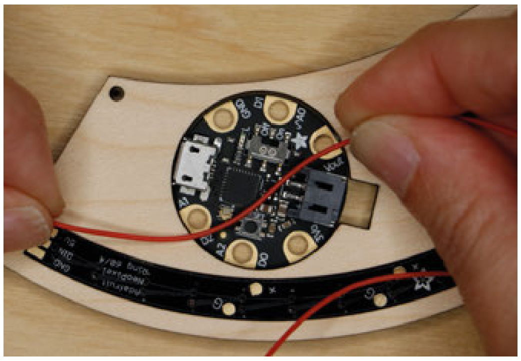
Use the provided files (**hsmag.cc/book2-assets**) to cut your pieces out on the laser cutter. For our design, the two back pieces are cut out of wood, and the front piece is cut out of acrylic as a diffusion layer. The front piece has two files: one for cutting the shape, and one for scoring line art on the surface. Be careful not to accidentally cut the scoring layer. You may need to experiment with the settings on your laser cutter to get the depth you prefer. We highly recommend doing some test cuts before cutting the final piece. →

## YOU'LL NEED

- **Gemma M0**
- **100 mAh lithium-ion battery (don't forget the charger!)**
- **NeoPixel 1/4 60 Ring – 5050 RGB LED** (product #1768)
- **Thin wood for laser cutting (we used Glowforge medium maple hardwood)**
- **Acrylic for laser cutting (we used translucent light blue acrylic)**
- **Double-sided tape**
- **Jewellery cord**
- **Clasp, cord ends, split jump rings**
- **Jewellery crimping pliers**
- **30 AWG silicone-coated stranded wire**
- **Soldering tools and supplies**







**Above & Right** All the laser-cut pieces should fit snugly together

Materials that are sold for use in laser cutters come with protective film on both sides to prevent scorching from the laser. Leave this film on the material while cutting. When your pieces are cut, you can remove the protective film on both sides. This can be a bit tedious when dealing with the tiny scored details, but just take your time and enjoy the reveal!

> **With the NeoPixel arc face down,** gently press it into place at the bottom of the piece. It will be a snug fit and should stay in place

## ASSEMBLE THE NECKLACE

Start with the back layer that will hold the electronic components. Lay the piece down on your surface, with the hole for the Gemma M0 on the left, and the battery hole on the right. With the NeoPixel arc face down, gently press it into place at the bottom of the piece. It will be a snug fit and should stay in place without additional support. If your NeoPixel arc doesn't fit, use a small file to widen the hole in the wooden piece where you need to. If the arc is loose, add a few small dabs of hot glue to keep it in place.

Lay the component piece on top of the other wooden piece, and check that each NeoPixel lines up with its hole in the front piece. With these two layers stacked together, put the Gemma M0 into its hole. If the Gemma M0 has any burrs around its edge, you may need to carefully file them down with a nail file first. The JST battery port on the Gemma M0 should align with the square notch pointing toward the battery hole. Plug your battery into the Gemma M0 and check that it fits into its hole as well. Make sure the battery fits comfortably and is not compressed. Use a file to enlarge any holes that don't fit their components. When you're satisfied that everything fits, remove the battery and set it aside before proceeding.

## BUILD THE CIRCUIT

It's time to wire everything together! We'll be soldering the connections as follows: GND on the NeoPixels to GND on the Gemma, data-in on the NeoPixels to D1 on the Gemma, and 5 V on the NeoPixels to Vout on the Gemma.

We want to reduce bulk on the back of the necklace as much as possible, so we'll keep our wires short and neat. Cut three pieces of wire that reach from the pads on the data-in side of the NeoPixel arc to their corresponding pads on the Gemma M0. Make sure the wires are a little longer than you need them for now; we will cut them down exactly later on.

Strip and tin one end of each wire, and tin the pads on the data-in side of the NeoPixel arc. Carefully solder one wire to each pad, being sure not to bridge the contacts on the NeoPixels – they're very close together, and it's easy to touch two of them by mistake. Solder the wires so that they point back towards the Gemma.

Take the end of each wire and hold it over its pin on the Gemma M0. Cut each wire exactly to length, leaving about 1 mm for soldering. Strip and tin about 1 mm of each wire, then tin GND, D1, and Vout on the Gemma M0. Solder each wire to its pad. Tweezers can be helpful here for precision. Don't worry too much if you burn the wood a little with your soldering iron – it won't show on the front.

## PROGRAM THE GEMMA M0

With the circuit assembled, let's program the Gemma M0 with CircuitPython and make our LEDs light up! Since we're using a tiny battery, we'll create an animation that only illuminates the NeoPixels for a short amount of time. This will increase the battery life of your necklace greatly while still allowing for a high impact look. Our code will intermittently send a colour, chosen at random, across the NeoPixel strip from left to right. Then the NeoPixels will darken from left to right and pause before sending the next colour.

Plug a micro USB cable into the Gemma M0 and connect it to your computer. The Gemma will show up as a drive called CIRCUITPY.

As always, make sure your Gemma M0 has the latest version of CircuitPython and updated libraries. Follow the guide at **hsmag.cc/ZKvDqa** to make sure your Gemma M0 and its libraries are up to date before starting to code. The code below uses the **neopixel.mpy** library, so you'll need to copy the latest version of the library into the **lib** folder on the CIRCUITPY drive.

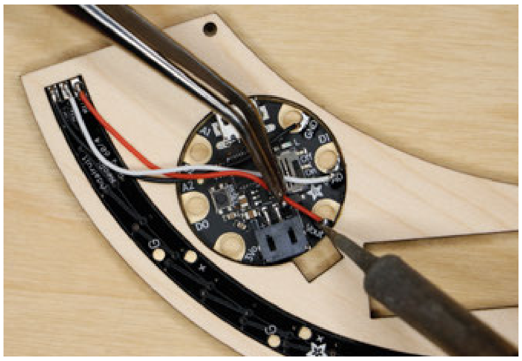
You can use any text editor of your choice to program the Gemma M0, but we like to use a Python code editor called Mu. To learn more about Mu and how to use it, visit Adafruit's guide on Mu here: **hsmag.cc/NtvMQr**.

In Mu, or your text editor, open the **main.py** file on the Gemma M0. The Gemma comes preloaded with example code, but we'll write our own colourful animation here instead. Select all the text in the file and delete it to start from scratch.

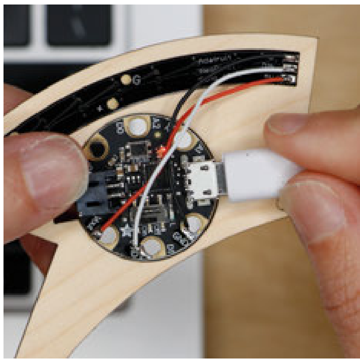
The first step is to import the libraries we'll be using and set up the NeoPixel arc as an object. We'll also create some variables that are common to NeoPixel sketches: `pixel_pin` and `num_pixels`. Adding these two variables at the top will make it easy to modify this code for other projects with different numbers of NeoPixels.

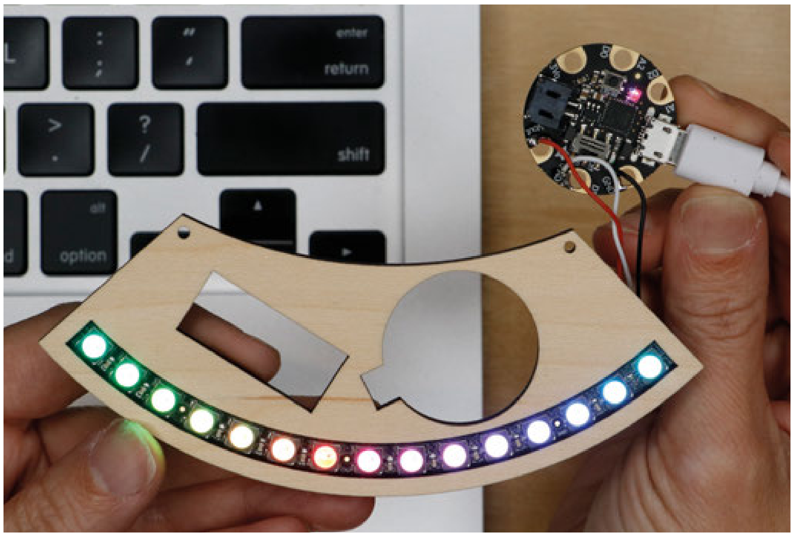
```
import time
import board
import neopixel
import random

pixel_pin = board.D1
num_pixels = 15

pixels = neopixel.NeoPixel(pixel_pin, num_pixels,
brightness=0.1, auto_write=False)
```

Next, we'll use a helper function called `wheel`, which will let us use a single number to assign a colour to the NeoPixels, instead of having to enter separate RGB values as (r, g, b). When we give it a number between 0 and 255, the `wheel` function will return the corresponding colour in (r, g, b) format. Giving the `wheel` function 256 will return (0, 0, 0), which is black. →



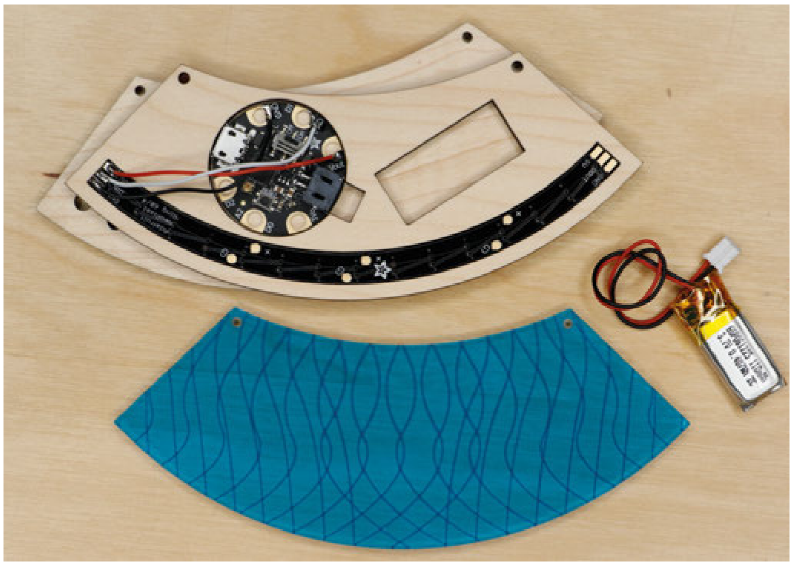
**Above** Cut the wires to the correct length to keep it neat

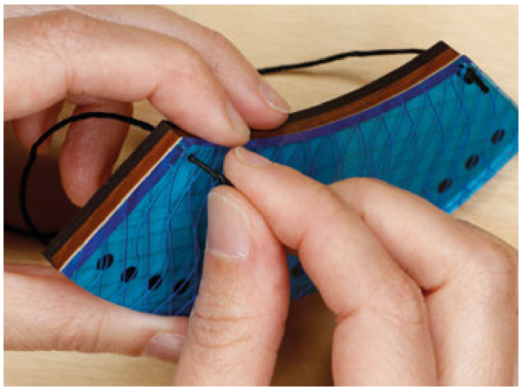




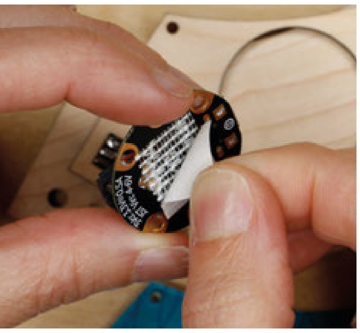
**Below** If you use a different board, you'll need to adjust the design

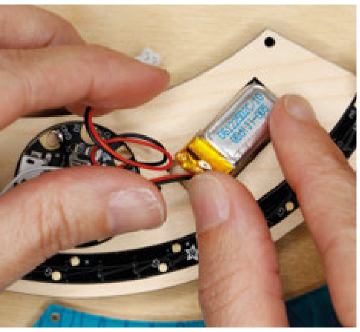
**Bottom** Everything works! Time to put it together





**Above** A final knot to bring everything together





**Above** Sticky tape, a maker's best friend

```
def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b -
back to r.
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170

    return (pos * 3, 0, 255 - pos * 3)
```

Now we're ready for the main loop. First, we need to choose a colour for the NeoPixels. To do this, we create a variable called **`color`** and assign it a random number value between 1 and 256 using **`random.randint`**. If the number generated is between 1 and 255, the NeoPixels will show the corresponding colour from the colour wheel. If 256 is generated, the pixel will turn off (black). If you're using a code editor like Mu, printing generated values like this in the serial monitor can help when debugging.

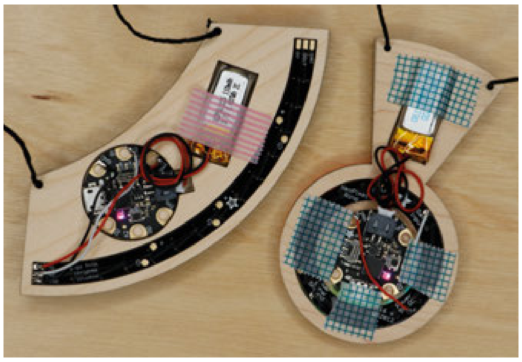
```
while True:
    color = random.randint(1, 256) #choose a
random color, includes black

    print("color", color)
```

Next, we have two **`for`** loops. The first **`for`** loop uses the **`color`** variable to fill the strip of pixels one at a time, starting at the first pixel (0), until the whole strip is filled. When that task is complete, the second **`for`** loop will take over, and start filling the strip with black, turning off the pixels. This will create a sort of chase effect from left to right on the strip (or clockwise in a circle if using a NeoPixel ring), with a new, randomly chosen colour each time the strip fills. Changing the **`time.sleep`** values will change the speed of the animation, so play with this until you like the effect.

```
    for j in range(num_pixels):
        pixels[j] = wheel(color) #set (the random
pixel + j) to the random color
        pixels.show()
        time.sleep(.01)
        j = j+1 #add 1 to j every time

    for i in range(num_pixels):
        pixels[i] = wheel(-1) #same as above but
set pixels to black
        pixels.show()
        time.sleep(.01)
        i = i+1

    time.sleep(2)
```

Saving the edited **main.py** file on the CIRCUITPY drive will automatically load the code onto the Gemma and you should see the colour chase animation appear on your necklace. If your NeoPixels are lighting up as expected, hooray! You can unplug the micro USB cable and move on to installing the Gemma M0. If your NeoPixels don't light up, double-check your connections and make sure you haven't bridged the tiny pads on the NeoPixel arc. For help with troubleshooting and more information about programming the Gemma M0 with CircuitPython, visit **hsmag.cc/ZKvDqa**.

**Above**
The acrylic helps diffuse the light with a little flair



## INSTALL THE ELECTRONICS

When your NeoPixels are lighting up as expected, you can tack the Gemma M0 into place. You'll want to make sure that you can access the micro USB port in the future, in case you want to update your necklace with new animations. You'll also need to be able to remove the battery's JST plug from the connector on the Gemma M0 for charging. For this reason, we recommend taping the components in place instead of permanently gluing them in. Double-sided carpet tape is both thin and strong, and works well for this type of application.

With both wood layers stacked together, use double-sided tape to affix the Gemma M0 in place in its hole. It will be stuck to the back of the front wooden piece. Again, make sure the JST port is facing the square notch. Use another piece of double-sided tape to stick the battery in place as well.

The battery may have long wires, but instead of shortening these, just wind them gently into a low profile coil on the back of your necklace. You can use a piece of single-sided tape to hold the coil in place. Do not put any stress on the point where the wires connect to the battery. It's a good idea to stabilise this point with some gaffer tape or a dab of E-6000 glue.

## FINISH ASSEMBLY

Place the acrylic layer in the front of the stack, on top of the first wood piece. With regular translucent acrylic, it's nice to see the scored lines on the front of the necklace, but if you are using fluorescent acrylic you may want the score lines to be next to the LEDs for a cool line-art effect. The choice is yours!

Cut two equal length pieces of jewellery cord. Each piece should be half of the final length of the necklace you want to make. For the necklaces shown here, the cords are about 10 cm long. Thread each piece through one of the holes at the top corners of the stack, from back to front. On the front of the necklace, tie a tight knot at the end of the cord.

At the other end of each cord, use crimping pliers to add cord ends for jewellery findings. Then add split jump rings and a clasp. Split jump rings are coiled so that they are less likely to open during wear. Pull the coil open slightly and wind the ring onto your cord end. Use the same process to add a clasp to one side, and you're done! The Gemma M0 has an on-board on/off switch, so flip that switch on and watch your necklace light up.

> **"With both wood layers stacked together,** use double-sided tape to affix the Gemma M0 in place in its hole"

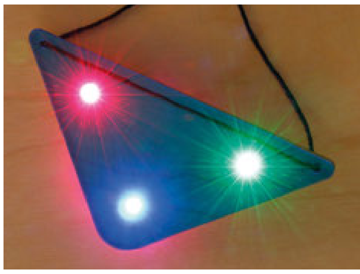
## GOING FURTHER

Once you've made one necklace, it's easy to make more in different shapes. Use our designs (from **hsmag.cc/book2-assets**) or make your own shapes. Instead of diffusing the NeoPixels through a layer of acrylic, try scoring the top wood layer with line art. You can use the same code for a few sewable NeoPixels, or even a NeoPixel ring. Just change the `num_pixels` variable to the number of NeoPixel LEDs in your new design.

Whether you make this project, or something completely different, we want to see it! Show us your builds at **hackspace@raspberrypi.org**!

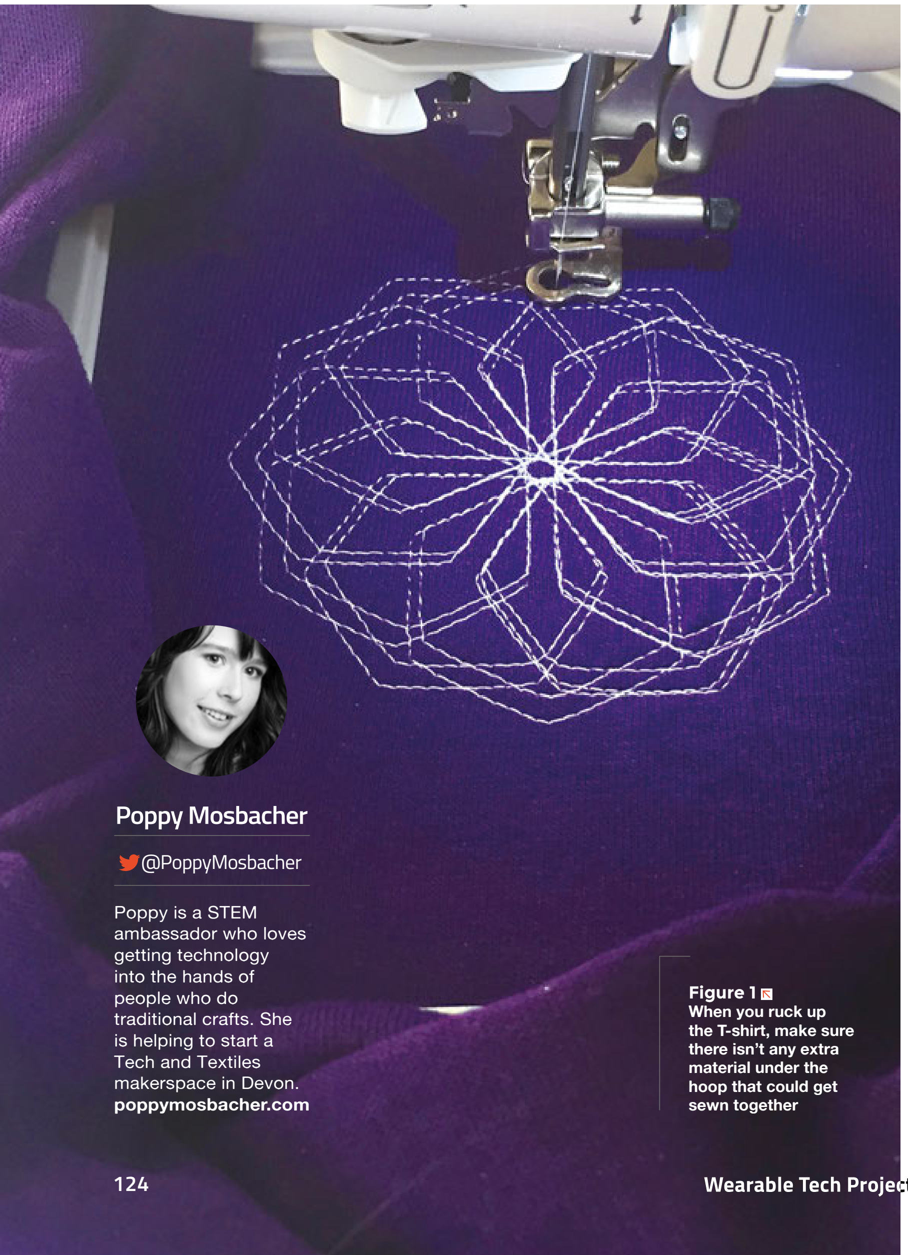
**Above**
If in doubt, tape!

**Below**
Unleash your creative juices and make your own light-up jewellery



# CNC embroidery with Turtlestitch

Automatically stitch original T-shirt designs



Poppy Mosbacher

@PoppyMosbacher

Poppy is a STEM ambassador who loves getting technology into the hands of people who do traditional crafts. She is helping to start a Tech and Textiles makerspace in Devon.
**poppymosbacher.com**

**Figure 1** When you ruck up the T-shirt, make sure there isn't any extra material under the hoop that could get sewn together

**Turtlestitch is a great crossover tool that introduces coding to people who sew, and textile projects to people who code.** You drag and drop Scratch-style blocks to create stunning shapes, set up an embroidery machine, and sit back as the needle automatically embroiders your design on a T-shirt or other fabric.

To open the program, go to **turtlestitch.org** and click on the cute little turtle in the centre of the page. It runs in the browser, so there's no need to install anything.

## FIRST FIND A FLAG

The most common way to start any Turtlestitch design is to click on the palette called Control, on the left-hand side of the screen, and drag the block with the green flag onto the scripting area in the centre.

Then, attach a 'reset' block just below the 'flag' block. The 'reset' block is useful when you make changes to your code. It clears the demo area (on the right) and centres the needle, shown as a turtle, each time the green flag is pressed.

## CODING THREAD

To make the interlocking hexagon design, shown in **Figure 1**, drag a 'repeat 10' block from the Control palette and attach it under the 'reset' block. Click on the number 10 and change it to 12 because the basic hexagon pattern will be repeated twelve times.

If all the hexagons met in the centre of the design, there would be a dense knot of overlapping stitches. So, you need to move the needle a few steps away from the centre before creating each shape. To do this, go to the Motion palette and attach a 'move 10 steps' block inside the 'repeat block'. Click on the number 10 and change the value to 5. This will leave a small unstitched circle in the centre of the design.

To create the hexagon shape, attach a 'repeat 10' Control block under the 'move' block. Change the value to 6 to create the six-sided shape. Go back to the Motion palette and attach a 'move 100 steps by 10 steps' block inside the 'repeat 6' block. The first number sets the length of the sides of the hexagon, and the second is how many stitches fit into that length. Add a 'turn clockwise' block under the 'move' block to create the angles of the hexagon. To work out the value for the 'turn' block, divide 360 by the number of sides in a hexagon; i.e. 360 / 6 = 60, so change the value to 60 degrees.

> Sit back as the needle automatically embroiders your design **on a T-shirt or other fabric**

Add a 'turn anti-clockwise' block under the 'repeat 6' block to offset each hexagon from the next. To work out the value for this Turn block, divide 360 by 12, which equals 30 (because the pattern is repeated twelve times within a circle).

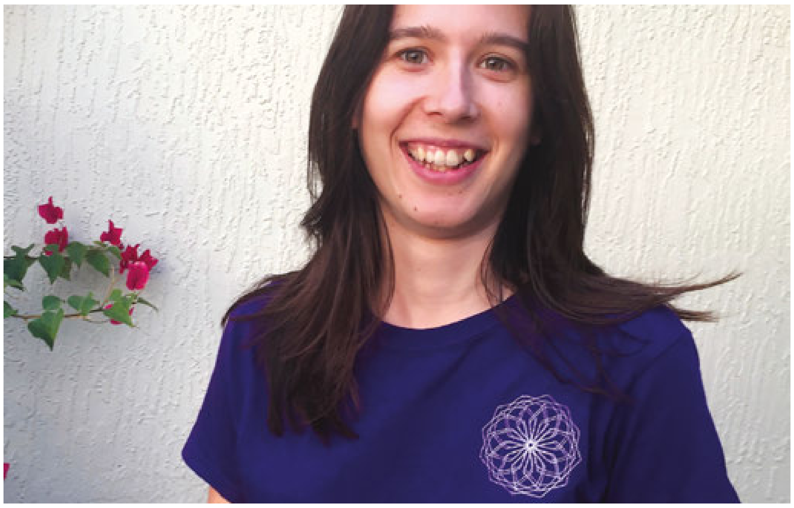
## DOUBLE UP

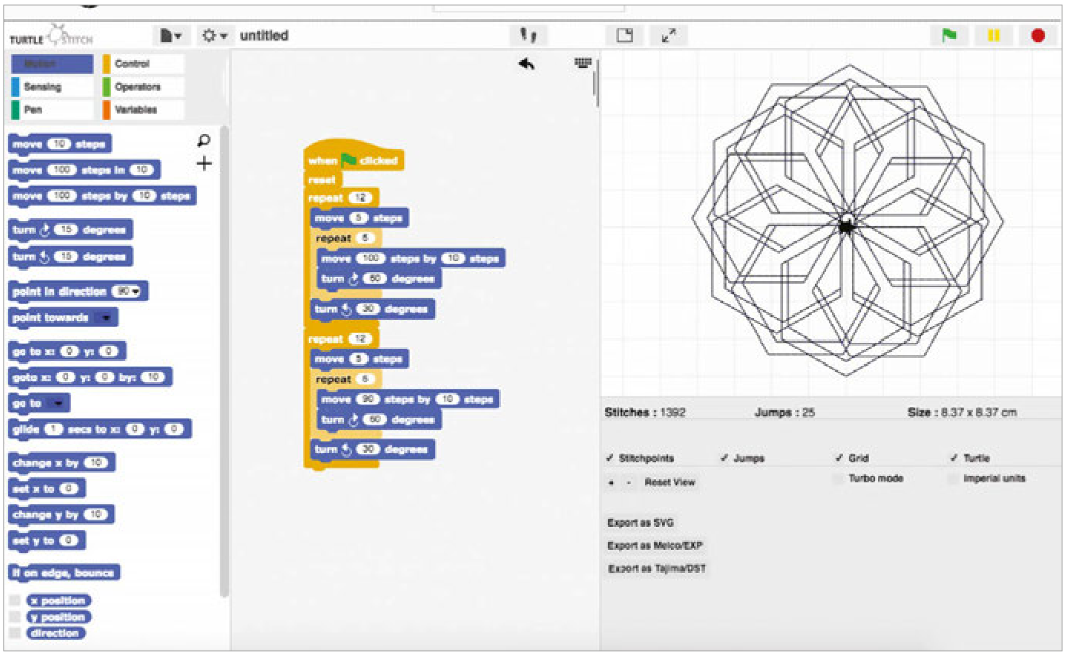
To complete the pattern, add another set of slightly smaller hexagons by clicking on the 'repeat 12' block and select Duplicate. This will copy and paste all the blocks attached underneath it. Add these new blocks to the bottom of your code. Change the value of the new 'move 100 steps' block to 90 to make smaller hexagons. All the other blocks can stay the same.

Press the green flag. What you see in the demo area is what will be stitched. If you press stop before the code finishes running, and save the design, the embroidery machine will also stop at that place.

## PLAY WITH OTHER SHAPES

To experiment, use different shapes as the basis of this repeating pattern, as shown in **Figure 2** (overleaf). Be sure to check the design will fit inside your embroidery hoop before stitching. The size appears underneath the demo area. →





```
when 🏴 clicked
reset
repeat 12
  move 5 steps
  repeat 6
    move 100 steps by 10 steps
    turn ↻ 60 degrees
  turn ↺ 30 degrees
repeat 12
  move 5 steps
  repeat 6
    move 90 steps by 10 steps
    turn ↻ 60 degrees
  turn ↺ 30 degrees
```

Stitches : 1392   Jumps : 25   Size : 8.37 x 8.37 cm

✓ Stitchpoints   ✓ Jumps   ✓ Grid   ✓ Turtle
Turbo mode   Imperial units

## YOU'LL NEED

- **Automatic embroidery machine** that reads EXP or DST file formats
- **Computer** with access to the internet
- **USB memory stick**
- **Plain T-shirt**
- **Scraps of T-shirt fabric** for testing
- **Machine embroidery thread**
- **Embroidery stabiliser**
- **A4 sheet of paper and printer**
- **Four bulldog clips** (optional)
- **Scissors**
- **Pins**

**Above** This design took three minutes to sew

**Left** This is all the code for the interlocking hexagon design

**QUICK TIP**

If you don't want the needle to start in the centre of the hoop, you can add x, y coordinates or 'move' blocks to your code.

Also under the demo area, you'll find three different options to save and export your design. Click on the one that works on your embroidery machine. The design will appear in your downloads folder. Simply transfer to a USB stick and plug into the slot on the embroidery machine. Save your design in the SVG format as well and print a copy on paper. This will help to align the design on your T-shirt later.

## THREADING BY NUMBERS

Modern embroidery machines have built-in guides for setting up. First, use the automatic winder to fill the machine's small bobbin with thread. You can use any colour because it will only be visible on the back of the fabric. Then insert the bobbin in its compartment under the sewing area, and follow the numbers to position the loose end of the thread.

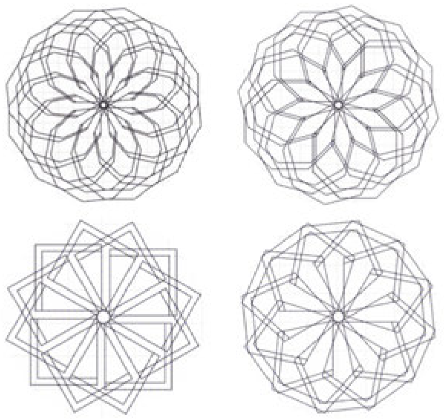
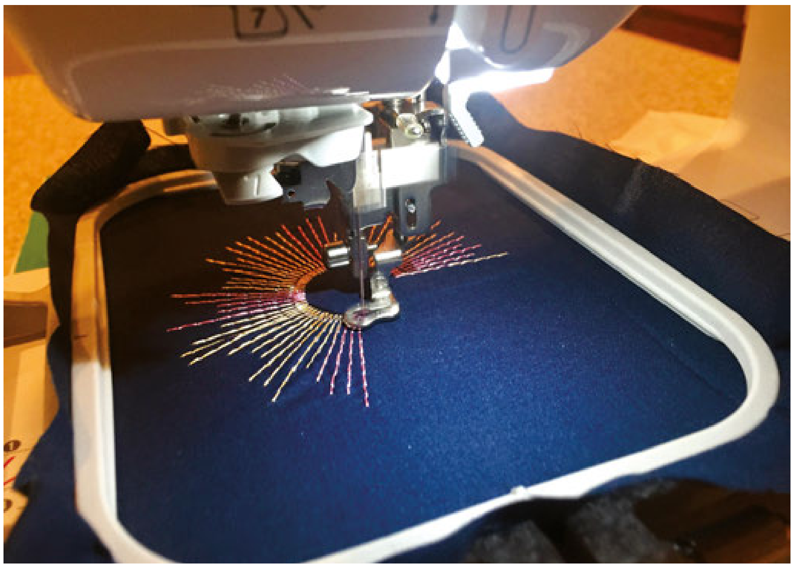


**Figure 2** Use different basic shapes to create new pattern variations



**Below** Create colourful effects with multicoloured thread or by pausing and switching thread part-way through. Code by turtle_fan

Also, follow the numbers to thread your top colour. When you reach the last number, there's usually a small lever to automatically thread the needle.

To do a test run, cut rectangles of spare T-shirt fabric and stabiliser approximately 4 cm larger than your hoop. Loosen the screw on the embroidery hoop and separate it into two hoops. Place the stabiliser fabric over the larger, outer hoop, and then the T-shirt fabric on top. Push the inner hoop into the outer hoop, trapping both layers of fabric. Adjust the fabric so that it is spread evenly over the bottom of the hoop and tighten the screw.

With the presser foot of the embroidery machine in the raised position, slide the hoop underneath it. Then, clip the hoop onto the gantry bar and lower the foot. Push the end of the thread through the hole in the foot.

## HANDS-FREE EMBROIDERY

Find your design on the screen of the embroidery machine. You may be able to rotate or resize the design if you want to. Then, press start. The machine will do the rest and stop when it's finished.

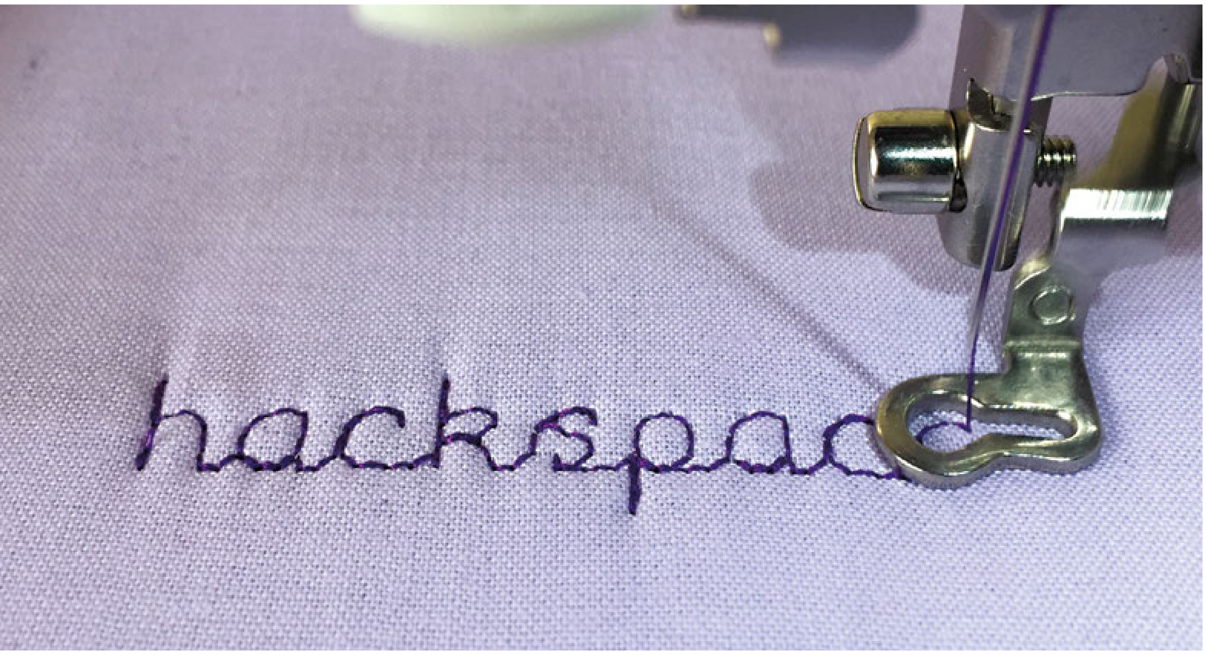
## GETTING YOUR HANDS ON AN EMBROIDERY MACHINE

There might be an automatic embroidery machine at a makerspace or library near you. They're more common in US makerspaces, but with lower online prices and recent improvements in free-to-use software (Turtlestitch and Ink/Stitch), they're starting to become popular in other countries too.

HackPGH in Pennsylvania have had their Brother SE400 embroidery machine for over a year and their president Chad says, "Other than a few needles breaking, it's been running well". It's also possible to hack old sewing machines; see **hsmag.cc/QohssH**.

Some makerspaces have industrial machines, such as the 15-thread machine donated to MakerFX in Orlando, Florida, and the Janome machine at Fab Lab Barcelona. These may work with this tutorial, but the setup will be different.

**Below** Suitable automatic embroidery machines look like regular sewing machines with an added gantry bar. They start from $350 in the US, or £500 in the UK



**Above** Turtlestitch user MicRun has created an alphabet block to personalise your designs: hsmag.cc/GIEGpg

**Figure 3** This Sedona Spiral design, by Turtlestitch contributor Richard, uses variables to increase the size on each turn

Afterwards, you just have to raise the foot and remove the hoop.

To prepare your T-shirt, trim your paper printout to the same size as your hoop, making sure the design is in the centre. Position it on your T-shirt where you want the picture to go. Put pins in the fabric around the edges of the paper, then remove the paper, and put a pin in the centre. Cut a rectangle of stabiliser

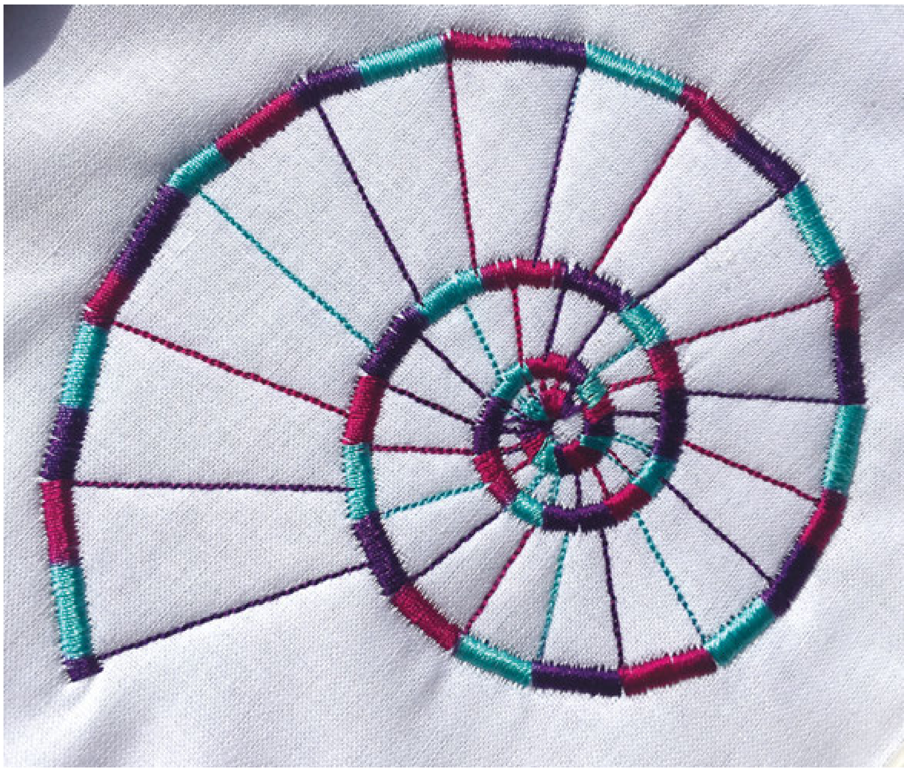
" Simply transfer to a USB stick **and plug into the slot on the embroidery machine** "

the same size as before, and place it inside the T-shirt so that the centre is over the middle pin.

Using the pins for alignment, attach the hoop to the stabiliser and the area of T-shirt you intend to sew. Remove the pins. Slide the hoop under the machine foot and attach as before. Gather up the excess T-shirt fabric and bunch it up around the edge of the hoop. You can also use bulldog clips to fix the fabric to the hoop, but it's not usually necessary. When it's ready to go, press start on the machine. The stabiliser is only needed to stop the T-shirt fabric stretching in the hoop, so trim it away from around your design when it's finished. ◻

## TURTLE **GEOMETRY**

Turtlestitch is based on turtle geometry, which was introduced in schools in the 1980s as a radical way to teach children maths and computing. However, it's not just for children. It also includes advanced mathematics and complex functions that can be used to produce fractals and Fibonacci patterns (see **Figure 3**).

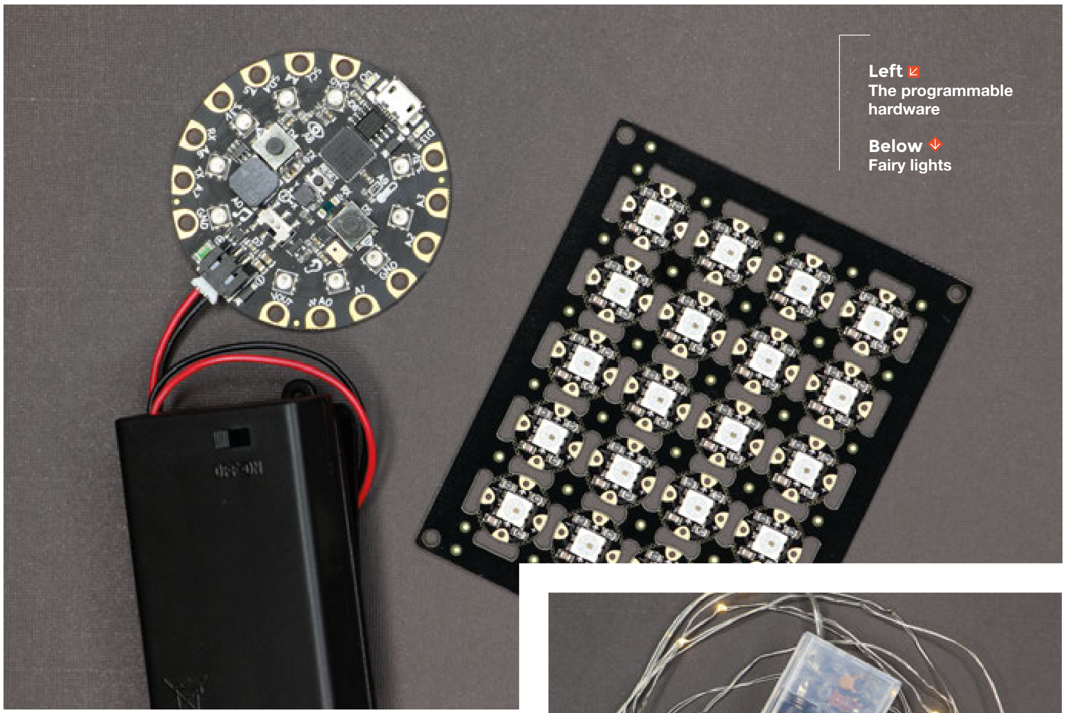


# Glowing LED skirt: with or without code

Create a magical-looking skirt using fairy lights or NeoPixels

**Soft, diaphanous tulle is perfect for diffusing LEDs!** Placing LEDs under layers of the delicate, translucent mesh creates a stunning visual effect that is almost magical. In this project, we'll modify the lining of a skirt so that a circuit can be easily installed for wearing, and removed for washing. Then we'll look at two options for LEDs: purchased fairy lights and a custom-made strand of NeoPixels. To drive our NeoPixels, we'll use one of our favourite Adafruit boards: the Circuit Playground Express.

Fairy lights are premade strands of lights with a battery pack attached. They usually take three AAA batteries and have about ten lights on the strand. Fairy lights are a quick and easy way to put lights into craft projects, garments, and costumes, and are a great no-soldering,



Left
The programmable hardware

Below
Fairy lights

no-code option for this project. Some fairy lights come with built-in animated effects, like twinkling or colour options. Keeping the circuit easily removable means you can swap out the light strand whenever you want to change things up.

For those wanting a more customisable project with a bit of soldering and code, we'll also make our own strand from scratch. Our custom strand will be controlled by a Circuit Playground Express programmed with CircuitPython. We'll use the on-board temperature sensor to slowly change the colour of the lights in our skirt according to ambient temperature. At a comfortable temperature inside, the skirt will glow a warm golden tone of yellow. Outside in the cold, the skirt will cool to a teal blue. Of course, you'll be able to change the colours to your liking, and tune the temperature parameters to your own environment. And as always, once built, you can reprogram the Circuit Playground Express whenever you get a new idea!

We love putting lights into wearable projects, but it's often not ideal to install them permanently. Always design your project to be cleanable, so that your hard work will last a lifetime. While some DIY electronic components are designed to be washable, the safest way to ensure that your project will not sustain damage during cleaning is to remove the circuit before washing. Snaps, hook-and-loop, magnets, and buttons are all great options for making a removable circuit. In our project, we are using lingerie strap guards. These are small pieces of ribbon with tiny snaps already installed on them, and can be purchased at sewing supply stores like **WAWAK.com**. We'll sew these onto the inside of the skirt to hold our circuit in place.

Human beings are almost always in motion, and even small movements can subject your circuit to wear and tear over time. Flexible silicone-coated stranded wire is a great choice for wearable projects like these, and is easier to manage over a large area than non-insulated conductive thread. However, you should still be careful with your completed circuit and take care not to snag wires when wearing your finished garment. In our project, we've kept the circuit on the front of the skirt, so that the wearer can sit down without crushing any of the electronic components we've added. Always keep the comfort of the wearer in mind and think about how the circuit will move when worn. Ensure that no hard edges will jab or scrape the wearer, and no electrical components will touch bare skin.

Tulle tip: if you purchase your skirt online, you may be underwhelmed when it arrives scrunched up in a tiny package. Don't worry, a little steam can turn that compressed mess into a voluminous vision! Hang the skirt up on a hanger, and use a hand-held steamer to steam out the wrinkles, separating each layer of →

## YOU'LL NEED

- **Layered tulle skirt with lining**
- **Battery-powered fairy lights**
- **Lingerie strap guards** (about ten per skirt)
- **Hand-sewing kit and/or sewing machine**
- **Seam ripper**
- **Knit fabric** (enough for a battery pocket)
- **Circuit Playground Express**
- **Flora RGB Smart NeoPixel version 2** (we used ten)
- **AAA ×3 battery pack with on/off switch**
- **Silicone-coated stranded wire**
- **Glue gun and hot glue**
- **Clear nail polish**
- **Soldering tools and supplies**

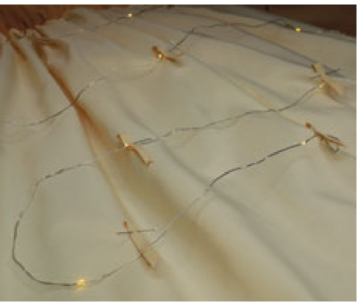
the skirt as you go. Do not use an iron for this, or you could melt the synthetic tulle. No steamer? No problem: hang the skirt in your bathroom while you take a hot shower, then gently tug the bottom of each layer to relax the wrinkles. Separating the layers of the skirt from each other will increase the diffusion effect and make the LEDs look spectacular!

## ADD THE STRAP GUARDS

Start by laying the skirt out on a table and folding back the top layers of tulle so that you can see the entire front of the lining. Lay the strand of lights over the skirt and decide where to attach the strap guards that will hold the circuit in place. For our two-metre strand, we used ten strap guards and spread them out for good coverage over the skirt. The battery pack and/or CPX will be located at the top of the skirt near the waistband. Because the skirt is narrower at the top and flares out at the bottom, you will probably have only two or three strap guards at the top, and four or five near the bottom. Pin the strap guards in place.

While the strap guards are pinned, you can carefully open each snap and lay the fairy lights in place to check the placement of the lights. Move the strap guards around until you are happy with the layout, but don't worry about being too precise. The lights will shift slightly during wear, so just make sure they can be nicely spread out over the front of the skirt.

Set the fairy lights aside and sew across each strap guard to attach them to the skirt. Strand lights are narrow, so sew your seam through the middle of each strap guard to hold the strand more securely. If you've got a sewing machine, you can make quick work of sewing these on. If not, hand-sew the strap guards in place using a strong back-stitch.



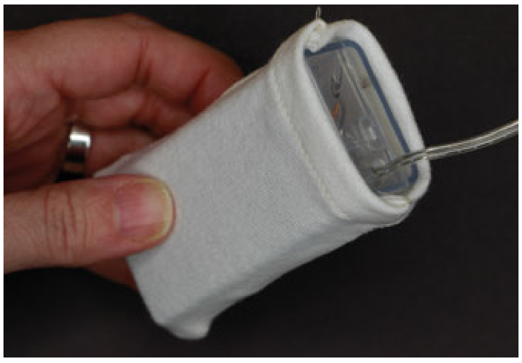
**Above** Pin, then sew, your lights in place

**Right** Finally, a pocket for a skirt

## MAKE A WIRE PASSTHROUGH

The battery pack will be located on the inside of the skirt just under the waistband, so we'll need to make a hole for the strand's wires to pass through. At one side, at about 5 cm below the waistband, locate the side seam of the skirt's lining. Use a seam ripper to open about 3 or 4 cm of the side seam. If you will be making the custom NeoPixel strand, make the hole big enough for a NeoPixel to fit through.

Once opened, reinforce the side seam above and below the opening by sewing over it for several centimetres. Then, dab a small amount of Fray Check on the raw edges of the opening so that the fabric does not unravel.



"Anchoring the battery pack to the waistband will keep it from swinging around"

## MAKE A BATTERY POCKET

To make a simple pocket for the battery pack, cut two pieces of stretch fabric about 2 cm bigger than the battery pack on all sides. On each piece, fold one of the short edges back about 1 cm and sew it down. These will make the opening of the pocket.

Next, lay the pieces together, right sides in, and sew the three raw edges together with about 1 cm seam allowance. Cut across the seam allowance of the two sewn corners (do not cut through your seam) and flip the pocket right side out. Check to make sure your battery pack fits inside, then remove the battery pack and set it aside.

Find a spot on the waistband for the pocket on the inside of the skirt, near the opening you made in the side seam. The battery pack is the heaviest part of the circuit, and anchoring it to the waistband will keep it from swinging around or pulling on the skirt during wear. Hand-sew the battery pack to the bottom edge of the waistband on the inside of the skirt using a whip-stitch or back-stitch.

## OPTION 1: FAIRY LIGHTS

Now let's make it light up! Place the whole circuit inside the skirt, and feed the end of the fairy lights through the hole you made at the side seam in the lining. The lights should be in front of the lining, and the battery pack should remain on the inside of the skirt.

For the first and last strap guard, make a loop in the fairy lights and snap the strap guard through the loop to keep the strand from sliding out at either end. For the others, simply snap the strap guards around the strand. Arrange your lights in the strap guards as you like them, and slip the battery pack into its pocket. Flip on the switch and see your skirt glow bright!

**Above**
Sewing the pocket in place

## OPTION 2: CUSTOM NEOPIXEL STRAND

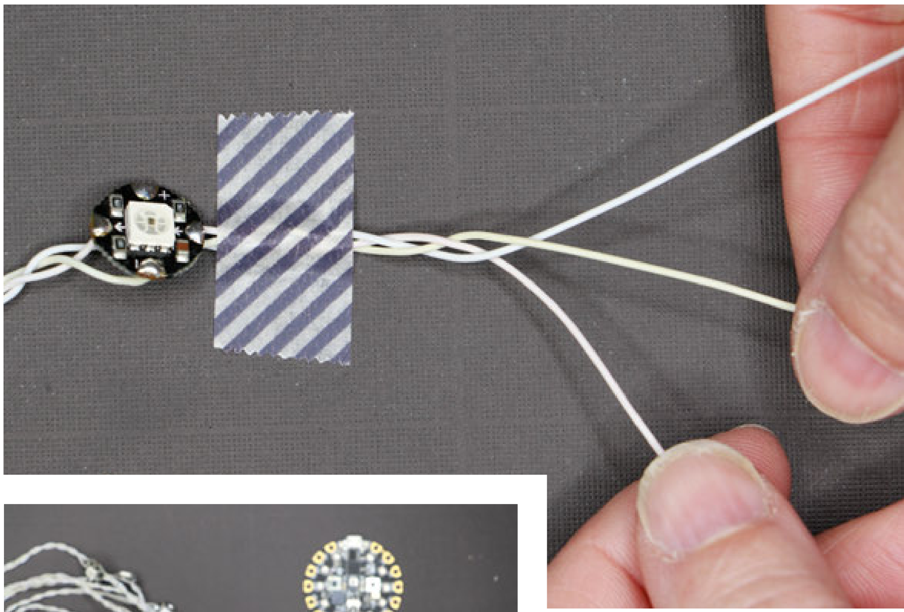
Tulle is delicate, and any sharp bits on our electronics could snag and damage the skirt. So we'll start by smoothing out the edges of the NeoPixels with an emery board or sandpaper.

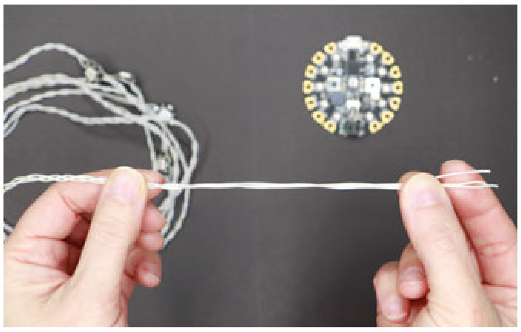
For the skirt shown, our NeoPixels are spaced out about 20 cm apart from each other on the strand. This matches the spacing in our fairy lights, making it easy to swap the strands in our skirt. Braiding the wires together keeps the circuit tidy but flexible. Using different-coloured wires is extremely helpful when braiding wires together, but dark colours may show under a light skirt. Try to use wires that match your skirt, and label wires as you work if you can't use different colours.

While working, pay attention to the direction of the arrows on each pixel – this is the direction that data flows through the pixel. All pixels in the strand must be oriented the same way, with the arrows all pointing away from the Circuit Playground Express. Check the arrow before soldering each pixel; it is very easy to accidentally turn the NeoPixels around while working.

Cut wires to length for your strand, adding 2 cm to compensate for length lost when braiding the wires. Begin soldering the NeoPixels together in a chain, feeding the wires through each pin hole from back to front. Start with the last pixel in the strand and work backwards to the first, connecting power to power, ground to ground, and data-out to data-in on each pixel. Each power and ground pin will have two wires – one from the previous pixel on the strand, and one going to the next pixel.

Between each pixel, braid the wires together gently to make a strand. Don't pull too tightly on the soldered points, the braid is just to keep the wires together in a bundle. It can be helpful to anchor the wires down to your work surface with light duty tape while you braid. After braiding, measure and trim your braid to the correct length before soldering to the next pixel.





**Above & Left**
Plaiting to keep everything neat

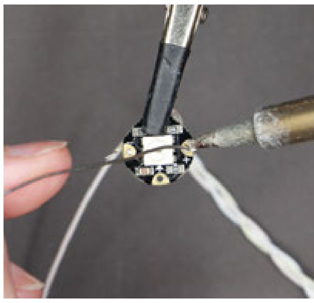
When you've got all your NeoPixels soldered and braided together, connect the first pixel in the strand to the Circuit Playground Express. Cut wires long enough to reach from the first pixel to the waistband. You may want to add additional length to allow for experimenting with different locations for the CPX. Braid the wires as before, but leave 6–7 cm unbraided. Solder power, ground, and data on the NeoPixels to Vout, GND, and A1 on the Circuit Playground Express, respectively.

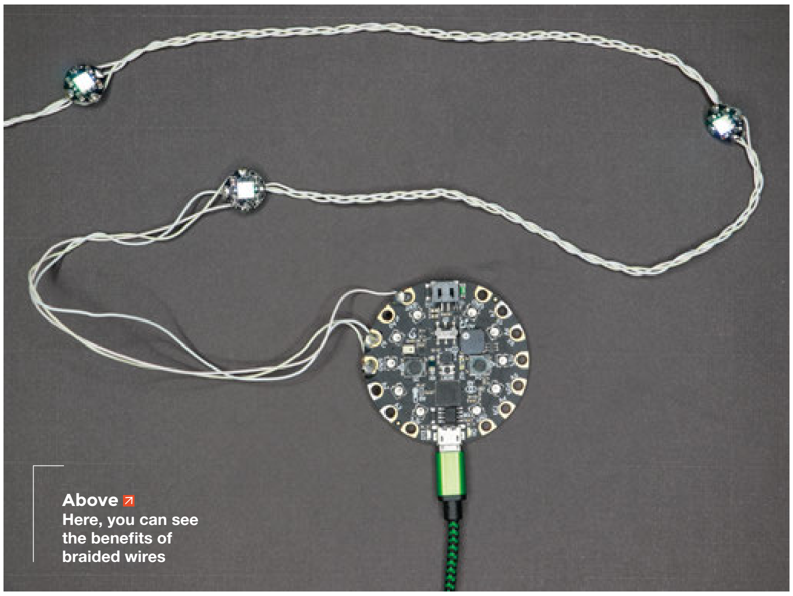
Note: stop here for a moment. Now that your circuit is complete, be aware that you have a very long, flexible circuit with exposed connections that could easily touch each other before we insulate them. We are about to connect the Circuit Playground Express to your computer and program it, which means that power will be flowing through the circuit. During this phase, it is extremely important that the exposed solder points do not touch each other, or they will short the circuit. In addition to damaging your circuit, this can be dangerous. To be safe, lay your circuit out in a line and tape it to your work surface before plugging it into your computer.

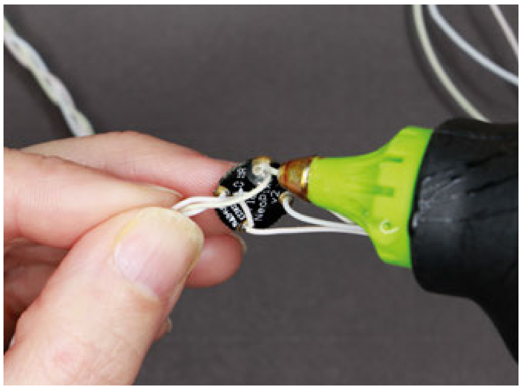
## PROGRAM THE CIRCUIT PLAYGROUND EXPRESS

Now we're ready to program the CPX and make our circuit light up. With your circuit secured to your →

**Below**
A drop of solder to keep everything together




**Above**
Here, you can see the benefits of braided wires


**Above**
Add a drop of hot insulation

work surface with no connections touching each other, plug the Circuit Playground Express into your computer with a micro USB cable.

Visit Adafruit's online guide at **hsmag.cc/EHawxq** to update your board with the latest version of CircuitPython and its libraries. If this is your first time working with a Circuit Playground Express or CircuitPython, the guide has excellent information on how to get started and tips for troubleshooting your work.

When plugged into your computer, the CPX will appear as a drive called CIRCUITPY. Find the **code.py** file on the drive, or create one if it doesn't exist yet. This is where we will write our code. CircuitPython code can be edited in any text editor, but we recommend Mu Editor, as it has helpful tools like a code checker and serial monitor for debugging. Open the file and erase any code already there, so we can start from scratch.

"**We've added a bit to the ambient room temperatures we expect,** to compensate for body heat"

## THE CODE

Our code (**skirt.py** at **hsmag.cc/book2-assets**) begins by importing the necessary libraries. Next, we create the pixels variable as a NeoPixel strand with ten pixels. We also set up the on-board thermistor, and create two variables called `lowTemp` and `highTemp`. Set these to the lowest and highest temperatures you want the skirt to react to. In our skirt, the board is placed close to the body, so we've added a bit to the ambient room temperatures we expect, to compensate for body heat.

```
import time
import adafruit_thermistor
import simpleio
import board
import neopixel

numpix = 10
pixels = neopixel.NeoPixel(board.A1, numpix,
brightness=.5)
thermistor = adafruit_thermistor.Thermistor(
    board.TEMPERATURE, 10000, 10000, 25, 3950)

lowTemp = 20
highTemp = 30
```

For our main loop, we ask the board to read the thermistor and name that value `temp`. For debugging, we'll print this temperature in the serial monitor.

```
while True:

    temp = thermistor.temperature
    print("Temperature is: C", (temp))
```

We also create a variable called `colorVal`, which will hold a colour value that we can use for our NeoPixels. To get that value, we are using `simpleio.map_range` to convert the incoming temperature to a number from 0 to 255. A temperature reading of 20 °C will equate to zero, and a temperature reading of 30 °C will equate to 255. Anything between 20 and 30 °C will be mapped to a proportional value somewhere between 0 and 255.

```
    colorVal = int(simpleio.map_range(temp,
lowTemp, highTemp, 0, 255))
```

Next, we use `colorVal` to create variables for the individual RGB values of our NeoPixels. This way, we can change them dynamically as the temperature changes. Red and blue will change inversely: the

NeoPixels will be redder when it's warm, and bluer when it's cold. A little bit of constant green will give us a nice warm yellow at the warm end, and a bright teal at the cool end.

Tune these colour values however you like to create the shades you want for your skirt. We print the colour values to the serial monitor for reference, which is helpful when changing with the colours.

```
    r = colorVal
    g = 100
    b = 255 - colorVal
    print("R: ", (r))
    print("G: ", (g))
    print("B: ", (b))
```

The last few lines fill the pixels with the colour values we've generated, and turn the strand on. A brief **`time.sleep`** at the end keeps everything running smoothly.

```
    pixels.fill((r, g, b))
    pixels.show()
    time.sleep(0.25)
```

Saving this code in your **code.py** file will update the code on the board and automatically load it. Did all your NeoPixels light up? Yay! If not, don't worry. Check the first pixel that doesn't light up and see if it is oriented correctly on the strand. Then check all your solder points for poor connections or shorts. When everything is lighting up properly, you're ready to move on!

## INSULATE THE STRAND

Once you've programmed your Circuit Playground Express and verified that everything is working, it's time to insulate the strand to provide strain relief and prevent short circuits. Heat up your glue gun and apply small drops of hot glue over the solder joins on each NeoPixel. Cover the whole solder pad and the beginning of each wire completely, so that the wire does not bend at the solder point. Paint the front of each NeoPixel with clear nail polish.

On the Circuit Playground Express, cover each of the three solder points with hot glue as well. Even though the connections are now reinforced, it's still a good idea to treat the circuit gently. Keep your circuit dry – skip this skirt on rainy days.

## INSTALL THE NEOPIXEL STRAND

As with the fairy lights, start by placing the circuit inside the skirt and feeding the NeoPixel strand through the opening in the side seam. Gently pull the strand through to the front of the lining and use the strap guards to hold the circuit in place. Adjust the placement of the pixels to get an even but organic distribution of lights.

To expose the thermistor on the Circuit Playground Express to the ambient temperature around the wearer, attach the Circuit Playground Express on the outside of the skirt. On our skirt, we placed it on the waistband at the side of the skirt, but you may need to experiment with placement to keep the CPX from reading body heat instead of room temperature.

Hand-sew the CPX to the waistband through four of the sew tabs for a secure hold. When it's time to launder your skirt or you want to change out the strand, use a seam ripper to remove these four stitches before removing the strand from the strap guards.

Plug the battery holder in and slip it into its pocket. Flip the switch to turn on your skirt and watch the colour adjust according to the ambient temperature of your environment!

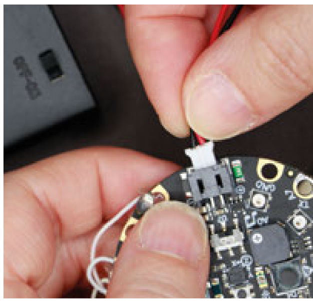
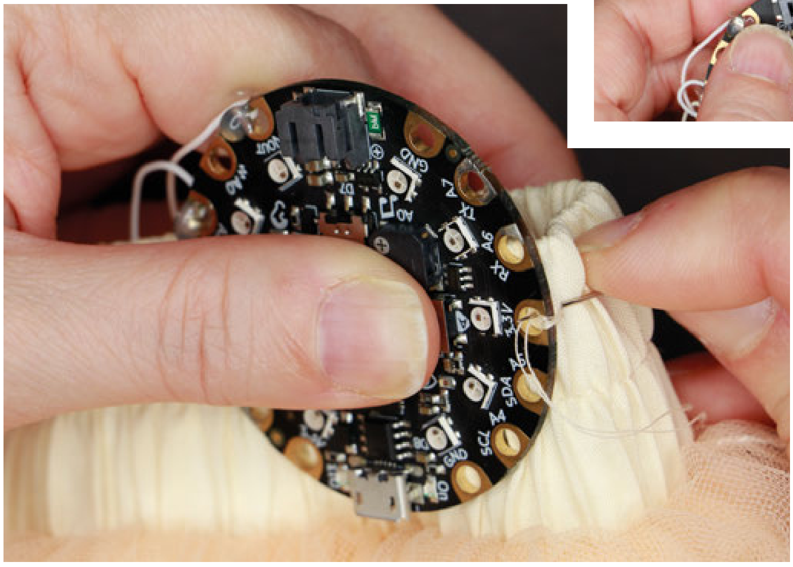
## WEAR IT YEAR ROUND

Frozen blue? Or sunshine yellow? This glowing skirt looks awesome at any temperature, and it's fun to watch it slowly change when you move to a different room or go outside. The change is gradual, which is a nice effect for such a large amount of light. And now that the circuit is built, the Circuit Playground Express is at your command! You can add animations and use the other built-in sensors to customise your build. Show us your Circuit Playground Express projects at **hackspace@raspberrypi.org**! □

**Below**
Placing the final pixel

**Middle**
Power on!

**Bottom**
Secure the Circuit Playground Express with a few stitches

TOP PROJECT

# Crazy Circuits dress

By Kitty Yeung @KittyArtPhysics

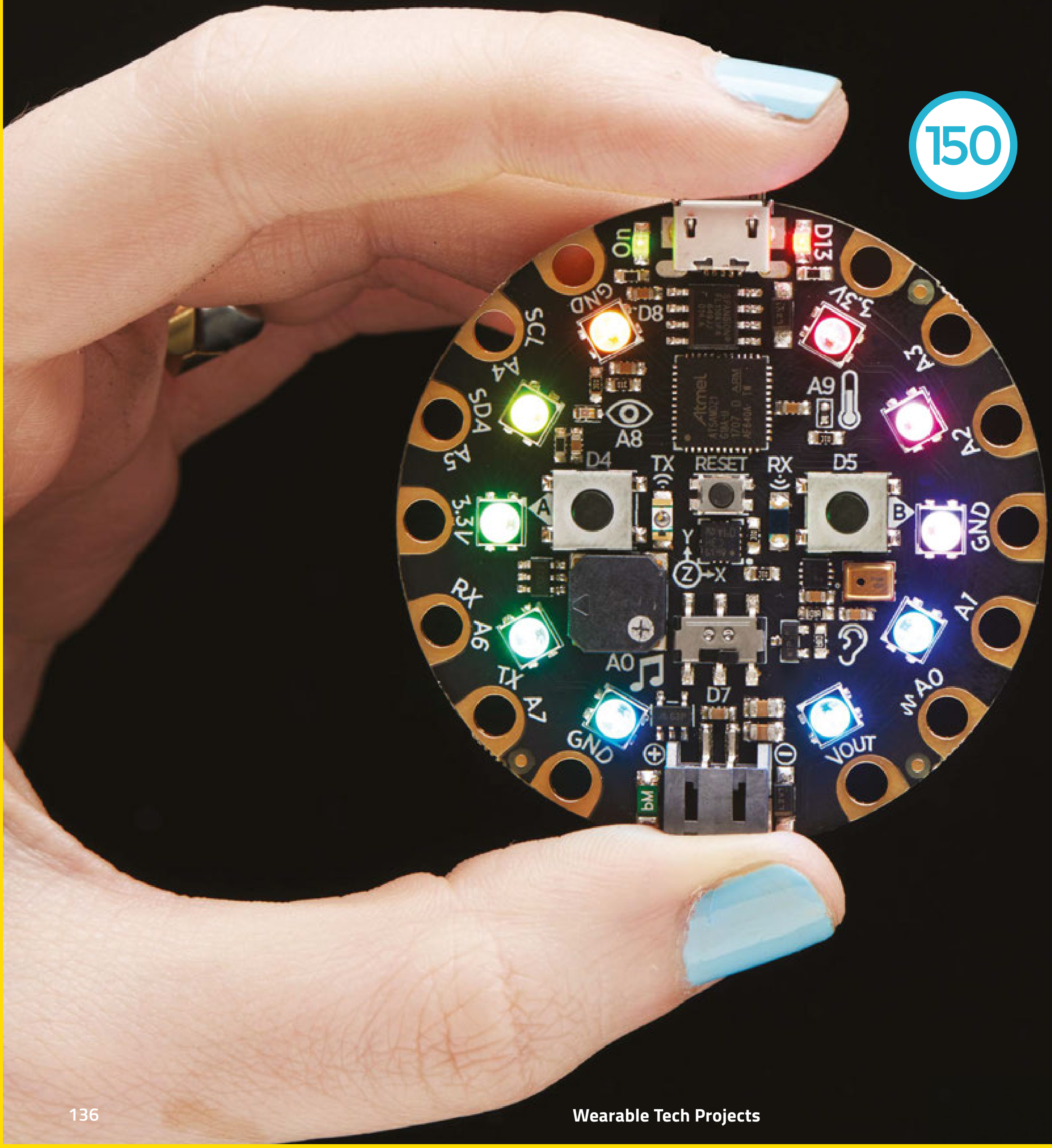
**This dress uses my painting of flowers as the fabric.** I cut the fabric according to the outlines of the flowers and stitched them into a dress. There are LEDs embedded underneath the flowers to light up like fireflies. The LEDs blink according to the wearer's heart rate, which is detected with an DFRobot EKG monitor. The microcontroller is an Arduino Nano, with Crazy Circuits' sewing breakout board. There are also additional LEDs under the tulles. The circuit is constructed with Crazy Circuits' sewing components and conductive tapes.

I recently published the construction process and story on Hackster.io: **hsmag.cc/EgjbeM**. □

**Right** We can see this working brilliantly at 127 BPM – disco heaven!

# GEAR

The tools you'll need, plus gear reviews and expert advice on how to create a product



150

# Cool tools for wearables

Consider adding these useful tools to your wearable electronics toolbox

**There's nothing like having the right tool for the job.** In the experimental world of wearables, sometimes the right tools come from unexpected places! In this deep dive, we'll look at our favourite tools for making wearable projects and why we love them.

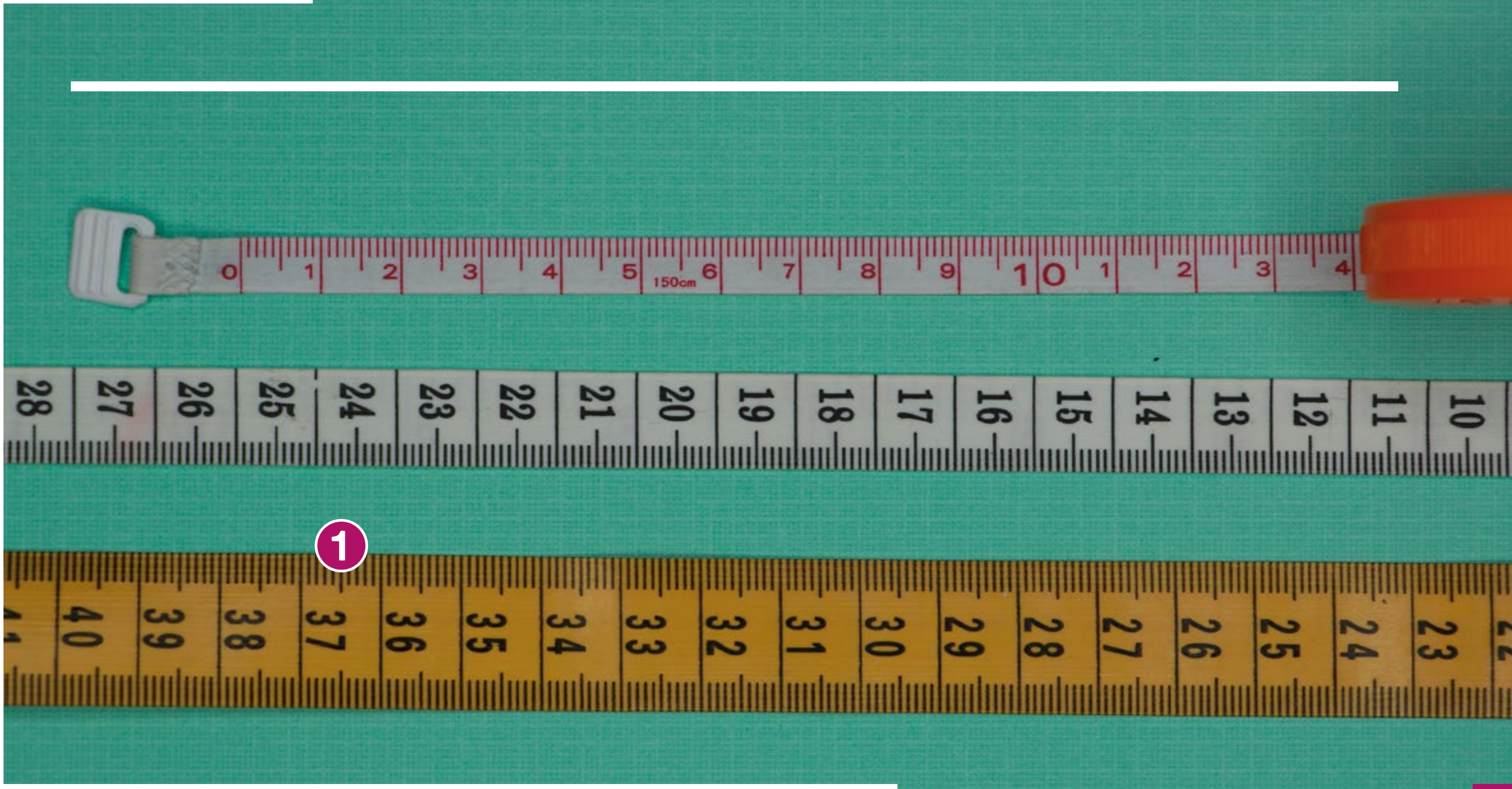
You don't need fancy tools to make great projects, but it's helpful to know what's out there. Some items on our list are necessary basics, and others are simply great to have on the occasions that you need them. Look for our handy icons to spot the tools we think are essential, multi-use, or special-purpose.

Wrapping technology around the human body is an exciting fusion of electronics and fashion, and brings new challenges to both worlds. Your wearable electronics toolbox will likely contain useful tools from both these disciplines, as well as some custom tools you'll make yourself. Working in a new field means you are on the cutting edge, and the right tool might not exist until you invent it!

E Essential

M Multi-use

S Special-purpose



# MEASURE TWICE

Make sure you take accurate measurements

**When building a project for the human body, measurements are the key to a perfect fit.** In addition to a basic ruler and yard stick, these measuring tools will help you get your dimensions right.

## 1 CLOTH TAPE MEASURE E

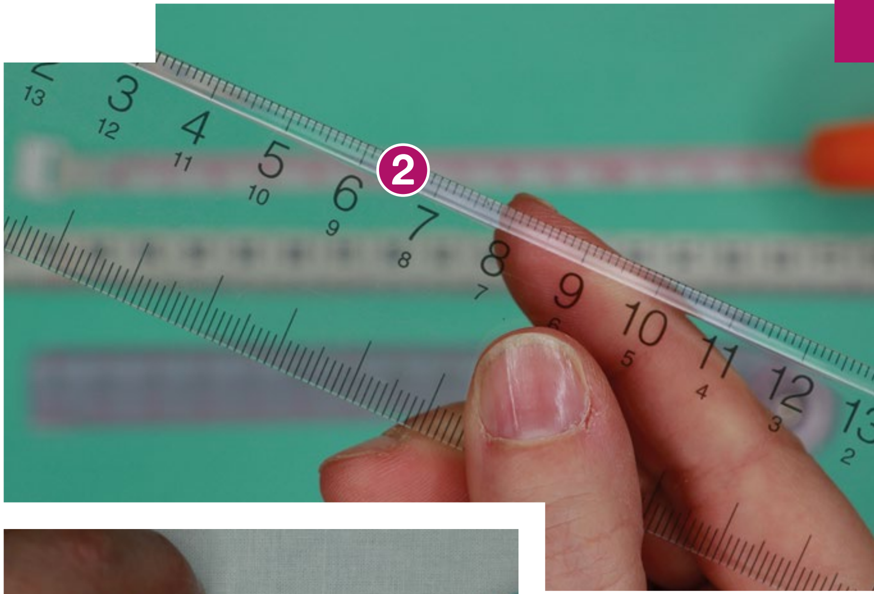
A flexible cloth measuring tape is essential for taking accurate body measurements. A retractable measuring tape won't turn into a tangled knot, and is handy for carrying with you to the fabric store or out on an inspiration hunt.

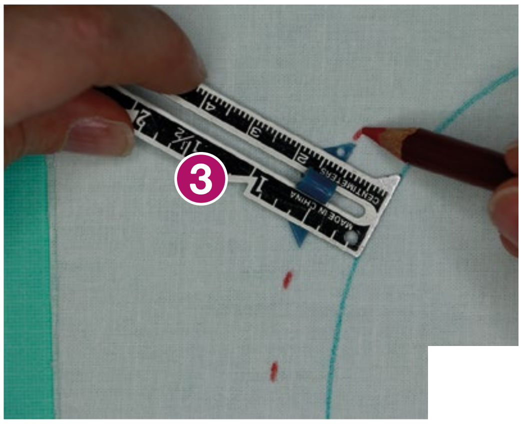
## 2 CLEAR RULER

A clear ruler makes it easy to take measurements off existing garments and objects. Clear quilting rulers have handy lines for marking seam allowances and tricky angles. Once you use a clear ruler for measurements, you'll wonder why anyone would ever measure with an opaque one.

## 3 SEWING GAUGE S

This super-useful tailor's tool takes the tedium out of marking seam allowances. You can also use a sewing gauge to quickly shift a line or transfer a measurement from one part of your project to another. The tool looks like a cross between a ruler and a calliper. To mark a seam allowance, set the slider to match your seam allowance, then hold the gauge perpendicular to your stitching line and slide along it, marking every few centimetres. Then, just connect the dots! →





**Above** A clear ruler is great, but never use one with a craft knife

**Left** A sewing gauge is ideal for marking seam allowances

# MAKE YOUR MARK

Different ways of marking your measurements

**Above** Choose the right marker for the material used

**To mark measurements on fabric, a sharp pencil or ball-point pen may suffice.** Coloured pencils are inexpensive, work well on woven fabrics, and you can use different colours for cutting lines, seam allowances, and other important markings. If you want your marks to be removable, consider these options…

## ① WATER SOLUBLE PENS

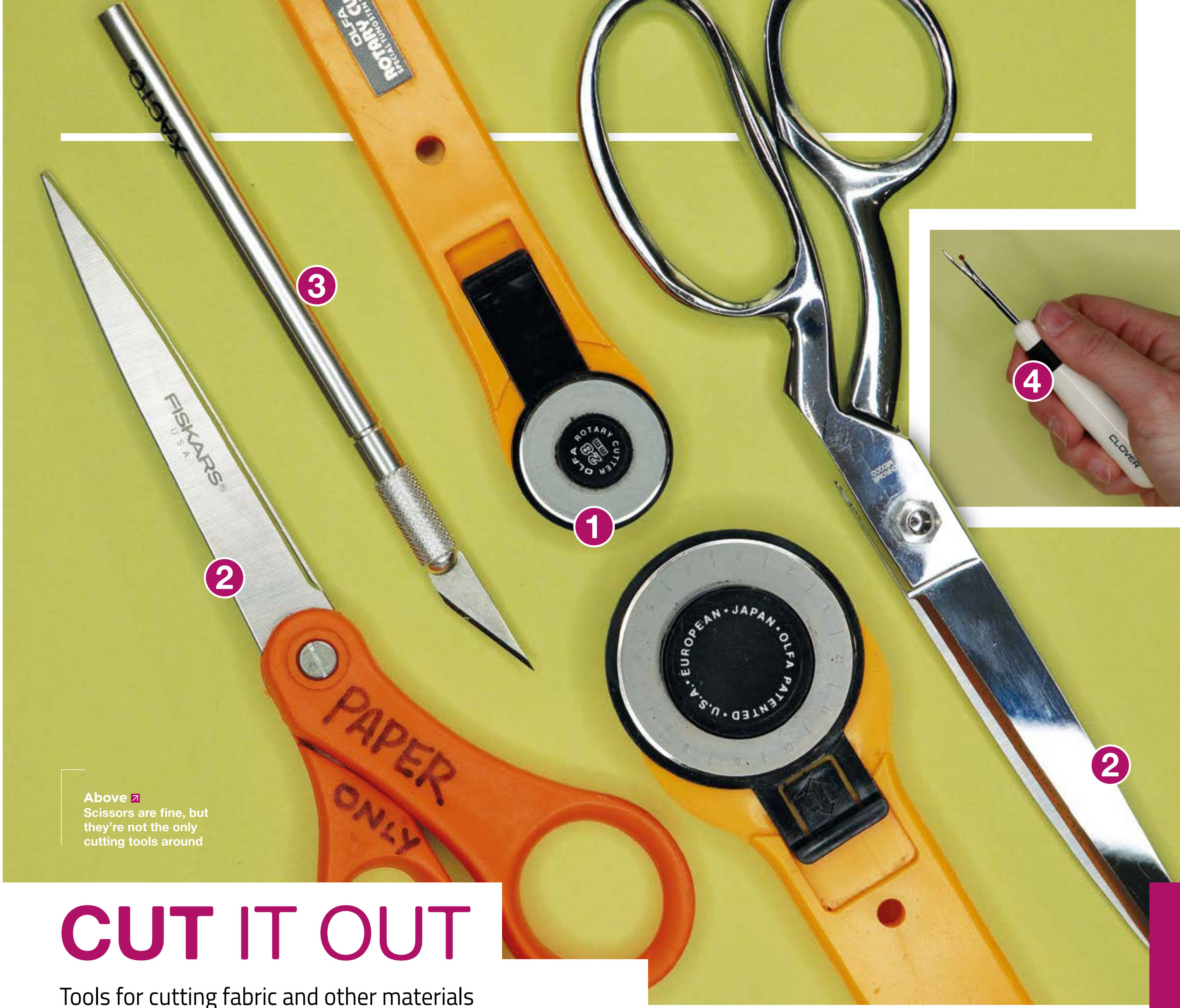
These pens usually contain blue or purple ink, which may or may not be visible against the colour of your fabric. Removability varies, and you'll want to test the pen on a scrap of your fabric to make sure it will remove completely with water. Let your test piece dry completely before pronouncing the test a success – in some cases the ink can reappear when dry. Do not use this type of pen on dry-clean-only fabrics, such as silk.

## ② TAILOR'S CHALK

Chalk is a great solution for making marks on fabric that may be sensitive to water, like wool, because in most cases the lines can be simply brushed away. Again, test on a scrap of fabric first. Tailor's chalk comes in palm-sized pieces or in pencil form, and can be easier to see on dark fabrics than water-soluble pens. Chalk is best for woven fabrics, but can be used with knit fabrics if care is taken not to stretch the fabric as you draw.

## ③ FRIXION PENS Ⓜ

These erasable pens by Pilot contain ink that becomes invisible when heat is applied. The pens feature an eraser tip for use on paper: rub the eraser over the ink until friction-generated heat renders the ink invisible. But they're great for fabric too! Instead of the on-board eraser, use an iron to apply heat to your lines when you no longer need them. The lines disappear before your eyes, and it really does feel a bit like magic! As always, you should test the ink on your fabric first, and also note that exposing the marks to cold temperatures can make them reappear.



Above
Scissors are fine, but they're not the only cutting tools around

# CUT IT OUT

Tools for cutting fabric and other materials

**Few tools are more perfect than a sharp pair of scissors.** To keep yours in top shape, keep one pair for use on fabric only and a separate pair for paper. Label them clearly so you don't get them mixed up, and your scissors will know what their jobs are!

## 1 ROTARY CUTTERS

If cutting a large piece, or many simple shapes, a rotary cutter can make your cuts more accurate and save time. Always use a rotary cutter on a cutting mat large enough to extend beyond the shape you're cutting.

## 2 SEWING SHEARS AND SNIPS S

You can cut fabric with regular craft scissors, as long as they are sharp, but if you will be working with fabric often, you may want to invest in a good pair of knife-edge dressmaker shears. Gingher is a classic brand known for making high-quality shears that can last for generations with proper sharpening. Dressmaker shears feature an offset handle designed for cutting fabric on a table. Resting their weight on the surface of the table makes for beautifully straight cuts. For clipping threads, a pair of palm-sized thread snips is efficient and easy to use.

## 3 CRAFT KNIFE AND CUTTING MAT E

While not great for fabric, a craft knife is perfect for cutting craft EVA foam, which is a very useful material for wearable projects. Using a cutting mat does more than just protect your work surface: it also keeps your blade sharp longer, and gives you more control for accurate and safe cutting.

## 4 SEAM RIPPER E

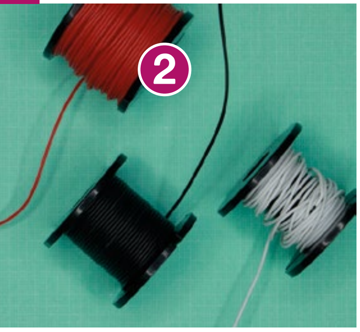
This tool is invaluable for adding electronics into garments. In many cases, you will want to route wires behind the lining of a garment, or into an existing pocket. Opening a seam will cause less damage than making holes in fabric that can unravel or fray. Use existing seams as much as possible in your projects, opening them with a seam ripper, and then hand-sewing them closed when you are finished. →

## SAFETY WARNING

Never use a plastic ruler as a straight edge for cutting with a craft knife. The soft plastic can catch the blade and steer the blade toward your hand. Always use a metal straight edge when cutting!

# CIRCUIT BUILDING

Tools to help you make sewn and soldered circuits



**Top** Essential tools and supplies for making soldered circuits

**Above** Silicone-coated stranded wire can be easily hidden

Sometimes a sewn circuit is the right choice, and other times you'll need to solder your connections. Here are our picks for being prepared for both!

## 1 SOLDERING SUPPLIES E

Basic soldering supplies include a soldering iron, solder, diagonal cutters, and wire strippers. A multimeter is essential for measuring voltage and troubleshooting connections within your circuit.

## 2 30 AWG SILICONE-COATED STRANDED WIRE S

This is the 'angel hair pasta' of electrical wire, and our absolute favourite wire for wearables. It is ultra-flexible, heat-resistant, and thin enough to thread in a darning needle. Silicone-coated stranded wire is available in larger gauges, but 30 AWG is our pick for wiring that can be hidden in almost anything.

## FUSIBLE INTERFACING

Apply iron-on fusible interfacing as a backing layer to keep the sewn circuit off of skin when worn. Like fabric, interfacing comes in both woven and knit varieties, and you'll want to match the weave of your fabric (woven interfacing for woven fabrics, knit for knits) to maintain the drape of your fabric.

## NAIL POLISH

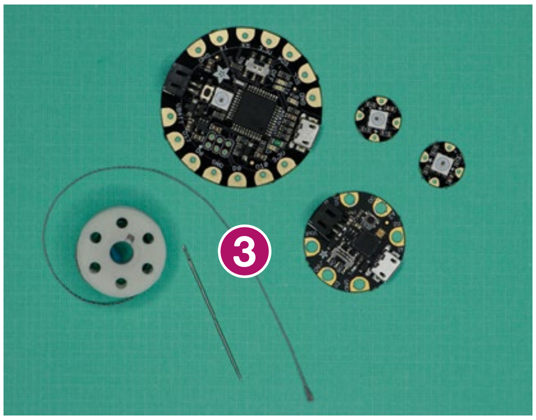
Use clear nail polish to paint the surface of your wearable components to insulate them and protect them from the elements. Paint knots tied in conductive thread to keep them from loosening.

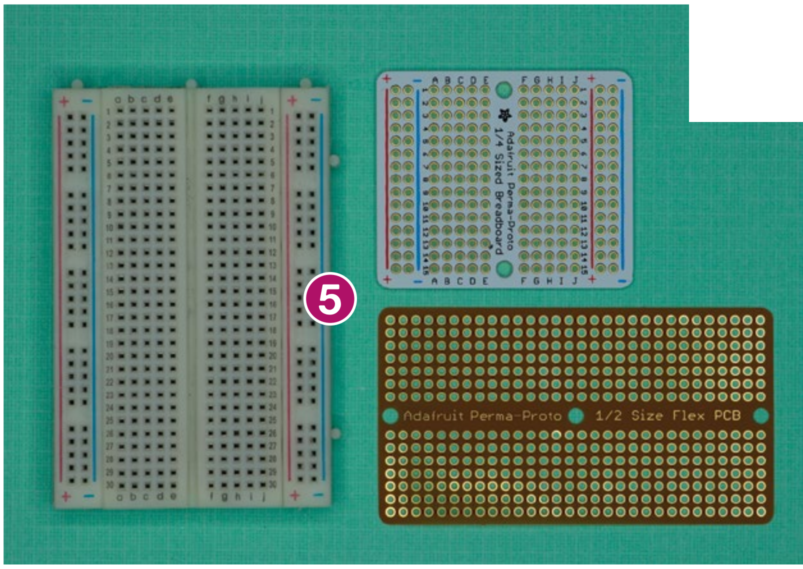
## 3 CONDUCTIVE THREAD E

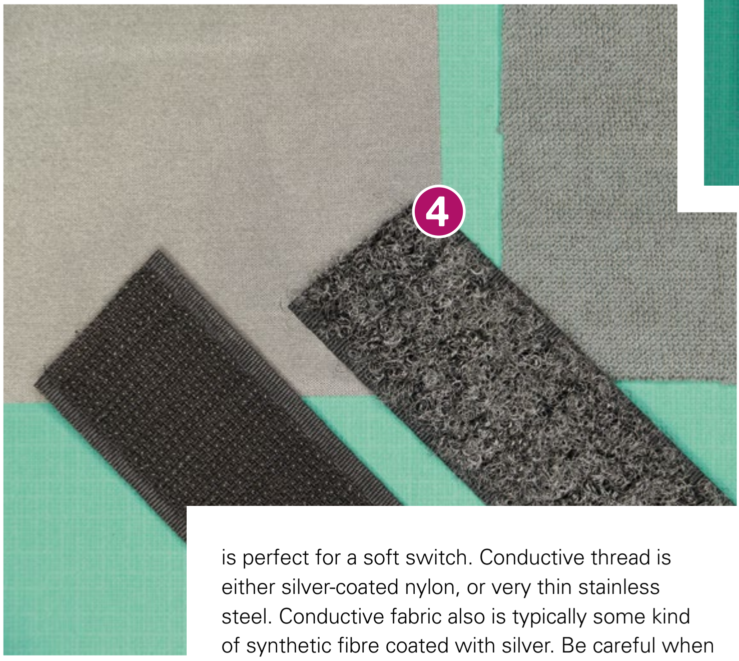
For sewn circuits, you'll need conductive thread to sew conductive traces between wearable electronics components like Adafruit's Gemma and Flora NeoPixels. Conductive thread can also be used to make sensors for capacitive touch inputs. Due to its resistance, conductive thread is not ideal for large circuits with very long runs of thread, and you will need to manually insulate your circuit to avoid short circuits.

## 4 SOFT CONDUCTIVE MATERIALS S

Conductive thread is just the beginning of soft conductive materials that are perfect for use in wearable projects. Conductive fabric can be used to make soft sensors; conductive hook-and-loop







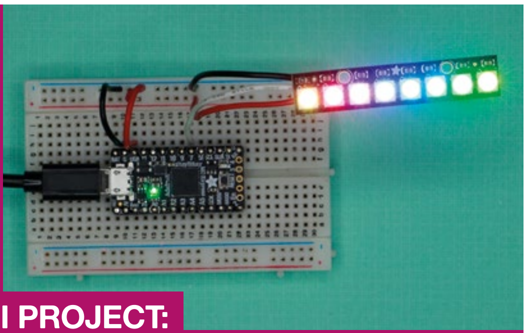
is perfect for a soft switch. Conductive thread is either silver-coated nylon, or very thin stainless steel. Conductive fabric also is typically some kind of synthetic fibre coated with silver. Be careful when using an iron on this fabric, and avoid steam.

## SNAPS, RIVETS, AND NAIL HEADS

Use metal snaps as conductive switches or sensors, or to make entire portions of your project conveniently removable. Rivets or eyelets can also be used as conductive sensors, or to make durable holes in your garment for wires to pass through. Snaps, rivets, and eyelets require a special tool, in the correct size, for installation.

## ⑤ PROTOTYPING

Breadboards and alligator clips are always useful for prototyping your circuit, and Perma-Proto boards by Adafruit make it easy to transfer your circuit into your wearable. Adafruit's Flexible Perma-Proto Board can even be cut into smaller pieces for a custom, flexible circuit board. Alligator clips can be tricky to use on tiny wires and wearable components, and sometimes the best solution is to make your own test components. →



## MINI PROJECT: DIY BREADBOARD-FRIENDLY NEOPIXELS

**All you need for this quick project is a small NeoPixel stick (Adafruit part 1426) or any other short run of NeoPixels, and three short lengths of hook-up wire.** Ours are 22 AWG solid-core wire, about 7 cm long. Use three different colours of wire, and solder one wire each to GND, DIN, and 5VDC on the NeoPixels. Strip about ½ cm of insulation of the wire ends, and you're done! Now you can build a quick breadboard prototype and write your code, then swap out the NeoPixels when you're ready to build your final project. Keep this handy in your electronics toolbox and you'll be ready to throw together a flashing breadboard circuit whenever inspiration strikes!

**Above** Numerous glues are available, suited to various materials

## MAKE IT **STICK**

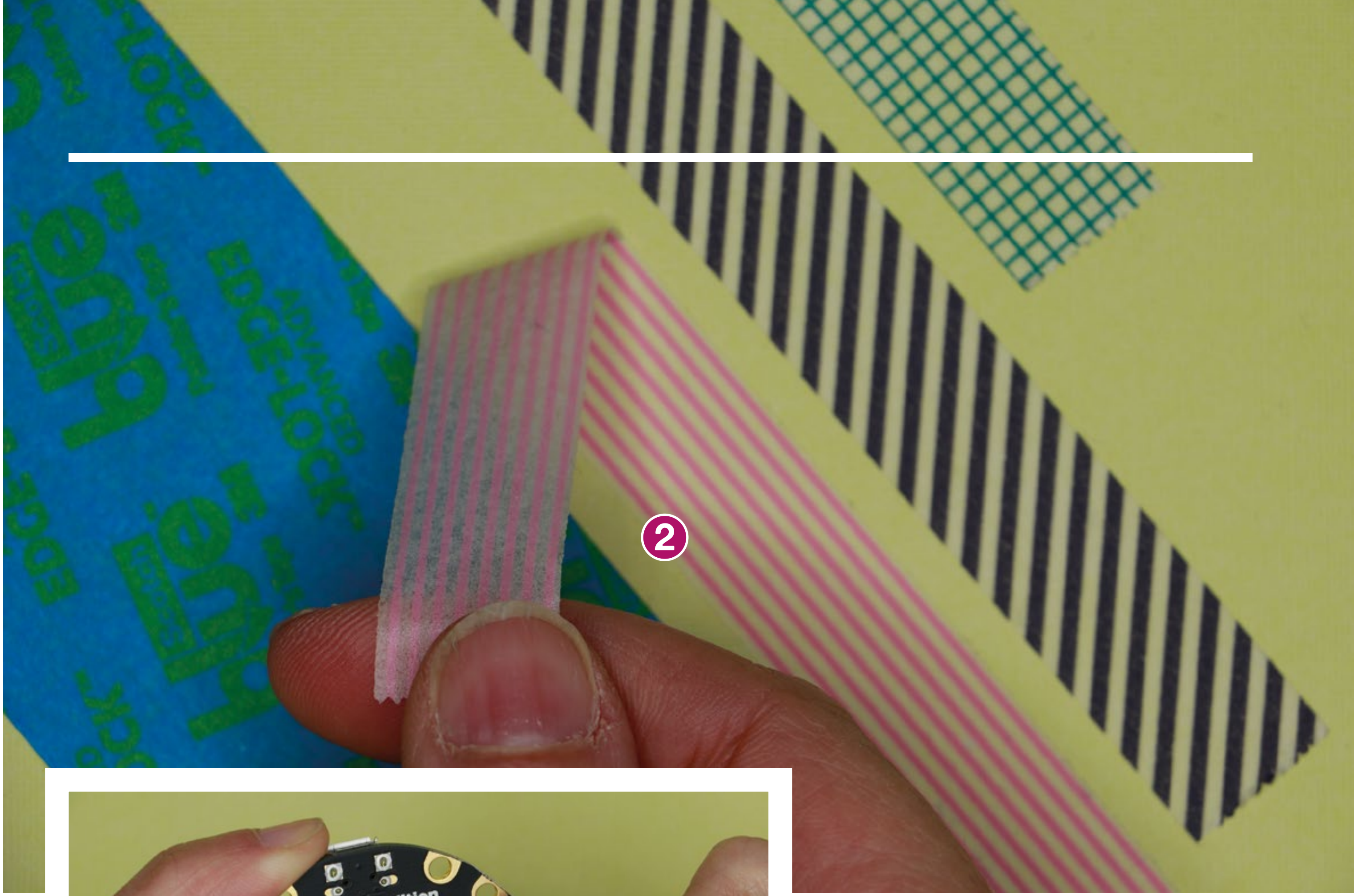
Sometimes the hardest part of a project is figuring out how to stick things together!

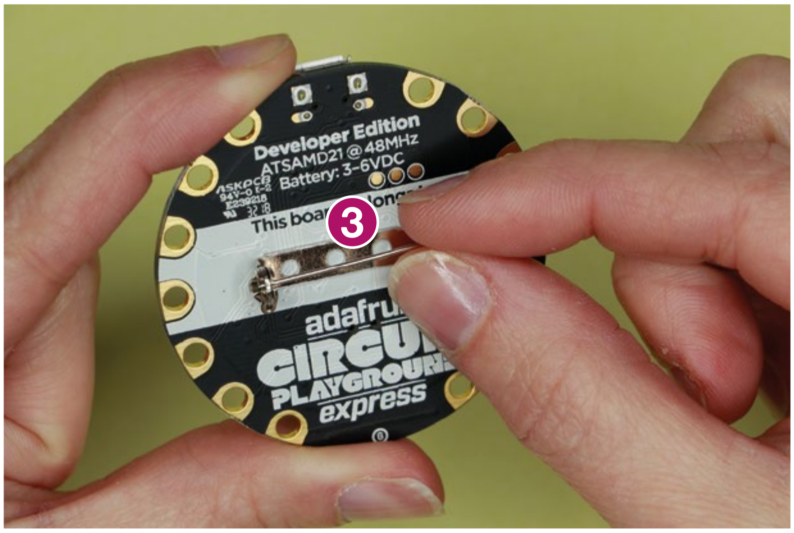
### 1 GLUES E M

Hot glue makes it into almost all of our wearable projects in one way or another. Use it for everything from attaching components to insulating a soldered connection. If you need to rework the soldered connection later, apply a drop of rubbing alcohol to the edge of the dried glue and peel it away. Cyanoacrylate glue is handy for keeping knots tied in conductive thread. And E6000 is a strong industrial adhesive that remains flexible when dry, and can be washed. Always wear a respirator when applying toxic glues like E6000, and let dry in a well-ventilated area or outside.

### FUSIBLE WEB S

Kind of like double-sided sticky tape for fabric, fusible web is a thin film of adhesive that is applied with heat. Sandwich it between two pieces of fabric,





iron with a press cloth, and the fabrics become fused together. Follow manufacturer's instructions for application and washing.

## HOOK-AND-LOOP FASTENER

Hook-and-loop fastener is endlessly useful in wearable projects. Just be careful to keep track of the hook (rough) side, as it can catch on things and damage soft fabrics. It's good to have both sticky-back and sew-on varieties on hand. The sticky-back type of hook-and-loop may work well on rigid surfaces like plastic and metal, but for a mechanical bond to fabric you'll want to sew it on.

## ② TAPES Ⓔ Ⓜ

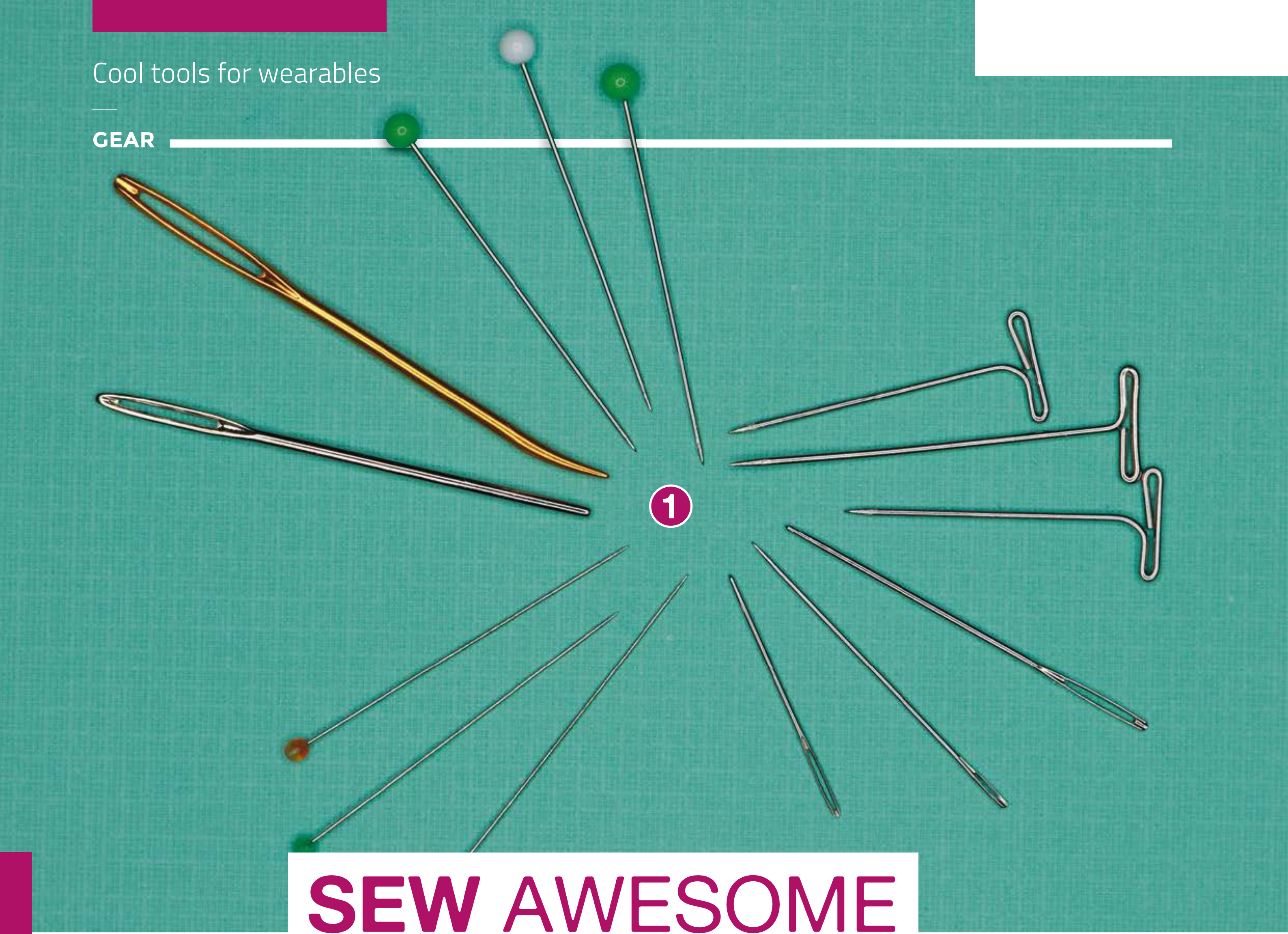
Masking tape, or blue painter's tape, is useful for creating patterns from rigid objects like helmets and shoes. Cover the item with tape, draw your pattern, remove the tape in one piece, and cut out. When you need a more permanent bond, double-sided foam tape is great for sticking microcontrollers and components to rigid surfaces like plastic. Electrical tape and Kapton tape are handy for insulating circuitry, and gaffer tape can add another layer of protection to a LiPo battery. Japanese washi paper tape in different colourful patterns makes it easy to label wires, and can be written on with a pen or pencil.

## ③ MAGNETS AND PIN-BACKS

To temporarily install electronic components in garments, consider name-tag magnets or pin-backs. Keeping your electronic components removable means you can wash your garment and keep your project forever! →

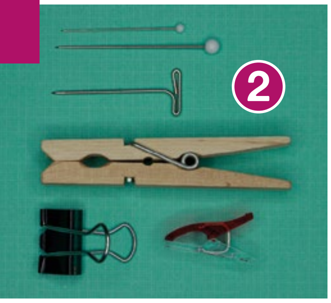
**Top** Tape comes in many types, from masking to electrical

**Above** Pin-backs make it easy to detach components



# SEW AWESOME

Use sewing tools to stitch fabric and hold things together



**Top** ↗ Pins and needles come in different shapes and sizes

**Above** When working with leather, use clips or pegs to hold it

A sewing supply store is a treasure-trove of tools and gadgets, with everything from point turners to button covers. Here are our top picks for wearables tools from the sewing store.

## HAND-SEWING SUPPLIES Ⓔ

Large-eye needles make it easy to thread regular and conductive thread. Blunt darning needles or yarn needles are perfect for feeding tiny silicone wires through casings and seams. A little beeswax applied to conductive thread can make sewing with it more manageable. A thimble is a necessity when pushing a needle through thick fabric. It takes time to get the hang of sewing with a thimble, but once mastered, your fingers will thank you.

## SEWING MACHINE Ⓢ

You don't need a fancy sewing machine to make amazing textile projects, you can make almost anything with just a straight stitch and a zig-zag. Think battery pack holders, arm bands, casings for NeoPixels, and more. If your sewing machine can make buttonholes, you've got a quick and easy way to make bound holes in fabric – perfect for wire passthroughs!

## ① PINS AND NEEDLES

There is a staggering array of different types of pins out there and, again, you can get by with simple straight pins for most projects. T-pins have a wide bar that makes them easy to grab; use them for pinning into styrofoam, cork, and foam board. Fork pins are good for tacking multiple layers of fabrics together, and there's an almost infinite list of uses for the good old safety pin. For knit fabrics, use ball-point pins to avoid snagging the fibres and creating runs in your fabric.

Likewise, when choosing needles for hand or machine sewing, use sharps or universal needles for woven fabrics, and ball-points for knits. Use a denim needle when working with heavyweight fabrics like denim or canvas, and use a special leather needle – with a wide, knife-edged tip – for sewing through leather. On your sewing machine, change your needle often, every time you start a new project. When your stitches have issues, a dull needle is often the culprit.

## ② CLIPS Ⓜ

When you can't pin, clip! When working with leather, any pin holes you make will be there forever. Instead of pins, use clips. Household clothes-pegs are easy

to use, gentle on the surface of the material, and you may already have them in your junk drawer. Binder clips have a strong grip, offer a wider gripping surface than clothes-pegs, and come in several sizes. Clover's Wonder Clips feature a flat underside for sliding across the surface of your sewing machine while you sew.

## ③ AWL AND LEATHER HOLE PUNCH Ⓢ

At some point, you'll need to make holes in your garments for passing wires. Cutting or punching holes into woven and knit fabrics causes fraying, runs, and damages the fabric. When possible, use an awl instead to work a hole between the fibres large enough to feed wires through. For leather, a leather hole punch with multiple sizes works well.

## ④ IRON Ⓔ

If you're currently sewing without an iron, prepare to up your game dramatically. Use the appropriate setting for your fabric, and keep the surface of your iron clean. Wrinkles in fabric look messy and make it impossible to cut your pattern accurately. Press your fabric before laying out your pattern, then press every seam as you sew twice. Press once over the seam as it was sewn, then open the seam flat and press the seam allowances open. This makes your seams strong and beautiful. A small craft iron is great for getting into tight nooks and crannies, or applying narrow strips of fusible interfacing over conductive thread traces.

## PRESS CLOTH AND TEFLON PRESSING SHEET

When pressing conductive fabrics and other synthetics, or when applying fusibles, use a press cloth to protect the fabric and keep your iron free of residue. A press cloth is just a small piece (think two sheets of A4 paper, side by side) of white or unbleached woven cotton, like muslin. Place the cloth over the fabric and press directly on top of the press cloth. When dirty or scorched, wash or discard. We love Teflon pressing sheets: they're reusable, smooth, translucent (so you can see what you're pressing!), and can last up to ten years under normal use.

## FRAY CHECK Ⓔ

This clear liquid is a must for unfinished fabric edges. A little goes a long way: dab some on along a raw edge to keep it from fraying, and place a drop on thread knots to keep them tied tight. If dried Fray Check in the bottle's nozzle closes it up over time, carefully poke it with a T-pin to clear it.

When two creative fields like electronics and textiles come together, innovation abounds. It's especially exciting for creators coming from one field to the other, discovering another rich world of tools to play with. Sometimes, just knowing a tool exists can bring new possibilities and spark ideas. And in the end, it'll be your idea that makes the project great, not your tools. So let your ideas fly – when you can sew and solder, you can make just about anything! □

**Left**
A leather hole punch with multiple sizes

**Above**
Craft irons enable you to get into the nooks and crannies

# LilyPad ProtoSnap Plus

$39.95 | sparkfun.com

By Ben Everard @ben_everard

**The LilyPad is based on the Arduino platform, but in a form that makes it easier to incorporate into wearable projects.** As well as the main microcontroller, there's a broad range of input and output add-ons under the LilyPad brand that are all designed to look good and to work with conductive thread.

The ProtoSnap Plus kit includes the LilyPad USB Plus microcontroller board (which includes six white LEDs in a bar graph, and one RGB LED), a light sensor, a button, a slide switch, eight sewable LEDs and a buzzer. All these parts come on a single PCB where they're connected via traces and can be used with no wiring or soldering. However, the individual parts can be snapped out so they can be rearranged before being sewn into a circuit.

Alongside this PCB, you get a micro USB cable, a 110 mAh LiPo battery, two bobbins of conductive thread, and a selection of needles. With this kit, you have everything you need to create your own wearable electronics. The LilyPad can even function as a battery charger (through the USB power supply), so it really does have everything you need in the kit.

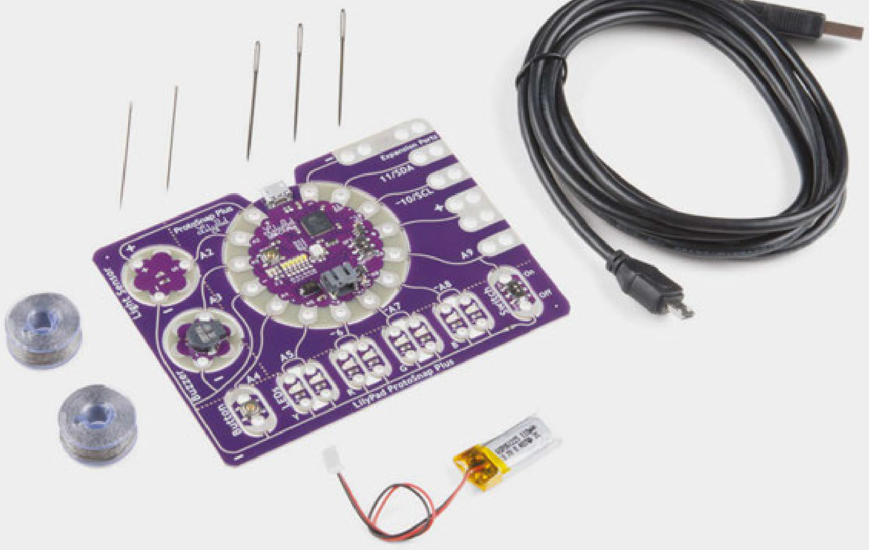
By shipping the various components on a single PCB, it makes it easy to get started – you don't have to fiddle around with connectors before testing out your code. Once you've got your code working, you can snap the components out and sew them into your wearable project.

The integrated on/off switch might sound like an unusual thing to praise on a bit of electronics, but it's useful for ensuring that you don't burn through the battery on your wearables before getting to the party (or wherever you're going).

The microcontroller gives you ten GPIOs, with seven of these taking analogue input and four with PWM output. This gives you enough pins to drive all the included hardware with an additional two available for other bits and pieces.

The LilyPad can be made to work with most 3.3 V electronics, but there's a series of peripheral boards that are designed to work well with sewable circuits, including accelerometers, temperature sensors, an XBee module, MP3 player, Bluetooth board, reed switch, protoboard, and plenty of flashing lights. They all come on the iconic purple PCBs and look great when sewn into fabric.

The LilyPad ProtoSnap Plus doesn't have an overwhelming number of features, but what it does have is well thought out and works well. It makes a fantastic introduction to sewable circuits. ◻



**Above** The 6ft (183 cm) USB cable makes it easy to plug in, even when sewn into an outfit

" By shipping the various components on a single PCB, **it makes it easy to get started** – you don't have to fiddle around with connectors before testing out your code "

**VERDICT**

A great introduction to wearable electronics, with everything you need to get started, but more forms of input would make more complex projects possible.

**9**/10

# Bearables Badge Kit

£14 | pimoroni.com

By Phil King @philking68

**At first glance, you might think Pimoroni's range of woodland-themed, multicoloured LED badges are just a cute wearable novelty item.** You'd be wrong. Not only can they be connected to and triggered by analogue sensors, using conductive thread, but they're designed to be hackable.

While the Bearables components are available individually, there are also two full kits, each featuring a badge, sensor, a generous 3 metres of conductive thread, and even some sewing needles. The only difference is that the Bear badge kit includes a tilt-switch motion sensor, while the Fox kit comes with a light sensor. Both badges and sensors have two metal hooks on the rear, around which you repeatedly loop the conductive metal thread used to connect them and sew them onto an item of clothing or a bag – you can't put them in the wash, though, as they don't like water.

Powered by a CR2032 coin cell that provides three to four days of fully lit use, the badges can be controlled manually using a small button on the edge to switch between various light patterns on the twelve coloured LEDs (blue, green, yellow, orange, red, and pink). Holding the button down for a few seconds puts the badge to sleep or, if a sensor is connected, selects trigger mode – in which case, depending on the sensor type, either motion or a lack of light will then activate the badge LEDs.

## PROGRAMMING THE BADGE

With no obvious connection ports, you may be wondering how you go about hacking the badge. The five tiny metal pads on the rear provide the means – via the I$^2$C bus – to program the badge's PIC16F1503 chip using a suitable board, such as a Raspberry Pi or micro:bit. To help you out, Pimoroni has created a Bearables Python library which allows you to switch the LED pattern, control individual LEDs to code your own patterns, use the button to trigger events, and read the voltage across the sewable hooks.

Those badge hooks can be used to read raw ADC values (0–255) from pretty much any analogue sensor, so you could attach different ones such as a sound sensor. You could also use the hooks to read pins that are pulled high or low on attached microprocessors, opening up all sorts of possibilities for using external data sources to trigger badge patterns.

**Above** Each badge features twelve LEDs of various single colours that can be lit in twelve preset patterns or your own custom creations

**Left** The badge and sensor may be sewn onto clothing using the supplied conductive thread

**VERDICT**

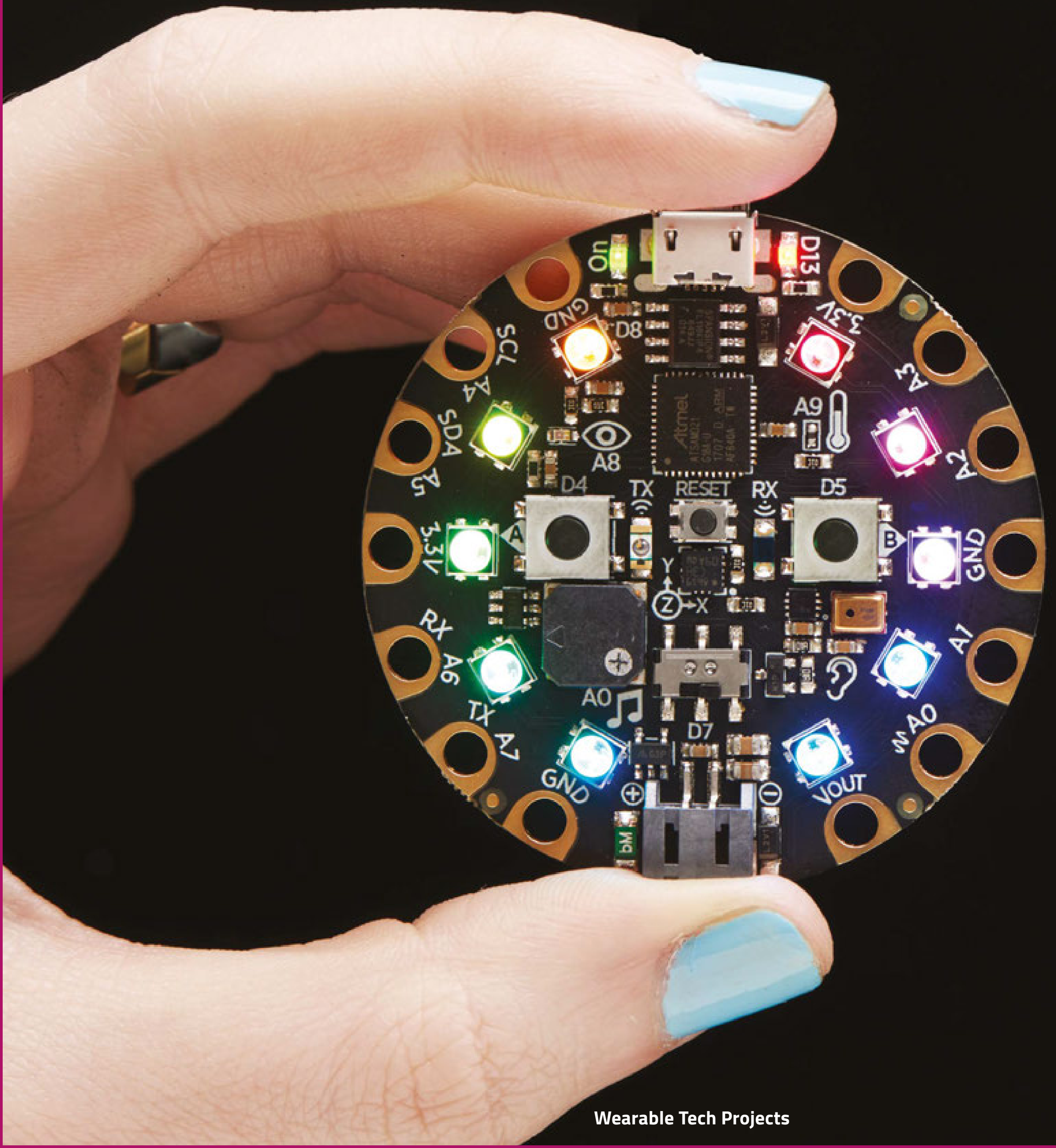
Great value for money, these cute LED badges are very hackable, offering a host of possibilities.

**9**/10

# Circuit Playground Express

**$24.95 / £23.70** | **adafruit.com** (US) / **pimoroni.com** (UK)

By **Ben Everard** ben_everard



**The Circuit Playground Express (CPX) is a programmable microcontroller board that makes it really easy to get started.** Plug it into a computer with a micro USB cable and you're ready to start coding.

The UF2 firmware takes software into the board in two ways: either you can press the reset button to enter programming mode and copy UF2 files into a new drive that will appear on your machine, or you can upload directly from the Arduino IDE. For beginners the first method will be easier, as you can generate these UF2 files from either the web-based MakeCode block-based editor or from a CircuitPython development environment.

On some versions of Windows, you'll need to install drivers, but on macOS, Linux, and Windows 10, you don't need to install anything if you use MakeCode – just plug your board into a USB port, then point your browser to **makecode.adafruit.com** to start your first project.

## BUILT-IN FEATURES

From here you can take advantage of the wide range of hardware that's packed onto the board. For output, there are ten NeoPixels and a speaker (not just a buzzer). For input, there are two buttons, a slide switch, an accelerometer, a temperature sensor, a microphone, and a light sensor. With all this, you can put together some pretty fancy projects straight away. For our first project, we made an indicator for cyclists. It uses the large holes on the CPX to attach it to the back of a cycling glove, then uses the accelerometer to recognise when the hand is held out indicating a turn and flashes the NeoPixels orange. This took 18 lines of dragged-and-dropped code in MakeCode (most of these were to ensure that there was a stable reading from the accelerometer) and no additional hardware. While hardly a complex project, it shows that you can build useful projects quickly with no additional hardware or software.

There's nothing on the Circuit Playground Express that's fundamentally unique – you can get the same sensors and outputs to attach to virtually any microcontroller. What makes this board special is the way it's brought together into a single package. It's hard to think of any project where you'll need all the features available, but given the £25 price tag, it's easy to justify the cost even if you only need one or two of the extra input or output options. By bringing them all onto the main board, there's no extra setup or wiring, and it's all supported by the software without having to add any libraries, which again makes it thankfully easy to get started.

The one obvious thing missing from the device is any form of networking. It does have a built-in infrared receiver and transmitter, I$^2$C, and UART, but no WiFi or Bluetooth. The other major limitation of the board is that it's not breadboard-friendly – it's far more suited to crocodile clips or banana plugs.

## MANY, MANY POSSIBILITIES

Whether or not these are limitations depends a lot on the sort of projects you're working on. The CPX isn't really suitable for Internet of Things-type applications. It's also not going to work well as a controller for building complex circuits – having just eight GPIOs limits the amount of hardware you can connect. Anyway, there are lots of microcontroller boards far more suited to these uses. However, the CPX makes it fantastically easy to get started with embedded and physical computing projects. You can build on the integrated hardware with the eight GPIOs. Seven of these can detect capacitive touch input, so to add more user input to your project, you just need a few crocodile clips and leads. As all the GPIOs can read analogue input, it's also trivial to add input from any device that gives a varying voltage as its output. There are also five PWM output pins for driving LEDs at different brightnesses. The MakeCode platform is a code repository as well as an IDE, and there's a set of tutorials from Adafruit to help you get started with the platform and the hardware.

" **The CPX is well suited to people getting their first microcontroller,** either buying one themselves for fun or as part of a taught course "

The CPX is well suited to people getting their first microcontroller, either buying one themselves for fun or as part of a taught course. For this purpose, it's genuinely hard to fault the CPX. It's easy to learn, with no (or minimal) software to install, yet at the same time allows you to use more advanced languages if you've got the knowledge and experience. It packs a fantastic range of input and output options onto the board, which means that you can dive right into some more interesting projects without getting immediately bogged down in attaching extra hardware. It's also easy to start building simple circuits, using crocodile clips.

Perhaps the ultimate test of any bit of hobbyist kit is whether or not it sparks excitement. For us, the CPX gave us a childlike sense of glee, because it makes so much so easy. This is great for beginners and any hobbyists who like to quickly prototype projects.

## VERDICT

**The best microcontroller board for introducing people to physical computing projects.**

**10**/10

# TinyLily Mini by TinyCircuits

A wearable microcontroller, suitable for dolls

**TINYCIRCUITS** ◆ **$9.95** | **tinycircuits.com**

By **Sophy Wong** sophywong

**Below** ◆
Small enough for almost any project

**Need a microcontroller for a tiny wearable project?** Break out your magnifying glass! The TinyLily Mini by TinyCircuits is an ultra-compact Arduino-compatible microcontroller designed for e-textiles. But don't be fooled by its size – with an Atmel ATmega328P on board (the same processor as Arduino Uno and LilyPad), the TinyLily Mini can drive complex projects and be hidden in the smallest wearables.

At one twelfth the size of a standard Arduino Uno, the TinyLily Mini is smaller than a UK 1p (or US dime) coin or a CR2032 battery. In fact it's so small, it's easy to lose – so keep it inside its plastic bag until you're ready to sew it to something. How could a microcontroller the size of a breath mint possibly compete with a full-size Arduino? We put the TinyLily Mini through its paces to find out if this tiny board measures up.

## WHAT YOU GET

The TinyLily Mini is available on its own for $9.95, or as part of a kit. We opted for the Starter Kit ($39.95) which comes with two TinyLily Mini processor boards, a USB adapter for programming, a battery connector, six sewable LEDs, and one incredibly tiny sewable button. That's more than enough bits and pieces to create some interesting wearables, and more components are available on the TinyCircuits website. The kit also comes with a quick-start guide, which is very handy for programming the TinyLily Mini for the first time.

The kit does not come with a battery, and it's worth noting that the included battery connector uses a JST-SH connector, which is smaller than the standard JST-PH connector on many LiPo batteries. If your LiPo batteries came from SparkFun or Adafruit, chances are they will not fit the TinyLily battery connector, and you'll want to purchase a new battery from TinyCircuits.

Although it doesn't come with the starter kit, TinyCircuits also makes a motor shield for TinyLily, which can drive one DC brushed motor. And, because the TinyLily Mini is Arduino-compatible, you can use it with other Arduino-compatible wearable components, like Adafruit's popular NeoPixels and SparkFun's LilyPad modules.

## YOU PROBABLY ALREADY KNOW HOW TO USE IT

Aside from its size, the TinyLily Mini shares the same form factor as many other microcontrollers designed for e-textiles. Its familiar circular shape and sew tabs are clearly inspired by Leah Buechley's iconic LilyPad Arduino. If you've used other wearable boards like LilyPad or Adafruit's Flora, you'll be up and running with TinyLily in no time. Programming with the Arduino IDE is pretty standard, just select 'Arduino Pro or Pro Mini (3.3V, 8MHz) w/ATmega328' in the boards menu, and upload your code.

There are eight sewable I/O pins on the processor board; four are digital pins and four are analogue/digital pins. Two pairs of sewable power and ground tabs are conveniently located at the top and bottom of the board. And if that's not enough, there are more solderable pins on the back of the board: three more I/O pins, reset, VCC, and GND. That's an impressive amount of capability for such a compact design, and the solder-friendly gold-plated tabs mean this board is useful for more than just e-textile applications.

## FORM VERSUS FUNCTION

To achieve its tiny size, some convenient features that are standard on other wearable boards were offloaded to separate modules, such as a JST battery connector and a USB adapter for programming. If you want a reset button or on/off switch (highly recommended for any wearable project), you'll have to add these to your circuit on your own. This is fine for many wearable projects, where you'll want to customise the locations of these components anyway.

Keep in mind that although the board itself is tiny, adding the battery connector increases its size, and the connection is a little awkward. The battery connector plugs into the board face-down, which means the connector doesn't lie flat against the surface of the project. This is more of an annoyance than a problem, but you'll want to plan around it in your wearable designs, as the connector has corners that could scratch fabric or skin. TinyLily Mini has no power regulator, so be sure your power source is within the 2.7 V – 5.5 V operating voltage. Trading convenience for customisation means this board is less beginner-friendly than other wearable boards, and more suited to makers who are already comfortable with e-textiles and Arduino.

The sew tabs on the TinyLily Mini are definitely small, but they're easy to sew to, and require fewer stitches to fill with conductive thread for a good connection. The sew tabs are labelled on the back of the board, likely due to lack of space on the front, and this makes it a bit tricky to be sure that you are sewing to the correct pin. It really helps to use a very small dab of hot glue to hold these tiny components in place while you sew. The board is washable; however, you'll want to take care not to bend or snag the male five-pin connector when washing.

## SMALL, MODULAR, AND POWERFUL

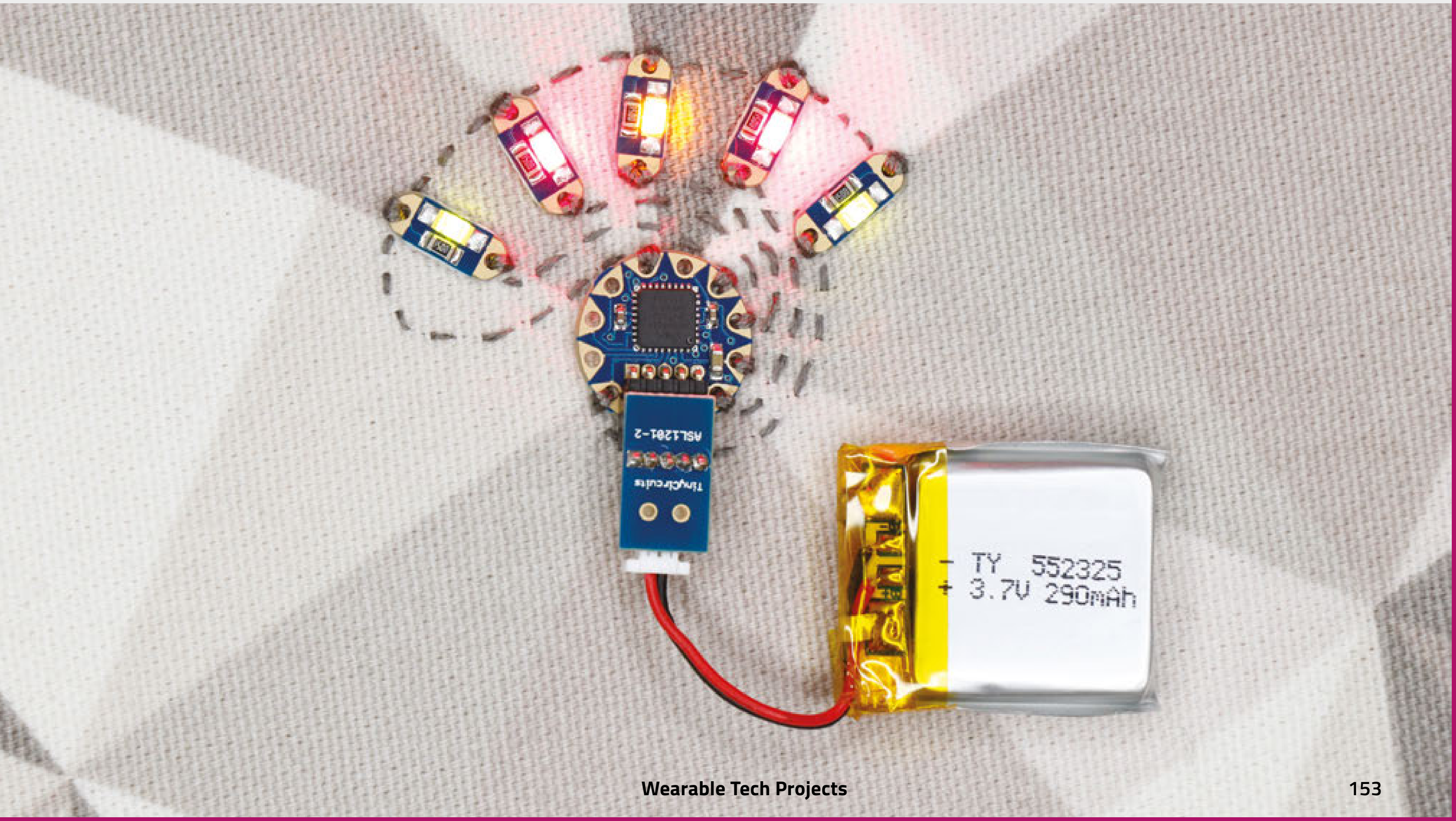
While other hardware companies are adding more features to their microcontrollers, TinyCircuits has stripped features away to create a board that is efficient and effective. Removing non-essential components from the main board results in a system that is customisable for every project. Small components like those in the TinyLily family make it easy to distribute your circuit invisibly throughout your wearable as you see fit, without having to sacrifice function. While it's not the board we'd recommend for your first project, the trade-off in convenience is easily worth it for intermediate and advanced makers seeking to fit electronic circuits invisibly into their wearables. □

**Below**
The sewable LEDs come in red, green, and amber

### VERDICT

**Like a magical blue police box, the TinyLily Mini microcontroller packs massive functionality into a mind-blowingly small footprint.**

**8**/10



# Pimoroni NFC nail stickers

Light up your manicure with NFC

**PIMORONI** ◆ £5 | shop.pimoroni.com

By **Sophy Wong** @sophywong

**Part nail art, part biohack, these NFC nail stickers from Pimoroni are awesomely cyberpunk.** Each sticker is a tiny flexible circuit that contains an NFC antenna and a tiny LED; when placed near an NFC signal, the LED lights up. This ultra-compact package, with no battery or exposed wiring, is a perfect fit for nail art!

NFC, or near-field communication, is a subset of RFID technology. Electronic devices with NFC capabilities can communicate with each other wirelessly across a distance of about 4 cm or less. This technology is used for contactless payment systems like Apple Pay and Google Pay, and for ID badge entry systems. Many smartphones now come with NFC capabilities, and makers have been using NFC components with Arduino to build creative projects, like keyless locks and password vaults. NFC technology is spreading quickly as an easy and secure way to pass data from one device to another, but power can be transmitted as well. That's how these LED nail stickers work: power comes from the electromagnetic field generated by an NFC signal.

**Below** ◆
**The nail stickers can sit on top of polish to give a coloured, light-up nail**

## OUR EXPERIENCE

For £5, you get one hand's worth: five nail stickers with one LED on each. All five stickers are the same size: 11 mm long by 9 mm wide. This size should fit well on most adult thumbnails, but using these on smaller nails will be a challenge. They'll fit easily on larger acrylic nails, which is how we've seen similar products styled. Applying the stickers was fairly easy, though it took a few tries to get a smooth application. In our tests, we secured the stickers with several coats of clear gel polish, which held nicely for about five days. Without a coating, the adhesive held for about 24 hours, before peeling up around the edges. A few drops of acetone easily removed the sticker, which is not reusable.

When lit, the LEDs are bright for their tiny size, but they definitely look best in low-light situations. Placing the stickers near an NFC device, like a smartphone with NFC enabled, makes the LED shine nicely. However, if you don't have an NFC smartphone, you'll have to find NFC fields out in the world to wave your hand in front of. This means your nails may only glow when you're paying for something at a store, or badging in at your office. If you're out for a day of shopping, this could actually be pretty often – you can bask in the glow of your manicure whenever you check out at a 'tap to pay' machine.

## GOING FURTHER

Of course, we're makers, so naturally we decided to build our own NFC station. For this, Pimoroni suggests using the stickers with an RFID/NFC shield for Arduino (£40) from Adafruit. We paired the shield with an Arduino Uno. With some handy code from Adafruit's quick-start guide, we had our own NFC reader up and running: passing the stickers within 2.5 cm of the shield made the LEDs glow satisfyingly bright. The shield is about 54 mm by 118 mm, and is not designed for wearable applications, so if you're thinking of cosplaying with these nail stickers, you'll have to cleverly build the Arduino and shield into some kind of hand-held prop, like a book. The LEDs need to be very close to the antenna on the shield to glow, and too much material over the antenna can obstruct the field, making this a challenging application.

It's important to know that these NFC stickers cannot be written to or read – they simply light up when placed inside an NFC field. This may feel like a bit of a dead end, but they could still be useful as wireless lights in projects. If nail art isn't your thing, you could stick these to objects, or even embed them in resin. As long as the sticker can be placed near an NFC signal, the LED should light up, and the effect is magical.

## NAILED IT

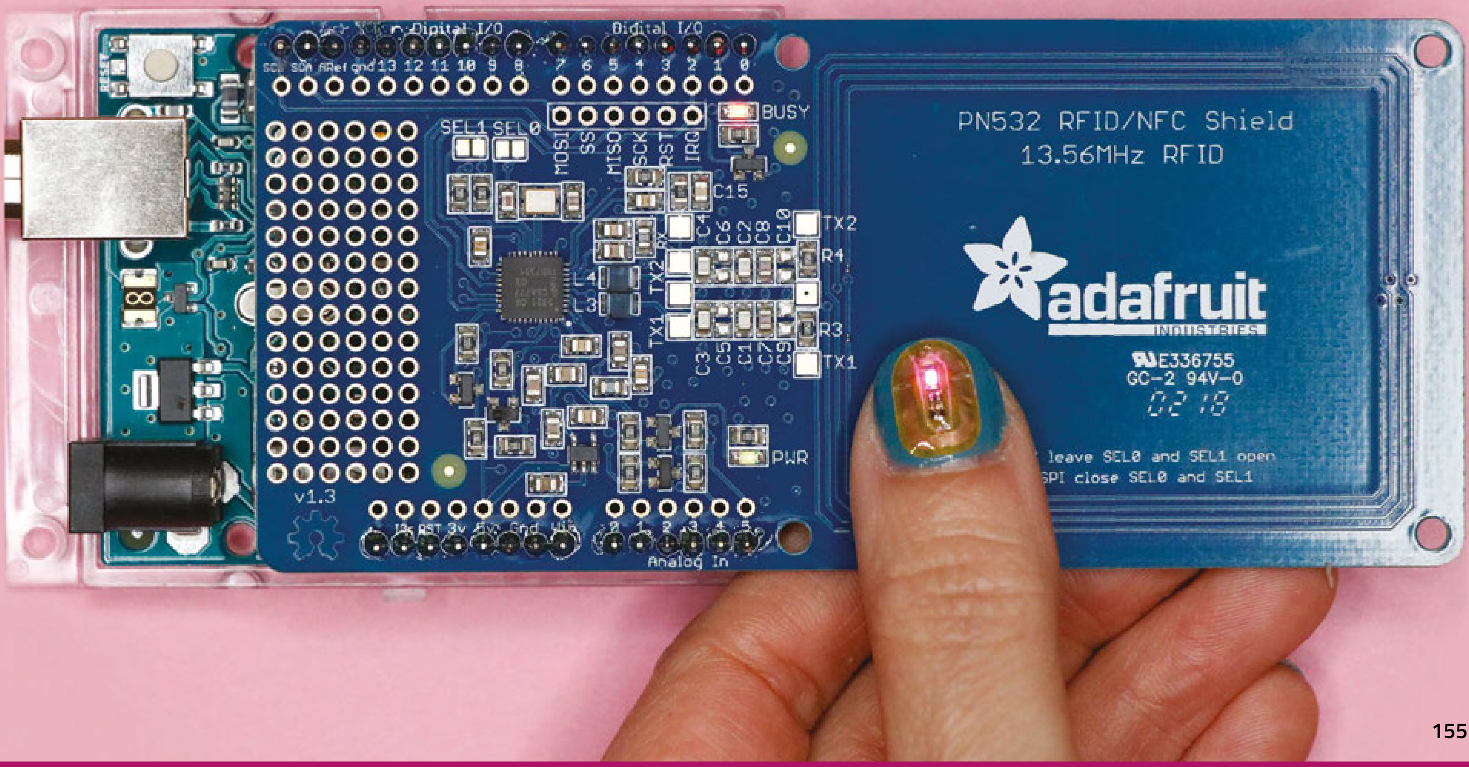
Tinkering with these nail stickers sparked our curiosity about NFC and RFID, and this author really enjoyed hunting down NFC fields while out and about. It was a sci-fi moment every time the LED lit up, and it revealed many invisible electromagnetic fields we pass through every day. While the stickers are a bit of a novelty without the ability to hold or transmit data, their low price and high impact makes it easy and exciting to get started with NFC and wearables. □

**Below** The nails need a source of NFC power, such as a mobile phone or NFC development board

## VERDICT

**Cyberpunk jewellery, NFC field-seeker, or pure curiosity. These stickers are great fun and easy to use.**

**8**/10



How I Made

# MINI.MU GLOVE

Turning an idea into a product

By **Helen Leigh**

**The MINI.MU is a make-it-yourself, gesture-sensing musical glove for children.** It is a collaboration between international recording and performing artist Imogen Heap, the MI.MU glove team, the super-friendly people at Pimoroni, and myself.

My name is Helen Leigh. Things I am: a children's author, an electronics nerd, and a maker of creative technology. Things I am not: a product designer, a marketing guru, or an engineer. I made the first version of this glove in a shared makerspace in Hackney, and six months later it's about to become a real product that people all around the world can buy. It's been an eye-opening journey for me as a maker, and I want to share the story of how the MINI.MU went from make to manufacture.

## HOW IT STARTED

I'm a hardware hacker and a musician. All the best things I've done combine these two things. I've made breadboard synths live on air on *BBC Click*, I've made DIY electric guitars in classrooms all over the country, and I've made a neon pink wearable capacitive touch synth in a field with my friend Phoenix Perry.

As I became more involved in music tech hacking, I became part of the Hackoustic and Music Tech Fest communities, where I met people making extremely odd musical instruments. At our awesome gatherings, I saw music made with marble runs (by Andrew Hockey), terrifying pipe organs made out of hacked Furbies (by Sam from Look Mum No Computer), a gong made out of a massive rusty saw blade (by Tim Yates), and interactive games with adorable singing capacitive touch robots (by Phoenix Perry).

I started to use these ideas as inspiration in the classrooms I visited, to teach subjects as diverse as music theory and physics, and saw the effect it had on the imagination and understanding of what can often be seen as very dry subjects. I also saw how naturally young people take to making noises, and

**Above** The MINI.MU glove: bringing technology and music to a school near you

**Right**
**Users can make the glove in different sizes to fit their hand**

how music can be used as a joyful vehicle for knowledge.

At around this time I saw a demonstration of a prototype MI.MU glove, invented by Grammy award-winning recording artist Imogen Heap and her team, and I immediately started thinking about how I could bring this magical instrument into the classrooms I worked in.

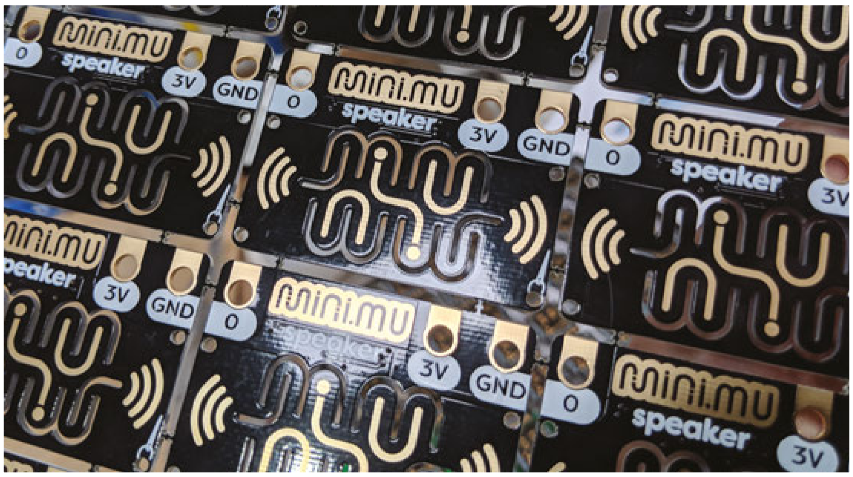
## WHAT IS THE MI.MU?

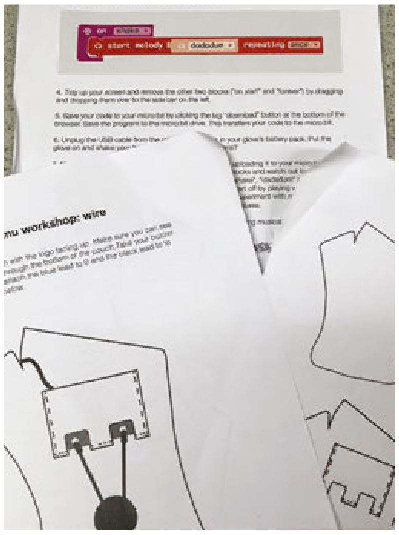
The MI.MU glove shows that there is a better way to make music than with sliders and buttons – through the complex movement of the human body. The MI.MU glove combines textiles and electronics, including flex sensors, gesture detection using IMU, haptic feedback, and state-of-the-art low-latency wireless communication. It uses special software to translate movement into MIDI or OSC messages, and has been used around the globe by world-leading musicians, including on Ariana Grande's world tour.

I couldn't stop thinking about this idea, so I contacted Imogen to ask if she'd mind if I designed a hack for kids, based on their technology. To my surprise, Imogen's response was to ask if I could make a prototype to show for their big upcoming investor event, which happened to be in ten days. That's ten days to design, prototype, classroom-test, and make a new product. Naturally, I said yes: who needs sleep when you can make musical instruments with lasers and sewing needles?

I needed to work fast and be very precise about what I wanted to make. The glove needed to be as cheap as possible, easy to make, and extremely fun. I needed to figure out how to strip out a lot of the sensors and simplify the functionality so we could find a way to make the idea of a gesture-controlled musical instrument accessible to a lot of people. The most obvious sensor for gesture-based anything is an →

**Below**
**The PCB speakers are easy to mount on fabric**




on shake
start melody dadadum repeating once
4. Tidy up your screen and remove the other two blocks ("on start" and "forever") by dragging and dropping them over to the side bar on the left.
5. Save your code to your micro:bit by clicking the big "download" button at the bottom of the browser. Save the program to the micro:bit drive. This transfers your code to the micro:bit.
mu workshop: wire

**Above** Good documentation is a key part of a good product

**Left** Early prototypes used a more conventional speaker

accelerometer, which can sense where it is in space on the x-axis, y-axis, and z-axis, which are also known as roll, pitch, and yaw.

I'd been using the accelerometer-based gesture control functionality built into the micro:bit for a while, and I knew it was cheap, fairly indestructible, and came with a child-friendly way of coding it. The board does have limitations with the sounds it can produce, but the more I played around with it the more I came to love the retro gaming chiptune-style square wave bleeps and bloops the micro:bit can make. To finish it off, I added a laser-cut felt glove pattern, a battery pack, and a simple speaker connected by just three wires.

All of the prototyping was done in Machines Room, my makerspace in London, with my friend Rehana Al-Soltane. Having access to a laser cutter, an electronics bench, and a decent sewing kit was invaluable for making rapid changes to the first prototype.

The first version of the MINI.MU glove was born in less than ten days, but straight away we knew it was an idea that had potential. This was where my expertise started to falter: I'm a maker, not a product designer! I'd made and sold small batches of DIY kits out of my makerspace before, but nothing that required proper manufacturing, packaging, or distribution.

The level of interest starting to build around this project started to feel overwhelming, so I reached out to someone who had done this before: the awesome Paul Beech from Pimoroni. I met Paul first at a party I threw during an Education Technology trade show called Bett. I invited teachers, technologists, artists, and makers to my makerspace and we all played games, shared learning, and had epic fun thinking about how education should be.

As Paul remembers it, "A month or so later she got in touch to casually say 'Had lunch with Imogen Heap. I have a great idea. Could I talk to Pimoroni about making it happen?'. We were keen, and super-happy that our work had not gone unnoticed."

This was where my expertise started to falter: I'm a maker, not a product designer!

After our initial chats, he invited me and Adam Stark, who heads up the MI.MU team, up to their factory in Sheffield to talk things over. The visit opened my eyes to the realities of how much work and organisation is involved in manufacturing and distributing maker products.

There was a whole massive wall of laser cutters constantly buzzing away, and giant shelves stuffed with well-organised electronics parts, packaging materials, and cool trinkets. Even better, there was a team of nerdy, friendly makers with tea, biscuits, and jokes. A maker's dream, essentially.

Adam said, "We were blown away by their mixture of positivity and professionalism. Their facilities are amazing, and they have a team who go about their work with a sense of joy and optimism. Pimoroni gave us the vital link we needed to help us take our ideas and prototypes and make them a reality that

**Left**
**Choosing the right fabric for the glove**

**Above**
**The author ready to lay down some grooves**

can be shared with thousands of people." Pimoroni guided us through the issues of manufacturing the prototype I'd made at scale. As Paul explained, "There's a mountain of difficulty you have to climb in making a product. Making one or ten of something is tough. Making 100 or 1000 is that much harder. Making more than that, and making the kit lovable, affordable, and interesting to shops, rather than just selling yourself, is really, really hard to get right.

## LEVELLING UP

"This was a glove that kids would use all over the world, so that meant that everything had to be quality. We had to make sure the felt had to have a proper Material Safety Data Sheet which says how safe and fireproof the material is, that the whole project had good documentation, and that all the boxes were ticked to allowed us to get crucial safety certification."

The choices seemed endless at this point, from the dozens of types of cardboard boxes we looked at, to the style of packaging, or the exact weight of felt and the size and shape of the needles supplied. We had to strike a balance between cost, practicality, durability, and design. Rachel Freire, the e-textiles expert and designer at MI.MU, explains: "The MI.MU glove has to balance beauty, comfort, functionality, and robustness. This is a lot for a product to do! We had to make sure that the MINI.MU's design aesthetics fitted with our vision for MI.MU. Seeing the final design for the MINI.MU was super-inspiring and having it as part of our world is brilliant. I love that each glove is handmade by the user, so they really understand how it works. It brings us back to the core of why our gloves exist: to be accessible and inspiring."

Before making the MINI.MU glove, when I thought about the cost of a product, I used to think about the cost of each of the things inside the box. However, much of the cost of a product can come from labour and machine use. For example, the original glove was laser-cut: perfect for prototyping, but on a larger scale it would cost too →

**Right**
**Pay attention to packaging, as it can be more complex than you might think**

**Right**
**The gloves can be personalised by the users to match their own style**

much employee and machine time. Instead, we decided to die-cut rectangles of felt that we then screen-printed with the pattern for kids to cut out themselves. This process could be automated far more simply and therefore turned out to be much more cost-effective for a bigger run of products. In any case, we realised that the screen print idea turned out to be a better design choice, as we were able to easily allow for different sized hands.

Solving the problems of making this kit, and getting an agreement, took a few months of researching suppliers and prototyping the look and feel of the kit. Alongside making sure it looked the way we wanted, we had to consider the price implications of every decision. It's not as simple as adding up how much everything costs and putting it on the market for a little more than that: you have to look at what people are willing to pay for the type of the thing you're making and make sure your product fits in the right bit of the marketplace.

We decided that we wanted to make the full kit available for under £40, including a micro:bit – not an easy task, especially when you take into account that a retailer will want to make a margin of 40% on that price. At every stage of the physical product development, I was working with young people, teachers, and families to make sure that what we were doing made sense. When you're designing something, it can be really easy to get blinkered and keep going with your own ideas, but that is never going to lead to a great product. If you're designing anything, whether it's a mass-manufactured product or a service like an educational club, you're going to make it way better if you ask the opinions of the people it is for. It sounds obvious but you'd be surprised at how many people don't do it!

You're going to make it way better if you ask the opinions of the people it is for

Something that helps me when I'm designing, especially for education and families, is to split the types of people I'm testing it with into two groups: users and clients. Users are the people who use the product – in this case, the children making the glove – and clients are the people who actually part with the cash to buy it – in this case, usually parents or a head of

**Left**
**Testing the glove with larger speakers**

**Left**
Short crocodile leads are easy to connect and stay in place during use

department in a school. To give your ideas the best chance of survival, you should make sure that you are making both groups happy. Another thing that's worth mentioning is that people – especially children – love to be asked their opinion. In schools, I ran workshops and explained that they were part of a special group of prototypers and testers that would help make this glove into a real product on a shelf. Every classroom and family I worked with took their 'job' very seriously, and many felt a great sense of empowerment and accomplishment because of the advice they gave us.

I tested the MINI.MU glove with over 250 different people in five sets of tests, and watched very carefully each time. How did people interact with it? What did they get wrong? What did they find difficult? How can we give the moments of joy we saw the best chance of being repeated for every child who makes the glove? Each time we tested the glove we changed something, and the end product is so much better for it.

## IS IT TIME TO PRESS 'GO' YET?

By this point in the process, we were all very happy with the MINI.MU. I spent a joyful week in Sheffield finalising designs, pressing 'go' on the pick-and-place machine (#dreamjob), and working on the teacher resources with our education writer dream team including myself, Lorraine Underwood, and Tanya Fish.

Probably the funniest part of this week was an afternoon spent lying on the floor with my arms at strange angles, while acting as the hand model for the packaging! We were almost ready to go, so we decided to finalise the paperwork. However, getting a contract in place turned out to be by far the slowest part of the process – taking almost as long as the entire rest of the prototyping and production process put together.

Adam, who heads up the MI.MU team, shared this advice about contracts, "Do not underestimate the time and complexity involved with getting a legal framework in place. We were 100% convinced Pimoroni were the right partner and just needed a simple, friendly, clarifying document on what we had agreed informally. So our task was easy, or so we thought! However, we still found that when it came to lawyers drafting contracts, there were a lot of hypotheticals to pick through. We also found that insurance companies place their own restrictions on what should be in a contract – so this should be taken into account."

After the contract went back and forth several times, we managed to get it signed and approved with just one day to spare before our planned pre-launch date on Ada Lovelace Day. It felt wonderful to finally set the MINI.MU free into the world, and the pre-launch itself was spectacular, with 80 gloves made by 80 kids in two days in London, and a simultaneous workshop happening on Imogen's tour date in Iceland.

## GO, GO, GO!

As this article goes to press [December 2018], we're taking delivery of the final bits and pieces to go in the box before it ships in the next few weeks. The sewable speaker boards have been delivered, the felt has been cut, and the packaging has gone to print. The biggest job we have left to do now is to start telling people about it! We've had a little bit of a head start with it going on Imogen's 40-date world tour, but getting a product off the shelves and into the hands of children is no mean feat.

We are so proud of what we've done so far. At MI.MU, we are all interdisciplinary artists and makers – we all work at the junction between art and technology. We hope that the MINI.MU can bring to children the idea that all creation is fun and they don't need to be put into the boxes that society creates for us. They can be both an artist and work with technology, that they can be creative with code and expressive with electronics. Above all, we hope they learn that they can achieve anything if they put their minds to it. □

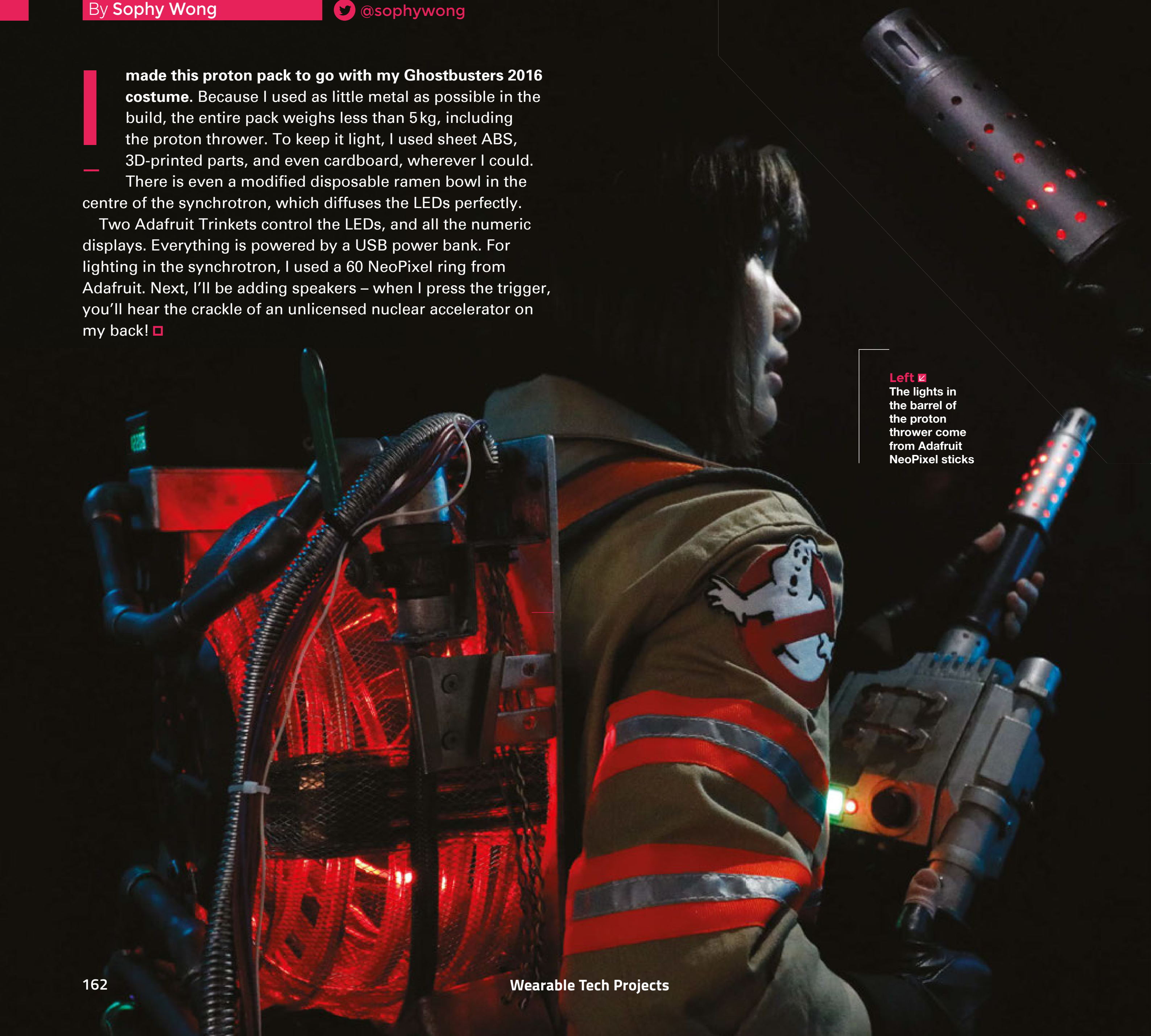
**Below**
The final kit, boxed up and ready to ship

# Ghostbusters 2016 Proton Pack

By Sophy Wong @sophywong

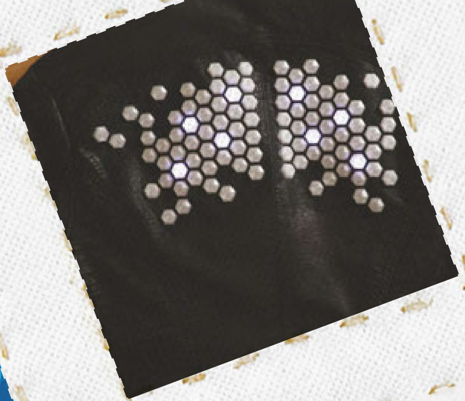
**I made this proton pack to go with my Ghostbusters 2016 costume.** Because I used as little metal as possible in the build, the entire pack weighs less than 5 kg, including the proton thrower. To keep it light, I used sheet ABS, 3D-printed parts, and even cardboard, wherever I could. There is even a modified disposable ramen bowl in the centre of the synchrotron, which diffuses the LEDs perfectly.

Two Adafruit Trinkets control the LEDs, and all the numeric displays. Everything is powered by a USB power bank. For lighting in the synchrotron, I used a 60 NeoPixel ring from Adafruit. Next, I'll be adding speakers – when I press the trigger, you'll hear the crackle of an unlicensed nuclear accelerator on my back!



**Left** The lights in the barrel of the proton thrower come from Adafruit NeoPixel sticks

220655
1488



Ditch the circuit board, step away from the computer: it's time to use your tech skills to make your clothes electronic. From LED fashion to games controllers, we've got wearable projects for all skill levels to add sparkle, control, and a unique look to your wardrobe.

- **Get started with electronics and programming**
- **Sew your own circuits with wearable components**
- **Stand out from the crowd with light-up clothes**
- **Make replica sci-fi accessories**

Q2 2019 **£10**

9 781912 047635

Raspberry Pi PRESS