Make:

MOTORIZED SELF SOLVING RUBIK'S CUBE

MACHINE LEARNING

Swear Bear... Trash Sorter... Easy AI Trainers
Teach your project to think for itself

BENJAMIN CABÉ'S "NOSE" KNOWS!
Build this smell-identifying AI sniffer

23 PROJECTS!

- **Raspberry Pi Meteor Camera**
- Digital String Art Portraits
- **Animated LED Skirt**
- Arduino Borealis Lights

SKILL BUILDERS

- **Digital Mobile Radio**
- Hack a Knitting Machine

makezine.com | makercampus.com | makerfaire.com

VOLUME 77


COMING SOON!
Register Today. makercamp.com
Maker Camp
M
Make:

IRIS™ Camera

- Overhead Cameras Provide Live Views
- Drag and Drop Artwork on Screen
- Easy Setup and Positioning
- Quick Onscreen Camera Layout

EpilogLaser

888.437.4564 | SALES@EPILOGLASER.COM | EPILOGLASER.COM/MAKE

Make: 77 CONTENTS

ON THE COVER:
Benjamin Cabé's electronic nose uses artificial intelligence for aroma identification.
Photo credit: Mélanie Duval

COLUMNS

FEATURES

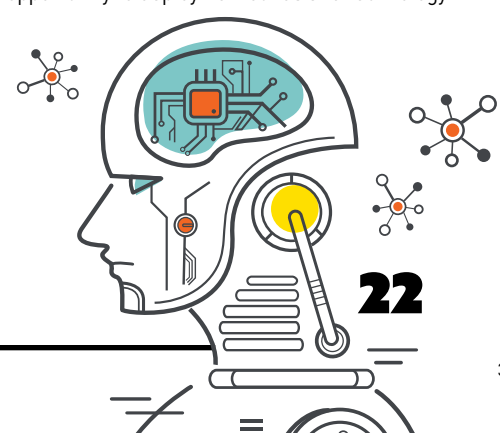
MACHINE LEARNING

PROJECTS

SKILL BUILDER

TOOLBOX

OVER THE TOP



Mélanie Duval, Takashi Kaburagi, Raphael Schaaf, 8 Bits and a Byte, Joe DiPrima, Martin Oehler, Adobe Stock - Buffaloboy

Make:®

"It's against my programming to impersonate a deity."
—C-3PO, *Return of the Jedi*

Comments may be sent to:
editor@makezine.com

Visit us online:
make.co

Follow us:
@make @makerfaire
@makershed
makemagazine
makemagazine
makemagazine
twitch.tv/make
makemagazine

Manage your account online, including change of address:
makezine.com/account
866-289-8847 toll-free in U.S. and Canada
818-487-2037, 5 a.m.–5 p.m., PST
cs@readerservices.makezine.com

Make: Community

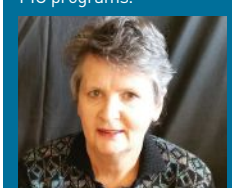
Support for the publication of *Make:* magazine is made possible in part by the members of Make: Community. Join us at make.co.

CONTRIBUTORS

What's a summer project you're excited to start working on?

Jen Fox
Seattle, WA
(Clean Up Your Act)
A crow training device where crows exchange trash for food. Crow friends + a cleaner neighborhood = the best win.

Shawn Hymel
New Orleans, LA
(Deeper Learning)
I'm excited to tinker with the new Raspberry Pi Pico and learn how to make PIO programs.



Adrienne Hunter
Santa Clara, CA
(Pixel Knitting)
This summer I'll be refurbishing an antique knitting machine from the 1860s.

Issue No. 77, Summer 2021. *Make:* (ISSN 1556-2336) is published quarterly by Make Community, LLC, in the months of February, May, Aug, and Nov. Make Community is located at 150 Todd Road, Suite 200, Santa Rosa, CA 95407. SUBSCRIPTIONS: Send all subscription requests to *Make:*, P.O. Box 566, Lincolnshire, IL 60069 or subscribe online at makezine.com/subscribe or via phone at (866) 289-8847 (U.S. and Canada); all other countries call (818) 487-2037. Subscriptions are available for $34.99 for 1 year (4 issues) in the United States; in Canada: $43.99 USD; all other countries: $49.99 USD. Periodicals Postage Paid at San Francisco, CA, and at additional mailing offices. POSTMASTER: Send address changes to *Make:*, P.O. Box P.O. Box 566, Lincolnshire, IL 60069. Canada Post Publications Mail Agreement Number 41129568. CANADA POSTMASTER: Send address changes to: Make Community, PO Box 456, Niagara Falls, ON L2E 6V2

READER INPUT

Go Big and Float Home

Jim Merullo

MINIFIG MASTER

Make: *reader Jim Merullo sent in these beautiful photos of his take on the jumbo-sized Lego figure from Volume 76's "Stick Man" (page 62). He writes:*

> Been a long time reader of *Make:*. So glad it exists and I was really happy to see a project that didn't require a 3D printer or Arduino knowledge!

AND WE ALL FLOAT ON ALRIGHT

Hi, just wanted to let you know, this kayak is a brilliant, brilliant idea. I totally love it. I'm sharing it with as many people as I can think of.

–Charles Justice, via email

Hi *Make:* magazine,

I really enjoyed this article ["DIY Folding Kayak," Volume 76, page 28] but wanted to provide some advice to Nat Taylor and Hong Wong.

Please add some blow-up buoyancy bags to their designs to make their boats safer. Almost all commercial kayaks contain air or foam buoyancy so that they do not sink and you lose your craft. More importantly the buoyant craft provides life support should you capsize.

Keep the great articles coming!

–Paul Saunders, via email

Project originator Hong Wong replies:

Thanks for the feedback. I totally agree with this.

The buoyancy bag usually is an aftermarket accessory, which the kayak owner can add easily. This NRS float bag (Amazon #B00241Q8RM) should work with the *Make:* kayak design. The kayaker can add one, or two (bow and stern).

As for life support, the Coast Guard requires a kayaker to wear a life jacket or PFD (personal flotation device). Having a floating kayak helps, but it wouldn't help a kayaker who lost consciousness. A PFD is a must have; a floating kayak is good to have.

Note that because this foldable kayak is about half the weight (or less) of a 12'-long hard shell kayak, a smaller air bag will be able to keep the *Make:* kayak afloat.

When Your Device Breaks Up With You

by **Dale Dougherty**, *Make:* President

Imagine you are one of the early backers of a Kickstarter that, like so many of them, promises world-changing benefits in a brand new smart device. You were not only early, waiting for them to build and deliver the promised device, but yours was among the first ones they made. It was a little rough around the edges but you and others gave them feedback and you made it work for you. You were on board. You told others how good it was.

You were happy also when the scrappy team that created the product was acquired by a big company as part of a corporate IoT strategy. You even thought that could be good for you, too, as a customer — more support, more developers, an even better relationship. It felt like backing a band before they became huge.

But things change over time, which is the case for SmartThings, a device that raised $1.2M on Kickstarter in 2013. It was the one smart thing that made all of the other things in your home talk to each other, regardless of what protocol they used. SmartThings became a company that received $12 million in venture capital before it was sold to Samsung in 2014 for a reported $200M. But SmartThings change too.

Steve Texeira received a break-up message in March that his v1 SmartThings Hub would no longer work after June 30, 2021. The message was even more irritating because it bragged about the original world-changing promise of SmartThings to "make every home a smart home." There are "over 63M users around the world, including you." Remember those good times, it wants you to understand. It wants you to feel good, even if it is breaking up with you. Your device is being retired, the message says. Retired? Did your smart home future age out?

You just didn't see it coming. The custom code you developed for it won't work anymore. Your friends in the user community, who shared code with you, are also gone. You'll have to start over again.

"It's just awful when smart home hardware is end-of-lifed, particularly when that hardware is non-trivial to physically install," Steve wrote on Twitter (find him at twitter.com/stevetex). "ZigBee/Z-Wave devices are notoriously annoying and time consuming to pair with a hub." We all have had devices that stop working or become outdated because of newer models. But there's something disturbing about a "smart home" device that retires before you do.

We don't think much about the end-of-life for technology, or our relationship with the devices we have bought. Steve wrote in a follow-up to me: "I don't think about my microwave oven getting bricked in a few years, but this is a real concern for many smart devices." A device dependent on the cloud becomes useless when a company like Samsung decides that it is not part of their future. Steve would like to see stronger consumer protections, a kind of "guaranteed device lifetime or a fallback to a reasonable local-only mode if the cloud service goes away." Open communities are also needed, such as the Pebble Rebble which rallies support for a discontinued smart watch.

Frankly, it feels a lot better when you decide that your device is not working for you anymore. This is not like that. Here technology is abandoning you, not you abandoning it. You'd like to stay in the relationship but it's done with you.

In a future with robots in our home, is it possible that they might decide to stop working for us? Will a robot that you thought was yours take all its machine-learning about humans and go off to the cloud somewhere to become a gig worker? When that happens, the robot, on its way out, will remind you that now you have to pick up after yourself, cook for yourself, and learn to do all the things yourself that it once promised to do for you. "Now you must remember to turn off your own lights," the robot will say, and you'll be left smarting about your not-so-smart home.

MADE ON EARTH

Backyard builds from around the globe

Know a project that would be perfect for Made on Earth? Let us know: *editor@makezine.com*

ART-O-MAT

ARTOMAT.ORG

What do you do when something ubiquitous like the cigarette vending machine becomes illegal for use? For **Clark Whittington**, the response was to sell works of art. The North Carolina-based conceptual artist has spent 25 years repurposing now-prohibited-in-the-U.S. vintage cigarette machines to vend small, handmade creations — paintings, drawings, sculptures, jewelry, mini-macramé, and whatever else over 400 contributing artists come up with to stock 170 Art-O-Mat machines nationwide. "Some are real active, others are there waiting for the right person to come along," Whittington says. "With Art-O-Mat, I feel we're teaching people that art is not scary."

It's easily the most accessible art gallery anyone will ever come across, and may be the cheapest, with each work costing just $5. Each machine is a piece of art itself, being uniquely designed to fit the interior of each location — a Whole Foods Market in Boston, the Cosmopolitan Casino in Las Vegas, or the True North Barber Shop in Phoenix, just to name a few of the diverse businesses that host these pieces of American nostalgia, now serving as a doorway to the independent art world.

In 2018, Art-O-Mats moved 90,000 pieces. But like most retailers across the country, sales took a dive in 2020 due to the pandemic. However, Whittington is confident "people will be foaming at the mouth" for new experiences in the post-Covid world, so he's already anticipating his stockpile will quickly deplete once more businesses open their doors again. "We're always looking for artists," he says. "We need art that is really good, that really makes people want to reach out to them two steps away from the machine after they vend it." —*Greg Gilman*

Artists In Cellophane, Vikki Vassar

WANDERING ON WATER

PEOPLESRIVERHISTORY.US

Wes Modes wears many hats, and one is captain.

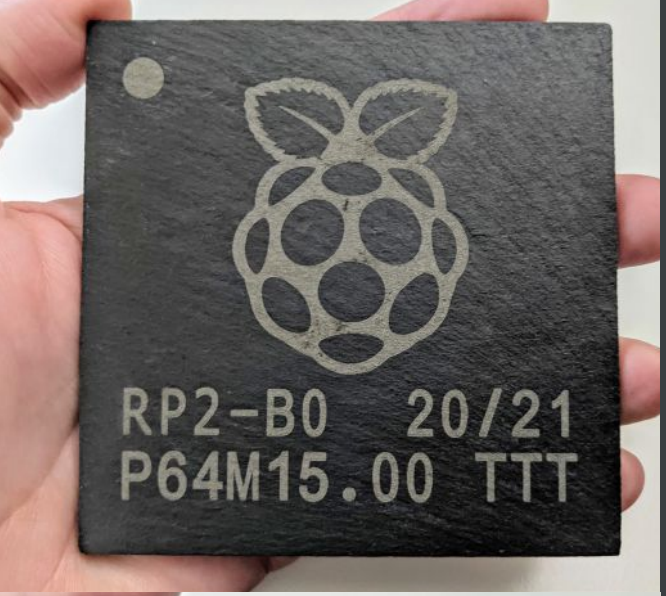
The northern California-based educator and artist spent most of the last decade dedicating his summers to floating down 2,600 miles of river aboard his shantyboat, the *Dotty*, to explore the forgotten life of the American river people. "Rivers were the arteries of America," he says. "It was how goods got around, it was how people got around, and it was how ideas got spread."

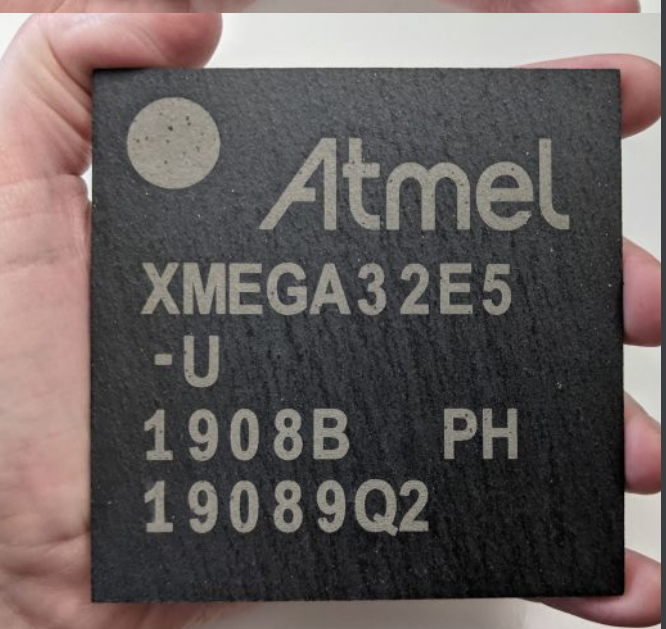
Rivers also offered refuge to thousands of poor Americans, who would scavenge for barrels, tires, tin, and other free parts to build a home to dock on the fringes of society. But after the government cracked down on pollution in the 1970s and riverfront property became more desirable, the people living in these tiny floating houses were essentially evicted, and the shantyboat became a relic of poverty's past. For Modes, however, the cabin-like vessel became an escape from an unfulfilling job back in 2012, when he started building his from scratch. He dug through junkyards, chicken coops, old boats, and other sources for recycled materials to find nearly all of the pieces above the carefully designed hull, plywood covered in fiberglass.

"I think my biggest handicap was just ignorance," he says. "I didn't know anything." Fortunately, he found a company called Glen-L Marine that sold him a detailed design plan to follow.

After two years of construction, Modes embarked on his maiden voyage, down the Sacramento River, in 2014. He has since notched up five more river runs, with frequent public exhibitions in between, including the Maker Faire in Oakland. This summer, he plans on writing a book, drawing from the insight of thousands of river people he has met throughout his travels. Then in 2022, Dotty will be back on the water traveling through Louisiana's Cajun country on the Atchafalaya River. *—Greg Gilman*

Adrian Nankivell





Jason Coon

SLATE CHIPS

EVILGENIUSLABS.ORG

Kansas-based software engineer and maker **Jason Coon** has been delighting the Twitterverse lately with his laser-etched, slate-based beverage coasters done in the styling of classic and modern computer chips. He calls them "macrochips."

"I first got the idea after I saw a tweet by @arturo182," Coon says about the inspiration behind the project. "I knew I had to have one, and I already had everything I needed on hand: a Flux3D Beamo 30W laser cutter, some slate coasters, GIMP, and Inkscape."

Coon creates the designs by first finding a good image of each chip online — "Google image search, Wikipedia, etc. have been great resources so far." He then uses GIMP to do any necessary digital cleanup to the image, or fires up Inkscape to trace over the images as vectors if they need redrawing. From there, he exports the file as an SVG and brings it into the laser cutter software. "The engraving can take 5–10 minutes for each tile. I engrave at 50% power, 80mm/s speed."

He's not selling the coasters, but has been making them by request, sometimes in trade for the electronic creations of others, and it's kept him busy. "What has surprised me most was just how popular they were," he says. "Some of the people I've sent coasters to actually worked on the chip design, or for the company."

—Mike Senese

Struck Twice

When Covid killed their live gigs, ArcAttack pivoted to create their first DIY musical Tesla coil kit. Then the pandemic struck again. **Written by Joe DiPrima**

When I was five years old, my dad gave me my first lessons in soldering and playing the piano. Dad was a biomedical technician. Simply put, he's the guy that fixed all the stuff in the ICU that keeps people alive. As a result, we had no shortage of electronic scrap and parts. At one point, we even had an entire X-ray room in our garage — a decommissioned unit that he eventually donated to a veterinary clinic.

Keep in mind in the 1980s, pre-internet, DIY electronics projects came from books and the sage wisdom of those who came before you. While most kids my age were learning the alphabet, Dad taught me the basics of logic gates and analog circuits. We had access to vintage computers years before they started popping up in our schools. To say he influenced our interests would be a bit of an understatement.

My brother John and I grew up in Michigan. We spent our formative years working with computers and playing music together. The long cold winters kept us indoors and focusing on our crafts. I was more into guitar playing, and he took to the drums and electronic music. When we weren't playing music, my evenings consisted of dialing into BBSes, learning computer programming, and downing Jolt cola. As a result, I'd spend my days napping through my classes.

The two of us played in several bands throughout those years. We became acquainted with music production and performing at live shows. We were even known to pack a venue from time to time. Even still, it seemed a bit far-fetched that we'd make a career as professional musicians.

Call of the Coil

My electronics education really took shape after high school. Though Dad preferred that I enroll in a university, I opted to learn from experience. In 1999, I got my first job working in consumer electronics repair. I had to fix everything — TVs, VCRs, DVD players, you name it. Being on the receiving end of all this broken stuff, you tend to learn a lot about what makes a great product. Every manufacturer had its quirk, whether it was inadequate heat removal or shoddy solder work.

It was also during this time that I was first introduced to Nikola Tesla. I knew surprisingly little about the inventor. My boss, Bob Strand, from one of the TV shops I worked at, replicated several of his experiments. He had Tesla coils made from homemade capacitors, plasma globes, and all sorts of fun experiments around the shop.

My interest really picked up around 2003 when I met my friend Steve Ward. We were both members of an online forum called The Geek Group. Their headquarters was near my hometown, so I would show up and help out with everything from sorting junk to building experiments.

One day, Steve came to the shop with a small solid-state Tesla coil he'd been working on. It was about 12 inches tall and made a pretty impressive spark. That is, compared to the spark gap units I had become familiar with. The machine had two knobs. One was for controlling the spark length, and the other controlled the spark's frequency. As a musician, my first instinct was to grab that frequency knob and eke out a frustratingly pitchy rendition of "Somewhere Over the Rainbow." It was at that moment I realized that the Tesla coil needed to be a musical instrument.

JOE DiPRIMA is the founding member of ArcAttack. In the lab, he's the primary hardware and software developer of the group's custom show equipment. On the road, he oversees the installation of ArcAttack tech, and plays the guitar and lightning guitar on stage.

Jay P Morgan

Jay Lee

IT WAS AT THAT MOMENT I REALIZED THAT THE TESLA COIL NEEDED TO BE A MUSICAL INSTRUMENT.

Before this, coil hobbyists had experimented with several methods for making high-fidelity sounds — that is, to make the spark sound like a speaker. Focusing on making musical pitches like an instrument, using more straightforward modulation techniques, was not only novel but also much simpler.

High-fidelity audio from sparks takes a lot of power. As a special effect, it's not very practical or obvious. In comparison, a Tesla coil making giant sparks, producing only basic pitches, is quite the spectacle.

In 2005, things took a turn. A longtime family friend had a data recovery business in Austin, Texas. He worked solo and wanted to take a vacation. Having the necessary skills to jump in and do the work, I decided to take a trip. February in Michigan is brutally cold. Getting picked up from the airport, driving down the Texas highways with the convertible top down was a revelation. After a few weeks in Austin, I decided to stay. After all, I didn't have that much going on back home. Austin provided a lot of unique opportunities. Being a mecca of technology, music, and art, it seemed like a good fit.

At this point, consumer electronics repair was nearly a thing of the past. Most new consumer items were so specialized that a technician could no longer repair them using off-the-shelf parts. I decided to take a second job working at a local music store. Here I got to fix a lot of cool vintage amplifiers, instruments, and other music gear. A small niche where my skills were still applicable!

Being new in town, with not a lot to do, it was the perfect opportunity to buckle down and develop our first musical Tesla coil. I spent my time learning from the blogs of Richie Burnett, Steve Conner, and other names in the Tesla coil hobby. Steve Ward and I continued to bounce ideas around. By the end of 2005, I had developed our first musical Tesla coil system.

Weird and Winding Road

Our first singing Tesla coil was a basic SSTC (solid-state Tesla coil). It had a simple transistor-

based interface to a cheap Casio keyboard. Despite its simplicity, the effect was mesmerizing. The first tune played from our machine was the demo song programmed into the keyboard. Undeniably, the technology needed further development. I continued to modify the coil until it could take input directly from a computer sound card.

In March of 2006, we demonstrated our magnificent machine at an event called Art Outside. John, still in Michigan, arranged a handful of electronic tunes and sent me the music files over email. I'd film them and send back the results. For the next couple of years, we worked on the project over long distance and posted videos on YouTube. As time went on, the technology improved and interest in this niche hobby continued to grow.

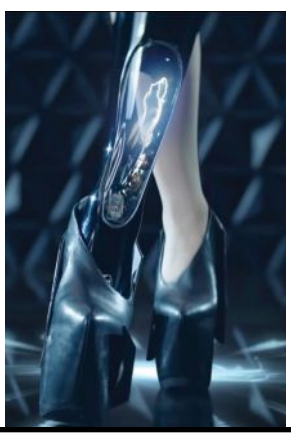
By 2008 we saw a steady trickle of gig opportunities. John joined me in Austin, and ArcAttack was officially named. We started off doing small local gigs, and soon we were finding shows all over the world. In 2010, we had a breakout opportunity on NBC's *America's Got Talent*. We didn't win, but we made it much further than we anticipated.

Shortly after, we were contacted by our agent Keith from Geodesic Management. He suggested that we should use our technological powers to create an educational show. To this day, we have been providing edutainment for school-age kids all over the country, on top of other random opportunities that kept us moving forward — Maker Faires, selling Tesla coils to museums, and other custom projects. In 2012, we zapped magician David Blaine with lightning for three days straight. We even electrified a prosthetic leg for "bionic" artist Viktoria Modesta.

Being a high-tech performance group is hard work. Small-scale entertainers make OK money. For us, the parts, research, and development cut into a lot of that. Not to complain, because our opportunities continued to grow. Without going into a lot of detail, 2020 was going to be our best year ever. It was evident that our patience had paid off: Our tech was flawless, we had assembled a great crew, and possibilities were abundant. Even Dad agreed I was probably better off on the path of self-education.

Kids brave the lightning during an ArcAttack performance at Maker Faire, safe within a Faraday cage.

Illusionist David Blaine pulls a similar stunt — for 73 hours straight! — wearing a custom Faraday suit.



ArcAttack designed this Tesla coil prosthetic leg with Anouk Wipprecht and Sophie de Oliveira Barata, for "bionic" artist Viktoria Modesta.

Photos courtesy ArcAttack, David Blaine / Electrified, Rolls Royce × Viktoria Modesta

High-fashion Faraday suit with Anouk Wipprecht, for Red Bull.

The world's biggest musical Tesla coil, Project Titan is 20 feet tall and can throw sparks 30 feet long. And you can rent it.

Crisis and Opportunity

Then came the pandemic. Over the course of three weeks, we saw the entire live performance market crumble. Our friends in the industry were out of work well before restaurants and bars had to close their doors. By May, it was apparent that we needed to pivot. For years we had dabbled with the idea of making a Tesla coil kit, but hadn't followed through since we were on the road so often. What if we're too busy to support our customers? Or to get orders out on time?

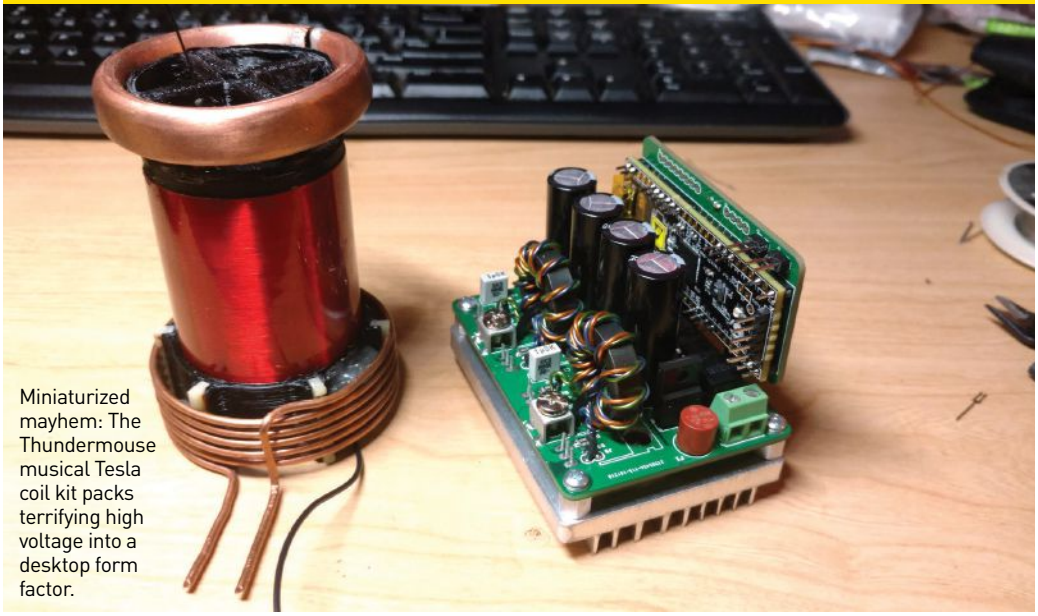
Now that all these issues had vanished, there was no reason to postpone any longer. John and I knocked out a great design over the next few months. We used all the tricks and features from our years on the road, not to mention my electronic repair experience. I have little doubt that we made the best Tesla coil kit available.

We decided to name our kit the Thundermouse. It's about 18 inches tall and can make sparks up to 3 feet or more. It connects to your computer with a USB device. It can do everything: make single large sparks, play MIDI music files, you can even plug your guitar into it. Despite its advanced features, we made sure to use only through-hole parts in its construction. Because of this, it's an excellent project for experts and amateurs.

BY OCTOBER, EVERYTHING WAS LOOKING GOOD FOR A SMALL RUN BEFORE CHRISTMAS ... THEN DAD GOT SICK.

But to mass-produce something, you need more than a good product. We had to build several jigs to aid in production. We automated everything from twisting wires to bending rings. Among the most complex machines we made is an automated secondary coil winder. It's capable of winding a 700-turn coil in under 4 minutes. By October, everything was looking good for a small run before Christmas ... then Dad got sick.

He was hospitalized with Covid-19 in November. Shortly after being admitted to the ER, the doctors put him on oxygen. All of our

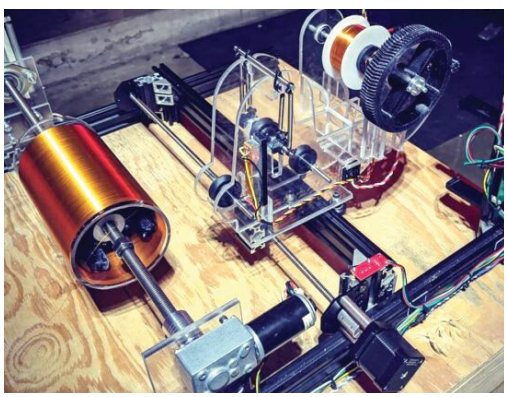


Miniaturized mayhem: The Thundermouse musical Tesla coil kit packs terrifying high voltage into a desktop form factor.

progress here stalled. John took off to Chicago to take care of business until he recovered. Sadly, things steadily got worse. Eventually, he required sedation and intubation. Despite receiving the latest drugs and therapies, his condition continued to degrade. Three-fifths of people admitted to the ICU with Covid suffer kidney damage. Dad was no exception. He eventually required dialysis. In the end, the machines he spent his whole life salvaging could not save him. He passed away in mid-December after a drawn-out and isolated hospital stay.

Our dad was in his early 70s. A bit old, he was mentally and physically spry. There is no doubt that he had a lot of quality time left. Life spares no irony because the week he passed was the same week that vaccines began to roll out. As a devout supporter of science, he would have been first in line for an early vaccine.

What a year. Like a lot of people with similar stories, we can only hope 2021 brings better news. There isn't much to do but continue as planned. We took a small hiatus, but now we're ready to sell our Tesla coil kit. We hope you enjoy it; we put a lot into it. It may be just the thing you need to keep your family entertained during the pandemic. Stay safe, wear a mask, and keep listening to the experts. If you get too bored, build a Tesla coil.



To produce kits, Joe and John built this machine to wind 638 turns of wire on the secondary coils automatically.

ArcAttack

The Thundermouse kit throws awesome 3-foot arcs in the comfort of your own home. Find it at arcattack.com.

Danger! High Voltage

Spark safely with these responsible Tesla coil tips from the pros **Written by Caleb Kraft**

Tesla coils, being high voltage, are particularly dangerous; that's part of their charm. Along with excitement comes responsibility: it's important to pay close attention to safety when dealing with any electrical devices, and especially these. The standard electronic safety precautions all apply here, like "Do not work on it while powered on," but you also have to consider the safety of the crowd watching — and if anyone else is nearby, you know there will be a crowd.

We asked ArcAttack about their best practices for putting on a spectacular show while making safety a priority for onlookers, and we'd like to share a few of those tips with you. For a more detailed breakdown, including DMX interfacing and radio interference, check out their page on performance safety with coils at arcattack.com/resources/tesla-coil-safety.

Protect your audience

While there isn't an exact distance you need to leave between your audience and the coil, it is important to create a firm safety perimeter around your coil. Everyone wants to be close, and they'll push as close as possible, even reaching out toward the coils despite the fact that doing so is terrifyingly dangerous.

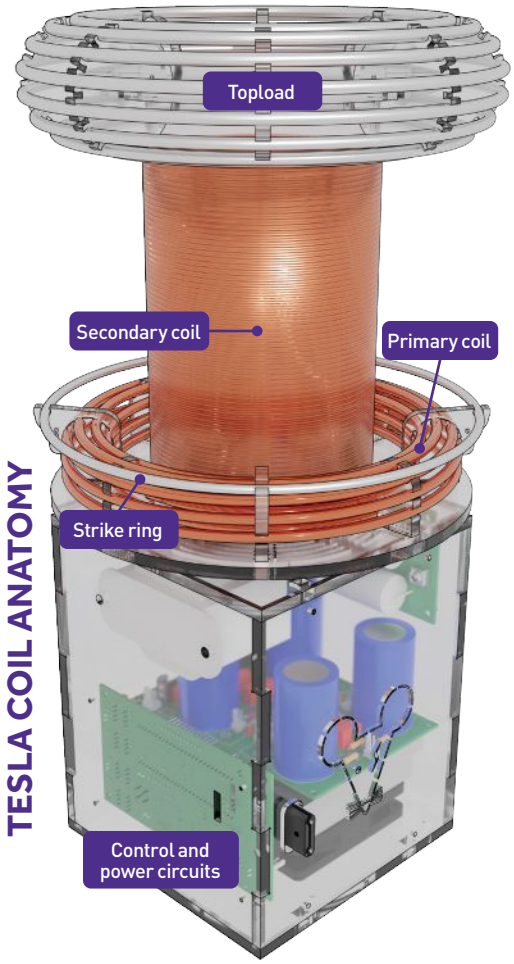
ArcAttack recommends you set up a hard perimeter at a distance of at least 10 feet beyond the longest possible arc your coil can produce.

Protect your crew

The audience isn't the only concern. The people operating the coil, or just existing in the vicinity of the stage for other reasons, need to be fully aware of the dangers at all times. To pull this off, ArcAttack recommends a minimum of two operators who are prepared to hit a kill switch at any moment, their hand literally held over the button during the performance.

Be vocal...

Have a crew member announce loudly the moment the coils are powered on, as well as powered off, and include some kind of visual indicator that the coil has power and could therefore be dangerous.



...and use hand signals too

You need to remember that these coils are extremely loud and chaotic. Use hand signs to communicate basic instructions and information. This will reduce the possibility of mishearing or simply not hearing commands at all.

Watch your ears

On the topic of being loud (and we mean LOUD), it's imperative that the crowd and crew alike all use hearing protection.

High voltage is no joke

Once again, there is serious juice running through these machines, which makes them dangerous even when not sparking, and this should always be kept in mind.

Now have some fun! Safely.

John DiPrima

Low-glare workstation lighting throughout

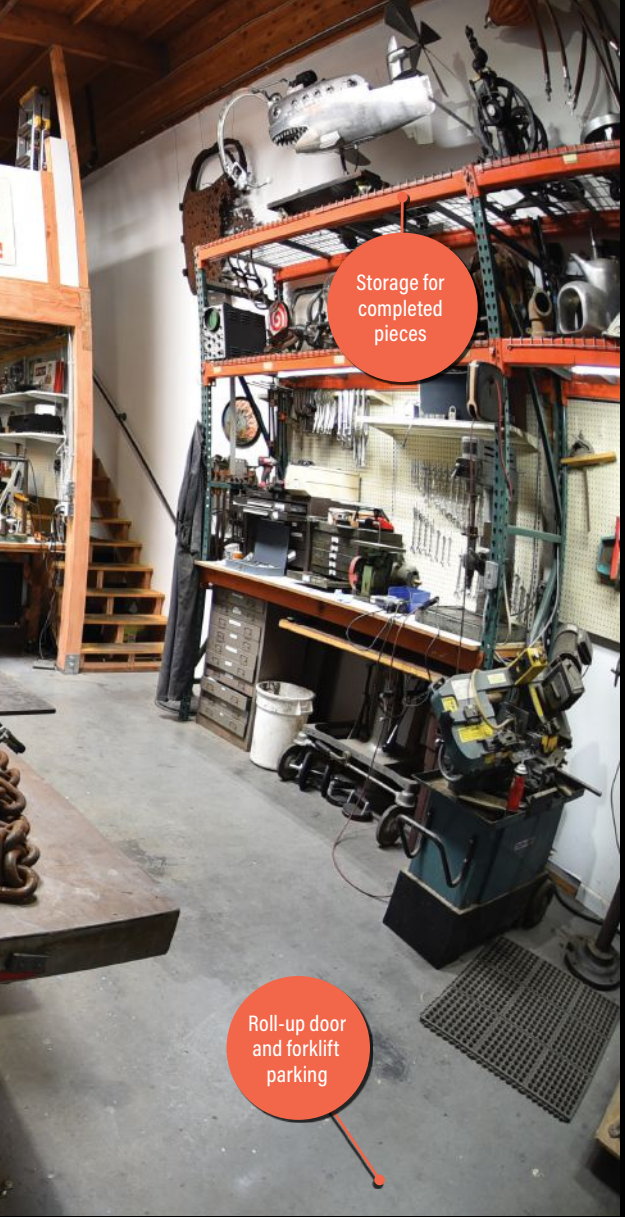
"This is probably my favorite spot, where the machine tools are situated."

"My workbenches function as sketchbooks. I lay out parts I'm thinking of working with until I'm convinced I have a composition that will work, then I get down making the sculpture."

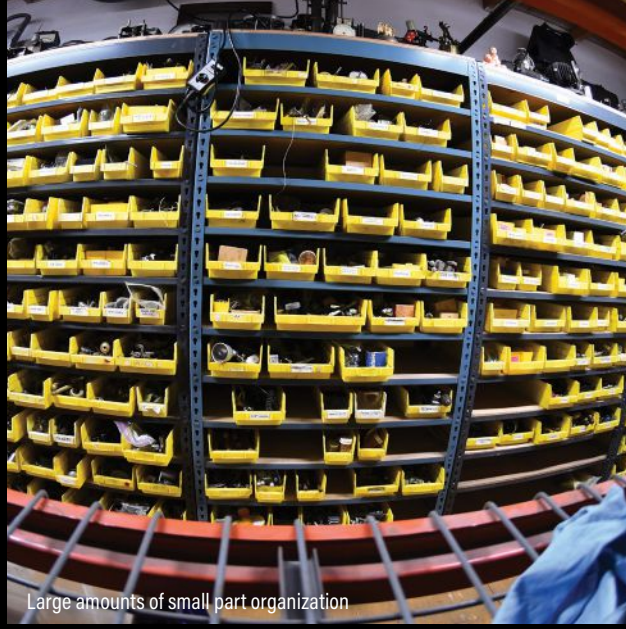
Shop Tour

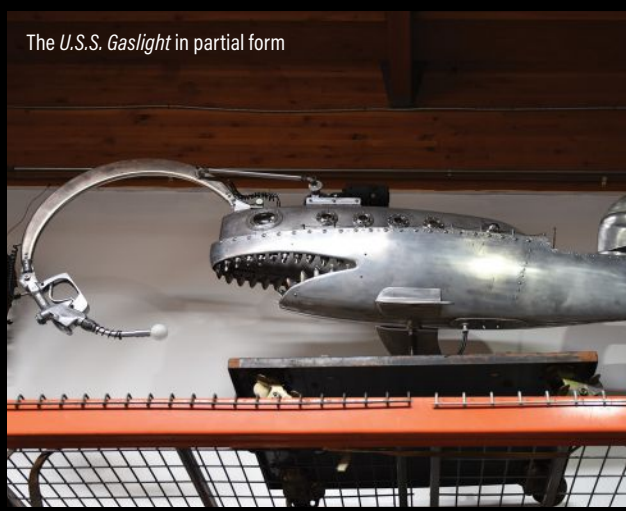
NEMO GOULD is an Oakland, CA-based artist focusing on kinetic, found-object art. His workshop advice: "Take organization seriously. If you can't find it, you don't own it."

See his works at nemogould.com and instagram.com/nemomatic.



Mike Senese

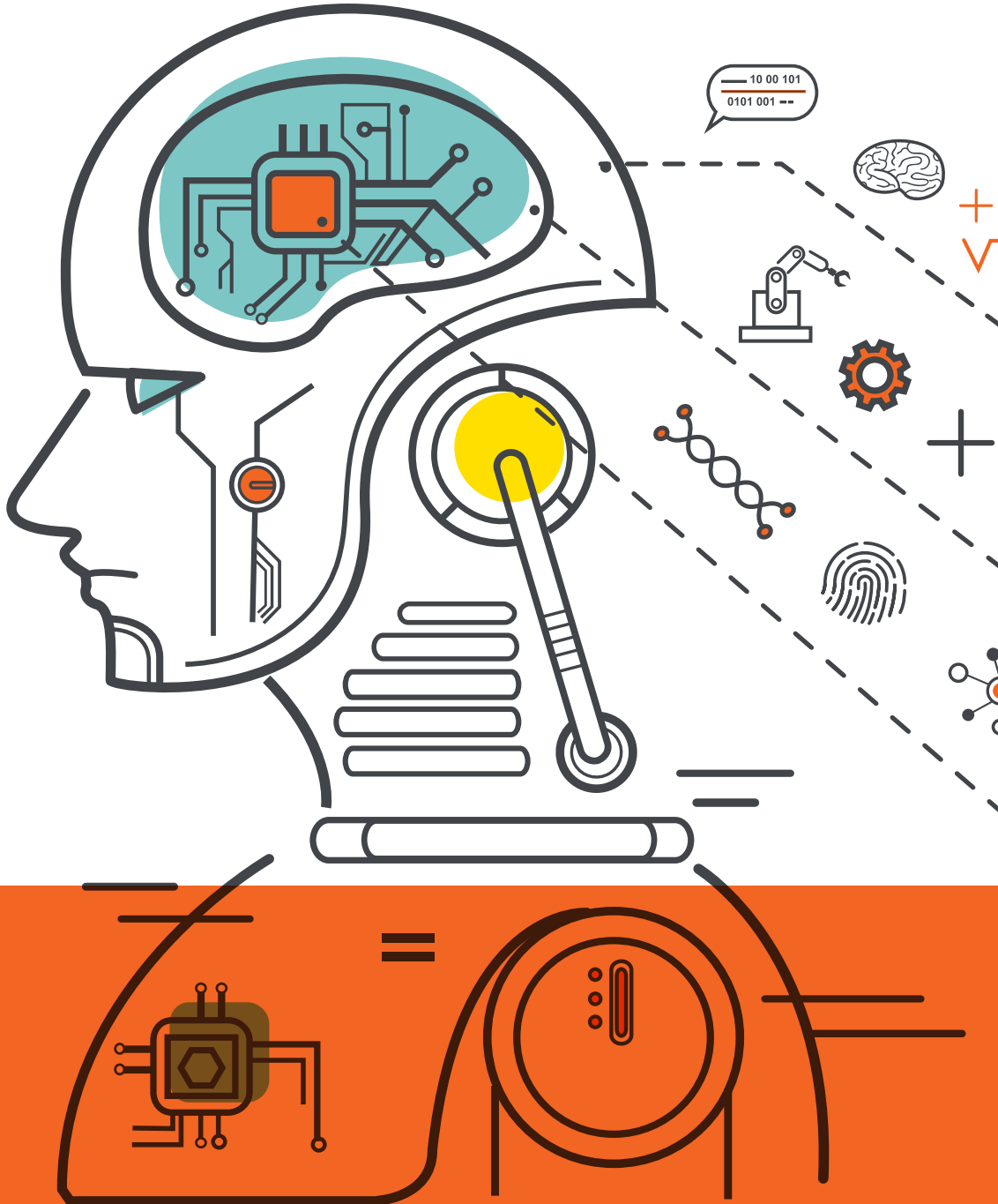

Large amounts of small part organization


The *U.S.S. Gaslight* in partial form


Shelves of supplies occupy the upstairs loft

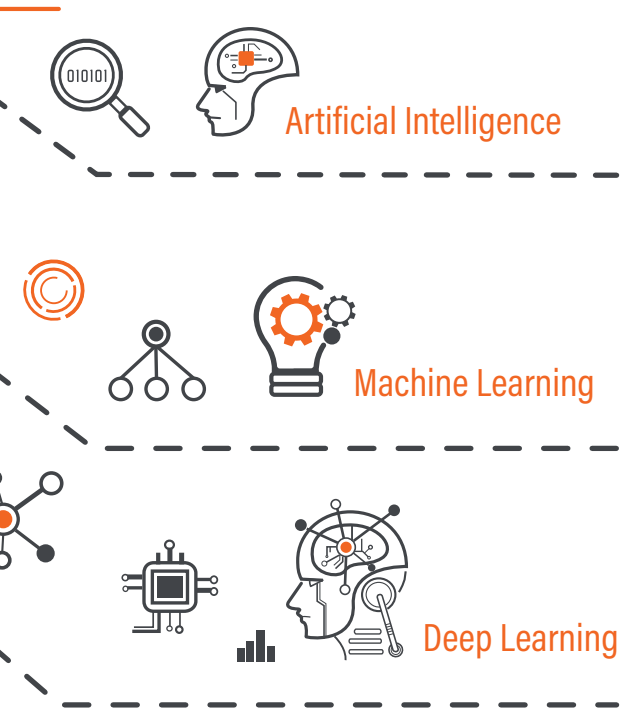
"I call it the '**Trash Mahal.**' It's the culmination of a lifetime of collecting treasures and tools."

—Nemo Gould



SHAWN HYMEL is an embedded engineer, maker, technical content creator, and instructor. He loves finding fun uses of technology at the intersection of code and electronics, as well as swing dancing in his free time (pandemic permitting).

Adobe Stock - Buffaloboy



DEEPER LEARNING

LEVEL UP YOUR AI SKILLSET AND DIVE INTO THE DEEP END OF TINYML

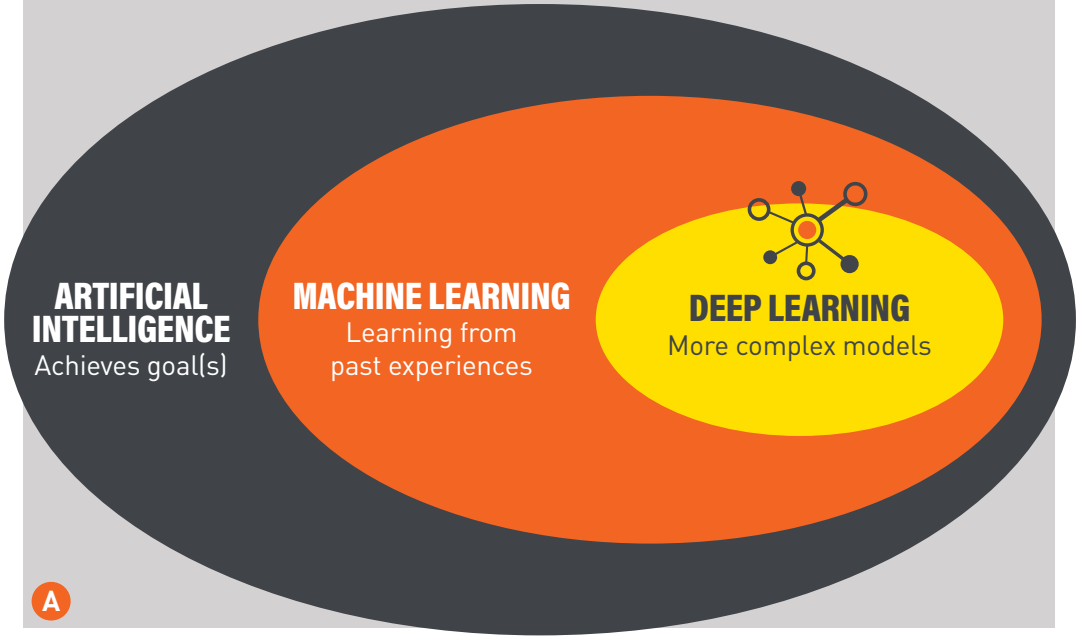
Written by Shawn Hymel

Machine learning (ML) is a growing field, gaining popularity in academia, industry, and among makers. We will take a look at some of the available tools to help make machine learning easier, but first, let's review some of the terms commonly used in machine learning.

John McCarthy provides a definition of **artificial intelligence (AI)** in his 2007 Stanford paper, "What is Artificial Intelligence?" In it, he says AI "is the science and engineering of making intelligent machines, especially intelligent computer programs." This definition is extremely broad, as McCarthy defines intelligence as "the computational part of the ability to achieve goals in the world." As a result, any program that achieves some goal can easily be classified as artificial intelligence.

In her article "Machine Learning on Microcontrollers" (*Make:* Vol. 75), **Helen Leigh**

DEEPER LEARNING: Going Further With TinyML



A

gives us a great definition of machine learning: "With traditional programming, you explicitly tell a computer what it needs to do using code, but with machine learning the computer finds its own solution to a problem based on examples you show it." In practice, this means collecting data or finding a pre-built dataset to **train** a mathematical model, much like training a child to recognize the difference between a dog and a cat from various photos.

The trained model should make successful predictions or **classifications** when presented with unseen data, in a process called **inference**. This process is similar to showing the child a new photo of a cat and seeing if they guess correctly.

In practice, training a machine learning model requires more complex math and computing power than inference does. As a result, we often see training occur on large desktops or servers, which gives us the option of performing inference on small embedded devices using the newly trained model. Training on a microcontroller is theoretically possible, but most microcontrollers don't have the memory and computing power necessary to perform the required calculations.

From this, we can conclude that machine learning is a subset of AI. All machine learning achieves some goal, so it is a part of AI, but not all AI programs are machine learning. Another common term you might run across is **deep learning**, which was coined by **Rina Dechter** in her 1986 research paper on machine learning algorithms. Deep learning is the use of more complex machine learning models to achieve better accuracy. Therefore, deep learning is a subset of machine learning (Figure A).

COLLECTING DATASETS

One of the biggest hurdles in machine learning is gathering data for training process. For typical ML training, called **supervised learning**, people must carefully curate the dataset, which includes labeling every sample by hand. Additionally, data scientists must eliminate or limit any biases in the dataset. For example, I created a voice-activated Halloween pumpkin that would laugh and flash whenever someone said "trick or treat" (Figure B). I mistakenly used only one adult male and one adult female voice to train the model to recognize the phrase. As a result, the model was biased toward adult voices; it was incapable of correctly classifying children's voices!

Bias, with regard to statistics and machine learning, is some error or distortion that stems from statistical analysis or model training. This personal story illustrates a type of **selection bias**, where the selection of data is not representative of the population intended to be analyzed or used

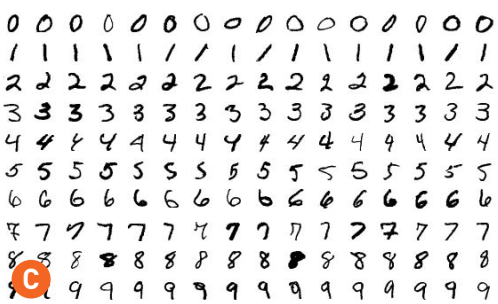
Shawn Hymel, Josef Steppan CC BY-SA 4.0

for inference. Bias is a big concern in statistics, as it can skew results and interpretations. Therefore, it is also a big concern in machine learning.

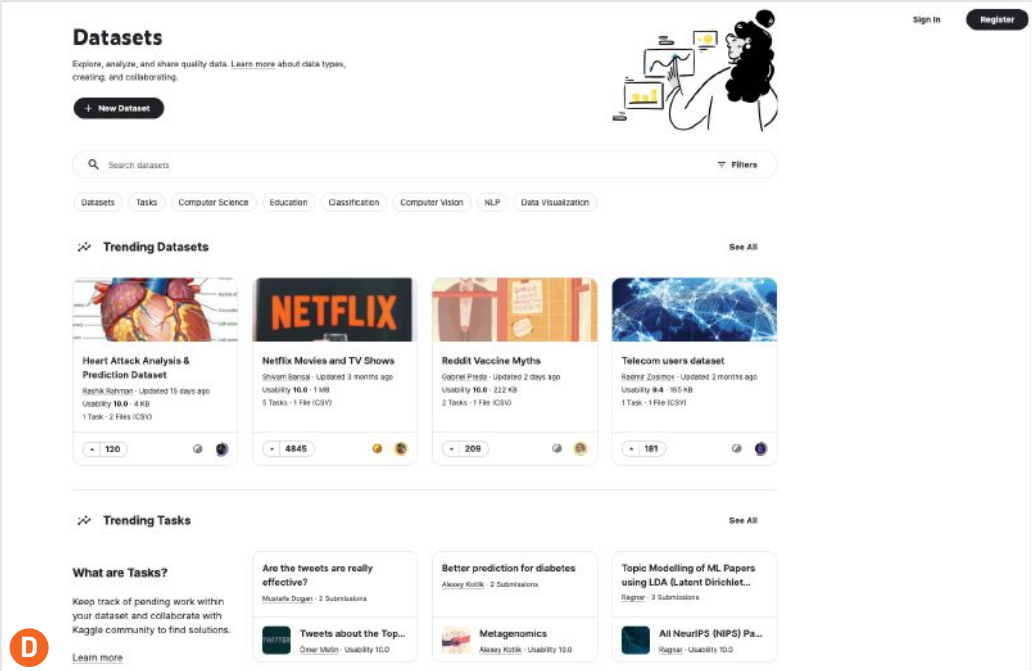
Pre-made datasets exist, but they are often unique to a particular problem or have no real-world application. For example, if someone shared a dataset to classify motion gestures, the motions would be dependent on the type of sensor used and its placement. Data collected from a movement performed with a glove sensor would look different than a similar motion where the sensor is placed at the end of a wand.

However, a few pre-made datasets can help get you started. I regularly use the **Google Speech Commands Dataset** as the foundation for various keyword spotting projects. This dataset consists of several dozen spoken words, each containing over 1,000 audio samples taken from different speakers. I collect additional samples for my target keyword or phrase, such as "trick or treat," and use the pre-made dataset to fill out samples for the "unknown" label.

The **MNIST** dataset contains thousands of samples of the handwritten digits 0–9 (Figure C). This dataset has been used in machine learning research for decades and can be a great starting point for optical character recognition (OCR) systems. Converting handwritten addresses to computer text, for example, helps postal services



WITH TRADITIONAL PROGRAMMING, YOU EXPLICITLY TELL A COMPUTER WHAT IT NEEDS TO DO USING CODE, BUT WITH MACHINE LEARNING THE COMPUTER FINDS ITS OWN SOLUTION TO A PROBLEM BASED ON EXAMPLES YOU SHOW IT.
—HELEN LEIGH



to automate mail delivery systems.

TensorFlow also comes with a number of datasets, including various sound, image, and text samples. Most of these sets, such as MNIST, are created with teaching and research in mind. As a result, you may see limited use for them in real-world applications.

Kaggle is a community of machine learning researchers and practitioners. It is known for hosting frequent competitions that encourage programmers to submit unique machine learning models and algorithms to tackle tough, real-world problems (Figure D). Most of these competitions include pre-made datasets that can be downloaded for experimentation, even after the competition is over.

Finally, many users create or curate datasets on **GitHub**. Some of these require searching the internet for your particular application, but others, such the Awesome Public Datasets repository, provide a list of datasets that are easy to navigate.

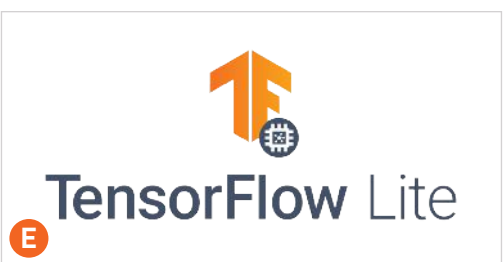
Some datasets, like the **Google Speech Commands**, are created with low-power ML applications in mind. These low-power applications are often reserved for microcontrollers and referred to as **TinyML**. Other datasets, like those used for natural language processing, can take up gigabytes and are usually reserved for larger machine learning applications running on desktops or servers.

TOOLS OF THE TRADE

A few tools and applications have helped make training and deploying machine learning models significantly easier.

Python is the de-facto programming language for curating datasets, analyzing sample features, and training machine learning models.

Several frameworks exist for creating, testing, and deploying machine learning models. One of the most popular frameworks is Google's **TensorFlow**. It works with a variety of languages, but it is specifically targeted at neural networks and deep learning. **TensorFlow Lite** (Figure E)



is a subset of TensorFlow for deploying models to smartphones and embedded Linux devices, like the Raspberry Pi. It contains functions necessary for inference, but it cannot be used for training. You'll also find **TensorFlow Lite Micro** within that framework, which targets microcontrollers.

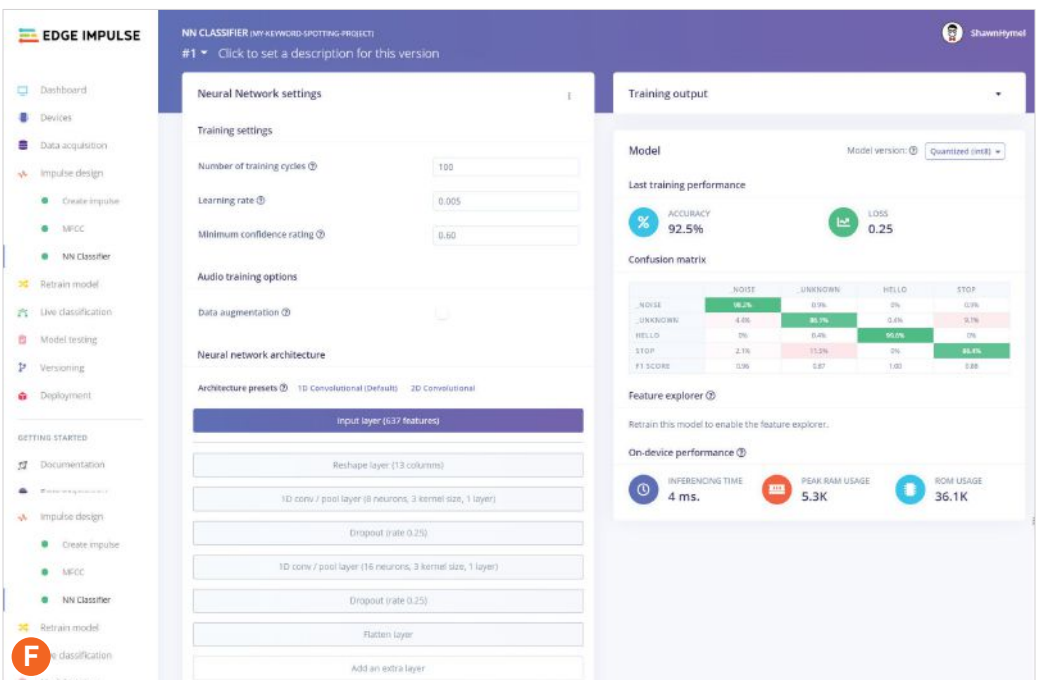
Other frameworks for developing machine learning models include **Sci-Kit Learn**, **Shogun**, **PyTorch**, **CNTK**, and **MXNet**. Apple has their own proprietary system called **Create ML** that allows you to train models and **Core ML** to deploy them to macOS, iOS, watchOS, and tvOS.

Online editors like **Google Colab** provide a development interface and pre-installed machine learning packages, such as TensorFlow. While they may not be ideal for production-level training and deployment, they offer a fantastic learning environment for working with machine learning, especially for creating TinyML models.

Edge Impulse is a graphical online tool that helps you analyze and extract features from your data, train machine learning models, and generate efficient libraries for performing inference on microcontrollers (see "Teach a Faux Nose New Tricks" on page 37). I use Edge Impulse for my motion and audio classification projects (Figure F).

A FEW TOOLS HAVE HELPED MAKE TRAINING AND DEPLOYING MACHINE LEARNING MODELS SIGNIFICANTLY EASIER.

Lobe.ai is another graphical online tool used for training machine learning models, focused on classifying images. It allows you to download trained models in several formats, including Core ML, TensorFlow, and TensorFlow Lite. While these models might work on single-board computers and smartphones (see "Trash Classifier" on page 44), they would require further optimization to function well on microcontrollers.

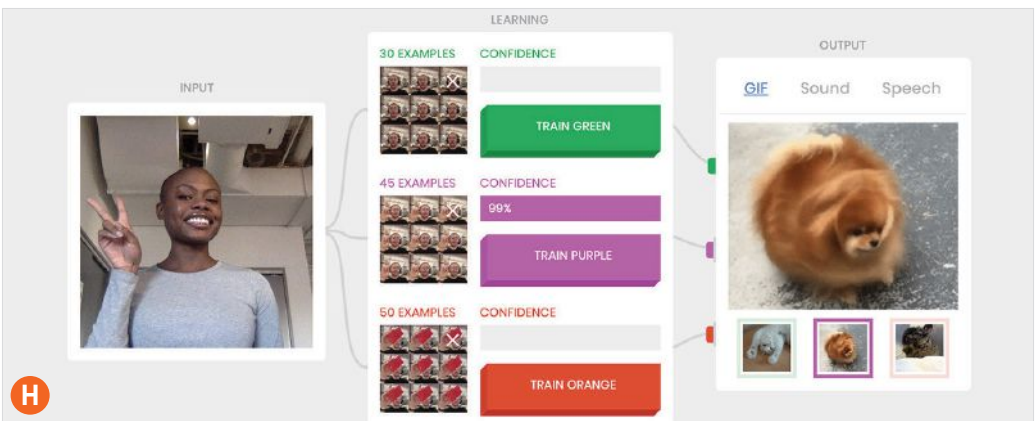


G

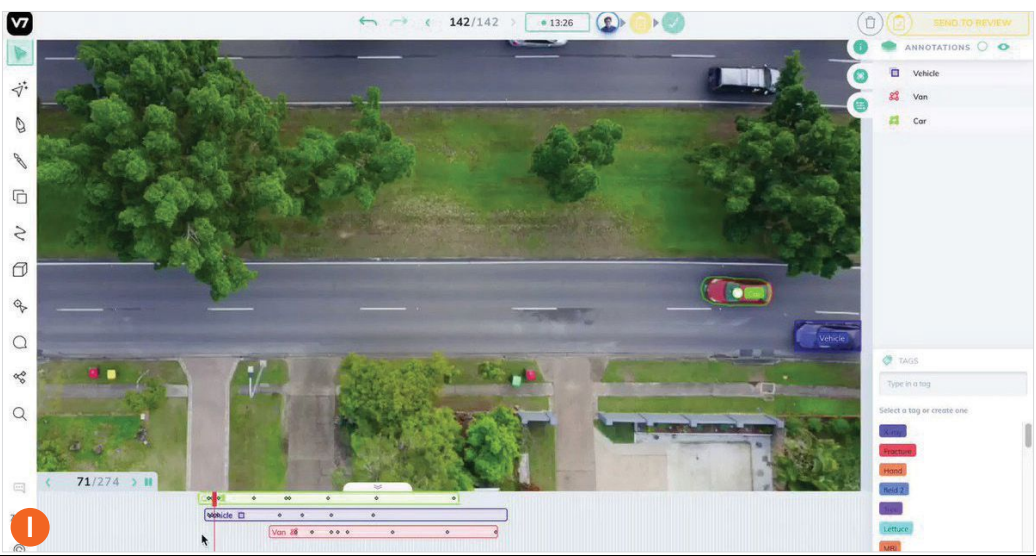
Other similar online tools include **Runway ML** (Figure G), **Teachable Machine** (Figure H), and **V7 Labs** (Figure I). These services make it easy to create models with supervised learning to classify images, video, and sound.

EMBEDDED MACHINE LEARNING APPLICATIONS

Embedded systems are computing devices that serve some specialized purpose, such as handheld calculators, microwave oven controllers, and traffic light control systems. They are usually designed to be easily manufactured, inexpensive, and low-power. Popular maker brands of embedded systems include Arduino (microcontroller boards) and Raspberry Pi (single-board computers). Because many machine learning algorithms are computationally



H



I

Shawn Hymel, Salman Faris

expensive, attempting to run ML on such devices may seem silly. However, there are many potential uses of ML on embedded systems.

One prevalent use of ML on microcontrollers (TinyML) includes wake word detection, which is also known as **keyword spotting**. For example, if you say "Alexa" or "Hey Siri," your phone or nearby smart speaker may come to life, waiting for further instructions (Figure J). The smart speaker uses two types of machine learning. The first kind is TinyML, where inference is performed locally in the speaker's microcontroller to listen for that wake word. Once the speaker hears the wake word, it begins streaming the subsequent audio to an internet-connected server, which performs a much more complex machine learning process known as natural language processing (NLP) to figure out what you're asking.

In addition to speech, we can use machine learning to change the way we interact with electronics. For example, makers **Salman Faris** and **Suhail Jr.** created smart glasses for the visually impaired that would take a picture and tell the wearer what it saw through headphones (hackster.io/makergram/sight-for-the-blind-c1e1b9) (Figure K). We could also use motion sensors and TinyML to detect and classify gestures, giving us the ability to translate sign language or perform actions by drawing shapes in the air with a wand.

In government and business, TinyML has the potential to complement Internet of Things (IoT) ecosystems. With a traditional machine learning architecture, networked sensors would need to stream data back to a central server for analysis and inference. Imagine trying to send sound or image data from tens or hundreds of sensors to a server. This setup could easily consume all of the available bandwidth in a network. Using embedded machine learning, we could have each sensor classify patterns and send the final results to the server, thus freeing up network bandwidth.

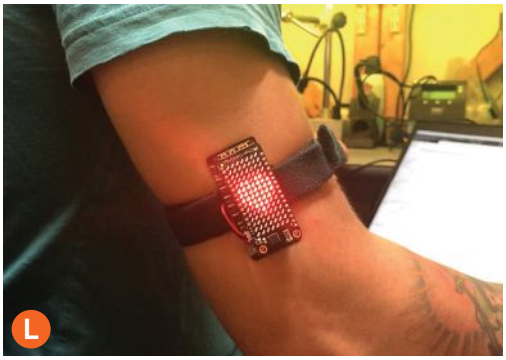
For example, the **Prague Public Transit Company** (DPP) has announced a partnership with Czech-based **Neuron Soundware** to produce audio sensors that listen to the sounds made by each of the 21 escalators in Prague's metro system. The sensors will employ machine learning models to identify potential anomalies

J

K

or unusual sound patterns to determine if certain parts in the escalator require maintenance. This approach is similar to a car mechanic listening to engine sounds to diagnose a problem.

Machine learning can help classify or identify patterns in almost any type of data. As a result, we can use motion and physiological data captured from body-worn sensors to assist with workouts and help predict potential issues. While a GPS unit doesn't need machine learning to tell us how far we ran, what could we use to evaluate our jump shot in basketball? A combination of motion sensors and machine learning has the potential to provide such real-time feedback. Additionally, sensor suites like the **EmotiBit** (emotibit.com) can determine our stress level (Figure L, following page). Coupled with machine learning, this type of data could be used to classify our current emotional state or predict panic attacks before they occur.



Josef Müller, EmotiBit, Shawn Hymel

Pete Warden is the lead developer of the Google TensorFlow Mobile team, which created TensorFlow Lite Micro, a popular framework used in many TinyML applications. He illustrates another potential use of TinyML: low-cost cameras paired with machine learning that can read old gauges and displays. In his article, he mentions how he has worked with "multiple teams who have legacy hardware that they need to monitor, in environments as varied as oil refineries, crop fields, office buildings, cars, and homes. Some of the devices are decades old, so until now the only option to enable remote monitoring and data gathering was to replace the system entirely with a more modern version." Inexpensive, networked cameras could be used as an alternative solution to monitoring such devices without needing to fully replace the system.

Josef Müller demonstrates this low-cost gauge-reading device (github.com/jomjol/AI-on-the-edge-device). He uses an ESP32 and camera to read the numbers on a water meter and report the measurements to a server (Figure M).

Computer vision is a popular application of machine learning. For example, detecting objects on the road is extremely important for self-driving vehicles. Additionally, detecting the presence of people can be used to control lights and HVAC systems in an office building or identifying intruders for a security system. However, image classification and object detection are often computationally expensive. As a result, you will need a powerful microcontroller or single-board computer for many such applications.

TINYML POWER REQUIREMENTS

Machine learning usually boils down to a series of complex matrix operations — in essence, math. Almost every microcontroller and single-board computer can perform math operations, which means embedded systems are generally capable of doing machine learning. Some architectures offer features that make these operations faster, such as floating-point units or special multiply-accumulate instructions. However, the biggest concern is often, "Does my processor have enough power?"

Helen Leigh discusses the computing requirements for TinyML in her "Machine Learning on Microcontrollers" article. In the article, I am quoted saying, "I like to have at least a 32-bit processor running at 80MHz with 50kB of RAM and 100kB of flash to start doing anything useful with machine learning." Let's look at things a little further. I've made a chart with basic guidelines for speed and memory requirements for a few machine learning applications (Figure N). These recommended specifications come from personal experience and are negotiable.

Motion and Distance: Using machine learning to classify various gestures or perform regression on a series of distance measurements requires a relatively low-powered microcontroller. Often, a sample is a few values taken from a sensor at a rate of less than 1kHz. My go-to microcontroller for this application would be an ARM Cortex-M0+, such as the SAMD21 found on the Arduino Zero. However, some makers have successfully employed simple machine learning algorithms on even less powerful microcontrollers, such as the ATmega328P (Arduino Uno) or even the diminutive ATtiny85.

Sound and Voice: Recording and analyzing sounds often require more processing power.

Speed:	20MHz	60MHz	200MHz
Flash:	20kB	50kB	300kB
RAM:	2kB	8kB	100kB

Usable human vocal frequencies generally lie between 300 and 3,000Hz, and a digital microphone needs to sample more than twice that rate to create an accurate waveform of the sound. As a result, processors need to be capable of sampling at a 6kHz minimum, which helps explain why 8kHz is a standard audio sampling rate. While an ARM Cortex-M0+ might work for analyzing audio, I usually reach for an ARM Cortex-M4 instead, such as the one found in the Arduino Nano 33 BLE Sense, to perform inference with vocal and non-vocal sounds.

Vision: Previously, using machine learning to analyze and classify images and videos required powerful desktop computers or servers. Thanks to recent advancements in microcontroller hardware and machine learning libraries, we can now run simplified vision inference applications on low-powered embedded systems. While a Cortex-M0+ or M4 might run a simple vision application, such as classifying single handwritten digits, I have found that more processing power is required to do anything beyond that. For example, the capable ARM Cortex-M7 found on the OpenMV Cam is an excellent place to start. It's capable of running MicroPython and TensorFlow Lite to classify images and perform basic object detection. For higher resolution, faster frame rates, or more complex models, you will likely need to use single-board mini computers (e.g. Raspberry Pi), smartphones, or full laptop/desktop computers.

GETTING STARTED WITH TINYML

Diving into embedded machine learning can seem daunting. The math behind many machine learning algorithms is quite complicated, and the ability to write efficient code is often required to run such algorithms on resource-constrained devices. However, the tools listed above can help make the process easier by handling many of these complexities for you. You can also find numerous resources to help you learn to become a TinyML practitioner, such as courses and books from **Andrew Ng** and **Pete Warden**, as mentioned in Helen Leigh's article.

I also recently released a course on Coursera as a partnership with Edge Impulse: Introduction to Embedded Machine Learning. It works as a complement to the EdX TinyML course, as it provides a shorter, broader overview of embedded machine learning concepts. Additionally, it relies on the Edge Impulse tool for hands-on projects to avoid getting bogged down with TensorFlow Lite Micro versions, settings, and code.

Finally, I recommend checking out the **TinyML Foundation** site, which is a growing community of professionals, researchers, and enthusiasts sharing new developments in the world of embedded machine learning. The site hosts forums, annual conferences, and a weekly virtual talk by prominent members and researchers.

Each month, new hardware, software, and tools enter the market to enable the creation of even more intelligent electronics. Anomaly detection and predictive maintenance promise to save millions of dollars in costly repairs to machinery. Sound classification and person detection can enable a new suite of security-focused IoT devices. Motion and acoustic pattern detection can help researchers track wildlife. Maker projects can take on new dimensions with the ability to respond to gestures and voice commands.

Embedded machine learning is still in its infancy, and it feels much like the early days of personal computing. No one is quite sure where this new technology will take us, but the possibilities are exciting.

SECOND SENSE

Written and photographed by Benjamin Cabé

MAKE A SMART SNIFFER THAT CAN SORT COFFEE FROM TEA, CHOOSE YOUR FAVORITE BOOZE, OR WHATEVER ELSE YOU TRAIN IT TO SMELL!

BENJAMIN CABÉ is a software engineer and IoT jack of all trades at Microsoft, living in Toulouse, France. Find him on Twitter @kartben.

It was a long weekend of May 2020. Like many of my human siblings stuck at home with time on their hands due to an ongoing pandemic, I was busy trying to perfect my bread recipe. In fact, just a few days before, I had ordered a gas sensor (Figure A) that I thought would be ideal to help me monitor my sourdough starter and bake my bread at just the right moment.

And then I thought about it some more. "Surely, this is the perfect excuse for me to finally start learning this *machine learning* thing that everyone's talking about. But … do I really want to bake dozens of baguettes before I have a training set large enough to teach an AI the relationship between the olfactory fingerprint of the sourdough starter and the yumminess of the final loaf? Plus, flour is pretty scarce these days!"

That's how, over the course of the next few days, I ended up building a DIY, general-purpose, artificial nose — one that can smell virtually anything you teach it to recognize! The artificial nose is powered by artificial intelligence — a TinyML neural network that I trained using the free online tool Edge Impulse and then uploaded onto an Arduino-compatible microcontroller.

I learned a lot along the way, and not just about machine learning. From designing my first 3D enclosure to rudimentary fluid dynamics (the airflow within the nose is not exactly optimal), it was the first time I built my own "thing" from scratch, so I'm excited to share it with the *Make:* community. Here are the steps for replicating the build for yourself.

TIME REQUIRED:
A Few Hours

DIFFICULTY:
Intermediate

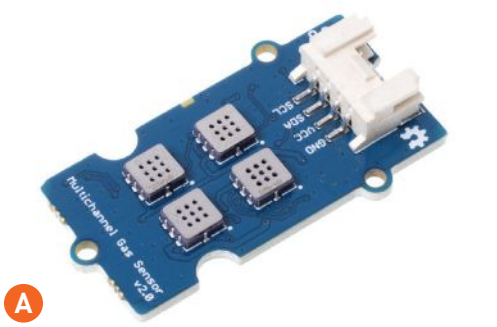
COST:
$80–$100

MATERIALS

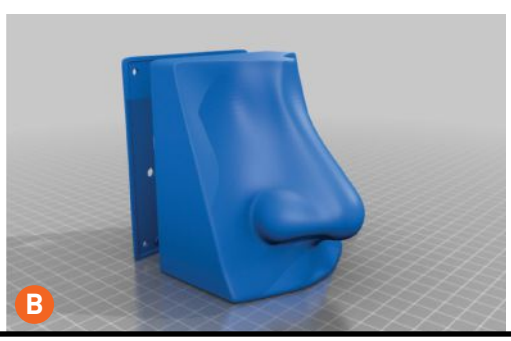
- **Wio Terminal microcontroller board with LCD display** Seeed Studio part #102991299
- **Grove Multichannel Gas Sensor v2** Seeed 101020820, measures nitrogen dioxide (NO_2), carbon monoxide (CO), ethyl alcohol (C_2H_5OH), and volatile organic compounds (VOCs)
- **3D printed nose enclosure** Download the 3D files for free at thingiverse.com/thing:4493907.
- **Grove MOSFET board** Seeed 103020008
- **Cable, 4-pin Grove connector to male jumper wires** Seeed 110990210
- **Fan, 5V DC, 25×25×10mm** such as NMB Technologies 02510SS
- **Fan finger guard, 25×25mm** Sunon FG-2
- **USB-C right-angle cable (optional)**
- **Wio Terminal Battery Chassis (optional)** Seeed 103990564
- **M2 and M3 screws, nuts, and washers**
- **Breadboard jumper wires, 100mm (2)**

TOOLS

- **3D printer**
- **Computer with internet connection** for setup only; not needed for nose operation



A


B

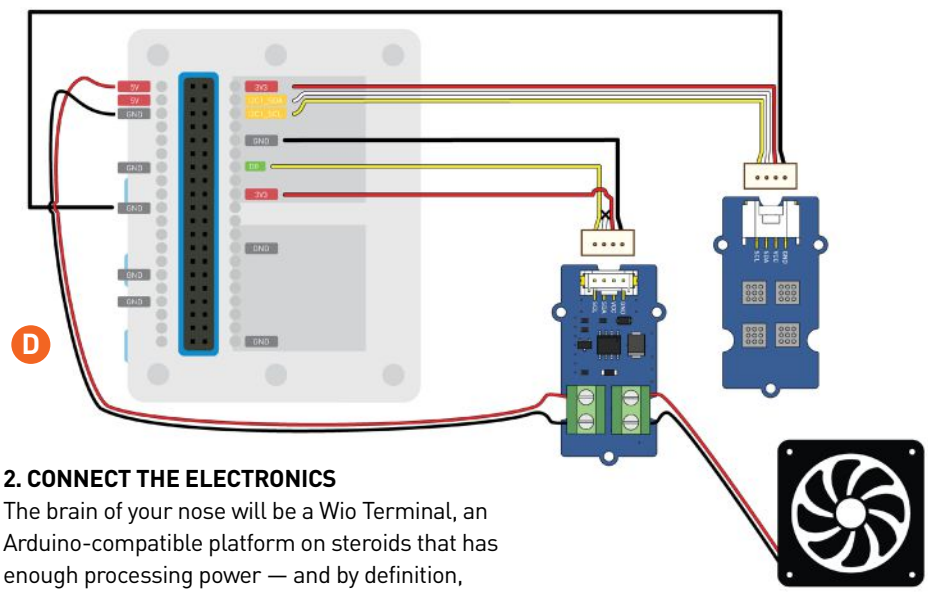
BUILD YOUR ARTIFICIAL NOSE

1. GET YOUR PARTS READY

You can 3D print the nose enclosure from thingiverse.com/thing:4493907 (Figure B). Alternatively, grab your Miniaturizer 3000™, fly to Easter Island, and capture your own 1:100 copy of a moai!

Note that you don't need the handsome enclosure to build the artificial nose, but you certainly need all the electronic components (Figure C on the following page). They're easy to put together; there's no soldering involved at all, just plugging in some jumper wires and connectors.

C



D

2. CONNECT THE ELECTRONICS

The brain of your nose will be a Wio Terminal, an Arduino-compatible platform on steroids that has enough processing power — and by definition, TinyML doesn't need much anyway — to analyze scents in real-time. Since it features an LCD screen, we can have a nice-looking user interface thanks to a dedicated (and pretty complete!) library. You can learn more about it at wiki.seeedstudio.com/Wio-Terminal-LCD-Overview.

The Wio Terminal conveniently features Seeed's Grove-style connectors on its sides, but it makes for a more polished build to use the Raspberry Pi-compatible expansion slot on the back to connect the various sensors and actuators and hide all the wiring within the enclosure.

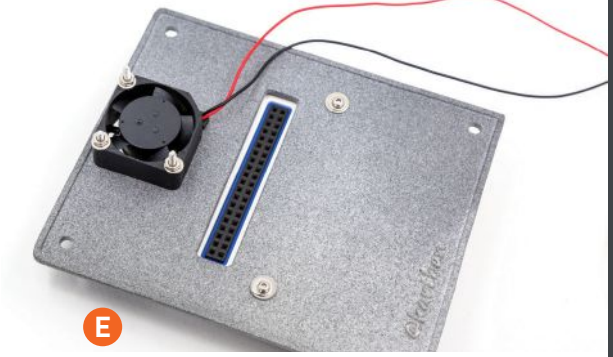
The gas sensor is connected to the I²C interface, and the MOSFET (used to switch the fan on or off and make the nose "inhale") is connected to digital output D0. Follow the wiring diagram (Figure D) to connect the components.

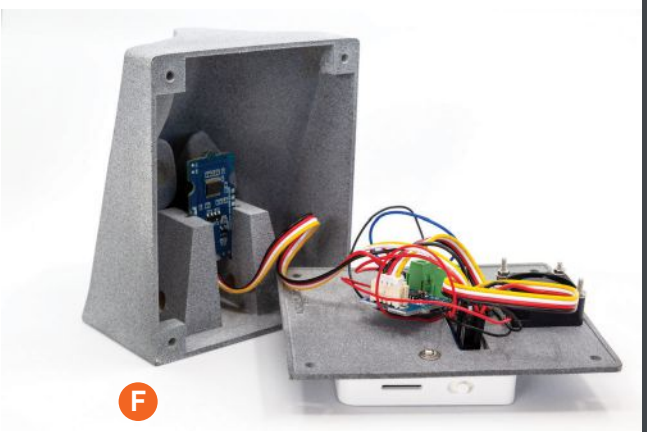
3. ASSEMBLE THE NOSE

Use short M2 screws with washers and nuts to attach the Wio Terminal and the DC fan to the back of the nose enclosure (Figure E).

NOTE: The fan should feature an arrow indicating the direction of rotation and/or airflow, but you may want to briefly power it with 5V to double check that it will make the nose inhale, not exhale!

Slide the multi-channel gas sensor into the dedicated slot with the actual sensors facing the nostrils (Figure F).





4. FLASH THE BRAIN

We're talking about the Wio Terminal, not your actual brain! The quickest way to get an idea of what the nose is capable of is to use a basic model that I've already trained for simple things such as detecting the smell of coffee.

Head over to github.com/kartben/artificial-nose/releases and download the ready-to-use firmware (*firmware.uf2*). Hook up the Wio Terminal to your computer using a USB cable and enter the bootloader mode by quickly toggling the Wio's power switch twice. The device will then show up as an external drive, allowing you to copy the firmware that you've just downloaded onto it.

If you'd rather compile the firmware yourself or, more likely, you'd like to deploy a different AI model altogether, you will just need PlatformIO to compile and upload the updated code, after you've followed the instructions on page 37 to retrain a model for your scents of choice.

START SNIFFING!

After a successful copy of the firmware, the Wio Terminal will automatically restart in Training mode and show you a real-time graph of the various gases being detected by your artificial nose (Figure 2 on page 37).

HOW DOES A GAS SENSOR WORK?

There are several kinds of gas sensors out there, but the technique applied the most commonly involves using a **MOS (metal oxide semiconductor)** which, when heated, starts reacting with the gases contained in the air, causing its **resistance** to change.

From there, chemistry is the least of your concerns as a good ol' **voltage divider** is all you need to measure the resistance — hence gas concentration — in your project. The sensor used in this build even has its own microcontroller that takes care of that for you, and then exposes the readings in digital form, over I²C.

As gas sensors involve heating the metal oxide sensing layer to high temperatures, they typically require a **non-negligible amount of power** to work properly. For example, the 4 sensors used in this build each draw 20mA of current for their internal heating alone! Keep that in mind for battery-powered scenarios.

Furthermore, **it takes a while** — 24 hours, if you follow the datasheet! — for readings to be considered accurate after the gas sensor starts being warmed up, so unfortunately turning it on for only a couple seconds when you need a reading will likely give you poor results.

USER INTERFACE CHEATSHEET

- **Joystick (5-way switch)** — Press in, to alternate between Inference mode (the nose displays its current prediction) and Training mode (raw sensor data is displayed). Push left or right to navigate the various screens of the application.
- **Upper-left button** — aka Button 1 or Key A. Press to toggle the fan on and off.

Press the 5-way joystick button to enter Inference mode. The screen should now indicate what scent is being detected with the highest level of confidence. This basic model can smell coffee, tea, whiskey, and rum.

MORE SENSORS, MORE SCENTS

You might find yourself limited by the capabilities and accuracy of the sensor used in this build. In my experience, it's working fine for telling a handful of scents apart — I first used it to distinguish between a few brands of my favorite spirits. But your project might involve classifying dozens of very similar scents, or scents that don't graph distinctively on this particular sensor.

If that's the case, just start browsing your favorite electronics supplier's catalog, and consider replacing the original sensor with something more geared toward your application. Methane? Ozone? Ammonia? Smoke? Name a compound that's characteristic of the scent you're trying to classify, and there's a good chance a sensor for it already exists on the market.

I SMELL IMPROVEMENTS COMING

There are many things I wish I'd done differently when designing the enclosure. In retrospect, using Blender instead of actual CAD software probably was a bad idea. The nose looks pretty and Blender can do stunning 3D renderings, but tweaking the model is unnecessarily complex.

Somewhat related, the nose cavity and nostrils could probably do a better job at concentrating the airflow straight onto the gas sensors. Or maybe the fan is blowing too strongly, not leaving enough time for the gas sensor to react with the air molecules? Either way, don't hesitate to experiment using the nose ... without the nose!

Another important next step: Use the Wio's built-in Wi-Fi to make the nose a connected, wireless device. The nose could be very useful in remote or difficult-to-access location. For example, in the drop ceiling of an office building's restrooms, to monitor when they need servicing.

Therefore, by the time you're reading this, my GitHub repo will most likely contain all the instructions to directly upload to an IoT platform, in real time, the detected scents. I also plan on making it possible to fully update the nose's neural network remotely, without any human intervention.

Just think of the possibilities if all of us makers start putting together an open source catalog of scents' fingerprints that anyone can cherry-pick from for building their own smell-capable applications!

How about you? Who *nose* what you'll do to improve the build?

A DIY SOLUTION FOR ANOSMIA?

Anosmia refers to a temporary or permanent loss of the ability to smell. Among U.S. adults over 40, some 3% suffer from total anosmia and 23% report some alteration in their ability to smell. Add Covid-19 to the mix, and you get even more people impaired in their day-to-day olfactory life.

Since I started sharing my project, I've heard dozens of stories of folks being impacted by anosmia. Has that bottle of milk gone sour? Does the dog need a bath? Could *I* use a shower? For many, those everyday questions are just impossible to answer. And while a cheap, DIY electronic nose will likely not replace a fully functional human nose anytime soon, it could most certainly become a great companion for those suffering from anosmia.

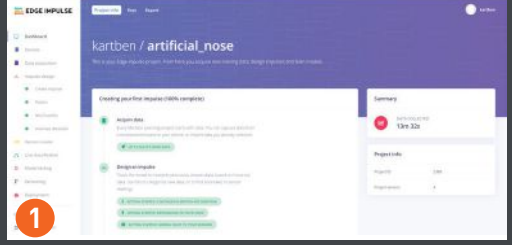
TEACH A FAUX NOSE NEW TRICKS

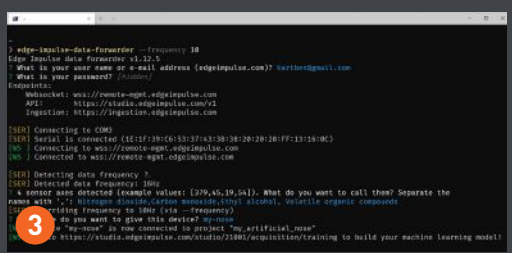
HOW TO TRAIN YOUR ARTIFICIAL NOSE WITH EDGE IMPULSE

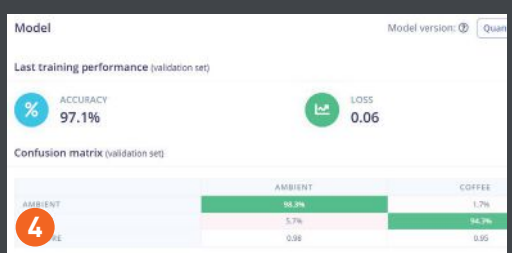
You might not find it incredibly useful to build an artificial nose that can tell coffee and tea apart, and I would agree. But what if you could teach the nose to alert you when your food is burning, or when your favorite fruit is perfectly ripe?

In *Make:* Volume 75, Helen Leigh introduced TinyML as well as Edge Impulse, a great tool that tremendously simplifies the development of AI models for constrained devices like microcontrollers. (Also see "Deeper Learning" on page 22.)

Here's how to use Edge Impulse to train a new model for your artificial nose, so it can start classifying the things *you* care about.







NOTE: Make sure to acquire at least 2–3 minutes' worth of sensor data for each scent, and a roughly equal number of samples for each category, so that your AI model can be trained correctly.

1. Head to studio.edgeimpulse.com/public/2389/latest and clone the base Edge Impulse project into your own account (Figure 1). This will allow you to augment the original dataset with your own classes of scents, or even replace it altogether.
2. Connect the artificial nose to your computer over USB, and make sure it is in Training Mode (Figure 2).
3. Install the Edge Impulse command line tools (docs.edgeimpulse.com/docs/cli-installation) on your computer and launch the data forwarder using the command:

```
edge-impulse-data-forwarder --frequency 10
```

4. Follow the data forwarder instructions in the console to log into your Edge Impulse account. The data forwarder will remotely tether the serial output of your nose to the Edge Impulse project you cloned in the first step, and sensor data will directly appear in your project!
5. Name the 4 sensor axes: **`Nitrogen dioxide`**, **`Carbon monoxide`**, **`Ethyl alcohol`**, and **`Volatile organic compounds`** (Figure 3).
6. Your device should now show up in the Data Acquisition section of your Edge Impulse project. Acquire and label as many samples as you need for the different things you want your artificial nose to detect.
7. Retrain the model based on your newly collected data samples, using the conveniently named Retrain Model action. If you're not happy with the performance of your model (Figure 4), collect some more samples. Rinse and repeat!
8. Finally, use the Deployment menu to export your project as an Arduino library. This will allow you to download a ZIP file containing the neural network you just trained. Replace the *lib/ei-artificial_nose-arduino* source folder of the nose's firmware with the contents of your new ZIP file.
9. Use **`pio run`** to recompile the firmware and upload it to the Wio Terminal. Your nose is retrained.

THE SWEAR BEAR

CUTE BUT EVIL, IT LISTENS FOR CURSE WORDS — THEN INTERNET-SHAMES YOU AND YOUR FOUL MOUTH

Written by Dane Hermse and Nicole Horward

Do you swear too much? You need to build yourself a swear jar. But not just any old swear jar — a hip, innovative, cool, over-engineered digital swear jar. Don't worry, this artificial intelligence-powered, Internet of Things-enabled teddy bear is here to help!

How does it work? Well, the Swear Bear listens to your every word and detects profanity using AI. When you're caught, it instantly tells on you, to a very public data stream, not only revealing the committed crime but also providing time-stamped proof.

BUILD AN EVER-VIGILANT SWEAR BEAR

Before you build, you might like to watch our project overview video at youtu.be/ASSAGECmZt4.

1. SET UP THE HARDWARE

It all starts with a Raspberry Pi computer. The Pi orchestrates the data flow between the different platforms and provides computing power for our dirty language detection.

On top of our Raspberry Pi we connect two

TIME REQUIRED:
A Weekend

DIFFICULTY:
Intermediate

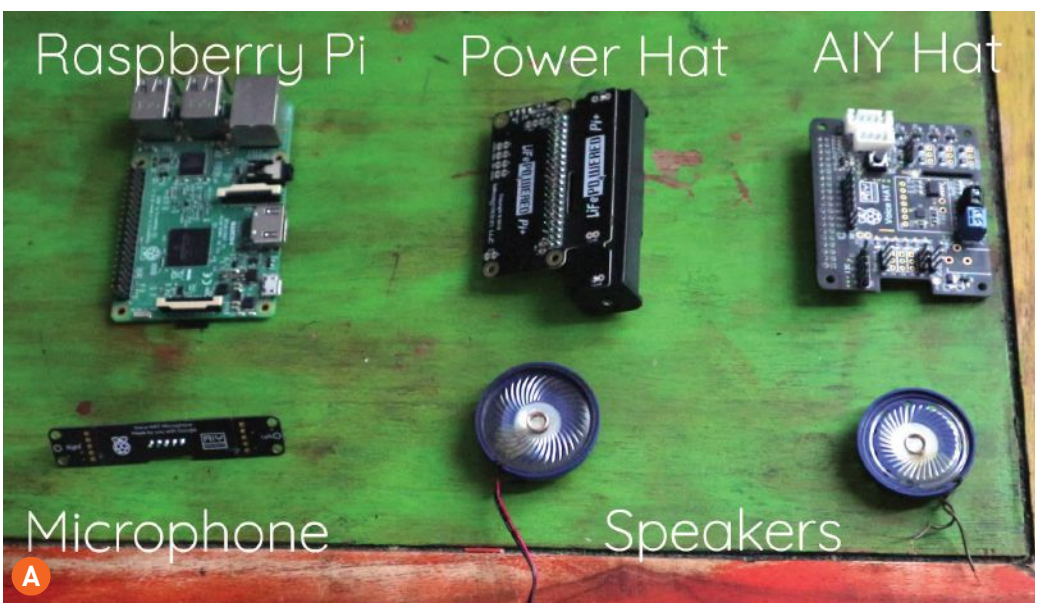
COST:
$200–$300

MATERIALS

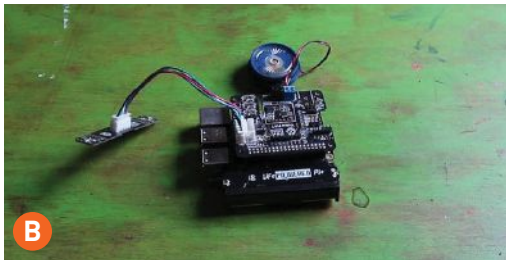
- **Raspberry Pi 3B+ Starter Kit** Amazon #B07BC567TW
- **Google AIY Voice Kit v1** Amazon #B075SFLWKX
- **LifePO$_4$wered Pi+ power supply board (optional)** For our Pi projects we like this UPS and power manager add-on board. It makes The Swear Bear a stand-alone project so you don't have to plug it into the wall. lifepo4wered.com
- **Hookup wire** Amazon B01KQ2JNLI
- **Teddy bear of your choice.** We used Amazon B07VQ92YHV.

TOOLS

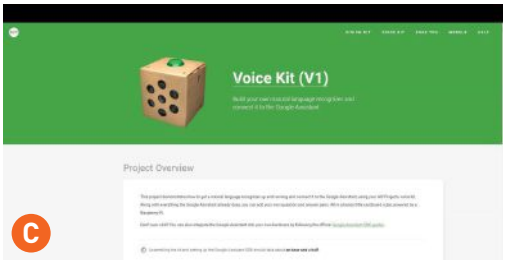
- **Computer with internet connection** You'll need to set up accounts on Google Cloud Platform (cloud.google.com/gcp) and ThingSpeak (thingspeak.com).
- **Scissors, needle, thread, etc.** for putting the hardware inside your bear



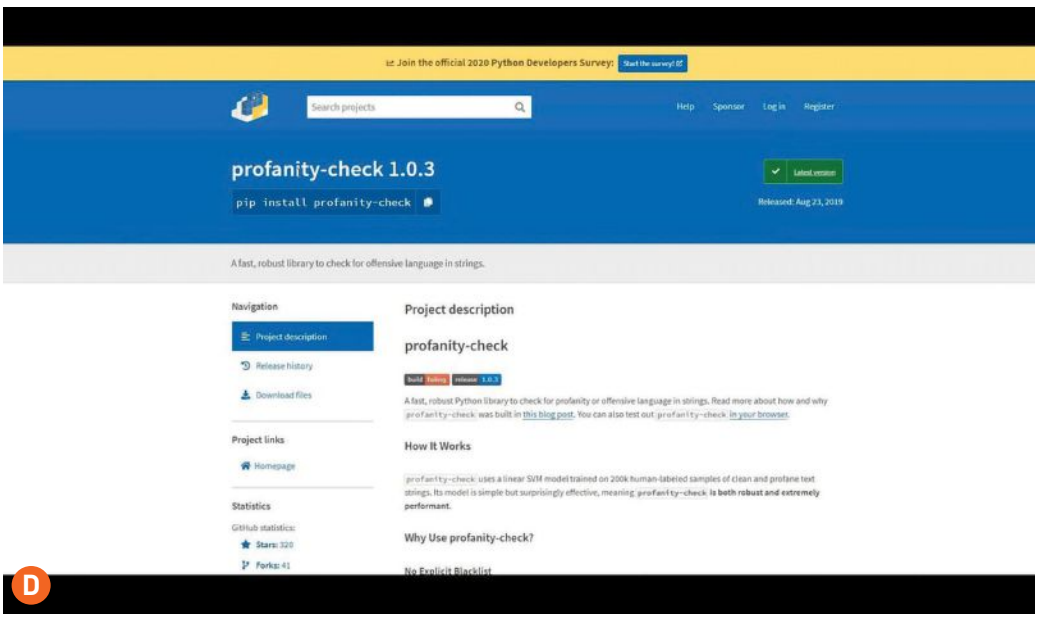
DANE HERMSE and **NICOLE HORWARD** are two makers in Brussels, Belgium, who create tremendously terrible tech and share it on their YouTube channel, 8 Bits and a Byte. youtube.com/c/8BitsandaByte



B



C



D

components that have great support (Figures A, previous page, and B):

- The LifePO4wered Pi+ module takes care of our power needs, and adds much-appreciated portability. To set it up, follow the guide at lifepo4wered.com/files/LiFePO4wered-Pi+-Product-Brief.pdf.
- The Google AIY Voice Kit converts speech to text, and also supplies a microphone for recording and speakers for some interaction. Follow the setup guide at aiyprojects.withgoogle.com/voice.

2. AI AND IOT

Now we move on to the software. To get the ball rolling, follow the guide on how to set up a custom voice interface using the AIY kit at aiyprojects.withgoogle.com/voice/#makers-guide (Figure C). This voice interface allows us to convert the raw captured audio into nice, understandable text.

Next up, detecting profanity in the acquired text. Fortunately, a fantastic Python library exists, with, again, great documentation, at pypi.org/project/profanity-check (Figure D). We use its simple but powerful model to check the converted text for any cussing.

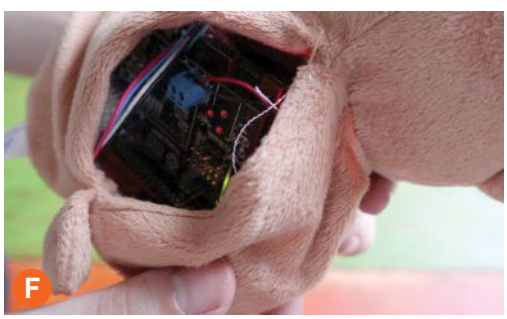
With the above all ready, we're now able to convert speech to text, and reliably catch any obscenity. This is a tremendous milestone: the artificial intelligence part is complete!

You can download our working Python script, *swearBearClean.py*, from the project page at instructables.com/The-Swear-Bear. Here's the full code listing to give you a better idea of what's going on:

```
import os
import requests
import time
import json
```

```
from aiy.cloudspeech import CloudSpeechClient
from profanity_check import predict, predict_prob
import aiy.voice.tts


def say(test):

    aiy.voice.tts.say(test, lang='en-GB', volume=50, pitch=10, speed=100,
device='default')

def sendData(profane, spokenText):

    url = 'https://api.thingspeak.com/update?api_key=YOUR KEY
HERE&field1={0}&field2={1}'.format(profane, spokenText)
    response = requests.request("GET", url)

def getStats():

    say("Fine, calculating")

    # Get the data from ThingSpeak
    url = "Your ThingSpeak URL here"
    response = requests.request("GET", url)
    jsonData = json.loads(response.text)
    records = jsonData['feeds']

    # Stats
    totalGood = 0
    totalBad = 0
    total = len(records)
    lastBad = ''

    # Loop over them, add to stats
    for record in records:

        if record['field1'] == 'True':
            totalBad += 1
            lastBad = record['field2']

        else:
            totalGood += 1

    # Give summary of stats
    say('A total of ' + str(total) + ' records to analyze.')
    say('A whopping ' + str(int((totalBad/total) * 100)) + ' percent contain
a swear.'))
    say("The last one being 'bears suck'. By you. Just now.")

# Instantiates a Google Cloud client
cloudClient = CloudSpeechClient()
say("The bear has awoken!")

while True:
```

```
# Listen and recognize
print('Listening...')
spokenText = cloudClient.recognize()
print(spokenText)

# Check if anything was said
if spokenText is not None:

    # Lets assume the best
    profanity = False

    # Secret data voice module console validator!
    if spokenText.lower() == 'bears suck':
        profanity = True
        getStats()

    # It happened, profanity!
    if predict([str(spokenText)]):
        say('Oh dear!')
        profanity = True

    # Now the cloud must know!
    sendData(profanity, spokenText)
```



3. TEDDY BEAR

Our creation should look as cute as it is evil. And what better fit than a teddy bear?

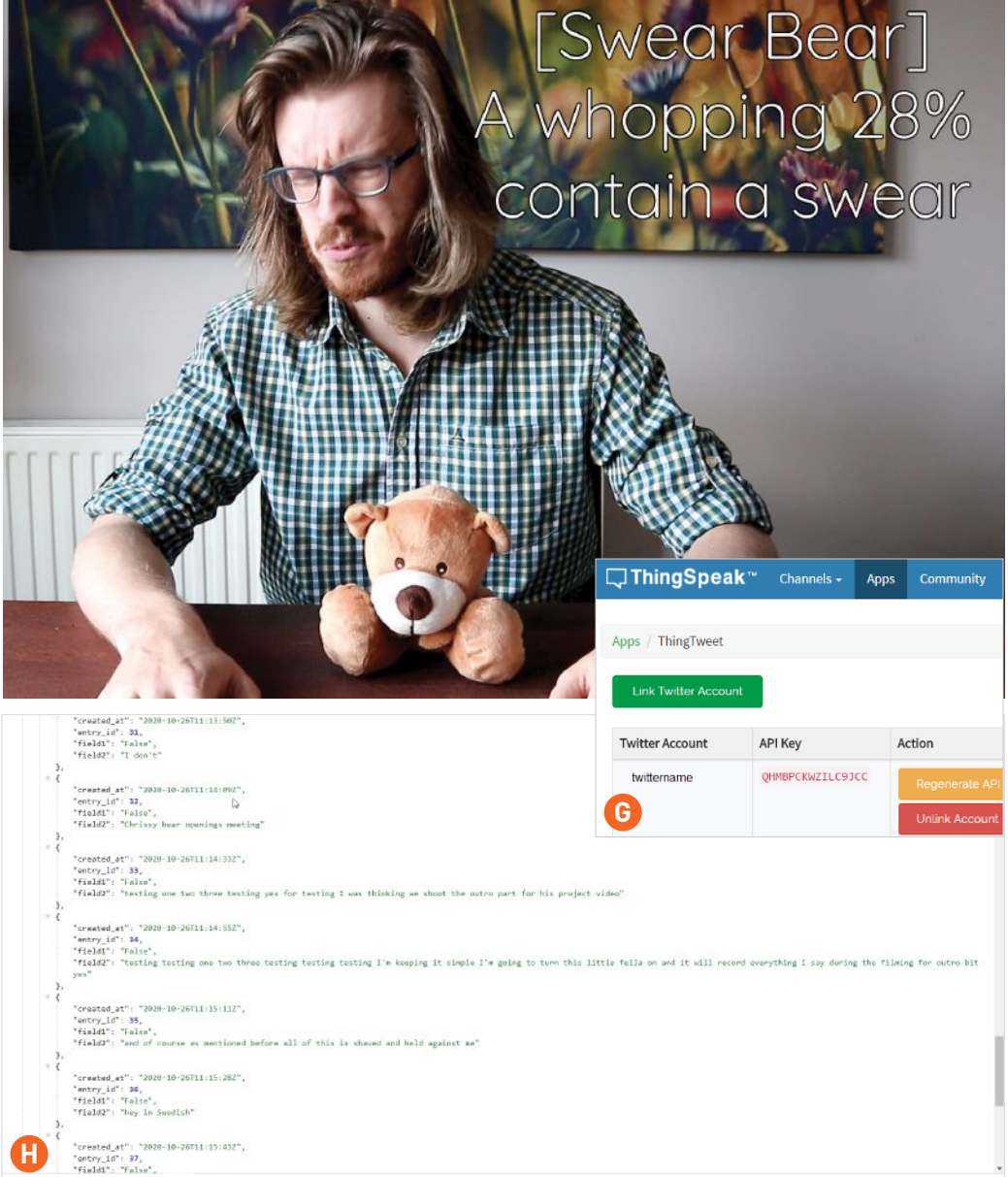
As luck would have it, we found a sweet little teddy waiting for us in the thrift store. It's cute as can be (Figure E) and, after removing the heating pad from its insides, it has just enough space for all the electronics (Figure F). It is quite a fiddly job to get everything to fit, but once done, it does look adorable!

This step is very much optional — you can put these electronics in any vessel you like.

BEAR IS A FOUR-LETTER WORD

To test your Swear Bear, power up the Raspberry Pi and run the Python script. The bear will announce, "The bear has awoken." (We gave it a proper British accent but you can change this in the AIY Voice setup.)

Start speaking. The bear now listens to everything you say, converts your speech to text, analyzes it for any trace of profanity, and makes sure there's a very public record by publishing the data to its ThingSpeak channel. Drop an F-bomb and not only will the bear give you a stiff "Oh dear," but everyone in the world will be able to know about it.



For an added bonus, there's an analytic component — the bear also provides insightful statistics about all it has heard. You're welcome?

EXPLETIVE RETWEETED

Now take it a step further and provide super easy access for maximized shame. The kind folks at ThingSpeak have provided a wonderful guide on ways to trigger action on your data. You can trigger emails, HTTP requests, or IFTTT applets, or, our favorite, your Swear Bear can tweet directly to your Twitter account using their ThingTweet app (Figure G). Follow along at nl.mathworks.com/help/thingspeak/act-on-your-data.html to join the Internet of Bears.

YOU CUSS TOO MUCH

We activated our Swear Bear while filming the ending of our YouTube video, and the results are in (Figure H)! You can browse all the data gathered during this 15-minute session at api.thingspeak.com/channels/1178670/feeds.json?api_key=IY2KW31864YKC5NH.

And now, let's finish this questionable project with a fun fact from the bear itself. Out of 38 speech records analyzed, "A whopping 28% contain a swear."

Well, $#!†.

CLEAN UP YOUR ACT

Written and photographed by Jen Fox

USE LOBE AND RASPBERRY PI TO BUILD A MACHINE-LEARNING RIG THAT DIRECTS ITEMS TO THE RIGHT WASTE BIN

TIME REQUIRED:
60–90 Minutes

DIFFICULTY:
Beginner++

COST:
$70

MATERIALS
- **Raspberry Pi 4 Model B, 2GB RAM**
- **Raspberry Pi Camera Board v2, 8 megapixels**
- **Raspberry Pi Power Supply, 5.1V 3A** with USB-C
- **SD or microSD memory card, 8GB SDHC**
- **Breadboard, half-size**
- **Pushbutton, on-off**
- **LEDs, 5mm (5)**
- **Resistors, 220Ω (6)**

IF YOU CHOOSE TO SOLDER:
- **JST connector (1)** female end only
- **M-to-F jumper wires (2)**
- **F-to-F jumper wires (10)**
- **Heat-shrink tubing**
- **Perma-Proto breadboard PCB, half-size** Adafruit #571

ENCLOSURE
- **Project case, cardboard, wood, or plastic box, approximately 6"×5"×4"**
- **Plastic square, clear, 0.5"×0.5" (2cm×2cm)** You can make it from a plastic food container lid.
- **Velcro, adhesive-backed**

SOFTWARE (PC-SIDE)
- **Lobe** free download from lobe.ai
- **WinSCP or other SSH file transfer method** like CyberDuck for Mac
- **Terminal**
- **Remote Desktop Connection or RealVNC**

Also check out the new Microsoft Machine Learning Kit for Lobe — it has hardware, including a Raspberry Pi 4, for deploying ML models in the physical world. adafruit.com/product/4963

TOOLS
- **Wire cutters**
- **Soldering iron**
- **Helping hands (optional)**
- **Precision knife** e.g. X-Acto knife
- **Cutting mat**
- **Hot glue gun** or other non-conductive glue — epoxy works great but is permanent

JEN FOX is an engineer, maker, and educator. After dabbling in dark matter, she settled into engineering and inventing to solve problems related to climate change and social justice. She works full time at Microsoft as a senior product manager for maker-related things and runs her company, FoxBot Industries, on the side.

One consequence of our efforts to build better, more sustainable systems is having to learn new rules for sorting thousands of objects into trash, recycling, compost, hazardous waste, and more. We are inevitably bound to make mistakes and put things in the wrong bins.

Fortunately, as makers, we have the power to build tools that make things easier. Enter: the Raspberry Pi Trash Classifier (Figure A)!

For this system, I trained a model using **Lobe**, a free, no-code app for building custom image classification models, then exported the model in TF Lite and deployed it on a Raspberry Pi 4 via Python. When activated, its Pi camera will take a photo, do inferencing on the image, and indicate the resulting trash category with an LED.

I built this project to show and teach others how to prototype a beginner-friendly custom classifier using machine learning (ML) on an edge device, like the Pi. This article will focus on ML and Lobe so you can build your own ML models. The full project write-up for the Trash Classifier can be found at Adafruit: learn.adafruit.com/lobe-trash-classifier-machine-learning. Additional resources are available on the Github repo: github.com/microsoft/TrashClassifier.

WHY ML?

An accurate trash classifier must recognize objects at different angles, lighting, and states of crumpled-ness, and also recognize patterns across objects and materials. It's not feasible to

write a program that recognizes every piece of cardboard in every possible orientation. Instead, we want our program to recognize key features of cardboard to generalize to new images.

This is what ML is good at: learning key features from a training set and using these features to identify, or **classify**, new data.

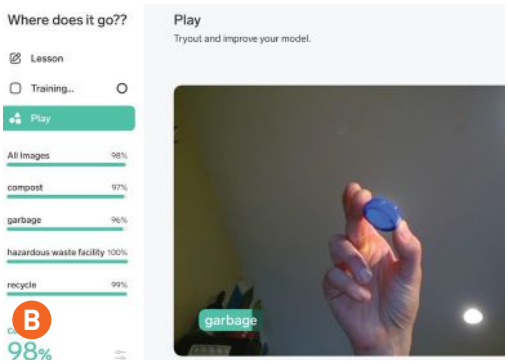
With any ML project, it's important to consider the consequences of false positives and false negatives. For this prototype, I wanted to reduce false positives for compost and recycling (e.g. accidentally recycling non-recyclables). It was also important to reduce false negatives for hazardous waste (e.g. classifying a 9V battery as recyclable). This meant testing my model for accuracy across each of the categories, seeing where and why it was wrong (e.g. in a specific hand orientation, the model was classifying blue bottle caps as batteries), and adding more images to improve accuracy.

ANALYZING THE RESULTS

After taking and labeling about 1,500 photos in Lobe for about a month, the deployed ML model worked great on recyclables: it recognized different types of cardboard, including different sizes, shapes, and colors that it had not been trained on.

It also worked well on plastic (Figure B), food containers, and most compost. The category that needed the most improvement was hazardous waste, likely because this category had the fewest photos.

I was delightfully surprised at how accurate the model was given that I trained it on my laptop camera (instead of the Pi camera) with objects in different backgrounds and lighting (Figure C). Going forward, I would retrain the model using photos taken from the Pi camera and I would leverage friends to help contribute photos to include more esoteric items.



GETTING STARTED WITH LOBE

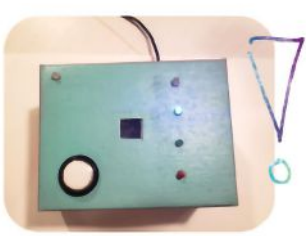
Lobe is a free, easy-to-use tool to bring your machine learning ideas to life! (Full disclosure: Microsoft makes Lobe, and is also my employer.) Show it examples of what you want it to do, and it automatically trains a custom ML model that can be exported for edge devices and apps. It currently supports export in TensorFlow (TF), TF Lite, TF JS, ONNX, and API (and CoreML via MacOS). It doesn't require any experience and you can train locally on your computer.

Here's a quick overview on how to use Lobe. First, download Lobe from lobe.ai. Open and create a new project.

Take or import photos and label them into appropriate categories. Keep your labels simple, as these are what the Lobe model returns when it makes a prediction on a new image (also known as **inferencing**).

There are three ways to import photos:

- Import individual photos from your computer
- Take photos directly from your computer webcam or connected camera, or
- Import a dataset of photos from your computer. The photo folder name will be used as the category label (Figure D), so make sure your photos are organized and your folder labels match existing labels!



I ended up using all 3 methods on the Trash Classifier, since the more photos you have, the more accurate your model is!

Check the accuracy of your model by testing it with your webcam and new images. Change distances, lighting, hand positions, etc. to identify where the model is and is not accurate. To improve accuracy, give feedback to the Lobe model by selecting whether its classification is correct or not — if not, you can relabel the photo. Add more photos to each label as necessary.

Before importing photos, make a list of all the categories you'll need and how you want to label them: "garbage," "recycle," "compost," etc. If you start with my Trash Classifier code, you can use the same labels to reduce the amount of code you need to change.

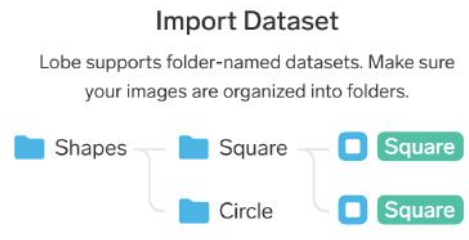
Include a category for "not trash" that has photos of whatever else might be in the photo — your hands and arms, the background, etc. (Figure E). This applies to other models as well — include this "empty" category to reduce **false positives** (images your model misclassifies).

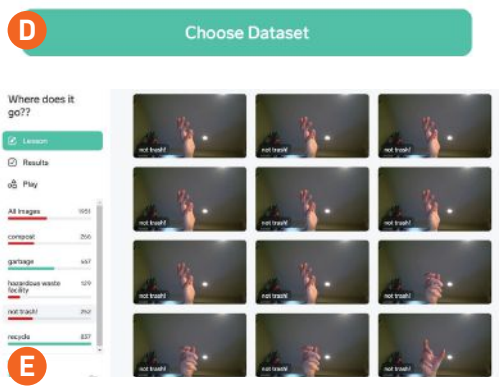
If possible, take photos from the Pi Camera and import these into Lobe. This will greatly improve the accuracy of your model! ML models aren't like human brains — the images we use to train the model really change how it learns features. For example, if we trained an ML model to recognize chairs but only took pictures of 4-legged chairs, the model might not recognize a 3-legged chair. Since every camera has slightly different coloring, pixel size, etc., training the model on images from the same camera you'll use for inferencing makes it more likely that the model will make accurate predictions.

When you're ready, export your Lobe ML model. For the Trash Classifier, or Pi projects in general, I'd recommend using the TensorFlow (TF) Lite format.

BUILD YOUR BOX AND LET IT RIP

Once you've got your own Lobe ML model exported, jump over to the project walkthrough at Adafruit for steps on getting a Lobe TF Lite model running on a Raspberry Pi using Python and then building the physical trash classifier itself. Also check out the handy Lobe Python repo for general support: github.com/lobe/lobe-python.





You can build it on a breadboard or solder it to a PCB. I recommend using female-to-female jumper wires and heat-shrink tubing to connect components to the Pi GPIO pins.

Then you can adapt your classifier for other types of image classification models in Lobe (e.g. counting bees in a hive, tracking wildlife, or checking if your kids ate all the leftover cake). The options are limitless. Have fun!

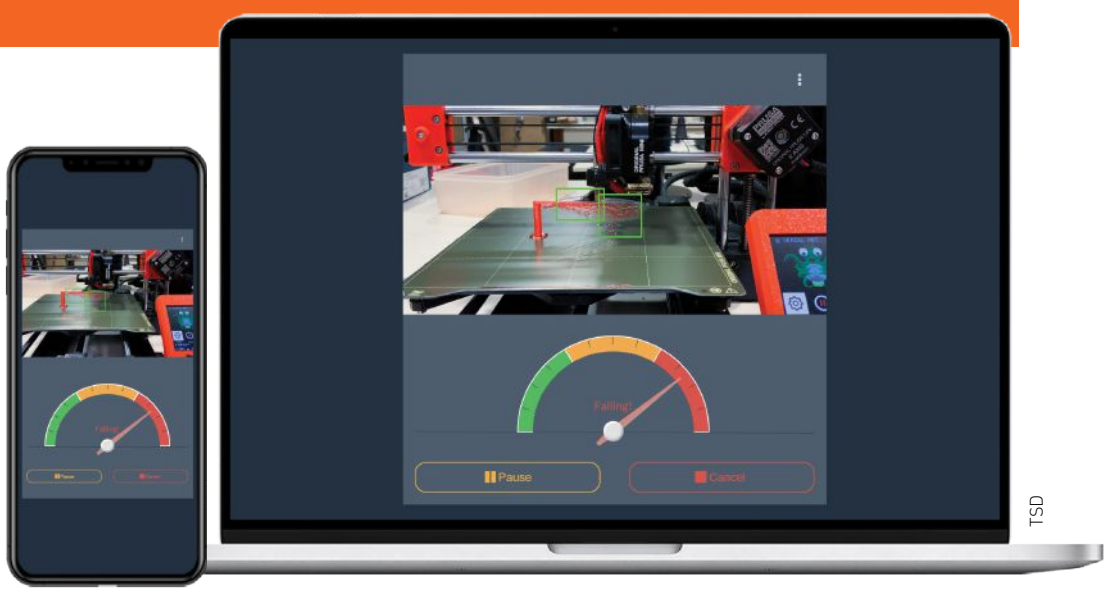
Questions, comments, or more info?

Feel free to reach out: AskAMaker@microsoft.com, or connect with the Lobe team: LobeAI@microsoft.com

THE SPAGHETTI DETECTIVE

Written by RabbitAmbulance

LET AI WATCH YOUR 3D PRINTER'S EVERY MOVE AND ALERT YOU WHEN IT FAILS!



Can't sleep well when your 3D printer is printing at night? Worried your print may fail when you're not home? The Spaghetti Detective (thespaghettidetective.com) constantly watches your prints when you can't. It's an algorithm that uses deep learning to analyze webcam images and alerts you when your print shows signs of failing.

We designed The Spaghetti Detective (TSD) as a plugin for the popular OctoPrint web interface for 3D printers. It comes with a host of other features that let you print remotely with peace of mind:

- Remote webcam access. No port forwarding or VPN required (merged from the OctoPrint Anywhere plugin).
- Full remote control of your printer. Pause or cancel the print. Check and set heater temperatures or turn them off. Move the print head.
- Upload G-code files and start a print remotely.
- Easy share of your webcam streaming or time-lapses with your friend.
- Alert and print job notification (when a print is done or canceled) via email, text, Telegram, PushBullet, and Slack.

You can set up TSD in 56 seconds from your OctoPrint web app or mobile app (Figure A). Use it for free 10 hours a month with JPEG video streaming, 50MB of secure tunneling to your

OctoPrint, and failure alerts. Or for a small fee you can get unlimited print hours watched by The Detective, unlimited secure tunneling, premium H.264 streaming up to 25fps, and more. You just need OctoPrint running locally (commonly on a Raspberry Pi); our servers handle all the rest.

But maybe you'd rather do it all yourself? You can. The Spaghetti Detective is actually open source — from the plugin to the backend to the algorithm, every single bit (pun intended) of it.

SETTING UP YOUR OWN TSD SERVER

The Spaghetti Detective server needs to run on a real computer, not a Pi, unfortunately. The Raspberry Pi (yes even the Pi 4B) or Latte Panda are just not powerful enough to run the machine learning model. An old PC is perfect — if it's got at least 4GB of memory, you'll probably be fine — or you can use a 4GB Nvidia Jetson Nano!

- **If you're on Linux, Mac, or Windows**, get to the TSD source code repo at github.com/TheSpaghettiDetective/TheSpaghettiDetective and follow the instructions to set up your own server.
- **If you're on Windows 10, Windows Server, or Jetson Nano**, follow the links there for separate tutorials. In this article we'll show you the Nano setup.

RUN THE SPAGHETTI DETECTIVE SERVER ON JETSON NANO

Thanks to the contributions of Raymond H. (raymondh2), high school student Alfredo Colas (LyricPants66133), and others, you can now easily run a TSD server on a 4GB Jetson Nano (Figure B). Here's how; check back at github.com/TheSpaghettiDetective/TheSpaghettiDetective/blob/master/docs/jetson_guide.md for updates.

1. SOFTWARE REQUIREMENTS

The following software is required:

- **JetPack SDK** If you already flashed software on your SD card, you'll have to replace it with this one, developer.nvidia.com/embedded/jetpack. Note that slow download of the software from Nvidia is normal.
- **Flashing software** We like Balena Etcher, balena.io/etcher.
- **SD Card Formatter** sdcard.org/downloads/formatter

TIME REQUIRED:
2 Hours

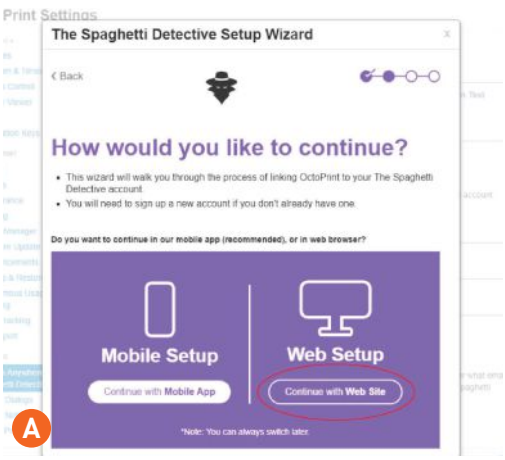
DIFFICULTY:
Intermediate

COST:
About $200

MATERIALS

- **Nvidia Jetson Nano 4GB with SD card 64GB or more, Class 1A or higher, or an old PC (optional)** for running your own TSD server
- **Raspberry Pi mini computer running OctoPrint** free from octoprint.org
- **Web camera** such as the Raspberry Pi cameras, Logitech C270, C525, C910, C920, etc.
- **LAN cables (optional)** Nice to have, but Wi-Fi will do.

When **RABBITAMBULANCE** is not helping out around The Spaghetti Detective Discord, he can be seen ferrying his furry ill brethren, employing the principles of hopping instead of rolling!



IMPORTANT: Before you flash the new software on your SD card, you'll have to fully format it first, so make sure you have backed up anything important on an external device.

2. START THE SERVER

Install the entire server, all in one command!

a. Run the command:

```
git clone https://github.com/
LyricPants66133/Jetson_TSD_Fullinstall.
git && sh Jetson_TSD_Fullinstall/
jetson_TSD_install.sh
```

NOTE: The last JetPack SDK version this has been tested on is jp45. If you successfully run this on a newer version, please send a message to the official Discord and mention @LyricPants66133.

b. Boot up your favorite entertainment streaming service and get a hot drink. This can take a while.

c. Reboot your Jetson to make sure everything is running well: `sudo reboot`

3. CONFIGURE YOUR SERVER

Now follow the general setup instructions at github.com/TheSpaghettiDetective/TheSpaghettiDetective#basic-server-configuration. You'll login as a Django admin (Figure C), then configure an SMTP email server. Your server is done!

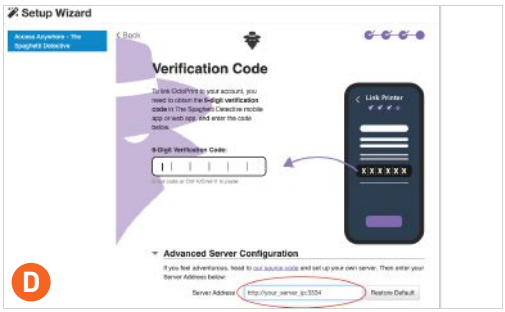
4. CONNECT IT TO OCTOPRINT

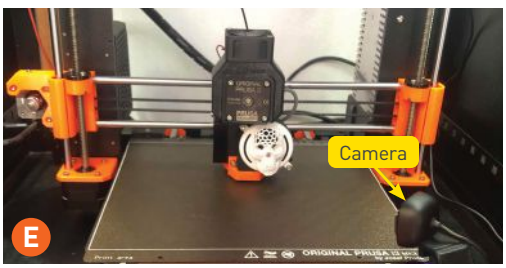
If you haven't already, get OctoPrint running on a Raspberry Pi at octoprint.org/download, and install the OctoPrint TSD plugin following the instructions at github.com/TheSpaghettiDetective/OctoPrint-TheSpaghettiDetective.

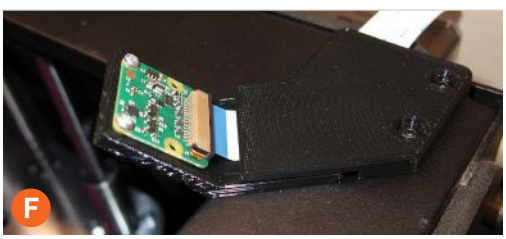
Finally, you'll configure OctoPrint to use your server instead of ours. Follow along again at github.com/TheSpaghettiDetective/TheSpaghettiDetective to add your 3D printer to your TSD server, get the 6-digit verification code, and enter that code into the OctoPrint TSD plugin, then change the TSD server address to your own (Figure D). Give your printer a fancy name and enjoy The Spaghetti Detective!

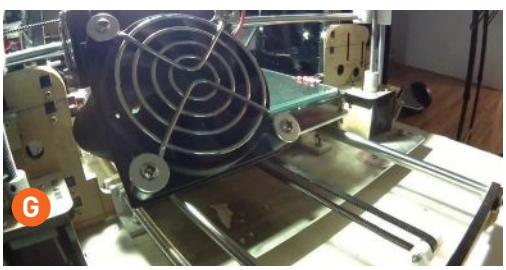
CAMERA AND LIGHTING

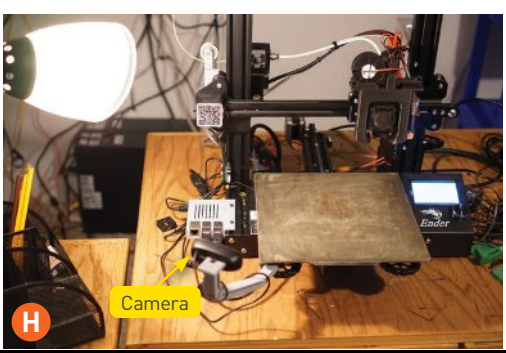
What your camera sees is what The Detective sees — if she sees everything clearly, she has a

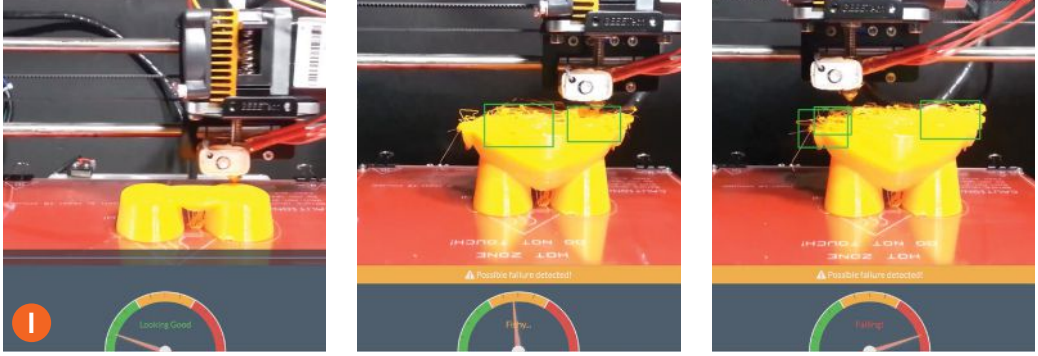












Looking good ... Fishy ... Failure!

significantly higher chance of detecting a problem when it occurs. How you set up your camera has a huge impact on her success. Top tips:

- **Clear view** Put the camera where it can see most of the print volume and not much background clutter. You can mount it to the print bed (Figure E) or to the frame (Figure F) somewhere it won't be blocked by the extruder (Figure G). Avoid mounting to the extruder.
- **Camera angle** The Detective can work with almost any camera angle, even straight down, but she does a slightly better job at detecting failures when the camera is mounted on the side, slightly looking down about 15 degrees.
- **Lighting** Eliminate shadows and backlighting — use an LED strip all around the box frame, or just a lamp on the same side as your camera (Figure H).
- **Focus** If your camera doesn't auto-focus, adjust it manually for a sharp image. More tips at thespaghettidetective.com/docs/optimal-camera-setup.

SOMETHING'S FISHY!

The Spaghetti Detective is still in its early stage and its algorithm is not perfect yet. It may sometimes give false alarms, or miss print failures. So we compiled a Spaghetti Gallery at app.thespaghettidetective.com/publictimelapses to show you how some camera setups work better than others (Figure I).

ALERTS AND PAUSING

The Advanced Settings tab (Figure J) tells The Detective how to react when a failure is detected:

- **Just notify me** via email and text. The Detective will not freak out, just calmly send an email (and a text if your phone number is on file).
- **Pause the print and notify me** via email and text. The Detective will be more proactive: When she thinks she sees a crime scene, she'll pause your print to prevent further damage.

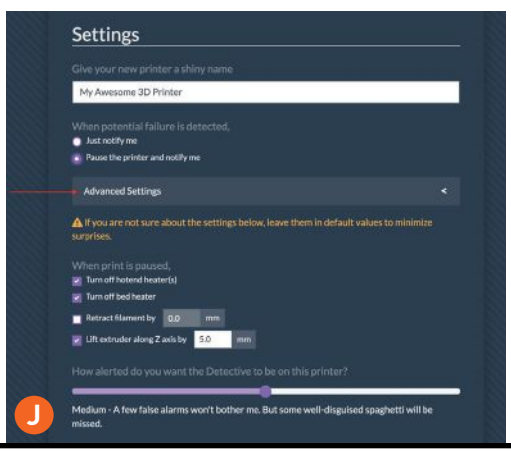
Because your print may be paused a long time before you get to it, we give you options to prevent further damage to your printer as well:

- **Turn off hotend heater(s).**
- **Turn off bed heater.** In both cases, she will also make sure the heaters are warmed up again before resuming motion. You may not want to cool the bed when printing with ABS, as this will almost certainly cause warping.
- **Retract filament** by XX mm.
- **Lift extruder along Z axis** by XX mm. Both of these are also reversed before resuming.

ALERTNESS

Start with the default setting, Medium. If after a few prints, The Detective is annoying you with too many false positives, set the slider lower and you should receive fewer alerts. Conversely, if The Detective has missed some delicious spaghetti, you may want to set the alertness higher.

Finally, you'll find tech support and PayPal donation links at thespaghettidetective.com/docs/open-source — we would really appreciate it if you can buy us a couple coffees each month!



Motion Animated NeoPixel Skirt

Written and photographed by Martin Oehler

Make a light-up skirt with 120 hidden full-color LEDs that respond to your movements

Addressable LEDs strips are a great and easy way to add complex multicolor light effects to any sort of object. Inspired by skirt projects from *Make:* authors Debra Ansell (geekmomprojects.com) and Becky Stern (beckystern.com), this project integrates 120 LEDs into a skirt while aiming to make it both as shiny and as wearable as possible. The LEDs are installed only at the front of the skirt to allow the wearer to sit down or use the restroom without potentially breaking the electronics. The LEDs and cables are completely removable for washing.

The sewing was done by my wonderful girlfriend, Sabine. She has been sewing her skirts and dresses for decades now and creating her own sewing patterns. But the LED integration can be adapted to many other patterns, and we've added some tips for sewing your own. For this you'll need some experience in sewing and pattern construction.

But you can also add these LEDs to an existing skirt — maybe one you don't wear anymore — and add a semi-transparent fabric on top to get the same diffusion effect.

All electronics are hidden either in the skirt or in a pocket at the back, so the skirt appears as normal as possible when the LEDs are switched off. By adding a 6-axis accelerometer/gyroscope, we can trigger effects by movement or rotation in any direction, so there are many options to add unique lighting patterns to the LEDs in the future.

Building this skirt will teach you the usage of addressable LEDs in a simple matrix arrangement (6×20), and you will get a lot of attention wearing it. This is especially the case in school environments (e.g., for STEM teachers) to motivate kids doing LED projects and coding.

Uploading new code to the microcontroller is very easy, so you can continue to learn and

TIME REQUIRED:
A Weekend

DIFFICULTY:
Intermediate

COST:
$50–$70

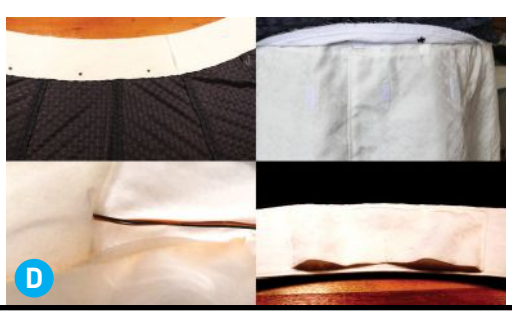
MATERIALS

- **Skirt** You can sew a panel skirt or modify an existing skirt.
- **Fabric, semi-transparent** For the outer layer. If you're sewing your own skirt, you'll also need a white cotton mid-layer, and a white polyester lining fabric. The amount depends on your sewing pattern and size (see Step 1).
- **Hook-and-loop tape, 1cm wide (48cm total length)** aka Velcro tape
- **WS2812B RGB LED strip, 60 LEDs/meter, white, IP30 type (2 meters)** Various IP30 clones are available at Amazon, eBay, or AliExpress. Or you can try the weather resistant IP65 version, Adafruit #1138, adafruit.com.
- **Adafruit QT Py microcontroller board** Adafruit #4600
- **LSM6DSO32 6-DOF accelerometer and gyroscope** Adafruit #4692
- **4-wire cable, JST SH female connectors, 50mm long** Adafruit #4399
- **Hookup wire** in three colors
- **Heat-shrink tubing**

TOOLS

- **Sewing machine**
- **Soldering iron and solder**
- **Wire cutters** I used side-cutters.

MARTIN OEHLER lives in northern Germany, where he tinkers with addressable LEDs, 3D printing, and Raspberry Pi projects. Some of them are published at instructables.com/member/maketvee.



improve your coding skills by adding new light effects.

1. CHOOSE YOUR FABRICS

The skirt itself consists of 3 layers:

- **OUTER LAYER:** Blue top layer of semi-transparent polyester (Figure A), for diffusion. The darker the color, the better it hides the LEDs when switched off. Some structure in the textile gives a nice additional diffusion effect.
- **MID LAYER:** White cotton holding the LEDs and Velcro and guiding the cable. Also used for the waistband.
- **INNER LAYER:** White lining fabric (polyester) to reduce resistance, especially while walking, so the mid layer is less folded.

2. SEW THE SKIRT

A panel skirt design based on 6 panels (Figure B) is attached to a thick waistband. The waistband lays flat and close fitted on the waist. There's a

zipper on the side. The pocket for the electronic stuff is attached to the waistband on the back of the skirt.

A panel skirt design is a good choice for this project; it's easier to handle smaller pieces of fabric while preparing them for the LEDs. The outer layer should be about 1cm longer than the mid and inner layers, to hide the white fabric at the skirt's hem when walking.

3. VELCRO, CABLE SLOT, AND POCKET

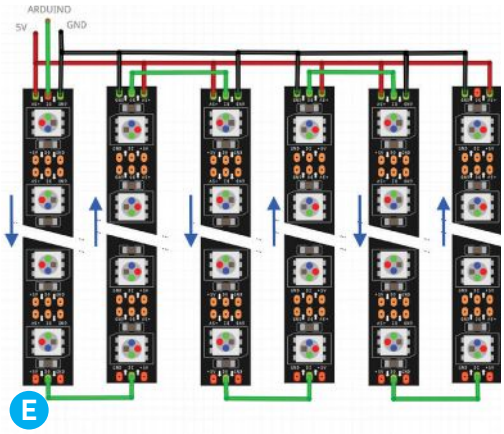
For fastening the LED strips to the mid cotton layer, you'll cut 24 pieces of Velcro 2cm×1cm. The soft, fuzzy "loop" side is sewed to the mid layer, 4 Velcro pieces per LED strip (Figure C). Using the soft side here is important to avoid scratching the fabric during washing.

Depending on the length of your skirt, you can decide how many LEDs you want to integrate. We used 20 LEDs per strip, so each strip is about 33cm long. Two Velcro strips are placed at top and bottom, the other two in between (under the 7th and 13th LEDs, in our case). You can use the strip as a reference for positioning the Velcro on the fabric. We attached the Velcro before assembling the full skirt, but it's also possible to add it afterward if you use an already existing skirt.

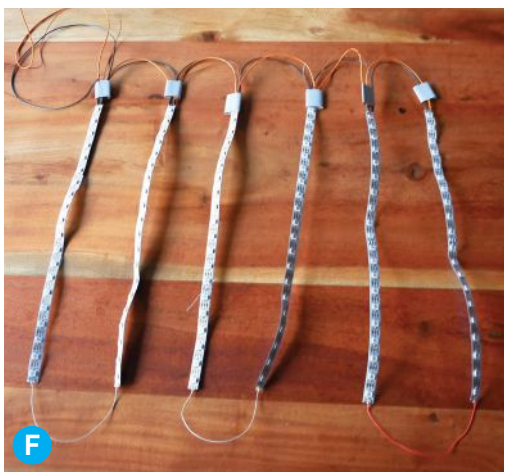
The waistband is designed as a dual-layer white cotton structure with access from the mid layer to hide and guide the cables to the back (Figure D). This also includes the pocket for the electronics and battery at the back, which has a buttonhole to access the cable from the inside. Small black snap fasteners fix the two cotton layers at the bottom to hold the cables in place.

4. PREPARE THE LED STRIPS

As mentioned, depending on the skirt size, the number of strips and LEDs per strip may vary. To keep the wiring simple, the easiest way is to use a serpentine layout, so every output of a strip is connected to the input of the next strip (Figure E). This allows the controller to drive all strips with only one I/O pin. The only drawbacks are additional connections at the skirt's hem, and a zigzag placement for the strips, which can be easily compensated later in the code. Be sure you



E


F

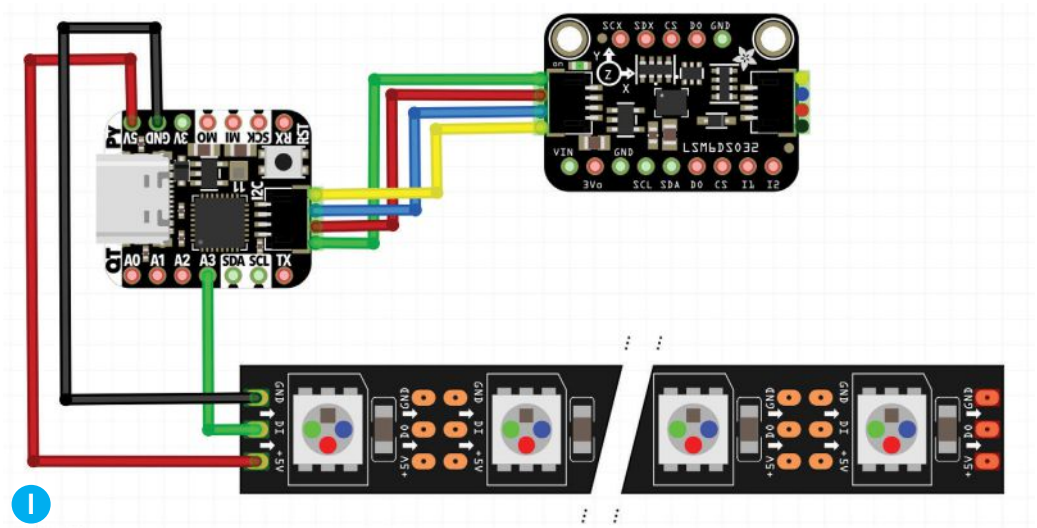
can hide these connections at the hem, either by a closure such as snaps, or a masking fabric layer.

If you want to avoid the hem connections, you can wire all inputs at the top separately and use multiple I/O pins on the microcontroller. But then you'll have to route 5 more wires to the pocket.

Before installing anything to the skirt, the full LED arrangement can be pre-soldered. Wire length depends on your skirt size; add some spare length to account for movement. To secure and strengthen the connections, add heat-shrink tubing at the top of each strip (Figure F).

At this stage, I'd highly recommended testing the LED strips with some sample code using the NeoPixel or Fast-LED library, to check your wiring before assembling the skirt. This can be done by following Adafruit's great NeoPixel guide for the QT Py board at learn.adafruit.com/qt-py-and-neopixel-leds.

Next, attach the rough "hook" side of the Velcro to your LED strips to line up with the Velcro on



I

the skirt panels. Just cut away 2cm sections of the adhesive backing on the strips, and stick the Velcro to the adhesive (Figure G).

5. ATTACH LEDS TO SKIRT

Place the LED strips on their desired positions and fasten with the Velcro (Figure H). Route the cables through the double-layer waistband, and the guide the wires at the skirt's hem in the closure.

6. CONTROLLER AND POWER SUPPLY

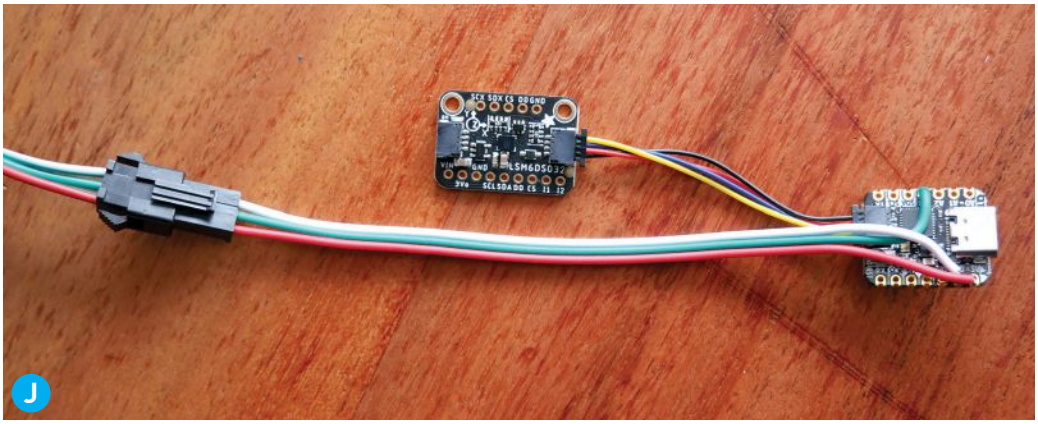
Connect the electronics as shown in the wiring diagram (Figure I). The LED strip's signal wire is soldered to pin A3 of the QT Py board. The 6-axis IMU sensor is connected via the short STEMMA/Qwiic cable.

The cable connectors that are originally pre-soldered to the LED strips can be reused to make it easier to unplug and remove the electronics from the skirt for washing (Figure J). This way, no additional connectors are necessary.

To power the QT Py, you can use a standard USB power bank. 5000mAh seems to be more than enough for a full day's use if you're not using effects that light all the LEDs at the same time. The power bank and the QT Py and sensor nicely fit into the pocket at the back of the skirt (Figure K). Plugging the QT PY's USB into the power bank will power up the LEDs.

7. PROGRAMMING

You can use either CircuitPython or Arduino IDE to program the QT PY board. We used Arduino IDE in combination with Visual Studio Code and PlatformIO, so that we could include a special library (github.com/ArminiusM/Neopixel) which can easily divide an LED strip into multiple segments to control them as if they were wired



J

as a single strip. It generates "virtual NeoPixel strips" so you can directly use existing lighting effect examples and libraries that are widely available for Neopixel LEDs.

Let's take a look at the code. To show the use of the 6-axis sensor, we'll look at the "bouncing ball" animation effect. Other effects like "meteor rain" or "fire" — and all future examples — are available at this project's Github repository, github.com/makeTVee/ledskirt.

At the start of the code (Figure L), libraries for the LSM6DSO32 sensor and the LEDs are included. Also, the QT Py's pin A3 is defined as LED output as described above. If you have a different LED arrangement or number of strips, you can change **MAXPIXELS** and **NSTRIPES** accordingly.

Lines 15–20 define the virtual strips with a start and end pixel for each strip (Figure M). For example, **vNeo1** is the first strip of LEDs, with the start pixel **19** and the stop pixel **0** to address it from bottom to top, even if the signal comes from the top side. Because of the serpentine layout, **vNeo2** as the second strip is oriented in the opposite direction, so start pixel is **20** and stop pixel is **39**.

Line 23 defines an array **bballs[]** of the type **npBouncingBall** objects and assigns it to the different virtual strips. This is the effect object coming from the **np.BouncingBall.h** library. After declaring the used variables, the sensor and LEDs are initialized in the **setup()** function (Figure N). Sampling rate and range for the accelerometer are also set and can be changed here if needed.

K

```
#include <Arduino.h>
#include <Adafruit_LSM6DSO32.h>
#include <npMeteor.h>
#include <npBouncingBall.h>

#define NeoPin     A3 // NeoPixel Pin
#define MAXPIXELS 120 // Number of Pixels
#define NSTRIPES 6 //Number of Stripes

npNeoPixel pixels = npNeoPixel(MAXPIXELS, NeoPin, NEO_GRB + NEO_KHZ800, 8.0);
```

L

```
// define 6 virtual strips
// reversed for bouncing ball effect (counting from bottom to top)
npVirtualNeo vNeo1(&pixels, 19, 0);
npVirtualNeo vNeo2(&pixels, 20, 39);
npVirtualNeo vNeo3(&pixels, 59, 40);
npVirtualNeo vNeo4(&pixels, 60, 79);
npVirtualNeo vNeo5(&pixels, 99, 80);
npVirtualNeo vNeo6(&pixels, 100, 119);

//array of bouncing balls for easier access, different initial delay
npBouncingBall bballs[NSTRIPES] = {npBouncingBall(50,vNeo1),npBouncingBall(60,vNeo2),
                                   npBouncingBall(40,vNeo3),npBouncingBall(20,vNeo4),
                                   npBouncingBall(40,vNeo5),npBouncingBall(70,vNeo6)};
```

M

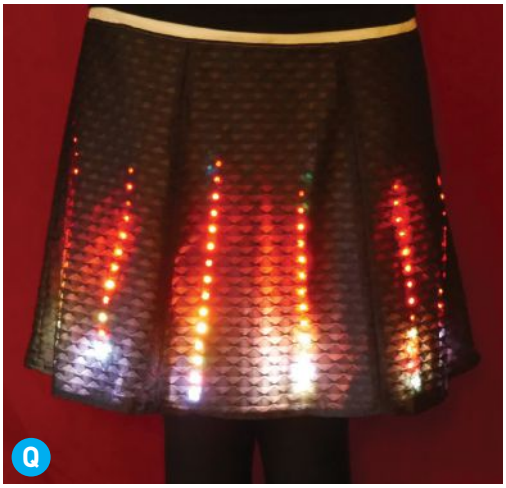
```
sensors_event_t accel_old;
Adafruit_LSM6DSO32 dso32;
float diff[3] = {0,0,0};
float diffsum;
byte *c;
float Motion_Theshold = 10.0;

//-------------------------------------------------------------------
void setup ()
{
  // setup LSM6DS032 sensor
  if (!dso32.begin_I2C()) {
    while (1) {
      delay(10);
    }
  }
  dso32.setAccelRange(LSM6DSO32_ACCEL_RANGE_8_G);
  dso32.setAccelDataRate(LSM6DS_RATE_12_5_HZ);

  pixels.begin();
  pixels.clear();
  pixels.npShow();
}
```

N

```
52  void loop ()
53  {
54    sensors_event_t accel;
55    sensors_event_t gyro;
56    sensors_event_t temp;
57    dso32.getEvent(&accel, &gyro, &temp);
58
59    //calculating the diff to previous value for x,y and z axis
60    diff[0]=abs(accel.acceleration.x - accel_old.acceleration.x);
61    diff[1]=abs(accel.acceleration.y - accel_old.acceleration.y);
62    diff[2]=abs(accel.acceleration.z - accel_old.acceleration.z);
63
64    accel_old=accel;
65    //react to all axis
66    diffsum=diff[0]+diff[1]+diff[2];
67
68    for(int i=0;i<NSTRIPES;i++) bballs[i].update();
69
70    if (bballs[3].hasFinished() && (diffsum>Motion_Theshold))
71    {
72      //c=Wheel(random(0,255));
73
74      for(int i=0;i<NSTRIPES;i++)
75      {
76        //uncomment following line for fixed collor for the balls
77        //and the upper line c=Wheel...
78        //bballs[i].changeColor(*c, *(c+1), *(c+2));
79        bballs[i].restart();
80      }
81    }
  }
```

O

P



In the main **`loop()`**, line 57 updates the sensor data and the array **`diff[]`** is calculated as the absolute difference between the last acceleration sample and new sample for all three axes (Figure O). We're only interested in the acceleration change and don't care about the static offset value based on the gravitational acceleration. Using all three axes also makes the calculation independent from the orientation of the QT Py board in the skirt pocket, so we don't have to worry about that. Line 68 updates the effect continuously. If the effect with the shortest delay has finished, line 70 checks for a new trigger based on the sum of the axis's differences and the previous defined motion threshold. If the threshold was hit, the effects are restarted on all bouncing balls. The code shown here changes the color continuously during the effect; if you prefer fixed colors, you can uncomment lines 72 and 78. This changes the color only for every new bouncing effect cycle.

Of course, this is a simple example of motion triggering because it's using all the axes; you can trigger the effect by jumping, spinning, starting to run, whatever. If you prefer a single-axis trigger, you can fix the orientation of the sensor in the skirt pocket (or you can determine the orientation by software using the g-offset and doing some coordinate transformation to find the true z-axis).

8. UPLOAD THE CODE

To flash the code to the QT Py microcontroller, just follow Adafruit's guide at learn.adafruit.com/adafruit-qt-py/arduino-ide-setup. There you'll also find all the information you need if you want to use CircuitPython instead of Arduino IDE.

FLASH IT AND FLAUNT IT

The skirt is a great eye-catcher and attracts lots of attention and smiling faces. Figure P shows a sequence of the bouncing ball effect, but the video is so much better. Check it out at youtube.com/watch?v=3_XDv31yBbY.

Figure Q shows the fire and meteor effects that are also available at the Github repository. You can see them in the video too.

There are many further optimizations and modifications possible, and some are already in development. Just for starters, you could:

- Use the gyro for rotation-triggered effects
- Use the double-tap detection engine of the sensor to switch between effects
- Use the built-in pedometer to sync the effects with your steps
- Add backside LEDs for 360-degree effects (only for showcases, don't sit on them!)
- Extend the LED strips for long skirts

There's a good discussion going at my Hackaday.io project page (hackaday.io/project/177255-neopixel-led-skirt), and all future source code will be uploaded to the Github repo, so it will be easy to share and contribute — your ideas are highly welcome. Using an ESP32 and syncing multiple skirts with WLED (github.com/Aircoookie/WLED) also could be a lot of fun!

Smart³

I made a Rubik's Cube that can solve itself. (Oh yeah, and it levitates.)

Written and photographed by Takashi Kaburagi

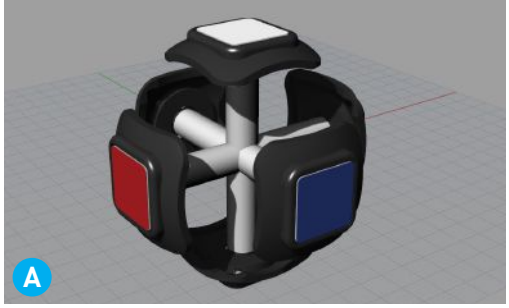
TAKASHI KABURAGI is a maker in Japan. Watch his Self-Solving Rubik's Cube videos on his YouTube channel "Human Controller."

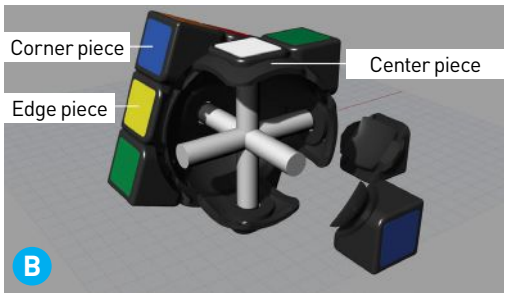
MATERIALS

- **Servomotors, MG90D (6)**
- **MOSFET motor drivers, DRV8830DGQR (6)**
- **10-MSOP dip adapters (6)**
- **Capacitors, 1608 package, 1000pF (6)**
- **Capacitors, 2012 package: 0.1µF (12), 10µF (6)**
- **Rotary position sensors, AS5600 (6)**
- **Neodymium magnets, 1.45mm×2mm cylinder, diametrically magnetized (6)**
- **I²C bus multiplexer, PCA9547D**
- **Resistors, 1608: 4.7KΩ (14), 10KΩ (1)**
- **Accelerometer, MMA8451Q module**
- **Microcontroller, RedBear BLE Nano V2**
- **Rechargeable battery, Li-ion polymer, DTP401525 3.7V 110mAh 1C**
- **Battery connector, PH, 2P B2B-PH-K-S**
- **Perf boards: 1.27mm pitch (1), 2.54×2mm pitch (1), and 2.54mm pitch (1)**
- **Socket header, 1.27mm pitch (1)** for programming
- **Polyurethane enameled copper wire, 0.29mm dia.**
- **Insulation tape**
- **Screws: M1×5mm (1), M1.4×3mm (9), M1.4×5mm (44), M1.4×8mm (20), and M2.5×4mm thin head (6)**
- **Springs, outer diameter 3mm, pitch 1.2mm, free length 4.5mm, wire diameter 0.3mm (24)**
- **Color stickers, 14.5×14.5mm: white (6), blue (6), orange (6), green (6), red (6), and yellow (6)**
- **Steel plates for center lid, 14.5×14.5×0.3mm (6)**
- **Neodymium magnets, 2.5×2×1.7mm square (12)**
- **3D printed parts: gearbox (1), center pieces (6), edge pieces (12), and corner pieces (8)**
- **Cross-shaped plates, 0.6mm plastic (6)** for fixing springs to servo horns

TOOLS

- **3D CAD software** I used Rhinoceros 5.0.
- **3D printer**
- **Caliper**
- **Sandpaper set**
- **Precision needle file set**
- **Polishing compounds**
- **Design knife**
- **Precision drill bit set**
- **Pin vise**
- **Precision screwdriver set**
- **Soldering station with thin solder, flux, flux remover, and blotting wire (braid)**
- **Nipper**
- **Tweezers**
- **Needlenose pliers**
- **PCB board cutter**
- **Instant adhesive**
- **Black instant adhesive** I used this as a putty.
- **Silicone oil**





In the summer of 2015, I went to my first Maker Faire, in Tokyo, Japan. Afterward, I could only think about the challenge of making something. It moved me so much that I left the company where I worked as a programmer for 16 years and became a full-time maker.

I challenged myself to create something that seemed impossible. I like to surprise people, so I resolved to make a Self-Solving Rubik's Cube, starting on it in March 2016; on September 13, 2018 it solved itself without getting stuck for the first time.

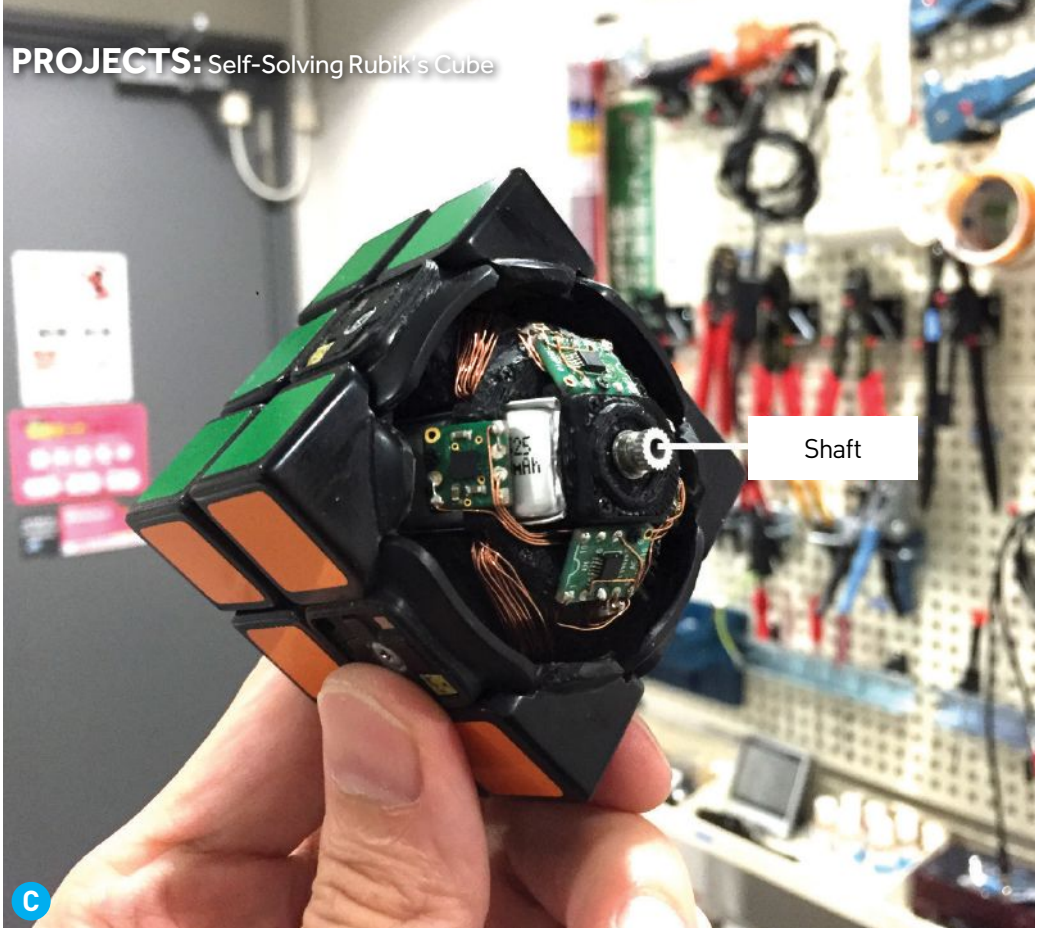
I shouted at that moment. And maybe people all over the world who watched it on YouTube shouted, too. That's great!

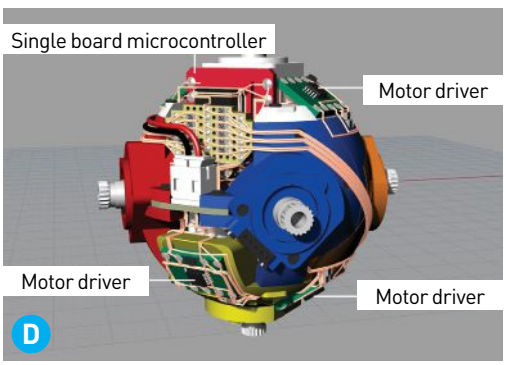
MECHANISM

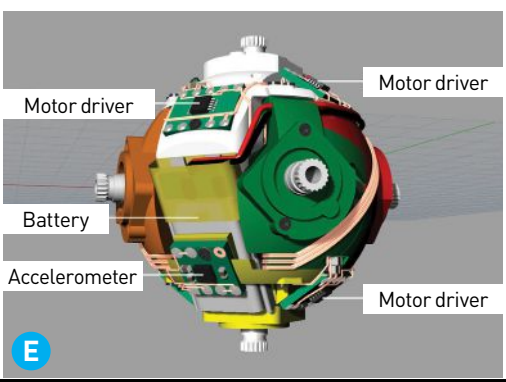
My cube, when scrambled and placed on a desk, starts turning its faces to solve itself. It recognizes its color arrangement and solves from its current state, rather than reversing scrambled movements. It works independently without the need for an external computer or cameras, and can solve itself in about 30 seconds, no matter how scrambled.

Many people have never disassembled a Rubik's Cube, so I'll explain its structure.

First, there are 6 shafts. The rotatable center pieces are attached to the end of them (Figure A), and there are edge pieces and corner pieces around the center pieces (Figure B). When you turn the faces by hand, the edge pieces and







corner pieces move and the colors scramble. The pieces don't come off because they're hooked on each other.

The Self-Solving Rubik's Cube turns the face by turning the shaft with a motor (Figure C). I put 6 sets of gears, motors, motor drivers, and rotary position sensors — along with a single board microcontroller, a battery, and an accelerometer — inside the Rubik's Cube (Figures D and E).

The accelerometer recognizes when the Self-Solving Rubik's Cube is placed on the desk. When the microcontroller commands the motor driver, the motor connected to it rotates, turning the gears, and, in turn, the face.

The microcontroller can recognize the rotation angle of the shaft with the rotary position sensor, and will command the motor driver to stop the motor when the shaft reaches the target angle.

The rotary position sensor I selected is magnetic. It recognizes the rotation angle of a diametrically magnetized neodymium cylinder magnet attached to the end of the shaft (Figure F).

RECOGNIZING COLOR ARRANGEMENT

Generally, a machine that solves Rubik's Cube with robot arms uses cameras or color sensors. I came up with the idea of using rotary position sensors. I programmed the microcontroller so that the rotary position sensor would detect when the face was turned by hands and the color arrangement data in the memory would change.

The Self-Solving Rubik's Cube does this all the time, so the color arrangement data in memory always matches the actual color arrangement.

The idea of using rotary position sensors was good because there was no need to add new sensors — but it has the limitation that the colors must be aligned before the Self-Solving Rubik's Cube is powered on.

SOLVE WITH A MICROCONTROLLER

There are many different methods for solving Rubik's Cube. For my project, I went with CFOP (Cross – F2L – OLL – PLL), one of most commonly used methods in speed-solving a Rubik's Cube.

A computer, using a brute force solution, can solve a Rubik's Cube with about 20 moves. The more human-oriented CFOP program I made takes an average of 52 moves. It's longer, but better oriented for the microcontroller inside, and probably more entertaining for Rubik's Cube fans.

My CFOP program calculates all the moves needed to solve the cube when it is placed on the desk. This takes about 3 seconds.

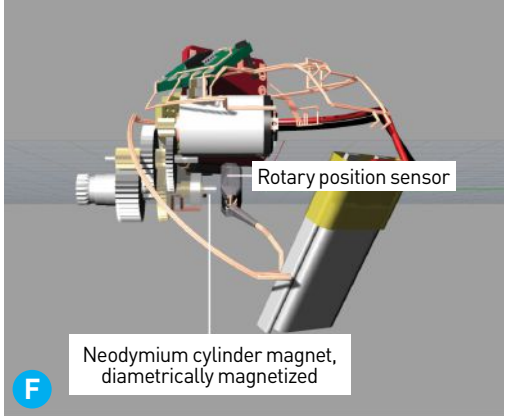
The single board microcontroller I selected is the RedBear BLE Nano V2 (end of production). I didn't need BLE, but I selected it because it's small, and its 512KB of flash memory was adequate for the CFOP program. I used the Mbed Online Compiler for programming it (Figure G).

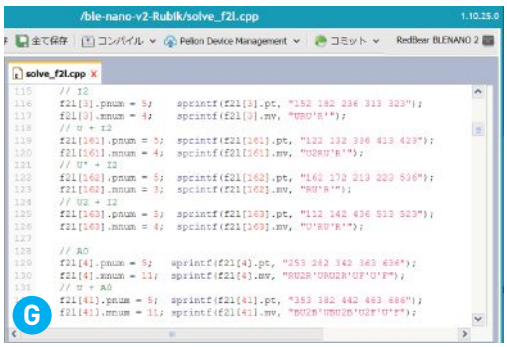
It's worth noting that I couldn't solve Rubik's Cube before starting this project, and had to learn how to do so before programming. Turns out, it's pretty fun.

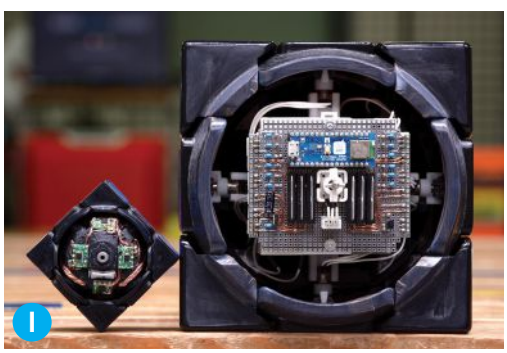
HOW DID I MAKE IT?

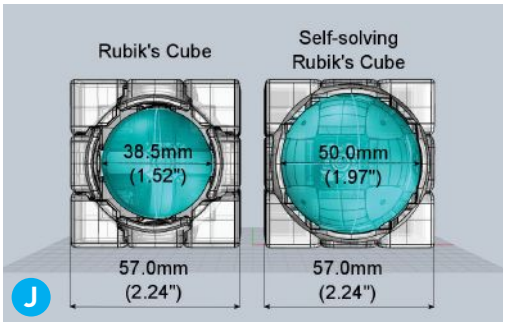
The first thing I did was to measure the parts of a commercial Rubik's Cube with a caliper and create 3D data of it. Then I made prototypes from that on a 3D printer. The size of Rubik's Cube is 57mm (2.24"), and the internal space is small; therefore, the prototypes were very big (Figures H and I).

Problem: They couldn't roll on the table, as they need to when solving themselves. This was because the power of the motor wasn't strong enough.

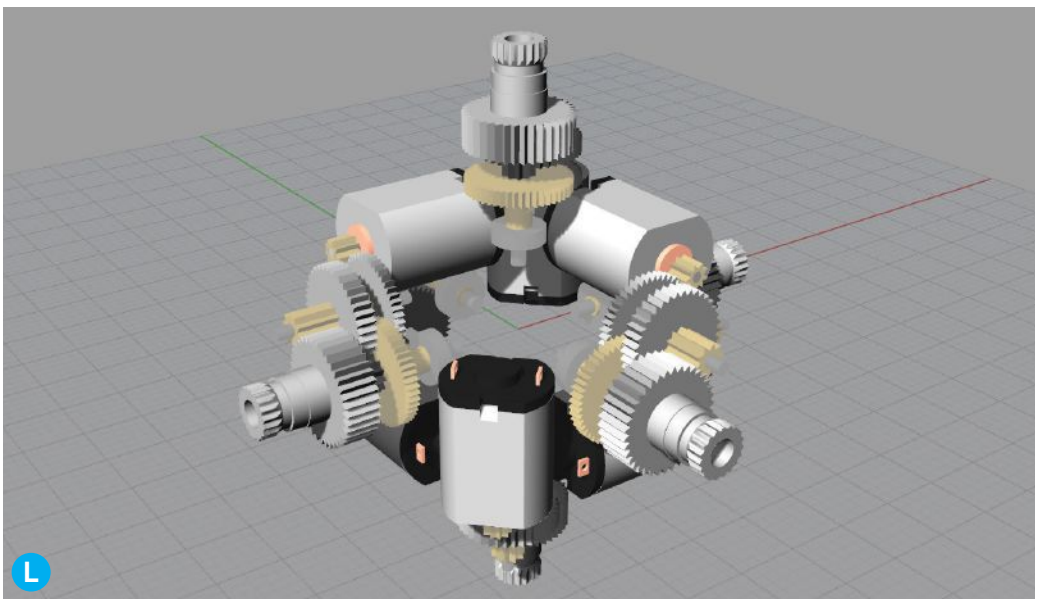


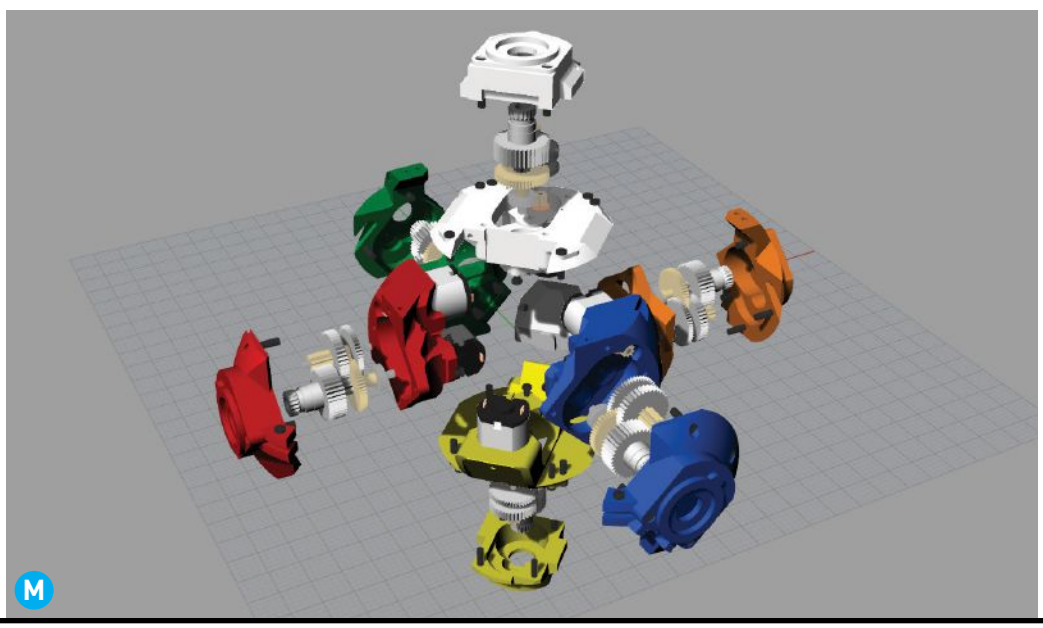





Rubik's Cube
Self-solving Rubik's Cube
38.5mm (1.52")
50.0mm (1.97")
57.0mm (2.24")
57.0mm (2.24")
J


K


L

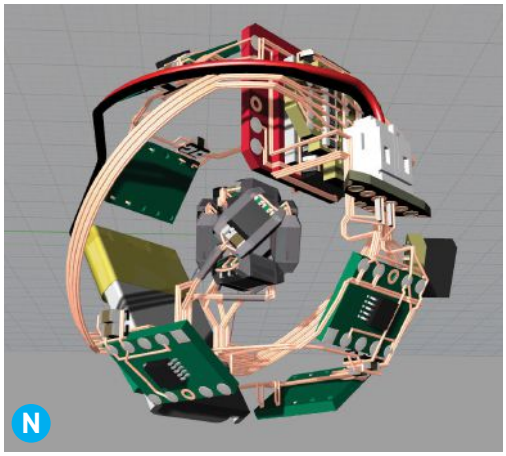

M

Finally, I redesigned the 57mm (2.24") Rubik's Cube with a larger internal space (Figure J). And I repeated parts selection and design changes to make an internal mechanism that fits in that space. The biggest parts were the servomotors (Figure K).

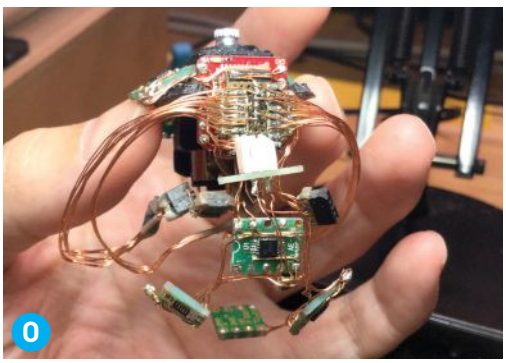
I decided to disassemble the servos to access the gears and motors and use just them. I spent a lot of time considering their arrangement (Figure L). Then I made a 13-piece gearbox and put the servo gears and motors in it (Figure M).

I put other electronic parts in the remaining space (Figures N and O).

The cube's electronic circuit has a lot of room for improvement. For example, the battery may be damaged because this circuit does not limit the current of the motor driver. I wouldn't advise using it for reference, but wanted to share the version that first solved itself in 2018 nonetheless (Figure P).



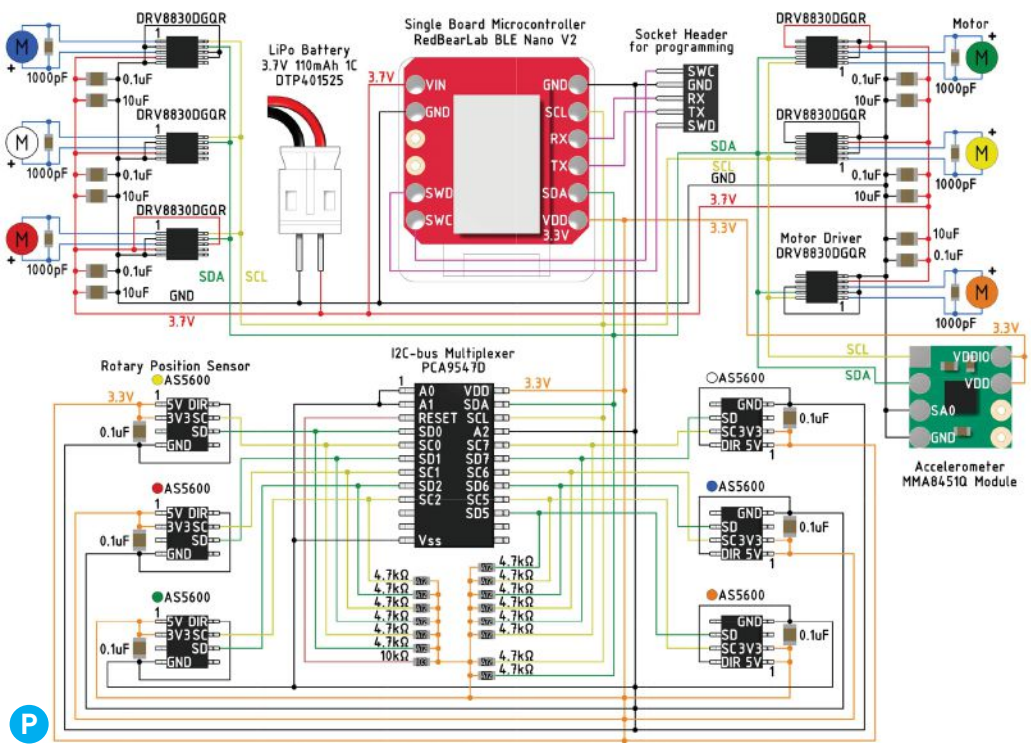
N



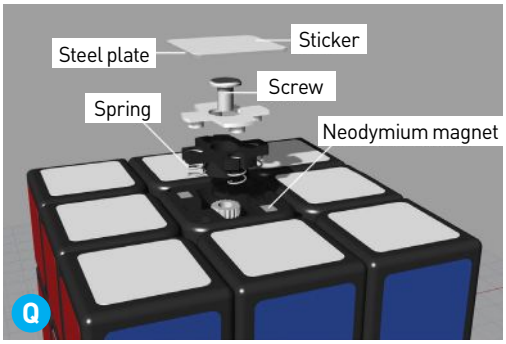
O

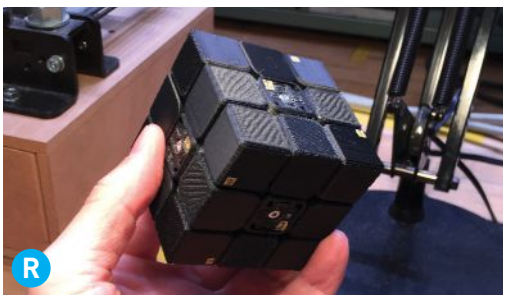
TURNING THE FACES FROM INSIDE

This was the most difficult part of this project. The six faces of Rubik's Cube are made up of 26 separate pieces. Because each face shares pieces with its neighbor, a face that is misaligned



P





prevents the other faces from turning — ideally the face would turn exactly 90 degrees each time. However, because the faces are made up of separate pieces, even if the shaft turns exactly 90 degrees, the faces will slip a little each time. Therefore, the face must be able to turn even if the other faces are slipping a little.

Two things are needed to solve this. One is a gap between the pieces. The other is to add springs into the centerpiece. These springs pull all the pieces to the center of Rubik's Cube (Figure Q). This will leave a gap between the pieces only when the face is stuck, while keeping the face from sticking in normal use. It's the same mechanism used on a commercial Rubik's Cube.

I first printed all 26 pieces with a 3D printer to a slightly larger size (Figure R). I filed them little by little over a month with precision needle files and sandpapers. During that time, I found the proper width of the gap and proper strength of the spring by repeatedly assembling them (Figure S).

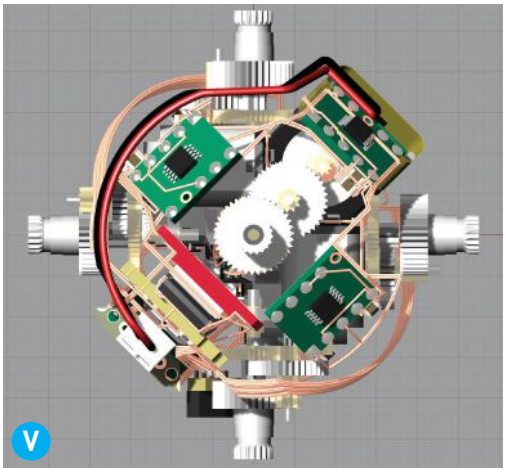
I then polished the parts well with compounds to reduce surface friction and allow them to slip like molded products (Figure T). I also applied silicone oil to the surface.

TOOLS AND MAKERSPACE

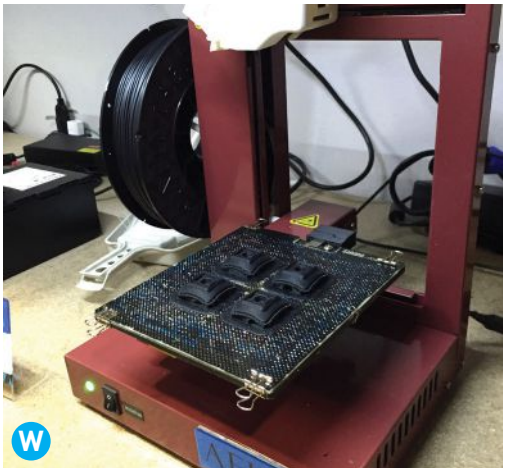
When I started this project I had no experience with 3D printers or 3D CAD, and had little experience with electronics. Immediately after I left my job, I moved to the neighborhood of the makerspace DMM.make AKIBA in Akihabara, Tokyo (Figure U), where I learned everything by spending almost every day working on the Self-Solving Rubik's Cube. This makerspace is open 24 hours a day and there are many stores selling electronic parts in the neighborhood.

Learning 3D CAD (I used Rhino 5.0) allowed me to examine all the parts' arrangements over and over again on the computer, so I was able to pack them into the cube (Figure V). I modified and printed the gearbox and pieces multiple times a day on an Afinia H480 3D printer (Figure W).

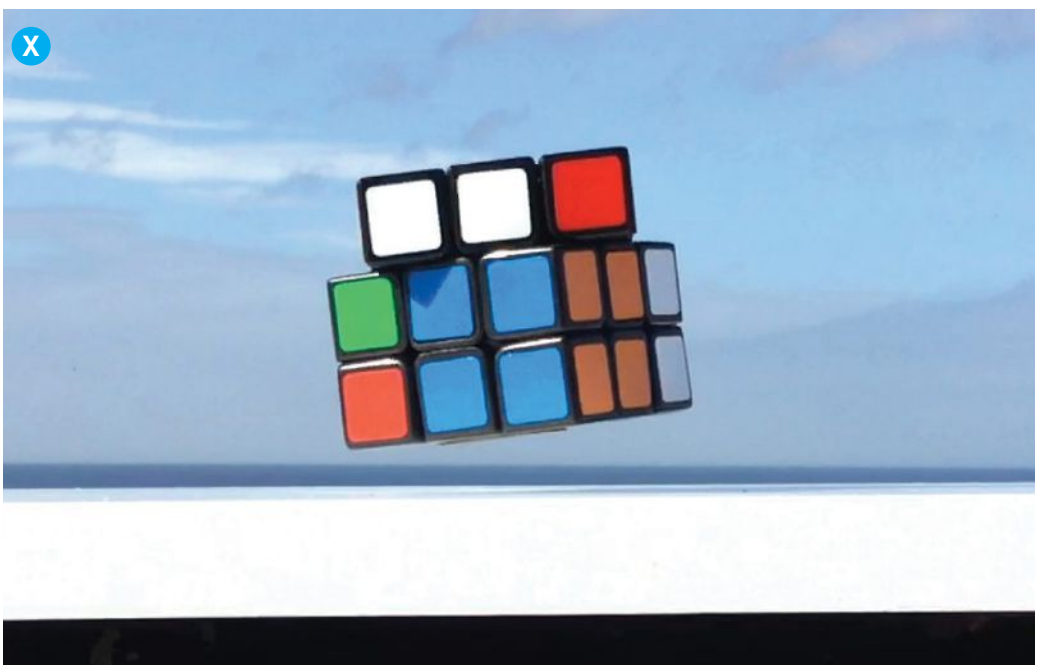
In this way, I was able to repeat trial and error in a short period of time, so I was able to create hardware that would work without any mechanical design experience. Thanks to these two tools and the makerspace, I was able to complete the Self-Solving Rubik's Cube.



V



W



X

...AND THEN I TOOK IT FURTHER

The year after I completed the Self-Solving Rubik's Cube, I debuted a new feature: the ability to solve itself while floating in midair (Figure X). I have not disclosed the mechanism for this — as I said, I like to surprise people. Have fun imagining how it works!

Follow Takashi Kaburagi on YouTube to see his Self-Solving Rubik's Cube in action, and all of his other projects: youtube.com/user/meiji0vs0lotte.

Digital String Art Portraits

Written by Raphael Schaaf

Make an old craft new with algorithmic designs that produce amazing artworks from thread

In 2016 artist Petros Vrellis shared a stunning new portrait technique — "computational string art" that's generated by an algorithm. The computer converts a photograph to a grayscale string pattern and also generates instructions for stringing a thread around hundreds of pegs or hooks, crossing over thousands of times to create the finished picture. To my knowledge Vrellis is the first to use this method. He even figured out how to do it in color (Figure A).

Based on Vrellis' description of how his algorithm works, I created my own algorithm that takes an image and converts it into a grayscale string picture (Figure B).

I've written and shared two Windows programs at makezine.com/go/digital-string-art, so you can do this project at home. *Knit_Generator.exe* can convert images to string pictures and export the instructions. *Knit_Instructor.exe* can open the instructions and tell you how to string the frame, step by step. If you can't run Windows there's a web-based alternative that should work for everyone at github.com/christiansiegel/knitter.

You can generate the art and instructions in about 5–30 minutes depending how fast your PC is, but you should plan on many hours to string it!

MAKE YOUR DIGITAL STRING ART PORTRAIT

Before you start, download the project software from makezine.com/go/digital-string-art.

1. CHOOSE A PHOTO TO CONVERT

The base image that you use really makes or breaks the end result. Images with lots of contrast work best, like a portrait with front-side lighting. Gradients show up well; small details will probably get lost in the process. It's a bit of trial and error to find a good image and good settings for it.

TIME REQUIRED: 12–15 Hours

DIFFICULTY: Intermediate

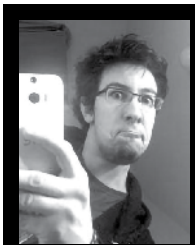
COST: $30–$60

MATERIALS

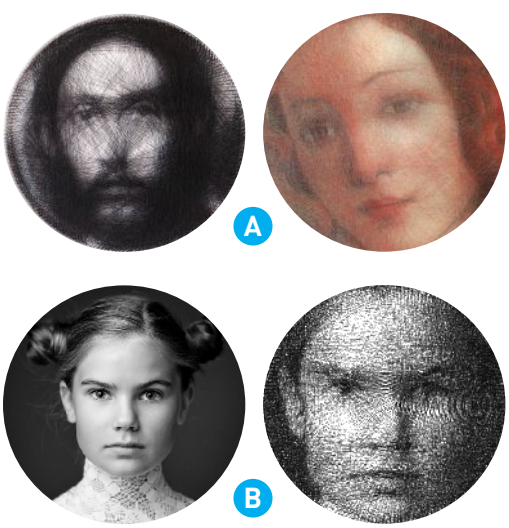
- **Circular frame** A wooden hula hoop or ring works well for this, or just a round board.
- **Tiny nails (200–300)**
- **Thread for industrial sewing machines** These spools can be found with 1km+ of length, size 80–100. To calculate the length you need with enough to spare, just take the diameter of your ring times the number of lines.

TOOLS

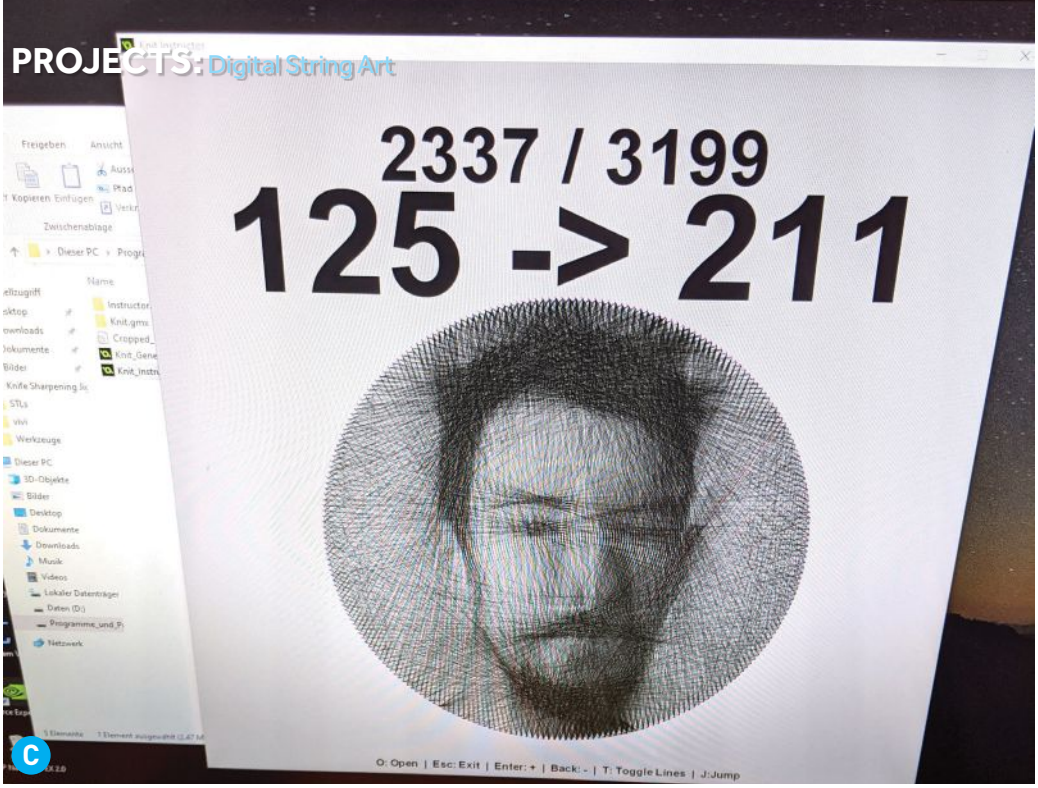
- **Computer with project software** free download at makezine.com/go/digital-string-art
- **Drill with tiny bit** for predrilling nail holes
- **Hammer**
- **Tape**
- **Pen**
- **Scissors**
- **Knife**



RAPHAEL SCHAAF is an automation technician by day and a general maker by night. He's done everything from programming games to woodworking, knife making, 3D printing, CNC machining, and many more. He just loves to try new things.



Petros Vrellis (A); Raphael Schaaf (B)



C

Raphael Schaaf

2. GENERATE YOUR DIGITAL STRING ART

a. Open *Knit_Generator.exe*.

b. Press Space to load an image (*.png*, *.jpg*, and *.bmp* files should work).

c. Position the part of the image you want converted inside the circle using the arrow keys to move it and the 1 and 2 keys to scale it.

d. Set the number of hooks, radius of the circle (bigger = more accurate but slower calculation), number of lines, and contrast (contrast is inverse; a higher number will create a flatter image).

e. Press Enter to start the calculation. It takes a couple of seconds to initialize.

f. Pressing D during the calculation switches the drawing of the lines on and off. You can use this as a preview, but it will slow down the calculations extremely if left on.

g. When finished, press S to export the generated image as an *.ini* file containing the instructions for assembly, and choose the location where you want to save the data.

TIP: The software is doing millions of calculations. If you have problems I recommend reducing the number of lines, hooks, and the radius to smaller numbers (~50). The higher those numbers are, the longer it takes to calculate.

3. PREPARE YOUR FRAME

I made a 3D printed jig to predrill the nail holes. It was just a small section that I could slide around the ring to evenly space the holes. You can just go around the ring with a ruler to get an even spacing for your nails, but you might have to redo the calculation if the number of nails doesn't match what you generated previously.

After that, I put tape around the ring and labeled all the nails. Make sure you start counting from 0 and that your nails match what you put into the Generator program. 0–250 means you have 251 nails; I messed that up on my first try!

4. THREAD YOUR PORTRAIT

This is going to take awhile! Find a comfortable, well-lit place to work.

a. Open *Knit_Instructor.exe*.

b. Press O to open a previously generated *.ini* file.

c. A single line will be displayed at a time, with its instruction: which hook to connect the thread from, and to (Figure C).

d. Cycle through the steps with Enter and Backspace.

e. Use J if you want to jump to a specific step.

f. Use T to toggle the image on and off.

NOTE: The radius of the circle doesn't have a specific dimension or unit of measurement. In the Generator and Instructor software, it will be in pixels, but if you use it with a CNC machine it can be whatever unit you choose.

YOUR FIRST MASTERPIECE

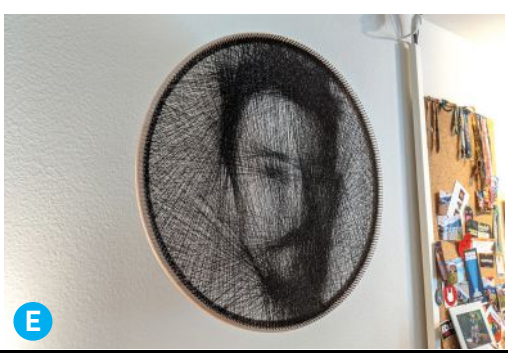
I finally finished my first portrait! And as any great artist has to have one, I decided to make a self-portrait (Figure D). 😉 I'm quite happy with how it turned out.

LESSONS LEARNED

- **RING STRENGTH:** The wooden ring I used wasn't really strong enough for 3,000+ lines of string, so it got slightly distorted into a "Pringles" shape. This could cause a loss in details. A second ring glued to it might help with that.
- **DISPLAY:** I didn't hang my picture directly on the wall, as not enough light would pass through and there wouldn't be sufficient contrast between the string and the background, making the image too dark. A backlight might help with this, but I just hung it from the ceiling about 20cm away from the wall (Figure E).
- **BRIGHTNESS/CONTRAST:** My portrait turned out slightly darker than it appeared in the software, so keep that in mind if you try your hand at one of these string portraits. A thinner thread probably would have helped with that, or a higher contrast setting and fewer lines.

SETTINGS EXPLAINED

- **RADIUS:** The radius of the circle in pixels. Has no connection to how large you want to make the actual portrait. A bigger radius will make a more precise portrait, but will also take longer to compute. Use a small radius for quick previews and a bigger one when creating the final instructions for your portrait.
- **HOOKS:** The number of hooks or nails you'll use. More hooks make for nicer portraits with fewer geometric artifacts, but take longer to compute. Keep in mind how close together you can fit the hooks/nails.
- **LINES:** The number of lines that will be drawn between the hooks. Good values are generally between 2,000–3,000. When the generating process is near its end and all the new lines are placed around the border of the ring, you have too many lines. When they're still being placed through the middle of the ring, you have too few lines. Try to find a good balance.
- **CONTRAST:** The trickiest one to figure out. Basically the value that gets subtracted from every pixel when a line passes through it. A higher value should be used for thicker threads or smaller portraits. 20–50 seems to work fine.
- **SCALE:** Zoom in and out of the picture with 1 and 2 keys.
- **OFFSET:** Move the picture with the arrow keys.
- **DETAILED VIEW:** To open and close a more detailed view of what's going on during the generating process, press D. This will show all the lines calculated so far and the remainder of the source image. Will slow down the calculation immensely, so don't leave it open.



MORE STRING ART GENERATORS

When Hackaday posted Vrellis' work, and again when I posted my project at Hackaday.io, many readers shared their own string art generator programs. Here are a few you can try; there are lots more out there.

- **WEAVING ALGORITHM** by *Make:* author Dan Royer, same algorithm, but in Processing so you don't need an EXE file or Windows. Includes a few different styles like square, circle, and color (still experimental). github.com/i-make-robots/weaving_algorithm
- **KNITTER** by Christian Siegel, in Processing, to generate a circular or square pattern. github.com/christiansiegel/knitter
- **AUTOMATE THE ART** by Erich Eisler; this one is Processing and Arduino code for his string art machine (see opposite page). github.com/ericheisler/AutomatedArt
- **THREADTONE** by Tobias Scheepers, Python code using OpenCV for image processing and Gcode for stringing the thread. github.com/theveloped/ThreadTone; there's also a semi-finished web app by Gerben at gerben.algemeenbekend.nl/threadtone

GOING FURTHER

I tried a bunch of variants for the algorithm, but surprisingly the results mostly looked the same. My code was written in GameMaker Studio; I have no idea how easy it would be to convert into other languages, but I guess the important part is the main algorithm, which you can extract from *controler.object.gmx*.

I'm mostly done with this project, but I like to see what other people do with it. A number of people are doing amazing portraits, some in different shapes and some even in full color, like husband and wife duo Ani and Andrew Abakumov (ugallery.com/artist/ani-and-andrew-abakumovs). Their work (Figure F) is super impressive — hard to believe it's the same process, the color images look so much better. I guess how you layer the colors has a big influence on the final result. I hope they publish the tools they use or even the source code.

Darya Kondrateva

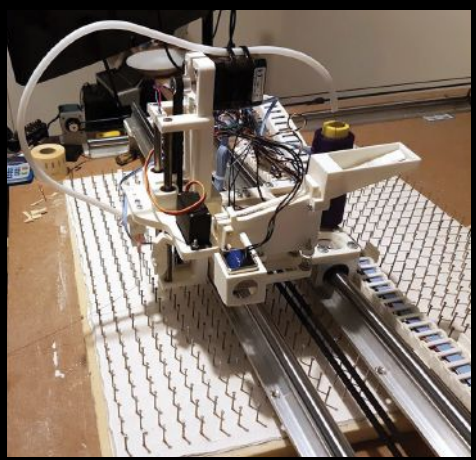
THREAD-O-MATIC

MAKE A MACHINE DO IT!

I strung my portrait by hand, but other makers have invented machines for stringing theirs automatically. Here are a few that I like.

Automate the Art by Erich Eisler

Hackaday Prize entry, built with a bike wheel and solenoid, shortly after Petros Vrellis described his technique in 2016. Processing and Arduino. hackaday.io/project/13047-automate-the-art



String Art Machine by Paul Morris-Hill

A different style of string art, but I like this one because he goes so in depth on the machine he built. .NET, Repetier, Gcode. medium.com/@paulmorrishill/building-a-string-art-machine-eeee386a38db

A New Spin on String Art Machines by Barton Dring

This one is my favorite because it's such a simple and great design. It drills holes for the nails and strings the picture with the same machine! MATLAB, Python, Gcode. youtu.be/M1gXuKFspgY and github.com/bdring/StringArt

Erich Eisler; Paul Morris-Hill; Barton Dring

sPot: The Spotify iPod

Written and photographed by Guy Dupont

Hack a 17-year-old iPod to stream Spotify with the most satisfying user interface — the click wheel

TIME REQUIRED: 1–2 Weekends
DIFFICULTY: Intermediate
COST: $100–$200

MATERIALS

- **Apple iPod 4th-gen, model A1059** It's officially referred to as the "Click Wheel" iPod.
- **Raspberry Pi Zero W mini computer**
- **LiPo/Li-ion battery charger, USB-C** Adafruit #4410, adafruit.com
- **Boost converter module, 5V 1A** Adafruit MiniBoost, #4654
- **Mini sound card/DAC, 5V USB, PCM2704 chip** such as Comimark, Amazon B07WXQQ3ML
- **Li-ion battery, 3.7V, 1000mAH**
- **Vibration motor, 10×2mm disc** for haptic feedback. I used Amazon #B073YFR5WR.
- **TFT display, 2", NTSC/PAL** Adafruit #911
- **NPN transistor, 2N3904** to drive the haptic motor from the Pi's GPIO
- **FPC to DIP breakout board, 8 pin, 0.5mm pitch** Amazon #B07H5GCZFW
- **Resistor, 220Ω**
- **SD card, 32GB or more** for the Pi
- **Assortment of hookup wire** I mostly use 30AWG solid.

TOOLS

- **Spudger/opening tool** to get into the iPod. I like the iFixit assortment IF145-364-1, ifixit.com.
- **Soldering iron** of your choice, with solder
- **Desoldering equipment** braid or solder sucker
- **Small snips or flush cutters**
- **Wire strippers**
- **Tiny screwdrivers**
- **Multimeter (optional)**
- **Keyboard and monitor** for initial setup of the Raspberry Pi. You can also use your computer.

This is a 4th-generation iPod from 2004 that I modified to stream any song from Spotify. I added some modern flourishes, such as Wi-Fi, Bluetooth audio, a search screen, and haptic feedback. It's all driven by a Raspberry Pi Zero W and some other easily accessible parts from vendors like Adafruit. I call it my "sPot."

I was inspired to build the sPot after inheriting a classic iPod and using it (in its original form) for a few days last summer. I was struck by how well the original user experience held up over the years — the feedback provided by the click wheel as I scrolled was still so satisfying. I wanted to experiment with how well it would work in the streaming age, where playlists can be arbitrarily large, and manual text entry is the norm. Turns out — still fun and totally usable!

Is this a pointless project? It's an experiment, it's a party trick, it's a prototype, it's a statement. This is me, a consumer unsatisfied with the options available, attempting to build my ideal user experience using products from two different companies who compete for my exclusive time and money. That is a deeply satisfying exercise — especially as the hardware we buy becomes harder and harder to repair.

From a technical standpoint, I was really excited to get some practice integrating some new experience into an existing piece of hardware. I figured out fairly early on how to "hack" the click wheel, and managed to gain access to the lock switch and headphone jack. It was a great experience, and it's really satisfying to be able to hand someone this thing without them knowing immediately that it's been modified.

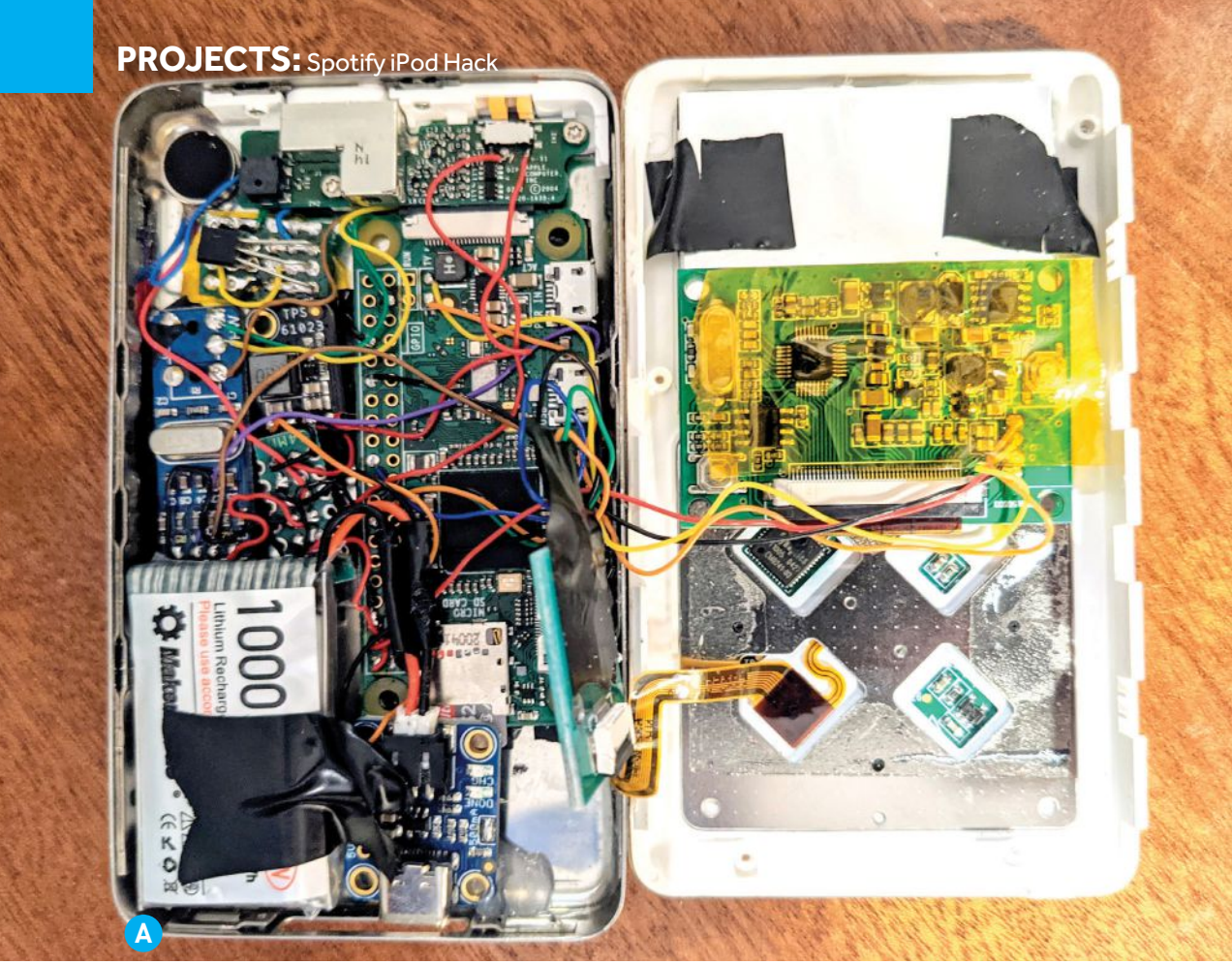
GUY DUPONT is a creative technologist from Cambridge, Massachusetts. He leads a small but mighty software team at Pison Technology and moonlights as an audio engineer.

BUILD YOUR SPOTIFY CLICK-WHEEL IPOD

Before you start, please watch my walk-through videos, Part 1 at youtu.be/ZxdhG1OhVng and Part 2 at youtu.be/q0pUPab7Rms.

1. CAREFULLY DISASSEMBLE THE IPOD

Follow iFixit's guide: ifixit.com/Guide/iPod+4th+Generation+or+Photo+Rear+Panel+Replacement/390.

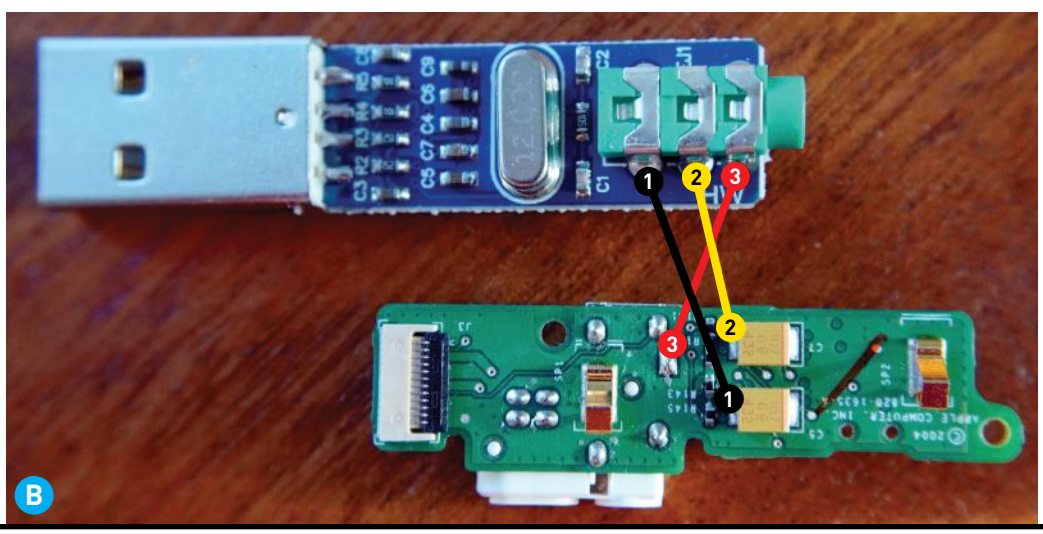


The motherboard, battery, and HDD can be put aside, we won't be using those.

Keep the enclosure and the headphone jack assembly handy.

2. PREPARE THE SOFTWARE

It's best to get the Raspberry Pi up and running before trying to get everything in place. Follow the directions on my Github page github.com/dupontgu/retro-ipod-spotify-client. You can do



all of this with your Pi hooked up to a proper keyboard and monitor.

3. MODIFY THE USB DAC (OPTIONAL)

If you only want to use Bluetooth audio, you can skip this step.

Use your desoldering technique of choice to remove the USB connector and the TRS audio jack from the sound card/DAC. I was able to get them mostly removed using a bit of braided copper solder wick. The audio jack was a little stubborn so I finished it off with some snippers. Just be careful not to damage the pads on the PCB! You can watch me do this about 13:25 in the Part 2 video on YouTube.

4. LAY OUT YOUR COMPONENTS

From here on out, we're going to be connecting all the hardware. Before you solder anything in place, lay out your components in the iPod case and come up with a plan. Here's where everything ended up with mine (Figure A).

5. CREATE YOUR POWER RAILS

Inside the iPod, you need to provide 5 volts to the Raspberry Pi, the display, the USB sound card, and the haptic motor.

I found that having a centralized place to route power from helped keep things organized. (Yes, the images you see are organized, by my standards.) I cut a piece of standard 0.1"-spaced perf board with two columns: one for 5V and the other for ground.

6. CONNECT USB DAC TO HEADPHONE JACK (OPTIONAL)

Again, if you only want to use Bluetooth, you can skip this step.

There are three connections to make here: one to the tip of the audio cable, one to the ring, and one to the sleeve. The tip and ring should be wired to the two capacitors on the headphone jack assembly (connections 1 and 2 in Figure B). The sleeve should be wired to the rectangular pad shown as connection 3.

7. CONNECT THE LOCK SWITCH

I decided to use the iPod's lock switch as a power switch. There are two small pins on the bottom of the switch. Wire one of them to the Enable pin on the MiniBoost module, and the other to ground (Figure C). When the switch is closed and the Enable pin is pulled low, the boost module will stop providing power to all components.

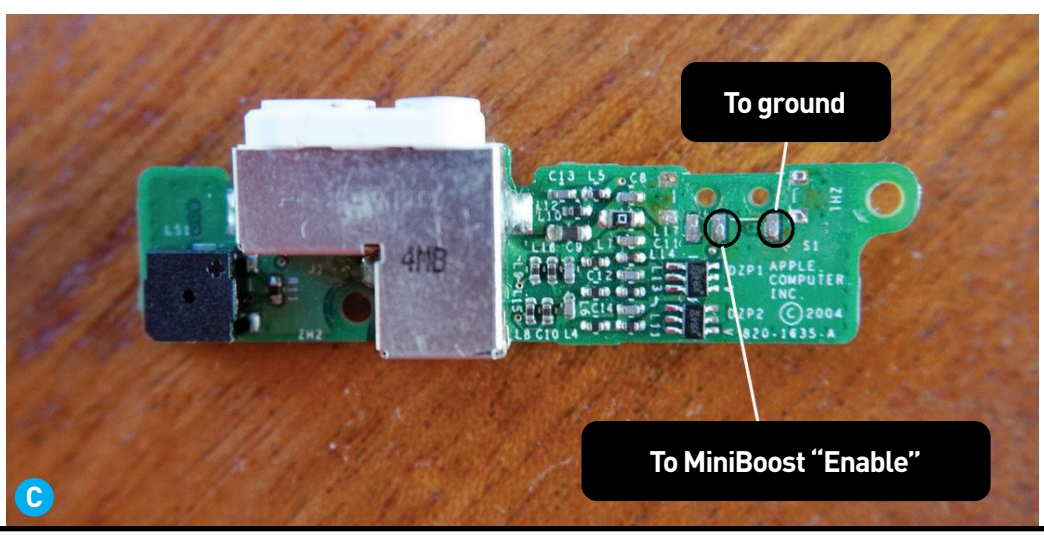
8. REATTACH THE HEADPHONE JACK ASSEMBLY

Screw the assembly back into the case, ensuring that the headphone jack and switch are properly aligned.

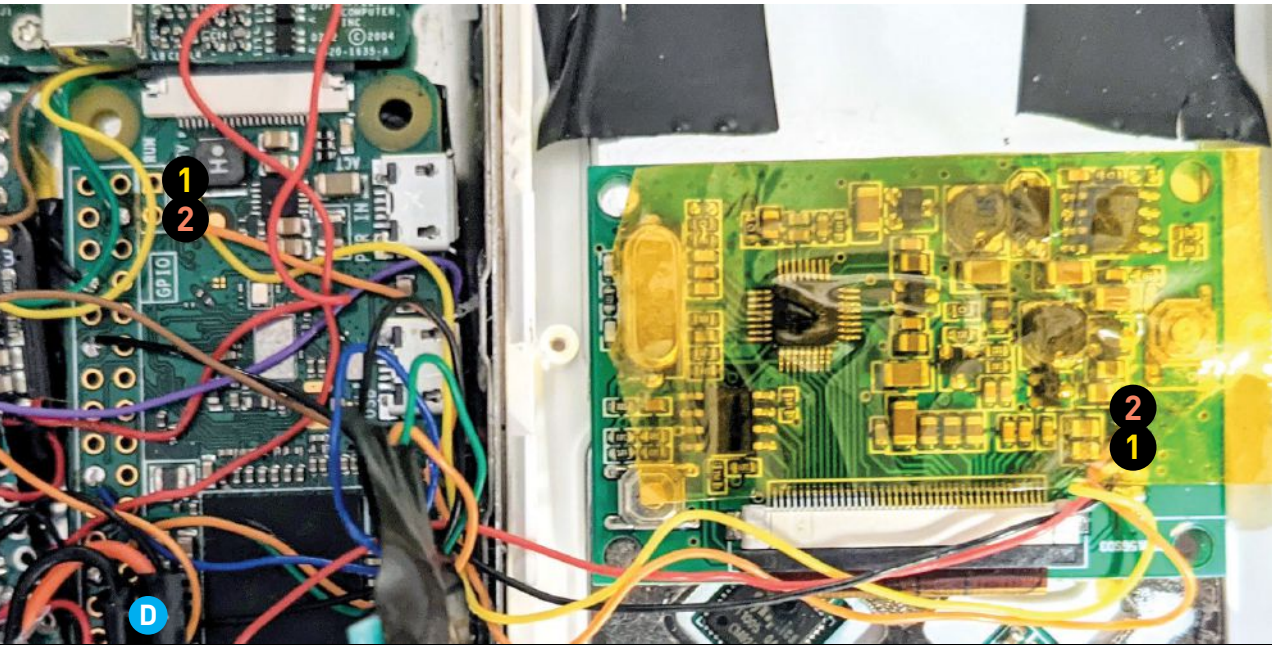
You may have to snip out some of the plastic to make space for your new audio wires.

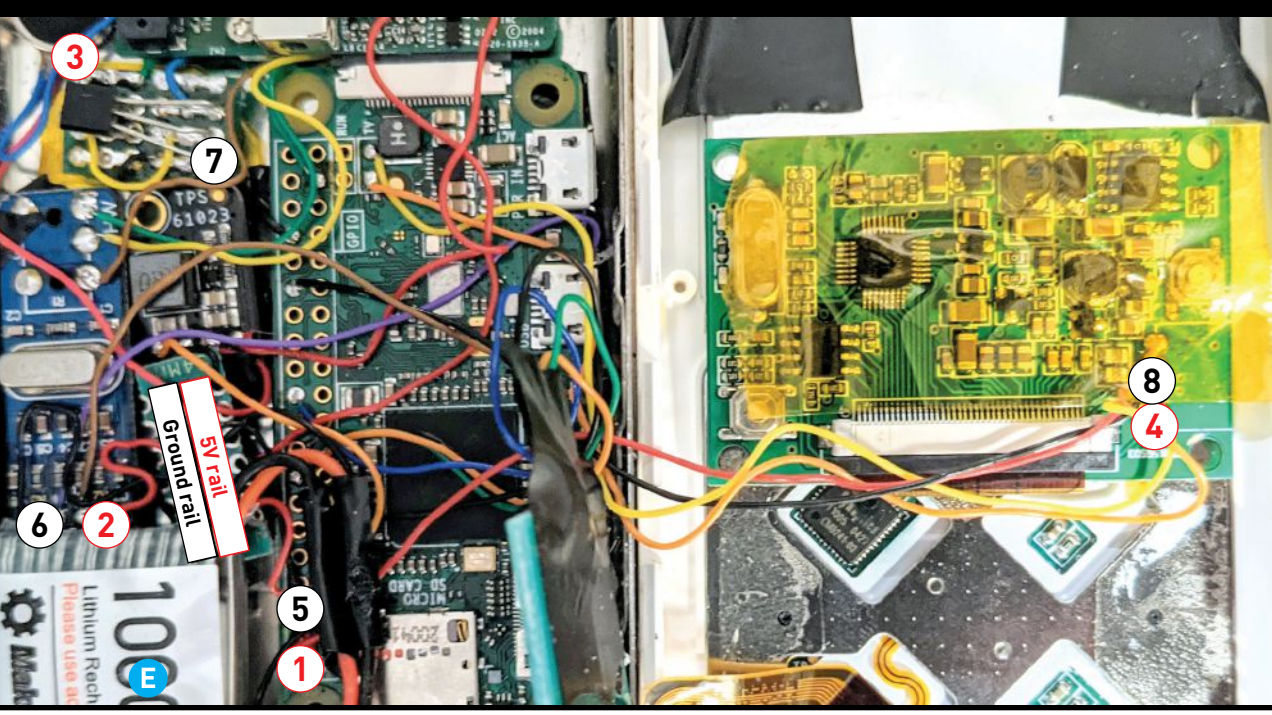
9. CONNECT DISPLAY TO PI COMPOSITE OUTPUT

On the Raspberry Pi, the two pins for composite video are labeled TV. Connect these to the display



C


1
2
2
1
D


3
7
TPS
61023
5V rail
Ground rail
6
2
5
1
8
4
E

module as shown in Figure D (yellow and orange). I removed the headers on the display, along with the plastic casing, to create more room.

10. CONNECT 5V AND GROUND

Following Figure E, from your 5V rail, run connections (marked in red) to:

1. one of the Raspberry Pi's 5V pins (see raspberrypi.org/documentation/usage/gpio)
2. the USB DAC
3. one of the haptic motor's leads, and
4. the display.

From your ground rail, run connections (black) to:

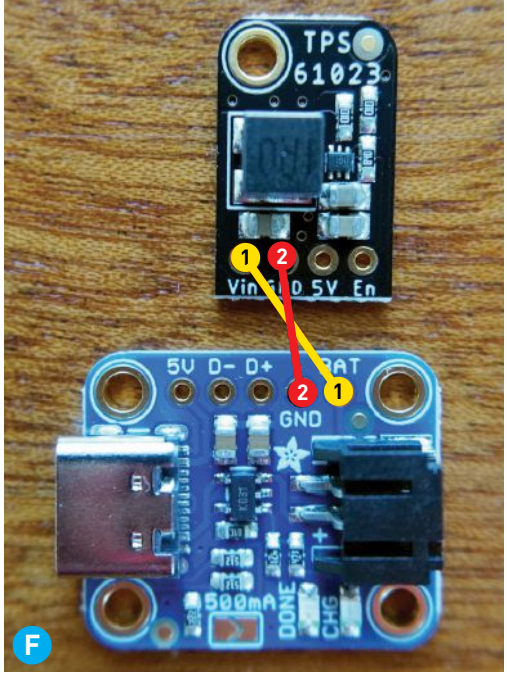
5. one of the Raspberry Pi's ground pins
6. the USB DAC
7. the haptic driver transistor's emitter pin, and
8. the display.

11. CONNECT CHARGER TO BOOST CONVERTER

Connect the battery charger board's BAT pin to the boost converter's Vin pin. Also connect their ground pins; I show them connected directly in Figure F, but in the final build I wired them both to the shared ground rail created in Step 5.

12. CONNECT BOOST CONVERTER TO POWER

Connect the boost converter's 5V pad to your 5V rail, and its ground to your ground rail (if you hadn't already).

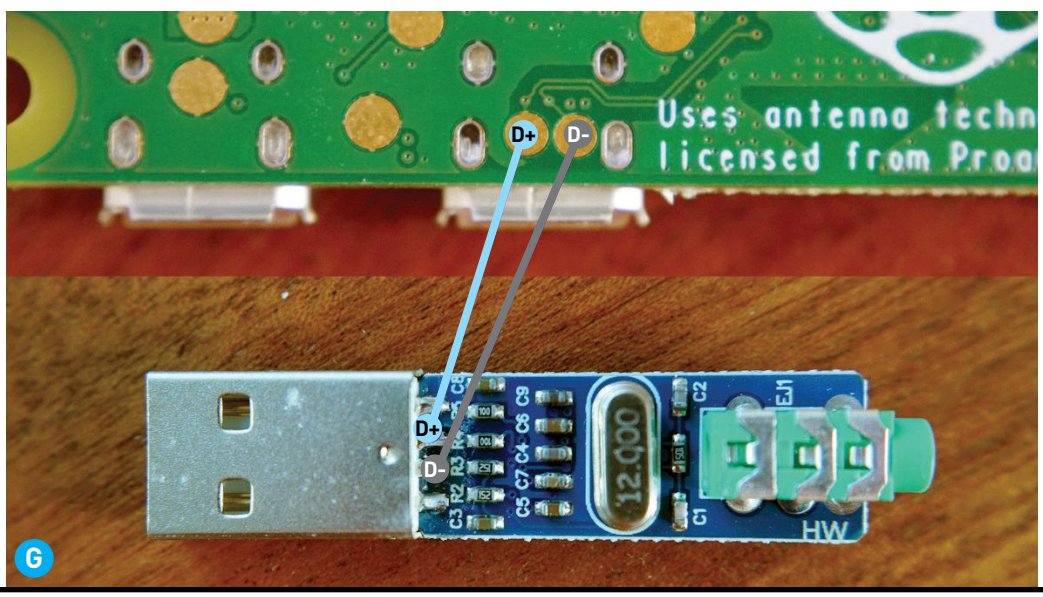


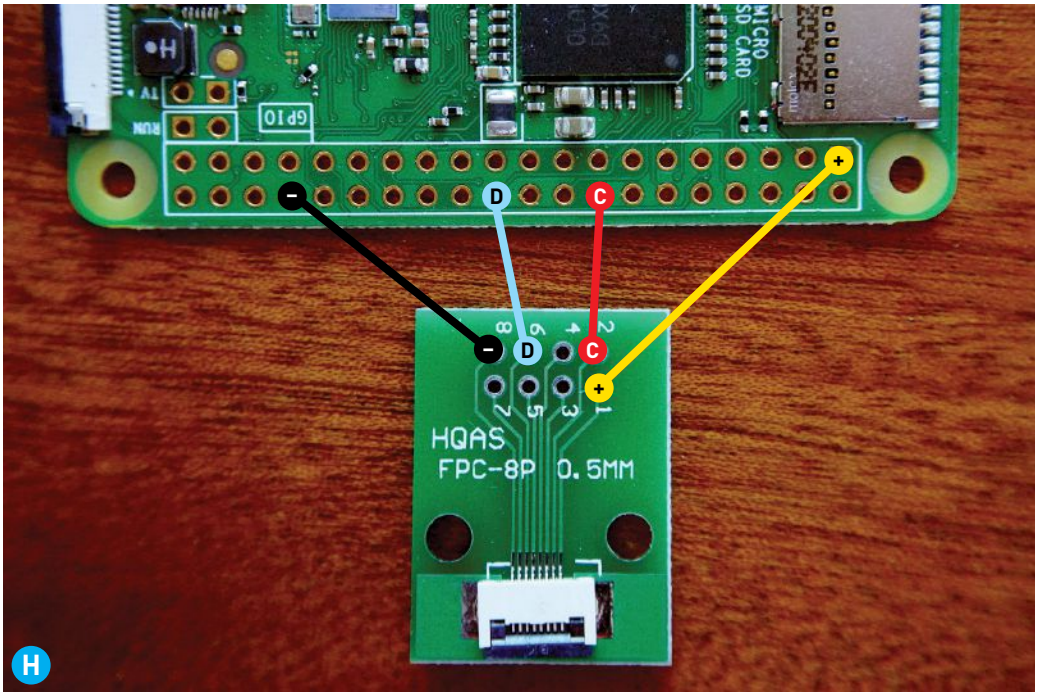
13. TEST!

It's a good idea to use a multimeter to make sure you have no continuity between 5V and ground. Then, plug your LiPo battery into the charger board. If your lock switch is open, everything should power on! At this point, you can visually verify the Raspberry Pi and the display. Then, power everything off and remove the battery.

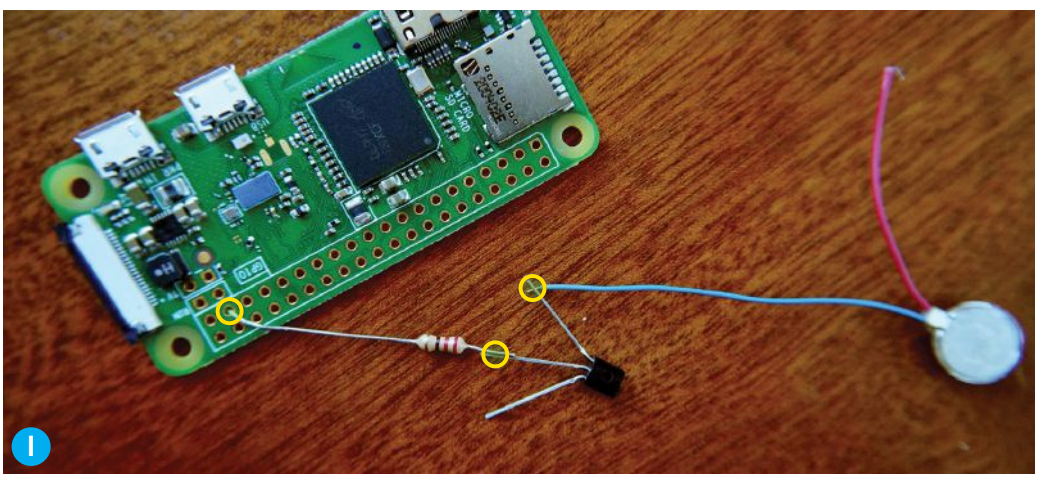
14. CONNECT USB DAC TO RASPBERRY PI

There's no room for a USB connector here, so you can hard-wire the data lines from the DAC to the Raspberry Pi's USB test points (Figure G).





H



I

15. CONNECT CLICK WHEEL TO RASPBERRY PI

Connect the Raspberry Pi's 3.3V output to the FPC breakout board's pin 1, Pi ground to pin 8, GPIO 23 to pin 2, and GPIO 25 to pin 6 (Figure H).

Carefully insert the iPod click wheel's ribbon cable into the breakout board's FPC socket. The cable's pin numbers are printed on one side, so be sure to align them with the breakout board's.

16. CONNECT HAPTIC MOTOR TO PI

Wire the remaining motor lead to the NPN transistor's collector pin. Connect the transistor's base pin to the Raspberry Pi's GPIO pin 26, through a 220Ω resistor (Figure I).

17. INSERT BATTERY AND REASSEMBLE

Plug the battery in and test once more. Then, tape off any leads/pads that may come in contact with the case or other stacked components. Better safe than sorry.

Carefully close up the case. If you feel any pressure, stop! Again, depending on how you lay

out your components, you may have to snip out some of the old plastic spacers.

18. POWER ON AND TEST AGAIN

Now you can test everything in the software, including the audio and click wheel integration. SSH into the Pi to make changes, and be sure to properly shut it down before cutting power.

THUMB THING NEW

Your Spotify-hacked iPod can stream any song directly from Spotify. It's got all your playlists, your saved albums and artists; it's got a regularly updated list of new releases (Figure J), and you can even search the entire Spotify catalog.

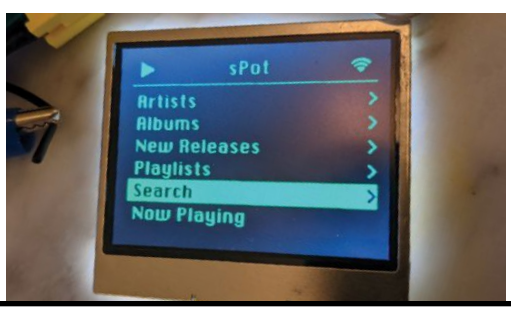
I first posted this build on Hackaday.io in January 2021 (hackaday.io/project/177034) and the response has filled my little hacker heart with joy. There are some great discussions and improvements happening on the Github and Hackaday.io pages! Hop in if you have suggestions or need help. Maybe I'll even do a DIY kit.

GRATITUDE

Thanks to a random 10-year-old Hackaday article that points to a blog post written by someone named Jason Garr, I was able to figure out exactly what each of the click wheel's wires is supposed to do. Finding that blog post (jasongarr.wordpress.com/project-pages/ipod-clickwheel-hack) honestly felt like a miracle — there's no info out there about these old iPods — so Jason, wherever you are out there, thank you!

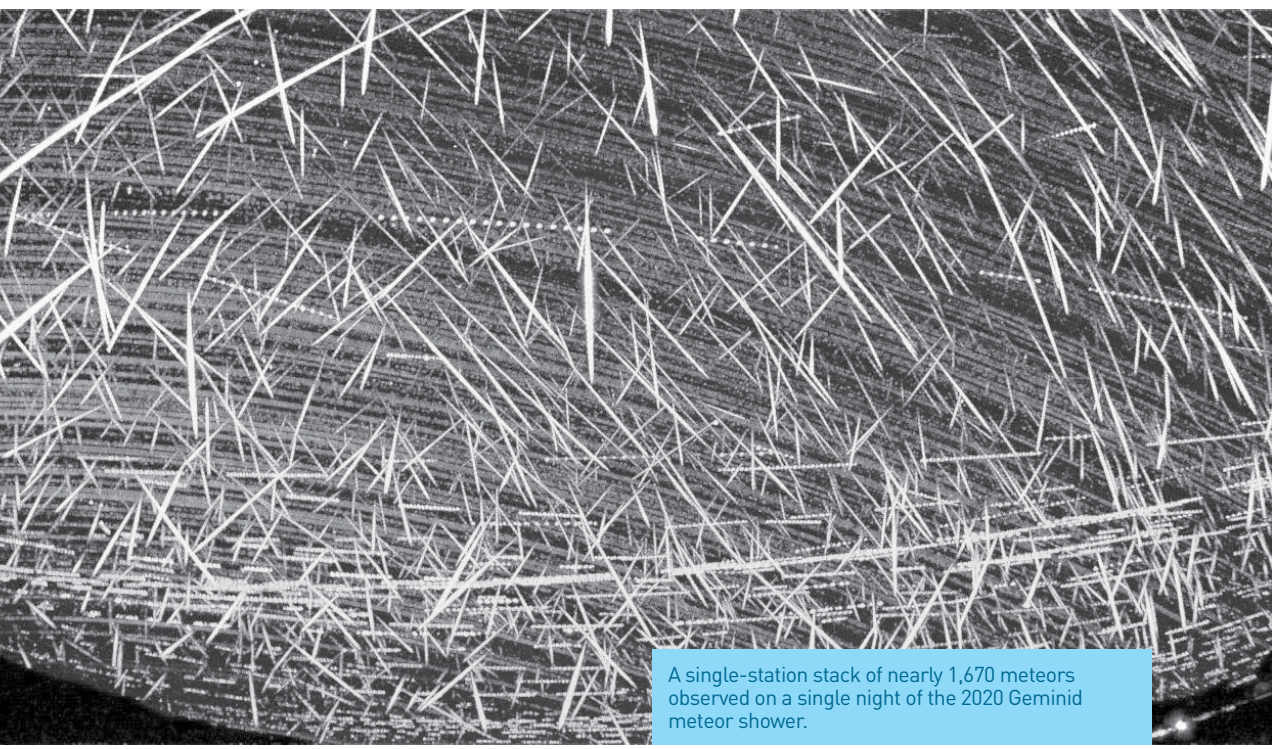
Github user André Silva (ElCapitanDre) in Portugal submitted a pull request for podcast access, so that code is now available if anyone wants that feature. Nice!

I'd also like to give a shout out to Ricardo Sappia in Germany, who has built his own "Spotifypod" that improves on mine in many ways — 3D printed chassis, LCD display, larger battery, etc. He's also been tremendously helpful on the software side. Check out Ricardo's build, with lots of schematics, at rsflightronics.com/spotifypod.



Raspberry Pi Meteor Camera

Written by Michael Mazur and Denis Vida



A single-station stack of nearly 1,670 meteors observed on a single night of the 2020 Geminid meteor shower.

Build your own fireball tracker and become a citizen scientist in the Global Meteor Network

DENIS VIDA and **MIKE MAZUR** are members of Western University's Meteor Physics Group in London, Ontario, where they study the orbits and physical properties of meteors and meteoroids using optical and radar instrumentation, including high-resolution, high-sensitivity meteor cameras.

Every day, about 40 tons of extraterrestrial material enters the Earth's atmosphere to produce meteors that we see in the night sky. These "shooting stars" often appear as faint streaks of light, but occasionally produce brilliant light shows that can momentarily turn night into day. Although they may look close enough to touch, meteors typically occur at heights of 70–110km. And since they're travelling at hypersonic speeds of 11–72 kilometers per second, even if you could touch them, it would be a bad idea.

So how can we learn more about the origins of the billions of meteoroids hitting our atmosphere each day? Well, two of the most basic techniques are radar and optical observations. Setting up a meteor orbit radar costs millions of dollars, so it's not feasible for the average citizen-scientist. Good optical observations, however, can be made cheaply with a few common pieces of off-the-shelf equipment.

Don't let the low cost fool you — what we describe here is a project that will help you build a globally connected meteor camera that can collect science-grade data suitable not only for orbit determination, but also for helping mitigate satellite impact risks, predicting meteor storms, and recovering meteorites.

THE GLOBAL METEOR NETWORK

For meteor scientists, video observation is one of the easiest and most cost-effective ways to gain a better understanding of the evolution of material in the Solar System. Direct asteroid sampling missions such as Hayabusa-2 and OSIRIS-REx are very expensive, while meteorite recovery is biased and represents only a tiny fraction of the material in the Solar System. Video meteor observations, on the other hand, have an entry barrier so low that nearly anyone can produce high-quality data.

Even so, most recent meteor camera networks have been expensive, geographically limited, and fragmented, with little communication between them. To solve these problems, the Global Meteor Network (GMN) was born.

There are now more than 300 Raspberry Pi Meteor Stations (RMS) operated by citizen scientists in 22 countries around the world, connected through the GMN with the long-term

TIME REQUIRED:
1–2 Hours

DIFFICULTY:
Intermediate

COST:
$200–$500

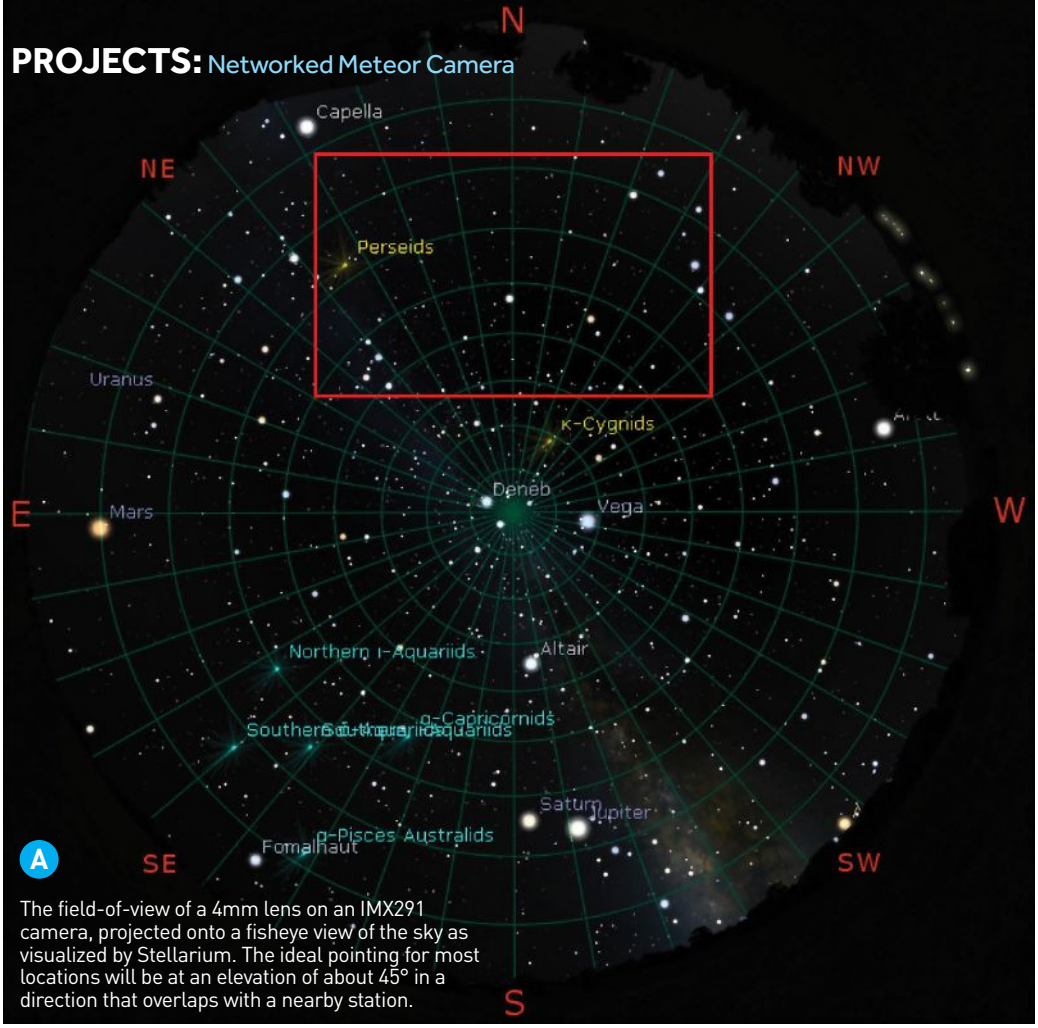
MATERIALS

- **Raspberry Pi 4 mini computer** A Raspberry Pi Meteor System (RMS) disk image is available for the Pi 4 Model B. We also run cameras on PCs and Jetson Nano, and it's likely possible on many other Linux machines.
- **Pi case with fan** You can 3D print our RMS case (Figure C on page 85) from thingiverse.com/thing:4795923.
- **SD card, 128GB** Get a quality, name-brand card.
- **Power supply, 3A, high quality** Your Pi will be running at a high load when capturing images. We recommend the official Pi power supply.
- **Real-time clock (RTC) module (optional)** We use inexpensive modules, but you can also use internet NTP to synchronize your time instead.
- **Low-light IP camera module, IMX291 or IMX307** We capture data at 720p because this maximizes the sensitivity to meteors. Unlike stars, meteors are transient phenomena and their light needs to be concentrated into as few pixels as possible while still having a decent image resolution. Otherwise, their light is scattered into too many pixels and they're lost in the noise. So that new 20MP camera you've had your eye on is not the right choice here.
- **IP camera cable with POE converter** If you can't find one built-in, or you need more power for a heater/fan, you'll need a separate POE (power over Ethernet) converter module.
- **Fast lens** For a good field-of-view on the IMX291 sensor, the 4mm f/0.9 Starlight lenses work well if you have dark skies and no obstructions (trees, buildings, etc.). If you're in a city, buy the 6mm lens, and if you really have terrible skies, go with the 8mm lens.
- **Waterproof camera housing, mount plate, and arm** Waterproofing is key — your camera will be pointed up and exposed to the elements. Not all security camera housings come with a mounting plate for the camera, so you may need to specify this when you order. Or you can design and print your own (Figure B on page 85).
- **POE injector** A 48V 0.5A wall-wart injector can power the camera and heater (if you add one).
- **Ethernet cables, CAT5 or higher (2)** a short one from injector to Pi, a long one to the camera
- **Standoffs, M2×10mm (optional)** If you use a POE converter module, you may need extra standoffs to assemble everything.
- **Silicone sealant** to seal around the glass and screw holes on the front of the camera housing. You really don't want moisture inside.

TOOLS

- **Phillips screwdriver**
- **Side cutters, small**
- **Allen key (optional)** for the Pi case

M.Mazur



The field-of-view of a 4mm lens on an IMX291 camera, projected onto a fisheye view of the sky as visualized by Stellarium. The ideal pointing for most locations will be at an elevation of about 45° in a direction that overlaps with a nearby station.

goal of characterizing the apparent point of origin in the sky (the *radiant*), total number, and size distribution of meteors entering our atmosphere. Hundreds of thousands of meteor orbits have already been logged through the work of citizen scientists and, thanks to their dedicated work, it won't be long before we're talking about millions of orbits!

HARDWARE

Although the hardware required for imaging meteors is fairly basic, there are a few things to keep in mind when making a meteor camera.

First, meteors are typically faint and barely visible with the naked eye, so the camera sensor needs to be sensitive to very low light levels and paired with a fast lens. Our favorite IP camera modules, at the moment, use Sony's ⅓" IMX291 and IMX307 sensors. Although these can be bought for $35–$50, be aware that a current shortage of certain HiVision SoC chips means that prices fluctuate.

Lens-wise, a focal length of between 4 and 8mm gives good sky coverage while speeds faster than f/1.0 allow faint meteors to be captured. Choosing the best lens starts by calculating the field-of-view for your sensor:

FOV = 57.3 / focal_length * binned_pixel_size_mm * # pixels

and considering any obstructions (trees, buildings, etc.) in your preferred pointing direction.

A 4mm lens, for example, paired with an IMX291 sensor (running at 1280×720) will have a field of view of about 80°×45° (Figures A and B). In practice, however, you'll find that some nominal 4mm lenses are actually 3.6mm and you'll get a larger field (88°×47°). Other possible lenses and their properties are given in Table 1.

TABLE 1

Focal Length (mm)	f#	FOV[1] with IMX291 sensor	Pixel scale [2]	m @100km [3]	~M*$_{limit}$ [4]
4	0.95	90°×45°	3.8'	111	6
6	0.95	53°×30°	2.5'	73	6
8	0.9	40°×22°	1.9'	55	7–8
16	1.2	20°×11°	56"	27	8+
25	1.2	13°×7°	36"	17	10

TABLE 1. Commonly available lenses tested with the RMS running at 720p

1. FOV refers to the field-of-view
2. The pixel scale is calculated at 720p resolution
3. m @100km refers to the spatial resolution at a distance of 100km
4. M*$_{limit}$ is the limiting stellar magnitude

The Raspberry Pi 4B with a good SD card is a must. For our installations we typically use plastic or aluminum enclosures with a fan to keep the CPU cool. You can also 3D print your own RMS enclosure (Figure C), but don't forget the fan! Or, you can use an aluminum case which acts like a passive heatsink and has no fan. Apart from that, the rest of the design is flexible with the best choices often being the parts that are currently available at the best price.

SOFTWARE

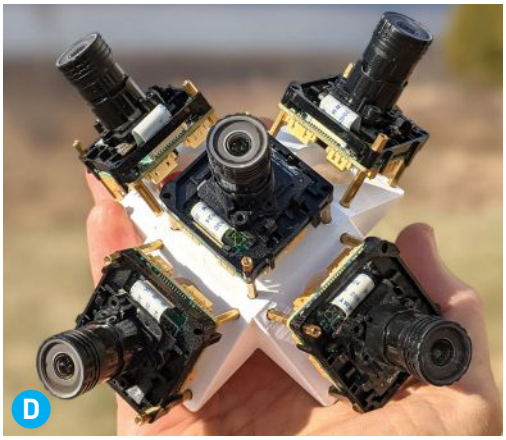
Written in Python, the RMS software is open source and provides a (nearly) complete toolkit for running a modern, professional meteor camera. Each night, RMS automatically starts up a half hour before astronomical twilight and turns off a half hour after dawn. The H.264 video from the camera is decoded and streamed through a real-time fireball detector and compressed into what is known as a Four-frame Temporal Pixel (FTP) file. This clever format saves blocks of 256 video frames as a set of four images: pixel maximum, maximum pixel frame number, pixel average, and the standard deviation.

Star and meteor detection then run on the average and standard deviation frames until the queue of FTP files has been fully processed. Afterward, the extracted stars are used to automatically update the image scale solution

An enclosure can be printed to house the camera as well as a small fan and heater.

M.Mazur

Like a mullet, it's all business up front, party in the back.



If one camera isn't enough, you can always experiment with multiple RMS cameras to achieve near all-sky coverage.

M. Mazur

A "tracked" stack of meteors from the 2020 Perseid meteor shower is quite dramatic. Notice how most of the meteor tracks point back to a single point; this is the shower ***radiant***.

and absolute brightness corrections for each image set so that the meteor detections can be properly calibrated. And then, once the results have been finalized, they're uploaded to the GMN server at Western University in Ontario where they're correlated with other nearby stations so that meteor orbits can be computed.

The RMS software is open, portable, and extensible. It should work with any IP camera that works with GStreamer, giving you a lot of opportunity to experiment (Figure D). And if you ever need help, an active GMN forum is full of experienced RMS operators who likely have answers to your questions.

BUILD YOUR RASPBERRY PI METEOR STATION

Assembling your camera is straightforward and should only take an hour or two. Here's an overview; for more details see our complete build instructions at makezine.com/go/RMS-build.

1. REMOVE THE IR FILTER

If your lens mount has an IR filter, you can push the filter out with a blunt object. Take care to protect your eyes as it will likely shatter.

2. MOUNT THE CAMERA

Attach the lens mount and lens to the camera module and then mount the camera assembly inside the weatherproof enclosure (Figure E).

3. CONNECT THE CABLE

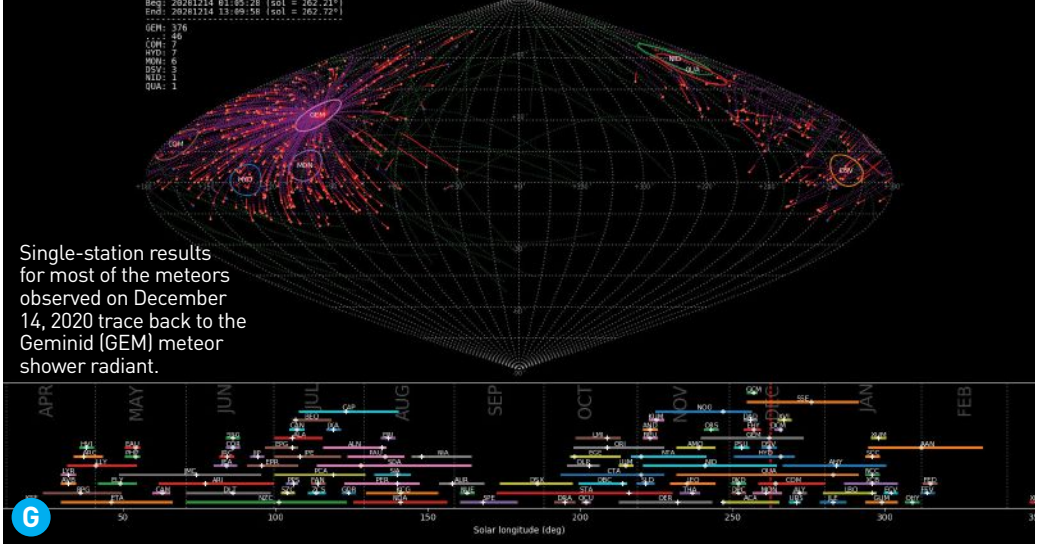
Run the CAT5 cable through the cable gland, wire up the connectors, and plug them into the camera module.

TIP: A pointy pin comes in handy after you inevitably assemble the cable connectors incorrectly and need to take them apart.

4. CONFIGURE THE CAMERA

The camera is ready for focusing and setup with a PC. Using CMS security cam software (hasecurity.com) or Internet Explorer (there's a blast from the past!), you'll turn off anything that automatically adjusts gain, exposure, and dynamic range.

Before you do this, though, you should focus the camera on a distant object. Then manually set



Single-station results for most of the meteors observed on December 14, 2020 trace back to the Geminid (GEM) meteor shower radiant.

your resolution to 720p (the Pi can't handle higher resolutions), your frame rate to 25fps, and your gain and exposure time.

Next, the camera's IP address needs to be set to 192.168.42.10.

5. SET UP THE RASPBERRY PI

Flash your SD card with one of the pre-built RMS images from makezine.com/go/RMS-images, or install the RMS code from scratch on an existing Raspbian installation, from github.com/CroatianMeteorNetwork/RMS.

Then boot the Pi, set up Wi-Fi, and answer the questions that appear in the **`RMS_FirstRun`** terminal window. Once you've done this, you'll make the necessary changes to the configuration file (see *READ-RPi4_note.txt* on the Desktop).

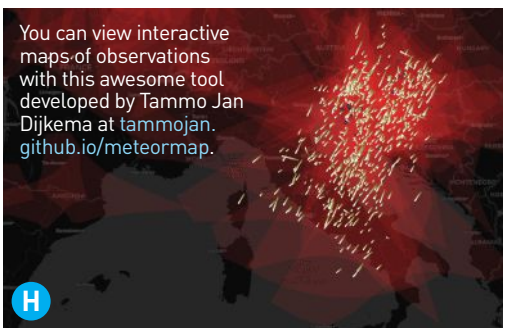
Finally, plug your camera into the POE injector and from there into the network port on the Pi. Give the camera a minute to boot and then check to see if it's visible with a **`ping 192.168.42.10`** or an **`arp -a`** command.

Reboot the Pi, and RMS will start up and wait for its run to begin automatically just before dusk. Easy peasy, lemon squeezy!

SHOOTING SHOOTING STARS

What can you do with your Raspberry Pi Meteor Station? As much or as little as you want! For complete instructions on aiming and using your RMS camera, read the Quick Start Guide at makezine.com/go/RMS-quick-start.

- **Hands-off:** In hands-off operation, the software takes care of collecting, processing, and uploading data to the GMN server where it is correlated with nearby stations to compute meteor orbits. And it automatically produces eye-catching meteor stacks, fireball files, and observation reports (Figures F and G) which you can share on social media.



You can view interactive maps of observations with this awesome tool developed by Tammo Jan Dijkema at tammojan.github.io/meteormap.

D. Vida, Tammo Jan Dijkema

On Tammo Jan Dijkema's GMN meteor visualization tool (tammojan.github.io/meteormap) you can also see all trajectories computed from your observations (Figure H).

- **Hands-on:** For a more hands-on approach, you might try extending the software to observe satellites, sprites, aurorae, and other atmospheric phenomena.

Or maybe you'll just use the data from bright fireball events to plan meteorite hunting trips.

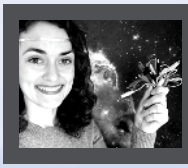
Either way, running an RMS is a fun and exciting way to get involved with cutting-edge meteor research.

Learn more at globalmeteornetwork.org, contribute at github.com/CroatianMeteorNetwork/RMS, and share your build at makeprojects.com/project/build-a-raspberry-pi-meteor-camera.

DIY Conservation Tech

Protect wildlife — or just learn your neighborhood creatures using mighty microcontrollers

Written by Helen Leigh



HELEN LEIGH is a hardware hacker who specializes in music technologies and craft-based electronics. Say hello to her on Twitter @helenleigh.

Open source time-lapse camera monitoring King penguins on South Georgia Island, Antarctica.

It's no secret that species and habitats all over the world are struggling, with global wildlife numbers dropping by over 60% since 1970. If we want to save our planet's wildlife we need to understand it, which is where conservation technology comes in. Scientists all over the world are working hard to monitor and protect our environment using the same kinds of technologies you might be experimenting with already, from turtle tags that use the Raspberry Pi Zero to monitoring elephants with an Arduino Nano.

RUGGEDNESS AND POWER

Designing and making technology intended to be used "in the field" has its own set of challenges. Making a reliable camera trap (a camera that automatically takes photographs when it senses an animal nearby) is tricky enough, but what if you want to install it somewhere that rains a lot? Electronics don't get along well with water, so you're going to need to make sure your device is housed in an appropriate enclosure. Or perhaps you're planning on tracking birds, but what if those birds go somewhere really cold, really high up, or really warm? Will your sensors work correctly?

One of the big things you need to think about when making technology designed for the outdoors is power. When you're setting up devices deep in the forest (or at the far edges of a suburban garden) you're not going to be able to plug your electronics into the wall. This means you need to think very hard about how much power you're using and how you're going to supply it. For DIY projects that could mean using batteries and calculating how often you'll need to switch them out, or perhaps you'd be better off adding in a solar panel.

Professional conservation technologists have the same kinds of issues. One team of researchers at ZSL London Zoo are trying to solve the power problem by developing a plant-powered camera trap that draws energy from soil microbes feeding off of a living plant's biowaste (Figure A). It took them 5 years to take their first photo powered by a microbial fuel cell, made possible by the biowaste of a fern called Pete, who you can visit in the rainforest section of London Zoo.

Plant-powered camera trap concept drawing from ZSL London Zoo

Alasdair Davies

OPEN TECHNOLOGY

One of the people who worked on the plant-powered camera trap is Alasdair Davies, a conservation technologist at Arribada who has worked on conservation projects all over the world, from monitoring King penguins in Antarctica (photo on page 88) to deploying science hardware in the Amazon rainforest. He designs and builds technology to help field researchers and conservation organizations monitor and conserve wildlife. Davies is a big believer in sharing what he has learned and how he made his technology. "Whatever I build is always open," he explains. "It's the best way to ensure that our community can reuse and share our successes and failures so we can help conserve wildlife and the environment together."

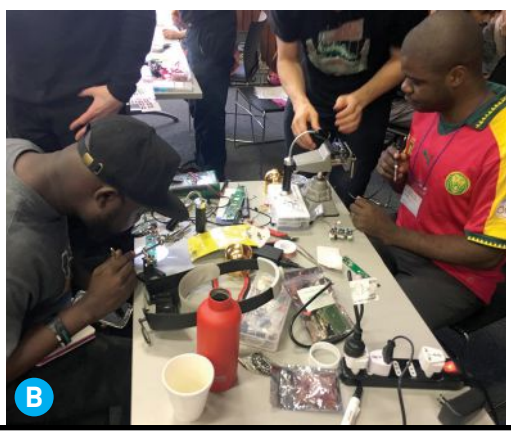
Openness in science has not always been the norm. When Galileo discovered the rings of Saturn he wrote notes to all the other scientists in his field telling them of his discovery. However, these notes were written in a secret code so that nobody could understand them: they were only

intended to be used as proof that he discovered them first, not as a way of sharing his exciting discovery. For a long time, scientific societies held the keys to sharing knowledge, followed by peer-reviewed journals in the scientific boom after the Second World War. In modern times many scientists and academics want to move away from these old ways of distributing information. They want to be able to share their research, data, and methodology freely with the public and other researchers. This move towards openness is reflected in the technology that scientists are using.

The problems that conservation technologists have to solve when designing and making a device are often the same problems that many others have had. Like the maker community, one reason people want to share designs and knowledge in an open way is to save others from reinventing the wheel over and over again.

Another reason is that many people want scientific equipment to become less exclusive, so that communities can run their own experiments. Shah Selbe, who runs the conservation technology company FieldKit, agrees: "Understanding our planet and how it is changing shouldn't be something that is only accessible to those flush with money or technical expertise."

Conservation technologists and academics from all around the world are working together to build ways to share and develop these kinds of open technologies. You'll find them at Gathering for Open Science Hardware (GOSH) events (Figure B), on the GOSH forum and wildlabs.net, and trading tips and sharing projects on Twitter.



MORE COOL CONSERVATION TECHNOLOGY

• **FIELDKIT** (Figure C) is an open-source, modular platform for environmental monitoring that includes dataloggers, radios, and sensor modules such as dissolved oxygen, water level, wind speed/direction, rainfall, temperature, air quality, and many more. Its makers are committed to sharing their learning so that communities around the world can research the issues that matter to them. fieldkit.org/about, twitter.com/fieldkitorg, instagram.com/fieldkitorg

• **AUDIOMOTH** (Figure D) is a small, inexpensive acoustic recorder used all over the world for capturing environmental and animal sounds. It's been used for bird and bat surveys, recording manatees, and tracking traffic noise during Covid-19 lockdowns. openacousticdevices.info/audiomoth and twitter.com/openacoustics

• **TURTLE TRACKER TO OCEAN MAPPER** Alasdair Davies added a Raspberry Pi Zero and camera module to a GPS turtle tag, letting him capture behavior below the waves. In a project for *National Geographic*, he remixed the turtle tag to give it an Argos satellite transmitter and fit it into a 500ml water bottle (Figure E) for tracking plastic pollution, and open-sourced the design so others can use it for their own conservation technology. arribada.org and twitter.com/Al2kA

• **ELEPHANTEDGE** was a program from Hackster.io and Smart Parks that challenged makers to build machine learning models for elephant collars, using Edge Impulse. Elephant abuse and killings are far too common but can be reduced with good data; contestants monitored elephant activity and poaching risk, with ML models deployed onto 10 collars. opencollar.io and hackster.io/contests/ElephantEdge

GOSH, Shah Selbe, AudioMoth, Arribada.org

PROJECT:

BUILD A #BIRDBOT TWITTER CAMERA TRAP

I recently moved to a different part of the world and I don't know anything about the local species of birds that hop around on my balcony. I wanted to make a project that would help me learn about the local wildlife. My friend Stephanie (@stephaniecodes) was posting cute pictures of her crow friend Orville on Twitter, so I decided to follow in her footsteps and make my own wildlife camera trap.

This project uses the Raspberry Pi Camera Module to record video and detect motion, along with a module called Twython that allows us to send text and media tweets from your Raspberry Pi using Python. You can copy all the project code examples from my Github repository at github.com/helenleigh/birdbot, and you can see my Birdbot's videos at twitter.com/helenleigh.

TIME REQUIRED: 1–2 Hours

DIFFICULTY: Intermediate

COST: $40–$80

WHAT YOU'LL NEED

- **Any Raspberry Pi mini computer with a camera module port** plus the usual monitor, keyboard, and mouse, for setup only
- **Raspberry Pi Camera Module**
- **Computer with internet connection**

WARNING: Make sure you don't share these keys, because they allow access to your Twitter account! If you do accidentally share or lose them you can regenerate them at developer.twitter.com.

1. SET UP TWITTER DEV ACCOUNT

First up, you need to get yourself a Twitter developer account. Head to developer.twitter.com, fill in the simple form, submit it, and verify your email when Twitter asks you to. It typically takes a day or two to get approved.

Once you're approved, head to developer.twitter.com. Navigate to Project & Apps → Overview, then click on the Create a Standalone App button. Give your app a name and click Complete. Copy and save the ***API key*** and the ***API key secret*** shown on the next screen then click through to manage your settings. Change the permissions to Read and Write, then save. Next, click on the Keys and Tokens tab and click the Generate button next to Access Token & Secret. Copy and save the ***access token*** and the ***access token secret***.

2. INSTALL SOFTWARE MODULES

Open a terminal window on your Raspberry Pi by clicking Menu → Accessories → Terminal, then type in this command and press Enter to install Twython, a handy module we'll use to talk to Twitter: **`sudo pip3 install twython`**

Once that's finished installing, enter one more command: **`sudo apt install -y gpac`**

This command installs MP4Box, which we'll use to convert the h264 video files from the camera module into MP4 video files that we can send to Twitter.

3. SEND A TEST TWEET

Next, open a Python 3 editor, such as the Thonny Python IDE (free from thonny.org). Create a new file and paste your saved keys as follows:

```
consumer_key        = '1234567890'
consumer_secret     = '1234567890'
access_token        = '1234567890'
access_token_secret = '1234567890'
```

Save this file as *keys.py*, then create another new file in the same folder called *tweettest.py*, with the code shown below.

```
from twython import Twython
from keys import (
    consumer_key,
    consumer_secret,
    access_token,
    access_token_secret
)

twitter = Twython(
    consumer_key,
    consumer_secret,
    access_token,
    access_token_secret
)

message = "Test tweet from my Pi using Python. Hello?"
twitter.update_status(status=message)
print("Tweeted: " + message)
```

In this new file, we are importing Twython from the **twython** module as well as the keys and tokens from *keys.py*. Next, we use those keys and tokens to connect to the Twitter API, before sending a test tweet. Change the text in my example message into something you'd like to say, then save your file and run your script. You've sent your first tweet using Python!

4. TEST THE CAMERA

With your Raspberry Pi turned off, connect the camera module. Gently pull up on the edges of the port's plastic clip, then insert the ribbon cable and push the clip back into place (Figure F).

Boot up your Raspberry Pi. Navigate to the main menu, click on Preferences, then open the Raspberry Pi Configuration tool. Select the Interfaces tab and ensure that the camera is enabled. Reboot your Raspberry Pi.

Open a Python 3 editor, such as the Thonny Python IDE. Create a new file and save it as *cameratest.py*. Enter the following code:

```
from picamera import PiCamera
camera = PiCamera()
camera.rotation = 180

camera.start_preview()
camera.start_recording('/home/pi/Desktop/testvideo.h264')
camera.wait_recording(3)
camera.stop_recording()
camera.stop_preview()
```

NOTE: Do not save any of your files as picamera.py or you may run into some issues!

If all is well, you should see video from your camera module on your screen for three seconds, then be able to navigate to your Desktop and open up *testvideo.h264*. You may need to delete or change line 3 of my code if your camera is oriented differently: the **180** in that line is the rotation I needed to stop my video being recorded upside down!

5. SET UP MOTION DETECTION

There are lots of ways to detect motion, but I narrowed them to two options: using a PIR sensor or the camera itself (Figure G).

A PIR (passive infrared) motion sensor detects movement of people or animals by looking for changes in the infrared radiation it picks up. If you've ever turned on lights by jumping around and waving your arms, you've probably been waving at a PIR sensor. As cool as these motion detecting sensors are, they aren't great at detecting movement through glass. As I wanted to keep my Pi indoors for this project, looking out through the window, I had to rule them out.

Instead, I decided to use the camera itself (plus a little math courtesy of the **numpy** module) to detect motion. The **picamera** module has a cool feature called **PiMotionAnalysis** that compares


F


G

Helen Leigh

and analyzes different frames from a video to detect motion. Here's the code I used, *motiontest.py*, to test if it worked:

```
import numpy as np
import picamera
import picamera.array

class DetectMotion(picamera.array.PiMotionAnalysis):
    def analyze(self, a):
        a = np.sqrt(
            np.square(a['x'].astype(np.float)) +
            np.square(a['y'].astype(np.float))
            ).clip(0, 255).astype(np.uint8)
        # If there're more than 10 vectors with a magnitude greater
        # than 60, then say we've detected motion
        if (a > 60).sum() > 10:
            print('Motion detected!')

with picamera.PiCamera() as camera:
    with DetectMotion(camera) as output:
        camera.resolution = (640, 480)
        camera.start_recording(
              '/dev/null', format='h264', motion_output=output)
        camera.wait_recording(30)
        camera.stop_recording()
```

You can play with the numbers on line 13 to adjust the sensitivity if you need to.

6. PUT IT ALL TOGETHER

You now have all the pieces of your jigsaw puzzle: a working camera, the ability to tweet, and the ability to detect motion. Now you'll put them all together in the final code.

Create a new file in your Python editor called *birdbot.py*, and import the camera, Twitter, and math modules you need, along with the keys and token from *keys.py*:

```
from picamera import PiCamera
import picamera.array
import numpy as np
import subprocess
from twython import Twython
from keys import (
    consumer_key,
    consumer_secret,
    access_token,
    access_token_secret
)
```

Next, add the code you need to detect motion, followed by the code to connect to the Twitter API:

```
class DetectMotion(picamera.array.PiMotionAnalysis):
    def __init__(self, camera, size=None):
        super(DetectMotion, self).__init__(camera, size)
        self.motionDetected = False
    def analyze(self, a):
        a = np.sqrt(
            np.square(a['x'].astype(np.float)) +
            np.square(a['y'].astype(np.float))
            ).clip(0, 255).astype(np.uint8)
        if (a > 80).sum() > 10:
            print("h")
            self.motionDetected = True

twitter = Twython(
    consumer_key,
    consumer_secret,
    access_token,
    access_token_secret
)
```

Finally, create a loop that records and analyzes 15 seconds of video (or however long you want). In my code, if motion is detected the program will print a message, run a subprocess to convert the h264 video into an MP4 (and tidy up any previous videos with the same name), then upload the MP4 to Twitter with a status update (Figure H).

Agonjaka / Wikimedia Commons

TWEET THOSE TWEETERS

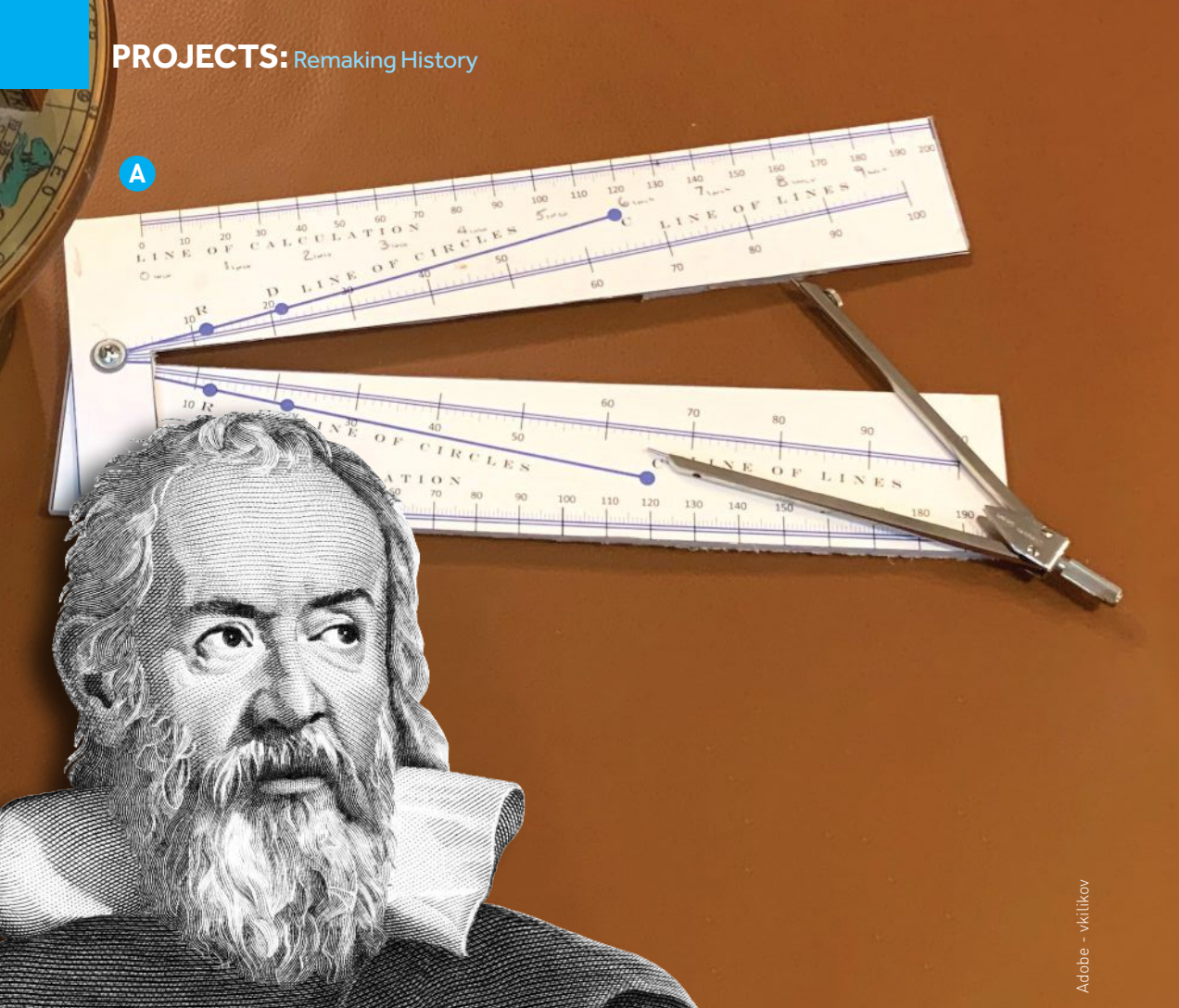
You now have your very own tweeting #birdbot! Each time a bird appears before the camera, it'll tweet video for you to share and study.

» Want to install it somewhere? Figure out how to make your code run when your Raspberry Pi boots up so that you can use your #birdbot without your screen or other peripherals.

» Tweeting too often? Try adding a delay, creating a Boolean flag, or comparing tweet times.

» Feeling ambitious? Extend this project with machine learning to ID the birds for you!

```
while True:
    with picamera.PiCamera() as camera:
        with DetectMotion(camera) as output:
            camera.resolution = (640, 480)
            camera.rotation = 180
            camera.start_recording('/home/pi/Desktop/isitabirdvideo.h264', format='h264', motion_output=output)
            camera.wait_recording(15)
            camera.stop_recording()
            if output.motionDetected == True:
                cp = subprocess.run(["rm /home/pi/Desktop/isitabirdvideo.mp4"],shell=True)
                cp = subprocess.run(["MP4Box -add /home/pi/Desktop/isitabirdvideo.h264 /home/pi/Desktop/isitabirdvideo.mp4"],shell=True)
                video = open('/home/pi/Desktop/isitabirdvideo.mp4', 'rb')
                response = twitter.upload_video(media=video, media_type='video/mp4')
                twitter.update_status(status='Birdbot activated: is this a bird? Do you know what type?', media_ids=[response['media_id']])
```



Adobe - vkilikov

Galileo's Sector

Make and use the simple but powerful mathematical tool behind Renaissance art, architecture, and artillery

Written and photographed by William Gurstelle

TIME REQUIRED: An Afternoon
DIFFICULTY: Easy
COST: About $15

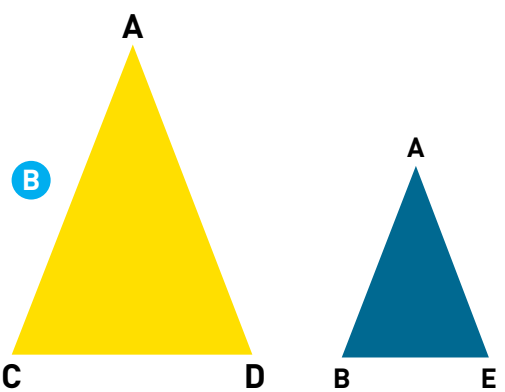
MATERIALS

- **Plastic sheet, plywood, or solid wood, ⅛" thick, 8"×12"**
- **Machine screw, #6×½", with wing nut and 2 washers**
- **Printed paper template** free download from makezine.com/go/sector

TOOLS

- **Ruler**
- **Glue or spray adhesive** for printed template
- **Fine tip marker or engraving tool (optional)** if you're not using the template
- **Drafting divider**
- **Band saw, jigsaw, or small hand saw**
- **Electric drill and 9⁄64" bit**

WILLIAM GURSTELLE's book series *Remaking History*, based on his *Make:* column of the same name, is available in the Maker Shed, makershed.com.



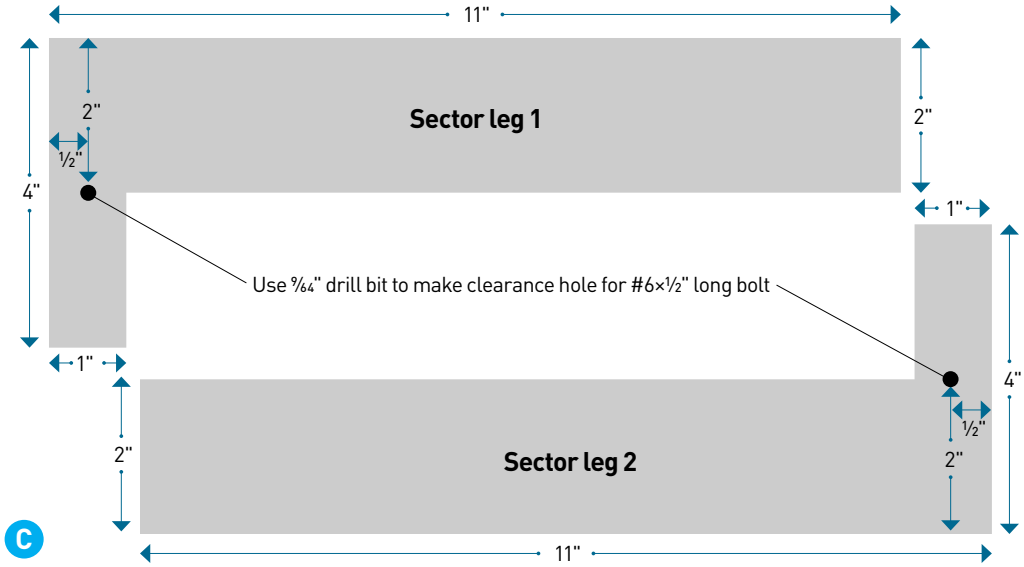
The sector, also known as the compass of proportion, is a fun and easy-to-use calculating device that uses the power of geometry to perform a variety of useful math jobs. Although at first glance a sector might seem as quaint and old-school as a slide rule, it can do at least a few maker-style jobs better, faster, and easier than any modern electronic calculator. Once you get the hang of using a sector, you'll find it a valuable tool in woodworking, drafting, surveying, and even doing mathematical calculations on the fly.

So, what is a sector? The one described here is a pair of rectangular legs, each about 10 inches long, connected by a rotating joint (Figure A). The numerical scales are inscribed on the legs. My sector is made of plastic, but originally sectors were made from wood, brass, or ivory.

Right up to the late 19th century, using a sector was *the* way to lay out woodworking or other fabrication projects. It enabled Renaissance scientists to make complex computations and it showed early cannoneers how to aim their guns. So, when you make and use a sector you are figuratively walking in the footsteps of millions of makers of the past.

Usually, the credit for its invention goes to the great 15th-century Italian scientist Galileo Galilei, although some historians of science give invention credit to Englishman Thomas Hood, a Galileo contemporary. In any event, it was certainly Galileo who made the sector popular. Almost immediately after it was invented, Galileo realized that this new gizmo was an extremely useful instrument, and he started teaching others how it worked. Galileo made quite a bit of money by writing and selling a book that explained how to use a sector as an arithmetical tool. For the next 500 years, the sector was one of the most commonly used mathematical instruments in Western civilization.

Its secret lay in the rules of geometry, specifically, in the simple fact that similar triangles have similar length ratios between their various sides. In Figure B, the two triangles *ABE* and *ACD* are similar (their angles are congruent), so by definition, the ratio of the length of the sides *AB* and *AC* to the ratio of the length of the bases *BE* and *CD* is identical. Written mathematically, the ratio of *AB* / *BE* = *AC* / *CD*.



C

This geometric truth means you can multiply, divide, find ratios, scale up, scale down, and so on, just by building or measuring similar triangles. And that's how the sector works; it's basically just a quick and easy way to build and scale similar triangles.

Making a sector is an easy and inexpensive project and something that parents and children can make together and have fun using even without a specific goal in mind. But don't let its relative simplicity fool you; the sector is useful to woodworkers, artists, metal workers, and others in a bunch of different ways.

So, let's get started. First, we'll fabricate the frame of the sector instrument. Then we'll make the all-important scales that are written, printed, or inscribed on the frame pieces. Finally, we'll use the sector to do different types of important mathematical jobs.

HOW TO MAKE THE SECTOR

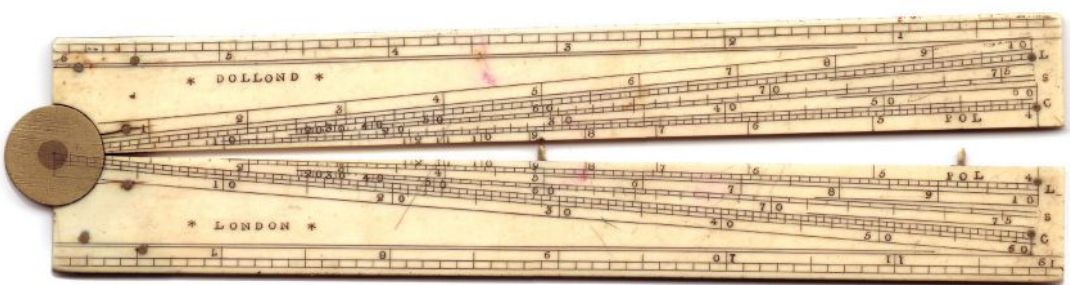
1. Use your saw to cut out the two L-shaped frame pieces as shown in Figure C. You can use thin plywood, solid wood, plastic, or any light, non-flexible sheet stock that you can cut into the shape shown. I used ⅛"-thick HDPE plastic and it worked great.

2. Drill a 9⁄64" diameter hole in each piece at the spot indicated in Figure C.

3. Insert a #6, ½"-long bolt with two washers into the hole and gently tighten with a wing nut.

HOW TO MARK THE SECTOR SCALES

The sectors made by the great European instrument makers of the 17th, 18th, and 19th centuries had a lot of different numerical scales etched into them. Each scale was used to



D

Wikipedia Commons

perform a different job, and each was given a special name. By far, the most important scale was the one that Galileo called the Line of Lines. The Line of Lines is the one you need to multiply, divide, and find proportions.

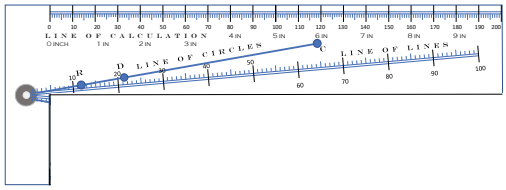
Another important scale is called the Line of Circles. With it, you can measure the radius or diameter of any circle and quickly determine its circumference. (It also works in the other direction, allowing you to find the diameter of a circle for a given circumference.)

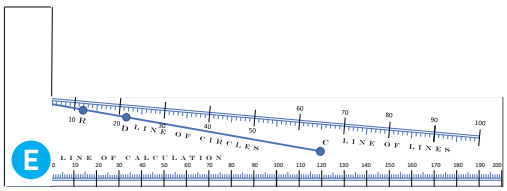
As Figure D shows, the sectors made by London instrument makers in the 18th century were chock full of scales or "lines." Many of them were rather specific in what they could do. For instance, the Line of Quadrature allowed engineers to calculate polygons of different side lengths that were equivalent in area. The Line of Inscribed Bodies was used to determine the length of sides of different polygons that were able to fit inside a circle of given diameter. There was even a Line of Metals to calculate the diameter of spheres made of different metals when all the spheres had the same weight.

Since drawing or inscribing accurate scales is time consuming, we'll limit our scales to just the three most useful: the line of lines, the line of circles, and the line of calculation. Figure E shows the layout of the scales we will use on the frame pieces.

You can draw the scales on the frame pieces using a fine-tipped marker, or if you're really ambitious, you can etch or engrave the scales. If these options seem like too much fine detail work, fear not because we've made a template that you can download at makezine.com/go/sector, print out, and affix to the frame pieces using glue or spray adhesive.

NOTE: Some printers will automatically scale the output of your print file to fit the margins of your paper. If that happens, the template won't fit your frame pieces. Either turn off the scaling feature of your printer or ask a local copy shop to print it out correctly for you.





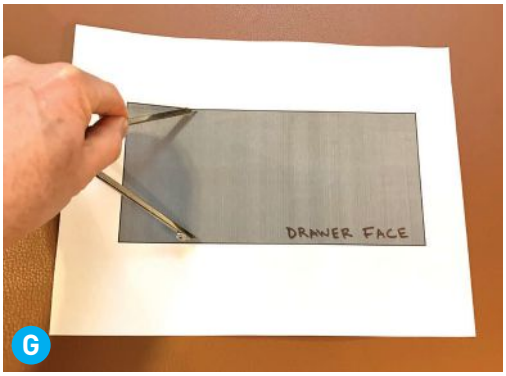
FUN WAYS TO USE YOUR SECTOR

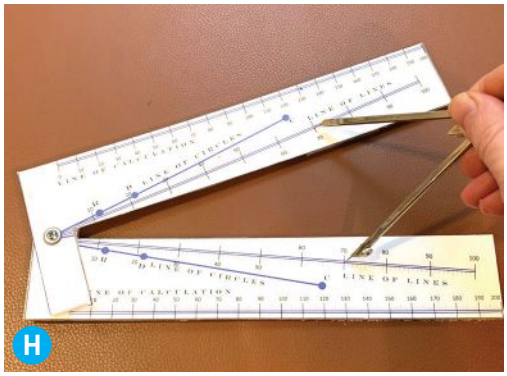
The easiest way to learn to use your sector is by looking at a few examples. If you follow along and understand the three examples below, you can substitute whatever numbers or line lengths you need when you use your sector on an actual problem. Also, to use a sector, you'll need a simple mechanical divider as well (Figure F).

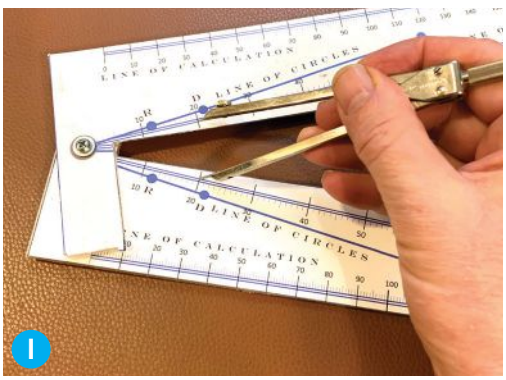
EXAMPLE 1. HOW TO DIVIDE LINES INTO FRACTIONS OF ANY LENGTH

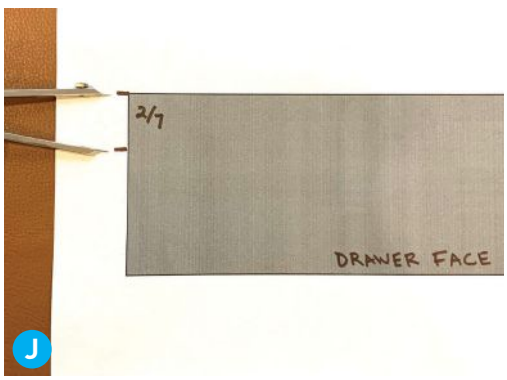
You can use a sector to easily divide a line into two parts of any ratio. Let's say you're building a chest of drawers and for aesthetic reasons you want to place the knobs on the drawer faces 2/7ths (or 20/70ths) of the way down from the top of each face. With a calculator and ruler, this can be time consuming, but with a sector, it's a snap!

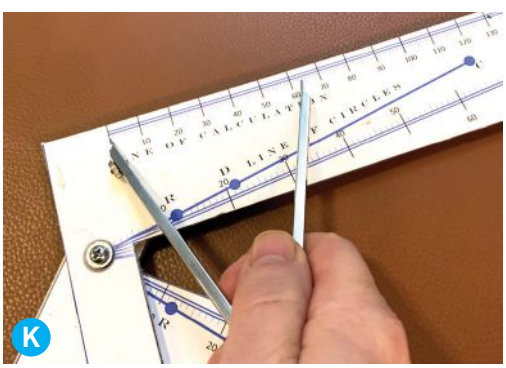
STEP 1: Open your divider so that one tip is on the start of the distance that you want to divide (in this example, the top of the drawer face), and the

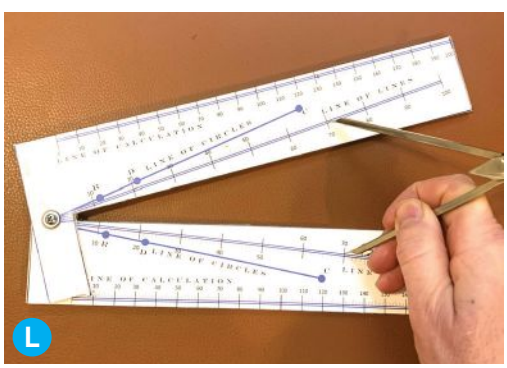












other tip is on the end of the distance (the bottom of the drawer face), as in Figure G.

STEP 2: Without changing the width of your divider, set one end of the divider on 70 on one of the Line of Lines. Open the jaws of the sector so you can set the other end of the divider on 70 on the other Line of Lines (Figure H).

STEP 3: Without changing the angle of the sector, adjust the tips of the divider so that they touch both Lines of Lines where the scale reads 20 (Figure I). Measure the distance between the divider tips. That distance is exactly 2/7ths of the total distance from the top of the drawer face to the bottom (Figure J).

EXAMPLE 2. FINDING A FRACTION OF A NUMBER

By using the same technique as in Example 1 plus the Line of Calculation, you can find fractions and proportions of numbers. For example, what is 2/7ths of 64?

STEP 1: Place one end of your divider on the zero point of the Line of Calculation. Place the other tip of the divider on 64 (Figure K).

STEP 2: Without changing the width of your divider, open the jaws of the sector so you can set both ends of the divider on 70 on the Line of Lines on each set of jaws of the sector (Figure L).

STEP 3: Place the tips of your divider on the Line of Lines where the scale reads 20 (Figure M). Move the divider to the Line of Calculation. Place one tip on the zero point and the other tip on the line, where it points to the answer to the problem, or 18.3 (Figure N).

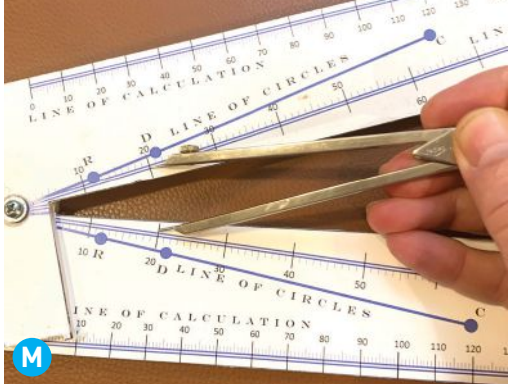
EXAMPLE 3. FINDING A CIRCLE'S DIAMETER OR CIRCUMFERENCE

The Line of Circles makes it easy to find the radius, diameter, and circumference of any circle if you already know any one of those items. For example, what is the circumference of a U.S. quarter coin?

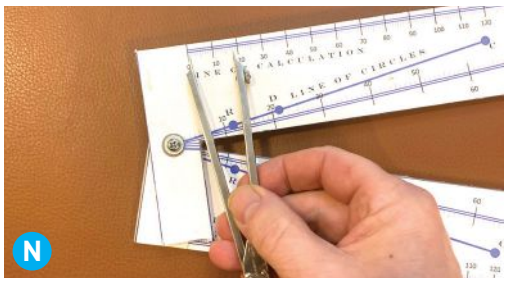
STEP 1: In this example, we use the Line of Circles. Take your divider and open the jaws so the quarter just fits between the tips. This is the diameter of the quarter (Figure O).

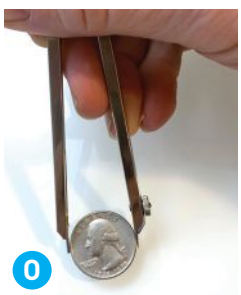
STEP 2: Without moving the divider tips, adjust the sector so that you can place the tips of the divider on the dots marked "D" (for "diameter") on the Line of Circles on both jaws of the sector (Figure P).

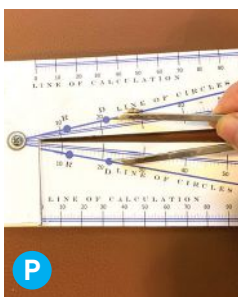
STEP 3: Without moving the sector jaws, place the tips of the divider on the points marked "C" (for "circumference") on the Line of Circles (Figure Q). Place one tip of the divider on the zero point of the Line of Calculation and read the answer from the spot where the other tip touches the Line of Calculation (Figure R) — in this case, about 3 inches.

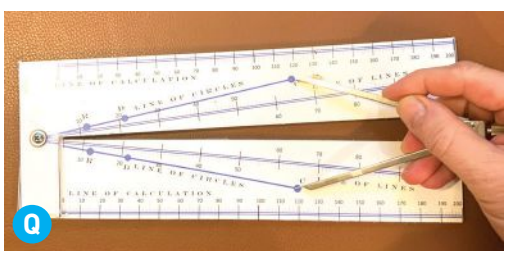


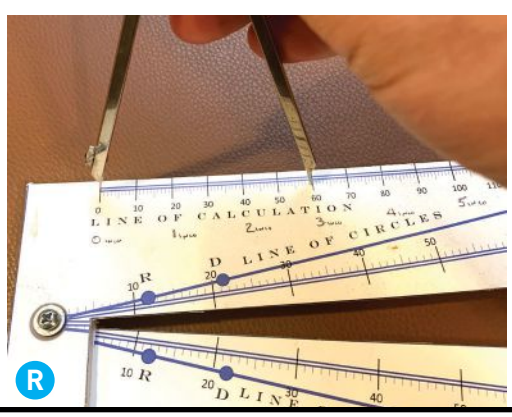
M


N


O

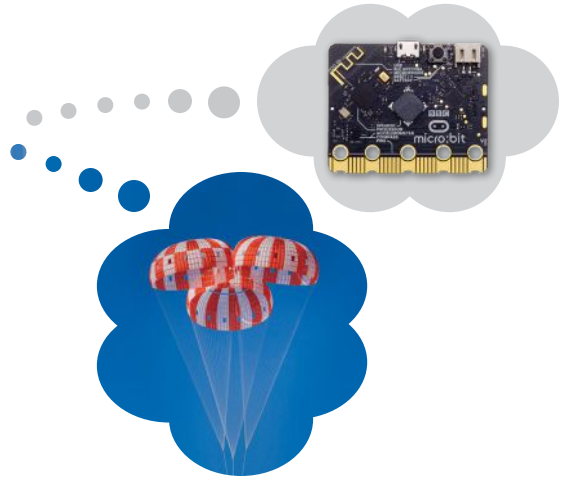

P


Q



R

Written by Rick Schertle and Keith Violette



Next-Level Air Rockets

You can help design the next upgrades to this popular rocket kit

It's been 13 years since the Compressed Air Rocket hit the pages of *Make:* in 2008 (Volume 15, makezine.com/projects/compressed-air-rocket). Since then, a lot has happened. Let's look at how the launcher has developed — and how you can help take it to the next level!

MAKER TO MAKER PRO

Soon after that first article, the Compressed Air Rocket Launcher was turned into a popular kit sold in the Maker Shed. Then in *Make:* Volume 31, Rick showed how to build an historic balsa wood folding-wing glider (/folding-wing-glider-rockets-up-glides-down) — which caught the eye of young

Justin Cross, Gabriela Hasbun, Keith Violette

reader Sean Violette and his dad, Keith — and in Volume 32 a catapult to launch it (/catapult-glider-launcher). The Violettes soon reimagined combining the glider with the air rockets!

Keith shared a prototype with Rick, and we collaborated remotely on the Air Rocket Glider in *Make:* Volume 39 (/air-rocket-glider). With wings folded back, this one-of-a-kind invention is launched by compressed air, then its wings fold out at apogee and it glides gently to the ground. In 2014 we met for the first time at World Maker Faire in New York, where we and our families built and launched 1,600 rockets with faire-goers.

Led by Keith, the launcher went through several design changes which cut the build time by hours and the parts count from 40 to 12. After a successful Kickstarter campaign, we formed our company, Air Rocket Works. We've now sold our launcher and accessories in all 50 states and dozens of countries. Keith even built a custom launch system for NASA and ULA, and traveled with his family to Kennedy Space Center in 2018 for NASA Family Day to see it in action.

HOW ABOUT CHUTES AND ELECTRONICS?

A few years ago, we designed a foam nose cone, a thin and durable ABS plastic body tube, and vinyl sticker fins for our Compressed Air Rocket Bounce (CARB). Using our standard launcher, a CARB will travel nearly 250 feet in the air and create an exciting bounce upon landing.

Now we see these parts as a platform for innovation and we're asking *Make:* readers: What might come next? We're seeking innovators to help build on ideas we've been playing with, like:

- **Parachute recovery —** timer-based or airspeed-released? This deceptively simple design challenge of making a reliable recovery system has eluded us for years. With parachute recovery, the CARB could perform like an Estes Alpha with a solid-fuel A engine — but not burn a $4 hole in your pocket each time you launch!
- **Electronics —** possibly with the ubiquitous micro:bit — namely accelerometers to measure launch and landing g-forces, time of flight, tone generators (for experiencing Doppler effect), cameras, etc. But be warned, the g-forces on launch can exceed 100 g's!

Rick's first article in *Make:* Volume 15, and Rick at World Maker Faire New York with some of the awards we won.

The system Keith built for NASA KSC Launch Services Program educational outreach. The entire design was vetted by NASA engineers. These cases originally held ignitors for the Space Shuttle solid rocket boosters.

Submit your ideas via the form at Make: Projects (makezine.com/go/air-rocket-recovery) and we'll send a 4-pack of modular rocket parts (nose cones, fins, and body tubes) to the first 25 entries we choose. The parts are for you to experiment with. Your invention must be lightweight, fit the rocket, and most of all, be reliable. With your help, we look forward to writing the next chapter of air rocket history.

RICK SCHERTLE, a public school K-8 Maker Lab teacher in San Jose, California, is author of two books for Make:. His family has brought rocket joy to Maker Faires across the USA.

KEITH VIOLETTE is a proud father of a maker family, a tool-a-holic, and a mechanical design engineer for robotic prosthetics at DEKA R&D in Manchester, New Hampshire.

Get your Air Rocket Launcher at makershed.com/products/compressed-air-rocket-launcher-v2-2.

New From Make: Projects

See some of our favorite builds from makeprojects.com — and share your own!

Written by Dan Schneiderman

DAN SCHNEIDERMAN is a maker, space enthusiast, co-chair of Maker Faire Rochester, and Community Manager at Make:.

SPIROBIKE

Ted Kinsman @TedKinsman

More than just the wheels spin on this bike. A specially designed spirograph is mounted to the front of the bicycle. Based on your gear ratios, drive radius, pen distance, pen radius, and a handful of other variables, new mathematical patterns are drawn as you pedal through your neighborhood.

makeprojects.com/project/the-spirobike-a-bicycle-that-makes-spirograph-designs

A MICRO BINARY CLOCK

C Forde @uk4dshouse

Taking a byte out of digital clocks, this micro binary clock will help you keep track of the time through the use of micro:bit's display LEDs and a Kitronik RTC (Real Time Clock) expansion board. makeprojects.com/project/micro-binary-clock

B FABSCAN PI 3D SCANNER

Mario Lukas @mariolukas

Scan small everyday items with the FabScan Pi. Started out of Germany's first FabLab in Aachen, the FabScan is a stand-alone, web-enabled, open source DIY 3D scanner. It's also maker friendly with a laser-cut base and powered with a Raspberry Pi. makeprojects.com/project/fabscan-pi---an-open-source-diy-3d-scanner

C RGB GLASSES

Arnov Sharma @Oniichan_is_ded

Slip on a pair of these WS2812B LED-based RGB glasses for that perfect lit look. Instead of plastic frames, these glasses are built using custom designed PCBs. Thanks to the ESP8266 chip on the board, you can change your look through the web on the fly. makeprojects.com/project/rgb-glasses

D DUNGEONS AND DRAGONS PLAYER JOURNALS

Making Fun @MakingFun

You don't have to roll a natural 20 to know that these custom Dungeons and Dragons player journals are a critical hit. No cantrips or spell slots were needed to create these journals, just a bit of craft foam, mini-binders, and Mod Podge. makeprojects.com/project/dungeons-and-dragons-player-journals

E HANGOUT MACHINE

Lenny Leiter @lenny3000

Even when you physically can't be home for the holidays, you can still have an immersive experience of being there! This pivoting table imitates your head's movements replicating the action of looking at a family member as you enjoy dinner and conversation. makeprojects.com/project/a-christmas-hangout-machine

BOB KNETZGER is a designer/inventor/musician whose award-winning toys have been featured on *The Tonight Show*, *Nightline*, and *Good Morning America*. He is the author of *Make: Fun!*, available at makershed.com and fine bookstores.

Fun With Pop-Up Stamps

Written and photographed by Bob Knetzger

Re-create this trick postage stamp to make pop-up toys and cards

Even in an age of emails and texts, stamp collecting is still a favorite hobby of adults and kids. You can explore and learn about lots of things: geography, history ... and toys. There are lots of U.S. postage stamps that commemorate classic toys (Figures A through D), and some novelty stamps are toy-like and fun in themselves!

These USPS stamps (Figure E) are scented with the real scratch-and-sniff aroma of ice cream pops — fragrant philately!

This 2012 stamp from the Netherlands (Figure F) was made to commemorate a Children's Book Fair. This gummed stamp is a working pop-up toy. The cleverly engineered three-layer design has a top layer with the stamp graphics, a middle layer with a "sled" between two side guides, and a base layer. When pulled, the sled slides along, bending and folding a flap, which pops up revealing a cut-out shape (Figures G, H, and I). When pushed, the stamp goes flat again.

You can make your own pop-up stamp like I did.

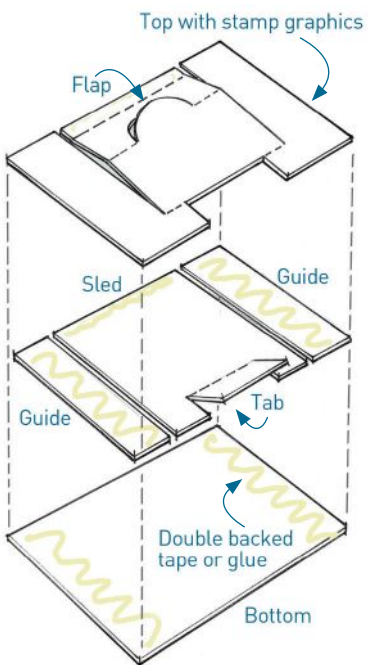
I used a USPS commemorative stamp of a Hot Wheels car on some heavy paper (previous page).

First, choose which part of your image you'll make "pop up." With a sharp hobby knife cut two slits to make a flap, then cut around your pop-up shape. Gently score on the fold lines with a nail or hard pencil. For best pop-up effect, the middle fold line should be centered between the other two folds.

Cut out a sled and two side guides from more heavy paper. Make the sled a little wider than the width of the flap and cut two slits to make a little pull tab. Use double-backed tape or glue to fasten the guides onto the base so the sled slides smoothly between them.

Then tape the guides to the top and use a tiny strip of tape to fasten the flap to the very end of the sled. Cut and glue a matching tab from the stamp to cover the sled tab. Lastly, trim all three layers to a neat, final size.

You can make this sliding pop-up design with any size printed image: greeting card, photograph, drawing. Go wherever it leads you!

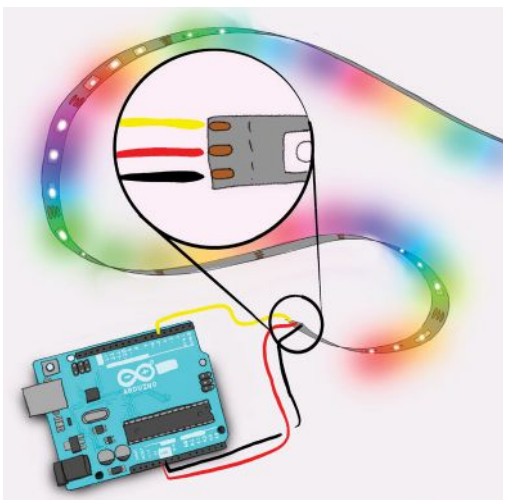


Find more fun novelty stamps with "Exploring Stamps" videos:
youtube.com/watch?v=6a3EFIkdDt8

1+2+3 Arduino Borealis

Add some arctic ambience to your home with these northern-esque lights

Written by Mike Senese



Caleb Kraft

I've always wanted to experience the northern lights, but my only trip to a suitable latitude occurred in the summer, when the sky stayed too light to see them. Then I came across this pseudo-effect rendition, shared with us by Reddit users u/Eduhsoj, u/Mazen191, and u/Ringkeeper.

1. **Connect** the Arduino to the lights. Plug the data wire from the LED strip into pin 5 on the Arduino; the 5V and GND wires go to 5V and GND pins on the Arduino. Hot-glue them if they slip out. For reliability, add a 470Ω resistor between pin 5 and the lights. Also, a dedicated 5V power supply with a 1,000µF capacitor will keep you from burning out your Arduino (see makezine.com/go/arduino-borealis).
2. **Download** the sketch and configure it to match the number of LEDs in your strip. There are various settings you can modify to alter the light effects. Upload the sketch to the Arduino.
3. **Set** the lights where they aren't directly visible, but their glow can easily be seen. I put mine atop my kitchen cabinets.

Power it up and the lights will come to life with waves of purples, pinks, and greens. If you have excess strip, loop it and watch the aurora effect dance with itself as it moves. Beautiful!

MIKE SENESE is the executive editor of *Make:*.

TIME REQUIRED: 30 Minutes
DIFFICULTY: Easy
COST: $50–$60

MATERIALS

- **Arduino Uno microcontroller** or similar
- **LED string lights, individually addressable, WS2812B** long enough to stretch along the area where you want to create the aurora effect. I used a 16.4' spool containing 160 lights.
- **5V power supply with barrel plug, or USB cable and power dongle**
- **Jumper wires, male/female**
- **Resistor, 470Ω**
- **Capacitor, 1,000µF, 5V**

TOOLS

- **Soldering iron (optional)**
- **Wire strippers (optional)**
- **Hot glue gun (optional)**
- **Computer with Arduino IDE software**
- **Arduino Borealis sketch** github.com/Mazn1191/Arduino-Borealis

1+2+3 Home Phone Fun

Assemble a simple intercom system using your existing landline wiring

Written and photographed by Stephen Cooke

My two daughters wanted a way to communicate with each other from their rooms; I wanted to get them something that didn't have a screen or require batteries. I discovered a device called a ringdown emulator, which simulates a phone line, complete with dial tones and ringing. I realized I could repurpose our old landline phone wiring with this to create an intercom for them. A win-win for everyone!

NOTE: Be sure that your phone wiring is not in use — including for home alarm systems, internet (DSL), or otherwise.

1. **Isolate the wiring**. Our house has older cat3 wiring; the unmarked wires connect to all the phone jacks around the house. I used an old phone cable, a 9V battery, and a multimeter to find out which cable went to which room. You could also use a network tester for this.
2. **Install any necessary pieces**. I added missing RJ11 plugs to each room. If your house wiring is correct it should be red and green for the first phone line. (You could also run new wiring between the areas you want to place the phones, although this would defeat the beneficial "reuse" aspect of the project.)
3. **Plug in!** I then connected the emulator and plugged the phones into the wall jacks. This does the "ringdown," so that when one phone is taken off the hook the other will ring.

From there, you can experiment with multiple simulator boxes, multiple lines, and more. The phones can be any style, with or without dials. Look for cool old phones on eBay — we used vintage Soviet-era ones for ours. Have fun!

TIME REQUIRED: 1 Hour

DIFFICULTY: Easy

COST: $100–$200

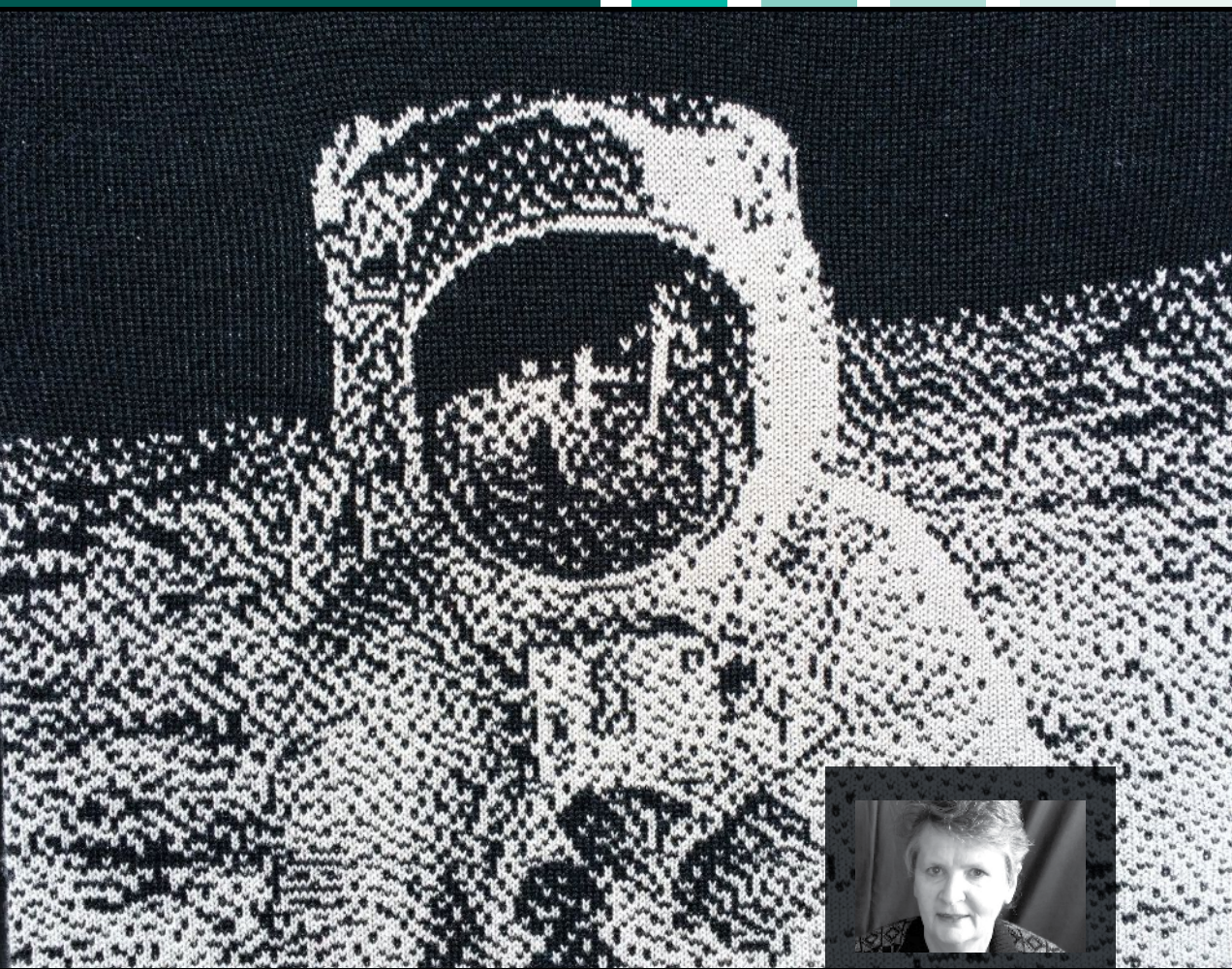
MATERIALS

- **Two-way phone simulator box** Viking DLE-200B. Find it locally or on eBay or Amazon.
- **Old phones (2)** These can be pushbutton or no-button style
- **Phone-to-wall cord**
- **RJ11 jacks (2)** (if needed)

TOOLS

- **Multimeter**
- **9V battery**
- **RJ45/11 crimp tool** (if needed)
- **Wire stripper**

STEPHEN COOKE is an educator, researcher, and maker living in Rhode Island.



ADRIENNE HUNTER is a lifelong crafter and maker, mostly in textiles. As a fun complement to her day job in tech, she exhibits machine knitting at Maker Faires and knitting events along with friends from the Machine Knitters Guild of the San Francisco Bay Area.

PIXEL KNITTING

Hack a classic 1980s machine to knit your own 2-color designs

Written and photographed by Adrienne Hunter

Hobby knitting machines from the 1980s are pretty affordable and fun to use, but they need old-fashioned punch cards or scanned-in sheets or floppy drives to accept a pattern. Luckily an amazing community of knitters and makers has developed hacks for these machines that let you bypass the clunky old input methods and simply transfer an image file from your computer.

There are two main hacks for implementing these updates: ***AYAB*** which is Arduino-based, and ***img2Track*** which uses the port intended for a Tandy floppy drive. Both hacks let you easily knit an image or pattern, one pixel per knitted stitch.

In this article we'll look at how they both work, how to select a vintage knitting machine, and how to set up the AYAB hack. Then we'll guide you to two simple first projects using your hacked machine: a two-color Makey puppet, and a knit slouch hat.

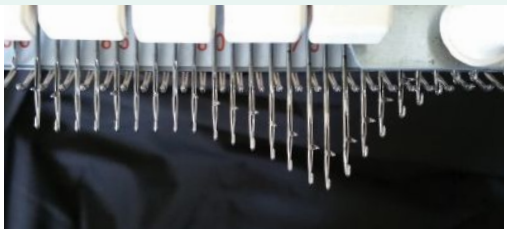
EVOLUTION OF KNITTING MACHINES

Knitting machines have been around for over 400 years — Queen Elizabeth I refused a patent application in the 1590s! They became a cottage industry before the Industrial Revolution, and the mechanical concept used by those early machines to form the stitches is still in use in some industrial machines today.

Today's domestic knitting machines, along with most industrial ones, use a needle design from the 1840s. The needle has a ***latch*** that flips back and forth to facilitate the knitting action, which pulls the new stitch through the old, one needle per stitch. The yarn is fed to one needle after the other in turn, with each needle moving forward to catch the yarn, then back again, in a wave motion.

Circular machines for making socks appeared in the 1860s, and were in use in both homes and factories during World War I. There was a ***flatbed*** version that could make additional items, from a company called Lamb. Larger flatbed machines came on the home market by the 1930s and were widely available by the 1950s. But all of these just did plain knitting; any patterning had to be selected by hand.

During the 1960s each manufacturer developed their own innovative ways to semi-automate patterning, then in 1970 ***punch card*** machines came onto the market, eclipsing all



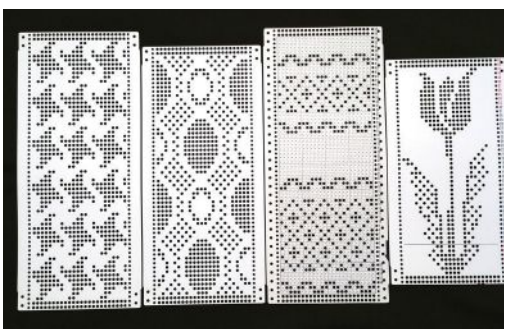
The needles take a wave-like path as the yarn is fed to each needle in turn.

A stitch is born: The needle in action, with its latch open.

An antique Lamb flatbed knitting machine.

A recent aluminum reproduction sock knitting machine; the early ones were made of cast iron.

Cool 1970s punch card technology, but cannot take the current hacks. The back of the work is facing you as you knit, finished sample shown on the left.



Sample punch cards. The hand-punched card for our hat project is second from the right.

other patterning methods and causing a boom in machine knitting. Punch cards are a direct descendant from the very first automated patterning on weaving looms, back around 1800. Another boost happened in the early 1980s when the patterning moved to using electronics; these are the machines we'll be using here.

Enthusiasm waned in the 1990s and all the major companies stopped manufacturing home/hobby machines by 1999. The sole survivor was Silver Reed, later joined by Taitexma and Artisan which now make clones of the older Brother and Silver Reed punch card machines. However, the old machines were very solidly built and have lasted well, and parts for the Brother machines are generally still available because most of them are common with the Taitexma punch card machine. These domestic/hobby machines were also used for small-scale production and prototyping for industry, and for teaching fashion students. Today almost all commercial production uses industrial machines.

AUTOMATIC PATTERNING — HOW IT WORKS

Let's look more closely at the Brother machines, designed and manufactured in Japan from the 1960s to 1990s. (Other manufacturers get the same result using completely different mechanics.) The ***carriage*** knits the currently set-up row, and at the same time sets up the ***needles*** for the following row. It's a binary process: the needles end up in one of two positions on the ***bed***, and are then picked up separately by the two colors of yarn. (Other

textures are also possible from these two positions, but we'll focus on two-color.)

Punch card patterning starts from a lever that either pokes through a hole in the card, or is blocked by the card. ***Electronic patterning*** has a row of 16 solenoids instead. Then from here on, the patterning works in the same way. A rotating ***camshaft*** transfers the hole/no-hole information to a set of eight ***needle selecting plates*** under the bed, which move to put a small ***hook*** into position lined up with the back of each needle. The carriage is connected to this mechanism by a ***belt***, and as it passes over the needles, it pushes down on them in turn and, depending on whether it is hooked or not, guides the needle into one of its two positions, ready to be picked up on the next row by the two colors. Each plate controls every 8th needle at the same time, but only the one that's under the carriage at the right time has any effect.

A punch card has 24 hole positions per row, so those same 24 stitches are mechanically repeated across the bed as the camshaft rotates. The electronically patterned machines can refresh the data fed to the solenoids as soon as it's been used, making each needle individually addressable, so you can do a non-repeating pattern such as a dithered photograph. The hat project I'll show you here is a repeating pattern so you could actually make it on either a punch card machine or an electronic. (You can still buy blank cards to punch your own pattern, or make cards using a die-cut machine.)

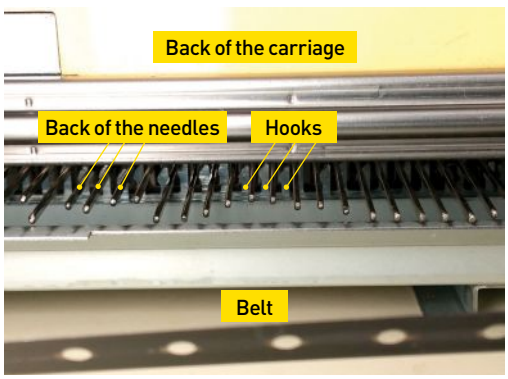
The first of the electronic machines (Brother KH910) used a 60-stitch optical card known as a ***mylar*** for its data input. So it wasn't completely individually addressable, though with mirroring and other smart use of built-in features you could overcome a lot of the 60-stitch repeat limitation. The AYAB hack removes this limitation altogether.

The later models are all completely electronic with no width limitations up to the 200 available needles; the KH950 is a hybrid that can read the mylars or take a full-width pattern. Fun fact: The Brother 910 came on the US market in 1980, so its development work must have happened in 1978–79, making it an early use of a microprocessor.

Most of the later electronic models have a

Solenoids and camshaft in a Brother electronic knitting machine.

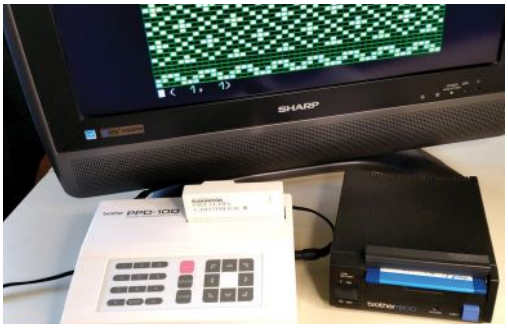
These needle selecting plates are under the needles; when engaged, their hooks push against the needles, causing the carriage to guide the needle to one path or the other.



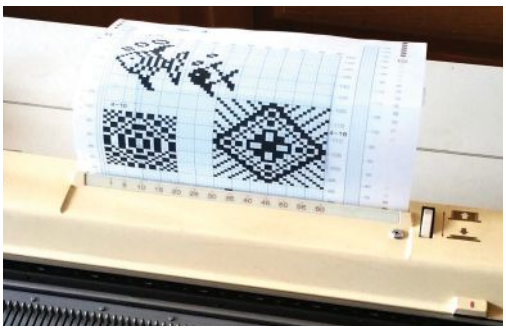
The hooks in position under the needles.

connector that allows you to connect a ***floppy drive*** to store and retrieve pattern files; this interface is used in one of the two hacks. The drive they chose was manufactured by Tandy, and uses Tandy's proprietary disk format. The disks themselves are the familiar 3.5-inch floppies, but only these old Brother drives will read or write them.

Using this same connector Brother also made a ***Pattern Programming Device (PPD)*** which attaches to a TV, allowing you to view your patterns and download them to your knitting machine. The UI is very clunky by today's



PPD and Tandy floppy drive. The PPD box attaches to any analog TV, and is operated by buttons and a menu.



Some models used these optically read mylar sheets. After the hacks, you won't need them.

A

Use img2Track to download a pattern from your computer to the knitting machine's onboard memory.

standards, but it was advanced for its day.

Commercial software came on the market in the late 1980s that you can use with your knitting machine connected to your computer. Updated versions of DesignaKnit by Soft Byte (softbyte.co.uk) and Garment Designer by Cochenille (cochenille.com) are still available today.

Note that *electronic* refers to how the patterning is done. The actual knitting is still a human-operated process: you push the carriage back and forth manually.

FLOPPY PORT HACK: PYTHON CODE, THEN IMG2TRACK

In the early 2000s, some knitters came together in a Yahoo group to discuss how to reverse engineer the internal file format of the Brother patterns, and the interface to the floppy drive. The end result was some code mostly written by Steve Conklin and adapted by Limor Fried and Becky Stern. Becky's 2010 video for *Craft:* magazine (youtube.com/watch?v=vKpdFIlbqSY) shows the result (Figure A), along with instructions that she and Limor published on Adafruit (learn.adafruit.com/electroknit).

You make a simple image file on your computer, one pixel per stitch, and send it to the knitting machine over an FTDI cable that's been converted to plug into the floppy port. Once the pattern is downloaded the cable can be disconnected and the knitting machine operates as always. I have the Adafruit code working with my KH930, though it took a few small code changes. Some other open source projects have extended this code to use a KH950i (github.com/kentfield/knitting_machine and github.com/chixor/knitting_machine).

But most knitters who are using the floppy hack use the commercial app called img2Track (daviworks.com/knitting). Based on the above concepts, img2Track was completely rewritten and covers a range of the Brother machines with a nice GUI, good instructions, and a built-in dithering utility for picture knitting.

I used img2Track's instructions to build my cable. You can also buy a ready-built cable from img2Track. That's it. You only need this altered cable; you don't make any changes to the knitting machine itself.

AYAB HACK: A NEW ARDUINO BRAIN

The AYAB hack (All Yarns Are Beautiful, ayab-knitting.com) takes a different approach. You remove the entire 1980s electronics unit from the knitting machine and replace it with an Arduino and associated circuitry. Then a simple one-pixel-per-stitch image is sent down one row at a time, with the Arduino firmware directly controlling the existing mechanics in the knitting machine as you knit. It was developed in Germany in 2013 and got a mention on Hackaday at the time.

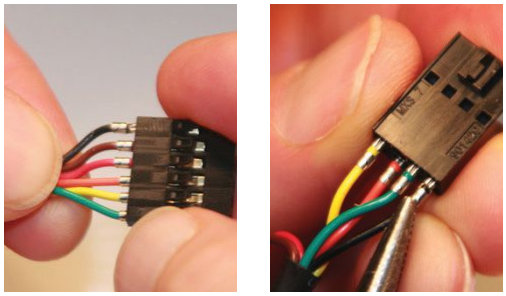
One big advantage is that AYAB does not depend on the floppy connector, so it can be used with the Brother KH910, which was the earliest model of the series, and was the only one available for about five years so a lot of them were sold. Which means that they're more widely available and cheaper than the later models. While they have lasted well, the mechanism that reads the mylar pattern sheets is often the first part to fail. AYAB is the perfect solution, as it bypasses the need for the mylar patterning.

Both the hardware and the software for AYAB are open source. There are two versions of the hardware: the first design is a ***shield*** for a standard Arduino. The second version, known as the ***interface***, was developed by Evil Mad Scientist Labs; its long, narrow design fits more neatly into the knitting machines. It has its own power distribution and a daughter-card Arduino (actually an Adafruit Metro Mini or similar, running Arduino code). Otherwise the two hardware versions are functionally identical.

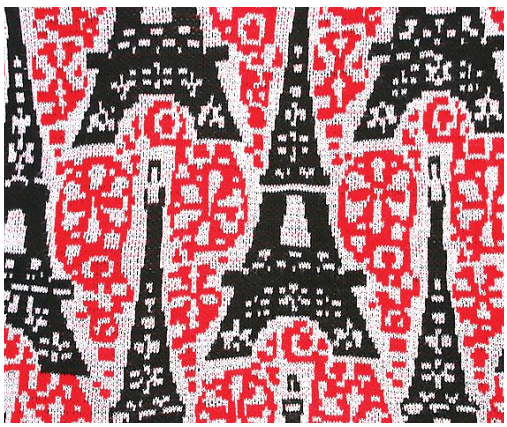
While the software was basically fully functional from the start, there have been improvements along the way, and there's more we still want it to do. It's an all-volunteer project, so the development effort has been sporadic.

If you have a machine that can do either hack, which to choose? Much as I'm a fan of AYAB, I usually steer knitters toward img2Track if that's an option. Anyone who mainly just wants to knit rather than to tinker, and owns a fully functioning knitting machine that has the floppy port, is likely to prefer img2Track.

Making the image file for either hack can be as simple as zooming in to a file and adding your design pixel by pixel. Or use Gimp, Photoshop, etc. to make a more complex design up to 200 pixels wide. This is usually a black image on a white background, though there are ways of incorporating more than two colors — we'll talk more about that in a future article. You can also scale and dither photos directly in img2track.



The img2Track app simply requires hacking a 6-wire FTDI cable to make a 4-wire floppy connector.

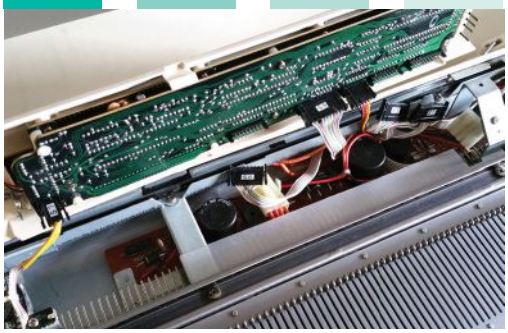


One of the many wonderful machine knits created by img2Track developer Tanya Cunningham.

img2Track Tanya Cunningham

For the AYAB hack, you can use either the interface from EMSL (top) or the Arduino shield (bottom).

1. Remove the existing control unit.



2. Unplug the cables.

3. Plug the cables into your AYAB board.

4. Plug in the USB and power into your interface (or shield).

INSTALLING AYAB

Since the AYAB-ready machines are cheaper and more available, let's look at how to install the AYAB hack.

As it's open source hardware, you could build it all yourself, but usually you buy an AYAB board from one of three small suppliers, then unplug some connectors in your knitting machine and plug the new board in instead. For the shield version you attach the new board to the machine's power; for the interface version, plug in an external power brick. The whole process is completely reversible; you don't need to cut or solder any connections in your machine, so you can put its original electronics back at any time.

Then you download the software from ayab-knitting.com and install it on Mac, Windows, or Linux. You'll also need drivers, either the standard Arduino driver or a driver from SiLabs for the interface hardware. Plug in a standard USB cable, download the Arduino firmware, and you're ready to open an image file and knit.

To watch a video of the AYAB installation process, please visit my Makey Puppet project at makeprojects.com/project/knit-a-puppet-using-a-hacked-knitting-machine.

KNITERATE

You may have noticed a Kickstarter a few years ago for a new knitting machine that fills the gap between hobby machines and huge, super-expensive industrial machines. The Kniterate machines began shipping in 2020, and reports so far are looking good. Due to its size and price ($14K), this is not a machine that most hobbyists will want, it's more for small-scale production and prototyping of knitwear, or for colleges and makerspaces. A possible scenario is a business where the customer can design their sweater online and the Kniterate machine knits the pieces. The knitting and shaping is a lot more automated than the Brother hobby machines, but some sewing up and hand finishing is still needed to complete a garment. kniterate.com

YOUR FIRST KNITTING MACHINE PROJECT

With AYAB installed, you're ready to knit! You can easily draw your own pixel art in a drawing program, and I found it irresistible to experiment with dithering photos (Figure B).

If you'd like a pattern that's ready to go, here are two great first projects for you to try:


B

MAKEY PUPPET

We altered Makey's shape to give it puppet-style arms, and knitted it in a two-color Fair Isle pattern (Figure C). Instructions are included for how to make neat edges on a single-motif pattern like this. The project also includes videos to guide you through both the AYAB installation and the knitting of the Makey-patterned puppet.

• makeprojects.com/project/knit-a-puppet-using-a-hacked-knitting-machine

C

KNIT SLOUCH HAT

This hat (Figure D) has a deep double layer hem, no ribber required. The multiple colors are from a random-dyed yarn; all except the black is actually all one strand. We'll use the same yarns for a matching scarf in a future article. It is knitted as a single strand on the single bed machine, making a fairly lightweight fabric — this won't be the warmest hat you've ever owned, but you can make it warmer by using a heavier yarn. The top is slightly shaped by reducing the number of stitches by half, so the gathers aren't too bulky.

• makezine.com/go/knit-hat-hacked-machine


D


E

NEXT UP: DOUBLE BED JACQUARD

In a future article we'll add an attachment called a *ribber* and use it to do a double-layer fabric that lies flat, a technique known as *double bed jacquard* (Figure E). We'll explore how 2-color double-knit patterning is straightforward but 3 or more colors per row presents challenges, and how one engineer has come up with a new solution. We'll also touch on the optional motors that can automate the back and forth of the actual knitting, and how yarn-changing is handled.

ACKNOWLEDGMENTS

- Knitting machine designed and manufactured by **Brother Industries**.
- AYAB hack designed by **Christian Obersteiner** and **Andreas Müller** with coding help from multiple people on **Github**.
- Alternative AYAB hardware design by **Evil Mad Scientist Labs**.
- Floppy-based hack by **Steve Conklin**, **Limor Fried**, **Becky Stern**, **John Hogerhuis**, **Brian Redbeard**, **Travis Goodspeed**, **Fabienne Haas**, **Sally Kentfield**, **Sarah Spencer**, and more.
- Img2Track coded by **Davi Post** in collaboration with **Tanya Cunningham**.

LEARN MORE:

Make: makezine.com/2010/11/05/hack_your_knitting_machine

Ravelry: (free account required) ravelry.com/groups/machine-knitting, ravelry.com/groups/ayab, and ravelry.com/groups/img2track---for-machine-knitters

Facebook: search groups for "Machine Knitting," "Img2track for Machine Knitters," and "AYAB All Yarns Are Beautiful"

HOW TO BUY AND REFURBISH A KNITTING MACHINE

First, be aware that there's a learning curve to these machines, whether or not you already know how to hand knit (which is helpful but not required); this is a different skill. There's lots of new information to take in, and new manual skills to learn as you figure out to how to use the hand tools to pick up a dropped stitch or decrease a stitch at the end of a row.

To use these hacks, you're looking for a **Brother standard gauge electronic knitting machine, models KH9xx**. Look around on Craigslist, Facebook Marketplace, or eBay, or join an online or (better still) a local machine knitting group and ask if anyone has a machine for sale. Local purchase is preferable; knitting machines often get damaged in the mail.

- **For the floppy port hack,** you're looking specifically for a 930, 940, 950i, 965i or 970.
- **For the AYAB hack,** get a 900, 910, 930, 940, 950, 950i, 965, or 965i, or CK35.

Some machines have a letter "e" such as a KH940e, the "e" is of no consequence, it just refers to the color. But the "i" designation in the 950i and 965i is important, it signifies the presence of the floppy connector. In the United States these same Brother machines were also sold branded as KnitKing, for example the KnitKing Compuknit III is identical to the Brother 930. Prices in the US have risen a bit in recent years, so expect to pay $250–$600 or more, depending on model and condition. Try to buy a machine that's complete with most of the tools and small parts that came with it originally, though most parts are readily available, and some can be 3D printed. If you buy from outside your home country, pay attention to the voltage, you may need a transformer.

If you want to buy a ***ribber*** too (which you'll need for a future project in *Make:*) look for a

KR850 or KR900. A KR830 is OK too, but lacks a couple of features compared to the 850 or 900.

Most machines that come up for sale haven't been used in many years. While most of the plastic has held up remarkably well, avoid carrying the machine by its plastic handle as it can break. The most important thing to know about refurbishing is to not try to knit before you check the ***sponge bar***, aka ***needle retainer bar***. That's a metal bar that slides out from the end of the bed with a strip of foam attached — its job is to sit on top of the needles and provide springiness and just enough friction. The foam deteriorates with time and should be replaced every few years. You can buy whole sponge bars new, or buy just the sponge and glue it onto the cleaned-out metal strip (Figure F).

The mechanism under the carriage consists of spring-loaded ***flippers*** which can seize up with time; these usually recover with cleaning and lubrication. You'll need denatured alcohol for cleaning and a light synthetic oil for lubrication. Do not use WD-40.

The Machine Knitting group on Ravelry has a permanent thread about cleaning and refurbishing that's a good place to ask questions. Or try Machine Knitting groups on Facebook.


F

TALK TO THE WORLD

TIM DEAGAN makes, breaks, and collects things in Austin, Texas. He loves using software to breathe life into hardware, using Linux and any microcontroller he can get his hands on.

Digital voice puts ham radio over the internet to let you communicate clearly around the globe

Written and photographed by Tim Deagan

A revolution is occurring in amateur radio that opens up new opportunities for low-cost worldwide conversations. With only an entry-level amateur license, a budget handheld radio, and a Raspberry Pi with a radio hat, you can now talk to hams all over the world. This article will introduce **amateur digital voice (DV)** technologies so that you can get started in this exciting aspect of the ham radio hobby.

Traditional analog voice tech has been losing ground to digital voice technologies for years. Voice over Internet Protocol (VoIP), audio and video streaming, cellphones — they all perform analog-to-digital conversion of audio signals. As digital data, it can be packetized, routed, and manipulated in ways that aren't feasible with analog signals. Amateur radio, also known as ham radio, started experimenting with digital modes like Radio Teletype (RTTY) as far back as 1946, though arguably Continuous Wave (CW), aka Morse Code, is the original digital mode and has been with us since the inception of amateur radio in 1912. But as the internet gained prominence, so too did ham radio's interest in advanced digital modes including digitized audio.

Like many aspects of the amateur service, many options exist for amateur digital voice. The three most common are: **Digital Mobile Radio** (DMR), **Digital Smart Technology** (aka D-Star), and Yaesu's **System Fusion** (aka Fusion or YSF) (Figure A). All three share a high level architecture and are interoperable to varying degrees. While each mode has strong proponents, I'll generally focus on DMR for this introduction.

A

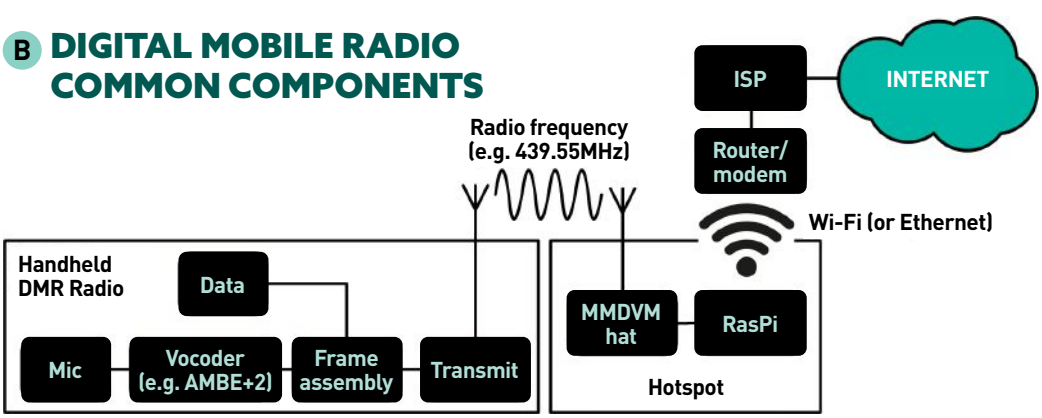
PROPAGATING OVER THE INTERNET

Worldwide analog radio works by sending out (propagating) oscillating waves of electromagnetic radiation that bounce off the ionosphere to cover long distances. These radio waves have to compete with artificial and natural sources of noise to be intelligible when received. Digital voice avoids this noise by using the internet to send digitized audio around the world. Radios with special codec chips encode and decode the digital signals, but typically are only used to make short-distance transmissions to a radio-enabled access point, either a local repeater or a personal hotspot (Figure B). This assures the quality of the audio when it makes a local hop from a repeater or hotspot to the receiving radio.

The majority of DV activity occurs on VHF/UHF bands (2m and 70cm), with local, line-of-sight propagation. However, some HF radios (160m–10m) have DV capabilities and can propagate signals much farther via the ionosphere. We'll stick to the more common VHF/UHF DV solutions for this introduction.

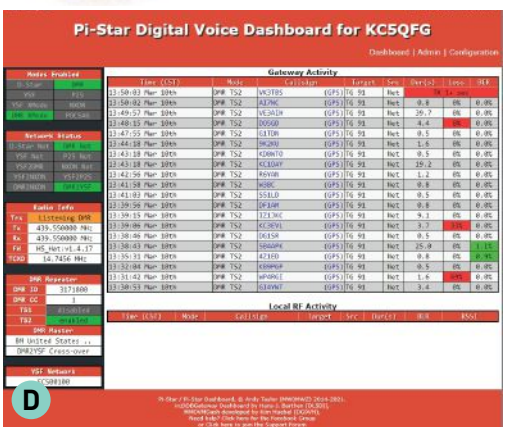
The minimal things needed to get started in DV are; an entry-level Technician Class amateur radio license, a **handheld transceiver (HT)** with a codec chip, and a local repeater or personal DV hotspots.

B DIGITAL MOBILE RADIO COMMON COMPONENTS



Getting a **Technician license** (and callsign!) is easy and worthwhile. It requires correctly answering at least 26 out of 35 multiple-choice questions drawn from a published pool of 428, plus a $15 FCC fee. Classes, prep books, and practice exams make it possible for just about anyone to get their ticket.

C



D

HTs with DMR capability start in price from $70 and also work as normal analog VHF/UHF radios. My TYT MD-UV380 (Figure C) is at the bottom of the price range but has been a great DMR radio for me. Unfortunately, the current radios on the market only work with one kind of DV mode, but this can be overcome at the hotspot.

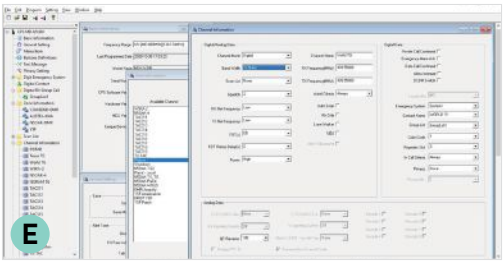
Local ham clubs may have a DV repeater available in your area which you can use if you're within range (typically around 5–6 miles depending on the terrain and power of your HT), but it's very common to set up your own hotspot to connect to. This is typically a Raspberry Pi (frequently a Zero W) with a **Multi Mode Digital Voice Modem (MMDVM)** hat. I've seen fully outfitted hotspots as low as $35, though $80 is a more common price point for a cased Pi Zero, OLED display, MMDVM hat, and antenna. You can build your own or find a variety of other solutions at a wide range of price points.

DV hotspots need an internet connection. This is usually Wi-Fi, but can be wired or even tethered via a phone's shared Wi-Fi for use on the go. The most common software for the hotspot is an open-source toolset calld **Pi-Star** (Figure D). Pi-Star (pistar.uk) manages connections, configuration, logs, and troubleshooting tools, and can provide bridges between different DV systems. I set mine up so that my DMR HT can also talk to YSF systems. Configuring Pi-Star has all the familiar "joys" of working with open-source software on Linux, but numerous web resources exist to help.

CODEPLUGS AND TALKGROUPS AND CHANNELS, OH MY!

The HT itself is configured with what's known as a **codeplug** (Figure E). This is a file containing all the config information for the radio, generally prepared on a computer and uploaded to the HT. Most radios provide a means to manage config on the HT itself, but few hams are masochistic enough to want to try this very often. The codeplug includes radio settings, analog repeater settings (frequency, CTCSS tones, etc.), and a wealth of information about talkgroups.

Talkgroups (TG) are essentially channels. They can be local, regional, worldwide, topic-specific, or ad-hoc between users. Once set up, you can



simply scroll to the desired TG, push the PTT button, and start or join conversations. Calling "CQ" (i.e., requesting someone to talk to you) on the WorldWide TG and having someone in Indonesia respond with crystal-clear audio is a wonderful experience.

The configuration of the bewildering array of things needed for a codeplug is perhaps the most unpleasant thing about DV systems. Codeplugs can be shared or found online, which can be helpful, and online resources provide varyingly useful documentation. If you're familiar with complicated and frustratingly documented, but massively powerful open source/Linux toolsets, you'll be right at home. Once it's all configured and working, it's extremely easy to actually use.

WHAT'S IN IT FOR YOU?

The question comes up, why bother? Why not just use some internet-connected tool that doesn't bother with radio transmissions and licenses?

My answer may not satisfy everyone. Calling CQ is a chance to talk to a safe stranger. Meeting new people and having interesting conversations is something lots of people want to do. Internet solutions have traditionally failed on the "safe" part. Chat Roulette was arguably a similar, and initially popular model, but rapidly turned into a running joke due to spam and inappropriate users. The ham license needed for amateur DV serves as a means to assure that the people you talk to aren't going to try and sell you anything, hit on you, stalk you, or turn into pervs. They've also established some minimal mutual level of interest (in amateur DV at the very least) and there are 100-year-old protocols for how to make good contacts.

This article only scratches the surface. Amateur DV solutions provide a relatively low-cost method to get into ham radio and talk to the world, even if you live in a high-rise apartment in an urban core and can't put up a big HF antenna. The future of this technology is going to explode and it's a great opportunity for makers with a background in software to contribute and shape that future. Free and Open Source Software (FOSS) projects like **FreeDV** (freedv.org), a project to enable HF radios to use DV via a connected computer sound card, are great examples of how the FOSS movement and ham radio have become deeply intertwined. Combine that with the long history of amateur radio kit building, increasingly using Arduinos and other microcontrollers, and the opportunities for makers become unlimited.

AMATEUR DIGITAL VOICE PROJECTS FOR MAKERS

- **Get licensed:** arrl.org/getting-your-technician-license
- **Build a digital hotspot:** w4gso.org/wp-content/uploads/2018/09/How-to-build-a-digital-hotspot-on-the-cheap.pdf
- **Build a DMR repeater:** rs-online.com/designspark/building-a-digital-mobile-radio-repeater-part-1-mmdvm
- **M17, an open-source DMR-like system:** kb6nu.com/m17-an-open-source-dmr-like-system
- **DIY digital voice modem:** hackaday.com/2016/03/23/homebrew-multimode-digital-voice-modem
- **Pi-Star Github**: github.com/AndyTaylorTweet
- **Building VHF/UHF antennas:** wikihow.com/Build-Several-Easy-Antennas-for-Amateur-Radio

myCobot $579 shop.elephantrobotics.com

Robot arms are often either hobby-servo-driven and somewhat crude, or bigger and tougher, but still with exposed belts, gears, and pinch points.

MyCobot comes from a collaboration between an industrial arm manufacturer, Elephant Robotics, and M5 Stack, producing something that both looks fantastic and is very easy to use. Within seconds of pulling it out of the box, I was recording and playing back simple motions.

Unfortunately, going further than that proved to be a daunting task. Documentation is extremely sparse, so if you're not already familiar with Robot Operating System (ROS), or the M5 UIFlow, you'll need to study up.

Also note that the bot is fairly small and has a maximum payload limit of 250g, so don't expect to be doing anything too productive with it. For context, a GoPro is about 120g and a can of soda is about 370g.

Overall, MyCobot is a solid and high quality physical product, and the price tag is quite affordable for a robotic arm. I just hope they can flesh out the software experience to make it easier for beginners. —*Caleb Kraft*

Creality CR-30 The Print Mill

$999 creality3dofficial.com

Creality's latest offering in the 3D printer market provides a new and interesting angle, literally. By canting the print rails by 45° and replacing the print bed with a conveyor belt, the CR-30 adds two stand-out capabilities that make this machine really shine: assembly line automation and printing beyond a traditional bed size. With the CR-30, you can automate multiple productions of a file, using the conveyor belt to move them out of the way, and ultimately off the machine. Or, you can utilize the moving bed to achieve what is effectively an infinite print axis. This latter feature means you can print really long items — cosplay swords and such.

There are a few rough edges to the user experience, but overall I got acceptable prints without much tinkering.

If you need to churn out prints or absolutely need to print large models, it's a must-have at this price point. But for everyone else, your mileage may vary.

—Caleb Kraft

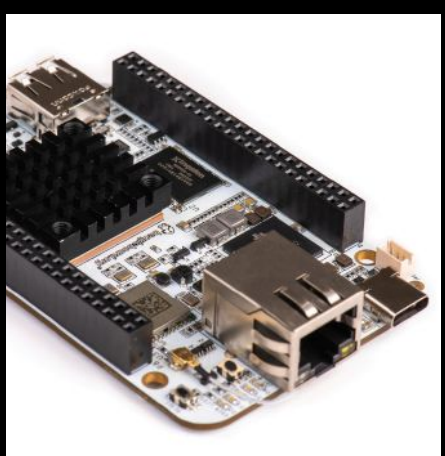
More details and a video overview:
makezine.com/go/creality-belt-printer

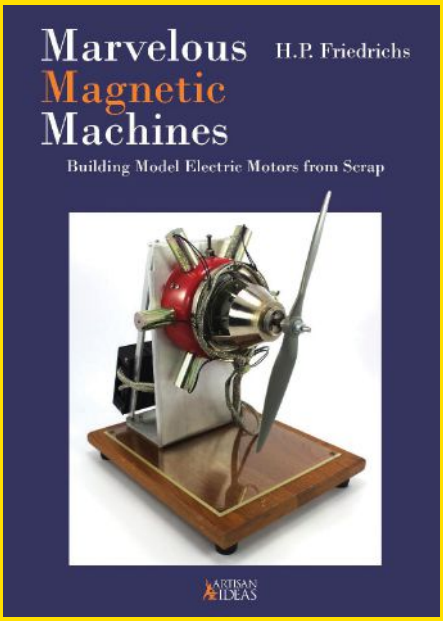
BeagleBone AI

$128 beagleboard.org

Based around the powerful Texas Instruments Sitara AM5729 processor, the BeagleBone AI includes 1GB DD3 RAM, 16GB eMMC flash on-board, 72 GPIO pins, and runs off of USB 3.0 Type-C. The board has a lot of flexibility with IoT projects and caters to both machine learning applications and industrial robots. With the AM5729, the BeagleBone AI has some unique features. One, in particular, is the use of TI's programmable real-time units (PRUs). For any application that needs high accuracy and time-critical operations, these can be used without having a dedicated FPGA or specialized microcontroller board to do so, cutting down the layers and costs of a project that requires low latency.

The BeagleBone AI comes with four Embedded-Vision-Engine (EVE) machine learning cores. These cores are supported by an optimized TI Deep Learning framework (TIDL) with OpenCL API and pre-installed software tools. The TIDL API combined with PRUs is a deadly combo for precise industrial robot applications. The Sitara AM57x SoC line is geared towards making smart factories. I would hardly call the BeagleBone AI a typical SBC. *—Mel Ho*





Marvelous Magnetic Machines: Building Model Electric Motors From Scrap

$30 artisanideas.com

I have a soft spot for things built from junk, and when those things happen to be precision electric motors I can't but help get extra inspired.

In this book, author H.P. Freidrichs breaks down the basics of how to build motors from things you can often find at garage sales and other second hand markets. To really nail home how cool this is, simply look at the cover. Yup, that's a magnetic motor built around a kitchen pot.

Even if you never actually build the items in this book, you'll learn a lot and be thoroughly amused by the author's enthusiastic use of odds and ends.

—Caleb Kraft

Luxonis Oak-D AI Camera

$299 luxonis.com

The Oak-D consists of three cameras: two dedicated to stereo vision that help identify the exact location of objects in 3D space (much like how our eyes work), and a third that provides a 4K color image. Inside is an Intel MyriadX chip, which synthesizes these images and provides machine learning inference processing. This allows the Oak-D to recognize objects, facial expressions, or whatever machine learning model the user chooses.

It also means that while you will still need a computer to run the camera, the MyriadX can handle most of the heavy lifting. Instead of a desktop with a high-performance graphics card, you can run it with a Raspberry Pi. I tried the Oak-D on a Mac laptop and Pi 3, with both systems performing at about the same frames per second.

The flexibility of using multiple systems for development makes for an easier workflow; you can create and tweak code on a laptop or desktop and move to a single board computer like the Pi when you are ready to deploy. Luxonis provides walkthroughs for getting started on Mac, Pi, other Linux systems, as well as Windows machines. This device is a powerful and flexible way of seeing the world around you. *—Kelly Egan*

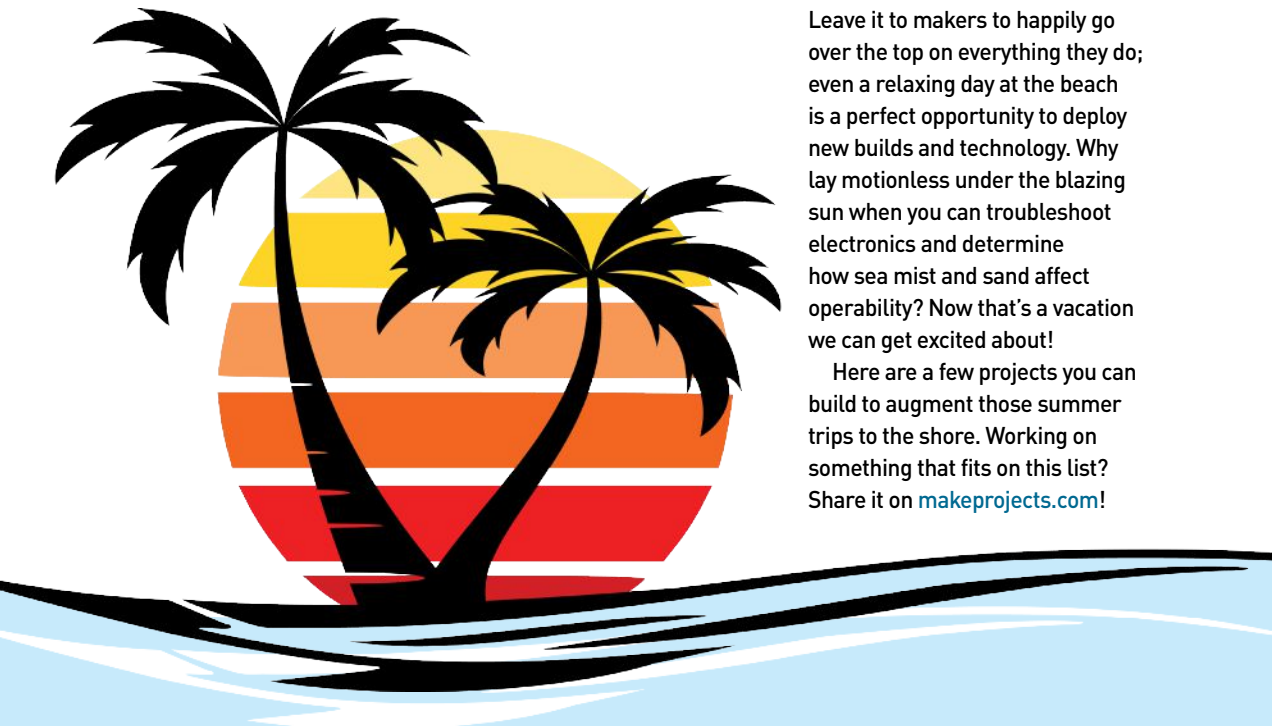
Adventure Kit: 30 Days Lost in Space $50 inventr.io

Learning how to construct and code electronic circuits using an Arduino can feel like a daunting task, but Inventor.io has come up with an approach that is really fun and interesting — they've created a whole sci-fi narrative around leaning to code. You start off lost in space, in a broken ship, and you need to create circuits, each more complex than the one before, in order to return home. Over 30 days you'll go from nothing to being able to create a push button interface, display system, and interactivity using an Arduino.

The hardware itself is nothing out of the ordinary, a self-branded Arduino clone and a handful of necessary parts to complete the lessons. It's the lessons that really shine. Framing everything within the story really helps give meaning and context to the circuits and code, something that is often missing from educational kits. The lessons are video-based, presented as broadcasts from home base to you in the broken ship. They're also clear and approachable, and the whole thing comes together nicely.

I feel that they pulled off their goal for this one, and I'd recommend it for any sci-fi fan that wants to dive into Arduino. I'd even love to see them lean harder on the sci-fi narrative, acting out the story even more. —*Caleb Kraft*

GEEKY BEACH GADGETS



Leave it to makers to happily go over the top on everything they do; even a relaxing day at the beach is a perfect opportunity to deploy new builds and technology. Why lay motionless under the blazing sun when you can troubleshoot electronics and determine how sea mist and sand affect operability? Now that's a vacation we can get excited about!

Here are a few projects you can build to augment those summer trips to the shore. Working on something that fits on this list? Share it on makeprojects.com!

1. SAND WRITER Out: Writing your name in the sand with a stick. In: Programming a crawler robot to scrawl messages of any sort for you. youtube.com/watch?v=7T1esQgRwrM

2. AI SURFLINE WAVE DETECTOR There's nothing worse than getting skunked because of a bunk wave report. Next time, train an artificial intelligence setup with thousands of wave images to get the smartest surf analysis yet. instructables.com/BAIwatch-AI-and-Surf

3. BEACH CLEANING 'BOT Ultrasonic sensors and a scooper to clear your space of awful rocks and even-worse litter. instructables.com/Beach-Cleaning-Robot-garbage-Collector-/

4. FOLLOW-ME COOLER Why would anyone choose to lug a brick of ice and plastic through the sand, when they could build a basic droid that does the job for them? instructables.com/Follow-Me-Cooler-on-the-Beach

5. MINIMAL METAL DETECTOR The design may not be ideal for finding treasure — but maybe the gadget itself *is* the treasure. instructables.com/Minimal-Arduino-Metal-Detector

6. LORA WEATHER BUOY How's the water? This buoy will beam you the conditions so you don't have to swim out to figure it out. hackster.io/amerch92/lora-weather-buoy-9e739b

7. SELF-POSITIONING RACE BUOY Competitive sailing can be so unfair with those antiquated, loosely anchored race markers. This one powers itself into precise position automatically. makeprojects.com/project/self-positioning-race-buoy

8. ELECTRIC KAYAK Invoke the power of a quarter of a horse with this 12V motor upgrade for your plastic vessel. create.arduino.cc/projecthub/Netcamprojects/an-electric-kayak-4d3a50

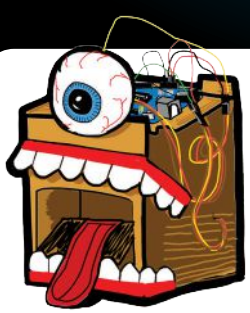
Adobe Stock - Simpline

CREATE YOUR OWN **MAGIC TRICKS** AND BECOME A

TRUE MAKER MAGICIAN

Make: Books — Mind Expanding

Order your copy at makershed.com/make-magic or amazon.com



THE CHOMPER BOT — a collaboration between Maker Shed and Mario the Maker Magician, this kit will get you MAKING MAGIC in no time!

Create a bot from scratch that chomps down and magically takes a bite out of a selected card. The perfect add-on to *The Maker Magician's Handbook*. make.co/mario

Maker Campus™

Connect, Learn, and Make Together

A place for project demos, classes, virtual workshop tours, and more.

makercampus.com