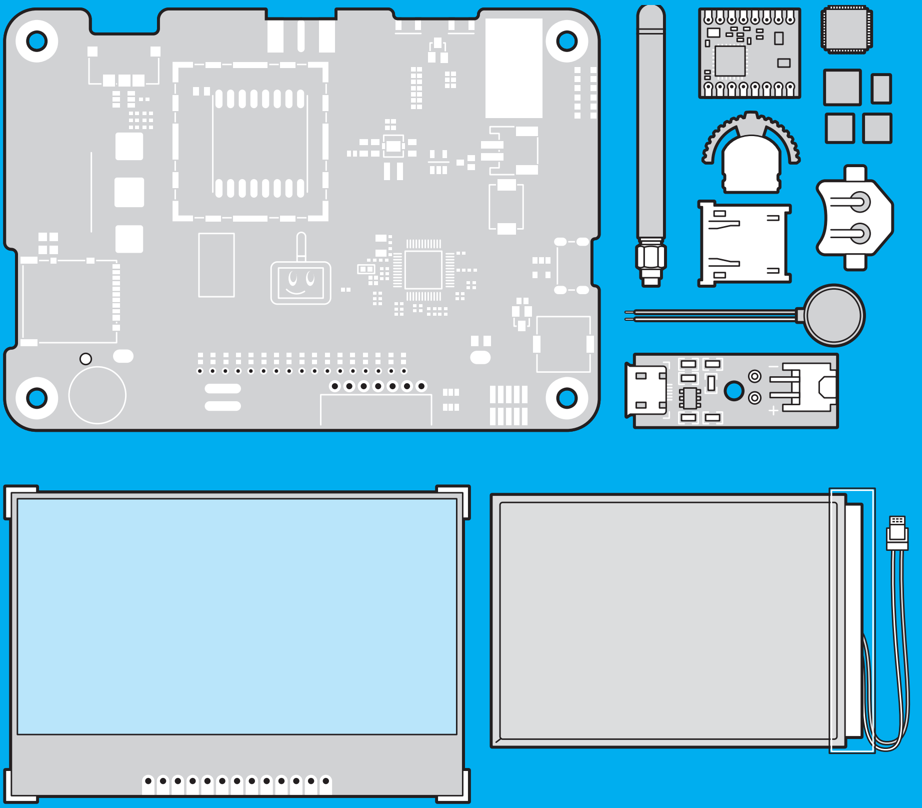
SPECTRUM COLLECTIONS VOL 1

Hands On

Experimenting & Exploring With Small Projects



Clocks | Replicas | Machine Learning | Radio | Science

IEEE

IEEE Spectrum

INTRODUCTION

For many of us, the joy of making things is why we got into fields like engineering in the first place. You take inert bits and pieces, combine them in the right way, and voilà! Something new comes to life.

But as careers progress, the chance to work on bigger and more sophisticated projects often requires either focusing on one small part of the whole, or dealing with higher and higher levels of design. Typically, you move further and further away from the workbench.

IEEE Spectrum's Hands On section is intended to put the whiff of solder back in your nostrils. Generally requiring no more than US $300 worth of parts—and often much less than that—they can usually be built in a weekend or two without recourse to specialized equipment.

However, Hands On articles differ from others intended for makers and hobbyists in that *Spectrum* focuses much more on the *why* than the details of the *how* of a project. We assume you know how to, say, read a resistor value—or how to Google it if you don't. We're more interested in the value a project can bring over and above the actual thing itself. What can a kit teach us about the history of technology, or a homebrew design about adapting technology in surprising ways?

We think there's deep value in experimenting and exploring with small projects, and this is a collection of some of our most popular examples from recent years, written by both experts and *Spectrum* staff. Even if you don't actually build any, we hope they spark some ideas, either for weekend projects of your own, or perhaps even for that big, sophisticated project at work.

On to the projects themselves. We don't know why, but engineers *really* love a funky timepiece, so we've included two—a kit that turns an obsolete oscilloscope cathode ray tube into a really neat clock, and a homebrew design that combines vintage space-age NASA surplus parts with a modern GPS receiver to show the hour, minute, and second via the stately sweep of old school needles.

We stay in the past for the next pair of articles. The first is a kit that lets you recreate the original PC, the Altair 8800. Building this machine disabused me of the notion that the Altair was little more than a toy with blinking lights, which is often the impression you get from commentators who have only seen pictures in history books. It's followed by another kit, this one replicating the scientific calculator that helped put inventor Clive Sinclair on the map. As with the Altair 8800, just reading the calculator's specs would make you think it's a barely functional kludge of a device, but building and operating it made me appreciate both the cleverness of the design and how it fit in with how engineers worked in the 1970s.

Next we jump to the present with two projects that explore the rapidly developing trend of employing AI outside the datacenter on relatively low-powered devices. We use Nvidia's Raspberry Pi-like Jetson Nano to make an automated tracking camera, and the even more resource-constrained Arduino-compatible SAMD51 processor for some handheld voice recognition.

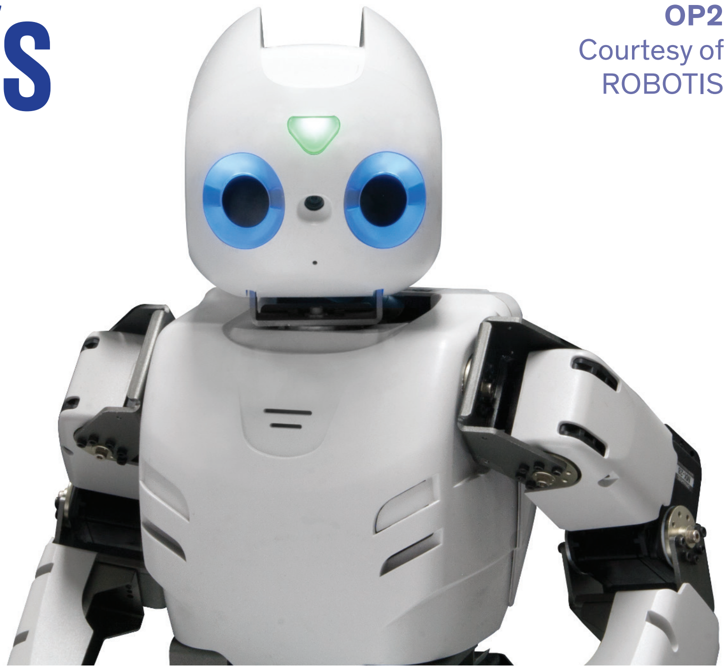
Another perennially popular subject for engineers is amateur radio. Though ham radio can sometimes feel a little stale, these two projects show how there's room for active digital experimentation, featuring a new modem for long distance data links and an Arduino shield that makes integrating a UHF/VHF transceiver with other electronics easy.

Finally, the last pair of articles takes us into the realm of citizen science. Electronics can let us see the world beyond the limitations of our senses, and these two Hands On entries take us from the scale of the galactic to the miniature, by building a radio telescope and digital microscope (from Lego!) respectively.

Working on the Hands On section is honestly one of the most fun things I've gotten to do as a technology editor in my career, and if you know of a project that might fit into *Spectrum*, drop me a line!

—**Stephen Cass**, senior editor
cass.s@ieee.org

The World's Best ROBOTS GUIDE Is Here!

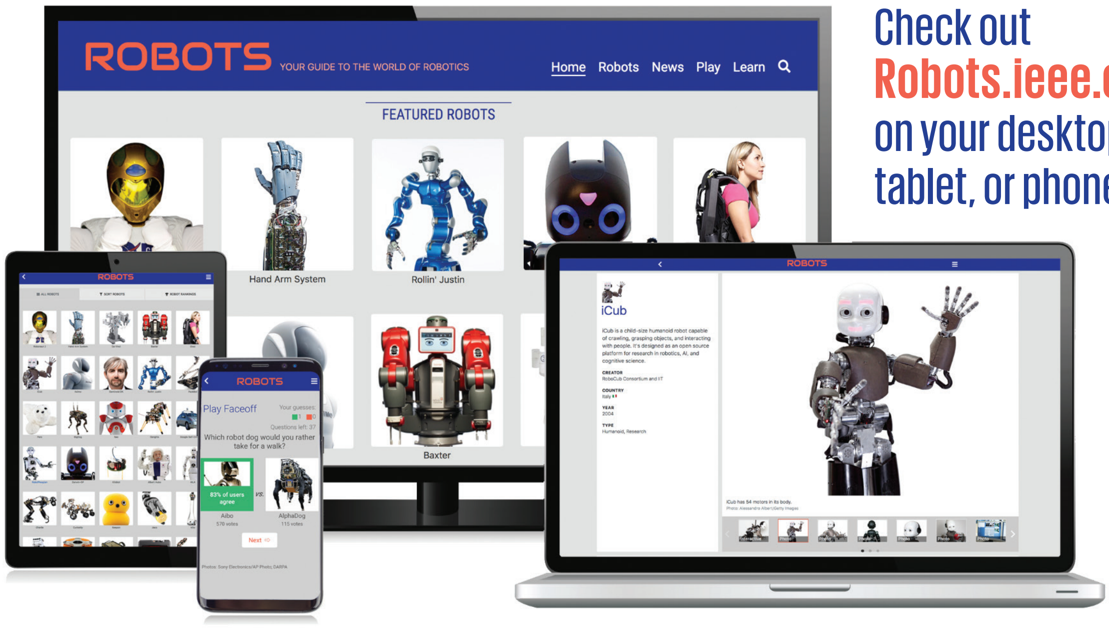


OP2
Courtesy of
ROBOTIS

ROBOTS.IEEE.ORG

IEEE Spectrum's **new ROBOTS site features more than 200 robots from around the world.**

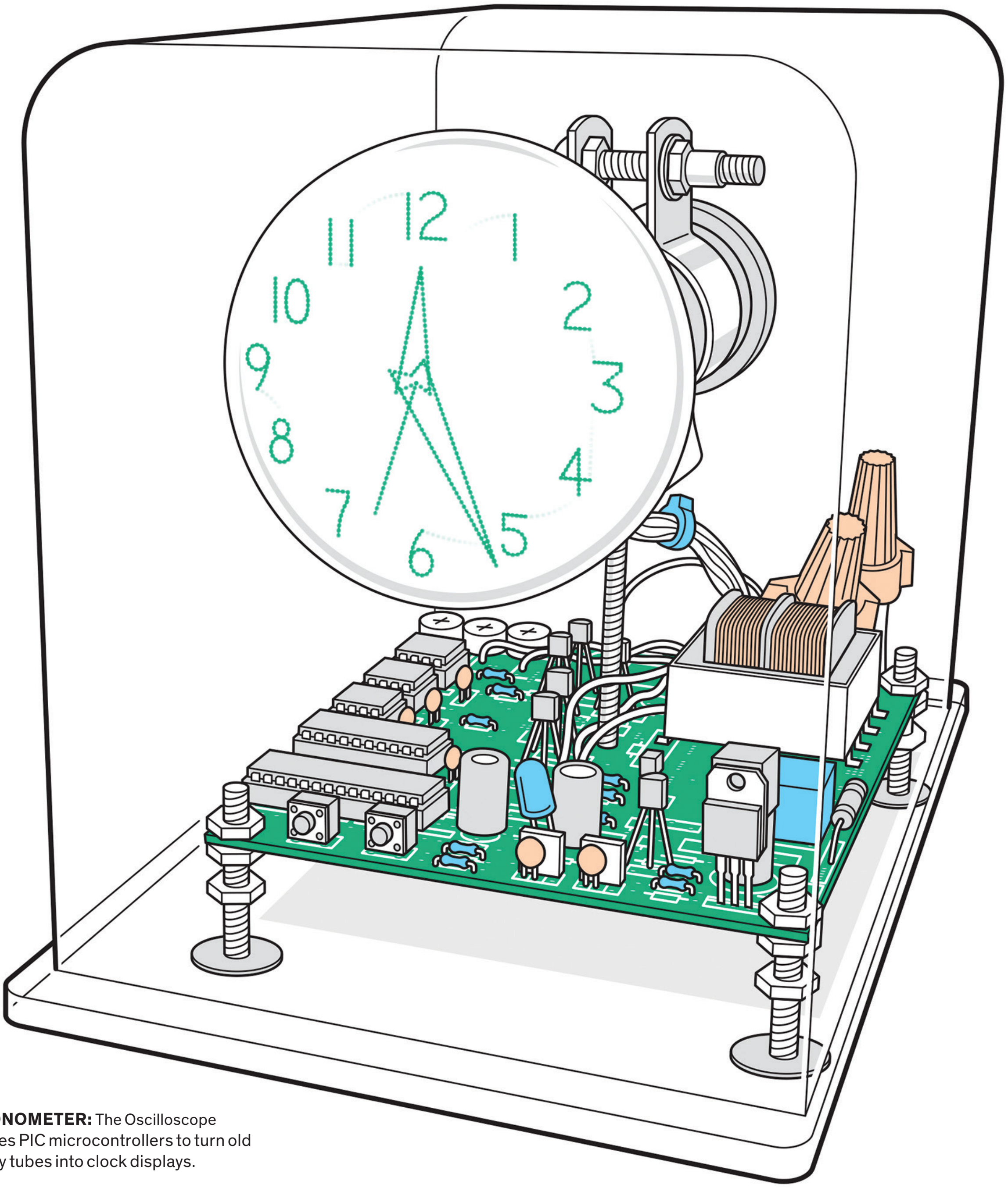
- Spin, swipe and tap to make robots move.
- Read up-to-date robotics news.
- Rate robots and check their ranking.
- View photography, videos and technical specs.
- Play *Faceoff*, an interactive question game.



Check out **Robots.ieee.org** on your desktop, tablet, or phone now!

CONTENTS

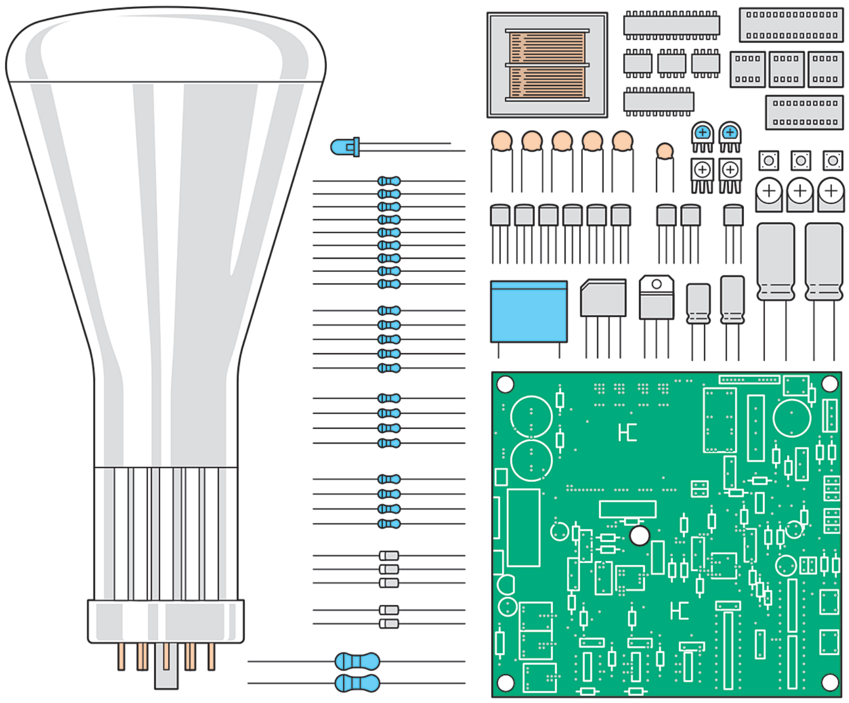
Hands On



CRT CHRONOMETER: The Oscilloscope Clock Kit uses PIC microcontrollers to turn old cathode-ray tubes into clock displays.

HANDS ON BY STEPHEN CASS

ILLUSTRATIONS BY James Provost



CLOCK COMPONENTS (clockwise from left)

1. The kit does not come with a CRT, but you can purchase one from sites like eBay. The kit will work with many different CRTs: You will need a lower voltage one designed for use in oscilloscopes. I built the kit using a DG7-6. **2.** A transformer and rectifier create the roughly 300 volts DC that the CRT requires. The clock can work on both 110- and 220-V power supplies selected via a jumper and can autodetect the frequency of utility AC power to use as a time base. **3.** The clock is driven by PIC microcontrollers and all the components are through-hole. **4.** The printed circuit board has space for optional extras, such as a battery backup, and Wi-Fi and GPS modules for setting the time.

OSCILLOSCOPE CLOCK

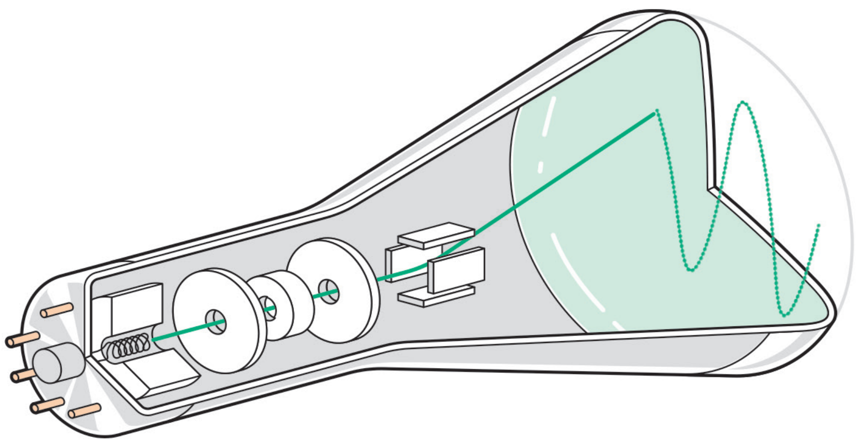
BUILD A VECTOR-GRAPHICS DISPLAY USING AN OLD CATHODE-RAY TUBE

ONCE UPON A TIME, THERE was a type of particle accelerator so popular that it was mass-produced by the million. Engineers and scientists at their benches, and folks at home in their living rooms, would carefully arrange themselves to watch the dancing glow of a beam of subatomic particles smashing into a phosphorescent screen. This attention-hogging accelerator was, of course, the cathode-ray tube (CRT), which reigned supreme as *the* electronic display technology for decades, before being unceremoniously relegated to the figurative and literal trash heap of history by flat-screen technologies.

But there are still CRTs to be found, and you can put some of them to great use with Howard Constantine's US $100 Oscilloscope Clock Kit. The kit works with many CRTs that were designed to be used in oscilloscope-type displays and operated with relatively low voltages—in the range of hundreds, rather than thousands, of volts.

As CRTs are becoming as unfamiliar to modern engineers as amplifier tubes did to the transistor generation, a quick recap of a few salient points is likely in order here. Oscilloscope-type CRTs are different from those found in televisions and most computer monitors. TV-type CRTs use magnetic fields generated by coils located outside the vacuum tube to deflect an electron beam, which is scanned line by line across the screen to build up what's called a raster image. Oscilloscope-type CRTs use two pairs of horizontally and vertically oriented plates located inside the tube to electrostatically deflect the beam: This approach was handy for oscilloscopes because an analog input voltage can control the vertical position of the beam directly (albeit filtered through some signal-conditioning circuitry), while an internal timing circuit controls the horizontal deflection, letting engineers see time-varying signals.

ILLUSTRATIONS BY James Provost



SUBATOMIC BEAM: In a CRT, electrons are boiled off the cathode and accelerated and focused before being steered by electrodes.

The beam's horizontal deflection can also be controlled with a second input voltage (called X-Y, or vector, mode). This made oscilloscope CRTs appealing to early computer-graphics pioneers, who pressed them into service as displays for things like radar defense networks. Some seminal computer games were made using vector displays, including arguably the first-ever video game, the 1958 *Tennis for Two*, and the massive 1979 arcade hit *Asteroids*. But vector displays struggled when it came to, say, showing bitmaps or even simply a large area filled with a solid color, and so eventually lost out to raster displays.

But CRT vector displays have a distinct look that's hard to replicate—not least when the screen is round, which was the easiest shape to make back in the day. (I do find it a little ironic that after decades of engineers striving to create the most perfectly flat and rectangular displays possible, smartphone makers have begun rhapsodizing about offering partially curved screens with rounded corners.)

The Oscilloscope Clock Kit allows you to recapture that look. The kit itself comprises the printed circuit board and all the accompanying components, including two PIC microcontrollers—one controls the clock and generates the graphics and text, while the other is dedicated to periodically shifting the clock around the screen to avoid phosphor burn-in. Normally you have to supply your own enclosure and CRT (eBay usually has listings), but as Constantine has a limited stock he uses to make fully assembled clocks for sale, he kindly let me buy a 7-centimeter-wide DG7-6 from him for $75 and one of his acrylic enclosures for $40.

Getting a clear enclosure was important to me because I wanted to be able to show off the tube for maximum effect, while also keeping fingers safely away from the electronics. This is important because even though the CRT is considered "low voltage," that still means 300 volts in some parts of the circuitry. Perhaps I was particularly skittish on this topic because of childhood memories: When I demonstrated a burgeoning propensity for taking things apart with a screwdriver, my father, a broadcast engineer, headed off any designs I might have had on our TV set with lurid true tales of people being zapped by the charge stored in a television's high-voltage capacitors even after the set had been unplugged.

Fortunately for my nerves, the clock kit's smaller capacitors pose much less of a hazard. Nonetheless, now that even 5 V is increasingly shunned as a circuit-melting torrent of electricity, I recommend builders work with a little more caution than they are used to, especially as checking stages of the build do require probing it with a multimeter when the board is plugged in.

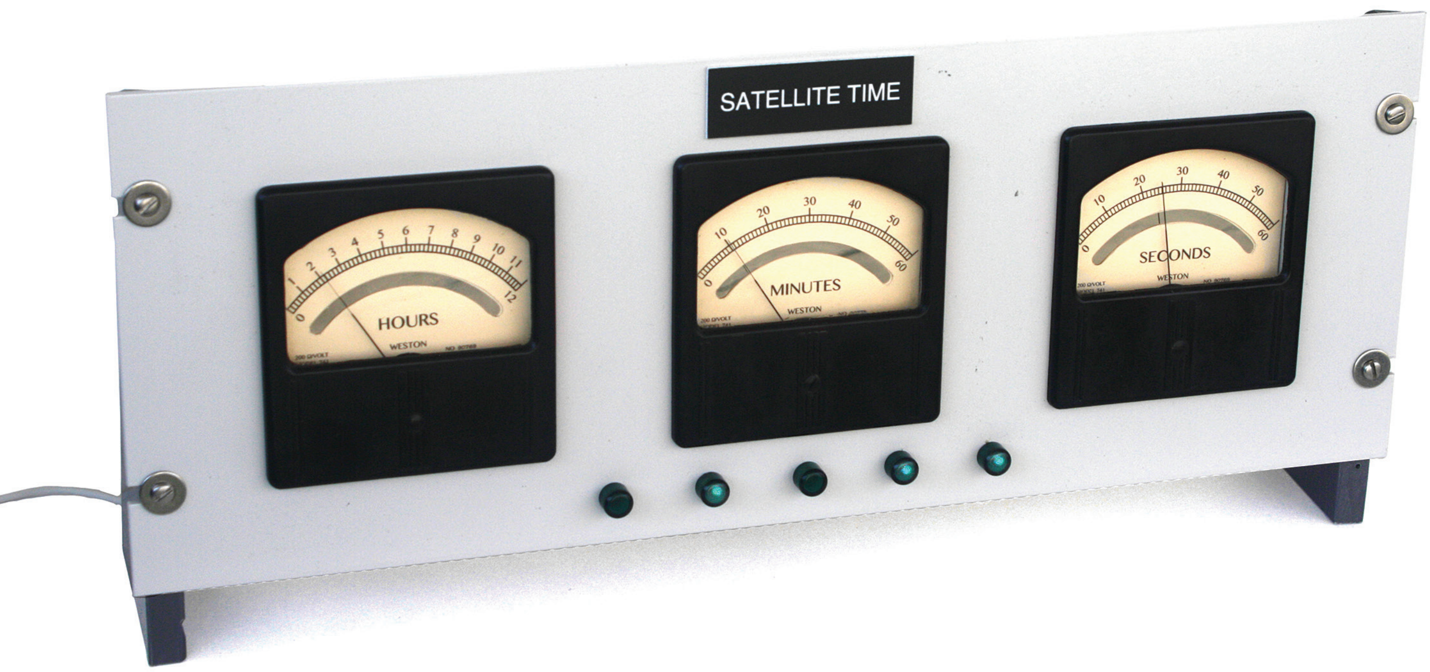
Soldering the through-hole components was straightforward, although because the tallest component—a transformer—is one of the first required, it's likely you'll need to use some kind of circuit-board holder rather than trying to lay the board flat when working. The biggest obstacle came when it was time to wire up my DG7-6 CRT. The kit provides 10 leads for operating a CRT—two that supply a few volts of alternating power to the heater filament to raise the cathode temperature enough for thermoelectric emission to come into play, one that has a constant negative voltage of about 295 V to provide the cathode's supply of electrons, three that connect to a train of accelerating and focusing electrodes in the neck of the CRT, and four that connect to the horizontal and vertical deflecting plate pairs. But my DG7-6 only had nine pins! A check of the DG7-6's data sheet (which I found on Frank Philipse's wonderful archive) showed that the cathode and one side of the heater filament shared a pin. A quick email to Constantine revealed the solution was a quick fix: All I had to do was jumper the cathode connection to one of the filament leads. After that, the instructions stepped me through the calibration steps required to produce a sharp bright test dot in the center of the screen rather than a fuzzy dim elliptical blob off to the side.

When building the kit, you can incorporate one of two optional $40 add-on circuits that eliminate the need to set the clock manually—a Wi-Fi module and a GPS module. Without one of those, the clock automatically detects whether it is plugged into a 50- or 60-hertz wall socket and uses that frequency as a reference time base. Setting the clock manually is simply a matter of pushing a "fast set" and "slow set" button until the clock shows the correct time.

The end result is an eye-catching timepiece that restores a CRT to its rightful place: the center of attention. ■

POST YOUR COMMENTS AT
spectrum.ieee.org/crtclock-jan2020

ILLUSTRATIONS BY James Provost



A RETRO TIMEKEEPER THAT NEVER WAS
APOLLO-ERA METERS AND GPS MERGE PAST AND PRESENT

TWO YEARS AGO, ON THE 48TH ANNIVERSARY OF THE APOLLO 11 LANDING, A CLOTH BAG THAT Neil Armstrong used to return the first lunar samples to Earth was sold at a Sotheby's auction for US $1.8 million. The seller had purchased it online two years earlier for a mere $995—a fantastically good deal for what turned out to be a precious artifact of the Apollo era. ● While I wasn't nearly so fortunate, I, too, got a good deal online for some hardware that probably contributed in some way to the Apollo program, though I don't know how exactly. I obtained three vintage analog panel voltmeters for $15 each from an eBay seller who had bought them from NASA's Marshall Space Flight Center, in Huntsville, Ala. ● I could tell from their art deco–inspired faces that the three Weston voltage meters were old when I first saw them online, but I didn't know how old. To my delight, I discovered manufacturing dates written on the back of the faceplates. These meters, it turns out, were made between May and December of 1955—and presumably shipped to Huntsville shortly afterward. ● At the time, NASA did not yet exist. Huntsville was, however, home to an Army installation, known as Redstone Arsenal, where missiles were being developed. In 1955, Wernher von Braun and numerous other German rocket scientists were hard at work there building rockets as part of the United States' ballistic missile program. This team would build the first U.S. satellite launcher, and later, after the site had become the Marshall Space Flight Center, von Braun and others in Huntsville helped to develop the giant Saturn boosters, which eventually sent the Apollo astronauts to the moon. ● Having scored three classy panel meters of some vague historical significance (I can

DAVID SCHNEIDER (5)

imagine von Braun having peered at their twitching needles), I wanted to do something fun with them. Inspired by a project I had seen on Hackaday, I decided to make a clock that would indicate the hours, minutes, and seconds by deflecting the needle on the analog voltmeters. But I would go a step further than the Hackaday project and combine the modern space age with the old. My clock would be synchronized with the atomic clocks carried on GPS satellites, while still looking like something that would be at home on a rack of instrumentation at some NASA facility during the Apollo program.

I had in my junk box a now-outdated GPS module, so I hooked that to an Arduino Nano, which I programmed to extract the time from the GPS signal. Then I opened up the meters and replaced the current-limiting resistors inside with ones of my own choosing so as to make the full-scale reading on each meter (originally 10 and 50 volts), to be somewhat less than 5 V. That allowed me to control the needle position on each meter using pulse-width modulation, which is easily output using certain dedicated pins on the Arduino.

The biggest challenge was to make it look like a period piece of equipment. For that, I first obtained a standard rack-mount "blanking" panel, one painted in a putty color, which contrasts nicely with the black Bakelite housings of the meters. I also purchased five vintage panel-lamp housings, into which I inserted LEDs for use as indicator lights.

Next came the trickiest part: changing the faces of the meters so that they could indicate time. The first step in that process was to remove the faceplate from one meter and scan it. Using that as a template, I designed new faceplates with Adobe Illustrator, ones that show hours, minutes, and seconds yet preserve the style of the original faces. I printed the new faces on ivory-colored stock cut from a manila folder, which helped to give them a vintage appearance.

A fiddly bit was cutting out arc-shaped openings for the meters' mirrors. Younger readers might not be aware that better analog meters, especially those used in volt-ohm meters ("VOMs," the predecessor to today's digital multimeters), included mirrors behind the faceplate, which helped you to judge whether you were viewing the needle square on. Without that, you could easily misread the indicated value because of parallax. My Marshall Space Flight Center meters included such mirrors, and I didn't want to cover them with my new faces. An X-Acto knife worked passably well for this task, although if you look hard, you can see the flaws.

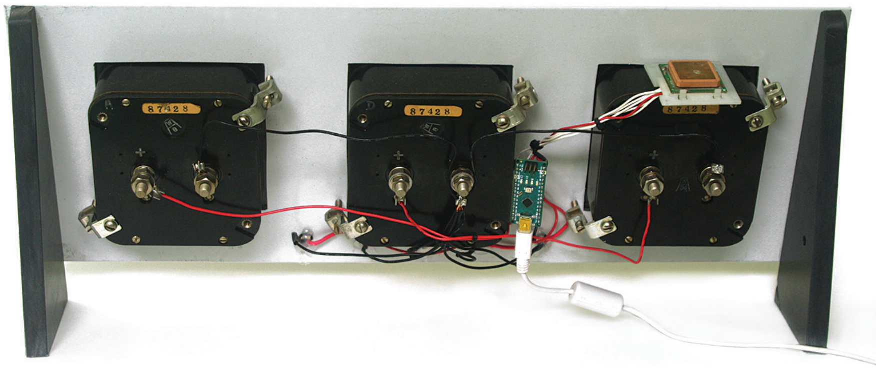
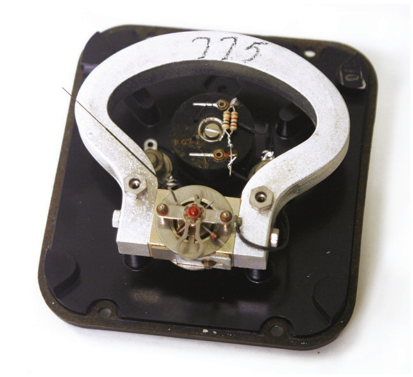
The final flourish was to affix a black plastic nameplate with the words "Satellite Time" in white lettering. That cost just a few dollars to have made and helps my clock look the part.

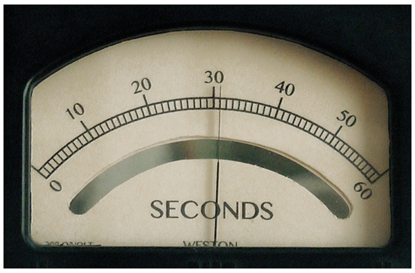
The three meters now show GPS time in my local time zone. The five panel lights indicate the number of GPS satellites in the sky. In theory, there can be as many as 16 overhead at any one time, so I display the number in binary using the five lights arrayed across the bottom of the clock.

Because my living room doesn't include a rack for my rack-mount panel, I cut a couple of supports out of half-inch (1.27-centimeter) thick gray PVC, allowing the clock to stand upright on any horizontal surface. Power is supplied to the Arduino through a standard wall wart.

Despite its unconventional appearance, this panel-meter-based clock is easy enough to read. And with its ear to the GPS constellation, it always tells the correct time. More important, I get a chuckle looking at it, knowing that it's built from components that probably contributed in some small way to putting astronauts on the moon a half century ago.

—DAVID SCHNEIDER



THE SWEEP OF TIME: I mounted three voltmeters on a rack panel [top]. A GPS module [affixed to the rightmost voltmeter] receives time transmissions and sends them to an Arduino. The Arduino converts the time to electrical signals representing hours, minutes, and seconds. I added resistors to the voltmeters so that they operate between 0 and a little less than 5 volts [second from top]. And I printed new faceplates [second from bottom], which I inserted into the meters [bottom].

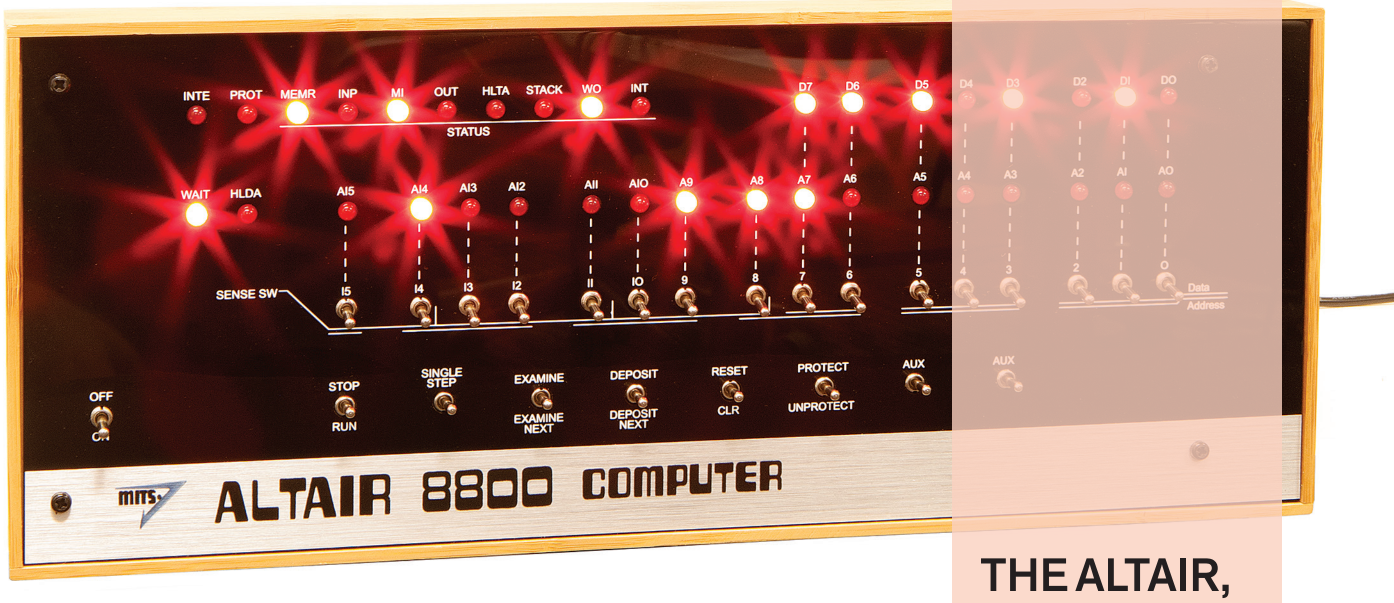
DAVID SCHNEIDER (5)

POST YOUR COMMENTS at https://spectrum.ieee.org/clock0719

RESOURCES

IEEE 696-1983

THE ALTAIR'S S-100 INTERNAL BUS DESIGN BECAME AN IEEE STANDARD, RETIRED IN 1994



THE ALTAIR, REINCARNATED

AN ARDUINO-BASED KIT TAKES YOU BACK TO 1974

RESOURCES_HANDS ON

The MITS Altair 8800 was the first commercially successful personal computer. Created by Ed Roberts in 1974, it was purchased by the thousands via mail order, proving there was a huge demand for computers outside universities and large corporations. Its influence was immense: For example, after seeing the Altair featured on the cover of the January 1975 issue of *Popular Electronics*, Bill Gates and Paul Allen founded Microsoft (then Micro-Soft) in order to write a Basic interpreter for the new machine. ● The Altair sold for US $439 in kit form. Original machines are now collectors' items that trade for thousands of dollars. Fortunately, there are some cheaper alternatives for people who want to get a direct understanding of the Altair computing experience. Modern kits that replicate the Altair hardware as faithfully as possible are available, as are purely virtual online simulators. Falling somewhere between a replica and a simulation is the $149 Altairduino kit from Chris Davis. The Altairduino duplicates the front panel of the Altair in all its LED- and switch-festooned glory while emulating the internal hardware (including some once fantastically expensive peripherals), using an Arduino Due. ● The Altairduino is derived from David Hansel's work on cloning the Altair with the Arduino Due and Arduino Mega 2560. If you want to build one of Hansel's designs from scratch, you can do so by following his free instructions on hackster.io. The advantage of Davis's kit is that it provides all the components, including a nice bamboo case and plastic front panel, along with a custom printed circuit board (PCB) that greatly simplifies construction. ● The original Altair's relatively large size means that most of the components are fairly well spaced out on the PCB. Even a beginner ▸

RANDI KLETT

could do much of the soldering, although a little bit more experience is required for trickier areas, such as the headers that connect to the Due. The fiddliest bit is adding the LED indicator lights. These are mounted on spacers, and it's best to put the front panel in place to ensure alignment, which can put you in one of those situations where you really wish you had an extra pair of hands to hold the panel, LED, and PCB tightly together while you solder. The online instructions are detailed and well illustrated, but I would recommend skipping forward and making sure you solder all the resistors in the kit before proceeding to add other, taller, components.

The Altairduino improves on the original Altair in two important respects. First, it offers modern interface options. You can connect an old-school terminal using an optional DB-9 connector (which I will stipulate should properly be called a DE-9 connector, so no need to send me letters this time!), but you can also use a soft terminal running on a computer via a USB connection, or even Bluetooth.

You do the initial configuration of the Altairduino via USB. The instructions are written for Microsoft Windows, so I had to do a little poking around the forums on the Altairduino site to figure out how to get my Mac to talk to the USB interface. Setting the baud rate to 115200 when launching the "screen" terminal command did the trick, and once I set the Bluetooth connection up as the power-on default, it was all smooth sailing.

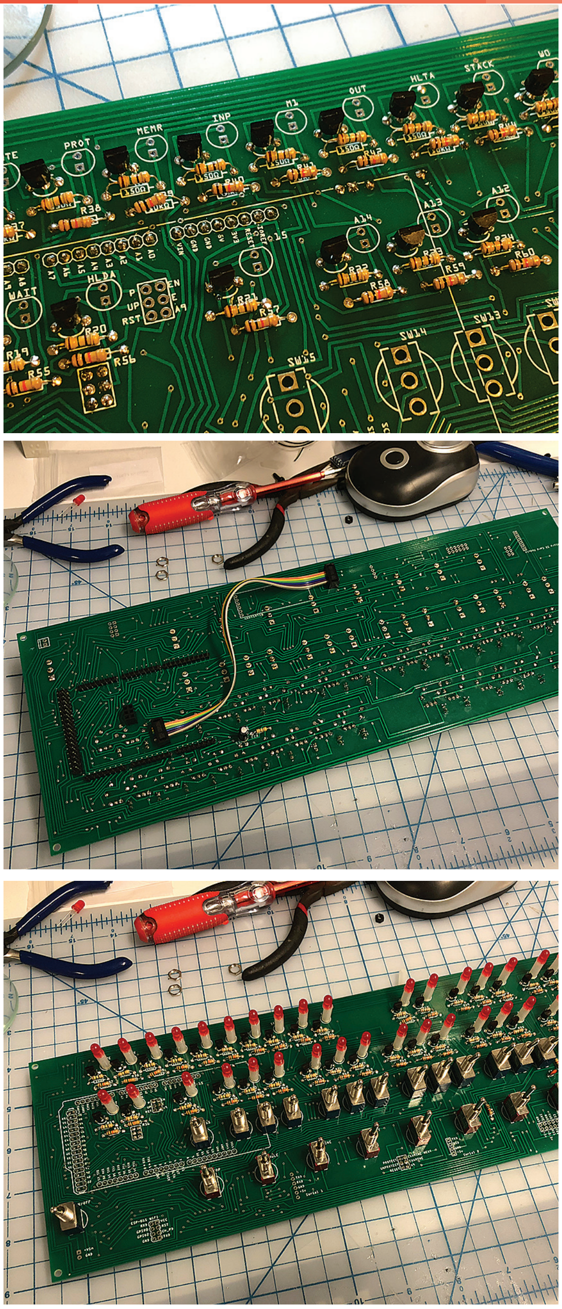
The second big improvement is that the Altairduino comes loaded with a lot of software. You can call up some programs purely by flipping various front panel switches, such as *Kill the Bit*, a game that hacked the Altair's memory-address indicator lights to act as a 1-dimensional display. Other programs are called up with a combination of switch throws and terminal commands. You can quickly fire up Microsoft's very first 4-kilobyte Basic (which gives you the option, on startup, to disable its sine, random number, and square root functions to save a little memory), or its more advanced 16-KB Basic. The latter has a number of programs you can load from the Altairduino's memory, including early computer game classics such as *Lunar Lander*, *Star Trek*, and *Hunt the Wumpus*.

You can put additional Altair software on a microSD card, which you plug into a reader that's soldered to the PCB (Davis conveniently offers a one-stop bundle on his website). Once the kit is assembled, you can't access the reader to swap out the card without unscrewing the case, but since the universe of Altair 8800 software isn't growing that rapidly, I'll manage. (That said, I did just see someone announce they had gotten a Forth compiler running on the Altairduino that I'd like to try.)

The card reader emulates the 88-HDSK hard disk that was available in the business version of the Altair, which sold in the late 1970s for $11,450 to $15,950. A number of disk images are available in the software bundle, including one with the CP/M operating system. The CP/M software also comes with a bunch of software, including parts 1, 2, and 3 of *Zork*, a pioneering text adventure game.

The Altairduino is a lovely kit that's an enormous amount of fun—it is surprisingly satisfying to control a computer by flipping switches versus, say, mouse clicks. More seriously, simply looking at pictures of early personal computers, with their blinking rows of lights and bulky cases, can leave the impression they were little more than toys. But engaging with this incarnation quickly demonstrates that the Altair was a capable system, and it becomes much clearer why it was that *this* machine came to be so critical in establishing the value of personal computing.

—STEPHEN CASS

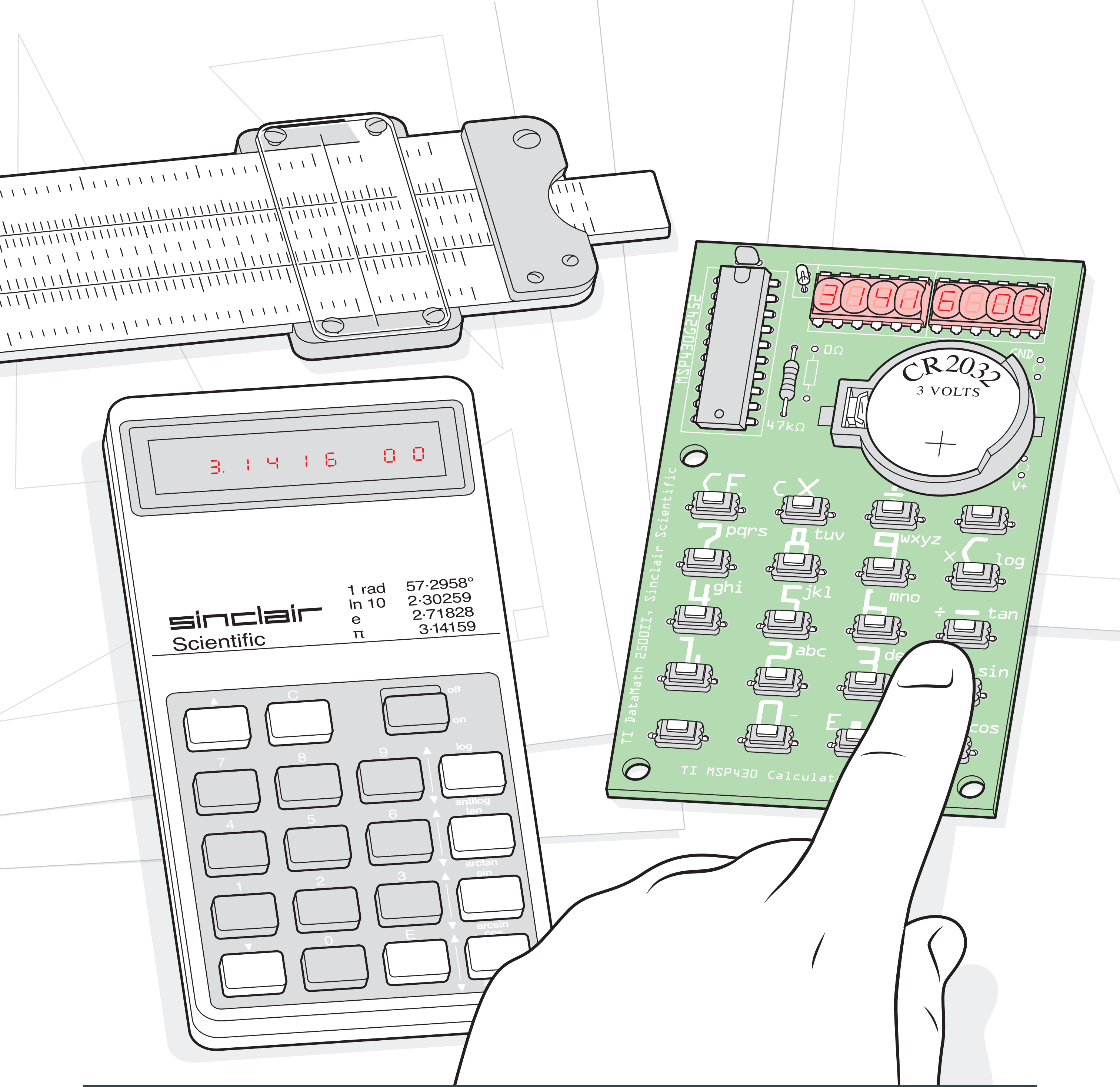


LIGHTS, SWITCHES, ACTION: A printed circuit board contains additional circuitry for driving LEDs [top], but most of the work is handed off to an Arduino Due plugged into a rear connector [middle]. The finished PCB [bottom] is mounted in the case.

STEPHEN CASS (3)

↗ POST YOUR COMMENTS at https://spectrum.ieee.org/altairduino0318

Hands On

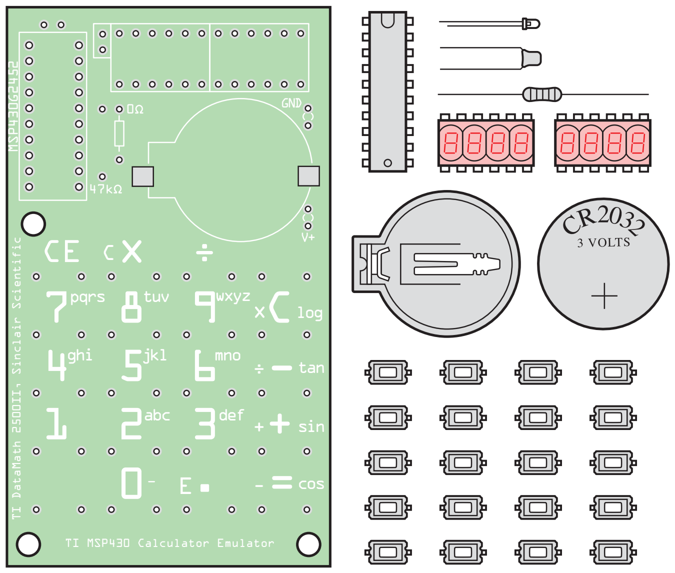


HANDS ON BY STEPHEN CASS

ILLUSTRATIONS BY James Provost

THE SINCLAIR SCIENTIFIC, REMADE

A CLASSIC CALCULATOR WAS REVERSE ENGINEERED FROM THE BARE METAL


MSP430G2452
0Ω
47kΩ
GND
V+
CE C X ÷
7 pqrs 8 tuv 9 wxyz x (log
4 ghi 5 jkl 6 mno ÷ − tan
1 2 abc 3 def + + sin
0 - E . - = cos
TI Datamath 2500II, Sinclair Scientific
TI MSP430 Calculator Emulator
CR2032
3 VOLTS

REFACTORED REPLICA: The original Sinclair Scientific was also sold in kit form. Chris Chung's version is smaller, uses fewer components, and can actually replicate *two* calculators based on the TMS080*x* family of CPUs: the TI Datamath, a simple arithmetical calculator, and the Sinclair Scientific. Hence, the printed silkscreen on the circuit board has layouts for both calculators.

Was the Sinclair Scientific calculator elegant? It certainly was a hit, gracing the cover of publications like *Popular Mechanics* after its release in 1974. Cleverly written firmware dragooned its limited processor, intended only for basic arithmetic, into performing way beyond specifications. This allowed Sinclair to sell a scientific calculator to countless folks who otherwise could not have afforded one. But it was also slow and sometimes inaccurate, provided barely enough mathematical functions to qualify as a scientific calculator, and was difficult for the uninitiated to use.

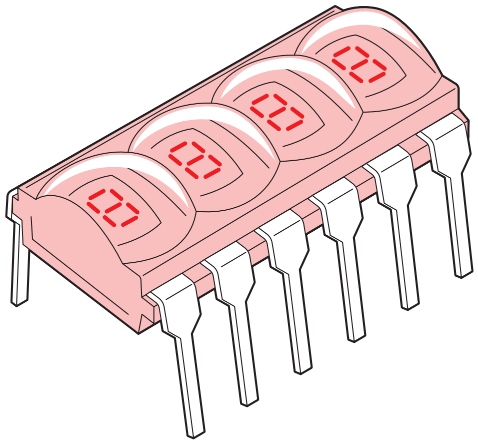
I'd vaguely known of this calculator because of its place as a milestone toward the birth of the British microcomputer industry and such beloved machines as the Sinclair ZX Spectrum. So I clicked when I came across Chris Chung's replica kit of the calculator on the Tindie marketplace. Then I read the description of how the original calculator worked—scientific notion only? no "equals" button?—and how the replica reproduced its behavior, by using an emulator running firmware that had been reverse engineered by *visually examining the metal of an original processor.* This I had to try.

Let's first get to the hardware. The kit is one of a number of Sinclair calculator replicas, but it wins points for simplicity: One chip and a credit-card-size printed circuit board are combined with a small handful of discrete components. Chung actually offers two kits: his original, developed in 2014 but put on Tindie in late 2019, displays numbers using two small QDSP-6064 bubble LED modules, which have the classic look of 1970s calculators but are long discontinued and hard to get. His 2020 update of the kit uses modern seven-segment LED displays. The difference in rarity is reflected in the price: The 2020 version costs US $39, while the original version costs $79. However, in a nice touch, the original version lets you have your cake and eat it with regard to the bubble LEDs: The through holes in the PCB are sized to create a friction fit. This means you don't have to solder in the modules, preserving them for some future project.

Both versions are functionally identical, based around a MSP430 mi-

DEPARTMENTS

ILLUSTRATIONS BY James Provost



THAT 70'S LOOK: Bubble LED modules were common in early calculators. The bubbles were lenses that magnified the small segments.

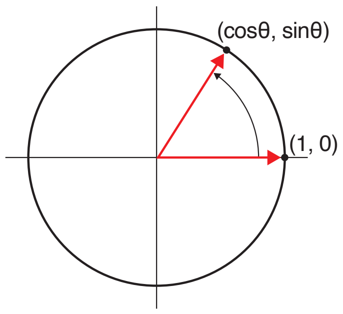
crocontroller. The MSP430 eliminates the need for most of the other components found in the Sinclair Scientific, and runs an emulator of the TMS080*x* family of chips. The TMS080*x* family was built by Texas Instruments (TI), and specific versions, like the TMS0805 used in the Sinclair Scientific, were differentiated by the contents of the chip's 3,520-bit ROM.

For many years, how Sinclair coaxed this chip into performing magic remained locked away in TMS0805 chip ROMs. That began to change in 2013, when Ken Shirriff got wind of the Sinclair Scientific after getting involved with the Visual 6502 team. This group likes to reverse engineer classic chips, for which the original design drawings are often lost. Sometimes this involves etching chip packages away with acid and carefully photographing the exposed silicon dies with a microscope to see the individual transistors. Shirriff was able to create a generic simulator of a TMS080*x* chip in JavaScript just by studying Texas Instruments' patent filings, but the specific code used in the Sinclair's TMS0805's ROM still eluded him, until Visual 6502 team member John McMaster took photographs of an exposed die in 2014.

Shirriff is no stranger to *IEEE Spectrum* due to his extensive work in the history of computing, so I emailed him to ask how he'd gone from a micrograph to working code. By looking at how the metal oxide gates were laid down in the ROM section of the chip, "I was able to extract the raw bits the same day," Shirriff wrote back. "Phil Mainwaring, Ed Spittles, and I spent another day figuring out how the raw bits corresponded to the code.…The code was 320 words of 11 bits, but the ROM was physically 55 rows and 64 columns.…Through a combination of examining the circuitry, analyzing patterns in the bits, and brute-force trying a bunch of combinations, we figured out the arrangement and could extract the code."

Once he had the code loaded into his simulator, Shirriff could tease out its workings. The algorithms used throughout "were essentially the simplest brute-force algorithms that could get an answer. But there were some interesting mathematical tricks they used to get more accuracy, not to mention programming tricks to get the code to fit," he explained. (If you want detailed explanations, Shirriff maintains an online version of his simulator that lets you step through the code, line by line.)

The Sinclair Scientific was able to reduce complexity by using reverse Polish notation, in which mathematical operators come after the numbers they are operating on—for instance, "5 + 4 =" becomes "5 4 +." The trigonometric functions used an iterated approximation technique that could take several seconds to produce results and were often accurate only to the first three significant figures. The calculator also used fixed scientific notation for everything—there was no way to enter a decimal point. So rather than entering "521.4," you'd enter "5214," which is displayed as "5.214"; then you'd press "E" and enter "2," making the number "5.214×10^2."
Only one number could be entered at a time.



DO THE TWIST: The Sinclair Scientific calculated trigonometric functions by repeatedly rotating an initial vector until the target angle was reached. It took several seconds to calculate most angles.

On paper this looks terrible, something you'd have used only if you couldn't afford, say, the HP-35, whose designers prided themselves on accuracy and functionality (although the HP-35 also used reverse Polish notation, it did so in a more sophisticated way).

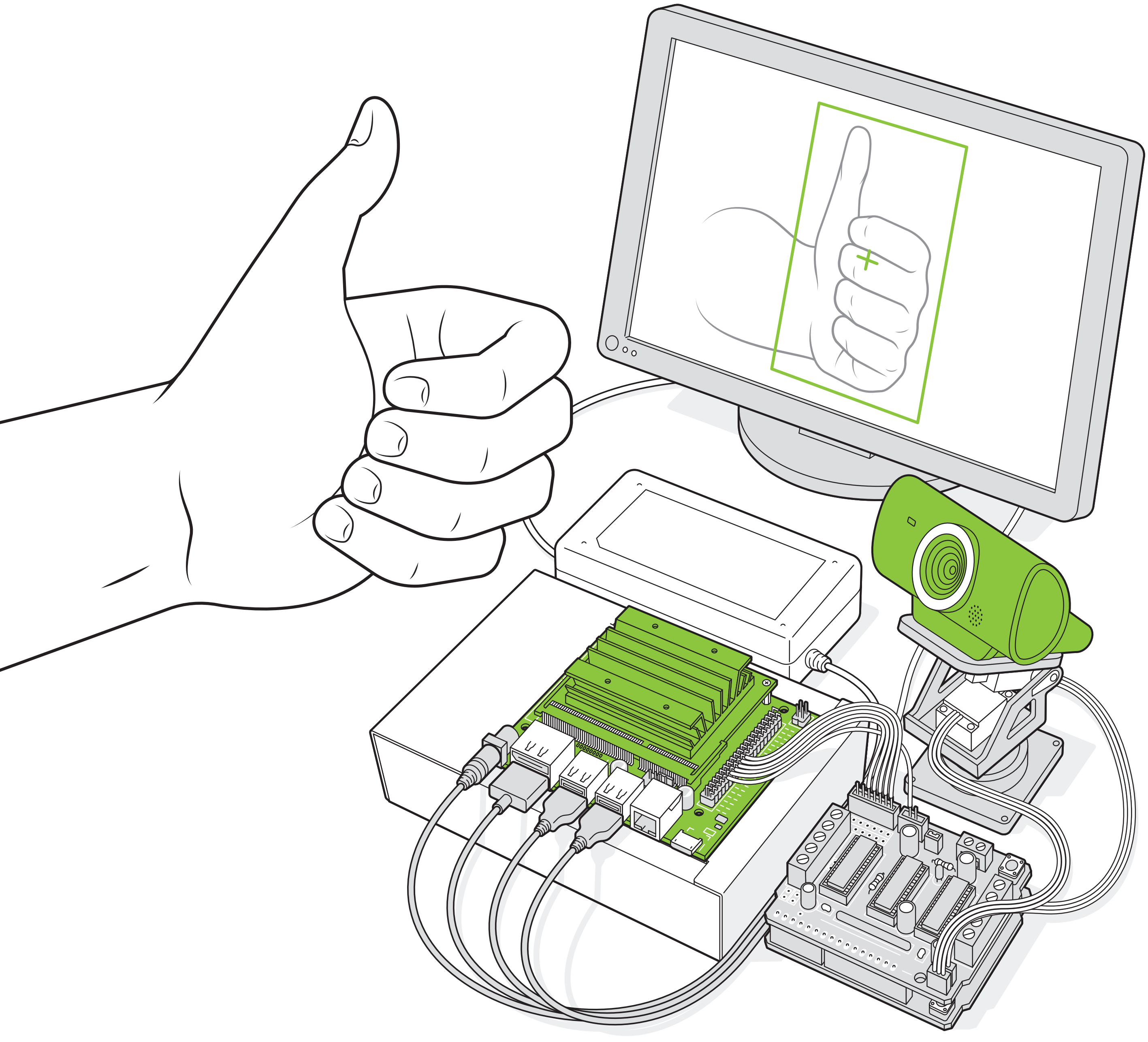
But Sinclair wasn't trying to compete against other calculators; he was competing against *slide rules.* I'd read that point in histories before, but I didn't really understand its meaning until I held this kit in my hand. By chance, I'd also recently purchased a vintage Pickett slide rule and learned how to do basic operations with it, thanks to the lessons available on the International Slide Rule Museum website. As I used the Sinclair Scientific, I was struck by how conceptually similar it was to using my slide rule. There, two- to three-figure accuracy is the norm, the rule's sliding "cursor" means you're passing just one number between scales, and you typically don't bother about magnitudes—you work with the most significant figures and make a mental estimate to insert the decimal point at the end, so that 52×2 and $5{,}200 \times 20$ are calculated in exactly the same way.

This realization is why I feel replica kits are so important—they are an easy way to provide the understanding that only comes from operating a device with your own hands. It was a great reminder that, as folks like Henry Petroski have pointed out, good design is not really something that exists as an abstract ideal but as something that exists only within a specific context. So, was the Sinclair Scientific elegant? For me, the answer is a resounding yes. **—STEPHEN CASS**

POST YOUR COMMENTS AT
spectrum.ieee.org/sinclair-jun2020

ILLUSTRATIONS BY James Provost

Hands On

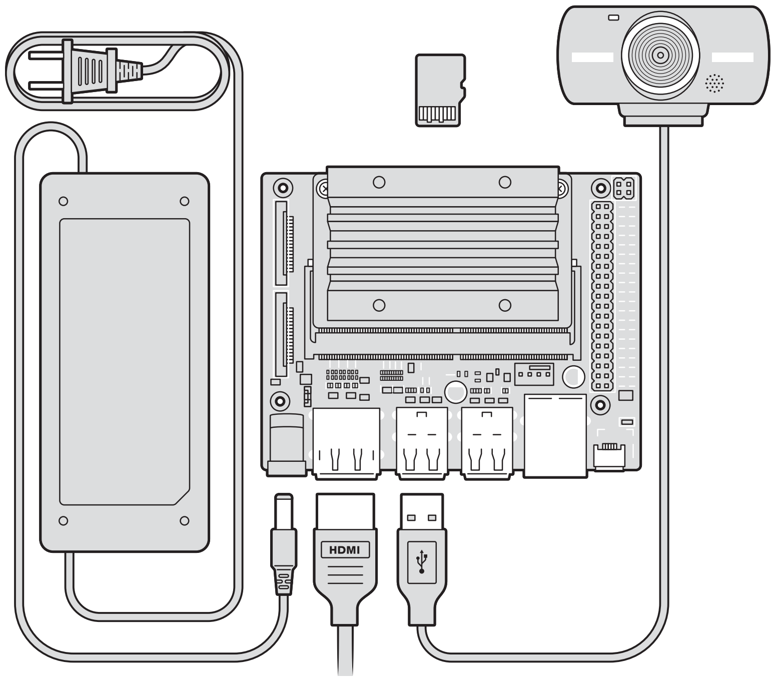


HANDS ON BY STEPHEN CASS

ILLUSTRATIONS BY James Provost

NVIDIA MAKES IT EASY TO EMBED AI

THE JETSON NANO PACKS A LOT OF MACHINE-LEARNING POWER INTO DIY PROJECTS



AI ENGINE: The Jetson Nano is the cheapest of a number of AI development kits made by Nvidia. Like many AI development boards it has hefty power requirements, so users are encouraged to buy an external power adapter that can supply at least 4 amperes. Lots of power in necessarily means lots of heat out, so the kit is dominated by a large heat sink which has screw holes for adding a fan if desired. The Nano has lots of input/output options for displays, USB peripherals, camera modules, and a Raspberry Pi–compatible 40-pin general purpose input/output header.

When opportunity knocks, open the door: No one has taken heed of the adage like Nvidia, which has transformed itself from a company focused on catering to the needs of video gamers to one at the heart of the artificial-intelligence revolution. In 2001, no one predicted that the same processor architecture developed to draw realistic explosions in 3D would be just the thing to power a renaissance in deep learning. But when Nvidia realized that academics were gobbling up its graphics cards, it responded, supporting researchers with the launch of the CUDA parallel computing software framework in 2006.

Since then, Nvidia has been a big player in the world of high-end embedded AI applications, where teams of highly trained (and paid) engineers have used its hardware for things like autonomous vehicles. Now the company claims to be making it easy for even hobbyists to use embedded machine learning, with its US $100 Jetson Nano dev kit, which was originally launched in early 2019 and rereleased this March with several upgrades. So, I set out to see just how easy it was: Could I, for example, quickly and cheaply make a camera that could recognize and track chosen objects?

Embedded machine learning is evolving rapidly. In April 2019, Hands On looked at Google's Coral Dev AI board which incorporates the company's Edge tensor processing unit (TPU), and in July 2019, *IEEE Spectrum* featured Adafruit's software library, which lets even a handheld game device do simple speech recognition. The Jetson Nano is closer to the Coral Dev board: With its 128 parallel processing cores, like the Coral, it's powerful enough to handle a real-time video feed, and both have Raspberry Pi–style 40-pin GPIO connectors for driving external hardware.

But the Coral Dev board is designed to provide the minimal amount of support

ILLUSTRATIONS BY James Provost

needed to prototype embedded applications. Its Edge TPU is on a removable module that can be plugged directly into target hardware as development progresses. The Coral carrier board that the processor module sits on has only one large USB port and is generally intended to be operated over a remote connection via a USB-C or Ethernet connector rather than being plugged in to a keyboard and screen directly.

The Jetson Nano has a similar module-and-carrier board approach, but its carrier board has a lot more going for stand-alone experimentation. It has four USB 3.0 ports for peripherals, HDMI and DisplayPort connectors, a Micro-USB port to supply power or allow remote operation, an Ethernet port, two ribbon connectors for attaching Raspberry Pi–compatible camera modules, and a barrel jack socket for providing the additional power needed for intensive computations. (I appreciated this last touch, because the Coral Dev board uses two USB-C ports side by side, one for power and the other for data, and I was always confusing the two.)

Nvidia's "Getting Started" instructions stepped me through downloading a customized version of the Ubuntu operating system designed for the Nano. This includes things like a Nano version of the established Raspberry Pi Rpi.GPIO library for accessing the GPIO header from Python scripts.

Then, as per the instructions, I began the free "Welcome to Getting Started With AI on Jetson Nano!" course. This includes an introductory primer on neural networks, followed by details on how to use the Nano in remote mode. In this mode, the Nano runs a local Web server that hosts a Jupyter notebook, the tool of choice for many researchers because it allows explanatory text and images to be intermingled with Python code.

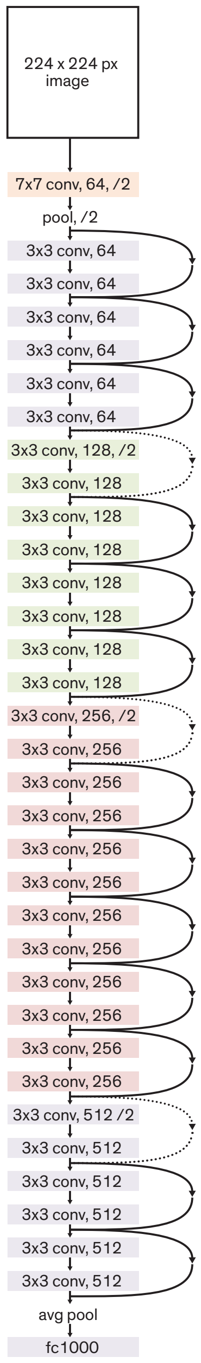
Unfortunately, I couldn't connect to the Nano's server: After some poking around online, it turns out that the Ubuntu version required for the "Welcome…" course is different from the version linked in the "Getting Started" instructions. There are actually a few places where Nvidia's generally excellent documentation, videos, and other tutorials have this kind of rough edge: Sometimes it's because things haven't been updated for the new board; other times it's simple mistakes, such as a sample script used for testing the output of the GPIO header that tries to import the original Rpi.GPIO library rather than the Jetson.GPIO library.

But soon I was up and running with the well-known ResNet-18 pretrained neural network, which can classify 1,000 objects. Through a provided notebook it is easy to modify the last layer of the network so that it recognizes just two things—say, thumbs-up and thumbs-down gestures—using images directly captured from the USB camera, and then retrain the network for gesture recognition.

Then I tried something a little more ambitious. Following along with Nvidia's suggested "Hello AI World" tutorial, I was soon running a 10-line Python script directly on the Nano's desktop that did real-time recognition with the ResNET-18 network. With a little bit of tweaking, I was able to extract the ID and center coordinates of every object recognized. A little more modification and the script toggled four GPIO pins to indicate if a particular type of object—say a bottle, or person—was above, below, left of, or right of center in the camera's view.

I hooked up an Arduino Uno with a motor shield to drive the two servos in a $19 pan/tilt head that I bought from Adafruit. I mounted my USB camera on the pan/tilt head and wired the Arduino to monitor the Nano's GPIO pins. When the Nano senses an off-center object and turns on the appropriate GPIO pin, the Arduino nudges the pan/tilt head in the direction that should bring the object toward the center of its field of view.

I was genuinely surprised how quickly I was able to go from opening the Nano's box to a working autonomous tracking camera (not counting the time I spent waiting for various supporting bits and pieces to arrive in the mail while isolated from my usual workbench at the *Spectrum* office!). And there are plenty of avenues for further expansion and experimentation. If you're looking to merge physical computing with AI, the Jetson Nano is a terrific place to start. **—STEPHEN CASS**



MULTILAYERED ANALYSIS: The software suite provided by Nvidia makes it easy to download many popular pretrained neural-network architectures that are suited for different tasks, such as identifying pedestrians or household objects. The ResNet-18 network pictured above can identify 1,000 such objects. You can make the network identify new, similar, objects by reconfiguring the final classification layer, which attaches labels to objects, and partially retraining the network.

POST YOUR COMMENTS AT
spectrum.ieee.org/jetson-jul2020

ILLUSTRATIONS BY James Provost

MAKING MACHINE LEARNING ARDUINO COMPATIBLE

A GAMING HANDHELD THAT RUNS NEURAL NETWORKS

I WANT TO FIND OUT WHAT HAPPENS WHEN WE BRING machine learning to cheap, robust devices that can have all kinds of sensors and work in all kinds of environments. And I want you to help. The kind of AI we can squeeze into a US $30 or $40 system won't beat anyone at Go, but it opens the door to applications we might never even imagine otherwise. ● Specifically, I want to bring machine learning to the Arduino ecosystem. This has become recently possible thanks to improvements in hardware and software. ● On the hardware side, Moore's Law may be running out of steam when it comes to cutting-edge processors, but the party's not over when it comes to microcontrollers. Microcontrollers based on 8-bit AVR processors dominated the Arduino ecosystem's early years=, for example, but in more recent years, embedded-chip makers have moved toward more powerful ARM-based chips. We can now put enough processing power into these cheap, robust devices to rival desktop PCs of the mid 1990s. ● On the software side, a big step has been the release of Google's TensorFlow Lite, a framework for running pretrained neural networks—also known as models—on so-called edge devices. Last April, *IEEE Spectrum*'s Hands On column looked at Google's Coral Dev Board, a single-board computer that's based on the Raspberry Pi form factor, designed to run TensorFlow Lite models. The Coral incorporates a dedicated tensor processing unit and is powerful enough to process a live video feed and recognize hundreds of objects. Unfortunately for my plan, it costs $150 and requires a hefty power supply, and its bulky heat sink and fan limit how it can be packaged.

But *fortunately* for my plan, Pete Warden and his team have done amazing work in bringing TensorFlow Lite to chips based on ARM's Cortex family of processors. This was great to discover, because at my open-source hardware company, Adafruit Industries, our current favorite processor is the 32-bit SAMD51, which incorporates a Cortex-M4 CPU. We've used the SAMD51 as the basis for many of our recent and up-

ADAFRUIT INDUSTRIES (3)

adafruit
PYGAMER
Test the TensorFlow lite voice model. Press and hold A button, say YES or NO, and see if the machine learned!
version 1.12.2
TensorFlowLite
SELECT
START
RESET
UCC: 2.4-5.5U Adjustable gain
OUT GND UCC

A BIT OF FUN: After she created an Arduino-compatible version of TensorFlow Lite, the author adapted a voice-recognition demo so that pressing a button and speaking into a microphone attached to a SAMD51-based PyGamer would play back different animations.

ADAFRUIT INDUSTRIES (3)

coming Arduino-compatible boards, including the PyGamer, a simple battery-powered gaming handheld. What if we could use it to literally put AI into people's hands?

Warden had created a speech-recognition model that can identify the words "yes" and "no" in an analog audio feed. I set about seeing if I could bring this to the PyGamer, and what I might do with a model that could recognize only two words. I wanted to create a project that would spark the imagination of makers and encourage them to start exploring machine learning on this kind of hardware.

I decided to make it as playful as possible. The more playful something is, the more forgivable its mistakes—there's a reason why Sony's artificial pet Aibo was given the form of a puppy, because real puppies are clumsy and sometimes run into walls, or don't follow instructions.

I recalled the original *Tron* movie, where the hero is stuck in cyberspace and picks up a sidekick of sorts, a single bit that can say only "yes" or "no," with an accompanying change of shape. The PyGamer has a 1.8-inch color display, with 192 kilobytes of RAM and 8 megabytes of flash file storage, enough to display snippets of video from *Tron* showing the bit's "yes" and "no" responses. The PyGamer's SAMD51 processor normally runs at 120 megahertz, which I overclocked to 200 MHz for a performance boost. I connected an electret microphone breakout board to one of the PyGamer's three JST ports.

Then I turned to the trickiest task: porting the TensorFlow Lite ARM code written by Warden and company into a library that any Arduino programmer can use (although not for every Arduino board! Even as a "lite" framework, the RAM requirements are far beyond the 2 kB of the Arduino Uno, for example).

I found the source code well written, so the biggest challenge became understanding how it handles incoming data. Data is not digested as a simple linear stream but in overlapping chunks. I also wanted to expose the code's capabilities in a way that would be familiar to Arduino programmers and wouldn't overwhelm them. I identified the functions most likely to be useful to programmers, so that data could be easily fed into a model from a sensor, such as a microphone, and the results outputted to the rest of the programmer's code for them to handle as they wish. I then created an Arduino library incorporating these functions, which you can find on Adafruit's Github repository.

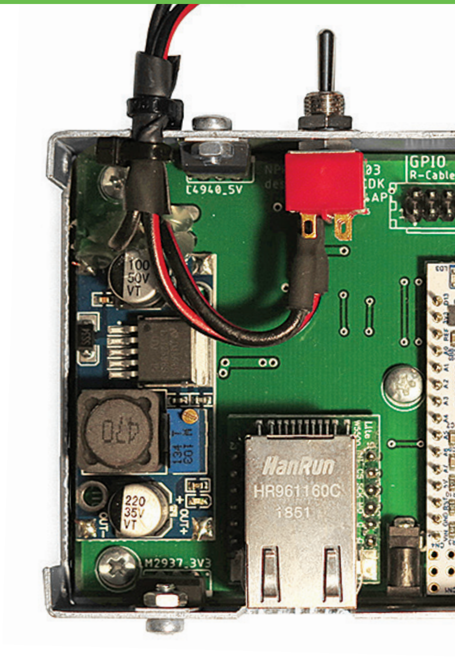
Putting it all together, I wrote a short program using the new library. Now when I press a button and speak into the PyGamer's attached microphone, the appropriate *Tron* clip is triggered if I say "yes" or "no," letting me walk around with my own animated sidekick.

Although this project is a limited (but fun) intro to machine learning, I hope it persuades more makers and engineers to combine AI with hardware explorations. Our next steps at Adafruit will be to make it easier to install different models and create new ones. With 200 kB of RAM, you could have a model capable of recognizing 10 to 20 words. But even more exciting than voice recognition is the prospect of using these cheap boards to gather data and run models built around very different kinds of signals. Can we use data from the PyGamer's onboard accelerometer to learn how to distinguish the user's movements in doing different tasks? Could we pool data and train a model to, say, recognize the sound of a failing servo or a switching power supply? What surprises could lie in store? The only way to find out is to try. **—LIMOR FRIED**

Editor's note: Limor Fried is a member of IEEE Spectrum*'s editorial board.*

POST YOUR COMMENTS at https://spectrum.ieee.org/arduino0819

1



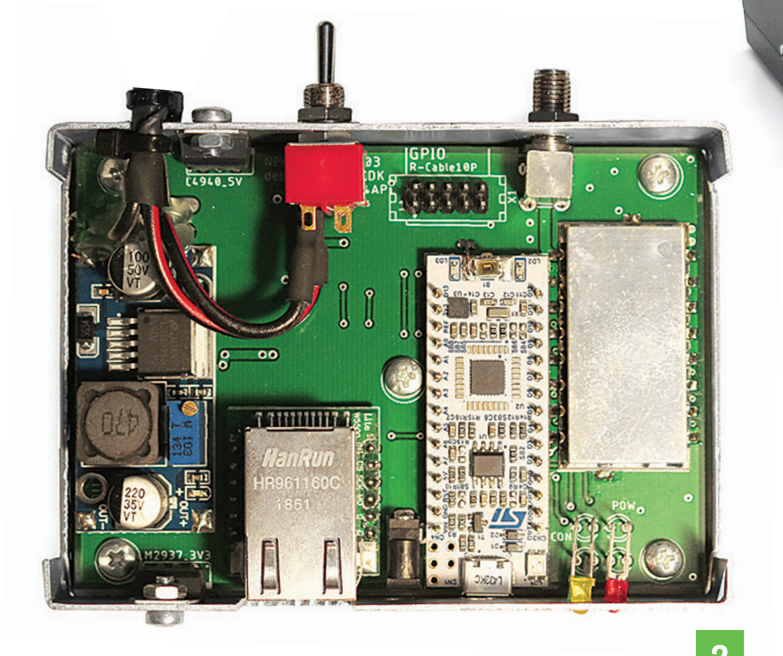
HAM RADIO DOES DISTANT DATA NETWORKING

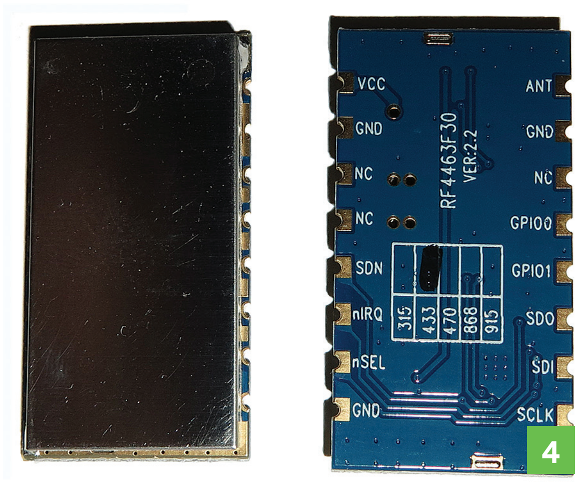
A NEW PROTOCOL RUNS ON SIMPLE HARDWARE AND SUPPORTS IPv4

I HAVE BEEN A HOBBYIST AND MAKER FOR ALMOST 15 years now. I like inventing things and diving into low-level things. In 2013, I was looking at a protocol called NBP, used to create a data network over amateur radio links. NBP was developed in the 2000s as a potential replacement for the venerable AX.25 protocol that's been in use for digital links since the mid-1980s. I believed it was possible to create an even better protocol with a modern design that would be easier to use and inexpensive to physically implement. ● It took six years, but the result is New Packet Radio (NPR), which I chose to publish under my call sign, F4HDK, as a nom de plume. It supports today's de facto universal standard of communication—the Internet's IPv4—and allows data to be transmitted at up to 500 kilobits per second on the popular 70-centimeter UHF ham radio band. Admittedly, 500 kb/s is not as fast as the megabits per second that flow through amateur networks such as the European Hamnet or U.S. AREDN, which use gigahertz frequencies like those of Wi-Fi. But it is still faster than the 1.2 kb/s normally used by AX.25 links, and the 70-cm band permits long-distance links even when obstructions prevent line-of-sight transmissions.

Initially, I considered using different frequency bands for the uplink and downlink connections: Downlinks would have used the DVB-S standard, originally developed for digital satellite television. Uplinks would have used a variation of FSK (frequency-shift keying) to encode data. But the complexity involved in synchronizing the uplink and downlink was too high. Then I tried using a software-defined radio equipped with a field-programmable gate array (FPGA). I had

F4HDK





1. Data links work over the medium-range UHF band, but for the best results, a directional antenna is used. **2.** The modem is principally a microcontroller attached to a radio transceiver. **3.** An amplifier designed for DMR radio links boosts the signal to required levels. **4.** The transceiver is a shielded module built around the Si4463 ISM chip, which operates at 434 megahertz.

some experience with FPGAs thanks to a previous project in which I had implemented a complete custom CPU using an Altera Cyclone 4 FPGA. The goal was to do all the modulation and demodulation using the FPGA, but again the method was too complex. I lost almost two years chasing these ideas to their dead ends.

Then, in one of those why-didn't-I-think-of-this-earlier moments, I turned to ISM (industrial, scientific, and medical) chips. These are transceivers designed to operate in narrow radio frequency bands that were originally allocated for noncommunication purposes, such as RF heating. However, the ISM band has become popular for communications as well because typically a license is not required for its use. In Africa, Europe, and North Asia, there is an ISM band lying inside the 70-cm ham radio band at 434 megahertz, so commercial ISM chips are available for this frequency.

I chose to build my hardware around the Si4463 ISM transceiver: It's cheap, flexible, and available in many modules and breakout boards, and it can handle a raw data rate up to 1 megabyte per second. It's designed for short-range applications, so the radio part of the chip is not optimal, but it works. In order to reach reasonable distances, you need an amplifier to provide more RF power. For my NPR plan, I needed an amplifier that can also switch very rapidly between transmitting and receiving. I found some widely available external 20-watt amplifiers for handheld radios designed for the European-developed Digital Mobile Radio (DMR) standard, which was ratified in 2005. In the DMR standard, radio equipment must be able to handle a complete transmit/receive cycle within 60 milliseconds. I established a minimum of an 80-ms-long cycle time for NPR with this bound in mind.

The ISM transceiver is connected to an Mbed Nucleo STM32 L432KC microcontroller, which uses an Arm Cortex CPU.

This microcontroller is in turn connected to an Ethernet interface, and it takes care of all the details of running the NPR protocol. Any connected PC or network sees the radio link as just another IPv4 connection with no need to install specific NPR software. The NPR modem can be configured over this link or via a USB connection. The total

F4HDK

cost of the hardware is about US $80, and a partner, Funtronics, will be making kits available for purchase online. If you want to build a modem yourself from scratch, detailed instructions and the NPR protocol software are available from my Hackaday project page.

The NPR protocol is based on a hub-and-spoke model, in which a central modem links several client modems. Currently there can be as many as seven modems, although I plan to expand this to 15. The theoretical maximum distance between a client modem and the central modem is 300 kilometers. This limit arises because NPR uses a managed time-division multiple access (TDMA) technique, in which the central modem and the clients each transmit on the same frequency but at different times, with the central modem dictating when each client can transmit, and making scheduling adjustments to account for time delays due to distance. The complete transmit/receive cycle is between 80 ms and 200 ms, depending on the exact type of modulation and data rate chosen.

The creation of the NPR protocol was a very fun part of the project for me: deciding how data should be packed and arranged inside radio frames and how the NPR modems should interact with each other. But after two more years it was time to stop working alone, so I shared NPR with my local ham radio community in France. By the end of 2018, we began testing it in real-world conditions. We have already achieved distances over 80 km, and I am now getting help from the global amateur community, especially in Germany. Currently, NPR is primarily being used to access existing local high-speed amateur radio networks from places that cannot have the line-of-sight radio links required for 2.4- and 5.6-gigahertz signals.

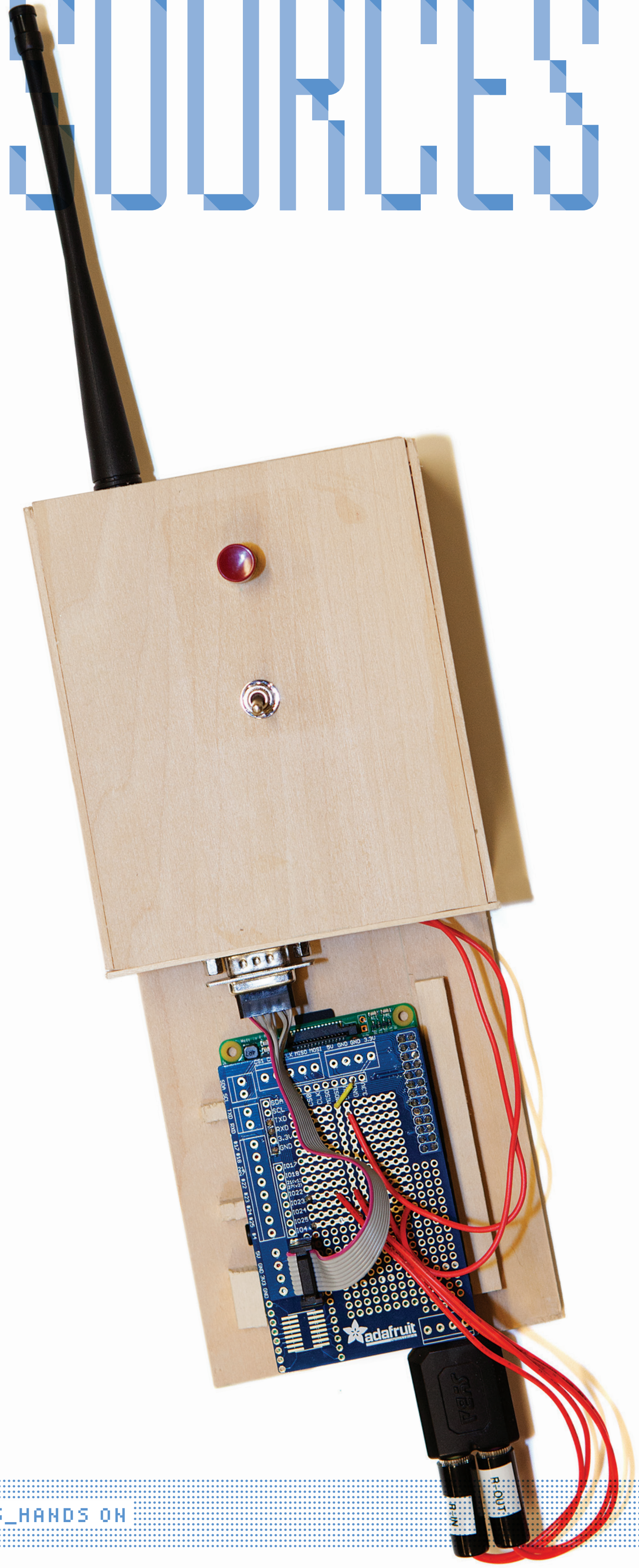
Although it's usable, I would be the first to admit that NPR is a young technology and probably not totally mature. In addition to increasing the number of clients that can be supported by a central modem, I have a number of enhancements in mind, such as adding support for QoS (quality of service) prioritization, so that NPR could be used to transmit digital voice; allowing Ethernet frames to be transported directly; and separating downlink and uplink frequencies. **—F4HDK**

↗ POST YOUR COMMENTS at https://spectrum.ieee.org/NPR1119

RESOURCES

735,405

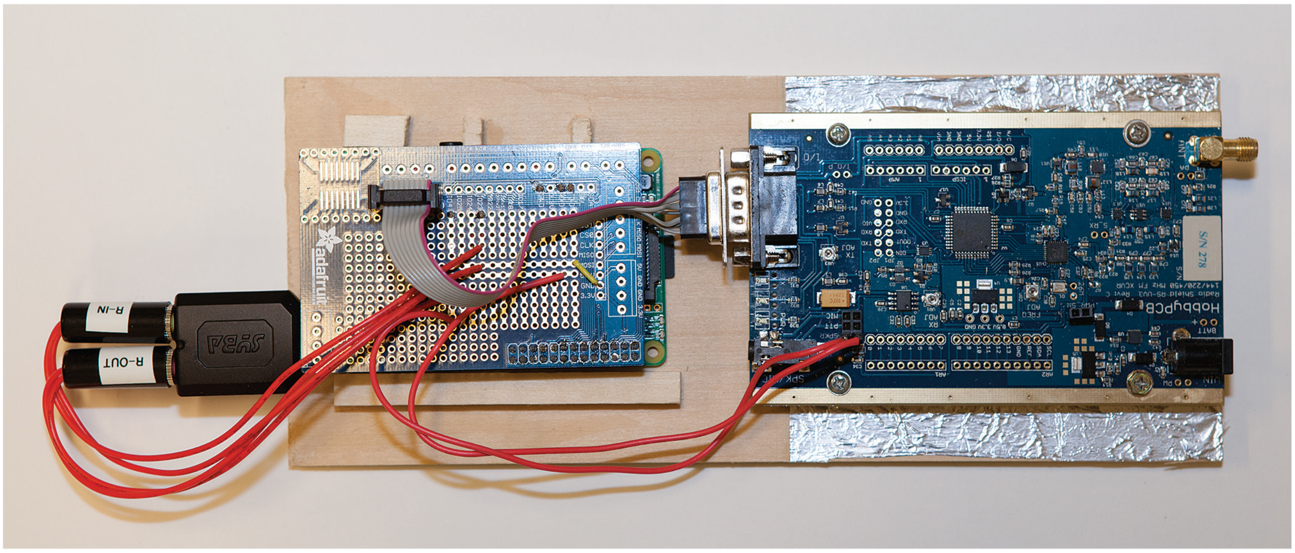
NUMBER OF PEOPLE IN THE UNITED STATES WITH AMATEUR RADIO LICENSES IN 2015

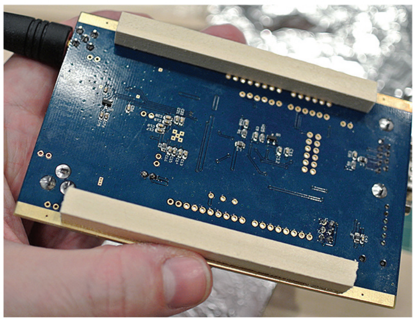


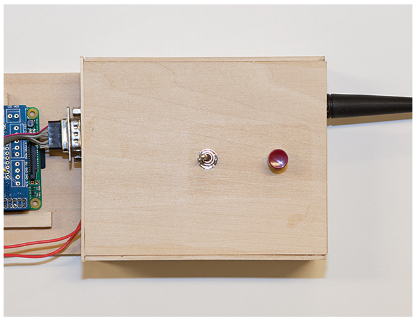
A HAM RADIO FOR MAKERS

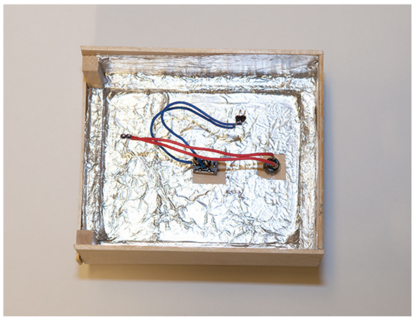
THE RS-UV3 LETS YOU BUILD YOUR OWN ARDUINO- OR PI-BASED RADIO

PHOTOGRAPH BY Randi Klett









The RS-UV3 is a shot in the arm for amateur radio. Mobile phones and the Internet have made the basic act of talking to a faraway person an everyday experience. This means that much of the appeal of ham radio is now in things like emergency response; technically challenging exercises such as bouncing signals off satellites or ultralow-power long-distance contacts; and exploring a host of digital communications modes.

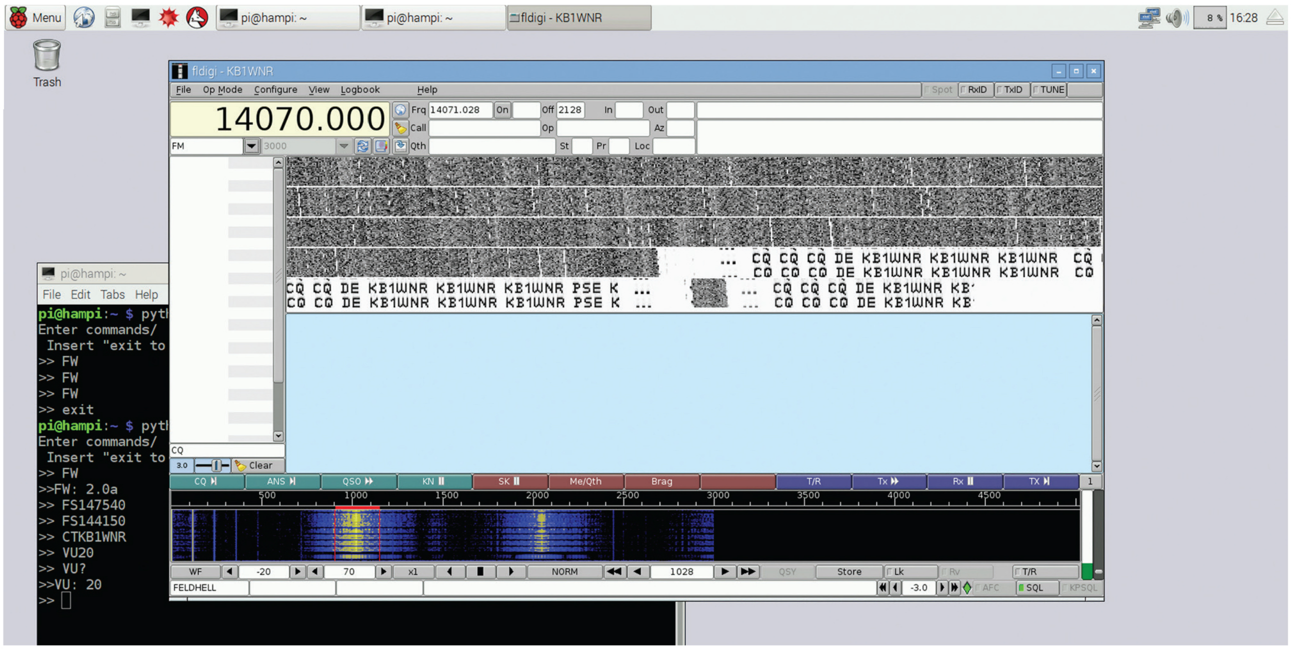
In some ways, trying out such digital modes has never been easier. Free desktop programs like Fldigi can work with the audio tones used in a smorgasbord of communications schemes, from the 1930s-era radio-fax Hellschreiber protocol to today's complete bulletin-board systems. But linking the computers running such software to radios is often surprisingly fiddly in the age of painless USB and Bluetooth. Except for high-end rigs, connecting a computer to a ham radio typically involves navigating legacy interfaces and connectors and can call for specialized additional equipment, a turnoff for makers who might otherwise be interested in the possibilities of radio.

But the US $90 RS-UV3 radio shield from HobbyPCB is an FM transceiver that's welcoming by design and built for makers from the ground up. It's not the only radio shield available for the Arduino that works in the UHF/VHF frequency bands, but the RS-UV3 is the most flexible one I've spotted when it comes to interfaces. As well as a legacy ham radio interface, the shield provides multiple ways to connect it to an Arduino, PC, or Raspberry Pi—and to two of those at once if required. By itself, the RS-UV3 can transmit with only 0.25 watt, but HobbyPCB plans to sell an add-on amplifier for more powerful transmissions.

(A quick legal aside: Using the RS-UV3 to transmit requires an amateur radio license. Depending on your country, you may also require a license to receive, so check your local regulatory regime.)

Unlike standard ham radios, the RS-UV3 can't operate as a stand-alone device. First you have to choose how to turn it on, either by soldering a jumper across the power line, or as I did, by wiring in a connector to which I attached a switch. The onboard processor then boots up in what's known as simplex mode (where the same frequency is used for both reception and transmission) at a default frequency of 146.52 megahertz. Plugging in a handheld speaker/microphone with a push-to-talk (PTT) button via a jack on the RS-UV3 will let you talk back and forth on that frequency. Alternatively, you could wire an electret microphone, small speaker, and PTT button directly to the shield.

BOTTOM LEFT AND OPPOSITE PAGE: STEPHEN CASS; THIS PAGE: RANDI KLETT (3)



```
pi@hampi:~ $ pyth
Enter commands/
 Insert "exit to
>> FW
>> FW
>> FW
>> exit
pi@hampi:~ $ pyth
Enter commands/
 Insert "exit to
>> FW
>>FW: 2.0a
>> FS147540
>> FS144150
>> CTKB1WNR
>> VU20
>> VU?
>>VU: 20
>>
```

```
                                 ... CQ CQ CQ DE KB1WNR KB1WNR KB1WNR  CQ
                                 ... CQ CQ CQ DE KB1WNR KB1WNR KB1WNR  CQ
CQ CQ DE KB1WNR KB1WNR KB1WNR PSE K ...       ... CQ CQ CQ DE KB1WNR KB
CQ CQ DE KB1WNR KB1WNR KB1WNR PSE K ...       ... CQ CQ CQ DE KB1WNR KB
```

RASPBERRY PI RADIO: The RS-UV3 was screwed to wooden supports [bottom left] and mounted. A USB adapter allows audio to pass between the RS-UV3 and a Pi, while commands are passed via a DB-9 connector [top left]. To reduce interference, I housed the RS-UV3 in a foil-lined enclosure [bottom right and middle]. Signals are decoded with Fldigi [above].

But most people who have a radio want to tune it to more than one frequency. You accomplish this, and other functions such as adjusting the volume, by setting up a serial connection with the RS-UV3's processor and sending it commands. (The setup is reminiscent of the old dial-up modem Hayes command codes.)

You can establish the serial connection in a number of ways, such as mounting an Arduino directly to the RS-UV3 using the provided through-holes and then wiring two of the Arduino's input/output pins to a header on the shield. Another option is to use a header intended for FTDI serial cables and connect it to a PC's USB port.

Or you can use the shield's versatile DB-9 connector. Along with serial transmit and receive lines, the connector also provides audio input and output to the transceiver and a PTT control line. These serial lines use 3.3 volts, rather than the typical 5 V. This is handy because 3.3 V is the operating voltage of the Raspberry Pi. Consequently, you can connect the RS-UV3 directly to the general-purpose input/output header (GPIO) of the Pi without much in the way of interface electronics.

And thanks to the upgrade in processing power that came with the release of the Raspberry Pi 2, the Pi now has enough horsepower to run Fldigi in addition to controlling the RS-UV3. So I did just that.

First, I built a simple hardware interface between the Pi's GPIO port and the RS-UV3's DB-9 connector using an old prototyping "hat" from Adafruit I had lying around. I also created a wooden enclosure (as is my wont) for the shield—lined with tinfoil to reduce radio frequency interference—with a power switch and a PTT button on top and a slot to hold the Pi.

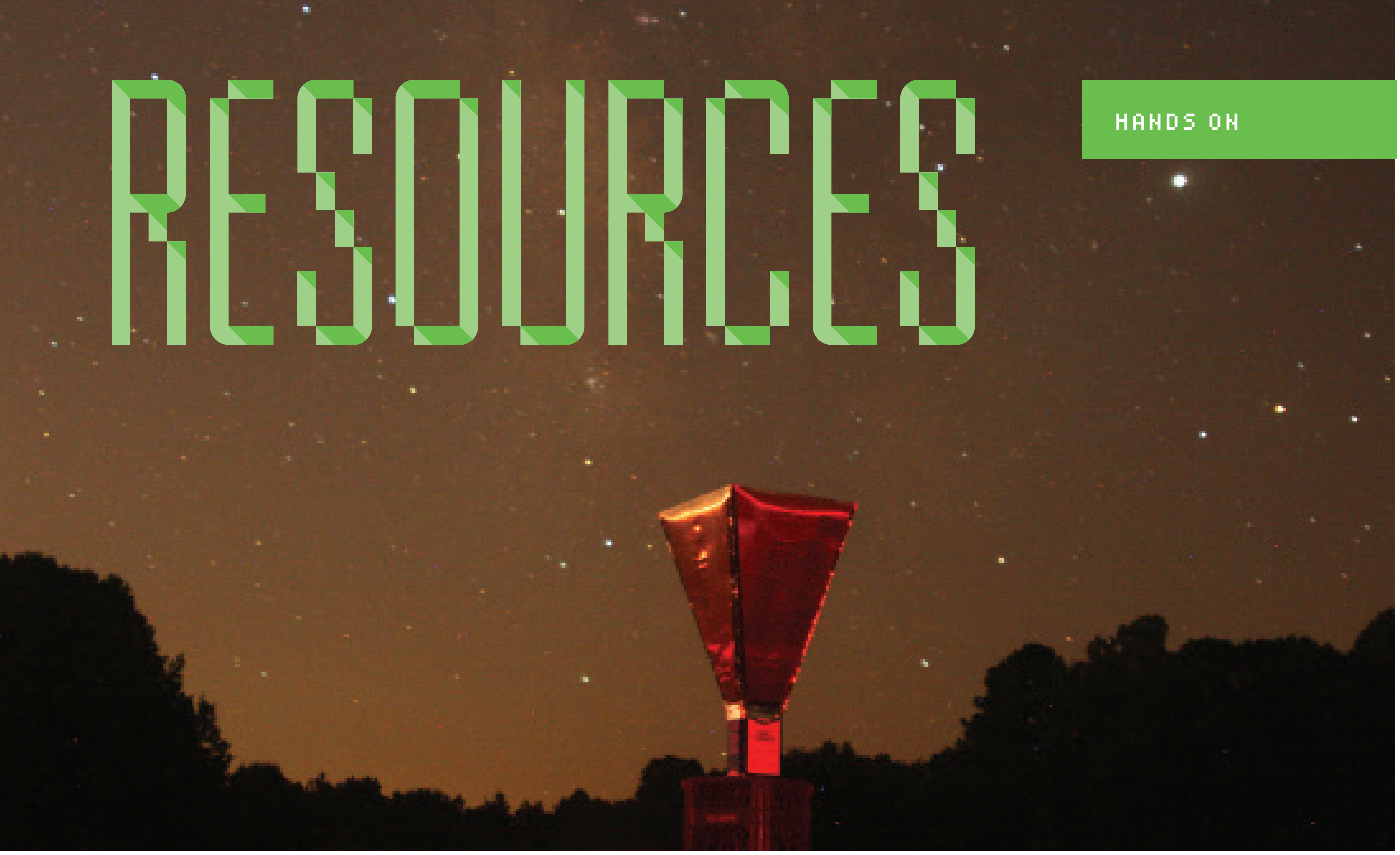
I used the Pi's configuration tool so that the Raspbian operating system wouldn't reserve the serial pins on the GPIO for its own use. I tweaked a Python script I found online, written by Fabio Varesano, to create a basic command-line terminal to send commands to the RS-UV3.

Then I turned my attention to using Fldigi to listen to and generate signals for the transceiver. I wired up two mono audio jacks to the RS-UV3's audio input and output feeds. Unfortunately, while the Pi has a built-in socket for audio output, it doesn't have one for audio input. So, following the advice of Lior Elazary's website, I purchased a Syba USB audio adapter for the Pi and plugged the jacks into that.

Installing Fldigi from its source code was straightforward (I followed Jeffrey Kopcak's online instructions, although without the remote access steps).

Firing everything up (with the Pi connected to a keyboard, mouse, and monitor), I was able to set the RS-UV3 to a suitable frequency and communicate via a number of digital modes to a nearby test rig (which consisted simply of me holding up a handheld ham radio to a laptop running Fldigi through its built-in loudspeakers and microphone in the quiet of *IEEE Spectrum*'s offices late at night). Success! I now have a dedicated digital radio rig, for less than $150 all told.

Future plans will require obtaining an amplifier, but I hope to build on this basic setup to create a complete packet radio bulletin-board system that will fit in a relatively small box. Just in case that whole mobile phone/Internet thing has a hiccup. —**STEPHEN CASS**



BUILD YOUR OWN RADIO TELESCOPE

TRACK THE MOVEMENT OF THE MILKY WAY

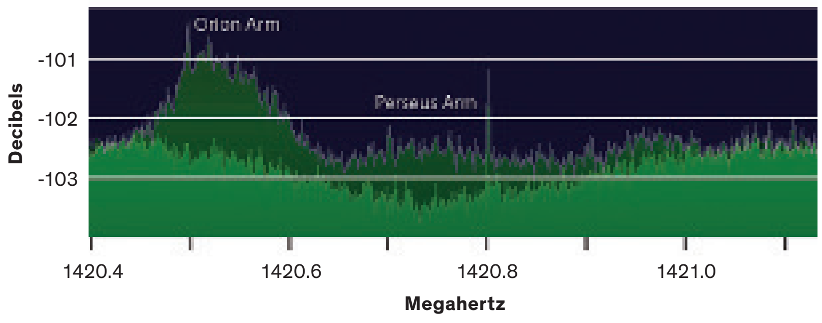
A young friend recently spent a week learning about radio astronomy at the Green Bank Observatory in West Virginia. His experience prompted me to ask: How big a radio antenna would you need to observe anything interesting? It turns out the answer is a half meter across. For less than US$150 I built one that size, and it can easily detect the motions of the spiral arms of the Milky Way galaxy. Wow!

My quest began with a used satellite-TV dish and a "cantenna" waveguide made from a coffee can placed at the dish's focus. Unfortunately, I had to abandon this simple approach when I figured out that a coffee can was too small to work for the wavelength I was most interested in: 21 centimeters. That is the wavelength of neutral hydrogen emissions, a favorite of radio astronomers because it can be used to map the location and motion of clouds of interstellar gas, such as those in the spiral arms of our galaxy.

Some research revealed that amateur radio astronomers were having success with modest horn antennas. So I copied the example of many in an online group called Open Source Radio Telescopes and purchased some aluminized foam-board insulation as antenna construction material. But I was troubled when my multimeter showed no evidence that the aluminized surface could conduct electricity. To make sure the material would have the desired effect on radio waves, I built a small box out of this foam board and put my cellphone inside it. It should have been completely shielded, but my cellphone received calls just fine.

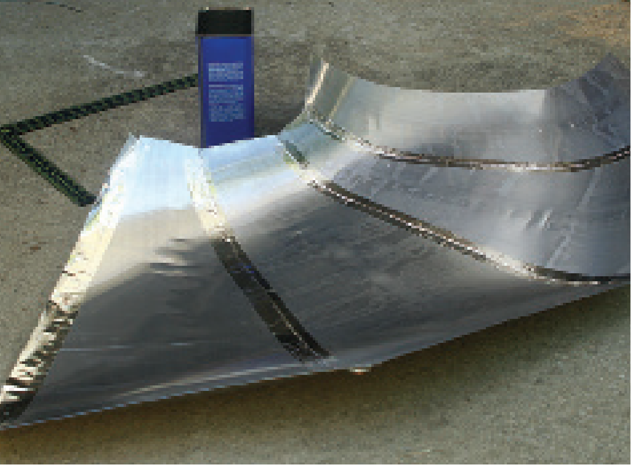
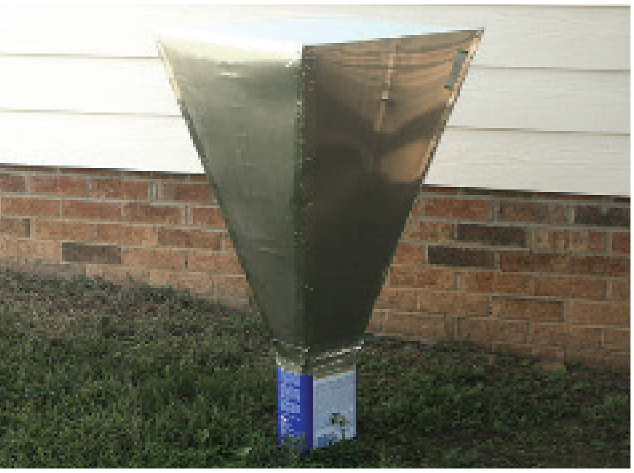
That experiment still perplexes me, because people definitely do build radio telescopes out of aluminized foam board. In any case, I abandoned the board and for $13 purchased a roll of 20-inch-wide (51-centimeter-wide) aluminum flashing—the thin sheet metal used to weatherproof tricky spots on roofs.

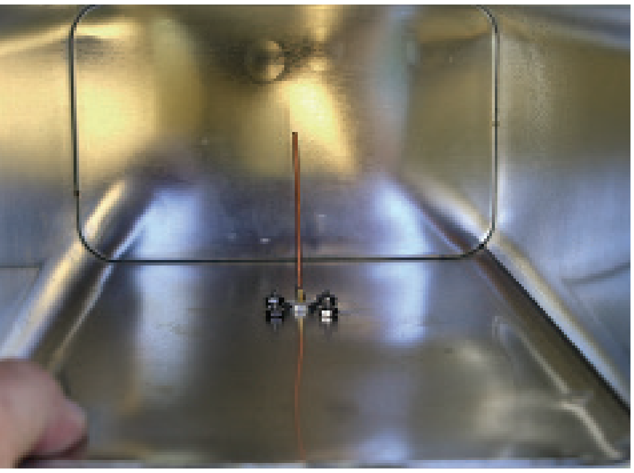
The width of my roll determined the aperture of my horn's wide end. The roll was 10 feet (3 meters) long, which limited the length of the four sides to 75 cm. An online calculator showed that a horn of those dimensions would have a respectable directional gain of 17 decibels. Some hours with snips and aluminized HVAC tape ($8) resulted in a small horn anten-



THE SKY ELECTRIC: The author made a horn antenna [top] out of aluminum flashing and a metal can. The antenna can detect emissions from the hydrogen gas in nearby arms of the galaxy: Dark green [above] represents the signal from the sky; light green indicates a comparison baseline system response with no signal.

DAVID SCHNEIDER



METAL ON METAL: Four sheets of aluminum flashing form the antenna horn [top]. The narrow end of the horn is attached to a waveguide made from an empty paint-thinner can [second from top]. The signal is picked up by a wire projecting into the waveguide and sent to a receiver via a coaxial cable [second from bottom and bottom].

na. Attaching a square of ordinary foam board (not the aluminized kind) to the open end made it plenty robust.

I also purchased a 1-gallon can of paint thinner ($9) and gave away its contents. The empty can serves as a waveguide feed at the base of the horn antenna. A handy online waveguide calculator told me this feed would have an operating range that nicely brackets the neutral-hydrogen line frequency of 1420 megahertz.

Some folks contributing to Open Source Radio Telescopes were using similar cans. But none of the projects' documentation showed exactly how to construct the feed's "pin": the part that picks up signals inside the waveguide and passes them to the telescope's receiver. Many cantenna tutorials say to make the pin a quarter of a wavelength long, which in this case works out to 53 millimeters. The tricky part is figuring out where to place it in the can—it needs to be a quarter of a wavelength from the base. However, in this case the relevant wavelength isn't 21 centimeters but what's called the guide wavelength, which corrects for the difference between how the signal propagates in free space versus inside the waveguide. An online tutorial and another calculator showed the appropriate distance from the base to be 68 mm. So that's where I drilled a hole to accommodate an N-type coaxial bulkhead connector that I had purchased on Amazon.com for $5, along with an N-to-SMA adapter ($7).

For my receiver, I went with a USB dongle that contains a television tuner plus a free software-defined radio application called HDSDR. (The software was chosen on the basis of a report from two amateur radio astronomers in Slovenia who had used it to good effect.)

I purchased the dongle from Nooelec.com ($37) because that company had also recently started selling a gizmo that seemed perfect for my application: It contains two low-noise amplifiers and a surface-acoustic-wave (SAW) filter centered on 1420 MHz ($38). The dongle itself provides power for the amplifier through the coaxial cable that connects them, a 30-cm (12-inch) length of coax purchased on Amazon.com ($9). The dongle just sits on the ground next to my horn and is attached to a Windows laptop through a USB extension cable.

At my instrument's "first light," I was able to detect the neutral hydrogen line with just a little squinting. After getting more familiar with the HDSDR software, I figured out how to time-average the signal and focus on the spectral plot, which I adjusted to display average power.

This plot distinctly showed a hydrogen "line" (really a fat bump) when I pointed my horn at the star Deneb, which is a convenient guide star in the constellation of Cygnus. Point at Cygnus and you'll receive a strong signal from the local arm of the Milky Way very near the expected 1420.4-MHz frequency. Point it toward Cassiopeia, at a higher galactic longitude, and you'll see the hydrogen-line signal shift to 1420.5 MHz—a subtle Doppler shift indicating that the material giving off these radio waves is speeding toward us in a relative sense. With some hunting, you may be able to discern two or more distinct signals at different frequencies coming from different spiral arms of the Milky Way.

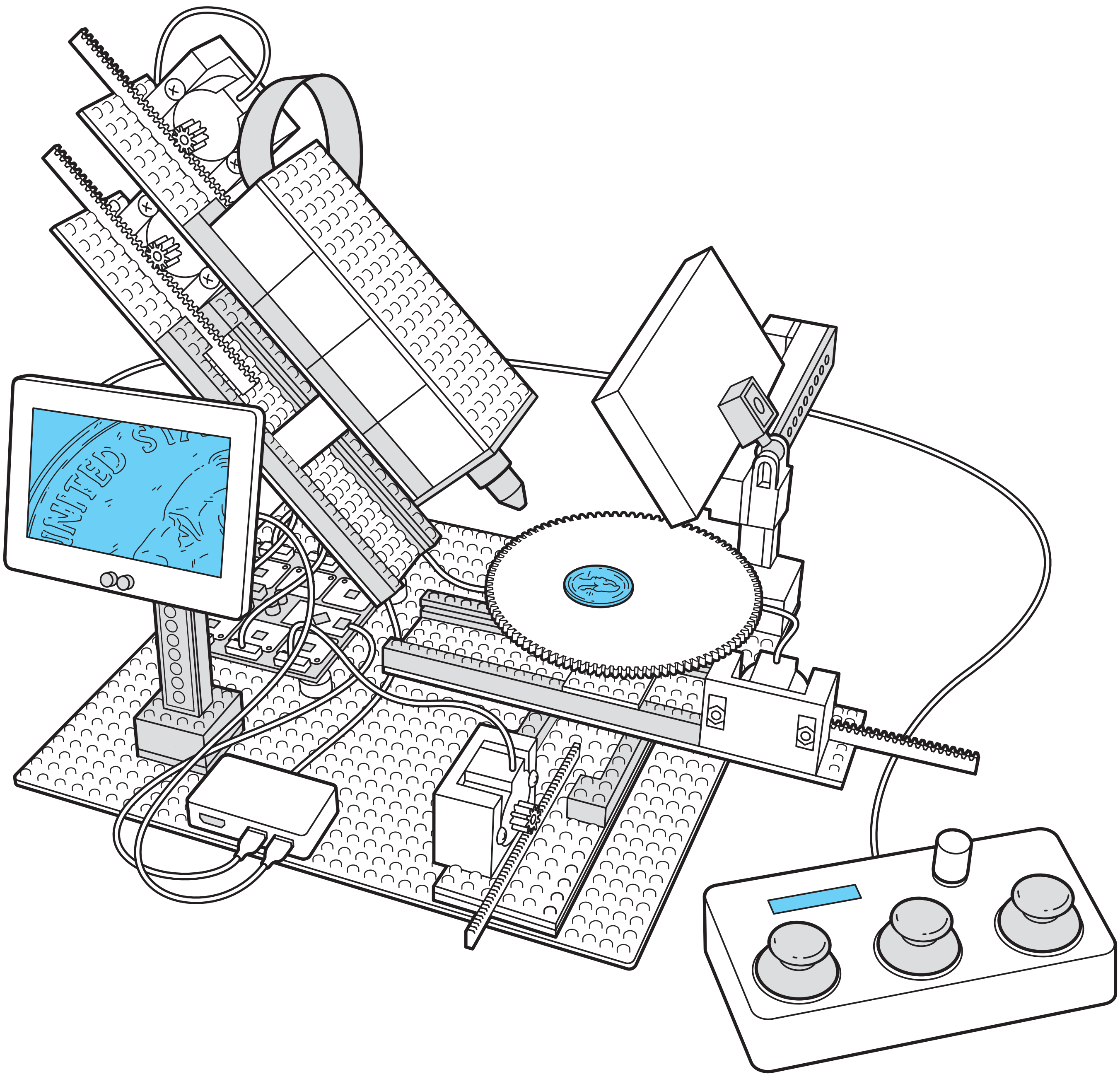
Don't expect to hear E.T., but being able to map the Milky Way in this fashion feels strangely empowering. It'll be $150 well spent.

—DAVID SCHNEIDER

↗ **POST YOUR COMMENTS** at https://spectrum.ieee.org/telescope1019

DAVID SCHNEIDER

Hands On

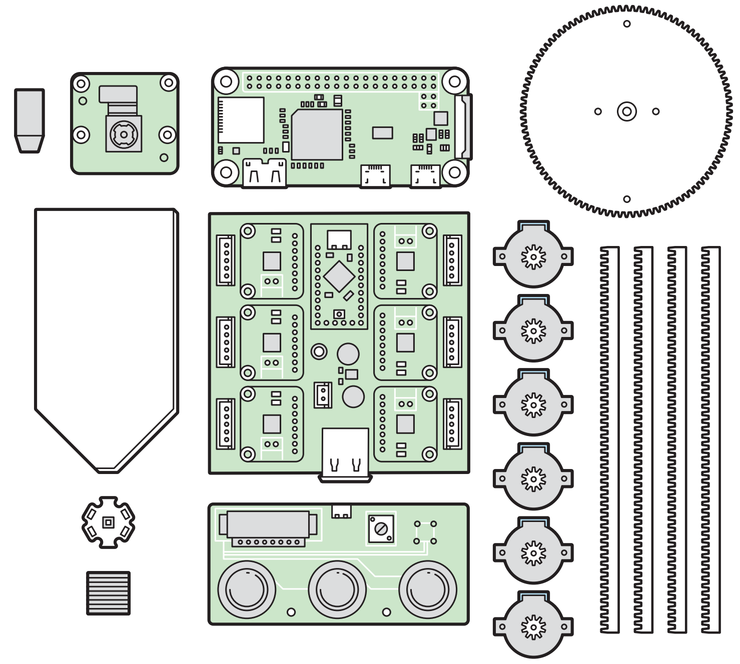


HANDS ON BY YUKSEL TEMIZ

ILLUSTRATIONS BY James Provost

THE LEGO MICROSCOPE

A VALUABLE LAB TOOL BEGAN AS A DIY PROJECT



MIXED MATERIALS: The imaging microscope design uses a potpourri of technologies and materials, including Lego pieces for its main structural components and 3D-printed cogs and racks to drive its moving parts. Stepper motors, which allow for precise movements, are powered by a motor driver board and controlled by an Arduino board. A Raspberry Pi Zero and Pi camera module are used to capture images. The original design incorporated custom-made control boards and parts printed on a high-resolution printer, but prior to public release the microscope was redesigned to be assembled with off-the-shelf boards and parts that could be printed on less expensive, low-resolution printers.

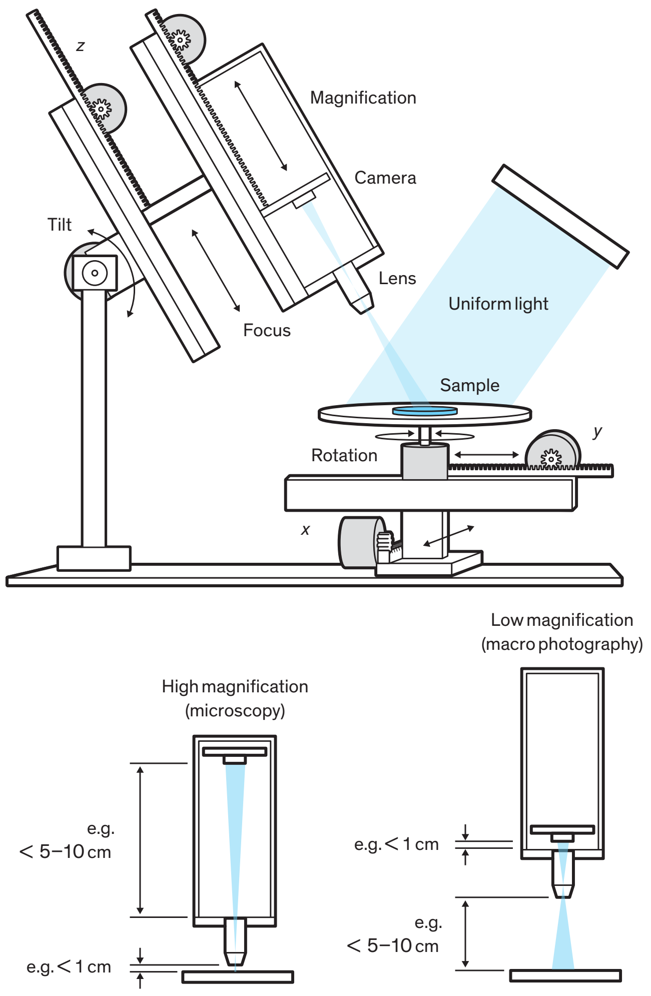
I AM A MEMBER OF A TEAM at IBM Research–Europe, in Zurich, developing microfluidic technologies for medical applications. Two years ago, I was asked to provide high-quality photos and videos of our microfluidic chips for a big tech event. I borrowed a 4K camcorder from a colleague, attached a macro lens to it, built a custom light diffuser using an LED matrix and polyester film, and positioned everything using a high-end tripod and a few micromanipulators. I was able to take eye-catching videos as liquids filled microfluidic channels. It was clear to me that this should be the new level of quality and style for our publications and presentations. However, my photo setup occupied half a bench in our lab and it required hours of fine adjustments to record one shot.

We have a tradition of inventing microscopes at IBM in Zurich. In 1981, Gerd Binnig and Heinrich Rohrer created the scanning tunneling microscope here. As a DIY enthusiast, I quickly found myself in my own quest to build a better setup. The result was a US $300 modular and motorized microscope that combines my three favorite adulthood hobbies: Arduino, Raspberry Pi, and Lego.

Taking a photo of a microfluidic chip is not easy. The chips are typically too big to fit into the field of view of a standard microscope, but they have fine features that cannot be resolved using a regular camera. Uniform illumination is also critical because the chips are often made of highly reflective or transparent materials. Looking at publications from other research groups, it's obvious that this is a common challenge. With this motivation, I devoted some of my free time to designing a multipurpose and compact lab instrument that can take macro photos from almost any angle.

My first prototype was a Raspberry Pi camera module mounted on a stage that could be moved in three dimensions using the mechanisms of linear stepper motors from old CD-ROM drives. The Raspberry Pi camera was an ideal choice because it allows the manual adjustment of critical parameters like ISO settings and exposure time. I carefully removed the plastic casing holding the lens, revealing

ILLUSTRATIONS BY James Provost



PERFECT ANGLE: The Lego microscope allows for a sample to be placed under uniform illumination provided by an LED backlight module. The sample can then be moved forward, back, left, and right and also rotated to find the desired view. The microscope body can be tilted up and down, and its proximity to the sample and focus adjusted as well to provide different degrees of magnification [bottom]. The focus is adjusted by moving a lensless camera module within a Lego housing, altering its distance from the objective lens at the base of the housing.

the CMOS image sensor, and engineered a delicate mechanism to move the lens back and forth in fine steps so I could take high-magnification macro photos. The setup worked well for a while, but it was fragile. I broke the lens mechanism and damaged the image sensor several times by accidentally forcing the moving parts beyond their limits.

I decided to take a different approach. I removed the lens entirely from the Pi camera. Then I took the objective lens from a low-cost USB microscope and mounted it on another CD-ROM linear actuator so that the objective lens could move back and forth along the optical axis of the Pi camera. I built a housing from Lego bricks to shield the camera's exposed sensor.

However, this attempt resulted in nothing but an appreciation of the high price tag of the linear stages used in microscopes: The travel distance of the CD-ROM actuator was too short to achieve a wide magnification range.

I switched to a lead-screw mechanism used in 3D printers. Instead of commonly used 8-millimeter-diameter threads, shafts, and bearings, I used 3-mm-diameter components to keep things compact. Also, moving the objective lens caused problems in eliminating stray light, so I decided to move the camera sensor instead. I built a stage that allowed the object being imaged to be moved along the *x* and *y* axes and rotated. In total, I wound up with six miniature stepper motors with gearboxes that allow me to move the stage, tilt the microscope, adjust its distance to the object, and focus the image.

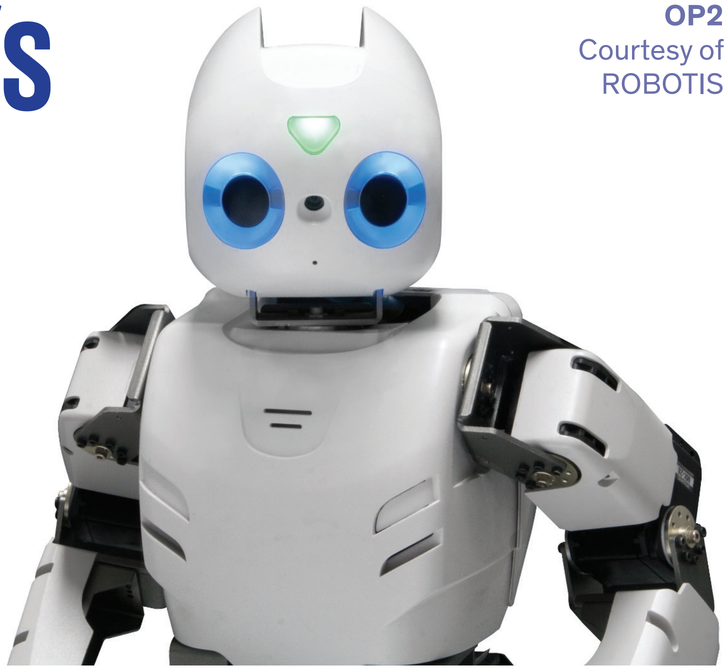
I often design my own Arduino boards for a more compact implementation. This time, I designed a board measuring 18 by 18 mm and featuring an ATtiny84 microcontroller and a DRV8834 stepper-motor driver. The setup gave surprisingly good image quality, for not only beauty pictures of chips but also for examining features with dimensions of a few micrometers, and even as a digital goniometer for measuring contact angles. I had started the project for a specific need, but it became clear to me that this could be a multipurpose imaging system that anyone can assemble and use at home or school.

IBM and my managers supported me in publicly sharing the assembly instructions. However, as I prepared the instructions for public releases, several issues started bothering me. I had built everything using a state-of-the-art 3D printer and a fully equipped mechanical workshop. Also, the small stepper motors I used were expensive and not available in popular hobby electronics stores. Programming the ATtiny84 via a dedicated ISP programmer was certainly not as easy as programming commercial Arduino boards with a USB port. I went back to the drawing board and redesigned everything using easily accessible components, for example Arduino boards and stepper-motor drivers from Adafruit Industries, and 28BYJ-48 stepper motors, which can be found anywhere for a few dollars. I also replaced the LED-matrix light source with a more maker-friendly and lower-cost version. I bought an LED backlight module from Adafruit for $3 and attached a high-power LED. The intensity was slightly lower than that of the original LED matrix but the uniformity was pretty good for both reflected- and transmitted-light microscopy. For the new linear actuators, I combined "sliding" pieces from Lego with a rack-and-pinion gear combination that I designed using FreeCAD's gear toolbox and printed using my personal Creality Ender 3 Printer. The new design worked as well as the previous one, if not better.

POST YOUR COMMENTS AT spectrum.ieee.org/scope-may2020

There are probably still many things that could be improved, and I hope this prototype persuades other makers to try new and better ideas. Could it replace a laboratory microscope? Maybe not, but it's a great solution for budget-constrained schools, which is why we have made the instructions open source for everyone to access and enjoy. ■

The World's Best ROBOTS GUIDE Is Here!

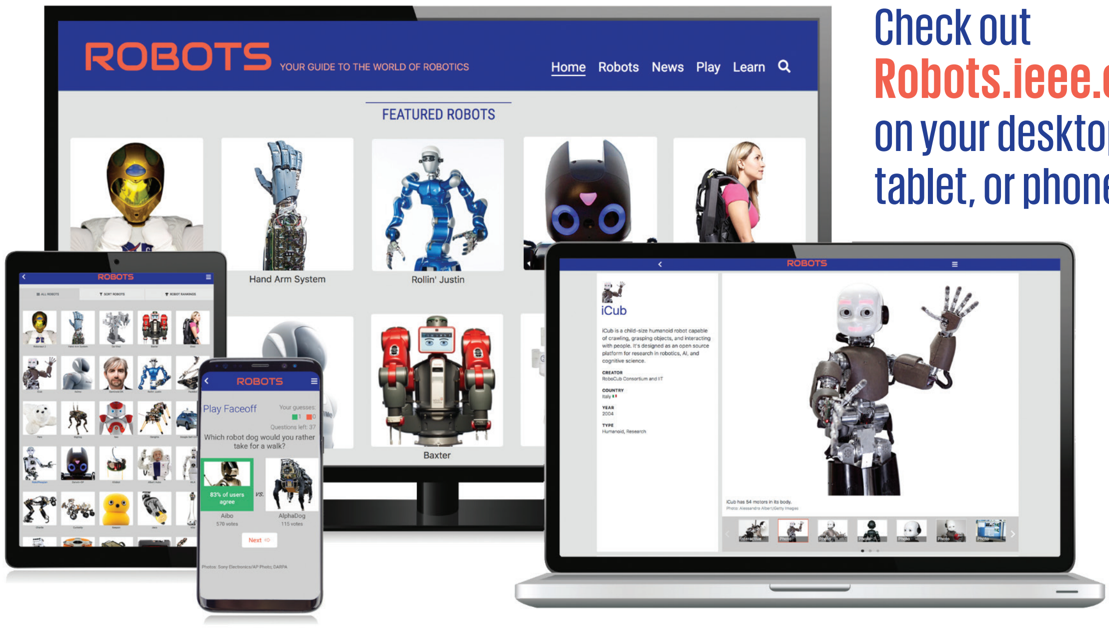


OP2
Courtesy of
ROBOTIS

ROBOTS.IEEE.ORG

IEEE Spectrum's new **ROBOTS site** features more than **200 robots** from around the world.

- Spin, swipe and tap to make robots move.
- Read up-to-date robotics news.
- Rate robots and check their ranking.
- View photography, videos and technical specs.
- Play *Faceoff*, an interactive question game.



Check out
Robots.ieee.org
on your desktop, tablet, or phone now!