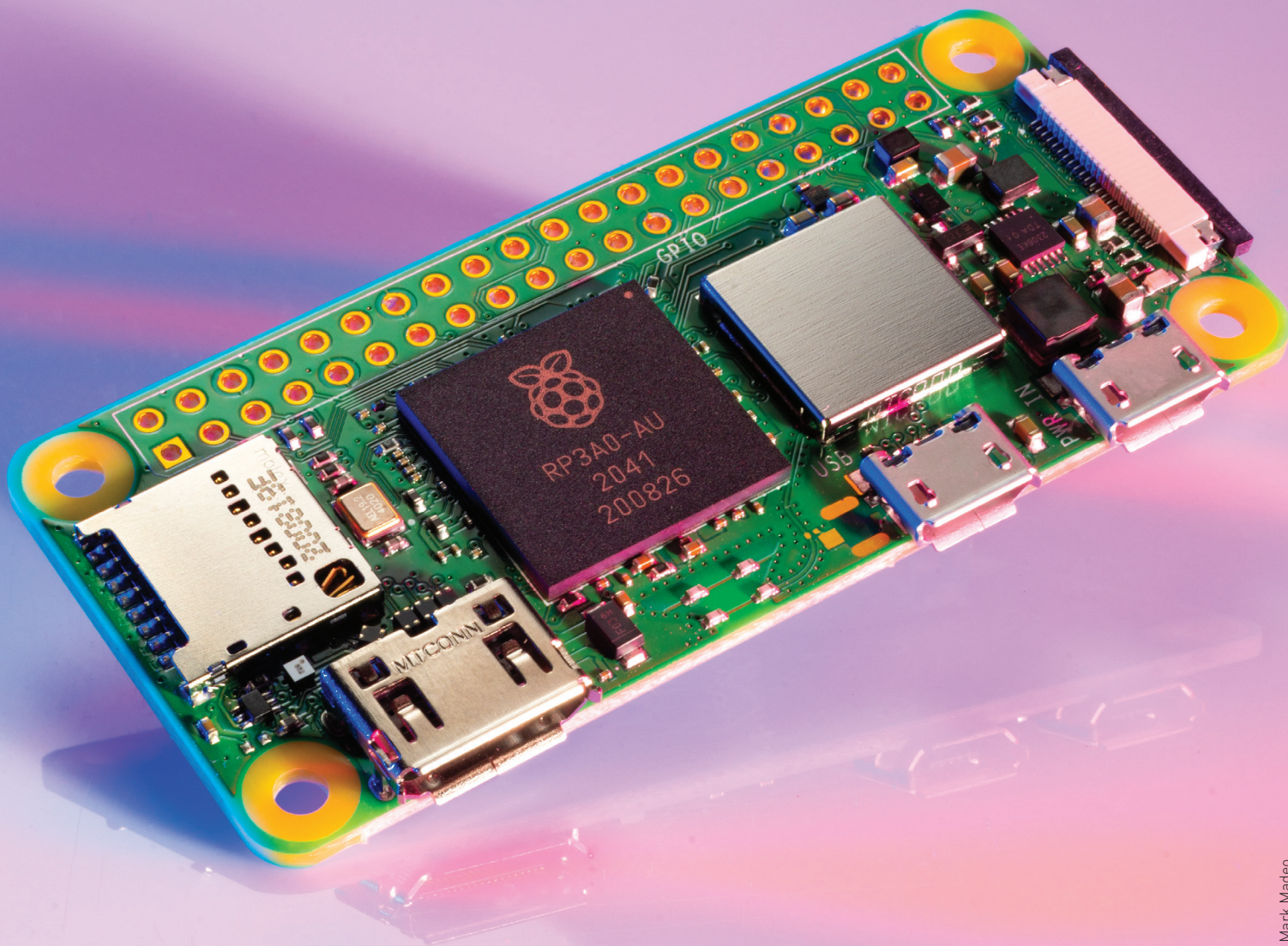


# Make:

Brought to you by *Make:* magazine

# THE Pi DAY COLLECTION

## 17 DIY RASPBERRY PI PROJECTS



Mark Madeo

[make.co](http://make.co) • [makezine.com](http://makezine.com) • [makerfaire.com](http://makerfaire.com) • [makershed.com](http://makershed.com)



# Step Right Up Raspberry Pi Fans!

**Make:** has cooked up something new and delicious just for you. Dive into this veritable feast of the best Pi projects from all corners of our magazine, shed, and faires. And don't forget to pass it around so your family & friends can join in on the fun too!

**Subscribe and Save Today!**

[subscribe.makezine.com/PIDAY](https://subscribe.makezine.com/PIDAY)

ON THE GO



IN PRINT



ONLINE

Adobe Stock-Production Perig

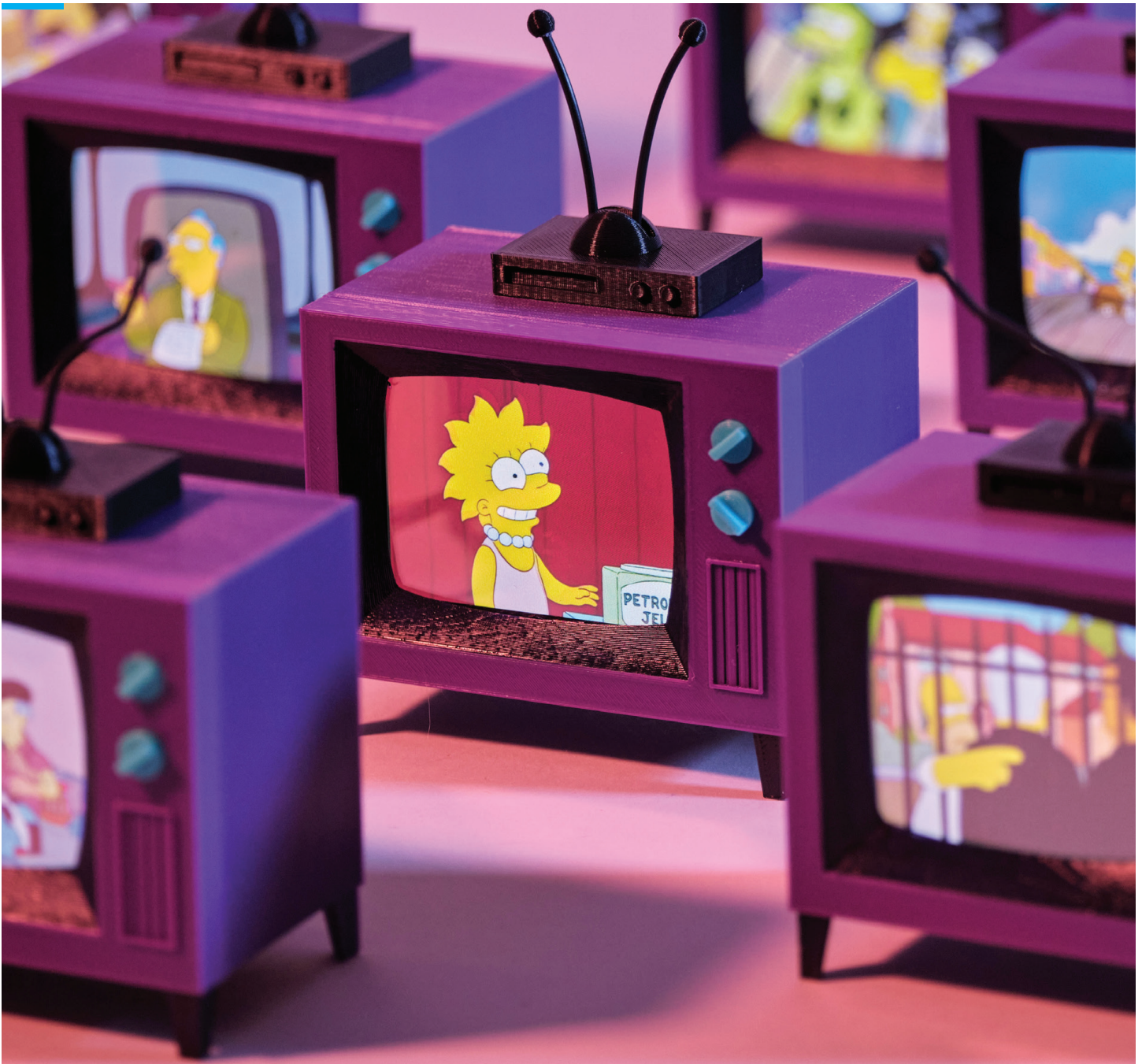


# Project Contents:

- Mini Simpsons TV 04
- Neverending Quotes 14
- Ambient Television Backlight 18
- Meteor Camera 26
- Magic GIF-Ball 32
- Intruder Alarm 38
- Birdsong Identifier 46
- Spotify iPod 47
- Open Smart Cam 55
- Wi-Fi Photo Booth 61
- Autonomous Donkey Car 62
- Weather Station 63
- Musical Drumming Robot 64
- Data Preserver 65
- Skycam Robot 66
- Remote Viewable Camera 67
- DIY Ring Doorbell 67







# Build a Mini Simpsons TV

Ay, caramba! There's always something good on — when you print, solder, and code this tiny tube

Written and photographed by Brandon Withrow



**TIME REQUIRED:****4–5 Hours +10 Hour Print Time****DIFFICULTY: Intermediate****COST: \$50–\$100****MATERIALS**

Buying through my Amazon affiliate links at [withrow.io/simpsons-tv-build-guide](http://withrow.io/simpsons-tv-build-guide) would help me cover my time making this guide — at no cost to you! Consider it one back scratching another, or something like that.

- » **Raspberry Pi Zero W, without header pins**  
This is important!
- » **TFT LCD touchscreen display, 2.8", 640×480**  
iUniker Raspberry Pi Zero W Screen
- » **Audio amplifier, 2.5W mono** Adafruit 2130
- » **Audio speaker, 1.5", 4Ω 3W** Gikfun  
LYSB01LN8ONG4-ELECTR
- » **MicroSD card, 64GB** SanDisk SDSQUA4-064G-GN6MA
- » **Micro USB to DIP adapter board** Amazon  
B07W844N43
- » **Micro USB male solder plug** Amazon  
B08SC94GP1 or B07G5ZY7MH
- » **Trim potentiometer, 1kΩ** Amazon  
B07QX3BX9R, B07C3XHVVX, or B07Q8RSNP7
- » **Switch, micro pushbutton** Amazon  
B07Y3C1MKR, B01MRP025V, or B07WSTFB47
- » **Wire, 22 gauge, black/red pair**
- » **3D printed parts** Download the free STL files from [thingiverse.com/thing:4943159](http://thingiverse.com/thing:4943159). I used 3 colors of Inland PLA+ filament: Purple ([amzn.to/38IAaAi](http://amzn.to/38IAaAi)), Light Blue ([amzn.to/2UWFmrc](http://amzn.to/2UWFmrc)), and Black ([amzn.to/3yoYIMX](http://amzn.to/3yoYIMX)).

**TOOLS**

- » **3D printer**
- » **Soldering iron** with a fine point tip
- » **Solder** I like 60/40 rosin core, around .032" diameter.
- » **Hot glue gun** The bedazzler!
- » **Hobby knife** Always cut away from yourself!
- » **Snips**
- » **USB flash drive with micro-USB adapter**
- » **Slotted screwdriver, small**
- » **Cyanoacrylate (CA) glue** aka super glue
- » **Acrylic paint, dark purple**
- » **Helping hands (optional)** such as QuadHands QH-WB-68. Not necessary, but makes soldering loads easier.



**BRANDON WITHROW** is a software developer and maker living in Kansas City, Missouri. A passion for animation led him into a career as a software developer. While working at Airbnb he developed Lottie, an open source animation library. In his free time he likes to build things in his workshop.

**What is it? A working desktop TV that plays *The Simpsons* on loop.** This project was born from a childhood spent in front of a TV, playing with Legos. I wanted to recreate the “always on” random nature of television, in a tiny desktop format. The videos are always “playing,” even when the screen is off. Like the television in the ancient days before the internet, you just turn it on and watch whatever it gives ya.

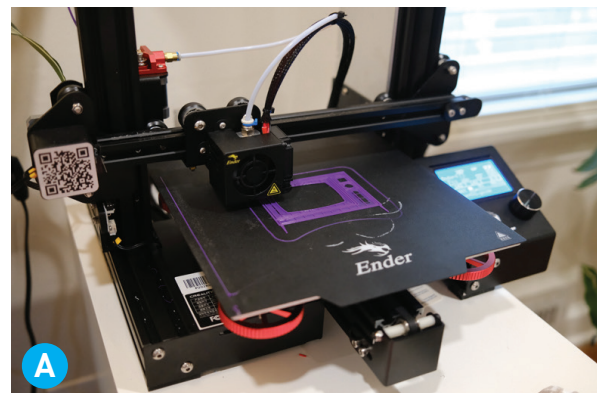
This guide will walk you through printing, building, and coding a small TV that will play videos (that you provide) at random. By the end of this guide you will have your very own tiny TV, with power and volume control. You can easily buy all the parts except the enclosure, which needs to be 3D printed. The TV is built with a Raspberry Pi microcomputer, running Linux. It might sound intimidating, but stick with me and it can be fun!

This guide is aimed at all skill levels but admittedly a few parts, like the soldering, are more intermediate than beginner. I believe you can do it, though — I’ve had reports from people completing the guide that this was their first time doing any of it! Pretty cool. The entire project, including print time, takes about 14 hours to complete.

**1. PRINT THE PARTS**

I designed the TV and parts in Fusion 360 and printed them on an Ender 3 Pro (Figure A). It’s a good idea to start these prints and let them run while you set up the hardware. All of the parts together take about 10 hours.

- **Filament** — I used 3 colors of Inland PLA+ filament: Purple for the case, Light Blue for the knobs, and Black for the VCR/antenna — but have fun with it! If you print it in some fun colors, send me a pic at [twitter.com/thewithra](https://twitter.com/thewithra).





- Print settings — Almost all the pieces can be printed without supports, and with minimal infill. The only exception is the front piece. Print it with a minimal support and a low infill (10%). All pieces should be printed with their build plate orientation matching Figure B for best results. The larger parts I printed in Standard Quality (0.2mm), the smaller parts in Dynamic Quality (0.16mm). Get those pieces printing and then come back so we can get to work on the fun stuff.

## 2. VIDEO ENCODING

This could take some time, so it's best to start this early as well. Your videos — your completely legally-owned videos — must be encoded into a specific format and put onto a thumb drive to transfer them to the Pi later. We're using the H264 format with a height of 480 pixels. Other codecs would be better/smaller, but support of codecs on the Pi is limited with the player we are using.

**NOTE:** The videos should *not* be encoded directly on the Pi Zero, unless you want to wait until the year 2050 before you can watch them. The Pi Zero is a single-core computer with minimal processing power. It's great for an endless number of things, but video encoding is not one of them.

There are of course a hundred ways we could do this. I've written a convenient script that will automatically encode all the videos in a folder to the proper format. First you need to install the video encoder FFmpeg onto your computer. There are plenty of guides out there for installing it; check out their GitHub page [github.com/adaptlearning/adapt\\_authoring/wiki/Installing-FFmpeg](https://github.com/adaptlearning/adapt_authoring/wiki/Installing-FFmpeg). Once you have FFmpeg installed, restart your computer and collect all your videos into a folder. Download my encoding script [github.com/buba447/simpsonstv/blob/main/videos/encode.py](https://github.com/buba447/simpsonstv/blob/main/videos/encode.py) and place it in the folder next to your videos.

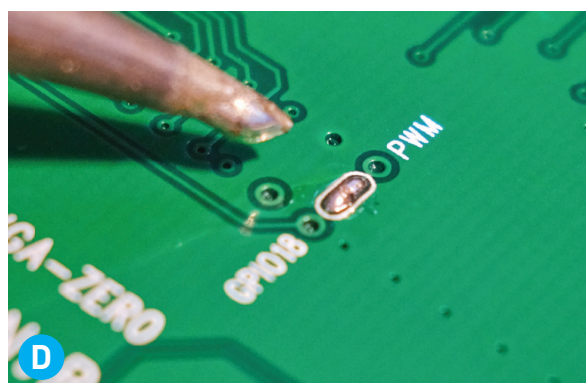
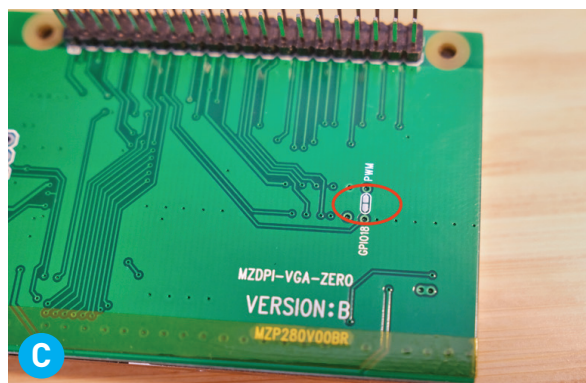
Open terminal, navigate to the folder with:

```
cd /path/to/the/folder
```

and run the script with:

```
sudo python encode.py
```

The script encodes all the videos one by one, and places them in a new subfolder called *encoded*.



Be patient! Do not close this terminal window, and keep your computer from sleeping while it runs. When it's done, go ahead and move the *encoded* folder over onto your thumb drive. I'll explain later how to move the videos onto the Pi.

## 3. HARDWARE SETUP

In order to have TV we need to have a screen, so let's start there. We must perform a little bit of surgery first. The screen has a function



to programmatically turn the backlight on and off. Unfortunately, that functionality is disabled by default. There's a jumper on the back of the screen that must be soldered to enable it.

We are going to make a quick solder. Plug in your soldering iron and warm it up. If you've never soldered before, don't fret — today you get to learn a new skill. I suggest getting a good set of helping hands; they really make it easier to solder small bits. Check out this beginner's guide to soldering ([makerspaces.com/how-to-solder](https://makerspaces.com/how-to-solder)) and by the end of this project you will be a pro.

Lay the screen facedown on your work surface with the pins facing north. You'll see the jumper off to the right (Figure C). You're going to solder this jumper, connecting the two pads. Place the iron against the pads to heat them up. After a second, apply the solder. Remember, always apply the solder to the pads, not the iron. The finished solder should look like Figure D.

The next bit is the hardest bit of soldering. If you are newer to soldering maybe skip ahead to Step 6 for practice and come back here afterward. Don't worry, you'll be a soldering star in no time.

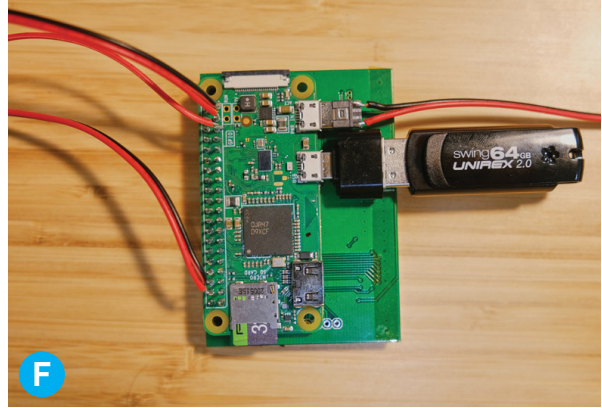
Place the Raspberry Pi Zero onto the 40-pin connector on the back of the LCD screen. It's important to keep the Raspberry Pi facing outward! I like to start by holding the corner of the Pi firmly in place and then quickly soldering the two leftmost pins (Figure E). I'm a southpaw like Stupid Sexy Flanders, so you might want to start with the two rightmost pins. Go slow and remember to heat the tiny pad and the pin with the iron, and apply the solder to the contacts, not the iron. Take breaks every few pins, you don't want to overheat the Pi board.

OK. Phew! Wipe the sweat off your forehead and get the shepherd's pie out of your knickers. You're a solder champion now.

#### 4. INSTALL THE SOFTWARE

Before we set up the rest of the hardware components, let's boot up the Pi and make sure everything works.

I've been updating the guide for this build as people run into issues with the software. So follow my latest instructions at [withrow.io/simpsons-tv-build-guide](https://withrow.io/simpsons-tv-build-guide) to prep your SD card, set up the Pi for headless boot, install the display



driver, and connect to the Pi from your computer using SSH. Then you'll run a few command-line instructions to install USBmount (it allows us to easily mount our USB drive when the time comes), cleverly route the audio over a spare GPIO pin, and install OMXPlayer (a lightweight video player).

Finally, install my script that will play videos on loop and also read the input from the front buttons. Run these two lines:

```
cd ~/
git clone https://github.com/buba447/simpsonstv
```

#### 5. MOVE THE VIDEOS ONTO YOUR PI

It's time to transfer the videos over! First, move all your videos to your USB thumb drive in a folder named *encoded*.

On the Pi, let's change directories:

```
cd ~/simpsonstv/videos
```

This is the directory where the video player will look for videos to play.

Plug your USB drive into the spare USB port on the Pi (Figure F); you'll need a Micro USB adapter for this. Once the USB drive is connected, copy all the videos from it to the Pi. Run the following command to copy the videos onto your Pi:

```
sudo cp -R /media/usb/encoded/. ~/simpsonstv/videos
```

This will probably take some time. Take a break, have a glass of water, go ponder the clouds.

OK now, final step! We want our video player and button control scripts to run every time the Pi boots. We will create a *service* for both scripts and set the Pi to run them whenever it boots. First, set up the button service:

```
sudo touch /etc/systemd/system/tvbutton.service
sudo nano /etc/systemd/system/tvbutton.service
```

Now paste the following into the editor:

```
[Unit]
Description=tvbutton
After=network.target

[Service]
WorkingDirectory=/home/pi/simpsonstv/
ExecStart=/usr/bin/python /home/pi/
simpsonstv/buttons.py
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

```
Next, set up the player service:
sudo touch /etc/systemd/system/
tvplayer.service
sudo nano /etc/systemd/system/tvplayer.
service
```

Paste the following into the editor and save:

```
[Unit]
Description=tvplayer
After=network.target

[Service]
WorkingDirectory=/home/pi/simpsonstv/
ExecStart=/usr/bin/python /home/pi/
simpsonstv/player.py
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

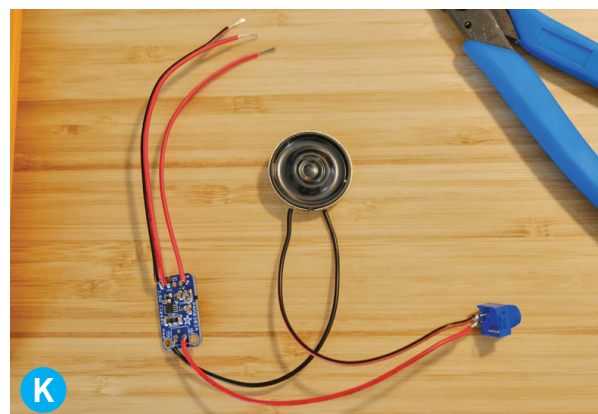
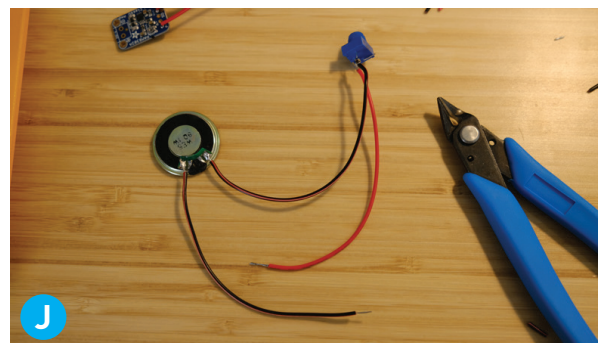
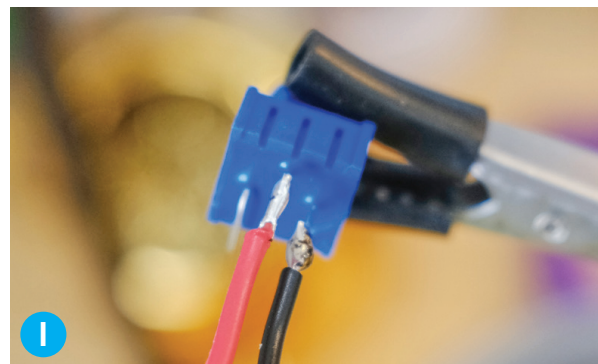
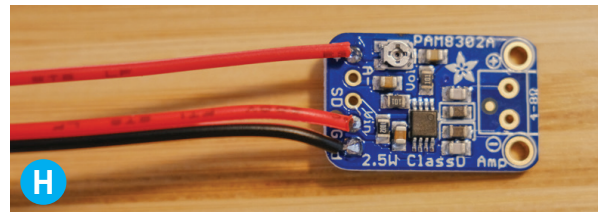
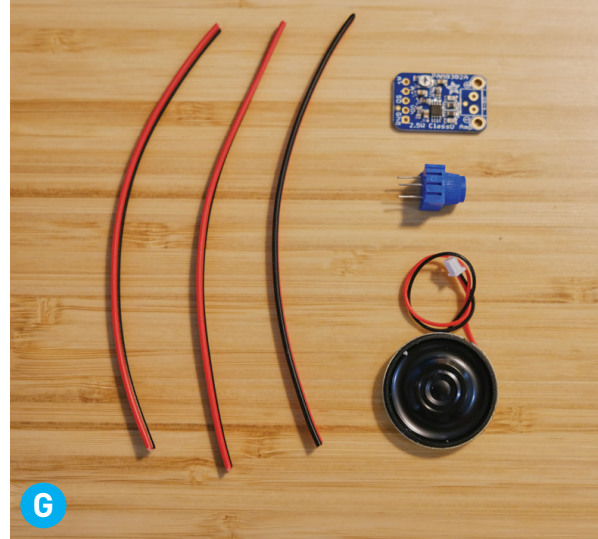
```
Now set these two services to start on boot:
sudo systemctl enable tvbutton.service
sudo systemctl enable tvplayer.service
```

```
Now shut down the Pi:
sudo shutdown -h now
```

Then unplug it from the power source. Phew, all this hacking has made me thirsty. Let's order a Tab. Take a break, and then let's finish up the hardware side.

## 6. SOLDER THE AUDIO CIRCUIT

Grab your trusty soldering iron and the rest of your components; you've got a little bit of





soldering left to do. It might be helpful to set this all up on a breadboard first, but if you are feeling embiggened by the noble spirit, go ahead and solder it up. I cut nearly all the wires to 4"–5" lengths. This makes final assembly a bit easier.

Let's start with the audio circuit. You'll need 3 wire pairs, the 1K potentiometer, the amplifier board, and the 4-ohm speaker (Figure **G**).

To start, let's solder up the outgoing wires on the amplifier. Take one black/red pair and solder the black to the ground and the red to the Vin pad next to the ground. These are the wires that supply the amplifier with power. Mark them if you'd like, or take a mental snapshot. Next, solder a single wire to the Audio In+ pad (also marked A+) which should be the outermost pad (Figure **H**). This is the wire that supplies the audio signal to the amp. We will connect these to the Pi later.

Now solder the speaker and potentiometer. A potentiometer, or *pot*, is a variable resistor with three legs. As the knob is turned, the resistance applied to the output is increased or decreased, which results in a lower or higher current. Essentially, it makes the signal weaker or stronger. This will control the volume going into the speaker. Solder a wire to the input (the center pin) and one to the output (the rightmost pin as shown in Figure **I**).

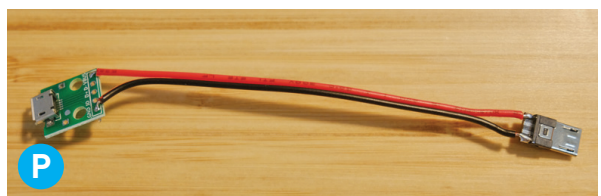
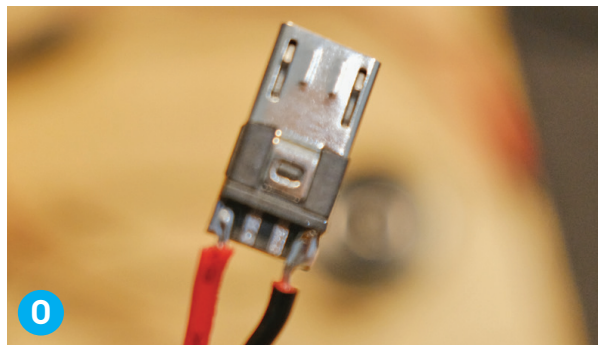
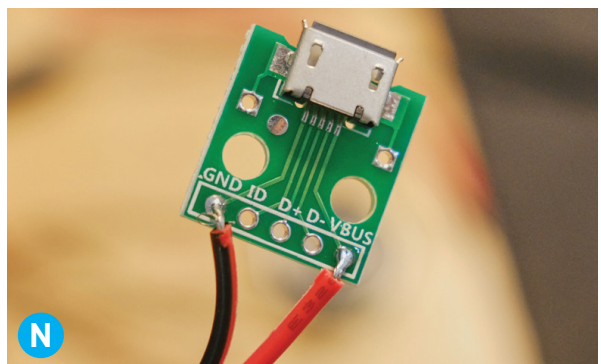
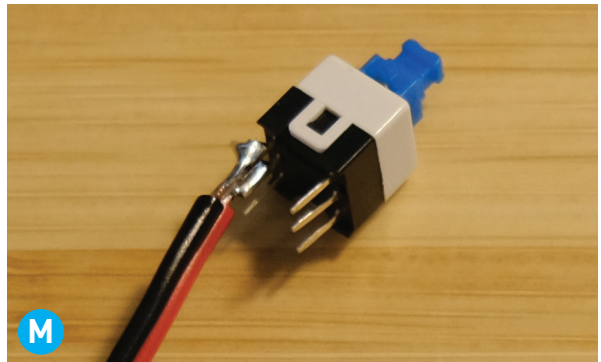
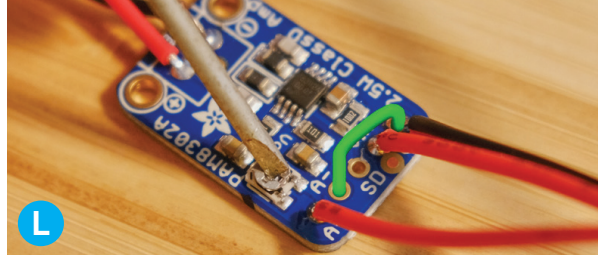
Solder a wire to one of the two speaker leads (doesn't matter which one). Next solder one of the wires coming from the pot to the other speaker lead (Figure **J**).

Connect the speaker circuit to the amplifier. There are two output pads on the opposite side of the amp from the input. Solder the loose speaker wire to one, and the loose wire from the pot to the other (Figure **K**).

Finally, solder a short jumper wire between the A– and GND pins of the amp (Figure **L**), and make sure the amplifier is turned all the way up. On the board you'll see a small silver knob with a Phillips screwdriver marking. Use a small screwdriver to turn the knob up (clockwise). Your speaker circuit is ready!

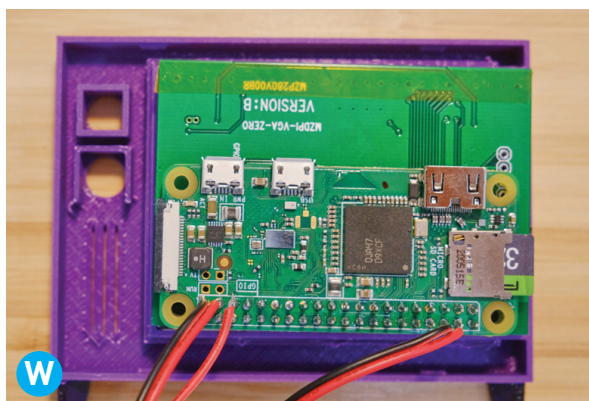
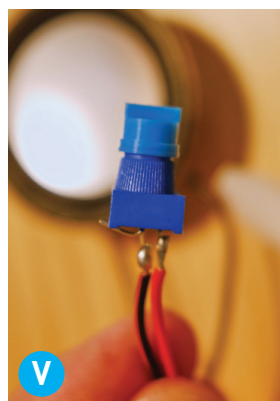
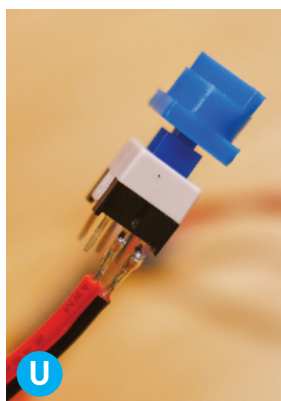
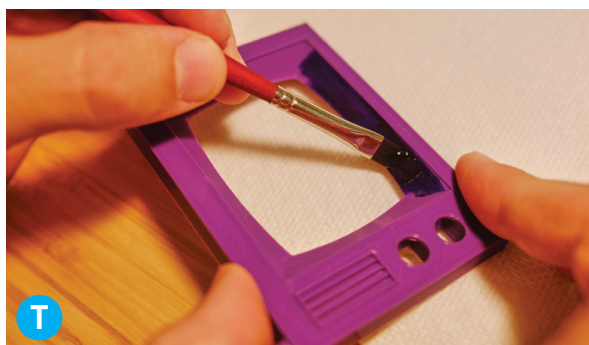
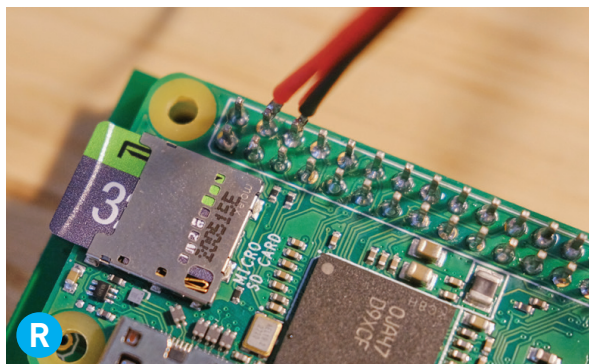
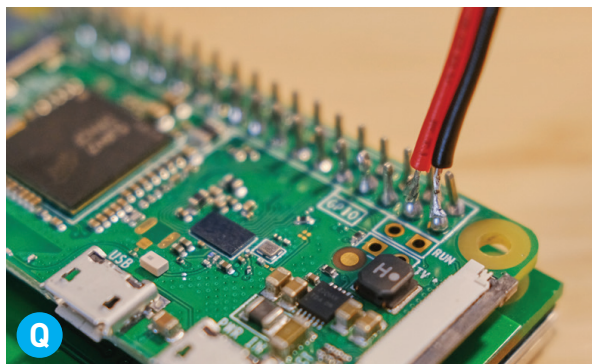
## 7. SOLDER THE POWER CIRCUIT

The power button is super easy to hook up. Solder a wire to the center pin, and another wire to the outermost right pin (Figure **M**).



The power cable is essentially a USB adapter cable. We need our Micro USB breakout board, a Micro USB header, and a short length of wire. Solder a red wire to the positive, and a black wire to the ground of the breakout board (Figure **N**).

Now solder the other end of the two wires — being careful to match the colors — to the USB header (Figures **O** and **P**).



## 8. CONNECT IT ALL TO THE PI

With the SD card facing north, solder the two wires of the power button to the Raspberry Pi Zero's GPIO pin 26 and the Ground pin that's below it (Figure **Q**). Check the pinout for your Pi online at [pinout.xyz](http://pinout.xyz).

Next, connect the audio circuit. Take the two power wires that you marked earlier and solder the red to the 5V+ pin and the black to the Ground pin in the upper right hand, next to GPIO 14 (Figure **R**).

Take the remaining wire and solder it to GPIO 19, which is next to the power button. OK, all the soldering should be done! Holster that bad boy, but do unplug it and wait for it to cool down first.

Let's make sure everything works. Plug in your Simpsons TV and wait for it to boot. The screen should stay off unless the screen power button

is engaged. Eventually some videos should start playing (Figure **S**)! Test that the power button toggles on and off and that the audio knob raises and lowers the volume. Phew! All the scary stuff is over. Time for the final stretch!

## 9. MOUNT COMPONENTS TO TV FRONT

Now let's put the housing together. If you want to go the extra mile (why would you come this far to not go a little further?), you can take some dark purple acrylic paint and paint a few details onto the case. Paint the legs and the inlay around the front of the screen (Figure **T**). It should be quick work; the case was designed for easy painting. If you mess up, use a wet paper towel to wipe off the paint. You'll probably want to do two coats.

Take the power knob, the one with the square



on the back, and push it down onto the power button (Figure U). There should be a bit of resistance and a satisfying snap.

Next take the volume knob and apply a bit of super glue to the back. Press the front of the pot into the knob. Hold it. Hold it. Hold it. There we go (Figure V).

Now you'll mount the components into the front of the TV. Be sure to remove the anti-scratch sticker from the front of the screen. Mmmmmhmmm, pleasing. Also remove the power extension cable from the Pi. Place the screen facedown onto the front housing. Make sure the screen is oriented properly, with the Pi on the south side (Figure W).

Plug in your trusty hot glue gun. First dab the top left corner of the screen (Figure X). Cover it with glue from the top edge down to the housing. Next run a bead of hot glue along the top and bottom edges of the screen (Figure Y). Be careful to not entirely fill the void in the case at the top or bottom where the casing snaps go.

Now place the power button in the top knob housing. First make sure it's disengaged: press the button and release until it's at its tallest position. Gently press it into the housing until the knob comes out the front (Figure Z). Be careful to not press the button itself in, or it will never be able to disengage. Use some hot glue to secure the button housing in place. Hold in place until the hot glue sets.

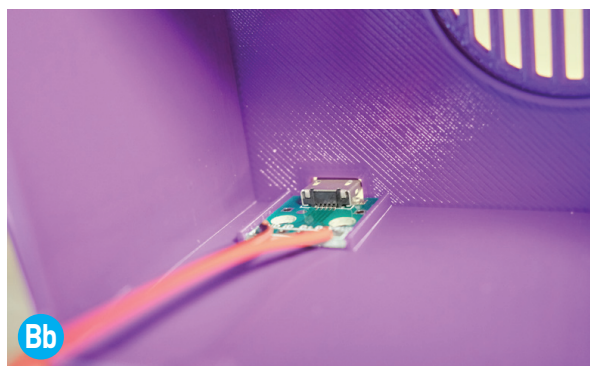
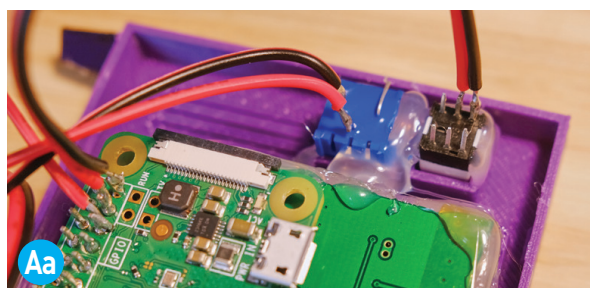
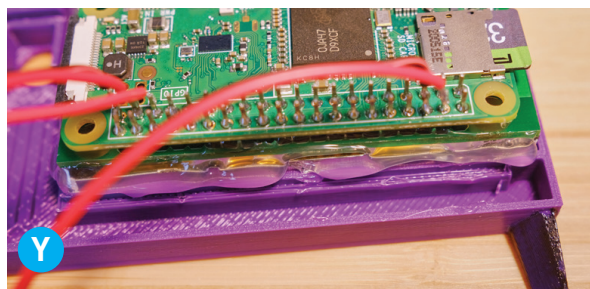
Follow the same procedure for the volume knob (Figure Aa). Don't press it in too firmly, or the knob won't turn.

## 10. MOUNT COMPONENTS IN TV HOUSING

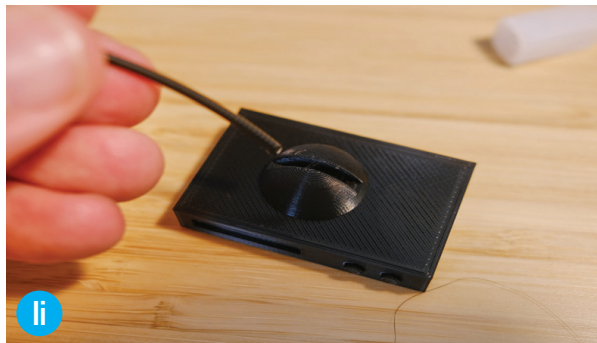
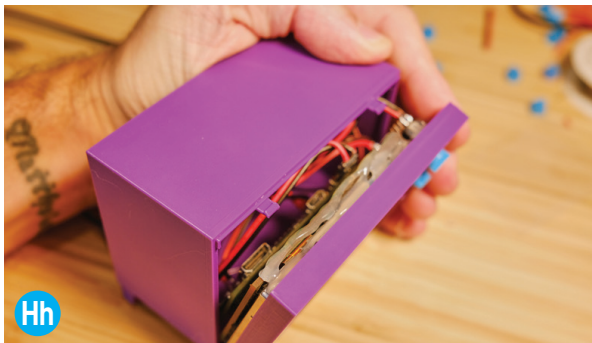
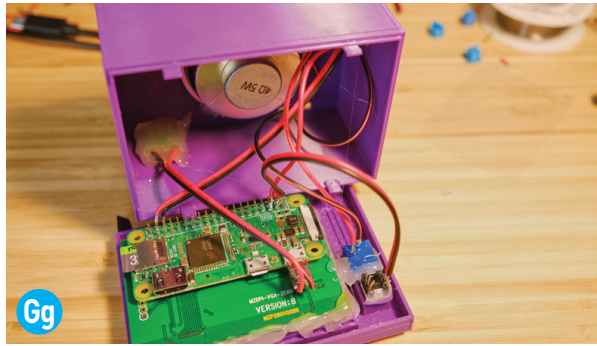
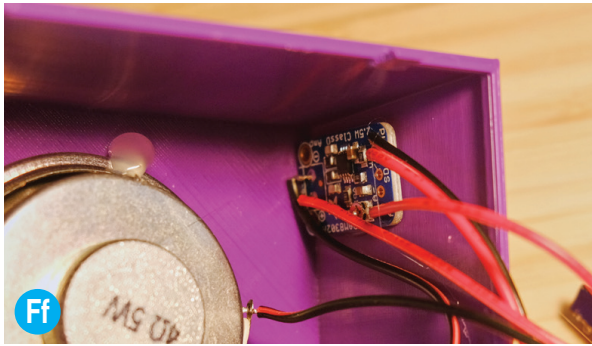
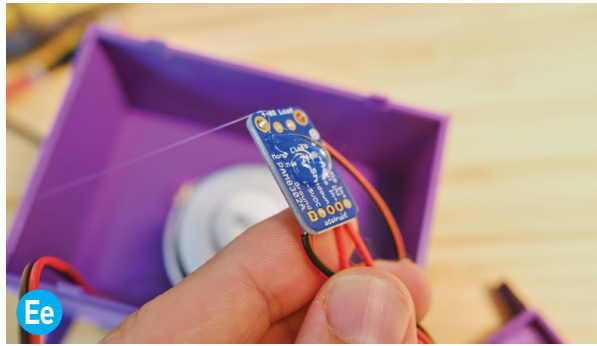
OK, home stretch! Grab your USB power extension cable. Place the Micro USB board on its seat in the bottom left corner of the large TV housing (Figure Bb). I find this next step is easier if you plug a Micro USB cable into it after placing it (leave everything unplugged from any power source). The cable helps to hold it in place.

Use a bit of hot glue to tack it into place. Then, generously cover the whole thing, connecting it to the housing. Wait for the glue to cool and turn white (Figure Cc) before moving on.

Place the speaker facedown on the back ring.









Use hot glue to secure it around the edges. You don't need to glue the whole thing, just a few points. Again, wait for the glue to set and check that it's secure before continuing (Figure Dd).

Finally, the amplifier. Put just a dab of glue on the underside (Figure Ee) and press it into the side wall of the enclosure (Figure Ff). Easy peasy. OK! Great work. You're done with the hot glue gun, so holster it next to your soldering iron. Check everything and make sure it's secure. Remove any hot glue spider webs that you find.

One more thing before we close it up: connect the power extension cable into the Pi's USB power port (Figure Gg).

Time to close it up! The front of the TV snaps onto the housing using four case snaps. Align the two pieces, and carefully fold any wires into the enclosure. Starting at the bottom, press and snap the front onto the case (Figure Hh). Nice!

Time to assemble the VCR. Cut two short lengths of black printer filament for the antenna. Using a bit of super glue, press them into the two antenna holes on top of the VCR (Figure Ii). Then glue on the two antenna toppers (Figure Jj).

Put a bit of super glue on the bottom and place the VCR on top of the TV (Figure Kk). Don't worry about being too precise; do you think Homer had his VCR perfectly centered?

Et voilà! You have a tiny TV.

Congrats on making it this far — you are awesome! Tweet me a pic of your finished TV @theWithra; I'd love to see it.

## IT'S KRUSTY TIME!

When the Pi is powered up through the USB port on the back of the TV, it starts playing episodes at random. When one episode ends, the next is randomly selected. The top button turns the screen on and off, while also muting the volume. The bottom knob turns the volume up and down.

After this project went viral on Reddit and Twitter this summer, I decided to make 10 TVs and raffle them off — with 100% of the proceeds going to four awesome charities: Living Lands and Waters, To Write Love on Her Arms, American Modeling Teachers Association, and the Highlands Museum & Discovery Center in my hometown of Ashland, Kentucky.

## LEVEL UP

But wait! There's more. Here are a few ideas for upgrading your Simpsons TV.

- **Faster boot time** — Yeah, I hate waiting for my Simpsons TV to boot too. The first version was built on PipaOS ([pipaos.mitako.eu](http://pipaos.mitako.eu)), a super lightweight version of Raspbian Jessie that booted almost instantly. It was fabulous. Unfortunately, as I was writing this guide, Jessie was discontinued and is no longer supported. There is a newer version of PipaOS built on Stretch, but I haven't tried it yet. You should be able to follow this guide with PipaOS, but I haven't tested it. You might run into some errors.
- **Gapless playback** — This is a fun one! I've managed to get near gapless playback working with OMXPlayer by running two instances of OMXPlayer and controlling them using DBUS. This is a little advanced and takes some finessing. The other catch is that you need to know the runtime of each video, which can be found with Fprobe. This adds another layer of complexity to the whole thing.
- **TV on-off graphics** — I was able to add TV graphics such as interstitials, on-off blinks, and even a little logo in the bottom corner using Kivy. Kivy is a free, lightweight graphics package for Python. It must be compiled for your OS of choice, which is a more advanced topic. I find that something like Kivy is best for the Pi Zero, since it is not powerful enough to play video and also render out HTML. If you decide to go down this rabbit hole there are a lot of interesting quirks to get OMXPlayer and Kivy to play together. I'd be happy to write a guide sometime if there's enough interest.

## BE COOL

Please, please do not build these and sell them, and definitely do not sell them with copyrighted material on them! ☹




Check for the latest version of this guide at [withrow.io/simpsons-tv-build-guide](http://withrow.io/simpsons-tv-build-guide). And if you find this guide helpful and want to buy me a coffee, you can donate at [makezine.com/go/withra-paypal](http://makezine.com/go/withra-paypal).



# Neverending Quotes

Get an everlasting stream of wisdom  
from Reddit, by glancing at a little  
e-paper box Written by Vanessa Bradley



If at first you  
don't succeed, find  
out if the loser  
gets anything.

—  
Bill Lyon



**VANESSA BRADLEY** is a reluctant coder, efficiently learning and forgetting languages every time she wants to build something. She is a Brit based in Switzerland.

We're going to show you how to make a little box of almost infinite wisdom — a cute display that scrapes highly-rated quotes from Reddit and displays them for you. We call it Neverending Quotes.

The finished project is a little e-paper ticker that sits on your desk, periodically pops on the internet, and brings you back a quote that people have voted as interesting. It lets you enjoy a neverending stream of wisdom at a glance, without going on the internet and drinking from the information firehose, falling down a rabbit hole, or being “nerd sniped” (XKCD joke).

It's an easy build that's suitable as a first project. It's also a nice way to get to grips with GitHub if you haven't used it before.

## REAL INTELLIGENCE

If you ask somebody what AI is, they'll tell you it's artificial intelligence. But if you ask somebody what intelligence is, you'll probably get a less convincing answer. We're not sure what intelligence is, but we are sure that people have it, so ideally we would avoid AI and just stick to regular I — real human intelligence.

Reddit uses a voting system called “karma”: someone submits a post and if people like it, or dislike it, they vote it up or down. It's a really simple mechanism, but behind every vote, there is a human brain. A “subreddit” like [r/quotes](#) is the brilliance of Reddit — it's a neverending stream of interesting quotes that's supplied and curated by people. The quotes are better than any database you can find, in that they're limitless, they come from the internet and beyond, and they're curated by thousands of human brains, not an algorithm.

The Neverending Quotes box is essentially just an e-paper screen attached to a Raspberry Pi Zero mini computer, and uses our open source Python code. The code picks a highly rated quote from [r/quotes](#), tidies it up a bit, and then displays it on the screen, giving your brain a little wisdom boost. And another, and another. You can buy one on our website ([veeb.ch](#)), or build one yourself. Here's how.

## 1. CONNECT THE HARDWARE

Plug your Waveshare e-paper “hat” into the

**TIME REQUIRED:** 2 Hours

**DIFFICULTY:** Easy

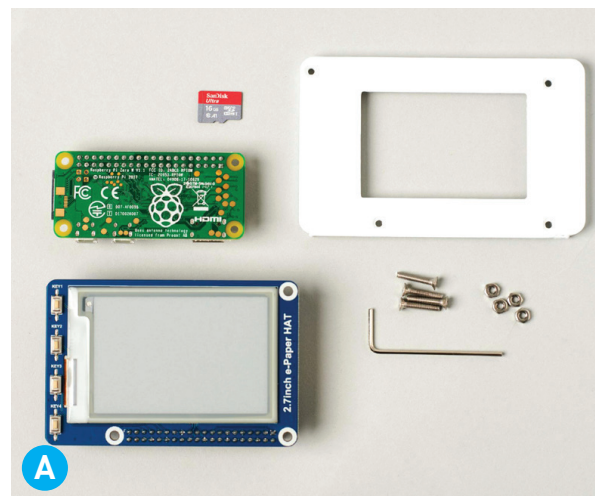
**COST:** \$50–\$90

## MATERIALS

- » **Raspberry Pi Zero WH mini computer** the version with headers
- » **SD Card, 8GB or more**
- » **Waveshare 2.7" e-paper display (EPD) Hat for Raspberry Pi** black and white, #13354, [waveshare.com](#)
- » **3D-printed frame** Download the free files for printing at [github.com/veebch/btcticker/blob/main/frames/Frame%20v6.stl](#), or buy a frame at [veeb.ch/store/p/ticker-enclosure](#).
- » **Machine screws, flat head socket cap, M2.5 size (4)** with nuts

## TOOLS

- » **Hex key** for screws
- » **3D printer (optional)**



Martin Spendiff

headers on the Raspberry Pi Zero (Figure A). Then plug the Pi into its power supply. That's it!

## 2. SET UP THE RASPBERRY PI

These instructions assume that your Raspberry Pi is already connected to the internet, happily running **pip**, and has **python3** installed. If you've never set up a Pi before, follow the Getting Started guide at [projects.raspberrypi.org](#).

If you are running the Pi headless, connect to your Raspberry Pi using SSH.

First, enable the SPI interface (**0**=on, **1**=off) by using the command:

```
sudo raspi-config nonint do_spi 0
```



### 3. INSTALL THE PROJECT CODE

Our Python script, **edify.py**, displays randomly chosen, highly rated quotes from Reddit ([r/quotes](https://www.reddit.com/r/quotes)). It can also be configured to display a word of the day, or an item from a flashcard text file (selected at random, according to weightings in the config file *config.yaml*).

First, clone Waveshare's e-paper libraries and our **edify** script:

```
cd ~
git clone https://github.com/waveshare/e-Paper
git clone https://github.com/veebch/edify.git
```

Move to the **edify** directory, copy the example config to *config.yaml*, and move the required Waveshare library over to the **edify** directory:

```
cd edify
cp config_example.yaml config.yaml
cp -r /home/pi/e-Paper/RaspberryPi_JetsonNano/python/lib/waveshare_epd .
rm -rf /home/pi/e-Paper
```

Install required Python3 modules using **pip**:

```
python3 -m pip install -r requirements.txt
```

If you'd like the script to persist once you close the session, use **screen**. Start a screen session:

```
screen bash
```

Run the script using:

```
python3 edify.py
```

Detach from the screen session using Ctrl-A followed by Ctrl-D. The unit will now pull data every 60 minutes (or whatever is specified in the configuration file) and update the display.

### 4. CONFIGURE (OPTIONAL)

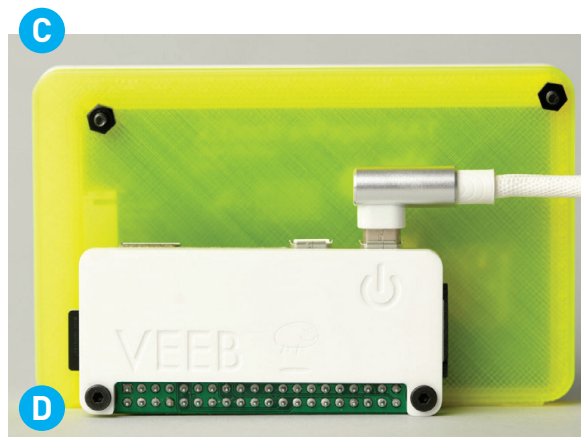
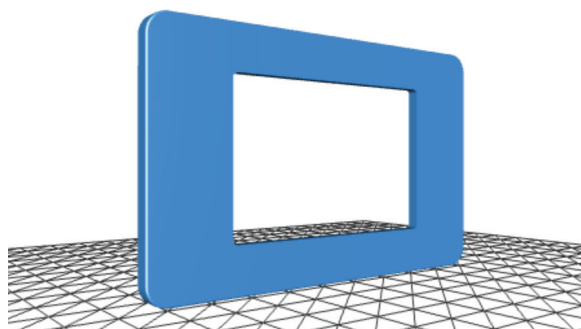
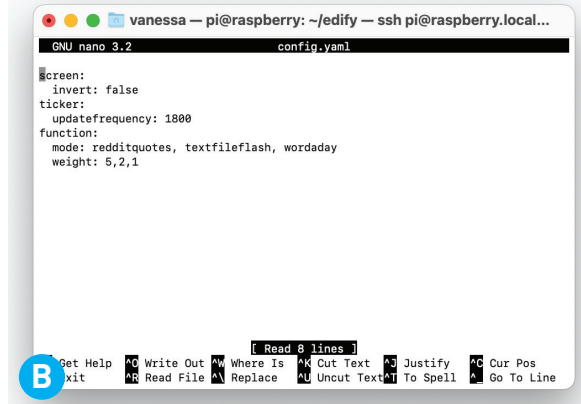
To adjust the refresh time (in seconds) you can edit the update frequency (Figure B) using:

```
nano config.yaml
```

### 5. ADD AUTOSTART

If you'd like the script to start automatically every time it is plugged in:

```
cat <<EOF | sudo tee /etc/systemd/
```



Martin Spendiff

```
system/edify.service
[Unit]
Description=edify
After=network.target
```

```
[Service]
ExecStart=/usr/bin/python3 -u /home/pi/edify/edify.py
WorkingDirectory=/home/pi/edify/
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi
```

```
[Install]
WantedBy=multi-user.target
EOF
```

Now enable the service you just made and reboot:



E



F

```
sudo systemctl enable edify.service
sudo systemctl start edify.service
sudo reboot
```

## 6. PUT IT IN A FRAME

If you have access to a 3D printer, our STL files for a front panel (Figure C) can be found at [github.com/veebch/btcticker/blob/main/frames/Frame%20v6.stl](https://github.com/veebch/btcticker/blob/main/frames/Frame%20v6.stl).

If you want to buy a frame, which includes a full enclosure, screws and a hex key, and an optional laser-cut acrylic backplate for the Pi (Figure D), check out [veeb.ch/store/p/ticker-enclosure](https://veeb.ch/store/p/ticker-enclosure).

## THE WISDOM MUST FLOW

Done! Your Pi currently scrapes quotes from Reddit (Figure E and F), or grabs something from a text file you can designate, or scrapes a “word of the day.”

The Python code could easily be adapted to scrape from other sources. We use this same setup for our Cryptocurrency Ticker (Figure G), which you can find at [veeb.ch/store/p/cryptocurrency-ticker](https://veeb.ch/store/p/cryptocurrency-ticker) and [github.com/veebch/btcticker](https://github.com/veebch/btcticker). Use your imagination! 🚀



G





# Bright Lights, Big TV

Written and photographed by Mike Senese

Elevate your home entertainment setup with this Pi-powered ambient television backlight system



**MIKE SENESE** is the former executive editor of *Make*:. A lifelong tinkerer and tech writer, he lives in the San Francisco Bay Area with his wife and son. Find him on the web: [mikesenese.com](http://mikesenese.com)



Televisions these days are technological marvels — absolutely massive yet paper-thin screens with crystal-clear 4K (or better) resolution, up to 120Hz refresh rates, high dynamic range (HDR) color and luminance, and other features that turn living rooms into amazing personal theaters. As great as these new TVs are though, the magic still ends at the thin bezel that frames the screen. But what if you could make your shows, movies, or games stretch beyond the television itself?

With a Raspberry Pi, a few pieces of HDMI hardware, a strip of assignable LED lights, and free software, you can do just that. The hardware processes your video content, detecting the specific colors that are closest to all edges of the screen and then driving the LED lights to correspondingly project the same colors outward behind the television. It all happens dynamically and in real time, creating an immersive lighting effect that makes the content feel even larger and more exciting.

This trick is known as *ambient lighting* or *bias lighting*. You can find a couple all-in-one ambient lighting systems that offer this effect from companies like Philips, but they're expensive and proprietary. Fortunately, it's not hard to put your own together, nor is it particularly expensive. Ready to get radiant? Here's how.

## 1. CONFIGURE THE HARDWARE

To get the LED backlights to display hues corresponding to what's shown on the screen, we'll be splitting the video signal from an external HDMI device (Apple TV, Chromecast, Roku, etc.) into two signals. One of those passes through to the television screen where it will be displayed normally, while the other gets routed via USB to the Raspberry Pi, where the HyperHDR software processes the signal down to just the video elements on the outer limits of the TV screen, then sends that signal to the LED light strip (Figure A).

## 2. PREPARE THE LED ELECTRONICS

There are a wide range of LED strip lights on the market and HyperHDR can handle most of them. It can even use external Philips Hue lights remotely, via the Hue Bridge — allowing you to

**TIME REQUIRED:** 2 Hours

**DIFFICULTY:** Moderate

**COST:** \$150

## MATERIALS

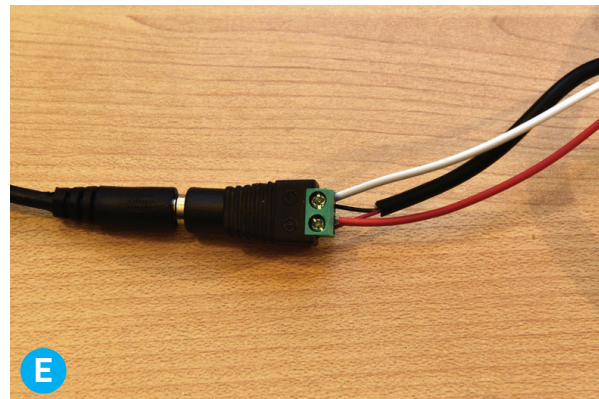
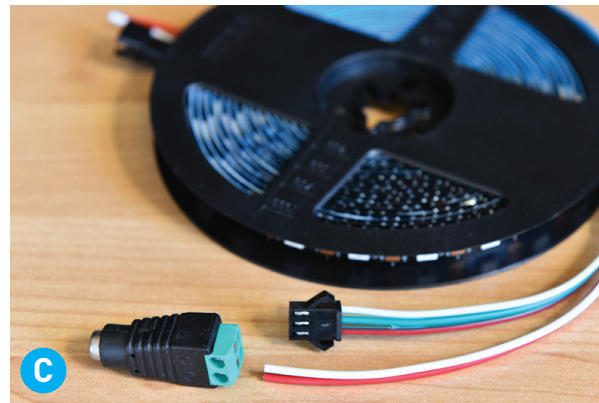
- » **Raspberry Pi single-board computer** Pi 3 or 4 recommended, but this will work with even the Pi Zero.
- » **MicroSD card, at least 8GB capacity** and SD card adapter
- » **LED strip, WS2812B, 5 meters**
- » **Streaming device, HDMI** e.g. Roku or Chromecast
- » **Power supply, 5V, 10A**
- » **Barrel jack power adapter, female, 5V**
- » **HDMI to USB capture card, 4K** aka USB video grabber. I used a generic non-HDR 4K USB 3.0 card with HDMI pass-through (\$30 on Amazon) for a basic setup, but to maintain your HDR signal and preserve Atmos audio, go with a separate, suitable HDMI splitter (e.g. Ezcoo SP12HAS, about \$65) and one-way HDMI-USB capture card (\$20). If you want CEC too, you'll need the SP12H28S instead (\$140).
- » **HDMI cable, 2.0 or 2.1** Use 2.1 for HDR.
- » **USB cable, A-A** to connect grabber to Pi
- » **USB cable, Micro or C** to match the power input of your Raspberry Pi
- » **Jumper wires, M-F (1 or 2)**
- » **Tape, double-sided foam (optional)** if your LED strips don't have adhesive backing. Or get creative with strip-light channeling and 3D-printed brackets.

## TOOLS

- » **Wire stripper**
- » **Screwdriver, precision**
- » **Computer** with Wi-Fi and internet access
- » **Keyboard and mouse, USB**







place additional lights alongside the television for even wider ambient light dispersion. Turn your whole room into the viewing experience!

For my build, I used WS2812B string lights. They're cheap and easy to find, and they run off 5V, so you can even build a power circuit that powers these and the Raspberry Pi with just one plug. I got a 5-meter (16.4-foot) strip with 60 pixels per meter. More pixels equals more resolution, right? Well, yes, although I'm not convinced that the improvement over the 30 pixels/meter option is worth the extra wattage needed to power them. (There are also advantages to using 12V strip lights, such as the WS2815 version — these don't suffer from voltage drops across longer lengths, as the 5V ones can. But they don't offer the easy option to power the Pi from the same power source.)

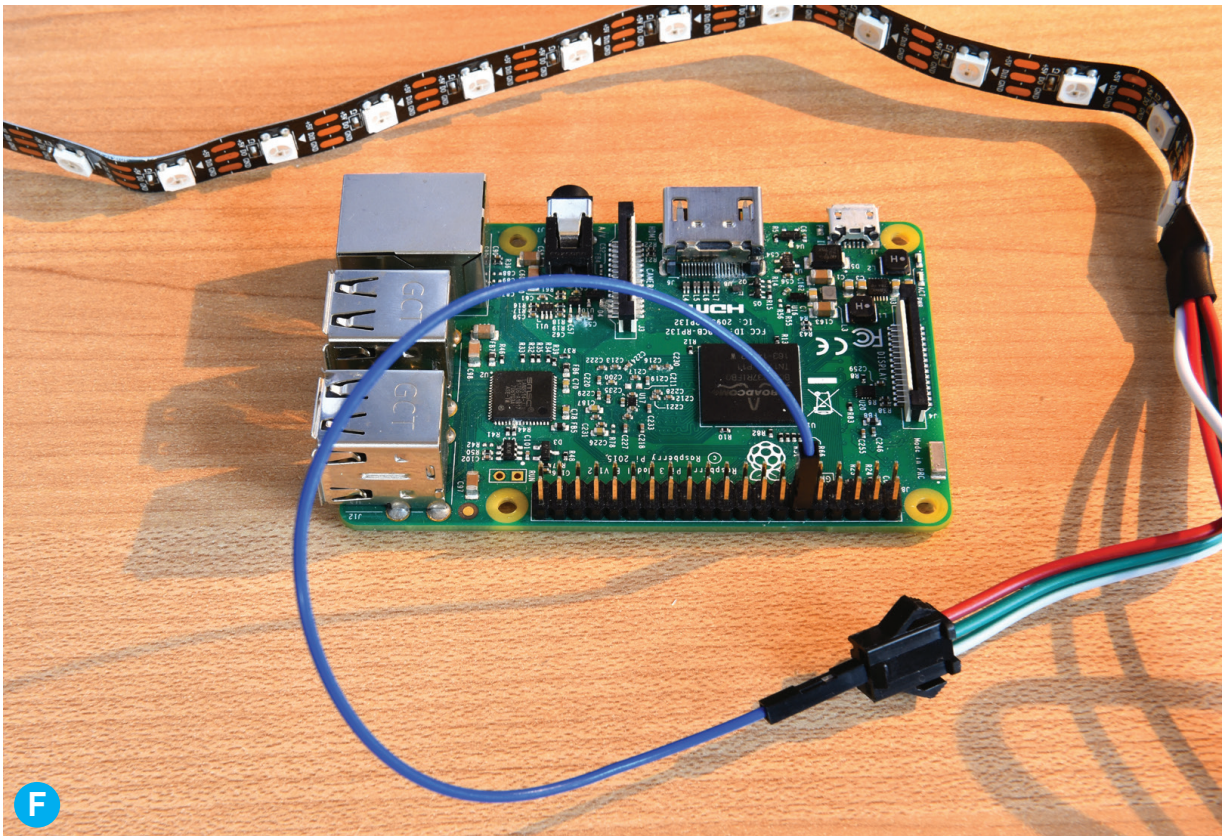
Speaking of power sources, you'll need one that is sufficient for lighting all those LEDs. We're talking 150–300 LEDs, potentially blasting a full, pure white light, meaning they'll be drawing maximum amperage. A standard phone charger won't cut it for that; you'll want to get a laptop-style power brick (Figure B) that matches the voltage of your string lights and has a fair amount

of amperage. Mine does 5V 10A, for 50 watts of power, but if I were wiser I'd get a bigger power supply that can go up to 300W, just for safety. They're not much more expensive, either.

To connect the LED light strip to the power source, use a 5V barrel jack adapter. Some power supplies include this, but you may have to buy one. They usually come in a small set; be sure you get the "female" side with it. Strip a small bit of insulation off the power leads from the LED lights, then connect these to the screw terminals of the female jack. The red wire goes to positive (+), the black or white wire goes to negative (-) (Figures C and D).

**NOTE:** If you're using 5V lights and want to couple in power for your Raspberry Pi, here's where you'll do it. Cut off the big end of a Micro-USB or USB-C cable (depending on which model Pi you have), find the red and black wires, strip off about 1/8" of insulation, and insert them in the corresponding + and - slots along with the wires that power the LED strip (Figure E). This will let you power the Pi from the same power supply.





For longer lengths (300 LEDs), also feed the trailing red and white wires into the barrel plug here as well, so the voltage and ground feed both the start and end of the lights.

Finally, use a M-F jumper cable to connect the Data line from the LED strip (it's the middle pin on most strips) to pin 18 on the Pi (it's the sixth pin from the bottom corner) as shown in Figure **F**.

**IMPORTANT:** If you decide to power the Pi separately, you'll also need to put a jumper between one of the GND pins and the ground wire on the LED strip, otherwise it won't work.

If you'd like, you can now plug the devices together. Connect your external HDMI streaming device into the HDMI capture card, an HDMI cable from there into the TV, and also a USB-A cable into a USB port of the Raspberry Pi. My Pi's USB jack supplies power to the capture card; look for the red light on it to know that it's working.

Then turn on the TV and your HDMI streaming device, and select the HDMI input that you're plugged into on the TV. If everything has power, you should now see your normal content on the TV screen.



### 3. ASSEMBLE AND MOUNT THE LEDs

For best results you're going to want a rectangle of LED lights around the back edge of your screen, close to the sides. Some people just affix the light strip with its built-in adhesive, folding the strip at the corners. Others build a dedicated frame to hold the lights at an angle, with pre-made corner connectors (Figure **G**). This is up to your personal preference.

Whatever you do, count the LED lights on each





side and mark them down. You'll need these numbers for the HyperHDR configuration.

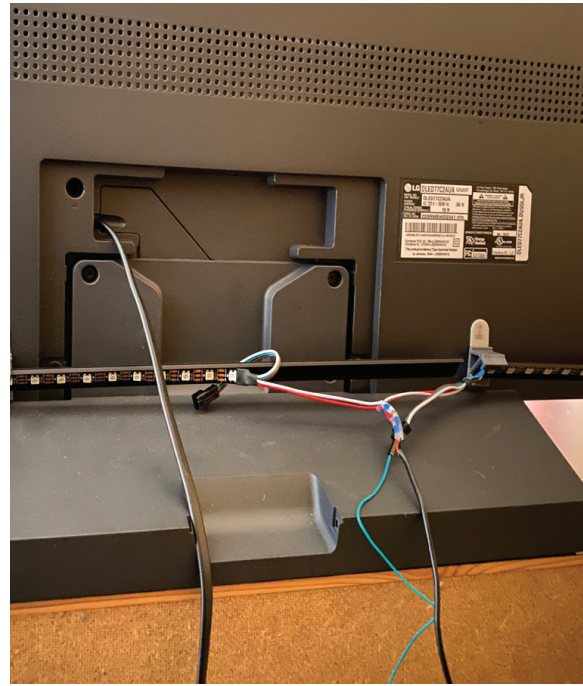
You'll also want to decide where the input for the light strip should start. Top corner? Bottom center (Figure H)? This is software-configurable as well. The default direction of the lights is clockwise (when looking at the screen from the front) but you can reverse that in the software if necessary. You can also input a gap setting in the software, measured in LED pixels, for instance if you leave a space in the light strip for the television stand. Mark all these numbers down, you'll need them in a moment.

Finally, mount or stash the HDMI hardware somewhere out of sight but convenient.

#### 4. DOWNLOAD AND INSTALL SOFTWARE

For this project, we'll use HyperHDR, a variation of the popular open source Hyperion software. It allows you to match the vibrance of the LED lights with what's displayed on your HDR-capable TV screen, without needing an expensive HDMI splitter that transmits the HDR signal on both split channels. Most HDMI splitters and capture cards downgrade the split-off signal to SDR, which results in less-rich color transmission on the surrounding lights. HyperHDR fixes that.

Download the latest version from [github.com/awawa-dev/HyperHDR/releases](https://github.com/awawa-dev/HyperHDR/releases). I'm running the



64-bit beta version of 19.0.0.0 on a Raspberry Pi 3. Look for *SD-card-image-19.0.0.0beta1-aarch64.img.xz* on the releases page, which also works on the Pi 4, Pi Zero 2 W, and rev 1.2 of the Pi 2. (Check out [github.com/awawa-dev/HyperHDR/wiki/Installation](https://github.com/awawa-dev/HyperHDR/wiki/Installation) for releases supporting other Pi models.)

Once the file is saved on your computer, use a card imaging app to build a bootable SD card from it. I like Etcher ([balena.io/etcher](https://balena.io/etcher)) but other options like Rufus ([rufus.ie/en](https://rufus.ie/en)) work great too. You can also use the Raspberry Pi Imager tool ([raspberrypi.com/software](https://raspberrypi.com/software)), which is a pretty slick method of creating many different types of images with the settings you want. Use a card with at least 8GB capacity.

Once the image is built you'll need to add a file to the card to give it your Wi-Fi credentials, which will allow you to log into its UI and configure the settings. To do this, open a text editor and input the following, with your own network details:

```
ctrl_interface=DIR=/var/run/wpa_
supplicant GROUP=netdev
update_config=1
country=PL
network={
  ssid="Name of your WiFi LAN"
  psk="Password for your WiFi LAN"
}
```

Save that text file with the name `wpa_supplicant.conf` and drag it onto the `/boot` folder of your card while it's still in your laptop.

Now you can boot up the Pi — insert the SD card in your Pi, plug it into the TV you'll be using with an HDMI cable, and power up the Pi. You'll see the familiar Raspberry Pi bootup screen. After a couple minutes of processing, it will ask you to log in (user **pi**, password **raspberrypi**), then change the default password and mark it down. I use a USB keyboard and mouse to input these directly into the Pi on-screen; you can also SSH into the Pi from a separate computer, but since this project already involves a TV screen, it's just as easy to input the info directly.

Next, find the IP address of the Raspberry Pi by running the `ifconfig` command in the terminal on the Raspberry Pi. Write it down, you'll want it to SSH into the Pi. You'll find it as the `inet` number at the start of the `wlan0` section, probably something like `192.168.xx.xx`. Alternatively, I use the Fing app ([fing.com](https://fing.com)) on a spare Android phone to find my Pi's IP address remotely; I find this is easier.

Now you can remove the USB peripherals and disconnect the Pi from the HDMI connection. From here you should be fine doing everything else via SSH and the HyperHDR user interface.

## 5. CHANGE USER TO ROOT

If you're using WS2812B LEDs, it's likely that you'll be using PWM to control them, via pin 18 (the default PWM pin for HyperHDR and many other Raspberry Pi LED projects).

However, one thing that isn't always communicated in ambient backlight project write-ups is that the Pi's PWM function can only be accessed and controlled if the Raspberry Pi is running in "root" mode. It's quite likely your

installation won't initially be in root; generally it is discouraged as it allows direct access to critical files. However, the LED lights won't turn on without it! So next, we'll enable root access on the Pi with the following steps:

- Open the Terminal tool.
- SSH into the Pi using that IP address you wrote down from the previous step, by typing:  
**ssh pi@<IP address of Raspberry Pi>**
- Input your password.
- Verify that your Pi is not already running as root. In the terminal, type:  
**sudo systemctl status hyperhdr@pi.service**

If, inside the results that pop up, you see **Active: active (running) since...** you are running as **pi** and not as **root**. Press Control-C on your keyboard to exit the results and go back to the command prompt.

- Now disable the **pi** user:  
**sudo systemctl disable --now hyperhdr@pi.service**
- Then enable the **root** user:  
**sudo systemctl enable --now hyperhdr@root.service**
- You can now verify that **root** is active by using that same status command:  
**sudo systemctl status hyperhdr@root.service**

If it says **active** you are good to go (Figure 1).

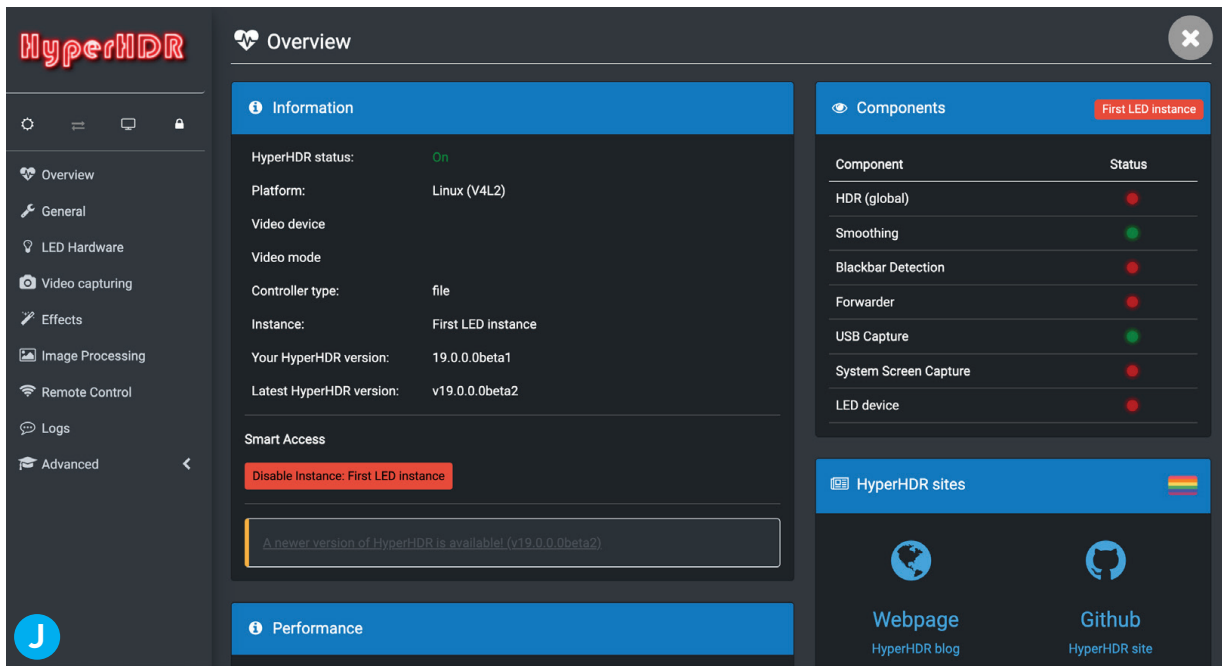
## 6. CONFIGURE THE UI

Now go to a separate computer, open a browser, and type in `http://hyperhdr:8090` to get to the HyperHDR user interface (Figure 2 on following page). You may get a prompt to choose a password here.

```
[pi@hyperhdr:~ $ sudo systemctl status hyperhdr@root.service
● hyperhdr@root.service - HyperHdr ambient light systemd service for user root
   Loaded: loaded (/etc/systemd/system/hyperhdr@.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-12-19 01:50:10 GMT; 27s ago
     Main PID: 2454 (hyperhdr)
        Tasks: 11 (limit: 779)
             CPU: 6.931s
    CGroup: /system.slice/system-hyperhdr.slice/hyperhdr@root.service
            └─2454 /usr/bin/hyperhdr
```







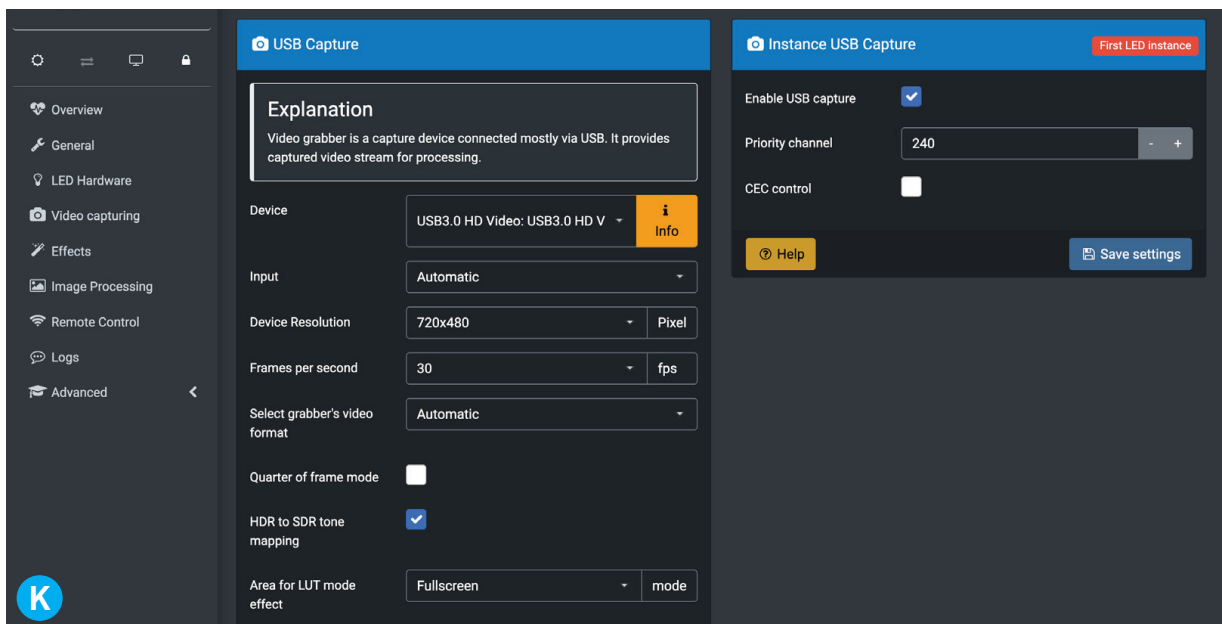
There are a ton of setting configurations in the software, and you'll need to explore those yourself for best performance with your own setup, but the main ones to jump to first to get things working are:

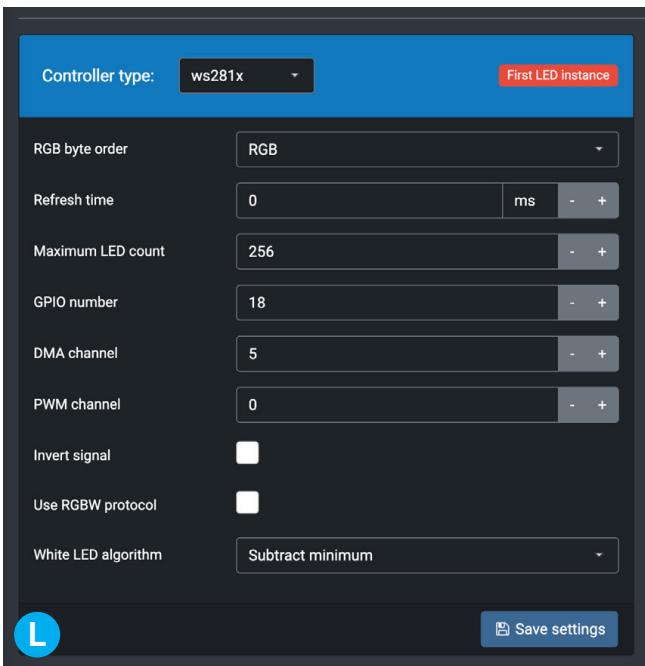
- **VIDEO CAPTURING:** Your HDMI grabber should automatically show up here, under the Device setting (Figure K). You can keep the resolution and frame rate on Automatic, but to keep things zippy, reduce the settings down to even 640x480 and 20FPS — we're not dealing with fine pixels. You can also activate HDR to SDR

tone mapping here. Click "Save settings."

On that same screen, in the Instance USB Capture section, click the checkbox to Enable USB capture, then click "Save settings."

- **LED HARDWARE:** Under the LED Controller tab, choose "ws281x" from the "Controller type" dropdown menu (Figure L). You may need to change your RGB order (there's a wizard for this under the Advanced menu). Input your total number of LEDs. Click "Save settings." You might now see some of your LEDs starting to light up at this point.





## MORE HELPFUL LINKS AND RESOURCES:

- HyperHDR Github repo: [github.com/awawa-dev/HyperHDR](https://github.com/awawa-dev/HyperHDR)
- Raspberry Pi's HyperHDR tutorial: [raspberrypi.com/tutorials/raspberry-pi-tv-ambient-lighting](https://raspberrypi.com/tutorials/raspberry-pi-tv-ambient-lighting)
- This Smart House video tutorial for Hyperion: [youtu.be/PY8\\_KYnxyul](https://youtu.be/PY8_KYnxyul)

Click the LED Layout tab next. Here's where you'll input those LED counts, gap length, and input position. Once you click Save settings, your full LED array should fill up with light!

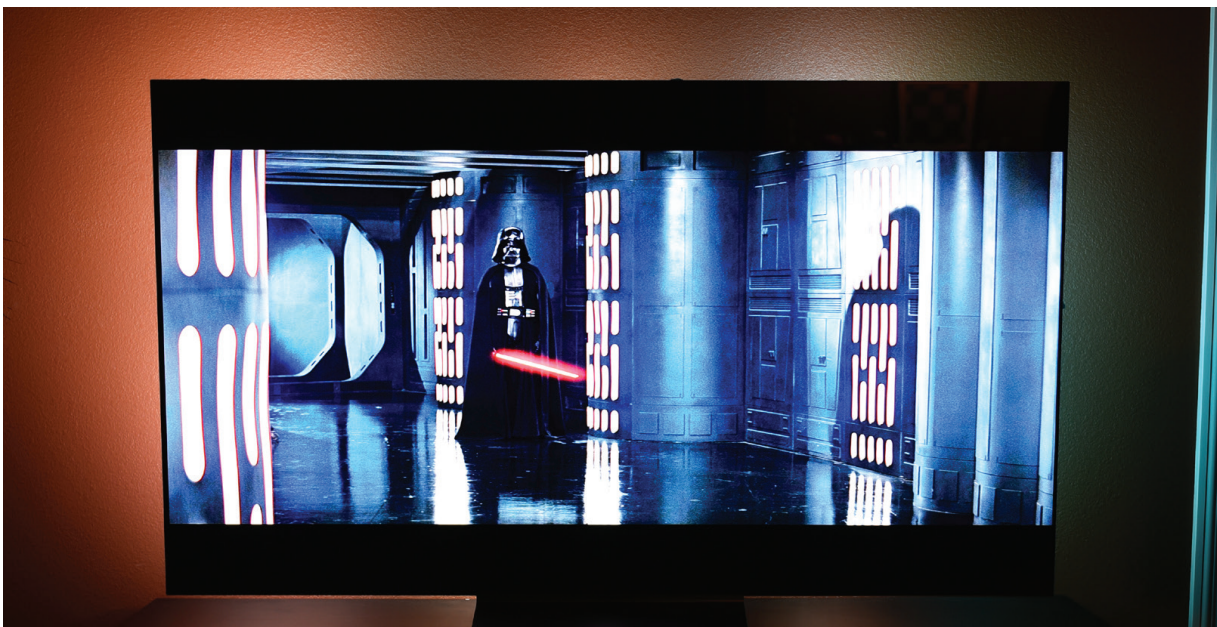
### TOTALLY IMMERSED

From here, if you have content playing on your HDMI streaming device, you should be able to see corresponding ambient lights playing. Cool, isn't it? You might need to fiddle with the software for a bit to get the LED placement correctly configured, along with color matching, black bar detection (recommended), and lots more.

Naturally, the effect is most noticeable in a darkened room or at night. You may find yourself repositioning the TV to get better light dispersion, or to prevent those bright LEDs being too visible by people standing alongside the television.

The Hyperion forum has guidance on setup and hiccups. Also, as noted in the materials list: Users with A/V receivers, surround sound, and CEC-connected devices will need to use slightly upgraded HDMI hardware to maintain those features. It quickly becomes a slippery slope.

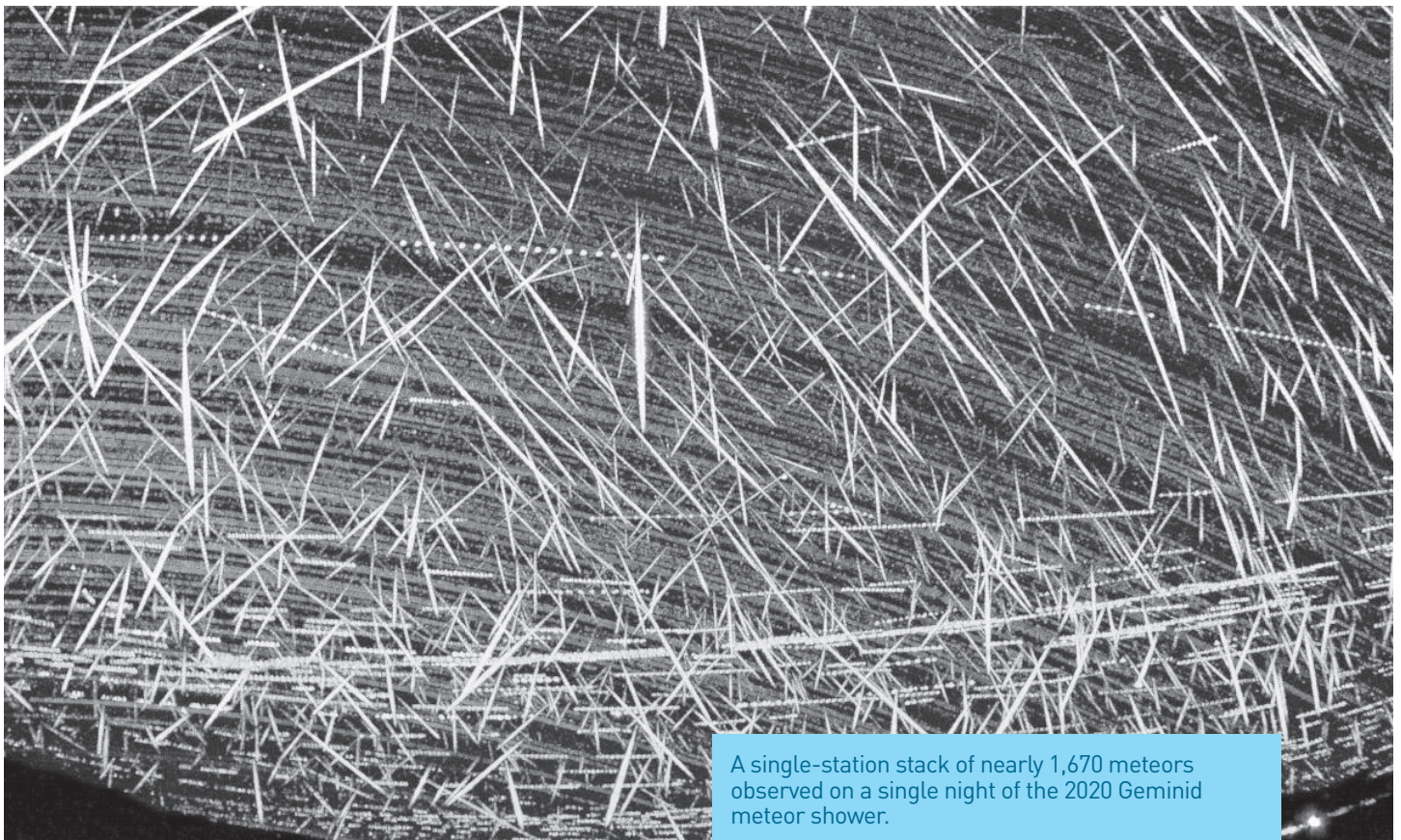
You've just added a new dimension to your home media center. Enjoy the immersion! 🎧





# 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**

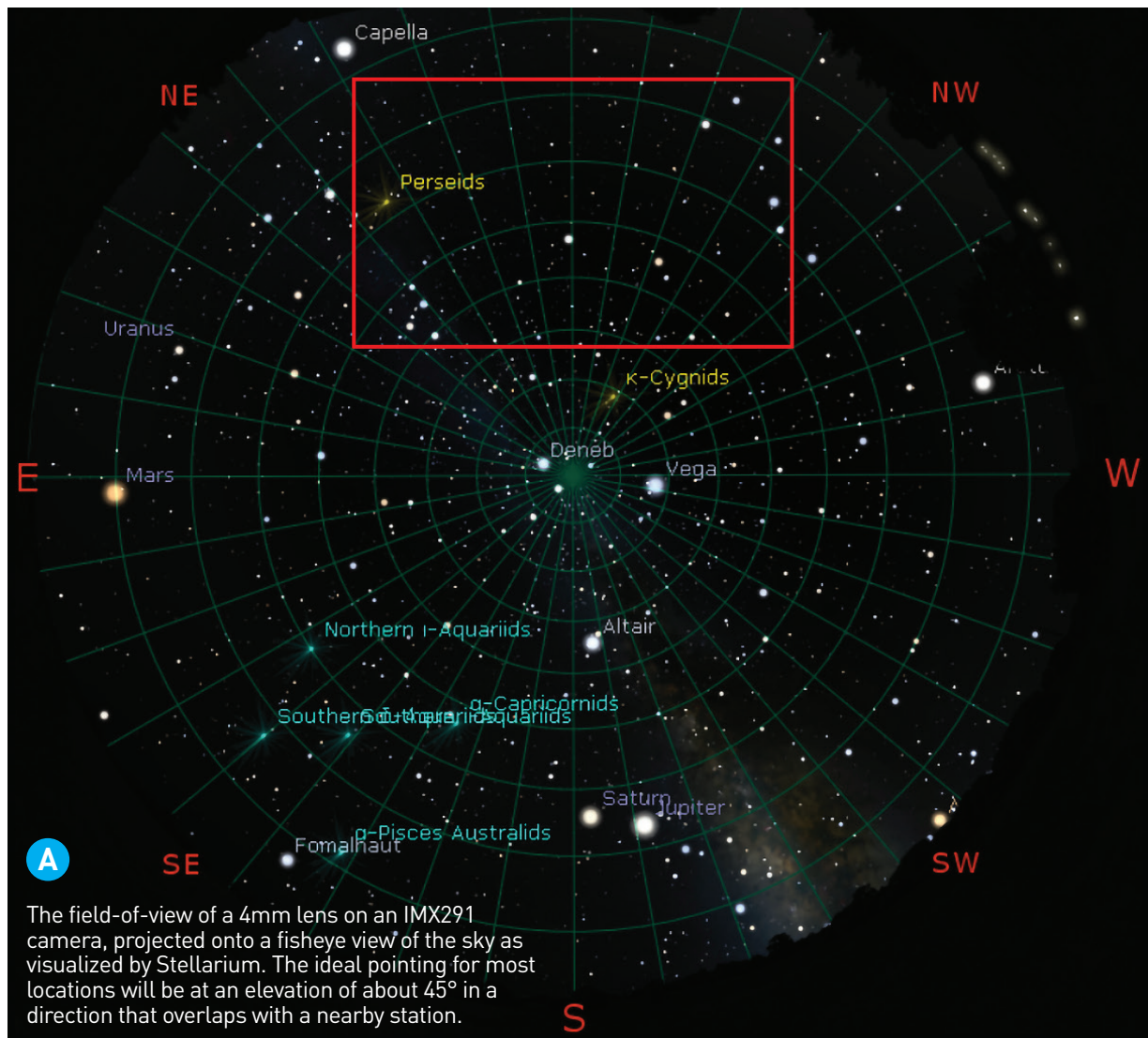
## MATERIALS

- » **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](https://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





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 1/3" 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:

$$\text{FOV} = 57.3 / \text{focal\_length} * \text{binned\_pixel\_size\_mm} * \# \text{ pixels}$$

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 <sup>(1)</sup> with IMX291 sensor	Pixel scale <sup>(2)</sup>	m @100km <sup>(3)</sup>	~M* <sub>limit</sub> <sup>(4)</sup>
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**

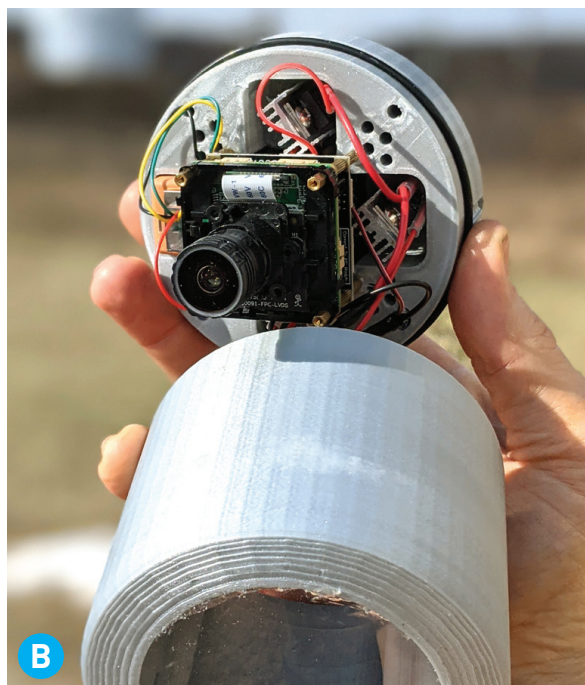
1. FOV refers to the field-of-view
2. The pixel scale is calculated at 720p resolution
3. m @100km refers to the spatial resolution at a distance of 100km
4. M\*<sub>limit</sub> is the limiting stellar magnitude

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



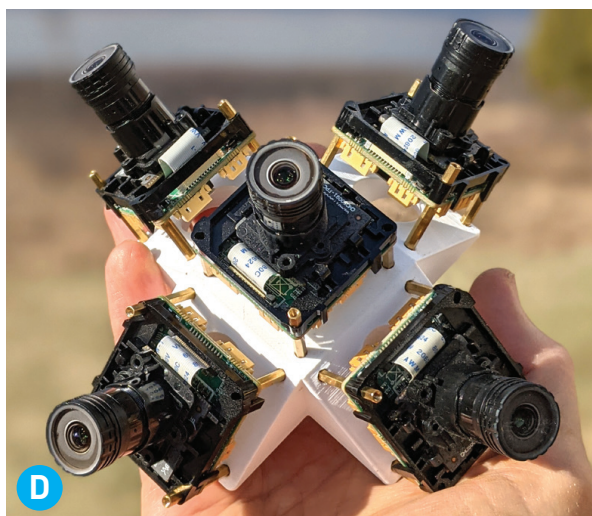
An enclosure can be printed to house the camera as well as a small fan and heater.



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

M. Mazur





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



**F** 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](http://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.

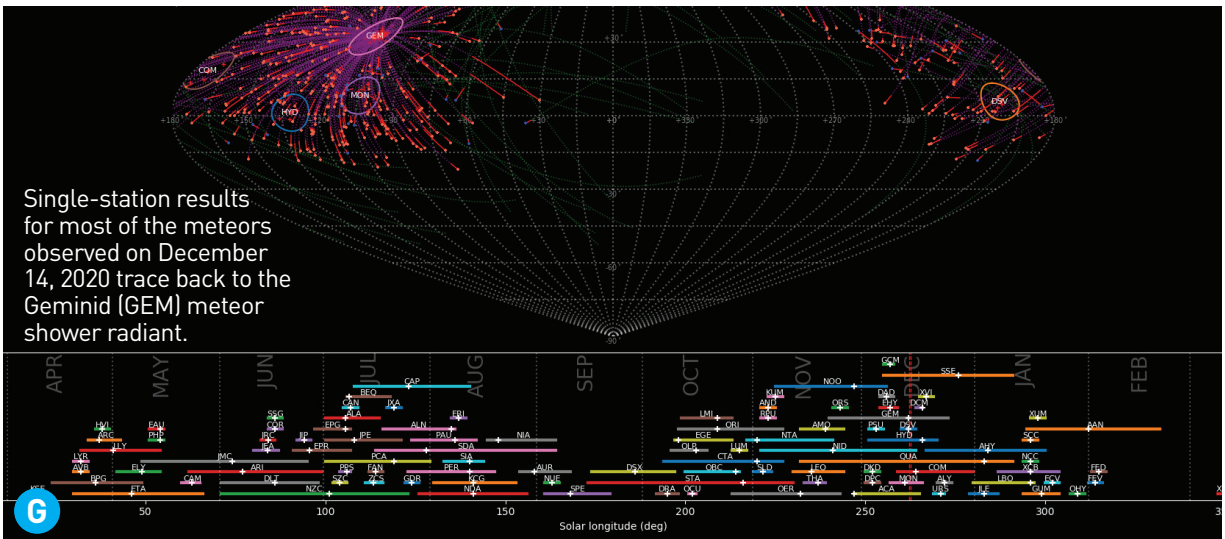
**TIP:** A pointy pin comes in handy after you inevitably assemble the cable connectors incorrectly and need to take them apart.

### 4. CONFIGURE THE CAMERA

The camera is ready for focusing and setup with a PC. Using CMS security cam software ([hasecurity.com](http://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

M. Mazur



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](http://makezine.com/go/RMS-images), or install the RMS code from scratch on an existing Raspbian installation, from [github.com/CroatianMeteorNetwork/RMS](http://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).

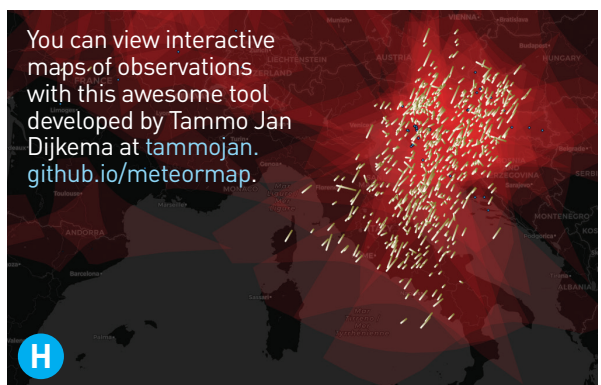
Finally, plug your camera into the POE injector and from there into the network port on the Pi. Give the camera a minute to boot and then check to see if it's visible with a `ping 192.168.42.10` or an `arp -a` command.

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](http://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



D. Vida, Tammo Jan Dijkema

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.

On Tammo Jan Dijkema's GMN meteor visualization tool ([tammojan.github.io/meteormap](http://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. ☑



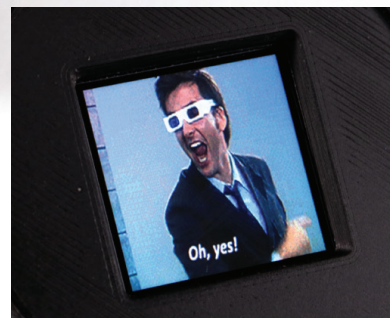
Learn more at [globalmeteornetwork.org](http://globalmeteornetwork.org), contribute at [github.com/CroatianMeteorNetwork/RMS](http://github.com/CroatianMeteorNetwork/RMS), and share your build at [makeprojects.com/project/build-a-raspberry-pi-meteor-camera](http://makeprojects.com/project/build-a-raspberry-pi-meteor-camera).



# The Magic GIF-Ball

Ask and behold your future told — in memes — with this new twist on a classic toy

Written and photographed by DJ Harrigan



**DJ HARRIGAN** is a designer and maker in Northern California. He has built various thingamajigs and written tutorials for Instructables.com, created puzzles for escape rooms, prototyped hardware for SLA 3D printers, and taught people how to solder and operate CNC lasers at makerspaces. He makes videos for Element14 Presents, and for his YouTube channel Mr. Volt, where he builds custom gizmos and animatronics from pop culture.

The classic Magic 8-Ball toy has been a mainstay of pop culture for decades: Ask a yes-or-no question about the future, then turn it over to see your answer “magically appear!” But it’s sorely limited by the 20 static replies imprinted on its floating icosahedron. We live in an age of endless stimulation, and equally limitless memes, so why not combine our modern entertainment sensibilities with the familiar form of that classic toy? Why not a Magic GIF-Ball?

I’m not the first person to decide that a fantastical fortune-telling sphere should answer with images instead of text, but I like to think I’ve made a tidier version that can easily be replicated by anyone with a basic knowledge of electronics and beginner’s grasp of the Linux command line. At its surface, this is still a toy, meant to surprise and delight your friends and loved ones as a spin on a familiar object. But it’s also an approachable project that tackles inputs, outputs, and programming (if you so choose) for the maker eager for more Raspberry Pi-related goodness.

## BUILD YOUR MAGIC GIF-BALL

Let’s do it. Before you build, you might like to watch my overview video at [youtube.com/watch?v=wDhnG030C2Q](https://youtube.com/watch?v=wDhnG030C2Q).

### 1. PRINT THE PARTS

You can find the parts packed in a tidy .zip over at Element14 ([makezine.com/go/element-14-magic-gif-ball](https://makezine.com/go/element-14-magic-gif-ball)). I’ve included the full assembly as a STEP file in addition to the individual STLs, so you can modify the design to your heart’s content.

There are seven parts to 3D print: the upper shell, lower shell, bezel, LCD retainer, mounting block, logo insert, and button retainer. I printed mine out of PLA at a 0.2mm layer height, but at higher resolutions your sphericity will of course be much better. For the shells, I recommend printing in dome vs. bowl orientation as this is less likely to warp. Print the logo vertically (Figure A), as the X-Y axis is much higher resolution than the Z (the logo is a very shallow curved piece, which is a challenge for all FDM 3D printers).

### 2. PAINT

Depending on what color filament you used, you’ll need to paint the shell and/or the logo pieces

**TIME REQUIRED:** A Weekend

**DIFFICULTY:** Easy/Intermediate

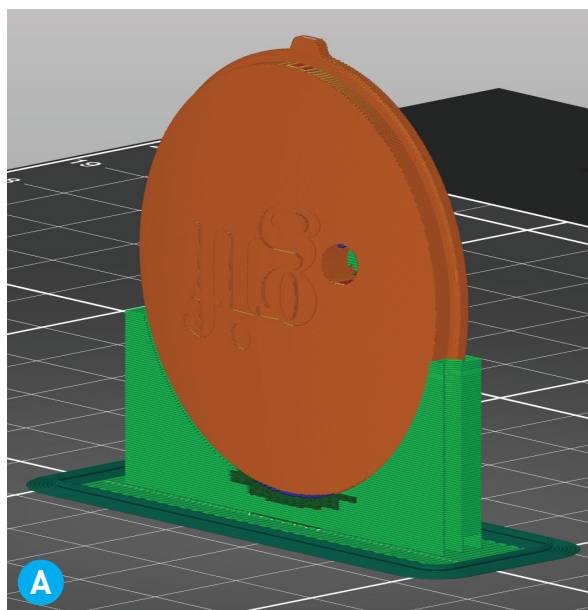
**COST:** \$100–\$200

## MATERIALS

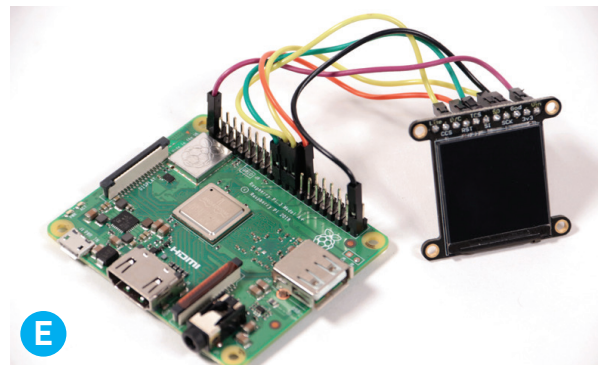
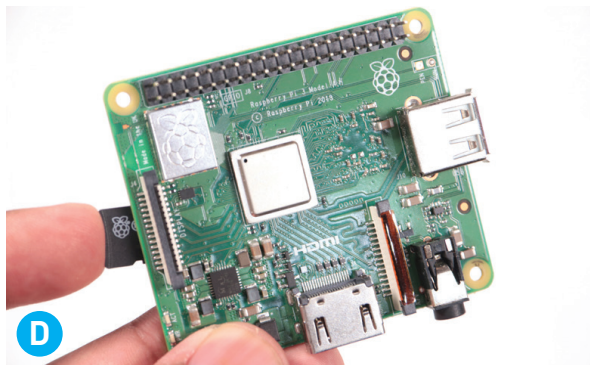
- » **Raspberry Pi Model 3 A+ mini computer**  
Newark 80AC9303, [newark.com](https://newark.com)
- » **LCD display breakout, 1.3", ST7789**  
Adafruit 4313, [adafruit.com](https://adafruit.com)
- » **Battery charger, PowerBoost 1000**  
Adafruit 2465
- » **Push-button power switch breakout**  
Adafruit 1400
- » **Tactile switch, 6mm, long plunger**  
Adafruit 1490
- » **Tilt-switch/vibration sensor** Adafruit 1766
- » **LiPo battery, 1800 mAh, single cell** such as  
Amazon B07TTD2SVC
- » **MicroSD card, 16GB** Adafruit 2820
- » **JST male connector** Adafruit 3814
- » **Jumper wires, female** Newark 42X1200
- » **Right angle headers, male, 0.1", 12 pos**
- » **Machine screws: M3×6mm (4) and M2.5×10mm (6)**
- » **CA glue** aka super glue
- » **3D printed parts** I printed them in black PLA filament, MatterHackers MH Build brand. Download the free 3D files for printing from Element14 [makezine.com/go/element-14-magic-gif-ball](https://makezine.com/go/element-14-magic-gif-ball) (free account required).
- » **Spray paint, white**

## TOOLS

- » **3D printer**
- » **Soldering iron**
- » **Paintbrush, fine tip**







(Figure **B**). If you print the logo in black, you can coat it a solid white and then scrape away (once dry) within the letters to reveal the black underneath, or as I did in the original version, use a fine tip brush to fill in the letters (Figure **C**). I'm not a fan of hand painting, but perhaps you have steadier hands than I!

### 3. SET UP THE RASPBERRY PI

There's very little to do Linux-wise to prepare your Pi (Figure **D**). First, flash the latest Pi OS onto your microSD card (Raspbian Lite is OK as the screen is run directly and doesn't mirror the X framebuffer). Raspberry Pi's new imager tool ([raspberrypi.org/software](http://raspberrypi.org/software)) makes this all much more streamlined if you haven't heard the word.

Get ready for basic headless setup by entering your Wi-Fi credentials. Once connected via SSH, you'll need to install the ST7789 Python module from Pimoroni ([github.com/pimoroni/st7789-python](https://github.com/pimoroni/st7789-python)) by entering the command:

```
sudo pip install st7789
```

They recommend installing some other common modules first, but I found that these were already

up to date in the current OS. Also, be sure to run `sudo raspi-config` and enable I<sup>2</sup>C and SPI.

### 4. WIRE AND TEST THE DISPLAY

Before connecting any other components, let's test our little TFT LCD display to make sure we've connected it properly and the software is good to go. With the Pi unpowered, use the female jumpers to make the following connections between the LCD and the Pi (Figure **E**); they'll communicate using the SPI serial protocol:

- 3v3 to any 3V pin
- TCS to Pi pin 7 — this is the SPI chip select pin for the TFT display
- SCK to pin 11 — the SPI clock input pin
- SI to pin 10 — the SPI MOSI pin (microcontroller out, serial in) to send data to the display
- D/C to pin 9 — the SPI data or command selector pin
- BL to pin 19
- GND to any Pi ground pin

Double-check your wiring, boot up the Pi, and `cd` into the folder `st7789-python/examples`. Then call:

```
python3 gif.py
```

And you should be greeted by Pimoroni's colorful "Deploy Rainbows" GIF.

## 5. PREPARE YOUR GIF SELECTION



The classic toy is restricted to a mere 20 answers, but we, dear reader, live in the future! Our GIF-Ball can for all intents and purposes house a virtually endless array of images, answers, and, most importantly, memes! So don't hold back when it comes to selecting your reply set.

If you're an 8-Ball purist, the original had 10 positive, 5 ambiguous, and 5 negative replies, so a mere 20 GIFs will be sufficient to capture that classic fortune-telling experience.

There are preselected GIFs in the download package in the *Images* folder, but if you're selecting your own, try to find images with a more square aspect ratio. The code will accommodate different image sizes and naming schemes, but keep in mind that while the mini IPS display is crisp, it's still only 240 pixels wide. Any image that ends in *.gif* will be randomly selected as a response.

## 6. SECURELY COPY YOUR FILES

Once you've curated your preferred selection of GIFs, it's time to slap 'em inside the Pi. Navigate to the directory on your main computer where you've got all your images, as well as the main program (*fortune.py*), and transfer them to the Pi via SCP (Secure Copy). For example:

```
scp *.gif pi@raspberrypi.local:
scp fortune.py pi@raspberrypi.local:
```

Your GIFs and the program should now be in */home/pi* so you can now test it with:

```
python3 fortune.py
```

**TIP:** Learn more about using Secure Copy at [raspberrypi.org/documentation/remote-access/ssh/scp.md](https://raspberrypi.org/documentation/remote-access/ssh/scp.md).

We haven't connected the tilt sensor yet, but you can short BCM pin 2 to GND and the program will register that as a "shake."

## 7. RUNNING THE PROGRAM AT BOOT

There are several options to have a program autorun, but we'll keep it simple. Just call:

```
sudo nano /etc/rc.local
```

After the line `exit 0`, add:

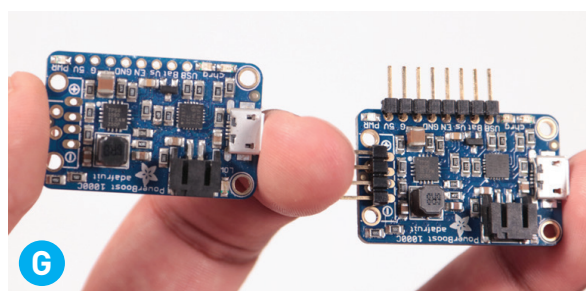
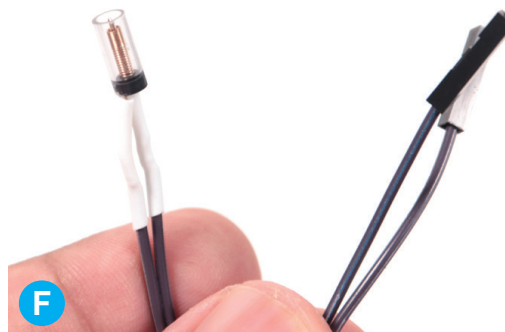
```
sudo bash -c '/usr/bin/python3 /home/pi/fortune.py' &
```

## 8. SOLDERING

Cut a female jumper in half and solder the halves to the leads of the vibration sensor (Figure F).

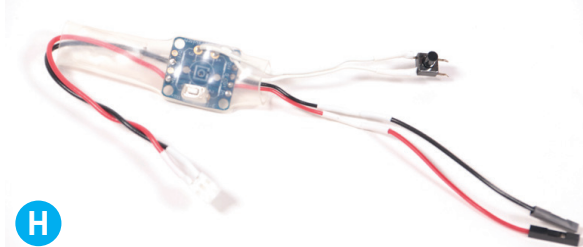
Solder a strip of 8 right-angle male headers to the PowerBoost side connections, and a strip of 4 to the power pads (Figure G).

Solder the JST socket to the pushbutton power switch (PPS) (red to IN, black to a GND connection). Add 3" leads to the pushbutton and solder to PPS pads 1 and 3. Cut and solder a red





jumper to the PPS OUT pin. Cut and solder a black jumper to the G pin of the PPS (Figure H).



### 9. ELECTRICAL CONNECTIONS

You already connected the LCD in step 4. Now, connect a jumper from the Pi's 5V pin to the +5V pin of the PowerBoost, and another jumper from a Pi ground pin to a GND pin of the PowerBoost.

Connect the jumper from the PPS OUT pin to the Bat pin of the PowerBoost. Connect the PPS G pin to a GND pin on the PowerBoost.

Lastly, attach one of the sensor pins to pin 2 on the Pi and the remaining pin to a GND pin on the Pi (Figures I and J).



### 10. PHYSICAL ASSEMBLY

Mount the switch in the square hole within the logo, using two M3 screws (Figure K). Glue the logo insert into the lower shell (Figure L).

Test-fit the clips in their matching holes, making sure they all seat at the same level when fully inserted. Put a couple drops of super glue on their tips and press them firmly down (Figure M).

Screw in the LCD retainer with two M3 screws, then the bezel with two more (Figure N).

Place the Raspberry Pi over the four mounting holes and align the mounting block above it, fastening it in place with four M2.5 screws (Figure O). Insert the battery (Figure P).

Insert the vibration sensor into one of the holes on the side of the mounting block (Figure Q). Secure the PowerBoost to the mounting block with two M2.5 screws (Figure R).

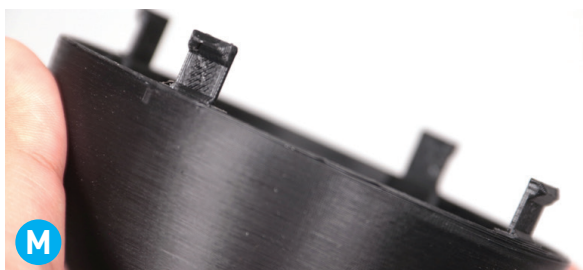
Wait a few hours for the glue to fully cure, then connect the two halves together, making sure not to pinch any stray wires (Figure S).

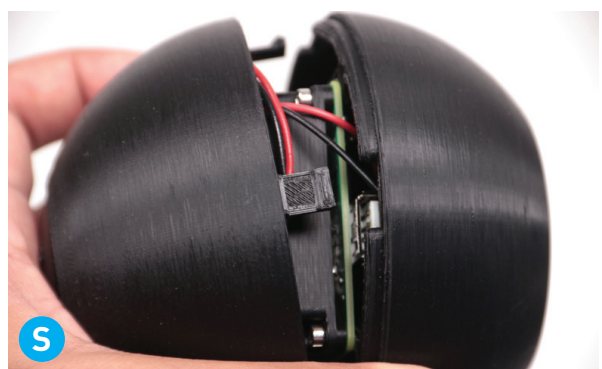
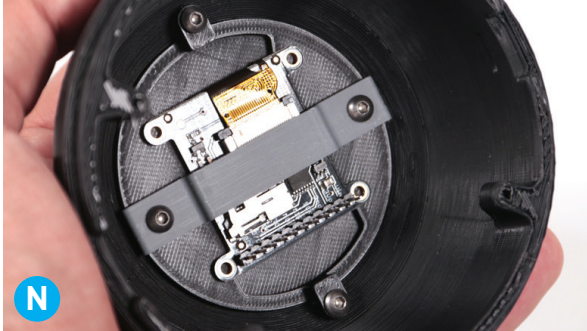


### NOW ASK THE REAL QUESTIONS

Your Magic GIF-Ball is toggled on and off via the pushbutton period in the .gif logo, which by Pi standards is considered a hard shutdown, so you may want to set it to read-only mode if you're worried about corruption ([medium.com/swlh/make-your-raspberry-pi-file-system-read-only-raspbian-buster-c558694de79](https://medium.com/swlh/make-your-raspberry-pi-file-system-read-only-raspbian-buster-c558694de79)).

Once the Pi boots up, using the Magic GIF-Ball is very similar to the original toy: Shake it up and receive the answers you so desperately seek! Should I buy Bitcoin? Should I eat fried chicken





every day? Will I find true love next Monday?

The GIFs are set to loop twice, since most are so short that they usually finish by the time the user is done shaking. Shaking the ball during a currently playing reply won't interrupt it, so if you want that mode of operation you'll have to tweak the code to your liking.

You might also notice that the charging port is not externally accessible, which means it'll have to be popped open to recharge. Is this silly? Yes. But I've shown the MGB to most of my friends and family, and there's still plenty of charge left months later. It's not exactly a daily use fortune-teller anyhow.

I've also pondered a few possible upgrades you may wish to explore:

- **Simplify it.** I went with the Pi as it was the least hassle to prove this as a concept, but this could be adapted to run via an ESP32, Teensy 4.0, or the new Pi Pico!
- **Unlimited GIFs!** It shouldn't be unreasonable



to pull GIFs down via Giphy or another memetic API, but you'll need to make sure your MGB has a constant internet connection.

- **Cleaner power.** Break up the smooth, portless surface with a handy jack for charging, or add another switch to trigger a safe shutdown.
- **Cut corners.** Did you know mini round LCDs are a thing now? Check out Pimoroni PIM570, for example. One of those would make an awesome display that matches the original aesthetic even better.
- **DIY the enclosure.** This project could easily be stuffed into a DIY Christmas ornament, hamster ball, or original 8-Ball.
- **Amp it up.** We've got all this processing power, why not have the MGB respond with full-fledged video clips? Perhaps for version 2... 🚀



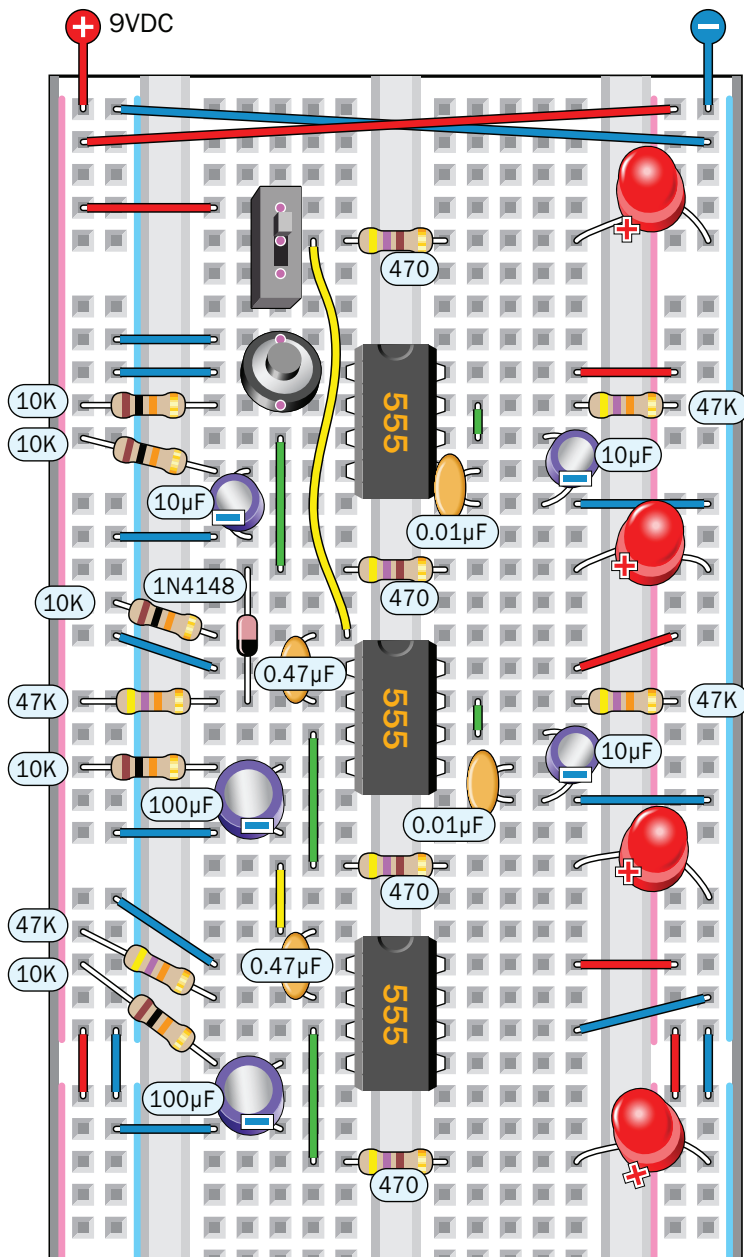
Build your own or make a mod? Leave a comment on the original project page: [makezine.com/go/element-14-magic-gif-ball](https://makezine.com/go/element-14-magic-gif-ball)



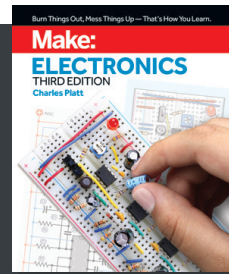
# Intruder ALERT!

Build this alarming project two ways — from bare components and from a Pico microcontroller

Written by Charles Platt with Fredrik Jansson



This project is adapted from the staggeringly popular *Make: Electronics Third Edition*, which has been completely re-re-written with most photos and schematics replaced and updated! [makershed.com/platt](http://makershed.com/platt)



**TIME REQUIRED:**  
1-2 Hours

**DIFFICULTY:** Intermediate

**COST:** \$20-\$30

**MATERIALS**

- » Resistors: 470Ω (4), 10kΩ (5), 47kΩ (4), and 470kΩ (2)
- » Capacitors: 0.01µF (2), 0.47µF (2), 10µF (3), and 100µF (2)
- » LEDs (4)
- » 555 timer ICs (3)
- » 1N4148 diode (1)
- » Tactile switch (1)
- » SPDT or DPDT switch (1)
- » 9V battery and battery clip with leads
- » Solderless breadboard
- » Raspberry Pi Pico microcontroller (optional)

Charles Platt

During 2021 I started writing a third edition of my book *Make: Electronics*. One of the pleasures of this task was that I could rethink all the circuits, and with the benefit of hindsight, I saw how to simplify one project in particular: The intrusion alarm.

You'd think an alarm should be easy enough. Just place some sensors on doors and windows, wire them to a beeper, and — job done!

But it's not so simple. First, the alarm has to notify you that all the doors and windows are closed before you activate it. Then you need a delay (which I call the Exit Delay) so that you can leave the area without causing the alarm to start making a noise. When you come back, you need another delay (which I call the Last Chance Delay) to prevent the alarm from beeping or wailing till you have a last chance to switch it off. Figure A shows what I mean.

The word *delay* suggests a component that measures time. That could be a timer, couldn't it? In fact you can build this circuit entirely with three 555 timers — the old-school chips that have outsold all other chips ever made. Along the way you'll learn about pullup and pulldown resistors, diodes, and coupling capacitors, which can be useful in many other applications

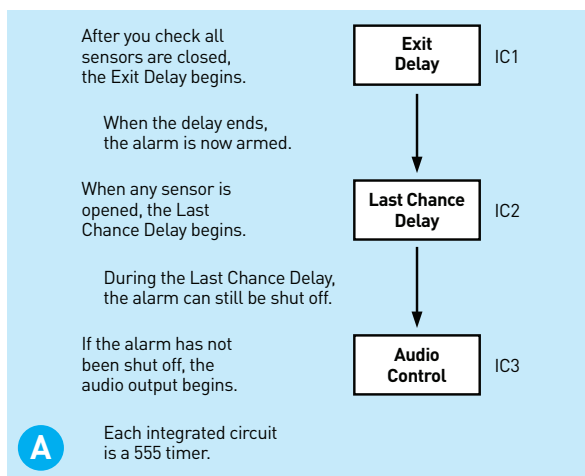
In Figure A, IC1, IC2, and IC3 are acronyms for *integrated circuits*, which are the three 555 timers.

## REED SWITCHES

A basic alarm installation often uses pairs of sensor modules such as those in Figure B. One module contains a magnet, while the other contains a *reed switch*.

This type of switch consists of a sealed glass capsule containing two magnetized contacts. A magnetic field will push them together (if they are normally open) or pull them apart (if they are normally closed). Figure C shows what a reed switch looks like, in case you are wondering, and Figure D shows how the sensor works. Letters N and S stand for north and south, the two poles of a magnet.

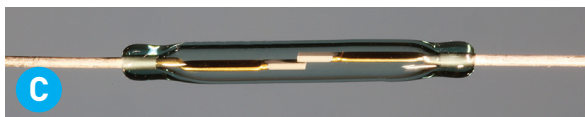
For an alarm circuit, you will need the type of sensor with contacts that are normally open. You mount each magnetic module on a door or a window, and you mount each switch module on the adjacent frame so that it almost touches the



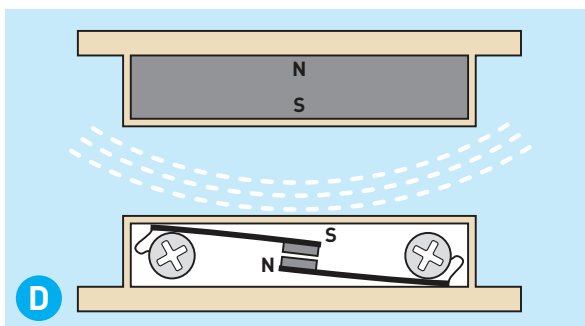
Overview of alarm behavior.



The two modules of an alarm sensor.



Closeup of a reed switch which is about 1/2" long.



How a reed switch works inside an alarm sensor. The dashed lines represent a magnetic field.

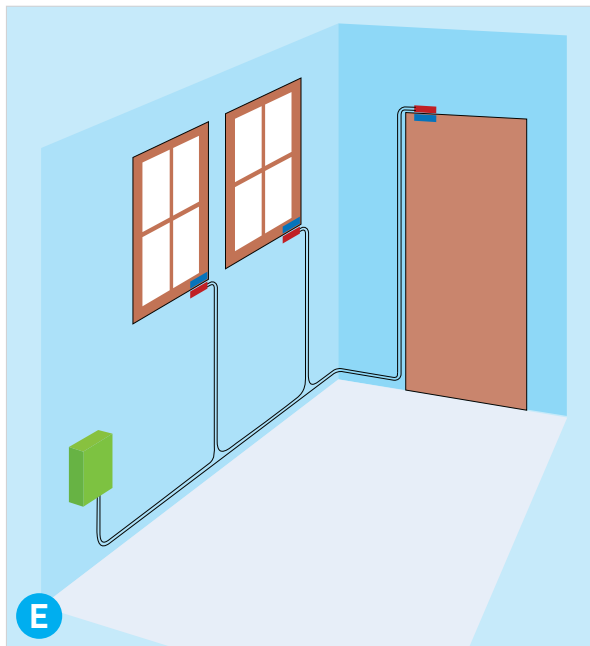


**CHARLES PLATT** is the author of the bestselling *Make: Electronics*, its sequel *Make: More Electronics*, the *Encyclopedia of Electronic Components Volumes 1–3*, *Make: Tools*, and *Make: Easy Electronics*. [makershed.com/platt](http://makershed.com/platt)



**FREDRIK JANSSON** is a researcher in physics and weather modeling, and coauthor of the *Encyclopedia of Electronic Components Volumes 2 and 3*.



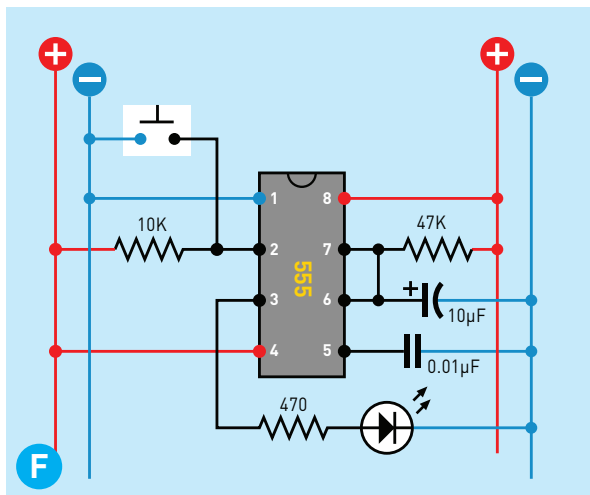


Alarm wiring using magnetic sensors.

magnetic module when the door or the window is shut. In this way, the magnets hold the reed switches closed.

In Figure E, the magnetic modules are blue rectangles while the switch modules are red and the alarm is in a green box. Because the switches are wired in series, if any switch opens, the continuity of the circuit is interrupted. This is known as a *break-to-make* type of circuit, because any break in the circuit — at any window or door — makes the alarm go off. Another advantage is that the circuit will also be triggered if someone tries to interfere with it by cutting the wire.

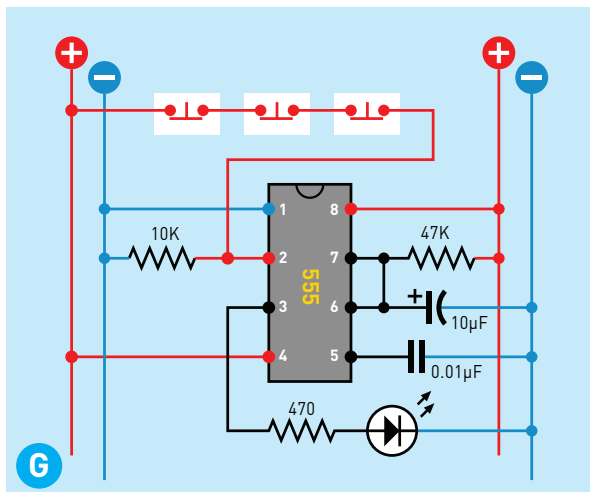
### TRIGGERING A TIMER



Test circuit demonstrating a timer as you would wire it on a dual-bus breadboard, with a pullup resistor.

Figure F shows a breadboard-layout test circuit for a 555 timer in *monostable mode*. Pin 2 is the “Trigger Pin,” and when its voltage is pulled down below  $\frac{1}{3}$  of the supply voltage, this triggers the timer, which emits a high pulse from Pin 3, the Output Pin. The duration of the pulse is controlled by the 47K resistor and 10 $\mu$ F capacitor, which create a pulse of about 3 seconds. You can learn more about 555 timers from any introductory source about electronics (including my own book, naturally).

You must not allow Pin 2 to float at an indeterminate voltage, so a 10K *pullup resistor* keeps the pin near the voltage of the power supply until the pushbutton bypasses it with a direct connection to negative ground. Pullup and *pulldown resistors* are important features of countless circuits.



The circuit in Figure F has been rewired to work with normally-closed alarm sensors and a pulldown resistor.

I wanted to use the timer in this mode as IC2, the Last Chance Delay timer, but I encountered a problem. When a door or window is opened in an alarm circuit, it doesn’t make a connection, like the pushbutton in Figure F. It breaks a connection. I had to revise the circuit as in Figure G, where I have indicated the sensor switches as three normally-closed buttons. Because they are attached to the positive power supply, Pin 2 is normally positive, as shown. But if any of the sensors are opened, the positive connection is cut off, and a 10K pulldown resistor makes Pin 2 negative, which triggers the timer.

Next I ran into another problem. What if someone should open a door or window, and leave it open indefinitely? The voltage on Pin 2 will

Charles Platt

stay low indefinitely, creating a Last Chance Delay that lasts indefinitely. If the delay doesn't end, the alarm will never start to make noise. Not good!

My answer was to insert a *coupling capacitor* and a pullup resistor, as in Figure H. So long as all the switches are closed, everything is stable. The voltage to Pin 2 remains positive, and the timer doesn't do anything. But when a switch is opened, the pulldown resistor (on the left) takes over, causing a short negative pulse to pass through the capacitor.

This is the odd thing about capacitors. They block DC current, but if you change the voltage suddenly on one side, a short pulse of current seems to emerge on the other side. This is known as *displacement current*, and at the risk of promoting myself, I have to tell you that I explain this concept elsewhere in my book *Make: Electronics*. The concept is often misunderstood, so I have included a thorough explanation in the Third Edition.

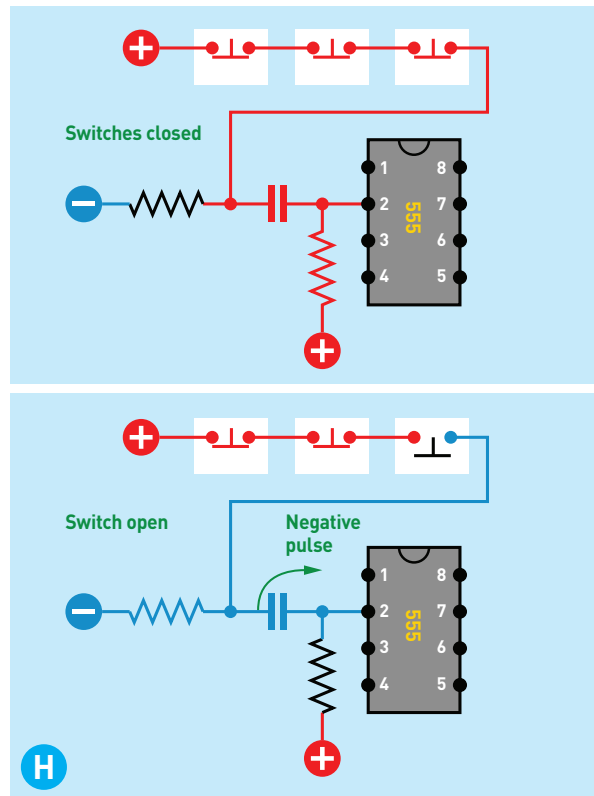
This circuit in Figure H will only work if the resistors and the capacitor are chosen correctly. How did I know what values to use? Well, I didn't! I just tried various values while using an oscilloscope to check the voltage on Pin 2, until I found an arrangement which worked reliably. A pullup resistor of 47K, a pulldown resistor of 10K, and a capacitor of 0.47μF was a good combination.

**NOTE:** Oscilloscopes are much cheaper than they used to be. For around \$100, you can buy either a handheld oscilloscope or a "virtual oscilloscope" which plugs into a USB port on a computer and generates a higher-res display. This is a very valuable tool if you want to build circuits.

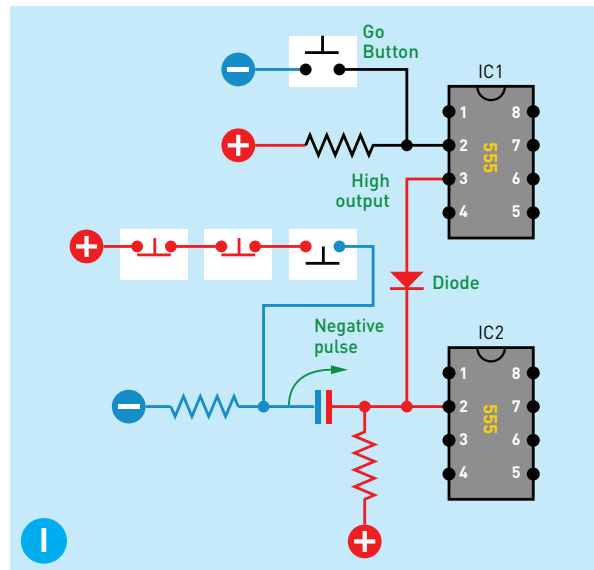
Maybe my arrangement with two resistors and a coupling capacitor seems complicated, but it keeps the timer happy, and if you want a circuit to function reliably, the components must be happy at all times.

### THE EXIT DELAY

Having figured out how to trigger IC2, my next step was to think about IC1, which creates the Exit Delay. During this delay, IC1 has to prevent IC2 from being triggered when you open a door to leave the area.



Simplified schematic showing how a coupling capacitor works.



IC1 now controls IC2 through a diode.

Figure I shows a simple way to do this. First, you press the "Go Button" when you're ready to go, and this starts a single high pulse as the Exit Delay from IC1. (No need for a coupling capacitor with the Go Button, because you will only press it briefly. You won't sit there holding it down.)

The high output from IC1 passes through a diode that connects directly with the Trigger Pin



of IC2. While this is going on, if an opened door or window creates a negative pulse, it's not powerful enough to get through, because the voltage through the diode overrides it.

How could I know this? By testing it. That was the only way.

I added a diode because the output from IC1 is normally low, when it isn't being triggered, and I didn't want the low output to get through to IC2.

### THE SEQUENCE

The sequence may be hard to understand, so I'll sum it up. You press the Go Button to start IC1, and it generates a high output that lasts long enough for you to leave the area. Now IC2 doesn't notice if you open a door, because the voltage from IC1 prevents IC2 from being triggered.

At the end of the exit delay, IC1's output goes low, but this is blocked by the diode, so it doesn't have any effect on IC2. The pullup resistor now keeps Pin 2 high on IC2, but if any door or window is opened, the negative pulse breaks through and triggers IC2.

When IC2 is triggered, its output goes high for a limited time — long enough for you to switch off

the alarm, if you're the one entering the space. If you don't switch off the alarm, the output from IC2 goes low at the end of its cycle, and this will start some noise — somehow. I haven't described that part yet.

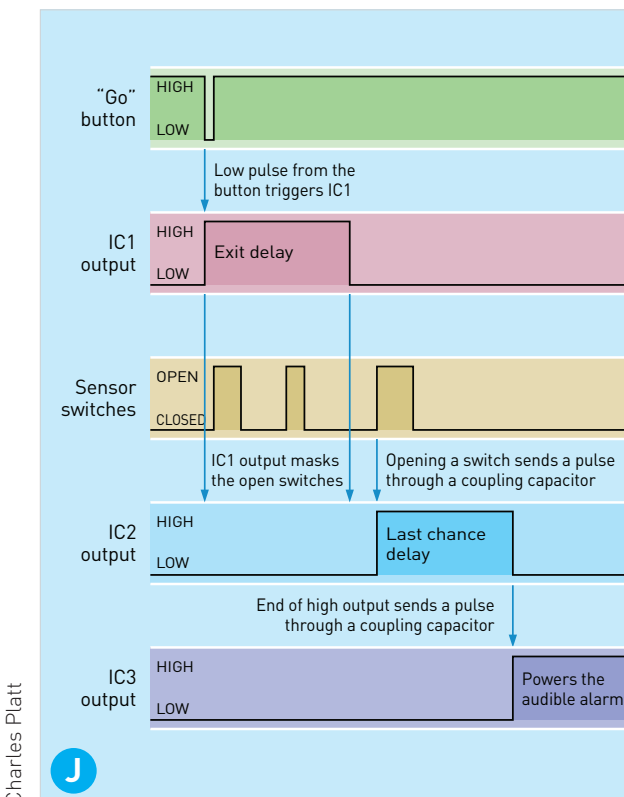
Figure J may help to clarify the sequence, in theory at least.

### THE NEVER-ENDING NOISE

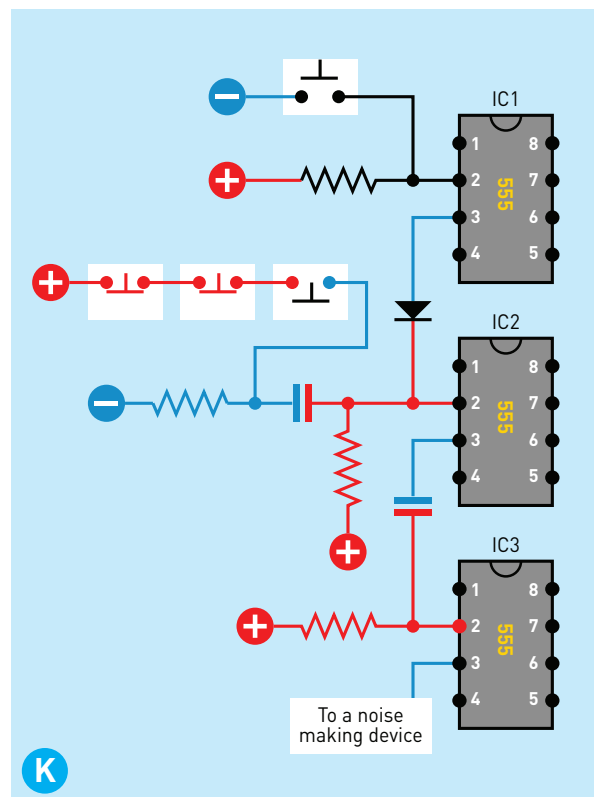
When I saw that the output from IC2 would drop from high to low at the end of the Last Chance Delay, I realized I could couple it through (guess what) another coupling capacitor, to trigger IC3. This connection is shown in Figure K, which shows the components at a moment when none of the timers is active.

Now, let's suppose there is an intruder, and IC2 triggers IC3. How long should its output stay high? Well — indefinitely! You want the sound of the alarm to continue until an authorized person (such as you) turns it off.

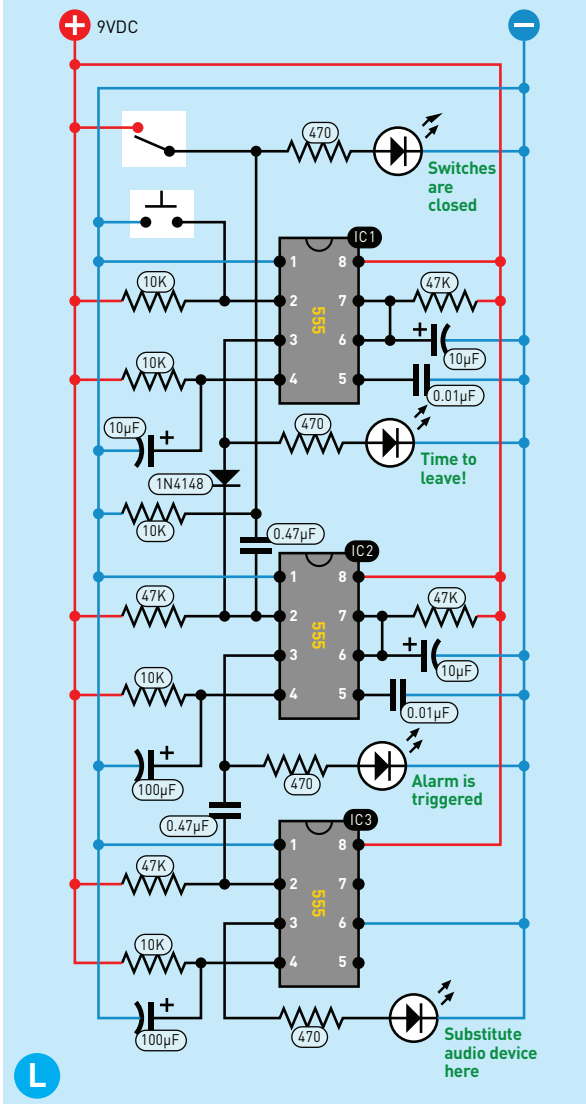
Fortunately there is a way to wire IC3 so that when its output starts, it never stops. This is known as *bistable mode*. The timer will deliver a high output until you shut it down. The secret is to



Sequence of events when the alarm is working.



Simplified schematic showing how the third timer completes the basic circuit.

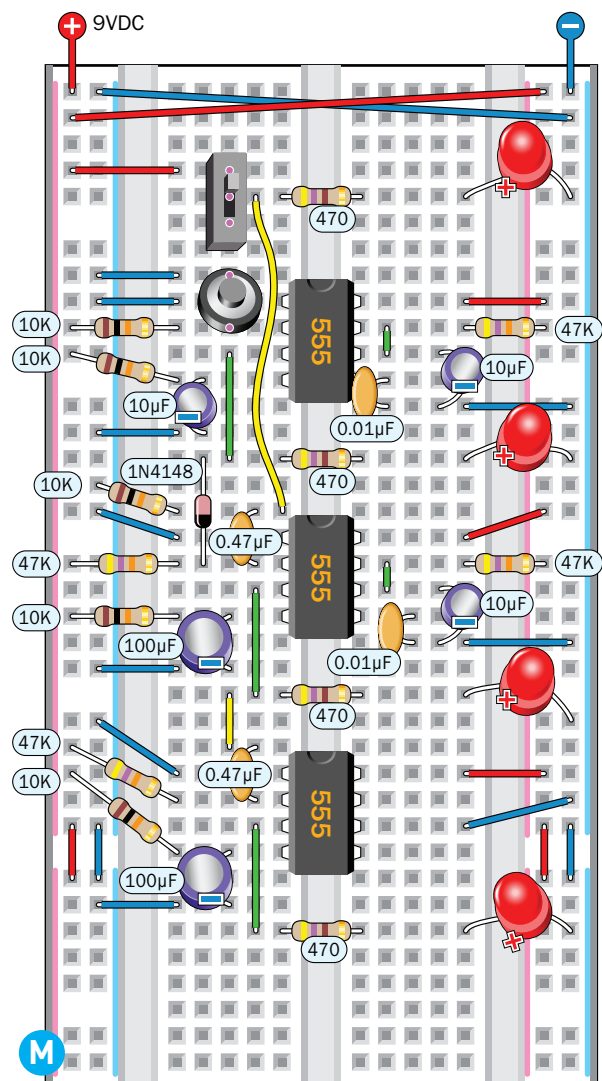


The complete circuit (without a noise maker).

eliminate the timing capacitor which is normally attached to Pin 6, and wire Pin 6 to negative ground. You can see this in Figure **L**, which shows a complete schematic for the circuit. The only thing missing is some kind of noise maker, but that could just be a piezo beeper which has an audio circuit built in. Figure **M** shows how the circuit can be breadboarded, using LEDs for testing purposes.

### HERE'S YOUR TESTING PROCEDURE:

- First close the switch at the top of the breadboard, which simulates closing all the doors and windows where the alarm system has been installed. The top LED lights up to confirm the continuity.
- Now you can press the Go Button, triggering IC1, which lights the "Time to leave!" LED. While this is on, the Exit Delay is in effect. You can open the sensor switch and nothing will



The breadboarded version.

happen. The circuit allows an exit delay of only 3 seconds for testing purposes, but that's sufficient for you to open and close the sensor switch (the door).

- At the end of the Exit Delay, the second LED goes out. Now the alarm is armed and ready, and if you open the sensor switch, this allows a negative pulse to trigger IC2, which begins the Last Chance Delay. During this period, the third LED lights up to warn you that you have one last chance to switch the alarm off.
- At the end of the Last Chance Delay, the output from IC2 drops to a low state, which sends a negative pulse through the coupling capacitor to trigger IC3, which switches on and keeps itself switched on, because it is wired in bistable mode. The last LED lights up to tell you that if you substitute a noise maker on the output of IC3, it will start making noise (and keep making noise until you turn it off).



To increase the delays from 3 seconds to 30 seconds, replace the two 47K resistors on the right with 470K resistors.

If you're wondering about the purpose of the 10 $\mu$ F and 100 $\mu$ F capacitors on the left, they prevent the 555 timers from triggering themselves when you first apply power to the circuit.

## FINISHING TOUCHES

Naturally if you want to create a finished version, you would solder the components to perforated board and put it in a project box. Figure N shows some components with a template for drilling holes in the lid of a box. I added an extra oscillator circuit to drive the speaker. Figure O shows it assembled.

One word about installing sensor modules on windows and doors: Remember, they have to be in *series*, not in parallel. Figure P shows how to do this with two-conductor wire.

I skipped some details in describing the project here, because I don't have unlimited space in the magazine. You'll find more explanations in my book — together with a confession describing how the circuit didn't work properly in the first version that I tried. I was happy when that happened, because when something doesn't work, it's always a valuable learning experience.

## USING A PICO

You can, of course, use a microcontroller instead of 555 timers. My collaborator Fredrik Jansson likes the new Raspberry Pi Pico chip, because it's so powerful and so affordable, so he wrote some code for it in MicroPython to behave the same way as my circuit, as shown in Figure Q. His comments in the code show you what the alarm program is doing at each step; you can type it in as you see it here, or download it from the project page at [makezine.com/go/intruder-alert](https://makezine.com/go/intruder-alert).

Fredrik's breadboarded circuit for the Pico is shown in Figure R, using sensor switches that are normally closed, and adding a reset button to boot the Pico. The components look wonderfully simple, but of course you do have to write the program, upload it, test it, track down your syntax errors, and upload it again.

Personally I prefer old-school components,



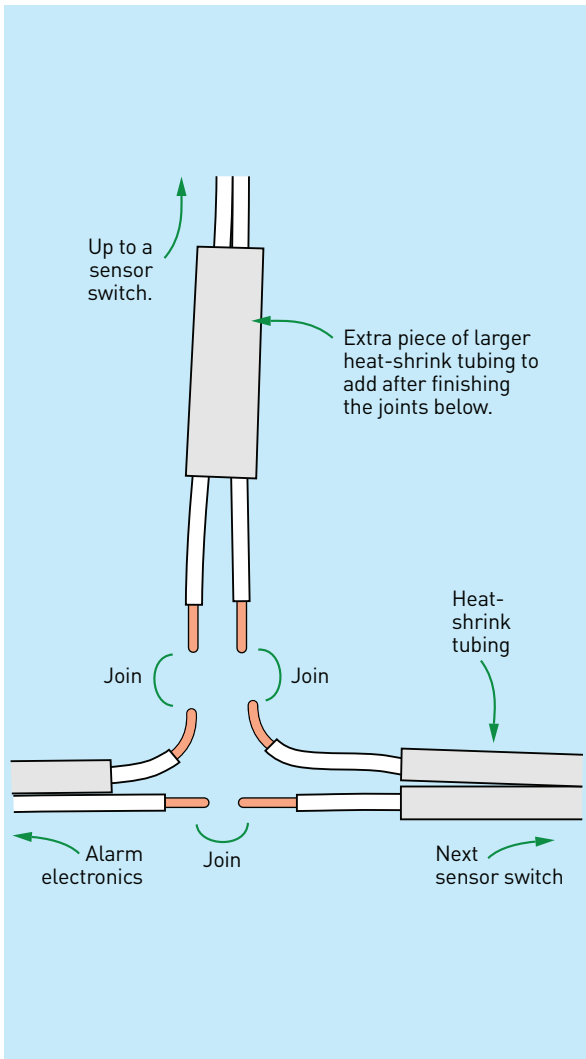
Components and layout for a project box.



The finished installation.

because I think they're fun — and you will learn about concepts such as diodes, pullup resistors, and coupling capacitors as you go along. 🚫

Charles Platt



```
# Raspberry Pico version of Make: Electronics alarm

from machine import Pin
from time import sleep

# LED outputs
l1 = Pin(17, Pin.OUT) # Switches are closed GP17
l2 = Pin(16, Pin.OUT) # Time to leave GP16
l3 = Pin(14, Pin.OUT) # Alarm is triggered GP14
l4 = Pin(15, Pin.OUT) # Alarm! GP15

# Inputs with pulldown
go = Pin(8, Pin.IN, Pin.PULL_DOWN) # Go button
sensors = Pin(20, Pin.IN, Pin.PULL_DOWN) # Sensor switches

# show sensor state on LED 1
l1.value(sensors.value())

# function to be called when the sensor pin changes
# show sensor state on LED 1
def sensors_change(p):
    l1.value(sensors.value())

# connect function to the sensor pin change interrupt
sensors.irq(trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING,
            handler=sensors_change)

# The alarm logic starts here

# Wait for GO button press
while go.value() == 0:
    pass

l2.value(1) # time to leave
sleep(10) # exit delay
l2.value(0)

# wait for sensors to open
while sensors.value() == 1:
    pass

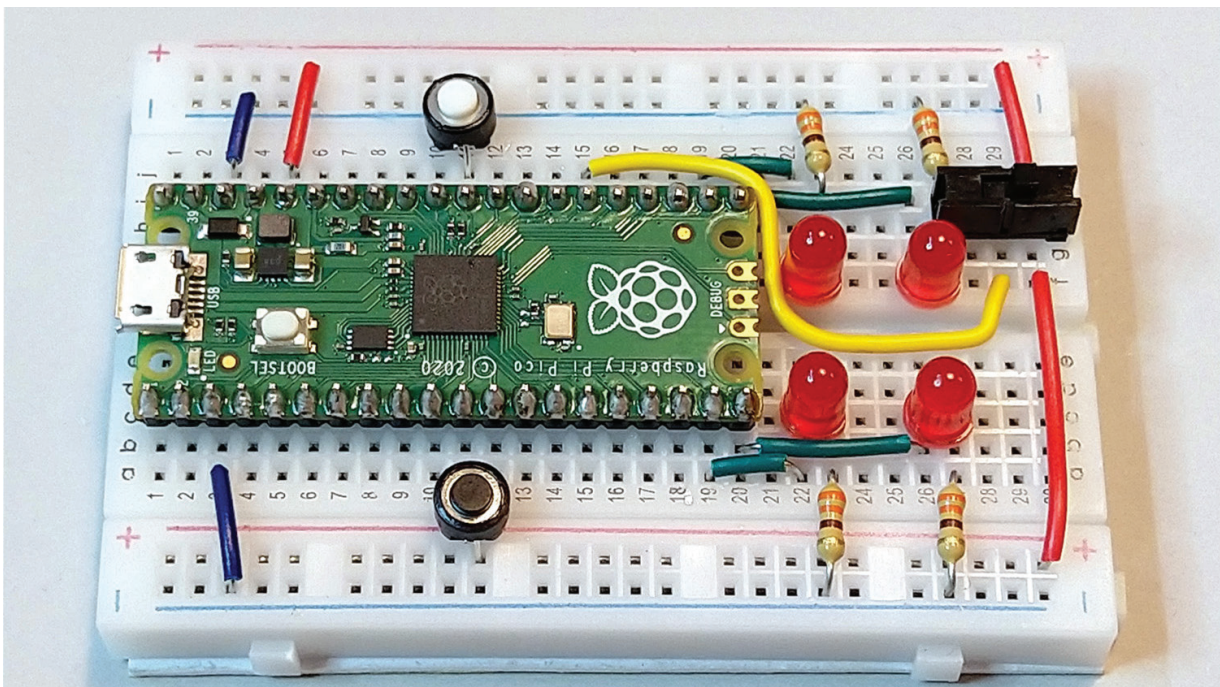
l3.value(1) # alarm is triggered
sleep(10) # last chance delay
l3.value(0)

l4.value(1) # Alarm!

# the alarm stays on until reset/power-off
while True:
    sleep(1)
```

**P** How to wire a branch in your sensor network.

**Q** Source code for a Raspberry Pi Pico microcontroller.



**R** Breadboarded alarm circuit using the Pico.



# Song Spotter!

## Teach a Raspberry Pi to identify 3,000 birds by sound alone

Written by Keith Hammond

In 2016 the bird nerds at the Cornell Lab of Ornithology collab'ed with coders at TU Chemnitz in Germany to create **BirdNET** — an artificial neural network that recognizes 3,000 different birds by sound alone ([birdnet.cornell.edu](http://birdnet.cornell.edu)). Next they turned it into a free phone app — imagine stashing John James Audubon, Sir David Attenborough, and the real James Bond (legendary ornithologist) in your pocket to help identify avian species when you're out on the trail.

But what if you'd like to ID every bird at your location, all the time? Now there's **BirdNET-Pi**, a dedicated version for installing on a Raspberry Pi ([birdnetpi.com](http://birdnetpi.com)). It does 24/7 recording and analysis, automatically extracts birdsongs and creates spectrograms (Figure A), and logs every species detected (Figure B). If you wish, it'll host its own web server for remote access, stream live audio, send notifications, and integrate with BirdWeather to share your data. It's an awesome citizen science project for just a few bucks!

The team has begun work on a TFLite version for the Arduino Nano 33 BLE, because it has a mic, says team leader Dr. Stefan Kahl, but "Right now, our focus is Raspberry Pi as an embedded platform because it can run Python and TFLite without having to sacrifice much performance." You can adapt BirdNET to the platform of your choice; it's an open source project at [github.com/kahst/BirdNET-Analyzer](https://github.com/kahst/BirdNET-Analyzer). 🐦



**KEITH HAMMOND** is editor-in-chief of *Make:* and believes ravens are the best bird, because they *rawk*.



Adobe Stock-Piotr Krzeslak, BirdNET-Pi

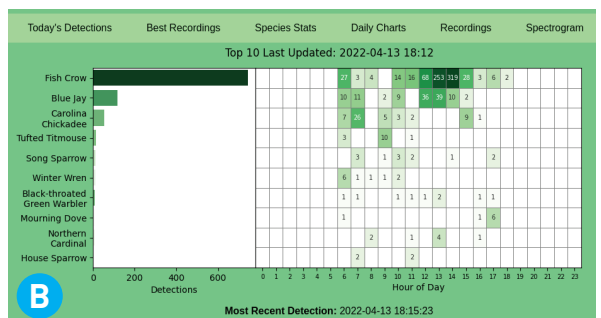
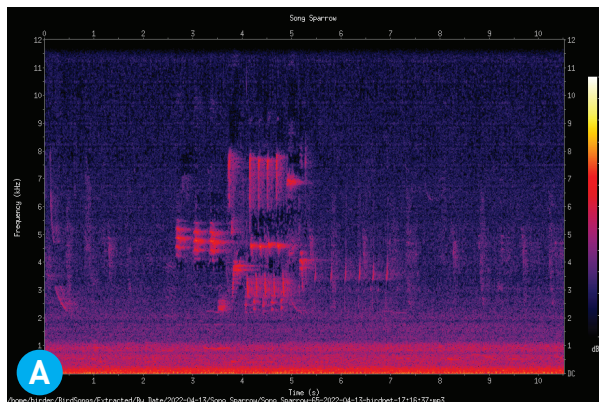
**TIME REQUIRED:** Minutes

**DIFFICULTY:** Easy

**COST:** \$20-\$30

### MATERIALS

- » **Raspberry Pi single-board computer** running Raspberry Pi OS Lite (64-bit); models 3x, 4, or Zero 2 W
- » **USB microphone**



# 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

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.

**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](http://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](http://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.



**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.

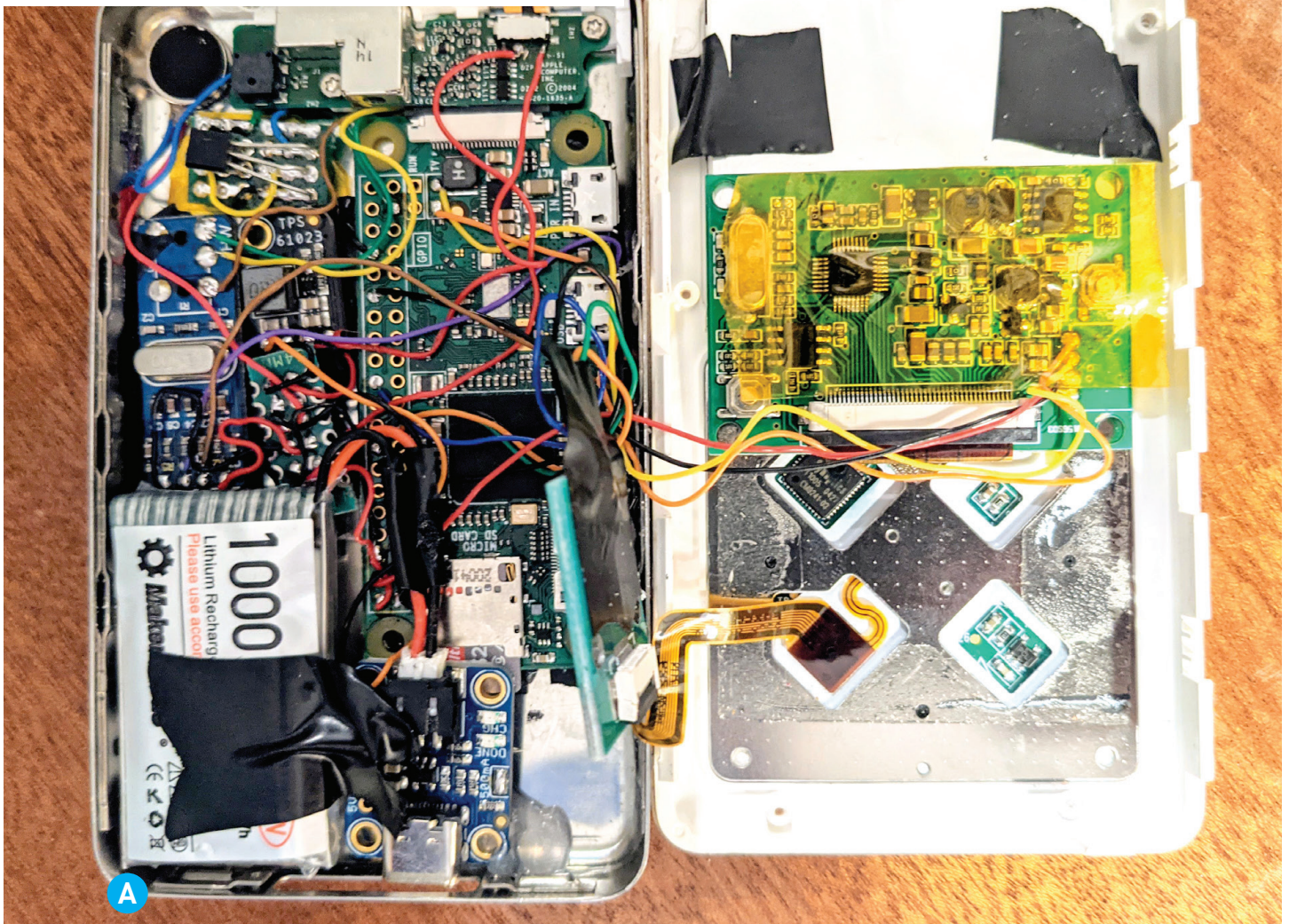
## BUILD YOUR SPOTIFY CLICK-WHEEL IPOD

Before you start, please watch my walk-through videos, Part 1 at [youtu.be/ZxdhG10hVng](https://youtu.be/ZxdhG10hVng) and Part 2 at [youtu.be/q0pUPab7Rms](https://youtu.be/q0pUPab7Rms).

### 1. CAREFULLY DISASSEMBLE THE IPOD

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



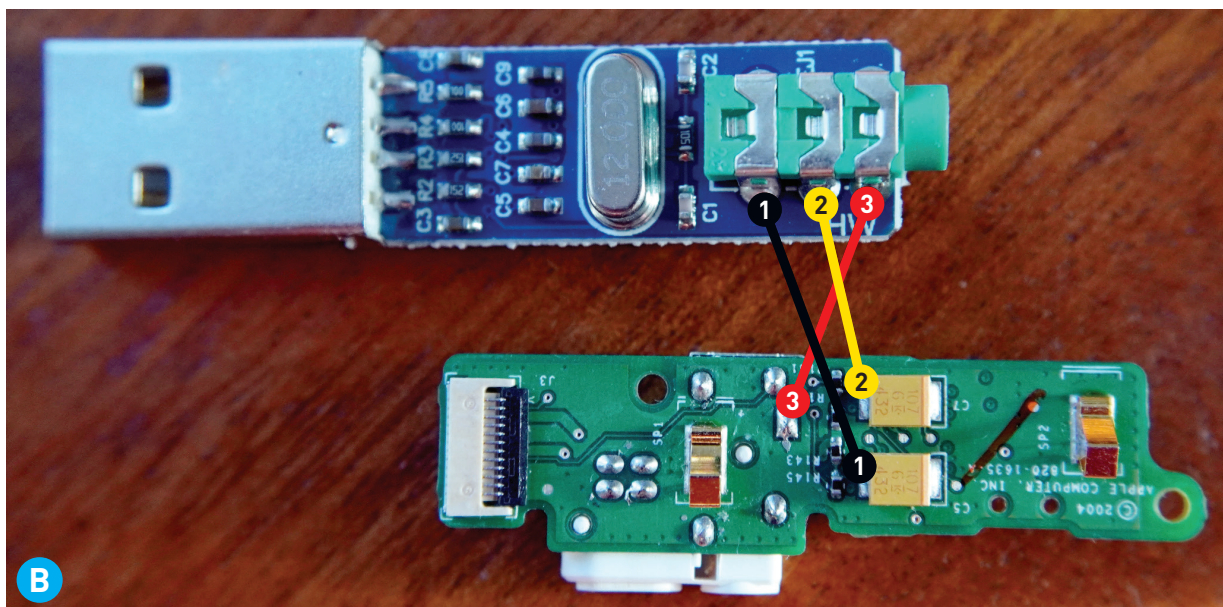


The motherboard, battery, and HDD can be put aside, we won't be using those.

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](https://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.

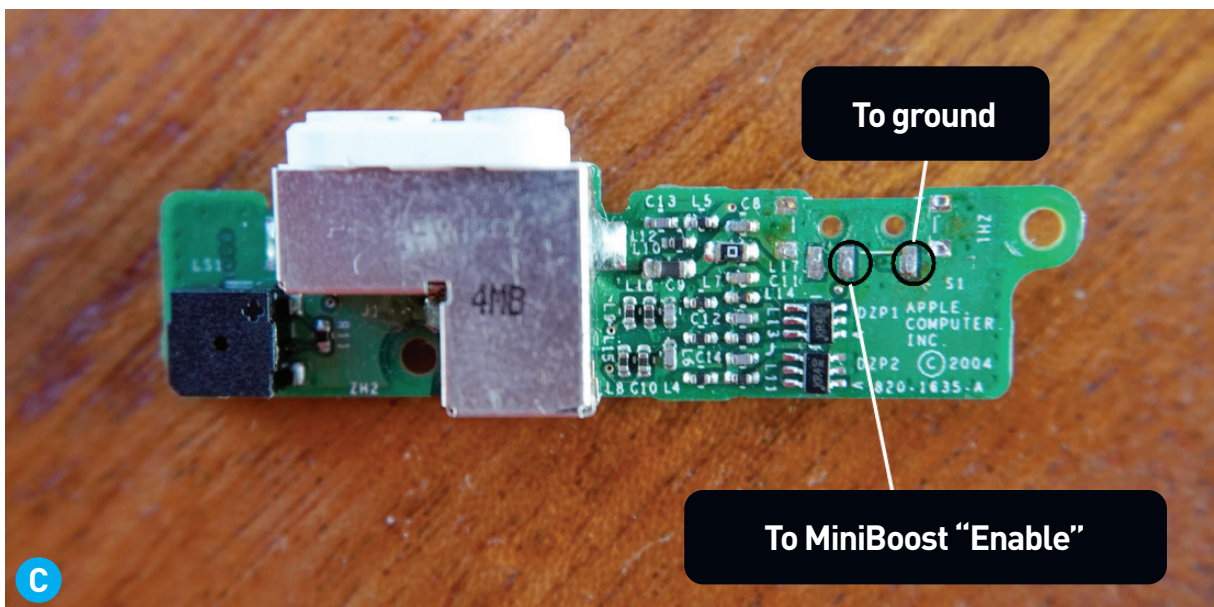
### 8. REATTACH THE HEADPHONE JACK ASSEMBLY

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

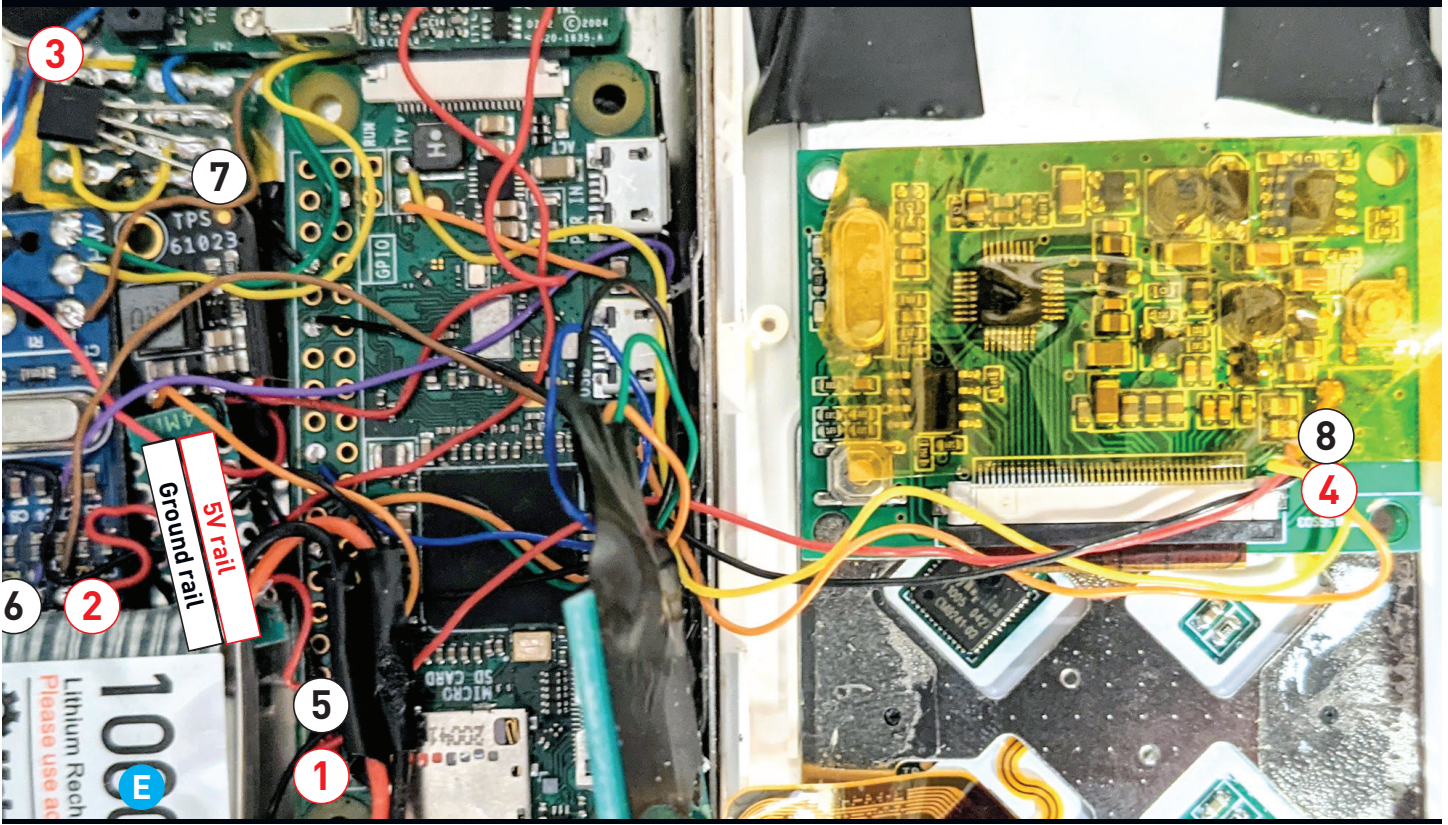
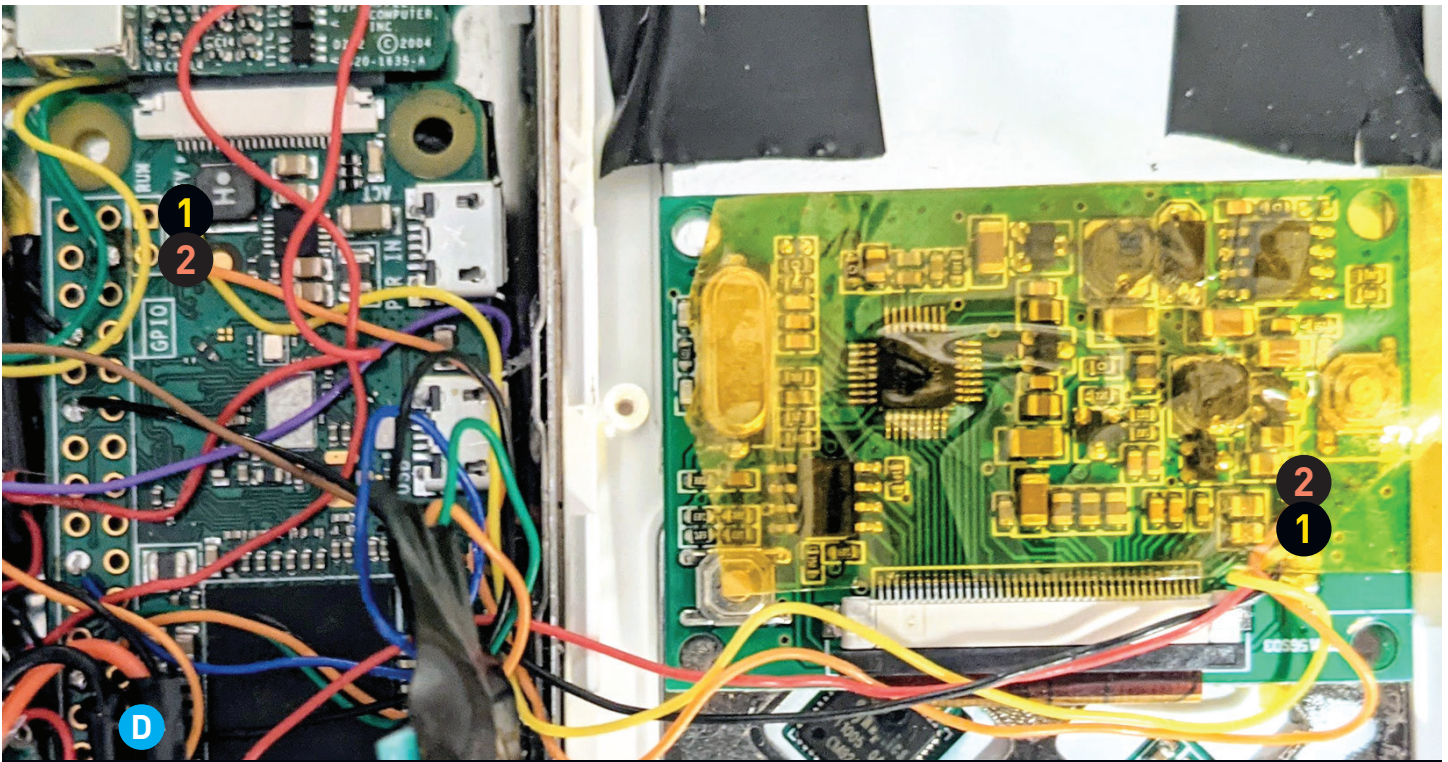
You may have to snip out some of the plastic to make space for your new audio wires.

### 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









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](http://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:

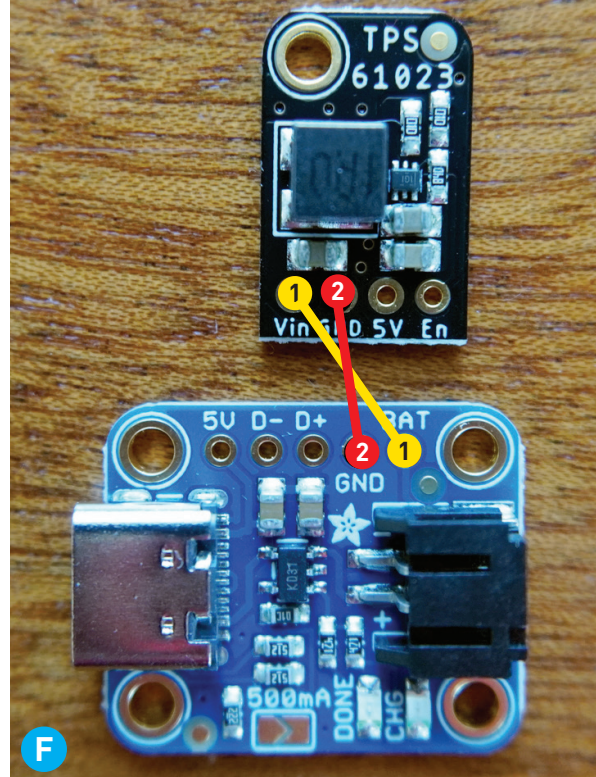
- ⑤ one of the Raspberry Pi's ground pins
- ⑥ the USB DAC
- ⑦ the haptic driver transistor's emitter pin, and
- ⑧ the display.

### 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).

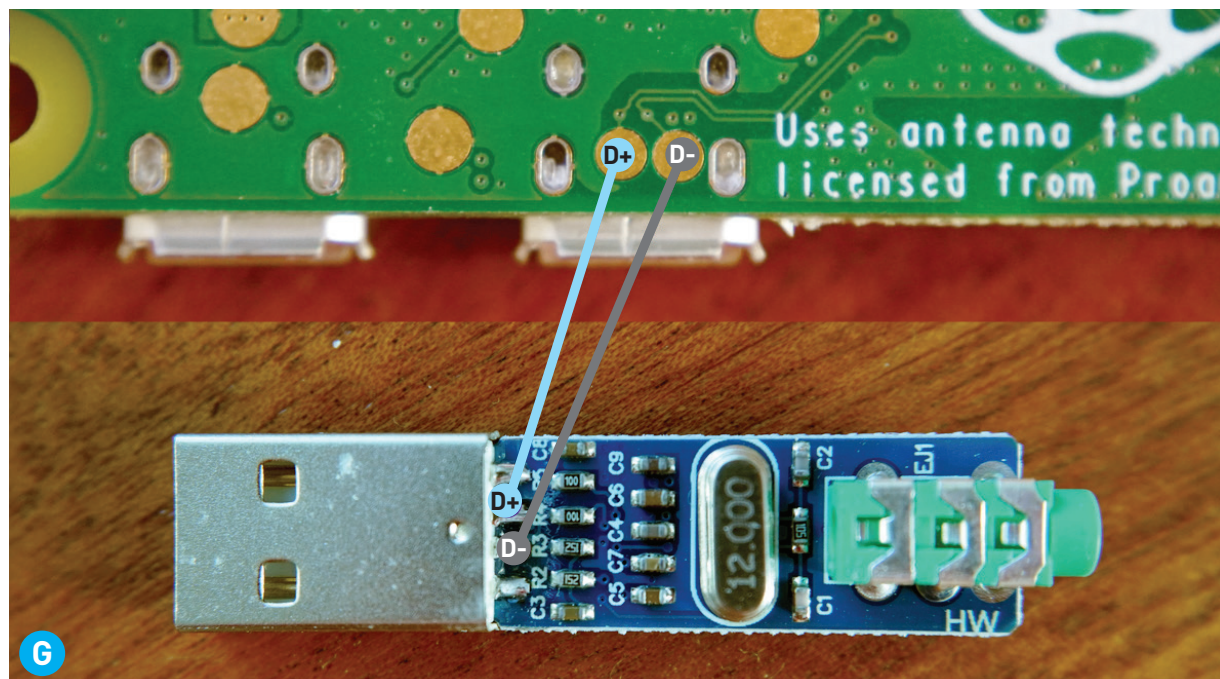


### 13. TEST!

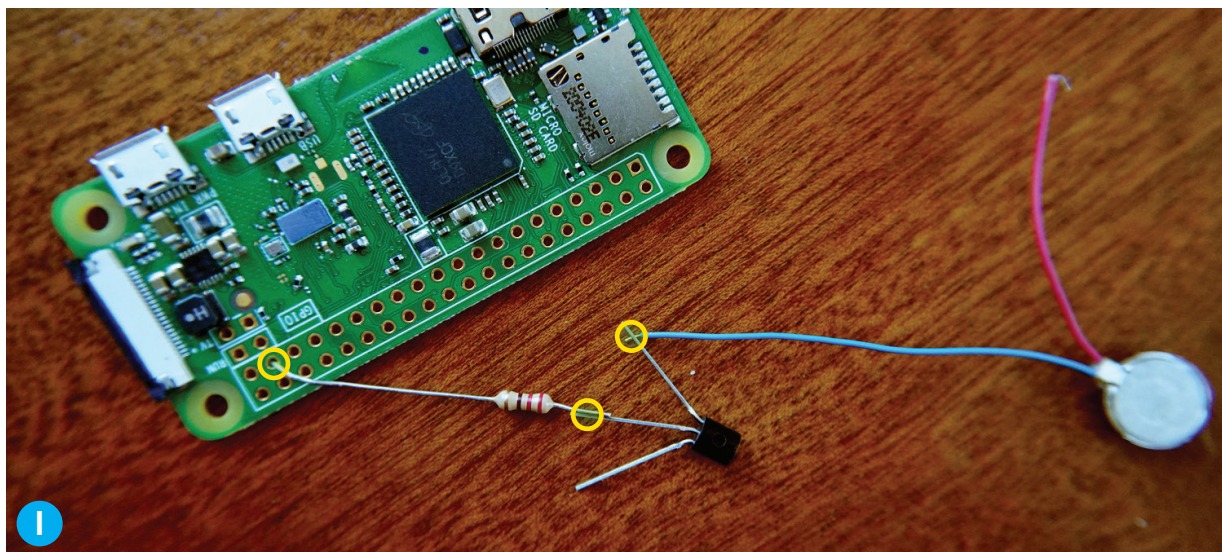
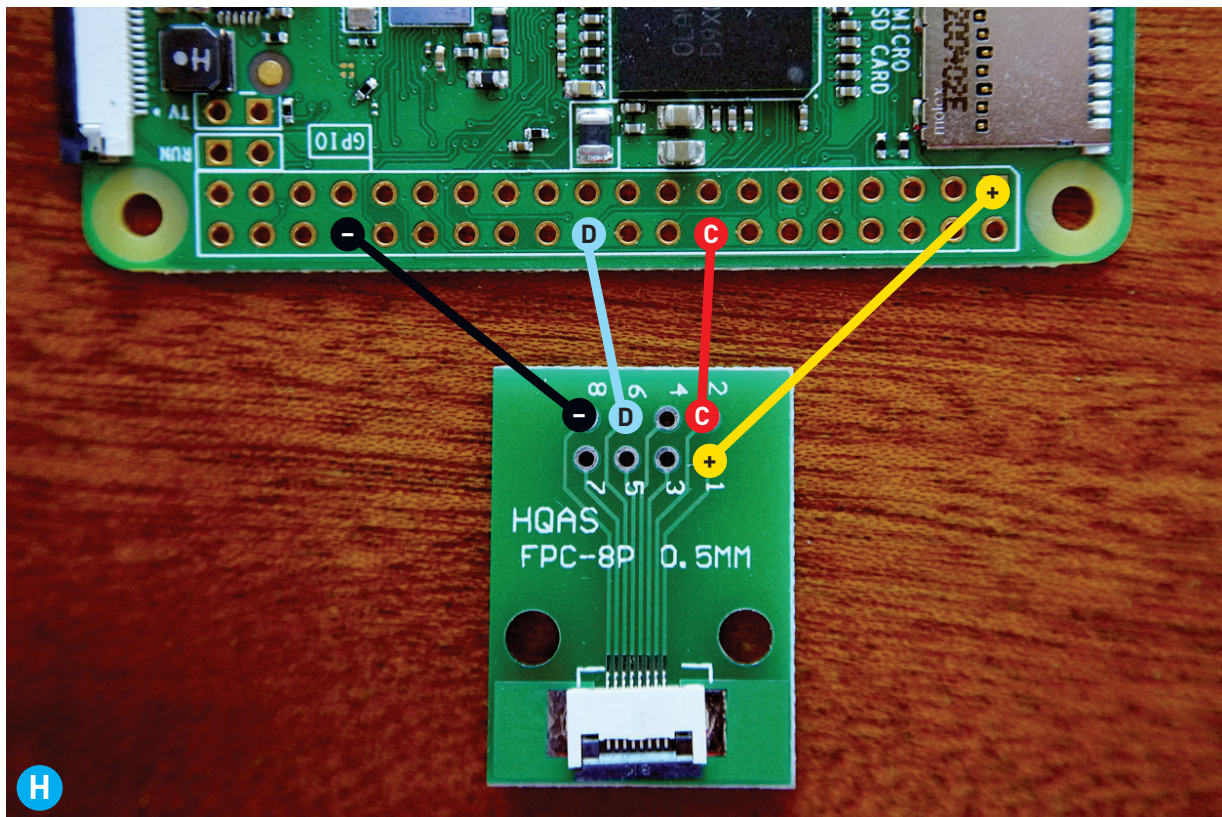
It's a good idea to use a multimeter to make sure you have no continuity between 5V and ground. Then, plug your LiPo battery into the charger board. If your lock switch is open, everything should power on! At this point, you can visually verify the Raspberry Pi and the display. Then, power everything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







### 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](https://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/](https://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](https://rsflightronics.com/spotifypod). 🎧



# Open Smart Cam

Free yourself from the corporate camera cloud with this motion-sensing Raspberry Pi/Arduino mash-up

Written and photographed by Eben Kouao



**EBEN KOUAO** makes DIY projects and prototypes, from delivery robots to electric skateboards and more. Watch his tutorials at [youtube.com/ebenkouao](https://youtube.com/ebenkouao).



**Security cameras can be a great way to monitor and protect your home while you're away.**

However, with the proliferation of cloud-based solutions on the market, you may feel uneasy about the idea of your own video footage being handled by a corporation's cloud service. Moreover, some of their useful features such as recording or screen capture are typically hidden behind a subscription plan. What if you could build your own security camera with a Raspberry Pi, an Arduino, and motion sensors?

In this guide, you'll build a DIY security camera that you have full control of, without the need for a subscription plan. You can include as many features as you want — after all, this is an open-source project!

**INTRODUCING THE PI SMART CAM**

The Pi Smart Cam is an open-source DIY camera designed for you to view live footage from your phone — or any device — remotely (Figure A). It can detect motion and send you alerts, like a Nest security cam, but privately on your local network.

This project builds on my previous “Live Streaming with Raspberry Pi” tutorial in *Make: Volume 76*, [makezine.com/projects/beginner-project-a-remote-viewing-camera-with-raspberry-pi](http://makezine.com/projects/beginner-project-a-remote-viewing-camera-with-raspberry-pi). (If you're new to working with a Raspberry Pi, it would be best to explore the previous tutorial first.) In this guide, we're taking things a step further by entering the world of microcontrollers.

**HOW IT WORKS**

**Pi Camera stream** — The Pi Smart Cam captures live footage from the Raspberry Pi Camera Module and uses Flask (a Python web framework) to create a live stream to any client device connected to the same network (Figure B). Flask provides the “bridge” between Python and an HTML web page, which supports Motion JPEG video — essentially a sequence of independent JPEG images in a video compression format.

**Arduino** — An Arduino microcontroller sends values to the Pi via USB serial connection. With this setup, we can make the Pi trigger an action when the PIR sensor detects motion, such as sending emails, turning lights on, or sounding an alarm.

**TIME REQUIRED:** 4–6 Hours

**DIFFICULTY:** Intermediate

**COST:** \$120–\$150

**MATERIALS**

Check the GitHub repo for the latest parts list: [github.com/EbenKouao/pi-smart-cam](https://github.com/EbenKouao/pi-smart-cam)

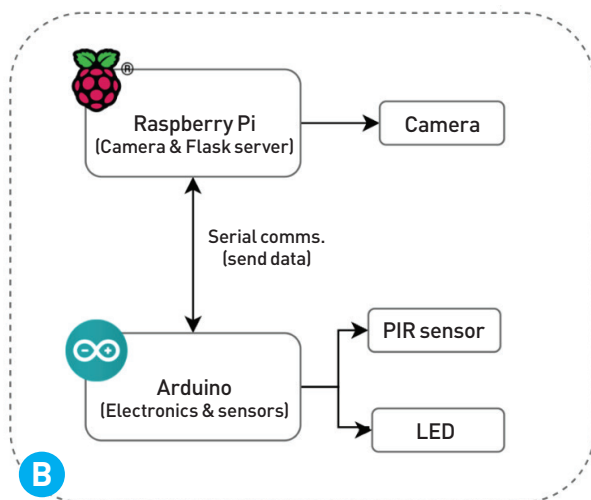
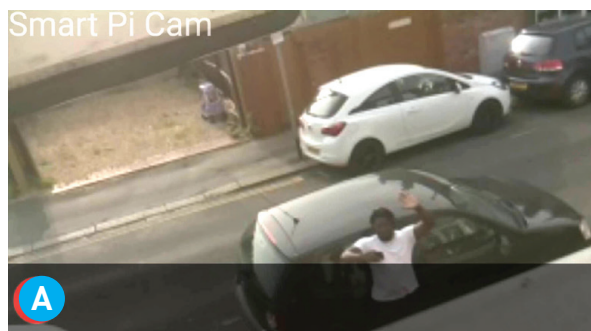
- » Raspberry Pi single-board computer with power supply; model 4B recommended
- » Raspberry Pi Camera Module v2 recommended
- » microSD card, 32GB+
- » Arduino Nano microcontroller board
- » PIR sensor, HC-SR501
- » Breadboard, half size
- » Jumper wires: male/female and male/male
- » Resistor, 220Ω
- » LED
- » Cable, USB-A to Mini-B

For the hardware chassis (optional):

- » 3D-printed parts free 3D files at the GitHub repo
- » Screws, self-tapping, M3×40mm (4)
- » Braided cable sleeve

**TOOLS**

- » 3D printer (optional)
- » Computer with Arduino IDE free from [arduino.cc/downloads](http://arduino.cc/downloads)



**PIR sensors** — Passive infrared sensors are used in a wide array of sensory equipment, from automatic floodlights that turn on when motion is detected, to CCTV cameras. In our case, the HC-SR501 PIR sensor (Figure C) acts as an additional sensory input working with the camera to detect the presence of a human within the sensor's range, even in complete darkness!

## BUILD YOUR PI SMART SECURITY CAMERA

Before you start, check my YouTube channel at [youtube.com/ebenkouao](https://youtube.com/ebenkouao). I've created a video tutorial complementing this article to help you visually build along.

### 1. SET UP YOUR RASPBERRY PI

This project assumes you have already created an image on the microSD card using the latest version of Raspbian, and that your Raspberry Pi is ready to go. For more details, read the previous Live Streaming tutorial referenced above.

#### 1a. Enable Pi Camera module

Plug your camera module into its connector on the Raspberry Pi board. Open a terminal window on the Pi and type:

```
sudo raspi-config
```

To enable the camera, select Interface Options → Camera Port, and select Enabled (Figure D).

To enable VNC Viewer for remote access to the Pi, select Interface Options → VNC, and select Enabled.

Restart your Pi to apply the updates.

#### 1b. Verify camera setup

Once you've rebooted your Pi, verify that your camera works by entering this command into your terminal:

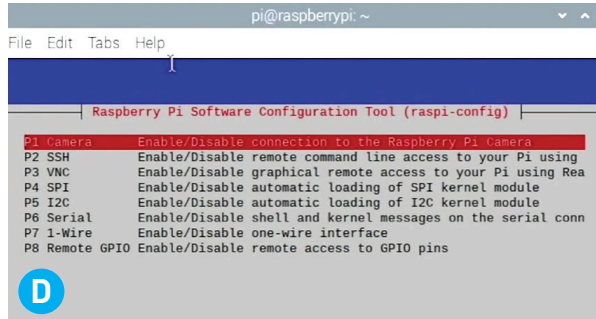
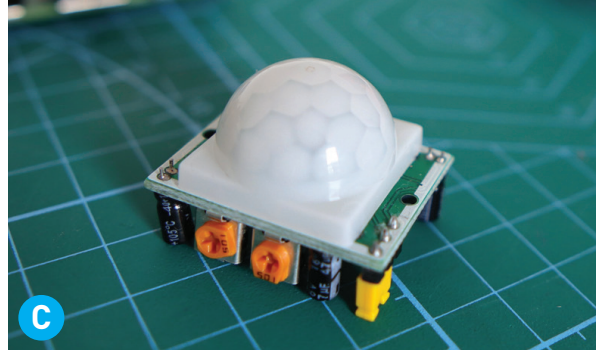
```
raspistill -o ~/Desktop/image.jpg
```

This will take a picture and store *image.jpg* on your desktop.

#### 1c. Install/update packages

In terminal, type:

```
sudo apt-get update
sudo apt-get upgrade
```



### 2. SET UP THE SMART CAM SOFTWARE

#### 2a. Install Pi Smart Cam from GitHub repo

Now that you've updated packages and verified that the camera module is working, you can proceed to install the Pi Smart Cam's dependencies.

Open a terminal and clone the Pi Smart Cam Repo:

```
git clone https://github.com/
EbenKouao/pi-smart-cam.git
```

See the repo's *readme* file for further installation dependencies and notes.

#### 2b. Launch Pi Smart Cam

Once the repo is cloned and dependencies are installed, you can navigate to the *code/examples/pi-camera-stream-pir-sensor* directory, and run the Video Streaming application in the terminal:

```
python3 main.py
```

Now devices connected to the same network can view the Pi camera's live stream (Figure E) by accessing this URL on port 5000: *<your\_raspberry\_pi\_ip>:5000*, e.g. *192.168.0.103:5000*



**NOTE:** You can find your Pi's IP address by entering `ifconfig` into your terminal command and looking for the `inet` address.

### 3. BUILD THE ARDUINO CIRCUIT

Connect the LED and the 220Ω resistor to the Arduino Nano's pin D5 and ground, as shown in Figure F. Connect the PIR sensor to Arduino pins D8, power, and ground, as shown. This sensor helps our Pi Smart Cam detect movements, even in the dark.

### 4. SET UP ARDUINO SOFTWARE

Follow the general Arduino setup guide at [arduino.cc/en/Guide/ArduinoUno](http://arduino.cc/en/Guide/ArduinoUno). Then upload the motion sensor code from the Github repo, *pi-motion-sensor.ino*, to your Arduino board.

The code snippets shown here demonstrate how the Arduino microcontroller receives values from the PIR sensor. The PIR sensor input is set as digital pin 8, or D8 (Figure G).

The `loop()` function (Figure H) runs continuously while the Arduino is powered on. In it,

- The Arduino receives the input readings from the PIR sensor on digital pin 8.
- The current PIR sensor state is stored as `pinStateCurrent`. Where `1` is **HIGH**, and `0` is **LOW**.
- If the PIR sensor detects motion, and the current state is different from the previous state, then the Arduino writes the value of `1` (representing motion detected) to the serial output, which is sent to the Raspberry Pi.

You can verify that you're all set up by viewing the Arduino Serial Monitor. You should be able to hover your hands in front of the sensor and view the value of `1` appearing on the monitor, representing motion detection!

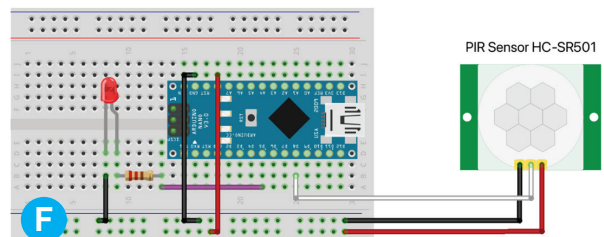
### 5. CONNECT ARDUINO TO RASPBERRY PI

For your smart cam to achieve the scenario of receiving a "motion detected" event and triggering an action such as sending an email, you'll need to connect the Arduino to the Raspberry Pi and send data between them. Plug the USB cable into both the Raspberry Pi USB port and the Arduino's USB B mini port (Figure I).

## How Does a PIR Sensor Work?

It detects heat energy in the surrounding environment. The PIR includes a pair of pyroelectric sensors and a lens to enhance the sensing range. The two sensors are next to each other, and when the signal differential between the two changes (e.g., a person walks into a room) the PIR sensor will change state. (You can learn more at [arrow.com/en/research-and-events/articles/understanding-active-and-passive-infrared-sensors](http://arrow.com/en/research-and-events/articles/understanding-active-and-passive-infrared-sensors).) This low-cost and very practical technology makes it an ideal sensor, especially when the camera is in the dark. No one can hide!

As a result, we can take advantage of the PIR motion sensor to trigger an action when an event has occurred.



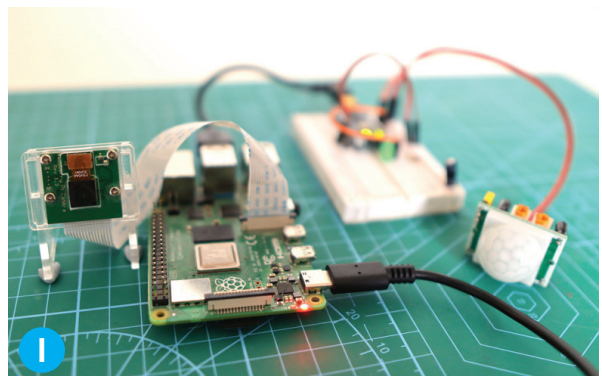
Breadboard: PIR Sensor & LED

```
G int pirSensor = 8; // Arduino Digital Pin 8
```

```
void loop() {
  pinStatePrevious = pinStateCurrent; // store previous state of motion sensor
  pinStateCurrent = digitalRead(pirSensor);

  if (pinStatePrevious == LOW && pinStateCurrent == HIGH){
    Serial.write("1"); // Send message to the Raspberry pi - Motion Detected
    digitalWrite(LedPin, LOW);
  }else if (pinStatePrevious == HIGH && pinStateCurrent == LOW) {
    Serial.println("0"); // Motion stopped - Send message to the Raspberry pi
    digitalWrite(LedPin, HIGH);
  }
}
```

H



```
while True:
    if ser.in_waiting > 0:
        line = ser.readline().decode('utf-8').rstrip()
        print("Time since last motion detected: ", time.time() - current_time)
    if (line == "1"): #if pir sensor value = 1 / HIGH
        detected = True
        print(line)
        # Trigger an action / Do Something!
```

J

```
ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
ser.reset_input_buffer()
```

K

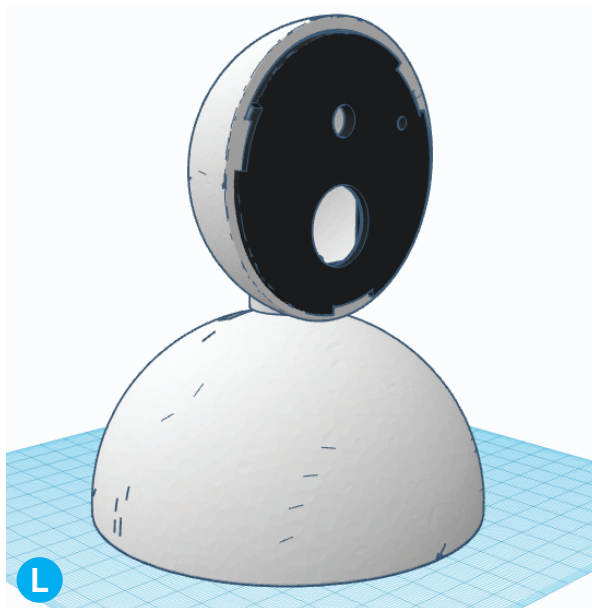
On your Raspberry Pi, the Arduino board should now appear as `/dev/ttyACM0` or `/dev/ttyUSB0`. You can verify this by typing `ls /dev/tty*` into the Pi terminal.

Now run the Pi Smart Cam code, `arduino_comms.py`. This Python application receives data via serial comms from the Arduino (Figure J). The output seen in the terminal prints the value **1**, representing the PIR sensor state being **HIGH**, meaning motion is detected!

Note that the baud rate is set to 9600 (Figure K).

### 6. 3D PRINT THE CAMERA HOUSING

If you have access to a 3D printer, you can optionally build your own camera chassis (Figures L, M, and N) to house the camera, PIR sensor, Arduino, and Raspberry Pi. You can find the 3D parts and assembly details on the GitHub repo. Please feel free to contribute any modifications or accessories to the repo!



L

### SOMEONE'S AT THE DOOR!

Your Pi Smart Cam is ready to keep an eye on things, indoors or out the window (Figures O and P). Now let's set up an action.

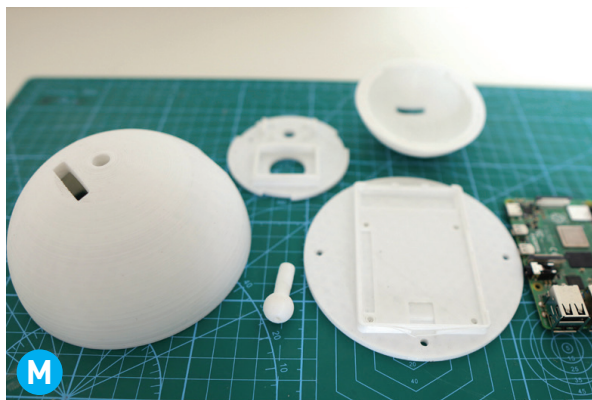
### SENDING A NOTIFICATION VIA EMAIL

In this example, we'll send a notification to our email with a camera frame capturing the exact motion detected (Figure O).

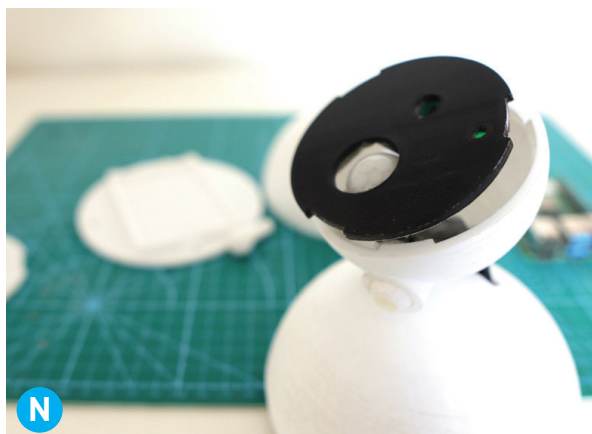
The example code `email_notification.py` (Figure R) sends emails to a target address including the attached camera frame, subject line, and message as part of the body using the SMTP Python library. SMTP is a protocol that handles sending emails between mail servers.

To configure your email notification, the following fields can be found in `main.py`:

```
pi_email = "<from-email>"
pi_app_password = "<app-password>"
pi_port = 465
pi_host = "smtp.gmail.com"
notification_recipient = "<to-email>"
```



M



N

**NOTE:** If you want to send email notifications via Gmail, enable 2FA and use App Passwords instead of storing the password as plain text. Find the latest information on the GitHub repo.



### PERFORMANCE AND SECURITY

The Pi's stream latency depends on your Wi-Fi network performance; expect a latency of 1–2 seconds, and in some cases as low as ~500ms.

Avoid using the camera in compromising spaces, and consider adding credentials/ encryption if you would like to expose your camera stream beyond your local network. (Learn more about cyber security on page 58.)

### MORE FEATURES

If you're interested in adding more features, check out the GitHub repo:

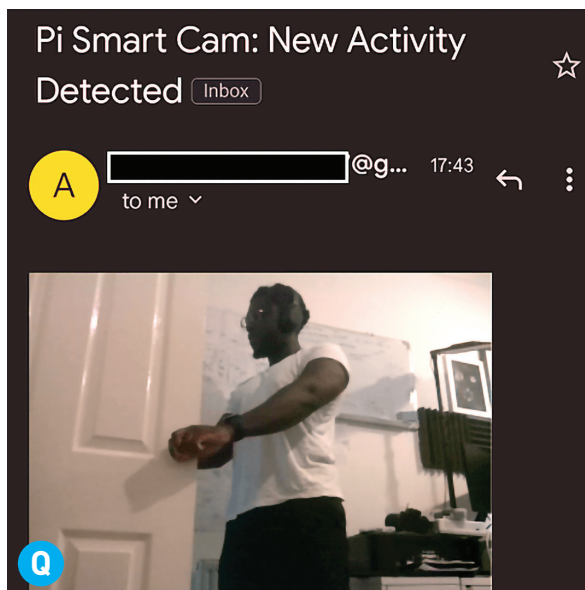
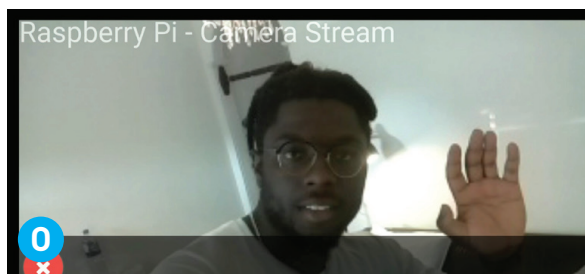
- Ring chime pushbutton — allows a user to press a button to ring and notify the user that someone is at the door.
- LED lights up — as visual feedback when a button is pressed or motion is detected.

### IMPROVEMENTS TO MAKE

Here are some improvements you may want to explore:

- Create a PCB, reducing the form factor further!
- Upgrade your camera module to use an HD camera/webcam.
- Build your own dedicated server using SSH tunnelling and view your stream from outside the home.
- Add face detection!

Whether you're a web developer, 3D printing wizard, or Pi enthusiast, I hope you'll consider contributing to this project. The option for an open-source camera stream with motion detection and picture capture capabilities provides a DIY alternative to the tightly coupled Big Tech security systems. With your Pi Smart Cam, you're in full control of your data. ☞



```
# only send email notification if motion is detected after X seconds
if(int(time.time() - current_time) > sensitivity_timer):

    current_time = time.time()
    print(line) # print output from Arduino Comms
    if(detected == True):
        detected = False
        take_picture(pi_email, pi_app_password, pi_port, pi_host, frame)
        print("email sent")
```



# Photo Booth

Maker: Kevin Osborn, Justin Shaw, Jenny Ching • [makezine.com/projects/raspberry-pi-photo-booth](http://makezine.com/projects/raspberry-pi-photo-booth)



Raspberry Pi and Wi-Fi! Use a Pi3 to make a touchscreen **photo booth** that instantly uploads to Google Photos. Go to [makezine.com/projects/raspberry-pi-photo-booth](http://makezine.com/projects/raspberry-pi-photo-booth) for the full build.



# Donkey Cars

Maker: Adam Conway & William Roscoe • [makezine.com/projects/build-autonomous-rc-car-raspberry-pi](https://makezine.com/projects/build-autonomous-rc-car-raspberry-pi)



The **Donkey autonomous car** is a very simple car. It's based on a Raspberry Pi computer, a camera, and a servo shield (or "hat") board to interface with the R/C car. Go to [makezine.com/projects/build-autonomous-rc-car-raspberry-pi](https://makezine.com/projects/build-autonomous-rc-car-raspberry-pi) for the full build.



# Weather Data

Maker: John M. Wargo • [makezine.com/projects/raspberry-pi-weather-station-mount](http://makezine.com/projects/raspberry-pi-weather-station-mount)



Use the Astro Pi's Sense HAT sensor board for the Raspberry Pi computer to easily build your own **Pi-based weather station** using off-the-shelf parts. Go to [makezine.com/projects/raspberry-pi-weather-station-mount](http://makezine.com/projects/raspberry-pi-weather-station-mount) for the full build.

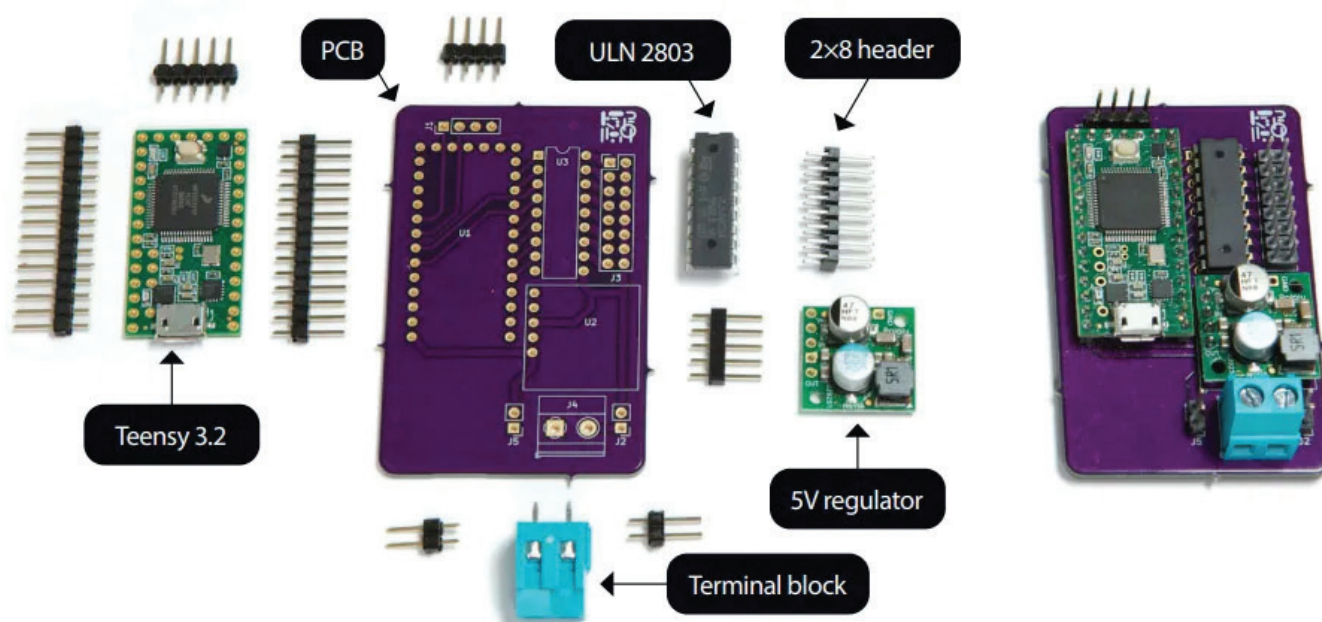


# Dr. Squiggles

Maker: Michael Krzyzaniak • [makezine.com/projects/dr-squiggles-an-ai-rhythm-robot](http://makezine.com/projects/dr-squiggles-an-ai-rhythm-robot)



Build a **musical drumming robot** that plays rhythms by tapping a surface. It can listen to you through its microphone, synchronize to your beat, imitate you, or learn from you! Go to [makezine.com/projects/dr-squiggles-an-ai-rhythm-robot](http://makezine.com/projects/dr-squiggles-an-ai-rhythm-robot) for the full build.





# Data Presever

Maker: Matt Reed • [makezine.com/projects/preserve-data-mason-jar-using-raspberry-pi](http://makezine.com/projects/preserve-data-mason-jar-using-raspberry-pi)

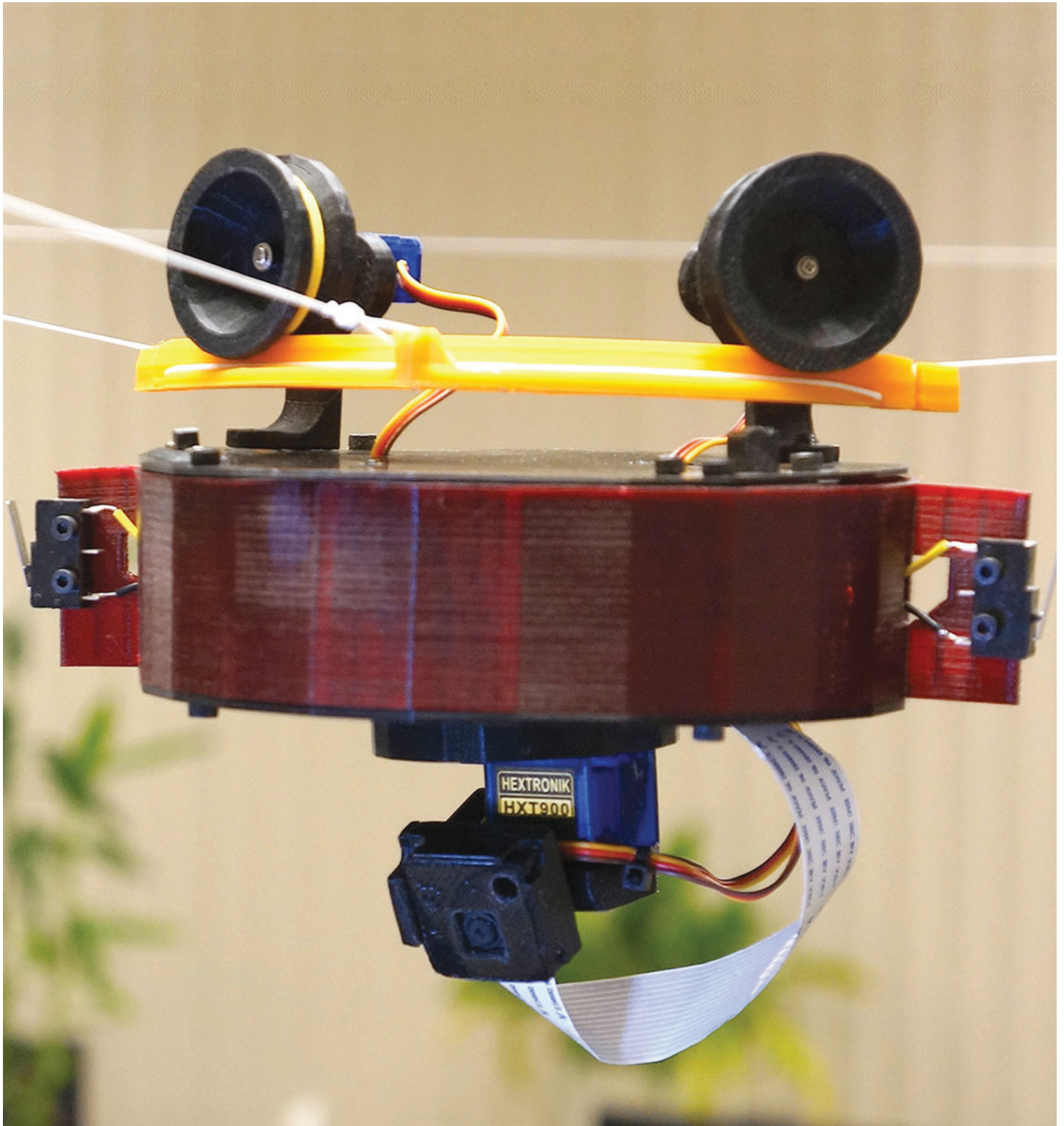


Build a Raspberry Pi **preserve jar to protect your data** with synced backups from your computers and phones that fits perfectly with in a mason jar. Go to [makezine.com/projects/preserve-data-mason-jar-using-raspberry-pi](http://makezine.com/projects/preserve-data-mason-jar-using-raspberry-pi) for the full build.



# Skycam

Maker: Brook Drum • [makezine.com/projects/3d-printed-raspberry-pi-skycam](http://makezine.com/projects/3d-printed-raspberry-pi-skycam)



Build a **Skycam robot** that travels around on a rope or string, can turn corners, takes video, stream live video, and can be controlled from your phone or any browser. Go to [makezine.com/projects/3d-printed-raspberry-pi-skycam](http://makezine.com/projects/3d-printed-raspberry-pi-skycam) for the full build.

# Pi Vids



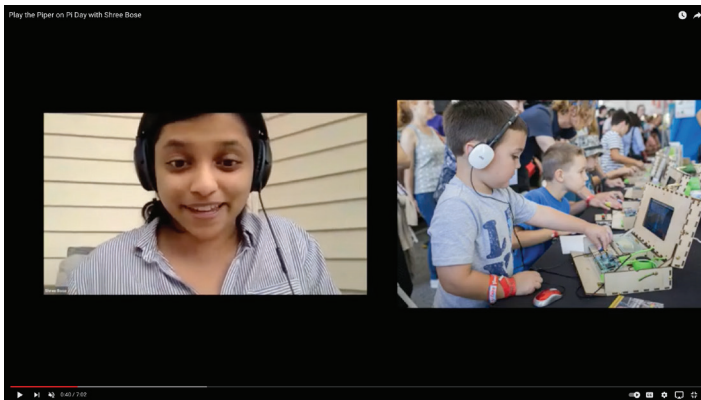
## How to Make a Remote Viewable Camera With Raspberry Pi (Beginner Project)

[youtu.be/zfBHD4v8hD0](https://youtu.be/zfBHD4v8hD0)



## How to Make a DIY Ring Doorbell — Remote Front Door Camera and Lock With Raspberry Pi

[youtu.be/9bJFWlVm\\_Fo](https://youtu.be/9bJFWlVm_Fo)



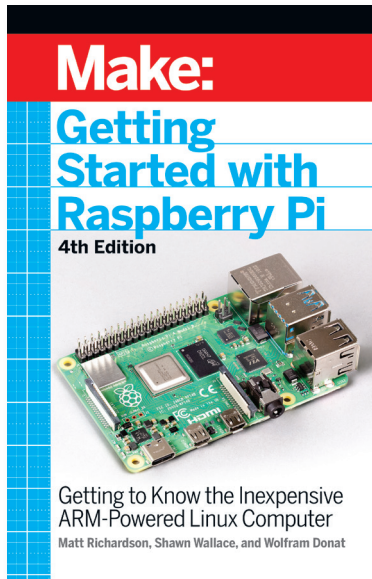
## Play the Piper on Pi Day with Shree Bose

On Pi Day 2021, Piper released a new product, Piper Make, a browser-based coding platform based on the Raspberry Pi Pico and a block-programming interface based on Google Blockly.

[youtu.be/tkpEeSqQsxs](https://youtu.be/tkpEeSqQsxs)

You can check out more *Make:* video content on our YouTube channel [youtube.com/MAKE](https://youtube.com/MAKE)



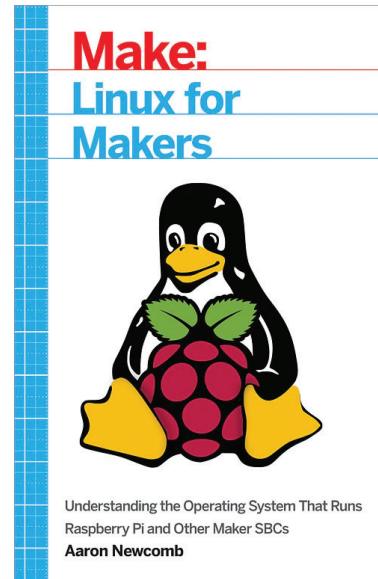


## Getting Started

Packed with a whole new arsenal of Pi possibilities, this 4th edition guides you on an adventure into the exciting world of Raspberry Pi models 4, Pico & Zero 2 W!

[makershed.com/collections/raspberry-pi/products/make-getting-started-with-raspberry-pi-4th-edition-print](https://makershed.com/collections/raspberry-pi/products/make-getting-started-with-raspberry-pi-4th-edition-print)

Price: \$24.99



## Linux for Makers

Linux is the ultimate operating system for Makers, and this book provides a strong foundation in its key principles. Grab it to unlock your projects' potential!

[makershed.com/collections/raspberry-pi/products/make-linux-for-makers](https://makershed.com/collections/raspberry-pi/products/make-linux-for-makers)

Price: \$24.99

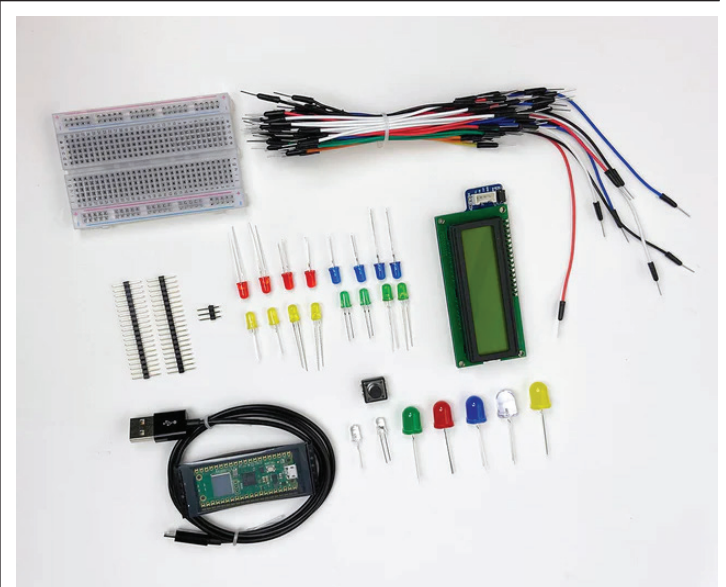
**Getting Started With Raspberry Pi, 3rd Edition** • JUMPSTARTING THE RASPBERRY PI ZERO W • **Make a Raspberry Pi-Controlled Robot** • RASPBERRY PI AND AVR PROJECTS • **Linux for Makers** • MAKING THINGS TALK, 3E • **Make: Action** • MAKE: BLUETOOTH • **Make: Sensors** • GETTING STARTED WITH SENSORS • **Make: magazine volumes 33, 35, 36, 38, 43, 44, 49, 51, 52, 53, 55, 57, 63, 68, 71 and 72**

## Pi-brary eCollection

Check out the Make: Pi-brary collection! With over 25 of the best Raspberry Pi eBooks & eMagazines, you can go beyond basic and explore new possibilities.

[makershed.com/collections/raspberry-pi/products/make-pi-brary-master-ecollection](https://makershed.com/collections/raspberry-pi/products/make-pi-brary-master-ecollection)

Price: \$49.95



## Pi Pico Pack

Take the plunge into tech-tastic Raspberry Pi fun with this delectable, ready-to-use bundle. Ideal for all makers itching to dive right in and explore their creativity!

[makershed.com/collections/raspberry-pi/products/make-pico-pack](https://makershed.com/collections/raspberry-pi/products/make-pico-pack)

Price: \$24.95

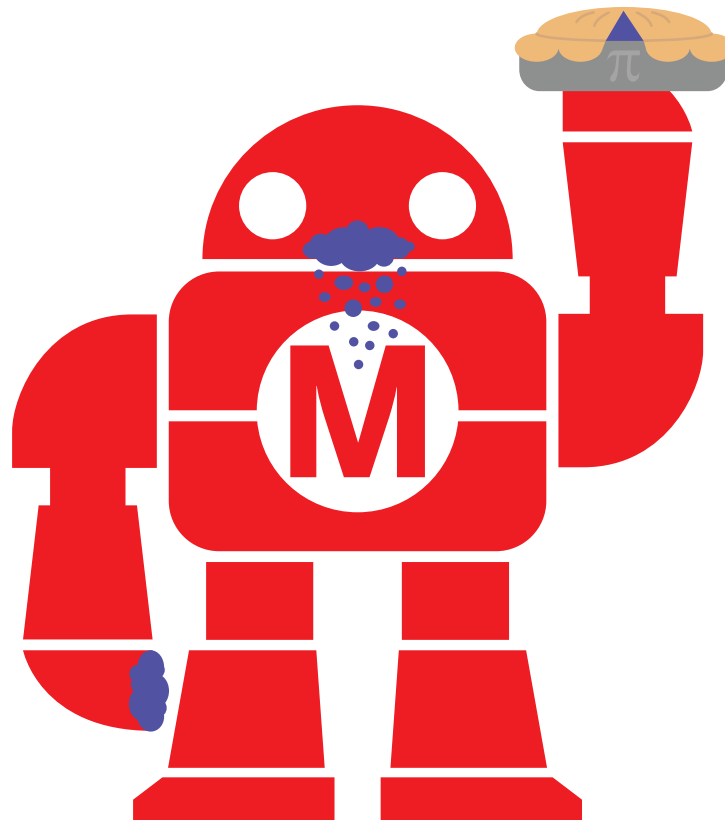


## Make: Pi Day T-Shirt

I like Pi, You like Pi, Makey LOVES Pi!

[makershed.com/collections/raspberry-pi/products/make-getting-started-with-raspberry-pi-4th-edition-print](https://makershed.com/collections/raspberry-pi/products/make-getting-started-with-raspberry-pi-4th-edition-print)

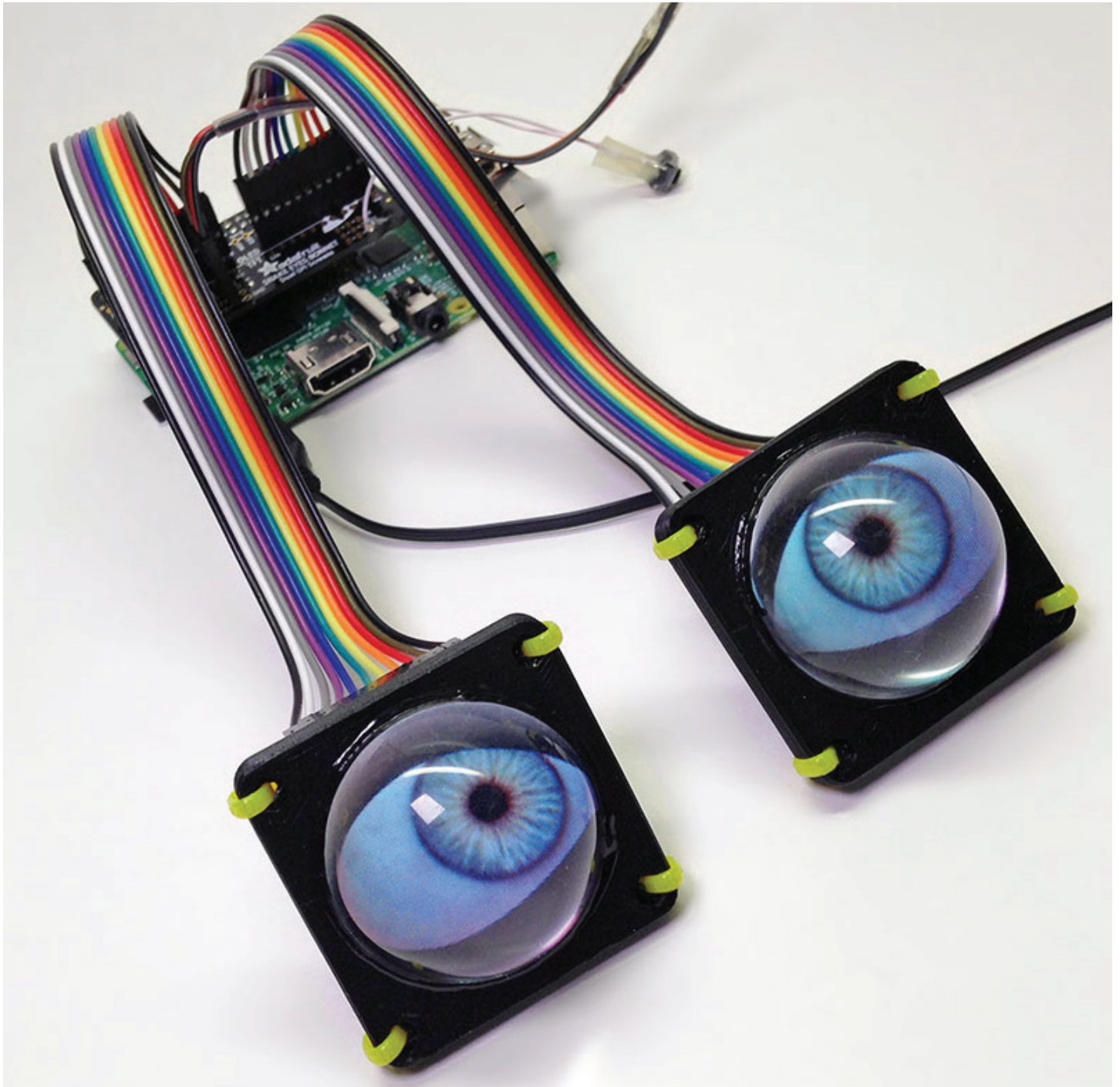
Price: \$25 Limited Run — Don't Miss Out!





# Eye Candy

Maker: Phil Burgess • [makerfaire.com/maker/entry/65205](http://makerfaire.com/maker/entry/65205)



**Sundry electric orbs** that move and blink but definitely do not have souls trapped by arcane ritual. Computer graphics and tiny displays replace noisy, bulky animatronics.

# Sisyphus

Maker: Bruce Shapiro • [makezine.com/article/craft/gorgeous-kinetic-table-art-bruce-shapiro](http://makezine.com/article/craft/gorgeous-kinetic-table-art-bruce-shapiro)



**Sisyphus** is a series of kinetic art sand tables powered by a computer-controlled magnet beneath the table that drives a steel ball around in patterns.



# Reef-Pi

Maker: Ranjib Dey

[makezine.com/article/technology/raspberry-pi/automate-coral-reef-tank-raspberry-pi](http://makezine.com/article/technology/raspberry-pi/automate-coral-reef-tank-raspberry-pi) • [makerfaire.com/maker/entry/60569](http://makerfaire.com/maker/entry/60569)

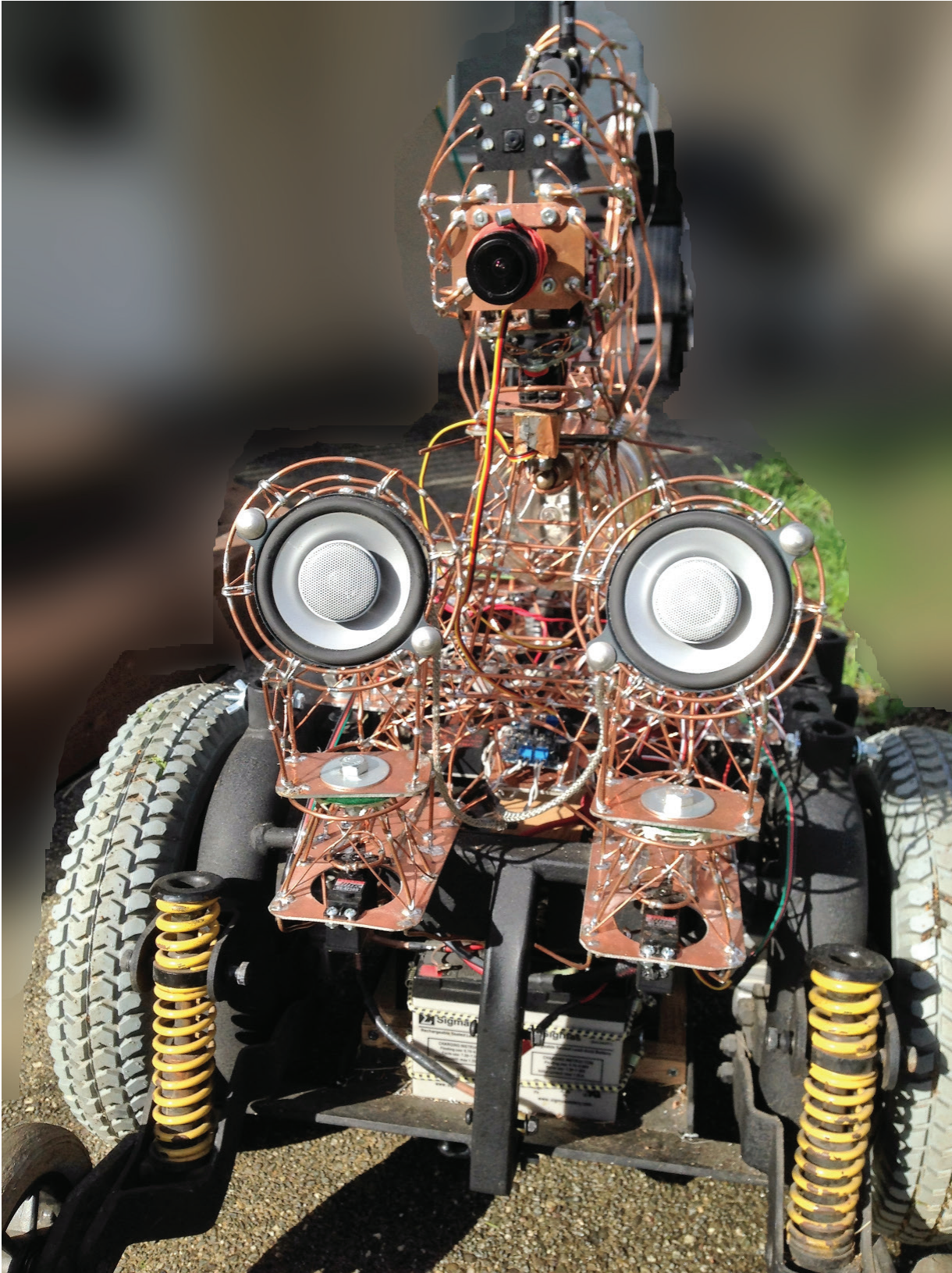


**Reef-pi** is an open source, open hardware-based reef tank controller. It allows automating day-to-day reef-keeping chores using Raspberry Pi.



# Copper Bot

Maker: Yolo Maker Group • [makerfaire.com/maker/entry/60067](http://makerfaire.com/maker/entry/60067)



The **CopperBot** project is a platform to explore the Raspberry Pi, Arduino, LED lights, sculpture, remote control as well as use up a few large radio tubes.



# Vintage Film Restoration

Maker: Joe Herman • [makerfaire.com/maker/entry/63189](http://makerfaire.com/maker/entry/63189)



Transfer old home movies to HD video, with professional-level quality using a **modified projector, and a Raspberry Pi & Camera**. Rescue those priceless memories!



# Tap Glo

Maker: Ben Hencke • [makerfaire.com/maker/entry/62950](http://makerfaire.com/maker/entry/62950)

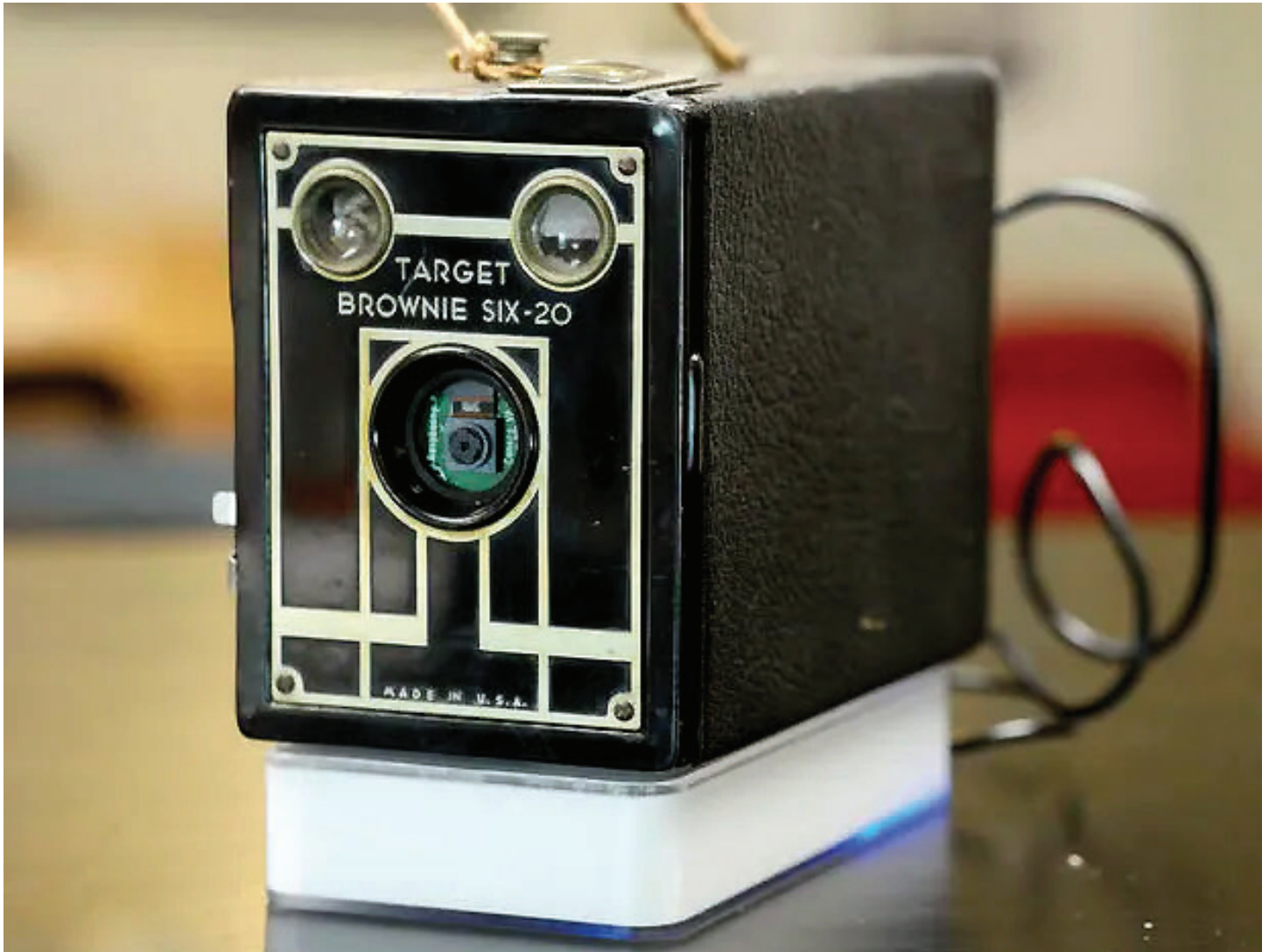


**TapGlo** is an interactive illuminated ping pong table comprised of lit up squares that black out each time you hit your opponents square.



# Tumbler Pi Camera

Maker: Becky Stern • [makezine.com/article/maker-news/becky-sterns-retro-raspberry-pi-tumblr-gif-camera](http://makezine.com/article/maker-news/becky-sterns-retro-raspberry-pi-tumblr-gif-camera)



This Raspberry Pi project by Becky Stern puts a clever animated **GIF-making apparatus** into a **vintage camera** housing and uploads the results to a Tumblr page.