



BUILD MANUAL · VERSION 1.0

SUPER 8 DIGITAL CARTRIDGE

A digital film cartridge you can print at home. It drops into the film chamber of a vintage Super 8 camera and records digitally, without modifying the camera in any way.

COMPUTER

Raspberry Pi Zero 2 W

SENSOR

OV5647 · 1296 × 972

TRIGGER

LDR on GPIO 4

LICENCE

MIT · open source



BEFORE YOU START

FIVE PARTS TO PRINT, FIVE THINGS TO BUY

Everything printable fits on one plate, so it is a single print job. Nothing needs supports, and there are no screws or fasteners anywhere in the design.

PRINT THESE

Cartridge body	Houses everything
Motor wheel	Two crossed light tunnels
Motor closer	Keeps the wheel seated
Sensor holder	Sets the lens plane
Cartridge cap	Closes the shell

About 40 g of filament. Black PLA suits it, but anything opaque works as long as the motor wheel comes off the bed clean enough to spin freely.

BUY THESE

Raspberry Pi Zero 2 W	With header pins
Waveshare OV5647	Plus ribbon cable
128 GB MicroSD	Class 10 / A1 or better
Lipo Rider Plus	Charger and 5 V boost
3.7 V 2600 mAh LiPo	Roughly 5 hours of use

Plus a white LED, an LDR, a 330 Ω and a 10 k Ω resistor, a small SPDT slide switch, and four jumper wires.

CAPTURE

Resolution	1296 × 972, 4:3, binned two-by-two from 2592 × 1944
Frame rate	17 fps, matched to the measured motor rate of this camera
Bitrate	50 Mbps H.264, keyframe every half second
Exposure	58 ms fixed, exactly one motor cycle, which collapses shutter banding to about 1%
Card capacity	Roughly five hours on 128 GB, at about 375 MB a minute

THE IDEA

IT LISTENS FOR THE SHUTTER

Nothing connects the cartridge to the camera, so it has to work out for itself when to record. A printed wheel sits where the film sprocket used to bite, and the camera's own motor turns it whenever you pull the trigger.

That wheel has two tunnels bored through it at right angles to each other, forming a cross with four openings around the rim. A white LED shines across the wheel and an LDR watches from the other side, so every quarter turn an arm of the cross lines up with the beam and light reaches the LDR, then the solid part of the wheel blocks it again. Every one of those changes is an edge on GPIO 4, and a steady stream of edges is what tells the Pi that the camera is running. Two seconds of silence and it assumes you have let go, so it closes the clip.

WIRING · FOUR CONNECTIONS

1

3V3

Feeds the LED through 330 Ω and the LDR through a 10 k Ω divider.

2

5V

From the Lipo Rider Plus boost output.

6

GND

Common ground, shared with the booster.

7

GPIO 4

The LDR junction. This is the trigger signal.

Get the threshold right. GPIO 4 needs to climb past 1.8 V when the light gets through, and drop below 0.8 V when a tunnel blocks it. If bright never reaches 1.8 V, the light tunnel is misaligned or the LED is too dim. If dark never falls below 0.8 V, move up to a 22 k Ω or 47 k Ω pull-down.

Recording never waits for the network. The WiFi decision runs in parallel, so a clip taken five seconds into boot still lands on the card. You just cannot browse it until the radio settles.

THREE SERVICES, RUNNING AT BOOT

`cam.py`

Watches GPIO 4 and writes H.264 straight to the card as `S8_2026_0001.mp4`. The counter survives reboots and restarts each year.

`s8serve.py`

Serves every clip on a small web page at port 8080, installable to your phone's home screen as a PWA.

`s8wifi.sh`

Looks for your home WiFi for fifteen seconds, and becomes its own hotspot if it cannot find it.

BUILD

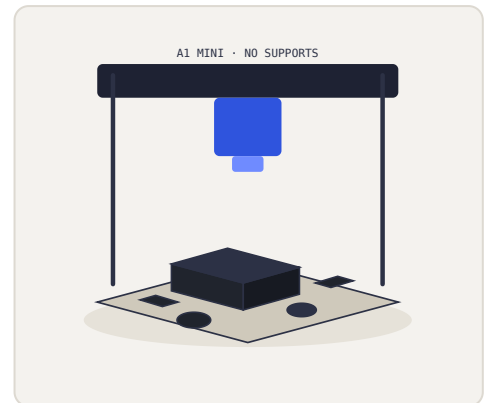
FROM A BLANK CARD TO YOUR FIRST ROLL

Seven steps, none of them difficult. The only long one is the install, and most of that is waiting for apt.

01

PRINT THE PARTS

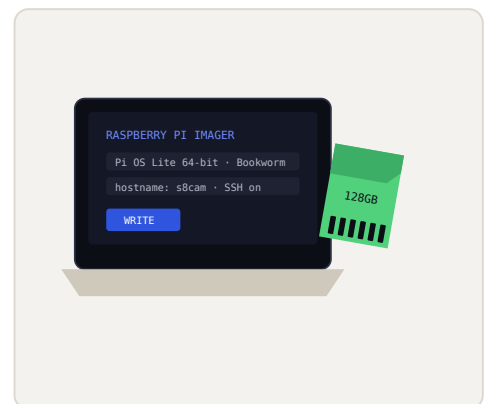
Five pieces, no supports, about forty grams of filament. Download the plate and send it straight to the printer. The one part worth checking afterwards is the motor wheel: it has to spin freely, so clean up any elephant's foot on the first layer before you assemble anything.



02

FLASH THE CARD

Use Raspberry Pi OS Lite 64-bit, and make sure it is Bookworm rather than Trixie, because Trixie has known problems with this camera. In the Imager's settings, set a hostname (`s8cam` works well), enable SSH, enter your WiFi details and pick a username you will remember. Choose the US keyboard layout, since the installer sets a matching locale.

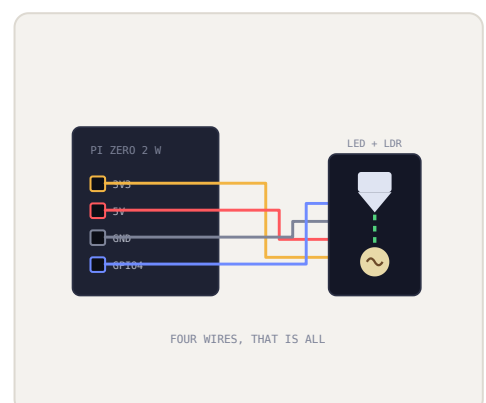


03

WIRE IT UP

Four connections and that is the whole loom: 3V3, 5V, ground and GPIO 4. Leave the Lipo Rider's own switch on and use the external EN switch as your power toggle instead, so you never have to open the cartridge to turn it off.

Blue side of the ribbon faces the board on a Pi Zero. Getting this backwards is the single most common reason the camera is not detected later.



04

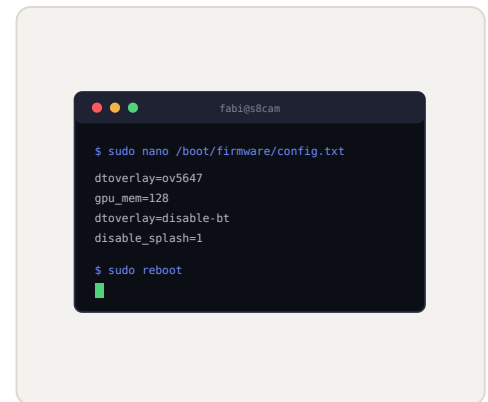
SET UP THE CAMERA

SSH in and paste this whole block. It enables the OV5647 overlay, gives the GPU 128 MB, turns off Bluetooth and the activity LED, and trims a few seconds off the boot. Then it reboots.

```
# on the Pi
sudo tee -a /boot/firmware/config.txt > /dev/null << 'EOF'

dtoverlay=ov5647
gpu_mem=128
dtoverlay=disable-bt
dtparam=act_led_trigger=none
hdmi_blanking=2
disable_splash=1
boot_delay=0
EOF

sudo reboot
```



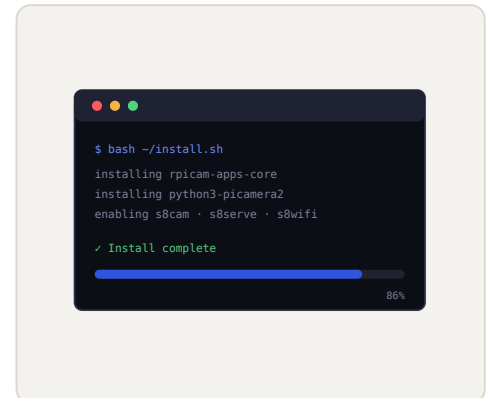
05

COPY THE FILES AND RUN THE INSTALLER

From the `code` folder on your computer, copy everything across and run the installer. It installs the camera stack, disables a dozen boot services the cartridge has no use for, drops in the hotspot configuration and enables the three services. It works out your username by itself, so there is nothing to edit first.

```
# from the code/ folder, replacing USER and HOST with your own
scp *.py *.sh *.service *.conf *.png USER@HOST.local:~/
ssh USER@HOST.local "bash ~/install.sh"
```

Do not reboot yet. The installer enables the services but leaves them stopped, which gives you a clean window to run the diagnostic in step six.

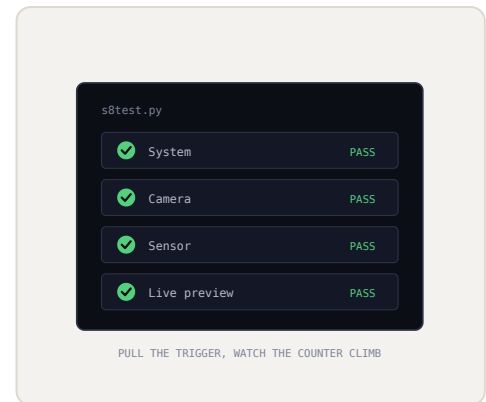


06

CHECK IT BEFORE YOU CLOSE IT UP

Running `python3 ~/s8test.py` walks through four checks: the system, the camera, the sensor and a live preview. The one that matters most is the sensor test. Pull the trigger while it is running and watch the edge counter climb, because that is the LDR seeing the beam get chopped. If the number stays put, sort the light tunnel out now rather than after the cap is on.

```
python3 ~/s8test.py          # all four checks
python3 ~/s8test.py --sensor  # edge counter only
python3 ~/s8test.py --preview # live H.264 preview
```

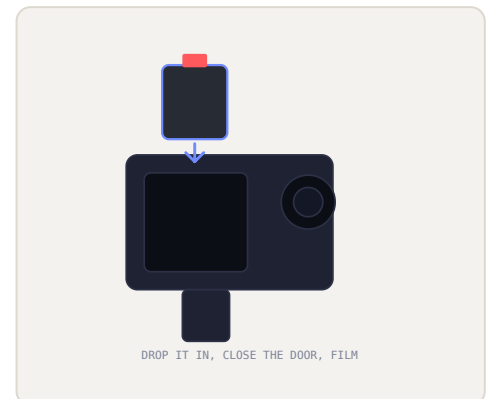


07

LOAD IT AND FILM

Reboot, give it twenty seconds, then drop the cartridge into the film chamber and close the door. Pull the trigger and it records. Your clips will be waiting in a browser on your phone, either at your hostname on port 8080 or, if there was no WiFi to join, on the cartridge's own hotspot.

Always shut the Pi down before cutting the power. Pulling the plug mid-write can corrupt the card. `sudo poweroff`, wait for the light to settle, then flip the switch.



TROUBLESHOOTING

WHEN SOMETHING IS NOT RIGHT

SYMPTOM

WHAT TO DO

Camera not detected after the install

Almost always the ribbon. Blue side faces the board on a Pi Zero, and both ends need re-seating. Then check with `rpigcam-hello --list-cameras`.

Nothing records when you pull the trigger

GPIO 4 is not crossing its threshold. Run `python3 ~/s8test.py --sensor` and watch the counter. Aim a torch at the LDR with the cartridge open and measure the junction: you want above 1.8 V bright and below 0.8 V dark. If dark stays high, fit a 22 kΩ or 47 kΩ pull-down.

The hostname will not resolve

Usually Avahi. `ping -c1 HOST.local`, and if that fails, SSH in by IP and run `sudo systemctl enable --now avahi-daemon.service`.

The hotspot never appears

Check `journalctl -u s8wifi.service -n 30`. Typically hostapd or dnsmasq did not install, so re-run `bash ~/install.sh`.

Port already in use after a glitch

`sudo fuser -k 8080/tcp 8888/tcp` then restart `s8cam` and `s8serve`.

The card is filling up

Roughly 375 MB a minute, so a 128 GB card holds about five hours. Copy the clips off with scp, then delete both the mp4 files and any orphaned h264 files.

Footage is too bright or too dark

Adjust the camera's own iris, or add an ND filter at the gate. Prefer that over changing the exposure value, which is chosen to cancel shutter banding rather than to set brightness.

TUNING

EVERYTHING YOU ARE LIKELY TO CHANGE

It all lives in one file, `s8common.py`. Edit it, copy it back to the Pi, and restart the affected service.

SETTING

WHAT IT DOES

<code>FPS</code> · 17	Match this to your own camera's measured motor rate. The diagnostic's edge counter is the easiest way to work out what that is.
<code>SHUTTER_EXPOSURE_US</code> · 58000	One full motor cycle. Change it together with FPS, or shutter banding comes back.
<code>BITRATE</code> · 50 Mbps	Drop to 35 if your card stutters.
<code>WIDTH</code> , <code>HEIGHT</code> · 1296 × 972	Full lens field of view, binned two-by-two from the sensor's native 2592 × 1944.
<code>SENSOR_GAIN</code> · 1.0	Lowest noise. Raise to 4–8 only if the camera's iris cannot let in enough light.
<code>STOP_DELAY</code> · 2.0 s	How long the sensor stays quiet before a clip is closed.
<code>SENSOR_PIN</code> · 4	The GPIO the LDR is wired to.
<code>MEDIA_PORT</code> · 8080	Where the gallery is served.

```
# after editing, copy it back and restart
scp s8common.py USER@HOST.local:~/
ssh USER@HOST.local "sudo systemctl restart s8cam.service s8serve.service"
```

Released under the MIT licence. Fork it, change it, make it better, and please do share what you build.

