LOCAL VOICE ASSISTANT · BUILD LOG

Freeing the Echo Dot

How we took a 2016 Amazon Echo Dot off Amazon's cloud and turned it into a fully local Home Assistant voice satellite, in one evening, from a Windows laptop.

Booting Fastboot TWRP Hacked fastboot

> **Inside:** the goal and the three-tier design we chose · the kit, files and LED colours you need to know · eight numbered steps with the real terminal output from this build · how to wire it into Home Assistant · what to do with the other Dots and the Nest Hub.

Device	Result	Date	Written for
Echo Dot 2nd Gen (RS03QR, "biscuit")	EchoLocal 0.0.7 · ESPHome API on :6053	22 September 2026	Kevin

Contents

1 · The exercise

The goal was a voice assistant for the house that runs **locally**: no Amazon, no Google, no cloud in the loop for everyday commands, but free to reach the internet when you actually ask it to look something up. Home Assistant (HA) is already running on a Raspberry Pi, so the assistant should plug into that rather than replace it.

We started by looking at the Google Nest Hub, then at the Amazon Echo Dot "pucks" already in the house. The Nest Hub turned out to be a poor candidate: its bootloader exploit was patched by Google in 2021, and its chip is far too small to run any speech or language model anyway. The Echo Dots were the opposite story. The 2016 second-generation Dot (model `RS03QR`) has a well-documented bootloader exploit and two open-source projects, **EchoLocal** and **EchoMuse**, that replace Alexa outright.

No cloud · On-device wake word · Native ESPHome device · Media player · LED ring feedback · Bluetooth proxy · Lux sensor

Along the way we also evaluated and rejected two shiny new things: Google's *Home MCP* (a cloud service that lets an AI agent control Google Home devices, which is the opposite direction to what we want) and TypeSafe's *Jev* model (a closed, US-hosted decision model that cannot generate speech or run locally). Neither moved the design.

2 · The design we landed on

Three tiers. The Dot is deliberately dumb: it listens for the wake word, streams audio, and speaks the reply. Everything clever lives on machines with fans.

Satellite · the Dot
Runs EchoLocal. Wake word detection on-device (microWakeWord / openWakeWord). Talks to HA over the ESPHome native API on port 6053. Also a media player, Bluetooth proxy and light sensor.

→

Home Assistant · the Pi
Runs the Assist pipeline. Matches sentences to devices locally. Whisper turns speech to text, Piper turns text to speech. Only unmatched sentences go to the LLM.

→

LLM brain · a box in a cupboard
Ollama with a Qwen3 4B or 8B model. Handles conversation, questions, and web lookups via a search tool. A mini PC, a Jetson Orin Nano, or an old PC with a GPU.

This document covers the first tier end to end: the Dot. The Home Assistant pipeline is the next job and is sketched in *What's next*.

3 · Before you start

Kit

- An Echo Dot 2nd generation. Check the label on the base for `RS03QR`. Nothing else in this guide applies to any other model.
- A Windows PC. We used an HP Envy laptop. The current unlock has a Windows path, so no Linux is needed unless the Dot cannot reach fastboot.
- A **micro-USB data cable**. Charge-only cables will not work. The one in the Dot's box is fine.
- The Dot's normal power supply. USB from the PC alone is not enough to boot it reliably.

Files

Put everything in one folder. We used `C:\Users\user\Desktop\Echo Dot Hack\` and extracted the amonet zip inside it, so all commands ran from `...\amonet-biscuit-v2.0.0\amonet`.

FILE	WHERE FROM	NOTES
`amonet-biscuit-v2.0.0.zip`	Attachments on the XDA thread (see Sources)	The exploit plus TWRP, scripts and `adb.exe`. Use v2.0.0, not v1.1.0.
`boot-root.zip`	Same thread attachments	Roots both slots and turns ADB on.
`update-kindle-biscuit_puffin-NS65741_user_8142_0013222530692.bin`	Amazon's CDN, link on FTVDB	Fire OS 6574.1, build 13222530692, 106 MB. MD5 `ead2ea9a9ca2fa1c708381a07c605356`.
`echolocal_windows_x86_64.exe`	EchoLocal release 0.0.7 on GitHub	The *release asset*, not the green "Code" button zip.
Kindle Fire USB driver	Amazon developer site	Install before plugging the Dot in. Google's USB driver is the fallback.

CHECK THE FIRMWARE DOWNLOAD BEFORE YOU FLASH IT

In a cmd window in the folder, run the line below. The hash printed must match the MD5 in the table exactly. A truncated download flashed to both slots is the one mistake TWRP cannot protect you from.

```
certutil -hashfile update-kindle-biscuit_puffin-NS65741_user_8142_0013222530692.bin MD5
```

Reading the LED ring

The Dot has no screen. The light ring is the only feedback you get, and every colour means something specific.

RING	MEANING	WHEN YOU SEE IT
🔵 Blue and cyan, spinning	Normal boot	Every power-on. Also what you get if the button press was missed.
🔵 Solid blue, stays on	Button held, but stock fastboot not offered	Reported in the thread. Retry timing, update firmware, or fall back to the test-point method.
🟠 Orange	Amazon setup mode, no WiFi	After the FireOS flash, before EchoLocal. Ignore it.
🟢 Solid green	Stock fastboot	The target of Step 1. Also flashes green when a TWRP install succeeds.
⚪ Solid white	TWRP recovery	After fastbrick, and whenever you run `adb reboot recovery`.

Spinning rainbow	Hacked fastboot	Only after the exploit is installed.
Red, a few seconds	A TWRP install failed	Stop and read the terminal.

TWO WAYS TO PERMANENTLY BRICK IT

Interrupting `fastbrick.bat` after its ten-second grace period, and flashing anything that touches the preloader, LK or TrustZone partitions. Neither happens if you follow the steps. Only ever flash stock firmware through TWRP, which protects those partitions.

4 · The steps

Every command below was typed into a Command Prompt opened **inside the extracted amonet folder**, because that is where `adb.exe` lives. Quickest way to get one: open the folder in Explorer, click in the address bar, type `cmd`, press Enter.

1 Get the Dot into fastboot

Target: a solid green ring

Deregister the Dot in the Alexa app first if it is still on your account. Then: **hold the action button** (the one marked with a dot, not the microphone-mute button), **plug the power in while holding**, and keep holding for a good ten seconds. When the ring goes solid green, let go. Now connect the USB cable to the PC.

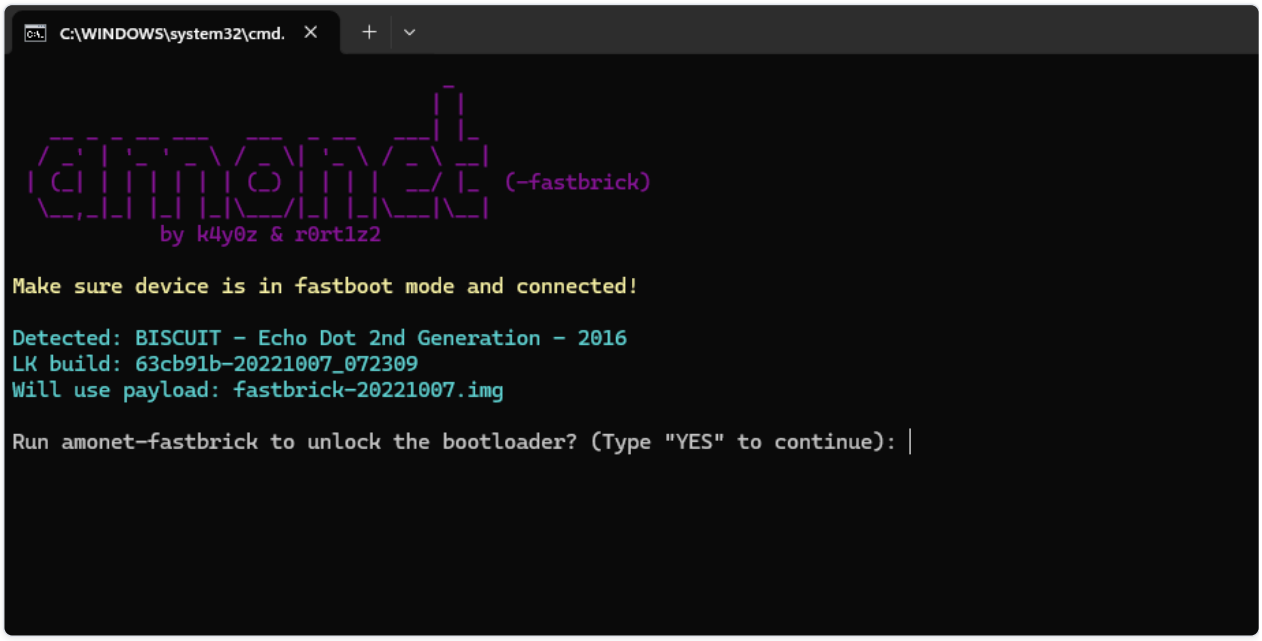
IF YOU GET BLUE INSTEAD OF GREEN

Spinning blue means the press was missed: hold earlier and longer. A *solid* blue that stays on is a known case in the unlock thread where stock fastboot is not reachable by button. The cheap fix to try first is letting the Dot update its firmware once via the Alexa app. If that fails, the thread's Option 2 applies: open the case and short a test point while running the Linux `bootrom-step.sh` script. We did not need it.

2 Break the bootloader with fastbrick

Windows, case closed, about a minute

Double-click `fastbrick.bat`. It should detect the Dot and print the bootloader build. The string `63cb91b-20221007_072309` is the one the exploit targets.



What success looks like. The script found the Dot, matched the LK build and chose its payload. It is waiting for you to type YES.

Type `YES` and press Enter. There is a ten-second grace period; after that, **do not touch the Dot, the cable or the laptop**. The ring fills as a progress bar, goes green, then the Dot reboots itself into TWRP with a solid white ring. The script window then just says "press any key to exit". That is normal.

3 Flash FireOS 6 to slot B

The Dot has two system slots, A and B. We fill both.

With the white ring showing, copy the `.bin` and `boot-root.zip` into the amonet folder if they are not there already, then confirm ADB sees the Dot:

```
adb devices
List of devices attached
SERIAL-REDACTED         recovery
```

Wipe, push the firmware and install it. TWRP writes to whichever slot is *inactive*.

```
adb shell twrp wipe cache
adb shell twrp wipe data
adb push update-kindle-biscuit_puffin-NS65741_user_8142_0013222530692.bin /sdcard/update.zip
adb shell twrp install /sdcard/update.zip
```

```
C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb shell twrp install /sdcard/update.zip
* daemon not running; starting now at tcp:5038
* daemon started successfully
Installing zip file '/sdcard/update.zip'
Unmounting System...
Flashing A/B zip to inactive slot: B
Step 1/2Step 2/2To flash additional zips, please reboot recovery to switch to the updated slot.
Done processing script file

C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>
```

First install. Note "Flashing A/B zip to inactive slot: B". The ring runs as a progress bar and ends green.

4 Switch slots and flash slot A

So the Dot boots whichever slot it picks

The install did not move the active slot, so we do it by hand. Check, then set it to B so that A becomes the inactive target:

```
adb shell bcbtool get_active
a
adb shell bcbtool set_active b
adb shell bcbtool get_active
b
```

Reboot into recovery, wait for the white ring, and install the same file again:

```
adb reboot recovery
adb shell twrp install /sdcard/update.zip
```

```
C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb shell twrp install /sdcard/update.zip
Installing zip file '/sdcard/update.zip'
Unmounting System...
Flashing A/B zip to inactive slot: A
Step 1/2Step 2/2To flash additional zips, please reboot recovery to switch to the updated slot.
Done processing script file

C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>
```

Second install. This time it says slot **A**. If it said B again, the slot switch did not take and you would be overwriting the same copy.

> **WHY NOT THE THREAD'S ONE-LINER?**
>
> The XDA guide uses a single long `bcbtool` command with a shell `case` statement. It works on Linux but its quoting is fragile in Windows cmd. Two plain commands do the same job with nothing to go wrong.

5 Root it with boot-root

Both slots in one go

```
adb push boot-root.zip /sdcard/
adb shell twrp install /sdcard/boot-root.zip
```

```
C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb shell twrp install /sdcard/boot-root.zip
Installing zip file '/sdcard/boot-root.zip'
Unmounting System...
- info slot _a: patching boot cmdline
- info slot _a: disabling dm-verity
- info slot _a: mounting system
- info slot _a: patching properties
- info slot _a: installing fos flags override
- info slot _a: patching adbd
- info slot _a: patching selinux policy
- info slot _a: patching fstab.mt8163
- info slot _a: neutering ota updates
- info slot _b: patching boot cmdline
- info slot _b: disabling dm-verity
- info slot _b: mounting system
- info slot _b: patching properties
- info slot _b: installing fos flags override
- info slot _b: patching adbd
- info slot _b: patching selinux policy
- info slot _b: patching fstab.mt8163
- info slot _b: neutering ota updates
- info done
Done processing script file

C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>
```

boot-root at work. For each slot it patches the boot command line, disables dm-verity, patches `adbd` to run as root, relaxes SELinux, and, importantly, **neuters OTA updates** so Amazon can never push firmware back onto it.

6 Boot and prove root

First boot of FireOS 6 takes a couple of minutes

```
adb reboot
```

Blue and cyan spin for a while, then it settles on orange because there is no WiFi. Ignore the orange; we never set it up with Amazon. Then:

```
adb devices
adb shell id
```

```
C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb devices
List of devices attached
                device


C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb shell su -c id
/system/bin/sh: su: not found

C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>
```

ADB is back, now in normal mode. The `su: not found` error was our mistake: boot-root does not install a separate `su` binary, it makes the ADB shell itself root.

```
C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>adb shell id
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin)
,3002(net_bt),3003(inet),3006(net_bw_stats),3009(readproc) context=u:r:su:s0

C:\Users\    \Desktop\Echo Dot Hack\amonet-biscuit-v2.0.0\amonet>
```

Proof. `uid=0(root)` with an `su` SELinux context. The Dot is unlocked and rooted. If the ADB list is empty after a couple of minutes, one power-cycle fixes it, as the thread warns.

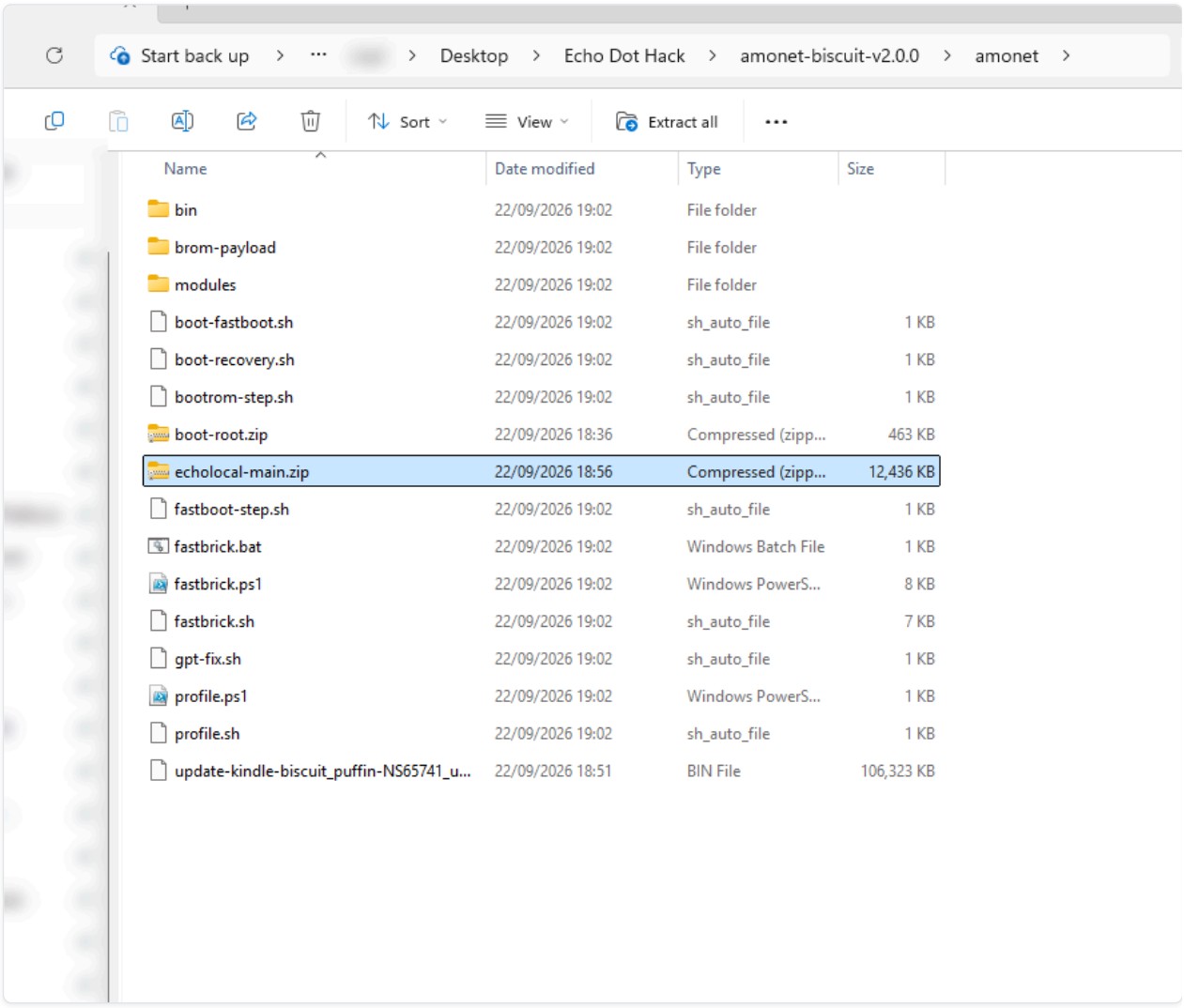
7 Install EchoLocal

This is the step that removes Alexa

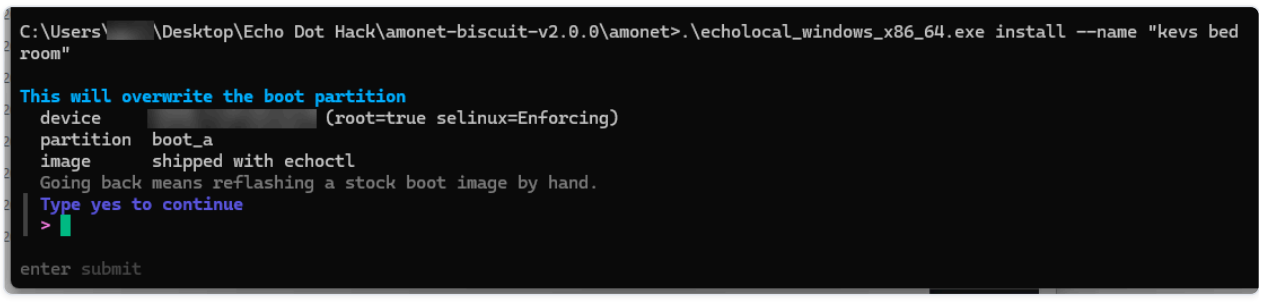
Download `echolocal_windows_x86_64.exe` into the amonet folder and run it from the same window. Quote the name if it has a space in it.

```
.\echolocal_windows_x86_64.exe install --name "kevs bedroom"
```

> **TWO MISTAKES WE MADE HERE, SO YOU DON'T HAVE TO**
>
> First, we downloaded `echolocal-main.zip` from GitHub's green Code button. That is the source code, not the program. The program is a *release asset* on the Releases page. Second, we typed the name without quotes, so cmd split "kevs bedroom" into two arguments.

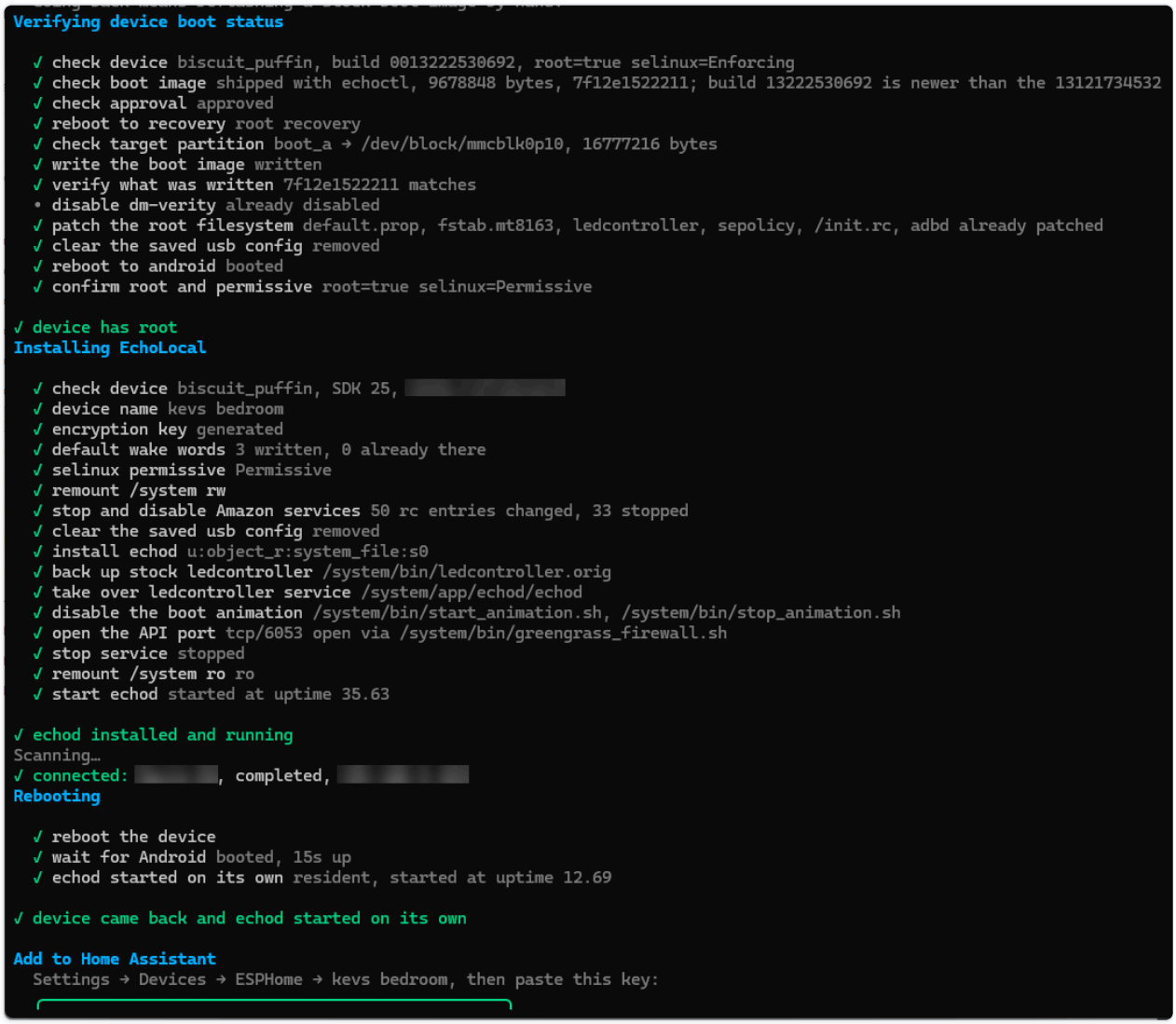


The wrong file. `echolocal-main.zip` is source code. Delete it and download the `.exe` from the release page instead.



The right tool talking to the right Dot. It sees root, targets `boot_a`, and warns that going back means flashing a stock boot image by hand. TWRP and the FireOS file mean that door stays open. Type `yes`.

It then asks for your WiFi name and password, flashes its own boot image, stops every Amazon service, opens the API port, and reboots the Dot to prove its daemon starts on its own.



The finished install. Read down the ticks: **50 Amazon services stopped**, `echod` installed and started, API on `tcp/6053`, connected to WiFi, rebooted, and `echod` came back on its own. At the bottom it prints the ESPHome encryption key. **Copy that key before closing the window.**

> **AT THIS POINT THE DOT IS DONE**
>
> It boots straight into EchoLocal, survives power cycles, and never contacts Amazon. Everything that follows is Home Assistant configuration.

8 Add it to Home Assistant

One network problem to fix first

Home Assistant connects *to* the Dot on port 6053, so the two must be able to see each other. In our house they could not: the Pi running HA is wired to the main router on `192.168.0.x`, while the Dot joined the Tenda Nova mesh WiFi, which was running as a second router with its own `192.168.5.x` network behind NAT. Nothing on the wired side can reach anything on the mesh.

DEVICE	ADDRESS	NETWORK
Main router	`192.168.0.1`	Wired, straight to the ISP
Home Assistant (Raspberry Pi)	`192.168.0.x`	Wired
Tenda Nova mesh	`192.168.0.x` on its WAN side	Runs its own `192.168.5.x` LAN
The Dot ("kevs bedroom")	`192.168.5.x`	Tenda WiFi [redacted], unreachable from HA

The fix: in the Tenda WiFi app, switch the Nova from router mode to **bridge (AP) mode**. It then just extends the main network over WiFi. The Dot reconnects to the same WiFi name and receives a `192.168.0.x` address, and so does every phone and gadget on the mesh. This also fixes HA discovery for everything else wireless. † Not needed in this house — see the note on page 11.

Then, in Home Assistant: **Settings → Devices & services**. "kevs bedroom" appears as a discovered ESPHome device. Click *Configure* and paste the key. If it has not appeared after a minute, add the ESPHome integration manually with the Dot's new IP and port `6053`. On the device page, pick an Assist pipeline and one of the three wake words EchoLocal loaded. Say it, and the ring lights up.

5 · What's next

The voice pipeline

In HA under **Settings → Voice assistants**, create an assistant using the Whisper add-on for speech-to-text and the Piper add-on for text-to-speech. Both run locally on the Pi or on the LLM box via the Wyoming protocol. For the conversation agent, add the Ollama integration pointing at the LLM box, pick a Qwen3 4B or 8B model, and turn on *Assist* control so the model can call HA's device tools. A web search tool gives it internet on request.

The LLM box

Anything always-on with 16 GB RAM handles the 4B model. A used N100 mini PC gives one to three second replies with Whisper medium. A Jetson Orin Nano Super or an old PC with a 12 GB GPU is only needed for the 8B model at speed.

The other Dots

Identical procedure. The green-ring trick worked on this unit, so it will most likely work on its siblings. Give each a different `--name`. Budget about 20 minutes per Dot now that the files are downloaded.

The Nest Hub

Two honest options: keep it stock and use HA Cast to show a dashboard on it, accepting that Gemini stays; or gut it, keep the case, screen, mics and speaker, and put a Raspberry Pi CM5 on a custom carrier board behind the panel. The original display has no public datasheet, so a known 7-inch DSI panel is the pragmatic swap. That is a separate project.

6 · Appendix

Button modes after the exploit

The exploit remaps the buttons. All of these are done by holding the button while connecting power.

HOLD	MODE	RING	USE
Volume Up	TWRP recovery	White	Flash zips, restore stock, reinstall EchoLocal
Volume Down	Hacked fastboot	Rainbow	`fastboot flash` of boot images
Mute, with USB connected	Preloader USBDL	None	Emergency unbrick with MTKClient. Never flash TEE1, LK or preloader here.
Action, about 2 s after power	Hacked fastboot	Rainbow	Same as Volume Down, per the thread author

If it will not boot

Do not keep power-cycling it. The preloader counts failed boots per slot and stops booting entirely when both run out. Instead, go to TWRP with Volume Up, and either reinstall the FireOS zip to the inactive slot or rerun the EchoLocal install. Recovery costs about ten seconds; a boot-loop costs an opened case.

Going back to Alexa

Possible but not a button press: boot TWRP, install the FireOS zip to both slots again (Steps 3 and 4) without boot-root, and the Dot boots stock FireOS 6 with the exploit still underneath. Since v2.0.0 of the exploit only FireOS 6 will boot; FireOS 5 images are no longer usable.

Sources

- XDA: *[UNLOCK][ROOT][TWRP][UNBRICK] Amazon Echo Dot 2nd Gen / 2016 (biscuit)* by Rortiz2, with contributions from k4y0z. xdaforums.com/t/…biscuit.4761416
- EchoLocal by ygelfand. github.com/ygelfand/echolocal, release 0.0.7.
- EchoMuse by wilbowes (the alternative, needs FireOS 5 and the v1.1.0 unlock). github.com/wilbowes/EchoMuse
- FTVDB firmware record for Fire OS 6574.1 build 13222530692. ftvdb.com/echo/firmware/com.amazon.biscuit.android.os
- Home Assistant voice documentation: Assist pipelines, Wyoming, ESPHome voice satellites. home-assistant.io/voice_control

---

Written up from the live session on 22 September 2026. Serial numbers, addresses, folder paths and the WiFi name are redacted for publication.
The ESPHome encryption key was never included.

Notes for publication

Added 23 September 2026, when this build log was published on the DECS Home Systems site.

Personal details are redacted.

The Windows username in the folder paths, the Echo Dot's serial number, the home Wi-Fi name and the local network addresses have been removed from the text and blurred in the terminal screenshots. The ESPHome encryption key was never included.

† Step 8: no mesh switching was needed.

In this house the Home Assistant Pi is dual-homed — wired to the main router and also joined to the mesh Wi-Fi — so it reached the Dot on the mesh network directly and nothing about the mesh had to change. Switching a mesh to bridge (AP) mode is still the general fix if your Home Assistant cannot see the Dot on another subnet.

Who wrote what.

This build log was written during the unlock session by a separate Claude chat, which did not know the house's Home Assistant setup. The network, the voice pipeline (Whisper on the GPU server, Gemini, Piper), the far-field tuning and the wall-panel visuals were done afterwards with Claude Code. The whole story is on the Echo Voice page: anchorapp100.github.io/globalguard/echo.html