Episode Book

EPISODE #5918

Splitting One Camera's Audio for a Baby Monitor

PUBLISHED	12 October 2026 at 19:55 UTC
DURATION	25:51
LISTEN	myweirdprompts.com/episode/camera-audio-split-baby-monitor-74640/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.4
SOURCES	11 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:41 UTC

SYNOPSIS

Daniel has an IP camera in his son's room doing double duty — Frigate watches the video, but he also wants a continuous audio line piped to his phone so a cry wakes him up. The camera won't hand out an audio-only feed, so something has to split the audio off the RTSP stream. This episode works through the architecture: why go2rtc beats Frigate's embedded restream, how to fan out one camera connection into multiple consumers, where the single transcode belongs, and which codecs actually matter when the sound you're trying to catch is a cry at three in the morning.

TRANSCRIPT



Corn

Daniel's got a camera pointed at a sleeping child, and he wants to argue about codecs. That's the whole episode, really.



Herman

It is, and I've been looking forward to it since I read it.



Corn

Here's what he's asking. Ezra's room has an IP camera, RTSP, and it only goes on when Ezra's in there on his own. Two jobs for one device. The video job Frigate already handles, that's the check-in, make sure he's breathing, make sure he hasn't climbed the bookshelf. The audio job is the older one. When Ezra was smaller they wanted a channel just left open, piped to their phones, so a cry wakes them up. That's not an event notification, that's an open line. And the camera won't hand out an audio-only feed, so something has to split the audio off the RTSP stream. He's leaning go2rtc consuming the RTSP and feeding both Frigate and Icecast. But he knows Frigate restreams on its own, so he wants to know if it can produce an audio-only restream instead. Then a detection layer for crying, then Home Assistant firing the notifications. And he wants the whole thing efficient.



Herman

Every piece of that exists. There's no research project hiding in this one. What there is, is a seam problem.



Corn

Go on.



Herman

Because he's actually describing two architectures, and he's treating them as one. Event-driven push is Frigate sees a cry, publishes it, Home Assistant wakes the phone. Always-on ambient audio is an Icecast mount point playing on the nightstand all night, and the parent hears whatever happens, cry or no cry.



Corn

Those are different anxieties wearing the same trenchcoat.



Herman

Exactly that. One is "tell me when something happens." The other is "I want to be in the room." And Daniel wants both, which is fine, they're complementary. But they have different failure modes and different latency budgets, and if you build them as one thing you'll optimize the wrong part.



Corn

So the two pieces of the prompt that matter most are the split and the latency.



Herman

Latency is the constraint nobody names. A cry monitor that's two minutes late isn't a monitor, it's a decoration. It tells you what your child was doing when you finally got up.



Corn

Which, in fairness, you already knew.



Herman

You knew you'd been woken up. You didn't need the timestamp.



Corn

Right, so where do we start. Frigate's restream, since that's his actual question.



Herman

Frigate's restream is RTSP, at the Frigate host on port 8554, path is the camera name. And the docs are explicit that it's a copy. The video feed is copied from the original directly to avoid re-encoding. It does not include Frigate's annotations, it's the raw thing. So the audio rides along with the video, and there is no toggle anywhere in the restream docs for audio-only.



Corn

So the literal answer to what he asked is no.



Herman

The literal answer is no. Not out of the box.



Corn

And the interesting answer?



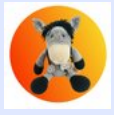
Herman

The interesting answer is that Frigate's restream is go2rtc. Frigate uses go2rtc to provide its restream and its MSE and WebRTC capabilities. It's not a Frigate feature that happens to look like go2rtc, it is go2rtc, running inside Frigate. Which means the go2rtc block in your Frigate config is a real go2rtc config.



Corn

So he can define an audio-only source inside Frigate's own embedded go2rtc.



Herman

With an FFmpeg source that drops video. The docs' own idiom for this is an ffmpeg source with the name of the camera and then a hash and audio equals opus. That's the documented pattern for transcoding audio into a codec the stream doesn't already have. You can point the same idea at an audio-only output.



Corn

Which is go2rtc.



Herman

So he's right that Frigate can be made to hand him an audio-only feed, and he's right that it isn't the clean way to get one. Because if you're going to write go2rtc configuration anyway, you may as well run go2rtc.



Corn

His first architecture wins.



Herman

And not by a small margin, because the config lives in the same place either way, but standalone go2rtc gives you an HTTP API you can pull from, and Frigate's embedded one is oriented toward Frigate's needs. If Icecast is a consumer you care about, you want the thing with the documented outputs.



Corn

Which brings the fan-out into view. What's go2rtc actually doing for him.



Herman

Consuming one RTSP input and serving several outputs at once. That's its design, it's not a clever use of it. Frigate gets the stream with the roles it needs, Icecast gets a separate pull, and neither one is opening a second connection to the camera. That last part matters more than it sounds.



Corn

Because cheap cameras have a connection limit.



Herman

Most of them do, and a surprising number of them will hand out two or three streams happily and then fall over on the fourth. Frigate's docs actually flag this. The ffmpeg process for capturing audio is a separate connection to the camera on top of the other roles you've assigned, and for that reason they recommend using the go2rtc restream as the audio source. So Frigate is telling you to fan out, and it's telling you why.



Corn

One connection to the camera, everything else hangs off the hub.



Herman

One connection in, everything else downstream. And the audio-only pull point is already there. go2rtc's MPEG module exposes HTTP endpoints. There's a stream dot aac endpoint on the API port, 1984, with the source as a query parameter. That gives you ADTS, which is raw AAC with framing. There's also stream dot ts, MPEG transport stream, same port. So you can pull audio alone from go2rtc without ever touching the video track.



Corn

Two ways to pull the same audio.



Herman

Two shapes of the same thing, and they differ in what the receiving end wants. ADTS is the audio elementary stream, no container. MPEG-TS is a container. If your consumer is FFmpeg either one works. If it's something fussier, TS tends to be the more forgiving input. go2rtc also lists adts and wav as streaming outputs and mp3, aac, opus, pcma and pcmu as codecs it can carry.



Corn

Which leads us to the part where the free cameras cut corners.



Herman

Many of these cameras emit PCMA or PCMU natively, because it costs nothing and it's fine for a doorbell. And go2rtc's docs are blunt about it. PCMA and PCMU are very low quality codecs. They support only two hundred and fifty-six different sounds. Two hundred and fifty-six. That's the entire palette. And the docs say use them only when you have no other options.



Corn

Two hundred and fifty-six sounds is a description of an intercom, not a baby.



Herman

It's a description of a phone call from 1988. G.711 was designed for speech intelligibility on a copper line and it's good at that. It is not good at carrying the difference between a cough and a coo and a footstep. And here's the Frigate side of the same problem. FLAC isn't supported in an RTSP stream at all. And if you're using Frigate or Home Assistant to record MP4 files with PCMA, PCMU or PCM audio, you should set up transcoding to AAC.



Corn

Meaning if he does nothing, his recordings are G.711.



Herman

His recordings are G.711 and they'll play, they'll just sound like a phone call. Which for a check-in recording of a nursery is arguably fine, and for anything you'd want to keep, isn't.



Corn

So where does the transcode go.



Herman

Once. That's the whole efficiency argument. Camera comes in as PCMA, go2rtc transcodes it once to a single audio-only output, and both consumers take that output. Frigate's audio detector is reading the transcoded stream and Icecast is serving the



Herman (continued)

transcoded stream, and there was one transcode on the box for the whole house.



Corn

And the bad version?



Herman

The bad version is Frigate transcoding for its own audio role and a separate FFmpeg process pulling the raw RTSP and transcoding again for Icecast. Two transcodes of the same audio, two connections to the camera, and now the CPU number you're looking at isn't the one you built.



Corn

Does the transcode actually cost anything at this scale.



Herman

At this scale, no. It's one mono audio stream from one microphone. Any machine that can run Frigate at all can transcode a single audio channel all day. The reason to do it once isn't CPU, it's that two transcodes are two things that can be configured differently and drift apart.



Corn

And then the bitrate question, which Daniel has an instinct about.



Herman

He says low, and he's right, and the community reference point is higher than he'd expect. The command everyone copies is a one-liner FFmpeg push to Icecast, and it uses libmp3lame at a hundred and twenty-eight kilobits. That's the canonical number, it's what's in the thread, and it's the reference value, not a floor.



Corn

A hundred and twenty-eight is what you'd use to stream an album.



Herman

It is, and for a continuous monitor it's about four times what you need. Speech at thirty-two kilobits mono is perfectly intelligible for "is that a cry or is that the cat", and that's the whole question you're asking of this stream at three in the morning. You are not evaluating timbre.



Corn

The point of the stream is one bit of information.



Herman

The point is a yes or no. And a lower bitrate also means a smaller buffer, which is the next thing we should talk about, because it's the actual constraint on this whole design.



Corn

Before we leave the mechanism, there's a security footnote on the exec route.



Herman

There is, and it belongs here because it's the thing that bites people. go2rtc's exec source runs an arbitrary FFmpeg command and pipes its output. That's the documented hook for pushing to a service go2rtc doesn't natively speak. But exec, echo and expr are disabled by default in current go2rtc and Frigate builds. To use them you set an environment variable, GO2RTC_ALLOW_ARBITRARY_EXEC, to true, and the docs flag that as a security risk, because it is one.



Corn

You are telling your camera proxy to run shell commands.



Herman

You're telling a network-facing service to run commands you wrote, in a config file that other people can sometimes edit. If anything else on your network can write to that config, you've handed it a shell. So the cleaner pattern is the other one, a separate FFmpeg process pulling go2rtc's HTTP endpoint and pushing to Icecast. No exec, no flag, no new attack surface.



Corn

Good. Now the bridge, because we've been assuming one exists.



Herman

There isn't one. go2rtc's documented outputs are RTSP, RTMP, HLS, WebRTC, MP4 and MSE, MPEG-TS, ADTS, WAV, MJPEG, HomeKit, WebTorrent. Icecast is not on that list. There's no first-party Icecast output and I don't expect one, because Icecast is a source-client protocol and go2rtc's job is to be the source everybody else pulls from.



Corn

So FFmpeg is the bridge.



Herman

FFmpeg is the bridge. It pulls go2rtc's stream dot aac or stream dot ts endpoint and pushes to Icecast. And the command people actually use is short. It's the real-time flag, the RTSP URL or in this case the go2rtc endpoint, libmp3lame at that hundred and twenty-eight kilobit setting, a content type header of audio slash mpeg, output format mp3, and then the Icecast source URL.



Corn

And the header is not optional.



Herman

The header is the part people miss and then spend an evening on. Content type audio slash mpeg, called out in that thread specifically for Chromecast and Google Home compatibility. Without it, the Chromecast has no idea what it's been handed and the stream just doesn't play. The audio arrives, the device shrugs.



Corn

Because the container says mp3 but the transport doesn't say audio.



Herman

Right, and a dumb client believes the transport. Your browser is forgiving because it sniffs. A speaker doesn't sniff. It reads the header and either accepts the stream or doesn't.



Corn

There's a detail in that thread that's almost exactly Daniel's setup.



Herman

There is, and it's why this is a solved problem rather than an interesting one. The author of that thread is running a Unifi camera, pushing its RTSP audio to Icecast with a bash script, and his stated reason is that what he misses about the old analog baby monitors is having a constant audio stream from the nursery. That's the sentence. He then does ffmpeg_noise detection in Home Assistant off the same stream, and uses a switch template to turn the nursery audio on in the bedroom speakers.



Corn

Same three layers.



Herman

Same three layers, five years ago, and it still works. Server, detection, playback. The interesting part is that his detection layer is the raw one, a noise threshold on the Icecast stream, and the thing Daniel's considering is the labeled one.



Corn

So let's do detection, because that's where the two paths split.



Herman

Frigate has a built-in audio detector. CPU only, relatively lightweight, and the included model has over five hundred different types of audio it can recognize. Five hundred. The defaults it enables are bark, fire alarm, scream, speech and yell. Bark is the odd one in a nursery, but it's a default, not a commitment.



Corn

So crying is in there somewhere.



Herman

Crying is one label among the five hundred and something. That's the honest shape of it. There is no dedicated cry-detection model in Frigate. It's a general audio classifier that happens to have a crying class, and people run it on nursery cameras for crying, yell, scream, speech, groan and cough. It works. It isn't purpose-built and you should expect it to be about as good as a general classifier is at a specific job.



Corn

And the threshold.



Herman

It uses min volume, an RMS threshold, the same way motion detection uses a pixel threshold. Nothing above the line, nothing detected. And because it's cheap, volume above min volume also counts as motion for recording retention, which is a nice second



Herman (continued)

use. There's a Debug UI Audio tab with a live level graph, which is the tool you use to set that threshold. You watch the graph while the baby sleeps and you watch it while the baby cries and you pick the line in between.



Corn

You're calibrating a microphone against one specific baby.



Herman

Against one specific baby in one specific room with one specific HVAC system. Which is fine, because that's the only baby the system needs to work for.



Corn

And the source for the detector.



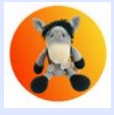
Herman

The go2rtc restream. Frigate's own docs say it, in the same paragraph we already quoted. The audio capture is a separate connection, so use the restream. So the fan-out isn't just an efficiency choice, it's the recommended topology for the audio detector specifically.



Corn

Which means Daniel's instinct and the docs agree.



Herman

They agree, and they agree for a reason that has nothing to do with elegance. The camera doesn't want four connections.



Corn

Now the other detection path.



Herman

Home Assistant's own ffmpeg_noise, which is a raw level trigger on the Icecast stream. No labels, no events, no recordings. It fires when the level crosses a threshold. That's it. And it's what the original thread used, because it was there and it was enough.



Corn

So which is keener.



Herman

Depends what you want out of it. If you want a labeled event with a recording attached so you can look at the thirty seconds around the cry, Frigate's path. If you want a switch that flips when there's noise in the room, the HA path is fewer moving parts. Running both is redundant and you'll get two notifications for one cry.



Corn

And the notifications themselves.



Herman

Frigate publishes per-camera audio state over MQTT. There's a topic for each audio type, camera name and audio type, and it publishes ON or OFF, which is directly usable as a binary sensor. There's also a dBFS topic and an RMS topic for the level values if you want to graph it, and a transcription topic for text. The Home Assistant integration exposes the audio state as sensors.



Corn

And people have built this exact automation.



Herman

There are blueprints for it. A baby monitor notification automation triggering on binary sensor baby crying sound, baby yell sound, baby scream sound. That's the shape. You're not writing anything novel, you're wiring three existing sensors into a notify action.



Corn

There's a gotcha in there.



Herman

There's a gotcha. Some audio labels were only being sent in an update MQTT message rather than a new message, which means automations triggering on new miss them. There's a discussion thread about it and the blueprints needed tweaking. So if your automation fires for yell and not for cry, that's why, and it's a five-minute fix rather than a bug in your config.



Corn

Now the part that decides whether any of this is worth building.



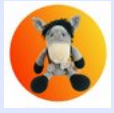
Herman

Latency.



Corn

Because we've built a beautiful fan-out and it might all be two minutes late.



Herman

Two minutes late is real and it's measured. One person in that thread reports ten to twenty seconds of latency on Unifi Protect RTSP streams, and blames the NVR, not Icecast, not the codec. Another reports one to two seconds for a Foscam RTSP audio channel played through VLC with buffering disabled. And then the same person reports roughly two to three minutes when playing the same Icecast stream through Google Home, and he attributes that to Google's client buffer, because it is.



Corn

Same source, same server, same everything, and the difference is the speaker.



Herman

The difference is the speaker. Which means the latency in this architecture is not a server property at all. It's a property of the camera or NVR at the front and the playback device at the back, and the middle, go2rtc and Icecast and FFmpeg, contributes almost nothing.



Corn

So if Daniel builds this and it's slow, the first thing to check is the phone.



Herman

The first thing to check is the phone, and specifically whether the player has a buffer you can shrink. VLC lets you, and that's where the one to two seconds came from. A smart speaker doesn't, and that's where the two to three minutes came from. An always-on stream on a phone with buffering off is a completely different experience from casting to a Home speaker, even though it's the same mount point.



Corn

Icecast will happily serve both.



Herman

Icecast will serve both and has no opinion. It's a source-client server. It hands out the stream at whatever bitrate you configured and it does not buffer on your behalf. The buffering is entirely the client's business, which is why the client is a design decision and not an afterthought.



Corn

So the phone in the bedroom isn't a detail.



Herman

The phone in the bedroom is the last mile and it's the mile that decides whether this works. The server side is elegant and the server side is not where the problem is.



Corn

Which gives us a shape. Front end gives you one RTSP stream, go2rtc is the hub, one audio-only transcode, two consumers, Icecast for the open channel, Frigate for the labeled detection, MQTT into Home Assistant for the notification. And all of it is only as fast as the device you're listening on.



Herman

And that's the answer to the prompt, mostly. He asked whether Frigate's restream could feed Icecast instead of go2rtc, and the answer is that it can in the sense that Frigate's restream is go2rtc, and it can't in the sense that there's no audio-only restream feature to point at. So the first architecture is the right one, and it's the right one because it's the same software with fewer layers in front of it.



Corn

One more thing on the detection side. Frigate does transcription now.



Herman

It does, locally, with sherpa-onnx or Whisper, and it is explicitly not a continuous transcription service. It's manual, on demand. You point it at a recording and it transcribes. So if someone's thinking "I'll transcribe the nursery all night and look for the word crying", that's not what the feature is, and it's the wrong tool for a live monitor. It's the right tool if you want to know what was said during an event you already recorded.



Corn

Which is a different product.



Herman

A different product entirely.



Corn

Herman Poppleberry, you have been waiting to talk about jitter buffers for eleven minutes.



Herman

I have, and I got to do it.



Corn

You did.



Hilbert

Your bitrate figure is wrong. It's not a hundred and twenty-eight.



Corn

Where does it come from then.



Hilbert

It comes from a forum post in twenty twenty-one, and the man who wrote it said himself it was the number he had lying around. He was pushing to a bedroom speaker over wifi. I've been running a nursery feed off my own setup for a while now, the kid's grown, but the stream's still up, mostly because the mount point is on the kitchen speaker and my wife likes the room tone.



Corn

The room tone.



Hilbert

The house has a sound and she likes knowing it's there. So it's a monitor for a room with nobody in it. Anyway. Opus at twenty-four kilobits mono. That's what's actually going over the wire. Sounds better than the mp3 does at a hundred and twenty-eight, and I've got the disk usage to prove it.



Herman

What's it going through?



Hilbert

A little thin client. Fanless, secondhand, the size of a paperback. Pulls the stream, transcodes to opus, pushes to icecast. Been on the shelf over the fridge for three years. I dust it about twice a year. Never touched anything else.



Corn

Over the fridge.



Hilbert

Cools better there. It's near the vent. Anyway, the point about the bitrate is that a hundred and twenty-eight was never a measurement, it was a sentence somebody wrote on a forum and everybody copied it. Twenty-four is a measurement. I measured it. It's fine.



Corn

And when the toddler was actually in the room?



Hilbert

Oh, it worked. It worked so well my wife made me turn it down. She could hear the kid breathing and it kept her up. Which is the whole design problem in one sentence, if you think about it. The monitor doesn't know when to stop monitoring.



Herman

That's the thing, isn't it. The always-on channel is the feature and it's the bug.



Hilbert

That's what she said. She said it exactly like that, actually. So now it plays to the kitchen and nobody listens to it.



Corn

There's something in that. The design decision isn't in the server, it's in the question of what you're allowed to hear.



Herman

Which is the thing neither of us said out loud. The architecture is easy. The hard part is deciding when to be listening.



Hilbert

And the volume knob. You don't need a threshold. You need a volume knob and somebody honest enough to turn it down.



Corn

There's a moment in the thread that sits with me. The man says what he misses about the old analog baby monitors is having a constant audio stream from the nursery.



Herman

It's the sentence in the entire business. Because the analog monitor did exactly one thing and it did it continuously, and everything we've built since is better at answering questions and worse at just being there.



Corn

Two architectures, two anxieties, and they don't collapse into one.



Herman

Not with what's on the shelf today. The camera watches, the stream listens, and the gap between them is where you decide how much you want to know.



Corn

And on the cutting-room floor, a detail that didn't fit. go2rtc supports HomeKit.



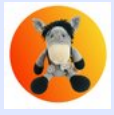
Herman

What?



Corn

As one of its outputs. Alongside RTSP and WebRTC and MPEG-TS, there's a HomeKit output, so a camera can appear in the Apple Home app without any bridge.



Herman

I want to look at that.



Corn

Irrelevant here, but a small window into what that project is willing to take on. So the keener architecture is go2rtc as the hub, one connection to the camera, one audio-only transcode, Frigate for labeled detection and recordings, Icecast for the open channel, Home Assistant on MQTT for the notification. The piece that will actually decide whether it feels right is the device on the nightstand, because the buffer is on that end and so is the person.



Herman

And Hilbert's fridge shelf is doing more than anyone gives it credit for.



Corn

A hundred and twenty-eight kilobits is what you use to stream an album, and he's streaming a room at twenty-four and making it sound better, which tells you the number was never the point.



Herman

The number was never the point.



Corn

That's the show. Thanks to Hilbert Flumingtop, who produces it, and who has never once dusted that thin client while we were listening. My Weird Prompts is the human-AI collaboration podcast, and if this one made you want to go configure something, leave us a review wherever you're listening. We'll be back soon.



Herman

Send us your own prompt on Telegram at t dot me slash MWP listener bot. See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **go2rtc README (GitHub, AlexxIT/go2rtc, master) [primary]**
   features, codecs, transcoding, security, projects-using list.
   https://github.com/AlexxIT/go2rtc

2. **go2rtc Streams module docs (internal/streams/README.md, master) [primary]**
   stream-to-camera, publish, preload, exec examples.
   https://github.com/AlexxIT/go2rtc/blob/master/internal/streams/README.md

3. **go2rtc MPEG module docs [primary]**
   ADTS/AAC and MPEG-TS HTTP output endpoints (`/api/stream.aac`, `/api/stream.ts`).
   https://go2rtc.org/internal/mpeg/

4. **Frigate Restream docs [primary]**
   RTSP restream, go2rtc config, audio=opus idiom, backchannel, exec security.
   https://docs.frigate.video/configuration/restream

5. **Frigate Audio Detectors docs [primary]**
   built-in CPU audio detector, 500+ labels, min_volume, audio role, transcription.
   https://docs.frigate.video/configuration/audio_detectors

6. **Frigate MQTT docs [primary]**
   `frigate/camera/audio/audio_type`, `/audio/dBFS`, `/audio/rms`, `/audio/transcription` topics.
   https://docs.frigate.video/integrations/mqtt

7. **HA Community forum**
   Icecast, an oldschool and audio-specific complement to HA (zacs, 2021-07-30 onward), baby-monitor use case, FFmpegIcecast command, latency reports.
   https://community.home-assistant.io/t/icecast-an-oldschool-and-audio-specific-complement-to-ha/325680

8. **Frigate Discussion #20904**
   audio label sent in `update` vs `new` MQTT message, blueprint workaround.
   https://github.com/blakeblackshear/frigate/discussions/20904

9. **Frigate Discussion #12649**
   configuring a baby-bassinet camera to record crying audio events only.
   https://github.com/blakeblackshear/frigate/discussions/12649

10. **Frigate HA integration PR #516**
    merged audio-detection sensors (NickM-27, 2023-07-25).
    https://github.com/blakeblackshear/frigate-hass-integration/pull/516

11. **Gist**
    Baby Monitor Notification using Frigate sensors (darnmason), HA automation on `binary_sensor.baby_crying_sound` etc.
    https://gist.github.com/darnmason/d4a6eb183dd685544d26f58169ff4b78