GetThermal+

User Guide

v1.0.0 — July 2026

Contents

Getting started

Reaching the camera

Give the camera power and a network:

- **Wired.** Plug in the Ethernet cable. The camera requests a DHCP address; read it off a Bluetooth scan (next section) or find it in your router's client list, where it appears as `gg-<serial>` — the processor board's full serial number, so the right unit is obvious even in a fleet. The camera also announces `gg-<serial>.local` over mDNS once a minute, so on networks and systems that support it, `http://gg-<serial>.local` reaches the camera by name. The serial is the **Board Serial** shown in Device Info on the Live page (Live view) — the processor board's, not the thermal core's, so the name stays the same if the core is ever swapped — and matching a name to a camera is exact. (Or use the static address if one was configured.)
- **No network configured yet.** The camera raises its own Wi-Fi access point named **GetThermal-Plus-Setup**. Join it and browse to `http://192.168.4.1/settings` — the settings page is open without login in setup mode so you can connect the camera to your network. Once it joins, reconnect to your own network and use the camera's new address.

Then open the camera's address in any browser: `http://<camera-ip>`.

The status LED on the camera narrates the boot: solid blue while it reaches for the network, one green triple-blip the moment it is on, then a brief green heartbeat every few seconds. If blue turns into a slow blink instead, the camera found no network and is waiting in its setup AP — check the cable or the router, or join the AP to configure WiFi. (A brief flash of white at power-on, if you catch one at all, is just the light's undriven default before the software takes over — blue is the first deliberate signal.) The full LED vocabulary is in System settings.

Finding the address with a Bluetooth scanner

The camera broadcasts its own IP address over Bluetooth Low Energy: its advertised device name **is** the address. No pairing, app account, or router access needed — any BLE scanner shows it.

1. Install a scanner app such as **nRF Connect** (Nordic Semiconductor, free on iOS and Android) or LightBlue.
2. Stand near the camera and scan. The camera advertises four times a second, so it appears within moments.
3. Look for a name starting with `GG-`:

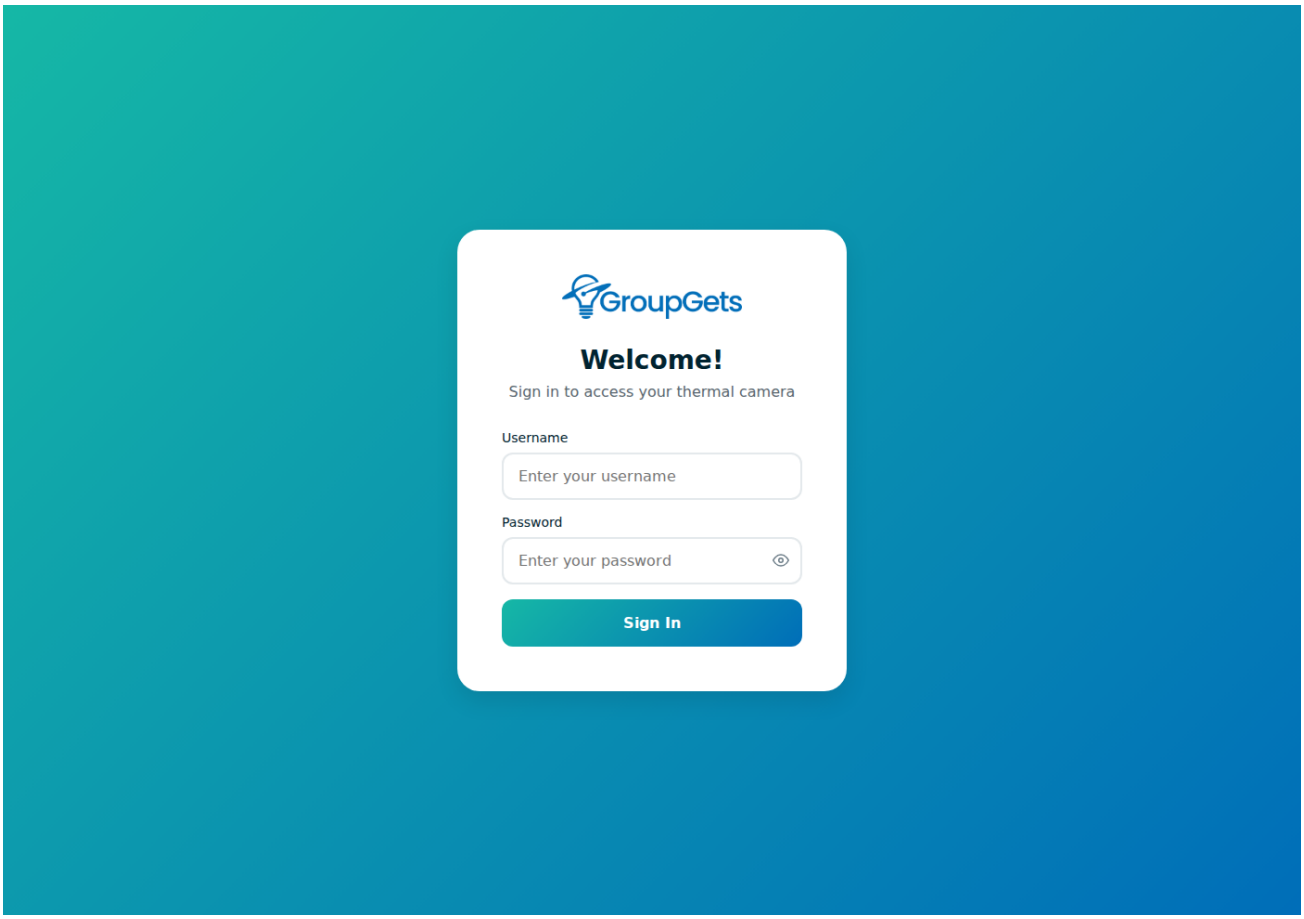
Advertised name	Meaning
`GG-E-192.168.1.42`	On **Ethernet** at `192.168.1.42`
`GG-W-192.168.1.42`	On **Wi-Fi** at `192.168.1.42`
`GG-AP-192.168.4.1`	In **GetThermal-Plus-Setup** mode — join that access point

Then browse to `http://` followed by the address in the name.

The name always reflects reality: the camera does not advertise until it actually holds an address (nothing shows while it is still waiting on DHCP), and the name follows every address or connection-type change within a minute (the advertiser deliberately restarts on a background tick, never mid-change). If several cameras are powering up at once, one you cannot see yet simply has no lease yet — rescan in a few seconds.

Advertising is on by default; the `ble.enabled` setting turns it off (see the appendix for setting it over the API).

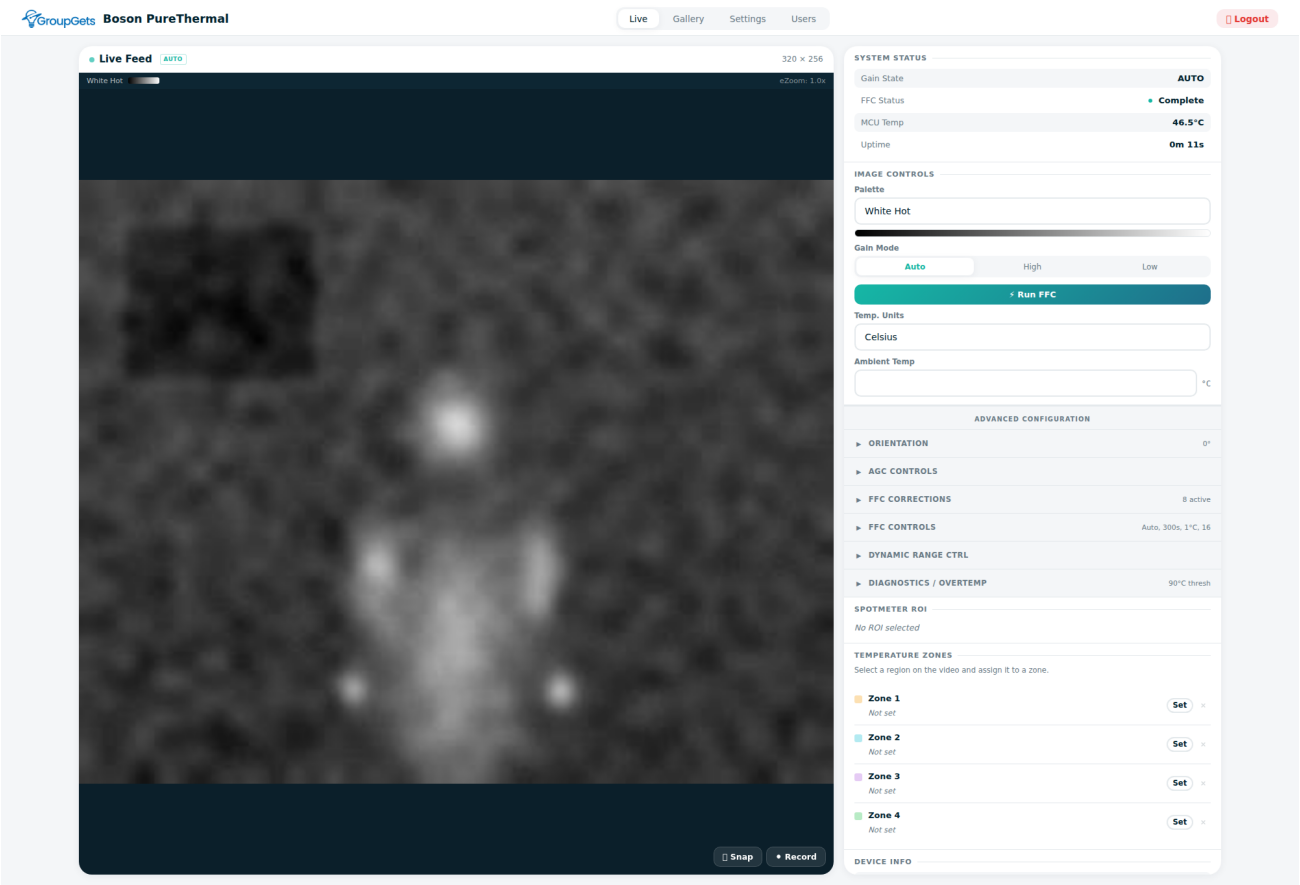
Signing in



The factory account is username `admin`, password `password`. Enter both and press **Sign In**. The eye icon in the password field shows what you typed.

Change the admin password before putting the camera on a shared network — the User accounts tutorial shows how, and it takes under a minute.

After five wrong passwords the camera delays further attempts from your address by five seconds each, so a mistyped password may make the next try feel slow. Signing in lands you on the Live page:



Finding your way around

The navigation bar at the top is the whole map:

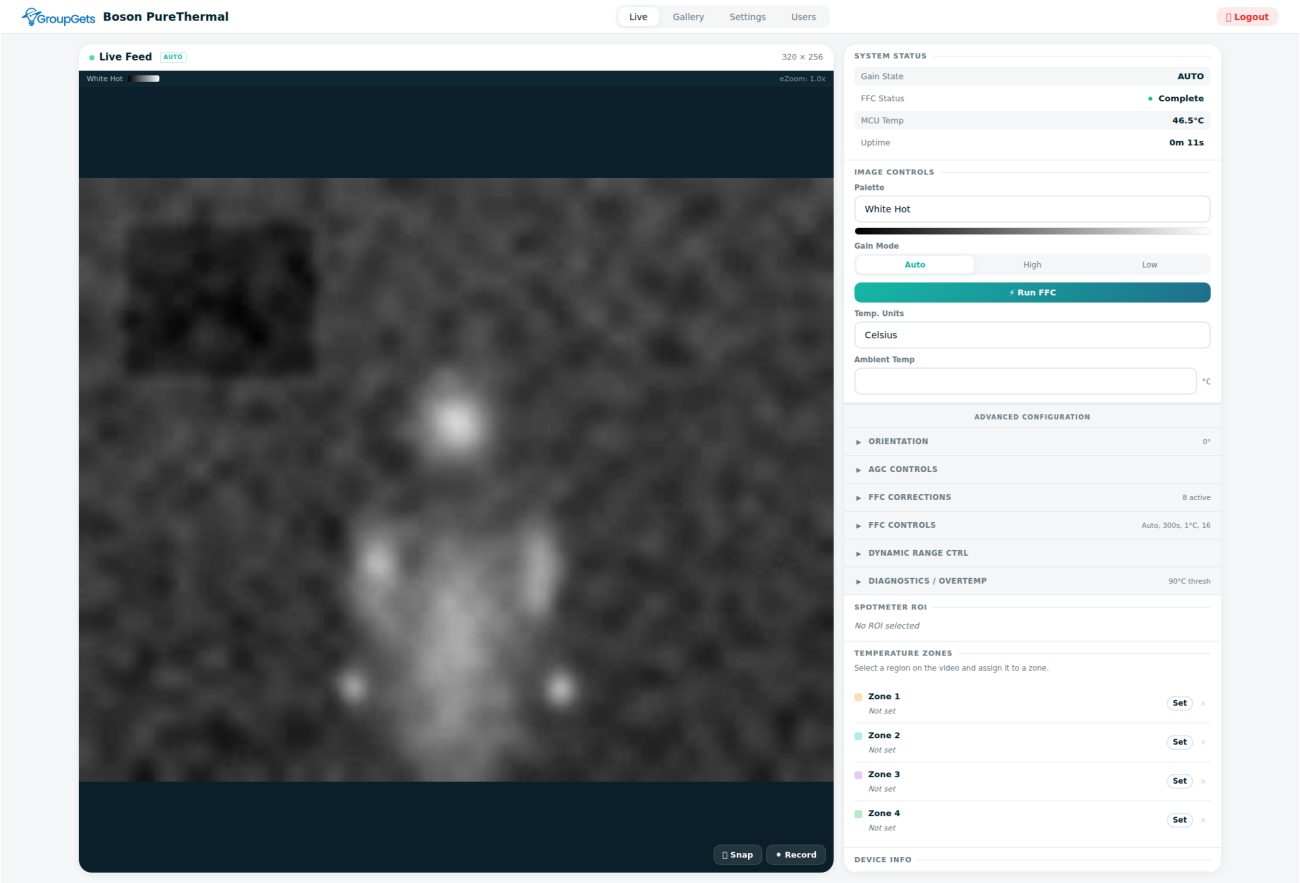
Page	What it holds
Live	The video feed, all camera controls, and sensor status
Gallery	Snapshots and recordings saved on the camera
Settings	Network, RTSP addresses, the status LED, backup, reboot, factory reset
Users	Accounts and passwords

Logout on the right ends the session and returns to the login page. Sessions live in camera memory, so a camera reboot signs everyone out.

Continue with Live view.

Live view

The Live page is where you watch the thermal feed and where every camera control lives.



Reading the screen

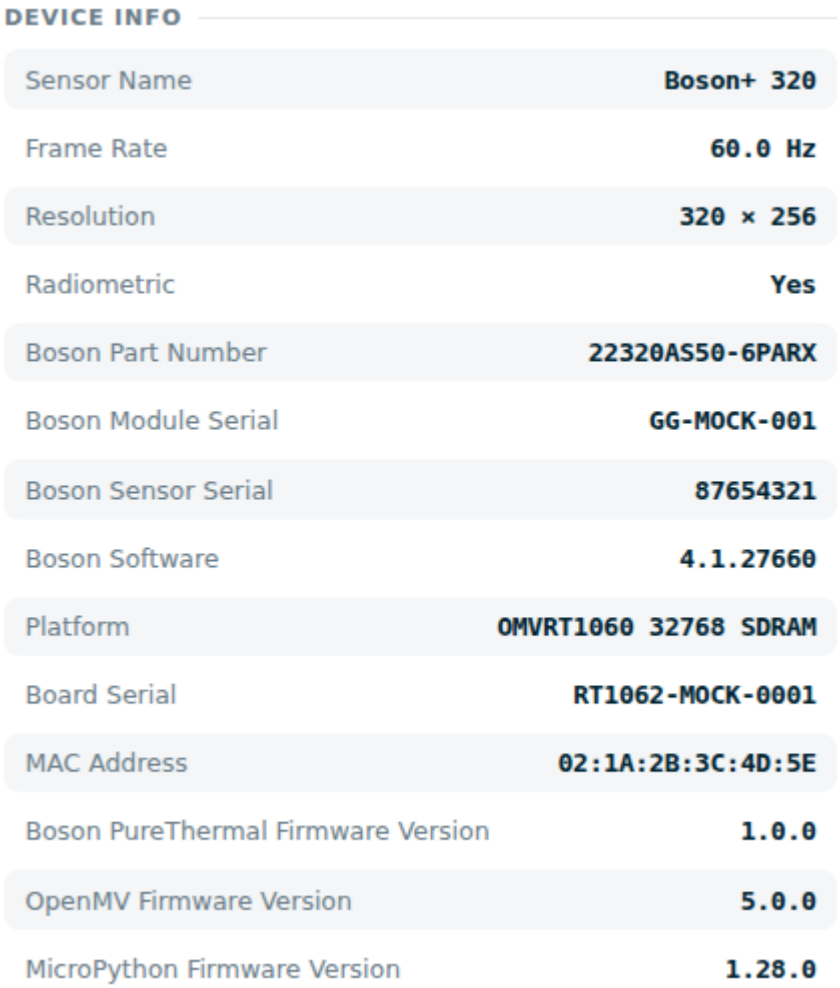
Around the video itself:

- The **strip above the video** shows the active palette with a color swatch on the left and the current eZoom level on the right.
- The **header row** shows a pulsing live dot while real sensor video is flowing, the active gain-mode badge, and the sensor resolution.

To read temperatures, drag a box on the video: the **Spotmeter ROI** panel fills in with that region's minimum, maximum, and mean (on a non-radiometric camera, its raw **Counts** too). For areas you want measured continuously, assign them to the **Temperature Zones**. How the numbers are made, corrected, and calibrated has its own tutorial — Measuring temperature.

The right-hand panel starts with **System Status** — gain state, FFC status, MCU temperature, uptime — refreshed every few seconds. **MCU Temp** is the processor's own die temperature, read from the chip's on-die sensor; it is a board reading, so it stays live even with no thermal core attached. The Boson's own focal-plane-array temperature lives under **Diagnostics / Overtemp** as *FPA Temp* — the camera exposes only that one sensor temperature, so it is not repeated here (on a non-radiometric camera the Temperature Estimation panel shows the same sensor, read live, because it is an input to every estimate). Uptime ticks forward on its own between refreshes; if the camera stops answering for several polls it freezes and turns amber with a "stale" marker, so a dead or disconnected camera can't keep the clock running. At the bottom, **Device Info** identifies the unit.

With **no thermal core attached** the board still boots and the web UI stays reachable — the video area shows a labelled grayscale test pattern instead of thermal video (a real frame stream, so Snap and Record still work against it). Every control the Boson would drive is hidden — Gain Mode, Run FFC, the AGC / FFC / Dynamic Range / Diagnostics accordions, Ambient Temp, and the Spotmeter and Temperature Zones panels — and everything it would report reads `--` rather than a fallback that could be mistaken for a live value.



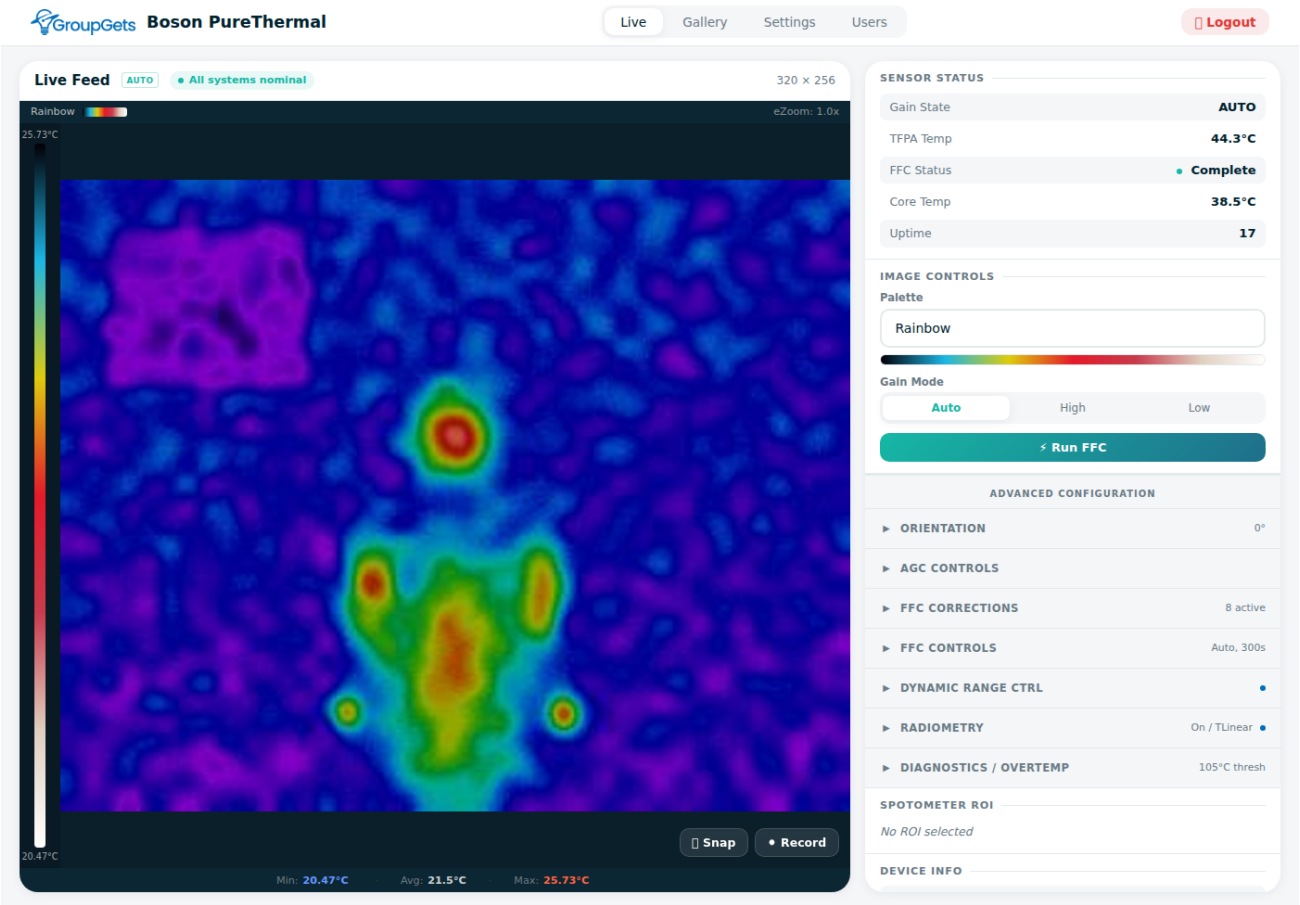
Several of the rows are worth distinguishing when quoting them in a support request:

- **Radiometric** reports whether the thermal core can measure temperature itself. On a radiometric core the spot meter and temperature zones read real, calibrated degrees. On a non-radiometric one the readings are *estimated* from raw sensor counts and every value is prefixed with a `~` to say so — and calibrating tightens them up considerably. Being radiometric is a factory calibration (an **R** in the part number), and is not the same thing as being a Boson+ — see Measuring temperature.
- **Boson Part Number**, **Boson Module Serial**, and **Boson Sensor Serial** all come from the thermal core: the Boson's FLIR part number, the camera module's serial, and the imaging sensor's serial.
- **Board Serial** is the RT1062 processor board's own serial (it is also the camera's network name: `gg-<board serial>` in the router's client list and over mDNS), independent of the thermal core.
- **MAC Address** is the unit's network hardware address — the identity that shows up in a router's DHCP lease list or a switch's MAC table. Ethernet and Wi-Fi share the same MAC on this board, so it is a single value.

- **Boson Software** is the thermal core's firmware revision, read from the sensor itself. Cores still on 2.x firmware predate the temperature command set entirely — the temperature panels lock with a banner saying so (Measuring temperature). **GetThermal+ Firmware Version** is the GroupGets application firmware.
- **OpenMV Firmware Version** is the OpenMV firmware the camera runs on (e.g. `5.0.0`), and **MicroPython Firmware Version** is the MicroPython release inside it (e.g. `1.28.0`) — both fixed by the loaded firmware image, so quote them alongside the GetThermal+ firmware version when reporting an issue.

Palettes

Pick a color mapping from the **Palette** dropdown: White Hot, Black Hot, Rainbow, Rainbow HC, Ironbow, Lava, Arctic, Glowbow, Graded Fire, and Hottest. The swatch bar under the dropdown previews the active gradient.

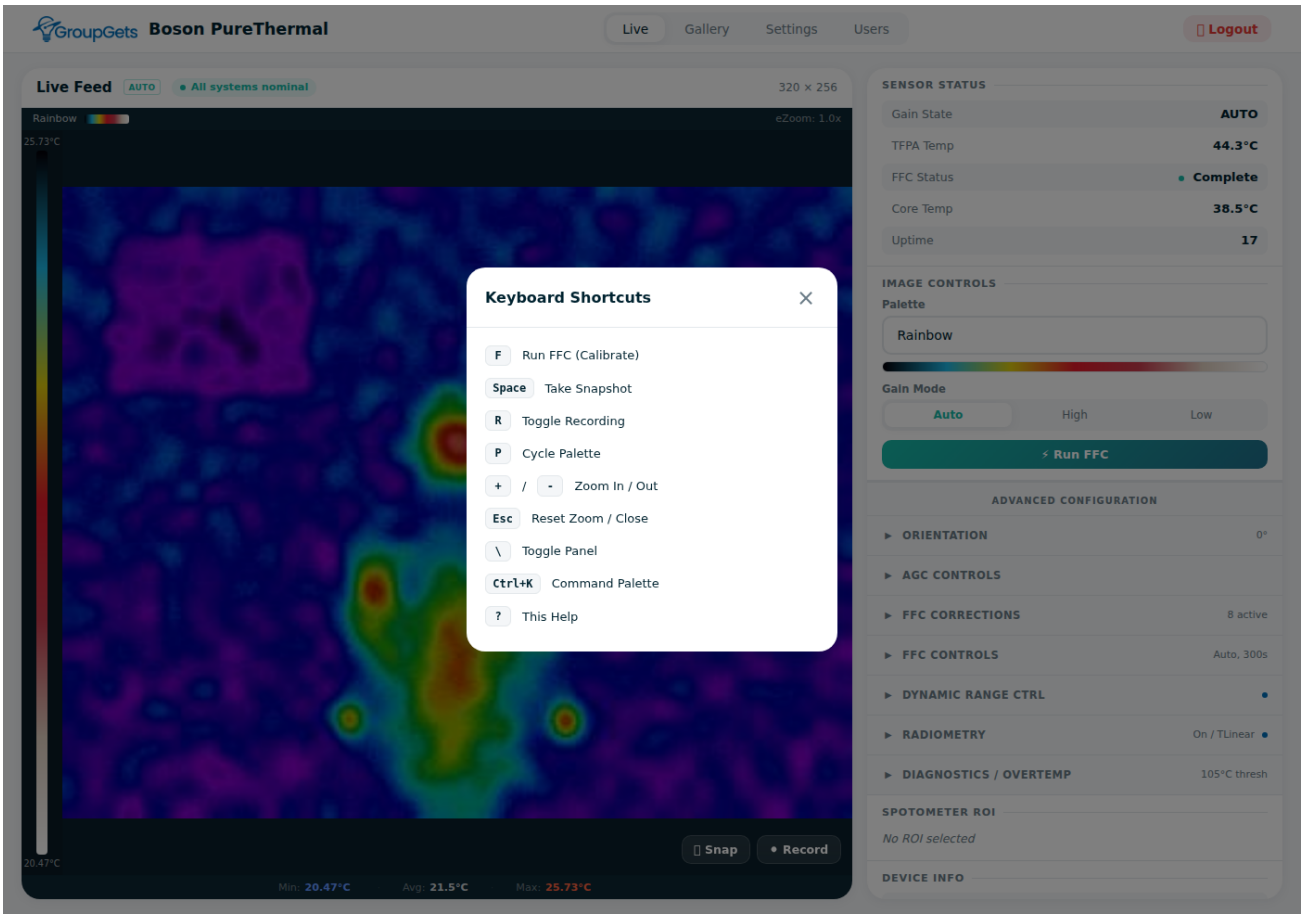


Zoom

Zoom with the `+` and `-` keys or the mouse wheel over the video, up to 8×. While zoomed, a picture-in-picture inset shows the full frame with your viewport outlined — drag the video to pan. `Esc` resets to 1×.

Keyboard shortcuts

Press `?` on the Live page for the full list:



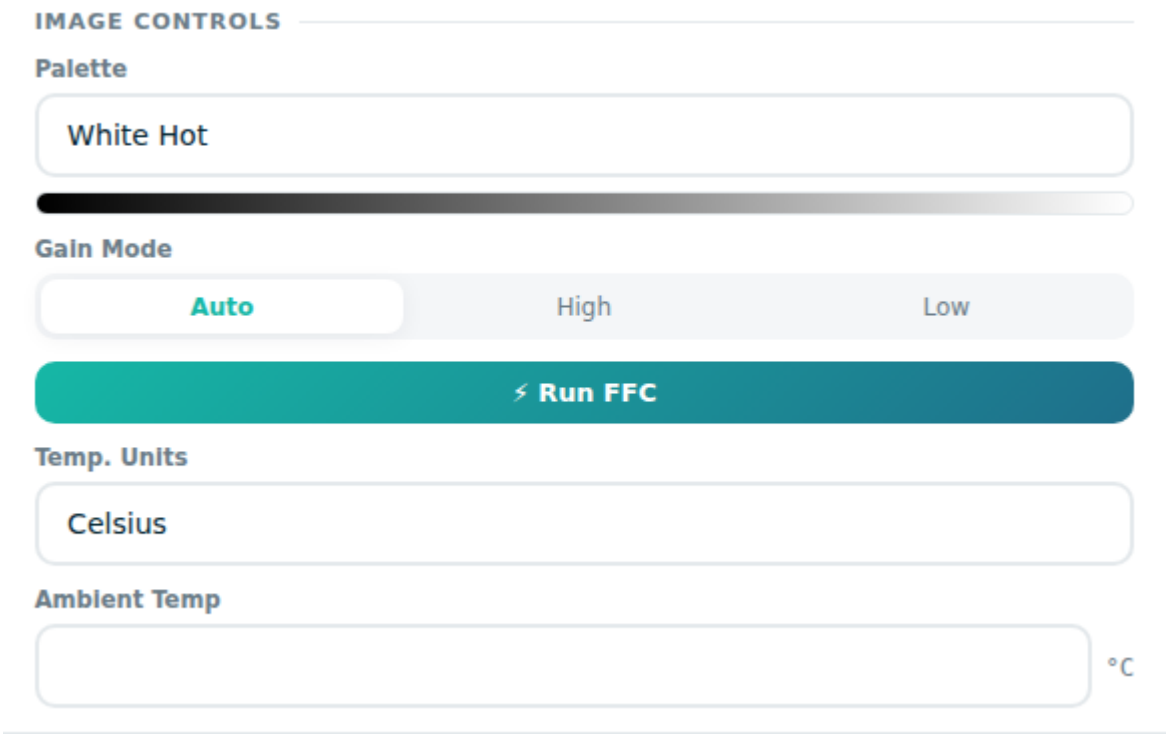
`F` runs a flat-field correction, `Space` takes a snapshot, `R` toggles recording, `P` cycles palettes, and `Ctrl+K` opens a command palette that reaches any control by typing its name.

Continue with Camera controls.

Camera controls

All image and sensor controls live in the panel on the right of the Live page. The essentials sit at the top under **Image Controls**; everything else is grouped into accordions under **Advanced Configuration**. Every change takes effect live and is saved on the camera, so it survives a reboot.

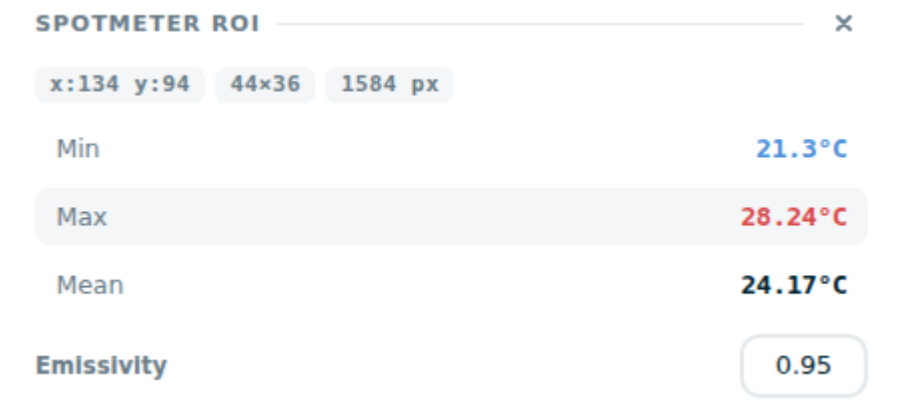
The essentials



- **Palette** — the color mapping, previewed by the swatch bar.
- **Gain Mode** — **High** gain gives the best sensitivity for normal scenes; **Low** gain extends the measurable range for very hot scenes; **Auto** lets the camera switch itself based on scene content.
- **Run FFC** — triggers a flat-field correction: the sensor's internal shutter closes for a moment and the image is re-calibrated. Run one whenever the image drifts or looks noisy. The FFC Status row above shows it completing.
- **Temp. Units** — the units (°C / °F / K) for every temperature the UI shows: the spot meter, the temperature zones, and the diagnostics read-outs all follow this setting. **Ambient Temp** below it is the room temperature that shiny targets reflect — one global value used by the emissivity correction (Measuring temperature).

The spot meter

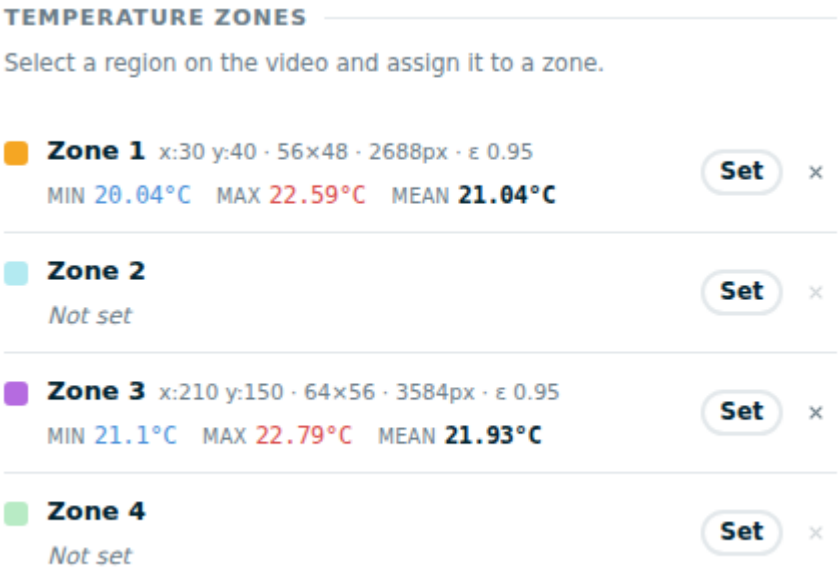
Drag a box directly on the video (at 1× zoom) to measure a region. The **Spotmeter ROI** panel fills in with the region's position and size plus its live minimum, maximum, and mean temperatures — and, on a non-radiometric camera, the raw **Counts** the estimate was converted from. Readings come from the hardware spot meter, whose window maxes out at 128×128 (64×64 on 320 models); a larger box is measured over a centered window of that size:



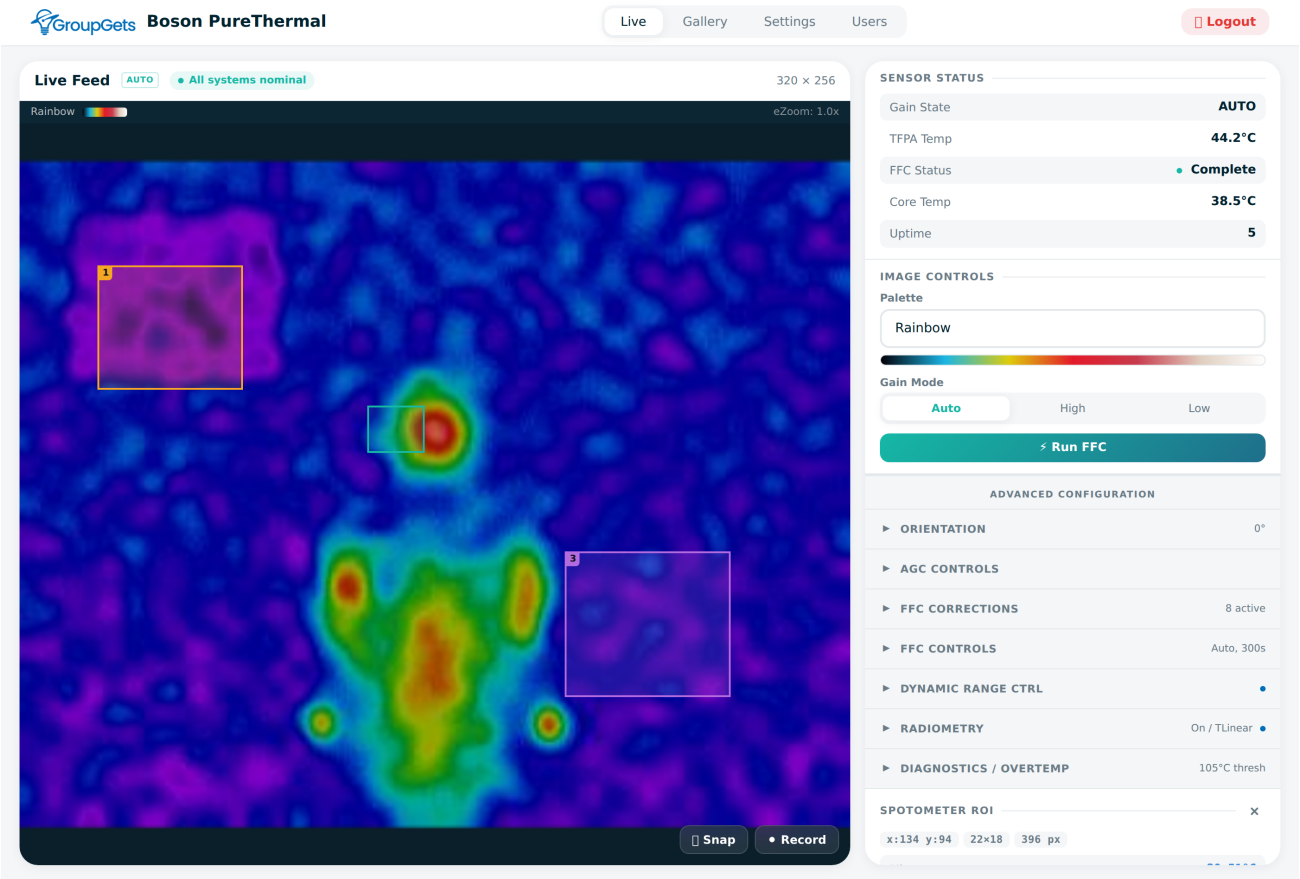
The **Emissivity** box in the panel describes what the region is pointed at (matte surfaces ~0.95, shiny metal far lower) — see Measuring temperature for why it matters enormously on metal. The × in the panel corner clears the region.

Temperature zones

The spot meter is a scratchpad — it follows your last drag. To *keep* an area measured, assign it to one of the four temperature zones in the **Temperature Zones** panel below the spot meter. Drag a region on the video, then press **Set** on Zone 1, 2, 3, or 4: the region is captured to that zone and its minimum, maximum, and mean start updating continuously. Each zone also captures the spot meter's emissivity at the moment you press Set (shown as `ε 0.97` beside its geometry) — set the emissivity first, then Set the zone.



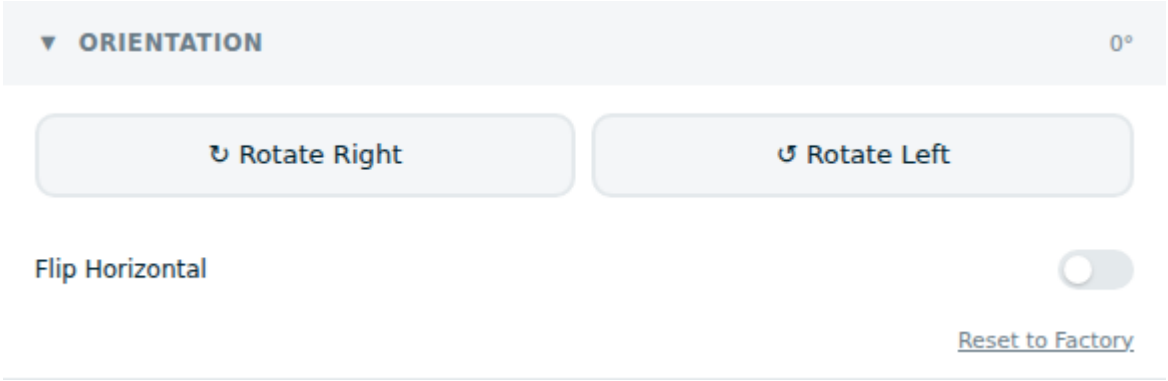
Each active zone draws a translucent coloured rectangle over the video — amber for Zone 1, cyan for Zone 2, violet for Zone 3, green for Zone 4 — matching the swatch next to its readout, so you can always tell which rectangle is which:



The × button beside a zone clears it; the other zones keep their slot and colour (clear Zone 2 and Zone 3 is still Zone 3, not renumbered).

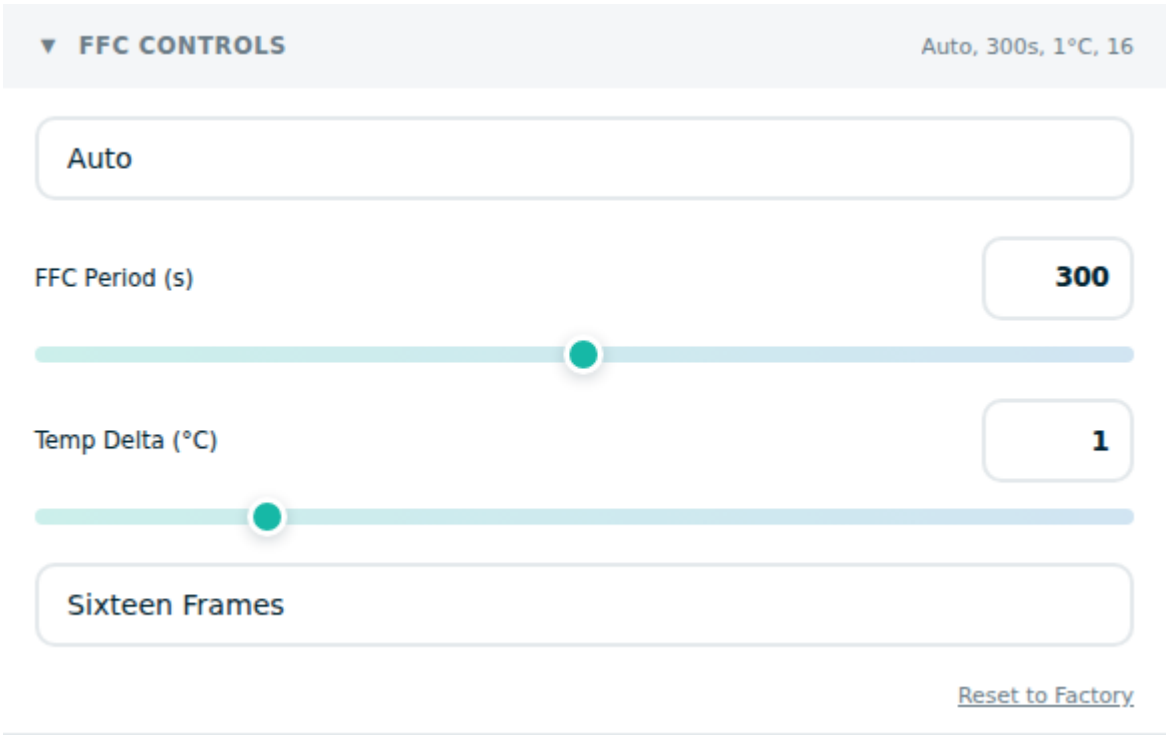
Zones are measured whether or not the browser is open and are served over the network for NVR/RTSP integrations — see *RTSP streaming* for the readings API. Configuration lives here on the Live page; the Settings page only links to the endpoint.

Orientation



Rotate the image in 90° steps and mirror it horizontally to match how the camera is mounted. Rotation and flip apply to everything downstream — the live view, snapshots, recordings, and RTSP.

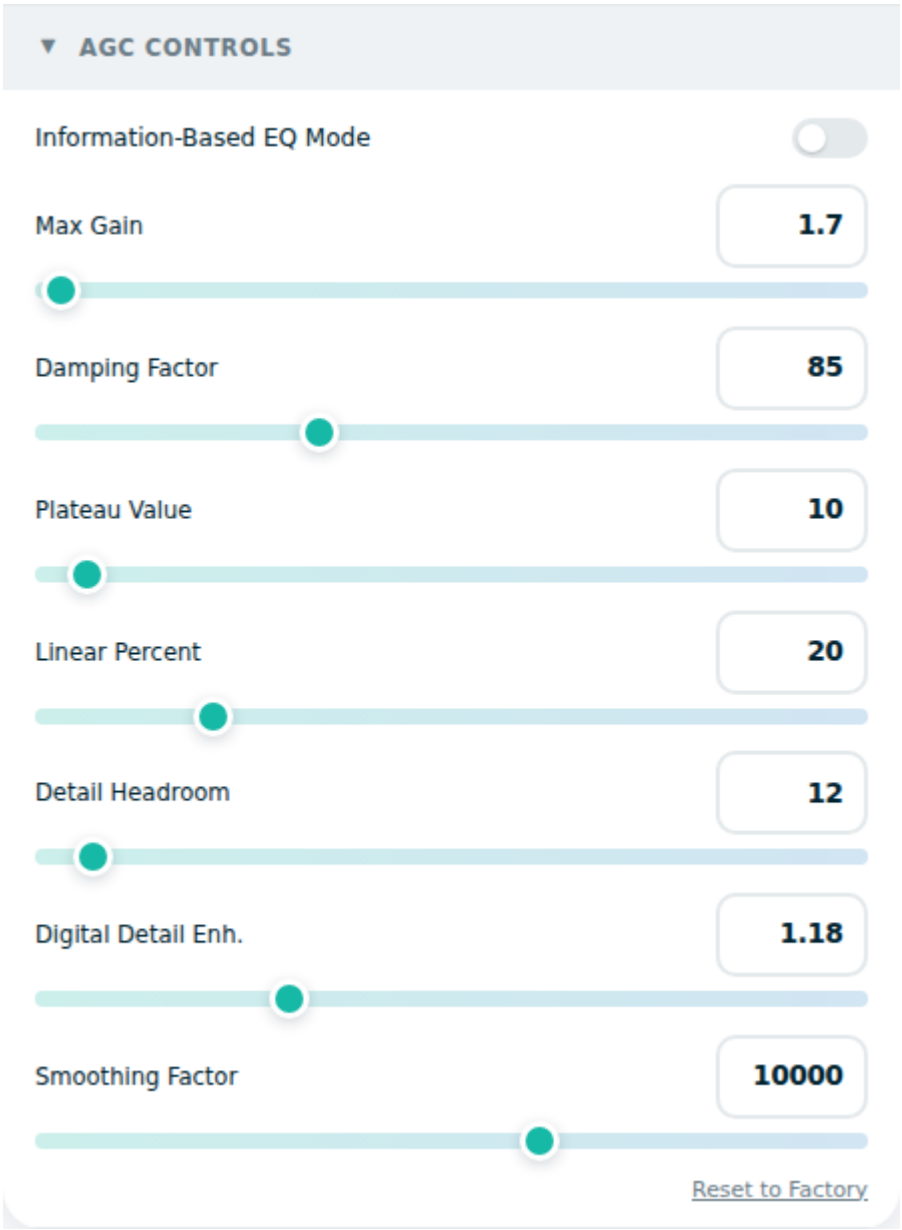
FFC corrections and FFC controls



FFC Controls decides when calibration happens: **Auto** re-calibrates on a period (the slider, up to 600 s) or when the sensor temperature drifts by the set delta; **Manual** calibrates only when you press Run FFC. Integration frames trades calibration speed against noise.

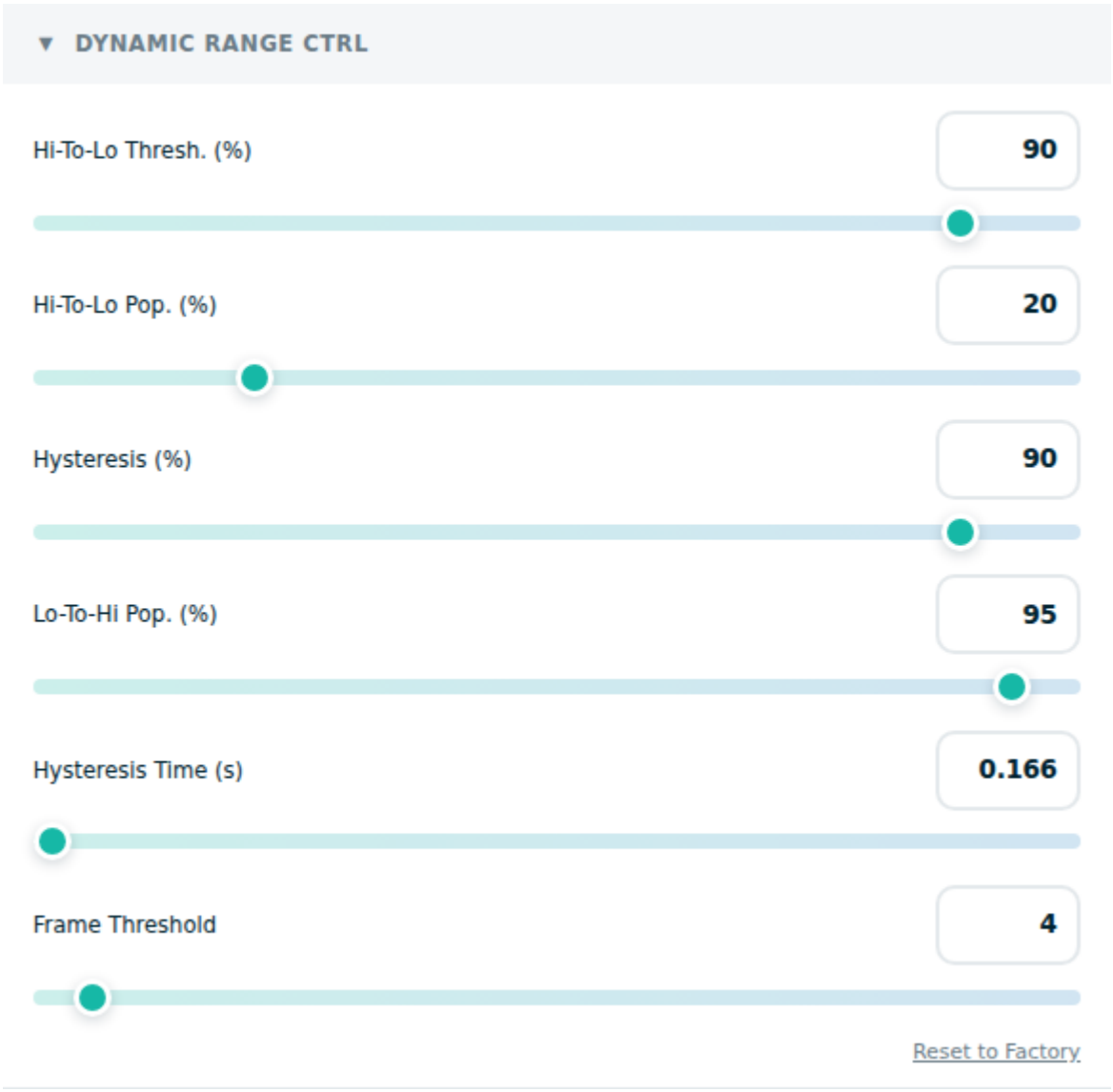
FFC Corrections exposes the sensor's built-in correction stages — defect replacement, column noise reduction, temporal filtering, and the rest — as individual toggles, with a factory reset button. The accordion summary shows how many are active. Leave these at defaults unless you have a datasheet-driven reason.

AGC controls



AGC maps the sensor's wide dynamic range onto the 8-bit image you see, and these sliders tune that mapping: linear percent, detail enhancement, max gain, damping, plateau, smoothing. Sliders a particular sensor rejects hide themselves rather than sit dead. Factory reset restores the balanced default tuning. (The API also exposes named presets — `surveillance`, `inspection`, `high_contrast` — for scripted setups; see the appendix.)

Dynamic range control



When **Gain Mode** (under Image Controls) is set to Auto, these controls govern the switch between high and low gain: what fraction of pixels must saturate before dropping to low gain, when to return, how much hysteresis prevents flapping, and how long (hysteresis time, frame threshold) a condition must hold before the switch fires. They have no effect in High or Low gain.

Radiometry

Radiometry is automatic — there is nothing to turn on. On a radiometric Boson the temperature engine (TLinear) is enabled at every boot, so the spot meter and temperature zones read in real degrees; there is no switch that can disable them.

Whether a camera is radiometric is a hardware property — a factory calibration, flagged by an **R** in the second-to-last digit of the FLIR part number (`…-6PARX` is radiometric, `…-6PAAX` is not) and shown as the **Radiometric** row in **Device Info**. It is *not* the same as being a Boson+: a Boson+ is routinely non-radiometric.

On a non-radiometric core the camera estimates temperatures from the sensor's raw counts and marks every reading with a `~`. Uncalibrated, the estimate can be off by tens of degrees; a one-point calibration on melting ice per gain state makes it genuinely useful, and the **Temperature Estimation** accordion (which only exists on those cameras) is where that happens. Emissivity and ambient-temperature corrections apply on both kinds of core. All of it — the physics, the calibration walkthrough, gain states, and the API — lives in its own tutorial: Measuring temperature.

Cameras whose Boson app firmware is 2.x predate the temperature commands entirely; the temperature panels lock with a banner saying so, the Dynamic Range Ctrl accordion and Ambient Temp input dim with them (2.x lacks the gain-switch commands too), and everything else works normally.

Diagnostics

The last accordion reads back the camera's overtemperature threshold and status, FPA (sensor die) temperature, and low-power state — useful when verifying an enclosure or a hot install. A core running far above its usual FPA temperature can refuse spot-meter reads until it cools (Troubleshooting).

Continue with Measuring temperature.

Measuring temperature

The camera puts numbers on what the thermal image shows. How trustworthy those numbers are depends on which FLIR core is inside, and the UI is honest about the difference at every step — this tutorial explains what you will see on each kind of camera and how to get the best numbers each can give.

Which camera do you have?

Look at the **Radiometric** row under **Device Info** on the Live page.

- **Radiometric: Yes** — the core carries a FLIR factory calibration and measures real, calibrated degrees itself. There is nothing to set up.
- **Radiometric: No** — the core cannot measure temperature on its own. The camera *estimates* temperatures from the sensor's raw counts instead, and marks every such reading with a leading `~`. Estimates get much better once you calibrate — see below.

Radiometric is a factory option, flagged by an **R** in the second-to-last digit of the FLIR part number (`…-6PARX` yes, `…-6PAAX` no). It is *not* the same thing as being a Boson+ — a Boson+ is routinely non-radiometric.

Very old Boson firmware (2.x, sold years ago) predates the temperature commands entirely — see Old Boson firmware at the end.

Reading temperatures

Drag a box on the video: the **Spotmeter ROI** panel shows that region's live minimum, maximum, and mean. The hardware spot meter has a maximum window (128×128 on 640 models, 64×64 on 320s) — a larger box is measured over a window of that size centered inside it. The spot meter follows your last drag — a scratchpad. To keep an area measured continuously, press **Set** on one of the four **Temperature Zones** below it; zones keep measuring whether or not a browser is open, and are served as JSON for scripts and NVRs (RTSP streaming).

On a non-radiometric camera each panel also shows the region's raw **Counts** — the sensor value the estimate was converted from. Together with the FPA temperature and gain on the Temperature Estimation panel, they are everything needed to reconstruct (and therefore defend) a reading. On a radiometric camera there are no counts to show and the rows stay hidden.

The units (°C / °F / K) follow **Temp. Units** in Image Controls, everywhere at once.

Emissivity and ambient temperature

Real surfaces do not radiate perfectly, and shiny ones barely radiate at all — most of what the camera sees off bare metal is the *room reflected in it*. On the bench, a soldering iron set to 750 °F read 385 °F until the camera was told the tip was shiny: at emissivity 0.33, two thirds of what it saw was the reflection.

Two settings correct for this, split by what they describe:

- **Emissivity** is a property of the **target**, so each region carries its own: the spot meter has an emissivity box in its panel, and each zone captures the spot meter's value at the moment you press **Set** (shown as `ε 0.97` in the zone's description; it is not editable after). Rough guide: matte surfaces — skin, paint, tape, water — sit near 0.95–0.98 and barely need correcting; bare or plated metal can be 0.2–0.4 and reads *far* too cold uncorrected.
- **Ambient Temp** is a property of the **room** — the temperature the target reflects. It is one global value, set next to Temp. Units.

Both apply on both kinds of camera. A radiometric core is handed the values and applies the correction itself; a non-radiometric one applies the same correction in the camera's own conversion. The radiometric core holds a single global emissivity, so the camera hands it each zone's value as that zone is sampled — time-shared exactly like the single hardware spot meter itself. Per-zone emissivity therefore works on both kinds of camera.

Calibrating a non-radiometric camera

Uncalibrated, the estimate runs on a shipped default fitted to a different unit — expect it to be off by tens of degrees. One calibration point fixes that.

Open the **Temperature Estimation** accordion (it only exists on non-radiometric cameras). It shows the live inputs — **Counts**, **FPA Temp** (the sensor die's own temperature, which the estimate is anchored on), and **Gain** (the per-unit number calibration solves for) — and the **Calibrated** line, which reads `uncalibrated (default gain)` until you calibrate.

The procedure:

1. **Let the camera warm up** — ten minutes or so, until FPA Temp stops climbing. The fit is taken against the die temperature, and fitting while it ramps bakes the ramp into the number.
2. **Make melting ice.** Crushed ice with a little water, stirred, is 0 °C / 32 °F by physics — no thermometer needed — and matte (emissivity 0.97), so a small emissivity error barely moves the fit.
3. **Aim the spot meter at the ice** — drag a box that is entirely ice.

4. In the Calibrate row, enter the ice temperature in your display units (`32` °F or `0` °C), leave the **ε** box at its `0.97` default (it describes the *reference* you are pointing at, not what you will measure later), and press **Set**.

The Calibrated line fills in with the whole fit — reference temperature, emissivity, the counts it produced, which gain state it was taken in, and the FPA temperature at the time — so the reading is reconstructible later:

```
32°F @ ε 0.97 · 16902 cts · FPA 143.96°F
high gain
```

Sanity-check it: aim at your palm (skin is ε ≈ 0.98) and expect roughly 93 °F / 34 °C.

Calibrate both gain states. The sensor has a high-gain state (normal scenes) and a low-gain state (very hot scenes), and its response differs between them by tens of percent (40% on one bench unit) — so each state carries its *own* gain and its own calibration. With Gain Mode set to **High**, calibrate on the ice; switch to **Low**, calibrate again on the same ice; then set Gain Mode back to **Auto**. Readings always convert with the state the camera is currently in, and the Gain row shows which (`1893419 (high)`). A state you never calibrated keeps its shipped default and says so.

Clear Calibration returns the current gain state to the shipped default. Calibrations persist across reboots, are keyed to the attached Boson (swap cameras and each unit's calibration follows it; a never-seen unit starts uncalibrated), and ride along in configuration export — which means a config exported from one camera carries *that unit's* calibration; recalibrate after cloning (appendix).

What accuracy to expect: a properly calibrated radiometric core is specified around ±5 °C; a calibrated estimate is worse and drifts with conditions (the same bench unit fitted 17% apart on two days at different die temperatures). On the bench, one ice calibration held from 0 °C to 400 °C across four materials — good enough to trust trends and compare regions, not a substitute for an R-suffix core in a report.

The `~` mark

Every estimated value — spot meter, zones, the JSON API — carries a `~` prefix (`approximate: true` in the API) so an estimate can never be mistaken for a measurement, calibrated or not. Radiometric readings never carry it.

Old Boson firmware (2.x)

Boson app firmware 2.x (check **Boson Software** under Device Info) predates the entire temperature command set — the spot meter, zones, and estimation cannot work on it, on any core. The camera detects this at boot and locks the temperature panels: they stay

visible but grayed, under a single banner explaining the firmware is too old to measure temperature. Firmware older still (1.x) also predates the tunable AGC commands and automatic gain switching: the AGC panel is replaced by the same kind of notice, and the Gain Mode control offers only High and Low. The Dynamic Range Ctrl accordion and the Ambient Temp input dim with them — 2.x firmware also predates the gain-switch commands, and ambient feeds the estimation that cannot run. Everything else — video, RTSP, snapshots, FFC, AGC, palettes — works normally. There is no field remedy; FLIR does not distribute the firmware files its update tool would need.

API quick reference

All under the camera's address; endpoints marked (auth) need a session (appendix).

Endpoint	What it does
`GET /api/spotmeter/zones`	Zone readings, no login — includes per-zone `counts`, `emissivity`, `approximate`, plus a `conversion` block (`backing`: `radiometric` / `estimated` / `unavailable`, active `gainState`, gain, calibration provenance)
`GET /api/camera/stats` (auth)	The Live-page spot meter's reading — same per-reading fields (`counts`, `approximate`, …) plus the same `conversion` block
`GET /api/spotmeter/diagnostics` (auth)	The raw inputs behind an estimate: counts, FPA temp, per-state gains, calibration record
`POST /api/spotmeter/calibrate` (auth)	`{"actual": 32, "unit": "F", "emissivity": 0.97}` — fit the active gain state's response
`POST /api/spotmeter/response` (auth)	`{"reset": true}` — clear the active state's calibration
`GET/POST /api/camera/measurement` (auth)	`{"emissivity": 0.95, "ambient": 71.6}` — the spot meter's ε and the global ambient (display units)

Continue with Snapshots and recordings.

Snapshots and recordings

Capturing

The two buttons in the corner of the live video do the work:

- **Snap** (or `Space`) saves the current frame as a JPEG.
- **Record** (or `R`) starts an MJPEG recording; the button turns into **Stop** until you press it again. Recordings cap themselves at **60 seconds** — long-form capture belongs on an NVR recording the RTSP stream (RTSP streaming).

Captures include whatever the live view shows — palette, rotation, and zoom all apply.

Media is stored on the camera's SD card. With no card fitted the buttons say so and refuse to capture — everything else about the camera works without one. The gallery holds up to **200 items** (a recording and its thumbnail count as one); at the cap new captures are refused with a message saying so — the camera never deletes anything by itself. The count shows beside the Gallery title.

Both buttons also refuse until the first video frame has arrived: before the stream has decoded a frame there is nothing to snapshot, and a recording would only create an empty file. They dim to show it, and pressing one tells you why rather than doing nothing. With no thermal core attached the live view streams a test pattern, and that *is* capturable: it is a real frame stream, which makes it a handy way to exercise the whole capture path with no hardware. In the same spirit, a recording that captures no frames is discarded rather than left on the card as a 0-frame `.mjpeg`.

Navigating away from the Live page (or closing the tab) stops an active recording cleanly — the file is finalized on the card rather than left open. Recordings are also flushed to the card every few seconds while running, so even a power cut mid-recording costs the last few seconds, not the whole file.

The buttons on the camera itself

The camera has two physical buttons, useful when it is mounted somewhere a browser isn't:

- **The user button (SW)** captures without the UI:
- **Press** — save a snapshot, exactly as the Snap button does. The status LED answers every saved snapshot — button or web UI — with a **single red pulse**, so you know the shot landed without a screen.

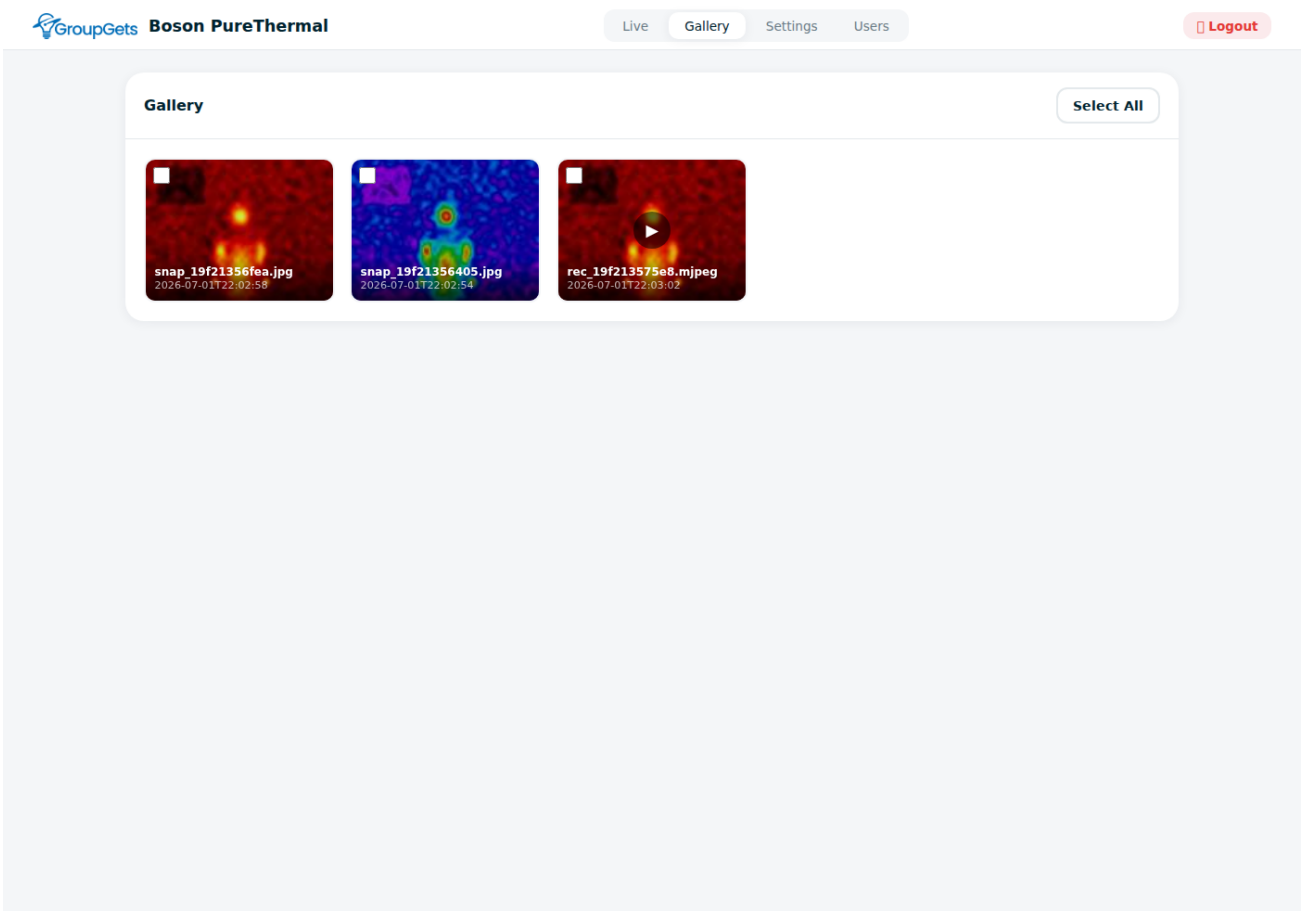
- **Hold** (over half a second) — start a recording; it records while you hold and stops when you release. The status LED turns **solid red** while recording (the status LED), so the camera confirms the hold before you let go.
- If a recording is already running — started from the button *or* the web UI — a press stops it instead of taking a snapshot: one button, and the action in progress always wins.

Button captures go through the same machinery as the UI, so all the same rules apply: media lands on the SD card, the 200-item cap refuses politely, recordings self-cap at 60 seconds, and an empty recording is discarded. With no SD card fitted the button does nothing except log why.

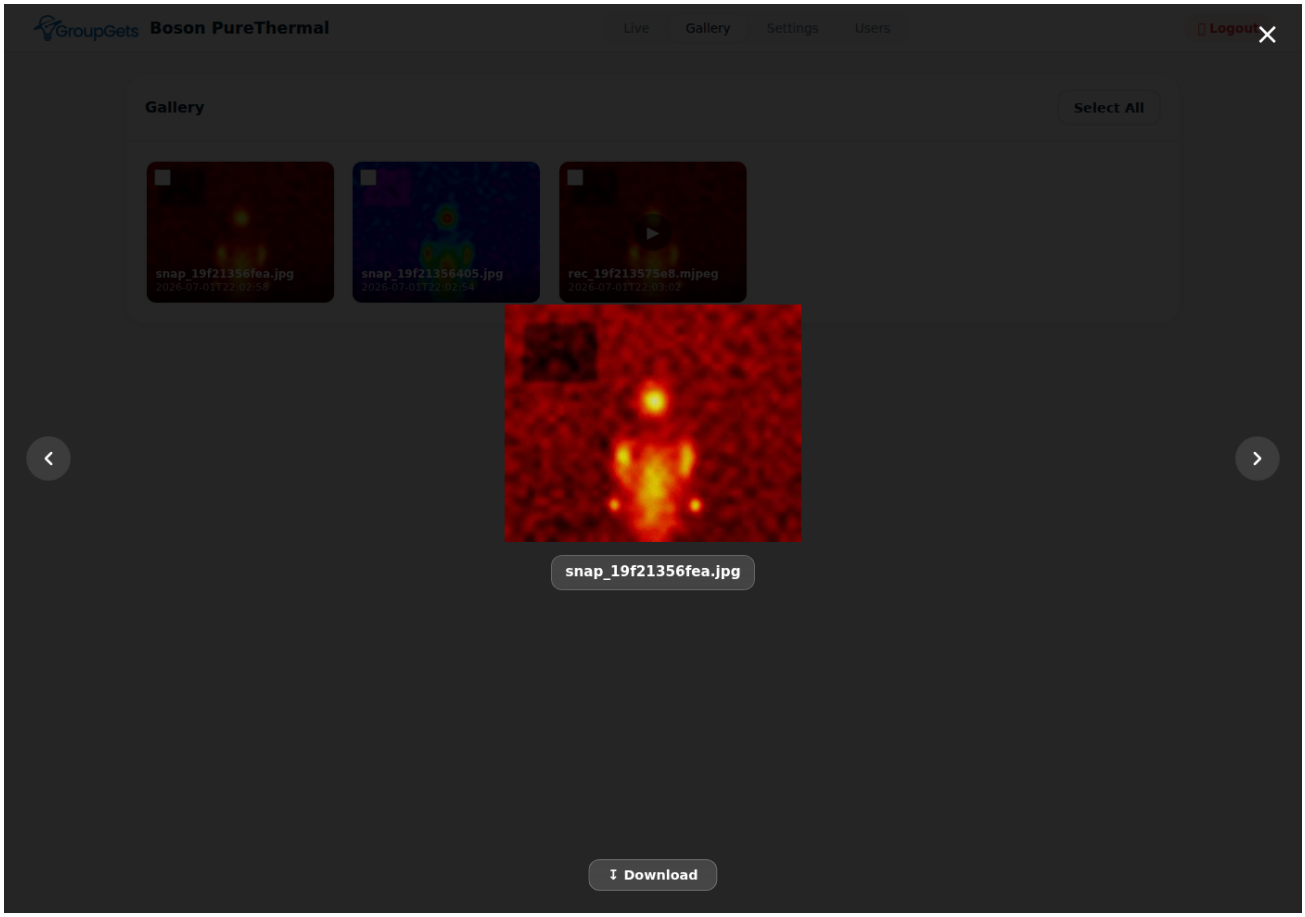
- **The power button (PWR)** is a hardware control, handled by the board's power-management circuitry with no firmware involvement. With the camera powered (PoE or USB), **hold for 5 seconds** to switch it off; when off, **press for 1 second** to switch it back on — the camera boots fresh, like any power cycle. Settings survive — they are saved when you change them — but a recording in progress is cut off the same way a power loss would cut it, keeping everything already flushed to the card.

The Gallery

The **Gallery** page lists everything captured, newest first — snapshots as thumbnails, recordings with a play badge:



Click any item to preview it full-size; arrows step through neighbors:



The download button in the preview fetches that single file.

Downloading and deleting in bulk

Tick the checkboxes on the tiles (or **Select All**), then:

- **Download** packs the selection into a single ZIP.
- **Delete** removes the selection from the camera permanently.

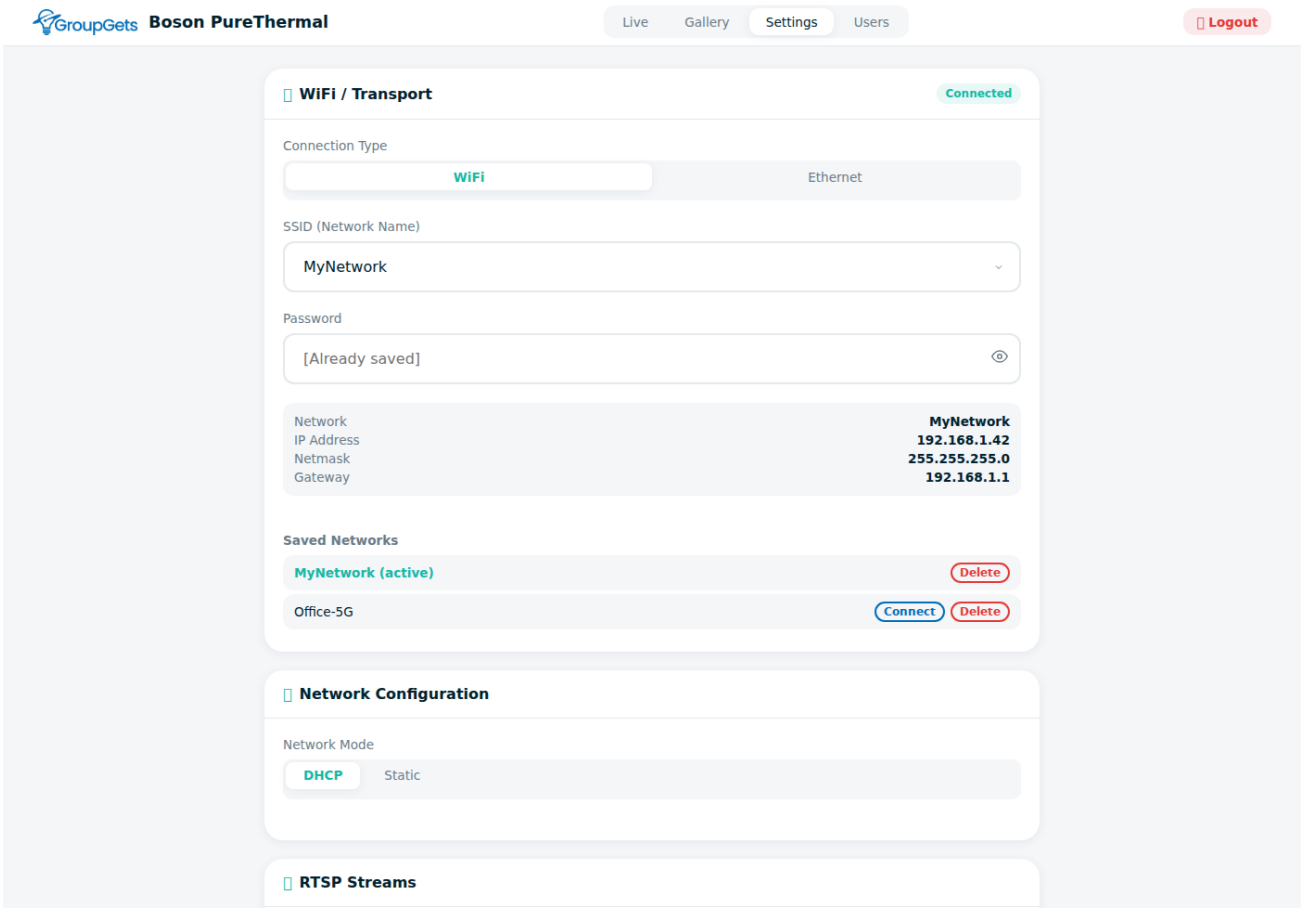
Recordings are `.mjpeg` files — concatenated JPEG frames. VLC and ffmpeg play them directly, and converting to MP4 is one command:

```
ffmpeg -i rec_XXXX.mjpeg -c:v libx264 recording.mp4
```

Continue with System settings.

System settings

The **Settings** page configures how the camera gets on the network, the status LED, and the system-level actions: configuration backup, reboot, and factory reset.



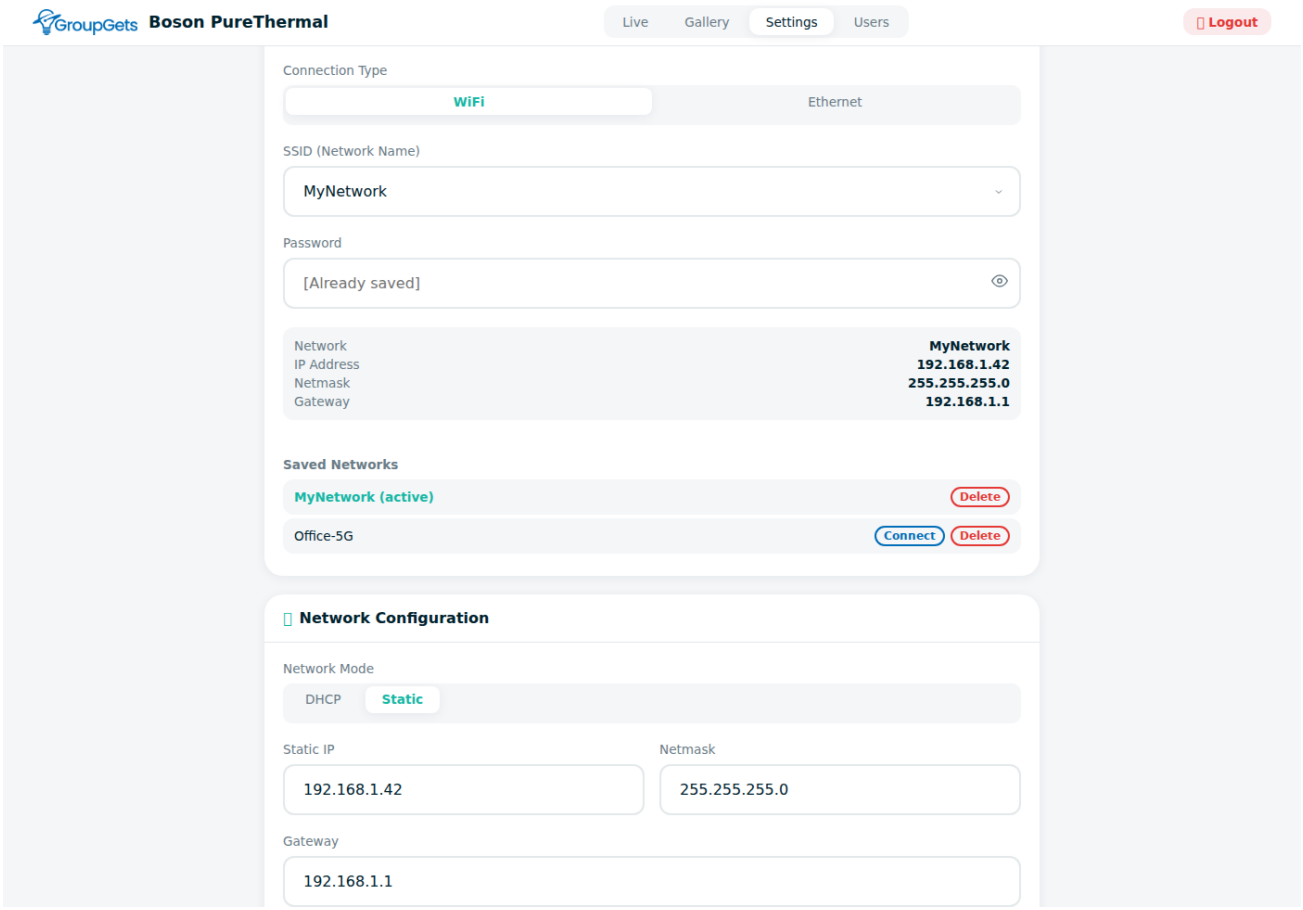
Choosing Wi-Fi or Ethernet

The **Connection Type** toggle at the top picks the transport. On **Ethernet** the camera uses the wired port. On **WiFi**, pick a network from the SSID dropdown (it lists the networks the camera can currently see, plus an **Other Network** entry for typing one in), enter the password, and press the connect button (labelled for the transport you picked — **Connect to WiFi** here). The panel shows the live connection — network name, IP address, netmask, gateway — once the camera is on.

Networks you have joined appear under **Saved Networks** with their passwords stored encrypted on the camera; **Connect** switches between them and **Delete** forgets one.

DHCP or a fixed address

Network Configuration selects addressing. **DHCP** takes whatever the router assigns. **Static** pins the camera to an address you choose:



Enter the IP, netmask, and gateway and save. The camera validates the combination — the address and gateway must sit in the same subnet — and applies it to the active transport.

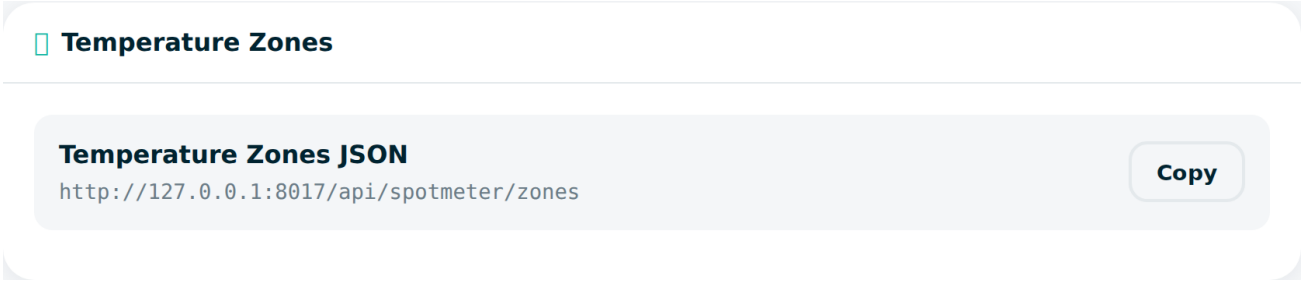
First-time setup mode

A camera with no usable network configuration raises its own access point, SSID **GetThermal-Plus-Setup**, at `192.168.4.1`. Join it and this same settings page opens without login so you can point the camera at your network. As soon as it connects, the access point goes away and the camera is reachable at its new address.

Two deliberate behaviors worth knowing: while the camera is still *trying* to join a saved network, the setup AP stays open — a mistyped password can never strand an unreachable camera, at the cost of the open AP persisting until a network connects. And the AP cannot be stopped by hand while it is the only active connection (the API refuses), for the same reason.

RTSP stream address and temperature zones

The **RTSP Streams** card shows the ready-to-use stream URL for the camera's current address, with a copy button — and below it, the **Temperature Zones** card:

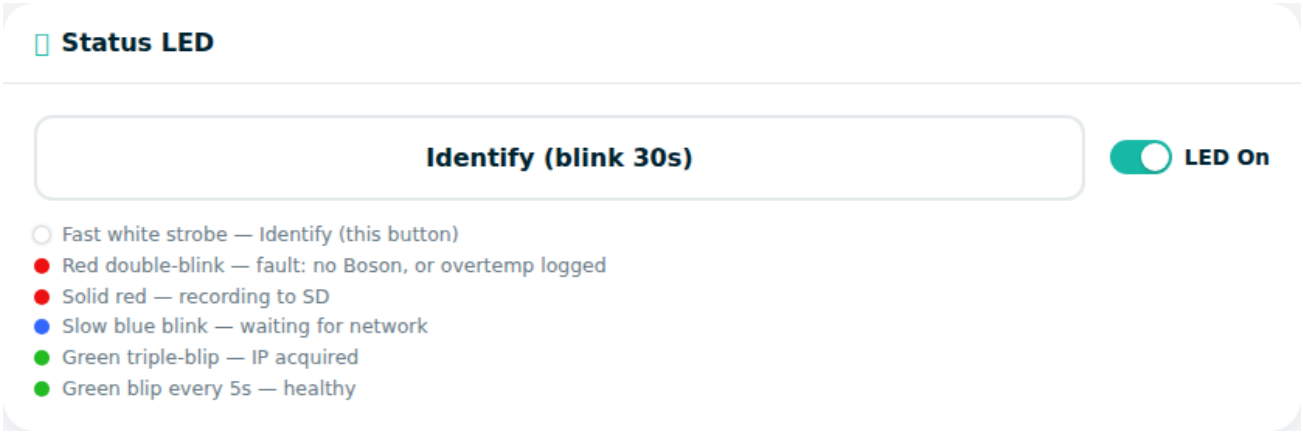


This card shows the readings endpoint and a copy button. The zone temperatures are served as JSON at that address (no login needed), so recorders and scripts can consume temperatures alongside the video — RTSP streaming shows how to read them remotely.

The zones themselves are configured on the Live page: drag a region on the video and assign it to a zone. There is nothing to enter here — the card is just the network hand-off.

The status LED

The **Status LED** card controls the light on the camera body:



It speaks six patterns, one at a time, each distinct by rhythm alone (so they read correctly regardless of color vision or viewing angle):

Pattern	Meaning
Fast white strobe	**Identify** — the button on this card blinks the camera for 30 s so you can find *this* unit among several
Red double-blink	Fault: no Boson found, or the core has logged an overtemperature event

Pattern	Meaning
Solid red	Recording to the SD card — don't unplug
Single red pulse	Snapshot saved (web UI or the camera button)
Solid blue	Booting: reaching for the network (link wait, DHCP)
Slow blue blink	No network connection — still trying, or waiting in the setup AP for you to configure one
Green triple-blip	Network acquired, once — worth watching for after plugging in
Brief green blip every 5 s	Healthy heartbeat

The **LED** switch turns the light off for deployments where a glowing camera is unwelcome. **Identify** still works with the LED switched off — an explicit "show me" outranks stealth.

Backup, reboot, factory reset

The **System** card at the bottom holds the maintenance actions:

- **Export Configuration** downloads the camera's settings as a JSON file — every camera control and network option in one portable backup.
- **Import Configuration** restores an exported file, which is also the way to clone a configuration onto a second camera. The appendix documents every setting in the exported file and how to script export/import.
- **Reboot Device** restarts the camera; the stream and all sessions drop. The camera returns in a few seconds, and everyone signs in again — sessions live in RAM and do not survive a reboot.

- **Factory Reset** returns every setting to factory defaults and clears saved networks and user accounts. The login returns to `admin` / `password`.

Continue with RTSP streaming.

RTSP streaming

Beyond the browser, the camera serves standard RTSP video for players, recorders, and NVR software. The stream is Motion JPEG over RTP — plain, codec-license-free, and understood by everything below.

The address is:

```
rtsp://<camera-ip>:554/mjpeg
rtsp://gg-<serial>.local:554/mjpeg   (where mDNS name resolution is available)
```

The Settings page shows it filled in for the camera's current address, with a copy button (see System settings).

VLC

Media → Open Network Stream, paste the URL, **Play**. Or from a shell:

```
vlc rtsp://192.168.1.50:554/mjpeg
```

ffplay / ffmpeg

```
ffplay rtsp://192.168.1.50:554/mjpeg
```

Recording the stream to a file without re-encoding:

```
ffmpeg -i rtsp://192.168.1.50:554/mjpeg -c copy -t 60 clip.avi
```

NVR and other consumers

Any consumer with generic RTSP support works: point it at the URL, and if it asks for a codec, choose MJPEG. The camera does not require RTSP authentication.

Practical notes:

- The camera serves up to **3 RTSP sessions at once** and rejects further SETUP requests. The browser MJPEG stream does not count against that limit, though an active browser viewer shares the frame budget, so RTSP frame rate dips while someone is also watching in a browser.

- Transport is **UDP only**. A client that insists on TCP-interleaved transport (`-rtsp_transport tcp`) is refused with `461 Unsupported Transport` — leave players on their default UDP setting.
- Palette, orientation, and all image controls apply to RTSP output too — the RTSP feed is the same frames the browser sees. While an RTSP client is connected, the grayscale palettes (White Hot, Black Hot) are encoded as color JPEGs internally — RTP's JPEG format cannot carry monochrome, and strict players like VLC render it as garbage. The picture looks identical; it switches back to the leaner monochrome encoding when the last RTSP client disconnects.
- Idle sessions are reaped after 60 seconds without keepalives.

Temperature zones

Video shows *where* something is hot; the zones API tells you *how* hot. The camera continuously measures up to five areas of the scene — the Live-page spot meter plus four zones — and serves the readings as JSON, with no login required — the same trust model as the streams, so an NVR or monitoring script can consume video and temperatures side by side.

Zone 0 is always the spot meter on the Live page. Up to four more zones are configured on the Live page — drag a region on the video and assign it to a zone. The **Settings** page links to the readings endpoint (System settings). Reading the zones is one request:

```
curl http://<camera-ip>/api/spotmeter/zones
```

```
{"ok": true, "maxZones": 5, "temperaturesAvailable": true,
 "conversion": {"backing": "estimated", "gainState": "high",
                "gain": 1893419.0, "approximate": true,
                "calibration": {"referenceK": 273.15, "gainMode": "high",
                                "counts": 16902, "emissivity": 0.97,
                                "fpaC": 62.2}},
 "zones": [
  {"zone": 1, "name": "door", "roi": {"x": 10, "y": 10, "w": 40, "h": 40},
   "kelvin": {"min": 294.6, "max": 296.0, "mean": 294.9},
   "display": {"min": 21.4, "max": 22.8, "mean": 21.7, "unit": "C"},
   "emissivity": 0.97, "approximate": true, "counts": 21605,
   "ageMs": 312, "stale": false}
]}
```

Every reading carries Kelvin values, the configured display unit, the zone it was measured over, its emissivity, and its age — the camera samples the zones in the background, so polling this address is instant and never slows the video. Sampling runs while zones are configured or readings were requested in the last few seconds, and idles otherwise; the first request after an idle spell may report `stale` readings while fresh samples arrive. An `error`

field on a zone names the reason there is no number. Zone 0 in the list is the Live-page spot meter (`maxZones` counts it).

The top-level `conversion` block says how to interpret the numbers:

- `"backing": "radiometric"` — the sensor measured real, factory-calibrated degrees. `approximate` is `false` and every `counts` field is `null` (test for null, not for the key's absence).
- `"backing": "estimated"` — a non-radiometric core: the camera converted raw counts itself. Every reading has `"approximate": true` (render it with a `~`), each zone carries the `counts` it was converted from, and the block includes the active gain state, the gain in use, and the calibration it came from — everything needed to reconstruct a reading (Measuring temperature).
- `"backing": "unavailable"` — this Boson's firmware predates the temperature commands; no temperatures will ever arrive. The response also sets `"temperaturesAvailable": false` and lists no zones.

Consumers should check the top-level `temperaturesAvailable` flag before trusting per-zone values; it is `false` only when no thermal core is attached or the core cannot serve temperature data at all. The same flag appears on `GET /api/camera/stats`.

For push instead of poll, the same address is also a server-sent-event stream — ask for it with the standard header (or `?stream=1`) and the camera emits a reading snapshot every half second:

```
curl -N -H 'Accept: text/event-stream' http://<camera-ip>/api/spotmeter/zones
```

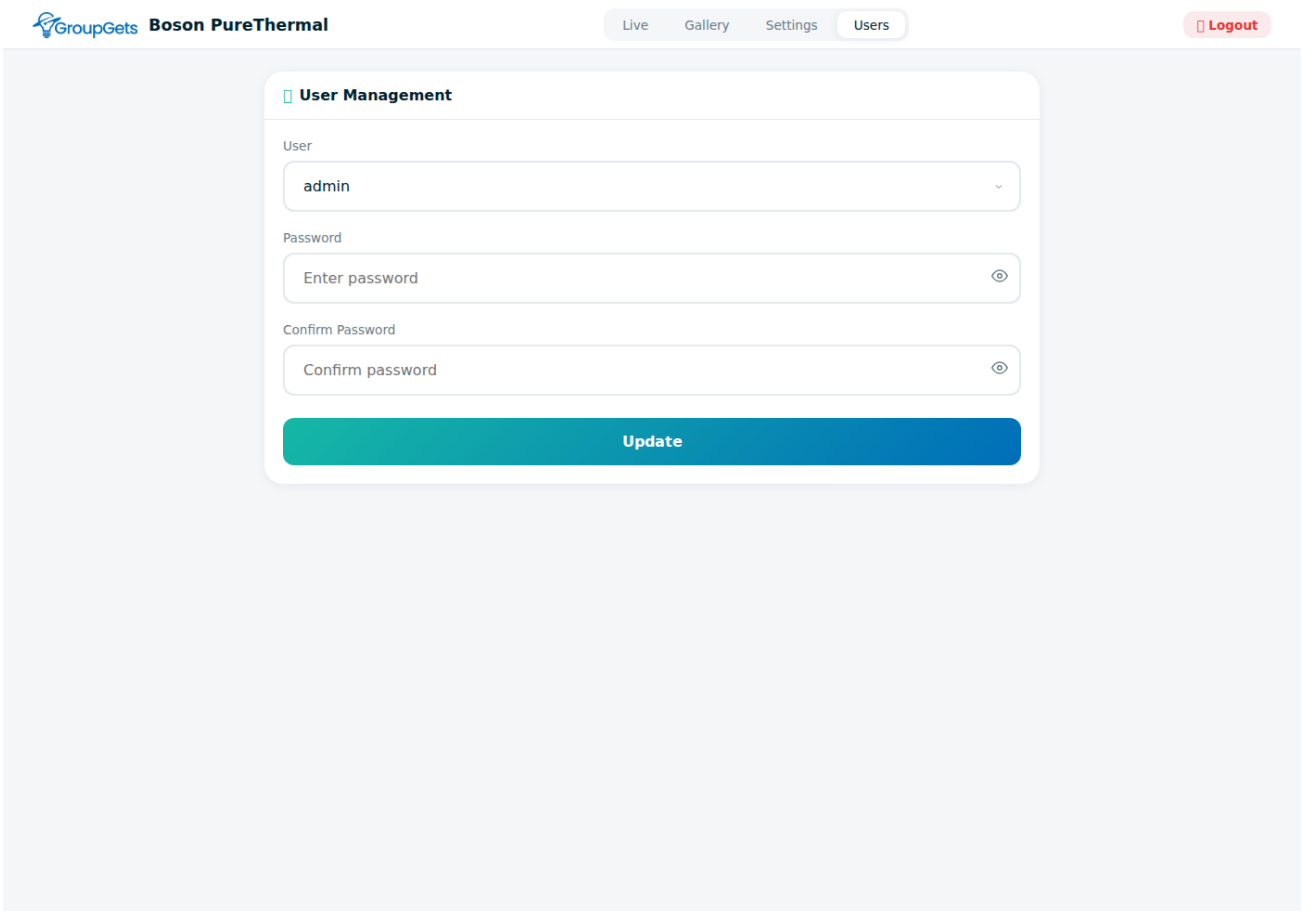
A handful of event-stream clients can subscribe at once; beyond that the camera answers `503` rather than degrade the video — fall back to polling for large fan-outs.

Zones ride along in configuration export/import, so a provisioned camera comes up measuring the right areas (appendix).

Continue with User accounts.

User accounts

The **Users** page manages who can sign in. Every account sees the same camera; accounts exist so people have their own credentials, not to grant viewing permissions. Signed in as `admin`, the page manages every account; other accounts can change their own password and nothing else — the API enforces that split, not just the page.

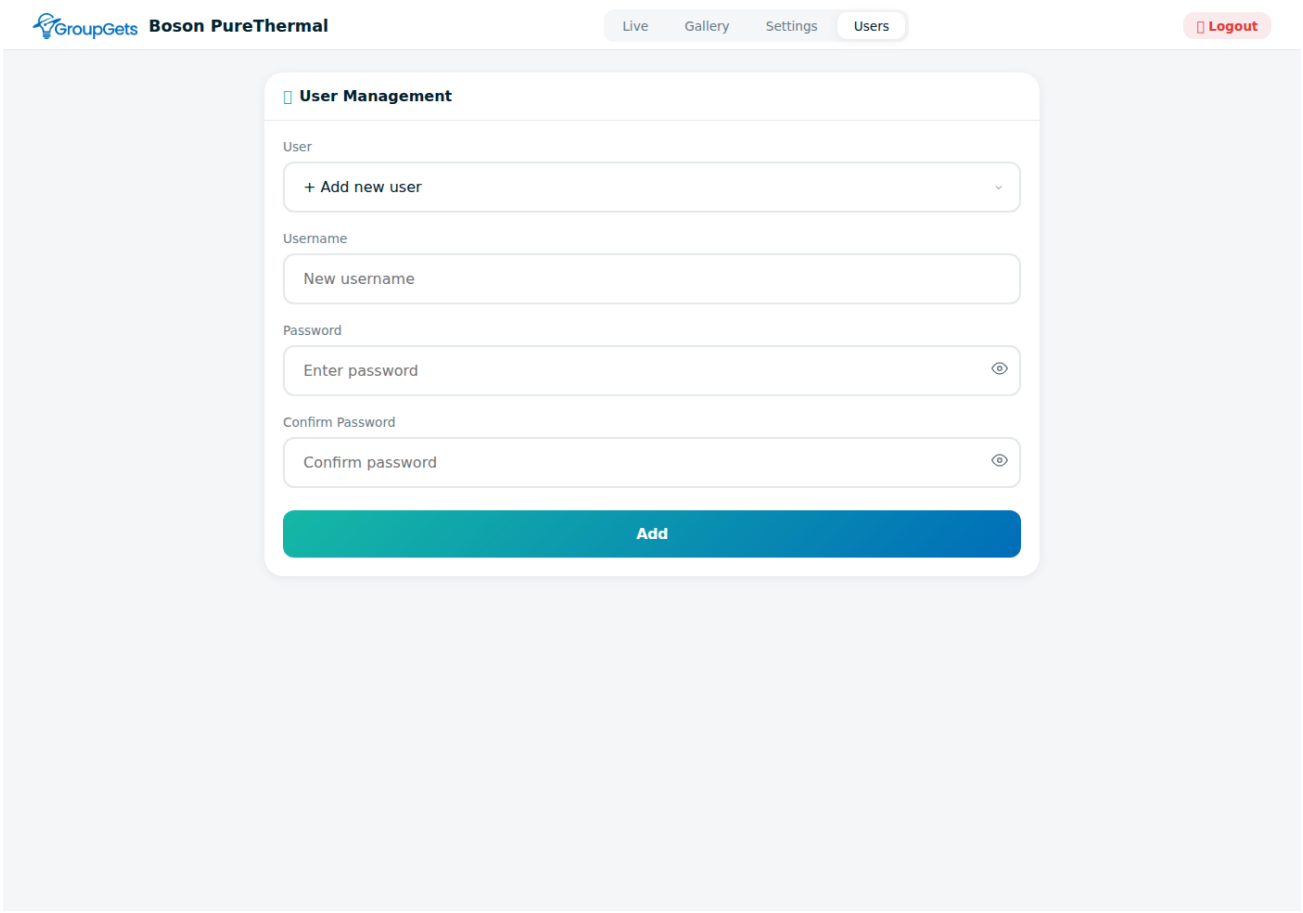


Changing a password

Pick the account from the dropdown, type the new password twice — the eye icons reveal what you typed — and press **Update**. Do this for `admin` on day one: the factory password is public knowledge.

Adding a user

Choose **+ Add new user** in the dropdown, fill in the username and both password fields, and press **Add**:



The new account can sign in immediately.

Removing a user

Select the account and press **Remove**. The `admin` account cannot be removed, so there is always a way in.

Good to know

- Passwords are stored salted and hashed on the camera, never in plain text. A configuration export contains no passwords.
- Changing or removing an account does not kick its active sessions; a reboot clears all sessions.
- Five failed sign-in attempts from one address slow further attempts by five seconds each — brute-forcing the login form is deliberately tedious.
- Lost the admin password? Factory-reset the camera to restore `admin` / `password` (this also clears settings and saved networks).

Setting up more than a handful of cameras? The appendix documents every camera setting and how to script accounts, Wi-Fi, and configuration across many cameras.

Updating the firmware

GroupGets publishes firmware updates for the camera as a single zip package. Inside it:

File	What it is
`firmware.bin`	The camera firmware
`romfs0.img`	The camera's built-in web interface
`getthermal-plus-user-guide.pdf`	This guide
`README.txt`	Package version and a quick-start summary
`sha256sums.txt`	Integrity checksums
`license.txt`	License terms

Updating keeps everything the camera remembers — settings, user accounts, and saved Wi-Fi networks all survive a firmware update.

What you need

- The update package as downloaded — no need to unzip it (this guide is inside it too).
- A USB cable between the camera and a computer.
- **OpenMV Viewer**, the free desktop tool that loads firmware onto the camera — download it from openmv.io/pages/download for Windows, macOS, or Linux.

Loading the firmware

1. Connect the camera to the computer over USB.
2. In OpenMV Viewer choose **Tools → Load Custom Firmware**.
3. Select the downloaded zip package itself. OpenMV Viewer flashes everything inside it — the camera firmware and the web interface — in one pass. Do not unplug the camera while the progress bar runs.
4. Reconnect the camera to its network and sign in. **Device Info** at the bottom of the Live page shows the versions now running — **GetThermal+ Firmware Version** for the application, plus the **OpenMV** and **MicroPython Firmware Versions** it ships with (Live view).

The camera keeps its network configuration, so it comes back at the same address — and if it got a new one from DHCP, a Bluetooth scan reads the new address off the air (Getting started).

Configuring many cameras after an update? The appendix covers exporting and importing whole configurations.

Appendix: camera settings and provisioning

Setting up one camera is a few minutes of clicking; setting up a hundred should not be a hundred times that. This appendix covers how the camera stores what it remembers, the export and import controls in the web UI, every key in the configuration, and the HTTP API that lets a script do the clicking for you.

What the camera stores

Everything the camera remembers lives inside the camera itself; a fitted SD card holds only snapshots and recordings.

State	Where it lives	Moves between cameras?
Settings — camera, network, BLE, LED, spotmeter	Inside the camera	Yes — the exported configuration carries all of it. One caveat: the spotmeter's temperature calibration is device-unique (see the provisioning notes)
User accounts	Inside the camera	Set on each camera; passwords are stored as salted hashes and never exported
Saved Wi-Fi networks	Inside the camera	**No** — each camera encrypts Wi-Fi passwords with its own unique key, so no other camera can use them
Snapshots and recordings	SD card	n/a

Two consequences of that split are worth knowing before provisioning:

- **Settings never touch the SD card.** Removing, swapping, or reformatting the card cannot lose configuration, and cloning a card from a configured camera clones nothing but its media. Without a card the camera runs normally; only Snap and Record are unavailable.
- **Settings are seeded once.** On first boot the camera fills in its built-in factory defaults and never overwrites your choices after that — firmware updates keep your settings. Only a factory reset returns everything to defaults (clearing accounts and saved networks with it).

Changing settings in the web UI

Day to day, settings change in the page where they live: camera and image controls on the **Live** page (Live view, Camera controls), network and RTSP on the **Settings** page (System

settings), and accounts on the **Users** page (User accounts). Every change is stored on the camera the moment it is made — there is no separate save step to forget.

For whole configurations at once, the **Settings** page has the controls that matter to this appendix:

- **Export Configuration** downloads every setting as one JSON file named `groupgets_config_<serial>_<date>.json` — a backup, and the template for configuring the next camera.
- **Import Configuration** loads an exported file and applies it immediately — on the camera it came from or on any other.
- **Factory Reset** returns a camera to a known starting state.
- **Reboot Device** restarts it.

For a handful of cameras, that is the whole workflow: configure one unit until it is exactly right, export, and import on the rest. Everything below exists for when clicking stops scaling — the same export, import, account, and Wi-Fi operations, driven over HTTP from a script.

Configuration reference

An exported configuration has five sections — `camera`, `network`, `ble`, `spotmeter`, and `system`. Import stores unknown keys verbatim inside `camera`, `network`, and `ble`; the `spotmeter` and `system` sections are whitelist-filtered to the keys below, and anything else in them is dropped. The tables list what the camera itself reads and writes.

`camera`

Key	Factory default	Meaning
`palette`	`"white_hot"`	Color mapping: `white_hot` (true grayscale), `black_hot`, `rainbow`, `rainhc`, `ironbow`, `lava`, `arctic`, `glowbow`, `graded_fire`, `hottest`. The retired `grayscale` ID is still accepted and stored as `white_hot`.
`rotation`	`0`	Image rotation: `0`, `90`, `180`, or `270`
`hmirror`	`false`	Horizontal mirror
`gain_mode`	`"auto"`	Sensor gain: `auto`, `high`, or `low`
`ffc_mode`	`"auto"`	Flat-field correction: `auto` or `manual`
`ffc_period_s`	`300`	Seconds between automatic FFCs
`ffc_temp_delta_c`	`1.0`	

Key	Factory default	Meaning
		Temperature drift (°C) that triggers an FFC
`ffc_integration_frames`	`16`	Frames averaged during an FFC
`temp_units`	`"C"`	Spot-meter units: `C`, `F`, or `K`
`zoom_level`	`1.0`	Digital zoom, `1.0`–`8.0`
`agc_preset`	`"surveillance"`	AGC tuning preset: `surveillance`, `inspection`, `high_contrast`, or `custom` (set automatically when a slider is moved off a preset)
`tlinear`	`true`	TLinear (radiometric pixel) output. Forced on at every boot — it is the always-on engine behind temperatures and is no longer user-toggleable
`dr_hi_to_lo_thresh`	`90`	Gain-switch threshold, high→low (% of range)
`dr_hi_to_lo_pop`	`20`	Pixel population for high→low switch (%)
`dr_lo_to_hi_pop`	`95`	Pixel population for low→high switch (%)
`dr_hysteresis`	`90`	Gain-switch hysteresis (%)
`dr_hyst_time`	`0.166`	Gain-switch hysteresis time (seconds a condition must hold)
`dr_frame_thresh`	`4`	Frames a gain-switch condition must persist
`radiometry_enabled`	`true`	Temperature-stabilization compensation. Forced on at every boot; no longer user-toggleable
`corr_ffc`, `corr_auto_gain_corr`, `corr_defect_replace`, `corr_scnr`, `corr_temporal_filter`, `corr_silent_nuc`, `corr_suppl_ffc`, `corr_lfsr`, `corr_srnr`	see file	Boson image-correction stages (booleans). Importing pushes them straight to the sensor; the camera otherwise treats them as sensor-owned state
`frame_rate_target`	`60.0`	Capture rate: `9.0` or `60.0`
`averager_enabled`	`false`	Frame averager on/off
`averager_count`	`1`	Frames averaged: `1`, `2`, `4`, or `8`

Key	Factory default	Meaning
`frame_skip`	`0`	Frames skipped between served frames, `0`–`16`
`video_source_enabled`	`true`	Video pipeline on/off
`video_source`	`"post_colorize"`	Tap point: `post_colorize` or `pre_agc`

`network`

Key	Factory default	Meaning
`mode`	`"dhcp"`	`dhcp` or `static`; static mode requires all three address fields
`transport`	`"auto"`	`auto` (Ethernet when its link is up, else Wi-Fi), `ethernet`, or `wifi`
`active_network`	`""`	SSID (from the saved list) the camera joins on boot
`auto_connect`	`true`	Join the active network automatically
`static_ip`	`"192.168.50.197"`	Static address (used when `mode` is `static`)
`netmask`	`"255.255.255.0"`	Static netmask
`gateway`	`"192.168.50.1"`	Static gateway
`captive_dns`	`true`	In setup-AP mode, answer every DNS query with the camera's address

`ble`

Key	Factory default	Meaning
`enabled`	`true`	BLE advertising on/off. The advertised name is always auto-derived — `GG-E-<ip>` (Ethernet), `GG-W-<ip>` (Wi-Fi), or `GG-AP-<ip>` (setup mode) — so any phone with a BLE scanner reads the camera's address off the air (Getting started walks through it)
`name`	`""`	Reserved; the firmware currently ignores it and always advertises the auto-derived name

`spotmeter`

Key	Factory default	Meaning
`zones`	`[]`	Up to four temperature zones measured continuously, each `{"x", "y", "w", "h"}` in sensor coordinates (unrotated — the Live page converts what you draw) plus an optional `"name"` and the `"emissivity"` captured from the spot meter when the zone was Set. Zone 0 (the Live-page spot meter) is not stored here. Readings are served without login at `GET /api/spotmeter/zones` (JSON, or server-sent events with `Accept: text/event-stream`); configuring zones requires a session — RTSP streaming covers the read side
`emissivity`	`0.95`	The spot meter's emissivity — a property of the target it is pointed at (Measuring temperature)
`ambient_c`	`22.0`	The reflected (room) temperature, °C — one global value used by the emissivity correction
`planck_b`	`1450.0`	The Planck-curve constant of the counts→temperature estimate. Leave it alone
`response_gains`	per model	Non-radiometric estimation: the fitted response gain per gain state (`{"high": …, "low": …}`). Written by calibration; **per-unit** — see the provisioning caveat below
`calibrations`	`{}`	Per-gain-state calibration provenance (reference temperature, emissivity, counts, FPA temp). Travels with `response_gains`

Settings follow the Boson

The sensor-owned slice of settings — everything in `spotmeter`, plus the camera keys that tune the core itself (gain, FFC, AGC preset, dynamic range, correction stages, the radiometry enables, frame rate) — is keyed to the attached Boson's **part number + serial**. Swap cameras and each one's zones, calibration, and tuning come back; a Boson the board has never seen starts from factory defaults instead of inheriting another unit's calibration or a different resolution's zone coordinates. Two bookkeeping keys appear in exports: `active_sensor` (who the live sections belong to) and `sensor_profiles` (the stashed slices of previously attached cameras). Board and viewer settings — network, LED, palette, rotation, temperature units — stay put across swaps.

`system`

Key	Factory default	Meaning
`led_enabled`	`true`	

Key	Factory default	Meaning
		The status LED (System settings). Off keeps the camera dark; the Identify strobe still works

Logging in from a script

Every configuration endpoint requires a session. `POST /login` with form fields `username` and `password`; script clients receive `{"ok": true}` and a session cookie named `s` (browsers get a redirect instead). Sessions live in RAM, so log in again after any reboot. Five failed attempts from one address add a five-second delay to each try.

```
CAM=http://192.168.50.197
curl -s -c jar -d 'username=admin&password=password' $CAM/login
```

Every example below reuses the cookie jar with `-b jar`.

Exporting and importing over HTTP

`GET /api/system/config/export` returns the same JSON the **Export Configuration** button downloads:

```
curl -s -b jar $CAM/api/system/config/export > golden.json
```

The export contains every settings section. User accounts and saved Wi-Fi networks are deliberately excluded — passwords never leave the camera.

`POST /api/system/config/import` accepts the same shape, whole or partial — send only the keys you want to change and everything else keeps its current value:

```
curl -s -b jar -H 'Content-Type: application/json' \
  -d @golden.json $CAM/api/system/config/import
```

Import does three things in order: applies hardware-backed values (gain, FFC mode, zoom, corrections, the AGC preset) to the sensor immediately, stores the merged settings in the camera, and — about half a second after replying — re-applies the network configuration if the import changed it. That delay lets the `{"ok": true}` response reach you before a new IP address or transport drops the connection; expect to reconnect at the camera's new address afterwards. An import that sets `"mode": "static"` with missing or empty address fields backfills them from the camera's current settings, then factory defaults — so a partial static import succeeds rather than erroring. The resulting combination is then validated structurally (dotted IPv4 form, usable host, contiguous netmask, gateway on the same

subnet) and a failure rejects the import with HTTP 400 before anything changes. Include all three fields explicitly when scripting static addressing.

Users and Wi-Fi

Accounts and Wi-Fi credentials are the two things an exported configuration does not carry, so a script sets them on each camera:

```
# Change the admin password (do this on every deployed unit)
curl -s -b jar -H 'Content-Type: application/json' \
  -d '{"username":"admin","password":"NEW-PASSWORD","action":"update"}' \
  $CAM/api/users

# Add an operator account ("delete" removes one; admin cannot be removed)
curl -s -b jar -H 'Content-Type: application/json' \
  -d '{"username":"operator","password":"OP-PASSWORD","action":"add"}' \
  $CAM/api/users

# Join and save a Wi-Fi network
curl -s -b jar -H 'Content-Type: application/json' \
  -d '{"transport":"wifi","ssid":"SiteNetwork","password":"WIFI-PASSWORD"}' \
  $CAM/api/system/wifi
```

Account passwords are hashed, and Wi-Fi passwords are encrypted with a key unique to each camera, before either is stored — which is also why Wi-Fi credentials cannot be cloned from one camera to another. The camera's API is plain HTTP, so provision on a trusted network.

Provisioning many cameras

Putting the pieces together: configure one camera in the web UI until it is exactly right, export it as the golden config, then replay that config onto every unit:

```
#!/usr/bin/env bash
# provision.sh — replay golden.json onto every camera in cameras.txt
set -euo pipefail

ADMIN_NEW='NEW-ADMIN-PASSWORD'

while read -r ip; do
  cam="http://${ip}"
  jar="$(mktemp)"
  echo "=== ${ip}"
  curl -sf -c "${jar}" -d 'username=admin&password=password' "${cam}/login" > /dev/null
  curl -sf -b "${jar}" -H 'Content-Type: application/json' \
    -d @golden.json "${cam}/api/system/config/import"
```

```
  curl -sf -b "${jar}" -H 'Content-Type: application/json' \
    -d "{\"username\":\"admin\",\"password\":\"${ADMIN_NEW}\",\"action\":\"update\"}" \
    "${cam}/api/users"
  curl -sf -b "${jar}" "${cam}/api/system/config/export" > "verify-${ip}.json"
  rm -f "${jar}"
done < cameras.txt
```

Practical notes for the batch run:

- **Building `cameras.txt`:** every powered-on camera advertises its IP address as its Bluetooth device name (`GG-E-<ip>` / `GG-W-<ip>`), so a phone running a BLE scanner collects the whole room's addresses without touching the router — Getting started walks through it.
- **Strip the calibration before cloning.** A non-radiometric camera's temperature calibration (`spotmeter.response_gains` / `spotmeter.calibrations`) is fitted to *that unit's* sensor — cloned onto another camera it is confidently wrong. `jq 'del(.spotmeter.response_gains, .spotmeter.calibrations)' golden.json` strips it; calibrate each unit on melting ice instead (Measuring temperature).
- **Leave `network` out of `golden.json`** (or import it last) when cameras should keep DHCP — a golden file exported from a static-IP camera would move every unit to the same address. `jq 'del(.network)' golden.json` strips it.
- **Factory-known state:** to start truly from scratch, `POST /api/system/factory` first. It returns every setting to factory defaults, clears saved Wi-Fi networks, resets the accounts to `admin`/`password`, and reboots immediately — the connection drops without a response, so give the camera time to boot and log in fresh.
- **Reboots:** `POST /api/system/reboot` behaves the same way. Neither touches the SD card.

That is the whole loop: one well-configured camera, one exported file, and a script that repeats it as many times as the job needs.

Troubleshooting

Most problems come down to one of the situations below.

- **The camera does not show up on the network.** Look at the status LED first: a slow blue blink means it is still waiting on a DHCP lease (System settings has the full LED vocabulary). Check the router's client list for `gg-<serial>`, or try `http://gg-<serial>.local` — the serial is the **Board Serial** shown in Device Info. Then scan for it with a Bluetooth scanner — the camera's advertised name is its IP address (Getting started). If no `GG-` name appears, the camera has no address yet: it is still waiting on DHCP, so rescan in a few seconds. A name of `GG-AP-192.168.4.1` means the camera opened its own **GetThermal-Plus-Setup** access point because it could not join a network — join that access point to configure it.
- **The camera became unreachable after changing network settings.** The Bluetooth name always shows the address the camera actually holds, so a scan tells you where it went — browse there. If it gave up on the new settings, it appears as `GG-AP-192.168.4.1` and setup mode is open again.
- **The admin password is lost.** A factory reset returns the login to `admin` / `password`. It also returns every setting to factory defaults and clears user accounts and saved Wi-Fi networks, so export a configuration backup first if you can still sign in with another account (System settings).
- **Signing in feels slow.** After five wrong passwords from one address, the camera delays each further attempt by five seconds. Type carefully and the delay stops counting against you after a correct sign-in.
- **Everyone got signed out at once.** Sessions live in camera memory, so any reboot or power cycle signs everyone out. An open page notices on its next refresh and returns to the login screen by itself.
- **Temperature readings say `--` or "Sensor control failed".** Three common causes, in order of likelihood:
- **The Boson core is very hot.** Check **FPA Temp** (Diagnostics, or the Temperature Estimation panel): well above where it usually sits — say beyond 70 °C — the core can refuse spot-meter reads until it cools. Improve the enclosure's airflow.
- **Boson app firmware 2.x.** Check **Boson Software** under Device Info: 2.x cores predate the temperature commands entirely, and the camera locks the temperature panels with a banner saying so. Video and everything else still works (Measuring temperature).

- **A moment of flat-field correction.** A single blank reading during the periodic FFC shutter click is normal and heals itself within a second or two.
- **Estimated temperatures are far off.** A non-radiometric camera runs on a shipped default until calibrated on this specific unit — one melting-ice calibration per gain state fixes it (Measuring temperature).
- **Snap and Record are grayed out.** No SD card is fitted (or the card is not readable). Snapshots and recordings need a card; everything else about the camera works without one (Snapshots and recordings).
- **An RTSP player will not connect.** Check the URL against the one shown on the Settings page (`rtsp://<camera-ip>:554/mjpeg`), and count viewers: the camera serves up to three RTSP sessions at once, so close one before opening another (RTSP streaming).

Getting help

If a problem is not on this list, contact GroupGets through groupgets.com. Two pieces of information make any support request faster to answer, and both are on the Live page under **Device Info** (Live view):

- The versions the camera is running — **GetThermal+ Firmware Version**, plus the **OpenMV** and **MicroPython Firmware Versions** listed under it.
- The camera's **Boson Module Serial**.

Add what you were doing, what you expected, and what happened instead — a snapshot or a photo of the screen helps too.