Water Flow Monitor - Project Documentation

Nov 12, 2026 · @kner – Elaborated by Claude

Overview

Two ESP32-S3 boards detect running water in a pipe by its sound and report it on a web page and two LEDs. The sensor node listens with a microphone and decides ON or OFF; the server node shows the state, raises an alarm when water runs too long, and keeps a log of every flow.

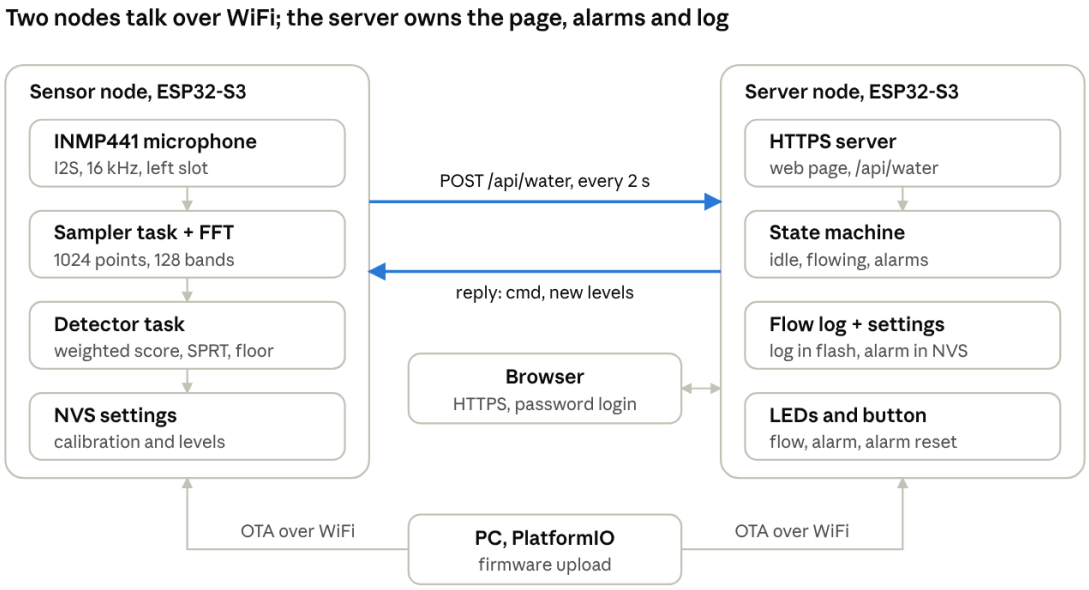
Part	Role	Notes
Sensor node	Microphone, FFT, detector, calibration	ESP32-S3 N16R8, mains powered, firmware 3.12
Server node	HTTPS server, web page, LEDs, flow log	ESP32-S3 N16R8, firmware 2.7
Microphone	Pipe noise	INMP441 (I2S, default) or MAX9814 (analog), selected by `USE_INMP441`
LEDs and button	Flow, alarm, alarm reset	Flow GPIO 9, alarm GPIO 10, button GPIO 11
Power	5 V supply for each node	Battery operation was dropped in firmware 3.0

What the system does:

- Detects water ON and OFF from the pipe noise, with a calibration done from the web page.
- Shows level, background, thresholds and a 2-minute graph live on a web page.
- Raises a flow alarm when water runs longer than a time you set in minutes (stored in NVS), and an offline alarm when the sensor is silent for 15 s.
- Logs every flow with start time, duration and strength (l, m, h) and offers it as CSV.
- Accepts wireless firmware updates (OTA) on both nodes.

System architecture

The sensor node decides ON or OFF on its own and reports to the server; the server shows the result, raises alarms and keeps the log.



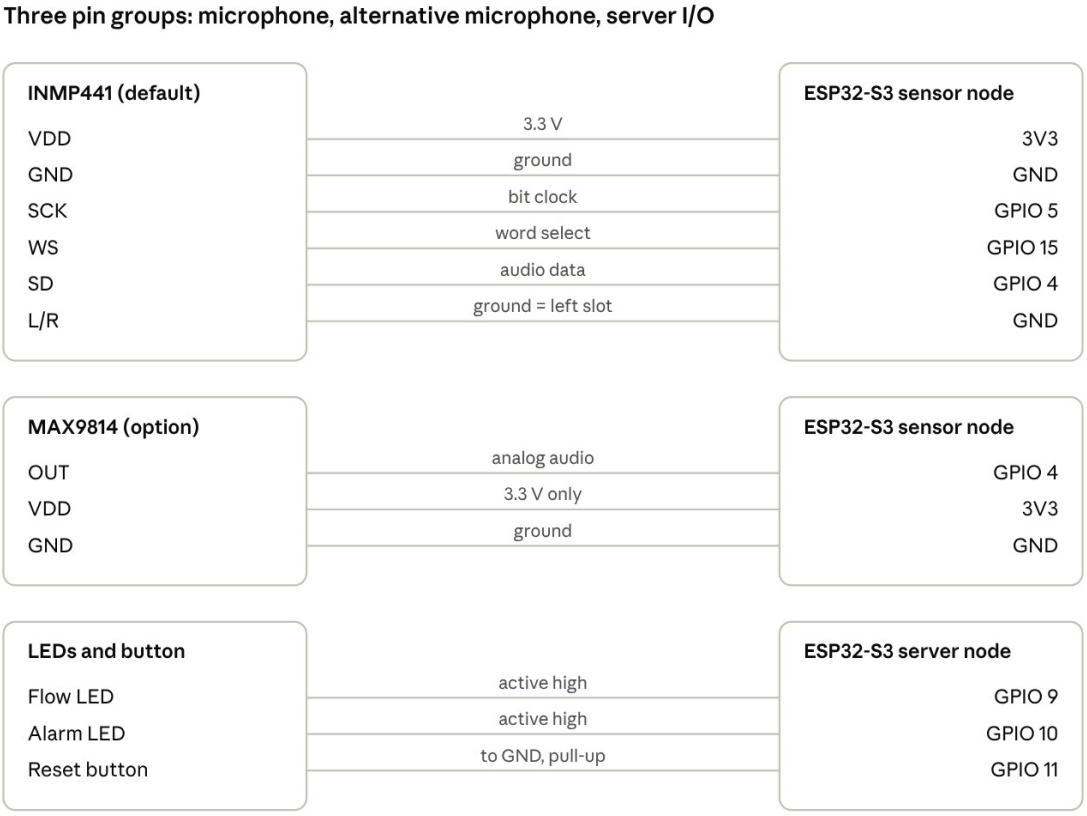
system architecture · 2 nodes, browser, PC

The highlighted link is the only data path between the nodes: the sensor posts its status every 2 s and the server answers with a command (calibrate, save) or new detection levels. The browser and the PC only talk to a node when you use the page or upload firmware.

- **Link security:** HTTPS with the server certificate pinned in the sensor; the sensor authenticates with a bearer token, the browser with a Basic login.
- **Offline alarm:** the server alarms when it hears nothing from the sensor for 15 s.
- **Both nodes keep their own data:** calibration and levels in the sensor's NVS; alarm time and flow log in the server.

Hardware and wiring

The sensor uses three signal pins for the INMP441 (or one analog pin for the MAX9814); the server uses two LED pins and a button.



pin connections · INMP441, MAX9814, LEDs and button

Both microphone variants use GPIO 4, so only one is wired at a time; the choice is the `USE_INMP441` switch in `sensor_node.h` (1 = INMP441, 0 = MAX9814). The stored calibration belongs to one microphone type, so recalibrate after switching.

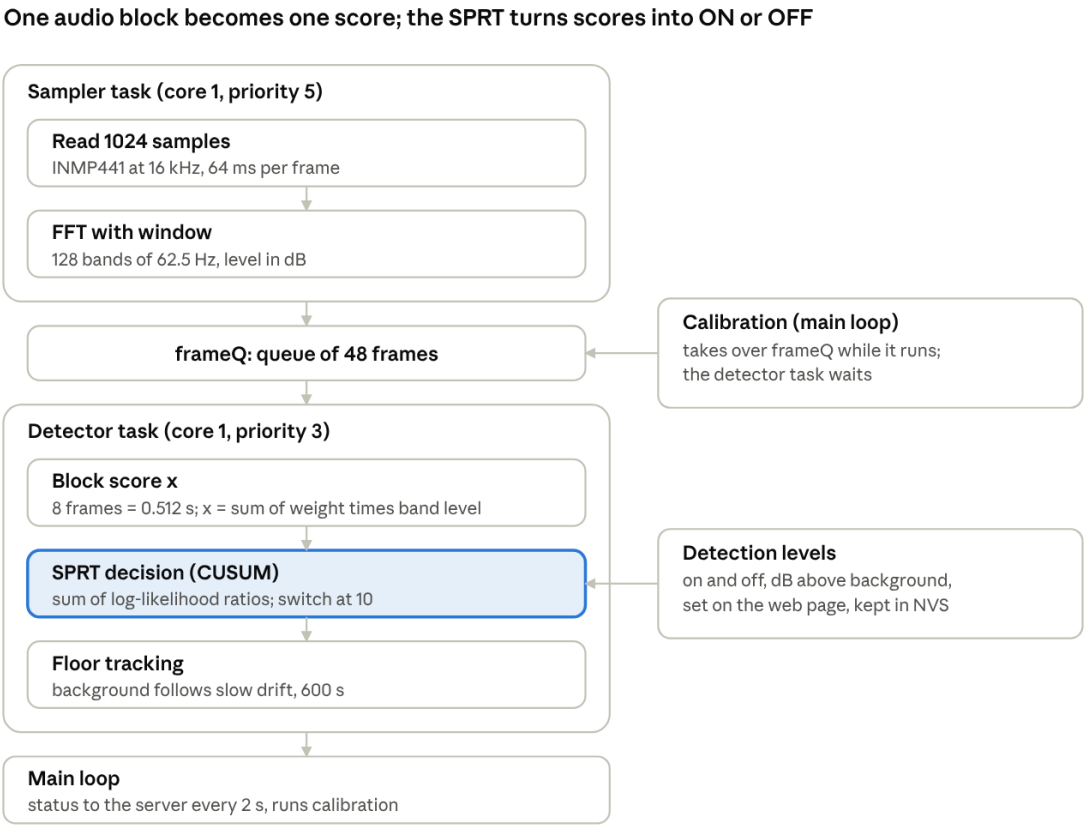
Pins to avoid on the ESP32-S3 N16R8

Pins	Why
GPIO 26 to 37	Used internally by the octal PSRAM and flash (GPIO 35 clamped the microphone signal at about 1.7 V in the first tests)
GPIO 0, 3, 45, 46	Strapping pins
GPIO 19, 20	USB
GPIO 1 to 10	ADC1, the only ADC that works reliably while WiFi runs

Supply: each node runs from a mains-fed 5 V supply. The INMP441 and MAX9814 run from 3.3 V only; a clean supply matters, because a dip resets the ESP32 (reset reason "brownout").

Sensor firmware

The sensor turns pipe noise into one number every 0.512 s and decides ON or OFF from a running statistical test, not from a single threshold crossing.



sensor signal chain · 2 tasks, 1 queue

The sampler fills the queue without waiting for the network; the detector only reads the queue. A slow upload therefore cannot cause lost audio.

Score. For every 0.512 s block the firmware averages the band levels (dB) with weights: x = sum over bands of (weight × level). The weights come from calibration (linear discriminant analysis): bands where ON and OFF differ clearly get a positive weight, noisy bands a small one. Two weight sets are computed and the better one is kept: *shape* (weights sum to zero, independent of loudness) and *level* (absolute loudness).

Decision. Each block adds a log-likelihood ratio to a running sum. The sum is clipped to ±6 per block and the state switches when it reaches 10, so a lone noise spike cannot switch the state but real water switches it within about two to three blocks.

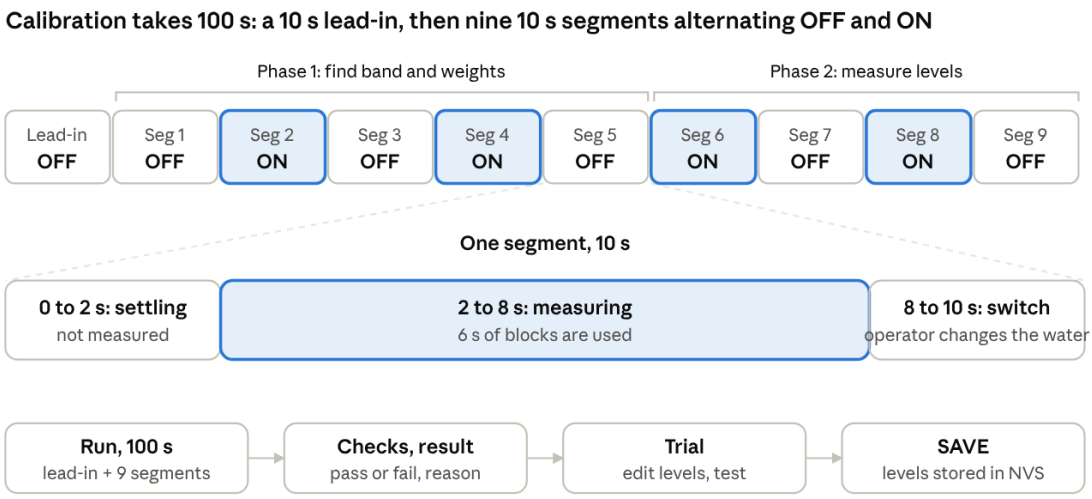
Background. While the state is OFF and the score is close to the floor, the floor follows the score slowly (time constant 600 s), limited to 30 % of the calibrated ON/OFF distance. The page shows "drift limit" when the limit is reached; recalibrate then.

Detection levels. The switch-on and switch-off levels are distances in dB above the floor. Proposed defaults are 4 and 2.5 times the idle noise, never below 1.0 and 0.5 dB. You can change them on the page at any time; they are stored in NVS.

Tasks and memory. Sampler and detector run on core 1, the OTA task on core 0 and the main loop (uploads, calibration) has a 16 KB stack. The 8 MB PSRAM is not used.

Calibration

Calibration teaches the sensor what your pipe sounds like with water off and on; it takes 100 s and you switch the water on cue from the web page.



calibration timeline · 100 s, 9 segments

The web page counts down steadily and tells you when to switch. Each 10 s segment gives you 2 s on each side of the measuring window to switch, so the measurement is never disturbed by the switching itself.

Procedure

1. Press CALIBRATE on the page. The sensor picks the request up with its next status message (within about 2 s).
2. Lead-in, 10 s: keep the water off.
3. Segments 1 to 9: water off in odd segments, on in even segments, switching during the last 2 s of each segment. Open the tap as for normal use.
4. The sensor evaluates the run and shows the result, or the reason for failure.
5. Trial: switch the water yourself and watch the indicator; edit the on and off levels if needed.
6. SAVE stores the values in the sensor's NVS (valid flag cleared, data written, flag set, then read back). CANCEL discards them.

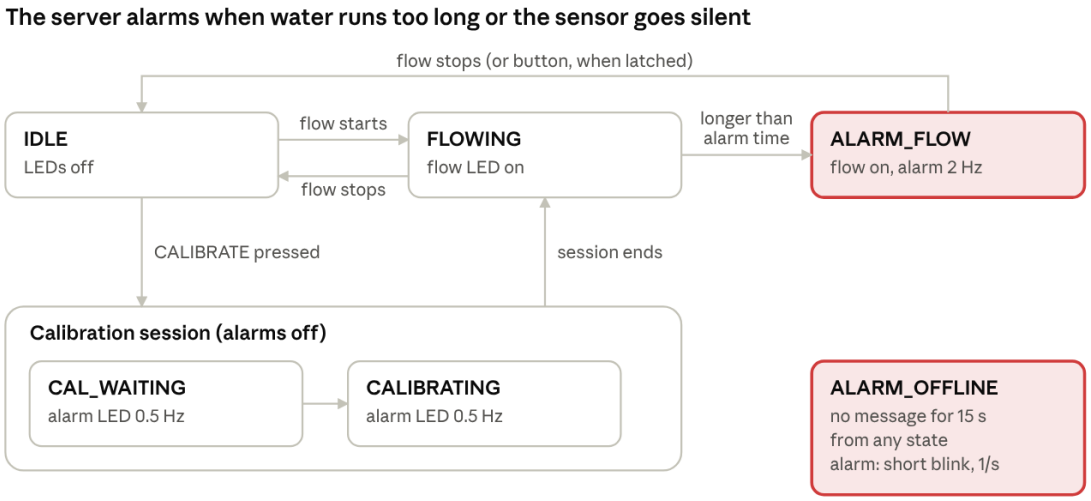
Checks (a failed check keeps the old values)

Check	Limit
Usable data per phase-2 segment	at least 6 blocks
Signal clipping	none beyond a small tolerance
Difference ON minus OFF	at least 3 dB
Separation (difference divided by the rms of the ON and OFF noise)	at least 4
Repeatability	the weaker ON segment is at least half the ON/OFF difference above the stronger OFF segment

The failure message names the numbers, for example the difference in dB, the noise of OFF and ON, and the separation of both scores (shape and level). A session also ends if the sensor sends nothing for 60 s, or after a 10 minute trial without a decision.

Server firmware

The server is an HTTPS server on the ESP32 (esp_https_server) plus a small state machine that drives the two LEDs; it needs no cloud service.



server state machine · 6 states

The flow alarm time is the number of minutes you enter on the page (default 2). With `LATCH_ALARM` set to true the flow alarm stays until the button is pressed; by default it ends when the water stops. Calibration suppresses all alarms because the water is switched on purpose.

Endpoints

Endpoint	Used by	Purpose
POST /api/water	sensor (bearer token)	status and calibration messages; the reply

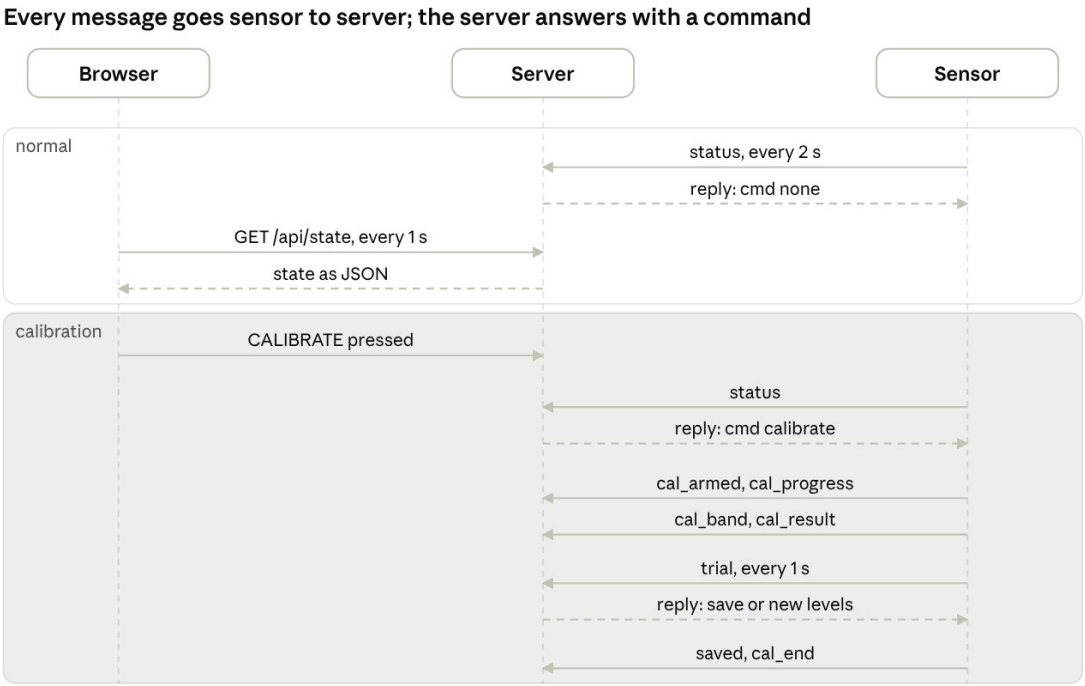
Endpoint	Used by	Purpose
		carries commands
GET /	browser	the web page
GET /api/state	browser	live state as JSON, polled every second
POST /api/cal/start, /save, /cancel	browser	calibration control
POST /api/cal/th	browser	detection levels (on, off in dB above background)
POST /api/config?alarmMin=N	browser	flow alarm time, 1 to 1440 minutes, stored in NVS
GET /api/flows	browser	last 30 flows and today's total
GET /api/flows.csv	browser	whole flow log as CSV

Security

- Browser: HTTP Basic login (`ADMIN_USER`, `ADMIN_PASS`); every POST also needs the header `X-Requested-With: cfg` (protection against forged requests from other sites).
- Sensor: `Authorization: Bearer <DEVICE_TOKEN>`; the sensor verifies the server with the pinned certificate and never accepts an unknown one.
- Five wrong passwords in a row lock the login for 60 s. At most 4 connections at once (each TLS session needs about 40 KB of RAM) and 3 KB per request body.
- A task watchdog (10 s) restarts the server if its main loop hangs. WiFi power saving is off, so replies and OTA handshakes arrive promptly.

Protocol

The sensor starts every exchange: it posts JSON to `/api/water` and the server answers in the same request, so the server never has to open a connection to the sensor.



message sequence · normal operation and calibration

A calibration request from the page waits on the server until the sensor's next status message; the reply then carries `cmd: calibrate` with a one-time `cmd_id`, so a repeated reply does not start a second run. During calibration the sensor sends no status messages, only the calibration messages shown above.

Message types from the sensor

type	when	main fields
status	every 2 s in normal operation	state, raw (level), floor, drift, th_on, th_off, off_on, off_off, delta, rssi, fw, up, reset
cal_armed	calibration starts	autocal flag
cal_progress	every second during the run	stage, t, seg, want (off or on), seg_left, total_left, metric
cal_band	after phase 1	f_low_hz, f_high_hz, d
cal_result	end of the run	ok, reason, mean_off, mean_on, sd_off, sd_on, separation_sigma, score, sigma_shape, sigma_level, segments
trial	every second while testing	metric, state, off_on, off_off, delta
saved	after SAVE	ok, reason
cal_end	session over	outcome (saved, cancelled, lost, timeout, failed),

type	when	main fields
		reason

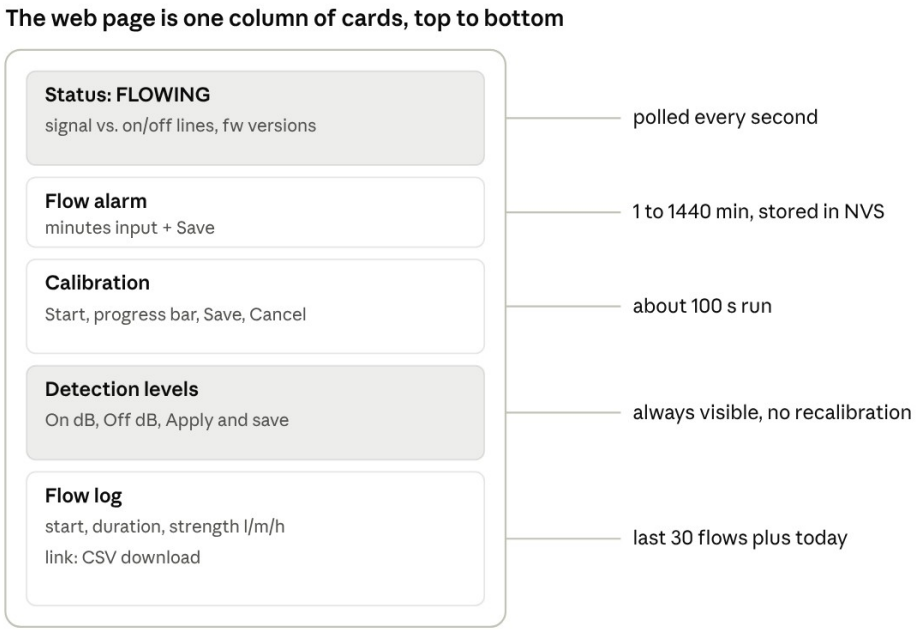
Reply from the server

`{"ok": true, "cmd": "none", "cmd_id": ""}`
`cmd` is one of `none`, `calibrate`, `save` or `cancel`. A reply to a status or trial message can also carry `th_on` and `th_off` (dB above the background); the server repeats them until the sensor reports the new values back.

Detecting restarts. Every status message carries the sensor's uptime in seconds and its last reset reason (for example power-on, crash, brownout). If the uptime drops during a calibration, the page says the sensor restarted and why.

Web page

The page is served over HTTPS and needs the browser login. It polls `/api/state` once a second.



web page layout sketch

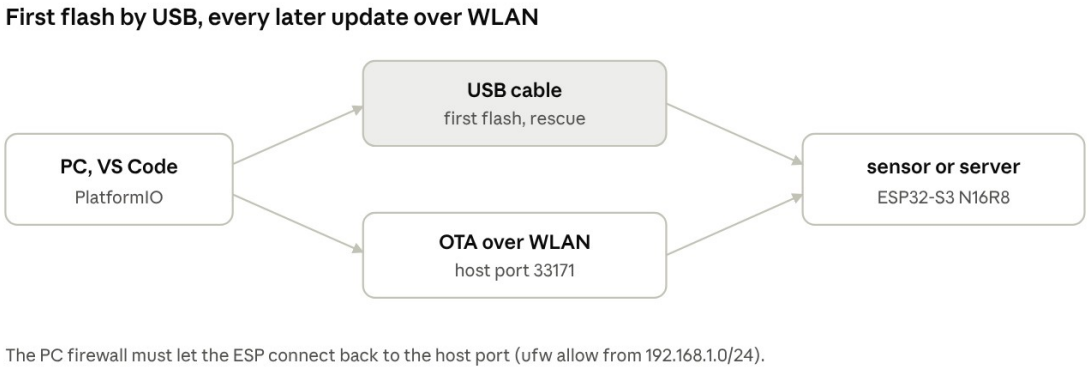
card	what you can do
Status	see IDLE, FLOWING or an alarm, the live signal against the on and off lines, sensor and server firmware versions, sensor uptime and reset reason
Flow alarm	set after how many minutes of continuous flow the alarm fires (1 to 1440), saved in NVS

card	what you can do
Calibration	start a run, watch the progress, then save or cancel the trial
Detection levels	change the on and off distances in dB above the background; outside a calibration they are stored in the sensor at once, during a trial they apply live and SAVE stores them
Flow log	read the last 30 flows and today's total; download the CSV (start,duration_s,strength)

Strength column. `l` means the flow stayed below 30 % of the calibrated step between OFF and ON, `m` below 70 %, `h` 70 % or more. In LibreOffice with German settings, the sum of all `h` rows times 0.1 is `=ZÄHLENWENN(C:C;"h")*0,1`.

Build, flashing and OTA

One PlatformIO project holds both nodes. `src/main.cpp` includes `sensor_node.h` or `server_node.h` depending on the environment, so `pio run -e sensor` and `pio run -e server` build the two firmwares. VS Code's upload button compiles first when needed; no separate compile step is required.



flashing workflow

Build settings that matter

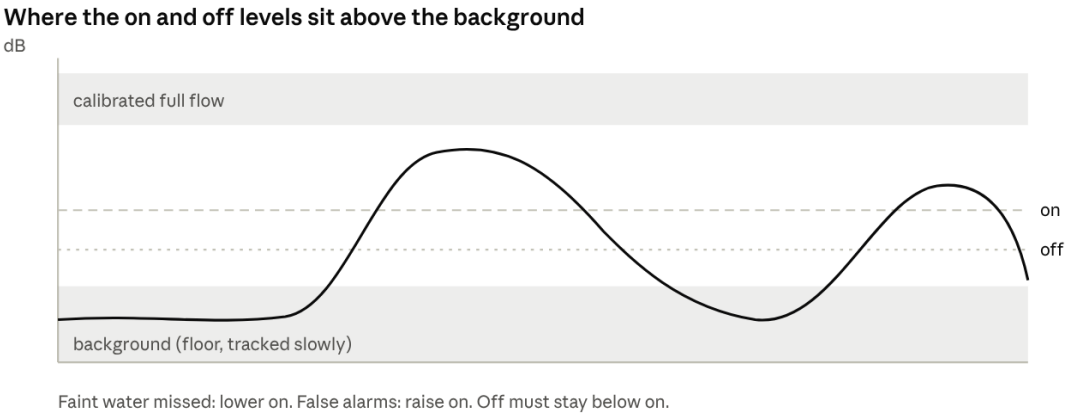
- `lib_ldf_mode = chain+`, and `FS` and `LittleFS` in `lib_deps` plus the include path `-I${platformio.packages_dir}/framework-arduinoespressif32/libraries/FS/src`, otherwise `FS.h` is not found.
- Large main-loop stack through `SET_LOOP_TASK_STACK_SIZE` (16 KB); sampler 8 KB, detector 6 KB.
- `Network.h` is included before `WiFi.h`.

OTA

- Hostnames `water-sensor` and `water-server`; the password defaults to the device token (sensor) or the admin password (server) unless `OTA_PASSWORD` is defined.

- Upload flags need a fixed host port: `--host_port=33171`, and the PC firewall must allow it from the local network.
- The sensor handles OTA in its own task on core 0, because its main loop is blocked for up to 1.7 s by HTTPS posts.
- The server writes the flow log to flash before an update starts.
- The first flash of a new firmware generation (for example sensor 3.10) still needs USB.

Tuning and troubleshooting



detection levels relative to the background

The detector switches on when the signal stays above the on line for long enough (SPRT/CUSUM) and off again below the off line. Both lines are editable on the page without a new calibration.

symptom	likely cause	fix
faint water not detected	on level too high	lower "on" in Detection levels
false alarms	on level too low, or background changed	raise "on"; recalibrate if the pipe noise changed
calibration: "repeated segments disagree"	someone used water, or valve switching too slow	keep water off for the OFF segments, switch within the 2 s window
calibration: separation too low	microphone too far from the pipe, or weak flow	move the sensor to the pipe, raise flow
calibration stops, sensor restarts	stack or power problem	the page shows the reset reason; check the supply
no signal at all (INMP441)	wiring or defective microphone	check L/R pin to GND and the I2S pins; try another module
OTA: no response	PC firewall	allow the host port 33171 from the local network

symptom	likely cause	fix
OTA: authentication failed	password differs from `upload_flags`	use the device token (sensor) or admin password (server)
server upload needs retries	WLAN sleep or weak signal	WLAN sleep is off in server 2.7; check RSSI
sensor shown offline	no message for 15 s	check WLAN and server address

Secrets and backup

Four files hold private values and exist only on your PC. Back them up somewhere private, and never commit or post them.

file	contains	if lost
`sensor_secrets.h`	WLAN name and password, server address, device token	rewrite by hand; the token must match the server's
`server_secrets.h`	WLAN credentials, admin login, device token	rewrite by hand; choose new values
`cert.h`	TLS certificate and private key	generate a new self-signed pair; the browser warns once again
`platformio.ini`	OTA upload passwords	rewrite; the passwords equal the device token and admin password

Nothing is lost on the devices themselves: calibration and alarm minutes live in NVS, the flow log in LittleFS, and both survive OTA updates.

Security notes

- Choose strong values for the admin password and the device token. Short words used while testing are easy to guess and were shown in chat, so replace them.
- The browser uses Basic auth over HTTPS; the sensor uses a Bearer token. Five failed logins lock the login for 60 s.
- Changing settings needs the header `X-Requested-With: cfg`, which blocks cross-site requests.

Version history and ideas

version	change
sensor 3.2	INMP441 microphone path (mono left slot), the reference state
sensor 3.3 to 3.9	calibration checks (rms spread, repeatability), larger task stacks, reset reason and uptime in status
sensor 3.10	OTA in its own task

version	change
sensor 3.11	detector uses the editable on and off levels
sensor 3.12	levels applied and stored from the page without recalibration
server 2.x	flow log with CSV, alarm minutes in NVS, firmware versions on the page, OTA
server 2.5 to 2.7	detection levels card, always visible; WLAN sleep off

Ideas not built yet

- A camera node (ESP32-S3 with camera) that reads the rotating pointer of the water meter and reports full turns or the angle.
- A 900 MHz radio link (E49-900M20S) instead of WLAN, for places without coverage.
- A port to the ESP32-C5.
- A battery variant of the sensor (old versions sensor 2.4 and server 1.7 are kept as backups).