esphome:

  name: "saltowl"

  friendly\_name: Salt Owl

esp8266:

  board: esp01\_1m

# Enable logging

logger:

# Enable Home Assistant API

api:

  encryption:

    key: "1hRF7mhHjtpM+rYixJKrhy9oE9fgeexbqZdtyb1UHIs="

ota:

  - platform: esphome

    password: "a95dafab2d8c84051f2cdcbebd0c28cf"

wifi:

  ssid: !secret wifi\_ssid

  password: !secret wifi\_password

  manual\_ip:

   static\_ip: 192.168.11.166

   gateway: 192.168.11.1

   subnet: 255.255.255.0

  # Enable fallback hotspot (captive portal) in case wifi connection fails

  ap:

    ssid: "Saltyowl-2 Fallback Hotspot"

    password: "EPVMmq21Ipzm"

captive\_portal:

# --- Sensors: raw ultrasonic distance readings ---

sensor:

  - platform: ultrasonic

    trigger\_pin: GPIO12

    echo\_pin: GPIO14

    name: "Left Chamber"

    id: left\_distance

    update\_interval: 10s

    unit\_of\_measurement: "cm"

    filters:

     - multiply: 100.0   # convert m → cm

  - platform: ultrasonic

    trigger\_pin: GPIO1

    echo\_pin: GPIO3

    name: "Right Chamber"

    id: right\_distance

    update\_interval: 10s

    unit\_of\_measurement: "cm"

    filters:

     - multiply: 100.0   # convert m → cm

  # --- Computed % full for Left chamber ---

  - platform: template

    name: "Left Chamber % Full"

    id: left\_percent

    unit\_of\_measurement: "%"

    accuracy\_decimals: 0

    update\_interval: 2s

    lambda: |-

      if (!id(left\_distance).has\_state()) return NAN;

      float dist = id(left\_distance).state;

      float full = id(left\_full).state;

      float empty = id(left\_empty).state;

      // Clamp

      dist = clamp(dist, full, empty);

      return 100.0 \* (empty - dist) / (empty - full);

  # --- Computed % full for Right chamber ---

  - platform: template

    name: "Right Chamber % Full"

    id: right\_percent

    unit\_of\_measurement: "%"

    accuracy\_decimals: 0

    update\_interval: 2s

    lambda: |-

      if (!id(right\_distance).has\_state()) return NAN;

      float dist = id(right\_distance).state;

      float full = id(right\_full).state;

      float empty = id(right\_empty).state;

      // Clamp

      dist = clamp(dist, full, empty);

      return 100.0 \* (empty - dist) / (empty - full);

# --- Configurable calibration values (via HA UI) ---

number:

  - platform: template

    name: "SaltOwl Left Full Distance"

    id: left\_full

    optimistic: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 10.0

  - platform: template

    name: "SaltOwl Left Empty Distance"

    id: left\_empty

    optimistic: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 30.0

  - platform: template

    name: "SaltOwl Right Full Distance"

    id: right\_full

    optimistic: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 10.0

  - platform: template

    name: "SaltOwl Right Empty Distance"

    id: right\_empty

    optimistic: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 30.0

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

V2 code

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Taken out

  # --- % Full Left Chamber (bottom-up logic) ---

  - platform: template

    name: "Left Chamber % Full"

    id: left\_percent

    internal: true

    unit\_of\_measurement: "%"

    accuracy\_decimals: 0

    update\_interval: 2s

    lambda: |-

      if (!id(left\_distance).has\_state()) return NAN;

      float dist = id(left\_distance).state;

      float fill\_top = id(left\_fill\_top).state;

      float max\_height = id(left\_max\_height).state;

      float bottom = fill\_top + max\_height;

      dist = clamp(dist, fill\_top, bottom);

      return 100.0 \* (bottom - dist) / max\_height;

  # --- % Full Right Chamber (bottom-up logic) ---

  - platform: template

    name: "Right Chamber % Full"

    id: right\_percent

    internal: true

    unit\_of\_measurement: "%"

    accuracy\_decimals: 0

    update\_interval: 2s

    lambda: |-

      if (!id(right\_distance).has\_state()) return NAN;

      float dist = id(right\_distance).state;

      float fill\_top = id(right\_fill\_top).state;

      float max\_height = id(right\_max\_height).state;

      float bottom = fill\_top + max\_height;

      dist = clamp(dist, fill\_top, bottom);

      return 100.0 \* (bottom - dist) / max\_height;

number:

  # --- Left Chamber Calibration ---

  - platform: template

    name: "SaltOwl Left Fill Top (cm from sensor)"

    id: left\_fill\_top

    optimistic: true

    internal: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 10.0

  - platform: template

    name: "SaltOwl Left Max Height (cm)"

    id: left\_max\_height

    optimistic: true

    internal: true

    min\_value: 1

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 20.0

  # --- Right Chamber Calibration ---

  - platform: template

    name: "SaltOwl Right Fill Top (cm from sensor)"

    id: right\_fill\_top

    optimistic: true

    internal: true

    min\_value: 0

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 10.0

  - platform: template

    name: "SaltOwl Right Max Height (cm)"

    id: right\_max\_height

    optimistic: true

    internal: true

    min\_value: 1

    max\_value: 100

    step: 0.5

    unit\_of\_measurement: "cm"

    initial\_value: 20.0

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

🎉 Awesome! Glad to hear it's compiling cleanly now — that’s a solid milestone. You're now running a nicely structured ESPHome setup with:

* 🧠 Calibration variables (offset, depth, threshold) stored and synced with HA
* 📡 Ultrasonic distance measurements
* 📈 Fill % computed immediately after sensor update
* 🚨 Alerts for low fill in each chamber and average
* ✅ Resilience across reboots thanks to restore\_value and on\_boot

If you’re at the **testing stage**, here are a few optional next steps you might consider:

**🧪 Dev/Test Ideas:**

* Temporarily reduce update\_interval to something like 10s while testing
* Add log\_level: DEBUG to your logger: block to see real-time values
* Use a text\_sensor: to display the **average fill** in one spot
* Add status\_led if your board supports it for visual state

**🛡️ Stability Tips:**

* Monitor how quickly HA resyncs values after a reboot
* Validate alert triggers by simulating low fill levels
* Keep an eye on the logs for noise or unexpected jumps in sensor readings

hmm, log spam....tell me more

**ChatGPT said:**

Ah yes — **log spam** in ESPHome can become a *real pain*, especially when:

* You're polling sensors frequently (even at 5s or 10s)
* You have template or copy sensors that report the same value repeatedly
* You’re trying to monitor the logs for *real issues*, but they get drowned out by constant updates

**🧾 Typical Symptoms of Log Spam**

You’ll see stuff like this every few seconds:

[11:02:05][D][sensor:127]: 'Left Chamber Sensor Distance': Sending state 63.45000 cm

[11:02:05][D][sensor:127]: 'Right Chamber Sensor Distance': Sending state 63.40000 cm

[11:02:05][D][sensor:127]: 'Left Chamber Fill (%)': Sending state 58.0 %

[11:02:05][D][sensor:127]: 'Right Chamber Fill (%)': Sending state 57.0 %

[11:02:05][D][binary\_sensor:036]: 'Left Chamber Fill Low': Sending state OFF

[11:02:05][D][binary\_sensor:036]: 'Right Chamber Fill Low': Sending state OFF

If values don’t change much — or at all — these logs just fill up your console, serial monitor, or OTA log view.

**✅ Strategies to Reduce Log Spam**

**1. Use filters: like delta or heartbeat**

In sensor: components, you can add filters like:

filters:

- delta: 1.0 # Only report if change > 1.0

- heartbeat: 60s # Always report at least every 60s

🔹 Example:

- platform: ultrasonic

trigger\_pin: GPIO12

echo\_pin: GPIO14

name: "Left Chamber Sensor Distance"

id: left\_distance

update\_interval: 10s

unit\_of\_measurement: "cm"

filters:

- multiply: 100.0

- delta: 1.0

- heartbeat: 60s

This way:

* You only log when something *actually changes*
* You still get occasional updates (every 60s) even if nothing changes

**2. Use internal: true for sensors you don't need in HA**

You're already doing this for left\_percent and right\_percent — nice!  
This means those internal template sensors **won’t log or publish** unless you explicitly expose them.

✅ Avoid exposing noisy sensors unless absolutely needed in HA.

**3. Disable logs from specific components**

You can’t turn off logs per sensor yet, but you can control the **log level globally**:

logger:

level: WARN

Valid levels:

* NONE
* ERROR
* WARN
* INFO (default)
* DEBUG
* VERBOSE
* VERY\_VERBOSE

So if you're not debugging, switching to WARN will cut out most routine sensor logs.

**4. Avoid update\_interval: on template: sensors if you're manually updating**

Since you’re already triggering % full updates via on\_value, make sure you **don’t have update\_interval: on the template sensor**. Otherwise, it still runs periodically and spams logs.

✅ Your current setup already does this — good!

**5. Use force\_update: false to suppress duplicate logs**

If a sensor reports the **same value** as last time, ESPHome won't log it unless:

* force\_update: true is set

So make sure you're not forcing updates unnecessarily:

sensor:

- platform: template

...

force\_update: false # <- default, can be omitted

**🔇 TL;DR — Reduce Log Spam by:**

| **What** | **Why** |
| --- | --- |
| filters: delta and heartbeat | Only log real changes |
| internal: true | Silence non-HA-facing sensors |
| logger.level: WARN | Reduce overall log verbosity |
| Avoid update\_interval: on manually updated sensors | Prevent double-processing |
| force\_update: false | Avoid duplicate state logs |

If you'd like, I can walk you through tailoring this to your current setup — or give you a “quiet mode” config template. Let me know how minimal you want the logs.