

Optimierungsanalyse Wetterstation (ESP32-S3)

Diese Seite fasst die Analyse der ESPHome-Konfiguration zusammen. Die Maßnahmen sind unterteilt in kritische Korrekturen, Code-Optimierungen und Erweiterungen (Software & Hardware).

1. Kritische Korrekturen & Bereinigungen

Die folgenden Punkte sollten priorisiert umgesetzt werden, um Fehler in den Daten zu vermeiden.

- **Batterie-Status (SoC) korrigieren:**
 - **Problem:** Der aktuelle Sensor ```{SmartSolar}` Batterie Prozent`` nutzt eine Spannungstabelle. Bei LiFePO4-Akkus (JBD BMS) ist die Spannungskurve zu flach für präzise Werte.
 - **Lösung:** Den Template-Sensor entfernen. Stattdessen den SoC direkt aus dem BMS auslesen (```bms0_capacity_remaining``` / ```nominal```).
- **Regenmesser Logik:**
 - **Beobachtung:** ```pulse_counter``` ist für Wippen-Regenmesser oft ungenau (Prellen, sehr niedrige Frequenz).
 - **Empfehlung:** Prüfen, ob bei Nieselregen Pulse verloren gehen. Falls ja, Umbau auf ```binary_sensor``` mit Filter ```delayed_on``` und einem ```counter```.
- **Taupunkt (Dew Point) reaktivieren:**
 - **Status:** Der Code ist aktuell auskommentiert.
 - **Maßnahme:** Einkommentieren! Der Taupunkt ist essenziell für Nebelvorhersage, Wolkenuntergrenze und Schwüle-Empfinden.
- **Datenflut JBD BMS:**
 - **Problem:** Das BMS sendet Dutzende Sensoren (Einzelzell-Spannungen, Bitmasken), die die Home Assistant Datenbank unnötig füllen.
 - **Lösung:** Bei reinen Debug-Werten ```internal: true``` setzen oder ```entity_category: diagnostic``` verwenden.

2. Code & Konfigurations-Optimierung

Maßnahmen für Stabilität und Energieeffizienz.

- **WiFi Power Save:**
 - Aktuell: ```power_save_mode: none```.
 - **Empfehlung:** Im Winter (bei geringem Solarertrag) auf ```light``` ändern, um ca. 50-100mA zu sparen.
- **Filter-Nutzung:**
 - Die Nutzung von ```delta``` und ```heartbeat``` ist bereits exzellent umgesetzt. Dies bei neuen Sensoren beibehalten.
- **INA3221 Integration:**
 - Der Shunt auf Kanal 3 misst den Verbrauch. Diesem Sensor muss eine feste ID gegeben werden (z.B. ```id: verbraucher_power```), um damit die Energiebilanz zu rechnen.

3. Neue Berechnungen (Software-Sensoren)

Diese Werte können **ohne neue Hardware** rein rechnerisch ermittelt werden.

Meteorologie

- **Absolute Luftfeuchtigkeit (g/m³):** Zeigt die tatsächliche Wassermenge in der Luft (wichtig für Lüftungsempfehlungen).
- **Wolkenuntergrenze (Cloud Base):** Berechnet die theoretische Höhe von Cumulus-Wolken ($((Temp - Taupunkt) * 125 + Höhe)$).
- **Frostgrenze im Boden:** Interpolation zwischen dem -5cm und -30cm Sensor, um zu errechnen, wie tief der Frost in den Boden eingedrungen ist.
- **Luftdichte (kg/m³):** Berechnet aus Druck, Temperatur und Gaskonstante.

Energie & Solar

- **Energiebilanz Heute (Wh):** $((Solar\ Ertrag\ (Yield) - Verbrauch\ (Integration))$. Zeigt sofort, ob der Akku geladen oder entladen wird.
- **Sonnenschein-Status:** Ein `binary_sensor`, der "ON" geht, wenn $Panel\ Spannung > Batterie\ Spannung + 5V$ ist (Workaround für fehlenden Lichtsensor).
- **Autonomiezeit (Tage):** $(Restkapazität\ (Ah) / Durchschnittsstrom\ (A) / 24)$. Zeigt an, wie lange die Station ohne Sonne überlebt.
- **Kabel-Diagnose:** Differenz zwischen `Victron Batterie Spannung` und `BMS Gesamtspannung`. Hohe Abweichungen deuten auf Korrosion hin.

4. Hardware Fehlteile (Empfehlung zur Nachrüstung)

Um aus der Station eine "vollständige" Wetterstation zu machen, fehlen folgende Sensoren:

Priorität	Sensor-Typ	Hardware Empfehlung	Nutzen
Hoch	Helligkeit / Licht	BH1750 (Lux) oder VEML6075 (UV)	Unterscheidung Sonnig/Bewölkt, UV-Warnung, unabhängige Sonnenscheindauer
Mittel	Bodenfeuchte	Kapazitiver Sensor (v1.2)	Gießempfehlung. Die Bodentempsensoren sind vorhanden, Feuchte fehlt.
Mittel	Feinstaub	SDS011 oder SPS30	Luftqualität (Winter: Kaminrauch, Frühling: Pollen-Indikator)
Niedrig	Blitzsensor	AS3935	Erkennung von Gewitterfronten (Distanzschätzung)
Wartung	Gehäuse-Feuchte	SHT3x / BME280	Überwachung des Elektronik-Gehäuses auf Kondenswasser (Silica-Gel Check)

5. YAML Code-Schnipsel (Copy & Paste)

Hier die wichtigsten Ergänzungen für die ``sensor:`` bzw. ``binary_sensor:`` Sektion.

```
# --- BERECHNETE WETTERWERTE ---

# Absolute Luftfeuchtigkeit
- platform: template
  name: "${WetterCalc} Absolute Luftfeuchtigkeit"
  unit_of_measurement: "g/m³"
  lambda: |-
    float T = id(aht_temp).state;
    float rh = id(aht_hum).state;
    if (std::isnan(T) || std::isnan(rh)) return NAN;
    float e = 6.112 * exp((17.67 * T) / (T + 243.5));
    return (e * rh * 2.1674) / (273.15 + T);

# Wolkenuntergrenze (Benötigt Taupunkt ID!)
- platform: template
  name: "${WetterCalc} Wolkenuntergrenze (NN)"
  unit_of_measurement: "m"
  lambda: |-
    float T = id(aht_temp).state;
    float td = id(taupunkt_sensor).state; // ID vom Taupunkt prüfen!
    if (std::isnan(T) || std::isnan(td)) return NAN;
    return ((T - td) * 125.0) + ${hoehe_in_m};

# Frostgrenze Tiefe (Interpolation Boden)
- platform: template
  name: "${WetterCalc} Frostgrenze Tiefe"
  unit_of_measurement: "cm"
  lambda: |-
    float t5 = id(dallas_temp_addr_2).state; // -5cm
    float t30 = id(dallas_temp_addr_3).state; // -30cm
    if (t5 > 0) return 0.0;
    if (t30 < 0) return 30.0;
    return 5.0 + ((0.0 - t5) * (30.0 - 5.0) / (t30 - t5));

# --- BERECHNETE ENERGIEWERTE ---

# Sonnenschein Erkennung (Via Solarpanel)
- platform: template
  name: "${WetterCalc} Sonnenschein Status"
  id: sun_is_shining
  lambda: |-
    // Wenn Panelspannung deutlich über Akkuspannung liegt
    if (id(pv).state > (id(bv).state + 4.0)) return true;
    if (id(panel_power).state > 15.0) return true;
    return false;
```

Spannungsabfall Kabel (Diagnose)

```
- platform: template
  name: "${WetterCalc} Spannungsverlust Kabel"
  unit_of_measurement: "V"
  lambda: |-
    return abs(id(bv).state - id(bms0_total_voltage).state);
```

From:
<https://www.drklipper.de/> - Dr. Klipper Wiki

Permanent link:
https://www.drklipper.de/doku.php?id=haussteuerung:esphome:ws_addons&rev=1765888432

Last update: 2025/12/16 13:33

