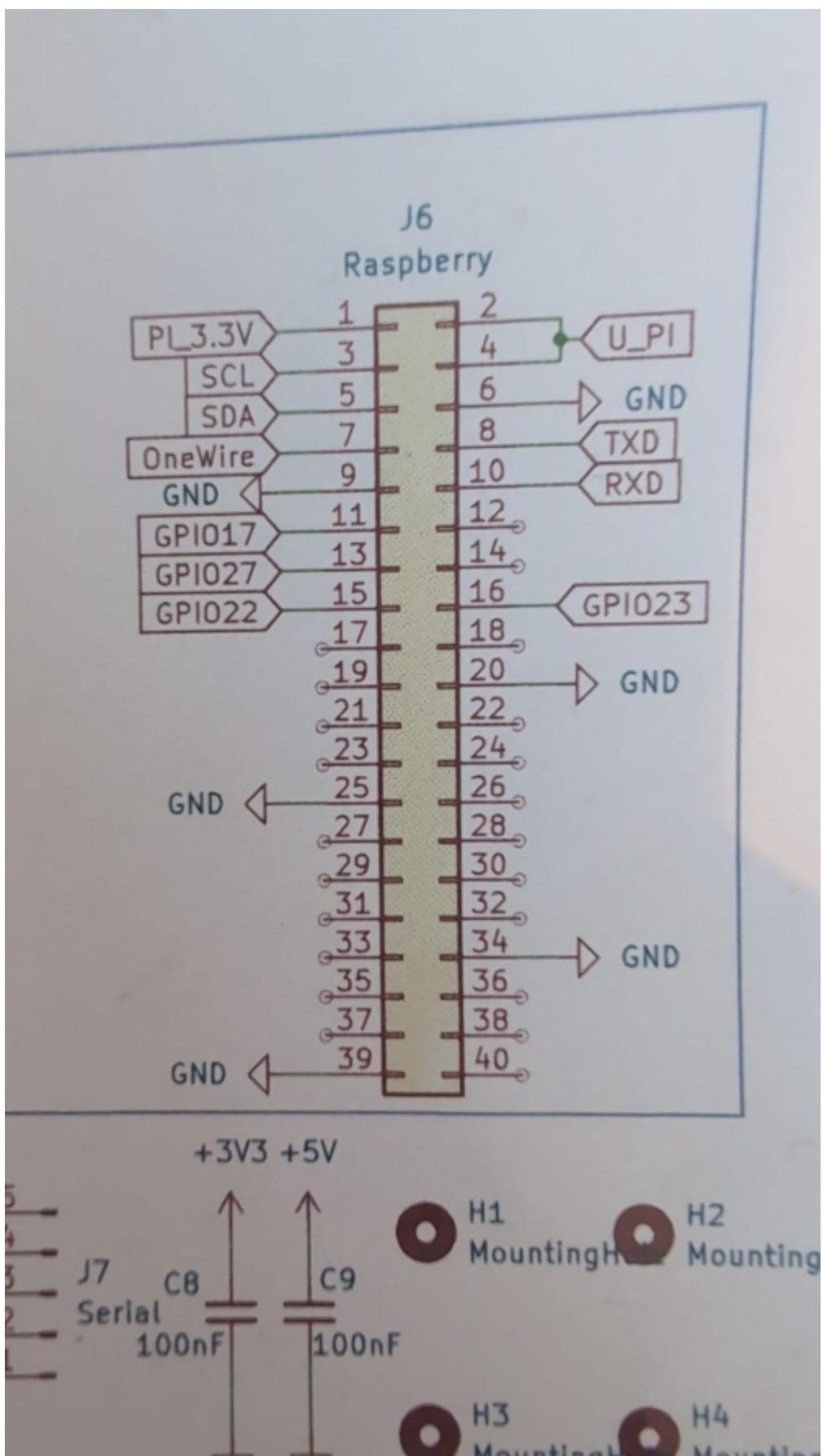


Wetterstation

KONVERTIERUNG Pi > ESP32

- Anschluss Pi



- Anschluss ESP32

Pi Pin	ESP32 Pin	Funktion
GPIO 5	GPIO21	I2C SDA

Pi Pin	ESP32 Pin	Funktion
GPIO 3	GPIO22	I2C SCL
GPIO 7	GPIO17	OneWire
GPIO21	GPIO25	Regenmesser
GPIO20	GPIO27	Windgeschwindigkeit
GPIO17	GPIO26	LED Wlan
GPIO27	GPIO18	LED Akku
GPIO22	GPIO19	LED Regen
GPIO23	GPIO23	LED Reserve
5V	Vin ??	5V Versorgung
3,3V		3,3V Kommen vom Pi !
6, 20, 25, 34, 39		GND

- Dallas Adressen

Sensor Nr	Dallas MAC	Dallas MAC (Pi)	Funktion
1	0xae01184286d5ff28	28-01184286d5ff	Bodentemperatur +5cm (Luft)
2	0x270318408bf1ff28	28-0318408bf1ff	Bodentemperatur -5cm (Oberflaeche)
3	0xd30118428444ff28	28-0118428444ff	Bodentemperatur -30cm
4	0x370118428919ff28	28-0118428919ff	Bodentemperatur -100cm

Extra Berechnungen U/I

Bereich/Sensor	Gemessene Werte	Abgeleiteter Wert	Formel	Beschreibung		
µController (5V)	Spannung (V_µC) Strom (I_µC)	Leistung (P_µC)	$P_{\mu C} = V_{\mu C} * I_{\mu C}$	Zeigt den aktuellen Leistungsverbrauch des Mikrocontrollers und aller angeschlossenen Peripheriegeräte in Watt; hilfreich zur Überwachung des Energiebedarfs und zur Erkennung von Anomalien wie Überlastungen.		
		Energie verbraucht (Wh_µC)	$Wh_{\mu C} = \int P_{\mu C} dt$ (Integration über Zeit)	Kumulierte Energie, die der µController und seine Komponenten über einen bestimmten Zeitraum verbraucht haben; ideal für langfristige Analysen, z. B. tägliche oder monatliche Verbrauchsbilanzen und Optimierungen.		
Akku (12V)	Spannung (V_Akku) Strom (I_Akku) (positiv: Entladung; negativ: Ladung)	Leistung (P_Akku)	$P_{Akku} = V_{Akku} * I_{Akku}$	Gibt die aktuelle Leistung an, die der Akku abgibt (bei Entladung) oder aufnimmt (bei Ladung) in Watt; ermöglicht die Echtzeit-Überwachung des Akku-Zustands und der Lade-/Entladeprozesse.		

		SOC (State of Charge) in % (genau)	$SOC = [(Kapazität_Ah - \int I_Akku dt * Effizienz) / Kapazität_Ah] * 100$ (Effizienz ~0.95; Initial aus V-LUT)	Berechnet den genauen Füllstand des Akkus in Prozent unter Berücksichtigung von Coulomb-Zählung und Lade-/Entladeeffizienz; verbessert die Genauigkeit im Vergleich zu reiner Spannungsmessung und hilft bei der Vorhersage der Restkapazität.		
		Restlaufzeit (h)	$Restlaufzeit = (SOC/100 * Kapazität_Ah) / I_{\mu C_eq}$ (nur bei Entladung)	Schätzt die verbleibende Betriebszeit in Stunden basierend auf dem aktuellen Füllstand und dem angepassten Verbrauchsstrom; nützlich für Alarne bei niedrigem Ladestand und Planung von Ladezyklen (nur relevant bei Entladung).		
		Energie entnommen/geladen (Wh_Akku)	$Wh_Akku = \int P_Akku dt$	Kumulierte Energiebilanz des Akkus, die entnommen oder geladen wurde; ermöglicht die Analyse von Zyklen, Degradation und Gesamteffizienz über längere Perioden.		
Solarpanel	Spannung (V_Solar) Strom (I_Solar)	Leistung (P_Solar)	$P_Solar = V_Solar * I_Solar$	Zeigt den aktuellen Energieertrag des Solarpanels in Watt; hilft bei der Bewertung der Sonneneinstrahlung und der Panel-Leistung in Echtzeit.		
		Energie erzeugt (Wh_Solar)	$Wh_Solar = \int P_Solar dt$	Kumulierter Energieertrag des Solarpanels über Zeit; eignet sich für Statistiken wie täglichen Ertrag, Saisonvergleiche und Systemoptimierung.		
Systemweit (kombiniert)	-	Effizienz Laderegler (Solar → Akku)	$Eff_Laden = (\frac{P_Akku}{P_Solar}) * 100$ (nur bei Ladung, I_Akku > 0)			Misst den Wirkungsgrad des Ladereglers, d. h. welcher Anteil der Solarleistung effektiv im Akku gespeichert wird; niedrige Werte können auf Verluste durch Wärme, falsche MPPT-Einstellungen oder Defekte hinweisen.
		Effizienz DC-DC-Wandler (Akku → µC)	$Eff_Wandler = (P_{\mu C} / P_Akku) * 100$ (nur bei Entladung, I_Akku < 0)	Berechnet den Wirkungsgrad des Spannungswandlers von 12V auf 5V; zeigt Verluste und hilft bei der Diagnose von Ineffizienzen oder Hardwareproblemen.		

		Gesamteffizienz (Solar → µC)	Eff_Gesamt = (P_µC / P_Solar) * 100 (bei direkter Solarversorgung)	Gibt den Gesamtwirkungsgrad des Systems von Solarerzeugung bis zum Verbrauch am µController an; nützlich für die Bewertung der Systemeffizienz und Identifikation von Optimierungspotenzialen.		
		Autarkie-Grad (%)	Autarkie = [min(P_Solar, P_µC) / P_µC] * 100	Prozentsatz, zu dem der µController-Verbrauch direkt durch Solarenergie gedeckt wird, ohne den Akku zu belasten; fördert die Analyse der Systemunabhängigkeit von externen Quellen.		

Sensoren

Sensor	Typ	Adresse	Notes	IO / NIO
AHT20	I2C	0x38	Variante AHT20 angeben!	IO
SHT 3x	I2C	0x44	Kein SHT2x wie bei Thomas	IO
INA 3221	I2C	0x40		
ADS1115	I2C	0x48		
BME280	I2C	0x76	BME und nicht BMP !	IO

AHT20

```
# AHT10/AHT20 Sensor
- platform: aht10
  variant: AHT20
  i2c_id: wetter_i2c_bus
  address: 0x38
  update_interval: 60s
  temperature:
    name: "AHT20 Aussentemperatur"
    icon: 'mdi:thermometer'
    id: aht_temp
  humidity:
    name: "AHT20 Aussenluftfeuchtigkeit"
    icon: 'mdi:water-percent'
    id: aht_hum
```

BME280

```
# --- ATMOSPHÄRISCHE SENSOREN ---
- platform: bme280_i2c
  i2c_id: wetter_i2c_bus
  address: 0x76
  update_interval: 60s
  temperature:
    name: "BME280 Temperatur"
```

Last update: haussteuerung:esphome:wetterstation https://drklipper.de/doku.php?id=haussteuerung:esphome:wetterstation&rev=1762495201
2025/11/07 07:00

```
icon: 'mdi:thermometer-alert'  
oversampling: 1x  
filters:  
  - or:  
    - heartbeat: 900s  
    - delta: 0.25  
pressure:  
  name: "BME280 Luftdruck (hPa)"  
  icon: 'mdi:gauge'  
  oversampling: 16x  
  filters:  
    - or:  
      - heartbeat: 900s  
      - delta: 0.35  
humidity:  
  name: "BME280 Luftfeuchte"  
  oversampling: 1x  
  filters:  
    - or:  
      - heartbeat: 900s  
      - delta: 0.25  
- platform: wifi_signal  
  name: "BME280 WiFi Signal"  
  update_interval: 900s
```

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

Permanent link:
<https://drklipper.de/doku.php?id=haussteuerung:esphome:wetterstation&rev=1762495201>

Last update: 2025/11/07 07:00

