Smarter Strom

Remote-Datenerfassung mit Raspberry PI Zero

Die Überwachung des Energieverbrauchs ist ein häufiger Anwendungsfall im Smart Home. In diesem Zusammenhang zeigte bereits der Fachbeitrag „Smarter Strom – Visualisierung der Stromdaten des ELV-USB-IEC in Home Assistant“ eine Möglichkeit auf, die Daten des Stromzählers über einen direkt am Raspberry PI angeschlossenen USB-IEC-Energiesensor in Home Assistant einzubinden. Da sich die Home-Assistant-Installation aber, anders als dort beschrieben, nicht immer in unmittelbarer Näher des Stromzählers befindet, werden in den folgenden Abschnitten weitere Alternativen für die Erfassung und Visualisierung von Stromdaten in Home Assistant vorgestellt.

Systemüberblick

Home Assistant ist eine Open-Source-Hausautomationssoftware, die auf einer Vielzahl von Hardwareplattformen wie einem Raspberry PI, NAS oder sogar einer virtuellen Maschine installiert werden kann. Nicht immer ist es dabei möglich oder gewünscht, die Installation in unmittelbarer Nähe des Stromzählers vorzunehmen. Um dieses Problem zu lösen, wird ein zusätzlicher Vermittler zwischen dem Energiesensor USB-IEC und Home Assistant benötigt.

Aufgrund seines guten Preis-Leistungs-Verhältnisses, seiner kompakten Abmessungen und der einfachen Handhabung eignet sich dazu besonders ein Raspberry PI Zero 2 W. An diesen kann über einen Micro-USB-zu-USB-A-Adapter der USB-IEC-Energiesensor angeschlossen werden. Für die Datenerfassung wird die Software „vzlogger“ und ein Mosquitto MQTT-Broker verwendet. Die Home-Assistant-Installation nimmt die Daten über einen MQTT-Client entgegen und kann diese im integrierten Energiedashboard oder einer individuellen An-

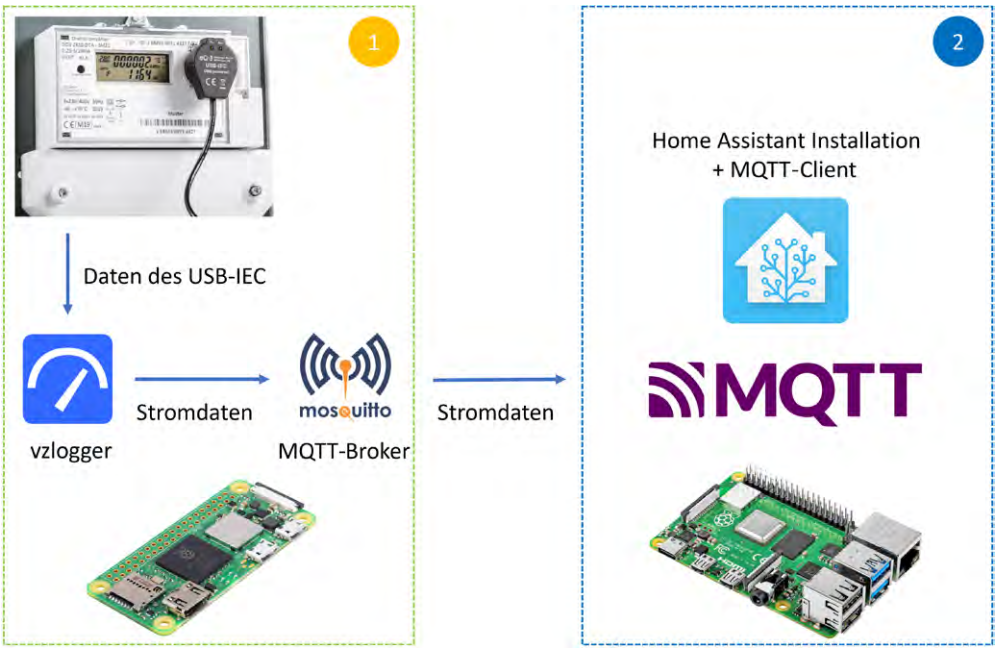

Bild 1: Systemübersicht

sicht darstellen. Das Gesamtsystem besteht also aus zwei Teilsystemen mit einer drahtlosen Verbindung (Bild 1). Die Abschnitte 2 und 3 beschreiben die Installation und Konfiguration aller Komponenten.

Vorbereitung Raspberry PI Zero 2 W und Stromzähler

Der Raspberry PI Zero 2 W nimmt die Daten des USB-IEC-Energiesensors entgegen und überträgt diese via MQTT an Home Assistant. Als Betriebssystem kann die aktuellste Version des Raspberry PI OS Lite in der 64-Bit-Version mithilfe des Raspberry-PI-Imagers installiert werden. Diese Option ist für den Raspberry PI Zero 2 W zwar noch nicht in der Raspberry-PI-Imager-Software verfügbar, jedoch kann das Betriebssystem von der offiziellen Raspberry-PI-Seite heruntergeladen werden.

Die Auswahl des heruntergeladenen Images ist im Bereich „Betriebssystem (OS)“ → „Use Custom“ möglich (Bild 2). Vor der Installation können zudem Anpassungen wie die Konfiguration des Hostnames, Nutzeraccounts oder eines WiFi-Netzwerks erfolgen (Bild 3). Zudem sollte der SSH-Dienst aktiviert werden, um später über einen Remotezugriff auf den PI zugreifen zu können (Bild 4). Nach Abschluss dieser Schritte kann das Image auf die SD Karte geschrieben werden.

Im nächsten Schritt kann die Verbindung zwischen Raspberry PI Zero und ELV-USB-IEC mithilfe des Micro-USB-zu-USB-A erfolgen. Vor Befestigung des USB-IEC-Energiesensors am Stromzähler sollte geprüft werden, ob die optische Schnittstelle und der InF-Mode aktiv sind. Die dazu notwendigen Schritte wurden ebenfalls bereits im Fachbeitrag „Smarter Strom – Visualisierung der Stromdaten des USB-IEC in Home Assistant“ beschrieben.

Installation vzlogger und MQTT-Broker

Das Tool vzlogger ist Teil der Software Volkszähler, die das Auslesen von Smart Metern ermöglicht. Die Installation kann über den SSH-Zugriff auf den Raspberry PI Zero z. B. aus der PowerShell heraus erfol-

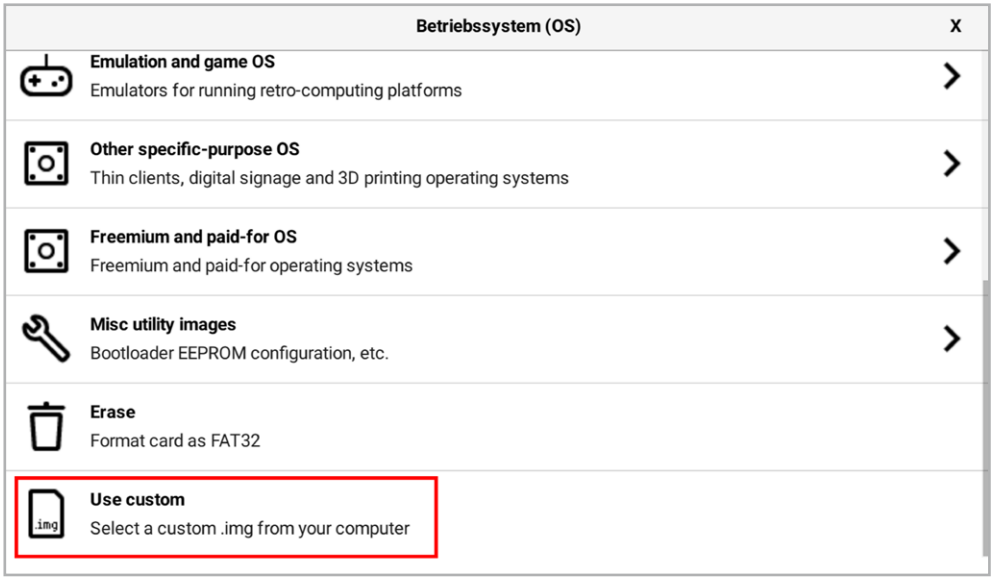

Bild 2: Auswahl des Images

gen. Die Verbindung wird dabei über das Kommando „ssh user@hostname“ aufgebaut. Anschließend erfolgt die Eingabe des zuvor gesetzten Passworts.

Im ersten Schritt sollten alle bereits vorhandenen Pakete mit den Befehlen „sudo apt update“ und „sudo apt upgrade“ aktualisiert werden. Anschließend ermöglicht der folgende Befehl die Installation der benötigten Abhängigkeiten:

sudo apt-get install build-essential git cmake pkg-config subversion libcurl4-openssl-dev libgnutls28-dev libsasl2-dev uuid-dev libtool libssl-dev libgcrypt20-dev libmicrohttpd-dev libltdl-dev libjson-c-dev libleptonica-dev libmosquitto-dev libunistring-dev dh-autoreconf

Das Klonen des GitHub-Repositorys erfolgt über den Befehl:

git clone https://github.com/volkszaehler/vzlogger.git

Innerhalb des Verzeichnisses befindet sich ein Installationsskript, das mit den folgenden Parametern aufgerufen werden sollte, um den gewünschten Funktionsumfang zu erhalten:

cd vzlogger

./install.sh vzlogger libjson libsml mqtt

Vor dem erstmaligen Starten des vzloggers muss eine Konfigurationsdatei angelegt werden, die relevante Einstellungen des angeschlossenen Smart Meters und des MQTT-Brokers beinhaltet. Standardmäßig sucht der vzlogger beim Start im Verzeichnis „etc“ nach dieser Datei, daher sollte sie mit folgendem Befehl dort erstellt werden:

sudo nano vzlogger.conf

Eine beispielhafte Konfigurationsdatei ist im Downlaodbereich dieses Fachbeitrags zu finden. Unterstützend stellt das Volkszähler-Projekt eine Config-Editor-Webseite bereit.

Bild 5 zeigt einen Ausschnitt aus der Konfiguration für das vorhandene Smart Meter. Durch das Setzen des Felds „enabled“ auf „true“ und des Felds „allowskip“ auf „false“ wird das Smart Meter zyklisch ausgelesen. Da aufgrund der hohen Datenmenge nicht jeder Datensatz übertragen werden soll, wird das Sendeintervall durch die Felder „aggtime“ und „aggfixedinterval“ auf eine Minute gesetzt. Die einzelnen Werte des Stromzählers werden im Bereich „channels“ definiert. Jeder „channel“ verfügt dabei über einen „identifier“, dem eine OBIS-Kennzahl zugeordnet ist. Falls weitere Werte benötigt werden, können diese später der Logdatei entnommen werden. Zunächst ist hier jedoch die Erfassung des Zählerstands und der Leistung relevant.

Am Ende der Konfiguration „meter“ ist die Angabe des Zählerprotokolls und des verwendeten USB-Ports notwendig. Da der Raspberry PI Zero 2 W nur über einen einzigen Port verfügt, ist dieser immer „/dev/ttyUSB0“.

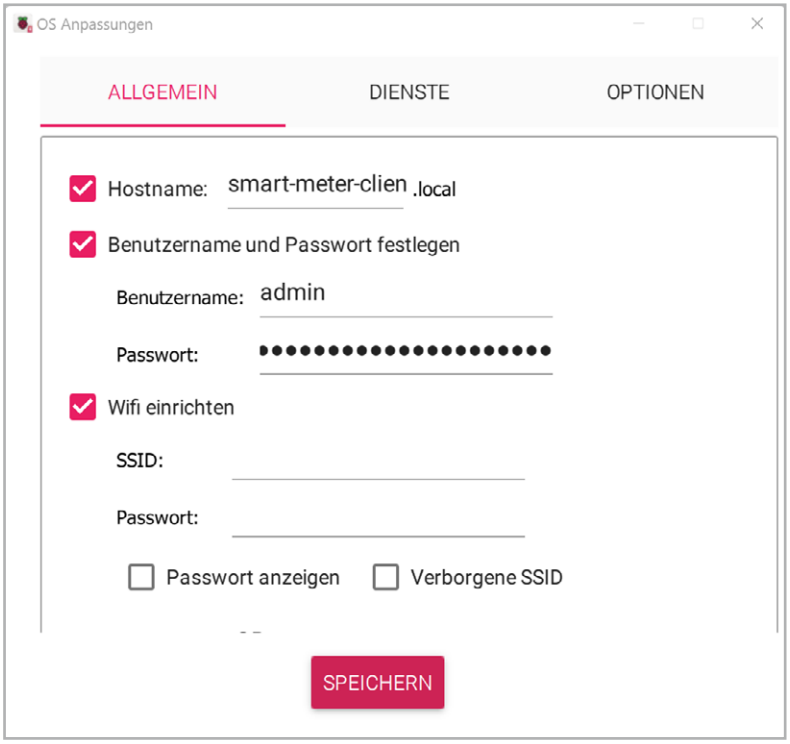

Bild 3: OS-Anpassungen

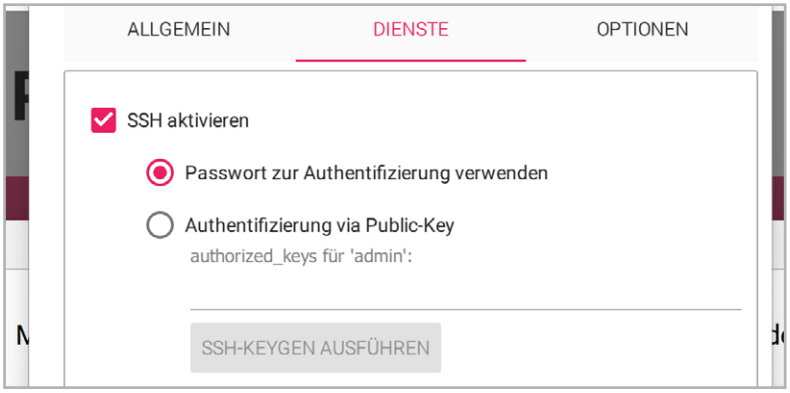

Bild 4: Aktivierung des SSH-Dienstes

```
"meters": [
  {
    "enabled": true,
    "allowskip": false,
    "aggtime": 60,
    "aggfixedinterval": true,
    "channels": [
        {
          "uuid": "0",
          "api": "null",
          "identifier": "1-0:1.8.0*255", //counter
          "aggmode": "max",
        },
        {
          "uuid": "1",
          "api": "null",
          "identifier": "1-0:16.7.0*255", //power
          "aggmode": "max",
        },
    ],
    "protocol": "sml",
    "device": "/dev/ttyUSB0",
    "pullseq": "",
    "baudrate": 9600,
    "parity": "8n1",
    "use_local_time": false
  }
],
```

Bild 5: Ausschnitt aus der vzlogger.conf-Datei

```
"mqtt": {
    "enabled": true,  // enable mqtt client. needs host and port as well
    "host": "127.0.0.1", // mqtt server addr
    "port": 1883, // 1883 for unencrypted, 8883 enc, 8884 enc cert needed,
    "cafile": "", // optional file with server CA
    "capath": "", // optional path for server CAs. see mosquitto.conf. Specif>
    "certfile": "", // optional file for your client certificate (e.g. client>
    "keyfile": "", // optional path for your client certficate private key (e>
    "keypass": "", // optional password for your private key
    "keepalive": 30, // optional keepalive in seconds.
    "topic": "vzlogger/data", // optional topic dont use $ at start and no / >
    "id": "", // optional static id, if not set "vzlogger_<pid>" will be used
    "user": "", // optional user name for the mqtt server
    "pass": "", // optional password for the mqtt server
    "retain": false, // optional use retain message flag
    "rawAndAgg": false, // optional publish raw values even if agg mode is us>
    "qos": 0, // optional quality of service, default is 0
    "timestamp": false // optional whether to include a timestamp in the payl>
```

Bild 6: MQTT-Konfiguration des vzloggers

```
[Mar 05 12:46:33][mtr0] Got 16 new readings from meter:
[Mar 05 12:46:33][mtr0] Reading: id=1-0:1.8.0*255/ObisIdentifier:1-0:1.8.0*255 value
[Mar 05 12:46:33][mtr0] Reading: id=1-0:2.8.0*255/ObisIdentifier:1-0:2.8.0*255 value
[Mar 05 12:46:33][mtr0] Reading: id=1-0:16.7.0*255/ObisIdentifier:1-0:16.7.0*255 val
[Mar 05 12:46:33][mtr0] Reading: id=1-0:36.7.0*255/ObisIdentifier:1-0:36.7.0*255 val
[Mar 05 12:46:33][mtr0] Reading: id=1-0:56.7.0*255/ObisIdentifier:1-0:56.7.0*255 val
[Mar 05 12:46:33][mtr0] Reading: id=1-0:76.7.0*255/ObisIdentifier:1-0:76.7.0*255 val
[Mar 05 12:46:33][mtr0] Reading: id=1-0:32.7.0*255/ObisIdentifier:1-0:32.7.0*255 val
```

Bild 7: Ausschnitt aus der Log-Datei des vzloggers

```
admin@smart-meter-client:/etc/mosquitto $ mosquitto_sub -t "vzlogger/#"
2887896.400000
2696.000000
2887916.100000
2698.000000
```

Bild 8: Testen der vzlogger-Installation

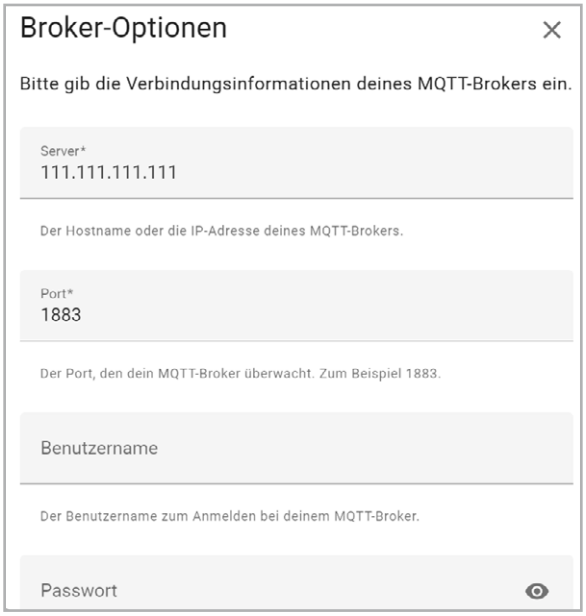

Bild 9: Konfiguration des MQTT-Clients in Home Assistant

Neben der Datenerfassung ist auch die Konfiguration des MQTT-Brokers notwendig (Bild 6). Der Host muss dabei auf den lokalen Mosquitto-Broker eingestellt werden (127.0.0.1).

Für die MQTT-Datenübertragung wird der Mosquitto-Broker installiert. Dies ist über den folgenden Befehl möglich:

sudo apt install mosquitto mosquitto-clients

Dabei wird neben dem Broker auch ein Client installiert, der für die Verifizierung der vzlogger-Funktionalität genutzt werden kann. Um einen Zugriff aus Home Assistant heraus zu ermöglichen, ist eine Anpassung der Konfigurationsdatei nötig:

sudo nano /etc/mosquitto/mosquitto.conf

Dort werden die folgenden Zeilen am Dateiende ergänzt:

listener 1883

allow_anonymous true

Wie jeder Systemdienst kann auch der Mosquitto-Broker zum Autostart hinzugefügt werden mit dem Befehl:

sudo systemctl enable mosquitto

Um nun die vollständige Installation des vzloggers und der MQTT-Kommunikation zu testen, werden beide Dienste zuerst gestartet:

sudo systemctl start mosquitto

sudo systemctl start vzlogger

Anschließend sollte im Verzeichnis „/var/log“ die Datei „vzlogger.log“ auftauchen (Bild 7).

Die Ausgabe der OBIS-Kennzahlen ermöglicht die Konfiguration und Übertragung weiterer Werte an Home Assistant durch das Anlegen eines neuen Channels in der Konfigurationsdatei. Wenn dort alle nötigen Werte eingestellt sind, kann die MQTT-Kommunikation lokal durch folgenden Befehl getestet werden:

mosquitto_sub -t „vzlogger/#“

Durch das Abonnieren des Topics „vzlogger/#“ erscheinen minütlich die aktuellen Werte des Smart Meters (Bild 8). Ist dies der Fall, kann der vzlogger analog zum Mosquitto-MQTT-Broker zum Autostart hinzugefügt werden:

sudo systemctl enable vzlogger

Konfiguration der Home-Assistant-Installation

Zur Visualisierung der Daten des USB-IEC wird eine lauffähige Home-Assistant-Installation benötigt. Eine Installationsanleitung dazu findet sich z. B. im Fachbeitrag Vermittler im smarten Zuhause Teil 6.

Die Entgegennahme der Daten erfolgt über die „MQTT-Integration“. Diese kann im Bereich „Einstellungen“ → „Geräte & Dienste“ über den Button „Integration Hinzufügen“ aktiviert werden. Über die Eingabe des Suchbegriffs „MQTT“ erscheint die passende Integration. In der Broker-Konfiguration muss die IP-Adresse des Raspberry PI Zero eingetragen werden. Anschließend kann die Konfiguration über den Button „Weiter“ abgeschlossen werden (Bild 9). Sofern der Broker gefunden wurde, erscheint die Integration in der Übersicht (Bild 10).

Die Auswahl der einzelnen Messwerte geschieht in der Datei „configuration.yaml“. Diese kann mithilfe des „File Editor“ Add-ons aus dem Home-Assistant-Add-on-Store (Bild 11) editiert werden. Dort wird über den Bezeichner „mqtt“ die zuvor installierte Integration referenziert. Im Anschluss wird je ein Sensor für den Zählerstand und die Leitung definiert. Der Wert des Feldes „state_topic“ richtet sich nach den Channels aus der vzlogger-Konfiguration.

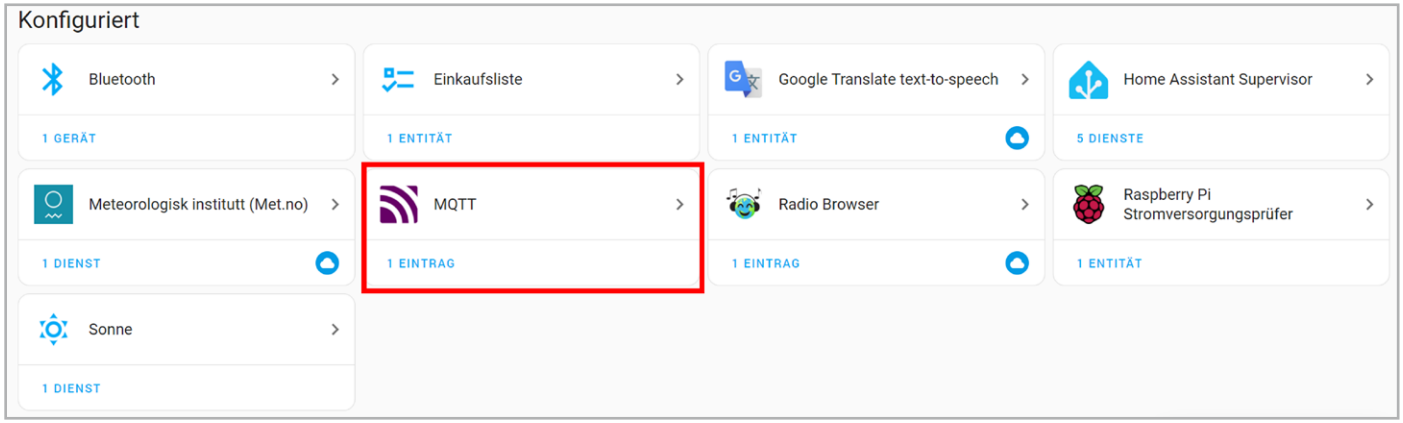

Bild 10: Übersicht der Integrationen

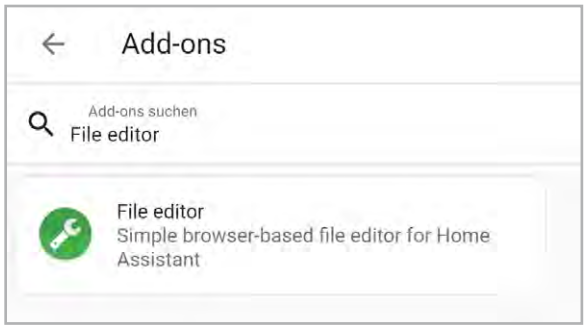

Bild 11: Suche des File-Editors im Add-on-Store

```
# MQTT Entitäten des abesetzten USB-IEC
mqtt:
  sensor:
    - name: "smart-meter-counter"
      state_topic: "vzlogger/data/chn0/agg"
      value_template: "{{ (value_json | float(0)) / 1000 }}"
      device_class: energy
      state_class: total_increasing
      unit_of_measurement: "kWh"
    - name: "smart-meter-power"
      state_topic: "vzlogger/data/chn1/agg"
      value_template: "{{value_json | round(0)}}"
      device_class: power
      state_class: total
      unit_of_measurement: "w"
```

Bild 12: Extraktion der benötigten Entitäten

Für eine bessere Lesbarkeit ist auch die Anpassung der Rohdaten durch das Feld „value_template“ möglich. Der betreffende Ausschnitt aus der Datei configuration.yaml ist in Bild 12 zu sehen.

Nach dem Speichern der Konfigurationsdatei sollten alle YAML-Konfigurationen neu geladen werden, um die neuen Entitäten verwenden zu können. Dies ist im Menü „Entwicklerwerkzeuge“ im Tab „YAML“ über die Schaltfläche „Alle YAML Konfigurationen“ möglich.

Visualisierung in Home Assistant

Ein wesentlicher Bestandteil von Home Assistant sind Dashboards zur Visualisierung und Steuerung der eingebundenen Smart-Home-Geräte. Für die Darstellung der Stromdaten wird in diesem Fall das integrierte Energiedashboard verwendet. Weitere Visualisierungsmöglichkeiten wurden bereits im Fachbeitrag „Smarter Strom – Visualisierung der Stromdaten des ELV-USB-IEC in Home Assistant“ beschrieben.

Das integrierte Energiedashboard kann in Home Assistant entweder direkt aus der Seitenleiste oder über das Menü „Einstellungen“ → „Dashboards“ ausgewählt werden. Insgesamt bietet es umfangreiche Möglichkeiten, alle Verbrauchswerte für Strom, Wasser und Gas zu erfassen. Falls eine Photovoltaikanlage vorhanden ist, können auch die Netzeinspeisung, die einzelnen Sonnenkollektoren und ggf. ein Batteriespeicher überwacht werden.

Die Konfigurationsseite des Energiedashboards kann aus der oberen rechten Ecke des Energiedashboards heraus aufgerufen werden (Bild 13). In der folgenden Ansicht (Bild 14) kann dann im Bereich „Netzverbrauch“ die entsprechende Entität des USB-IEC-Energiesensors ausgewählt werden (Bild 15).

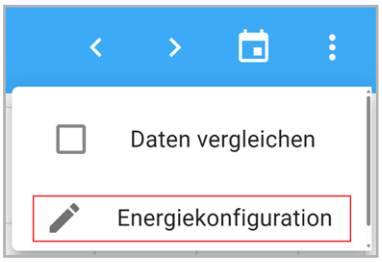

Bild 13: Konfiguration des Energiedashboards

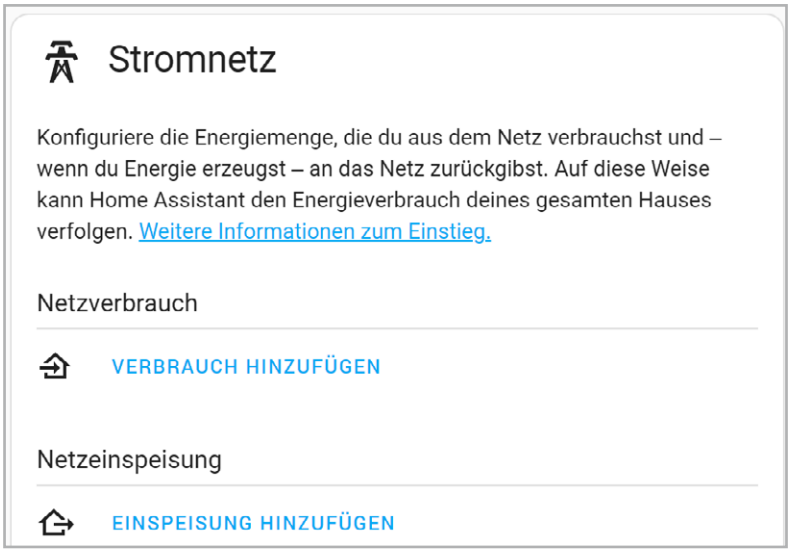

Bild 14: Konfiguration des Stromnetzes

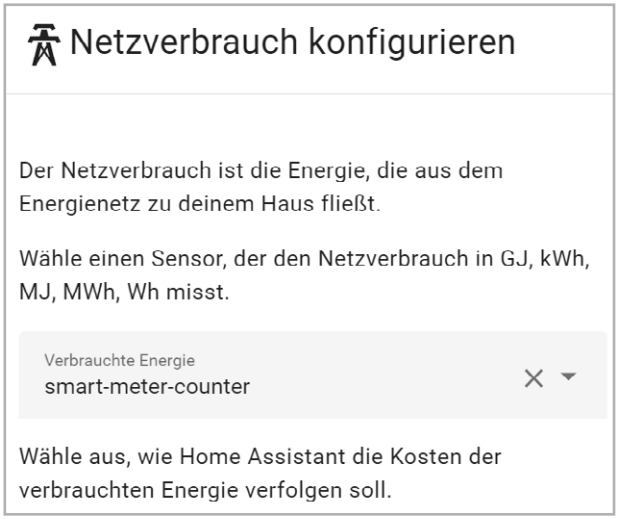

Bild 15: Auswahl der Stromzähler-Entität des USB-IEC

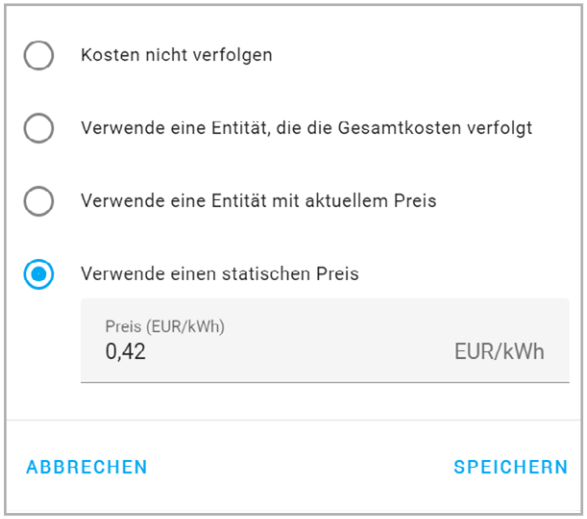

Bild 16: Angabe eines Strompreises

Neben der Überwachung der Zählerstände gibt es auch Möglichkeiten zur Berechnung der Stromkosten. Diese ergeben sich, wie in Bild 16 dargestellt, entweder durch eine weitere Entität für die Gesamtkosten oder eine Entität mit dem aktuellen bzw. statischen Preis. Am Ende wird die Konfiguration über den Button „Speichern" geschlossen.

Bild 17 zeigt das fertige Energiedashboard. Da der Zählerstand stundenweise grafisch dargestellt wird, sind unmittelbar nach der Inbetriebnahme noch keine Daten vorhanden. Neben dem Energieverbrauch werden auch die Energieverteilung und die Kosten visualisiert.

Zusammenfassung und Fazit

Die in diesem Artikel beschriebene Umsetzung ermöglicht eine kostengünstige und flexible Erfassung der Smart-Meter-Daten.

Durch die Nutzung der vzlogger-Software wird die Kommunikation zwischen dem USB-IEC-Energiesensor und dem Raspberry PI abstrahiert und die erfassten Daten können leicht über MQTT an Home Assistant übertragen werden. Innerhalb der Home-Assistant-UI können dann aus dem integrierten Energiedashboard wichtige Informationen zur Überwachung und Optimierung des Stromverbrauchs abgelesen werden. ELV

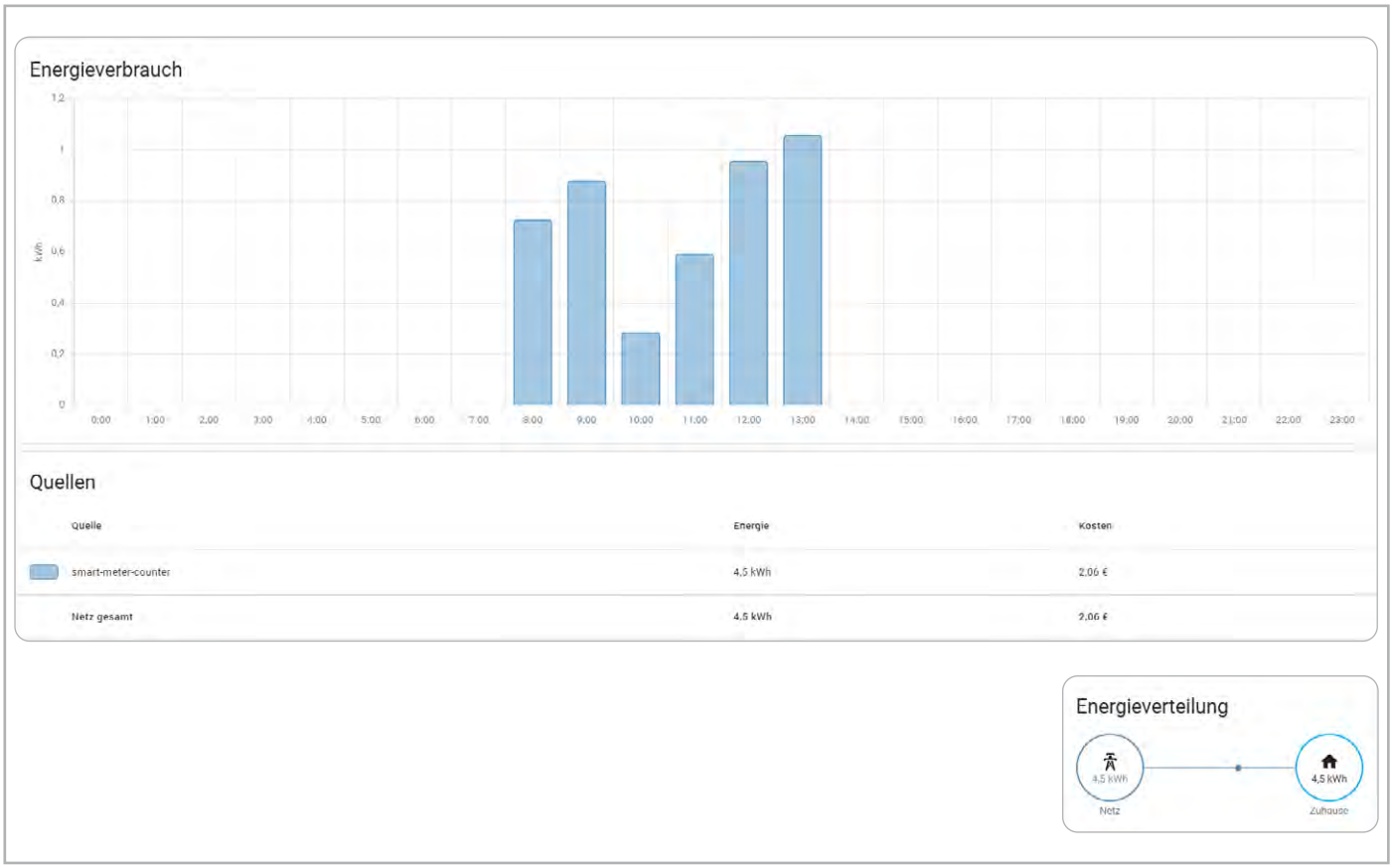

Bild 17: Energiedashboard zur Erfassung der Netzbezugs