Raspberry Pi als APC-USV-Server für Home Assistant

Aufbau: APC Smart-UPS SMT1500I → USB → Raspberry Pi mit apcupsd → TCP 3551 → Home Assistant. Getestet mit Raspberry Pi B+ und Raspberry Pi OS Legacy Lite.

Später ist serielles RS-232-Modbus vorgesehen (Stabilere Verbindung)

1. Raspberry Pi und USV vorbereiten

1. Raspberry Pi OS Lite installieren, SSH und Netzwerk einrichten; eine feste IP/DHCP-Reservierung verwenden. Pi und Netzwerk für einen Stromausfall ebenfalls über die USV versorgen.
2. USV zunächst per USB mit dem Pi verbinden. Im USV-Menü **Advanced / Configuration / Modbus** aktivieren.
3. Software installieren:

```
sudo apt update
sudo apt install -y apcupsd cron
```

2. apcupsd konfigurieren

```
sudo nano /etc/apcupsd/apcupsd.conf
```

Folgende Werte setzen; keine doppelten aktiven Einträge stehen lassen:

```
UPSCABLE usb
UPSTYPE modbus
DEVICE
POLLTIME 30
NETSERVER on
NISIP 0.0.0.0
NISPORT 3551
ONBATTERYDELAY 6
BATTERYLEVEL 10
MINUTES 5
TIMEOUT 0
KILLDELAY 0
```

Begründung: `DEVICE` bleibt bei USB-Modbus leer. `POLLTIME 30` liest die USV ungefähr alle 30 Sekunden aus. `NISIP 0.0.0.0` erlaubt Netzwerkzugriffe: Port `3551` nur im vertrauenswürdigen LAN freigeben. Die Batteriegrenzen betreffen apcupsd auf dem Pi; andere Geräte werden dadurch nicht automatisch heruntergefahren.

Bei älteren Installationen ggf. `ISCONFIGURED=yes` in `/etc/default/apcupsd` setzen. **ACHTUNG:**

Die Zeiten unbeding an die eigenen Bedürfnisse anpassen. Einige Proxmox Server brauchen realtiv lange zum sicheren Herunterfahren. Dort müssen BATTERYLEVEL und MINUTES erhöht werden. Diese beiden sind eine ODER Entscheidung. Je nachdem, welche Kondition zuerst eintritt, also wie hier 10% Batterielevel oder 5 Minuten Restlaufzeit.

```
sudo systemctl enable --now apcupsd
sudo systemctl restart apcupsd
apcaccess status | grep -E '^(DATE|STATUS|END APC)'
```

DATE muss aktuell sein; STATUS lautet normalerweise ONLINE.

3. In Home Assistant einbinden

Einstellungen → Geräte & Dienste → Integration hinzufügen → APC UPS Daemon. IP des Pi und Port **3551** eintragen.

Die HA-Integration fragt standardmäßig **alle 60 Sekunden** ab. Die 30 Sekunden POLLTIME auf dem Pi bedeuten daher keine 30-Sekunden-Aktualisierung in HA. Bei einem Freeze können Werte kurzzeitig deutlich älter werden. Für Abschaltautomationen die Datenaktualität und das Ausfallverhalten berücksichtigen.

4. Dienst bei Prozessabsturz neu starten

```
sudo systemctl edit apcupsd
```

```
[Service]
Restart=on-failure
RestartSec=10s
```

```
sudo systemctl daemon-reload
sudo systemctl restart apcupsd
```

Grenze: Ein eingefrorener USB-Modbus-Datenstrom stoppt den Prozess nicht zwingend. Deshalb ist zusätzlich ein Watchdog nötig.

5. USB-Modbus-Freeze automatisch beheben

Beobachteter Fehler:END APC zeigt die aktuelle Abfragezeit, aber DATE bleibt auf der letzten erfolgreichen USV-Datenübertragung stehen. Ein Neustart von apcupsd stellte die Kommunikation wieder her. **Nicht** die Netzspannung LINEV vergleichen: Ein konstanter Messwert ist kein Fehlernachweis.

Der Watchdog läuft **jede Minute**, meldet ein DATE-Alter über **120 Sekunden**, bestätigt den Fehler nach **10 Sekunden** und startet apcupsd neu. Zwischen zwei Neustarts liegen mindestens **15**

Minuten. LOWBATT und SHUTTING DOWN sind geschützt. Ereignisse werden im Journal protokolliert.

Skript erstellen:

```
sudo nano /usr/local/sbin/apcupsd-watchdog.sh
```

Den folgenden Inhalt **ab der ersten Zeile** einfügen. Der komplette Skriptinhalt steht in einem einzigen DokuWiki-Codeblock:

```
#!/usr/bin/env bash
set -u

MAX_AGE=120
COOLDOWN=900
STATE=/run/apcupsd-watchdog.last-restart
MODE="${1:-}"

case "$MODE" in
    ""|--check) ;;
    *) echo "Aufruf: $0 [--check]" >&2; exit 2 ;;
esac

exec 9>/run/apcupsd-watchdog.lock
if [ "$MODE" = "--check" ]; then
    flock -w 5 9 || { echo "BELEGT: Watchdog laeuft."; exit 2; }
else
    flock -n 9 || exit 0
fi

probe() {
    reason=""; age=""; ups_status=""; health="fault"
    report=$(timeout 10s /usr/sbin/apcaccess status 2>/dev/null) ||
report=""
    date_text=$(printf '%s\n' "$report" | awk '$1 == "DATE" {sub(/^[^:]*:[
\t]*/, ""); print; exit}')
    ups_status=$(printf '%s\n' "$report" | awk '$1 == "STATUS"
{sub(/^[^:]*:[ \t]*/, ""); print; exit}')

    case "$ups_status" in
        *"SHUTTING DOWN"*|*"LOWBATT"*) health="protected"; return ;;
        *"COMMLOST"*) reason="STATUS=COMMLOST"; return ;;
    esac

    if [ -z "$date_text" ] || [ -z "$ups_status" ]; then
        reason="DATE oder STATUS fehlt"; return
    fi

    stamp=$(date -d "$date_text" +%s 2>/dev/null) || stamp=""
    case "$stamp" in
        ""|*[!0-9]*) reason="DATE ungueltig"; return ;;
    esac
```

```
    age=$(($(date +%s) - stamp))
    if [ "$age" -lt -60 ]; then
        health="clock"; reason="Systemzeit pruefen"
    elif [ "$age" -gt "$MAX_AGE" ]; then
        reason="DATE ist ${age}s alt (Grenze ${MAX_AGE}s)"
    else
        health="ok"
    fi
}

probe
if [ "$MODE" = "--check" ]; then
    case "$health" in
        ok) echo "OK: DATE ${age}s alt, STATUS=$ups_status"; exit 0 ;;
        protected) echo "GESCHUETZT: $ups_status"; exit 0 ;;
        clock) echo "WARNUNG: $reason"; exit 2 ;;
        *) echo "FEHLER: $reason (kein Neustart im Testmodus)"; exit 1 ;;
    esac
fi

case "$health" in ok|protected|clock) exit 0 ;; esac
sleep 10
probe
case "$health" in ok|protected|clock) exit 0 ;; esac

last=0
if [ -s "$STATE" ]; then
    read -r last < "$STATE"
    case "$last" in ""|*[!0-9]*) last=0 ;; esac
fi
now=$(date +%s)
if [ "$((now - last))" -lt "$COOLDOWN" ]; then exit 0; fi
printf '%s\n' "$now" > "$STATE"

logger -p daemon.warning -t apcupsd-watchdog "Datenfehler: $reason;
Neustart."
if ! timeout 30s systemctl restart apcupsd; then
    logger -p daemon.err -t apcupsd-watchdog "Neustart fehlgeschlagen."
    exit 1
fi

sleep 35
probe
if [ "$health" = "ok" ]; then
    logger -t apcupsd-watchdog "Wiederhergestellt: DATE ${age}s alt,
STATUS=$ups_status."
else
    logger -p daemon.err -t apcupsd-watchdog "Noch keine aktuellen Daten:
$reason"
```

```
    exit 1
fi
```

Wichtig: Die erste Zeile der Skriptdatei muss `#!/usr/bin/env bash` sein, ohne Leerzeile davor. Danach testen und Cronjob anlegen:

```
sudo chmod 750 /usr/local/sbin/apcupsd-watchdog.sh
sudo bash -n /usr/local/sbin/apcupsd-watchdog.sh
sudo /usr/local/sbin/apcupsd-watchdog.sh --check
sudo crontab -e
```

Dort **nur diesen einen** Watchdog eintragen:

```
* * * * * /usr/local/sbin/apcupsd-watchdog.sh
```

Alte Watchdogs deaktivieren. Kontrolle:

```
sudo crontab -l
systemctl is-active cron
sudo journalctl -t apcupsd-watchdog --since "24 hours ago" --no-pager
```

Praxisnachweis: Am 06.11.2026 erkannte der Watchdog ein DATE-Alter von 164 s, startete `apcupsd` neu und stellte die Daten wieder her. Das umgeht den Freeze, beseitigt aber nicht dessen USB-Modbus-Ursache. Die USV-Abschaltung darf nicht allein von möglicherweise veralteten HA-Sensorwerten abhängen.

.

Umstieg auf serielles Modbus (RS-232)

Bei der SMT1500I sind USB-Modbus-Freezes dokumentiert. **RS-232-Modbus** vermeidet den betroffenen USB-Modbus-Kommunikationsweg und ist voraussichtlich stabiler, ohne eine absolute Garantie zu bieten.

Benötigt werden das passende **APC-Kabel AP940-0625A** (USV-Buchse mit RJ45/RJ50-Bauform auf DB9) und ein Linux-kompatibler **USB-RS232-Adapter**. Der APC-Anschluss ist **kein Ethernet und kein RS485**. Wegen der speziellen Pinbelegung kein beliebiges Kabel verwenden.

Pinout: APC Smart-UPS 5G cable pins and signals

Nach dem Umstecken serielle Schnittstelle ermitteln:

```
ls -l /dev/serial/by-id/
```

Beispiel:

```
DEVICE /dev/serial/by-id/usb-Prolific_Technology_Inc._USB-
```

```
Serial_Controller_D-if00-port0
```

In `/etc/apcupsd/apcupsd.conf` nur die Verbindung ändern (Gerätepfad anpassen):

```
UPSCABLE smart
UPSTYPE modbus
DEVICE /dev/serial/by-id/usb-DEIN_ADAPTER
```

```
sudo systemctl restart apcupsd
apcaccess status | grep -E '^(DATE|STATUS|END APC)'
```

Port 3551, Home-Assistant-Integration und Watchdog bleiben gleich. Die serielle Variante muss nach dem Umbau noch praktisch getestet werden.

---

6. Fehlersuche

```
apcaccess status | grep -E '^(DATE|STATUS|END APC)'
sudo systemctl status apcupsd --no-pager
sudo journalctl -u apcupsd -n 40 --no-pager
sudo journalctl -t apcupsd-watchdog -n 30 --no-pager
sudo  journalctl -u apcupsd --since "today"
```

- DATE alt, END APC aktuell: USB-Modbus-Daten eingefroren.
- DATE aktuell, HA-Werte alt: Netzwerk oder HA-Integration prüfen.
- Im Test waren Pi-Unterspannung (`throttled=0x0`) und USB-Autosuspend (`power/control=on`) unauffällig.

7. Zusätzliche Netzwerk-Clients anbinden (Master/Slave-Betrieb)

Wenn an der APC Smart-UPS 1500 mehrere Server oder Systeme (z. B. ein Proxmox-Server, ein PC oder ein zweites NAS) betrieben werden, müssen auch diese über einen Stromausfall informiert werden. Da das USB-Kabel physisch im Raspberry Pi (**Master**) steckt, können weitere Systeme als **Netzwerk-Clients (Slaves)** eingerichtet werden. Diese lauschen über das Netzwerk dem Pi.

Die Installation von `apcupsd` auf dem Ziel-Client erfolgt analog zu den Schritten 1 und 2. In der Konfigurationsdatei des Clients müssen jedoch die Netzwerk-Parameter angepasst werden.

Schritt 1: Konfiguration auf dem Client-System anpassen

Datei auf dem zusätzlichen Server öffnen:

```
```

```
sudo nano /etc/apcupsd/apcupsd.conf
```

Folgende Werte setzen und darauf achten, dass keine doppelten, aktiven Einträge in der Datei stehen bleiben (alte USB-Einträge mit # auskommentieren):

```
# Verbindungstyp auf Netzwerk (Ethernet) umstellen
UPSCABLE ether
UPSTYPE net

# IP-Adresse und Port des Raspberry Pi (Masters) eintragen
DEVICE 192.168.10.248:3551

# Abfrage-Intervall zum Master (in Sekunden)
POLLTIME 30

# Erhöhte Sicherheits-Grenzwerte für den Client
BATTERYLEVEL 20
MINUTES 10
```

Dienst auf dem Client neu starten:

```
sudo systemctl restart apcupsd
```

Erklärung der Sicherheits-Grenzwerte (Wichtig!)

Die Werte BATTERYLEVEL und MINUTES wurden auf dem Client bewusst **höher angesetzt** als auf dem Raspberry Pi (Master):

* **BATTERYLEVEL 20:** Der Client fährt bereits bei **20 % Restkapazität** kontrolliert herunter. * **MINUTES 10:** Der Client fährt herunter, sobald die verbleibende Akkulaufzeit unter **10 Minuten** fällt.

Der Kaskaden-Ablauf im Ernstfall: Durch diese Staffelung wird verhindert, dass der Raspberry Pi (Master) zeitgleich mit den Clients herunterfährt. Der Pi bleibt länger aktiv und stellt sicher, dass er die Abschalt-Befehle im Netzwerk erfolgreich an alle Clients senden kann. Erst wenn die Clients sicher ausgeschaltet sind und die USV weiter gegen die kritische 5-%-Marke sinkt, fährt sich der Pi als allerletztes Gerät im Rack herunter.

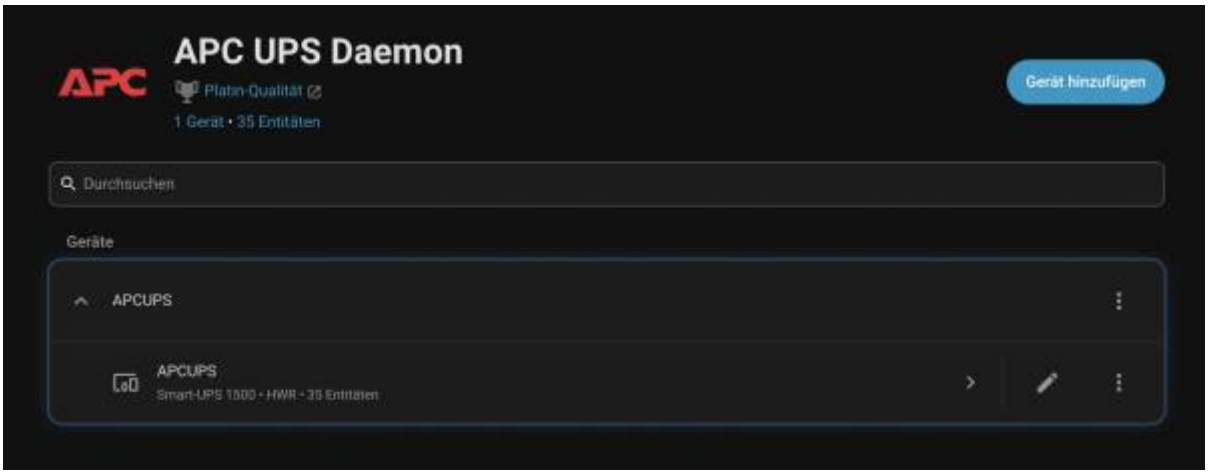
Parameter	Einstellung	Einfache Erklärung
UPSCABLE	`ether`	Sagt dem System, dass die USV nicht per Kupferdraht (USB), sondern virtuell über das Netzwerk-Kabel (Ethernet) erreichbar ist.
UPSTYPE	`net`	Stellt den Treiber in den Client-Modus. Das System sucht nicht nach physikalischer Hardware, sondern lauscht auf USV-Daten aus dem Netzwerk.

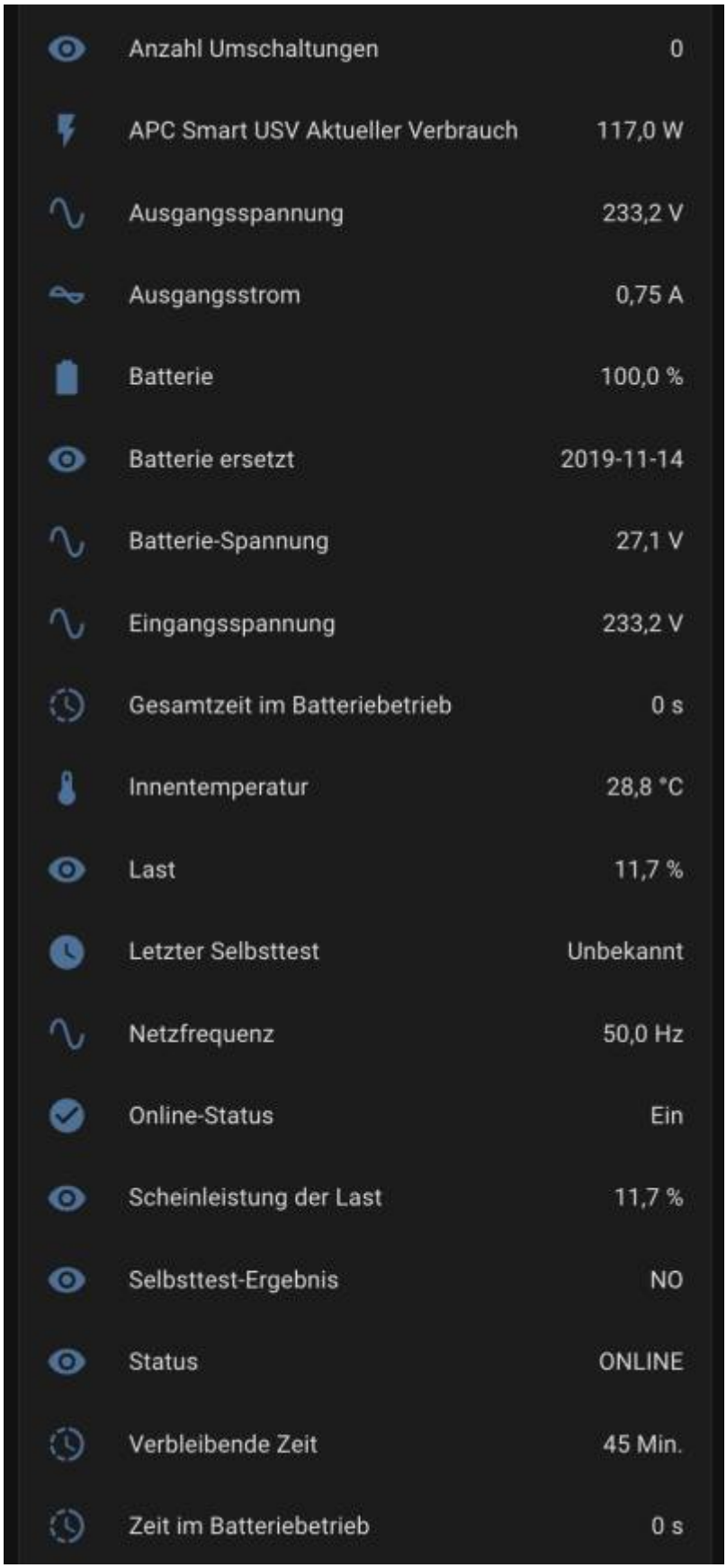
Dokumentation bei Kolja Link

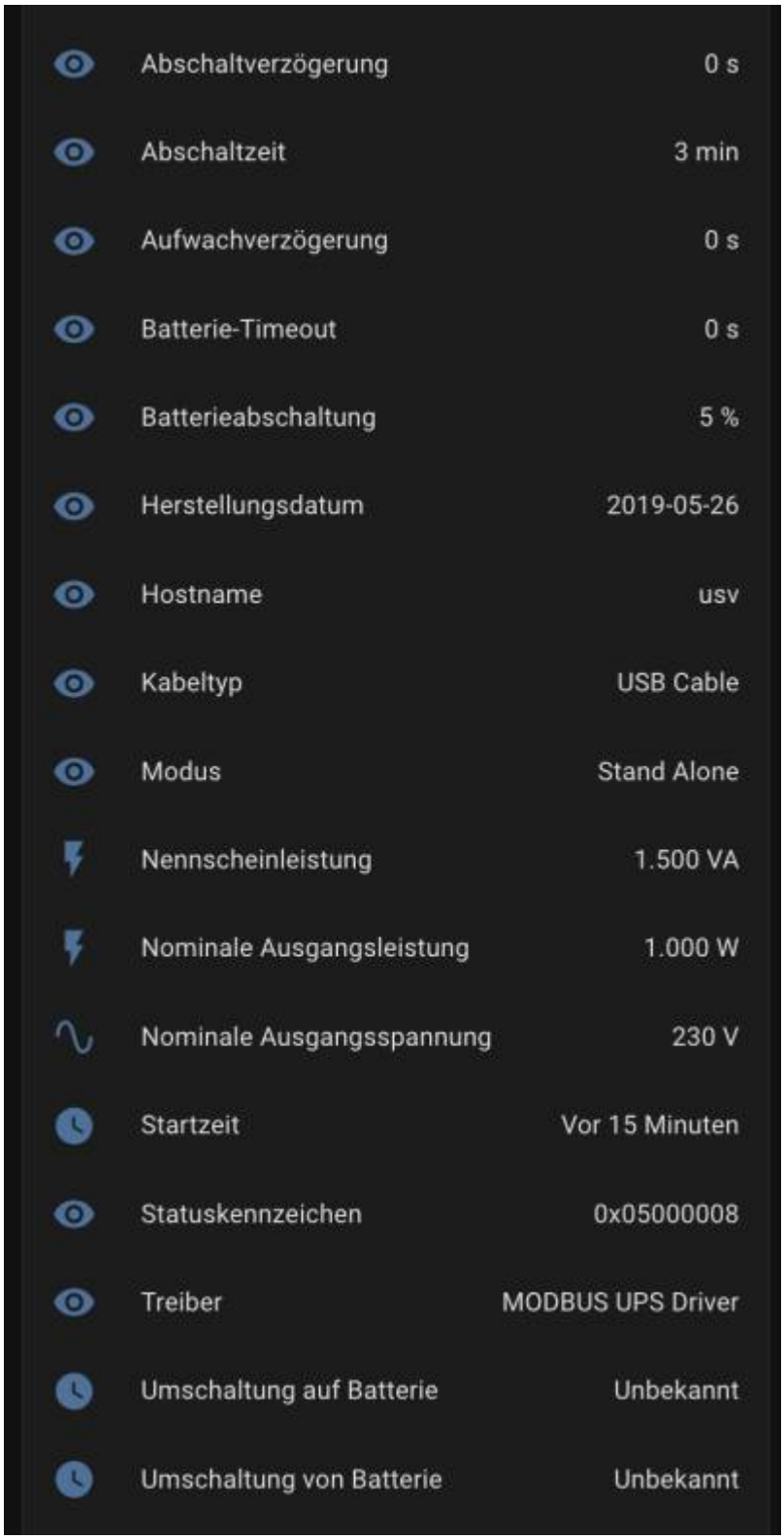
Quellen und weiterführende Links

- Home Assistant – APC UPS Daemon Integration
- Debian – apcaccess, DATE und END APC
- apcupsd – Konfigurationsreferenz
- apcupsd – Bekannter USB-Modbus-Freeze bei SMT1500I
- apcupsd – Benutzerhandbuch mit serieller Modbus-Konfiguration
- Schneider Electric – Kompatible APC-Kommunikationskabel
- Schneider Electric – Serielles Kabel AP940-0625A
- Schneider Electric – Warnung vor falschen seriellen Kabeln

Bilder




Anzahl Umschaltungen 0
APC Smart USV Aktueller Verbrauch 117,0 W
Ausgangsspannung 233,2 V
Ausgangsstrom 0,75 A
Batterie 100,0 %
Batterie ersetzt 2019-11-14
Batterie-Spannung 27,1 V
Eingangsspannung 233,2 V
Gesamtzeit im Batteriebetrieb 0 s
Innentemperatur 28,8 °C
Last 11,7 %
Letzter Selbsttest Unbekannt
Netzfrequenz 50,0 Hz
Online-Status Ein
Scheinleistung der Last 11,7 %
Selbsttest-Ergebnis NO
Status ONLINE
Verbleibende Zeit 45 Min.
Zeit im Batteriebetrieb 0 s



From:
https://wiki.mahlen.eu/ - **Smart-Home Wiki**

Permanent link:
https://wiki.mahlen.eu/doku.php?id=home_assistant:home_assistant_rezepte:apcusv_einbinden

Last update: **07.11.2026**