USVWatch - USV-Status Benachrichtigung per E-Mail auf UGREEN NAS (UGOS Pro)

Handbuch (DE/EN) - Version v1.00 - Stand 10.02.2026

Community-Lösung, kein offizielles UGREEN-Produkt. Verwendung auf eigene Verantwortung.

Note: The English manual starts after the German section at Page 9.

0. Hinweis zu Installationspfad und Pfadänderungen

Dieses Handbuch nutzt als Beispiel den Ordner:

/volume1/docker/usvwatch

Du kannst USVWatch auch in einem anderen Ordner betreiben. Wenn du einen anderen Pfad verwendest oder den Ordner später verschiebst, passe bitte folgende Stellen an:

1. **Beispiel-Kommandos** im Handbuch (alle Aufrufe von python3 …/usvwatch.py)
2. **Cron Minute-Guard** Eintrag (Pfad zu usvwatch-loop.sh)
3. Optional: Falls du Log- oder State-Dateien in einen anderen Ordner legen möchtest, passe die Pfade entsprechend an.

Wichtig: In der mitgelieferten Version ist usvwatch-loop.sh so gebaut, dass es automatisch im eigenen Ordner läuft. Du musst dort in der Regel nichts anpassen, solange das Script zusammen mit usvwatch.py im gleichen Ordner liegt.

0.1 Schnellstart (DE)

- Ordner anlegen (z.B. /volume1/docker/usvwatch) und usvwatch.py + usvwatch.env hineinkopieren.
- usvwatch.env ausfüllen
- usvwatch.env schützen: chmod 600 /volume1/docker/usvwatch/usvwatch.env
- usvwatch-loop.sh ausführbar machen: chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh
- Status prüfen: python3 /volume1/docker/usvwatch/usvwatch.py --print-status
- Testmail senden: python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
- Automatisierung einrichten (empfohlen: Cron Minute-Guard), 10-Sekunden-Loop bei Systemstart.

1. Einleitung

USVWatch ist ein Python-Skript zur Benachrichtigung über den USV/UPS Status auf UGREEN NAS (UGOS Pro). Es liest den Status über NUT (upsd) aus, erkennt wichtige Ereignisse (z.B. Stromausfall und Rückkehr) und versendet eine HTML-Mail per SMTP. Python 3 ist auf UGOS bereits vorinstalliert, das Skript nutzt ausschließlich die Python-Standardbibliothek.

2. Funktionsumfang

- Erkennt Stromausfall (OB) und Rückkehr zum Netzbetrieb (OL).
- Erkennt Low Battery (LB) sowie optionale Warnungen: Akkuladung unter Schwelle, Laufzeit unter Schwelle.
- Optionaler Tagesreport (zeitgesteuert).
- HTML-Report mit Kurzstatus (Akkuladung, Laufzeit, Spannung, Last) und optional kompletter NUT Variablenliste.
- Status-Codes (OL/OB/LB usw.) als Legende in der Mail (optional).
- Sprachwahl DE/EN per ENV (USVWATCH_LANG).
- Optional: NUT Kommandos (INSTCMD) ausführen (z.B. beeper.mute) und Ergebnis per Mail senden.
- Spam-Schutz über State-Datei und Re-Send Intervalle.

3. Paket-Inhalt und Ordnerstruktur

Im Paket enthalten:

- usvwatch.py - Hauptskript (NUT Abfrage, Statuslogik, HTML-Mail).
- usvwatch.env - Konfiguration (Sprache, NUT, SMTP, Schwellwerte, Optionen).
- usvwatch-loop.sh - Wrapper für den 10-Sekunden-Loop (wird mitgeliefert).

Empfohlener Speicherort ist ein Ordner auf einem Daten-Volume, z.B.:

```
/volume1/docker/usvwatch/
```

4. Voraussetzungen

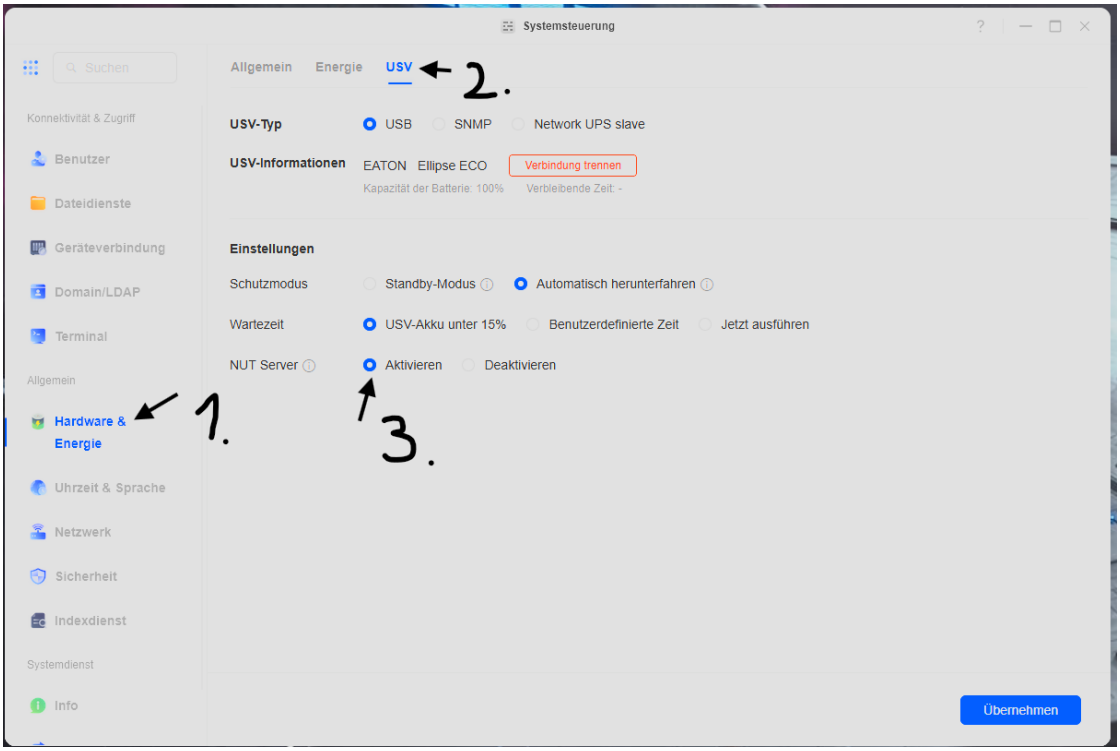
- UGREEN NAS mit UGOS Pro und aktivierter USV-Funktion (Systemsteuerung -> Hardware & Energie -> USV).
- SSH-Zugriff (für Installation und Cronjob).
- Python 3 (auf UGOS bereits vorinstalliert).
- NUT Dienst erreichbar (Standard: localhost:3493).
- Zugriff auf einen SMTP-Server (Port 587 STARTTLS oder 465 SMTPS).

5. Installation auf UGREEN NAS

Schnellstart (7 Schritte)

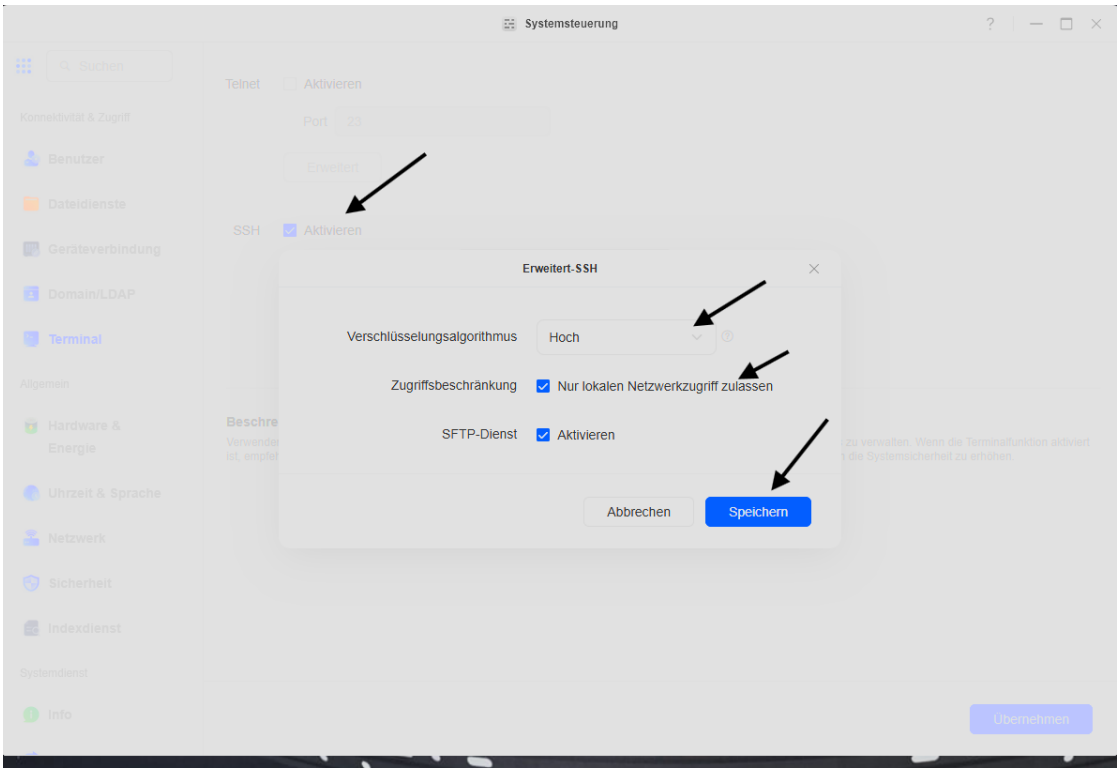
USV in UGOS aktivieren und prüfen, dass die USV erkannt wird.

NUT Server aktivieren: Systemsteuerung -> Hardware & Energie -> USV -> NUT Server -> Aktivieren.

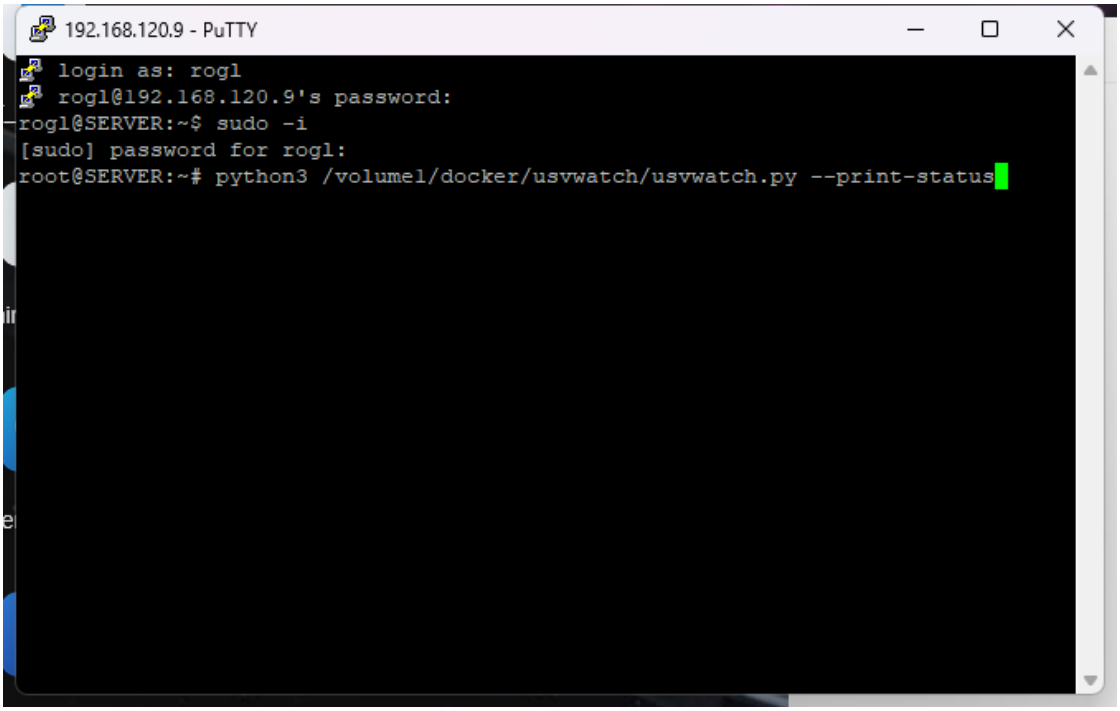


1. Dateien auf das NAS kopieren (z.B. nach /volume1/docker/usvwatch/).
2. usvwatch.env anpassen (MAIL_TO, SMTP_* , NUT_* , Schwellwerte usw.).
3. SSH auf dem NAS aktivieren: Systemsteuerung -> Terminal -> SSH aktivieren. Empfehlung: "Nur lokalen Netzwerkzugriff zulassen".

Beispiel (UGOS Pro): SSH nur im lokalen Netzwerk aktivieren.



4. PuTTY Verbindung zum NAS herstellen (Windows). Beispiel: Host/IP 192.168.120.4, Port 22.



5. Root-Rechte erlangen: sudo -i
6. Skript testen: python3 /volume1/docker/usvwatch/usvwatch.py --print-status und danach python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
7. Cronjob anlegen (Minute-Guard): crontab -e und einen Eintrag hinzufügen (jede Minute).

Beispiel Cron-Eintrag:

```
* * * * * /bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start >/dev/null 2>&1
```

5.1 Dateien kopieren

1) Projektordner anlegen (Beispiel):

```
mkdir -p /volume1/docker/usvwatch
```

2) usvwatch.py und usvwatch.env in diesen Ordner kopieren (per SMB oder Datei-App).

3) Rechte setzen (empfohlen):

```
chmod 600 /volume1/docker/usvwatch/usvwatch.env
chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh
```

5.2 Konfiguration usvwatch.env

Öffne usvwatch.env und passe die Variablen an. Kurzüberblick (vollständige Liste siehe unten):

Variable	Beschreibung	Beispiel/Default
USVWATCH_LANG	Sprache (de oder en)	de
NUT_HOST	NUT Server Hostname/IP	127.0.0.1
SMTP_HOST	SMTP Hostname/IP	smtp.example.com
SMTP_PORT	Port 587 (STARTTLS) oder 465 (SMTPS)	587
SMTP_TLS	TLS aktivieren (1/0)	1
SMTP_TLS_VERIFY	Zertifikat prüfen (1/0)	1
SMTP_USER	SMTP Benutzer (optional)	nas@example.com
SMTP_PASS	SMTP Passwort/App-Passwort (optional)	APP_PASSWORD
MAIL_TO	Empfänger (kommagetrennt möglich)	you@example.com

5.3 Vollständige ENV-Referenz

Alle Optionen aus usvwatch.env (Defaultwerte siehe Datei):

Bereich	Variablen (Auszug)	Kurzbeschreibung
Allgemein	USVWATCH_LANG, HOST_LABEL, STATE_DIR, DEBUG	Sprache, Host-Anzeige, State-Verzeichnis, Debug.
NUT	NUT_HOST, NUT_PORT, NUT_TIMEOUT, NUT_UPS_NAME, NUT_USERNAME, NUT_PASSWORD	NUT Verbindung und optional Auth.
SMTP	SMTP_HOST, SMTP_PORT, SMTP_TLS, SMTP_TLS_VERIFY, SMTP_USER, SMTP_PASS, SMTP_TIMEOUT	SMTP Verbindung, TLS und Login.
Mail	MAIL_FROM, MAIL_TO, MAIL_CC, MAIL_BCC, SUBJECT_PREFIX	Absender/Empfänger, Betreff-Prefix.
Alerts	ALERT_ON_BATTERY, ALERT_BACK_ONLINE, ALERT_LOW_BATTERY, ALERT_UNREACHABLE, ALERT_RECOVERED	Welche Ereignisse eine Mail auslösen.
Schwellwerte	CHARGE_THRESHOLD_PERCENT, RUNTIME_THRESHOLD_MIN, THRESHOLD_RESEND_MIN	Warnungen + optionales Re-Send.
Reports/Details	ENABLE_DAILY_REPORT, DAILY_REPORT_TIME, INCLUDE_ALL_VARS, INCLUDE_STATUS_LEGEND, ALL_VARS_*	Tagesreport, Details und Filter.
Loop/Logging	LOOP_INTERVAL_SECONDS, LOOP_LOG, LOOP_PIDFILE	10s Loop (Wrapper), Log und PID-Datei.

6. Verwendung

6.1 Hilfe und Test

Hilfe anzeigen:

```
python3 /volume1/docker/usvwatch/usvwatch.py --help
```

Testmail senden (ohne Statusänderung):

```
python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
```

Status als JSON ausgeben (Debug):

```
python3 /volume1/docker/usvwatch/usvwatch.py --print-status
```

Kommandos anzeigen und Beispiel-Kommando ausführen:

```
python3 /volume1/docker/usvwatch/usvwatch.py --list-commands
python3 /volume1/docker/usvwatch/usvwatch.py --run-cmd beeper.mute --run-cmd-mail

# Wrapper ist im Paket enthalten
chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh

# Start
/bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start

# Stop
/bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh stop

# Log prüfen
tail -n 200 /volume1/docker/usvwatch/usvwatch-loop.log
```

6.2 Sprache per ENV/CLI

USVWATCH_LANG kann in usvwatch.env gesetzt werden oder temporär per ENV überschrieben werden:

```
USVWATCH_LANG=en python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
```

Die ENV-Variante hat Priorität vor dem Wert in der env-Datei. Das gilt auch für Betreff und Texte.

6.3 Alternative ENV-Datei

Wenn du mehrere Konfigurationen nutzen möchtest, kannst du den Pfad per --env setzen:

```
python3 /volume1/docker/usvwatch/usvwatch.py --env /volume1/docker/usvwatch/usvwatch.env --print-status
```

7. Status und Logik

7.1 Status-Flags (NUT ups.status)

- OL: Netzbetrieb (On Line).
- OB: Batteriebetrieb (On Battery).
- LB: Batterie schwach/kritisch (Low Battery).
- Weitere Flags (z.B. RB, CHRG, DISCHRG, BYPASS) werden in der Legende erklärt, falls INCLUDE_STATUS_LEGEND=1.

7.2 Schwellwerte und Re-Send

CHARGE_THRESHOLD_PERCENT und RUNTIME_THRESHOLD_MIN lösen Warnmails aus, wenn die Werte unterschritten werden (falls aktiviert).

THRESHOLD_RESEND_MIN steuert, ob bei dauerhaftem Unterschreiten nach X Minuten erneut gesendet wird (0 = nie).

7.3 NUT Kommandos und Selbsttest (optional)

Mit --list-commands siehst du verfügbare INSTCMDs. Nicht jede USV unterstützt Selbsttests über NUT.

Falls ein Kommando 'ERR USERNAME-REQUIRED' liefert, prüfe NUT_USERNAME und NUT_PASSWORD in usvwatch.env. Die Werte sind in der Beispiel-ENV bereits vorbelegt (UGOS Standard). Bitte nichts in UGOS/NUT ändern.

7.4 Tagesreport (optional)

ENABLE_DAILY_REPORT=1 sendet täglich zur Uhrzeit DAILY_REPORT_TIME einen Report, auch ohne Ereignis.

7.5 Unreachable/Recovered

Wenn der NUT Dienst nicht erreichbar ist, kann eine Mail gesendet werden (ALERT_UNREACHABLE=1). Bei Wiederherstellung optional ALERT_RECOVERED=1.

8. Automatisierung

8.1 Cron Minute-Guard (empfohlen)

Beispiel: jede Minute prüfen und starten (ohne Konsolen-Ausgabe):

```
* * * * * /bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start >/dev/null 2>&1
```

Cron bearbeiten (als root):

```
crontab -e
```

8.2 Dateien für Loop (Wrapper)

usvwatch-loop.sh ist im Paket enthalten. Es hält eine PID-Datei und verhindert doppelte Instanzen.

Das Wrapper-Skript startet einen 10-Sekunden-Loop und ruft usvwatch.py in jeder Runde einmal auf.

9. Logs, State und Dateien

- Logdatei (Wrapper): usvwatch-loop.log (Pfad siehe Wrapper).
- PID-Datei (Wrapper): usvwatch.loop.pid (verhindert doppelte Instanzen).
- State-Datei: usvwatch_state.json (letzter Status, Zeitstempel und Re-Send Timer, im STATE_DIR).

Empfehlung: STATE_DIR und Logs auf ein Daten-Volume legen, um die Systempartition zu schonen.

10. Troubleshooting

- Keine Mail: SMTP-Daten prüfen, Ports 587/465, Firewall, App-Passwort verwenden. SMTP_TLS und SMTP_TLS_VERIFY prüfen.
- Status nicht erreichbar: NUT Dienst prüfen (Systemsteuerung -> USV), NUT_HOST/NUT_PORT, Netzwerk/Firewall.
- Kommando ERR INVALID-ARGUMENT: Kommando ist für dein UPS-Modell/Driver nicht verfügbar. Erst --list-commands prüfen.

Zum Debuggen: DEBUG=1 setzen und Logdatei prüfen.

11. Sicherheitshinweise

- Das Skript läuft typischerweise als root (Cron). Daher nur aus vertrauenswürdiger Quelle verwenden.
- NUT Zugangsdaten in UGOS nicht ändern. USV-Steuerung erfolgt auf eigenes Risiko.

12. Update und Deinstallation

Update: usvwatch.py durch neue Version ersetzen, usvwatch.env behalten. Danach einmal --print-status und --test-mail ausführen.

Deinstallation: Ordner löschen (inkl. Logs/State). Cron-Eintrag entfernen, falls gesetzt.

13. Hinweis

Dieses Paket ist eine Community-Lösung. Keine Gewährleistung. Nutzung auf eigene Verantwortung.

USVWatch - UPS status notifications for UGREEN NAS (UGOS Pro)

Handbook (DE/EN) - Version v1.00 - Updated 10.02.2026

Community solution, not an official UGREEN product. Use at your own risk.

0. Note about install path and moving the folder

This manual uses the following example path:

/volume1/docker/usvwatch

You can install USVWatch in a different folder. If you use a different path, or move the folder later, update these locations:

1. **Example commands** in this manual (all python3 …/usvwatch.py calls)
2. **Cron minute-guard** entry (path to usvwatch-loop.sh)
3. Optional: If you want logs or state files in a different directory, adjust the paths accordingly.

Important: The included usvwatch-loop.sh is designed to run from its own directory automatically. In most cases you do not need to edit it, as long as usvwatch-loop.sh and usvwatch.py stay in the same folder.

0.1 Quick start (EN)

- Create a folder (for example /volume1/docker/usvwatch) and copy usvwatch.py + usvwatch.env into it.
- Fill out usvwatch.env
- Protect the env file: chmod 600 /volume1/docker/usvwatch/usvwatch.env
- Make usvwatch-loop.sh executable: chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh
- Print status (debug): python3 /volume1/docker/usvwatch/usvwatch.py --print-status
- Send a test email: python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
- Automation: (recommended) cron minute-guard, 10-second loop at boot.

1. Introduction

USVWatch is a Python script that sends UPS status notifications from a UGREEN NAS (UGOS Pro). It reads data via NUT (upsd), detects important events (for example on-battery and back on line), and sends an Outlook-friendly HTML email via SMTP. Python 3 is preinstalled on UGOS, and the script uses only the Python standard library.

2. Features

- Detects power failure (OB) and return to line power (OL).
- Detects low battery (LB) plus optional warnings: charge below threshold, runtime below threshold.
- Optional daily report (time-based).
- HTML report with summary (charge, runtime, voltage, load) and optional full NUT variable list.
- Status code legend (OL/OB/LB etc.) in the email (optional).
- Language selection DE/EN via env (USVWATCH_LANG).
- Optional: run NUT commands (INSTCMD) such as beeper.mute and email the result.
- Anti-spam using a state file and configurable re-send intervals.

3. Package contents and folder layout

Included files:

- usvwatch.py - main script (NUT query, status logic, HTML email).
- usvwatch.env - configuration (language, NUT, SMTP, thresholds, options).
- usvwatch-loop.sh - wrapper for the 10-second loop (included).

Recommended location is a folder on a data volume, for example:

```
/volume1/docker/usvwatch/
```

4. Requirements

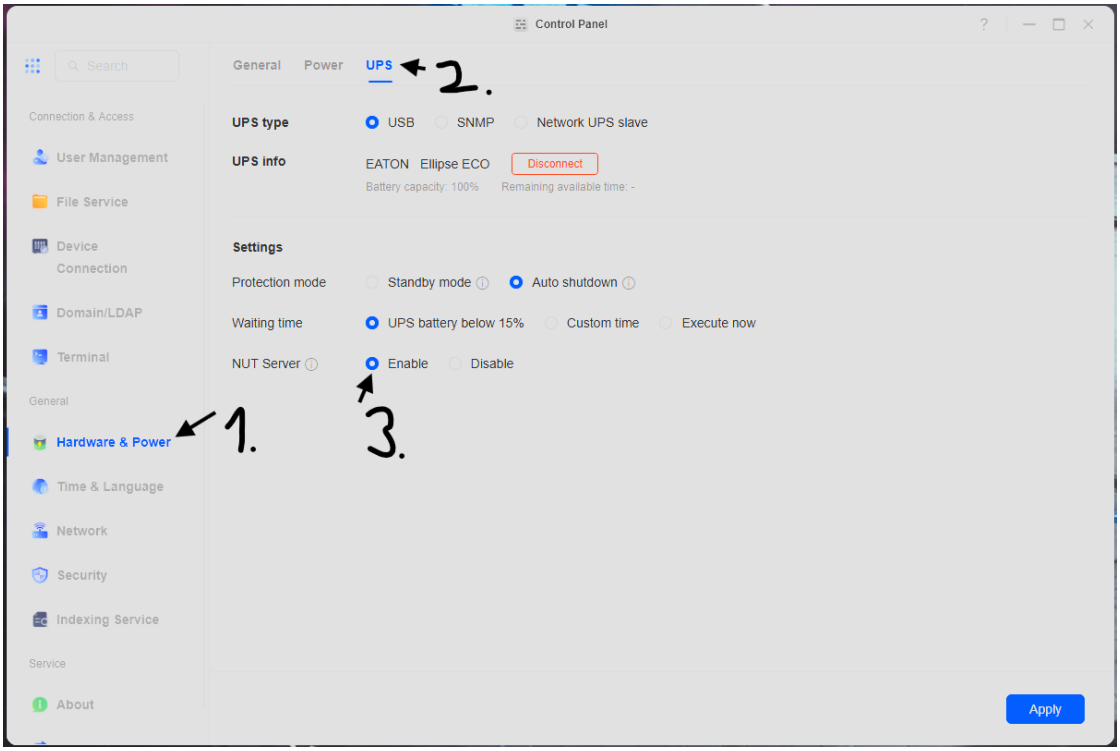
- UGREEN NAS running UGOS Pro with UPS enabled (Control Panel -> Hardware & Power -> UPS).
- SSH access (for installation and cron).
- Python 3 (preinstalled on UGOS).
- NUT service reachable (default: localhost:3493).
- Access to an SMTP server (port 587 STARTTLS or 465 SMTPS).

5. Installation on UGREEN NAS

Quick start (7 steps)

Connect the UPS and make sure it is detected in UGOS.

Enable the NUT server: Control Panel -> Hardware & Power -> UPS -> NUT Server -> Enable.

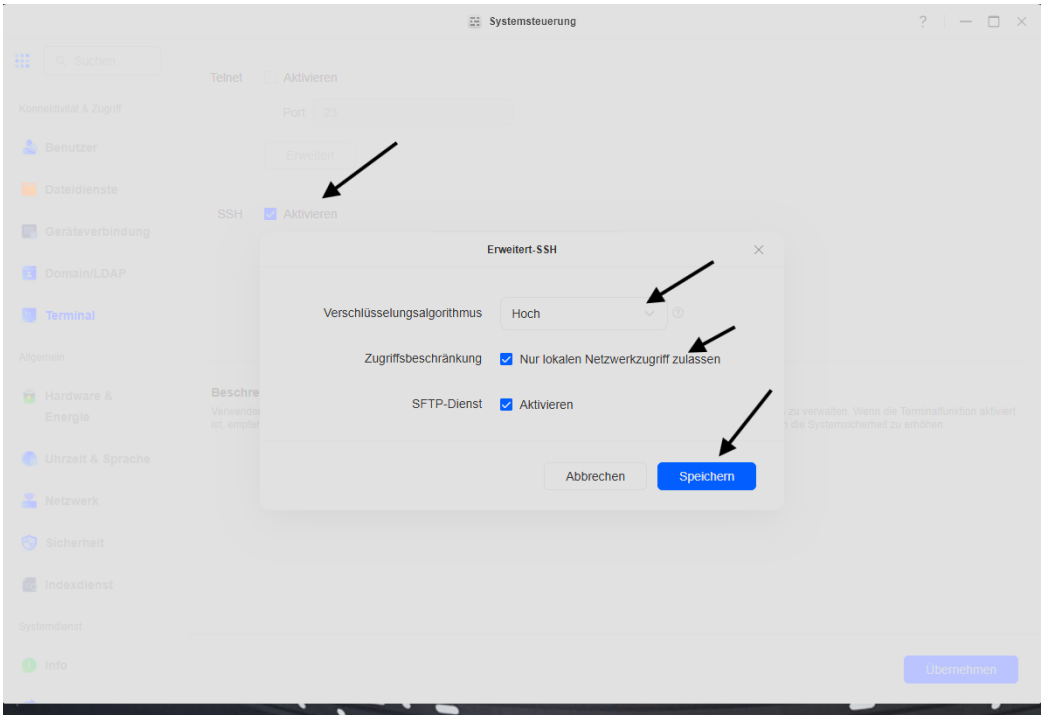


1. Copy the files to the NAS (for example to /volume1/docker/usvwatch/).

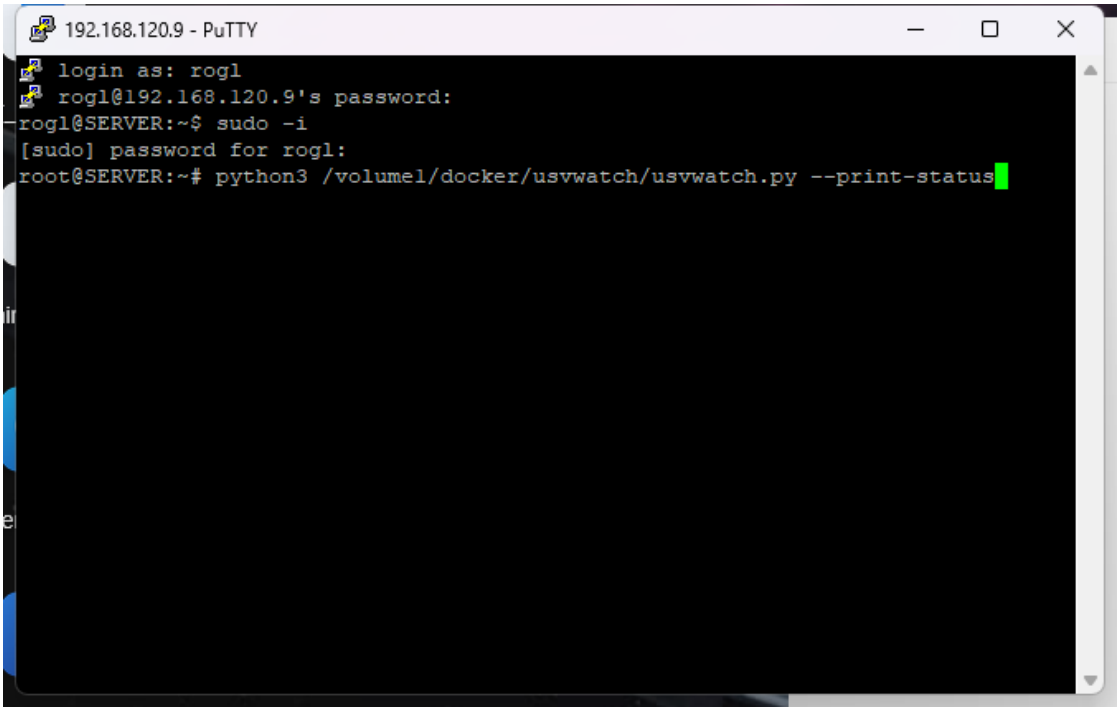
2. Edit usvwatch.env (MAIL_TO, SMTP_* , NUT_* , thresholds, etc.).

3. Enable SSH on the NAS: Control Panel -> Terminal -> Enable SSH. Recommendation: allow local network access only.

Example (UGOS Pro): enable SSH for local network access only.



4. Connect via PuTTY (Windows). Example: Host/IP 192.168.120.4, Port 22.



5. Become root: sudo -i

6. Test the script: python3 /volume1/docker/usvwatch/usvwatch.py --print-status and then python3 /volume1/docker/usvwatch/usvwatch.py --test-mail

7. Create the cron job (minute-guard): crontab -e and add an entry (every minute).

Example cron entry:

```
* * * * * /bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start >/dev/null 2>&1
```

5.1 Copy files

1) Create the project folder (example):

```
mkdir -p /volume1/docker/usvwatch
```

2) Copy usvwatch.py and usvwatch.env into that folder (via SMB or the file app).

3) Set permissions (recommended):

```
chmod 600 /volume1/docker/usvwatch/usvwatch.env
chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh
```

5.2 Configure usvwatch.env

Edit usvwatch.env and adjust the variables. Quick overview (full list below):

Variable	Description	Example/Default
USVWATCH_LANG	Language (de or en)	de
NUT_HOST	NUT server hostname/IP	127.0.0.1
SMTP_HOST	SMTP hostname/IP	smtp.example.com
SMTP_PORT	Port 587 (STARTTLS) or 465 (SMTPS)	587
SMTP_TLS	Enable TLS (1/0)	1
SMTP_TLS_VERIFY	Verify cert (1/0)	1
SMTP_USER	SMTP user (optional)	nas@example.com
SMTP_PASS	SMTP password/app password (optional)	APP_PASSWORD
MAIL_TO	Recipients (comma-separated supported)	you@example.com

5.3 Full env reference

All options from usvwatch.env (see the file for defaults):

Area	Variables (excerpt)	Short description
General	USVWATCH_LANG, HOST_LABEL, STATE_DIR, DEBUG	Language, host label, state directory, debug.
NUT	NUT_HOST, NUT_PORT, NUT_TIMEOUT, NUT_UPS_NAME, NUT_USERNAME, NUT_PASSWORD	NUT connection and optional auth.
SMTP	SMTP_HOST, SMTP_PORT, SMTP_TLS, SMTP_TLS_VERIFY, SMTP_USER, SMTP_PASS, SMTP_TIMEOUT	SMTP connection, TLS and login.
Mail	MAIL_FROM, MAIL_TO, MAIL_CC, MAIL_BCC, SUBJECT_PREFIX	Sender/recipients, subject prefix.
Alerts	ALERT_ON_BATTERY, ALERT_BACK_ONLINE, ALERT_LOW_BATTERY, ALERT_UNREACHABLE, ALERT_RECOVERED	Which events trigger an email.
Thresholds	CHARGE_THRESHOLD_PERCENT, RUNTIME_THRESHOLD_MIN, THRESHOLD_RESEND_MIN	Warnings + optional re-send.
Reports/Details	ENABLE_DAILY_REPORT, DAILY_REPORT_TIME, INCLUDE_ALL_VARS, INCLUDE_STATUS_LEGEND, ALL_VARS_*	Daily report, details and filters.
Loop/Logging	LOOP_INTERVAL_SECONDS, LOOP_LOG, LOOP_PIDFILE	10s loop (wrapper), log and PID file.

6. Usage

6.1 Help and testing

Show help:

```
python3 /volume1/docker/usvwatch/usvwatch.py --help
```

Send a test email:

```
python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
```

Print status as JSON:

```
python3 /volume1/docker/usvwatch/usvwatch.py --print-status
```

List commands and run an example command:

```
python3 /volume1/docker/usvwatch/usvwatch.py --list-commands
python3 /volume1/docker/usvwatch/usvwatch.py --run-cmd beeper.mute --run-cmd-mail

# Wrapper is included in the package
chmod +x /volume1/docker/usvwatch/usvwatch-loop.sh

# Start
/bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start

# Stop
/bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh stop

# Check log
tail -n 200 /volume1/docker/usvwatch/usvwatch-loop.log
```

6.2 Language via env/CLI

Set USVWATCH_LANG in usvwatch.env or override it temporarily via environment variable:

```
USVWATCH_LANG=en python3 /volume1/docker/usvwatch/usvwatch.py --test-mail
```

The environment variable overrides the value from the env file (subject and texts).

6.3 Alternative env file

If you use multiple configs, pass an explicit env file via --env:

```
python3 /volume1/docker/usvwatch/usvwatch.py --env /volume1/docker/usvwatch/usvwatch.env --print-status
```

7. Status and logic

7.1 Status flags (NUT ups.status)

- OL: On line (utility power).
- OB: On battery.
- LB: Low battery / critical.
- Other flags (RB, CHRG, DISCHRG, BYPASS, etc.) are explained in the legend when INCLUDE_STATUS_LEGEND=1.

7.2 Thresholds and re-send

CHARGE_THRESHOLD_PERCENT and RUNTIME_THRESHOLD_MIN trigger warning emails when the values drop below the thresholds (if enabled).

THRESHOLD_RESEND_MIN controls optional repeat emails for persistent warnings (0 = never).

7.3 NUT commands and self test (optional)

Use --list-commands to see available INSTCMDs. Not every UPS supports self tests via NUT.

If a command returns 'ERR USERNAME-REQUIRED', verify NUT_USERNAME and NUT_PASSWORD in usvwatch.env. Defaults are already filled in the example env (UGOS default). Please do not change UGOS/NUT settings.

7.4 Daily report (optional)

ENABLE_DAILY_REPORT=1 sends a report every day at DAILY_REPORT_TIME, even without events.

7.5 Unreachable/Recovered

If the NUT service becomes unreachable, an email can be sent (ALERT_UNREACHABLE=1). When it recovers, optionally send ALERT_RECOVERED=1.

8. Automation

8.1 Cron minute-guard (recommended)

Example: run every minute and start the loop if needed (no console output):

```
* * * * * /bin/sh /volume1/docker/usvwatch/usvwatch-loop.sh start >/dev/null 2>&1
```

Edit cron (as root):

```
crontab -e
```

8.2 Loop helper files (wrapper)

usvwatch-loop.sh is included in the package. It keeps a PID file and prevents duplicate instances.

The wrapper runs a 10-second loop and calls usvwatch.py once per cycle.

9. Logs, state and files

- Wrapper log: usvwatch-loop.log (path in wrapper).
- Wrapper PID file: usvwatch.loop.pid (prevents duplicates).
- State file: usvwatch_state.json (last status and re-send timers, in STATE_DIR).

Tip: put STATE_DIR and logs on a data volume to reduce writes to the system partition.

10. Troubleshooting

- No email: verify SMTP settings, ports 587/465, firewall, use an app password. Check SMTP_TLS and SMTP_TLS_VERIFY.
- Status unreachable: check UPS/NUT service (Control Panel -> UPS), NUT_HOST/NUT_PORT and network/firewall.
- ERR INVALID-ARGUMENT: the command is not supported by your UPS/driver. Run --list-commands first.

For debugging: set DEBUG=1 and check the wrapper log.

11. Security notes

- The script usually runs as root (cron). Only use it from trusted sources.
- Do not change NUT credentials on UGOS. Executing UPS commands is at your own risk.

12. Update and uninstall

Update: replace usvwatch.py with the new version, keep usvwatch.env. Then run --print-status and --test-mail once.

Uninstall: delete the folder (including logs/state). Remove the cron entry if configured.

13. Disclaimer

This is a community project. No warranty. Use at your own risk.