Addons - MQTT

MQTT (Message Queuing Telemetry Transport) ist ein leichtgewichtiges Protokoll zur Nachrichtenübermittlung, das speziell für Geräte mit begrenzten Ressourcen und Netzwerkbandbreite entwickelt wurde. Es wird häufig in Internet-of-Things (IoT)-Anwendungen eingesetzt. Hier sind einige zentrale Punkte:

- **Leichtgewichtig**: Geringer Overhead und niedriger Energieverbrauch, ideal für ressourcenbeschränkte Geräte.
- **Publish/Subscribe-Modell**: Geräte (Clients) können Nachrichten an Themen (Topics) veröffentlichen (publish) oder Nachrichten von Themen abonnieren (subscribe), was eine flexible und skalierbare Kommunikation ermöglicht.
- **Zuverlässigkeit**: Bietet verschiedene QoS (Quality of Service)-Stufen, um die Zuverlässigkeit der Nachrichtenübertragung sicherzustellen.
- **Asynchron**: Unterstützt eine asynchrone Kommunikation, wodurch die Geräte nicht ständig verbunden sein müssen.
- **Offenes Protokoll**: MQTT ist ein offenes Protokoll, das von der OASIS (Organization for the Advancement of Structured Information Standards) standardisiert wurde.

Durch diese Eigenschaften eignet sich MQTT besonders gut für die Vernetzung und Kommunikation von IoT-Geräten, wie Sensoren, Aktoren und anderen eingebetteten Systemen.

Installation

Beschreibung:

MQTT (Message Queuing Telemetry Transport) ist ein leichtgewichtiges Protokoll zur Nachrichtenübermittlung, das speziell für Geräte mit begrenzten Ressourcen und Netzwerkbandbreite entwickelt wurde. Es wird häufig in Internet-of-Things (IoT)-Anwendungen eingesetzt. Hier sind einige zentrale Punkte:

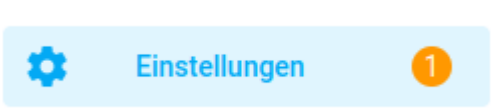
- **Leichtgewichtig**: Geringer Overhead und niedriger Energieverbrauch, ideal für ressourcenbeschränkte Geräte.
- **Publish/Subscribe-Modell**: Geräte (Clients) können Nachrichten an Themen (Topics) veröffentlichen (publish) oder Nachrichten von Themen abonnieren (subscribe), was eine flexible und skalierbare Kommunikation ermöglicht.
- **Zuverlässigkeit**: Bietet verschiedene QoS (Quality of Service)-Stufen, um die Zuverlässigkeit der Nachrichtenübertragung sicherzustellen.
- **Asynchron**: Unterstützt eine asynchrone Kommunikation, wodurch die Geräte nicht ständig verbunden sein müssen.
- **Offenes Protokoll**: MQTT ist ein offenes Protokoll, das von der OASIS (Organization for the Advancement of Structured Information Standards) standardisiert wurde.

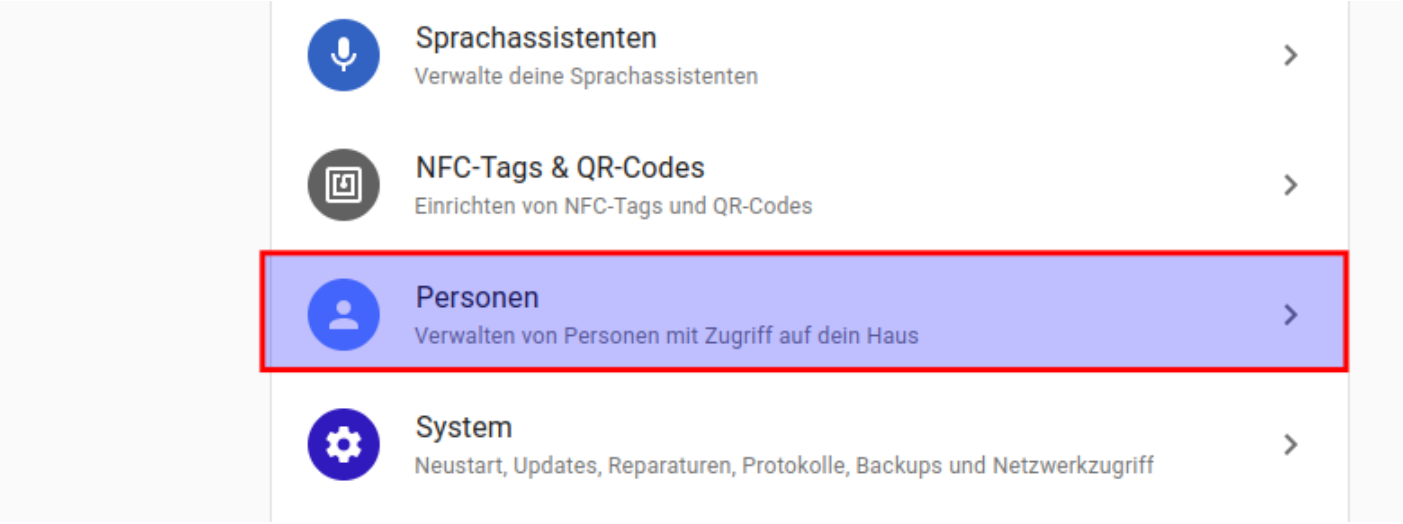
Durch diese Eigenschaften eignet sich MQTT besonders gut für die Vernetzung und Kommunikation von IoT-Geräten, wie Sensoren, Aktoren und anderen eingebetteten Systemen.

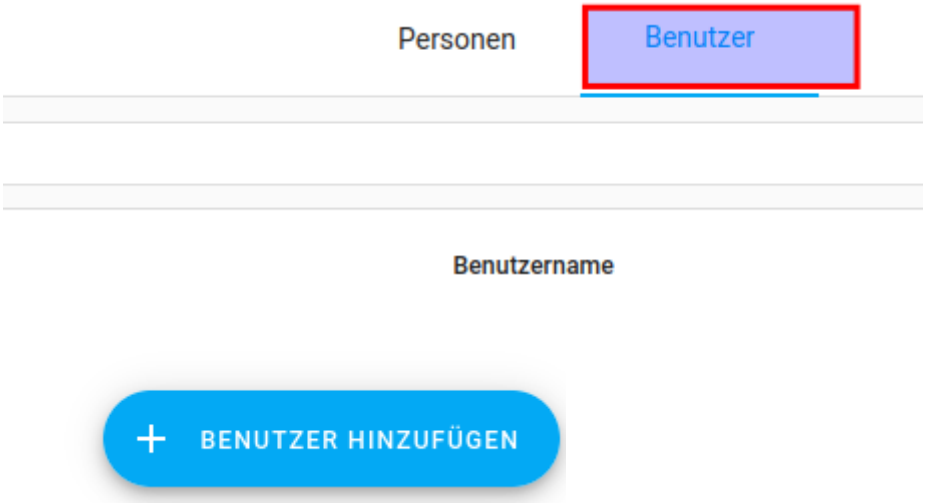
Voraussetzungen:

Einen zusätzlichen Benutzer den wir für MQTT Verwenden wollen.

Dazu auf Einstellungen -> Personen -> auf den Register Benutzer -> Benutzer hinzufügen

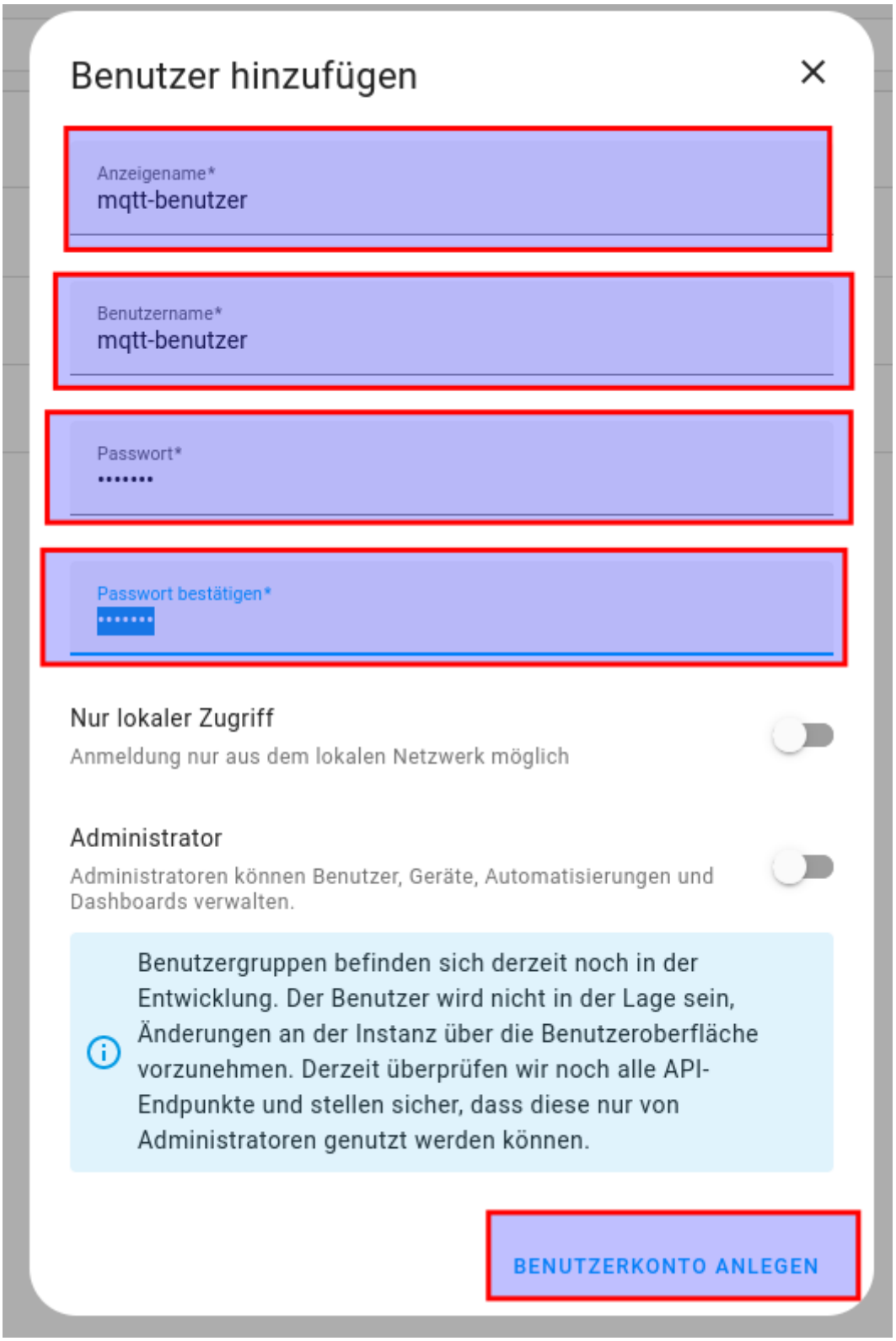






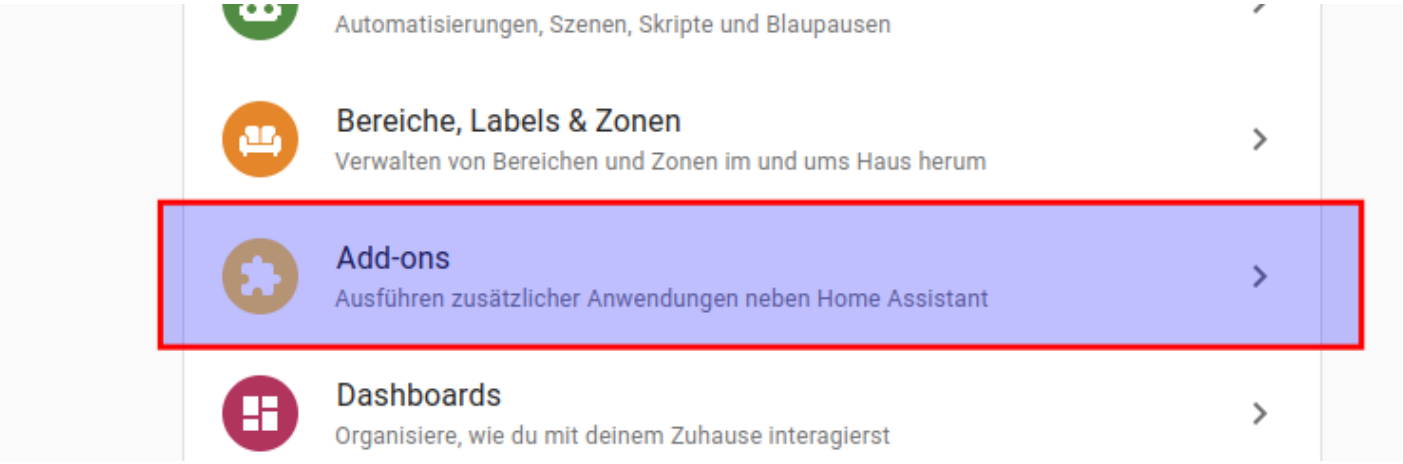
Nun einen Benutzernamen und Kennwort vergeben.

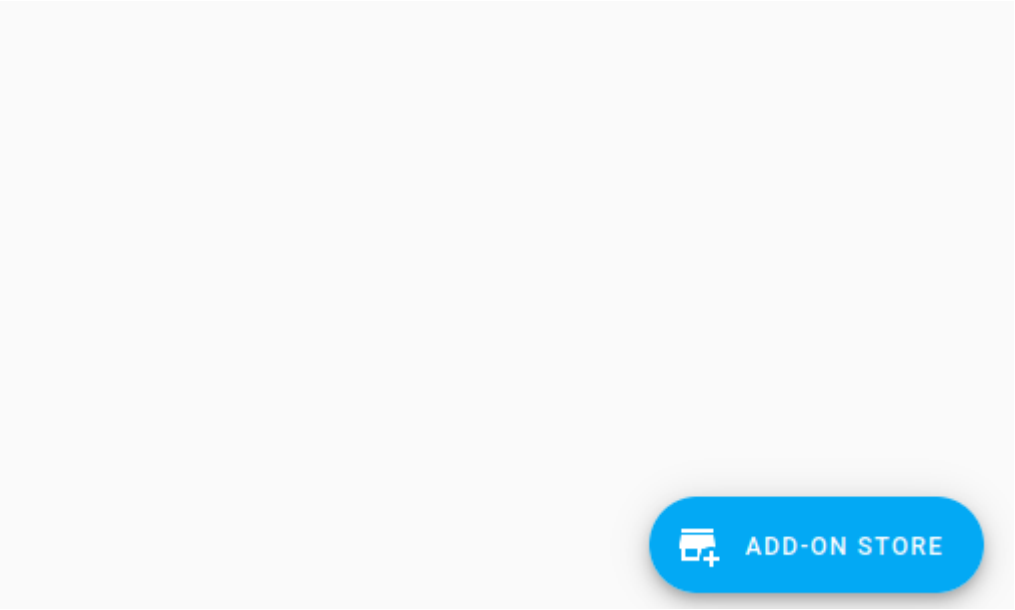
Benutzer MQTT-Benutzer und Kennwort denkt euch eins aus, dann auf Benutzer anlegen.
Das Benutzerpassword darf keine Sonderzeichnen wie das Ausrufezeichen enthalten!!!



Installation MQTT Broker:

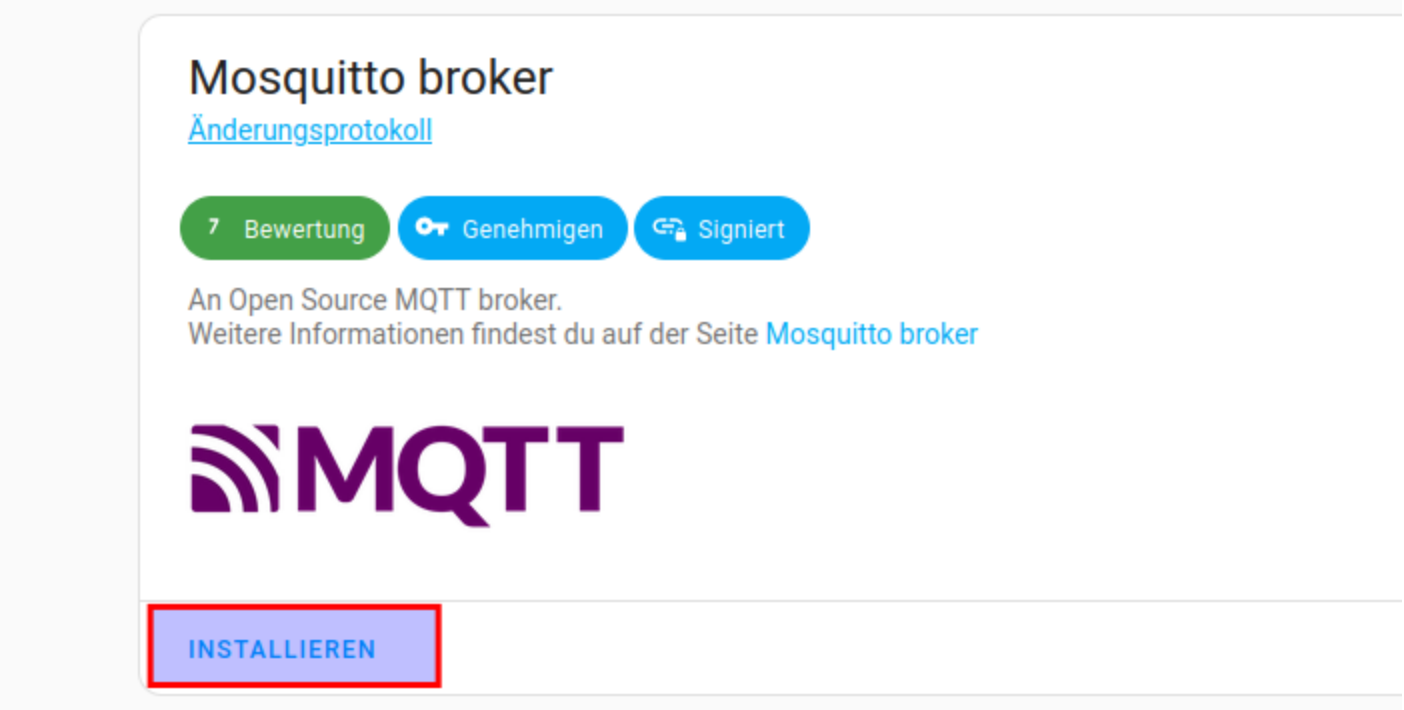
Dazu auf Einstellungen Addons -> Addoon store klicken





dort in der suche mqt eingeben. Dann auf den Offiziellen MQTT Broker klicken

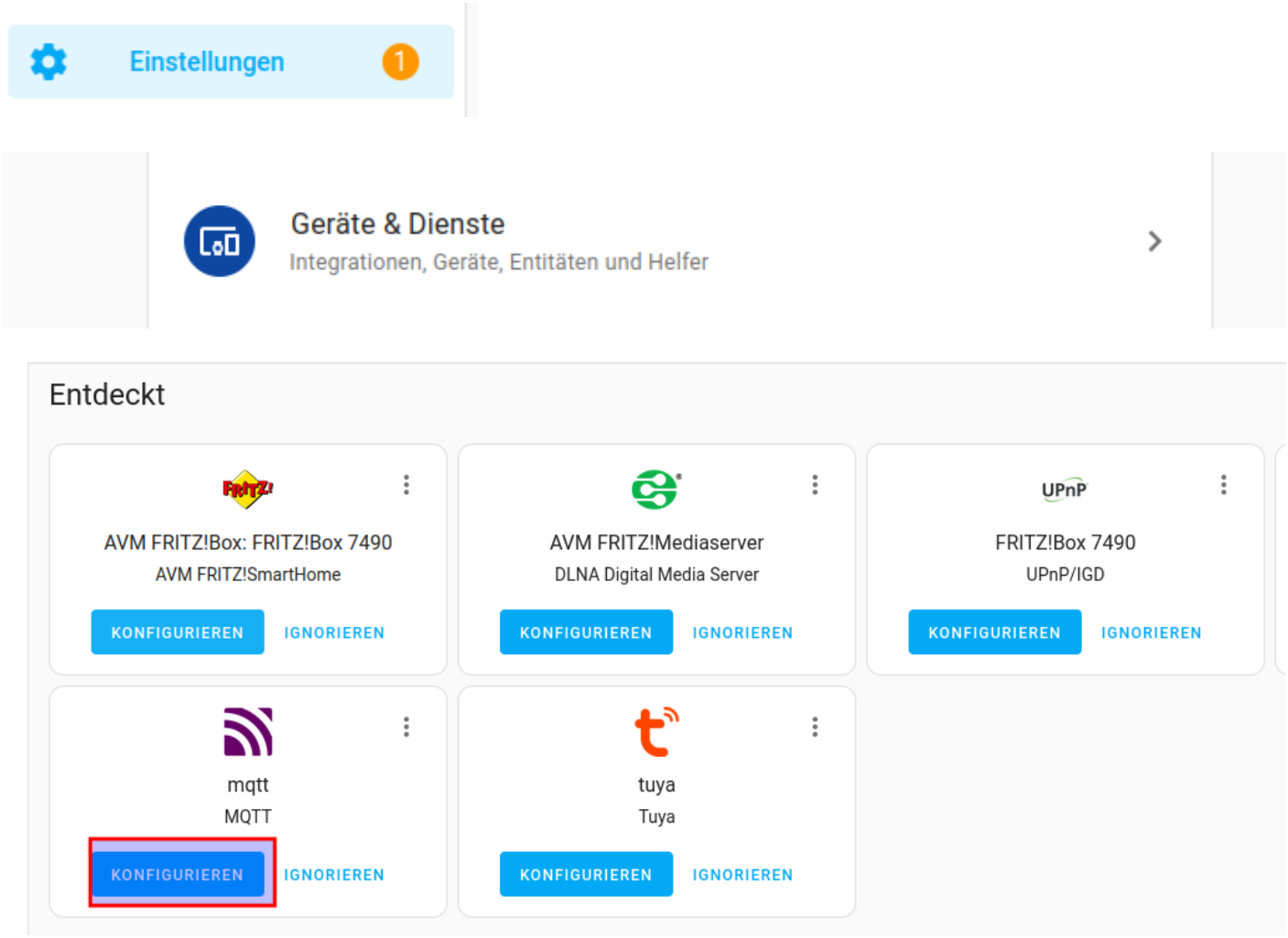
Dann installieren sagen



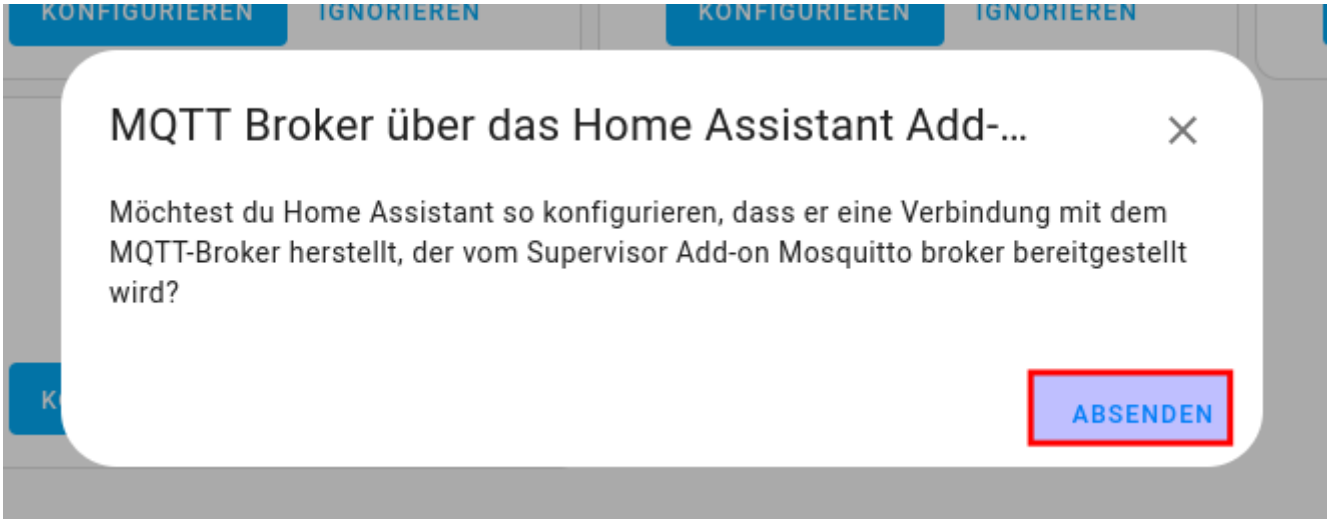
Nach der Installation auf starten klicken.
Der Broker selbt braucht keien Konfiguration.

Die MQTT Integration konfigurieren:

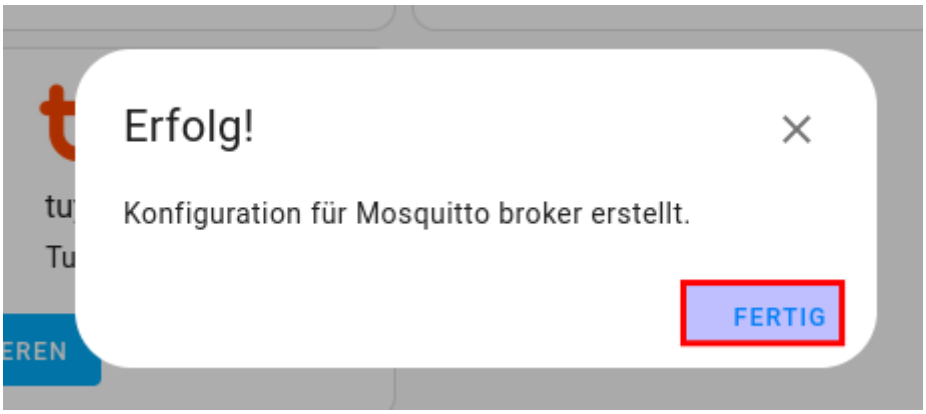
Dazu auf Einstellungen-> Geräte -> MQTT



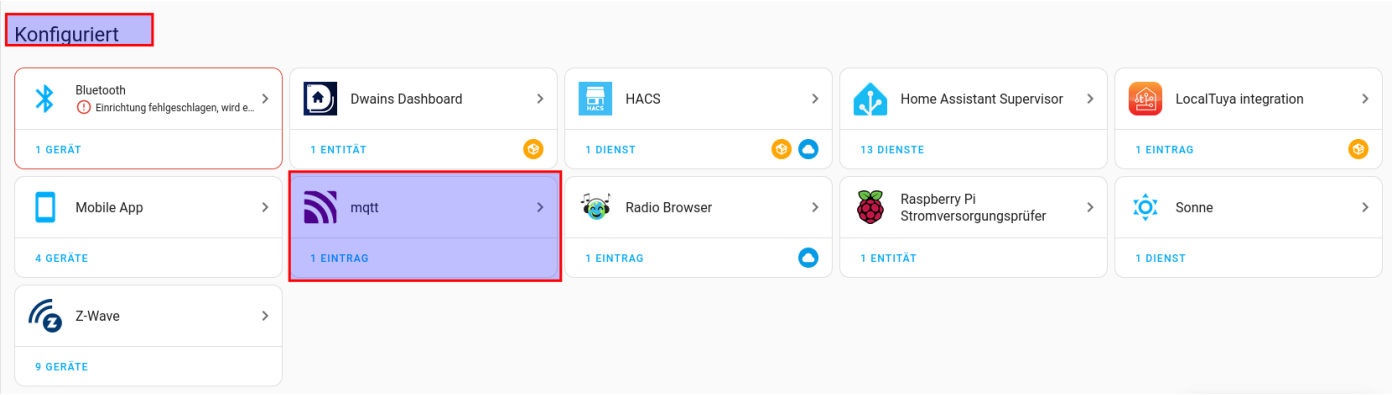
Nun auf Absenden klicken



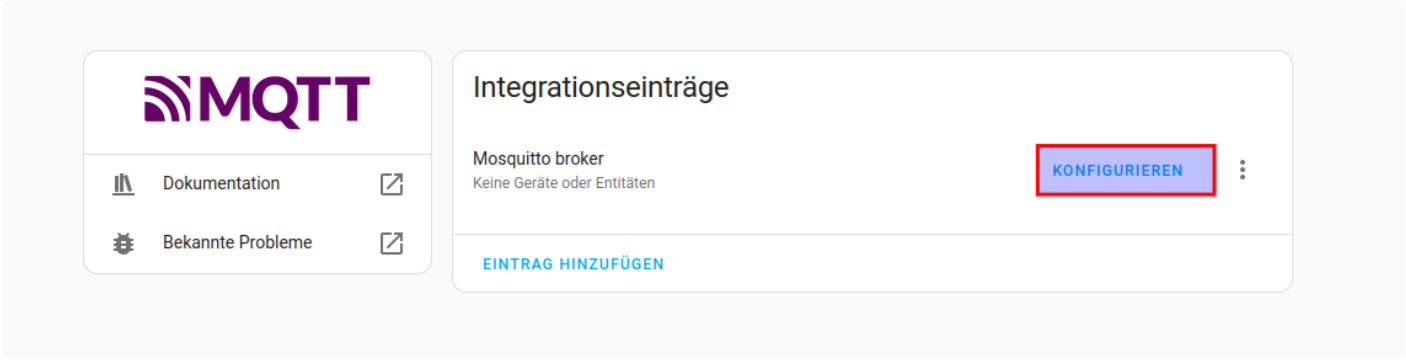
Dann auf fertig klciken



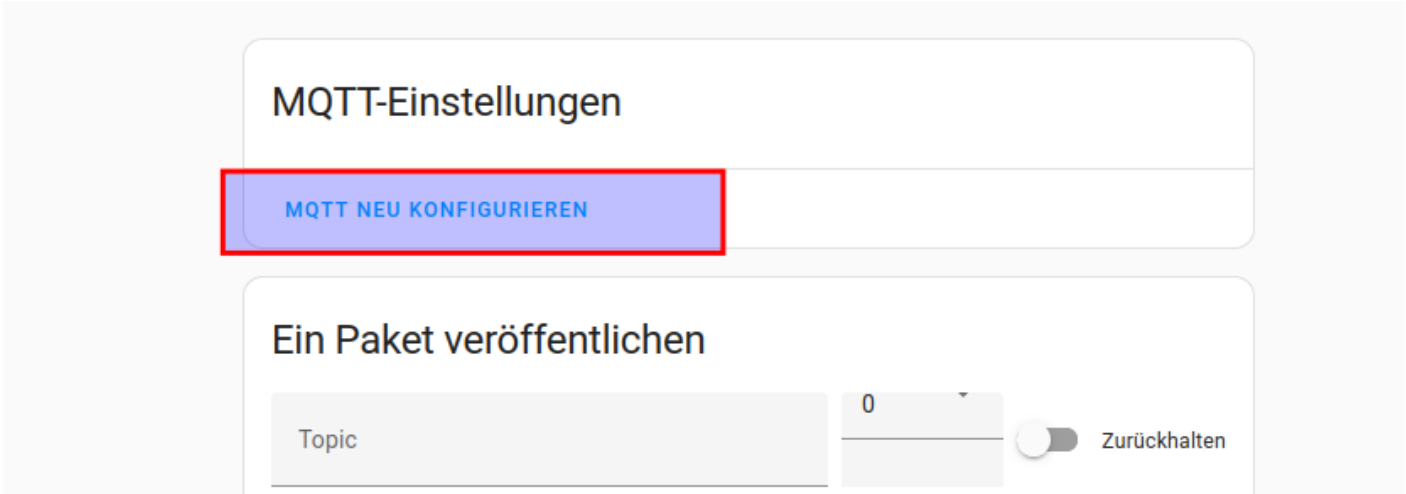
Nun sehen wir das MQTT im Bereich konfiguriert steht, und klciken diesen an



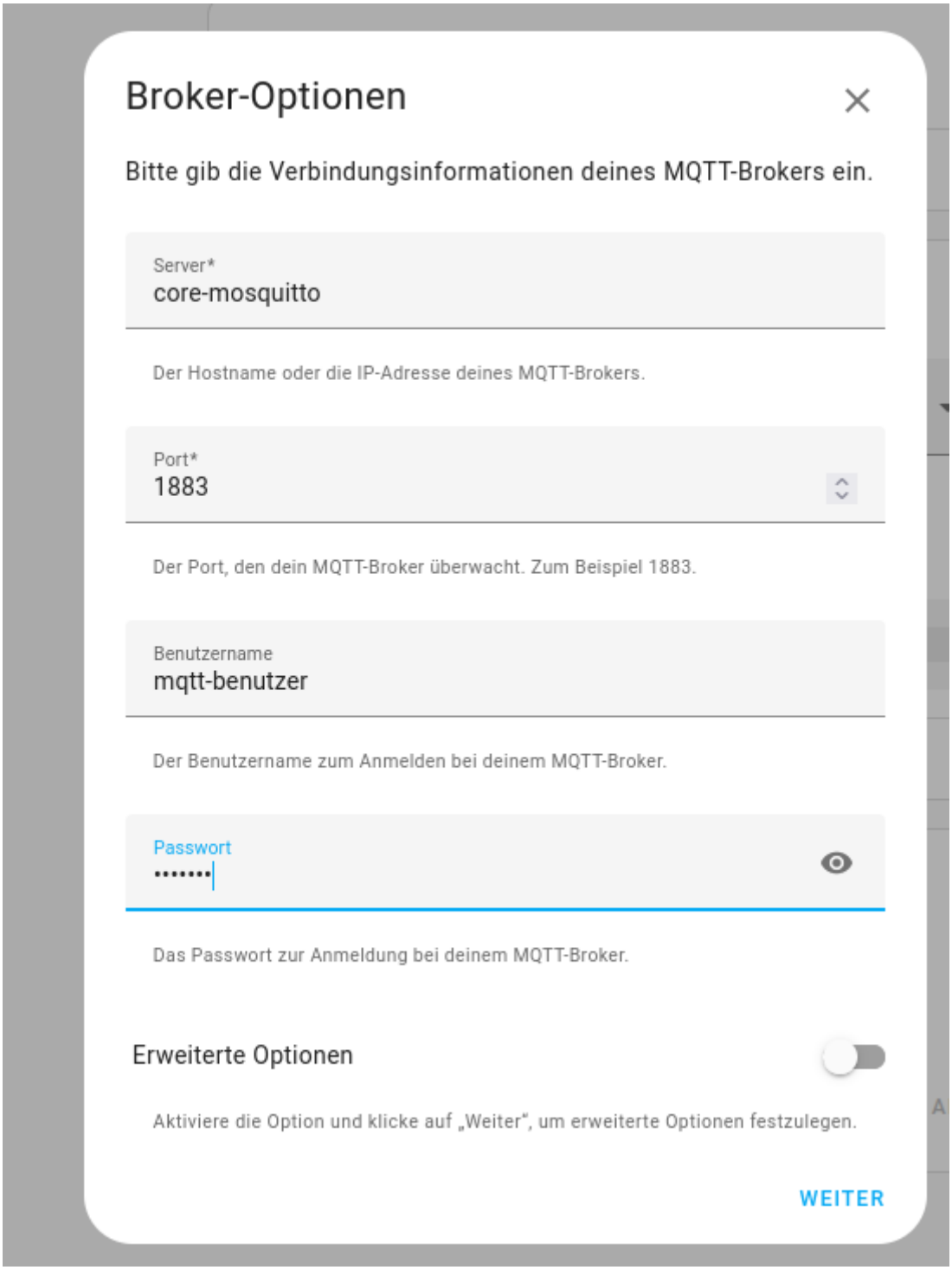
Nun auf konfigurieren klicken



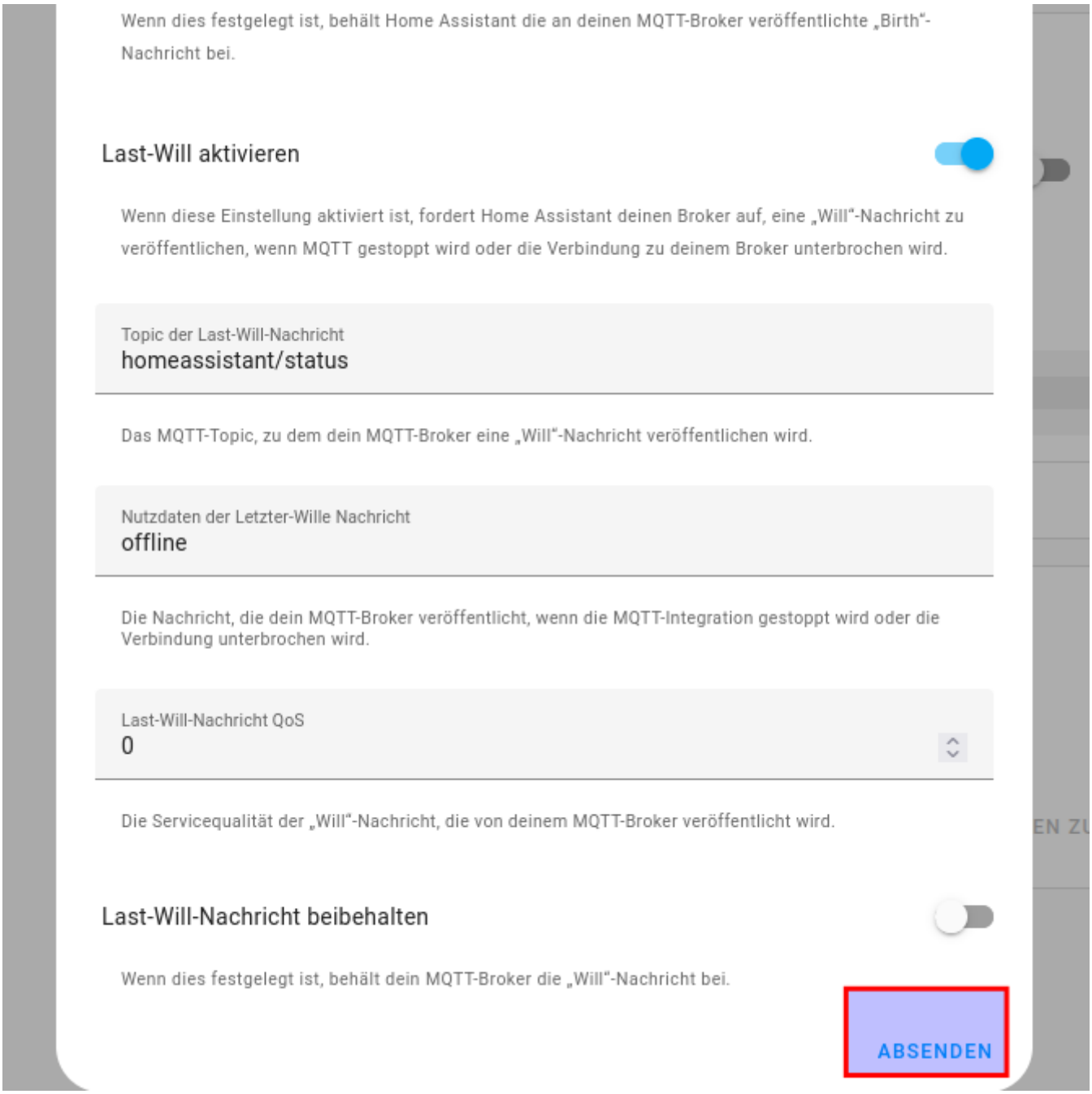
Nun auf erneut konfigurieren klciken



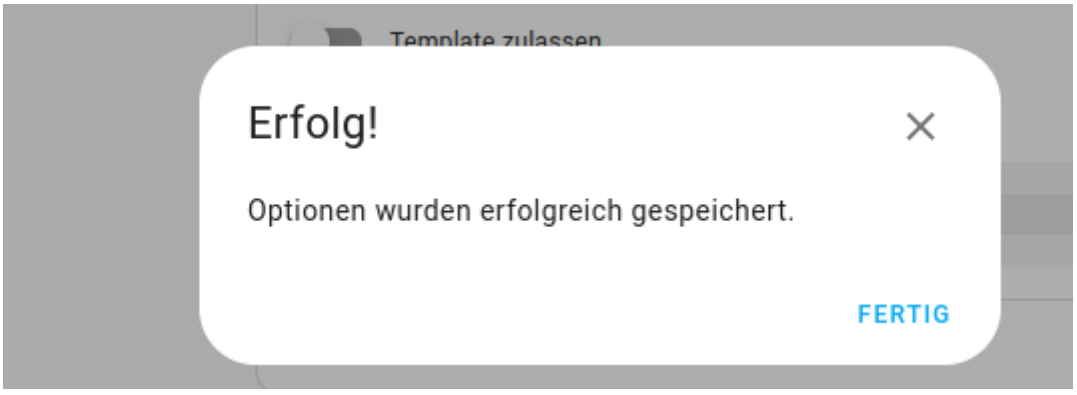
Und die Benutzerdaten mit unseren neuen austauschen
Nur Benutzername und Kennwort, alles andere so lassen



Nun hier auch alles so lassen und auf absenden klciken



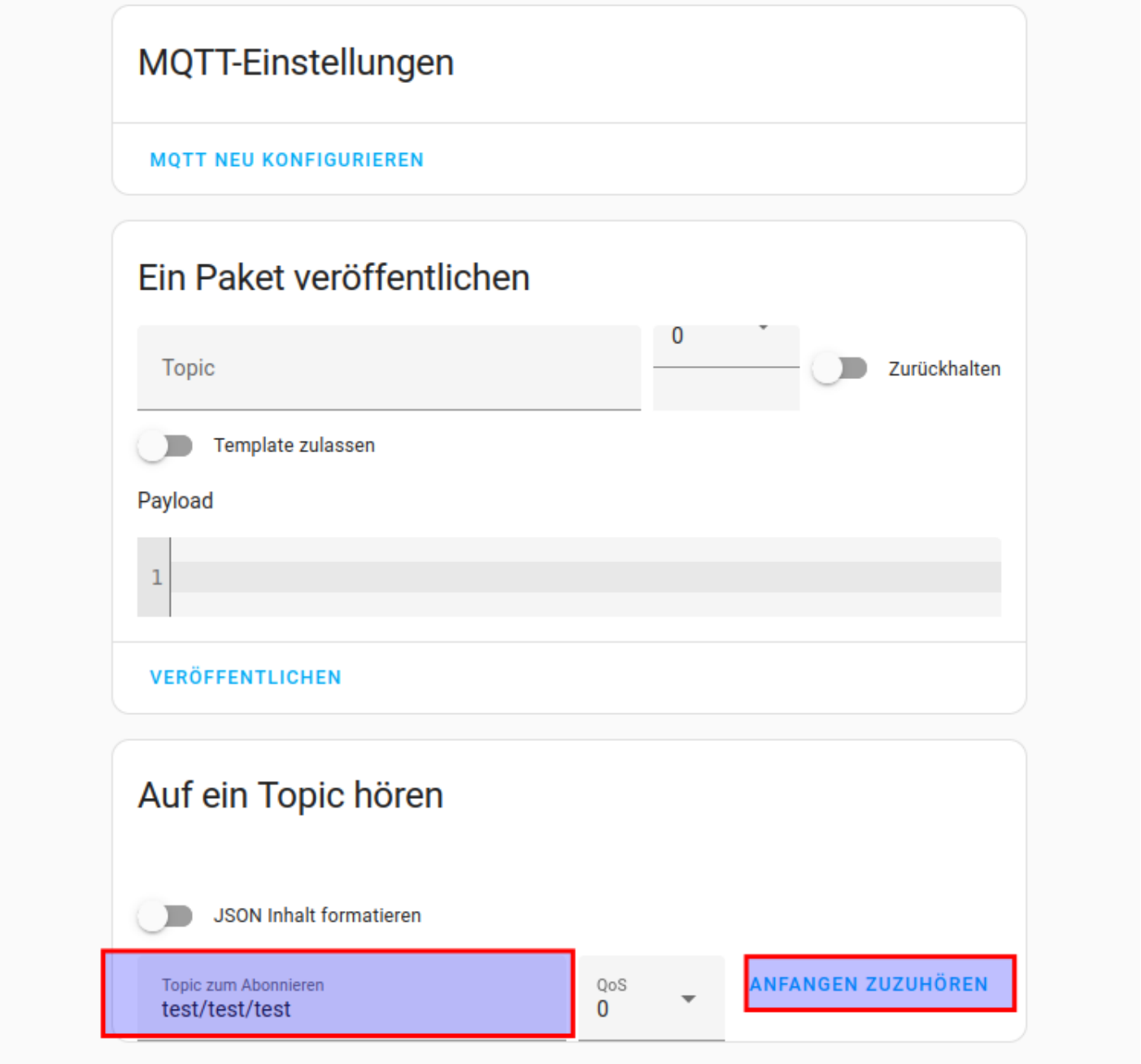
Nun auf fertig klicken



Testen des Brokers:

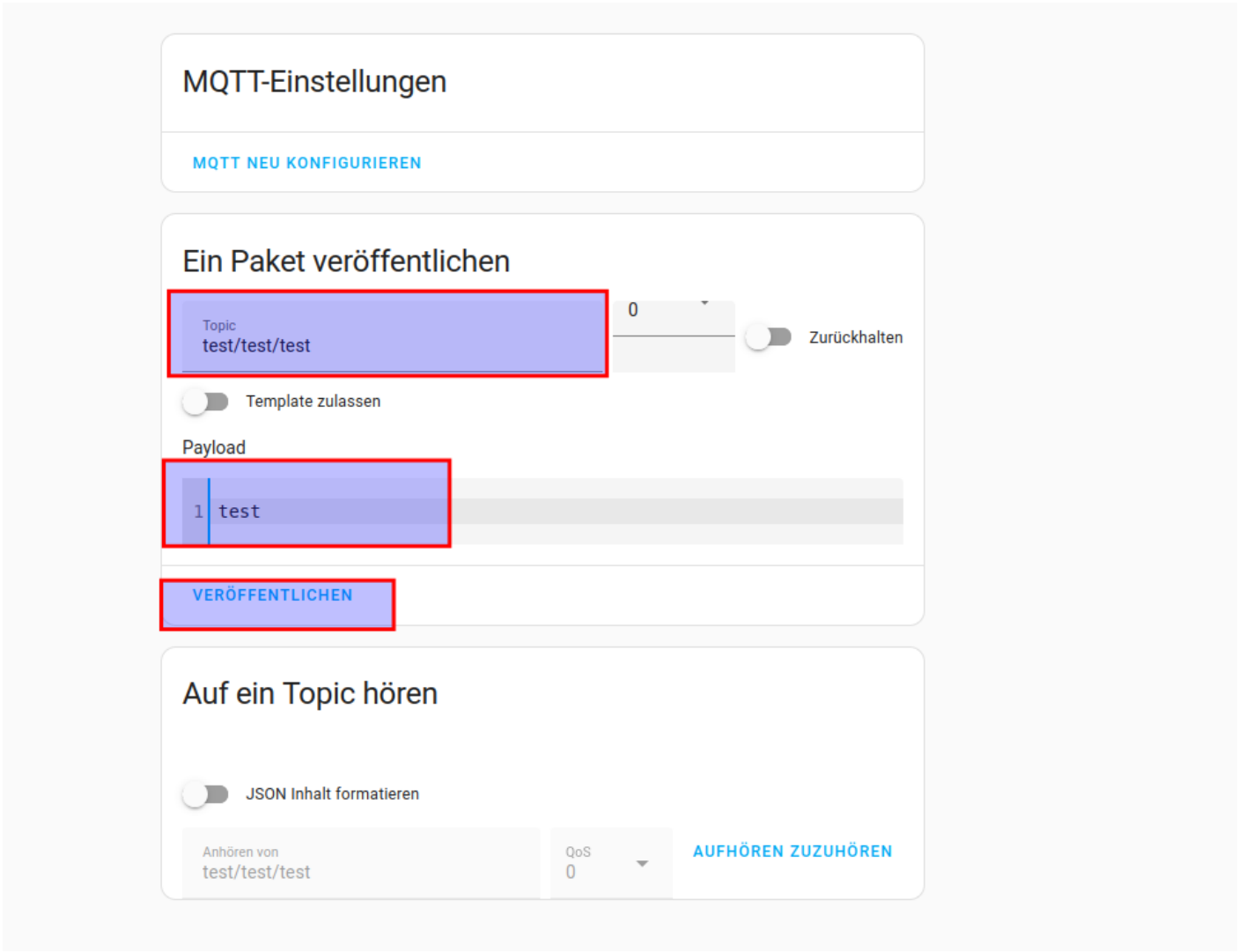
Bei Topüic zum Abonnieren

test/test/test eingeben und auf anfangen zuzuhören klicken

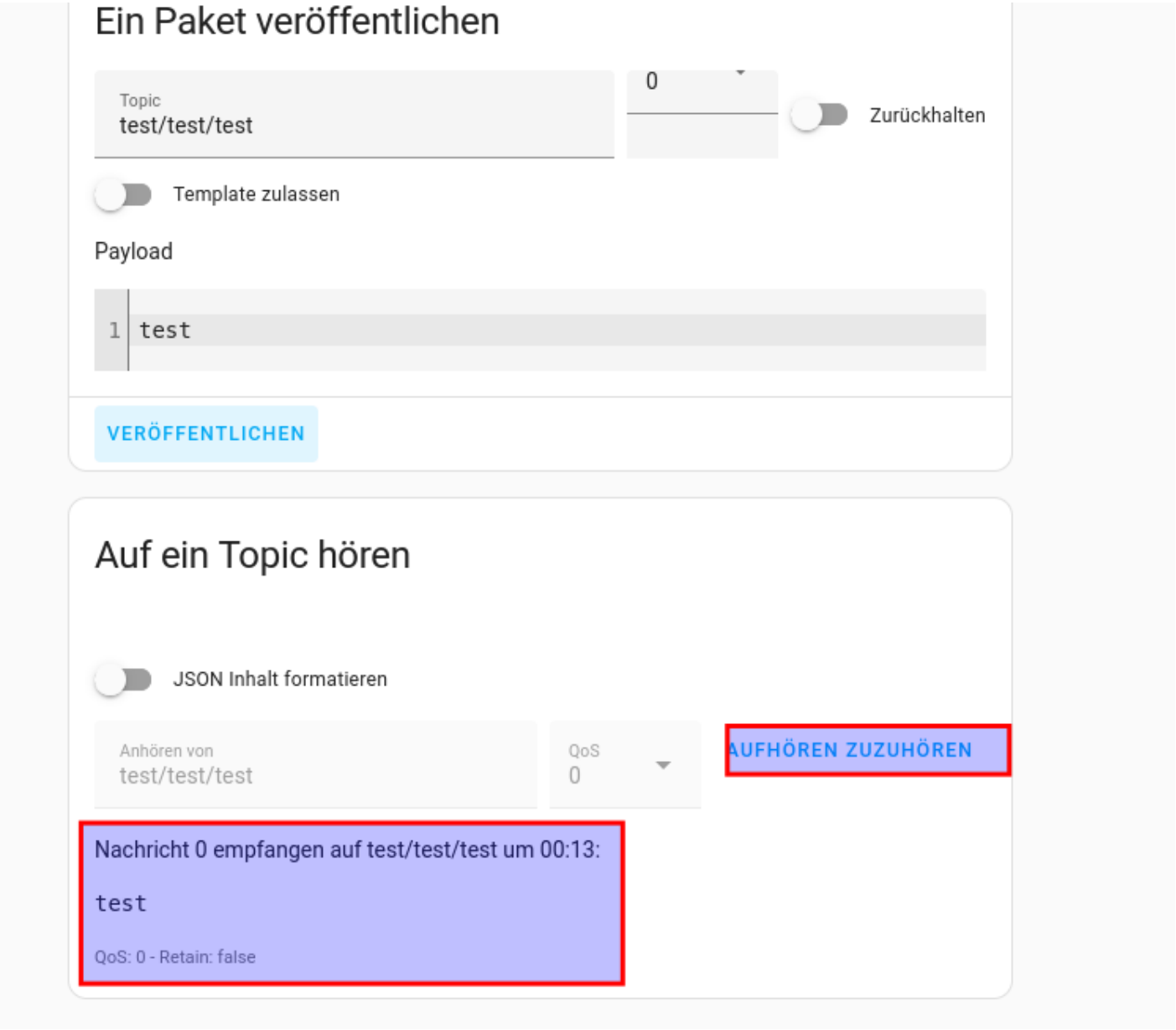


Num im payload topic auch test/test/test eingeben

Und als payload den Text test angeben und auf veröffentlichen klicken



Nun sehen wir unsere Nachricht ist angekommen und können auf , aufhören zuzuhören klicken. Der MQTT Broker funktioniert



Mit dieser Funktion kann man auch nachher eigene Topics testen ob sie denn ankommen. Zum debuggen ideal.

Python MQTT Scripte

Beschreibung:

Python Scripte für MQTT

Scripte:

proc2mqtt_linux - Ein Script was prüft, läuft ein Process oder nicht:

Verzeichnis Struktur, achtet auf euren Usernamen

```
/home/stefan/proc2mqtt/
├── proc2mqtt.py
├── .env
├── start.sh
└── proc2mqtt.service
```

Das Script:

```
#!/usr/bin/env python3


import time
import psutil
import paho.mqtt.client as mqtt
import argparse

# □□ Argumente einlesen
parser = argparse.ArgumentParser(description="Monitor a process and publish MQTT on start/stop.")
parser.add_argument("--broker", required=True, help="MQTT broker IP or hostname")
parser.add_argument("--port", type=int, default=1883, help="MQTT broker port (default: 1883)")
parser.add_argument("--topic", required=True, help="MQTT topic to publish to")
parser.add_argument("--process", required=True, help="Process name to monitor (e.g. cura)")
parser.add_argument("--interval", type=int, default=5, help="Check interval in seconds (default: 5)")
parser.add_argument("--onmsg", default="started", help="Message when process starts")
```

```
parser.add_argument("--offmsg", default="closed", help="Message when process stops")
parser.add_argument("--username", help="MQTT username (optional)")
parser.add_argument("--password", help="MQTT password (optional)")
args = parser.parse_args()

# □□ MQTT Client konfigurieren
client = mqtt.Client()
if args.username and args.password:
    client.username_pw_set(args.username, args.password)

client.connect(args.broker, args.port)
client.loop_start()

was_running = None

def is_proc_running(name):
    for proc in psutil.process_iter(['name']):
        try:
            if name.lower() in proc.info['name'].lower():
                return True
        except (psutil.NoSuchProcess, psutil.AccessDenied):
            continue
    return False

# □□ Hauptloop
while True:
    running = is_proc_running(args.process)
    if running != was_running:
        msg = args.onmsg if running else args.offmsg
        print(f"[MQTT] {args.process} is now {msg}")
        client.publish(args.topic, msg, qos=1, retain=True)
        time.sleep(0.5)  # optional
        was_running = running
    time.sleep(args.interval)
```

Aufruf, wenn man kein Systemd benutzen will, sonder einfach nur manuell starten möchte

```
python3 proc2mqtt.py   --broker 192.168.177.20 \
--port 1883 \
--username mqtt_user \
```

```
--password mqtt_password \
--topic proc/gnome-calculator \
--process gnome-calculator \
--onmsg started \
--offmsg closed
```

nun die .env Datei

```
MQTT_BROKER=192.168.177.20
MQTT_PORT=1883
MQTT_USERNAME=mqtt_username
MQTT_PASSWORD=mqtt_password
MQTT_TOPIC=proc/gnome-calculator
PROCESS_NAME=gnome-calculator
ONMSG=started
OFFMSG=closed
INTERVAL=5
```

Die start.sh

```
#!/bin/bash
set -a
source "$(dirname "$0")/.env"
set +a

exec python3 "$(dirname "$0")/proc2mqtt.py" \
    --broker "$MQTT_BROKER" \
    --port "$MQTT_PORT" \
    --username "$MQTT_USERNAME" \
    --password "$MQTT_PASSWORD" \
    --topic "$MQTT_TOPIC" \
    --process "$PROCESS_NAME" \
    --onmsg "$ONMSG" \
    --offmsg "$OFFMSG" \
    --interval "$INTERVAL"
```

Rechte anpassen ausführbar machen die start.sh

```
chmod +x /home/stefan/proc2mqtt/start.sh
chmod 600 /home/stefan/proc2mqtt/.env
```

Nun die system.d Datei

```
sudo nano /etc/systemd/system/proc2mqtt.service
```

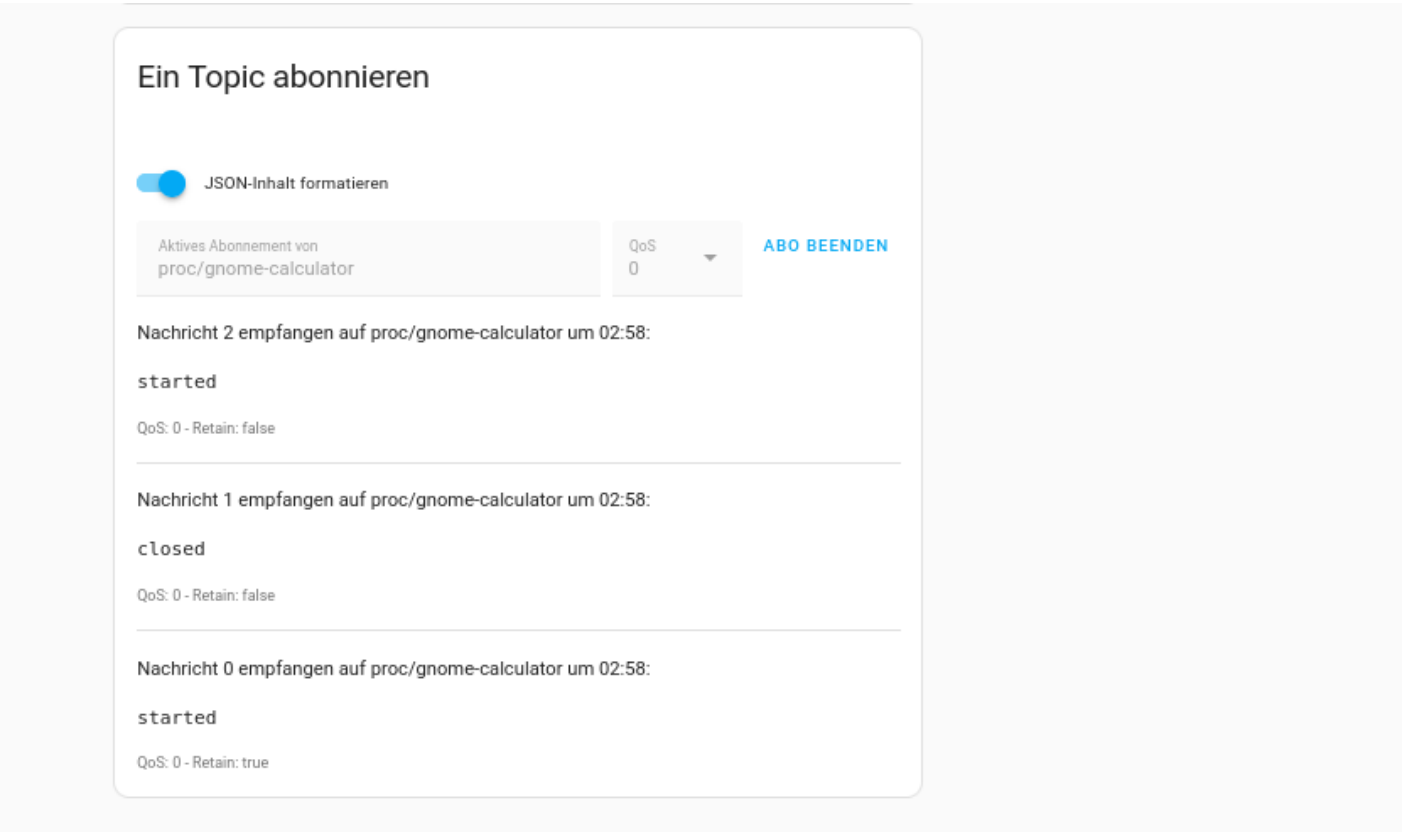
Inhalt, Achte auf den Username das ist der vom Linux Benutzer hier im Beispiel stefan, ja das script wird mit den rechten des Benutzers ausgeführt

```
# /etc/systemd/system/proc2mqtt.service
[Unit]
Description=MQTT Process Monitor
After=network.target

[Service]
ExecStart=/home/stefan/proc2mqtt/start.sh
WorkingDirectory=/home/stefan/proc2mqtt
Restart=always
User=stefan
EnvironmentFile=/home/stefan/proc2mqtt/.env

[Install]
WantedBy=multi-user.target
```

Nun den Dienst aktivieren

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable --now proc2mqtt.service
```

Und in der Home Assistant MQTT Integration



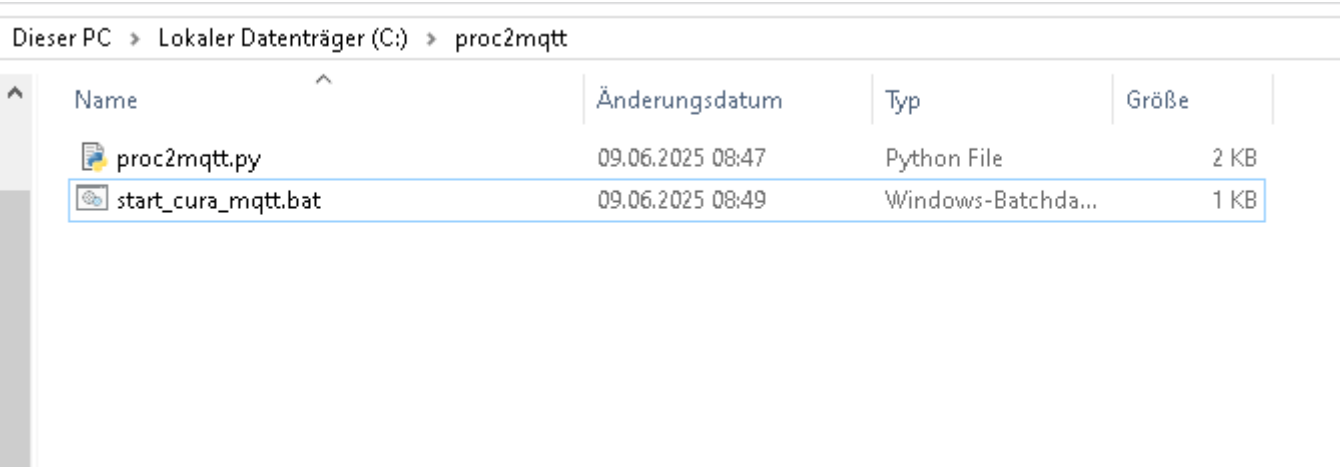
proc2mqtt_Windows - Ein Script was prüft, läuft ein Process oder nicht:

python Module installieren

Python installiern unter Windows siehe hier

```
pip install psutil paho-mqtt
```

Verzeichnis Struktur



Das Script

```
#!/usr/bin/env python3

import time
import psutil
import paho.mqtt.client as mqtt
import argparse

# □□ Argumente einlesen
parser = argparse.ArgumentParser(description="Monitor a process and publish MQTT on start/stop.")
parser.add_argument("--broker", required=True, help="MQTT broker IP or hostname")
parser.add_argument("--port", type=int, default=1883, help="MQTT broker port (default: 1883)")
parser.add_argument("--topic", required=True, help="MQTT topic to publish to")
parser.add_argument("--process", required=True, help="Process name to monitor (e.g. cura)")
parser.add_argument("--interval", type=int, default=5, help="Check interval in seconds (default: 5)")
parser.add_argument("--onmsg", default="started", help="Message when process starts")
parser.add_argument("--offmsg", default="closed", help="Message when process stops")
parser.add_argument("--username", help="MQTT username (optional)")
parser.add_argument("--password", help="MQTT password (optional)")
args = parser.parse_args()

# □□ MQTT Client konfigurieren
client = mqtt.Client()
if args.username and args.password:
    client.username_pw_set(args.username, args.password)

client.connect(args.broker, args.port)
client.loop_start()

was_running = None

def is_proc_running(name):
    for proc in psutil.process_iter(['name']):
        try:
            if name.lower() in proc.info['name'].lower():
                return True
        except (psutil.NoSuchProcess, psutil.AccessDenied):
            continue
    return False
```

```
# □□ Hauptloop
while True:
    running = is_proc_running(args.process)
    if running != was_running:
        msg = args.onmsg if running else args.offmsg
        print(f"[MQTT] {args.process} is now {msg}")
        client.publish(args.topic, msg, qos=1, retain=True)
        time.sleep(0.5)  # optional
        was_running = running
    time.sleep(args.interval)
```

Aufruf, wie in der Bat Datei

Die start bat, wichtig ist das bei der installation python zu den path variablen hinzugefügt wurde, ansonsten den kompletten pfad angeben. Den Prozessnamen immer .exe angeben
Nutze pythonw.exe, damit kein Konsolenfenster angezeigt wird! Nutze Python wenn du ein Konsolenfenster zum Debuggen haben willst

start_cura_mqtt.bat

```
@echo off
pythonw "C:\proc2mqtt\proc2mqtt.py" --broker 192.168.177.20 --port 1883 --username mqtt_username --password mqtt_password --topic proc/calc --process CalculatorApp.exe --onmsg started --offmsg closed
```

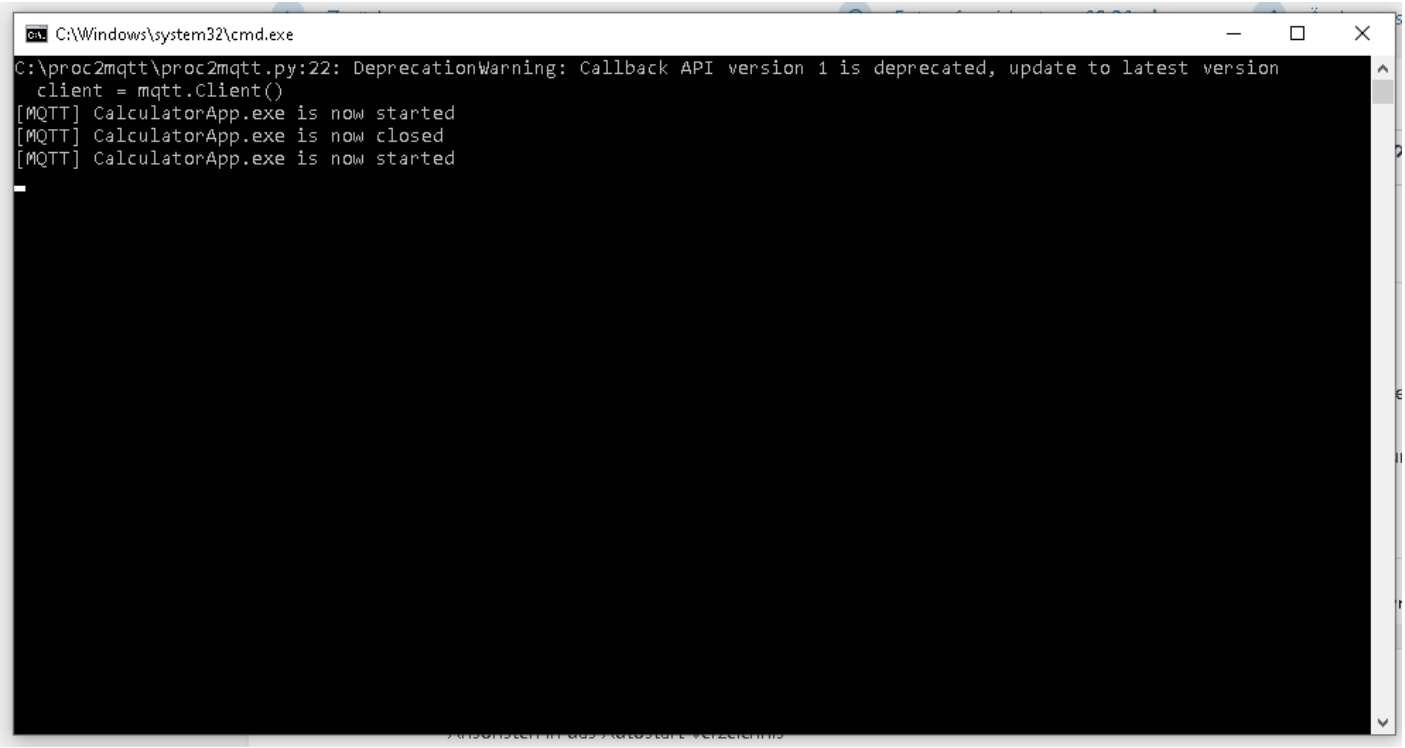
Nun per Taskplaner das ding in die Autostart, wenns Benutzerunabhängig prüfen soll.
Ansonsten in das Autostart Verzeichnis

Oben in die Adressezeile

```
shell:startup
```

eintippen und schon ist man im Autostartverzeichnis des Benutzers.

Dort die Bat rein. Fertig



Die Ausgabe in der MQTT Geräte integration