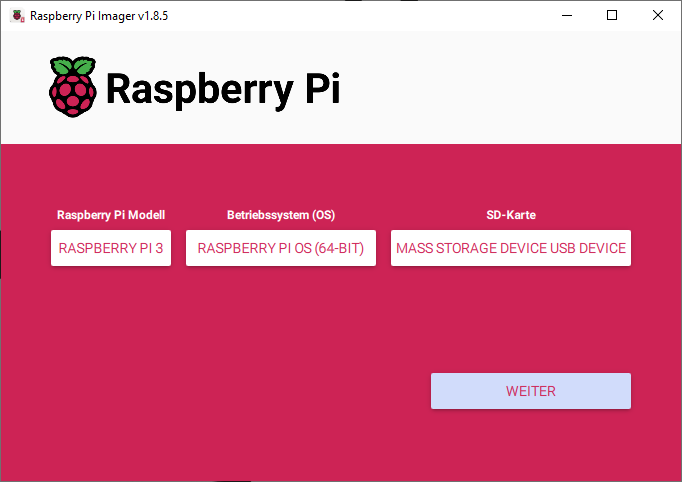
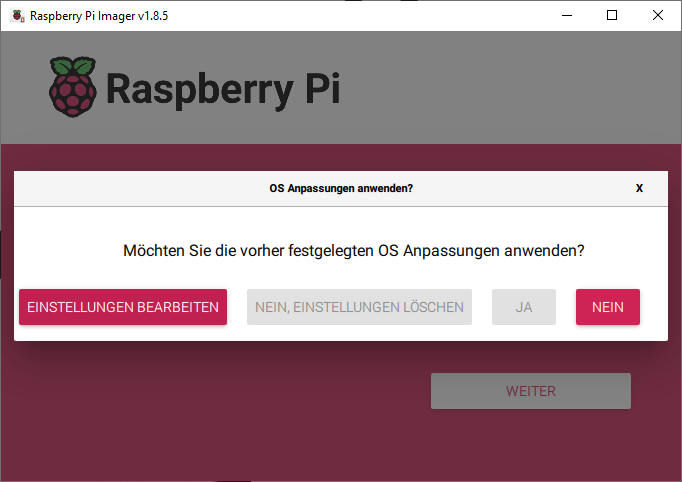
Voraussetzungen:

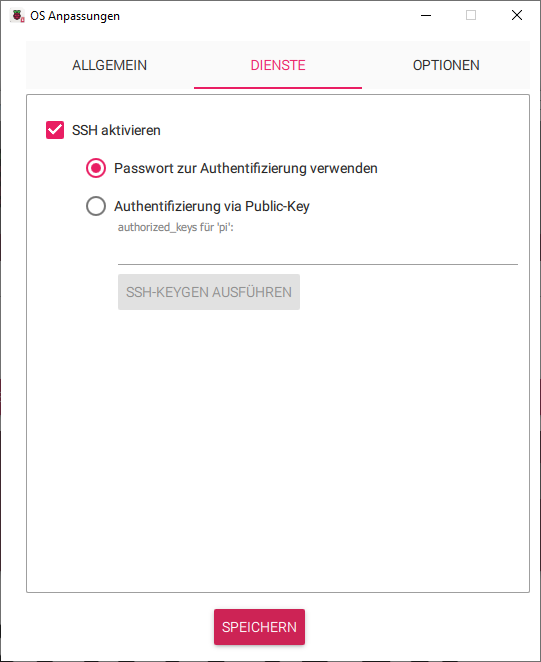
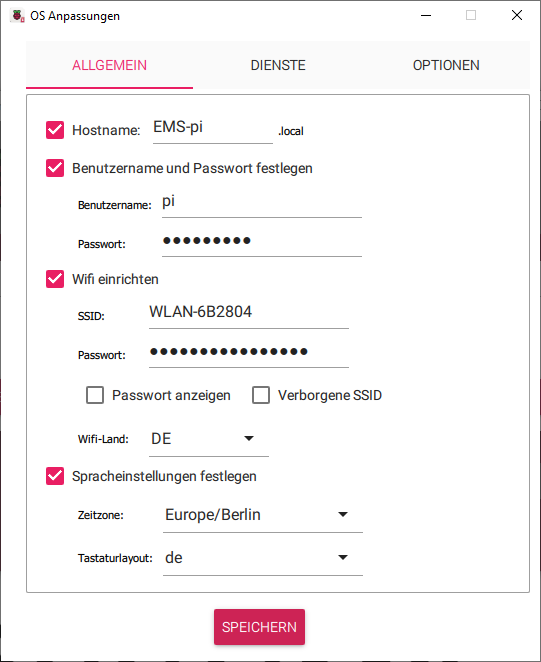
Zweiter PC (Linux, Mac oder Windows)  
 Die Screenshots wurden unter Windows erstellt. Daher können die Darstellungen und genutzten Tools auf anderen Betriebssystemen abweichen.  
Raspberry Pi (die Screenshots stammen von einem RPi 3B+)  
MircoSD-Karte min. 8 GB (besser 16 GB)  
ggf. LAN-Kabel  
Programm Raspberry Pi-Imager ( <https://www.raspberrypi.com/software/> )  
Wavestar USB-to-RS485 Stick

Anmerkungen:  
Alle Eingaben am PC oder RPi sind in „Courier“ und kursiv geschrieben.  
Die Anleitung ist geschrieben für den Hostname „ems-pi“ und als User wurde „pi“ gewählt. Solltet ihr etwas anderes gewählt haben, müsste ihr die entsprechenden Stellen natürlich anpassen.

Am PC das Programm Raspberry Pi Imager downloaden, installieren und starten.  
Micro-SD Karte in den PC einlegen.  
  
Dein RPi Modell auswählen.  
WICHTIG!! Als Betriebssystem: Raspberry Pi OS (Other) 🡪 Raspberry Pi OS Full (64-bit)  
SD-Karte auswählen und weiter drücken



Im folgenden Dialog können die Einstellungen OS direkt bei der Installation eingestellt werden:

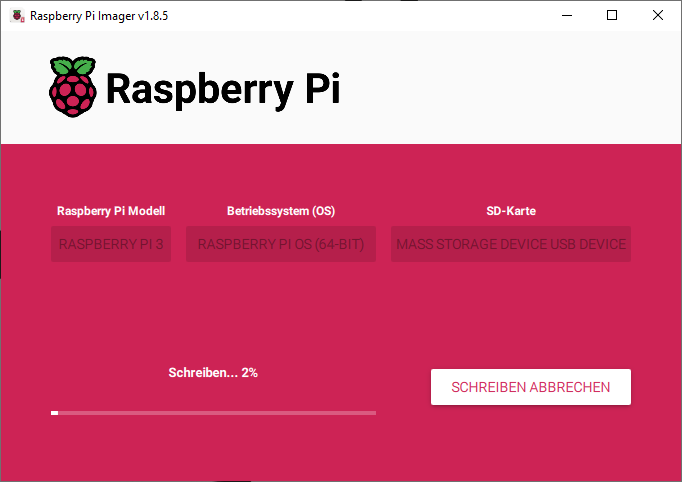
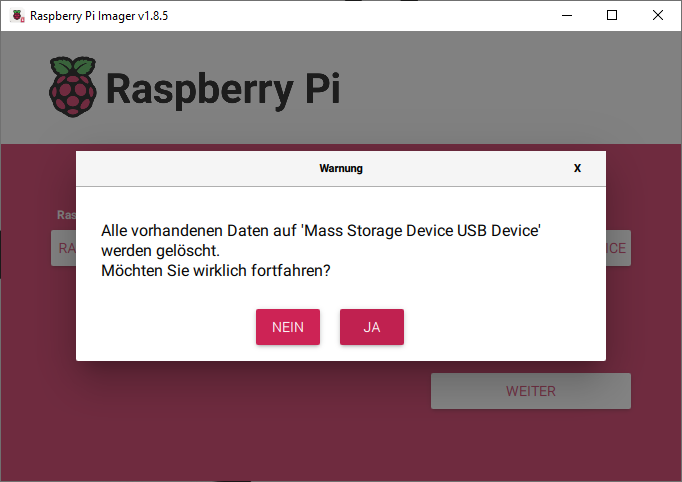




Erste Reiter „Allgemein“  
Hostname: so findet ihr den RPi im Netzwerk und könnt euch ohne die IP zu kennen mit ihm verbinden.  
Benutzername und Passwort: wird zur Verbindung mit dem RPi benötigt.  
Wifi: hier eure Zugangsdaten des WLAN eintragen (falls ihr WLAN nutzen möchtet. Bei LAN kann das Häckchen raus)

Zweiter Reiter „Dienste“  
SSH aktivieren 🡪 Passwort zur Authentifizierung verwenden (das ist das Passwort vorhin!)  
diese Einstellung wird benötigt um sich Remote über ssh auf dem RPi zu verbinden.

Zum Abschluss auf speichern drücken.  
Nun ist im Dialog von vorhin („OS Anpassungen verwenden?“) „JA“ anklickbar.

Nun nochmal mit „JA“ die Warnung bestätigen, dass die SD-Karte mit allen Daten gelöscht wird.  
  
  
Nun heißt es warten bis die Daten heruntergeladen, auf die Karte geschrieben und Verifiziert wurden.  
Nach Abschluss bis Karte in den RPi einlegen, Wavestar USB-to-RS485 Stick einstecken und den RPi starten.  
  
Bei Windows in der Zwischenzeit die „Eingabeaufforderung“ öffnen (Windows-Taste + R 🡪 im Dialog dann einfach „cmd“ eingeben und mit Enter bestätigen)  
In der Zwischenzeit sollte der RPi sich mit dem Netzwerk verbunden haben. Kontrollieren könnt ihr das in der Eingabeaufforderung mit   
*ping ems-pi*Im optimalen Fall erhaltet ihr Antwort von ems-pi inkl. Der IP-Adresse eures Pi. Nun geht es richtig los:

*ssh pi@ems-pi*  
die nun folgende Meldung (erscheint nur beim ersten Verbindungsversuch) bestätigen.  
*yes* 🡨 muss ausgeschrieben sein! Ein einfaches „y“ reicht hier nicht.  
Passwort eingeben. (nicht wundern. Es werden keine Sternchen o.ä. bei der Eingabe erscheinen.)  
nun seid ihr über ssh mit dem RPi verbunden.  
  
*sudo apt-get update*  
*sudo apt-get upgrade* („do you want to continue“ 🡪 „y“ hier reicht ein einfaches „y“ aus)  
*sudo apt-get install python3  
sudo apt-get install python3-serial  
sudo apt-get install python3-paho-mqtt  
pip3 install paho-mqtt --break-system-packages*pip3 install pyserial --break-system-packages *sudo raspi-config* im erscheinenden Programm „Interface“ > „Serial Port“ > „No“ > „Yes“  
  
jetzt ist reboot angebracht:  
sudo reboot  
  
die ssh-Verbindung wird getrennt. Nach dem Reboot (ca. 2-3 Min) wieder mit dem RPi verbinden:  
ssh pi@ems-pi

Jetzt kann es weitergehen:  
*mkdir ems\_mqtt  
cd ems\_mqtt*  
  
wget -O ems\_rs485\_to\_mqtt.py <https://raw.githubusercontent.com/SuNzZeR/EMS_RS485_to_MQTT/7c6fc4c3257a13929e1fccf787c147d9efe8c88d/ems_rs485_to_mqtt.py>  
**Achtung! Das ist eine Zeile! Der Zeilenumbruch hinter „ems\_rs485\_to\_mqtt.py“ kommt von Word!**

Nun ist das Programm auf euren RPi in den Ordner */home/pi/ems\_mqtt* heruntergeladen.

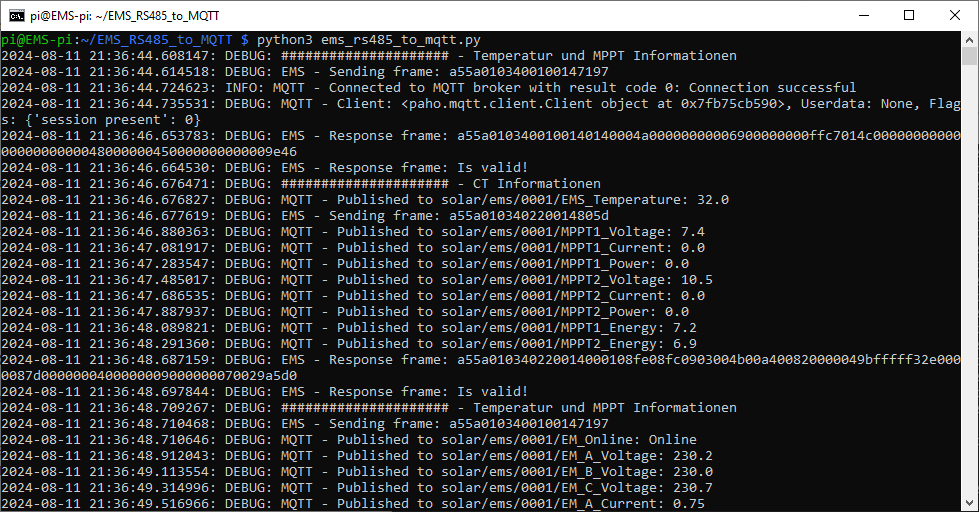
*sudo ls -l /dev/ttyUSB\** oder, wenn ttyUSB „no such file“ ausgibt: *sudo ls -l /dev/ttyACM\**  
  
den gefundenen Port merken. (hier: */dev/ttyUSB0 )*

*sudo nano ems\_rs485\_to\_mqtt.py*

Im Editor nun die Parameter anpassen:  
*LOG\_LEVEL = logging.INFO* -> ändern auf *LOG\_LEVEL = logging.DEBUG*(wird später wieder auf INFO geändert. Wir brauchen aber erstmal DEBUG, damit wir sehen können, ob alles funktioniert)  
  
LOG\_FILE = "/home/pi/ems\_mqtt/ems\_logfile.log"  
  
RS485\_PORT = "/dev/ttyUSB0" (oder eben /dev/ttyACM0 )

MQTT\_BROKER = "192.168.178.123" IP eures mqtt-brokers (bei mir gleich mit der HA-IP)  
MQTT\_PORT = 1883  
MQTT\_USERNAME = "mqtt"  
MQTT\_PASSWORD = "12345"

Nun den Editor mit STRG+X beenden. Speichern nicht vergessen! „Save modified buffer?“ -> y

python3 ems\_rs485\_to\_mqtt.py  
wenn alles richtig gelaufen ist, sollte nun euer Bildschirm mit Daten aus dem EMS geflutet werden:  
Auf dem USB-Stick sollten neben der leuchtenden Power-LED nun auch die TX-LED und, viel wichtiger, die RX-LED blinken  
  
Script nun mit STRG+C wieder beenden.  
  
nochmal den Editor öffnen  
*sudo nano ems\_rs485\_to\_mqtt.py*und dort das LOG\_LEVEL wieder von DEBUG auf INFO ändern, damit euer Logfile nicht überläuft. Editor mit STRG+C schließen und mit y speichern nicht vergessen.  
  
Jetzt wird das Programm noch als Service eingerichtet und beim booten automatisch gestartet:

**Service einrichten**

Erstelle eine neue Service-Datei für systemd

sudo nano /etc/systemd/system/ems\_mqtt.service

Füge folgenden Inhalt ein (bitte Pfade und Username ggf. anpassen)

[Unit]  
Description=RS485 to MQTT Service for EMS  
After=network.target  
  
[Service]  
ExecStart=/usr/bin/python3 /home/pi/ems\_mqtt/ems\_rs485\_to\_mqtt.py  
WorkingDirectory=/home/pi/ems\_mqtt  
StandardOutput=inherit  
StandardError=inherit  
Restart=always  
User=pi  
  
[Install]  
WantedBy=multi-user.target

Aktiviere den neuen Service, sodass er beim Booten automatisch gestartet wird.

sudo systemctl enable ems\_mqtt.service

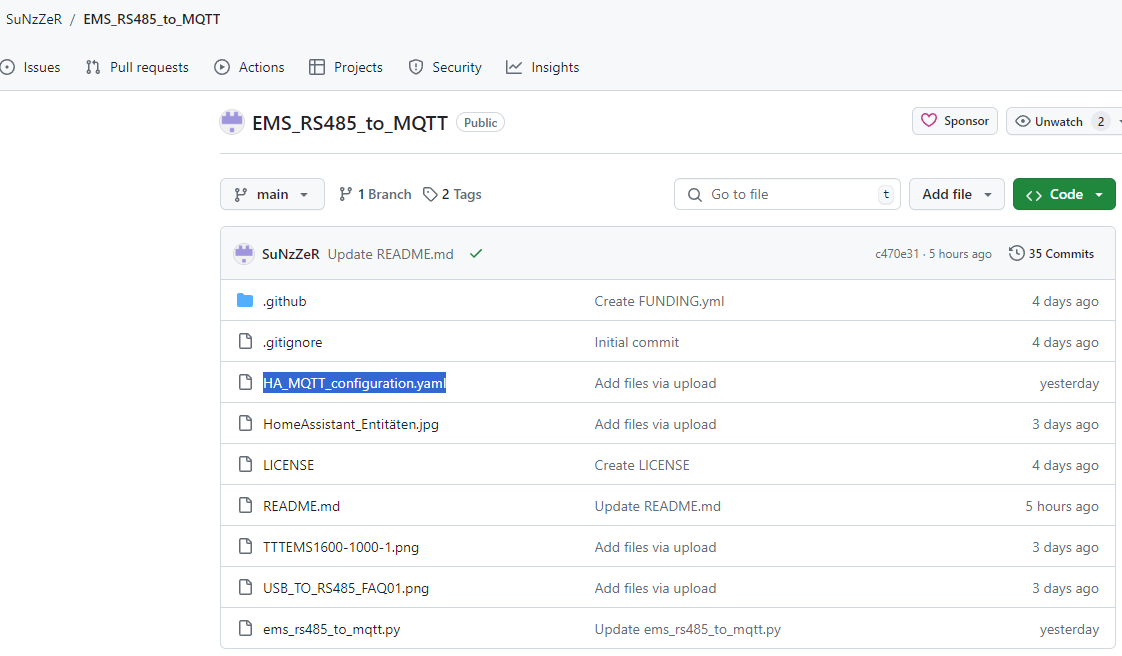
Starte den Service

sudo systemctl start ems\_mqtt.service

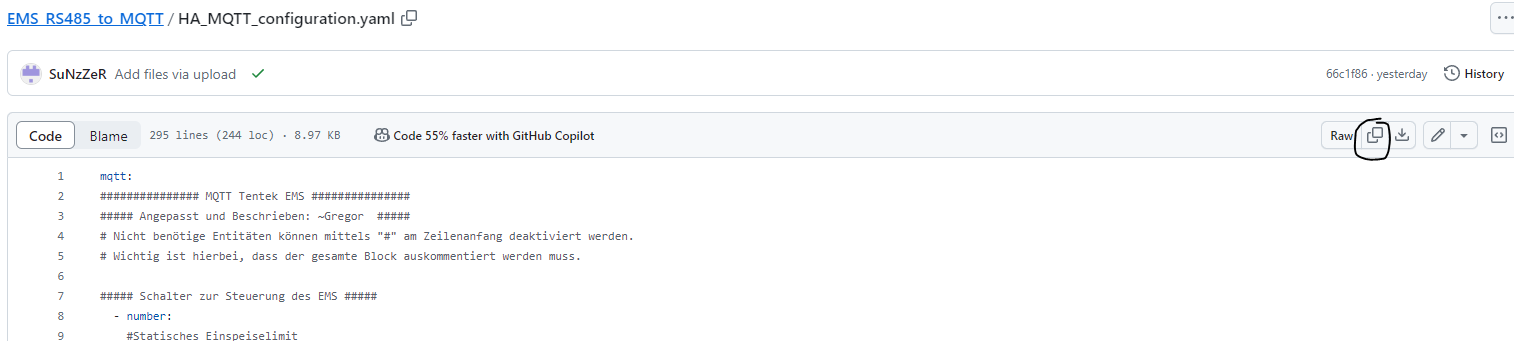
Herzlichen Glückwunsch!  
Du hast das Tool ems\_to\_mqtt erfolgreich eingerichtet.

Nun muss nur noch im HA deine configuration.yaml angepasst werden:

Im Github ist die Datei „HA\_MQTT\_configuration.yaml“



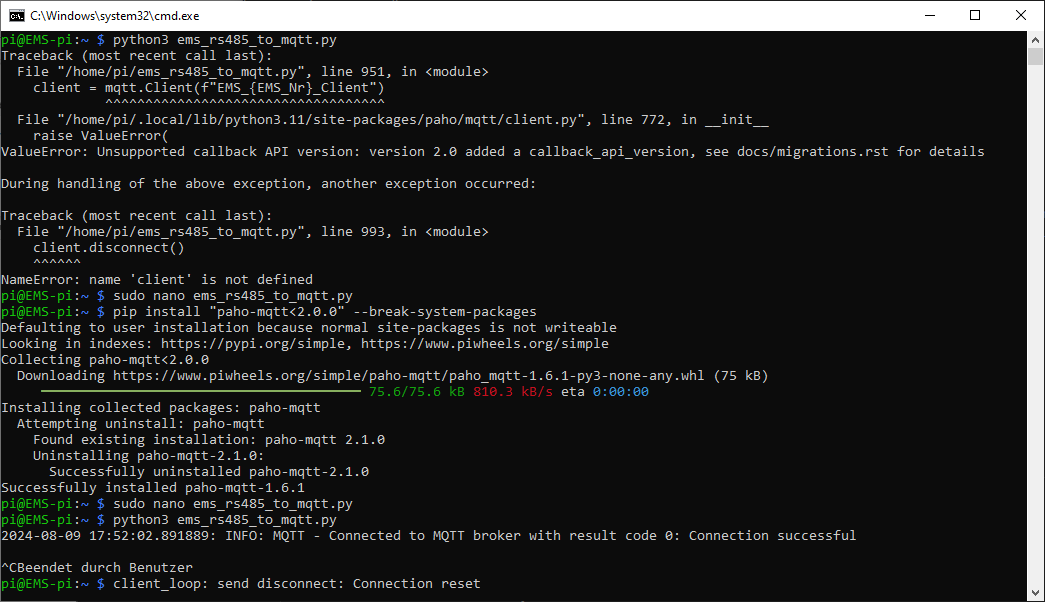
Diese durch Anklicken öffnen und oben rechts auf „copy raw file“ klicken.



Nun den Inhalt in deine Configuration.yaml einfügen.  
Falls du dort schon einen Block „mqtt“ haben solltest, bitte dort einfügen und die Zeile 1 („mqtt:“) entfernen.  
Ansonsten einfach am Ende einfügen, speichern.  
Dann „Entwicklerwerkzeuge“, „Konfiguration prüfen“ & bei fehlerfrei „neu starten“.

Nach dem Neustart findest Du im HA fast 50 neue Entitäten.

**Fehlerbilder:**  
paho-mqtt: Unsupported callback API version: version 2.0  
Lösung: pip install „paho-mqtt<2.0.0 –break-system-packages



Response frame ist leer:  
Lösung: Verkabelung vermutlich fehlerhaft. Prüfen.  
