

ScanShredd Pi Setup

[CC BY-SA 4.0 @tamberg](#), 23.08.2023

Für

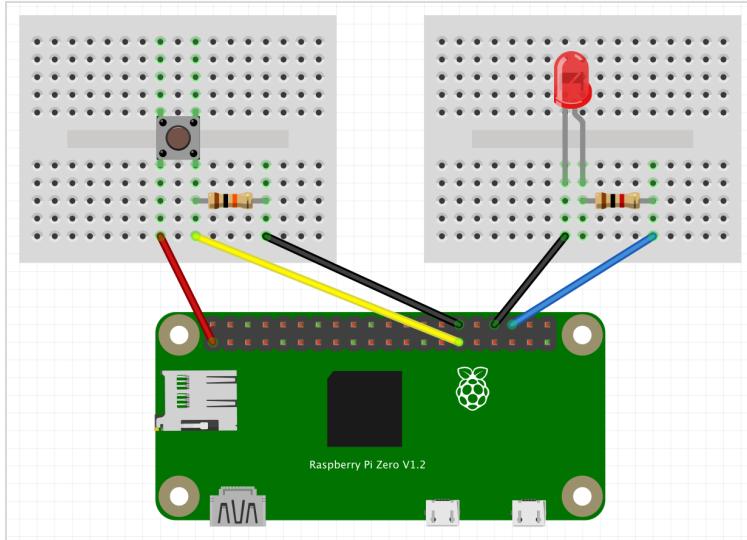
peter@auchli.com
<https://www.auchli.com>

Überblick

Überblick.....	1
Hardware vorbereiten.....	2
SD Card erstellen.....	2
Zugriff auf Pi via SSH.....	7
Kamera freischalten.....	9
Software einrichten.....	11
Node.js und npm installieren.....	11
Symlink auf raspistill einrichten.....	12
Programm photo.js installieren und testen.....	12
Programm switch.js installieren und testen.....	12
Programm upload.js installieren und testen.....	13
Programm scanshredd.js installieren und testen.....	13
Dienst einrichten.....	13
Dienst scanshredd.service einrichten und starten.....	13
Dienst scanshredd.service stoppen (optional).....	14
Dienst scanshredd.service löschen (optional).....	14
Troubleshooting.....	14

Hardware vorbereiten

- [Pi Zero WH](#) (ab jetzt Pi) mit [Pi Cam](#) via [Kabel](#) verbinden
- Sensor und Aktuator verbinden (siehe auch [Pi Pinout](#)):
Button (bzw. digitaler Input) an GPIO/BCM Pin 5 mit 10kΩ, 3.3V
LED (bzw. digitaler Output) an GPIO/BCM Pin 16 mit 1kΩ



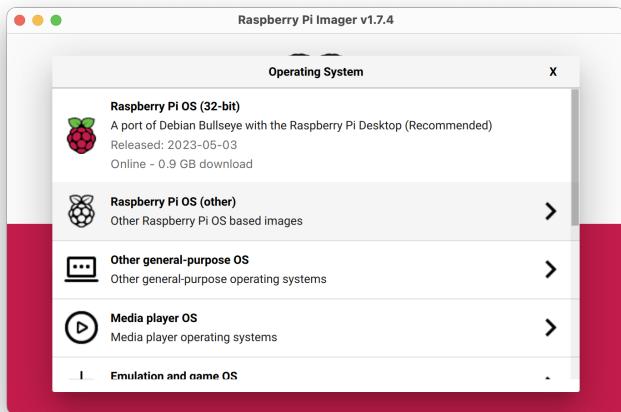
SD Card erstellen

Auf dem PC ([Mac](#) od. Win od. Linux):

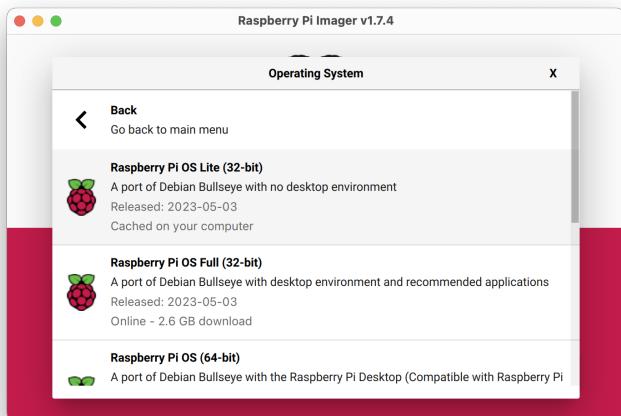
- <https://www.raspberrypi.com/software/> > *Raspberry Pi Imager* runterladen (einmal)
- *Raspberry Pi Imager* installieren (einmal)
- Micro SD Card einlegen (in PC)
- *Raspberry Pi Imager* starten
- Choose OS klicken:



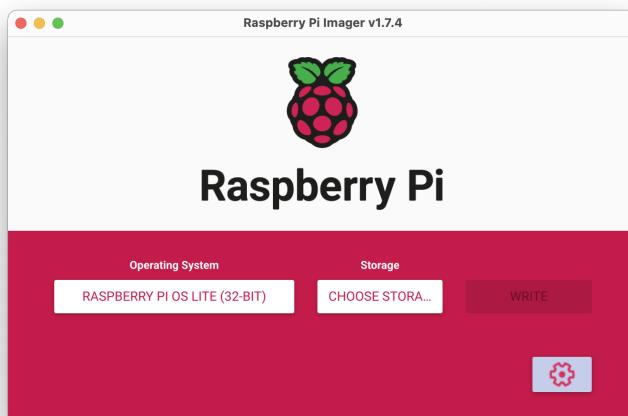
- *Raspberry Pi OS (other)* anwählen:



- *Raspberry Pi OS (lite)* anwählen:

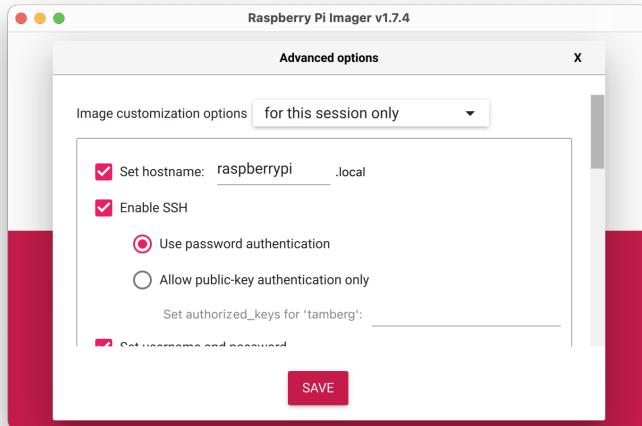


- Settings öffnen, bzw. Zahnrad Icon klicken:



- Settings ausfüllen:

*Set hostname: raspberrypi.local
 Enable SSH
 (x) Use password authentication*



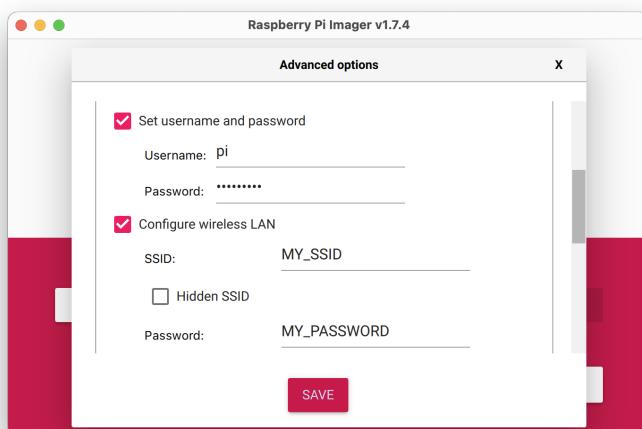
- Settings ausfüllen (Fortsetzung):

Set username and password

*Username: pi
 Password: raspberry*

Configure wireless LAN

*SSID: MY_SSID
 Password: MY_PASSWORD
 Wireless LAN country: GB (nicht CH)*



- Save klicken:



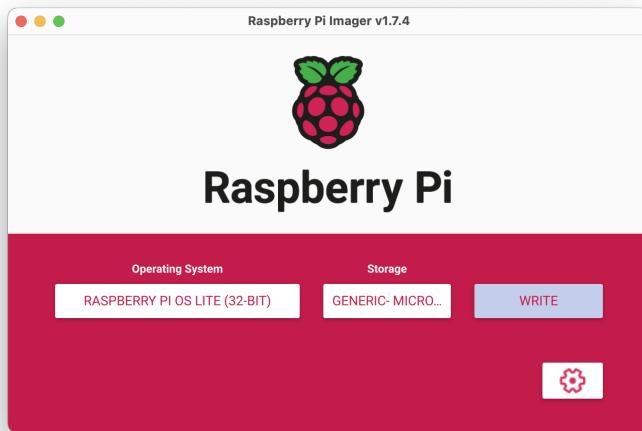
- Choose storage klicken:



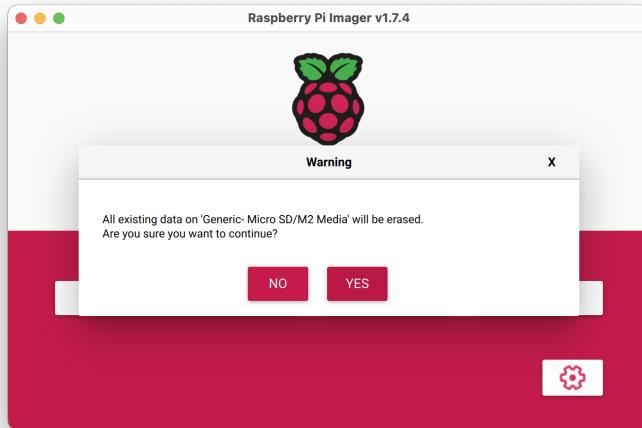
- SD Card anwählen:



- Write klicken:



- Warnung mit Yes bestätigen:



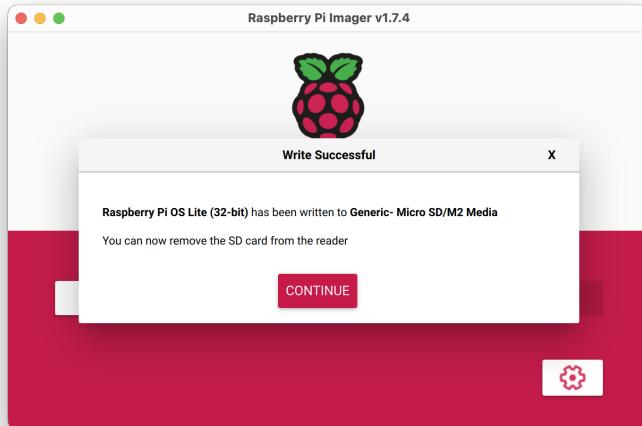
- OS Passwort eingeben:



- SD Card wird geschrieben:



- Resultat - SD Card ist fertig geschrieben und kann ausgeworfen werden.



Zugriff auf Pi via SSH

Voraussetzung: SD Card erstellen.

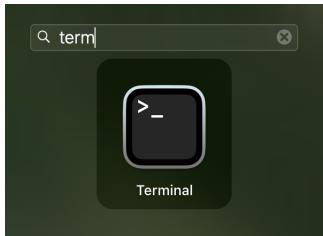
Auf dem Pi:

- SD Card einlegen
- Pi mittels Micro USB Kabel an PC (oder Strom-Adapter) anstecken

Auf dem PC (Mac oder Win oder Linux):

- PC mit demselben Wi-Fi Netzwerk verbinden wie oben, z.B.
SSID: *MY_SSID*
Passwort: *MY_PASSWORD*

- Terminal Anwendung öffnen:



- \$ ssh pi@raspberrypi.local eingeben, danach yes und dann das Passwort raspberry (Kommandos jeweils ohne \$-Zeichen, dafür alle Eingaben mit ENTER abschliessen):

```
tamberg — pi@raspberrypi: ~ — ssh pi@raspberrypi.local — 80x24
tamberg@mac ~ % ssh pi@raspberrypi.local
The authenticity of host 'raspberrypi.local (fe80::67fb:4154:725c:26e7%en0)' can
't be established.
ED25519 key fingerprint is SHA256:HP2Gq8GbAW83QkV7l+BZCz8sLh6vPXMFe+Jct+AMPg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi.local' (ED25519) to the list of known ho
sts.
pi@raspberrypi.local's password:
Linux raspberrypi 6.1.21+ #1642 Mon Apr  3 17:19:14 BST 2023 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

- \$ passwd um ein neues Passwort zu setzen:

```
tamberg — pi@raspberrypi: ~ — ssh pi@raspberrypi.local — 80x24
pi@raspberrypi:~ $ passwd
Changing password for pi.
Current password:
New password:
```

- Resultat - Zugriff auf Pi via SSH ist erstellt:

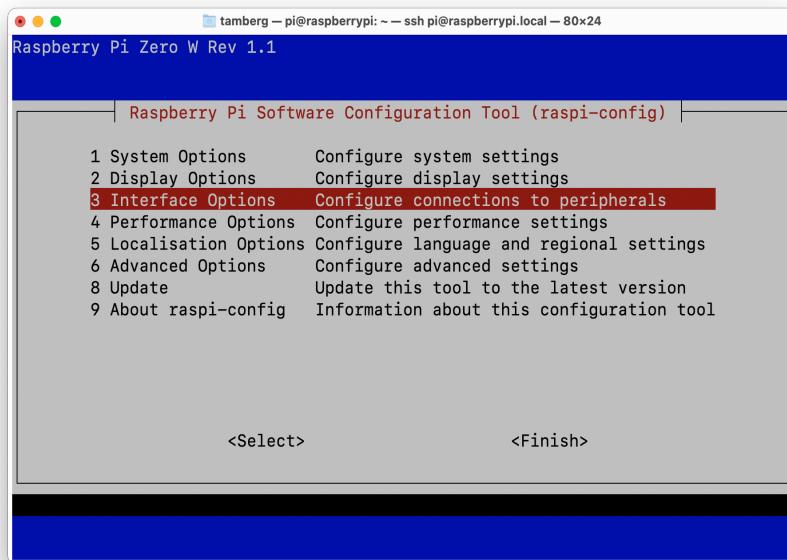
```
tamberg — pi@raspberrypi: ~ — ssh pi@raspberrypi.local — 80x24
pi@raspberrypi:~ $
```

Kamera freischalten

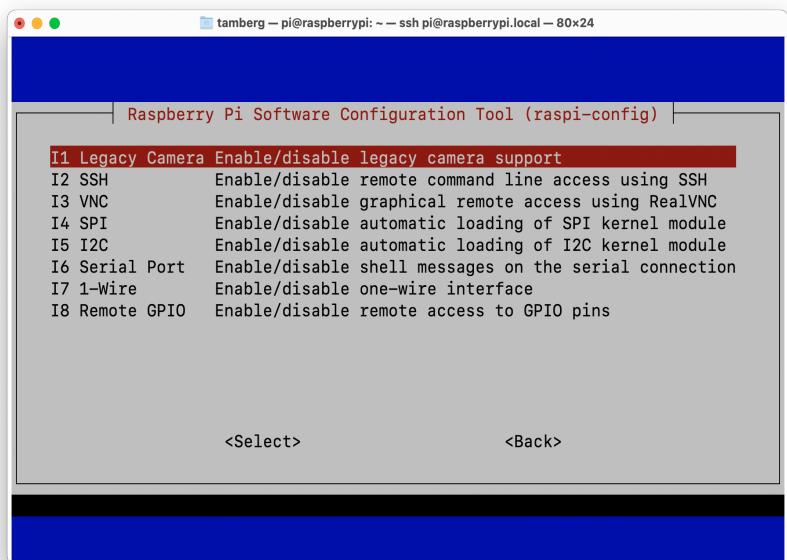
Voraussetzung: Zugriff auf Pi via SSH.

Auf dem Pi:

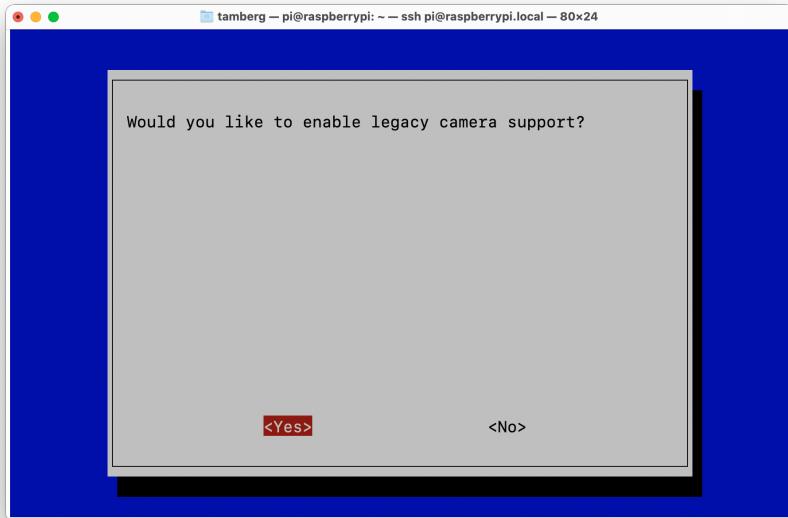
- `$ sudo raspi-config`
- *Interface Options* anwählen (danach jeweils *ENTER*):



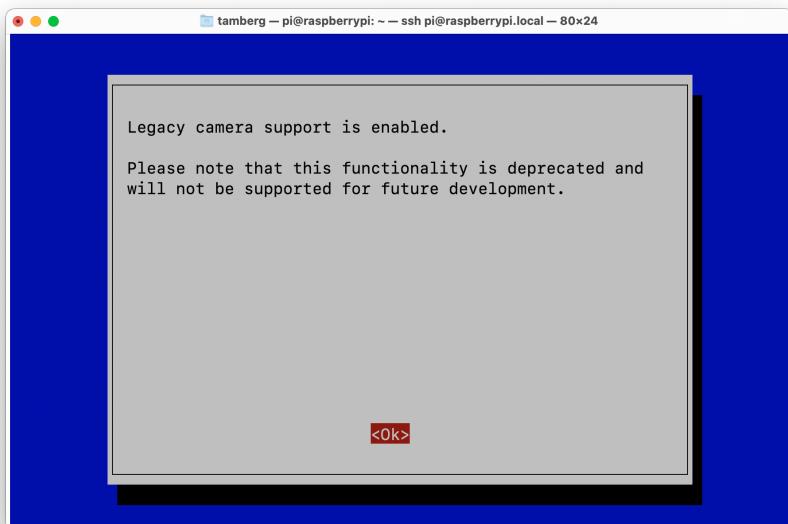
- *Legacy Camera Enable ...* anwählen:



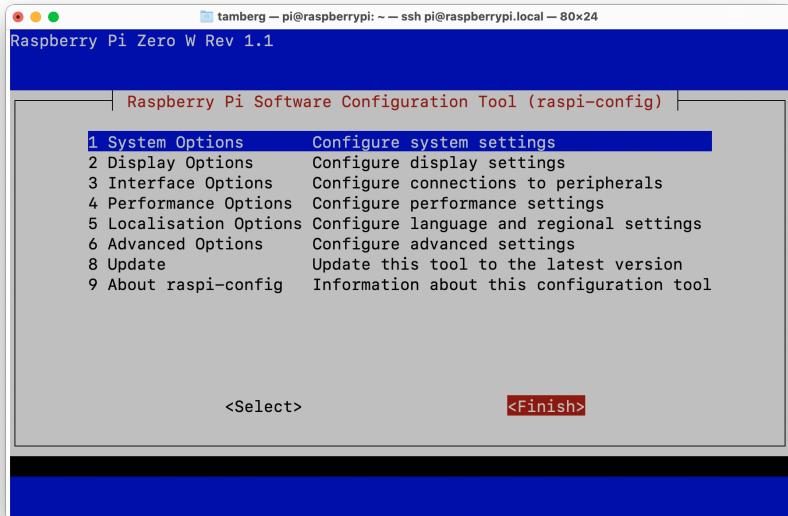
- Yes anwählen:



- *Ok* anwählen:



- *Finish* anwählen:



- Reboot? Yes anwählen:



- Resultat - Kamera ist (nach Reboot) freigeschaltet.

Software einrichten

Voraussetzung: Zugriff auf Pi via SSH und Hardware einrichten.

Node.js und npm installieren

Auf dem Pi:

- \$ cd ~

- `$ wget https://unofficial-builds.nodejs.org/download/release/v20.5.1/node-v20.5.1-linux-armv6l.tar.xz`
- `$ tar xvfJ node-v20.5.1-linux-armv6l.tar.xz`
- `$ sudo cp -R node-v20.5.1-linux-armv6l/* /usr/local`
- `$ rm -rf node-*`
- `$ sudo reboot`
- `$ node -v`
- `$ npm -v`
- Resultat - Node.js und npm sind installiert.

Symlink auf *raspistill* einrichten

Auf dem Pi:

- `$ sudo mkdir -p /opt/vc/bin`
- `$ sudo ln -s /usr/bin/raspistill /opt/vc/bin/raspistill`
- Resultat - Raspistill ist unter /opt/vc/bin/raspistill aufrufbar.

Programm *photo.js* installieren und testen

Auf dem Pi:

- `$ cd ~`
- `$ npm install raspicam`
- `$ wget -O photo.js \`
`https://raw.githubusercontent.com/tamberg/scanshredd-pi/main/photo.js`
- `$ node photo.js`
- `$ ls -al photo.png`
- Resultat - Photo aufgenommen und in Datei *photo.png* gespeichert.

Auf dem PC (Mac oder Win oder Linux):

- Terminal öffnen
- `$ cd ~/Desktop`
- `$ scp -P 22 pi@raspberrypi.local:/home/pi/photo.png ~/Desktop/photo.png`
- `$ open photo.png`
- Resultat - Photo wird auf dem PC angezeigt.

Programm *switch.js* installieren und testen

Auf dem Pi:

- `$ cd ~`
- `$ npm install rpi-gpio`
- `$ wget -O switch.js \`
`https://raw.githubusercontent.com/tamberg/scanshredd-pi/main/switch.js`
- `$ node switch.js`
- Resultat - Button Zustand wird in der Konsole angezeigt als true/false, LED ist an/aus.

Programm *upload.js* installieren und testen

Auf dem Pi:

- \$ cd ~
- \$ npm install request
- \$ wget -O upload.js \
<https://raw.githubusercontent.com/tamberg/scanshredd-pi/main/upload.js>
- \$ node upload.js
- Resultat - <https://partydesjahres.ch/nichts/photobooth.php> zeigt das Photo.

Programm *scanshredd.js* installieren und testen

Auf dem Pi:

- \$ cd ~
- \$ npm install raspicam
- \$ npm install rpi-gpio
- \$ npm install request
- \$ wget -O scanshredd.js \
<https://raw.githubusercontent.com/tamberg/scanshredd-pi/main/scanshredd.js>
- \$ node scanshredd.js
- Resultat - Drücken auf den Button löst ein Photo aus und lädt es hoch auf <https://partydesjahres.ch/nichts/photobooth.php>, steuert LED bzw. Relais.

Dienst einrichten

Voraussetzung: Software einrichten, insbesondere das Programm *scanshredd.js*.

Der Dienst *scanshredd.service* sorgt dafür, dass dieses Programm immer läuft.

Dienst *scanshredd.service* einrichten und starten

Auf dem Pi:

- \$ sudo wget -O /lib/systemd/system/scanshredd.service \
<https://raw.githubusercontent.com/tamberg/scanshredd-pi/main/scanshredd.service>
- \$ sudo systemctl daemon-reload
- \$ sudo systemctl enable scanshredd.service
- \$ sudo systemctl start scanshredd.service
- \$ sudo reboot
- \$ ps aux | grep scanshredd
- Resultat - Service läuft als Prozess, auch nach Reboot.

Dienst *scanshredd.service* stoppen (optional)

Auf dem Pi:

- `$ sudo systemctl stop scanshredd.service`
- Resultat - Service gestoppt, bis zum nächsten Reboot.

Dienst *scanshredd.service* löschen (optional)

Auf dem Pi:

- `$ sudo rm /etc/systemd/system/multi-user.target.wants/scanshredd.service`
- `$ sudo rm /lib/systemd/system/scanshredd.service`
- Resultat - Service gelöscht.

Troubleshooting

Pi USB Power aus- und wieder einstecken :)