

# Raspberry Pi IoT Image

## Inhaltsverzeichnis

1	Raspberry Pi Zugangsdaten	1
2	Installation Shell Script	2
3	Konfigurationsdateien editieren	5
4	Installationen testen und initialisieren	7
5	Raspberry Pi Image verkleinern	9
	Referenzen	13

Für eine möglichst reibungslose Durchführung der Übungen ist ein vorkonfiguriertes Image hilfreich. Dies gewährleistet, dass alle Studierenden mit der gleichen Software arbeiten und die Übungen nicht durch Installationsprobleme verzögert werden.

Für die Übungen mit den Sensoren sind Python und die entsprechenden Sensorbibliotheken erforderlich. Die Installation dieser Bibliotheken ist in den Übungsanleitungen beschrieben. Die Übungen mit dem IoT Stack mit MQTT, InfluxDB, Node-Red und Grafana erfordern eine Installation der Software auf dem Raspberry Pi. Die Installation dieser ist nicht in der Übungsanleitung abgedeckt.

### Hinweis

**Hinweis:** Für das Kopieren mehrerer Images auf SD Karten können diese einzeln auf die SD Karten geschrieben werden oder empfehlenswert für das Kopieren von mehreren SD Karten lohnt es sich auch eine SD Karten Kopierstation zu verwenden, wie z.B. die [Renkforce Speicherkarten-Kopierstation](#)

## 1 Raspberry Pi Zugangsdaten

Notiere die Zugangsdaten für den Raspberry Pi und die einzelnen Softwarekomponenten in der untenstehenden Tabelle. Die Zugangsdaten werden für die Übungen benötigt.

**Tab. 1:** Zugangsdaten zu den einzelnen Konten notieren, gerade bei Datenbanken geht dies leicht vergessen.

Konto	User	Password	Kommentar
Raspberry Pi			
influxdb			organisation:
grafana			

## 2 Installation Shell Script

Shell Script für die Installation der erforderlichen Bibliotheken Script `install_iot_software.sh` mit dem unten aufgeführten Code erstellen und dem Script die Rechte für die Ausführung setzen mit

```
sudo chmod +x install_iot_software.sh
```

Script mit folgendem `sh install_iot_software.sh` oder `./install_iot_software.sh` Befehl ausführen:

### Hinweis

InfluxDB Version: Die Version der InfluxDB auf die neuste Version im Script anpassen.

Shell Script: `install_iot_software.sh` - Bash Script zu Installation der Software und Libraries für das fächerübergreifende Modul 5200 IoT Installation von:

- Python Libraries Beispielcode für die Pimoroni Sensoren
- Klonen der Libraries mit Beispielen in Documents/Libraries
- Jupyter Notebook
- MQTT, Mosquitto Broker und Clients
- Node-Red
- InfluxDB
- Grafana
- VNC

`install_iot_software.sh`: Bash Installations-Script mit Unix Zeilenumbrüchen EOL

```
#!/bin/bash
COL='\033[0;32m' # Primary Color
NC='\033[0m' # No Color
echo "${COL}Raspberry Pi Version${NC}"
```

## Hinweis

**Zeilenumbrüche EOL** Zeilenunterbrüche von Textdateien unterscheiden sich zwischen Unix `LF` und Windows `CRLF` Systemen. Bash Script müssen auf Linux Systemen Unix Zeilenumbrüche (EOL) `LF` aufweisen. Dies ist vor allem dann wichtig, wenn das Script auf einem Windows System erstellt wird. Windows verwendet standardmäßig `CRLF` Zeilenumbrüche. In Visual Studio Code können die EOL Zeichen mit `Ctrl+Shift+P` und `Change End of Line Sequence` auf `LF` umgestellt werden oder über die Statusleiste unten rechts bei `CRLF`. In Notepad++ kann dies über `Edit/EOL Conversion/Unix (LF)` umgestellt werden.

```
hostnamectl
cat /proc/cpuinfo | grep Model

echo "${COL}Raspberry Pi aktualisieren${NC}"
sudo apt update
sudo apt full-upgrade -y
sudo apt autoremove -y

echo "${COL}VNC Installation${NC}"
sudo apt-get install realvnc-vnc-server
sudo apt-get install realvnc-vnc-viewer

echo "${COL}i2c-tools Installation${NC}"
sudo apt install python3-smbus
sudo apt install -y i2c-tools

echo "${COL}Mosquitto Server, Clients und Python Libraries Installation${NC}"
sudo apt install mosquitto -y
sudo apt install mosquitto-clients -y
sudo systemctl enable mosquitto.service
sudo systemctl restart mosquitto

echo "${COL}Node-Red Installation${NC}"
curl -sL https://raw.githubusercontent.com/node-red/linux-
↪ installers/master/deb/update-nodejs-and-nodered -o
↪ update-nodejs-and-nodered.sh
chmod +x update-nodejs-and-nodered.sh
bash update-nodejs-and-nodered.sh --confirm-root --confirm-pi --confirm-install
↪ --no-init
rm update-nodejs-and-nodered.sh
sudo systemctl enable nodered.service
sudo systemctl start nodered.service

echo "${COL}InfluxDB Installation${NC}"
```

```
wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.1-arm64.deb
sudo dpkg -i influxdb2-2.7.1-arm64.deb
sudo service influxdb start

echo "${COL}Grafana Installation${NC}"
sudo apt-get install -y apt-transport-https software-properties-common wget
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee
↪ /etc/apt/keyrings/grafana.gpg > /dev/null
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable
↪ main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
sudo apt-get update
sudo apt-get install grafana -y

sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl enable grafana-server.service
sudo apt autoremove -y

echo "${COL}Klonen der Pimoroni Python Bibliotheken${NC}"
echo "${COL}in Documents/Libraries${NC}"
cd Documents
mkdir Libraries
cd Libraries
git clone https://github.com/pimoroni/st7789-python
git clone https://github.com/pimoroni/bme680-python
git clone https://github.com/pimoroni/icm20948-python
git clone https://github.com/pimoroni/as7262-python
git clone https://github.com/pimoroni/max30105-python
git clone https://github.com/pimoroni/vl53l5cx-python
git clone https://github.com/pimoroni/mlx90640-library
cd ../../
```

Zip Datei des Ordners Libraries erstellen für das Backup der Beispielcode der einzelnen Sensoren.

```
cd ~/Documents/
zip -r Libraries.zip Libraries
```

## Installation der Python Libraries mit *venv*

Mit dem Upgrade auf *Raspberry Pi Bookworm* wird Python auf die Version 3.11 aktualisiert. Ab dieser Version können Python Libraries nicht mehr systemweit installiert werden. *Python Virtual Environment* sind nun erforderlich, entweder pro Projekt oder pro Userkonto siehe <http://rptl.io/venv> für weitere Informationen.

Erstelle eine Python Virtual Environment .env im Homeordner ~/ mit folgenden Befehlen für den User iot und aktiviere die Virtual Environment mit source ~/.env/bin/activate und deaktiviere die Virtual Environment mit deactivate. Wenn die Virtual Environment projebasiert im Projektordner erstellt wird reicht es den Pfad zum Projektordner anzugeben z.B. python -m venv ~/Development/Projekt1/env und die Virtual Environment wird im Projektordner erstellt.

```
python -m venv ~/env
source ~/env/bin/activate
```

Requirements.txt Datei erstellen mit den Python Libraries, die in der Virtual Environment installiert werden sollen. Die Datei kann mit nano requirements.txt erstellt und mit Ctrl+o und Ctrl+x gespeichert werden. Python Libraries in der Virtual Environment installieren mit pip install -r requirements.txt. Nutze für die Installation folgende requirements.txt Datei. Kontrolliere mit pip list ob die Libraries installiert wurden.

*requirements.txt:* Python Libraries für die Pimoroni Sensoren

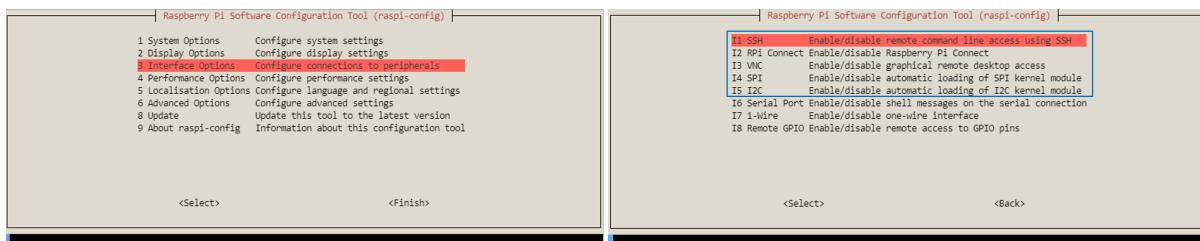
```
matplotlib
scipy
pigments
numpy
jupyter
rpi.gpio
spidev
pillow
numpy
st7789
bme680
icm20948
as7262
max30105
vl53l5cx-ctypes
RPI.GPIO
adafruit-blinka
```

### 3 Konfigurationsdateien editieren

#### Raspberry Pi Einstellungen für I2C, SPI, SSH, VNC anpassen

**Interface Options aktivieren** In den Raspberry Pi Konfigurationseinstellungen mit sudo raspi-config die Interface Optionen *I2C, SPI, SSH und ev VNC (Virtual Network Computing)* oder *RPi Connect* aktivieren. VNC, RPi Connect Hollingworth (2024)<sup>1</sup> werden nur benötigt, wenn der Raspberry Pi über Fernzugriff gesteuert werden soll.

<sup>1</sup>RPi Connect erlaubt über einen Relay Server die Verbindung zu einem Raspberry Pi herzustellen über WebRTC wie Browser Clients für Zoom, Slack, Microsoft Teams. Der Relay Server wird nur für die Erstellung der Verbin-



**Abb. 1:** Über die Konsole `raspi-config` starten und über das Konsolenmenü (links) die Einstellungen in den *Interface Options* (rechts) *I2C*, *SPI*, *SSH* und ev *VNC* aktivieren.

## Baud rate des I2C Protokolls anpassen

Baud rate des I2C Protokolls mit `sudo nano /boot/firmware/config.txt` öffnen und folgende Zeile ergänzen. Mit `Ctrl+o` und `Ctrl+x` speichern (siehe [Raspberry Pi Ltd 2024b](#)).

```
dtoverlay=i2c_arm,arm_baudrate=400000
```

## Mosquitto Server Konfiguration

**Mosquitto Server Konfiguration anpassen** mit `sudo nano /etc/mosquitto/mosquitto.conf` und am Ende der Datei folgendes einfügen ([Eclipse Foundation 2021](#)).

```
listener 1883
allow_anonymous true
```

Mosquitto Server neu starten mit `sudo systemctl restart mosquitto`

**Node-Red Einstellungen initialisieren** mit `sudo node-red admin init`

- yes installation customise settings
- yes keep settings file at default location
- no setup user security
- yes enable project features
- select *manual* workflow
- select *default theme*
- yes allow function nodes to load external modules

```
Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.
 Settings file · /root/.node-red/settings.js
```

dung benötigt und wird von Raspberry Pi kostenlos betrieben. RPi Connect installieren mit `sudo apt install rpi-connect` und mit `sudo reboot` neu starten.

```
User Security
=====
 Do you want to setup user security? · No

Projects
=====
The Projects feature allows you to version control your flow using a local git
↳ repository.

 Do you want to enable the Projects feature? · Yes
 What project workflow do you want to use? · manual - you must manually commit
↳ changes

Editor settings
=====
 Select a theme for the editor. To use any theme other than "default", you will
↳ need to install @node-red-contrib-themes/theme-collection in your Node-RED user
↳ directory. · default
 Select the text editor component to use in the Node-RED Editor · monaco (default)

Node settings
=====
 Allow Function nodes to load external modules? (functionExternalModules) · Yes

Settings file written to /root/.node-red/settings.js
```

## 4 Installationen testen und initialisieren

### Mosquitto Broker

Testen ob der Mosquitto Broker und Clients lokal auf dem Raspberry Pi funktionieren. Erstelle einen Subscriber der für das Topic `iot/temperature` eine Subscription erstellt.

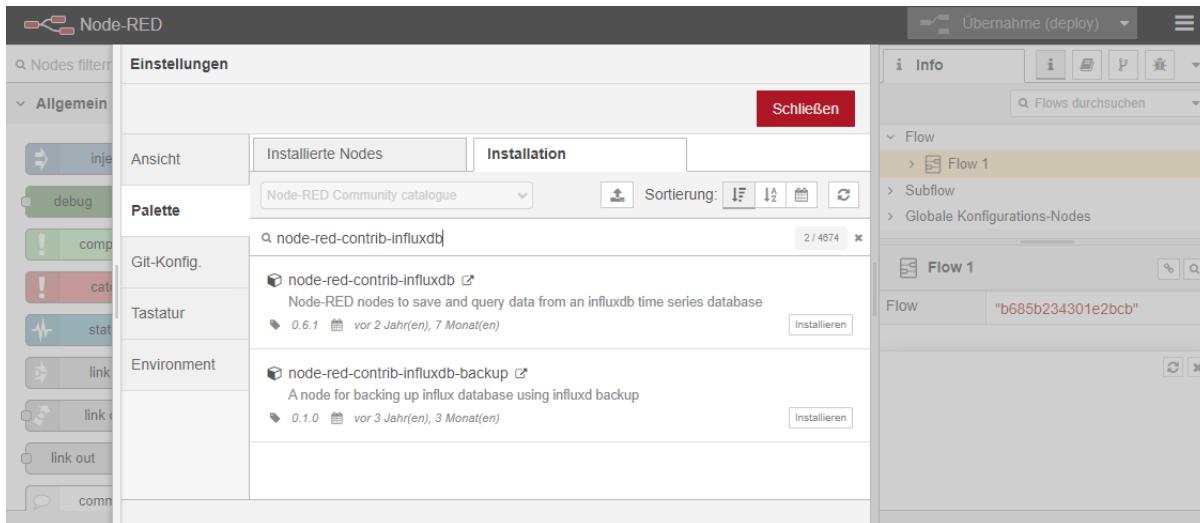
```
mosquitto_sub -h 127.0.0.1 -v -t 'iot/temperature'
```

Öffne ein zweite Shell und erstelle einen Publisher für dasselbe Topic

```
mosquitto_pub -h 127.0.0.1 -t 'iot/temperature' -m 'Aussentemperatur: 22° Celsius'
```

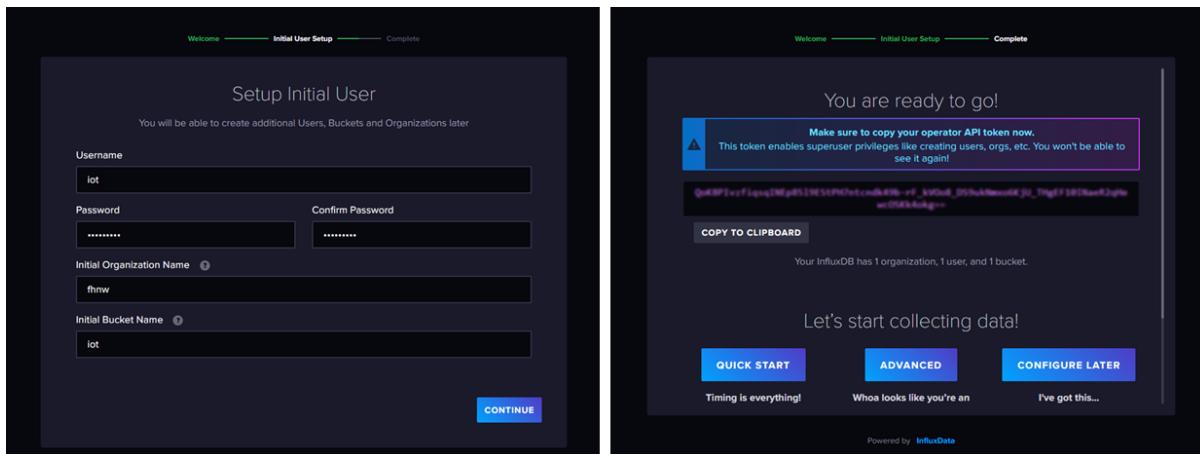
### Node-Red

Node-Red Server testen : <IP-Adresse>:1880 <http://192.168.1.205:1880> Influxerweiterung  
Palette installieren: *Hamburgermenu oben-rechts / Paletten verwalten* und unter *Palette / Installation* die Palette `node-red-contrib-influxdb` installieren.



## InfluxDB

InfluxDB Server testen: <IP-Adresse>:8086 <http://192.168.1.205:8086> Den *Initial User* erstellen mit *Username*, *Password*, *Organisation* und einem *Bucket Name*, der ersten Datenbank. Im Anschluss mit Logout/Login den Zugang verifizieren.

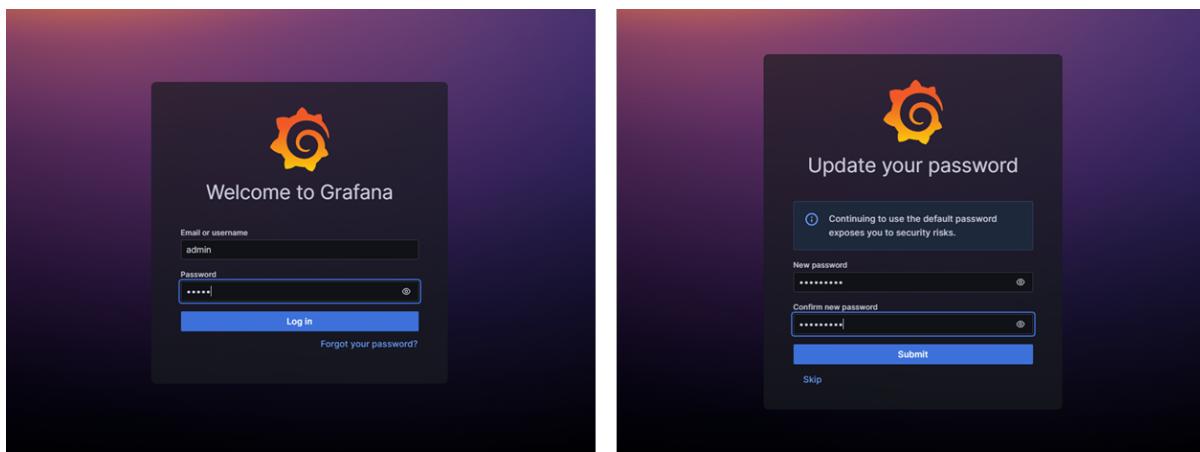


### Hinweis

**Operator API token speichern!** Das Superuser Passwort wird nur einmal angezeigt. Das **Operator API token** kopieren und an einem sicheren Ort speichern! Nach dem Logout/Login ist das Passwort nicht mehr sichtbar.

## Grafana

Grafana Server testen: <IP-Adresse>:3000 <http://192.168.1.205:3000> Grafana startet mit dem Default Admin User: `admin` und Passwort: `admin`. Das Passwort ändern und im Anschluss mit Logout/Login den Zugang verifizieren.



## Jupyter Notebook

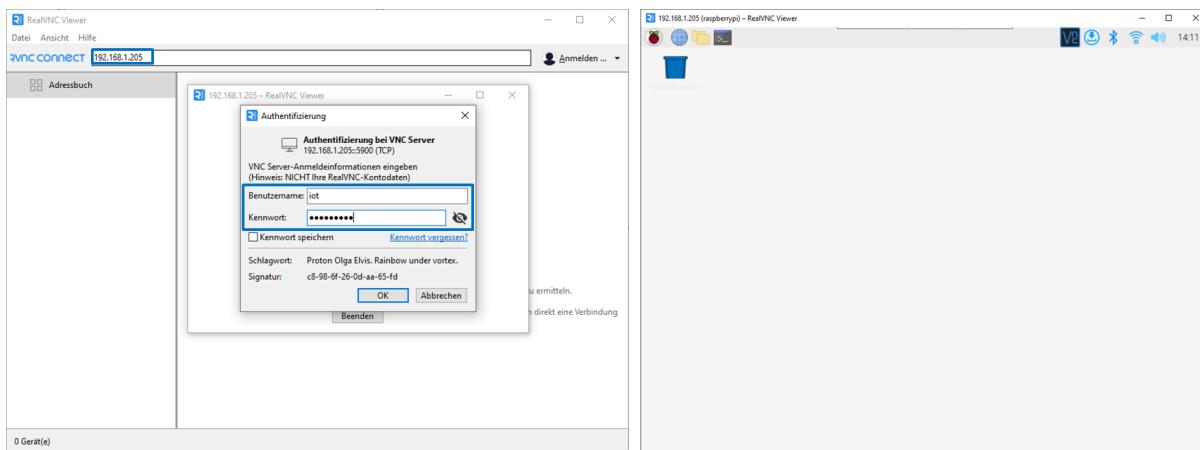
Jupyter Notebook testen <IP-Adresse>:9999 <http://192.168.1.205:9999>

```
# jupyter notebook installation testen
jupyter-notebook --no-browser --ip=192.168.1.205 --port 9999 --notebook-dir
↪ ~/Documents
```

## VNC Testen

VNC Viewer auf dem PC installieren. [Real VNC Viewer](#) (standalone .exe, d.h. es ist keine Installation erforderlich).

**Hinweis:** Beim Starten des VNC Viewers muss man sich nicht anmelden sondern nur die Option **Verwenden Sie RealVNC Viewer ohne sich anzumelden** wählen.



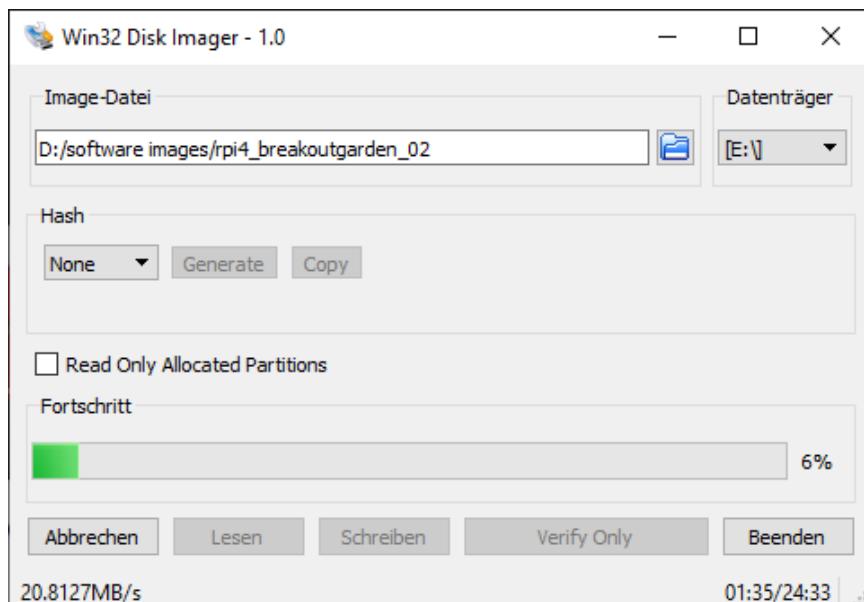
## 5 Raspberry Pi Image verkleinern

Das Raspberry Pi Image kann mit dem Tool [PiShrink](#) verkleinert werden. PiShrink kann auf dem Raspberry Pi oder auf einen anderen Linux System wie Ubuntu oder einer virtuellen Machine mit einem Linux installiert und genutzt werden.

1. Backup Image auf den lokalen Rechner mit [Win32DiskImager](#) lesen und in eine Image Datei schreiben.
2. Virtuelle Machine mit Linux Ubuntu/Debian mit *gemeinsamen Ordner* starten
3. Mit dem Script <https://github.com/Drewsif/PiShrink> das Image verkleinern
4. SD Karte Formatieren mit [SD Card Formatter](#)
5. Image auf SD Karte schreiben mit *Win 32 Disk Imager* oder *Raspberry Pi Imager*
6. Für die Archivierung kann das Image mit 7zip zusätzlich komprimiert werden (Archive Format .xz und Kompressionsstufe Ultra)

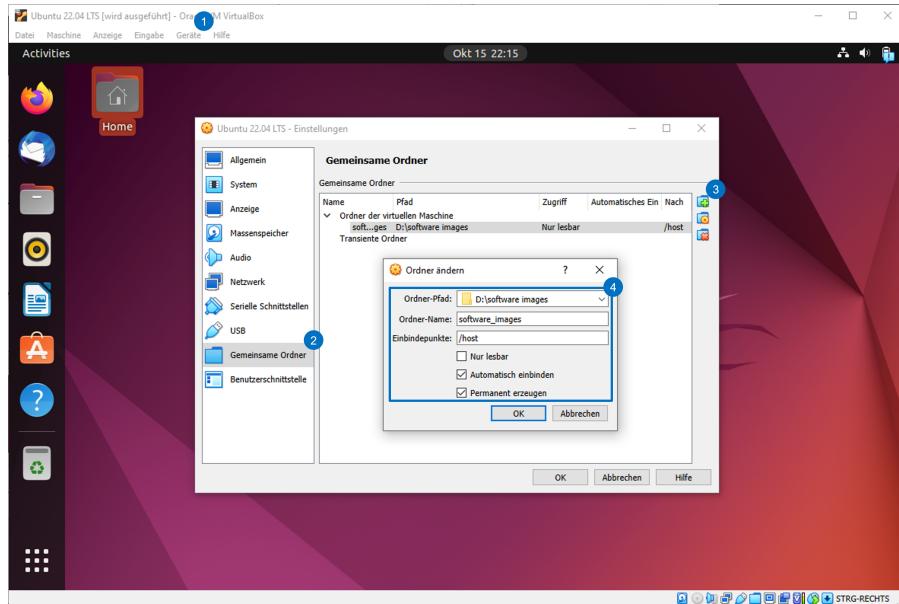
Tutorial: [How I Backup and Shrink My SD Card Images - Youtube](#)

Backup Image mit Win32DiskImager schreiben, mit der Funktion *Lesen/Read* und diese als Image Datei auf den lokalen Rechner speichern.



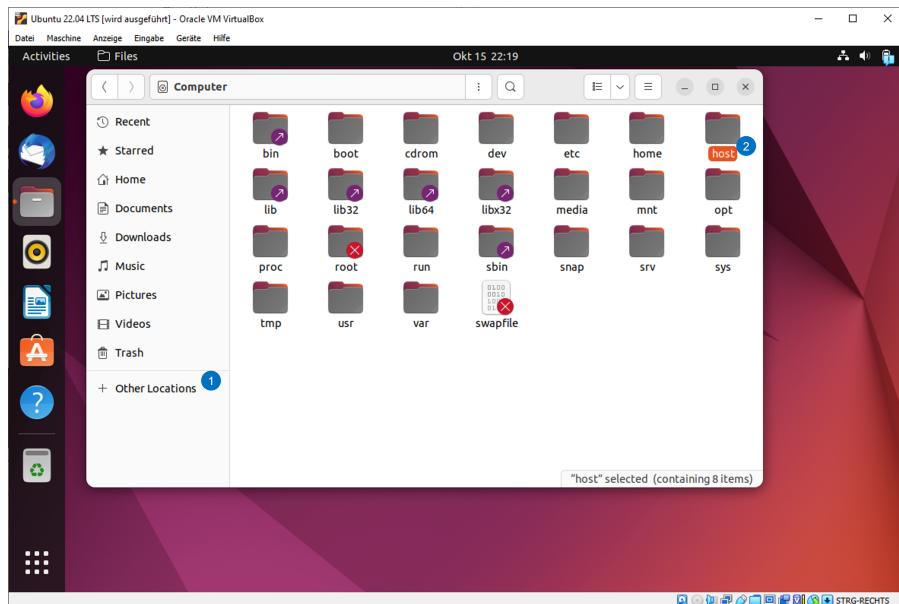
**Abb. 2:** Backup Image der SD Karte schreiben mit Win32DiskImager und der Funktion *Lesen/Read*

Virtuelle Machine mit VM [VirtualBox](#) starten und in den Einstellungen der VM über *Geräte/Gemeinsame Ordner* einen gemeinsamen Ordner zwischen VM und dem lokalen Rechner einrichten.



**Abb. 3:** Gemeinsamer Ordner in der VM VirtualBox einrichten

Das Image sollte nun im File Explorer unter “+ Other Locations” als Ordner *host* eingebunden und sichtbar sein.



**Abb. 4:** Gemeinsamer Ordner *host* auf der VM Virtualbox eingebunden

Öffne nun ein Terminal damit das Script `pishrink.sh` installiert und ausgeführt werden kann.

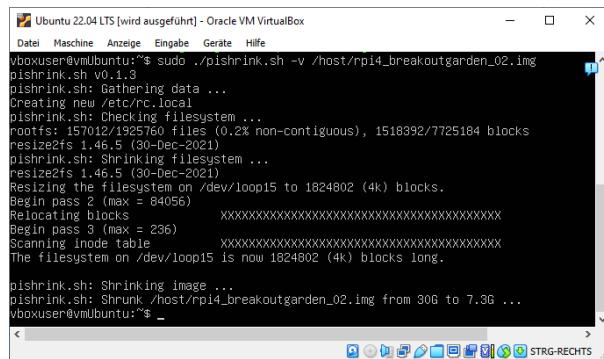
**Tipp:** Falls das Terminal nicht öffnet mit `Ctrl+Alt+F3` in das TTY Terminal wechseln.

**Tipp:** Falls der VM User noch keine Adminrechte hat (bei einer Neuinstallation von Ubuntu ist das `root` Passwort noch nicht gesetzt), können diese nachträglich gesetzt werden mit `sudo`

passwd. Bei der Passwortabfrage das aktuelle User Passwort eingeben und das neue Root Passwort zweimal eingeben. Anschliessend als Root User anmelden mit `su -`.<sup>2</sup>

GitHub: <https://github.com/Drewsif/PiShrink>

```
# install prerequisites
sudo apt install parted xz-utils
# download the pishrink script
wget -O ./pishrink.sh
↳ https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh
# make the script executable
chmod +x ./pishrink.sh
# run pishrink on the .img file
sudo ./pishrink.sh -v /host/<rpi-imagefile>.img
# Komprimiere die Image als .xz Archive (eher langsam)
xz -z -9e /host/<rpi-imagefile>.img
```

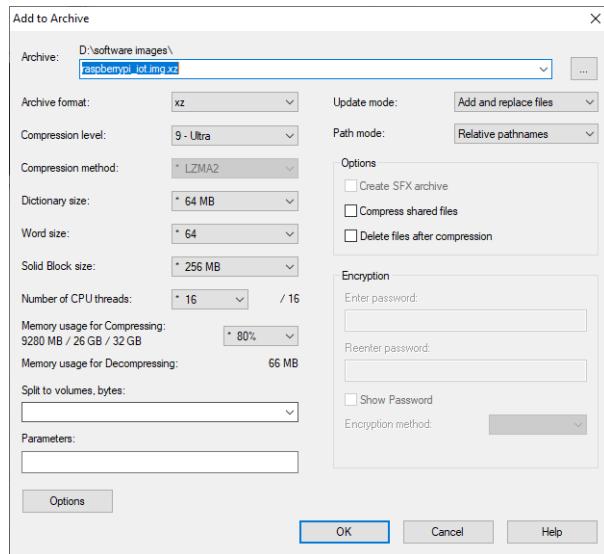


**Abb. 5:** Screenshot des Terminals während der Ausführung von pishrink

Image mit 7-Zip zusätzlich komprimieren File Kontextmenu **7-Zip/Add to Archive** und als Archive Format **.xz** sowie den Compression Level **9-Ultra** wählen.

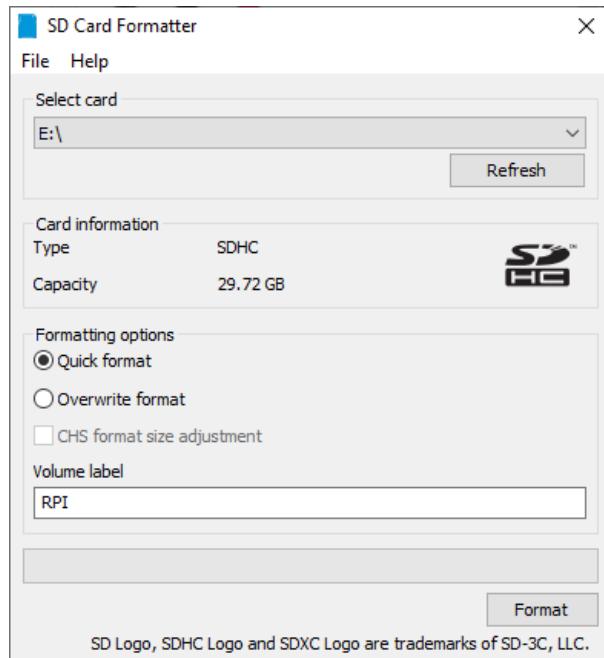
---

<sup>2</sup>Einem User Adminrechte zuweisen: `sudo usermod -a -G sudo <username>`, Kontoinformationen id <username> aufzeigen.



**Abb. 6:** 7-Zip Einstellungen für die zusätzliche Kompression des Images mit .xz.

Falls die SD Karte schon früher für Raspberry Pi Images benutzt wurde und mehrere Partitionen aufweist, kann es helfen die Karte mit dem [SD Card Formatter](#) nochmals von Grund auf zu formatieren.



**Abb. 7:** Screenshot des SD Card Formatter

## Referenzen

- Eclipse Foundation, 2021. Authentication Methods. *Eclipse Mosquitto*.  
Hollingworth, G., 2024. Raspberry Pi Connect. *Raspberry Pi*.

Raspberry Pi Ltd, 2024a. Raspberry Pi Connect Beta - Access Your Raspberry Pi from Anywhere. *Raspberry Pi*.

Raspberry Pi Ltd, 2024b. Raspberry Pi Documentation Config.Txt. *Raspberry Pi*.