NVMe SSD's nutzen (Pi 5)

> **Wichtig**
> Die folgende Anleitung bezieht sich nur auf den **Raspberry Pi 5**!

YouTube Video #79

Erklärungen

PCIe

PCI Express (PCIe) ist ein serielles Hochgeschwindigkeitsübertragungsprotokoll für die Datenübertragung zwischen Hauptplatinen und Peripheriegeräten. Es verwendet differenzielle Signalpaare und Punkt-zu-Punkt-Verbindungen, wodurch höhere Datenraten im Vergleich zu älteren parallelen Busstandards wie PCI ermöglicht werden. PCIe bietet mehrere Lanes (Datenspurgruppen), wodurch die Bandbreite je nach Anforderungen skaliert werden kann. Diese Lanes können in verschiedenen Konfigurationen wie x1, x4, x8 und x16 vorliegen. Durch seine Flexibilität und hohe Leistungsfähigkeit wird PCIe weitläufig für Grafikkarten, SSDs, Netzwerkkarten und andere Hochleistungsperipheriegeräte eingesetzt.

NVMe

> **Hinweis**
> NVMe ist nicht SSD mSATA !

NVMe (Non-Volatile Memory Express) ist ein Protokoll, das speziell für den Zugriff auf NAND-basierten Flash-Speicher optimiert ist, wie er in modernen Solid-State-Drives (SSDs) verwendet wird. NVMe ermöglicht eine effiziente und parallele Datenübertragung zwischen dem Host (CPU) und der SSD über den PCIe-Bus. Im Gegensatz zu älteren Protokollen wie AHCI reduziert NVMe Latenzen und maximiert die Bandbreite, was zu deutlich schnelleren Lese- und Schreibgeschwindigkeiten führt. NVMe-SSDs werden zunehmend in Hochleistungsrechnern und Datenspeicherlösungen eingesetzt, um die Leistungsfähigkeit von Flash-Speicher optimal zu nutzen.

M.2 (NGFF)

M.2 (= NGFF = Next Generation Form Factor) ist ein Formfaktor und Schnittstellenstandard für kompakte, kleine Erweiterungskarten, die in M.2-Steckplätzen auf Hauptplatinen eingesteckt werden. M.2-Steckplätze unterstützen verschiedene Schnittstellen, darunter SATA und PCIe, wodurch sie vielseitig für den Anschluss von SSDs, WLAN-Karten und anderen Erweiterungen sind. Der M.2-Formfaktor ermöglicht es, Platz zu sparen und die Verbindung von Hochleistungsgeräten wie NVMe-SSDs direkt über den PCIe-Bus für schnelle Datenübertragungen zu unterstützen.

M Key

“M Key” ist eine Bezeichnung für einen der physischen Schlüssel (Keying) in M.2-Steckplätzen. Der M.2-Steckplatz selbst kann verschiedene Schlüssel oder Buchsen haben, um die Kompatibilität mit unterschiedlichen Arten von M.2-Modulen sicherzustellen. In einem M.2-Steckplatz mit einem M Key ist der Schlitz so gestaltet, dass er speziell für M.2-Module mit einem M Key passt. M-Key-Module können verschiedene Arten von Schnittstellen verwenden, einschließlich PCIe (PCI Express) und SATA, und werden häufig für Hochleistungsgeräte wie NVMe-SSDs verwendet.

- https://www.delock.de/infothek/M.2/M.2.html
- https://de.wikipedia.org/wiki/M.2
- https://www.atpinc.com/blog/what-is-m.2-M-B-BM-key-socket-3

Hinweise

- Bevor eine NVMe oder der Adapter dazu installiert wird, immer den Pi stromlos machen!
- Die WD Green SSDs sind laut Pimoroni nicht zu empfehlen. Laufwerke wie das SN350 führen wohl öfters zu Problemen am CM4 und Pi 5.
- Es empfiehlt sich erstmal von SD zu booten, um den Bootloader & Bootloader Einstellungen anzupassen!

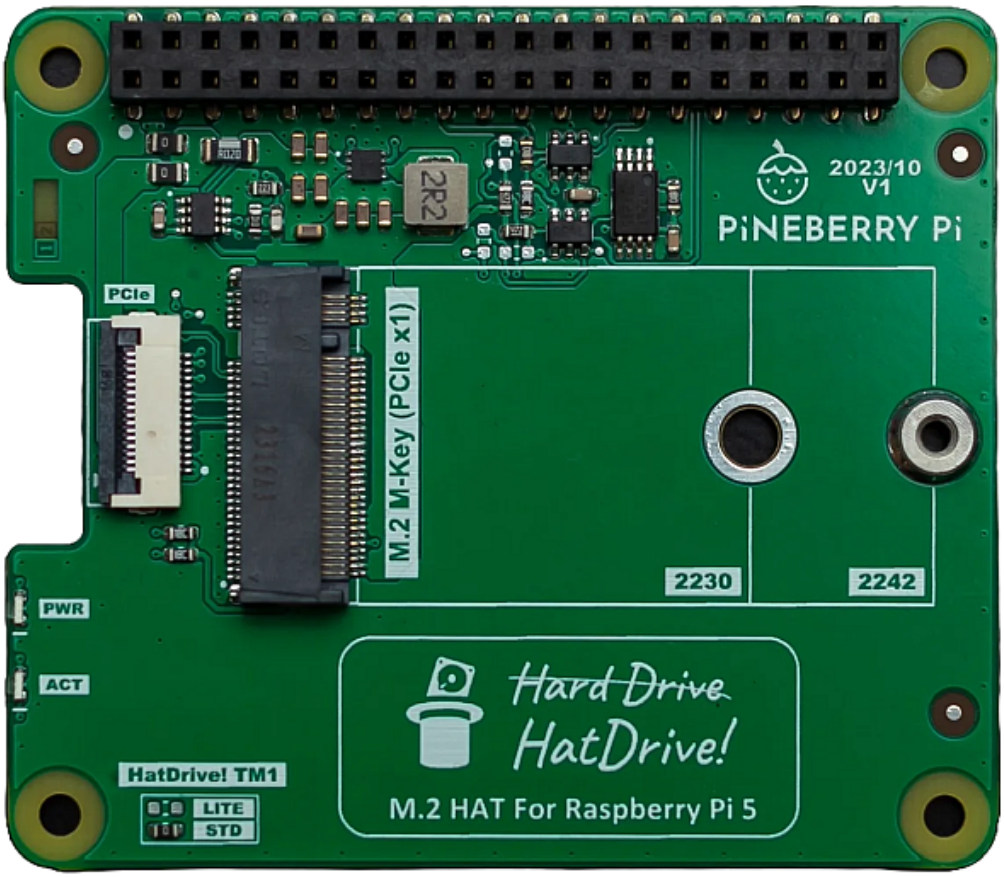
Hardware

Für den Betrieb am Pi wird ein NVMe Adapter benötigt, der an den PCIe Port angeschlossen wird. Der USB auf NVMe Adapter dient vornehmlich zum Bespielen einer NVMe unter Windows. Er funktioniert aber auch mit einem Raspberry Pi - allerdings mit Geschwindigkeitseinbußen. Und er wird behandelt wie ein USB-Stick.

NVMe Base

- https://shop.pimoroni.com/products/nvme-base?variant=41219587178579
- Unterstützt alle gängigen Formate von 2230 bis 2280

HatDrive!

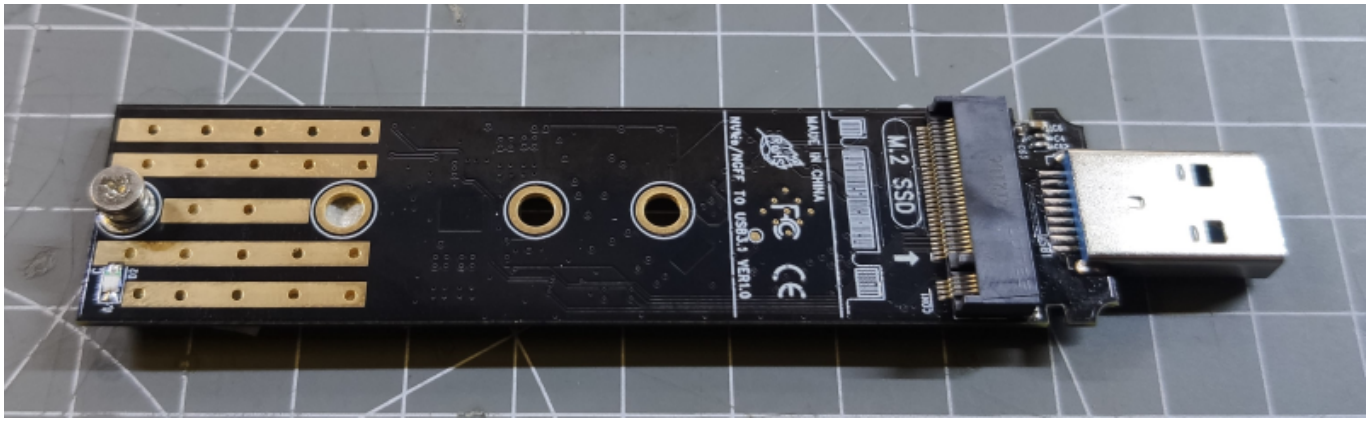


- https://pineberrypi.com/products/hat-top-2230-2240-for-rpi5
- https://pineberrypi.com/products/hatdrive-bottom-2230-2242-2280-for-rpi5
- Zwei Versionen verfügbar

Typ X 1001

TBD

USB auf NVMe Adapter



Solche Adapter gibt es bei eBay / Amazon für 10-15€. Zu finden unter der Bezeichnung **USB NVMe**

Adapter.

Bootloader Update

> **Achtung**
> Ohne aktuelle Firmware (Bootloader) wird die NVMe erst gar nicht erkannt!
> **Hinweis**
> Der Bootloader wird auf dem Board selber geflasht. Also selbst bei einem Wechsel von SD / USB / NVMe bleibt er in seiner Version erhalten!

- `sudo apt update -y && sudo apt upgrade -y && sudo apt install fio gparted piclone -y`
- Version prüfen
  `sudo rpi-eeprom-update`

```
pi@Pi5Test:~ $ sudo rpi-eeprom-update
*** UPDATE AVAILABLE ***
BOOTLOADER: update available
   CURRENT: Wed 13 Sep 10:37:06 UTC 2023 (1694601426)
    LATEST: Wed  6 Dec 18:29:25 UTC 2023 (1701887365)
   RELEASE: default (/lib/firmware/raspberrypi/bootloader-2712/default)
            Use raspi-config to change the release.
```

- Update durchführen auf eine Version > 06.12.2023
  `sudo rpi-eeprom-update -a`
  `sudo reboot`

Alternativ kann die Firmware auch über `sudo raspi-config` auf den aktuellen Stand gebracht werden:

- `sudo raspi-config`
  - 6 Advanced Options
  - A5 Bootloader Version
  - E1 Latest Use the latest version boot ROM software
  - Die Frage mit YES beantworten
  - Und den Reboot durchführen lassen

Bootloader Konfig

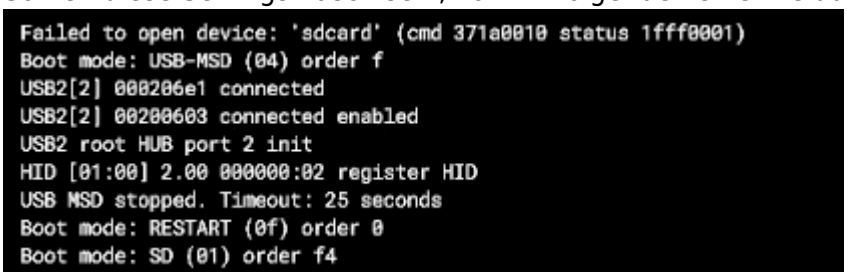
> **Hinweise**
> 1) Die Bootreihenfolge kann generell wie hier beschrieben eingestellt werden. Kann von NVMe nicht gebootet werden fällt der Pi auf SD-Karte oder USB-Stick zurück.
> 2) Die Bootreihenfolge lässt sich auch mit `sudo raspi-config` (6 Advanced Options → A4 Boot Order) anpassen. Allerdings wird dabei nicht der Eintrag `PCIE_PROBE=1` ergänzt was bei bestimmten NVMe SSD's zu Bootproblemen führen kann.
> 3) Die Bootloader Konfig kann man sich auch mit `sudo rpi-eeprom-config` nur anzeigen lassen.

Um von NVMe booten zu können, müssen die Bootloader Parameter angepasst werden:

- `sudo rpi-eeprom-config --edit`
  - BOOT_ORDER muss auf 0xf416 geändert werden:
    `BOOT_ORDER=0xf416`
  - Folgender Eintrag muss ggf. hinzugefügt werden:
    `PCIE_PROBE=1`
- Mit STRG+x dann Y und `Enter` das Editieren beenden
- Es kommt dann folgende Ausgabe

```
Updating bootloader EEPROM
....
   CURRENT: Fri  5 Jan 15:57:40 UTC 2024 (1704470260)
    UPDATE: Fri  5 Jan 15:57:40 UTC 2024 (1704470260)
    BOOTFS: /boot/firmware
'/tmp/tmp.adUegKWjEM' -> '/boot/firmware/pieeprom.upd'
Copying recovery.bin to /boot/firmware for EEPROM update

EEPROM updates pending. Please reboot to apply the update.
To cancel a pending update run "sudo rpi-eeprom-update -r".
```

- Neustart mittels
  `sudo reboot`
- Sollten diese Settings flasch sein, kommt folgende Fehlermeldung beim Booten:
  

  In dem Fall muss man den Pi von SD-Karte bzw. USB-Stick booten und die Settings erneut kontrollieren!

Grundlegende Tests

PCIe aktivieren

> **Hinweis**
> Die NVMe wurde trotz fehlendem Eintrag erkannt mit der letzten Firmware.

- `sudo nano /boot/firmware/config.txt`
  - folgenden Eintrag in der Datei ergänzen
    `dtparam=pciex1`
  - Den Editor mit STRG+X, dann Y und `Enter` verlassen
- `sudo reboot`

PCIe checken

- Prüfen kann man den aktivierten PCIe Port mittels dmesg

```
pi@Pi5Test:~ $ pi@Pi5Test:~ $ dmesg |grep brcm-pcie |grep 'PCI host
bridge'
[    0.410070] brcm-pcie 1000110000.pcie: PCI host bridge to bus
0000:00
[    0.530924] brcm-pcie 1000120000.pcie: PCI host bridge to bus
0001:00
```

  - hier werden 2 Host Bridges gelistet. Ist der Port deaktiviert, liefert dmesg nur eine Bridge.

- Eine weitere Möglichkeit besteht mit `lspci`

```
pi@Pi5Test:~ $ lspci
0000:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712
(rev 21)
0000:01:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd
NVMe SSD Controller 980
0001:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712
(rev 21)
0001:01:00.0 Ethernet controller: Device 1de4:0001
```

  - Auch hier werden 2 bridges gelistet wenn PCIe aktiviert ist.

PCIe V3 Mode aktivieren

Offiziell unterstützt der Raspberry Pi 5 nur PCIe 2 - der Bus würde als mit 5GB/s arbeiten. Es ist allerdings möglich den PCIe Bus auf die Version 3 einzustellen. In dem Fall arbeitet der Bus mit 10GB/s.

- `sudo nano /boot/firmware/config.txt`
  - folgenden Eintrag in der Datei ergänzen
    `dtparam=pciex1_gen=3`
  - Den Editor mit STRG+X, dann Y und `Enter` verlassen
- `sudo reboot`

PCIe V3 checken

- Prüfen kann man den V3 Mode wieder mit dmesg
  `dmesg |grep brcm-pcie |grep link`
- Wenn PICe V3 aktiviert ist, läuft ein PCIe Port mit 8.0 GT/s
  - PCIe V3 aktiviert

```
pi@Pi5Test:~ $ dmesg |grep brcm-pcie |grep link
[    0.613326] brcm-pcie 1000110000.pcie: link up, 8.0 GT/s PCIe
x1 (!SSC)
[    0.733329] brcm-pcie 1000120000.pcie: link up, 5.0 GT/s PCIe
x4 (!SSC)
```

  - PCIe V2 aktiviert

```
pi@Pi5Test:~ $ dmesg |grep brcm-pcie |grep link
[    0.517003] brcm-pcie 1000110000.pcie: link up, 5.0 GT/s PCIe
x1 (!SSC)
[    0.637007] brcm-pcie 1000120000.pcie: link up, 5.0 GT/s PCIe
x4 (!SSC)
```

> **Was ist GT/s?**
> Dazu gibt es hier ein paar Erklärungen :
> https://de.wikipedia.org/wiki/Transfers_pro_Sekunde
> https://3roam.com/pcie-bandwidth-calculator/
> https://forum.huawei.com/enterprise/en/pcie-transfer-rate-and-available-bandwidth/thread/667265205186478081-667213852955258880

NVMe prüfen

- `dmesg |grep nvme`

```
pi@Pi5Test:~ $ dmesg |grep nvme
[    0.633625] nvme nvme0: pci function 0000:01:00.0
[    0.633663] nvme 0000:01:00.0: enabling device (0000 -> 0002)
[    0.637575] nvme nvme0: Shutdown timeout set to 8 seconds
[    0.646457] nvme nvme0: allocated 64 MiB host memory buffer.
[    1.150493] nvme nvme0: 4/0/0 default/read/poll queues
[    1.581354]  nvme0n1: p1
[    4.906692] EXT4-fs (nvme0n1p1): mounted filesystem with ordered
data mode. Quota mode: none.
```

- `lsblk`

```
pi@Pi5Test:~ $ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sdb           8:16   1  28.6G  0 disk
├─sdb1        8:17   1   512M  0 part /boot/firmware
└─sdb2        8:18   1  28.1G  0 part /
nvme0n1     259:0    0 238.5G  0 disk
└─nvme0n1p1 259:1    0 238.5G  0 part /nvme
```

- `lspci -v`
  - → Beispiel : `0000:01:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd NVMe SSD Controller 980 (prog-if 02 [NVM Express])`
- grafische Tools
  - `export XAUTHORITY=~/.Xauthority`
  - `sudo gnome-disks`
  - `sudo gparted`

Verwendung

Es gibt durchaus verschiedene Szenarien zur Einrichtung und Verwendung einer NVMe SSD.

- **Booten von einer NVMe SSD erstellt mit Raspberry Pi Imager**
  - Es ist ein USB auf NVMe Adapter nötig!
  - Bootloader Update
  - Bootloader Konfiguration (erfordert Booten von SD-Karte oder USB-Stick)
  - Raspberry Pi Image auf NVMe SSD spielen (mittels Raspberry Pi Imager)
  - Anpassen der config.txt
    - Die Anpassung erfolgt unter Windows mit einem Editor (z.B. Notepad)
    - Evtl. muss der USB Adapter 1x neu in den USB Port gesteckt werden weil der Pi Imager das Medium auswirft.
    - Dann auf der Partition mit dem Namen “bootfs” die Datei `config.txt` editieren. Folgenden Inhalt einfügen:
      ```
      dtparam=pciex1
      dtparam=pciex1_gen=3
      ```
  - Die NVMe über einen passenden Adapter am Pi einstecken, SD-Karte bzw. USB-Stick entfernen und den Pi booten.
- **Booten von einer NVMe SSD geklont mittels piclone**
  - Bootloader Update
  - Bootloader Konfiguration (erfordert Booten von SD-Karte oder USB-Stick)
  - Anpassen der config.txt
    - Die Anpassung erfolgt unter Linux mit einem Editor (z.B. nano)
    - `sudo nano /boot/config.txt` und folgenden Inhalt einfügen:
      ```
      dtparam=pciex1
      dtparam=pciex1_gen=3
      ```
    - Speichern mittels STRG+x dann Y und dann `Enter`
  - Raspberry Pi Image auf NVMe SSD klonen (mittels der Pi Software `piclone`)
- **NVMe SSD als zuätzliche Platte verwenden**
  - Update Firmware
  - Anpassen der config.txt
  - SSD Mount

Boot (Raspberry Pi Imager)

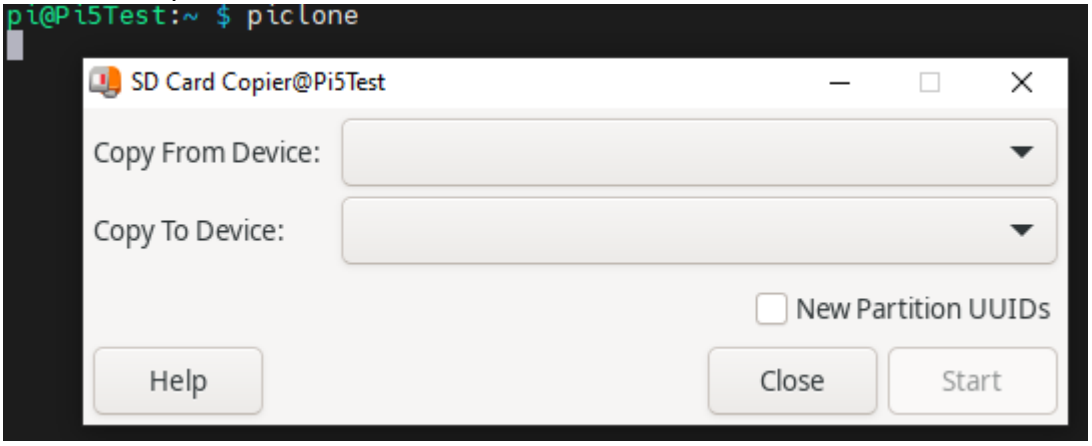
- Die NVMe in einen passenden USB auf NVMe Adapter installieren.
- Das “Konstrukt” in den Windows PC stecken und den Raspberry Pi Imager starten (ggf. Update machen!)
- Als **Raspberry Pi Modell** den **RASPBERRY PI 5** wählen
- Als **Betriebssystem** eine **64Bit Bookworm** Variante wählen (Sonst gibt es u.A. Probleme mit dem Bootloader Update)
- Als **SD-Karte** sollte der USB auf NVMe mit der NVMe SSD angeboten werden
- Schreiben lassen und fertig
- Die NVMe “am” Pi installieren, Strom drauf und er sollte von der NVMe booten 🙂

Boot (piclone)

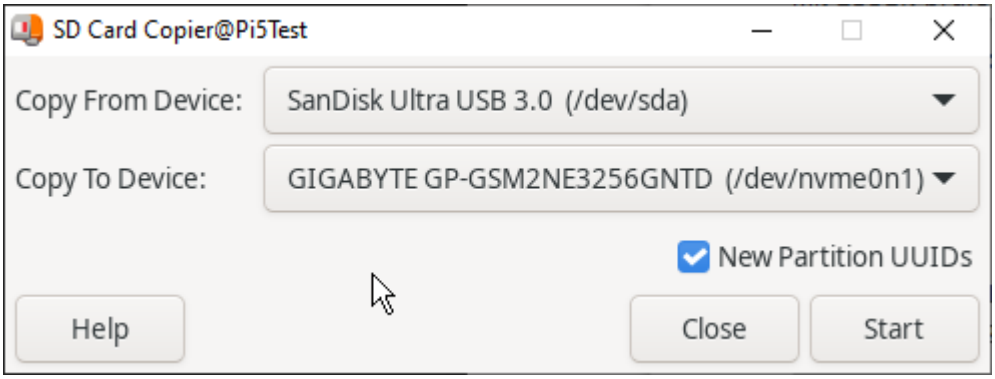
- Zunächst mal muss der Pi von einer SD-Karte oder einem USB-Stick booten (64Bit BookWorm Image)
- Mit `lsblk` prüfen, ob die NVMe auch erkannt wurde (nvme0..):

```
pi@Pi5Test:~ $ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda           8:0    1  28.6G  0 disk
├─sda1        8:1    1   512M  0 part /boot/firmware
└─sda2        8:2    1  28.1G  0 part /
nvme0n1     259:0    0 238.5G  0 disk
├─nvme0n1p1 259:1    0   512M  0 part
└─nvme0n1p2 259:2    0   238G  0 part
```

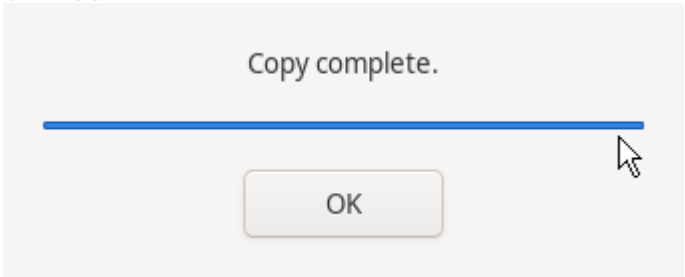
- Im Terminal `piclone` starten:



- Oben das aktuelle Bootmedium auswählen (in dem Fall ein USB-Stick) und unten die NVMe als Ziel:



- `New Partition UUIDs` kann man im Normalfall anhaken
- Dann auf **Start** klicken, die Sicherheitsfrage mit `Yes` beantworten und warten. Wenn alles geklappt hat, kommt das



- piclone schließen
- Das System herunter fahren mittels `sudo poweroff`
- Das System stromlos machen, den USB-Stick (oder die SD-Karte) entfernen
- Strom wieder dran und der Pi sollte von der NVMe booten 🙂

SSD Mount

- UUID ermitteln
  `sudo blkid`

```
pi@Pi5Test:~ $ sudo blkid
/dev/nvme0n1p1: UUID="077d58d3-7ea8-4eba-88f6-49472da6ba98"
BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="a8380a80-01"
/dev/sda2: LABEL="rootfs" UUID="4aa56689-dcb4-4759-90e6-179beae559ac"
BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="6b2f5983-02"
/dev/sda1: LABEL_FATBOOT="bootfs" LABEL="bootfs" UUID="EF6E-C078"
BLOCK_SIZE="512" TYPE="vfat" PARTUUID="6b2f5983-01"
```

  Hier ist es dann aus der nvme Zeile das **a8380a80-01**

- Einen Ordner erstellen in den die NVMe gemountet werden kann
  `sudo mkdir /nvme`
- Die Zugriffsrechte auf den aktuellen User (hier pi) umbiegen
  `sudo chown pi:pi /nvme`
- Die fstab Datei anpassen um das Dateisystem automatisch einzubinden
  `sudo nano /etc/fstab`
  eine Zeile einfügen mit der korrekten PARTUUID und dem richtigen Ordner
  `PARTUUID=a8380a80-01 /nvme auto rw,relatime,nofail 0 0`
  - Datei speichern mit STRG+x dann Y dann `Enter`
- `sudo reboot`
- Sollte die SSD nicht eingebunden werden unter `/nvme` kann man das folgendermaßen analysieren
  `sudo mount -a`

Speedtest

fio

- `sudo apt update -y && sudo apt upgrade -y && sudo apt install fio gparted -y`
- Eine Datei anlegen
  `cd ~ && nano chkspeed.sh`
  und folgenden Inhalt reinkopieren:

```
#!/bin/bash
echo "Testing Sequential READ"
rm -rf $1/random_read.fio
```

```
fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --
name=test --filename=$1/random_read.fio --bs=64k --iodepth=64 --size=$2
--readwrite=read |grep READ
echo "Testing Random     READ"
rm -rf $1/random_read.fio
fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --
name=test --filename=$1/random_read.fio --bs=64k --iodepth=64 --size=$2
--readwrite=randread |grep READ
echo "Testing Sequential WRITE"
rm -rf $1/random_read.fio
fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --
name=test --filename=$1/random_read.fio --bs=64k --iodepth=64 --size=$2
--readwrite=write |grep WRITE
echo "Testing Random     WRITE"
rm -rf $1/random_read.fio
fio --randrepeat=1 --ioengine=libaio --direct=1 --gtod_reduce=1 --
name=test --filename=$1/random_read.fio --bs=64k --iodepth=64 --size=$2
--readwrite=randwrite |grep WRITE
rm -rf $1/random_read.fio
```

- Die Datei ausführbar machen
  `chmod +x chkspeed.sh`
- Das zu testende Speichermedium sollte am Pi angeklemmt sein und als Dateisystem ext4 haben. Testen kann man das mit `lsblk -f` :

```
pi@Pi5Test:~ $ lsblk -f
NAME        FSTYPE FSVER LABEL  UUID
FSAVAIL FSUSE% MOUNTPOINTS
nvme0n1
├─nvme0n1p1 vfat   FAT32 bootfs EF6E-C078
438.1M    14% /boot/firmware
└─nvme0n1p2 ext4   1.0   rootfs 4aa56689-dcb4-4759-90e6-179beae559ac
209.5G     5% /
```

- Hinweis: Normalerweise sollte man ein Laufwerk nicht testen, wenn es als Bootmedium verwendet wird. Aber für einen einfachen Test auf einem unbelasteten Pi 5 sollte es trotzdem klappen 🙂
- Test starten mit `sudo ~/chkspeed.sh ~ 1GB`

```
pi@Pi5Test:~ $ sudo ~/chkspeed.sh ~ 1GB
Testing Sequential READ
   READ: bw=785MiB/s (823MB/s), 785MiB/s-785MiB/s (823MB/s-823MB/s),
io=1024MiB (1074MB), run=1305-1305msec
Testing Random     READ
   READ: bw=799MiB/s (838MB/s), 799MiB/s-799MiB/s (838MB/s-838MB/s),
io=1024MiB (1074MB), run=1281-1281msec
Testing Sequential WRITE
  WRITE: bw=671MiB/s (704MB/s), 671MiB/s-671MiB/s (704MB/s-704MB/s),
io=1024MiB (1074MB), run=1526-1526msec
```

```
Testing Random     WRITE
  WRITE: bw=698MiB/s (731MB/s), 698MiB/s-698MiB/s (731MB/s-731MB/s),
io=1024MiB (1074MB), run=1468-1468msec
```

- USB-Sticks und SD-Karten werden normalerweise direkt als Laufwerke eingehängt:

```
pi@Pi5Test:~ $ lsblk -f
NAME        FSTYPE FSVER LABEL  UUID
FSAVAIL FSUSE% MOUNTPOINTS
sda
├─sda1      vfat   FAT32 bootfs EF6E-C078
437.6M    14% /media/pi/bootfs
└─sda2      ext4   1.0   rootfs 4aa56689-dcb4-4759-90e6-179beae559ac
14.9G    42% /media/pi/rootfs
mmcblk0
├─mmcblk0p1 vfat   FAT32 bootfs EF6E-C078
435.5M    15% /media/pi/bootfs1
└─mmcblk0p2 ext4   1.0   rootfs 4aa56689-dcb4-4759-90e6-179beae559ac
50.2G     8% /media/pi/rootfs1
nvme0n1
├─nvme0n1p1 vfat   FAT32 bootfs EF6E-C078
438.1M    14% /boot/firmware
└─nvme0n1p2 ext4   1.0   rootfs 4aa56689-dcb4-4759-90e6-179beae559ac
209.5G     5% /
```

  - Hier ein Beispiel mit USB-Stick (sda), SD-Karte (mmcblk0) und NVMe (nvme0n1).
- Sollte ein Laufwerk nicht gemountet werden, muss es wie folgt gemountet werden (dies ist nur ein Beispiel!):
  - `sudo mkdir /mountdir`
  - `sudo mount /dev/nvme0n1p1 /mountdir/`
  - Anschließend kann das Medium mittels `sudo ~/chkspeed.sh /mountdir 1GB` getestet werden
  - Unmount mittels `sudo umount /dev/nvme0n1p1`

LibreOffice

- Das ist eine 500MB Installation ...
- `sudo apt install libreoffice`
- `libreoffice`

SMART Test

- `sudo apt install smartmontools nvme-cli -y`

nvme

- Übersicht anzeigen
  `sudo smartctl -a /dev/nvme0`

- Error Log ansehen
  `sudo smartctl -l error /dev/nvme0`

```
pi@Pi5Test:~ $ sudo smartctl -l error /dev/nvme0
smartctl 7.3 2022-02-28 r5338 [aarch64-linux-6.1.0-rpi7-rpi-2712]
(local build)
Copyright (C) 2002-22, Bruce Allen, Christian Franke,
www.smartmontools.org

=== START OF SMART DATA SECTION ===
Error Information (NVMe Log 0x01, 16 of 16 entries)
Num   ErrCount  SQId   CmdId  Status  PELoc          LBA  NSID    VS
  0       1358     0  0x1002  0x4213  0x028            0     0     -
  1       1357     0  0x1000  0x4213  0x028            0     0     -
  2       1356     0  0x3007  0x4005  0x028            0     0     -
```

- Gesamtstatus ansehen
  `sudo smartctl -H /dev/nvme0`

```
pi@Pi5Test:~ $ sudo  smartctl -H /dev/nvme0
smartctl 7.3 2022-02-28 r5338 [aarch64-linux-6.1.0-rpi7-rpi-2712]
(local build)
Copyright (C) 2002-22, Bruce Allen, Christian Franke,
www.smartmontools.org

=== START OF SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED
```

nvme

- Geräteliste ausgeben
  `sudo nvme list`

```
pi@Pi5Test:~ $ sudo nvme list
Node                  Generic               SN                   Model
Namespace Usage                      Format           FW Rev
--------------------- --------------------- -------------------- ------
---------------------------------- --------- --------------------------
---------------- --------
/dev/nvme0n1          /dev/ng0n1            SN195108904273
GIGABYTE GP-GSM2NE3256GNTD               1         256.06  GB / 256.06
GB    512   B +  0 B   EDFM00.2
```

- Übersicht anzeigen
  `sudo nvme –smart-log /dev/nvme0n1`

```
pi@Pi5Test:~ $ sudo nvme --smart-log /dev/nvme0n1
Smart Log for NVME device:nvme0n1 namespace-id:ffffffff
```

```
critical_warning                        : 0
temperature                             : 6°C (279 Kelvin)
available_spare                         : 100%
available_spare_threshold               : 5%
percentage_used                         : 1%
endurance group critical warning summary: 0
Data Units Read                         : 1,697,108 (868.92 GB)
Data Units Written                      : 3,901,500 (2.00 TB)
host_read_commands                      : 15,448,378
host_write_commands                     : 42,856,162
controller_busy_time                    : 1,107
power_cycles                            : 986
power_on_hours                          : 17,079
unsafe_shutdowns                        : 160
media_errors                            : 0
num_err_log_entries                     : 1,358
Warning Temperature Time                : 0
Critical Composite Temperature Time     : 0
Thermal Management T1 Trans Count       : 0
Thermal Management T2 Trans Count       : 0
Thermal Management T1 Total Time        : 0
Thermal Management T2 Total Time        : 0
```

- Error Log ansehen
  `sudo nvme error-log /dev/nvme0n1`

Sonstiges

Diese Punkte sind normal nicht Bestandteil vom YT Video!

Downgrade des Bootloaders

> **Warnung**
> Nur wichtig für Tests!

- Update eeprom
  `sudo rpi-eeprom-update -d -f /lib/firmware/raspberrypi/bootloader-2712/default/pieeprom-2023-12-06.bin`
- Eine Liste mit älteren Bootloader Ständen findet man unter
  `/lib/firmware/raspberrypi/bootloader-2712/default`
- rpi-boot-eeprom Releases
  https://github.com/raspberrypi/rpi-eeprom/releases

Links

- PCIe - PCI Express

https://www.elektronik-kompendium.de/sites/com/0904051.htm

- https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html
- https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html#enabling-pcie
- Raspberry Pi Bootoptions https://github.com/raspberrypi/firmware/blob/master/boot/overlays/README
- SMART Werte von NVMe SSDs auslesen \\https://www.thomas-krenn.com/de/wiki/SMART_Werte_von_NVMe_SSDs_auslesen