# Sovol SV08 Mainstream Klipper

- basierend auf: https://github.com/Rappetor/Sovol-SV08-Mainline
- ❗ Diese Anleitung behandelt nur das eMMC Update. Es geht hier nicht um die SD Karten Variante. Wer dazu nähere Infos braucht kann das im Github nachlesen!

## YouTube Video #137


## Vorbereitungen

- Es wird ein 'Makerbase MKS EMMC-ADAPTER V2 USB 3.0' USB Adapter benötigt um die eMMC zu lesen / schreiben.

- Es macht durchaus Sinn sich eine neue eMMC zu beschaffen. Die Originale ist erstens recht klein mit 8GB und einen “Rückweg” zu haben zum Urzustand ist nie verkehrt.
- Ein ST-Link V2 (Mini) wird benötigt, um die beiden Controller auf dem Mainboard im Druckerwith und am Kopf mit einer neuen Firmware zu beschreiben.
  Ich gehe da einen anderen Weg als die Original Anleitung. Man kann unter Linux auch relativ einfach die Firmware flashen und da braucht es nicht extra noch ein Windows Tool wie den

  STM32CubeProgrammer. Der ist nämlich öfter mal recht zickig in der Anwendung 🙂


# Sicherung

Es macht durchaus Sinn vorher die Konfig bzw. den ganzen Konfig Ordner vom original Klipper zu sichern. Es gibt zwei Möglichkeiten wie das klappen kann Online und Offline ...

## Drucker Online

Der Drucker ist noch im Netzwerk zu erreichen. Dann kann man mittels MobaXTerm eine SSH Verbindung aufbauen und die Daten über den MobaXTerm eingebauten SFTP Client abrufen. Einfach in den Ordner `printer_data/config` gehen und alle Dateien in einen Windows Ordner sichern. Alternativ geht das auch mit WinSCP oder SCP, macht es aber deutlich komplizierter vom Aufruf.

Der SSH Zugang zum Original Sovol Image ist User: **sovol** und Passwort: **sovol**

## Drucker Offline

Wenn der Drucker Offline ist - also nicht mehr im Netzwerk erreichbar oder wenn er einfach nicht mehr startet - dann braucht ihr den eMMS Adapter. Dort kommt dann die ausgebaute eMMC rein und ihr müsst euch dann eine **Linux Maschine** suchen. Unter Windows ist nämlich die Partition auf der die wichtigen Daten liegen, nicht sichtbar. In Linux müsst ihr ggf. noch die Partition mounten und könnt dann die Daten kopieren. Das alles setzt natürlich voraus, dass die eMMC auch noch heile ist.

**Linux**
Hier am Beispiel mit einem Raspberry Pi:

- eMMC mit Adapter in den Pi stecken
- in der Konsole mal ein `dmesg` absetzen. Das sollte am Ende irgendwas mit USB Storage und ggf. scsi bzw. sd liefern. In etwa so

```
[4100988.683264] usb 2-1: new SuperSpeed USB device number 2 using
xhci_hcd
[4100988.702054] usb 2-1: New USB device found, idVendor=05e3,
idProduct=0747, bcdDevice= 8.19
```

```
[4100988.702090] usb 2-1: New USB device strings: Mfr=3, Product=4,
SerialNumber=5
[4100988.702104] usb 2-1: Product: USB Storage
[4100988.702116] usb 2-1: Manufacturer: Generic
[4100988.702126] usb 2-1: SerialNumber: 000000000819
[4100988.710552] usb-storage 2-1:1.0: USB Mass Storage device detected
[4100988.711280] scsi host0: usb-storage 2-1:1.0
[4100989.716940] scsi 0:0:0:0: Direct-Access     Generic  STORAGE
DEVICE   0819 PQ: 0 ANSI: 6
[4100990.033126] sd 0:0:0:0: [sda] 15269888 512-byte logical blocks:
(7.82 GB/7.28 GiB)
[4100990.033883] sd 0:0:0:0: [sda] Write Protect is off
[4100990.033889] sd 0:0:0:0: [sda] Mode Sense: 87 00 00 00
[4100990.034602] sd 0:0:0:0: [sda] Write cache: disabled, read cache:
enabled, doesn't support DPO or FUA
[4100990.056061]  sda: sda1 sda2
[4100990.056338] sd 0:0:0:0: [sda] Attached SCSI removable disk
[4100990.067057] sd 0:0:0:0: Attached scsi generic sg0 type 0
```

- Wenn das gefunden wurde ist schon mal gut 🙂 Mit `lsblk` dann mal die Massenspeicher listen

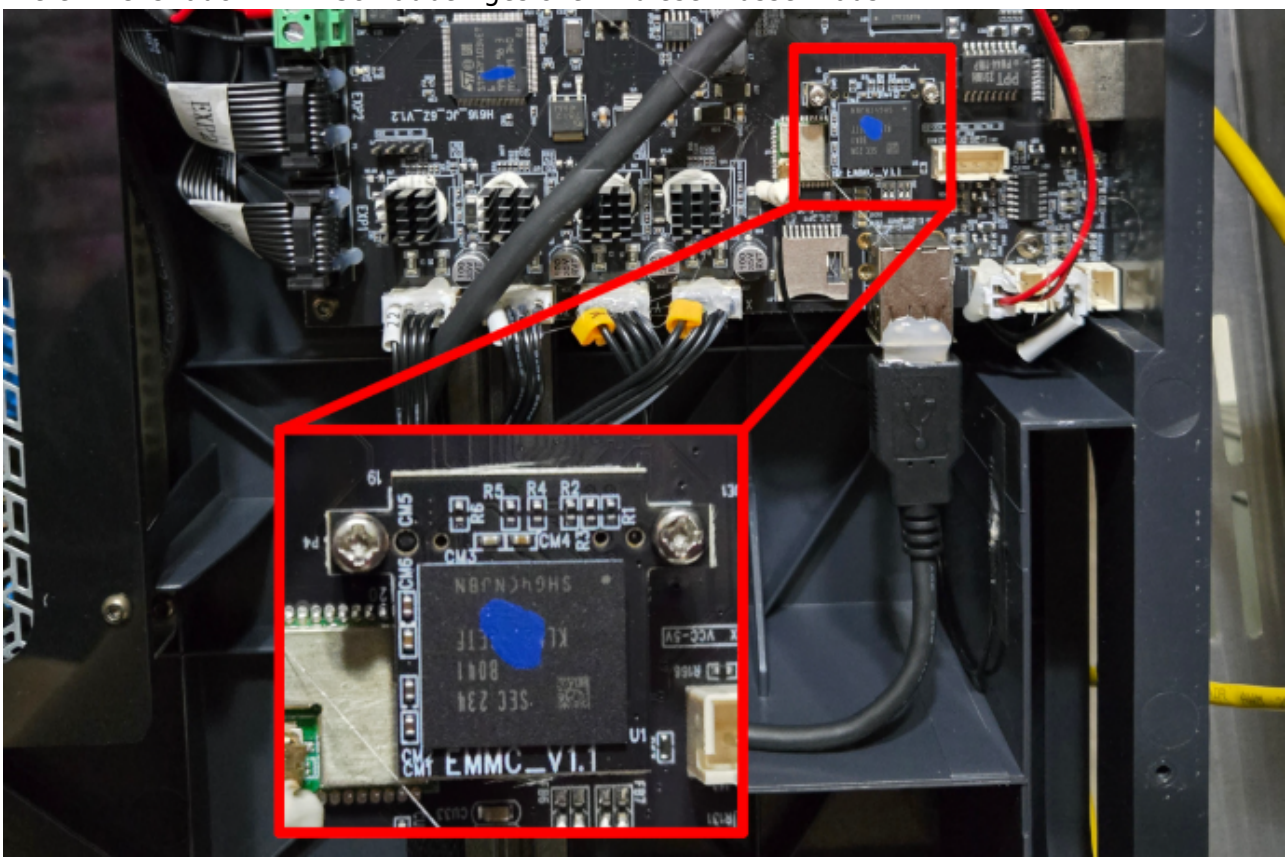
```
pi@testpi4:~ $ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    1  7.3G  0 disk
├─sda1        8:1    1  256M  0 part
└─sda2        8:2    1  6.9G  0 part
mmcblk0     179:0    0 58.9G  0 disk
├─mmcblk0p1 179:1    0  512M  0 part /boot/firmware
└─mmcblk0p2 179:2    0 58.4G  0 part /
```

- Wie man schon beim dmesg gesehen hat suchen wir irgendwas mitr sda ... In unserem Fall die größere Partition. Bei mir ist das `sda2` - das kann bei euch aber durchaus ander heißen!
- Jetzt müssen wir das Laufwerk / die Partition noch im Dateisystem einbinden, denn unter `MOUNTPOINTS` wird nichts gelistet.
  - `sudo mkdir /mnt/emmc`
  - `sudo mount /dev/sda2 /mnt/emmc/`
  - `cd /mnt/emmc`
    Hier könnt ihr nun mit `ls` den Inhalt listen.
  - `cd /mnt/emmc/home/sovol/printer_data/config` und hier findet ihr eure alte Konfig.
  - Ich kann nur empfehlen das mit MobaXTerm zu machen. Denn dort hat man für die SSH Verbindung zusätzlich eine SFTP Verbindung (linke Seite vom Fenster).

# eMMC ausbauen

- **Den Drucker stromlos machen!**
- Den Drucker auf die Seite legen und auf der Unterseite das Blech lösen (6 Schrauben).

- Die eMMC ist auch mit 2 Schrauben gesichert - diese müssen raus 🙂




- Danach kann man die eMMC einfach abziehen.

# Image auf eMMC schreiben

- Verwende - wie schon erwähnt - am besten eine neue eMMC mit mehr Speicher (64GB z.B.)
- Verwende als Image die V2.3.4 Version!
  Das Image bekommt man hier : https://github.com/bigtreetech/CB1/releases/tag/V2.3.4
  Direkter Link :
  https://github.com/bigtreetech/CB1/releases/download/V2.3.4/CB1_Debian11_minimal_kernel5.16_20240319.img.xz
- Die neue eMMC in den Adapter stecken (auf die Richtung achten!) und in den PC steckern.
- Raspberry Pi Imager geht sehr gut zum schreiben:
  - OS auswählen → Eigenes Image
  - Das geladene Image auswählen
  - SD Karte auswählen → hier dann die eMMC auswählen
  - OS Anpassungen → **NEIN**
    Die können wir bei dem Image nicht verwenden!
  - Weiter, warten, fertig 🙂

### Version 3.0.0

Es wird davon abgeraten das V3.0.0 Image zu verwenden. Scheinbar gibt es da noch kleinere

Probleme. Es sei aber erwähnt, dass es einen Pull Request im Github gibt, mit dem das V3 auch laufen soll:
https://github.com/Rappetor/Sovol-SV08-Mainline/pull/123
Ich lasse das mal so stehen. Wer das V3 Image testen will, kann das mit den Infos ja versuchen. Es ist aber nicht Bestandteil dieser Anleitung.

# System Konfig

Damit das System sauber läuft müssen ein paar Dateien kopiert / angepasst werden.
Diese Dateien kann man alle über den Windows Explorer erreichen! Sie liegen in dem Laufwerk was BOOT heißt.

## Device Tree

- Die Datei `sun50i-h616-sovol-emmc.dtb` von hier https://github.com/Rappetor/Sovol-SV08-Mainline/tree/main/files-used/dtb-files muss in den Ordner `/dtb/allwinner/`
- Die vorhandenen Dateien können dort liegen bleiben.

## BoardEnv.txt

- Alles unter dem `#------------#` lassen wir bestehen - **bitte hier nichts ändern** !
- Alles darüber ändern wir:

```
bootlogo=false
overlay_prefix=sun50i-h616
fdtfile=sun50i-h616-sovol-emmc
console=display
overlays=uart3 ws2812 spidev1_1
```

    - Ich bin übrigens der Meinung das hier die Github Anleitung falsch liegt. Beim CB1 Image müssen Ovberlays in einer Zeile stehen. Sonst greift nur das Letzte!
- Die fertige Datei sieht dann in etwa so aus:

```
bootlogo=false
overlay_prefix=sun50i-h616
fdtfile=sun50i-h616-sovol-emmc
console=display
overlays=uart3 ws2812 spidev1_1
#------------------------------------------------#
rootdev=UUID=795df55f-3e45-4625-a9cb-f6706b356274
rootfstype=ext4
```

## system.cfg

- Hier kann man seine WLAN Settings eintragen

download

```
#-----------------------------------------#
check_interval=5        # Cycle to detect whether wifi is
connected, time 5s

eth=eth0        # Ethernet card device number
wlan=wlan0      # Wireless NIC device number

hostname="SV08"

##############################################
# System time zone setting, default Time zone: Etc/UTC (UTC,
+0000)
# More settable time zones can be viewed by running the command:
timedatectl list-timezones
#TimeZone="Asia/Shanghai"

##############################################
# HDMI klipperScreen rotation
# ks_angle: Rotation angle
#       normal: 0; inverted: 180;
#       right: 90; left: 270;
ks_angle="inverted"

##############################################
# wifi name
WIFI_SSID="XXXXXXX"
# wifi password
WIFI_PASSWD="MySecret0815"

##############################################
# BTT-PAD7 (ON/OFF)
BTT_PAD7="OFF"
# touch vibration effects
TOUCH_VIBRATION="OFF"
# touch sound effects
TOUCH_SOUND="OFF"
# Automatic brightness adjustment
AUTO_BRIGHTNESS="OFF"
##############################################
```

- Wenn das WLAN Passwort Sonderzeichen enthält müssen diese ggf. entwertet werden. Aus **$** wird dann z.B. **\$**
- Für HDMI sollte man die Screen Rotation auf `inverted` stellen

# erstes Booten

- **Der Drucker ist immer noch stromlos!**
- Die neue eMMC wieder in den Drucker / ins Meinboard einsetzen. Auf die Richtung achten!
- Jetzt erstmal den Drucker auf die Beine stellen und Strom drauf geben.
- Einschalten ...
- Wenn ihr einen HDMI Bildschirm angeschlossen habt, dann seht ihr jetzt evtl. ein Boot Logo, das sagt aber noch nichts aus. Es ist nur ein Startbild - kein Systemstatus. Wir werden erst später Klipperscreen einrichten und ein Bild sehen!
- Wenn alles richtig eingetragen wurde sollte sich der Drucker mit eurem WLAN verbinden. Die IP müsst ihr im Router rausfinden oder mit MobaXTerm und den Tools (Network Scanner) suchen.
- Login mittels SSH sollte jetzt klappen und **User / Passwort ist biqu/biqu**

# Updates & Klipper

- System Updaten
  `sudo apt update && sudo apt upgrade`
- Ein paar sinnvolle Tools nachinstallieren
  `sudo apt install -y git git-lfs silversearcher-ag wavemon hexedit sudoku tcpdump iptraf mc htop dcfldd nano usbutils ranger tldr ncdu can-utils multitail fd-find lsof x11vnc terminator minicom cutecom joystick jstest-gtk i2c-tools speedtest-cli iotop fio ir-keytable curl inxi && mkdir -p ~/.local/share && tldr -u`
- kiauh besorgen
  `cd ~ && git clone https://github.com/dw-0/kiauh.git`
- kiauh starten
  `./kiauh/kiauh.sh`
  Wir verwenden die Version 6 und starten mit **Auswahl 3**
  - Install Klipper, Moonraker, Mainsail, Crowsnest, KlipperScreen
  - KIAUH main menu → Advanced → Extra Dependencies: → [Input Shaper]
- Moonraker nachinstallieren
  - `cd ~/`
  - `git clone https://github.com/mainsail-crew/moonraker-timelapse.git`
  - `cd ~/moonraker-timelapse`
  - `make install`
  - `nano ~/printer_data/config/moonraker.conf` und folgendes einfügen

```
[update_manager timelapse]
type: git_repo
primary_branch: main
path: ~/moonraker-timelapse
origin: https://github.com/mainsail-crew/moonraker-timelapse.git
managed_services: klipper moonraker

[timelapse]
##   Following basic configuration is default to most images and
don't need
##   to be changed in most scenarios. Only uncomment and change it
if your
```

```
##   Image differ from standart installations. In most common
scenarios
##   a User only need [timelapse] in their configuration.
output_path: ~/timelapse/                ##   Directory where the
generated video will be saved
frame_path: /tmp/timelapse/              ##   Directory where the
temporary frames are saved
ffmpeg_binary_path: /usr/bin/ffmpeg      ##   Directory where
ffmpeg is installed
```

    - im Mainsail Interface Timelapse einrichten
    - im Slicer später TIMELAPSE_TAKE_FRAME einbinden!
- Ab hier ist ein Standard Klipper System installiert. Ein `sudo reboot` könnte man ausführen wenn es nach crowsnest noch nicht gemacht wurde.

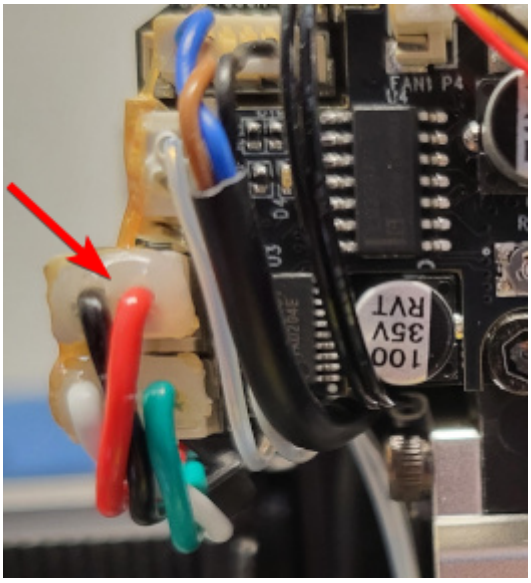
# Klipper Addons (Sovol)

- `cd ~ && git clone https://github.com/Rappetor/Sovol-SV08-Mainline && cd ~/Sovol-SV08-Mainline`
- RESTORE THE SOVOL ADDONS
    - `cd ~/Sovol-SV08-Mainline/files-used/sovol-addons/`
    - `cp * ~/klipper/klippy/extras/`
- GRAB BASE PRINTER CONFIGURATION
    - `cd ~/Sovol-SV08-Mainline/files-used/config/`
    - `cp -r * ~/printer_data/config`
- Jetzt wäre eine gute Idee auch den Start GCode anzupassen im Slicer:
  siehe → https://github.com/Rappetor/Sovol-SV08-Mainline?tab=readme-ov-file#step-5---configure-printerklipper--addons
- `sudo reboot`

# Serielle Ports

Die seriellen Ports kann man aus der alten Konfig raus friemeln, aber es geht auch anders … 🙂

- Am Kopfboard den Stromstecker abziehen




- Jetzt in der SSH Konsole folgendes eingeben
  `ls -lR /dev/ | grep -v '\->\s../tty' | grep -e 'tty[[:alpha:]]' -e serial`
  Ihr kriegt dann ne Menge Text, aber wir brauchen nur das hier (/dev/serial/by-id):

```
/dev/serial/by-id:
lrwxrwxrwx 1 root root 13 Jul 18 19:28 usb-
Klipper_stm32f103xe_30FFD90533475430175551551-if00 -> ../../ttyACM1
```

- Da das Kopfboard keinen Strom mehr hat kann es nur das normale **Mainboard** sein. Und bei mir ist der benötigte serielle Port dann:
  `/dev/serial/by-id/usb-Klipper_stm32f103xe_30FFD90533475430175551551-if00`
- Jetzt den Strom vom Kopfboard wieder dran stecken ...
- In der SHH Konsole das Kommando wiederholen. Und nun bekommen wir 2 Ports bei `/dev/serial/by-id` :

```
/dev/serial/by-id:
lrwxrwxrwx 1 root root 13 Jul 18 19:28 usb-
Klipper_stm32f103xe_30FFD90533475430175551551-if00 -> ../../ttyACM1
lrwxrwxrwx 1 root root 13 Jul 18 19:43 usb-
Klipper_stm32f103xe_51FF6E064849824936080767-if00 -> ../../ttyACM0
```

- Nuja, das **Kopfboard** ist jetzt einfach die ID die wir noch nicht haben 🙂 Bei mir also
  `/dev/serial/by-id/usb-Klipper_stm32f103xe_51FF6E064849824936080767-if00`
- Fertig - ohne großes Rumgesuche 😎
- Jetzt müsst ihr das nur noch in der printer.cfg eintragen.
  - Mainsail im Browser öffnen
  - auf Machine → und dort auf printer.cfg klicken
  - dort die Funde eintragen mcu = Mainboard und extra_mcu = Kopfboard

```
[mcu]
serial: /dev/serial/by-id/usb-
Klipper_stm32f103xe_30FFD90533475430175551551-if00
restart_method: command
```

```
[mcu extra_mcu]
serial: /dev/serial/by-id/usb-
Klipper_stm32f103xe_51FF6E064849824936080767-if00
restart_method: command
```

- oben auf “Save and Restart” und fertig 🙂

> Anmerkung : Jetzt sind auch endlich diese Krachlüfter mal leise und das Display geht auch wieder 😀

# stlink compilieren

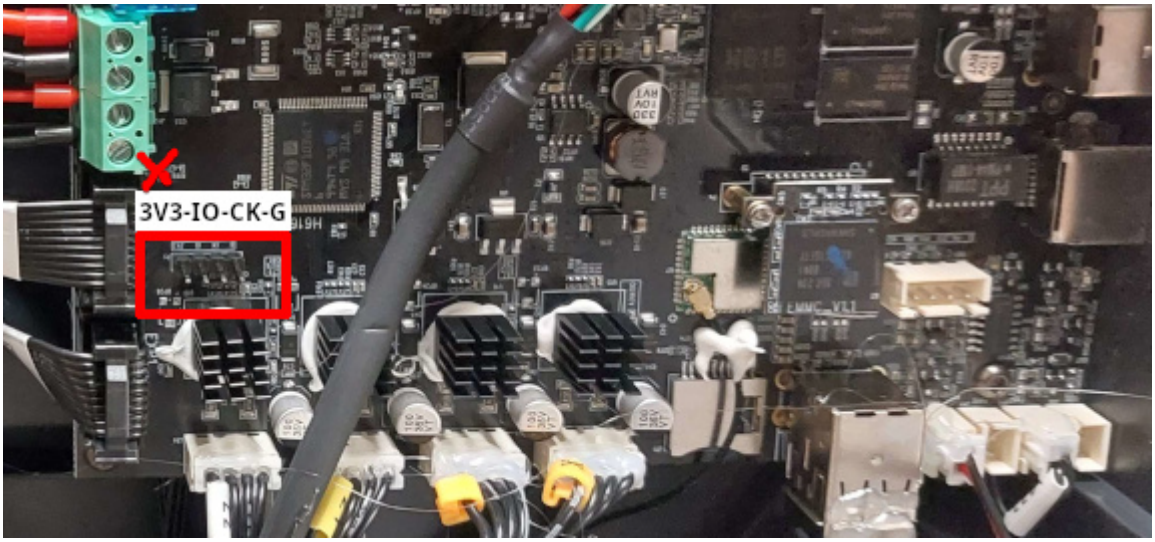
Damit wir die Firmware auf den beiden Boards im Drucker direkt mit dem eingebauten Linux flashen können brauchen wir die Toolsammlung stlink. Damit wir einen aktuellen Stand haben compilieren wir das schnell zusammen 🙂

- `sudo apt install gcc build-essential cmake libusb-1.0-0 libusb-1.0-0-dev libgtk-3-dev pandoc -y`
- `cd ~ && mkdir build -p && cd build`
- `git clone https://github.com/stlink-org/stlink.git && cd stlink && git checkout 4832a96bd19ca31f476ae67aa87359c1c09fb8b9`
- `make clean && make release`
- `sudo make install && sudo ldconfig`

# ST-Link V2 anschließen

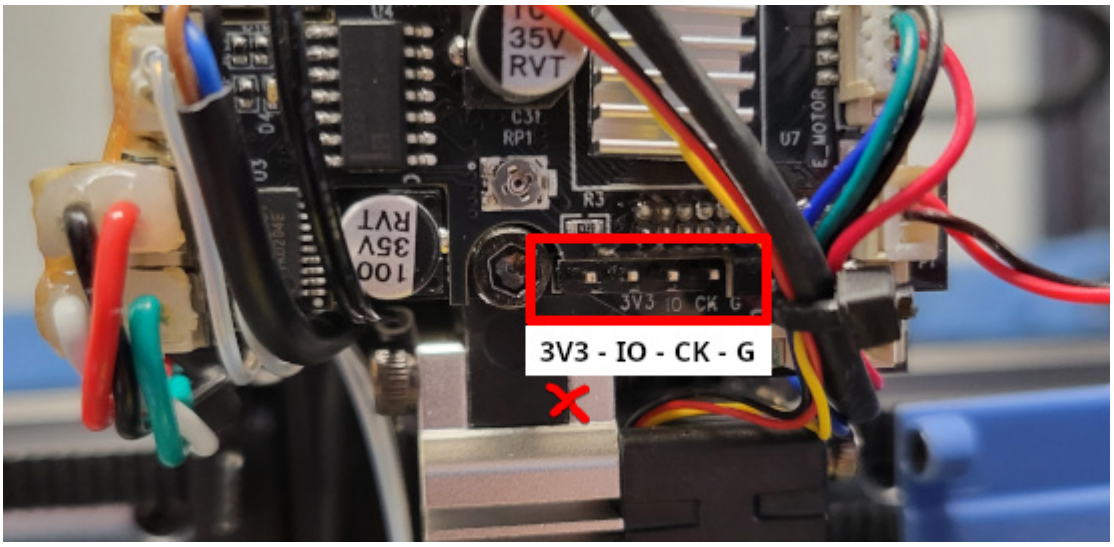
Entgegen der Original Anleitung wo alles unter Windows gemacht wird, mache ich das etwas anders - vermutlich auch einfachert 😉
Der Drucker bleibt beim Lesen / Schreiben des Flash an. Aus diesem Grund auch **keine 3,3V anschließen**!

## Mainboard




- Hier werden nur 3 Leitungen angeschloissen vom ST-Link
  - IO → DIO
  - CK → Clock
  - G → GND / Ground
  - **Der 3V3 Pin bleibt leer / nicht angeschlossen!**

### Kopfboard




- Auch hier nur 3 Leitungen
  - IO → DIO
  - CK → Clock
  - G → GND / Ground
  - **Der 3V3 Pin bleibt leer / nicht angeschlossen!**

## Backup Firmware

Man kann die alte Firmware auslesen. Ob das Sinn ergibt muss jeder für sich selber entscheiden 🙂
Hier ist jedenfalls meine Version wie ich die Original Firmware sichere.

#### Hinweis

Nicht wundern wenn während dem Flashen oder Auslesen der Drucker in den Shutdown Modus geht. Das ist normal weil das Board nicht mehr erreichbar ist. Nach einem Drucker Reboot ist alles wieder ok.

**Hinweis 2**
Zur Not liegen im Github auch noch Original Versionen :
https://github.com/Rappetor/Sovol-SV08-Mainline/tree/main/firmware-backups

## Mainboard

- Den ST-Link V2 Adapter anschließen wie vorher gezeigt mit nur 3 Pins.
- `st-info --probe` ausführen

```
biqu@SV08:~$ st-info --probe
Found 1 stlink programmers
  version:    V2J29S7
  serial:     4000500005000057365053 4E
  flash:      524288 (pagesize: 2048)
  sram:       65536
  chipid:     0x414
  dev-type:   F1xx_HD
```

- Wir müssen jetzt die Flash Größe berechnen die wir später beim Lesen angeben. Einfach mit dem Wert der hinter flash stehen.
  524288 / 1024 = **512k**
- Jetzt können wir auch schon den Flash lesen:
  `st-flash read firmware_main.bin 0x8000000 512k`
  Sollte das Lesen ein “Failed to connect to target” ausgeben, einfach nochmal das Kommando starten.
- Als Ergebnis sollte man jetzt eine 128k großes firmware_main.bin Datei haben
  `-rwx—— 1 biqu biqu 512K Jul 19 05:52 firmware_main.bin`

## Kopfboard

- Den ST-Link V2 Adapter anschließen wie vorher gezeigt mit nur 3 Pins.
- `st-info –probe` ausführen

```
biqu@SV08:~$ st-info --probe
Found 1 stlink programmers
  version:    V2J29S7
  serial:     4000500005000057365053 4E
  flash:      131072 (pagesize: 1024)
  sram:       20480
  chipid:     0x410
  dev-type:   STM32F1xx_MD
```

- Wir müssen jetzt die Flash Größe berechnen die wir später beim Lesen angeben. Einfach mit den Werten die hinter flash stehen.

131072 / 1024 = **128k**

- Jetzt können wir auch schon den Flash lesen:
  `st-flash read firmware_head.bin 0x8000000 128k`
  Sollte das Lesen ein “Failed to connect to target” ausgeben, einfach nochmal das Kommando starten.
- Als Ergebnis sollte man jetzt eine 128k großes firmware_head.bin Datei haben
  `-rwx—— 1 biqu biqu 128K Jul 19 05:09 firmware_head.bin`

# Klipper flashen

Auch das Flashen mache ich nicht unter Windows. Wie gehabt machen wir das direkt im Linux mit Boardmitteln 🙂

**Katapult** wird auf beiden Boards installiert. Das ist ein Bootloader, der es uns später ermöglicht, viel einfacher eine neue Klipper Version aufzuspielen.

## Katapult (beide Boards)

- `cd ~ && git clone https://github.com/Arksine/katapult`
- `pip3 install pyserial`
- `cd ~/katapult && make menuconfig`
- Die Katapult Konfiguration ist für beide Boards erstmal gleich :

```
Katapult Configuration v0.0.1-104-gfcb2f84
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F103)  --->
[ ] Only 10KiB of RAM (for rare stm32f103x6 variant) (NEW)
[ ] Disable SWD at startup (for GigaDevice stm32f103 clones) (NEW)
    Build Katapult deployment application (Do not build)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (USB (on PA11/PA12))  --->
    Application start offset (8KiB offset)  --->
    USB ids  --->
()  GPIO pins to set on bootloader entry
[ ] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[ ] Enable Status LED
```

- mit Q und Y beenden
- `make clean && make -j4`
- Jetzt jeweils das Mainboard und Kopfboard mittels ST-Link V2 anschließen (siehe oben - nur 3 Pins)
  - `st-flash erase`
  - `st-flash write ~/katapult/out/katapult.bin 0x8000000`
  - Das ganze dauert nur 3 Sekunden und sollte mit “Flash written and verified! jolly good!” enden.
  - Vorgang mit dem anderen Board wiederholen

- Das wars auch schon 🙂
- **Jetzt den Drucker mal runter fahren, stromlos machen, 5 Sekunden warten und neu starten!**

## Klipper Mainboard

- `cd ~/klipper`
- `make menuconfig`

```
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F103)  --->
[ ] Only 10KiB of RAM (for rare stm32f103x6 variant) (NEW)
[ ] Disable SWD at startup (for GigaDevice stm32f103 clones) (NEW)
    Bootloader offset (8KiB bootloader)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (USB (on PA11/PA12))  --->
    USB ids  --->
[*] Optimize stepper code for 'step on both edges' (NEW)
(PA1,PA3) GPIO pins to set at micro-controller startup
```

- mit Q und Y beenden
- `make clean && make -j4`
- Jetzt flashen wir (ohne den ST-Link) direkt über Katapult das Board. Dafür brauchen wir den seriellen Port den wir vorher für das Kopfboard ermittelt haben. Das war bei mir `/dev/serial/by-id/usb-Klipper_stm32f103xe_30FFD90533475430175551-if00`. Bei euch wird die ID aber anders aussehen.
- `sudo systemctl stop klipper.service`
- `make -j4 flash FLASH_DEVICE=/dev/serial/by-id/usb-katapult_stm32f103xe_30FFD90533475430175551-if00`
  ❗ Wir müssen hier nur `usb-Klipper` durch `usb-katapult` ersetzen!
- Am Ende sollte er “**CAN Flash Success**” melden
- `sudo systemctl start klipper.service`

## Klipper Kopfboard

- `cd ~/klipper`
- `make menuconfig`

```
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
    Micro-controller Architecture (STMicroelectronics STM32)  --->
    Processor model (STM32F103)  --->
[ ] Only 10KiB of RAM (for rare stm32f103x6 variant) (NEW)
[ ] Disable SWD at startup (for GigaDevice stm32f103 clones) (NEW)
```

```
    Bootloader offset (8KiB bootloader)  --->
    Clock Reference (8 MHz crystal)  --->
    Communication interface (USB (on PA11/PA12))  --->
    USB ids  --->
[*] Optimize stepper code for 'step on both edges' (NEW)
(PA6) GPIO pins to set at micro-controller startup
```

- mit Q und Y beenden
- `make clean && make -j4`
- Jetzt flashen wir (ohne den ST-Link) direkt über Katapult das Board. Dafür brauchen wir den seriellen Port den wir vorher für das Kopfboard ermittelt haben. Das war bei mir `/dev/serial/by-id/usb-Klipper_stm32f103xe_51FF6E064849824936080767-if00`. Bei euch wird die ID aber anders aussehen.
- `sudo systemctl stop klipper.service`
- `make -j4 flash FLASH_DEVICE=/dev/serial/by-id/usb-katapult_stm32f103xe_51FF6E064849824936080767-if00`
  Wir müssen hier nur `usb-Klipper` durch `usb-katapult` ersetzen!
- Am Ende sollte er "**CAN Flash Success**" melden
- `sudo systemctl start klipper.service`

## Reboot

Es ist keine dumme Idee, nach dem ganzen Flashen den Drucker mal durch zu booten
`sudo reboot`