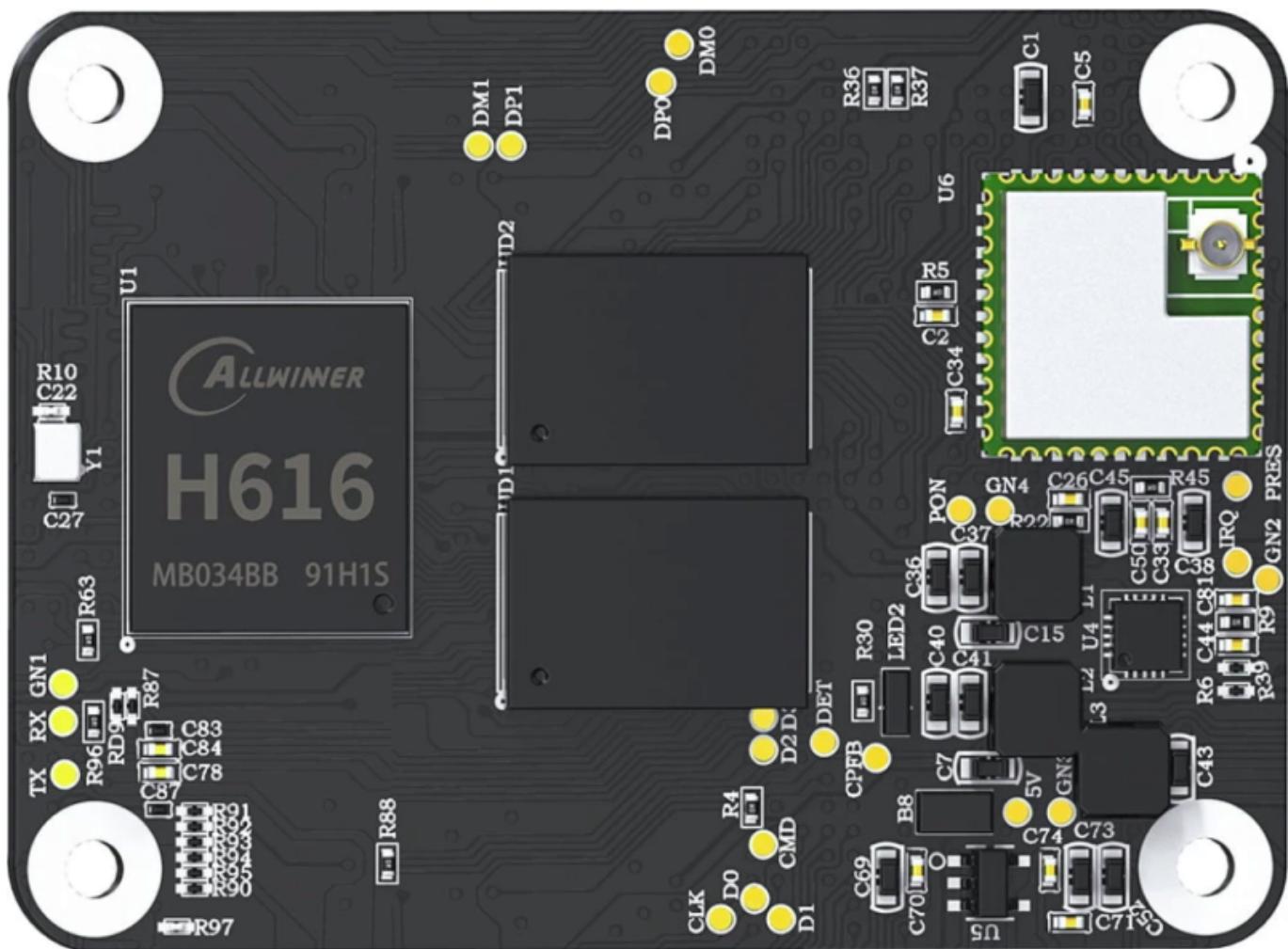


BTT CB1



YouTube Video #88



Sponsor

Vielen Dank an den Sponsor für dieses Video:

- **BigTreeTech** für das Manta M8P und CB1

Pros / Cons

- + Linux Kernel Quellen und Config File
- + Reicht für den normalen Klipper Betrieb
- Preisliche gleich teuer wie CM4 mit 1GB & WLAN (2GB nur 5€ mehr)
- - keine Anzeige für SD Aktivität
- - Keine Kamera (CSI) / Display (DSI) nutzbar
- - schlechte Doku bezüglich SPI / I2C
- - Kernel Updates nur über neues Image
- - kein PCIe
- - kein USB 3.0
- - SPI nur eingeschränkt nutzbar
- - kein I2C
- - externe Antenne
- - wenige Overlays verfügbar
- - wird schnell heiß im Betrieb
- - kein Bluetooth

Fazit

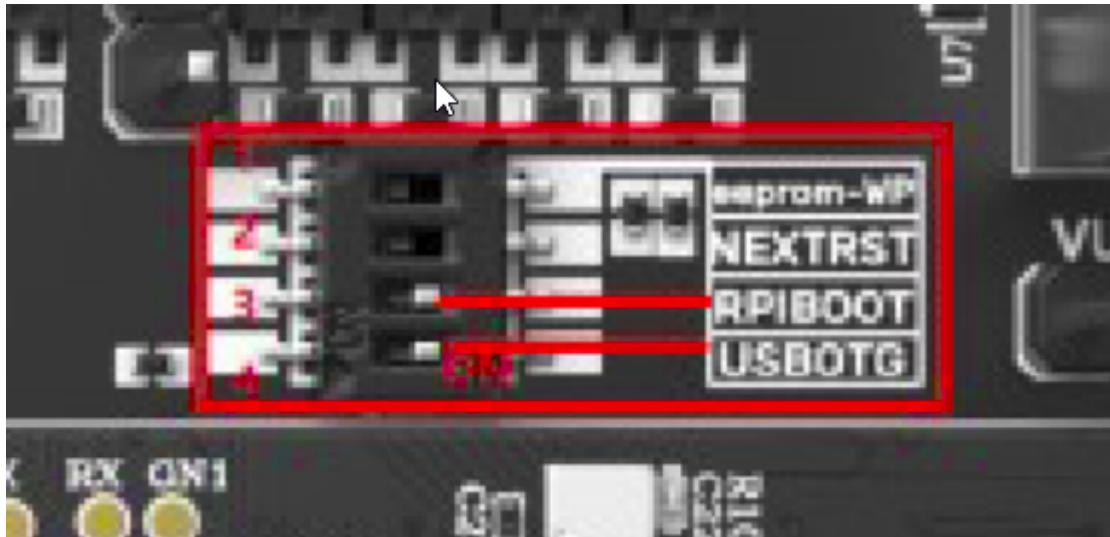
Aus meiner Sicht ist ein CB1 nur bedingt als "Bastelrechner" für Maker einsetzbar. In einem 3D Drucker erfüllt es aber durchaus seinen Zweck.

Adater Boards

- BTT Pi Adapter
<https://github.com/bigtreeetech/PI4B-Adapter/tree/master>
- Waveshare Adapter
<https://www.waveshare.com/cm4-io-base-b.htm>
- BTT Manta M4P / M5P / M8P, Manta E3EZ

Mäusekino

Für das CB1 / CM4 Modul mit eMMC Speicher müssen die Schalter RPIBOOT und USBOTG gesetzt sein zum Aufspielen des Systems.



Für den normalen Betrieb müssen diese Schalte ausgeschaltet sein!

Default User / Passwort

- User : **biqu**
- Passwort : **biqu**

Image Installation



Das BTT CB1 (wie auch der BTT Pi 1.2) laufen **nicht mit dem Raspberry Pi Image!**
Es muss ein Image von BTT genutzt werden das an den SBC angepasst ist.

- Image Download
<https://github.com/bigtreetech/CB1/releases>
Hier kann der aktuelle Stand vom BTT Image geladen werden. Grundsätzlich reicht für den Betrieb von Klipper das Minimal Image (Stand jetzt wäre das : CB1_Debian11_minimal_kernel5.16_20230712.img.xz). In dem größeren Image (CB1_Debian11_Klipper_kernel5.16_20230712.img.xz) ist u.A. noch ein Grafiktreiber für den SBC integriert. Klipper läuft mit beiden Images problemlos.
 - **Wichtig** : Bei dem kleinen image muss Klipper & Co mittels kiauh nachinstalliert werden!
- Image mit dem Raspberry Pi Imager auf eine SD Karte schreiben
<https://www.raspberrypi.com/software/>
 - OS Wählen
 - ganz unten auf “Eigenes Image”
 - Hier jetzt die geladene XZ Datei auswählen
 - SD Karte wählen
 - Schreiben (Einstellungen kann man nicht mit schreiben lassen. Die sind nicht kompatibel zu dem Image!)
- Wlan einrichten
Um das WLAN einzurichten muss auf der SD Karte eine Datei angepasst werden:
 - Auf dem Laufwerk “BOOT” die Datei system.cfg mit einem Texteditor öffnen
 - Wifi Settings anpassen:

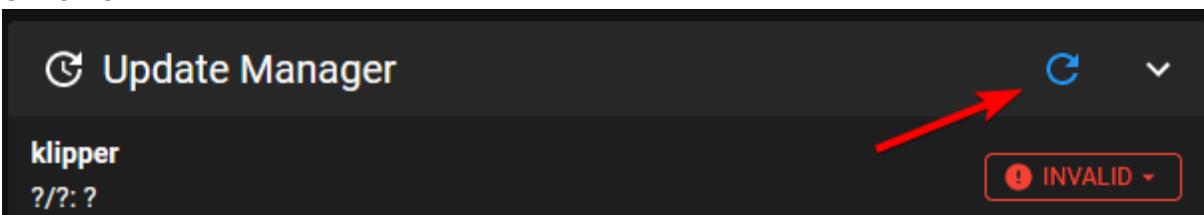
```
# wifi name
WIFI_SSID="WLAN_SSID"
# wifi password
WIFI_PASSWD="WLAN_PASSWORD"
```

- Hostname ggf. anpassen
Wird auch in der Datei `system.cfg` eingetragen:
`hostname="BTT-CB1"`
- Karte in den BTT Pi einsetzen
- ggf. Kühlkörper und WLAN Antenne nicht vergessen anzubringen



Updates

- per SSH einloggen
- Updates und ein paar Tools installieren
`sudo apt update && sudo apt upgrade -y && sudo apt install -y git silversearcher-ag wavemon hexedit sudoku tcpdump iptraf mc htop dcfldd nano usbutils ranger tldr ncdt can-utils multitail fd-find && mkdir -p ~/.local/share && tldr -u`
- überflüssige Dienste entfernen
`sudo apt autoremove alsamixer modem* cups* pulse* avahi* -y`
- IP ermitteln
`ip a`
- Mainsail Weboberfläche öffnen über die IP
 - Unter **Mashine** einmal die Repos neu laden um die roten "Invalid" Meldungen zu entfernen



- Jetzt alle Komponenten aktualisieren lassen mit **UPDATE ALL COMPONENTS**

Linux MCU einrichten

Wer das BTT CB1 mit einem ADXL345 Sensor für Input Shaper ausstatten möchte, sollte den Linux Prozess noch einrichten. Der Sorgt dafür, dass die GPIO Pins vom BTT CB1 aus Klipper Sicht genutzt werden können und ermöglicht eben den Betrieb von extra Sensoren wie dem ADXL345.

- auf dem BTT CB1 einloggen mittels SSH
- `cd ~/klipper/`
- `sudo cp ./scripts/klipper-mcu.service /etc/systemd/system/`
- `sudo systemctl enable klipper-mcu.service`
- `make menuconfig`
 - hier wählt man dann Microcontroller Architecture Linux process aus

```
[*] Enable extra low-level configuration options
    Micro-controller Architecture (Linux process) --->
    () GPIO pins to set at micro-controller startup
```

- mit Q beenden und mit Y speichern
- sudo service klipper stop
- make flash -j4
- sudo service klipper start

Minimale printer.cfg

Um Klipper vorübergehend in Betrieb zu nehmen, kann man folgende Mini Konfiguration verwenden:

[printer.cfg](#)

```
[include mainsail.cfg]
[mcu]
serial : /tmp/klipper_host_mcu

#[mcu Board]
#serial : /dev/serial/by-id/usb-
Klipper_stm32f407xx_2B0035001147393437303337-if00

[printer]
kinematics: none
max_velocity: 1000
max_accel: 1000
```



Mit dieser Konfig kann der Drucker natürlich rein gar nichts. Aber man wird erstmal alle Fehlermeldungen in MainSail los.

CAN

- CAN lässt sich am CB1 nur mit einem Buskoppler wie dem z.B. dem U2C oder einem Board im Bridge Mode nutzen.
- Das Aufsteckmodul ist hier nicht nutzbar, da der passende Anschluss fehlt.
- Beim Manta Board ist der USB direkt verdrahtet! Wer hier CAN nutzen möchte muss den Bridge Mode verwenden.

CAN Bus aktivieren

Damit der Adapter im Betriebssystem auch erkannt wird muss eine Netzwerk Interface Konfiguration (`/etc/network/interfaces.d/can0`) angelegt werden. Es ist möglich das die Datei schon im BTT Image eingepflegt ist ...

- sudo nano /etc/network/interfaces.d/can0
- folgenden Inhalt in der Datei einfügen :

```
allow-hotplug can0
iface can0 can static
    bitrate 1000000
    up ifconfig $IFACE txqueuelen 1024
```

- Editor mit **STRG + X** → **dann Y** → **dann Enter** verlassen
- System neu starten mittels
sudo reboot
- Nach dem Reboot sollte der Befehl ip a ein CAN Interface listen :

```
can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc pfifo_fast state UP group
default qlen 1024
    link/can
```

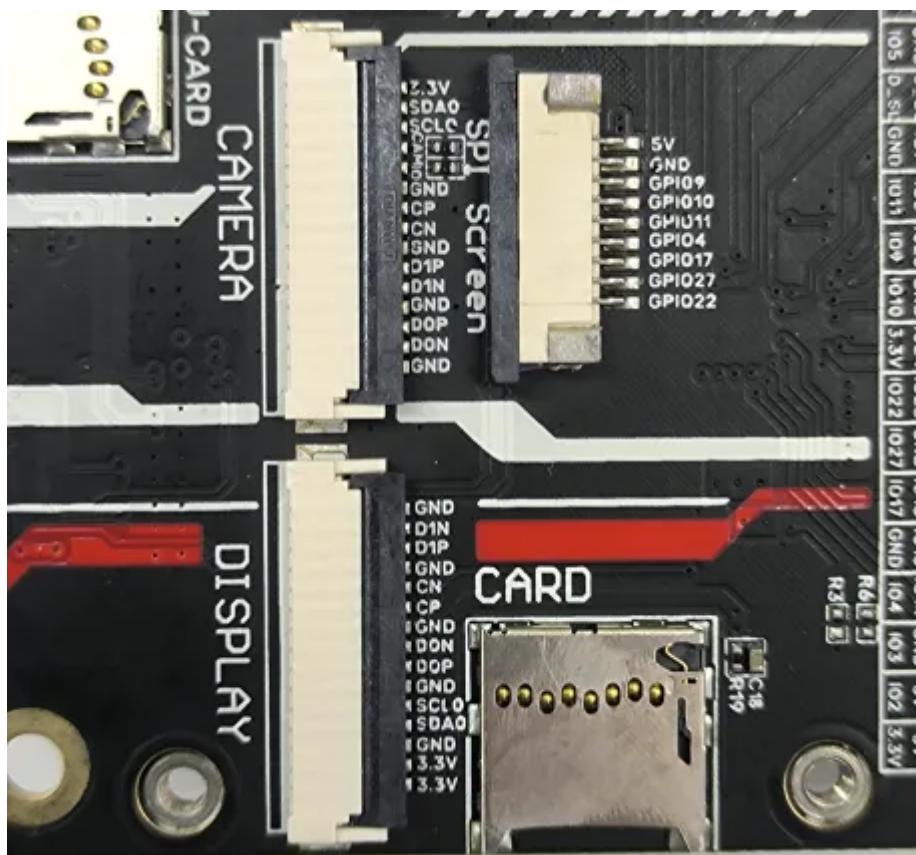
- Wichtig dabei ist **state UP**. Wird hier ein DOWN angegeben, ist das Interface nicht betriebsbereit!

Can Bus Test

- sudo systemctl stop klipper.service
- ~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0
- sudo systemctl start klipper.service

kein DSI / CSI

Die meisten Board bieten die Anschlüsse für CSI (Kamera) und DSI (Monitor). Wie hier z.B. beim Manta Board:



Diese Anschlüsse funktionieren aber **NICHT mit dem CB1!**

HDMI Display

Am HDMI Anschluss sollte nach dem Start sofort eine Ausgabe erfolgen. Anpassungen kann man an zwei Stellen vornehmen:

- sudo nano /boot/BoardEnv.txt

```
## Specify HDMI output resolution (eg. extraargs=video=HDMI-A-1:800x480-24@60)
#extraargs=video=HDMI-A-1:1024x600-24@60
```

- Hier in der zweiten Zeile die # entfernen und die Auflösung anpassen
 - `sudo nano /boot/system.cfg`

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

- Hiermit kann die grafische Ausgabe auf dem HDMI Port gedreht werden.



Die Textkonsole bleibt davon leider unberührt:::

Manta Boards

Auf einem M8P Manta Board lässt sich der HDMI Ausgang nur betreiben, wenn das Board extra mit USB-C versorgt wird. Zusätzlich muss der 5V USB Jumper gesetzt sein!

Nachtrag 09.02.2024

Einige User berichten das der HDMI Ausgang auch mit dem CB1 und 24V Versorgung funktioniert. Hier muss sich BTT mal zu Wort melden, warum es u.A. bei mir nicht klappt.

ADXL345 (SPI)

Hinweis 1

Die Default CS Pins können nicht in Klipper genutzt werden. Da kommt ein Fehler "Unable to open out GPIO chip line"

Hinweis 2

Für Klipper einen freien GPIO verwenden und den als CS Pin nutzen (Beispiel gpio74)

CB1 Pin	Function	CB1 GPIO
PH6	SPI1 CLK	GPIO230
PH7	SPI1 MOSI	GPIO231
PH8	SPI1 MISO	GPIO232
PC10	CS	GPIO74



- Aktivieren über sudo nano /boot/BoardEnv.txt
 - bei den Overlays spidev1_1 hinzufügen → Bsp: overlays=spidev1_1 ir
 - **ACHTUNG** !
- Die [Linux MCU](#) muss laufen!
- Klipper Konfig erweitern

```
[adxl345]
cs_pin: BTTP1:gpio74
spi_bus: spidev1.1

[resonance_tester]
```

```
accel_chip: adxl345
probe_points:
  60, 60, 20 # an example
```



- Evtl. den Namen BTTP1 anpassen an den Namen der eigenen Linux MCU!!

Tests

- Wenn ein SPI Bus über das Overlay aktiviert wurde sollte der im Device Verzeichnis /dev auftauchen 'nach einem Reboot'

```
biqu@BTT-CB1:~$ ls /dev
autofs          cuse      gpiochip1  kmsg    loop6       net      rtc0      sunxi_soc_info  tty15
block           disk      hidraw0   lirc0   loop7       null    serial    tty        tty16
btrfs-control   dri       hugepages log     loop-control ppp     shm      tty0      tty17
bus              encryptfs hwrng    loop0   mapper    psaux   snd      tty1     tty18
cec0             fb0      i2c-0    loop1   mem      ptmx    spidev1.1  tty10    tty19
char             fd       i2c-1    loop2   mmcblk1   pts     stderr   tty11    tty20
console          full     i2c-2    loop3   mmcblk1p1 random  stdin    tty12    tty21
core             fuse     initctl  loop4   mmcblk1p2 rfkill  stdout   tty13    tty22
cpu_dma_latency  gpiochip0 input    loop5   mqueue   rtc    sunxi-reg  tty14
biqu@BTT-CB1:~$
```

- Hier wurde folgendes in der /boot/BoardEnv.txt eingetragen :
`overlays=spidev1_1 ir`
- Wenn der Bus in /dev vorhanden ist kann man in der MainSail Konsole den Input Shaper Sensor abfragen mit ACCELEROMETER_QUERY

- Bei einer erfolgreichen Abfrage liefert der Sensor Werte

```
11:23 accelerometer values (x, y, z): 0.000000, -1406.391290, -7890.783629
```

```
11:23 ACCELEROMETER_QUERY
```

- Wenn der Sensor nicht antwortet kommt eine Fehlermeldung

```
11:24 Invalid adxl345 id (got 0 vs e5).
```

```
11:24 Invalid adxl345 id (got 0 vs e5).
```

```
This is generally indicative of connection problems
(e.g. faulty wiring) or a faulty adxl345 chip.
```

In dem Fall Konfig und Verkabelung prüfen!

GPIO nutzen (Linux MCU)

Die GPIO Pins vom BTT Pi können von Klipper angesteuert werden. Man muss nur mit der Pin Bezeichnung aufpassen ...

Pin Berechnung

Um die GPIO Pins in Klipper nutzen zu können muss ggf. umgerechnet werden. Die Pins sind im Normalfall nach dem Schema **PxNN** benannt. **x** kann dabei von A..G gehen und **NN** ist eine Zahl. Das nutzt aber in Klipper nichts, weil dort die richtigen GPIO Nummern angegeben werden müssen. Dafür gibt es folgende Rechnung:

- Der Buchstabe wird in eine Zahl gewandelt. A = 0, B = 1 ... G = 6

- Die Zahl * 32
- Anschließend noch die NN Nummer addieren
- Beispiel PC15
 - C = 2, NN = 15
 - $2*32 + 15 = 79$
 - PC15 → gpio79

In der Klipper Konfig trägt man an der Stelle für den Pin dann gpio79 und **nicht** PC15 ein.

Beispiel OutPin

Einen einfachen Ausgang zum Schalten von was auch immer kann man so realisieren:

```
[output_pin OutPin]
pin    : gpio79
pwm    : false
value  : 0
```

ACHTUNG

Hier darf nicht einfach irgendeine Last angeschlossen werden. Die GPIO Pins können nämlich kaum Strom abgeben. Als im Zweifel mit einem Mosfet arbeiten oder jemanden fragen der sich damit

auskennt 

Ergebnis in MainSail



Links

- <https://github.com/bigtreetech/CB1#40-pin-gpio>
- <https://github.com/So6Rallye/BTT-Pi/blob/master/BIGTREETECH%20Pi%20V1.2%20-%20Board%20Fan%20Pin%20Configuration>

Temp

CB1 in Klipper

```
[temperature_sensor CB1]
sensor_type: temperature_host
```

Lüfter in Klipper

Dieses Beispiel steuert einen Lüfter über den CB1 eingebauten Temperatursensor:

```
[temperature_fan Case_fans]
pin: PD4
sensor_type: temperature_host
off_below: 0.4
min_temp: 10
max_temp: 90
target_temp: 55
control: pid
pid_Kp: 2
pid_Ki: 4
pid_Kd: 0.1
```

Konsole

- lm-sensors Paket installieren um die Temperaturen der Kerne in der Konsole auslesen zu können:
sudo apt install lm-sensors
- Abfrage über
sensors

grafische Auswertung

Hinweis

Diese Anleitung klappt nur wenn eine grafische Umgebung auf dem CB1 installiert ist. Bei dem "großen" Image von BTT ist das der Fall.

Weiterhin braucht man MobaXTerm um die Ausgaben über den X Server zu bekommen.

- sudo apt install python3-virtualenv python3-tk lm-sensors -y
- virtualenv grapher
- cd grapher/
- source bin/activate
- pip3 install matplotlib
- nano grapher.py

```
import tkinter
import matplotlib
import matplotlib.pyplot as plt
from collections import deque
from datetime import datetime
import subprocess
import time

def run_linux_program():
    command = 'sensors |grep crit |cut -d "+" -f2|cut -d "C" -f1'
    result = subprocess.run(command, capture_output=True, text=True,
shell=True)
    print(result.stdout.strip()[:4])
    return float(result.stdout.strip()[:4])
```

```

# Initialisierung des Diagramms
matplotlib.use('TkAgg')
print(f"Interactive mode: {matplotlib.is_interactive()}")
print(f"matplotlib backend: {matplotlib.rcParams['backend']}")
plt.ion()
fig, ax = plt.subplots()
values = deque(maxlen=100)

try:
    while True:
        values.append(run_linux_program())

        ax.clear()
        ax.plot(values)
        ax.set_title('CB1 Temp Monitor')
        ax.set_xlabel('Zeit')
        ax.set_ylabel('Temp')

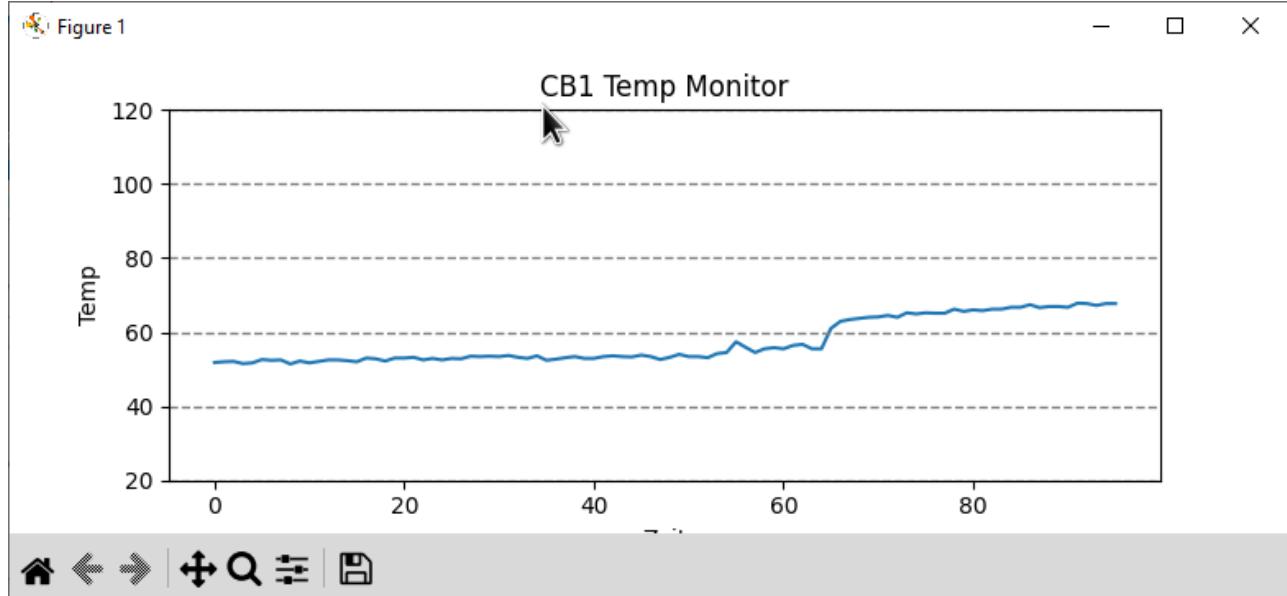
        # Festlegen von 5 Ticks auf der Y-Achse
        y_ticks = [20, 40, 60, 80, 100, 120]
        ax.set_yticks(y_ticks)

        for y_tick in y_ticks:
            ax.axhline(y=y_tick, linestyle='dashed', color='gray',
linewidth=1)

        plt.draw()
        plt.pause(1.0)
except KeyboardInterrupt:
    pass
finally:
    plt.ioff()
    plt.show()

```

- Starten kann man den Code mittels `python3 grapher.py`



Abbrechen mit STRG + C

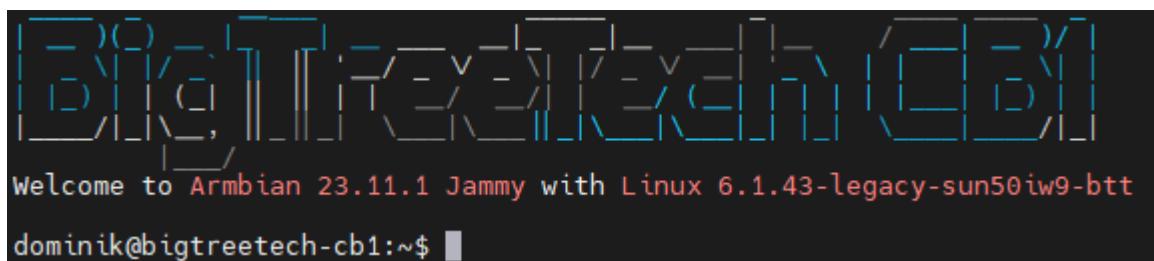
Stresstest durchführen

- sudo apt install stress
- stress -c 4 -m 4

Klipperscreen

- Lief am Adapter Board und Manta (nur mit USB-C) sofort mit dem “großen” Image

Armbian Images



```
[REDACTED] Welcome to Armbian 23.11.1 Jammy with Linux 6.1.43-legacy-sun50iw9-btt dominik@bigtreetech-cb1:~$
```

Gefühlt ist das Image etwas “zäh”.

- <https://www.armbian.com/bigtreechip-cb1/>
- https://docs.armbian.com/User-Guide_Getting-Started/

Links

- CB1 Github Repo
<https://github.com/bigtreechip/CB1>
- Vergleich CM4 / CB1
<https://www.learningtopi.com/sbc/cb1/bigtreechip-cb1/>
- Armbian Image
<https://www.armbian.com/bigtreechip-cb1/>

From:
<https://drklipper.de/> - Dr. Klipper Wiki



Permanent link:
https://drklipper.de/doku.php?id=klipper_faq:sbcs:btt_cb1

Last update: **2024/02/09 07:33**